UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

JULHO DE 1979

UM TERMINAL GRÁFICO INTERATIVO BASEADO NA TECNOLOGIA DO 'TV RASTER'

Por : Clésio Luis Tozzi

Orientador : Prof. Dr. Manuel de Jesus Mendes

Tese apresentada à Faculdade de Engenharia FEC-UNICAMP como parte dos requisitos exigidos para obtenção do título de DOUTOR EM ENGENHARIA

AGRADECIMENTO

O agradecimento especial ao orientador Prof. Dr. Manuel de Jesus Mendes e ao Prof. Dr. José Encarnação pelas ideias, discussões, incentivo, apoio e oportunidades oferecidas.

Aos Prof. Dr. Mario Jino, Prof. Dr. Márcio Andrade, Prof. Dr. W. Strasser, Rolf Lindner e Leo Magalhães pelas frutíferas discussões.

Aos colegas Beatriz Daltrini e Paulo Berardi, e a Miriam Chinelato e Rose Levy pelo apoio na edição das figuras e textos aquí apresentados.

Ao CNPq, UNICAMP (BRASIL) e ao GMD, TH DARMSTADT (ALEMANHA) pelo apoio financeiro.

A todas as demais pessoas e entidades que de algum modo colaboraram para o desenvolvimento deste trabalho.

RESUMO

O trabalho trata do desenvolvimento de um terminal gráfico interativo baseado na tecnologia do 'TV Raster', com estrutura dinâmica da 'picture memory' e conversão em tempo real do 'display file', possibilitando a rápida atualização da imagem, requisito necessário às aplicações interativas como: controle de processos, CAD, etc.

ÍNDICE

CAPÍTULO 1 - INTRODUÇÃO

Apesar do constante desenvolvimento que vem ocorrendo na tecnologia dos semicondutores, permitindo o desenvolvimento de sistemas de computação cada vez mais potentes e velozes, os meios de comunicação destas máquinas com o meio exterior (entrada e saída de dados) ainda representam um ponto de estrangulamento que impede a obtenção de maior eficiência e dificulta sobremaneira seu uso.

Dentre os meios usados para essa comunicação, os 'displays' baseados nos CRT (tubo de raios catódicos) vem se apresentando como um dos mais eficientes dispositivos de saída de dados (terminais alfanuméricos) e nos últimos anos a inclusão de capacidade gráfica a tais dispositivos facilitou a comunicação com o meio exterior (figuras mostram-se como uma linguagem mais natural), aumentou a taxa de saída de dados, e tornou possível a entrada de dados através de processos até então não convencionais como o 'light-pen', os 'touch panels', etc.

Estes dispositivos, de modo geral, baseiam-se nos 'line drawing displays' e nos 'storage tube displays', exigindo circuitos sofisticados e em alguns casos não preenchem de modo completo os requisitos exigidos para certas aplicações como controle de processos em tempo real. O uso da tecnologia de televisão, barata e altamente desenvolvida, deu origem aos ' TV Raster displays', que geram a imagem através da varredura sequencial das linhas da tela. O principal obstáculo ao uso desta tecnologia, reside na dificuladade de transformação da descrição da figura ('display file') nos pontos correspondentes da tela dentro do tempo disponível para isso. Várias soluções são usadas na tentativa de contornar-se tal restrição e entre elas podem ser citadas os displays celulares, o uso do 'frame buffer', etc, todas elas, de modo geral, impossibilitando o uso de tais dispositivos em aplicações nas quais se exige alta velocidade de resposta e atualização da imagem.

Uma solução definitiva para os problemas dos 'raster displays' reside no desenvolvimento de um conjunto capaz de realizar a conversão citada, observando as restrições de tempo impostas. Na literatura nenhuma solução defintiva foi apresentada até agora, e o que se busca neste trabalho é, principalmente, o desenvolvimento da arquitetura de um terminal gráfico inteligente do tipo 'raster', capaz de preencher a maioria dos requisitos necessários para aplicações em tempo real.

No capítulo dois deste trabalho são expostos os principais requisitos necessários aos sistemas interativos utilizados no controle de processos, concluindo-se pela necessidade de dispositivos gráficos. Discutem-se, também as principais tecnologias para implementação destes dispositivos gráficos, assim como processos de entrada e saída de dados para os mesmos.

No capítulo tres apresenta-se a arquitetura de um terminal gráfico interativo baseado na tecnologia do 'TV raster' onde, para a maioria das aplicações, foram eliminadas as restrições de conversão do 'display-file' em tempo real, e de atualização da imagem, pelo desenvolvimento de estruturas hardware operando em alta velocidade. Este capítulo está dividido em cinco seções:

- o conversor em tempo real e o acumulador;
- a 'picture memory' e seu controlador;
- a interface da 'picture memory' com o microprocessador usado para controle;
- o software mínimo necessário para o sistema;
- a validação do 'hardware' e 'software' por simulação;

que constituem, basicamente o corpo deste trabalho, e foi desenvolvido através da frutífera cooperação do autor deste com R. Lindner (TH Darmstadt), cabendo a este último a autoria da quase totalidade do primeiro ítem citado acima.

No capítulo quatro expõem-se vários modelos para interação homem-máquina, resumindo-se as principais característi

cas necessárias às interfaces usadas para esta finalidade. Discute-se também a adaptabilidade da estrutura do terminal proposto, que preenche quase todos os requisitos exigidos, o que recomenda seu uso para controle em tempo real para os mais diversos tipos de processos.

No capítulo cinco resumem-se as principais conclusões deste trabalho e apresentam-se sugestões para pesquisas futuras.

CAPÍTULO 2 - A INTERFACE HOMEM-MÁQUINA NO CONTROLE DE PROCESSOS

No transcorrer dos últimos anos a automatização dos processos industriais tem aumentado significativamente e o homem, que anteriormente tinha sua posição junto às máquinas, foi sendo deslocado para salas de controle remotas onde, a partir de informações obtidas através de 'displays', impressoras e outros dispositivos, pode, quase que simultaneamente, manter o controle sobre diversos processos concomitantes.

Tal nível de automatização só foi possível de se atingir graças ao rápido desenvolvimento da tecnologia dos semicondutores, e o advento dos microprocessadores que tornou economicamente viável o processamento distribuido e consequentemente o controle distribuido de processos num grande número de aplicações |1,2|.

Paralelamente ao desenvolvimento dos semicondutores, também as técnicas de comunicação visual estiveram em desenvolvimento. Os CRTs (tubos de raios catódicos) que sempre se mostraram o meio mais apropriado de comunicação homem-computador-homem foram substituidos pelos tubos de TV. Dispositivos de entrada de dados como a 'light-pen', 'joystick' e o uso de conjuntos de caracteres especiais, através de teclados orientados para aplicações específicas foram sendo gradativamente introduzidos, dando origem aos 'displays' atuais que são hoje peça essencial em qualquer sala de controle ou monitoramento. Esse fato obrigou o desenvolvimento de novas técnicas e definições para a realização do diálogo homem-máquina.

2.1 - Considerações sobre os sistemas interativos

As funções atribuidas aos sistemas interativos podem ser classificadas em três grupos |3|:

- Supervisão e controle de processos;
- Tratamento de falhas;

. Aperfeiçoamento e avaliação de sistemas.

As funções do primeiro grupo referem-se à supervisão e controle de processos em execução, pela detecção e correção de possíveis desvios das variáveis internas do sistema. As do segundo grupo dizem respeito ao tratamento de situações anormais provocadas por falhas em equipamentos e computadores. As ações relacionadas com este grupo consistem na detecção, identificação , compensação e correção de falhas. As do terceiro grupo visam o constante aperfeiçoamento de métodos e equipamentos existentes, através da avaliação dos seus desempenhos e da sugestão de novas idéias para seu projeto e utilização.

Dentro de cada um destes grupos a atribuição das funções para o homem ou a máquina sugere uma classificação de um sistema de acordo com a tabela a seguir:

. Operação manual;
. Operação automática;
. Operação manual assistida pelo computador.

Dentre os três modos sugeridos o último deles, operação manual assistida pelo computador, é o mais comumente usado, uma vez que os sistemas puramente automáticos apresentam grandes dificuldades de implementação enquanto os puramente manuais apresentam baixa eficiência. Além disso a operação manual assistida pelo computador apresenta a vantagem de associar as capacidades complementares do homem e da máquina evitando seus pontos fracos. Deste modo ações frequentes devem ser alocadas à máquina enquanto as ações que raramente ocorrem, por exemplo a tomada de decisões durante falhas ou momentos críticos, o reconhecimento de padrões, predição, dedução, improvisação devem ser alocadas ao homem, uma vez que tais ações difícilmente podem ser modeladas ou definidas.

Os sistemas interativos devem ser ainda classificados de acordo com o modo de operação ou seja:

. Operação 'on-line'
. Operação 'off-line'

cada uma delas exigindo diferentes taxas na demanda de informação.

A maioria das operações 'on-line' necessita de rápida atualização da informação externa, para uso do operador, assim como rápidas ações em respostas às exigências do processo. Estas ações são realizadas através dos dispositivos de entrada. Deste modo a primeira atividade no projeto de um sistema interativo deve ser um estudo dos dispositivos de entrada e saída e de técnicas que permitam a melhor exploração dos mesmos.

2.2 - Dispositivos de entrada e saída

Dentre os dispositivos de saída de dados de um computador os mais comuns são os 'displays' baseados nos CRT ou tubos de TV que já substituiram quase que completamente os teletipos e impressoras. Para uma avaliação mais profunda das vantagens e limitações do seu uso em sistemas interativos, alguns de seus parâmetros diretamente relacionados com os fatores humanos devem ser analisados. Dentre os criterios óticos temos:

Luminância: é a intensidade de uma superficie medida em candela por metro quadrado.Para uma superficie sem fontes internas de luz a luminância é igual ao coeficiente de reflexão vezes a iluminação, onde iluminação é medida em lumens por metro quadrado. Brilho é a luminância perceptível. CRTs com luminância entre 80 e 160 são aceitáveis, enquadrando-se nessa faixa a maioria dos CRTs disponíveis.

Contraste:é a razão entre a soma das luminâncias do caracter e do fundo pela luminância do fundo. Contrastes entre 15 e 30 são aceitáveis. A iluminação do ambiente influi consideravelmente no contraste, principalmente em áreas altamente iluminadas. O contraste é um fator importante do projeto de sistemas interativos uma vez que o seu efeito no desempenho humano é maior do que o da luminância.

'Flicker': é o fenômeno que ocorre quando a taxa de regeneração é menor que a frequência de fusão da imagem . Este fenômeno depende da persistência do fósforo da tela. A frequencia ideal de regeneração encontra-se em torno dos 50Hz.

Acuidade visual: considerando-se um ângulo de três minutos para a acuidade visual determina-se um tamanho mínimo de 3.05mm para os carateres. Experiências mostram que caracteres maiores não aumentam consideravelmente o desempenho. O tamanho da matriz de pontos para a formação dos caracteres (normalmente 5 x 7) é um fator importante relacionado com o desempenho humano.

Cursor: cursores retangulares mostram-se como melhor adaptados para informação alfanumérica, enquanto pontos ou cruzes apresentam melhor desempenho para aplicações gráficas. A frequência ideal de oscilação encontra-se nos 3Hz.

Além dos critérios óticos para análise dos 'displays', outros relacionados com segurança, compatibilidade e aplicações específicas também devem ser considerados. A tabela 1.1. apresenta um resumo destes critérios |4|, a maioria auto-explicativos, tornando-se dispensável maiores comentários sobre os mesmos.

Apesar da variedade de alternativas quanto aos dispositivos de entrada de dados muitos deles não são apropriados a tarefas gerais, restringindo a escolha a poucos deles. O mais comum deles, o teclado, não se apresenta como um dispositivo ergonômico sob o ponto de vista dos fatores humanos, devido a sua forma geométrica. Em aplicações industriais as teclas são muito numerosas e pequenas para o operador que normalmente não se senta junto ao teclado como um datilográfo. Teclados como os usados em telefone, do tipo 4 x 3, são uma solução melhor, embora limitem sobremaneira a linguagem de comando a ser usada. Outros dispositivos comuns são a 'light-pen', 'mouse' e 'tablet' . Estes dispositivos, de grande utilidade nas aplicações gráficas, em CAD ('computer aided design') ou CAM (computer aided manufacturing'), mostram-se valiosos instrumentos para aplicação em escritórios e laboratórios, mas perdem grande parte da flexibilidade quando não se encontram sempre à mão do utilizador.

Os 'touch panels' que vêm se popularizando e que permitem indicar o comando desejado com um simples toque no painel, bem como os 'trackbals' e 'joysticks', mostram-se úteis

Critério ótico	Brilho Contraste Resolução Cor
Critérios de segurança	Precisão Realização Duração
Critérios do sistema	Desempenho Tensão de trabalho Tempo de chaveamento Compatibilidade Fluxo de dados
Critérios de aplicabilidade	Resistência ao choque e vibração Resistência à temperatura Resistência à humidade Peso Volume

Tabela 1.1 - Critérios de avaliação para os 'displays'

para o posicionamento de objetos, mas são pouco flexivêis para realização de diálogos.

Como conclusão sobre a entrada de dados num sistema, pode-se dizer que a escolha do dispositivo é função da tarefa a ser executada. Se o usuário permanece sentado junto a um console, monitorando muitos processos e tomando decisões, a escolha deve recair sobre um teclado padrão e uma 'light-pen'. Por outro lado se as entradas de comando estão distribuidas geográficamente entre os instrumentos a serem controlados, os 'touch -panels' e os teclados do tipo 4 x 3 devem ser escolhidos. É claro que tais indicações tem um carater geral, não se invalidando outras indicações para aplicações específicas.

2.3 - Diálogo e linguagem de comando

O acesso do usuário ao sistema e a suas facilidades é feito de modo geral através de uma linguagem de comando que pode ser considerada como um dos fatores determinantes na eficiência da utilização do sistema. Esta linguagem de comando deve ser consistente, poderosa, modular , de fácil aprendizado e utilização e dotada de facilidades para a simplificação do diálogo usuário-máquina |5|.

Os comandos de uma linguagem podem ser organizados de dois modos: 1) um grande número de comandos específicos associados a poucos argumentos; 2) um pequeno número de comandos gerais associados a um conjunto vasto de argumentos. Na maioria dos sistemas, que normalmente contam com comandos dos dois tipos citados, os comandos específicos são frequentemente mais usados já que tais comandos devido aos poucos argumentos associados a eles evitam uma sobrecarga da memória do operador e proporcionam assim uma taxa menor de erros.

A linguagem de comando deve ainda proporcionar comandos através dos quais cálculos imediatos ou testes em diferentes pontos do processo possam ser executados, bem como permitir o acesso a arquivos onde informação sobre o processo ou o sistema esteja armazenada.

Tão importante quanto a linguagem de comando é o estilo do diálogo. O diálogo pode básicamente ser realizado de três modos: 1) dirigido pelo usuário ou pelo sistema; 2) através de menus; 3) um diálogo livre.

O diálogo onde a direção é determinada pelo computador e o usuário é levado a uma escolha forçada é um dos mais apropriados para tarefas rotineiras. Este tipo de diálogo permite o uso das técnicas de menu, que demanda pouco tempo do operador na execução do comando, requer pequeno tempo de treinamento e reduz a possibilidade de erros se a entrada consistir de um pequeno conjunto de opções. Já os diálogos dirigidos pelos usuários ou de resposta livre são recomendados para a execução de tarefas mais complexas, uma vez que permitem um diálogo mais flexí

vel. Porém, só devem ser executados por usuários experientes e que possuam razoavel conhecimento do sistema dado que a possibilidade de errros e falhas aumenta consideràvelmente neste caso. É razoável considerar a possibilidade de troca de um modo de diálogo para outro a medida que o desenvolvimento do processo o permita.

O estilo do diálogo é também influenciado por outros fatores como a distribuição da informação, a discriminação desta informação em diferentes classes, o uso de teclados dirigidos, etc. A discriminação da informação pode ser implementada, na saída, pelo uso de diferentes intensidades de brilho, 'blinking', cor,caracteres especiais, letras maiúsculas e minúsculas, a partição da tela em áreas, etc., e, na entrada,pelo uso de teclas representando diferentes coisas, desde frases ou comandos completos até figuras gráficas.

2.4 - Um modelo físico para um sistema interativo

O exposto anteriormente localiza e traz para discussão alguns dos fatores mais importantes relacionados com o projeto de controle de processos através de sistemas interativos. Seguramente muitos outros fatores devem ser considerados, uma vez que este é um campo de pesquisa complexo no qual existem ainda muitos problemas em aberto.Porém cumpre os objetivos idealizados pelo autor que procura simplesmente localizar dentro de um cenário mais amplo pontos cruciais ligados ao projeto de interfaces homem-máquina.

Muitas das características requeridas pelos sistemas interativos são atualmente implementáveis. Contudo, grande parte do progresso em sistemas interativos ainda é barrado pelas limitações das interfaces homem-máquina, não obstante o constante desenvolvimento da tecnologia relacionada com os dispositivos de entrada e saída.

2.4.1 - Terminais inteligentes

Um sistema interativo ideal, em termos

práticos, poderia ser descrito pelo diagrama de blocos da figura 2.1 onde os três módulos prinicipais são:

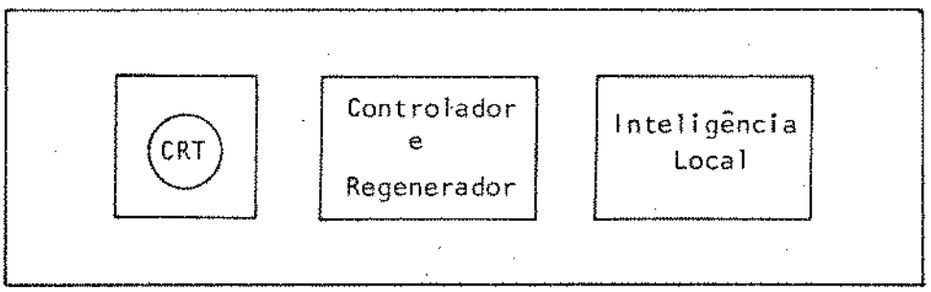


Figura 2.1 - Diagrama de blocos de um sistema interativo

- Unidade de entrada e saída (CRT);
- Unidade de controle e regeneração da imagem;
- Unidade local de cálculo.

que caracterizam os assim chamados terminais inteligentes.

Terminal inteligente é o terminal no qual parte do processamento é realizado por um pequeno computador ou processador contido no proprio terminal. Para ser considerado como um terminal inteligente, a capacidade de computação do mini-computador do terminal deve estar disponível ao usuário, de modo que o mesmo possa programa-lo.

Um terminal inteligente interativo deve possuir as seguintes características básicas |6|:

- Memória local;
- Capacidade de interação com o usuário;
- Programas armazenados;
- Capacidade de processamento local;
- Comunicação 'on-line' com o computador central ou banco de dados;
- Dispositivo de entrada de dados orientados para o ser humano: teclado, 'light-pen', etc.;
- Dispositivos de saída de dados orientados para o ser humano: impressoras, CRT, etc.

As mudanças ocorridas na tecnologia dos

semicondutores tiveram um impacto significativo nos custos relativos dos diversos componentes de um computador e dos sistemas de comunicação. Enquanto os custos dos circuitos e memórias decresceram rápidamente, os custos de comunicação permaneceram praticamente constantes.

Nos sistemas 'time-sharing' e outros sistemas de acesso múltiplo, existem razões para centralização, particularmente de banco de dados, nos casos onde diferentes usuários podem acessar e alterar dados. Entretanto, este não é o caso em controle de processos, onde não existem razões para centralização do processamento. Com o desenvolvimento dos sistemas digitais é economicamente mais atrativo fazer o máximo de computação localmente, utilizando-se terminais inteligentes, de modo a minimizar a necessidade de transmissão de dados e consequentemente os custos relativos a comunicação. Isto é válido no caso de aplicações gráficas, tal como CAD ('computer aided design'), onde é necessária a transmissão de dados em alta velocidade para manter o tempo de resposta ao usuário o menor possível. A possibilidade de se resolver grande parte dos problemas relativos a CAD com processamento local e com um mínimo de transmissão de dados para ou do computador central melhora sensívelmente o tempo de resposta ao usuário, bem como reduz os custos de modo geral. Resumindo, a capacidade de processamento no terminal minimiza os custos (reduz o uso da linha), otimiza o tempo e torna realizaveis tarefas anteriormente inviáveis por restrições impostas pela linha de comunicação.

O terminal inteligente deve ser conectado a um computador central. O programa que está sendo executado e outros frequentemente usados, estão armazenados localmente no terminal juntamente com os dados do usuário. No computador central estão armazenados os programas de utilidade ('libraries'), banco de dados usados por múltiplos usuários e programas que podem ser transferidos ao terminal quando requisitados. O computador central também é responsável pela computação de grandes quantidades de dados que por motivos quaisquer,não possa ser realizado no terminal.

Por tanto, o computador central deve possuir:

- Alta capacidade de memória, para armazenamento de dados e programas relativos aos sistemas periféricos;
- Sistemas periféricos caros ou pouco usados, cujo uso não é economicamente viável nas instalações periféricas;
- Capacidade de processamento e computação para manipulação de dados e resolução de problemas ou de parte de problemas grandes demais para serem manipulados no terminal.

2.4.2 - Terminais inteligentes e sistemas gráficos

Parte das necessidades de um sistema interativo é satisfeita com o uso de terminais alfanuméricos convencionais. A associação dos terminais inteligentes a sistemas com capacidade gráfica pode satisfazer mais um dos requisitos,ou seja, uma representação externa o mais próxima possível da linguagem natural |7|.

Um terminal gráfico inteligente deve incluir as seguintes partes:

- Um pequeno computador com memória local;
- Dispositivo de saída gráfica (CRT):
- Teclado;
- Dispositivo para entrada gráfica;
- Outros periféricos e dispositivos para controle dos mesmos.

Nas aplicações gráficas, a utilização da capacidade local de processamento aplicada aos algoritmos de transformação do 'display file' como, por exemplo, translação da

figura ou criação de vistas em perspectivas, evita competição com outros terminais ligados ao sistema. Do computador central exige-se somente uma supervisão de alto nível, tornando possível a produção de imagens sofisticadas com bom tempo de resposta e baixo custo.

Deve-se ressaltar que a eficiência da comunicação com o usuário é função direta da qualidade de imagem gerada, que por sua vez, é função da tecnologia usada para a apresentação da imagem. Na seção 2.4.3 as principais tecnologais disponíveis e suas características são discutidas.

2.4.3 - 'Displays' gráficos

A maioria dos 'displays' gráficos existentes podem ser classificados em três grupos: os 'Line drawing displays' os 'Storage tube displays' e os 'Raster Scan 'displays' [8,9].

A grande maioria das técnicas gráficas interativas tem sido orientada para os 'Line drawing displays', e a principal razão para isso é que o modo mais simples para geração de imagens no CRT é o deslocamento do feixe de eletrons numa sequência arbitrária de retas.

O maior problema nos 'Line drawing displays' encontra-se no processo de regeneração da imagem. O fósforo emissor de luz do CRT deve ser excitado repetidamente para se obter uma figura estável na tela, o que torna necessária a existência de uma memória de regeneração ('picture memory'), onde é armazenada a informação sobre a figura. O conteúdo desta memória deve, então, ser passado repetidamente ao CRT, pelo processador de 'picture memory', para obter-se a regeneração da imagem na frequência necessária.

Algumas considerações devem ser feitas sobre o 'display file'. Se estruturas elaboradas são usadas para definição do desenho (por ex. 'pictures', figuras, gráficos, tabelas) obtem-se uma economia de memória e uma grande rapidez de interação; por outro lado, isto acarreta um aumento muito grande

na complexidade do processador do 'display file', o que leva a se procurar um ponto adequado de equilibrio entre a economia de memória, rapidez na interação e complexidade do processador.

Para determinação deste ponto de equilibrio deve-se voltar a atenção para aplicações onde se fez necessário o uso de transformações como: escalonamento, rotação, translação, 'clipping' e perspectiva e o modo como as mesmas são obtidas.

Na figura 2.2 o diagrama de blocos mostra os três processos esenciais para a geração da figura no CRT:

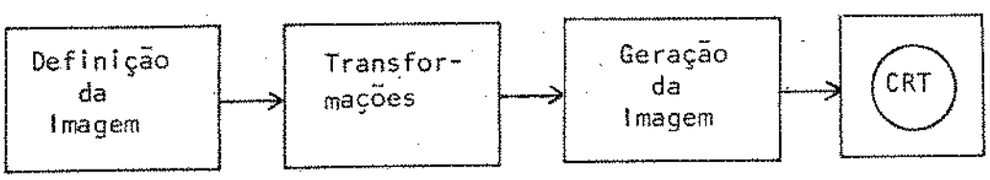


Figura 2.2 - Geração da imagem no CRT

- Definição da imagem : geralmente executada pelo programa de aplicação;
- Transformação: onde os algoritmos de transformação são aplicados e partes da imagem fora do 'display' são removidas por 'clipping';
- Geração da imagem: onde instruções para o CRT são fornecidas.

Ve-se de imediato que existem várias posições onde a memória auxiliar, usada para regeneração da imagem, pode ser colocada, das quais apenas duas são adequadas imediatamente antes ou imediatamente depois do bloco de transformação, (figura 2.3).

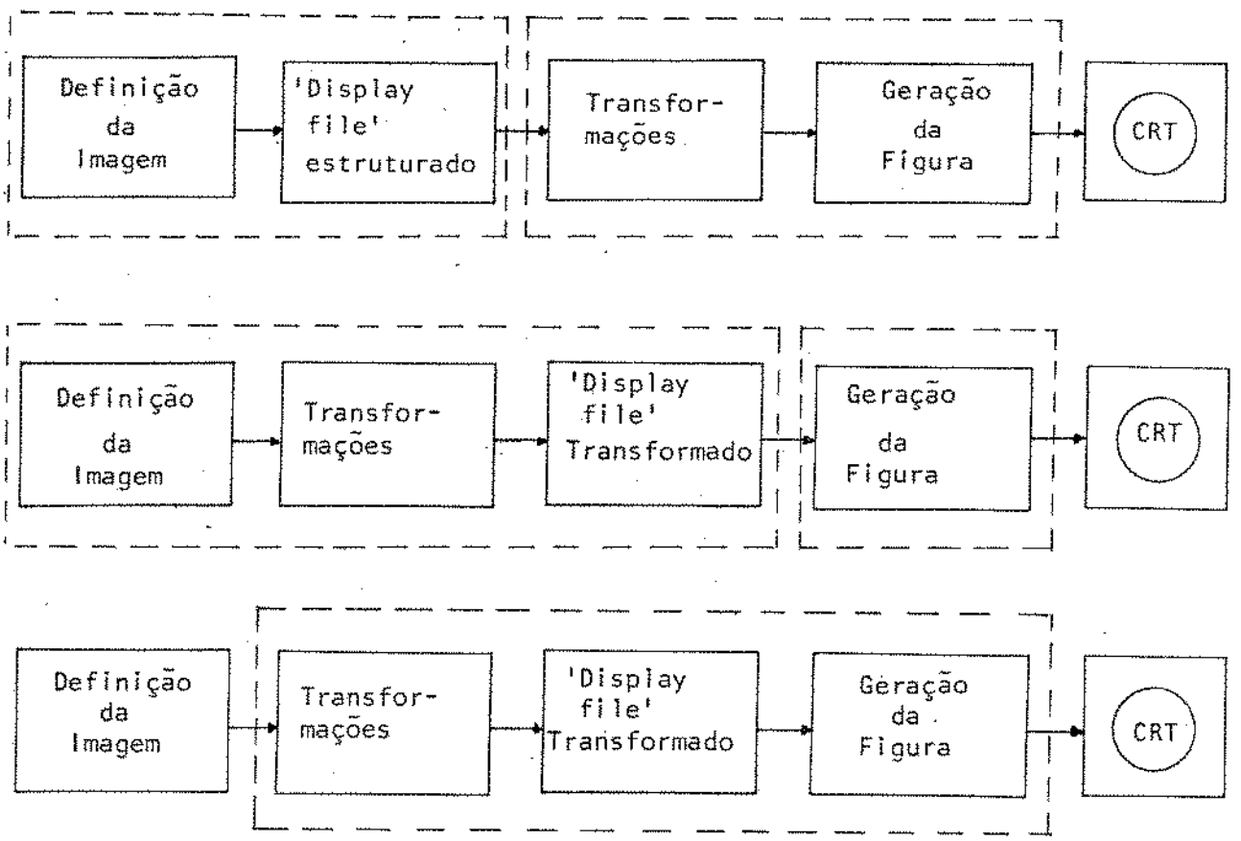


Figura 2.3 - Modos de geração da imagem

Deste modo para o processador do 'display file' somente três opções podem ser consideradas:

- Ele deve interpretar dados altamente estruturados e aplicar as transformações que devem estar embutidas no proprio processador;
- Interpretar os dados como já transformados e não estruturados;
- Executar as transformações e armazena-las na memória de regeneração de modo não estruturado.

Esta última opção apresenta-se como a solução mais viável, uma vez que a introdução de microprocessadores e a queda no custo das memórias permitirão a produção de sistemas altamente interativos a custos relativamente baixos.

Ao lado dos 'Line drawing displays' podem ser colocados os 'storage tube displays' . Estes 'displays' são mais baratos devido essencialmente a duas condições:

- A capacidade de armazenar informações por um período qualquer de tempo, no 'storage tube', o que dispensa a necessidade de regeneração da imagem e, portanto, de memória;
- O processador do 'display file' não exige alta velocidade de operação e pode ser bastante simplificado.

A principal desvantagem destes 'displays' se encontra na impossibilidade de tratar individualmente os ítens que formam uma figura. Qualquer modificação na figura requer a reconstrução de toda a imagem. Outra desvantagem reside na impossibilidade do uso da 'light-pen', porque a regeneração da imagem não implica no processamento do 'display file'. Apesar das diferentes características entre esta tecnologia e a anterior,a organização do 'display file' é práticamente a mesma.

Durante os últimos anos tornou-se possível a construção de'displays'gráficos usando monitores de TV.Uma das vantagens que ressalta imediatamente é o uso da tecnologia da televisão,altamente desenvolvida e relativamente barata.Além disso como a imagem é gerada por varredura sequencial, o aumento do número de elementos da figura não leva à ocorrencia de ' flicker'. Por outro lado o 'display file' para um 'display' deste tipo deve ser organizado como uma matriz de pontos ('frame buffer') ao invés de uma lista de segmentos. Isto requer grande quantidade de memória, onde cada segmento da figura é representado por pontos espalhados pelo 'frame buffer'. A velocidade de interação diminui por causa do processo de de computação dos pontos relativos a cada elemento e da dificuldade de modificação de segmentos individuais, dado o espalhamento, por todo o 'frame buffer' dos pontos que compõe que cada elemento da figura. Outro problema consiste na descontinuidade das linhas, provocada pelo quantização que é feita no processo de determinação dos pontos. Resumindo, os três principais problemas associados a esta tecno

logia são: o tamanho e custo do 'frame buffer', o processo de conversão de elementos para pontos e os efeitos da quantização.

Uma das questões relativas aos 'raster displays' refere-se à validade do uso do mapeamento da imagem em pontos ('frame buffer').Para economizar memória todos os outros modos de codificação da imagem requerem a construção de um processador capaz de decodificar a informação e passa-lo ao CRT. A solução mais usada é o'display'organizado por células, que consiste de um'display'alfanumérico, onde as figuras são obtidas com a ajuda de um conjunto especial de caracteres,contendo quase todos os possíveis segmentos que podem ser colocados dentro do espaço reservado a um caracter. O problema com a codificação deste tipo aparece quando elementos são adicionados ou retirados da figura. Se elementos são adicionados à figura, as células resultantes devem ser sobrepostas às existentes; se elementos são retirados, operações mais complexas são necessárias, ou a imagem deve ser completamente reconstruída prejudicando severamente o tempo de resposta do sistema.

O modo mais atrativo de codificação da imagem, para economizar memória, consiste num método idêntico ao usado nos 'Line drawing displays' para a definição dos segmentos que formam a figura. Além da vantagem relativa à economia de memória, esta representação permite a compatibilidade de sistemas, e o emprego do mesmo 'software' para os dois tipos de 'displays'. O problema nesta solução encontra-se na necessidade de um processador capaz de realizar o que se chama de conversão do 'display file' em tempo real, ou seja um dispositivo capaz de processar todo o 'display file' no tempo gasto para a varredura de uma linha do video, convertendo seu conteúdo em sinais apropriados ao CRT.

Nenhuma solução completa para o problema da conversão acima foi até agora apresentada na literatura.Algum progresso foi obtido com a proposição do uso de sistemas híbridos que associam o 'frame buffer' com o conversor. O 'frame buffer' armazena os dados gerados pelo conversor, eliminando desta forma as restrições impostas pela operação em tempo real, li

mitando com tudo a velocidade de resposta do 'display'.

2.5 - Objetivos

Do exposto fica claro que para aplicações interativas somente dois tispos de 'displays' são apropriados: os 'Line drawing' e os 'TV Raster'; isto é aqueles que permitem o uso da 'light-pen'.

Para aplicações como CAD, CAM, controle de processos os dois tipos de 'display' apresentam vantagens que devem ser consideradas:

- Os 'Line drawing' possibilitam a descrição pelo 'display file' e consequente tratamento individual de cada elemento da figura;
- Os 'TV Raster' possibilitam a sobreposição e digitalização de imagens e figuras; esta última técnica importante, por exemplo, em reconhecimento de padrões.

Como conclusão pode-se dizer que o 'display' ideal é aquele que associa a descrição da imagem pelo 'display file', com geração da imagem pelo sistema de varredura. Este método possui, porém, a restrição do conversor em tempo real que limita a velocidade de operação do sistema restringindo seu uso.

Um dos objetivos do autor é a definição de um sistema interativo como o descrito no parágrafo anterior no qual as restrições sejam práticamente eliminadas, pela adoção de técnicas especiais para o conversor. Como principal colaboração do trabalho, à área de desenvolvimento de 'displays', considera-se a conceituação e projeto de um sistema controlador da ' picture memory', baseado em um microcomputador (também utilizado para dotar o sistema de inteligência) e num conjunto de 'hardware'especilamente projetado operando de modo sequencial. Este controlador permite o acesso de dados da memória num tempo virtual reduzido, possibilitando satisfazer as condições impostas pela conversão em tempo real. Permite também a manipulação com alta efi

ciência e rapidez dos dados contidos no 'display file' resultando na quase que instantânea atualização da imagem e torna viáveis aplicações em tempo real, difícilmente realizáveis nos 'displays' em uso. Às características acima deve ser adicionada a capaciadede local de cálculo, dada pelo microprocessador, o que torna a maioria das soluções para aplicações já existentes fácilmente transladáveis para o sistema (práticamente independente do microprocessador usado) seja pela reedição do sistema operacional básico do terminal ou pela reedição de pequenas partes do programa de aplicações.

CAPÍTULO 3 - UM TERMINAL GRÁFICO BASEADO NA TECNOLOGIA DO 'TV RASTER'

Neste capítulo do trabalho será descrito funcionalmente o terminal mencionado no ítem 2.5., cuja estrutura básica é apresentada, simplificadamente na figura 3.1. Seus componentes básicos são o microprocessador, a 'picture memory', o controlador da 'picture memory' e os circuitos de conversão e acumulação dos pontos ativos dos elementos do 'display file'. Esses componentes de acordo com suas funções, estarão associados à execucão de funções rápidas e lentas; as funções rápidas estão baseadas em circuitos de alta velocidade, cujos ciclos são da ordem de 60 nanosegundos e as funções lentas estão mais relacionadas com os circuitos de interfaceamento da'picture memory' com os periféricos, cujos ciclos são da ordem de microsegundos (relógio do microprocessador)

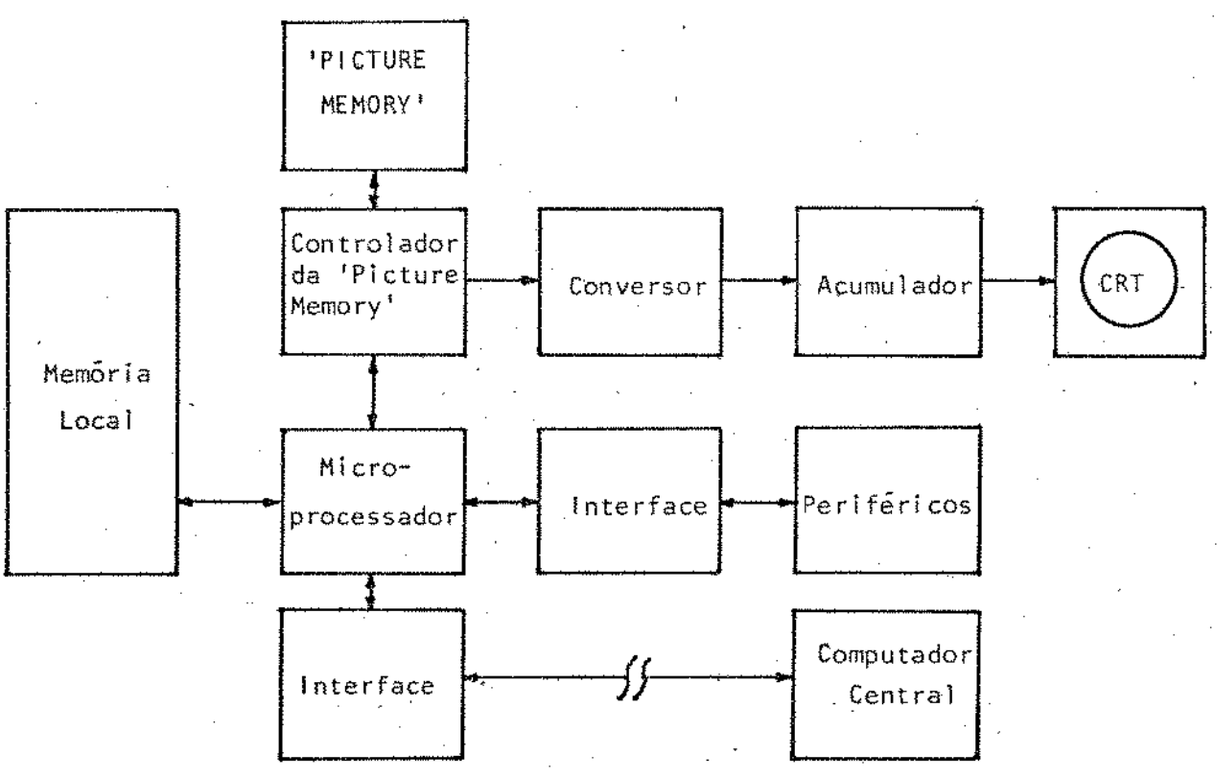


Figura 3.1 - Diagrama de blocos do terminal gráfico proposto

As funções lentas executadas básicamente num micro processador, referem-se a:

. Comunicação com o computador central;

. Comunicação com o usuário;

. Controle dos periféricos locais (exceto o CRT);

. Operações na 'picture memory';

. Execução de programas locais.

O microprocessador terá acesso também às memórias locais do tipo RAM onde estão armazenados:

. Programas de aplicação, carregados a partir do computador central ou periféricos locais;

. Descrição da figura ('display file'), de modo estruturado ou não, cuja finalidade é a execução de transformações localmente;

e do tipo ROM que conterão:

. Sistema operacional básico, cuja finalidade é a inicialização do sistema, controle dos periféricos locais e comunicação com o computador central;

. Programas de acesso à 'picture memory', que permitirão a transferência de dados e atualização da 'picture memory'.

Por funções rápidas comprende-se o acesso ao CRT que é feito através dos circuitos de alta velocidade constituidos básicamente por:

. Conversor do 'display file' em tempo real;

. Acumulador;

. Controlador da 'picture memory'.

Na figura 3.2. são apresentados os principais blocos dos circuitos de alta velocidade.

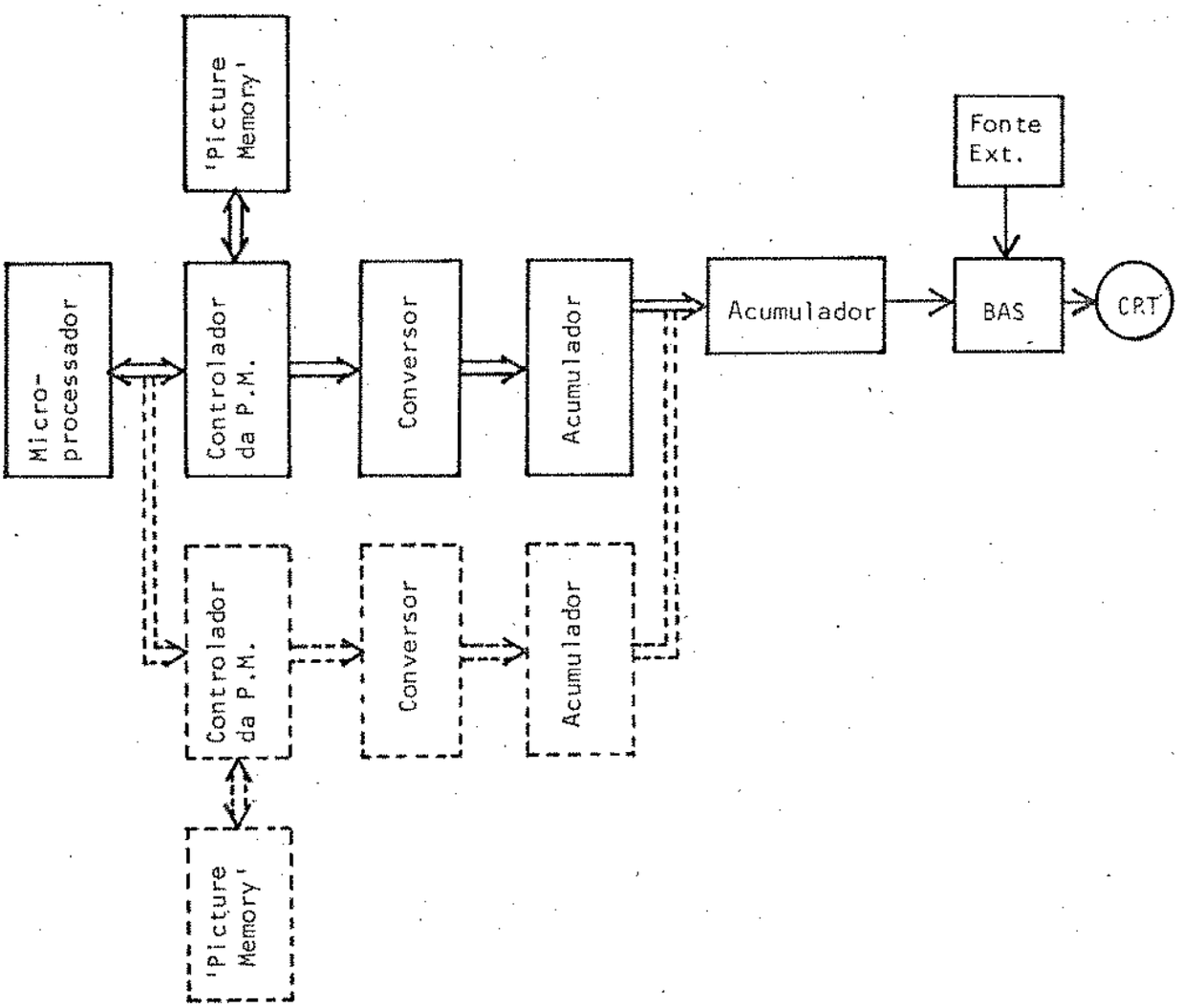


Figura 3.2 - Extensão da versão mínima

O controlador da 'picture memory' acessa dados na 'picture memory' e transfere-os ao conversor de tempo real, a medida que são requisitados pelo mesmo. Uma vez processados os dados determinam os pontos da linha que devem ser ativados para representação do elemento computado, sendo tais pontos, então, armazenados num acumulador, formado por registros de largura igual ao número de pontos das linhas da tela. Ao fim do processamento de todo o conteúdo da 'picture memory' o conteúdo deste acumulador é transferido a um outro acumulador de onde é sequencialmente enviado aos circuitos de geração dos sinais analógicos usados para controle do CRT. O processo de conversão, é então, reiniciado para a geração dos pontos ativos da próxima linha a ser varrida na tela. Este processo deve ser executado, no máximo, em aproximada

mente 64 microsegundos, que é o tempo necessário para a varredura de uma linha nos padrões de televisão.

O diagrama da figura 3.2. corresponde a versão mínima do sistema. O mesmo pode ser estendido através da inclusão de novas unidades ('picture memory' + controlador + conversor) que atuarão em paralelo, permitindo a descrição de figuras com maior grau de complexidade. A inclusão, em paralelo, de outras unidades de acumulação permite o uso de diferentes intensidades em preto e branco, o uso de cores e um maior grau de resolução.

Para facilitar a descrição que se segue o 'hardware' do sistema será dividido em três partes:

- Conversor em tempo real do 'display file' e acumulador dos pontos ativos;
- 'Picture memory' e o seu controlador;
- O microprocessador e sua interface com o controlador da 'picture memory'.

3.1 - Conversor do 'display file' e acumulador

O diagrama de blocos da figura 3.3 mostra os componentes principais do sistema de conversão e acumulação. Do controlador da 'picture memory' o conversor recebe informações sobre a figura a ser gerada, descrita por um conjunto básico de instruções e coordenadas devidamente armazenadas na memória no seguinte formato:

1 000 *** *** **w RGB	vetor
1 001 *** *** *** ***	ponto
1 010 *** *** *** ***	área
1 011 *** *** *** ***	caracter
1 111 *** *** *** ***	fim do arquivo ('end of file')
0 *** *** *** *** ***	coordenada ou caracter

onde os bits R, G e B representam as cores básicas ou intensidade em preto e branco e o bit W a apresentação intermitente ou

não do elemento ('blink'). Para as coordenadas usam-se 15 bits, permitindo-se a descrição de figuras numa área de 32k x 32k pontos. Como a figura gerada na tela contém apenas 512 x 384 pontos verifica-se que o 'display' atua como um 'viewport', sendo necessário, portanto, a definição da coordenada relativa ao canto esquerdo inferior do'display'.

A medida que a informação é recebida, a mesma é decodificada e transferida aos registros internos. Ao ser recebida uma instrução, identificada pelo bit mais a esquerda,ela é tranferida ao registro de instrução e então decodificada. De acordo com a instrução, diferentes números de parâmetros são necessários como por exemplo, duas coordenadas para pontos; quatro coordenadas para vetores (para o primeiro elemento) e dois para os subsequentes; coordenadas e sequência de caracteres. Para isso um decodificador-controlador atúa sobre os registros de coordenadas carregando-os convenientemente e inibindo o controlador da 'picture memory' até que os dados carregados tenham sido processados. Os registros 'viewport' e 'linha' devem conter as coordenadas do canto esquerdo inferior do 'display' e o índice da linha que está sendo processada, respectivamente, sendo o primeiro atualizado através do microprocessador, e o segundo pelo controlador cada vez que termina a varredura de uma linha.

Uma vez carregados os registros acima, inicia-se o pré-processamento do elemento com a finalidade de se otimizar o uso da unidade aritmética. Como a área para descrição da figura (área virtual) é muito maior que a área visível, muitos elementos contidos na descrição, seguramente, não possuem intersecção com a linha que está sendo processada, ou a mesma não é visível. Definindo-se:

IWL : a distância entre duas linhas na tela;

IYL : a ordenada da linha que está sendo processada;

ILB e IRB : os limites a esquerda e a direita do 'viewport' (limites do quadro visível)

qualquer vetor descrito pelos pares de coordenadas (X_1, Y_1) e (X_2, Y_2) que satisfaçam qualquer par das desigualdades

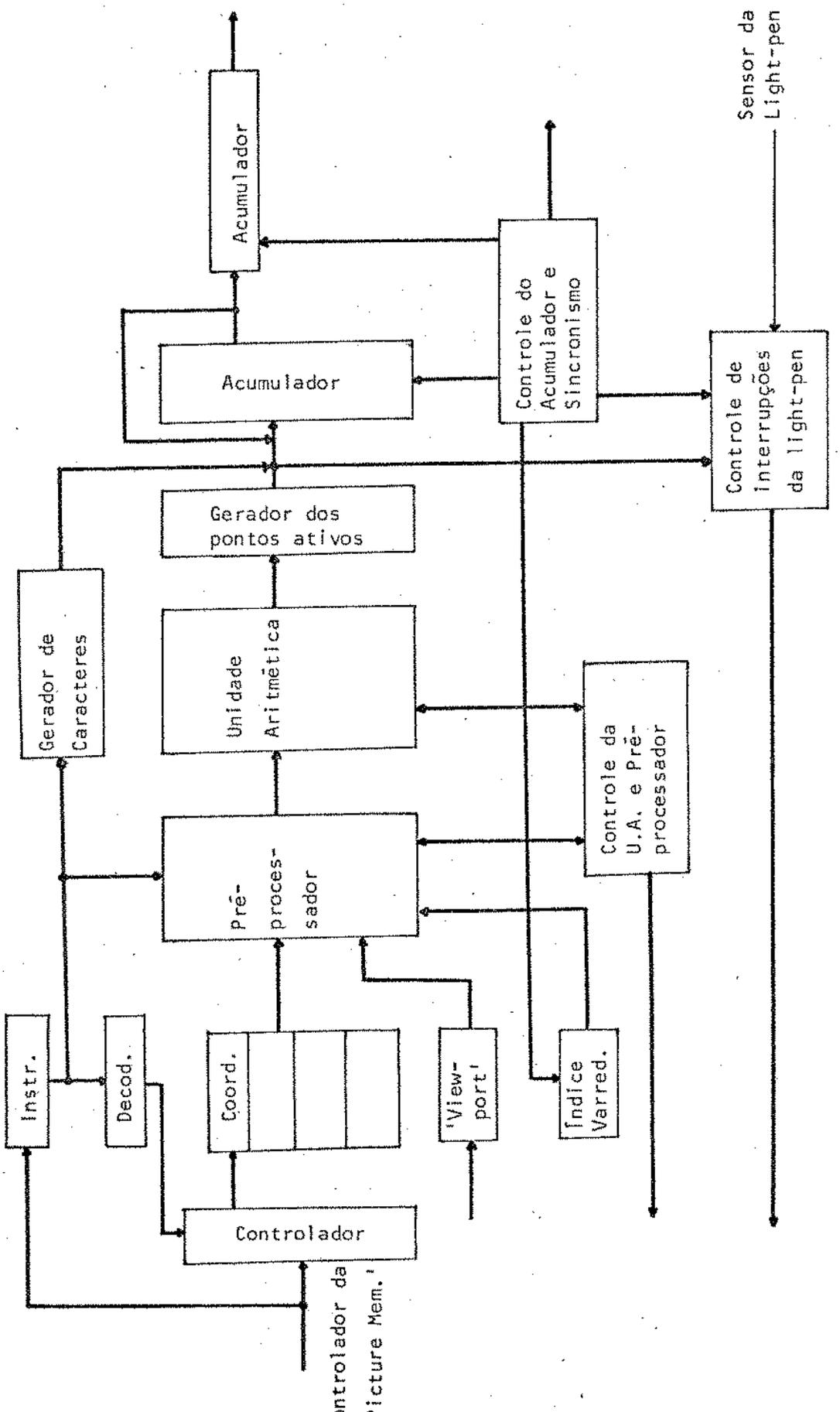


Figura 3.3 - Diagrama de blocos dos circuitos rápidos

$Y_2 < IYL - 1/2\ ILW$ e $Y_1 < IYL - 1/2\ ILW$;

$Y_2 > IYL + 1/2\ ILW$ e $Y_1 > IYL + 1/2\ ILW$;

$X_2 < ILB$ e $X_1 < ILB$;

$X_2 > IRB$ e $X_2 > IRB$;

e os pontos descritos pelas coordenas $(X_1\ Y_1)$ que satisfaçam qualquer das desigualdades

$Y_1 < IYL - 1/2\ ILW$;

$Y_1 > IYL + 1/2\ ILW$;

$X_1 < ILB$;

$X_1 > IRB$;

não precisam ser processados pela unidade aritmética, uma vez que não possuem qualquer ponto ativo sobre a linha que está sendo calculada.

O circuito para detecção deste elementos não efetivos é implementado a partir de comparadores rápidos, que permitem o processamento de um número relativamente grande de elementos durante um ciclo da unidade aritmética garantindo o contínuo e efetivo funcionamento desta última. Para cada elemento rejeitado, um novo é solicitado ao controlador da 'picture memory' até que um elemento efetivo seja encontrado. Nestas condições o processamento é interrompido e somente reiniciado, quando a unidade aritmética estiver livre para o recebimento e processamento das coordenadas deste elemento.

A unidade aritmética é a responsável pela determinação dos pontos ativos de cada elemento, através do cálculo dos limites esquerdo e direito desses pontos. Seu funcionamento está baseado na equação:

$$X_s = X_1 + \frac{(X_2 - X_1)}{(Y_2 - Y_1)} (Y_s - Y_1) \qquad (3.1)$$

que define o valor da intersecção no eixo X de um vector definido pelos pares de coordenadas (X_1, Y_1) e (X_2, Y_2) com a li

nha horizontal Y_s.

A primeira vista, se o caso $Y_2 = Y_1$ não for considerado, nenhuma dificuldade é encontrada sob o ponto de vista matemático. Porém sob o aspecto de implementação em 'hardware' e da conversão em pontos para a varredura surgem alguns pontos que devem ser considerados:

- A distância entre os pontos nos eixos X e Y são diferentes, devido ao uso dos padrões de televisão. Este fato torna o valor Y_s não inteiro forçando o uso de mecanismos para operação em ponto flutuante;
- A conexão entre os diversos segmentos nos quais é dividido um vetor para geração da imagem pelo processo de varredura. Este fato não é evidente da equação acima, mas está diretamente relacionada com a precisão dos cálculos;
- A conexão de segmentos de vetores distintos que possuem coordenadas comuns.

Um algoritmo para determinação dos limites dos pontos ativos de um vetor é proposto e discutido em |10| e um resumo é apresentado aqui.

A determinação dos limites esquerdo (IXL) e direito (IXR) é obtida através de duas extensões da equação 3.1, considerando-se a constante (ILW) que relaciona a distância entre duas linhas da tela.

$$IXL = X_1 + \frac{(X_2 - X_1)\,((Y_s + 0.5ILW) - Y_1)}{(Y_2 - Y_1)} \qquad (3.2)$$

$$IXR = X_1 + \frac{(X_2 - X_1)\,((Y_s - 0.5ILW) - Y_1)}{(Y_2 - Y_1)}$$

O algoritmo para solução das equações acima é implementado em 'hardware', e tem como entrada as coordenadas do vetor, o índice da linha que está sendo calculada e as coordenadas

referentes ao 'viewport'. A saída é composta pelos limites esquerdo e direito do segmento do vetor computado e por um indicador de validade ('flag') dos resultados obtidos. A estrutura do algoritmo é apresentada na figura 3.4.

O algoritmo inicia-se no canto superior esquerdo e para facilidade de implementação das multiplicações e divisões alguns sinais são tratados separadamente e as operações são realizadas em valores absolutos. É utilizada a representação em complemento de dois.

A primeira ação é o cálculo do valor de Δ_y em valor absoluto (IDY) e o sinal dessa diferença (SDY). O segundo passo é a divisão da distância entre duas linhas (ILW) pelo valor de Δ_y (IDY), que é realizada através da determinação do inverso de Δ_y e posterior multiplicação por ILW. Este procedimento é usado para contornar as dificuldades advindas da divisão, dado que a determinação do inverso de Δ_y pode ser fácilmente obtida pelo uso de uma memória PROM, tendo em sua entrada o valor de Δ_y e gerando na saída o valor inverso normalizado, representado por uma mantissa (IRY) e um expoente (ERY). Este último será usado indiretamente para ajustar a mantissa de outros números de modo a facilitar as operações subsequentes.

Em paralelo ao cálculo de Δ_y é realizado também o cálculo de Δ_x (IDX) e determinado o respectivo sinal desta diferença (SDX). Como resultado da multiplicação de IRY e IDX é obtido IQU que é o quociente Δ_X/Δ_Y . Para a realização desta multiplicação é utilizado um multiplicador implementado em 'hardware'. O resultado da multiplicação possui o dobro de bits dos números originais e a perda de precisão resultante é contornada pelo fato de um dos fatores se encontrar normalizado.

Simultâneamente com os cálculos descritos é realizado o cálculo de $Y_s - Y_1$ (ver equação 3.1). Para isso é necessária a consideração dos fatores relativos ao 'viewport' , ou seja a ordenada da linha inferior da tela (IYO), gerando-se, assim uma ordenada relativa à tela (IYF) obtida da subtração (IY1 - IYo). Obtido IYF a próxima ação é a divisão deste valor pela distância entre duas linhas (ILW) para obtenção do índice

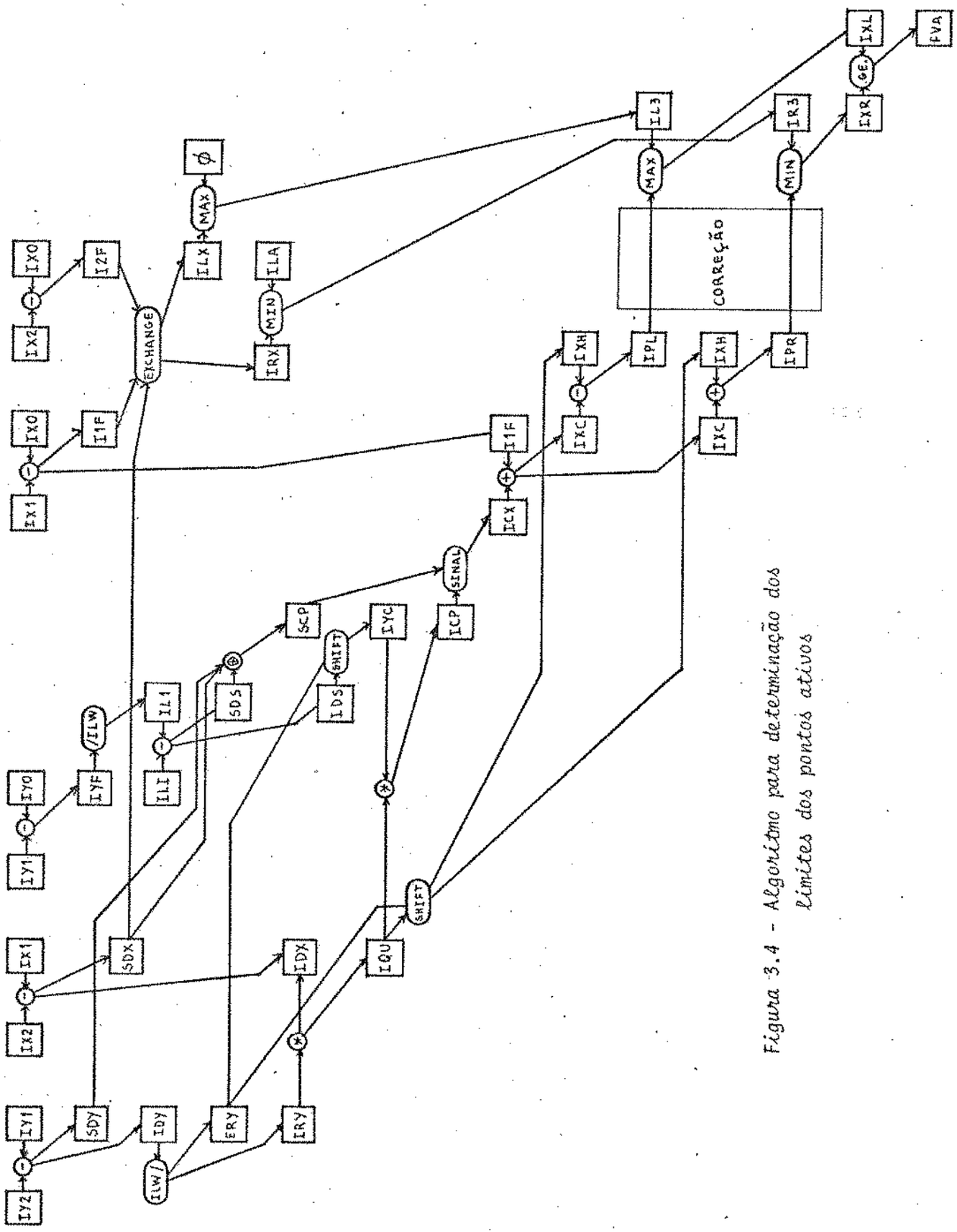


Figura 3.4 - Algoritmo para determinação dos limites dos pontos ativos

da linha (IL1) da coordenada Y do ponto 1. A seguir é feito o cálculo da diferença entre o índice da linha que esta sendo processada (ILI) e o valor do índice (IL1). O resultado desta operação será representado por (IDS) e (SDS), que contêm o valor absoluto da diferença e o sinal do resultado,respectivamente.

Neste ponto o quociente Δ_X/Δ_Y não está ainda determinado. Portanto, sem a introdução de atrasos, e baseando-se no valor de ERY, pode ser realizado o deslocamento de IDS ajustando-se as mantissas para que o produto de IDS por IQU tenha como resultado expoente zero. Neste procedimento não existe perda de precisão, pois o deslocamento é feito para a esquerda e não há ocorrência de 'overflow', porque números grandes requerem pequenos deslocamentos. Para valores grandes de IDS correspondem pequenos valores de Δ_y e consequentemente valores grandes para ERY. Após o deslocamento de IDS para obtenção de IYC é feito o produto de IYC por IQU dando como resultado ICP (palavra dupla).

O próximo passo é a representação em complemento de dois (ICX) do valor absoluto de ICP, para que as adições possam ser realizadas. O sinal é obtido pela operação lógica 'ou' exclusivo dos três sinais SDY, SDX e SDS. A complementação da mantissa, se necessária, será feita pela mesma unidade aritmética e lógica (ALU) que será usada para a realização das outras operações.

A seguir efetúa-se a correção da coordenada X, de modo semelhante à de Y, pela subtração da abcissa (IXO)do lado esquerdo da tela, e obtêm-se a posição relativa (IlF) da abcissa X do ponto 1. O último passo para a solução de 3.1 é a adição de ICX e IlF que resulta no valor X_s, centro do segmento de pontos ativos (IXC).

Para a solução das equações 3.2 resta a consideração do termo (Δ_x/Δ_y) * 0.5 IWL. Este termo já disponível em IQU deve apenas ser deslocado para a posição correta com o auxilio de ERY, obtendo-se (IXH) que somado e subtraido a IXC determina os limites a esquerda (IPL) e a direita (IPR), respectivamente, do segmento de pontos ativos.

Para completar o algoritmo os valores obtidos devem ser comparados com os limites do quadro e com as coordenadas originais dos pontos. Se o segmento obtido excede estes limites, ele deve ser corrigido cortando-se a parte excedente. Se o segmento se encontra fora do quadro a saída deve ser declarada como invalida. Para que essa correção ocorra faz-se necessária a determinação da abcissa relativa (I2F) do ponto 2, e a ordenação dos valores relativos I1F e I2F. Desta última resultam os valores a esquerda (ILX) e a direita (IRX) que são comparados aos limites do quadro: o valor da abcissa do lado direito do quadro (ILA) para o limite a direita e zero para para o limite a esquerda. Tomam-se o valor máximo (IL3) entre ILX e zero, e o valor minimo (IR3) entre IRX e ILA. Obtem-se finalmente os limites a direita (IXR) e a esquerda (IXL) tomando-se o valor máximo entre IPR e IR3, respectivamente.

Considerando-se as caracteristicas dos elementos disponíveis atualmente, o tempo para o processamento de um vetor pode ser calculado através do caminho crítico, chegando-se a um valor em torno de 930 nanosegundos. Este valor é perfeitamente aceitável, pois é inferior ao tempo necessário para a execução de um ciclo de acumulação (explicado a seguir) não se criando, portanto, qualquer estrangulamento no sistema.

O processo de acumulação dos pontos efetivos da linha deve ser modular e o mais simples possível para tornar assim o processo independente do número de pontos de cada linha na tela.

A solução para a acumulação é esquematizada na figura 3.5. Ela está baseada em módulos de 64 bits cada,que possuem como entrada os valores dos limites a esquerda (IL3) e a direita (IR3) dos pontos ativos e um sinal de relógio com período de 45 nanosegundos. Este sinal é também usado para controlar a transferência de sinais do acumulador para o módulo BAS. O funcionamento do processo de acumulação é descrito a seguir.

O sinal do relógio é dividido por quatro e

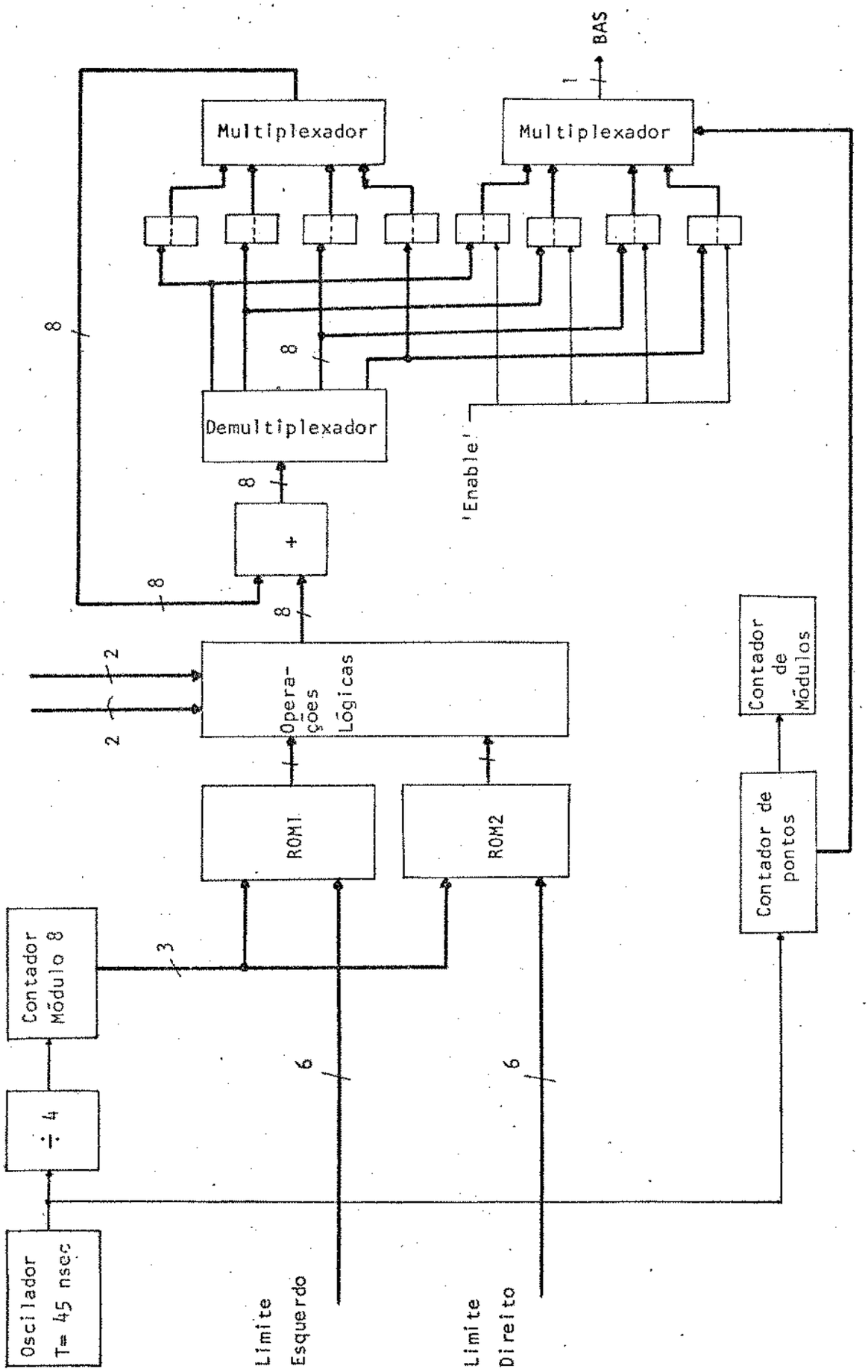


Figura 3.5 - Acumulador dos pontos ativos

a onda obtida (T= 180nsec) é usada como entrada num contador módulo 8, cujas saídas fornecem sequencialmente os endereços de cada um dos sub-módulos (8 bits) em que está dividido o módulo. O endereço do sub-módulo juntamente com os valores de IL3 e IR3 são usados para endereçamento das ROM1 e ROM2, respectivamente. Na saída destas memórias, duas sequências de oito bits cada uma são geradas e sua multiplicação lógica dá como resultado os pontos ativos correspondentes a IL3 e IR3. Por exemplo, se IL3= 36 e IR3 = 53 a seguinte sequência de valores é obtida na saída das ROMs.

ENDEREÇO SUBMÓDULO	ROM1	ROM2	ROM1.ROM2
000	00000000	11111111	0000 0000
001	00000000	11111111	0000 0000
010	00000000	11111111	0000 0000
011	00000000	11111111	0000 0000
100	00001111	11111111	0000 1111
101	11111111	11111111	1111 1111
110	11111111	11111100	1111 1100
111	11111111	00000000	0000 0000

ou seja, a ROM1 tem como saída valor 'zero' nos bits correspondentes às posições anteriores a IL3 e valor 'um' nos demais e a ROM2 valor 'um' nos bits correspondentes às posições anteriores a IR3 e valor 'zero' nos demais.

Para a realização de um ciclo completo de acumulação é necessário a execução de oito subciclos, consumindo-se um tempo igual a 45 x 4 x 8 = 1440 nanosegundos. Em cada subciclo efetua-se a soma lógica (aritmética no caso em que diferentes intensidades são usadas) do conteúdo do submódulo ao resultado gerado pelo produto lógico da saída das ROMs, e o rearmazenamento dos valores resultantes.

O processo é repetido para cada par de valores gerados pela unidade aritmética, durante o tempo gasto para a varredura e o retraço correspondentes a linha anterior. Ao final deste período o resultado, ao invés de ser rearmazenado no

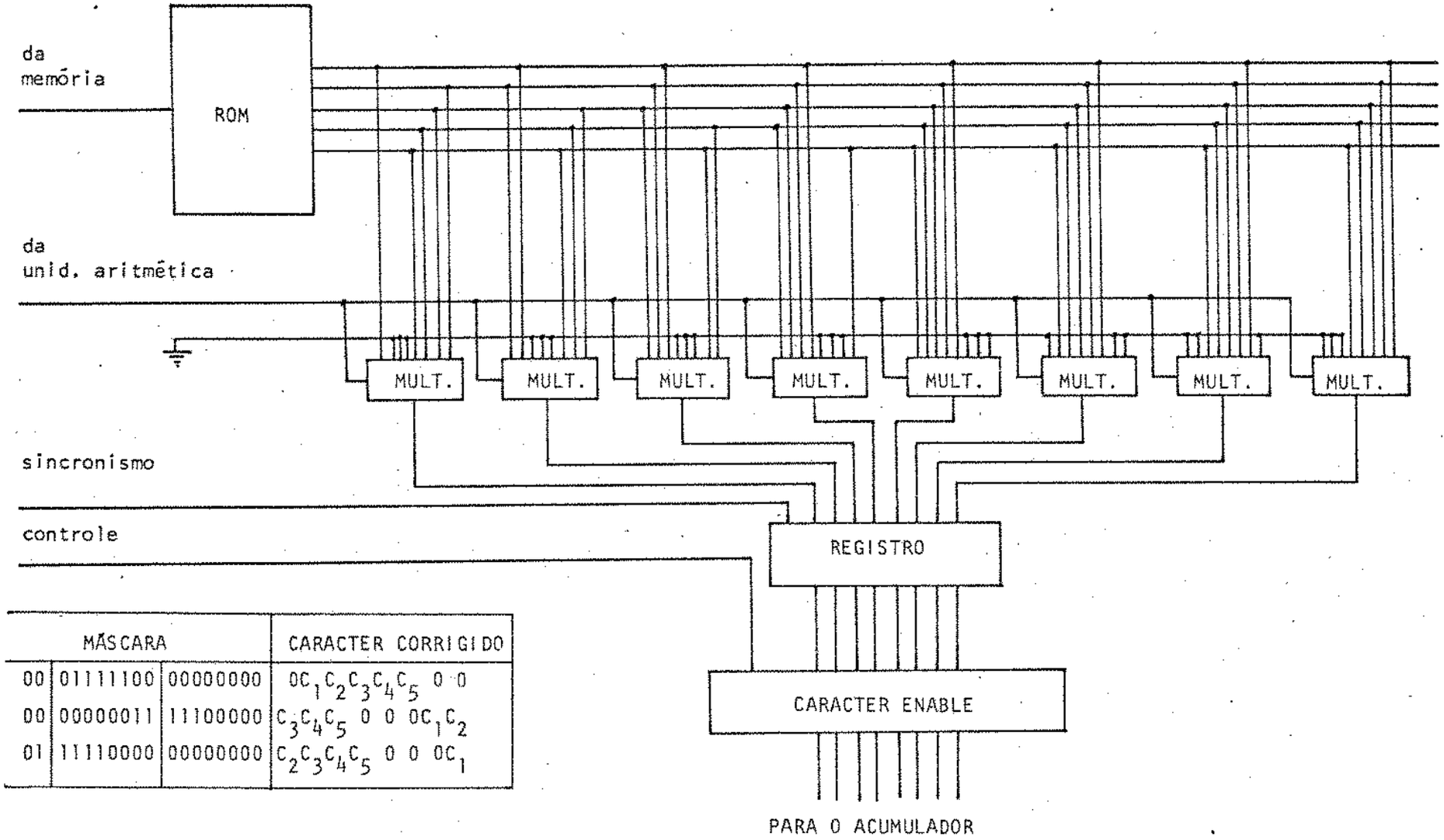
acumulador, é transferido para os registros de varredura, que possuem arquitetura semelhante à do acumulador; ao mesmo tempo o acumulador é reinicializado. Finalizada a transferência,o conteúdo dos registros de varredura passa a ser transmitido sequencialmente ao CRT (através do BAS), sob o controle do multiplexador 3 (ver figura 3.5).

Para que seja possível a associação de vários módulos de acumulação, na formação de linhas do comprimento desejado, quatro outras entradas são usadas, com o bloco de operações lógicas utilizando a informação disponível nestas linhas. A finalidade destas linhas é localizar a posição de cada módulo em relação aos valores limites do segmento de pontos ativos e determinar o seu funcionamento correto, Isto é, somente os módulos correspondentes a IL3 e IR3 devem atuar de modo descrito, enquanto os demais blocos geram sequências de valores 'um' ou 'zero' de acordo com sua posição relativa ao segmento.

Para completar a descrição desta parte do circuito discute-se a seguir o modo de geração de caracteres e o controle de interrupção por 'light-pen'.

A geração de caracteres é obtida pelo uso simultâneo do conversor do 'display file' e de uma ROM endereçada pelos código e linha do caracter. As coordenadas dos caracteres são processadas pelo conversor do 'display file', como se fossem vetores horizontais de extensão equivalente a dois caracteres, gerando uma sequência de pontos ativos que serão usados para o mascaramento dos sinais gerados pela ROM. Isto é, os sinais provenientes da ROM são multiplicados lógicamente pelos pontos ativos e o resultado é armazenado no acumulador.

Como a acumulação dos pontos ativos é realizada através dos submódulos, existe a necessidade de ajuste da saída da ROM, de modo a ter-se um ajuste da máscara com a posição do caracter. Isto é obtido através do deslocamento circular da saída da ROM, de um número de posições determinada pelos três últimos bits do limite esquerdo gerado pelo conversor do 'display file'. A figura 3.6 ilustra o funcionamento desta parte do circuito e mostra alguns exemplos.



MÁSCARA			CARACTER CORRIGIDO
00	01111100	00000000	$0C_1C_2C_3C_4C_5\ 0\ 0$
00	00000011	11100000	$C_3C_4C_5\ 0\ 0\ 0C_1C_2$
01	11110000	00000000	$C_2C_3C_4C_5\ 0\ 0\ 0C_1$

Figura 3.6 - Gerador de caracteres

O circuito detector de interrupções da 'light pen' possui dois registros onde são armazenados os valores correspondentes ao índice da linha, e a posição do ponto dentro da qual ocorreu a interrupção. Estes valores são obtidos pela transferência, ativada pelo sensor da 'light-pen', do conteúdo dos registros-contadores do controlador dos registros de varredura.

Durante a geracão do quadro seguinte, circuitos comparadores são ativados e um sinal é transmitido ao controlador da 'picture memory', quando o ponto ativo correspondente ao ponto de interrupção é gerado no processo de acumulação. Por intermedio desse sinal os registros de endereço da 'picture memory' são transferidos ao microprocessador (como é descrito na secção 3.3) permitindo assim a identificação do elemento selecionado na figura.

3.2 - 'Picture memory' e o seu controlador

Para a geração de cada linha é necessário que se execute a conversão de todo o conteúdo do 'display file', o que implica na leitura completa do 'display file' para cada linha gerada. Se considerarmos que uma quantidade razoável de memória se faz necessária para a descrição de figuras mais complexas, conclui-se de imediato que o tempo de acesso para leitura ou escrita de cada palavra da memória é inferior ao tempo de acesso em memórias disponíveis comercialmente. Ou seja, o acesso ao contéudo da 'picture memory' é um ponto crítico que exige o uso de estruturas especiais para ser contornado.

Além disso, a atualização do conteúdo dessa memória também exige tempo, o que torna ainda mais crítico o problema enunciado acima. Se não realizada de modo eficiente, a atualização da imagem torna-se lenta e a qualidade do terminal deteriora-se, diminuindo sensivelmente a faixa de aplicações possíveis (por exemplo: aplicações em tempo real).

A solução aquí apresentada para o problema satisfaz as condições desejadas: compatibilidade do tempo de acesso e eficiência na atualização. Além disso, constitui-se uma so

lução modular, permitindo facilmente a extensão da 'picture memory' e possibilitando a geração de imagens mais complexas. Sob o ponto de vista de implementação, também apresenta vantagens pelo uso da moderna tecnologia dos microprocessadores, que facilita a construção, o uso e a manutenção do conjunto.

Para facilidade de projeto e sem grandes perdas para a taxa de regeneração da imagem foram definidos módulos de memória de 1.5 K palavras de 16 bits, o que, por sí só, torna possível um grau razoável de complexidade da figura. Num módulo podem ser armazenados 300 vetores posicionados aleatoriamente ou, 760 vetores em um polígono ou, mesmo 2800 caracteres em sequência representando 35 linhas de 80 caracteres cada. O tempo de acesso de 60 nanosegundos embora um pouco superior ao desejado é aceitável. A redução do tamanho do módulo para 1.2 K palavras permitiria a leitura completa da 'picture memory' no tempo desejado, porém acarretaria uma maior complexidade dos circuitos de controle dada a natureza dos endereços a serem tratados. O 'display file' deve ser armazenado na memória já transformado e não são permitidas descrições estruturadas pois a leitura da memória é sequencial. A atualização e a leitura do conteúdo da memória são executadas simultâneamente o que garante o alto desempenho do conjunto. Uma descrição dos circuitos e de seu funcionamento é apresentada nas seções seguintes.

3.2.1 - Descrição do funcionamento da 'picture memory'

Com o objetivo de melhorar a eficiência da atualização do conteúdo da 'picture memory' e, portanto, da imagem, uma nova arquitetura, totalmente transparente ao usuário, é proposta para a implementação da 'picture memory'. Em cada módulo são usadas 2K palavras de memória ao invés de 1.5K palavras. A memória é dividida em quatro blocos, utilizando-se apenas três desses blocos(1.5K palavras)para a descrição da figura. O quarto bloco, através da relocação dinâmica do conteúdo da 'picture memory', torna possível a realização das correções ao mesmo tempo emque a 'picture memory' é lida para a regeneração da imagem.

O esquema de relocação dinâmica é facilmente compreendido com o auxílio da figura 3.7. Os quatro blocos da memória são numerados 0, 1, 2 e 3, que correspondem aos dois bits mais significativos dos endereços em cada bloco (00, 01,10 e 11). Toda palavra lida na memória é reescrita 3 blocos adiante na mesma posição relativa dentro do bloco. Por exemplo, se uma palavra é lida na posição 100 do bloco zero, ela é reescrita na posição 100 do bloco 3; se ela é lida no bloco um é reescrita no bloco zero e assim por diante.

Este esquema póde ser implementado a partir do uso de dois contadores de endereço, um para leitura e outro para escrita, carregados convenientemente no início de cada varredura da 'picture memory'. Assim a abertura de espaços para inserção de novos dados, ou a eliminação de palavras da 'picture memory' é facilmente realizada somando-se ou subtraindo-se do contador de escrita o número de posições desejadas, quando o mesmo atinge a posição em que se quer efetuar a correção.

A figura 3.7 mostra exemplos de correções na posição 10 da memória. Vê-se de imediato que o endereço ocupado na memória por um elemento não corresponde ao seu endereço efetivo no 'display file', uma vez que a posição inicial é constantemente mudada. Assim, ao ser atingida a posição relativa correspondente ao endereço 10 do 'display file', no caso 00010, o contador dos endereços de escrita é corrigido para 00100, no caso de abertura de espaço para duas palavras. O exemplo mostra também a eliminação das palavras 10 e 11, obtida pela correção do contador de escrita de 00100 para 00010. A vantagem da utilização deste processo, em relação aos processos convencionais, reside no fato de que após uma leitura completa da memória qualquer número de espaços pode ser aberto, ganhando-se, portanto, em eficiência.

Como o tempo de acesso das memórias mais rápidas está em torno de 50 nanosegundos, muito próximo do valor do tempo disponível para a leitura de cada palavra, a operação só é possível com o uso de circuitos operando em 'pipe line',descritos em |11| e apresentados aqui resumidamente.

			OPERAÇÃO NORMAL	INSERIR PALAVRAS	APAGAR PALAVRAS
	00...000	A	I	I	I
	00...001	B	J	J	J
	...010	C	K		M
0	011	D	L		N
	100	E	M	K	O
	101	F	N	L	P
	110	G	O	M	Q
	111	H	P	N	R
	00...000	I	Q	O	S
	001	J	R	P	END
	010	K	S	Q	
1	011	L	END	R	
	100	M		S	
	101	N		END	
	110	O			
	111	P			
	10...000	Q			
	001	R			
	010	S			
2	011	END			
	100				
	101				
	110				
	111				
	11...000		A	A	A
	001		B	B	B
	010		C	C	C
3	011		D	D	D
	100		E	E	E
	101		F	F	F
	110		G	G	G
	111		H	H	H
SEQUÊNCIA PCREAD			00...000 ⋮ 10...011	00...000 ⋮ 10...011	00...000 ⋮ 10...011
SEQUÊNCIA PCWRITE			11...000 ⋮ 01...011	11...000 ⋮ 00...001 00...100 ⋮ 01...101	11...000 ⋮ 00...011 00...010 ⋮ 01...001

Figura 3.7 - Relocação dinâmica da 'picture memory'

A figura 3.8 apresenta um diagrama geral do controlador e da 'picture memory'. Para interfaceamento do microprocessador com a 'picture memory' são necessários cinco registros denominados IR1, IR2, IR3, IR4 e IR5 e com as seguintes funções e características:

IR1 : tem o comprimento das palavras da 'picture memory' e armazena temporariamente dados que são transferidos do microprocessador para a 'picture memory'.

IR2 : tem o comprimento de quatro bits e atua como um registro de controle armazenando o 'status' do controlador e o código da função a ser executada sobre a 'picture memory'. A função de cada um dos seus bits é a seguinte:

CW4, CW3 - código da função a ser executada:

00 - abrir e preencher espaço
01 - apagar uma ou várias palavras
10 - escrever uma única palavra
11 - transferir uma palavra da 'picture memory' para o microprocessador

CW2 - dado pronto: este bit é colocado em 'um' toda vez que uma palavra é transferida do microprocessador para o registro IR1 e retorna a zero quando o conteúdo de IR1 é transferido para a 'picture memory'.

CW1 - instrução pronta: este bit é colocado em 'um' pelo microprocessador e indica que os registros IR3, IR4 e IR5 foram devidamente carregados, isto é o controlador pode iniciar a execução da instrução selecionada. Este bit permanece na condição verdadeira até o final da execução da instrução quando é colocado em zero pelo controlador da 'picture memory'.

O registro IR2, portanto, contém o 'status' do sistema e deve ser lido pelo microprocessador antes da transferência de novos dados ou da execução de novas instruções.

IR3 : Este registro tem 11 bits de comprimento ou seja, capacidade para o endereçamento de 2K palavras e, quando as instruções (00) ou (01) são selecionadas, é carregado pelo

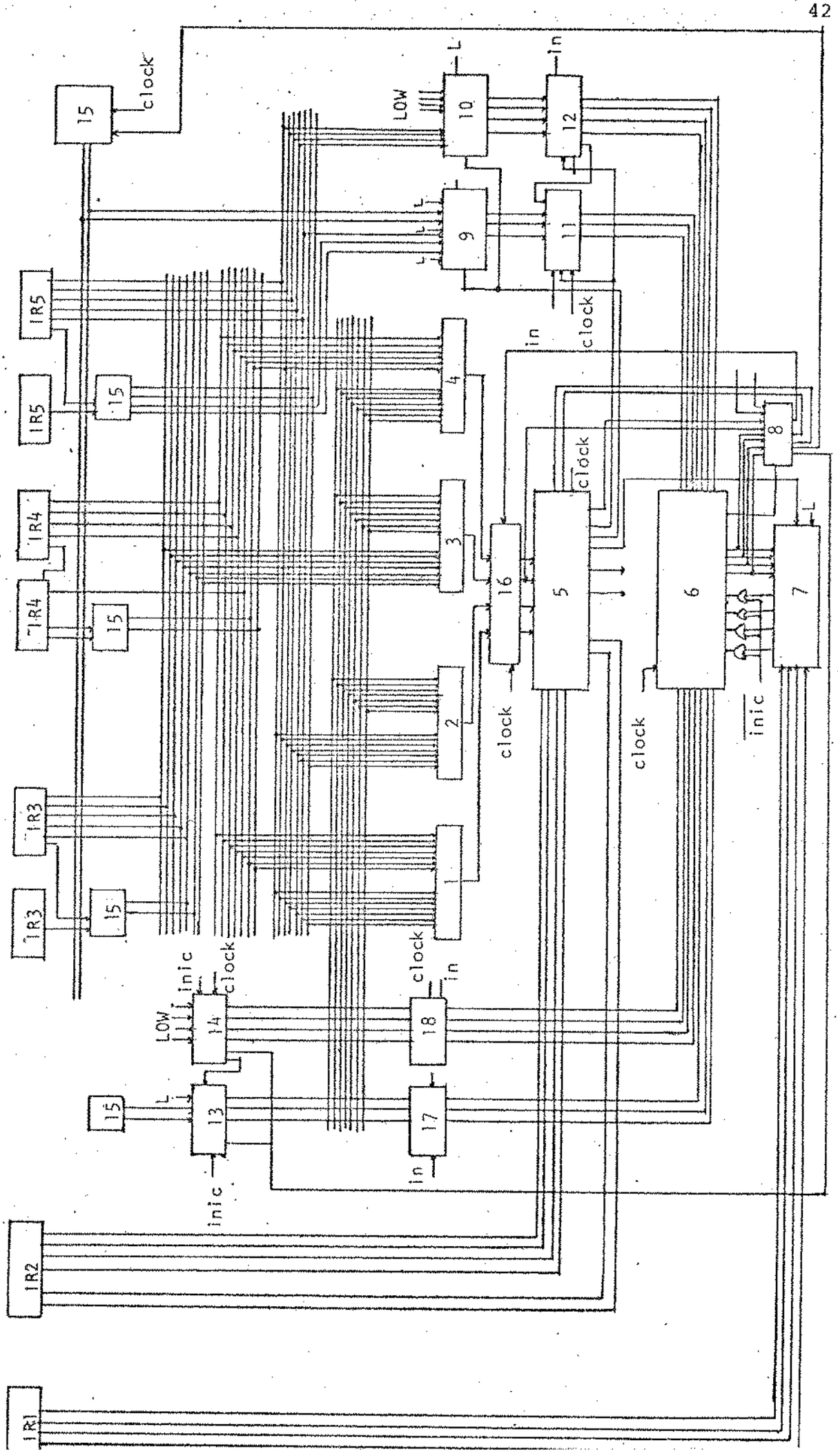


Figura 3.8 - Registros de interface e controlador da 'picture memory' (ver nota 1 na seção 3.5)

microprocessador com o endereço da primeira palavra do espaço a ser aberto ou apagado. Para as outras instruções ele contêm o endereço da palavra selecionada.

IR4 : Este registro é usado somente durante a execução da instrução (00), tem a mesma extensão de IR3, e funciona como apontador para a palavra que está sendo transferida do microprocessador para a 'picture memory'. No início da execução da instrução ele possui o mesmo conteúdo de IR3 e é incrementado toda vez que uma palavra é transferida de IR1 para a 'picture memory'.

IR5 : tem 11 bits de comprimento, é usado somente quando uma das instruções (00) ou (01) é selecionada e contem o endereço da última posição do bloco de palavras a serem inseridas ou apagadas.

Os endereços indicados por estes registros correspondem aos endereços do 'display file' vistos pelo usuário (endereços relativos). Como estes endereços não correspondem aos endereços da memória (endereços efetivos) é necessário que o circuito disponha de meios para a conversão dos mesmos. Além disso, para reiniciar a leitura, cada vez que o fim da memória é atingido precisa-se carregar os registros de endereço com valores apropriados, função dos valores previamente usados. Note-se que apenas os dois bits mais significativos do endereço devem ser corrigidos, de acordo com a tabela 3.1, pois os demais permanecem corretos.

O circuito de seleção é realizado com flip-flops tipo D sensíveis a borda, montados como contador decrescente, e o circuito de conversão é implementado a partir de somadores, com o transporte inicial previamente selecionado.

O contador de endereço de leitura (PCREAD) é reinicializado toda vez que a instrução 'end of file' é lida. Os dois bits mais significativos são carregados a partir do seletor de endereços e os demais são colocados em zero. O contador de endereços de endereços de escrita (PCWRITE), de modo semelhante ao contador de endereços de leitura, é reinicializado pela instrução 'end of file' ou quando qualquer

MSB do endereço de leitura anterior	Próximos MSB	
	Leitura	Escrita
00	11	10
01	00	11
10	01	00
11	10	01

Tabela 1.2.a - Seleção do endereço inicial (MSB)

Endereço relativo	MSB do endereço efetivo inicial							
	Leitura				Escrita			
MSB	00	01	10	11	00	01	10	11
00	00	01	10	11	11	00	01	10
01	01	10	11	00	00	01	10	11
10	10	11	00	01	01	10	11	00
11	-	-	-	-	-	-	-	-

Tabela 1.1.b - Conversão dos endereços relativo-efetivo (MSB)

correção é realizada sobre a 'picture memory'. No primeiro caso os dois bits mais significativos são obtidos do seletor de endereços e os demais colocados em zero; no segundo caso, o contador de endereços é carregado com o conteúdo do registro de interface IR5, com os dois bits mais significativos devidamente corrigidos pelo conversor de endereços. A seleção dos dados a serem carregados no PCWRITE é realizada por um circuito multiplexador acionado pelo bloco de controle.

Um conjunto de quatro comparadores (blocos 1, 2, 3 e 4 na figura 3.8), implementando a partir de comparadores de 4 bits montados em cascata, permite, sob as piores condições de atraso, a comparação de palavras de 24 bits em 33 nanosegundos. No circuito eles são denominados C1, C2, C3 e C4 e as respectivas saídas são verdadeiras se o par de registros: IR2-IR3, PCREAD-IR3, PCREAD-IR1 e PCREAD-IR2, respectivamente, tem o mesmo conteúdo.

Para que a implementação do circui

to satisfaça as restrições de tempo, o mesmo é dividido em três partes interfaceadas por registros de armazenamento temporário. (estruturas 'pipe line'). Estas três partes referem-se às funções de comparação, controle e acesso à memória, cada uma delas executadas em 60 nanosegundos.

Com esta estrutura quatro endereços diferentes são usados simultâneamente. Enquanto comparações são feitas entre o endereço n+1 e os registros de interface, o endereço n e o conteúdo do registro de armazenamento temporário é usado para controle do circuito, e os endereços n-1 e n-2, armazenados nos registros da memória são usados para realização dos ciclos de leitura e escrita,respectivamente. Deve-se notar que embora sejam usados quatro endereços, apenas três passos são necessários na estrutura 'pipe line', porque os ciclos de leitura e escrita são realizados simultâneamente pelo carregamento adequado de PCREAD e PCWRITE. Os blocos 16 e 17 da figura 3.8 representam os registros temporários dos comparadores e do contador de endereços de leitura. Os dois registros temporários da memória encontram-se dentro do bloco 6, apresentado em detalhes na figura 3.9. Todos os registros temporários são atualizados pela borda de subida do sinal do relógio. O registro temporário dos comparadores, durante o período de reinicialização dos contadores de endereço pode ser desacoplado forçando suas saídas para zero. Isto é usado para se evitar a propagação de erros que podem ocorrer quando os registros de interface ou os contadores de endereço são carregados. Este ponto será discutido posteriormente.

A memória e os seus circuitos de controle correspondem ao bloco 6 da figura 3.8 que é apresentado em detalhes na figura 3.9.Para efeitos de implementação a memória é dividida em 8 blocos de 256 palavras.Esta divisão permite leitura e escrita simultâneas sem conflitos de memória para correções de até 256 palavras de uma só vez.Note-se que este último número é função do número de blocos na implementação da memória.Em cada ciclo de memória somente dois blocos são ativados, um para leitura e outro para escrita. Os três bits mais significativos dos endereços de leitura e escrita são processados por

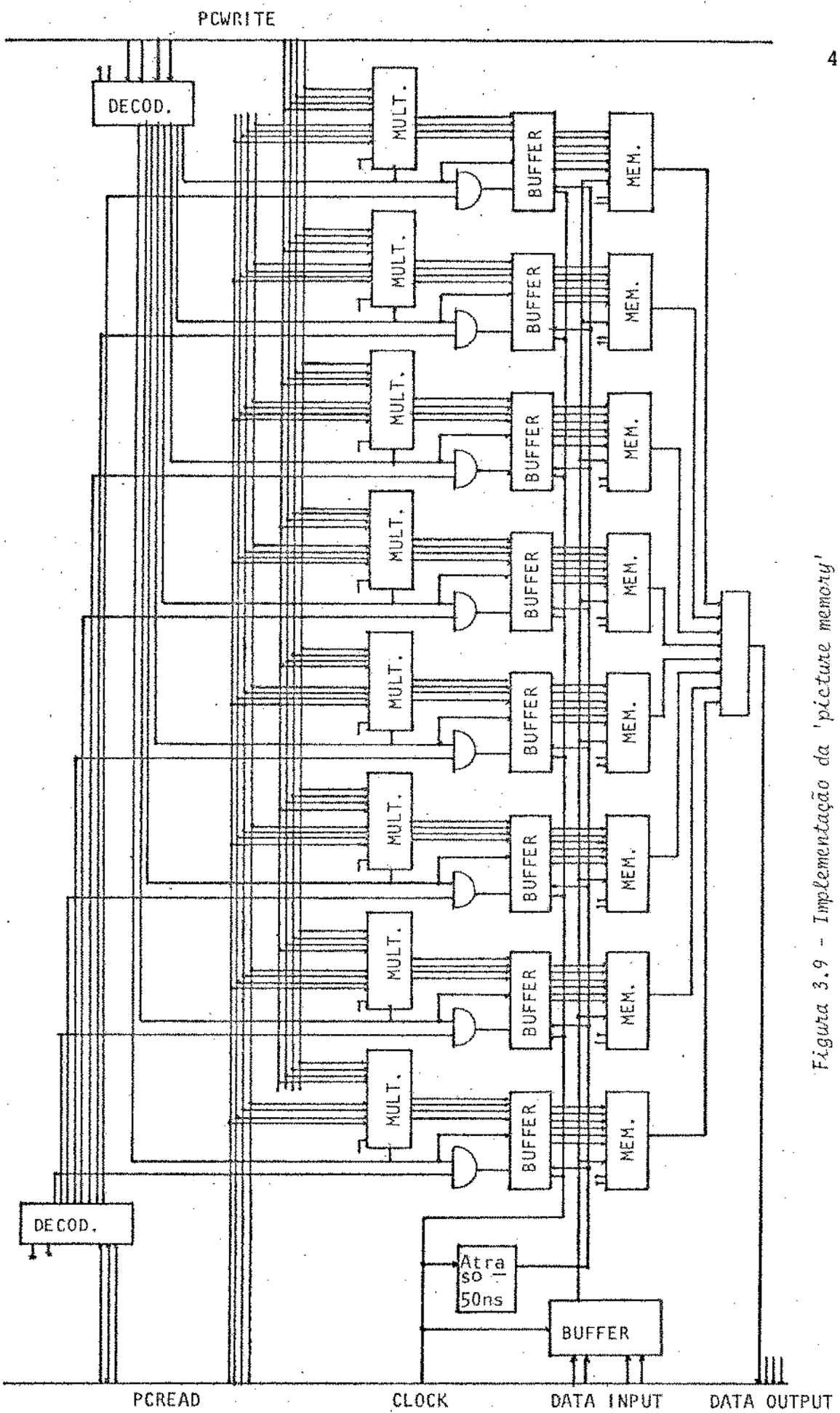


Figura 3.9 - Implementação da 'picture memory'

dois decodificadores e as saídas correspondentes são ligadas a circuitos 'AND' de duas entradas cujas saídas são usadas para seleção dos blocos desejados.

O modo de operação de cada bloco, ativado para leitura ou escrita é obtido do decodificador dos três bits mais significativos do endereço de escrita. Os endereços dentro dos blocos selecionados são obtidos usando-se as saídas do mesmo decodificador do endereço de escrita para chavear uma coluna de multiplexadores de modo a conectar os bits menos significativos do endereço de escrita ao bloco selecionado para escrita e os bits menos significativos do endereço de leitura a todos os outros blocos.

Decorridos 50 nanosegundos após a ocorrência da borda de subida do pulso de relógio, o processo de seleção dos blocos da memória deve estar completo, quando, então, os bits do registro temporário da memória relativos ao controle de seleção são atualizados. Os outros bits correspondentes aos endereços e os dados são atualizados 10 nanosegundos após a atualização dos bits de controle, tempo que corresponde ao próximo pulso do relógio (ver figura 3.10). Tal procedimento se faz necessário para satisfazer as condições de 'hold time' requeridos pela memória. O ciclo de escrita na memória pode ser suprimido inibindo-se o decodificador dos bits mais significativos do endereço de escrita.

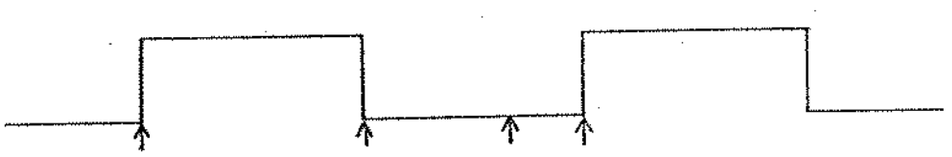


Fig. 3.10 Diagrama de tempo para atualização dos dados de entrada da memória

As funções de controle de todo o circuito são realizados pelos blocos 5 e 8 da figura 3.8 As funções executadas pelo bloco 8 geram os sinais necessários para a reinicialização da varredura da memória, após a instrução 'end

of file'.O bloco 5, por outro lado, é responsável pelos sinais que permitem a execução da instrução selecionada. Neste bloco as instruções são decodificadas e, de acordo com os valores das saídas do conjunto de comparadores, são gerados os sinais que permitem a atualização da palavra de controle ('status word') e a transferência do conteúdo do registro de interface IR5 para o contador de endereços de escrita ou do conteúdo do registro de interface IR1 para a memória.

O circuito correspondente ao bloco 5 é apresentado na figura 3.11. Sempre que qualquer ação sobre a 'picture memory' é solicitada pelo microprocessador, o bit CW1 da palavra de controle é colocada na condição verdadeira e a execução da instrução é iniciada quando a saída do comparador C3 (PCREAD = IR1) assume pela primeira vez a condição verdadeira. A saída do flip-flop formado por 5a e 6b é colocada em zero e permanece nesta condição até que a instrução 'end of file' seja encontrada, exceto se a instrução selecionada é (00). Esta instrução necessita mais de uma varredura da memória para ser executada e o flip-flop citado muda de condição somente quando a instrução for completada, condição indicada pelo comparador C1 (IR2 = IR3), ou seja, todo o espaço aberto na memória já estiver carregado com novos valores vindos do microprocessador. Os mesmos sinais usados para atuar sobre o flip-flop mencionado são também usados para selecionar o registro da interface IR5 e transferir seu conteúdo para o contador de endereços de escrita ou, ainda, para mudar o estado do bit CW1 da palavra de controle quando a instrução selecionada tiver sido completamente executada.

Na execução de uma instrução (00), durante o intervalo de tempo entre abertura e preenchimento do espaço, o conteúdo destas posições não devem ser usados para se evitar distorções na imagem. Desse modo, estas palavras são lidas na memória mas não devem ser transferidas ao conversor do 'display file'. Um indicador de erro ('error flag 1'),flip-flop formado por 5c e 6a, assume a condição verdadeira quando a primeira palavra deste espaço é atingida pelo contador de endereço de leitura (saída do comparador C3 verdadeira) e permanece neste estado até que a última palavra do espaço seja atingida (saí

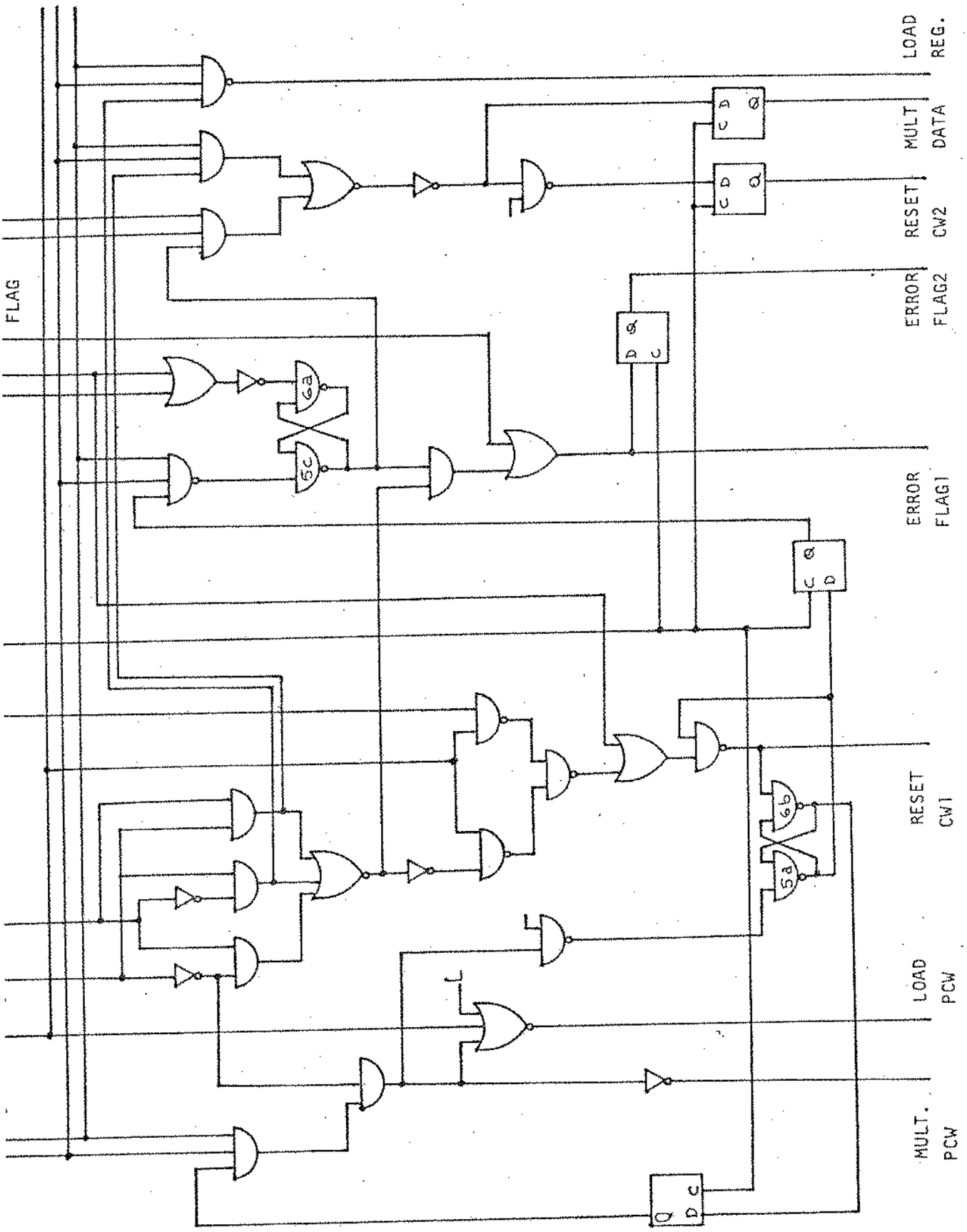


Figura 3.11 - Controlador

da do comparador C2 verdadeira).

O mecanismo usado para carregamento do espaço aberto na memória esta baseado no fato de que toda palavra lida da memória é reescrita três blocos adiante. O registro de interface IR4 é usado como um apontador e indica o endereço da memória para o qual o conteúdo do registro de interface IR1 deve ser transferido. Quando a posição indicada por IR4 é atingida pelo contador de endereços de escrita, indicado pelo comparador C4, e o dado em IR1 está pronto para ser transferido, indicado pelo bit CW2 da palavra de controle, o multiplexador de dados da memória (bloco 7 na figura 3.8) é chaveado e o conteúdo de IR1, ao invés do valor lido da memória, é transferido ao registro temporário de escrita da memória, atualizando o conteúdo da posição indicada. Ao mesmo tempo o registro IR4 é incrementado, apontando agora para a próxima palavra a ser preenchida, e o bit CW2 da palavra de controle é complementado atualizando o 'status' do sistema. Os flip-flops 11a e 12a geram atrasos apropriados para o chaveamento do multiplexador de dados da memória e para a atualização do 'status' pois os ciclos de escrita e leitura são defasados de uma posição de memória. Se a instrução (10) é selecionada o mesmo procedimento é usado, simplificado porque nesse caso apenas uma palavra deve ser escrita. Quando a saída do comparador C3 e o bit CW2 da palavra de controle são verdadeiros o multiplexador de dados da memória é chaveado e o processo já descrito é repetido.

A última instrução a ser discutida, transferência de uma única palavra da 'picture memory' para o microprocessador, é realizada do seguinte modo: quando esta instrução é selecionada e a saída do comparador C3 indica que a palavra desejada foi lida na 'picture memory' é gerado o sinal 'load reg' que permite o carregamento do registro de interface correspondente a esta função (descrita na seção 2.3) no próximo pulso do relógio. O mesmo sinal deve ser usado para interromper o microprocessador e iniciar a transferência do dado do registro de interface para o microprocessador.

As funções de reinicialização da

varredura da memória são executadas pelo bloco 8 da figura 3.8, mostrado na figura 3.12. Os quatro bits mais significativos de toda palavra lida da memória são transferidos a um registro e então decodificados. Se a instrução 'end of file' é detetada, os quatro bits mais significativos da palavra verdadeiros, o registro citado é inibido e prepara-se o reinicio da varredura da memória. Um diagrama de tempo com os sinais mais importantes deste procedimento é apresentado na figura 3.13.

Após a instrução 'end of file' ter sido encontrada são necessários quatro ciclos de memória para que os contadores de endereço sejam carregados convenientemente e durante os quais palavras que não pertencem à descrição da figura são lidas. Para evitar que estas palavras sejam transferidas ao conversor do 'display file' um indicador de erro (error flag 2) é colocado na condição verdadeira, permanecendo neste estado até que todo o processo de reinicialização tenha sido completado.

Quando o seletor de endereço de memória é atualizado, pelos sinais gerados pela instrução 'end of file', (ver figura 3.13) os conversores de endereço também o são. O atraso causado por este evento, somado ao atraso dos comparadores é maior do que 60 nanosegundos, tornando possível a propagação de erros causados por valores errados nas saídas dos comparadores. Esta propagação é evitada inibindo-se o registro de armazenamento temporário dos comparadores durante o ciclo de relógio que se segue a atualização do seletor de endereços, forçando-se, portanto, a propagação de zeros.

Se a descrição da figura ocupa menos que 256 palavras, podem ocorrer simultaneamente ciclos de leitura e escrita no mesmo bloco de memória durante a reinicialização da varredura da memória, provocando erros pois os contadores de endereço para leitura (PCREAD) e escrita (PCWRITE) não são carregados simultaneamente. Como o ciclo de escrita não é essencial durante esta fase, o mesmo é suprimido inibindo-se o decodificador dos bits mais significativos do contador de endereços de escrita realizando-se apenas o ciclo de leitura.

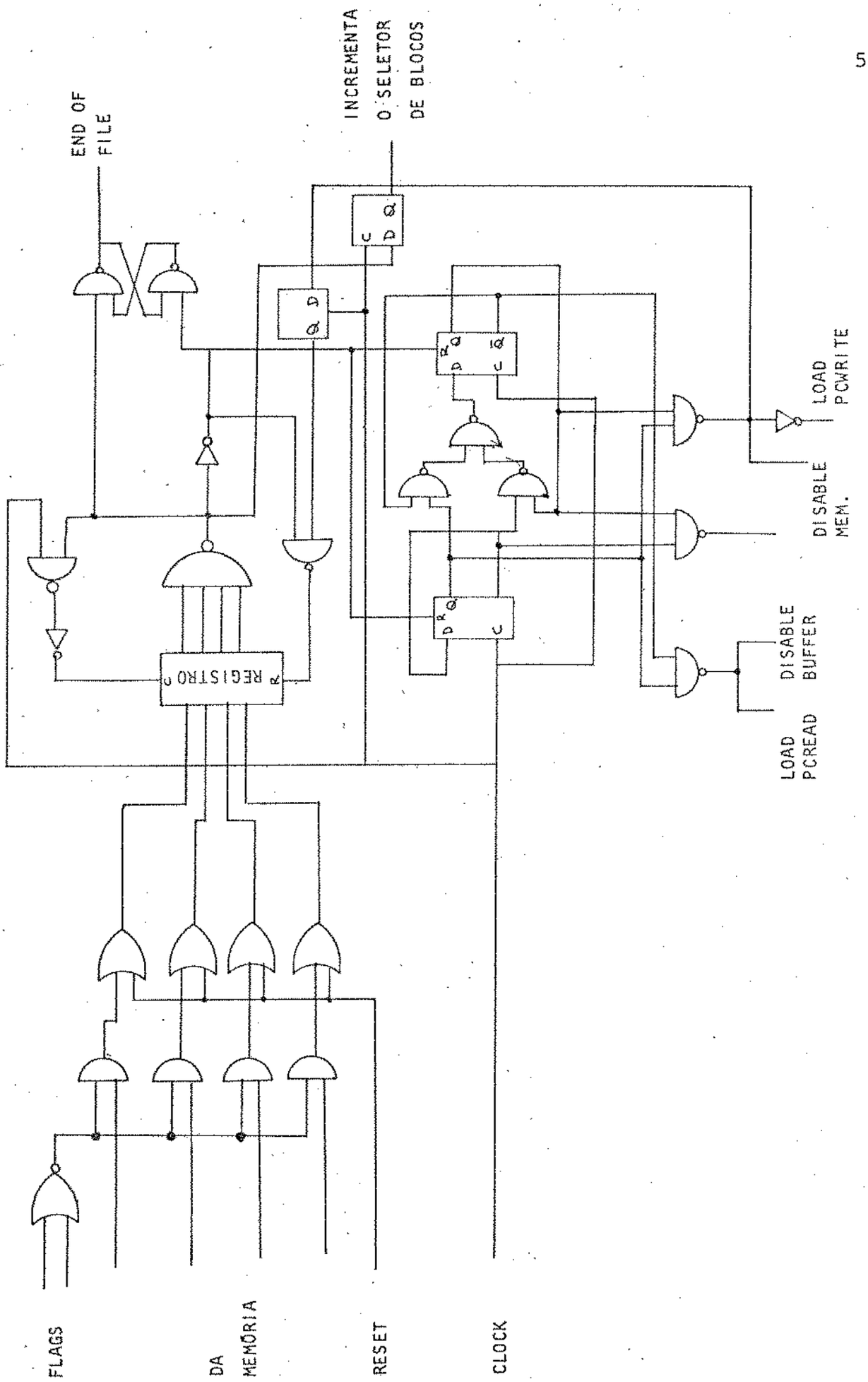


Figura 3.12 - Controlador do reinício da varredura da memória

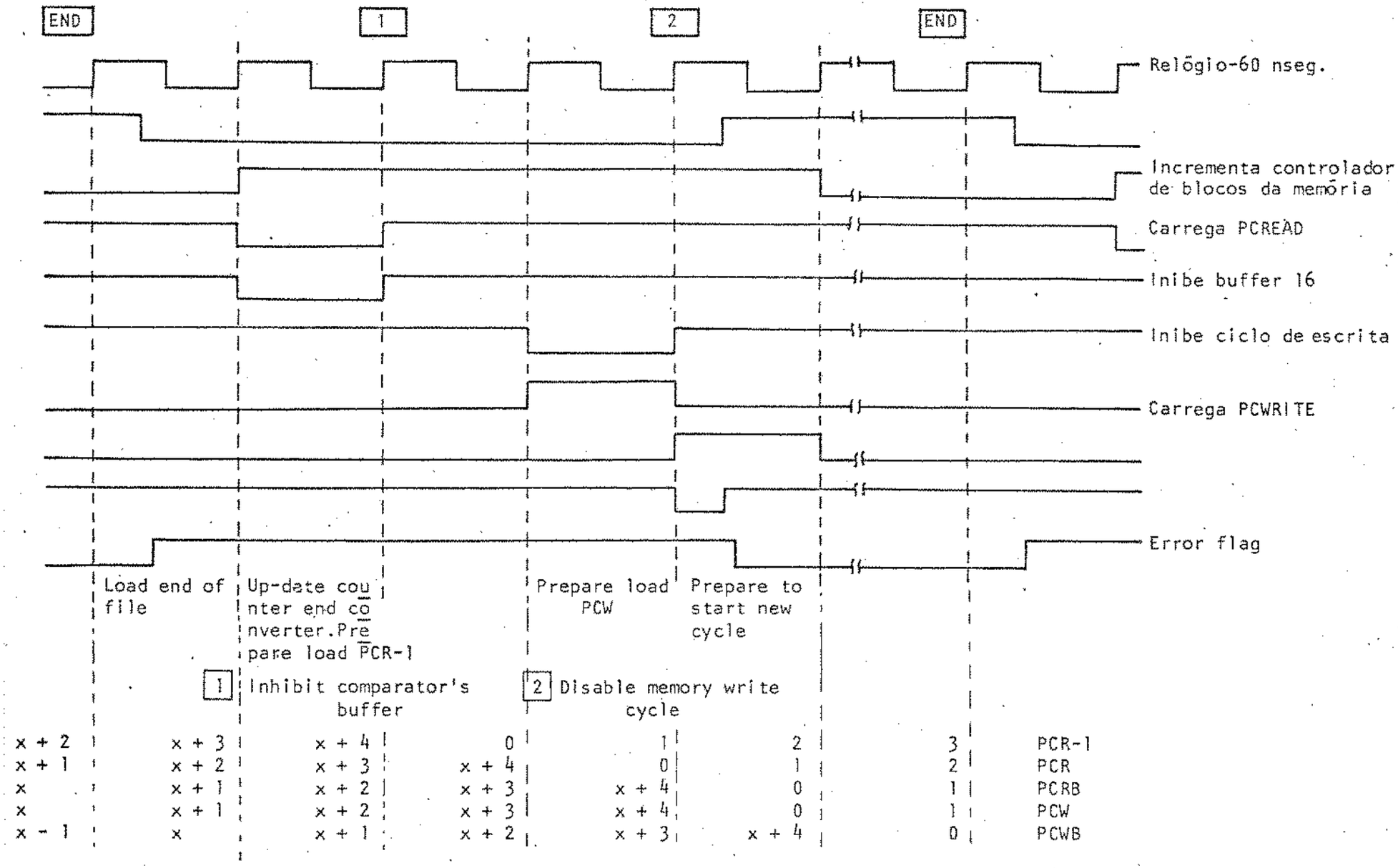


Figura 3.13 - Diagrama de tempo dos sinais para reinício da varredura da picture memory

O circuito detecta e ignora instruções 'end of file' que se encontram dentro de espaços de memória que estão sendo atualizados; e, ao ser ligado, força a execução desta instrução, para que a varredura da memória seja iniciada com calores apropriados.

3.3 - A interface microprocessador - 'picture memory'

Toda ação sobre a 'picture memory' é realizada através dos registros de interface, Estes registros, vistos do microprocessador, estão divididos em dois grupos denominados entrada e saída e estão conectados as respectivas barras de dados do mesmo. Um diagrama mostrando estes registros, o fluxo de dados e controle é apresentado na figura 3.14. A seguir descreve-se resumidamente a interface microprocessador - 'picture memory' apresentada com maiores detalhes em |12|.

A troca de dados entre o terminal (micro-processador) e o computador central (usuário) é feita em série e,para a troca de dados entre o microprocessador e os registros de interface, quatro canais paralelos são usados, dois de saída e dois de entrada.

Como a arquitetura do microprocessador escolhido, Zilog 80, baseia-se num 'bus' de oito bits |13|, e a extensão das palavras da 'picture memory' é de 16 bits, são necessários dois passos para a transferência de dados do microprocessador para o registro IR1. O registro IR1 está dividido em dois registros de 8 bits, com endereços diferentes um dos outros, carregados em duas operações de saída como se explica a seguir. Os registros IR3, IR5 e a palavra de controle IR2 são carregadas em uma só operação de saída pois somente 11 bits, são necessários para endereçamento da 'picture memory' e a palavra de controle (IR2) possui somente 4 bits. A transferência, tanto dos dados da 'picture memory' como do conteúdo dos registros de interrupção pela 'light pen', para o microprocessador, é realizada em duas operações de entrada. Embora possuam 12 bits, estes últimos registros devem ser transferidos em dois passos devido às restrições impostas pelo 'bus' de oito bits do microprocessador.

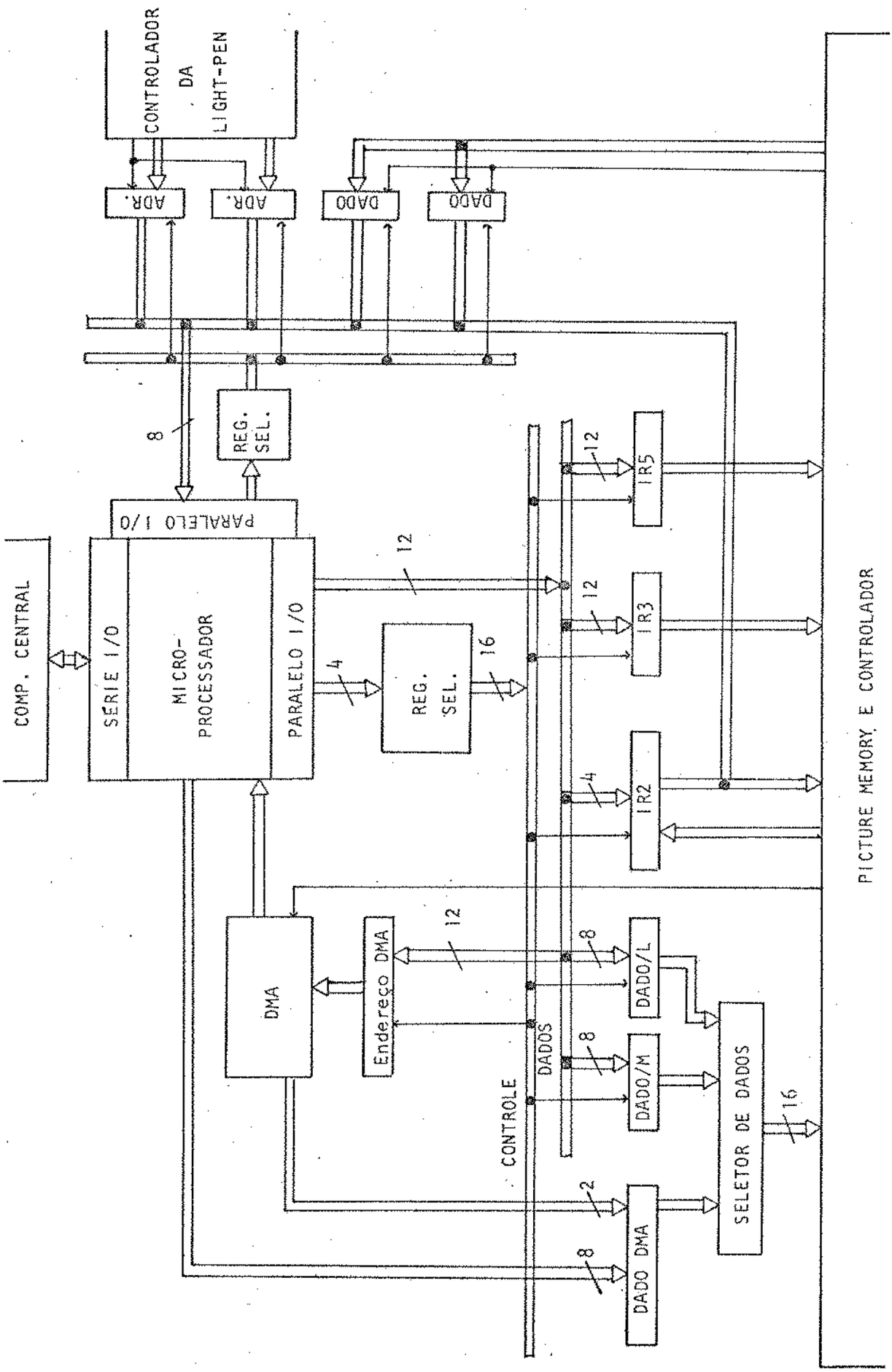


Figura 3.14 - Microprocessador e interface com a picture memory

A entrada e saída paralelas de dados são implementadas usando-se duas Z80 PIOS, cada uma composta de duas portas bidirecionais programáveis e independentes, e com sinalização da transferência de dados |14|. A saída consiste de duas portas que operam no modo zero (saída), concatenados de modo a operar como um registro de 16 bits. Os 4 bits mais significativos são usados para selecionar um dos registros de interface, e os 12 bits restantes para a transferência do dado propriamente dito. A seleção é feita por um decodificador dos 4 bits mais significativos que ativa o registro selecionado e permite a transferência dos dados, os 12 bits menos significativos,ao ser ativado o sinal 'data ready'.

A entrada consiste também de duas portas,uma operando no modo saída e a outra no modo entrada. Através da porta que opera no modo saída um dos registros é selecionado e seu conteúdo transferido ao microprocessador através da porta que opera no modo entrada. Note que o registro IR2, que contem o 'status' do controlador, pode ser lido, permitindo que o microporcessador verifique o estado do controlador antes da execução de qualquer instrução.

Outra característica importante da interface são os circuitos que permitem o acesso direto a memória (DMA) do microprocessador durante a transferência de dados do microprocessador para a 'picture memory'. As principais vantagens obtidas são: a garantia da transferência de dados para a 'picture memory' na taxa desejada, uma palavra cada varredura da 'picture memory' (tempo 64 microsegundos); e a capacidade de microprocessador é usado somente para iniciar o processo de transferência de dados pelo carregamento dos registros de interface, e a transferência de dados, propriamente ditos, é efetuada pelo canal de DMA. Isto aumenta o tempo disponível para processamento local, tornando o terminal menos dependente do computador central. Se somente uma palavra deve ser transferida do microprocessador para a 'picture memory' (instrução (10)), a transferência é realizada através dos registros DATA/M e DATA/L (ver figura 3.14), chaveando-se corretamente o seletor de dados para a 'picture memory'.

O diagrama de blocos do circuito usado para a realização do DMA é mostrado na figura 3.15. O contador de 8 bits e o registro (blocos 1 e 2 da figura, respectivamente) são carregados pelo microprocessador com o endereço inicial do bloco a ser transferido. Durante a transferência de dados o flip-flop 5 está ativo e é gerado um sinal de requisição do 'bus' se o flip-flop 3 também o estiver (ação sobre a 'picture memory' requisitada). Quando a requisição é aceita, o sinal Busak |13| é gerado pelo microprocessador ativando o contador quatro que inicia a contagem usando o mesmo relógio do microprocessador. As saídas do contador são usadas para geração dos sinais de controle dos ciclos de memória que se seguem.

Como a palavra da 'picutre memory' é de 16 bits e do microprocessador é de 8 bits, são necessários dois ciclos de leitura para a transferência do dado. Quando o contador 4 está no estado 1, o flip-flop 6 é apagado gerando os sinais MREQ e RD. No estado tres o mesmo flip-flop é ativado, o contador 1 é incrementado, apontando agora para a próxima palavra da memória do microprocessador, e o conteúdo da posição lida é armazenado no registro R1. Para o estado interno 4 o procedimento é repetido de modo a acessar a segunda palavra do dado. No estado 5, o sinal de requisição do 'bus' é desativado, liberando o microprocessador. No estado 6, o contador 1 é incrementado, atualizadno o endereço para acesso a memória do microprocessador, o registro R2 é carregado com a segunda parte do dado e o bit de controle CW1 da palavra de controle, é ativado.

Um novo ciclo de acesso deireto a memória será gerado somente quando a transferência, do último dado acessado tiver sido concluída, condição indicada pelo bit CW2 da palavra de controle. O processo é repetido até que todo o bloco tenha sido transferido, desativando-se o bit CW1 da palavra de controle e inibindo-se o circuito de DMA através do flip-flop 3.

As funções de regeneração da memória não são implementadas uma vez que podem ser executadas sem qualquer problema pelo microprocessador pois somente dois ciclos de acesso direto a memória são realizados de cada vez. Na figura 3.16 é apresentado um diagrama típico dos sinais gerados pelo circuito descrito.

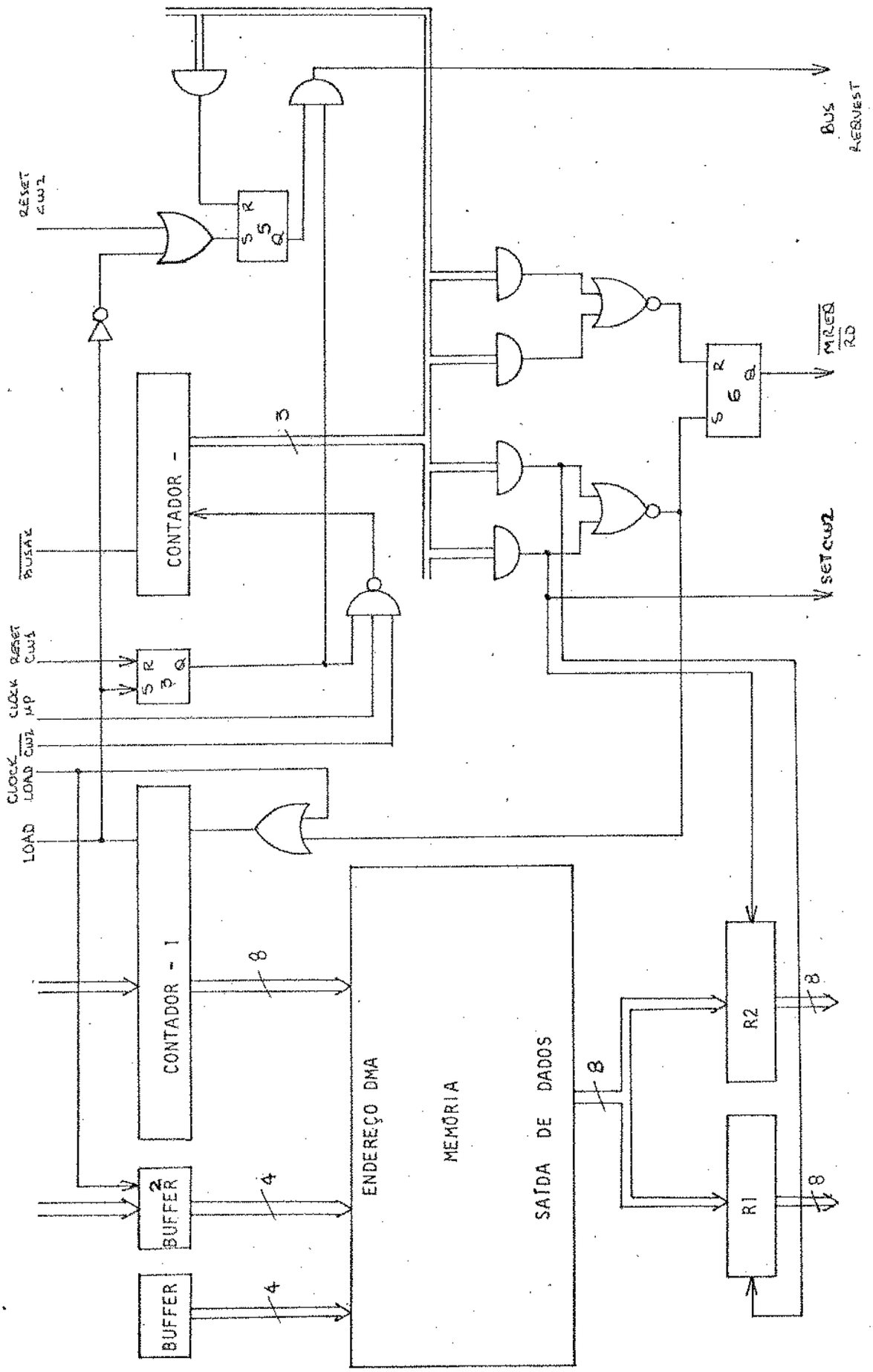


Figura 3.15 - Controlador do acesso direto à memória

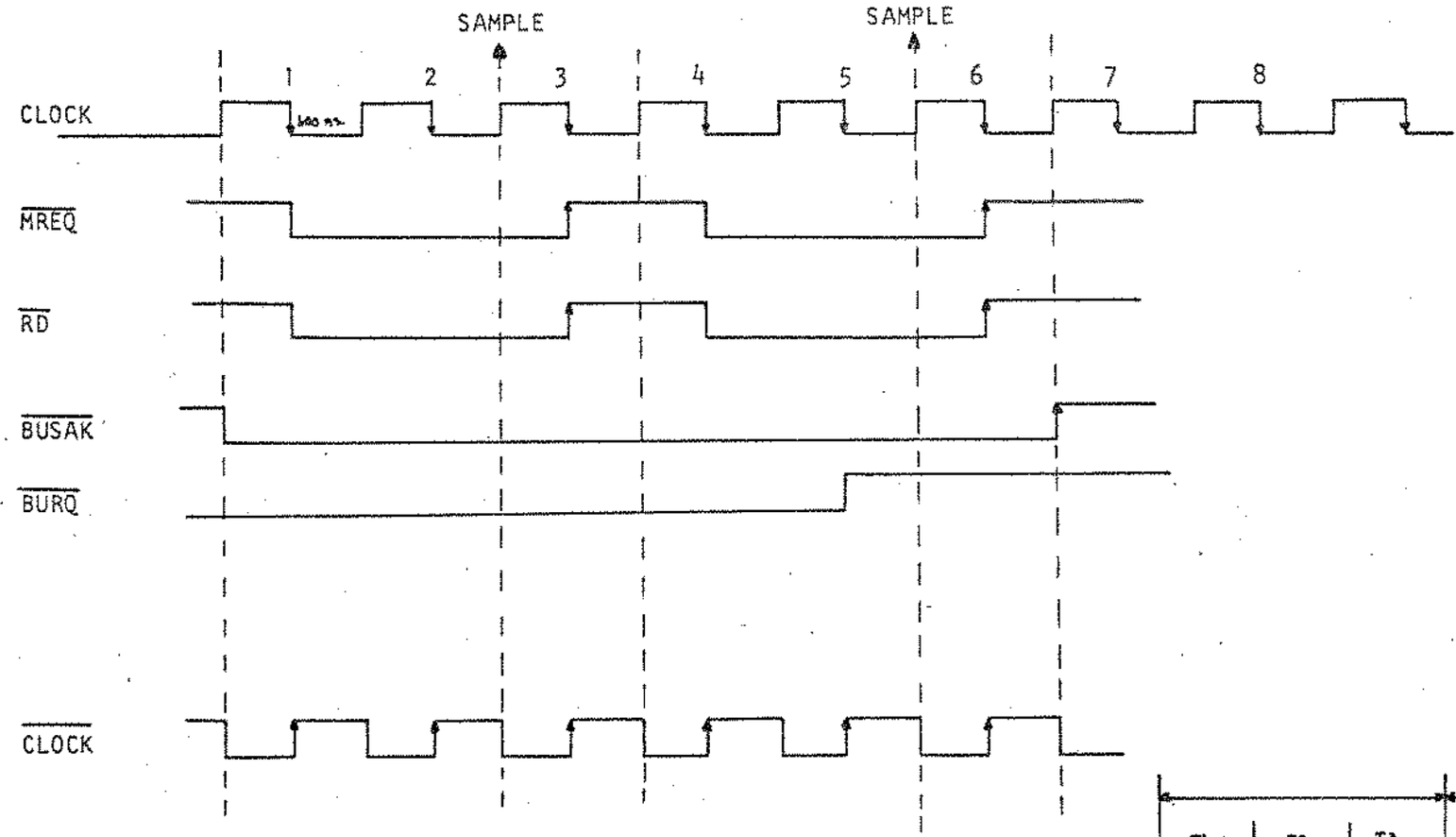
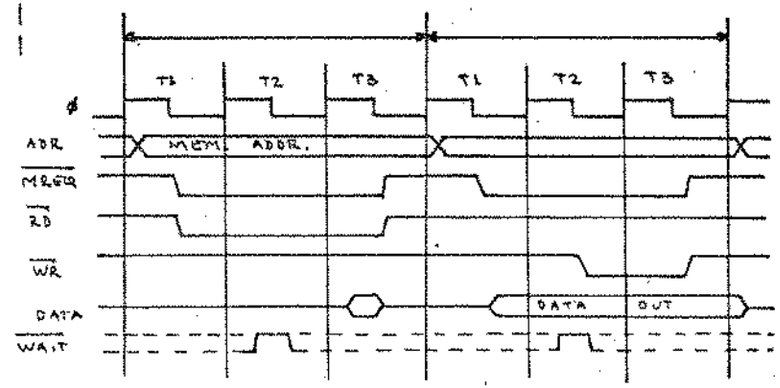


Figura 3.16 - Diagrama de tempo dos sinais para controle do DMA



Ciclo de leitura ou escrita típico

Outro ponto que deve ser citado é a estrutura de interrupção do microprocessador. Toda ação sobre o microprocessador é causada por interrupções, exigindo-se o uso de circuitos para codificação e definição de prioridades das mesmas. O microprocessador é programado para responder as interrupções no modo dois |13|, isto é, quando uma interrupção é aceita, um endereço de 16 bits é formado pelos 8 bits do registro I do microprocessador e os 8 bits do codificador de interrupções, com o qual pode-se fazer uma chamada para qualquer posição da memória. Uma tabela com os endereços iniciais das subrotinas de atendimento às interrupções é mantida na memória e é usada para o endereçamento indireto das mesmas. O sinal de interrupção é amostrado no fim da execução de cada instrução e, se estiver ativo, ao invés da execução da próxima instrução, gera-se um sinal que permite a entrada dos 8 bits que compõe a parte menos significativa do endereço.

3.4 - Software básico para o microprocessador

Para a realização das funções inerentes ao controle da interface e para a capacitação local de cálculo, é necessário dotar o microcomputador de um software básico mínimo. Este software básico, residente em memória não volátil (ROM), deve ser capaz de inicializar o sistema, efetuar o tratamento de interrupções e realizar qualquer ação sobre a 'picture memory'. O 'software' mínimo necessário está resumido na tabela 3.3.

Os programas de inicialização do sistema, conjunto A da tabela 3.3 atuam como um 'bootstrap' e são executados sempre que o sistema é ligado, definindo o modo de operação das unidades de entrada e saída, parâmetros de interrupção, etc.

Os programas do conjunto B recebem os comandos externos e efetuam sua execução. Estes comandos podem ser divididos em dois grupos, o primeiro relacionado com ações na interface e o segundo com ações sobre o próprio microcomputador. Estas últimas incluem:

- carregamento da memória do microprocessador com dados ou programas do usuário;

a - inicialização

alocação da memória: . pilha
. DMA
. espaço para transferência de parâmetros
. espaço de trabalho
. vetor de endereços de interrupção.

definição do modo de operação PIOS
definição do modo de operação USART

b - carregamento e seleção de comandos
ações sobre a interface ('picture memory')
operações na memória do micro processador:
. carregamento
. saída de dados
. execução
...

c - operações de saída na interface
carregamento dos registros de interface para execução das instruções definidas na seção 3.2

d - operações de entrada na interface
leitura do estado do controlador
leitura dos dados gerados por interrupção pela 'light-pen'

e - atendimento de interrupções:
. usuário (TTY, HOST)
. 'light-pen'
. saídas paralelas
:

TABELA 3.3 - Software básico

- saída de dados ou transferência de dados do micro processador para o computador central;
- execução dos programas do usuário, executando-se saltos para as posições onde os programas estão armazenados.

Nos terceiros e quartos blocos estão os programas relacionados com a interface da 'picture memory', cujas funções são o carregamento adequado dos registros de interface e a leitura dos registros de 'status' e de interrupções pela 'light pen'. Estes programas, como descrito em 3.3, efetuam o carregamento e leitura dos registros de interface, através dos canais de saída paralela, concatenando os dados por meio de diversas operações de entrada e saída. Atualmente, com a disponibilidade dos microprocessadores de 16 bits, tal procedimento pode ser simplificado efetuando-se tais operações em um só passo. O uso de um microprocessador de 16 bits também implica num sensível aumento da potência local de cálculo e na facilidade de programação de aplicações específicas. Dentro da mesma filosofia enquadram-se os microprocessadores 'bit-slice', que, além disso, permitem a microprogramação de funções especiais. Como a arquitetura da interface é praticamente independente do microprocessador usado, a troca do mesmo corresponde unicamente a reedição de partes dos programas, permitindo-se facilmente a atualização do projeto.

Os diagramas de blocos da figura 3.17 mostram, sem detalhes, o fluxo de controle e ações do 'software' básico mínimo para funcionamento do sistema. Como foi exposto na seção anterior, toda atividade do microprocessador é iniciada pelo mecanismo de interrupção, que através do 'bloco seletor' dirige o microprocessador para a opção requerida. O bloco seletor pode ser atingido por dois caminhos, a partir do estado de espera, quando da inicialização do sistema, ou a partir do programa do usuário, pela ocorrência de qualquer das interrupções projetadas: canais de saída paralela, teletipo (computador central), controlador da 'picture memory' e controlador da 'light-pen'.

As interrupções pelo teletipo são seguidas da leitura de um comando, cinco deles correspondentes a ações na

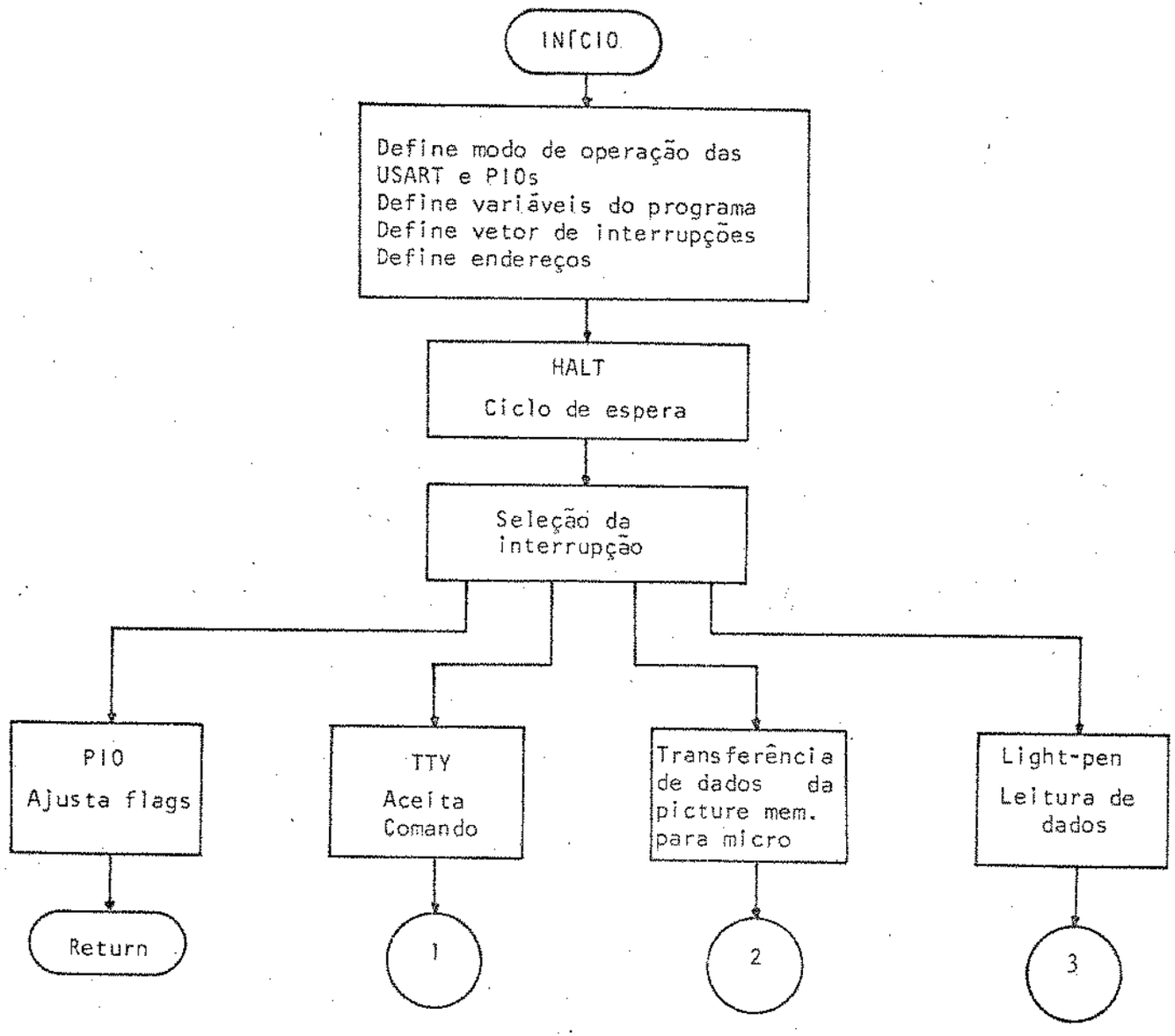


Figura 3.17a - Diagrama de blocos do software básico

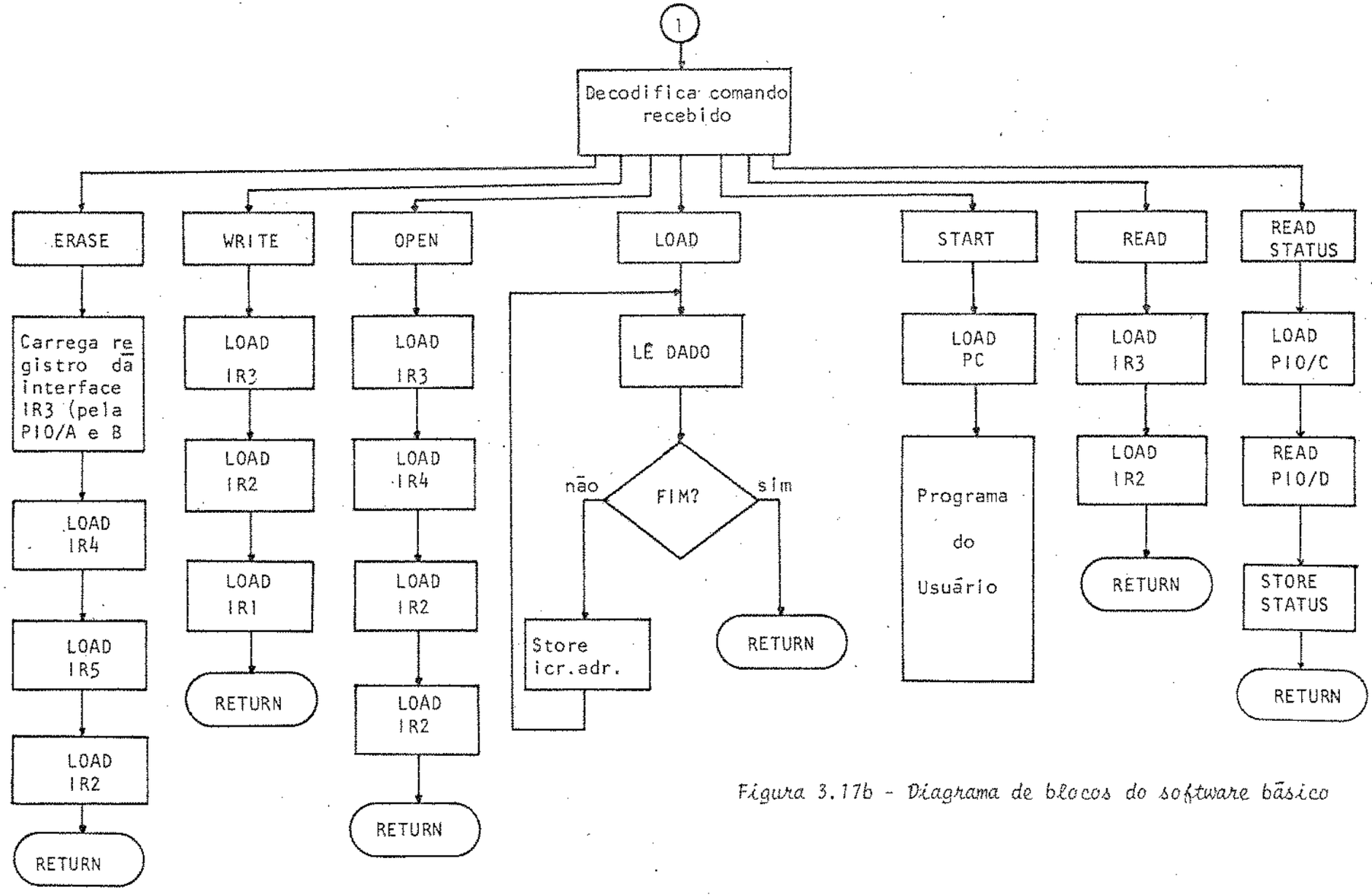


Figura 3.17b - Diagrama de blocos do software básico

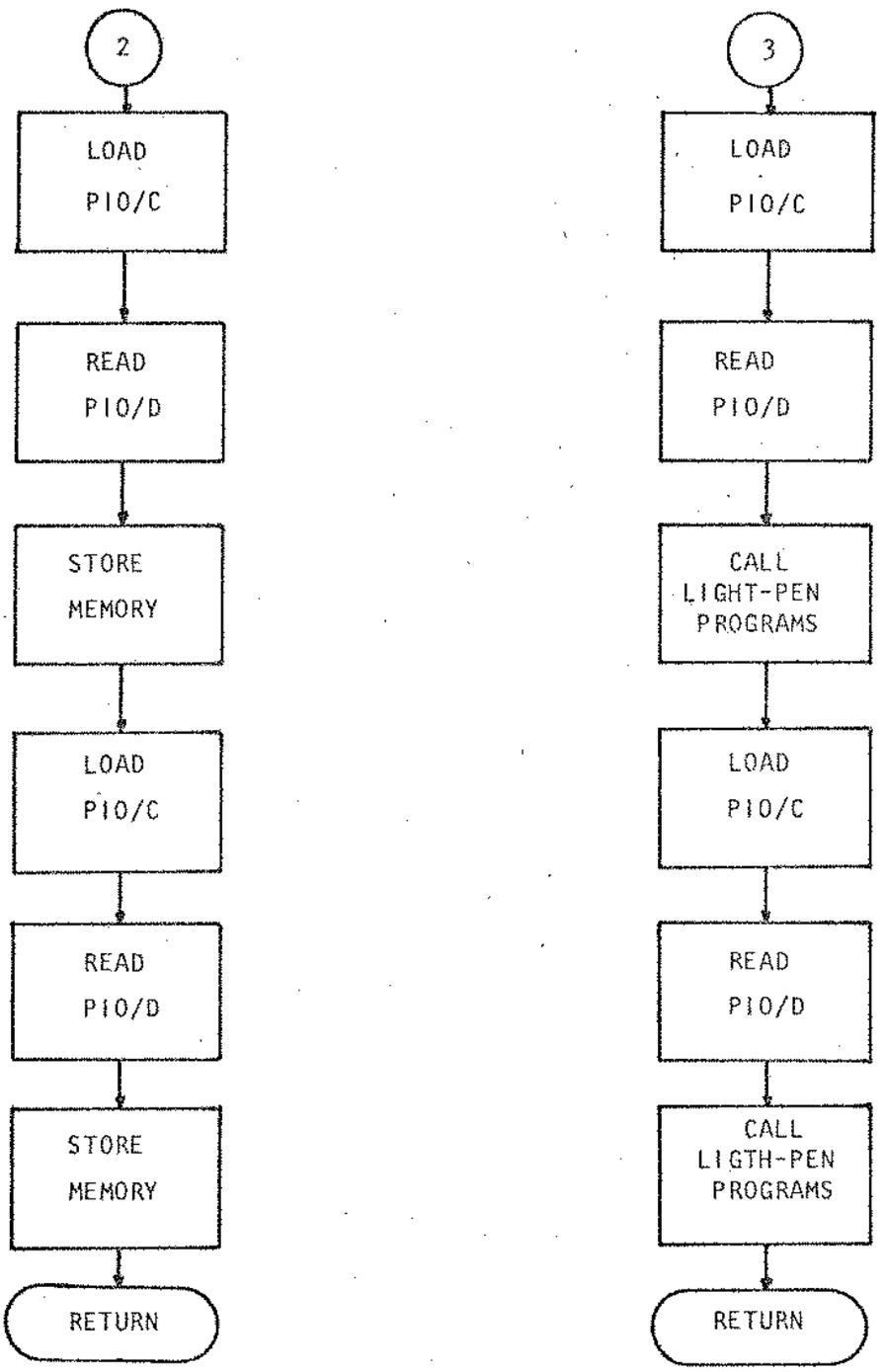


Figura 3.17c - Diagrama de blocos do software básico

'picture memory' ou seu controlador (open, erase, write, read, instruções 00,01,10,11, respectivamente) e leitura do 'status' do controlador; e dois deles correspondentes a ações no próprio microprocessador: carregamento da memória do microprocessador e execução do programa do usuário (figura 3.17.b). Nas demais interrupções uma sequência fixa de comandos é executada (figura 3.17.c). Deve-se notar que, por ser o software básico implementado sob a forma de subrotinas, a execução de qualquer das suas partes pode ser iniciada por comandos inseridos no programa do usuário.

3.5 - Simulação do 'hardware' e 'software' descritos.

Para verificar a correção e funcionamento dos circuitos projetados para o terminal gráfico descrito nas seções anteriores os mesmos foram simulados em computador |15|, e os principais motivos para isso foram:

- O teste e a eliminação de erros no circuito é muito mais fácil prinicipalmente para circuitos complexos e de alta velocidade;
- Correções efetuadas durante a simulação não ocupam tempo na busca de componentes e na conexão dos mesmos ao circuito (soldagem, alimentação , etc.);
- Documentação do circuito e mapa de conexões são obtidas sem qualquer esforço extra;
- A partir da simulação os circuitos são implementados sem grandes dificuldades.

O simulador 'SIMGA5 A subroutine Package for Low Level Digital Hardware Simulation' |16|, que permite a descrição de circuitos a nível de 'gates' (ver resumo no apêndice 1) foi utilizado. Pela análise dos caminhos críticos, baseada nos tempos de atraso característicos constantes nos catálogos de referência, comprovou-se a viabilidade dos circuitos e a observancia dos limites de tempo impostos para o funcionamento dos

mesmos.

Para a simulação da estrutura 'pipe line'do controlador da 'picture memory', um dos pontos mais críticos em função do tempo, usaram-se os piores valores de atraso. A 'picture memory', projetada utilizando-se pastilhas SN74S206 ('256 bit read/write memory'), revelou-se um ponto de estrangulamento que forçou o aumento do tempo do ciclo do relógio para os 60 nanosegundos. Das simulações* verificou-se também, que se memórias mais rápidas estiverem disponíveis o ciclo do relógio poderá ser reduzido a 50 nanosegundos, sem problemas para as demais partes do circuito, possibilitando a leitura completa do 'display file' no tempo disponível para a varredura de uma linha. Todos os demais circuitos foram testados exaustivamente e verificou-se a consistência dos resultados obtidos. A documentação obtida através da simulação não é anexada ao trabalho e será objeto de publicação posterior juntamente com os circuitos projetados.

Para verificação do funcionamento da interface da 'picture memory' com o microprocessador,assim como o 'software' básico para este último, descrito na seção 3.4, foi usada a estrutura apresentada na figura 3.18. Nesta estrutura o microprocessador é conectado a um minicomputador (Varian 73) no qual foram simulados (TH Darmstadt, Alemanha), a nível de registros e utilizando-se a linguagem Fortran, a 'picture memory' e

---

*Nota 1: A 'picture memory' foi projetada com 1.5K palavras de memória, exingindo-se 11 bits para seu endereçamento. Devido à limitação da memória do computador usado e para obter-se maior rapidez na simulação, a 'picture memory', para efeitos de simulação, foi reduzida a 256 palavras, necessitando-se apenas 7 bits para endereçamento. Os resultados obtidos deste modo pemanecem válidos porque os comparadores e os contadores são síncronos e a adição dos circuitos à memória para obtenção dos 16 bits é feita em paralelo, não se aumentando o tempo de atraso. Os circuitos de controle não são afetados pois independem do comprimento das palavras da 'picture memory'.

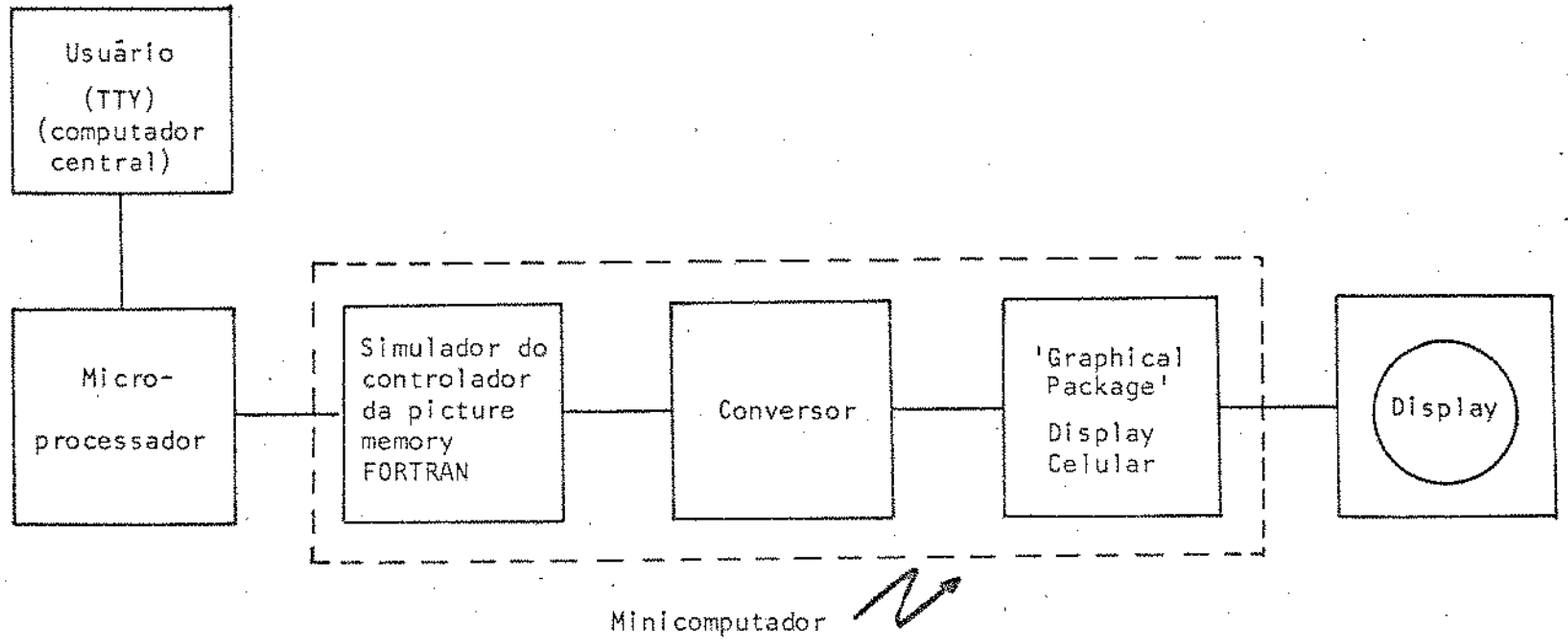


Figura 3.18 - Estrutura usada para teste do software básico

o respectivo controlador |12|. Para tal simulação as condições de tempo foram relaxadas, pois a simulação a nível de 'gates' mostrou a viabilidade dos circuitos projetados, permitindo o uso de um relógio interno simulado. O microprocessador recebe comandos e dados do teletipo (teletipo atuando como computador central) e pelo 'software' básico efetua a transferência de dados para os registros de interface simulados no minicomputador. Deste modo o simulador, de acordo com seus estados internos, realiza a transferência destes dados para um vetor de características semelhantes a 'picture memory' projetada. Este vetor é processado através de um pacote gráfico, gerando-se os códigos apropriados para um 'display' colorido organizado a base de células (Dornier, TH Darmstadt, Alemanha). Tal simulação permite a avaliação do desempenho da interface projetada, a observância dos requisitos básicos necessários ao 'software', bem como a possibilidade de desenvolvimento de programas de aplicação paralelamente à implementação dos circuitos. Detalhes das subrotinas usadas nesta simulação são apresentadas em |12|.

CAPÍTULO 4 - MODELO DE INTERAÇÃO

No transcorrer dos últimos anos os computadores deixaram de ser uma ferramenta usada apenas por especialistas na solução de seus problemas e tornaram-se disponíveis a pessoas não especializadas através de terminais interativos. Projetistas e construtores, que até então procuravam aperfeiçoar seus equipamentos concentrando seus esforços no desenvolvimento da tecnologia eletrônica, para melhorar o fluxo de informação entre os diversos componentes através de 'hardware' e'software', descobriram a necessidade de se considerar também o usuário e suas características como parte integrante do projeto.

No desenvolvimento de um sistema interativo quatro fatores devem ser considerados:

. a tarefa a ser executada;

. o usuário;

. o terminal disponível;

. as informações armazenadas no computador.

Além disso, no relacionamento homem-máquina, deseja-se juntar as capacidades do homem e da máquina de modo que nada seja feito por um deles se o outro puder fazê-lo melhor.Em outras palavras, no projeto de um sistema interativo as funções a serem executadas devem ser distribuídas de modo eficiente.Funções onde, por exemplo, a criatividade ou o reconhecimento de padrões se faz necessário, devem ser delegadas ao homem, enquanto a manipulação de grandes quantidades de dados e cálculos devem ser delegadas a máquina.

Recentemente muitas publicações tem lançado algumas idéias e recomendações que procuram facilitar o projeto de sistemas interativos |17|. Estes modelos na sua maioria, foram desenvolvidos isoladamente e dificilmente conseguem atingir completamente seus objetivos. Como exemplos significativos desta

tentativa de modelagem dos sistemas interativos podem ser citados os seguintes trabalhos : |18| procura desenvolver as bases teóricas para desenvolvimento de sistemas inteligentes dotados de processos de inferência de conhecimento, para auxiliar os usuários no desenvolvimento de sua tarefa; |19| aborda princípios como o aprendizado e desempenhos humanos na realização de uma tarefa como principais fatores a serem considerados num projeto de sistemas interativos; | 4 | propõe a consideração de sistemas de malha fechada (feedback), e sistemas adaptativos tomando em consideração distúrbios no diálogo, dificuldades na execução da tarefa e no aprendizado para trazer a máquina o mais próximo possível da realidade do usuário; |20| procura modelar a entrada e saída de um processo interativo por meio de linguagens, considerando a simetria da entrada e saída sob os aspectos léxico,sintático e semântico.

A metodologia da interação homem-máquina foi objeto de recente 'workshop' realizado pelo IFIP em Seillac(França) em Maio de 1979, e um dos principais temas constituiu-se na análise dos modelos propostos pela literatura, procurando definir um modelo mais amplo e geral que representasse efetivamente o processo interativo homem-máquina.

Baseados nos trabalhos anteriormente citados alguns modelos foram apresentados e discutidos. Desta discussão surgiu um modelo para interação capaz de orientar o projeto de sistemas interativos futuros. Para melhor compreensão do modelo obtido segue-se uma descrição mais detalhada das idéias apresentadas.

Em |19| define-se como base para os sistemas interativos a existência de:

- uma base para representação do conhecimento;
- um conjunto de procedimentos capazes de inferir conhecimentos e utilizar as informações armazenadas em ambas bases.

A justificativa para tal modelo advém do fato de que, durante o diálogo com a máquina, o homem espera que a má

quina responda corretamente suas questões ou então tome ações em resposta a suas requisições. Para que respostas ou ações corretas sejam fornecidas ou tomadas uma descrição detalhada faz-se necessária, tornando o diálogo, na maioria das vezes cansativo e improdutivo. Porém, se homem e máquina possuírem um conhecimento comum sobre o objeto do diálogo,uma grande quantidade de detalhes pode ser inferida, tornando o diálogo sensivelmente mais simples. Portanto o computador deve, se o objetivo é facilitar o processo de diálogo, armazenar conhecimento suficiente sobre sistema, levando-nos à conclusão de que a representação, o armazenamento, a busca, a atualização e a utilização do conhecimento são técnicas básicas nos sistemas envolvendo interações homem-máquina.

O modo de representação de conhecimentos varia, em geral, de ser humano para ser humano e, naturalmente é diferente para o homem e para a máquina. Para que o conhecimento possa ser usado, cada entidade inteligente (homem ou máquina) deve poder converter o seu conhecimento para uma outra forma, sem que se perca o seu significado,tornando obrigatória a existência de uma representação externa capaz de ser compreendida por ambas as entidades (linguagens, padrões gráficos ...). Após a transformação do conhecimento para uma forma externa o receptor transforma essa informação para a sua forma de representação de conhecimento (transformação essa que consiste de 1) transformação sintática, reconhecimento de padrões, etc; 2) transformações semânticas tais como inferencia dedutiva/indutiva; 3) solução de problemas e armazena-a em suas bases de dados. Além da capacidade de uso do conhecimento o sistema deve ter a capacidade de receber e armazenar novos conhecimentos, isto é, a capacidade de aprendizado.

Em |18| conclui-se ainda que, qualquer sistema,mesmo de alta qualidade e inteligência, dificilmente encontrará usuários se não satisfizer certas condições físicas e econômicas tais como tempo de resposta, custo, etc. No mesmo nível pode ser colocada a forma de representação externa que deve ser comum ao homem e a máquina. Esta representação deve ser a mais próxima possível da linguagem usada cotidianamente pelo homem,

por exemplo, a linguagem natural e representação gráfica. Deve-se ressaltar ainda que a forma de representação externa é um dos mais importantes fatores na aceitação ou não do sistema pelo usuário.

Em |19| afirma-se que maior flexibilidade no diálogo homem-máquina pode ser obtida se a execução de uma tarefa for separada do seu planejamento e esse planejamento for suportado pelo sistema interativo. Recomenda-se a subdivisão das tarefas em subtarefas e tarefas elementares, e o uso de ações alternativas definidas como 'modo direto' e 'excursão'.

No início da interação homem-máquina o usuário possui duas alternativas :

- ele pode submeter um comando, se ele já conhece a associação tarefa-comando;
- ele pode requisitar informação para executar a ação necessária para completar sua tarefa.

Se na submissão de um comando o usuário toma o caminho direto, sua tarefa é executada imediatamente. Se ao contrário a opção, 'excursão' é tomada ele é informado sobre as tarefas e ações que o sistema está apto a suportar permitindo ao usuário refinar seu conhecimento sobre o sistema antes de fazer a associação comando tarefa (planejamento 'on-line'). Tal modelo apresenta características importantes uma vez que permite a diferenciação de usuários mais experimentados daqueles menos experimentados tornando assim o sistema mais satisfatório.

Em | 4 | propõe-se um modelo interativo baseado na teoria de sistemas de controle. Neste trabalho é introduzido pela primeira vez na literatura o conceito de um 'observador controlador' como um mecanismo especial capaz de trazer homem e máquina para o mesmo contexto. O conceito de 'observador controlador' pode ser visto como um conjunto de algoritmos (implementados em software ou hardware) que, num nível superior da hierarquia do sistema, está constantemente coletando informação sobre o diálogo, o processo, o desempenho, e atua de modo adaptativo procurando sempre manter a máquina o mais próximo pos

sível da expectativa do usuário.

Deste modo um sistema interativo adaptativo deve ser capaz de controlar o diálogo e, ajustar este diálogo baseado em parâmetros relacionados com a tarefa (ex.: grau de dificuldade, situações críticas, ...) e nos parâmetros físicos e psicológicos relacionados com o usuário. Tal ajuste do diálogo pode ser manisfestado externamente por : 1) pelo aumento ou diminuição da velocidade com que o diálogo é executado (tempo de resposta da máquina), 2) inversão da iniciativa do diálogo, que pode ocorrer em situações críticas ou no caso de baixa eficiência do usuário durante o diálogo. O diálogo pode ainda ser finalizado caso o objetivo almejado não consiga ser atingido ou quando as ações tomadas pelo usuário colocam em risco a existência do sistema.

Dentro de um modelo deste tipo, em que parâmetros relacionados física e psicologicamente com o usuário são considerados, a qualidade do estímulo dado ao usuário é de extrema importância. Este estímulo na quase totalidade dos casos é dado óticamente através de 'displays', dos quais o usuário separa a informação necessária para fins de controle daquele usada para monitoramento do sistema. Deste modo, devem ser considerados conjuntamente num sistema interativo:

- os algoritmos de controle do sistema propriamente dito;
- os modelos e parâmetros para descrição física e psicológica do homem;
- os algoritmos para adaptação do diálogo homem-máquina;
- a geração de estímulos para o usuário (linguagem de comunicação ou representação externa) e os dispositivos através dos quais estes estímulos são produzidos.

Uma vez descritos os modelos acima , pode-se agora descrever o modelo descrito em |21|. Este modelo, cujo diagrama de blocos é mostrado na figura 4.2, é na realidade um compro

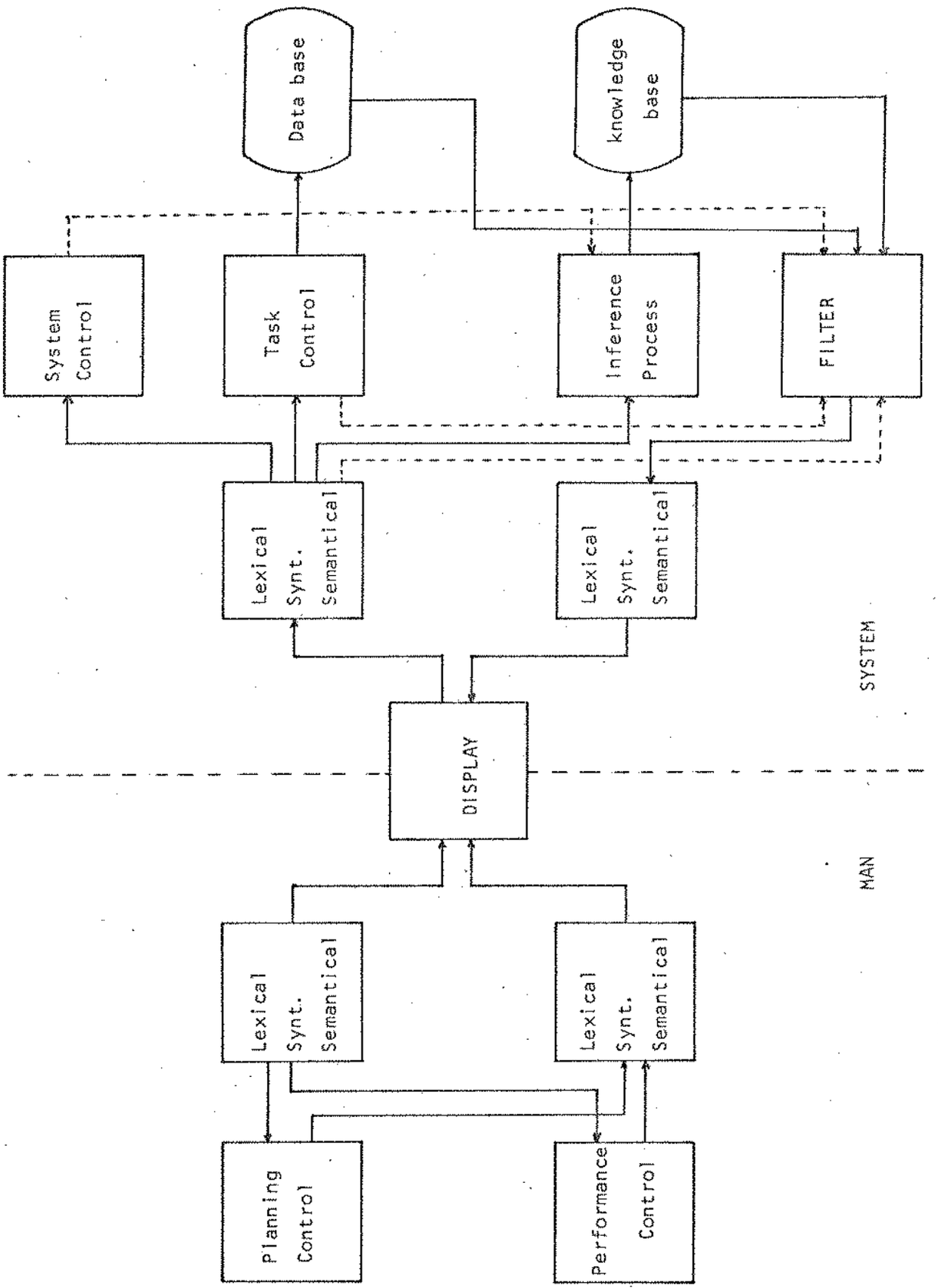


Figura 4.1 - Diagrama de blocos de um sistema interativo

misso entre os três modelos já descritos.

O modelo é composto por duas partes: homem e máquina trabalhando juntos através de uma interface (denominada 'display'). Cada um dos componentes deste sistema (homem e máquina) está por sua vez dividido em dois níveis: do lado do homem, planejamento e execução, e do lado da máquina,supervisão e execução. Do lado do homem, no nível superior, está alocado o controle do planejamento, ou seja a solução para a pergunta:'o que fazer?', enquanto no nível inferior encontra-se o mecanismo de controle de execução, ou a solução para 'como fazer?'|19|.Do lado da máquina um número maior de funções devem ser implementadas:

- O controle da tarefa propriamente dita (função básica do sistema);
- O estabelecimento de uma base de dados e de uma base de conhecimento para suporte do sistema interativo |18|;
- Algoritmos ou procedimentos capazes de realizar a inferência de conhecimentos do processo em andamento |18|;
- Um conjunto de filtros que, suportado pelo controle da tarefa, pela base de dados e conhecimentos e diretamente influenciado pelo sistema, possa gerar a representação externa necessária para o diálogo |4|;
- Um sistema de controle, definido em |4| como 'observador controlador', capaz de atuar de modo adaptativo para manter o equilíbrio da interação.

O sistema deve operar nos dois níveis para que se obtenha a manutenção do equilíbrio. Deste modo durante a execução de uma tarefa o controle permanece no nível inferior,só passando para o nível superior quando disturbios ou outros motivos impeçam a obtenção do equilíbrio desejado para a interação.Neste ponto o conjunto observador-controlador,do lado da máquina,e o contro

le do planejamento, do lado do homem, devem interagir procurando reestabelecer o equilíbrio desejado e transferir a ação ao nível inferior assim que o equilíbrio procurado for alcançado.

Dentre as muitas conclusões relacionadas com os sis teams interativos |21|, as principais são as seguintes:

I - O modelo descrito facilita a compreensão de al guns fatores importantes relacionados com os sistemas interativos:

. O modo como a interação deve ser realizada;

. Avaliação do grau de inteligência necessária em sistemas interativos;

. As diferenças entre diferentes tipos de inter faces;

. O problema da realimentação.

II - Embora os computadores se tornem cada vez mais baratos, rápidos e dotados de alta capacidade de armazenamento de informação, a interface com o meio exterior continua sendo o principal obstáculo a produção de sistemas interativos.

5.- CONSIDERAÇÕES FINAIS

Para concluir este trabalho, algumas considerações sobre os resultados obtidos, a metodologia utilizada e desenvolvimentos futuros neste campo de pesquisa são apresentados.

Embora o 'display'gráfico proposto não tenha sido implementado pode-se assegurar sua aplicabilidade nos mais diversos campos devido à facilidadede atualização da imagem. Esta rapidez na atualização advém da estrutura dinâmica da'picture memory' e da conversão em tempo real do 'display file'contido nesta memória. O sistema de módulos usados para sua construção, permite o aumento da capacidade de descrição de figuras pela multiplicação dos módulos da 'picture memory' , do controlador e do conversor em tempo real, bem como permite o uso de cores e diferentes intensidades pela multiplicação dos dispositivos de acumulação. Como consequência, as restrições quanto a complexidade das figuras podem ser eliminadas e a qualidade da comunicação pode ser melhorada pelo uso de recursos de cores, intensidades, etc.

O uso da tecnologia dos microprocessadores torna a implementação muito mais simples; a totalidade dos circuitos necessários para o controle do terminal e comunicação com os demais periféricos foram substituídos pelo 'software'. Como os programas podem ser facilmente substituídos, o sistema torna-se adaptável a qualquer situação específica,sem esforço excessivo. Além disso, o uso do microprocessador preenche as necessidades de cálculo local (terminais inteligentes), tornando o sistema menos dependente do computador central. O terminal proposto pode ser usado, sem maiores problemas em aplicações já implementadas em outros tipos de microprocessador ou minicomputador. Isto requer a substituição do microprocessador proposto pelo usado na aplicação existente e a reedição do ' software ' básico do terminal.

Sob o aspecto dos sistemas interativos o terminal satisfaz todas as características necessárias: representação externa a mais próxima possível da linguagem natural (gráficos),

qualidade dos estímulos gerados, velocidade de resposta, inteligência (capacidade local de cálculo), etc. Quanto as aplicações, ele mostra-se adequado ao controle de processos CAD, CAM, etc., pela velocidade de operação, sobreposição de imagens e mixagem de sinais pré-definidos, economizando memória, esforço computacional e melhorando os tempos de resposta.

A metodologia utilizada para o desenvolvimento do trabalho, simulação em computador, mostrou-se de grande eficiência, e constitui-se num subproduto de grande utilidade para o desenvolvimento de outros trabalhos dentro da área de sistemas digitais. Os circuitos propostos, bem como o 'software'básico, foram exaustivamente testados por simulação observando-se a viabilidade e a correção dos mesmos, fornecendo subsídios para posterior implementação. Pode-se assegurar que o trablaho desenvolvido constitui-se numa contribuição não só à carente área de desenvolvimento de sistemas gráficos mas tambem às áreas de sistemas digitais (qualquer que seja a sua aplicação), pela qualidade da ferramenta criada e das ideias expostas.

Como continuação do trabalho exposto, propõe-se:

- A modificação da estrutura de controle da 'picture memory' de modo a permitir o uso de descrição estruturada. Isso implica na reconsideração do sistema de interrupções pela 'light pen' (inclusão de pilhas de endereço) e na definição de um conjunto mais amplo e poderoso de instruções para o controlador da 'picture memory' (jumps, returns, etc.) para o tratamento de informação estruturada;
- O desenvolvimento de 'hardware' para tratamento de estruturas tri-dimensionais (linhas ou áreas escondidas). Isso implica na definição de meios de correlação da coordenada Z com os pontos ativos de cada elemento, ou na implementação de algoritmos de pré-processamento ou processamento paralelo;
- A introdução de modificações para utilização do sistema proposto na transmissão de imagens por te

lefone ou outros dispositivos.

- O desenvolvimento e implementação por 'software' ou 'hardware' da estrutura adaptativa definida neste trabalho como 'observador-controlador';
- O desenvolvimento de programas utilitários adaptados ao sistema proposto.

6.- REFERÊNCIAS

|1 | ERNST D., 'New trends in the applications of process computers', IFAC, 7th. Triennial World COngress - 1978 Plenary papers. pp. 2327 - 2336. Pergamon Press.

|2 | HOWARD A.R., 'Join micros into intelligent networks', Electronic Design - March 1975. pp. 52-57.

|3 | RINJSDORP J.E. e ROUSE W.B., 'Design of Man-Machine Interfaces in Process Control'. Digital COmputer Applications to Process COntrol, IFAC and North-Holland Co ., 1977. pp. 705-720.

|4 | TOZZI C.L., 'Man-Machine Communication in Process Control', Position Paper - Seillac II, IFIP W.G.5.2 - Workshop on Methodology of Interaction, May 1979, Proceedings and Position Papers to be published by North -Holland - 1980.

|5 | MILLER L.A. e THOMAS J.C., 'Behavioral Issues in the use of interactive systems'. Int. J.Man-Machine Studies 1977 . pp. 509-536.

|6 | HOBBS L.C., 'Terminals', 15th IEEE Computer Conference Distributed Processing - 1977 - pp. 4.6-4.17.

|7 | FOLEY J.D. e WALLACE V.L., 'The art of Natural Man-Machine Conversation'. Proceedings of the IEEE, April 1974 - pp. 462-471.

|8 | NEWMAN W.M., 'Trends in Graphic Display Design'. IEEE Transactions on Computers, Vol. C-25, nº12, December 1975 - pp. 1321-1325.

|9 | FOLEY J.D., 'Introduction to Raster Graphics Tutorial'. Fifth Annual Conference on Computer Graphics and Interactives Techniques. Siggraph/ACM. August 1978.

|10| LINDNER R. e TOZZI C.L., 'Detailed Concept for a realization of an Advanced TV Raster Display Terminal'. TH Darmstadt, FG Graphische Datenverarbeitung - GDV 77-1 1977.

|11| TOZZI C.L., 'A Special Hardware for Picture Memory Management' TH: Darmstadt, Fachgebietes Graphisch-Interactive Systeme- GRIS 78-4 - 1978.

|12| HANUSA H. e TOZZI C.L., 'Requirement for the Basic Software of a Microprocessor Controlled Terminal'.TH Darmstadt, Fachgebietes Grapisch Interactive Systeme, GRIS 78-7 - 1978

|13| Z80 - MCB Hardware User's Manual - Zilog January 1978.

|14| Z80- PIO Technical Manual - Zilog - 1978

|15| BRAVER M.A. e FRIEDMAN A.D., 'Diagnosis and Reliable Design of Digital Systems'. Computer Science Press, Inc.-1976.

|16| LINDNER R., 'SIMGA5 A Subroutine Package for Low Level Digital Hardware Simulation! TH. Darmstadt, FG Graphische Datenverarbeitung GDV 76-5 - 1976.

|17| BENNETT J.L., 'The User Interface in Interactive Systems', Annual Review of Information Science and Technology. Ed. by Cuadra vol.7 pp. 159-196. - 1972.

|18| OHSUGA S., 'Towards Intelligent Interactive Systems', IFIP W.G.5.2. Workshop on Methodology of Interaction May 1979, Proceedings and Position Papers to be published by North-Holland - 1980.

|19| DZIDA W. , HERDA S. e ITZENFELDT W.D., 'A Paradigma for Task-Oriented Man-Computer Interaction'-IFIP W.G. 5.2 Workshop on Methodology of Interaction, May 1979, Proceedings and Position Papers to be published by North Holland - 1980.

|20| FOLEY J.D., 'The Structure of Interactive Command Languages'.IFIP W.G.5.2 Workshop on Methodology of Interaction, May 1979. Proceedings and Position Paper to be published by North-Holland - 1980

|21| GUEDY R.A. and all editors, 'Final Report Seillac II IFIP W.G.5.2 Workshop on Methodology of Interaction , May 1979, Proceedings and Position Papers to be published by North-Holland - 1980.

APÊNDICE 1

Simulação de Circuitos Digitais

'SIMGA5 A subroutine Package for

Low Level Digital Hardware Simulation'

- RESUMO -

1. PRINCÍPIO DE SIMULAÇÃO

O comportamento físico dos circuitos de chaveamento pode ser caracterizado pelo fato de mudanças dos sinais de entrada resultarem mudanças dos sinais de saída após um certo tempo (delay) dependente do tipo de mudança ocorrida na saída (transição de um estado high-low ou Low- high).

O conjunto de subrotinas desenvolvidas, simula circuitos combinando circuitos mais elementares ("gates"). O modelo para estes"gates" mais elementares permite trabalhar com diferentes atrasos, para os diferentes tipos de transição nos sinais e tornam não efetivos sinais que mudam muito rapidamente ("spikes").

Os "gates" elementares possuem uma variável de saída denominada GATAUS, que é definida por uma função lógica denominada GATSPE. Esta variável de saída pode ser um estado estável ou instável e sua condição é definida pela variável AUSTIM. Estado estável significa, que o "gate" elementar permanece no seu estado atual até que ocorra uma mudança na sua entrada. Estado instável significa que a saída do "gate" mudará seu valor após o tempo de atraso, a não ser que uma nova mudança na entrada volte a saída do "gate" ao seu estado original.

Os "gates" elementares são do tipo "master-slave" e portanto permitem realimentação. A variável de saída GATAUS é o "slave" e é controlada pelo "master" que é representado pela variável AUSTIM, que assume valores negativos (-1) para estados estáveis e possue o valor (positivo) do tempo de atraso para as situações instáveis.

O tempo de simulação, definido pela variável TIME, é incrementado do valor mínimo de atrazo, definido em AUSTIM, ocorrendo então a atualização das variáveis GATAUS e AUSTIM.

2. ESTRUTURA DO SIMULADOR

Para que se possa ter uma idéia mais geral do simulador, sua estrutura é apresentada na figura 1.

A subrotina SIMGA5 recebe os comandos do usuário do sistema e chama a subrotina COMND5, por sua vez, chama a subrotina NORMC5, que reduz o comando recebido a somente tres caracteres, e a subrotina ISTRI5 que transforma em um número inteiro a sequência de caracteres recebidos após o comando. A subrotina COMND5 -

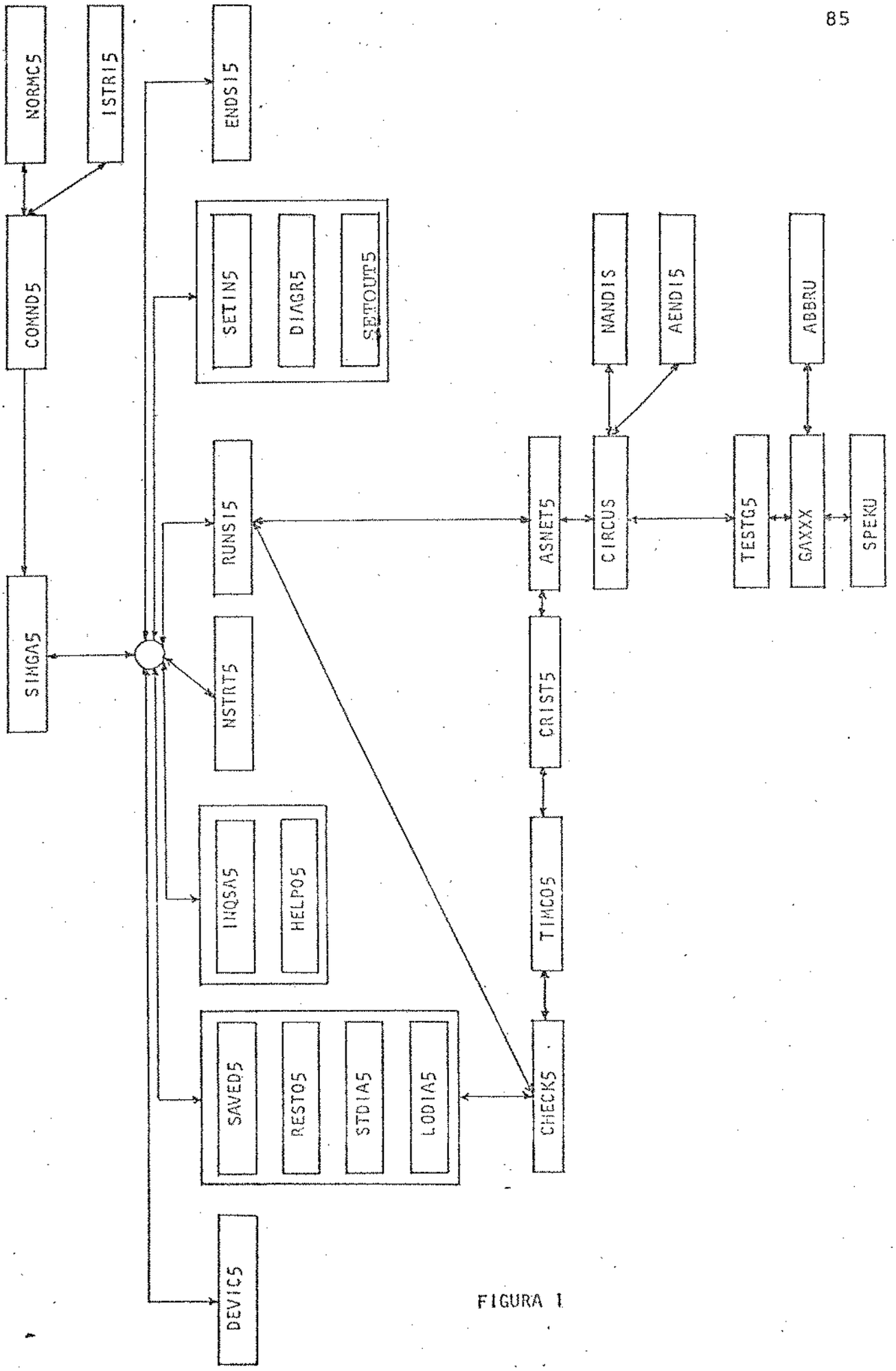

NORMCS
ISTRIS
ENDSIS
COMNDS
SETINS
DIAGRS
SETOUTS
NANDIS
AENDIS
ABBRU
RUNSIS
ASNETS
CIRCUS
TESTGS
GAXXX
SPEKU
SIMGAS
NSTRTS
CRISTS
INQSAS
HELPOS
TIMCOS
SAVEQS
RESTOS
STDIAS
LODIAS
CHECKS
DEVICS

FIGURA 1

retorna um conjunto de parâmetros que permite a SIMGA5 chamar uma das treze - funções implementadas no simulador e descritas a seguir:

NSTRT5 - (newstart) reinicializa completamente o conjunto de dados usado na simulação;

SETIN5 - (sets an input) define o valor da entrada de um "gate" (high ou low);

DIAGR5 - (diagram) define uma coluna no diagrama usado para documentação;

SETOU5 - (sets an output) define o valor para a saída de um "gate" (high ou low);

RUNSI5 - (runs the simulation) desenvolve a simulação e determina o tempo de duração da mesma.

INQSA5 - (inquires subaddress) utilizado no caso de "gates" estruturados, para determinação do subgate;

HELPO5 - (Helps operator) ajuda o operador, imprimindo a lista de todos os comandos permitidos pelo simulador;

SAVED5 - (saves data) salva o conjunto de dados de simulação, escrevendo-os, em disco, no arquivo definido;

RESTO5 - (restores) recarrega o conjunto de dados de simulação, a partir de um arquivo definido, e residente em disco.

STDIA5 - (stores diagram) guarda o conjunto de dados que definem o diagrama usado para documentação, escrevendo-os em disco, no arquivo definido.

LODIA5 - (loads diagram) carrega o conjunto de dados que definem o diagrama usado para documentação, a partir do arquivo definido.

DEVIC5 - (device) permite que a documentação seja obtida num arquivo criado em disco, ou no próprio terminal usado para o diálogo operador - simulador.

ENDIS5 - (end of simulation) organiza o fim da simulação.

A subrotina RUNS15 é a entrada para o simulador propriamente dito. A partir desta surotina é feita a chamada da subrotina ASNET5, que permite através das subrotinas CIRCU5, HANDI5, AENDI5 e GAXXX realizar a conexão dos "gates" elementares para formação dos circuitos, atualização da documentação, execução de funções definidas por cada "gate",cálculo do próximo valor da variável que define o tempo e através da subrotina SPEKU a atualização das saídas dos "gates".

3. O SIMULADOR PROPRIAMENTE DITO

Para que se possa introduzir com maior facilidade o funcionamento do simulador propriamente dito, um "gate" elementar será usado e a subrotina para simulação deste "gate" discutida.

O "gate" selecionado foi o "Quad 2 Input Nand - SN7400 , cuja subrotina é apresentada no apêndice 1. Uma rápida leitura desta subrotina permite tornar-se familiar com as mais importantes variáveis, que são encontradas numa área comum. Uma breve descrição das mais importantes destas variáveis é dada a seguir:

GATAUS: variável de saída dos "gates" elementares ;

GATSPE: variável onde é armazenado temporariamente o valor da variável de saída;

AUSTIM: variável onde é armazenado o estado da variável a ela associada (estado estável ou instável):

GATPUN: variável de controle da simulação;

TIME : valor atual do tempo de simulação;

AUSZAL: número de "gates" elementares que formam um "gate" mais complexo;

NR : número do "gate" sendo processado;

GATEIN: variável de entrada do"gate".

Para diminuir o tempo de processamento, toda informação sobre o estado de todos

"gates" é armazenada numa área comum. Por esta razão a quantidade de "gates" e o número de"gates" elementares num "gate" é limitado. (250 "gates" com 20 "gates" elementares e 20 entradas cada é um valor razoável).

Voltando a subrotina GA828, existem dois modos de operação em qualquer subrotina que define um "gate" básico. Dependendo da variável GATFUN, ou são construídas - listas com dados sobre o gate ou suas funções são executadas.

Na construção das listas para o "gate" em questão (que não tem subgates) somente o número de "gates elementares é fornecido ao vetor AUSZAL. O "chip" SN7400 somente necessita 4 "gates" elementares e então 16 "gates" elementares são desperdiçados. Isto é característico para "chips" muito simples com ANDs, NANDs, ORs, Buffers, Inverters. Circuitos mais complexos possuem um fator de utilização me- / lhor.

No "loop" dedinido por 100, é definida a função lógica do "gate":

GATSPE = .NOT. (GATEIN (1,NR).AND.GATEIN (2.NR))

GATSPE terá como resultado o valor que será assumido pela variável de saída - GATAUS, após decorrido o tempo de transição. A seguir é feita a chamada da subrotina SPEKU que primeiro testa o estado antigo do "gate", então calcula o estado futuro e atualiza a variável AUSTIM, se necessário. Este procedimento é mostrado no diagrama de blocos desta subrotina, na figura 2. Na saída da subrotina SPEKU as variáveis AUSTIM (NI, NR) e GATAUS (NI, NR) contem toda a informação sobre o "gate"elementar.

4. "GATES" ESTRUTURADOS

Um dos mais importantes atributos do simulador desenvolvido é a possibilidade de trabalhar com "gates" estruturados.

O uso de "gates" estruturados permite a combinação de vários "gates" de modo a formar um novo "gate". O usuário do simulador vê então, um só gate e somente tem necessidade de conectar e controlar um só gate. O uso de "gates" estruturados - permite a implementação dos mais complexos "chips" ou circuitos construídos a - partir destes "chips", como o caso de microprocessadores.

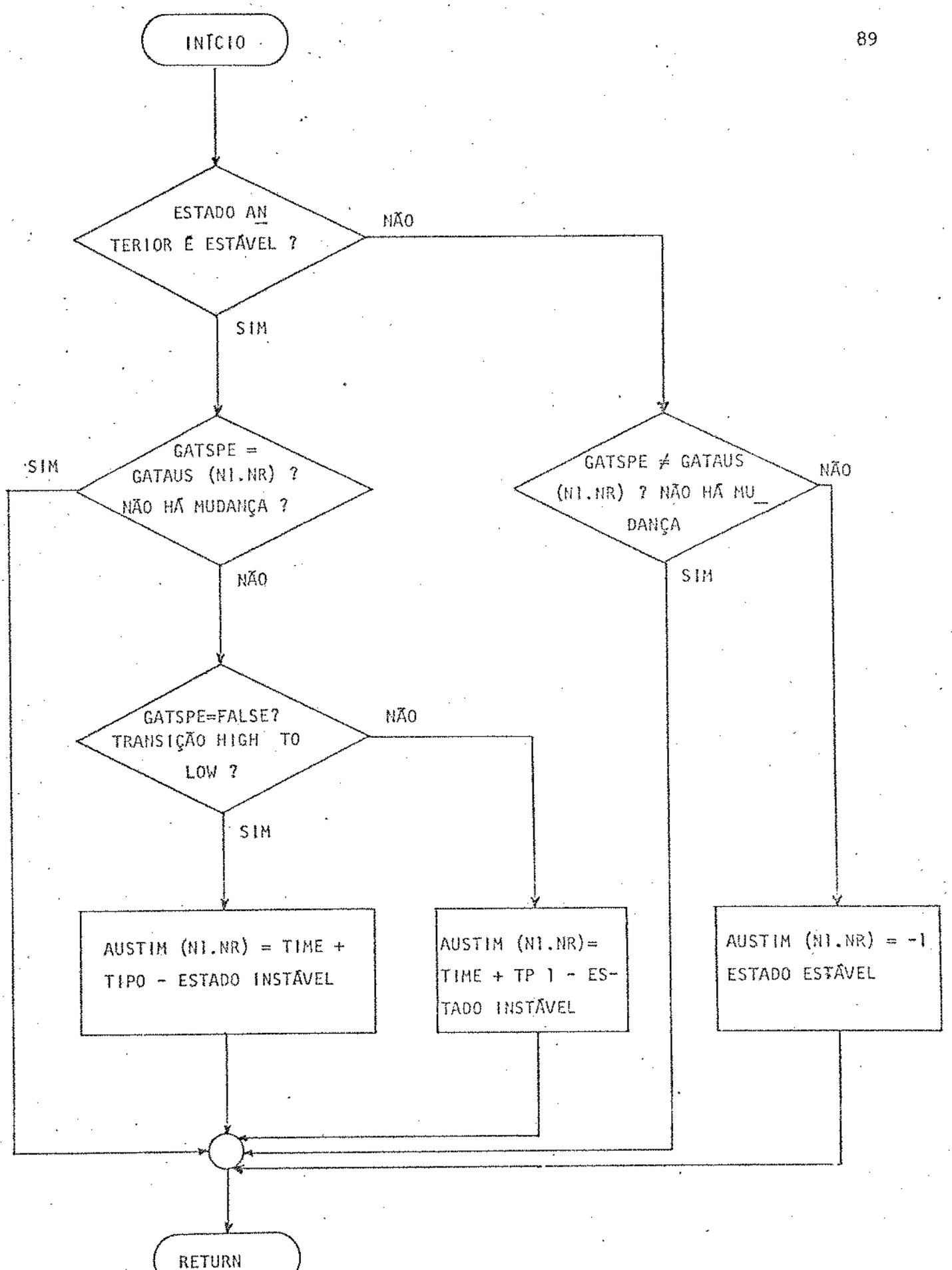

INÍCIO
ESTADO ANTERIOR É ESTÁVEL ?
NÃO
SIM
GATSPE = GATAUS (NI.NR) ? NÃO HÁ MUDANÇA ?
SIM
NÃO
GATSPE ≠ GATAUS (NI.NR) ? NÃO HÁ MUDANÇA
NÃO
SIM
GATSPE=FALSE? TRANSIÇÃO HIGH TO LOW ?
NÃO
SIM
AUSTIM (NI.NR) = TIME + TIPO - ESTADO INSTÁVEL
AUSTIM (NI.NR)= TIME + TP 1 - ESTADO INSTÁVEL
AUSTIM (NI.NR) = -1 ESTADO ESTÁVEL
RETURN

FIGURA 2

5. UM EXEMPLO DO USO DO SIMULADOR

Considere a simulação de um circuito muito simples, mostrado na figura 3., um circuito divisor por 3, construído a partir de flip-flops tipo D sensível a borda e gates nand. Dois chips serão usados, "dual D Type positive edge triggerede flip-flop with preset and clear (SN74574), numerado no circuito com 1, e - "quad 2 input nand (SN7400), numerado no circuito com 2.

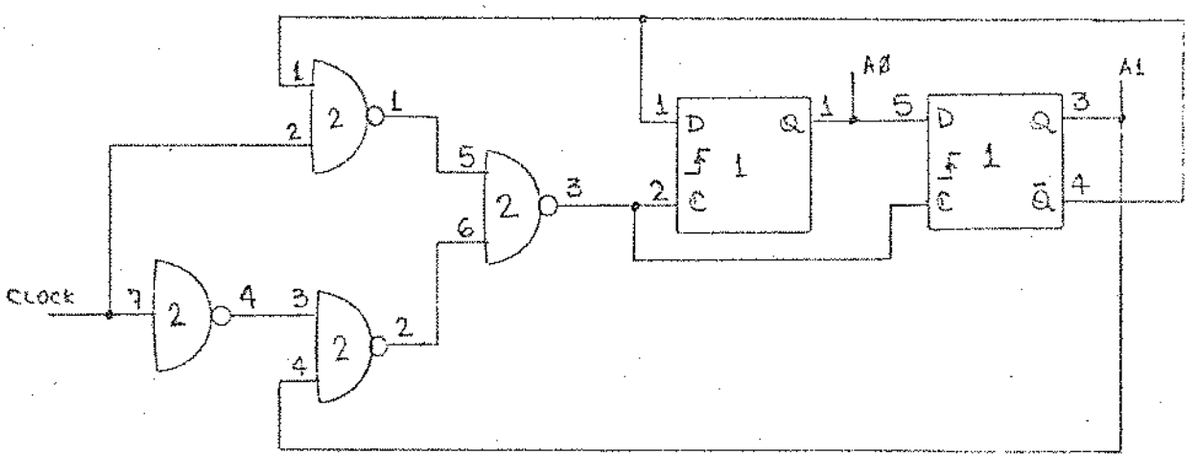
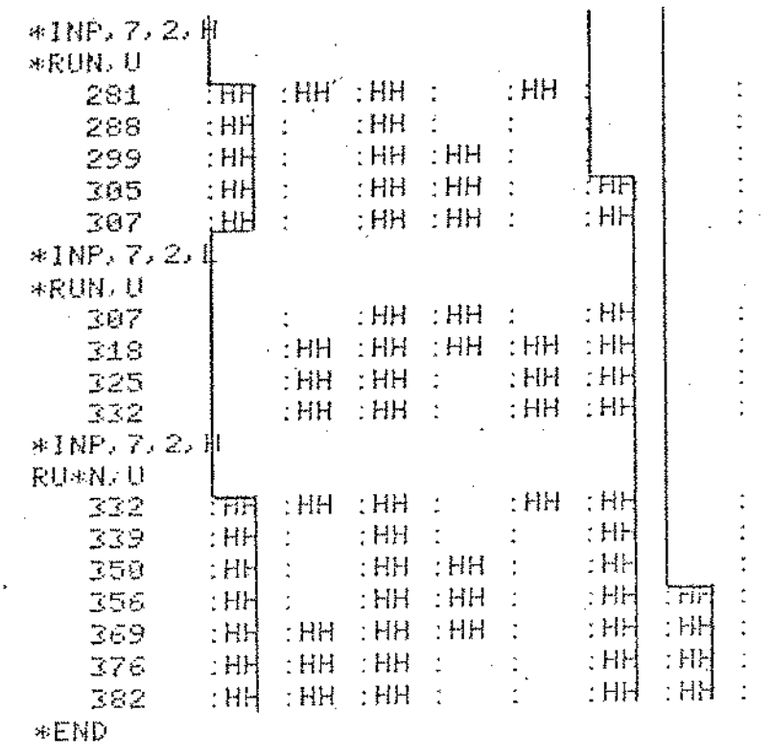


FIGURA 3

O diagrama de tempo apresentado a seguir foi obtido através do simulador e mostra na coluna um a variável de entrada (clock), nas colunas 2, 3,4,5, as saídas do "Chip" 1 (SN7400), na figura numerados respectivamente 1, 2, 3 e 4 e nas colunas 7 e 8 as saídas do "chip" 2 (SN74574), numeradas respectivamente com 1 e 3 na figura e que são os pontos de saída do circuito, obtendo-se em 1 e 3 os mesmos sinais, deslocados de um pulso de entrada; conforme mostrado no diagrama que segue:

```
 RUN,U
RUN UNTIL LAST CHANGE HAPPENED.
                1   2   3   4   5   6   7
                1   0   0   0   0   0   0
                7   1   2   3   4   1   3
TIME(NS)        2   2   2   2   2   1   1
        0        :HH :HH :   :HH
       51        :HH :HH :   :HH
        0        :HH :HH :   :HH
       51        :HH :HH :   :HH
*DEV,DSK
*INP,7,2,H
*RUN,U
       51    :HH :HH :HH :   :HH
       58    :HH :   :HH :   :
       69    :HH :   :HH :HH :
       75    :HH :   :HH :HH :   :HH
       77    :HH :   :HH :HH :   :HH
*INP,7,2,L
*RUN,U
       77        :   :HH :HH :   :HH
       88        :HH :HH :HH :HH :HH
       95        :HH :HH :   :HH :HH
      102        :HH :HH :   :HH :HH
*INP,7,2,H
*RUN,U
      102    :HH :HH :HH :   :HH :HH
      109    :HH :   :HH :   :   :HH
      120    :HH :   :HH :HH :   :HH
      126    :HH :   :HH :HH :   :HH :HH
      139    :HH :HH :HH :HH :   :HH :HH
      146    :HH :HH :HH :   :   :HH :HH
      152    :HH :HH :HH :   :   :HH :HH
*INP,7,2,L
*RUN,U
      152        :HH :HH :   :   :HH :HH
      163        :HH :HH :   :HH :HH :HH
      170        :HH :   :   :HH :HH :HH
      181        :HH :   :HH :HH :HH :HH
      188        :HH :   :HH :HH     :HH
      189        :HH :   :HH :HH     :HH
*INP,7,2,H
*RUN,U
      189    :HH :HH :   :HH :HH     :HH
      196    :HH :HH :   :HH :       :HH
      207    :HH :HH :HH :HH :       :HH
      214    :HH :HH :HH :   :       :HH
      220    :HH :HH :HH :   :       :HH
*INP,7,2,L
*RUN,U
      220        :HH :HH :   :       :HH
      231        :HH :HH :   :HH     :HH
      238        :HH :   :   :HH     :HH
      249        :HH :   :HH :HH     :HH
      256        :HH :   :HH :HH
      267        :HH :HH :HH :HH
      274        :HH :HH :   :HH
      281        :HH :HH :   :HH
```



6. NOTAS

Pelo exposto nos ítens anteriores, o simulador mostrou-se uma valiosa ferramenta para o desenvolvimento e teste de circuitos digitais de qualquer complexidade.

Além disso, o mesmo tem sido de grande utilidade no ensino, permitindo a demonstração e análise de efeitos tais como "spikes", atrasos, etc. com grande facilidade e sem a necessidade de utilização de equipamentos de alta sensibilidade.

DOCUMENTAÇÃO COMPLETA DO CIRCUITO DA FIGURA 3

```
^C

.RUN SIMGA5

*HELP
HELP OPERATOR TEXT LISTING.
===========================

INP,NO OF INPUT,NO OF GATE,DESIRED VALUE
    GIVES VALUE TO INPUT OF GATE
DIA,TYPE OF SIGNAL,NO OF SIGNAL,NO OF GATE,NO OF COLUMN
    DEFINES DIAGRAM COLUMN
RUN,TIME (U FOR UNDIFINED)
    RUNS SIMULATION
SAV,FILENAME
    SAVES SIMULATION DATA TO FILE
RES,FILENAME
    RESTORES SIMULATION DATA FROM FILE
STD,FILENAME
    STORES DIAGRAM DEFINITION TO FILE
LOD,FILENAME
    LOADS DIAGRAM DEFINITION FROM FILE
NEW
    NEWSTARTS SIMULATION
END
    ENDS SIMULATION
DEV,NAME OF DEVICE
    ASSIGNS DEVICE OR FILE TO LOGICAL UNIT NUMBER
SOP,NO OF OUTPUT,NO OF GATE,DESIRED VALUE
    GIVES VALUE TO OUTPUT OF GATE
ISA,NO OF GATE
    INQUIRES SUBADDRESS OF GATE



INÍCIO DA SIMULAÇÃO
```

```
 TY DOCUM DSM
DIA,I,7,2,1
DIA,O,1,2,2
DIA,O,2,2,3
DIA,O,3,2,4
DIA,O,4,2,5
DIA,O,1,1,6
DIA,O,3,1,7
```

definição das saídas e entradas desejadas no diagrama de tempo.

```
RUN,U
 RUN UNTIL LAST CHANGE HAPPENED.
              1   2   3   4   5   6   7
              I   0   0   0   0   0   0
              7   1   2   3   4   1   3
 TIME(NS)     2   2   2   2   2   1   1
        0 :   :   :   :   :   :   :   :
        0   :HH :   :   :   :   :   :   :
        4   :HH :   :   :   :   :   :   :
        4   :HH :   :   :   :   :HH :HH :
        2   :HH :   :   :   :   :HH :HH :
        6   :HH :   :   :   :   :   :   :
        5   :HH :   :   :   :   :   :   :
       11   :HH :   :HH :HH :   :   :   :
        6   :HH :   :HH :HH :   :   :   :
       17   :HH :   :HH :HH :   :HH :   :
        2   :HH :   :HH :HH :   :HH :   :
       19   :HH :   :HH :HH :   :HH :   :
INP,7,2,L
RUN,U
 RUN UNTIL LAST CHANGE HAPPENED.
              1   2   3   4   5   6   7
              I   0   0   0   0   0   0
              7   1   2   3   4   1   3
 TIME(NS)     2   2   2   2   2   1   1
        0 :HH :   :HH :HH :   :HH :   :
       19 :   :   :HH :HH :   :HH :   :
       11 :   :   :HH :HH :   :HH :   :
       30 :   :HH :HH :HH :HH :HH :   :
        7 :   :HH :HH :HH :HH :HH :   :
       37 :   :HH :HH :   :HH :HH :   :
        7 :   :HH :HH :   :HH :HH :   :
       44 :   :HH :HH :   :HH :HH :   :
SOP,1,1,L                                          Reset Flip-Flop
 CHANGE VALUE OF OUTPUT( 1,  1) FROM
  AUSTIM=     -1 ,GATAUS=1 TO                      A 0
  AUSTIM=     -1 ,GATAUS=0
RUN,U
 RUN UNTIL LAST CHANGE HAPPENED.
              1   2   3   4   5   6   7
              I   0   0   0   0   0   0
              7   1   2   3   4   1   3
 TIME(NS)     2   2   2   2   2   1   1
        0 :   :HH :HH :   :HH :HH :   :
       44 :   :HH :HH :   :HH :   :   :
        4 :   :HH :HH :   :HH :   :   :
       48 :   :HH :HH :   :HH :HH :   :
        1 :   :HH :HH :   :HH :HH :   :
       49 :   :HH :HH :   :HH :   :   :
        2 :   :HH :HH :   :HH :   :   :
       51 :   :HH :HH :   :HH :   :   :          Clock - High
INP,7,2,H
RUN,U
 RUN UNTIL LAST CHANGE HAPPENED.
              1   2   3   4   5   6   7
              I   0   0   0   0   0   0
              7   1   2   3   4   1   3
 TIME(NS)     2   2   2   2   2   1   1
        0 :   :HH :HH :   :HH :   :   :
       51 :HH :HH :HH :   :HH :   :   :
```

```
        7 :HH :HH :HH :    :HH :    :    :
       58    :HH :    :HH :    :    :    :
         11 :HH :    :HH :    :    :    :
       69    :HH :    :HH :HH :    :    :
          6 :HH :    :HH :HH :    :    :
       75    :HH :    :HH :HH :    :HH :    :
          2 :HH :    :HH :HH :    :HH :    :
       77    :HH :    :HH :HH :    :HH :    :
```

Clock low

```
INP,7,2,L
RUN,U
 RUN UNTIL LAST CHANGE HAPPENED.
                1   2   3   4   5   6   7
                1   0   0   0   0   0   0
                7   1   2   3   4   1   3
 TIME(NS)       2   2   2   2   2   1   1
          0 :HH :    :HH :HH :    :HH :    :
       77   :    :    :HH :HH :    :HH :    :
         11 :    :    :HH :HH :    :HH :    :
       88   :    :HH :HH :HH :HH :HH :    :
          7 :    :HH :HH :HH :HH :HH :    :
       95   :    :HH :HH :    :HH :HH :    :
          7 :    :HH :HH :    :HH :HH :    :
      102   :    :HH :HH :    :HH :HH :    :
```

Clock high

```
INP,7,2,H
RUN,U
 RUN UNTIL LAST CHANGE HAPPENED.
                1   2   3   4   5   6   7
                1   0   0   0   0   0   0
                7   1   2   3   4   1   3
 TIME(NS)       2   2   2   2   2   1   1
          0 :    :HH :HH :    :HH :HH :    :
      102   :HH :HH :HH :    :HH :HH :    :
          7 :HH :HH :HH :    :HH :HH :    :
      109   :HH :    :HH :    :    :HH :    :
         11 :HH :    :HH :    :    :HH :    :
      120   :HH :    :HH :HH :    :HH :    :
          6 :HH :    :HH :HH :    :HH :    :
      126   :HH :    :HH :HH :    :HH :HH :
         13 :HH :    :HH :HH :    :HH :HH :
      139   :HH :HH :HH :HH :    :HH :HH :
          7 :HH :HH :HH :HH :    :HH :HH :
      146   :HH :HH :HH :    :    :HH :HH :
          6 :HH :HH :HH :    :    :HH :HH :
      152   :HH :HH :HH :    :    :HH :HH :
```

Clock low

```
INP,7,2,L
RUN,U
 RUN UNTIL LAST CHANGE HAPPENED.
                1   2   3   4   5   6   7
                1   0   0   0   0   0   0
                7   1   2   3   4   1   3
 TIME(NS)       2   2   2   2   2   1   1
          0 :HH :HH :HH :    :    :HH :HH :
      152   :    :HH :HH :    :    :HH :HH :
         11 :    :HH :HH :    :    :HH :HH :
      163   :    :HH :HH :    :HH :HH :HH :
          7 :    :HH :HH :    :HH :HH :HH :
      170   :    :HH :    :    :HH :HH :HH :
         11 :    :HH :    :    :HH :HH :HH :
      181   :    :HH :    :HH :HH :HH :HH :
          7 :    :HH :    :HH :HH :HH :HH :
```

```
    188    :    :HH :    :HH :HH :    :HH :
        1  :    :HH :    :HH :HH :    :HH :
    189    :    :HH :    :HH :HH :    :HH :
INP,7,2,H                                          Clock high
RUN,U
 RUN UNTIL LAST CHANGE HAPPENED.
              1   2   3   4   5   6   7
              1   0   0   0   0   0   0
              7   1   2   3   4   1   3
 TIME(NS)     2   2   2   2   2   1   1
        0  :    :HH :    :HH :HH :    :HH :
    189    :HH :HH :    :HH :HH :    :HH :
        7  :HH :HH :    :HH :HH :    :HH :
    196    :HH :HH :    :HH :    :    :HH :
       11  :HH :HH :    :HH :    :    :HH :
    207    :HH :HH :HH :HH :    :    :HH :
        7  :HH :HH :HH :HH :    :    :HH :
    214    :HH :HH :HH :    :    :    :HH :
        6  :HH :HH :HH :    :    :    :HH :
    220    :HH :HH :HH :    :    :    :HH :
INP,7,2,L                                          Clock low
RUN,U
 RUN UNTIL LAST CHANGE HAPPENED.
              1   2   3   4   5   6   7
              1   0   0   0   0   0   0
              7   1   2   3   4   1   3
 TIME(NS)     2   2   2   2   2   1   1
        0  :HH :HH :HH :    :    :    :HH :
    220    :    :HH :HH :    :    :    :HH :
       11  :    :HH :HH :    :    :    :HH :
    231    :    :HH :HH :    :HH :    :HH :
        7  :    :HH :HH :    :HH :    :HH :
    238    :    :HH :    :    :HH :    :HH :
       11  :    :HH :    :    :HH :    :HH :
    249    :    :HH :    :HH :HH :    :HH :
        7  :    :HH :    :HH :HH :    :HH :
    256    :    :HH :    :HH :HH :    :    :
       11  :    :HH :    :HH :HH :    :    :
    267    :    :HH :HH :HH :HH :    :    :
        7  :    :HH :HH :HH :HH :    :    :
    274    :    :HH :HH :    :HH :    :    :
        7  :    :HH :HH :    :HH :    :    :
    281    :    :HH :HH :    :HH :    :    :
INP,7,2,H                                          Clock high
RUN,U
 RUN UNTIL LAST CHANGE HAPPENED.
              1   2   3   4   5   6   7
              1   0   0   0   0   0   0
              7   1   2   3   4   1   3
 TIME(NS)     2   2   2   2   2   1   1
        0  :    :HH :HH :    :HH :    :    :
    281    :HH :HH :HH :    :HH :    :    :
        7  :HH :HH :HH :    :HH :    :    :
    288    :HH :    :HH :    :    :    :    :
       11  :HH :    :HH :    :    :    :    :
    299    :HH :    :HH :HH :    :    :    :
        6  :HH :    :HH :HH :    :    :    :
    305    :HH :    :HH :HH :    :HH :    :
        2  :HH :    :HH :HH :    :HH :    :
    307    :HH :    :HH :HH :    :HH :    :
```

```
INP,7,2,L                                                    Clock low
RUN,U
 RUN UNTIL LAST CHANGE HAPPENED.
              1   2   3   4   5   6   7
              I   0   0   0   0   0   0
              7   1   2   3   4   1   3
 TIME(NS)     2   2   2   2   2   1   1
         0 :HH :    :HH :HH :    :HH :   :
       307 :   :    :HH :HH :    :HH :   :
        11 :   :    :HH :HH :    :HH :   :
       318 :   :   :HH :HH :HH :HH :HH :   :
         7 :   :   :HH :HH :HH :HH :HH :   :
       325 :   :   :HH :HH :    :HH :HH :   :
         7 :   :   :HH :HH :    :HH :HH :   :
       332 :   :   :HH :HH :    :HH :HH :   :
                                                             Clock high
INP,7,2,H
RUN,10                                                       Simulação 10
      10 NANOSECONDS RUN.                                    nanosegundos
              1   2   3   4   5   6   7
              I   0   0   0   0   0   0
              7   1   2   3   4   1   3
 TIME(NS)     2   2   2   2   2   1   1
         0 :    :HH :HH :    :HH :HH :   :
       332 :  :HH :HH :HH :    :HH :HH :   :
         7 :  :HH :HH :HH :    :HH :HH :   :
       339 :  :HH :    :HH :    :   :HH :   :
         3 :  :HH :    :HH :    :   :HH :   :
       342 :  :HH :    :HH :    :   :HH :   :
RUN,20
      20 NANOSECONDS RUN.                                    Simulação 20
              1   2   3   4   5   6   7                      nanosegundos
              I   0   0   0   0   0   0
              7   1   2   3   4   1   3
 TIME(NS)     2   2   2   2   2   1   1
         0 :HH :    :HH :    :   :HH :   :
       342 :  :HH :    :HH :    :   :HH :   :
         8 :  :HH :    :HH :    :   :HH :   :
       350 :  :HH :    :HH :HH :    :HH :   :
         6 :  :HH :    :HH :HH :    :HH :   :
       356 :  :HH :    :HH :HH :    :HH :HH :
         6 :  :HH :    :HH :HH :    :HH :HH :
       362 :  :HH :    :HH :HH :    :HH :HH :
                                                             Clock low
INP,7,2,L
RUN,U
 RUN UNTIL LAST CHANGE HAPPENED.
              1   2   3   4   5   6   7
              I   0   0   0   0   0   0
              7   1   2   3   4   1   3
 TIME(NS)     2   2   2   2   2   1   1
         0 :HH :    :HH :HH :    :HH :HH :
       362 :  :   :HH :HH :    :HH :HH :
         7 :  :   :HH :HH :    :HH :HH :
       369 :  :HH :HH :HH :    :HH :HH :
         4 :  :HH :HH :HH :    :HH :HH :
       373 :  :HH :HH :HH :HH :HH :HH :
         3 :  :HH :HH :HH :HH :HH :HH :
       376 :  :HH :HH :    :HH :HH :HH :
         4 :  :HH :HH :    :HH :HH :HH :
       380 :  :HH :    :    :HH :HH :HH :
        11 :  :HH :    :    :HH :HH :HH :
```

```
      391    :    :HH :    :HH :HH :HH :HH :
           7 :    :HH :    :HH :HH :HH :HH :
      398    :    :HH :    :HH :HH :    :HH :
           1 :    :HH :    :HH :HH :    :HH :
      399    :    :HH :    :HH :HH :    :HH :
INP,7,2,H
RUN,3
      3 NANOSECONDS RUN.
                1   2   3   4   5   6   7
                I   0   0   0   0   0   0
                7   1   2   3   4   1   3
 TIME(NS)       2   2   2   2   2   1   1
         0 :    :HH :    :HH :HH :    :HH :
      399  :HH :HH :    :HH :HH :    :HH :
         3 :HH :HH :    :HH :HH :    :HH :
      402  :HH :HH :    :HH :HH :    :HH :
INP,7,2,L                                          "spike" 3 nanose-
RUN,U                                               gundos não efetivo
 RUN UNTIL LAST CHANGE HAPPENED.
                1   2   3   4   5   6   7
                I   0   0   0   0   0   0
                7   1   2   3   4   1   3
 TIME(NS)       2   2   2   2   2   1   1
         0 :HH :HH :    :HH :HH :    :HH :
      402  :    :HH :    :HH :HH :    :HH :
         0 :    :HH :    :HH :HH :    :HH :
      402  :    :HH :    :HH :HH :    :HH :
END
```

SUBROTINA GA828 QUAD 2-INPUT NAND SN7400

```
      SUBROUTINE GA828(NR)
C*****QUAD 2-INPUT NAND SN7400.
C*****1A=GATEIN(1,NR), 1B=GATEIN(2,NR), 1Y=GATAUS(1,NP).
C*****2A=GATEIN(3,NR), 2B=GATEIN(4,NR), 2Y=GATAUS(2,NR).
C*****3A=GATEIN(5,NR), 3B=GATEIN(6,NR), 3Y=GATAUS(3,NR).
C*****4A=GATEIN(7,NR), 4B=GATEIN(8,NR), 4Y=GATAUS(4,NR).
     1 INTEGER DIATIM,GATFUN,GATZAL,MAXGAT,MAXPIN,TIME,ZEIT,
     AAUSIN(20,250),AUSZAL(250),INTEDA(100),KURVEL(30),SUBADR(250),
     ANR
      LOGICAL GATSPE,GATAUS(20,250),GATEIN(20,250),
     AIMPDIA(30),LOGIDA(100)
      COMMON AUSTIN,AUSZAL,DIATIM,GATAUS,GATEIN,GATFUN,GATSPE,
     AGATZAL,IMPDIA,INTEDA,KURVEL,LOGIDA,MAXGAT,MAXTIM,SUBADR,TIME,ZEIT
      IF(GATFUN.NE.1) GOTO 2
C*****BUILD UP LISTS FOR NAND GATE.
      AUSZAL(NR)=4
      GOTO 10000
C*****EXECUTE MAIN GATE FUNCTIONS.
    2 DO 100 I=1,4
      K=2*(I-1)
      GATSPE=.NOT.(GATEIN(K+1,NR).AND.GATEIN(K+2,NR))
      CALL SPENU(NR,I,7,11)
  100 CONTINUE
10000 RETURN
      END
```