Paulo Cesar Berardi

Este exemplar corresponde à redação final da tese defendida por Paulo Cesar Berardi e aprovada pela comissão julgadora em 02/08/1989.

Projeto e implementação de hardware e software para uma placa gráfica de média/alta resolução com capacidade de processamento local

Tese apresentada à Faculdade de Engenharia Elétrica da Universidade Estadual de Campinas para obtenção do título de Mestre em Engenharia Elétrica (Area Automação)

Orientador: Prof. Dr.
Clésio Luiz Tozzi

Campinas - SP
1989

RESUMO

Esta dissertação descreve o projeto e implementação do protótipo de uma placa gráfica, com capacidade local de processamento (processador MC68008), resolução de 512x380 pixels, 16 cores simultâneas escolhidas dentro de um "pallet" de 4096 cores, e taxa de regeneração de 60 Hz. Mantendo-se a mesma arquitetura e com a utilização de componentes mais atuais, a resolução da placa pode ser expandida para 1024x1024 pixels e 256 cores simultâneas.

Foi desenvolvido também um núcleo gráfico, residente em EPROM, para a execução das funções de inicialização, recepção, analise e execução dos comandos. Utilizando a capacidade de processamento da placa gráfica, foram incorporados ao núcleo gráfico primitivas mais complexas como circulos, elipses, textos, bem como todas as funções de gerenciamento de segmentos, janelas (windows) e viewports. Uma das caracteristicas mais importantes do software desenvolvido é a sua transportabilidade/reusabilidade, que possibilita, pela reescrita de alguns módulos utilitários, sua implantação em ambientes gráficos diversos.

ABSTRACT

This dissertation describes the project and implemantation of a graphics board prototype, with local processing capacity (MC 68008 processor), resolution of 512x380 pixels, 16 simultaneous colors from a pallet of 4096 colors, and refresh rate of 60Hz. Using the same architecture with up-to-date components, the reolution can be expanded up to 1024x1024 pixels with 256 simultaneous colors.

A built-in graphics kernel was also developed for the purpose of board initialization and reception, analysis and processing of commands. Using the processing power of the graphics board, more complex primitives were incorporated such as circles, ellipses, text, segmentation, management functions, windows and viewports. One of the main characteristics of the software is its transportability/reusability, which allows its migration to other graphics enviroments with few changes in basic modules.

INDICE

1. INTRODUÇAO

A motivação principal para este trabalho surgiu da necessidade de se encontrar uma placa gráfica de média/alta resolução, que permitisse a realização das funções necessárias à interface Homem-Máquina de um Sistema Digital de Controle Distribuido (SDCD) em desenvolvimento no Centro Tecnológico Para Informática (CTI).

Para seleção da placa gráfica, três requisitos iniciais foram propostos: 1. capacidade local de processamento, permitindo a liberação do processador do SDCD para atividades de controle; 2. interface para o barramento VME, permitindo a compatibilização com as demais placas de controle digital e aquisição de dados já desenvolvidas; 3. disponibilidade no mercado nacional, uma vez que a tecnologia do SDCD deveria ser repassada à industria.

Uma pesquisa de mercado mostrou, quando do início do projeto, que não se dispunha de tal produto no mercado nacional. Apenas um fabricante nacional produzia terminais gráficos de alta resolução a custo que inviabilizava o projeto SDCD. Ainda hoje esta situação persiste no segmento industrial.

O resultado desta pesquisa levou à decisão de um projeto próprio, com posterior implementação da placa gráfica.

Após a fase inicial de projetó, verificou-se que o mesmo apresentava uma abrangência maior do que a inicialmente proposta

e que supunha a utilização da placa gráfica apenas para o Sistema SDCD. Além disso, dado que as placas gráficas que se dispunha no mercado eram placas gráficas de baixa/média resolução sem capacidade local de processamento e dirigidas principalmente ao mercado dos compatíveis PC-IBM, vislumbrou-se um espaço tecnológico que poderia ser preenchido pelo projeto em questão.

Assim, tomando como componentes básicos, o processador MC68008, o controlador de vídeo TMS34061 e memórias do tipo VIDEO RAM 4161, que representavam a mais recente evolução neste tipo de dispositivo, iniciou-se o desenvolvimento da placa gráfica, cujos aspectos de projeto e implementação são discutidos neste trabalho.

A apresentação deste trabalho é feita nos seguintes capítulos:

. No capítulo 2 é apresentado um resumo dos aspectos relativos à terminais de vídeo alfanuméricos, semigráficos e gráficos.

. No capítulo 3 são discutidos aspectos ligados a arquitetura de terminais de vídeo gráficos, levando-se em conta as características das memórias de vídeo com relação a problemas de tempo de acesso e conflitos.

. No capítulo 4 são discutidos alguns dos processadores/controladores de vídeo existentes, analisando-se suas principais características visando sua aplicação em terminais gráficos.

. No capítulo 5 são apresentadas as duas implementações realizadas de terminais gráficos e também são discutidos aspectos relacionados ao desempenho destes.

. No capítulo 6 é apresentado de forma resumida o software implementado visando aplicações genéricas em computação gráfica.

. No capítulo 7 são apresentados os aspectos de engenharia, as dificuldades encontradas e sugestões para novos desenvolvimentos.

2. PRINCIPIOS BASICOS DE TERMINAIS DE VIDEO

As utilização de recursos gráficos vêm obtendo a cada dia maior importância na área de computação, aumentando o número de aplicações e favorecendo o desenvolvimento de técnicas que visam tornar a interface entre o homem e a máquina mais amigável, sendo beneficiadas principalmente pelo baixo custo dos componentes e pelo desenvolvimento das tecnologias de alta integração. Isto se deve principalmente ao desenvolvimento de novas gerações de sistemas gráficos que se utilizam cada vez mais dos benefícios alcançados pelos dispositivos eletrônicos de baixo custo e alto grau de integração, como por exemplo os dispositivos de memória que são fundamentais em terminais gráficos e os circuitos controladores/processadores.

Os terminais gráficos atualmente em uso baseiam-se principalmente nas tecnologias: "storage tubes", "vector scan", e "raster scan". A tecnologia "raster scan" que combina os circuitos de TV convencionais, aliados à grande quantidade de memória necessária para armazenar a informação gráfica, se tornou absoluta na nova geração de sistemas gráficos, devido aos fatos já mencionados no parágrafo anterior.

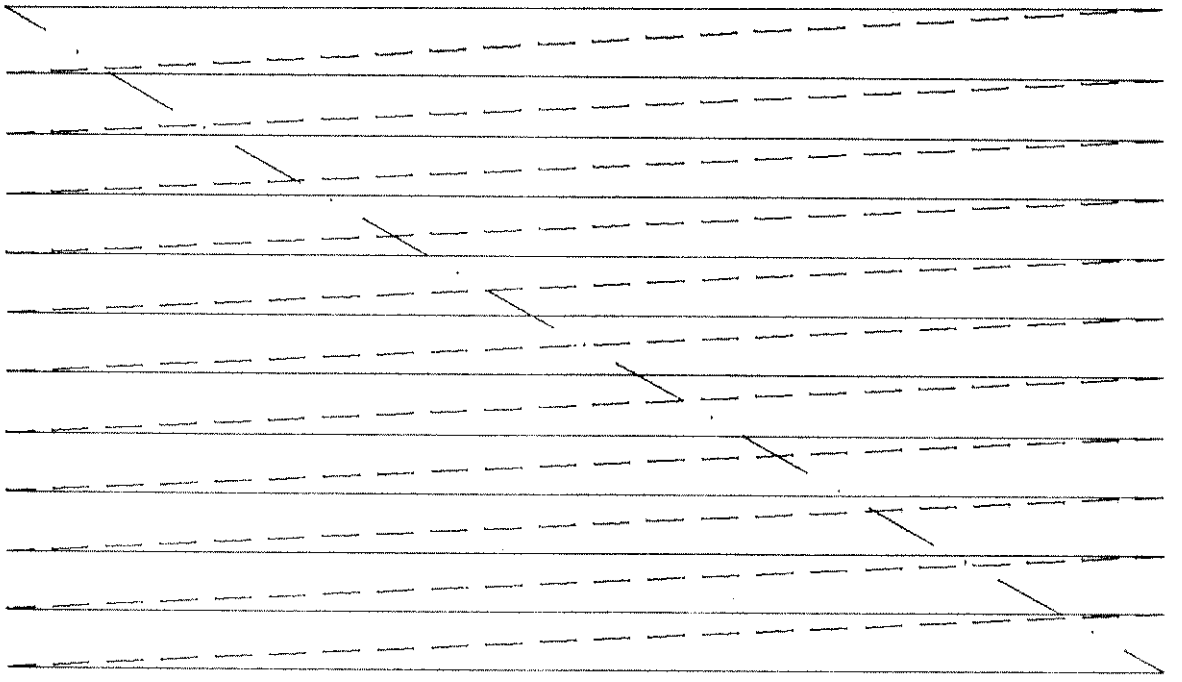
2.1. Tecnologia "Raster Scan"

A tecnologia "raster scan" baseia-se na utilização de um Tubo de Raios Catódicos (CRT), no qual um feixe de elétrons percorre uma tela recoberta de fósforo, seguindo uma trajetória

padronizada, e tendo este feixe de elétrons uma intensidade controlada durante o trajeto, de acordo com a imagem a ser formada.

O padrão com que o feixe percorre a tela é chamado "raster" e é composto de linhas horizontais paralelas. Inicialmente o feixe sai do canto superior esquerdo da tela e percorre-a no sentido horizontal até o lado direito. Neste ponto ele é desviado para o canto esquerdo, porém um pouco abaixo da linha anterior, e inicia uma nova linha horizontal para a direita. Esta sequência é repetida até que o final da tela seja atingido. Neste instante o feixe retorna ao canto superior esquerdo e a sequência se reinicia (figura 2.1).

Durante esta trajetória, quando o feixe passa pelos pontos que devem ser acesos, sua intensidade é aumentada, fazendo com que o fósforo da tela se acenda. Ao final do traçado de uma tela, tem-se a imagem completa. Um aspecto a ser salientado é que o tempo que cada ponto permanece aceso depende da qualidade do fósforo com que foi revestida a tela. Se este tempo não for suficientemente longo, quando o feixe atingir o final da tela, os pontos acesos no início da mesma já estarão apagados, ocorrendo o piscamento ("flicker") da imagem. De qualquer maneira o tempo que estes pontos permanecem acesos é limitado, fazendo com que exista um tempo máximo para que a regeneração da imagem comece a ser realizada, de modo a manter para o observador uma imagem estável.



——————— Linha de "Raster"

— — — — — Retraco Horizontal

——— · ——— Retraco Vertical

Figura 2.1 – Deslocamento do Feixe de Eletrons no padrao "Raster Scan"

Um outro aspecto importante a se ressaltar é que a taxa com que as imagens são geradas (ou regeneração da imagem) independe da quantidade de informação que é apresentada na tela.

Nesta tecnologia baseiam-se tanto os terminais gráficos como os terminais alfanuméricos e semigráficos, variando apenas a estrutura de memória e o controlador utilizado. Com o propósito de estabelecer uma base que permita a comparação dos diversos circuitos controladores de vídeo, que será feita no Capitulo 4, far-se-á aqui uma breve descrição das arquiteturas dos terminais alfanuméricos e gráficos.

2.2. Terminal alfanumérico

Considerando um terminal alfanumérico com 24 linhas de 80 caracteres, onde cada um destes caracteres é formado por uma matriz de 8 x 12 pontos (figura 2.2), isto corresponde à existência na tela de um total de 184.320 pontos. Esta informação porém é armazenada na memória de vídeo de forma compacta em um vetor de 1920 bytes (24 linhas x 80 colunas), correspondendo a cada um dos caracteres um código de 8 bits.

Isto implica na existência de um gerador de caracteres onde cada caractere é armazenado na sua forma final, isto é, uma matriz de 8 x 12 pontos como mostra a figura 2.2.

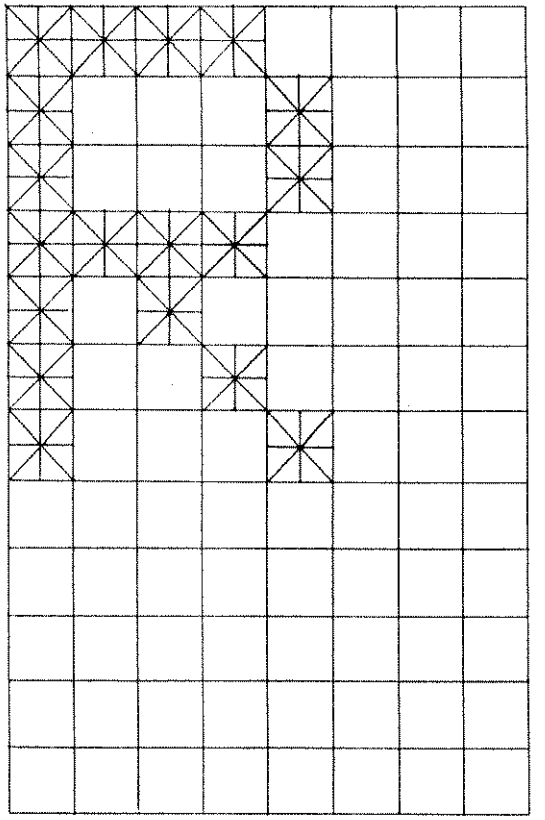


Figura 2.2 – Matriz de Pontos no Gerador de Caracteres Para a Letra "R"

O endereço da memória de vídeo é dividido como mostrado na figura 2.3, isto é, em linhas de caracteres e a geração da imagem é feita dividindo-se os caracteres em linhas de raster.

Assim, um byte que corresponde a um caractere da linha de caracteres é lido da memória de vídeo e apresentado ao gerador caracteres, juntamente com o endereço que corresponde à linha de raster corrente, criando o endereço correspondente à posição da ROM geradora de caracteres. Esta contém o padrão a ser mostrado, num total de 8 bits, que é carregado em um registrador de deslocamento, que se encarrega da serialização dos dados, como mostra a figura 2.4.

A geração de uma linha completa de raster demanda então a leitura dos 80 bytes da memória de vídeo, que representam a linha corrente de caracteres. Para a geração de uma linha de caracteres, os mesmos bytes são apresentados ao circuito gerador de caracteres, um número de vezes igual ao número de pontos verticais que formam a célula do caractere, ou número de linhas de raster de cada caractere (figura 2.5).

De modo análogo a geração de uma tela corresponde à reprodução do processo de geração de uma linha de caracteres, o número de vezes igual àquele de linhas de caracteres por tela.

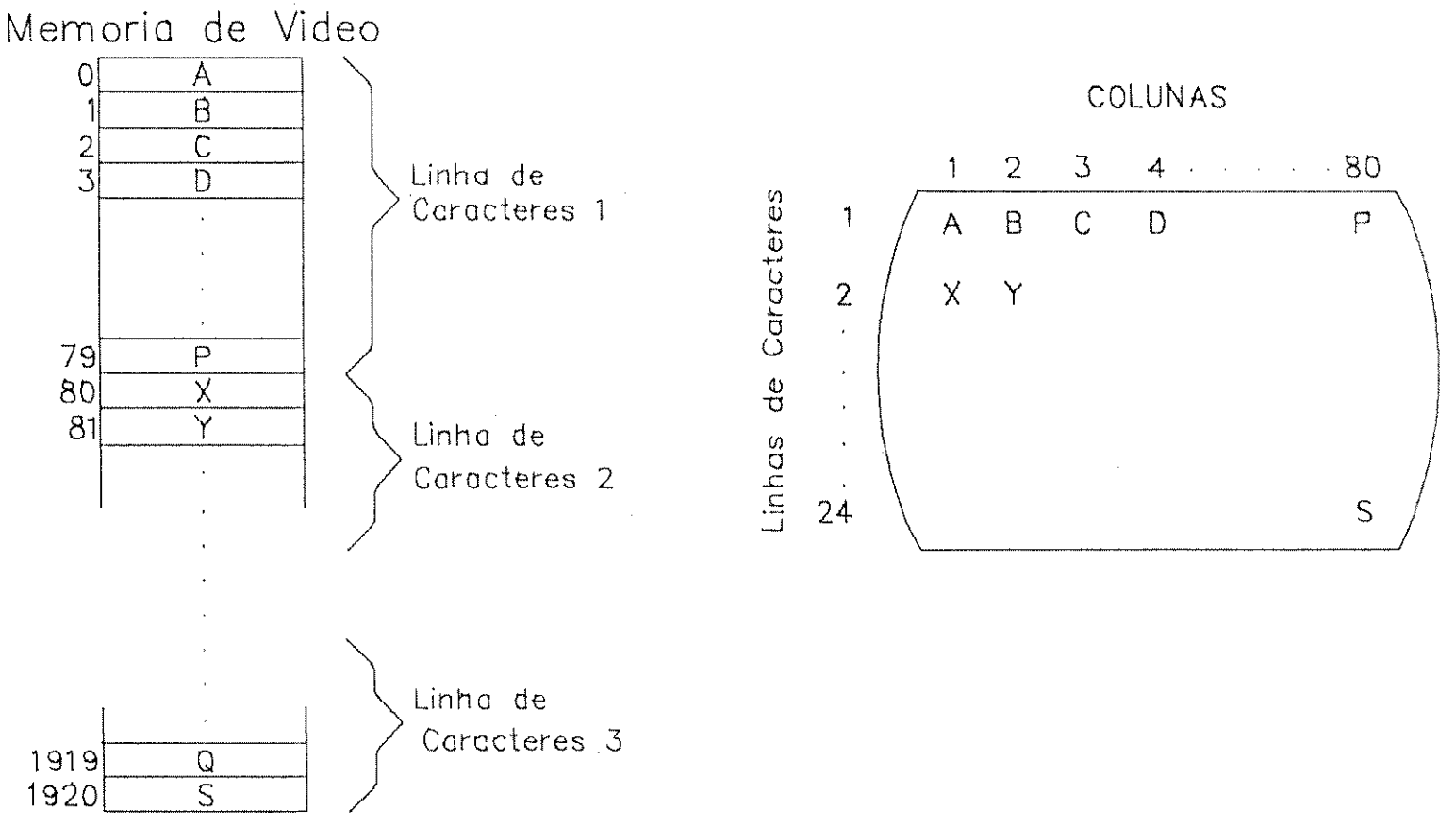


Figura 2.3 – Distribuicao dos Caracteres na Memoria de Video e sua Apresentacao na Tela do Terminal

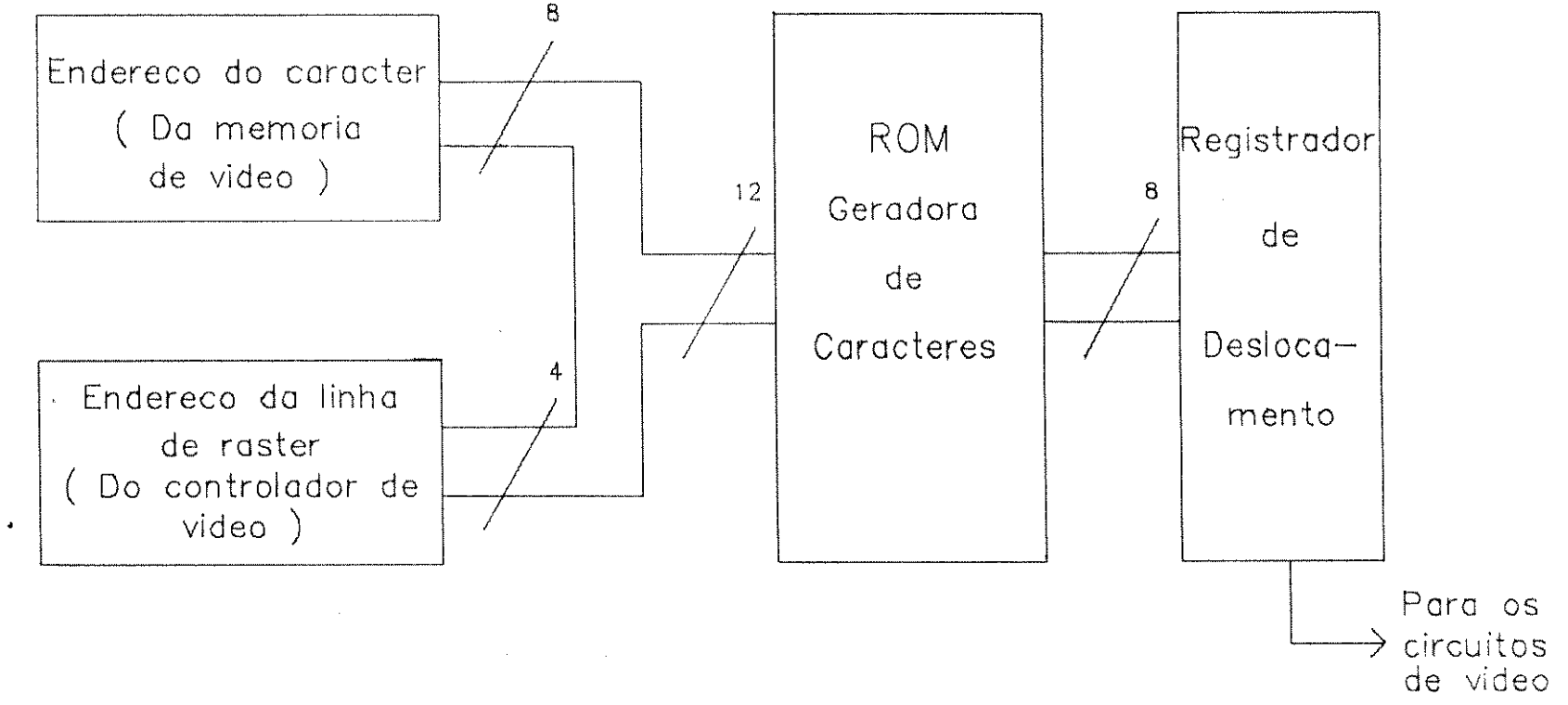


Figura 2.4 – Logica de Geracao de Caracteres

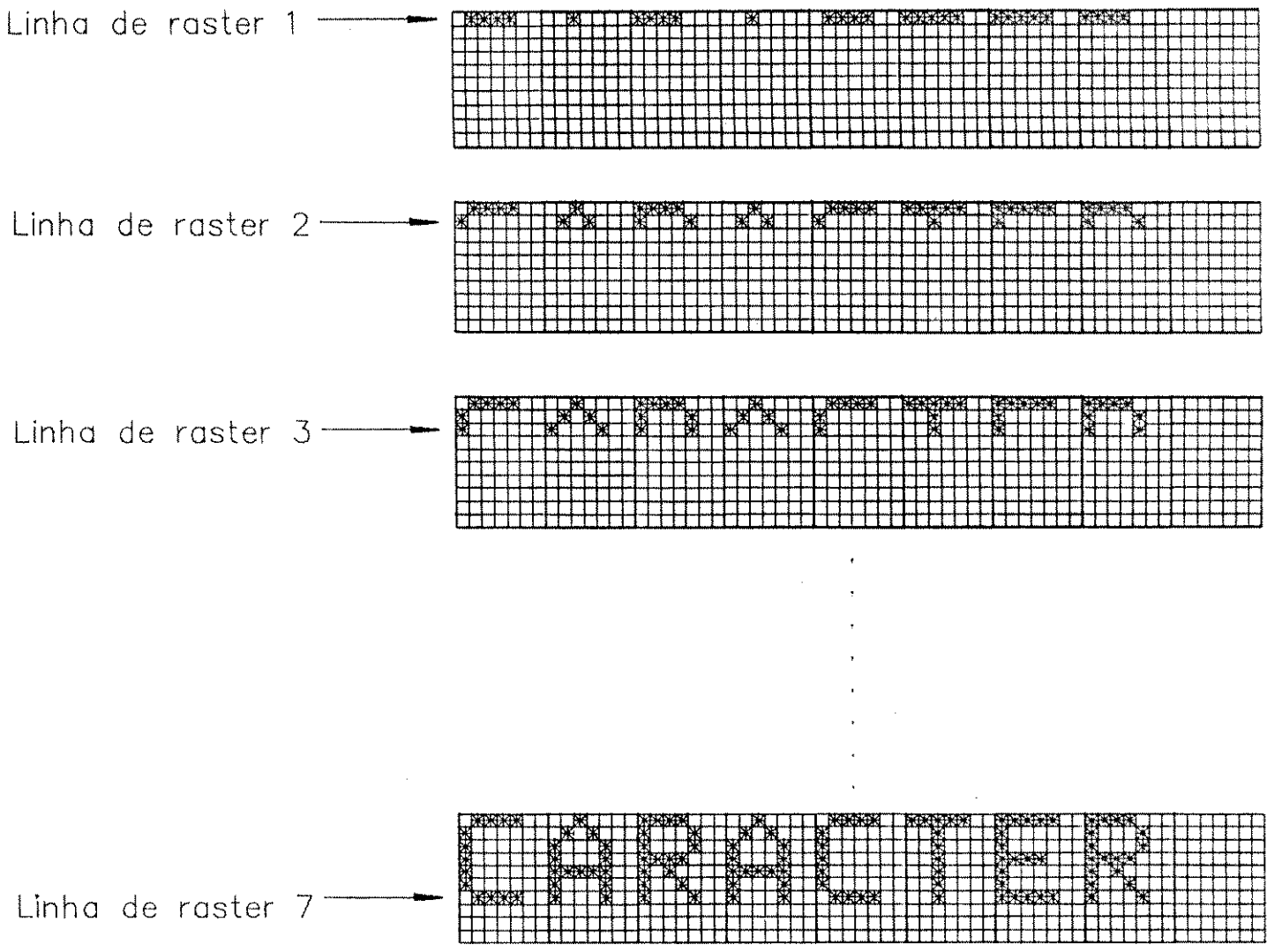


Figura 2.5 – Formacao dos Caracteres Atraves do Processo de "Raster Scan"

Desta forma, os controladores de vídeo para este tipo de aplicação devem apresentar essencialmente facilidades para programação do:

. número de caracteres por linha

. número de linhas de raster por linha de caractere

. número de linhas de caracteres por tela,

de forma a possibilitar a geração de endereços da memória de vídeo na sequência adequada.

O acesso a cada byte na memória de vídeo, considerando os dados do exemplo, uma taxa de regeneração de 30 quadros por segundo e as características dos CRT comerciais, com relação ao tempo de retraço vertical e horizontal, ocorre a cada 620 nS. Obviamente, esta alta taxa de acesso à memória de vídeo, causa conflitos no acesso a memória pela CPU e controlador sendo a primeira para atualização do conteúdo da memória e a segunda para a regeneração da imagem, obrigando o uso de mecanismos de resolução de conflitos no acesso à memória, como por exemplo a utilização de "buffers" para a linha de caracteres corrente. Este método reduz o acesso do controlador à memória de vídeo, com a leitura de 80 bytes a cada 0,76 mS aproximadamente, aumentando assim a disponibilidade da memória para a CPU.

A utilização de cores e atributos como "blink", sublinhado, reverso, etc, são codificados em um vetor de igual dimensão ao da memória de vídeo e utilizados para controle do gerador de caracteres. Deste modo, a simples duplicação da memória permite o armazenamento destes atributos, como mostra a figura 2.6.

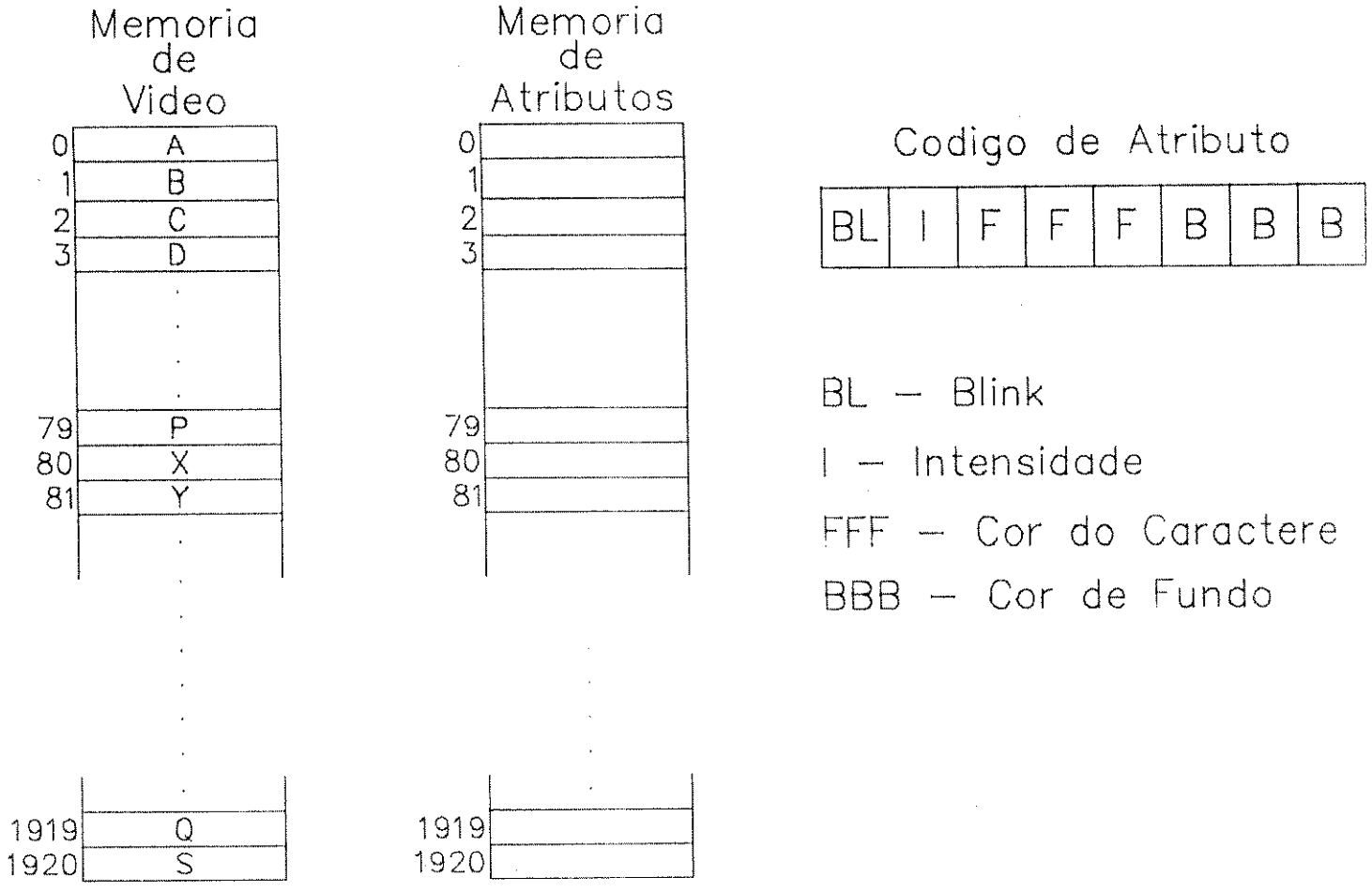


Figura 2.6 – Associacao entre a Memoria de Video e a Memoria de Atributos num Terminal Alfanumerico Colorido

No caso de terminais semigráficos, o processo é exatamente igual ao dos terminais alfanuméricos, exceto pelo fato que a matriz do gerador de caracteres apresenta símbolos gráficos especiais. Deve-se notar que pela utilização de multiplexadores, os conjuntos de padrões alfanuméricos e gráficos poderão ser utilizados simultâneamente.

2.3. Terminal gráfico (bit-mapped)

No tocante à sua arquitetura o terminal gráfico difere essencialmente do terminal alfanumérico pela ausência do gerador de caracteres (figura 2.7), uma vez que os dados correspondentes à imagem estão armazenados diretamente na memória de video. Assim, para a imagem de 640 x 288 pontos, de igual tamanho à descrita para o terminal alfanumérico, são necessários 184.320 bits ou 22,5 kbytes de memória.

Diferentemente do modo alfanumérico, cada byte da memória de imagem é acessado uma só vez por quadro e transferido ao registrador de deslocamento para a serializaçãc dos mesmos. Do ponto de vista de acesso à memória pelo controlador, os tempos são equivalentes aos do terminal alfanumérico, porém a solução dos problemas de conflito no acesso à memória torna-se mais difícil, uma vez que a simples utilização de "buffers" de linha, semelhante ao utilizado anteriormente, não é satisfatória pois o mesmo deve ser atualizado a cada linha de raster. Ainda, a quantidade

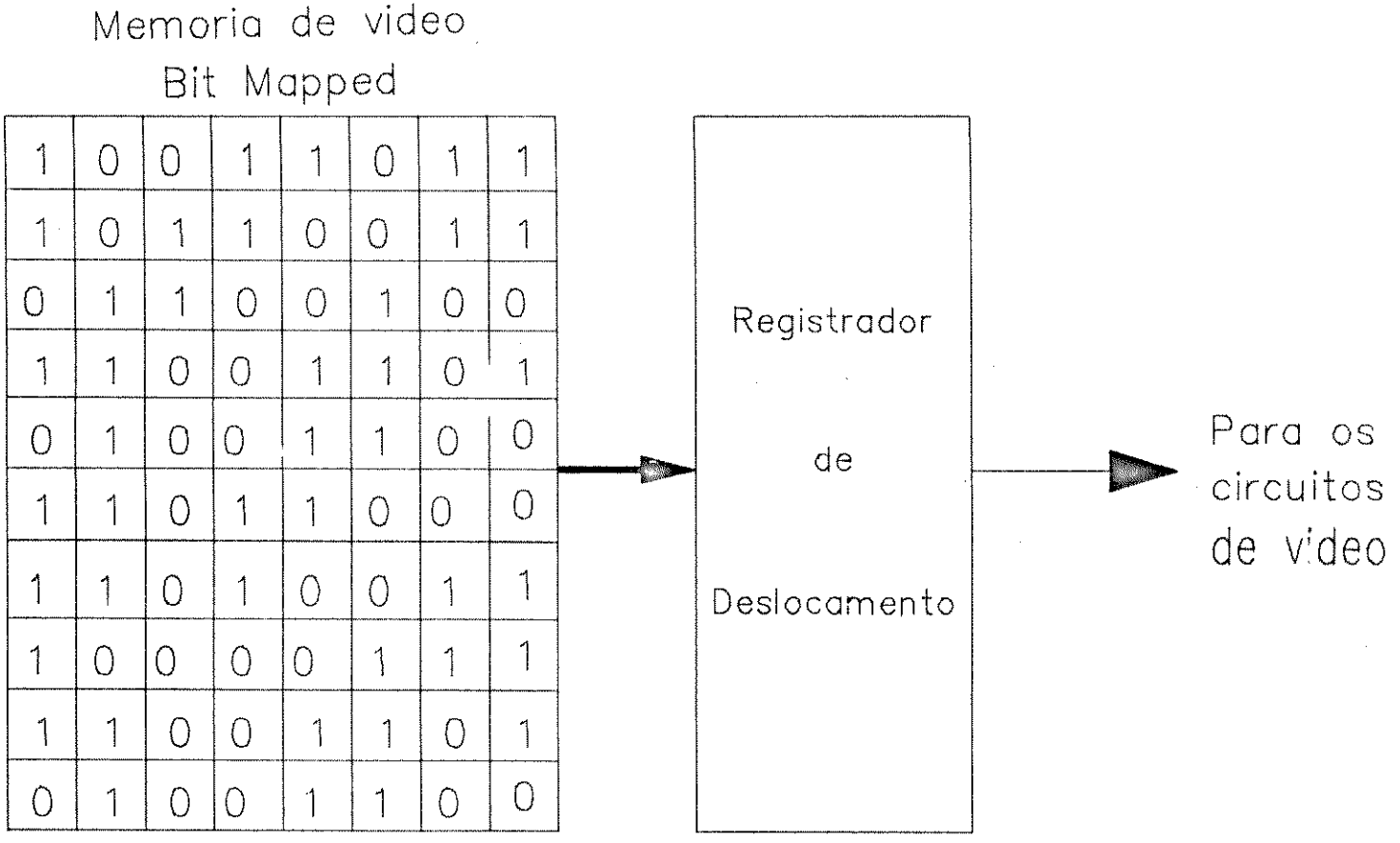


Figura 2.7 – Estrutura de Um Terminal Grafico "Bit Mapped"

de acessos do processador à memória de vídeo é muito maior, visto que cada pixel é uma unidade de informação independente, enquanto no caso do terminal alfanumérico a unidade de informação é o caractere (12 x 8 pixels).

A introdução de cores ou níveis de cinza num terminal deste tipo é feita pela utilização de mais de um bit para a codificação de um pixel; duas implementações são bastante utilizadas, a de um único plano de memória, onde a informação é organizada como mostra a figura 2.8. Isto implica que para a mesma resolução e considerando-se 4 cores (2 bits), deve-se ter o dobro do número de bytes. O acesso a cada byte deve ser feito na metade do tempo uma vez que a taxa de regeneração é mantida constante, agravando a disponibilidade de tempo para acesso da CPU à memória de vídeo. A outra solução consiste na utilização de planos de memória como mostra a figura 2.9. Neste caso, o tempo de acesso permanece igual, porém devem ser acrescidos mais registradores de deslocamento (um por plano), e em contrapartida o acesso do processador à memória de vídeo é bastante dificultado, visto que a informação de um pixel está distribuida por diferentes bytes. Este fato implica em múltiplos acessos à memória ou na utilização de mecanismos especiais de acesso que levam à maior complexidade do hardware.

Um outro aspecto que afeta fortemente os terminais gráficos e que não tem excessiva importância nos alfanuméricos é a resolução. Dado que a informação no terminal gráfico é o ponto, a representação de figuras geométricas fica excessivamente prejudicada em terminais de média e baixa resolução, com o aparecimento

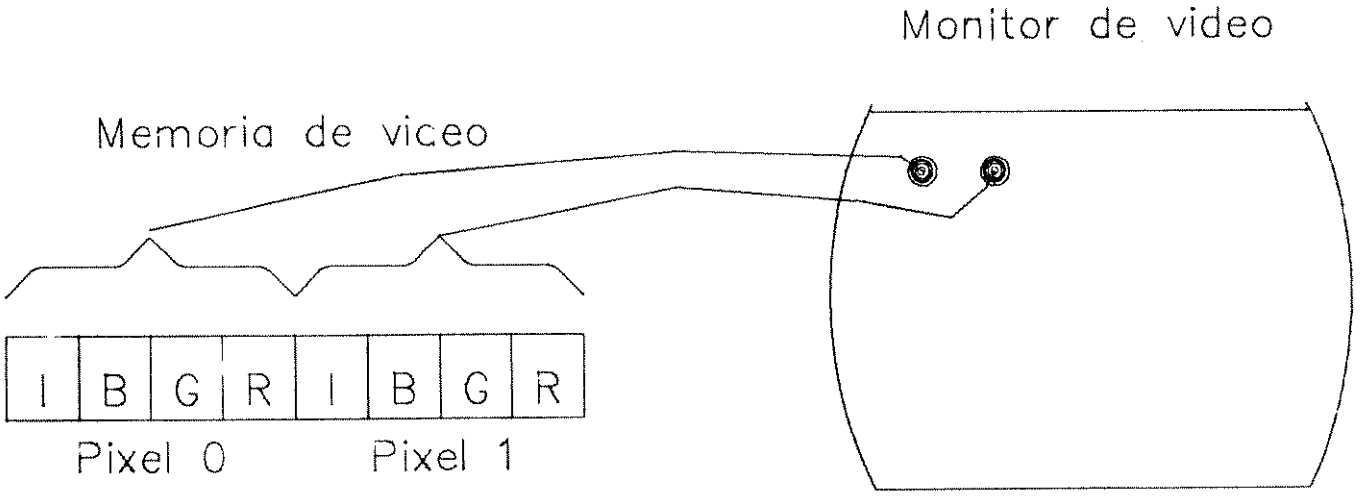


Figura 2.8 – Mapeamento dos Bits de Memoria de Video Para a tela do monitor

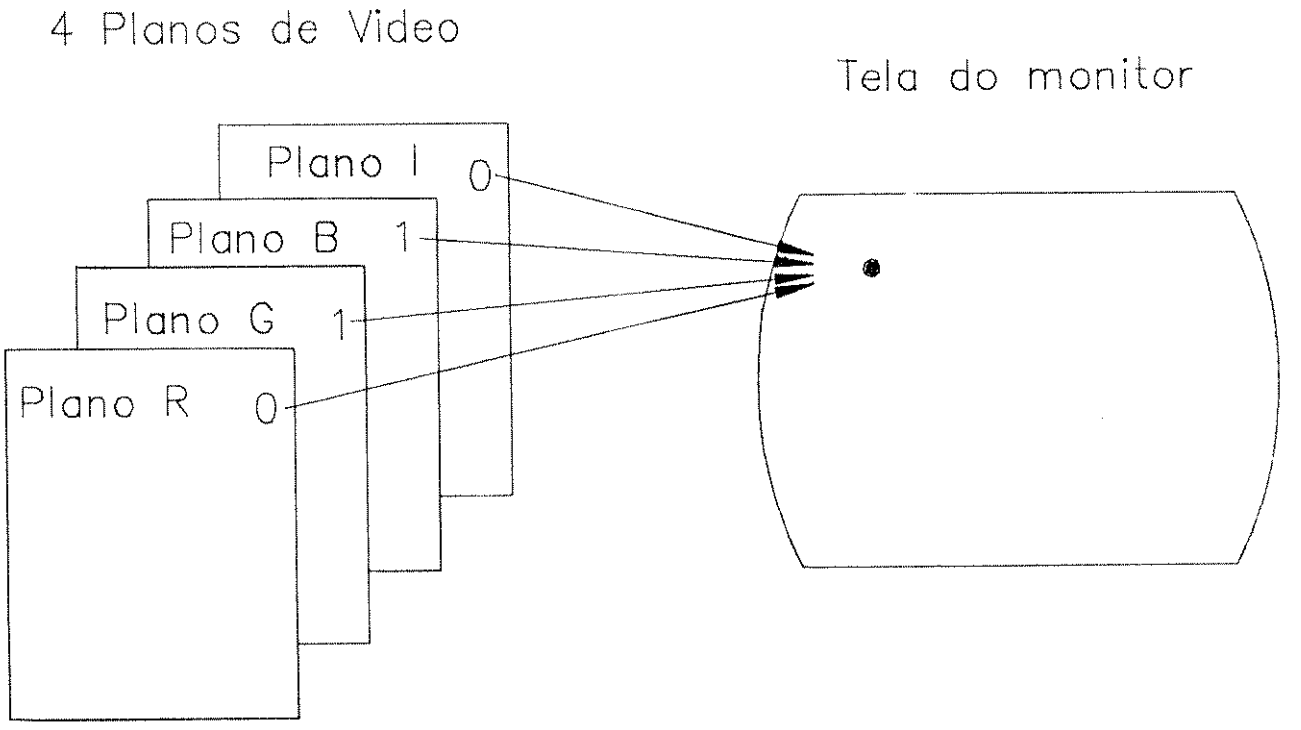


Figura 2.9 — Memoria de Video Distribuida em Planos

do efeito denominado "jagging". A eliminação deste efeito do ponto de vista do hardware (1) é obtida pelo aumento do número de pontos, o que obriga à diminuição do tempo de acesso à memória para se manter a taxa de regeneração. Dependendo da utilização de regeneração, no modo entrelaçado ou não entrelaçado, o tempo de acesso à memória é reduzido a tal ponto que a frequência de relógio do sistema atinge valores que causam problemas de banda de passagem nos CRT (circuitos analógicos) além de implicar na utilização de tecnologias de circuitos digitais mais velozes. A solução do acesso à memória está intimamente ligada à arquitetura utilizada.

Deve-se considerar também que tanto os controladores específicos para aplicação em terminais alfanuméricos como para aplicação em terminais gráficos são semelhantes nos aspectos de geração de endereços e sincronismo, sendo que nos controladores gráficos é exigido um número maior de linhas de endereços, uma vez que normalmente existem mais informações nos terminais gráficos. Existe ainda a possibilidade da existeêcia de hardware específico para a geração de primitivas gráficas, visando aumentar o desempenho do sistema.

---

(1) mesmo no caso de terminais de alta resolução este efeito é observado, e para sua eliminação é necessário o uso de algoritmos denominados "anti-aliasing" predominantemente realizados em software

Não pode ser esquecida num terminal gráfico a necessidade de um processador que permita a inicialização e gerência do terminal bem como do firmware para a implementação dos algoritimos que realizam a tranformação das primitivas em pixels. A geração destes pixels (geração da imagem e atualização da mesma) pode ser feita tanto pelo processador do terminal gráfico, ou por um processador dedicado incorporado ao sistema gráfico, aumentando a eficiência de todo o conjunto. A velocidade do sistema gráfico fica então dependente dos processadores utilizados. Microprocessadores de 32 bits são hoje utilizados e mesmo controladores de vídeo com capacidade de geração de primitivas por hardware são incorporados aos terminais gráficos mais modernos.

3. ASPECTOS DA ARQUITETURA DE UM TERMINAL DE VIDEO GRAFICO COLORIDO USANDO TECNOLOGIA "RASTER SCAN"

Um sistema de vídeo gráfico a cores é composto por quatro blocos básicos: o processador de vídeo (CPU), a memória de vídeo (Display RAM), o controlador de vídeo (CRTC), e o monitor, interligados como mostra a figura 3.1.

3.1. Definição dos blocos componentes de um sistema de vídeo gráfico.

3.1.1. Monitor de vídeo

Uma vez que o controlador de vídeo para a regeneração da imagem deve acessar a memória de imagem, respeitando as restrições com relação à temporização dos sinais de vídeo, é importante que se entendam algumas das definições basicas usadas em terminais gráficos.

O padrão com que um feixe de eletrons percorre o Tubo de Raios Catódicos (CRT) à medida em que ele atravessa a tela da esquerda para a direita e de cima para baixo, compõe-se de um conjunto de linhas horizontais paralelas como as mostradas na figura 3.2. Este padrão é chamado "raster" ou varredura, e cada linha traçada é chamada linha de raster ou "scan". A taxa com que são formadas imagens completas ou quadros é chamada taxa de "refresh", medida em quadros por segundo.

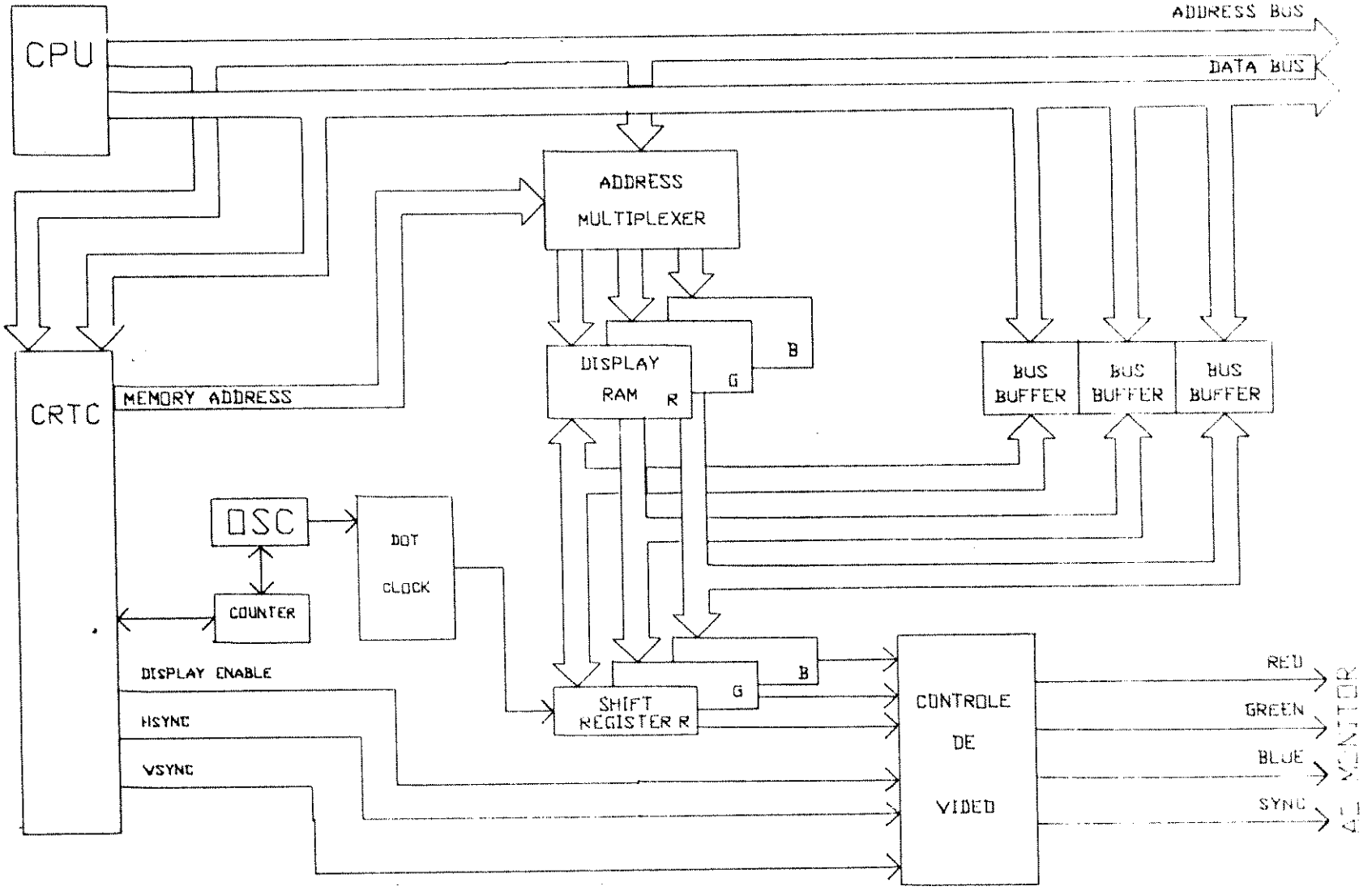


Figura 3.1 – Diagrama Tipico de Um Sistema de Video Grafico Mostrando seus Principais Componentes

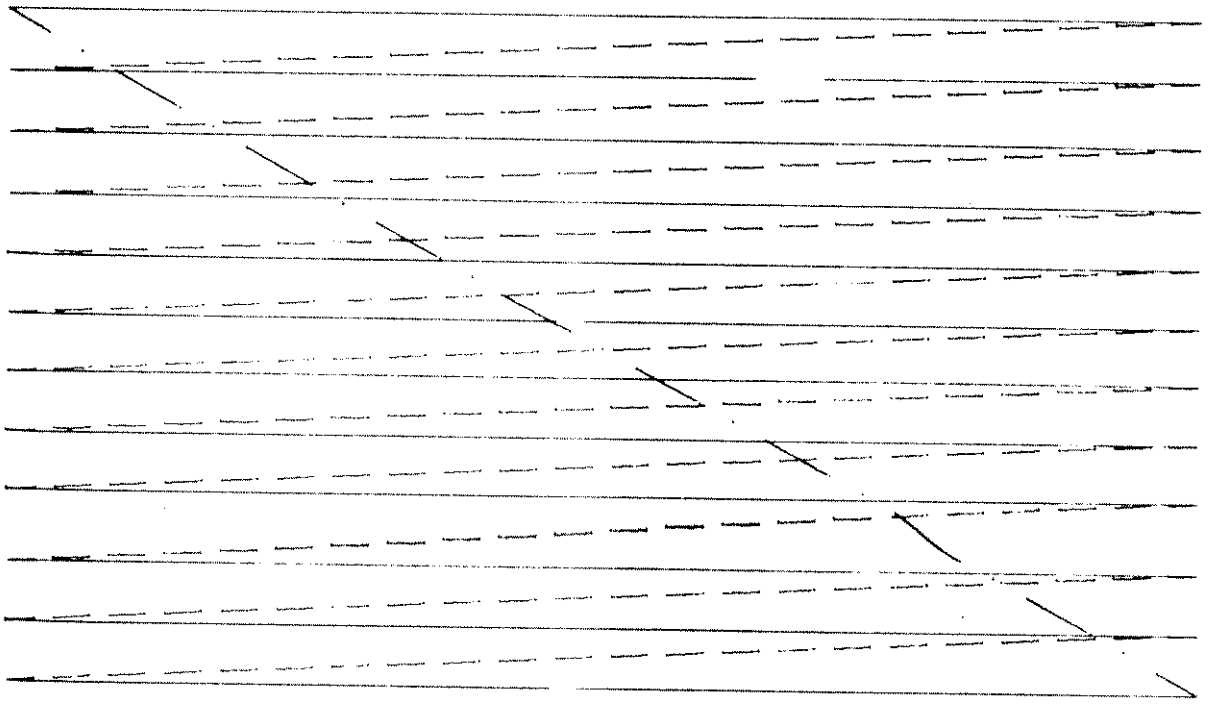


Figura 3.2 – Deslocamento do Feixe de Eletrons no Padrao "Raster Scan"

Um certo tempo é necessário para que o feixe de eletrons retorne da parte direita da tela até a esquerda. Este tempo é chamado tempo de retraço horizontal; do mesmo modo, temos o retraço vertical, quando o feixe chega ao fim da tela (canto inferior direito) e retorna ao inicio da mesma (canto superior esquerdo) e o tempo gasto é definido como tempo de retraço vertical.

Uma outra definição utilizada em terminais de vídeo, desenvolvida para as transmissões de TV comerciais, é o conceito de entrelaçamento, que significa a alternancia do traçado de linhas pares em um quadro e linhas impares no outro quadro, formando estes dois uma imagem completa. Cada quadro de N linhas de raster é dividido em dois campos entrelaçados de N/2 linhas cada. O campo ímpar contém as linhas horizontais 1, 3, 5,.., N-1, e o campo par contém as linhas 2, 4, 6, ..., N. As linhas do campo par se encontram precisamente entre as linhas do campo ímpar. Um exemplo de como esta técnica pode ser utilizada em um terminal de vídeo, é mostrado na figura 3.3, onde em *a* é mostrado a formação de um caractere no modo não entrelaçado; em *b*, o mesmo caractere no modo sincronismo entrelaçado; e em *c*, no modo vídeo e sincronismo entrelaçados onde alterando-se as informações dos campos par e ímpar, é possível dobrar o número de informações.

Normalmente, os terminais de vídeo tem uma taxa de 60 quadros por segundo, ou seja, a imagem é gerada 60 vezes por segundo num sistema não entrelaçado ou 30 quadros por segundo num sistema

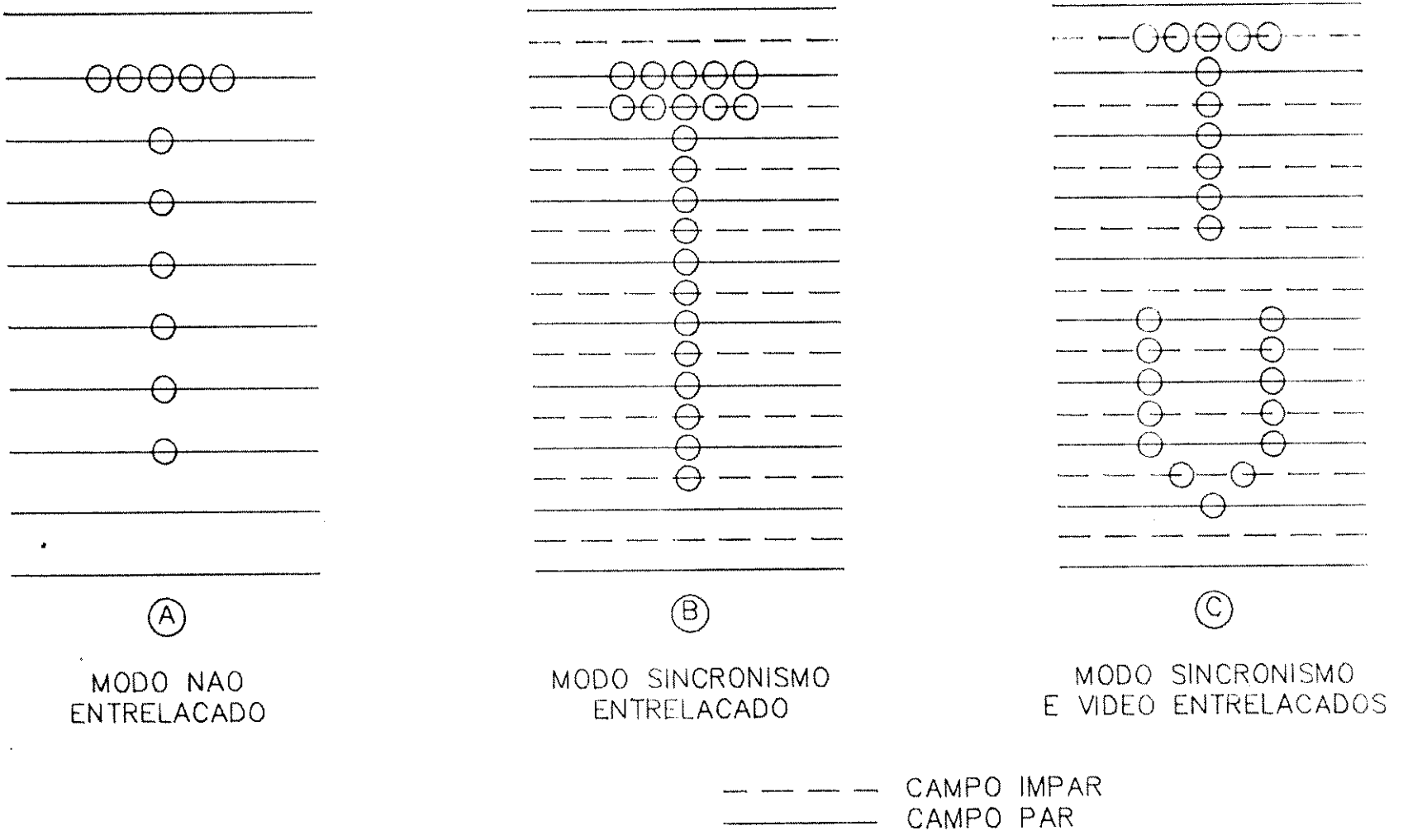


Figura 3.3 – Modos de Video

entrelaçado, alternando campos pares e ímpares. Imagens geradas a taxas inferiores a 30 quadros por segundo, apresentarão o efeito de piscamento ou "flicker".

Na formação de uma imagem pelo processo de "raster" os seguintes tempos devem ser considerados: tempo de retraço vertical, tempo de retraço horizontal, e tempo de vídeo, que correspondem às áreas marcadas 1,2 e 3 respectivamente na figura 3.4.

A área 3 representa o tempo utilizado pelo vídeo ativo, ou seja, para a representação da imagem através de seus pixels na tela. Este tempo é produto do numero de linhas visíveis por tela, o número de pixels por linha e o tempo de cada pixel.

A área 2 representa o tempo necessário para que o feixe de eletrons seja desviado da direita da tela para a esquerda no retraço e a área 1 representa o tempo necessário para que o feixe seja desviado do fim da tela para o começo da mesma. Durante os períodos 1 e 2, a memória de vídeo não é utilizada pela lógica de "refresh" de vídeo. A soma dos tempos 1, 2 e 3 representa o tempo de um quadro. A Tabela 3.1 mostra um resumo dos tempos necessários para as várias resoluções de vídeo, supondo 60 quadros por segundo não entrelaçados. Os tempos de retraço horizontal e vertical apresentados, são valores obtidos de monitores de vídeo comerciais.

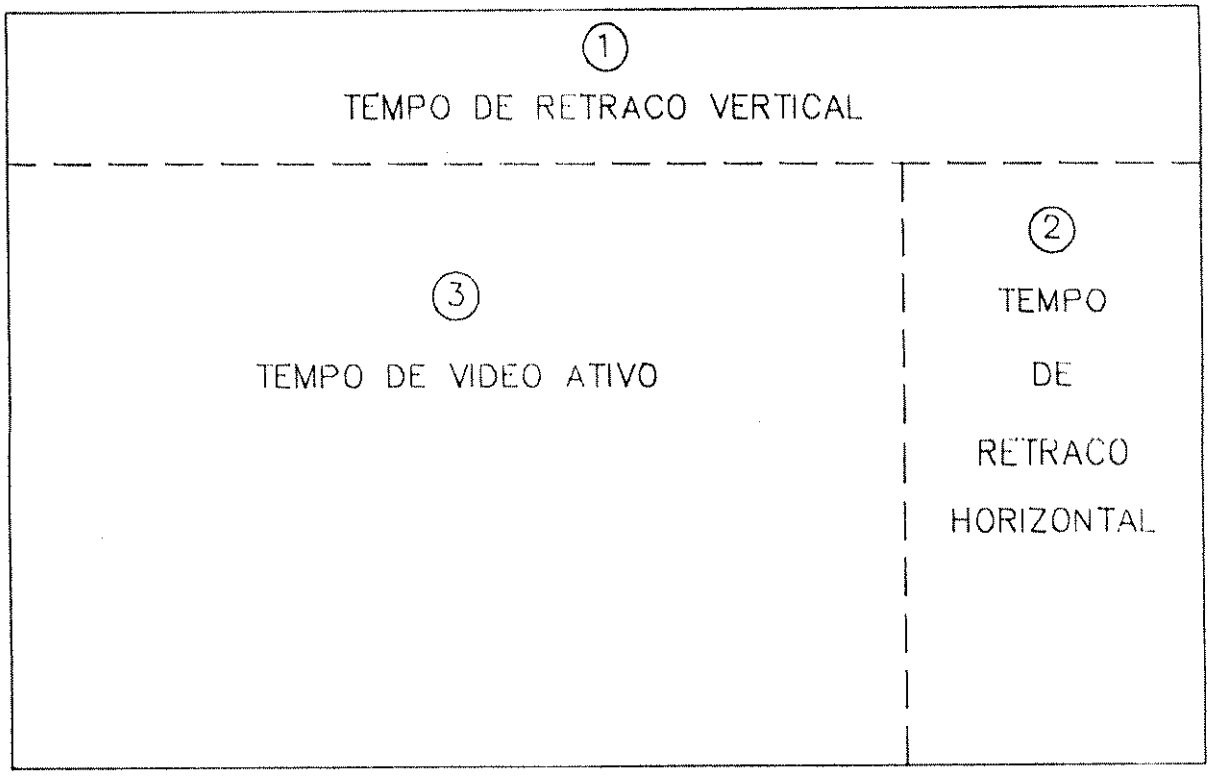


Figura 3.4 — Distribuicao dos Tempos de Video

TABELA 3.1

NUMERO DE LINHAS	NUMERO DE COLUNAS	TAMANHO DA MEMORIA	TEMPO DE RETRACO VERTICAL	TEMPO DE RETRACO HORIZONTAL	TEMPO DE QUADRO ATIVO	TEMPO TOTAL DE RETRACO	TEMPO DE UM PIXEL
384	512	192K	900 uS	7.0 uS	13.08 mS	3.58 mS	66.5 nS
512	640	320K	800 uS	6.5 uS	12.54 mS	4.12 mS	38.3 nS
768	1024	768K	700 uS	6.0 uS	11.36 mS	5.30 mS	14.4 nS
1024	1280	1280K	650 uS	5.4 uS	10.43 mS	6.17 mS	8.0 nS

Outro aspecto a ser considerado com relação aos monitores de vídeo é a relação entre os sinais de "blank" (apagamento do feixe de eletrons) e os sinais de sincronismo. Como mostra a figura 3.5.a, o sinal de blank ocorre antes do sincronismo horizontal e o tempo entre estes é chamado "horizontal front porch" e desaparece em um tempo chamado "horizontal back porch" depois do desaparecimento do sinal de sincronismo horizontal. O mesmo ocorre para o sinal de sincronismo vertical (figura 3.5.b). Estes tempos servem basicamente para a centralização do quadro na tela do monitor e são, na maioria dos controladores de vídeo, definidos através de registradores programáveis.

Quando se utiliza o modo entrelaçado, os tempos "vertical back porch" e "vertical front porch" são utilizados para a geração dos quadros pares e impares conforme mostra a figura 3.6, onde é apresentado a temporização dos sinais de sincronismo

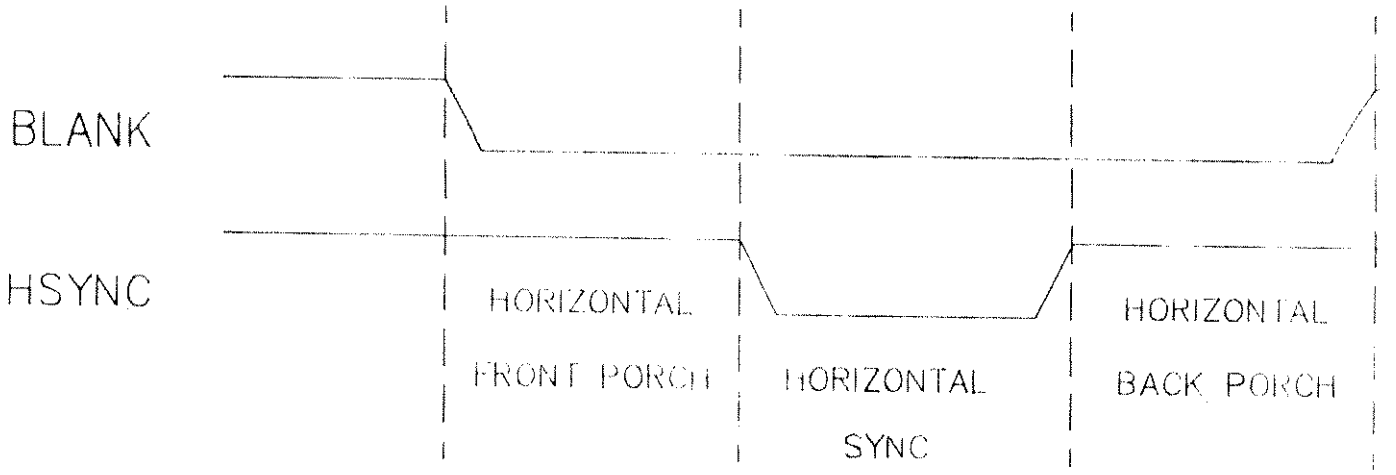


Figura 3.5.A – Tempos de Sincronismo Horizontal e Blank

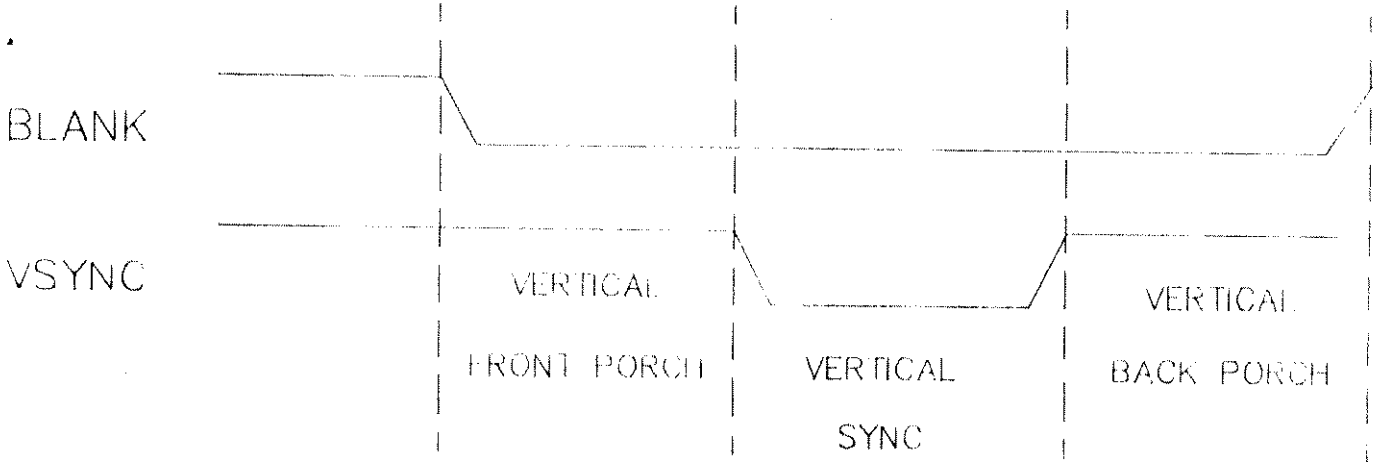


Figura 3.5.B – Tempos de Sincronismo Vertical e Blank

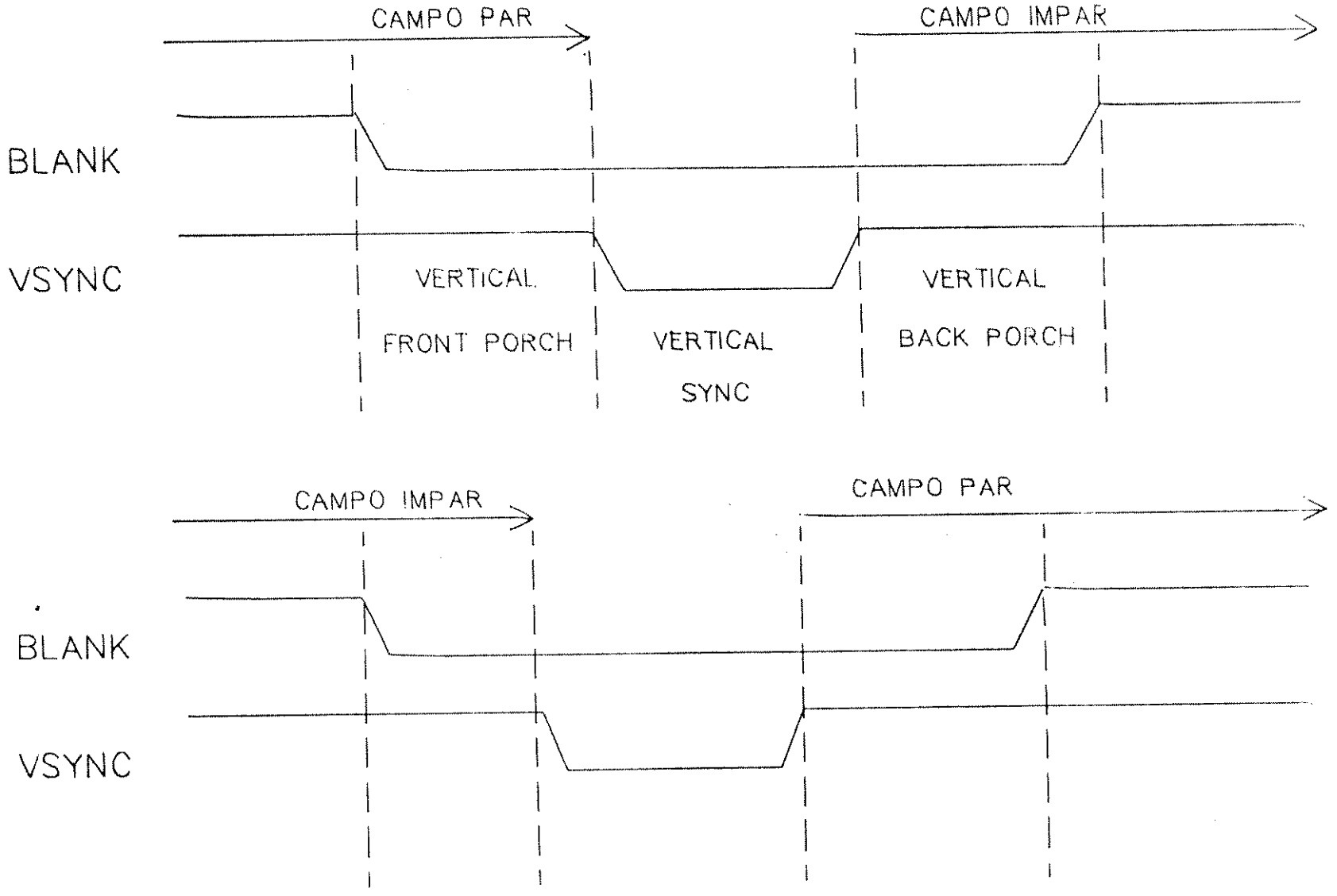


Figura 3.6 – Sinais de Sincronismo e Blank no Modo Entrelacado

vertical e blank. O sinal de sincronismo vertical (VSYNC), que precede um campo impar, começa e termina coincidindo com o inicio dos pulsos de sincronismo horizontal, como ocorre no modo não entrelaçado. Já o sinal VSYNC que precede o campo par, começa e termina precisamente no meio do intervalo entre os inicios de dois pulsos de sincronismo horizontal como pode ser visto na figura 3.6. O sinal de BLANK começa e termina coincidindo com o inicio dos pulsos de sincronismo horizontal. A duração do sinal BLANK que segue um campo par tem o tempo de uma linha horizontal a menos do que o que segue o campo impar. Isto serve para assegurar que uma linha completa de dados é mostrada no inicio e no final de cada campo.

3.1.2. O processador de vídeo (CPU)

O processador de vídeo é o sistema encarregado de preencher a memória de vídeo com pixels correspondentes às primitivas gráficas ou comandos recebidos do usuário. Deve ser capaz de executar os cálculos necessários para a geração dos pixels, numa velocidade suficientemente rápida tal que algumas classes de figuras como: retas, polígonos, caracteres, círculos, etc, sejam geradas no menor intervalo de tempo possível. Deve também executar a operação denominada "raster-op" que permite copiar uma área de memória de vídeo para outra, com ou sem a realização de operações lógicas entre os dados fonte e destino, além das tarefas de inicialização do sistema e comunicação.

3.1.3. O controlador de vídeo

O controlador de vídeo, também chamado controlador de "refresh" (regeneração da imagem), tem como função a geração de sinais de sincronismo e endereços de memória tal que o conteúdo da memória de vídeo seja lida na sequencia apropriada e com temporização precisa. E também responsável pela tranferência dos dados lidos da memória para a entrada do circuito de vídeo, que converte os dados binários em sinais analógicos correspondentes às intensidades ou cores, adequadas ao monitor (CRT).

A cada acesso do controlador à memória de vídeo, a informação correspondente a um conjunto de pixels é lido e armazenado no buffer de video. Além da compatibilização dos tempos de acesso à memória, este buffer tem como finalidade a conversão dos dados do formato paralelo para formato série. Se o buffer de vídeo for um registrador de deslocamento com carregamento paralelo, a serialização pode ocorrer sem hardware adicional.

3.1.4. A memória de vídeo

A memória de video, chamada "frame buffer memory" ou "display RAM" ou ainda "bit map", retém o conjunto de valores que representa a imagem. O formato do arranjo, por exemplo 512 x 512 ou 1024 x 1024 define a resolução da memória, sendo cada posição do arranjo, um elemento de imagem, ou simplesmente "pixel". Um arranjo de memória contendo um bit por pixel é chamado plano de memória. Uma memória de vídeo pode ter múltiplos planos e portan-

to armazenar vários bits por pixel. O número de bits por pixel é chamado de resolução de cores de uma memória de imagem. Por exemplo, sistema com 24 planos de memória, oito planos para cada uma das três cores primárias, vermelho, verde e azul, possui resolução de cores igual a 24, possibilitando a representação de 2^{24} cores, ou seja 16.777.216 cores.

Como pode ser observado na figura 3.1 a memória de vídeo é acessada tanto pela CPU, para a atualização da imagem, como pelo CRTC para a regeneração da tela, ocasionando assim a ocorrencia de conflitos no acesso à mesma. A taxa de conflitos no acesso à memória é um fator importante no desempenho final do sistema gráfico.

O desempenho de um sistema gráfico pode ser avaliado através do tempo gasto entre o pedido do usuário e a geração da primitiva gráfica correspondente. Considerando que para a geração da primitiva, uma grande quantidade de acessos à memória RAM é necessária, torna-se fundamental para o bom desempenho do sistema que a memória de vídeo permaneça disponível para acessos da CPU a maior parcela de tempo possível, o que torna o projeto desta memória fator de fundamental importancia na arquitetura de um sistema gráfico.

A implementação da memória deve ser um arranjo tal que se comporte como uma memória "dual port" ou "multiport", isto é,

podem ser acessadas por dois elementos, ou seja , dispõem de dois caminhos nas suas entradas e saídas, partilhados por estes elementos. A figura 3.7 mostra um exemplo de uma memória do tipo comum (porta única) usada num sistema de vídeo. Neste caso, existem dois caminhos de acesso à memória: um para o processador de vídeo e um para o controlador de vídeo.

Alguns arranjos de memória possuem espaço suficiente para armazenar duas imagens (ou páginas) completas, tanto em resolução como em quantidade de planos. Isto é, o buffer de memória é maior que a resolução da tela em número de pixels. Neste caso é possível ao processador acessar uma das páginas, enquanto o controlador acessa a outra. Quando uma nova imagem é gerada, o controlador passa a acessar esta memória enquanto o processador começa a alterar a outra, escrevendo nela as mudanças feitas anteriormente na outra página; em seguida escreve as novas mudanças necessárias, fazendo com que para cada alteração da imagem, sejam necessárias duas alterações na memória. Esta técnica é chamada de "double buffering".

3.2 Características das memórias para terminais de vídeo gráficos

Os displays gráficos têm sua memória de vídeo baseada em dispositivos semicondutores, RAM's estáticas ou dinâmicas. Com o atual estágio da tecnologia destes dispositivos, as memórias dinâmicas têm sido mais utilizadas devido ao seu baixo custo e grande capacidade de armazenamento. Porém a necessidade dos cic-

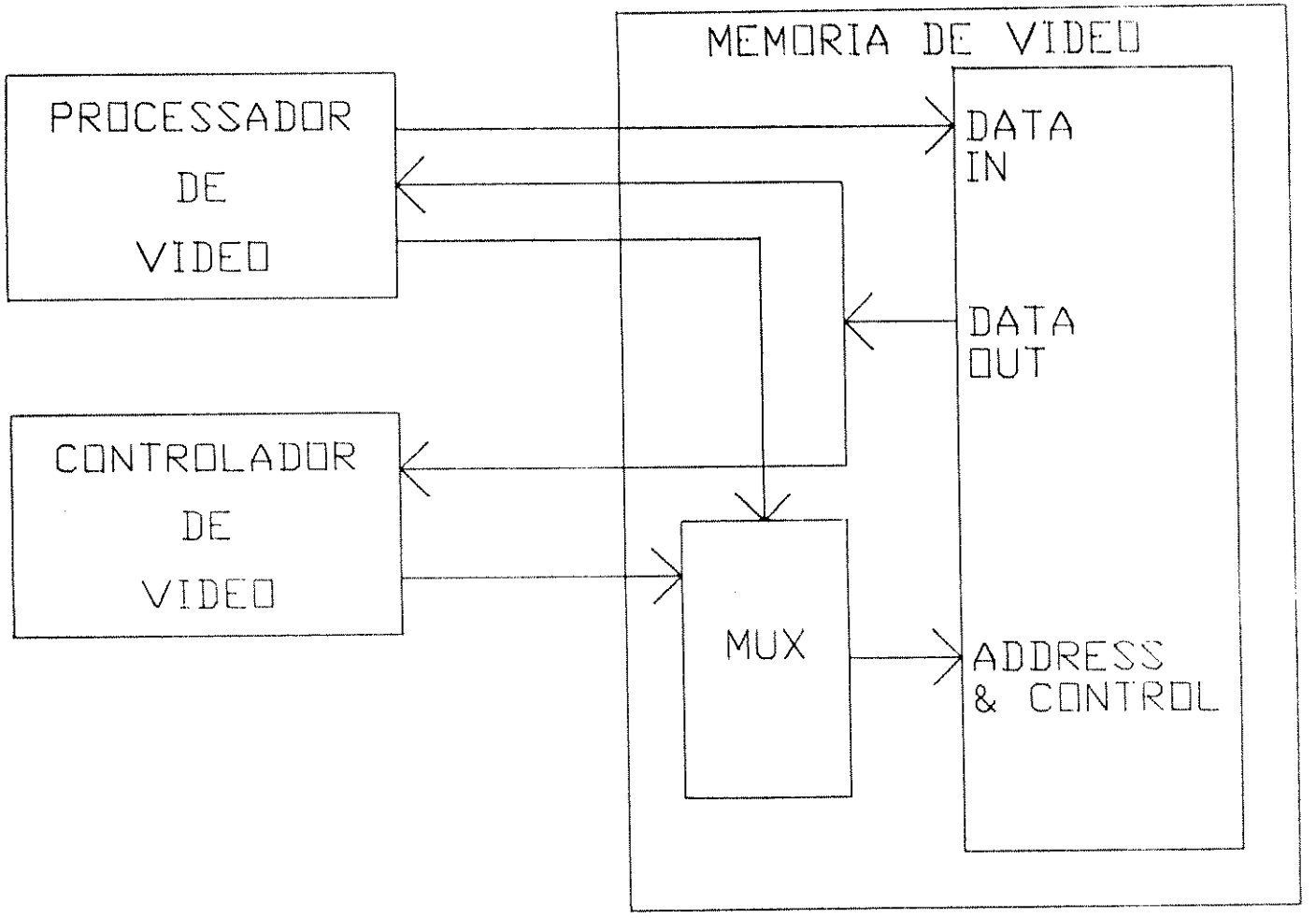


Figura 3.7 – Memoria Comum Usada Como Memoria "Dual Port"

los de refresh, necessários para que sejam mantidas as informações nestes dispositivos, torna o projeto mais difícil, não afetando contudo nem o custo nem o desempenho do sistema. As memórias dinâmicas exigem que ciclos de "refresh" sejam executados a uma taxa de aproximadamente um ciclo a cada quatro milisegundos, para cada uma das linhas do arranjo interno de dados no dispositivo. Circuitos especiais de vídeo podem ser projetados para que esta taxa seja atendida no acesso da memória para a regeneração da imagem, sem a necessidade de circuitos especiais dedicados ao refresh.

O ciclo de refresh do dispositivo de memória necessita de uma parcela pequena de tempo se comparado aos acessos tanto do controlador como do processador. No caso da utilização de circuitos especiais para o refresh da memória, estes devem ser projetados de tal forma a não tomar tempo demais da memória, permitindo o fácil acesso à mesma, tanto pelo processador como pelo controlador. Apesar disto, duas caracteristicas são fundamentais nos componentes de memória: tempo de acesso, que é o menor tempo entre o componente ser endereçado e os dados estarem disponíveis nas saídas; e ciclo de memória, que é o tempo mínimo necessário entre dois acessos consecutivos ao mesmo componente.

Outra caracteristica importante é a multiplexação das linhas de endereços em memórias de grande capacidade, com a finalidade de diminuir a quantidade de pinos necessários para estes sinais. Deste modo, a memória está organizada como uma matriz dividida em linhas e colunas, e para o acesso aos dados, torna-se necessária

a divisão do endereço em dois grupos, acionando-se sinais de RAS (row address strobe) e CAS (column address strobe), para cada um dos conjuntos correspondentes.

Devido à necessidade de se manter uma alta taxa de transferência de dados, várias técnicas de acesso à memória de vídeo foram desenvolvidas para permitir a rápida obtenção das informações armazenadas. Estas técnicas são discutidas a seguir.

3.2.1 Modo Página

No modo página, os endereços internos da memória são divididos em linhas e colunas. Um dispositivo de 64 Kbits possui 16 linhas de endereços, sendo 8 delas para linha e 8 para colunas. Devido à arquitetura interna do dispositivo, o acesso a uma nova coluna dentro da mesma linha é executado de uma maneira mais rápida do que no acesso normal. No modo normal, o ciclo de memória é de 260 ns, ao passo que no modo página, este tempo cai para 125 ns (50%). Deve-se ressaltar nesta técnica, o fato do endereço da linha permanecer inalterado por um tempo limitado, precisando ser atualizado em seguida, caso contrário seu conteúdo será perdido. Este fato faz com que o projeto de um terminal onde o tempo de uma linha de raster seja maior do que o tempo de retenção do endereço, se torne dificil. Para contornar este problema, alguns fabricantes fornecem componentes com modo página estendido, os quais são capazes de reter a informação de endereço de linha por 75 us, tornando-se viável para utilização na maioria

dos projetos de terminais de video.

3.2.2 Modo Nibble

Este modo permite que quatro bits de dados sejam acessados a cada ciclo de RAS. Neste tipo de componente, organizado externamente como 64K x 1, a estrutura interna é 16K x 4. A cada ciclo de RAS, os oito bits mais significativos do endereço são usados para endereçar uma linha, e os seis bits menos significativos, o endereço da coluna. Assim, quatro bits de dados são armazenados num "latch" interno ao componente, e que poderão ser acessados através dos dois bits menos significativos do endereço. A cada um dos três pulsos subsequentes de CAS, os outros três bits de dados são acessados sequencialmente. Este tipo de organização permite taxas de transferencia da ordem de 80 ns por bit.

3.2.3 Modo Ripple

Neste modo, a memória opera de maneira similar ao modo página, porém circuitos especiais de "look ahead" fazem com que os tempos de acesso sejam da ordem de 40 ns por bit contra os 125 ns por bit no modo página.

3.2.4 Vídeo RAM's

Os dispositivos tipo vídeo RAM's são componentes desenvolvidos recentemente visando especificamente as aplicações gráficas. Nestes componentes, o conceito de "dual port" já está incorporado ao dispositivo, eliminando grande parte do problema de partilhamento dos dados da memória pela CPU e pelo controlador de vídeo. Este aumento de desempenho é conseguido eliminando-se a necessidade do controlador de vídeo acessar constantemente a memória para a regeneração da imagem.

A organização de uma vídeo RAM típica é mostrada na figura 3.8, onde temos um arranjo interno de 256 x 256 bits (64 Kbits) e um registrador de deslocamento de 256 bits que é controlado externamente. Assim, para o acesso normal da CPU, um bit dentro do arranjo é alterado, ao passo que para o acesso do controlador, 256 bits contidos em uma das 256 linhas de dados são transferidos ao registrador de deslocamento, que passa a ser controlado externamente.

O aumento de velocidade nas Vídeo RAM's é devido basicamente à presença de uma arquitetura que permite a transferencia de 256 bits do arranjo de memória para os registradores de deslocamento internos ao componente num ciclo de 260 ns; após a tranferencia, o componente passa a se comportar como dois dispositivos independentes: uma memória tipo RAM dinâmica de 64 Kbits com ciclo de acesso de 260 ns e um registrador de deslocamento de 256 bits que pode ser usado com um clock independente de até 25 MHz para

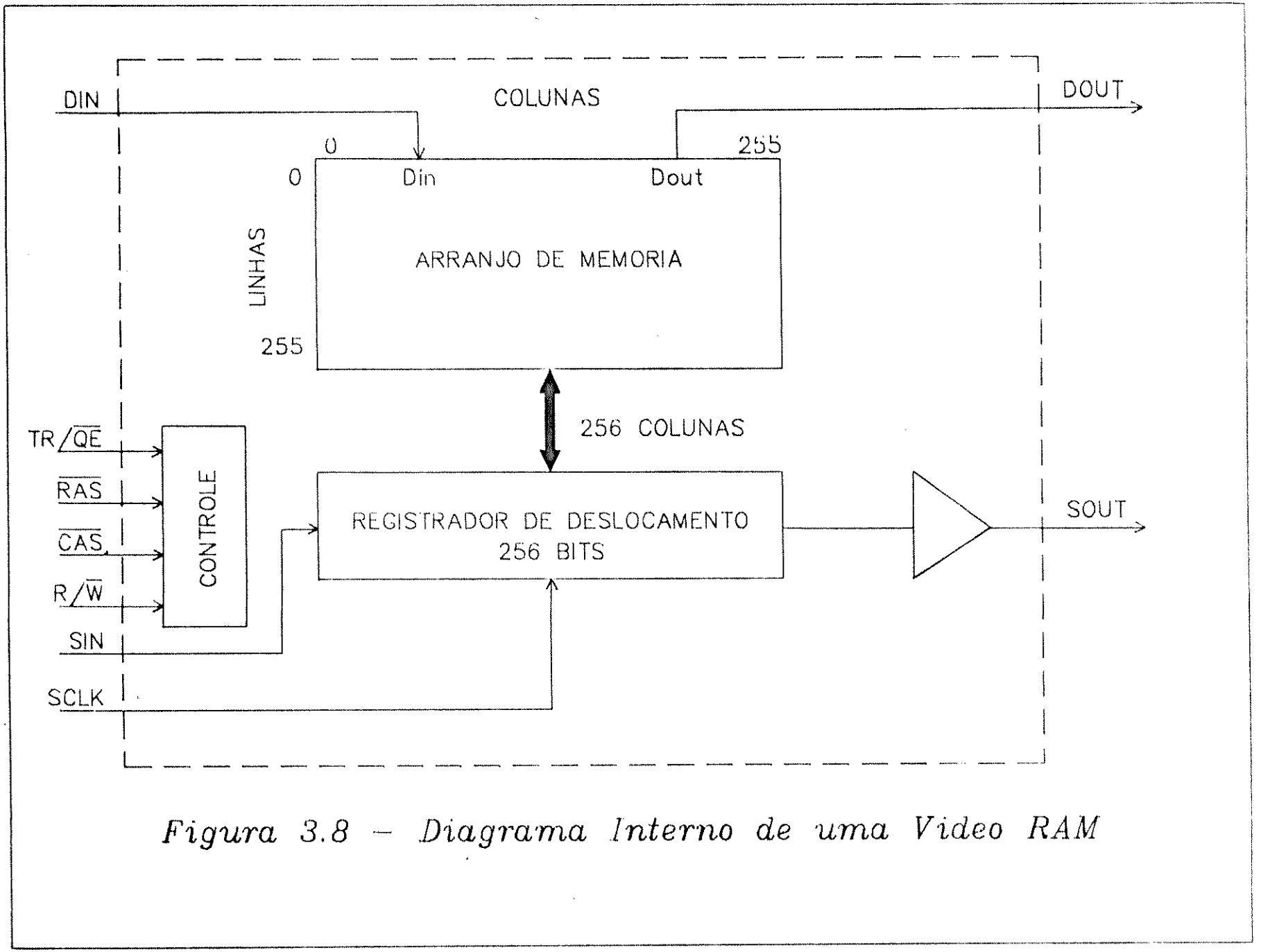


Figura 3.8 – Diagrama Interno de uma Video RAM

manter a regeneração da imagem.

3.3 Organização da memória de vídeo

A caracteristica mais desejável em um terminal de vídeo é a apresentação de uma imagem estável. Isto só é possivel se o projeto dos circuitos de acesso à memória através do controlador for feito de forma a permitir a maior taxa de transferencia de informação entre estes. Por outro lado, deseja-se que novas imagens sejam geradas também a uma alta velocidade, exigindo disponibilidade da memória para o acesso através da CPU. O fundamental deste conflito é como o CRTC irá fornecer os dados à lógica de saida de video, quando o tempo de acesso à memória é maior do que o tempo do pixel. Para a solução deste impasse, deveremos desenvolver a interface com a CPU visando a maior disponibilidade possivel para o acesso da memória pelo processador de vídeo.

Com estas restrições analisaremos a organização de uma memória para um terminal gráfico monocromático com 512 linhas de 512 pixels funcionando a 30 quadros por segundo entrelaçados. Esta memória seria implementada com dispositivos de memória RAM dinâmica de 16K x 1 e para sua realização são necessários 16 destes componentes organizados como mostra a figura 3.9. Na Tabela 3.2 temos algumas caracteristicas para terminais monocromáticos com várias resoluções (os tempos são os mesmos para os terminais coloridos, alterando-se somente o número de componentes). Para o

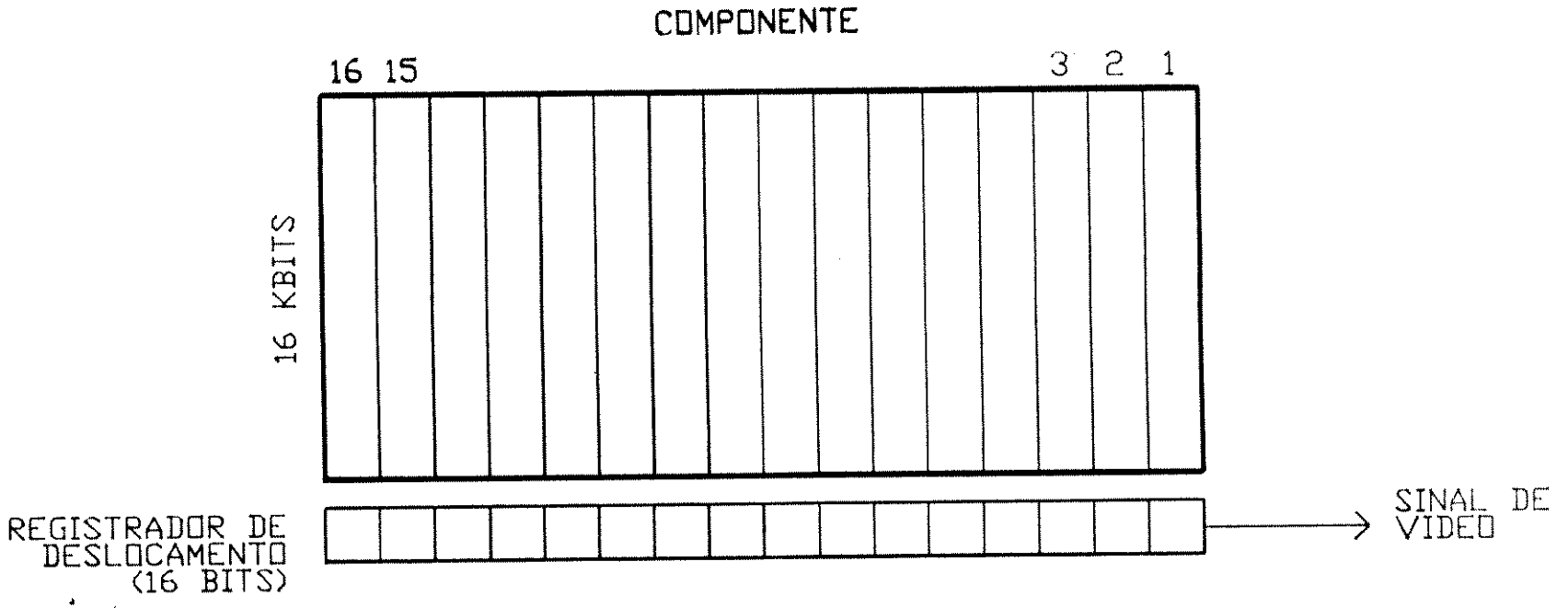


Figura 3.9 — Memoria de Video de Um Terminal com Resolucao de 512 x 512 Pixels

TABELA 3.2

AREA VISIVEL PIXELS x LINHA	TAXA DE REFRESH	ENTRE-LACADO	TEMPO DE RETRACO VERTICAL (us)	TEMPO DE RETRACO HORIZONTAL (us)	TEMPO TOTAL DA LINHA (us)	TEMPO DO PIXEL (ns)
512x485	30	SIM	1271	10.9	63.56	102.8
640x485	30	SIM	1271	10.9	63.56	82.3
512x512	30	SIM	1203	10.9	60.4	96.7
1024x768	30	SIM	1250	7	40.1	32.37
1024x1024	30	SIM	1250	7	30.11	22.57
1280x960	30	SIM	1250	7	32.12	19.62
1280x1024	30	SIM	1250	7	30.11	18.06
512x485	60	NAO	1250	7	31.79	48.41
640x485	60	NAO	1250	7	31.79	38.73
512x512	60	NAO	1250	7	30.11	45.14
1024x768	60	NAO	600	4	20.92	16.52
1024x1024	60	NAO	600	4	15.69	11.42
1280x960	60	NAO	600	4	16.74	9.95
1280x1024	60	NAO	600	4	15.69	9.13

terminal do exemplo acima temos os seguintes tempos:

. retraço vertical 1203 us

. retraço horizontal 10,9 us

. tempo da linha 60.4 us

. tempo do pixel 96,7 ns

Para os componentes de memória, valores da ordem de 400 ns são necessários para o acesso. Como o tempo para cada pixel é da ordem de 100 ns, temos a necessidade de obter no mínimo 4 pixels a cada acesso à memória. Por outro lado, como são necessarios 16 componentes para a memória, podemos então organizar a memória como mostra a figura 3.10. Com esta estrutura, a cada acesso do CRTC, 16 bits de dados são transferidos para o registrador de deslocamento, o que significa a obtenção de informação para o

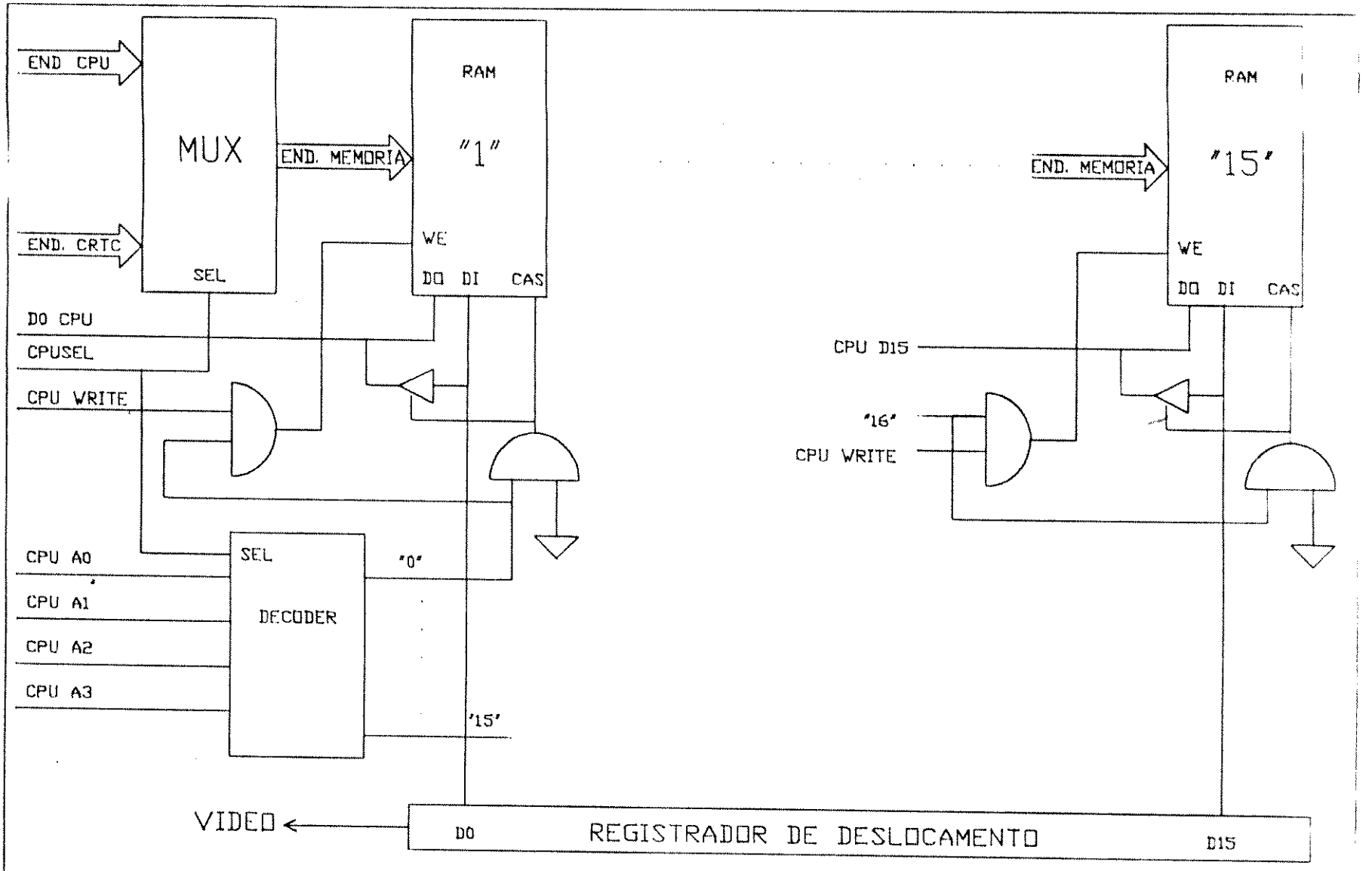


Figura 3.10 – Organizacao da Memoria de Video para um Terminal Monocromatico de 512 x 512 Pixels

monitor por um periodo igual a 1600 ns. Como o tempo de acesso à memória é da ordem de 400 ns, isto implica que a cada 1600 ns dispõe-se de 1200 ns para o acesso à memória pela CPU, resultando em uma disponibilidade de 75%. Na Tabela 3.3, estão resumidos os valores de disponibilidade obtidos utilizando-se componentes de memória com diferentes capacidades. Ressalta-se na Tabela 3.3 uma disponibilidade de memória de 87,5% obtida com a utilização de componentes 4K x 1 e registrador de deslocamento de 64 bits. A figura 3.11 mostra a distribuição do tempo disponivel para acessos da CPU e do CRTC para uma linha de raster.

Conclue-se também dos resultados da Tabela 3.3 que à medida que aumenta a capacidade de memória mantendo-se a largura de 1 bit por palavra, a porcentagem do ciclo de memória usada pelo CRTC aumenta. Na Tabela 3.3 tem-se para um terminal de 1024 x 1024 pixels, utilizando-se componentes de 64 K x 1, a taxa de uso pelo CRTC é 100% considerado a taxa de regeneração da imagem de 30 quadros por segundo. No caso de componentes de 256K x 1 sómente 4 componentes são necessários para a realização do terminal, porém não sobra tempo para a atualização da memória pela CPU. Um único componente de 256K x 1 pode armazenar um plano de imagem de 512 x 512 pixels, mas não poderá ser usado uma vez que não atende aos requisitos de tempo de acesso de aproximadamente 100 ns necessário para manter a regeneração da imagem no video. Nestas condições resta ainda o recurso das atualizações da memória de video serem executadas durante os tempos de retraço horizontal e vertical.

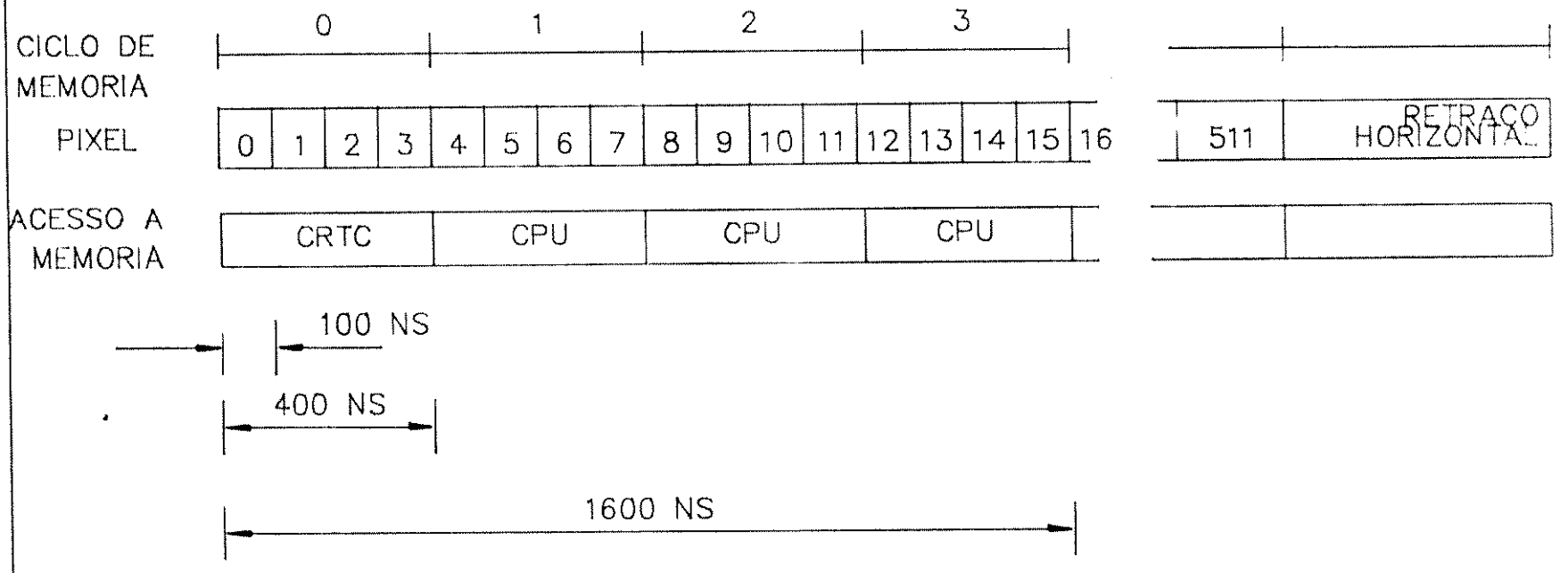


Figura 3.11 – Distribuicao do Tempo de uma Linha de Raster entre o Controlador e o Processador de Video

TABELA 3.3

			30 HZ ENTRELACADOS 100 NS POR PIXEL		60 HZ NAO ENTRELACADOS 45 NS POR PIXEL	
TAMANHO DO COMPONENTE	NUMERO DE COMPONENTES	NUMERO DE PIXELS POR ACESSO	TEMPO ENTRE ACESSOS DO CRTC	% USADA PELO CRTC	TEMPO ENTRE ACESSOS DO CRTC	% USADA PELO CRTC
4 K x 1	64	64	6400	12,5	2280	14
16 K x 1	16	16	1600	25	575	64
64 K x 1	4	4	400	100	180	*
256 K x 1	1	1	100	*	45	*

Podemos observar também na Tabela 3.2 que apesar do tempo de um pixel em um sistema de 640 X 485 ser menor do que aquele dos sistemas de 512 X 512, o ciclo de acesso à memória é maior. Neste último, são necessários 96,7 ns por acesso, ao passo que no primeiro, são necessários 82,3 ns. No primeiro são armazenados 5 bits a cada acesso, contra 4 bits no segundo, o que nos dá 388 ns contra 412 ns. Estes dois tempos poderão ser atendidos pelos componentes de 64K x 1 (tempo de acesso da ordem de 300 ns). Isto nos leva à conclusão de que quanto maior o número de bits transferidos simultaneamente aos registradores de deslocamento, maior será a disponibilidade da memória para acessos pela CPU.

Para a realização de terminais coloridos, torna-se necessária a inclusão de outros planos de memória, obtendo-se então uma estrutura como a da figura 3.12. E importante ressaltar que neste caso os tempos de linha, retraços, etc. são semelhantes aos dos

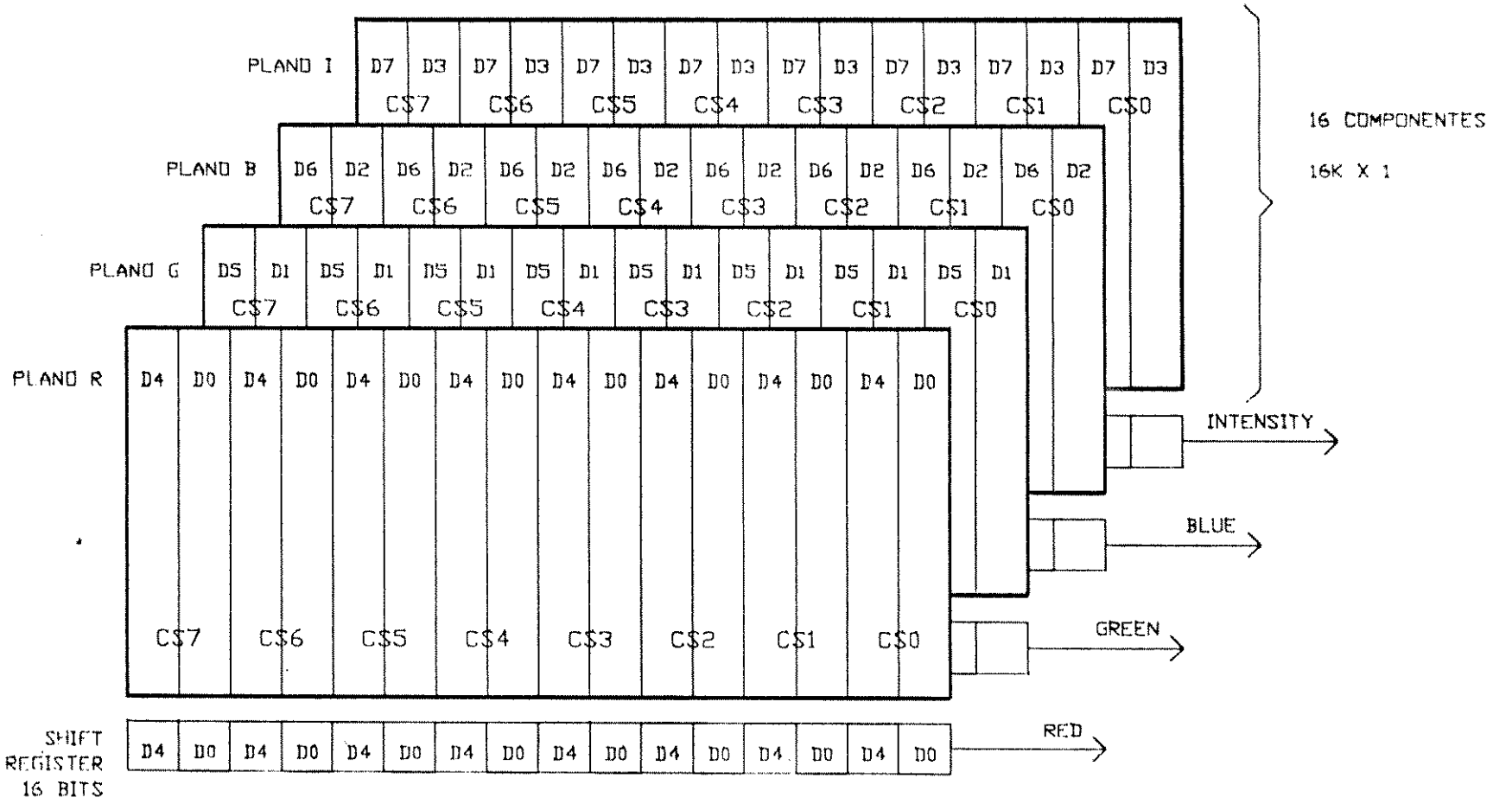


Figura 3.12 – Configuracao da Memoria de Video para um Terminal de 512 x 512 e 4 Planos

terminais monocromáticos.

Como já exposto anteriormente, o acesso à memória de vídeo pelo CRTC obedece um padrão de tempo bem determinado, fazendo com que este tenha prioridade de acesso à memória, obrigando portanto que técnicas especiais de sincronização sejam desenvolvidas para o partilhamento da memória entre o CRTC e a CPU.

Nota-se também que, quanto maior a largura dos dados tranferidos aos registradores de deslocamento, maior é a disponibilidade da memória para as atualizações pela CPU. Este fato faz com que as vídeo RAM´s tornem-se altamente úteis, uma vez que a presença dos registradores internos de 256 bits, elevam a disponibilidade da memória para aproximadamente 95% (como será mostrado no Capítulo 4).

3.4 Look-up table

Existe um recurso que permite que seja aumentado o número de cores na tela de um terminal gráfico sem que seja necessário aumentar o número de planos de memória. Esta técnica baseia-se no uso de uma memória de alta velocidade chamada "Look-up Table". Com a utilização desta técnica, os dados armazenados na memória de vídeo não são enviados diretamente aos circuitos analógicos de saida, porém vão a uma memória intermediária de alta velocidade. Este dado então serve como endereço dentro da memória intermediária ("look-up table"). A figura 3.13 mostra o exemplo de utilização da "look-up table". No caso da figura, o dado da memória de vídeo referente ao pixel é composto de 8 bits. Este dado serve como endereço para uma das 256 posições da tabela, onde estão armazenados 24 bits em cada posição. Estes 24 bits são divididos em três grupos de 8 bits cada, sendo um grupo para cada uma das cores primárias. Cada conjunto de 8 bits é então direcionado a um conversor D/A, ligados respectivamente às cores primárias vermelho, verde e azul do monitor de vídeo.

Deve-se notar ainda que num determinado instante, somente são possíveis 256 combinações de cores na tela. Alterando-se o conteúdo das posições da "look-up table", poderemos ter novas cores. Como cada posição armazena 24 bits, teremos um total de 2^{24} cores, porem somente 256 combinações simultaneas.

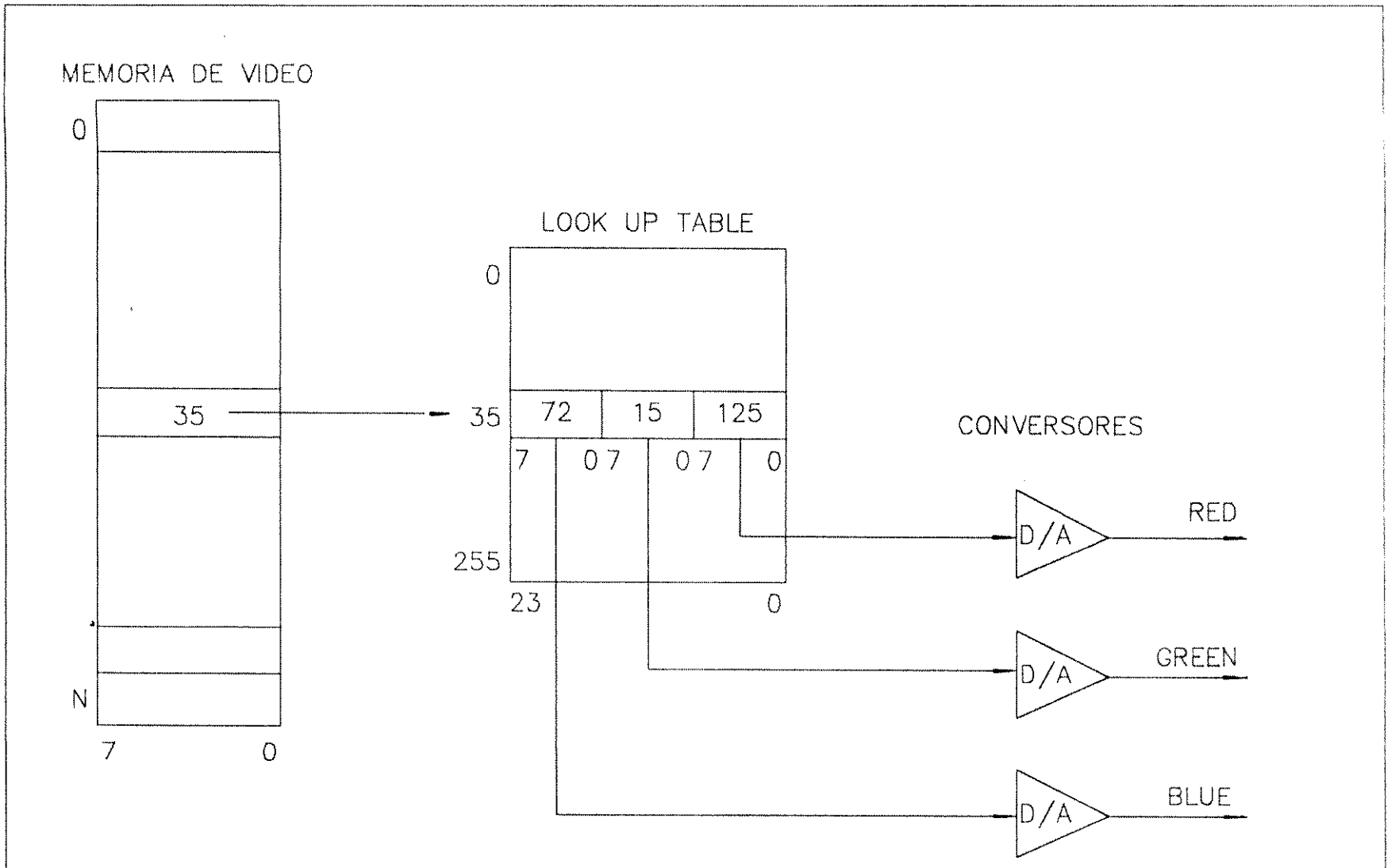


Figura 3.13 – Utilizacao de "Look-Up Table"

4. CIRCUITOS CONTROLADORES DE VIDEO

Os controladores de vídeo gráfico são dispositivos que incorporam fundamentalmente as funções para a geração do "timing" necessário à regeneração da imagem através de endereços de memória e sinais de sincronismo. Recentemente novas funções que facilitam a geração dos pixels correspondentes às primitivas vem sendo adicionadas a êles, tornando-os mais eficientes. Deste modo pode-se classificá-los em duas categorias básicas: controladores de "refresh" de tela e controladores com conjunto de funções fixas.

4.1. Controladores de "refresh" de tela

Os controladores de "refresh" de tela têm como função básica buscar as informações na memória de vídeo, apresentá-las aos circuitos de saída e gerar os sinais de sincronismo horizontal e vertical. Nos sistemas onde são utilizados este tipo de controlador a CPU coloca os dados gráficos (pixels) na memória de vídeo, ou seja, desenha as figuras enquanto o controlador retira estas informações colocando-as em formato apropriado aos circuitos analógicos de vídeo.

Podemos considerar dois tipos destes controladores: os dirigidos para operação no modo caractere e os dirigidos para o modo memória mapeada. Essas duas categorias em alguns controladores são comuns, permitindo sua utilização nos dois modos. Iremos analisar resumidamente os controladores 8275 da Intel e 6845 da

Motorola, que foram desenvolvidos basicamente para aplicações em terminais alfanuméricos, o controlador SCN2674 da Signetics, que apresenta características tanto para aplicações gráficas como alfanuméricas, e o controlador TMS34061 da Texas, que visa aplicações em terminais gráficos.

4.1.1. Controlador 8275 da Intel

O controlador de vídeo 8275 fabricado pela Intel permite que sejam projetados terminais de vídeo com várias organizações em termos de número de linhas e colunas, graças aos seus registros internos programaveis, que possibilitam esta flexibilidade. Além disso, o componente dispõe de dois buffers de 80 bytes cada, os quais amenizam os problemas de contenção de memória.

Por outro lado, o 8275 não dispõe de lógica de endereçamento de memória, que deverá ser fornecida por dispositivos externos (por exemplo, um controlador de DMA). A contenção de memória fica portanto resolvida através deste dispositivo. A função de "scroll" também deverá ser executada por uma lógica externa.

Outro aspecto a ser ressaltado com relação ao 8275 é sua facilidade para o interfaceamento com a CPU.

O 8275 possue registros internos que podem ser acessados pela CPU para orientar seu funcionamento e fornecer informações à CPU sobre seu estado atual. Existem três tipos básicos de registros: "status", comando e parâmetros.

Os comandos básicos são: reset, torna o sinal de vídeo ativo ou desativo, inicializa contadores, leitura do registro de "light-pen", carga do registro de posicionamento de cursor, habilitação ou não das interrupções.

Os registros de parâmetros permitem que sejam programados o número de caracteres por linha, o número de linhas de caracteres por quadro, o número de linhas de caracteres por retraço vertical, número de linhas de raster por linha de caracteres, número de caracteres por retraço horizontal, formato do cursor (piscante ou não, bloco ou sublinhado), número de requisições de DMA por tempo, quantas transferencias por ciclo de DMA.

Na figura 4.1 temos um diagrama simplificado de um terminal de vídeo alfanumérico utilizando o 8275.

Os sinais de saida do 8275 são: código do caractere, contador de linha de scan, sincronismo horizontal e vertical, supressão do sinal de vídeo ("blank"), cursor, indicador de vídeo reverso, indicador de alta intensidade e detecção de "light-pen".

O 8275 pode ser utilizado para aplicações em terminais gráficos, porém exigiria uma grande quantidade de lógica externa, tornando-o desaconselhável para este tipo de aplicação.

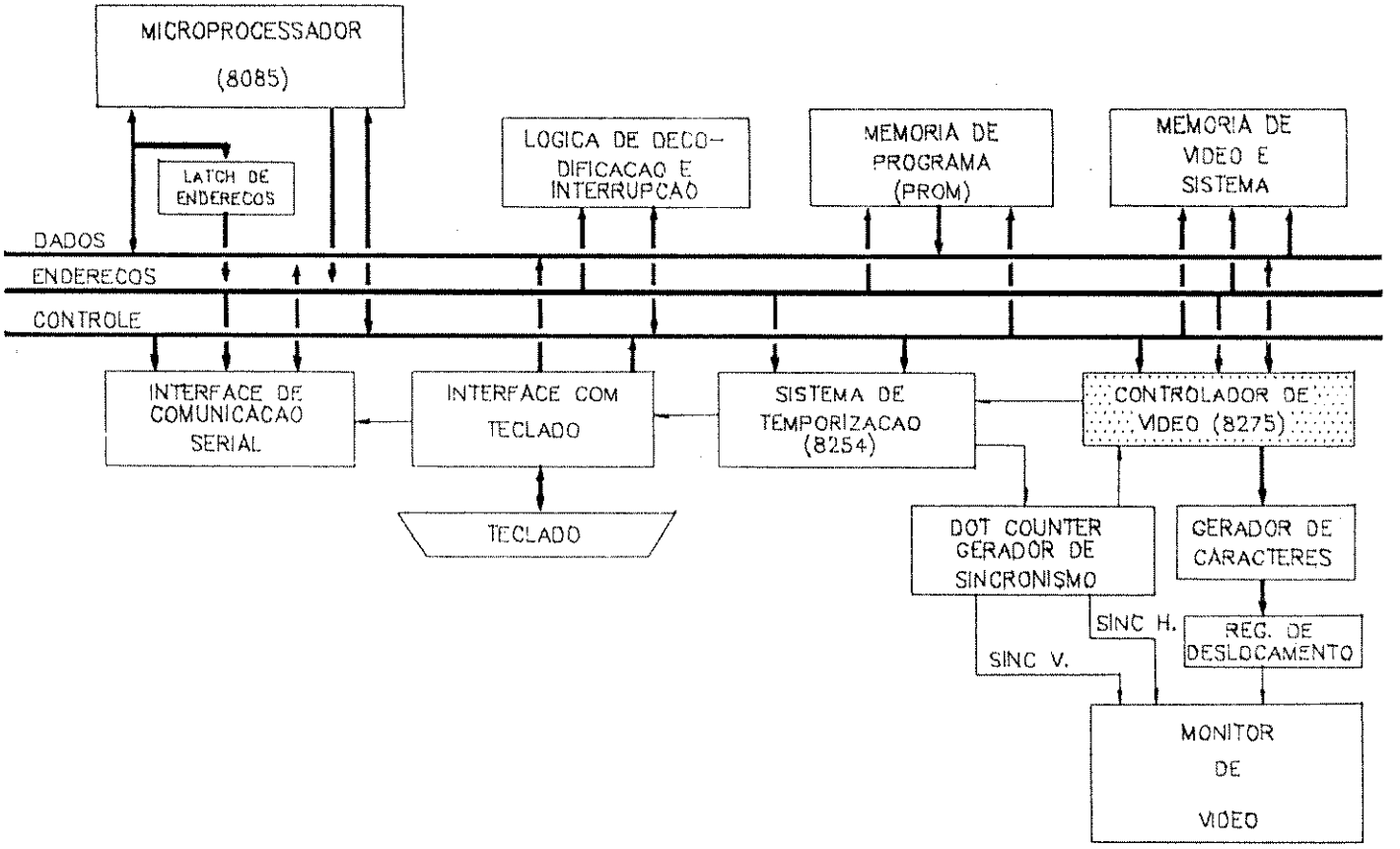


Figura 4.1 – Diagrama Simplificado de Um Terminal Alfanumerico Utilizando o 8275, Sem o Recurso do Controlador de DMA

4.1.2. Controlador 6845 da Motorola

O controlador de vídeo 6845 desenvolvido pela Motorola, é um componente que agrupa algumas funções requeridas por um terminal de vídeo. Apresenta lógica de endereçamento de memória, lógica de "scroll", geração dos sinais de sincronismo, lógica de "light-pen" e lógica de cursor.

Um dos pontos negativos apresentados por este componente é a ausencia de uma lógica de contenção de memória, obrigando que circuitos especiais para este fim sejam utilizados.

A interface do componente com a CPU é facilmente executada, sendo que com as CPU´s da família Motorola de 8 bits ela é direta.

Registradores internos permitem que os seguintes parâmetros sejam programados: número total de caracteres na linha (incluindo retraço horizontal), número de caracteres visiveis na linha, posição de início e duração do pulso de sincronismo horizontal, número total de linhas de caracteres por quadro, número de linhas de caracteres visíveis por quadro, posição e largura do pulso de sincronismo vertical, número de linhas de raster por linha de caracteres, endereço inicial da memória, endereço do cursor. Podem ainda ser programados modos entrelaçados ou não, formato do cúrsor e ajuste da frequencia vertical, adicionando-se tempo de linha de raster ao tempo do quadro. Existe um registro que armazena a informação sobre as coordenadas onde ocorreu a ativação do

"light-pen". Na figura 4.2 mostramos um diagrama de blocos simplificado de um terminal de vídeo alfanumérico baseado no controlador 6845 da Motorola.

Os sinais de saída do componente são: 14 linhas para endereço de memória, 5 linhas para endereço de linha de raster, sincronismo horizontal e vertical, habilitação do sinal de vídeo, cursor e "light-pen".

No capítulo 5 será apresentado um terminal gráfico colorido usando o 6845.

4.1.3. Controlador de vídeo avançado SCN2674 da Signetics

O controlador de vídeo avançado (AVDC) SCN2674 da Signetics é um dispositivo programável para ser utilizado tanto em terminais de vídeo gráficos como alfanuméricos. E capaz de gerar sinais de sincronismo horizontal e vertical bem como trabalhar nos modos entrelaçado ou não entrelaçado.

Entre suas principais características, podem ser destacadas: programação de 1 a 256 caracteres por linha, de 1 a 16 linhas de raster por linha de caractere, de 1 a 128 linhas de caracteres por quadro, modo "bit-mapped", programação dos geradores de sincronismo horizontal e vertical.

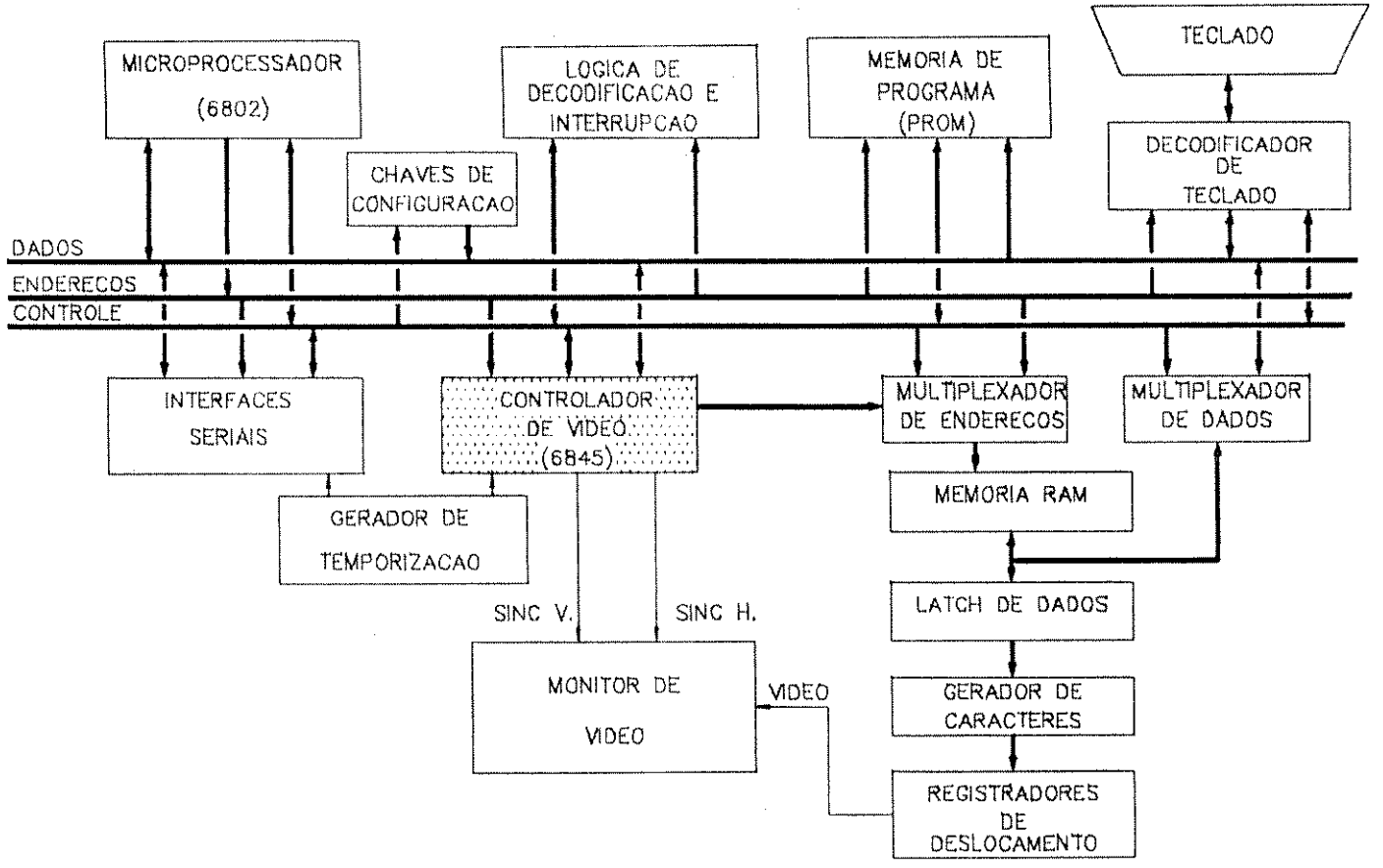


Figura 4.2 – Diagrama de Blocos Simplificado de Um Terminal de Video Alfanumerico Utilizando o Controlador 6845

Outra grande vantagem do 2674 é a possibilidade de subdivisão da tela em "fatias" horizontais chamadas "splits", onde cada uma destas está associada com porções da memória de vídeo, podendo conter informações alfanuméricas ou gráficas. Desta maneira é possivel obter-se uma região da tela com informações gráficas e outra região com informações alfanuméricas.

Com relação à resolução de conflitos no acesso à memória de vídeo, existem alguns mecanismos que podem ser utilizados visando atenuar o problema. No primeiro deles, chamado modo independente, é colocado um "latch" de um caractere, entre a CPU e a memória de vídeo. Quando a CPU necessita acessar a memória de vídeo para escrever um dado, ela envia este dado ao "latch", e escreve um comando no controlador. Este se encarrega de transferir o dado do "latch" para a memória assim que esta estiver disponível. A transferencia no sentido oposto, da memória para a CPU, é executada de maneira semelhante, com a CPU solicitando o dado ao controlador, e este o transfere para o "latch" quando a memória estiver disponível para o acesso. Esta disponibilidade da memória nos dois casos corresponde aos periodos de "blank" do sinal de vídeo.

Existem também os modos "buffer" partilhado e transparente, nos quais a memória de vídeo é parte do endereço normal da CPU. Nestes modos a CPU acessa a memória de vídeo via "buffers tri-state". A CPU informa ao controlador que necessita transferir dados e em resposta a este pedido o controlador libera um sinal avisando que está ocupado, permanecendo assim até que seja possí-

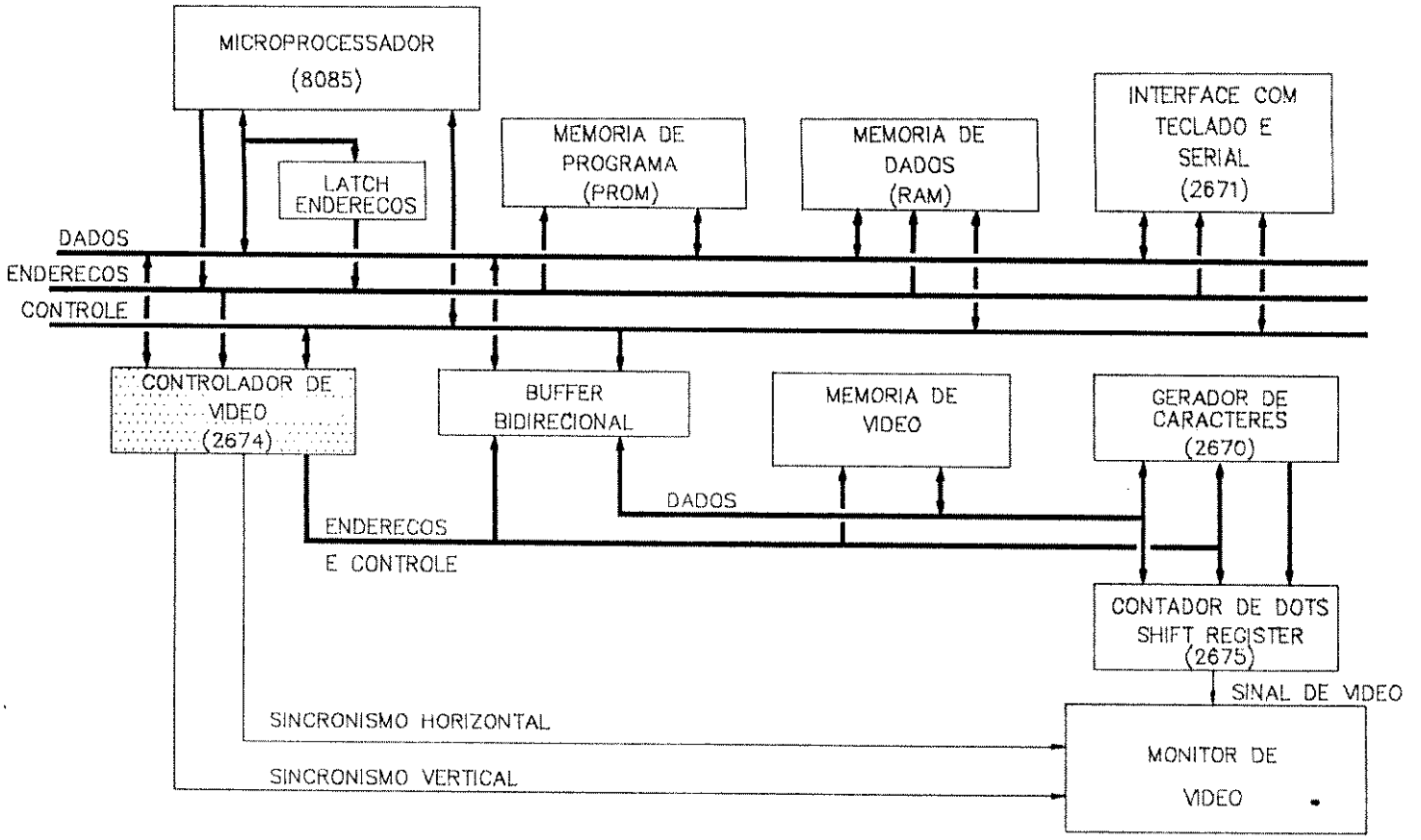
vel a transferencia durante o periodo de "blank".

Outra maneira de trabalho do 2674 é o modo "row", no qual é utilizado um "buffer" intermediário de linha de caracteres. Assim, durante a primeira linha de raster de cada linha de caracteres, são transferidos via DMA, os códigos dos caracteres daquela linha para o "buffer", liberando portanto a memória durante as próximas linhas de raster.

Na figura 4.3 é apresentado um diagrama simplificado de um terminal de vídeo alfanumérico utilizando o SCN2674.

4.1.4. Controlador de sistema de vídeo TMS34061 da Texas

O controlador de sistema de vídeo TMS34061 da Texas é um dispositivo que controla o vídeo e memórias de um sistema de vídeo gráfico "bit-mapped". Apesar de visar fundamentalmente o controle de dispositivos do tipo Vídeo RAM´s, é também compatível com memórias tipo estática e dinâmicas convencionais. Além disso, sua interface com a CPU é implementada facilmente, já que dispõe de lógica que o torna capaz de interfacear com barramentos tanto síncronos como assíncronos. Com estas características, torna-se um elemento com alto desempenho para projetos de terminais de vídeo gráficos. Suportando resoluções de até 4096 x 4096 pixels, apresenta ainda a característica de ser um controlador de "refresh" para memórias dinâmicas, gerando os endereços multiplexados, sinais de RAS e CAS, lógica de controle para carga dos registra-



Figrua 4.3 – Diagrama de Blocos Simplificado de Um Terminal Alfanumerico Utilizando os Componentes da Familia do do Controlador 2674

dores de deslocamento presentes nas VRAM's, sinais de sincronismo horizontal e vertical, e sinal de habilitação do sinal de vídeo.

Como a regeneração da imagem em terminais que usam memórias do tipo VRAM requer uma porcentagem pequena do tempo total da memória e como o TMS34061 executa a função de regeneração da imagem automaticamente, pode-se ainda utilizá-lo para o controle de RAM dinâmica utilizada pela CPU.

O TMS34061 possui registradores internos com as finalidades já expostas para o 6845 e, além disso, possui registradores X e Y, que permitem que pixels sejam acessados por suas coordenadas e não pelo seu endereço de memória. Esta facilidade permite que algoritmos de traçado de linha sejam implementados com grande facilidade.

Como este controlador foi utilizado no terminal desenvolvido, mostrado no capítulo 5, analisaremos alguns detalhes importantes deste componente. Sua estrutura interna é apresentada na figura 4.4

O TMS34061 pode ser programado para executar ciclos de "refresh" de dispositivos tipo RAM dinâmicas, através de 3 bits presentes no Registro de Controle 1, os quais determinam o número de ciclos de "refresh" executados durante uma linha de raster. Este número pode variar de 0 a 7. Durante o ciclo de "refresh" o controlador libera um conjunto de 9 linhas de endereço que serão

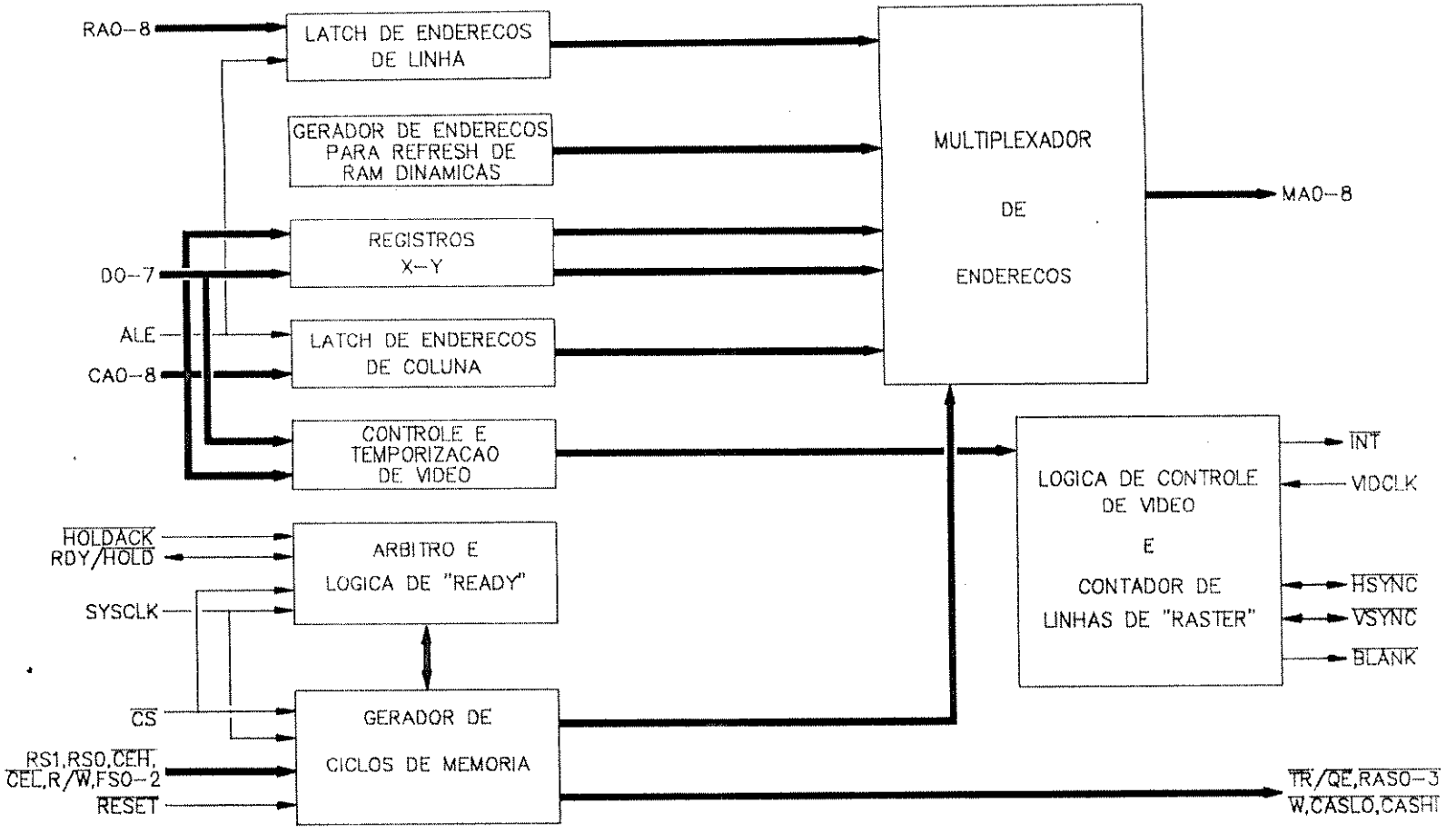


Figura 4.4 – Diagrama de Blocos da Arquitetura do Controlador TMS34061 da Texas

utilizadas em conjunto com o sinal RAS (row strobe address) para a função de regeneração das informações contidas em uma das linhas do arranjo da memória dinâmica.

Outra caracteristica importante do 34061 é a capacidade de endereçamento X-Y, que oferece a possibilidade de que CPU´s com capacidade reduzida de endereçamento sejam capazes de acessar todos os pontos da tela. O conteúdo dos registros X e Y substituem as linhas de endereços RA e CA da entrada do componente, como fonte de endereços para a memória. Outra possibilidade é o ajuste dos endereços X e Y após o acesso (incremento, decremento, sem alteração, e zerado). Com esta técnica é possível formar um endereço de 20 bits.

Outro bloco importante é o árbitro, que determina quando a CPU, a lógica de carga dos registradores de deslocamento internos à vídeo RAM ou a lógica de "refresh" podem acessar a memória de vídeo. Uma vez que a regeneração da imagem e os ciclos de "refresh" não consomem mais que 6% dos ciclos de memória, o árbitro normalmente garante o acesso à CPU. Quando existe conflito, o árbitro atribui as seguintes prioridades: qualquer ciclo que esteja ocorrendo, regeneração da imagem, ciclo de "refresh" atrasado mais de 1/2 linha horizontal, ciclo requisitado pela CPU e "refresh" de RAM dinamica.

Outra função do árbitro é determinar qual a maneira do TMS34061 sinalizar a CPU sobre seu estado atual. Existem três modos: "ready" síncrono, "wait", e "hold/hold acknowledge". Cada

um deles se torna mais apropriado a determinados tipos de CPU.

Existe ainda o gerador de ciclos de memória, que irá gerar os sinais de interface com os componentes de memória de acordo com os ciclos seguintes: acesso direto da CPU à memória, acesso indireto da CPU via registros X-Y, acesso da CPU para tranferir dados das VRAMs aos registradores de deslocamento, acesso da CPU para tranferir dados dos registradores de deslocamento para as VRAMs, ciclos gerados internamente para a regeneração da imagem (transferencia das VRAMs para os registradores de deslocamento), e ciclos de "refresh".

Os registros internos para programação são os seguintes:

. final do sincronismo horizontal,
. final do blank horizontal,
. início do blank horizontal,
. total horizontal,
. final do sincronismo vertical,
. final do blank vertical,
. início do blank vertical,
. total vertical,
. registros de controle 1 e 2,
. registro de endereço X-Y,
. registro de offset X-Y,
. registros de inicio de memória.

4.2. Controladores com conjunto de instruções fixas

Os controladores com conjunto de instruções fixas são dispositivos que, além das funções normais aos controladores de "refresh" de tela, apresentam ainda um conjunto de funções para a geração de primitivas gráficas. Estes controladores vêm se tornando a cada dia mais sofisticados, podendo ser considerados verdadeiros computadores de vídeo.

Apresentaremos as características resumidas dos seguintes controladores desta categoria: 7220 da NEC, HD63484 da Hitachi e TMS34010 da Texas.

4.2.1. Controlador 7220 da NEC

O primeiro controlador de vídeo com conjunto de intruções fixas disponível no mercado foi o NEC 7220. Incluindo lógica para regeneração de imagem, controle de "refresh" para RAMs dinâmicas, lógica para geração de primitivas gráficas e interface para "light-pen", este dispositivo é capaz de gerar primitivas tais como retas, arcos, retângulos, caracteres gráficos e preenchimento de área, as quais são transferidas como blocos de 8 x 8 pixels para a memória de vídeo e podem ser utilizadas como padrão para preenchimento de áreas. Executa também funções de zoom e pan. Sua velocidade para a geração de primitivas é de aproximadamente 500 ns por pixel.

O fator que limita este componente é o fato de ele não poder

executar regeneração de imagem e geração de primitivas ao mesmo tempo, uma vez que ele não possui lógica eficiente para a resolução dos conflitos de acesso à memória, nem tampouco dispõe dos sinais para interfaceamento com dispositivos do tipo VRAM. Se houver necessidade de acesso à memória durante o ciclo de regeneração, a tela apresentará "chuvisco". Caso este efeito seja indesejável, a velocidade de formação das primitivas será bastante menor, uma vez que só será possivel o acesso durante os tempos de retraço tanto horizontal como vertical.

A capacidade de memória de vídeo é de 512 Kbytes, o que permite uma resolução de 2048 x 2048 pixels com 1 bit cada. A imagem pode ainda ser gerada pela transferencia de dados da memória do processador para a memória de vídeo via DMA.

O 7220 pode dividir a tela em duas áreas (split), sendo cada uma delas especificada por um registro de início e um de tamanho em linhas. Se a segunda área não for utilizada, a primeira deve ser especificada como tendo a área de vídeo total.

Na figura 4.5 apresentamos um diagrama simplificado da estrutura interna do 7220.

4.2.2. Controlador HD63484 da Hitachi

O controlador de vídeo HD63484 (ACRTC - Advanced CRT Controller) desenvolvido pela Hitachi surgiu como uma evolução natural

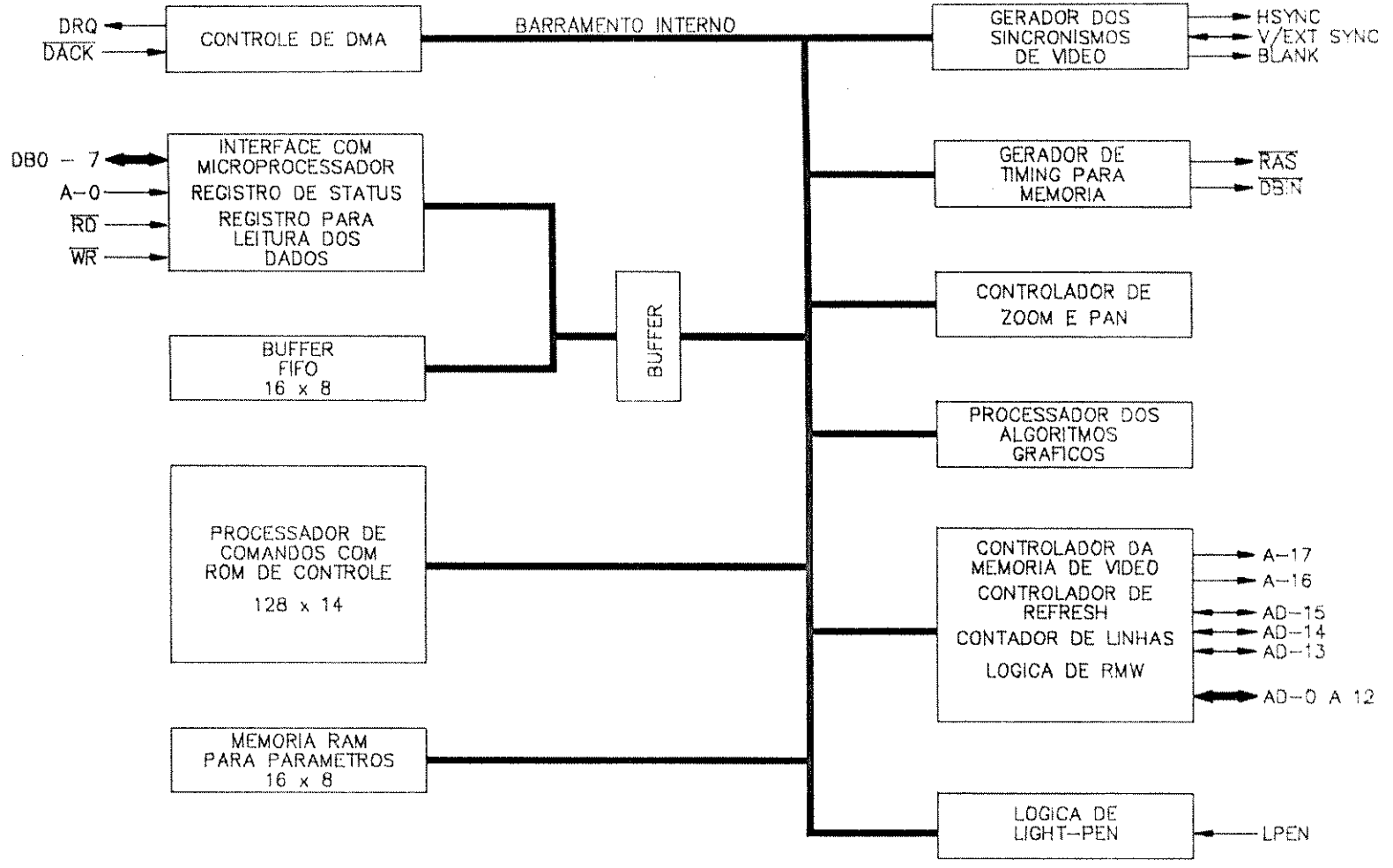


Figura 4.5 – Diagrama Interno do Controlador NEC 7220

do 7220. Este componente utiliza técnicas de "interleaving" para o acesso à memória, ou seja, o ciclo da memória é dividido em dois: um para a regeneração da imagem, e outro para a geração de pixels. Isto implica, para a manutenção do mesmo "bandwidth", que o tempo de acesso à memória seja reduzido à metade, com consequencias na arquitetura da memória.

Este controlador permite a utilização de uma memória de até 2 Mbytes que, dependendo do número de planos, possibilita a definição de uma memória de vídeo que varia de 4096 x 4096 pixels para um único plano a 1024 x 1024 pixels para 8 planos. Permite ainda a utilização de 128 Kbytes de memória para dados alfanuméricos.

Possui ainda: capacidade de controle do display, permitindo o uso de 4 telas independentes, controladas a partir de registradores de split; capacidade de zoom (1 a 16 vezes); e capacidade de "scroll" vertical e horizontal.

Este controlador permite a utilização de três tipos de cursor: cursor convencional para caracteres (bloco); cross-hair; e cursor gráfico (definido pelo usuário). O tipo de cursor é selecionado por programação.

Internamente este processador gráfico possui da ordem de duzentos bytes de registradores e memória RAM utilizados para sua programação. Esta programação inclui definição dos tempos de sincronismo, blank, retraço, entrelaçamento, etc. e registradores

de split, cursor e geração de primitivas gráficas.

As primitivas implementadas pelo HD63484 incluem: retas, retângulos, polilinhas, polígonos, círculos, elipses, arcos, retângulos preenchidos, paint, dot, graph copy, pattern.

Na figura 4.6 apresentamos um diagrama simplificado da estrutura interna do controlador HD63484.

4.2.3. Controlador TMS34010 da Texas

O controlador TMS34010 (GSP - Graphics System Processor) desenvolvido pela Texas é baseado em um processador de 32 bits tipo RISC (Reduced Instruction Set Computer), capaz de executar 6 milhões de instruções por segundo (MIPS). Incorpora em sua arquitetura um "Barrel Shifter" para executar rotação de bits em alta velocidade, uma estrutura capaz de executar as operações "Raster Op", controlador de sinais de vídeo, controlador para dispositivos do tipo VRAM. A memória local pode ser qualquer combinação de dispositivos VRAM, RAM dinâmica, RAM estática, ou ROM.

O componente dispõe de uma memória tipo "cache" de 256 bytes, a qual acelera a execução de algoritmos gráficos, uma vez que "loops" curtos podem estar contidos nesta memória.

Pode-se ainda combinar múltiplos GSP's, com finalidades diferentes (por exemplo, um para o controle de cada uma das cores básicas).

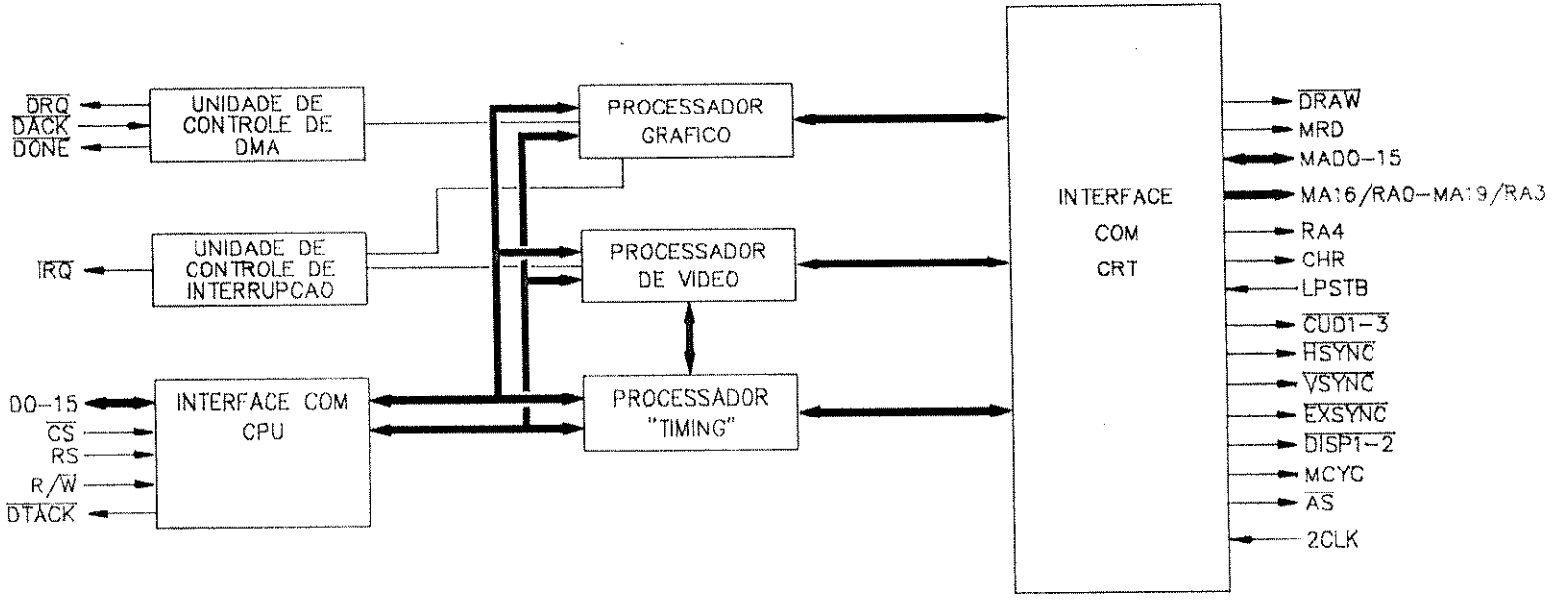


Figura 4.6 – Diagrama de Blocos do Controlador HD63484 da Hitachi

O TMS34010 é capaz de suportar tanto endereçamento linear como XY, sendo o linear útil para o gerenciamento dos programas e dados, e o XY mais conveniente para o tratamento dos dados gráficos e gerenciamento de janelas e outras funções de tela. O GSP possue 30 linhas de endereço externo (32 internamente), capaz de gerenciar 4 Gbits.

A estrutura interna foi otimizada para algoritmos gráficos, utilizando-se de técnicas como o processamento paralelo de pixels.

Seus registradores gráficos internos permitem que sejam programados todos os parâmetros de vídeo, bem como máscaras para cores, condições para interrupções, tamanho de pixels. Além dos registradores para uso gráfico, possui ainda registradores de uso geral, contador de programa (PC), apontador de pilha (SP), os quais permitem seu funcionamento autônomo, uma vez que possue um conjunto de 127 intruções de uso geral, com adaptações para aritmética de ponto flutuante, intruções gráficas e gerenciamento de janelas.

Na figura 4.7 apresentamos um diagrama simplificado da estrutura interna do controlador/processador TMS34010.

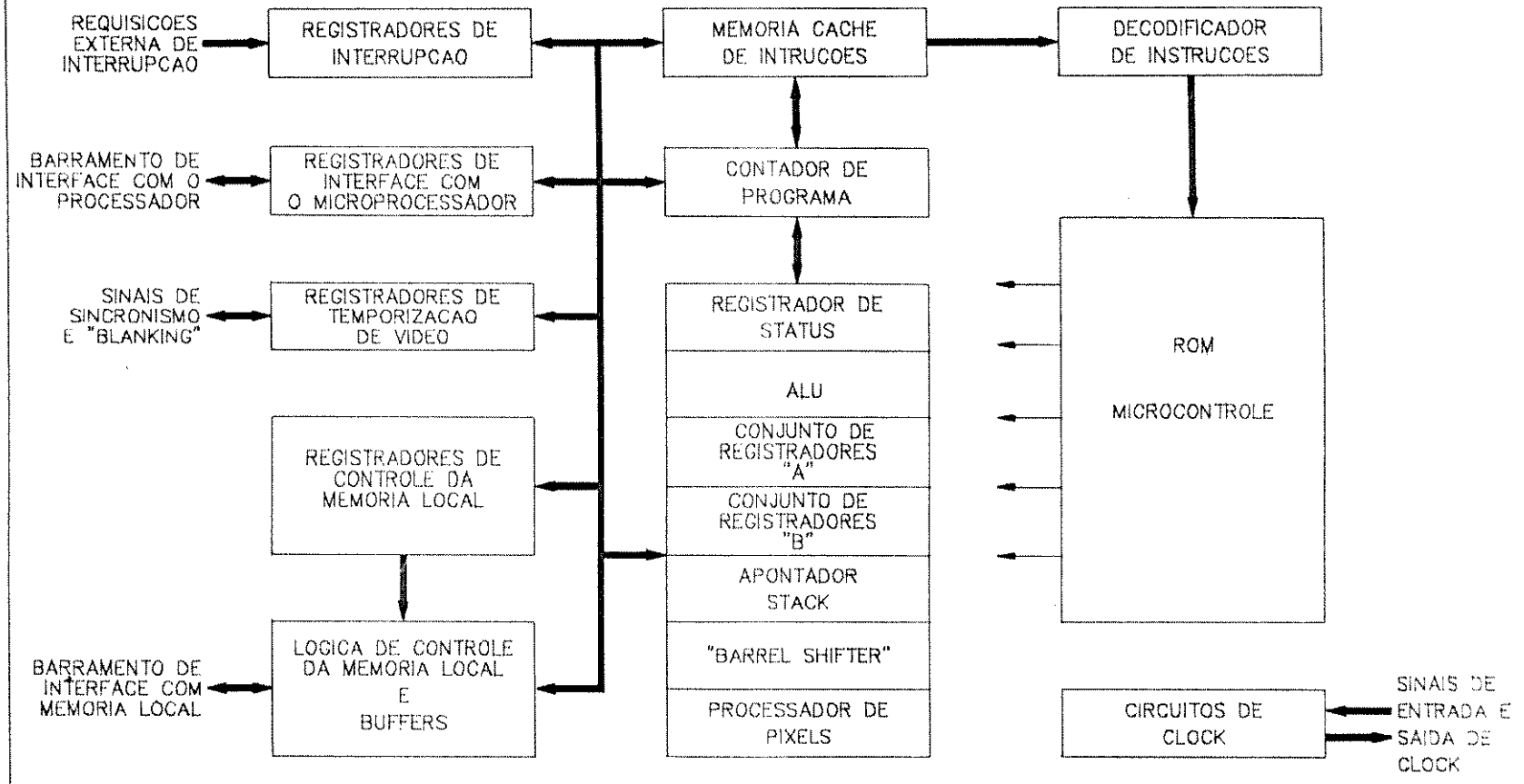


Figura 4.7 – Diagrama de Blocos do Processador TMS34010 da Texas

4.3. Aspectos a serem considerados na utilização de controladores de vídeo

Na utilização dos controladores de vídeo descritos anteriormente para a implementação de terminais gráficos, alguns aspectos deverão ser considerados. Os itens resolução, velocidade e custo, irão indicar uma das três possibilidades: controlador alfanumérico, controlador gráfico ou processador de vídeo. A escolha do controlador é feita baseada em parâmetros que são importantes para a arquitetura do sistema. Vamos analisar alguns destes parâmetros

4.3.1. Endereçamento

A quantidade de linhas de endereços apresentadas pelo controlador indica a resolução máxima possivel para o terminal nele baseado. Os controladores específicos para aplicação gráfica tem como caracteristica o endereçamento linear, ou seja, as posições de memória são consideradas em sequencia linear, ao passo que os controladores voltados para aplicação em terminais alfanuméricos apresentam um conjunto de linhas de endereços lineares e um conjunto de endereços para linhas de raster dentro do caractere. Neste caso torna-se necessária uma combinação destes dois conjuntos para a formação de endereços lineares necessários para aplicação em terminais gráficos.

4.3.2. Frequência de operação

Os contadores internos dos controladores são atualizados através de um sinal de "clock" de caracteres, o qual possui uma frequência máxima de operação. Este fato deve ser considerado ao se projetar a velocidade com que os dados são transferidos da memória de vídeo para os registradores de deslocamento. Nos controladores alfanuméricos, a frequência de caracteres indica a taxa com que estes são transferidos para o gerador de caracteres. Quando se utiliza controladores alfnuméricos para aplicações gráficas, esta frequência determina a taxa máxima de transferência de um grupo de bits aos registradores de deslocamento. Deve-se obter um compromisso entre a frequência de caracteres e o número de bits transferidos aos registradores de deslocamento, isto é, uma arquitetura de memória que permite a transferência de maior núnero de bits simultaneamente.

4.3.3. Programação do controlador

Outro aspecto importante na escolha de um controlador é a facilidade de programação dos diferentes parâmetros de vídeo. Deverão ser programáveis os seguintes itens:

a. número de pontos ou caracteres por linha de raster
b. número de linhas de raster ou caracteres por tela
c. endereço inicial da memória de vídeo onde serão buscados os dados que aparecerão na primeira linha de raster
d. programação da duração e posição dos sinais de sincronismo horizontal e vertical, visando a utilização de diferentes

modelos de monitores de vídeo disponíveis

4.3.4. Mecanismos para resolução de conflitos

Existem três técnicas basicas para resolução de conflito no acesso à memória de vídeo. No primeiro, a memória de vídeo é acessada pela CPU apenas durante os tempos de retraço horizontal e vertical, período em que o controlador não acessa a memória de vídeo para a regeneração da imagem. No segundo, o acesso pela CPU ocorre intercaladamente com o acesso do controlador. O terceiro caso, consiste na utilização de buffers de linha visando reduzir a necessidade de acesso pelo controlador, utilizando-se então técnicas de DMA ou interrupções; este último modo é de pouco interesse no caso de terminais gráficos.

4.3.5. Geração de primitivas

Com relação à geração de primitivas gráficas podemos considerar duas configurações básicas: na primeira, utilização de uma CPU convencional e um controlador de regeneração de imagem, e na segunda, utilização de um processador gráfico com conjunto de instruções fixas e função de regeneração embutida.

No primeiro caso, teremos a estrutura do terminal conforme mostra a figura 4.8, onde se nota que a CPU acessa diretamente a memória para a geração das primitivas gráficas, ao passo que o controlador acessa diretamente esta mesma memória para a regene-

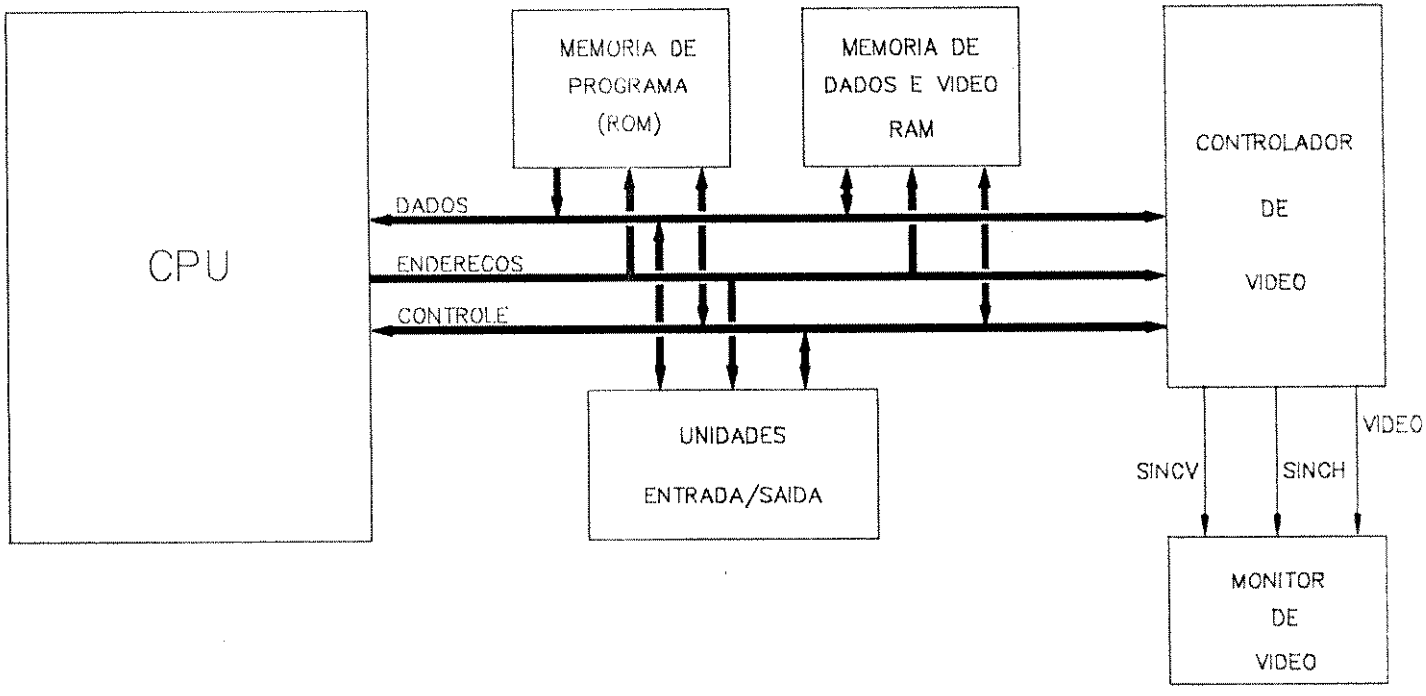


Figura 4.8 – Exemplo de Um Terminal de Video no Qual a Memoria de Video e Partilhada pelo Processador e Contolador

ração da imagem. Já no segundo caso, um terminal cuja estrutura é mostrada na figura 4.9, as primitivas gráficas são geradas diretamente pelo controlador de vídeo que, neste caso, também é o responsável pela regeneração da imagem. Neste caso a CPU não tem acesso direto aos dados armazenados na memória de vídeo.

Devemos salientar que no primeiro caso teremos uma maior flexibilidade quanto ao conjunto de primitivas gráficas em prejuízo da velocidade de conversão da primitiva em pixels. No segundo caso teremos maior velocidade na conversão da primitiva em pixels porém as funções gráficas são reduzidas às funções disponíveis no controlador, sendo, portanto, um conjunto fixo. Obviamente outras primitivas gráficas não constantes do conjunto fixo poderão ser geradas a partir das primitivas existentes (por exemplo, "set pixel"), porém, o desempenho em relação ao caso anterior deve ser menor.

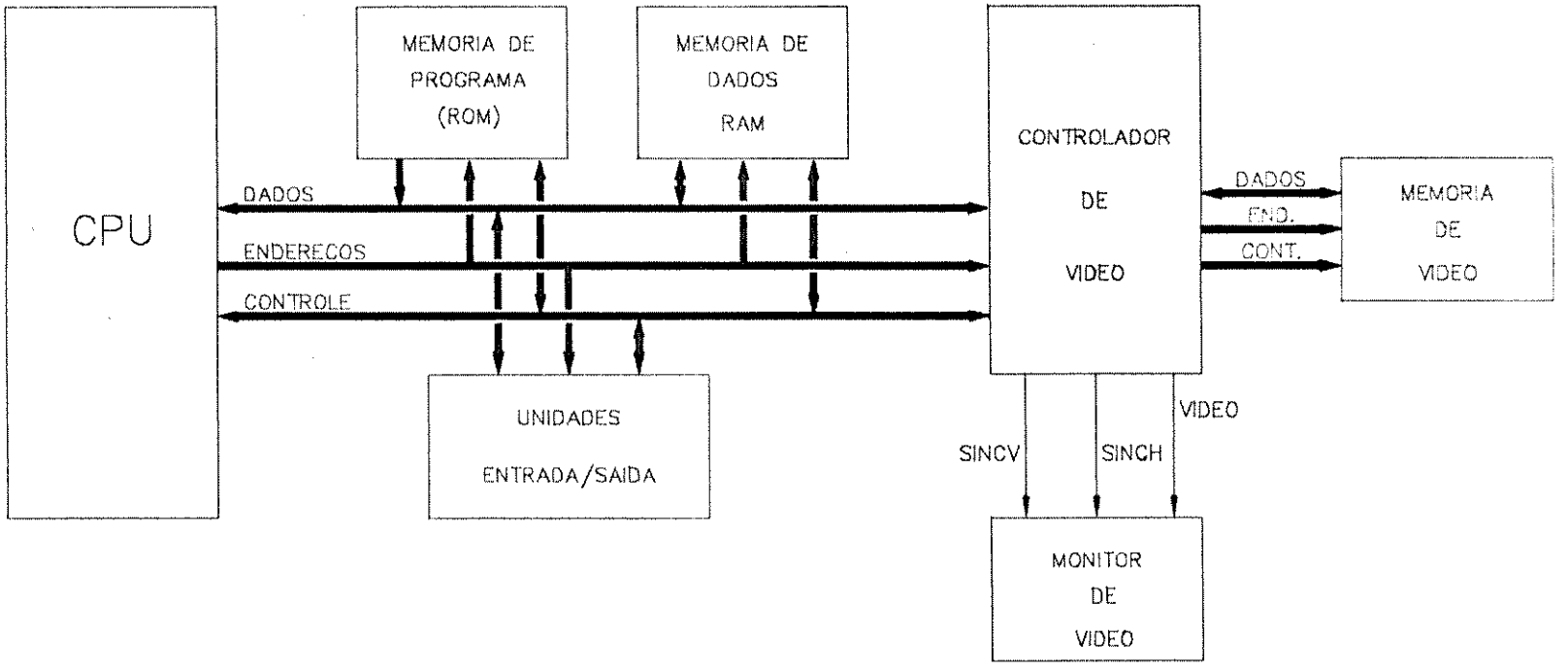


Figura 4.9 — Exemplo de Arquitetura de Terminal de Video no qual a CPU nao tem Acesso Direto a Memoria de Video

5. IMPLEMENTAÇAO DE TERMINAIS GRAFICOS

Visando compreender melhor os problemas encontrados no desenvolvimento de "hardware" para terminais gráficos e analisar o desempenho de algoritmos utilizados em computação gráfica, foi realizado um primeiro projeto, no qual foi possível observar e estudar os aspectos discutidos nos capítulos 3 e 4. Em seguida, com base nestes resultados, passou-se a um segundo projeto que permitisse a obtenção de melhor desempenho e utilização de componentes de memória mais sofisticados, no caso as vídeo RAM's. Na sequência apresenta-se a descrição dos terminais desenvolvidos.

5.1. Terminal gráfico colorido de baixa resolução

Este projeto teve por finalidade a obtenção de uma estrutura que permitisse o estudo dos problemas encontrados no desenvolvimento de placas gráficas, sendo que o custo e o tempo para o seu desenvolvimento deveriam ser reduzidos. O sistema gráfico desenvolvido usa uma CPU 68000 e um controlador Motorola 6845, componentes facilmente encontrados no mercado. A arquitetura da CPU, com registradores internos de 32 bits e barramento de dados de 16 bits, possibilitou que fossem facilmente implementados alguns algoritmos gráficos, permitindo assim a análise global do sistema.

O diagrama de blocos do terminal implementado é mostrado na figura 5.1, e suas principais características são: resolução de 256 x 256 pixels com quatro planos de memória, permitindo portan-

to 16 combinações de cores; operação com frequência vertical de 60 Hz no modo não entrelaçado; frequência horizontal de 15700 Hz, e frequência de pixels de 5 Mhz.

5.1.1. Mecanismo de resolução de conflitos

Como foi visto no Capitulo 3, o controlador de vídeo 6845 não possui mecanismos especiais para solução do conflito de acesso à memória, sendo portanto a solução destes conflitos o primeiro passo para o projeto do sistema. Para isso foram analisadas as características específicas de cada uma das partes que se pretendia utilizar.

O 6845 quando utilizado com CPU's da familia 6800 (8 bits), possui uma caracteristica particular, isto é, permite o intercalamento dos acessos à memória de vídeo pela utilização do sinal Ø2. As CPU's da família 6800 trabalham com um relógio de duas fases, sendo possível utilizar um múltiplo inteiro da fase 2 do relógio como "clock" de caracteres do 6845, tornando possivel o intercalamento dos acessos à memória conforme mostra a figura 5.2.

A CPU 68000 trabalha normalmente no modo assincrono, apresentando porém sinais de controle capazes de fazer com que se torne compatível com os componentes da familia 6800. No entanto, se optarmos por tornar a memória de vídeo um dispositivo síncrono, o sistema se tornaria muito lento, uma vez que, neste modo,

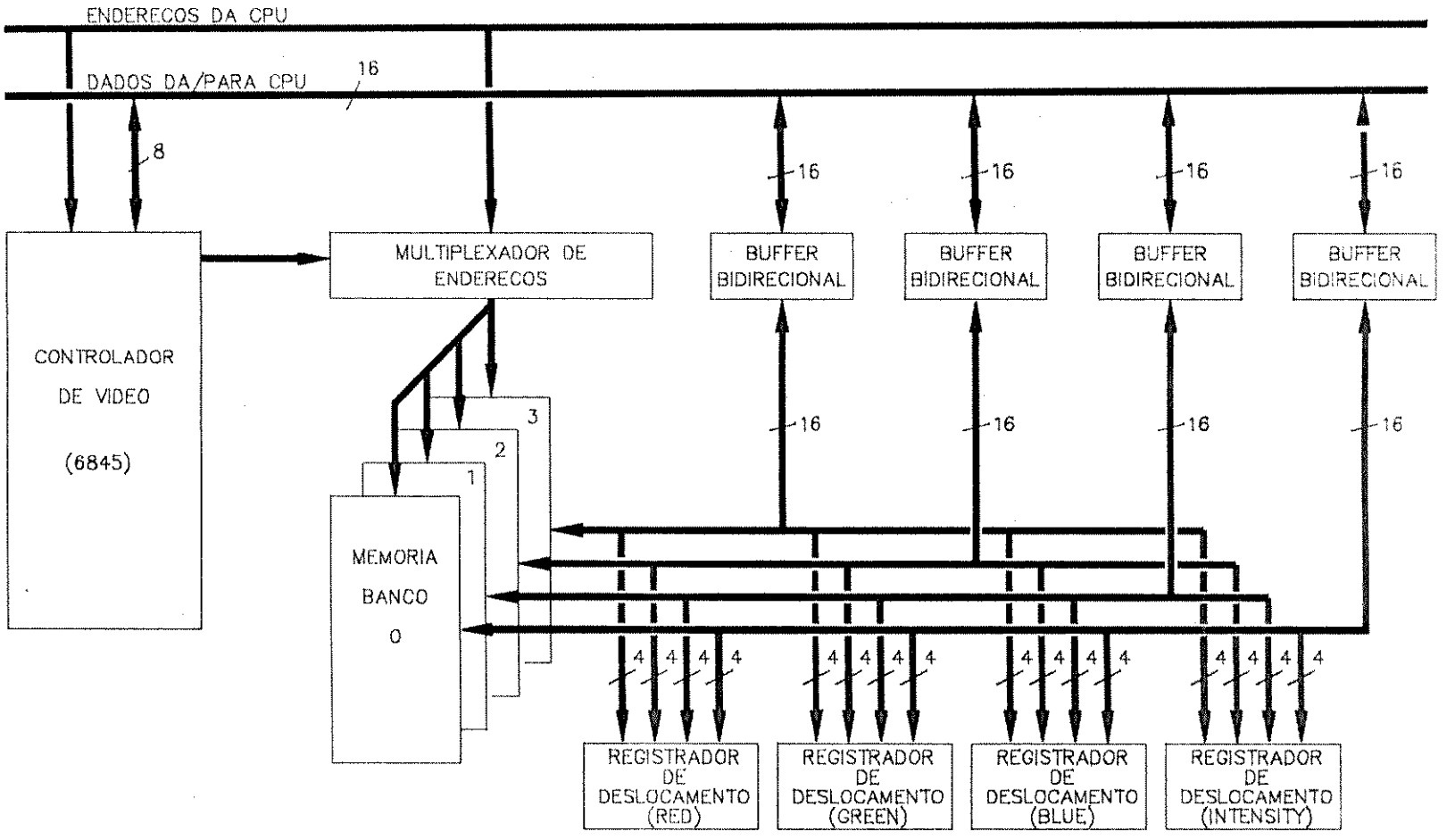


Figura 5.1 — Diagrama de Blocos do Terminal Colorido de Baixa Resolucao Implementado

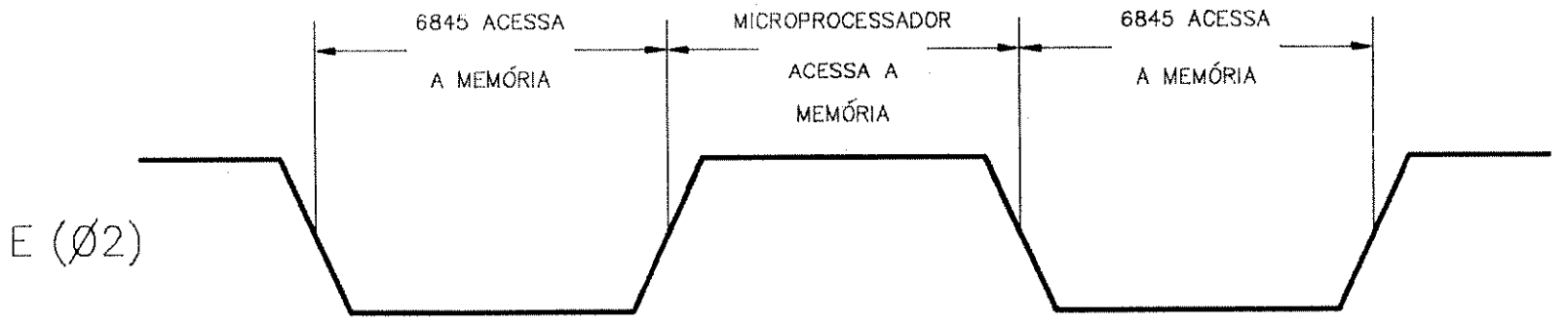


Figura 5.2 – Processo de Entrelacamento de Acessos a Memoria de Video Pelo Controlador e Processador Que Pode Ser Usado Para Resolucao de Conflitos Quando se Usa o MC6845

cada acesso da CPU 68000 ao dispositivo síncrono toma 10 ciclos de relógio contra 4 ciclos no modo assíncrono.

Para contornar este problema foi encontrado um meio de utilizar-se o intercalamento de acessos, mantendo a característica assíncrona da CPU. Para isto foi utilizado um circuito capaz de fornecer quatro sinais principais de temporização conforme o esquema apresentado nas figuras 5.3.A e 5.3.B. Estes sinais são: "clock" de pixels (DOTCLK), "clock" de caracteres (CRTCLK), sinal de início de acesso pelo controlador (SR) e sinal de sincronismo (PHI2).

O "clock" de pixels é utilizado para comandar os registradores de deslocamento. O "clock" de caracteres é utilizado pelo 6845 para comandar seus contadores internos, onde um caractere é representado por um conjunto de 16 pixels. Os sinais SR e 10MHzB são usados para comandar o acesso à memória de vídeo pelo controlador, gerando os sinais PCS ("chip select") e PLOAD (carga dos registradores de deslocamento).

O sinal PHI2 é usado para controlar o acesso à memória de vídeo pela CPU. Uma vez que a CPU opera no modo assíncrono, cada acesso à memória só é completado quando a CPU recebe o sinal DTACK ("data transfer acknoweledge"). No circuito apresentado, quando a CPU acessa a memória, os sinais A15B e E1B estão em nível lógico 0, liberando o flip-flop 13 e o registrador de deslocamento 14. Quando houver a transição do sinal PHI2 do

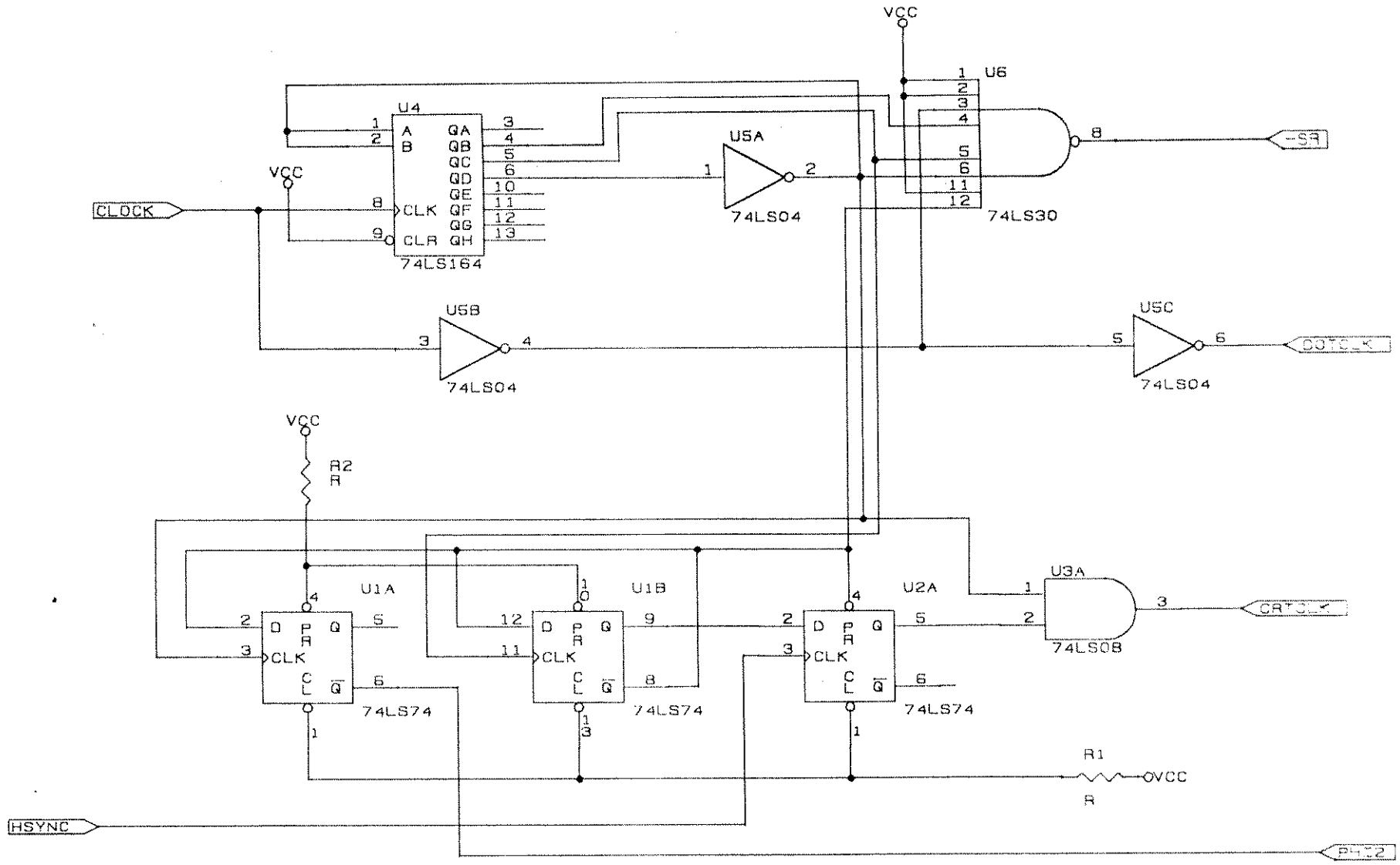


Figura 5.3.A - Logica de Geracao dos Sinais SR, DOTCLK, CRTCLK e PHI2

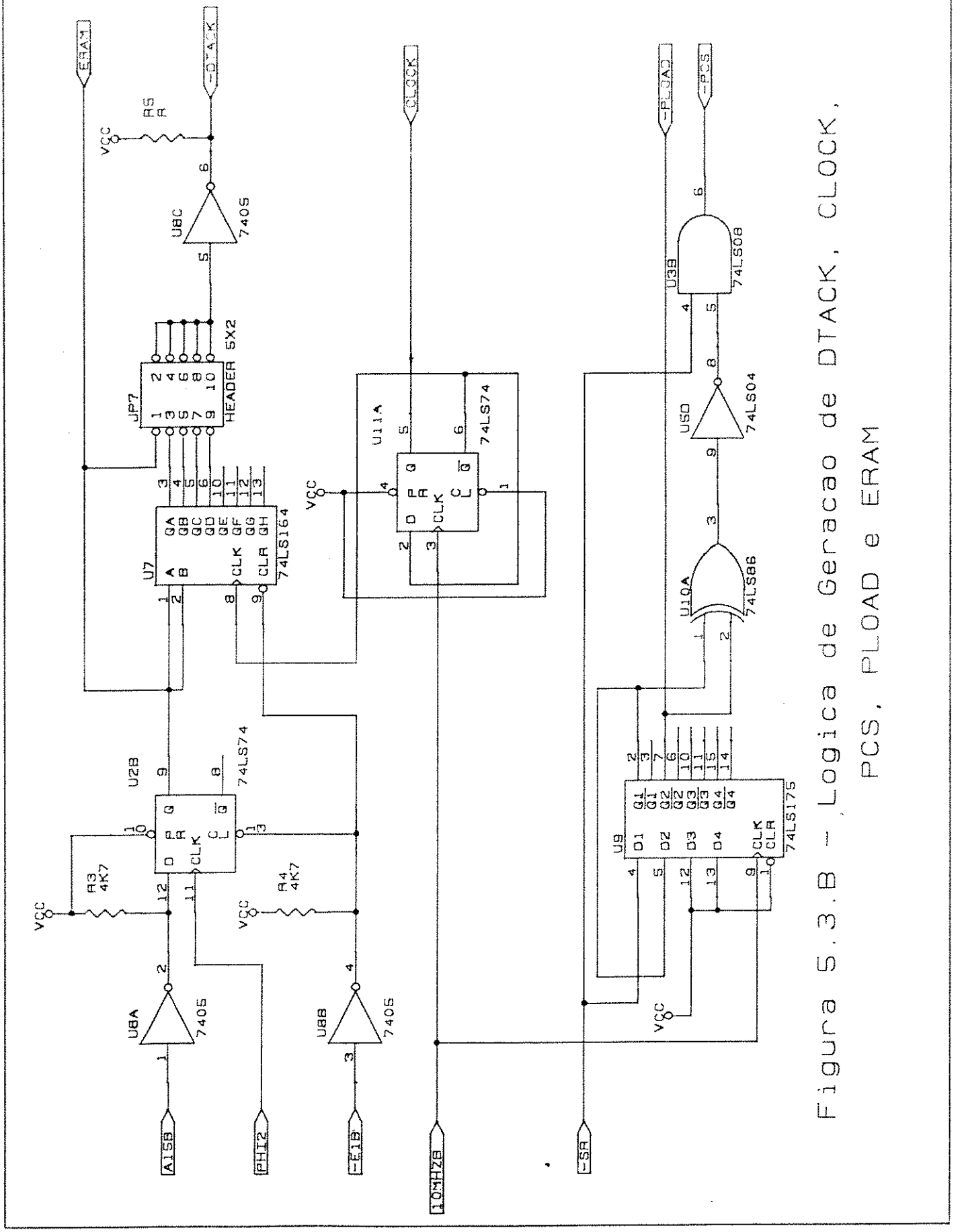


Figura 5.3.B - Logica de Geracao de DTACK, CLOCK, PCS, PLOAD e ERAM

nível 0 para o nível 1, aparecerá o sinal ERAM que liberará os buffers para a memória, e depois de um tempo selecionado através do jumper J1 ocorrerá o sinal de DTACK. O diagrama de tempo para estes sinais é mostrado na figura 5.4.

5.1.2. Organização da memória de vídeo

A memória de vídeo deste projeto foi implementada utilizando-se 16 componentes tipo RAM estática 6116 de 2 Kbytes organizadas como mostra a figura 5.5.A. Esta disposição dos componentes visa facilitar o acesso do ponto de vista do programador, tornando a memória continua em termos de pixels. Do ponto de vista da CPU a memória está organizada como um bloco de 16 Kwords de 16 bits, onde cada palavra armazena informação de 4 pixels conforme mostra a figura 5.5.B. Os pixels são armazenados na sequência mostrada pela figura 5.5.C.

A cada acesso do 6845, 64 bits são transferidos aos registradores de deslocamento, sendo 16 bits para cada um dos registradores que correspondem aos sinais de vídeo R, G, B, e I, conforme mostra a figura 5.5.D. No primeiro acesso são transferidos os pixels armazenados nos endereços 0, 2, 4 e 6, que correspondem aos primeiros 16 pixels da tela; no segundo acesso os pixels armazenados nos endereços 8, 10, 12 e 14 são lidos e que correspondem aos 16 pixels seguintes; e assim sucessivamente, até atingir-se o fim da memória.

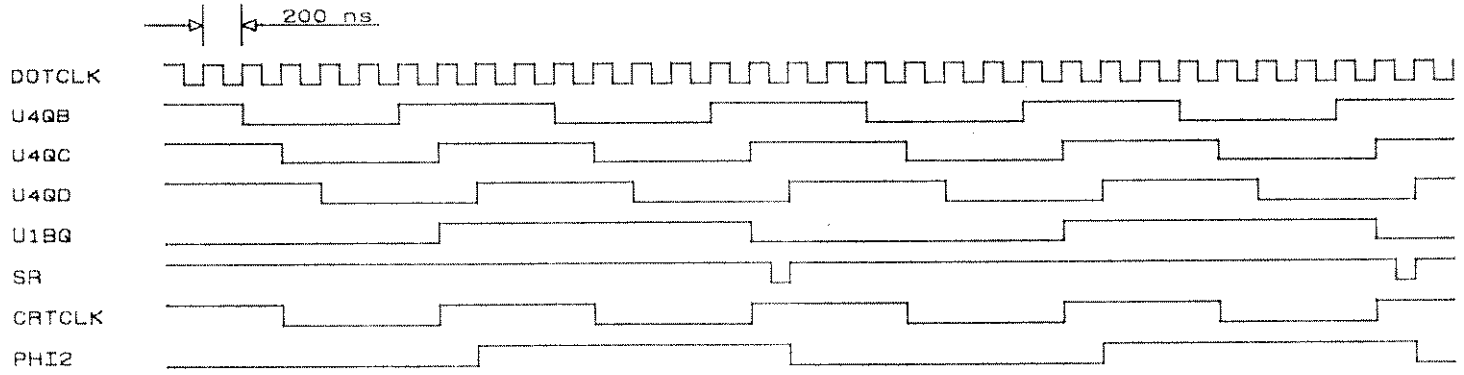
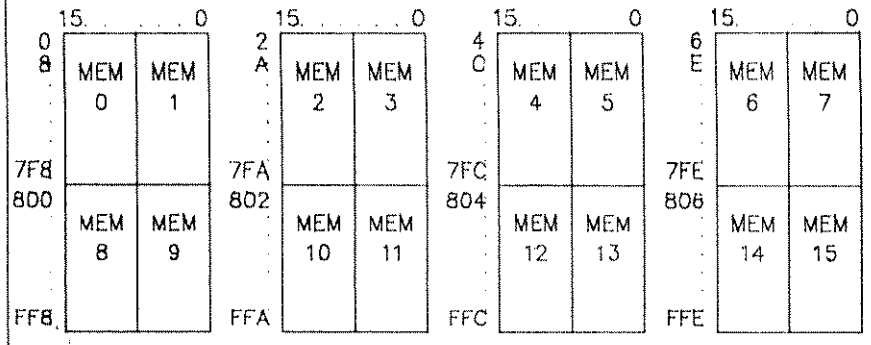
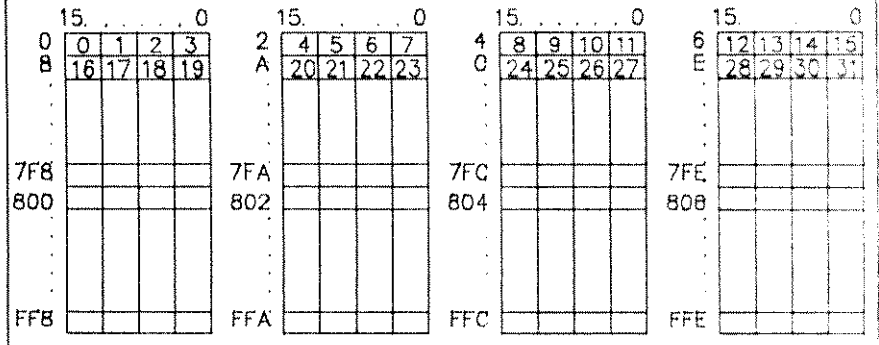


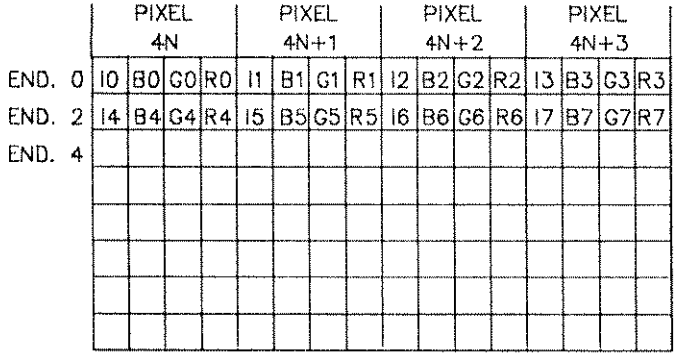
Figura 5.4 - Temporizacao dos Sinais dos Circuitos das Figuras 5.3.A e 5.3.B



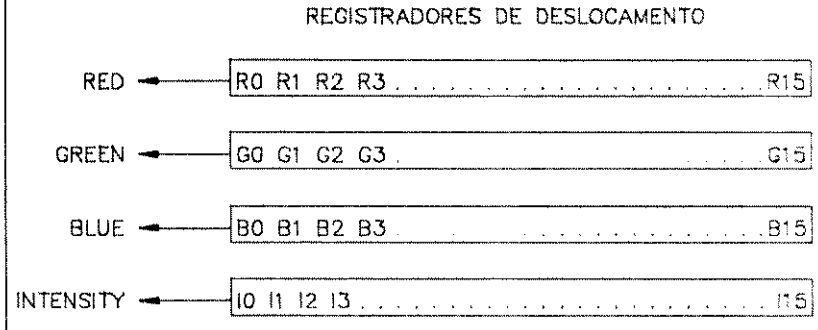
(A) – Disposicao dos Componentes de Memoria



(B) – Disposicao dos Pixels Dentro da Memoria



(C) – Distribuicao dos Bits RGBI nas Posicoes de Memoria



(D) – Distribuicao dos Bits RGBI nos Registradores de Deslocamento

Figura 5.5 – Organizacao da Memoria de Video

O controlador foi programado para 16 caracteres por linha de raster, uma vez que a cada acesso do 6845 são transferidos 16 pixels e a linha de vídeo é composta por 256 pixels.

5.1.3. Processador utilizado

Como já foi dito anteriormente, o processador utilizado foi o MC68000 da Motorola, que é um processador de uso geral, tendo como principal característica a estrutura interna de 32 bits, e um poderoso conjunto de instruções e modos de endereçamento, o que possibilita uma grande flexibilidade no desenvolvimento de software. Apesar de sua estrutura interna de 32 bits, externamente seu barramento de dados é de 16 bits, o que implica em 2 acessos à memória para cada palavra de 32 bits. Possui ainda 24 linhas de endereços o que possibilita o acesso a 16 Mbytes de memória.

5.1.4. Vantagens apresentadas

A vantagem deste tipo de estrutura utilizada no terminal gráfico é fazer com que a memória de vídeo permaneça mais tempo disponível para acessos da CPU, em relação a um terminal que utilizasse apenas os tempos de retraço horizontal e vertical para acessos da CPU. No casos da utilização dos tempos de retraço, a disponibilidade da memória para resolução de 256 x 256 pixels é de aproximadamente 30%, ao passo que com a estrutura apresentada torna-se de 50%; que representa um ganho significativo (66%).

5.1.5. Desvantagens

A principal desvantagem apresentada na estrutura do terminal implementado é que acessos da CPU a pixels individuais não possivel, uma vez que uma palavra de memória representa quatro pixels, obrigando portanto que o firmware implementado tenha mecanismos para detectar, dentro de uma palavra, qual dos conjuntos de 4 bits deva ser alterado, fazendo com que instruções de deslocamento e operações extras sejam executadas para a determinação do pixel individual.

5.1.6. Firmware implementado

O software para o terminal está organizado em três blocos:

. módulo principal,

. módulo de interpretação,

. módulo de apoio gráfico.

O módulo principal é responsável pela recepção e análise dos comandos, os quais serão interpretados no módulo de interpretação.

No módulo de apoio gráfico estão as rotinas responsáveis pelos algoritmos gráficos. Foram implementados as funções: apaga a tela, move cursor para determinada posição, muda cor de linha ou texto, desenha linha, escreve texto, circulo, ativa/desativa preenchimento de poligonos, cor de preenchimento, desenha retân-

gulo, define tamanho de caracteres, define distância entre caracteres, inicio de poligono irregular (begin panel), fim de poligono irregular (end panel), elipse.

Os módulos principal e de interpretação foram escritos em linguagem Pascal, ao passo que as rotinas gráficas foram escritas em Assembler.

5.2. Terminal colorido de média resolução

Este projeto teve por finalidade básica a implementação de uma placa de vídeo gráfica inteligente para ser utilizada em sistemas com barramento VME, tendo como diretivas principais:

a. a placa deveria comportar-se como um periférico do sistema, ou seja, atuar como escrava no barramento;
b. deve suportar um pacote gráfico e deve conter firmware para geração de primitivas gráficas e para tratamento de objetos 2D.
c. utilização de controlador de vídeo classe II, vídeo RAM, e CPU 680XX.

As características gerais do terminal implementado são:

a. resolução de 512 H x 382 V pixels;
b. frequência horizontal de 27 Khz;
c. frequência vertical de 60 Kz (não entrelaçados);
d. frequência de pixels de 17 Mhz;
e. 4 planos de memória de vídeo (16 cores simultâneas).

O sistema foi implementado em uma placa padrão europeu duplo, que é vista pelo barramento como uma placa *DTB slave A24 D8* (24 linhas de endereço e 8 linhas de dados). Esta estrutura é mostrada na figura 5.6. Para a comunicação entre o equipamento DTB master e a CPU interna à placa de vídeo utilizou-se uma memória "Dual Port". A função de cada um dos blocos que compõem o sistema será resumida nos itens seguintes.

5.2.1. Processador utilizado

O processador utilizado nesta implementação, o MC68008, apresenta como característica fundamental compatibilidade a nível de software e estrutura interna de registradores, com o MC68000, tornando confortável o transporte do software desenvolvido anteriormente para este terminal. Outro aspecto considerado foi o tamanho físico do componente, uma vez que a placa VME possui dimensões reduzidas e a utilização do 68000 tornar-se-ia mais complicada. O processador 68008 dispõe de 20 linhas de endereço, o que permite o acesso a 1 Mbyte de memória, suficiente para a aplicação presente.

A característica de funcionamento assíncrono do 68008 faz com que a cada acesso do componente a qualquer dispositivo periférico só termine com o sinal DTACK (Data Transfer Acknowledge) gerado pelo dispositivo acessado (esta característica assíncrona também está presente no barramento VME). Assim sendo, existe um circuito de geração de DTACK para os acessos da CPU aos

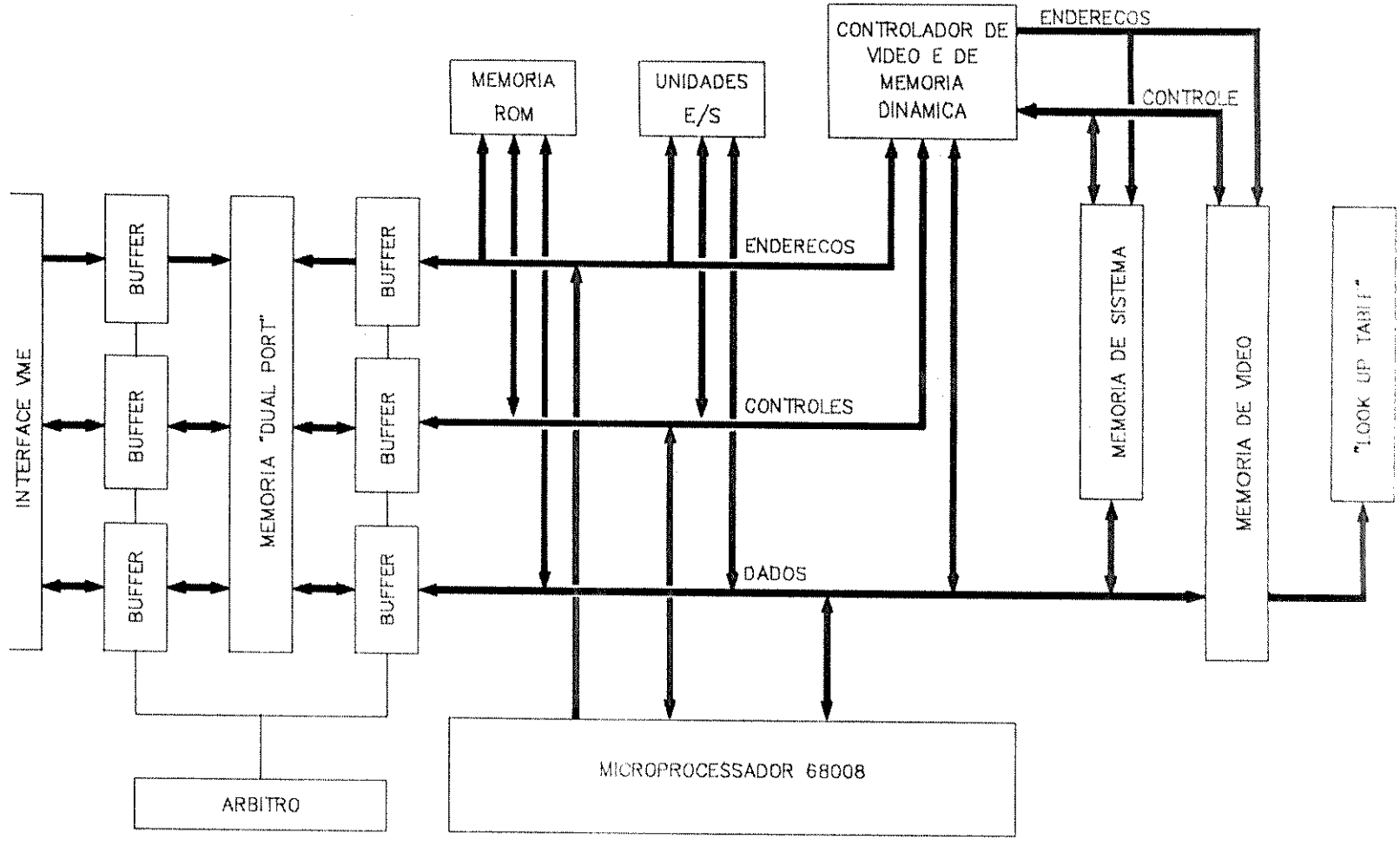


Figura 5.6 – Diagrama de Blocos do Terminal de Video Grafico Colorido de Media Resolucao Implementado

dispositivos ROM, unidades de E/S da familia 68000 e à memória "dual port", e um circuito que gera o DTACK através do sinal READY do TMS34061, para os acessos à RAM de sistema e VRAM.

A CPU 68008 tem por finalidade a geração das primitivas gráficas, recebendo instruções através da memória de comunicação e preenchendo a memória de vídeo com a configuração de bits adequada à formação da imagem desejada. E também responsável pela comunicação com as unidades de E/S locais ("tablet", "mouse", teclado, etc). O circuito da CPU é apresentado na figura 5.7. Os sinais importantes presentes na figura são:

CLOCK - Sinal de relógio da CPU, operando na frequência de 8.867 MHZ, gerado a partir de um oscilador a cristal com frequência de 17.734 MHZ, que é o "dot clock", ou seja a frequência dos pixels na tela.

VPA - sinal responsável pelo funcionamento síncrono da CPU, utilizado na comunicação entre a CPU e componentes da família 6800. Neste caso o componente utilizado foi o 6850, que é o controlador de interface serial.

IPL - combinação de dois sinais que geram interrupções para a CPU, com quatro níveis diferentes. Foram implementadas interrupções para interface serial e escrita na memória "dual port".

DTACK - sinal responsável pelo reconhecimento de dados prontos nas tranferências entre a CPU e os dispositivos assíncronos. Indica o final do ciclo presente.

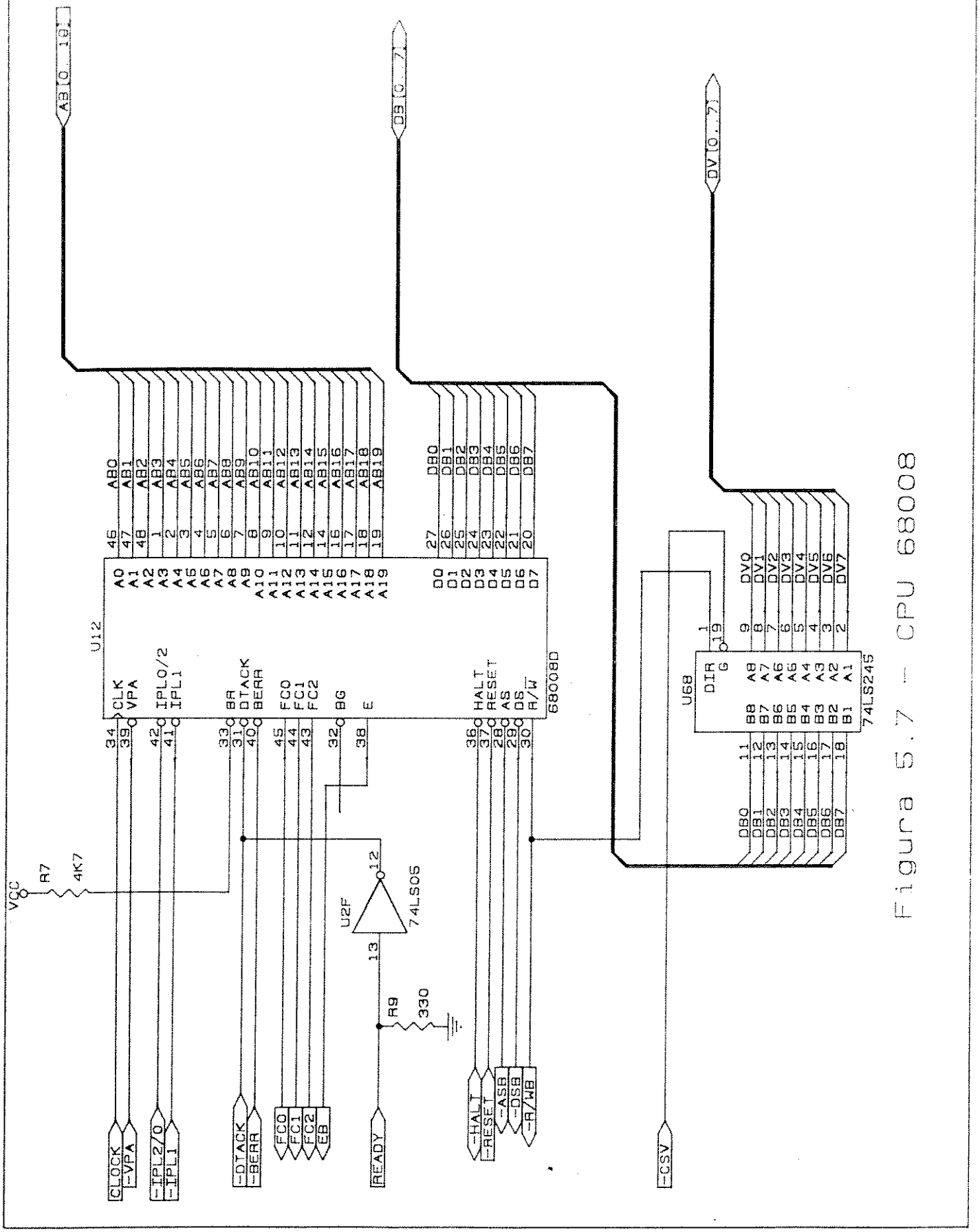


Figura 5.7 - CPU 68008

BERR - (Bus Error) sinal responsável pela interrupção de um ciclo de barramento no qual o dispositivo endereçado demora para responder (não ocorre o sinal de DTACK depois de um certo tempo). Este sinal causa uma interrupção na CPU, fazendo com que ela retorne a funcionar em um ponto conhecido.

FC0-FC2 - sinais responsáveis pela identificação do ciclo de barramento corrente. No circuito desenvolvido, foi utilizado para reconhecimento de interrupção.

EB - Sinal de clock para dispositivos lentos, utilizado para sincronizar o acesso aos componentes da família 6800.

READY - sinal proveniente do controlador de vídeo TMS34061, informando que o acesso ao controlador ou às memórias de vídeo ou RAM de sistema está completo. Este sinal gera um DTACK.

HALT - neste circuito, este sinal é usado em conjunto com o sinal de RESET para reinicializar a CPU.

RESET - utilizado na reinicialização do software residente na placa em um ponto conhecido.

ASB - (Address Strobe) - indica que os endereços presentes no barramento de endereços estão estáveis e um novo ciclo de acesso irá ter início.

DSB - (Data Strobe) - indica que os dados presentes no barramento de dados estão estáveis no ciclo de escrita, ou que a CPU está pronta para ler os dados no ciclo de leitura.

R/WB - sinal que informa se o ciclo corrente do barramento é um ciclo de leitura ou escrita de dados.

AB0-AB19 - barramento de endereços composto de 20 linhas.

DB0-DB7 - barramento de dados composto de 8 linhas.

CSV - sinal que indica que o ciclo corrente corresponde a um acesso ao controlador de vídeo ou às memória de vídeo ou de sistema. E utilizado neste circuito para a liberação do buffer do barramento de dados que é direcionado aos circuitos de vídeo.

DV0-DV7 - barramento de dados isolado para os circuitos de vídeo.

5.2.2. Controlador de vídeo e mecanismo de resolução de conflitos no acesso à memória de vídeo

O sistema de controle de vídeo e memória é responsável pela geração dos sinais de sincronismo horizontal e vertical, resolução dos conflitos de acesso à memória de vídeo, geração dos ciclos de "refresh" para as memórias tipo RAM dinâmicas (vídeo e sistema), geração dos sinais de controle para as vídeo RAM's, e geração dos ciclos de regeneração da imagem. E baseado no componente TMS34061 da Texas.

O controlador de vídeo TMS34061 possui dois mecanismos distintos para a solução dos conflitos de acesso à memória de vídeo. No primeiro deles, o acesso da CPU à memória será prolongado caso um dos ciclos internos do controlador esteja em andamento ("refresh" das RAM dinamicas ou carga dos registradores de deslocamento). Assim, se a CPU acessar a memória de vídeo durante o andamento de um destes dois ciclos, um sinal READY externo é gerado,

podendo ser usado como "wait state" para a CPU, fazendo com que esta prolongue seu ciclo de acesso à memória até o término do ciclo interno do controlador.

No outro mecanismo, a cada um dos ciclos internos do controlador, um sinal externo é gerado, que será utilizado para colocar a CPU em estado de "HOLD", parando portanto o seu funcionamento nos instantes em que o controlador acessa a memória.

Evidentemente o primeiro mecanismo é mais adequado, uma vez que a CPU só sofrerá atraso em seus ciclos, quando estes coincidirem com os acessos do controlador. No nosso projeto, o primeiro caso foi utilizado.

De qualquer maneira, são executados apenas três ciclos de "refresh" da memória dinâmica por linha de raster (com duração de 300 ns cada), e um ciclo de carga dos registradores de deslocamento a cada duas linhas de raster (também de 300 ns cada) visto a arquitetura utilizada para a memória. A temporização dos sinais de controle ocorre de acordo com o que está mostrado na figura 5.8. Portanto, como o período da linha de raster é de 33 us, e a cada duas linhas são utilizados um total de 7 ciclos de 300 ns (6 para "refresh" e um de carga de registradores de deslocamento), temos a utilização de 2,1 us em cada 66 us, o que nos dá uma disponibilidade de 96,8% da memória para acessos da CPU.

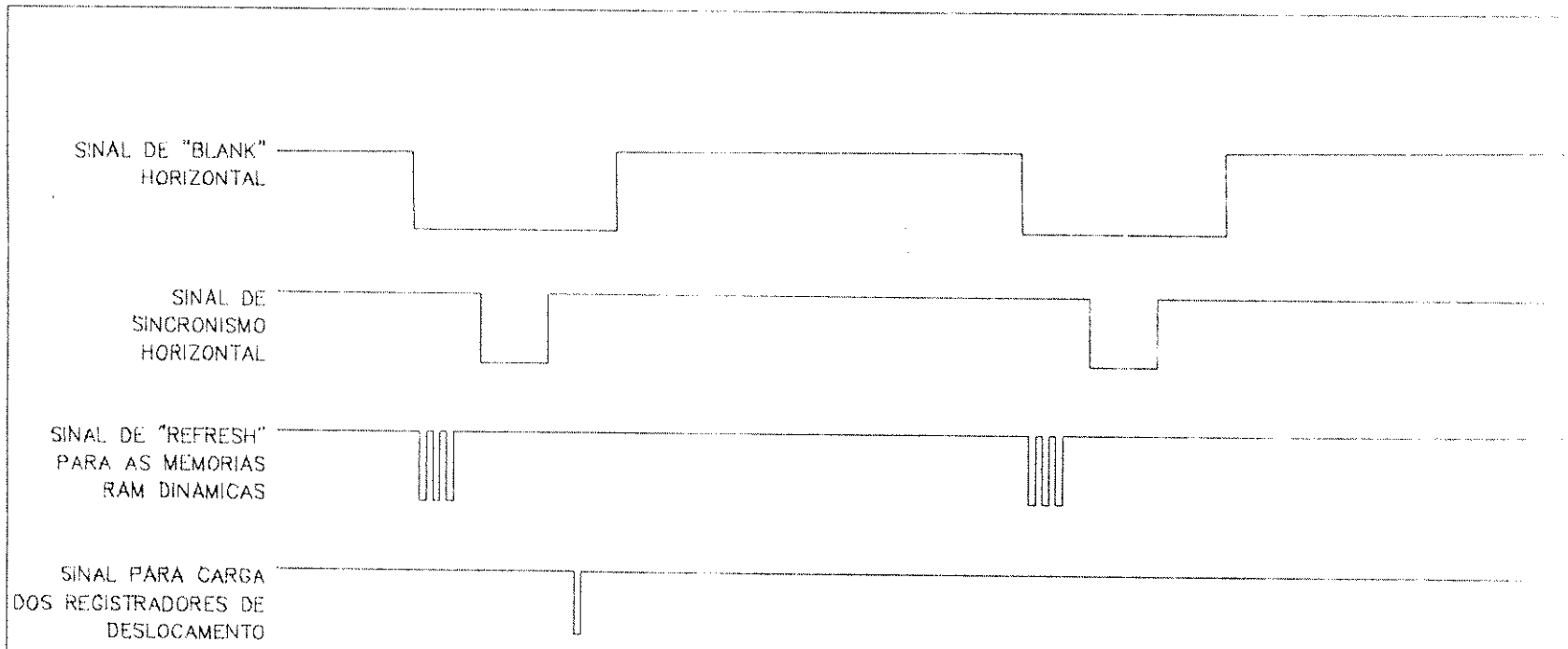


Figura 5.8 – Relacao Entre os Sinais de "Blank", Sincronismo Horizontal, "Refresh" das RAMs Dinamicas e Carga dos Registradores de Deslocamento, Gerados pelo Controlador TMS34061

O circuito do controladodor de vídeo é mostrado na figura 5.9, onde destacamos os seguintes sinais:

FS0-FS2 - sinais que indicam o tipo de acesso que a CPU irá executar, por exemplo, acesso à memória de vídeo ou sistema, acesso aos registros internos do controlador, acesso à memória via registros XY, ou carga direta dos registradores de deslocamento internos às vídeo RAM.

VIDCLK - clock de vídeo, sinal responsável pela geração de todos os sinais relativos à parte de video, bem como a atualização dos contadores de caracteres, contadores de linhas de raster, contadores para geração de sincronismos de vídeo. Seu valor é de 4.43 MHZ, que corresponde ao DOTCLK dividido por quatro.

CLOCK - sinal responsável pelo timing da interface entre a CPU e o controlador de vídeo. E igual ao CLOCK da CPU ou seja 8.867 MHZ.

CSV - sinal de seleção do controlador de vídeo. Qualquer acesso ao controlador ou às memórias de vídeo ou de sistema deve ser indicado por este sinal.

ALE - sinal de strobe indicando que os sinais de entrada do controlador de vídeo (CSV, RA0-RA8, CA0-CA8, FS0-FS2) estão estáveis.

R/WB - sinal que indica se o ciclo de acesso ao controlador de vídeo ou memórias, é de leitura ou escrita.

RAM1/2 - sinal que indica qual banco da memória de vídeo será acessado. A combinação deste com o sinal DSB é decodificada para a formação dos sinais CEL e CEH. Estes dois sinais são responsáveis pela ativação dos sinais

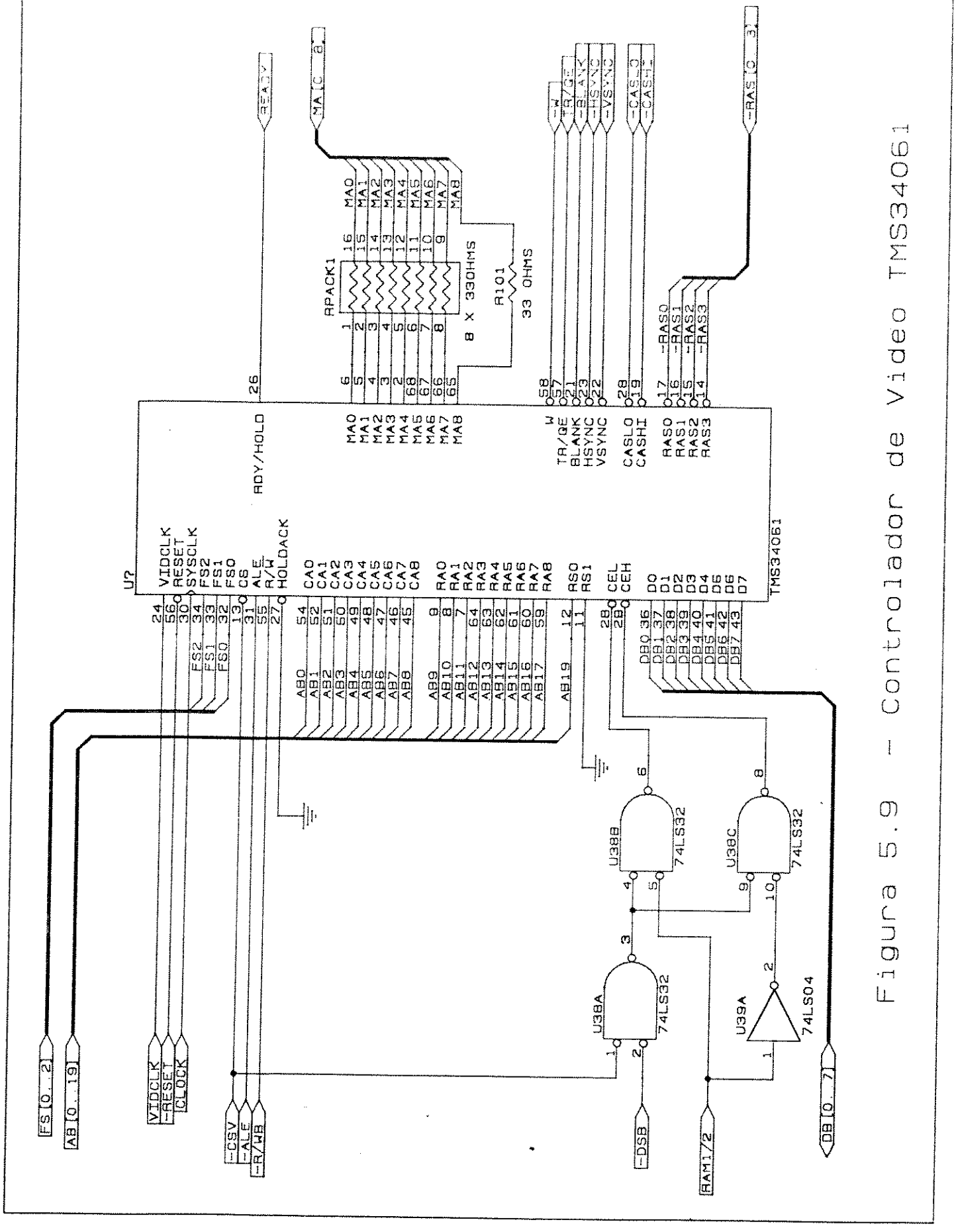


Figura 5.9 - Controlador de Video TMS34061

CASLO e CASHI, que selecionam respectivamente os bancos 0 e 1 da memória de vídeo.

READY - sinal que indica se o controlador já terminou ou não a transferência corrente.

MA0-MA8 - sinais de endereços multiplexados, gerados pelo controlador de vídeo a partir dos endereços RA e CA, ou dos registros X e Y.

W - sinal de controle de escrita gerado pelo controlador de vídeo de acordo com o sinal R/WB.

TR/QE - sinal gerado pelo controlador de vídeo, responsável pelo controle de acesso aos registradores de deslocamento presentes nas memórias video RAM.

BLANK - sinal gerado pelo controlador de vídeo indicando intervalos onde o sinal de vídeo estará inativo.

HSYNC e VSYNC - sinais de sincronismo horizontal e vertical gerados pelo controlador de vídeo.

CASLO e CASHI - sinais para controle das memórias. Usados em conjunto com os sinais RAS.

RAS0-RAS3 - usados em conjunto com os sinais CASHI e CASLO, indicando bancos de memória. O conjunto RAS0 e CASLO seleciona a memória de sistema. O Conjunto RAS1 e CASLO seleciona um dos bancos da memória de vídeo. O conjunto RAS1 e CASHI seleciona o outro banco da memória de vídeo. Os sinais RAS são ainda responsáveis pelo "refresh" das memórias dinâmica.

5.2.3. Memória de sistema

A memória de sistema é composta por 8 pastilhas do tipo RAM dinâmica tipo 4256, formando um total de 256 Kbytes. E usada como memória de trabalho do firmware implementado. O diagrama da memória é mostrado na figuras 5.10.

5.2.4. Memória de vídeo

A memória de vídeo é composta por dispositivos tipo vídeo RAM, num total de 16 componentes formando 128 Kbytes de memória. São utilizados componentes do tipo TMS4161 da Texas, contendo cada um deles, 64 Kbits, e um registrador de deslocamento de 256 bits.

O arranjo de memória foi organizado como mostra a figura 5.11. Esta organização deve-se ao fato do dispositivo que implementa a "look-up table", necessitar em suas entradas da informação correspondente a dois pixels de cada vez, obrigando então que a estrutura de memória implementada forneça os dados desta forma. Por outro lado, a frequência dos funcionamento dos registradores de deslocamento é reduzida pela metade, sendo ainda verdade que cada carga dos registradores de deslocamento corresponde a duas linhas de raster.

Deve-se salientar que pela lógica de decodificação dos endereços, o arranjo de memória é visto pela CPU do seguinte modo: os bytes de endereços 0 a 255 estão no banco 0, ao passo que os de

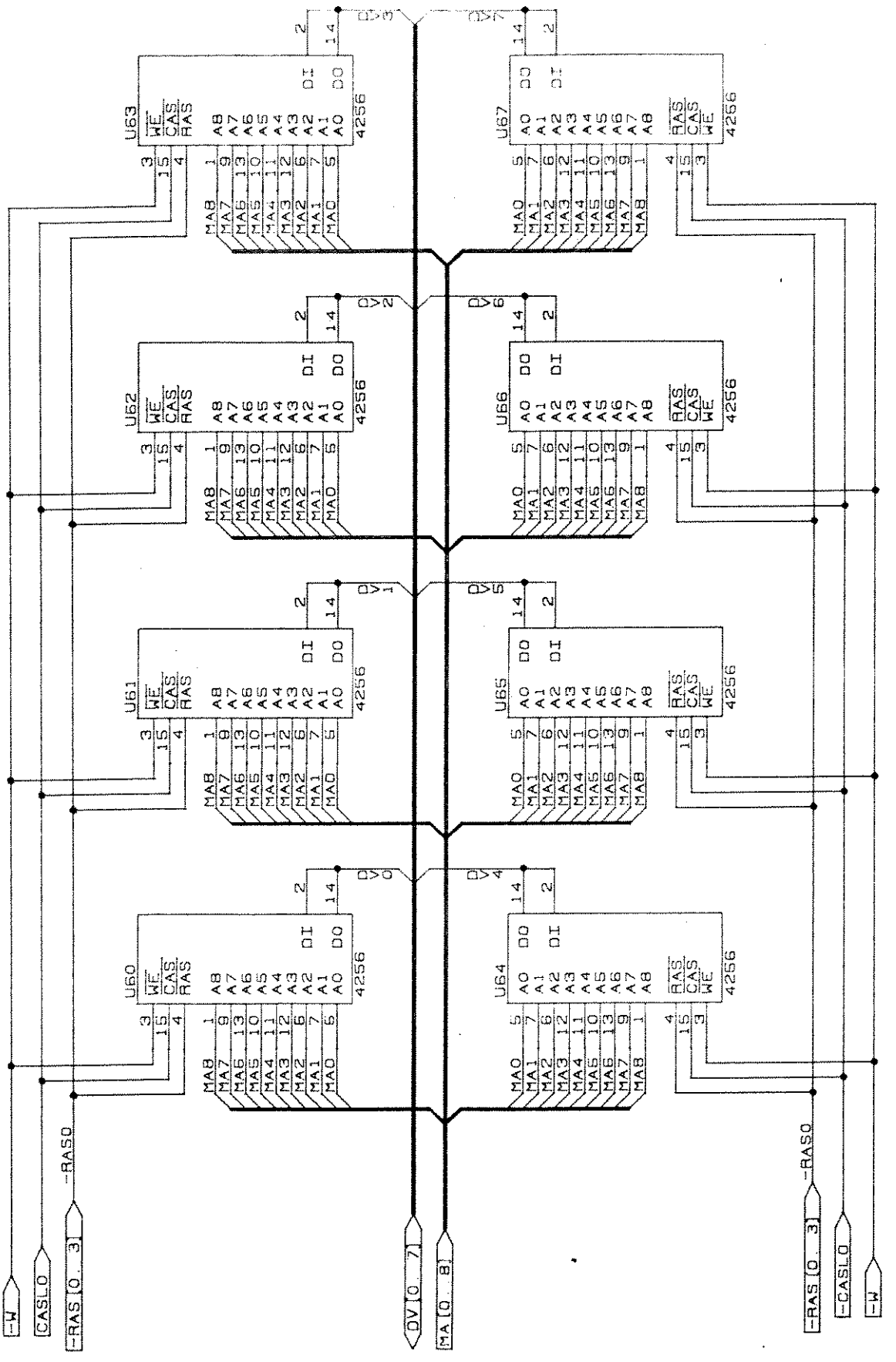


Figura 5.10 - Memoria de Sistema (256Kbytes)

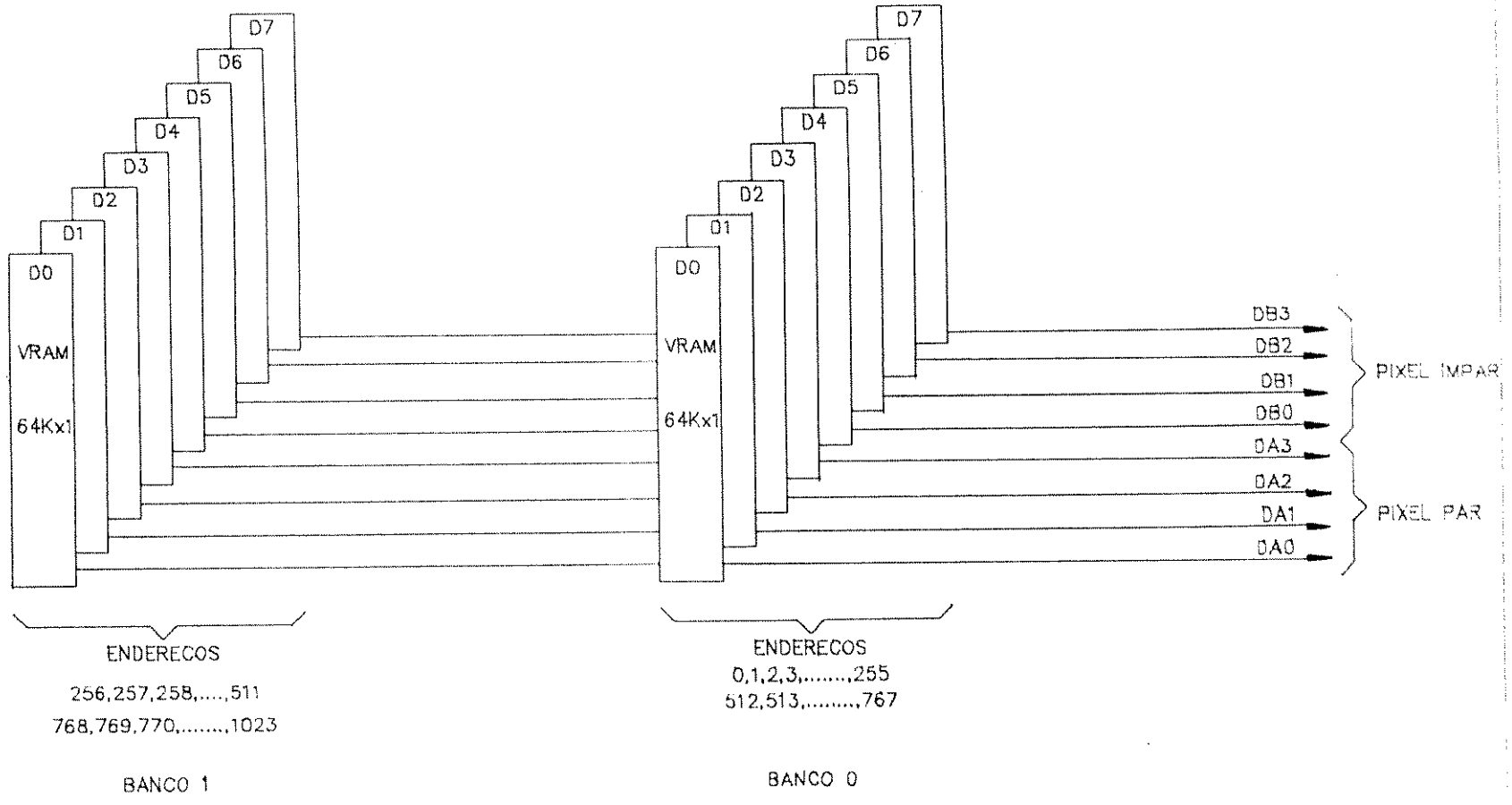


Figura 5.11 – Organizacao da Memoria de Video Para o Terminal Grafico Colorido de Media Resolucao Implementado

endereços 256 a 511 estão no banco 1. Os de endereços 512 a 767 no banco 0, e assim sucessivamente. O byte de endereço 0 compõe os pixels 0 e 1, que serão apresentados ao pallet ao mesmo tempo. Deste modo, os pixels armazenados no banco 0 correspondem às linhas de raster pares, ao passo que os armazenados no banco 1 correspondem às linhas ímpares. Como a saída dos registradores de deslocamento do banco 1 estão conectadas às entradas dos registradores do banco 0, é como se dispuséssemos de registradores de 512 posições cada.

Outro fato que deve ser salientado é que este arranjo de memória permite resolução de 512 x 512 pixels.

Nas figuras 5.12A, B, C e D,é mostrada a configuração com que foi montado o arranjo de memórias, onde os sinais já foram descritos anteriormente com exceção do sinal SRCLK, que é o clock responsável pelo deslocamento dos dados nos registradores de deslocamento.

5.2.5. Pallet

A pallet é responsável pelo mapeamento do conjunto de cores a serem mostrados no vídeo. Internamente são três conversores D/A de 4 bits cada, e uma memória de 16 palavras de 12 bits, sendo possível deste modo a escolha de um grupo de 16 cores para exibição simultânea dentre um conjunto de 4096 cores. E utilizado o componente TMS34070 da Texas.

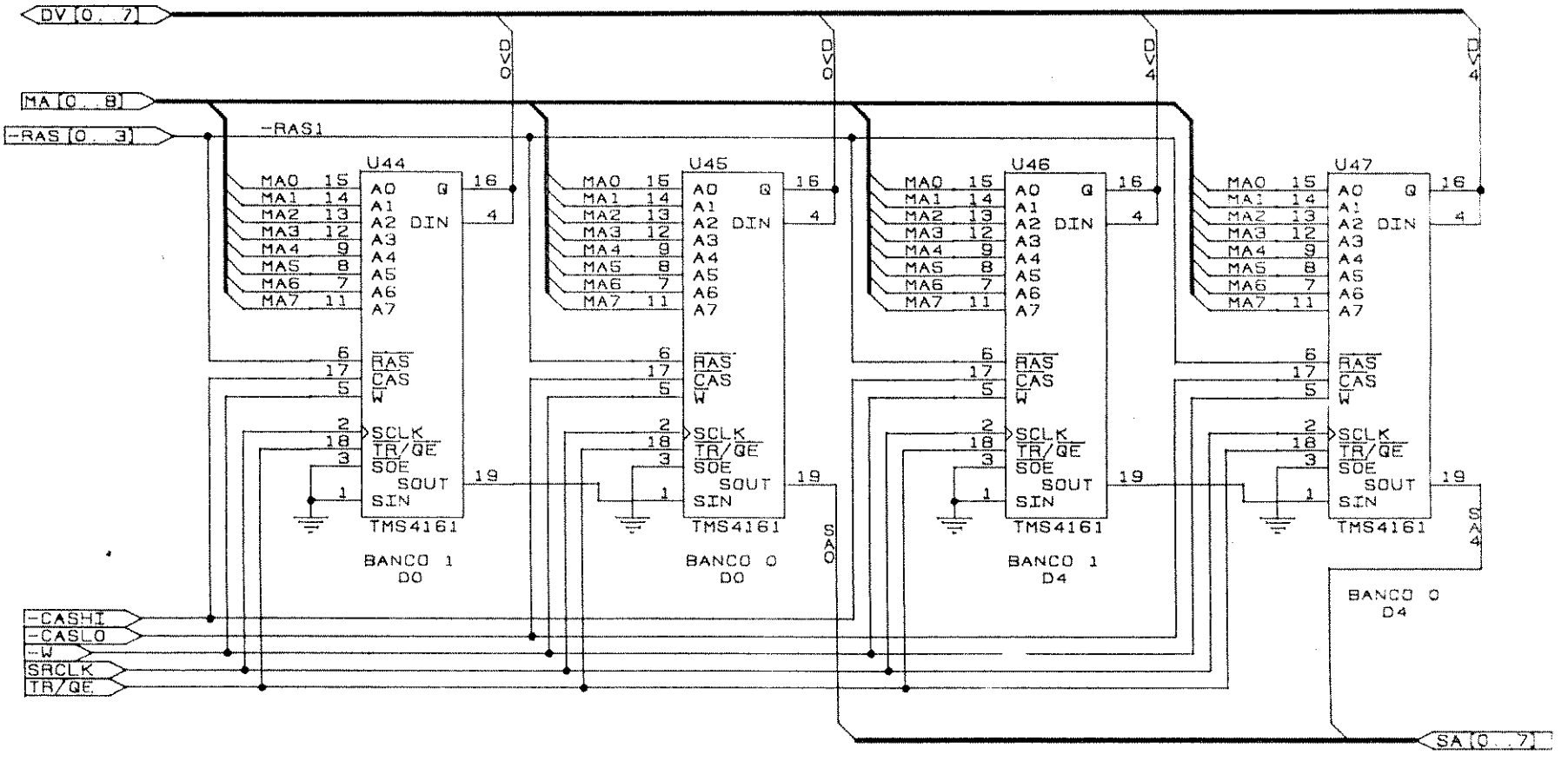


Figura 5.12.A - Memoria de Video (Parte 1)

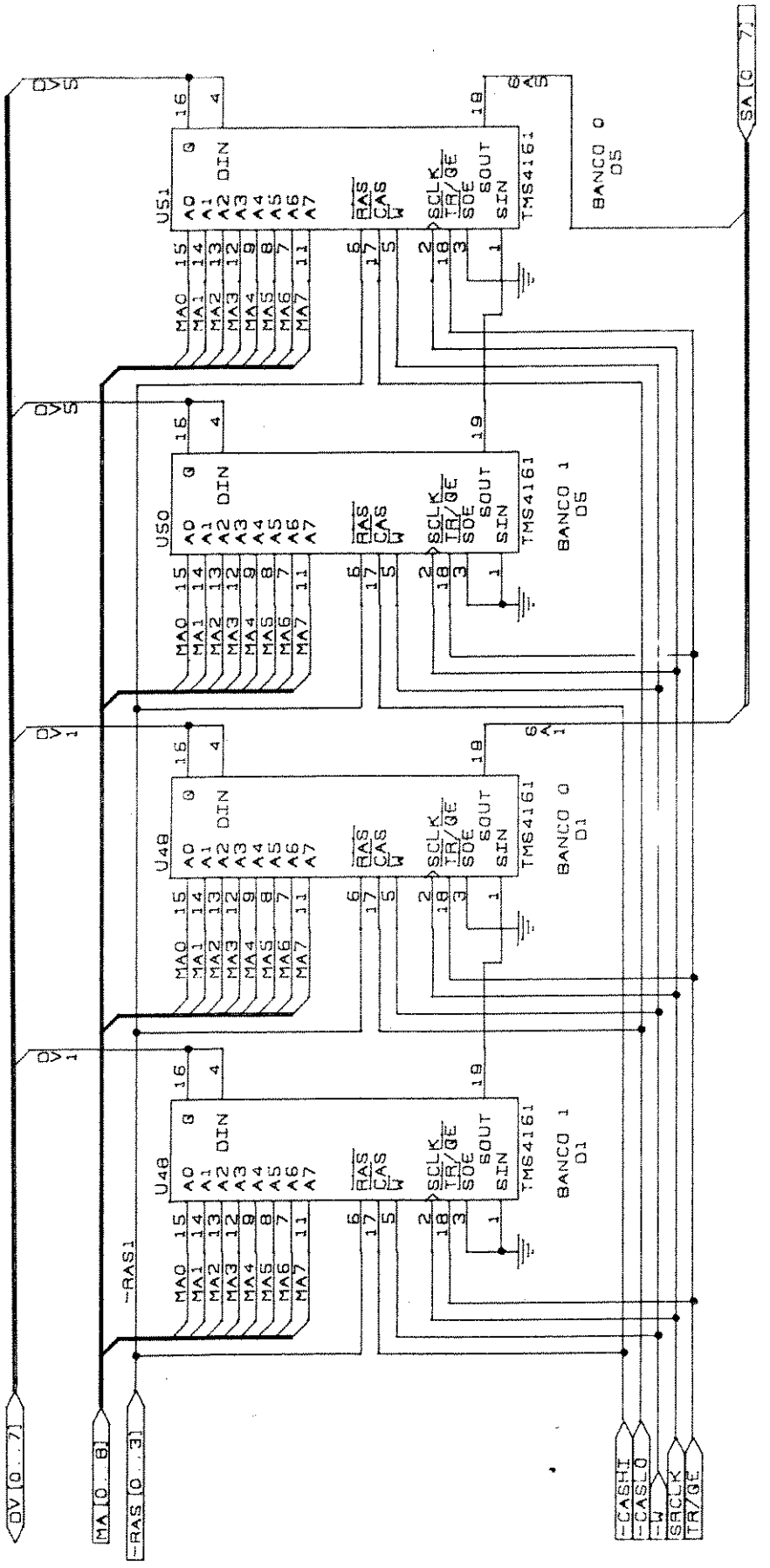


Figura 5.12.B - Memoria de Video (Parte 2)

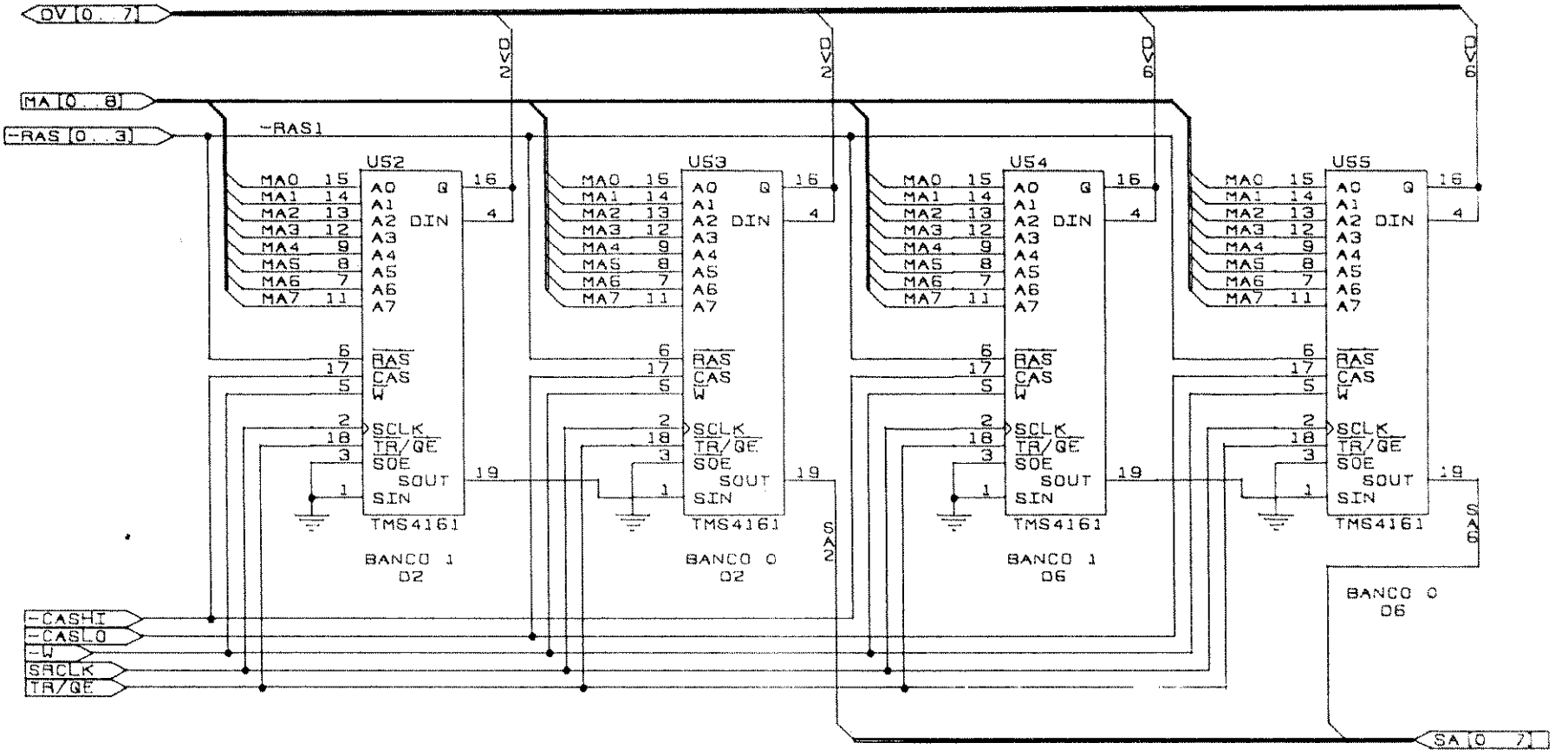


Figura 5.12.C - Memoria de Video (Parte 3)

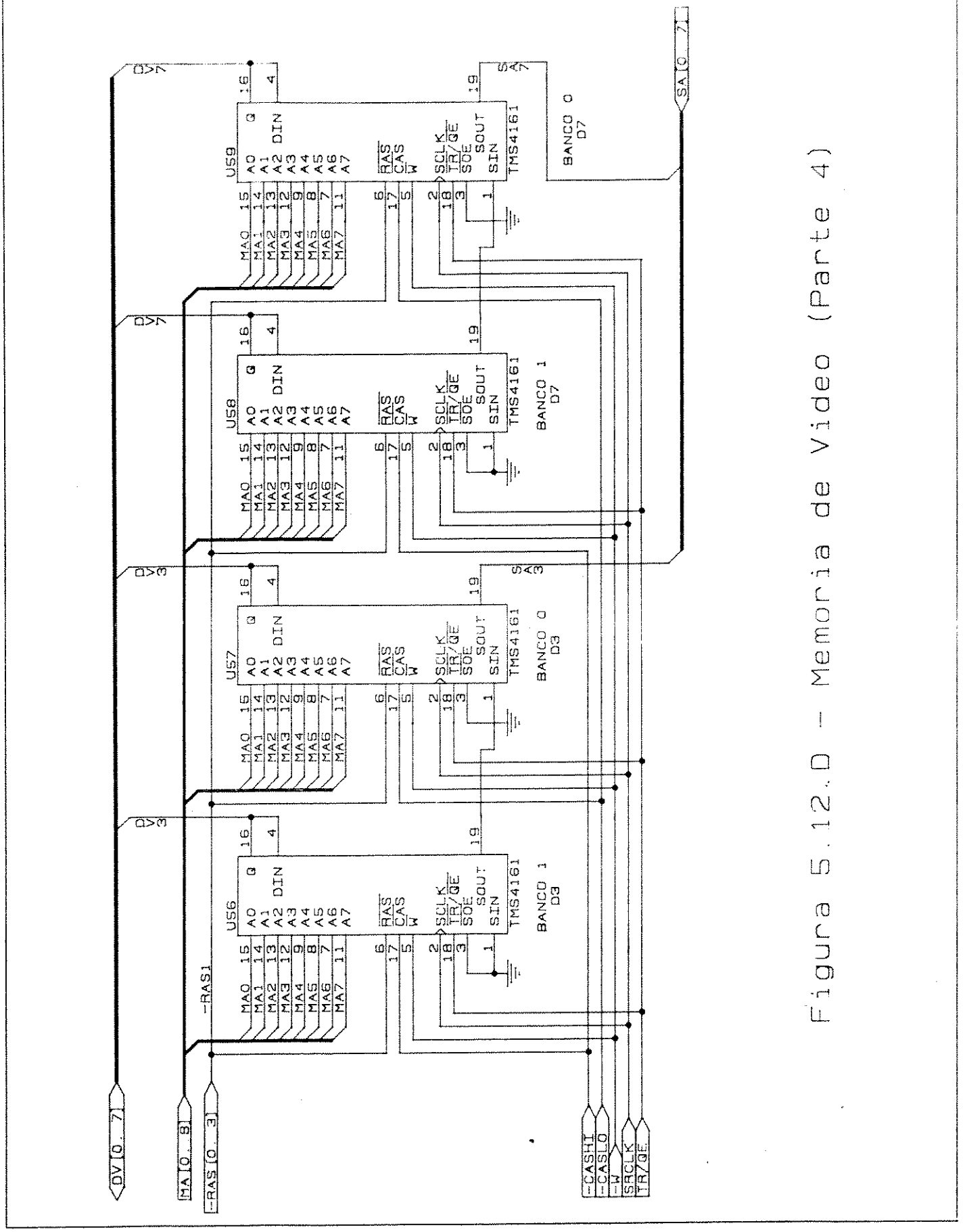


Figura 5.12.D - Memoria de Video (Parte 4)

O diagrama desta parte do circuito é mostrado na figura 5.13. O TMS34070 tem as seguintes características de interface:

O sinal 2*CLOCK ou DOTCLK representa a frequência com que os pixels serão mostrados na tela. Neste caso esta frequência é de 17.334 MHZ.

O sinal BLANK ligado à entrada DATEN do componente faz com que as saídas de vídeo se tornem inativas durante os tempos de retraço horizontal e vertical.

O sinal VSYNC ligado à entrada MODE e em conjunto com o sinal DUMP ligado em nivel lógico 0 faz com que a primeira linha de cada quadro seja interpretada como dados a serem carregados na "look up table" interna ao TMS34070 e não como pixels propriamente ditos. Assim sendo, o número de linhas de raster é uma a menos que a capacidade de memória existente.

Os sinais SA0-SA7 são provenientes das saídas dos registradores de deslocamento das memórias de vídeo, conforme foi mostrado na figura 5.11, sendo que SA0 a SA3 representam um pixel e SA4 a SA7 o próximo. Portanto, num dado instante, dois pixels estão presentes na entrada do pallet, sendo que este os multiplexa e fornece os níveis analógicos RGB sequencialmente em suas saídas.

O sinal SRCLK, que é o clock para os registradores de deslocamento, só existe durante os períodos de vídeo ativo, sendo mascarado pelo componente U50A. O sinal CLKOUT do componente U41,

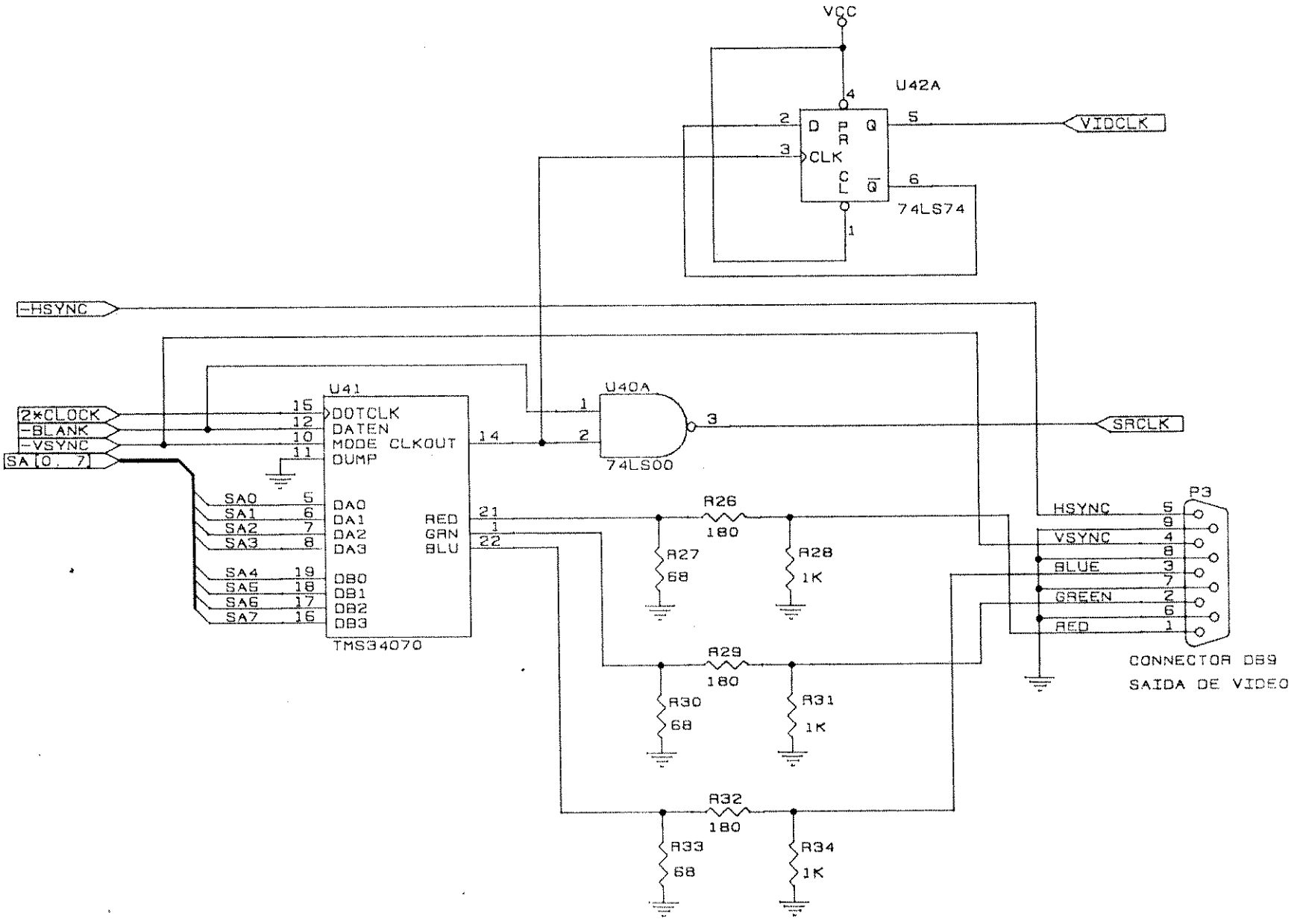


Figura 5.13 - Pallet e Interface com Monitor de Video

é o DOTCLK dividido por dois.

O sinal VIDCLK é o clock para o controlador de video existe sempre. Corresponde a DTCLK dividido por quatro.

O conector P3 é a interface com o monitor de video, onde aparecem os sinais em HSYNC e VSYNC em nivel TTL, e os sinais analógicos RED, GREEN e BLUE. Os resistores presentes no circuito, formam os adaptadores de impedância da saida do pallet para a entrada do monitor de video.

5.2.6. Unidades de Entrada e Saida

As unidades de E/S são responsáveis pela comunicação entre a placa e o meio externo através de periféricos gráficos como hardcopy, plotter, teclados, mouses, etc. Nesta implementação não foi montada a interface paralela. Foi montada uma interface serial conforme mostra a figura 5.14. O conector P2 é a interface de comunicação serial implementada. Pode-se notar o uso do sinal EB como clock do dispositivo. O sinal CS50, gerado pela lógica de decodificação é um sinal sincrono, conforme veremos adiante.

5.2.7. Memória ROM

Na memória tipo ROM está armazenado o firmware da placa responsável pela geração de primitivas gráficas, comunicação, E/S, e outras atividades. O diagrama desta parte do circuito é

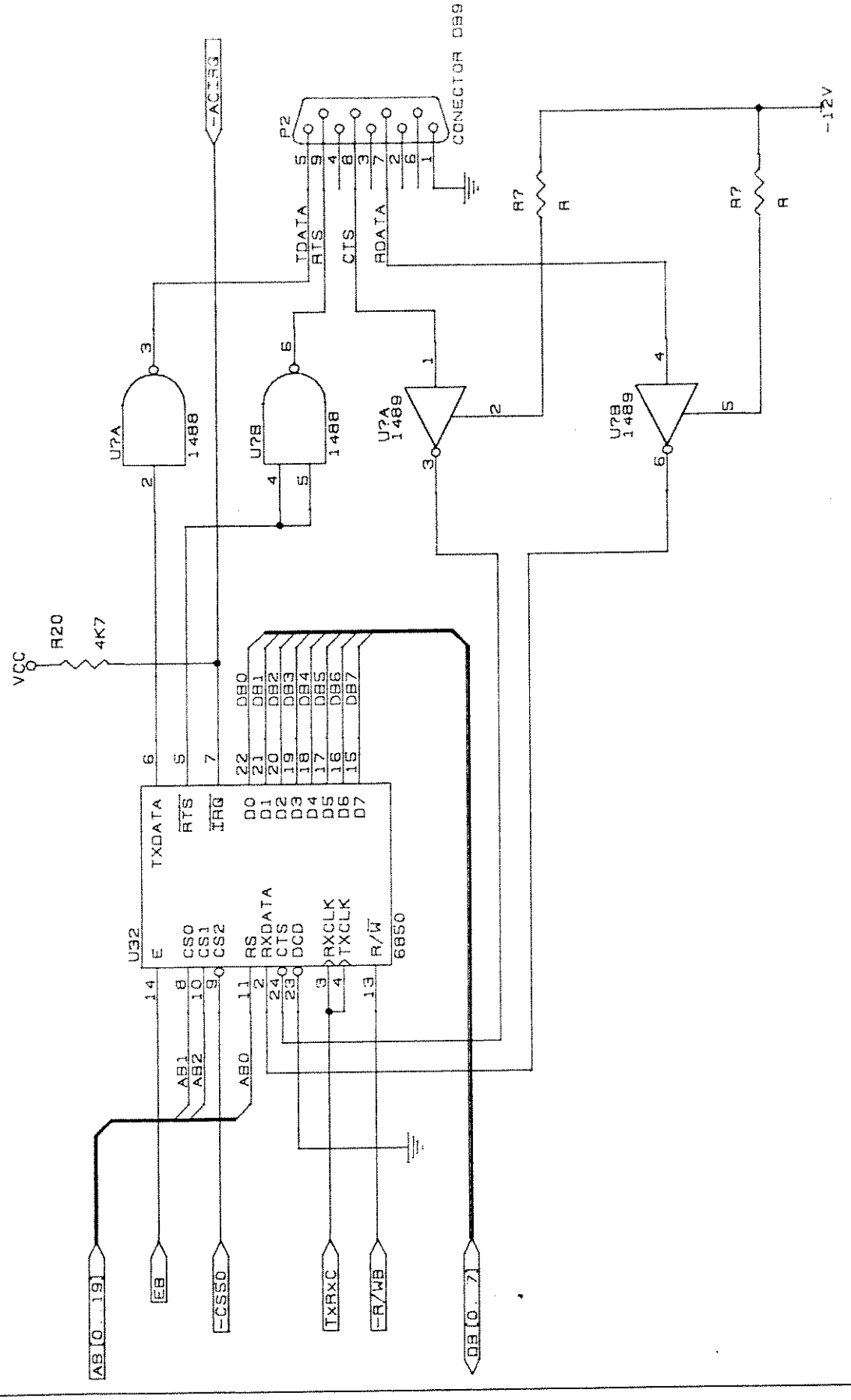


Figura 5.14 - Interface Serial

mostrado na figura 5.15. Existe previsão para até 128 Kbytes de memória EPROM, sendo que podem ser montados na placa dois dispositivos do tipo 2764, 27128, 27256, ou 27512. Estes dispositivos são selecionáveis através do jumper JP5.

5.2.8. Interface VME e árbitro

A interface com o barramento VME e o árbitro formam o subsistema responsável pela resolução de conflitos nos acessos à memória "dual port" de comunicação, uma vez que esta memória pode ser acessada tanto pelo barramento VME como pela CPU interna da placa (68008).

O circuito que implementa esta função é apresentado na figura 5.16. Neste circuito os componentes U1 e U2A formam o decodificador de endereços com que a memória irá responder aos acessos do barramento VME. Este endereço é selecionável em bancos de 64 K através da DIP-SWITCH SW1. O sinal gerado por este decodificador é o -SELEXT. Os componentes U3A, U3B e U5A, servem para gerar interrupção na CPU interna, quando forem escritos dados no endereço correspondente ao banco de 64K da memória e com o bit de endereço A14 do barramento VME com nível lógico 0. Esta interrupção não será gerada para um acesso normal, ou seja com A14 em nível lógico 1.

Os sinais +SELEXT e +SELINT, este último proveniente da lógica de decodificação interna à placa, são direcionados ao componente U2B. O circuito formado por U2B, U4A e U4B executa a

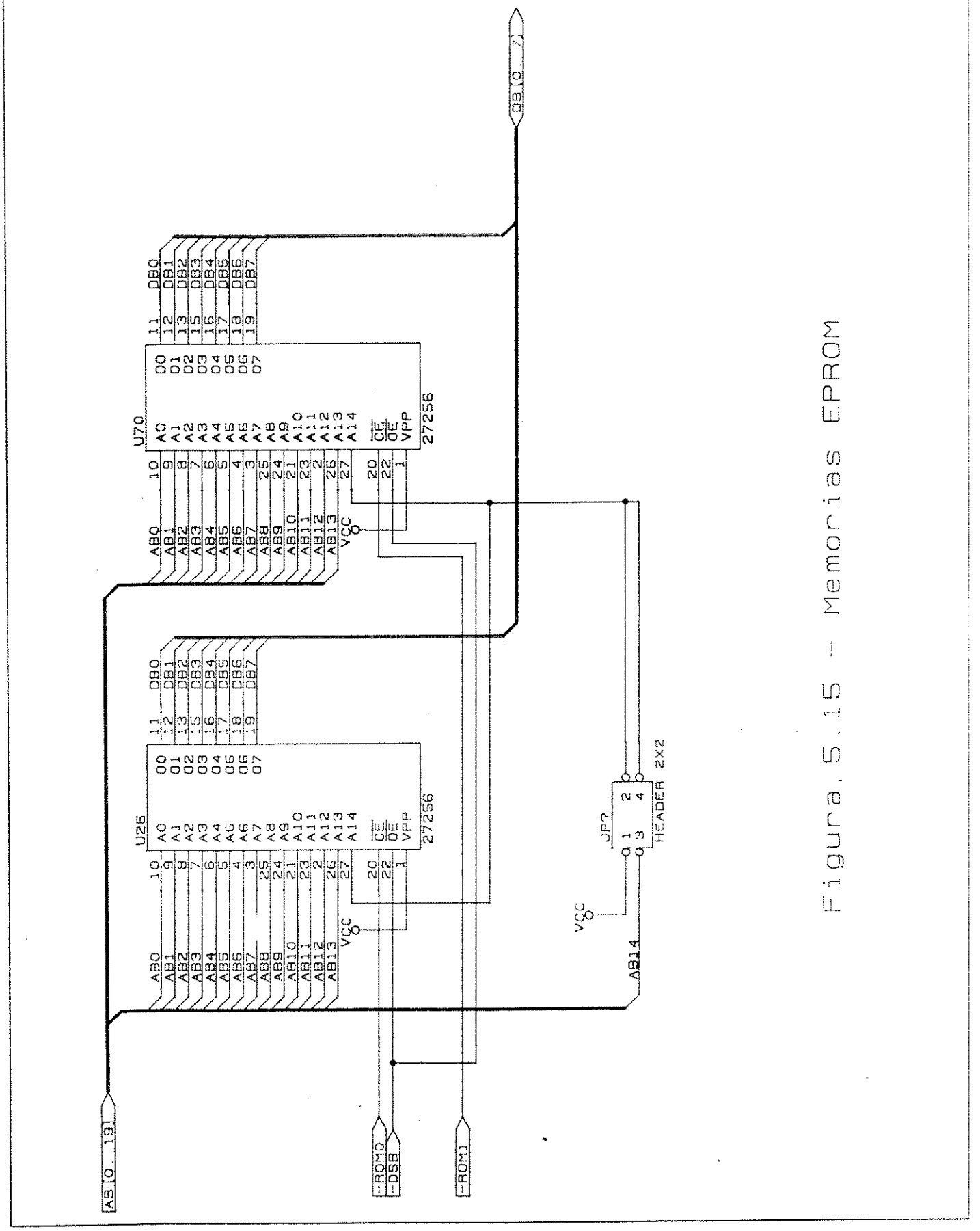


Figura 5.15 – Memórias EPROM

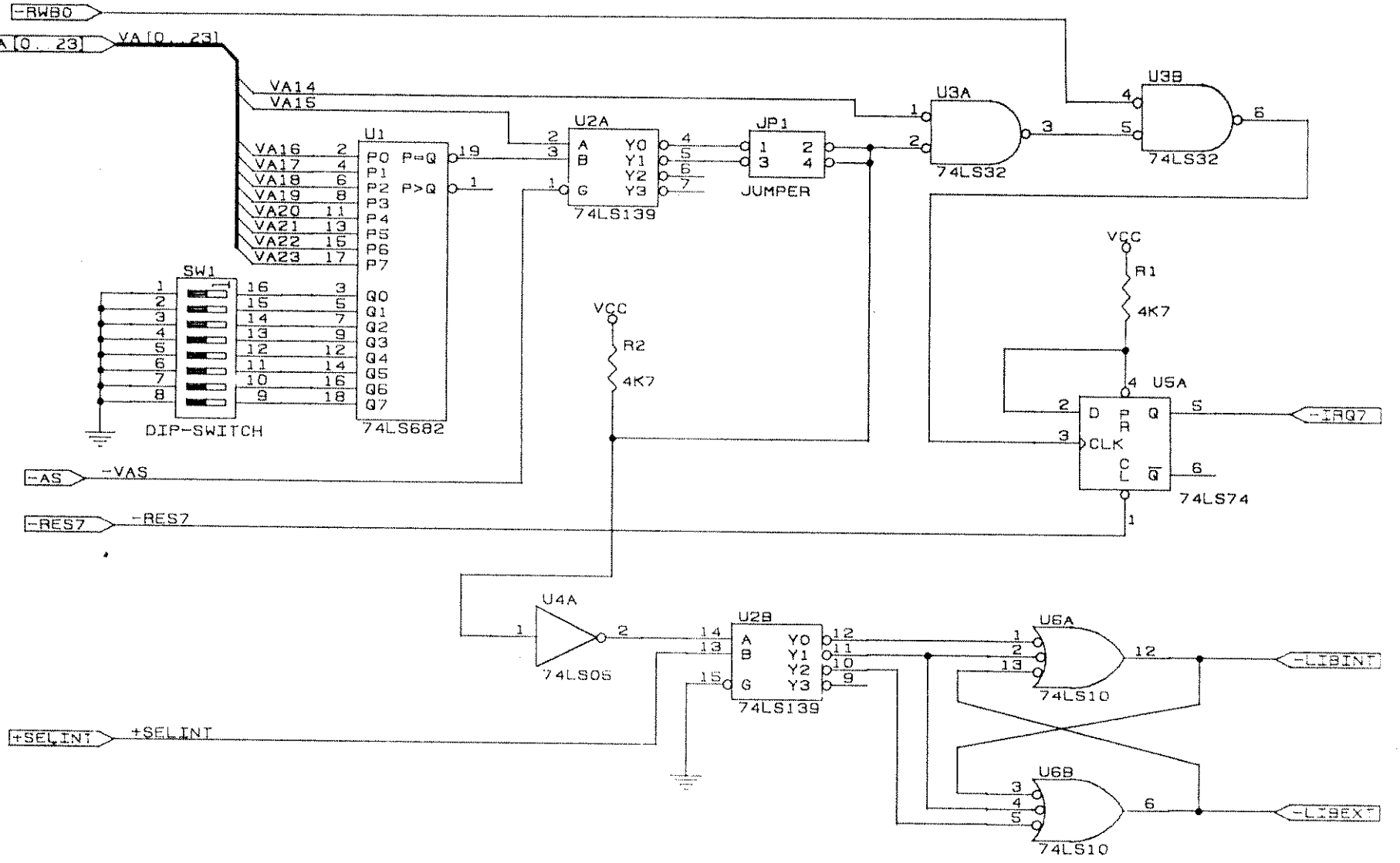


Figura 5.16 - Logica de Decodificacao do Barramento VME e Arbitro

função de arbitro. Assim este circuito libera os sinais -LIBINT e -LIBEXT de acordo com os sinais de entrada +SELINT e +SELEXT. No caso dos dois sinais de entrada irem para nivel lógico 1 ao mesmo tempo, um deles será atendido inicialmente, permanecendo o segundo em situação de espera até que o primeiro seja liberado.

Os sinais -LIBINT e -LIBEXT são responsáveis pela liberação dos buffers de acesso à memória "dual port" de comunicação e pela lógica de geração de DTACK para a CPU interna e para o barramento VME. Na figura 5.17 é mostrado o diagrama da lógica de geração do sinal de DTACK para o barramento VME e os buffers que controlam o acesso à memória de comunicação pelo mesmo.

No circuito da figura 5.17 temos os seguintes sinais:

VA1-VA23 - sinais de endereço do barramento VME.

VLDS - strobe de dados do barramento VME.

VR/W - sinal que indica ciclo de leitura ou escrita do barramento VME.

VD0-VD7 - sinais de dados do barramento VME.

AA0-AA12 - barramento de endereços para a memória "dual port".

DD0-DD7 - barramento de dados para a memória "dual port".

CSL - sinal de seleção para o componente de memória RAM "dual port".

RWL - sinal de leitura ou escrita para o componente de memória RAM "dual port".

RWBO - sinal para a lógica de interrupção da CPU interna pelo barramento VME.

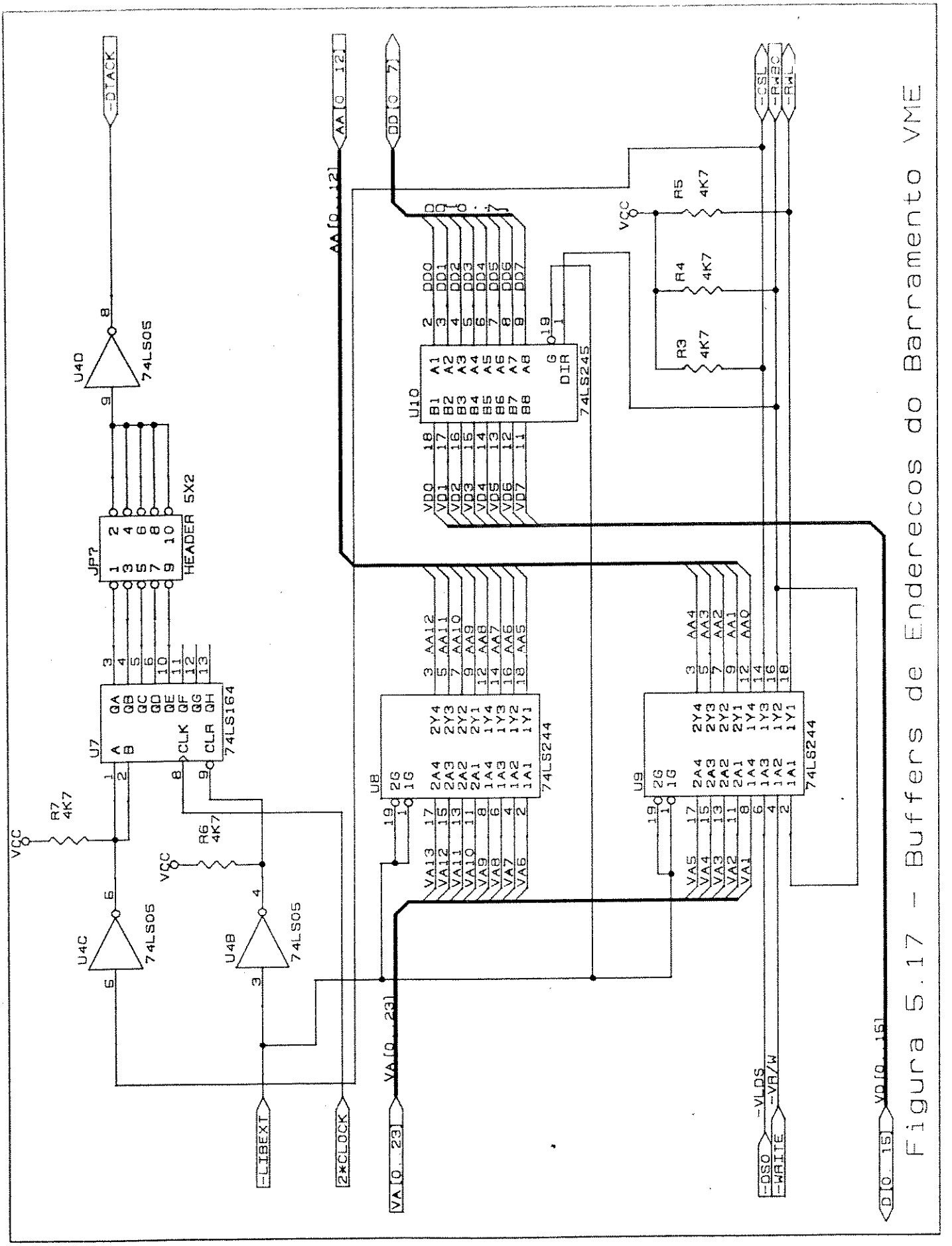


Figura 5.17 - Buffers de Enderecos do Barramento VME

O sinal LIBEXT proveniente do arbitro, libera os buffers de endereços e dados, ao mesmo tempo que, em conjunto com o sinal CSL, liberam a geração do sinal DTACK para o barramento VME a partir do componente U7. O tempo para a geração do sinal DTACK é selecionável pelo jumper JP1.

5.2.9. Lógica de decodificação dos endereços da CPU interna

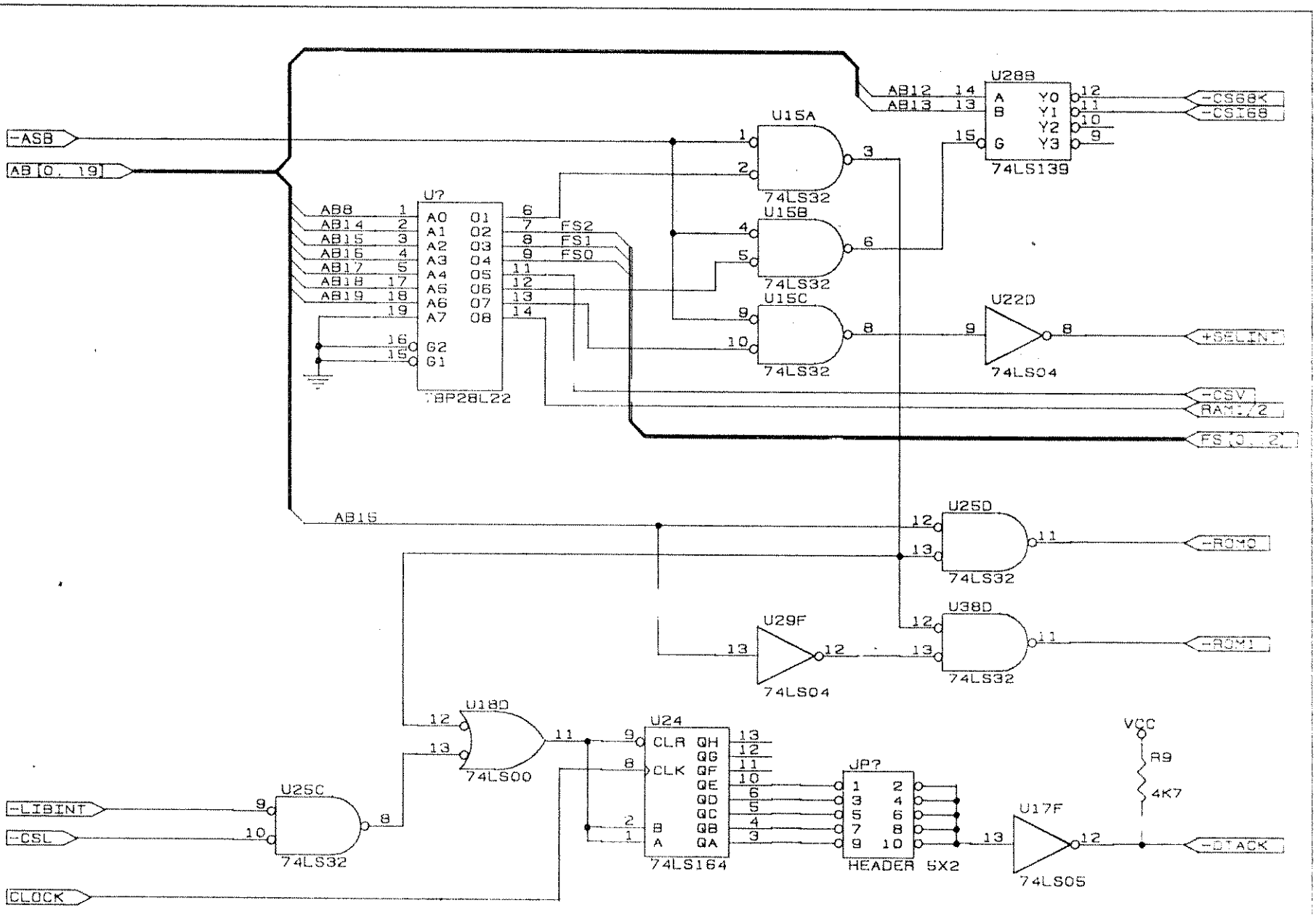
A lógica de decodificação de endereços é responsável pela geração dos sinais necessários para seleção dos componentes do sistema. Na figura 5.18 podemos verificar a parte do circuito implementado para esta função. O principal componente presente nesta lógica é a PROM com lógica TTL de alta velocidade, TBP28L22, a qual gera os sinais de seleção a partir dos sinais de endereço da CPU. Os sinais gerados são:

CS68K - sinal para seleção de componentes periféricos da família 68000 (não foi utilizado no projeto).

CSI68 - sinal responsável pela selecão de componentes periféricos da família 68000. A partir deste sinal são gerados os sinais VPA e CS50, este último responsável pela seleção do 6850 (interface de comunicação serial).

+SELINT - sinal responsável pela seleção da memória de comunicação entre a CPU interna da placa e a CPU do barramento VME.

CSV - sinal responsável pela seleção do controlador de vídeo. Em conjunto com os sinais RAM1/2 e FS0-FS3



Figrua 5.18 - Logica de Decodificacao de Enderecos e Geracao do Sinal DTACK

executam a seleção de memória de video bancos 1 e 2, memória de sistema, registradores internos do controlador de vídeo, carga dos registradores de deslocamento existentes nas RAM de vídeo.

ROM0 e ROM1 - sinais responsáveis pelo acesso às memórias tipo EPROM.

Deve-se notar que para os acessos aos componentes da familia 68000, estes já dispõem de lógica interna para a geração do sinal DTACK. Os acessos à familia 6800 são sincronos, o que é sinalizado pelo sinal VPA, não necessitando portanto da geração do sinal de DTACK. O controlador de vídeo por sua vez também gera o sinal de DTACK, a partir do sinal READY.

Resta portanto a geração do sinal de DTACK para acessos à memória EPROM e a memória de comunicação. Este circuito é composto pelos componentes U25C, U18D, U24 e U17F. O jumper JP4 deve ser ajustado para a geração do sinal de DTACK de acordo com o componente mais lento entre as memórias EPROM e RAM que é utilizada como memória de comunicação

5.2.10. Vantagens apresentadas

Como já foi mostrado anteriormente, a grande vantagem deste terminal é a disponibilidade da memória para acessos da CPU. Isto é conseguido graças à presença dos registradores de deslocamento de 256 bits nas vídeo RAM's. Com a disponibilidade de 96,8%, a CPU pode trabalhar com sua eficiencia máxima, praticamente não

existindo o problema dos conflitos de acesso à memória.

5.2.11. Desvantagens

A principal desvantagem apresentada é a mesma do terminal anterior, ou seja, o acesso a pixel individualmente não é possível, sendo que neste caso são acessados 2 pixels de cada vez, uma vez que a CPU possui somente 8 linhas de dados.

5.2.12. Firmware implementado

O software residente na placa (firmware) é responsável por sua inicialização e pela recepção, análise e execução dos comandos gráficos. O objetivo principal deste firmware é simplificar ao máximo a interface com o sistema de forma a minimizar a sobre carga introduzida pela placa.

No capitulo 6 será detalhado este software, com a apresentação do seu diagrama de blocos e de alguns dos algoritmos implementados.

6. FIRMWARE IMPLEMENTAOO

Para o funcionamento da placa gráfica foi desenvolvido um software residente em memória EPROM (firmware), responsável pelas funções de inicialização, recepção, análise e execução dos comandos, conforme mostrado na figura 6.1. O objetivo principal deste software é simplificar ao máximo a interface com o sistema hospedeiro de modo a minimizar a sobrecarga introduzida pela placa de vídeo.

Cerca de 90% do código foi implementado através da Linguagem C, e o restante foi implementado em "Assembler" do 68000.

Todo o software foi desenvolvido com auxilio da Unidade de Emulação 8540, do sistema de desenvolvimento 8561, da Tektronix, ocupando atualmente cerca de 40Kbytes de programa.

A estrutura modular associada a implementação em linguagem de alto nível são responsáveis por uma característica marcante do software desenvolvido: a transportabilidade. A experiência adquirida com o seu desenvolvimento possibilitou seu transporte para os seguintes ambientes:

- IBM PC, onde atua como uma biblioteca de funções gráficas (pacote NUGRA);
- Sistema VAX/VMS-785 com terminal de vídeo de 640x480 pixels como apoio para simulação de células flexiveis de manufatura.

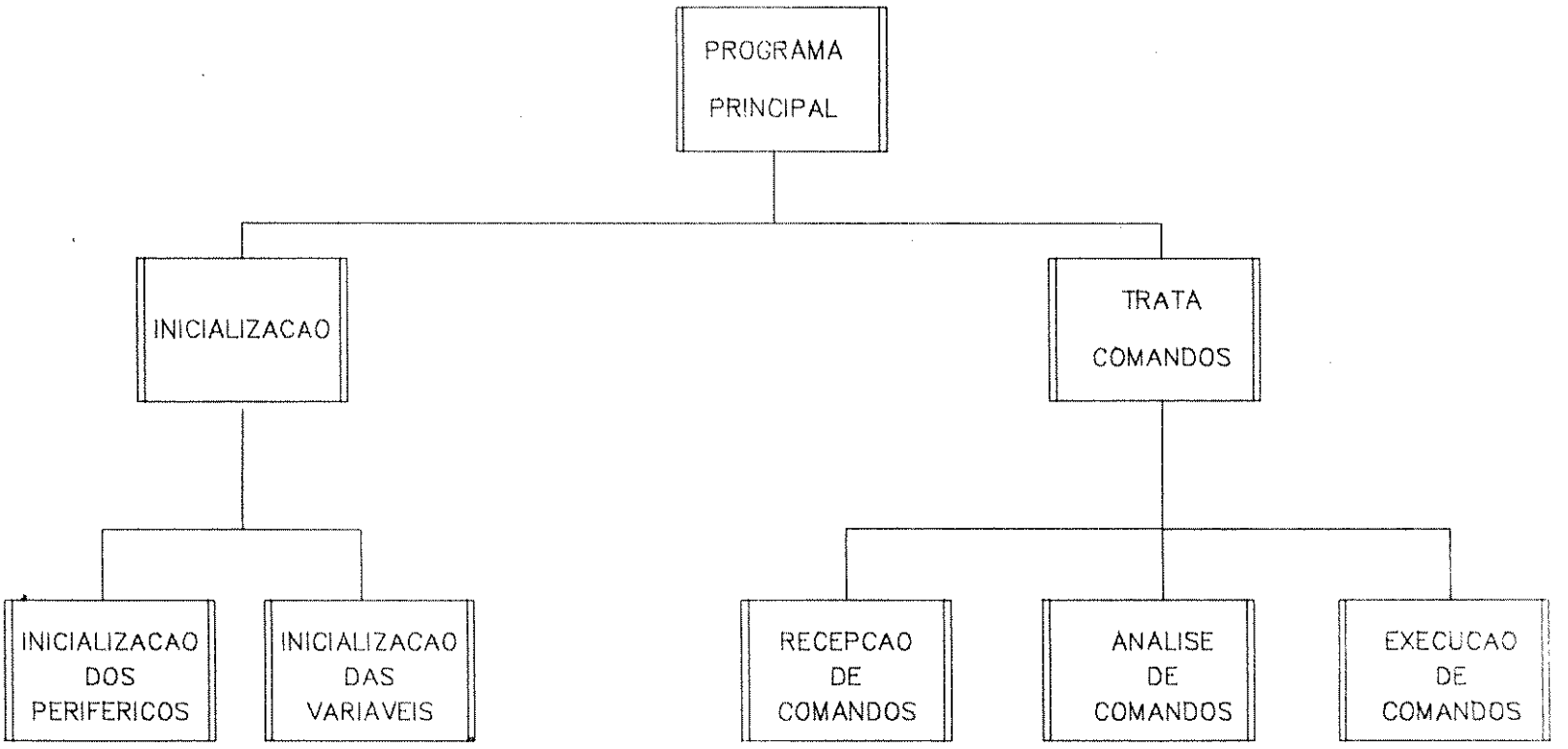


FIGURA 6.1 – ESTRUTURA DO SOFTWARE

- Sistema HOMUK (UNICAMP), onde contrala a placa gráfica PIG01/02 emulando o terminal gráfico Tektronix 4107, com algumas extensões.

Nos próximos itens são apresentados os módulos componentes do firmware, caracteristicas da utilização do núcleo, comandos gráficos e alguns algoritmos utilizados para implementação de primitivas gráficas.

6.1. Módulos do firmware

A seguir são apresentados de modo simplificado os principais módulos do software implementado.

6.1.1. Inicialização

O módulo de inicialização está dividido em duas tarefas: inicialização dos periféricos e inicialização das variáveis. No caso dos periféricos são inicializados a memória "dual port", para indicar que a placa está apta a receber comandos; além disso é inicializado também o controlador de vídeo.

A inicialização de variáveis visa colocar os parâmetros "default" para todas as primitivas gráficas.

6.1.2. Recepção de comandos

A interface com o sistema hospedeiro, como já foi dito, é realizada através de uma memória "dual port" dividida lógicamente em dois blocos: comandos e controle. A inserção de um comando e uma indicação na região de controle ativa o módulo de recepção de comandos (figura 6.2). Este módulo retira o comando da memória "dual port" e indica na região de controle que a execução de um comando está em andamento, estando portanto a placa ocupada. Após o comando ter sido executado, o sistema indica que a placa está pronta para receber outro comando através da palavra de "status", e opcionalmente pode indicar a ocorrência de um erro ou preencher a área de respostas com os elementos associados ao comando (figura 6.3).

6.1.3. Análise dos comandos

No módulo de análise (figura 6.4), o comando é decomposto e analisado sintaticamente (para verificar a validade do comando), e semanticamente (verificar se os parâmetros estão dentro da faixa estabelecida para aquele comando). No caso da ocorrência de erro, o comando está cancelado e o usuário imediatamente informado através das palavras de "status" e erro.

6.1.4. Execução dos comandos

Após a análise, o comando está num formato apropriado para a execução (figura6.5), sendo então ativado o módulo de execução

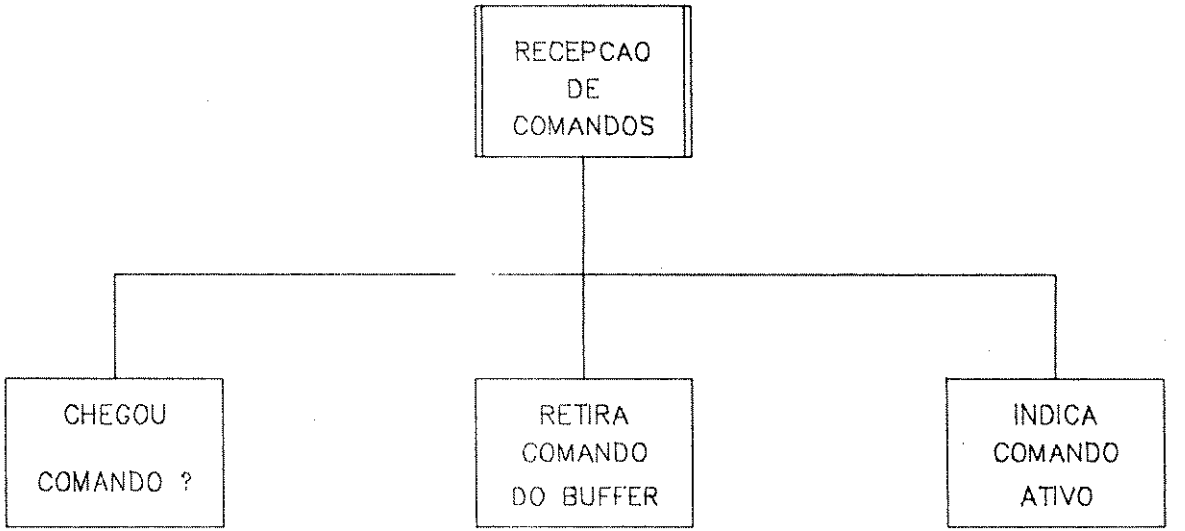


FIGURA 6.2 – RECEPCAO DE COMANDOS

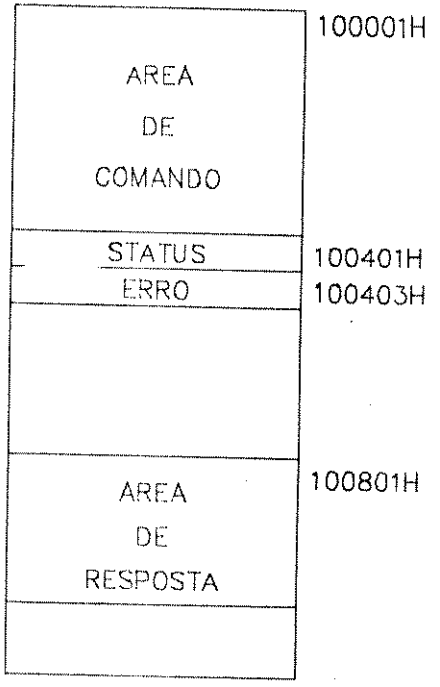


FIGURA 6.3 – REGIAO DE COMANDO DA MEMORIA "DUAL PORT"

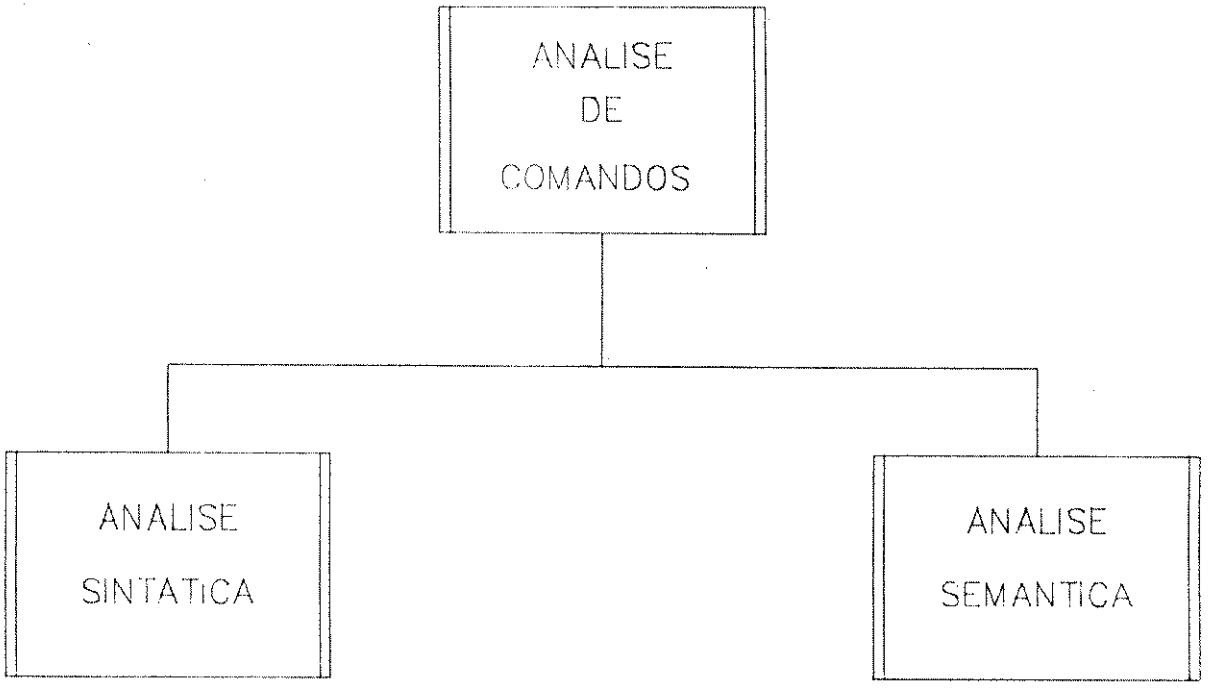


FIGURA 6.4 – ANALISE DE COMANDOS

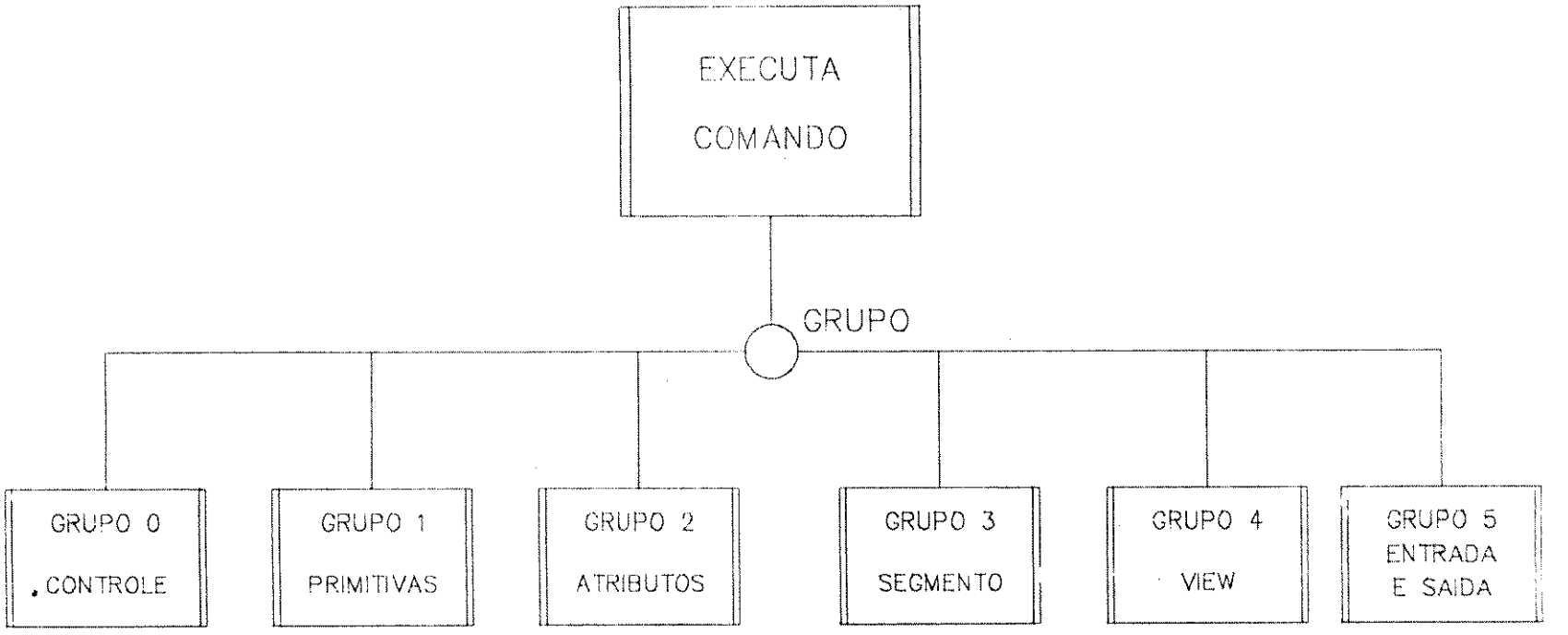


FIGURA 6.5 – EXECUTA COMANDO

das tarefas correspondentes para a geração de imagens ou armazenamento de parâmetros correspondentes.

6.2. Implementação no sistema VME

A placa gráfica desenvolvida foi instalada em um sistema VME da "MOTOROLA Semiconductor Products Inc." com as seguintes caracteristicas:

- Placa de CPU MVME110-1 com processador 68000 operando com frequência de 8Mhz.
- Placa de memória MVME202 com 512 Kbytes.
- Placa controladora de unidades de disco tipo Winchester e Floppy M68RWTN1.
- 2 Unidades de "floppy disk" 5 1/4"
- 1 Unidade tipo "winchester" 10 Mbytes

O sistema operacional utilizado é o Versados Versão 4.4, o qual é orientado para o desenvolvimento de sistemas baseados em microprocessdores, bem como a execução de sistemas de aplicação de Tempo Real com multiprogramação. Neste contexto, o esquema de utilização do software gráfico pode ser observado na figura 6.6.

Toda a interface que o aplicativo possue em relação ao software gráfico está centrada na memória "dual port" da placa de video; a eficiência alcançada com esta implementação libera o programa aplicativo do grande volume de processamento exigido pela manutenção das saídas gráficas.

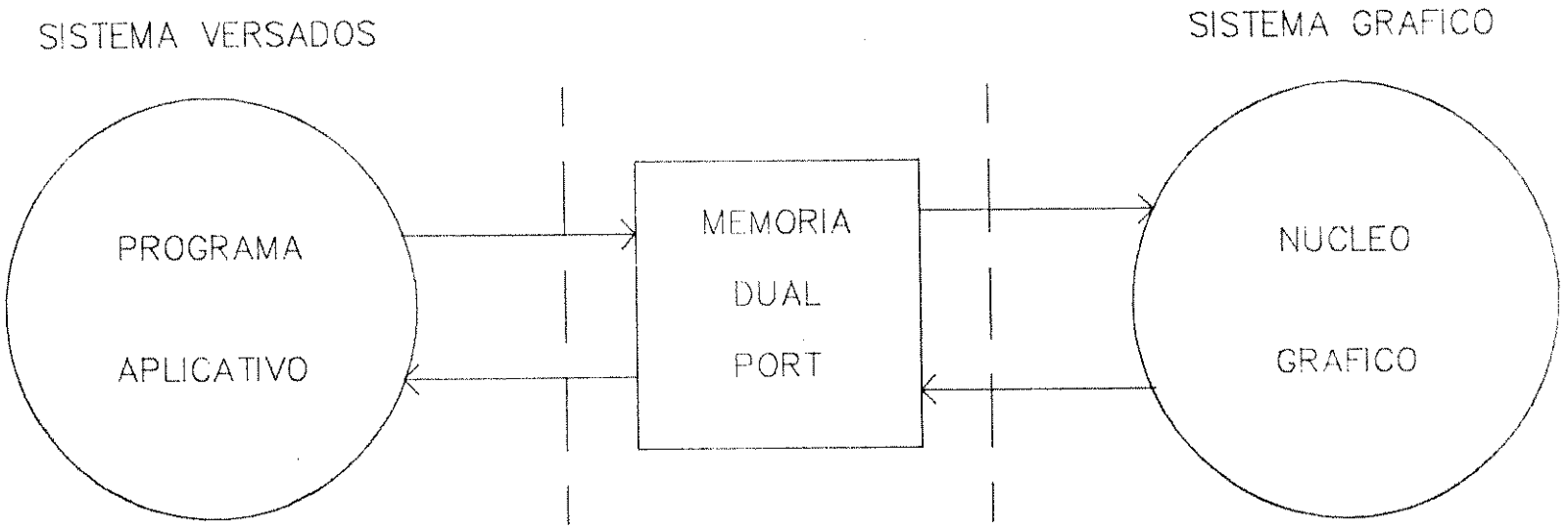


FIGURA 6.6 – ESTRUTURA DO SOFTWARE IMPLEMENTADO

A título ilustrativo, foi implementado um programa aplicativo escrito em Linguagem PASCAL (68K/010 Pascal 2.3 da MOTOROLA), que adquiria comandos gráficos de uma base residente em disco, e os enviava a placa de vídeo formando uma sucessão de telas num Monitor colorido NEC MULTISYNC modelo JC140P3A. Este conjunto foi apresentado na feira da SUCESSU 87 em São Paulo.

6.3. Comandos gráficos

A descrição dos comandos possue a seguinte sintaxe:

comando [parâmetros .....]

onde:

comando - identifica o comando,

parâmetros - são itens de dados especificos associados a cada comando.

Os comandos gráficos implementados foram divididos nos seguintes grupos:

0 : comandos de controle (0-29)

1 : primitivas gráficas (30-59)

2 : atualização de atributos (60-89)

3 : controle de segmentos (90-119)

4 : controle de viewing (120-149)

5 : entrada (150-179)

O arranjo **COM** que aparece na descrição dos comandos representa o início da area de comandos na memória "dual port"

6.3.1 Comandos de Controle

a - apaga tela (APGTEL)

descrição: limpa o display

código: 00

array de inteiros: com[0]=0

6.3.2. Primitivas gráficas

As primitivas gráficas estabelecem a representação de um objeto na forma gráfica. As seguintes primitivas estão atualmente implementadas:

a - ponto absoluto (PTABS)

descrição: acende pixel na posição absoluta (x,y)

código: 30

array de inteiros: com[0] = 30

com[1] = x

com[2] = y

b - ponto relativo (PTREL)

descrição: acende um pixel deslocado de dx,dy em relação à posição atual do cursor

código: 31

array de inteiros: com[0] = 31

com[1] = dx

com[2] = dy

c - move absoluto (MVABS)

descrição: executa um movimento absoluto do cursor para o ponto (x,y)

código: 32

array de inteiros: com[0] = 32

com[1] = x

com[2] = y

d - move relativo (MVREL)

descrição: move o cursor dx,dy em relação à posição atual do cursor

código: 33

array de inteiros: com[0] = 33

com[1] = dx

com[2] = dy

e - linha absoluta (LNABS)

descrição: traça um segmento de reta delimitado por x1,y1 e x2,y2

código: 34

array de inteiros: com[0] = 34

com[1] = x1

com[2] = y1

com[3] = x2

com[4] = y2

f - linha relativa (LNREL)

descrição: traça um segmento de reta delimitado pela posição atual do cursor (cpx, cpy) e por cpx+x e cpy+y

código: 35

array de inteiros: com[0] = 35

com[1] = x

com[2] = y

g - poligono (POL)

descrição: define um polígono de n vértices. O número maximo

de vértices permitido é 98

código: 36

array de inteiros: com[0] = 36

com[1] = n

com[2] = x1

com[3] = y1

⋮

com[m] = xn-1

com[m+1] = yn-1

h - texto (TEXT)

descrição: escreve um texto a partir da posição x,y. O texto resultante é dependente dos atributos associados a texto que estão atualmente ativos

código: 37

array de inteiros: com[0] = 37

com[1] = tamanho do texto

com[2] = char1

com[3] = char2

⋮

com[n] = charn

i - circulo/elipse (CIRC1)

descrição: executa um circulo (se os raios forem iguais) ou uma elipse com centro xc,yc e raio no eixo x=a e no eixo y=b

código: 38

array de inteiros: com[0] = 38

com[1] = xc

com[2] = yc

com[3] = a

com[4] = b

6.3.3. Atualização de atributos

Os atributos são elementos que descrevem as características das primitivas gráficas de saída. Os seguintes atributos estão atualmente implementados:

a - ativa preenchimento (BEGINPANEL)

descrição: ativa o preenchimento de áreas como polígono, círculo e elipse

código: 60

array de inteiros: com[0] = 60

b - cor preenchimento (CORPREEN)

descrição: estabelece a cor para o preenchimento de áreas

código: 61

array de inteiros: com[0] = 61

com[1] = cor

c - rotação do texto (ROTTEXTO)

descrição: estabelece angulo de rotação para textos

código: 62

array de inteiros: com[0] = 62

com[1] = ângulo

d - direção do texto (DIRTEXTO)

descrição: estabelece a direção do texto

0 = para cima

1 = para baixo

2 = para direita

3 = para esquerda

código: 63

array de inteiros: com[0] = 63

com[1] = direção

e - tamanho do texto (TAMTEXTO)

descrição: estabelece o tamanho do caractere

código: 64

array de inteiros: com[0] = 64

com[1] = largura

com[2] = altura

com[3] = espaçamento

f - cor do texto (CORTEXTO)

descrição: estabelece a cor do caractere

código: 65

array de inteiros: com[0] = 65

com[1] = cor do texto

g - cor da linha (CORLINHA)

descrição: estabelece a cor da linha a ser traçada

código: 66

array de inteiros: com[0] = 66

com[1] = cor da linha

h - desativo preenchimento (ENDPANEL)

descrição: desativa o preenchimento de areas

código: 67

array de inteiros: com[0] = 67

i - borda para primitiva (PBORDA)

descrição: indica a existência e a cor da borda para

primitivas com areas preenchidas

código: 68

array de inteiros: com[0] = 68

com[1] = 0 - sem borda, 1 - com borda

com[2] = cor da borda

j - mapa de cores (MAPAPALL)

descrição: estabelece para um dado indice de cor, uma nova coloração, brilho e saturação segundo o modelo HLS

código: 69

array de inteiros: com[0] = 69

com[1] = indice

com[2] = coloração (valor entre 0 e 360)

com[3] = brilho (valor entre 0 e 100%)

com[4] = saturação (valor entre 0 e 100)

k - tipo da linha (TIPLIN)

descrição: determina o tipo de linha. São 5 os tipos disponíveis:

0 - contínuo

1 - pontilhado

2 - traço-ponto

3 - tracejado

4 - tracejado longo

código: 70

array de inteiros: com[0] = 70

com[1] = tipo de linha

6.3.4. Controle de segmentos

O segmento é uma entidade onde se define um objeto e respectivos atributos através do agrupamento de primitivas gráficas de saída. Primitivas nos segmentos não são manipuláveis, sendo somente possível a manipulação do segmento. Desta forma são possiveis a deleção, alteração de atributos, translação e rotação de segmentos. Atualmente estão implementadas as seguintes operações sobre segmentos.

a - abre segmento (ABRESEG)

descrição: abre um novo segmento. O nome do segmento é identificado por um inteiro entre 1 - 32767

código: 90

array de inteiros: com[0] = 90

com[1] = número do segmento

b - fecha segmento (SEGFECHA)

descrição: fecha um segmento aberto

código: 91

array de inteiros: com[0] = 91

c - ponto pivo segmento (PIVOTSEG)

descrição: define um ponto de origem para um dado segmento. Este ponto pode ser alterado mesmo que o segmento já tenha sido identificado

código: 92

array de inteiros: com[0] = 92

com[1] = número do segmento

com[2] = x

com[3] = y

d - deleta segmento (DELSEG)

descrição: apaga um segmento especifico. Quando o número de segmentos for igual a -1 implica em remover todos os segmentos existentes

código: 93

array de inteiros: com[0] = 93

com[1] = número do segmento

e - renomeia segmento (RENAMESEG)

descrição: o segmento antigo passa a ser identificado por um novo nome. O identificador do segmento antigo torna-se novamente disponível para ser utilizado em uma nova operação de abre segmento

código: 94

array de inteiros: com[0] = 94

com[1] = nome antigo

com[2] = nome novo

f - visibilidade do segmento (VISIBSEG)

descrição: estabelece visibilidade para segmento

0 - invisivel

1 - visivel

2 - inverte estado atual

o número do segmento -1 indica todos os segmentos

código: 95

array de inteiros: com[0] = 95

com[1] = número do segmento

com[2] = código de visibilidade

g - escala segmento (ESCALASEG)

descrição: escala um segmento de x em relação à abcissa e y em relação à ordenada

código: 96

array de inteiros: com[0] = 96

com[1] = número do segmento

com[2] = escala x

com[3] = escala y

h - rotacionar segmento (ROTSEG)

descrição: rotacionar um segmento de n graus

código: 97

array de inteiros: com[0] = 97

com[1] = número do segmento

com[2] = ângulo de rotação

i - translada segmento (TRANSLSEG)

descrição: tranlada segmento para a posição x,y. Nesta operação o ponto pivô do segmento sempre vai coincidir com o ponto x,y

código: 98

array de inteiros: com[0] = 98

com[1] = número do segmento

com[2] = posição x

com[3] = posição y

j - tranforma segmento (TRANSFSEG)

descrição: opera as três tranformações (escalonamento, rotação e translação) ao mesmo tempo sobre um segmento específico

código: 99

array de inteiros: com[0] = 99

com[1] = número do segmento

com[2] = ângulo de rotação

com[3] = escala x

com[4] = escala y

com[5] = deslocamento x

com[6] = deslocamento y

k - cópia segmento (COPIASEG)

descrição: inclue cópia de um segmento existente em um segmento aberto

código: 100

array de inteiros: com[0] = 100

com[1] = número do segmento a ser copiado

6.3.5. Controle de "view"

"Views" são telas lógicas que permitem o agrupamento de segmentos relacionados. A cada tela está associado, além dos atributos, um mapeamento de coordenadas do mundo ("window") em coordenadas do dispositivo de saída ("viewport"). Os seguintes comandos relacionados com "views" estão atualmente implementados:

a - seleciona view (SELVIEW)

descrição: defina e ativa uma view especificada. A view é identificada por um inteiro entre 1 e 64

código: 120

array de inteiros: com[0] = 120

com[1] = número do segmento a ser

b - redesenha view (REDESVIEW)

descrição: redesenha a view. Se o número da view for 0, significa view corrente, se for -1 significa todas as views

código: 121

array de inteiros: com[0] = 121

com[1] = número da view

c - deleta view (DELVIEW)

descrição: deleta view especificada. Se view = -1 implica em deletar todas as views

código: 122

array de inteiros: com[0] = 122

com[1] = número da view

d - ativar borda de view (BORDAVIEW)

descrição: ativar borda para view corrente

código: 123

array de inteiros: com[0] = 123

com[1] = 0 - sem borda, 1 - com borda

e - setar cor para view (CORVIEW)

descrição: estabelece atributo de cor de fundo e cor de borda para view corrente

código: 124

array de inteiros: com[0] = 124

com[1] = cor de fundo

com[2] = cor de borda

f - viewport para view (DVIEWPORT)

descrição: define viewport para view corrente

código: 125

array de inteiros: com[0] = 125

com[1] = x1

com[2] = y1

com[3] = x2

com[4] = y2

g - window para view (DWINDOW)

descrição: estabelece window para view corrente

código: 126

array de inteiros: com[0] = 126

com[1] = x1

com[2] = y1

com[3] = x2

com[4] = y2

6.3.6 Entrada

a - ativa cursor (ATCURSOR)

descrição: ativa cursor para view corrente, movimenta o cursor (para cima, baixo, esquerda e direita), obtem novo ponto na tela e desativa cursor. Ao final do comando, a posição do cursor pode ser obtida na área de resposta na memória "dual port"

código: 150

array de inteiros: com[0] = 150

6.4. Algoritmos básicos implementados

A seguir são apresentados os algoritmos implementados para as funções basicas. Para facilidade de compreensão estão apresentadas em linguagem "C", devendo ser salientado que estas foram implementadas em "Assembler". Observe que outras funções foram básicas, como por exemplo círculo, arco, texto, que estão implementadas no núcleo gráfico e que foram escritas em linguagem de alto nível podem ser reescritas para utilização de possíveis facilidades implementadas em hardware.

```
/* ESTE MODULO E RESPONSAVEL PELA GERACAO DE UMA LINHA  RETA
   E PELA ATIVACAO DE UM PIXEL.

   A FUNCAO A SER CHAMADA E reta(xl,yl,x2,y2) ONDE (xl,yl) e
   (x2,y2) SAO OS PONTOS INICIAIS E FINAIS DA RETA RESPECTI-
   MENTE.
   ALEM DE TRACAR A RETA ,SAO EXECUTADAS TAMBEM OPERACAOES /
   DE CLIPPING E VIEWING.
   PARA QUE ESTAS OPERACAOES SEJAM  EXECUTADAS  CORRETAMENTE
   E NECESSARIO QUE AS SEGUINTES VARIAVEIS SEJAM GERADAS EX-
   TERNAMENTE:
   (wxl,wyl) ponto minimo da window
   (wx2,wy2) ponto maximo da window
   (vxl,vyl) ponto minimo da view
   (vx2,vy2) ponto maximo da view
   sx : fator de escalonamento x na transformacao de window
        para view deve ser gerada externamente   na  recepcao
        de um comando view.
        Esta aproximacao foi tomada p/ maior rapidez sendo
        que: sx=1.0*(vx2-vxl)/(wx2-wxl);
   sy : identico a sx sendo que:
        sy=1.0*(vy2-vyl)/(wy2-wyl);

   NOTE QUE sx e sy sao float e o restante e int.
   DESTA FORMA ESTAO DISPONIVEIS AS SEGUINTES ROTINAS:
   clipline(pxl,pyl,px2,py2) *efetua clipping de linha na view*
   clipixel(x,y) *efetua clipping do ponto na view*
   int *x,*y;

   viewpixel(x,y)  *efetua mapeamento window -> view*
   int *x,*y;

   reta(xl,yl,x2,y2) *efetua uma reta de (xl,yl) a (x2,y2)*
   int xl,yl,x2,y2;
*/

#include "def.h"

/* constantes */

#define LCODE 0x01      /* left of view code */
#define RCODE 0x02      /* rigth of view code */
#define BCODE 0x04      /* below view code */
#define TCODE 0x08      /* above view code */
#define MCODE 0x00      /* inside window code*/
```

Listagem 1

```
abs_int(i)
int i;
{
        /* valor absoluto de um numero inteiro */
        return((i > 0) ? i : i * -1);
}

clipline(px1,py1,px2,py2)
int *px1,*py1,*px2,*py2;
{
        unsigned int p1code,p2code;
        int x,y,x1,y1,x2,y2;
        int tmp,clipstat=FALSE;
        float fp1,fp2;

        x1=(*px1);
        y1=(*py1);
        x2=(*px2);
        y2=(*py2);

        do   /*line clipping*/
        {
            /* examina ponto final*/
            p2code=MCODE;
            if (x2 < wx1)
                    p2code= (p2code | LCODE);
            else if (x2 > wx2)
                    p2code= (p2code | RCODE);
            if (y2 < wy1)
                    p2code= (p2code | TCODE);
            else if (y2 > wy2)
                    p2code= (p2code | BCODE);

            /* examina ponto inicial*/
            p1code=MCODE;
            if (x1 < wx1)
                    p1code= (p1code | LCODE);
            else if (x1 > wx2)
                    p1code= (p1code | RCODE);
            if (y1 < wy1)
                    p1code= (p1code | TCODE);
            else if (y1 > wy2)
                    p1code= (p1code | BCODE);

            if ((p1code | p2code) == FALSE)
              /* set flag indicando reta dentro da view*/
```

Listagem 1 – Continuacao

```
            clipstat=TRUE;
        else if ((plcode & p2code) == FALSE)
        {
            /* se p1 no interior troque p1 e p2*/
            if (plcode==FALSE)
            {
                tmp=x1;
                x1=x2;
                x2=tmp;
                tmp=y1;
                y1=y2;
                y2=tmp;
                tmp=plcode;
                plcode=p2code;
                p2code=tmp;
            }

            /*cruza borda esquerda ?*/
            if ((plcode & LCODE) != FALSE)
            {
                fp1=1.0*(y2-y1)*(wx1-x1);
                fp2=1.0*(x2-x1);
                y=y1+(fp1/fp2);
                x=wx1;
            }
            /*cruza borda direita ?*/
            else if ((plcode & RCODE) != FALSE)
            {
                fp1=1.0*(y2-y1)*(wx2-x1);
                fp2=1.0*(x2-x1);
                y=y1+(fp1/fp2);
                x=wx2;
            }
            /* cruza borda superior*/
            else if ((plcode & BCODE) != FALSE)
            {
                fp1=1.0*(x2-x1)*(wy2-y1);
                fp2=1.0*(y2-y1);
                x=x1+(fp1/fp2);
                y=wy2;
            }
            /* cruze borda inferior*/
```

Listagem 1 – Continuacao

```
                        else if ((p1code & TCODE) != FALSE)
                        {
                                fp1=1.0*(x2-x1)*(wy1-y1);
                                fp2=1.0*(y2-y1);
                                x=x1+(fp1/fp2);
                                y=wy1;
                        }
                        x1=x;
                        y1=y;
                }

        }
        while ((clipstat==FALSE) && ((p1code & p2code) == FALSE));
        /* enquanto linha nao esta dentro da view ou total-
                mente fora*/

        if (clipstat)
        {
                /* realiza mapeamento window -> view*/
                x1=sx*(x1-wx1) + vx1;
                x2=sx*(x2-wx1) + vx1;
                y1=sy * (y1-wy1) + vy1;
                y2= sy * (y2-wy1) + vy1;
                /* retorne coordenadas da reta e clipping status*/
                *px1=x1;
                *py1=y1;
                *px2=x2;
                *py2=y2;
        }
        return(clipstat);
}

reta(x1,y_1,x2,y2)
int x1,y_1,x2,y2;
{
        int abs_int(),x,y,i,t1,t2,d,a,b,dx,dy;

        if (clipline(&x1,&y_1,&x2,&y2))
        {
                a=abs_int((x2-x1));
                b=abs_int((y2-y_1));

                if ((x2-x1) > 0)
                        dx=1;
                else
                        dx=(-1);
```

Listagem 1 – Continuacao

```
                if ((y2-y_1)>0)
                        dy=1;
                else
                        dy=(-1);

                x=x1;
                y=y_1;

                t1=2*a;
                t2=2*b;
                d=b-a;


                for (i=0;i<(a+b);i++)
                {
                        marca(x,y);
                        if (d > 0)
                        {
                                y+=dy;
                                d-=t1;
                        }
                        else
                        {
                                x+=dx;
                                d+=t2;
                        }
                }
        }
}

clipixel(x,y)
int *x,*y;
/*esta rotina efetuara o clipping do pixel sobre a view
  e ativara o pixel correspondente */
{
        if (*y > vy1 && *y < vy2)
                if (*x > vx1 && *x < vx2) marca (*x,*y);
}

viewpixel(x,y)
int *x,*y;
/*esta rotina efetuara a transformacao window -> view
  sob o ponto (x,y) */
{
        *x=sx * (*x-wx1) + vx1;
        *y=sy * (*y-wy1) + vy1;
}
```

Listagem 1 – Continuacao

7. CONCLUSOES

Como resultado deste trabalho dispõe-se em laboratório do protótipo da placa gráfica projetada, implementada para resolução de 512x380 pixels, 16 cores simultâneas escolhidas dentro de um "pallet" de 4096 cores, taxa de regeneração de 60Hz. A restrição do protótipo com relação ao número de cores e resolução deve-se basicamente ao tamanho da placa utilizada e a densidade dos componentes de memória disponíveis à época da implementação. Mantendo-se a mesma arquitetura e com a utilização de componentes mais atuais, a resolução poderia ser expandida para 1024x1024 pixels e 256 cores simultâneas.

Paralelamente ao desenvolvimento do hardware, implementou-se um núcleo gráfico, residente em memória EPROM, para a execução das funções de inicialização, recepção, análise e execução de comandos. A disponibilidade de processamento na placa gráfica permitiu a incorporação ao núcleo gráfico de primitivas mais complexas como circulos, elipses, textos, bem como todas as funções de gerenciamento de segmentos, janela (windows) e viewports. Uma das caracteriscticas mais importantes do software desenvolvido é a sua transportabilidade/reusabilidade, que possibilita, pela escrita de alguns módulos utilitarios, sua implementação em ambientes gráficos diversos, como por exemplo as placas gráficas PIP e PIG, disponíveis no Laboratório de Computação Gráfica do DCA/FEE Unicamp, e que rodam hoje sob este núcleo gráfico.

É importante observar que, em projetos de desenvolvimento tecnlógico, a disponibilidade de componentes e ferramentas têm influência fundamental nas soluções adotadas. Este, também, é o caso deste projeto, em que muitas das soluções apresentadas, embora sub-ótimas, justificam-se principalmente pela disponibilidade de componentes e recursos.

As soluções adotadas mostraram-se apropriadas, e apresentaram bons resultados. O protótipo, apesar de ter sido implementado em "wire-wrapping", apresenta índice de falhas praticamente nulo e o Núcleo Gráfico desenvolvido apresenta excelente interface com o processador hospedeiro e alta adequação às mais diversas aplicações.

A utilização da técnica de "wire-wrapping" para a implementação do protótipo resultou em uma alta sensibilidade a ruidos, que foi eliminada pelo reforço dos condutores de alimentalção elétrica, bem como uso intensivo de componentes de desacoplamento.

Também foram encontrados dificuldades na seleção de um monitor de vídeo que respondesse adequadamente às frequências envolvidas no processo de regeneração da imagem. A solução foi obtida pela utilização de um monitor Multisync da NEC, que permite a sincronização automática em uma faixa de frequências relativamente larga. Prevê-se que para a implementação do terminal de alta resolução 1024x1024 pixels, o monitor de vídeo adqueada será a

principal restrição.

Avalia-se que a substituição do presente controlador de vídeo não deverá aumentar substancialmente o desempenho dá placa gráfica, o mesmo podendo-se dizer sobre o processador MC68008.

O desempenho apresentado pelo sistema (hardware + software) mostra-se perfeitamente adequado à sua utilização como terminal gráfico em sistemas multi-usuários, estações de trabalho, computadores pessoais e aplicações industriais.

Para aplicações que demandam alto desempenho, como realismo e animação, a estrutura de memória, devidamente expandida, adequa-se perfeitamente. Por outro lado, a disponibilidade de processamento apresenta-se muito exigua, exigindo para sua melhoria a imcorporação de co-processador de ponto flutuante, ou mesmo a incorporação de circuitos especificos para implementação das primitivas gráficas mais complexas (hidden lines, hidden surfaces, shading, etc), utilizando-se de técnicas de circuitos VLSI, ou ainda recursos de processamento paralelo, que são motivos de estudos em projetos atualmente em desenvolvimento no CTI e no DCA/FEE Unicamp.

Para finalizar, devemos ressaltar que o presente projeto, além dos protótipos implementados, contribuiu para a formação de recursos humanos e o estabelecimento de uma metodologia de desenvolvimento de sistemas, entendendo-se sistemas como o desenvolvimento simultâneo do hardware e do software, adequada às condições

dos Centros de Pesquisa nacionais, onde a disponibilidade de recursos financeiros, equipamentos, ferramentas e facilidades para importação de componentes, às vezes se situam muito aquém das condições ideais. Ainda, este trabalho representa uma contribuição ao desenvolvimento tecnológico nacional, somando seus resultados a outros projetos equivalentes que foram desenvolvidos ou estão em desenvolvimento em diversos Centros de Pesquisa e Universidades.

8.BIBLIOGRAFIA

[01] ASSARPOUR, H.; "Bit Mapped Graphics Using the UPD7220A". NEC. pp. 1-6.

[02] BERARDI, P.C; TOZZI, C.L.; NEVES JR; O.R.; "Definição de um Processador Gráfico para uma Estação de Trabalho". XVII Congresso Nacional de Informática. 1984.

[03] BERG, C.R.; "Computer Graphics Displays: Windows for Process Control. IEEE CG & A. Mai 1983. pp. 43-55.

[04] CARASSO, G.; GOETTSCH, R; & SYRIMIS, N; "Controller Chip Puts Text and Graphics on the Same BitMap". Eletronic Design. Jun 1985. pp. 119-126.

[05] CLARK, G.B; "Multiport Video Ram VS 32KX8 Dram Graphics Comparison. Texas Instruments. 1984.

[06] DETTMER, R.; "Chips for Graphics - Making the Pixels Fly".IEE Eletronic & Power. March 1987.

[07] FINKE, D.L.; "Dynamic Ram Architectures for Graphics Application". AFIPS CONF. PROC. 52. 1983. pp. 479-485.

[08] FOLEY, J.D.; VAN DAN, A; "Fundamentals of Interactive Computer Graphics". 2 ED. March 1983.

[09] GILOI, W.K.; "Interactive Computer Graphics". 1978.

[10] GULLEY, D.W.; "Joining Text and Graphics Enhances Vide Performance". Computer Design. Aug 1984. pp. 19-26.

[11] HARRINGTON, S.; "Computer Graphics. A Programming Approach". 2 ED . 1985.

[12] "HD63484 ACRTC - Advanced CRT Controller". Hitachi.

[13] "82720/GDC Applications Manual". Intel. July 1983.

[14] KANE, G.; "CRT Controller HandBook". 1980.

[15] KATSURA, K.; MAEJIMA, H.; & MINORIKAWA, K.; "Controller for Graphic Display". Hitachi Review Vol. 33 n´ 5. 1984.

[16] LEIBSON, S.H.; "Graphics Controller ICS". EDM. Feb. 1986. pp. 104-118.

[17] LEIVINTHAL, A.; & PORTER, T.; "Chap - A SIMD Graphics Processor". Computer Graphics Vol 18 n´ 3. July 1984.

[18] MYERS, R. E.; "Microcomputer Graphics for the IBM-PC".2 ED. 1979.

[19] NEWMAN, W.M., & SPROLL, R.F., "Principles of Interactive Computer Graphics". 11 ED 1979.

[20] NOVAK, M.; & PINKAM, R.; "Inside Graphics System, from Top to Botton". Eletronic Design, Vol 31 n° 15. 1983.

[21] OHR, S.; "How Silicon IC are Reshaping the Graphics Picture". Eletronic Design. June 1986.

[22] PERSIANO, R. C. M.; OLIVEIRA, A. A. F.; "Introdução a Computação Gráfica". V Esc. de Comp. - Belo Horizonte - 1986.

[23] PINKHAN, R.; "Video Ram Excels at Fast Graphics". 11 ED 1979. pp. 1-8.

[24] PINKHAN, R.; "Video Memory Technology and Applications". Texas Instruments. 1984. pp. 1-10.

[25] PINKHAN, R.; "A Hight Speed Dual Port Memory with Simultaneous Serial and Randon Mode Access for Video Applications". IEEE Journal of Solid State Circuits. Dez 1984.

[26] PISANO, J.; & YABLONSKI, R.; "CRT Terminal Architecture Provides Cost-Effective Customizing Versatility". Computer Design. Jan 1980. pp. 133-139.

[27] RIVOIRE III, C. & WILLY, J.; "One Chip VME Bus Interface Builds Compact Graphics Boards". Eletronic Design. Janeiro 1980. pp.97-100.

[28] RYHBERG, D.L.; "Using the MC68000 and the MC6845 for a Color Graphics System". Motorola Inc. 1981. pp. 1-27.

[29] "2670/71/72/73 CRT Set Application Briefs". Signetics. APP NOTE 403. Feb 1982.

[30] TELFORD, L.; "Opening the Window to your Process". C & I. Oct 1984.

[31] "TMS4161, 65536 - Bit Multiport Video Ram". Texas Instruments. 1983. pp. 1-22.

[32] "Graphics Display Technologies". Texas Instruments. 1984.

[33] "High Performance Memory Access with the TMS4161". Texas Instruments. 1984. pp. 1-4.

[34] "TMS34061 Evaluation Module Users Guide". Texas Instruments. 1985. pp. 1.1-5.10 Anexo A.1-A.3.

[35] "Creating a Graphics System for a BitMap Display". Texas Instruments. 1985.

[36] "Color Graphics Controller Board Users Guide". Texas Instruments. 1985.

[37] "TMS34070 Users Guide". Texas Instruments. 1985.

[38] "TMS34061 Users Guide". Texas Instruments. 1985.

[39] WHITTON, M.C.; ENGLAND, N.; "Manage Design Trade-Offs in High-End Graphics Board". Eletronic Design. March 1987. pp. 77-84.

[40] WILLIANSON, B.; RICKERT, P.; "Dedicated Processor Shrinks Graphics System to 3 Chips". Eletronic Design . March 1987.

[41] WINTJES, B.; GUTTAG, K.; ROSKELL, D.; "First Graphics Processor Takes Complex Orders to Run Bit-Mapped Displays". Eletronic Design. Jan. 1986. pp. 73-80.

[42] WITTON, M. C.; "Memory Design for raster Graphics". IEEE CG & A. Mar 1984. pp. 48-65.

[43] WOOTEN, D.; "Nibble-Mode Technique Simplifies Complex Ram Designs". EDN Feb. 1983. pp. 231-242.

[44] YAJNIK, K.; "SCC63484 Advanced CRT Controller - Application Note". Signetics. Mar 1987.

[45] JASSAL, T. P. S.; "Understanding Computer Graphics". Dataquest. Mar 1986. pp. 67-68.

[46] REGANATI, M. R. P. L.; AZEVEDO, H.; "NUGRA- Núcleo Gráfico". Documento técnico interno do Instituto de Automação, DTIA 003/88. Set 1988