Universidade de São Paulo
Instituto de Física e Química de São Carlos

**Estudo e Implementação de
"Software" Dedicado para um
Sistema de Visualização de Imagens**

Agma Juci Machado Traina

**Tese apresentada ao
Instituto de Física e Química de São Carlos,
para a obtenção do título de
Doutora em Ciências (Física Aplicada).**

Orientador: Prof. Dr. Jan Frans Willem Slaets

Departamento de Física e Ciência dos Materiais
São Carlos - 1991

UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE FÍSICA E QUÍMICA DE SÃO CARLOS

MEMBROS DA COMISSÃO JULGADORA DA TESE DE DOUTORADO DE AGMA JUCI MACHADO TRAINA APRESENTADA AO INSTITUTO DE FÍSICA E QUÍMICA DE SÃO CARLOS, DA UNIVERSIDADE DE SÃO PAULO, EM 08.7.1991

COMISSÃO JULGADORA:

Prof. Dr. Jan Frans Willem Slaets

Prof. Dr. Alberto Tannus

Prof. Dr. Carlos Antonio Ruggiero

Prof. Dr. Clésio Luiz Tozzi

Prof. Dr. José Hiroki Saito

Dedico este trabalho:

Aos meus filhinhos, Tatiana e Carlos Augusto, pela compreensão, apesar de tão pequenos, de meus períodos de trabalho extra;

Ao Caetano, pelo companheirismo, incentivo e carinho sempre presentes em nossa vida;

À minha mãe, Maria Conceição, pelo imenso amor e dedicação por todos nós.

Agradecimentos

Ao Prof. Dr. Jan Frans Willem Slaets, pela influência tão grande na minha formação profissional, por sempre ter-me oferecido ótimas oportunidades de aprendizado, e sempre estar pronto a solucionar minhas dificuldades, possibilitando que este trabalho fosse levado a termo.

Ao Prof. Dr. Caetano Traina Jr., meu marido, pelo auxílio na utilização do sistema de janelas do GeO, desenvolvido por seu grupo do SCE-ICMSC, e pelos discussões e auxílios em todas as horas.

À desenhista do Grupo de Instrumentação Eletrônica do IFQSC, Maria Benedita (Benê), pela prontidão e dedicação com que realizou os desenhos contidos nesta tese, trabalhando inclusive em suas férias.

À Lia Mara C. Chinaglia, pela sua boa vontade, estando continuamente pronta a nos prestar auxílio e pelo excelente trabalho de secretaria.

Aos amigos Mônica e Gonzalo Travieso, pelas conversas e discussões tão proveitosas.

À Maria Stela Paiva pelo auxílio com o TMS.

Ao Lúrio O. B. Almeida, técnico do LIE, pelo imenso auxílio com a arquitetura dedicada, pelas fotos que ilustram o trabalho e por tudo que precisei para desenvolver o meu trabalho.

Aos técnicos do LIE, Ailton, Marcos e Ivanilda, por estarem sempre dispostos a me ajudar em tudo.

À Renata P. Fortes, pelas discussões sobre o "software".

À secretária Maria Helena, do grupo de Cristalografia, por ajudar-me com os assuntos de secretaria.

Ao Grupo de Ressonância Magnética do IFQSC, pelo apoio dado ao trabalho e pelo fornecimento das imagens para validação do trabalho.

A minha mãe, pelo imenso auxílio no cuidado dos meus filhos, para que eu tivesse a tranquilidade necessária para desenvolver este trabalho.

Ao Departamento de Computação e Estatística, do Instituto de Ciências Matemáticas de São Carlos, pela liberação a mim concedida, para que pudesse realizar este trabalho.

A Fundação do Banco do Brasil, pelo financiamento outorgado ao projeto.

A todas as demais pessoas que direta, ou indiretamente auxiliaram para que este trabalho pudesse ser realizado.

Índice

Lista de Figuras	i
Lista de Tabelas	iv
Resumo	v
Abstract	vi
Capítulo I - Introdução	1
1.1 - Considerações Iniciais	1
1.2 - Objetivo do Trabalho.	3
1.3 - Contribuições deste Trabalho.	3
1.4 - Descrição do Trabalho.	4
1.4 - Apresentação do Trabalho.	6
Capítulo II - Introdução sobre Aspectos de Processamento Digital de Imagens Utilizados no Sistema Desenvolvido	7
2.1 - Introdução.	7
2.2 - Considerações Sobre Imagens.	8
2.3 - Tipos de Manipulação de Imagens.	10
2.4 - Transformações Radiométricas.	11
2.4.1 - Operações Pontuais sobre Imagens.	12
2.4.1.1 - Histogramas de Intensidades.	12
2.4.2 - Operações Locais Sobre a Imagem.	15
2.4.2.1 - Operação de Convolução.	16
2.4.2.2 - Exemplos de Transformações não Lineares.	18
2.5 - Transformações Geométricas.	21
2.6 - Operações Algébricas.	23
2.7 - Tabelas "Look-up".	24
2.8 - Conclusões.	25
Capítulo III - Sistema de Processamento de Imagens para Tomografia Computadorizada por Ressonância Magnética	26
3.1 - Introdução.	26
3.2 - Plataforma Computacional Utilizada.	26
3.3 - Coleta dos Dados Usando Ressonância Magnética.	27
3.4 - Processamento dos Dados.	28
3.5 - Manipulações Efetuadas pelo Sistema.	29

3.6 - Conclusões.	33
------------------------	----

Capítulo IV - Aspectos Relevantes da Arquitetura Dedicada Baseada no

GSP TMS34010 para o Ambiente Gráfico Desenvolvido	39
4.1 - Introdução.	39
4.2 - Arquitetura do GSP.	39
4.2.1 - Organização da Memória.	40
4.2.2 - Tipos de Dados Suportados em Hardware.	42
4.2.2.1 - "Fields"	42
4.2.2.2 - Bytes	43
4.2.2.3 - Pixels	43
4.2.2.3.1 - Mapeamento dos Pixels na Tela	44
4.2.2.3.2 - Métodos de Endereçamento de Pixels	44
4.2.2.4 - Matriz Bidimensional de pixels	45
4.2.3 - Registradores do GSP.	46
4.2.3.1 - Registradores de Propósito Geral.	47
4.2.3.2 - Registrador de "Status".	49
4.2.3.3 - Registradores de Entrada/Saída (E/S).	50
4.2.3.4 - Registradores de Interface com o Hospedeiro.	51
4.2.3.5 - Registradores Dedicados ao Controle da Memória Local.	53
4.3 - Instruções Gráficas	54
4.3.1 - Instrução FILL	54
4.3.2 - Instrução PIXBLT	54
4.3.3 - Instrução PIXT	55
4.3.4 - Instrução DRAV	55
4.3.5 - Instrução LINE	56
4.3.6 - Instrução CPW	56
4.4 - Operações Gráficas.	56
4.4.1 - Operação de Expansão de Cor.	57
4.4.2 - Operações com Janelas.	57
4.4.3 - Operações de Processamento de pixels.	58
4.4.4 - Máscara de Planos.	60
4.5 - Ferramentas para Desenvolvimento de Software.	61
4.5.1 - "Assembler"	61
4.5.2 - "Archiver"	61
4.5.3 - "Linker"	62

4.5.6 - Utilitário para Conversão de Código	62
4.5.7 - Simulador	62
4.5.8 - Programa para Depuração de Código - SDB340.	63
4.5.9 - Compilador para a Linguagem "C".	63
4.5.10 - Carregador SDBL.	64
4.6 - Especificações da Arquitetura Desenvolvida.	64

Capítulo V - Ferramentas de Apoio ao Desenvolvimento de Interfaces com o Usuário

o Usuário	66
5.1 - Introdução.	66
5.2 - Classificação das Ferramentas para a Construção de Interfaces com o Usuário.	67
5.3 - Ferramentas de Auxílio à Construção da Comunicação Homem/Máquina.	69
5.3.1 - Sistemas de Gerenciamento de Janelas.	69
5.2.2 - Gerenciadores de "menus".	74
5.2.3 - Padrões de Apresentação.	76
5.3 - Conclusões.	77

Capítulo VI - Ambiente Gráfico para Visualização de Imagens

Tomográficas	78
6.1 - Introdução.	78
6.2 - O Ambiente Gráfico Desenvolvido.	79
6.3 - Elementos que Compõem o Ambiente Gráfico.	80
6.3.1 - Gerenciador de Janelas.	81
6.3.2 - Gerenciador de "menus".	85
6.3.3 - Subsistema de Comunicação com o Hospedeiro.	87
6.3.4 - Subsistema de Transferência de Imagens Tomográficas de RM.	96
6.4 - Descrição das Estruturas de Dados Utilizadas.	96
6.4.1 - Estrutura de Dados para o Gerenciamento de Janelas - LIE_janela.	96
6.4.2 - Estrutura de Dados LIE_tipo_janela.	99
6.4.3 - Estrutura de Dados LIE_fonte.	100
6.4.4 - Estrutura de Dados lie_prof_janela.	101
6.4.5 - Estrutura de Dados LIE_comunica.	101
6.4.6 - Estrutura de Dados ArqSpec.	103

6.4.7 - Estruturas de Dados pdmenu e pdmdef.	104
6.5 - Um Exemplo.	105
6.6 - Conclusões.	111
 Capítulo VII - Conclusões Gerais e Linhas de Futuras Pesquisas	113
7.1 - Conclusões.	113
7.2 - Linhas de Futuras Pesquisas	116
7.2.1 - Extensão do Ambiente Gráfico.	116
7.2.2 - Processamento de Imagens em Arquitetura Paralela.	116
7.3.3 - Utilização de Sistemas Especialistas como Auxílio ao Diagnóstico.	117
7.2.4 - Visualização de Imagens em Três Dimensões (3D).	118
 Bibliografia	120
 Apêndice A - Listagem do Sistema de Processamento de Imagens	A-1
 Apêndice B - Manual de Referência do Ambiente Gráfico	B-1
 Apêndice C - Manual de Referência do Subsistema de Comunicação com o Hospedeiro	C-1
 Apêndice D - Arquivos de "Headers" para o Ambiente Gráfico	D-1
 Apêndice E - Listagem das Rotinas que Compõem o Ambiente Gráfico	E-1

Lista de Figuras

Figura 1 - Ilustração de um sistema gráfico.	2
Figura 2 - Etapas do processamento interativo de Imagens.	8
Figura 3 - Representação de um pixel P com 8-bits (8 planos) de profundidade.	10
Figura 4 - Imagem de um corte transversal de cabeça humana e seu histograma associado (obtidos do sistema de processamento e visualização de imagens desenvolvido no LIE).	13
Figura 5 - Efeito de uma transformação dos níveis de cinza efetuado sobre o histograma.	14
Figura 6 - Manipulação de janelas de intensidade da imagem através do histograma.	15
Figura 7 - Sinais "ruído" (a) e degrau" (b).	18
Figura 8 - Efeitos de filtros lineares ao "ruído" e à "borda": (a) efeitos de atenuação; (b) borramento da borda; (c) ampliação do ruído; (d) realce da borda.	19
Figura 9 - Efeito do filtro da mediana sobre o "ruído" e o "degrau".	20
Figura 10 - Uma transformação geométrica de 90 graus.	22
Figura 11 - Organização de uma LUT, para um sistema de pixels de 8 bits.	25
Figura 12 - Visualização de uma imagem tomográfica pelo sistema. Corte	34
Figura 13 - Imagem anterior com manipulação da tabela de cores.	34
Figura 14 - Imagem e seu histograma associado, preparando a manipulação do histograma.	35
Figura 15 - Manipulação da imagem através do seu histograma.	35
Figura 16 - Exemplos de operações sobre as imagens.	
(a) Imagem com filtragem espacial tipo "passa-baixas"	36
(b) Imagem com filtragem espacial tipo "passa-altas"	36
(c) Imagem com bordas enfatizadas pelo operador de Roberts	37
(d). Negativo da imagem.	37
Figura 17 - Fluxograma do Sistema de Processamento e Visualização de Imagens, desenvolvido na estação gráfica Interpro	38
Figura 18 - Diagrama em blocos do GSP.	40

Figura 19 - Estrutura lógica de como é feito o endereçamento na memória do GSP.	41
Figura 20 - Mapa de memória do GSP.	41
Figura 21 - Mapeamento de pixels na memória.	44
Figura 22 - Registradores dos conjuntos A e B.	47
Figura 23 - Registrador de "status" do GSP.	50
Figura 24 - Mapa de memória dos registradores de E/S.	51
Figura 25 - Tela apresentada pelo simulador.	62
Figura 26 - Diagrama em blocos da arquitetura dedicada utilizada.	65
Figura 27 - Níveis de um software que dispõe de interface com o usuário.	69
Figura 28 - Relacionamento entre o usuário e a máquina, através da arquitetura Servidor-Cliente.	71
Figura 29 - Relacionamento entre janelas, clientes e eventos.	72
Figura 30 - Exemplo de solicitação de uma operação de saída gráfica por um X Client.	73
Figura 31 - Exemplo de menu "pull-down".	75
Figura 32 - Exemplo de menu "pop-up".	76
Figura 33 - Hierarquia do Ambiente Gráfico.	79
Figura 34 - Especificação de uma janela.	81
Figura 35 - Exemplo de uma tela com janelas.	82
Figura 36 - Apresentação dos tipos e respectiva largura de bordas de janelas.	84
Figura 37 - Menu "pull-down" hierárquico.	86
Figura 38 - Menu "pop-up", onde há uma entrada textual e três entradas tipo botão.	86
Figura 39 - Leitura indireta da memória local do GSP pelo processador hospedeiro.	89
Figura 40 - Esquema de comunicação entre o GSP e o hospedeiro.	90
Figura 41 - Diagrama de blocos do carregador SDBLR.	91
Figura 41 - Diagrama de blocos do carregador SDBLR -Continuação.	92
Figura 42 - Organização da tela do monitor do hospedeiro, quando o SDBLR está em execução.	93
Figura 43 - Estrutura de dados LIE_janela.	98
Figura 44 - Visão da hierarquia de janelas da figura 35.	99
Figura 45 - Estrutura de dados LIE_tipo_janela.	99
Figura 46 - Estrutura de Dados LIE_fonte.	100
Figura 47 - Estrutura de dados lie_prof_janela.	101

Figura 48 - Estrutura de dados LIE_comunica.	103
Figura 49 - Estrutura de dados ArqSpec.	103
Figura 50 - Estrutura de dados pdmenu.	105
Figura 51 - (a) Fotografia que apresenta o sistema como um todo.	107
Figura 51 (b) - Fotografia que apresenta a tela do aplicativo em execução.	107
Figura 51 (c) - Fotografia que apresenta a imagem anterior com manipulação do histograma.	108

Lista de Tabelas

Tabela 1 - Opções de manipulações oferecidas pelo sistema de processamento de imagens.	31
Tabela 2 - Opções de Operações sobre imagens.	32
Tabela 3 - Opções que retornam estatísticas sobre regiões da imagem.	33
Tabela 4 - Descrição dos registradores do conjunto B.	48
Tabela 5 - Definição dos campos do registrador de "status".	50
Tabela 6 - Opções de processamento booleano de pixels.	59
Tabela 7 - Opções de processamento aritmético (ou cor) de pixels.	60
Tabela 8 - Rotinas para E/S pertencentes ao nível de Apoio do ambiente gráfico, e sua correspondência às da Linguagem "C".	94
Tabela 9 - Equivalência das funções de E/S (GSP ↔ Hospedeiro).	95

Resumo

Este trabalho descreve o desenvolvimento de um sistema de "software" para processamento, visualização e manipulação de imagens tomográficas obtidas por RM.

Primeiramente, foi desenvolvido um sistema para uma estação de trabalho Interpro 32C, da Intergraph. Este sistema, foi desenvolvido de forma amigável, utilizando-se janelas e "menus", para efetuar a interação com o usuário. Foram analisadas e implementadas várias operações de processamento de imagens, tais como: filtros digitais, manipulação da imagem através do seu histograma e medidas estatísticas sobre regiões da imagem.

A experiência e o conhecimento obtido com esta implementação, foram utilizados para definir e desenvolver um novo ambiente gráfico, para uma arquitetura de "hardware" baseada no processador gráfico TMS34010 da Texas Instruments, previamente desenvolvida no Laboratório de Instrumentação Eletrônica (LIE). Este ambiente, que suporta menus e janelas, foi desenvolvido utilizando-se das instruções gráficas inerentes à família TMS340. Foram desenvolvidas rotinas de comunicação com o hospedeiro, as quais permitem, por exemplo, que as imagens armazenadas no hospedeiro sejam transmitidas para a arquitetura dedicada. As mesmas rotinas foram utilizadas para depurar e validar o sistema gráfico.

Abstract

The present work describes the development of a software system to process, visualize and manipulate Magnetic Resonance Images.

First an application software was developed for an Interpro 32C Intergraph workstation. In this menu and window based software, image operations such as filters, transformations and image manipulation based on histograms with statistical data was implemented and evaluated.

The knowledge and experience obtained with this implementation was subsequently used to define and design a new graphic environment for a dedicated TMS340 based hardware architecture previously developed in our laboratory. This new toolkit provides window and menu support, and was developed through extensive use of the powerful TMS340 native graphics instructions. Debugging and tracing routines were developed based on an implemented host (PC) - graphics processor board communication system. The same communication software was also used in the examples to perform menu selected downloads of images and functions from the host.

Capítulo I

Introdução

1.1 - Considerações Iniciais

Os sistemas computacionais evoluíram muito desde o início do desenvolvimento dos computadores. Em seus primórdios, a entrada e saída de informações era feita essencialmente através da utilização de papel, através de listagens, de cartões ou fitas perfuradas. Dessa forma toda a interação dos sistemas computacionais com o meio externo era feita "off line". A partir da década de 70, com o advento dos sistemas operacionais de tempo compartilhado e a disseminação dos terminais de vídeo, os usuários passaram a ter um contato mais direto com os sistemas computacionais, em sistemas ditos "on line". No entanto mesmo utilizando-se de terminais de vídeo, estes em geral manipulavam apenas textos, e grande parte das informações continuaram sendo transmitidas de forma textual. Os terminais gráficos existentes nessa época, além de serem muito caros, eram de tecnologia vetorial, cujo emprego em aplicações que requeressem a representação de áreas, imagens, cor, etc. eram inviáveis.

O emprego da tecnologia "raster" para a construção de terminais de vídeo tornou-se viável com o aumento da densidade das memórias e a diminuição de seu custo. Assim, comparada à tecnologia vetorial, pode-se dizer que a tecnologia "raster" suporta tanto imagens matriciais ("raster"), quanto imagens vetoriais. Com o advento dessa tecnologia, possibilitou-se que, através da utilização de dispositivos gráficos, emergisse o desenvolvimento de interfaces gráficas com o usuário.

A utilização de dispositivos gráficos é uma forma mais natural e eficiente de comunicação do usuário com o computador, pois a informação pode ser transmitida e recebida de uma forma muito mais amigável. Isto é refletido por um bem conhecido provérbio chinês: "Uma figura vale mais do que 1000 palavras".

A figura 1 apresenta uma representação em blocos de um sistema gráfico. Pode-se ver que o usuário interage com o sistema através do dispositivo gráfico de entrada/saída,

usualmente o vídeo do computador e um teclado. Um dos pontos chave para que um sistema computacional tenha sucesso é que se consiga fazer com que o usuário realmente "alcance" tudo que ele tem a oferecer. Dessa forma, uma das premissas para os projetistas de software é construir sistemas de fácil utilização por qualquer usuário e não apenas por peritos em computação. A busca de interfaces "amigáveis" com o usuário, levou ao conceito de se procurar mostrar ao usuário todo o elenco de suas opções através de texto, ou mesmo apresentando-as através de pequenos desenhos (ícones) que seriam facilmente associados a sua função. Um ponto relevante desses novos conceitos é que, através de sistemas operacionais que permitem multitarefa, pode-se permitir ao usuário trabalhar em várias atividades simultaneamente, mantendo para cada atividade o seu ambiente próprio.

Figura 1 - Ilustração de um sistema gráfico.

Dessa forma, a construção de interfaces com o usuário, apoiadas diretamente em ícones, menus, janelas (gráficos), etc., tem se tornado um padrão para os projetistas de software.

Muitas áreas têm recebido contribuições oriundas da computação e, atualmente, a medicina tem sido uma grande contemplada; tanto na área de visualização, processamento e manipulação de imagens, quanto para a definição de diagnósticos através de utilização de sistemas especialistas [RHO_90] [RHO_91] [STA_91]. Considera-se mesmo que, dentro de dois anos, mais da metade dos hospitais e centros de saúde em todo o mundo contarão com sistemas de arquivamento e apresentação de imagens computadorizadas [FIN_90]. O

presente trabalho trata assim justamente do estudo e da implementação de um ambiente gráfico para visualização de imagens obtidas por Tomografia por Ressonância Magnética (ToRM).

1.2 - Objetivo do Trabalho.

O objetivo deste trabalho é o de avaliar e determinar as necessidades, que um ambiente de visualização de imagens por ToRM deve conter. Para isso, um protótipo de um tal sistema foi efetivamente construído, incorporando recursos que abrangem todas as facetas das quais um ambiente que seja um produto final deve dispor. Além disso, foi levado em consideração o fato de que as "facilidades" implementadas devem ser colocadas de forma que o usuário não se sinta inibido de utilizar o sistema, por achar que apenas peritos em computação poderiam efetivamente utilizá-lo. Este sistema de Visualização e Processamento de imagens foi desenvolvido de uma forma modular, de modo que pudesse ser migrado para outros computadores sem muito esforço.

Assim, com o desenvolvimento desse sistema, obtiveram-se os pré-requisitos necessários à implementação de um ambiente gráfico para a arquitetura dedicada a visualização das imagens. Esse ambiente gráfico foi construído para fornecer ao projetista de software as ferramentas necessárias à construção de sistemas interativos, para visualização de imagens por ToRM e também para suportar o sistema de visualização desenvolvido na fase anterior do trabalho.

1.3 - Contribuições deste Trabalho.

Destacam-se como contribuições mais importantes deste trabalho o estudo e a implementação de um sistema de visualização de imagens tomográficas obtidas por RM, que é composto por dois ambientes. O primeiro deles, é um sistema de visualização e processamento de imagens, onde várias operações para processamento e manipulação de imagens tomográficas foram implementadas e colocadas à disposição do usuário, através de uma interface amigável.

O segundo sistema é um ambiente gráfico, que fornece ferramentas para a construção de sistemas de visualização de imagens em uma arquitetura dedicada. Este

ambiente oferece primitivas gráficas para a construção de sistemas interativos. Assim, procurou-se explorar a disponibilidade de características específicas de hardware, para o tratamento de operações gráficas existentes no sistema. Desta forma, pôde-se construir um ambiente gráfico que poderá ser utilizado por um projetista de "software", para a construção de sistemas interativos de visualização. A principal forma de interação é através de ícones, aos quais são associados "menus", sendo que esta forma de interface é bastante utilizada em sistemas interativos.

Dessa maneira, foi criado um conjunto de ferramentas que permite, a pessoas que possuem um bom conhecimento de software e dos processos físicos envolvidos com a obtenção de imagens tomográficas (físicos e analistas de software), a construção de ambientes que sejam facilmente compreensíveis e amigáveis para serem usados pelos usuários finais (médicos e radiologistas). O conjunto de ferramentas desenvolvido neste trabalho é expansível, permitindo que novas técnicas e procedimentos que estão ou venham a ser pesquisados possam ser incluídos nos ambientes construídos, ao mesmo tempo que incorpora também um conjunto de ferramentas que procura fazer com que os ambientes criados interajam com os usuários finais de maneira amigável e familiar. Isto é obtido aliando-se recursos tais como gerenciadores de janelas e menus aos procedimentos de manipulação e processamento de imagens. Assim, o amplo conjunto de recursos que os processos de obtenção e manipulação de imagens tomográficas propiciam, podem ser configurados com facilidade pelos especialistas, em pacotes mais facilmente assimiláveis e mais rapidamente utilizados por parte dos usuários finais (não especialistas).

1.4 - Descrição do Trabalho.

Este trabalho faz parte de um projeto existente no Departamento de Física e Ciência dos Materiais (DFCM) do Instituto de Física e Química de São Carlos (IFQSC) - USP, que é a construção de um tomógrafo computadorizado que utiliza a técnica de Ressonância Magnética (RM) - TORM.

O trabalho aqui apresentado, pode ser dividido basicamente em duas partes:

- O desenvolvimento de um Sistema de Processamento e Visualização de Imagens (SPVI) tomográficas obtidas por RM.
- O desenvolvimento de um ambiente gráfico para uma Arquitetura dedicada à Visualização de Imagens tomográficas, desenvolvida no Laboratório de

Instrumentação Eletrônica (LIE) do IFQSC - USP. Assim, torna-se possível que o processo de visualização das imagens tomográficas possa ser transferido para a arquitetura dedicada.

A primeira parte, envolve o desenvolvimento de um sistema de processamento e visualização de imagens, em uma "estação de trabalho". Este trabalho, permitiu a avaliação tanto da implementação de vários recursos de processamento de imagens para RM, quanto da Interface Homem/Máquina implementada, analisando a sua validade para a aplicação proposta. Esta avaliação foi muito importante para a segunda fase do projeto, conforme será visto a seguir. Como plataforma de desenvolvimento deste sistema de processamento de imagens foi utilizada uma estação de trabalho Interpro 32C da Intergraph, a qual utiliza um processador *RISC (Reduced Instruction Set Computer)*, denominado Clipper¹.

Através do conhecimento adquirido com a construção deste sistema, em nível de software de apoio, tornou-se possível construir a segunda etapa do trabalho: o ambiente gráfico para a arquitetura dedicada.

A arquitetura de hardware para este projeto, foi desenvolvida pelo Grupo de Instrumentação Eletrônica deste departamento, utilizando-se do processador TMS34010 da Texas. Esse processador, é o primeiro de uma família de processadores gráficos (TMS340), de última geração.

O desenvolvimento do sistema de processamento de imagens na estação de trabalho, foi executado primeiramente e realizado paralelamente à construção da arquitetura dedicada. O software básico para a arquitetura dedicada foi desenvolvido a seguir, incorporando características desejáveis para a aplicação em RM.

1.4 - Apresentação do Trabalho.

A tese é dividida em 7 capítulos, os quais são brevemente descritos a seguir:

No capítulo II, são introduzidos aspectos relevantes de processamento digital de imagens, muitos dos quais foram utilizados e/ou implementados no sistema de

¹ equipamento obtido através de financiamento da Fundação Banco do Brasil (FBB).

processamento de imagens desenvolvido.

O capítulo III descreve como foi implementado o sistema de processamento de imagens para tomografia por Ressonância Magnética e as operações por ele realizadas, apresentando, ainda, exemplos de imagens por ele manipuladas.

No capítulo IV, apresenta-se uma visão geral sobre a arquitetura dedicada, baseada no processador gráfico TMS34010 (Texas Instruments), para visualização de imagens, desenvolvido no LIE. Nessa apresentação procura-se ressaltar os pontos mais relevantes da arquitetura em termos da construção do ambiente gráfico desenvolvido.

No capítulo V, apresenta-se uma revisão sobre Ambientes de Gerenciamento de Janelas, levando-se em conta que a sua utilização é a base para a construção de interfaces com o usuário. As ferramentas atualmente utilizadas para a construção de interfaces Homem/Máquina são compostas basicamente por menus, janelas, ícones, e algum dispositivo de E/S manipulável com facilidade pelo usuário. Neste capítulo também apresenta-se como essas ferramentas podem ser gerenciadas por um sistema de software.

O capítulo VI, mostra como foi implementado o ambiente gráfico para a arquitetura dedicada (TMS34010). Apresenta a hierarquia de desenvolvimento do ambiente, os seus componentes e as principais estruturas de dados utilizadas. A sua utilização é ilustrada através de um exemplo.

No capítulo VII, são apresentadas as conclusões gerais sobre o trabalho realizado, bem como as linhas de futuras pesquisas abertas pelo presente trabalho.

Capítulo II

Introdução sobre Aspectos de Processamento Digital de Imagens Utilizados no Sistema Desenvolvido

2.1 - Introdução.

A área de Processamento Digital de Imagens tem crescido consideravelmente nas últimas duas décadas, com a utilização de imagens e gráficos em uma grande variedade de aplicações, aliado ao fato de que a tecnologia de construção de computadores também tem se aprimorado, possibilitando a utilização de sistemas mais eficientes e mais baratos. Muitas áreas vêm utilizando Sistemas de Processamento Digital de Imagens, tais como: indústria, medicina, agricultura, pesquisas espaciais, meteorologia, etc.

Muitos sistemas de processamento de imagens, possuem a característica de serem interativos, ou seja, necessitam da presença de um operador humano (usualmente um especialista na área de aplicação do sistema), participando do fluxo de informações do sistema [SAN_85]. A figura 2 ilustra esquematicamente as etapas presentes numa seção típica do uso de um sistema de processamento de imagens.

O interesse deste trabalho está na aplicação de processamento digital de imagens em medicina. Isto porque através de sistemas de processamento de imagens pode-se ajudar ao usuário, neste caso o médico, a visualizar "melhor" detalhes de imagens por eles anteriormente vistas apenas de forma estática e sem nenhum auxílio [ADA_90]. Este é justamente o caso de Sistemas de Processamento de Imagens para Tomografia Computadorizada, pois o usuário pode manipular as imagens interativamente, da forma que lhe for mais conveniente.

Figura 2 - Etapas do processamento interativo de Imagens.

Neste capítulo é apresentada uma visão global das principais características de processamento digital de Imagens, sendo que tais características foram utilizadas e/ou implementadas pelo sistema de processamento de imagens desenvolvido.

2.2 - Considerações Sobre Imagens.

Neste trabalho sempre que se referir a imagens, deve-se entender *imagens digitais*, ou seja, são aquelas que por algum processo de amostragem ou aquisição representam uma imagem abstrata.

Matematicamente, Imagem pode ser definida como uma função bidimensional $A(x,y)$ definida em uma certa região do plano.

$$A : [0, r] \times [0, s] \rightarrow [0, t]$$

Assim a imagem é definida num retângulo $[0, r] \times [0, s]$, e os valores tomados estão contidos no intervalo $[0, t]$. Ao valor $A(x, y)$ no ponto (x, y) dá-se o nome de *Nível de cinza*.

Imagens são geralmente definidas como uma grande coleção de dados, que estão frequentemente dispostos em forma de uma matriz ou alguma outra estrutura de dados discreta [PAV_82]. Por exemplo, se a dimensão de uma imagem é de 1024 por 1024 pontos, e a cor associada a cada ponto é representada por um valor representado com n -bits, então ela ocupa $n/8$ MBytes de memória (em computador de palavra de 8 bits). Dessa forma muitos sistemas gráficos dedicam um cuidado especial a memória que armazena a imagem, a qual normalmente é chamada de "**Frame Buffer**".

Uma imagem, é representada através de um conjunto de valores, onde cada valor é um número que descreve os atributos de um pixel na imagem. Usualmente estes números são inteiros positivos, que representam a intensidade na escala de tons de cinza de um ponto da imagem, ou a cor associada ao ponto pela "look-up table" (ver seção 2.7) em sistemas que comportam este procedimento. Quando utiliza-se escalas de tons de cinza, usualmente associa-se a tonalidade de cinza mais escura, o preto, ao nível zero da escala de cinza, e o mais claro, o branco, ao maior valor permitido na escala. Ou seja, se cada pixel é representado em um sistema de 8-bits, o preto é associado ao valor zero, e o branco ao valor 255.

As dimensões do conjunto de valores que especifica a imagem são chamadas de **largura** e **altura** da imagem, e o número de bits associado com cada pixel da matriz é chamado de **profundidade** da imagem. A profundidade da imagem está associada ao número de planos de memória em que uma imagem é mapeada, sendo que cada bit da palavra que armazena o valor do pixel reside em um plano da imagem ("frame buffer") [ACQ_82]. Na figura 3 mostra-se um esquema de um pixel p com profundidade 8.

Figura 3 - Representação de um pixel P com 8-bits (8 planos) de profundidade.

Através de Processamento Digital de Imagens, pode-se analisar e modificar imagens por computador. Os objetivos deste processamento são [MAS_89]:

1. Extrair informação da imagem, ou seja, reconhecer elementos que compõem a imagem, permitir medir elementos contidos na imagem, classificá-los e compará-los com algum modelo colocado.
2. Transformar a imagem (aumentando contraste, realçando bordas, corrigindo alguma imperfeição gerada na imagem durante o processo de aquisição), de forma que a informação seja mais facilmente discernível por um analista humano.

2.3 - Tipos de Manipulação de Imagens.

Na análise de imagens, a entrada do processamento é uma imagem, enquanto a saída é uma descrição não pictórica da imagem. Este processo pode ser entendido como de "redução de dados", em que se diminui o volume dos dados, mas mantém-se o conteúdo da informação relevante para uma dada aplicação [MAS_89].

Na manipulação de imagens em geral, tanto as entradas quanto as saídas são imagens. Duas grandes classes de transformações são:

- *Transformações Radiométricas*: onde os valores de níveis de cinza dos pontos da imagem são alterados, sem modificação de sua geometria.
- *Transformações Geométricas*: onde a geometria da imagem é alterada, mantendo-se o máximo possível os valores dos níveis de cinza.

As transformações radiométricas e geométricas podem ser feitas com a finalidade de eliminar distorções da imagem, introduzidas geralmente pelo próprio sistema de aquisição de imagens ("restauração"), ou enfatizar certas características da imagem ("realce").

2.4 - Transformações Radiométricas.

Pode-se ter basicamente dois tipos de transformações ou operações radiométricas:

Restauração - que visa corrigir alguma distorção sofrida pela imagem, e

Realce - que procura enfatizar alguma característica de interesse da imagem. Por vezes estes dois objetivos produzem resultados coincidentes.

É o caso, por exemplo, quando uma imagem sofreu uma distorção que diminuiu seu contraste; uma transformação que realce as bordas dos objetos das imagens pode, de fato, restaurá-la.

Embora muitas das técnicas de restauração e realce sejam as mesmas (por exemplo, filtragem), os objetivos e enfoques divergem num e noutro caso. O procedimento geral da restauração é a modelagem do processo de distorção para tentar invertê-lo. No realce esta preocupação não existe, pois no realce as técnicas utilizadas são, na maioria, heurísticas, não havendo compromisso com a imagem original [MAS_89].

As transformações radiométricas podem também ser classificadas quanto ao seu grau de abrangência, em:

1. **Operações Pontuais:** onde o nível de cinza de um ponto na imagem transformada depende só do nível de cinza do ponto da imagem original (ou nas imagens originais se houver mais de uma imagem de entrada ("frames")).
2. **Operações Locais:** onde o novo nível de cinza de um ponto depende não só de seu antigo nível de cinza, mas também dos níveis de cinza de seus pontos vizinhos.

2.4.1 - Operações Pontuais sobre Imagens.

Algoritmos que realizam *Operações Pontuais* percorrem a imagem e utilizam o valor do pixel associado a cada ponto, modificando a escala de tonalidades de cinza de uma imagem. Por exemplo caso deseja-se tornar uma imagem mais brilhante, pode-se adicionar algum valor a todos os pontos da imagem. Por exemplo, em um sistema de 8-bits por pixel, os valores associados vão de 0 a 255, pode-se definir uma função $brilho(p) = p + 40$, cuja função é tornar a imagem mais "clara". Através de operações pontuais pode-se também corrigir sombras que porventura tenham surgido junto com o processo de aquisição da imagem. Ou seja, define-se uma função cujo efeito reverta a operação de sombreamento indesejável, porém isto requer que se estime as funções de sombreamento.

Assim, uma operação pontual que toma uma imagem $A(x,y)$ e produz outra imagem $B(x,y)$, pode ser definida matematicamente como:

$$B(x, y) = f[A(x, y)]$$

2.4.1.1 - Histogramas de Intensidades.

O **Histograma de Intensidades** indica para cada nível de cinza da imagem a quantidade de pontos mapeados com tal nível. Ele contém uma informação global sobre os "objetos" da imagem. Se todos os pontos da imagem são de um mesmo objeto, o histograma mostra a probabilidade condicional $p(z/objeto)$ de um ponto possuir um dado nível de cinza z , sendo que o ponto pertence ao objeto. A figura 4 apresenta uma imagem e seu histograma de intensidades associado.

Figura 4 - Imagem de um corte transversal de cabeça humana e seu histograma associado (obtidos do sistema de processamento e visualização de imagens desenvolvido no LIE).

Através de Histogramas de Intensidades pode-se obter uma maneira simples para se manipular o contraste de imagens. Dessa forma, pode-se definir as "janelas" de intensidade que se queira manipular. Por exemplo, se ao construir o histograma de uma imagem vê-se que não se está utilizando toda a abrangência dos níveis de cinza, resultando uma imagem com pouco contraste, pode-se forçar que ela seja mapeada usando todos os níveis de cinza, ou uma faixa deles ("janela"), ou mesmo aumentar o contraste em apenas uma região dos níveis de intensidade da imagem [GON_87].

Suponha-se uma imagem $A(x,y)$ e seu histograma $H_i(A)$. Se um ponto p da imagem

está mapeado com o nível N dentro da faixa de valores Δ_1 , cujo nível mais baixo é $min1$ e o mais alto é $max1$, e através de manipulação no histograma deseja-se transformá-lo para o valor p' com nível N' , o qual estará dentro da faixa Δ_2 cujo valor mínimo é $min2$ e máximo $max2$, a função de transformação será:

$$f(N) = \frac{(N-min1)(max2-min2)}{(max1-min1)} + min2$$

$$N' = f(N)$$

A figura 5 ilustra a transformação do histograma $H_1(A)$ da imagem A para o histograma $H_2(A)$.

Figura 5 - Efeito de uma transformação dos níveis de cinza efetuado sobre o histograma.

Esse procedimento de manipulação do histograma para ampliar ou diminuir o contraste de uma imagem é mostrado na figura 6. Na parte (a) apresenta-se um histograma de uma imagem, na qual especifica-se uma janela de visualização, e na parte (b) o histograma da mesma imagem depois da manipulação do histograma. No capítulo 3, as figuras 15 e 16 ilustram este procedimento no sistema desenvolvido no LIE.

Figura 6 - Manipulação de janelas de intensidade da imagem através do histograma.

2.4.2 - Operações Locais Sobre a Imagem.

Algoritmos que realizam *Operações Locais* utilizam informação dos valores dos pontos vizinhos para modificar o valor de um ponto, ou mesmo para verificar a existência de alguma propriedade neste ponto da imagem.

Operações Locais são tipicamente utilizadas para filtragem espacial e alteração da própria estrutura da imagem. Elas podem "aguçar" a imagem, acentuando as mudanças de intensidades (através de filtros Passa-Altas), ou "suavizar" a imagem, tornando as mudanças de intensidade menos abruptas (através de filtros Passa-Baixas), ou podem prover outras formas de se melhorar a imagem. Assim, pode-se procurar formas na imagem através de "padrões de busca" ("match"), obter medidas dos elementos da imagem, definir bordas na imagem, remover ruído suavizando a imagem [DAW_87].

As transformações locais são divididas em dois grandes grupos:

Transformações Lineares: são aquelas em que os pixels vizinhos ao que está sendo calculado influenciam no novo valor segundo uma combinação linear dos mesmos. Tais operações incluem superposição e convolução de imagens, o que é usualmente utilizado para implementar filtros discretos [PRA_78].

Transformações não Lineares: utilizam outra forma de arranjo que não a linear.

2.4.2.1 - Operação de Convolução.

Um dos algoritmos clássicos de processamento de imagens é aquele que efetua a **Convolução** entre imagens, sendo comumente usado para realizar filtragem espacial e procurar peculiaridades na imagem.

A operação de convolução substitui o valor de um ponto com o valor obtido através de uma operação linear do ponto e de seus vizinhos, sendo que é atribuído "peso" aos vizinhos e ao ponto. Estes "pesos" são colocados em uma matriz, usualmente de dimensão pequena, chamada de máscara. Se para obter um novo valor de um ponto deseja-se que apenas os vizinhos imediatos influenciem, escolhe-se uma máscara de dimensão 3x3, onde o ponto base para a geração do novo ponto é colocado no centro da máscara.

Dadas duas imagens $A(x,y)$ e $B(x,y)$, com $x=0,1,...,M-1$ e $y=0,1,...,N-1$, a convolução (discreta) destas duas imagens é uma terceira imagem $C(x,y)$ com $x=0,1,...,M-1$ e $y=0,1,...,N-1$, dada pela seguinte expressão:

$$C(x, y) = \frac{1}{M \cdot N} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} A(m, n) \cdot B(x-m, y-n)$$

e para os valores da borda da imagem, que se encontram fora dos intervalos $[0, M-1]$ e $[0, N-1]$, deve-se utilizar as seguintes relações:

$$A(x, -y) = A(x, N-y)$$

$$A(-x, y) = A(M-x, y)$$

As relações acima são equivalentes a considerar uma imagem $A(x,y)$ definida em $[0, M-1]$ e $[0, N-1]$, como definida em $[-M, M-1]$ e $[-N, N-1]$. O resultado da convolução de A por B num ponto p é na realidade uma média ponderada dos pontos de A , onde os pesos são dados pela imagem B .

Como foi mencionado acima, a operação de convolução é a base para efetuar-se filtragem espacial de imagens. A imagem que indica os pesos para a convolução normalmente possui dimensão pequena e ímpar (3, 5, 7 ou 9), pois a operação de convolução envolve uma grande quantidade de cálculos, e quanto maior o máscara maior

será o tempo de processamento da operação.

A seguir serão apresentadas algumas máscaras normalmente utilizadas para filtragem espacial [MAS_89]:

$$\begin{pmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{pmatrix}$$

Implementa um filtro-passa baixas (de suavização).

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

Normalmente utilizada para "aguçar" uma imagem, pois como alguns pesos são negativos eles tendem a realçar diferenças.

$$(a) \begin{pmatrix} 1 & 1 & 1 \\ 1 & -2 & 1 \\ -1 & -1 & -1 \end{pmatrix} \quad (b) \begin{pmatrix} -1 & -1 & -1 \\ 1 & -2 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Apresenta uma máscara de filtro para detecção de borda ao (a) norte e (b) ao sul do ponto.

$$(a) \begin{pmatrix} -1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & 1 & 1 \end{pmatrix} \quad (b) \begin{pmatrix} 1 & 1 & -1 \\ 1 & -2 & -1 \\ 1 & 1 & -1 \end{pmatrix}$$

Apresenta uma máscara de filtro para detecção de borda a (a) leste e (b) a oeste do ponto.

O realce de bordas, independente da direção pode ser obtido através de máscaras como:

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix} \quad \begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix} \quad \begin{pmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{pmatrix}$$

Pode-se também utilizar máscaras que contenham algum padrão que se quer

procurar na imagem. Neste caso a máscara é normalmente chamada de tentativa ("template"), onde o tamanho da máscara depende do padrão a ser procurado [DAW_87].

2.4.2.2 - Exemplos de Transformações não Lineares.

Usualmente as transformações não lineares conseguem uma melhor relação sinal/ruído na imagem produzida do que por transformações lineares [LEE_90]. Os filtros lineares utilizados para atenuar os ruídos de uma imagem (passa-baixas) produzem uma imagem "borrada", ou seja as bordas não ficam mais tão nítidas. Este efeito indesejado pode ser minorado através de transformações (filtros) não lineares [DAW_87].

Tome-se como exemplo os sinais unidimensionais dos tipos "ruído" e "degrau" mostrados na figura 7 [MAS_89]. Os efeitos de uma transformação linear (filtro) dos tipos média $[1/3, 1/3, 1/3]$ e diferença $[-1, 3, -1]$ são mostrados na figura 8. Pode-se notar nestes casos os efeitos de atenuação do ruído (a), "borramento" da borda (b), ampliação do ruído (c) e realce da borda (d). Os efeitos indesejados mostrados na figura 8 (c) e (d) podem ser minorados através de filtros não lineares, como será visto a seguir.

Figura 7 - Sinais "ruído" (a) e "degrau" (b).

Figura 8 - Efeitos de filtros lineares ao "ruído" e à "borda": (a) efeitos de atenuação; (b) borramento da borda; (c) ampliação do ruído; (d) realce da borda.

Por exemplo, no *filtro da mediana*, os pontos da vizinhança do ponto (x,y) são ordenados, e tomado como novo valor para (x,y) o valor mediano obtido pela ordenação. O efeito do filtro da mediana (unidimensional) que envolve 3 elementos (o ponto e seus vizinhos mais próximos) é mostrado na figura 9. Note-se que na parte (a) da figura o ruído foi eliminado (não somente atenuado), enquanto em (b) o degrau permaneceu inalterado.

Figura 9 - Efeito do filtro da mediana sobre o "ruído" e o "degrau".

É também usual, em vez de se utilizar a mediana da vizinhança, escolher o valor máximo ou o valor de ordem qualquer. Se a imagem é binária, para se obter o valor mínimo entre dois pontos, pode-se utilizar a operação lógica E dos valores da vizinhança dois a dois, e para o valor máximo a operação lógica OU.

Uma outra alternativa utilizada é tomar-se o valor mais frequente de uma vizinhança, ou seja, utilizar a *Moda*, entre eles.

Pode-se utilizar vizinhança seletiva de um ponto, isto é, tomar a média não em toda a vizinhança, mas sim numa sub-vizinhança que satisfaça alguma propriedade dada, como uniformidade, ou seja, que apresente um valor médio próximo do valor do ponto (x,y) . O uso do filtro da vizinhança seletiva presume que a imagem seja composta de regiões razoavelmente uniformes. Assim, a seleção da vizinhança procura fazer com que o ponto (x,y) seja operado só com pontos de sua região.

Pode-se também utilizar filtros não-lineares para detectar, bordas e linhas curvas. Para o realce de bordas o método não-Linear mais simples é o **Operador Gradiente de Roberts (GR)**, dado por [MAS_89]:

$$GR(x, y) = [[A(x, y) - A(x-1, y-1)]^2 + [A(x, y-1) - A(x-1, y)]^2]^{\frac{1}{2}}$$

onde: $A(x, y)$ é a imagem dada.

Uma desvantagem do operador de Roberts é sua assimetria, pois, dependendo da direção, certas bordas são mais realçadas que outras, mesmo tendo igual magnitude.

Um operador gradiente para delimitação de bordas mais sofisticado (3x3), é o **Operador de Sobel (GS)**, o qual é frequentemente utilizado como o primeiro passo para algoritmos de visão computacional [DAW_87]. Este operador é dado por:

$$GS(x, y) = (X^2 + Y^2)^{\frac{1}{2}}$$

onde:

$$\begin{cases} X = [A(x-1, y+1) + 2 \cdot A(x, y+1) + A(x+1, y+1)] - \\ \quad [A(x-1, y-1) + 2A(x, y-1) + A(x+1, y-1)] \\ Y = [A(x-1, y-1) + 2 \cdot A(x-1, y) + A(x-1, y+1)] - \\ \quad [A(x+1, y+1) + 2A(x+1, y) + A(x+1, y-1)] \end{cases}$$

2.5 - Transformações Geométricas.

As transformações geométricas mudam o arranjo espacial dos pixels da imagem. Muitas vezes são utilizadas para corrigir distorções causadas pelo processo de aquisição, ou mesmo para efetuar alguma manipulação solicitada pelo usuário. Transformações geométricas típicas são: rotação, translação, ampliação, ou diminuição da imagem. Algoritmos para efetuarem estas operações podem ser apresentados como um conjunto de equações que mapeiam um pixel localizado na posição (x, y) em um novo endereço, (x', y') [DAW_87].

Na realidade, em processamento digital de imagens, geralmente são necessários dois algoritmos separados para a maior parte das transformações geométricas. O primeiro é o algoritmo que define a *movimentação* dos pixels, como ele *move* a partir da posição inicial para a posição final.

Como exemplo, pode-se tomar a imagem ilustrada na figura 10. Ali uma imagem

é rotacionada de 90 graus. Um pixel na posição (0,0) é mapeado segundo a fórmula:

$$x' = (Y\text{SIZE} - 1) - y \quad \text{e} \quad y' = x$$

onde YSIZE é o número de pixels no eixo y.

Figura 10 - Uma transformação geométrica de 90 graus.

Em muitas aplicações, é necessário preservar a continuidade dos objetos que definem a imagem. Isso pode ser feito transformando-se ponto a ponto a imagem, ou transformando-se apenas pontos chave da imagem, fazendo que os outros pontos acompanhem a transformação por algum outro processo. Pode-se ver que o primeiro processo é geralmente mais oneroso em termos de tempo de computação do que o segundo.

É conveniente especificar matematicamente o relacionamento espacial entre pontos das imagem de entrada e da imagem de saída. A definição geral para operações geométricas é [CAS_79]:

$$g(x, y) = f(x', y') = f(a(x, y), b(x, y))$$

onde: $f(x, y)$ é a imagem de entrada e $g(x, y)$ é a imagem de saída. As funções $a(x, y)$ e $b(x, y)$ especificam as transformações espaciais. Se essas transformações forem contínuas, a continuidade será preservada na nova imagem.

O segundo algoritmo necessário para uma transformação geométrica é um algoritmo para interpolação dos valores dos níveis de cinza. Na imagem de entrada $f(x, y)$, os valores dos níveis de cinza são definidos somente para valores inteiros de x e y . Mas na equação geral acima, pode-se ver que o valor do nível de cinza para $g(x, y)$ pode ser tomado a partir de coordenadas não inteiras (pela continuidade) de $f(x, y)$. Se a operação geométrica é

considerada como um mapeamento de f para g , pixels de f podem ser mapeados entre pixels de g e vice-versa. Assim novamente deve ser realizada alguma operação (ex. escala) para corrigir a distorção de pontos não inteiros [CAS_79].

2.6 - Operações Algébricas.

Por definição, **operações Algébricas**, são aquelas que produzem uma imagem que é a soma, diferença, produto ou quociente pixel a pixel de suas imagens. No caso de soma ou multiplicação mais de duas imagens podem estar envolvidas.

As quatro operações algébricas de processamento de imagens podem ser explicitadas por [RIT_90]:

$$\begin{aligned}C(x, y) &= A(x, y) + B(x, y) \\C(x, y) &= A(x, y) - B(x, y) \\C(x, y) &= A(x, y) \times B(x, y) \\C(x, y) &= A(x, y) / B(x, y)\end{aligned}$$

onde: $A(x, y)$ e $B(x, y)$ são imagens de entrada e $C(x, y)$ é a imagem resultante.

A seguir são apresentadas algumas aplicações imediatas para as operações algébricas:

1. Uma importante utilização de adição de imagens, é para se obter a média de múltiplas imagens de uma mesma cena. Isto é eficazmente utilizado para reduzir os efeitos de ruídos aleatórios aditivos. Pode também ser utilizado para colocar o conteúdo de uma imagem sobrepondo outra.
2. Subtração de imagens pode ser utilizada para remover algum padrão indesejável presente em uma imagem. A subtração é também utilizada para detectar mudanças entre duas imagens da mesma cena. Por exemplo, pode-se detectar movimento subtraindo-se imagens sequenciais de uma cena. Subtração de imagens pode ser usada para calcular o gradiente, uma função muito usada para detecção de bordas (ver seção 2.4.2.2). Uma utilização típica em medicina, é para se comparar duas imagens de uma mesma região do corpo. Obtém-se a imagem da região (ou órgão) normal e também após injetar um meio de contraste, adquire outra imagem. Após esta operação, pode-se comparar as duas através de subtração digital. Assim, por exemplo, se o coração é o órgão em apreciação, consegue-se obter uma imagem apenas

das artérias e veias.

3. Multiplicação e divisão de imagens podem ser utilizadas, por exemplo, para corrigir possíveis defeitos de um digitalizador em que a sensibilidade do sensor de luz varia conforme a posição dos pontos da imagem. Multiplicar uma imagem por uma "máscara" pode esconder certas regiões da imagem deixando expostos apenas objetos de interesse.

Ao se efetuar operações sobre as imagens, deve-se estar atento para que o novo valor obtido não ultrapasse o domínio de valores, permitido pela amplitude dos níveis de intensidade em que a imagem está mapeada. Se este cuidado não for tomado, pode-se obter valores totalmente inconsistentes, levando-se a uma degeneração da imagem.

2.7 - Tabelas "Look-up".

Em sistemas que permitem trabalhar com cores, cada cor é uma composição de três cores primárias: vermelho, verde e azul (do inglês, RGB). E os níveis de cinza também são uma composição das três cores, sendo que todas entram em partes iguais. Logo, o número de bits utilizados para representar o valor do pixel é dividido entre as três cores. Assim, se um pixel possui profundidade 1 (um bit por pixel), o pixel poderá ser pintado apenas com duas cores (2^1) (por exemplo, em preto e branco). Se o pixel utiliza dois bits, poderá produzir no máximo quatro cores (2^2), e assim por diante. Usualmente 6 bits são suficientes para permitir a visualização da imagem de forma que não se perceba descontinuidade entre as tonalidades de cinza, mas 8 ou mais bits podem ser necessários em algumas aplicações. Assim, para monitores coloridos, sugere-se que três vezes a quantidade de bits necessária é um bom valor, ou seja, 8 bits para cada cor primária aditiva (vermelha, verde e azul - RGB), produzindo 24 bits por pixel [FOL_91].

Em muitos sistemas gráficos é desejável poder-se alterar rapidamente a coloração da imagem sem alterar a sua forma. Assim permite-se alterar a aparência visual de uma imagem, sem na realidade alterar os próprios dados que a definem. Por exemplo, mostrar todos os pixels abaixo de um certo valor com o preto, ou mesmo a partir de uma imagem monocromática criar uma imagem em pseudo-cor.

Um dispositivo que facilita grandemente essa manipulação com cores, sem comprometer a eficiência da operação de visualização de imagens são as tabelas de Busca

("Look-up Table" (LUT)), que os controladores de monitores por varredura vem utilizando. A tabela "look-up" possui tantas entradas quanto o número de valores armazenados pelo pixel, ou seja, se o pixel tem profundidade 8, a LUT terá 256 (2^8) entradas. Assim, o valor do pixel não indica a cor em que ele é mapeado, mas sim a entrada na LUT onde a cor está armazenada. A figura 11 ilustra o procedimento de mapeamento de um pixel através da LUT.

Figura 11 - Organização de uma LUT, para um sistema de pixels de 8 bits.

2.8 - Conclusões.

Os objetivos de processamento de imagens incluem modificação ou transformação da imagem para melhorar sua aparência ou enfatizar alguma de suas características, medição de elementos da imagem, e reconhecimento de elementos na imagem [DAW_87].

Neste capítulo foram apresentadas diversas técnicas comumente utilizadas em processamento de imagens. Algumas dessas técnicas são de especial interesse para a visualização de imagens obtidas por TORM. Será visto no próximo capítulo como o sistema de Processamento de Imagens desenvolvido as utilizou em sua implementação.

Capítulo III

Sistema de Processamento de Imagens para Tomografia Computadorizada por Ressonância Magnética

3.1 - Introdução.

Com o desenvolvimento de um tomógrafo que utiliza a técnica de Ressonância Magnética (RM), no Departamento de Física e Ciência dos Materiais do IFQSC-USP, tornou-se necessário construir um módulo que efetuasse o processamento das imagens obtidas. Isso foi feito com o intuito de auxiliar ao usuário, na interpretação das imagens. Ou seja, através de operações de processamento de imagens, pudesse melhorar o contraste da imagem, trabalhar com várias janelas de visualização, comparar várias "visões" simultâneas da imagem, etc.

Neste capítulo será apresentado o sistema de software desenvolvido com esta finalidade.

3.2 - Plataforma Computacional Utilizada.

O sistema de software foi desenvolvido em uma estação gráfica InterPro 32C de fabricação da Intergraph, que dispõe de uma tela de 1184 x 884 pontos, e pode apresentar até 64 cores simultâneas escolhidas de um total de 4096. Esta estação possui uma grande restrição a nível de visualização, pois permite que apenas dezesseis níveis de cinza puro sejam utilizados para apresentar as imagens. Esta quantidade de níveis de intensidade não é boa para a visualização de imagens em preto e branco, pois o olho humano ainda consegue distinguir as variações entre um nível e outro, e as imagens assim quantizadas ficam com aparência de "blocos". As imagens geradas pelo tomógrafo possuem

internamente 256 níveis de cinza. Como a estação Interpro permite que apenas 16 níveis de intensidade sejam apresentados simultaneamente, uma imagem visualizada na Interpro, também o pode ser na arquitetura dedicada, sem qualquer alteração (a qual será apresentada no capítulo 4).

A restrição colocada acima não inviabilizou o Sistema de Processamento de Imagens [SLA_89b] [TRA_89], devido ao fato de que esta máquina foi utilizada como uma plataforma intermediária de avaliação e desenvolvimento de software. Este sistema foi desenvolvido com o intuito de ser migrado para a arquitetura dedicada [SLA_89a] [PAI_90] que paralelamente foi sendo desenvolvida no Laboratório de Instrumentação Eletrônica do IFQSC - USP.

O software foi todo elaborado em linguagem "C" utilizando chamadas do ambiente gráfico Environ V que é suportado pela estação gráfica InterPro 32C. O software básico do sistema de Processamento de Imagens foi elaborado modularmente com o intuito de ser facilmente migrado para outras arquiteturas de computadores, procurando facilitar a interação com o usuário, através da utilização de "menus", envio de mensagens ao usuário relativas a cada passo do processamento, e entrada de dados através de utilização do "mouse".

3.3 - Coleta dos Dados Usando Ressonância Magnética.

A técnica de Ressonância Magnética (RM) para coleta de dados de estruturas internas é descrita por Panepucci et alli em [PAN_85]. Basicamente, pode-se dizer que para obter-se dados por RM, deve-se aplicar um campo magnético intenso na amostra a ser medida, o qual orienta os eixos de rotação dos núcleos de todos os átomos da amostra no sentido do campo.

Aplicando-se pulsos de radiofrequência e gradientes de campos magnéticos ortogonais, faz-se com que os núcleos executem o movimento de precessão temporariamente. O tempo que cada núcleo atômico leva para retornar à posição inicial, e a maneira como o faz depende de cada elemento atômico, de suas ligações químicas, e da intensidade local do campo magnético. Estas características permitem a aplicação de várias técnicas de aquisição de dados que resultam na obtenção de imagens, visualizadas como imagens tomográficas.

3.4 - Processamento dos Dados.

Aos dados coletados pelo tomógrafo é aplicada uma Transformada de Fourier Bidimensional, transformando assim as frequências e suas respectivas amplitudes do sinal amostrado em pontos espaciais com níveis de intensidade associados (este é o processo de reconstrução de imagens). Esses dados são normalizados em uma faixa de intensidades adequada, que atualmente é de 256 níveis, gerando finalmente a imagem a ser visualizada.

O processamento dos dados é dividido em dois tipos:

- o processamento dos dados efetuado no próprio processo para se obter a imagem, ou seja, dentro da própria Transformada de Fourier [JAI_89];
- o processamento efetuado depois que a imagem já foi obtida.

Um exemplo do primeiro tipo de processamento, é a interpolação da imagem durante o processo de reconstrução, ou seja, a Transformada de Fourier permite interpolar os dados coletados antes de se obter a imagem final, conseguindo-se assim uma ampliação da imagem para o tamanho desejado, com grau de integridade maior do que se comparado a técnicas de interpolação convencionais. Outro tipo de interpolação pode ser efetuado através das técnicas de interpolação convencionais, utilizando-se até mais do que um algoritmo de interpolação de cada vez [CAM_82] [CAM_83]. Algoritmos de interpolação explícitos fazem parte do conjunto de Processamento das imagens do segundo tipo, o qual é efetuado depois que a imagem já foi obtida, já a interpolação na própria Transformada de Fourier faz parte do primeiro tipo.

Depois de obtida a imagem, o software efetua apenas operações do segundo tipo sobre a mesma.

As imagens podem ser visualizadas atualmente nos tamanhos 128x128, 128x256, 256x256, 256x512, 512x512 pontos.

Imagens maiores depois de serem processadas, podem ser visualizadas em partes, ou seja, pode-se quebrá-las em quadrantes e visualizá-los separadamente.

Juntamente com a visualização das imagens na estação gráfica permite-se ao usuário manipular dinamicamente a tabela de cores da imagem (ver figuras 12 e 13). Dessa forma

o usuário pode interativamente alterar níveis de cores conforme o seu interesse, manipulando níveis de cinza em RGB. Esta manipulação da tabela de cores, faz com que o usuário possa alterar os níveis de intensidade (contraste) de regiões de interesse que foram previamente demarcadas.

Mas o aumento do contraste de uma faixa de pontos pode ser conseguido fazendo-se uma distribuição linear ou não dos tons de cinza disponíveis para as intensidades dos pontos. Um modo de permitir-se ao usuário escolher a divisão dos níveis de intensidade, é através de histogramas que mostrem para cada nível de intensidade a frequência de pontos a ele associada. Dessa forma pode-se fazer uma divisão linear ou não, que mostre regiões com determinada faixa de intensidades mais interessante para o usuário. Também juntamente com a visualização da imagem o usuário pode solicitar para que o histograma da imagem mostrada seja exposto (ver figura 14).

Outro ponto interessante para a manipulação das imagens de tomografia é a delimitação de áreas geométricas para determinar características específicas da imagem. No caso do tomógrafo de RM seriam parâmetros de densidade, constantes de tempo T1 e T2, medidas de tamanho, volume, etc.

3.5 - Manipulações Efetuadas pelo Sistema.

O sistema de software apresenta tipicamente três janelas: uma janela que mostra a imagem processada, uma segunda janela que mostra a tabela de cores em que a imagem está mapeada, permitindo-se alterá-la dinamicamente e uma terceira janela na qual apresenta-se o histograma normalizado da imagem, ou seja, para cada nível de cinza a quantidade de pontos da imagem a ele associado. Nesta versão do sistema, o histograma apresenta os pontos agrupados em faixas de 16 níveis de intensidade. Permite-se ao usuário definir pelo histograma, utilizando-se do "mouse", uma faixa de níveis de intensidade que ele deseja alterar na tabela de cores, facilitando a escolha de uma faixa de níveis pelo número de pontos da imagem a ela associada (ver figuras 14 e 15).

Outra opção que o usuário possui é a manipulação da imagem através de operações de filtragens, implementadas através de filtros espaciais tipo "passa-altas" e "passa-baixas",

e também operadores para delimitação de bordas. O sistema permite manipular até três imagens simultâneas, as quais são mantidas em três acumuladores *Imagem_A*, *Imagem_B* e *Imagem_C*. Pode-se efetuar operações aritméticas (adição e subtração) sobre as imagens, ou seja somar duas imagens obtidas por filtragens diferentes, ou fazer a diferença entre várias imagens (ver exemplo na figura 16). Também há previsão para a implementação das operações lógicas de "E", "OU" e "NÃO" (sendo realizadas por bolsista de Iniciação Científica). Tais imagens já operadas podem ser armazenadas em arquivos para depois serem utilizadas.

O sistema implementa uma operação de restauração, que corrige uma deformação da imagem oriunda do processo de aquisição. Em algumas imagens o sinal coletado perdeu intensidade de forma logarítmica nas bordas, e procurou-se corrigir isto multiplicando-se as regiões afetadas da imagem pela função inversa (ver tabela 2, entrada *Nivelamento*).

Foram implementados também os cálculos estatísticos de *média*, *desvio-padrão*, e *relação Sinal / Ruído* em regiões definidas pelo usuário, através do "mouse" ("rubberband") ou através da especificação das coordenadas do retângulo da imagem.

As tabelas 1, 2 e 3 apresentam sucintamente as opções de manipulação que o sistema oferece. A figura 17 apresenta um fluxograma do sistema desenvolvido.

<i>Entrada no Menu</i>	<i>Operação Realizada</i>
Imagem	Lê a imagem do arquivo especificado em disco, e a mostra na janela de imagem.
Histograma	Calcula e mostra o histograma da imagem presentemente em visualização (imagem corrente).
Tabela de Cores	Se a tabela de Cores não está mapeada na tela, ela é mapeada, e pode-se então manipular seus níveis de intensidade. Se a tabela estiver mapeada, apenas faz com que os níveis de cinza alterados retornem aos níveis de cinza puro originais.
Operações	Abre um sub-menu (ver tabela 2) com as operações permitidas sobre a imagem corrente.
Estatísticas	Abre um sub-menu (ver tabela 3) com os cálculos estatísticos oferecido.

Tabela 1 - Opções de manipulações oferecidas pelo sistema de processamento de imagens.

Operação	Sub_menu	Operação Realizada
Restaura	Original	Imagem corrente volta a ser a originalmente lida.
	Imagem_B	Imagem corrente é a do acumulador B.
	Imagem_C	Imagem corrente é a do acumulador C.
Op. de Roberts	-	Efetua o cálculo do Operador Gradiente de Roberts sobre a imagem corrente.
Op. de Sobel	-	Efetua o cálculo do Operador Gradiente de Sobel sobre a imagem corrente.
Passa-Baixas	-	Efetua uma filtragem espacial sobre a imagem corrente.
Passa-Altas	-	Efetua uma filtragem espacial sobre a imagem corrente.
Filtro da Mediana	-	Efetua uma filtragem não Linear sobre a Imagem corrente.
Salvar	Imagem_B $\leftarrow$ Imagem_A	Acumulador B recebe a imagem do Acumulador A.
	Imagem_C $\leftarrow$ Imagem_A	Acumulador C recebe a imagem do Acumulador A.
	Arquivo	O arquivo especificado recebe a imagem do corrente.
Operar	Troca Imagem_A com Imagem_C	Troca a imagem do acumulador A com a do acumulador B.
	Troca Imagem_B com Imagem_C	Troca a imagem do acumulador B com a do acumulador C.
	Imagem_A + Imagem_B	Efetua a soma da imagem do acumulador A com a do Acumulador B.
	Imagem_A - Imagem_B	Efetua a diferença da imagem do acumulador A com a do Acumulador B.
Nivelamento	Total	Efetua operação de restauração da imagem nivelando-a totalmente.
	Esquerda	Efetua operação de restauração da imagem nivelando seu lado esquerdo.
	Direita	Efetua operação de restauração da imagem nivelando seu lado direito.

Tabela 2 - Opções de Operações sobre imagens.

<i>Estatísticas</i>	<i>Operação Realizada</i>
Média	Calcula a média de uma região da imagem especificada "posicionalmente", ou através do "mouse" (via "Rubberband").
Desvio-Padrão	Calcula o desvio-padrão de uma região da imagem especificada "posicionalmente", ou através do "mouse" (via Rubberband).
Relação Sinal/Ruído	Calcula a relação Sinal/ruído de uma região da imagem especificada "posicionalmente", ou através do "mouse" (via Rubberband).

Tabela 3 - Opções que retornam estatísticas sobre regiões da imagem.

3.6 - Conclusões.

Neste capítulo foi apresentado o Sistema de Processamento de Imagens desenvolvido. Este sistema é visto como a parte do sistema de tomografia que interagiria diretamente com o usuário. Para a sua transferência para a arquitetura dedicada desenvolvida no LIE apresentada no capítulo 4 desta tese, foi necessário a construção de um ambiente gráfico que possibilitasse a utilização da arquitetura em um nível mais alto de programação. Agora a sua migração para a arquitetura baseada no TMS34010 é um trabalho que pode ser realizado com facilidade.

Fotos obtidas da tela do Equipamento Intergraph
InterPro 32C.

Figura 12 - Visualização de uma imagem tomográfica pelo sistema. Corte transversal de uma laranja.

Figura 13 - Imagem anterior com manipulação da tabela de cores.

Figura 14 - Imagem e seu histograma associado, preparando a manipulação do histograma.

Figura 15 - Manipulação da imagem através do seu histograma.

Figura 16 - Exemplos de operações sobre as imagens.
(a) Imagem com filtragem espacial tipo "passa-baixas".

(b) Imagem com filtragem espacial tipo "passa-altas".

(c) Imagem com bordas enfatizadas pelo operador de Roberts.

(d). Negativo da imagem.

Figura 17 - Fluxograma do Sistema de Processamento e Visualização de imagens, desenvolvido na estação gráfica Interpro.

Capítulo IV

Aspectos Relevantes da Arquitetura Dedicada Baseada no GSP TMS34010 para o Ambiente Gráfico Desenvolvido

4.1 - Introdução.

Neste capítulo é descrita sumariamente a arquitetura dedicada para visualização de imagens tomográficas construída no Laboratório de Instrumentação Eletrônica (LIE) do IFQSC-USP, com ênfase nas suas características que apoiaram mais diretamente a construção do ambiente gráfico descrito no capítulo 6. Uma apresentação completa da arquitetura é feita por Paiva em [PAI_90].

Esta arquitetura baseia-se no Processador Gráfico TMS34010 (GSP) da Texas Instruments, que é um avançado microprocessador de 32 bits, de propósito geral que é otimizado para aplicações em sistemas gráficos. Além do microprocessador, o circuito integrado inclui também um controlador gráfico, capaz de gerar os sinais de sincronismo para o vídeo.

O GSP pode ser configurado para servir tanto em ambiente "host_based" quanto em "stand_alone". Sistemas baseados em múltiplos dispositivos TMS34010 são implementados usando características especiais do GSP local e interfaces hospedeiras.

4.2 - Arquitetura do GSP.

O GSP é um processador de 32 bits, sendo que para os dutos internos, ULA, e o espaço de endereçamento são utilizados 32 bits, resultando num endereçamento de 128 Mbytes de memória. Possui ainda 30 registradores de 32 bits para operações de propósito

geral, um "stack pointer" de 32 bits e uma memória "cache" de instrução de 256 bytes.

A figura 18 apresenta o diagrama de blocos do GSP [TEX_86a], onde além do TMS34010 foram também colocados os seguintes circuitos:

- circuito para regeneração (*refresh*) de memórias do tipo "DRAM";
- circuito para gerar os sinais de sincronismo do vídeo;
- circuito para controle da atualização da tela;
- circuito para facilitar a comunicação com um processador hospedeiro.

Figura 18 - Diagrama em blocos do GSP.

4.2.1 - Organização da Memória.

O TMS34010 é um processador que permite o endereçamento de palavras de apenas um bit de memória, ou seja, cada endereço composto por 32 bits aponta para um individual bit da memória. A figura 19 ilustra a estrutura lógica da memória. A memória é acessada como uma sequência contínua de bits. Grupos de bits adjacentes formam uma estrutura de dados chamada de "*fields*" (ver seção 4.2.2.1).

Figura 19 - Estrutura lógica de como é feito o endereçamento na memória do GSP.

A figura 20 apresenta o mapa de memória do GSP, o qual encontra-se dividido em 3 regiões:

- Vetores de Interrupção;
- Registradores de Entrada Saída;
- Uso geral.

Figura 20 - Mapa de memória do GSP.

A comunicação com a memória é feita através de um duto de dados de 16 bits, e a cada acesso à memória, o GSP lê ou grava uma palavra (16 bits). O endereço de cada palavra tem os 4 bits menos significativos (*BmS*) iguais a zero. Estes 4 *BmS*s são usados para o endereçamento dos bits dentro da palavra.

Exemplo:

endereço de memória	Memória	Endereço de cada Bit	
		LSB	MSB
FFFF0000H	1ª Palavra	(FFFF0000 até FFFF000F)	
FFFF0010H	2ª Palavra	(FFFF0010 até FFFF001F)	
FFFF0020H	3ª Palavra	(FFFF0020 até FFFF002F)	
FFFF0030H	4ª Palavra	(FFFF0030 até FFFF003F)	
:	:	:	:
FFFF00E0H	15ª Palavra	(FFFF00E0 até FFFF00EF)	
FFFF00F0H	16ª Palavra	(FFFF00F0 até FFFF00FF)	
FFFF0100H	17ª Palavra	(FFFF0100 até FFFF010F)	

4.2.2 - Tipos de Dados Suportados em Hardware.

O TMS34010 suporta as seguintes estruturas de dados a nível de instruções: "fields", "bytes", pixels, e matrizes de pixels. Nas próximas subsecções estas estruturas serão definidas.

4.2.2.1 - "Fields"

São estruturas de dados configuráveis, cujo comprimento pode ser definido entre 1 e 32 bits. No TMS34010 podem ser configurados dois "fields" simultaneamente. A instrução SETF define o "field" (ver [TEX_86a]). Um "field" pode começar e terminar em qualquer endereço de bit. Exemplificando tome-se o seguinte trecho de programa:

```
SETF  5,0,1      ; Configura o "field" 1 para 5 bits
MOVE  @15503H,A0,1 ; Transfere para A0 os 5 bits que
                  ; se encontram armazenados a partir do endereço 15503H.
```

Considerando que, antes da execução MOVE, o conteúdo do endereço 15500H seja o seguinte:

endereço	Dado
15500H	7770H

então o endereço 15503H aponta para o 4 bit do dado 7770H, e após a execução da instrução MOVE, o registrador A0 estará com o seguinte valor:

A0: 0000000EH

pois o "field" 1 foi configurado para preencher os bits restantes da transferência com zeros.

4.2.2.2 - Bytes

São um caso particular de "fields", sendo que o comprimento é definido em 8 bits, com sinal estendido. Podem ser endereçados a partir de qualquer bit dentro da palavra.

4.2.2.3 - Pixels

O termo pixel possui dois significados no contexto de um sistema gráfico baseado no TMS34010. Fora do GSP, pixel é um "picture element", sendo o menor elemento que pode ser endereçado na superfície do vídeo. Para o GSP, um pixel lógico é uma estrutura de dados suportado pelo seu conjunto de instruções. Os pixels são configuráveis, podendo ter comprimento: 1, 2, 4, 8 ou 16 bits (sempre uma potência de 2). No caso do pixel ser menor que 16 bits, a palavra de memória (16 bits) contém 2 ou mais pixels.

Os pixels devem estar alinhados com a palavra na memória e não devem transpor os limites de uma palavra. Isto é, não podem ter parte de seu valor armazenado em uma palavra, e outra parte em palavra diferente. Como exemplo, seja o endereço FFF0B050H de uma palavra que contém 4 pixels de 4 bits. Assim o endereço de cada pixel na palavra será:

1º pixel: FFF0B050H

2º pixel: FFF0B054H

3º pixel: FFF0B058H

4º pixel: FFF0B05CH

O tamanho do pixel deve ser definido através do registrador de E/S, denominado PSIZE, cujo endereço é C0000150H. Exemplificando-se, pode-se usar o seguinte conjunto

de instruções para estabelecer o tamanho do pixel para 4 bits:

```
SETF 16,0,0      ; "field" 0 é de 16 bits
MOVK 4,A0         ; o tamanho do pixel é armazenado em A0
MOVE  A0,@C0000150H,0 ; armazena o tamanho do pixel no reg. de E/S ; PSIZE
```

4.2.2.3.1 - Mapeamento dos Pixels na Tela

A informação de uma linha horizontal de pixels na tela, normalmente é armazenada em palavras consecutivas na memória. Os pixels de uma palavra são desenhados, iniciando por aquele que ocupa a posição menos significativa na palavra. Dessa forma, este pixel aparece então na posição mais à esquerda da tela, enquanto o pixel que ocupa a posição mais significativa da palavra aparece na posição mais à direita (ver figura 21). Assim, para se desenhar um caracter na tela, este deve ser desenhado de forma invertida na memória.

Figura 21 - Mapeamento de pixels na memória.

4.2.2.3.2 - Métodos de Endereçamento de Pixels

O GSP permite duas formas de endereçamento de pixels:

- **Linear** - especificado pelo seu endereço de memória;
- **XY** - especificado pelas suas coordenadas de tela.

No endereçamento XY, o endereço é armazenado em um registrador de 32 bits de propósito geral, e as componentes X e Y são tratadas como inteiros de 16 bits com sinal. A componente X ocupa os 16 bits menos significativos (BmS) do registrador, e a

componente Y ocupa os 16 bits mais significativos (BMS) do registrador.

O endereçamento XY permite ao projetista de software trabalhar em "alto nível", em vez de ter que mapear o pixel em termos de sua posição interna (memória), o que é a nível de máquina; além disso também facilita as funções de recorte em janelas ("window clipping"). Mas o principal, é que o software pode ser migrado facilmente de uma configuração de tela para outra, pois detalhes como o número de bits por pixel, e o comprimento em X da tela são transparentes para o software.

O TMS34010 automaticamente converte o endereço XY de um pixel para a sua forma linear, em todas as instruções que usam o endereçamento XY. Um ponto relevante a ser levantado é como isto é feito. O GSP utiliza a seguinte fórmula para calcular o endereço linear de um pixel, quando é dado o seu endereço XY:

$$\text{Endereço} = [(Y * \text{"display pitch"}) \text{ OU } (X * \text{tamanho do pixel})] + \text{offset}$$

O termo "*display pitch*", refere-se à diferença entre os endereços de memória de dois pixels situados em posições adjacentes verticalmente. Pode-se também pensar no "*display pitch*" como o número de palavras da memória de vídeo que contém os pixels de uma linha inteira da tela do monitor de vídeo.

Para o endereçamento XY, o "*display pitch*" deve ser uma potência de dois. O endereçamento linear impõe poucas restrições, e em particular, o "*display pitch*" pode ser qualquer valor múltiplo de 16, isto é, com os 4 BmSs iguais a zero.

O cálculo do "pitch" pode ser feito pela expressão:

$$\text{"pitch"} = (n^{\circ} \text{ de pixels na linha}) * (n^{\circ} \text{ de bits/pixel})$$

Por exemplo, se a tela comporta 512 pixels em cada linha, e se o número de bits/pixel é 8, tem-se:

$$\text{"pitch"} = 512 * 8 = 4096 = 1000H$$

Quando se usa o endereçamento XY, o GSP permite configurar a origem da tela no canto superior esquerdo, ou no canto inferior esquerdo da tela. A posição da origem é controlada pelo bit ORG do registrador de E/S, DPYCTL.

4.2.2.4 - Matriz Bidimensional de pixels

O conjunto de instruções do TMS34010 inclui poderosas instruções para

manipulação de conjunto de pixels, chamadas de *PixBlt*, que manipulam matrizes de pixels sobre a tela e a memória. Como *Matriz bidimensional de pixels*, pode-se entender uma área retangular da tela (ou da memória) de largura *DX* e altura *DY*. Este tipo de dado é manipulado por instruções gráficas como a PIXBLT (transfere um bloco de pixels), ou a FILL (preenche um bloco de pixels com uma cor e/ou padrão). Utilizando-se uma única instrução PIXBLT, pode-se mover a matriz de uma área de memória para outra, podendo-se ainda combiná-las através de operações booleanas ou aritméticas. Uma matriz de pixels é especificada em termos de sua altura, largura, "pitch", e endereço inicial. Se o endereço inicial for especificado no formato XY, a matriz será referida como uma matriz em formato XY também. Caso o endereço inicial seja especificado em formato linear, a matriz também será referida neste formato.

A utilização de uma matriz em formato XY tem a vantagem de ser independente do endereço de memória, e de permitir a utilização dos modos de verificação de janela do GSP, que só são realizados com o modo de endereçamento XY. Porém a desvantagem do formato XY reside na necessidade de o "pitch" ser uma potência de dois.

4.2.3 - Registradores do GSP.

Nesta seção serão apresentados os registradores internos do TMS34010.

	31 (MSB)	0(LSB)		31 (MSB)	0(LSB)	
A0			B0	SADDR		SOURCE ADDRESS (PIXBLTS)
A1			B1	SPITCH		SOURCE PITCH
A2			B2	DADDR		DESTINATION ADDRESS (PIXBLTS) AND FILLS)
A3			B3	DPITCH		DESTINATION PITCH
A4			B4	OFFSET		"OFFSET"
A5			B5	WSTART		WINDOW START
A6			B6	WEND		WINDOW END
A7			B7	DY/DX		DELTA Y/ DELTA X
B8			B8	COLOR0		COLOR0 (BINARY PIXBLTS)
B9			B9	COLOR1		COLOR1 (BINARY PIXBLTS, FILL AND DRAW)
B10			B10	Registrador Temporario		Usados para armazenamento de instruções temporárias PIXBLT e FILL
B11			B11	Registrador Temporario		
B12			B12	Registrador Temporario		
B13			B13	Registrador Temporario		
B14			B14	Registrador Temporario		
SP	Ponteiro da Pilha		SP	Ponteiro da Pilha		

Figura 22 - Registradores dos conjuntos A e B.

4.2.3.1 - Registradores de Propósito Geral.

O TMS34010 possui dois conjuntos de registradores de propósito geral - conjunto A e conjunto B - cada um com 15 registradores de 32 bits. Os dois conjuntos compartilham um "stack pointer" (SP) também de 32 bits, o qual gerencia automaticamente a pilha do sistema quando são ativadas as interrupções e quando há chamadas de subrotinas.

A figura 22 traz uma ilustração dos registradores de propósito geral.

O conjunto A pode ser exclusivamente utilizado para armazenamento de dados e sua manipulação. Não é associado a esses registradores nenhuma função dedicada que utilize diretamente o hardware gráfico.

A tabela 4 apresenta uma explicação sucinta dos registradores do conjunto B.

Registrador	Mnemônico	Descrição
B0	SADDR	"Source Address". Endereço do canto superior esquerdo do vetor de pixels origem (do pixel de ordem mais baixa do vetor). SADDR pode estar em format linear ou XY.
B1	SPTCH	"Source Pitch". Diferença linear entre o início de duas linhas adjacentes de um vetor de pixels tomado como origem.
B2	DADDR	"Destination Address". Endereço do canto superior esquerdo do vetor de pixels destino (do pixel de ordem mais baixa do vetor). DADDR pode estar em format linear ou XY.
B3	DPTCH	"Destination Pitch". Diferença linear entre o início de duas linhas adjacentes no vetor de pixels destino.
B4	OFFSET	"Offset". Endereço linear do pixel correspondente a origem da tela (X=0,Y=0).
B5	WSTART	"Window Start Address". Endereço XY do canto superior esquerdo da janela.
B6	WEND	"Window End Address". Endereço XY do canto inferior direito da janela.
B7	DYDX	"Delta Y/Delta X". Os 16 BMSs do registrador especificam a largura (dimensão X) do vetor fonte em termos de pixels ou bits, dependendo da instrução usada. Os 16 BMSs especificam a altura (dimensão Y) do vetor origem. Se DY=0 ou DX=0 nada é movido.
B8	COLOR0	Valor do pixel correspondente a "cor 0". COLOR0 contém a cor de fundo fonte a ser usada durante uma operação de bit-expansão (PIXBLT B,XY ou PIXBLT B,L).
B9	COLOR1	Valor do pixel correspondente a "cor 1". COLOR1 contém a cor de frente fonte a ser usada durante uma operação de bit-expansão, "fill", ou "draw".
B10-B14		Registradores temporários para a instrução PixBlt.
SP	SP	Stack Pointer. SP contém o endereço do bit que está no topo da pilha.

Tabela 4 - Descrição dos registradores do conjunto B.

4.2.3.2 - Registrador de "Status".

O registrador de "Status" (ST), é um registrador especial de 32 bits, que especifica o "status" do processador. O ST também contém diversos parâmetros que especificam as características dos dois tipos de dados programáveis, "field" 0 e "field" 1.

A figura 23 ilustra o registrador de "status", e na tabela 5 são listadas as funções associadas aos campos do ST.

Nº do Bit	Nome do Campo	Função
0-4	FS0	<i>Field Size 0</i> . Comprimento em bits do primeiro campo ("field") de dados na memória.
5	FE0	<i>Field Extend 0</i> . Este bit determina se o campo na memória é estendido com zeros, ou com o bit de sinal quando o campo é armazenado em um registrador de propósito geral de 32 bits. (FE0=0 ⇒ Extensão com Zeros, FE0=1 ⇒ Extensão pelo sinal)
6-10	FS1	<i>Field Size 1</i> . Comprimento em bits do segundo campo ("field") de dados na memória.
11	FE1	<i>Field Extend 1</i> . Este bit determina se o campo na memória é estendido com zeros, ou com o bit de sinal quando o campo é armazenado em um registrador de propósito geral de 32 bits. (FE1=0 ⇒ Extensão com Zeros, FE1=1 ⇒ Extensão pelo sinal)
12-20	-	<i>Reservado.</i>
21	IE	<i>Interrupt Enable</i> . Principal bit para habilitar/desabilitar interrupção. IE=0 ⇒ Todas as interrupções mascaráveis desabilitadas. IE=1 ⇒ Todas as interrupções mascaráveis habilitadas.
22-24	-	<i>Reservado.</i>
25	PBX	<i>Escutando PixBlt</i> . Indica no retorno de uma interrupção, se a interrupção ocorreu entre instruções, ou se durante uma instrução PIXBLT ou FILL. 0 = int. ocorren fora dos limites de uma instrução PIXBLT ou FILL. 1 = interrupção ocorreu durante uma instrução PIXBLT ou FILL.
26-27	-	<i>Reservado.</i>
28	V	<i>Overflow</i> . O bit é estabelecido conforme a execução da instrução.
29	Z	<i>Zero</i> . O bit é estabelecido conforme a execução da instrução.
30	C	<i>Carry</i> . O bit é estabelecido conforme a execução da instrução.
31	N	<i>Negativo</i> . O bit é estabelecido conforme a execução da instrução.

Tabela 5 - Definição dos campos do registrador de "status".

Figura 23 - Registrador de "status" do GSP.

4.2.3.3 - Registradores de Entrada/Saída (E/S).

Os registradores de entrada/saída do TMS34010 ocupam os endereços C0000000H até C00001FF. Estes registradores podem ser acessados diretamente pelo GSP, podendo também serem acessados pelo processador hospedeiro através dos registradores que compõem a interface com o hospedeiro [TEX_86a]. A figura 24 ilustra o mapa de memória dos registradores de E/S.

>0000 01FD	REFCNT	DRAM REFRESH COUNT
>0000 01ED	DPYADR	DISPLAY ADDRESS
>0000 01DD	VCOUNT	VERTICAL COUNT
>0000 01CD	HCCOUNT	HORIZONTAL COUNT
>0000 01BD	DPYTAP	DISPLAY TAP POINT
>0000 A1AD	RESERVED	
>0000 019D		
>0000 018D		
>0000 017D		
>0000 016D	PMASK	PLANE MASK
>0000 015D	PSIZE	PIXEL SIZE
>0000 014D	CONVDP	CONVERSION (DESTINATION PITCH)
>0000 013D	CONVSP	CONVERSION (SOURCE PITCH)
>0000 012D	INTPEND	INTERRUPT PENDING
>0000 011D	INTENB	INTERRUPT ENABLE
>0000 010D	HSTCTLH	HOST CONTROL (8 MSB'S)
>0000 00FD	HSTCTL	HOST CONTROL (8 LSB'S)
>0000 00ED	HSTADRH	HOST ADDRESS (16 MSB'S)
>0000 00DD	HSTADRL	HOST ADDRESS (16 LSB'S)
>0000 00CD	HSTDATA	HOST DATA
>0000 00BD	CONTROL	CONTROL
>0000 00AD	DPYNT	DISPLAY INTERRUPT
>0000 009D	DPYSTRT	DISPLAY START
>0000 008D	DPYCTL	DISPLAY CONTROL
>0000 007D	VTOTAL	VIDEO TOTAL
>0000 006D	VSBLNK	VERTICAL START BLANK
>0000 005D	VEBLNK	VERTICAL END BLANK
>0000 004D	VESYNC	VERTICAL END SYNC
>0000 003D	HTOTAL	HORIZONTAL TOTAL
>0000 002D	HSBLNK	HORIZONTAL START BLANK
>0000 001D	HEBLNK	HORIZONTAL END BLANK
>0000 000D	HESYNC	HORIZONTAL END SYNC

Figura 24 - Mapa de memória dos registradores de E/S.

4.2.3.4 - Registradores de Interface com o Hospedeiro.

Os cinco seguintes registradores de E/S são dedicados à comunicação com o hospedeiro:

HSTADRH: (Host interface Address, high word) Contém os 16 bits mais significativos (BMS) de um endereço de 32 bits usado pelo hospedeiro para acessar indiretamente a memória local do GSP.

HSTADRL: (Host interface Address, low word) Contém os 16 BmSs de um

endereço de 32 bits usado pelo hospedeiro para acessar indiretamente a memória local do GSP.

HSTCTLH: (Host interface Control, high byte) Contém 7 bits programáveis que controlam as seguintes funções de comunicação com o hospedeiro:

NMI (bit 8) - Interrupção não mascarável

NMIM (bit 9) - NMI modo bit

INCW (bit 11) - Incrementa o endereço durante escrita

INCR (bit 12) - Incrementa o endereço durante leitura

LBL (bit 13) - "lower byte last"

CF (bit 14) - descarrega a "cache"

HLT (bit 15) - Pára ("halt") a execução do TMS34010

O bit 10 é reservado.

HSTCTL: (Host interface Control, low byte), os bits 0 a 7 são reservados.

HSTDATA: (Host interface data) Contém o dado que será gravado ou lido na memória local.

Através destes registradores o TMS34010 pode efetuar as seguintes operações:

- Transferir diretamente mensagens de comando ou de "status";
- Transferir indiretamente blocos de dados de/para a memória local;
- Receber pedidos de interrupção do hospedeiro;
- Transmitir pedidos de interrupção para o hospedeiro.

A capacidade para transferir grandes blocos de dados torna a interface com o hospedeiro extremamente flexível. Por exemplo, o hospedeiro pode transferir blocos de comandos para o GSP, pode parar ("halt") o GSP temporariamente para carregar um novo programa para o GSP executar, ou pode ler blocos de dados gráficos gerados pelo GSP.

Os registradores de interface com o hospedeiro são mapeados em 4 posições consecutivas de 16 bits, no espaço de endereçamento de memória ou de E/S do hospedeiro. Neste caso, os registradores HSTCTL e HSTCTLH são acessados pelo hospedeiro, como os 8 BmSs e os 8 BMSs respectivamente, de uma única posição de memória.

O registrador HSTCTL (host control) controla funções tal como a da transferência de pedidos de interrupção. Estas solicitações são tipicamente usadas pelo software para coordenar a transferência de blocos de dados para a memória local do GSP. O HSTCTL

também permite ao hospedeiro descarregar a "cache" do GSP, parar ("halt") a execução no GSP, e transmitir pedidos de interrupções não mascaráveis para o GSP.

O hospedeiro utiliza os outros três registradores para acessar indiretamente blocos de dados dentro da memória local do GSP. Os registradores HSTADRL e HSTADRH armazenam o endereço (32 bits) que indica a localização da palavra na memória do GSP, e o registrador HSTDATA armazena o dado a ser transferido de/para a memória local, sob controle do hospedeiro. A interface com o hospedeiro pode ser programada para automaticamente incrementar o endereço após cada transferência, fazendo com que o hospedeiro possa acessar rapidamente um bloco de dados sequenciais. Esta técnica é usualmente chamada de Acesso Direto à Memória (do inglês - DMA).

4.2.3.5 - Registradores Dedicados ao Controle da Memória Local.

Seis dos registradores de E/S são dedicados ao controle da memória local do GSP. São eles:

- **CONTROL** : contém parâmetros que controlam a operação da interface com a memória local.
 - RM (bit 2) - modo de refrescamento da DRAM
 - RR (bits 3-4) - taxa de refrescamento da DRAM
 - T (bit 5) - habilita transparência
 - W (bits 6-7) - modo de detecção de violação de janela
 - PBH (bit 8) - PixBlt em direção horizontal
 - PBV (bit 9) - PixBlt em direção vertical
 - PPOP (bits 10-14) - operação de processamento de pixel
 - CD (bit 15) - desabilita a "cache"
 - Bits 0 e 1 são reservados.
- **CONVDP** : fator de conversão do "pitch" destino. É usado durante conversão de endereçamento XY para linear, em um endereço destino.
- **CONVSP** : fator de conversão do "pitch" origem. É usado durante conversão de endereçamento XY para linear, em um endereço origem.
- **PMASK** : registrador de máscara de planos. Habilita ou não os vários planos de

memória da imagem, para pixels contituídos por vários bits.

- **PSIZE** : registrador de tamanho de pixel. Especifica o tamanho do pixel (em bits). Os tamanhos possíveis são: 1, 2, 4, 8 e 16 bits.
- **REFCNT** : contador para controle do refrescamento de memórias dinâmicas (DRAM).

4.3 - Instruções Gráficas

Conforme já mencionado anteriormente, o TMS34010 possui instruções específicas para a manipulação de pixels. Eles podem ser indicados através de seu endereço em formato linear ou XY. Serão apresentadas a seguir alguns detalhes sobre as instruções e operações gráficas do GSP.

4.3.1 - Instrução FILL

Esta instrução preenche uma matriz de pixels com o valor especificado no registrador COLOR1. A matriz pode ser definida tanto no formato linear quanto no XY (FILL L ou FILL XY).

Os parâmetros necessários a sua execução devem estar previamente armazenados no registradores que são utilizados por esta instrução (ver [TEX_86a]). Durante a sua execução podem ser feitas operações de processamento de pixels, verificação de janelas, máscara de planos e transparência.

4.3.2 - Instrução PIXBLT

O conjunto de operações "raster", chamadas *PixBlt* (pixel Block Transfer), são utilizadas para manipulação de matrizes bidimensionais de bits ou pixels. A matriz de pixels é definida pelos seguintes parâmetros:

- O endereço inicial da matriz (o endereço do pixel com o endereço mais baixo);
- A largura DX (o nº de pixels por linha);
- A altura DY (o nº de linhas de pixels);

- O "pitch" da matriz (a diferença entre o início de duas linhas sucessivas)

Dois operandos devem ser especificados numa instrução **PIXBLT**, a matriz de pixels fonte e a matriz de pixels destino. Ambas devem ter a mesma largura e o mesmo comprimento, embora possam ter "pitches" diferentes. Esta instrução combina cada pixel da matriz fonte com o pixel correspondente da matriz destino, utilizando as operações gráficas selecionadas previamente.

Em algumas das instruções **PIXBLT** pode-se especificar qual o canto da tela que se deseja tomar como origem. Assim, a seleção é feita pelos bits **PBV** e **PBH** do registrador **CONTROL**.

Tanto para as instruções **PIXBLT** quanto para as **FILL**, o tempo de execução, depende do tamanho da matriz de pixels utilizada. Como isso poderia vir a causar atraso ao atendimento de instruções de alta prioridade, essas instruções verificam a ocorrência de interrupções a intervalos regulares durante sua execução.

4.3.3 - Instrução **PIXT**

No **TMS34010** as instruções **PIXT** (*pixel transfer*) podem ser utilizadas para transferir pixels individuais de uma localização para outra. Esta transferência pode ser feita a nível de registrador para memória, de memória para registrador, ou mesmo de memória para memória.

O endereço do pixel pode ser especificado tanto em formato linear quanto em formato **XY**.

A instrução **PIXT** pode ser usada juntamente com verificação de violação de janela, com operações de processamento de pixels booleanas ou aritméticas, máscara de planos e transparência.

4.3.4 - Instrução **DRAV**

A instrução **DRAV** é usada principalmente para a implementação de algoritmos incrementais, para desenhar círculos, elipses, arcos e outras curvas. Esta instrução desenha um pixel para a posição apontada por um registrador, o ponteiro (no registrador)

é então incrementado para apontar para o próximo pixel. O valor do pixel é obtido do registrador COLOR1.

Esta instrução pode ser utilizada juntamente com as operações gráficas (booleanas, aritméticas, verificação de janelas, máscara de planos e transparência).

4.3.5 - Instrução LINE

Para o desenho de linhas retas pode-se usar a instrução DRAV, mas este caso pode ser tratado com vantagem no TMS34010 através da instrução LINE. Uma instrução LINE implementa o "loop" interno do algoritmo de Bresenham para desenhar linhas.

Assim como a instrução DRAV, apresentada acima, a instrução LINE também pode ser usada em conjunto com as operações gráficas.

4.3.6 - Instrução CPW

Esta instrução compara um pixel (ponto) com os limites de uma janela, utilizando apenas um estado de máquina para comparar as coordenadas XY do ponto com todos os quatro cantos da janela.

Com esta instrução pode-se verificar se uma linha (definida por dois pontos) está inteiramente dentro ou inteiramente fora de uma janela. Linhas inteiramente fora podem ser rejeitadas, significando que não mais consumirão tempo de processamento.

4.4 - Operações Gráficas.

O TMS34010 provê diversos métodos para processar os valores fonte e destino de pixels antes que o resultado seja escrito no destino. Estas operações incluem:

- Operações de processamento de pixel aritméticas e booleanas;
- Uma máscara de planos que especifica quais os bits dos pixels que serão alterados durante as operações.
- Transparência, é uma opção que permite que objetos escritos sobre a tela tenham regiões (transparentes), em que os objetos que estão atrás de outros possam

ser visualizados.

As operações de processamento de pixels, máscara de planos, e transparência podem ser usadas simultaneamente, podendo também serem combinadas com qualquer das instruções que desenham pixels descritas nas seções anteriores.

O TMS34010 possui características tais como a verificação automática de janelas, suportando muito bem a construção de ambientes gráficos baseados em janelas. Há 3 formas de realizar essa verificação:

- Recortando uma imagem, para que ela se restrinja a janela;
- Solicitando uma interrupção, se foi feita uma tentativa para escrever um pixel fora de uma janela;
- Solicitando uma interrupção, se foi feita uma tentativa para escrever um pixel dentro de uma janela;

4.4.1 - Operação de Expansão de Cor.

O TMS34010 permite, através do uso das instruções PIXBLT B,XY e PIXBLT B,L, a realização da operação de expansão de cor. Nesta operação, a forma da imagem pode ser armazenada separadamente dos atributos de cor e intensidade. Esta forma pode ser armazenada de maneira comprimida através de um mapa de bits (um bit por pixel) contendo "1s" e "0s", em uma memória que não seja a de vídeo.

A operação de expansão de cor adiciona cor à forma no momento em que a mesma é desenhada na memória de vídeo. Neste caso, o registrador COLOR0 contém a cor que será atribuída ao bit "0", enquanto que o registrador COLOR1 contém a cor que será atribuída ao bit "1".

Este recurso possui utilidade na definição dos caracteres para texto, e em aplicações que gerem dinamicamente, formas ou padrões. O benefício imediato deste recurso está na redução da área necessária para armazenar estes caracteres ou formas.

4.4.2 - Operações com Janelas.

Uma *janela* é uma região retangular da memória de vídeo, que é especificada em

termos das coordenadas XY dos pixels de seus dois cantos extremos (mínimo X e Y, e máximo X e Y). Os pixels do canto são considerados como pertencentes a janela. As operações de verificação de janelas são possíveis apenas com o modo de endereçamento XY, isto é, o endereço do pixel destino deve estar especificado no formato XY.

Os registradores B5 (*wstart*) e B6 (*wend*) especificam a janela. Há 4 modos de se efetuar a verificação de janelas. A seleção de um modo é feita através do campo W, do registrador CONTROL, da seguinte forma:

W=0 : a verificação de janela é desabilitada;

W=1 : é feito um pedido de interrupção, na tentativa de se escrever dentro da janela;

W=2 : é feito um pedido de interrupção, na tentativa de se escrever fora da janela;

W=3 : escreve somente os pixels que estejam dentro da janela;

Os modos de verificação de janela afetam somente a escrita de pixels, realizada com as instruções gráficas PIXBLT, FILL, LINE, DRAB e PIXT.

Uma operação freqüente em ambientes com janelas, é o cálculo da região onde dois retângulos se interceptam. Essa operação é possível com as instruções PIXBLT e FILL. Através de verificação de janelas, pode-se realizar esta operação de intersecção de duas formas:

1. a matriz destino é recortada ("preclipped" numa janela retangular, antes que a instrução PIXBLT (ou FILL) inicie.
2. a intersecção da matriz destino com a janela retangular é calculada, mas os pixels não são transferidos.

4.4.3 - Operações de Processamento de pixels.

O TMS34010 possui 16 operações booleanas de processamento de pixels, e 6 operações aritméticas. As operações booleanas são executadas bit a bit, e os pixels podem ser de 1, 2, 4, 8 ou 16 bits. Nas operações aritméticas os pixels podem ser de 4, 8 ou 16 bits e são tratados como inteiros com complemento de 2.

PPOP	Operação
00000	Fonte → Destino
00001	Fonte <i>AND</i> Destino → Destino
00010	Fonte <i>AND</i> ~Destino → Destino
00011	0s → Destino
00100	Fonte <i>OR</i> ~Destino → Destino
00101	Fonte <i>XOR</i> Destino → Destino
00110	~Destino → Destino
00111	Fonte <i>NOR</i> Destino → Destino
01000	Fonte <i>OR</i> Destino → Destino
01001	Destino → Destino
01010	Fonte <i>XOR</i> Destino → Destino
01011	~Fonte <i>AND</i> Destino → Destino
01100	1s → Destino
01101	~Fonte <i>OR</i> Destino → Destino
01110	Fonte <i>NAND</i> Destino → Destino
01111	~Fonte → Destino

Tabela 6 - Opções de processamento booleano de pixels.

A operação de processamento de pixel é selecionada no campo **PPOP** do registrador **CONTROL**. Quando um pixel é lido (fonte), ele é logicamente ou aritmeticamente combinado com o pixel destino correspondente, de acordo com a operação em **PPOP**, e o resultado é escrito no pixel destino. As tabelas 6 e 7 listam os 22 códigos possíveis em **PPOP** e seus significados.

PPOP	Operação
10000	Fonte + Destino → Destino
10001	ADDS ¹ (Fonte, Destino) → Destino
10010	Destino - Fonte → Destino
10011	SUBS ² (Fonte, Destino) → Destino
10100	MAX (Fonte, Destino) → Destino
10101	MIN (Fonte, Destino) → Destino
10110 - 11111	Reservado

Tabela 7 - Opções de processamento aritmético (ou cor) de pixels.

4.4.4 - Máscara de Planos.

A máscara de planos é um mecanismo em hardware que protege determinados bits dos pixels. A memória de vídeo é organizada em *planos de bits* (ou planos de cores), e cada bit de um pixel é associado a um plano, ou seja, o número de planos é igual ao número de bits do pixel (ver figura 3).

Através da máscara de planos pode-se manipular um ou mais planos independentemente dos demais. Por exemplo, se uma memória de vídeo possui quatro planos, três deles podem ser dedicados a manipulação de gráficos em 8 cores, enquanto o quarto plano pode ser usado para sobrepor texto em uma cor. Pode-se estabelecer a máscara de planos para que o texto seja modificado sem afetar os gráficos e vice-versa.

O registrador **PMASK** armazena a máscara de planos. Cada bit da máscara de planos corresponde ao bit da mesma posição no pixel. Os bits *1s* especificam que o bit correspondente (plano de bits) está protegido, e os bits *0s* especificam que os bits correspondentes (plano de bits) podem ser modificados.

A máscara de planos exerce influência somente durante as instruções gráficas **PIXBLT**, **FILL**, **PIXT**, **DRAV** e **LINE**.

¹ ADDS - adiciona com saturação. Quando ocorre overflow, o resultado é substituído por 1s.

² SUBS - subtrai com saturação. Quando ocorre overflow o resultado é substituído por 0s.

4.5 - Ferramentas para Desenvolvimento de Software.

O TMS34010 dispõe de um conjunto de ferramentas, executáveis no hospedeiro (neste trabalho, o IBM-PC) para desenvolvimento de software, o qual é listado a seguir:

- Assembler
- Archiver
- Linker
- Utilitário para conversão de código
- Simulador
- Programa para Depuração de Código - SDB340
- Compilador C
- Loader SDBL

As ferramentas da linguagem montadora para o TMS34010, criam e usam arquivos objeto que estejam em "*Common Object File Format*" ou *COFF*. Neste formato, os arquivos objeto contém blocos separados (seções) de código e dado, que podem ser carregados em diferentes espaços de memória.

4.5.1 - "Assembler"

O assembler (*GSPA*) converte os arquivos fonte em linguagem montadora, para arquivos objeto em linguagem de máquina. Os arquivos fonte podem conter instruções, diretivos do Assembler, e diretivos macro. Pode-se usar os diretivos do Assembler para controlar vários aspectos do processo de montagem, tais como a formato da listagem, alinhamento dos dados, e como o código fonte é colocado nas seções.

4.5.2 - "Archiver"

Permite a criação de bibliotecas. Por exemplo, através do *GSPAR* pode-se criar uma biblioteca de macros, e neste caso, o Assembler pesquisa a biblioteca para buscar os macros que foram referenciados no arquivo fonte. Pode-se criar também bibliotecas com arquivos objeto, nesse caso as referências externas serão buscadas na biblioteca durante a execução do Linker.

4.5.3 - "Linker"

O Linker (*GSPLNK*) combina vários arquivos objeto em um único arquivo executável. Quando o módulo executável é criado, é efetuada a relocação e determinação das referências externas. O Linker utiliza como entrada os arquivos objeto em formato COFF criados pelo Assembler.

Os diretivos do Linker permitem que se combinem seções do arquivo objeto, símbolos para um endereço específico ou internos a regiões específicas da memória do TMS34010, e permitindo também definir ou redefinir símbolos globais.

4.5.6 - Utilitário para Conversão de Código

A função deste utilitário é converter um arquivo objeto COFF, para o formato hexadecimal da INTEL ou TEKTRONIX, pois a maioria dos programadores de EPROM não aceitam arquivos objeto COFF como entrada. O arquivo convertido pode ser então utilizado pelo programador de EPROM.

4.5.7 - Simulador

O simulador (*GSPSIM*) é uma ferramenta de depuração, que simula as funções do TMS34010 em um ambiente gráfico configurável. O simulador permite projetar, implementar, e avaliar tanto sistema gráficos quanto sistema não gráficos.

A tela de visualização e o conjunto de comandos do simulador mostra e permite controlar informações acerca do "status" da máquina, oferecendo recursos para que o usuário controle o programa em desenvolvimento.

O simulador apresenta dois conjuntos separados de informação:

1. O estado da máquina, contendo os valores dos registradores dos arquivos A e B (ou os de E/S), o valor do PC, SP, flags, registrador de "status", o estado da "cache", o tamanho do pixel, etc. A figura 25 ilustra esta apresentação.
2. O ambiente gráfico, que permite a visualização de figuras ou imagens.

Figura 25 - Tela apresentada pelo simulador.

O simulador contém um conjunto de comandos que possibilitam:


```

GSP      Register and Machine Status   S/W Simulator   fs 16/32      PS= 0  PM=0000
fe 0/0  w=off pp= S -> D

Reg File A                                Reg File B
A0 00000000    A8 00000001    B0 00000000 saddr    B8 FFFFFFFF color0
A1 00000000    A9 000000F1    B1 00001000 sptch    B9 00000000 color1
A2 00000000    A10 00000001    B2 01000100 daddr    B10 00000000 temp_x
A3 00000000    A11 000FFFFF1    B3 00001000 dptch    B11 00000000 temp_y
A4 00000000    A10 00000002    B4 00000800 offset    B12 FFFFFFFF tempda
A5 00000000    A13 0000000EE    B5 00000000 watart    B13 00000000 tempst
A6 00000000    A14 00000003    B6 01FF0200 wend      B14 00000000 tempct
A7 00000000    SP FFFFF000    B7 00000000 dydx

Normal stop mode.                        Cache miss      Clk= 4
st 00000010    NCZV=0100      ITPVH=00000    SP=FFFFFF00    Ctl=0000
pc 00000000    0000 INVALID OP      ; INVALID OP

Command[0] HELP

```

- a alteração dos registradores e "flags", individualmente ou por bloco;
- a alteração ou visualização do conteúdo de uma posição ou de um bloco de memória;
- estabelecer "breakpoints";
- executar o programa passo a passo, ou por grupo de instruções, ou o programa completo;
- modificar parâmetros como o tamanho do pixel, o estado da cache, o tamanho do "field", etc, de forma a alterar o estado da máquina.

4.5.8 - Programa para Depuração de Código - SDB340.

O TMS34010 possui ainda uma ferramenta para depuração de programas chamada de SDB340 (*software Development Board*), a qual deve ser utilizada com uma placa gráfica que contenha o TMS34010.

A utilização desta ferramenta é parecida com a do simulador, pois é dotada por um conjunto de comandos semelhantes, diferenciando-se dele por efetivamente acessar o GSP.

4.5.9 - Compilador para a Linguagem "C".

A Texas Instruments, fornece um compilador "C", que está em uma posição híbrida entre a versão 1 (Kerningham e Ritchie) [KER_78] e a versão ANSI [KER_88] [SCD_89].

O compilador C é composto por quatro partes:

- o pré-processador para a linguagem C, *GSPCPP*;
- o "parser", *GSPCC*;
- o gerador de código intermediário, *GSPCG*;
- o "assembler", *GSPA*.

Após serem efetuados todos os passos de compilação de um arquivo, obtém-se um código objeto que deverá ser "link"-editado com as bibliotecas pertinentes.

4.5.10 - Carregador SDBL.

Esta ferramenta possibilita o carregamento de um programa objeto na memória local do GSP. A transferência do programa do hospedeiro para o GSP é realizado através dos registradores que compõem a interface com o hospedeiro.

Esta ferramenta foi estendida para suportar o subsistema de comunicação com o hospedeiro desenvolvido (ver seção 6.3.3).

4.6 - Especificações da Arquitetura Desenvolvida.

A arquitetura dedicada desenvolvida [PAI_90] é constituída pelos seguintes elementos:

- o processador gráfico TMS34010;
- memória de sistema, implementada com RAMs dinâmicas;
- memória de vídeo, implementada com VRAMs;
- circuito para controle da temporização do vídeo;
- conversor D/A de 8 bits, utilizado na geração do sinal de vídeo, com os 256 níveis de cinza.

A figura 26 apresenta o diagrama em blocos desta arquitetura.

Figura 26 - Diagrama em blocos da arquitetura dedicada utilizada.

As características da arquitetura são as seguintes:

- Frequência externa do TMS34010: 20 MHz;
- Frequência do pixel: 10 MHz;
- Resolução da tela: 256 linhas com 512 pixels, e pixel de 8 bits;
- Interface Serial RS232-C: para conexão de um "mouse" com taxa de transmissão de 1200 Bauds, 1 "stop bit", sem paridade;
- Memória de Sistema (DRAM): 2 x 128 Kbytes;
- Memória de Vídeo (VRAM): 3 x 128 Kbytes.

Capítulo V

Ferramentas de Apoio ao Desenvolvimento de Interfaces com o Usuário

5.1 - Introdução.

Os avanços na tecnologia de equipamentos computacionais vêm possibilitando a construção de terminais gráficos cada vez mais poderosos, e dessa forma os programas para controlá-los tornam-se também cada vez mais sofisticados, pois precisam acompanhar esta evolução. Atualmente, com a crescente tendência de utilização de sistemas interativos, suas interfaces Homem/Máquina (H/M) vem causando uma verdadeira revolução em termos de demanda de novas técnicas para desenvolvimento desses programas [HIX_89].

O processo de construção de um sistema de software por si só é uma tarefa complexa. Quando trata-se de um sistema interativo - que precisa de uma interface Homem/Máquina - o sistema não é julgado apenas pela sua capacidade de cumprir a tarefa para a qual ele foi construído, mas também pela sua característica de se comunicar com o usuário. De fato, se os usuários não conseguirem efetivamente manipular um sistema interativo, ou a capacidade computacional do sistema pode não ser plenamente utilizada, ou ele se tornará ineficiente.

O desenvolvimento de interfaces H/M é bastante diferente do desenvolvimento de software convencional. Os métodos tradicionais, técnicas e ferramentas normalmente utilizadas para desenvolvimento de software geralmente não provêm um auxílio eficiente para a construção de interfaces [HIX_89]. Assim, um campo aberto à pesquisa é o de tecnologia para a construção de interfaces H/M que sejam: eficientes; tenham um elevado grau de "amigabilidade com o usuário"; e sejam independentes da máquina para o qual foram construídos (portabilidade). Com esses três objetivos a serem alcançados, muitos pesquisadores têm criado padrões de como uma interface com o usuário deve ser (em

termos de aparência), quais os componentes utilizados na construção de interfaces, e mesmo em ferramentas que auxiliem ao projetista de software a construir suas interfaces.

Nas próximas seções será apresentada uma classificação dessas ferramentas, bem como algumas ferramentas de auxílio a construção de interfaces H/M.

5.2 - Classificação das Ferramentas para a Construção de Interfaces com o Usuário.

Segundo Myers [MYE_89], existem dois grandes grupos de ferramentas que apoiam a construção de interface com o usuário: *"toolkits" para Interface com o Usuário* e *Sistemas de Desenvolvimento de Interface com o Usuário*. A seguir é apresentado um resumo sobre eles.

- ***"Toolkits" para Interface com o Usuário***, são bibliotecas que oferecem técnicas de interação, onde técnica de interação é um modo de se utilizar um dispositivo físico de entrada (tal como "mouse", teclado, mesa digitalizadora, ou "rotary knob") para receber um dado, o qual pode ser um comando, número, nome, etc, criando um processo de realimentação de informações sobre a tela. Exemplos de técnicas de interação são "menus" e utilização de "scroll bars". Um programador então utiliza o "toolkit" para escrever programas, o qual organiza tais técnicas de interação, de forma a se obter softwares "amigáveis". Deve-se notar que a qualidade do software em termos de "amigabilidade" é de responsabilidade toda do programador, pois os "toolkits" não oferecem qualquer garantia nem facilidade de desenvolvimento ao programador. Uma desvantagem da utilização de "toolkits", é que eles oferecem estilos de interação limitados e são difíceis de se utilizar sem algum auxílio [LEE_89]. Exemplos de "toolkits": Macintosh Toolbox, o Motif da Open Software Foundation [BAR_91], o XView e o X.11 Toolkit para o Sistema de Gerenciamento de Janelas X Window [HEL_90] [JOH_90].

- Um ***Sistema de Desenvolvimento de Interface com o Usuário - SDIU***, é um conjunto integrado de ferramentas que auxiliam o programador a criar e gerenciar muitos aspectos das interfaces. Os SDIUs auxiliam tanto na fase de projeto quanto na de implementação de interfaces. Também devem manipular todos os aspectos

da interface, o que inclui todos os ítems visíveis na tela e tudo o que se refira ao diálogo entre o usuário e a aplicação. Dessa forma os SDIUs devem:

- Gerenciar o "mouse" e os outros dispositivos de entrada,
- Validar os dados fornecidos pelo usuário,
- Realizar tratamento de erros que o usuário cometer,
- Efetuar um processo de realimentação para mostrar que as solicitações de entrada foram recebidas,
- Prover auxílio ao usuário, em forma de explicação e exemplos em todas as fases,
- Atualizar a tela quando as saídas geradas pelo aplicativo são alteradas,
- Permitir ao usuário personalizar seu aplicativo.

Para efetuar tudo isso, os SDIUs geralmente contam com:

- um "toolkit";
- um componente que efetua o controle de diálogo entre o sistema e o usuário, o qual trata os eventos criados;
- um ambiente de programação que orienta a construção do código da interface;
- um editor gráfico baseado em "mouse" para especificar a localização dos elementos gráficos, (um exemplo é o utilitário *devguide* da SUN Inc. para o SDIU OpenWindows [SUN_89b]);
- um módulo de análise que avalia a interface construída automaticamente, tomando como base uma série de regras e diretrizes, ou mesmo que guarde a sequência de solicitações feitas pelo usuário ao sistema para posterior avaliação pelo projetista da interface.

As considerações feitas anteriormente indicam que existe uma ligação muito estreita entre um "toolkit" e um SDIU. A figura 27 mostra os vários níveis de um software que disponha de interface com o usuário. Essa figura sugere algumas regras entre estes níveis. Por exemplo, um programa de aplicação tem acesso a todos os níveis de programação isto é, o programador pode explorar tudo o que é oferecido em cada nível.

Figura 27 - Níveis de um software que dispõe de interface com o usuário.

5.3 - Ferramentas de Auxílio à Construção da Comunicação Homem/Máquina.

Nesta seção serão consideradas ferramentas (que por sua vez também são sistemas de software) que destinam-se a facilitar a construção de programas auxiliando a implementar, ou mesmo executando por si só, a comunicação do software com as pessoas que o utilizam.

5.3.1 - Sistemas de Gerenciamento de Janelas.

Um **Sistema de Gerenciamento de Janelas (SGJ)** é um software que auxilia o usuário a monitorar e controlar os diferentes contextos dos programas aplicativos, separando-os fisicamente em diferentes regiões de uma ou mais telas do computador [MYE_88]. A estas regiões dá-se usualmente o nome de *Janelas*. Estas janelas podem conter diversos tipos de informações, principalmente textuais ou gráficas.

Em sistemas computacionais que oferecem processamento multi-tarefa, os Gerenciadores de Janelas tornaram-se populares principalmente porque eles permitem que várias atividades logicamente separadas sejam colocadas simultaneamente em diferentes regiões da tela do terminal. Isso atende a necessidades reais, pois durante grande parte

do tempo a atenção de um usuário de computação está deslocando-se de uma atividade para outra, tais como editar vários arquivos de uma vez para compor outro e depois compilá-lo. Enquanto o arquivo é compilado, o usuário pode ler suas mensagens de correio, verificar o tempo dispendido com seu processo, ou seja, realizar outras tarefas enquanto espera que o computador finalize alguma outra por ele solicitada. Utilizando-se sistemas de janelas estas atividades podem ser efetuadas de forma a exigir menos esforço do usuário.

Apesar de janelas em vídeo já existirem anteriormente, o conceito de Sistema de Gerenciamento de Janelas, compartilhando vídeo e teclado, como um paradigma de interface com o usuário foi efetivamente implantado com o sucesso do computador MacIntosh [WIL_84], muito embora os conceitos envolvidos tenham sido utilizados também por outros fabricantes, como por exemplo o sistema "Star" da Xerox [JOH_89].

Outro fator relevante é que, com o avanço da Ciência de Computação e sua utilização em várias áreas, tais como engenharia, física, medicina, agricultura, biblioteconomia, advocacia, etc, tornou-se de grande importância que usuários não treinados em computação pudessem utilizar os sistemas computacionais facilmente. Dessa forma a construção de "softwares" que utilizem interfaces "amigáveis" com o usuário tornou-se uma premissa para projetistas de tais sistemas. A utilização de janelas, "menus", ícones (que podem ser vistos como um caso especial de janelas), entradas interativas através de "mouse", mesas digitalizadoras, e a gerência desses recursos é justamente a função de um Sistema de Desenvolvimento de Interfaces com o Usuário - SDIU, como foi discutido acima.

Muitas vezes os Sistemas de Gerenciamento de Janelas são confundidos com Gerenciadores de janelas, mas é bom colocar que um Gerenciador de Janelas faz parte de um Sistema de Gerenciamento de Janelas [FOL_90]. Assim, um Sistema de Gerenciamento de Janelas é dividido em duas partes. A primeira é um *Gerenciador de Janelas*, através do qual o usuário interage para solicitar que janelas sejam criadas, modificadas, movidas, fechadas, abertas, ou que as operações usuais sejam efetuadas sobre elas. A segunda componente é o *Sistema de Janelas*, o qual efetua realmente as operações de criação, modificação, translação, fechamento e abertura das janelas.

O Gerenciador de Janelas é construído sobre o Sistema de Janelas, ou seja, o Gerenciador de Janelas utiliza os serviços oferecidos pelo Sistema de Janelas. Nesta

estrutura hierárquica também são colocados os pacotes gráficos de alto nível e os programas de aplicação.

Os programas construídos utilizando-se um Sistema de Gerenciamento de Janelas que podem atender a diversos processos simultaneamente são comumente chamados de programas **Clientes** ("Clients"), os quais utilizam os recursos do Sistema de Gerenciamento de janelas, o qual nesse contexto é geralmente chamado de **Servidor** ("Server"). Em alguns Sistemas de Gerenciamento de Janelas que utilizam a arquitetura do tipo servidor-cliente, tal como o X Window System [SCH_86] e o NeWS [SUN_87], o próprio Gerenciador de Janelas também é visto como sendo um cliente especial. A figura 28 mostra como o sistema de janelas pode se relacionar com o sistema operacional e com os programas de aplicação.

Figura 28 - Relacionamento entre o usuário e a máquina, através da arquitetura Servidor-Cliente.

Alguns Sistemas de Janelas, tal como o X Window e o NeWS [SUN_90c], são classificados como "independentes de especificação", ou seja, podem ser colocados sobre eles diferentes Gerenciadores de Janelas, cada um com uma "aparência" diferente para seus componentes. Isso dá-se devido ao fato de que o Gerenciador de Janelas, e não o Sistema de Janelas, é que determina qual é a "aparência" das janelas, e como o usuário interage com elas. Assim, tal como muitos programas de aplicação diferentes podem ser construídos

sobre pacotes gráficos, muitos Gerenciadores de Janelas podem ser construídos sobre um Sistema de Janelas "independente de especificação". Para que tal fato ocorra, o Sistema de Janelas deve ser projetado para suportar uma grande variedade de especificações de Gerenciadores de Janelas. Naturalmente, em um ambiente específico, o Gerenciador de Janelas e o programa de aplicação precisam estar utilizando uma mesma especificação de "aparência" para a interface com o usuário.

Caso a interface de programação para o sistema de janelas seja claramente definida e implementada via intercomunicação entre processos, os clientes do sistema de janelas podem estar em computadores diferentes, desde que estejam conectados a aquele que armazena o Sistema de Janelas através de uma rede de alta velocidade. Esta comunicação é feita principalmente através de mensagens que carregam eventos disparados entre uma máquina e outra. A utilização de comunicação entre processos possibilita que aplicações com demanda intensiva de computação possam residir em computadores poderosos, enquanto o usuário interage com a aplicação através de uma estação de trabalho. Logo este modelo servidor-cliente é justamente uma visão sofisticada de um protocolo para terminal virtual [FOL-90].

Uma possível arquitetura distribuída que utiliza Sistemas de Janelas baseada em eventos gerados pelo usuário pode ser vista na figura 29.

Figura 29 - Relacionamento entre janelas, clientes e eventos.

Com o fato de que, normalmente, grande parte do custo de um sistema computacional é devido ao software, é cada vez mais interessante que os sistemas de software desenvolvidos sejam independentes de máquina. Por preconizar tal característica, e também porque sua utilização em sistema UNIX tem sido largamente aceita, o Sistema X Window, que foi desenvolvido pelo MIT, vêm se tornando um padrão de fato, para a construção de interfaces com o usuário nesses sistemas [BAR_91]. Pode-se dizer que um sistema de software construído utilizando o Sistema X Window como sistema de janelas, utiliza aquela estrutura colocada acima das aplicações servidor-cliente.

O sistema X Window possui basicamente três módulos: o X Server, o X Client e o X Protocol.

O X Server é o módulo do X Window que encapsula o hardware, possibilitando que os programas aplicativos (X Client) utilizem os recursos disponíveis pelo hardware, ou seja, todas as ações de entrada de dados pelos dispositivos de entrada, como de saída gráfica devem ser realizadas pelo X Server. O Sistema X Window apresenta "facilidades" para a sua utilização em sistemas computacionais baseados em rede. Isto, porque a própria filosofia de construção do X Window é dirigida a eventos e mensagens [Rad_90]. Os diferentes tipos de mensagens são coletivamente chamados de X Protocol. Uma mensagem pode ser uma solicitação para desenhar uma linha, outra para desenhar um círculo, outra para escrever um texto, e assim por diante. Além disso o X Server pode enviar de volta ao X Client mensagens especiais, tais como mensagens de erros ou mesmo outros eventos. Essas mensagens especiais também fazem parte do X Protocol. O X Client é o programa de aplicação do usuário que solicita os recursos oferecidos pelo X Server. A figura 30 apresenta um exemplo de utilização de um Sistema X Window para efetuar-se uma saída gráfica.

Figura 30 - Exemplo de solicitação de uma operação de saída gráfica por um X Client.

5.2.2 - Gerenciadores de "menus".

As unidades de vídeo com memória diretamente acessível pelo computador permitem a apresentação de dados ao usuário de uma forma rápida e flexível. Em conjunto com algum dispositivo de coleta de dados (teclado ou "mouse" em geral), podem atuar como um dispositivo lógico de entrada. Isto é conseguido apresentando-se na tela dispositivos típicos de entrada, tal como botões de seleção, potenciômetros, listas de opções e assim por diante. Essa representação de dispositivos de entrada são tradicionalmente chamados de *"menus"*. Dessa forma mostram-se ao usuário, através de um dispositivo de vídeo, várias possibilidades de escolha, parâmetros que podem ser modificados, etc., sendo que o usuário através do teclado ou de um dispositivo apontador, indica suas opções.

Assim como em outras formas de interfaceamento com o usuário, os "menus" podem ser criados diretamente pelos programas usando apenas uma linguagem de programação tradicional, ou serem produzidos através da utilização de "toolkits", ou mesmo dispor de um Gerenciador específico que coordene sua atividade [TRA_90]. Os "toolkits" oferecem a facilidade de representar vários tipos de "menus" na tela, os quais são escolhidos e preparados pelo programa de aplicação, os quais uma vez ativados, efetuam a troca de informações com o usuário, e após a conclusão de que a escolha do usuário foi completamente satisfeita, retorna tais indicações para o aplicativo.

A operação dos "menus" é geralmente sobre as primitivas de acesso as janelas, e dessa forma as características próprias dessas primitivas repassam para a manipulação de "menus", tanto a nível da interface com o software aplicativo quanto a nível da interface com o usuário. Por exemplo, se para a manipulação de janelas utiliza-se um Gerenciador de Janelas, a flexibilidade desse gerenciador é transmitida aos "menus", os quais em última instância utilizam-se de janelas para serem apresentados no vídeo.

A terminologia para tipos de "menus" na literatura não é consistente, sendo que cada "toolkit" ou gerenciador de "menus" classifica-os segundo seus próprios critérios. Dessa forma neste trabalho os "menus" serão classificados basicamente em dois tipos: "menus" dos tipos "pull-down" e "pop-up".

Os "menus" "pull-down" são aqueles que oferecem ao usuário um leque de opções, sendo que cabe ao usuário apenas escolher alguma delas. O termo "pull-down" tem relação

com a forma com que o "menu" se apresenta, a qual lembra um pergaminho que se abre. Este "menu" pode possuir sub-"menus" associados às suas entradas, sendo desta forma hierárquico. A figura 31 mostra um exemplo de um "menu" "pull-down".

Figura 31 - Exemplo de "menu" "pull-down".

Os "menus" "pop-up" oferecem ao usuário mais formas de interação do que os "menus" "pull-down", porque além de permitirem a escolha de uma opção, também permitem que o usuário entre com valores numéricos, textos, manipule potenciômetros, etc. Por este "menu" estar sempre apresentado em uma estrutura que lembra uma janela, em alguns sistemas é chamado de "pop-up window" [SUN_90b]. A figura 32 apresenta um exemplo de um "menu" "pop-up".

Editor de texto: Procura e Troca

Procura: _____

Troca com: _____

Opções: ☐ Confirma cada uma ☐ Ignora caixa

Figura 32 - Exemplo de "menu" "pop-up".

5.2.3 - Padrões de Apresentação.

A "aparência" de uma interface com o usuário de um sistema de software, deve-se ao fato de ser desejável compatibilizar dois requisitos contraditórios: por um lado os sistemas de software devem facilitar o aprendizado de sua utilização, e a melhor maneira de conseguir isso, é fazendo com que a sua utilização, ou seja fácil e naturalmente intuitiva, ou seja em decorrência de algum conhecimento anterior, como por exemplo ser parecida com a utilização de outro software que desempenhe a mesma operação de maneira semelhante. Por outro lado, como cada fabricante de software deseja que o seu produto tenha características marcantes e visíveis que o distingua do demais, entra-se em conflito.

Assim, muitos fabricantes mantêm sua forma de apresentar suas interfaces como é o caso do Environ V da Intergraph [INT_88a], onde a aparência do seus aplicativos está intimamente relacionada com o seu ambiente gráfico para a construção de interfaces com o usuário.

Outros fabricantes porém, preferem trabalhar com especificações de "aparência" de seus "softwares" um pouco mais abrangentes. Este é o caso da Sun Microsystems que em parceria com a AT&T desenvolveu o OPEN LOOK, que é uma especificação de formato, de como um ambiente de janelas para um sistema computacional que comporte múltiplas tarefas deve ser visualizado [SUN_90a] [SUN_90b]. O OPEN LOOK decorreu do trabalho pioneiro sobre interfaces gráficas realizado pela Xerox na década de 70, e atualmente é o

novos padrões de apresentação utilizados pelos Gerenciadores de Janelas que as estações Sun utilizam.

5.3 - Conclusões.

Neste capítulo foram apresentadas as atuais técnicas que estão sendo desenvolvidas para suportar o interfaceamento de sistemas de software com seus usuários.

Pode-se ver que os projetistas de software têm se preocupado grandemente em produzir sistemas que possuam interface "amigável" com o usuário, e que além disso não estejam presos a alguma máquina específica. Assim, tem surgido como um padrão de fato [WIL_91] a utilização do Sistema de Janelas X Window, que por ser independente de máquina tem se tornado muito atraente para a construção de interfaces em sistemas que devem ser portáteis. Pode-se notar também que o X Window é um sistema grande e complexo, e que devido a essa característica nem sempre pode ser adotado, quando se leva em conta o tamanho da máquina que deverá comportar um sistema de software. Mas a filosofia nele embutida de manipulação de janelas com vários atributos, gerenciamento de eventos, e utilização de sistemas distribuídos podem ser utilizadas mesmo em sistemas menores.

Capítulo VI

Ambiente Gráfico para Visualização de Imagens Tomográficas

6.1 - Introdução.

O objetivo prático deste trabalho, é a elaboração de um sistema dedicado, capaz de visualizar e manipular as imagens obtidas pelo tomógrafo por Ressonância Magnética, em desenvolvimento no Departamento de Física e Ciência dos Materiais do IFQSC [PAN_87] [TAN_87].

A construção da arquitetura dedicada baseada no processador TMS34010 [PAI_90], no Laboratório de Instrumentação Eletrônica (LIE), viabilizou o desenvolvimento de um ambiente gráfico, que possibilitasse ao usuário a utilização dos recursos oferecidos pelo hardware .

Tendo em vista a particularidade das aplicações e das características do TMS34010, optou-se pelo desenvolvimento próprio de um ambiente gráfico. Dessa forma, o software resultante aproveitaria melhor as características do hardware ; e seria também compatível com as necessidades do sistema de tomografia. Os sistemas mais utilizados atualmente, como XWindow e NeWS, são muito grandes e complexos para a arquitetura dedicada, não sendo adequados ao sistema de visualização em desenvolvimento.

As rotinas que constituem o ambiente gráfico foram desenvolvidas em linguagem de programação "C" na sua grande maioria, contudo as rotinas que acessam diretamente os recursos do GSP foram escritas em linguagem de montagem do processador TMS34010.

Nas seções a seguir será detalhado o ambiente gráfico desenvolvido.

6.2 - O Ambiente Gráfico Desenvolvido.

Neste trabalho **Ambiente Gráfico** será definido informalmente como um sistema de "software" que, para uma dada arquitetura, fornece primitivas gráficas básicas, as quais podem ser facilmente acessadas por um aplicativo escrito em linguagem de alto nível. Além disso, um Ambiente gráfico deve fornecer ferramentas para a construção de aplicativos cuja interface suporte o conceito de "Amigabilidade com o Usuário". Exemplos típicos são: construção de janelas, "menus", tratamento de imagens, bem como todo o gerenciamento entre essas estruturas. Essas premissas foram levadas em conta para o projeto do ambiente gráfico desenvolvido.

Quanto ao ambiente gráfico optou-se por desenvolvê-lo de forma hierárquica e modular em três níveis: Nível Básico, Nível de Apoio e Nível de "Toolkit". Assim os níveis superiores são funcionalmente dependentes dos níveis inferiores, e além disso, somente o nível mais baixo é dependente de máquina. Essa estrutura é apresentada na figura 33.

Figura 33 - Hierarquia do Ambiente Gráfico.

A seguir é descrito cada nível da hierarquia do ambiente gráfico:

Nível Básico: Neste nível estão as primitivas que solicitam as operações que acessam diretamente as primitivas gráficas que o hardware suporta. Tais primitivas são em grande parte escritas em linguagem montadora do TMS34010. Como exemplo, pode-se tomar as primitivas para desenhar pontos, linhas, retângulos, inicializar registradores do TMS34010, etc. As próprias estruturas de dados que

gerenciam todo o ambiente gráfico (janelas, "menus", etc) encontram-se neste nível, fora do alcance do projetista de "software".

Nível de Apoio: Neste nível estão as primitivas que, utilizando aquelas do nível básico preparam o "hardware" para as operações gráficas do nível de "toolkit". Também encontram-se aqui as primitivas de comunicação com o "hospedeiro", as que solicitam-lhe recursos, tais como acesso a disco, teclado, transferência de imagens, etc., bem como as primitivas de gerenciamento do ambiente gráfico. Todas estas primitivas foram implementadas como rotinas escritas em linguagem "C".

Nível de "Toolkit": Este constitui-se no nível onde estão as primitivas usualmente utilizadas para a construção dos aplicativos. Incluem-se aqui todas as primitivas de manipulação de "menus", janelas, imagens, tratamento de eventos, etc.

Tal como pode-se notar na figura 33, os aplicativos construídos utilizando este ambiente, fazem-no através de chamadas às primitivas do nível de "toolkit". Contudo aplicativos que demandem uma maior dependência do ambiente ou do "hardware" podem efetuar chamadas também às primitivas do nível de apoio. As primitivas do nível de apoio suprem as necessidades que são normalmente supridas por um ambiente de programação em linguagem "C". Isso deve-se ao fato de o acesso à dispositivos de entrada/saída a partir do TMS34010 poder ser feito tanto diretamente aos seus dispositivos, quanto àqueles conectados ao "hospedeiro". Assim, este ambiente provê recursos que viabilizam e flexibilizam qualquer tipo de acesso.

6.3 - Elementos que Compõem o Ambiente Gráfico.

O ambiente gráfico é composto por diversos elementos que podem ser utilizados por um aplicativo para utilizar os recursos da arquitetura. Tais recursos podem ser agrupados como: Gerenciador de Janelas; Gerenciador de "menus"; Subsistema de Comunicação com o Hospedeiro; Subsistema de Visualização de Imagens. Os recursos disponíveis no

ambiente gráfico são descritos nas seções seguintes, e os módulos de "software" que os implementam são apresentados nos apêndices deste trabalho. Na seção 6.5 é apresentado um exemplo de um aplicativo que utiliza o ambiente gráfico.

6.3.1 - Gerenciador de Janelas.

A principal estrutura do ambiente gráfico desenvolvido é a de **Janela**. Uma **Janela** é uma região retangular da tela (ver figura 34), a qual é definida pelas coordenadas de seus quatro cantos. A indicação dessas coordenadas é padronizada em todo o ambiente como sendo a indicação de seu canto superior esquerdo (na figura: (x_1, y_1)) e o canto inferior direito (na figura: (x_2, y_2)).

Figura 34 - Especificação de uma janela.

As janelas se associam de forma hierárquica, onde a janela principal (raiz da hierarquia de janelas), é a própria tela do monitor do sistema de visualização. A figura 35 mostra um exemplo de como poderia ficar a visualização de um sistema constituído por 9 janelas. Nesse exemplo, a tela é a janela raiz da hierarquia (R), e possui três janelas

filhas: A, B e C. Por sua vez, a janela A possui três janelas filhas: A1, A2 e A3; a janela B não possui filhas; e a janela C possui duas filhas: C1 e C2.

Figura 35 - Exemplo de uma tela com janelas.

A maneira como as janelas podem se associar deve respeitar algumas restrições:

- Uma janela não pode ultrapassar os limites da janela que é seu pai, ou seja, as janelas A1, A2 e A3 são inscritas na janela pai (A), assim como as janelas A, B e C são inscritas na janela raiz (tela).
- Os atributos de uma janela pai são transmitidos à filha, a menos que sejam explicitamente definidos na hora da criação. Esses atributos podem ser: tamanho e posição da janela, eventos permitidos, quais os planos de memória utilizados, cor de frente, cor de fundo, posição do cursor, arquivo de caracteres fonte.
- O sistema impõe que sempre uma janela deve ser considerada como a **janela corrente**.
- Existem operações que somente são realizadas sobre a janela corrente. Um exemplo é a criação de uma outra janela: uma nova janela somente pode ser criada como filha da janela corrente.

O sistema fornece dois tipos básicos de janelas: Janela com borda e Janela sem borda.

• **Janela com Borda:** é a janela que possui uma moldura. Esta moldura pode ser de três tipos:

Traço-duplo: a borda é realçada por linhas mais grossas, e admite cabeçalho e elevador de rolamento vertical.

Traço-simples: a borda tem espessura normal e também admite cabeçalho e elevador de rolamento vertical.

Borda-singela: a borda tem espessura normal mas não possui cabeçalho nem elevador de rolamento. É a borda que utiliza-se em janelas que estão comportando "menus" (ver seção 6.3.2).

a) Janela com borda do tipo Traço-duplo.

b) Janela com borda do tipo Traço-simples.

c) Janela com borda do tipo Borda-singela.

Figura 36 - Apresentação dos tipos e respectiva largura de bordas de janelas.

A figura 36 mostra como é visualizada cada um dos três tipos de janela com borda, e também a largura da moldura (em pixels).

Quando o usuário solicita que seja criada uma janela com borda, na realidade o sistema cria duas janelas: a janela que contém a moldura, e uma janela filha que ocupa a região de trabalho que o usuário solicitou. Dessa forma, se um usuário solicita que seja criada uma janela com borda de traço-duplo, e que a definição da janela seja o canto superior esquerdo em (50,50) e o canto inferior direito em (100,100), na realidade a janela de trabalho será a região entre (58,70) e (85,98) por causa da reserva de espaço feita para conter a borda da janela. O controle sobre o tamanho da área efetiva de trabalho da janela deve ser feito pelo usuário, levando-se em conta a largura de cada tipo de borda. Deve-se notar que a associação entre a borda da janela, e a janela de trabalho propriamente dita no sistema de gerenciamento de janelas é que a janela de trabalho é filha da janela que contém a borda, porém esta última não é normalmente acessível ao usuário. Existe uma única primitiva que gera uma janela com borda, qualquer que seja sua borda, a qual é chamada `Cria_jan_borda`.

- **Janela sem Borda:** este tipo de janela delimita uma região retangular da tela, a qual será utilizada como uma área de trabalho pelo aplicativo.

Janelas sem borda criam apenas a janela solicitada, utilizando toda a área solicitada pela definição da janela. Existe uma única primitiva que gera uma janela sem borda, a qual é chamada `Cria_janela`.

Como ao criar-se uma janela o Gerenciador de janelas transmite os atributos de uma janela pai à janela filha, alterando-se apenas os atributos explicitamente referenciados, as primitivas que efetuam a criação de janelas - **Cria_janela** e **Cria_jan_borda** - recebem como parâmetros pares de atributo-valor, os quais serão atribuídos à janela criada. No Manual de Referência do Ambiente Gráfico, apêndice B, são apresentados exemplos para a utilização de cada rotina.

6.3.2 - Gerenciador de "menus".

A utilização de "menus" para a interação com o usuário, obtendo-se dados que sejam de interesse para o software aplicativo, permitindo corrigir ou atualizar esses dados, confirmar opções e dados, ou mostrar os dados correntes numa estrutura igual à usada para a coleta de dados, a qual o usuário já está acostumado, é uma das formas de se tornar o software "amigável".

Basicamente os "menus" podem ser classificados em dois tipos [FOLEY_90]:

- "menus" estáticos: que são permanentemente visíveis. Estes "menus" são colocados uma vez na tela quando o aplicativo começa a ser executado e só são retirados quando o aplicativo termina.
- "menus" dinâmicos: tornam-se visíveis apenas quando são requisitados. Os tipos mais conhecidos, como já visto anteriormente, são os chamados "menus" tipo "pull-down" e "pop-up".

O ambiente gráfico desenvolvido conta com "menus" do tipo "pull-down", os quais podem ser colocados de forma hierárquica. As estruturas de dados, e mesmo o sistema de software já foram projetados para também suportarem "menus" do tipo "pop-up". A manutenção ou não de um "menu" na tela é opção do aplicativo. A figura 37 mostra a evolução de um processo de escolha nesse tipo de menu.

A figura 38 apresenta um "menu" do tipo "pop-up", da forma como o ambiente gráfico pode suportar.

Figura 37 - "menu" "pull-down" hierárquico. (a) O primeiro "menu" mostra onde o cursor está posicionado. O cursor pode ser movido para cima ou para baixo para selecionar-se o item. (b) No "menu" hierárquico, o item que possui um sub-"menu" associado apresenta uma seta lateral, e ao andar-se com o cursor para a direita aciona-se o sub-menu.

Figura 38 - "menu" "pop-up", onde há uma entrada textual e três entradas tipo botão.

O "menu" "pull-down" foi desenvolvido segundo a filosofia do ambiente gráfico **Environ V** da Intergraph [INT_88a], ampliando-se apenas a forma como as opções podem ser mostradas. No **Environ V**, todas as opções de um "menu" são mostradas simultaneamente, e o controle das dimensões do "menu" é feito pelo usuário. No ambiente gráfico desenvolvido, o "menu" "pull_down" não precisa mostrar simultaneamente todos

os ítems, permitindo especificar na criação do "menu" quantos ítems serão mostrados ao mesmo tempo, e os demais ítems serão mostrados conforme o usuário solicitar, através do rolamento do menu.

Quando um "menu" é criado, o ambiente gráfico coloca-o sobre uma janela (janela com borda, do tipo borda-singela), e o Gerenciador de Janelas passa também a ter controle sobre o menu. Isso é feito para simplificar a forma de gerenciar "menus", utilizando-se os recursos já descritos para o gerenciamento de janelas.

6.3.3 - Subsistema de Comunicação com o Hospedeiro.

A arquitetura dedicada baseada no processador gráfico TMS34010 (GSP), como já foi visto no capítulo 1, utiliza o microcomputador IBM-PC como hospedeiro. A capacidade de transferir grandes blocos de dados para o GSP torna a interface com o hospedeiro bastante ágil. Por exemplo, o hospedeiro pode transferir blocos de comandos para o GSP, pode parar o GSP temporariamente para carregar um novo programa para o GSP executar, ou mesmo pode ler blocos de dados gráficos gerados pelo GSP.

A interface de comunicação com o hospedeiro, que pode ser acessada tanto pelo hospedeiro quanto pelo GSP (ver seção 4.2.3.4), ocupa os seguintes registradores que são mapeados ou em 4 regiões consecutivas de 16 bits no espaço de endereço de memória do GSP, ou em registradores de E/S [TEXAS_86].

HSTCTL - Este registrador controla funções tais como: a transferência de pedidos de interrupção e o código de estado em 3 bits entre o hospedeiro e o TMS34010. Estas solicitações são tipicamente utilizadas pelo software para coordenar a transferência de grandes blocos de dados através da memória local do GSP. Outras funções do HSTCTL são: permitir que o hospedeiro imponha a descarga da "cache" de instrução do GSP, parar a execução do GSP, e transmitir solicitação de interrupção não mascarável ao GSP.

HSTADRH e HSTADRL - Estes registradores formam um único registrador, que contém um endereço de 32 bits o qual aponta para uma palavra de memória corrente a ser acessada.

HSTDATA - Este registrador armazena o dado transferido de/para a memória local do GSP sob o controle do processador hospedeiro. A interface com o

hospedeiro pode ser programada para automaticamente incrementar o endereço após cada transferência, e dessa forma o hospedeiro pode obter um acesso rápido a um bloco de dados armazenado em posições consecutivas de memória.

A figura 39 ilustra o procedimento para a leitura de um bloco de palavras (de 16 bits) na memória local (do GSP), cujo endereço inicial é N . Deve-se assumir que está ativado o autoincremento do endereço indicado após o acesso ao dado. Em (a), o hospedeiro armazena os 32 bits do endereço N nos registradores HSTADRH (16 bits) e HSTADRL (16 bits). A operação de se carregar a segunda parte do endereço em HSTADRH, faz com que a interface do GSP com o hospedeiro automaticamente inicie um ciclo de leitura à memória local (do GSP). O ciclo de leitura, mostrado em (b), transfere o conteúdo do endereço de memória N para o registrador HSTDATA. Em (c), o processador hospedeiro lê o registrador HSTDATA, buscando o dado previamente lido do endereço N . A leitura do HSTDATA pelo processador hospedeiro, faz com que o GSP incremente de forma automática o conteúdo do par HSTADRL e HSTADRH de 16, tal como é mostrado em (d). O conteúdo do novo endereço é lido em HSTDATA, como é mostrado em (e). Este dado estará disponível em HSTDATA na próxima leitura feita pelo processador hospedeiro. O processo mostrado nos ítems (c), (d), e (e), repete-se para cada palavra lida da memória local do GSP. Um procedimento análogo é feito para o procedimento de escrita, onde o bit INCW do HSTCTL é colocado em 1.

Figura 39 - Leitura indireta da memória local do GSP pelo processador hospedeiro.

O TMS34010 possui entre as ferramentas para desenvolvimento de software um carregador (loader) chamado **SDBL**, o qual verifica se o GSP está pronto, e em caso positivo carrega um programa executável na memória local do GSP. Isso é feito através dos registradores da interface com o hospedeiro vistos acima.

Através do SDBL, carrega-se um programa no GSP e não se tem mais qualquer comunicação com o hospedeiro. Ou seja, os recursos disponíveis apenas no hospedeiro, tais

como: teclado e disco, não ficam disponíveis ao programa de aplicação. Contudo, um programa aplicativo com alto grau de interatividade e que além do mais trate de imagens (estas armazenadas em disco) precisa utilizar tais recursos. É necessário também que as operações a serem realizadas o façam com a maior eficiência possível. Assim, foi utilizada a técnica de Acesso Direto à Memória (DMA) do GSP sempre que possível, como é o caso das transferências de blocos de dados entre o GSP e o disco do hospedeiro. Por outro lado, quando os dados devem ser transferidos byte a byte, tal como é o caso de transferências de caracteres digitados no teclado, ou mostrados no vídeo do hospedeiro, a técnica utilizada foi a de "hand-shake", levando-se em conta que o processamento em paralelo da transferência e do aplicativo em execução no GSP fica inevitavelmente comprometido devido à necessidade do aplicativo depender dos dados transferidos ou da confirmação da transferência.

Dessa forma foi feita uma extensão ao SDBL, criando-se uma nova ferramenta a que chamou-se *SDBLR* (SDBL com suporte de comunicação Residente), a qual após carregar o programa objeto na memória local do GSP, permanece residente no hospedeiro, monitorando um "buffer" de comunicação colocado entre o GSP e o hospedeiro. Este "buffer" possui 20 bytes, pois dessa forma acomoda perfeitamente a estrutura de dados *LIE_comunica* (ver seção 6.4.5), a qual contém os campos de controle para os dispositivos do hospedeiro. Na figura 40 é ilustrado o esquema de comunicação entre o GSP e o hospedeiro através do "buffer" de comunicação. A figura 41 apresenta o diagrama de blocos da ferramenta *SDBLR*.

Figura 40 - Esquema de comunicação entre o GSP e o hospedeiro.

Figura 41 - Diagrama de blocos do carregador SDBLR.

Figura 41 - Diagrama de blocos do carregador SDBLR - Continuação.

Além disso foi necessário desenvolver um subsistema de comunicação com o hospedeiro, o qual gerenciaria todas as solicitações de operações a disco e E/S executadas nos periféricos do hospedeiro, transmitindo e recebendo os dados envolvidos.

Os recursos do hospedeiro gerenciados são:

- teclado,

- vídeo,
- disco.

A transferência dos dados entre a arquitetura gráfica baseada no TMS34010 e o Hospedeiro é feita utilizando-se de 20 bytes, ao qual é associada uma instância da estrutura LIE_comunica (seção 6.4.5), que possibilita o controle pelo SDBLR de qual recurso do hospedeiro o aplicativo em execução no GSP está solicitando, e também da própria transferência dos dados. A forma de transferência dos dados é através de acesso direto a memória (DMA) do GSP.

O ambiente de interação com o GSP implementado no hospedeiro IBM-PC permite ao usuário controlar e monitorar o andamento da execução de um aplicativo no GSP que utilize recursos do hospedeiro. Para a criação do ambiente no IBM-PC utilizou-se o núcleo gráfico desenvolvido no Departamento de Computação e Estatística - ICMSC - USP, que faz parte do Sistema GEO [TRAINA_90].

Utilizando-se esse núcleo, a tela do hospedeiro foi dividida em três janelas: uma janela na parte superior que recebe os dados enviados pelo TMS para serem mostrados na tela do hospedeiro; uma janela na parte inferior que mostra os dados que o hospedeiro está transmitindo para o TMS, e uma janela pequena no centro que indica se o teclado está transmitindo para o TMS (ON), ou se está a serviço do próprio hospedeiro (LOCAL). A figura 42 mostra como a tela do hospedeiro é organizada pelo subsistema de Comunicação.

Figura 42 - Organização da tela do monitor do hospedeiro, quando o SDBLR está em execução.

Como foi visto na seção 6.2, no nível de Apoio do ambiente gráfico, estão as primitivas que compõem o subsistema de Comunicação com o hospedeiro. Tais primitivas estão implementadas como rotinas que possuem nomes parecidos com as da biblioteca padrão da linguagem "C", que efetuam a mesma função. Por exemplo: as funções **i_printk** que efetua uma saída formatada na tela do hospedeiro, e **v_printk** que o faz na tela do TMS, de forma equivalente à **printf** da linguagem "C". Da mesma forma foram construídas outras funções equivalentes as da linguagem "C". A tabela 8 apresenta a correspondência entre as rotinas do nível de Apoio, e as da biblioteca da linguagem "C" que possuem a mesma função.

Função Realizada	Biblioteca Padrão "C"	No GSP	No Hospedeiro
lê caracter no teclado	getch()	-	i_getc()
escreve caracter	putch (caracter)	v_putc (caracter)	i_putc (caracter)
realiza saída formatada	printf (fmt,...)	v_printk (fmt, ...)	i_printk (fmt, ...)
lê cadeia de caracteres	gets (*string)	-	i_getk (*cadeia)
escreve cadeia de caracteres	puts (string)	v_puts (string) v_putk (cadeia)	i_puts (string) i_putk (cadeia)
Abre arquivo	open (arquivo, modo)	-	i_open (arquivo, modo)
Fecha arquivo	close (Narquivo)	-	i_close (Narquivo)
Lê dado de arquivo	read (Narq,buffer,tam)	-	i_read (Narq, buffer, tam)
Escreve dados no arquivo	write (Narq, buffer, tam)	-	i_write (Narq,buffer,tam)
Posiciona no arquivo	seek (Narq, offset, como)	-	i_seek (Narq, offset, como)

Tabela 8 - Rotinas para E/S pertencentes ao nível de Apoio do ambiente gráfico, e sua correspondência às da Linguagem "C".

O subsistema de Comunicação pode ser dividido em duas partes:

1. Um conjunto de rotinas que são executadas no hospedeiro, e realizam as funções de E/S normalmente realizadas pela biblioteca da linguagem "C";
2. Um outro conjunto de rotinas que executam sincronizadas com aquelas em execução no hospedeiro, e complementam as operações necessárias à comunicação.

No Apêndice C encontra-se a descrição das rotinas que efetuam as funções de E/S no hospedeiro e no GSP,

A tabela 9 ilustra como os pares de funções do subsistema de comunicação operam para a transferência de dados. Mostra a equivalência das funções do GSP com as do hospedeiro.

Função Realizada	No GSP	No Hospedeiro
Obtém um caracter do hospedeiro	i_getc	tms_sendc
Envia um caracter para o hospedeiro	i_putc	tms_recevec
Verifica existência de caracter transmitido pelo hospedeiro	i_pollc	tms_send_poll
Verifica existência de caracter transmitido pelo GSP	-	tms_receive_poll
Inicializa busca de arquivos, posiciona no primeiro	i_arqinicio	tms_disco_ARQINICIO
Prossegue a busca de arquivos, posiciona no seguinte	i_arqproximo	tms_disco_ARQPROXIMO
Abre arquivo indicado no hospedeiro	i_open	tms_disco_ABREARQ
Fecha arquivo indicado no hospedeiro	i_close	tms_disco_FECHARQ
Lê dados indicados do arquivo do hospedeiro	i_read	tms_disco_LEARQ
Escreve dados indicados no arquivo do hospedeiro	i_write	tms_disco_ESCARQ
Posiciona ponteiro no arquivo do hospedeiro	i_seek	tms_disco_POSICIONARQ

Tabela 9 - Equivalência das funções de E/S (GSP ↔ Hospedeiro).

6.3.4 - Subsistema de Transferência de Imagens Tomográficas de RM.

O objetivo principal da construção do ambiente gráfico sobre o GSP foi para possibilitar a transferência de imagens coletadas pelo sistema de tomografia do IFQSC para a arquitetura dedicada a visualização de imagens, permitindo ao usuário efetuar manipulações sobre a imagem.

A operação de transferência das imagens foi colocada em um subsistema a parte, apesar de necessitar de comunicação entre o hospedeiro e o GSP, foi necessário fazer-se um tratamento diferenciado sobre os dados que compõem a arquivo de imagem.

O arquivo de imagens gerado pelo sistema de tomografia contém os dados formatados, pelo fato de facilmente compatibilizá-lo com todos os módulos que constituem o sistema.

A primitiva que efetua a operação de leitura de uma imagem no disco do hospedeiro, e a transfere para o GSP foi chamada de `i_le_imagem`. Esta primitiva abre o arquivo de imagem no hospedeiro e o transfere para o GSP. O GSP cria uma janela com borda que comporte exatamente a imagem, e coloca a imagem na janela para que o usuário possa manipulá-la. Na seção 6.5 o exemplo apresentado ilustra como a primitiva `i_le_imagem` opera.

6.4 - Descrição das Estruturas de Dados Utilizadas.

Nessa seção é feita uma descrição das estruturas de dados utilizadas na implementação do ambiente gráfico. O apêndice D lista os arquivos de "headers" do sistema, onde as estruturas estão definidas.

6.4.1 - Estrutura de Dados para o Gerenciamento de Janelas - LIE_janela.

Esta estrutura define o nó básico da estrutura de dados que é disposta hierarquicamente em forma de árvore, indicando como as janelas do ambiente gráfico estão relacionadas. A estrutura `LIE_janela` contém as informações principais sobre uma janela, e nela também estão dispostas outras estruturas de dados (esquematizadas como listas), que armazenam informações tidas como secundárias. Cada nó da estrutura é alocado

dinamicamente na memória de sistema do GSP. A figura 43 apresenta a estrutura de dados LIE_janela. Na figura 44 é ilustrado como ficaria a hierarquia de janelas para a figura 35.

O sistema mantém em uma variável global *lie_jan_corrente* qual é a janela corrente que o sistema está tratando. Isto é feito porque as operações de janelas são na sua maioria definidas sobre a janela corrente.

A estrutura LIE_janela faz parte do Nível Básico do ambiente gráfico, bem como as rotinas que manipulam a hierarquia em forma de árvore.

O significado de cada um dos campos da estrutura LIE_janela é colocado a seguir, devendo-se notar que os primeiros quatro campos da estrutura armazenam os ponteiros que constroem a árvore de janelas.

PAI - (32 bits) armazena o ponteiro para o nó LIE_janela que armazena as informações da janela pai desta janela.

PRIM_FILHO - (32 bits) armazena o ponteiro para o nó LIE_janela que armazena as informações da janela que é o primeiro filho desta janela. A partir deste ponteiro pode-se recuperar a lista de filhos de uma dada janela. Sempre que é criado um filho de uma janela, o nó que especifica este filho é colocado na cabeça da lista.

PROX_IRMAO - (32 bits) armazena o ponteiro para o nó LIE_janela que armazena as informações da janela que é o próximo irmão desta janela na lista.

IRMÃO_ANTERIOR - (32 bits) armazena o ponteiro para o nó LIE_janela que armazena as informações da janela que é o irmão anterior desta janela na lista.

TIPO - (32 bits) armazena o ponteiro para a estrutura LIE_tipo_janela que armazena as informações sobre a aparência dessa janela.

SENTINELA - (32 bits) campo utilizado para consistência da estrutura de dados.

Na criação de um nó dessa estrutura é atribuído o valor 1234 a esse campo, e sempre que se recuperar um nó, este campo é testado.

XLO - (32 bits) armazena a posição no eixo X do canto superior esquerdo da janela.

YLO - (32 bits) armazena a posição do eixo Y do canto superior esquerdo da janela.

XHI - (32 bits) armazena a posição no eixo X do canto inferior direito da janela.

YHI - (32 bits) armazena a posição no eixo Y do canto inferior direito da janela.

PLANE_MASK - (16 bits) máscara que especifica quais os planos que estão ativos na memória. Para o sistema atual que conta com 8 planos de memória, se

todos os planos estiverem ativos, em **PLANE_MASK** deve estar com o valor 0000H.

EVENT_MASK - (16 bits) máscara que especifica quais os eventos permitidos para essa janela.

COR_FRENTE - (32 bits) armazena qual é a cor de frente para esta janela.

COR_FUNDO - (32 bits) armazena qual é a cor de fundo para esta janela.

X_CURSOR - (32 bits) armazena a posição corrente do mouse no eixo X.

Y_CURSOR - (32 bits) armazena a posição corrente do mouse no eixo Y.

ARQ_FONTE - (32 bits) armazena o ponteiro para o arquivo de fontes que essa janela pode acessar.

BASE - (32 bits) armazena o ponteiro para a região da memória do sistema que armazena a região da tela onde a imagem foi mapeada. É usado para a operação de refrescamento da tela do GSP.

RESERVADO 1 - (32 bits) campo reservado para futuras ampliações.

RESERVADO 2 - (32 Bits) campo reservado para futuras ampliações.

ESTADO - (16 bits) especifica se a janela está mapeada na tela (bit 0 é 1), ou se a janela presentemente não está mapeada (bit 0 é 0).

Figura 43 - Estrutura de dados LIE_janela.

Figura 44 - Visão da hierarquia de janelas da figura 35.

6.4.2 - Estrutura de Dados LIE_tipo_janela.

Nesta estrutura são armazenadas as especificações de aparência de uma janela. Os nós dessa estrutura também são alocados dinamicamente na memória de sistema do GSP, e acoplados à estrutura principal LIE_janela. A figura 45 abaixo mostra a forma dessa estrutura, e a explicação de seus campos é colocada a seguir.

Figura 45 - Estrutura de dados LIE_tipo_janela.

tipo_borda - (8 bits) especifica qual o tipo da borda:

bit 0 : 0 → borda sem traço.

1 → borda com traço.

bit 1 : 0 → borda com 1 traço.

1 → borda com 2 traços.

bit 2 : 0 → sem elevador horizontal.
 1 → com elevador horizontal.

bit 3 : 0 → sem elevador vertical.
 1 → com elevador vertical.

bit 4 : 0 → janela aberta.
 1 → janela fechada.

bit 5 : 0 → janela com borda ampla.
 1 → janela com borda singela (usada em "menus")

bit 6-8: Reservados para futuras ampliações.

z_janela - (8 bits) índice do vetor *lie_prof_janela* que indica qual é a profundidade da janela atual com relação às demais janelas. Será utilizado para tratamento de eventos de sobreposição de janelas.

texto - (32 bits) ponteiro para a cadeia de caracteres (nome) associada a janela.

elev_v - (32 bits) armazena a posição do elevador vertical em relação ao eixo Y.

elev_h - (32 bits) armazena a posição do elevador horizontal em relação ao eixo X.

limite_v - (32 bits) armazena a posição máxima que o elevador vertical pode assumir na janela.

limite_h - (32 bits) armazena a posição máxima que o elevador horizontal pode assumir na janela.

6.4.3 - Estrutura de Dados LIE_fonte.

Nessa estrutura de dados é armazenada a especificação dos fontes de caracteres que o ambiente gráfico suporta. O ambiente gráfico trabalha com uma variável global, *lie_fonte_corrente*, que especifica qual é o fonte correntemente utilizado pela janela.

A figura 46 apresenta a estrutura LIE_fonte.

Figura 46 - Estrutura de Dados LIE_fonte.

Os campos da estrutura **LIE_fonte** significam:

fonte - (32 bits) armazena o endereço do arquivo que contém o mapa de bits para cada caracter.

tam_car_x - (32 bits) armazena qual a largura do caracter em pixels de tela.

tam_car_y - (32 bits) armazena qual a altura do caracter em pixels de tela.

6.4.4 - Estrutura de Dados **lie_prof_janela**.

A estrutura **lie_prof_janela** é um vetor de **MAXJANELAS** (número máximo de janelas que o ambiente gráfico permite por aplicativo) elementos, onde cada elemento é um ponteiro (32 bits) para uma lista de janelas (**lie_janela**). O índice do vetor indica qual a lista de janelas que sobrepõem outra. Assim no índice 0 (zero) do vetor encontra-se a lista de janelas que sobrepõem todas as outras (as que estão em cima), e no índice **MAXJANELAS-1** as janelas que todas sobrepõem (as estão embaixo).

A figura 47 apresenta o vetor **lie_prof_janela**.

Figura 47 - Estrutura de dados **lie_prof_janela**.

6.4.5 - Estrutura de Dados **LIE_comunica**.

Essa estrutura de dados é utilizada pelo gerenciador de comunicação. Os seus campos, que serão explicados abaixo, são acoplados ao "buffer" de comunicação que é mapeado na memória de sistema do GSP. A figura 48 ilustra esta estrutura.

controle_teclado - (16 bits) especifica o controle do teclado do hospedeiro. Se estiver com 0 (**DADO_ACEITO**) o hospedeiro não enviou caracter, ou o caracter

enviado já foi aceito. Se estiver com 1 (DADO_PRONTO), o GSP já pode tratar o dado enviado pelo hospedeiro.

dado_teclado - (16 bits) armazena o dado enviado pelo hospedeiro.

controle_vídeo - (16 bits) especifica o controle do vídeo do hospedeiro. Se estiver com 0 (DADO_ACEITO) o hospedeiro já escreveu o caracter, ou nenhum caracter foi enviado. Se estiver com 1 (DADO_PRONTO), o GSP já enviou um caracter, mas o hospedeiro ainda não o processou.

dado_vídeo - (16 bits) armazena o dado enviado pelo GSP.

controle_disco - (16 bits) especifica qual a operação em disco solicitada pelo GSP.

Pode conter os seguintes valores:

DADO_ACEITO (0) : Não existe nenhuma operação pendente.

ARQINICIO (1) : buscar a primeira entrada no diretório do disco do hospedeiro que aceita a especificação fornecida.

ARQPROXIMO (2) : buscar a próxima entrada no diretório do disco do hospedeiro que aceita a especificação fornecida anteriormente por uma chamada de ARQINICIO.

LEARQ (3) : ler arquivo especificado.

ESCARQ (4) : escrever no arquivo especificado.

ABREARQ (5) : abrir o arquivo especificado.

FECHARQ (6) : fechar o arquivo especificado.

POSICIONARQ (7) : posicionar em um endereço dentro do arquivo especificado.

LEIMAGEM (8) : abre um arquivo de imagens no hospedeiro e o transfere para o GSP. É feito o processamento de conversão dos dados e a imagem é mostrada em uma janela no GSP.

CONTINUA (9) : prepara o controle para a próxima operação em disco.

dado_disco - (16 bits) passa um dos parâmetros de uma operação em disco cujo significado depende da operação em andamento.

endereço_disco - (32 bits) passa um dos parâmetros de uma operação em disco cujo significado depende da operação em andamento.

auxiliar_disco - (32 bits) passa um dos parâmetros de uma operação em disco cujo significado depende da operação em andamento.

Figura 48 - Estrutura de dados LIE_comunica.

6.4.6 - Estrutura de Dados ArqSpec.

Esta estrutura de dados é utilizada para manipulações com arquivos. Armazena as informações do cabeçalho de um arquivo que são de utilidade para o usuário. Essas informações estão colocadas no campos descritos a seguir. A figura 49 mostra a estrutura ArqSpec.

fattr - (16 bits) indica o estado ou o tipo do arquivo.

bit 0: Arquivo R/O.

bit 1: Arquivo Oculto (Hidden File).

bit 2: Arquivo do Sistema.

bit 3: Nome do disco.

bit 4: Arquivo de subdiretório.

ftime - (16 bits) indica a hora de criação ou última modificação do arquivo. Está codificada segundo o padrão de hora do sistema DOS.

fdate - (16 bits) indica a data de criação ou última modificação do arquivo. Está codificada segundo o padrão de data do sistema DOS.

fsize - (32 bits) indica o tamanho do arquivo em bytes.

fname - (16 bytes) armazena o nome e extensão do arquivo.

Figura 49 - Estrutura de dados ArqSpec.

6.4.7 - Estruturas de Dados pdmenu e pdmdef.

Esta estrutura de dados armazena as informações necessárias para a construção de "menus". Foi construída para ser utilizada tanto por "menus" do tipo "pull-down" quanto "pop-up".

A figura 50 apresenta a estrutura pdmenu, a qual é utilizada como item básico para a construção do vetor pdmdef que armazena as informações para todos os "menus" utilizados por um aplicativo.

Uma estrutura pdmenu armazena os ponteiros para os vetores que contém as informações. Os vetores são passados pelos aplicativos, mas não são copiados em estruturas internas, para economia de memória. Isso significa que uma vez criados os "menus", os aplicativos devem evitar alterações nesses vetores, uma vez que são usados sempre que um "menu" é ativado. A seguir é feita a descrição dos campos da estrutura pdmenu:

cor - (4 x 32 bits) vetor de 4 posições que armazena as cores com que devem ser mostrados os seguintes elementos do menu:

cor[0] = armazena cor de fundo;

cor[1] = armazena cor da borda;

cor[2] = armazena cor dos textos regulares;

cor[3] = armazena cor do texto em destaque.

nro_entries - (32 bits) armazena o número máximo de entradas que o "menu" possui.

scr_size - (32 bits) armazena o número de entradas que são mostradas simultaneamente na tela (pois o "menu" pode sofrer rolamento ("scrool")).

entry - (32 bits) armazena o ponteiro para um vetor de cadeias. Cada cadeia deve conter o texto de uma entrada do menu.

cad - (32 bits) armazena o ponteiro para um vetor de cadeias alternativas. É usado apenas para "menus" "pop-up".

command - (32 bits) armazena o ponteiro para um vetor de inteiros, o qual indica para cada entrada do "menu" qual é o valor de retorno associado, que deve retornar se aquela entrada for a escolhida.

6.4.7 - Estruturas de Dados *pdmenu* e *pdmdef*.

Esta estrutura de dados armazena as informações necessárias para a construção de "menus". Foi construída para ser utilizada tanto por "menus" do tipo "pull-down" quanto "pop-up".

A figura 50 apresenta a estrutura *pdmenu*, a qual é utilizada como item básico para a construção do vetor *pdmdef* que armazena as informações para todos os "menus" utilizados por um aplicativo.

Uma estrutura *pdmenu* armazena os ponteiros para os vetores que contém as informações. Os vetores são passados pelos aplicativos, mas não são copiados em estruturas internas, para economia de memória. Isso significa que uma vez criados os "menus", os aplicativos devem evitar alterações nesses vetores, uma vez que são usados sempre que um "menu" é ativado. A seguir é feita a descrição dos campos da estrutura *pdmenu*:

cor - (4 x 32 bits) vetor de 4 posições que armazena as cores com que devem ser mostrados os seguintes elementos do menu:

cor[0] = armazena cor de fundo;

cor[1] = armazena cor da borda;

cor[2] = armazena cor dos textos regulares;

cor[3] = armazena cor do texto em destaque.

nro_entries - (32 bits) armazena o número máximo de entradas que o "menu" possui.

scr_size - (32 bits) armazena o número de entradas que são mostradas simultaneamente na tela (pois o "menu" pode sofrer rolagem ("scroll")).

entry - (32 bits) armazena o ponteiro para um vetor de cadeias. Cada cadeia deve conter o texto de uma entrada do menu.

cad - (32 bits) armazena o ponteiro para um vetor de cadeias alternativas. É usado apenas para "menus" "pop-up".

command - (32 bits) armazena o ponteiro para um vetor de inteiros, o qual indica para cada entrada do "menu" qual é o valor de retorno associado, que deve retornar se aquela entrada for a escolhida.

link - (32 bits) armazena o ponteiro para um vetor de inteiros que indica para cada entrada do menu, se existe um sub"menu" associado, e se existir qual é ele.

retorno - (32 bits) armazena o ponteiro para um vetor de inteiros que indica o valor de retorno associado a alguma entrada. É usado apenas para "menus" "pop-up".

poslin - (32 bits) armazena o ponteiro para um vetor de inteiros. É usado apenas em "menus" "pop-up" para, juntamente com o vetor **poscol** definir a posição de cada peça no "lay-out" do menu.

poscol - (32 bits) armazena o ponteiro para um vetor de inteiros. É usado apenas em "menus" "pop-up" para, juntamente com o vetor **poslin** definir a posição de cada peça no "lay-out" do menu.

cursor - (32 bits) indica qual a entrada que o cursor estava quando o "menu" foi encerrado na última vez em que foi chamado. Não pode ser usado pelos aplicativos.

inipos - (32 bits) indica qual a entrada que ocupava o topo do "menu" quando este foi encerrado pela última vez em que foi chamado. Não pode ser usado pelos aplicativos.

Figura 50 - Estrutura de dados pdmenu.

6.5 - Um Exemplo.

Nessa seção é apresentado um exemplo de um aplicativo construído utilizando esse ambiente. Nesse exemplo apresenta-se um "menu" "pull-down" principal com 5 opções:

LE IMAGEM - Caso essa opção seja a escolhida, é montado um sub"menu" "pull-down" com rolamento, em cujas entradas são listados os arquivos contidos no diretório corrente que possuem extensão PIC. Esses arquivos, por convenção, armazenam imagens. Ao usuário escolher uma imagem, o arquivo correspondente é aberto no hospedeiro e transmitido ao GSP. No GSP a imagem é colocada dentro de uma janela com borda, e o "menu" de opções volta a estar ativo.

MAXIMIZA HISTOGRAMA - Ao escolher essa opção, é efetuada uma operação de "alargamento da imagem". Ou seja é realizada uma redistribuição dos níveis de cinza da imagem permitindo-se uma melhor visualização. Este tipo de operação é comumente feito através de manipulação interativa do histograma da imagem. Devido a restrições de espaço de memória dessa versão, foi implementado apenas a operação dessa forma, sem interação com o usuário. A saturação é a máxima permitida, levando-se o nível mais alto da imagem ao nível mais alto permitido (nessa arquitetura é de 255), e o nível mais baixo em zero. Todos os demais níveis da imagem acompanham o deslocamento imposto pelos pontos limites.

APAGA - Apaga a janela com a cor de fundo especificada para essa janela.

SAI DE CIMA - Retira o "menu" da tela para poder-se visualizar inteiramente a tela. Ao teclar-se qualquer tecla o "menu" retorna a tela.

FIM - Finaliza o programa

A figura 51 apresenta uma seqüência de visualização de imagens utilizando o exemplo acima. A seguir é colocada a listagem desse programa exemplo.

Figura 51 - (a) Fotografia que apresenta o sistema como um todo.

Figura 51 (b) - Fotografia que apresenta a tela do aplicativo em execução.

Figura 51 (c) - Fotografia que apresenta a imagem anterior com manipulação do histograma.

Listagem do Programa Exemplo

```

/*****c
 * Programa para testar a comunicacao entre o PC (host) e o TMS
 *
 * E' usado a tecnica de "hand-shake".
 *
 * Testa a visualizacao de uma imagem a partir de um arquivo no host .
 *
 *****/

#include "data_str.h"
#include "lie_sgj.h"
#include "i_disco.h"

#define MAXARQIMAG 20 /* Este programa pode tratar ate 20 arq de imagem */
#define TAMNP 5 /* Tamanho do menu principal */

/*----- Variaveis Globais -----*/
extern lie_janela *lie_jan_corrente, raiz;
extern struct LIE_comunica comunica;

main()
{
    cadeia teste, kcv;
    static imag_xlo, imag_ylo, min, max;

    lie_janela *janl;
    int i, j, aux, aux2, nivel, erro;
    static int menu_idl, menu_idp, nlin=0, ncol=0;
    static int coml, comp;
    static int red[4] = {0,0x3,0x3,0x0},
               green[4] = {0,0x4,0x7,0x3},
               blue[4] = {0,0x1,0x7,0x3};
    static cadeia text_font = (cadeia)"-."; /* Inicializa a cadeia */

    static cadeia entryp1[MAXARQIMAG];
    static int commandl[MAXARQIMAG];
    static int linkl[MAXARQIMAG];
    ArqSpec arquivo[MAXARQIMAG];

    /* Definicao do menu principal */
    static cadeia entryp[] = {(cadeia)"-Le Imagem",
                              (cadeia)"-Maximiza Hist.",
                              (cadeia)"-Apaga",
                              (cadeia)"-Sai de Cima",
                              (cadeia)"-Fim"};

    static int commandp[] = {1, 5, 2, 3, 4};
    static int linkp[] = {0,0,0,0,0};

    Inicializa(); /* Inicializa o TMS e apaga a tela */
    Limpa_jan_corrente();

    /* Obter todos os arquivos com extencao .PIC, preparando o
       menu com os nomes dos arquivos */
    /* teste = kcv (teste); */
    teste[0]=6; teste[1]=5; teste[2]='*'; teste[3]='.';
    teste[4]='P'; teste[5]='I'; teste[6]='C';

```



```

i=i_arqinicio (teste, 23, &arquivo[0]);

if (i == 0) { /* Se tem ao menos um arquivo com extencao .PIC... */
    max=0;
    for (j=0; j<MAXARQIMAG - 1; j++) {
        entry1[j]=arquivo[j].fname;
        /* Limpar as cadeias onde vao os nomes */
        for (i=entry1[j][1]; i < entry1[j][0]; i++)
            entry[j][i+2]=' ';
        if (entry1[j][1]>max) max=entry1[j][1];
        command1[j]=TAMNP+1+j; /* O menu pai tem TAMNP entradas */
        link1[j]=0;
        i = i_arqproximo(&arquivo[j+1]);
        if (i!= 0) break;
    }

    /* Criar os menus */
    j++;
    for (i=0; i<j; i++) entry1[i][1]=max;

    Mn_PDcria (red, green, blue, text_font, j, j<8 ?j:8, entry1,command1,
                link1, &menu_id1);

    /* Estabelecer uma hierarquia */
    linkp[0]=menu_id1;
    Mn_PDcria (red, green, blue, text_font, TAMNP, TAMNP, entryp,
                commandp, linkp, &menu_idp);

    while (1) {
        i=Mn_PDprocessa (menu_idp, 0, 10, 1, &comp);
        if (i != 0) { /* Deu erro */
            i_printk (" Erro =%d \n", i);
            break;
        }

        if (comp==4) /* FIM */
            break;

        if (comp == 2) { /* Apaga a janela */
            Limpa_jan_corrente ();
            continue;
        }

        if (comp == 3) { /* Tira o menu de Cima */
            i_getc();
            continue;
        }

        if (comp > TAMNP) { /* Ler uma imagem */
            j=comp-(TAMNP+1);
            i=i_open(arquivo[j].fname, 0);

            /* A jan. da imagem sera aberta em (50,40)*/
            Jan_corrente (&raiz);
            imag_xlo=50; imag_ylo=40;
            erro=i_le_imagem (i,imag_xlo,imag_ylo,arquivo[j].fname,
                              &nlin, &ncol);

            i_close (i);
            continue;
        }
    }
}

```

```

        if (comp == 5) { /* Maximiza a Imagem (Aumenta contraste) */
            if (nlin==0) /* Nao foi lida nenhuma imagem */
                continue;
            /* Calcular minimo e maximo da Imagem */
            min=1000; max=0;
            for (i=0; i< nlin; i++)
                for (j=0; j<ncol; j++) {
                    aux=get_pixel(imag_xlo+j, imag_ylo+i);
                    if (aux > max) max=aux;
                    if (aux < min) min=aux;
                }
            aux2=max-min;
            for (i=0; i< nlin; i++)
                for (j=0; j<ncol; j++) {
                    aux=get_pixel(imag_xlo+j, imag_ylo+i);
                    nivel= (255 * (aux - min))/aux2;
                    put_pixel(nivel, imag_xlo+j, imag_ylo+i);
                }
            continue;
        } /* Fim do while (1) */
    } /* Fim do if <existe arquivo> */
    else { /* Nenhum arquivo com extencao .PIC */
        v_printk (" Nenhum arquivo .PIC\n");
    }

    i_printk ("\n\n          Processamento Encerrado.          ");
    exit (0);
}

```

6.6 - Conclusões.

Neste capítulo foi apresentado o ambiente gráfico desenvolvido para a arquitetura dedicada à visualização de imagens construída no LIE.

Este ambiente se destina a apoiar a construção de programas aplicativos que devem ser executados na arquitetura dedicada, provendo um conjunto de primitivas de software, as quais colocam à disposição do programador de aplicativos todos os recursos do hardware de uma maneira unificada e facilmente acessível através de primitivas de alto nível. Essas primitivas por sua vez foram implementadas de maneira a obter o máximo desempenho da arquitetura, tal como a utilização de Acesso Direto à Memória sempre que possível.

As primitivas foram construídas no sentido de estender a biblioteca padrão da linguagem "C", de maneira a facilitar seu aprendizado e utilização por parte de pessoas treinadas na utilização dessa linguagem. Assim, buscou-se também uma completeza na definição do conjunto de primitivas, buscando atender à necessidade de se dar apoio aos aplicativos que devam utilizar tal arquitetura para a visualização de imagens de uma forma genérica.

Embora tal arquitetura tenha a princípio capacidade computacional para suportar também o processamento (ao menos parcial) das imagens manipuladas, as restrições de memória e o tempo de processamento envolvidos tornam desvantajosa tal aplicação da arquitetura atual. Assim, não foram implementadas primitivas para o suporte a essas operações, pois considerou-se que esse processamento seria melhor efetuado no ambiente de processamento de imagens descrito no Capítulo 3. No entanto, é viável a ampliação da arquitetura de forma a tornar eficiente esse tipo de processamento (o que já encontra-se em fase de projeto no LIE), e nesse caso as primitivas já desenvolvidas para o ambiente Intergraph poderiam ser transportadas para esta arquitetura e integradas ao ambiente gráfico.

Capítulo VII

Conclusões Gerais e Linhas de Futuras Pesquisas

7.1 - Conclusões.

Apesar de todo o avanço tecnológico na área de sistemas digitais, atualmente, máquinas com poder computacional para suprir as necessidades de processamento, visualização e manipulação de imagens continuam relativamente caras. Além disso, muitas vezes todos os recursos oferecidos por ela, tais como: hardware dedicado a operações gráficas vetoriais e ambiente multiusuário; não são utilizados em uma única aplicação. Dessa forma, máquinas dedicadas são atraentes em tais sistemas, pois suprem as necessidades a elas requeridas (apesar de somente a elas) e são baratas quando comparadas às de propósito geral. Assim, optou-se por desenvolver uma arquitetura dedicada para a visualização de imagens tomográficas.

Este trabalho descreve o estudo e a implementação de um sistema de visualização de imagens tomográficas obtidas por RM, que é composto por dois ambientes. Um deles, o desenvolvimento de um sistema de visualização e processamento de imagens, onde várias operações para processamento e manipulação de imagens tomográficas foram implementadas e colocadas à disposição do usuário, através de uma interface amigável.

O conhecimento adquirido pelo estudo e implementação dessas técnicas de visualização de imagens, viabilizou a definição e a construção de um ambiente gráfico dedicado, que fornecesse ferramentas para a construção de sistemas de Visualização de Imagens, como o desenvolvido no primeiro ambiente. Este segundo ambiente, é então, um sistema de software construído para uma arquitetura dedicada, que oferece primitivas gráficas para a construção de sistemas interativos, os quais necessitam de interfaces com o usuário amigáveis. Neste ambiente, procurou-se explorar a disponibilidade de características específicas de hardware, para o tratamento de operações gráficas existentes no sistema. Assim pôde-se construir um ambiente gráfico que poderá ser utilizado por um projetista de software, para a construção de sistemas interativos de visualização de

imagens obtidas por ToRM. Para o desenvolvimento deste ambiente gráfico, seguiu-se o padrão visual do ambiente ENVIRON V, da Intergraph. Neste ambiente, a principal forma de interação é através de ícones, aos quais são associados "menus", sendo que esta forma de interface é bastante utilizada em sistemas interativos.

A arquitetura dedicada baseada no processador gráfico TMS34010, foi desenvolvido especialmente para visualização de imagens por ToRM. No entanto, devido às limitações atuais desse protótipo, principalmente no que tange a disponibilidade de memória, essa arquitetura não apresenta condições adequadas para suportar também todo o ambiente de processamento de imagens. O desenvolvimento e a implementação do sistema de Processamento e Visualização de Imagens foi feito de forma modular, levando-se em conta que posteriormente poderia ser transportado para outras arquiteturas de hardware.

Pode-se dizer que o sistema de software resultante aproveita bem as características das plataformas utilizadas. As primitivas básicas do ambiente ENVIRON V, utilizado para a implementação do sistema na plataforma comercial, foram implementadas no ambiente gráfico desenvolvido para a arquitetura dedicada, que por sua vez aproveita intensamente os recursos gráficos do processador TMS34010.

O projeto do ambiente de visualização de imagens, desenvolvido na plataforma computacional dedicada, foi feito inicialmente utilizando-se técnicas de análise e projeto estruturado. No entanto, isso mostrou-se desvantajoso quando da tentativa de implementação do projeto resultante na arquitetura dedicada. Isso porque, o suporte de funções de alto nível oferecido pela arquitetura não era contemplado pelo projeto. Analisando-se a situação, chegou-se a conclusão de que as técnicas tradicionais de análise e projeto de sistemas de software não são adequadas quando a arquitetura provê um conjunto pré-determinado, não configurável, de primitivas básicas que executam funções de alto nível. Tais técnicas pressupõem que a construção dos módulos e funções seja feita de acordo com as necessidades especificadas, ao passo que em tais arquiteturas, para se aproveitar o suporte de alto nível oferecido, as funções e módulos desenvolvidos devem se conformar às características impostas para o uso desse suporte.

Assim, decorre que utilizando-se das técnicas tradicionais, o software resultante não permite explorar as características da arquitetura, as quais passam a ser sub-utilizadas. Em consequência dessa constatação, e da necessidade de se obter um sistema eficiente em função dos recursos disponíveis, o projeto do ambiente foi refeito, aproveitando-se do projeto anterior apenas as partes que, uma vez confrontadas com a implementação voltada

para essa arquitetura em particular, mostrassem uma alta taxa de aproveitamento dos recursos oferecidos. Pode-se dizer então que foram empregadas técnicas opostas às normalmente utilizadas, onde partiu-se da implementação para então analisar-se o empacotamento de funções já efetuado, e em seguida, com as funções desmembradas obter-se um diagrama de funções que permitiu a validação da implementação com a especificação do ambiente que se tinha por objetivo.

Na prática diária da utilização de sistemas de visualização de imagens, pôde-se notar que sistemas de visualização muito sofisticados, ou foram prontamente descartados, ou tiveram suas capacidades subutilizadas, pois o médico acabava se "perdendo" na sua utilização. Assim, um fato importante é que os sistemas interativos para aplicações desse tipo devem procurar manter o ambiente familiar ao usuário. Ou seja, múltiplas janelas sobreponíveis com manipulação sofisticada, podem se tornar um estorvo em vez de um benefício do sistema. Dessa forma, tem-se procurado desenvolver sistemas interativos de visualização e manipulação de imagens que realmente atendam às necessidades dos radiologistas, e que sejam de fácil aprendizado [FIN_90].

Outro ponto é o grau de conhecimento do usuário. Para sistemas de diagnóstico, onde usualmente são radiologistas acostumados a visualizar imagens com aparência de radiografia por Raio-X convencional, sistemas que permitam a manipulação de cores não monocromáticas usualmente não são bem aceitos; e mesmo o processo de reconstrução 3D é subutilizado. Já para sistemas de classificação e auxílio a cirurgia (*Computer Assisted Surgery - CAS*), sistemas mais complexos, que além de permitirem a visualização 3D das imagens, também permitem cortes em vários ângulos, para estudo prévio da anatomia interna da amostra estão sendo grandemente utilizados em importantes centros cirúrgicos [ADA_90]. Para utilizar tais sistemas investe-se no aprendizado específico de seus usuários.

Acreditamos assim ter demonstrado a validade do projeto de um sistema dedicado para visualização de imagens obtidas por ToRM, bem como ter projetado toda a base em nível de ambiente gráfico, para a implementação de um sistema de visualização de baixo custo, alta resolução e que permite a construção de interfaces H/M amigáveis.

A construção desse sistema computacional preenche assim uma etapa de capacitação necessária para o desenvolvimento de um sistema tomográfico por RM completamente nacional.

7.2 - Linhas de Futuras Pesquisas

Pode-se dizer que o âmago deste trabalho consiste na avaliação e no desenvolvimento de um sistema de visualização e manipulação de imagens tomográficas. Esse sistema foi construído de forma modular, levando-se em conta que deveria ser capaz de receber extensões, e também que poderia vir a ser transferido para outras arquiteturas. A seguir são apresentados alguns projetos que deverão dar continuidade ao trabalho.

7.2.1 - Extensão do Ambiente Gráfico.

A atual configuração da arquitetura dedicada em termos de memória de trabalho (3 bancos de 128 Kbytes), restringe a forma como o ambiente gráfico pode executar suas funções. Mesmo as operações de sobreposição de janelas ficam limitadas, pois ao levar-se em conta que é necessário 128 Kbytes para armazenar-se uma imagem de 256x512 pixels (cada pixel com 8 bits), pode-se deduzir que sistemas interativos necessitam de uma capacidade de memória muito grande. Leve-se em consideração, por exemplo, o fato de que a sobreposição dessa imagem inteira por uma única janela consome no mínimo 128 Kbytes de memória de trabalho (sistema) para esta operação.

Assim, uma necessidade urgente para que o ambiente gráfico possa ser utilizado plenamente, e possa suportar outros procedimentos interativos, consiste na expansão da memória disponível.

Devido a tais restrições, alguns elementos de interação não foram implementados, apesar de já serem previstos nas estruturas que suportam o ambiente já implementado, bem como por seu núcleo. Entre esses elementos incluem-se primitivas para manipulação de eventos, "menus" "pop-up" e ícones. Como esta extensão à arquitetura dedicada já está sendo realizada, em breve tornar-se-á possível incluir tais características ao ambiente gráfico.

7.2.2 - Processamento de Imagens em Arquitetura Paralela.

O processamento e a manipulação de imagens demanda um poder computacional considerável. O processador utilizado atualmente (TMS34010) não supre essa demanda.

Assim é necessário a utilização de arquiteturas com maior poder computacional, para efetuar a parte de cálculo necessária à reconstrução da imagem e de seu posterior processamento. Uma possibilidade é a utilização de arquiteturas paralelas, empregando-se processadores dedicados a esta tarefa. No Grupo de Instrumentação Eletrônica do DFCM - IFQSC, está sendo projetada uma arquitetura baseada em *Transputers* para este fim.

Para que a transferência de dados entre o GSP e um hospedeiro com alta capacidade de processamento de dados não seja um gargalo, está em consideração a utilização de um membro mais novo da família TMS340, que é o TMS34020. Esta escolha é natural, devido a compatibilidade de software, e por o TMS34020 possuir um poder computacional de 6 a 50 vezes maior do que o TMS34010, e maior rapidez para transferência de dados, pois tanto o duto de dados interno quanto o externo é de 32 bits [PET_88].

7.3.3 - Utilização de Sistemas Especialistas como Auxílio ao Diagnóstico.

A utilização de sistemas integrados de visualização e classificação de imagens é uma das formas de se colocar a disposição do profissional de medicina ferramentas automáticas que auxiliam a formação de diagnóstico [MAG_91]. Para isso é necessário que o sistema reconheça características específicas em imagens, que possam alimentar um motor de inferência, apoiado em uma base de conhecimentos. Esse enfoque, leva à necessidade de integrar ao sistema de processamento e visualização de imagens, ferramentas apoiadas em técnicas de inteligência artificial [KRI_91]. É também necessário um gerenciador de bases de dados, apoiado em modelos não convencionais, que permitam organizar e recuperar as informações de forma a que o médico possa construir um diagnóstico utilizando-se de comparações com situações semelhantes, previamente armazenadas no sistema, sem que haja necessidade que o próprio médico manipule um volume muito grande de informações.

O gerenciador de bases de dados, deve dispor de recursos para armazenagem eficiente de imagens, bem como deve prover meios de recuperação de imagens baseados em características facilmente definíveis e bem compreensíveis por parte do usuário. Deve possibilitar a integração de imagens, com outras informações, tais como: textos,

informações gráficas geométricas, valores numéricos, etc. Daí a necessidade de gerenciadores apoiados em modelos mais flexíveis e poderosos do que os tradicionais.

Por outro lado, a quantidade de informações que um sistema desse tipo pode vir a armazenar é muito grande, e o escrutínio delas, em busca de dados e imagens que se referem a determinado aspecto de interesse, em um determinado instante pode ser tão trabalhoso, que torna o sistema inutilizável. A tarefa de reconhecimento dos aspectos relevantes das informações armazenadas, e a respectiva triagem para que apenas aquelas de real interesse sejam apresentadas, deve ser levada a cabo por um sistema especialista, que opere sobre regras que representem o conhecimento de como as operações de reconhecimento e triagem devem ser efetuadas.

Um tal sistema representa um auxílio com as mesmas características de uma junta médica, onde diversas especialidades de trabalho médico podem ser avaliadas através de diferentes conjuntos de regras. Isso propicia ao médico usuário do sistema, uma análise equivalente a diferentes pontos de vista, guiando-o na formulação de um diagnóstico amparado em estudos de casos anteriores.

Existem grupos de pesquisa neste campus trabalhando em sistemas especialistas e em sistemas de bases de dados não convencionais (Grupo de Inteligência Artificial e Grupo de Base de Dados e Engenharia de Software). Esses grupos têm desenvolvido sistemas que podem ser integrados ao sistema de processamento e visualização de imagens descrito neste trabalho, propiciando a construção de um sistema integrado que suporte essas facilidades.

7.2.4 - Visualização de Imagens em Três Dimensões (3D).

A apresentação de imagens em 3D, tem sido um grande auxílio à medicina. Isso porque detalhes anatômicos complexos podem ser mais facilmente visualizados dessa forma, do que em uma imagem visualizada em apenas 2D. No entanto ainda atualmente, os radiologistas são treinados para, a partir de cortes em 2D, mentalmente formarem um modelo mental desses dados em 3D. Embora técnicas de visualização de imagens em 2D sejam adequadas para interpretação pelos radiologistas, estes têm dificuldade para transmitir este modelo mental para o pessoal envolvido no tratamento, que efetivamente deverá cuidar do paciente.

A atividade do radiologista envolve apenas parte do trabalho médico, uma vez que

seguindo ao diagnóstico radiológico, o paciente deve ser submetido a tratamento por outro médico, que usualmente não tem o mesmo treinamento para formar uma imagem mental das imagens visualizadas em 2D. Imagens em 3D apresentam detalhes anatômicos em um formato facilmente compreensível, sem treinamento especializado ou experiência anterior [NEY_90].

Por estas considerações pode-se perceber que uma das formas mais promissoras de auxílio ao usuário de um sistema de visualização de imagens, é justamente a capacidade de vê-las em 3D. As principais restrições à utilização de tais imagens são: o tempo de processamento capaz de produzi-las; e os terminais de vídeo de alta qualidade (alta resolução espacial, e capacidade de permitir a visualização de pelo menos 256 níveis de quantização), para poderem suportá-las. Estas restrições estão sendo ultrapassadas através de processadores poderosos e terminais gráficos sofisticados, que têm-se tornado bastante acessíveis [STA_91].

O projeto do tomógrafo por RM em desenvolvimento no IFQSC, pretende contemplar também esta forma de visualização de imagens. Para que isso seja feito, dois pré-requisitos tornam-se necessários. O primeiro consiste no desenvolvimento de software que incorpore técnicas de *rendering* volumétrico [NEY_90] [LEV_90] [TIE_90]. O segundo consiste na obtenção dos dados pelo tomógrafo, de forma a se poder reconstruí-los em 3D. Nesse aspecto a técnica de obtenção de dados por ToRM é bastante adequada, pois pode-se obtê-los nas três dimensões, o que facilita extremamente o processo.

Bibliografia

- [ACQ_82] Acquah, J.; Foley, J.; Sibert, J.; Wenner, P.: **"A Conceptual of Raster Graphics Systems"**, ACM Computer Graphics, Vol. 16, No. 3, pp. 321-328, julho de 1982.
- [ADA_90] Adams, L.: et alli: **"Computer-Assited Surgery"**, IEEE Computer Graphics & Applications, Vol. 10, No.3, pp. 43-51, maio de 1990.
- [BAR_91] Barkakati, N.: **X Window System Programming**, SAMS - Macmillan Computer Publishing, Carmel, Indiana, 1991.
- [BRI_90] Brittain, D. L.: **"Portability of Interactive Graphics Software"**, IEEE Computer Graphics & Applications, Vol. 10, No. 4, pp. 70-75, julho de 1990.
- [CÂM_83] Câmara Neto, G.; Mascarenhas, N. D. A.: **"Methods for Image Interpolation through FIR Filter Design Techniques"**, Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, abril 1983.
- [CÂM_82] Câmara Neto, G.; Mascarenhas, N. D. A., Mendes, C. L.: **"Projeto de Interpoladores em Imagens Digitais por Meio de Métodos de Janelamento"**, in Anais da 34a. Reunião Anual da SBPC, julho de 1982.
- [CAS_79] Castleman, K. R.: **Digital Image Processing**, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1979.
- [DAW_87] Dawson, B. M.: **"Introduction to Image Processing Algorithms"**, Byte, pp. 169-186, março de 1987.
- [FIN_90] Finlayson-Dutton, G.: **"The Current State of PACS"**, Applied Radiology, pp. 15-19, agosto 1990.
- [FOL_90] Foley, J.D.; van Dam, A.; Feiner, S.K.; Hughes, J.F.: **Computer Graphics - Principles and Practice**, Second Edition, Addison-Wesley Publishing Company, 1990.
- [GON_87] Gonzalez, R.C.; Wintz, P.: **Digital Imagem Processing**, Second Edition, Addison-Wesley Publishing Company, 1987.
- [HEL_90] Heller, D.: **"XView Programming Manual"**, O'Reilly & Associates, INC., CA, 1990.
- [HIX_89] Hix, D.: **"User Interfaces: Opening a Window on the Computer"**, IEEE Software, pp. 8-10, janeiro de 1989.
- [INT_88a] Intergraph Corp.: **Environ V - C Programmer's Reference Manual**, Huntsville, Al, 1988.
- [INT_88b] Intergraph Corp.: **CLIX System Guide**, Huntsville, Al, 1988.

- [JAI_89] Jain, A.K.: *Fundamentals of Digital Image Processing*, Prentice-Hall Inc., 1989.
- [JOH_89] Johnson, J: et alli: "*The Xerox Star a Retrospective*", IEEE Computer, Vol. 22, No. 9, pp 11-26, setembro de 1989.
- [JOH_90] Johnson, E. F.; Reichard, K.: *Advanced X Window Applications Programming*", MIS - Management Information Source, Inc. Press, 1990.
- [KER_78] Kerningham, B.; Ritchie, D.: *The C Programming Language*, Prentice-Hall, Englewood Cliffs, New Jersey, 1978.
- [KER_88] Kerningham, B.; Ritchie, D.: *The C Programming Language - 2ª Edition*, Prentice-Hall, Englewood Cliffs, New Jersey,
- [KRI_91] Kriewall, T.J.; Long, J.M.: "*Computer-Based Medical Systems*", IEEE Computer, Vol. 24, No. 3, pp. 9-12.
- [LEE_89] Lee, Ed: *User-interface Development Tools*", IEEE Software, pp. 31-36, maio de 1989.
- [LEE_90] Lee, M. E.; Redner, R.A.: "*A Note on the Use of Nonlinear Filtering in Computer Graphics*", IEEE Computer Graphics & Applications, Vol. 10, No. 3, pp. 23-29, maio de 1990.
- [LEV_90] Levoy, M.: "*A Hybrid Ray Tracer for Rendering Polygon and a Volume Data* ", IEEE Computer Graphics & Applications, Vol. 10, No. 2. pp. 33-40, março de 1990.
- [LEW_90] Lewis, T. G.; Oman, P. W.: "*The Challenge of Software Development*", IEEE Software, pp. 9-12, novembro de 1990.
- [MAG_86] Magalhães, L.P.: *Computação Gráfica - Interfaces em Computação Gráfica*, I Escola Brasileiro-Argentina de Informática, UNICAMP, 1986.
- [MAG_91] Maguire Jr., G.Q.; et alli: "*Graphics Applied to Medical Image Registration*", IEEE Computer Graphics & Applications, pp. 20-27, Vol 11, No. 2, março de 1991.
- [MAS_89] Mascarenhas, N.D.A.; Velasco, F.R.D.: *Processamento Digital de Imagens*, IV Escola Brasileiro-Argentina de Informática, Universidade Católica de Santiago del Estero, Termas do Rio Hondo - Argentina, 1989.
- [MYE_88] Myers, B.D.: "*A Taxonomy of Window Manager User Interfaces*", IEEE Computer Graphics & Applications, Vol. 8, No. 5. pp. 65-84, setembro de 1988.
- [MYE_89] Myers, B.D.: "*User-Interface Tools: Introduction and Survey*", IEEE Software, pp. 15-23, janeiro de 1989.

- [NEY_90] Ney, D.R.; et alli: **"Volumetric Rendering of Computed Tomography Data: Principles and Techniques"**, IEEE Computer Graphics & Applications, Vol. 10, No. 2. pp. 24-32, março de 1990.
- [PAI_90] Paiva, M.S.V.: **"Projeto de Uma Arquitetura de Hardware para Visualização de Imagens Digitais"**, tese de doutoramento apresentada ao Instituto de Física e Química de São Carlos - USP, 20/11/90.
- [PAN_85] Panepucci, H.; Donoso, J.P.; Tannús, A.; Beckamann, N.; Bonagamba, T.: **"Novas Imagens do Corpo"**, Ciência Hoje, Vol 20, No. 4, pp. 46-56, setembro/outubro de 1985.
- [PAN_87] Panepucci, H.; Beckamann, N.; Tannús, A.; Bonagamba, T. J.: **"NMR Imaging In: Recent Advances in Organic NMR Spectroscopy"**, editores J. B. Lambert e Roberto Rittner, Norell Press (USA) 1987, cap. 13, pp. 171-181.
- [PAV_82] Pavlidis, T.: **Algorithms for Graphics and Image Processing**, Computer Science Press, Inc., 1982.
- [PET_88] Peterson, Ron; et alli: **"Taking the Wraps off the 34020"**, Byte, Vol. 13, No. 9, pp. 257-272, setembro de 1988.
- [PRA_78] Pratt, W.K.: **Digital Image Processing**, Wiley-Interscience Publication, John Willey & Sons, 1978.
- [RAD_90] Radcliffe, M.J.: **DESQview/X Extending DOS into de 21st Century**, Quarterdeck Office Systems, Pico Boulevard, Santa Monica, CA, 1990.
- [RIT_90] Ritter, G. X.: **"Image Algebra: An Overview"**, Computer Vision, Graphics, and Image Processing, 49, pp. 297-331, 1990.
- [RHO_90] Rhodes, M. L.: **"Computer Graphics in Medicine"**, IEEE Computer Graphics & Applications, Vol. 10, No. 2, pp. 20-24, março de 1990.
- [RHO_91] Rhodes, M. L.: **"Computer Graphics in Medicine: The Past Decade"**, IEEE Computer Graphics & Applications, Vol. 11, No. 1, pp. 42-44, janeiro de 1991.
- [ROB_91] Robinson, K.; et alli: **"Graphics Workstations: A European Perspective"**, IEEE Computer Graphics & Applications, Vol. 11, No. 2, pp. 91-103, março de 1991.
- [ROS_90] Rosenfeld, A.: **"Image Analysis and Computer Vision: 1989"**, Computer Vision, Graphics, and Image Processing, pp. 188-240, 1990.
- [SAN_85] Santellano, J.; Barrera, J.: **"Sistemas de Processamento de Imagens"**, Publicação INPE-3686-PRE/833, outubro de 1985.

- [SCH_86] Scheifler, R.W.; Gettys, J.: *"The X Window System"*, ACM Transactions on Graphics, Vol. 5, No. 2, pp. 79-109, abril de 1986.
- [SCD_89] Schildt, H.: *C - Completo e Total*, Makron Books do Brasil, 1989.
- [SLA_89a] Slaets, J. W.; Paiva, M. S.; Almeida, L. O.: *"O Processador Gráfico TMS34010 - Uma Arquitetura para Visualização de Imagem em Tomografia por RMN"*, in Anais do 2o. Simpósio Brasileiro de Computação Gráfica e Processamento de Imagem, pp. 236-245, Águas de Lindóia, SP, abril de 1989.
- [SLA_89b] Slaets, J. F. W.; Traina, A. J. M.: *"Um Sistema de Processamento de Imagens para Tomografia Computadorizada por RMN"*, in Anais do II Simpósio Brasileiro de Computação Gráfica e Processamento de Imagens, pp. 281-290, Águas de Lindóia, SP, abril de 1989.
- [STA_91] Staudhammer, J.: *"Computer Graphics Hardware"*, IEEE Computer Graphics & Applications, Vol. 11, No. 1, pp. 42-44, janeiro de 1991.
- [SUN_87] Sun Microsystems: *NeWS Technical Overview*, Sun Microsystems, Mountain View, CA, 1990.
- [SUN_89a] Sun Microsystems: *OPEN LOOK Graphical User Interface Functional Specification*, Addison-Wesley Publishing Company, 1989.
- [SUN_89b] Sun Microsystems: *OpenWindows Developer's Guide 1.1 - Installation Manual*, Sun Microsystems, Mountain View, CA, 1990.
- [SUN_90a] Sun Microsystems: *Sun Software Configuration Guide*, Sun Microsystems, Mountain View, CA, 1990.
- [SUN_90b] Sun Microsystems: *OPEN LOOK Graphical User Interface Application Style Guidelines*, Addison-Wesley Publishing Company, 1990.
- [SUN_90c] Sun Microsystems: *X11/NeWS Version 2 Server Guide*, Sun Microsystems, Mountain View, CA, 1990.
- [TAN_87] Tannús, A. : *"Desenvolvimento de Tecnologia de Tomografia por Ressonância Magnética Nuclear"*, tese de doutoramento apresentada ao IFQSC-USP, agosto de 1987.
- [TEX_86a] Texas Instruments: *TMS34010 User's Guide*, 1986.
- [TEX_86b] Texas Instruments: *TMS34010 Software Development Board User's Guide*, 1986.
- [TEX_87] Texas Instruments: *TMS34010 Assembly Language Tools User's Guide*, 1987.

- [TIE_91] Tiede, U.; et alli: *"Investigation of Medical 3D-Rendering Algorithms"*, IEEE Computer Graphics & Applications, Vol. 10, No. 2. pp. 41-53, março de 1990.
- [TRA_89] Traina, A. J. M.; Slaets, J. F.: *"Desenvolvimento de 'Software' Básico para o Processamento e Visualização de Imagens Tomográficas Obtidas por RM"*, Revista Brasileira de Engenharia Biomédica, Vol. 6, No. 2, pp. 116-122.
- [TRA_90] Traina Jr., C.: *"Sistemas de Gerenciamento de Bases de Dados e Sistemas de Gerenciamento de Interface com o Usuário: Um Modelo Genérico para a Integração Conceitual e Física"*, tese de Livre-docência apresentada ao ICMSC-USP, junho 1990.
- [UPS_89] Upson, C.; et alli: *"The Application Visualization System: A Computacional Environment for Scientific Visualization"*, IEEE Computer Graphics & Applications, Vol. 9, No. 4, pp. 30-42, julho de 1989.
- [WIL_84] Willians, G.: *"The Apple MacIntosh Computer"*, Byte, Vol. 9, No. 2, fevereiro de 1984.
- [WIL_91] Wilson, P.R.: *"Standards: Past Tense and Future Perfect?"*, IEEE Computer Graphics & Applications, Vol. 11, No. 1, pp. 44-49, janeiro de 1991.
- [WIL_91a] Wilson, M.; Conway, A.: *"Enhanced Iteration Styles for User Interfaces"*, IEEE Computer Graphics & Applications, Vol. 11, No. 2, pp. 79-90, março de 1991.

Apêndice A

Listagem do Sistema de Processamento de Imagens


```

/*****
*
* Este programa faz parte do projeto da FINEC numero 11378-1
* Junto ao Instituto de Física e Química de São Carlos - USP
* intitulado:
* "Desenvolvimento de um Sistema Modular para Processamento Grafico"
*
*****/

#include <stdio.h>
#include <tools.h>
#include <math.h>
/*-----
Versao 2 : Data de implementacao: 02-06-88

Implementando a aceitacao de selecao de tonalidades atraves
da tabela de cores (versao 1.2).

Alteracoes: 22-12-88
    Efetua o Processamento para varios tamanhos de imagens.

    13-02-89
    Implementacao de filtros Lineares
        Passa_baixas
        Passa_altas

    Implementacao de filtros nao Lineares
        Roberts
        Mediana

    07-03-89
    Implementacao das Operacoes de Composicao de Imagens
        (Trabalha com tres imagens: IA,IB e IC)

    17-11-89
    Implementacao de medidas estatisticas de sobre regioes das
    imagens:
        - Media Aritmetica
        - Desvio Padrao

    17-01-90
    Implementacao de medidas de relacao sinal ruido.

-----*/

/*****
*
* Este programa gera uma imagem em 128 niveis de tonalidade
* a partir de uma tabela de niveis de intensidade, que vai de
* 0 a 127. As imagens possuem 256 colunas por 128 linhas.
*
* Obtem-se os 128 niveis de tonalidade a partir dos 16 niveis
* de tonalidades de cinza puros atraves da tecnica dos "meios-tons"
* (half-tonning) com pontos da imagem mapeados em blocos de pixels
* de tamanho 3x3.
*
* Ha 3 opcoes implementadas de manipulacao da imagem:
*
* 1 - Desenha a imagem obtida
*
* 2 - Fornece a tabela de tonalidades de cinza utilizadas, e per-
* mite alterar qualquer tonalidade interativamente.
*
* 3 - Fornece a apresentacao do histograma normalizado dos ni-
* veis de intensidade dos pontos da imagem.
*
*****/

#define TEXT_FONT      "/usr/ib32/resrc/rfont/pulaski.10"
#define NONIE          "/usr/ib32/resrc/rfont/nonie.i.12"
#define FIX            "/usr/ib32/resrc/rfont/fix.9"
#define ABS(x)         (((x) < 0) ? -(x) : (x))

```

```

#define MAX(x,y)       (((x) > (y)) ? (x) : (y))
#define NLINHAS        512
#define NCOLUNAS       512
#define DIM_MASC        3
#define PI              3.141592653

static char *buffer1, *buffer2, *buffhist,*buff_img;
unsigned bufsize1, bufsize2, bimsize, bhistsize;
char *malloc();
FILE *fopen();

int kn_count=50;
char kn_buffer[50];

int cur_event_mask, refr_mask;

int vs_no,vsr_no, w_no, wt_no, wh_no,wr_no,
    aleat, larg_pix;
float baixas[NLINHAS][NCOLUNAS], altas[NLINHAS][NCOLUNAS];
float ajeita[NCOLUNAS];
int janela, button, transition, time_t;

int fhist;
int flag_IA;
int flagtc, flagimg, flaghist, flag_hist_alt;
int mb_cmd, x, y, xm, ym, font, font2;
int xlör, ylor, xhir, yhir;
char ctonalid[3];
char cindex[3];

int xmin=1, ymin=1,
    inicx, inicy;
int cor[128], cor_alt[128], hist[16], hist_alt[16];
FILE *fopen(),*imagem;

/* Especificacao da tabelas de cinzas */
int red[] = {0,0,0,2,3,4,5,6,7,8,9,10,11,12,13,14,15, 0,1,2,3,4,5,6,7,8,9,
    10,15,0,12,15};
int green[] = {5,4,0,2,3,4,5,6,7,8,9,10,11,12,13,14,15, 0,1,2,3,4,5,6,7,8,
    9,10,1,15,15,15};
int blue[] = {6,5,0,2,3,4,5,6,7,8,9,10,11,12,13,14,15, 1,2,3,4,5,6,7,8,9,
    10,11,1,0,14,15};

int inaltred[] = {0,0,0,2,3,4,5,6,7,8,9,10,11,12,13,14,15,
    0,1,2,3,4,5,6,7,8,9,10,15,0,12,15};
int inaltgreen[] = {5,4,0,2,3,4,5,6,7,8,9,10,11,12,13,14,15,
    0,1,2,3,4,5,6,7,8,9,10,1,15,15,15};
int inaltblue[] = {6,5,0,2,3,4,5,6,7,8,9,10,11,12,13,14,15,
    1,2,3,4,5,6,7,8,9,10,11,1,0,14,15};

```

```

/*-----
Esta rotina calcula o espaco necessario da memoria para guardar uma
linha da tela.
-----*/
get_put_line_size (x1,y1,x2,y2)
int x1,y1,x2,y2;
{
    return (MAX (ABS (x1-x2), ABS (y1-y2)) + 1);
}

/*-----
Esta rotina calcula o espaco necessario da memoria para guardar uma
regiao da tela
-----*/
get_put_block_size (xlo,ylo,xhi,yhi,numplanes)
int xlo,ylo,xhi,yhi;
int numplanes;
{
    int xsize, ysize;
    int temp;

    xsize = (ABS (xlo-xhi)+8)/8;
    ysize = ABS (ylo-yhi) + 1;
    temp = xsize * ysize;
    return ((temp+(temp+1))*numplanes);
} /* fim da rotina get_put_block_size */

/*-----
Esta rotina recebe um valor numerico em uma cadeia de caracteres
e a converte para um valor real (float)
-----*/
float Hoyia_num()
{
    float num;

    Clear_keyin_data();
    Wait_for_events (KEYIN_EVENT, &cur_event_mask);
    if (KEYIN_EVENT & cur_event_mask) {
        Get_keyin_data (&kn_count);
        sscanf (kn_buffer, "%f", &num);
    }

    return (num);
}

/*-----
Esta rotina gera outros tons de cinza a partir do ton de cinza
basico através da tecnica de "micio-ton" (halfiftoing), usando
bloco de 3 pixels de largura (3x3), e gerado 8 tons a partir de
um basico, ou seja
| 0 0 0| | 0 0 0| | 0 0 0| | 0 0 0| | 1 0 1| | 1 0 1| | |
| 0 0 0| | 0 1 0| | 1 1 0| | 1 1 0| | 1 1 1| | 1 1 1| | 1 1 1|
| 0 0 0| | 0 0 0| | 0 0 0| | 0 1 0| | 0 1 0| | 1 1 0| | 1 1 1|
-----*/
pinto (w_no, cor, x, y)
int w_no, cor, x, y;
{
    int l, resto, intensidade;

    intensidade=cor/8; /* Um entre os 16 niveis de intensidade */
    resto=cor%8;
    intensidade++; /* Incrementa intensidade para deslocar de 1 a VLT */

    weight (w_no,0); /* Largura do ponto e igual a 1 */
    if (intensidade == 16) fgcolor (w_no, intensidade - 1);
    else fgcolor (w_no, intensidade + 1);
    for (i=0; i< resto; ++i)
    {
        switch (aleat) {

```

```

case 0: pnt(w_no, x, y); /* | 0 0 0| */
        break; /* | 0 1 0| */
case 1: pnt(w_no, x-1, y); /* | 0 0 0| */
        break; /* | 1 0 0| */
case 2: pnt(w_no, x, y+1); /* | 0 0 0| */
        break; /* | 0 0 0| */
case 3: pnt(w_no, x+1, y); /* | 0 0 0| */
        break; /* | 0 0 1| */
case 4: pnt(w_no, x+1, y-1); /* | 0 0 1| */
        break; /* | 0 0 0| */
case 5: pnt(w_no, x-1, y+1); /* | 0 0 0| */
        break; /* | 0 0 0| */
case 6: pnt(w_no, x+1, y-1); /* | 0 0 1| */
        break; /* | 0 0 0| */
case 7: pnt(w_no, x-1, y-1); /* | 1 0 0| */
        break; /* | 0 0 0| */

    } /* Fim do case */
    aleat=(aleat == 7) ? 0: aleat+1;
} /* Fim da rotina pinte */

/*-----
Esta rotina manipula a tabela de cores (VLT) especificada, alterando a
sua composicao.
-----*/
tabcor(x,y)
int x,y;
{
    static int ton;
    static int flag;

    /* Se nao e a primeira entrada colocar o bloco no lugar */
    if (flagtc) {
        hidecursor(wt_no);
        putblock (wt_no, 31L, 23*ton, 5, 23*(ton+1), 70, buffer1, bufsizel);
        free (buffer1);
        flagtc=0; /* Especifica que o buffer foi desalocado */
        showcursor (wt_no);
        flushbuffer (wt_no);
    }

    /* Ha 3 faixas (pensando-se no eixo y) que devem ser testadas na tab */
    if ((y<=70) && (y>=5)) { /* E selecao de uma tonalidade de cinza */
        ton=x/23;
        if (ton > 15) /* Tonalidade inexistente */
            return (1);

        /* Guardar a memory raster e desenhar um quadrado na tonalidade */
        hidecursor(wt_no);
        bufsizel=get_put_block_size (23*ton, 5, 23*(ton+1), 70, 5);
        if ((buffer1=malloc(bufsizel)) ==0) {
            printf ("Nao conseguiu alocar espaco (1)");
            return (1);
        }
        flagtc=1; /* Especifica que ha buffer1 alocado */
    }

```

```

getblock (wt_no, 31L, 23*ton, 5, 23*(ton+1), 70, buffer1, bufsize1);
fgcolor(wt_no, 29);
move (wt_no, 23*ton, 70); /* ponto inicial do quadrado */
draw (wt_no, 23*(ton+1), 70); /* desenha primeira linha */
draw (wt_no, 23*(ton+1), 5); /* desenha segunda linha */
draw (wt_no, 23*ton, 5); /* desenha terceira linha */
draw (wt_no, 23*ton, 70); /* desenha quarta linha */
flushbuffer (wt_no);

/* Preenche o campo TONALIDADE */
putblock (wt_no, 31L, 290, 300, 330, 320, buffer2, bufsize2);
sprintf (ctonalid, "%d", ton);
drawsymbols (wt_no, font, 290, 100, ctonalid, strlen(ctonalid));
flushbuffer (wt_no);

/* Preenche os valores de VERMELHO, VERDE e AZUL */
putblock (wt_no, 31L, 20, 240, 60, 260, buffer2, bufsize2);
sprintf (cindex, "%d", red[ton+1]);
drawsymbols (wt_no, font, 30, 240, cindex, strlen(cindex));
flushbuffer (wt_no);

putblock (wt_no, 31L, 180, 240, 220, 260, buffer2, bufsize2);
sprintf (cindex, "%d", green[ton+1]);
drawsymbols (wt_no, font, 190, 240, cindex, strlen(cindex));
flushbuffer (wt_no);

putblock (wt_no, 31L, 315, 240, 355, 260, buffer2, bufsize2);
sprintf (cindex, "%d", blue[ton+1]);
drawsymbols (wt_no, font, 325, 240, cindex, strlen(cindex));
showcursor (wt_no);
flushbuffer (wt_no);
}

else
{
if ((y<=160) && (y>=120)) { /* Selecionou o "+" */
flag=1; /* Indica que deve ser incrementada a cor (+) */
if ((x<=65) && (x>=25)) { /* Vermelho */
putblock (wt_no, 31L, 20, 240, 60, 260, buffer2, bufsize2);
altcor (flag, red, ton, cindex);
drawsymbols (wt_no, font, 30, 240, cindex, strlen(cindex));
flushbuffer (wt_no);
}
else
if ((x<=225) && (x>=185)) { /* Verde */
putblock (wt_no, 31L, 180, 240, 220, 260, buffer2, bufsize2);
altcor (flag, green, ton, cindex);
drawsymbols (wt_no, font, 190, 240, cindex, strlen(cindex));
flushbuffer (wt_no);
}
else
if ((x<=360) && (x>=320)) { /* Azul */
putblock (wt_no, 31L, 315, 240, 355, 260, buffer2, bufsize2);
altcor (flag, blue, ton, cindex);
drawsymbols (wt_no, font, 325, 240, cindex, strlen(cindex));
flushbuffer (wt_no);
}
} /* fim do if */
else
{
if ((y<=230) && (y>=190)) { /* Selecionou o "-" */
flag=0; /* Indica que deve ser decrementada a cor (-) */
if ((x<=65) && (x>=25)) { /* Vermelho */
putblock (wt_no, 31L, 20, 240, 60, 260, buffer2, bufsize2);
altcor (flag, red, ton, cindex);
drawsymbols (wt_no, font, 30, 240, cindex, strlen(cindex));
flushbuffer (wt_no);
}
else
if ((x<=225) && (x>=185)) { /* Verde */

```

```

putblock (wt_no, 31L, 180, 240, 220, 260, buffer2, bufsize2);
altcor (flag, green, ton, cindex);
drawsymbols (wt_no, font, 190, 240, cindex, strlen(cindex));
flushbuffer (wt_no);
}
else
if ((x<=360) && (x>=320)) { /* Azul */
putblock (wt_no, 31L, 315, 240, 355, 260, buffer2, bufsize2);
altcor (flag, blue, ton, cindex);
drawsymbols (wt_no, font, 325, 240, cindex, strlen(cindex));
flushbuffer (wt_no);
}
} /* fim do if */
} /* fim do segundo else */
} /* fim do else */
} /* Fim da rotina tabcor */

/*-----
Esta rotina altera a tabela de cores (VLT) que esta sendo acessada pelo
programa, conforme as especificacoes do usuario.
-----*/
altcor (flag, cor, ton, cindex)
char cindex[3];
int flag, cor[16], ton;
{
/* Se flag=1 a tonalidade sera aumentada, caso contrario diminuida */
if ((flag) && (ton<16))
if (cor[ton+1]<15) cor[ton+1]++;

if ((flag == 0) && (ton<16))
if (cor[ton+1]>0) cor[ton+1]--;
Set_vlt (vs_no, red, green, blue); /* Atualiza a tabela de cores */

sprintf (cindex, "%d", cor[ton+1]);
flushbuffer (wt_no);
} /* fim da rotina altcor */

/*-----
Esta rotina verifica em que posicao do histograma foi solicitada a
redistribuicao do mesmo.
-----*/
char trata_hist (wh_no, x, y, cluster_hist)
int wh_no, x, y, cluster_hist;
{
char resposta [4];
int i, j, nivel_ini, nivel_fin, resto_ini, resto_fin;
int delta, resto, inic, ini_dist, fin_dist;

if ((y<27) || (y>267)) { /* Esta fora dos limites do histograma */
Clear_message (1);
Clear_message (2);
Display_message (1, 3, "Ponto solicitado fora do histograma !");
return;
}

x=y-27;
nivel_ini=x/16;
resto_ini=x%16;

if (resto_ini)
nivel_ini++;

Clear_message (1);
Clear_message (2);
Display_message (1, 3,
"Clique o botao central no segundo nivel desejado!");

```

```

Clear_button_data();
Display_message(2,3, "Clique o botao da esquerda do mouse para
abandonar");

flushbuffer (w_no);
Wait_for_events(BUTTON_EVENT, &cur_event_mask);
Get_button_data (&janela,&x,&y,&button,&transition,&time_t);
if ( button != MIDDLE_BUTTON ) {
    Clear_message(1); /* Limpa o campo de mens. mais a esqr.*/
    Clear_message(2); /* Limpa o campo de mens. mais a esqr.*/
    return;
}

if (janela != wh_no) {
    Clear_message(1);
    Display_message(1,3,"Ponto fora da janela do histograma !");
    return;
}

if ((y<27) || (y>288)) { /* Esta fora dos limites do histograma */
    Clear_message(1);
    Clear_message(2);
    Display_message(1,3, "Ponto solicitado fora do histograma !");
    return;
}

x=y-27;
nivel_fin=x/16;
resto_fin=x%16;

if (resto_fin)
    nivel_fin++;

nivel_ini=16-nivel_fin;
nivel_fin=16-nivel_fin;
/* Qual o menor nivel? */
if (nivel_ini > nivel_fin) { /* Troca */
    x=nivel_ini;
    nivel_ini=nivel_fin;
    nivel_fin=x;
}

ini_dist=nivel_ini;
fin_dist=nivel_fin;

if (nivel_fin < 15)
    fin_dist=14;

if (nivel_ini > 0)
    ini_dist=1;

delta= nivel_fin - nivel_ini+1;

x=delta*(*cluster_hist); /* x <- num de pontos a serem redistribuidos */

resto=x % (fin_dist - ini_dist + 1);
x/=fin_dist - ini_dist + 1;

/* Inicializacao do vetor cor_alt, com os 16 tons de cinza */
inic=ini_dist * (*cluster_hist);
for (i=ini_dist; i<=fin_dist; i++){
    for (j=inic; j<inic+x; j++){
        cor_alt[j]=i;
        if (resto) {
            cor_alt[j]=i;
            inic=inic+x+1;
            resto--;
        }
    }
}

```

```

    else {
        inic=inic+x;
    }
}

/* Saturando em baixo */
for (i=0; i<ini_dist * (*cluster_hist); i++)
    cor_alt[i]=0;

/* Saturando em cima */
for (i=127; i >= j; i--)
    cor_alt[i]=15;

*cluster_hist=x;

/* Desenhando a delimitacao do histograma */
fgcolor (wh_no,28);
hidecursor (wh_no);
move (wh_no, 80, 288-((nivel_ini)*16));
draw (wh_no, 100, 288-((nivel_ini)*16));
draw (wh_no, 100, 288-((nivel_fin+1)*16));
draw (wh_no, 80, 288-((nivel_fin+1)*16));
flushbuffer (wh_no);
showcursor (wh_no);
flushbuffer (wh_no);

recebe_resposta ("Pode plotar a imagem ( S(im) ou N(ao) )? ",resposta);

if ((resposta[0] == 's') || (resposta[0] == 'S'))
    return('s');
else return ('n');

} /* fim da rotina trata_hist */

/*-----
Esta rotina realiza a apresentacao do Histograma para uma dada imagem
que ja esta sendo apresentada ao usuario.
-----*/

histograma(hist)
int hist[16];
{
    int i, max,taxa, plot;
    static char message1[]="
        O histograma esta desenhado com uma proporcao de 1 para ";
    char message2[100];

    /* Especificacao de texto para o Histograma */
    static char nropont1[] = {"Numero"};
    static char nropont2[] = {"de Pontos"};
    static char nivint1[] = {"Niveis de"};
    static char nivint2[] = {"Intensidade"};
    static char menos[] = {"-"};

    Erase_win (wh_no);
    fgcolor (wh_no,30);
    /* Desenhar o texto especificando a origem */
    drawsymbols (wh_no, font2,1050,296,nropont1,strlen(nropont1));
    drawsymbols (wh_no, font2,1050,316,nropont2,strlen(nropont2));
    drawsymbols (wh_no, font2,12,2,nivint1,strlen(nivint1));
    drawsymbols (wh_no, font2,5,14,nivint2,strlen(nivint2));
    flushbuffer (wh_no);

    /* Desenhar os eixos do Histograma */
    move (wh_no,20,288);
    draw (wh_no,1150,288); /* Eixo X */
    move (wh_no,20,288);
    draw (wh_no,20,28); /* Eixo Y */
    flushbuffer (wh_no);
}

```

```

for (i=1; i<17; ++i)
    drawsymbols (wh_no,font2,13,283-(i*16),menos, strlen(menos));
flushbuffer (wh_no);

/* Procurando o maior elemento do histograma */
max=0;
for (i=0; i<16; ++i)
    if (max < hist[i]) max=hist[i];

taxa=1;
/* 1130 e o limite para o tamanho do bloco do histograma */
while (max/taxa > 1130)
    ++taxa;
/* Espec. a taxa de conversao do histograma */
Clear_message (1);
Clear_message (2);
sprintf (message2, "%s %d", message1, taxa);
Display_message (1,3,message2);

/* Desenhando os blocos do Histograma */
for (i=0; i<16; ++i) {
    plot=hist[i]/taxa;
    fgcolor (wh_no,i+1); /* Estabelece cor de frente */
    rectf (wh_no,22,288-((i+1)*16),22+plot,286-(i*16));
}

/* Salvar Memory Raster da janela do Histograma */
if (flaghist) /* Ja ha bloco alocado */
    free (buffhist);
bhistsize=get_put_block_size(0,0,1174,333,5);
if ((buffhist=malloc(bhistsize)) ==0) {
    printf ("Nao conseguiu alocar espaco (3)");
    Exit_tools();
    exit (3);
}
hidecursor (wh_no);
getblock (wh_no, 31L, 0, 0, 1174, 333,buffhist,bhistsize);
flaghist=1;
showcursor (wh_no);
flushbuffer (wh_no);
} /* fim da rotina histograma */

/*-----
             Esta rotina desenha a tabela de cores
-----*/
destab()
{
    int i,x;

    /* Especificacao de texto para a Tabela de Cores */
    static char mred[] = {"Vermelho"};
    static char mgreen[] = {"Verde"};
    static char mblue[] = {"Azul"};
    static char mton[] = {"Tonalidade"};
    static char mais[] = {"+"};
    static char menos[] = {"-"};

    Erase_win (wt_no);
    x=23;
    for (i=1; i<17; ++i) {
        fgcolor (wt_no, i);
        rectf (wt_no,x*(i-1),5,(x*(i-1))+23,70);
    }
    flushbuffer (wt_no);

    fgcolor (wt_no,31);
    Load_symbol_file (NONIE, ffont);

    /* Desenhando o texto especificando a origem */
    hidecursor (wt_no);

```

```

drawsymbols (wt_no, font,5,90,mred, strlen(mred));
drawsymbols (wt_no, font,175,90,mgreen, strlen(mgreen));
drawsymbols (wt_no, font,322,90,mblue, strlen(mblue));
drawsymbols (wt_no, font,130,300,mton, strlen(mton));
flushbuffer (wt_no);

/* Desenhando os blocos que permitam alterar as tonalidades */
/* Para o Vermelho */
move (wt_no, 25,120);
draw (wt_no, 65,120);
draw (wt_no, 65,160);
draw (wt_no, 25,160);
draw (wt_no, 25,120);
drawsymbols (wt_no, font,40,130,mais, strlen(mais));

move (wt_no, 25,190);
draw (wt_no, 65,190);
draw (wt_no, 65,230);
draw (wt_no, 25,230);
draw (wt_no, 25,190);
drawsymbols (wt_no, font,40,200,menos, strlen(menos));

/* Para o Verde */
move (wt_no, 185,120);
draw (wt_no, 225,120);
draw (wt_no, 225,160);
draw (wt_no, 185,160);
draw (wt_no, 185,120);
drawsymbols (wt_no, font,200,130,mais, strlen(mais));

move (wt_no, 185,190);
draw (wt_no, 225,190);
draw (wt_no, 225,230);
draw (wt_no, 185,230);
draw (wt_no, 185,190);
drawsymbols (wt_no, font,200,200,menos, strlen(menos));

/* Para o Azul */
move (wt_no, 320,120);
draw (wt_no, 360,120);
draw (wt_no, 360,160);
draw (wt_no, 320,160);
draw (wt_no, 320,120);
drawsymbols (wt_no, font,335,130,mais, strlen(mais));

move (wt_no, 320,190);
draw (wt_no, 360,190);
draw (wt_no, 360,230);
draw (wt_no, 320,230);
draw (wt_no, 320,190);
drawsymbols (wt_no, font,335,200,menos, strlen(menos));

/* Obter o bloco de fundo para a Tonalidade */
bufsize2=get_put_block_size (290,300,330,320,5);
if ((buffer2=malloc(bufsize2)) ==0) {
    printf ("Nao conseguiu alocar espaco (2)");
    exit (2);
}
getblock (wt_no, 31L, 290,300,330,320,buffer2,bufsize2);

showcursor (wt_no);
flushbuffer (wt_no);
} /* fim da rotina destab */

/*-----
             Esta rotina desenha a imagem tomografica
-----*/
des_imagem(ponto,nlin,ncol, xlo,ylo, xhi,yhi,hist,cor)
int ponto[][NCOLUNAS],hist[16],cor[128];
int nlin, ncol, xlo, ylo, xhi, yhi;

```

```

{
int i,j, ii,jj;
int dy;

int num_covers;
int cov_win [MAX_WINDOW];
int vet_xlo [MAX_WINDOW];
int vet_ylo [MAX_WINDOW];
int vet_xhi [MAX_WINDOW];
int vet_yhi [MAX_WINDOW];

/* Zerando o vetor de dados do Histograma */
for (i=0;i<16;++i)
    hist[i]=0;

dy=0;

Clear_message (1);
Clear_message (2);
Display_message (1,3,"Plotando a Imagem ... ");

Inq_covering_windows (w_no, &num_covers, cov_win, vet_xlo, vet_ylo,
    vet_xhi,vet_yhi);

/* printf ("num_cov=%d \n", num_covers); */

if (num_covers)
    Pop_win_to_top (w_no);

hidecursor (w_no);
weight (w_no, larg_pix-1);

for (i=0; i<nlin; ++i) {
    ii=inicx;
    jj=inicy+dy;
    for (j=0; j<ncol; ++j) {
        if ((ponto[i][j] > 127)) {
            ponto[i][j]=127;
        }
        if ((ponto[i][j] < 0)) {
            ponto[i][j]=0;
        }
        ++(hist[cor[ponto[i][j]]]); /* Prepara para a aprest. do hist */
        fgcolor(w_no,cor[ponto[i][j]]+1); /*Estab. a intens. do ponto*/
        pnt (w_no,ii,jj);

/* -- Sem halftoning --
        pnte (w_no, ponto[i][j], ii, jj);
*/

        ii=ii+larg_pix; /* Desl. o ponto de 1, por causa de sua largura */
    }

    dy=dy+larg_pix;
    ii=inicx;
    jj=inicy+dy;

    if (nlin == 64) { /* Escreve duas vezes a linha com os 256 pontos */
        for (j=0; j<ncol; ++j) {
            fgcolor(w_no,cor[ponto[i][j]]+1); /* Seta a intens. do
                ponto incrementada para desloc
                a VLT */

            pnt (w_no,ii,jj);

/* -- Sem halftoning --
            pnte (w_no, ponto[i][j], ii, jj);
*/

            ii=ii+larg_pix; /* Desloca o ponto de 1, */
        } /* por causa de sua largura */
        flushbuffer(w_no);
        dy=dy+larg_pix;
    }
}

```

```

} /* fim do for i<= nlin */

flushbuffer (w_no);
Clear_message (1);
Clear_message (2);
Display_message(1,3,"Fim do processamento da imagem !");

if (fhist) { /* Histograma deve ser atualizado */
    histograma (hist);
    Display_message(2,3,"Histograma atualizado !");
}

/* Salvar Memory Raster da janela de Imagem */
if (flagimg) /* Ja ha bloco alocado */
    free (buff_img);

bimgsize=get_put_block_size(0,0,xhi-xlo,yhi-ylo,5);
if ((buff_img=malloc(bimgsize)) ==0) {
    printf ("Nao conseguiu alocar espaco (1)");
    Exit_tools();
    exit (1);
}

getblock(w_no, 31L, 0, 0, xhi-xlo, yhi-ylo,buff_img,bimgsize);
flagimg=1;
showcursor(w_no);
flushbuffer(w_no);
} /* fim da rotina des_imagem */

/*
-----
Meta rotinas le uma imagem tomografica e a plota.
-----*/
le_des_imagem(ponto,ponto_IA,nlin,ncol, xlo,ylo, xhi,yhi,hist,cor)
int nlin, ncol, xlo, ylo, xhi, yhi;
ponto[NLINHAS][NCOLUNAS],ponto_IA[NLINHAS][NCOLUNAS],hist[16],cor[128];

int i,j, ii,jj;
int dy;

int num_covers;
int cov_win [MAX_WINDOW];
int vet_xlo [MAX_WINDOW];
int vet_ylo [MAX_WINDOW];
int vet_xhi [MAX_WINDOW];
int vet_yhi [MAX_WINDOW];

/* Zerando o vetor de dados do Histograma */
for (i=0;i<16;++i)
    hist[i]=0;

/* Desenha a imagem */
if (ncol <=256)
    inicx=xmin+125;
else inicx=xmin;
inicy=ymin;
dy=0;

Clear_message (1);
Clear_message (2);
Display_message (1,3,"Lendo e Plotando a Imagem ... ");

Inq_covering_windows (w_no, &num_covers, cov_win, vet_xlo, vet_ylo,
    vet_xhi,vet_yhi);

if (num_covers)

```

```

Pop_win_to_top (w_no);

hidecursor (w_no);
weight (w_no, larg_pix-1);

for (i=0; i<nlin; ++i) {
    ii=iniy;
    jj=iniy+dy;
    for (j=0; j<ncol; ++j) {
        fscanf (imagem, "%d", &ponto[i][j]); /* Le o ponto */
        if ((ponto[i][j] > 127))
            ponto[i][j]=127;
        if ((ponto[i][j] < 0))
            ponto[i][j]=0;
        ponto_IA[i][j]=ponto[i][j];
        ++(hist[cor(ponto[i][j])]); /* Prepara para a aprest. do hist */
        fgcolor (w_no, cor(ponto[i][j])+1); /* Estab. a intens. do ponto */
        pnt (w_no, ii, jj);
    }

    /* -- Sem halftoning --
    pnte (w_no, ponto[i][j], ii, jj);
    */

    ii=ii+larg_pix; /* Desl. o ponto de sua largura */
    dy=dy+larg_pix;
    ii=iniy;
    jj=iniy+dy;

    if (nlin == 64) { /* Escreve duas vezes a linha com os 256 pontos */
        for (j=0; j<ncol; ++j) {
            fgcolor (w_no, cor(ponto[i][j])+1); /* Seta a intens. do
            ponto incrementada para
            desloc a VLT */
            pnt (w_no, ii, jj);
        }
    }

    /* -- Sem halftoning --
    pnte (w_no, ponto[i][j], ii, jj);
    */

    ii=ii+larg_pix; /* Desloca o ponto de 1, */
    /* por causa da sua largura */
    flushbuffer (w_no);
    dy=dy+larg_pix;
    } /* fim do for i<= nlin */

flag_IA=1;
flushbuffer (w_no);
Clear_message (1);
Clear_message (2);
Display_message (1,3, "Fim do processamento da imagem !");

if (fhist) { /* Histograma deve ser atualizado */
    histograma (hist);
    Display_message (2,3, "Histograma atualizado !");
}

/* Salvar Memory Raster da janela de Imagem */
if (flagimg) /* Ja ha bloco alocado */
    free (buff_img);

bimgsize=get_put_block_size (0,0,xhi-xlo,yhi-ylo,5);
if ((buff_img=malloc (bimgsize)) ==0) {
    printf ("Nao conseguiu alocar espaco (1)");
    Exit_tools ();
    exit (1);
}

getblock (w_no, 31L, 0, 0, xhi-xlo, yhi-ylo, buff_img, bimgsize);
flagimg=1;

```

```

showcursor (w_no);
flushbuffer (w_no);
} /* fim da rotina le_des_imagem */

/*-----
Esta rotina realiza um filtro passa-baixas sobre uma matriz (256x256)
O filtro e definido pela matriz [0.1111 0.1111 0.1111]
                                [0.1111 0.1111 0.1111]
                                [0.1111 0.1111 0.1111]
-----*/

passa_baixas (m,n, imagem, imagem_filt)
int m,n;
int imagem[][NCOLUNAS], imagem_filt[][NCOLUNAS];
{
    int dimasc;
    static float masc_baixas_3 [3][3] = {
        {0.1111, 0.1111, 0.1111},
        {0.1111, 0.1111, 0.1111},
        {0.1111, 0.1111, 0.1111}
    };

    static float masc_baixas [3][3] = {
        {0.0625, 0.125, 0.0625},
        {0.125, 0.25, 0.125},
        {0.0625, 0.125, 0.0625}
    };

    int i,j,k,l,soma;
    char resposta[4];

    Display_message (1,3, "Calculando ...");
    convolucao (imagem, m,n, masc_baixas, 3, imagem_filt);
    return ;
} /* fim da rotina passa_baixas */

/*-----
Esta rotina realiza um filtro passa-altas sobre uma matriz (256x256)
O filtro e definido pela matriz [ 0 -1 0 ]
                                [-1 5 -1 ]
                                [ 0 -1 0 ]
-----*/

passa_altas (m,n, imagem, imagem_filt)
int m,n;
int imagem[][NCOLUNAS], imagem_filt[][NCOLUNAS];
{
    static int dimasc=3;
    static float masc_altas [3][3] = {
        { 0.0, -1.0, 0.0},
        {-1.0, 5.0, -1.0},
        { 0.0, -1.0, 0.0}
    };

    Display_message (1,3, "Calculando ...");
    convolucao (imagem, m,n, masc_altas, dimasc, imagem_filt);
} /* fim da rotina passa_altas */

/*-----
Esta rotina efetua a convolucao de uma funcao (filtro) representada
pela matriz "masc_filt" que possui dimensoes "dimasc"x"dimasc",
com a imagem representada pela matriz "imagem" que possui dimensoes "m"x"n".
-----*/

convolucao (imagem, m,n, masc_filt, dimasc, imagem_filt)
int imagem[][NCOLUNAS], imagem_filt[][NCOLUNAS];
int m,n,dimasc;
float masc_filt[][DIM_MASC];
{
    int i,j,k,l;
    float soma;

```

```

char resposta[4];
int mat_aux[NLINHAS][NCOLUNAS];

/* Convolucao da imagem com a mascara */
for (i=1; i<m-1; i++)
  for (j=1; j<n-1; j++) {
    soma=0.0;
    for (k=0; k<dimasc; k++)
      for (l=0; l<dimasc; l++)
        soma=soma+(imagem[i-k][j-l] * masc_filt[k][l]);
    mat_aux[i][j]=soma;
  }

for (i=1; i<m-1; i++)
  for (j=1; j<n-1; j++)
    imagem_filt[i][j]=mat_aux[i][j];

/* Ignora as linhas e colunas perifericas */
for (j=0; j<n; j++) {
  imagem_filt[0][j]=imagem[0][j];
  imagem_filt[m-1][j]=imagem[m-1][j];
}
for (i=0; i<m; i++) {
  imagem_filt[i][0]=imagem[i][0];
  imagem_filt[i][n-1]=imagem[i][n-1];
}
test_normaliza (imagem_filt, imagem_filt, m,n);
} /* fim da rotina convolucao */

/*-----
Nota rotina efetua a filtragem nao linear atraves do operador
gradiente de Roberts (realca de bordas)
-----*/
roberts(mat_ent, mat_saida, m,n)
int mat_ent[][NCOLUNAS], mat_saida[][NCOLUNAS];
int m,n;
{
  int i,j;
  int mat_aux[NLINHAS][NCOLUNAS];
  char resposta[4], flag;

  /* Nao executa o cruzamento nas linhas e colunas perifericas da imagem */
  for (i=0; i<m-1; i++)
    for (j=0; j<n-1; j++)
      mat_aux[i][j]=ABS(mat_ent[i][j]- mat_ent[i+1][j+1]) +
        ABS(mat_ent[i][j+1]- mat_ent[i+1][j]);

  for (j=0; j<n; j++) /* Despreza a ultima linha da matriz copiando-a */
    mat_aux[m-1][j]=mat_ent[m-1][j];

  for (i=0; i<m; i++) /* Despreza a ultima coluna da matriz copiando-a */
    mat_aux[i][n-1]=mat_ent[i][n-1];

  test_normaliza (mat_aux, mat_saida, m,n);
} /* fim da rotina roberts */

/*-----
Nota rotina normaliza a matriz "ponto" para a faixa 0 a 127,
A matriz normalizada e "ponto_norm".
-----*/
normaliza (ponto, ponto_norm,m,n)
int ponto[][NCOLUNAS], ponto_norm[][NCOLUNAS];
int m,n;
{
  int i,j;

```

```

int max, min;
float a, b;

/*----- Normalizar a Imagem na faixa de 0 a 127 */
min=0x7fffffff; /* maior elemento armazenavel */
max= -min ;

for (i=0; i<m; i++)
  for (j=0; j<n; j++) {
    if (min > ponto[i][j]) min=ponto[i][j];
    if (max < ponto[i][j]) max=ponto[i][j];
  }

a=127./(max-min);
b=127. *((float)min/(max-min));

for (i=0; i<m; i++)
  for (j=0; j<n; j++)
    ponto_norm[i][j]=a*ponto[i][j] - b + 0.5;
} /* fim da rotina normaliza */

```



```

/*-----
Nota rotina verifica conforme "flag" se a "mat_ent" deve ser normalizada.
Se "mat_ent" for normalizada, a matriz normalizada retornara em
"mat_sai".
A dimensao das matris e de "mm".
-----*/
test_normaliza (mat_ent, mat_sai, m,n)
int mat_ent[][NCOLUNAS], mat_sai[][NCOLUNAS];
int m,n;
{
    int i,j;
    char resposta[4];

    /* Quer re-normalizar a Imagem Obtida? */

    recebe_resposta ("Deve-se re-normalizar a Imagem ( S(im) ou N(ao) )? ",
        resposta);

    if ((resposta[0] == 's') || (resposta[0] == 'S')) {
        normaliza (mat_ent, mat_sai,m,n);
        return ;
    }

    for (i=0; i<m; i++)
        for (j=0; j<n; j++)
            mat_sai[i][j]=mat_ent[i][j];
} /* fim da rotina test_normaliza */

/*-----
Nota rotina implementa o filtro da mediana, isto e, para cada ponto da
imagem e selecionada uma vizinhanca de ordem 2, e os pontos sao ordena-
dos sendo substituido o ponto pelo ponto medio da vizinhanca.
-----*/
mediana (m,n,mat_ent,mat_sai)
int m,n, mat_ent[][NCOLUNAS], mat_sai[][NCOLUNAS];
{
    int i,j,k,l,ii, vet[9], mat_aux[NLINHAS][NCOLUNAS];
    int dim_med=9;
    int laux, jaux, x;

    Clear_message (1);
    Clear_message (2);
    Display_message (1,3,"Calculando ...");

    for (i=1; i<m-1; i++)
        for (j=1; j<n-1; j++) {
            /* Copia a vizinhanca do ponto (i,j) em vet */
            ii=0;
            for (k=1-1; k<=i+1; k++)
                for (l=1-1; l<=j+1; l++) {
                    vet[ii]=mat_ent[k][l];
                    ii++;
                }

            /* Ordenar o vetor -vet- */
            for (iaux=0; iaux<dim_med; iaux++) {
                k=iaux;
                x=vet[iaux];
                for (jaux=iaux+1; jaux<=dim_med; jaux++) {
                    if (vet[jaux] < x) {
                        k=jaux;
                        x=vet[k];
                    }
                }
                vet[k]=vet[iaux];
                vet[iaux]=x;
            }
        }
}

```

```

}

mat_aux[i][j]=vet[(dim_med)/2];
} /* fim do for j */

/* Ignora as linhas e colunas perifericas */
for (j=0; j<n; j++) {
    mat_aux[0][j]=mat_ent[0][j];
    mat_aux[m-1][j]=mat_ent[m-1][j];
}
for (i=0; i<m; i++) {
    mat_aux[i][0]=mat_ent[i][0];
    mat_aux[i][n-1]=mat_ent[i][n-1];
}

test_normaliza (mat_aux, mat_sai, m,n);

} /* Fim da rotina mediana */

/*-----
Nota rotina inicializa o vetor cor com a primeira distribuicao uniforme.
-----*/
inico (cor)
int cor[128];
{
    int i,j,inic;

    /* Inicializacao do vetor cor, com os 16 tons de cinza */
    inic=0;
    for (i=0; i<16; ++i) {
        for (j=inic; j<inic+8; ++j) cor[j]=i;
        inic=inic+8;
    }
} /* Fim da rotina inicor */

/*-----
Nota rotina efetua uma rotacao vertical da imagem.
A linha (horizontal) delimitadora da rotacao e especificada atraves do
mouse.
-----*/
rot_imagem (w_no,nlin,ncol,ponto,xlo,ylo,xhi,yhi)
int nlin,ncol,ponto[][NCOLUNAS];
{
    int i,j,posic,k,l;
    int auxponto[NLINHAS][NCOLUNAS];
    static int x,y;

    rubberlin (w_no,&x,&y);

    if ((posic=y/larg_pix) > nlin)
        return;

    k=0;
    for (i=posic; i<nlin; i++) {
        for (j=0; j<ncol; j++)
            auxponto[k][j]=ponto[i][j];
        k++;
    }
    for (i=0; i<posic; i++) {
        for (j=0; j<ncol; j++)
            auxponto[k][j]=ponto[i][j];
        k++;
    }

    for (i=0; i<nlin; i++)
        for (j=0; j<ncol; j++)
            ponto[i][j]=auxponto[i][j];
    des_imagem(ponto,nlin, ncol, xlo,ylo, xhi,yhi,hist,cor);
} /* Fim da rotina rot_imagem */

```

```

/*-----
Nota rotina efetua o rubberband de 1 linha na janela w_no.
-----*/
rubberlin (w_no,x,y)
int w_no,*x,*y;
{
    int flag,time_no;
    char *buffer;
    unsigned bufsize;

    bufsize=get_put_block_size (200,10,450,10,5);
    if ((buffer=malloc(bufsize)) == 0){
        printf ("Nao conseguiu alocar espaco (1)\n");
        return;
    }

    weight (w_no,0);
    fgcolor (w_no, 28); /* Rubberband sera desenhado em vermelho */
    Start_timer (20,1,&time_no);
    hidecursor (w_no);
    linestyle (w_no, 0xf9f9);

    do
    {
        flushbuffer (w_no);
        Mouseposition (w_no, x,y,&flag);
        if (flag==0) continue;
        getblock (w_no, 31L,200,*y,450,*y,buffer,bufsize);
        move(w_no,200,*y); /* Ponto inicial da linha */
        draw (w_no,450,*y);
        flushbuffer (w_no);
        Wait_for_events(TIMER_EVENT, &cur_event_mask);
        Clear_timer_data();
        putblock (w_no,31L,200,*y,450,*y,buffer,bufsize);
        flushbuffer (w_no);
        Wait_for_events(TIMER_EVENT, &cur_event_mask);
        Clear_timer_data();
        Inq_events (&cur_event_mask);
    }
    while (! (BUTTON_EVENT & cur_event_mask));
    showcursor(w_no);

    flushbuffer (w_no);
    Clear_button_data();

    free(buffer);
    linestyle (w_no, 0xffff);
    /* Desenha a linha final */
    hidecursor (w_no);
    move(w_no,200,*y); /* Ponto inicial da linha */
    draw(w_no,450,*y);

    showcursor(w_no);
    flushbuffer (w_no);
} /* fim da rotina rubberlin */

/*-----
Nota rotina efetua o rubberband de 4 linhas na janela w_no,
Iniciando no ponto (x1,y1).
-----*/
rubberquad (w_no,x1,y1,x2,y2)
int w_no,x1,y1,*x2,*y2;
{
    int flag,time_no,xx,yy, sx,sy, lado, ladox, ladoy,
    xlo1, xhi1, ylo2, yhi2,
    xlo3, xhi3, ylo4, yhi4;

```

```

char *buffer1, *buffer2,*buffer3,*buffer4;
int bufsize1,bufsize2, bufsize3, bufsize4;

bufsize1=get_put_block_size (0,1,1184,1,5);
bufsize2=get_put_block_size (1,0,1,884,5);
bufsize3=bufsize1;
bufsize4=bufsize2;
if ((buffer1=malloc(bufsize1)) == 0){
    printf ("Nao conseguiu alocar espaco (1)\n");
    return;
}
if ((buffer2=malloc(bufsize2)) == 0){
    printf ("Nao conseguiu alocar espaco (1)\n");
    return;
}
if ((buffer3=malloc(bufsize3)) == 0){
    printf ("Nao conseguiu alocar espaco (1)\n");
    return;
}
if ((buffer4=malloc(bufsize4)) == 0){
    printf ("Nao conseguiu alocar espaco (1)\n");
    return;
}

weight (w_no,0);
fgcolor (w_no, 28); /* Rubberband sera desenhado em vermelho */
Start_timer (20,1,&time_no);
hidecursor (w_no);
linestyle (w_no, 0xf9f9);
move (w_no, x1,y1);

do
{
    flushbuffer (w_no);
    Mouseposition (w_no,x2,y2,&flag);
    if (flag == 0) continue; /* Mouse fora da janela */
    ladox= *x2 - x1;
    ladoy= y1 - *y2;
    lado=MAX (ABS(ladox),ABS(ladoy));

    sx=((x2 > x1) ? 1: -1;
    sy=((y2 > y1) ? 1: -1;
    xx=sx*lado;
    yy=sy*lado;

    /* Prepara para desenhar a Primeira linha */
    if (xx > 0) { xlo1=x1; xhi1=x1+xx; }
    else { xlo1=x1+xx; xhi1=x1; }
    bufsize1=get_put_block_size(xlo1-2,y1,xhi1+2,y1,5);
    getblock (w_no,31L,xlo1-2,y1,xhi1+2,y1,buffer1, bufsize1);

    /* Prepara para desenhar a Segunda linha */
    if (yy > 0) { ylo2=y1; yhi2=y1+yy; }
    else { ylo2=y1+yy; yhi2=y1; }
    bufsize2=get_put_block_size(x1+xx,ylo2-2,x1+xx,yhi2+2,5);
    getblock (w_no,31L,x1+xx,ylo2-2,x1+xx,yhi2+2,buffer2, bufsize2);

    /* Prepara para desenhar a Terceira Linha */
    if (xx > 0) { xlo3=x1; xhi3=x1+xx; }
    else { xlo3=x1+xx; xhi3=x1; }
    bufsize3=get_put_block_size(xlo3-2,y1+yy,xhi3+2,y1+yy,5);
    getblock (w_no,31L,xlo3-2,y1+yy,xhi3+2,y1+yy,buffer3, bufsize3);

    /* Prepara para desenhar a Quarta Linha */
    if (yy > 0) { ylo4=y1; yhi4=y1+yy; }
    else { ylo4=y1+yy; yhi4=y1; }
    bufsize4=get_put_block_size(x1,ylo4-1,x1,yhi4+2,5);
    getblock (w_no,31L,x1,ylo4-2,x1,yhi4+2,buffer4, bufsize4);

```

```

flushbuffer (w_no);

/* Desenha as quatro linhas */
if (xx < 0) draw (w_no, xl+xx+1,y1);
else draw(w_no,xl+xx-1,y1);
move (w_no,xl+xx,y1);

if (yy < 0) draw(w_no,xl+xx, y1+yy+1);
else draw(w_no,xl+xx, y1+yy-1);
move (w_no,xl+xx,y1+yy);

if (xx < 0) draw(w_no,xl-1, y1+yy);
else draw(w_no,xl+1, y1+yy);
move (w_no,xl,y1+yy);

if (yy < 0) draw(w_no,xl, y1-1);
else draw(w_no,xl, y1+1);
move (w_no,xl,y1);
flushbuffer (w_no);

Wait_for_events(TIMER_EVENT, &cur_event_mask);
Clear_timer_data();

putblock(w_no,31L,xlo-2,y1,xhi+2,y1,buffer1, bufsize1);
putblock(w_no,31L,xl+xx,ylo-2,xl+xx,yhi+2, buffer2,bufsize2);
putblock (w_no,31L,xlo-2,y1+yy,xhi+2,y1+yy, buffer3,bufsize3);
putblock (w_no,31L,xl,ylo-2,xl,yhi+2,buffer4, bufsize4);
flushbuffer (w_no);
Wait_for_events(TIMER_EVENT, &cur_event_mask);
Clear_timer_data();

Inq_events (&cur_event_mask);
}
while (!(BUTTON_EVENT & cur_event_mask));
showcursor(w_no);

flushbuffer (w_no);
Clear_button_data();

free(buffer1);
free(buffer2);
free(buffer3);
free(buffer4);
linestyle (w_no, 0xffff);

/* Desenha o quadrado final */
hidecursor (w_no);
move(w_no,xl,y1); /* Ponto inicial do Rubberband */
draw(w_no,xl+xx, y1);
draw(w_no,xl+xx, y1+yy);
draw(w_no,xl, y1+yy);
draw(w_no,xl, y1);

*x2=xl+xx;
*y2=y1+yy;
showcursor(w_no);
flushbuffer (w_no);

} /* fim da rotina rubberquad */

/*-----
Esta funcao retorna a media aritmetica de uma regio de uma matriz dada
Ponto [alin][ncol] elementos. A partir da especificacao das posicoes
desta matriz (xlo,ylo) e (xhi,yhi).
-----*/
float Media (ponto, alin, ncol, xlo,ylo, xhi,yhi)

int nlin, ncol, xlo,ylo, xhi,yhi, ponto[][NCOLUNAS];
int i,j,aux, pos_xlo, pos_xhi, pos_ylo, pos_yhi;

```

```

float soma;

if ((xhi == xlo) || (yhi == ylo)) return(-1.);

if ((pos_xlo=(xlo-inicx)/larg_pix) > nlin) return (-1.);
if ((pos_xhi=(xhi-inicx)/larg_pix) > nlin) return (-1.);
if ((pos_ylo=(ylo-inicy)/larg_pix) > ncol) return (-1.);
if ((pos_yhi=(yhi-inicy)/larg_pix) > ncol) return (-1.);
soma=0.0;
for (i=pos_xlo; i<=pos_xhi; i++)
for (j=pos_ylo; j<=pos_yhi; j++)
soma=soma+ponto[i][j];
aux=(pos_xhi - pos_xlo + 1)*(pos_yhi - pos_ylo + 1);
return (soma/aux);

} /* fim da funcao Media */

/*-----
Esta funcao retorna a media aritmetica de uma regio de uma matriz dada
Ponto [nlin][ncol] elementos. A partir da especificacao das posicoes
desta matriz (xlo,ylo) e (xhi,yhi).
-----*/
float Media2 (ponto, alin, ncol, xlo,ylo, xhi,yhi)
int nlin, ncol, xlo,ylo, xhi,yhi, ponto[][NCOLUNAS];
{
int i,j,aux;
float soma;

if ((xhi == xlo) || (yhi == ylo)) return(-1.);

soma=0.0;
if (xhi > ncol) xhi=ncol;
if (yhi > nlin) yhi=nlin;
for (i=xlo; i<=xhi; i++)
for (j=ylo; j<=yhi; j++)
soma=soma+ponto[i][j];

aux=(xhi - xlo + 1)*(yhi - ylo + 1);
return (soma/aux);

} /* fim da funcao Media2 */

/*-----
Esta rotina escreve um valor real 'num_fl' em uma string 'mesg1'
gerando 'mesg2'
-----*/
Recebe_Float (num_fl, mesg1)
char mesg1[];
float num_fl;
{
char mesg2[100];

if (num_fl == -1.) {
Clear_message (1);
Clear_message (2);
Display_message (1,3,"Valor da Media Invalido !");
}
else
{
sprintf (mesg2,"%s %f", mesg1,num_fl);
Clear_message (1);
}
}

```

```

        Clear_message (2);
        Display_message (1,3, msg2);
    }
} /* fim da rotina Escreve_Float */

/*-----
Esta funcao retorna o desvio padrao da uma regio da matriz dada
Ponto[nlin][ncol] de elementos.
-----*/
float Des_padrao (ponto, nlin, ncol, media, xlo,ylo, xhi,yhi)
{
    int nlin, ncol, xlo,ylo, xhi,yhi, ponto[][NCOLUNAS];
    float media;

    {
        int i,j, pos_xlo, pos_xhi, pos_ylo, pos_yhi;
        float soma, desvio;

        if ((xhi == xlo) || (yhi == ylo)) return(0);

        if ((pos_xlo=xlo/larg_pix) > nlin) return (0);
        if ((pos_xhi=xhi/larg_pix) > nlin) return (0);
        if ((pos_ylo=ylo/larg_pix) > ncol) return (0);
        if ((pos_yhi=yhi/larg_pix) > ncol) return (0);
        soma=0.0;
        for (i=pos_xlo; i<=pos_xhi; i++)
            for (j=pos_ylo; j<=pos_yhi; j++)
                soma=soma+((ponto[i][j] - media)*(ponto[i][j] - media));

        desvio=sqrt(soma/((pos_xhi - pos_xlo)*(pos_yhi - pos_ylo)));
        return (desvio);
    }
} /* fim da funcao Des_padrao */

/*-----
Esta rotina efetua o nivelamento das bordas da Imagem que esta em
ponto IA. As bordas sao definidas como o primeiro quarto da imagem,
e o ultimo quarto da imagem. O nivelamento e feito atraves da funcao
sia truncada na regio inicial PI/8.
-----*/
nivelamento (inic, fim, nlin, ncol, num, ponto)
{
    int inic, fim, nlin, ncol, num, ponto[][NCOLUNAS];
    {
        int i,j;
        float aux1,aux2;

        aux1=3*PI/2;
        aux2=(PI/8)/ncol;
        for (i=0; i<ncol/4; i++)
            ajeita[i]=ajeita[ncol-1-i]=1+num*(1. + sin(aux1 + (aux2 *
(4*i-ncol))));
        for (i=0; i<nlin; i++)
            for (j=0; j<fim; j++)
                ponto[i][j]=(float)ponto[i][j]*ajeita[j];
    }
} /* Fim da rotina nivelamento */

/*-----
Esta rotina recebe uma mensagem em msg1, e retorna a resposta.
-----*/
recebe_resposta (msg1, resposta)
char *msg1, *resposta;
{
    Clear_message (1);
    Clear_message (2);
    keybd_bell(1);
    Process_keyin(1,3,msg1);
    Clear_keyin_data();
    Clear_menubar_data();

    Wait_for_events(KEYIN_EVENT,&cur_event_mask);

```

```

if ( KEYIN_EVENT & cur_event_mask ) {
    Get_keyin_data(&kn_count);
    sscanf( kn_buffer, "%s", resposta);
}

} /* Fim da rotina recebe_resposta */

/*-----
PROGRAMA PRINCIPAL
-----*/
main()
{
    int i, j, inic, aux x, x1,y1, x2,y2;
    int ponto IA[NLINHAS][NCOLUNAS], ponto IB[NLINHAS][NCOLUNAS],
        salva,flag, ponto IC[NLINHAS][NCOLUNAS], ponto[NLINHAS][NCOLUNAS],
        mat_aux[NLINHAS][NCOLUNAS];
    static int nlin, ncol, xlo, ylo, xhi,yhi;
    int cluster_hist;
    static char pont, file[50], resposta[6],
        msg1_med[100],
        msg2_med[100],
        msg1_pad[100],
        msg2_pad[100];
    float aux, aux_med, aux_padrao;
    float Media();
    double sin();
    FILE *saida;

    int num;
    int pd_com, menu_id, menu_id0, menu_id1,
        menu_id2, menu_id3, menu_id4, pd_com4;

    static char *entry[]={"Restaura","Operador de Roberts", "Passa-Baixas",
        "Passa-Altas", "Filtro da
        Mediana","Salva","Operar",
        "Nivelamento","Menubar"};
    static int com[]={1,4,5,6,7,8,11,15,18};
    static int link[]={0,0,0,0,0,0,0,0,0};

    static char *entry0[]={"Total","Esquerdo", "Direito"};
    static int com0[]={15,16,17};
    static int link0[]={0,0,0};

    static char *entry1[]={"Original","Img-B", "Img-C"};
    static int com1[]={1,2,3};
    static int link1[]={0,0,0};

    static char *entry2[]={"Img_B <- Img_A", "Img_C <- Img_A", "Arquivo"};
    static int com2[]={8,9,10};
    static int link2[]={0,0,0};

    static char *entry3[]={"Troca Img_A com Img_C", "Troca Img_A com Img_B",
        "Img_A + Img_B", "Img_A - Img_B"};
    static int com3[]={11,12,13,14};
    static int link3[]={0,0,0,0};

    /* Menu para as Medidas Estatisticas */

    static char *entry4[]={"Media","Desvio_padrao","Rel
    Sinal/Ruido","Menubar"};
    static int com4[]={1,2,3,4};
    static int link4[]={0,0,0,0};

    /* Especificacao das cores do menu */
    static int mred[]={13,0,0,6},
        mgreen[]={13,0,0,0},
        mblue[]={11,0,8,1};

```

```

/* Inicializacao das "tools" */
Enter_tools();
Inq_displayed_vs(&vs_no); /* vs_no <=n da tela virtual */

cluster_hist=8;
flag=0;
fhist=0; /* Histograma nao esta sendo mostrado */
flagtc=0; /* Estabelece que nao ha bloco da tabela de cores guardado */
flagimg=0; /* Estabelece que nao ha bloco(memory raster) da imagem guardado */
flaghist=0; /* Estab. que nao ha bloco(memory raster) do histograma guard. */
flag_IA=0; /* Estab. que nao foi armazenada nenhuma imagem em Imagem A */
flag_hist_alt=0; /* Estabelece que o histograma nao esta alterado */

Create_win (vs_no,"Imagem", 5, 60, 781, 573, &w_no);
Create_win (vs_no,"Tabela de Cores",795,60,1178,450,&wt_no);
Create_win (vs_no,"Histograma",5,487,1179,820,&wh_no);

Create_quick_pd (mred,mgreen,mblue,TEXT_FONT,3,0,entry1,
com1,link1, &menu_id1);
link[0]=menu_id1;

Create_quick_pd(mred,mgreen,mblue,TEXT_FONT,3,0,entry2,
com2,link2,&menu_id2);
link[5]=menu_id2;

Create_quick_pd(mred,mgreen,mblue,TEXT_FONT,4,0,entry3,com3,link3,
&menu_id3);
link[6]=menu_id3;

Create_quick_pd (mred,mgreen,mblue,TEXT_FONT,
3,0,entry0,com0,link0,&menu_id0);
link[7]=menu_id0;
Create_quick_pd (mred,mgreen,mblue,TEXT_FONT,
9,0,entry, com,link, &menu_id);
if (Create_quick_pd (mred,mgreen,mblue,TEXT_FONT,4,0,entry4, com4,
link4, &menu_id4))

Set_logo("Imagem");
Set_vit (vs_no, red, green, blue); /* Le a tabela de cinzas */
Set_message_mode (3); /* O Menubar e dividido em suas linhas */
Set_keyin_buffer( kn_buffer, kn_count);

Display_win(w_no);
flushbuffer(w_no);

Clear_message(3);
Display_message(3,1,"Instituto de Fisica e Quimica de Sao Carlos - USP");

Enable_events (BUTTON_EVENT | KEYIN_EVENT | MENUBAR_EVENT |
TIMER_EVENT | DELETE_EVENT | REFRESH_EVENT);

/* Estabelece os comandos ao menubar */
Attach_menubar ("Imagem", 0);
Attach_menubar ("Histograma", 1);
Attach_menubar ("Tabela de Cores", 2);
Attach_menubar ("Operacoes",3);
Attach_menubar ("Estatisticas",4);
Attach_menubar ("Fim", 5);
Update_menubar();

inico(cor);

Display_message (1,3, "Selecione uma opcao do Menubar, teclando o
botao do meio do mouse.");

```

```

mb_cmd=0;

menubar: do
{
/* Aguarda selecao de comando no menubar */
Clear_menubar_data();
Clear_button_data();
Clear_refresh_data();
Wait_for_events ( MENUBAR_EVENT | BUTTON_EVENT | REFRESH_EVENT,
&cur_event_mask );

if (MENUBAR_EVENT & cur_event_mask) { /* Foi selecao no menubar */
Get_menubar_data (&mb_cmd, &xm, &ym);

flag=0;
switch (mb_cmd) {
case 0: /* Imagem */
/* abrir o arquivo de dados (pontos da imagem) */
Clear_keyin_data();
while ( !flag) {
recebe_resposta ("Entre com o nome do arquivo: ",
file);
if (!strcmp(file, "quit") || !strcmp(file,"exit")) {
Clear_message(1);
goto menubar;
}
if ( !( imagem = fopen( file, "r" ) ) ) {
Clear_message(1);
Display_message (1,3, "Arquivo nao encontrado !");
flushbuffer(w_no);
}
else flag=1;
} /* Fim do while */

fscanf(imagem,"%d,%d",&nlin,&ncol);

if ((ncol==512) && (nlin==512))
larg_pix=1;
else
larg_pix=2;

/* Inicializando o vetor "ajeita" para levantar as
bordas */
for (i=0;i<ncol; i++)
ajeita[i]=1.;

Erase_win (w_no);
aleat=0;

xlo=5;
ylo=60;
yhi=573;
if ((ncol > 256) && (nlin == 256))
xhi=1030;
else
xhi=785;

Modify_win (w_no,xlo,ylo,xhi,yhi);

Erase_win (w_no);
Display_win(w_no);
flushbuffer (w_no);

Clear_message(1);
Display_message(1,3,"Leitura dos Dados ... ");

le_des_imagem(ponto,ponto_IA,nlin, ncol, xlo,ylo,
xhi,yhi, hist,cor);

```

```

        break;

case 1: /* Histograma */
    fhist=1; /* Foi solicitado apresentacao do
             histograma */
    Display_win(wt_no);
    flushbuffer(wt_no);
    Erase_win (wt_no);
    Clear_message(1);

    Load_symbol_file (FIX, &font2);

    if (flag_hist_alt) {
        printf ("histograma alternativo\n");
        histograma (hist_alt);
    }
    else {
        printf ("histograma normal\n");
        histograma (hist);
    }

    break;

case 2: /* Tabela de cores */
    Display_win(wt_no);
    Erase_win (wt_no);
    Set_vlt (vs_n0, inaltred, inaltgreen, inaltblue);
    for (i=0; i<32; i++) { /* Reinicializa a tabela de
                           cores */
        red[i]=inaltred[i];
        green[i]=inaltgreen[i];
        blue[i]=inaltblue[i];
    }
    flushbuffer(wt_no);
    Clear_message(1);
    Clear_message(2);
    destab();
    break;

case 3: /* Operacoes e Filtros */
    Clear_message(1);
    Clear_message(2);
    Display_message (1,3,
        "Pode-se manipular 3 imagens. IA, IB e IC.");
    Display_message (2,3,
        "A imagem IA e utilizada como imagem Principal !");
    pd_com=0; /* Inicializacao */
    while (pd_com != 18) {
        Process_pd (menu_id,0,800,20,&pd_com);

        switch (pd_com) {
            case 1: /* Restaura Imagens - ORIGINAL- */
                for (i=0; i<nlin; ++i)
                    for (j=0; j<ncol; ++j)
                        ponto_IA[i][j]=ponto[i][j];
                flag_IA=1; /* Estab. que a Imag A esta
                           carregada */
                des_imagem(ponto_IA,nlin, ncol, xlo,ylo,
                           xhi,yhi,hist,cor);
                flag_hist_alt=0; /* Volta ao inicial */
                for (i=0; i<128; ++i)
                    cor_alt[i]=cor[i];
                cluster_hist=8;
                break;

            case 2: /* Restaura Imagens - Imagem B - */
                for (i=0; i<nlin; ++i)
                    for (j=0; j<ncol; ++j)
                        ponto_IA[i][j]=ponto_IB[i][j];

                recebe_resposta ("Deve-se plotar a imagem

```

```

        restaurada ( S(im) ou N(ao) )? ",
        resposta);

        if ((resposta[0] == 's') ||
            (resposta[0] == 'S')) {
            flag_IA=1;
            des_imagem(ponto_IA,nlin, ncol, xlo,ylo,
                       xhi,yhi,hist,cor);
        }
        break;

case 3: /* Restaura Imagens - Imagem C - */
    for (i=0; i<nlin; ++i)
        for (j=0; j<ncol; ++j)
            ponto_IA[i][j]=ponto_IC[i][j];

    recebe_resposta ("Deve-se plotar a imagem
        restaurada ( S(im) ou N(ao) )? ",
        resposta);

    if ((resposta[0] == 's') ||
        (resposta[0] == 'S')) {
        flag_IA=1;
        des_imagem(ponto_IA,nlin, ncol, xlo,ylo,
                   xhi,yhi,hist,cor);
    }
    break;

case 4: /* Operador de Roberts */
    if (flag_IA) {
        roberts(ponto_IA, ponto_IA,nlin,ncol);
        des_imagem (ponto_IA,nlin,ncol,
                    xlo,ylo,xhi,yhi,hist,cor);
    }
    else {
        Clear_message (1);
        Clear_message (2);
        Display_message (1,3,"Nao existe nenhuma
            imagem ja restaurada em
            IA!");
        Display_message (2,3,"Deve-se
            primeiramente utilizar
            a opcao RESTAURA
            ORIGINAL");
    }
    break;

case 5: /* Filtro Passa Baixas */
    if (flag_IA) {
        passa_baixas(nlin,ncol,ponto_IA,ponto_IA);
        des_imagem (ponto_IA,nlin,ncol,
                    xlo,ylo,xhi,yhi,
                    hist,cor);
    }
    else {
        Clear_message (1);
        Clear_message (2);
        Display_message (1,3,"Nao existe nenhuma
            imagem ja restaurada em
            IA!");
        Display_message (2,3,"Deve-se
            primeiramente utilizar
            a opcao RESTAURA
            ORIGINAL");
    }
    break;

case 6: /* Filtro Passa altas */
    if (flag_IA) {
        passa_altas(nlin,ncol,ponto_IA,ponto_IA);

```

```

        des_imagem (ponto_IA,nlin,ncol,
                    xlo,ylo,xhi,yhi,
                    hist,cor);
    }
    else {
        Clear_message (1);
        Clear_message (2);
        Display_message (1,3,"Nao existe nenhuma
            imagem ja restaurada em
            IA!");
        Display_message (2,3,"Deve-se
            primeiramente utilizar
            a opcao RESTAURA
            ORIGINAL");
    }
    break;

case 7: /* Filtro da Mediana */
    if (flag_IA) {
        mediana (nlin,ncol,ponto_IA,ponto_IA);
        des_imagem (ponto_IA,nlin,ncol,
                    xlo,ylo,xhi,yhi,
                    hist,cor);
    }
    else {
        Clear_message (1);
        Clear_message (2);
        Display_message (1,3,"Nao existe nenhuma
            imagem ja restaurada em
            IA!");
        Display_message (2,3,"Deve-se
            primeiramente utilizar
            a opcao RESTAURA
            ORIGINAL");
    }
    break;

case 8: /* Salva Imagens - Imagem A -> Imagem B */
    for (i=0; i<nlin; ++i)
        for (j=0; j<ncol; ++j)
            ponto_IB[i][j]=ponto_IA[i][j];
    break;

case 9: /* Salva Imagens - Imagem A -> Imagem C */
    for (i=0; i<nlin; ++i)
        for (j=0; j<ncol; ++j)
            ponto_IC[i][j]=ponto_IA[i][j];
    break;

case 10: /* Salva Imagens - Imagem A -> Arquivo */
    recebe_resposta ("Entre com o nome do
        arquivo: ", file);
    if (!strcmp(file, "quit") ||
        !strcmp(file, "exit")) {
        Clear_message(1);
        goto menubar;
    }
    if ( !( saida = fopen( file, "w" ) ) ) {
        Clear_message(1);
        Display_message (1,3, "Nao conseguiu abrir
            o arquivo !");

        Exit_tools();
        exit (1);
    }

    Clear_message (1);
    Display_message (1,3,
        "Gerando o arquivo de Saidas ...");

    fprintf(saida,"%d,%d\n",nlin, ncol);

```

```

        for (i=0; i<nlin; i++) {
            for (j=0; j<ncol; j++)
                fprintf (saida,"%d ", ponto_IA[i][j]);
            fprintf (saida,"\n");
        }
        fprintf (saida,"\n");
        fclose (saida);
        break;

case 11: /* Operar - Troca Imagem-A com Imagem-C */
    for (i=0; i<nlin; ++i)
        for (j=0; j<ncol; ++j) {
            salva=ponto_IC[i][j];
            ponto_IC[i][j]=ponto_IA[i][j];
            ponto_IA[i][j]=salva;
        }
    des_imagem (ponto_IA,nlin, ncol, xlo,ylo,
                xhi,yhi,
                hist,cor);
    break;

case 12: /* Operar - Troca Imagem-A com Imagem-B */
    for (i=0; i<nlin; ++i)
        for (j=0; j<ncol; ++j) {
            salva=ponto_IA[i][j];
            ponto_IA[i][j]=ponto_IB[i][j];
            ponto_IB[i][j]=salva;
        }
    des_imagem(ponto_IA,nlin, ncol, xlo,ylo,
                xhi,yhi,hist,cor);
    break;

case 13: /* Operar: Imagem-A <- Imagem-A + Imagem-B */
    for (i=0; i<nlin; ++i)
        for (j=0; j<ncol; ++j)
            ponto_IA[i][j]=ponto_IA[i][j] +
                ponto_IB[i][j];
    test_normaliza (ponto_IA,ponto_IA,nlin,ncol);
    des_imagem(ponto_IA,nlin, ncol, xlo,ylo,
                xhi,yhi,hist,cor);
    break;

case 14: /* Operar: Imagem-A <- Imagem-A - Imagem-B */
    for (i=0; i<nlin; ++i)
        for (j=0; j<ncol; ++j)
            ponto_IA[i][j] = ponto_IA[i][j] -
                ponto_IB[i][j];
    test_normaliza (ponto_IA,ponto_IA,nlin,ncol);
    des_imagem (ponto_IA,nlin, ncol, xlo,ylo,
                xhi,yhi,hist,cor);
    break;

case 15: /* Levanta as bordas da Imagem */
    recebe_resposta ("Entre com o fator de
        multiplicacao da funcao: ",
        resposta);
    if (!strcmp(resposta, "quit") ||
        !strcmp(resposta, "exit")) {
        Clear_message(1);
        goto menubar;
    }
    num=atoi(resposta);
    Clear_message (1);
    Display_message (1,3, "Aguarde ...");
    nivelamento(0,ncol,nlin,ncol,num,ponto_IA);
    des_imagem (ponto_IA,nlin, ncol,
                xlo,ylo,xhi,yhi,
                hist,cor);
    break;

```

```

case 16: /* Levanta o lado esquerdo da imagem */
recebe_resposta ("Entre com o fator de
multiplicacao da funcao: ",
resposta);

if (!strcmp(resposta, "quit") ||
!strcmp(resposta, "exit")) {
Clear_message(1);
goto menubar;
}

Clear_message(1);
Display_message(1,3, "Aguarde ...");
num=atoi(resposta);
nivelamento(0,ncol/4,nlin,ncol,num,ponto_IA);
des_imagem(ponto_IA,nlin,ncol,xlo,ylo,
xhi,yhi,hist,cor);
break;

case 17: /* Levanta o lado esquerdo da imagem */
recebe_resposta ("Entre com o fator de
multiplicacao da funcao: ",
resposta);

if (!strcmp(resposta, "quit") ||
!strcmp(resposta, "exit")) {
Clear_message(1);
goto menubar;
}

Clear_message(1);
Display_message(1,3, "Aguarde ...");
num=atoi(resposta);
nivelamento(3*ncol/4,ncol,nlin,
ncol,num,ponto_IA);
des_imagem(ponto_IA,nlin,ncol,xlo,ylo,
xhi,yhi,hist,cor);
break;

case 18: /* Fim */
break;

} /* fim do case */
} /* fim do while */
break;

case 4: /* Medidas Estatisticas */
pd_com4=0; /* Inicializacao */
while (pd_com4 != 4) {
Process_pd(menu_id4,0,800,20,&pd_com4);

switch (pd_com4) {
case 1: /* Media */
Clear_message(1);
Clear_message(2);
Display_message(1,3,"Entre via MOUSE a regioa
da imagem que sera feita a Media!");
Display_message(2,3,"Clique o botao central
do MOUSE no ponto inicial, e
posteriormente tambem no ponto
final");
Clear_button_data();
Wait_for_events (BUTTON_EVENT,
&cur_event_mask);
Get_button_data (&janela,&x1,&y1,&button,
&transition,&time_t);
if (button != MIDDLE_BUTTON) {
Clear_message(1);
Clear_message(2);
break;
}
}
}

```

```

if (janela != w_no) {
Clear_message(1);
Display_message(1,3, "Ponto especificado
fora da janela de imagem");
}

rubberquad (w_no,x1,y1,&x2,&y2);
Clear_message(2);
Display_message(2,3,"Calculando ...");
if (x1 > x2) { /* Troca */
aux_x=x1;
x1=x2;
x2=aux_x;
}
if (y1 > y2) { /* Troca */
aux_x=y1;
y1=y2;
y2=aux_x;
}

aux_med=Media (ponto_IA, nlin, ncol, x1,y1,
x2,y2);
Escreve_Float (aux_med,"Valor Media= ");
break;

case 2: /* Desvio-Padiao */
Clear_message(1);
Clear_message(2);
Display_message(1,3,"Entre via MOUSE a regioa
da imagem que sera calculado
o Desvio Padiao!");
Display_message(2,3,"Clique o botao central
do MOUSE no ponto inicial, e
posteriormente tambem no
ponto final");
Clear_button_data();
Wait_for_events (BUTTON_EVENT,
&cur_event_mask);
Get_button_data (&janela,&x1,&y1,&button,
&transition,&time_t);
if (button != MIDDLE_BUTTON) {
Clear_message(1);
Clear_message(2);
break;
}

if (janela != w_no) {
Clear_message(1);
Display_message(1,3,"Ponto especificado
fora da janela de imagem");
}

rubberquad (w_no,x1,y1,&x2,&y2);
Clear_message(2);
Display_message(2,3,"Calculando ...");
if (x1 > x2) { /* Troca */
aux_x=x1;
x1=x2;
x2=aux_x;
}
if (y1 > y2) { /* Troca */
aux_x=y1;
y1=y2;
y2=aux_x;
}

aux_med=Media (ponto_IA, nlin, ncol, x1,y1,
x2,y2);
if (aux_med == -1.) {
Clear_message(1);
Clear_message(2);
Display_message(1,3,"Valor do Desvio

```



```

        break;
        /* Padrao Invalido! */;
    }
    /* Calcula o desvio Padrao */
    aux_padrao = Des_padrao
    (ponto_IA, nlin, ncol, aux_med,
    xl, yl, x2, y2);
    sprintf (mesq2_pad, "%s %f",
    mesq1_pad, aux_padrao);
    Clear_message (1);
    Clear_message (2);
    Display_message (1, 3, mesq2_pad);
    break;

case 3: /* RELACAO SINAL/RUIDO */
    Clear_message (1);
    Clear_message (2);
    Display_message (1, 3,
    "Entre com a posicao da origem da regio:");
    Process_keyin (2, 3, " Entre com X: ");
    x1=Keyin_num();

    Clear_message (2);
    Process_keyin (2, 3, " Entre com Y: ");
    y1=Keyin_num();

    Clear_message (1);
    Clear_message (2);
    Process_keyin (1, 3, "Entre com o numero de
    pontos da imagem no eixo X: ");
    x2=Keyin_num();

    Clear_message (2);
    Process_keyin (2, 3, "Entre com o numero de
    pontos da imagem no eixo Y: ");
    y2=Keyin_num();

    /* Desenha um retangulo contornando
    a regio */
    weight (w_no, larg_pix-1);

    fgcolor (w_no, 28); /* sera desenhado em
    vermelho */
    hidecursor (w_no);
    move (w_no, (x1*larg_pix)+inix,
    (y1*larg_pix)+inicy);
    draw (w_no, ((x1+x2)*larg_pix)+inix,
    (y1*larg_pix)+inicy); /* a prim.linha */
    draw (w_no, ((x1+x2)*larg_pix)+inix,
    ((y1+y2)*larg_pix)+inicy); /* segunda
    linha */
    draw (w_no, (x1*larg_pix)+inix,
    ((y1+y2)*larg_pix)+inicy); /* terceira
    linha */
    draw (w_no, (x1*larg_pix)+inix,
    (y1*larg_pix)+inicy); /* quarta linha */
    flushbuffer (w_no);
    showcursor (w_no);
    flushbuffer (w_no);

    /* Calcula a media da regio */
    aux_med=Media2 (ponto, nlin, ncol, x1, yl,
    x1+x2, y1+y2);

    Escreve_Float (aux_med, mesq1_med);
    break;

case 4: /* MENUBAR */
    break;
} /* Fim do case pd_com4 */

```

```

        } /* Fim do while pd_com4 */
        break;

case 5: /* Fim */
    Clear_message (1);
    Erase_win (w_no);
    Erase_win (wt_no);
    Erase_win (wh_no);
    break;

} /* fim do switch */
} /* fim do then MENUBAR */

else { /* Foi selecao de BUTTON_EVENT */
    if (BUTTON_EVENT & cur_event_mask) { /* Foi selecao no BUTTON */
        Get_button_data (&janela, &x, &y, &button, &transition, &time_t);

        if (janela == wt_no && button == MIDDLE_BUTTON)
        { /* A solicitacao foi em TABELA DE CORES */
            Mouseposition (wt_no, &x, &y, &flag);
            if (flag != 0) /* Foi solic. algum ponto da tab. de
            cores */
                tabcor(x,y);
        }
        else /* Pode ter sido solicitado no histograma */
        {
            if (janela == wh_no && button == MIDDLE_BUTTON)
            {
                Mouseposition (wh_no, &x, &y, &flag);

                if (flag != 0) /* Foi solic. algum bloco do
                histograma */
                {
                    if (trata_hist (wh_no, x, y, &cluster_hist == 's')
                    des_imagem (ponto_IA, nlin, ncol, xlo, ylo, xhi, yhi,
                    hist_alt, cor_alt);
                    flag_hist_alt=1; /* Histograma Alterado */
                }
            }
            else /* foi na janela de Imagem */
            {
                if (janela == w_no && button == MIDDLE_BUTTON)
                {
                    Mouseposition (w_no, &x, &y, &flag);
                    rot_imagem (w_no, nlin, ncol, ponto_IA,
                    xlo, ylo, xhi, yhi);
                }
            }
        }
        /* fim do else HISTOGRAMA */
        Clear_button_data();
    } /* Fim do else BUTTON & cur_event_mask */

    /* E solicitacao de "refresh"? */
    if (cur_event_mask & REFRESH_EVENT) {
        Get_refresh_data (&wr_no, &var_no, &xlor, &ylo, &xhir, &yhir,
        &refr_mask);
        if ((wr_no == w_no) && flagimg) { /* "Refresh" a janela
        imagem */
            hidecursor (w_no);
            putblock (w_no, 31L, 0, 0, xhi-xlo, yhi-ylo, buff_img, bimgsize);
            showcursor (w_no);
            flushbuffer (w_no);
            flagimg=1;
            Clear_refresh_data ();
        }
        else {
            if ((wr_no == wh_no) && flaghist) { /* "Refresh" a janela
            do histograma */
                hidecursor (wh_no);
                putblock (wh_no, 31L, 0, 0, 1174, 333, buffhist, bhistsize);
                showcursor (wh_no);
            }
        }
    }
}

```

```

flushbuffer (wh_no);
flaghist=1;
Clear_refresh_data ();
}
else /* "Refresh" a tabela de cores */
if (wr_no == wt_no) /* "Refresh" a janela */
destab(); /* da tabela de Cores */
} /* fim do else */
} /* fim do if REFRESH_EVENT */
} /* fim do else button */

} /* fim do while */
while (mb_cmd != 5); /* FIM */
Exit_tools ();
}

```

Apêndice B

Manual de Referência do Ambiente Gráfico

Considerações sobre o Manual

Este manual agrupa as rotinas do ambiente gráfico em três categorias:

- ***Rotinas Básicas - LIEGSP:*** são as rotinas que dependem diretamente da arquitetura, ou que fazendo parte do sistema, não é desejável que o usuário comum tenha acesso a elas. Em sua grande maioria são escritas em linguagem assembly do TMS34010.
- ***Rotinas de Apoio - LIESJ:*** são as rotinas que preparam os dispositivos periféricos para serem utilizados pelo nível de rotinas de mais alto nível, ou realizam operações sobre as estruturas de dados que armazenam toda a informação sobre o ambiente.
- ***Rotinas que compõem o Gerenciador de Janelas - LIEGJ:*** são as rotinas que o programador utiliza para a criação e manutenção de janelas. Elas são responsáveis pela "aparência" das mesmas.

CRIA_ARVORE

MANIPULA ESTRUTURA(3LIESJ)

NOME

Cria_arvore - Inicializa a estrutura de Janelas.

SINOPSE

```
void Cria_arvore (maxlin, maxcol)
int maxlin, maxcol;
```

DESCRIÇÃO

A rotina *Cria_arvore* inicializa a estrutura de janelas, e o primeiro nó dessa estrutura, que corresponde à raiz da árvore, e à tela inteira do vídeo, que poderá ser utilizada. As variáveis *maxlin* e *maxcol* definem o tamanho máximo da tela em pixels. A variável *raiz* é definida estaticamente, e pode passar a ser acessada a partir de outros módulos. A raiz passa a ser definida para o fonte corrente, o qual deve ser estabelecido antes da chamada desta rotina.

NOTA

- O valor de *maxlin* e *maxcol* indicados definem a área do vídeo que poderá ser utilizada posteriormente.
- O fonte corrente será tomado como fonte da raiz, e deve ser definido antes da chamada desta rotina.

VEJA TAMBÉM

Cria_filho(3LIESJ), Retira_filho(3LIESJ), Ret_subarvore(3LIESJ), Inicializa(3LIEGJ).

EXEMPLO

```
#include <data_str.h>
#include <sysconst.h>
extern byte charset[96][16];
lie_fonte lie_fonte_corrente = {
    (byte *)charset,
    8, 16};

...
Cria_arvore(YMAX, XMAX);
```

CRIA_FILHO

MANIPULA ESTRUTURA(3LIESJ)

NOME

Cria_filho - cria uma estrutura de janela.

SINOPSE

```
#include <data_str.h>

int Cria_filho(ret_filho, corrente)
lie_janela **retfilho, *corrente;
```

DESCRIÇÃO

Esta rotina cria uma janela subordinada (filho) da janela indicada como *corrente*, recebendo por herança os parâmetros de definição dessa janela. Imediatamente seguindo à chamada desta rotina, deve-se proceder à definição dos parâmetros reais que essa janela deverá ter.

RETORNO

Valor de Retorno da Rotina (tipo int):

Indica se houve erro (vide ERROS:).

ret_filho:

descritor da janela criada.

RESTRIÇÕES

- *corrente* deve ser uma janela válida já existente.

NOTA

- Todos os parâmetros da janela *corrente* indicada são duplicados para a janela *ret_filho* criada.

ERROS

- 0 - Tudo OK. Janela criada
- 1 - Janela *corrente* indicada não é uma janela válida.
- 2 - Memória insuficiente para esta operação.

CRIA_FILHO

MANIPULA ESTRUTURA(3LIESJ)

VEJA TAMBÉM

Cria_arvore(3LIESJ), Retira_filho(3LIESJ), Ret_subarvore(3LIESJ).

EXEMPLO

```
#include <data_str.h>
extern lie_janela *raiz;
lie_janela *novajan;

Cria_filho (&novajan, raiz); /* Cria uma
    janela diretamente na tela */
```

CRIA_JAN_BORDA

CRIA JANELA(3LIEGJ)

NOME

Cria_jan_borda - Cria uma janela com borda.

SINOPSE

```
#include <data_str.h>

int Cria_jan_borda (wid, parval)
int parval;
lie_janela **wid;
```

DESCRIÇÃO

Esta rotina cria uma janela com borda, como descendente da janela corrente. Os parametros dessa janela são assumidos como tendo os mesmos valores da janela corrente, a menos daqueles que são explicitamente alterados através de um par parâmetro-valor especificado (*parval*). O identificador da nova janela é retornado como *wid*. A janela apenas é criada, porém não se torna corrente, e nem é mapeada na tela. Para que uma janela se torne corrente, usa-se a rotina *Jan_corrente*(3LIEGJ). Para mapear a janela, usa-se a rotina *Mostra_Janela*(3LIEGJ).

RETORNO

Valor de Retorno da Rotina (tipo int):

Indica se houve erro (vide ERROS:).

wid:

identificador da janela criada.

RESTRIÇÕES

- Os limites (x,y) da janela devem ser especificados em coordenadas relativas à janela corrente.
- O cursor é especificado em coordenadas relativas à janela criada.
- Os limites (x,y) da janela criada devem estar inteiramente contidos dentro da janela corrente, caso contrário a janela será truncada ("clipping") para esses limites.

CRIA_JAN_BORDA

CRIA_JANELA(3LIEGJ)

PARAMETROS TRATADOS

XLO YLO	XHI YHI
PLANE_MASK	EVENT_MASK
COR_FRENTE	COR_FUNDO
X_CURSOR	Y_CURSOR
COR_FRENTE_BORDA	
COR_FUNDO_BORDA	
TITULO	
SEM_BORDA	COM_BORDA
ELEV_VERTICAL	
ELEV_HORIZONTAL	
TRACO_SIMPLES	
TRACO_DUPLO	
SOBREPOE	
BORDA_SINGELA	

NOTA

Deve-se atentar para a exatidão da correspondência entre atributo e valor. Caso exista a falta de algum parâmetro, os argumentos serão tomados em seqüência, levando a resultados imprevisíveis.

ERROS

- 0 - Tudo OK. Janela criada corretamente.
- 1 - Estrutura da janela indicada danificada.
- 2 - Memória insuficiente para esta operação.
- 10 - Margens fora de limite. Janela criada mas truncada.
- 11 - Parâmetro Inválido.

VEJA TAMBÉM

Jan_corrente(3LIEGJ), Mostra_janela(3LIEGJ),
Cria_janela(3LIEGJ).

CRIA_JAN_BORDA

CRIA_JANELA(3LIEGJ)

EXEMPLO

```
/* Criar uma janela descendente diretamente da
tela inteira:
*/
extern lie_janela raiz;
lie_janela *cria;
. . .
Jan_corrente (&raiz);
/* torna a raiz janela
corrente */
Cria_jan_borda (&cria,
XLO, 20, YLO, 10,
XHI, 110, YHI, 200,
COR_FRENTE, 0xFFFF,
COR_FUNDO, 0,
COR_FRENTE_BORDA,
0xAAAAAAAA,
COR_FUNDO_BORDA, 0x44444444;
FIM);
```

CRIA_JANELA

CRIA_JANELA(3LIEGJ)

NOME

Cria_janela - Cria uma janela básica sem borda.

SINOPSE

```
#include <data_str.h>

int Cria_janela (wid, parval)
int parval;
lie_janela **wid;
```

DESCRIÇÃO

Esta rotina cria uma janela básica, como descendente da janela corrente. Os parâmetros dessa janela são assumidos como tendo os mesmos valores da janela corrente, a menos daqueles que são explicitamente alterados através de um par parâmetro-valor (*parval*) especificado. O identificador da nova janela é retornado como *wid*. A janela apenas é criada, porém não se torna corrente, e nem é mapeada na tela. Para que uma janela se torne corrente usa-se a rotina *Jan_corrente*(3LIEGJ). Para mapear a janela, usa-se a rotina *Mostra_Janela*(3LIEGJ).

RETORNO

Valor de Retorno da Rotina (tipo int):
Indica se houve erro (vide ERROS:).
wid:
identificador da janela criada.

RESTRIÇÕES

- Os limites (x,y) da janela devem ser especificados em coordenadas relativas à janela corrente.
- O cursor é especificado em coordenadas relativas à janela criada.
- Os limites (x,y) da janela criada devem estar inteiramente contidos dentro da janela corrente, caso contrário a janela será truncada ("clipping") para esses limites.

PARAMETROS TRATADOS

CRIA_JANELA

CRIA_JANELA(3LIEGJ)

```
XLO YLO XHI YHI
PLANE_MASK EVENT_MASK
COR_FRENTE COR_FUNDO
X_CURSOR Y_CURSOR
SOBREPOE
```

NOTA

Deve-se atentar para a exatidão da correspondência entre atributo e valor. Caso exista a falta de algum parâmetro, os argumentos serão tomados em sequência, levando a resultados imprevisíveis.

ERROS

- 0 - Tudo OK. Janela criada corretamente.
- 1 - Estrutura da janela indicada danificada.
- 2 - Memória insuficiente para esta operação.
- 10 - Margens fora de limite. Janela criada mas truncada.
- 11 - Parâmetro Inválido.

VEJA TAMBÉM

Jan_corrente(3LIEGJ), *Mostra_janela*(3LIEGJ),
Cria_jan_borda(3LIEGJ).

EXEMPLO

```
/* Criar uma janela sem borda descendente
diretamente da tela inteira: */
extern lie_janela raiz;
lie_janela *cria;
. . .
Jan_corrente (&raiz); /* torna a raiz a janela
corrente */
Cria_janela (&cria,
XLO, 20, YLO, 10,
XHI, 110, YHI, 200,
FIM);
```


ESCONDE_JANELA

MANIPULA_JANELA(3LIEGJ)

NOME

Esconde-janela - Retira uma janela da tela.

SINOPSE

```
#include <data_str.h>

int Esconde_janela (wid)
lie_janela *wid;
```

DESCRIÇÃO

A rotina *Esconde-janela* retira da tela a janela indicada, mantendo a definição da mesma na estrutura de janelas, de maneira que esta possa ser mostrada posteriormente através de uma chamada da rotina *Mostra_janela*(3LIEGJ). A janela pai da janela retirada torna-se a janela corrente. Podem ser escondidas janelas com ou sem borda.

RETORNO

Valor de Retorno da Rotina (tipo int):
Indica se houve erro (vide ERROS:).

RESTRIÇÕES

- A janela indicada deve ser uma janela válida.
- Não pode ser escondida a janela raiz (própria tela).

ERROS

- 0 - Tudo OK. Janela foi escondida.
- 1 - Estrutura de janela indicada danificada, ou não existe janela corrente.

VEJA TAMBÉM

Mostra_janela(3LIEGJ).

FIXA_JAN_FISICA

MANIPULA ESTRUTURA(3LIESJ)

NOME

Fixa_jan_fisica - Estabelece o ambiente de Hardware para a janela.

SINOPSE

```
#include <data_str.h>

int Fixa_jan_fisica (wid);
lie_janela *wid;
```

DESCRIÇÃO

Esta rotina estabelece os parâmetros físicos do hardware que permitem realizar as operações seguintes sobre a área de vídeo correspondente à janela *wid* indicada. São fixados os registradores "PMASK", "WSTART", "WEND", CONTROL, COLOR0 e COLOR1.

RETORNO

Valor de Retorno da Rotina (tipo int):
Indica se houve erro (vide ERROS:).

ERROS

- 0 - Tudo OK.
- 1 - Estrutura da janela indicada está inconsistente.

VEJA TAMBÉM

JanCorrente(3LIEGJ).

INICIALIZA**INICIALIZA SISTEMA(3LIEGJ)****NOME**

Inicializa - Prepara o Sistema para iniciar as operações.

SINOPSE

Inicializa()

DESCRIÇÃO

Esta rotina inicializa o ambiente de execução do Sistema de Gerenciamento de Janelas para aceitar solicitações de operações. Esta deve ser a primeira rotina do sistema a ser chamada, pois caso contrário as operações não serão adequadamente atendidas. A tela de vídeo é apagada, e o TMS34010 é inicializado.

RETORNO

Valor de Retorno da Rotina (tipo int):
Indica se houve erro (vide ERROS:).

RESTRIÇÕES

- Esta deve ser a primeira rotina do SGJ a ser chamada.

ERROS

- 0 - Tudo OK.
- 1 - Erro na inicialização. Sistema não está em operação.

EXEMPLO

```
main()
{
/* declarações de variáveis */
. . .
/* Início do procedimento */
Inicializa(); /* Iniciliza TMS e apaga a
tela */
. . .
```

JAN_CORRENTE**MANIPULA JANELA(3LIEGJ)****NOME**

Jan_corrente - Torna corrente a janela indicada.

SINOPSE

```
#include <data_str.h>

int Jan_corrente(wid)
lie_janela *wid;
```

DESCRIÇÃO

Esta rotina torna "corrente" a janela indicada. Janela corrente é aquela subentendida para realizar diversas operações, entre elas as operações de saída de dados. Assim, sempre que é solicitada a saída de dados, esta sempre tem efeito na janela corrente.

RETORNO

Valor de Retorno da Rotina (tipo int):
Indica se houve erro (vide ERROS:).

ERROS

- 0 - Tudo OK.
- 1 - Estrutura da janela indicada está inconsistente.

EXEMPLO

```
/* Cria uma janela na tela e a torna corrente */
#include <data_str.h>
extern lie_janela raiz;
lie_janela *jan1;
Jan_corrente (&raiz); /* torna a raiz a janela
corrente */
Cria_janela (&jan1,
XLO, 20, YLO, 10,
XHI, 110, YHI, 200, FIM);
Jan_corrente (jan1); /* torna a janela criada
corrente */
```

LIMPA_JAN_CORRENTE**MANIPULA JANELA(3LIEGJ)****NOME**

Limpa_jan_corrente - Limpa a Janela Corrente.

SINOPSE

```
void Limpa_jan_corrente();
```

DESCRIÇÃO

Esta rotina preenche toda a janela corrente com o pixel de cor de fundo definido para a janela.

RESTRIÇÕES

- Deve existir uma janela corrente consistente, e que esteja mapeada no vídeo.

VEJA TAMBÉM

Jan_corrente(3LIEGJ), Mostra_janela(3LIEGJ),
Cria_jan_borda(3LIEGJ), Cria_janela(3LIEGJ).

EXEMPLO

```
lie_janela *jan1;

/* Criar uma janela subordinada 'a
   corrente */
Cria_janela (&jan1, XLO, 20, YLO, 10,
             XHI, 110, YHI, 200,
             COR_FUNDO, 0,
             COR_FRENTE, 0xFFFF,
             FIM);
Jan_corrente(jan1);      /* Tornar corrente e
                          limpar a janela cujo
                          identificador é jan1
                          */

Mostra_janela();
Limpa_jan_corrente();
```

LISTA_ARVORE**MANIPULA ESTRUTURA(3LIESJ)****NOME**

Lista_arvore - Lista a estrutura de janelas.

SINOPSE

```
#include <data_str.h>

void Lista_arvore (wid)
lie_janela *wid;
```

DESCRIÇÃO

Esta rotina lista na interface de vídeo do hospedeiro a sub-estrutura de janelas atual a partir do nó correspondente à janela *wid* indicada. O propósito desta rotina é o de rastrear a estrutura, para a depuração de aplicativos.

VEJA TAMBÉM

Cria_arvore(3LIESJ), Cria_filho(3LIESJ), Retira_filho(3LIESJ),
Ret_subarvore(3LIESJ).

EXEMPLO

```
#include <data_str.h>
extern lie_janela *raiz;

...
Lista_arvore (raiz); /* Lista toda a
                     estrutura de janelas */
```

NOME

Mn_PDcria - Definir um Menu Pull-Down.

SINOPSE

```
#include <data_str.h>

int Mn_PDcria (red, green, blue,
               text_font, nro_entries,
               scr_size, entry, command,
               link, menu_id_ptr)

unsigned int red[4], green[4], blue[4];
/* Intensidade da cor, no indice:
   0 - cor do fundo
   1 - cor da borda
   2 - cor dos textos regulares
   3 - cor da entrada em highlight */
cadeia text_font; /* Nome do arquivo de fontes
                  para os caracteres */
int nro_entries; /* Número maximo de entradas
                 no menu */
int scr_size; /* número de entradas
               simultaneas na janela
               */
cadeia entry[]; /* Nomes das entradas */
int command[]; /* Número que deve retornar
                se essa entrada for
                escolhida */
int link[]; /* identidade do menu que
             pode dar prosseguimento 'a
             essa entrada, se ela for
             escolhida */
register int *menu_id_ptr; /* Identidade
                           desse menu,
                           atribuido pela
                           rotina */
```

DESCRIÇÃO

Esta rotina cria um Menu Pull-Down segundo a definição dada pelo usuário. As entradas do menu são definidas pelos vários vetores *entry*, *command*, e *link*, sendo que a cada índice do vetor corresponde uma entrada. O número máximo de entradas *nro_entries* determina a quantidade de entradas que o menu pode comportar. Do total de entradas do menu, apenas a quantidade indicada em *scr_size* é mostrada simultaneamente na tela, sendo que as demais são mostradas através de operações de "scroll".

Os vetores indicadores de cores *red*, *green*, e *blue* determinam a intensidade com que cada cor básica participa da formação da cor na parte correspondente do menu. Para isso, a cor é composta utilizando-se os bits de mais baixa ordem de cada entrada. A quantidade de bits utilizada depende do número de planos de cor que está sendo utilizado pelo hardware.

A entrada *text_font* destina-se a ser usada para indicar o fonte de letra a ser utilizado para escrever-se as entradas do menu, quando existir mais de um fonte definido no sistema. Caso deva ser utilizado o fonte corrente, deve ser indicado o valor <NULO>.

A variável *menu_id_ptr* no retorno da rotina conterá o identificador do menu criado, o qual deve ser utilizado para se indicar que este menu deve ser ativado, ou para se estabelecer a hierarquia de menus através do *link* de cada entrada. O *link*, se for indicado como zero indica que a entrada correspondente é folha, e se escolhida corresponde à uma escolha final, terminando o processamento do menu. Caso esse valor não seja zero, deve indicar um identificador de menu válido, o qual corresponde ao sub-menu ativado quando a entrada correspondente for escolhida.

Uma vez criado um menu, este pode ser ativado através da rotina *Mn_PDprocessa(3LIESJ)*. Sub-menus escolhidos durante o processamento de um menu são por sua vez ativados automaticamente quando escolhidos.

RETORNO

Valor de Retorno da Rotina (tipo int):
Indica se houve erro (vide ERROS:).

menu_id_ptr - identificador do menu criado.

RESTRIÇÕES

- Deve ser indicado um número de entradas *nro_entries* de no mínimo duas entradas.
- Não pode ser excedido o número máximo de menus criados definido como MAXMENU no arquivo <sysconst.h>.

ERROS

- 0 - Não houve erro.
- 12 - Número máximo de definições excedido (máximo MAXMENU).

VEJA TAMBÉM

Mn_PDprocessa(3LIESJ).

EXEMPLO

```
main()
{
    /* Definições do menu */
    int menu_princ;
    /* Definição das cores basicas */
    static int red[4] = {0,7,1,0},
              green[4] = {2,7,1,2},
              blue[4] = {0,7,0,2};
    static char *text_font = 0;
    static char *entry[]={
        "\x0f\x0f Opcao 1",
        "\x0f\x0f Opcao 2",
        "\x0f\x0f Opcao tres",
        "\x0f\x0f Opcao quatro",
        "\x0f\x0f Parametros",
        "\x0f\x0f Fim"
    };
    static int command[] = {1,2,3,4,50,500};
    static int link[] = {0,0,0,0, 0, 0};
    static int i; /* Retorno do processa menu */
```

```
/* Cria menu */
Mn_PDcria (red, green, blue, text_font, 6,
           4,
           entry, command, link, &menu_princ);
i = Mn_PDprocessa
    (menu_princ,0,5,55,&comp);
```

NOME

Mn_PDprocessa - Ativa o processamento do Menu indicado.

SINOPSE

```
#include <data_str.h>

int Mn_PDprocessa (menu_id, screen_nro,
                  lin_org, col_org, command_ptr)

int menu_id;      /* Número de identidade do
                  menu que deve ser ativado,
                  devolvido pela rotina
                  Mn_PDcria(). */
int screen_nro;   /* Número da tela de vídeo
                  que deve ser usada. */
int lin_org, col_org; /* Posição do canto
                  superior esquerdo
                  onde o menu deve ser
                  a b e r t o , e m
                  coordenadas do vídeo
                  em termos de linha e
                  coluna. */
int *command_ptr; /* Número da entrada
                  escolhida. Corresponde ao
                  número do vetor command
                  definido para esta entrada
                  para este menu. */
```

DESCRIÇÃO

Esta rotina ativa o processamento do menu indicado em *menu_id*, mostrando-o na tela, e permitindo que o usuário movimente o cursor segundo sua vontade. O menu a ser mostrado deve ter sido previamente definido pela rotina *Mn_PDcria*. Quando uma entrada é escolhida, retorna-se o valor correspondente à entrada atualmente em highlight na variável *command_ptr*. A janela correspondente ao menu é aberta na tela tendo como coordenadas de seu canto superior esquerdo aquelas indicadas em *lin_org* e *col_org*. Caso a janela não possa ser totalmente mostrada na janela a partir dessas coordenadas, estas são reduzidas em direção ao canto superior esquerdo da tela, até que a janela possa ser inteiramente mapeada. As dimensões da janela são calculadas automaticamente através das

entradas definidas para o menu.

A variável *screen_nro* não está sendo utilizada, sendo mantida na definição por motivo de compatibilidade com o ambiente "Environ V".

RETORNO

Valor de Retorno da Rotina (tipo int):
Indica se houve erro (vide ERROS:).

command_ptr - código de comando associado à entrada escolhida.
Caso nenhuma entrada tenha sido escolhida, retorna o valor -1.

- O menu indicado deve ter sido definido como um menu "pull-down"
- As dimensões do menu devem caber na tela.

ERROS

- 0 - Não houve erro. resultado válido
- 13 - menu_id indicado não válido
- 14 - menu_id indicado não definido

VEJA TAMBÉM

Mn_PDcria(3LIEGJ).

EXEMPLO

```

/* Utilizando-se do menu definido no exemplo
da rotina Mn_PDcria: */
i = Mn_PDprocessa (menu_princ,0,5,55,&comp);
if (comp == 7) break;
switch(comp){
    case 1 : /* Opção 1 */
        . . .
        break;
    case 2 : /* Opção 2 */
        . . .
        break;
    case 3 : /* Opção tres */
        . . .
        break;
    case 4 : /* Opção quatro */
        . . .
        break;
    case 5 : /* Trata parametros */
        . . .
        break;
    case 6 : /* Finaliza */
        . . .
        break;
}

```

NOME

Mostra_janela - Mapeia a janela corrente no vídeo.

SINOPSE

int Mostra_janela()

DESCRIÇÃO

A rotina *Mostra_janela* mapeia e mostra no vídeo a janela corrente. Caso a janela corrente seja uma janela com borda, a borda é mostrada também, juntamente com os atributos correntes para essa janela e borda.

RETORNO

Valor de Retorno da Rotina (tipo int):
Indica se houve erro (vide ERROS:).

RESTRICÇÕES

- Deve haver uma janela corrente consistente.

NOTA

Esta rotina efetua todo o tratamento para permitir a manipulação de janelas com ou sem borda.

ERROS

- 0 - Tudo OK.
- 1 - Estrutura da janela corrente danificada, ou não existe janela corrente.

VEJA TAMBÉM

Esconde_janela(3LIEGJ).

MOVE_CURSOR

MANIPULA JANELA(3LIEGJ)

NOME

Move_cursor - Posiciona o cursor da janela corrente.

SINOPSE

```
Move_cursor (x, y);  
int x, y;
```

DESCRIÇÃO

Movimenta o cursor da janela corrente para a posição absoluta (em relação à tela) indicada.

RETORNO

Valor de Retorno da Rotina (tipo int):
Indica se houve erro (vide ERROS:).

RESTRIÇÕES

- Deve existir uma janela corrente válida.

ERROS

- 0 - Tudo OK.
- 1 - Estrutura da janela corrente danificada, ou não existe janela corrente.

VEJA TAMBÉM

Cria_janela(3LIEGJ), Posiciona(3LIEGJ).

EXEMPLO

```
Move_cursor (10, 20); /* Move o cursor para  
a coordenada absoluta  
do vídeo (10,20). */
```

RET_SUBARVORE

MANIPULA ESTRUTURA(3LIESJ)

NOME

Ret_subarvore - Retira uma subarvore da estrutura de janelas.

SINOPSE

```
#include <data_str.h>  
  
int Ret_subarvore(wid)  
lie_janela *wid;
```

DESCRIÇÃO

Esta rotina elimina da estrutura de janelas a janela *wid* indicada, bem como todas as janelas suas subordinadas. Note-se que a rotina não verifica se a janela está mostrada no vídeo, o que implica a necessidade de se retirar a janela (e suas subordinadas) do vídeo, através da rotina *Esconde_janela* antes da chamada desta rotina.

RETORNO

Valor de Retorno da Rotina (tipo int):
Indica se houve erro (vide ERROS:).

RESTRIÇÕES

- A janela *raiz* não pode ser retirada.

ERROS

- 0 - Tudo OK.
- 1 - Janela indicada inconsistente.
- 4 - A janela *raiz* não pode ser retirada.

VEJA TAMBÉM

Cria_arvore(3LIESJ), Cria_filho(3LIESJ), Retira_filho(3LIESJ),
Esconde_janela(3LIEGJ), Retira_janela(3LIEGJ).

RETIRA_FILHO

MANIPULA ESTRUTURA(3LIESJ)

NOME

Retira_filho - Retira um nó da estrutura de janelas.

SINOPSE

```
#include <data_str.h>

int Retira_filho (wid)
lie_janela *wid;
```

DESCRIÇÃO

Esta rotina retira a janela *wid* da estrutura de janelas. Note-se que a rotina não verifica se a janela está mostrada no vídeo, o que implica a necessidade de se retirar a janela do vídeo, através da rotina *Esconde_janela* antes da chamada desta rotina. Para que esta rotina possa retirar a janela, é necessário que esta não tenha nenhuma outra janela subordinada.

RETORNO

Valor de Retorno da Rotina (tipo int):
Indica se houve erro (vide ERROS:).

RESTRIÇÕES

- A janela a ser retirada não deve estar mapeada no vídeo, e nem possuir janelas subordinadas.

ERROS

- 0 - Tudo OK.
- 1 - Janela indicada inconsistente.
- 3 - A janela possui filhos, não pode ser retirada.
- 4 - A janela raiz não pode ser retirada.

RETIRA_FILHO

MANIPULA ESTRUTURA(3LIESJ)

VEJA TAMBÉM

Cria_arvore(3LIESJ), Cria_filho(3LIESJ), Ret_subarvore(3LIESJ),
Esconde_janela(3LIEGJ), Retira_janela(3LIEGJ).

RETIRA_JANELA

MANIPULA_JANELA(3LIEGJ)

NOME

Retira_janela - Retira a janela especificada da estrutura de janelas.

SINOPSE

```
#include <data_str.h>

int Retira_janela (wid)
lie_janela *wid;
```

DESCRIÇÃO

Retira a janela especificada como *wid* da hierarquia de Janelas. Caso a janela tenha sido criada com a opção SOBREPOE (vide *Cria_janela(3LIEGJ)*), e esteja presentemente mapeada na tela, então a janela também será desmapeada da tela.

RETORNO

Valor de Retorno da Rotina (tipo int):
Indica se houve erro (vide ERROS:).

RESTRIÇÕES

- Se a janela especificada não foi criada com a opção de sobreposição, ela será apenas retirada da estrutura hierárquica, mas não será desmapeada da tela.
- Não se pode retirar uma janela que possua filhos, ou a janela raiz.

ERROS

- 0 - Tudo OK.
- 1 - Estrutura da janela indicada está danificada.
- 3 - Janela especificada possui filhos.
- 4 - Janela especificada é a raiz.

RETIRA_JANELA

MANIPULA_JANELA(3LIEGJ)

VEJA TAMBÉM

Limpa_janela(3LIEGJ), Cria_janela(3LIEGJ).

EXEMPLO

```
/* Criar uma janela descendente direta da tela
inteira: */
extern lie_janela raiz;
lie_janela *cria;
Jan_corrente (&raiz); /* torna a raiz janela
corrente */

/* Neste caso a janela não sera desmapeada da
tela, apenas da estrutura:*/
Cria_janela (&cria,
            XLO, 20,   YLO, 10,
            XHI, 110,  YHI, 200,
            FIM);
Ret_janela (cria);
. . .
/* Neste caso a janela sera desmapeada da tela
e da estrutura:*/
Cria_janela (&cria,
            XLO, 20,   YLO, 10,
            XHI, 110,  YHI, 200,
            SOBREPOE, SIM,
            FIM);
Ret_janela (cria);
. . .
```

DRAW_CAR

SAIDA DE CARACTERES(3LIEGSP)

NOME

draw_car - escreve um carácter no vídeo.

SINOPSE

```
int draw_car (pc, x, y)
char *pc;
int x, y;
```

DESCRIÇÃO

Esta rotina escreve um carácter no vídeo, no retângulo cujo vértice esquerdo superior é definido como estando na posição indicada por *x* e *y*. O carácter é indicado pelo ponteiro *pc*, e é impresso segundo o fonte presentemente definido como corrente.

RETORNO

Valor de Retorno da Rotina (tipo int):
Constante zero.

RESTRICÇÕES

- A rotina não verifica se a posição indicada corresponde à uma posição válida do vídeo, portanto é necessário que as coordenadas *x* e *y* estejam consistentes, caso contrário uma posição errônea, ou invasão de memória poderá ocorrer.

EXEMPLO

```
/* Escrever o vetor "vet" de 10 caracteres */
extern lie fonte lie fonte_corrente;
char vet[]="1234567890";
char aux;
int i;

aux=vet;
for (i=0; i<10; i++)
    draw_car (aux++,
              x+(i*lie_fonte_corrente.tam_car_x),
              y);
```

ERASE_SCREEN

MANIPULA TELA(3LIEGSP)

NOME

erase_screen - Limpa toda a tela.

SINOPSE

```
void erase_screen()
```

DESCRIÇÃO

Esta rotina coloca todos os pixels da tela com o valor 0x1515 (cinza claro). O tamanho da tela em pixels é obtido pelas constantes *XMAX* e *YMAX* definidas no arquivo *<sysconst.h>*. Note-se que esta rotina apaga a tela independentemente de existirem ou não janelas mapeadas. Se isso ocorrer, a atualização de informações em janelas posteriores à chamada desta rotina ficam truncadas. Portanto, esta rotina somente deve ser chamada quando não houverem janelas mapeadas.

RESTRICÇÕES

- Não devem existir janelas mapeadas na tela.

FILL

MANIPULA TELA(3LIEGSP)

NOME

fill - preenche um retângulo com uma cor.

SINOPSE

```
void fill(width, height, xleft, ytop)
long width, height; /* largura e altura do retângulo */
long xleft, ytop; /* coord's do canto superior esquerdo do
retângulo */
```

DESCRIÇÃO

Esta função preenche um retângulo da tela, cuja especificação é dada pelas variáveis *width* e *height*. As coordenadas (*xleft*, *yleft*) especificam o canto superior esquerdo do retângulo. A cor que preencherá o retângulo deve ser previamente carregada no registrador COLOR1 (B9), antes de se chamar esta função.

VEJA TAMBÉM

control_register (3LIEGSP)

EXEMPLO

```
...
/* Este trecho do programa preenche o
retângulo de largura 100 e altura 200, cujo
canto superior esquerdo é (0,0) com a cor
0x1515. O conteúdo do registrador CONTROL
influencia como será preenchido o retângulo */
set_color1 (0x1515);
fill(100,200,0,0);
...
```

CONTROL_REGISTER MANIPULA REGISTRADOR CONTROL(3LIEGSP)

NOME

get_control - Retorna o conteúdo do registrador CONTROL.
set_control - Estabelece o conteúdo do registrador CONTROL.

SINOPSE

```
int get_control()
int set_control(mval)

int mval; /* mascara que sera colocada no
reg. CONTROL */
```

DESCRIÇÃO

Estas rotinas obtêm/colocam o valor do registrador CONTROL do TMS34010.

RETORNO

Valor de Retorno da Rotina (qualquer) (tipo int):
Conteúdo atual do registrador CONTROL.

VEJA TAMBÉM

No manual *TMS34010: User's Guide* [TEX_86a], é apresentada uma descrição dos campos de funções do registrador CONTROL.

MANIPULA PIXEL

MANIPULA TELA(3LIEGSP)

NOME

get_pixel - Obtém o valor do pixel na coordenada especificada.
put_pixel - Estabelece o valor do pixel na coordenada especificada.

SINOPSE

```
int get_pixel (x, y)
void put_pixel (valor, x, y)
int valor;
int x, y;
```

DESCRIÇÃO

A rotina *get_pixel* obtém o valor (cor) do pixel na coordenada especificada. O valor do pixel é indicado nos "n" bits menos significativos do valor de retorno da rotina, onde "n" é o número de planos de imagem (o tamanho do pixel) do sistema em uso. Os demais bits desse valor voltam como "0".

A rotina *put_pixel* coloca o valor (cor) indicado em *valor* no pixel indicado, usando para isso os "n" bits menos significativos.

A coordenada indicada por *x* e *y* é tomada como relativa ao canto superior esquerdo da tela.

RETORNO

Valor de Retorno da Rotina (tipo int):
Indica o valor do pixel na coordenada indicada.

PERCORRE_DIRETÓRIO

PERCORRE DIRETÓRIO(3LIEGSP)

NOME

i_arqinicio - Inicia busca de arquivos, posiciona no primeiro.
i_arqproximo - Prossegue busca de arquivos, posiciona no seguinte.

SINOPSE

```
#include <data_str.h>

e      r      r      o      =
i_arqinicio(specname, atributos, cabecalho);
erro = i_arqproximo(cabecalho);

int erro;                      Código de erro
                                retornado pelo DOS
cadeia specname; Cadeia contendo a
                                especificação de procura
int atributos;                Atributos dos
                                arquivos a serem
                                procurados
ArqSpec *cabecalho;           Cabeçalho do arquivo
                                encontrado
```

DESCRIÇÃO

A rotina *i_arqinicio* percorre o diretório corrente ou o diretório especificado pelo path em *specname* em busca da primeira entrada que atenda ao nome de arquivo indicado. Esse nome pode conter wild_cards (* e ?). Além disso, esta rotina armazena as informações necessárias para que a rotina *i_arqproximo* possa encontrar a próxima entrada que atenda a mesma indicação.

A rotina *i_arqproximo* percorre as entradas de diretório restantes procurando pela próxima que atenda ao padrão de arquivo indicado como os argumentos *specname* e *atributos* da última chamada da rotina *i_arqinicio*. Esses argumentos são mantidos internamente, de maneira que esta rotina pode operar corretamente.

Cada arquivo encontrado tanto por *i_arqinicio* quanto por *i_arqproximo* retorna uma estrutura do tipo *ArqSpec*, definida no arquivo *data_str.h*.

Para que um nome de arquivo seja encontrado, o nome deve

PERCORRE_DIRETÓRIO PERCORRE DIRETÓRIO(3LIEGSP)

corresponder ao padrão de procura, bem como deve corresponder aos *atributos* solicitados. A especificação em *specname* pode ser qualquer nome correto, incluindo a especificação de driver, subdiretórios e wild-cards no nome do arquivo (na indicação de subdiretórios wild cards não são aceitos).

O parâmetro *atributos* indica os tipos de arquivos que devem ser considerados para a procura. O valor desse atributo é uma concatenação por OR dos seguintes valores:

- 0 - Arquivo normal
- 1 - Arquivo Read-only
- 2 - Arquivo Oculto (Hidden)
- 4 - Arquivo de Sistema
- 16 - Subdiretório

Dessa forma por exemplo, o valor 23 acessa todas as entradas do diretório, enquanto o valor 2 acessa todas as entradas de arquivos normais, ocultos ou não.

RETORNO

Valor de Retorno da Rotina (tipo int):
Indica se houve erro (vide ERROS:).

MÉTODO

A rotina *i_arqinicio* é usada juntamente com a rotina *i_arqproximo* para realizar uma busca em todo o diretório pelo(s) arquivo(s) que atende(m) ao padrão indicado em *specname*. A rotina *i_arqinicio* recebe a indicação do padrão a ser procurado, e dos atributos dos arquivos que devem atender a esse padrão. Essas indicações são armazenadas para serem usadas por chamadas posteriores de *i_arqproximo*, e retorna o primeiro arquivo encontrado (se houver). Chamadas posteriores de *i_arqproximo* irão retornando os arquivos seguintes que atendem ao padrão, até que seja indicado o erro 18, a partir do qual nenhum novo arquivo será achado, o que encerra a sequência de buscas.

ERROS

PERCORRE_DIRETÓRIO PERCORRE DIRETÓRIO(3LIEGSP)

- 0 - Tudo OK, arquivo encontrado;
- 2 - Path não encontrado
- 18 - Não existe nenhuma entrada que atenda ao padrão procurado.

EXEMPLO

```
cadeia specname;  
ArqSpec *cabecalho;  
  
/* Escrever o nome de todos os arquivos do  
diretório corrente do disco A: */  
specname=kcvrt ("-.a:*.*"); /* Conv. para  
formato Cadeia */  
erro = i_arqinicio (specname, 23, cabecalho)  
while (erro == 0) {  
    v_printk ("%a\n", cabecalho.fname);  
    erro = i_arqproximo (cabecalho);  
}
```

NOME

i_close - Fecha um arquivo.
i_open - Abre um arquivo.
i_read - Le dados de um arquivo.
i_seek - Posiciona dentro de um arquivo.
i_write - Escreve dados em um arquivo.

SINOPSE

```
#include <data_str.h>

erro = i_close (file);
file = i_open (nome, modo);
erro = i_read(file, buffer, num);
erro = i_seek (file, offset, como);
erro = i_write (file, buffer, num);

int file;           Índice do arquivo
                     aberto.
cadeia nome;        Nome do arquivo a ser
                     aberto
int modo;           Modo de abertura do
                     arquivo:
int erro;           Código de erro
                     retornado pelo DOS
                     para a operação.
unsigned char *buffer; Endereco do buffer de
                     transferência.
int num;            Número de bytes a
                     serem transferidos.
long offset;        Número de posições
                     (bytes) a deslocar o
                     ponteiro.
int como;          Forma de deslocar o
                     ponteiro.
```

DESCRIÇÃO

A rotina *i_close* fecha o arquivo indicado no hospedeiro. O arquivo é indicado através da variável *file*, obtida quando o arquivo foi aberto através de uma chamada anterior da rotina *i_open*.

A rotina *i_open* abre o arquivo indicado no hospedeiro, retornando um valor inteiro que identifica univocamente esse arquivo entre os arquivos correntemente abertos. O *modo* de abertura corresponde aos mesmos valores de abertura das rotinas da biblioteca padrão de "C". A posição corrente do arquivo é o primeiro byte do arquivo.

O *modo* de abertura indica:

- 0 - Abrir o arquivo para leitura;
- 1 - Abrir o arquivo para escrita;
- 2 - Abrir o arquivo para leitura e escrita.

A rotina *i_read* lê dados do arquivo no *buffer* indicado. Os dados são lidos desde a posição corrente do arquivo, até o máximo de *num* bytes. Caso essa quantidade de bytes ultrapasse o final do arquivo, apenas os bytes restantes são lidos. O número de bytes efetivamente lidos retorna como o valor de retorno da rotina. Caso esse número seja diferente de *num*, significa que o final do arquivo foi atingido. Após uma leitura, a posição corrente do arquivo passa a ser o primeiro byte seguindo ao último byte lido.

A rotina *i_seek* movimenta a posição corrente do arquivo indicado, o número de bytes indicado em *num*, de acordo com o parâmetro como:

- 0 - *num* bytes a partir do início do arquivo;
- 1 - *num* (positivo ou negativo) bytes a partir da posição corrente;
- 2 - *num* bytes voltando a partir do final do arquivo.

A rotina *i_write* escreve dados no arquivo a partir do *buffer* indicado. Os dados são escritos da posição corrente do arquivo, até o máximo de *num* bytes. Os dados escritos sobrepõem aqueles já existentes no disco. Caso os dados ultrapassem o final anterior do

arquivo, este é estendido. Caso essa quantidade de bytes esgote o espaço disponível do disco, apenas os bytes que cabem são lidos. O número de bytes efetivamente escritos retorna como o valor de retorno da rotina. Caso esse número seja diferente de *num*, significa que não foi possível escrever todo o buffer indicado. Após uma escrita a posição corrente do arquivo passa a ser o primeiro byte seguindo ao último byte escrito.

RETORNO

Valor de Retorno da Rotina *i_close* (tipo int):

ZERO: Fechou com sucesso.

-1: Não pode fechar o arquivo.

Valor de Retorno da Rotina *i_open* (tipo int):

-1: o arquivo não pode ser aberto.

valor positivo: indica o índice do arquivo aberto.

Valor de Retorno da Rotina *i_read* (tipo int):

ZERO: Fim de arquivo.

Valor positivo: número de bytes efetivamente lidos.

Valor de Retorno da Rotina *i_seek* (tipo int):

Valor positivo: posição corrente do ponteiro.

-1: pesquisa ilegal.

Valor de Retorno da Rotina *i_write* (tipo int):

Valor positivo: número de bytes efetivamente escritos.

-1: Houve erro de escrita.

NOTA:

Essas rotinas possuem utilização semelhante às rotinas de acesso a arquivos da linguagem "C".

NOME

i_getc - Obtém caráter do teclado do hospedeiro.

i_putc - Escreve um caráter no vídeo do hospedeiro.

SINOPSE

```
int carin = i_getc()
```

```
i_putc(caract)
```

```
unsigned char caract, carin;
```

DESCRIÇÃO

A rotina *i_getc* lê um caráter do teclado do hospedeiro IBM-pc. Caso não exista um caráter digitado, espera-se até que um caráter seja teclado. O valor *carin*, de retorno da rotina é o valor ASCII do caráter digitado.

A rotina *i_putc* escreve o caráter *caract* no vídeo do hospedeiro IBM-pc. Caso ainda exista um caráter anteriormente enviado ainda não recebido pelo hospedeiro, espera-se até que o caráter anterior seja escrito. O valor de retorno da rotina é o valor ASCII do caráter impresso.

RETORNO

Valor de Retorno da Rotina (tipo int):

Valor ASCII do caráter lido/escrito.

RESTRIÇÕES

• Podem ser digitados qualquer dos valores ASCII, que os mesmos serão recebidos pela rotina *i_getc*. No entanto, os valores ASCII que correspondem a movimentos do cursor serão tratados como tal pela rotina *i_putc*.

NOTA

Essas rotinas possuem utilização semelhante às rotinas tratamento de caracteres da linguagem "C".

EXEMPLO

```
/* Para ecoar todos os caracteres digitados
   até que seja teclado um caráter <^Z>:
   */
int c;
while ( (c=i_getc() ) != 26) i_putc(c);
```

NOME

i_getk - Le uma cadeia de caracteres do teclado do hospedeiro.

SINOPSE

```
#include <data_str.h>

int car = i_getk(cad, como);
int como;
cadeia cad;
int car;
```

DESCRIÇÃO

A rotina *i_getk* chama a rotina *_vgetk()* indicando que devem ser aceitos apenas os sinais de entrada do teclado, o que é realizado pela rotina *i_getc()*.

A rotina *_vgetk* lê do teclado uma sequência de caracteres, e a coloca na cadeia indicada. A cadeia pode ser editada durante a execução da rotina, através das seguintes teclas:

- <INS> - troca entre modo de inserção ou sobreposição;
- - apaga o caráter sobre o qual o cursor está;
- <BackSpace> - apaga o caráter anterior ao cursor;
- <Seta Esquerda> - volta o cursor um caráter sem apagar;
- <Seta Direita> - avança o cursor para a direita uma

posição;

- <HOME> - Coloca o cursor no início da cadeia;
- <END> - Coloca o cursor no final da cadeia e desabilita modo inserção;
- <F1> - Restaura o caráter seguinte da cadeia anterior;
- <F3> - Restaura a cadeia anterior. Note-se que a cadeia anterior vai sendo substituída por novos caracteres teclados.

Caracteres imprimíveis são colocados na cadeia. Caracteres não imprimíveis distintos das teclas de controle são ignorados. Os caracteres de controle <Seta Acima>, <Seta Abaixo>, <PgUp>, e <PgDn> podem ser considerados terminadores do processo de edição ou não, dependendo do valor da variável *como*: se 0 não os considera terminadores; se 1 esses caracteres são terminadores. As

teclas <RETURN>, <Line Feed> e <^C> sempre são consideradas como terminadores. A rotina devolve como valor de retorno o código do caráter que terminou a edição. Quando a cadeia é totalmente preenchida, os demais caracteres sobrepoem-se ao último anterior. Se o tamanho da cadeia for menor ou igual a zero, a rotina não é executada, e retorna com o valor -1.

RETORNO

Valor de Retorno da Rotina (tipo int):

- 1 - Cadeia indicada tem tamanho nulo ou negativo.
- >0 - Código ASCII do caráter que encerrou a entrada da cadeia. Pode ser os caracteres:

<Line Feed>	(10)	
<Return>	(13)	
<^C>	(3)	
<Seta Acima>	(200)	se <i>como</i> == 0
<Seta Abaixo>	(208)	se <i>como</i> == 0
<PgUp>	(201)	se <i>como</i> == 0
<PgDn>	(209)	se <i>como</i> == 0

RESTRIÇÕES

- A cadeia deve ter capacidade para armazenar ao menos 1 caracter.
- Devem estar determinados os parâmetros básicos do Sistema de Janelas, tais como fonte corrente, e o mecanismo de comunicação com o hospedeiro.

ERROS

- 1 - Cadeia nula ou de tamanho negativo.

VEJA TAMBÉM

v_inflag(3LIEGSP).

EXEMPLO

```
unsigned char cad[32]={30,0};
i_getk(cad, 0);
```

NOME

i_le_imagem - Lê no hospedeiro uma imagem para uma janela.

SINOPSE

```
#include <data_str.h>

int erro = i_le_imagem (file, xlo, ylo,
titulo, *nlin, *ncol);

int erro;          Valor de retorno da rotina.
int file;          Arquivo de onde será lida a
imagem.
int xlo, ylo       Posição relativa à janela
corrente, do canto superior
esquerdo da janela a ser
aberta para mostrar a
imagem.
int *nlin, *ncol;  Tamanho da janela criada
pela rotina para armazenar a
imagem.
cadeia titulo;     Cabeçalho a ser colocado na
janela.
```

DESCRIÇÃO

Esta rotina lê do arquivo indicado (que deve ser previamente aberto com a rotina *i_open*(3LIEGSP)) uma imagem no formato de imagens, obtendo o número de linhas (*nlin*) e colunas (*ncol*), e abrindo uma janela filho da janela corrente, nas coordenadas relativas à essa janela correspondente à *xlo* e *ylo*. A imagem é lida e transferida diretamente para essa janela. Note-se que a janela corrente deve ter tamanho suficiente para mostrar a imagem a partir da coordenada indicada, caso contrário a imagem será truncada. Caso a cadeia *titulo* não seja nula, seu conteúdo é mostrado no cabeçalho da janela criada. A janela criada retorna da rotina como a janela corrente.

RETORNO

Valor de Retorno da Rotina (tipo int):

-1: o arquivo não pode ser aberto.

valor positivo: indica o índice do arquivo aberto para a leitura da janela (*file*).

RESTRICÇÕES

- A janela corrente deve ter tamanho suficiente para mostrar a imagem a partir da coordenada indicada, caso contrário a imagem será truncada.
- O arquivo indicado deve ter uma imagem armazenada segundo o padrão de imagens.

ERROS

-1 - o arquivo indicado não está acessível, ou não está aberto.

retorno > 0 - código do arquivo indicado para leitura da imagem.

VEJA TAMBÉM

Jan_corrente(3LIEGJ), *i_open*(3LIEGSP).

EXEMPLO

```
/* Mostrar a imagem armazenada no arquivo
   "minha.img" numa janela criada
   diretamente sobre a tela. */
#include <data_str>
extern lie_janela raiz;
int file, nl, nc;

file=i_open(kcvt ("-.minha.img",0);
Jan_corrente (&raiz);
i_le_imagem (file, 50, 40, &nl, &nc,
             kcvt ("-.Minha Imagem") );
```

NOME

i_pollc - Verifica se há caráter teclado.

SINOPSE

```
int flag = i_pollc()
```

DESCRIÇÃO

A rotina *i_pollc* verifica se foi digitado um caráter no teclado do Hospedeiro IBM-pc. Caso não exista um caráter digitado, a rotina retorna com o valor ZERO (0). Caso exista um caráter teclado, o valor de retorno é o valor ASCII do caráter digitado, porém o caráter não é retirado do buffer.

RETORNO

Valor de Retorno da Rotina (tipo int):

0 - não existe caráter digitado.

>0 - Valor ASCII do caráter lido/escrito.

RESTRIÇÕES

- Apesar da rotina retornar o código do caráter digitado, este não é retirado do buffer, de maneira que chamadas sucessivas desta rotina sempre retornará o mesmo caracter. Para retirar o caracter, é necessário que seja chamada a rotina *i_getc*.

VEJA TAMBÉM

i_getc(3LIEGSP).

EXEMPLO

```
int c;
while (1)
    if (i_pollc() == 0)
        faz_outra_coisa();
    else
        trata_caracter(i_getc());
```

NOME

i_printk - Saída formatada de dados no vídeo do hospedeiro.

v_printk - Saída formatada de dados no vídeo do GSP.

SINOPSE

```
#include <data_str.h>
```

```
void i_printk (fmt, args)
void v_printk (fmt, args)
char *fmt;
char args;
```

DESCRIÇÃO

A rotina *i_printk* envia caracteres para o vídeo do hospedeiro IBM-PC.

A rotina *v_printk* envia caracteres para o vídeo do próprio TMS.

Ambas as rotinas usam o esquema de formatação da mesma maneira que a rotina *printf*(3).

VEJA TAMBÉM

i_putc(3LIEGSP), *v_wpuc*(3LIEGSP).

EXEMPLO

```
i_printk
    ("Isto é escrito no hospedeiro: %d\n",
    10);
v_printk ("Isto é escrito no TMS: %d\n",
    10);
```

Posiciona**MANIPULA JANELA(3LIEGJ)****NOME**

POSICIONA - Posiciona o cursor na janela especificada.

SINOPSE

```
int Posiciona(jan, x, y)
int x, y;
lie_janela *jan,
```

DESCRICAO

Posiciona o cursor dentro da janela especificada - "jan".

RETORNO

Valor de Retorno da Rotina:
Erro (vide ERROS:).

RESTRICOES

A janela deve ter sido definida corretamente, e sua estrutura deve estar integra.

ERROS

- 0 - Tudo OK.
- 1 - Estrutura da janela indicada danificada.

VEJA TAMBEM

move_cursor(3LIEGJ).

INIT_SDB**INICIALIZA GSP(3LIEGSP)****NOME**

init_sdb - Inicializa o GSP.

SINOPSE

```
int init_sdb()
```

DESCRICAÇÃO

Esta rotina inicializa o processador gráfico GSP, de maneira a que as operações de manipulação da tela e dos recursos de E/S gráficas possam ser efetuadas. Esta rotina deve ser chamada uma única vez por qualquer programa, antes que qualquer operação de exibição ou comunicação deste pacote possam ser executadas.

Esta rotina é chamada pela rotina *Inicializa(3LIESJ)*, a qual prepara também a estrutura de janelas para operação do pacote gráfico. Assim, esta rotina somente deverá ser chamada por programas que não utilizem a rotina *Inicializa(3LIESJ)*, quando as operações de manipulação de janelas não forem necessárias, realizando-se diretamente o acesso à tela do TMS, sem utilização do Sistema de Janelas.

RETORNO

Não existe valor de retorno.

RESTRICÇÕES

- Esta rotina deve ser chamada uma única vez, antes de qualquer outra operação do sistema gráfico.
- Caso esta rotina seja chamada, a rotina *Inicializa(3LIESJ)* não pode ser usada, bem como o Sistema de Janelas.

INIT_SDB

INICIALIZA GSP(3LIEGSP)

EXEMPLO

```
main()
{
  /* Declaração de variáveis */
  . . .
  init_sdb();
  . . .
}
```

KCVT

MANIPULA CADEIAS(3LIEGSP)

NOME

kcv - Converte vetor para cadeia de caracteres.

SINOPSE

```
cadeia cad = kcv (vet)
cadeia cad;
unsigned char *vet;
```

DESCRIÇÃO

Esta rotina determina o número de caracteres que existe em um vetor de caracteres desde a posição de índice 2 até a ocorrência do primeiro caráter <NULL>. Para isso é necessário que as posições de índice 0 e 1 contenham respectivamente os caracteres <-> e <.>, os quais são substituídos pela rotina pelo número de caracteres determinado. Dessa maneira o vetor passa a representar uma cadeia no padrão CCMX&CO (Cadeia com campo máximo e corrente - Sistema GeO [TRA_89]).

A finalidade desta rotina é converter para o formato padrão CCMX&CO um vetor inicializado como uma constante string pelo compilador.

Note-se que os dois primeiros caracteres devem ser <-> e <.>, os quais são mudados para o tamanho máximo e corrente da cadeia, o que faz com que apenas a primeira chamada da rotina para um mesmo vetor seja efetiva. Chamadas consecutivas não encontrarão os caracteres <-> e <.>, e serão rejeitadas, deixando a cadeia inalterada. Por outro lado, se a cadeia for modificada posteriormente, o valor da string original inicialmente inicializada pelo compilador passará a conter o valor modificado.

RETORNO

Valor de Retorno da Rotina (tipo int):

Ponteiro para a cadeia alterada, o qual é o mesmo ponteiro passado como o argumento da rotina.

RESTRICÇÕES

- O tamanho máximo da cadeia não pode ultrapassar 128 bytes, caso contrário esse será o tamanho considerado.
- A cadeia original, caso esteja com os caracteres <-> e <.> deve ser encerrada com um caráter <NULL>.

EXEMPLO

Em ambos os exemplos abaixo a cadeia "cad" passa a representar uma cadeia de tamanho máximo 7, tamanho corrente 6, contendo o valor "cadeia":

Exemplo 1:

```
char cad[]="- .cadeia";
```

```
    . .
kcv t (cad);
```

Exemplo 2:

```
char *cad;
```

```
    . .
cad=kcv t ("- .cadeia");
```

NOME

move_rect - Move retângulo de pixels.

SINOPSE

```
void move_rect (ncol, nlin, xorg, yorg, xdest,
ydest)
long ncol, nlin;  Número de colunas e de
                  linhas de ambos os
                  retângulos.
long xorg, yorg;  Coordenada do canto superior
                  esquerdo do retângulo
                  origem.
long xdest, ydest; Coordenada do canto
                  superior esquerdo do
                  retângulo destino.
```

DESCRIÇÃO

A rotina *move_rect* copia o retângulo origem especificado através de seu número de colunas *ncol* e de linhas *nlin*, cujo canto superior esquerdo está na coordenada de tela indicada por *xorg* e *yorg*, para o retângulo destino, de mesmo tamanho, cujo canto superior esquerdo está nas coordenadas de tela indicada por *xdest* e *ydest*.

RESTRICÇÕES

- Esta rotina opera na região de memória de tela, e portanto pode ser utilizada para a cópia de retângulos entre diferentes páginas de tela, porém não pode ser utilizada para a cópia de dados entre tela e matrizes de pixels.
- Como a cópia é efetuada diretamente pixel a pixel da origem para o destino, se houver sobreposição entre os retângulos origem e destino, o resultado pode ser imprevisível.

EXEMPLO

```
/* Copia um retângulo 20 linhas abaixo */
move_rect (10, 20, 20, 20, 40, 20)
```

PCHAR

ENTRADA/SAÍDA(3LIEGSP)

NOME

pchar - Desenha um caráter na posição corrente do cursor.

SINOPSE

```
int pchar (caract)
char caract;
```

DESCRIÇÃO

A rotina *pchar* desenha um caráter na posição indicada pelo cursor, dentro da janela corrente, e segundo o fonte corrente. Uma vez desenhado o caracter, o cursor é deslocado na linha para que chamadas consecutivas da rotina desenhem os caracteres em sequência. Os seguintes caracteres de controle são também interpretados:

- <Back Space> - volta o cursor o espaço de um caracter;
- <Line Feed> - Avança o cursor para a linha seguinte;
- <Return> - Posiciona o cursor no início da linha corrente;
- <Tab> - Posiciona o cursor numa posição múltipla de 8 caracteres na linha corrente, a frente da posição atual do cursor.

Antes do caráter ser escrito, a posição corrente do cursor é avaliada para verificar se está dentro da janela corrente. Caso esteja fora da janela, abaixo da mesma, considera-se a necessidade de um rolamento vertical da janela, para posicionar o cursor dentro da mesma. Caso o cursor esteja acima da janela, nenhuma ação é tomada, o mesmo ocorrendo caso o cursor esteja a direita ou a esquerda da janela.

RETORNO

Valor de Retorno da Rotina (tipo int):

A rotina sempre retorna a constante Zero.

PCHAR

ENTRADA/SAÍDA(3LIEGSP)

ERROS

0 - Tudo OK.

NOTA

Esta rotina é usada internamente para as operações de saída de caracteres na tela do TMS, tendo portanto a intensão de ser utilizada como rotina interna do sistema.

VEJA TAMBÉM

EXEMPLO

```
/* Escrever todos os caracteres do fonte
   corrente, 10 caracteres por linha: */
for (i=32; i<128; i++)
    if ( 10*(int)(i/10)==i ) {
        pchar (10); pchar (13)
    }
    else {
        pchar(i);
```


COR DE FRENTE E FUNDO ESTABELECE CORES(3LIEGSP)

NOME

`set_color0` - Estabelecer cor de fundo.
`set_color1` - Estabelecer cor de frente.

SINOPSE

```
int set_color0 (cor)
int set_color1 (cor)

long cor;
```

DESCRIÇÃO

Estas rotinas determinam a cor respectivamente de fundo e de frente com que são desenhadas as primitivas gráficas e as letras nas chamadas subsequentes. A cor é determinada em função do número de planos de cor do sistema corrente. Se o tamanho do pixel é de n bits, o padrão de cor deve ser replicado na cor $32/n$ vezes, até que se completem os 32 bits do valor da *cor*.

RETORNO

Valor de Retorno da Rotina (tipo int):
Valor de cor especificado.

RESTRIÇÕES

- Caso o valor não replique a cor ao longo dos 32 bits, cores diferentes serão alocadas em diferentes colunas de pixels, resultando em um padrão heterogêneo de cores.

EXEMPLO

```
/* Alocar a cor de frente 0x45, e a cor 0x8
para um sistema com 8 planos de bits: */
set_color0 (0x08080808);
set_color1 (0x45454545);
```

V_INSFLAG

ENTRADA/SAÍDA(3LIEGSP)

NOME

`v_insflag` - Estabelece condição inicial para inserção/sobreposição de caracteres.

SINOPSE

```
int flagv = v_insflag(flag)
register short flag, vflag;
```

DESCRIÇÃO

A rotina `v_insflag` estabelece a condição inicial de chamada da rotina `i_getk(3LIEGSP)`. Se *flag*=0, a rotina sobrepõe os caracteres digitados, caso contrário os caracteres são inseridos. Note-se que o conteúdo antigo da cadeia é sempre descartado, independentemente do valor de *flag*. O valor de retorno desta rotina é o valor de *flag* anterior. Note-se que o uso da tecla <INS> durante a execução da rotina `i_getk(3LIEGSP)` altera o valor do *flag*, mesmo para outras chamadas da rotina `i_getk(3LIEGSP)`.

RETORNO

Valor de Retorno da Rotina (tipo int):
Valor anterior de *flag*.

NOTA

Existe um *flag* do sistema que registra a condição atual de sobreposição ou inserção. Esse valor é sobreposto pela chamada da rotina `v_insflag` ou através do uso da tecla <INS> quando a rotina `i_getk` está ativa.

VEJA TAMBÉM

`i_getk(3LIEGSP)`

EXEMPLO

```
/* ler uma cadeia com inserção ligada a
princípio: */
v_insflag(1);
i_getk(cad, 0);
```

NOME

v_lugc - Obter a posição do cursor na janela corrente.
v_posic - Posicionar o cursor na janela corrente.

SINOPSE

```
void v_lugc (plin, pcol)
void v_posic (lin, col)

short *plin, *pcol;
short lin, col;
```

DESCRIÇÃO

A rotina *v_lugc* obtém a coordenada corrente do cursor na janela corrente, em coordenadas relativas a essa janela (em coordenadas de linha e coluna - posição de caracteres em fonte corrente).

A rotina *v_posic* coloca o cursor da janela corrente na posição indicada.

RETORNO

plin e *pcol* retornadas pela rotina *v_lugc*:
 As variáveis apontadas por *plin* e *pcol* obtêm a posição corrente do cursor nessa janela.

EXEMPLO

```
/* Avançar o cursor 3 pixels na horizontal, e
5 pixels na vertical: */

short lin, col;
v_lugc (&lin, &col);
v_posic ((short(lin+5), short(col+3)));
```

NOME

v_wputc - Escreve caráter na janela corrente.
v_wputk - Escreve cadeia de caracteres na janela corrente.
v_wputs - Escreve vetor de caracteres na janela corrente.

SINOPSE

```
int v_wputc(c)
v_wputs(str)
v_wputk(cad)

short c;
char *str;
cadeia cad;
```

DESCRIÇÃO

A rotina *v_wpuc* coloca o caráter *c* indicado no vídeo, na posição atual do cursor, na janela corrente.

A rotina *v_wputs* coloca o vetor de caracteres *str* indicado no vídeo, a partir da posição atual do cursor. Avança o cursor para o final do vetor escrito. Os caracteres são escritos com o atributo de vídeo corrente, na janela corrente.

A rotina *v_wputk* coloca a cadeia de caracteres *cad* indicada no vídeo, a partir da posição atual do cursor. Avança o cursor para o final da cadeia escrita. Os caracteres são escritos com o atributo de vídeo corrente, na janela corrente.

Os caracteres são colocados na janela corrente, na posição corrente do cursor, utilizando-se os atributos de vídeo corrente, tal como cores de frente e fundo, e o fonte de caracteres corrente.

RETORNO

Valor de Retorno da Rotina *v_wputc* (tipo int):
Código do carácter escrito.

Valor de Retorno das Rotinas *v_wputs* e *v_wputk* (tipo int):
Constante zero.

EXEMPLO

```
unsigned char cad[42]= {40, 3, 'a', 'b',  
                        'c'};  
v_wputc ((short) '>');  
v_wputs  
{"Cadeia entrada pelo teclado:\n"};  
v_getk (cad,0); v_putk(cad);
```

Apêndice C

Manual de Referência do Subsistema de Comunicação com o Hospedeiro

*Protótipos das Rotinas que Executam as
Funções de E/S no Hospedeiro*

TMS_SENDC

COMUNICAÇÃO GSP/IBM(3LIECGI)

NOME

tms_sendc -- Transmite o caráter digitado para o GSP.

SINOPSE

```
int tms_sendc (carac)
unsigned char carac;
```

DESCRIÇÃO

A rotina tms_sendc transmite o caráter indicado em *carac*, digitado no teclado do hospedeiro IBM-PC para o GSP. Caso exista um caráter anteriormente transmitido e ainda não aceito pelo GSP, aguarda-se pela recepção do mesmo.

NOTAS

- Caso mais de um caráter seja enviado antes que o GSP receba o primeiro, a rotina entra em modo de espera, até que o caráter anterior tenha sido recebido.
- A rotina correspondente que executa as funções complementares de comunicação de E/S no GSP é a *i_getc(3LIEGSP)*.

VEJA TAMBÉM

tms_send_poll(3LIECGI), i_getc(3LIEGSP),
i_pollc(3LIEGSP).

TMS_RECEIVEC

COMUNICAÇÃO GSP/IBM(3LIECGI)

NOME

tms_receivec - Recebe do GSP um caráter para ser mostrado no vídeo.

SINOPSE

```
int tms_receivec ()
```

DESCRIÇÃO

A rotina tms_receivec recebe do GSP um caráter, para que o mesmo seja mostrado no vídeo do hospedeiro. Caso não exista um caráter recebido, espera-se até que o GSP envie um caráter. O valor de retorno da rotina é o valor ASCII do caráter recebido.

RETORNO

Valor de Retorno da Rotina (tipo int):
Valor ASCII do caráter recebido do GSP.

NOTAS

- Caso esta rotina seja chamada e não exista um caráter enviado pelo GSP para ser mostrado no vídeo, a rotina aguarda para que isso ocorra, "travando" a operação do programa. Para evitar isso, é conveniente que a rotina *tms_receive_poll(3LIECGI)* seja chamada antes, para garantir a existência a priori de um caráter enviado.
- As rotinas correspondentes que executam as funções complementares de comunicação de E/S no GSP são os seguintes: *i_putc(3LIEGSP)*, *i_pollc(3LIEGSP)*, *i_printk(3LIEGSP)*, *i_putk(3LIEGSP)*, *i_puts(3LIEGSP)*.

VEJA TAMBÉM

tms_receive_poll(3LIECGI), i_putc(3LIEGSP),
i_pollc(3LIEGSP), i_printk(3LIEGSP), i_putk(3LIEGSP),
i_puts(3LIEGSP).

TMS_SEND_POLL COMUNICAÇÃO GSP/IBM(3LIECGI)

NOME

`tms_send_poll` -- Verifica se existe carácter transmitido pelo hospedeiro.

SINOPSE

```
int tms_send_poll ()
```

DESCRIÇÃO

A rotina `tms_send_poll` verifica se existe um carácter transmitido pelo hospedeiro IBM-PC e ainda não aceito pelo GSP. Caso exista, retorna-se o código ASCII do carácter. Se não existir um carácter não aceito, a rotina retorna Zero (0).

RETORNO

Valor de Retorno da Rotina (tipo int):

- 0 - Não existe carácter enviado ainda não recebido pelo GSP.
- >0 - Valor ASCII do carácter já enviado.

NOTA

- Esta rotina não retira o carácter do buffer, o que deve ser feito pela rotina `tms_sendc(3LIECGI)`.

VEJA TAMBÉM

`tms_send(3LIECGI)`

TMS_RECEIVE_POLL COMUNICAÇÃO GSP/IBM(3LIECGI)

NOME

`tms_receive_poll` -- Verifica se existe carácter enviado pelo GSP.

SINOPSE

```
int tms_receive_poll ()
```

DESCRIÇÃO

A rotina `tms_receive_poll` verifica se existe um carácter enviado pelo GSP e ainda não aceito pelo hospedeiro. Caso exista, retorna-se o código ASCII do carácter. Se não existir um carácter não aceito, a rotina retorna Zero (0).

RETORNO

Valor de Retorno da Rotina (tipo int):

- 0 - Não existe carácter enviado ainda não recebido pelo GSP.
- >0 - Valor ASCII do carácter já enviado.

NOTA

- Esta rotina não retira o carácter do buffer, o que deve ser feito pela rotina `tms_receivec(LIEGSP)`.

VEJA TAMBÉM

`tms_receivec(LIEGSP)`

NOME

tms_disco -- Atende pedido de manipulação de arquivos.

SINOPSE

```
int tms_disco()
```

DESCRIÇÃO

A rotina tms_disco recebe do GSP a solicitação de uma operação em disco e a executa. Caso não exista nenhuma solicitação pendente, a rotina retorna imediatamente. Havendo alguma solicitação, a operação solicitada é executada. A decodificação dos parâmetros particulares a cada solicitação é feita especificamente em função da operação solicitada.

Os dados são recebidos e/ou transmitidos diretamente ao buffer de comunicação. Quando alguma operação em disco necessita que um arquivo seja especificado, tal especificação é transmitida na estrutura de dados ArqSpec (ver seção 6.4.6).

As operações tratadas são:

ARQINICIO - Obtém o cabeçalho do primeiro arquivo no hospedeiro que se adequa a especificação indicada.

ARQPROXIMO - Obtém o cabeçalho do próximo arquivo do hospedeiro que se adequa a especificação indicada.

LEARQ - Lê o arquivo indicado no hospedeiro e o transmite ao GSP.

ESCARQ - Escreve os dados indicados pelo GSP no arquivo especificado do hospedeiro.

ABREARQ - Abre o arquivo especificado pelo GSP no hospedeiro.

FECHARQ - Fecha o arquivo especificado pelo GSP no hospedeiro.

POSICIONARQ - Posiciona o arquivo no byte subsequente ao *offset* solicitado.

LEIMAGEM - Lê o arquivo especificado em disco, o qual contém os dados de uma imagem tomográfica formatados conforme o sistema de Processamento de Imagens desenvolvido [TRA_89], e a transfere para o GSP.

NOTA

- As rotinas correspondentes que executam as funções complementares de comunicação de E/S no GSP são as seguintes: *i_arqinicio(3LIEGSP)*, *i_arqproximo(3LIEGSP)*, *i_open(3LIEGSP)*, *i_close(3LIEGSP)*, *i_read(3LIEGSP)*, *i_write(3LIEGSP)*, *i_seek(3LIEGSP)*, *i_le_imagem(3LIEGSP)*.

VEJA TAMBÉM

i_arqinicio(3LIEGSP), *i_arqproximo(3LIEGSP)*, *i_open(3LIEGSP)*, *i_close(3LIEGSP)*, *i_read(3LIEGSP)*, *i_write(3LIEGSP)*, *i_seek(3LIEGSP)*, *i_le_imagem(3LIEGSP)*.

Apêndice D

Arquivos de "Headers"

Para o Ambiente Gráfico

- *data_str.h*
- *i_disco.h*
- *lie_sgj.h*
- *sysconst.h*


```

/*****
*
*   filename: data_struct.h
*
*   Este arquivo contem a definicao das estruturas de dados utilizadas
*   para a manutencao do ambiente grafico WINDOW-LIE.
*
*-----
*   Historico:
*   - Data de criacao: 06-02-90      .... Agn J. M. Traina
*   - Inclusao da estrutura de fonte corrente      08-12-90
*   - Inclusao da estrutura para tipo de janela    31-01-91
*****/

```

```

#define DADO_PRONTO 1
#define DADO_ACEITO 0

```

```

typedef unsigned char byte, *cadeia;

```

```

typedef struct LIE_tipo_janela { /* Mantem a especificacao da
                                aparencia da janela */
    byte tipo_borda; /* bit 0 -> sem/com borda (0/1)
                       bit 1 -> 1/2 tracos
                       bit 2 -> sem/com elev. Vertical
                       bit 3 -> sem/com elev. Horiz.
                       bit 4 -> aberta/colapsada
                       bit 5 -> borda ampla/singela */
    byte z_janela; /* qual a profundidade da janela,
                    indice do vetor prof_janela */
    char *texto; /* A cadeia de carac que Contem o nome
                  da janela */
    int elev_v, /* Posicao do elevador Vert.
                (em termos de Y) */
        elev_h; /* Posicao de elevador Horiz.
                (em termos de X) */
    int limite_v, /* Posicao maxima que o elevador
                  pode ocupar */
        limite_h;
} lie_tipo_janela;

```

```

typedef struct LIE_janela { /*Definicao do no basico */
    struct LIE_janela *pai; /* Ponteiro para o pai */
    struct LIE_janela *prim_filho;
    struct LIE_janela *prox_irmao;
    struct LIE_janela *ant_irmao;
    lie_tipo_janela *tipo;
    int sentinela; /* Campo para teste de consistencia */
    int xlo, /* (xlo,ylo) definem canto superior */
        ylo, /* esquerdo da janela */
        xhi, /* (xhi,yhi) definem canto inferior */
        yhi; /* direito da janela */
    unsigned short plane_mask;
    unsigned short event_mask;
    unsigned int cor_frente,
        cor_fundo;
    int x_cursor, /* (x_cursor,y_cursor) definem a */
        y_cursor; /* posicao inicial do cursor na
                  janela */
    void *arq_fonte; /* Ponteiro para o arquivo de
                     textos fonte */
    int *base; /* Endereco da area de memoria p.
                salvamento */
    int spare1; /* Reservados p. futura ampliacao */
    int spare2;
    byte estado; /* Estado(bit 0) = 1, jan. mapeada na tela
                  = 0, jan. nao mapeada */
} lie_janela;

```

```

typedef struct LIE_fonte { /* Definicao do fonte de letras corrente */
    void *fonte;
    int tam_car_x, tam_car_y; /* Especf. tamanho de
                              cada caracter */
} lie_fonte;

```

```

struct LIE_prof_janela { /* Cada elem. da estrutura indica
                          prof. de cada janela */
    lie_janela *qual;
    byte profundidade;
} ;

```

```

struct LIE_comunica { /* Estrutura de comunicacao com host IBM-PC */
    short controle_teclado,
        dado_teclado,
        controle_video,
        dado_video,
        controle_disco,
        dado_disco;
    int endereco_disco,
        auxiliar_disco; } ;

```

```

/*****
*
*   filename: i_disco.h
*
*   Este arquivo contem a definicao de todas constantes e das estruturas
*   de dados utilizadas para se fazer qualquer acesso a disco.
*   Faz parte do modulo de comunicacao, utilizado pela parte de host
*   IBM-PC.
*
*-----
*   Historico:
*
*   - Data de criacao: 13-02-91      .... Agn J. M. Traina
*****/

```

```

#define ARQINICIO 1
#define ARQPROXIMO 2
#define LEARQ 3
#define ESCARQ 4
#define ABREARQ 5
#define FECHARQ 6
#define POSICIONARQ 7
#define LEIMAGEM 8
#define CONTINUA 9

```

```

typedef struct { /* Estrutura do cabecalho de arquivo */
    short fattr; /* Attribute (1 byte) */
    unsigned short ftime; /* Time stamp */
    unsigned short fdate; /* Date stamp */
    long fsize; /* File size in bytes */
    unsigned char fname[16]; /* File name */
} ArqSpec;

```

```

/*****
*
*   filename: lia_sgj.h
*
*   Este arquivo contem a definicao de todos os parametros
*   que podem ser usados nas rotinas do ambiente grafico
*   WINDOW-LIX.
*
*-----
*
*   Historico:
*   - Data de criacao: 28-01-91      .... Agnes J. M. Truina
*****/

#define MIN_PAR      -4  /* Menor indice de parametro valido */
#define MAX_PAR      22  /* Maior indice de parametro valido */

#define FIM          0   /* parametro para finalizar uma      *
                        *      lista de parametros      */
#define SIM          1   /* Resposta afirmando um parametro */
#define NAO          0   /* Resposta Negando um parametro */

#define PASSA_INT     -1  /* Parametro inteiro para ignorar */
#define PASSA_SHORT   -2  /* Parametro short para ignorar */
#define PASSA_BYTE    -3  /* Parametro byte ou char para ignorar */
#define PASSA_PONTEIRO -4 /* Parametro ponteiro para ignorar */

#define XLO           1   /* (xlo,ylo) definem canto superior */
#define YLO           2   /*      esquerdo da janela */
#define XHI           3   /* (xhi,yhi) definem canto inferior */
#define YHI           4   /*      direito da janela */
#define PLANE_MASK    5
#define EVENT_MASK    6
#define COR_FRENTE     7
#define COR_FUNDO      8
#define X_CURSOR      9   /* (x_cursor,y_cursor) definem posicao */
#define Y_CURSOR     10   /*      inicial do cursor na janela */
#define FONTE         11  /* O arquivo de textos fonte */

#define TAM_CAR_X     12  /* Especf. tamanho (pixels) do carc. em X */
#define TAM_CAR_Y     13  /* Especf. tamanho (no. de pixels) em Y */

#define COR_FRENTE_BORDA 14 /* Para Esp. do tipo da janela */
#define COR_FUNDO_BORDA 15
#define TITULO         16
#define COM_BORDA       17
#define ELEV_VERTICAL  18
#define ELEV_HORIZONTAL 19
#define TRACO_DUPLA    20
#define SOBREPOE       21
#define BORDA_SINGELA  22

union paramval { /* Estrutura de Dados usadas para se obter o par */
    int *parametro; /* ATRIBUTO - VALOR */
    int *valorint;
    short *valorshort;
    unsigned char *valorbyte;
    unsigned int **valorpont;
};

```

```

/*****
*
*   filename: sysconst.h
*
*   Este arquivo contem as constantes de hardware do
*   sistema grafico utilizado.
*
*-----
*
*   Historico:
*   - Data de criacao: 24-01-91      .... Agnes J. M. Truina
*****/

#define XMAX          460
#define YMAX          230

#define MAXJANELAS    256

#define MN_NCAR 32     /* Numero maximo de caracteres numa entrada de
                        *      menu Pull-Down */

#define MAXMENU        60 /* Numero maximo de menus definidos */
#define NICONEMAX      2
#define INI_FONTE      20 /*Codigo do primeiro carc. do fonte */
#define PIXEL_FISICO   8

```

Apêndice E

Listagem das Rotinas que Compõem o Ambiente Gráfico

- *Listagem das Rotinas Escritas em Assembly do Processador TMS34010*
- *Listagem das Rotinas Escritas em Linguagem "C"*
- *Listagem das Rotinas que Compõem o Subsistema de Comunicacao com o Hospedeiro*

Listagem das Rotinas Escritas em

Assembly do Processador TMS34010

As rotinas são colocadas por ordem alfabética de arquivos.

charset.asm

```
.width      132

*-----
*----- TMS34010 - LIE -----
*-----
* Estrutura de dados, que armazena o fonte (bit-map) dos caracteres
* ASCII.
* Esta estrutura contém os padrões de caracteres para fonte com
* espessamento constante. As dimensões dos caracteres são de 8 pixels
* de largura e 16 pixels de altura. Cada grupo de 16 bytes representa
* o "bit-map" para um caractere. Os padrões são listados em ordem
* através do código ASCII. No que no início da tabela são guardados
* alguns caracteres especiais (seta a direita (código 20)). Assim,
* do código 20 (definido como INIFOWR) ao 31 são reservados pelo
* sistema, e de 32 em diante são os ASCII.
*
*-----
* Historico:
* 11/25/86...Original version written.....Miha Denio
* 3/03/87...Modified for VGA-like appearance of font...Miha Denio
* 20/02/91...Ampliação dos códigos .....Agma J. M. Traina
*-----
;
; .title 'monospaced font'
; .file 'charset.asm'
;
;
; DECLARACAO DO ROTULO GLOBAL
;
; .globl _charset
;
;
; ESTRUTURA DE DADOS
;
_charset:
.byte >00,>00,>00,>02,>06,>0C,>1C,>3C ; Seta para Direita
;                                     (código 20)
.byte >7C,>3C,>1C,>0C,>06,>02,>00,>00
.byte >00,>00,>00,>00,>00,>00,>00,>00 ; código 21
.byte >00,>00,>00,>00,>00,>00,>00,>00
.byte >00,>00,>00,>00,>00,>00,>00,>00 ; código 22
.byte >00,>00,>00,>00,>00,>00,>00,>00
.byte >00,>00,>00,>00,>00,>00,>00,>00 ; código 23
.byte >00,>00,>00,>00,>00,>00,>00,>00
.byte >00,>00,>00,>00,>00,>00,>00,>00 ; código 24
```

```
.byte >00,>00,>00,>00,>00,>00,>00,>00
.byte >00,>00,>00,>00,>00,>00,>00,>00 ; código 25
.byte >00,>00,>00,>00,>00,>00,>00,>00
.byte >00,>00,>00,>00,>00,>00,>00,>00 ; código 26
.byte >00,>00,>00,>00,>00,>00,>00,>00
.byte >00,>00,>00,>00,>00,>00,>00,>00 ; código 27
.byte >00,>00,>00,>00,>00,>00,>00,>00
.byte >00,>00,>00,>00,>00,>00,>00,>00 ; código 28
.byte >00,>00,>00,>00,>00,>00,>00,>00
.byte >00,>00,>00,>00,>00,>00,>00,>00 ; código 29
.byte >00,>00,>00,>00,>00,>00,>00,>00
.byte >00,>00,>00,>00,>00,>00,>00,>00 ; código 30
.byte >00,>00,>00,>00,>00,>00,>00,>00
.byte >00,>00,>00,>00,>00,>00,>00,>00 ; código 31
.byte >00,>00,>00,>00,>00,>00,>00,>00
.byte >00,>00,>00,>00,>00,>00,>00,>00 ; Inicio dos Caracteres ASCII

.byte >00,>00,>00,>00,>00,>00,>00,>00 ; <SPACE>
.byte >00,>00,>18,>18,>18,>18,>18,>18 ; <!>, etc.
.byte >18,>18,>00,>18,>18,>00,>00,>00
.byte >00,>00,>66,>66,>66,>24,>00,>00
.byte >00,>00,>00,>00,>00,>00,>00,>00
.byte >00,>00,>00,>24,>24,>24,>7E,>24
.byte >7E,>24,>24,>24,>00,>00,>00,>00
.byte >00,>00,>00,>08,>1C,>2A,>0A,>1C
.byte >28,>2A,>1C,>08,>00,>00,>00,>00
.byte >00,>00,>7E,>42,>25,>25,>12,>10
.byte >08,>28,>54,>54,>20,>00,>00,>00
.byte >00,>00,>06,>09,>09,>06,>06,>09
.byte >11,>51,>21,>5E,>00,>00,>00,>00
.byte >00,>00,>18,>18,>18,>00,>00,>00
.byte >00,>00,>00,>00,>00,>00,>00,>00
.byte >00,>00,>30,>18,>0C,>06,>06,>06
.byte >06,>06,>0C,>18,>30,>00,>00,>00
.byte >00,>00,>0C,>18,>30,>60,>60,>60
.byte >60,>60,>30,>18,>0C,>00,>00,>00
.byte >00,>00,>00,>00,>00,>36,>1C,>7F
.byte >1C,>36,>00,>00,>00,>00,>00,>00
.byte >00,>00,>00,>00,>00,>18,>18,>7E
.byte >18,>18,>00,>00,>00,>00,>00,>00
.byte >00,>00,>00,>00,>00,>00,>00,>00
.byte >00,>00,>00,>18,>18,>10,>08,>00
.byte >00,>00,>00,>00,>00,>00,>00,>7E
.byte >00,>00,>00,>00,>00,>00,>00,>00
.byte >00,>00,>00,>00,>00,>00,>00,>00
.byte >00,>00,>00,>18,>18,>00,>00,>00
.byte >00,>00,>60,>60,>30,>30,>18,>18
.byte >0C,>0C,>06,>06,>00,>00,>00,>00
.byte >00,>00,>3E,>63,>63,>73,>73,>6B
.byte >67,>67,>63,>63,>3E,>00,>00,>00
.byte >00,>00,>18,>1C,>1E,>18,>18,>18
.byte >18,>18,>18,>18,>7E,>00,>00,>00
.byte >00,>00,>3E,>63,>60,>60,>60,>30
.byte >18,>0C,>06,>47,>7F,>00,>00,>00
.byte >00,>00,>3E,>63,>60,>60,>60,>38
.byte >60,>60,>60,>63,>3E,>00,>00,>00
.byte >00,>00,>30,>38,>3C,>3C,>36,>36
.byte >33,>7F,>30,>30,>78,>00,>00,>00
.byte >00,>00,>7F,>43,>03,>03,>3F,>60
.byte >60,>60,>60,>63,>3E,>00,>00,>00
.byte >00,>00,>3E,>63,>03,>03,>03,>3F
.byte >63,>63,>63,>63,>3E,>00,>00,>00
.byte >00,>00,>7F,>63,>60,>30,>30,>18
.byte >18,>18,>18,>18,>18,>00,>00,>00
.byte >00,>00,>3E,>63,>63,>63,>63,>3E
.byte >63,>63,>63,>63,>3E,>00,>00,>00
.byte >00,>00,>3E,>63,>63,>63,>63,>7E
.byte >60,>60,>60,>30,>1E,>00,>00,>00
.byte >00,>00,>00,>00,>00,>00,>18,>18
.byte >00,>00,>00,>18,>18,>00,>00,>00
```


color0.asm color1.asm

```

        .width 132
*-----
* TMS34010 LIT
*-----
* Funcao set_color0
*
* Atribui a COLOR0 o valor especificado. Dado um pixel de tamanho n
* (n bits), o valor deve ser replicado no registrador de 32 bits 32/n
* vezes. Por exemplo se é dado um pixel de tamanho 4, e o valor indicado
* é 6, o valor da entrada para a funcao deve ser 0x66666666.
*
* Esta funcao é projetada para ser chamada por um programa em C.
*
*-----
* Usa-se: set_color0(pval);
*
* Descricao dos argumentos:
*   int pval; /* valor do pixel replicado */
*
* Retorno no registrador A8: Void (indefinido).
*
* Registradores alterados: A8, B8
*-----
* Funcao set_color1
*
* Atribui a COLOR1 o valor especificado. Dado um pixel de tamanho n
* (n bits), o valor deve ser replicado no registrador de 32 bits 32/n
* vezes. Por exemplo se é dado um pixel de tamanho 4, e o valor indicado
* é 8, o valor da entrada para a funcao deve ser 0x88888888.
*
* Esta funcao é projetada para ser chamada por um programa em C.
*
*-----
* Usa-se: set_color1(pval);
*
* Descricao dos argumentos:
*   int pval; /* valor do pixel replicado */
*
* Retorno no registrador A8: Void (indefinido).
*
* Registradores alterados: A8, B8
*
*-----
* Revisao historica:
* 2/10/86...Original version written.....Gwyn Guidy
* 5/16/86...Added TI copyright notice.....Jerry Van Aken
* 7/03/90...Adicionado ao sistema LIT.....Agma Juci M. Traina
*-----
;
;   .title 'set color 0'
;   .file 'color0.asm'
;
;
;   DECLARACAO GLOBAL PARA NOME FUNCAO
;
;   .globl _set_color0,_set_color1
;
;   ENTRY POINT
;
;
;_set_color0:
;   MOVE    *-A14,A8,1
;   MOVE    A8,B8
;   RETS    2
;

```

```

;   DECLARACAO GLOBAL PARA NOME FUNCAO
;
;   ENTRY POINT
;
;_set_color1:
;   MOVE    *-A14,A8,1
;   MOVE    A8,B8
;   RETS    2
;   .end

```

comdata.asm

```

*-----
* TMS34010 - LIT
*-----
* Historico:
* 30-01-91 .....Agma Juci M. Traina
*-----
;*****
;* Declara secao de comunicacao com ambiente Host IBM-PC
;*****
;_comunica:
;   .sect "COM" ; Declara section COMMUNICATION
;   .globl _comunica
;   .word 0,0 ; Inicializa buffer de teclado
;   .word 0,0 ; Inicializa buffer de video
;   .word 0,0 ; Inicializa buffer de disco
;   .int 0,0 ;

```

control.asm

```

        .width 132
*-----
* TMS34010 - LIT
*-----
* Funcao set_control
*
* Estabelece uma mascara para o registrador CONTROL
*
* Usa-se: set_control(mval);
*
* Descricao dos argumentos:
*   int mval; /* mascara que sera colocada no reg. CONTROL */
*
* Retorno em A8: Void (indefinido).
*
* Registradores alterados: A8, CONTROL
*
*-----
* Funcao get_control
*
* Retorna o conteudo do registrador CONTROL
*
*-----
* Usa-se: get_control();
*
* Retorno em Reg. A8: Valor do CONTROL em 32-bit.
*
* Registradores alterados: A8
*
* Estas funcoes sao projetadas para serem chamadas por um Programa GSP_C.
*
*-----
* Data: 19-11-90
*-----

```

draw_box.asm

E - 7

DONE :

drawcar.asm

```

        .width    132
*-----
*-----  TM34010 - LIM  -----
*-----
* Funcao draw_car
*
* Esta funcao projetada para ser chamada por um programa em GSP-C
*-----
* Usa-se: draw_car(pc,x,y);
*
* Descricao dos argumentos:
* char *pc;      /* ponteiro para o caracter */
* int x,y;       /* posicao onde sera escrito o caracter */
*
* Retorno em A6: Void (indefinido)
*
* Registrador alterado: A6
*-----
* Historico:
* 06-12-90 ..... Agma J. M. Traina
*-----
;
        .title    'desenha caracter'
        .file      'draw_car.asm'
;
;
        DECLARAR NOME DA FUNCAO GLOBAL
;
        .globl    _draw_car
;
;
; O Ponteiro para o arquivo de fontes, e de seus atributos e buscado na
; estrutura lie_fonte_corrente.
;
        .globl    lie_fonte_corrente
;
;
        DEFINE CONSTANTS
        .nolist
        .copy gspregs.inc
        .list
;
;
        ENTRY POINT
;
_draw_car:
        MMTM      SP,A0,A1,A2,A4
        MMTM      SP,B0,B1,B2,B7,B10,B11,B12,B13,B14

        SETF      16,0,0
        SETF      32,0,1
        MOVE      *-STK,A0,1      ; Obtem o ponteiro para o caracter c
        MOVE      *-STK,A1,1      ; Obtem posicao de x
        MOVE      *-STK,A2,1      ; Obtem posicao de y

;
        Estabelece o "pitch"
        MOVE      @lie_fonte_corrente+32,A4,1
        MOVE      A4,SPTCH
        LMO       SPTCH,B0
        MOVE      B0,@CONVSP,0

;
        Estabelece DXDY

```

```

        MOVE      @lie_fonte_corrente+32,A4,1      ; A4 = tam_x
        MOVE      @lie_fonte_corrente+64,A5,1      ; A5 = tam_y
        SLL       16,A5
        MOVY      A5,A4      ; A4 = (y,x)
        MOVE      A4,DYDX

;
        Estabelece DADDR
        SLL       16,A2
        MOVY      A2,A1      ; A1 = y :: x
        MOVE      A1,DADDR

;
        O ponteiro do caracter esta em A0
        MOVE      A0,SADDR

        PIXBLT    B,XY      ; desenha o caracter

fim:
        MMFM      SP,B0,B1,B2,B7,B10,B11,B12,B13,B14
        MMFM      SP,A0,A1,A2,A4
        MOVE      SPTCH,A8      ; restore original CONVSP
        LMO       A8,A8      ;
        MOVE      A8,@CONVSP,0      ;
        RETS      2
        .end

```

fill.asm

```

        .width    132
*-----
*-----  TM34010 - LIM  -----
*-----
* fill function
*
* Esta funcao preenche um retangulo da tela, cuja especificacao e
* dada pelas variaveis "width" e "height". As coordenadas (xleft,yleft)
* especificam o canto superior esquerdo do retangulo. A cor que preenchera
* o retangulo deve ser previamente carregada no registrador COLOR1 (B9),
* antes de se chamar esta funcao.
*
* Esta funcao e projetada para ser chamada por um programa GSP-C.
*-----
* Usa-se: fill(width, height, xleft, ytop);
*
* Descricao dos argumentos:
* long width, height; /* largura e altura do retangulo */
* long xleft, ytop; /* coord's do canto superior esquerdo do ret. */
*
* Retorno em A6: Void (undefined)
*
* Registro alterado: A6
*-----
* Historico:
* 2/ 5/86...Original version written.....John Black
* 8/12/90...Adequacao ao sistema LIM .....Agma J. M. Traina
*-----
;
;
        DECLARAR NOME DA FUNCAO GLOBAL
;
        .globl    _fill
;
;
        ENTRY POINT
;
_fill:

```



```

        SETF      32,0,1          ;EU
        MOVE      A0,*-SP,1        ;salva registrador
        MMTM      SP,B2,B7,B10,B11,B12,B13,B14 ; Estes reg. sao alt. pela
FILL
        MOVE      *-A14,A0,1        ;Obtem 'width'
        MOVE      *-A14,A8,1        ;Obtem 'height'
        SLL       16,A8             ;
        MOVY      A8,A0             ;concatena width & height
        MOVE      A0,B7             ;Carrega DYDX
        MOVE      *-A14,A0,1        ;Obtem xleft
        MOVE      *-A14,A8,1        ;Obtem ytop
        SLL       16,A8             ;
        MOVY      A8,A0             ;concatena xleft & ytop
        MOVE      A0,B2             ;Carrega DADDR
        FILL      XY               ;"fill" o retangulo
        MMTM      SP,B2,B7,B10,B11,B12,B13,B14
        MOVE      *SP+,A0,1        ;recupera registrador
        RETS      2                ;
        .end

```

getpixel.asm

```

        .width    132
*-----
*          MM34010 - LIE
*-----
* Funcao get_pixel
*
* Retorna o valor do pixel nas coordenadas (xs,ys). As coordenadas
* sao especificadas relativamente ao canto superior esquerdo da tela.
* Se o tamanho do pixel e' n bits, o valor do pixel esta nos n LSBs
* do valor retornado; os 32-n MSBs sao 0s.
*
* Esta funcao e projetada para ser chamada por um programa em GWP-C.
*
* Usa-se: get_pixel(xs, ys);
*
* Descricao dos argumentos:
* int xs, ys; /* coordenadas do pixel fonte */
*
* Retorno em A8: Valor do Pixel
*
* Registrador alterado: A8
*
* Historico:
* 05/12/90.....Agma Jaci Machado Traina
*-----
;
; .title 'get pixel'
; .file 'getpixel.asm'
;
;
; DECLARAR NOME DA FUNCAO GLOBAL
;
; .globl _get_pixel
;
; ENTRY POINT
;
_get_pixel:
        MOVE      A0,*-SP,1        ;salva registrador
* Retira os argumentos da pilha.
        MOVE      *-A14,A0,1        ;Obtem 'xs'
        MOVE      *-A14,A8,1        ;Obtem 'ys'

```

```

* Calcula o endereco de memoria do pixel.
        SLL       16,A8             ;
        MOVX      A0,A8             ;concatena xs, ys
        CVXYL     A8,A8             ;Obtem endereco de memoria
                                   2 ; do pixel
* Le o pixel.
        PIXT      *A8,A8           ;
* Recupera os registradores e retorna.
        MOVE      *SP+,A0,1
        RETS      2                ;retorna
        .end

```

get_rect.asm

```

        .width    132
*-----
*          MM34010 - LIE
*-----
* Funcao get_rect
*
* Copia os pixels contidos em uma regioe retangular de tela,
* em um vetor de pixels especificado (copia da memoria de video para
* a memoria do sistema). O retangulo fonte e identificado por suas
* coordenadas do canto superior esquerdo, e sua altura e largura.
* o vetor destino e identificado por seu endereco base e seu "pitch".
* O "pitch" e a diferenca entre o endereco base de duas linhas
* adjacentes do vetor.
*
* Esta funcao e projetada para ser chamada por um programa em GWP-C.
*
* Usa-se: get_rect(width, height, srclft, srctop, dstarray, dstpitch);
*
* Descricao dos parametros:
* long width, height, /* largura e altura do retang. fonte */
* long srclft, srctop; /* coord. do canto sup. esq. do ret. fonte */
* long dstpitch; /* pitch do vetor destino */
* short dstarray[]; /* endereco base do vetor destino */
*
* Retorno em A8: Void (indefinido)
*
* Registro alterado: A8
*
* Historico:
* 2/ 5/86...Original version written.....John Black
* 10/20/86...Added TI copyright notice.....Stanley Anton
*-----
;
; DECLARAR NOME DA FUNCAO GLOBAL
;
; .globl _get_rect
;
;
; DEFINE CONSTANTES
;
; .nolist
; .copy gspregs.inc
; .list
;
; ENTRY POINT
;
_get_rect:
        MMTM      SP,A0,A1,A2,A3,A4,A5 ;Salva registradores A
        MMTM      SP,B0,B1,B2,B3,B7
        SETF      16,0,0

```

```

MOVE    *-A14,A0,1      ;Obtem WIDTH
MOVE    *-A14,A1,1      ;Obtem HEIGHT
MOVE    *-A14,A2,1      ;Obtem SRC LEFT
MOVE    *-A14,A3,1      ;Obtem SRC TOP
MOVE    *-A14,A4,1      ;Obtem DST pointer
MOVE    *-A14,A5,1      ;Obtem DST PITCH
MOVE    B3,B1           ;Estabelece SRC PITCH
MOVE    A5,B3           ;Estabelece novo DST PITCH
MOVE    @CONVSP,A8,0     ;Salva antigo CONVSP PITCH
MOVE    @CONVDP,@CONVSP,0
SLL     16,A3           ;Estabelece SOURCE ADDRESS formato XY
MOVY    A3,A2
MOVE    A2,B0
MOVE    A4,B2           ;Estabelece DEST ADDRESS em
                        ; formato LINEAR
                        ;Estabelece DYDX
SLL     16,A1
MOVY    A1,A0
MOVE    A0,B7           ;Estabelece DYDX
PIXBLT  XY,L           ;Move os Dados
MOVE    A8,@CONVSP,0
MMFM    SP,B0,B1,B2,B3,B7
MMFM    SP,A0,A1,A2,A3,A4,A5 ; Recupera os registradores A
RETS    2
.end

```

init_sdb.asm

```

.width 132
*-----
*----- RMS34010 - LIM -----
*-----
* Funcao _INIT_SDB
*
* Inicializa os registradores do GSP de: temporizacao da video,
*  refreshamento da video, e parametros graficos utilizados.
* FREQUENCIA HORIZONTAL DE 15750 Hz e
* FREQUENCIA DE PIXEL = 10MHz
*
* Esta funcao foi projetada para ser chamada por um programa GSP-C
*
*-----
* Usa-se: init_sdb()
*
* Na chamada:
* Nao sao passados argumentos para esta rotina.
*
* Na saida:
* Os registradores de I/O e os registradores B estao inicializados.
* Nenhum registrador A e alterado.
*
* Retorno em A8: void (indefinido)
*-----
* Historico:
* 10.01.90 ..... Stala e Agui
* 12.11.90 ..... Agui (adequacao dos IO reg.)
*-----
;
; .title 'Inicializa GSP'
; .file 'init_sdb.asm'
;
; Declaracao do nome da funcao GLOBAL
;
; .globl _init_sdb
;

```

```

; Define CONSTANTES
; --Copia gspregs.hdr, mas nao lista aqui.
; .nolist
; .copy gspregs.inc
; .list
;
; ENTRY POINT
;
_init_sdb:
* Inicializa registros de I/O.
SETF    16,0,0
MOVI    IOREGS1,B0      ;Aponta para valores de I/O
MOVI    HESYNC,B1       ;End. base para I/O register
MOVK    12,B2           ;Carrega contador de loop.
LOOP1:
MOVE    *B0+,*B1+,0
DSJS    B2,LOOP1
;
;
MOVI    IOREGS2,B0      ;Aponta para valores de I/O
MOVI    INTENB,B1       ;End. base para I/O register
MOVK    7,B2            ;Carrega contador de loop.
LOOP2:
MOVE    *B0+,*B1+,0
DSJS    B2,LOOP2
;
;
* Inicializa registros B.
MOVI    BREGS,B0
MMFM    B0,SPTCH,DPTCH,OFFSET,WSTART,WEND,COLOR0,COLOR1
;
* Terminado. Return.
* trap 29
RETS    0                ; 0, pois nao foi passado param. algum
;
* Define valores do reg. B-file.
BREGS:
.long    >FFFFFFF       ;COLOR1 = B9
.long    >00000000      ;COLOR0 = B8
.long    >00FF01FF      ;WEND = B6 depende do tamanho
                        ; logico do pixel
.long    >00000000      ;WSTART = B5
.long    >00000000      ;OFFSET = B4 EU, pois nao esta
                        ; com palleta
.long    >00001000      ;DPTCH = B3 512 pixels de
                        ; 8 bits (na tela!)
.long    >00000000      ;DADDR = B2
.long    >00000008      ;SPTCH = B1
;
* Define valores dos I/O register.
IOREGS1:
.word    >0007           ;HESYNC = >C0000000
.word    >001D           ;HEBLNK = >C0000010
.word    >0097           ;HSBLKN = >C0000020
.word    >009E           ;HTOTAL = >C0000030
;
.word    >0005           ;VESYNC = >C0000040
.word    >000a           ;VEBLNK = >C0000050
.word    >00fc           ;VSBLNK = >C0000060
.word    >0105           ;VTOTAL = >C0000070
;
.word    >F010           ;DPYCTL = >C0000080
.word    >FFFC           ;DPYSTRT = >C0000090
.word    >FFFF           ;DPYINT = >C00000A0
.word    >00C0           ;CONTROL = >C00000B0
;
IOREGS2:
.word    >0000           ;INTENB = >C0000110
.word    >0000           ;INTPEND = >C0000120
.word    >001C           ;CONVSP = >C0000130

```

```

.word    >0013      ;CONVDP  = >C0000140
.word    >0008      ;PSIZE   = >C0000150, tamanho
                        ;       logico do pixel.
.word    >0000      ;PMASKL  = >C0000160
.word    >0000      ;PMASKH  = >C0000170

.end

```

moverect.asm

```

.width    132
*-----
* TMS34010 - LIT
*-----
* Funcao move_rect
*
* Move os pixels no retangulo fonte especificado para o retangulo
* destino. A largura 'width' e a altura 'height' sao comuns aos dois
* retangulos. A posicao dos retangulos e' especificada pelas
* coordenadas da seu canto superior esquerdo.
*
* Esta funcao e projetada para ser chamada por um programa em GSW-C.
*
* Usa-se: move_rect(width, height, srclft, srctop, dstlft, dsttop)
* long width, height; /* Largura e Altura de ambos os retangulos*/
* long srclft, srctop; /* coord. do canto sup. esq. (ret. fonte) */
* long dstlft, dsttop; /* coord. do canto sup. esq. (ret. destino) */
*
* Retorne em A8: Void (indefinido)
*
* Registro alterado: A8
*-----
* Historico:
* 2/ 5/86...Original version written.....John Black
* 10/28/86...Added code for overlapping regions.....Stanley Acton
*-----
* Registradores Usados:
* A0 = WIDTH
* A1 = HEIGHT
* A2 = SRC LEFT
* A3 = SRC TOP
* A4 = DST LEFT
* A5 = DST TOP
*-----
*
* .globl _move_rect
*
* DEFINE CONSTANTES
* .nolist
* .copy qspregs.inc
* .list
*
*
* ENTRY POINT
*
*_move_rect:
    MM1M    SP,A0,A1,A2,A3,A4,A5,A6,A7; Salva os registradores
    MM1M    SP,B0,B1,B2,B7
    SETF    16,0,0
    MOVE    @CONTROL,A6      ;salva o valor do reg CONTROL atual
    MOVE    A6,A7           ; A7 atua como reg. CONTROL
    MOVE    *-A14,A0,1      ;Obtem WIDTH
    MOVE    *-A14,A1,1      ;Obtem HEIGHT
    MOVE    *-A14,A2,1      ;Obtem SRC LEFT
    MOVE    *-A14,A3,1      ;Obtem SRC TOP

```

```

MOVE    *-A14,A4,1      ;Obtem DST LEFT
MOVE    *-A14,A5,1      ;Obtem DST TOP
ANDI     >FCFF,A7      ;default e inc-x and inc-y
CMP      A2,A4          ; Verifica em que direcao
                        ; "to blit Horizontally"
JRLE     CHECK_Y        ;Verifica y
ORI      >0100,A7       ;Decrementa x

CHECK_Y:  CMP          A3,A5      ;Verifica em que direcao
                        ; "to blit Vertically"
JRLE     DEFOK          ;Default OK.
ORI      >0200,A7       ;Decrementa y

DEFOK:   MOVE    A7,@CONTROL      ;Armazena nova palavra de controle
MOVE     B3,B1      ;Estabelece "source pitch"
MOVE     @CONVSP,A8
MOVE     @CONVDP,@CONVSP
SLL      16,A3          ;Estabelece endereco fonte em
                        ; formato XY

MOVE     A3,A2
MOVE     A2,B0
SLL      16,A5          ;Estabelece endereco destino
                        ; em formato XY

MOVE     A5,A4
MOVE     A4,B2
SLL      16,A1          ;Concatena DYDX
MOVE     A1,A0
MOVE     A0,B7          ;Estabelece DYDX
PIXBLT   XY,XY          ;Move o dado
MOVE     A6,@CONTROL      ;Recupera palavra de controle original
MOVE     A8,@CONVSP
MMFM     SP,B0,B1,B2,B7
MMFM     SP,A0,A1,A2,A3,A4,A5,A6,A7 ; Recupera os registradores
RETS     2
.end

```

peekbreg.asm

```

.width    132
*-----
* TMS34010 - LIT
*-----
* funcao peek_breg
*
* Retorna o contendo do registrador de arquivo B especificado
* (32 bits). O registrador 'e especificado pelo seu numero, que
* pode ser de 0 a 15.
*
* Esta funcao e projetada para ser chamada por um programa em GSW-C.
*
* Usa-se: peek_breg(regno);
*
* Descricao dos argumentos:
* long regno; /* numero de registrador de arquivo B (0 a 15) */
*
* Retorne em A8: Valor do reg. B especificado (32-bit)
*
* Registrador alterado: A8
*
* Nota:
* -- Esta funcao utiliza apenas os 4 LSBs do numero do registrador,
* ignorando os demais
*-----
* Historico:
* 2/10/86...Versao Original .....Gwyn Guidy
*-----

```

```

;
; .title 'peek B-file reg'
; .file 'peekbreg.asm'
;
;
; DECLARAR NOME DA FUNCAO GLOBAL
;
; .globl _peek_breg
;
; ENTRY POINT
;
_peek_breg:
    MOVE    A0,*-SP,1      ; Salva A0
    MOVE    *-A14,A8,1     ; Obtem numero do register
    ANDI    15,A8          ; Separa apenas os 4 LSBs
    SLL     5,A8           ; Converte para o offset
    MOVI    SWITCH,A0      ; Obtem o endereco base
                                ; da tabela
    ADD     A8,A0          ; Base + offset
    JUMP    A0             ; Desvia para a posicao
                                ; correta na tabela

SWITCH:
    MOVE    B0,A8          ;
    JRUC    EXIT           ;
    MOVE    B1,A8          ;
    JRUC    EXIT           ;
    MOVE    B2,A8          ;
    JRUC    EXIT           ;
    MOVE    B3,A8          ;
    JRUC    EXIT           ;
    MOVE    B4,A8          ;
    JRUC    EXIT           ;
    MOVE    B5,A8          ;
    JRUC    EXIT           ;
    MOVE    B6,A8          ;
    JRUC    EXIT           ;
    MOVE    B7,A8          ;
    JRUC    EXIT           ;
    MOVE    B8,A8          ;
    JRUC    EXIT           ;
    MOVE    B9,A8          ;
    JRUC    EXIT           ;
    MOVE    B10,A8         ;
    JRUC    EXIT           ;
    MOVE    B11,A8         ;
    JRUC    EXIT           ;
    MOVE    B12,A8         ;
    JRUC    EXIT           ;
    MOVE    B13,A8         ;
    JRUC    EXIT           ;
    MOVE    B14,A8         ;
    JRUC    EXIT           ;
    MOVE    B15,A8         ;
    JRUC    EXIT           ;

EXIT:
    MOVE    *SP+,A0,1      ; Carrega A0
;
; trap 29
; RETS 2
; .end

```

pixel.asm

```

; .width 132
;-----TMS34010 - LTX-----
;

```

```

* Funcao set_pixel_size
*
* Estabelece um valor para o registrador PSIZE (1,2,4,8,16)
*
* Esta funcao e projetada para ser chamada por um programa GSP-C.
*-----
* Usa-se: set_pixel_size(pval);
*
* Descricao dos argumentos:
* int pval; /* valor do pinal (pode ser replicado se tam. pinal<16) */
*
* Retorno em A8: Void (Indefinido).
*
* Registradores alterados: A8, PSIZE
*-----
* Funcao get_pixel_size
*
* Retorna o contendo do registrador PSIZE.
*
* Esta funcao e projetada para ser chamada por um Programa GSP-C.
*-----
* Usa-se: get_pixel_size();
*
* Retorno em A8: Valor de PSIZE em 32-bit.
*
* Registrador alterado: A8
*-----
* Data: 01-10-90 .....Agua Joci M. Traima
*-----

;
; .title 'pixel'
; .file 'pixel.asm'
;
;
; DECLARE GLOBAL FUNCTION NAME
;
; .globl _set_pixel_size
; .globl _get_pixel_size
;
; .nolist
; .copy gspregs.inc
; .list
;
; ENTRY POINT
;
_set_pixel_size:
    SETF    32,0,1
    SETF    16,0,0
    MOVE    *-A14,A8,1      ;
    MOVE    A8,@PSIZE,0     ; Registrador PSIZE
    RETS    2
;
; ENTRY POINT
;
_get_pixel_size:
    SETF    16,0,0
    MOVE    @PSIZE,A8,0     ; A8 <- PSIZE
    RETS    0               ; Como nao ha parametros de entrada
                                ; o pc e inc. por 32
; .end

```

point.asm

```

        .width 132
*-----
*          TMS34010 - LIT
*-----
* Funcao point
*
*   Desenha um unico pixel nas coordenadas (x,y) da tela.
*   A cor do pixel esta no registrador COLOR1 (B9).
*
*   Esta funcao e projetada para ser chamada por um programa em GSW-C.
*-----
* Usa-se: point(x, y);
*
* Descricao dos argumentos:
*   long x, y; /* coord. do pixel */
*
* Retorna em A8: Void (indefinido)
*
* Registro alterado: A8
*-----
* Historico:
*   2/ 5/86...Original version written.....John Black
*   11/19/86...Modified header info, shortened code.....Jerry Van Aken
*-----
        .title 'draw a point'
        .file 'draw_pt.asm'
;
;
;   DECLARAR NOME DA FUNCAO GLOBAL
;
        .globl _point
;
;   ENTRY POINT
;
_point:
        MOVE    A0,*-SP,1      ;salva registro na pilha
        MOVE    *-A14,A0,1     ;Obtem 'x'
        MOVE    *-A14,A8,1     ;Obtem 'y'
        SLL     16,A8          ;
        MOVY    A8,A0          ;forma endereco xy address
        DRAV    A8,A0          ;desenha pixel na tela
        MOVE    *SP+,A0,1      ;recupera registro
        RETS     2              ;
        .end

```

pokebreg.asm

```

        .width 132
*-----
*          TMS34010 - LIT
*-----
* Funcao poke_breg
*
*   Armazena o valor especificado (32 bits) no registrador B
*   identificado pelo argumento regno. Os 4 LSBs de regno selecionam
*   um registrador B (0 a 15). Os outros MSBs de regno sao ignorados.
*
*   Este programa e projetado para ser chamado por um programa GSW-C.
*-----
* Usa-se: poke_breg(regno, valor);

```

```

*
* Descricao dos argumentos:
*   short regno; /* numero do registrador B (0 a 15) */
*   long valor; /* valor em 32-bit para ser armazenado no registrador */
*
* Retorna em A8: Void (indefinido).
*
* Registro alterado: A8
*-----
* Historico:
*   2/10/86...Original version written.....Gwyn Guidy
*   5/16/86...Added TI copyright notice.....Jerry Van Aken
*-----
;
;   .title 'poke B-file reg'
;   .file 'pokebreg.asm'
;
;
;   DECLARAR NOME DA FUNCAO GLOBAL
;
        .globl _poke_breg
;
;   ENTRY POINT
;
_poke_breg:
        MOVE    A0,*-SP,1      ;Salva A0
        MOVE    *-A14,A8,1     ;Obtem regno
        ANDI    15,A8          ;Mantem apenas os 4 LSBs
        SLL     5,A8           ;Converte regno para offset
        MOVI    SWITCH,A0      ;Chaveia para endereco
;
;   base da tabela
;   Base + offset
;   Obtem valor do argumento
;   Desvia para a tabela
SWITCH:
        MOVE    A8,B0          ;
        JRUC    EXIT           ;
        MOVE    A8,B1          ;
        JRUC    EXIT           ;
        MOVE    A8,B2          ;
        JRUC    EXIT           ;
        MOVE    A8,B3          ;
        JRUC    EXIT           ;
        MOVE    A8,B4          ;
        JRUC    EXIT           ;
        MOVE    A8,B5          ;
        JRUC    EXIT           ;
        MOVE    A8,B6          ;
        JRUC    EXIT           ;
        MOVE    A8,B7          ;
        JRUC    EXIT           ;
        MOVE    A8,B8          ;
        JRUC    EXIT           ;
        MOVE    A8,B9          ;
        JRUC    EXIT           ;
        MOVE    A8,B10         ;
        JRUC    EXIT           ;
        MOVE    A8,B11         ;
        JRUC    EXIT           ;
        MOVE    A8,B12         ;
        JRUC    EXIT           ;
        MOVE    A8,B13         ;
        JRUC    EXIT           ;
        MOVE    A8,B14         ;
        JRUC    EXIT           ;
        MOVE    A8,B15         ;
EXIT:
        MOVE    *SP+,A0,1      ;Recupera A0

```

putpixel.asm

```
.width 132
*-----
*----- TM34010 - LIE -----
*-----
* Funcao put_pixel
*
* Recupera o "valor" do pixel especificado nas coordenadas da tela
* (xd,yd). Esta posicao leva em conta a orientacao dos eixos
* previamente especificada.
* Se o tamanho do pixel e de n bits, o valor do pixel esta
* armazenado apenas os n LSBs, os outros 32-n MSBs sao ignorados.
*
* Esta funcao e projetada para ser chamada por um programa em GSP-C.
*-----
* Usa-se: put_pixel(valor, xd, yd);
*
* Descricao dos argumentos:
* int valor; /* valor do pixel */
* int xd, yd; /* coordenadas do pixel destino */
*
* Retorna em A6: Void (indefinido)
*
* Registro alterado: A6
*-----
* Historico:
* 05/12/90.....Agma Jaci M. Traina
*-----
;
; .title 'put pixel'
; .file 'putpixel.asm'
;
; DECLARAR NOME DA FUNCAO GLOBAL
;
; .globl _put_pixel
;
; ENTRY POINT
;
_put_pixel:
;
; MMTM SP,A0,A1 ;salva registradoress
* Retira argumentos da pilha.
; MOVE *-A14,A8,1 ;Obtem 'valor'
; MOVE *-A14,A0,1 ;Obtem 'xd'
; MOVE *-A14,A1,1 ;Obtem 'yd'
* Coloca o endereco em format XY.
; SLL 16,A1 ;
; MOVY A1,A0 ;
* Escreve o pixel.
; PIXT A8,*A0.XY ;
* Recupera os registradoress e retorna.
; MMTM SP,A0,A1 ;
; RETS 2 ;retorna
; .end
```

put_rect.asm

```
.width 132
*-----
*----- TM34010 - LIE -----
*-----
* Funcao put_rect
*
* Move o contendo do array de pixels especificado, para uma area
* retangular da tela. A largura (width) e a altura (height) do
* retangulo devem ser especificadas. O array fonte e' identificado
* por seu endereco base e pitch. O pitch e' a diferenca entre os
* enderecos iniciais de duas linhas adjacentes do array. A posicao
* destino e' fornecida em termos de coordenadas do seu canto
* superior esquerdo.
*
* Esta funcao e projetada para ser chamada por um programa em GSP-C.
*-----
* Usa-se: put_rect(srcarray, srcpitch, width, height, dstleft, dsttop)
* long width, height; /* largura e altura dos retang. */
* long dstleft, dsttop; /* coord do canto sup. esq. do destino */
* long srcpitch; /* pitch array fonte */
* short srcarray[]; /* endereco base do array fonte */
*
* Valor retornado em A6: Void (indefinido)
*
* Registros alterados: A6
*-----
* Historico:
* 2/ 5/86...Original version written.....John Black and Gwyn Guidy
* 10/20/86...Added TX copyright notice.....Stanley Aotom
*-----
;
; .globl _put_rect
;
; ENTRY POINT
;
_put_rect:
; MMTM SP,A0,A1,A2,A3,A4,A5 ;Salva od reg.
; MMTM SP,B0,B1,B7,B10,B11,B12,B13,B14
; SETF 16,0,0
; MOVE *-A14,A4,1 ;Obtem SRC pointer
; MOVE *-A14,A8,1 ;Obtem SRC pitch
; MOVE *-A14,A0,1 ;Obtem WIDTH
; MOVE *-A14,A1,1 ;Obtem HEIGHT
; MOVE *-A14,A2,1 ;Obtem DST LEFT
; MOVE *-A14,A3,1 ;Obtem DST TOP
; MOVE A8,B1 ;Estabelece novo SOURCE PITCH
; MOVE A4,B0 ;Estabelece SOURCE ADDRESS
; ; formato linear
; SLL 16,A3
; MOVY A3,A2
; MOVE A2,B2 ;Estabelece DEST ADDRESS formato XY
; SLL 16,A1 ;Concatena DYDX
; MOVY A1,A0
; MOVE A0,B7 ;Estabelece DYDX
; PIXBLT L,XY ;Move os dados
; MMTM SP,B0,B1,B7,B10,B11,B12,B13,B14
; MMTM SP,A0,A1,A2,A3,A4,A5 ;Recupera Regsittradores
; RETS 2
; .end
```

Listagem das Rotinas Escritas em Linguagem "C"

criajan.c

```
#include "data_str.h"
#include "lie_sgj.h"

extern lie_janela *lie_jan_corrente;

/*----- TMS34010 - LIE -----*/
/*
 * Arquivo: CriaJanela.c
 * Funcao: Cria_janela
 */
/*
 * Historico:
 * - Data de criacao: 24-01-91 ..... Agma Juci Machado Traina
 * - 03-02-91 Inclusao da opcao da sobreposicao Agma J. M. T.
 */
/*-----*/

/*
 * NOME:
 * Cria_janela
 */
/*
 * SINTAXE:
 * int Cria_janela (wid, parval)
 * int parval;
 * lie_janela **wid;
 */
/*
 * DESCRICAO:
 * Cria uma janela basica, como descendente da janela corrente.
 * Os parametros dessa janela sao assumidos como tendo os mesmos
 * valores da janela corrente, a menos daquelas que sao explicitamente
 * alteradas atraves de um par parametro-valor especificado. O
 * identificador da nova janela e' retornado como "wid".
 * A janela apenas e' criada, porem nao se torna corrente, e nem
 * e' mapeada na tela. Para que uma janela se torne corrente, usa-se
 * a rotina Jan_corrente(). Para mapear a janela, usa-se a rotina
 * Mostra_Janela().
 */
/*
 * RETORNO:
 * wid:
 * identificador da janela criada.
 */
/*
 * Valor de Retorno da Rotina:
 * Erro (vide ERROS:).
 */
/*
 * RESTRICOES:
 * - Os limites (x,y) da janela devem ser especificados em coordena-
 * das relativas 'a janela corrente.
 * - O cursor e' especificado em coordenadas relativas 'a janela
 * criada.
 * - Os limites (x,y) da janela criada devem estar inteiramente
 * contidos dentro da janela corrente, caso contrario a janela
 * sera truncada ("clipping") para esses limites.
 */
/*
 * ERROS:
 * 0 - Tudo OK. Janela criada corretamente.
 * 1 - Estrutura da janela indicada danificada.
 */
```

```
*
 * 2 - Memoria insuficiente para esta operacao.
 * 10 - Margens fora de limite. Janela criada mas truncada.
 * 11 - Parametro Invalido.
 */
/*
 * PARAMETROS TRATADOS
 * XLO XLO XHI XHI
 * FLANK MARK EVENT MARK
 * COR_FUENTE COR_FUNDO
 * X_CURSOR Y_CURSOR
 * CURSORON
 */
/*
 * VEJA TAMBM:
 * Jan_corrente(3), Mostra_janela(3)
 */
/*
 * EXEMPLO:
 * // Criar uma janela descendente diretamente da tela inteira: //
 * extern lie_janela *cria, raiz;
 * Jan_corrente (&raiz); // torna a raiz janela corrente //
 * Cria_janela (&cria,
 *             XLO, 20, YLO, 10,
 *             XHI, 110, XHI, 200,
 *             FIM);
 */
/*
 * NOTA
 * A lista de argumentos e colocada na pilha STK (A14 do TMS), como
 * essa pilha cresce para enderecos altos da memoria, ao retirar-se
 * os parametros da pilha, o ponteiro para os parametros deve ser
 * decrementado.
 */
/*-----*/

int Cria_janela (wid, parval)
int parval;
lie_janela **wid;
{
    union paramval pv;
    long largura;
    int aux, erro;
    lie_janela *criada;

    pv.valorint=&parval;

    /* Coloca a janela na estr. de dados, copiando todos os parametros
       presentes no pai para o filho. */
    erro = Cria_filho (&criada, lie_jan_corrente);
    if (erro != 0) return erro;
    *wid=criada;

    /* Tratar de cada parametro passado, individualmente */
    while (1) {
        if (*pv.parametro < MIN_PAR || *pv.parametro > MAX_PAR)
            return 11; /***** 11 = Parametro Invalido. *****/

        switch (*pv.parametro--) {

            case XLO:
                aux = *pv.valorint--;
                aux += lie_jan_corrente->xlo;
                if (aux < Criada->xlo) {
                    erro=10; /***** 10 = Margens fora de limite ****/
                    break;
                }
                if (aux > Criada->xhi) {
                    erro=10; /***** 10 = Margens fora de limite ****/
                    break;
                }
                Criada->xlo=aux;
                break;

            case YLO:

```

```

    aux = *pv.valorint--;
    aux += lie_jan_corrente->ylo;
    if (aux < criada->ylo) {
        erro=10; /***** 10 = Margens fora de limite ****/
        break;
    }
    if (aux > criada->yhi) {
        erro=10; /***** 10 = Margens fora de limite ****/
        break;
    }
    criada->ylo=aux;
    break;

case XHI:
    aux = *pv.valorint--;
    aux += lie_jan_corrente->xlo;
    if (aux > criada->xhi) {
        erro=10; /***** 10 = Margens fora de limite ****/
        break;
    }
    if (aux < criada->xlo) {
        erro=10; /***** 10 = Margens fora de limite ****/
        break;
    }
    criada->xhi=aux;
    break;

case YHI:
    aux = *pv.valorint--;
    aux += lie_jan_corrente->ylo; /* em relacao a janela pai */
    if (aux > criada->yhi) {
        erro=10; /***** 10 = Margens fora de limite ****/
        break;
    }
    if (aux < criada->ylo) {
        erro=10; /***** 10 = Margens fora de limite ****/
        break;
    }
    criada->yhi=aux;
    break;

case PLANE_MASK:
    criada->plane_mask = (short) (*pv.valorshort--);
    break;

case EVENT_MASK:
    criada->event_mask = (short) (*pv.valorshort--);
    break;

case COR_FRENTE:
    criada->cor_frente = *pv.valorint--;
    break;

case COR_FUNDO:
    criada->cor_fundo = *pv.valorint--;
    break;

case X_CURSOR:
    criada->x_cursor = *pv.valorint--;
    break;

case Y_CURSOR:
    criada->y_cursor = *pv.valorint--;
    break;

case SOBREPOE:
    if (*pv.valorint-- == NAO)
        criada->base = (int *)0;
    else {
        largura=criada->xhi - criada->xlo + 1+ 16;

```

```

    largura=(long)(largura / 16)*16; /* ajustar word boundary */
    criada->base=(int *)malloc(
        ( largura * (criada->yhi - criada->ylo +1) *
          get_pixel_size() ) / 8);
    if (criada->base == 0)
        erro=2; /***** Memoria insuficiente para a operacao ****/
    }
    break;

case FONTE:
    criada->arq_fonte = *pv.valorpont--;
    break;

case PASSA_INT:
    pv.valorint--;
    break;

case PASSA_SHORT:
    pv.valorshort--;
    break;

case PASSA_BYTE:
    pv.valorbyte--;
    break;

case PASSA_PONTEIRO:
    pv.valorpont--;
    break;

case FIM:
    return erro;

default: /* NOTA:
    * Trata-se de parametro que nao deve ser tratado
    * por esta rotina. Considera-se como padrao um
    * deslocamento de um valor inteiro. Note-se que
    * se o tipo de dado do valor ocupar um espaco de
    * memoria diferente do ocupado por um inteiro,
    * entao ira' ocorrer um deslocamento erroneo do
    * avanco do ponteiro de varredura de parametros,
    * causando uma perda dos parametros seguintes, com
    * resultados imprevisiveis!
    * Parametros nao tratados (a principio ja tratados
    * previamente por outras rotinas) devem ser
    * substituidos por PASSA_INT, PASSA_SHORT,
    * PASSA_BYTE, ou PASSA_PONTEIRO.
    * * * * *
    */

    pv.valorint--;
    break;

} /* Fim do Switch */
} /* Fim do While */
} /* Fim da rotina cria_janela */

```

criajانبorda.c

```

#include "data_str.h"
#include "lie_sgj.h"

extern lie_janela *lie_jan_corrente;

/*-----
*-----  TMS34010 - LIE  -----

```



```

*-----*
* Arquivo: CriaJamb.c
* Funcao Cria_jan_borda
*-----*
* Historico:
* - Data da criacao: 31-01-91 ..... Agma Juci Machado Traina
*-----*
*
* NOME:
*   Cria_jan_borda
*
* SINTAXE:
*   int Cria_jan_borda (wid, parval)
*   int parval;
*   lie_janela **wid;
*
* DESCRICAO:
*   Cria uma janela com borda, como descendente da janela corrente.
*   Os parametros dessa janela sao assumidos como tendo os mesmos
*   valores da janela corrente, a menos daqueles que sao explicitamente
*   alterados atraves de um par parametro-valor especificado. O
*   identificador da nova janela e' retornado como "wid".
*   A janela apenas e' criada, porem nao se torna corrente, e nem
*   e' mapeada na tela. Para que uma janela se torne corrente, usa-se
*   a rotina Jan_corrente(). Para mapear a janela, usa-se a rotina
*   Mostra_Janela().
*
* RETORNO:
*   wid:
*       identificador da janela criada.
*
*   Valor de Retorno da Rotina:
*       Erro (vide ERROS:).
*
* RESTRICOES:
*   - Os limites (x,y) da janela devem ser especificados em coorde-
*     nadas relativas 'a janela corrente.
*   - O cursor e' especificado em coordenadas relativas 'a janela
*     criada.
*   - Os limites (x,y) da janela criada devem estar inteiramente
*     contidos dentro da janela corrente, caso contrario a janela
*     sera truncada ("clipping") para esses limites.
*
* ERROS:
*   0 - Tudo OK. Janela criada corretamente.
*   1 - Estrutura da janela indicada danificada.
*   2 - Memoria insuficiente para esta operacao.
*   10 = Margens fora de limite. Janela criada mas truncada.
*   11 = Parametro Invalido.
*
* PARAMETROS TRATADOS
*   XLO YLO XHI YHI
*   FLAME MASK EVENT MASK
*   COR_FRENTE COR_FUNDO
*   X_CURSOR Y_CURSOR
*   COR_FRENTE_BORDA COR_FUNDO_BORDA
*   TITULO
*   SEM_BORDA COM_BORDA
*   ELEV_VERTICAL ELEV_HORIZONTAL
*   TRACO_SIMPLES TRACO_DUPLA
*   SOBREPONHA
*   BORDA_SINGELA
*
* VEJA TAMDEM:
*   Jan_corrente(3), Mostra_janela(3)
*
* EXEMPLO:
*   // Criar uma janela descendente diretamente da tela inteira: //
*   extern lie_janela *cria, raiz;

```

```

*   Jan_corrente (&raiz); // torna a raiz janela corrente //
*   Cria_jan_borda (&cria,
*                   XLO, 20, YLO, 10,
*                   XHI, 110, YHI, 200,
*                   FIM);
*
* NOTA
*   A lista de argumentos e colocada na pilha STK (Alô do TMS), como
*   essa pilha cresce para enderecos altos de memoria, se retirar-se
*   os parametros da pilha, o ponteiro para os parametros deve ser
*   decrementado (normalmente em outros compiladores - AT&T - seria
*   incrementado).
*-----*/
int Cria_jan_borda (wid, parval)
int parval;
lie_janela **wid;
{
    void *malloc();
    cadeia kvct();
    union paramval pv;
    long largura;
    int aux, erro, trata_par;
    int xlo_i, ylo_i, xhi_i, yhi_i, cor_frente_i, cor_fundo_i,
        x_cursor_i, y_cursor_i,
        tipo_borda;
    lie_janela *criada_e, *criada_i, *corrente;
    lie_tipo_janela *ptipo;

    pv.valorint=&parval;

    /* Coloca a janela externa na estr. de dados, copiando todos os
       parametros presentes no pai para o filho. */
    corrente= lie_jan_corrente;
    erro = Cria_filho (&criada_e, lie_jan_corrente);
    if (erro != 0) return erro;

    cor_frente_i=criada_e->cor_frente;
    cor_fundo_i=criada_e->cor_fundo;
    x_cursor_i=criada_e->x_cursor;
    y_cursor_i=criada_e->y_cursor;

    if ((ptipo = (lie_tipo_janela *)malloc (sizeof(lie_tipo_janela))) == 0)
    {
        erro = 2; /*** Nao ha espaco de memoria disponivel ***/
        return erro;
    }

    criada_e->tipo = ptipo;
    ptipo->tipo_borda = 0x05; /* Tipo da janela Default:
        0 0 0 1 0 1
        borda plena, aberta, sem elev h, com elev v, 1 traco, com borda */
    ptipo->z_janela=0; /* A ser tratado quando ativar SOBREPONCAO */
    ptipo->texto = (char *)0;

    /* Tratar de cada parametro passado, individualmente */
    trata_par=1;
    while (trata_par) {
        if (*pv.parametro < MIN_PAR || *pv.parametro > MAX_PAR) {
            erro = 11; /*** 11 = Parametro Invalido. ***/
            break;
        }
        switch (*pv.parametro--) {
            case XLO:
                aux = *pv.valorint--;
                aux += lie_jan_corrente->xlo;
                if (aux < Criada_e->xlo) {
                    erro=10; /*** 10 = Margens fora de limite ***/

```

```

        break;
    }
    if (aux > criada_e->xhi) {
        erro=10; /* 7**** 10 = Margens fora de limite ****/
        break;
    }
    criada_e->xlo=aux;
    break;

case YLO:
    aux = *pv.valorint--;
    aux += lie_jan_corrente->ylo;
    if (aux < criada_e->ylo) {
        erro=10; /* 7**** 10 = Margens fora de limite ****/
        break;
    }
    if (aux > criada_e->yhi) {
        erro=10; /* 7**** 10 = Margens fora de limite ****/
        break;
    }
    criada_e->ylo=aux;
    break;

case XHI:
    aux = *pv.valorint--;
    aux += lie_jan_corrente->xlo;
    if (aux > criada_e->xhi) {
        erro=10; /* 7**** 10 = Margens fora de limite ****/
        break;
    }
    if (aux < criada_e->xlo) {
        erro=10; /* 7**** 10 = Margens fora de limite ****/
        break;
    }
    criada_e->xhi=aux;
    break;

case YHI:
    aux = *pv.valorint--;
    aux += lie_jan_corrente->ylo; /* em relacao a janela pai */
    if (aux > criada_e->yhi) {
        erro=10; /* 7**** 10 = Margens fora de limite ****/
        break;
    }
    if (aux < criada_e->ylo) {
        erro=10; /* 7**** 10 = Margens fora de limite ****/
        break;
    }
    criada_e->yhi=aux;
    break;

case PLANE_MASK:
    criada_e->plane_mask = (short) (*pv.valorshort--);
    break;

case EVENT_MASK:
    criada_e->event_mask = (short) (*pv.valorshort--);
    break;

case COR_FRENTE:
    cor_frente_i = *pv.valorint--;
    break;

case COR_FUNDO:
    cor_fundo_i = *pv.valorint--;
    break;

case X_CURSOR:
    x_cursor_i = *pv.valorint--;
    break;

```

```

case Y_CURSOR:
    y_cursor_i = *pv.valorint--;
    break;

case FONTE:
    criada_e->arq_fonte = *pv.valorpont--;
    break;

case SOBREPOE:
    if (*pv.valorint-- == NAO)
        criada_e->base = (int *)0;
    else {
        largura=criada_e->xhi - criada_e->xlo + 1 + 16;
        largura=(long)(largura / 16)*16; /* ajustar word boundary */
        criada_e->base=(int *)malloc(
            (largura * (criada_e->yhi - criada_e->ylo + 1) *
             get_pixel_size()) / 8);
        if (criada_e->base == 0)
            erro=2; /* 7**** Memoria insuficiente para a operacao ****/
    }
    break;

case PASSA_INT:
    pv.valorint--;
    break;

case PASSA_SHORT:
    pv.valorshort--;
    break;

case PASSA_BYTE:
    pv.valorbyte--;
    break;

case PASSA_PONTEIRO:
    pv.valorpont--;
    break;

/* Tratamento de parametros especificos de janela com borda */
case COR_FRENTE_BORDA:
    criada_e->cor_frente = (int) (*pv.valorint--);
    break;

case COR_FUNDO_BORDA:
    criada_e->cor_fundo = (int) (*pv.valorint--);
    break;

case TITULO:
    ptipo->texto = (char *)kcvt((char *) (*pv.valorpont--));
    break;

case COM_BORDA:
    if (*pv.valorint-- == NAO)
        ptipo->tipo_borda = 0; /* Janela sem borda nao tem
                                mais nada */
    else
        ptipo->tipo_borda |= 1; /* Tem borda e e' com um traco */
    break;

case BORDA_SINGELA:
    if (*pv.valorint-- == NAO)
        ptipo->tipo_borda |= 0xDF; /* Janela com borda ampla */
    else
        ptipo->tipo_borda |= 0x21; /* Tem borda e e' com um traco */
    break;

case ELEV_VERTICAL:
    if (*pv.valorint-- != NAO)
        ptipo->tipo_borda |= 0x04;

```

```

        else ptipo->tipo_borda &= 0xFB;
        break;

case ELEV_HORIZONTAL:
    if (*pv.valorint-- != NAO)
        ptipo->tipo_borda |= 0x08;
    else ptipo->tipo_borda &= 0xF7;
    break;

case TRACO_DUPLA:
    if (*pv.valorint-- != NAO) {
        ptipo->tipo_borda &= 0xFD;
        ptipo->tipo_borda |= 0x01; /* Tem borda e e' com um traco */
    }
    else
        ptipo->tipo_borda |= 0x03; /* Tem borda e e' com 2 tracos */
    break;

case FIM:
    trata_par=0;
    break;

default: /* NOTA:
    * Trata-se de parametro que nao deve ser tratado
    * por esta rotina. Considera-se como padrao um
    * deslocamento de um valor inteiro. Note-se que
    * se o tipo de dado do valor ocupar um espaco de
    * memoria diferente do ocupado por um inteiro,
    * entao ira' ocorrer um deslocamento erroneo do
    * avancado do ponteiro de varredura de parametros,
    * causando uma perda dos parametros seguintes, com
    * resultados imprevisiveis!
    * Parametros nao tratados (a principio ja tratados
    * previamente por outras rotinas) devem ser
    * substituidos por PASSA_INT, PASSA_SHORT,
    * PASSA_BYTE, ou PASSA_PONTEIRO.
    * * * * *
    pv.valorint--;
    break;

    } /* Fim do Switch */
} /* Fim do While */

/* Criar a janela interna (do usuario) */
if (((criada_e->tipo)->tipo_borda & 0x20) & 0x20) {
    xlo_i=4;
    ylo_i=3;
    xhi_i=criada_e->xhi - criada_e->xlo - 3;
    yhi_i=criada_e->yhi-criada_e->ylo - 2;
}
else {
    xlo_i=8;
    ylo_i=20;
    xhi_i=criada_e->xhi-criada_e->xlo - 16;
    yhi_i=criada_e->yhi-criada_e->ylo - 3;
}

/* Cria a janela interna para o usuario. */
Jan_corrente (criada_e);
Cria_janela (&criada_i, XLO, xlo_i, YLO, ylo_i,
             XHI, xhi_i, YHI, yhi_i,
             COR_FRENTE, cor_frente_i, COR_FUNDO, cor_fundo_i,
             X_CURSOR, x_cursor_i, Y_CURSOR, y_cursor_i,
             SOBREPOE, NAO,
             FIM);

/* Restaura como corrente a janela que estava na chamada da rotina */
lie_jan_corrente = corrente;
Jan_corrente (corrente);

```

```

        *wid=criada_i;
        return erro;
    } /* Fim da rotina cria_janela */

```

des_pont.c

```

#include "data_str.h"
#include "lie_sgj.h"

extern lie_janela *lie_jan_corrente;

/*-----
*----- TMS34010 - LITE -----
*-----
* Arquivo: des_pont.c
* Funcoes: des_ponto (jan, x, y)
*-----
* Historico:
* - Data de criacao: 16-04-91 ..... Agma Juci Machado Traina
*-----
*-----
* NOME:
* des_ponto
*
* SINOPSE:
* int des_ponto (jan, x, y)
* int x, y;
* lie_janela *jan,
*
* DESCRICAO:
* Finta um pixel dentro da janela especificada - "jan", com
* a cor de fundo previamente estabelecida.
*
* RETORNO:
* Valor de Retorno da Rotina:
* Erro (vide ERROS:).
*
* RESTRICOES:
* - A janela deve ter sido definida corretamente, e sua estrutura
* deve estar integra.
*
* ERROS:
* 0 - Tudo OK.
* 1 - Estrutura da janela indicada danificada.
*
* VEJA TAMBEM:
* des_linha(3)
*-----*/

des_linha (jan, x, y)
int x, y;
lie_janela *jan,
{
    if (jan->sentinela != 1234 )
        return 1; /* Estrutura de janela deteriorada */

    jan->x_cursor = x;
    jan->y_cursor = y;
    return 0;
}

```

erase_sc.c

```

/*-----
*-----      TMS34010 - LIE      -----
*-----
* Funcao erase_screen
*
* Esta funcao e projetada par ser chamada por um C program
* Supoe-se que a tela possui 256 linhas por 512 pixels. Cada pixel
* com 8 bits (8 planes).
*-----
*
* NAME
*   erase_screen - Limpa toda a tela.
*
* SYNOPSIS
*   void erase_screen()
*
* DESCRICAO
*   Esta rotina coloca todos os pixels da tela com o valor 0x1515
*   (cinza claro). O tamanho da tela em pixels e' obtido pelas
*   constantes XMAX e YMAX definidas no arquivo <sysconst.h>.
*   Note-se que esta rotina apaga a tela independentemente de
*   existirem ou nao janelas abertas.
*   Se isso ocorrer, a atualizacao de informacoes em janelas
*   posteriores 'a chamada desta rotina ficam truncadas.
*   Portanto, esta rotina somente deve ser chamada quando nao
*   houverem janelas abertas.
*
* RESTRICOES
*   - Nao devem existir janelas abertas na tela.
*
*-----
* Historico:
*   - Data de criacao: 28-09-90      ..... Agma Juci Machado
*-----
*/

#include "sysconst.h"

void erase_screen()
{
    long c1, c2, c3;

    c1 = peek_breg(9);
    c2 = get_pixel_size(); /* Salva registradores */
    c3 = get_control();

    set_color1(0x1515); /* Estabelece cor de frente ( que FILL usa ) */
    set_pixel_size ((long)PIXEL_FISICO);
    set_control (0x0000); /* Desabilita transparencia e contr. de janela */
    fill (XMAX,YMAX,0,0); /* A tela possui YMAX linhas com XMAX pixels
cada */

    set_control (c3);
    set_pixel_size (c2); /* Re-estabelece os registradores */
    set_color1(c1);
    return;
}

```

formatk.c

```

/*-----
*-----
*
* AUTOR: Caetano Traina Junior
* DATA INICIAL: 12 de Outubro de 1986
* VERSAO ATUAL: 1.1
*
* DATA ULTIMA MODIFICACAO: 04/12/90
*
* ROTINAS:
*   char * fmtvrt(int *ap, int base, char *ap, int len)
*   _formatk (int (*sai)(),char *fmt, char *argp)
*
* MODIFICACOES:
*   Data: 12/10/86      Sobre a versao: Original Artec
*   Rotinas _formatk
*   Motivo: A rotina original do kit de distribuicao
*   Artec era a rotina _format. A mudanca consistiu
*   em substituir o tratamento de cadeias com '\0'
*   pelo tratamento com campos de controle. Alem disso
*   as seguintes mudancas foram efetuadas:
*   - O caracter usado para preencher um campo pode
*     ser especificado.
*   - Novos caracteres de formato foram
*     implementados:
*       a = cadeia (substituiu o s)
*       b = numero em binario
*       o = numero octal
*       t = tabula para essa posicao
*   - Forma designados sinonimos para os caracteres:
*       d = i
*       h = x
*       minusculas = minusculas
*
* #M1 Data: 04/12/90      Sobre a versao: 1.0
*   Rotinas _formatk
*   Motivo: Inclusao do formato k = imprima cadeia,
*           incluindo os bytes
*           de maximo e corrente.
*
* #M2 Data: 26/01/91      Sobre a versao: 1.0
*   Rotinas _formatk
*   Motivo: - corrigir impressao de mais de um formato
*           %k na mesma linha de formato.
*           - tratamento dos caracteres de escape "\c".
*           - inclusao no ambiente para o TMS34010
*           ..... Agma Juci M. Traina
*
*-----
*/

/*-----
*
* A rotina _formatk corresponde 'a rotina _format da
* printf, com as seguintes diferencas sobre o formato geral que e':
*   %0lac
*
* O caracter usado para preencher um campo pode agora ser
* especificado como o caracter que sucede ao "|". Por exemplo
* a sequencia "%|*5d" se tiver que escrever o valor "18" ira'
* escrever "***18". Alem disso, o caracter padrao 0 pode
* continuar a ser usado, por exemplo "%08d" corresponde a "%108d".
* Cadeia e' interpretada como consistindo de 2 bytes inicial
* se for usado o formato "%a". Nao existe mais o formato "%s"
* para vetor de caracteres terminados em "\0"
*
* Os seguintes caracteres foram implementados:
*
*   a = cadeia
*   b = numero em binario
*   c = caracter unico
*   d = decimal com sinal (wi)
*
*-----
*/

```

```
static char *fmtcvt(ap, base, cp, len)
int *ap; register char *cp;
int base, len;
{
    register unsigned long valor;
    static char digits[]="0123456789ABCDEF";
    if (len == sizeof(long)) valor = *(long *)ap;
    else if (base > 0) valor = *(unsigned *)ap;
    else valor = *ap;

    len = 0;
    if (base < 0) {
        base = -base;
        if ((long)valor < 0) { valor = -valor; len = 1; }
    }
    do *--cp = digits[(int)(valor%base)]; while (valor /= base);
    if (len) *--cp='-';
    return cp;
}
```

```
int (*sai)();
char *fmt;
char *argp;
{
register int c;
union {
        int *ip;
        char *cp;
        char **cpp;
#ifdef FLOAT
        double *dp;
#endif
    } args;
auto int ContCar = 0;
int JustEq, preenche;
int maxwidth, width;
int i, k;
char *cp;
int primeiro;
char *pfmt;
auto char s[34];
/* #M1 */
/* #M1 */
```

E - 35

E - 36

```

        maxwidth = maxwidth*10 + c - '0';
i = sizeof(int);
if ((c&0xDF) == 'L') {
    c = *fmt++;
    i = sizeof(long);
}

switch ( c&0xDF ) {
case 'O':
    k = 8;
    goto IntConvert;
case 'U':
    k = 10;
    goto IntConvert;
case 'X':
case 'H':
    k = 16;
    goto IntConvert;
case 'B':
    k=2;
    goto IntConvert;
case 'D':
case 'I':
    k = -10;

IntConvert:
    cp = _fmtcvt((int *)args.cp, k, s+32, i);
    if (!JustEsq) preenche=' ';
    args.cp -= i;
    break;

case 'A':
    cp = *args.cpp--;
    i = **cp;
    ++cp;
    goto havelen;

case 'K':
    JustEsq = 1;
    if (primeiro>=0) { /* impressao da cadeia
        width = primeiro;
        if (width<0) width=0;
        cp = *args.cpp--;
        i = **cp;
        ++cp;
        primeiro--;
        goto havelen;
    }
    else { /* impressao de [m:c]
        width=0;
        cp = *args.cpp;
        i=*cp++;
        k=*cp++;
        cp=s+32; *--cp=']';
        cp = _fmtcvt((int *)&k, -10, cp, 1);
        *--cp='[';
        cp = _fmtcvt((int *)&i, -10, cp, 1);
        *--cp='[';
        fmt=pfmt;
        i=primeiro+s+32-cp;
        goto havelen;
    }

case 'T':
    i=0;
    cp=s+32;
    width=width-ContCar;
    if (width<0) width=0;
    goto havelen;

```

```

#ifdef FLOAT
case 'E':
case 'F':
case 'G':
    ftoa(*args.dp--, s, maxwidth==10000?6:maxwidth, c-'e');
    i = strlen(cp = s);
    maxwidth = 200;
    goto havelen;
#endif

case 'C':
    c = *args.ip--;
default:
    *(cp = s+31) = c;
    break;
} /* Fim do switch */
i = (int)((s+32) - cp);

havelen:
    if ( i > maxwidth ) i = maxwidth;

    if ( JustEsq ) { /* Se o campo deve ser completado ... */
        if ((*cp == '-' || *cp == '+') && preenche == '0') {
            --width;
            if ((*sai)(*cp++) == -1) return -1;
        }
        for (; width-- > i; ++ContCar)
            if ((*sai)(preenche) == -1) return -1;
    }

    for ( k = 0 ; *cp && k < i ; ++k )
        if ((*sai)(*cp++) == -1) return -1;
    ContCar += k;

    if ( !JustEsq ) for (; width-- > i; ++ContCar)
        if ((*sai)(preenche) == -1) return -1;

    } /* Fim do if Formato */
    } /* Fim do While */
return ContCar;
}

```

i_discorot.c

```

#include "data_str.h"
#include "i_disco.h"

extern struct LIE_comunica comunica;

/*-----
* NOME:
*   i_arqinicio - Inicializa a busca por arquivos, posiciona no
*                 primeiro
*
* SINOPSE:
*   erro = i_arqinicio(specname,atributos,cabecalho);
*   int    erro          Código de erro retornado pelo DOS
*   cadeia specname     Cadeia contendo a especificação de
*                         procura
*   int    atributos     Atributos dos arquivos a serem procurados
*   ArqSpec *cabecalho   Cabecalho de arquivo encontrado
*
* DESCRIÇÃO:
*   A função "i_arqinicio" vasculha o diretório corrente ou o
*   diretório especificado pelo path em "specname" em busca da primeira
*   entrada que atende ao nome de arquivo indicado em "specname".

```



```

*
* RETORNO:
*   O valor de retorno e' um int, indicando:
*   -1: o arquivo nao pode ser aberto.
*   valor positivo: indica o indice do arquivo aberto.
*-----*/

i_open(nome, modo)
cadeia nome;
int modo;
{
    /* Espera host acabar operacao anterior */
    while (comunica.controle_disco != DADO_ACEITO);
    comunica.endereco_disco=(int)nome;
    comunica.dado_disco=(int)modo;
    comunica.controle_disco= ABREARQ;

    /* Espera host acabar operacao */
    while (comunica.controle_disco != DADO_ACEITO);
    return comunica.dado_disco;
}

/*-----*/
* FOME:
*   i_close -- Fecha o arquivo indicado no host IBM_PC.
*
* SINOPSE:
*   erro = i_close(file);
*   int file;          Indice do arquivo a ser fechado.
*   int erro;         Codigo da erro retornado pelo DOS.
*
* DESCRICAO:
*   A rotina i_close fecha o arquivo indicado no hospedeiro. O arquivo
*   e' indicado atraves da variavel file, obtida quando o arquivo foi
*   aberto atraves de uma chamada anterior da rotina i_open.
*
* RETORNO:
*   O valor de retorno e' um int, indicando
*   ERRO: Fechou com sucesso.
*   -1: Nao pode fechar o arquivo.
*-----*/

i_close(file)
int file;
{
    /* Espera host acabar operacao anterior */
    while (comunica.controle_disco != DADO_ACEITO);
    comunica.dado_disco=(int)file;
    comunica.controle_disco= FECHARQ;
    /* Espera host acabar operacao */
    while (comunica.controle_disco != DADO_ACEITO);
    return comunica.dado_disco;
}

/*-----*/
* FOME:
*   i_read -- Le dados do arquivo indicado no host IBM-PC.
*
* SINOPSE:
*   erro = i_read(file, buffer, num);
*   int erro;         Codigo da erro retornado pelo DOS
*   int file;         Indica do arquivo a ser lido.
*   unsigned char *buffer; endereco do buffer de leitura.
*   int num;           Numero de bytes a serem lidos.
*
* DESCRICAO:
*   A rotina i_read le dados do arquivo no buffer indicado. Os dados

```

```

* sao lidos desde a posicao corrente do arquivo, ate' o maximo de num
* bytes. Caso essa quantidade de bytes ultrapasse o final do arquivo,
* apenas os bytes restantes sao lidos. O numero de bytes efetivamente
* lidos retorna como o valor de retorno da rotina. Caso esse numero
* seja diferente de num, significa que o final do arquivo foi atingido.
* Apes uma leitura, a posicao corrente do arquivo passa a ser o primeiro
* byte seguindo ao $ltimo byte lido.
*
* RETORNO:
*   O valor de retorno e' um int, indicando
*   ERRO: Fim de arquivo.
*   Valor positivo: numero de bytes efetivamente lidos.
*-----*/

i_read(file, buffer, num)
int file;
unsigned char *buffer;
int num;
{
    /* Espera host acabar operacao anterior */
    while (comunica.controle_disco != DADO_ACEITO);
    comunica.dado_disco=file;
    comunica.endereco_disco=(int)buffer;
    comunica.auxiliar_disco=num;
    comunica.controle_disco= LEARQ;

    /* Espera host acabar operacao */
    while (comunica.controle_disco != DADO_ACEITO);
    return comunica.dado_disco;
}

/*-----*/
* FOME:
*   i_write -- Escreve dados no arquivo indicado no host IBM_PC.
*
* SINOPSE:
*   erro = i_write (file, buffer, num);
*   int erro;         Codigo da erro retornado pelo DOS
*   int file;         Indica do arquivo a ser escrito.
*   unsigned char *buffer; endereco do buffer de escrita.
*   int num;           Numero de bytes a serem escritos.
*
* DESCRICAO:
*   A rotina i_write escreve dados no arquivo a partir do buffer
*   indicado. Os dados sao escritos da posicao corrente do arquivo,
*   ate' o maximo de "num" bytes. Os dados escritos sobrepõem aquelas
*   ja' existentes no disco. Caso os dados ultrapassem o final anterior
*   do arquivo, este e' estendido. Caso essa quantidade de bytes esgote
*   o espaco disponivel do disco, apenas os bytes que cabem sao lidos.
*   O numero de bytes efetivamente escritos retorna como o valor de
*   retorno da rotina. Caso esse numero seja diferente de "num",
*   significa que nao foi possivel escrever todo o buffer indicado.
*   Apes uma escrita a posicao corrente do arquivo passa a ser o
*   primeiro byte seguindo ao ultimo byte escrito.
*
* RETORNO:
*   O valor de retorno e' um int, indicando
*   Valor positivo: numero de bytes efetivamente escritos.
*   -1: Houve erro da escrita.
*-----*/

i_write (file, buffer, num)
int file;
unsigned char *buffer;
int num;
{
    /* Espera host acabar operacao anterior */
    while (comunica.controle_disco != DADO_ACEITO);

```



```

comunica.dado_disco=file;
comunica.endereco_disco=(int)buffer;
comunica.auxiliar_disco=num;
comunica.controle_disco= ESCARQ;

/* Espera host acabar operacao */
while (comunica.controle_disco != DADO_ACEITO);
return comunica.dado_disco;
}

/*-----*/
* NOME:
* i_seek -- Posiciona ponteiro no arquivo indicado no host IBM_PC.
*
* SINOPSE:
* erro = i_seek (file, offset, como);
* int erro;          Codigo de erro retornado pelo DOS
* int file;          Indice de arquivo a ser escrito.
* long offset;       Numero de posicoes (bytes) a deslocar
*                      no ponteiro.
* int como;          Forma de deslocar o ponteiro.
*
* DESCRICAO:
* A rotina i_seek movimenta a posicao corrente de arquivo indicado
* o numero de bytes indicado em "num", de acordo com o parametro "como":
* 0 - num bytes a partir do inicio do arquivo;
* 1 - num (positivo ou negativo) bytes a partir da posicao corrente;
* 2 - num bytes voltando a partir do final do arquivo.
*
* Valor de Retorno da Rotina i_seek (tipo int):
* Valor positivo: posicao corrente do ponteiro.
* -1: pesquisa ilegal.
*
*-----*/
i_seek (file, offset, como)
int file;
long offset;
int como;
{
/* Espera host acabar operacao anterior */
while (comunica.controle_disco != DADO_ACEITO);
comunica.dado_disco=(int)file;
comunica.endereco_disco=(int)offset;
comunica.auxiliar_disco=como;
comunica.controle_disco= POSICIONARQ;

/* Espera host acabar operacao */
while (comunica.controle_disco != DADO_ACEITO);
return comunica.dado_disco;
}

```

i_getk.c

```

#include "data_struct.h"

short_inseresn=0;
extern lie_janela *lie_jan_corrente;
extern lie_fonte lie_fonte_corrente;

/*-----*/
* AUTOR: Castano Traina Junior
* DATA INICIAL: 18 de Setembro de 1987
* VERSAO ATUAL: 1.0
* ROTINAS:
* v_insfalg (short flag)

```

```

* i_getk (cadeia cad, short como)
* _vgetk (int (*rotina)(), cadeia cad, short como)
*                      (interna)
*
* MODIFICACOES:
* Data: 18/02/91      Sobre a versao: 1.0
* Rotina: Todas
* Motivo: Adaptacao para operar no DOS
* Autor: Agma Juci Machado Traina
*
*-----*/
/*-----*/
* v_insfalg - Estabelece condicao inicial para insercao/sobreposicao
* de caracteres.
*
* SINOPSE
* int flagv = v_insfalg(flag)
* register short flag, vflag;
*
* DESCRICAO
* A rotina v_insfalg estabelece a condicao inicial da chamada
* da rotina i_getk(3LINGUW). Se flag=0, a rotina sobrepoe os caracteres
* digitados, caso contrario os caracteres sao inseridos. Nota-se que
* o conteudo antigo da cadeia e' sempre descartado, independentemente
* de valor de flag. O valor de retorno desta rotina e' o valor de flag
* anterior. Nota-se que o uso da tecla <INS> durante a execucao da
* rotina i_getk(3LINGUW) altera o valor de flag, mesmo para outras
* chamadas da rotina i_getk(3LINGUW).
*
* RETORNO
* Valor de Retorno da Rotina (tipo int):
* Valor anterior de flag.
*
* NOTA
* Existe um flag do sistema que registra a condicao atual de
* sobreposicao ou insercao. Esse valor e' sobreposto pela chamada
* da rotina v_insfalg ou através do uso da tecla <INS> quando a rotina
* i_getk esta ativa.
*
* VEJA TAMBEM
* i_getk(3LINGUW)
*
*-----*/
int v_insfalg(flag)
register short flag;
{
register short velho;
velho= inseresn;
_inseresn=(flag!=0);
return (velho);
}

/*-----*/
* i_getk - Le uma cadeia de caracteres do teclado do hospedeiro.
*
* SINOPSE
* #include <data_str.h>
*
* int car = i_getk(cad, como)
* int como;
* cadeia cad;
* int car;
*
* DESCRICAO
* A rotina i_getk chama a rotina _vgetk() indicando que devem ser
* aceitos apenas os sinais de entrada do teclado, o que e' realizado
* pela rotina i_geta().

```



```

/* Empurra toda a cadeia uma posicao para a direita */
for (k=i++; k<=jmax; k++) {
    il=aux[k];
    v_wputc(aux[k]=c);
    c=il;
}
/* Verificar se com o empurrao nao aumentou uma linha */
v_lugc (&lin1, &col1);
if ( (lin1==_linpe) && (col1==0+1) && (i<jmax) )
    --lin;
/* Colocar o cursor no lugar */
v_posic(lin, col+1);
v_wputc(0);
++jmax;
}
else { /* Aqui corresponde ao modo sobrepos */
    v_wputc(aux[i++]=c); v_wputc(0);
    if (i>jmax) jmax=i;
}
}
else { /* se a tecla obtida for controle: */
    switch (c) {
        case 8: /* Back Space */
        case 203: /* Seta para a Esquerda */
            if (i==0) break;
            v_lugc (&lin, &col); /* obter onde ficara o cursor */
            if (--col<0) {
                col=_colpe;
                lin--;
            }
            v_posic(lin, col);
            --i;
            if (c==203) break; /* se seta para a esquerda e' so' */
            --jmax; /* Deslocar um caracter para a esquerda */
            for (k=i; k<jmax; k++) v_wputc(aux[k]=aux[k+1]);
            /* Coloca no final um caracter de fundo */
            v_wputc(0); v_wputc(' ');
            v_posic(lin, col);
            break;

        case 187: /* F1 - Coloca um caracter */
            if ( (i==jmax) && (jmax<*cad) ) ++jmax;
        case 205: /* Seta para a Direita */
            if (i== jmax) break;
            v_wputc(aux[i++]); v_wputc(0);
            break;

        case 211: /* Delete */
            if (i==jmax) break;
            v_lugc (&lin, &col); /* obter onde ficara o cursor */
            --jmax; /* Deslocar um caracter para a esquerda */
            for (k=i; k<jmax; k++) v_wputc(aux[k]=aux[k+1]);
            /* Coloca no final um caracter de fundo */
            v_wputc(0); v_wputc(' ');
            v_posic(lin, col);
            break;

        case 200: /* Seta Acima */
        case 208: /* Seta Abaixo */
        case 201: /* Page Up */
        case 209: /* Page Down */
            if (!como) break;

        case 10: /* Line Feed */
        case 13: /* return */
        case 3: /* Cntrl C */
            *cad=jmax;
            inseresn=0;
            lie_jan_corrente->yhi=auxyhi; /* Restaura yhi */
            return (c);
    }
}

```

```

case 199: /* HOME */
/* volta caracter a caracter, reposicionando o cursor */
for (--i; i>=0; i--) {
    v_lugc (&lin, &col);
    if (--col<0) {
        col=_colpe; lin--;
    }
    v_posic(lin, col);
    v_wputc(aux[i]);
    v_posic(lin, col);
}
i=0;
break;

case 189: /* F3 = restaura cadeia anterior */
if (*cad > jmax) jmax=*cad;
case 207: /* END */
for (k=i; k<jmax; k++) v_wputc(aux[k]);
v_wputc(0);
i=jmax;
inseresn=0; /* Indo ao fim, desativa modo insercao */
break;

case 210: /* Insera */
inseresn=!_inseresn;
break;
} /* fim do switch */
} /* fim do if */
} /* fim do while (que nunca termina por aqui!) */
/* fim da rotina i_getk() */
}

```

i_vidkey.c

```

#include "data_str.h"
#include "lie_sgj.h"

extern struct LIE_comunica comunica;

/*****
 *
 *      filename: i_vidkey.c
 *
 *      Este arquivo contem as rotinas de comunicacao, para o teclado e o
 *      entre o IBM e o Host IBM-PC.
 *
 *-----
 *
 *      Historico:
 *      - Data da criacao: 11-02-91      ..... Agma J. M. Traina
 *****/

/*****
 *
 *      i_getc - Obtem caracter do teclado do hospedeiro.
 *
 *      SINCOPSE
 *      int caria = i_getc()
 *      unsigned char carin;
 *
 *-----
 *
 *      *DESCRICAO
 *      A rotina i_getc le um caracter do teclado do hospedeiro IBM-pc.
 *      Caso nao exista um caracter digitado, espera-se ate' que um
 *      caracter seja teclado. O valor de retorno da rotina e' o valor
 *      ASCII do caracter digitado.
 *
 *      *RETORNO
 *      Valor de Retorno da Rotina (tipo int):
 */

```

```

*      Valor ASCII do caracter lido/escrito.
*
* RESTRICOES
* - Podem ser digitados quaisquer dos valores ASCII, que os mesmos
* serao recebidos pela rotina i_getc. No entanto, os valores ASCII que
* correspondam 'a movimentos do cursor serao tratados como tal pela
* rotina i_putc.
*
* VEJA TAMBEEM
* i_poll(3LIBGSP), i_putc(3LIBGSP)
*-----*/

i_getc()
{
    short caracter;
    while (comunica.controle_teclado != DADO_PRONTO); /* Espera host liberar
o dado */
    caracter = comunica.dado_teclado;
    comunica.controle_teclado=DADO_ACEITO;
    return (int) caracter;
}

/*-----
*      i_pollc - Verifica se h caracter teclado.
*
* SINOSE
*      int flag = i_pollc()
*
* DESCRICAO
* A rotina i_pollc verifica se foi digitado um caracter no teclado
* do Hospedeiro IBM-PC. Caso nao exista um caracter digitado, a
* rotina retorna com o valor ZERO (0). Caso exista um caracter
* teclado, o valor de retorno e' o valor ASCII do caracter digitado,
* porem o caracter nao e' retirado do buffer.
*
* RETORNO
*      Valor de Retorno da Rotina (tipo int):
*      0 - nao existe caracter digitado.
*      >0 - Valor ASCII do caracter lido/escrito.
*
* RESTRICOES
* - Apesar da rotina retornar o codigo do caracter digitado, este
* nao e' retirado do buffer, de maneira que chamadas sucessivas desta
* rotina sempre retornar o mesmo caracter. Para retirar o caracter, e'
* necessario que seja chamada a rotina i_getc(3LIBGSP).
*
* VEJA TAMBEEM
*      i_getc(3LIBGSP).
*-----*/

i_pollc()
{
    if (comunica.controle_teclado == DADO_PRONTO)
        return (int)comunica.dado_teclado;
    return (int)0;
}

/*-----
*      i_putc - Escreve um caracter no video do hospedeiro.
*
* SINOSE
*      i_putc(caract)
*      unsigned char caract;
*
* DESCRICAO

```

```

* A rotina i_putc escreve o caracter "caract" no video do hospedeiro
* IBM-PC. Caso ainda exista um caracter anteriormente enviado ainda
* nao recebido pelo hospedeiro, espera-se ate' que o caracter
* anterior seja escrito. O valor de retorno da rotina e' o valor
* ASCII do caracter impresso.
*
* RETORNO
*      Valor de Retorno da Rotina (tipo int):
*      Valor ASCII do caracter lido/escrito.
*
* RESTRICOES
* - Podem ser digitados quaisquer dos valores ASCII, que os mesmos
* serao recebidos pela rotina i_getc. No entanto, os valores ASCII
* que correspondam 'a movimentos do cursor serao tratados como tal
* pela rotina i_putc.
*
* VEJA TAMBEEM
*      i_getc(3LIBGSP)
*-----*/

i_putc(caract)
unsigned char caract;
{
    /* Espera que nao haja caracter anterior nao aceito */
    while (comunica.controle_video != DADO_ACEITO);

    comunica.dado_video=(short)caract;
    comunica.controle_video=DADO_PRONTO;
    return (int) caract;
}

/*-----
*      i_printk - Saida formatada de dados no video do hospedeiro.
*
* SINOSE
*      #include <data_str.h>
*
*      void i_printk (fmt, args)
*      char *fmt;
*      char args;
*
* DESCRICAO
* A rotina i_printk envia caracteres para o video do hospedeiro
* IBM-PC, usando o esquema de formatacao da mesma maneira que a rotina
* printf(3).
*
* VEJA TAMBEEM
*      i_putc(3LIBGSP), v_wpus(3LIBGSP).
*-----*/

int i_printk (fmt, args)
char *fmt;
char args;
{
    extern int _iputc();
    return _formatk(_iputc, fmt, &args);
}

/*-----
* Rotina interna usada para formatacao. Coloca no video um
* caracter pela rotina pchar().
*/
_iputc(c)
unsigned char c;
{

```

```

i_putc(c);
if ( c == '\n') i_putc(13);
}

```

inicializa.c

```

/*****
 *
 * Este arquivo contem a rotina de inicializacao do sistema.
 *
 * Esta rotina utiliza as rotinas Cria_arvore e init_sdb.
 * A Rotina Cria_arvore(), inicializa as estruturas
 * de dados do sistema, abrindo a janela raiz como sendo
 * a tela.
 * A Rotina init_sdb(), inicializa os registradores
 * do processador TMS34010.
 *****/
-----
 *
 * Historico:
 * - Data de criacao: 20-11-90 ..... Agma J. M. Traina
 *****/
 * NOME
 * Inicializa - Prepara o Sistema para iniciar operacoes.
 *
 * SINOPSE
 * Inicializa()
 *
 * DESCRICAO
 * Esta rotina inicializa o ambiente de execucao do Sistema
 * de Gerenciamento de Janelas para aceitar solicitacoes de
 * operacoes. Esta deve ser a primeira rotina do sistema a ser
 * chamada, pois caso contr rio as operacoes nao serao adequadamente
 * atendidas. A tela de video e' apagada, e o TMS e' inicializada.
 *
 * RETORNO
 * Valor de Retorno da Rotina (tipo int):
 * Indica se houve erro (vide ERROS:).
 *
 * RESTRICOES
 * - Esta deve ser a primeira rotina do SGJ a ser chamada.
 *
 * ERROS
 * 0 - Tudo OK.
 * 1 - Erro na inicializacao. Sistema nao est em operacao.
 *****/
-----
#include "data_str.h"
#include "sysconst.h"

/*-----
-----
VARIAVEIS GLOBAIS
-----*/
lie_janela raiz=0; /* Definicao da raiz da estrutura de janelas */
extern byte charset[96][16]; /* charset deve ser associado ao arquivo de
fontes char_set */

lie_fonte lie_fonte_corrente = ( (byte *)charset, 8,16);
lie_janela *lie_jan_corrente = &raiz;
extern struct LIE_comunica comunica;

struct LIE_prof_janela prof_janela[MAXJANELAS]=0; /* Pode-se ter
ate MAXJANELAS definidas simultaneamente. */

```

```

/*-----
-----
Definicao de Icones do Sistema
-----*/
short int LIE_icone[2*16]={0x7FFF, 0x6003, 0x5005, 0x4809, /* Delir a */
0x4411, 0x4221, 0x4141, 0x4081, /* janela */
0x4141, 0x4221, 0x4411, 0x4809,
0x5005, 0x6003, 0x7FFF, 0x0000,

0x7FFF, 0x6003, 0x5005, 0x4809, /* Encolhe */
0x4411, 0x43E1, 0x4221, 0x4221, /* a janela */
0x4221, 0x43E1, 0x4411, 0x4809,
0x5005, 0x6003, 0x7FFF, 0x0000};

lie_fonte lie_icones=( (byte *)LIE_icone, 16,16);

/*-----*/
Inicializa()
{
long ps;
init_sdb();
comunica.controle_video=0;
ps=get_pixel_size(); /* Obtem o tamanho do pixel logico */
Cria_arvore (YMAX, ((2*PIXEL_FISICO)/ps * XMAX)/2); /* ajuste para
ququer hardware */
}

```

jancorrente.c

```

#include "data_str.h"
#include "lie_sgj.h"

extern lie_janela *lie_jan_corrente;

/*-----
-----
TMS34010 - LIE
-----*/
 *
 * Arquivo: JanCorrente.c
 * Funcoes: int Jan_corrente (wid)
 * int Mostra_janela ();
 *
 * Esta funcao e projetada par ser chamada por um GSP-C program
 *
 * Historico:
 * - Data de criacao: 25-01-91 ..... Agma Juci Machado Traina
 *****/
-----
 * NOME:
 * Jan_corrente
 *
 * SINOPSE:
 * int Jan_corrente (wid)
 * lie_janela *wid;
 *
 * DESCRICAO:
 * Torna corrente a janela indicada.
 *
 * RETORNO:
 * Valor de Retorno da Rotina:
 * Erro (vide ERROS:).
 *
 * RESTRICOES:

```

```

* - A janela deve ter sido definida corretamente, e sua estrutura
*   deve estar integra.
*
* ERROS:
* 0 - Tudo OK. Janela indicada e' a corrente.
* 1 - Estrutura da janela indicada danificada.
*
* VEA TAMBEM:
* cria_janela(SLINGS)
*
* EXEMPLO:
* // Tomar corrente a propria tela inteira: //
* antesm lie_janela raiz;
* Jan_corrente (raiz);
*-----*/

int Jan_corrente (wid)
lie_janela *wid;
{
    if (wid->sentinela != 1234) /* testa consistencia! */
        return 1; /***** Estrutura da janela inconsistente ****/
    lie_jan_corrente=wid;
    Fixa_jan_fisica (lie_jan_corrente);
    return 0;
} /* Fim da rotina Jan_corrente */

```

limpajan.c

```

/*-----
*----- TMS34010 - LIE
*-----
* Funcao Limpa_jan_corrente
*
* Esta funcao e projetada par ser chamada por um GNU-C program
*
*-----
* Limpa_jan_corrente - Limpa a Janela Corrente.
*
* SINOPSE
* void Limpa_jan_corrente();
*
* DESCRICAO
* Esta rotina preenche toda a janela corrente com o pixel de cor
* de fundo definido para a janela.
*
* RESTRICOES
* - Deve existir uma janela corrente consistente, e que esteja
*   mapeada no video.
*
* VEA TAMBEM
* Jan_corrente(SLINGS), Mostra_janela(SLINGS), Cria_jan_borda(SLINGS),
* Cria_janela(SLINGS).
*
* EXEMPLO
* lie_janela *janl;
*
* Cria_janela (&janl, /* Criar uma janela subordinada 'a corrente */
*             XLO, 20, YLO, 10,
*             XHI, 110, YHI, 200,
*             COR_FUNDO, 0, COR_FRENTE, 0xfff,
*             FIM);
* Jan_corrente(janl); /* Tomar corrente e limpar a janela
*                       cujo identificador e' janl */
* Mostra_janela();
* Limpa_jan_corrente();

```

```

* Mostra_janela();
* Limpa_jan_corrente();
*
*-----
* Historico:
* - Data de criacao: 24-01-91 ..... Agnes Juci Machado
*-----*/

#include "data_str.h"

extern lie_janela *lie_jan_corrente;

void Limpa_jan_corrente()
{
    long cl;

    cl = peek_breg(9); /* Salva cor de frente */

    /* Estabelece cor de frente ( que FILL usa ) */
    set_color1((long) lie_jan_corrente->cor_fundo);

    fill (lie_jan_corrente->xhi - lie_jan_corrente->xlo + 1,
          lie_jan_corrente->yhi - lie_jan_corrente->ylo + 1,
          lie_jan_corrente->xlo, lie_jan_corrente->ylo);

    set_color1(cl);
    return;
}

```

kvvt.c

```

typedef unsigned char *cadeia;

/*-----
*-----
*-----
* AUTOR: Caetano Traina Junior
* DATA INICIAL: 5 de outubro de 1989
* VERSAO ATUAL: 1.0
* NOTAS:
* kvvt (cadeia)
*
* MODIFICACOES:
* Data: --/--/-- Sobre a versao: 1.0
* Rotina: kvvt
* Motivo: Inclusao no Ambiente grafico para o TMS34010*
* .....Agnes Juci M. Traina
*
*-----
*/

/*-----
* NOME:
* kvvt -- Converte para cadeia.
*
* SINOPSE:
* cadeia kvvt (cad)
* cadeia cad;
*
* DESCRICAO:
* Esta rotina determina o numero de caracteres que existe em
* um vetor de caracteres desde a posicao da indice 2 ate' a
* ocorrencia do primeiro caracter NULL. Para isso e' necessario que
* as posicoes de indice 0 e 1 contenham respectivamente os
* caracteres '-' e '.', os quais sao substituidos pela rotina pelo
* numero de caracteres determinado. Dessa maneira o vetor passa a
* representar uma cadeia no padrao CMMSCCO.
* A finalidade desta rotina e' converter para o formato padrao

```

```

* OCEMCO um vetor inicializado como uma constante string pelo
* compilador.
* Note-se que os dois primeiros caracteres devem ser '-' e
* '.', e s'ao mudados para o tamanho maximo e corrente da cadeia, e
* que faz com que apenas a primeira chamada da rotina para um mesmo
* vetor seja efetiva. Chamadas consecutivas nao concentramos o
* caracteres '-' e '.' e s'ao rejeitadas, deixando a cadeia
* inalterada. Por outro lado, se a cadeia for modificada
* posteriormente, o valor da string original inicialmente
* inicializada pelo compilador passará a conter o valor modificado.
* Se a cadeia deve ser alterada e posteriormente reiniciada com o
* mesmo valor inicial, use a rotina "hdef".
*
* EXEMPLO:
* Em ambos os exemplos abaixo a cadeia "cad" passa a representar
* uma cadeia de tamanho maximo 7, tamanho corrente 6, contendo o
* valor "cadeia":
* Exemplo 1:
*   char cad[]="-..cadeia";
*   ...
*   kvvt(cad);
*
* Exemplo 2: char *cad;
*   ...
*   cad=kvvt("-..cadeia");
*
* VEA TAMBEM:
*   hdef();
*-----*/
cadeia kvvt (cad)
cadeia cad;
{
register int i;
register cadeia cp;

if (cad[0] == '-' && cad[1] == '.') {
i=0; cp=&(cad[2]);
while (*cp++) i++;
if (i<128) cad[0] = (cad[1] - i) + 1;
}
return cad;
}

```

le_imagem.c

```

#include "data_str.h"
#include "le_sgj.h"
#include "i_disco.h"

/*----- Variaveis Globais -----*/
extern lie_janela *lie_jan_corrente, raiz;
extern struct LIE_comunica comunica;

/*-----*/
* NOTAS:
*   i_le_imagem -- Le um arquivo de imagem no host, e transfere para
*   o TMS, numa janela.
*
* SINOPSE:
*   #include <data_str.h>
*
*   int erro = i_le_imagem (file, xlo, ylo, titulo, *alin, *ncol);
*
*   int erro;          Valor de retorno da rotina.
*   int file;          Arquivo de onde ser lida a imagem.
*   int xlo, ylo       Posicao relativa 'a janela corrente, do

```

```

*   canto superior esquerdo da janela a ser
*   aberta para mostrar a imagem.
*   int *alin, *ncol;   Tamanho da janela criada pela rotina
*                       para armazenar a imagem.
*   cadeia titulo;      Cabeçalho a ser colocado na janela.
*
* DESCRICAO:
*   Esta rotina le do arquivo indicado (que deve ser previamente
*   aberto com a rotina i_open(3LIEG69)) uma imagem no formato de imagem,
*   obtendo o numero de linhas (alin) e colunas (ncol), e abrindo uma
*   janela filha da janela corrente, nas coordenadas relativas 'a essa
*   janela correspondente 'a xlo e ylo. A imagem e' lida e transferida
*   diretamente para essa janela. Note-se que a janela corrente deve ter
*   tamanho suficiente para mostrar a imagem a partir da coordenada
*   indicada, caso contr rio a imagem ser truncada. Caso a cadeia titulo
*   nao seja nula, seu contendo e' mostrado no cabeçalho da janela criada.
*   A janela criada retorna da rotina como a janela corrente.
*
* RETORNO:
*   0 valor de retorno e' um int, indicando:
*   -1: o arquivo nao pode ser aberto.
*   valor positivo: indica o indice do arquivo aberto.
*
* RESTRICOES
*   - A janela corrente deve ter tamanho suficiente para mostrar a
*   imagem a partir da coordenada indicada, caso contr rio a imagem
*   ser truncada.
*   - O arquivo indicado deve ter uma imagem armazenada segundo o
*   padrao da imagem.
*
* ERROS
*   1 - o arquivo indicado nao esta acessivel,
*       ou nao esta aberto.
*   retorno>0 - codigo de arquivo indicado para leitura da imagem.
*
* VEA TAMBEM
*   Jan_corrente(3LIEGJ), i_open(3LIEG69).
*
* EXEMPLO
*   /* Mostrar a imagem armazenada no arquivo "minha.img" numa
*   janela criada diretamente sobre a tela. */
*   #include <data_str>
*   extern lie_janela raiz;
*   int file, nl, nc;
*
*   file=i_open(kvvt("-.minha.img",0);
*   Jan_corrente (&raiz);
*   i_le_imagem (file, 50, 40, &nl, &nc,
*   kvvt("-.Minha Imagem" ));
*
*-----*/

```

```

i_le_imagem (file, xlo, ylo, titulo, nlin, ncol)
int file, xlo, ylo, *nlin, *ncol;
cadeia titulo;
{
int x,y, xhi,yhi, aux, janl, erro;

/* Espera host acabar operacao anterior */
while (comunica.controle_disco != DADO_ACEITO);
comunica.dado_disco=file;
comunica.controle_disco= LEIMAGEM;

/* Espera host acabar operacao */
while (comunica.controle_disco != DADO_ACEITO);
aux=comunica.endereco_disco;

*ncol=(0x0000FFFF & aux);
*nlin=((0xFFFF0000 & aux) >> 16);
xhi=xlo + *ncol;

```

```

yhi=ylo + *nlin;
comunica.controle_disco= CONTINUA;

/* v_printk("janela aberta em [%d,%d] [%d,%d]\n",
   xlo-8, ylo-20, xhi+16, yhi+3); */
erro= Cria_jan_borda (&jan1, XLO, xlo-8, YLO, ylo-20,
   XHI, xhi+16, YHI, yhi+3,
   COR FUNDO, 0xFFFF, COR FRENTE, 0,
   SOBREP OE, NAO, /* Por enq. nao tem RAM suficiente */
   COR FRENTE BORDA, 0xAAAA, COR FUNDO_BORDA, 0x5555,
   TITULO, titulo,
   TRACO_DUPLO, NAO,
   FIM);

Jan_corrente (jan1);
Mostra_janela();

for (y=ylo; y < yhi; y++)
  for (x=xlo; x < xhi; x++) {
    while (comunica.controle_disco != DADO_ACEITO);
    aux=comunica.dado_disco & 0x000000FF;
    aux=(aux << 8) | aux;
    aux=(aux << 16) | aux;
    comunica.controle_disco= CONTINUA;
    put_pixel ((long)aux, x,y);
  }
comunica.controle_disco=DADO_ACEITO;
return erro;
}

```

mn_pd.c

```

#include "sysconst.h"
#include "data_str.h"
#include "lie_sgj.h"

/*****
* ARQUIVO: /MMNOT/MM_PD.C
* AUTOR: Castano Traina Junior
* DATA INICIAL: 23 de Setembro de 1987
* VERSAO ATUAL: 1.1
* NOTAS:
*   Mn_PDaria
*   Mn_PDprocessa
*   _hentry
* MODIFICACOES:
*   Data: 29/10/87   Sobre a versao: 1.0
*   Rotina:   as tres
*   Motivo:   Modificar a indicacao do menu armazenado,
*             que era feita pelo indice de um vetor da
*             forma _pdmnu[menu_id].x para o
*             ponteiro mn->x.
*   Data: 13/05/88   Sobre a versao: 1.0
*   Rotinas:   Mn_PDaria e Mn_PDprocessa
*   Motivo:   Os nomes antigos foram substituidos pelos
*             atuais: Create_quick_pd ----> Mn_PDaria
*             Process_pd ----> Mn_PDprocessa
*             As indicaçoes de mensagens de erro foram
*             unificadas.
*   Data: 26/02/91   Sobre a versao: 1.0
*   Rotinas:   Mn_PDaria e Mn_PDprocessa
*   Motivo:   Adequar-se a utilizacao no ambiente
*             TMS34010.
*****/

```

```

*
*****
struct __pdmnu { /* Cada estrutura ocupa 60 bytes! */
  int cor[4];
  int nro_entries, scr_size;
  cadeia *entry, *cad;
  int *command, *link, /* pd.link = pu.caract */
  *retorno;
  int *poslin, *poscol;
  int cursor, inipos; } __pdmnu[MAXMENU]=0;

char _pdmdef[MAXMENU]={0,0,0,0,0,0,0,0,0,0};
extern lie_fonte lie_fonte_corrente;
extern lie_janela *lie_jan_corrente, raiz;

/*-----
* Mn_PDaria - Definir um Menu Pull-Down.
*
* SINTAXE
* #include <data_str.h>
*
* int Mn_PDaria (red, green, blue, text_font, nro_entries,
*   scr_size, entry, command, link, menu_id_ptr)
*
* unsigned int red[4], green[4], blue[4]; // Intensidade da
*   cor, no indice:
*   0 - cor de fundo
*   1 - cor da borda
*   2 - cor dos textos regulares
*   3 - cor da entrada em highlight //
* cadeia text_font; // Nome do arquivo de fontes para os caract. //
* int nro_entries; // Numero maximo de entradas no menu //
* int scr_size; // numero de entradas simultaneas na janela //
* cadeia entry[]; // Nomes das entradas //
* int command[]; // Numero que deve retornar se essa
*   entrada for escolhida //
* int link[]; // identidade do menu que pode dar prosseguimen-
*   to 'a essa entrada, se ela for escolhida//
* register int *menu_id_ptr; // Identidade desse menu,
*   atribuido pela rotina //
*
* DESCRICAO
* Esta rotina cria um Menu Pull-Down segundo a definicao dada pelo
* usuario. As entradas do menu sao definidas pelos varios vetores
* "entry", "command", e "link", sendo que a cada indice do vetor
* corresponde uma entrada. O numero maximo de entradas "nro_entries"
* determina a quantidade de entradas que o menu tem. Do total de
* entradas do menu, apenas a quantidade indicada em "scr_size" e'
* mostrada simultaneamente na tela, sendo que as demais sao mostradas
* atraves de operacoes de "scroll".
*
* Os vetores indicadores de cores "red", "green", e "blue" determinam
* a intensidade com que cada cor participa da formacao da cor na
* parte correspondente do menu. Para isso, a cor e' composta
* utilizando-se dos bits de mais baixa ordem de cada entrada.
* A quantidade de bits utilizada depende do numero de planos de cores
* que esta' sendo utilizado pelo hardware.
*
* A entrada "text_font" destina-se a ser usada para indicar o fonte
* de letra a ser utilizado para escreverem-se as entradas do menu, quando
* existir mais de um fonte definido no sistema. Caso deva ser utilizado
* o fonte corrente, deve ser indicado o valor <NULL>.
*
* A variavel "menu_id_ptr", retorno da rotina, contem' o
* identificador do menu criado, o qual deve ser utilizado para se
* indicar que este menu deve ser ativado, ou para se estabelecer
* a hierarquia de menus atraves do "link" de cada entrada. O "link",
* se for indicado como nulo, indica que a entrada correspondente e'
* folha, e se escolhida corresponde 'a uma escolha final, terminando

```



```

* o processamento do menu. Caso esse valor nao seja zero, deve
* indicar um identificador de menu valido, o qual corresponde ao
* sub-menu ativado quando a entrada correspondente for escolhida.
*
* Uma vez criado um menu, este pode ser ativado atraves da rotina
* Mn_PDprocessa(3LINESJ). Sub-menus escolhidos durante o processamento de
* um menu sao por sua vez ativados automaticamente quando escolhidos.
*
* RETORNO
* Valor de Retorno da Rotina (tipo int):
* Indica se houve erro (vide ERROS:).
*
* menu_id_ptr - identificador do menu criado.
*
* RESTRICOES
* - Deve ser indicado um numero de entradas "nro_entries" de no
* minimo duas entradas.
* - Nao pode ser excedido o numero maximo de menus criados definido
* como MAXMENU no arquivo <sysconst.h>.
*
* ERROS
* 0 - Nao houve erro.
* 12 - Numero maximo de definicoes excedido (maximo MAXMENU).
*
* VALIA EXEMPLO
* Mn_PDprocessa(3LINESJ).
*
* EXEMPLO
* main()
* {
* // Definicoes de menu //
* int menu_princ;
* // Definicao das cores basicas //
* static int red[4] = {0,7,1,0},
* green[4] = {2,7,1,2},
* blue[4] = {0,7,0,2};
* static char *text_font = 0;
* static char *entry[] = {"\x0f\x0f Opcao 1",
* "\x0f\x0f Opcao 2",
* "\x0f\x0f Opcao tres",
* "\x0f\x0f Opcao quatro",
* "\x0f\x0f Parametros",
* "\x0f\x0f Fim"};
* static int command[] = {1,2,3,4,50,500};
* static int link[] = {0,0,0,0,0,0};
* static int i; // Retorno da processa menu //
*
* // Cria menu //
* Mn_PDcria (red, green, blue, text_font, 6, 4, entry,
* command, link, &menu_princ);
* i = Mn_PDprocessa (menu_princ,0,5,55,&comp);
*
* -----*/
Mn_PDcria (red, green, blue, text_font, nro_entries, scr_size,
entry, command, link, menu_id_ptr)
unsigned int red[4], green[4], blue[4]; /* Intensidade da cor, no indice:
0 - cor do fundo
1 - cor da borda
2 - cor dos textos regulares
3 - cor da entrada em highlight */
cadeia text_font; /* Nome do arquivo de fontes para os
caracteres */
int nro_entries; /* Numero maximo de entradas no menu */
int scr_size; /* numero de entradas simultaneas na
janela */
cadeia entry[]; /* Nomes das entradas */
int command[]; /* Numero que deve retornar se essa
entrada for escolhida */
int link[]; /* identidade do menu que pode dar

```

```

prosseguimento 'a essa entrada, se
ela for escolhida */
register int *menu_id_ptr; /* Identidade desse menu, atribuido
pela rotina */
{
register int i;
register struct __pdmenu *mn;
cadeia kcvf();

/* Verificar qual o numero do menu que pode ser usado */
*menu_id_ptr=-1;
for (i=0; i<MAXMENU; ++i)
if (!pdmdef[i]) {
*menu_id_ptr=i;
break;
}
if (*menu_id_ptr== -1) return 1;
pdmdef[i]=1; /* 1--> Menu PullDown */

mn=&pdmnu[*menu_id_ptr];

for (i=0; i<4; ++i) { /* Red 2 bits, Green 3 bits, Blue 3 bits */
mn->cor[i] = (((red[i]&0x3)<<6) | ((green[i]&0x7)<<3) |
(blue[i]&0x7)));
mn->cor[i] |= mn->cor[i]<<8;
mn->cor[i] |= mn->cor[i]<<16; /* Compl. 32 bits */
}
if (scr_size>nro_entries) scr_size=nro_entries;
mn->nro_entries = nro_entries;
mn->scr_size = scr_size;
for (i=0; i<nro_entries; i++) {
entry[i]=kcvf(entry[i]);
}
mn->entry = &entry[0];
mn->command = &command[0];
mn->link = &link[0];
mn->cursor=0;
mn->inipos=0;
(*menu_id_ptr)++;
return 0;
}

/*-----
* Mn_PDprocessa - Ativa o processamento do Menu indicado.
*
* SINOPSE
* #include <data_str.h>
*
* int Mn_PDprocessa (menu_id, screen_nro, lin_ory, cel_ory,
command_ptr)
*
* int menu_id; /* Numero da identidade do menu
que deve ser ativado, devolvido
pela rotina Mn_PDcria(). */
* int screen_nro; /* Numero da tela de video que deve ser usada. */
* int lin_ory, cel_ory; /* Posicao do canto superior esquerdo onde
o menu deve ser aberto, em coordenadas
de
video em termos de linha e coluna. */
* int *command_ptr; /* Numero da entrada escolhida. Corresponde ao
numero do vetor "command" definido para esta
entrada para este menu. */
*
* DESCRICAO
* Esta rotina ativa o processamento do menu indicado em "menu_id",
mostrando-o na tela, e permitindo que o usuario movimente o cursor
segundo sua vontade. O menu a ser mostrado deve ter sido previamente
definido pela rotina Mn_PDcria. Quando uma entrada e' escolhida,
retorna-se o valor correspondente 'a entrada atualmente em highlight

```

```

Mn_PDprocessa (menu_id, screen_nro, lin_org, col_org, command_ptr)
int menu_id;          /* Numero de identidade do menu que deve ser
                        ativado, devolvido pela rotina Mn_PDcria() */
int screen_nro;        /* Numero da tela de video que deve ser usada */
int lin_org, col_org; /* Posicao do canto superior esquerdo onde o
                        menu deve ser aberto, em coordenadas do video
                        em termos de linha e coluna */
int *command_ptr;      /* Numero da entrada escolhida. Corresponde ao
                        numero do vetor command definido para esta entrada
                        para este menu. */

{
    register short i, j;
    static short ncol, y1, xl, y2, x2;
    static cadeia umaentry;

```

```

if (i != 0) {
    i_printk ("erro=%d", i);
    return i;
}

Jan_corrente (jan1);
Mostra_janela(); /* Mostra a janela corrente */

/* Dimensoes da janela em unidades de caracteres */
l1=0;
c1=0;
l2=mn->scr_size;
c2=ncol;

```

```

/* Escrever as entradas */
for (i=0, j=mn->inipos; i<mn->scr_size; ++j, ++i)
    _hentry (mn, mn->cor[0], j, l1, c2);

/* Deixar em HighLight a linha onde esta o cursor */
highlight=mn->cursor;
_hentry(mn, mn->cor[3], highlight, l1, c2);

v_posic (highlight-mn->inipos,1);

/* Aceitar a entrada escolhida */
while (1) {
    /* Verificar se as codicoes de highlight mudaram */
    v_lugc(&lin, &col);
    if (col >= c1 && col <= c2 && lin >= l1 && lin <= l2) {
        /* Entao o cursor esta na janela */
        if (highlight >= 0) {
            if (lin-l1!=highlight-mn->inipos) {
                /* Mudou a linha onde esta o cursor */
                _hentry(mn, mn->cor[0], highlight, l1, c2);
                highlight=(lin-l1)+mn->inipos;
                _hentry(mn, mn->cor[3], highlight, l1, c2);
            }
            else { /* antes nao estava na janela. fazer highlight */
                highlight=(lin-l1)+mn->inipos;
                _hentry (mn, mn->cor[3], highlight, l1, c2);
            }
        }
        else { /* o cursor nao esta na janela */
            if (highlight >= 0) { /* antes estava na janela */
                _hentry (mn, mn->cor[0], highlight, l1, c2);
                highlight=-1;
            }
        }
    }
    v_posic (lin, col); /* restaura a posicao do cursor */
    /* Obter a proxima acao do usuario */
    c=i_getc();
    switch (c) {
        case 200: /* Seta Acima */
            lin--;
            if (lin==l1-1 && col >= c1 && col <= c2) {
                ++lin;
                if (mn->inipos>0) {
                    highlight=-mn->inipos;
                    /* Reescrever as entradas */
                    for (i=0, j=mn->inipos; i<mn->scr_size; ++j, ++i)
                        _hentry (mn, mn->cor[0], j, l1, c2);
                    /* Deixar em HighLight a linha onde esta o cursor */
                    _hentry(mn, mn->cor[3], highlight, l1, c2);
                }
            }
            break;
        case 208: /* Seta Abaixo */
            lin++;
            if (lin==l2 && col >= c1 && col <= c2) {
                --lin;
                if (mn->inipos+mn->scr_size < mn->nro_entries) {
                    _hentry (mn, mn->cor[0], highlight, l1, c2);
                    highlight=(lin-l1) + (mn->inipos);
                    v_posic (lin+1, col); v_putchar(0);
                    _hentry (mn, mn->cor[3], highlight, l1, c2);
                }
            }
            break;
        case 203: /* Seta esquerda */
            col--;
            break;
        case 205: /* Seta Direita */
            col++;
    }
}

```

```

break;
case 13: /* Return */
case 211: /* DEL */
    if (highlight >= 0) {
        mn->cursor = highlight;
        if (mn->link[highlight] == 0) { /* O item escolhido nao
                                         tem link */
            *command_ptr = mn->command[highlight];
            Retira_janela(janl);
            Jan_corrente (janant);
            return (0);
        }
        else { /* Se tem um link */
            car2xy(lin,c2-1, &xl,&yl); /* Coord. em pixel */
            xl+=janl->xlo;
            yl+=janl->ylo; /* Transf. coordenadas de tela */
            Mn_PDprocessa (mn->link[highlight],
                           screen_nro, yl, xl, command_ptr);
            if (*command_ptr >= 0) {
                Retira_janela (janl);
                Jan_corrente (janant);
                return (0);
            }
        }
        break;
    }
    else { /* Retorno sem haver um item em highligh */
        *command_ptr=-1;
        Retira_janela (janl);
        Jan_corrente (janant);
        return (0);
    }
} /* Fim do switch */
v_posic(lin, col);
} /* fim do while (Nunca sai por aqui) */

/*-----
* Coloca uma entrada da janela na cor indicada. (Rotina Interna)
*/
_hentry (mn, cor, entrada, l1, c1)
register int cor;
short l1, c1, entrada;
register struct __pdmnu *mn;
{
    register cadeia umaentry;
    unsigned char aux;
    lie_jan_corrente->cor_fundo=cor; /* E passado a cor do fundo da entrada */
    v_posic (entrada - mn->inipos,0);
    umaentry = (cadeia) mn->entry[entrada];
    aux=umaentry[1];

    if (aux>MN_NCAR) umaentry[1]=MN_NCAR; /* Cadeia limitada em MN_NCAR */
    v_wputk(umaentry);
    umaentry[1]=aux;
    v_posic (entrada - mn->inipos, c1-1); /* Colocar a seta a direita */
    if (mn->link[entrada])
        pchar(INI_FONTE);
    else
        pchar(' ');
    return 0;
}

```

mostra.c

```
#include "data_str.h"
#include "lie_sgj.h"
#include "sysconst.h"

extern lie_janela *lie_jan_corrente;
extern lie_fonte lie_fonte_corrente, lie_icones;

/*-----
*----- TMS34010 - LIE
*-----
* Arquivo: mostra.c
* Funcoes: int Mostra_janela ();
*          int Recorda_janela (vid);
* Esta funcao e projetada par ser chamada por um GWT-C program
*-----
* Historico:
* - Data de criacao: 01-02-91 ..... Agma Jaci Machado Traina
* - 03-02-91 Inclusao de salva tela para sobreposicao de janelas
*                               Agma.
*-----
*-----
* ERROS:
* Mostra_janela
*-----
* SINGLOS:
* int Mostra_janela ()
*-----
* DESCRICAO:
* A rotina Mostra_janela mapeia e mostra no video a janela corrente. Caso
* a janela corrente seja uma janela com borda, a borda e' mostrada
* tambem, juntamente com os atributos correntes para essa janela e borda.
*-----
* RETORNO
* Valor de Retorno da Rotina (tipo int):
* Indica se houve erro (vide ERROS:).
*-----
* RESTRICOES:
* - Deve haver uma janela corrente consistente.
*-----
* ERROS:
* 0 - Tudo OK. Janela corrente mapeada.
* 1 - Estrutura da janela corrente danificada.
*-----
* VALS TMS34010:
* Cria_janela(3LIEGJ), Jan_corrente(3LIEGJ).
*-----
* EXEMPLO:
* // Mapear a janela corrente: //
* Mostra_janela ();
*-----
* // Criar uma janela descendente diretamente da tela inteira, e
* mapea-la para uso imediato: //
* extern lie_janela raiz;
* lie_janela *cria;
* Jan_corrente (&raiz);
* Cria_janela (&cria,
*             XLO, 20,   YLO, 10,
*             XHI, 110,  YHI, 200,
*             FIM);
* Jan_corrente (cria);
* Mostra_janela ();
*-----*/
```

```
int Mostra_janela ()
{
    int max;
    long largura;
    lie_janela *pai, *corrente;
    lie_tipo_janela *tborda;
    lie_fonte aux;

    if (lie_jan_corrente->sentinela != 1234) /* testa consistencia! */
        return 1; /***** Estrutura da janela inconsistente ****/

    corrente=lie_jan_corrente;

    /* Verificar se a janela possui bordas */
    if ((pai=lie_jan_corrente->pai) != FIM)
        if (pai->tipo != 0) { /* a janela tem borda */
            Jan_corrente (pai);
            pai->estado |= 0x01; /* Mapeia janela pai */
            if (pai->base != 0) { /* Salva a tela antes de mapear a janela */
                largura=pai->xhi - pai->xlo + 1+ 16; /* ajustar word boundary*/
                largura=(long)(largura / 16)*16;
                get_rect ( pai->xhi - pai->xlo + 1, pai->yhi - pai->ylo + 1,
                    pai->xlo, pai->ylo, (short *)pai->base,
                    largura * get_pixel_size());
            }
            Limpa_jan_corrente();
            tborda=lie_jan_corrente->tipo;
            if (tborda->tipo_borda & 0x01) { /* E' com borda */
                if (tborda->tipo_borda & 0x20) { /* E' com borda singela */
                    des_borda_singela();
                }
                else { /* E' com borda ampla */
                    if (tborda->tipo_borda & 0x02) /* Borda dupla */
                        des_borda_dupla();
                    else /* Borda simples */
                        des_borda_simples();
                }
                if (tborda->texto != 0) {
                    /* Escrever a cadeia titulo da janela.
                     Posicionar, verificar quanto cabe e escrever... */
                    Move_cursor (23, 2);
                    max= pai->xhi-pai->xlo - 33 - NICONEMAX*16;
                    max=max/lie_fonte_corrente.tam_car_x;
                    if (max < tborda->texto[1]) tborda->texto[1]=max;
                    v_printk ("%a", tborda->texto);

                    /* ... Colocar o icone de apagar janela. */
                    Move_cursor (5,2);
                    aux=lie_fonte_corrente;
                    lie_fonte_corrente=lie_icones;
                    pchar(INI_FONTE); /* Desenha o icone de Apagar a
                                         Janela */

                    /* ... Colocar o icone de Fechar janela. */
                    Move_cursor ( pai->xhi- pai->xlo-21,2);
                    pchar(INI_FONTE+1); /* Desenha o icone de
                                         Fechar a Janela */

                    lie_fonte_corrente=aux;
                }
            } /* Fim do elseif janela com borda ampla */
        }
    Jan_corrente (corrente);
    corrente->estado |= 0x01; /* janela do usuario mapeada */
    Limpa_jan_corrente();
} /* Fim da rotina Mapeia_janela */

des_borda_simples()
{
    lie_janela *j;
}
```

```

j=lie_jan_corrente;

/*
fill (j->xhi - j->xlo, 1, j->xlo, j->ylo); Linha superior externa
fill (j->xhi - j->xlo - 17, 1, j->xlo + 5, j->ylo + 18); interna

fill (2, j->yhi - j->ylo, j->xlo, j->ylo); linha lateral esquerda ext.
fill (2, j->yhi - j->ylo - 18, j->xlo+5, j->ylo + 18); interna

fill (2, j->yhi - j->ylo, j->xhi - 1, j->ylo); lin. lateral direita ex.
fill (2, j->yhi - j->ylo - 18, j->xhi - 14, j->ylo + 18); interna

fill (j->xhi - j->xlo, 1, j->xlo, j->yhi); Linha inferior externa
*/
draw_box (j->xhi - j->xlo, j->yhi - j->ylo, j->xlo, j->ylo);
draw_box (j->xhi - j->xlo - 17, j->yhi - j->ylo, j->xlo + 5, j->ylo+18);
}

des_borda_dupla()
{
    lie_janela *j;
    j=lie_jan_corrente;

    fill (j->xhi - j->xlo, 2, j->xlo, j->ylo); /* Linha superior externa */
    fill (j->xhi - j->xlo - 17, 1, j->xlo + 5, j->ylo + 18); /* interna */

    fill (3, j->yhi - j->ylo, j->xlo, j->ylo); /*linha lateral esquerda ext.
*/
    fill (2, j->yhi - j->ylo - 18, j->xlo+5, j->ylo + 18); /* interna */

    fill (3, j->yhi - j->ylo, j->xhi - 2, j->ylo); /* lin. lateral direita
ex.*/
    fill (2, j->yhi - j->ylo - 18, j->xhi - 14, j->ylo + 18); /* interna */

    fill (j->xhi - j->xlo, 2, j->xlo, j->yhi-1); /* Linha inferior externa
*/
}

/*-----
int des_borda_singela ()
{
    lie_janela *mid;
    mid=lie_jan_corrente;
    draw_box (mid->xhi - mid->xlo, mid->yhi - mid->ylo, mid->xlo, mid->ylo);
}

/*-----
* NOME:
* Esconde_janela
*
* SINTAXE:
* int Esconde_janela(wid)
* lie_janela *wid;
*
* DESCRIÇÃO:
* Retira da tela a janela corrente, mantendo a estrutura de
* janelas inalterada.
*
* RETORNO:
* Valor de Retorno da Rotina:
* Erro (vide ERROS:).
*
* RESTRIÇÕES:
* - A janela corrente deve estar consistente.
*
* ERROS:

```

```

* 0 - Tudo OK. Janela corrente mapeada.
* 1 - Estrutura da janela corrente danificada.
*
* VEJA TAMBÉM:
* Cria_janela(3), Mostra_janela(3)
*
* EXEMPLO:
* // Mapear a janela corrente: //
* Esconde_janela (jid);
*
*-----*/

int Esconde_janela (wid)
lie_janela *wid;
{
    lie_janela *pai;
    long largura;

    if (wid->sentinela != 1234) /* testa consistencia! */
        return 1; /***** Estrutura da janela inconsistente *****/

    /* Verificar primeiro se a janela nao tem borda. */
    if ( ((pai=wid->pai) != FIM) && (pai->tipo != 0) ) {
        /* A janela tem borda. Verificar se a janela esta mapeada. */
        if ( ((pai->estado & 0x01) == 1) && (pai->base != 0) ) {
            Jan_corrente (pai);
            largura=pai->xhi - pai->xlo + 1+ 16; /* ajustar word boundary */
            largura=(long) (largura / 16)*16;
            put_rect ( (short *)pai->base,
                        largura * get_pixel_size(),
                        pai->xhi - pai->xlo + 1, pai->yhi - pai->ylo + 1,
                        pai->xlo, pai->ylo);
            pai->estado &= 0xFE; /* Janela desmapeada */
            Jan_corrente (pai->pai); /* Como a janela c. borda o seu pai
*/
                                /* e' a borda, fixa para pai->pai */
                            }
            }
        else {
            /* Verificar se a propria janela esta mapeada, nao tem borda. */
            if (wid->estado & 0x01 == 1 && wid->base != 0) {
                Fixa_jan_fisica(wid);
                largura=wid->xhi - wid->xlo + 1+ 16; /* ajustar word boundary */
                largura=(long) (largura / 16)*16;
                put_rect ( (short *)wid->base,
                            largura * get_pixel_size(),
                            wid->xhi - wid->xlo + 1, wid->yhi - wid->ylo + 1,
                            wid->xlo, wid->ylo);
                wid->estado &= 0xFE;
                Jan_corrente (wid->pai);
            }
        }
    }
    return 0;
}

```

movecursor.c

```

#include "data_str.h"
#include "lie_sgl.h"

extern lie_janela *lie_jan_corrente;

/*-----
*----- TMS34010 - LIE -----
*-----

```

```

* Arquivo: MoveCursor.c
* Funcoes: Move_cursor(x, y)
*
* Esta funcao e projetada par ser chamada por um GNU-C program
*
*-----
* Historico:
* - Data da criacao: 01-02-91 ..... Agma Juci Machado Traina
*-----
* NOME:
* Move_cursor
*
* SINOPSE:
* int Move_cursor(x, y)
* int x, y;
*
* DESCRICAO:
* Posiciona o cursor dentro da janela corrente.
*
* RETORNO:
* Valor de Retorno da Rotina:
* Zero (vide ERROS:).
*
* RESTRICOES:
* - A janela deve ter sido definida corretamente, e sua estrutura
  deve estar integra.
*
* ERROS:
* 0 - Tudo OK. Janela indicada e' a corrente.
* 1 - Estrutura da janela indicada danificada.
*
* VALIA TAMBEM:
* cria_janela(3)
*
* EXEMPLO:
* // Tornar corrente a propria tela inteira: //
* entao lie janela raiz;
* Jan_corrente (&raiz);
*-----*/

Move_cursor(x, y)
int x, y;
{
    if (lie_jan_corrente->sentinela != 1234)
        return 1;
    lie_jan_corrente->x_cursor = x;
    lie_jan_corrente->y_cursor = y;
    return 0;
}

```

pchar.c

```

/*-----
*----- TMS34010 - LIE
*-----
* Funcao pchar
*
* Esta funcao e projetada par ser chamada por um GNU-C program
*
*-----
* Historico:
* - Data da criacao: 10-12-90 ..... Agma Juci Machado
*-----
*
* pchar - Desenha um caracter na posicao corrente do cursor.
*

```

```

* SINOPSE
* int pchar (caract)
* char caract;
*
* DESCRICAO
* A rotina pchar desenha um caracter na posicao indicada pelo cursor,
* dentro da janela corrente, e segundo o fonte corrente. Uma vez
* desenhado o caracter, o cursor e' deslocado na linha para que
* chamadas consecutivas da rotina desenhem os caracteres em sequencia.
* Os seguintes caracteres de controle sao tambem interpretados:
* <Back Space> - volta o cursor e espaco de um caracter;
* <Line Feed> - Avanca o cursor para a linha seguinte;
* <Return> - Posiciona o cursor no inicio da linha corrente;
* <Tab> - Posiciona o cursor numa posicao multipla de 8 caracteres
* na linha corrente, a frente da posicao atual do cursor.
*
* Antes do caracter ser escrito, a posicao corrente do cursor e'
* avaliada para verificar se esta' dentro da janela corrente. Caso
* esteja fora da janela, abaixo da mesma, considera-se a necessidade de
* um rolamento vertical da janela, para posicionar o cursor dentro da
* mesma. Caso o cursor esteja acima da janela, nenhuma acao e' tomada,
* e o mesmo ocorrendo caso o cursor esteja a direita ou a esquerda da
* janela.
*
* RETORNO
* Valor de Retorno da Rotina (tipo int):
* A rotina sempre retorna a constante Zero.
*
* NOTA
* Esta rotina e' usada internamente para as operacoes de saida de
* caracteres na tela do TMS, tendo sido portanto construida com a
* intensao de ser utilizada como rotina interna do sistema.
*
* EXEMPLO
* // Escrever todos os caracteres do fonte corrente, 10
* caracteres por linha: //
* for (i=32; i<128; i++)
*     if ( 10*(int)(i/10)==i ) {
*         pchar (10); pchar (13)
*     }
*     else {
*         pchar(i);
*     }
*-----*/

```

```

#include "data_str.h"
#include "sysconst.h"

```

```

extern lie_fonte lie_fonte_corrente;
extern lie_janela *lie_jan_corrente;

```

```

pchar(c)
char c;
{
    char *desloc;
    int sy, tx, ty, aux;

    /* Verifica se o caracter nao cabe na janela, entao faz scroll */
    ty = lie_jan_corrente->yhi - lie_jan_corrente->ylo; /* tam. da janela */
    if ((lie_jan_corrente->y_cursor + lie_fonte_corrente.tam_car_y) > ty) {
        /* Fazer scroll de um bloco equivalente */
        sy = lie_jan_corrente->y_cursor + lie_fonte_corrente.tam_car_y - ty;
        tx = lie_jan_corrente->xhi - lie_jan_corrente->xlo + 1;
        move_rect (tx, ty - sy + 1,
                  lie_jan_corrente->xlo, lie_jan_corrente->ylo + sy + 1,
                  lie_jan_corrente->xlo, lie_jan_corrente->ylo);

        /* Apagar o bloco */
        set_color1 ((long)lie_jan_corrente->cor_fundo);
        fill (tx, sy, lie_jan_corrente->xlo, lie_jan_corrente->yhi - sy);
    }
}

```

```

set_color1 ((long)lie_jan_corrente->cor_frente);

lie_jan_corrente->y_cursor = ty - lie_fonte_corrente.tam_car_y;
}

/* Nao e caracter especial */
if ( c >= INI_FONTE) {
    c=INI_FONTE; /* O conjunto de caracteres esta definido a
                  partir de 20 */
    aux = lie_fonte_corrente.tam_car_x *
          lie_fonte_corrente.tam_car_y * c;
    desloc =(char *) lie_fonte_corrente.fonte + (aux/8); /* Pois
                                                         desloca 'char' */
    set_color1 ((long)lie_jan_corrente->cor_frente);
    set_color0 ((long)lie_jan_corrente->cor_fundo);

    draw_car (desloc,
              lie_jan_corrente->x_cursor + lie_jan_corrente->xlo,
              lie_jan_corrente->y_cursor + lie_jan_corrente->ylo);

    /* Atualiza posicao do cursor */
    lie_jan_corrente->x_cursor +=lie_fonte_corrente.tam_car_x;
    return 0;
}

switch (c) {
case 8: /* Back Space */
    lie_jan_corrente->x_cursor --lie_fonte_corrente.tam_car_x;
    break;

case 127: /* Delete */
    break;

case 10: /* Line Feed */
    lie_jan_corrente->y_cursor+=lie_fonte_corrente.tam_car_y;
    break;

case 13: /* Carriage Return */
    lie_jan_corrente->x_cursor=0; /* Posiciona no inicio da
                                linha */
    break;

case 9: /* HTab */
    aux=lie_jan_corrente->x_cursor / 8;
    lie_jan_corrente->x_cursor = aux * 8 + 8;
    break;

default: return 0;
} /* fim do case */
}

```

posiciona.c

```

#include "data_str.h"
#include "lie_sgj.h"

extern lie_janela *lie_jan_corrente;

/*-----
*----- TMS34010 - LIE -----
*-----
* Arquivo: Posiciona.c
* Funcoes: Posiciona (jan, x, y)
*

```

```

* Esta funcao e projetada para ser chamada por um GW-C program
*
*-----
* Historico:
* - Data de criacao: 16-04-91 ..... Agma Jaci Machado Traina
*-----
*-----
* NOME:
* Posiciona
*
* SINOPSE:
* int Posiciona(jan, x, y)
* int x, y;
* lie_janela *jan,
*
* DESCRICAO:
* Posiciona o cursor dentro da janela especificada - "jan".
*
* RETORNO:
* Valor de Retorno da Rotina:
* Erro (vide ERROS:).
*
* RESTRICOES:
* - A janela deve ter sido definida corretamente, e sua estrutura
  deve estar integra.
*
* ERROS:
* 0 - Tudo OK.
* 1 - Estrutura da janela indicada danificada.
*
* VEJA TAMBM:
* cria_janela(SLIBW).d
*-----*/

Posiciona(jan, x, y)
int x, y;
lie_janela *jan,
{
    if (jan->sentinela != 1234 )
        return 1; /* Estrutura de janela deteriorada */
    jan->x_cursor = jan->xlo + x;
    jan->y_cursor = jan->ylo + y;
    return 0;
}

```

retjan.c

```

#include "data_str.h"
#include "lie_sgj.h"

extern lie_janela *lie_jan_corrente;

/*-----
*----- TMS34010 - LIE -----
*-----
* Arquivo: Retjan.c
* Funcao Retira_janela
*
* Esta funcao e projetada para ser chamada por um GW-C program
*
*-----
* Usa-se: Retira_janela (wid);
* onde:

```

```

*      int wid; // especifica a janela a ser retirada //
*
*-----
* Historico:
* - Data de criacao: 07-02-91      ..... Agma Juci Machado Traina
*-----
* NOME:
* Retira_janela
*
* SINOPSE:
* int Retira_janela (wid)
*
*   lie_janela *wid;
*
* DESCRICAO:
* Retira a janela especificada por "wid" da hierarquia de
* janelas. Se a janela especificada foi criada com opcao de SOBREPONER,
* a janela tambem sera desmapeada da tela.
*
* RETORNO:
* Valor de Retorno da Rotina:
* Erro (vide ERROS:).
*
* RESTRICOES:
* - Se a janela especifica nao foi criada com opcao de sobrepesicao,
* ela apenas sera retirada da estrutura hierarquica, mas nao sera
* desmapeada da tela.
*
* ERROS:
* 0 - Tudo OK. Janela criada corretamente.
* 1 - Estrutura da janela indicada danificada.
* 3 - Janela especificada possui filhos.
* 4 - Janela especificada e' a raiz.
*
* VALIA TAMBEM:
* Limpa_janela(3LINESJ), Cria_janela(3LINESJ).
*
* EXEMPLO:
* // Criar uma janela descendente diretamente da tela inteira: //
* extern lie_janela *cria, raiz;
* Jem corrente (raiz); // torna a raiz janela corrente //
* Cria_janela (cria,
*             XLO, 20,   YLO, 10,
*             XHI, 110, YHI, 200,
*             FIM);
* Ret_janela (cria);
* .....
* Neste caso a janela nao sera desmapeada da tela, apenas
* da estrutura.
*-----*/

int Retira_janela (wid)
lie_janela *wid;
{
    int erro;
    lie_janela *auxjan;

    if (wid->estado == 1) /* A janela esta mapeada */
        Esconde_janela(wid);

    /* Verifica se a janela pai possui borda */
    if ( ((auxjan->pai) != FIM) && (auxjan->tipo != 0) )
        /* Sim, entao retira tambem a borda (pai) */
        wid=auxjan;

    auxjan=wid->pai;
    erro = Ret_subarvore (wid);

```

```

if ( !erro ) Jan_corrente (auxjan); /* A janela pai torna-se corrente */
return erro;
} /* Fim da rotina Retira_janela */

```

rot_arvore.c

```

#include "data_str.h"
#include "lie_sqj.h"

void *malloc();

extern lie_janela raiz;
extern lie_fonte lie_fonte_corrente;

/*****
*
*   filename: rot_arvo.c
*
*   Este arquivo contem as rotinas de manipulacao da estrutura em
*   arvore que controla as janelas do sistema WINDOW-LIE.
*
*-----
* Historico:
* - Data de criacao: 06-02-90      ..... Agma J. M. Traina
* - Definicao de fonte de caracte  ..... 05-12-90
* - Definicao de janela corrente  ..... 06-12-90
*-----*/

/* Cria_arvore -- inicializa o primeiro no' da arvore.
*
* SINOPSE
* void Cria_arvore (maxlin, maxcol)
* int maxlin, maxcol;
*
*
* DESCRICAO
* A rotina Cria_arvore inicializa a estrutura de janelas, e o primeiro
* no' dessa estrutura, que corresponde a raiz da arvore, e a tela
* inteira do video, que poder ser utilizada. As variaveis "maxlin" e
* "maxcol" definem o tamanho maximo da tela em pixels. A variavel
* "raiz" e' definida estaticamente, e pode passar a ser acessada a
* partir de outros modulos. A raiz passa a ser definida para o fonte
* corrente, o qual deve ser estabelecido antes da chamada desta rotina.
*
* NOTA
* - O valor de "maxlin" e "maxcol" indicados definem a area do video
* que podera' ser utilizada posteriormente.
* - O fonte corrente sera' tomado como fonte da raiz, e deve ser
* definido antes da chamada desta rotina.
*
* VALIA TAMBEM
* Cria_filho(3LINESJ), Retira_filho(3LINESJ), Ret_subarvore(3LINESJ),
* Inicializa(3LINESJ).
*
* EXEMPLO
* #include <data_str.h>
* #include <sysconst.h>
* extern byte charset[96][16];
* lie_fonte lie_fonte_corrente = { (byte *)charset, 8, 16};
*
* Cria_arvore(256X, 256X);
*-----*/

void Cria_arvore (maxlin, maxcol)
int maxlin, maxcol;

```



```

(
    raiz.pai=FIM; /* A raiz nao tem pai */
    raiz.prim_filho=FIM;
    raiz.prox_irmao=FIM; /* Todas as outras listas estao vazias */
    raiz.ant_irmao=FIM;
    raiz.tipo=FIM; /* Nao e associado tipo a raiz */
    raiz.sentinel=1234; /* Estabelece o valor de checagem */
    raiz.xlo=0;
    raiz.ylo=0;
    raiz.xhi=maxcol;
    raiz.yhi=maxlin;
    raiz.plane_mask=(unsigned short) 0x0000; /* trabalha com todos os planos
*/
    raiz.event_mask=(unsigned short) 0x000F; /* Seta os 4 bits de eventos
ja'
                                definidos */
    raiz.cor_frente=(unsigned int) 0xFFFFFFFF; /* Cor mais clara */
    raiz.cor_fundo=0x00000000; /* Cor mais escura */
    raiz.x_cursor=0; /* Posicao inicial do cursor */
    raiz.y_cursor=0;
    raiz.base=(int *)0;
    raiz.arq_fonte=lie_fonte_corrente.fonte;
    raiz.estado=0x01; /* raiz sempre esta mapeada */
    Fixa_jan_fisica(&raiz);
} /* Fim da rotina Cria_arvore */

/*-----
* Fim_jan_fisica - Estabelece o ambiente de Hardware para a janela.
*
* SINOPSE
* @include <data_str.h>
*
* int Fim_jan_fisica (wid);
* lie_janela *wid;
*
* DESCRICAO
* Esta rotina estabelece os parametros fisicos do hardware que permitem
* realisar as operacoes seguintes sobre a area de video correspondente
* a janela wid indicada. Sao fixados os registradores "PMASK", "WSTART",
* "WEND", control, color0 e color1.
*
* RETORNO
* Valor de Retorno da Rotina (tipo int):
* Indica se houve erro (vide ERROS:).
*
* ERROS
* 0 - Tudo OK.
* 1 - Estrutura da janela indicada esta inconsistente.
*
* VELA TRAVESSA
* JanCorrente(3LIEAJ).
*
*-----*/

Fixa_jan_fisica (janela)
lie_janela *janela;
{
    unsigned short pmask, control;
    short *pointer;
    unsigned int dim, pixel_size;

    if (janela->sentinel != 1234) /* testa consistencia! */
        return 1; /* Estrutura da janela inconsistente */

    pixel_size=get_pixel_size();
    switch (pixel_size) {
        /* A mascara tem que ser replicada no registrador PMASK, conforme o
        tamanho do pixel */
        case 16:

```

```

        pmask=0xFFFF & janela->plane_mask;
        break;

    case 8:
        pmask=0x00FF & janela->plane_mask;
        pmask |= (pmask<<8);
        break;

    case 4:
        pmask=0x000F & janela->plane_mask;
        pmask |= (pmask<<4);
        pmask |= (pmask<<8);
        break;

    case 2:
        pmask=0x0003 & janela->plane_mask;
        pmask |= (pmask<<2);
        pmask |= (pmask<<4);
        pmask |= (pmask<<8);
        break;

    case 1:
        pmask=0x0001 & janela->plane_mask;
        pmask |= (pmask<<1);
        pmask |= (pmask<<2);
        pmask |= (pmask<<4);
        pmask |= (pmask<<8);
        break;
    }
    pointer=(short *)0xC0000160;
    *pointer=(short)pmask;

    /* Setar ativar Window Clipping. */
    control=get_control();
    control |= 0x00C0; /* Faz clipping sem gerar interrupcao */
    set_control( (int)control);

    /* Setar os limites da Window
    WSTART: || Window start Y | Window start X || */
    dim = 0x0000FFFF & janela->ylo;
    dim <= 16;
    dim |= 0x0000FFFF & janela->xlo;
    poke_breg ((short)5, (long) dim);

    /* WEND: || Window end Y | Window end X || */
    dim = 0x0000FFFF & janela->yhi;
    dim <= 16;
    dim |= 0x0000FFFF & janela->xhi;
    poke_breg ((short)6, (long) dim);

    /* Estabelece as cores de Frente e Fundo */
    set_color1 ((long)janela->cor_frente);
    set_color0 ((long)janela->cor_fundo);

    return 0;
} /* Fim da seta_jan_corrente */

/*-----
* Cria_filho - cria uma estrutura de janela.
*
* SINOPSE
* @include <data_str.h>
*
* int Cria_filho(ret_filho, corrente)
* lie_janela **retfilho, *corrente;
*
* DESCRICAO
* Esta rotina cria uma janela subordinada (filho) da janela indicada

```

```

* como corrente, duplicando os parametros da definicao dessa janela.
* Imediatamente seguindo 'a chamada desta rotina, deve-se proceder 'a
* definicao de parametros reais que essa janela devera' ter.
*
* RETORNO
*   Valor de Retorno da Rotina (tipo int):
*       Indica se houve erro (vide ERROS:).
*   ret_filho:
*       descritor da janela criada.
*
* RESTRICOES
*   - corrente deve ser uma janela ja' existente valida.
*
* NOTA
*   - Todos os parametros da janela corrente indicada sao duplicados
*     para a janela ret_filho criada.
*
* ERROS
*   0 - Tudo OK. Janela criada
*   1 - Janela corrente indicada nao e' uma janela valida.
*   2 - Memoria insuficiente para esta operacao.
*
* VEJA TAMBEEM
*   Cria_arvore(3LINESJ), Retira_filho(3LINESJ), Ret_subarvore(3LINESJ).
*
* EXEMPLO
*
*   #include <data_str.h>
*   extern lie_janela *raiz;
*   lie_janela *nova_jan;
*
*   Cria_filho(&nova_jan, raiz); /* Cria uma janela diretamente
*                               na tela */
*
*-----*/
Cria_filho (ret_filho, corrente)
lie_janela **ret_filho, *corrente;
{
    lie_janela *filho;
    if (corrente->sentinela != 1234) /* testa consistencia! */
        return (1); /* Argumento de janela inconsistente */

    filho = (lie_janela *)malloc(sizeof(lie_janela));
    if (filho == FIM)
        return (2); /* Memoria insuficiente para esta operacao */
    *ret_filho=filho;

    /* Acerta os Ponteiros das listas, sendo que a insercao e sempre no
       inicio da lista */
    filho->pai=corrente;
    filho->prim_filho=FIM;
    filho->prox_irmao=corrente->prim_filho;
    filho->ant_irmao=FIM;
    filho->tipo=FIM;
    if (corrente->prim_filho != FIM) /* Inclui na lista de filhos, se
                                     nao e o primeiro */
        (corrente->prim_filho)->ant_irmao=filho;
    corrente->prim_filho=filho;

    /* Cria os campos de Informacao do filho, com os mesmos atributos
       que o pai */
    filho->sentinela=1234;
    filho->xlo=corrente->xlo;
    filho->ylo=corrente->ylo;
    filho->xhi=corrente->xhi;
    filho->yhi=corrente->yhi;
    filho->plane_mask=corrente->plane_mask;

```

```

filho->event_mask=corrente->event_mask;
filho->cor_frente=corrente->cor_frente;
filho->cor_fundo=corrente->cor_fundo;
filho->x_cursor=0;
filho->y_cursor=0;
filho->base=(int *)0;
filho->arq_fonte=corrente->arq_fonte;
filho->estado=0x00; /* janela nao mapeada */

return (0);
} /* Fim da rotina Cria_filho */

```

```

/*-----*/
*   Retira_filho - Retira um no da estrutura de janelas.
*
*   SINOPSE
*   #include <data_str.h>
*
*   int Retira_filho (wid)
*   lie_janela *wid;
*
*   DESCRICAO
*   Esta rotina retira a janela "wid" da estrutura de janelas. Note-se
*   que a rotina nao verifica se a janela esta' mostrada no video, o que
*   implica a necessidade de se retirar a janela do video, atraves da
*   rotina Esconde_janela antes da chamada desta rotina. Para que esta
*   rotina possa retirar a janela, e' necessario que esta nao tenha nenhuma
*   outra janela subordinada.
*
*   RETORNO
*   Valor de Retorno da Rotina (tipo int):
*       Indica se houve erro (vide ERROS:).
*
*   RESTRICOES
*   - A janela a ser retirada nao deve estar mapeada no video, e
*     nao possuir janelas subordinadas.
*
*   ERROS
*   0 - Tudo OK.
*   1 - Janela indicada inconsistente.
*   3 - A janela possui filhos, nao pode ser retirada.
*   4 - A janela raiz nao pode ser retirada.
*
*   VEJA TAMBEEM
*   Cria_arvore(3LINESJ), Cria_filho(3LINESJ), Ret_subarvore(3LINESJ),
*   Esconde_janela(3LINESJ).
*
*-----*/

```

```

Retira_filho (filho)
lie_janela *filho;
{
    lie_janela *aux_jan;

    if (filho->sentinela != 1234)
        return (1); /* Argumento de janela inconsistente */

    /* verificar se nao e' a raiz */
    if (filho == &raiz)
        return (4); /* A raiz nao pode ser retirada ! */

    /* Verificar se o "filho" tem filhos */
    if (filho->prim_filho != FIM)
        return (3); /* Nao pode retirar janela que possua filhos */

    /* Desaloca a area de sobreposicao */
    if (filho->base != 0) {
        free(filho->base);
    }
}

```

```

        filho->base=0;
    }
    /* Retira a lista lie_tipo_janela associada */
    if (filho->tipo != 0) {
        free(filho->tipo);
        filho->tipo=0;
    }

    /* "filho" e' o primeiro da Lista? */
    aux_jan = filho->pai;
    if (aux_jan->prim_filho == filho) { /* E' o primeiro ! */
        aux_jan->prim_filho=filho->prox_irmao;
        (filho->prox_irmao)->ant_irmao=FIM;
    }
    else { /* Nao e' o primeiro ! */
        aux_jan=filho->ant_irmao;
        aux_jan->prox_irmao=filho->prox_irmao;

        aux_jan=filho->prox_irmao;
        if (aux_jan != FIM) /* Nao e' o ultimo da lista */
            aux_jan->ant_irmao=filho->ant_irmao;
    }
    filho->sentinela=0; /* Destroi a sentinela */
    free (filho);

    return 0;
} /* Fim da rotina Retira_filho */

/*-----*/
*      Ret_subarvore - Retira uma subarvore da estrutura de janelas.
*
*      SINOPSE
*      #include <data_str.h>
*
*      int Ret_subarvore(wid)
*      lie_janela *wid;
*
*      DESCRICAO
*      Esta rotina elimina da estrutura de janelas a janela "wid"
*      indicada, bem como todas as janelas suas subordinadas. Nota-se
*      que a rotina nao verifica se a janela esta' mostrada no video,
*      o que implica a necessidade de se retirar a janela (e suas
*      subordinadas) do video, atraves da rotina Segunda_janela(3LINESJ)
*      antes da chamada desta rotina.
*
*      RETORNO
*      Valor de Retorno da Rotina (tipo int):
*      Indica se houve erro (veja ERROS:).
*
*      RESTRICOES
*      - A janela raiz nao pode ser retirada.
*
*      ERROS
*      0 - Tudo OK.
*      1 - Janela indicada inconsistente.
*      4 - A janela raiz nao pode ser retirada.
*
*      VEJA TAMBEEM
*      Cria_arvore(3LINESJ), Cria_filho(3LINESJ), Retira_filho(3LINESJ),
*      Segunda_janela(3LINESJ).
*      -----*/

Ret_subarvore (sub)
lie_janela *sub;
{
    lie_janela *aux;

    /* i_printk ("retirando %lx\n", sub);          */
    if (Retira_filho(sub) == 0) return 0;
}

```

```

/* Se possui filhos ! */
aux=sub->prim_filho;
while (aux != FIM) {
    Ret_subarvore (aux);
    aux=aux->prox_irmao;
}
return (Retira_filho(sub));
} /* Fim da rotina Ret_subarvore */

```

```

/*-----*/
*      Lista_arvore - Lista a estrutura de janelas.
*
*      SINOPSE
*      #include <data_str.h>
*
*      void Lista_arvore (wid)
*      lie_janela *wid;
*
*      DESCRICAO
*      Esta rotina lista na interface de video do hospedeiro a
*      sub-estrutura de janelas atual a partir de no' correspondente
*      'a janela "wid" indicada. O proposito desta rotina e' o de
*      rastrear a estrutura, para a depuracao de aplicativos.
*
*      VEJA TAMBEEM
*      Cria_arvore(3LINESJ), Cria_filho(3LINESJ), Retira_filho(3LINESJ),
*      Ret_subarvore(3LINESJ).
*
*      EXEMPLO
*      #include <data_str.h>
*      extern lie_janela *raiz;
*
*      Lista_arvore (raiz); // Lista toda a estrutura de janelas //
*
*      -----*/

```

```

void Lista_arvore (sub)
lie_janela *sub;
{
    int i;
    lie_tipo_janela *tipoj;
    i=sizeof(lie_janela);
    i_printk ("***** %8lx ***** %d\n", sub,i );
    i_printk ("pai %lx", sub->pai);
    i_printk (" prim_filho %8lx", sub->prim_filho);
    i_printk (" prox_irmao %8lx", sub->prox_irmao);
    i_printk (" ant_irmao %8lx\n",sub->ant_irmao);
    i_printk ("tipo %8lx\n", sub->tipo);
    if (sub->tipo != 0) {
        tipoj=sub->tipo;
        i_printk (": borda >>%6b", tipoj->tipo_borda);
        i_printk (" z_janela %d", tipoj->z_janela);
        if (tipoj->texto == 0) i_printk (" texto:NULL\n");
        else i_printk (" texto:%a\n", tipoj->texto);
    }
    i_printk ("sentinela %d", sub->sentinela);
    i_printk (" xlo %d", sub->xlo);
    i_printk (" ylo %d", sub->ylo);
    i_printk (" xhi %d", sub->xhi);
    i_printk (" yhi %d\n", sub->yhi);
    i_printk ("plane_mask %x", (int)sub->plane_mask);
    i_printk (" event_mask %x", (int)sub->event_mask);
    i_printk (" cor_frente %x", (int)sub->cor_frente);
    i_printk (" cor_fundo %x\n", (int)sub->cor_fundo);
    i_printk ("x_cursor %d", sub->x_cursor);
    i_printk (" y_cursor %d\n", sub->y_cursor);
    i_printk ("arg_fonte %8lx", sub->arg_fonte);
    i_printk (" estado >%4b", sub->estado);
    i_printk (" base=%x\n\n", sub->base);
}

```

```

sub=sub->prim_filho;
while (sub != FIM) {
    Lista_arvore (sub);
    sub=sub->prox_irmao;
}
}

```

v_printk.c

```

/*****
 * Original: Cestano Traina Junior
 * DATA INICIAL: 21 de Maio de 1987
 * VERSAO ATUAL: 1.0
 *
 * NOTAS:
 *   v_printk (const char *formato, ... )
 *
 * MODIFICACOES:
 *   Data: 05/02/91      Sobre a versao:
 *   Rotina:
 *   Motivo: Correcao para aceitar constante Octal
 *           ..... Agma Juci M. Traina
 *****/
*/

/*-----*/
* i_printk - Saida formatada de dados no video do hospedeiro.
* SINOPSE
*   #include <data_str.h>
*
*   void v_printk (fmt, args)
*   char *fmt;
*   char args;
*
* DESCRICAO
*   A rotina v_printk envia caracteres para o video do proprio TMS.
*   Os caracteres indicados sao colocados no video, a partir da
*   posicao atual do cursor, usando o esquema de formatacao igual
*   ao da rotina "printf". Avanca o cursor para o final do
*   vetor escrito. Os caracteres sao escritos com o atributo de
*   cor corrente.
* VERA TMS
*   i_printk(3LINGSF), i_puts(3LINGSF), v_wputc(3LINGSF).
*
* EXEMPLO
*   v_printk ("Isto e' escrito no TMS: %d\n", 10);
*
/*-----*/

int v_printk (fmt, args)
char *fmt;
char args;
{
    extern int vputc();
    return _formatk(_vputc, fmt, &args);
}

/*-----*/
* Rotina interna usada para formatacao. Coloca no video um
* caracter pela rotina pchar().
*/
vputc(chr)
int chr;
{

```

```

pchar (chr);
if (chr == '\n') pchar(13); /* if chr==<LF> pchar(<CR>); */
return 0;
}

```

v_screen.c

```

/*****
 * Original: Cestano Traina Junior
 * DATA INICIAL: 2 de Setembro de 1987
 * VERSAO ATUAL: 1.0
 *
 * NOTAS:
 *   v_wputc (int caract)
 *   v_puts (const char *str)
 *   v_putk (const cadeia vetor)
 *   xy2car (x, y, lin, col)
 *   car2xy (lin, col, x, y)
 *   v_luga (lin, col)
 *   v_posic (lin, col)
 *
 * MODIFICACOES:
 *   Data: 18/02/91      Sobre a versao: 1.0
 *   Rotina: Todas
 *   Motivo: Adaptacao para o TMS. Criacao das rotinas:
 *           xy2car, car2xy, v_luga, v_posic.
 *   Autor: Agma Juci Machado Traina
 *****/
*/

#include "data_struct.h"

extern lie_janela *lie_jan_corrente;
extern lie_fonte lie_fonte_corrente;

/*-----*/
* v_wputc - Escreve caracter na janela corrente.
*
* SINOPSE
*   int v_wputc(c)
*   short c;
*
* DESCRICAO
*   A rotina v_wputc coloca o caracter "c" indicado no video, na
*   posicao atual do cursor, na janela corrente.
*
*   O caracter e' colocado na janela corrente, na posicao corrente
*   do cursor, utilizando-se os atributos de video corrente, tal como
*   cores de frente e fundo, e o fonte de caracteres corrente.
*
* RETORNO
*   Valor de Retorno da Rotina v_wputc (tipo int):
*   Codigo do caracter escrito.
*
/*-----*/

v_wputc(c)
short c;
{
    short col, lin;
    short _linhome, _colhome;
    short _linpe, _colpe;

    c &= 0xff;
    _linhome=0;
    _colhome=0;

```

```

/* Obtém dimensões da janela corrente como ((hi-lo+1) - (tamcar-1)), */
xy2car (lie_jan_corrente->xhi - lie_jan_corrente->xlo -
        lie_fonte_corrente.tam_car_x,
        lie_jan_corrente->yhi - lie_jan_corrente->ylo -
        lie_fonte_corrente.tam_car_y,
        &linpe, &colpe); /* ... em unidades de caracteres. */

/* Verificar se o cursor esta dentro da janela atual. */
if ( (c>=0x20) || (c==0) ) {
    v_lugc (&lin, &col); /* obter onde esta' o cursor */
    if (col < _colhome) col=_colhome;
    else if (col > _colpe) {
        col=_colhome;
        lin++;
    }
    if (lin<_linhome) {
        lin=_linhome;
        col=_colhome;
    }
    v_posic (lin, col);
}
pchar(c);
return c;
}

/*-----*/
*      v_wputs - Escreve vetor de caracteres na janela corrente.
*
*      SINOPSE
*      v_wputs(str)
*      char *str;
*
*      DESCRICAO
*      A rotina v_wputs coloca o vetor de caracteres "str" indicado no
*      video, a partir da posicao atual do cursor. Avança o cursor para
*      o final do vetor escrito. Os caracteres são escritos com o
*      atributo de video corrente, na janela corrente.
*
*      Os caracteres são colocados na janela corrente, na posicao
*      corrente do cursor, utilizando-se os atributos de video corrente, tal
*      como cores de frente e fundo, e o fonte de caracteres corrente.
*
*      RETORNO
*      Valor de Retorno da Rotina v_wputs (tipo int):
*      Constante zero.
*
/*-----*/

v_wputs(str)
char *str;
{
    while(*str)
        if (v_wputc (*str++) == '\n') v_wputc(0x0d);
    return 0;
}

/*-----*/
*      v_wputk - Escreve cadeia de caracteres na janela corrente.
*
*      SINOPSE
*      v_wputk(cad)
*      cadeia cad;
*
*      DESCRICAO
*
*      A rotina v_wputk coloca a cadeia de caracteres "cad" indicada
*      no video, a partir da posicao atual do cursor. Avança o cursor para
*      o final da cadeia escrita. Os caracteres são escritos com o atributo
*      de video corrente, na janela corrente.
*

```

```

*      Os caracteres são colocados na janela corrente, na posicao
*      corrente do cursor, utilizando-se os atributos de video corrente, tal
*      como cores de frente e fundo, e o fonte de caracteres corrente.
*
*      RETORNO
*      Valor de Retorno da Rotina v_wputk (tipo int):
*      Constante zero.
*
/*-----*/

v_wputk(str)
cadeia str;
{
    register short i;
    i=*(++str);
    while(i--)
        if (v_wputc (*(++str)) == '\n') v_wputc(0x0d);
    return 0;
}

/*-----*/
*      Esta rotina e' uma rotina interna. Obtém o endereço em unidades
*      de caracter a que corresponde uma dada posicao x,y na janela corrente.
*
xy2car (x, y, lin, col)
short x, y, *lin, *col;
{
    *lin=y/lie_fonte_corrente.tam_car_y;
    *col=x/lie_fonte_corrente.tam_car_x;
}

/*-----*/
*      Esta rotina e' interna. Obtém o endereço em unidades de pixel
*      relativo 'a posicao "x","y" na janela corrente, a partir de posicao de
*      linha e coluna.
*
car2xy (lin, col, x, y)
short lin, col, *x, *y;
{
    *y=lin*lie_fonte_corrente.tam_car_y;
    *x=col*lie_fonte_corrente.tam_car_x;
}

/*-----*/
*      Esta rotina e' interna. Obtém a posicao corrente do cursor, em
*      unidades de linha e coluna.
*
v_lugc(lin, col) /* obter onde ficara o cursor */
short *lin, *col;
{
    xy2car (lie_jan_corrente->x_cursor, lie_jan_corrente->y_cursor,
            lin, col);
}

/*-----*/
*      Esta rotina e' interna. Posiciona o cursor em unidades de linha e coluna
*
v_posic(lin, col)
short lin, col;
{
    car2xy (lin, col,
            &lie_jan_corrente->x_cursor, &lie_jan_corrente->y_cursor);
}

```

Listagem das Rotinas que Compõem o

Subsistema de Comunicacao com o Hospedeiro

run_host.c

```

/*-----
*-----      TMS34010 - LYN      -----
*-----
*
* Este Módulo deve ser compilado para rodar em maquina IBM_PC.
* Esta' preparado para utilizar compilador ASTEC C V4.1
*
* Aqui estao as rotinas que efetuam a comunicacao com o host.
* Sao as que devem ser executadas pelo GSP
*-----*/

#define DADO_PRONTO 1
#define DADO_ACEITO 0

#define ARQINICIO 1
#define ARQPROXIMO 2
#define LEARQ 3
#define ESCARQ 4
#define ABREARQ 5
#define FECHARQ 6
#define POSICIONARQ 7
#define LEINAGEM 8
#define CONTINUA 9

/* IBM addresses */
#define IHADDH 0xC0007F00 /* Address high word */
#define IHADDL 0xC0007E00 /* Address low word */
#define IHCTRL 0xC0007D00 /* Host control register */
#define IHDATA 0xC0007000 /* Host data register */

#define HOST_CONTROL_HIGH 0xC0000100

typedef struct { /* Estrutura do cabecalho de arquivo */
    int fattr; /* Attribute (1 byte) */
    unsigned int ftime; /* Time stamp */
    unsigned int fdate; /* Date stamp */
    long fsize; /* File size in bytes */
    unsigned char fname[15]; /* File name */
    unsigned char dummy; /* Para alinhar estrutura */
} ArqSpec;

unsigned long controle_teclado, /* Espec. qual o estado do dado (ACEITO ou PRONTO */
                dado_teclado, /* Qual e' o dado (caracter) */
                controle_video,
                dado_video,
                controle_disco, /* Espec. qual e' a operacao a ser feita */

/*
                dado_disco, /* Quais atributos */
                endereco_disco, /* Qual o cabecalho (FCB) */
                auxiliar_disco; /* Nome do arquivo (specname) */

/*-----
Esta rotina e' chamada pelo SUBR para controlar a comunicacao

```

```

do host IBM-PC com o TMS, logo apos carregar o programa do TMS.
-----*/
run_host(endereco)
unsigned long endereco;
{
    /* Inicializacao dos enderecos de comunicacao */
    controle_teclado=endereco;
    dado_teclado=endereco+16;
    controle_video=endereco+32;
    dado_video=endereco+48;
    controle_disco=endereco+64;
    dado_disco=endereco+80;
    endereco_disco=endereco+96;
    auxiliar_disco=endereco+128;

    /* Criacao do ambiente de comunicacao */
    ambiente();
}

#define COR_BORDA 0x26
#define COR_JANELA 0x07
#define COR_FLAGS 0x15

/*-----
Esta rotina cria o ambiente de comunicacao, que contem tres janelas,
a de cima, que mostra o que e' recebido do TMS, a de baixo que
registra o que o host IBM-PC esta' enviando para o TMS, e uma
pequena, que indica qual o estado do teclado (ON/OFF).
-----*/
ambiente()
{
    unsigned int c, i;
    int estado_teclado;

    v_getc(); /* Aguarda uma tecla digitada para entrar no ambiente */
    v_clear(' ');
    v_wniveis(3); /* abre tres janelas */
    v_setatr(COR_JANELA);

    /* Criacao da janela de recepcao (TMS -> PC) */
    v_wnivatual(0);
    v_wdef(0,0,13,78);
    v_wmold(2,COR_BORDA);
    v_wclear(' ');

    /* Criacao da janela de transmissao (PC -> TMS) */
    v_wnivatual(1);
    v_wdef(15,0,22,78);
    v_wmold(2,COR_BORDA);
    v_wclear(' ');

    /* Criacao da janela de flags de operacao */
    v_wnivatual(2);
    v_wdef(14,40,14,70);
    v_setatr(COR_FLAGS);
    v_wclear(176);
    v_wposic(0,0); v_wprintrk("Teclado TMS ");
    estado_teclado=1; /* Teclado ON */

    /* Colocacao dos titulos das janelas */
    v_setatr(COR_BORDA);
    v_posic(0,10); v_printrk("\xB5 Recebendo do TMS \xC6");
    v_posic(14,10); v_printrk("\xB5 Transmite para o TMS \xC6");
    v_setatr(COR_JANELA);

    while(1) {
        /* Enviar e tratar dado digitado no teclado */
        if ((c=v_poll()) != -1) { /* Retorna -1 qdo nao tem caracter teclado */

```

```

/* Trata ^S e ^Q */
if (c == 0x13) while (v_getc() != 0x11);

if (c == 27) { /* Escape alterna teclado ON/OFF */
    c=v_getc(); /* Retira Caractere <ESCAPE> */
    v_wmvatual (2);
    v_setatr (COR_FLAGS);
    v_wposic (0,0);
    if ( (estado_teclado = (estado_teclado==0))--1)
        v_wprintk ("Teclado TMS ");
    else
        v_wprintk ("Teclado LOCAL");
    v_setatr (COR_JANELA);
}
else { /* Nao e' caracter <ESCAPE> */
    if (estado_teclado == 1) {
        if ( tms_send_poll() == 0 ) {
            tms_sendc (c=v_getc());
            v_wmvatual (1); v_wputc (c);
        }
        else
            v_getc(); /* TMS nao quer o caracter */
    }
    else {
        c=v_getc();
        if (c==26) {
            v_wmvatual(1);
            v_wpop();
            v_wposic(15,0);
            exit();
        }
    }
}
} /* Fim do if (v_pool()) */

/* Verificar e tratar Dado enviado pelo TMS */
c=tms_receive_poll();
if (c != 0) {
    v_wmvatual (0);
    v_wputc (tms_receivec());
}

/* Verificar e Tratar Solicitacao de operacao em disco pelo TMS */
if (hi_peek (controle_disco) != DADO_ACEITO) {
    v_wmvatual(1);
    tms_disco();
}
} /* Fim do While(1) */
return ;
}

/*-----
Esta rotina transmite um caracter do TMS e passa para o TMS.
Caso exista um caracter anteriormente transmitido ainda nao aceito
pelo TMS, aguarda-se pela rececao do mesmo.
-----*/
tms_sendc (caract);
unsigned char caract;
{
    short caract;
    while (hi_peek(controle_teclado) != DADO_ACEITO); /* Espera TMS aceitar
o dado */
    hi_poke (dado_teclado, caract);
    hi_poke (controle_teclado, DADO_PRONTO);
    return 0;
}

/*-----

```

```

Esta rotina recebe do TMS um caracter. Caso nao exista um
caracter recebido, espera-se ate' que haja um caracter enviado
pelo TMS. O valor de retorno da rotina e' o valor ASCII do
caracter recebido.
-----*/
tms_receivec()
{
    int caract;
    while (hi_peek(controle_video) != DADO_PRONTO); /* Espera TMS enviar o
dado */
    caract = hi_peek (dado_video);
    hi_poke (controle_video, DADO_ACEITO);
    return caract;
}

/*-----
Esta rotina verifica se existe um caracter transmitido pelo TMS
que nao tenha sido aceito pelo TMS. Caso exista um caracter
anteriormente transmitido ainda nao aceito pelo TMS, retorna-se o
codigo do caracter. Caso nao haja caracter nao aceito, retorna-se ZERO.
-----*/
tms_send_poll()
{
    if (hi_peek (controle_teclado) == DADO_PRONTO)
        return hi_peek (dado_teclado);
    return 0;
}

/*-----
Esta rotina verifica se existe um caracter enviado pelo TMS.
Caso nao exista um caracter enviado, retorna-se ZERO, caso contrario
o valor de retorno da rotina e' o valor ASCII do caracter enviado.
Note-se que esta rotina nao aceita o caracter, o que deve ser feito
pela rotina tms_receivec().
-----*/
tms_receive_poll()
{
    if (hi_peek (controle_video) == DADO_PRONTO)
        return hi_peek (dado_video);
    return 0;
}

/*-----
* Rotina Para Acesso a DISCO *
-----*/

/*-----
Esta rotina recebe do TMS a solicitacao de uma operacao em disco,
e a executa completamente.
-----*/
tms_disco()
{
    int noper, i, j;
    ArqSpec cabecalho;
    long end transf;
    unsigned int aux1, aux2, *auxp;
    unsigned char *auxc;
    static unsigned char specname[130]={127,0}, buffer[1024];
    int atributos, modo, num, old_host_control_high, nlin, ncol;

    /* Definicao dos enderecos dos registros de interface do GSP */
    unsigned int *hctrl=(unsigned int *) IHCTRL;
    unsigned int *haddl=(unsigned int *) IHADDL;
    unsigned int *haddh=(unsigned int *) IHADDH;
    unsigned int *hdata=(unsigned int *) IHDATA;

    noper = hi_peek(controle_disco);

```

```

switch (noper) {
case ARQINICIO:
    atributos = hi_peek (dado_disco);

    /* Obter o endereco onde esta a especificacao do nome do arquivo */
    aux1 = hi_peek (auxiliar_disco);
    aux2 = hi_peek (auxiliar_disco+16);
    end_transf=((long)aux2<<16) | ((long)aux1);

    /* Obter a especificacao */
    aux1 = hi_peek (end_transf);
    specname[1]=(unsigned char) (0x00FF & (aux1>>8));
    for (i=0, j=2; i< (specname[1]+1)/2; i++) {
        end_transf+=16;
        aux1 = hi_peek (end_transf);
        specname[j++] = (unsigned char) (aux1 & 0x00FF);
        specname[j++] = (unsigned char) (0x00FF & (aux1>>8));
    }

    /* Posicionar no primeiro arquivo em disco que atende a specname */
    i=v_arqinicio (specname, atributos, &cabecalho);
    if (i != 0) { /* Deu ERRO */
        hi_poke(dado_disco, i);
        hi_poke(controle_disco, DADO_ACEITO);
        break;
    }
    hi_poke(dado_disco, 0); /* Devolve nao deu erro! */

    /* Transferir o cabecalho obtido */
    /* Obter o endereco para onde transferir a espec. do arquivo */
    aux1 = hi_peek (endereco_disco);
    aux2 = hi_peek (endereco_disco+16);
    end_transf=((long)aux2<<16) | ((long)aux1);
    auxp=(unsigned int *) &cabecalho;
    /* transferir... */
    for (i=0; i< (sizeof(ArqSpec)+1)/2; i++) {
        hi_poke (end_transf, (int) (*auxp));
        auxp++;
        end_transf+=16;
    }
    hi_poke (controle_disco, DADO_ACEITO);
    break;

    /****** ----- */

case ARQPROXIMO:
    /* Posicionar no arquivo seguinte em disco que atende ao
       specname anteriormente usado */
    i=v_arqproximo (&cabecalho);
    if (i != 0) { /* Deu ERRO */
        hi_poke(dado_disco, i);
        hi_poke(controle_disco, DADO_ACEITO);
        break;
    }
    hi_poke(dado_disco, 0); /* Devolve nao deu erro! */

    /* Transferir o cabecalho obtido */
    /* Obter o endereco para onde transferir a espec. do arquivo */
    aux1 = hi_peek (endereco_disco);
    aux2 = hi_peek (endereco_disco+16);
    end_transf=((long)aux2<<16) | ((long)aux1);
    auxp=(unsigned int *) &cabecalho;
    /* transferir... */
    for (i=0; i< (sizeof(ArqSpec)+1)/2; i++) {
        hi_poke (end_transf, (int) (*auxp));
        auxp++;
        end_transf+=16;
    }
    hi_poke (controle_disco, DADO_ACEITO);
    break;
}

```

```

/****** ----- */

case LEARG:
    j=hi_peek(dado_disco); /* indice do arquivo (file) */
    num=hi_peek(auxiliar_disco); /* numero de bytes a transferir */

    /* Obter o endereco de onde ler os dados */
    aux1 = hi_peek (endereco_disco);
    aux2 = hi_peek (endereco_disco+16);

    /* ----- transferir "num" bytes do arquivo "j" no endereco
       "aux2|aux1" ----- */

    /* Ativa auto incremento na escrita */
    old_host_control_high = hi_peek (HOST_CONTROL_HIGH);
    hi_poke(HOST_CONTROL_HIGH, 0x0800 | old_host_control_high);
    /* 0x0800 Increment on write cycles */

    /* Acerta o endereco dos registros de enderecamento */
    *haddl = aux1;
    *haddh = aux2;

    /* transfer the buffer to the GSP address space */
    aux1=0;
    while (num) {
        /* Le os dados e transfere para o GSP */
        aux2 = (num < 1024) ? num : 1024;
        num -= aux2;

        /* Le ate' 1024 bytes de cada vez */
        aux1 += read (j, buffer, aux2);
        aux2 >= 1;
        auxp=(unsigned int *) &buffer[0];
        for (i=0; i< aux2; i++)
            *hdata=(int) (*auxp++);
    }

    /* ----- Transferencia completada. ----- */
    hi_poke (HOST_CONTROL_HIGH, old_host_control_high);
    hi_poke(dado_disco, aux1);
    hi_poke(controle_disco, DADO_ACEITO);
    break;

    /****** ----- */

case ESCARG:
    j=hi_peek(dado_disco); /* indice do arquivo (file) */
    num=hi_peek(auxiliar_disco); /* numero de bytes a transferir */

    /* Obter o endereco de onde ler os dados */
    aux1 = hi_peek (endereco_disco);
    aux2 = hi_peek (endereco_disco+16);

    /* ----- transferir "num" bytes do arquivo "j" no endereco
       "aux2|aux1" ----- */

    /* Ativa auto incremento na leitura */
    old_host_control_high = hi_peek (HOST_CONTROL_HIGH);
    hi_poke(HOST_CONTROL_HIGH, 0x1000 | old_host_control_high);
    /* 0x1000 Increment address before local read cycles */

    /* Acerta o endereco dos registros de enderecamento */
    *haddl = aux1;
    *haddh = aux2;

    /* transfer the buffer to the GSP address space */
    aux1=0;
    while (num) {
        /* Transfere os dados do GSP e escreve */
        aux2 = (num < 1024) ? num : 1024;

```



```

        num -= aux2;

        /* Le ate' 1024 bytes de cada vez */
        auxp=(unsigned int *)buffer[0];
        for (i=0; i< (aux2>>1); i++)
            *auxp++=(int) *hdata;
        aux1 += write (j, buffer,aux2);
    }

    /* ---- Transferencia completada. ---- */
    hi_poke (HOST_CONTROL_HIGH, old_host_control_high);
    hi_poke(dado_disco, aux1);
    hi_poke(controle_disco, DADO_ACEITO);
    break;

    /***** ----- */

case ABREARQ:
    modo = hi_peek (dado_disco);

    /* Obter o endereco onde esta a especificacao do nome do arquivo*/
    aux1 = hi_peek (endereco_disco);
    aux2 = hi_peek (endereco_disco+16);
    end_transf=((long)aux2<<16) | ((long)aux1);

    /* Obter a especificacao */
    aux1 = hi_peek (end_transf);
    specname[1]=(unsigned char) (0x00FF &(aux1>>8));
    for (i=0, j=2; i< (specname[1]+1)/2; i++) {
        end_transf+=16;
        aux1 = hi_peek (end_transf);
        specname[j++] = (unsigned char) (aux1 & 0x00FF);
        specname[j++] = (unsigned char) (0x00FF & (aux1>>8));
    }
    specname[specname[1]+2]=0; /* Finaliza a cadeia */

    /* Abrir o arquivo indicado */
    i=open(&specname[2], modo);
    hi_poke(dado_disco, i);
    hi_poke(controle_disco, DADO_ACEITO);
    break;

    /***** ----- */

case FECHARQ:
    j=hi_peek(dado_disco); /* indice do arquivo (file) */
    i=close(i);
    hi_poke(dado_disco, i);
    hi_poke(controle_disco, DADO_ACEITO);
    break;

    /***** ----- */

case POSICIONARQ:
    j=hi_peek(dado_disco); /* indice do arquivo (file) */
    aux1=hi_peek(endereco_disco); /* offset low */
    aux2=hi_peek(endereco_disco+16); /* offset high */
    end_transf=((long)aux2<<16) | ((long)aux1); /* offset */
    modo=hi_peek(auxiliar_disco); /* como */

    i=lseek(j,end_transf, modo);
    hi_poke(dado_disco, i);
    hi_poke(controle_disco, DADO_ACEITO);
    break;

case LEIMAGEM:
    aux1 = hi_peek (dado_disco);
    auxc=(unsigned char *)buffer;

    /* Le a primeira linha: (nlin,ncol) */

```

```

        read (aux1, auxc,1);
        while ( *auxc++ >= ' ' )
            read (aux1, auxc,1);
        sscanf(buffer,"%(&d,&d)",&nlin,&ncol);

        hi_poke (endereco_disco+16L, nlin);
        hi_poke (endereco_disco, ncol);
        hi_poke (controle_disco, DADO_ACEITO);

        for (i=0; i<nlin; i++) {
            for (j=0; j<ncol; j++) {
                /* Pular Separadores */
                read (aux1, buffer,1);
                while ( (*buffer < '0') || (*buffer > '9') )
                    read (aux1, buffer,1);

                /* Ler valor do Ponto */
                aux2=0;
                while ( (*buffer >= '0') && (*buffer <= '9') ) {
                    aux2=aux2*10+buffer[0];
                    read (aux1, buffer,1);
                }
                while (hi_peek(controle_disco) != CONTINUA); /* Espera */
                hi_poke (dado_disco, aux2);
                hi_poke (controle_disco, DADO_ACEITO);
            }
        }
        break;

    } /* Fim do Switch */
    return 0;
}

```

sdblr.c

```

/*****
/* SDB340 Standalone COFF loader      (c)1986 Texas Instruments Inc. */
/* Comunicacao com Host                LIE 1991 ... Agma J. M. Traina */
/*****
/* Titulo: SDBLR                                L */
/*
/* Carrega e Executa um arquivo COFF esperando para gerenciar a
/* comunicacao com o host
/*
/* Usa-se: SDBLR [ COFF Filename ]
/*
/*
/*****

#include "stdio.h"

#define TRUE      1
#define FALSE     0

#define VERSION  "1.00022287"

/* IBM addresses */
#define IHADDH    0xC0007F00 /* Address high word */
#define IHADDL    0xC0007E00 /* Address low word */
#define IHCTRL    0xC0007D00 /* Host control register */
#define IHDATA    0xC0007000 /* Host data register */

/* Host control register bit masks */
#define HLT       0x8000 /* Halt GSP */
#define CF        0x4000 /* cache flush */

```

```

#define IBL      0x2000    /* lower byte last */
#define INCR     0x1000    /* Increment on read cycles */
#define INCW     0x0800    /* Increment on write cycles */
#define NMIMODE  0x0200    /* NMI mode (no stack usage) */
#define NMI      0x0100    /* Non maskable int */
#define INTOUT   0x0080    /* MSG Interrupt out */
#define MSGOUT   0x0070    /* Message out */
#define INTIN    0x0008    /* MSG Interrupt in */
#define MSGIN    0x0007    /* Message in */

#define DEFCTL   0xDA00    /* default setup for CONTROL reg */

#define GSP_PROGRAM_PC 0xFFC00001

#define NMI_LOW  0xFFFFFEE0 /* NMI vector address */
#define NMI_HIGH 0xFFFFFEF0

#define SRAM_ADR 0x04000000 /* address of shadow ram switch */

#define HOST_CONTROL_HIGH 0xC0000100
#define HOST_CONTROL_LOW  0xC00000F0

/* Optional file header is always 28 for GSP linker */
#define OPT_FILE_HEADER_LENGTH 28

/* global definitions */
unsigned int *hctrl;
unsigned int *haddl;
unsigned int *haddh;
unsigned int *hdata;

static unsigned int gsp_program[] = {
    0x0360, /* dint */
    0x0550, /* setf 16,0,0 */
    0x09E8, /* movi >10000,A8 */
    0x0001, 0x0000, /* into safe memory */
    0x4D0E, /* move A8,STK */
    0x4D0F, /* move A8,SP */
    0x09E8, /* movi >5A5A5A5A,A8 */
    0x5A5A, 0x5A5A, /* acknowledge host */
    0x0588, /* move A8,8HSTADRL,0 */
    0x00D0, 0xC000, /* by writing */
    0x2E08, /* srl 16,A8 */
    0x0588, /* move A8,8HSTADRH,0 */
    0x00E0, 0xC000, /* a unique number */
    0xC0FF, /* jrnc $ */
};

char pcname[6];
long endereco;

int load_sdb(char *obj_file);

main(argc,argv)
int argc;
char *argv[];
{
    unsigned long address;
    int cnt, timeout, ack;

    /* setup host port address addresses */
    hctrl=(unsigned int *)IHCTRL;
    haddl=(unsigned int *)IHADDL;
    haddh=(unsigned int *)IHADDH;
    hdata=(unsigned int *)IHDATA;
    strcpy(pcname,"LIE-IBM-PC");

    printf("\nSDB340 COFF Loader - %s\n",pcname);
    if (argc != 2) {

```

```

printf("Version %s(Modified LIE A.J.M.Traina 25/2/91)\n",VERSION);
printf("(c) Copyright 1987, Texas Instruments Inc.\n");
printf("\nusage: sdb1 [filename]\n");
}
else {
    /* initialize host control register */
    *hctrl=(unsigned int)DEFCTL;

    ack = FALSE;
    timeout = 18;

    while ((ack == FALSE) && (--timeout)) {
        halt_gsp();
        sram_on();

        /* load TMS34010 program into SDB memory */
        address = GSP_PROGRAM_PC;
        for (cnt = 0; cnt < sizeof(gsp_program) * 2; cnt++) {
            hi_poke(address,gsp_program[cnt]);
            address+=0x10;
        }

        /* execute the program */
        exe_nmi(GSP_PROGRAM_PC);
        halt_gsp();

        if (*haddl == 0x5A5A) {
            ack = TRUE;
            cache_on();
            if (load_sdb(argv[1]) == 0) {
                /* printf ("Endereco de COM: %lx\n", endereco ); */
                run_host (endereco);
            }
        }
        if (ack == FALSE)
            printf("[ No Response from SDB ]\n");
    }

    /*
     * Halt the GSP
     */
    halt_gsp()
    {
        unsigned int data;

        data = *hctrl; /* read host control register */
        data |= HLT; /* set the halt bit */
        data |= CF; /* cache flush */
        *hctrl = data;
    }

    /*
     * turn cache on
     */
    cache_on()
    {
        unsigned int data;
        unsigned int mask;

        data = *hctrl; /* read host control register */
        mask = CF;
        data |= ~mask; /* set cache on */
        *hctrl = data;
    }
}

```

```

/*
 * Turn shadow ram on
 */
sram_on()
{
    unsigned long address;

    address=(unsigned long)SRAM_ADR;
    hi_poke(address,0x0001); /* turn on shadow ram */
}

exe_nmi(pc)
unsigned long pc;
{
    unsigned int data,mask;
    unsigned long nmi_addr;

    nmi_addr = (long)NMI_LOW;
    data=((pc&0x0000FFFF));
    hi_poke(nmi_addr,data);

    nmi_addr = (long)NMI_HIGH;
    data = ((pc & 0xFFFF0000) >> 16);
    hi_poke(nmi_addr,data);

    data = *hctrl; /* read host control register */
    mask = HLT; /* compliment of HLT will be reset mask */
    data ^= ~mask; /* reset the halt bit */
    data |= NMI; /* set NMI */
    data |= NMIMODE; /* set NMI mode = no stack usage */
    *hctrl = data; /* tell the GSP to go */
}

/*
 * Peek GSP memory indirect through Host Port
 */
unsigned int hi_peek(address)
unsigned long address;
{
    *haddl = (unsigned int)(address & 0x0000FFFF);
    *haddh = (unsigned int)((address & 0xFFFF0000) >> 16);
    return(*hdata);
}

/*
 * Poke GSP memory indirect through Host Port
 */
hi_poke(address,data)
unsigned long address;
unsigned int data;
{
    *haddl = (unsigned int)(address & 0x0000FFFF);
    *haddh = (unsigned int)((address & 0xFFFF0000) >> 16);
    *hdata = data;
}

error(str)
char *str;
{
    printf("[ Error: %s ]\n",str);
}

int load_sdb(obj_file)
char *obj_file;
{
    FILE *inp,*fopen();

    long ip; /* GSP program counter */
    long symbol_table_file_pos;

```

```

/*
 * Main Header structure
 */
struct {
    unsigned short f_magic; /* magic number */
    unsigned short f_nscns; /* number of section headers */
    long int f_tmdar; /* time data */
    long int f_symptr; /* pointer to symbol table */
    long int f_nsyms; /* no. of entries in symbol table */
    unsigned short f_opthdr; /* no. of bytes in optional header */
    unsigned short f_flags; /* describes type of OBJ file */
} fil_head, *fh; /* end of structure */

/*
 * Optional File Header
 */
struct {
    short magic; /* magic number */
    short vstamp; /* version */
    long int tsize; /* text size (bytes) */
    long int dsize; /* initialized data size (bytes) */
    long int bsize; /* uninitialized data size (bytes) */
    long int entry; /* entry point */
    long int text_start; /* base address of text */
    long int data_start; /* base address of data */
} op_fil_head, *ofh; /* end of structure */

/*
 * Section Headers
 */
struct sect_head {
    char s_name[8]; /* section name */
    long int s_paddr; /* physical address of section */
    long int s_vaddr; /* virtual address of section */
    long int s_size; /* size of sections */
    long int s_scnptr; /* pointer to raw data */
    long int s_relptr; /* ptr to relocation entries */
    long int s_lnnoptr; /* file ptr to line number entries */
    unsigned short s_nreloc; /* number of entries */
    unsigned short s_nlnno; /* number of line number entries */
    long int s_flags; /* flags equals a section type */
};

unsigned int int_count; /* character count for read */
long int offset; /* offset for seeks in file */
short i,ndx; /* counters for sect_head[] */
int old_host_control_high;

struct sect_head *sh,*calloc();
int no_suffix;
int k, wrd_strm[100];

inp = fopen(obj_file,"r"); /* open file for reading */
if (inp == NULL) {
    /* check for suffix. If none, try .out and .obj */
    no_suffix = TRUE;
    for (i= 0; obj_file[i]; i++)
        if (obj_file[i] == '.')
            no_suffix = FALSE;

    /* no suffix, add .OUT and try again */
    if (no_suffix) {
        obj_file[i] = '.';
        obj_file[i+1] = 'o';
        obj_file[i+2] = 'u';
        obj_file[i+3] = 't';
        obj_file[i+4] = 0;

        /* Check again to see if the load file is present. */
        inp = fopen (obj_file, "r"); /* OPEN THE FILE OBJ_FILE FOR

```

```

                                READ */
if (inp == NULL) { /* IS IT A GOOD OPEN? */
    /* i still points to the end of the original string */
    obj_file[i+1] = 'o';
    obj_file[i+2] = 'b';
    obj_file[i+3] = 'j';
    /* obj_file[i+4] still equals 0 */

    /* Check again to see if the load file is present. If the */
    /* file name had a suffix then this will be redundant */
    /* This is a little bit inefficient but this is not time */
    /* critical code and it makes for the placement of only */
    /* one call to the error routine */

    inp = fopen(obj_file,"r"); /* open .OBJ file for read */
    } /* end of if (inp == NULL) 2 */
    } /* end of if (no suffix) */
    } /* end of if (inp == NULL) 1 */

if (inp == NULL) { /* IS IT A GOOD OPEN */
    error("File not found");
    return (2);
}
else { /* good open */

    /* inp is now the FILE pointer of the file storing the load data */

    /* halt the GPS, flush the cache, and turn on increments on write */
    old_host_control_high = hi_peek (HOST_CONTROL_HIGH);
    hi_poke (HOST_CONTROL_HIGH, 0xC800 | old_host_control_high);
    /* 0xC800 --> 0x8000 Halt GSP
       0x4000 cache flush
       0x0800 Increment on write cycles */

    fh = ffil_head; /* assign pointer for main file header */
    ofh = fop_fil_head; /* assign pointer for optional header */
    int_count = 20; /* file header contains 20 bytes */

    if (fread(fh, int_count, 1, inp) == 0) { /* read main file header */
        error("Invalid COFF header"); /* IF ERROR ON READ PRINT ERROR */
        goto ERROR_EXIT;
    }

    if (fil_head.f_magic != 0x90) {
        error("File not COFF");
        goto ERROR_EXIT;
    }

    offset = (long)(fil_head.f_opthdr); /* set offset for file seek */
    if (offset == 0) { /* NOT EX IF NO OPTIONAL FILE HEADER */
        ip = 0; /* SET INSTR. PTR TO LOAD ADDRESS */
        printf("[ Unlinked Coff file: PC = 0 ]\n"); /* data not linked */
    }
    else {
        if (fil_head.f_opthdr != OPT_FILE_HEADER_LENGTH)
            goto ERROR_EXIT;

        if (fread(ofh, fil_head.f_opthdr, 1, inp) == 0)
            goto ERROR_EXIT;
        ip = op_fil_head.entry; /* set gsp PC */
        /* printf ("magic=%d \nversion=%d \ntext size=%d \n"
           "initialized data size=%d \n"
           "uninitialized data size=%d \n"
           "entry point=%lx \n"
           "text_start=%lx \n"
           "data_start=%lx \n", ofh->magic,
           ofh->vstamp, ofh->tsize, ofh->dsiz, ofh->bsize,
           ofh->entry, ofh->text_start, ofh->data_start); */

    } /* end else */
}

```

```

offset += 201; /* make offset relative to file */
int_count = 40; /* number of bytes to read in sect. header */
sh = calloc(40, fil_head.f_nscns); /* dynamically allocate memory */
ndx = 0;

/* endereco=op_fil_head.data_start; */

printf ("[ Loading: %s ]\n", obj_file); /* write message */
while (fil_head.f_nscns) { /* terminated by num. of sections */
    fil_head.f_nscns--;

    if (fseek (inp, offset, 0) == -1) /* seek to next section header */
        goto ERROR_EXIT;

    if (fread (sh+ndx, int_count, 1, inp) == 0) /* read section header */
        goto ERROR_EXIT;

    /* printk ("Name:%c%c%c%c%c%c%c%c:"
       " phaddr=%lx, vaddr=%lx, size=%ld, scnptr=%lx, \n"
       " relptr=%lx, lnnoptr=%lx, nreloc=%d, nlnno=%d, flags>>1b\n\n",
       sh[ndx].s_name[0], sh[ndx].s_name[1],
       sh[ndx].s_name[2], sh[ndx].s_name[3],
       sh[ndx].s_name[4], sh[ndx].s_name[5],
       sh[ndx].s_name[6], sh[ndx].s_name[7],
       sh[ndx].s_paddr, sh[ndx].s_vaddr, sh[ndx].s_size,
       sh[ndx].s_scnptr,
       sh[ndx].s_relptr, sh[ndx].s_lnnoptr, sh[ndx].s_nreloc,
       sh[ndx].s_nlnno, sh[ndx].s_flags);

    if (sh[ndx].s_name[0] == 'C' && sh[ndx].s_name[1] == 'O' &&
        sh[ndx].s_name[2] == 'M' && sh[ndx].s_name[3] == 0)
        endereco = sh[ndx].s_paddr;

    if (fseek(inp, sh[ndx].s_scnptr, 0) == -1) /* seek to raw data */
        goto ERROR_EXIT;

    if ((sh[ndx].s_vaddr & 0x0000000F)) /* bit add? */
        error("Relocation on bit boundary");
    else {
        /* set address in gsp to start loading section data */
        /* set up the address registers */
        *haddl = (unsigned int) sh[ndx].s_vaddr;
        *haddh = (unsigned int) (sh[ndx].s_vaddr >> 16);

        /* read the input section data 512 words at a time, dump it
           to the gsp in the same size chunks */
        sh[ndx].s_size >>= 4; /* get # words in input section */
        if (sh[ndx].s_scnptr == 0)
            sh[ndx].s_size = 0;

        while (sh[ndx].s_size) {
            /* read section data and dump to the gsp */
            int w_size; /* # bytes to write to gsp */

            /* get # bytes to write to gsp */
            w_size = (sh[ndx].s_size < 1664) ?
                sh[ndx].s_size << 1: /* read remainder if < 1664 words */
                3328; /* else read 1024 bytes */
            sh[ndx].s_size -= (w_size >> 1); /* update bytes left */

            /* read the buffer off of the file */
            if (fread (hdata, w_size, 1, inp) == 0)
                goto ERROR_EXIT;

        } /* end dump data to GSP */
    } /* end else */

    offset += 401; /* inc offset address by 40 */
    ndx += 1; /* inc index into memory */
} /* end of first while */

```

```

END_LOAD_COFF:
    fclose (inp); /* CLOSE THE FILE INP */
    hi_poke(HOST_CONTROL_HIGH,old_host_control_high);
    exe_nmi(ip);
    return(0); /* Loading done, NO errors */
} /* end of outer else */

ERROR_EXIT:
    fclose (inp); /* close the input file */
    /* return host control high register to original value */
    hi_poke (HOST_CONTROL_HIGH,old_host_control_high);
    return (1);
} /* end of loader */

```