1. Classificação *INPE-COM.1/TDL*
CDU.: 681.326.32

2. Período

4. Distribuição
interna [ ]
externa [X]

3. Palavras Chaves (selecionadas pelo autor)
MICROPROGRAMAÇÃO
EMULADOR DE MEMÓRIAS

5. Relatório nº
INPE-1489-TDL/009

6. Data
Maio, 1979

7. Revisado por
Eduardo W. Bergamini

8. Título e Sub-Título
EMULADOR DE MEMÓRIAS DE MICROCONTROLE AUXILIADO POR COMPUTADOR

9. Autorizado por
Nelson de Jesus Parada
Diretor

10. Setor *DEE/DRH* Código

11. Nº de cópias *17*

12. Autoria *Paulo Faria Santos Amaral*

14. Nº de páginas *161*

15. Preço

13. Assinatura Responsável *Paulo Faria Santos Amaral*

16. Sumário/Notas

A finalidade deste trabalho é apresentar o projeto e a simulação de um sistema muito útil para suporte em microprogramação. Ele pode operar sob o controle do sistema operacional do minicomputador HP2116B ou em um modo de operação independente. O sistema permite ao usuário carregar, testar, modificar e depurar um microprograma no seu ambiente de tempo real, durante a fase de projeto do dispositivo microprogramado, e fazer a diagnose e manutenção do protótipo final. Uma vez que um microprograma está correto, o sistema gera uma fita de papel perfurada de tal modo que ele pode ser diretamente carregado em uma memória do tipo PROM por um programador de PROM.

17. Observações *Tese de Mestrado em Eletrônica e Telecomunicações/Sistemas Digitais e Analógicos, aprovada em 09 de abril de 1979.*

Aprovada pela Banca Examinadora
em cumprimento dos requisitos exigidos
para a obtenção do Título de Mestre em
Eletrônica e Telecomunicações/Sistemas
Digitais e Analógicos

Dr.Claudio Roland Sonnenburg — Presidente

Dr.Eduardo Whitaker Bergamini — Orientador

Dr.Antonio Marcos de A.Massola — Membro da Banca -convidado-

Eng.Arry Carlos Buss Filho, MSc. — Membro da Banca

Paulo Faria Santos Amaral — Candidato

São José dos Campos, 09 de abril de 1979

INDICE

ABSTRACT

The purpose of this work is to present the system design and simulation of a very useful microprogramming support system. It can operate under the operating system of the minicomputer HP2116B or in a stand alone fashion. The system allows the user to load, test, modify and debug a microprogram in its real time environment, during the design phase of the microprogrammed device, and to make the diagnosis and maintenance of the final prototype. Once a microprogram is certified, the system generates a paper tape so that it can be directly loaded in PROM's by a PROM programmer.

LISTA DE FIGURAS

LISTA DE TABELAS

CAPÍTULO I

MICROPROGRAMAÇÃO E SUAS CARACTERÍSTICAS

1.1 - INTRODUÇÃO

Para a utilização de um equipamento eletrônico, assim como outros equipamentos e comodidades do mundo moderno, como o automóvel, por exemplo, não é necessário um conhecimento profundo do equipamento em si. Este fato levou ao conceito da "caixa preta" onde a preocupação principal reside nas entradas, saídas e funções executadas pelo sistema, e não na maneira em que estas funções são executadas.

Para o usuário, o computador é uma caixa preta controlada por um conjunto de comandos conhecido como conjunto de instruções de máquina ou conjunto de instruções do usuário. Estas instruções, colocadas na memória principal do computador, dirigem o processador na execução de funções básicas, tais como mover um dado da memória para um registro de uso geral, executar operações aritméticas, controlar entrada e saída, etc. É através deste conjunto de instruções que se comunica ao computador o que deve ser feito para que uma determinada tarefa seja executada.

É possível implementar em circuitos ("hardware) este conjunto de instruções, ou seja, projetar circuitos que buscam, na memória principal do computador, decodificam e executam cada instrução do seu conjunto de instruções. Esta implementação, chamada de controle convencional ou de lógica aleatória (em inglês "hardwired"),era utilizada nos computadores antigos e ainda o é em algumas máquinas atuais.À medida que a tecnologia do computador evoluiu, e ainda evolui,conjuntos de instruções cada vez maiores e mais poderosos são exigidos pelo usuário, o que torna a sua implementação, utilizando o controle convencional, uma tarefa extremamente complexa.

Outro aspecto, além da complexidade do conjunto de instruções, é a versatilidade da máquina. É evidente que problemas diferentes podem ser mais facilmente resolvidos, empregando-se conjuntos de instruções diferentes, orientados para o problema específico. Por exemplo, uma aplicação numérica requer um conjunto de instruções com operações aritméticas sofisticadas, enquanto um sistema de processamento de dados requer um conjunto de instruções orientado para funções de manipulação de dados e edição. Para que os dois fossem satisfatórios e economicamente resolvidos por uma mesma máquina, utilizando o controle convencional, significaria que, se possível, uma parte do circuito da máquina deveria ser substituída, dependendo da aplicação do computador.

Uma solução mais elegante foi encontrada pelo Prof.M.V. Wilkes do Laboratório de Matemática da Universidade de Cambridge no começo da década de 1950. Ele reconheceu que os circuitos de controle de um computador operam em uma série de passos discretos, bastante semelhante à execução de um programa e, então, propôs o conceito de microprogramação, pela qual a unidade de controle de um computador poderia ser programada para usar os outros componentes do sistema de uma maneira predefinida.

Em um processador microprogramado, as operações básicas da máquina são controladas por microinstruções. Estas microinstruções controlam a conexão de registros e barramentos, testes e alterações de biestáveis, operações aritméticas, etc. A sequência de microinstruções ou microprograma, é guardada em um lugar chamado memória de controle, memória de microcontrole ou memória de microprograma. Desta forma, o conjunto de instrução da máquina é definido pelo conteúdo da memória de microcontrole, ao invés de por um circuito fixo, como no caso do controle convencional. Para se alterar o conjunto de instruções do processador bastaria então, alterar o conteúdo da memória de microcontrole.

Desta maneira, a microprogramação resolve com eficiência e economia, tanto o problema da complexidade de implementação do conjunto de instruções, como também o problema da versatilidade necessária dos mesmos.

Este capítulo tem por finalidade dar uma idéia das implementações convencional e microprogramada, dos conceitos existentes em microprogramação, de uma comparação entre as duas implementações e da necessidade de desenvolvimento de uma ferramenta de apoio à microprogramação.

1.2 - CONTROLE CONVENCIONAL E CONTROLE MICROPROGRAMADO

Um computador digital pode ser funcionalmente dividido nas quatro unidades básicas mostradas na Figura I.1 (Agrawala, 1976).

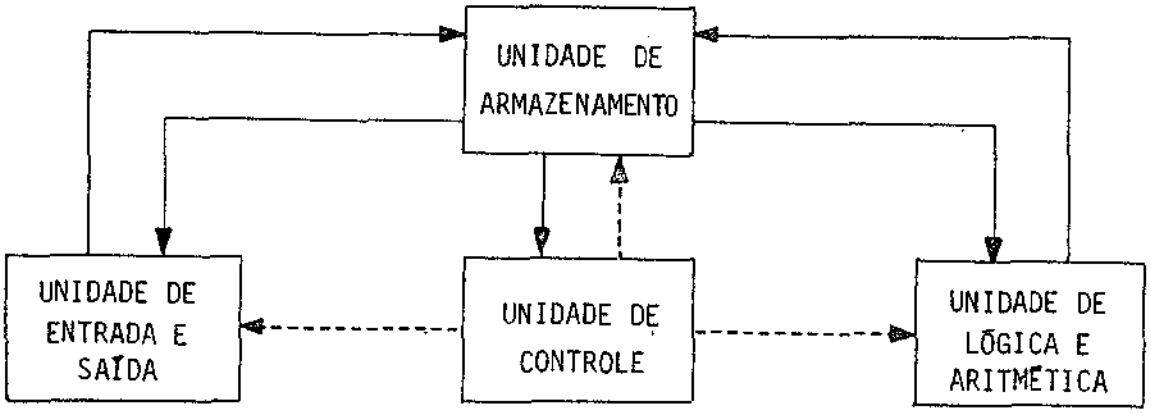

Fig. I.1 - Organização funcional de um computador digital.

A unidade de entrada e saída é responsável pela comunicação entre o computador e o meio externo, convertendo informação conveniente para a aplicação externa em informação aceitável pelo computador e vice-versa.

A unidade de armazenamento é responsável pelo armazenamento de informação, seja de dados ou de programas. Ela engloba a memória principal do computador, registros de uso geral, etc.

A unidade de lógica e aritmética transforma os dados, executando operações lógicas (E, OU, NE, etc.), operações aritméticas (soma, subtração, multiplicação, etc.) e operações de deslocamento.

A unidade de controle tem por finalidade fornecer os sinais de controle atuais para as outras unidades do computador e determinar, a partir de informação nela contida e também recebida das outras unidades, quais os próximos sinais de controle a serem gerados.

É o tipo de implementação da unidade de controle que caracteriza o computador quanto a sua eficiência, rapidez e versatilidade. Como foi dito, esta implementação pode ser do tipo convencional ("Hardwired") ou do tipo microprogramada.

1.2.1 - UNIDADE DE CONTROLE CONVENCIONAL

Uma unidade de controle convencional é implementada, utilizando-se combinações de portas lógicas, contadores, biestáveis, decodificadores, etc., todos interconectados de uma maneira quase que aleatória. Uma unidade de controle deste tipo é inflexível e tem dificuldades de projeto, que resultam em configurações não estruturadas bastante complexas.

A Figura I.2 mostra um diagrama de blocos bem simplificado, de uma unidade de controle deste tipo.

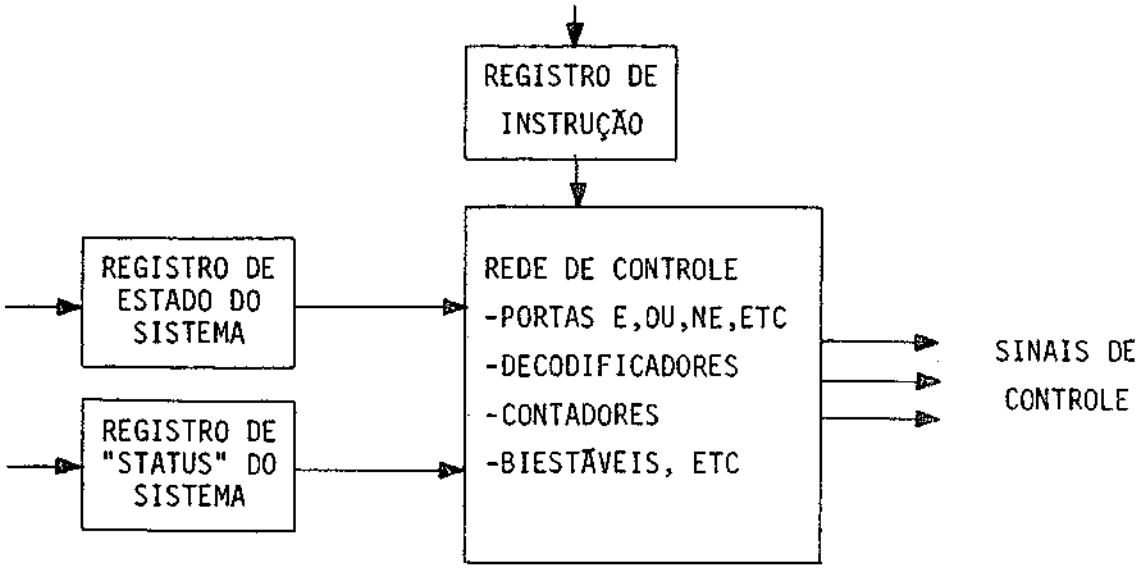

Fig. I.2 - Unidade de controle convencional ("Hardwired")

A tarefa principal da unidade de controle é sequenciar as microoperações de máquina e emitir sinais para controlar pontos do sistema. Esta tarefa pode ser subdividida em duas fases: 1) estabelecer o estado de controle durante cada ciclo de máquina; 2) determinar corretamente o próximo estado na sequência de controle.

Nesta implementação, a partir da informação contida no registro de instrução, estado e "status" do sistema, a rede de controle gera os sinais de controle, dos quais uma parte é usada para a determinação do próximo estado de controle e a outra para controlar as outras unidades do sistema.

Este tipo de implementação foi bastante utilizada na fase em que os conjuntos de instrução eram primitivos, ou seja, um pequeno número de estados de controle eram suficientes para a implementação de todo o conjunto de instruções. Outra razão da sua utilização anterior é o fator custo da lógica. Esta implementação tende a minimizar o número de circuitos, o que era bastante conveniente na época em que o cus-

to da lógica era elevado.

À medida em que os computadores foram ficando complexos, com conjunto de instruções altamente sofisticados e o custo da lógica foi decrescendo, a implementação microprogramada foi ganhando popularidade nos projetos de sistema digitais de computação, sendo praticamente utilizada em todos eles atualmente.

1.2.2 - UNIDADE DE CONTROLE MICROPROGRAMADA

O projeto de uma unidade de controle microprogramada é caracterizado por uma estrutura bastante regular onde os sinais de controle, ao invés de serem gerados por uma rede de lógica aleatória, como no caso da unidade convencional, estão armazenados nas palavras da memória de controle.

A Figura I.3 mostra um diagrama de blocos bem simplificado de uma unidade de controle microprogramada.

Na implementação microprogramada, o estado corrente de controle da máquina é definido pelo endereço corrente da memória de controle, enquanto o conteúdo desta posição, que é a microinstrução, fornece a informação necessária para estabelecer os sinais de controle e escolher o próximo endereço de microinstrução. O sequenciador de microprograma gera o endereço da próxima microinstrução a ser executada, baseado em: 1) informação do próximo endereço que ele recebe da microinstrução corrente; 2) entradas vindas da instrução de máquina pela memória de mapeamento e 3) sinais de "status" e entradas condicionais que ele recebe do resto do sistema.

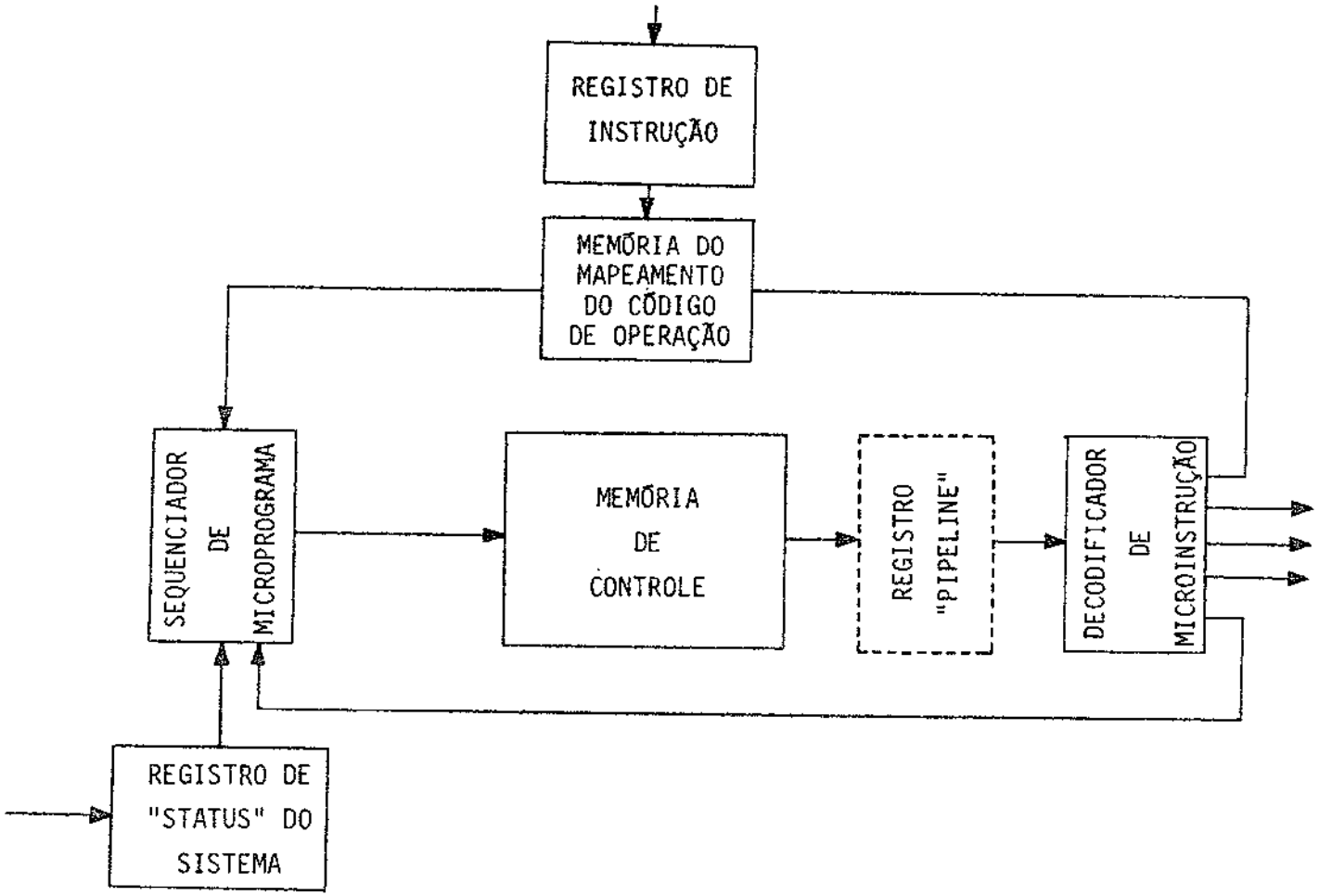

Fig. I.3 - Diagrama de blocos de uma unidade de controle microprogramada.

Vários conjuntos de microinstrução, chamados de microprogramas, são armazenados na memória de controle do sistema. O código de operação de instrução de máquina (macroinstrução) é mapeado pela memória de mapeamento do código de operação, apontando para a microrotina apropriada na memória de controle. A execução da microrotina, pelo circuito, resulta no controle de todas as operações fundamentais, tais como a transferência de dados de um registro para outro ou transformações elementares nos dados. Com isto, técnicas de programação são agora de utilidade para o projetista de sistemas digitais. Mudando-se estas microrotinas ou trocando-se as memórias de controle, altera-se a complexidade funcional da máquina, fazendo-se com que ela execute um novo conjunto de instruções ou emule outra máquina.

A aleatoriedade dos sinais de controle, que é dependente das unidades a serem controladas, ainda existe na unidade de controle microprogramada, só que agora na forma de dados armazenados na memória de controle.

A técnica de microprogramação resulta numa lógica de circuitos muito mais fácil de ser projetada e muito mais estruturada e organizada. O baixo custo e vantagens oferecidas pelos dispositivos LSI permitem o projeto de unidades de controle microprogramadas cada vez mais complexas. A principal desvantagem da implementação microprogramada é que a sua velocidade é normalmente menor do que a da unidade convencional.

Pode-se aumentar consideravelmente a velocidade da unidade de controle microprogramada, colocando-se um registro, chamado de registro "pipeline", na saída da memória de controle. Com este procedimento a busca da próxima microinstrução a ser executada é feita em paralelo com a execução da microinstrução corrente, o que aumenta consideravelmente a frequência de execução das microinstruções. Cuidado deve ser tomado, ao se utilizar esta técnica, com as microinstruções de desvio condicionais, cujas condições dependem da execução da microinstrução corrente, o que torna difícil, durante esta execução, prever qual será a próxima microinstrução a ser executada.

Outra maneira de minimizar a desvantagem da velocidade em implementações microprogramadas é que, sendo, nesta implementação, fácil o controle paralelo de unidades funcionais, pode-se projetar um circuito onde o paralelismo de operações seja bastante desenvolvido, fazendo-se com que várias operações, antes executadas sequencialmente, sejam agora executadas em paralelo sob o controle da unidade microprogramada.

1.3 - CARACTERÍSTICAS DA MICROPROGRAMAÇÃO

A personalidade de um projeto digital é determinada pela sua unidade de controle.

Numa unidade de controle microprogramada podemos separar os aspectos que a caracterizam em três: 1) projeto do sequenciador de microprograma, que gera o próximo endereço de microinstrução para sequenciar os microprogramas; 2) projeto da memória de controle, onde são guardadas as microinstruções; 3) projeto das microinstruções propriamente dito.

1.3.1 - SEQUENCIADOR DE MICROPROGRAMA

O sequenciador de microprogramas de uma unidade de controle microprogramada, como já foi dito, é responsável pela geração do endereço da próxima microinstrução a ser executada no microprograma. Este sequenciador pode ser desde um simples registro contador até dispositivos LSI extremamente sofisticados, dependendo do tipo da unidade de controle a ser implementado.

É a estrutura do sequenciador de microprograma que determina as características gerais da unidade de controle e os mecanismos disponíveis para se implementar eficiente e modularmente os microprogramas.

Para garantir uma microprogramação eficiente e modular, o sequenciador de microprogramas, junto com o resto da unidade de contro

le, deve permitir a implementação em microprogramas das três estruturas básicas de controle, mostradas na Figura I.4.

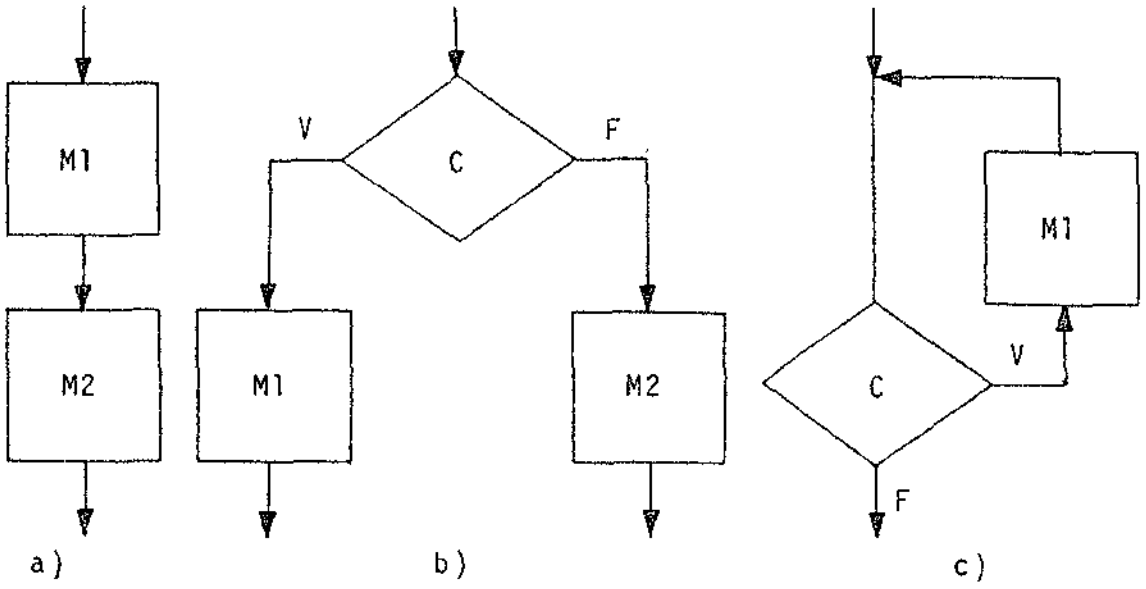

Fig. I.4 - Estruturas básicas de controle implementadas em microprogramas.

Nesta Figura I.4, M1 e M2 representam módulos de microprogramas e C uma determinada condição possível de ser testada. A parte (a) representa a estrutura; M1 então M2, ou seja, se o módulo de microprograma M1 for executado, então o módulo de microprograma M2, será executado. A parte (b) representa a estrutura: se C for verdade, então M1, e se falsa, então M2, ou seja, somente um módulo de microprograma, M1 ou M2, será executado, dependendo da condição C. A estrutura da parte (c) representa: enquanto C for verdadeira execute M1, ou seja, o módulo de microprograma M1 será executado enquanto a condição C for verdadeira.

Um diagrama de blocos típico de um sequenciador de microprogramas LSI comercial é mostrado na Figura I.5 (Advanced Microdevices, 1976a)

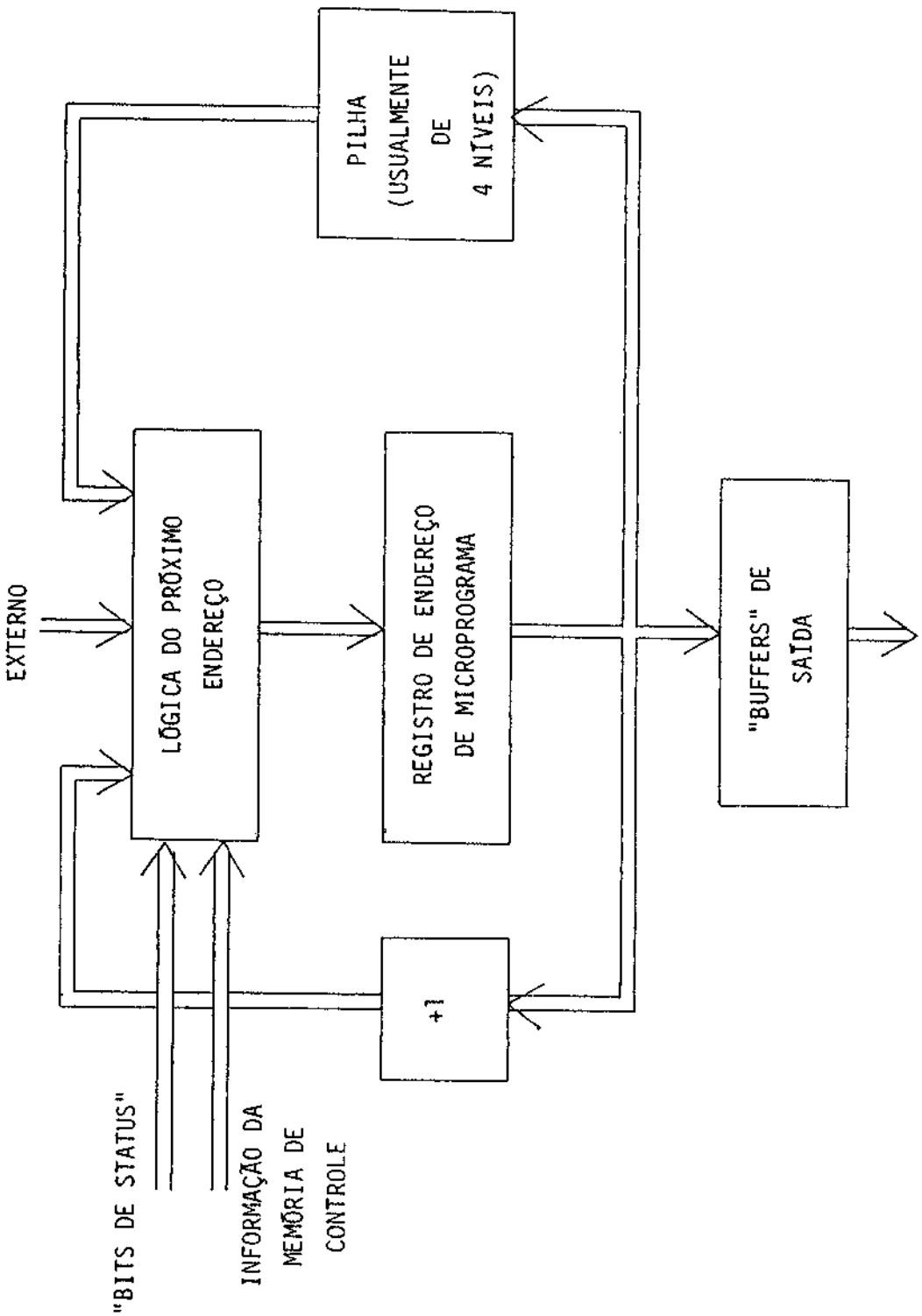

Fig. I.5 - Sequenciador de microprograma LSI típico.

A parte lógica do próximo endereço determina a fonte de endereço a ser carregada no registro de endereço de microprograma.A escolha de endereço é determinada pelos "bits" do sistema, pela informação do próximo endereço vinda da microinstrução corrente e outras, podendo ser uma execução sequencial (incrementa o conteúdo de registro do endereço de microprograma), desvios condicionais e incondicionais, e operações que mexem com a pilha "Push" e "Pop").

Estas operações de pilha são executadas durante "loops" em microprogramas e desvios para sub-rotinas, quando os endereços de retorno são temporariamente armazenados na pilha com comandos "push" e retirados com comandos "pop".

O "buffer" de saída é colocado para servir de interface entre o sequenciador de microprograma e a memória de controle.

1.3.2 - MEMÓRIA DE CONTROLE

A memória de controle contém as microinstruções que especificam os passos, através dos quais a unidade de controle sequencia e controla as operações paralelas das unidades funcionais do sistema digital.

Os fatores importantes no projeto de uma memória de controle de um sistema microprogramado são: estrutura, tamanho, velocidade e modos de carregamento.

A Figura I.6 ilustra os vários tipos de estruturas de memória de controle, classificadas segundo Agrawala, 1976.

O primeiro tipo é a estrutura mais simples e comum de memória de controle, onde existe uma microinstrução em cada palavra de controle.

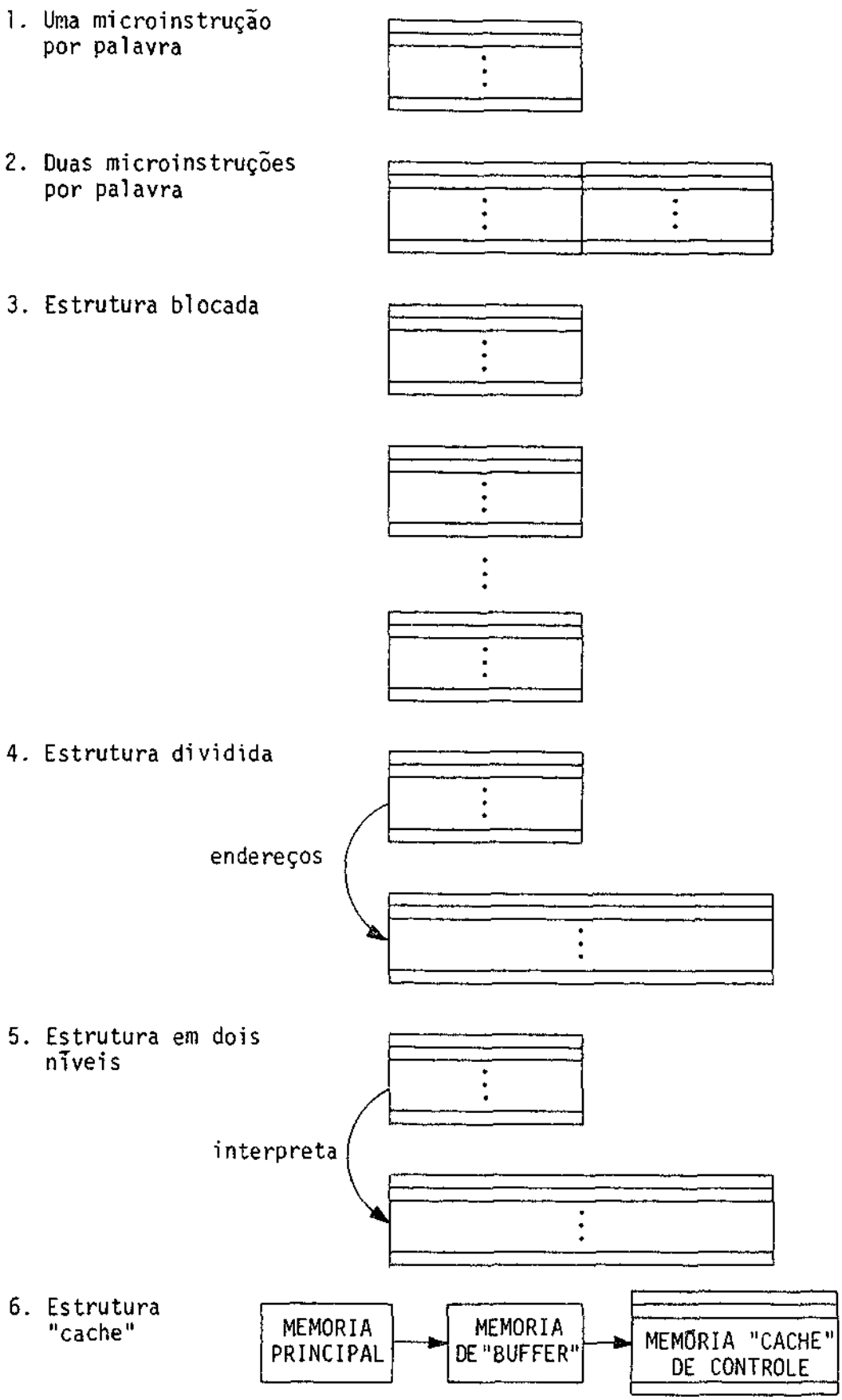

Fig. I.6 - Tipos de estrutura de memória de controle.

No segundo tipo, duas microinstruções ocupam cada palavra de controle, fazendo com que as duas sejam lidas ao mesmo tempo, reduzindo o número de acessos à memória de controle, e aumentando a velocidade da unidade de controle.

No terceiro tipo, a memória de controle é dividida em blocos, também chamados de páginas. Neste esquema existem dois tipos de endereços de memória de controle: 1) endereços de microinstruções contidas no mesmo bloco de memória; 2) endereços de microinstruções em blocos diferentes de memória. A vantagem nesta estrutura é que um menor número de "bits" é utilizado para os endereços de microinstruções no mesmo bloco.

No quarto tipo, a memória de controle é dividida em duas unidades de armazenamento com tamanho de palavras diferentes. A primeira microinstrução fornece condições iniciais e determina a execução de microinstruções que residem no segundo bloco. Esta segunda microinstrução tem um comprimento maior, podendo exercer um maior controle sobre os recursos da máquina. Esta implementação requer uma quantidade de "bits" menor do que a estrutura (1) quando várias microinstruções, menores referem-se a mesma microinstrução maior.

Na estrutura em dois níveis (estrutura 5), as microinstruções no segundo nível interpretam as microinstruções do primeiro nível, da mesma forma que as microinstruções interpretam as instruções de máquina. Como neste caso, a vantagem obtida é a flexibilidade.

No sexto tipo, as microinstruções são executadas a partir de uma pequena memória de alta velocidade carregada, a partir de uma memória "buffer" mais lenta ou da própria memória principal do sistema. A vantagem desta estrutura é que o microprogramador não precisa se preocupar com o número de microinstruções que o seu microprograma irá ocupar.

O tamanho da memória de controle é determinado pelo número de palavras que ela possui e pelo número de "bits" que cada palavra tem. O número de "bits" em uma microinstrução, geralmente, depende do número de operações primitivas existentes no sistema, do paralelismo necessário na execução destas operações primitivas e da representação utilizada no controle destas operações. O tamanho das microinstruções nos computadores comerciais varia de 16 a 100 "bits", tendo uma memória de controle com o número de palavras variando de 256 a 4096.

O tempo necessário para se ler um dado da memória de controle em um registro, normalmente o registro "pipeline", pode ser um fator limitante na velocidade final do sistema. A velocidade da memória de controle é um fator que determina o seu custo, devendo ser determinada em um estudo comparativo com a velocidade dos outros recursos do sistema. Se a execução das microinstruções deve ficar continuamente atrazada, esperando a execução de operações em outras unidades do sistema, uma memória de controle extremamente rápida pode não ser justificável.

O modo de carregamento das microinstruções na memória de controle determina a microprogramabilidade do sistema digital. Se a memória de controle for tal que o usuário não tenha nenhum acesso às informações, tais como formato, campos das microinstruções, etc., o sistema digital é dito microprogramado. Se estas informações forem tais que permitam ao usuário microprogramar o sistema, mas numa velocidade baixa, como no caso de memórias de controle do tipo PROM, o sistema é dito microprogramável. Se estas informações forem disponíveis e a velocidade de carregamento da memória de controle for eletrônica, como no caso de memórias de controle utilizando memórias do tipo RAM de alta velocidade, o sistema é dito dinâmicamente microprogramável.

1.3.3 - PROJETO DAS MICROINSTRUÇÕES

O projeto das microinstruções deve levar em conta a informação que é necessária para controlar os componentes do circuito e determinar como esta informação é arranjada em uma palavra de microinstrução.

Normalmente uma microinstrução é composta de duas partes: 1) dos "bits" que correspondem aos sinais de controle que definem e controlam todas as operações elementares a serem executadas; 2) dos "bits" que identificam e controlam o endereço da próxima microinstrução a ser executada.

Geralmente existe uma grande variedade de esquemas para o formato, comprimento e execução das microinstruções. O formato, usualmente, refere-se a definição dos vários campos que compõe a microinstrução. O formato das microinstruções pode ser fixo ou variável. No formato fixo, cada "bit" da microinstrução é sempre interpretado da mesma maneira, enquanto que no formato variável, o significado dos "bits" ou campos na microinstrução mudam de acordo com certas condições do sistema. Formatos variáveis resultam em microinstruções mais curtas, mas requerem lógica adicional e atrasos para o controle do formato.

No projeto de microinstruções, uma característica essencial é o número de recursos de circuito que cada microinstrução controla, ou seja, o número de microoperações que cada microinstrução pode executar. Quanto a esta característica, as microinstruções são classificadas em verticais e horizontais, se bem que esta classificação corresponde aos extremos das possibilidades.

Microinstruções verticais geralmente ocasionam a execução de apenas uma microoperação em cada ciclo de relógio. Elas se assemelham às instruções de máquinas clássicas, consistindo de uma operação e poucos operandos.

Microinstruções horizontais ocasionam a execução de várias microoperações que, em geral, são executadas em paralelo. Esta possibilidade leva a microinstruções mais compridas, com muitos campos de controle. Com esta capacidade de executar microoperações em paralelo, o número de microinstruções em um microprograma diminue, diminuindo o tempo e simplificando a execução das microinstruções, aumentando a velocidade da máquina. Nas microinstruções verticais, o tamanho da palavra de controle é menor, mas a execução torna-se mais complicada e são necessárias várias microinstruções para a execução dos microprogramas, tornando a máquina mais lenta.

Outra característica importante no projeto de microinstruções é o grau de codificação das mesmas. Se cada "bit" da microinstrução é usado exclusivamente para controlar uma linha de controle do sistema, então a microinstrução é chamada de não-codificada ou de forma explodida. Este formato leva a microinstruções e velocidades de execução maiores.

Um esquema alternativo identifica os grupos de linhas de controle mutuamente exclusivas e as controlam por um único campo de controle codificado na microinstrução. A Figura I.7 mostra um esquema dos dois casos.

O esquema de codificação dos "bits" de controle leva a microinstruções menores, sem muita perda de flexibilidade, mas com uma certa perda em paralelismo e velocidade de execução, além de um aumento de lógica necessária para a decodificação.

A forma de buscar e executar uma microinstrução é outra característica no projeto de microinstruções. Esta característica é determinada pelo paralelismo nas fases de execução da microinstrução corrente e pela busca da próxima microinstrução a ser executada.

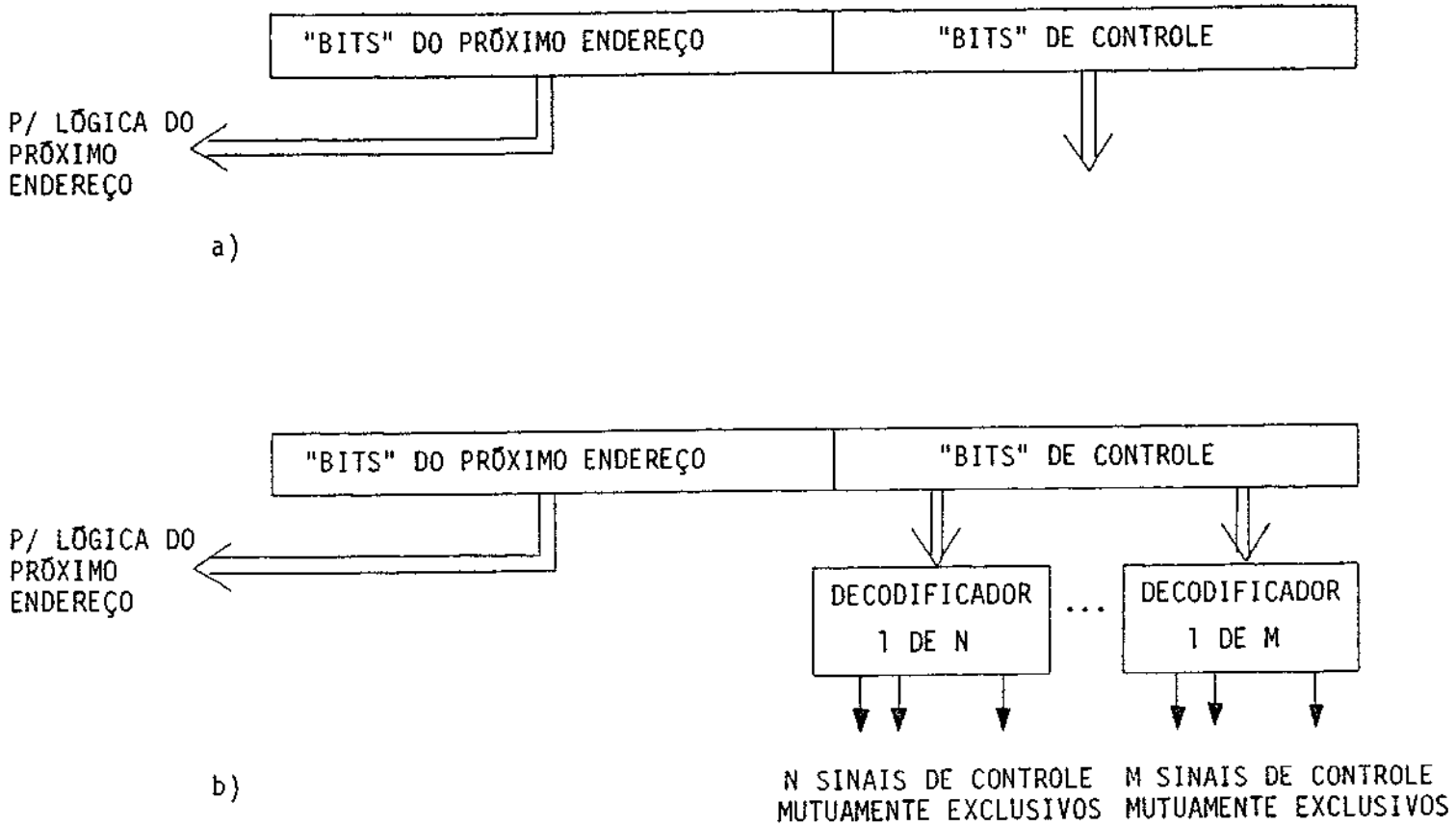

Fig. I.7 - a) Formato de microinstrução não codificada.
b) Formato de microinstrução codificada.

Na implementação em série, como mostra a Figura I.8 a), a busca da próxima microinstrução não é iniciada enquanto a microinstrução corrente não for executada. A vantagem desta implementação é a simplicidade da realização em detrimento da velocidade de processamento.

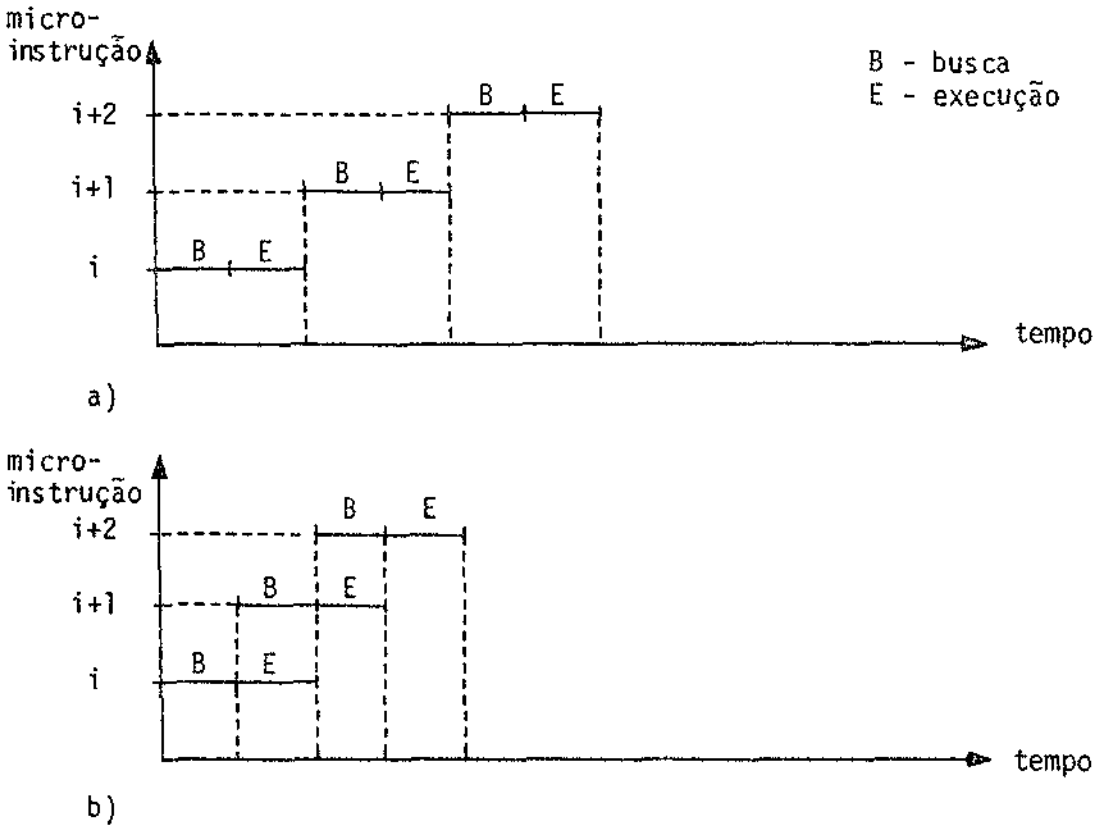

Fig. I.8 - a) Implementação em série; b) Implementação em paralelo.

Na implementação paralela a busca da próxima microinstrução é executada em paralelo com a execução da microinstrução corrente. A vantagem desta implementação é a economia de tempo da busca da microinstrução, tendo como desvantagem a complicação acarretada no circuito, principalmente em microinstruções de desvio condicional, cuja condição depende da execução da microinstrução corrente. Quando isto acontece, a busca da próxima microinstrução tem que ser interrompida até que a execução da microinstrução corrente termine.

Outra característica, associada ao projeto de microinstruções, refere-se ao número de fases do relógio necessárias para executar uma microinstrução. Quanto a esta característica, as microinstruções são divididas em monofásicas e polifásicas.

A microinstrução monofásica é a mais simples das duas, porque cada microinstrução requer um pulso do relógio para a sua execução. Isto faz com que os microprogramas sejam maiores, mas com microinstruções de comprimento menor.

Por outro lado as microinstruções polifásicas são mais complexas porque os sinais de controle de uma mesma microinstrução são efetivados em sequência, durante vários pulsos do relógio. Estas microinstruções são chamadas de polifásicas síncronas, se o diagrama de tempo de execução de microinstrução for o mesmo para todas as microinstruções, e assíncronas, se o número de pulsos do relógio para a execução de microinstrução dependem da complexidade das microoperações que ela executa. A implementação polifásica permite a sobreposição de funções a serem executadas, aumentando a velocidade efetiva de operação do sistema.

Durante o projeto das microinstruções, o projetista deve balancear o tamanho da memória de controle, velocidade e esquemas complicados de diagramas de tempo de execução de microinstruções, flexibilidade de desvios, e quantidade e complexidade de circuitos adicionais para tratar e decodificar os campos de controle das microinstruções.

Em geral, o projeto das microinstruções é fortemente influenciado pelo projeto do fluxo de dados da seção de processamento e pelo conjunto de instruções de linguagem de máquina a ser executada pelo processador.

1.4 - VANTAGENS DA IMPLEMENTAÇÃO MICROPROGRAMADA SOBRE A CONVENCIONAL

A técnica de microprogramação está sendo largamente aplicada em terminais, calculadoras, controladores de periféricos, unidades centrais de processamento, comunicadores "modems", unidades de processamento aritmético, etc., em substituição à lógica aleatória antes utilizada.

Existem numerosas vantagens da implementação microprogramada sobre a implementação utilizando o controle por lógica aleatória (Signetics Memory Systems, 1970). Algumas dessas vantagens são citadas a seguir:

1) Redução da lógica aleatória para controle do sistema, resultando em uma arquitetura melhor estruturada.

A parte de controle de um sistema digital não microprogramado é geralmente bastante rígida. Esta parte é composta de contadores, biestáveis, redes de decodificação, etc. Nos sistemas microprogramados, uma estrutura regular de memória substitui a maior parte destes dispositivos de controle. Um certo grau de aleatoriedade ainda existe, mas agora restrita à sequência de dados armazenados na memória de controle. A arquitetura torna-se realmente mais estruturada, porque todo o controle é feito da mesma maneira, mas sem perda de flexibilidade.

2) A emulação de vários dispositivos (não somente processadores) pode ser feita por um único dispositivo de propósito geral, alterando-se o conteúdo de sua memória de controle.

Quando se deseja projetar um certo número de dispositivos de fins específicos, por exemplo, controladores de periféricos, às vezes é melhor e mais econômico projetar um dispositivo mais geral e, através da sua microprogramação, adaptá-lo para emular a função de cada um dos dispositivos para o fim específico desejado.

3) Rotinas de diagnose e manutenção podem ser fácil e eficientemente implementadas na unidade de controle do sistema.

Um dos principais problemas da maioria dos sistemas de diagnose é que uma grande parte do sistema deve estar funcionando corretamente para que a diagnose possa ser executada. Ainda mais, as instruções de máquina tendem a estimular porções grandes do sistema, tornando muito difícil apontar as falhas com precisão.

Implementações microprogramadas são verdadeiramente convenientes para a manutenção e diagnose. Basta que uma pequena parte do sistema esteja funcionando (a unidade de controle), para que seja possível rodar os programas de diagnose e manutenção. Além disto, através da memória de controle pode-se ter acesso a testes que não seriam possíveis utilizando-se, somente, instruções de máquina.

4) Mudanças e adaptações específicas que se fizerem necessárias no sistema podem, em princípio, ser facilmente realizadas alterando-se o conteúdo da memória de controle.

Os dispositivos microprogramados, devido ao fato de possuirem um programa armazenado na memória de controle, são, em geral, bem mais maleáveis do que os dispositivos não microprogramados. Características necessárias para adaptar o dispositivo para uma determinada aplicação podem ser facilmente alteradas ou acrescentadas, em geral, alterando-se apenas o conteúdo da memória de controle.

5) O desempenho de dispositivos microprogramados pode ser aumentado com custos menores do que os custos para sistemas convencionais.

Para sistemas muito simples, o controle com lógica aleatória é um modo econômico de se implementar o projeto. À medida que a complexidade do sistema aumenta, o custo da lógica de controle para estes sistemas pode aumentar numa razão bastante rápida. Em projetos mi

croprogramados, o custo de um sistema simples é, em geral, um pouco maior. Em compensação, acrescentar características ao microprograma, de modo a aumentar seu desempenho, requer apenas uma alteração, ou mesmo um aumento da sua memória de controle, o que pode tornar os custos, do projeto, menores.

6) A definição final do sistema pode ser adiada com maior facilidade, durante o ciclo de projeto do mesmo.

À medida que o sistema é projetado ou aplicado, sempre surgem novas idéias de como melhorá-lo. Talvez pior ainda, é quando no decorrer do mesmo descobre-se que partes importantes foram esquecidas. Em projetos microprogramados estes problemas são facilmente contornáveis pois, sendo sua estrutura flexível, melhoramentos e mudanças podem ser feitos, muitas vezes, com a simples alteração do conteúdo de sua memória de controle. O tempo necessário para estas alterações é bem pequeno e, além de tudo, os efeitos destas alterações na documentação de "hardware" do sistema é mínimo, quando somente o conteúdo da memória de controle precisa ser alterado.

7) Documentação e custos de treinamento de manutenção podem ser reduzidos.

Certamente a parte mais difícil de um sistema para ser documentada e ensinada a técnicos de manutenção é a parte que contém a lógica aleatória de controle em sistemas convencionais. Reduzindo-se a quantidade de lógica aleatória, esta dificuldade também é reduzida. Ainda mais, os técnicos de manutenção em sistemas microprogramados trabalham com uma mesma estrutura básica, que pode ser no entanto, programada de formas diferentes. Em sistemas convencionais, eles devem estar familiarizados com o sequenciamento e, também com a estrutura de controle dos vários modos do sistema.

Como já foi dito anteriormente, a principal desvantagem da implementação microprogramada em relação à implementação convencio

nal é que, em geral, a microprogramada é um pouco mais lenta do que a convencional. Esta desvantagem pode ser compensada com a utilização de técnicas como as descritas no item 1.3.

1.5 - NECESSIDADE DE UMA FERRAMENTA DE APOIO À MICROPROGRAMAÇÃO

Do citado no item 1.4, podemos perceber que as vantagens dos sistemas com implementação microprogramada sobre os convencionais são extremamente dependentes da possível capacidade de gerar,carregar, alterar, acrescentar e testar, com facilidade, microprogramas a serem carregados ou já carregados nas memórias de controle, tanto na fase de projeto como na fase de manutenção do sistema.

Quando o tamanho do microprograma é pequeno, esta tarefa pode ser feita manualmente, com algum sacrifício, mas quando a memória de controle cresce, tanto em número de palavras como no tamanho da palavra, esta tarefa se torna bastante difícil e demorada,requerendo uma ferramenta suporte.

As ferramentas mais utilizadas para testar microprogramas são:

1) Programa simulador executado em outro computador;

2) Emulador de circuitos.

As vantagens de um simulador sobre os emuladores é que facilidades para testes em microprogramas, tais como execução passo-a-passo, pontos pré-determinados de parada, sequenciamento das microinstruções e verificação dos conteúdos de registros podem ser facilmente implementadas no programa.

As desvantagens do programa simulador para testes de microprogramas são: 1) à medida que circuitos LSI mais complexos vão sendo utilizados na implementação do sistema, a sua simulação vai ficando cada vez mais trabalhosa e difícil; 2) existe a necessidade de progra

mas especiais para cada tipo de circuito a ser simulado; 3) a simulação valida apenas a lógica dos microprogramas, ou seja, não existe a simulação em tempo real.

Já o emulador de circuitos tem como principal vantagem o fato dos testes dos microprogramas serem feitos em tempo real ou numa velocidade bem próxima da real, com o circuito, em funcionamento sujeito a todas as suas condições ambientais, tais como atrasos nas portas, carga na entrada, carga na saída, etc.

Se as facilidades existentes em um programa simulador fossem introduzidas em um emulador de circuitos e este fosse independente do circuito a ser testado, ou seja, o circuito onde os microprogramas seriam executados, esta seria a ferramenta ideal para a validação de microprogramas.

Este trabalho visa o desenvolvimento de uma ferramenta deste tipo denominada de EMMAC; que significa Emulador de Memórias de Microcontrole Auxiliado por Computador.

Para que esta ferramenta seja independente do circuito onde os microprogramas serão executados e para que estes sejam executados, sujeitos a todas as condições ambientais, pode-se fazer apenas a emulação das memórias de controle do sistema, controle do relógio e monitoramento de pontos de teste escolhidos no sistema, como mostra a Figura I.9.

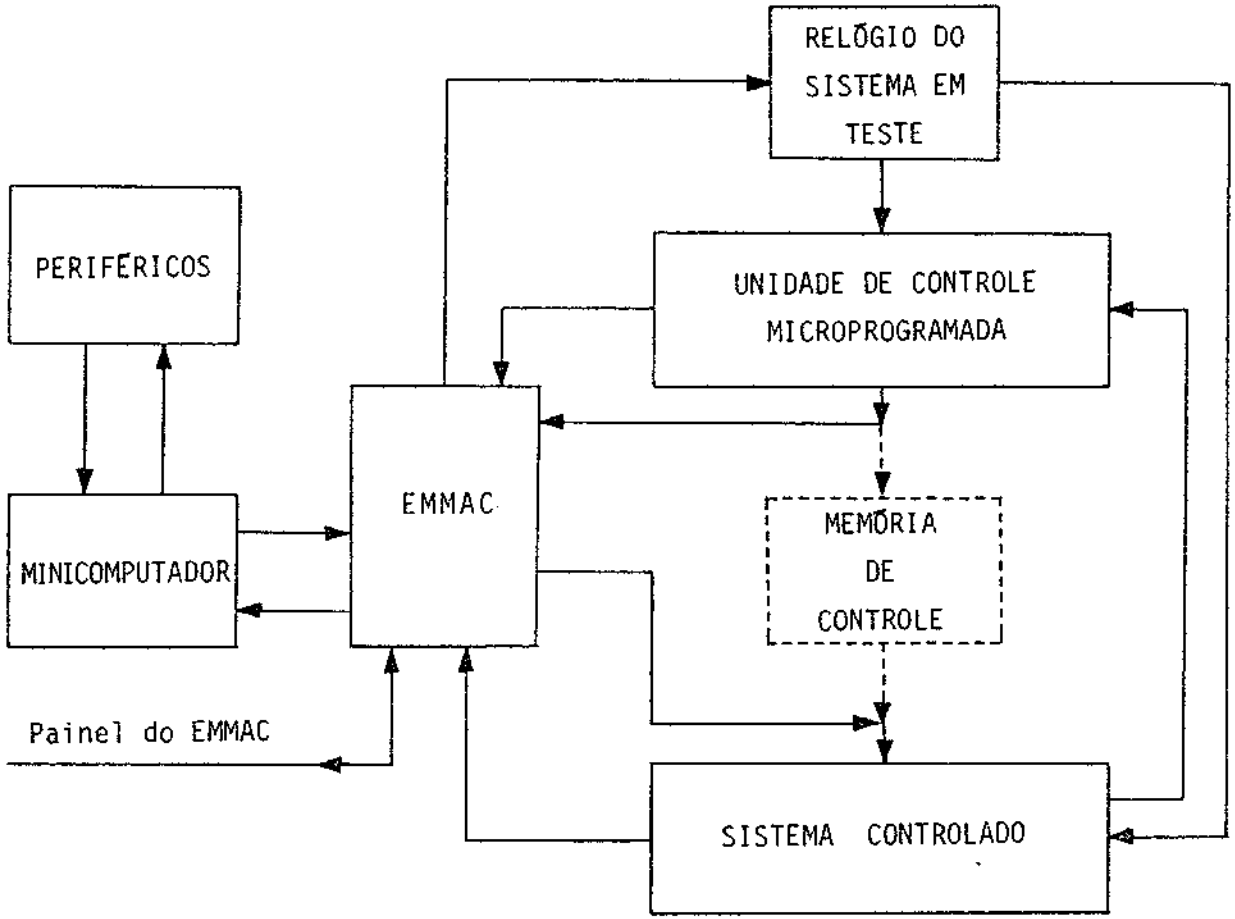

Fig. I.9 - Idéia do emulador de memórias de microcontrole auxiliado por computador (EMMAC).

O sistema proposto será acoplado ao minicomputador HP 2116B e consistirá de uma parte de "software", residente no minicomputador, que tornará a manipulação e análise de microprogramas, armazenados nas memórias de controle, bastante fácil e eficiente; e de uma parte de "hardware" que processará os comandos gerados pelo programa no minicomputador ou por chaves no painel, a fim de emular os PROMs de controle do projeto microprogramado, com a utilização de memórias do tipo RAM de alta velocidade, além de prover facilidades quanto ao controle do relógio da montagem em teste e facilitar a execução de rotinas de diagnose para o sistema já montado.

Este sistema para monitoramento e testes de circuitos microprogramados deverá permitir todas estas facilidades, sem causar alterações fundamentais no "hardware" do sistema a ser testado, além da substituição de suas memórias de controle pelos terminais de saída do sistema de monitoramento.

CAPÍTULO II

FUNÇÕES DO SISTEMA

O Emulador de Memórias de Microcontrole Auxiliado por Computador (EMMAC) visa facilitar ao máximo as tarefas de carregar, analisar e corrigir erros de microprogramas ("firmware") armazenados nas memórias de controle, bem como de dados armazenados em memórias auxiliares, seja de mapeamento ou de nanoinstruções (memórias utilizadas para reduzir o comprimento da palavra de controle), tanto na fase de projeto do sistema e de suas alterações, como na fase em que é possível rodar, para teste, rotinas de diagnose durante a manutenção, com o objetivo de detectar possíveis falhas do sistema.

Este sistema de emulação foi desenvolvido, apoiado no minicomputador HP2116B e seus periféricos, mas na falta deste, ele também pode ser utilizado como um equipamento de bancada, se bem que com uma eficiência de utilização bastante reduzida.

Como mostra a Figura II.1, o emulador de memórias de microcontrole consiste de uma parte de "hardware" que deve aceitar instruções ou do minicomputador ou de chaves de painel do emulador, que poderá emular as memórias de controle, de mapeamento ou de nanoinstruções com memórias RAMs de alta velocidade, controlar o relógio do sistema em teste e monitorar, em tempo real, um certo número de pontos de teste criteriosamente escolhidos. Este circuito deve também se comunicar com o minicomputador, quando a ele estiver acoplado. A outra parte é um "software" residente no HP2116B que gera as instruções aceitas pela parte de "hardware", de modo que toda a operação de emulação e análise dos dados, armazenados nas diversas memórias de controle, pode ser devidamente controlada.

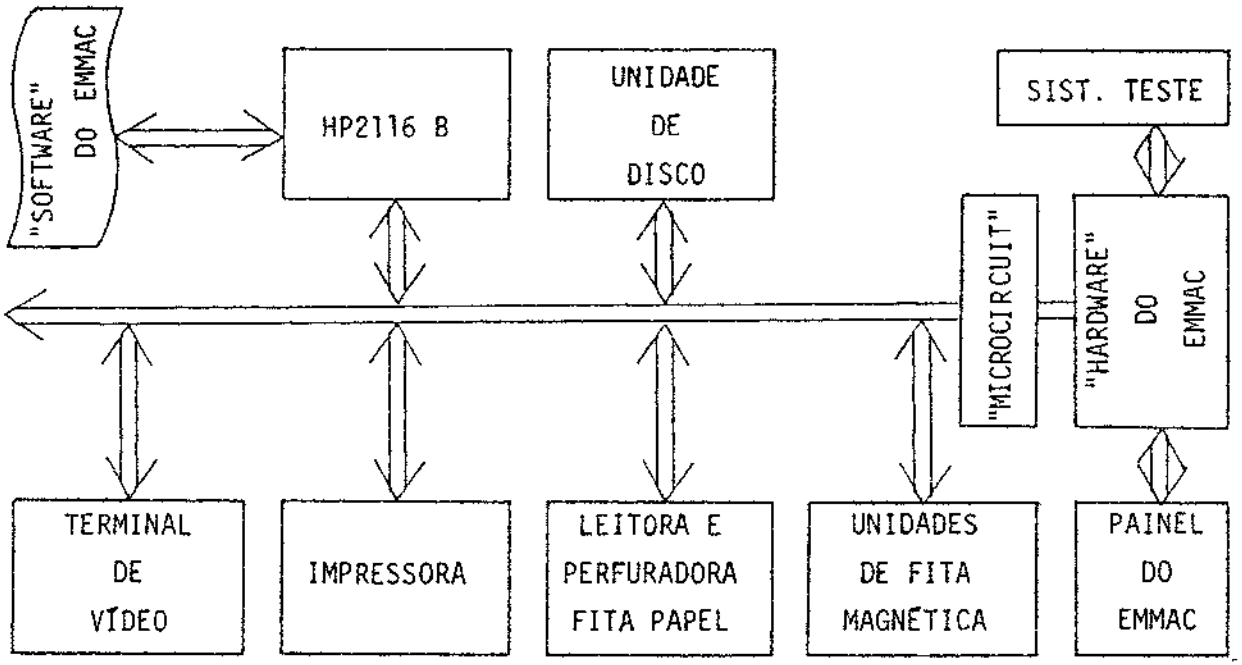

Fig. II.1 - Configuração do emulador de memórias de microcontrole auxiliado por computador (EMMAC).

2.1 - O SISTEMA HP2116B DE APOIO AO EMMAC

O emulador de memórias de microcontole, auxiliado por computador, foi desenvolvido, apoiado no sistema minicomputador HP2116B do INPE.

Este sistema, como mostra a Figura II.1, consiste de:

1) Unidade central de processamento HP2116B, com palavra de 16 "bits", memória de 16K palavras, ciclo de memória de 1,6Ms, e um conjunto de 70 instruções;
2) Unidade de disco modelo HP12960A, com capacidade de 4,9M "bits" e um tempo médio de procura de 30ms;
3) Terminal de video modelo TVA80 da Scopus, com 80 caracteres por linha;
4) Impressora de linha modelo HP12987A, com sistema de impressão do tipo matriz de pontos, com 132 caracteres por linha e uma velocidade de 200 linhas por minuto;
5) Duas unidades de fita magnética modelo D 3030 da Datamec, com densidade de gravação de 800 "bits" por polegada;
6) Leitora de fita de papel modelo HP12925A, com velocidade de leitura de até 500 caracteres por segundo;
7) Perfuradora de fita de papel modelo HP12926A, com velocidade de perfuração de até 75 caracteres por segundo.

O sistema operacional é o sistema DOS III ("Disc Operating System") tendo suporte para as linguagens ASSEMBLER, BASIC,FORTRAN e ALGOL.

2.2 - FUNÇÕES DO SISTEMA EMULADOR DE MEMÓRIAS DE MICROCONTROLE AUXILIADO POR COMPUTADOR

As funções do sistema emulador de memórias de microntrole auxiliado por computador podem ser divididas em duas classes, dependen

do do estado em que se encontra o sistema a ser testado. Estas duas são:

1) Funções do sistema para testes e detecção de erros na fase de projeto e alterações em microprogramas;

2) Funções de sistemas para manutenção e diagnose do sistema microprogramado, após sua confecção.

2.2.1 - FUNÇÕES DO SISTEMA PARA TESTES E DETECÇÃO DE ERROS NA FASE DE PROJETO E ALTERAÇÕES EM MICROPROGRAMAS

Durante a fase do projeto ou de sua alteração, a utilidade do emulador consiste em validar e corrigir, se necessário, microprogramas utilizados para implementar os algorítmos do sistema em teste.

Para tornar esta tarefa rápida, fácil e eficiente é preciso que certas funções sejam executadas pelo sistema. Estas funções são as seguintes:

1) Controle do relógio do sistema em teste.

O emulador deverá controlar o relógio do sistema para que a depuração dos microprogramas seja fácil. Este controle deve ter as seguintes capacidades:

a) De parar o relógio;

b) De executar o microprograma passo a passo;

c) De executar um certo número especificado de passos;

d) De executar um microprograma até que um endereço determinado da memória de controle seja atingido;

e) Deixar o relógio livre.

2) Inicialização do endereço da memória de controle e dos dispositivos por ela controlados.

3) Ler e escrever o conteúdo de um módulo de memória, previamente selecionado, em qualquer periférico do minicomputador HP2116B.

Esta facilidade é útil quando se deseja verificar o conteúdo da memória de controle, alterá-lo se preciso e também sair com os dados em um formato conveniente para utilização em um programador de memórias PROMs, diretamente acoplado ao HP2116B.

4) Salvar o estado atual do sistema emulador em um dado arquivo e recuperá-lo quando necessário.

Esta facilidade é útil quando se deseja, por algum motivo, interromper o teste em execução. Assim sendo, quando for preciso retomá-lo, as condições anteriores serão facilmente restabelecidas.

5) Monitorar, em tempo real, um certo número prescrito de pontos do sistema em teste, em uma dada frequência submúltipla da frequência do relógio.

O sistema de emulação de memórias de microcontrole deverá ser tal que estas funções sejam executadas com modificações mínimas na placa do circuito em teste (se possível, nenhuma além de substituir as PROMs de controle por soquetes especiais conectados a RAMs que as emulem).

2.2.2 - FUNÇÕES PARA MANUTENÇÃO E DIAGNOSE DO SISTEMA MICROPROGRAMADO, APÓS SUA CONFECÇÃO

Para se fazer a manutenção e diagnose do sistema em teste, ou seja, executar testes nos circuitos do mesmo, a fim de detectar erros na forma mais precisa possível e, caso existam, o lugar onde ocorrem estas falhas, o emulador de memórias de microcontrole deverá ser capaz de executar as seguintes funções:

1) Ter possibilidade de fazer rodar e armazenar seja em disco, fita de papel ou fita magnética, programas de diagnose para cada unidade componente do projeto, funcionando corretamente e guardando os estados de diversos pontos do circuito em instantes de tempo diferentes, ou seja, para diversas funções executadas pelo sistema analisado;

2) Substituir as memórias de controle do sistema a ser testado sem alterá-lo fisicamente e carregar nas memórias substitutas os mesmos microprogramas de diagnose da função 1, e executá-los, armazenando então os estados dos mesmos pontos do circuito, sob diagnose;

3) Comparar os resultados obtidos na função 2 com os dados obtidos na função 1 e imprimir mensagens de erro informando, se necessário, em que pontos do circuito ocorreram discrepâncias.

Com estas três funções, basta que o sequenciador da unidade de controle do sistema esteja funcionando corretamente, para que todas as outras unidades do mesmo possam ser testadas por diagnose automática.

Este tipo de diagnose tem a vantagem de não ocupar nenhuma posição da memória de controle do projeto, permitindo que as rotinas de diagnose possam ser grandes, tornando o nível de detecção de falhas bastante acurado.

CAPÍTULO III

PROJETO DO EMULADOR DE MEMÓRIAS DE MICROCONTROLE

O objetivo deste capítulo é dar uma idéia de como foi projetado o circuito do emulador de memórias de microcontrole auxiliado por computador.

A implementação deste circuito foi feita utilizando a técnica de microprogramação, o que torna o circuito de controle bem mais fácil de ser implementado, além de tornar a adaptação deste circuito para outros tipos de computadores bastante imediata e simplificada.

O circuito do EMMAC deve ter a capacidade de executar todas as funções, anteriormente descritas no capítulo II, devendo ter ainda as seguintes características:

1) Adaptar-se ao minicomputador, de tal forma que sejam permitidas a escrita e a leitura de qualquer uma de suas memórias, bem como o controle de todas as funções requeridas;

2) Simular qualquer tipo de memória PROM, desde que esta memória tenha no máximo a configuração de 512 palavras de 8 "bits" e até 2 controles de "tri-state", do tipo ativo baixo. Como exemplo, podemos citar os PROMs da "Texas Instruments" SN74S288, SN74S287, SN74S471 e SN74S472;

3) Quando o sistema emulador estiver funcionando no modo de emulação em tempo real, o tempo de acesso típico de leitura de suas memórias deverá ser menor do que 80 nanosegundos. Esta característica permitirá que a maioria dos dispositivos microprogramados, utilizando a tecnologia atual, tenha os seus microprogramas testados na sua máxima frequência de relógio, ou seja, em tempo real;

4) Um mínimo de modificações no sistema em teste (se possível nenhuma) deverá ser feito para adaptar o mesmo ao sistema emula

dor, de tal forma que a utilização deste seja possível, mesmo quando o sistema em teste esteja na forma de um protótipo final;

5) Aceitar os microprogramas na forma gerada pela linguagem de geração de microcódigo para sistemas microprogramados a ser desenvolvida;

6) Ser capaz de emular uma quantidade de "bits" de palavra de memória de microcontrole, para cobrir uma faixa considerável de projetos microprogramados, em especial, para os projetos que estão em desenvolvimento no INPE.

O circuito do emulador de memórias de microcontrole pode ser subdividido nas seguintes partes:

1) Interface com o minicomputador HP2116B;

2) Circuito interno do emulador, que, por sua vez, pode ser subdividido em:

   a) Circuito para emulação das memórias de microcontrole;

   b) Circuito de controle do relógio do sistema em teste;

   c) Circuito de monitoramento das pontas de teste;

   d) Circuito de controle do emulador.

3) Interface com o sistema em teste.

3.1 - INTERFACE COM O MINICOMPUTADOR HP2116B

A interface entre o sistema emulador de memórias de microcontrole e o minicomputador HP2116B foi feita através do cartão de interface "microcircuit", existente no minicomputador (Hewlett-Packard, 1968).

Este cartão de interface permite a transferência bidirecional de dados, em linhas separadas, entre o minicomputador e um dispositivo de entrada e saída, empregando lógica DTL/TTL, com níveis de tensão e atrazos compatíveis. Esta transferência é feita utilizando-se

o sistema de interrupção de entrada e saída ou a opção de testes de "flag" do dispositivo, bastando para tanto alterar-se a posição de um "jumper" do cartão de "microcircuit".

Um diagrama de blocos, mostrando a conexão entre o sistema emulador e o minicomputador, é apresentado na Figura III.1

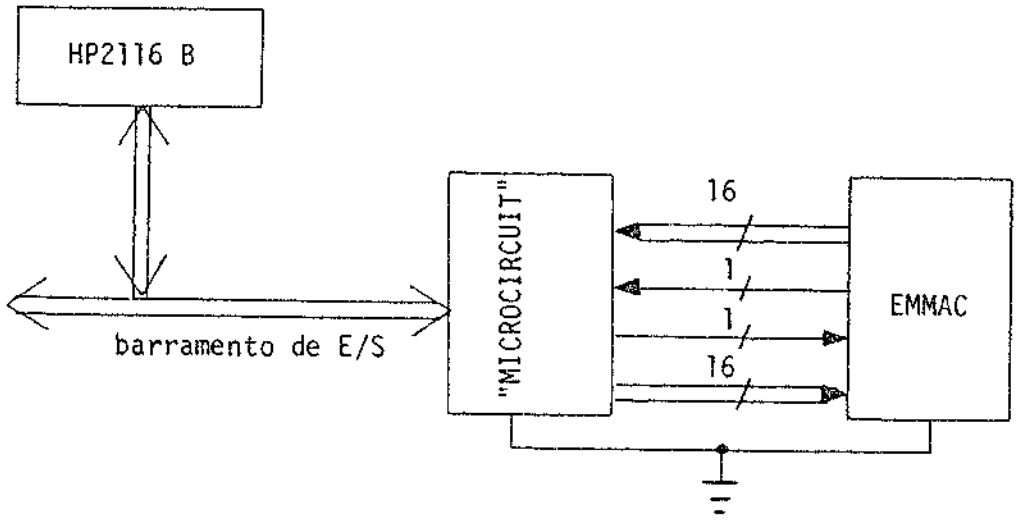

Fig. III.1 - Diagrama de blocos da conexão entre o sistema emulador e o HP2116B.

Um diagrama de blocos, bem simplificado, da interface "microcircuit" é apresentado na Figura III.2.

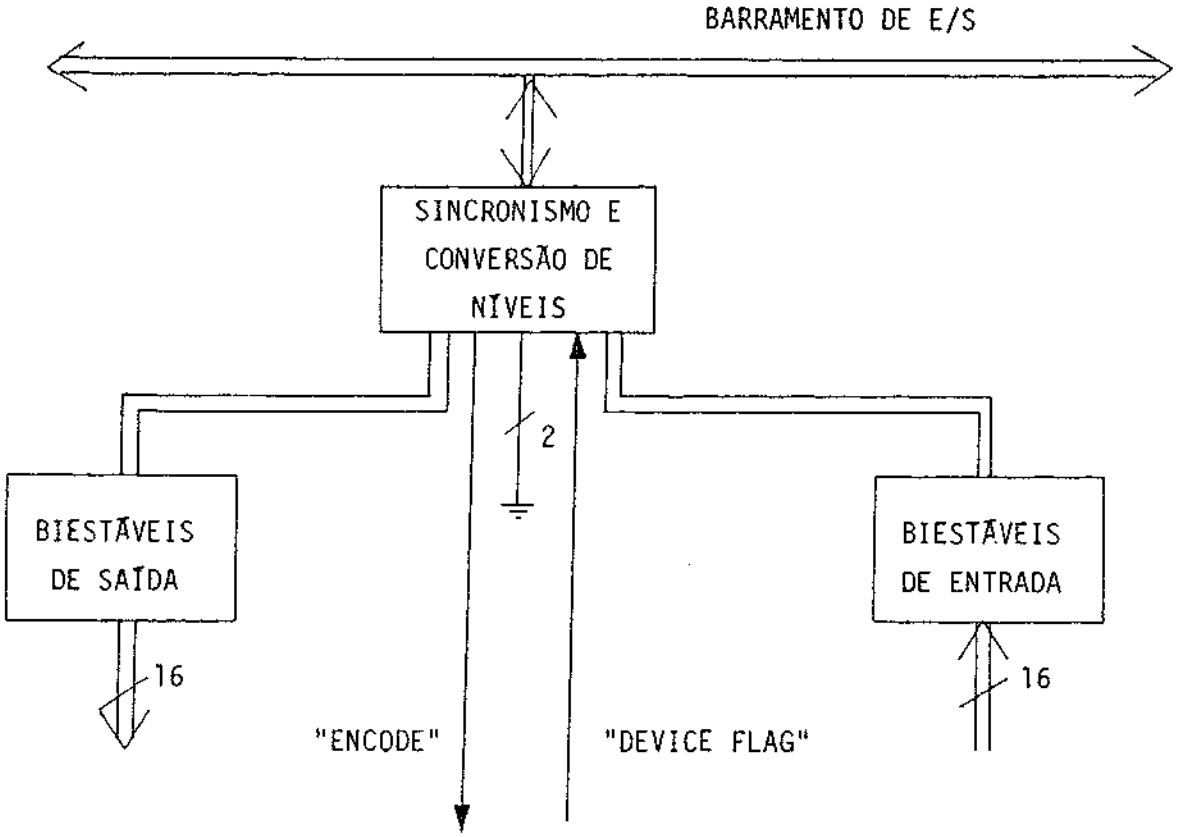

Fig. III.2 - Diagrama de blocos simplificado da interface "microcircuit".

Esta interface é composta de um conjunto de 16 biestáveis, controlados pelo minicomputador para a saída de dados ou controle para o dispositivo externo, e outro conjunto de 16 biestáveis, que é controlado pelo dispositivo externo para a entrada de dados no minicomputador. Um sinal chamado "device flag" é entrada do cartão da "microcircuit", fazendo com que os dados sejam carregados nos registros de entrada e, ao mesmo tempo, gerando um pedido de interrupção para o minicomputador. Um sinal chamado "encode" é saída do cartão interface "microcircuit", codificando o tempo para o dispositivo externo e, opcionalmente, carregando os registros de saída da interface com o conteúdo de um dos seus acumuladores.

3.2 - CIRCUITO INTERNO DO EMULADOR

Para que as funções do emulador de memórias de microcontrole, descritas anteriormente no capítulo II, possam ser executadas, 4 unidades de circuitos funcionais são necessárias. Estas unidades são as seguintes:

1) Circuito para emulação das memórias de microcontrole;

2) Circuito de controle do relógio do sistema em teste;

3) Circuito de monitoramento das pontas de teste;

4) Circuito de controle do emulador.

A interconexão entre estas 4 unidades de circuito, bem como a interconexão com o minicomputador são apresentadas no diagrama de blocos da Figura III.3. Cada uma destas unidades funcionais será apresentada no diagrama de blocos nos itens que se seguem.

Nestes diagramas de blocos foi adotada a convenção para os sinais de entrada e saída, simbolizada na Tabela III.1.

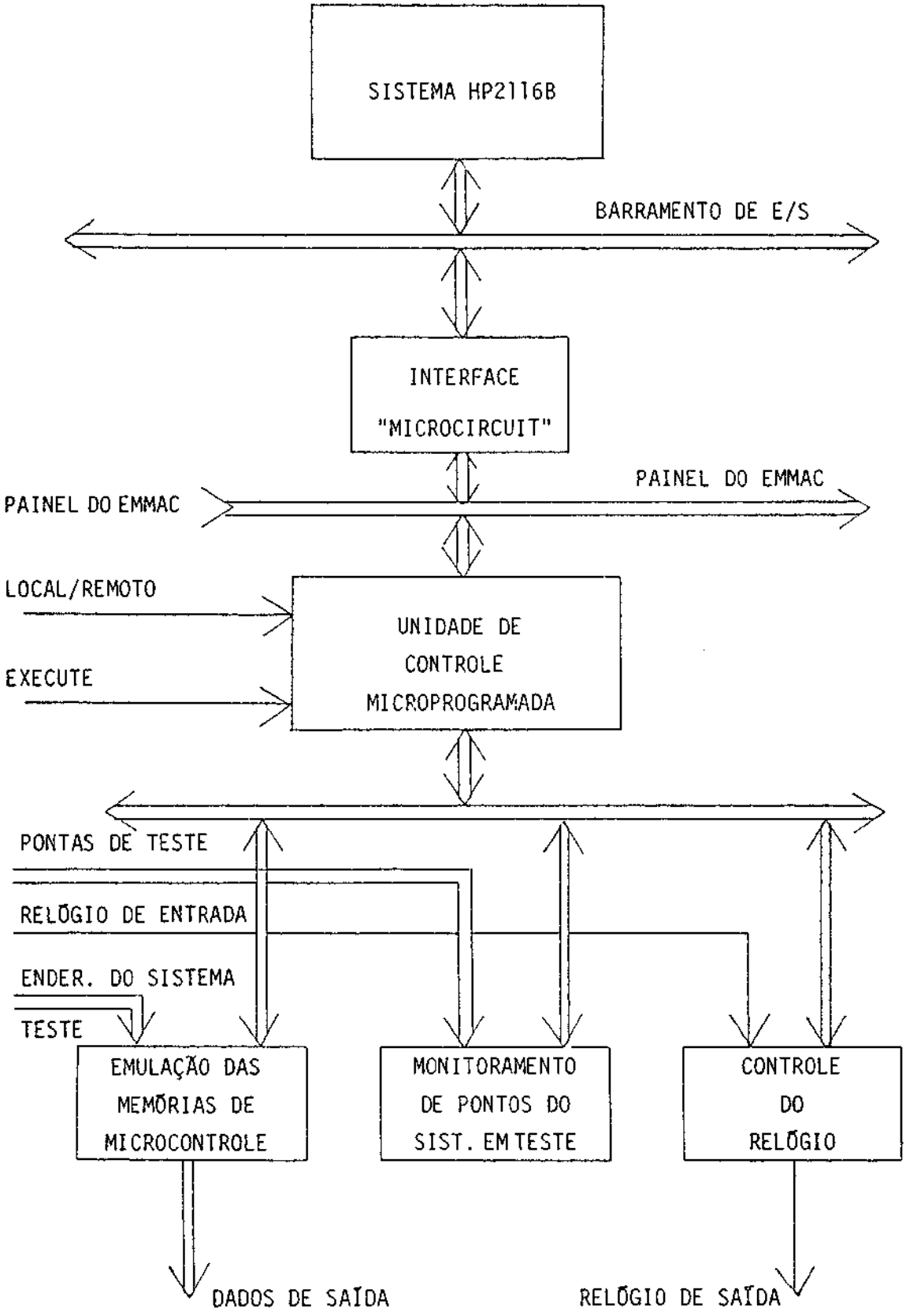

Fig. III.3 - Diagrama de blocos do sistema emulador (EMMAC).

TABELA III.1

CONVENÇÃO PARA OS SINAIS DE ENTRADA E SAÍDA

▷	SINAL VINDO DO REGISTRO DE INSTRUÇÃO DO EMULADOR
▶	SINAL PARA O REGISTRO DE SAÍDA DE DADOS DO EMULADOR
≻	SINAL VINDO DO SISTEMA EM TESTE
⪢	SINAL PARA O SISTEMA EM TESTE
⇨	SINAL VINDO DA MEMÓRIA DE MICROCONTROLE DO EMULADOR
⊳	SINAL PARA OUTRA UNIDADE INTERNA DO EMULADOR

3.2.1 - CIRCUITO PARA EMULAÇÃO DAS MEMÓRIAS DE MICROCONTROLE

Como está representado na Figura III.4, o circuito para emulação das memórias de microcontrole é composto de um módulo básico para a emulação de memórias tipo PROM, de até 512 palavras de 8 "bits", com até dois controles de alta impedância ("tri-state").

O circuito para a emulação das memórias de controle é capaz de conter até 20 destes módulos, o que permite uma palavra de controle de 160 "bits".

O controle deste circuito de emulação de memórias de microcontrole é feito pelos "bits" vindos da memória de controle da unidade de controle microprogramada. Estes "bits" coordenam as atividades de leitura e escrita nos módulos de emulação, bem como o chaveamento do acesso para os módulos.

As saídas de dados para o sistema teste é feita através de "latches" cada um possuindo dois controles independentes de "tri-state", definidos apenas pelo circuito do sistema em teste. Estes controles permitem o acesso a leitura das memórias do módulo de emulação, mesmo quando estas memórias estiverem desvinculadas pela lógica no sistema em teste (alta impedância).

As memórias RAMs do módulo de emulação podem ser endereçadas pelo barramento de endereços, vindos do sistema em teste, ou pela saída do contador de endereços. Este chaveamento é controlado pelo "bit" de controle SE.

O registro de seleção e os decodificadores de leitura e de escrita selecionam qual dos 20 módulos de emulação de PROMs do sistema emulador será endereçado para leitura e/ou escrita.

O contador de endereços serve para endereçar uma dada posição de uma memória de um módulo de memória, a ser lida ou escrita, e, ao mesmo tempo, preparar o acesso que deverá ser feito na posição seguinte.

Os mnemônicos utilizados no circuito, que aparecem na Figura III.4, e os seus significados, estão descritos na Tabela III.2.

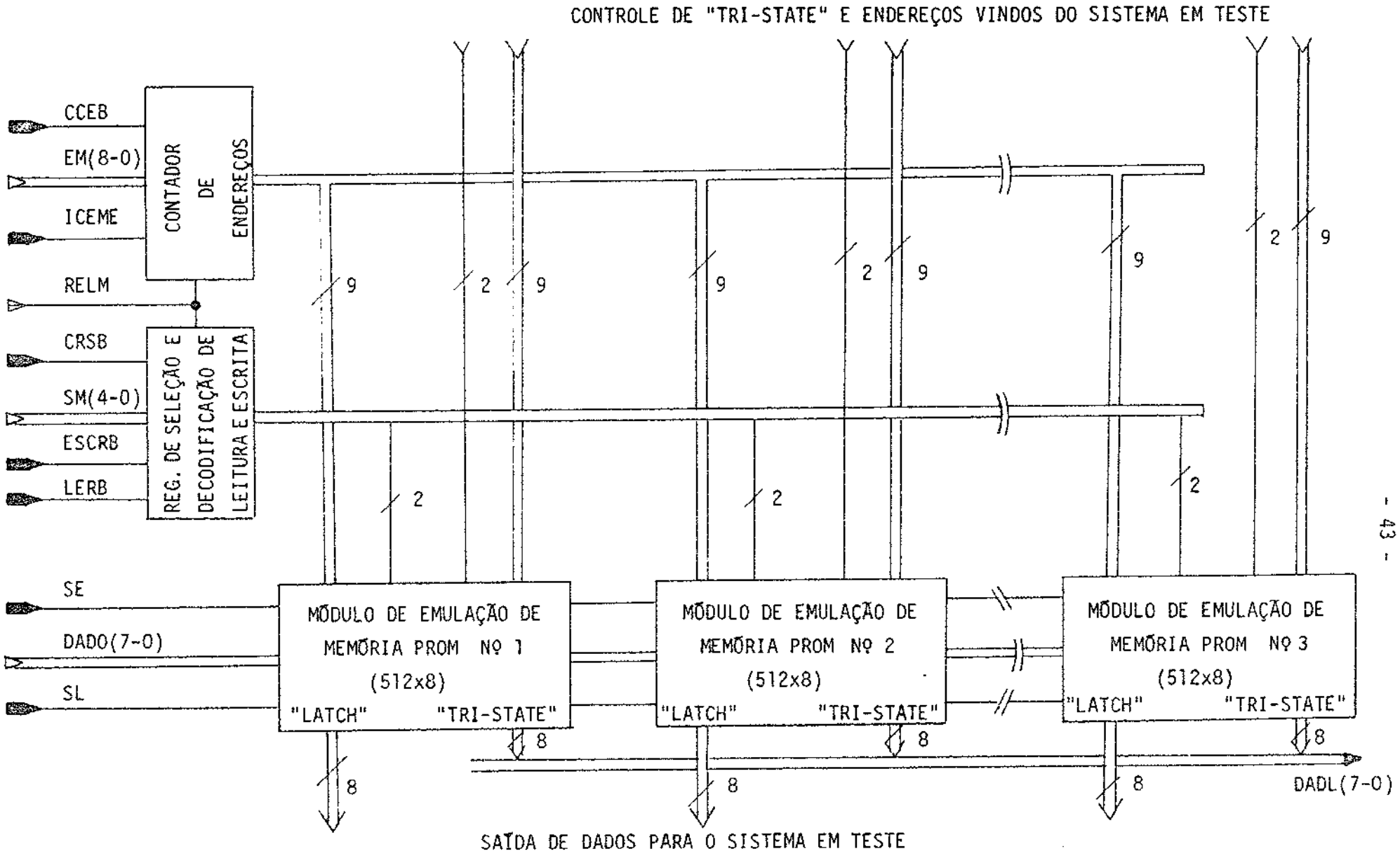

Fig. III.4 - Circuito para emulação das memórias de microcontrole.

TABELA III.2

SIGNIFICADO DOS MNEMÔNICOS DO CIRCUITO DA FIGURA III.4

CCEB	..... CARREGA O CONTADOR DE ENDEREÇO
EM	..... ENDEREÇO DA MEMÓRIA DO SISTEMA EMULADOR
RELM	..... RELÓGIO DO SISTEMA EMULADOR
SE	..... SELECIONA ENDEREÇAMENTO DOS MÓDULOS DE EMULAÇÃO
SM	..... SELECIONA MÓDULO DE EMULAÇÃO A SER LIDO OU ESCRITO
CRSB	..... CARREGA REGISTRO DE SELEÇÃO
ESCRB	..... CONTROLE DE ESCRITA DO MÓDULO DE EMULAÇÃO
DADO	..... DADO A SER ESCRITO NO MÓDULO DE EMULAÇÃO SELECIONADO
ICEME	..... INCREMENTA CONTADOR DE ENDEREÇOS DO MÓDULO DE EMULAÇÃO
LERB	..... CONTROLE DE LEITURA DO MÓDULO DE EMULAÇÃO
SL	..... SEGURADOR DO "LATCH" DE SAÍDA DO MÓDULO DE EMULAÇÃO
DADL	..... DADO LIDO DE UM MÓDULO DE EMULAÇÃO SELECIONADO

O procedimento para se ler ou escrever um certo dado, em um dado módulo de emulação do sistema emulador é o seguinte:

1) Seleciona-se a memória a ser lida ou escrita, carregando-se o valor correspondente do módulo no registro de seleção;

2) Carrega-se o endereço da primeira posição, a ser lida ou escrita, no registro contador de endereços;

3) Dá-se um comando de leitura ou escrita através dos sinais de controle LERB ou ESCRB que lê o dado da posição selecionada e o coloca no barramento DADL(7-0), transferindo-o para o registro de saída ou escreve o dado do barramento DADO(7-0) na posição selecionada do módulo de emulação selecionado, incrementan

do ou não automaticamente o contador de endereços;

4) Se outro dado precisar ser lido ou escrito na posição seguinte da mesma memória, que é o caso mais comum, apenas o passo número 3 precisará ser executado.

Este procedimento é bastante eficiente para o acesso a posições sequenciais de um mesmo módulo de memória, que é o caso mais comum existente nos comandos da linguagem a ser descrita no capítulo IV.

Quando o comando EXECUTE for dado, o controle SE deverá selecionar o barramento de endereços, vindo do sistema em teste, e o controle SL deverá deixar os "latches" de saída destravados.

3.2.2 - CIRCUITO DE CONTROLE DO RELÓGIO DO SISTEMA EM TESTE

Como foi especificado, nas funções definidas no capítulo II, são 4 os tipos possíveis de controle do relógio do sistema em teste:

1) Relógio passo-a-passo;
2) Relógio livre;
3) Relógio livre durante um certo número de pulsos;
4) Relógio livre até um certo endereço da memória de controle ser atingido.

O circuito que controla estas 4 condições é apresentado na Figura III.5.

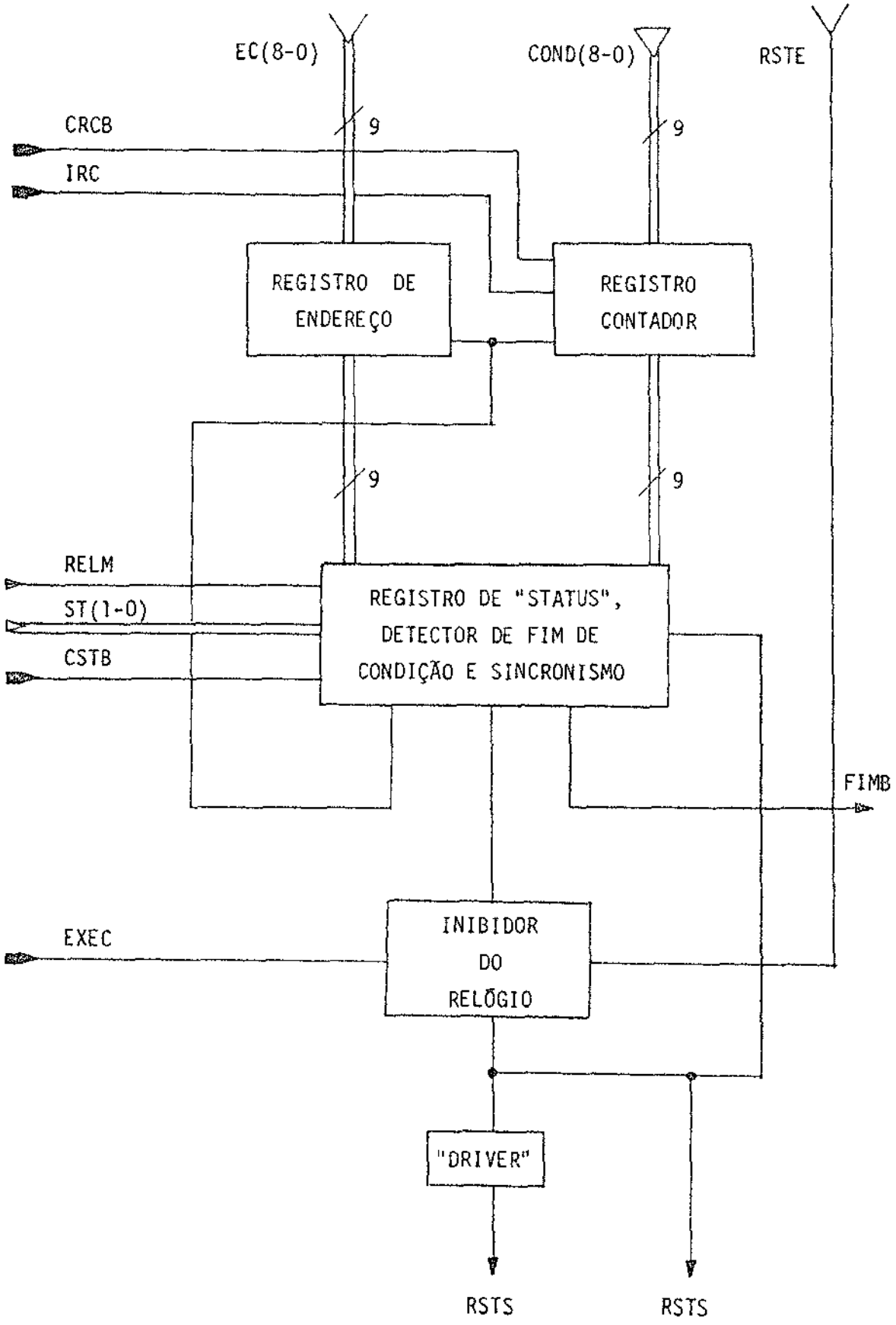

Fig. III.5 - Circuito de controle do relógio do sistema em teste.

Os mnemônicos dos vários sinais de controle deste circuito, bem como os seus significados são apresentados na Tabela III.3.

TABELA III.3

MNEMÔNICOS DO CIRCUITO DE CONTROLE DO RELÓGIO, DO SISTEMA EM TESTE, E OS SEUS SIGNIFICADOS

RSTE	RELÓGIO FORNECIDO PELO SISTEMA EM TESTE (ENTRADA)
RSTS	RELÓGIO PARA O SISTEMA EM TESTE (SAÍDA)
COND	CONDIÇÃO DE PARADA DO RELÓGIO PARA O SISTEMA EM TESTE
IRC	INCREMENTA O REGISTRO CONTADOR
EC	ENDEREÇO DE CONTROLE
CRCB	CARREGA O REGISTRO CONTADOR
RELM	RELÓGIO DO SISTEMA EMULADOR
ST	"STATUS" DO MODO DE CONTROLE DO RELÓGIO
CSTB	CARREGA O REGISTRO DE "STATUS"
EXEC	EXECUTAR (LIBERA O RELÓGIO NO "STATUS" DEFINIDO)
FIMB	FIM (CONDIÇÃO DE PARADA FOI ATINGIDA)

O relógio originário do sistema em teste é conectado à entrada RSTE e a saída RSTS será o novo relógio do sistema em teste.

Esta saída é controlada por um "bit" da memória de controle do sistema emulador, EXEC, sincronizado com o relógio de entrada, e pela saída do dectetor de fim de condição, cujas entradas são também sincronizadas com o relógio de entrada.

A escolha do modo de controle do relógio do sistema em teste é selecionada, carregando-se o registro de "status" pela entrada ST(1-0), ficando esta condição selecionada até que um novo valor seja carregado neste registro.

O valor no registro de "status" varia de 0 a 3,correspondendo respectivamente aos modos passo-a-passo, livre, livre durante um certo número de pulsos e livre até um certo endereço da memória ser atingido.

Tanto o número de pulsos desejados, como o endereço a ser atingido deverão ser carregados no registro contador antes de um comando de execução, dado pelo "bit" de controle EXEC, ser executado.

Um sinal de fim de condição é enviado para a unidade de controle, através do sinal FIMB, para indicar que o fim de condição foi atingido.

3.2.3 - CIRCUITO DE MONITORAMENTO DAS PONTAS DE TESTE

Este circuito tem por finalidade armazenar até 256 últimos estados, de até 32 pontas de testes conectadas a pontos escolhidos do sistema em teste, com uma dada frequência de amostragem, em tempo real.

O esquema da Figura III.6 consiste de uma lógica de controle de armazenamento, de um contador de endereços, de até 4 módulos de monitoramento, e de uma lógica de decodificação de leitura.

Os mnemônicos dos sinais utilizados neste circuito, bem como os seus significados estão apresentados na Tabela III.4.

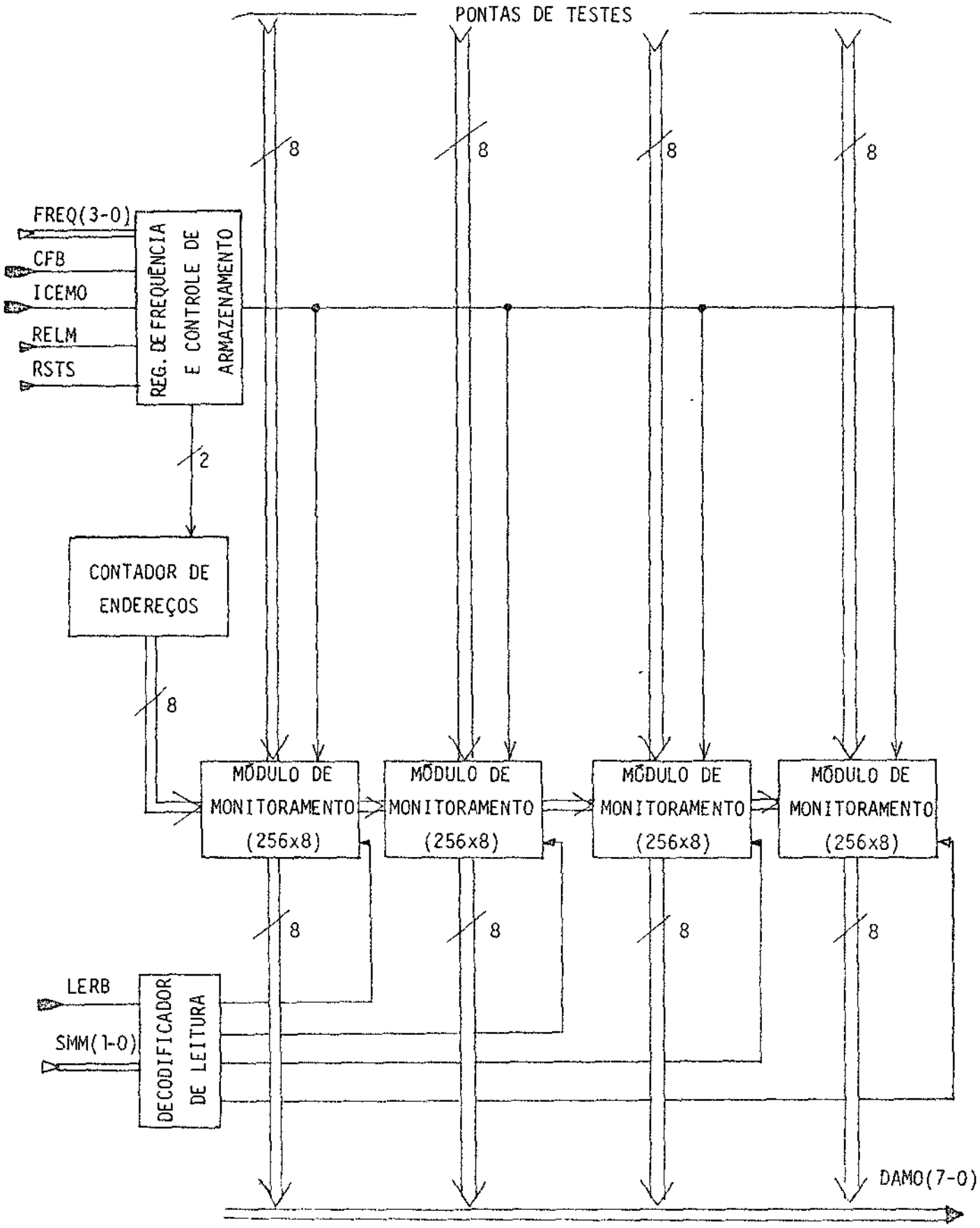

Fig. III.6 - Circuito de monitoramento das pontas de testes em tempo real.

TABELA III.4

MNEMÔNICOS DO CIRCUITO DE MONITORAMENTO DAS PONTAS DE TESTE

ICEMO	INCREMENTA O CONTADOR DE ENDEREÇOS
CFB	CARREGA O REGISTRO DE FREQUÊNCIA
FREQ	FREQUÊNCIA DE ARMAZENAMENTO DAS PONTAS DE TESTE
DAMO	DADO DE SAÍDA DOS MÓDULOS DE MONITORAMENTO
SMM	SELECIONA MÓDULO DE MONITORAMENTO A SER LIDO
LERB	CONTROLE DE LEITURA DO MÓDULO DE MONITORAMENTO

O registro contador de frequência é carregado com o valor desejado e decrementado a cada pulso RSTS do relógio. Quando o seu valor atinge zero, o circuito de controle de armazenamento gera um sinal que na próxima subida do relógio RSTS faz com que os estados das pontas de teste sejam armazenados nos módulos de monitoramento, o registro contador de endereços seja incrementado e o valor do registro contador de frequência seja restaurado com o valor inicial, recomeçando o ciclo de contagem de frequência.

O registro de frequência é inicialmente carregado com o valor presente no barramento FREQ(3-0), por uma instrução do sistema emulador.

A leitura dos estados armazenados nas memórias do tipo RAM existentes nos módulos de monitoramento, é feita de 8 em 8 "bits" através do barramento DAMO(7-0), selecionado pelo decodificador de leitura, através do barramento SMM(1-0) e do sinal vindo da memória de microcontrole do sistema emulador LERB.

3.2.4 - CIRCUITO DE MICROCONTROLE DO SISTEMA EMULADOR

O circuito de microcontole do sistema emulador, representado na Figura III.7, será microprogramado, podendo funcionar no modo LOCAL ou no modo REMOTO.

A vantagem do circuito ser microprogramado é que,além de facilitar a sua implementação, torna o "hardware" do sistema independente do "software" do minicomputador que o supervisiona. Isto torna o sistema emulador adaptável, por simples mudança no "firmware", para aceitar comandos gerados por outro computador.

No modo LOCAL, o sistema emulador aceita instruções das 16 chaves situadas no painel e as executa quando a chave EXECUTE for pressionada.

No modo REMOTO, o sistema emulador recebe instruções do minicomputador HP2116B, através do registro de saída da interface "microcircuit", e as executa quando recebe o sinal "ENCODE",também proveniente da "microcircuit".

Uma vez terminada a execução de uma instrução, se houver algum dado como resposta, eles serão colocados no registro de saída do sistema emulador e, se o modo for REMOTO, um sinal "DEVICE FLAG" é enviado para o minicomputador, através da interface "microcircuit", indicando que o minicomputador já pode ler o dado.

As instruções do sistema emulador são apresentadas na Tabela III.5.

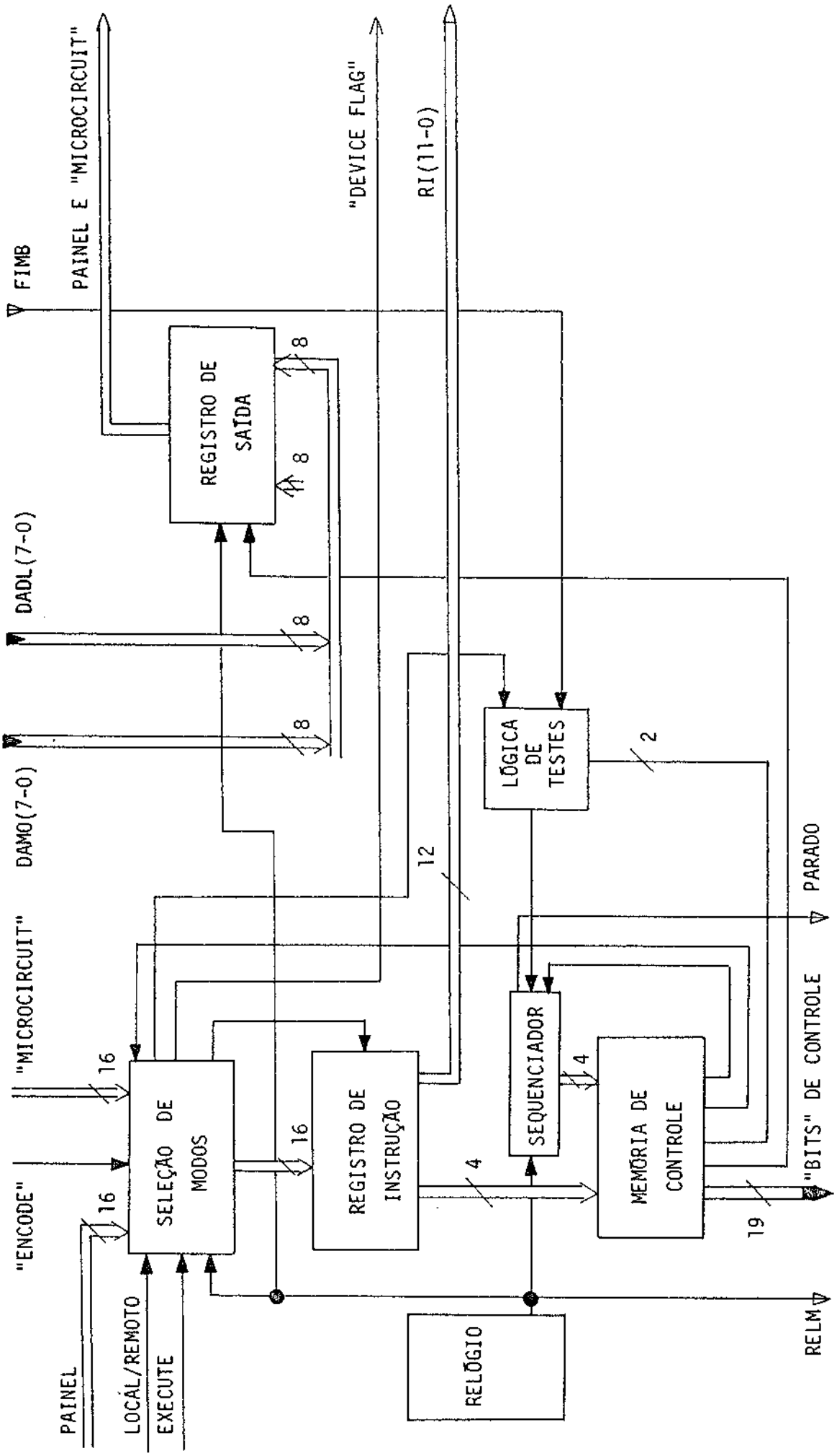

Fig. III.7 - Circuito de controle do emulador.

TABELA III.5

INSTRUÇÕES IMPLEMENTADAS NO SISTEMA EMULADOR

CÓDIGO	MNEMÔNICO	DESCRIÇÃO
00	LERI m	LER O MÓDULO DE EMULAÇÃO E INCREMENTAR O CONTADOR
01	ESCR m	ESCREVER m NO MÓDULO DE EMULAÇÃO SELECIONADO
02	CARSE m	CARREGAR O REGISTRO DE SELEÇÃO COM O VALOR m
03	CACES m	CARREGAR m NO CONTADOR DE ENDEREÇO DO CIRCUITO DE EMULAÇÃO
04	CARC m	CARREGAR m NO REGISTRO CONTADOR DO CIRCUITO DE CONTROLE DO RELÓGIO
05	CARS m	CARREGAR m NO REGISTRO DE "STATUS" DO CONTROLE DO RELÓGIO
06	CARF m	CARREGAR m NO REGISTRO DE FREQUÊNCIA DO CIRCUITO DE MONITORAMENTO
07	INCMO	INCREMENTAR O CONTADOR DE ENDEREÇO DO CIRCUITO DE MONITORAMENTO
08	LERMO m	LER O DADO DO MÓDULO DE MONITORAMENTO m
09	EXEC	EXECUTE LIBERANDO O RELÓGIO NO MODO ESPECIFICADO
10	- - -	não utilizado
11	LER m	LER m DO MÓDULO DE EMULAÇÃO SELECIONADO
12	ESCRI m	ESCREVER m NO MÓDULO DE EMULAÇÃO E INCREMENTAR O CONTADOR
13	EXECC	EXECUTAR COM O ENDEREÇO VINDO DO CONTADOR DE ENDEREÇOS
14	- - -	não utilizado
15	PARE	INIBE O RELÓGIO DO SISTEMA EM TESTE

Obs.: m é um número a ser carregado ou lido através do sistema emulador

A instrução a ser executada pelo sistema emulador é carregada no registro de instrução da unidade de controle, sendo os 4 "bits" mais significativos reservados para o código de operação da instrução, e os 9 menos significativos, para os dados a serem carregados nas diversas partes do sistema emulador, segundo a Tabela III.6.

TABELA III.6

ASSOCIAÇÃO DO CAMPO DE DADOS DO REGISTRO DE INSTRUÇÃO, COM OS BARRAMENTOS DOS VÁRIOS SUBSISTEMAS DO SISTEMA EMULADOR

BARRAMENTO	"BITS" CORRESPONDENTES DO RI
EM(8-0)	RI(8-0)
SM(4-0)	RI(4-0)
DADO(7-0)	RI(7-0)
COND(8-0)	RI(8-0)
ST(1-0)	RI(1-0)
FREQ(3-0)	RI(3-0)
SMM(1-0)	RI(1-0)

Obs.: Os "bits" do registro de instrução (RI) não referenciados, além dos do código de operação, não são utilizados.

O sequenciamento das microinstruções é feito pelo sequenciador, sendo o seu incremento controlado pela saída da lógica de teste, permitindo incrementos condicionais.

A saída do sequenciador compõe os 4 "bits" menos significativos do endereço da memória de controle, o que limita em 16 o número de microinstruções permitidas para cada instrução do sistema emulador.

Os únicos barramentos que contêm dados para a saída do sistema emulador são os barramentos DADL(7-0) e DADO(7-0), que entram nos 8 "bits" menos significativos do registro de saída, da unidade de controle do sistema emulador.

Este registro é lido pelo minicomputador no final da execução de uma instrução do sistema emulador, quando o sinal "DEVICE FLAG" for enviado para a interface "microcircuit".

O formato da palavra de controle do sistema de monitoramento é apresentada na Figura III.8. É a partir desta palavra de controle que o sistema emulador provê os "bits" de controle para cada subsistema.

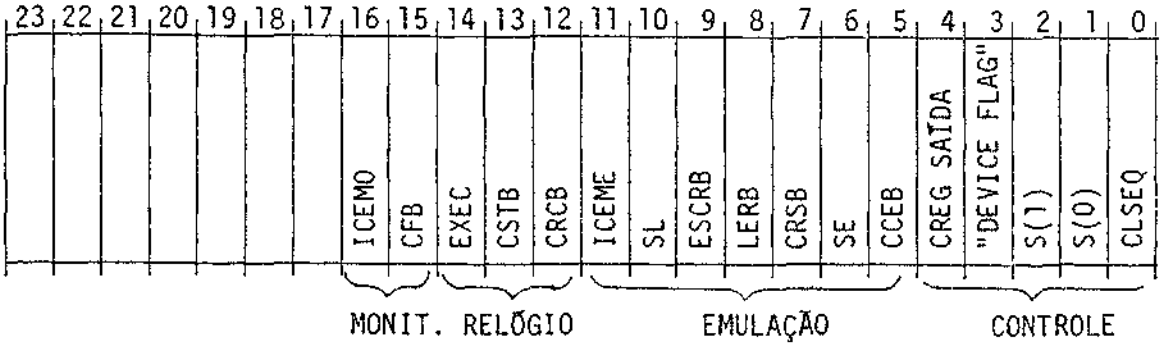

Fig. III.8 - Formato da palavra de controle.

Devido ao pequeno número de "bits" de controle, a microprogramação é estritamente horizontal, permitindo uma grande flexibilidade e eficiência em eventuais modificações funcionais do sistema emulador.

Não foi necessária a utilização de um registro "pipeline" na saída da memória de controle, porque não é necessária uma grande velocidade do sistema, uma vez que as instruções do sistema emulador ape

nas iniciam as condições de emulação, utilizando-se do seu relógio RELM.

A simulação do circuito do EMMAC, utilizando a linguagem de simulação CDL, é apresentada no apêndice A.

3.3 - INTERFACE COM O SISTEMA EM TESTE

Caso a memória PROM, a ser utilizada no sistema em teste, for a PROM SN74S472 (512x8"bits"), nenhuma interface será necessária entre o sistema emulador e o sistema em teste, bastando tirar a PROM do seu soquete e conectar o chicote do sistema emulador.

Caso seja utilizada outra PROM, deverá ser feito um soquete especial que compatibilizará os pinos e a capacidade desta PROM, com os pinos e a capacidade da memória PROM SN74S472.

CAPÍTULO IV

LINGUAGEM DE UTILIZAÇÃO DO SISTEMA EMMAC

Este capítulo tem por finalidade descrever os comandos da linguagem de utilização do sistema emulador de memórias de microcontrole.

Esta linguagem não tem o sentido comum de linguagem de programação, como o FORTRAN ou ALGOL, pois cada comando é analisado e executado de uma maneira interpretativa,ou seja, não existe o sentido de programa, pois cada comando e sua execução são independentes de outros comandos.

Os comandos desta linguagem tem por finalidade gerar instruções para o circuito do emulador de memórias de microcontrole, de tal forma que as funções do sistema, descritas no capítulo II, sejam corretamente implementadas.

Um conjunto minimizado de comandos, utilizando a filosofia dos comandos do sistema operacional HP2116B (DOS III),(Hewlett, Packard, 1973), foi considerado e adotado, sendo que todos os comandos deste conjunto são emitidos através do terminal de vídeo do minicomputador.

Este conjunto é formado dos seguintes comandos:

1 - ABORTE
2 - ARMAZENE
3 - COPIE
4 - DIAGNOSE
5 - ESCREVA
6 - EXECUTE
7 - FIM
8 - FREQUENCIA
9 - INICIE
10 - LEIA
11 - LISTE
12 - MONITOR
13 - MUDE
14 - PARE
15 - PFITA
16 - RELOGIO
17 - SIGA
18 - TEMINAL
19 - TROQUE

A implementação destes comandos foi feita utilizando a linguagem ALGOL do HP2116B, sendo que o código gerado ocupou, aproximadamente, 8K palavras de 16 "bits".

O diagrama de execução dos comandos é apresentado na Figura IV.1, abaixo:

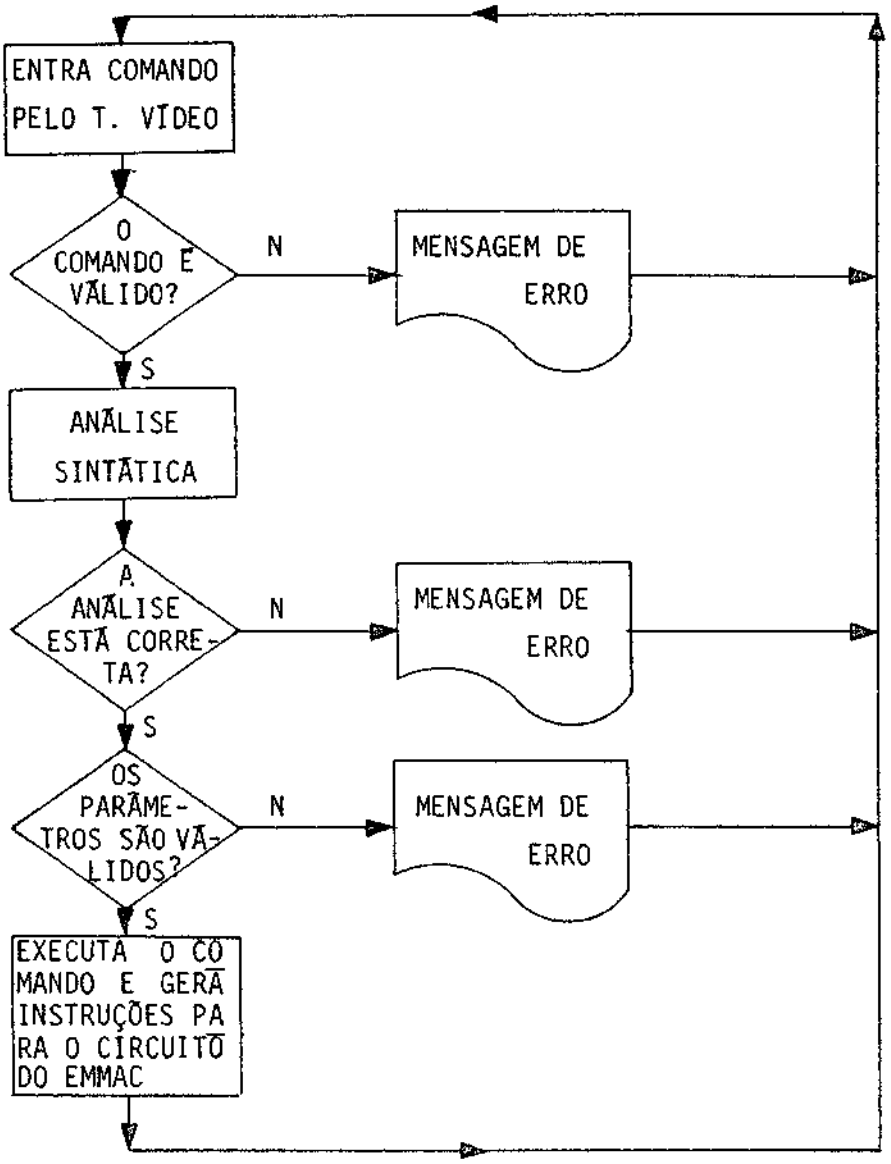

Fig. IV.1 - Diagrama de execução dos comandos da linguagem de utilização do EMMAC.

Para a identificação dos comandos, apenas as duas primeiras letras do seu nome são necessárias. Esta medida tem por finalidade simplificar a sua utilização e também economizar um pouco de memória.

Os comandos serão apresentados com uma descrição da sua sintaxe, utilizando-se a metalinguagem BNF ("Backus-Naur Form"), seguida de exemplos e de uma descrição da sua utilização.

Na descrição da sintaxe, os seguintes símbolos não terminais são utilizados, sendo que as suas definições serão dadas a seguir:

<terminal de entrada> ::= T1|T2|T5|T7|T8
<terminal de saída> ::= T1|T2|T4|T6|T7|T8

onde a seguinte equivalência existe:

T1 - Terminal de vídeo
T2 - Unidade de disco
T4 - Unidade perfuradora de fita de papel
T5 - Unidade leitora de fita de papel
T6 - Unidade impressora de linha
T7 - Unidade de fita magnética
T8 - Unidade de fita mangética

<nome do arquivo> ::= {identificador de até 5 caracteres,letras ou números sendo que o primeiro é uma letra}

<identificador de memória> ::= {número decimal entre 1 e 20, incluindo os extremos}

A definição destes não terminais, antecedendo a definição da sintaxe da linguagem, se deve ao fato deles fazerem parte da definição de quase todos os comandos.

Nos exemplos de utilização, são listados do lado direito, os mnemônicos das instruções geradas para o circuito do emulador de memórias de microcontrole. Estes mnemônicos estão na Tabela III.5.

4.1 - COMANDO ABORTE

4.1.1 - SINTAXE DO COMANDO

<comando ABORTE> ::= *ABORTE

Exemplos:

*ABORTE

*AB

4.1.2 - UTILIZAÇÃO DO COMANDO

Este comando serve para terminar outro comando, cuja execução ainda não terminou, por exemplo, para terminar um comando EXECUTE.

Após o comando ABORTE, o sistema emulador retorna ao estado inicial de aceitar comandos,através do terminal de vídeo do HP2116B.

4.2 - COMANDO ARMAZENE

<comando ARMAZENE> ::= *ARMAZENE,<especificação do arquivo>;|
*ARMAZENE,<especificação do arquivo>, (<endereço inferior>-<endereço superior>);|
*ARMAZENE,<especificação do arquivo>,<conjunto de memórias>;|
*ARMAZENE,<especificação do arquivo>,<conjunto de memórias>,(<endereço inferior>-<endereço superior>);|

<especificação do arquivo> ::= <terminal de saída>|
<terminal de saída>'<nome do arquivo>'

<conjunto de memórias> ::= <identificador de memória>|
<identificador de memória>,<conjunto de memó
rias>

<endereço inferior> ::= {número decimal de 0 a 511}

<endereço superior> ::= {número decimal de 0 a 511, maior do que o símbo
lo não terminal <endereço inferior>}

Exemplos:

```
*ARMAZENE,T2'KKKU',1,20,(0-3);
MEMORIA  1
                                                    CARSE   1
                                                    CACES   0
                                                    LERI    0
POS    0 DADO    1
                                                    LERI    1
POS    1 DADO    1
                                                    LERI    1
POS    2 DADO    1
                                                    LERI    1
POS    3 DADO    1
MEMORIA 20
                                                    CARSE  20
                                                    CACES   0
                                                    LERI    1
POS    0 DADO   20
                                                    LERI   20
POS    1 DADO   20
                                                    LERI   20
POS    2 DADO   20
                                                    LERI   20
POS    3 DADO   20

*ARMAZENE,T5,1,24,(1234-345);
          ^
 TERMINAL INVALIDO
            ^
 IDENT. DE MEMORIA INVALIDO
                 ^
 NUMERO INVALIDO
 COMANDO NAO EXECUTADO
```

4.2.2 - UTILIZAÇÃO DO COMANDO

Este comando serve para salvar, em um dado arquivo, o conteúdo das memórias do emulador, quando a operação tiver que ser interrompida. Ele salva apenas as memórias especificadas pelos identificadores de memória constantes do comando. Caso as memórias a serem salvas não sejam especificadas, por exclusão, todas as memórias serão salvas.

As posições das memórias a serem salvas são definidas pelo endereço inferior e superior constante do comando. Caso estes endereços não sejam definidos, todas as posições de memória serão salvas.

Os dados armazenados por este comando podem ser recuperados pelo comando COPIE, definido a seguir.

O símbolo não terminal *<terminal de saída>* deverá ser tal que permita entrada e saída de dados, ou seja, se for utilizado o terminal de vídeo ou a unidade impressora de linha, os dados não poderão ser recuperados pelo comando COPIE.

Quando o terminal de saída especificado for o disco, o usuário deverá dar o nome de um arquivo, anteriormente criado, onde os dados serão armazenados.

O formato dos dados é pré-definido, não tendo o usuário do sistema emulador qualquer controle sonre ele.

Caso o *<terminal de saída>* especificado seja o disco, o usuário deverá antes de emitir o comando, criar um arquivo no disco, com a diretiva do sistema operacional mostrada abaixo (Hewlett-Packard, 1973):

:STORE, B,<nome do arquivo>, 40

Caso o <*terminal de saída*> especificado seja o terminal de fita magnética, o comando perguntará ao usuário se o arquivo já existe ou não, ou seja, se é ou não a primeira vez que o comando ARMAZENE é dado naquela posição ou naquela fita. O usuário deverá responder SIM,caso seja verdade, e NÃO, caso contrário. No caso que a resposta seja NÃO, o comando destruirá o que estiver armazenado na fita, criando um arquivo para o comando. Caso a resposta seja SIM, o comando só modificará as memórias especificadas no comando, deixando as outras intactas.

4.3 - COMANDO COPIE

4.3.1 - SINTAXE DO COMANDO

<comando COPIE> ::= *COPIE,<especificação do arquivo>;|
*COPIE,<especificação do arquivo>, (<endereço inferior>--<endereço superior>);|
*COPIE,<especificação do arquivo>,<conjunto de memórias>,(<endereço inferior>--<endereço superior>);

<especificação do arquivo>::=<terminal de entrada>|
<terminal de entrada>'<nome do arquivo>'

<conjunto de memórias>::=<identificador de memória>|
<identificador de memória>,<conjunto de memórias>

<endereço inferior>::={número decimal de 0 a 511}

<endereço superior>::={número decimal de 0 a 511, maior do que o símbolo não terminal <endereço inferior>}

Exemplos:

```
*COPIE,T2'KKKU',1,20,(0-3);
MEMORIA  1
                                                    CARSE  1
                                                    CACES  0
                                                    ESCR   1
                                                    LERI 257
>>  ERRO DE ESCRITA NA MEMORIA  1  POS=  0  DADO=  1  DADO CARREGADO=257


*TE,1;
 ^
 COMANDO INVALIDO


*COPIE,T2'KKKU',1,20,(0-3);
 ^
 COMANDO INVALIDO


*TE,1;
 ^
 COMANDO INVALIDO


*SIGA
POS   0 DADO 257
                                                    ESCR   1
                                                    LERI 257
>>  ERRO DE ESCRITA NA MEMORIA  1  POS=  1  DADO=  1  DADO CARREGADO=257


*ABORTE
ABORTADO


*COPIE,T10,1,24,(90),4;
       ^
 TERMINAL INVALIDO
            ^
 IDENT. DE MEMORIA INVALIDO
                  ^
 ; ESPERADO
 COMANDO NAO EXECUTADO
```

4.3.2 - UTILIZAÇÃO DO COMANDO

Este comando serve para recuperar os dados armazenados em um dado arquivo, por um comando ARMAZENE anterior, quando a operação do sistema emulador tiver que ser reiniciada.

Ela deverá ser compatível com o comando que guardou os dados no terminal de onde eles serão recuperados, ou seja, deverá ser recuperada a mesma quantidade de memórias do mesmo arquivo, especificado pelo comando ARMAZENE.

Os dados escritos nas memórias do sistema emulador serão lidos automaticamente e comparados com os dados que deveriam ser escritos. Caso haja algum erro de escrita na memória do sistema emulador, uma mensagem de erro será dada no terminal de vídeo, indicando a posição, o dado que deveria ter sido escrito, além daquele dado que foi efetivamente escrito, bem como o número da memória onde ocorreu o erro.

No caso de um erro de escrita em uma memória do sistema emulador, a execução do comando COPIE será interrompida, podendo ser abortada por um comando ABORTE ou continuada por um comando SIGA. Em caso de erro, somente estes dois comandos serão válidos.

Com este procedimento, fica garantido que os dados corretos serão armazenados nas memórias do sistema emulador.

4.4 - COMANDO DIAGNOSE

4.4.1 - SINTAXE DO COMANDO

<comando DIAGNOSE>::= *DIAGNOSE,<nº do comando>,<especificação do arquivo de entrada>,<terminal de saída>,<nº de estados>;

<especificação do arquivo de entrada>::=<terminal de entrada>|
<terminal de entrada>
'<nome do arquivo>'

<nº do comando>::= {número decimal inteiro e positivo}

<nº de estados>::= {número decimal positivo de 1 a 256}

Exemplos:

```
*DIAGNOSE,1,T2'KKKU',T6,250;
INCMD
INCMD
INCMD
INCMD
INCMD
INCMD
INCMD
PAGINA  1      COMANDO DIAGNOSE

               PONTAS DE PROVA

    00000000 11111111 22222222 33333333
EST 01234567 01234567 01234567 01234567
LERMO  0
LERMO  1
LERMO  2
LERMO  3
INCMD
  1 00000001 00000001 00000010 00000011 00000000 00000001 00000010 00000011


*SIGA
LERMO  0
LERMO  1
LERMO  2
LERMO  3
INCMD
  2 00000001 00000001 00000010 00000011 00000000 00000001 00000010 00000011


*ABORTE
ABORTADO


*DIAGNOSE,0,T6,T4,345,RE;
         ^
 NUMERO INVALIDO
              ^
 TERMINAL INVALIDO
                  ^
 TERMINAL INVALIDO
                     ^
 NUMERO INVALIDO
                        ^
 ; ESPERADO
 COMANDO NAO EXECUTADO
```

4.4.2 - UTILIZAÇÃO DO COMANDO

Este comando serve para ajudar na localização de erros, no circuito do sistema em teste.

O primeiro passo deste comando é recuperar os dados armazenados nas memórias de monitoramento, dados estes que são o resultado do armazenamento dos estados das pontas de teste, quando o mesmo micro programa que gerou os estados armazenados no *<terminal de entrada>* foi executado.

O segundo passo deste comando é comparar os estados armazenados nas memórias de monitoramento, com os estados armazenados no terminal ou arquivo especificado pelo símbolo não terminal *<especificação do arquivo de entrada>*. Se houver algum erro na comparação, uma mensagem de erro no terminal de saída é impressa, identificando a ponta ou pontas de testes (até 32 pontas de testes), onde ocorreram as discrepâncias, o que ocasiona a interrupção da execução deste comando. Se o usuário quiser continuar a comparação dos estados restantes, um comando SIGA deverá ser emitido para a execução. Caso contrário, um comando ABORTE termina a execução do comando DIAGNOSE.

O arquivo, especificado como terminal de entrada, deverá ter sido gerado por um comando MONITOR, armazenando os estados das mesmas pontas de testes, para o mesmo microprograma carregado na memória de controle, quando o circuito não apresentava falhas.

O programa da memória de microcontrole que gerou os estados dos terminais de monitoramento, armazenados pelo comando MONITOR, deverá ter sido guardado por um comando ARMAZENE e carregado nas memórias de microcontrole, antes da execução do comando DIAGNOSE, por um comando COPIE.

4.5 - COMANDO ESCREVA

4.5.1 - SINTAXE DO COMANDO

<comando ESCREVA>::= *ESCREVA,<lista de parâmetros>;

<lista de parâmetros>::=<especificação da memória>,<especificação do arquivo de saída>|
<especificação da memória>,<especificação do arquivo de saída>,<formato>

<especificação da memória> ::=<identificador de memória>|
<identificador de memória>(<endereço inferior>)|
<identificador de memória>(<endereço inferior>-<endereço superior>)

<endereço inferior>::={nº decimal de 0 a 511}

<endereço superior>::={nº decimal de 0 a 511, maior do que o endereço inferior}

<especificação do arquivo de saída> ::=<terminal de saída>|
<terminal de saída>'<nome do arquivo>'

<formato>::= HEX|OCT|BIN|DEC

Exemplos:

```
*ESCREVA,5(0-5),T6,BIN;
                                                    CARSE   5
                                                    CACES    0
                                                    LERI    0
                                                    LERI    5
                                                    LERI    5
                                                    LERI    5
                                                    LERI    5
                                                    LERI    5
MEMORIA  5
```

```
POS    0 DADO 00000101
POS    1 DADO 00000101
POS    2 DADO 00000101
POS    3 DADO 00000101
POS    4 DADO 00000101
POS    5 DADO 00000101
```

```
*ESCREVA,25(345-90),T5,PIN;
         ^
 IDENT. DE MEMORIA INVALIDO
                ^
 NUMERO INVALIDO
                    ^
 TERMINAL INVALIDO
                        ^
 FORMATO ESPERADO
 COMANDO NAO EXECUTADO
```

4.5.2 - UTILIZAÇÃO DO COMANDO

Este comando serve para listar o conteúdo de uma dada memória de microcontrole, do sistema emulador, em um dado terminal de saída. Ele poderá ser utilizado, por exemplo, quando o microprograma já estiver depurado, podendo,então,o conteúdo de cada memória ser listado em fita de papel, para poder alimentar um programador de memórias do tipo PROM.

A memória a ser listada é definida pelo símbolo não terminal *<especificação da memória>*, podendo ser totalmente listada, ou apenas uma posição, ou um determinado número de posições em sequência.

Quando apenas o *<identificador de memória>* for encontrado, todas as posições da memória serão listadas. Se apenas o *<endereço inferior>* entre parênteses, após o *<identificador de memória>*, for especificado, somente o conteúdo daquela posição será listado, e se os dois

endereços forem especificados, os conteúdos das posições compreendidas entre o *<endereço inferior>* e o *<endereço superior>* serão listados, incluindo os endereços extremos.

O símbolo não terminal *<especificação do arquivo de saída>*, como o próprio nome está dizendo, identifica o arquivo e/ou o terminal onde os conteúdos das memórias devem ser listados.

Os dados serão listados também no terminal de vídeo, independente do terminal especificado como saída de dados.

O formato dos dados de saída é especificado pelo símbolo não terminal *<formato>*, podendo ser hexadecimal, octal, binário ou decimal. Caso não seja previamente definido, por exclusão, o formato dos dados a serem listados será binário.

4.6 - COMANDO EXECUTE

4.6.1 - SINTAXE DO COMANDO

<comando EXECUTE> ::= *EXECUTE

Exemplo:

```
*EXECUTE
                                                    EXEC
```

4.6.2 - UTILIZAÇÃO DO COMANDO

Uma vez que as condições iniciais do sistema emulador já tiverem sido especificadas pelos outros comandos, o comando EXECUTE, apenas, liberará o relógio do sistema em teste, no modo especificado pelo comando RELÓGIO.

Caso o comando RELÓGIO não tenha ainda sido dado, o comando EXECUTE implicará no modo de relógio passo-a-passo.

Este comando é interrompido por um comando PARE ou terminado por um comando ABORTE, ou por uma condição de finalização do relógio.

4.7 - COMANDO FIM

4.7.1 - SINTAXE DO COMANDO

<comando FIM> ::= *FIM

Exemplo:

```
*FIM
```

4.7.2 - UTILIZAÇÃO DO COMANDO

Este comando serve para terminar o monitoramento, retornando o comando do sistema, pelo minicomputador, para o seu sistema operacional.

4.8 - COMANDO FREQUENCIA

4.8.1 - SINTAXE DO COMANDO

<comando FREQUENCIA> ::= *FREQUENCIA,<fator de divisão>;

<fator de divisão> ::= (número decimal de 1 a 16)

Exemplos:

```
*FREQUENCIA,10;
```

```
*FREQUENCIA,45;
              ^
 NUMERO INVALIDO
 COMANDO NAO EXECUTADO
```

4.8.2 - UTILIZAÇÃO DO COMANDO

Este comando é utilizado para definir a frequência submúltipla, da frequência do relógio, para armazenamento e monitoramento, em tempo real, dos estados das pontas de teste sob observação. Por exemplo, se o <*fator de divisão*> for 2, significa que, de dois em dois pulsos do relógio, os estados das pontas de testes serão armazenados.

A utilidade deste comando fica evidenciada pelos comandos DIAGNOSE E MONITOR.

Se este comando não for utilizado, o sistema emulador assumirá que o valor do fator de devisão é 1.

4.9 - COMANDO INICIE

4.9.1 - SINTAXE DO COMANDO

```
<comando INICIE> ::= *INICIE,<iniciação>;|
                     *INICIE,<iniciação>,(<nº de pulsos>);

<iniciação> ::=<condições iniciais>|
               (<endereço>)

<condições iniciais> ::=<identificador de memória>=<nº decimal>|
                        <identificador de memória>=<nº decimal>, <condi
                         ções iniciais>
```

<nº decimal> ::= {número decimal de 0 a 511}

<endereço> ::= {número decimal de 0 a 511}

<nº de pulsos> ::= {número decimal positivo de 0 até 511}

Exemplos:

```
*INICIE,5=255,(2);
                                                    CACES   0
                                                    CARSE  5
                                                    LER $$$
                                                    ESCR 255
                                                    LER   0
>>  ERRO DE ESCRITA NA MEMORIA  5  POS=  0  DADO=255  DADO CARREGADO=  0

*SIGA
                                                    CARS  0
                                                    EXECC
                                                    EXECC
                                                    CARSE  5
                                                    ESCR $$$
                                                    LER   0
>>  ERRO DE ESCRITA NA MEMORIA  5  POS=  0  DADO=$$$  DADO CARREGADO=  0

*SIGA

*INICIE,(500),(1);
                                                    CACES   0
                                                    CACES 500
                                                    CARS  0
                                                    EXECC

*INICIE,1=366,2=3777,3=456,20=431,(-1);
          ^
 NUMERO INVALIDO

 NUMERO INVALIDO

 NUMERO INVALIDO

 NUMERO INVALIDO

 NUMERO INVALIDO

 SEPARADOR INVALIDO

   ESPERADO
 COMANDO NAO EXECUTADO
```

4.9.2 - UTILIZAÇÃO DO COMANDO

Este comando é utilizado para forçar condições iniciais e executá-las no sistema em teste, em uma microinstrução ou um setor isolado de uma microinstrução. Por exemplo, uma condição que poderá ser forçada por este comando é a "inicialização" do endereço da memória de controle do sistema em teste.

No caso onde as condições iniciais devem ser diretamente fornecidas, este comando salva o conteúdo de uma determinada posição de memória (posição 0), carrega as condições iniciais definidas no comando e executa o número de pulsos do relógio, do sistema em teste, especificado pelo não terminal *<nº de pulsos>*. Em seguida ele restaura o conteúdo anterior daquela posição de memória.

A condição do relógio, após executado o comando, é o passo-a-passo.

No caso em que apenas o endereço da memória de microcontrole é fornecido, ela é endereçada e o número de pulsos de relógio, do sistema em teste, especificado pelo símbolo não terminal *<nº de pulsos>* é liberado. Por exclusão, se o número de pulsos não for especificado, será considerado o valor 1.

4.10 - COMANDO LEIA

4.10.1 - SINTAXE DO COMANDO

<comando LEIA> ::= *LEIA,<lista de parâmetros>;

<lista de parâmetros> ::= <especificação da memória>,<especificação do arquivo de entrada>|
<especificação da memória>,<especificação do arquivo de entrada>,<formato>

<especificação da memória> ::=<identificador de memória>|
<identificador de memória>(<endereço inferior>)|
<identificador de memória>(<endereço inferior>-<endereço superior>)

<endereço inferior> ::= {nº decimal de 0 a 511}

<endereço superior> ::= {nº decimal de 0 a 511, menor do que o endereço inferior}

<especificação do arquivo de entrada> ::= <terminal de entrada>|
<terminal de entrada> ,'<nome do arquivo>'

<formato> ::= HEX|OCT|BIN|DEC

Exemplos:

```
*LEIA,1(3-5),T2'KKKU',BIN;
                                                    CARSE   1
                                                    CACES    3
                                                    ESCR    1
                                                    LERI    0
>>  ERRO DE ESCRITA NA MEMORIA   1   POS=   3   DADO=   1   DADO CARREGADO=   0

*SIGA
POS    3 DADO    0
                                                    ESCR    0
                                                    LERI    0
POS    4 DADO    0
                                                    ESCR    1
                                                    LERI    0
>>  ERRO DE ESCRITA NA MEMORIA   1   POS=   5   DADO=   1   DADO CARREGADO=   0

*SIGA
POS    5 DADO    0

*LEIA,T1,1(0-512),FEC;
     ^
 IDENT. DE MEMORIA INVALIDO
       ^
 TERMINAL INVALIDO
             ^
 FORMATO ESPERADO
 COMANDO NAO EXECUTADO
```

4.10.2 - UTILIZAÇÃO DO COMANDO

Este comando serve para carregar uma dada memória, do sistema emulador, através de um dado terminal de entrada. Ele poderá ser utilizado, por exemplo, para carregar os dados gerados por um "assembler" para microprogramação, fornecidos sejam em fita de papel perfurada, fita magnética, etc., nas memórias de microcontrole do sistema emulador.

A memória a ser carregada é definida pelo não terminal *<especificação da memória>*, podendo ser total ou parcialmente carregada.

Quando apenas o identificador de memória for encontrado, toda a memória será carregada. Se apenas o endereço inferior for especificado entre parênteses após o *<identificador de memória>*, somente aquela posição será carregada e se os dois endereços forem especificados, as posições compreendidas entre o *<endereço inferior>* e o *<endereço superior>* serão carregadas, incluindo os extremos.

O não terminal *<especificação do arquivo de entrada>* especifica o terminal e/ou arquivo que contêm os dados a serem carregados nas memórias de microcontrole do sistema emulador.

O formato dos dados existentes no terminal ou arquivo a ser lido, é especificado pelo símbolo não terminal *<formato>*, podendo ser haxadecimal, octal, binário ou decimal. Se indefinido, o formato considerado será o binário.

A toda escrita na memória do sistema emulador, será feita uma leitura após esta escrita, seguida de uma comparação desta leitura com o dado enviado para a escrita. Se houver alguma discrepância, o comando imprime no terminal de vídeo uma mensagem de erro indicando a memória, a posição onde ocorreu o erro, bem como o dado que deveria ter sido carregado e o dado que foi efetivamente carregado.

Caso ocorra algum erro, o carregamento dos dados restan

tes é interrompido, podendo ser continuado por um comando SIGA ou terminado por um comando ABORTE. Este procedimento permite detectar os erros de escrita que poderiam ocorrer nas memórias do sistema emulador.

4.11 - COMANDO LISTE

4.11.1 - SINTAXE DO COMANDO

```
<comando LISTE> ::= *LISTE,(<endereço inferior>);|
                    *LISTE,(<endereço inferior>-<endereço superior>);|
                    *LISTE,<parâmetros>;|
                    *LISTE,<parâmetros>,(<endereço inferior>);|
                    *LISTE,<parâmetros>,(<endereço inferior> - <endereço
                    superior>);|
                    *LISTE;

<parâmetros>::=<especificação das memórias>|
               <especificação das memórias>,<terminal de saída>|
               <especificação das memórias>,<formato>|
               <especificação das memórias>,<terminal de saída>, <forma
               to>|
               <terminal de saída>|
               <terminal de saída>,<formato>|
               <formato>

<especificação das memórias> ::= <identificador de memória>|
                                 <identificador de memória>,<especifica
                                 ção das memórias>

<formato> ::= HEX|OCT|BIN|DEC
```

Exemplos:

```
*LISTE,1,2,3,T6,BIN,(0-3);
```

```
CACES    0
CARSE   1
LER $$$
CARSE   2
LER $$$
CARSE   3
LER $$$
```

```
POS          1          2          3
  0 11111111 11111111 11111111
```

```
CACES    1
CARSE   1
LER 255
CARSE   2
LER 255
CARSE   3
LER 255
```

```
POS          1          2          3
  1 11111111 11111111 11111111
```

```
CACES    2
CARSE   1
LER 255
CARSE   2
LER 255
CARSE   3
LER 255
```

```
POS          1          2          3
  2 11111111 11111111 11111111
```

```
CACES    3
CARSE   1
LER 255
CARSE   2
LER 255
CARSE   3
LER 255
```

```
POS          1          2          3
  3 11111111 11111111 11111111
```

```
*LISTE,1,2,3,4,5,6,7,8,9,10,T6,HEX,(0);
```

```
CACES    0
CARSE   1
LER 255
CARSE   2
LER 255
CARSE   3
LER 255
CARSE   4
LER $$$
CARSE   5
LER $$$
CARSE   6
LER $$$
```

```
                                                    CARSE  7
                                                    LER $$$
                                                    CARSE  8
                                                    LER $$$
                                                    CARSE  9
                                                    LER $$$
                                                    CARSE 10
                                                    LER $$$
POS      1  2  3  4  5  6  7  8  9 10
  0    FF FF FF FF FF FF FF FF FF FF
*LISTE,1,23,1T,(345-511),DEC;
        ^
 IDENT. DE MEMORIA INVALIDO
           ^
 NUMERO DO TERMINAL ESPERADO
              ^
 FORMATO OU ( ESPERADO
                 ^
 FORMATO OU ( ESPERADO
 COMANDO NAO EXECUTADO
```

4.11.2 - UTILIZAÇÃO DO COMANDO

Este comando serve, apenas, para se obter uma listagem do conteúdo de uma ou várias memórias do sistema emulador.

Quando as memórias a serem listadas não forem especificadas pelo não terminal *<especificação das memórias>*, todas as memórias serão listadas a partir das posições referenciadas pelos endereços, inferior e superior, especificados. Se apenas o endereço inferior for especificado, somente aquela posição será listada e se nenhum endereço for especificado, toda a memória será listada.

O terminal de saída só poderá ser ou o terminal de vídeo ou a impressora de linha. Se outro terminal não autorizado for especificado, será interpretado como sendo o terminal de vídeo.

O formato pode ser hexadecimal, octal, binário ou decimal. Por exclusão ele será haxadecimal.

4.12 - COMANDO MONITOR

4.12.1 - SINTAXE DO COMANDO

<comando MONITOR> ::= *MONITOR,<nº do comando>,<especificação do arquivo>,<nº de estados>;|
*MONITOR,<nº do comando>,<especificação do arquivo>;|
*MONITOR,<nº do comando>,<nº de estados>;|
*MONITOR,<nº do comando>;

<nº do comando> ::= {número decimal positivo}

<especificação do arquivo> ::= <terminal de saída>|
<terminal de saída>'<nome do arquivo>'

<nº de estados> ::= {número decimal, inteiro positivo de 1 a 256}

Exemplos:

```
*MONITOR,1,T6,5;
INCMO
INCMO
  :     } 252
INCMO
LERMO   0
LERMO   1
LERMO   2
LERMO   3
INCMO
LERMO   0
LERMO   1
LERMO   2
LERMO   3
INCMO
LERMO   0
LERMO   1
LERMO   2
LERMO   3
INCMO
LERMO   0
LERMO   1
LERMO   2
LERMO   3
INCMO
```

```
LERMO  0
LERMO  1
LERMO  2
LERMO  3
INCMO

PAGINA  1      COMANDO MONITOR   1

               PONTAS DE PROVA

     00000000  11111111  22222222  33333333
EST  01234567  01234567  01234567  01234567
  1  00000000  00000001  00000010  00000011
  2  00000000  00000001  00000010  00000011
  3  00000000  00000001  00000010  00000011
  4  00000000  00000001  00000010  00000011
  5  00000000  00000001  00000010  00000011

*MONITOR,100,T10,511;
             ^
 TERMINAL INVALIDO
                 ^
 NUMERO INVALIDO
 COMANDO NAO EXECUTADO
```

4.12.2 - UTILIZAÇÃO DO COMANDO

Este comando serve para monitorar, em tempo real,até 256 estados de até 32 pontas de testes, a serem escolhidos no sistema em teste.

A frequência em que estes pontos são monitorados é determinada pelo comando FREQUENCIA.

Os estados destas pontas de teste são armazenados em um terminal de saída, especificado pelo símbolo não terminal *<terminal de saída>*, sendo também sempre mostrados no terminal de vídeo do minicomputador. Se o terminal de saída não for especificado, estes estados são apenas escritos no terminal de vídeo. Caso o terminal de saída seja o

disco, deverá ser dado o nome do arquivo, criado pelo usuário, onde os dados serão armazenados.

O símbolo não terminal *<nº de estados>* indica quantos dos 256 últimos estados serão armazenados no terminal de saída. Se indefinido, este não terminal será considerado igual a 256.

4.13 - COMANDO MUDE

4.13.1 - SINTAXE DO COMANDO

<comando MUDE> ::= *MUDE,<especificação da fonte>,<especificação do destino>;

<especificação da fonte> ::= <identificador de memória>|
<identificador de memória>(<endereço inferior>)|
<identificador de memória>(<endereço inferior>-<endereço superior>)

<especificação do destino> ::=<especificação da fonte>

<endereço inferior> ::= {número decimal de 0 a 511}

<endereço superior> ::= {número decimal entre 0 a 511, maior do que o endereço inferior}

Exemplos:

```
UDE,1(0-1),5(510-511);
                                                CARSE  1
                                                CACES   1
                                                LER   1
                                                CARSE  5
                                                CACES 511
                                                ESCR   1
                                                LER 100
  ERRO DE ESCRITA NA MEMORIA  5  POS=511  DADO=  1  DADO CARREGADO=511
```

```
*SIGA
                                                    CARSE   1
                                                    CACES    0
                                                    LER    1
                                                    CARSE   5
                                                    CACES 510
                                                    ESCR    1
                                                    LER 511
>>  ERRO DE ESCRITA NA MEMORIA   5   POS=510   DADO=   1   DADO CARREGADO=511

*SIGA

*NUDE,24(10-8),5(0);

 IDENT  DE MEMORIA INVALIDO

 NUMERO INVALIDO

 NUMERO INVALIDO
 COMANDO NAO EXECUTADO
```

4.13.2 - UTILIZAÇÃO DO COMANDO

Este comando serve para mover o conteúdo de uma região da memória de controle para outra região, da mesma ou de outra memória de controle.

A especificação da fonte deverá ser compatível, em número de posições, com a especificação do destino.

Como deve acontecer com o comando LEIA, os dados deslocados são lidos e comparados com os dados de origem e, caso ocorra algum erro, uma mensagem é enviada para o terminal de vídeo.

4.14 - COMANDO PARE

4.14.1 - SINTAXE DO COMANDO

<comando PARE> ::= *PARE

Exemplo:

```
*PARE                                                    PARE
```

4.12.2 - UTILIZAÇÃO DO COMANDO

Este comando, serve para interromper uma determinada instrução do sistema emulador, que esteja sendo executada.

Quando uma instrução tiver sido interrompida por um comando PARE, ela poderá ser continuada por um comando SIGA, ou terminada por um comando ABORTE.

4.15 - COMANDO PFITA

4.15.1 - SINTAXE DO COMANDO

<comando PFITA> ::= *PFITA,<fita magnética>,<nº de "records">;

<nº de "records">::= {número inteiro positivo ou negativo}

<fita magnética> ::= T7|T8

Exemplos:

```
*PFITA,T7,20;

*PFITA,T8,4;
```

4.15.2 - UTILIZAÇÃO DO COMANDO

Este comando tem por finalidade melhorar a utilização de fitas magnéticas, fazendo com que o usuário possa ter mais de um arquivo em uma mesma fita magnética.

Cada arquivo criado pelo sistema EMMAC é suficiente ou para armazenar os conteúdos dos módulos de emulação do EMMAC ou para armazenar os estados do módulo de monitoramento, gerados pelo comando MONITOR. No primeiro caso, o tamanho do arquivo é de 20 "records" e no segundo caso o tamanho é de 4 "records".

Se o *<nº de "records">* especificado no comando for positivo, a fita será posicionada no primeiro "record" após este número, e se o *<nº de "records">* for negativo, a fita voltará este número especificado de "records", posicionando-se no início do "record" anterior.

Então, se o usuário tiver uma fita, onde o primeiro arquivo é um arquivo contendo os conteúdos dos módulos de emulação, e ele quiser criar um arquivo, na mesma fita, com um comando MONITOR, ele pode dar um comando PFITA com o número de "records" 20 e depois emitir o comando MONITOR.

4.16 - COMANDO RELOGIO

4.16.1 - SINTAXE DO COMANDO

<comando RELOGIO> ::= *RELOGIO,<opções>;

<opções> ::= PAP|LIVRE|ATE<limite>|COND<endereço>

<endereço> ::= {nº decimal de 0 a 511}

<limite> ::= {nº decimal inteiro e positivo até 511}

Exemplos:

```
*RELOGIO,   PAP;
                                                    CARS  0

*RELOGIO,   LIVRE;
                                                    CARS  1

*RE,  ATE 256;
                                                    CARC 256
                                                    CARS  3

*RELOGIO,   COND 221;
                                                    CARC 221
                                                    CARS  2

*RELOGIO, ATE 610;
           ^
 NUMERO INVALIDO
 COMANDO NAO EXECUTADO
```

4.16.2 - UTILIZAÇÃO DO COMANDO

Este comando é utilizado para determinar o modo do relógio, do sistema em teste, que será liberado quando um comando EXECUTE for dado.

As opções deste comando são as seguintes:

PAP - execução passo-a-passo;

LIVRE - execução livre; só será interrompida por um comando PARE ou por um comando ABORTE, ou por condições do sistema em teste;

ATÉ<limite> - execução livre durante o número de pulsos especificado pelo não terminal <limite>;

COND<endereço> - execução livre até que o endereço especificado seja atingido pelo sequenciador de microprograma do sistema em teste.

Se este comando não for emitido na sequência de monitoramento, por exclusão, será executado o modo de relógio passo-a-passo.

4.17 - COMANDO SIGA

4.17.1 - SINTAXE DO COMANDO

<comando SIGA> ::= *SIGA

Exemplo:

```
*SIGA
```

4.17.2 - UTILIZAÇÃO DO COMANDO

Este comando é utilizado para continuar a execução de um comando do sistema emulador que tiver sido interrompida, ou por um comando PARE, ou por uma condição de erro.

4.18 - COMANDO TERMINAL

4.18.1 - SINTAXE DO COMANDO

<comando TERMINAL> ::= *TERMINAL,<número>;

<número> ::= 1|6

Exemplo:

```
*TERMINAL, 6;
```

4.18.2 - UTILIZAÇÃO DO COMANDO

Este comando serve para definir o terminal, onde os comandos do sistema EMMAC serão listados.

Se o não terminal número, for igual a 1, o terminal de listagem será o terminal de vídeo do HP2116B. Se este parâmetro for 6, o terminal de listagem será a impressora de linha. Se este comando não for emitido, o terminal de listagem utilizado será o terminal de vídeo.

4.19 - COMANDO TROQUE

4.19.1 - SINTAXE DO COMANDO

<comando TROQUE> ::= *TROQUE,<especificação da fonte>,<especificação do destino>;

<especificação da fonte> ::= <identificador de memória>|
<identificador de memória>(<endereço inferior>)|
<identificador de memória>(<endereço inferior>-<endereço superior>)

<especificação do destino> ::= <especificação da fonte>

<endereço inferior> ::= {número decimal de 0 a 511}

<endereço superior> ::= {número decimal de 0 a 511 maior do que o endereço inferior}

Exemplos:

```
*TROQUE,1(0),5(511);
                                                    CARSE   1
                                                    CACES    0
                                                    LER    8
                                                    CARSE   5
                                                    CACES  511
                                                    LER    4
                                                    ESCR    8
                                                    LER    1
>>  ERRO DE ESCRITA NA MEMORIA   5   POS=511   DADO=   1   DADO CARREGADO=511 .

*SIGA
                                                    CARSE   1
                                                    CACES    0
                                                    ESCR    4
                                                    LER  511
>>  ERRO DE ESCRITA NA MEMORIA   1   POS=   0   DADO=   4   DADO CARREGADO=511

*SIGA
```

```
*TROQUE,1(7-2),23(10-4);
          ^
 NUMERO INVALIDO
               ^
 IDENT. DE MEMORIA INVALIDO
                    ^
 NUMERO INVALIDO
                         ^
 NUMERO INVALIDO
 COMANDO NAO EXECUTADO
```

4.19.2 - UTILIZAÇÃO DO COMANDO

Este comando é semelhante ao comando MUDE, anteriormente descrito, só que ao invés de apenas mover o conteúdo de uma região da memória de microcontrole para outra região, da mesma ou de outra memória, ele troca os conteúdos das regiões especificadas.

CAPÍTULO V

UTILIZAÇÃO DO EMULADOR DE MEMÓRIAS DE MICROCONTROLE

Como já foi definido no Capítulo II, o emulador de memórias auxiliado por computador pode ser utilizado em duas etapas, distintas, de projetos, envolvendo a técnica de microprogramação. Estas etapas são as seguintes:

1) utilização do sistema emulador na fase de testes e detecção de erros em microprogramas durante o projeto, e eventuais alterações no mesmo, após sua conclusão;

2) utilização do sistema emulador na fase de manutenção e diagnose do projeto já concluído.

Na primeira etapa, o sistema emulador deve funcionar de uma maneira bastante interativa com o projetista, o que deverá facilitar bastante a tarefa de carregar, alterar e tirar erros de microprogramas do sistema em teste.

Na segunda etapa, para que o sistema emulador seja eficiente, é necessário que o projetista organize e documente certos arquivos que poderão ficar armazenados em disco, fita de papel ou fita magnética.

As secções seguintes, deste capítulo, têm por finalidade dar uma idéia do procedimento necessário para a utilização do sistema emulador, nas duas etapas de projeto consideradas acima.

5.1 - UTILIZAÇÃO DO EMULADOR DE MEMÓRIAS DE MICROCONTROLE NA ETAPA DE TESTES, DETECÇÃO DE ERROS E ALTERAÇÕES EM MICROPROGRAMAS

Nesta etapa de utilização do sistema emulador, o procedimento a ser seguido pode ser dividido nos seguintes passos:

1) o projetista deverá de alguma forma, gerar os microprogramas, implementando os algorítmos necessários para o seu sistema,que serão carregados nas memórias de controle do referido sistema. Esta geração poderá ser feita manualmente ou através de uma linguagem de geração de microprogramas, que terá como saída um arquivo contendo, em um dado formato, os microprogramas a serem carregados nas memórias de controle;

2) deverá ser feita, então, a substituição das PROMs a serem emuladas no sistema em teste, pelos soquetes de tomada do sistema emulador. A saída do relógio, do sistema em teste, deverá ser conectada na entrada do relógio do painel do sistema emulador, e a saída do relógio do painel deverá entrar onde foi feita a derivação do relógio, no sistema em teste, para então ser distribuído pelos seus circuitos;

3) utilizando-se de um comando LEIA, todas as memórias, a serem emuladas, deverão ser carregadas com os dados disponíveis no arquivo, obtido na fase 1;

4) um comando RELOGIO deverá ser dado, a fim de programar o funcionamento do relógio no modo desejado. É interessante que na primeira fase de testes, este modo seja o modo passo-a-passo e,então, depois de testado o microprograma, passe para um dos modos do relógio com ritmo livre, que testará o sistema em tempo real, sujeito, assim, a todas as suas condições ambientais (atrasos nas portas, carga nos circuitos, etc.);

5) caso seja necessário monitorar os 256 últimos estados de alguns pontos do sistema em teste, as pontas de prova do sistema emulador deverão ser conectadas aos pontos desejados, e um comando FREQUENCIA deverá ser dado para especificar com que frequência estes dados serão armazenados. É interessante que no início da depuração dos microprogramas, o barramento de endereços da memória de controle seja monitorado por estas pontas de provas,

a cada pulso do relógio, tornado possível seguir precisamente a ordem de execução das microinstruções, ficando bastante fácil corrigir os erros dos microprogramas, por exemplo na execução dos desvios;

6) um comando INICIE deverá ser, então,dado para iniciar algumas condições no sistema em teste como, por exemplo, o endereço inicial do sequenciador de microinstruções da unidade de controle;

7) com todas as fases anteriores executadas, um comando EXECUTE deverá ser dado para que os microprogramas sejam executados. Se o modo do relógio for o passo-a-passo, um comando EXECUTE deverá ser dado para cada passo de execução;

8) caso seja necessário interromper a execução do microprograma, um comando PARE poderá ser dado, se o modo de relógio não for o passo-a-passo. Após a execução de um comando PARE, o programador, através dos comandos LEIA, ESCREVA, LISTE, MUDE, TROQUE e MONITOR, terá todas as facilidades para editar o seu microprograma, até que este possa ser executado corretamente;

9) caso o programador deseje interromper a utilização do sistema emulador, um comando ARMAZENE poderá ser dado, salvando o conteúdo de todas as memórias do EMMAC que estiverem sendo utilizadas em um dado arquivo, definido pelo programador. Este arquivo poderá ser recuperado por um comando COPIE, quando o emulador voltar a ser utilizado. Os comandos ARMAZENE e COPIE,tornam possível a utilização do sistema emulador na depuração de vários microprogramas, por vários usuários, ficando cada um armazenado em um arquivo próprio;

10) quando o microprograma estiver completamente depurado,utilizando-se de um comando ESCREVA, o projetista poderá obter uma listagem em fita de papel, do conteúdo de cada memória de microcon

trole. De posse desta fita e de um programador de memórias PROM que a aceita para leitura, o projetista poderá programar ("queimar") as PROMs a serem utilizadas no projeto final.

Seguindo-se estes 10 passos, a depuração e a alteração de microprogramas torna-se rápida e facilitada, o que deverá poupar bastante tempo ao projetista e propiciar um alto nível de determinismo na programação final das memórias PROMs.

5.2 - UTILIZAÇÃO DO EMULADOR DE MEMÓRIAS DE MICROCONTROLE NA ETAPA DE MANUTENÇÃO E DIAGNOSE DO PROJETO CONCLUÍDO

O procedimento para a utilização do EMMAC, nesta etapa, pode ser sub-dividido em duas fases:

1) fase preparatória da diagnose;
2) fase de diagnose de rotina.

Os passos para a execução da fase preparatória da diagnose são os seguintes:

1) uma vez que o projetista tenha certeza que o sistema está funcionando corretamente, ele deverá programar rotinas de diagnose, em microprogramas, para cada parte funcional do seu circuito. Cada uma destas rotinas farão testes em vários pontos de cada uma destas partes funcionais. Estas rotinas ficarão armazenadas, em um arquivo, seja em disco, fita de papel ou magnética, através do comando ARMAZENE;

2) o próximo passo é carregar, por um comando COPIE, estas rotinas nas memórias de emulação do sistema emulador, fazendo-se, antes, a substituição das PROMs de controle, do sistema em teste, pelos soquetes de derivação do emulador de memórias de microcontrole;

3) o projetista deverá selecionar até 32 pontos de teste e colocar as pontas de prova do emulador nestes pontos, dando, em seguida, um comando FREQUENCIA que determinará a frequência de armazenamento dos estados destes pontos;

4) em seguida, ele deverá dar um comando RELOGIO e um comando EXECUTE, que armazenará os estados destes pontos de teste,selecionados para várias funções comandadas pela rotina de diagnose, que deverá estar armazenada na memória de controle;

5) através de um comando MONITOR, o projetista deverá guardar os estados destes pontos em um arquivo definido por ele;

6) volta-se então para o passo 3, até que um número de pontos de teste e estados correspondentes, considerados suficientes, tenham sido armazenados no arquivo;

7) o último passo deve consistir na documentação dos arquivos, onde todas as informações referentes aos pontos de teste observados, tais como números de estados, frequência de armazenamento, nomes de arquivo, unidades funcionais relacionadas, bem como a sequência das operações ou algoritmos, componham o arquivo formado.

Os passos para à execução da fase de diagnose de rotina deverão ser os seguintes:

1) o projetista deverá carregar, através de um comando COPIE, as rotinas de diagnose armazenadas em um certo arquivo, na fase preparatória, após ter feito a substituição das PROMS de controle pelos soquetes de derivação para o sistema emulador;

2) o próximo passo é selecionar os mesmos pontos de testes do passo número 3, da fase preparatória, dando o mesmo comando FREQUENCIA;

3) dando o mesmo comando RELOGIO do passo número 4, da fase preparatória, o projetista poderá dar,então, um comando DIAGNOSE, que executará as rotinas e comparará os novos estados com os estados anteriormente armazenados pelo comando MONITOR. Caso haja alguma diferença, a execução é interrompida com uma mensagem de erro, para que uma análise e correção do erro seja feita;

4) volta-se então para o passo número 2, até que todos os estados armazenados, na fase preparatória, sejam corretamente comparados.

Esta utilização do emulador de memórias de microcontrole, auxiliado por computador, requer um trabalho adicional de preparar as rotinas de diagnose e armazenar os estados dos pontos de testes, mas uma vez feita esta parte, com uma boa documentação, a manutenção do sistema, bem como a sua diagnose, tornam-se rotineiras e eficientes.

CAPÍTULO VI

CONCLUSÕES E SUGESTÕES PARA TRABALHOS FUTUROS

Este trabalho está sendo implementado como parte do projeto Unidade Central de Processamento Microprogramado (ASTROP), do Grupo de Sistemas Digitais e Analógicos do INPE.

Da sua implementação já foram realizadas as seguintes etapas:

1) implementação e testes do programa de utilização do emulador de memórias de microcontrole no HP2116B;

2) projeto a nível de placas e circuitos integrados, microprogramação e simulação de todas as unidades funcionais do EMMAC;

3) construção da caixa, painel traseiro e painel frontal, e testes do painel frontal do EMMAC;

4) início da construção da unidade de controle em "wire-wrap".

Espera-se que, o circuito esteja totalmente montado e testado, em agosto de 1979.

As sugestões para trabalhos futuros relacionados a este são:

1) implementação de uma linguagem para geração de microprogramas que gere uma saída compatível com o EMMAC. Esta linguagem já esta sendo feita;

2) acoplamento de um programador de memórias PROM ao HP2116B, de tal forma que a programação dos microprogramas testados seja efetuada de maneira automática. Este acoplamento já está sendo feito.

É bom frizar que, embora este equipamento tenha sido desenvolvido visando um suporte para desenvolvimento de equipamentos microprogramados, ele poderá ser utilizado para emular memórias em outros projetos que não utilizem esta técnica. Um exemplo é a emulação das memórias principais de um sistema, utilizando um microprocessador cujos programas, uma vez testados, deverão ser armazenados em memórias ROM ou EPROM.

AGRADECIMENTOS

Ao Dr. Eduardo W. Bergamini pelo apoio e orientação no desenvolvimento deste trabalho.

À Banca Examinadora pelas sugestões e comentários durante a apresentação preliminar.

Ao José Benedito Soares Jr. (Nino) pela ajuda na obtenção das listagens de computador.

Ao pessoal do Departamento de Recursos Humanos pela ajuda nas questões burocráticas.

BIBLIOGRAFIA

ADVANCED MICRO DEVICES. *Microprogramming handbook and Am2900 emulation.* 2nd. ed. Sunnyvale, Cal., c1976a. 47p.

_____. *A microprogrammed 16-bit computer.* Sunnyvale, Cal., c1976b. seções

_____. *The Am2900 family data book.* Sunnyvale, Cal., 1976c. 172p.

AGRAWALA, A.K.; RAUSCHER, T.G. *Foundations of microprogramming: architeture, software and applications.* New York, N.Y., Academic Press, c1976. 416p. (ACM Monograph Series).

AMARAL, P.F.S.; MARTINS, R.C.O. *CDL - Manual de referência.* São José dos Campos, INPE, jun., 1978. (INPE-1277-NTI/105).

COMPUTER SCIENCE CORP. INFORMATION NETWORK DIC. *CSC AMDASM reference manual.* Los Angeles, Cal., Preliminary Edition, 1976.

HEWLETT-PACKARD. *Microcircuit interface: Computer interface.* Palo Alto, Cal., 1968. 15p.

_____. *DOS III disc operating system.* Cupertino, Cal., c1973.

INTEGRATED COMPUTER SYSTEMS. *187:Bit-slice microprocessor, PLA's and microprogramming: course notes from learning tree.* Culver City, Cal., c1976.

MICK, J.R.; SCHOPMEYER, R. *Advanced microprogramming design tools shorten Am2900 family design cycle.* S.N.T. 6p.

SIGNETICS MEMORY SYSTEMS. *Application notes: Design of microprogrammable systems.* Sunnyvale, Cal., 1970. 10p.

TEXAS INSTRUMENTS. *The semiconductor memory data book for design engineers.* Dallas, Tex., c1975. 271p.

APÊNDICE A

1. SIMULAÇÃO DO SISTEMA EMULADOR, UTILIZANDO A LINGUAGEM DE SIMULAÇÃO CDL, IMPLEMENTADA NO B6700

O objetivo deste apêndice é mostrar como foi feita a simulação do emulador de memórias de microcontrole, utilizando a linguagem de simulação CDL, e apresentar os resultados obtidos.

1.1 - LINGUAGEM DE SIMULAÇÃO

A linguagem CDL ("Computer Design Language")(Amaral, 1978) foi introduzida pelo seu autor, Yaoham Chu, em 1965). Ela foi proposta como uma linguagem de descrição de sistemas digitais, fornecendo um meio de simulação do seu funcionamento e também de documentação do sistema.

O nível de descrição do CDL é o de operação entre registros ("RT level"), sendo o sistema digital descrito como um conjunto de registros e operadores, que controlam e modificam o fluxo de dados entre os registros.

Sendo assim, uma simulação de um sistema na linguagem CDL é composta de duas fases:

1) Descrição de sistema em CDL - A descrição do sistema em CDL deve compreender a descrição da estrutura de armazenamento, da estrutura de controle, e do comportamento, no tempo, do sistema descrito;

2) Simulação do sistema através do simulador CDL - Entradas e programas de testes são executados no sistema descrito, e uma saída dos resultados obtidos é listada.

A linguagem CDL é do tipo "nonprocedural language", ou seja, a sequência de comandos, que especifica o funcionamento do sistema, não define implicitamente a ordem em que estes devem ser executados, como acontece normalmente em linguagens de programação. Em CDL, todos os comandos são rotulados e a ordem em que são especificados não corresponde, necessariamente, à ordem em que eles serão executados. Esta última é definida dinamicamente através dos valores lógicos (verdadeiro ou falso), atribuído à cada rótulo, a cada mudança de estado do sistema. Ações simultâneas são especificadas através de comandos com o mesmo rótulo.

O compilador e o simulador CDL implantado no INPE,corresponde à versão III do CDL implementado na Universidade de Karlsrhure. Esta versão, como mostra a Figura A.1, está constituída por dois programas: um programa tradutor e um programa simulador. O programa tradutor recebe a descrição do sistema, a ser simulado em cartões perfurados,e, após analizá-la, gera um conjunto de tabelas e uma forma intermediária chamada de "Polish String". O programa simulador, que é composto de 6 rotinas, como mostra a Figura A.1, sob o controle de um conjunto de comandos de controle de simulação e com o auxílio das tabelas montadas pelo programa tradutor, executa interpretativamente o cordão polonês ("Polish String"), que descreve o funcionamento do sistema digital em estudo. A execução desta forma intermediária é realizada através de uma rotina do programa simulador, utilizando um ciclo de controle chamado de ciclo de rótulo ("Label cycle"), que consiste na busca dos rótulos verdadeiros em cada instante de tempo e na execução do comando correspondente. Este mecanismo de sincronização de eventos condiciona a simulação de qualquer sistema digital, descrito em CDL, a uma operação síncrona, tornando difícil a simulação de eventos assíncronos que possam existir no sistema.

1.2 - DESCRIÇÃO DO SISTEMA EMMAC UTILIZANDO A LINGUAGEM CDL

Como o emulador de memórias de microcontrole recebe muitas entradas externas, vindas do seu painel ou do circuito em teste,para que sua simulação fosse possível, foi necessário acrescentar na descrição do sistema, memórias e registro externos, a fim de simular estas entradas e o seu sequenciamento.

A Figura A.2 representa as entradas externas necessárias à simulação do EMMAC e como elas foram descritas no programa CDL. O significado de cada uma delas, apresenta-se na parte de descrição do sistema, no programa simulador.

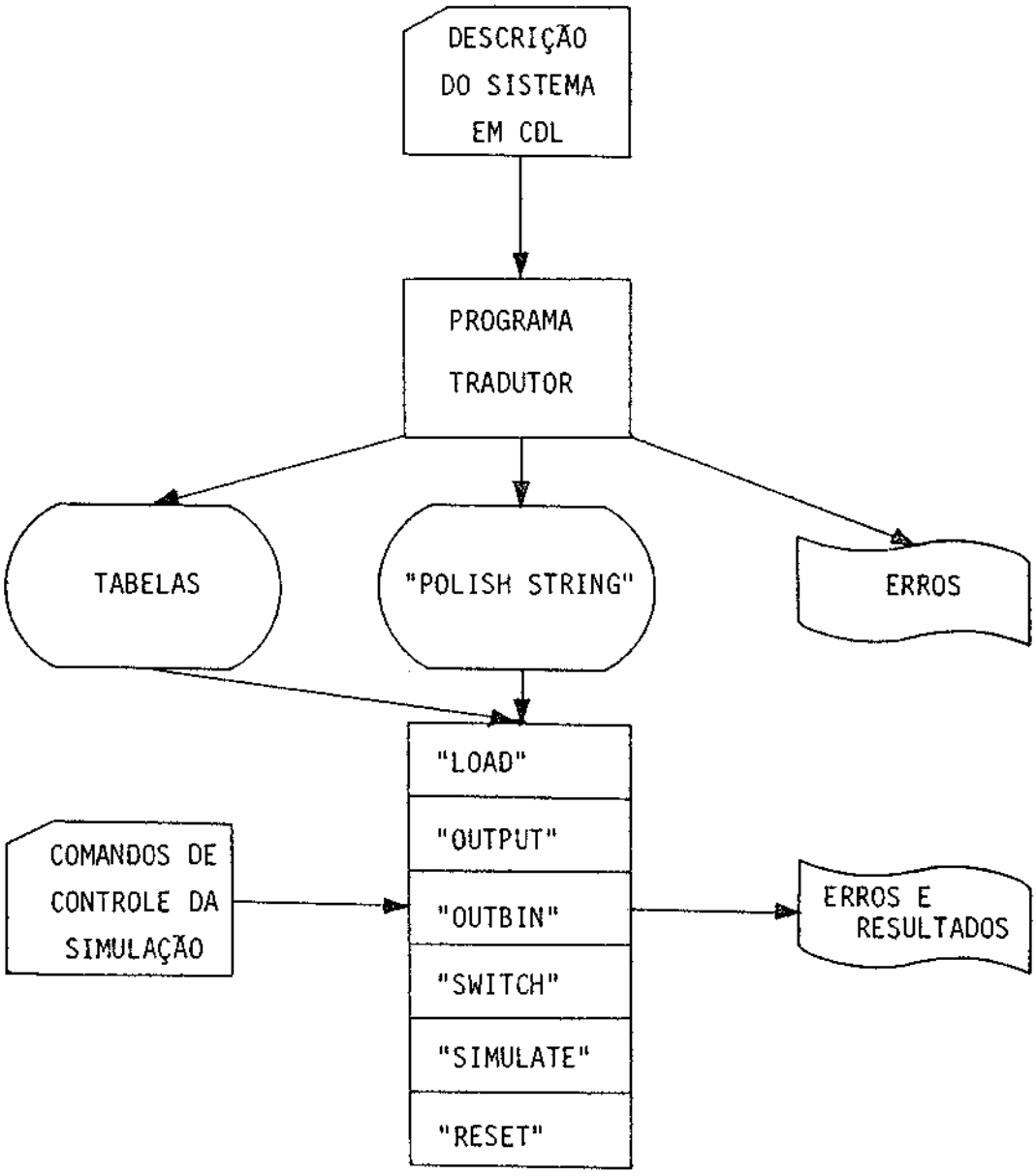

Fig. A.1 - Configuração do programa CDL implantado no INPE.

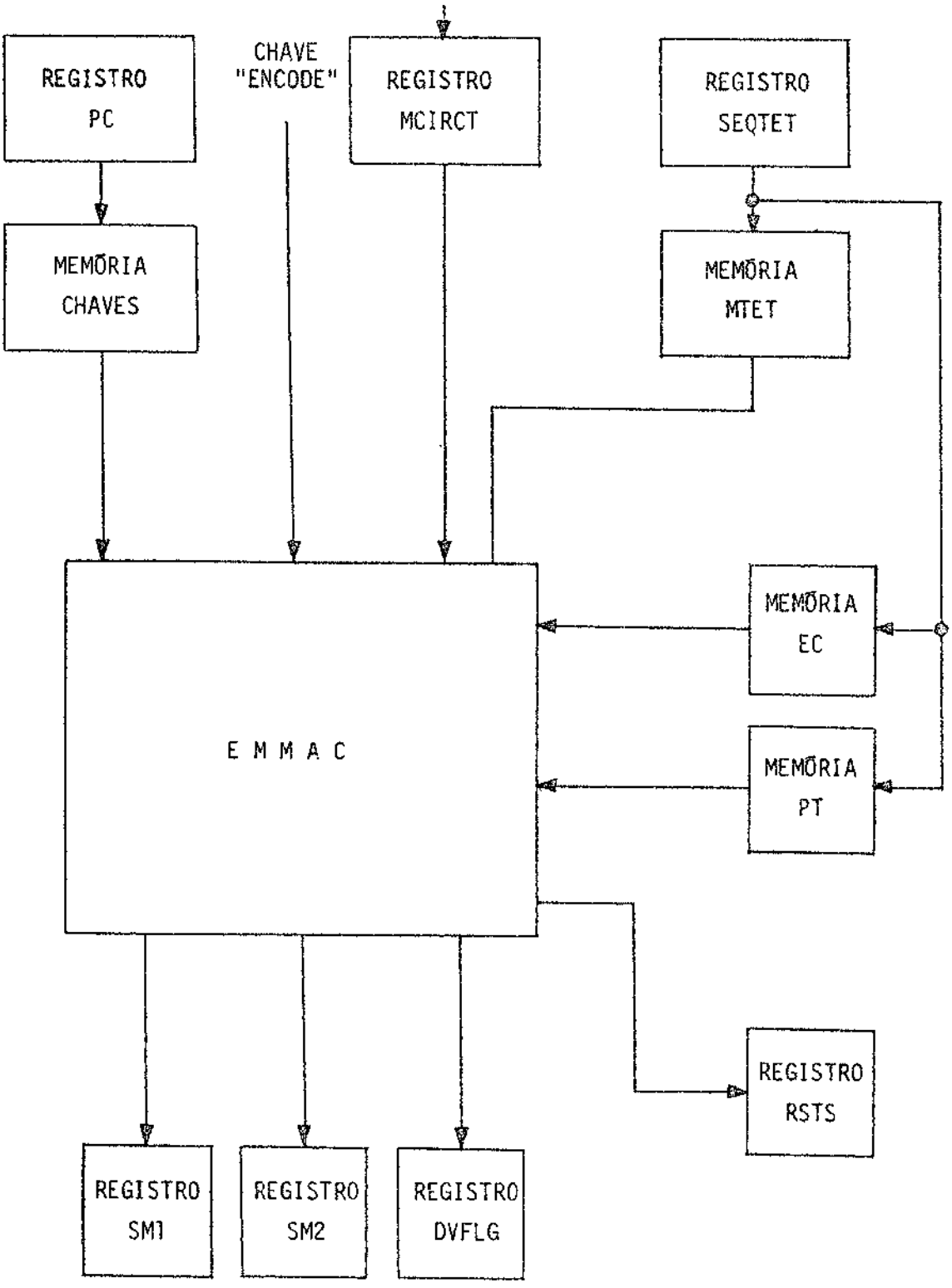

Fig. A.2 - Entradas e saídas externas descritas no programa simulador do EMMAC.

Na linguagem de simulação CDL, todas as constantes utilizadas são números octais, ou seja, todos os números que aparecerem na descrição e simulação do sistema digital estão na base 8.

A parte de descrição do sistema no programa simulador é constituída de duas sub-partes:

1) A descrição de registros, memórias, chaves, decodificadores e relógio existente no sistema a ser simulado, e operações combinacionais entre eles, normalmente definidas como. terminais ("TERMINAL");

2) As operações de transferências associadas ao relógio,definidas como comandos rotulados.

O relógio do emulador de memórias de microcontrole foi definido, em CDL, como tendo duas fases, sendo que a primeira fase, RELM (0), simula a transição positiva do relógio verdadeiro e a segunda fase, RELM(1), simula a transição negativa. O relógio de entrada do sistema em teste, RSTE, foi suposto como sendo o mesmo relógio RELM,interno, do emulador de memórias de microcontrole. Isto foi necessário porque o CDL permite a definição de apenas um relógio por sistema descrito.

Os operadores do CDL, utilizados na descrição do sistema EMMAC, e o seu significado estão listados na Tabela A.1.

TABELA A.1

OPERADORES CDL UTILIZADOS E SEUS SIGNIFICADOS

OPERADOR	SIGNIFICADO
-	Operação de concatenação de variáveis
+	Operação lógica OU
*	Operação lógica E
=	Operação de transferência entre registros
' ou ≥	Operação de negação lógica
.EQ.	Operação de igualdade aritmética
.COUNT.	Operação de incrementar registro

A listagem da descrição do sistema EMMAC com os seus componentes internos, descritos no item 3.2 do capítulo III, e suas entradas externas, mostradas na Figura A.2, é dada a seguir. Apenas 2 dos 20 módulos de memória de microcontrole, M1 e M2, foram descritos, para simplificar a simulação.

```
CDL VERSION 3                                   032979

STRANSLATE


*MAIN
C
C
C***SIMULACAO DO EMULADOR DE MEMORIAS DE MICROCONTROLE AUXILIADO POR
C   COMPUTADOR***
C
C
C***  E M M A C  ***
C
C
C***DECLARACAO DOS REGISTROS,MEMORIAS E TERMINAIS AUXILIARES***
C
C
C CHAVES(0-37,17-0)..MEMORIA QUE SIMULA AS 16CHAVES DO PAINEL
C PC(5-0)............REGISTRO DE ENDERECAMENTO DAS INSTRUCOES
C MCIRCT(17-0).......REGISTRO QUE SIMULA OS 16 FLIP-FLOPS DA MICROCIRCUI
C ENCODE(UM,ZERO)....CHAVE QUE SIMULA O SINAL ENCODE DA MICROCIRCUIT
C DVFLG..............REGISTRO QUE SIMULA O SINAL DEVICE FLAG DA MICROCIR
C SEQTET(5-0)........REGISTRO QUE SIMULA UM SEQUENCIADOR DO SIT.EM TESTE
C PT(0-37,0-37)......MEMORIA QUE SIMULA AS PONTAS DE TESTE DO CIRC.MONIT
C MTET(0-37,0-25)....MEMORIA QUE SIMULA OS END.EXTERNOS E OS TRI-STATE
C EC(0-37,10-0)......MEMORIA QUE SIMULA OS END. DE CONTROLE DO CIRC.EXT.
C SM1(7-0)...........REGISTRO QUE SIMULA A SAIDA DO MODULO 1
C SM2(7-0)...........REGISTRO QUE SIMULA A SAIDA DO MDULO 2
C
C
C***DECLARACAO EM CDL DAS VARIAVEIS ACIMA***
C
C
 REGISTER,PC(5-0),MCIRCT(17-0),DVFLG,SEQTET(5-0),SM1(7-0),SM2(7-0)
C
C
 MEMORY, PT(SEQTET)=PT(0-37,0-37),MTET(SEQTET)=MTET(0-37,0-25),
1        CHAVES(PC)=CHAVES(0-37,17-0),
1        EC(SEQTET)=EC(0-37,10-0)
C
C
 SWITCH,ENCODE(UM,ZERO)
C
C
C EM1(10-0).........ENDERECO VINDO DO SIST.EM TESTE PARA O MODULO 1
C ENM1AB,ENM1BB.....CONTR.DE 3-STATE PARA O MODULO 1
C EM2(10-0).........ENDERECO VINDO DO SIST. EM TESTE PARA O MODULO 2
C ENM2AB,ENM2BB.....CONTR.DE 3-STATE PARA O MODULO 2
C
C
 TERMINAL,EM1(10-0)=MTET(SEQTET)(0-10),
1         ENM1AB=MTET(SEQTET)(11),
1         ENM1BB=MTET(SEQTET)(12),
1         EM2(10-0)=MTET(SEQTET)(13-23),
1         ENM2AB=MTET(SEQTET)(24),
1         ENM2BB=MTET(SEQTET)(25)
```

```
C
C***UNIDADE DE CONTROLE DO EMMAC***
C
C
C***DECLARACAO DOS REGISTROS E MEMORIAS E CHAVES DA UNIDADE DE CONTROLE*
C
C
C RI(17-0).........REG DE INSTRUCAO DA UNIDADE DE CONTROLE
C MISEQ(7-0).......MICROSEQUENCIADOR DA UNIDADE DE CONTROLE
C RELM(0)..........RELOGIO DO EMMAC. TRANSICAO POSITIVA
C RELM(1)..........RELOGIO DO EMMAC. TRANSICAO NEGATIVA
C EXAS.............REG DE EXECUTE ASSINCRONO COM O RELM
C EXSI.............REG DE EXECUTE SINCRONO COM O RELOGIO RELM
C PARADO...........REG QUE AVISA QUANDO O EMMAC ESTA PARADO
C MCONTR(0-377,27-0)..MEMORIA DE CONTROLE DO EMMAC,256X24BITS
C RESETB(ZERO,UM)..CHAVE DE INICIALIZACAO
C RELOCB(ZERO,UM)..CHAVE DE SELECAO. REMOTO=UM,LOCAL=ZERO
C EXECUT(UM,ZERO)..CHAVE DE EXECUTAR INSTRUCOES NO MODO LOCAL
C LIGDES(DES,LIG)..CHAVE DE LIGAR E DESLIGAR O EMMAC
C
C
C***DECLARACAO EM CDL DAS VARIAVEIS ACIMA***
C
C
 REGISTER,RI(17-0),MISEQ(7-0),EXAS,EXSI,PARADO
C
 MEMORY,MCONTR(MISEQ)=MCONTR(0-377,27-0)
C
 SWITCH,RESETB(ZERO,UM),RELOCB(ZERO,UM),EXECUT(UM,ZERO),LIGDES(DES,LIG)
C
 CLOCK,  RELM(1)
C
C
C***SIMULACAO DA PARTE COMBINACIONAL DA UNIDADE DE CONTROLE***
C
C
C***SINAIS DE CONTROLE DO MUX DE ENTRADA DO REG DE INSTRUCAO***
C
C
 TERMINAL,SO(17-0)=EXSI-EXSI~EXSI-EXSI-EXSI-EXSI~EXSI-EXSI-
1                  EXSI-EXSI~EXSI-EXSI-EXSI-EXSI-EXSI-EXSI,
1          S(17-0)=RELOCB(UM)~RELOCB(UM)~RELOCB(UM)~RELOCB(UM)~RELOCB(UM)
1                  ~RELOCB(UM)~RELOCB(UM)~RELOCB(UM)~RELOCB(UM)~RELOCB(UM)
1   ~RELOCB(UM)~RELOCB(UM)~RELOCB(UM)~RELOCB(UM)~RELOCB(UM)~RELOCB(UM)
C
C
C***ENTRADA DO REGISTRO DE INSTRUCAO***
C
C
 TERMINAL,ENTRI(17-0)=MCIRCT*S+(CHAVES(PC)*SO+RI*SO≥)*S≥
C
C
C***SINAL DE RESET E CLEAR DE EXAS***
C
C
 TERMINAL, REST=RESETB(UM)*(RI(17)*RI(16)*RI(15)*RI(14))≥,
1          CL=REST*EXSI≥
C
C
C***SINAL DE PARADA DA UNIDADE DE CONTROLE***
```

```
C
C
 TERMINAL,PARDOB=MISEQ(0)+MISEQ(1)+MISEQ(2)+MISEQ(3)
C
C
C***DEFINICAO DOS BITS DE CONTROLE***
C
C
C CRSAID...CARREGA REGISTRO DE SAIDA
C CCEB      CARREGA CONTADOR DE ENDERECOS DO MODULO DE EMULACAO
C SE        SELECIONA ENDERECAMENTO DO MODULO DE EMULACAO, INT=0,EXT=1
C CRSB      CARREGA REGISTRO DE SELECAO DE LEITURA E ESCRITA
C LERB      LER O MODULO SELECIONADO OU MEMORIA DE MONITORAMENTO
C ESCRB     ESCREVER NO MODULO SELECIONADO
C SL        TRAVA O LATCH DE SAIDA PARA OPERACOES INTERNAS
C ICEME     INCREMENTA CONTADOR DE ENDERECOS DO MODULO DE EMULACAO
C CRCB      CARREGA O REGISTRO CONTADOR DO CIRCUITO DE CONTROLE DO REL.
C CSTB      CARREGA O REG DE STATUS DO CIRCUITO DE CONTROLE DO RELOGIO
C EXEC      EXECUTA, LIBERA O RELOGIO NO STATUS DEFINIDO
C CFB       CARREGA O REG DE FREQUENCIA DO MODULO DE MONITORAMENTO
C ICEMO     INCREMENTA O CONTADOR DE ENDERECO DO CIRC. DE MONITORAMENTO
C
C
 TERMINAL,CRSAID=MCONTR(MISEQ)(4),
1          CCEB=MCONTR(MISEQ)(5),
1          SE=MCONTR(MISEQ)(6),
1          CRSB=MCONTR(MISEQ)(7),
1          LERB=MCONTR(MISEQ)(10),
1          ESCRB=MCONTR(MISEQ)(11),
1          SL=MCONTR(MISEQ)(12),
1          ICEME=MCONTR(MISEQ)(13),
1          CRCB=MCONTR(MISEQ)(14),
1          CSTB=MCONTR(MISEQ)(15),
1          EXEC=MCONTR(MISEQ)(16),
1          CFB=MCONTR(MISEQ)(17),
1          ICEMO=MCONTR(MISEQ)(20)
C
C
C***SINAL DE DEVICE FLAG***
C
C
 TERMINAL,DEVFLA=RELOCB(UM)*MCONTR(MISEQ)(3)
C
C
C***CONTROLADOR DE ESCRITA E LEITURA DOS MODULOS DE EMULACAO***
C
C
C***DECLARACAO DOS REGISTROS E DECODIFICADORES DO CONTROLADOR***
C
C
C CEME(10-0).....CONTADOR DE ENDERECO DOS MODULOS DE EMULACAO
C RSEL(4-0).....REGISTRO DE SELECAO DE MODULO DE EMULACAO
C RSAID(7-0).....REGISTRO DE SAIDA DE DADOS
C DECSEL(0-23)....DECODIFICADOR DE SELECAO DE MODULO DE EMULACAO
C
C
C***DECLARACAO EM CDL DAS VARIAVEIS ACIMA****
C
C
 REGISTER,CEME(10-0),RSEL(4-0),RSAID(7-0)
C
```

```
 DECODER,DECSEL(0-23)=RSEL
C
C
C***SINAIS DE CONTROLE PARA OS MODULOS 1 E 2***
C
C
 TERMINAL,ESC11B=DECSEL(0)&,
1         ESC21B=DECSEL(1)&
C
C
C***DECLARACAO DOS REG, MEMORIAS E TERMINAIS DOS MODULOS DE EMULACAO***
C
C
C REM1(10-0).......REGISTRO VIRTUAL DE ENDERECAMENTO DA MEMORIA 1
C M1(0-777,7-0)....MEMORIA DO MODULO DE EMULACAO 1, 512X8 BITS
C REM2(10-0).......REGISTRO VIRTUAL DE ENDERECAMENTO DA MEMORIA 2
C M2(0-777,7-0)....MEMORIA DO MODULO DE EMULACAO 2, 512X8 BITS
C LATCH1(7-0)......LATCH DA SAIDA EXTERNA DO MODULO 1
C LATCH2(7-0)......LATCH DA SAIDA EXTERNA DO MODULO 2
C
C
C***DECLARACAO EM COL DAS VARIAVEIS ACIMA***
C
C
 REGISTER,REM1(10-0),REM2(10-0),LATCH1(7-0),LATCH2(7-0)
C
 MEMORY,M1(REM1)=M1(0-777,7-0),M2(REM2)=M2(0-777,7-0)
C
C
C***DEFINICAO DO SINAL DE CONTROLE DO MUX DE ENDERECAMENTO***
C
C
 TERMINAL,SEN(10-0)=SE-SE-SE-SE-SE-SE-SE-SE-SE
C
C
C***DEFINICAO DAS ENTRADAS DOS REG VIRTUAIS DE ENDERECAMENTO***
C
C
 TERMINAL,EREM1(10-0)=CEME*SEN&+EM1*SEN,
1         EREM2(10-0)=CEME*SEN&+EM2*SEN
C
C
C***DEFINICAO DOS SINAIS DE CONTROLE DE LEIT E ESCR DOS MODULOS 1 E 2***
C
C
 TERMINAL,L1=ESC11B+LERB,
1         L2=ESC21B+LERB,
1         CE1B=ESC11B+ESCRB,
1         CE2B=ESC21B+ESCRB,
1         CL1B(7-0)=L1-L1-L1-L1-L1-L1-L1-L1,
1         CL2B(7-0)=L2-L2-L2-L2-L2-L2-L2-L2
C
C
C***DECLARACAO DOS REGISTROS E TERMINAIS DO CIRC DE CONTR DO RELOGIO***
C
C
C RC(10-0).......REGISTRO CONTADOR DO CIRCUITO DE CONTROLE DO REL
C RCOND(10-0)....REGISTRO DE CONDICAO DO CIRC DE CONTROLE DO RELOGIO
C RST(1-0).......REGISTRO DE STATUS DO CIRC DE CONTROLE DO RELOGIO
C EXECU..........REGISTRO DE LIBERACAO DO RELOGIO
C PAP............REGISTRO DE MODO PASSO-A-PASSO
```

```
C CONd.....REGISTRO DE MODO CONDICIONAL ATE UM DET. ENDERECO
C LIV.....REGISTRO DE MODO LIVRE ATE UM NUMERO DE PULSOS
C RSTS.....REGISTRO AUXILIAR PARA IMPRESSAO DO REL DE SAIDA
C
C
C***DECLARACAO EM CDL DAS VARIAVEIS ACIMA***
C
C
 REGISTER,RC(10-0),RCOND(10-0),RST(1-0),EXECU,PAP,COND,LIV,RSTS
C
C
C***DEFINICAO DAS CONDICOES DE FIM E DO REL DE SAIDA***
C
C
 TERMINAL,IGUAL=RCOND,EQ,RC,
1          COUT=RC(10)*RC(7)*RC(6)*RC(5)*RC(4)*RC(3)*RC(2)*RC(1)*RC(0),
1          FIMB=PAP≥*RST(1)≥*RST(0)≥+RST(1)≥*RST(0)*COND≥*RST(1)*RST(0)≥+
1          LIV≥*RST(1)*RST(0),
1          TRSTS=RELM(0)*EXECU*FIMB
C
C
C***SAIDA DO MUX DE TESTE DA UNIDADE DE CONTROLE***
C
C
 TERMINAL,ENT=EXSI*MCONTR(MISEQ)(2)≥*MCONTR(MISEQ)(1)≥+MCONTR(MISEQ)(2)≥
1          *MCONTR(MISEQ)(1)+FIMB≥*MCONTR(MISEQ)(2)*MCONTR(MISEQ)(1)≥
C
C
C***DECLARACAO DOS REGISTROS E MEMORIAS DO CIRC DE MONITORAMENTO***
C
C
C RM00(7-0).....REG DE ARMAZENAMENTO DAS PONTAS DE TESE 0-7
C RM01(7-0).....REG DE ARMAZENAMENTO DAS PONTAS DE TESTE 8-15
C RM02(7-0).....REG DE ARMAZENAMENTO DAS PONTAS DE TESTE 16-23
C RM03(7-0).....REG DE ARMAZENAMENTO DAS PONTAS DE TESTE 24-31
C CEMO(7-0).....CONTADOR DE ENDERECOS DA MEMORIA DE MONITORAMENTO
C MM00(0-377,7-0),MEMORIA DE MONITORAMENTO DAS PONTAS 0-7
C MM01(0-377,7-0),MEMORIA DE MONITORAMENTO DAS PONTAS 8-15
C MM02(0-377,7-0),MEMORIA DE MONITORAMENTO DAS PONTAS 16-23
C MM03(0-377,7-0),MEMORIA DE MONITORAMENTO DAS PONTAS 24-31
C RFREQ(3-0).....REGISTRO DE FREQUENCIA DO CIRC DE MONITORAMENTO
C CFREQ(3-0).....CONTADOR DE FREQUENCIA DO CIRC DE MONITORAMENTO
C DECLMO(0-3).....DECODIFICADOR DE LEITURA DAS MEMORIAS DE MONIT.
C ARMA.....REGISTRO DE CONTROLE DE ESCRITA
C
C
C***DECLARACAO EM CDL DAS VARIAVEIS ACIMA***
C
C
 REGISTER,RM00(7-0),RM01(7-0),RM02(7-0),RM03(7-0),CEMO(7-0),RFREQ(3-0),
1          CFREQ(7-0),ARMA
C
 MEMORY,MM00(CEMO)=MM00(0-377,7-0),MM01(CEMO)=MM01(0-377,7-0),
1        MM02(CEMO)=MM02(0-377,7-0),MM03(CEMO)=MM03(0-377,7-0)
C
 DECODER,DECLMO(0-3)=RI(1-0)
C
C
C***SINAIS DE CONTROLE DE LEITURA DAS MEMORIAS DE MONIT.***
C
C
```

```
 TERMINAL,CLM0=LERB≥+DECLM0(0)≥,
1          CLM1=LERB≥+DECLM0(1)≥,
1          CLM2=LERB≥+DECLM0(2)≥,
1          CLM3=LERB≥+DECLM0(3)≥,
1          CLM00(7-0)=CLM0-CLM0-CLM0-CLM0-CLM0-CLM0-CLM0-CLM0,
1          CLM01(7-0)=CLM1-CLM1-CLM1-CLM1-CLM1-CLM1-CLM1-CLM1,
1          CLM02(7-0)=CLM2-CLM2-CLM2-CLM2-CLM2-CLM2-CLM2-CLM2,
1          CLM03(7-0)=CLM3-CLM3-CLM3-CLM3-CLM3-CLM3-CLM3-CLM3,
C
C
C***DEFINICAO DOS SINAIS PARA ARMAZENAMENTO DOS ESTADOS E INC. CFREQ***
C
C
 TERMINAL,ARMAZ=CFREQ(0)*CFREQ(1)*CFREQ(2)*CFREQ(3),
1          LCFREQ=CFB*ARMAZ≥
C
C
C***DEFINICAO DA SAIDA DE DADOS DADL, INTERNA***  .
C
C
 TERMINAL,DADL(7-0)=M1(REM1)*CL1B≥+M2(REM2)*CL2B≥+MM00(CEM0)*CLM00≥+
1          MM01(CEM0)*CLM01≥+MM02(CEM0)*CLM02≥+MM03(CEM0)*CLM03≥
C
C
C***FIM DA PARTE COMBINACIONAL***
C
C
C
C
C***OPERACOES DE CHAVES EXTERNAS***
C
C
 /LIGOES(LIG)/  EXAS=0,MISEQ(3-0)=0,EXECU=0,RESETB=UM,PC=0,COND=0
 /RESETR(ZERO)/  EXAS=0,MISEQ(3-0)=0,EXECU=0,RESETB=UM,COND=0
 /EXECUT(ZERO)/  EXAS=1,PC=PC.COUNT.,  EXECUT=UM
 /ENCODF(ZERO)/ IF (RELOCB(UM)) THEN (IF (EXECUT(UM)) THEN (EXAS=1))
C
C
C***OPERACOES COMANDADAS PELAS TRANSICOES DO RELOGIO RELM***
C
C
C***UNIDADE DE CONTROLE***
C
C
 /RELM(0)/
C
C
      MISEQ(7-4)=RI(17-14),
      EXSI=EXAS*CL,
      IF (CL≥) THEN (EXAS=0),
      PARAND=PARDOB≥,
      IF (PEST≥+MCONTR(MISEQ)(0)≥) THEN (MISEQ(3-0)=0)
                                    ELSE (IF (ENT) THEN (MISEQ(3-0)=
                                                         MISEQ(3-0).COUNT.,
                                                  MISEQ(7-4)=ENTRI(17-14))),
      DVFLG=DEVFLA
C
 /RELM(0)*EXSI/  RI=ENTRI
C
C
C***CONTROLADOR DE MEMORIA E REGISTRO DE SAIDA***
```

```
C
C
 /RELM(0)*CCEB≥/  CEME=RI(10-0)
C
C
 /RELM(0)*CCEB*ICEME/  CEME=CEME.COUNT.
C
C
 /RELM(0)*CRSB≥/  RSEL=RI(4-0)
C
C
 /RELM(0)*CRSAID≥/  RSAID=DAUL
C
C
C***MODULO DE EMULACAO DE MEMORIAS PROMS***
C
C
 /RELM(0)/
C
C
      REM1=EREM1.
      REM2=EREM2.
      IF (SL) THEN (IF (ENM1AB≥*ENM1BB≥) THEN (SM1=LATCH1).
                    IF (ENM2AB≥*ENM2BB≥) THEN (SM2=LATCH2))
              ELSE (LATCH1=M1(REM1).
                    LATCH2=M2(REM2).
                    IF (ENM1AB≥*ENM1BB≥) THEN (SM1=M1(REM1)).
                    IF (ENM2AB≥*ENM2BB≥) THEN (SM2=M2(REM2)))
C
C
 /RELM(1)*CE1B≥/  M1(REM1)=RI(7-0)
C
C
 /RELM(1)*CE2B≥/   M2(REM2)=RI(7-0)
C
C
C***CIRCUITO DE CONTROLE DO RELOGIO***
C
C
 /(RELM(0)+CRCB)*(TRSTS+CRCB≥)/
C
C
       RCOND=EC(SEQTET).
      IF (CRCB≥) THEN (RC=RI(10-0))
                 ELSE (IF (RST(0)*RST(1)) THEN (RC=RC.COUNT.)).
C
C
 /RELM(0)*CSTB≥/  RST=RI(1-0).
C
C
C
C
 /RELM(0)/   RSTS=TRSTS. ARMA=ARMAZ≥.
C
C
 /RELM(1)*EXECU*FIMB/
C
       PAP=1.
      COND=IGUAL.
      LIV=COUT.
C
```

```
  /RELM(1)/   EXECU=EXEC
C
C
C***CIRCUITO DE MONITORAMENTO DAS PONTAS DE TESTE***
C
C
 /RELM(0)*CFBΣ/  RFREQ=RI(3-0)
C
C
 /RELM(0)*CFBΣ*TRSTS/ IF (LCFREQΣ) THEN (CFREQ=RFREQ)
                                    ELSE (CFREQ=CFREQ.COUNT.)
C
C
 /RELM(0)*EXECUΣ*TRSTS/  IF (ICEMO+ARMAZ*EXECU) THEN (CEMO=CEMO.COUNT.)*
                          RMO0=PT(SEQTET)(0-7),
                          RMO1=PT(SEQTET)(10-17),
                          RMO2=PT(SEQTET)(20-27),
                          RMO3=PT(SEQTET)(30-37)
C
C
 /RELM(1)/  IF (ARMAΣ) THEN (MMO0(CEMO)=RMO0,
                                          MMO1(CEMO)=RMO1,
                                          MMO2(CEMO)=RMO2,
                                          MMO3(CEMO)=RMO3)
C
C
C***OPERACOES SINCRONAS NO CIRCUITO EXTERNO***
C
C
 /RELM(0)*TRSTS/ SEQTET=SEQTET.COUNT.,
C
C
C***FIM DA PARTE DE TRADUCAO DO SIMULADOR***
C
C
 END
MEMORY REQUIRED FOR TRANSLATION =   1986 WORDS.
MEMORY REQUIRED FOR SIMULATION  =   2693 WORDS.
```

1.3 - SIMULAÇÃO DO SISTEMA EMMAC

Para a simulação do sistema, descrito anteriormente, é necessário, através dos comandos de simulação da linguagem CDL, definir os registros e posições de memórias a serem listados, as interrupções de chaves e o ciclo de rótulo onde elas ocorrem, e as condições iniciais dos registros e das memórias internas ao EMMAC e, também, das suas entradas externas.

O comando OUTPUT do CDL, especifica quais os registros e posições de memórias que serão listados, e de quantos em quantos ciclos de rótulo ou tempos do relógio eles serão listados. Por exemplo, o comando

```
*OUTPUT      CLOCK(1)=RI,EXAS,M1(0),LATCH1,M2(2)
```

especifica que os conteúdos dos registros RI, EXAS e LATCH1 e das posições de memória 0 de M1 e 2 de M2, serão listados a cada tempo do relógio. Um tempo do relógio é contado a cada vez que o relógio passa pela sua fase número 0. Os conteúdos dos registros serão listados com o nome do registro, seguido de um sinal de igual e do valor em octal do seu conteúdo ao terminar o tempo do relógio. As posições de memórias serão listadas com o número octal correspondente à posição, seguido de um sinal de igual e do valor em octal do conteúdo daquela posição ao terminar o tempo do relógio, sendo identificada a memória pela ordem pedida no comando "OUTPUT". Além das variáveis pedidas, ainda é listado, para cada tempo do relógio, o número do ciclo de rótulo, os rótulos verdadeiros naquele ciclo e o número de tempos do relógio já simulados.

O comando de saída acima, produz a seguinte listagem para cada tempo do relógio:

```
*******************************************
LABEL CYCLE n                TRUE LABELS       CLOCK TIME m
                           / ROT 1 /
                           /  ...  /
                           / ROT k /
RI=030001       EXAS=0       000001=000
LATCH1=071      000002=130
*******************************************
```

O comando SWITCH do CDL indica antes de qual ciclo de rótulo a chave especificada será colocada na posição desejada. Por exemplo, o comando

```
*SWITCH      1,LIGDES=LIG
```

indica que a chave LIGDES será colocada na posição LIG, antes do primeiro ciclo de rótulo. A todo comando SWITCH corresponde uma interrupção de chave, que ocasionará a listagem das variáveis especificadas no comando OUTPUT.

O comando LOAD do CDL especifica as condições iniciais,a serem carregadas nos registros e memórias, antes de começar a simulação.

O comando SIM do CDL, ocasiona o início da simulação, especificando o número de ciclos de rótulos a serem simulados e o número de repetição de um mesmo rótulo tolerado, ou seja, por exemplo, o comando

```
*SIM      60,3
```

indica que a simulação constará de 60 ciclos de rótulos, sendo que se o mesmo conjunto de rótulos for verdadeiro, durante 3 ciclos de rótulos seguidos, a simulação será terminada.

Para melhor validar o funcionamento do circuito do emulador de memórias de microcontrole, sua simulação foi dividida em duas partes:

1) Simulação das operações internas ao emulador de memórias de microcontrole;
2) Simulação das operações associadas às entradas externas vindas do sistema em teste.

Nas duas simulações o EMMAC recebe instruções vindas das chaves do painel, simuladas pela memória CHAVES, endereçada pelo registro PC.

A memória de controle, MCONTR, do circuito do emulador é carregada no comando LOAD com as microinstruções em octal, necessárias para executar todas as instruções listadas na Tabela III.5, do capítulo III.

1.3.1 - SIMULAÇÃO DAS OPERAÇÕES INTERNAS AO EMMAC

As instruções do emulador de memórias de microcontrole testadas nesta simulação foram armazenadas na memória CHAVES, sendo a sua execução iniciada através de um comando SWITCH, simulando o botão EXECUTE do painel.

Estas instruções estão listadas na Tabela A.2, junto com suas posições na memória CHAVES, o código em octal correspondente e o ciclo de rótulo da listagem de saída, onde a execução da instrução é concluída.

TABELA A.2

INSTRUÇÕES SIMULADAS EM CDL

INSTRUÇÃO	POSIÇÃO (octal)	CÓDIGO (octal)	CICLO DE RÓTULO
CACES 1	CHAVES(1)	030001	7
CARSE 1	CHAVES(2)	020001	13
ESCRI 10	CHAVES(3)	130010	19
ESCR 20	CHAVES(4)	010020	25
CACES 1	CHAVES(5)	030001	31
LERI	CHAVES(6)	000000	37
LER	CHAVES(7)	120000	43
CARC 5	CHAVES(10)	040005	49
CARS 0	CHAVES(11)	050000	55
CARF 10	CHAVES(12)	060010	63
INCMO	CHAVES(13)	070000	69
LERMO	CHAVES(14)	100000	75

A seguir é apresentada a listagem dos comandos de simulação, para as operações internas ao EMMAC, e dos resultados obtidos da simulação.

```
$SIMULATE

CDL VERSION 3                                        032979
                                        S I M U L A T I O N
*OUTPUT   CLOCK(1)=RI,EXAS,EXSI,MISEQ,PARADD,OVFLG,CEME,RSEL,RSAID,
          M1(0),M1(1),M1(2),M1(3),M1(4),M1(5),LATCH1,SM1,
          M2(0),M2(1),M2(2),M2(3),M2(4),M2(5),LATCH2,SM2,
          RCOND,RC,RST,RSTS,RMOD0,RMOD1,RMOD2,RMOD3,MMOD0(0),
          MMOD0(1),MMOD1(0),MMOD1(1),MMOD2(0),MMOD2(1),MMOD3(0),
          MMOD3(1),RFREQ,CFREQ,CEMO
*SWITCH   1,LIGDES=LIG
*SWITCH   1,RELOCA=ZERO
*SWITCH   3,EXECUT=ZERO
*SWITCH   11,EXECUT=ZERO
*SWITCH   17,EXECUT=ZERO
*SWITCH   25,EXECUT=ZERO
*SWITCH   33,EXECUT=ZERO
*SWITCH   41,EXECUT=ZERO
*SWITCH   47,EXECUT=ZERO
*SWITCH   55,EXECUT=ZERO
*SWITCH   63,EXECUT=ZERO
*SWITCH   71,EXECUT=ZERO
```

```
         *SWITCH  101,EXECUT=ZERO
         *SWITCH  107,EXECUT=ZERO
         *LOAD
 MCONTR(0-1)=133661,137250,
 MCONTR(20-21)=133661,132650,
 MCONTR(40-41)=133661,133470,
 MCONTR(60-61)=133661,133630,
 MCONTR(100-101)=133661,123670,
 MCONTR(120-121)=133661,113670,
 MCONTR(140-142)=133661,33663,33670,
 MCONTR(160-161)=133661,333670,
 MCONTR(200-201)=133661,133650,
 MCONTR(220-223)=133661,133673,171765,133660,
 MCONTR(240-241)=133661,133250,
 MCONTR(260-261)=133661,136650,
 MCONTR(300-303)=133661,133673,171665,133660,
 MCONTR(320)=133661,
 MCONTR(340)=133661,
 MCONTR(360)=133661,
         *LOAD
 CEMO=0,SEQTET=0,MM00(1)=35,
 M1(0-5)=0,10,20,30,40,50,
 M2(0-5)=5,15,25,35,45,55,
 MTET(0-5)=0,20004,40010,60010,100014,120020,
 EC(0-5)=0,1,2,3,4,5,
 PT(0-6)=3774040177 4,542177577,37777777777,1234567 1234,74563211234,0,2,
 CHAVES(1-5)=30001,20001,130010,10020,30001,
 CHAVES(6-12)=0,120000,40005,50000,60010,
 CHAVES(13-16)=70000,100000,110000,170000,
         *SIM      116,64
```

OUTPUT OF SIMULATION

```
SWITCH INTERRUPT
    LIGDES = LIG
  RI      = 000000    EXAS    = 0         EXSI    = 0
  MISEQ   = 000       PARADD  = 0         DVFLG   = 0
  CEME    = 000       RSEL    = 00        RSAID   = 000
  000000  = 000       000001  = 010       000002  = 020
  000003  = 030       000004  = 040       000005  = 050
  LATCH1  = 000       SM1     = 000       000000  = 005
  000001  = 015       000002  = 025       000003  = 035
  000004  = 045       000005  = 055       LATCH2  = 000
  SM2     = 020       RCOND   = 120       RC      = 040
  RST     = 2         RSTS    = 1         RM00    = 065
  RM01    = 041       RM02    = 000       RM03    = 000
  000000  = 226       000001  = 035       000000  = 001
  000001  = 000       000000  = 000       000001  = 000
  000000  = 000       000001  = 000       RFREQ   = 00
  CFREQ   = 214       CEMO    = 000
**************************************************
LABEL CYCLE    1                TRUE LABELS                CLOCK TIME =    .1
                              /RELM(0)/
                              /RELM(0)/
                              /RELM(0)/
                              /RELM(0)*EXECU≥+T
  RI      = 000000    EXAS    = 0         EXSI    = 0
  MISEQ   = 000       PARADD  = 1         DVFLG   = 0
  CEME    = 000       RSEL    = 00        RSAID   = 000
```

```
  000000 = 000        000001 = 010        000002 = 020
  000003 = 030        000004 = 040        000005 = 050
  LATCH1 = 000        SM1    = 000        000000 = 005
  000001 = 015        000002 = 025        000003 = 035
  000004 = 045        000005 = 055        LATCH2 = 000
  SM2    = 000        RCOND  = 120        RC     = 040
  RST    = 2          RSTS   = 0          RM00   = 377
  RM01   = 202        RM02   = 003        RM03   = 374
  000000 = 226        000001 = 035        000000 = 001
  000001 = 000        000000 = 000        000001 = 000
  000000 = 000        000001 = 000        RFREQ  = 00
  CFREQ  = 214        CEMD   = 000
**************************************************

SWITCH INTERRUPT
    EXECUT = ZERO
  RI     = 000000     EXAS   = 1          EXSI   = 0
  MISEQ  = 000        PARADD = 1          DVFLG  = 0
  CEME   = 000        RSEL   = 00         RSAID  = 000
  000000 = 000        000001 = 010        000002 = 020
  000003 = 030        000004 = 040        000005 = 050
  LATCH1 = 000        SM1    = 000        000000 = 005
  000001 = 015        000002 = 025        000003 = 035
  000004 = 045        000005 = 055        LATCH2 = 000
  SM2    = 000        RCOND  = 120        RC     = 040
  RST    = 2          RSTS   = 0          RM00   = 377
  RM01   = 202        RM02   = 003        RM03   = 374
  000000 = 226        000001 = 035        000000 = 001
  000001 = 000        000000 = 000        000001 = 000
  000000 = 000        000001 = 000        RFREQ  = 00
  CFREQ  = 214        CEMD   = 000
**************************************************
LABEL CYCLE    3              TRUE LABELS              CLOCK TIME =     2
                            /RELM(0)/
                            /RELM(0)/
                            /RELM(0)/
                            /RELM(0)*EXECU2+T
  RI     = 000000     EXAS   = 1          EXSI   = 1
  MISEQ  = 000        PARADD = 1          DVFLG  = 0
  CEME   = 000        RSEL   = 00         RSAID  = 000
  000000 = 000        000001 = 010        000002 = 020
  000003 = 030        000004 = 040        000005 = 050
  LATCH1 = 000        SM1    = 000        000000 = 005
  000001 = 015        000002 = 025        000003 = 035
  000004 = 045        000005 = 055        LATCH2 = 000
  SM2    = 000        RCOND  = 120        RC     = 040
  RST    = 2          RSTS   = 0          RM00   = 377
  RM01   = 202        RM02   = 003        RM03   = 374
  000000 = 226        000001 = 035        000000 = 001
  000001 = 000        000000 = 000        000001 = 000
  000000 = 000        000001 = 000        RFREQ  = 00
  CFREQ  = 214        CEMD   = 000
**************************************************
LABEL CYCLE    5              TRUE LABELS              CLOCK TIME =     3.
                            /RELM(0)/
                            /RELM(0)*EXSI/
                            /RELM(0)/
                            /RELM(0)/
                            /RELM(0)*EXECU2+T
  RI     = 030001     EXAS   = 0          EXSI   = 0
  MISEQ  = 061        PARADD = 1          DVFLG  = 0
```

```
  CEME    = 000       RSEL    = 00        RSAID   = 000
  000000  = 000       000001  = 010       000002  = 020
  000003  = 030       000004  = 040       000005  = 050
  LATCH1  = 000       SM1     = 000       000000  = 005
  000001  = 015       000002  = 025       000003  = 035
  000004  = 045       000005  = 055       LATCH2  = 000
  SM2     = 000       RCOND   = 120       RC      = 040
  RST     = 2         RSTS    = 0         RM00    = 377
  RM01    = 202       RM02    = 003       RM03    = 374
  000000  = 226       000001  = 035       000000  = 001
  000001  = 000       000000  = 000       000001  = 000
  000000  = 000       000001  = 000       RFREQ   = 00
  CFREQ   = 214       CEM0    = 000
**************************************************
LABEL CYCLE    7                TRUE LABELS              CLOCK TIME =     4
                               /RELM(0)/
                               /RELM(0)*CCEB≥/
                               /RELM(0)/
                               /RELM(0)/
                               /RELM(0)*EXECU≥+T
  RI      = 030001    EXAS    = 0         EXSI    = 0
  MISEQ   = 060       PARADD  = 0         OVFLG   = 0
  CEME    = 001       RSEL    = 00        RSAID   = 000
  000000  = 000       000001  = 010       000002  = 020
  000003  = 030       000004  = 040       000005  = 050
  LATCH1  = 000       SM1     = 000       000000  = 005
  000001  = 015       000002  = 025       000003  = 035
  000004  = 045       000005  = 055       LATCH2  = 000
  SM2     = 000       RCOND   = 120       RC      = 040
  RST     = 2         RSTS    = 0         RM00    = 377
  RM01    = 202       RM02    = 003       RM03    = 374
  000000  = 226       000001  = 035       000000  = 001
  000001  = 000       000000  = 000       000001  = 000
  000000  = 000       000001  = 000       RFREQ   = 00
  CFREQ   = 214       CEM0    = 000
**************************************************

SWITCH INTERRUPT
    EXECUT = ZERO
  RI      = 030001    EXAS    = 1         EXSI    = 0
  MISEQ   = 060       PARADD  = 0         OVFLG   = 0
  CEME    = 001       RSEL    = 00        RSAID   = 000
  000000  = 000       000001  = 010       000002  = 020
  000003  = 030       000004  = 040       000005  = 050
  LATCH1  = 000       SM1     = 000       000000  = 005
  000001  = 015       000002  = 025       000003  = 035
  000004  = 045       000005  = 055       LATCH2  = 000
  SM2     = 000       RCOND   = 120       RC      = 040
  RST     = 2         RSTS    = 0         RM00    = 377
  RM01    = 202       RM02    = 003       RM03    = 374
  000000  = 226       000001  = 035       000000  = 001
  000001  = 000       000000  = 000       000001  = 000
  000000  = 000       000001  = 000       RFREQ   = 00
  CFREQ   = 214       CEM0    = 000
**************************************************
LABEL CYCLE    9                TRUE LABELS              CLOCK TIME =     5
                               /RELM(0)/
                               /RELM(0)/
                               /RELM(0)/
                               /RELM(0)*EXECU≥+T
  RI      = 030001    EXAS    = 1         EXSI    = 1
```

```
MISEQ  = 060      PARADO = 1        DVFLG  = 0
CEME   = 001      RSEL   = 00       RSAID  = 000
000000 = 000      000001 = 010      000002 = 020
000003 = 030      000004 = 040      000005 = 050
LATCH1 = 000      SM1    = 000      000000 = 005
000001 = 015      000002 = 025      000003 = 035
000004 = 045      000005 = 055      LATCH2 = 000
SM2    = 000      RCOND  = 120      RC     = 040
RST    = 2        RSTS   = 0        RMD0   = 377
RMD1   = 202      RMD2   = 003      RMD3   = 374
000000 = 226      000001 = 035      000000 = 001
000001 = 000      000000 = 000      000001 = 000
000000 = 000      000001 = 000      RFREQ  = 00
CFREQ  = 214      CEMD   = 000
**************************************************
LABEL CYCLE   11                TRUE LABELS              CLOCK TIMF =    6
                               /RELM(0)/
                               /RELM(0)*EXSI/
                               /RELM(0)/
                               /RELM(0)/
                               /RELM(0)*EXECU≥+T
RI     = 020001   EXAS   = 0        EXSI   = 0
MISEQ  = 041      PARADO = 1        DVFLG  = 0
CEME   = 001      RSEL   = 00       RSAID  = 000
000000 = 000      000001 = 010      000002 = 020
000003 = 030      000004 = 040      000005 = 050
LATCH1 = 000      SM1    = 000      000000 = 005
000001 = 015      000002 = 025      000003 = 035
000004 = 045      000005 = 055      LATCH2 = 000
SM2    = 000      RCOND  = 120      RC     = 040
RST    = 2        RSTS   = 0        RMD0   = 377
RMD1   = 202      RMD2   = 003      RMD3   = 374
000000 = 226      000001 = 035      000000 = 001
000001 = 000      000000 = 000      000001 = 000
000000 = 000      000001 = 000      RFREQ  = 00
CFREQ  = 214      CEMD   = 000
**************************************************
LABEL CYCLE   13                TRUE LABELS              CLOCK TIMF =    7
                               /RELM(0)/
                               /RELM(0)*CRSB≥/
                               /RELM(0)/
                               /RELM(0)/
                               /RELM(0)*EXECU≥+T
RI     = 020001   EXAS   = 0        EXSI   = 0
MISEQ  = 040      PARADO = 0        DVFLG  = 0
CEME   = 001      RSEL   = 01       RSAID  = 000
000000 = 000      000001 = 010      000002 = 020
000003 = 030      000004 = 040      000005 = 050
LATCH1 = 000      SM1    = 000      000000 = 005
000001 = 015      000002 = 025      000003 = 035
000004 = 045      000005 = 055      LATCH2 = 000
SM2    = 000      RCOND  = 120      RC     = 040
RST    = 2        RSTS   = 0        RMD0   = 377
RMD1   = 202      RMD2   = 003      RMD3   = 374
000000 = 226      000001 = 035      000000 = 001
000001 = 000      000000 = 000      000001 = 000
000000 = 000      000001 = 000      RFREQ  = 00
CFREQ  = 214      CEMD   = 000
**************************************************

SWITCH INTERRUPT
```

```
   EXECUT = ZERO
  RI      = 020001    EXAS    = 1         EXSI    = 0
  MISEQ   = 040       PARADD  = 0         DVFLG   = 0
  CEME    = 001       RSEL    = 01        RSAID   = 000
  000000  = 000       000001  = 010       000002  = 020
  000003  = 030       000004  = 040       000005  = 050
  LATCH1  = 000       SM1     = 000       000000  = 005
  000001  = 015       000002  = 025       000003  = 035
  000004  = 045       000005  = 055       LATCH2  = 000
  SM2     = 000       RCOND   = 120       RC      = 040
  RST     = 2         RSTS    = 0         RMD0    = 377
  RMD1    = 202       RMD2    = 003       RMD3    = 374
  000000  = 226       000001  = 035       000000  = 001
  000001  = 000       000000  = 000       000001  = 000
  000000  = 000       000001  = 000       RFREQ   = 00
  CFREQ   = 214       CEMD    = 000
**************************************************
LABEL CYCLE   15                TRUE LABELS              CLOCK TIME =    8
                               /RELM(0)/
                               /RELM(0)/
                               /RELM(0)/
                               /RELM(0)*EXECUTE+T
  RI      = 020001    EXAS    = 1         EXSI    = 1
  MISEQ   = 040       PARADD  = 1         DVFLG   = 0
  CEME    = 001       RSEL    = 01        RSAID   = 000
  000000  = 000       000001  = 010       000002  = 020
  000003  = 030       000004  = 040       000005  = 050
  LATCH1  = 000       SM1     = 000       000000  = 005
  000001  = 015       000002  = 025       000003  = 035
  000004  = 045       000005  = 055       LATCH2  = 000
  SM2     = 000       RCOND   = 120       RC      = 040
  RST     = 2         RSTS    = 0         RMD0    = 377
  RMD1    = 202       RMD2    = 003       RMD3    = 374
  000000  = 226       000001  = 035       000000  = 001
  000001  = 000       000000  = 000       000001  = 000
  000000  = 000       000001  = 000       RFREQ   = 00
  CFREQ   = 214       CEMD    = 000
**************************************************
LABEL CYCLE   17                TRUE LABELS              CLOCK TIME =    9
                               /RELM(0)/
                               /RELM(0)*EXSI/
                               /RELM(0)/
                               /RELM(0)/
                               /RELM(0)*EXECUTE+T
  RI      = 130010    EXAS    = 0         EXSI    = 0
  MISEQ   = 261       PARADD  = 1         DVFLG   = 0
  CEME    = 001       RSEL    = 01        RSAID   = 000
  000000  = 000       000001  = 010       000002  = 020
  000003  = 030       000004  = 040       000005  = 050
  LATCH1  = 000       SM1     = 000       000000  = 005
  000001  = 015       000002  = 025       000003  = 035
  000004  = 045       000005  = 055       LATCH2  = 000
  SM2     = 000       RCOND   = 120       RC      = 040
  RST     = 2         RSTS    = 0         RMD0    = 377
  RMD1    = 202       RMD2    = 003       RMD3    = 374
  000000  = 226       000001  = 035       000000  = 001
  000001  = 000       000000  = 000       000001  = 000
  000000  = 000       000001  = 000       RFREQ   = 00
  CFREQ   = 214       CEMD    = 000
**************************************************
LABEL CYCLE   19                TRUE LABELS              CLOCK TIME =   10
```

```
                                    /RELM(0)/
                                    /RELM(0)*CCEB*ICE
                                    /RELM(0)*CRSAID≥/
                                    /RELM(0)/
                                    /RELM(0)/
                                    /RELM(0)*EXECU≥+T
  RI      = 130010    EXAS    = 0         EXSI    = 0
  MISEQ   = 260       PARADO  = 0         OVFLG   = 0
  CEME    = 002       RSEL    = 01        RSAID   = 226
  000000  = 000       000001  = 010       000002  = 020
  000003  = 030       000004  = 040       000005  = 050
  LATCH1  = 000       SM1     = 000       000000  = 005
  000001  = 010       000002  = 025       000003  = 035
  000004  = 045       000005  = 055       LATCH2  = 000
  SM2     = 000       RCOND   = 120       RC      = 040
  RST     = 2         RSTS    = 0         RM00    = 377
  RM01    = 202       RM02    = 003       RM03    = 374
  000000  = 226       000001  = 035       000000  = 001
  000001  = 000       000000  = 000       000001  = 000
  000000  = 000       000001  = 000       RFREQ   = 00
  CFREQ   = 214       CEM0    = 000
*****************************************************

SWITCH INTERRUPT
    EXECUT = ZERO
  RI      = 130010    EXAS    = 1         EXSI    = 0
  MISEQ   = 260       PARADO  = 0         OVFLG   = 0
  CEME    = 002       RSEL    = 01        RSAID   = 226
  000000  = 000       000001  = 010       000002  = 020
  000003  = 030       000004  = 040       000005  = 050
  LATCH1  = 000       SM1     = 000       000000  = 005
  000001  = 010       000002  = 025       000003  = 035
  000004  = 045       000005  = 055       LATCH2  = 000
  SM2     = 000       RCOND   = 120       RC      = 040
  RST     = 2         RSTS    = 0         RM00    = 377
  RM01    = 202       RM02    = 003       RM03    = 374
  000000  = 226       000001  = 035       000000  = 001
  000001  = 000       000000  = 000       000001  = 000
  000000  = 000       000001  = 000       RFREQ   = 00
  CFREQ   = 214       CEM0    = 000
 *****************************************************
 ABEL CYCLE   21                TRUE LABELS                CLOCK TIME =    11
                                    /RELM(0)/
                                    /RELM(0)/
                                    /RELM(0)/
                                    /RELM(0)*EXECU≥+T
  RI      = 130010    EXAS    = 1         EXSI    = 1
  MISEQ   = 260       PARADO  = 1         OVFLG   = 0
  CEME    = 002       RSEL    = 01        RSAID   = 226
  000000  = 000       000001  = 010       000002  = 020
  000003  = 030       000004  = 040       000005  = 050
  LATCH1  = 000       SM1     = 000       000000  = 005
  000001  = 010       000002  = 025       000003  = 035
  000004  = 045       000005  = 055       LATCH2  = 000
  SM2     = 000       RCOND   = 120       RC      = 040
  RST     = 2         RSTS    = 0         RM00    = 377
  RM01    = 202       RM02    = 003       RM03    = 374
  000000  = 226       000001  = 035       000000  = 001
  000001  = 000       000000  = 000       000001  = 000
  000000  = 000       000001  = 000       RFREQ   = 00
  CFREQ   = 214       CEM0    = 000
```

```
*****************************************************
LABEL CYCLE   23                TRUE LABELS              CLOCK TIME =   12
                               /RELM(0)/
                               /RELM(0)*EXSI/
                               /RELM(0)/
                               /RELM(0)/
                               /RELM(0)*EXECU≥+T
  RI      = 010020    EXAS    = 0         EXSI    = 0
  MISEQ   = 021       PARADO  = 1         OVFLG   = 0
  CEME    = 002       RSEL    = 01        RSAID   = 226
  000000  = 000       000001  = 010       000002  = 020
  000003  = 030       000004  = 040       000005  = 050
  LATCH1  = 000       SM1     = 000       000000  = 005
  000001  = 010       000002  = 025       000003  = 035
  000004  = 045       000005  = 055       LATCH2  = 000
  SM2     = 000       RCOND   = 120       RC      = 040
  RST     = 2         RSTS    = 0         RM00    = 377
  RM01    = 202       RM02    = 003       RM03    = 374
  000000  = 226       000001  = 035       000000  = 001
  000001  = 000       000000  = 000       000001  = 000
  000000  = 000       000001  = 000       RFREQ   = 00
  CFREQ   = 214       CEM0    = 000
*****************************************************
LABEL CYCLE   25                TRUE LABELS              CLOCK TIME =   13
                               /RELM(0)/
                               /RELM(0)*CRSAID≥/
                               /RELM(0)/
                               /RELM(0)/
                               /RELM(0)*EXECU≥+T
  RI      = 010020    EXAS    = 0         EXSI    = 0
  MISEQ   = 020       PARADO  = 0         OVFLG   = 0
  CEME    = 002       RSEL    = 01        RSAID   = 226
  000000  = 000       000001  = 010       000002  = 020
  000003  = 030       000004  = 040       000005  = 050
  LATCH1  = 000       SM1     = 000       000000  = 005
  000001  = 010       000002  = 020       000003  = 035
  000004  = 045       000005  = 055       LATCH2  = 000
  SM2     = 000       RCOND   = 120       RC      = 040
  RST     = 2         RSTS    = 0         RM00    = 377
  RM01    = 202       RM02    = 003       RM03    = 374
  000000  = 226       000001  = 035       000000  = 001
  000001  = 000       000000  = 000       000001  = 000
  000000  = 000       000001  = 000       RFREQ   = 00
  CFREQ   = 214       CEM0    = 000
*****************************************************

SWITCH INTERRUPT
    EXECUT = ZERO
  RI      = 010020    EXAS    = 1         EXSI    = 0
  MISEQ   = 020       PARADO  = 0         OVFLG   = 0
  CEME    = 002       RSEL    = 01        RSAID   = 226
  000000  = 000       000001  = 010       000002  = 020
  000003  = 030       000004  = 040       000005  = 050
  LATCH1  = 000       SM1     = 000       000000  = 005
  000001  = 010       000002  = 020       000003  = 035
  000004  = 045       000005  = 055       LATCH2  = 000
  SM2     = 000       RCOND   = 120       RC      = 040
  RST     = 2         RSTS    = 0         RM00    = 377
  RM01    = 202       RM02    = 003       RM03    = 374
  000000  = 226       000001  = 035       000000  = 001
  000001  = 000       000000  = 000       000001  = 000
```

```
  000000 = 000       000001 = 000       RFREQ  = 00
  CFREQ  = 214       CEMO   = 000
*****************************************************
LABEL CYCLE   27                TRUE LABELS              CLOCK TIME =   14
                                /RELM(0)/
                                /RELM(0)/
                                /RELM(0)/
                                /RELM(0)*EXECU≥+T
  RI     = 010020    EXAS   = 1         EXSI   = 1
  MISEQ  = 020       PARADD = 1         OVFLG  = 0
  CEME   = 002       RSEL   = 01        RSAID  = 226
  000000 = 000       000001 = 010       000002 = 020
  000003 = 030       000004 = 040       000005 = 050
  LATCH1 = 000       SM1    = 000       000000 = 005
  000001 = 010       000002 = 020       000003 = 035
  000004 = 045       000005 = 055       LATCH2 = 000
  SM2    = 000       RCOND  = 120       RC     = 040
  RST    = 2         RSTS   = 0         RMDD   = 377
  RMD1   = 202       RMD2   = 003       RMD3   = 374
  000000 = 226       000001 = 035       000000 = 001
  000001 = 000       000000 = 000       000001 = 000
  000000 = 000       000001 = 000       RFREQ  = 00
  CFREQ  = 214       CEMO   = 000
*****************************************************
LABEL CYCLE   29                TRUE LABELS              CLOCK TIME =   15
                                /RELM(0)/
                                /RELM(0)*EXSI/
                                /RELM(0)/
                                /RELM(0)/
                                /RELM(0)*EXECU≥+T
  RI     = 030001    EXAS   = 0         EXSI   = 0
  MISEQ  = 061       PARADD = 1         OVFLG  = 0
  CEME   = 002       RSEL   = 01        RSAID  = 226
  000000 = 000       000001 = 010       000002 = 020
  000003 = 030       000004 = 040       000005 = 050
  LATCH1 = 000       SM1    = 000       000000 = 005
  000001 = 010       000002 = 020       000003 = 035
  000004 = 045       000005 = 055       LATCH2 = 000
  SM2    = 000       RCOND  = 120       RC     = 040
  RST    = 2         RSTS   = 0         RMDD   = 377
  RMD1   = 202       RMD2   = 003       RMD3   = 374
  000000 = 226       000001 = 035       000000 = 001
  000001 = 000       000000 = 000       000001 = 000
  000000 = 000       000001 = 000       RFREQ  = 00
  CFREQ  = 214       CEMO   = 000
*****************************************************
LABEL CYCLE   31                TRUE LABELS              CLOCK TIME =   16
                                /RELM(0)/
                                /RELM(0)*CCFB≥/
                                /RELM(0)/
                                /RELM(0)/
                                /RELM(0)*EXECU≥+T
  RI     = 030001    EXAS   = 0         EXSI   = 0
  MISEQ  = 060       PARADD = 0         OVFLG  = 0
  CEME   = 001       RSEL   = 01        RSAID  = 226
  000000 = 000       000001 = 010       000002 = 020
  000003 = 030       000004 = 040       000005 = 050
  LATCH1 = 000       SM1    = 000       000000 = 005
  000001 = 010       000002 = 020       000003 = 035
  000004 = 045       000005 = 055       LATCH2 = 000
  SM2    = 000       RCOND  = 120       RC     = 040
```

```
RST     = 2         RSTS    = 0         RMDO    = 377
RMDI    = 202       RMD2    = 003       RMD3    = 374
000000 = 226        000001 = 035        000000 = 001
000001 = 000        000000 = 000        000001 = 000
000000 = 000        000001 = 000        RFREQ   = 00
CFREQ   = 214       CEMD    = 000
**************************************************

SWITCH INTERRUPT
    EXECUT = ZERO
RI      = 030001    EXAS    = 1         EXSI    = 0
MISEQ   = 060       PARADD  = 0         DVFLG   = 0
CEME    = 001       RSEL    = 01        RSAID   = 226
000000 = 000        000001 = 010        000002 = 020
000003 = 030        000004 = 040        000005 = 050
LATCH1 = 000        SM1     = 000       000000 = 005
000001 = 010        000002 = 020        000003 = 035
000004 = 045        000005 = 055        LATCH2 = 000
SM2     = 000       RCOND   = 120       RC      = 040
RST     = 2         RSTS    = 0         RMDO    = 377
RMDI    = 202       RMD2    = 003       RMD3    = 374
000000 = 226        000001 = 035        000000 = 001
000001 = 000        000000 = 000        000001 = 000
000000 = 000        000001 = 000        RFREQ   = 00
CFREQ   = 214       CEMD    = 000
**************************************************
LABEL CYCLE   33                TRUE LABELS                CLOCK TIME =   17
                              /RELM(0)/
                              /RELM(0)/
                              /RELM(0)/
                              /RELM(0)*EXECUTE+T
RI      = 030001    EXAS    = 1         EXSI    = 1
MISEQ   = 060       PARADD  = 1         DVFLG   = 0
CEME    = 001       RSEL    = 01        RSAID   = 226
000000 = 000        000001 = 010        000002 = 020
000003 = 030        000004 = 040        000005 = 050
LATCH1 = 000        SM1     = 000       000000 = 005
000001 = 010        000002 = 020        000003 = 035
000004 = 045        000005 = 055        LATCH2 = 000
SM2     = 000       RCOND   = 120       RC      = 040
RST     = 2         RSTS    = 0         RMDO    = 377
RMDI    = 202       RMD2    = 003       RMD3    = 374
000000 = 226        000001 = 035        000000 = 001
000001 = 000        000000 = 000        000001 = 000
000000 = 000        000001 = 000        RFREQ   = 00
CFREQ   = 214       CEMD    = 000
**************************************************
LABEL CYCLE   35                TRUE LABELS                CLOCK TIME =   18
                              /RELM(0)/
                              /RELM(0)*EXSI/
                              /RELM(0)/
                              /RELM(0)/
                              /RELM(0)*EXECUTE+T
RI      = 000000    EXAS    = 0         EXSI    = 0
MISEQ   = 001       PARADD  = 1         DVFLG   = 0
CEME    = 001       RSEL    = 01        RSAID   = 226
000000 = 000        000001 = 010        000002 = 020
000003 = 030        000004 = 040        000005 = 050
LATCH1 = 000        SM1     = 000       000000 = 005
000001 = 010        000002 = 020        000003 = 035
000004 = 045        000005 = 055        LATCH2 = 000
```

```
SM2     = 000       RCOND   = 120       RC      = 040
RST     = 2         RSTS    = 0         RMD0    = 377
RMD1    = 202       RMD2    = 003       RMD3    = 374
000000 = 226        000001 = 035        000000 = 001
000001 = 000        000000 = 000        000001 = 000
000000 = 000        000001 = 000        RFREQ   = 00
CFREQ   = 214       CEMD    = 000
*************************************************************
LABEL CYCLE    37                 TRUE LABELS                  CLOCK TIME =    19
                                 /RELM(0)/
                                 /RELM(0)*CCEB*ICE
                                 /RELM(0)*CRSAID2/
                                 /RELM(0)/
                                 /RELM(0)/
                                 /RELM(0)*EXECU2+T
RI      = 000000    EXAS    = 0         EXSI    = 0
MISEQ   = 000       PARADD  = 0         DVFLG   = 0
CEME    = 002       RSEL    = 01        RSAID   = 010
000000 = 000        000001 = 010        000002 = 020
000003 = 030        000004 = 040        000005 = 050
LATCH1 = 000        SM1     = 000       000000 = 005
000001 = 010        000002 = 020        000003 = 035
000004 = 045        000005 = 055        LATCH2 = 000
SM2     = 000       RCOND   = 120       RC      = 040
RST     = 2         RSTS    = 0         RMD0    = 377
RMD1    = 202       RMD2    = 003       RMD3    = 374
000000 = 226        000001 = 035        000000 = 001
000001 = 000        000000 = 000        000001 = 000
000000 = 000        000001 = 000        RFREQ   = 00
CFREQ   = 214       CEMD    = 000
*************************************************************

SWITCH INTERRUPT
    EXECUT = ZERO
RI      = 000000    EXAS    = 1         EXSI    = 0
MISEQ   = 000       PARADD  = 0         DVFLG   = 0
CEME    = 002       RSEL    = 01        RSAID   = 010
000000 = 000        000001 = 010        000002 = 020
000003 = 030        000004 = 040        000005 = 050
LATCH1 = 000        SM1     = 000       000000 = 005
000001 = 010        000002 = 020        000003 = 035
000004 = 045        000005 = 055        LATCH2 = 000
SM2     = 000       RCOND   = 120       RC      = 040
RST     = 2         RSTS    = 0         RMD0    = 377
RMD1    = 202       RMD2    = 003       RMD3    = 374
000000 = 226        000001 = 035        000000 = 001
000001 = 000        000000 = 000        000001 = 000
000000 = 000        000001 = 000        RFREQ   = 00
CFREQ   = 214       CEMD    = 000
*************************************************************
LABEL CYCLE    39                 TRUE LABELS                  CLOCK TIME =    20
                                 /RELM(0)/
                                 /RELM(0)/
                                 /RELM(0)/
                                 /RELM(0)*EXECU2+T
RI      = 000000    EXAS    = 1         EXSI    = 1
MISEQ   = 000       PARADD  = 1         DVFLG   = 0
CEME    = 002       RSEL    = 01        RSAID   = 010
000000 = 000        000001 = 010        000002 = 020
000003 = 030        000004 = 040        000005 = 050
LATCH1 = 000        SM1     = 000       000000 = 005
```

```
   000001 = 010        000002 = 020        000003 = 035
   000004 = 045        000005 = 055        LATCH2 = 000
   SM2    = 000        RCOND  = 120        RC     = 040
   RST    = 2          RSTS   = 0          RM00   = 377
   RM01   = 202        RM02   = 003        RM03   = 374
   000000 = 226        000001 = 035        000000 = 001
   000001 = 000        000000 = 000        000001 = 000
   000000 = 000        000001 = 000        RFREQ  = 00
   CFREQ  = 214        CEM0   = 000
**************************************************
LABEL CYCLE   41                TRUE LABELS              CLOCK TIMF =   21
                               /RELM(0)/
                               /RELM(0)*EXSI/
                               /RELM(0)/
                               /RELM(0)/
                               /RELM(0)*EXECU≥+T
   RI     = 120000    EXAS   = 0          EXSI   = 0
   MISEQ  = 241       PARADO = 1          DVFLG  = 0
   CEME   = 002       RSEL   = 01         RSAID  = 010
   000000 = 000       000001 = 010        000002 = 020
   000003 = 030       000004 = 040        000005 = 050
   LATCH1 = 000       SM1    = 000        000000 = 005
   000001 = 010       000002 = 020        000003 = 035
   000004 = 045       000005 = 055        LATCH2 = 000
   SM2    = 000       RCOND  = 120        RC     = 040
   RST    = 2         RSTS   = 0          RM00   = 377
   RM01   = 202       RM02   = 003        RM03   = 374
   000000 = 226       000001 = 035        000000 = 001
   000001 = 000       000000 = 000        000001 = 000
   000000 = 000       000001 = 000        RFREQ  = 00
   CFREQ  = 214       CEM0   = 000
**************************************************
LABEL CYCLE   43                TRUE LABELS              CLOCK TIMF =   22
                               /RELM(0)/
                               /RELM(0)*CRSAID≥/
                               /RELM(0)/
                               /RELM(0)/
                               /RELM(0)*EXECU≥+T
   RI     = 120000    EXAS   = 0          EXSI   = 0
   MISEQ  = 240       PARADO = 0          DVFLG  = 0
   CEME   = 002       RSEL   = 01         RSAID  = 020
   000000 = 000       000001 = 010        000002 = 020
   000003 = 030       000004 = 040        000005 = 050
   LATCH1 = 000       SM1    = 000        000000 = 005
   000001 = 010       000002 = 020        000003 = 035
   000004 = 045       000005 = 055        LATCH2 = 000
   SM2    = 000       RCOND  = 120        RC     = 040
   RST    = 2         RSTS   = 0          RM00   = 377
   RM01   = 202       RM02   = 003        RM03   = 374
   000000 = 226       000001 = 035        000000 = 001
   000001 = 000       000000 = 000        000001 = 000
   000000 = 000       000001 = 000        RFREQ  = 00
   CFREQ  = 214       CEM0   = 000
**************************************************

SWITCH INTERRUPT
    EXECUT = ZERO
   RI     = 120000    EXAS   = 1          EXSI   = 0
   MISEQ  = 240       PARADO = 0          DVFLG  = 0
   CEME   = 002       RSEL   = 01         RSAID  = 020
   000000 = 000       000001 = 010        000002 = 020
```

```
  000003 = 030        000004 = 040        000005 = 050
  LATCH1 = 000        SM1    = 000        000000 = 005
  000001 = 010        000002 = 020        000003 = 035
  000004 = 045        000005 = 055        LATCH2 = 000
  SM2    = 000        RCOND  = 120        RC     = 040
  RST    = 2          RSTS   = 0          RM00   = 377
  RM01   = 202        RM02   = 003        RM03   = 374
  000000 = 226        000001 = 035        000000 = 001
  000001 = 000        000000 = 000        000001 = 000
  000000 = 000        000001 = 000        RFREQ  = 00
  CFREQ  = 214        CEMO   = 000
**************************************************
LABEL CYCLE   45              TRUE LABELS              CLOCK TIME =   23
                             /RELM(0)/
                             /RELM(0)/
                             /RELM(0)/
                             /RELM(0)*EXECU2+T
  RI     = 120000    EXAS   = 1          EXSI   = 1
  MISEQ  = 240       PARADO = 1          OVFLG  = 0
  CEME   = 002       RSEL   = 01         RSAID  = 020
  000000 = 000       000001 = 010        000002 = 020
  000003 = 030       000004 = 040        000005 = 050
  LATCH1 = 000       SM1    = 000        000000 = 005
  000001 = 010       000002 = 020        000003 = 035
  000004 = 045       000005 = 055        LATCH2 = 000
  SM2    = 000       RCOND  = 120        RC     = 040
  RST    = 2         RSTS   = 0          RM00   = 377
  RM01   = 202       RM02   = 003        RM03   = 374
  000000 = 226       000001 = 035        000000 = 001
  000001 = 000       000000 = 000        000001 = 000
  000000 = 000       000001 = 000        RFREQ  = 00
  CFREQ  = 214       CEMO   = 000
**************************************************
LABEL CYCLE   47              TRUE LABELS              CLOCK TIME =   24
                             /RELM(0)/
                             /RELM(0)*EXSI/
                             /RELM(0)/
                             /RELM(0)/
                             /RELM(0)*EXECU2+T
  RI     = 040005    EXAS   = 0          EXSI   = 0
  MISEQ  = 101       PARADO = 1          OVFLG  = 0
  CEME   = 002       RSEL   = 01         RSAID  = 020
  000000 = 000       000001 = 010        000002 = 020
  000003 = 030       000004 = 040        000005 = 050
  LATCH1 = 000       SM1    = 000        000000 = 005
  000001 = 010       000002 = 020        000003 = 035
  000004 = 045       000005 = 055        LATCH2 = 000
  SM2    = 000       RCOND  = 120        RC     = 040
  RST    = 2         RSTS   = 0          RM00   = 377
  RM01   = 202       RM02   = 003        RM03   = 374
  000000 = 226       000001 = 035        000000 = 001
  000001 = 000       000000 = 000        000001 = 000
  000000 = 000       000001 = 000        RFREQ  = 00
  CFREQ  = 214       CEMO   = 000
**************************************************
LABEL CYCLE   49              TRUE LABELS              CLOCK TIME =   25
                             /RELM(0)/
                             /RELM(0)/
                             /(RELM(0)+CRCB)*(
                             /RELM(0)/
                             /RELM(0)*EXECU2+T
```

```
   RI      = 040005    EXAS    = 0         EXSI    = 0
   MISEQ   = 100       PARADD  = 0         DVFLG   = 0
   CEME    = 002       RSEL    = 01        RSAID   = 020
   000000  = 000       000001  = 010       000002  = 020
   000003  = 030       000004  = 040       000005  = 050
   LATCH1  = 000       SM1     = 000       000000  = 005
   000001  = 010       000002  = 020       000003  = 035
   000004  = 045       000005  = 055       LATCH2  = 000
   SM2     = 000       RCOND   = 000       RC      = 005
   RST     = 2         RSTS    = 0         RMD0    = 377
   RMD1    = 202       RMD2    = 003       RMD3    = 374
   000000  = 226       000001  = 035       000000  = 001
   000001  = 000       000000  = 000       000001  = 000
   000000  = 000       000001  = 000       RFREQ   = 00
   CFREQ   = 214       CEMD    = 000
*****************************************************

SWITCH INTERRUPT
     EXECUT = ZERO
   RI      = 040005    EXAS    = 1         EXSI    = 0
   MISEQ   = 100       PARADD  = 0         DVFLG   = 0
   CEME    = 002       RSEL    = 01        RSAID   = 020
   000000  = 000       000001  = 010       000002  = 020
   000003  = 030       000004  = 040       000005  = 050
   LATCH1  = 000       SM1     = 000       000000  = 005
   000001  = 010       000002  = 020       000003  = 035
   000004  = 045       000005  = 055       LATCH2  = 000
   SM2     = 000       RCOND   = 000       RC      = 005
   RST     = 2         RSTS    = 0         RMD0    = 377
   RMD1    = 202       RMD2    = 003       RMD3    = 374
   000000  = 226       000001  = 035       000000  = 001
   000001  = 000       000000  = 000       000001  = 000
   000000  = 000       000001  = 000       RFREQ   = 00
   CFREQ   = 214       CEMD    = 000
*****************************************************
LABEL CYCLE    51                TRUE LABELS              CLOCK TIME =    26
                                /RELM(0)/
                                /RELM(0)/
                                /RELM(0)/
                                /RELM(0)*EXECU≥+T
   RI      = 040005    EXAS    = 1         EXSI    = 1
   MISEQ   = 100       PARADD  = 1         DVFLG   = 0
   CEME    = 002       RSEL    = 01        RSAID   = 020
   000000  = 000       000001  = 010       000002  = 020
   000003  = 030       000004  = 040       000005  = 050
   LATCH1  = 000       SM1     = 000       000000  = 005
   000001  = 010       000002  = 020       000003  = 035
   000004  = 045       000005  = 055       LATCH2  = 000
   SM2     = 000       RCOND   = 000       RC      = 005
   RST     = 2         RSTS    = 0         RMD0    = 377
   RMD1    = 202       RMD2    = 003       RMD3    = 374
   000000  = 226       000001  = 035       000000  = 001
   000001  = 000       000000  = 000       000001  = 000
   000000  = 000       000001  = 000       RFREQ   = 00
   CFREQ   = 214       CEMD    = 000
*****************************************************
LABEL CYCLE    53                TRUE LABELS              CLOCK TIME =    27
                                /RELM(0)/
                                /RELM(0)*EXSI/
                                /RELM(0)/
                                /RELM(0)/
```

```
                                    /RELM(0)*EXECU≥+T
  RI      = 050000    EXAS    = 0         EXSI    = 0
  MISEQ   = 121       PARADO  = 1         DVFLG   = 0
  CEME    = 002       RSEL    = 01        RSAID   = 020
  000000  = 000       000001  = 010       000002  = 020
  000003  = 030       000004  = 040       000005  = 050
  LATCH1  = 000       SM1     = 000       000000  = 005
  000001  = 010       000002  = 020       000003  = 035
  000004  = 045       000005  = 055       LATCH2  = 000
  SM2     = 000       RCOND   = 000       RC      = 005
  RST     = 2         RSTS    = 0         RM00    = 377
  RM01    = 202       RM02    = 003       RM03    = 374
  000000  = 226       000001  = 035       000000  = 001
  000001  = 000       000000  = 000       000001  = 000
  000000  = 000       000001  = 000       RFREQ   = 00
  CFREQ   = 214       CEM0    = 000
**************************************************
LABEL CYCLE   55                TRUE LABELS                CLOCK TIMF =   28
                                    /RELM(0)/
                                    /RELM(0)/
                                    /RELM(0)*CSTB≥/
                                    /RELM(0)/
                                    /RELM(0)*EXECU≥+T
  RI      = 050000    EXAS    = 0         EXSI    = 0
  MISEQ   = 120       PARADO  = 0         DVFLG   = 0
  CEME    = 002       RSEL    = 01        RSAID   = 020
  000000  = 000       000001  = 010       000002  = 020
  000003  = 030       000004  = 040       000005  = 050
  LATCH1  = 000       SM1     = 000       000000  = 005
  000001  = 010       000002  = 020       000003  = 035
  000004  = 045       000005  = 055       LATCH2  = 000
  SM2     = 000       RCOND   = 000       RC      = 005
  RST     = 0         RSTS    = 0         RM00    = 377
  RM01    = 202       RM02    = 003       RM03    = 374
  000000  = 226       000001  = 035       000000  = 001
  000001  = 000       000000  = 000       000001  = 000
  000000  = 000       000001  = 000       RFREQ   = 00
  CFREQ   = 214       CEM0    = 000
**************************************************

SWITCH INTERRUPT
    EXECUT = ZERO
  RI      = 050000    EXAS    = 1         EXSI    = 0
  MISEQ   = 120       PARADO  = 0         DVFLG   = 0
  CEME    = 002       RSEL    = 01        RSAID   = 020
  000000  = 000       000001  = 010       000002  = 020
  000003  = 030       000004  = 040       000005  = 050
  LATCH1  = 000       SM1     = 000       000000  = 005
  000001  = 010       000002  = 020       000003  = 035
  000004  = 045       000005  = 055       LATCH2  = 000
  SM2     = 000       RCOND   = 000       RC      = 005
  RST     = 0         RSTS    = 0         RM00    = 377
  RM01    = 202       RM02    = 003       RM03    = 374
  000000  = 226       000001  = 035       000000  = 001
  000001  = 000       000000  = 000       000001  = 000
  000000  = 000       000001  = 000       RFREQ   = 00
  CFREQ   = 214       CEM0    = 000
**************************************************
LABEL CYCLE   57                TRUE LABELS                CLOCK TIMF =   29
                                    /RELM(0)/
                                    /RELM(0)/
```

```
                                /RELM(0)/
                                /RELM(0)*EXECU≥+T
  RI      = 050000    EXAS    = 1         EXSI    = 1
  MISEQ   = 120       PARADO  = 1         DVFLG   = 0
  CEME    = 002       RSEL    = 01        RSAID   = 020
  000000  = 000       000001  = 010       000002  = 020
  000003  = 030       000004  = 040       000005  = 050
  LATCH1  = 000       SM1     = 000       000000  = 005
  000001  = 010       000002  = 020       000003  = 035
  000004  = 045       000005  = 055       LATCH2  = 000
  SM2     = 000       RCOND   = 000       RC      = 005
  RST     = 0         RSTS    = 0         RM00    = 377
  RM01    = 202       RM02    = 003       RM03    = 374
  000000  = 226       000001  = 035       000000  = 001
  000001  = 000       000000  = 000       000001  = 000
  000000  = 000       000001  = 000       RFREQ   = 00
  CFREQ   = 214       CEM0    = 000
**************************************************.
LABEL CYCLE   59                TRUE LABELS                CLOCK TIME =   30
                                /RELM(0)/
                                /RELM(0)*EXSI/
                                /RELM(0)/
                                /RELM(0)/
                                /RELM(0)*EXECU≥+T
  RI      = 060010    EXAS    = 0         EXSI    = 0
  MISEQ   = 141       PARADO  = 1         DVFLG   = 0
  CEME    = 002       RSEL    = 01        RSAID   = 020
  000000  = 000       000001  = 010       000002  = 020
  000003  = 030       000004  = 040       000005  = 050
  LATCH1  = 000       SM1     = 000       000000  = 005
  000001  = 010       000002  = 020       000003  = 035
  000004  = 045       000005  = 055       LATCH2  = 000
  SM2     = 000       RCOND   = 000       RC      = 005
  RST     = 0         RSTS    = 0         RM00    = 377
  RM01    = 202       RM02    = 003       RM03    = 374
  000000  = 226       000001  = 035       000000  = 001
  000001  = 000       000000  = 000       000001  = 000
  000000  = 000       000001  = 000       RFREQ   = 00
  CFREQ   = 214       CEM0    = 000
**************************************************
LABEL CYCLE   61                TRUE LABELS                CLOCK TIME =   31
                                /RELM(0)/
                                /RELM(0)/
                                /RELM(0)/
                                /RELM(0)*CFB≥/
                                /RELM(0)*CFB≥+TRS
                                /RELM(0)*EXECU≥+T
  RI      = 060010    EXAS    = 0         EXSI    = 0
  MISEQ   = 142       PARADO  = 0         DVFLG   = 0
  CEME    = 002       RSEL    = 01        RSAID   = 020
  000000  = 000       000001  = 010       000002  = 020
  000003  = 030       000004  = 040       000005  = 050
  LATCH1  = 000       SM1     = 000       000000  = 005
  000001  = 010       000002  = 020       000003  = 035
  000004  = 045       000005  = 055       LATCH2  = 000
  SM2     = 000       RCOND   = 000       RC      = 005
  RST     = 0         RSTS    = 0         RM00    = 377
  RM01    = 202       RM02    = 003       RM03    = 374
  000000  = 226       000001  = 035       000000  = 001
  000001  = 000       000000  = 000       000001  = 000
  000000  = 000       000001  = 000       RFREQ   = 10
```

```
  CFREQ  = 000       CEMO   = 000
**************************************************
LABEL CYCLE   63                TRUE LABELS
                                /RELM(0)/
                                /RELM(0)/
                                /RELM(0)/
                                /RELM(0)*CFB≥/
                                /RELM(0)*CFB≥+TRS
                                /RELM(0)*EXECU≥+T
  RI      = 060010   EXAS   = 0         EXSI   = 0
  MISEQ   = 140      PARADD = 0         OVFLG  = 0
  CEME    = 002      RSEL   = 01        RSAID  = 020
  000000  = 000      000001 = 010       000002 = 020
  000003  = 030      000004 = 040       000005 = 050
  LATCH1  = 000      SM1    = 000       000000 = 005
  000001  = 010      000002 = 020       000003 = 035
  000004  = 045      000005 = 055       LATCH2 = 000
  SM2     = 000      RCOND  = 000       RC     = 005
  RST     = 0        RSTS   = 0         RMD0   = 377
  RMD1    = 202      RMD2   = 003       RMD3   = 374
  000000  = 226      000001 = 035       000000 = 001
  000001  = 000      000000 = 000       000001 = 000
  000000  = 000      000001 = 000       RFREQ  = 10
  CFREQ   = 010      CEMO   = 000
**************************************************

SWITCH INTERRUPT
    EXECUT = ZERO
  RI      = 060010   EXAS   = 1         EXSI   = 0
  MISEQ   = 140      PARADD = 0         OVFLG  = 0
  CEME    = 002      RSEL   = 01        RSAID  = 020
  000000  = 000      000001 = 010       000002 = 020
  000003  = 030      000004 = 040       000005 = 050
  LATCH1  = 000      SM1    = 000       000000 = 005
  000001  = 010      000002 = 020       000003 = 035
  000004  = 045      000005 = 055       LATCH2 = 000
  SM2     = 000      RCOND  = 000       RC     = 005
  RST     = 0        RSTS   = 0         RMD0   = 377
  RMD1    = 202      RMD2   = 003       RMD3   = 374
  000000  = 226      000001 = 035       000000 = 001
  000001  = 000      000000 = 000       000001 = 000
  000000  = 000      000001 = 000       RFREQ  = 10
  CFREQ   = 010      CEMO   = 000
**************************************************
LABEL CYCLE   65                TRUE LABELS                    CLOCK TIME =    33
                                /RELM(0)/
                                /RELM(0)/
                                /RELM(0)/
                                /RELM(0)*EXECU≥+T
  RI      = 060010   EXAS   = 1         EXSI   = 1
  MISEQ   = 140      PARADD = 1         OVFLG  = 0
  CEME    = 002      RSEL   = 01        RSAID  = 020
  000000  = 000      000001 = 010       000002 = 020
  000003  = 030      000004 = 040       000005 = 050
  LATCH1  = 000      SM1    = 000       000000 = 005
  000001  = 010      000002 = 020       000003 = 035
  000004  = 045      000005 = 055       LATCH2 = 000
  SM2     = 000      RCOND  = 000       RC     = 005
  RST     = 0        RSTS   = 0         RMD0   = 377
  RMD1    = 202      RMD2   = 003       RMD3   = 374
  000000  = 226      000001 = 035       000000 = 001
```

```
  000001 = 000      000000 = 000      000001 = 000
  000000 = 000      000001 = 000      RFREQ  = 10
  CFREQ  = 010      CEMO   = 000
*****************************************************
LABEL CYCLE   67                TRUE LABELS              CLOCK TIMF =   34
                                /RELM(0)/
                                /RELM(0)*EXSI/
                                /RELM(0)/
                                /RELM(0)/
                                /RELM(0)*EXECU2+T
  RI     = 070000   EXAS   = 0        EXSI   = 0
  MISEQ  = 161      PARADO = 1        OVFLG  = 0
  CEME   = 002      RSEL   = 01       RSAID  = 020
  000000 = 000      000001 = 010      000002 = 020
  000003 = 030      000004 = 040      000005 = 050
  LATCH1 = 000      SM1    = 000      000000 = 005
  000001 = 010      000002 = 020      000003 = 035
  000004 = 045      000005 = 055      LATCH2 = 000
  SM2    = 000      RCOND  = 000      RC     = 005
  RST    = 0        RSTS   = 0        RM00   = 377
  RM01   = 202      RM02   = 003      RM03   = 374
  000000 = 226      000001 = 035      000000 = 001
  000001 = 000      000000 = 000      000001 = 000
  000000 = 000      000001 = 000      RFREQ  = 10
  CFREQ  = 010      CEMO   = 000
*****************************************************
LABEL CYCLE   69                TRUE LABELS              CLOCK TIMF =   35
                                /RELM(0)/
                                /RELM(0)/
                                /RELM(0)/
                                /RELM(0)*EXECU2+T
  RI     = 070000   EXAS   = 0        EXSI   = 0
  MISEQ  = 160      PARADO = 0        OVFLG  = 0
  CEME   = 002      RSEL   = 01       RSAID  = 020
  000000 = 000      000001 = 010      000002 = 020
  000003 = 030      000004 = 040      000005 = 050
  LATCH1 = 000      SM1    = 000      000000 = 005
  000001 = 010      000002 = 020      000003 = 035
  000004 = 045      000005 = 055      LATCH2 = 000
  SM2    = 000      RCOND  = 000      RC     = 005
  RST    = 0        RSTS   = 0        RM00   = 377
  RM01   = 202      RM02   = 003      RM03   = 374
  000000 = 226      000001 = 035      000000 = 001
  000001 = 000      000000 = 000      000001 = 000
  000000 = 000      000001 = 000      RFREQ  = 10
  CFREQ  = 010      CEMO   = 001
*****************************************************

SWITCH INTERRUPT
    EXECUT = ZERO
  RI     = 070000   EXAS   = 1        EXSI   = 0
  MISEQ  = 160      PARADO = 0        OVFLG  = 0
  CEME   = 002      RSEL   = 01       RSAID  = 020
  000000 = 000      000001 = 010      000002 = 020
  000003 = 030      000004 = 040      000005 = 050
  LATCH1 = 000      SM1    = 000      000000 = 005
  000001 = 010      000002 = 020      000003 = 035
  000004 = 045      000005 = 055      LATCH2 = 000
  SM2    = 000      RCOND  = 000      RC     = 005
  RST    = 0        RSTS   = 0        RM00   = 377
  RM01   = 202      RM02   = 003      RM03   = 374
```

```
   000000 = 226        000001 = 035        000000 = 001
   000001 = 000        000000 = 000        000001 = 000
   000000 = 000        000001 = 000        RFREQ  = 10
   CFREQ  = 010        CEMO   = 001
*****************************************************
LABEL CYCLE   71                 TRUE LABELS              CLOCK TIME =   36
                                /RELM(0)/
                                /RELM(0)/
                                /RELM(0)/
                                /RELM(0)*EXECU≥+T
   RI     = 070000     EXAS   = 1          EXSI   = 1
   MISEQ  = 160        PARADD = 1          DVFLG  = 0
   CEME   = 002        RSEL   = 01         RSAID  = 020
   000000 = 000        000001 = 010        000002 = 020
   000003 = 030        000004 = 040        000005 = 050
   LATCH1 = 000        SM1    = 000        000000 = 005
   000001 = 010        000002 = 020        000003 = 035
   000004 = 045        000005 = 055        LATCH2 = 000
   SM2    = 000        RCOND  = 000        RC     = 005
   RST    = 0          RSTS   = 0          RMDD   = 377
   RMD1   = 202        RMD2   = 003        RMD3   = 374
   000000 = 226        000001 = 035        000000 = 001
   000001 = 000        000000 = 000        000001 = 000
   000000 = 000        000001 = 000        RFREQ  = 10
   CFREQ  = 010        CEMO   = 001
*****************************************************
LABEL CYCLE   73                 TRUE LABELS              CLOCK TIME =   37
                                /RELM(0)/
                                /RELM(0)*EXSI/
                                /RELM(0)/
                                /RELM(0)/
                                /RELM(0)*EXECU≥+T
   RI     = 100000     EXAS   = 0          EXSI   = 0
   MISEQ  = 201        PARADD = 1          DVFLG  = 0
   CEME   = 002        RSEL   = 01         RSAID  = 020
   000000 = 000        000001 = 010        000002 = 020
   000003 = 030        000004 = 040        000005 = 050
   LATCH1 = 000        SM1    = 000        000000 = 005
   000001 = 010        000002 = 020        000003 = 035
   000004 = 045        000005 = 055        LATCH2 = 000
   SM2    = 000        RCOND  = 000        RC     = 005
   RST    = 0          RSTS   = 0          RMDD   = 377
   RMD1   = 202        RMD2   = 003        RMD3   = 374
   000000 = 226        000001 = 035        000000 = 001
   000001 = 000        000000 = 000        000001 = 000
   000000 = 000        000001 = 000        RFREQ  = 10
   CFREQ  = 010        CEMO   = 001
*****************************************************
LABEL CYCLE   75                 TRUE LABELS              CLOCK TIME =   38
                                /RELM(0)/
                                /RELM(0)*CRSAID≥/
                                /RELM(0)/
                                /RELM(0)/
                                /RELM(0)*EXECU≥+T
   RI     = 100000     EXAS   = 0          EXSI   = 0
   MISEQ  = 200        PARADD = 0          DVFLG  = 0
   CEME   = 002        RSEL   = 01         RSAID  = 035
   000000 = 000        000001 = 010        000002 = 020
   000003 = 030        000004 = 040        000005 = 050
   LATCH1 = 000        SM1    = 000        000000 = 005
   000001 = 010        000002 = 020        000003 = 035
```

```
   000004 = 045        000005 = 055        LATCH2 = 000
   SM2    = 000        RCOND  = 000        RC     = 005
   RST    = 0          RSTS   = 0          RM00   = 377
   RM01   = 202        RM02   = 003        RM03   = 374
   000000 = 226        000001 = 035        000000 = 001
   000001 = 000        000000 = 000        000001 = 000
   000000 = 000        000001 = 000        RFREQ  = 10
   CFREQ  = 010        CEM0   = 001
**************************************************
LABEL CYCLE   77                TRUE LABELS                    CLOCK TIME =   39
                               /RELM(0)/
                               /RELM(0)/
                               /RELM(0)/
                               /RELM(0)*EXECU≥+T
   RI     = 100000    EXAS   = 0          EXSI   = 0
   MISEQ  = 200       PARADO = 1          OVFLG  = 0
   CEME   = 002       RSEL   = 01         RSAID  = 035
   000000 = 000       000001 = 010        000002 = 020
   000003 = 030       000004 = 040        000005 = 050
   LATCH1 = 000       SM1    = 000        000000 = 005
   000001 = 010       000002 = 020        000003 = 035
   000004 = 045       000005 = 055        LATCH2 = 000
   SM2    = 000       RCOND  = 000        RC     = 005
   RST    = 0         RSTS   = 0          RM00   = 377
   RM01   = 202       RM02   = 003        RM03   = 374
   000000 = 226       000001 = 035        000000 = 001
   000001 = 000       000000 = 000        000001 = 000
   000000 = 000       000001 = 000        RFREQ  = 10
   CFREQ  = 010       CEM0   = 001
**************************************************

                SIMULATION ENDS AFTER THE REQUIRED NUMBER OF LABEL CYCLES

        END OF CONTROL STATEMENTS.  SIMULATION TERMINATED.
```

1.3.2 - SIMULAÇÃO DAS OPERAÇÕES ASSOCIADAS ÃS ENTRADAS EXTERNAS VINDAS DO SISTEMA EM TESTE

Esta simulação tem por finalidade validar a operação do circuito do emulador de memórias de microcontrole, quando ele estiver com todas as condições iniciais carregadas e funcionando sob o controle das entradas externas, vindas do sistema em teste.

Como mostra a Figura A.2, as entradas vindas do sistema externo são simuladas por memórias endereçadas pelo registro contador SEQTET, e as saídas para o sistema externo, por registros. Sempre que um pulso do relógio, RSTS, é liberado para o sistema externo, o registro SEQTET é incrementado, de tal forma que novas entradas são enviadas para o emulador de memórias de microcontrole.

Os 11 "bits" mais significativos da memória MTET,simulam a entrada de endereço EM1, de 9 "bits", para o módulo de emulação 1 e os dois sinais de controle de "tri-state", ENM1AB e ENM1BB, para o mesmo módulo. Os 11 "bits" menos significativos simulam as entradas EM2, ENM2AB e ENM2BB para o módulo de emulação 2.

O barramento de endereço da memória de controle, do sistema em teste, é simulado pela memória EC de 9 "bits".

As 32 pontas de teste para monitoramento de estados de pontos arbitrários, do sistema em teste, são simuladas pela memória PT de 32 "bits".

As saídas das memórias de controle emuladas pelos módulos de emulação M1 e M2, endereçadas pelas entradas EM1 e EM2, são simuladas pelos registros SM1 e SM2. É bom frisar que o emulador de memórias de microcontrole é capaz de emular outras memórias, do sistema em teste, que não sejam memórias de microcontrole, ou seja, não é necessário que as entradas de endereçamento para os módulos de emulação sejam, obrigatoriamente, o barramento de endereços da memória de controle, EC

no caso. Como veremos, nesta simulação a entrada EM1 é o barramento de controle e a entrada EM2 não é, ou seja, o módulo de emulação 1 está emulando uma memória de controle, o que não acontece com o módulo de emulação M2 que poderia, por exemplo, estar emulando uma memória de mapeamento de código de operação do sistema em teste.

A saída do relógio controlado para o sistema em teste é simulada pelo registro RSTS. O relógio de entrada, como foi dito anteriormente, foi suposto como sendo o próprio relógio RELM do emulador.

Os valores em octal das entradas e o endereço correspondente do SEQTET, utilizados na simulação, estão listadas na Tabela A.3.

TABELA A.3

VALORES EM OCTAL DAS ENTRADAS UTILIZADAS NA SIMULAÇÃO
E CONTEÚDOS CORRESPONDENTES DOS MÓDULOS DE EMULAÇÃO M1 e M2

SEQTET	EC	EM1	ENM1AB	ENM1BB	EM2	ENM2AB	ENM2BB	PT0	PT1	PT2	PT3	M1	M2
0	0	0	0	0	0	0	0	377	202	001	374	00	05
1	1	1	0	0	1	0	0	005	210	377	177	10	15
2	2	2	0	0	2	0	0	377	377	377	377	20	25
3	3	3	0	0	2	0	0	123	277	162	234	30	35
4	4	4	0	0	3	0	0	345	311	022	234	40	45
5	5	5	0	0	4	0	0	000	000	000	000	50	55

As condições iniciais do registro contador e do registro de "status" do circuito de controle do relógio são carregadas pelas instruções CARC 4 (40004), CARS 2 (50002), o que corresponde a liberar o relógio RSTS, do sistema em teste, até que o endereço de controle 4 seja atingido.

A frequência de armazenamento dos pontos de teste foi definida como sendo 2 para esta simulação, ou seja, de dois em dois pulsos do relógio RSTS, os estados serão armazenados. Esta condição corresponde a carregar 16 no registro de frequência do circuito de monitoramento, o que é feito com a instrução CARF 16 (60016).

A execução destas instruções, comandadas pela chave EXECUTE, é concluída no ciclo de rótulo número 21, como mostra a listagem no final deste apêndice.

Neste ciclo de rótulo é comandada a execução da instrução EXEC (110000), que fará com que o emulador de memórias de microcontrole fique no modo comandado pelas entradas externas e libere o relógio RSTS, no modo especificado, para o sistema em teste.

A partir do ciclo de rótulo número 27, o relógio RSTS para o sistema em teste é liberado. SM1 e SM2, que correspondem às saídas para o sistema em teste, dos módulos de emulação 1 e 2, são carregados de acordo com a Tabela A.3. Existe o atraso de um pulso nesta saída devido ao fato de que, em CDL, toda memória só pode ser endereçada por registro, o que faz com que, embora não existindo no circuito do emulador, tenha que se colocar um registro na entrada de endereçamento das memórias M1 e M2 (Registros REM1 e REM2 na descrição do programa).

No ciclo de rótulo número 35, vemos que as memórias de monitoramento foram carregadas com os dados da memória PT para valores de SEQTET de 1 e 3, o que corresponde a armazenar os estados das pontas de teste de dois em dois pulsos do relógio, como desejado. Neste ciclo de rótulo, o registro RCOND de condição do circuito de controle do relógio, que armazena a todo pulso de relógio RSTS o valor do endereço de controle, torna-se igual ao endereço armazenado no registro contador RC, o que determina a parada do relógio RSTS, ficando o emulador de memórias de microcontrole parado, pronto para executar novas instruções vindas ou do painel ou do minicomputador.

Outras simulações com condições diferentes foram feitas, validando ainda mais o funcionamento do emulador de memórias de micro controle.

A seguir é apresentada a listagem da simulação descrita neste apêndice.

```
        $SIMULATE


        CDL VERSION 3                                          032979
                                              S I M U L A T I O N
        *OUTPUT    CLOCK(1)=RI,EXAS,EXSI,MISEQ,PARADD,DVFLG,CEME,RSEL,RSAID,
                   M1(0),M1(1),M1(2),M1(3),M1(4),M1(5),LATCH1,SM1,
                   M2(0),M2(1),M2(2),M2(3),M2(4),M2(5),LATCH2,SM2,
                   RCOND,RC,RST,RSTS,RM00,RM01,RM02,RM03,MM00(0),
                   MM00(1),MM01(0),MM01(1),MM02(0),MM02(1),MM03(0),
                   MM03(1),RFREQ,CFREQ,CEMO
        *SWITCH    1,LIGDES=LIG
        *SWITCH    1,RELOCB=ZERO
        *SWITCH    3,EXECUT=ZERO
        *SWITCH    11,EXECUT=ZERO
        *SWITCH    17,EXECUT=ZERO
        *SWITCH    25,EXECUT=ZERO
        *LOAD
MCONTR(0-1)=133661,137250,
MCONTR(20-21)=133661,132650,
MCONTR(40-41)=133661,133470,
MCONTR(60-61)=133661,133630,
MCONTR(100-101)=133661,123670,
```

```
MCONTR(120-121)=133661,113670,
MCONTR(140-142)=133661,33663,33670,
MCONTR(160-161)=133661,333670,
MCONTR(200-201)=133661,133650,
MCONTR(220-223)=133661,133673,171765,133660,
MCONTR(240-241)=133661,133250,
MCONTR(260-261)=133661,136650,
MCONTR(300-303)=133661,133673,171665,133660,
MCONTR(320)=133661,
MCONTR(340)=133661,
MCONTR(360)=133661,
        *LOAD
SEQTET=0, CEM0=377,
M1(0-5)=0,10,20,30,40,50,
M2(0-5)=5,15,25,35,45,55,
MTET(0-5)=0,20004,40010,60010,100014,120020,
EC(0-5)=0,1,2,3,4,5,
PT(0-6)=37740401774,542177577,37777777777,12345671234,74563211234,0,2,
CHAVES(1-5)=40004,50002,60016,110000,17000,
        *SIM      60,60
```

OUTPUT OF SIMULATION

```
SWITCH INTERRUPT
   LIGDES = LIG
  RI      = 000000    EXAS    = 0         EXSI    = 0
  MISEQ   = 000       PARADO  = 0         DVFLG   = 0
  CEME    = 000       RSEL    = 00        RSAID   = 000
  000000  = 000       000001  = 010       000002  = 020
  000003  = 030       000004  = 040       000005  = 050
  LATCH1  = 000       SM1     = 000       000000  = 005
  000001  = 015       000002  = 025       000003  = 035
  000004  = 045       000005  = 055       LATCH2  = 000
  SM2     = 020       RCOND   = 120       RC      = 040
  RST     = 2         RSTS    = 1         RMD0    = 065
  RMD1    = 041       RMD2    = 000       RMD3    = 000
  000000  = 226       000001  = 000       000000  = 001
  000001  = 000       000000  = 000       000001  = 000
  000000  = 000       000001  = 000       RFREQ   = 00
  CFREQ   = 214       CEM0    = 377
**************************************************
LABEL CYCLE    1                TRUE LABELS                CLOCK TIME =     1
                               /RELM(0)/
                               /RELM(0)/
                               /RELM(0)/
                               /RELM(0)*EXECU2+T
  RI      = 000000    EXAS    = 0         EXSI    = 0
  MISEQ   = 000       PARADO  = 1         DVFLG   = 0
  CEME    = 000       RSEL    = 00        RSAID   = 000
  000000  = 000       000001  = 010       000002  = 020
  000003  = 030       000004  = 040       000005  = 050
  LATCH1  = 000       SM1     = 000       000000  = 005
  000001  = 015       000002  = 025       000003  = 035
  000004  = 045       000005  = 055       LATCH2  = 000
  SM2     = 000       RCOND   = 120       RC      = 040
  RST     = 2         RSTS    = 0         RMD0    = 377
  RMD1    = 202       RMD2    = 003       RMD3    = 374
  000000  = 226       000001  = 000       000000  = 001
  000001  = 000       000000  = 000       000001  = 000
```

```
000000 = 000        000001 = 000        RFREQ  = 00
CFREQ  = 214        CEMO   = 377
*****************************************************

WITCH INTERRUPT
   EXECUT = ZERO
RI      = 000000    EXAS   = 1          EXSI   = 0
MISEQ   = 000       PARADO = 1          OVFLG  = 0
CEME    = 000       RSEL   = 00         RSAID  = 000
000000 = 000        000001 = 010        000002 = 020
000003 = 030        000004 = 040        000005 = 050
LATCH1 = 000        SM1    = 000        000000 = 005
000001 = 015        000002 = 025        000003 = 035
000004 = 045        000005 = 055        LATCH2 = 000
SM2     = 000       RCOND  = 120        RC     = 040
RST     = 2         RSTS   = 0          RMOO   = 377
RMO1    = 202       RMO2   = 003        RMO3   = 374
000000 = 226        000001 = 000        000000 = 001
000001 = 000        000000 = 000        000001 = 000
000000 = 000        000001 = 000        RFREQ  = 00
CFREQ  = 214        CEMO   = 377
*****************************************************
ABEL CYCLE    3                TRUE LABELS              CLOCK TIME =    2
                               /RELM(0)/
                               /RELM(0)/
                               /RELM(0)/
                               /RELM(0)*EXECUZ+T
RI      = 000000    EXAS   = 1          EXSI   = 1
MISEQ   = 000       PARADO = 1          OVFLG  = 0
CEME    = 000       RSEL   = 00         RSAID  = 000
000000 = 000        000001 = 010        000002 = 020
000003 = 030        000004 = 040        000005 = 050
LATCH1 = 000        SM1    = 000        000000 = 005
000001 = 015        000002 = 025        000003 = 035
000004 = 045        000005 = 055        LATCH2 = 000
SM2     = 000       RCOND  = 120        RC     = 040
RST     = 2         RSTS   = 0          RMOO   = 377
RMO1    = 202       RMO2   = 003        RMO3   = 374
000000 = 226        000001 = 000        000000 = 001
000001 = 000        000000 = 000        000001 = 000
000000 = 000        000001 = 000        RFREQ  = 00
CFREQ  = 214        CEMO   = 377
*****************************************************
ABEL CYCLE    5                TRUE LABELS              CLOCK TIME =    3
                               /RELM(0)/
                               /RELM(0)*EXSI/
                               /RELM(0)/
                               /RELM(0)/
                               /RELM(0)*EXECUZ+T
RI      = 040004    EXAS   = 0          EXSI   = 0
MISEQ   = 101       PARADO = 1          OVFLG  = 0
CEME    = 000       RSEL   = 00         RSAID  = 000
000000 = 000        000001 = 010        000002 = 020
000003 = 030        000004 = 040        000005 = 050
LATCH1 = 000        SM1    = 000        000000 = 005
000001 = 015        000002 = 025        000003 = 035
000004 = 045        000005 = 055        LATCH2 = 000
SM2     = 000       RCOND  = 120        RC     = 040
RST     = 2         RSTS   = 0          RMOO   = 377
RMO1    = 202       RMO2   = 003        RMO3   = 374
000000 = 226        000001 = 000        0000   = 001
```

```
   000001 = 000        000000 = 000        000001 = 000
   000000 = 000        000001 = 000        RFREQ  = 00
   CFREQ  = 214        CEMD   = 377
**************************************************
LABEL CYCLE    7                TRUE LABELS                CLOCK TIMF =     4
                              /RELM(0)/
                              /RELM(0)/
                              /(RELM(0)+CRCB)*(
                              /RELM(0)/
                              /RELM(0)*EXECUΣ+T
   RI     = 040004    EXAS   = 0          EXSI   = 0
   MISEQ  = 100       PARADD = 0          DVFLG  = 0
   CEME   = 000       RSEL   = 00         RSAID  = 000
   000000 = 000       000001 = 010        000002 = 020
   000003 = 030       000004 = 040        000005 = 050
   LATCH1 = 000       SM1    = 000        000000 = 005
   000001 = 015       000002 = 025        000003 = 035
   000004 = 045       000005 = 055        LATCH2 = 000
   SM2    = 000       RCOND  = 000        RC     = 004
   RST    = 2         RSTS   = 0          RMD0   = 377
   RMD1   = 202       RMD2   = 003        RMD3   = 374
   000000 = 226       000001 = 000        000000 = 001
   000001 = 000       000000 = 000        000001 = 000
   000000 = 000       000001 = 000        RFREQ  = 00
   CFREQ  = 214       CEMD   = 377
**************************************************

SWITCH INTERRUPT
    EXECUT = ZERO
   RI     = 040004    EXAS   = 1          EXSI   = 0
   MISEQ  = 100       PARADD = 0          DVFLG  = 0
   CEME   = 000       RSEL   = 00         RSAID  = 000
   000000 = 000       000001 = 010        000002 = 020
   000003 = 030       000004 = 040        000005 = 050
   LATCH1 = 000       SM1    = 000        000000 = 005
   000001 = 015       000002 = 025        000003 = 035
   000004 = 045       000005 = 055        LATCH2 = 000
   SM2    = 000       RCOND  = 000        RC     = 004
   RST    = 2         RSTS   = 0          RMD0   = 377
   RMD1   = 202       RMD2   = 003        RMD3   = 374
   000000 = 226       000001 = 000        000000 = 001
   000001 = 000       000000 = 000        000001 = 000
   000000 = 000       000001 = 000        RFREQ  = 00
   CFREQ  = 214       CEMD   = 377
**************************************************
LABEL CYCLE    9                TRUE LABELS                CLOCK TIMF =     5
                              /RELM(0)/
                              /RELM(0)/
                              /RELM(0)/
                              /RELM(0)*EXECUΣ+T
   RI     = 040004    EXAS   = 1          EXSI   = 1
   MISEQ  = 100       PARADD = 1          DVFLG  = 0
   CEME   = 000       RSEL   = 00         RSAID  = 000
   000000 = 000       000001 = 010        000002 = 020
   000003 = 030       000004 = 040        000005 = 050
   LATCH1 = 000       SM1    = 000        000000 = 005
   000001 = 015       000002 = 025        000003 = 035
   000004 = 045       000005 = 055        LATCH2 = 000
   SM2    = 000       RCOND  = 000        RC     = 004
   RST    = 2         RSTS   = 0          RMD0   = 377
   RMD1   = 202       RMD2   = 003        RMD3   = 374
```

```
   000000 = 226        000001 = 000        000000 = 001
   000001 = 000        000000 = 000        000001 = 000
   000000 = 000        000001 = 000        RFREQ  = 00
   CFREQ  = 214        CEMD   = 377
**************************************************
LABEL CYCLE   11                TRUE LABELS              CLOCK TIMF =     6
                               /RELM(0)/
                               /RELM(0)*EXSI/
                               /RELM(0)/
                               /RELM(0)/
                               /RELM(0)*EXECU≥+T
   RI     = 050002    EXAS   = 0          EXSI   = 0
   MISEQ  = 121       PARADD = 1          OVFLG  = 0
   CEME   = 000       RSEL   = 00         RSAID  = 000
   000000 = 000       000001 = 010        000002 = 020
   000003 = 030       000004 = 040        000005 = 050
   LATCH1 = 000       SM1    = 000        000000 = 005
   000001 = 015       000002 = 025        000003 = 035
   000004 = 045       000005 = 055        LATCH2 = 000
   SM2    = 000       RCOND  = 000        RC     = 004
   RST    = 2         RSTS   = 0          RMD0   = 377
   RMD1   = 202       RMD2   = 003        RMD3   = 374
   000000 = 226       000001 = 000        000000 = 001
   000001 = 000       000000 = 000        000001 = 000
   000000 = 000       000001 = 000        RFREQ  = 00
   CFREQ  = 214       CEMD   = 377
**************************************************
LABEL CYCLE   13                TRUE LABELS              CLOCK TIMF =     7
                               /RELM(0)/
                               /RELM(0)/
                               /RELM(0)*CSTB≥/
                               /RELM(0)/
                               /RELM(0)*EXECU≥+T
   RI     = 050002    EXAS   = 0          EXSI   = 0
   MISEQ  = 120       PARADD = 0          OVFLG  = 0
   CEME   = 000       RSEL   = 00         RSAID  = 000
   000000 = 000       000001 = 010        000002 = 020
   000003 = 030       000004 = 040        000005 = 050
   LATCH1 = 000       SM1    = 000        000000 = 005
   000001 = 015       000002 = 025        000003 = 035
   000004 = 045       000005 = 055        LATCH2 = 000
   SM2    = 000       RCOND  = 000        RC     = 004
   RST    = 2         RSTS   = 0          RMD0   = 377
   RMD1   = 202       RMD2   = 003        RMD3   = 374
   000000 = 226       000001 = 000        000000 = 001
   000001 = 000       000000 = 000        000001 = 000
   000000 = 000       000001 = 000        RFREQ  = 00
   CFREQ  = 214       CEMD   = 377
**************************************************

SWITCH INTERRUPT
    EXECUT = ZERO
   RI     = 050002    EXAS   = 1          EXSI   = 0
   MISEQ  = 120       PARADD = 0          OVFLG  = 0
   CEME   = 000       RSEL   = 00         RSAID  = 000
   000000 = 000       000001 = 010        000002 = 020
   000003 = 030       000004 = 040        000005 = 050
   LATCH1 = 000       SM1    = 000        000000 = 005
   000001 = 015       000002 = 025        000003 = 035
   000004 = 045       000005 = 055        LATCH2 = 000
   SM2    = 000       RCOND  = 000        RC     = 004
```

```
RST     = 2         RSTS    = 0         RM00    = 377
RM01    = 202       RM02    = 003       RM03    = 374
000000 = 226        000001 = 000        000000 = 001
000001 = 000        000000 = 000        000001 = 000
000000 = 000        000001 = 000        RFREQ   = 00
CFREQ   = 214       CEM0    = 377
*************************************************
LABEL CYCLE   15              TRUE LABELS          CLOCK TIME =    8
                            /RELM(0)/
                            /RELM(0)/
                            /RELM(0)/
                            /RELM(0)*EXECU≥+T
RI      = 050002    EXAS    = 1         EXSI    = 1
MISEQ   = 120       PARADD  = 1         DVFLG   = 0
CEME    = 000       RSEL    = 00        RSAID   = 000
000000 = 000        000001 = 010        000002 = 020
000003 = 030        000004 = 040        000005 = 050
LATCH1 = 000        SM1     = 000       000000 = 005
000001 = 015        000002 = 025        000003 = 035
000004 = 045        000005 = 055        LATCH2 = 000
SM2     = 000       RCOND   = 000       RC      = 004
RST     = 2         RSTS    = 0         RM00    = 377
RM01    = 202       RM02    = 003       RM03    = 374
000000 = 226        000001 = 000        000000 = 001
000001 = 000        000000 = 000        000001 = 000
000000 = 000        000001 = 000        RFREQ   = 00
CFREQ   = 214       CEM0    = 377
*************************************************
LABEL CYCLE   17              TRUE LABELS          CLOCK TIME =    9
                            /RELM(0)/
                            /RELM(0)*EXSI/
                            /RELM(0)/
                            /RELM(0)/
                            /RELM(0)*EXECU≥+T
RI      = 060016    EXAS    = 0         EXSI    = 0
MISEQ   = 141       PARADD  = 1         DVFLG   = 0
CEME    = 000       RSEL    = 00        RSAID   = 000
000000 = 000        000001 = 010        000002 = 020
000003 = 030        000004 = 040        000005 = 050
LATCH1 = 000        SM1     = 000       000000 = 005
000001 = 015        000002 = 025        000003 = 035
000004 = 045        000005 = 055        LATCH2 = 000
SM2     = 000       RCOND   = 000       RC      = 004
RST     = 2         RSTS    = 0         RM00    = 377
RM01    = 202       RM02    = 003       RM03    = 374
000000 = 226        000001 = 000        000000 = 001
000001 = 000        000000 = 000        000001 = 000
000000 = 000        000001 = 000        RFREQ   = 00
CFREQ   = 214       CEM0    = 377
*************************************************
LABEL CYCLE   19              TRUE LABELS          CLOCK TIME =   10
                            /RELM(0)/
                            /RELM(0)/
                            /RELM(0)/
                            /RELM(0)*CFB≥/
                            /RELM(0)*CFB≥+TRS
                            /RELM(0)*EXECU≥+T
RI      = 060016    EXAS    = 0         EXSI    = 0
MISEQ   = 142       PARADD  = 0         DVFLG   = 0
CEME    = 000       RSEL    = 00        RSAID   = 000
000000 = 000        000001 = 010        000002 = 020
```

```
  000003 = 030      000004 = 040      000005 = 050
  LATCH1 = 000      SM1    = 000      000000 = 005
  000001 = 015      000002 = 025      000003 = 035
  000004 = 045      000005 = 055      LATCH2 = 000
  SM2    = 000      RCOND  = 000      RC     = 004
  RST    = 2        RSTS   = 0        RM00   = 377
  RM01   = 202      RM02   = 003      RM03   = 374
  000000 = 226      000001 = 000      000000 = 001
  000001 = 000      000000 = 000      000001 = 000
  000000 = 000      000001 = 000      RFREQ  = 16
  CFREQ  = 000      CEM0   = 377
**************************************************

SWITCH INTERRUPT
    EXECUT = ZERO
  RI     = 060016   EXAS   = 1        EXSI   = 0
  MISEQ  = 142      PARADD = 0        DVFLG  = 0
  CEME   = 000      RSEL   = 00       RSAID  = 000
  000000 = 000      000001 = 010      000002 = 020
  000003 = 030      000004 = 040      000005 = 050
  LATCH1 = 000      SM1    = 000      000000 = 005
  000001 = 015      000002 = 025      000003 = 035
  000004 = 045      000005 = 055      LATCH2 = 000
  SM2    = 000      RCOND  = 000      RC     = 004
  RST    = 2        RSTS   = 0        RM00   = 377
  RM01   = 202      RM02   = 003      RM03   = 374
  000000 = 226      000001 = 000      000000 = 001
  000001 = 000      000000 = 000      000001 = 000
  000000 = 000      000001 = 000      RFREQ  = 16
  CFREQ  = 000      CEM0   = 377
**************************************************
LABEL CYCLE   21                TRUE LABELS              CLOCK TIME =   11
                               /RELM(0)/
                               /RELM(0)/
                               /RELM(0)/
                               /RELM(0)*CFB2/
                               /RELM(0)*CFB2+TRS
                               /RELM(0)*EXECU2+T
  RI     = 060016   EXAS   = 1        EXSI   = 1
  MISEQ  = 140      PARADD = 0        DVFLG  = 0
  CEME   = 000      RSEL   = 00       RSAID  = 000
  000000 = 000      000001 = 010      000002 = 020
  000003 = 030      000004 = 040      000005 = 050
  LATCH1 = 000      SM1    = 000      000000 = 005
  000001 = 015      000002 = 025      000003 = 035
  000004 = 045      000005 = 055      LATCH2 = 000
  SM2    = 000      RCOND  = 000      RC     = 004
  RST    = 2        RSTS   = 0        RM00   = 377
  RM01   = 202      RM02   = 003      RM03   = 374
  000000 = 226      000001 = 000      000000 = 001
  000001 = 000      000000 = 000      000001 = 000
  000000 = 000      000001 = 000      RFREQ  = 16
  CFREQ  = 016      CEM0   = 377
**************************************************
LABEL CYCLE   23                TRUE LABELS              CLOCK TIME =   12
                               /RELM(0)/
                               /RELM(0)*EXSI/
                               /RELM(0)/
                               /RELM(0)/
                               /RELM(0)*EXECU2+T
  RI     = 110000   EXAS   = 0        EXSI   = 0
```

```
   MISEQ   = 221        PARADO = 1          DVFLG   = 0
   CEME    = 000        RSEL   = 00         RSAID   = 000
   000000 = 000         000001 = 010        000002 = 020
   000003 = 030         000004 = 040        000005 = 050
   LATCH1 = 000         SM1    = 000        000000 = 005
   000001 = 015         000002 = 025        000003 = 035
   000004 = 045         000005 = 055        LATCH2 = 000
   SM2     = 000        RCOND  = 000        RC      = 004
   RST     = 2          RSTS   = 0          RM00    = 377
   RM01    = 202        RM02   = 003        RM03    = 374
   000000 = 226         000001 = 000        000000 = 001
   000001 = 000         000000 = 000        000001 = 000
   000000 = 000         000001 = 000        RFREQ   = 16
   CFREQ   = 016        CEM0   = 377
*****************************************************
LABEL CYCLE    25                TRUE LABELS                CLOCK TIME =    13
                                /RELM(0)/
                                /RELM(0)/
                                /RELM(0)/
                                /RELM(0)*EXECU≥+T
   RI      = 110000     EXAS   = 0          EXSI    = 0
   MISEQ   = 222        PARADO = 0          DVFLG   = 0
   CEME    = 000        RSEL   = 00         RSAID   = 000
   000000 = 000         000001 = 010        000002 = 020
   000003 = 030         000004 = 040        000005 = 050
   LATCH1 = 000         SM1    = 000        000000 = 005
   000001 = 015         000002 = 025        000003 = 035
   000004 = 045         000005 = 055        LATCH2 = 000
   SM2     = 000        RCOND  = 000        RC      = 004
   RST     = 2          RSTS   = 0          RM00    = 377
   RM01    = 202        RM02   = 003        RM03    = 374
   000000 = 226         000001 = 000        000000 = 001
   000001 = 000         000000 = 000        000001 = 000
   000000 = 000         000001 = 000        RFREQ   = 16
   CFREQ   = 016        CEM0   = 377
*****************************************************
LABEL CYCLE    27                TRUE LABELS                CLOCK TIME =    14
                                /RELM(0)/
                                /RELM(0)/
                                /(RELM(0)+CRCB)*(
                                /RELM(0)/
                                /RELM(0)*CFB≥+TRS
                                /RELM(0)*EXECU≥+T
                                /RELM(0)*TRSTS/
   RI      = 110000     EXAS   = 0          EXSI    = 0
   MISEQ   = 222        PARADO = 0          DVFLG   = 0
   CEME    = 000        RSEL   = 00         RSAID   = 000
   000000 = 000         000001 = 010        000002 = 020
   000003 = 030         000004 = 040        000005 = 050
   LATCH1 = 000         SM1    = 000        000000 = 005
   000001 = 015         000002 = 025        000003 = 035
   000004 = 045         000005 = 055        LATCH2 = 005
   SM2     = 005        RCOND  = 000        RC      = 004
   RST     = 2          RSTS   = 1          RM00    = 377
   RM01    = 202        RM02   = 003        RM03    = 374
   000000 = 226         000001 = 000        000000 = 001
   000001 = 000         000000 = 000        000001 = 000
   000000 = 000         000001 = 000        RFREQ   = 16
   CFREQ   = 017        CEM0   = 377
*****************************************************
LABEL CYCLE    29                TRUE LABELS                CLOCK TIME =    15
```

```
                                /RELM(0)/
                                /RELM(0)/
                                /(RELM(0)+CRCB)*(
                                /RELM(0)/
                                /RELM(0)*CFB≥+TRS
                                /RELM(0)*EXECU≥+T
                                /RELM(0)*TRSTS/
  RI      = 110000    EXAS    = 0         EXSI    = 0
  MISEQ   = 222       PARADO  = 0         DVFLG   = 0
  CEME    = 000       RSEL    = 00        RSAID   = 000
  000000  = 000       000001  = 010       000002  = 020
  000003  = 030       000004  = 040       000005  = 050
  LATCH1  = 000       SM1     = 000       000000  = 005
  000001  = 015       000002  = 025       000003  = 035
  000004  = 045       000005  = 055       LATCH2  = 005
  SM2     = 005       RCOND   = 001       RC      = 004
  RST     = 2         RSTS    = 1         RM00    = 005
  RM01    = 210       RM02    = 377       RM03    = 177
  000000  = 226       000001  = 000       000000  = 001
  000001  = 000       000000  = 000       000001  = 000
  000000  = 000       000001  = 000       RFREQ   = 16
  CFREQ   = 016       CEMD    = 000
**************************************************
LABEL CYCLE   31                TRUE LABELS                CLOCK TIME =   16
                                /RELM(0)/
                                /RELM(0)/
                                /(RELM(0)+CRCB)*(
                                /RELM(0)/
                                /RELM(0)*CFB≥+TRS
                                /RELM(0)*EXECU≥+T
                                /RELM(0)*TRSTS/
  RI      = 110000    EXAS    = 0         EXSI    = 0
  MISEQ   = 222       PARADO  = 0         DVFLG   = 0
  CEME    = 000       RSEL    = 00        RSAID   = 000
  000000  = 000       000001  = 010       000002  = 020
  000003  = 030       000004  = 040       000005  = 050
  LATCH1  = 010       SM1     = 010       000000  = 005
  000001  = 015       000002  = 025       000003  = 035
  000004  = 045       000005  = 055       LATCH2  = 015
  SM2     = 015       RCOND   = 002       RC      = 004
  RST     = 2         RSTS    = 1         RM00    = 377
  RM01    = 377       RM02    = 377       RM03    = 377
  000000  = 005       000001  = 000       000000  = 210
  000001  = 000       000000  = 377       000001  = 000
  000000  = 177       000001  = 000       RFREQ   = 16
  CFREQ   = 017       CEMD    = 000
**************************************************
LABEL CYCLE   33                TRUE LABELS                CLOCK TIME =   17
                                /RELM(0)/
                                /RELM(0)/
                                /(RELM(0)+CRCB)*(
                                /RELM(0)/
                                /RELM(0)*CFB≥+TRS
                                /RELM(0)*EXECU≥+T
                                /RELM(0)*TRSTS/
  RI      = 110000    EXAS    = 0         EXSI    = 0
  MISEQ   = 222       PARADO  = 0         DVFLG   = 0
  CEME    = 000       RSEL    = 00        RSAID   = 000
  000000  = 000       000001  = 010       000002  = 020
  000003  = 030       000004  = 040       000005  = 050
  LATCH1  = 020       SM1     = 020       000000  = 005
```

```
   000001 = 015        000002 = 025        000003 = 035
   000004 = 045        000005 = 055        LATCH2 = 025
   SM2    = 025        RCOND  = 003        RC     = 004
   RST    = 2          RSTS   = 1          RM00   = 123
   RM01   = 227        RM02   = 162        RM03   = 234
   000000 = 005        000001 = 000        000000 = 210
   000001 = 000        000000 = 377        000001 = 000
   000000 = 177        000001 = 000        RFREQ  = 16
   CFREQ  = 016        CEMD   = 001
*****************************************************
LABEL CYCLE    35                TRUE LABELS              CLOCK TIME =    18
                                 /RELM(0)/
                                 /RELM(0)/
                                 /(RELM(0)+CRCR)*(
                                 /RELM(0)/
                                 /RELM(0)*CFB≥+TRS
                                 /RELM(0)*EXECU≥+T
                                 /RELM(0)*TRSTS/
   RI     = 110000     EXAS   = 0          EXSI   = 0
   MISEQ  = 222        PARADO = 0          OVFLG  = 0
   CEME   = 000        RSEL   = 00         RSAID  = 000
   000000 = 000        000001 = 010        000002 = 020
   000003 = 030        000004 = 040        000005 = 050
   LATCH1 = 030        SM1    = 030        000000 = 005
   000001 = 015        000002 = 025        000003 = 035
   000004 = 045        000005 = 055        LATCH2 = 025
   SM2    = 025        RCOND  = 004        RC     = 004
   RST    = 2          RSTS   = 1          RM00   = 345
   RM01   = 315        RM02   = 022        RM03   = 234
   000000 = 005        000001 = 123        000000 = 210
   000001 = 227        000000 = 377        000001 = 162
   000000 = 177        000001 = 234        RFREQ  = 16
   CFREQ  = 017        CEMD   = 001
*****************************************************
LABEL CYCLE    37                TRUE LABELS              CLOCK TIME =    19
                                 /RELM(0)/
                                 /RELM(0)/
                                 /RELM(0)/
   RI     = 110000     EXAS   = 0          EXSI   = 0
   MISEQ  = 223        PARADO = 0          OVFLG  = 0
   CEME   = 000        RSEL   = 00         RSAID  = 000
   000000 = 000        000001 = 010        000002 = 020
   000003 = 030        000004 = 040        000005 = 050
   LATCH1 = 040        SM1    = 040        000000 = 005
   000001 = 015        000002 = 025        000003 = 035
   000004 = 045        000005 = 055        LATCH2 = 035
   SM2    = 035        RCOND  = 004        RC     = 004
   RST    = 2          RSTS   = 0          RM00   = 345
   RM01   = 315        RM02   = 022        RM03   = 234
   000000 = 005        000001 = 123        000000 = 210
   000001 = 227        000000 = 377        000001 = 162
   000000 = 177        000001 = 234        RFREQ  = 16
   CFREQ  = 017        CEMD   = 001
*****************************************************
LABEL CYCLE    39                TRUE LABELS              CLOCK TIME =    20
                                 /RELM(0)/
                                 /RELM(0)/
                                 /RELM(0)/
                                 /RELM(0)*EXECU≥+T
   RI     = 110000     EXAS   = 0          EXSI   = 0
   MISEQ  = 220        PARADO = 0          OVFLG  = 0
```

```
   CEME    = 000       RSEL    = 00        RSAID   = 000
   000000  = 000       000001  = 010       000002  = 020
   000003  = 030       000004  = 040       000005  = 050
   LATCH1  = 040       SM1     = 040       000000  = 005
   000001  = 015       000002  = 025       000003  = 035
   000004  = 045       000005  = 055       LATCH2  = 035
   SM2     = 035       RCOND   = 004       RC      = 004
   RST     = 2         RSTS    = 0         RMD0    = 000
   RMD1    = 000       RMD2    = 000       RMD3    = 000
   000000  = 005       000001  = 345       000000  = 210
   000001  = 315       000000  = 377       000001  = 022
   000000  = 177       000001  = 234       RFREQ   = 16
   CFREQ   = 017       CEMD    = 001
*****************************************************
LABEL CYCLE   41                TRUE LABELS              CLOCK TIME =   21
                               /RELM(0)/
                               /RELM(0)/
                               /RELM(0)/
                               /RELM(0)*EXECU2+T
   RI      = 110000    EXAS    = 0         EXSI    = 0
   MISEQ   = 220       PARADD  = 1         OVFLG   = 0
   CEME    = 000       RSEL    = 00        RSAID   = 000
   000000  = 000       000001  = 010       000002  = 020
   000003  = 030       000004  = 040       000005  = 050
   LATCH1  = 040       SM1     = 040       000000  = 005
   000001  = 015       000002  = 025       000003  = 035
   000004  = 045       000005  = 055       LATCH2  = 035
   SM2     = 035       RCOND   = 004       RC      = 004
   RST     = 2         RSTS    = 0         RMD0    = 000
   RMD1    = 000       RMD2    = 000       RMD3    = 000
   000000  = 005       000001  = 000       000000  = 210
   000001  = 000       000000  = 377       000001  = 000
   000000  = 177       000001  = 000       RFREQ   = 16
   CFREQ   = 017       CEMD    = 001
*****************************************************
LABEL CYCLE   43                TRUE LABELS              CLOCK TIME =   22
                               /RELM(0)/
                               /RELM(0)/
                               /RELM(0)/
                               /RELM(0)*EXECU2+T
   RI      = 110000    EXAS    = 0         EXSI    = 0
   MISEQ   = 220       PARADD  = 1         OVFLG   = 0
   CEME    = 000       RSEL    = 00        RSAID   = 000
   000000  = 000       000001  = 010       000002  = 020
   000003  = 030       000004  = 040       000005  = 050
   LATCH1  = 040       SM1     = 040       000000  = 005
   000001  = 015       000002  = 025       000003  = 035
   000004  = 045       000005  = 055       LATCH2  = 035
   SM2     = 035       RCOND   = 004       RC      = 004
   RST     = 2         RSTS    = 0         RMD0    = 000
   RMD1    = 000       RMD2    = 000       RMD3    = 000
   000000  = 005       000001  = 000       000000  = 210
   000001  = 000       000000  = 377       000001  = 000
   000000  = 177       000001  = 000       RFREQ   = 16
   CFREQ   = 017       CEMD    = 001
*****************************************************
LABEL CYCLE   45                TRUE LABELS              CLOCK TIME =   23
                               /RELM(0)/
                               /RELM(0)/
                               /RELM(0)/
                               /RELM(0)*EXECU2+T
```

```
   RI      = 110000    EXAS    = 0         EXSI    = 0
   MISEQ   = 220       PARADD  = 1         DVFLG   = 0
   CEME    = 000       RSEL    = 00        RSAID   = 000
   000000  = 000       000001  = 010       000002  = 020
   000003  = 030       000004  = 040       000005  = 050
   LATCH1  = 040       SM1     = 040       000000  = 005
   000001  = 015       000002  = 025       000003  = 035
   000004  = 045       000005  = 055       LATCH2  = 035
   SM2     = 035       RCOND   = 004       RC      = 004
   RST     = 2         RSTS    = 0         RMD0    = 000
   RMD1    = 000       RMD2    = 000       RMD3    = 000
   000000  = 005       000001  = 000       000000  = 210
   000001  = 000       000000  = 377       000001  = 000
   000000  = 177       000001  = 000       RFREQ   = 16
   CFREQ   = 017       CEMD    = 001
**************************************************
LABEL CYCLE    47                TRUE LABELS                CLOCK TIME =    24
                                 /RELM(0)/
                                 /RELM(0)/
                                 /RELM(0)/
                                 /RELM(0)*EXECU≥+1
   RI      = 110000    EXAS    = 0         EXSI    = 0
   MISEQ   = 220       PARADD  = 1         DVFLG   = 0
   CEME    = 000       RSEL    = 00        RSAID   = 000
   000000  = 000       000001  = 010       000002  = 020
   000003  = 030       000004  = 040       000005  = 050
   LATCH1  = 040       SM1     = 040       000000  = 005
   000001  = 015       000002  = 025       000003  = 035
   000004  = 045       000005  = 055       LATCH2  = 035
   SM2     = 035       RCOND   = 004       RC      = 004
   RST     = 2         RSTS    = 0         RMD0    = 000
   RMD1    = 000       RMD2    = 000       RMD3    = 000
   000000  = 005       000001  = 000       000000  = 210
   000001  = 000       000000  = 377       000001  = 000
   000000  = 177       000001  = 000       RFREQ   = 16
   CFREQ   = 017       CEMD    = 001
**************************************************

                SIMULATION ENDS AFTER THE REQUIRED NUMBER OF LABEL CYCLES

        END OF CONTROL STATEMENTS.  SIMULATION TERMINATED.
```