

UNIVERSIDADE ESTADUAL DE CAMPINAS  
FACULDADE DE ENGENHARIA DE CAMPINAS  
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

INTERFACE DE ENTRADA PARA UM  
NÚCLEO GRÁFICO

42/81

POR : Roberto Pinto Martins  
ORIENTADOR: Prof. Mário Jino

Tese de mestrado apresentado a Fa-  
culdade de Engenharia de Campinas  
da Universidade Estadual de Campi-  
nas.

DEZEMBRO - 1981

UNICAMP  
BIBLIOTECA CENTRAL

## SUMÁRIO

O objetivo deste trabalho constitui-se de dois propósitos. O primeiro, é a definição de um modelo para uma interface de entrada de comandos/dados para o núcleo gráfico GKS, utilizável por dispositivos com características completamente distintas. O segundo, é a implementação do modelo apresentado, utilizando um terminal gráfico do tipo "line-drawing" com regeneração e cinco dispositivos de entrada gráfica.

## ABSTRACT

The objective of this thesis is twofold. Firstly, we define a model of an input interface for the Graphical Kernel System (GKS), which can be used for devices of completely distinct functional and physical characteristics. Next, an implementation of this model is developed for a line-drawing regenerative graphics terminal and five graphic input devices.

Aos meus pais,  
Aos meus amigos.

O que une verdadeiramente as pessoas  
Não é o laço consanguíneo  
Mas a alegria  
E o respeito à individualidade de cada um.

R.P.Martins

## ÍNDICE

Cap. 1. Introdução.....	1
Cap. 2. Dispositivos para Entrada de Informação Gráfica.....	5
2.1. Dispositivos Físicos de Entrada Gráfica.....	5
2.1.1. A "Lightpen".....	5
2.1.2. O "Joystick", O "Control Ball", e o "Mouse".....	6
2.1.3. O "Tablet".....	6
2.1.4. O Teclado .....	8
2.2. Dispositivos Lógicos de Entrada e a Informação Gráfica Gerada.....	8
2.2.1. O "Locator".....	8
2.2.2. O "Valuator".....	10
2.2.3. O "Button".....	10
2.2.4. O "Pick".....	10
2.2.5. O "String".....	11
2.3. Simulação de Dispositivos Lógicos de Entrada....	11
2.3.1. Simulação do "Locator".....	13
2.3.2. Simulação do "Valuator".....	13
2.3.3. Simulação do "Choice".....	13
2.3.4. Simulação do "Pick".....	14
2.3.5. Simulação do "String".....	14
Cap. 3. O Núcleo GKS .....	15
3.1. Os Níveis de Implementação.....	18
3.2. As Transformações de Coordenadas.....	18
3.3. O Conceito de Segmento.....	18
3.4. Os Níveis de Aparência no GKS.....	20
3.5. O GKS e as Saídas Gráficas.....	20

3.5.1. Os Atributos das Primitivas de Saída.....	21
3.5.2. O "Pick Identifier".....	21
3.6. O Conceito de "Workstation".....	22
3.6.1. Os Atributos da "Workstation".....	23
3.6.2. A Identificação dos Dispositivos de Entrada na "Workstation".....	24
3.7. O GKS e as Entradas Gráficas .....	25
3.7.1. O "Request".....	25
3.7.2. O "Sample".....	26
3.7.3. O "Event".....	26
3.8. A Fila de Eventos.....	26
3.8.1. A Inserção .....	27
3.8.2. A Recuperação .....	27
3.8.3. A Eliminação .....	28
3.9. As Funções Especiais de Entrada .....	28
Cap. 4. Um Modelo para o Processamento Gráfico Dependente do Dispositivo .....	30
4.1. Os Conceitos de Independência e Dependência do Dispositivo .....	30
4.2. O Monitor Gráfico .....	32
4.3. O Modelo Proposto .....	35
4.3.1. O Modelo Interno .....	35
4.3.2. O Modelo Externo .....	41
4.4. O Monitor Gráfico como parte do "Device Driver".	42
Cap. 5. Implementação de uma Interface de Entrada para uma "Workstation" Interativa.....	44

5.1. Considerações Gerais sobre a Implementação de um Processador de Entrada.....	44
5.1.1. A Distribuição do Processamento.....	44
5.1.2. A Continuidade Sintática.....	45
5.1.3. O Problema da Entrada Incorreta de Dados.	47
5.2. O Modelo de Visualização .....	50
5.2.1. A Visualização do Modo de Operação.....	52
5.2.2. A Visualização da Ativação do Dispositivo Lógico .....	52
5.3. A Simulação .....	53
5.4. A Entrada de Comandos/Dados.....	54
5.4.1. O Dispositivo "Locator".....	54
5.4.2. O Dispositivo "Valuator".....	54
5.4.3. O Dispositivo "Choice".....	55
5.4.4. O Dispositivo "Pick".....	55
5.4.5. O Dispositivo "String".....	56
5.4.6. O Procedimento para Entrada de Comandos/ Dados.....	56
5.4.7. O "Buffer" para Entrada de Comandos/Dados	56
5.5. O Comando de Sinalização do Operador.....	57
5.6. A Correção de Erros.....	59
5.7. A Transferência de Informação entre Dispositivo Lógico e Programa de Aplicação.....	60
5.7.1. A Informação de "Locator".....	60
5.7.2. A Informação de "Valuator".....	61
5.7.3. A Informação de "Pick".....	61
5.8. O Mapa de Correlação .....	61
5.9. O Decodificador.....	63
5.10. A Tabela de Estado.....	65

5.11. O Controle dos Dispositivos.....	66
5.12. A Transferência de Informação.....	66
5.13. O Pedido de Comandos/Dados.....	68
5.14. O "Buffer" de Eventos.....	69
5.15. O Terminal .....	71
Cap. 6. Considerações Finais .....	73
Bibliografia .....	75

## 1. INTRODUÇÃO

A Computação gráfica pode ser considerada como o conjunto de todos os meios utilizados para o tratamento de imagens por computador; ela envolve a geração, a representação, a manipulação, o processamento ou avaliação de imagens bem como sua associação com informações não gráficas armazenadas em arquivos do computador /1/. A computação gráfica engloba o uso de diversas técnicas; algumas são bem conhecidas e utilizadas; outras ainda são tópicos de pesquisa. A computação gráfica aparece em quase todos os campos nos quais o computador é usado; por exemplo, matemática, medicina, controle de processo, projeto assistido por computador, etc. /2/.

Este uso tão generalizado da computação gráfica se deve a dois fatores, segundo GILOI /1/:

1. Embora a computação gráfica não seja o único meio de representar a informação, ela é seguramente o único meio razoável de fazê-lo. Isto é reconhecido até mesmo por pessoas que nunca ouviram o termo computação gráfica, quando dizem: "Uma figura vale por mil palavras".
2. Tipos especiais de interação homem/máquina são podem ser conseguidas com o uso da computação gráfica. Em certos casos, não há maneira mais direta para comunicação homem/máquina do que a visualização fornecida pela computação gráfica.

Hoje em dia, o que há de "comum" entre os sistemas gráficos existentes, é a sua "total incompatibilidade". Ou seja, estes sistemas são dedicados a aplicações específicas e são altamente dependentes do hardware instalado. Diante da ampla diversidade de interesses dos usuários, e a variedade de dispositivos gráficos existentes, com poucas características em comum, perguntamos: É possível uma padronização gráfica? Na própria história da computação encontramos um precedente recente de padronização

no desenvolvimento de uma linguagem não-gráfica, o FORTRAN. FOLEY /4/ afirma: "Considerável esforço tem sido devotado a padronização do FORTRAN para pacotes estatísticos e para compartilhamento de programas desenvolvidos para aplicações em engenharia. Como consequência, programas não-gráficos são rotineiramente compartilhados por instalações com hardware distintos, apesar do custo representado pelo esforço de adaptação e pela perda de eficiência dos programas. Esforço similar tem sido feito para a padronização gráfica".

Trabalhos recentes discutindo as vantagens e desvantagens da padronização gráfica podem ser encontrados na literatura /3,4,5/. FOLEY /4/ descreve as três principais vantagens que um sistema gráfico padronizado poderia trazer.

#### 1. INDEPENDÊNCIA DO DISPOSITIVO

Uma instalação gráfica poderia utilizar vários dispositivos (como "plotters", "displays"), com características totalmente diferentes, para o mesmo programa de aplicação.

#### 2. PORTABILIDADE DO PROGRAMA

Portabilidade ou independência do computador, é a vantagem mais significativa da padronização; as aplicações gráficas podem ser transportadas sem muito esforço de uma instalação para outra. Assim, programas gráficos do PDP-10 e que fazem uso do terminal GT40 poderiam ser transportados para o CDC 7600, para fazer uso do terminal IMLAC. Neste exemplo, estão ilustradas não somente a portabilidade do programa mas também a independência do dispositivo. Em termos práticos, um programa pode ser chamado de PORTÁTIL se o custo (em esforço de programação, etc.) de modificação do mesmo para adaptá-lo a outro computador é menor do que o custo de reescrevê-lo.

### 3. PORTABILIDADE DO PROGRAMADOR

Isto implica na desnecessidade de se ensinar novas convenções, técnicas e conceitos a programadores deslocados de uma instalação para outra.

As desvantagens da padronização, também segundo FOLEY /4/, são muito parecidas com as desvantagens de se utilizar uma linguagem de programação não-gráfica padronizada pois, assim como os computadores, os "plotters" e terminais interativos também apresentam uma variedade enorme de características de arquitetura e desempenho. O uso de qualquer padronização gráfica ou pacote gráfico de alto nível, conduzirá a ineficiência gerada pelo não aproveitamento dos detalhes de arquitetura. Contudo, este foi exatamente o argumento utilizado contra o FORTRAN e outras linguagens de alto nível; entretanto, hoje são reconhecidas as vantagens proporcionadas pelas linguagens de programação de alto nível. Ainda estamos no início da aceitação do uso de pacotes gráficos padronizados, e uma realidade acerca deles é que "sempre existirão usuários descontentes - uns reclamarão de seu desempenho, outros das facilidades disponíveis e outros de ambos".

Duas propostas de padronização têm ganhado corpo nos últimos anos; uma é de origem americana, conhecida como CORE SYSTEM, foi projetada por um grupo da ACM ("Association for Computing Machinery, INC.") e a sua definição preliminar data de agosto de 1977 /6/; a outra é de origem alemã, conhecida como GRAPHICAL KERNEL SYSTEM (GKS), apresentado pelo NI ("Normung Institute") também em 1977.

Este trabalho tem dois objetivos. O primeiro, é a definição de um modelo para a interface de entrada de comandos/ dados para o GKS, utilizável por dispositivos com características completamente diferentes. O segundo, é a implementação deste modelo para um terminal gráfico do tipo "line-drawing" com regeneração, incluindo um dispositivo de entrada de cada classe lógica.

No Capítulo 2., descrevemos os dispositivos gráficos de entrada, apresentamos o conceito de dispositivo lógico e sugerimos simulações e critérios que tornarão mais eficiente a comu

nicação homem/máquina. No Capítulo 3., descrevemos sucintamente o GKS, considerando suas potencialidades de geração, representação e manipulação de objetos gráficos. No Capítulo 4., propomos o modelo geral da interface de entrada de comandos/dados. No Capítulo 5., abordamos itens relacionados à comunicação homem/máquina e apresentamos a implementação do modelo desenvolvido no Capítulo 4., adaptando-o às condições específicas do terminal utilizado. Finalmente, no Capítulo 6., tecemos algumas considerações finais sobre o trabalho desenvolvido.

Para encerrar esta introdução faremos algumas observações relativas aos termos técnicos presentes neste trabalho. À medida que uma área vai sendo desenvolvida, termos específicos desta área vão sendo criados, com o propósito de facultar a comunicação entre seus profissionais. Em nosso país, isto explica em parte a parafernália de termos técnicos importados, presentes em quase todas as áreas técnicas, e particularmente na área de computação. O fato de "ainda" não possuímos nossos próprios termos, não justifica as agressões cometidas por profissionais contranosso vernáculo (mesmo oralmente), por exemplo, consagrando anglicismos como "deletar", "debugar", etc. Com relação aos termos técnicos presentes neste trabalho, o critério de tradução adotado é descrito a seguir.

Desde que estamos trabalhando com uma proposta de padronização, estaríamos sendo incoerentes com a própria padronização caso traduzíssemos seus termos-chave. Por outro lado, só serão traduzidos aqueles termos que possuam correspondentes consagrados em nosso vernáculo, a fim de evitar a criação de termos talvez mesmo desprovidos de sentido. Os termos não traduzidos serão grifados com o propósito de indicar sua origem alienígena.

## 2. DISPOSITIVOS PARA ENTRADA DE INFORMAÇÃO GRÁFICA

Na comunicação homem/máquina, duas linguagens se distinguem: uma é a linguagem com a qual a máquina se comunica com o homem; a outra é a linguagem de comando /8/, através da qual o homem, utilizando dispositivos de entrada apropriados, informa à máquina suas intenções. Infelizmente, não existe uma metodologia sistemática para o projeto destas linguagens /8/; as técnicas existentes variam desde a utilização de MENUS até o reconhecimento de caracteres "on-line" /8/.

Este capítulo tem o objetivo de fornecer uma idéia de que sejam dispositivos físicos e lógicos de entrada gráfica e a sua importância como meio de comunicação com a máquina. Para isto, apresentamos os dispositivos mais usuais, o tipo de informação gerada através de cada um, e sugerimos métodos de simulação de dispositivos lógicos.

### 2.1. DISPOSITIVOS FÍSICOS DE ENTRADA GRÁFICA

A única função de um dispositivo de entrada gráfica é a de fornecer, de algum modo, uma indicação da informação gerada por seu intermédio. O computador pode usar esta informação como dado de entrada, e usá-la para controlar algum processo. A seguir, apresentaremos os dispositivos de entrada gráfica que têm sido utilizados com maior frequência em instalações gráficas.

#### 2.1.1. A LIGHTPEN

É o único dispositivo de entrada que permite ao usuário apontar diretamente um objeto na tela. A LIGHTPEN /1,8,9/ possui em uma de suas extremidades uma fotocélula que gera um sinal quando o feixe de elétrons cruza seu campo de visão. Este sinal é utilizado para identificar o objeto pela sua posição. Na Figura 2.1a, mostra-se o esquema de uma LIGHTPEN.

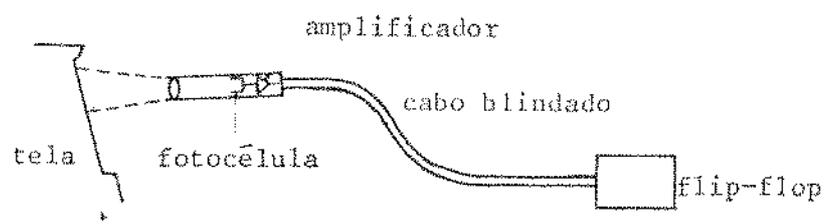
### 2.1.2. O JOYSTICK, O CONTROL BALL E O MOUSE

São todos dispositivos usados para controlar movimentos de cursor sobre a tela, ou seja, são basicamente dispositivos de posicionamento. O JOYSTICK /1,8,9/ é composto por uma alavanca que pode ser movida com dois graus de liberdade (para frente-para trás, para direita-para esquerda). Com movimentos de alavanca nas duas direções o cursor pode ser deslocado para qualquer ponto da tela. Um esboço deste dispositivo pode ser visto na Figura 2.1b. O CONTROL BALL /1,8,9/ é composto por uma bola que pode ser rodada com a mão em qualquer direção. O ângulo de rotação dá a direção de movimento do cursor. Um esboço de um CONTROL BALL pode ser visto na Figura 2.1c. CONTROL BALL e JOYSTICK são dispositivos muito usados em controle de tráfego aéreo. O MOUSE /1,8,9/ é composto por uma pequena caixa movimentada manualmente possuindo em sua parte inferior um CONTROL BALL ou duas rodas perpendiculares. Movendo o MOUSE através de uma superfície plana, o cursor pode ser movido como nos casos anteriores. Um esboço do MOUSE pode ser visto na Figura 2.1d.

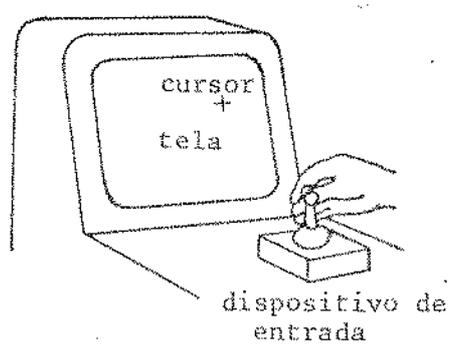
Os três dispositivos mencionados fornecem dois valores que geram as coordenadas do cursor. Desta forma, sempre que o cursor se encontrar em uma posição, suas coordenadas estarão disponíveis diretamente do estado do dispositivo, dado pelos dois valores acima.

### 2.1.3. O TABLET

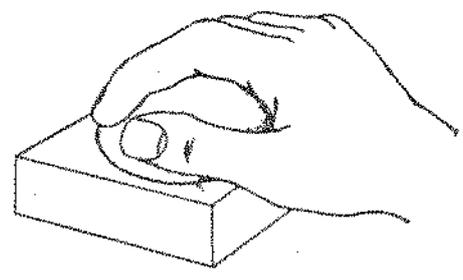
É uma superfície plana, usualmente separada do DISPLAY. Com auxílio do STYLUS, qualquer posição desta superfície poderá ser detectada. O STYLUS e o TABLET /1,8,9/ podem desempenhar as mesmas funções que o lápis e o papel desempenham para o desenhista. Por esta razão, seu principal uso tem sido em aplicações onde se deseja traçar características gráficas tais como, linhas de contorno, desenho de mapa, etc. Um esboço deste dispositivo pode ser visto na Figura 2.1e.



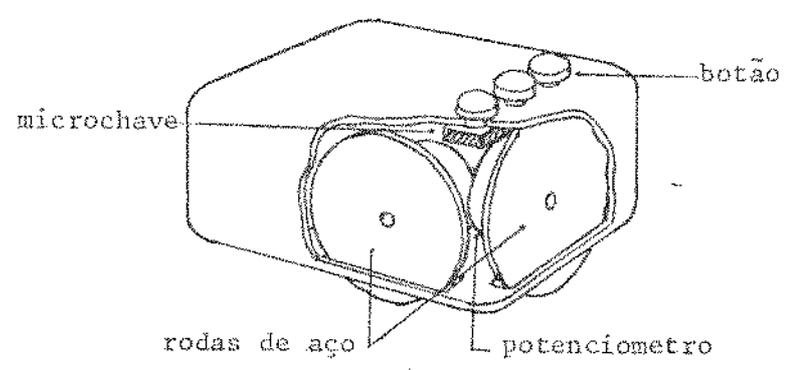
(a) "Lightpen"



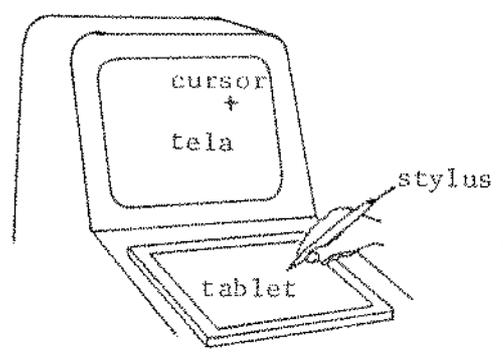
(b) "Joystick"



(c) "Control Ball"



(d) "Mouse"



(e) "Tablet"

FIG. 2.1 - Dispositivos de Entrada

#### 2.1.4. O TECLADO

Teclados /1,8,9/ são dispositivos que permitem ao usuário fornecer informação alfanumérica. Podem ser subdivididos em dois grupos: teclados alfanuméricos convencionais e teclados de funções programáveis. Ambos os grupos podem aparecer em um mesmo dispositivo físico, ou separadamente; e podem ou não fazer parte do DISPLAY. O que os distingue é o fato de os comandos serem dados com muito mais rapidez no segundo grupo pois a função de cada tecla é programada a priori.

#### 2.2. DISPOSITIVOS LÓGICOS DE ENTRADA E A INFORMAÇÃO GRÁFICA GERADA

O universo de informações geradas nos dispositivos de entrada é subdividido em seis subconjuntos, segundo DEECKER e PENNY /10/.

1. Posição-coordenada
2. Vetor posição-coordenada
3. Valor numérico
4. Identificador de função
5. Identificador de objeto
6. Texto

Estes seis tipos de informação podem ser gerados através de cinco classes de dispositivos lógicos: LOCATOR, VALUATOR, CHOICE, PICK e STRING. Os quatro primeiros foram definidos por FOLLEY e WALLACE /11/ e o quinto é acrescentado por razões de universalidade.

Um mapeamento relacionando os dispositivos lógicos e o tipo de informação gerada é visto na Figura 2.2. /9/.

Uma descrição dos dispositivos lógicos auxilia a compreensão desta classificação.

##### 2.2.1. O LOCATOR

Dispositivos desta classe são utilizados para indicar uma posição e/ou orientação. Dependendo da aplicação podem i

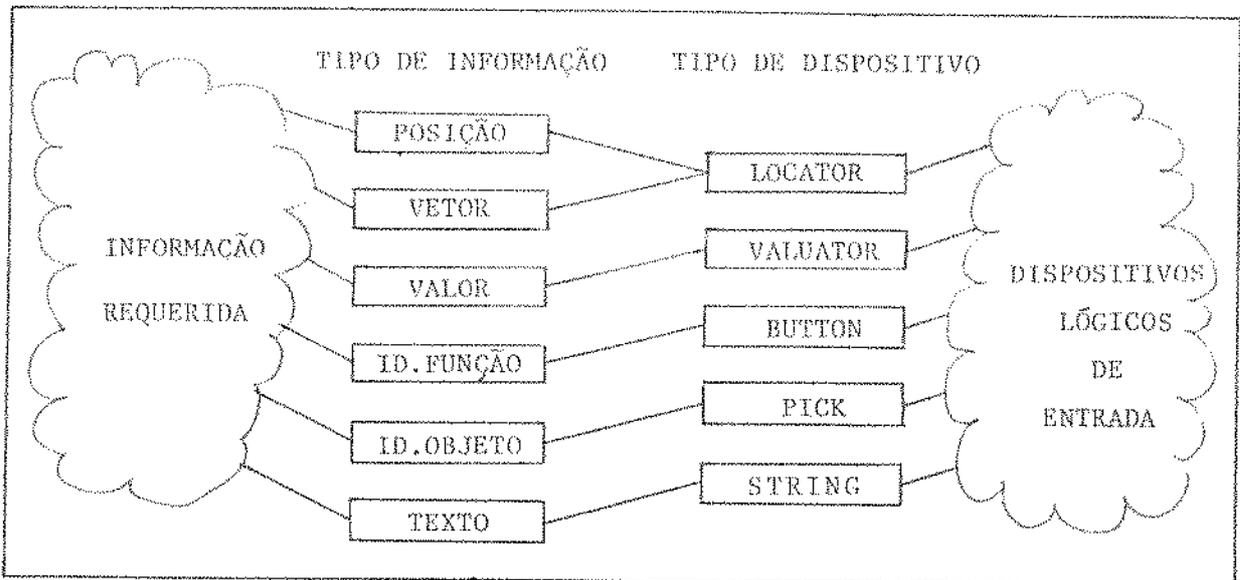


FIG. 2.2 - Mapeamento dispositivo-informação gerada

dentificar posições em uma, duas, três ou mais dimensões. No caso de navegação espacial, por exemplo, seis coordenadas são necessárias para especificar uma posição. Seus protótipos são o JOYSTICK, o CONTROL BALL, o MOUSE e o TABLET.

#### 2.2.2. O VALUATOR

Dispositivos desta classe são utilizados para retornar informações numéricas não relacionadas à imagem, e.g., o valor de um resistor, um ângulo de rotação, um fator de "zoom" etc. Seu protótipo é o potenciômetro.

#### 2.2.3. O BUTTON

Dispositivos desta classe são utilizados para selecionar ações a serem executadas. Seu protótipo é o teclado de funções programáveis e contém tipicamente de 16 a 32 teclas. Os BUTTONS não são necessariamente dispositivos autônomos, aparecem freqüentemente anexados a outros dispositivos. Por exemplo, uma chave em uma LIGHTPEN é freqüentemente usada para sinalizar ao sistema a confirmação de uma ação. Chaves no STYLUS de um TABLET são usadas com o mesmo propósito.

#### 2.2.4. O PICK

Esta classe destaca-se das quatro outras por atuar em informações já existentes em uma base de dados. São freqüentemente usados com imagens estruturadas com o propósito de identificar objetos na tela, e.g., uma linha, uma certa região, um resistor, etc. O programa, através do uso da linguagem de comando deve decidir o que fazer com o objeto apontado ou com a estrutura associada ao mesmo. A linguagem de comando pode ser implícita ou explícita. No caso de ser implícita, a decisão dependerá do estado atual (e possivelmente do estado anterior) do programa. Caso contrário deve ser tomada pelo próprio operador.

### 2.2.5. O STRING

Dispositivos desta classe são utilizados para entrar com informação de caráter alfanumérico. Em sistemas interativos, com uma linguagem de comando poderosa (incluindo facilidade de comandos HELP), seu uso é quase imperativo. Seu protótipo é o teclado convencional.

A Tabela 2.1. apresenta uma descrição sucinta dos dispositivos físicos de entrada mais usuais em sistemas de computação gráfica.

A classificação dos dispositivos em classes lógicas, isto é, de acordo com suas funções e não por suas idiossincrasias técnicas, tem implicação não apenas na metodologia de projeto mas também na padronização de sistemas gráficos, por permitir desenvolvimento de software independente do dispositivo. Em outras palavras, o programador de aplicação pode conceber seu programa através de rotinas que manipulem informação proveniente do LOCATOR, do VALUATOR, sem se preocupar com o dispositivo físico que está gerando as informações /1/.

### 2.3. SIMULAÇÃO DE DISPOSITIVOS LÓGICOS DE ENTRADA

O enquadramento de um dispositivo de entrada em uma certa classe não restringe a utilização deste dispositivo a esta classe em particular. Não é de se esperar que todo sistema gráfico possua em dispositivo físico (protótipo) para cada classe lógica; não obstante, as aplicações requeridas pelos usuários do sistema não levam em conta tal fato, exigindo informações não disponíveis nos dispositivos físicos existentes. Felizmente, este não é o tipo de restrição encontrado usualmente, pois podemos lançar mão de simulações e, dentro de certos critérios, os dispositivos simulados chegam a se adaptar ao operador até mesmo melhor do que os próprios dispositivos físicos.

FOLEY e WALLACE /11/ referem-se à simulação da seguinte forma: "A flexibilidade de se substituir um dispositivo físico por outro, apresenta a vantagem de experimentação e otimização em linguagens interativas com respeito ao aspecto visual e a con

	FUNCIONALIDADE		PRECISÃO	CUSTO			SATISFAÇÃO DO USUÁRIO		Considerações Especiais
	Classe do dispositivo	Classes da Informação	Resolução e Acurácia	Custo do Hardware	Desenvolvimento de Software	"Overhead" de Software	Tipo de Entrada	Realimentação	
"Tablet"	"Locator"	Posição, Vetor, Id.de função, Id.de objeto,	0,05" a 0,001" até $\pm$ 0,005"	Médio a Elevado	Médio	Baixo	Indireto (algumas vezes é direto e gráfico)	"Display" independente ou indicação da posição digital	Algumas unidades não se prestam para utilização interativa "on line".
"Joystick"	"Locator"	Posição, Vetor, Id.de objeto,	Resolução dependente do conversor A/D de $\pm$ 10% a 0,1%	Baixo a Médio	Baixo	Baixo	Indireto e Tátil	Tátil e/ou "display" independente	Grande variedade de tipos para ajustar-se à maioria das aplicações.
"Mouse e Control ball"	"Locator"	Posição, Vetor, Id.de objeto,	"Joystick" 10 - 1000 pontos distinguíveis de modo exato	Baixo a Médio	Baixo	Baixo	Indireto e Tátil	Tátil e/ou "display" independente	Posicionamento relativo de "mouse"
potenciometro	"Valuator"	Valor, Posicionamento em duas dimensões, se usado aos pares.	Semelhante à do "Joystick"	Baixo	Baixo	Baixo	Indireto e Tátil	Tátil e/ou "display" independente	Vários tipos de potenciômetros: tambor, deslizante, de alavanca.
Seletores de Funções	"Button"	Id.de função		Baixo a Médio	Baixo	Baixo	Indireto e Tátil	Visual (Posição da chave)	É possível a utilização de botões programáveis por software.
"Lightpen"	"Pick"	Id.de objeto Posição, Vetor, Id.de função	Dependente do "Display"	Médio	Alto	Médio a Alto	Gráfico e Direto	"Display" independente	Utilizável como apontador em "displays" com regeneração ou como locators em "displays" de tipo "raster-scan"

TAB. 2.1 - CARACTERÍSTICAS DOS DISPOSITIVOS GRÁFICOS DE ENTRADA

tinuidade tátil"; e acrescentam: "Enquanto qualquer dispositivo físico pode ser usado como qualquer dispositivo virtual, tais experimentações são necessárias porque dispositivos físicos não são necessariamente intercambiáveis e psicologicamente equivalentes".

A seguir, apresentaremos algumas simulações usuais ou úteis em algumas aplicações

### 2.3.1. SIMULAÇÃO DO LOCATOR

Esta classe encontra-se talvez mais freqüentemente disponível através da LIGHTPEN e o "tracking-cross" do que com seus próprios protótipos. O "tracking-cross" acompanha os movimentos da mão, e as coordenadas de seu centro podem ser obtidas diretamente de posições reservadas de memória. Quatro BUTTONS podem ser usados com a mesma finalidade. Cada BUTTON comanda o movimento do "tracking-cross" em uma direção (para baixo, para cima, para a direita, para a esquerda). Se a linguagem de comando é orientada para teclado, este método pode ser atrativo, pois preserva a continuidade tátil.

### 2.3.2. SIMULAÇÃO DO VALUATOR

Esta classe pode ser simulada por teclado, através de uma seqüência de dígitos. No entanto, a simulação por LIGHTPEN, através do uso de potenciometro deslizante ou potenciometro circular é elegante, natural e dá continuidade visual ao operador. Se a LIGHTPEN está sendo usada para simular outros dispositivos, fornece também continuidade tátil.

### 2.3.3. SIMULAÇÃO DO BUTTON

Esta classe é simulada freqüentemente pela LIGHTPEN em DISPLAYS com regeneração, ou por dispositivos da classe LOCATOR em "storage tube", ambas com o auxílio da técnica de MENU.

#### 2.3.4. SIMULAÇÃO DO PICK

Esta classe pode ser simulada usando dispositivos da classe LOCATOR, os quais controlam o movimento de um cursor. Posicionando o cursor próximo ao objeto, este pode ser identificado por software /1,12/. Se em uma aplicação, o usuário desejar apontar para o interior do objeto e não para o objeto, o uso de simulação é imperativo /12/.

WALLACE e FOLEY /11/ sugerem: "Três BUTTONS podem simular um PICK. Se cada entidade na tela for aumentando sucessivamente sua intensidade por um período, assim que a entidade a ser selecionada intensificar-se, o operador acionará o primeiro BUTTON. Acionando o segundo, a intensificação se dará na ordem inversa, com intervalos maiores. Assim que a entidade a ser identificada voltar a intensificar-se, o terceiro fará a identificação".

#### 2.3.5. SIMULAÇÃO DO STRING

A LIGHTPEN ou dispositivos da classe LOCATOR podem ser utilizados para simular esta classe. Esta simulação certamente consumirá muito tempo em aplicações reais e trará aborrecimentos ao operador.

GILOI /1/ faz uma advertência: "Em nossa opinião, "bons" sistemas interativos, aptos a realizarem trabalhos práticos, deverão ser equipados com, no mínimo três dispositivos:

1. LIGHTPEN (ou JOYSTICK em caso de "storage tube")
2. Teclado
3. Teclado de Funções Programáveis

Acreditamos que o TECLADO seja dispositivo compulsório em interações, tanto pela facilidade de entrada de dados quanto pela flexibilidade em dar comandos.

### 3. O NÚCLEO GRÁFICO GKS

Os objetivos e a motivação para a padronização de software gráfico são a portabilidade e a independência do dispositivo. Um sistema gráfico padronizado deve ser definido tendo como finalidade atender à uma vasta gama de usuários com diferentes objetivos, e ser facilmente entendido pelos programadores e usuários.

A padronização na área gráfica permitiria a produção de dados gráficos em um computador, e a sua saída sucessiva ou simultânea em diferentes dispositivos gráficos. A saída simultânea em diferentes dispositivos é importante quando é necessário trabalhar com diferentes seções de uma mesma imagem. Tão importante quanto esta independência de dispositivos de saída é a independência dos dispositivos de entrada. Isto é conseguido utilizando-se dispositivos lógicos de entrada, associados aos dispositivos físicos de entrada, como está descrito na Seção 2.2.

Neste capítulo estamos preocupados em fornecer uma idéia global das capacidades gráficas do GKS, e introduzir termos que serão utilizados nos próximos capítulos. A descrição do GKS pode ser encontrada em /7/.

O GRAPHICAL KERNEL SYSTEM (GKS) é uma proposta de padronização que consiste de um conjunto de funções para processamento de dados gráficos, independente dos periféricos utilizados, da linguagem de programação e das aplicações específicas. O modelo de camadas apresentado na Figura 3.1 mostra a utilização do GKS em um sistema gráfico /7/.

A Figura 3.2 mostra um sistema gráfico que tem o GKS para suporte básico de processamento gráfico. Cada uma das interfaces está descrita por DALTRINI, et al. /13/. Em particular, a interface de saída é objeto da tese de mestrado de MOLINARO /14/.

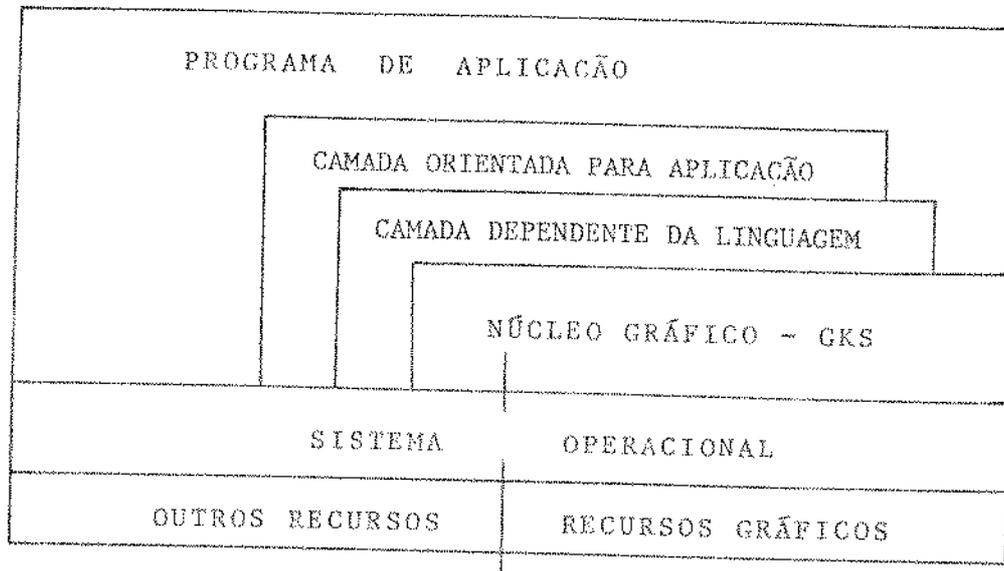


FIG. 3.1 - Modelo de camadas do GKS

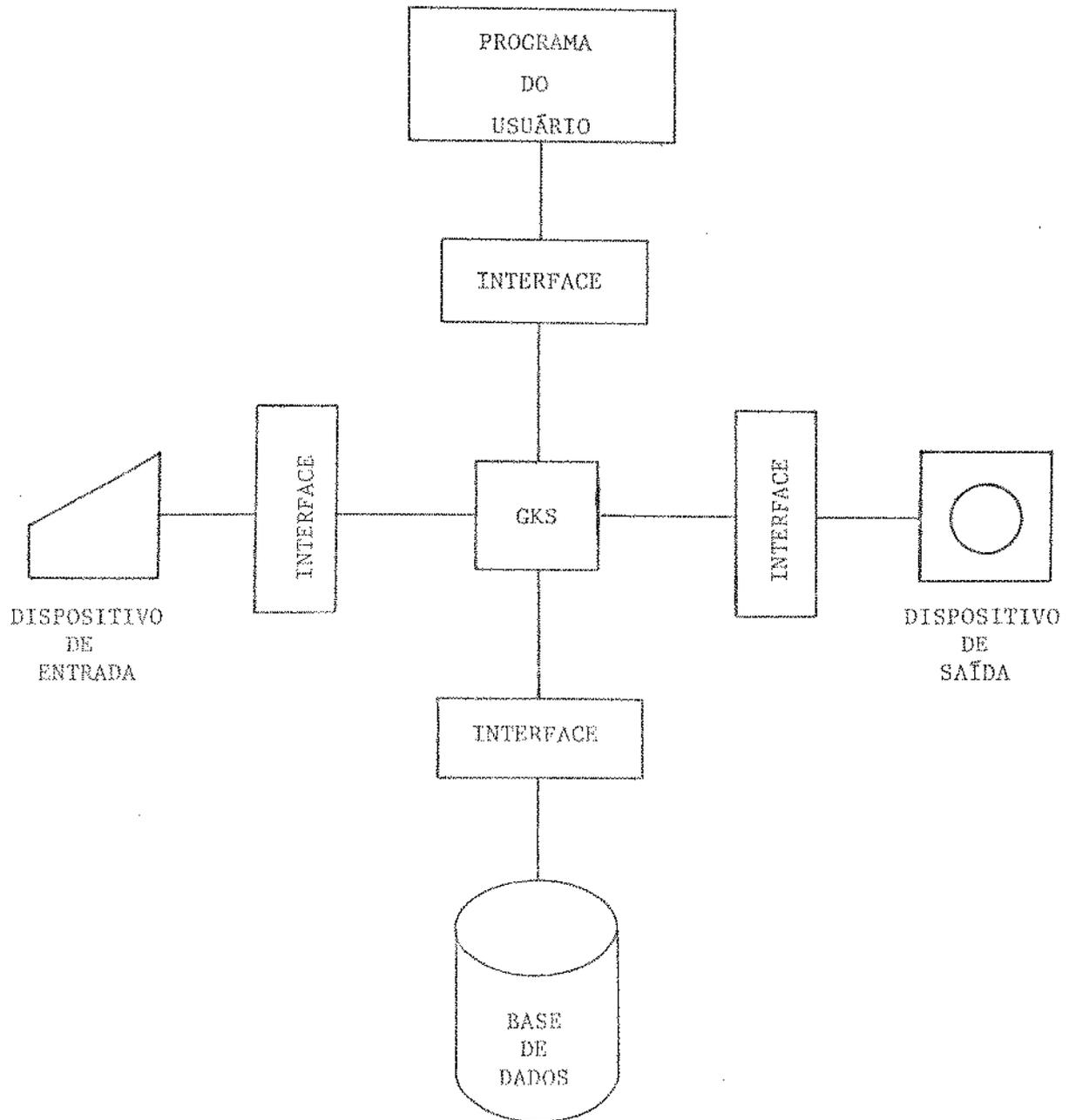


FIG. 3.2 - Um sistema gráfico baseado no GKS

### 3.1. OS NÍVEIS DE IMPLEMENTAÇÃO

O fato do GKS operar com uma grande variedade de dispositivos gráficos, desde simples "plotters" e "storage tube" até "displays" interativos, levou a adoção de seis níveis compatíveis e ascendentes de implementação /7/. Desta forma, uma instalação pode escolher o nível que melhor se adequa às suas necessidades, evitando penalidades de espaço e/ou tempo com a inclusão de capacidades desnecessárias. Estes níveis não serão descritos neste trabalho, bastando esclarecer que as características ora descritas correspondem ao nível seis.

### 3.2. AS TRANSFORMAÇÕES DE COORDENADAS

No GKS, as funções de saída têm características bidimensionais e entrada/saída podem referenciar uma ou mais WORKSTATIONS simultaneamente. A transformação de coordenadas ocorre em dois estágios. No primeiro, tem-se a transformação para cada primitiva, ou seja, coordenadas do mundo do usuário (WC) são mapeadas em uma representação interna ao GKS, as coordenadas normalizadas (NDC). No segundo, para cada WORKSTATION, as coordenadas normalizadas são mapeadas nas coordenadas do dispositivo (DC) /7/. A transformação de coordenadas pode ser vista na Figura 3.3.

### 3.3. O CONCEITO DE SEGMENTO

Uma característica peculiar ao GKS é a forma como este trata o segmento. Poderíamos definir o segmento como sendo a unidade básica manipulável de qualquer imagem. Através do segmento pode-se estruturar uma imagem em partes que a compõem, segmentos podem ser criados ou eliminados, nomes de segmentos podem ser mudados. Atributos de segmento podem ser dinamicamente modificados, ou seja, durante a execução do programa a prioridade do segmento pode ser alterada, segmentos podem ser feitos visíveis ou invisíveis, detectáveis ou não pela classe PICK, segmentos podem ser transformados e também podem mudar de estado (pisca/não pisca), com o objetivo de chamar a atenção do operador.

Segmentos podem também ser inseridos em outros seg

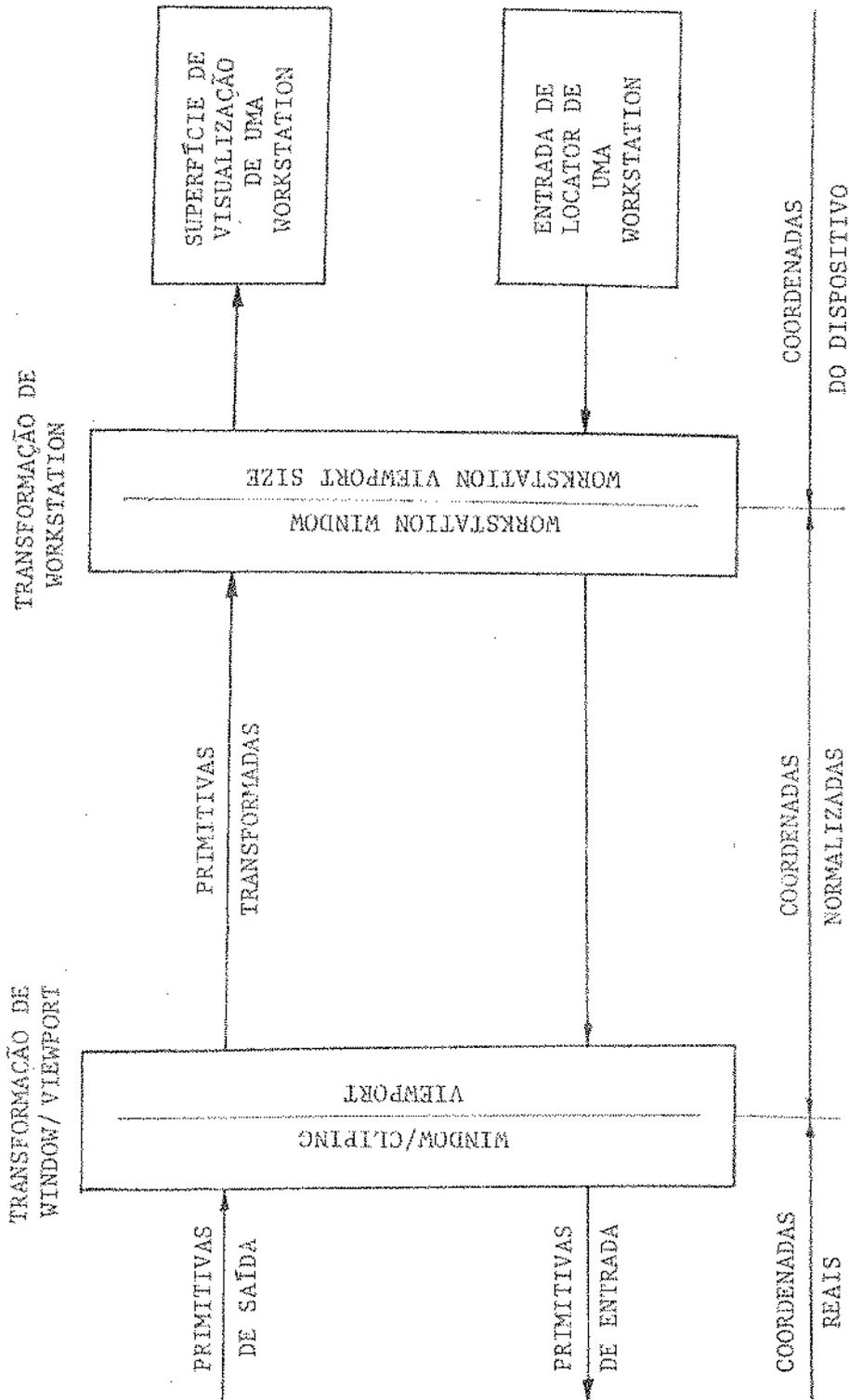


FIG. 3.3 - Transformações e Sistemas de Coordenadas no GKS

mentos e mostrados simultaneamente ou alternadamente em diferentes WORKSTATIONS. Com estas características, é possível editar arquivos gráficos em DISPLAYS interativos e obter uma cópia definitiva do arquivo editado, até mesmo em outra escala, em um PLOTTER como "hardcopy".

Todas estas possibilidades de manipulação são possíveis porque os segmentos são armazenados em um dispositivo virtual, a WORKSTATION SEGMENT STORAGE /7/. Contudo, primitivas gráficas, definidas a seguir, podem ser criadas fora de segmento e, neste caso, não serão armazenadas na WORKSTATION virtual, não sendo possível nenhuma manipulação com as mesmas. Desta forma, primitivas fora de segmento serão perdidas nos casos em que uma mudança na imagem torne necessário apagar todo o DISPLAY como ocorre em WORKSTATIONS do tipo "storage tube".

#### 3.4. OS NÍVEIS DE APARÊNCIA NO GKS

Como a transformação de coordenadas, a atribuição da aparência em uma primitiva é definida em dois estágios /15/. No primeiro, um atributo simbólico é associado à primitiva; no segundo, o atributo simbólico é mapeado nas capacidades da WORKSTATION determinando a aparência da primitiva. Ambos os estágios estão sob controle do programa de aplicação. Deste modo, o programa de aplicação pode, por exemplo, desenhar uma linha em cor vermelha num PLOTTER e a mesma pontilhada num DISPLAY sem a necessidade de gerá-la novamente para o DISPLAY:

#### 3.5. O GKS E AS SAÍDAS GRÁFICAS

As informações gráficas geradas pelo GKS são compostas por elementos gráficos básicos que podem ser usados para construir imagens. Estes elementos são chamados primitivas de saída e são decompostas em três primitivas para "line-drawing", uma para texto e duas para "raster", ou seja:

POLYLINE: Dada uma seqüência de pontos, gera um conjunto de

segmentos de reta conectados.

POLYMARKER: Dado um conjunto de pontos, gera marcas pré-definidas nestes pontos.

GENERALIZED: São funções gerais relacionadas com capacidades especiais das WORKSTATIONS como: desenho de curvas  
DRAWING "spline", arcos circulares e arcos elípticos.  
PRIMITIVE

TEXT: Dada uma posição, gera uma seqüência de caracteres a partir da mesma.

FILL AREA: Dado o número de pontos e suas coordenadas, um polígono é preenchido com uma cor uniforme.

PIXEL ARRAY: Definida a região e as cores, um "array" de "pixels" com cores individuais é desenhado.

### 3.5.1. OS ATRIBUTOS DAS PRIMITIVAS DE SAÍDA

São atributos simbólicos associados às primitivas de saída e utilizados em conjunto com os atributos das WORKSTATIONS, a fim de controlar a maneira pela qual as primitivas serão apresentadas. Os atributos são: PEN NUMBER, TEXT NUMBER, TEXT SIZE and SPACING, MARKER SIZE e PICK IDENTIFIER. Somente um atributo será discutido neste trabalho, o PICK IDENTIFIER, o qual está relacionado com o procedimento de detecção dos dispositivos da classe PICK. Todos os demais atributos mencionados estão detalhados em //.

### 3.5.2. O PICK IDENTIFIER

É um número natural utilizado para identificar primitivas dentro de segmentos; não pode ser utilizado para manipulações. Este nível de identificação é relevante em aplicações onde é necessário distinguir um grande número de partes da imagem. A inexistência de manipulação evita a criação de segmentos artificiais e sem significado.

Embora o nome do segmento seja único, o mesmo PICK IDENTIFIER pode ser atribuído a uma única primitiva ou a um conjunto de primitivas dentro do segmento. O valor "default" do PICK IDENTIFIER é 0 (zero), e é atribuído às primitivas quando o segmento é criado; o valor -1 (menos um) torna as primitivas não detectáveis.

Observe-se que a detectabilidade é um atributo do segmento; no entanto, dentro de um segmento detectável, é possível a existência de primitivas não detectáveis; isto também diminui a necessidade de criação de segmentos artificiais.

### 3.6. O CONCEITO DE WORKSTATION

O termo WORKSTATION, utilizado até agora, necessita ser melhor explicado. É na maneira como o GKS "enxerga" a WORKSTATION que se encontra grande parte de sua flexibilidade. Uma WORKSTATION totalmente equipada atende aos seguintes quesitos /7,15/:

- . Não mais que uma superfície de saída endereçável.
- . Permite o uso de superfícies menores do que a máxima; ao mesmo tempo assegura que nada será gerado fora da superfície especificada.
- . Possui capacidade para gerar vários tipos de linhas, cores, tipo e tamanho de caracteres, etc.
- . Possui no mínimo um dispositivo de entrada para cada classe de dispositivos lógicos de entrada.
- . Permite entradas do tipo REQUEST, SAMPLE e EVENT.

As WORKSTATIONS podem ser classificadas como pertencentes a três grupos:

1. WORKSTATION de saída (e.g. PLOTTERS)
2. WORKSTATION de entrada (e.g. TABLET)
3. WORKSTATION interativa (e.g. REFRESH DISPLAY)

Para cada WORKSTATION presente num sistema gráfico de

ve existir uma descrição de suas capacidades e características na WORKSTATION DESCRIPTION TABLE /7/. Um programa de aplicação identifica uma WORKSTATION pelo WORKSTATION IDENTIFIER (W.I.), que pode ser um número ou um nome.

A segunda transformação de coordenadas mencionada na Seção 3.2. ocorre para cada WORKSTATION e pode ser designada individualmente, permitindo que a mesma primitiva gráfica apareça com diferentes escalas em diferentes DISPLAYS de acordo como que for determinado pelo programa de aplicação.

Desta forma, o conceito de WORKSTATION fornece meios para adaptar o programa de aplicação às capacidades de hardware dos terminais, e, além de tratar dispositivos gráficos de E/S como dispositivos lógicos de E/S, o GKS trata uma coleção destes dispositivos como uma unidade funcional, a WORKSTATION que é como o operador tende a "enxergar" este conjunto de dispositivos /15/.

### 3.6.1. OS ATRIBUTOS DA WORKSTATION

São os atributos que permitem a uma mesma primitiva (ou conjunto de primitivas) ser mostrada em diferentes WORKSTATIONS com aparências diversas. Estes atributos são: o OPEN REPRESENTATION, o TEXT REPRESENTATION, o DEFERRAL STATE e a WORKSTATION TRANSFORMATION; os quais serão descritos a seguir.

#### . PEN REPRESENTATION

É a tabela que controla a aparência de todas as primitivas de saída, exceto a primitiva PIXEL ARRAY, incluindo o tipo de linha (sólida, pontilhada, etc.), a largura da linha e a cor.

#### . TEXT REPRESENTATION

É a tabela que controla a aparência das primitivas de

texto, ou seja, o tipo de caractere (padrão, romano, itálico, etc.) e precisão do texto.

#### . DEFERRAL STATE

Permite ao programa de aplicação levar em consideração certas capacidades da WORKSTATION, adiando ou não certas mudanças na imagem.

#### . WORKSTATION TRANSFORMATION

É usada para mapear as coordenadas NDC em DC, para cada WORKSTATION individualmente. A WORKSTATION TRANSFORMATION é designada especificando a WORKSTATION WINDOW em NDC e a WORKSTATION VIEWPORT em DC (dada em metros).

### 3.6.2. A IDENTIFICAÇÃO DOS DISPOSITIVOS DE ENTRADA NA WORKSTATION

Os dispositivos de entrada em uma WORKSTATION são identificados pela tripla: WORKSTATION IDENTIFIER, INPUT CLASS e INPUT DEVICE NUMBER, descritos a seguir:

#### . WORKSTATION IDENTIFIER

Informa ao programa de aplicação de qual WORKSTATION se origina a informação;

#### . INPUT CLASS

Informa a qual classe de dispositivo lógico pertence a informação;

#### . INPUT DEVICE NUMBER

Informa qual o dispositivo físico de entrada, pertence a uma certa classe, em uma dada WORKSTATION, gerou a informação. Assim, uma WORKSTATION pode ter um TABLET e um JOYSTICK para a classe LOCATOR, sem problemas de inconsistência.

### 3.7. O GKS E AS ENTRADAS GRÁFICAS

As funções de entrada gráfica providas pelo GKS podem ser agrupadas em cinco classes, especificando o tipo de primitiva de entrada: LOCATOR, VALUATOR, CHOICE, PICK e STRING. O esclarecimento do significado de cada uma destas classes pode ser obtido do capítulo anterior, tendo em mente que as palavras CHOICE e BUTTON são sinônimos e que as alternativas fornecidas pela classe CHOICE são números inteiros.

Vamos fazer um parêntese para discutir como o GKS define as informações fornecidas pelos dispositivos das classes PICK e LOCATOR. WELLER, et al. /12/ discutem vantagens e desvantagens de certos métodos utilizados para se identificar elementos na tela. No que diz respeito ao GKS, esta identificação é feita em dois níveis hierárquicos. No nível superior, identifica-se o segmento (através do nome) ao qual o elemento apontado pertence. No nível inferior, identifica-se a primitiva ou o conjunto de primitivas ao qual o mesmo elemento pertence. A identificação do segundo nível se dá através do PICK IDENTIFIER. Esta identificação pode ser obtida através de uma estrutura a qual chamamos mapa de correlação e está descrita no capítulo seguinte. A classe LOCATOR retorna ao programa de aplicação coordenadas do usuário como está descrito no início do capítulo.

No GKS, as entradas gráficas podem ser obtidas de qualquer WORKSTATION aberta /7/ e são fornecidos três mecanismos distintos para a obtenção da informação: REQUEST, SAMPLE e EVENT, os quais são descritos a seguir.

#### 3.7.1. O REQUEST

É uma forma síncrona que o programa de aplicação utiliza para obter informação, ou seja, é feito um pedido de leitura do dispositivo. O GKS espera até que a entrada ocorra. A ocorrência da entrada deve ser sinalizada pelo operador. O REQUEST é equivalente ao READ do FORTRAN.

### 3.7.2. O SAMPLE

É também uma forma síncrona para obtenção da informação, contudo, o GKS inspeciona o último dado gerado por um certo dispositivo, sem interferência do operador.

### 3.7.3. O EVENT

Com o dispositivo operando neste modo, o operador tem a possibilidade de fornecer comandos e/ou dados, ou seja, gerar eventos. As entradas do tipo EVENT também devem ser sinalizadas pelo operador.

No GKS, um dispositivo físico fornece entrada em apenas um dos três modos, num dado instante. Desta forma, é necessário manter um controle a fim de estabelecer o modo para cada dispositivo em um certo instante. Isto é conseguido através das funções ENABLE e DISABLE, descritas a seguir.

#### . DISABLE

Permite ao programa de aplicação dar REQUEST para o dispositivo neste estado, inibindo entradas para SAMPLE ou EVENT.

#### . ENABLE

Ou permite ao programa de aplicação amostrar os dispositivos, ou permite entradas destes para a fila de eventos.

Todos os três modos de entrada estão relacionados com a interação do usuário com o GKS, e são controlados pelo programa de aplicação.

### 3.8. A FILA DE EVENTOS

Os comandos e/ou dados gerados por qualquer dispositivo operando no modo EVENT são armazenados em uma estrutura independente do dispositivo, a qual chamaremos fila de eventos.

### 3.8.1. A INSERÇÃO

Os comandos e/ou dados oriundos das várias WORKSTATIONS são arranjados em ordem cronológica de sua geração.

### 3.8.2. A RECUPERAÇÃO

A fila poderá ser inspecionada pelo programa de aplicação a fim de recuperar um comando/dado e identificar a sua origem. Embora a inserção de comandos e/ou dados na fila seja sequencial, a saída não necessariamente o é. Existem três maneiras distintas de recuperar comandos/dados na fila, os quais serão descritos a seguir.

#### 1. AWAIT EVENT

Devolve ao programa de aplicação a identificação da origem do dado mais antigo na fila;

#### 2. AWAIT WORKSTATION EVENT

Devolve ao programa de aplicação a identificação da origem do dado mais antigo na fila, de uma WORKSTATION específica;

#### 3. AWAIT DEVICE EVENT

Devolve ao programa de aplicação a identificação da origem do dado mais antigo na fila de uma WORKSTATION específica e com um DEVICE NUMBER específico;

O tempo é parâmetro de entrada comum às três funções apresentadas. A título de exemplificação, vejamos o que ocorre se um AWAIT EVENT é especificado e a fila está vazia. Neste caso, a função ou esperaria a ocorrência de uma entrada, ou o esgotamento do tempo especificado. No segundo caso, o valor 0 (zero) será retornado ao programa de aplicação.

Após a identificação da origem do comando/dado o programa de aplicação decide se quer ou não utilizá-lo. Em caso afirm

mativo, o dado é recuperado pelo uso das funções GET , por exemplo, GET LOCATOR, GET VALUATOR, etc. O uso de qualquer uma das três funções mencionadas fará com que o comando/dado seja eliminado, quer tenha sido ou não recuperado.

### 3.8.3. A ELIMINAÇÃO

Em complemento às funções de recuperação de dados, o GKS provê funções FLUSH, as quais são utilizadas pelo programa de aplicação a fim de eliminar dados inúteis. Estas funções são:

#### . FLUSH ALL EVENTS

Elimina todos os eventos presentes na fila.

#### . FLUSH WORKSTATION EVENTS

Elimina todos os eventos de uma WORKSTATION específica.

#### . FLUSH DEVICE EVENTS

Elimina todos os eventos de um dispositivo específico em uma WORKSTATION específica.

### 3.9. AS FUNÇÕES ESPECIAIS DE ENTRADA

Descreveremos a seguir duas funções do GKS que contribuem para uma maior flexibilidade das entradas.

#### . ASSIGN STRING TO CHOICE

Permite ao operador utilizar o teclado como dispositivo da classe CHOICE . Teclando uma seqüência pré definida de caracteres, o operador indica uma alternativa.

#### . ASSIGN SEGMENT DO CHOICE

Permite ao operador selecionar alternativas através de um dispositivo operando como PICK. Com esta função, o segmento

apontado não retornará as informações esperadas de um dispositivo da classe PICK, mas fará com que o PICK IDENTIFIER da primitiva selecionada seja utilizado como alternativa do dispositivo CHOICE.

#### 4. UM MODELO PARA O PROCESSAMENTO GRÁFICO DEPENDENTE DO DISPOSITIVO

A idéia básica no projeto de sistemas gráficos de propósito geral é a identificação dentro do sistema, das partes dependentes (DD) e independentes do dispositivo (ID), visando a implementação de entradas/saídas gráficas completamente independentes do programa do usuário. Neste capítulo, apresentamos uma discussão sobre a interação entre as partes ID/DD, e propomos um modelo para a implementação da parte dependente do dispositivo. Acreditamos que este modelo seja válido para descrever vários tipos de WORKSTATIONS interativas.

##### 4.1. OS CONCEITOS DE INDEPENDÊNCIA E DEPENDÊNCIA DO DISPOSITIVO

O termo "device driver" será utilizado para discriminar a parte dependente do dispositivo que apoia o dispositivo gráfico. O "device driver" gera código dependente do dispositivo e trata de interação também da forma dependente do dispositivo.

Para o entendimento das componentes ID/DD para a entrada, é interessante observar o Capítulo 2. onde descrevemos os conceitos de dispositivos físicos e lógicos e o tipo de informação relativa a cada um.

A Figura 4.1 /16/ apresenta uma separação lógica entre entrada/saída gráfica e o programa do usuário. O processador de entrada tem contato direto com os dispositivos físicos; contudo, no momento em que a informação passa através do mesmo, a conexão dispositivo físico-informação é desfeita, surgindo a conexão dispositivo lógico-informação. A partir daí, está gerada a informação independente do dispositivo.

O GKS tem a sua maneira peculiar de: controlar os dispositivos de entrada, por exemplo, controlando o modo de operação de cada um através das funções ENABLE e DISABLE; requisitar dados através de pedidos de REQUEST ou SAMPLE; permitir eco através de SET ECHO e SET ECHO POSITION; permitir maior flexibilidade ao o

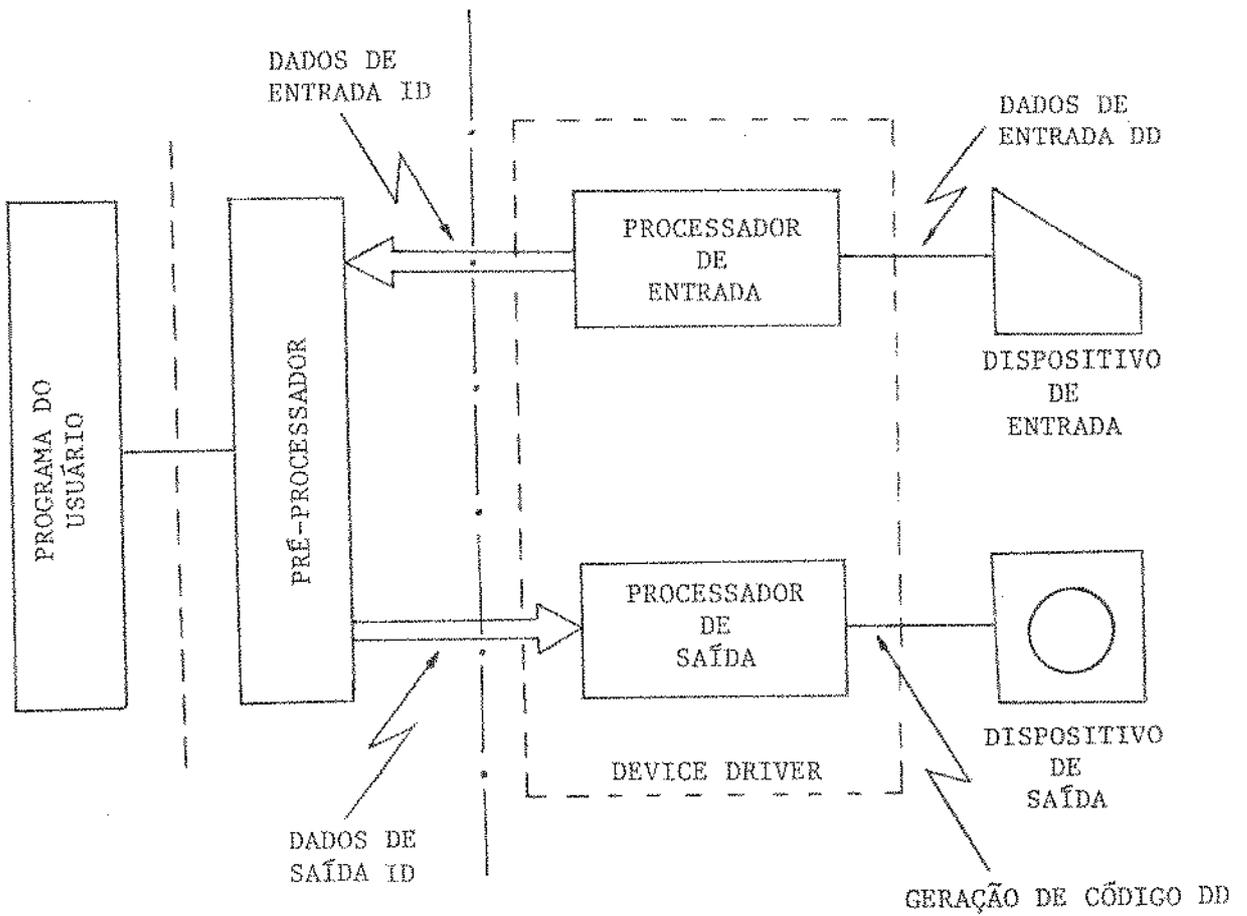


FIG. 4.1 - O Device Driver e os dados ID/DD

perador através de SET CHOICE STRING, ASSIGN STRING TO CHOICE e ASSIGN SEGMENT TO CHOICE. Todas estas funções podem ser aplicadas a todos os dispositivos capazes de manusear com tais capacidades, portanto, elas são gerais e códigos independentes do dispositivo podem ser providenciados a cada uma. A partir da geração do código ID, o "device driver", e mais particularmente o processador de saída, através de passagens de parâmetros convenientes gera códigos dependentes do dispositivo, proporcionando saídas gráficas e possibilidades de operação com os dispositivos de entrada em uma WORKSTATION particular.

A Tabela 4.1 apresenta: uma possível codificação independente do dispositivo para as funções REQUEST, SAMPLE e ENABLE, e os parâmetros de E/S do "device driver" para estas funções.

FUNÇÃO	CÓDIGO ID	PARÂMETROS
REQUEST	100	DC, DN, ARRAY
SAMPLE	200	
ENABLE	300	DC, DN, MODO

TAB. 4.1 - CÓDIGO ID e PARÂMETROS passados ao DEVICE DRIVER

DC ("Device Class"), DN ("Device Number") e MODO são parâmetros de entrada para "device driver". ARRAY é parâmetro de saída para o caso SAMPLE e pode ser de entrada e/ou saída para o caso REQUEST, dependendo da conveniência desejada.

No capítulo seguinte apresentaremos os códigos DD gerados para uma implementação.

#### 4.2. O MONITOR GRÁFICO

Chamaremos de monitor gráfico àquela parte do DEVICE DRIVER (Fig. 4.1) residente no periférico.

Um monitor gráfico deve ser capaz de gerenciar todo o processamento gráfico e não gráfico relacionado com o sistema. Embora pareça estranho um monitor gráfico lidar com informações não gráficas, isto é perfeitamente cabível em virtude das informações não gráficas serem componentes do sistema global. A Figura 4.2 apresenta um diagrama dos blocos componentes do monitor gráfico.

Apresentamos a seguir uma descrição sucinta de cada um dos blocos componentes da parte gráfica.

#### . SAÍDA

É a parte gráfica responsável pela maneira como os dados devem ser visualizados. Esta visualização se manifesta de três modos: adição, modificação e eliminação.

##### .ADIÇÃO

Neste modo, comandos e dados são adicionados ao "display processor program"; geralmente, estes comandos e dados são gerados a partir de informações armazenadas em uma base de dados.

##### .MODIFICAÇÃO

Neste modo, comandos e/ou dados existentes no "display processor program" são modificados, por exemplo, para tomar um segmento sensível ou insensível aos dispositivos da classe PICK.

##### .ELIMINAÇÃO

Neste modo, comandos e dados existentes no "display processor program" são eliminados.

#### . ENTRADA

É a parte gráfica que proporciona ao usuário meios para entrada de comandos e/ou dados utilizados no controle do programa de aplicação, durante sua execução. Pode ser vista como o conjunto

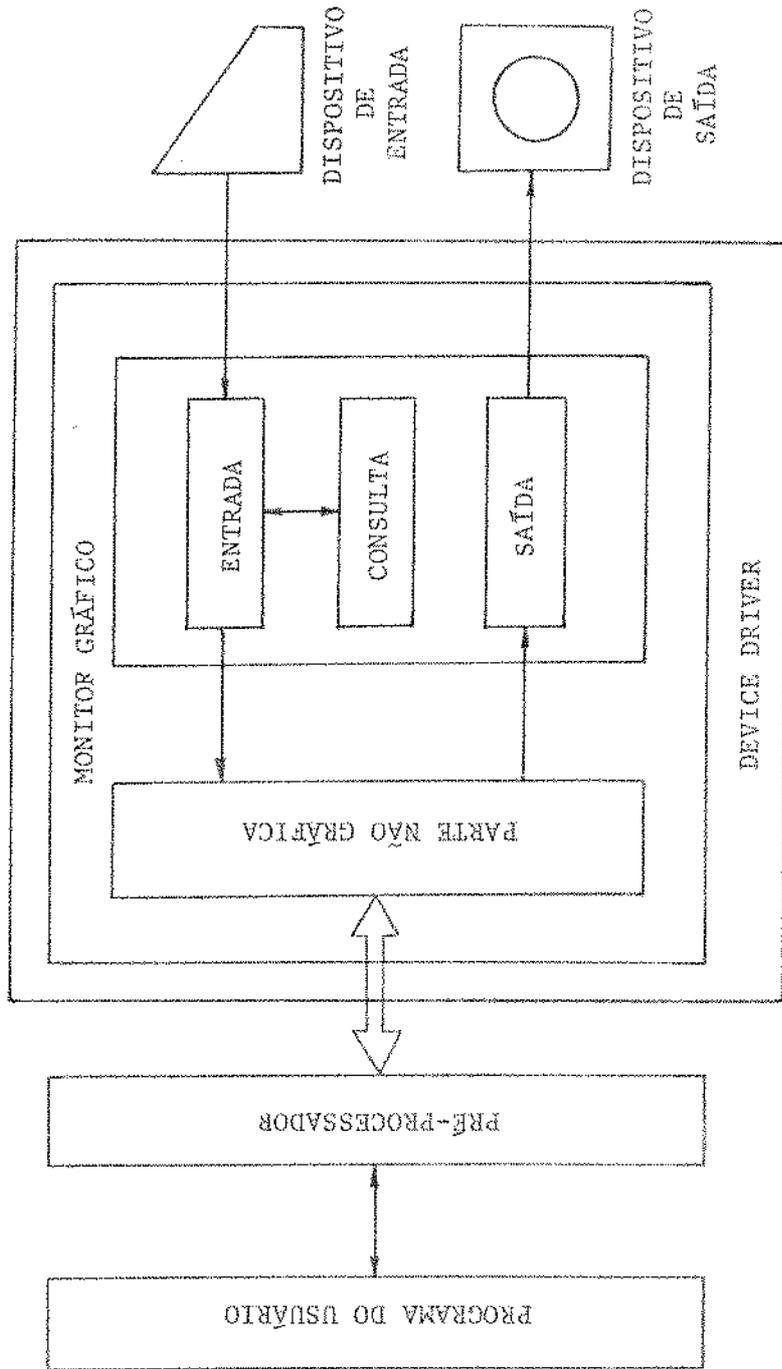


FIG. 4.2 - O Monitor Gráfico e a conexão interna/externa de seus componentes.

de software e hardware utilizados para este fim. A entrada não lida apenas com elementos gráficos, envolvendo também elementos não relacionados à imagem, por exemplo, entrada de um valor a através de potenciômetro, citado no capítulo anterior.

#### . CONSULTA

É a parte gráfica relacionada com consultas a tabelas, fundamental no apoio às entradas gráficas.

#### . PARTE NÃO GRÁFICA

É o suporte da parte gráfica, envolvendo a parte local do protocolo de comunicação e outras funções que serão descritas no transcorrer deste capítulo.

Este monitor gráfico atende o que acreditamos seja o mínimo de processamento local necessário para o desempenho aceitável do sistema. Contudo, se um incremento do processamento local é possível, os blocos já existentes podem ser expandidos e novas rotinas acrescentadas a estes. Por exemplo, se o periférico suportar todo o DEVICE DRIVER descrito na Seção 4.1., seu processador de saída poderá ser decomposto nas partes gráfica e não gráfica do monitor, sem interferir nos blocos existentes. Neste trabalho estamos preocupados com os blocos de "entrada", "consulta" e "parte não gráfica", os quais serão detalhados posteriormente.

### 4.3. O MODELO PROPOSTO

A seguir apresentaremos o modelo do monitor propriamente dito, o qual será abordado sobre dois pontos de vista. O modelo interno e o modelo externo.

#### 4.3.1. O MODELO INTERNO

Este modelo é apresentado sob a forma de blocos e ligações lógicas, seguido do significado e função que cada bloco re

presenta no modelo. Vamos iniciar aglutinando as rotinas do monitor em três processos, como pode ser visto na Figura 4.3, e descrevendo cada um dos processos.

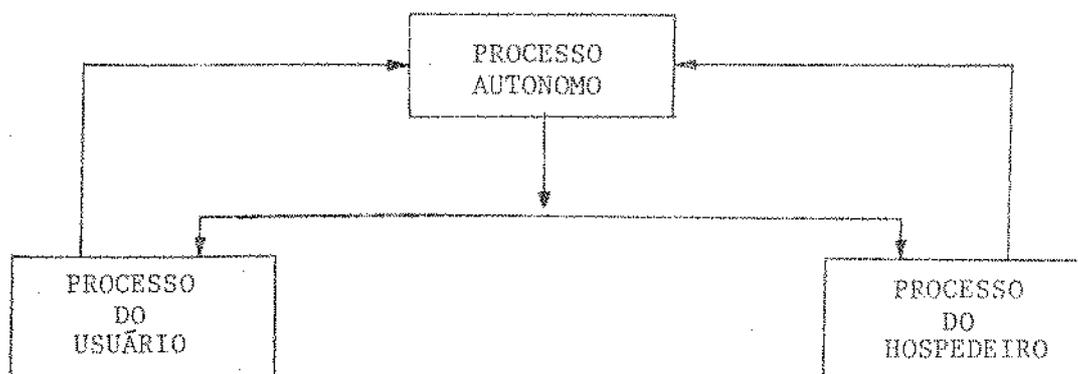


FIG. 4.3 - Processos do monitor gráfico

#### . PROCESSO AUTONOMO

É o processo que existe independentemente da existência de qualquer ação, seja ela tomada no terminal ou enviada pelo hospedeiro.

#### . PROCESSO DO USUÁRIO

Envolve os processos ativados por ações do operador, ou outro mecanismo automático no periférico.

#### . PROCESSO DO HOSPEDEIRO

Envolve os processos ativados pelo hospedeiro.

A Figura 4.4 apresenta o modelo interno proposto para o monitor gráfico. Antes de passarmos à descrição de cada bloco, gostaríamos de alertar para o fato de que o fluxo de controle apresentado, não reflete a única alternativa para ativação dos blocos. Desde que o bloco correntemente ativo permita, mecanismos de interrupção podem desviar o controle para outros blocos a qualquer momento.

A seguir, descreve-se cada um dos blocos componentes do modelo interno do monitor gráfico.

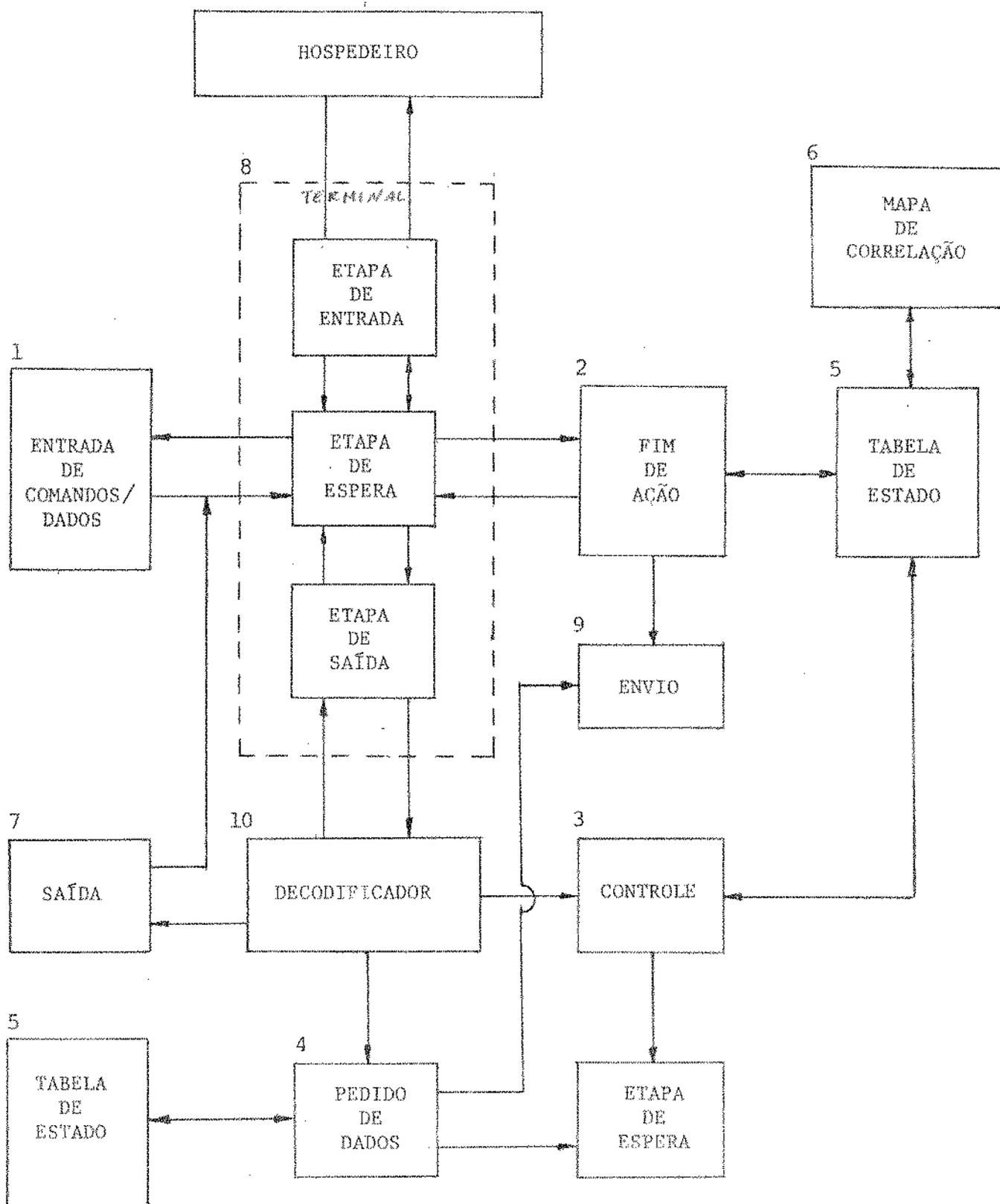


FIG. 4.4 - Modelo interno para o monitor gráfico.

#### . ETAPA DE ESPERA

O sistema sempre se encontrará nesta etapa do processo autônomo, se "nada existe a realizar".

#### . ETAPA DE ENTRADA

Quando o processo autônomo se encontra nesta etapa, a comunicação do usuário com o hospedeiro se realiza como se o periférico fosse um terminal não gráfico, ou seja, independentemente dos modos de entrada REQUEST, SAMPLE ou EVENT. Isto pode ser visto como um quarto modo de entrada, e diz respeito aos recursos e facilidades de interrupção intrínsecos a cada sistema operacional; os comandos introduzidos desta forma recebem atendimento imediato. Podemos tirar proveito desta situação, pois deste modo o operador pode controlar o sistema sem que este lhe tenha facultado esta oportunidade. Por exemplo, o operador pode desejar saber: em que estado o sistema se encontra, que comandos pode dar, qual o espaço disponível em disco, qual o caminho percorrido para chegar ao ponto onde está, pode desejar terminar a execução de uma função, ou mesmo do programa, etc.

#### . ETAPA DA SAÍDA

Tem duas sub-etapas relevantes. Durante a primeira, busca-se o dado procedente do hospedeiro e passa-se o controle ao bloco DECODIFICADOR. Este pode ativar ou não certos procedimentos (e.g. "disable Locator 1"). Se nenhum processo é ativado, o controle passa a segunda sub-etapa de saída, que pode ter a função de fornecer ao operador informações associadas a comandos do tipo HELP, a mensagens de erros, etc.

A este bloco foi dado o nome "terminal", pelo fato de seu comportamento ser muito parecido com o comportamento de um terminal normal.

#### O BLOCO ENTRADA DE COMANDOS/DADOS

Pertence ao bloco de entrada da Fig.4.2 e tem a finalidade de proporcionar ao usuário meios de atuar com os dispositi

tivos de entrada.

#### O BLOCO FIM DE AÇÃO

Pertence também ao bloco de entrada da Fig.4.2 , ou controla a inserção definitiva de comandos e/ou dados na fila de eventos, ou devolve pedidos de REQUEST ao programa de aplicação.

Como poderá ser visto no próximo capítulo, através da apresentação dos algoritmos que implementam cada bloco proposto, os blocos de "Entrada de comandos/dados" e "Fim de ação", são suficientes para que o operador possa corrigir erros localmente; exceto se o dispositivo está apto a operar no modo SAMPLE.

#### O BLOCO DECODIFICADOR

Pertence a parte não gráfica da Fig. 4.2 e sua função é decodificar as mensagens enviadas pelo hospedeiro,ativando as rotinas convenientes.

#### O BLOCO CONTROLE

Pertence ao bloco de entrada da Fig. 4.2 e sua função é controlar os estados dos dispositivos atuando na tabela de estado.

#### O BLOCO PEDIDO DE DADOS

Pertence ao bloco de entrada da Fig. 4.2 ; ou informa ao operador que dados estão sendo requisitados pelo programa de aplicação (caso de pedido de REQUEST ), ou envia os dados pedidos sem qualquer diálogo com o operador (caso de pedido de SAMPLE).

#### O BLOCO ENVIO

Pertence à parte não gráfica da Fig. 4.2 e é o bloco responsável pelo envio de informações dos dispositivos de entrada gráfica ao periférico. Sua implementação depende do sistema operacional do hospedeiro.

## O BLOCO TABELA DE ESTADO

Pertence ao bloco de consulta da Fig. 4.2 e contém todos os "flags" dando informações sobre o estado de todos os dispositivos de entrada, na WORKSTATION.

## O BLOCO MAPA DE CORRELAÇÃO

Pertence ao bloco de consulta da Fig. 4.2 ; é uma estrutura dependente do dispositivo e tem como finalidade atender à classe PICK e a modificações dinâmicas dos atributos do segmento e da WORKSTATION . Estamos interessados na parte relacionada com a classe PICK, pois é esta que diz respeito à entrada. Portanto, sua estrutura deve estar relacionada com o identificador do item apontado (no caso do GKS este identificador são o "segment name" e o "pick identifier"), e a maneira pela qual se fará a identificação, uma vez que esta pode variar de terminal para terminal.

Existem duas maneiras distintas de se apontar um item sobre a tela. Uma delas é apontar o item diretamente (por exemplo, com uma LIGHTPEN), a outra é posicionar um símbolo padrão próximo ao item (por exemplo, com um JOYSTICK). A primeira só é possível nos DISPLAYS com regeneração, e a identificação do item pode ser feita, ou pelo seu endereço no "display processor program" , ou pelas coordenadas do ponto apontado. A identificação pelo endereço torna o mapa de correlação mais simples, desde que o endereço pode ser usado como chave para entrada no mapa; a identificação do item é feita sem nenhum cálculo adicional. Infelizmente, os "displays" tipo "raster" não possuem esta característica, e o item deve ser identificado a partir das coordenadas do ponto. Isto pode requerer grande quantidade de testes, e o mapa de correlação, se não é mais complexo estruturalmente, necessita de um maior volume de informações.

A identificação posicionando um símbolo padrão, geralmente um cursor, embora possa ser utilizada por qualquer DISPLAY, possui o mesmo tipo de restrição dos "displays" tipo "raster".

Esta restrição é agravada pois deve ser considerado que o operador não posicionará o cursor exatamente sobre o item desejado, aumentando, portanto, a quantidade de testes necessários a identificação. Algoritmos para este tipo de identificação são discutidos por WELLER et al. /12/.

Não desejamos que alguém, baseado na exposição anterior, tire conclusões apressadas sobre a melhor maneira de se obter a identificação. Além das questões expostas, devem ser considerados os requisitos requeridos pela aplicação específica.

De todos os blocos apresentados, o mapa de correlação é o que oferece maiores restrições para implementação local. Isto se deve ao volume de informação necessário nos três casos, além do tempo de processamento nos dois últimos. Bases de dados podem ser utilizadas para armazenar as informações requeridas pelo mapa.

#### 4.3.2. O MODELO EXTERNO

Genericamente, o modelo externo se refere à percepção do sistema pelo usuário. Aqui restringir-nos-emos à percepção pelo usuário da interface de entrada.

Para evitar confusão semântica, esclarecemos que o termo percepção não se refere à maneira pela qual o usuário se informa acerca de suas ações e/ou ações do programa de aplicação, este tipo de informação está relacionado tanto às características dos dispositivos de saída quanto de entrada. O significado atribuído ao termo percepção refere-se às facilidades, de que os usuários devem ter conhecimento a fim de executarem suas ações com segurança.

Abordaremos a seguir três itens relacionados com a interação do usuário, sem nos preocuparmos em definir uma posição definitiva sobre a melhor maneira do usuário obter informações acerca dos mesmos. Acreditamos que em um modelo externo devam estar presentes informações relacionadas com:

- . Disponibilidade de cada dispositivo na WORKSTATION - este tipo de informação proporciona ao usuário a segurança de que está operando com o dispositivo em um modo disponível e desejado.
- . Correção de um comando/dado - este tipo de informação assegura ao usuário que sua correção foi efetivada.
- . Valor numérico do dado - esta facilidade permite ao usuário verificar o valor da grandeza sendo introduzida.

No próximo capítulo apresentaremos uma maneira do usuário se informar acerca dos itens propostos.

#### 4.4. O MONITOR GRÁFICO COMO PARTE DO DEVICE DRIVER

A Tabela 4.2 apresenta um resumo dos blocos da Figura 4.4 e seus enquadramentos dentro da nossa classificação de processos e divisão de rotinas do monitor gráfico.

PROCESSO	BLOCOS
AUTONOMO	8
USUÁRIO	1,2
HOSPEDEIRO	3,4,7

ROTINAS	BLOCOS
ENTRADA	1,2,3,4
CONSULTA	5,6
SAÍDA	7
NÃO GRÁFICA	8,9,10

TAB. 4.2 - Relação processos/blocos, rotinas/blocos

O processo do usuário corresponde ao que chamamos na Figura 4.1 de processador de entrada. Os blocos 3 e 4, da Figura 4.4, pertencentes ao processo do hospedeiro, foram enquadrados como parte do bloco de entrada por estarem relacionados apenas às entradas gráficas. O bloco 7 (saída), da forma como foi proposto, não deve ser confundido com o processador de saída (Fig. 4.1), que tem funções muito mais gerais, como a geração dos códigos DD. Contudo, o bloco 7 pode ser visto como um componente do processador de saída.

Na Figura 4.1 explicitamos apenas as partes diretamente envolvidas no processo de entrada/saída gráficas. Por esta razão, a parte de consulta e a parte "não gráfica" foram omitidas. O que deve ficar claro é que o monitor gráfico é apenas um subconjunto do DEVICE DRIVER que constitui o conjunto dependente do dispositivo.

## 5. IMPLEMENTAÇÃO DE UMA INTERFACE DE ENTRADA PARA UMA WORKSTATION INTERATIVA

Neste capítulo discutiremos na seção inicial questões gerais relacionadas à implementação de uma interface de entrada e nas seções subsequentes apresentaremos a implementação desta interface, a qual acopla os dispositivos de entrada ao Núcleo Gráfico, como pode ser visto na Figura 3.2. O modelo para a implementação de uma interface como esta foi discutido no Capítulo 4., e o mesmo foi adaptado aos nossos recursos, sendo estes decompostos em hardware e software. O hardware é composto por um computador PDP-10 e um terminal gráfico interativo do tipo "line-drawing" - GT40, o qual possui: para processamento local um minicomputador PDP-11/05 com 16 Kbytes de memória principal, e para entradas gráficas dois dispositivos - um TECLADO e uma LIGHTPEN. O software é composto pelo sistema operacional do DEC-10, aliado ao "cross-assembly" MACDLX, o qual gera linguagem de máquina do PDP-11 a partir de programas em "assembly", permitindo posterior carregamento e execução.

### 5.1. CONSIDERAÇÕES GERAIS SOBRE A IMPLEMENTAÇÃO DE UM PROCESSADOR DE ENTRADA

Nesta seção abordaremos três questões que, em nossa opinião, devem ser observadas com cuidado ao se implementar um processador de entrada.

#### 5.1.1. A DISTRIBUIÇÃO DO PROCESSAMENTO

Quando estamos lidando com sistemas descentralizados, existem partes do sistema que devem ser necessariamente implementados no hospedeiro, e partes que devem necessariamente ser implementadas no periférico. Qualquer inversão de papéis tornará o sistema inútil para uso prático. Por outro lado, existem partes que podem ser implementadas no hospedeiro ou no periférico, e o projetista deve adotar uma solução de compromisso considerando: tempo de resposta, capacidade de processamento local, memória dispo

nível, etc.

Ainda com relação ao problema de onde implementar o que, fazemos as considerações que se seguem. Não é de se esperar que o homem tenha o mesmo comportamento diante de todas as ações por ele tomadas. Isto foi discutido por MILLER /17/, FOLLEY e WALLACE /11/. Os dois últimos definem três níveis de ações, a saber: o nível léxico, o nível sintático e o nível semântico. O que existe de comum em todos estes níveis é que eles requerem realimentação ao usuário. Para atender a nossos propósitos é suficiente preocuparmos-nos apenas com o nível léxico, sobre o qual FOLLEY e WALLACE /11/ afirmam: "Neste nível, estão todas as ações que são feitas por reflexo, de maneira natural ou treinada, e o tempo de resposta às mesmas não deve ultrapassar 50ms e concluem: "O homem não tolera interrupção de seus reflexos".

As ações de entrada de comandos e/ou dados devem ser classificadas como léxicas, pois requerem realimentação quase imediata. Desta exposição, chegamos a conclusão de que os sistemas de aquisição e realimentação devem estar no periférico.

### 5.1.2. A CONTINUIDADE SINTÁTICA

Esta seção está baseada no trabalho de FOLLEY e WALLACE /11/, e é incluída aqui pela relevância das questões abordadas. A continuidade sintática permite que o usuário faça uso de um sistema com o mínimo de auto-treinamento e de compensação por inadequação do mesmo, além de assumir particular importância em sistemas destinados à acessibilidade de muitos usuários, ou mesmo quando cada usuário não faz acesso freqüente aos mesmos.

A continuidade sintática é definida em três níveis: continuidade visual, continuidade tátil e continuidade contextual, as quais serão descritos a seguir.

#### . A CONTINUIDADE VISUAL

A idéia de continuidade visual é a de que em uma da

da sentença (ação), a atenção visual deveria se restringir a uma única área que pode se deslocar, mas de modo contínuo, ao longo do decorrer da sentença (tomada de ação). Por exemplo, uma LIGHTPEN ou um cursor, freqüentemente fornecem uma indicação visual de ação, ou uma seqüência de objetos cintilantes pode desviar a atenção de uma área para outra. Por outro lado, a necessidade de se preocupar por um campo subitamente deslocado durante uma ação é indesejável e não-natural, como também o fornecimento de mensagens de erro em dispositivo separado o é.

#### . A CONTINUIDADE TÁTIL

A continuidade tátil se refere à necessidade de evitar que se tateie, ou se procure com as mãos, joelho ou pés, uma vez que a sentença tenha sido iniciada. Desta forma, não deve ser necessário mover a mão para um dispositivo independente, durante a seqüência de execução de uma ação, especialmente para dispositivos que não podem ser encontrados em uma posição fixa, como a LIGHTPEN.

#### . A CONTINUIDADE CONTEXTUAL

Refere-se à ausência de efeitos colaterais não identificados, resultantes das ações do usuário. É melhor atingida quando são fornecidas respostas imediatamente perceptíveis para reforçar o efeito de cada passo na ação. Se o reforço, ou realimentação é visível, ela deve estar localizada dentro de um campo primário de visão para aquela parte da ação. A continuidade contextual pode ser preservada em prejuízo da continuidade visual, se o usuário é treinado a olhar para algum local padrão, a fim de obter reforços especiais, tal como diagnóstico. Este local pode ser um outro ponto da tela, um segundo DISPLAY, ou pode estar num teletipo adjacente.

Apesar destas virtudes algumas vezes se neutralizam mutuamente, e serem freqüentemente difíceis de se obter, linguagens gráficas conversacionais de sucesso normalmente incluem es

tas propriedades.

Se o sistema não apresenta continuidade sintática, usuários experientes se auto-treinarão a fim de compensar por esta falha. Por exemplo, o usuário pode se treinar inconscientemente para antecipar a localização de um campo com o objetivo de recuperar a continuidade visual, ou para acompanhar mudanças descontínuas de contexto.

### 5.1.3. O PROBLEMA DA ENTRADA INCORRETA DE DADOS

Os projetistas de sistemas devem levar em consideração que os usuários não são infalíveis; entradas incorretas ocorrerão, e o usuário não deve ser excessivamente penalizado por isto. CUNHA /18/ cita um exemplo típico: "Em programas FORTRAN verifica-se na entrada de dados numéricos quando, por engano, introduzir-se um caractere ilegal numa seqüência numérica, e.g., uma letra". O usuário pode pagar caro por este engano, perdendo eventualmente tudo que tiver sido feito desde o início da execução. Em sistemas interativos, este comportamento é intolerável e não deve ocorrer.

Existe apenas uma maneira do usuário informar-se a respeito de um erro ocorrido que é através da realimentação. Contudo, a realimentação não é usada apenas com o propósito de informar erros. Por exemplo, se a execução de um comando é lenta, é benéfico ao usuário a confirmação de que o comando foi aceito e está sendo executado, ou está em uma fila de espera. A Figura 5.1 ilustra os três tipos de realimentação caracterizados por SPROLL e NEWMAN /8b/, descritos a seguir.

#### . A REALIMENTAÇÃO DE COMANDOS

Informa ao usuário se o comando foi aceito, em que estágio se encontra ou se houve erros em sua introdução. Estes erros podem ser léxicos, sintáticos, semânticos e comandos introduzidos fora de uma seqüência pré-determinada.

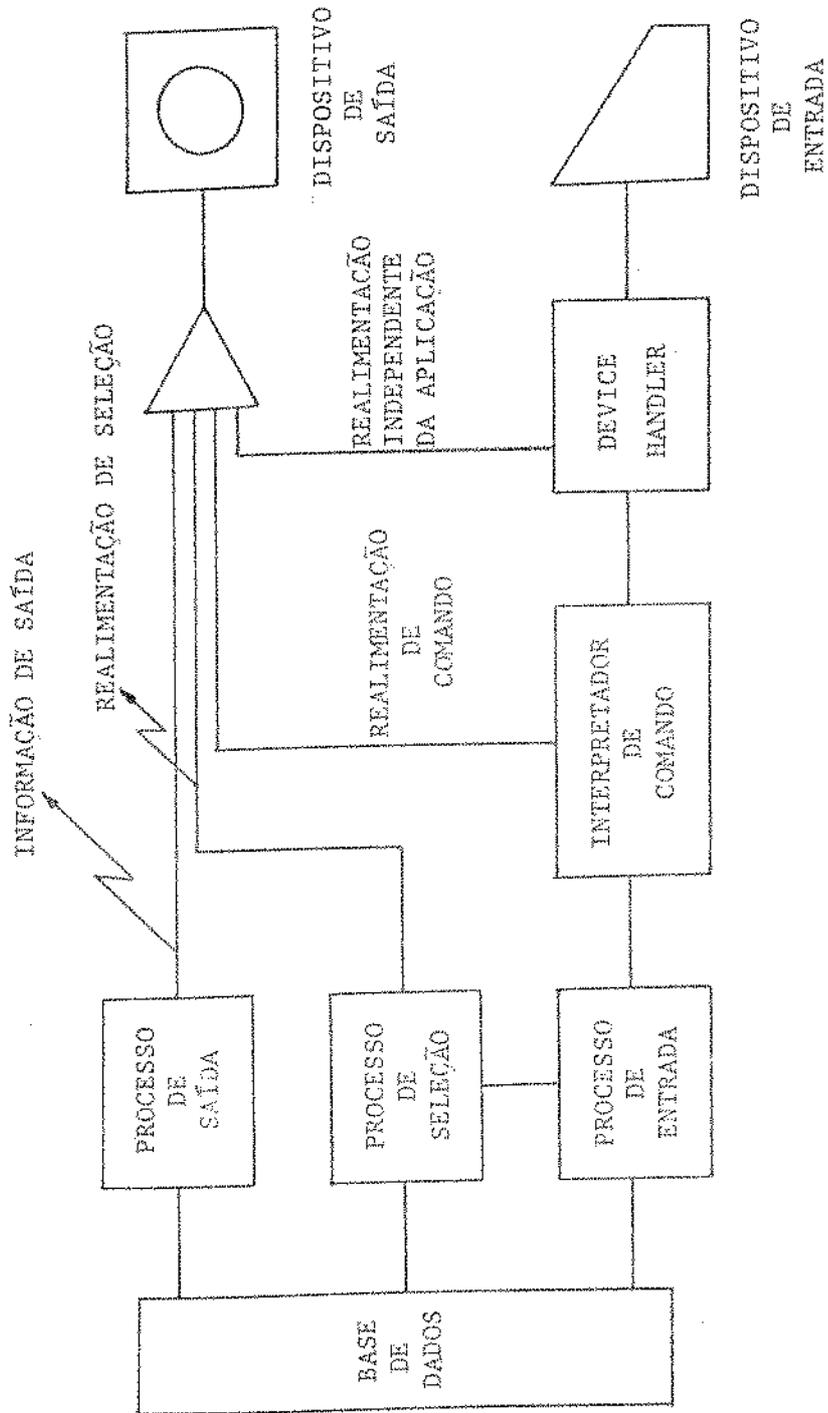


FIG. 5.1 - Modelo expandido de um processo interativo e tipos de realimentação.

## . A REALIMENTAÇÃO DE SELEÇÃO

Utilizado em aplicações com base de dados. Em casos onde a seleção de um item na tela dependa de consultas a uma base de dados, a informação de que o item foi realmente selecionado deve ser providenciada, por exemplo, variando a intensidade do item selecionado.

## . A REALIMENTAÇÃO INDEPENDENTE DA APLICAÇÃO

São aquelas não relacionadas à interpretação de comandos ou consultas à base de dados. Por exemplo, uma das mais básicas é a realimentação do movimento do cursor. De forma análoga, o eco é a realimentação dos caracteres teclados.

Consideremos duas formas de detecção de erros: uma a nível de usuário, com auxílio de realimentação independente da aplicação, e a outra a nível de sistema, com o auxílio de realimentação de comando e de seleção. Todos os erros introduzidos pelo operador podem ser detectados a nível de usuário, no entanto quando este falha, o sistema deve encarregar-se de fazê-lo. Por exemplo, se o operador insere um comando ilegal, ou entra com um dado numérico não pertencente à faixa pré-estabelecida, o sistema deve não apenas detectar o erro, mas emitir mensagens especificando o tipo de ilegalidade ocorrida, e oferecer ao usuário nova oportunidade de entrada.

Para a detecção de erro a nível de sistema, CUNHA /18/ sugere: "Trata-se de realizar testes exaustivos aos comandos e dados introduzidos pelo usuário, e fazê-lo ao nível mais baixo possível, não só para tornar a recuperação mais econômica, como para minimizar o risco de se gerarem situações incontrolláveis..." Entretanto, esta problemática está fora de nosso escopo e não será abordada.

Estamos preocupados com a detecção e correção a nível de usuário, por acreditarmos que este processo está intimamente relacionado à entrada, e ser de grande valia para o desempenho do

sistema. Além de ser o mais baixo nível de detecção, resultando numa correção mais econômica (por exemplo, menor tempo), evita frustração do usuário, pois, o sistema oferece a oportunidade de corrigir imediatamente as informações incorretas introduzidas, se o comando <fim de ação> não foi ainda acionado. Como exemplo desta facilidade podemos citar o papel que a tecla "rubout" desempenha junto ao sistema operacional do DEC-10.

## 5.2. O MODELO DE VISUALIZAÇÃO

Este modelo define a maneira pela qual o usuário se informará a respeito do estado dos dispositivos de entrada, bem como de suas entradas correntes.

A superfície de visualização do terminal gráfico GT40 /19/ é de 1024 X 768 unidades de "raster". Desta forma, a relação dos módulos dos vetores  $X = (x,0)$  e  $Y = (y,0)$  é:  $|X|/|Y| = 4/3$ , onde  $x$  e  $y$  representam respectivamente as maiores ordenada e abcissa, sem violação da borda. Como a relação é diferente da unidade, assim que o GKS proceder ao mapeamento de coordenadas normalizadas (NDC) para coordenadas do dispositivo (DC), ocorrerá uma distorção da imagem. Isto pode ser facilmente contornado, assegurando uma relação unitária, ou seja, não permitindo que o módulo do vetor  $X$  ultrapasse 768 unidades.

Na WORKSTATION DESCRIPTION TABLE existe uma posição reservada para se declarar a superfície máxima de visualização da WORKSTATION. Esta declaração é feita de duas formas distintas - uma declarando a superfície em metros e a outra em unidades do dispositivo. Desta forma, a segunda seria declarada 768 x 768, e a primeira  $x \times x$ , onde  $x$  é um número real (DC) representando a conversão das 768 unidades de "raster" para metros.

Com isto temos uma superfície de 256 x 768 sem utilização para efeitos de saída gráfica. Esta faixa, aliada às características do terminal, pode ser utilizada para orientação do operador durante a execução do programa de aplicação. A Figura 5.2 apresenta o modelo de visualização proposto, o qual será des

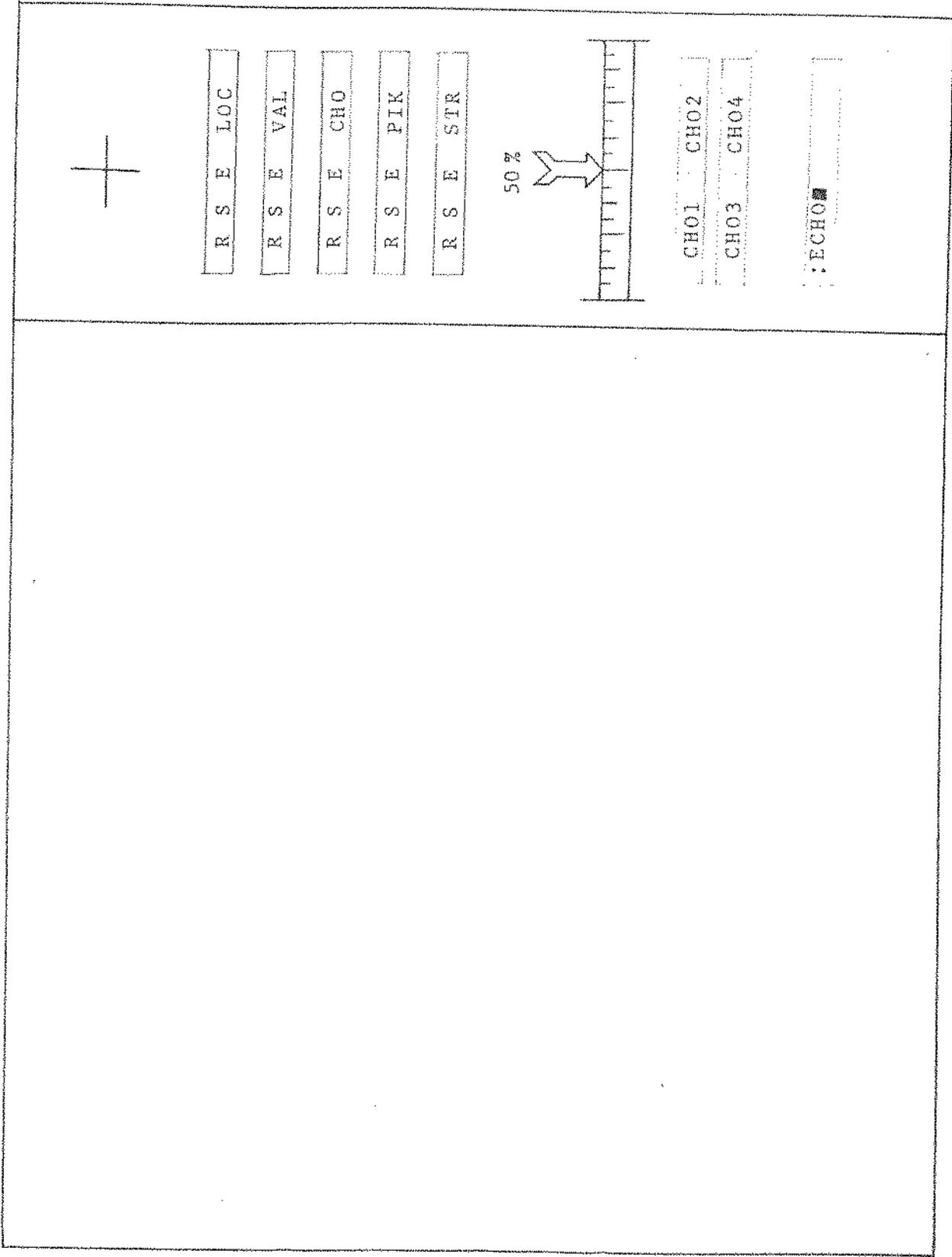


FIG. 5.2 - Um Modelo de Visualização para entradas gráficas

crito a seguir.

### 5.2.1. A VISUALIZAÇÃO DO MODO DE OPERAÇÃO

Na Figura 4.2 , as seqüências de três letras R, S e E indicam o modo de operação de cada dispositivo lógico a elas as sociado. Isto é feito da seguinte maneira:

- . Se nenhuma das três letras pisca, o dispositivo associado en contra-se no estado DISABLE.
- . Se a letra R pisca, houve um pedido de REQUEST para o disposi tivo associado.
- . Se a letra S pisca, o dispositivo associado encontra-se no es tado ENABLE, e está apto a operar no modo SAMPLE.
- . Se a letra E pisca, o dispositivo associado encontra-se no es tado ENABLE, e está apto a operar no modo EVENT.

### 5.2.2. A VISUALIZAÇÃO DA ATIVAÇÃO DO DISPOSITIVO LÓGICO

Diremos que um dispositivo lógico de entrada está a tivo caso as duas condições a seguir se verificarem simultaneamente.

- . Ou o dispositivo se encontra no modo ENABLE, ou houve pedido de REQUEST para operação do mesmo e;
- . O operador manifesta, através de ação correta, sua intenção de operar com o dispositivo. Por exemplo, para operar o disposi tivo LOCATOR, ele deve apontar a LIGHTPEN para o cursor, e es perar que a interrupção ocorra.

O operador visualiza a ativação de um dispositivo ob servando o mnemônico correspondente a este. Se o mnemônico está piscando, o dispositivo está ativado, caso contrário, está desa tivado. Assim, em um dado momento, todos os dispositivos podem se encontrar ativados. Um dispositivo ativado está apto a receber o

comando <fim de ação>. Este comando, além de desativar o dispositivo, encaminhará as informações associadas de acordo com o modo de operação do mesmo.

### 5.3. A SIMULAÇÃO

Como foi mencionado anteriormente, a simulação não tem somente o objetivo de superar o obstáculo da inexistência do protótipo físico, podendo a sua utilização ser atrativa, por exemplo, para obter continuidade tátil.

Para implementação das classes PICK e STRING serão alocados a LIGHTPEN e o TECLADO, respectivamente. Embora possam ser simulados, não vemos vantagem em fazê-lo diante das características atrativas oferecidas pelos protótipos disponíveis.

FOLEY e WALLACE /11/ dizem: "Linguagens de entrada que adotam a filosofia de um único dispositivo físico para seqüências de ações, enfatizam a continuidade tátil, e apresentam grandes atrativos quando podem ser aplicadas". Esta frase sugere que as simulações devem se proceder, tanto quanto possível, com um único dispositivo físico.

Embora o TECLADO seja o dispositivo "mais completo" para interações, um dispositivo como a LIGHTPEN apresenta atrativos sobre o mesmo, notadamente a equivalência psicológica entre os dispositivos simulado e físico. Por exemplo, a classe VALUATOR pode ser simulada facilmente por LIGHTPEN ou por TECLADO. Entretanto, se estamos simulando um potenciômetro, é natural que o operador tenha uma sensação mais realística de um potenciômetro, se faz uso da LIGHTPEN em conjunto com uma escala como se mostra na Figura 5.3, em vez de simplesmente emitir valores via TECLADO.

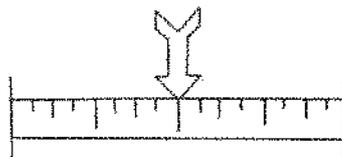


FIG. 5.3 - Potenciômetro deslizante simulado por LIGHTPEN

Diante do quadro exposto aqui, e na Seção 2.3., acreditamos ser conveniente o uso da LIGHTPEN a fim de simular os dispositivos da classe LOCATOR, VALUATOR e CHOICE. Desde que a LIGHTPEN atua através de interrupção dos processadores gráfico e não gráfico, cujo endereço é fixo, tudo que temos a fazer é salvar o endereço e as coordenadas da interrupção, continuar a execução do processamento gráfico, e identificar o propósito do usuário. A partir deste ponto, as rotinas são específicas de cada dispositivo lógico, e tudo ocorrerá como se não houvesse simulação.

#### 5.4. A ENTRADA DE COMANDOS/DADOS

Nesta seção descreveremos as simulações para os três dispositivos mencionados anteriormente, e a forma pela qual o usuário entrará com comandos/dados, relacionados com os cinco dispositivos presentes. O algoritmo que trata da entrada de comandos/dados é também descrito nesta seção.

##### 5.4.1. O DISPOSITIVO LOCATOR

O cursor mostrado na Figura 5.2, juntamente com a LIGHTPEN, simulam o LOCATOR. O centro do cursor contém as coordenadas de referência, e assim que a LIGHTPEN é apontada para o mesmo, uma interrupção ocorre, e o cursor é movido de acordo com o algoritmo de implementação. Detalhes destes algoritmos são discutidos em muitos trabalhos /1,8,20/. O algoritmo adotado permite que o cursor seja movido em quatro sentidos (para cima, para baixo, para a direita, para a esquerda), e as coordenadas de seu centro são atualizadas a cada interrupção do dispositivo.

##### 5.4.2. O DISPOSITIVO VALUATOR

Como mostrado na Figura 5.2, o VALUATOR é simulado por dois segmentos de reta horizontais, e um marcador de posição, dando a idéia de um potenciômetro deslizante. Este funcionará segundo a concepção do GKS, ou seja, dada a faixa em que os valores devem variar, digamos entre LI e LS, tudo se passará como se po

tenciais numericamente iguais a LI e a LS fossem aplicados aos seus terminais. Como o segmento de reta sensível à LIGHTPEN é horizontal, apenas a abscissa é de interesse, e assim que a interrupção ocorre, o marcador é atualizado. O operador se informará acerca do valor corrente pela posição do marcador.

#### 5.4.3. O DISPOSITIVO CHOICE

A alternativa adotada na simulação desta classe foi a técnica de MENU, a qual é universalmente utilizada por sua facilidade de manipulação e pequena margem de erros incorridos com a mesma /1,8/.

Esta simulação mantém coerência com a filosofia do GKS quanto ao fato de se utilizar esta classe tanto via escolha direta dos "light buttons" disponíveis, quanto via TECLADO através de instruções ASSIGN STRING TO CHOICE, e ASSIGN SEGMENT TO CHOICE discutidas no Capítulo 2.

Utilizamos quatro funções programáveis, às quais podem ser adicionadas outras de uma forma trivial. Cada opção no MENU tem uma função definida por software, e pode variar de aplicação para aplicação. A única desvantagem do controle por software sobre os dispositivos da classe BUTTON, é que a definição funcional de um BUTTON pode ser mudada sem qualquer mudança física que possa alertar o usuário /9/.

Para entrar com comandos, o usuário deve apontar a LIGHTPEN para o "light button" desejado até perceber que este está piscando; o piscamento de um "light button" indica a efetivação da escolha.

#### 5.4.4. O DISPOSITIVO PICK

Com o uso da LIGHTPEN, qualquer item pertencente à imagem pode ser apontado e identificado como se descreve a seguir. Em um caso geral, a imagem é composta de segmentos, e estes são compostos de itens. O operador, apontando um item do segmento fa

rã com que todo o segmento pisque, e a identificação do segmento será discutido mais adiante neste capítulo.

#### 5.4.5. O DISPOSITIVO STRING

Tão natural quanto o uso da LIGHTPEN para apontar itens na tela, é o uso do TECLADO para entrar com seqüência de caracteres. Para o TECLADO, entretanto, um conhecimento mínimo de datilografia é requerido. O eco dos caracteres teclados ocorre em uma posição fixa da tela e é indicada na Figura 5.2.

#### 5.4.6. O PROCEDIMENTO PARA ENTRADA DE COMANDOS/DADOS

A seguir apresentaremos, em linguagem natural, o procedimento usado para a implementação da entrada de comandos / dados.

##### PROCEDIMENTO ENTRADA DE COMANDOS/DADOS

BEGIN

(Determine com qual dispositivo o usuário  
deseja fornecer informação)

IF

(Dispositivo está ENABLE .OR. Houve pe  
dido de REQUEST para o dispositivo)

THEN

BEGIN

(Atualize o BUFFER de entrada  
do dispositivo)

END

(Ignore o dado)

ELSE

END

#### 5.4.7. BUFFER PARA ENTRADA DE COMANDOS/DADOS

A fim de tornar uniforme a manipulação das informações de todos os dispositivos, para todos os modos de operação ,

para cada dispositivo, definimos um BUFFER de entrada, no qual a informação de cada dispositivo estará armazenada. Uma vez disponível, esta será encaminhada ao seu destino, utilizando as mesmas rotinas, independentemente do BUFFER de origem.

Os BUFFERS de entrada devem conter as informações correntes dos dispositivos ativados ou as informações mais recentes dos desativados. Desta forma, os pedidos de SAMPLE serão atendidos pelas informações armazenadas nos respectivos BUFFERS.

Os BUFFERS para os cinco dispositivos são descritos a seguir:

. BUFFER DE LOCATOR

BLOCX: contém a abscissa do centro do cursor.

BLOCY: contém a ordenada do centro do cursor.

. BUFFER DE VALUATOR

BVAL: contém a abscissa utilizada na normalização.

. BUFFER DE CHOICE

BCHO: contém o "light button" selecionado.

. BUFFER DE PICK

BPIK: contém o endereço do item apontado.

. BUFFER DE STRING

BNCAR: contém o número de caracteres.

BSTR: contém a seqüência de caracteres.

## 5.5. O COMANDO DE SINALIZAÇÃO DO OPERADOR

O comando <fim de ação> é a forma com a qual o operador sinaliza ao monitor que uma dada ação está concluída. Pelo fato de permitirmos que todos os dispositivos em uma WORKSTATION possam estar ativos simultaneamente, é necessário que um comando <fim de ação> seja providenciado para cada dispositivo.

Os próprios mnemônicos utilizados para visualizar a ativação dos dispositivos são os "light buttons" para o comando <fim de ação> de todos os dispositivos lógicos que são implementados pela LIGHTPEN. Desta forma, o comando <fim de ação> para o LOCATOR é dado apontando com a LIGHTPEN para o mnemônico LOC da Figura 5.2 ; para o VALUATOR, apontando para VAL, e assim sucessivamente; por outro lado, no TECLADO este comando é realizado tecendo "carriage return" <CR>. A filosofia aqui adotada para executar o comando <fim de ação> preserva a continuidade tátil.

A seguir apresentaremos, em linguagem natural, o procedimento usado na implementação do comando <fim de ação>.

#### PROCEDIMENTO SINALIZAÇÃO

BEGIN

(Determine a qual dispositivo pertence o comando <fim de ação>. Consulte o estado deste dispositivo)

IF (Dispositivo está ENABLE  
para EVENT)

THEN

BEGIN

(Mova o conteúdo do BUFFER de entrada deste dispositivo para a fila de eventos)

END

ELSE

BEGIN

IF (Dispositivo está DISABLE .AND.

Houve pedido de REQUEST para o mesmo)

THEN

BEGIN

(Transfira o conteúdo do BUFFER de entrada do dispositivo para o hospedeiro)

END

ELSE

BEGIN

(Ignore o comando <fim de ação>)

END

END

END

## 5.6. A CORREÇÃO DE ERROS

Mencionamos na Seção 5.1.3. a influência que um sistema de correção de erros tem no desempenho do sistema. Descrevemos a seguir a nossa concepção sobre esta correção a nível de usuário, e observamos que esta é possível somente para os modos de operação que permitem o comando <fim de ação>, ou seja, para os dispositivos que estejam operando em REQUEST ou EVENT.

### . O CASO LOCATOR

O operador tem a liberdade de posicionar o cursor em qualquer local da tela. Desta forma, as posições intermediárias são todas desprezadas, considerando-se apenas a posição do cursor no instante em que o comando <fim de ação> for executado.

### . O CASO VALUATOR

O operador tem a liberdade de posicionar o marcador em qualquer local do segmento de reta. As mesmas considerações do caso acima são válidas aqui.

### . O CASO CHOICE

Caso o operador selecione um "light button", e a seguir verifica que não é o desejado, a simples escolha de um outro será interpretada pelo monitor como o desejo de substituição do primeiro. O operador, observando o "light button" piscando, terá a confirmação da escolha; se esta estiver de acordo com sua expectativa, o comando <fim de ação> pode ser executado; caso contrário, outro "light button" deve ser escolhido.

### . O CASO PICK

Ao apontar para um segmento, este imediatamente mudará de estado, ou seja, caso estiver piscando, deixará de piscar, se não estiver, piscará. Em uma imagem mais ou menos complexa, onde segmentos se confundem pelo efeito do campo visual da LIGHTPEN (ver Fig.2.1a ), o operador, não raramente, selecionará segmentos incorretamente. Desta forma, é necessário uma nova tentativa cuja efetivação provocará a mudança de estado dos dois segmentos

selecionados pelas tentativas anterior e corrente. A efetivação do comando <fim de ação> causará o retorno do segmento ao estado anterior.

#### . O CASO STRING

Uma tecla especial, de RUBOUT, é utilizada com o objetivo de corrigir caracteres introduzidos por erro do operador. Um segundo tipo de erro seria a tentativa de se introduzir mais caracteres do que o máximo definido; isto é solucionado fazendo com que o monitor ignore estes caracteres.

### 5.7. A TRANSFERÊNCIA DE INFORMAÇÃO ENTRE DISPOSITIVO LÓGICO E PROGRAMA DE APLICAÇÃO

As informações presentes nos BUFFERS de entrada do LOCATOR, VALUATOR e PICK não são utilizadas pelo programa de aplicação, na forma em que estão armazenadas nos mesmos; em mapeamento relacionando estas informações deve estar presente. A realização deste mapeamento requer um número reduzido de operações, sendo as mais complexas de multiplicação e divisão reais. Nos processadores que dispõem desta facilidade, sua realização deve ser local. Em nosso caso, estas operações seriam feitas em software, e diante das limitações de memória, isto é infactível; assim, o hospedeiro será o responsável por sua realização

#### 5.7.1. A INFORMAÇÃO DE LOCATOR

O BUFFER do LOCATOR contém informações das coordenadas em unidades do dispositivo (em nosso caso unidades de "raster") e devem ser transformadas para coordenadas do usuário (WC), a fim de serem utilizadas pelo programa de aplicação. Esta transformação é executada em dois estágios: No primeiro, o ponto é mapeado de unidades do dispositivo para coordenadas normalizadas (NDC), levando-se em consideração as dimensões da superfície visível do DISPLAY; no segundo, mapeia-se NDC para WC, consultando-se o GKS STATE LIST sobre a WINDOW atual. Portanto, o procedimento é o inverso ao da geração das primitivas gráficas de saída, descrito no

## Capítulo 3.

### 5.7.2. A INFORMAÇÃO DE VALUATOR

O BUFFER de VALUATOR contém a abscissa  $x$ , em unidades do dispositivo; esta deve ser transformada segundo a equação:

$$\text{VALOR} = ((x - XI)/(XF - XI)) (LS - LI) + LI, \quad \text{onde:}$$

XI (XF): representa a posição inicial (final) do segmento de reta que simula o potenciômetro deslizante.

LI (LS): representa o limite inferior (superior) dos possíveis valores assumidos pela variável VALOR, definidos pelo programa de aplicação.

$x$ : corresponde a posição corrente do marcador,  $XI \leq x \leq XF$ .

VALOR: variável real que representa o valor passado ao programa de aplicação; é descrita por uma equação linear.

### 5.7.3. A INFORMAÇÃO DE PICK

O BUFFER de PICK contém o endereço, no "display processor program", do item apontado. Como a informação preconizada pelo GKS é o SEGMENT NAME e o PICK IDENTIFIER, uma associação endereço/informação é necessária; uma parte do mapa de correlação se incumbirá desta tarefa.

## 5.8. O MAPA DE CORRELAÇÃO

Na Seção 4.3.1. foi descrito um mapa de correlação em linhas gerais; aqui preocupar-nos-emos com a implementação de um mapa específico. Como utilizamos um "display line-drawing" com regeneração e somente a LIGHTPEN como dispositivo da classe PICK, e o "display processor program" tem uma estrutura linear, a associação endereço/informação é implementada de uma maneira simples. A Figura 5.4 mostra a tabela que relaciona os campos necessários para esta associação. Nesta tabela os campos têm os seguintes sig

nificados:

- . Campo NOME : Contém os SEGMENT NAMES.
- . Campo APTK : Contém os apontadores para os primeiros PICK IDENTIFIERS dos segmentos.
- . Campo PICK : Contém os PICK IDENTIFIERS dos segmentos.
- . Campo END : Contém os endereços, no "display processor program" dos PICK IDENTIFIERS.

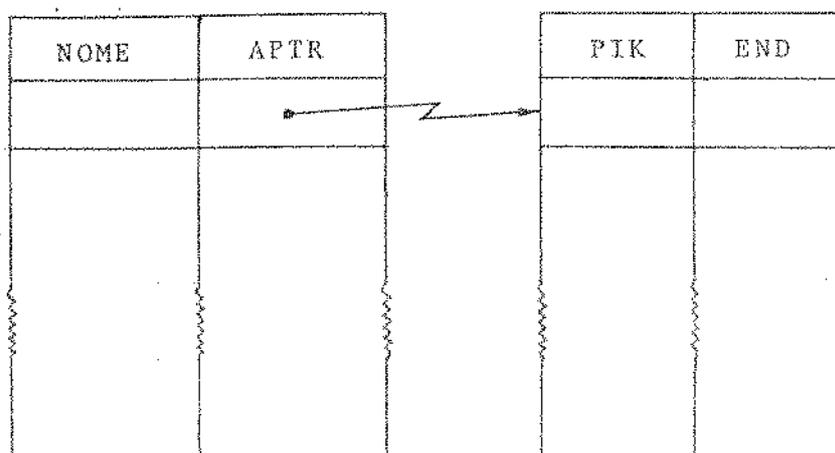


FIG. 5.4 - Tabela de pick do mapa de correlação

Se o endereço é conhecido, a identificação do SEGMENT NAME e PICK IDENTIFIER é obtida por uma simples busca binária. Este procedimento é simples e a quantidade de memória utilizada é pequena, quando comparados ao que seria necessário caso estivéssemos utilizando um dispositivo da classe LOCATOR para simular a classe PICK.

Desde que o campo NOME deve conter até oito caracteres, o número de bytes necessários para se armazenar localmente a tabela de pick é dado por:

$$NBYTES = 10 * NSEG + 4 * NPIK , \text{ onde}$$

NSEG : representa o número de segmentos da imagem.

NPIK : representa o número de "pick identifier" na imagem.

Assim, a quantidade de memória requerida para se armazenar a tabela de pick está diretamente relacionada à estruturação da imagem, a qual é difícil de se prever para diferentes aplicações. O que é mais fácil de se prever, em nosso caso, é o número de vetores presentes na imagem, pois este é limitado por capacidades de hardware do terminal. A Fig.5.5 apresenta os limites dentro dos quais não ocorre cintilação da imagem. Baseado nesta figura, acreditamos que seja razoável supor uma imagem composta por 700 vetores. Considerando que 3/4 destes vetores devem ser identificados por seus PICK IDENTIFIER e 1/4 pelo segmento (o que não é o pior caso), teríamos que dispor de quase 4000 bytes apenas para o armazenamento da tabela de pick; isto é impraticável dentro dos atuais limites de memória disponíveis. Desta forma, todo o mapa de correlação será implementado no hospedeiro. MOLINARO /14/ discute este mapa incluindo as partes pertencentes à saída.

#### 5.9. O DECODIFICADOR

Todas as mensagens entre hospedeiro e periférico devem passar por uma decodificação, ou seja, este bloco analisa os códigos dependentes do dispositivo (DD) e ativa as rotinas correspondentes a cada código. Por exemplo, o envio da seqüência dos bytes 175<sub>g</sub>, 154<sub>g</sub>, ativará o bloco de controle que atuará na tabela de estado, tornando o dispositivo LOCATOR apto a operar no modo SAMPLE. A Tabela 5.1 contém as relações entre os códigos ID e DD para cada dispositivo de entrada, onde a parte DD é codificada em ASCII.

Observe da Tabela 4.1 que embora o parâmetro "Device Number" (DN) também seja necessário, foi omitido pelo fato de existir apenas um dispositivo em cada classe.

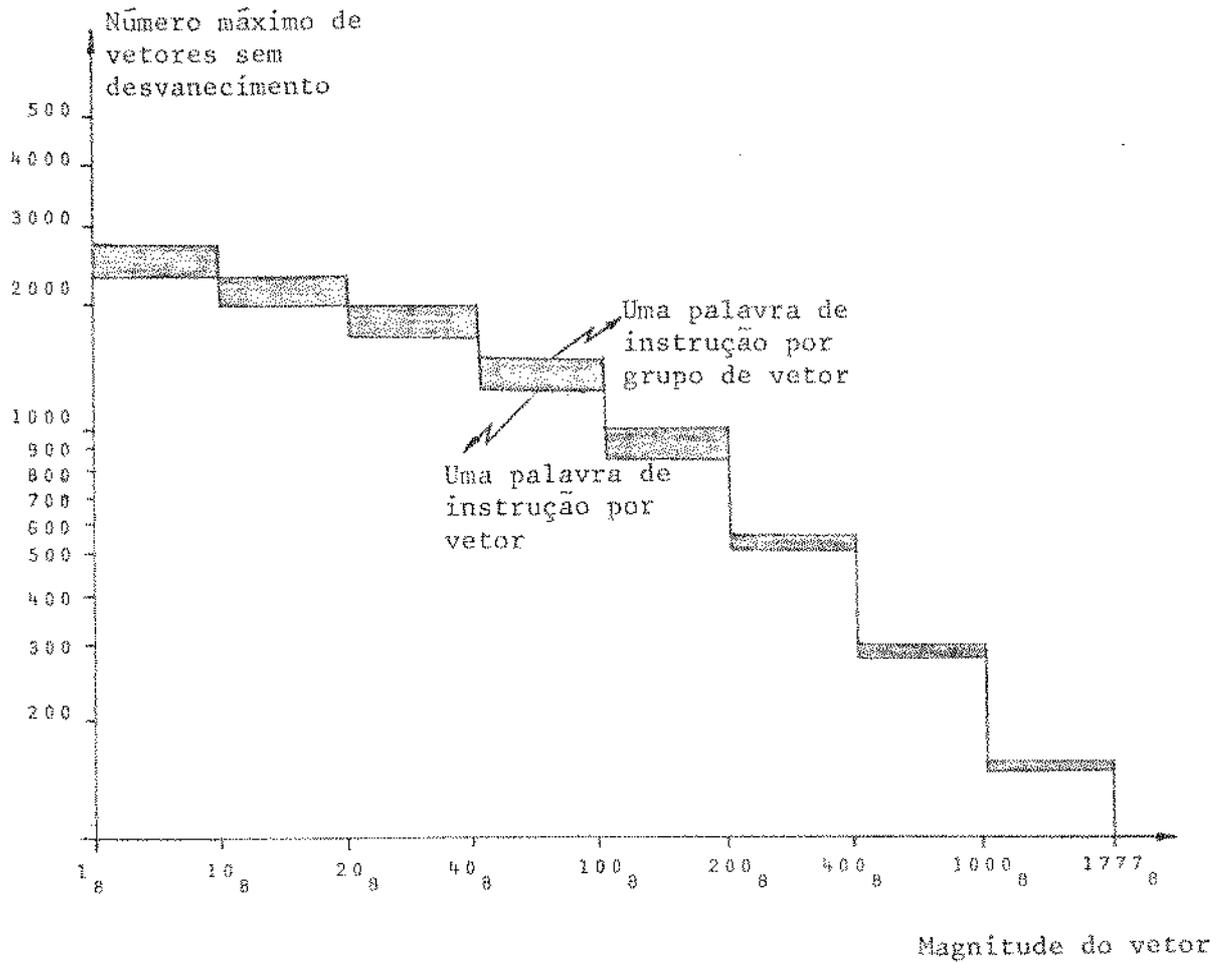


FIG. 5.5 - Número de vetores/magnitude para operação sem desvanecimento.

X		DEVICE CLASS (DC)				
CÓDIGO ID		LOC	VAL	CHO	PIK	STR.
100		{,A	{,B	{,C	{,D	{,E
200		{,F	{,G	{,H	{,I	{,J
300	MODO 0	{,K	{,N	{,Q	{,T	{,W
	MODO 1	{,L	{,O	{,R	{,U	{,X
	MODO 2	{,M	{,P	{,S	{,V	{,Y

TAB. 5.1 - Conversão ID/DD em dispositivos de entrada

#### 5.10. A TABELA DE ESTADO

Como estamos trabalhando com cinco dispositivos de entrada, em três modos, serão necessários quinze "flags" para controlá-los. Adicionamos a estes os cinco responsáveis pelas informações de pedido de REQUEST. A seguir, apresentaremos os "flags" relacionados ao dispositivo LOCATOR;

- . PRELOC : Quando "set", informa que está havendo pedido de REQUEST
- . REQLOC : Quando "set", o dispositivo está DISABLED.
- . SAMLOC : Quando "set", o dispositivo está ENABLED para SAMPLE.
- . EVELOC : Quando "set", o dispositivo está ENABLED para EVENT.

Segundo o GKS, todos os dispositivos de entrada em uma WORKSTATION são inicializados automaticamente no modo DISABLE. Observe que PRELOC está "set" se e somente se REQLOC está "set".

### 5.11. O CONTROLE DOS DISPOSITIVOS

É composto pelas quinze rotinas que controlam o modo de operação de cada dispositivo de entrada, atuando na tabela de estado. A seguir apresentaremos as três rotinas relacionadas ao dispositivo LOCATOR; as rotinas para os outros dispositivos executam ações similares.

- . DISLOC : Torna o dispositivo DISABLED.
- . SENLOC : Torna o dispositivo ENABLED para operar no modo SAMPLE.
- . EENLOC : Torna o dispositivo ENABLED para operar no modo EVENT.

### 5.12. A TRANSFERÊNCIA DE INFORMAÇÃO

Todos os comandos/dados enviados ao hospedeiro, exceto aqueles que vão diretamente via o bloco TERMINAL, passam pelo bloco ENVIO da Figura 4.4. Existem certos caracteres que são interpretados a priori como caracteres de controle para o sistema operacional do hospedeiro. No caso do PDP-10, se X representa um caractere de controle, de sete bits, sua representação em octal é  $0 < X < 40$ . Embora o PDP-11/05 possua oito linhas de dados para E/S, não podemos enviar comandos/dados na forma de bytes, pois, corremos o risco de um destes bytes ser interpretado como caractere de controle pelo sistema operacional. Uma solução possível e apresentada em /19/, é colocar em cada byte efetivamente transmitido apenas seis bits dos bytes da mensagem original. Isto é vantajoso quando um grande volume de informação deve ser transmitido.

Em nosso caso, levando-se em consideração que o fluxo de mensagens periférico/hospedeiro é pequeno, optamos por uma codificação com quatro bits com significado em cada byte transmitido. A codificação e decodificação são triviais e se encontram descritas na Figura 5.6; onde:

I(S): São os quatro bits menos (mais) significativos do byte a ser enviado.

Com este procedimento, nenhum byte de comando/ dado será confundido como caractere de controle.

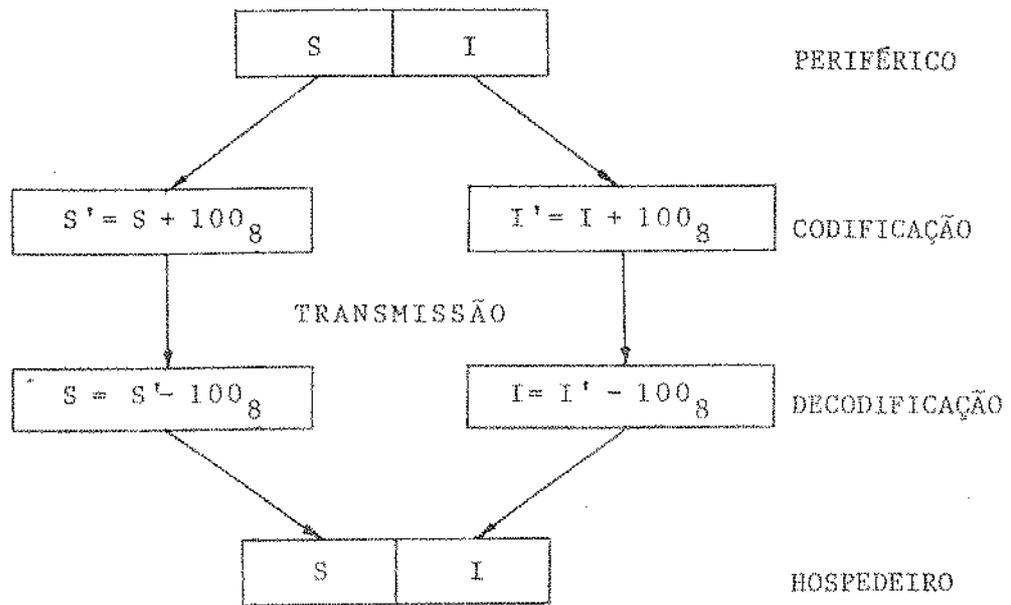


FIG. 5.6 - Codificação e decodificação no procedimento de transmissão.

A seguir apresentaremos, em linguagem natural, o procedimento usado na implementação do envio de informações gráficas ao hospedeiro.

#### PROCEDIMENTO ENVIO

```

BEGIN
    CHESUM = 0
    (Transmitida N)
    IF (N. NE. 0) THEN
        BEGIN
            CHESUM = CHESUM + N
            (Transmitida N)
            REPEAT
                CHESUM = CHESUM + (byte de número N)
                (Transmitida byte de número N)
                N = N - 1
            UNTIL (N = 0)
            (Transmitida CHESUM)
        END
    END
END
  
```

- . N : Variável que contém o número de bytes a ser enviado.
- . CHESUM: Variável que contém a soma aritmética de todos os bytes enviados; é usada para detectar erros na transmissão.

### 5.13. O PEDIDO DE COMANDOS/DADOS

Este bloco está ligado às ações de diálogo, onde o programa de aplicação solicita comandos/dados; esta solicitação pode requerer ou não ações do operador. Em nosso trabalho, o bloco "pedido de dados" é constituído de três sub-blocos:

1. sub-bloco de REQUEST
2. sub-bloco de SAMPLE
3. sub-bloco de atualização da fila de eventos

Os dois primeiros deverão estar presentes em qualquer implementação que inclua os modos de entrada REQUEST e SAMPLE. O sub-bloco de REQUEST é composto por cinco rotinas que atuam na tabela de estado, controlando os pedidos de REQUEST. Por exemplo, a rotina PRELOC controla o pedido de REQUEST para LOCATOR. O sub-bloco SAMPLE é composto por cinco rotinas que atuam nos cinco BUFFERS dos dispositivos de entrada permitindo que as informações contidas nestes BUFFERS sejam enviadas ao hospedeiro. Por exemplo, a rotina SRELOC atua no BUFFER do LOCATOR, e devolve as coordenadas correntes deste dispositivo em unidades de "raster".

A seguir apresentaremos, em linguagem natural, o procedimento usado na implementação do pedido de comandos/dados.

## PROCEDIMENTO PEDIDO DE COMANDOS/DADOS

```

BEGIN
  IF (Pedido é do tipo REQUEST)           THEN
    BEGIN
      ("Set" o "flag" correspondente na tabela
       de estado, informe ao operador, e espere
       fim de ação)
    END
    ELSE
      BEGIN
        (Transfira o conteúdo do BUFFER de entrada
         do dispositivo para o hospedeiro)
      END
    END

```

## 5.14. O BUFFER DE EVENTOS

O BUFFER de eventos ou fila secundária de eventos surge como alternativa para solucionar o problema da comunicação interface de entrada/ fila de eventos, como apresentado por DALTRINI et al. /13/.

Em sistemas operando em "time-sharing" não é concebível interrupções assim que os eventos são gerados. Desta forma, é necessário criar um BUFFER que armazene os eventos na ordem em que foram gerados pelos diversos dispositivos de entrada, nas diversas WORKSTATIONS. A fila de eventos pode ser atualizada sempre que uma função do tipo AWAIT ou FLUSH ocorra. A Figura 5.7 ilustra as relações lógicas entre o GKS e as entradas por evento.

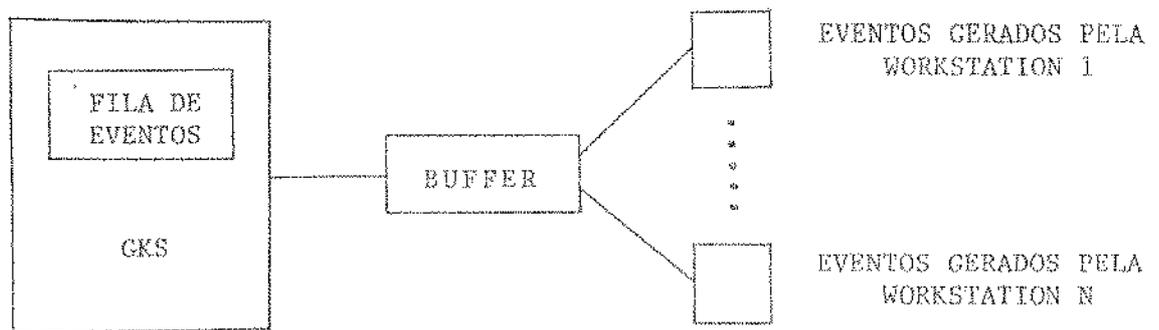


FIG. 5.7- Aquisição de eventos das WORKSTATIONS pelo GKS

Desde que os eventos devem ser armazenados na ordem cronológica de sua geração, uma fila circular pode ser utilizada para implementar a fila de eventos. Esta pode estar em um processador independente, ou em uma WORKSTATION operando com capacidade ociosa. Em nosso caso, foi implementada no próprio PDP-11/05, mais pela inexistência de outro processador e pela necessidade de se testar as rotinas, do que por sua operação ociosa. Existe uma farta literatura sobre algoritmos que implementam filas circulares /21,22/ e não os acrescentaremos a este trabalho.

O problema de "overflow" do BUFFER pode ser solucionado de várias maneiras; contudo, o melhor é assegurar que isto nunca aconteça. Em nosso trabalho, definimos um BUFFER de 400 bytes; se o usuário julgar este número insuficiente, os endereços do programa monitor poderão ser alterados a fim de aumentá-lo; tão logo o BUFFER se complete, uma mensagem é enviada ao operador. Acompanhando cada comando/dado no BUFFER deverá estar presente a tripla identificadora do dispositivo, ("Workstation Identifier", "Device Class", "Device Number"), como descrito no Capítulo 3. Os comandos de STRING são precedidos também pelo número de caracteres que o compõem. A Tabela 5.2 mostra a convenção adotada para a identificação dos cinco dispositivos na WORKSTATION GT40, sendo a tripla identificadora codificada em ASCII.

X		DC				
WI	DN	LOC	VAL	CHO	PIK	STR
60	61	61	62	63	64	65

TAB. 5.2 - Identificação dos dispositivos na "Workstation".

Vamos exemplificar a representação interna do BUFFER para a seguinte seqüência de eventos: do dispositivo STRING com os caracteres ZOOM; do dispositivo LOCATOR com as coordenadas  $X = 1231_8$  e  $Y = 73_8$ ; e do dispositivo PICK com o endereço  $25502_8$ .

A Figura 5.8 ilustra tal representação, considerando-se uma palavra de 16 bits e endereçamento por bytes.

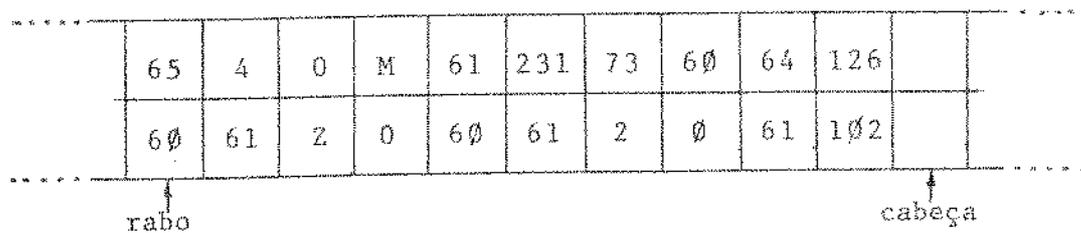


FIG. 5.8 - Representação de eventos em uma fila circular.

### 5.15. O TERMINAL

Neste momento, acreditamos que seja suficiente complementar o que existe descrito na Seção 4.3.1. Este procedimento está baseado no programa de "bootstrap" armazenado em ROM, no próprio terminal gráfico GT40 /19/.

Como o monitor gráfico não tem caractere de controle, as mensagens enviadas pelo hospedeiro não necessitam de codificação, e podem ser transmitidas byte a byte, sem o "overhead" que seria introduzido ao se transmitir separadamente bits de um mesmo byte.

Como o operador utiliza o TECLADO para introduzir informações do dispositivo STRING, e para controlar diretamente o

sistema, é óbvia a necessidade de se providenciar uma maneira do monitor determinar o que o operador deseja fazer. Se o primeiro caractere teclado é CONTROL G <↑G>, os próximos deverão ser enviados ao hospedeiro sem qualquer influência do modo de operação do dispositivo STRING, até que um CARRIAGE RETURN <CR> ocorra.

#### PROCEDIMENTO TERMINAL

BEGIN

WHILE (NOT.ITRUE) . AND . (NOT.OTRUE) DO

BEGIN

(Fique em "loop" esperando ação do operador ou mensagem do hospedeiro)

END

IF (OTRUE) THEN

BEGIN

(Decida o que fazer com o dado do teclado. "Reset" OTRUE)

END

ELSE

BEGIN

(Busque a mensagem que o hospedeiro está enviando. "Reset" ITRUE. Ative o bloco DECODIFICADOR. Se nenhum procedimento é ativado, a mensagem do hospedeiro deve ser mostrada)

END

END

- . OTRUE : Variável lógica. Se VERDADEIRA, sinaliza que um caractere deve ser lido do teclado.
- . ITRUE : Variável lógica. Se VERDADEIRA, sinaliza que o hospedeiro enviou um caractere.

## 6. CONSIDERAÇÕES FINAIS

Este trabalho e o desenvolvido em paralelo por MOLINARO /14/ são complementares entre si, e as implementações correspondentes formam o subsistema para geração de dados dependentes do dispositivo para uma WORKSTATION interativa, o GT40. Estes dois trabalhos definem o DEVICE DRIVER descrito no Capítulo 4., cuja implementação deverá ser utilizada como parte do sistema global apresentado na Figura 3.2, tão logo o Núcleo Gráfico - GKS e as interfaces do usuário e da base de dados estejam implementados. A seguir, discutiremos alguns aspectos que dizem respeito ao modelo do monitor gráfico proposto no Capítulo 4. e as questões de implementação.

O conceito de dispositivos lógicos apresentado no Capítulo 2., permite o desenvolvimento de software gráfico independentemente dos dispositivos físicos através dos quais as informações são geradas; o modelo para o monitor gráfico mostra de forma clara que entradas e saídas gráficas podem ser tratadas independentemente umas das outras. Uma vez que o monitor apresentado visa atender essencialmente aos aspectos relacionados à entradas gráficas, ou seja, atende apenas ao mínimo de processamento local necessário para interações, o mesmo pode ser aplicado a qualquer tipo de WORKSTATION interativa. O seu caráter modular facilita a implementação, pois a função de cada rotina é facilmente identificável; além disso, é evidente a sua extensibilidade a processadores com maior capacidade local.

Para a implementação do monitor gráfico proposto foi utilizada a linguagem "assembly" do PDP-11, pois além da pequena quantidade de processamento local, uma linguagem de alto nível não permite a plena utilização dos recursos de um computador para as operações necessárias. Se há uma grande quantidade de processamento local, uma linguagem de alto nível pode ser atrativa; além disso, a implementação do bloco TERMINAL poderá requerer uma estrutura mais elaborada e eficiente, para que não haja perda de informações provenientes do hospedeiro, provocada pelo incremento das tarefas executadas pelo processador local.

Finalmente, descrevemos a maneira pela qual todas as rotinas implementadas foram testadas.

O fato do Núcleo Gráfico - GKS estar em fase de implementação, não impede que sejam feitos testes a fim de se verificar o funcionamento do monitor implementado. Estes testes foram subdivididos em duas etapas. Na primeira, via TECLADO, e utilizando o bloco TERMINAL, o usuário informa ao hospedeiro os modos de operação dos dispositivos; a partir deste ponto o usuário pode introduzir localmente comandos/dados como se o sistema gráfico estivesse implementado. Na segunda são criados, através das facilidades do sistema operacional do DEC-10, arquivos com informações gráficas geradas no periférico; estes arquivos são examinados para se verificar a coerência com os dados gerados.

## BIBLIOGRAFIA

- /1/ GILOI, W.K. - Interactive Computer Graphics. Prentice-Hall, New Jersey, 1978.
- /2/ CRESTIN, J.P. and M.LUCAS - What Might be Computer Graphics ? , Metodology in Computer Graphics, ed. Guldj, R.A. and H.A.Tucker, North Holland, 1979.
- /3/ HOPGOOD, F.R.A. - Is a Graphics Standard Possible?, *ibid*
- /4/ FOLEY, J.D. - A Standard Computer Graphics Subroutine Package, Computer Structures, Vol. 10, 1979, pp. 141-147.
- /5/ BAECKER, R. - Towards an Effective Characterization of Graphical Interaction. Paper submitted to the Seillac II Workshop on the Methodology of Interaction - August 1978.
- /6/ Status Report of the Graphics Standard. Planning Committee of ACM/SIGGRAPH. Computer Graphics, Vol.11, n<sup>o</sup>3, 1977.
- /7/ Information Processing Graphical Kernel System (GKS), Functional Description version 6.2.
- /8a/ NEWMAN, W.N., and R.F.SPROLL - Principles of Interactive Graphics. Mc Graw-Hill, New York, 1973.
- /8b/ NEWMAN, W.M., and R.F.SPROLL - Principles of Interactive Graphics. Mc Graw-Hill, New York, 1980.
- /9/ OHLSON, M. - System Design Considerations for Graphics Input Devices, IEEE Computer, Nov. 1978, pp. 9-18.
- /10/ DEECKER, G.F.P. and J.P.PENNY - Standard Input Forms for Interactive Computer Graphics, Computer Graphics, Vol.11, n<sup>o</sup> 1, Spring 1977, pp. 32 - 40.
- /11/ FOLEY, J.D. and V.L.WALLACE - The Art of Natural Man/Machine Conversation, Proc. IEEE, Vol. 62, n<sup>o</sup> 4, April 1974, pp. 462-471.

- /12/ WELLER, D.L., E.D. CARLSON, G.M. GIDDINGS, F.P. PALERMO, R. WILLIAMS and S.N. ZILLES - Software Architecture for Graphical Interaction, IBM System Journal, Vol. 19, Nº 3, 1980, pp. 314 - 330.
- /13/ DALTRINI, B.M., R.P. MARTINS, L.F.R. MOLINARO, M. JINO and M.L. ANDRADE NETO - Graphical Kernel System for Process Control.
- /14/ ENCARNAÇÃO, J., G. ENDERLE, K. KANSY, G. NEES, E.G. SCHLECHTENDAHL, J. WEISS and P. WIBKIRCHEN - The Workstation Concept of GKS and the Resulting Conceptual Differences to the GSPC Core System, Quarterly Reporter of SIGGRAPH/ACM, Computer Graphics, vol. 14, Nº 3, 1980, pp. 226 - 229.
- /15/ ENCARNAÇÃO, J., G. NEES - Recommendations on Methodology in Computer Graphics, Methodology in Computer Graphics, ed. GUEDJ, R.A. and H.A. TUCHER, North Holland, 1979.
- /16/ MILLER, R.B. - Response time in man-computer conversational transactions in 1968, Fall Joint Computer Conf. AFIPS Conf. Proc., Vol. 33, pt. I. Montreale, N.J.: AFIPS Press, 1968, pp. 267 - 277.
- /17/ CUNHA, J.D. - Sistema Gráfico Interativo Configurável para diferentes aplicações, Tese apresentada ao concurso para especialista do Laboratório Nacional de Engenharia Civil, Lisboa, Março de 1981.
- /18/ Guia do Usuário do GT40, Digital Equipment Corporation - Maynard, Massachusetts, 2nd printing, September 1973.
- /19/ PRINCE, M.D. - Interactive Graphics for Computer - Aided Design Reading, Mass : Addison - Wesley, 1971
- /20/ HOROWITZ, E. and S. SAHNI - Fundamentals of Data Structures. Computer Science Press, INC., Potomac, Maryland, 1976.
- /21/ BERZTISS, A.T. - Data Structures - Theory and Practice. Academic Press, INC. New York, 1971.