UNIVERSIDADE FEDERAL DA PARAÍBA

DISSERTAÇÃO DE MESTRADO

PROTÓTIPO DE UM SISTEMA PARA CONTROLE DE TRÁFEGO URBANO A TEMPO FIXO EM REDE

Ilton Luiz Barbacena

1994

ILTON LUIZ BARBACENA

PROTÓTIPO DE UM SISTEMA PARA CONTROLE DE TRÁFEGO URBANO A TEMPO FIXO EM REDE

Dissertação apresentada ao Curso de MESTRADO EM ENGENHARIA ELÉTRICA da Universidade Federal da Paraíba, em cumprimento às exigências para obtenção do grau de Mestre.

ÁREA DE CONCENTRAÇÃO: PROCESSAMENTO DA INFORMAÇÃO

MISAEL ELIAS DE MORAIS

Orientador

B228p Barbacena, Ilton Luiz.
Protótipo de um sistema para controle de tráfego urbano a tempo fixo em rede / Ilton Luiz Barbacena. - Campina Grande, 1994.
96 f.

Dissertação (Mestrado em Engenharia Elétrica) - Universidade Federal da Paraíba, Centro de Ciências e Tecnologia, 1994.
"Orientação : Prof. Dr. Misael Elias de Morais".
Referências.

1. Sistemas - Programação. 2. Semáforo - Programação - Sistema de Controle Tempo Fixo. 3. Sinalização para Controle de Tráfego. 4. Dissertação - Engenharia Elétrica. I. Morais, Misael Elias de. II. Universidade Federal da Paraíba - Campina Grande (PB). III. Título

CDU 004.45(043)

PROTÓTIPO DE UM SISTEMA PARA CONTROLE DE TRÁFEGO URBANO A TEMPO FIXO EM REDE

ILTON LUIZ BARBACENA

Dissertação Aprovada em 05.10.1994

MISAEL ELIAS DE MORAIS, Dr.Ing., UFPB
Orientador

JORGE CESAR ABRANTES DE FIGUEIREDO, D.Sc., UFPB
Componente da Banca

NILTON PEREIRA DE ANDRADE, Ph.D., UNB
Componente da Banca

CAMPINA GRANDE - PB
Outubro - 1994

AGRADECIMENTOS

Agradeço àquelas pessoas que direta ou indiretamente contribuiram para que este trabalho pudesse ser concluído, e em especial ao prof^{o} **Misael Elias de Morais** pela convivência e orientação.

Aos amigos Cortez, Giovani, Bezerra, Moreira, Marony, Aléssio, Rogério, Evilassi, Ricardo Leão, Marcílio Accioly e tanto outros, pelas constantes discussões sobre o andamento do trabalho.

Ao companheiro, conterrâneo e amigo, **Claudio Afonso Fleury** pelo auxílio e incentivo nas horas de necessidades.

À minha esposa **Mazélia** e meus filhos **Marcell** e **Marcella** que conseguiram compreender-me e apoiar nos momentos difíceis, e acima de tudo a Deus por ter-me proporcionado a conclusão deste trabalho.

RESUMO

O alto grau de sofisticação das modernas técnicas de sinalização para controle de tráfego e a necessidade de se implantar sistemas de controle centralizado operado por computador, exige requisitos funcionais dos semáforos de tráfego com características de controle automático. Apresentamos uma descrição funcional da arquitetura de um sistema que utiliza um método rápido e eficiente para fazer toda a programação semafórica, em sistemas de Controle Tempo Fixo. Este método é associado a uma interface amigável, de forma a proporcionar ao engenheiro de tráfego o completo controle dos parâmetros envolvidos.

ABSTRACT

The elevated level of sofistication used on modern sinalization technics applied to traffic control and the need of centralized system controls operated by computer, demand functional requirements on traffic semaphores that features automatic control. We present a system architecture functional description which uses a fast and efficient method to semaphoric program on a Fixed Time Traffic Control Systems. This method is associated to a friendly interface, in such way to give a complete control on envolved parameters to Traffic Engineer.

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS E ABREVIATURAS

bit - variável booleana, representada por apenas dois estados:1 e 0.

byte - palavra de oito bit's

filas - parte posterior de um comboio de veículos

CET - Controlador Eletrônico de Tráfego

CMOS - Complementary Metal-Oxide Semiconductor

CPU - Central Processor Unit

DOS - Disk Operating System

EPROM - Erasable Programmable Read-Only Memory

EEPROM - Electrically Erasable Programmable Read-Only Memory

GND - Referência de sinal

I/O - Input / Output

IDE - Integrated Development Environment

Latch - trava de dados

LSB - Least Significative Byte/Bit

MCS-51 - conjunto de microcontroladores, fabricado pela INTEL, com o conjunto de instruções comuns.

MSB - Most Significative Byte/Bit

Pla - extensão do nome de arquivo de dados com programação semafórica

POO - Programação Orientada a Objeto

RAM - Random Access Memory

RS-232 - Padrão de especificações mecânicas e elétricas p/ comunicação serial

RTC - Real Time Clock

RXD - Sinal/pino de recepção da porta serial do microcontrolador

TXD - Sinal/pino de transmissão da porta serial do microcontrolador

setar - atribuir o nível lógico 1 a um bit ou a uma variável do tipo bit

slot - conector padrão de reserva, para eventuais expansões

STANDBY- Espera, reserva

triac - componente eletrônico usado para chaveamento de cargas em 220 V ou 110 V, a partir de um determinado nível lógico (*bit*)

TTL - Transistor-Transistor Logic

VGA - Video Graphics Array

µC - microcontrolador

INTRODUÇÃO

Desde a antiguidade, sempre houve necessidade de determinar prioridades entre os diversos tipos de movimentos de pessoas, animais e meios de locomoção. Estas prioridades são necessárias para se evitar as várias possibilidades de ocorrência de conflitos entre os diversos movimentos. Imaginem se não existissem filas, como seria o acesso ao caixa de bancos no horário de pico? Quem seria o primeiro passageiro a entrar no transporte coletivo urbano no horário de pico? No trânsito, isto não é diferente.

As cidades são caracterizadas por possuírem um volume alto de movimentos de pessoas, que podem ou não utilizar algum meio de locomoção como: bicicletas, patins, motos, automóveis, ônibus, etc. O movimento destas pessoas, quando desprovido de algum meio de locomoção, é chamado de **movimento de pedestres**.

Para disciplinar os diversos movimentos presentes em cruzamentos de vias são elaborados regulamentos e leis, com objetivo de reduzir os riscos de acidentes e aumentar as condições de fluidez. Por exemplo, os pedestres devem andar sempre pelas calçadas, os veículos sempre pelas vias, as calçadas e as vias devem estar sempre desobstruídas, etc. Entretanto, é impossível evitar o conflito entre movimentos simultâneos de veículos e pedestres quando estão no mesmo nível, em um cruzamento de vias.

Estatísticas recentes demostram que a maior causa de mortes de pessoas nos grandes centros urbanos são decorrentes de acidentes de trânsito. As principais causas são: excesso de velocidade, alcoolismo, sinalização precária ou inexistente, desatenção dos motoristas e pedestres, impunidade, falta de educação para o trânsito, etc.

Entre as diversas medidas que são tomadas para minimizar a ocorrência destes conflitos podemos citar:

- Construção e manutenção de calçadas para pedestres;
- Calçamento e manutenção das vias;
- Construção de passarelas para pedestres (passagem em nível);
- Construção de viadutos (passagem em nível para veículos);
- Sinalização e manutenção das vias;
- Sinalização nas Interseções de vias;
- Adoção de Semáforos ou "sinaleiras";
- Melhoria na Legislação de Trânsito;
- Mais fiscalização (Policial de Trânsito).

Para minimizar os conflitos crônicos entre veículos e veículos com pedestres nos grandes centros urbanos, tem-se verificado uma melhoria nos controladores de tráfego, decorrente principalmente dos avanços tecnológicos. Este controle que inicialmente era feito pelo homem (guarda de trânsito), atualmente é feito com equipamentos elétricos, eletro-eletrônicos e até mesmo por sistemas sofisticados de controle.

Propõe-se neste trabalho, um *Controlador Eletrônico de Tráfego* - CET baseado em microcontroladores para ser instalado nos semáforos, que juntamente com as sinalização horizontal (pintura de piso) e vertical (placas de regulamentação), coordenará o direito de passagem entre os diversos movimentos (veículos ou pedestres). Este equipamento poderá controlar uma ou mais interseções de vias, com facilidade e flexibilidade na programação dos diversos planos de tráfegos.

A interface com o usuário do CET consiste de um microcomputador padrão IBM-PC, utilizando-se de teclado e monitor de vídeo. Através do software de interface é possível cadastrar toda a *programação semafórica* pelo engenheiro de tráfego e, posteriormente, ser enviada através da porta de comunicação serial.

Este trabalho consiste em cinco capítulos, apresentados a seguir.

No capítulo 1, apresentam-se alguns conceitos básicos de engenharia de tráfego seguidos de alguns exemplos, necessários para a compreensão de alguns parâmetros de entrada na elaboração de uma programação semafórica. Apresenta-se ainda no capítulo 1, algumas discussões sobre os tipos de controles adotados atualmente, bem como, a caracterização do controlador proposto, objeto deste trabalho.

No capítulo 2, apresenta-se a arquitetura básica do Controlador Eletrônico de Tráfego - CET, onde discute-se algumas considerações de **hardware** tais como: escolha do microcontrolador como unidade de controle, de memórias e do RTC. Neste capítulo são discutidos também a interface de comunicação serial e a interface da placa de fases.

No capítulo 3, apresenta-se a estrutura de **software** do CET, as ferramentas utilizadas durante o desenvolvimento e os *diagramas estruturados* correspondentes as rotinas ou funções principais.

No capítulo 4, apresenta-se a estrutura do software da interface, onde abordaremos também os aspectos referentes à escolha da linguagem adotada e à forma de apresentação e seleção das funções disponíveis.

O capítulo 5, apresentam-se os aspectos relacionados à implementação e testes para validação do CET.

Finalmente, apresentam-se as conclusões, juntamente com uma relação de sugestões de melhoria, com objetivo de amparar novos pesquisadores que esperamos surgir para dar continuidade a este trabalho.

SISTEMA DE ENGENHARIA DE TRÁFEGO

Neste capítulo abordaremos alguns conceitos básicos utilizados em Engenharia de Tráfego, que serão utilizados ao longo do trabalho.

Descreveremos também sobre os critérios adotados pelos engenheiros de tráfego para implantação de uma sinalização semafórica em um determinado ponto da via, ou interseção de vias, em detrimento a outro tipo de solução.

Finalmente, faremos uma descrição dos tipos de controladores semafóricos utilizados atualmente, bem como, a caracterização de nosso equipamento, objeto deste trabalho.

1.1 - Conceitos Básicos

Ruas e avenidas são o meio físico de circulação dos veículos de uma cidade, assim como, nas calçadas devem circular os pedestres. Em um cruzamento de vias, existirão sempre movimentos que não poderão fluir simultaneamente, pois serão conflitantes entre si. Desta forma, é necessário estabelecer alguma norma de controle de direito de passagem para reduzir os riscos de acidentes e aumentar a fluidez no cruzamento [CONTRAN, 1979].

Nas vias de baixo volume de tráfego, os conflitos entre veículos são resolvidos através de uma regra implícita: *o primeiro a chegar é o primeiro a atravessar.* No caso do conflito entre veículos e pedestres, o veículo sempre tem prioridade. Agora, se veículos chegarem simultaneamente no cruzamento, a disputa pelo direito de passagem resulta em discórdia e discussão. Este conflito agrava-se com o aumento do volume de tráfego no cruzamento. Neste caso, é necessário que o poder público determine as prioridades entre os diversos movimentos no cruzamento, implantando uma sinalização.

A escolha do tipo de sinalização a ser adotada, segue normas rígidas que são estudadas em Engenharia de Tráfego, baseada nas características físicas do cruzamento e no volume de tráfego que circula naquele cruzamento.

De uma maneira geral, a implantação de um semáforo, por aumentar o tempo de travessia da interseção, é o *último* recurso adotado para estabelecer o controle do direito de passagem.

Algumas medidas devem ser consideradas, antes de se proceder estudos que justificariam a utilização de semáforos [CONTRAN, 1979], tais como:

- implantação ou melhoria da sinalização vertical (placas de regulamentação e advertência) e sinalização horizontal (pintura de piso, tachas, picolés, etc.);
- remoção de interferências que prejudicam a visibilidade da sinalização;
- mudança da geometria da interseção (canalização física para separar movimentos conflitantes, faixas especiais de conversão, rótulas, etc.);
- melhoria da iluminação;
- controle de velocidade de aproximação (placas, lombadas, etc.)

Em seguida, vamos enumerar alguns termos técnicos comuns nesta área, que passaremos a adotar.

1.1.1 - Termos Típicos

- Cruzamento ou Interseção

É formado pela interseção de duas ou mais vias.

- Movimentos

É formado pelos deslocamentos de veículos e pedestres. Quando não citados explicitamente, trata-se de movimentos veiculares.

- Semáforo

O semáforo é um instrumento de controle modificável que informa as prioridades de passagem para motoristas ou pedestres em uma via ou em uma

interseção de vias, utilizando-se de sinais luminosos. De forma geral, durante a implantação de um ***semáforo***, faz-se necessário a implantação das sinalizações horizontais (faixa de pedestre) e verticais.

O termo semáforo de tráfego é considerado uma instalação completa, incluindo os sinais luminosos (portafocos, lentes, lâmpadas, coluna, braço projetado, etc.), os fios elétricos, o *controlador de tráfego*, etc.

- Semáforo veicular

O objetivo principal do semáforo veicular é autorizar/proibir o movimento de veículos de uma corrente de tráfego.

Este semáforo utiliza portafocos com focos redondos e os seguintes estados para os focos: verde, amarelo, vermelho. A seqüência de ativação deverá ser verde, amarelo e vermelho. Embora não regulamentado, utiliza-se também o amarelo piscante.

Cada um destes estados tem um significado, conforme descrito a seguir:

verde: os condutores de veículos que recebem a indicação luminosa neste estado podem seguir em frente, efetuando a travessia do cruzamento, a não ser que sejam impedidos fisicamente por outro dispositivo de controle de tráfego ou autoridade legal. Devem, no entanto, ceder o direito de passagem aos veículos e/ou pedestres que se encontrem legalmente na área da interseção;

amarelo: os condutores de veículos que recebem uma indicação luminosa neste estado devem parar o veículo, antes de entrar na interseção, e permanecer parados até que recebam autorização de passagem através de luz verde ou autoridade legal. Caso não seja possível parar, sem risco para segurança do tráfego, devem continuar em frente e cruzar a interseção. A fim de não se proceder a uma interrupção brusca de movimento estipulou-se um tempo para o amarelo (**atenção**), que é uma situação intermediária entre movimento e parada. Ao receber a indicação amarelo, os motoristas são alertados sobre a proximidade da mudança, porém tem tempo suficiente para reagir a ela.;

vermelho: os condutores de veículos que recebem indicação luminosa neste estado devem parar o veículo antes de entrar na interseção, e permanecer parados até que recebam autorização de passagem através de luz verde ou autorização legal.

Amarelo Piscante: Este estado é utilizado normalmente simultaneamente em todas as fases semafóricas, sendo ativada quando o controlador está com problemas ou está em manutenção. É muito comum também usar este estado em horários de baixo volume de tráfego, normalmente após a meia noite.

- Semáforo para pedestre

Trata-se de um semáforo que utiliza portafocos quadrado com dois focos cujos estados são respectivamente: verde, vermelho piscante e vermelho. No foco verde deverá existir uma máscara, agregada à parte interna da lente, representando a figura de um boneco andando. No foco vermelho, da mesma forma , porém um boneco parado.

As cores das lentes obedecem ao padrão dos semáforos veiculares e têm os significados:

boneco verde fixo: os pedestres que recebem indicação luminosa neste estado podem atravessar a via, enquanto os veículos devem permanecer parados na área de interseção;

boneco vermelho intermitente: os pedestres que recebem indicação luminosa neste estado, e que já iniciaram a travessia, devem procurar terminá-la, e aqueles que ainda não a iniciaram devem parar antes de entrar na interseção e permanecer parados até que recebam autorização de passagem do "boneco verde" ou de autorização legal;

boneco vermelho fixo: os pedestres que recebem indicação luminosa neste estado devem parar antes de atravessar a via, e permanecer assim até que recebam autorização de travessia através do "boneco verde" ou autoridade legal.

- Fase Semafórica

É uma seqüência completa de indicação de cores dos focos, que permite a um conjunto de vias o controle simultâneo do direito de passagem. No caso da fase veicular temos fisicamente três focos, e uma fase é completada após passar pela seqüência verde, amarelo e vermelho. No caso da fase de pedestre temos dois focos, e uma fase é completada após passar pela seqüência verde, vermelho piscante e vermelho.

- Ciclo ou tempo de ciclo

É o tempo gasto para completar uma fase semafórica. A partir deste período, começa a repetir periodicamente, as cores dos focos desta fase.

- Estágio

É formado por intervalos dentro de um ciclo, onde são alternados os movimentos das correntes de tráfego (Figura 1.1b). Isto significa que cada estágio corresponde do início ao fim do verde, para cada fase semafórica, em um mesmo cruzamento.

- Período de Entreverdes

É período entre o término de verde de uma fase e o início do tempo de verde da outra fase. Este período é caracterizado por ser curto.

- Vermelho Total

É um período durante o qual todos os focos das fases semafórica ficam com a cor vermelha. Este período é utilizado para dar maior segurança na travessia de veículos e pedestres.

- Diagrama de Tempo

É uma representação gráfica que associa os instantes de mudança dos estágios com a seqüência de cores e duração dos estados de cada fase .

Vejamos um exemplo em que devemos instalar um semáforo em um cruzamento com as seguintes características (fornecidas pelo engenheiro de tráfego):

- interseção de duas vias de sentido único (mão única).
- tempos de verde: 30 e 25 seg.
- tempos de amarelo = 3 seg.
- tempos de vermelho total = 2 seg.
- não considerar os movimentos de pedestres.

A Figura 1.1a representa o cruzamento em questão. A Figura 1.1b representa os dois estágios possíveis: o estágio 1, quando são permitidos os movimentos oriundos da aproximação pela rua A e o estágio 2, quando são permitidos os movimentos oriundos da aproximação pela rua B. Para obtermos o diagrama de tempos, devemos observar os valores dos tempos de cada estado das fases. Observe que a fase 1 após 30 seg, deve entrar no amarelo, passando pelo vermelho total quando começa o verde da fase 2, e até o final deste verde, do amarelo e do vermelho total desta fase, a fase 1 permanece em vermelho. De modo análogo obtemos a fase 2, considerando que enquanto uma fase está no verde ou amarelo a outra deverá encontrar-se no vermelho. A Figura 1.2 mostra o digrama de tempos.

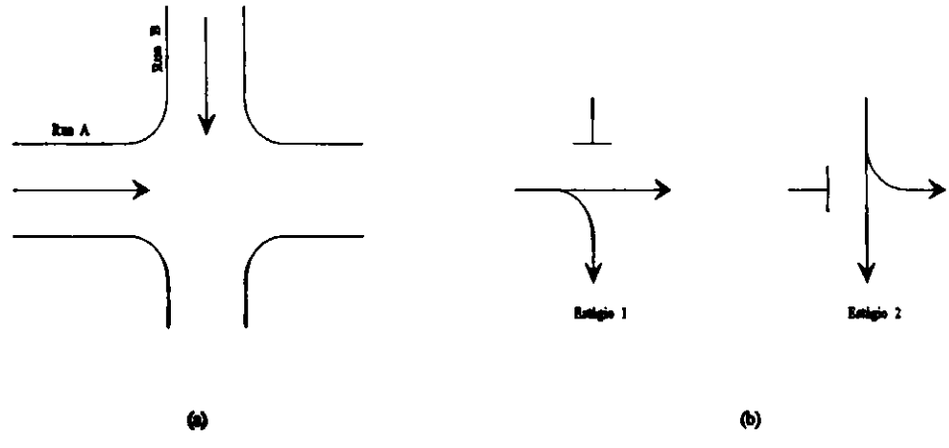


Figura 1.1 - Movimentos de aproximações e diagramas de estágios

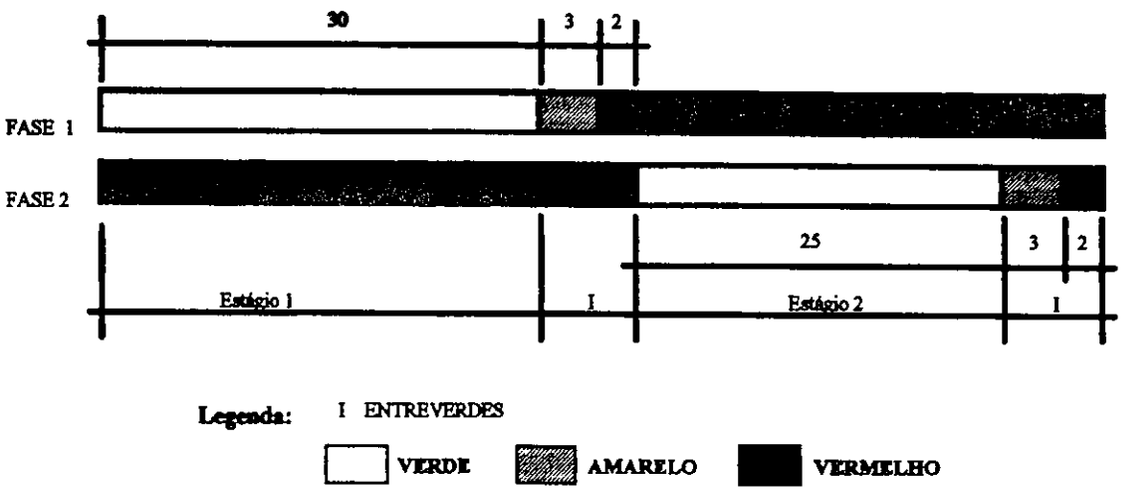


Figura 1.2 - Diagrama de tempos

Para calcularmos o tempo de ciclo, basta efetuarmos a seguinte soma:

ciclo = verde1 + amarelo1 + vm_total1 + verde2 + amarelo2 + vm_total2

ciclo = 30 + 3 + 2 + 25 + 3 + 2 = 65 segundos.

Neste caso o período entreverdes será de :

I = 3 + 2 = 5 segundos.

1.1.2 - Plano de Tráfego

O engenheiro de tráfego é quem determinará quando e por quanto tempo, uma fase de tráfego deve acionar os focos das fases semafórica, estabelecendo uma das seguintes situações: verde, amarelo, vermelho, vermelho piscante e amarelo piscante.

Estes dados naturalmente são determinados através de cálculos com base em levantamentos estatísticos do cruzamento e adotando-se uma das metodologias da Engenharia de Tráfego, como por exemplo o método de Webster [WEBSTER, 1966].

O tempo de ciclo, duração e instantes de início dos estágios, e as cores dos focos constituem um conjunto de parâmetros denominados *planos de tráfego*, cujos valores são programados nos controladores de tempo fixo [DENATRAN, 1979]. Isto significa que cada plano de tráfego deve corresponder um diagrama de tempos, e que quanto maior a quantidade de *fases semafóricas* mais dados terão o plano de tráfego.

Alguns controladores de tráfego permitem a programação de mais de um plano de tráfego, que podem servir para diversas situações existentes ao longo do dia, como por exemplo: pico da manhã, pico da tarde, fora do pico, fim de semana, após a meia noite, etc.

1.2 - Critérios para Instalação de Semáforos

A necessidade da instalação de um semáforo depende de critérios adotados em engenharia de tráfego, em função da capacidade da interseção sem o semáforo e seus problemas. Entretanto devemos lembrar que este tipo de sinalização não traz a cura certa para todos os problemas das interseções.

1.2.1 - Considerações Iniciais

A instalação de semáforos em um cruzamento tem *vantagens* e *desvantagens*, entre os quais podemos enumerar:

Vantagens:

a - Estabelece um movimento ordenado do tráfego (blocos);

b - Reduz a freqüência de certos tipos de acidentes, como as batidas laterais;

c - Providencia um meio de interrupção do tráfego intenso, para permitir que outro tráfego de veículos ou pedestres entre na corrente principal ou cruze com a máxima segurança com demoras mínimas;

d - Dá economia considerável em relação ao controle humano, em interseções onde os sinais são necessários durante longos períodos;

e - Dá confiança ao motorista, para atravessar o cruzamento; e

f - Quando coordenados adequadamente, asseguram fluxo contínuo ou quase contínuo de veículos em velocidade apropriada para a via ("onda verde").

Desvantagens:

a - Aumentam em muitas instalações, a demora na interseção, especialmente fora dos períodos de pico;

b - Provável aumento de certos acidentes, como as batidas atrás;

c - Podem causar demoras desnecessárias e desrespeito por parte do motorista, quando implantado sem necessidade;

d - Quando mal sincronizados, causam demoras excessivas, aumentando a irritação dos motoristas; e

e - Desvio do tráfego para rotas alternativas que podem ser menos adequadas.

1.2.2 - Justificativa

Os critérios que justificam a implantação de um semáforo em um cruzamento [DENATRAN, 1979], referem-se a:

a - volume veiculares mínimos em todas as aproximações;

b - interrupção de tráfego contínuo;

c - volumes conflitantes em interseções de cinco ou mais aproximações;

d - volumes mínimos de pedestres que cruzam a via principal;

e - índice de acidente e os diagramas de colisão;

f - melhoria de sistema progressivo, ou implementação de velocidade constante em uma via com mais de um semáforo (onda verde);

g - controle de áreas congestionadas;

h - combinação de critérios;

i - situações locais específicas.

Estes critérios estão devidamente tabulados e justificados na referência citada.

1.3 - Tipos de Controles

O *controlador de tráfego* é um equipamento que comanda o semáforo através do envio de pulsos elétricos para comutação das luzes dos focos, efetuando assim, o controle dos movimentos de veículos e pedestres no sistema viário. O instante em que

os pulsos são enviados é definido através de uma programação interna, cuja lógica pode ser simples ou complexa, dependendo do tipo de controlador.

Existem basicamente dois tipos de controladores: controladores de tempo fixo e controladores por demanda de tráfego.

Nos *controladores de tempo fixo* o tempo de ciclo e os instantes de mudança dos estágios são constantes. Isto significa que o diagrama de tempo é fixo para cada plano de tráfego, independente da demanda instantânea de tráfego. Nestas circunstâncias, pode-se elaborar planos de tráfegos para diversas situações existentes ao longo do dia, como por exemplo: pico da manhã, pico da tarde, fora do pico, após meia noite, etc.

Nos *controladores por demanda de tráfego* são utilizados detectores de veículos e lógica de decisão. Caracterizam-se por possuírem tempo de ciclo variável, ajustando-se, dinamicamente, às flutuações de tráfego que podem ocorrer num cruzamento. Neste caso, ao tempo de verde associado a um determinado estágio, deverá ser programado os seguintes valores: *verde mínimo, verde máximo e extensão de verde.* O mínimo período de verde corresponde ao tempo necessário para passagem segura de um veículo, ou travessia de pedestre no cruzamento. A partir da duração mínima, são adicionados extensões de verdes, acionadas pela detecção de veículos na faixa de tráfego com direito de passagem. O número de extensões será limitado pelo máximo período de verde programado.

A escolha do tipo de controlador para implantação em um cruzamento, dependerá da estratégia de controle adotada. Existem três categorias básicas que decreveremos a seguir:

1.3.1 - Controles Isolados

Neste tipo de controle são considerados apenas os volumes de veículos existentes no cruzamento. Utilizados em cruzamentos distantes de outros semáforos, não havendo qualquer compromisso de sincronismo. Neste caso, pode utilizar-se tanto o controlador com tempos fixos como o controlador por demanda de tráfego.

1.3.2 - Controle em Rede Aberta

Neste tipo de controle, também conhecido como *sistema progressivo* ou *sincronizado* ou *onda verde,* a preocupação é dar continuidade de movimentos em uma determinada via. Desta forma, o condutor deverá pegar todos os verdes de todos os semáforos em uma via, se estiver implementando uma determinada velocidade fixa. Para conseguirmos 100% de sincronismo é necessário que todos os semáforos desta via tenham um mesmo tempo de ciclo fixo ,e que a via seja de sentido único. Neste caso, devemos utilizar *controladores a tempo fixo.*

1.3.3 - Controle em Rede Fechada

Neste tipo de controle são consideradas todas as intersecções sinalizadas de uma determinada região. Um exemplo típico são os centros urbanos das grandes cidades, onde praticamente existem semáforos em cada cruzamento, e estão muitos próximos entre si, formando uma malha semafórica. Este tipo de controle caracteriza-se pelo uso de um *computador central*, comunicando-se *on-line* com todos os controladores da rede. Desta forma, são implementadas algumas facilidades, tais como:

a- flexibilidade e facilidade de mudança de planos de tráfego a tempo fixo;

b- facilidade de supervisão pelo operador de sistema;

c- implantação de estratégias mais complexas.

1.4 - Proposta deste Trabalho

Nossa proposta é trabalhar com controles a tempo fixo com programação de planos de tráfego através dos **trechos.** Trecho é um intervalo de tempo de um plano de tráfego, em que as cores dos focos permanecem constantes para todas as fases semafórica. A quantidade de trechos depende do plano de tráfego, sendo possível subdividir trechos grandes em trechos menores, para facilitar a programação.

De acordo com nossa proposta, a programação completa para um *controle em tempo fixo* de um cruzamento, deve incluir os seguintes dados:

1 - Planos de Tráfego;

2 - Tabela de Trocas dos Planos;

3 - Tabela de Conflitos para os focos verdes.

Para cada *Plano de Tráfego*, o engenheiro de tráfego deverá fornecer os seguinte dados:

1 - quantidade de trechos;

2 - para cada trecho:

. duração em segundos

. cor dos focos para cada fase semafórica

Na *Tabela de Trocas de Planos*, o engenheiro de tráfego deverá fornecer os seguintes dados:

1 - horário de entrada de plano, que inclui:

. dias da semana

. horas

. minutos

. segundos

2 - número do plano, que deverá ficar ativo, a partir deste horário.

A *Tabela de Conflito* indicará ao sistema quais as fases semafórica em que nunca poderão ocorrer verdes simultâneos. Isto é necessário para o caso de um controlador controlar fases semafórica de cruzamentos diferentes. Neste caso, os verdes dentro de um mesmo cruzamento serão conflitantes entre si, e os verdes de cruzamentos diferentes serão permitidos. O sistema deverá fazer periodicamente uma leitura na saída dos focos verdes, comparar com a tabela de conflitos, e se existir conflito (situação em

que houve uma possível falha de hardware) atuará desligando todos os focos e entrando no módulo piscante.

Para exemplificar este procedimento, apresentaremos como exemplo os cruzamentos ilustrados na Figura 1.3, com os movimentos de aproximação indicados na figura, onde o engenheiro de tráfego deseja implantar um controle a tempo fixo adotando o diagrama de estágios da Figura 1.4 e com a planta de localização das fases semafórica dado na Figura 1.5.

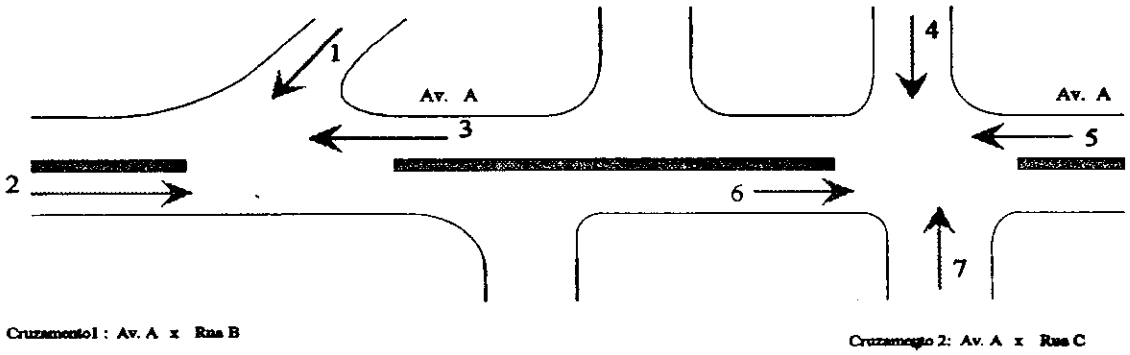


Figura 1.3 - Planta das vias e movimentos de aproximações

Figura 1.4 - Diagrama de estágios

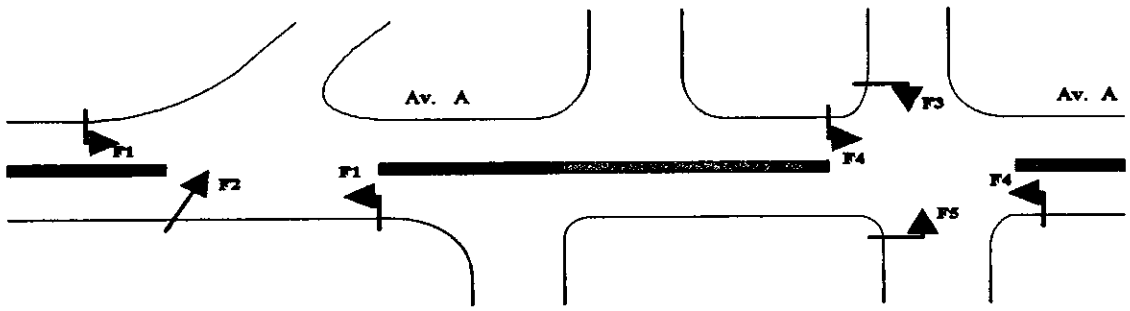


Figura 1.5 - Planta de localização das fases semafórica (portafocos)

Neste exemplo, são fornecidos ainda os seguintes dados:

(a) Tempos por plano de Tráfego

- Plano 1:

verde fase 1	= 21 seg	verde fase 2	= 19 seg
verde fase 3	= 13 seg	verde fase 4	= 12 seg
verde fase 5	= 10 seg	amarelos	= 3 seg
vermelho estendido	= 2 seg	defasagem	= 5 seg

- Plano 2:

verde fase 1	= 30 seg	verde fase 2	= 25 seg
verde fase 3	= 20 seg	verde fase 4	= 18 seg
verde fase 5	= 12 seg	amarelos	= 3 seg
vermelho estendido	= 2 seg	defasagem	= 5 seg

- Plano 3:

verde fase 1	= 37 seg	verde fase 2	= 28 seg
verde fase 3	= 20 seg	verde fase 4	= 24 seg
verde fase 5	= 16 seg	amarelos	= 3 seg
vermelho estendido	= 2 seg	defasagem	= 5 seg

Observações:

1- Os tempos de amarelo e vermelho estendido são comuns para todas as fases semafórica, no plano de tráfego.

2- A defasagem citada nos planos refere-se desde o início do verde da fase 3, até o início do verde da fase 1.

3 - O valor da defasagem, leva em consideração a distâncias entre os cruzamentos, a velocidade que se deseja implementar na via para cada plano de tráfego e principalmente a eliminação de "filas", evitando congestionamentos.

(b) Horários de entrada de planos, ao longo da semana:

Tabela 1.1 -Tabela de trocas de planos

Número do Plano	Horário			Dias da Semana						
	Horas	Min	Seg	D	S	T	Q	Q	S	S
1	6	30	0	S	S	S	S	S	S	S
1	10	0	0	-	S	S	S	S	S	-
1	21	0	0	-	S	S	S	S	S	-
1	15	0	0	-	-	-	-	-	-	S
2	8	30	0	-	S	S	S	S	S	-
2	11	45	0	-	S	S	S	S	S	-
3	23	45	0	S	S	S	S	S	S	S

O relógio do controlador ao atingir cada um dos horários previstos na Tabela 1.1, verifica se está programada a troca para aquele dia da semana e, em caso afirmativo, o controlador deverá passar a atualizar os focos de acordo com o novo plano de tráfego previsto na tabela.

Para transpormos os dados do exemplo de acordo com nossa proposta, devemos inicialmente elaborarmos os planos de tráfego para cada plano, conforme mostrados na Figura 1.6.

A partir destes dados devemos subdividir o ciclo em trechos. Feito isso, devemos obter o tempo de cada trecho e a cor dos focos de cada fase semafórica para todos os trechos. As Tabelas 1.2, 1.3 e 1.4 mostra as cores dos focos e os tempos por trecho.

Com a tabela de trocas dos planos e os planos de tráfego caracterizados pelos seus trechos, para completar a programação do controlador falta apenas verificar os conflitos de verdes não permitidos, que neste caso são:

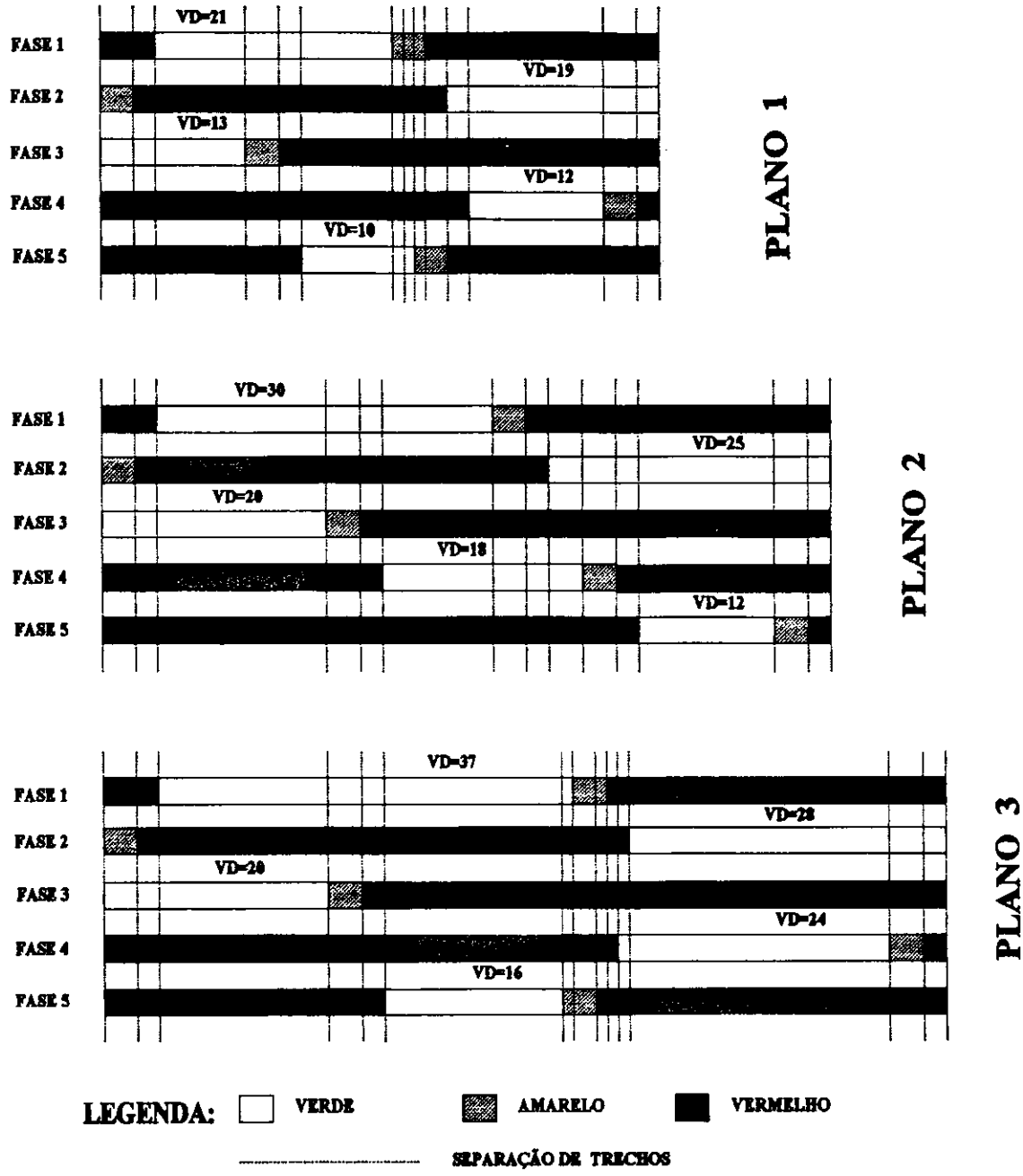


Figura 1.6 - Diagramas de tempos

Fase 1 conflita com Fase 2 - cruzamento 1

Fase 3 conflita com Fase 4 - cruzamento 2

Fase 3 conflita com Fase 5 - cruzamento 2

Fase 4 conflita com Fase 5 - cruzamento 2

Tabela 1.2 -Cores e tempos para plano 1

Trecho	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Fase 1	R	R	G	G	G	G	Y	Y	Y	R	R	R	R	R
Fase 2	Y	R	R	R	R	R	R	R	R	R	G	G	G	G
Fase 3	G	G	G	Y	R	R	R	R	R	R	R	R	R	R
Fase 4	R	R	R	R	R	R	R	R	R	R	R	G	Y	R
Fase 5	R	R	R	R	R	G	G	G	Y	Y	R	R	R	R
Tempo(s)	3	2	8	3	2	8	1	1	1	2	2	12	3	2

Tabela 1.3 - Cores e tempos para plano 2

Trecho	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Fase 1	R	R	G	G	G	G	Y	R	R	R	R	R	R	R
Fase 2	Y	R	R	R	R	R	R	R	G	G	G	G	G	G
Fase 3	G	G	G	Y	R	R	R	R	R	R	R	R	R	R
Fase 4	R	R	R	R	R	G	G	G	G	Y	R	R	R	R
Fase 5	R	R	R	R	R	R	R	R	R	R	R	G	Y	R
Tempo(s)	3	2	15	3	2	10	3	2	3	3	2	12	3	2

Tabela 1.4 - Cores e tempos para plano 3

Trecho	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Fase 1	R	R	G	G	G	G	G	Y	Y	R	R	R	R	R
Fase 2	Y	R	R	R	R	R	R	R	R	R	R	G	G	G
Fase 3	G	G	G	Y	R	R	R	R	R	R	R	R	R	R
Fase 4	R	R	R	R	R	R	R	R	R	R	G	G	Y	R
Fase 5	R	R	R	R	R	G	Y	Y	R	R	R	R	R	R
Tempo(s)	3	2	15	3	2	16	1	2	1	1	1	23	3	2

Onde: G = verde
Y = amarelo
R = vermelho

Nos próximos capítulos descreveremos o funcionamento do hardware e software do equipamento eletrônico proposto, para que a partir dos dados fornecidos pelo engenheiro de tráfego, conforme exemplo anterior, este equipamento assuma o controle do tráfego.

HARDWARE PROPOSTO

Neste capítulo apresentaremos a proposta de arquitetura de hardware, para atender as características de um controlador eletrônico semafórico a tempo fixo. Tendo em vista a necessidade de armazenar programas e dados, fazer verificação de relógio interno juntamente com a tabela de troca de planos para tomada de decisão quanto a troca de planos, leva-nos a utilização de um *microprocessador* ou *microcontrolador* para fazer o controle deste processo. O detalhamento completo da placa principal e das placas de fases são mostrados nos Anexos 1 e 2, respectivamente.

2.1 - Arquitetura Básica

O *microcontrolador* é um componente eletrônico que já tem incorporado em seu invólucro vários blocos de funções, permitindo a construção de sistemas compactos e poderosos, comparando-se com os sistemas baseados em *microprocessadores*. Nos microprocessadores estes blocos de funções são tratados fora do invólucro por outro *chip*, como por exemplo: memórias, temporizadores programáveis, portas paralelas e seriais, etc.

Atualmente existem diversas famílias de microcontroladores disponíveis no mercado, fabricados pelas grandes indústrias de componentes eletrônicos, tais como: INTEL, MOTOROLA, PHILIPS, SIEMENS, NATIONAL, ITAUCOM, etc. Entre elas, destaca-se a INTEL, por ter sido uma das pioneiras no fornecimento destes componentes.

Ao estudar qual microcontrolador seria mais adequado às nossas necessidades, levamos em consideração os seguintes parâmetros: custo, facilidade de aquisição, ferramentas de software disponíveis, perspectiva tecnológica, documentação técnica, características técnicas e recursos disponíveis. Finalmente optamos por trabalhar com o

microcontrolador 8031 da INTEL, por preencher a maioria dos requisitos. Suas principais características [Silva, 1990] são mostradas na Tabela 2.1. Na memória RAM interna convive os bancos de registradores, as variáveis internas quando definidas no software e o Stack Pointer.

Tabela 2.1 - Características do microcontrolador Intel 8031

CARACTERÍSTICAS	DESCRIÇÃO
Tipo de CPU	8 bits
RAM interna	128 bytes
RAM Externa	até 64 Kbytes
Rom	Externa, até 64 Kbytes
Linhas de Entrada/Saída	31 disponíveis + Ram Externa
Frequência de Operação	até 12 Mhz
Fonte de Alimentação	+ 5 Volts (única)
Bits Endereçáveis	128 Bits
Número de Instruções	255, 44% de 1 byte e 41% com 2 bytes
Expansões	Memória e Entrada/Saída
Fontes de Interrupções	5, com dois níveis de prioridades
Compatibilidade com TTL	Entradas e Saídas
Consumo de Potência	Baixo
Tecnologia	CMOS
Dados Adicionais	- 4 Banco de Registradores de 8 bits - Pilha de até 128 bits (Ram interna) - Compiladores C e Assembly, disponíveis

Utilizando o microcontrolador 8031 e levando-se em consideração as características que deve possuir um controlador a tempo fixo, projetamos o hardware completo, cujo diagrama de blocos pode ser visto na Figura 2.1. Na placa da CPU estão as memórias, RTC, circuito de reset, etc.

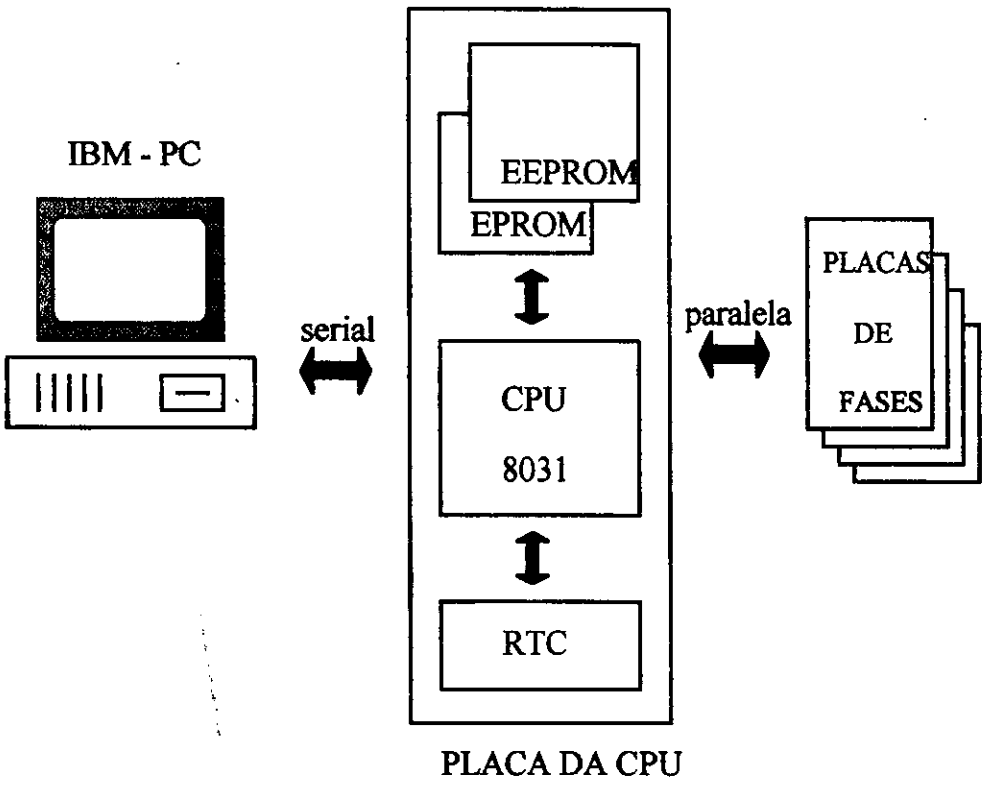


Figura 2.1 - Diagrama de blocos do hardware

As principais partes do hardware são:

1 - Placas de Fases, com interfaces de fases

2 - RTC - Relógio de Tempo Real

3 - Memórias: EEPROM e EPROM

4 - CPU 8031

5 - Interface Serial

A seguir passaremos a descrever cada uma destas partes.

2.2 - Memórias

Neste projeto trabalhamos com memórias RAM, EPROM e EEPROM. A memória RAM utilizada foi a própria memória interna do microcontrolador, local onde são armazenadas as variáveis internas definidas no software de controle, os dados manipulados e o próprio Stack Pointer.

A memória EPROM adotada foi a 2764, com capacidade máxima de armazenamento de 8 Kbytes [AMD, 1991], embora tenhamos utilizado apenas 60% desta capacidade. Nesta memória não volátil, fica gravado o software de controle do equipamento. Após a elaboração do programa fonte, o compilador gera o código objeto que, após passar pelo processo de ligação, o arquivo é convertido para o formato Intel e posteriormente gerado o *código binário*, que finalmente pode ser gravado na EPROM através de um gravador externo. Para fazer a gravação é necessário apagar a memória antes, caso ela não esteja limpa. O processo de apagamento ocorre através de exposição à luz ultravioleta, por um período pré-determinado (de 30 a 300 segundos). Efetuando-se a gravação do arquivo binário na EPROM, a pastilha é então novamente incorporada fisicamente ao equipamento.

A memória EEPROM adotada foi a 2864, com capacidade máxima de armazenamento de 8 Kbytes [XICOR, 1989], e este dado é que limita a capacidade de controle do equipamento. Nesta memória também não volátil, fica armazenada toda a programação semafórica, ou dados relativos ao controle do tráfego. Estes dados são obtidos através do software da interface, rodando em equipamento externo em ambiente MS-DOS, e transmitidos através de comunicação serial deste equipamento com o controlador. Este tipo de memória caracteriza-se por permitir a gravação de dados diretamente pela CPU, sem desconectá-la do circuito, como se fosse uma RAM. Entretanto, o processo de gravação é lento, em torno de 5 ms por cada byte. Desta forma, a gravação completa de dados em sua capacidade máxima, gasta aproximadamente 41 segundos. Este fato pouco influi no projeto, visto que, em

condições normais, feito uma gravação, uma próxima gravação só ocorrerá depois de dias ou meses.

O mapa de dados gravados na EEPROM, é o seguinte:

1 - Cabeçalho de Dados, conforme Tabela 2.2;

2 - Planos de Tráfego, conforme Tabela 2.3;

3 - Tabela de Trocas, conforme Tabela 2.4

4 - Código de Erros.

Tabela 2.2 - Cabeçalho dos dados da EEPROM

Endereço da Eprom (Hexa)	Significado do dado gravado neste Endereço (valor em Hexa)	Endereço da Eprom (Hexa)	Significado do dado gravado neste Endereço (valor em Hexa)
0000	Número. do controlador	000C	Conflitos (fase 8)
0001	Total de Fases	000D	Cor Fases 1/2 - Pl.. Especial
0002	Total Trechos	000E	Cor Fases 3/4 - Pl.. Especial
0003	Total de Planos	000F	Cor Fases 5/6 - Pl.. Especial
0004	Total de Trocas	0010	Cor Fases 7/8 - Pl.. Especial
0005	Conflitos (fase 1)	0011	dia da gravação
0006	Conflitos (fase 2)	0012	dia do mês
0007	Conflitos (fase 3)	0013	ano
0008	Conflitos (fase 4)	0014	nome do arquivo (início)
0009	Conflitos (fase 5)	...	...
000A	Conflitos (fase 6)		nome do arquivo (Fim)
000B	Conflitos (fase 7)		F4 → Fim do Cabeçalho

A partir do endereço inicial da EEPROM é gravado o cabeçalho dos dados em ordem seqüencial conforme a Tabela 2.2, encerrando-se com o byte "F4", após a gravação do último caracter do nome do arquivo correspondente da interface.

Tabela 2.3 - Início da tabela de planos na EEPROM

Endereço (Hexa)	Plano	Trecho	Byte Gravado (Significado)
00C8	1	1	tempo (segundos)
00C9	1	1	Cores da fases 1/2
00CA	1	1	Cores da fases 3/4
00CB	1	1	Cores da fases 5/6
00CC	1	1	Cores da fases 7/8
00CD	1	2	tempo (segundos)
00CE	1	2	Cores da fases 1/2
00CF	1	2	Cores da fases 3/4
00D0	1	2	Cores da fases 5/6
00D1	1	2	Cores da fases 7/8
00D2	1	3	tempo (segundos)
...	...	...	...
	1	n	Cores da fases 7/8
	1	n	F8 ➜ Fim deste Plano

Os planos de tráfego são gravados a partir do endereço "00C8" em ordem seqüencial de trechos, com cada plano ocupando um total de 160 bytes, conforme

Tabela 2.3. O fim de cada plano ocorre quando for gravado o byte "F8" no início de um novo trecho. Para cada trecho são reservados cinco bytes, o primeiro corresponde ao tempo de duração do trecho e os demais representam, em ordem seqüencial, as cores dos focos por fase, sendo que cada byte corresponde a duas fases semafóricas. Isto significa que o primeiro byte representa as cores das fase 1 (Nible menos significativo) e fase 2 (Nible mais significativo), o segundo byte as cores das fases 3 e 4, e assim sucessivamente. Na Tabela 2.3 é mostrado os dados para o plano 1. O plano 2, começa a partir do endereço "0168", que corresponde ao endereço inicial do plano 1 acrescido de 160 bytes. O byte "F4" é gravado no endereço correspondente ao fim do último plano programado (em lugar do byte "F8"), para indicar fim da gravação de planos.

A tabela de trocas é gravada a partir do endereço "0BB8", conforme mostrado na Tabela 2.4. Cada bit correspondente ao byte week, representa um dia da semana, que quando setado significa programado para aquele dia. Esta tabela é gravada em ordem seqüencial de trocas, encerrando-se com o byte "F4", no início de uma nova troca.

Tabela 2.4 - Tabela de trocas na EEPROM

Endereço (hexa)	**Byte gravado (significado)**	**Endereço (hexa)**	**Byte gravado (significado)**
0BB8	Horas - troca 1	0BBF	segundos - troca 2
0BB9	minutos - troca 1	0BC0	week - troca 2
0BBA	segundos - troca 1	0BC1	Plano - troca 2
0BBB	week - troca 1	0BC2	Horas - troca 3
0BBC	Plano - troca 1	...	...
0BBD	Horas - troca 2	...	Plano - troca n
0BBE	minutos - troca 2	....	F4 → Fim de trocas

Os códigos de erros são gravados a partir do endereço "00BE", quando da ocorrência de alguns erros previstos no software, terminando com o byte "F4".

2.3 - Interfaces Externas

Na essência, a CPU fica monitorando a comunicação serial e atendendo as interrupções para monitoriação e atualização das placas de fases em função dos dados programados em EEPROM.

2.3.1 - Interface e Placa de Fases

A cada segundo, o software principal atende uma interrupção que atualiza as placas de fases, gravando as cores de cada fase semafórica, de acordo com a programação em EEPROM. O circuito de interface, presente em cada placa de fase, é mostrado na Figura 2.2. Neste caso, temos um endereço para cada placa de fases, correspondente a duas fases semafóricas, onde cada metade do byte escrito na porta ou latch aciona, separadamente, sua fase semafórica. A configuração de bits gravadas pelo software de controle, correspondente a cada fase semafórica, fica restrita a uma das seis possibilidades de entradas permitidas mostradas na Figura 2.2, que são: verde, amarelo, vermelho, amarelo piscante, vermelho piscante e apagado.

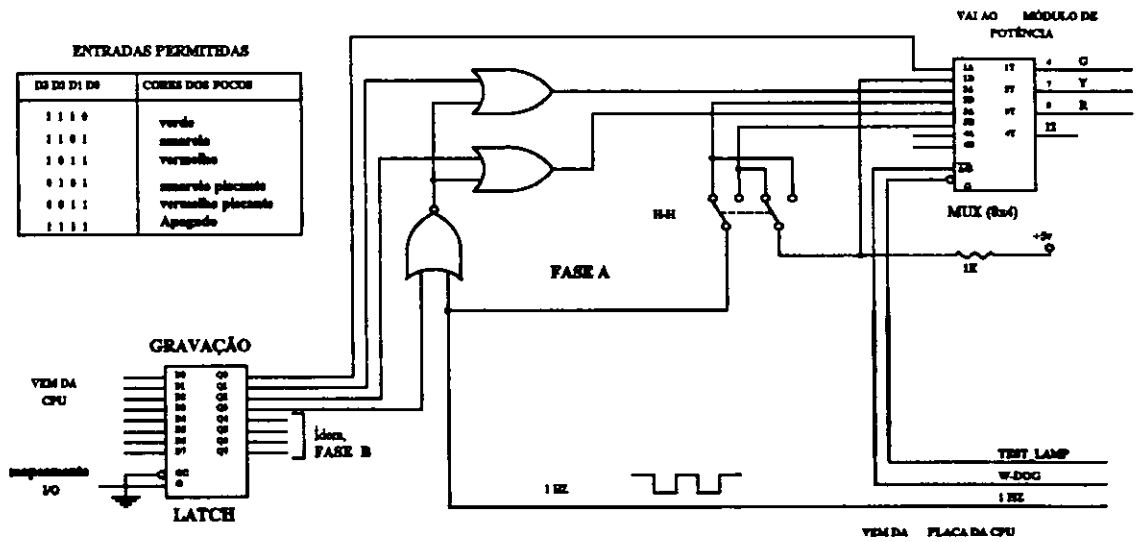


Figura 2.2 - Circuito de interface de fases

Observe que o bit D3 e D7, quando setado, inibe a atuação do piscante na saída. As saídas G, Y, e R acionarão os triacs quando estiverem em nível lógico "0", através dos foto-aclopadores, conforme mostrado na Figura 2.3. Desta forma, apenas uma das saídas estará ativada (G, Y ou R), quando $\overline{A}/B$ e G1 estiverem em nível lógico "0". O MUX de saída possui ainda os sinais "G1" e "$\overline{A}/B$" [TEXAS, 1985], que são comuns a todas as placas de fases, e que não são controlados pela CPU.

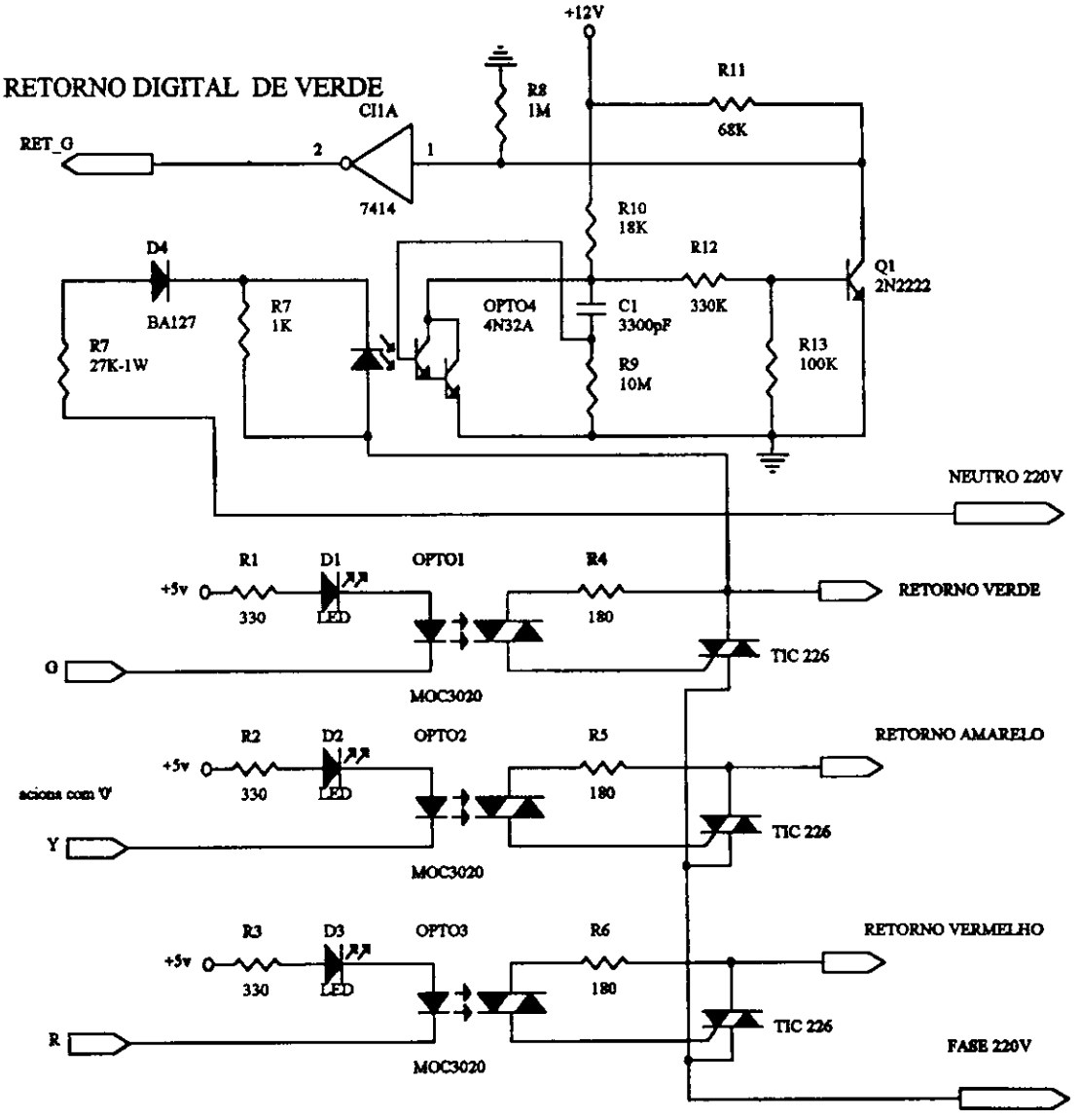


Figura 2.3 - Módulo de potência

O sinal "G1" quando em nível lógico "1", leva todas as saídas a "0", acendendo todas as lâmpadas dos focos. Este sinal é usado para verificar a existência de lâmpadas queimadas, utilizando-se para isso apenas uma chave push-botton.

O sinal "$\overline{A}$/B" está conectado ao sinal W-DOG, que faz o controle e monitoração da CPU. Quando W-DOG = 1, estarão nas saídas os sinais provenientes das entradas B do MUX. Neste caso, todas as fases estarão em amarelo ou vermelho piscante, dependendo da posição da chave H-H de cada placa. Quando W-DOG = 0, as saídas serão monitoradas pelo software de controle.

O circuito da Figura 2.3 também supervisiona as Lâmpadas de verdes, retornando um sinal digital do estado das lâmpadas dos focos verdes, para controle da CPU. Isto é necessário porque a ocorrência de um triac de verde em curto-circuito, causaria sérios problemas ao tráfego. É importante observar que tanto o circuito de acionamento como o de retorno digital são isolados opticamente da tensão de 220 Volts.

2.3.2 - Mapeamento de Memória e I/O

Os endereços das placas de fases foram mapeadas em memória, juntamente com os endereços da EEPROM e do RTC. A Figura 2.4 mostra a decodificação destes endereços [TEXAS, 1985], na placa da CPU. O endereço das fases inicia em "E000", sendo que, cada placa de fases é endereçada na própria placa com auxílio da chave S1, conforme Figura 2.5.

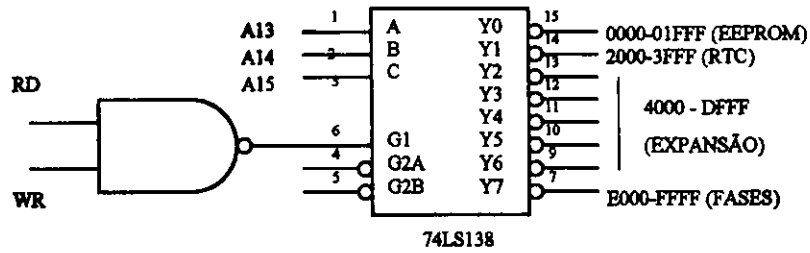


Figura 2.4 - Mapeamento de memória e I/O

O endereço completo de cada placa de fases, que corresponde a duas fases semafóricas, será obtido em função das posições da chave S1 da Figura 2.5, conforme Tabela 2.5. Teremos então CS2, para habilitar o latch de escrita e CS1 para habilitar o latch de leitura, com um mesmo endereço, em cada placa.

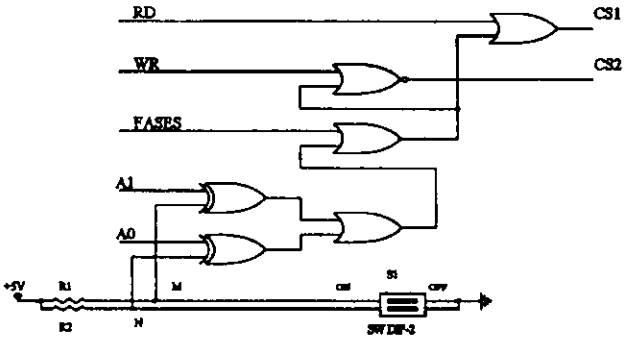


Figura 2.5 - Mapeamento das placas de fases

Tabela 2.5 - Endereçamento final das placas de fases

S1		M	N	Placa	Fases Semafóricas	Endereço (Hexa)
ON	ON	0	0	1	1 e 2	D000
ON	OFF	0	1	2	3 e 4	D001
OFF	ON	1	0	3	5 e 6	D002
OFF	OFF	1	1	4	7 e 8	D003

2.3.3 - Interface Serial

A interface com o usuário dar-se-á via computador padrão IBM-PC, em ambiente MS-DOS, através do programa "CET.EXE". Este programa aciona o controlador via comunicação serial. Em nosso protótipo implementamos o padrão RS-232C [TEXAS,1989], por tratar-se de um equipamento que não estará conectado em rede com outro equipamento. O padrão RS-485 é o recomendado quando for necessário

fazer sincronismo com outro equipamento, principalmente quando tratar-se de distâncias inferiores a dois quilômetros.

O microcontrolador 8031 dispõe de pinos próprios para comunicação serial (sinais RXD, TXD), inclusive protocolos próprios que já possibilita a transmissão de bytes, ao invés de bits. Desta forma, para estabelecer uma comunicação serial no padrão RS-232C, basta fazer a conversão dos níveis de tensões presentes nos pinos 10 e 11 do microcontrolador conforme figura 2.6, e o restante fica por conta do software de controle.

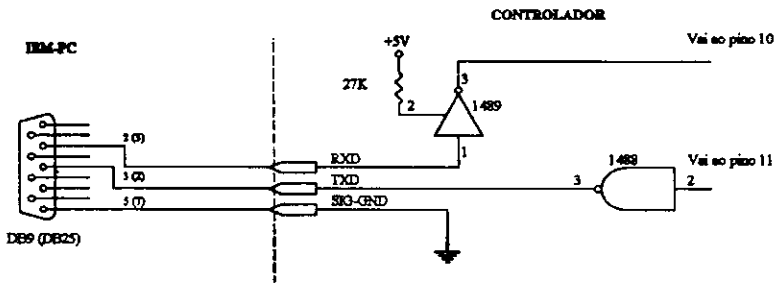


Figura 2.6 - Detalhe das ligações para canal serial

2.3.4 - Relógio de Tempo Real

Este componente implementa um relógio completo com alarme e calendário. Existem no mercado vários RTCs, com diferenças em custo e implementações de funções adicionais, tais como: quantidade de interrupções programáveis, quantidade de RAM interna adicional, bateria embutida no invólucro, etc.

Optamos por trabalhar com o CI MC146818A, principalmente pelo preço e facilidade de aquisição. As principais características deste componente são:

- Relógio e Calendário completos;
- Três níveis de interrupções programáveis;
- Gerador de onda quadrada programável;
- RAM interna adicional de 50 Kbytes;
- Barramento de dados multiplexados, compatível com a CPU 8031;

Este componente possui 64 endereços de RAM interna, sendo que os primeiros dez endereços são reservados para os contadores de tempo e calendário, os quatros endereços seguintes para os registradores de controle e os endereços restantes (50 bytes) para uso geral. A Tabela 2.6 mostra a função de cada byte, para os quatorze primeiros endereços. Os valores nos contadores podem ser gravados e lidos tanto em BINÁRIO com em BCD, conforme inicialização no registrador B. Os endereços referidos na tabela devem ser acrescidos de "2000" (hexa), conforme mostrado na Figura 2.4.

Tabela 2.6 - Endereços do RTC

ENDEREÇOS (HEXA)	DESCRIÇÃO	BINÁRIO (HEXA)	BCD
0	segundos	00 - 3B	00 - 59
1	segundos (alarme)	00 - 3B	00 - 59
2	minutos	00 - 3B	00 - 59
3	minutos (alarme)	00 - 3B	00 - 59
4	horas	00 - 17	00 - 23
5	horas (alarme)	00 - 17	00 -23
6	dia da semana	01 - 07	01 - 07
7	dia do mês	01 - 1F	01 - 31
8	mês	01 - 0C	01 - 12
9	ano	01 - 63	00 - 99
A	Registrador A	leitura / escrita	
B	Registrador B	leitura / escrita	
C	Registrador C	somente leitura	
D	Registrador D	somente leitura	

A Figura 2.6 mostra os sinais envolvidos na conexão da CPU com o RTC. Embora não tenhamos usado bateria em nosso protótipo, isto será necessário no projeto definitivo, para que, com a falta de energia o horário não seja afetado.

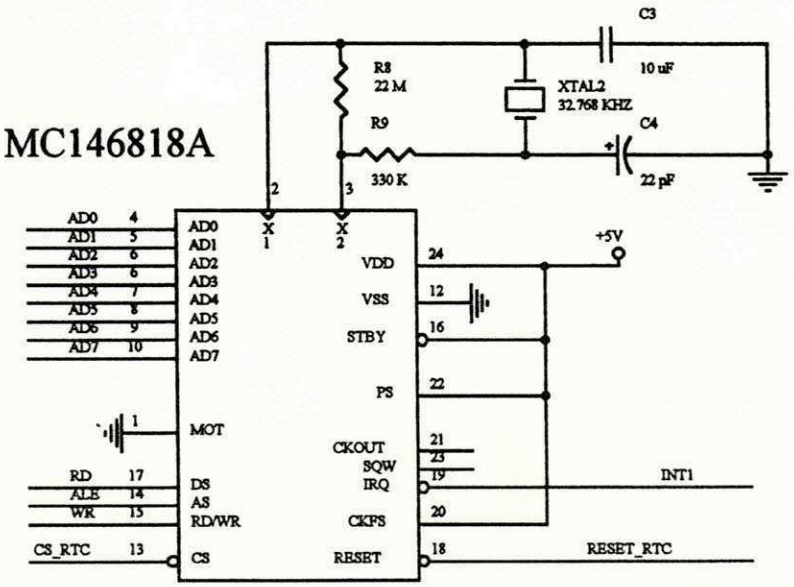


Figura 2.7 - Relógio de tempo real (RTC)

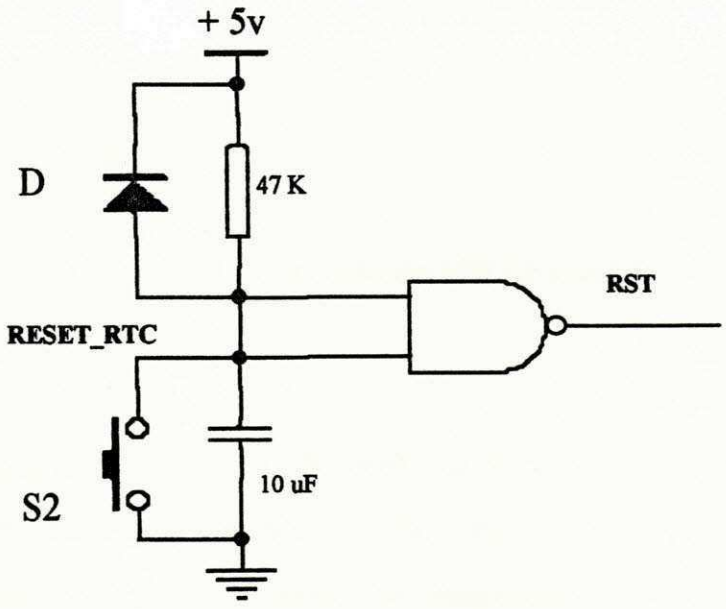


Figura 2.8 - Circuito de reset

Os sinais AD0-AD7, RD, ALE, WR e INT1 são conectados diretamente no microcontrolador, via barramentos. O sinal CS_RT é conectado ao pino 14, do CI 74LS138 da Figura 2.4. O sinal RESET_RTC, não pode ser conectado diretamente ao reset da CPU, pois o mesmo é habilitado em "0". A Figura 2.7 mostra o circuito onde são gerados o reset da CPU (RST) e o RESET_RTC. Ao energizarmos o equipamento, durante a carga do capacitor C, ocorrerá uma reset geral. Isto também pode ser conseguido, manualmente, através da chave S2.

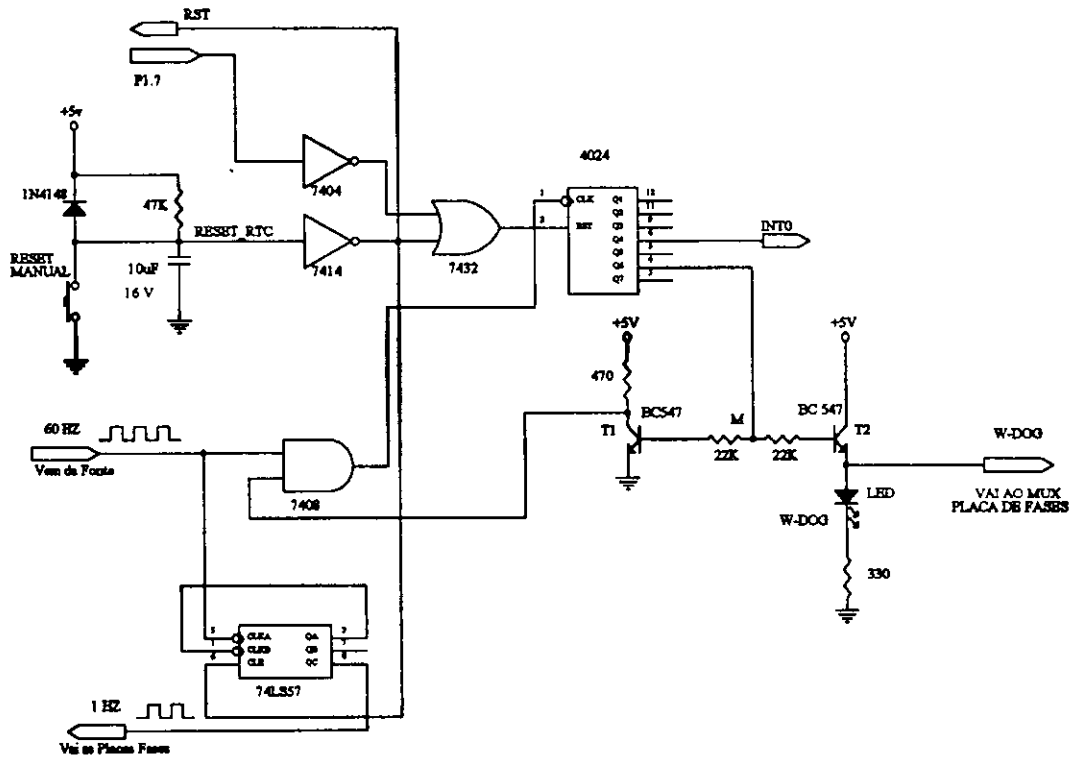


Figura 2.9 - Circuito de cão de guarda

O circuito da Figura 2.9 apresenta o circuito que gera os sinais W-DOG e 1 HZ, presentes no circuito da Figura 2.2. O circuito de Cão de Guarda tem a função de monitorar a CPU, forçando-a a atender periodicamente a interrupção INT0, que deve estar programada, com prioridade máxima. No atendimento a esta interrupção, a CPU deve produzir um pulso em P1.7 (bit 7 da porta P1), resetando o contador 4024,

impedindo-o de atingir sua contagem máxima. Em uma eventual falha da CPU, ao atingir sua contagem máxima, neste caso com M = 1, teremos os transistores T1 e T2 conduzindo. Na condução de T1, leva a saída do CI 7408 a "0", independentemente do trem de pulsos de 60 HZ, permanecendo neste estado indefinidamente enquanto M=1, parando a contagem do contador. Na condução de T2, o led de W-DOG é acionado levando W-DOG = 1, afetando as saídas do MUX da Figura 2.2, forçando amarelo ou vermelho piscante nos focos. A condição de funcionamento normal só será restabelecida forçando-se um reset geral, manualmente ou desligando e ligando a fonte de energia.

Este circuito contém em seu bojo o circuito da Figura 2.7, inclusive a opção de reset manual.

2.4 - Capacidade de Expansão

O detalhamento completo das placas da CPU e de fases podem ser vistas nos Anexos 1 e 2, respectivamente. Observe que o cabo que interliga a placa da CPU com todas as placas de fases é único, possuindo 18 vias, contendo os seguintes sinais: AD0-AD7, RD, WR, A0, A1, W-DOG, T_LAMP, 1 HZ, MAP1(FASES), +5V e GND.

Se incluirmos neste cabo o sinal A3 e substituirmos a chave de endereçamento das placas de fases por uma outra de três posições, estaremos expandindo a capacidade de 4 (quatro) para 8 (oito) placas de fases, ou para até 16 (dezesseis) fases semafóricas. Evidentemente que com dezesseis fases semafóricas, teremos que alterar o formato dos dados na EEPROM e conseqüentemente fazer algumas modificações nos software de controle e da interface.

Esta possibilidade de expansão pouco influi no custo do equipamento, porém a prática tem mostrado que um equipamento com mais de oito fases semafóricas somente seria utilizado em cruzamentos muito complexos ou em vários cruzamentos muito próximos entre si, ou seja, situações raras.

Quando utilizamos um equipamento para controlar mais de três cruzamentos devemos considerar aspectos tais como: custo do controlador, custo dos cabos de ligações do equipamento aos portafocos nos cruzamentos, a queda de tensão em função da distância do controlador a cada cruzamento e a robustez do controle (uma falha no controlador afeta muitos cruzamentos). Neste caso, o uso de mais de um controlador deve ser considerado no planejamento.

2.5 - Considerações sobre as Ferramentas Utilizadas

Optamos por montar nosso protótipo em placas universais com fios e soquetes em WIRE-WRAP, devido a facilidade de se fazer eventuais mudanças durante o período de testes.

Para a elaboração dos diagramas elétricos, diagramas de ligações e relação de material, tínhamos a nossa disposição as seguintes ferramentas de software: ORCAD, PROTEL e TANGO.

Todos eles com documentação razoável, com interfaces semelhantes e bibliotecas de componentes satisfatórias. Como não pretendíamos confeccionar o projeto final da placa, com lay-out dos componentes e detalhes de ligações, qualquer uma destas ferramentas poderiam ser adotada. Optamos por trabalhar com o *Tango*, dado a existência de uma cultura local com esta ferramenta.

SOFTWARE BÁSICO

Neste capítulo abordaremos sobre a estrutura do software de controle desenvolvido para o controlador eletrônico semafórico a tempo fixo, bem como, suas rotinas principais. Abordaremos também sobre as ferramentas de software disponíveis, para trabalhar com o microcontrolador adotado.

3.1 Considerações sobre as Ferramentas Disponíveis

Um dos parâmetros adotados para optarmos pelo microcontrolador Intel 8031, foi, sem dúvida, a existência de várias ferramentas de software disponíveis para esta pastilha, provavelmente por ter sido uma das primeiras lançadas no mercado. Entre estas ferramentas podemos citar:

1 - Placas de desenvolvimento;

2 - Emuladores em ambiente MS-DOS e/ou UNIX;

3 - Compiladores C e assembly;

4 - Assembladores e desassembladores diversos;

5 - Ambiente de desenvolvimento completos: Editor, Compiladores C e Assembly, Link, Debugador e Conversores para formato Intel e Binário;

Estas ferramentas estão disponíveis no mercado, porém tivemos acesso apenas a três delas:

1 - Emul51 - Emulador de Microcontrolador 8051

2 - C- COMPILER-51

3 - Avcase

O EMUL51 é uma placa com sistema de desenvolvimento e depuração baseado em microcontroladores da família 8051, em ambiente MS-DOS. Esta placa foi

desenvolvida na UFPB com orientação do prof. Misael Elias de Morais, podendo ser instalada em qualquer slot de 8 Bits, em computadores compatíveis com IBM-PC. Neste ambiente é possível carregar e rodar um programa, com recursos de interromper e continuar de onde tinha parado, marcar breakpoints por software, executar passo a passo, visualizar conteúdos dos registradores e das memórias podendo inclusive alterá-los, "desassemblar" as instruções e diversas outras facilidades. O inconveniente deste produto para o nosso projeto é o fato da placa utilizar a interrupção INT1 para interface com o ambiente de desenvolvimento, portanto tornando esta interrupção indisponível para os projetos de usuário.

No pacote de software C-COMPILER-51 da Keil Elektronik GmbH estão incluídos: C-COMPİLER, Assembly-Compiler, Linker / Locater, Library-Manager and Include's, Object-File-Converter e Support-Shell. Esta versão foi concebida para ambiente MS-DOS, atendendo satisfatoriamente todos os microcontroladores da família MCS-51 da Intel.

No pacote de software Avcase distribuído pela Anacon, entre todos o mais difundidos, estão incluídos basicamente todos os módulos do pacote anterior, porém com uma documentação invejável e com atenção especial a uma nova roupagem ao ambiente de depuração, tornado este software extremamente atrativo.

Optamos por trabalhar com o C-COMPILER-51, novamente devido a existência de uma cultura local, o que facilita o esclarecimento de eventuais dúvidas ao longo do trabalho.

3.2 Programa principal

Quando trabalhamos com linguagem C, cujo código é formado por um conjunto de funções, por convenção o programa principal ou função principal é aquela função denominada *main()*, porque as demais função são disparadas a partir desta.

Nesta função são inicializadas todas as variáveis, as interrupções, e os parâmetros para a comunicação serial e em seguida fica em *loop*, aguardando a ocorrência de uma eventual interrupção para atendimento ou uma solicitação para comunicação serial.

A Figura 3.1 mostra o *Diagrama estruturado* [Schnupp. P et al, 1978 & Arakaki. R et al, 1990] para o programa principal, onde neste caso, no laço principal a CPU fica sujeito ao atendimento de comunicação serial e das interrupções programadas. Observe que existem duas opções de saídas possíveis do laço principal (loop2):

1 - Quando na rotina de comunicação serial for alterarado os dados da eeprom (gravação de novos dados ou modificações dos dados existentes) ou quando for alterado os dados no RTC (alterado o horário). Neste caso, retorna ao início do loop1;

2 - Quando for verificado no início de cada trecho, após atualizar as cores dos focos, a ocorrência de verdes conflitantes (a configuração de verdes acesos não passa pelo crivo da tabela de conflitos). Neste caso, pela gravidade do fato devido aos danos que podem ser causados ao tráfego, o programa é abortado saindo dos laços 1 e 2, entrando no módulo piscante e travando. Esta condição apenas será alterada com o *reset geral*, que pode ocorrer manualmente ou com a falta de energia.

No protótipo inicial não foi implementado o circuito de w-dog que tem a função de supervisionar o funcionamento da CPU e gerar o sinal pulsante de 1 Hz para produzir o amarelo e vermelho piscante. Neste caso, o sinal pulsante é gerado através da interrupção INT0 no pino P1.6 do microcontrolador e pino de controle de w-dog do MUX de saída das placas de fases é conectado ao pino P1.5 do microcontrolador, que no início do programa principal é aterrado. Ao se implementar no hardware o circuito de w-dog, no software basta modificar a inicialização da variável global *w_dog_presente* de "0" para "1".

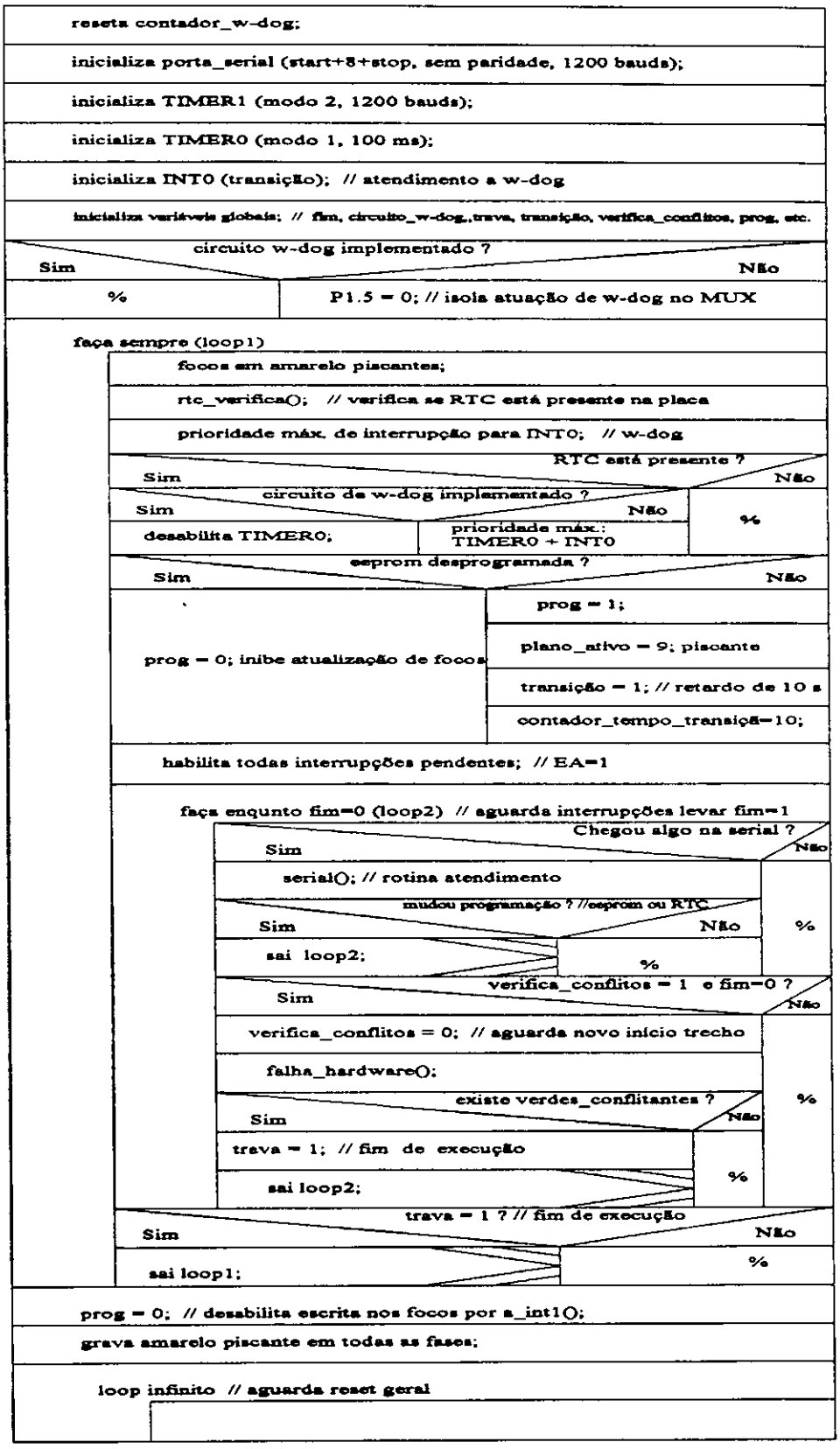


Figura 3.1 - Programa principal

Foi considerado também a alternativa de funcionamento do controlador sem a implementação do RTC, neste caso, não é gerada a interrupção via RTC através de INT1 para atualização dos focos, conforme Figura 3.2. A interrupção que deve ocorrer a cada segundo é gerado pelo contador TIMER0 do microcontrolador ao final da contagem de dez interrupções de 100 ms, conforme mostrado na Figura 3.4.

A função rtc_verifica() é responsável pela verificação da presença do RTC na placa, veja a Figura 3.3. Observe que quando o RTC não estiver presente na placa, o atendimento a INT1 fica desabilitado.

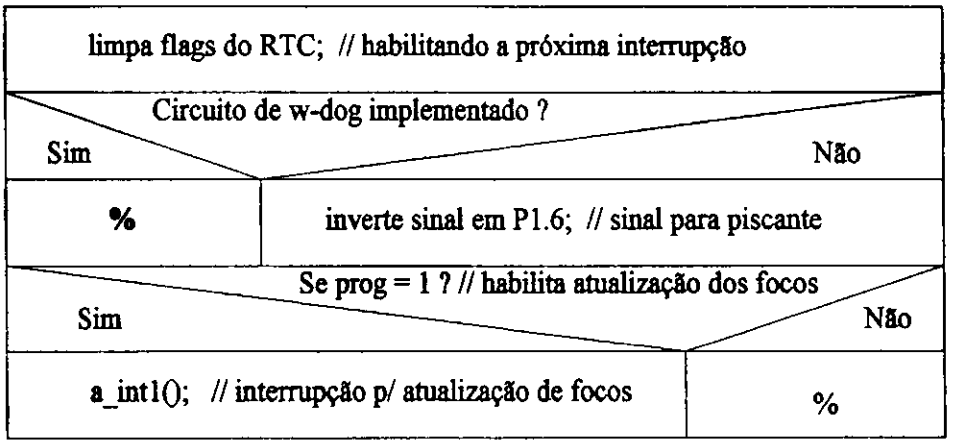


Figura 3.2 - Rotina de atendimento a interrupção INT1

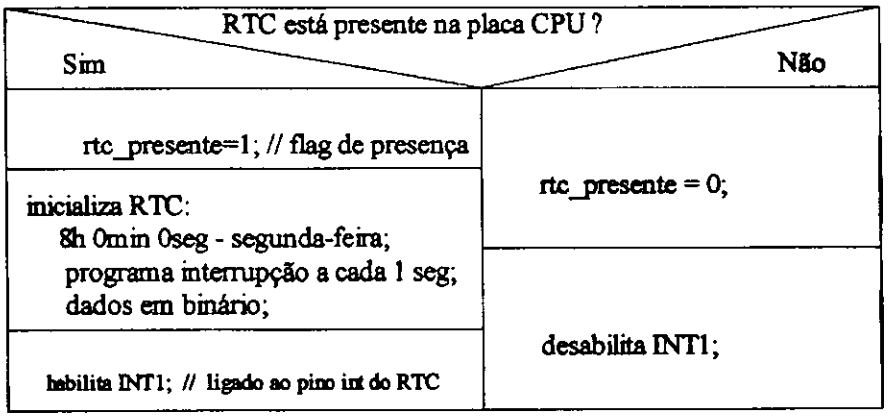


Figura 3.3 - Rotina rtc_verifica()

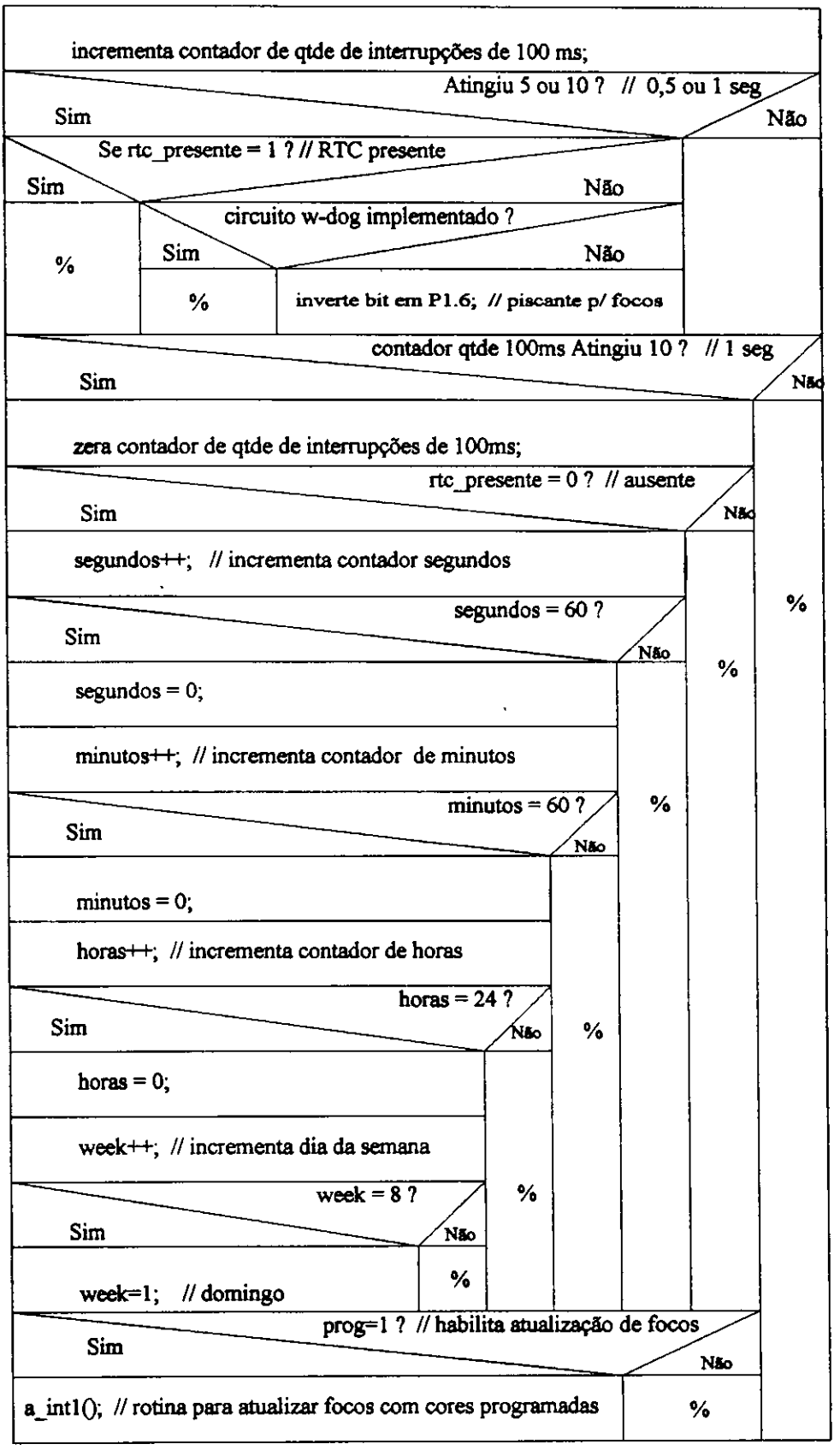


Figura 3.4- Rotina de atendimento a interrupção TIMER0

3.3 - Rotinas Principais

A seguir apresentaremos as principais rotinas ou funções utilizadas no sistema. Como vimos no programa principal, no loop principal (loop2), o sistema fica aguardando uma solicitação para inicializar uma comunicação serial ou a ocorrência de alguma das seguintes interrupções:

- INT0 ➡ para zerar contador de w-dog;
- INT1 ➡ para atendimento à interrupção gerada pelo RTC;
- TIMER0 ➡ para simular relógio e gerar sinal pulsante;

A solicitação ao atendimento a interrupção gerada por INT0, ocorre quando o circuito de w-dog está presente no circuito e o contador atingiu seu valor crítico, conforme indicado na Figura 2.9. Nesta rotina a CPU apenas reseta este contador forçando um pulso através do pino P1.7.

O atendimento das interrupções INT1 e TIMER0 mostrados nas Figuras 3.2 e 3.4 respectivamente, tem o mesmo objetivo, que é acionar a rotina de atendimento a cada segundo (*a_int1()*) se a atualização de focos estiver habilitado (prog=1), além de gerar o sinal pulsante para as placas de fases. Observe que apenas uma destas interrupções aciona a rotina *a_int1()*, em função da presença do RTC na placa.

3.3.1 - Rotina de Interrupção a cada segundo (*a_int1()*)

Esta rotina é responsável pela atualização dos focos a cada segundo, em função do plano que está em operação. Ao final de cada plano é verificado se existe troca de planos programados, em função da tabela de troca e do horário atual gerado pelo RTC ou simulado pelo microcontrolador através da interrupção TIMER0.

A Figura 3.5 mostra o *Diagrama estruturado* desta rotina. A Figura 3.6 mostra o diagrama para a função troca_planos(), que poderia estar incluído na Figura 3.5. Apresentamos em separado para não carregar muito o primeiro diagrama.

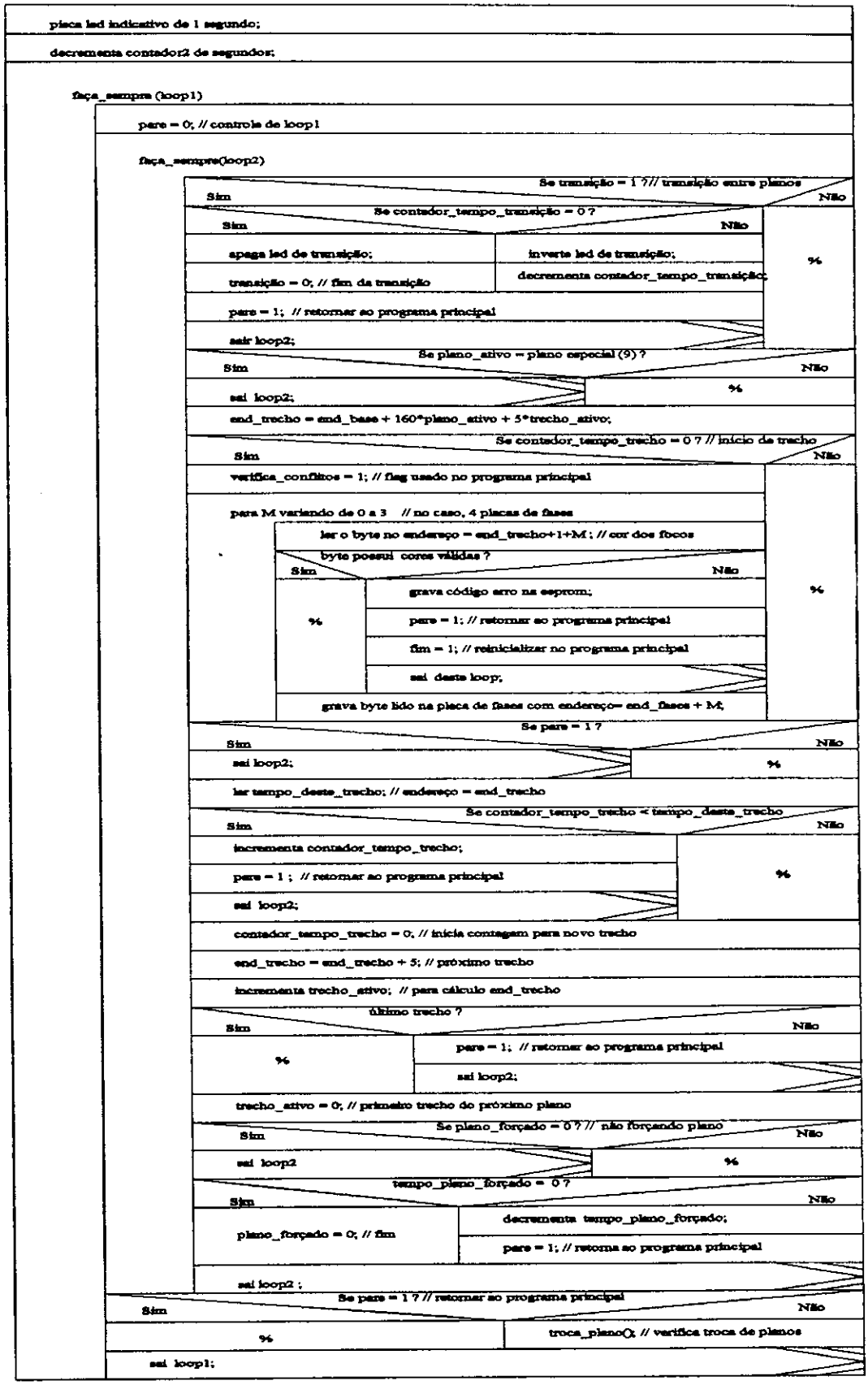


Figura 3.5 - Rotina a_int1()

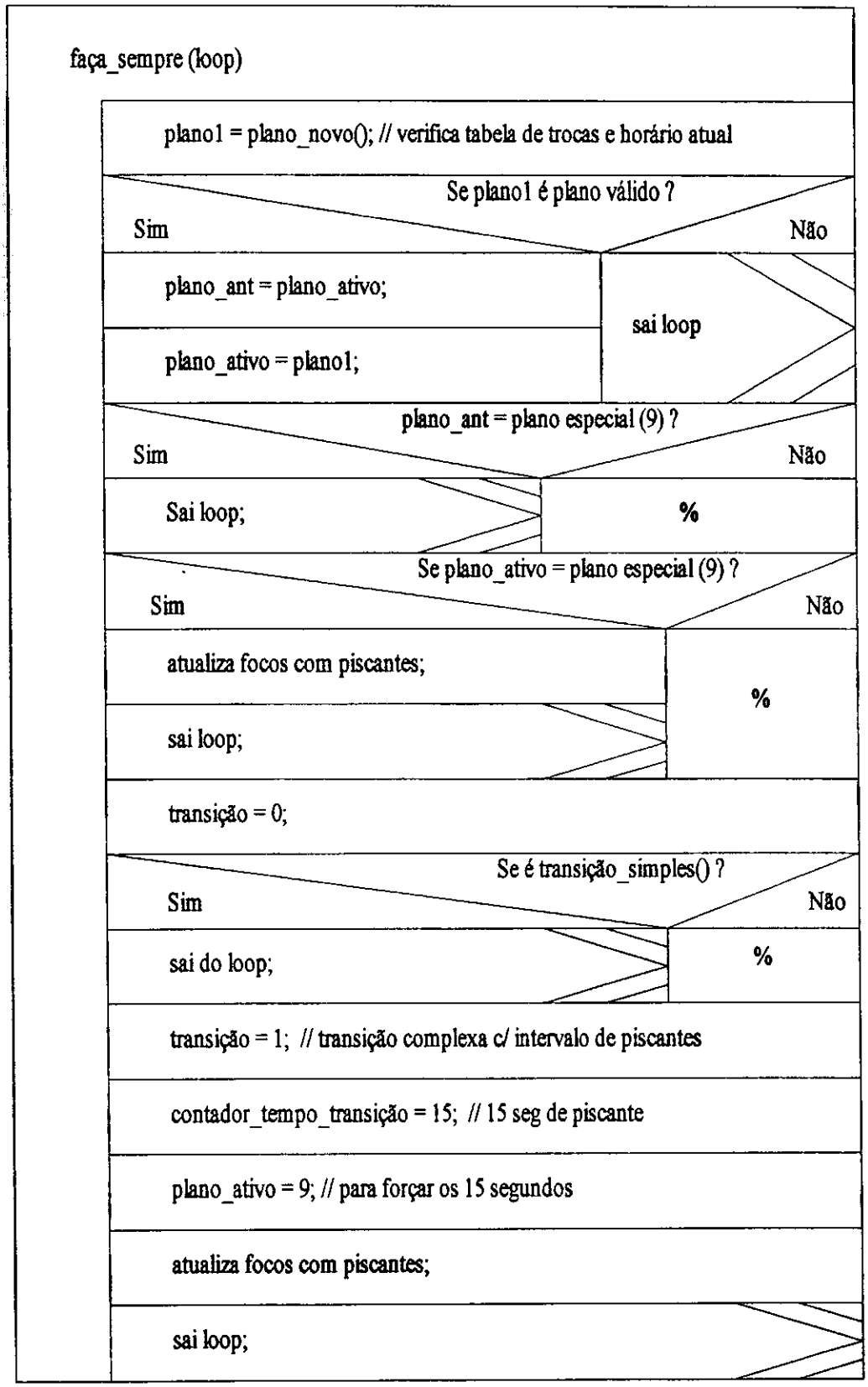


Figura 3.6 - Rotina troca_plano()

Tabela 3.1 - Descrição de variáveis

Variável	Rotina	Descrição
prog (bit)	atendimento INT1 atend. TIMER0	permite a chamada a *a_int1()*, para atualização dos focos
trava (bit)	principal	coloca os focos em amarelo piscante e não muda mais os focos;
fim (bit)	principal, a_int1()	flag de controle do loop2 do programa principal;
um_seg (bit)	principal a_int1()	flag para fazer a verificação de falhas de hardware, apenas após a atualização de focos;
transição (bit)	principal a_int1()	inibe atendimento a *a_int1()*, enquanto o contador_tempo_transição > 0
plano_ativo (char)	principal a_int1()	plano atual em operação, quando igual a 9 trata-se do plano piscante, e no atendimento a *a_int1()* é verificado apenas se ocorrerá nova troca de plano.
trecho_ativo (bit)	a_int1()	trecho atual em operação do plano_ativo. Os focos das fases estão com estas cores.
tempo_deste_trecho (char)	a_int1()	tempo de duração deste trecho em segundos, em que suas cores estarão presentes nos focos.
contador_tempo_trecho (char)	a_int1()	contador de tempo para controle de mudanças do trecho ativo
rtc_presente (bit)	rtc_verifica(), principal, a_int1()	indica a presença do chip RTC na placa: rtc_presente=1 ⇨ presente
pare (bit)	a_int1()	flag de controle loop1

Nos diagramas apresentados aparecem algumas variáveis definidas como globais que são utilizadas para controle dos laços, que são: transição, pare, um_seg, fim, plano_forçado, prog, rtc_presente e trava. Estas variáveis por assumirem apenas dois valores: 0 e 1, podem ser definidas do tipo *bit*, para ocuparem pouco espaço na memória RAM interna do microcontrolador. A Tabela 3.1 descreve a função de algumas varáveis já mencionadas:

Observe que no programa principal a variável *transição* é inicializada em 1 com a variável *contador_tempo_transição* = 10 e *plano_ativo*=9. Isto significa que ocorrerão dez atendimentos a *a_int1()*, correspondendo a 10 segundos (neste protótipo), em que as cores dos focos não serão modificadas e no último atendimento chamará a função *troca_plano()*. A função troca_plano() verificará o horário atual (RTC ou simulador) e a tabela de trocas atualizando a variável plano_ativo com o plano que entrará em execução no próximo atendimento a *a_int1()*. Este tempo é necessário na instalação e manutenção do equipamento para acomodação do tráfego neste período.

A rotina troca_plano() após verificar qual será o próximo plano que deve entrar em operação através da rotina *plano_novo()*, analisa a transição das cores dos focos de cada fase do último trecho do plano anterior e as cores do primeiro trecho do novo plano. Caso ocorra uma transição não permitida, por exemplo: verde para vermelho pulando o amarelo, é implementado uma transição entre planos com piscantes em todas as fases por 15 segundos. As Figuras 3.7 e 3.8 mostram os diagramas das funções plano_novo() e transicao_simples().

Para facilitar a visualização da Figura 3.7, parte dela, foi mostrada separadamente nas Figuras 3.7a e 3.7b. Para verificar a tabela de trocas, no final de cada plano, devemos inicialmente fazer a leitura no relógio (RTC ou simulado) e verificar se existe troca programada para aquele dia da semana que deve entrar naquele momento. Se não existir, então é verificado nos dias anteriores, qual deveria ser a *última troca programada.*

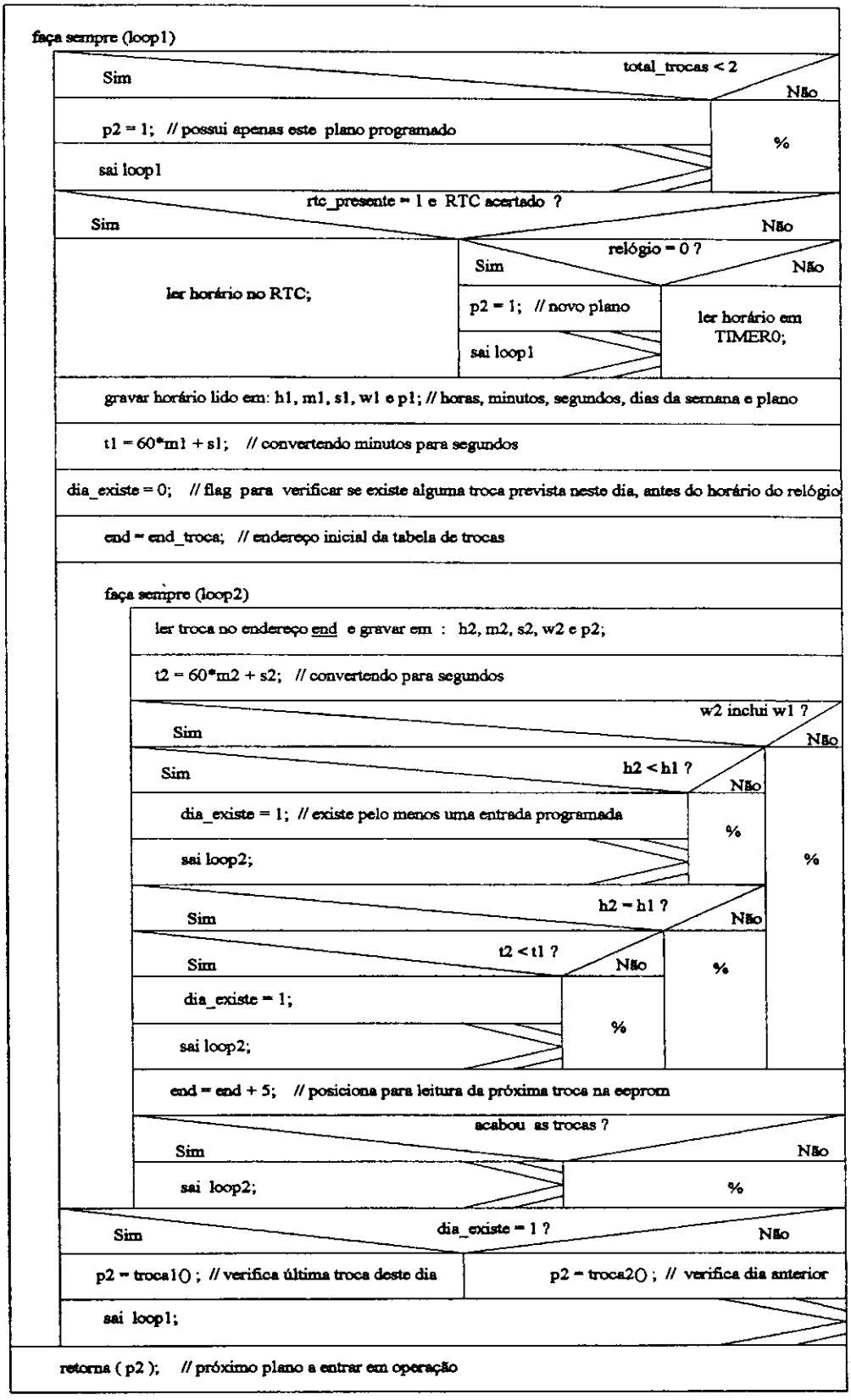


Figura 3.7 - Rotina plano_novo()

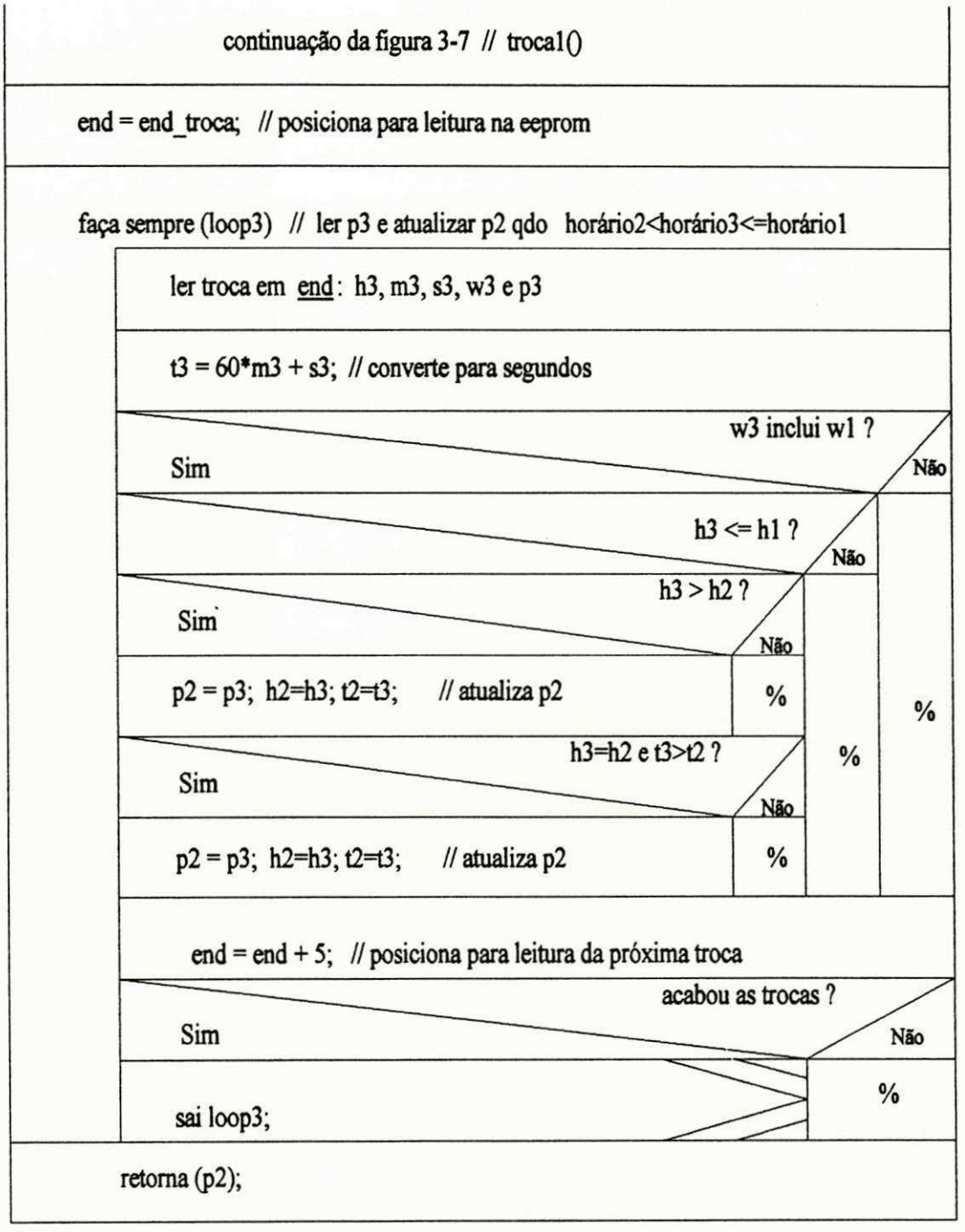


Figura 3.7a - Rotina troca1()

Quando o RTC estiver presente na placa ou quando a CPU estiver simulando o relógio através de TIMER0, porém, não tenha sido acertado o horário, através da interface, não haverá troca de planos. Neste caso o plano ativo será sempre o plano 1.

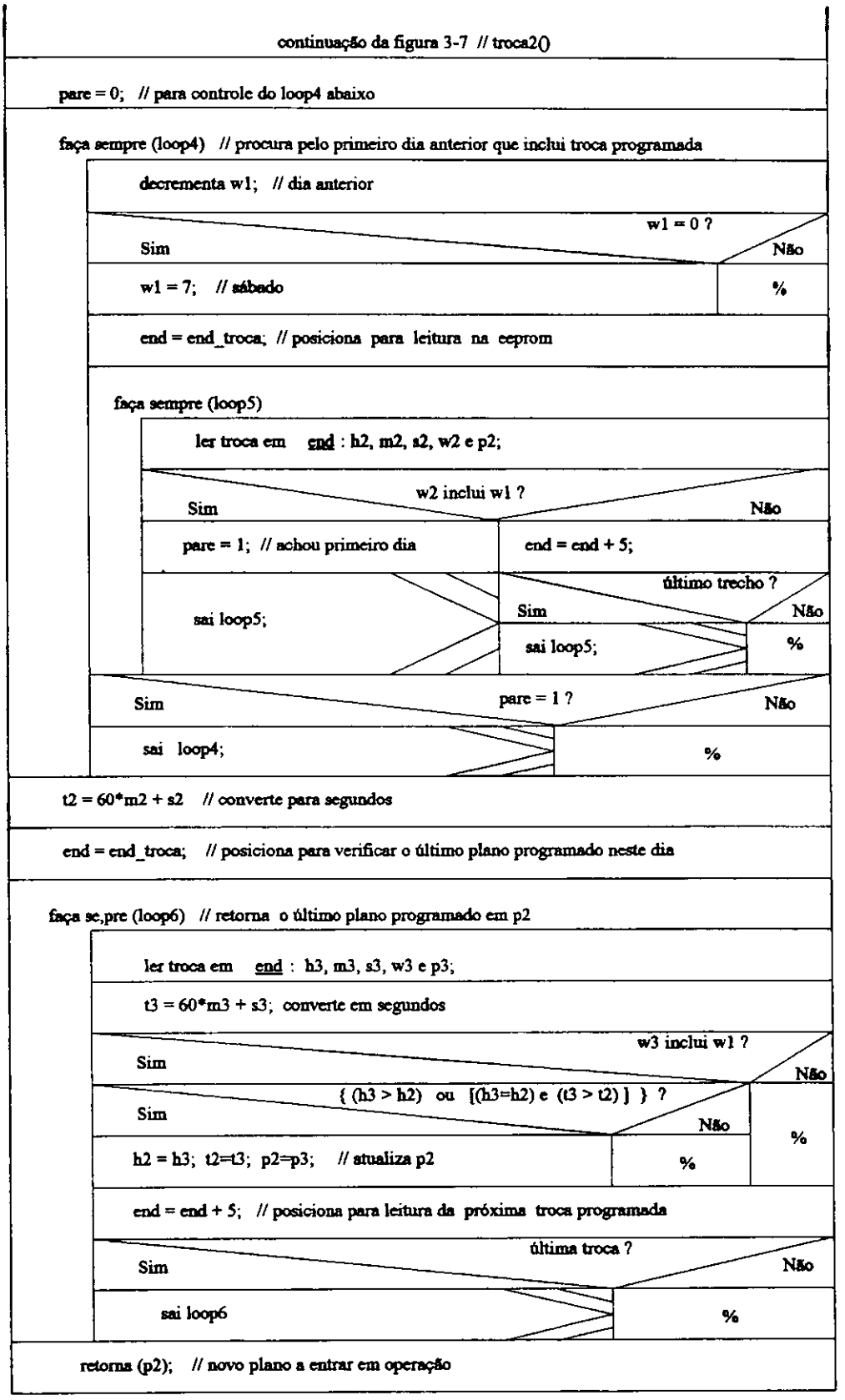


Figura 3.7b - Rotina troca2()

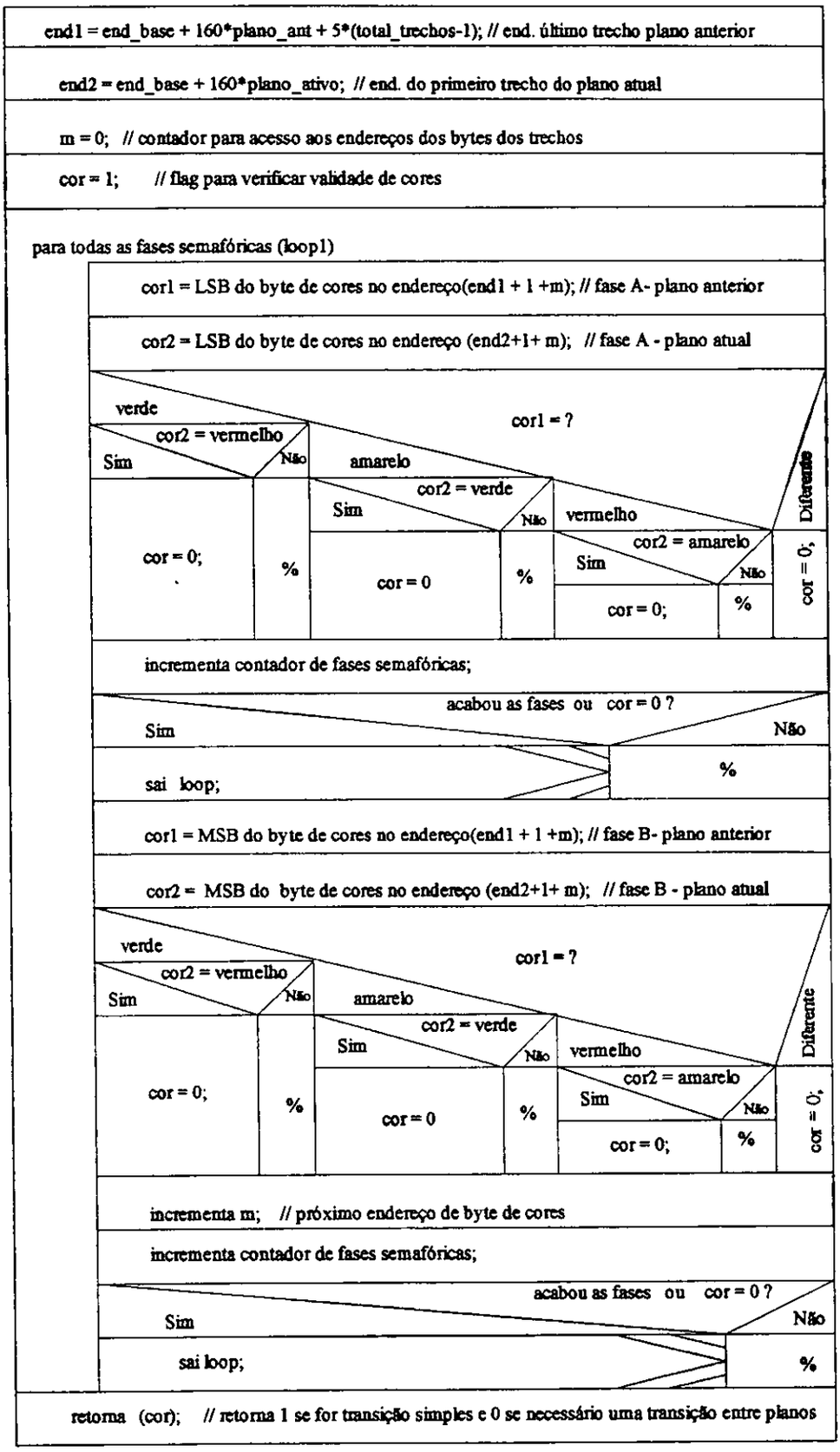


Figura 3.8- Rotina transição_simples()

3.3.2 - Rotina de Atendimento a Comunicação serial

Na inicialização da CPU, a porta serial é configurada para modo2, transmissão assíncrona de 8 bits (start + 8 + stop), sem paridade, com velocidade de 1200 bauds, e o flag de interrupção serial (**ES**) do registrador IE - *Interrupt Enable* é desabilitado.

Toda vez que chegar algum byte no canal serial, ao final da recepção, o bit **RI** do registrador de controle SCON - *Serial Port Control* da CPU é setado. Quando isto ocorrer, vimos da Figura 3.1 que a função **serial()** é chamada. Observe que uma comunicação serial sempre é iniciada com o outro equipamento, desta forma podemos dizer que o controlador está se comunicando no modo escravo.

A Figura 3.9 representa o *Diagrama estruturado* para esta função. Quando chega um byte pela porta serial, é verificado o valor deste byte e chamada uma das rotinas requeridas.

A Figura 3.10 representa a rotina para gravação na serial, onde é passado como parâmetro apenas o byte a ser enviado pela serial. A Figura 3.11 representa a rotina para leitura de byte na serial, onde retorna o byte lido. Estas rotinas são usadas em todas as rotinas mencionadas na Figura 3.10, que são acionadas em função da solicitação do software da interface que está em execução no IBM-PC, através do primeiro byte recebido na serial pelo controlador.

A rotina *ler_eeprom()* faz a leitura da programação presente na eeprom e envia através do canal serial para a interface. A Figura 3.13 mostra o diagrama desta rotina. Para ocorrência de erros de transmissão utilizamos o protocolo de repetição do mesmo byte transmitido, com o byte **F1** abortando a comunicação.

A rotina *grava_eeprom()* faz a gravação da programação dos planos de tráfego, tabela de troca de planos e cabeçalho do arquivo da interface na eeprom. A Figura 3.14 mostra o diagrama desta rotina. Esta rotina faz uso da rotina *g_eeprom()*, mostrada na Figura 3.12, que grava apenas um byte em um determinado endereço, e fica aguardando a eeprom realizar sua gravação interna através do *bit data polling* [Xicor, 1990].

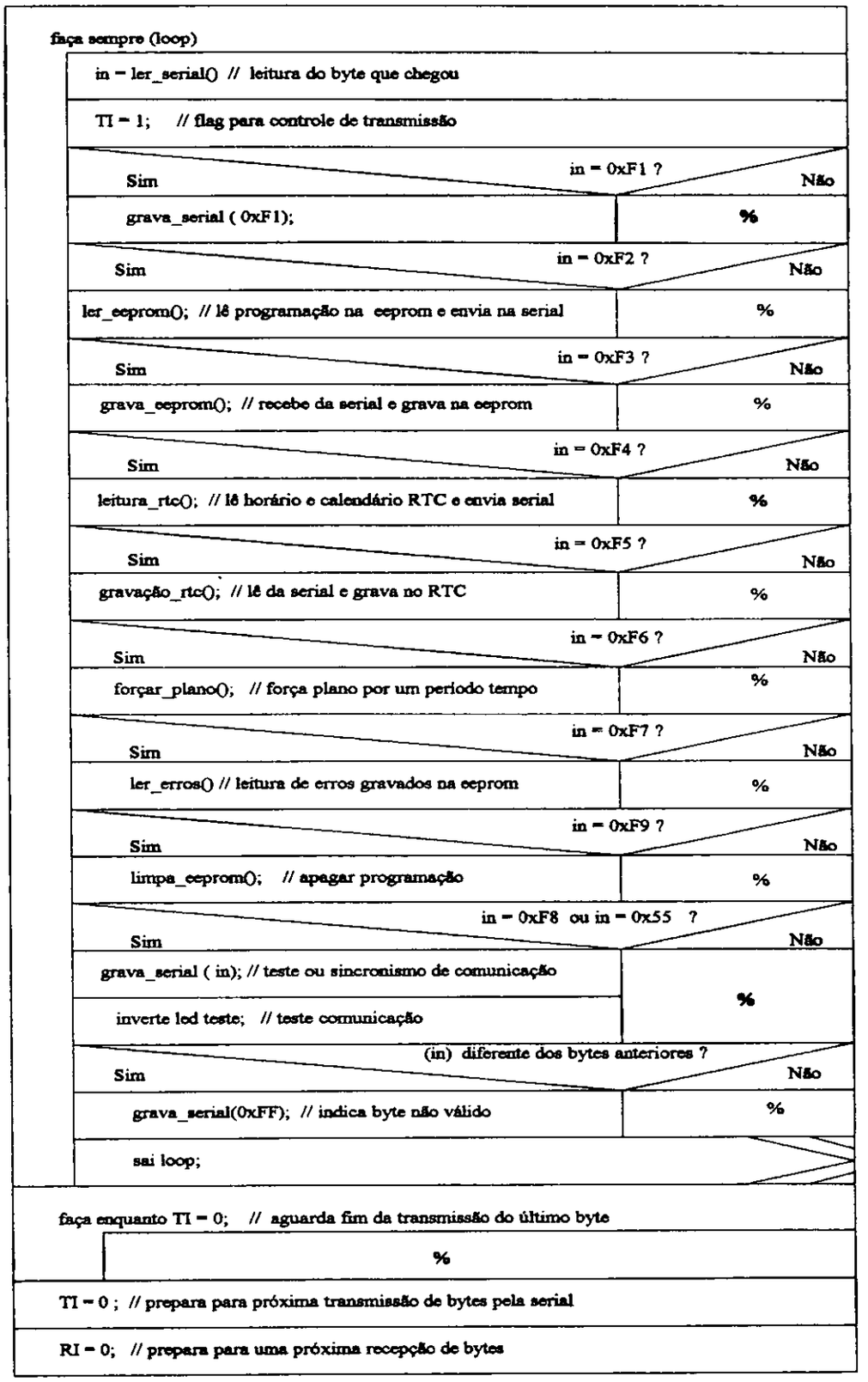


Figura 3.9- Rotina serial()

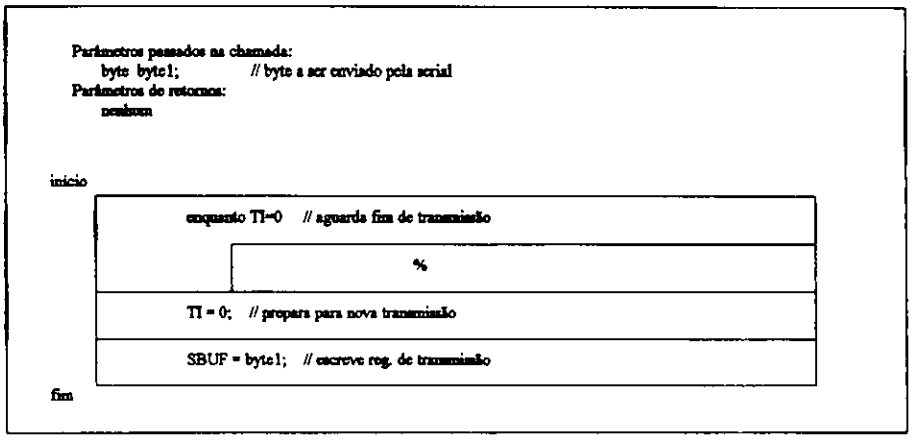


Figura 3.10 - Rotina grava_serial()

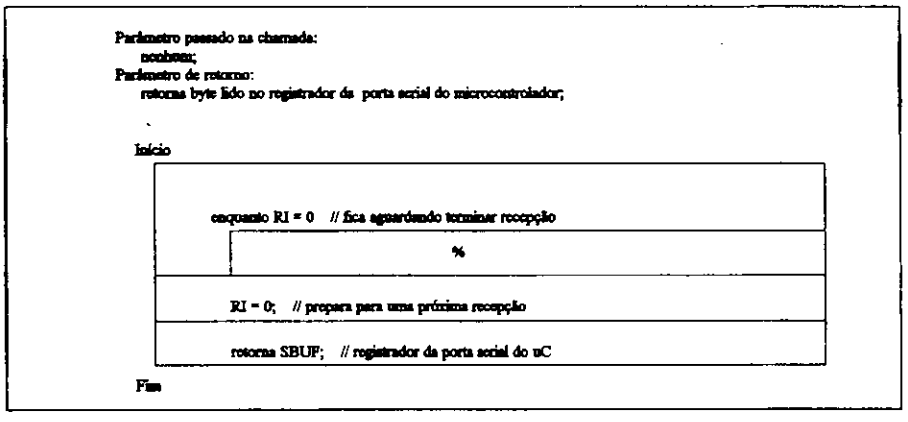


Figura 3.11- Rotina ler_serial()

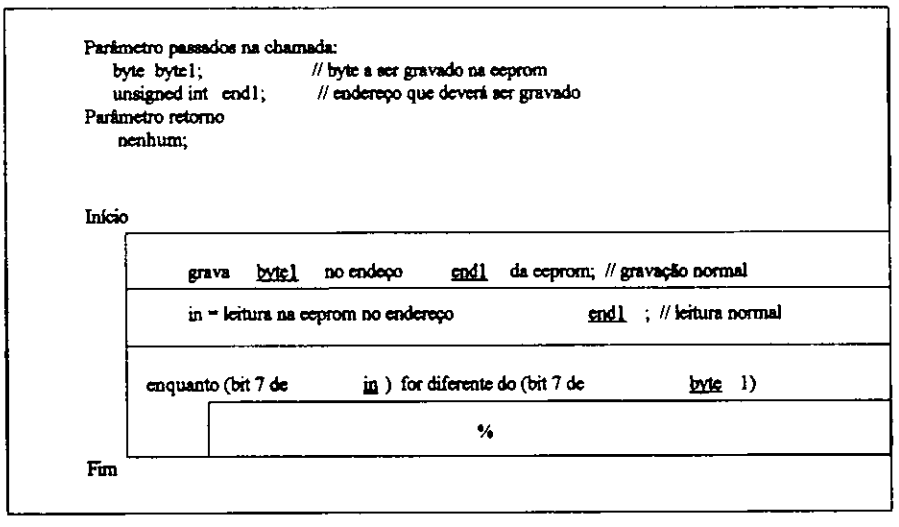


Figura 3.12 - Rotina g_eeprom()

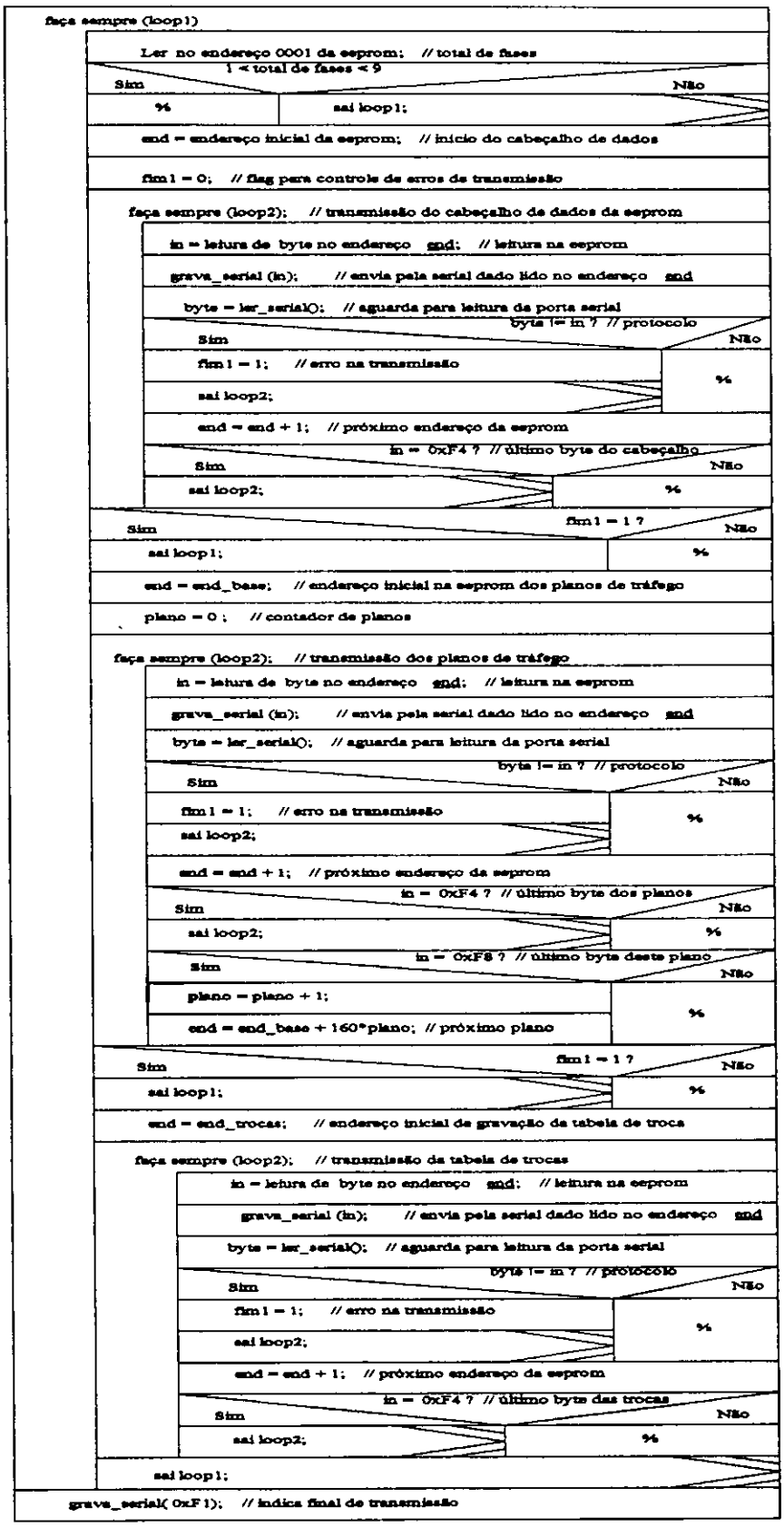


Figura 3.13 - Rotina ler_eeprom()

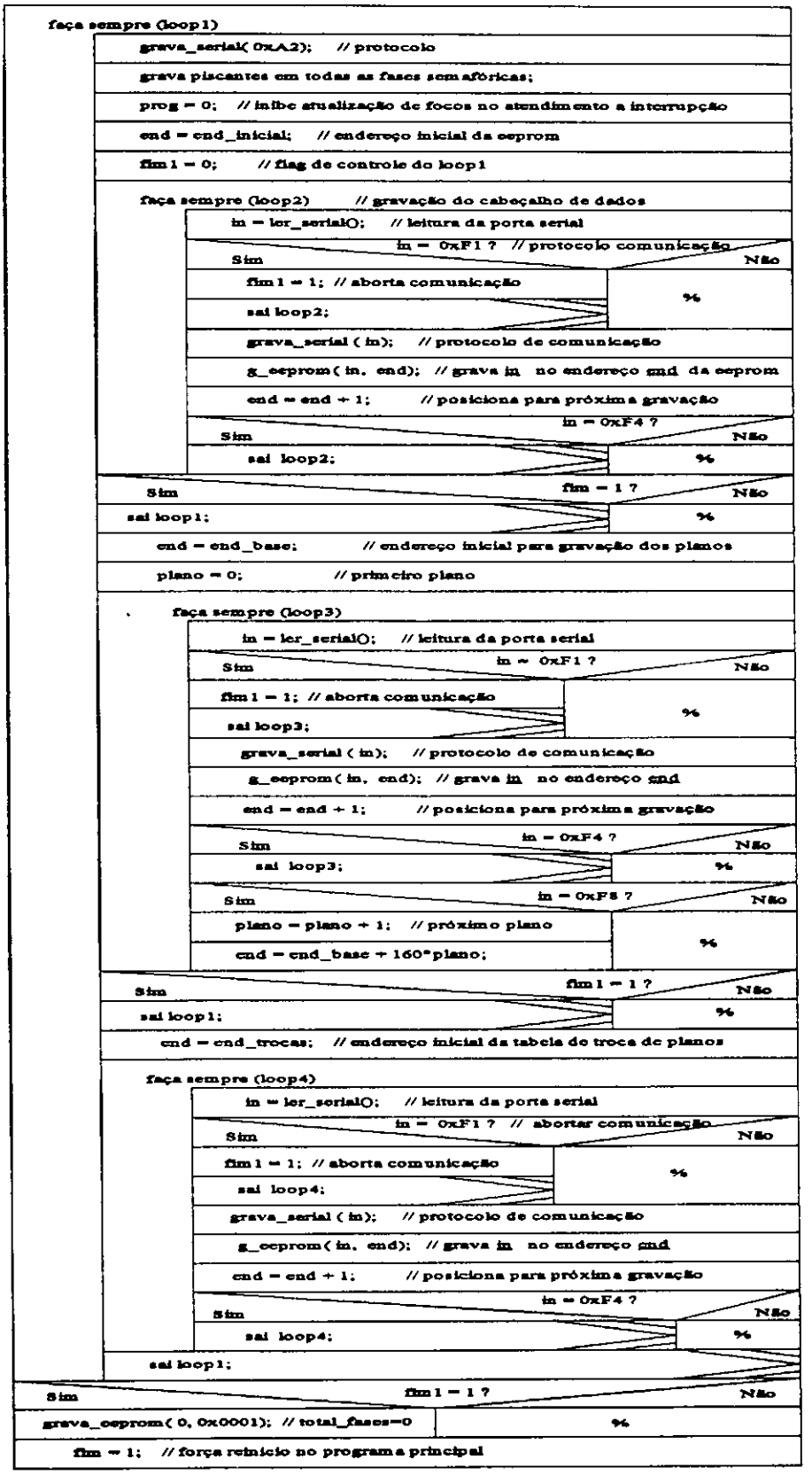


Figura 3.14 - Rotina grava_eeprom()

A rotina *grava_rtc()* é mostrada no diagrama da Figura 3.15. Esta rotina verifica se o *chip* (RTC) está presente na placa, e em caso afirmativo, faz a atualização do seu relógio de acordo com o relógio do IBM-PC. Quando na placa da CPU não existir o RTC, a atualização das variáveis para simulação de relógio pela CPU é feita através da interrupção TIMER0.

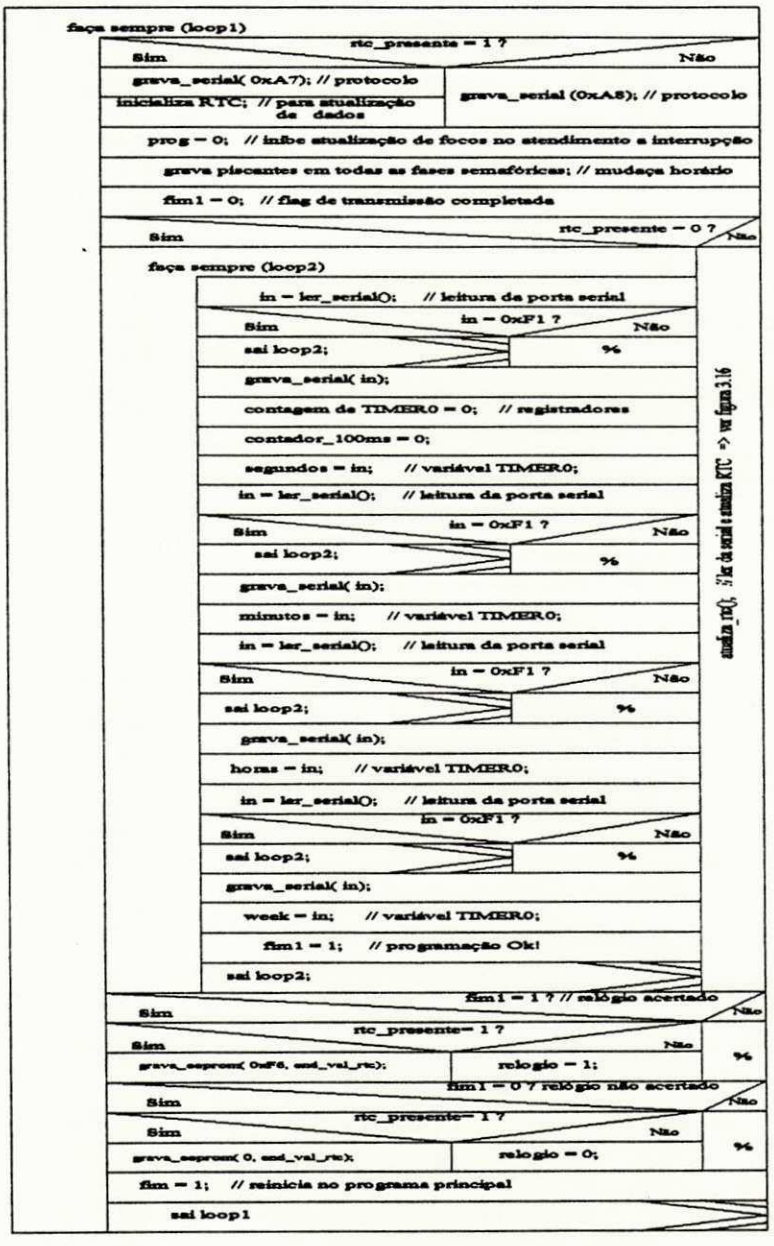


Figura 3.15 - Rotina grava_rtc()

O diagrama da Figura 3.15 faz chamada a rotina *atualiza_rtc()*, para atualização do relógio e calendário do RTC. Esta rotina na verdade pode estar inserida no diagrama anterior, porém para facilitar a visualização desta , a rotina atualiza_rtc() é mostrada na Figura 3.16.

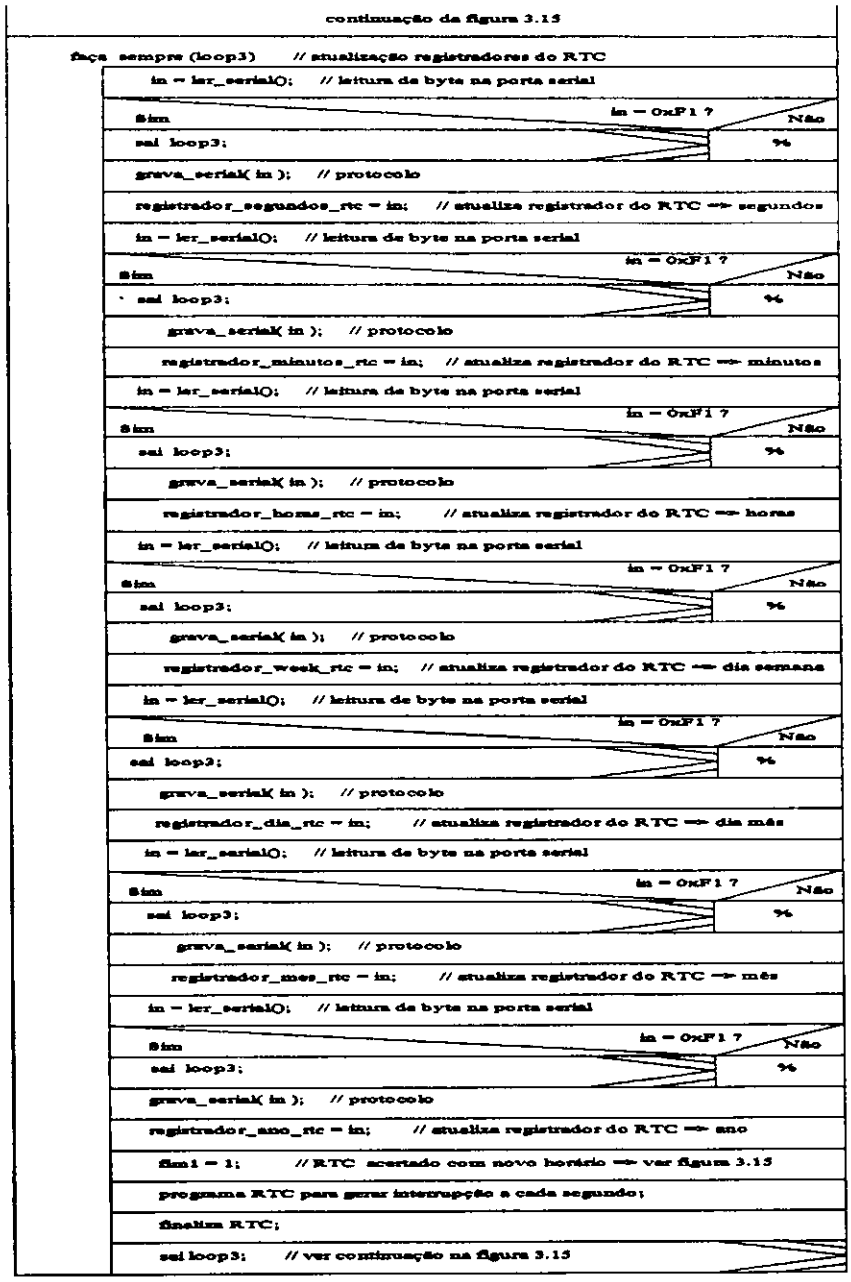


Figura 3.16 - Rotina atualiza_rtc()

A rotina *leitura_rtc()* faz a leitura dos registradores de tempo e calendário do RTC, quando este estiver presente na placa, e envia estes dados para a porta serial. Quando o relógio for simulado pela CPU, é enviado apenas o conteúdo das variáveis segundos, minutos, horas e week , presentes no atendimento a interrupção TIMER0. A Figura 3.17 mostra o diagrama desta rotina.

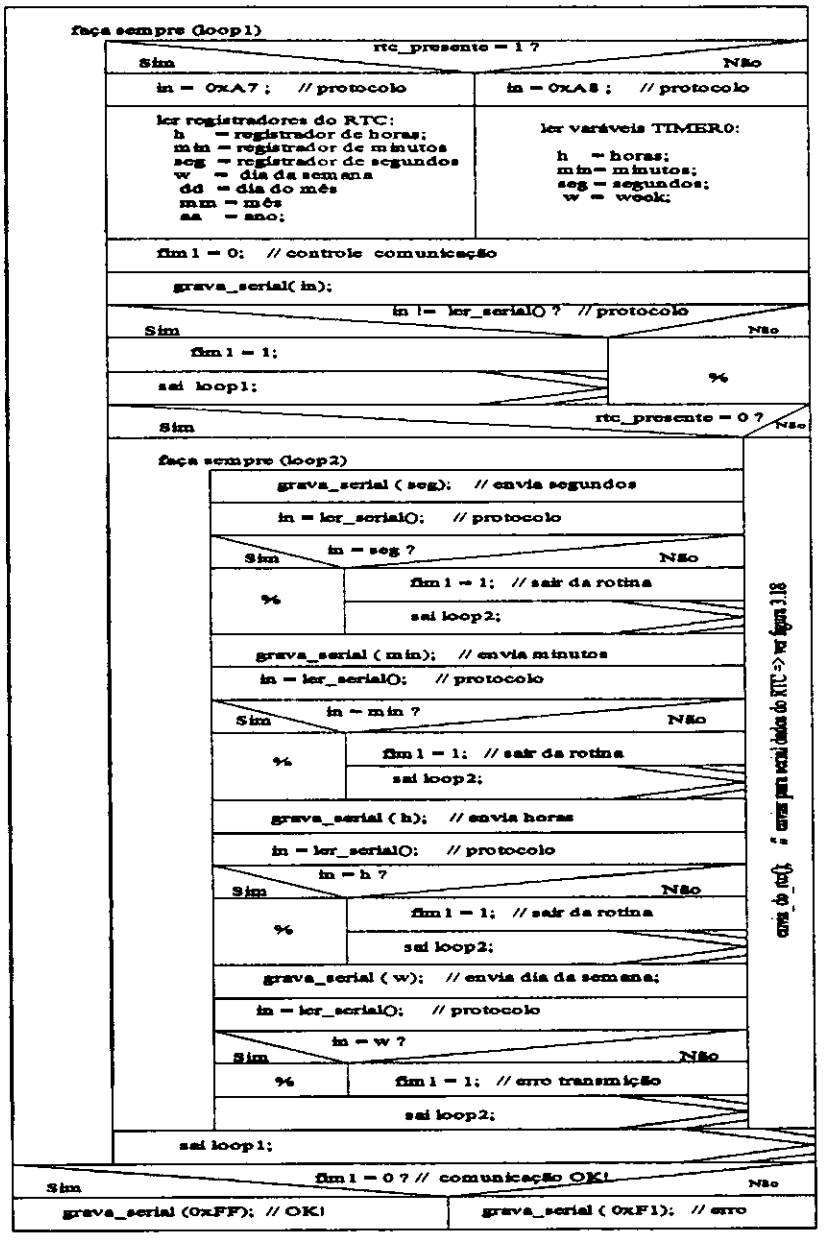


Figura 3.17 - Rotina leitura_rtc()

A Figura 3.18 mostra a parte da Figura 3.17, onde o RTC está presente na placa, referenciada como *envia_do_rtc()*. Neste caso, a rotina envia para o canal serial o horário e o calendário atual.

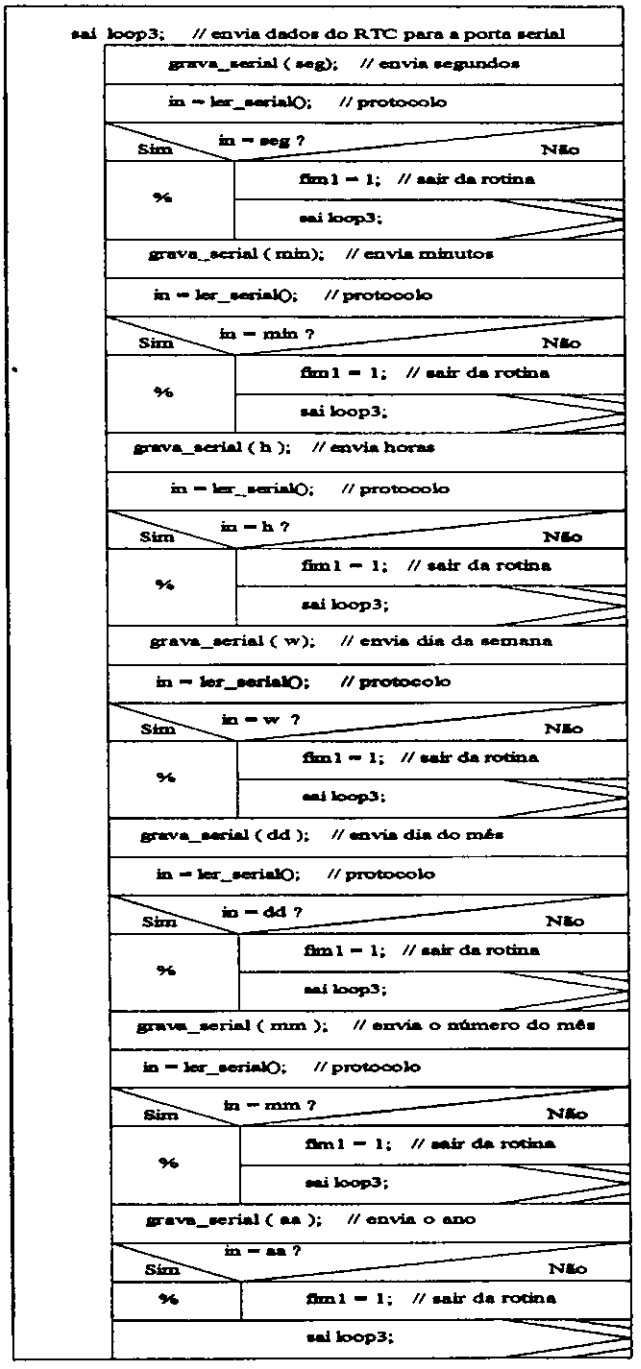


Figura 3.18 - Rotina envia_do_rtc()

A rotina *forçar_plano()* é usada para forçar a execução de um plano por um determinado período de tempo. Durante este período, não é verificado a tabela de troca de planos. Ao final do período é verificado a tabela de troca de planos e atualizado o plano ativo corrente, de acordo com o relógio. A Figura 3.19 mostra o diagrama desta rotina.

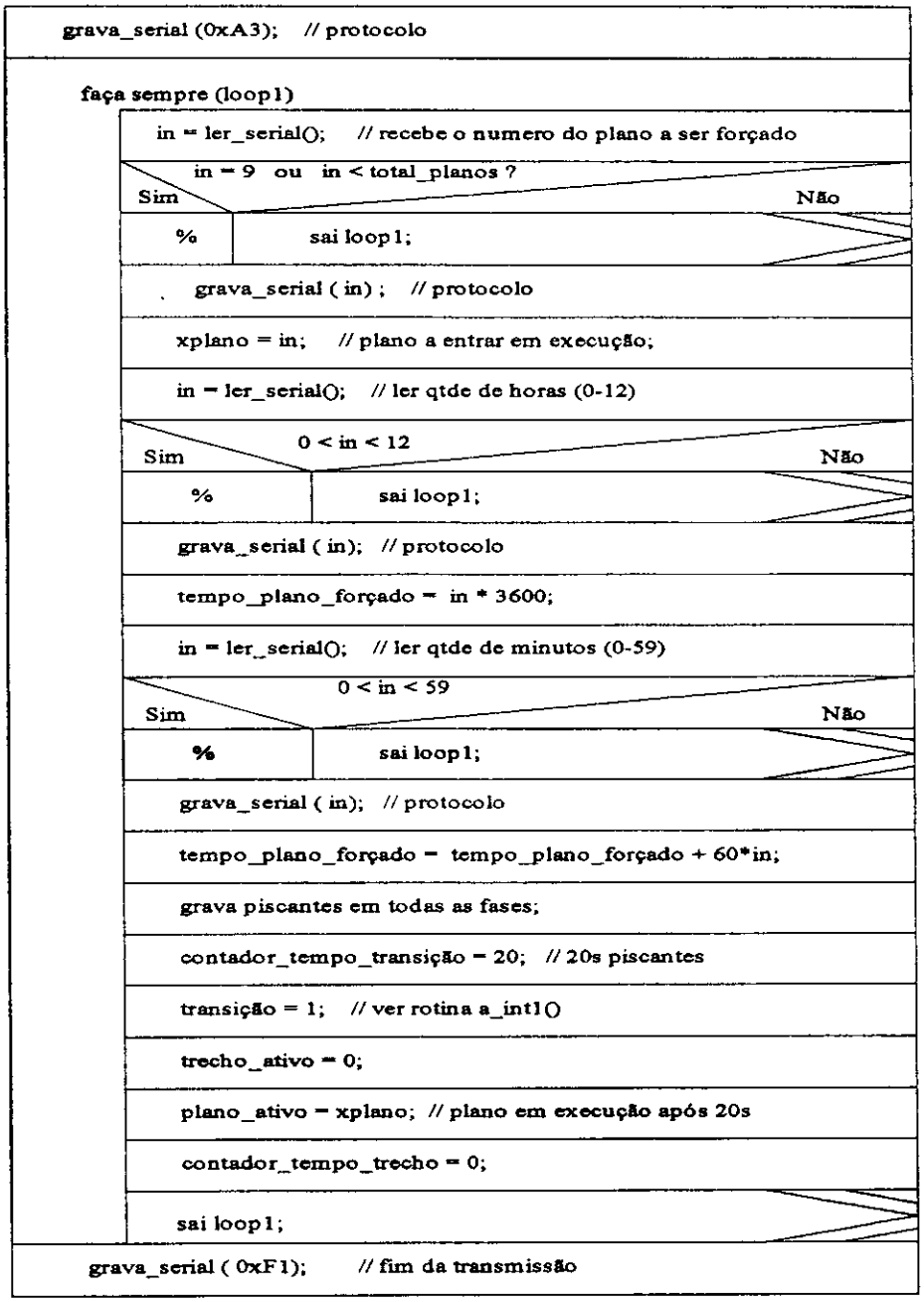


Figura 3.19 - Rotina forçar_plano()

A rotina *ler_erros()* é usada para verificar se existe ou se já ocorreu algum erro durante a atualização dos focos. Os erros detectados são: curto-circuito em algum triac de verde de alguma fase semafórica, erro na leitura da memória EEPROM e ocorrência de verdes conflitantes. Na próxima seção abordaremos sobre as rotinas responsáveis por esta verificação. Se ocorrer algum destes erros é gravada um código de erro na memória EEPROM em endereços específicos. A Figura 3.20 mostra o diagrama desta rotina.

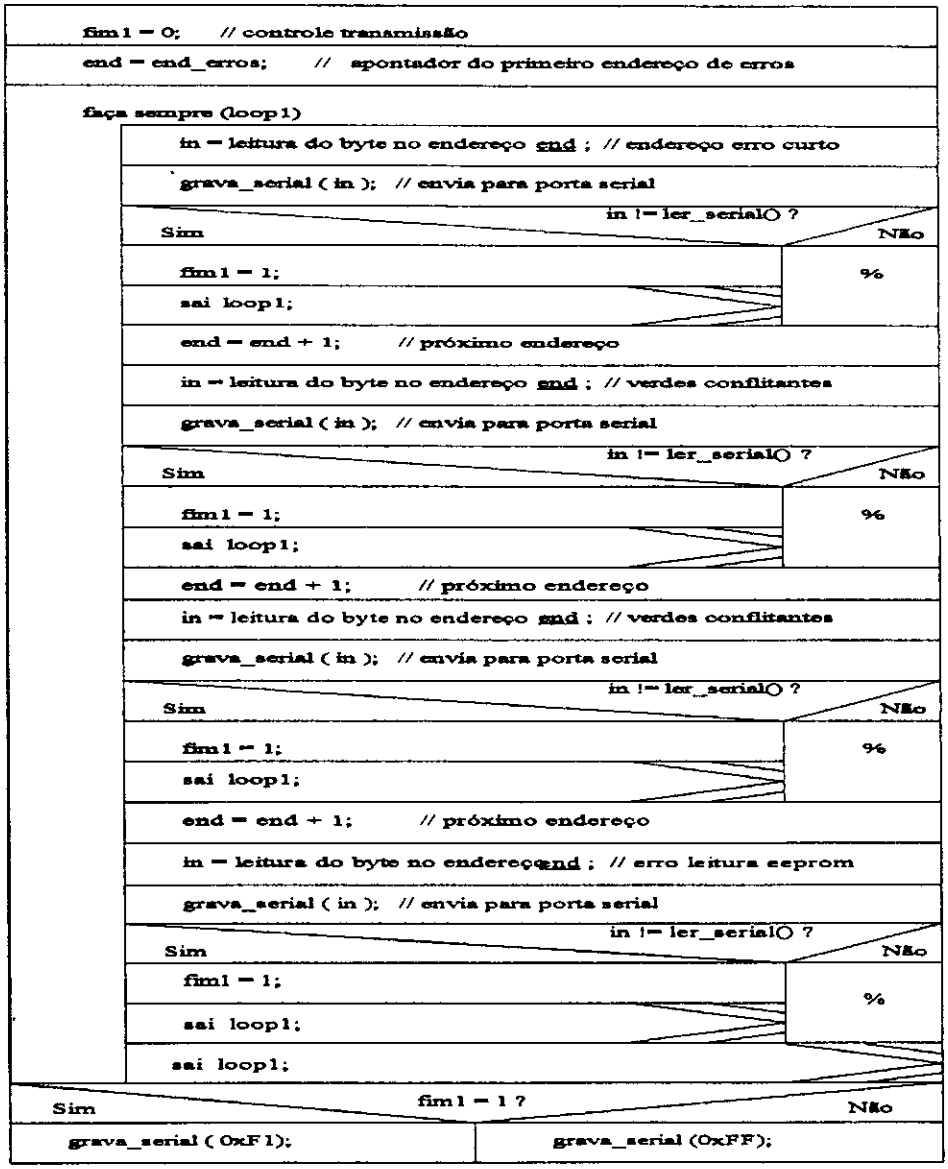


Figura 3.20 - Rotina ler_erros()

A rotina *limpa_eeprom()* foi concebida inicialmente para ser usada apenas durante o desenvolvimento do software no processo de debugação, pois quando gravamos uma nova programação, os dados anteriores são apagados durante o processo. Para apagarmos uma programação semafórica na eeprom basta apagarmos alguns endereços chaves. Neste caso, a interface envia o byte 0xF9 em hexadecimal, conforme mostrado na Figura 3.9, e a rotina *limpa_eeprom()* faz o restante. A figura 3.21 mostra o diagrama desta rotina.

```
grava_eeprom ( 0,  endereço de total de fases);  // cabeçalho
grava_eeprom( 0, end_erros);
grava_eeprom( 0,  end_erros+1);
grava_eeprom( 0,  end_erros+2);
grava_eeprom( 0,  end_erros+3);
grava_eeprom( 0, end_base);            // início planos tráfegos
grava_eeprom( 0, end_trocas);          // inicio tabela de trocas
grava_eeprom( 0, end_val_rtc);         // RTC não acertado
fim = 1 ;                              // reinicia no programa principal
grava_serial (0xF1);        // fim da transmissão
```

Figura 3.21 - Rotina limpa_eeprom()

3.3.3 - Rotinas de verificação de falhas

Para a verificação de falhas do controlador, utilizamos três tipos de testes durante a sua operação: teste de falha dos triac's de verde, verificação de ocorrência de verdes conflitantes e erro na leitura de cores na memória EEPROM. O teste de falha

nos triac's é realizado pela rotina *falha_hardware()*, após cada escrita nos focos. A Figura 3.22 mostra o diagrama desta rotina.

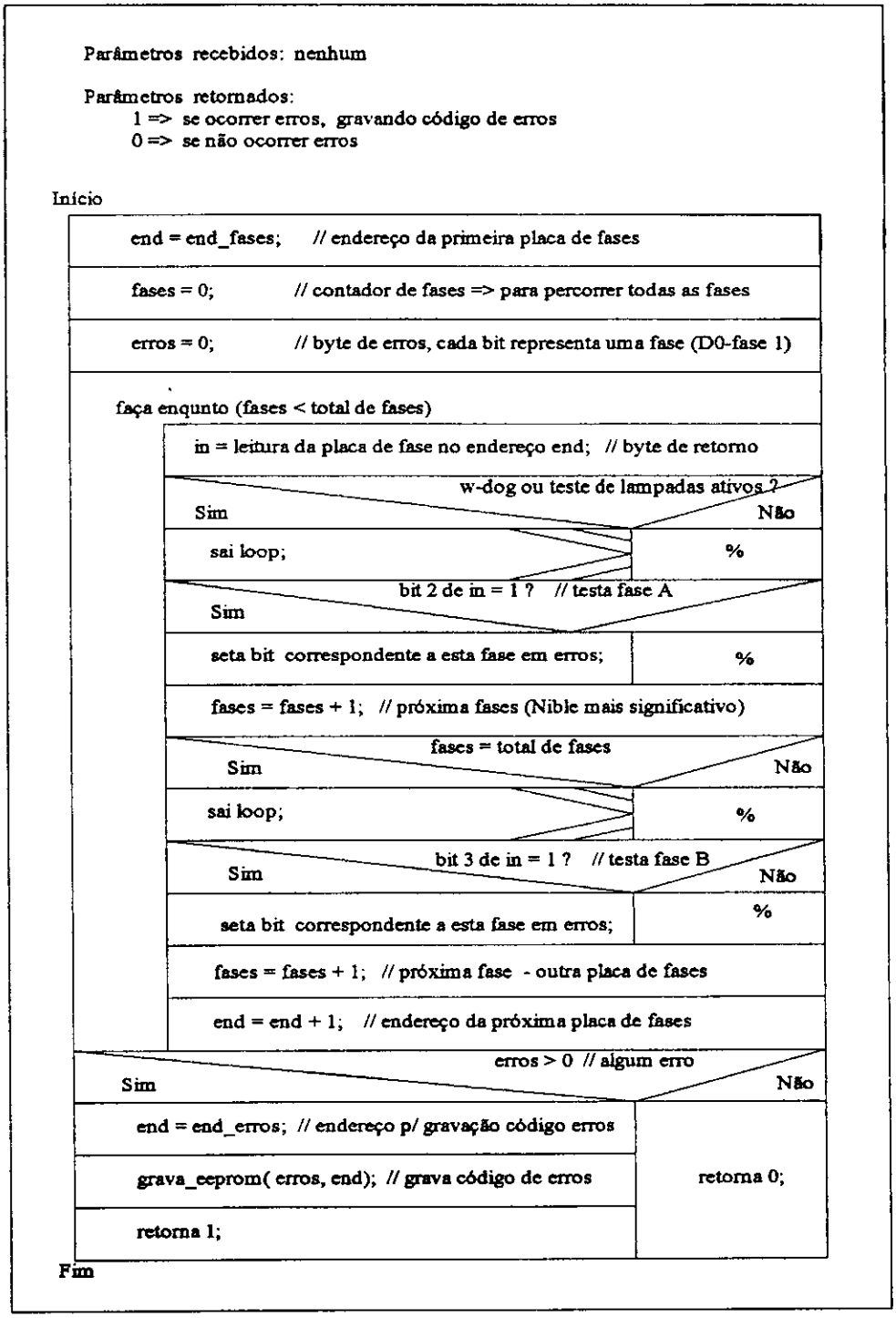


Figura 3.22 - Rotina falha_hardware

A verificação da ocorrência de verdes conflitantes é feita através da rotina *verdes_confl()*, após cada escrita de cores nos focos, comparando-se os focos de verdes acesos com a tabela de conflitos. A Figura 3.23 mostra o diagrama desta rotina.

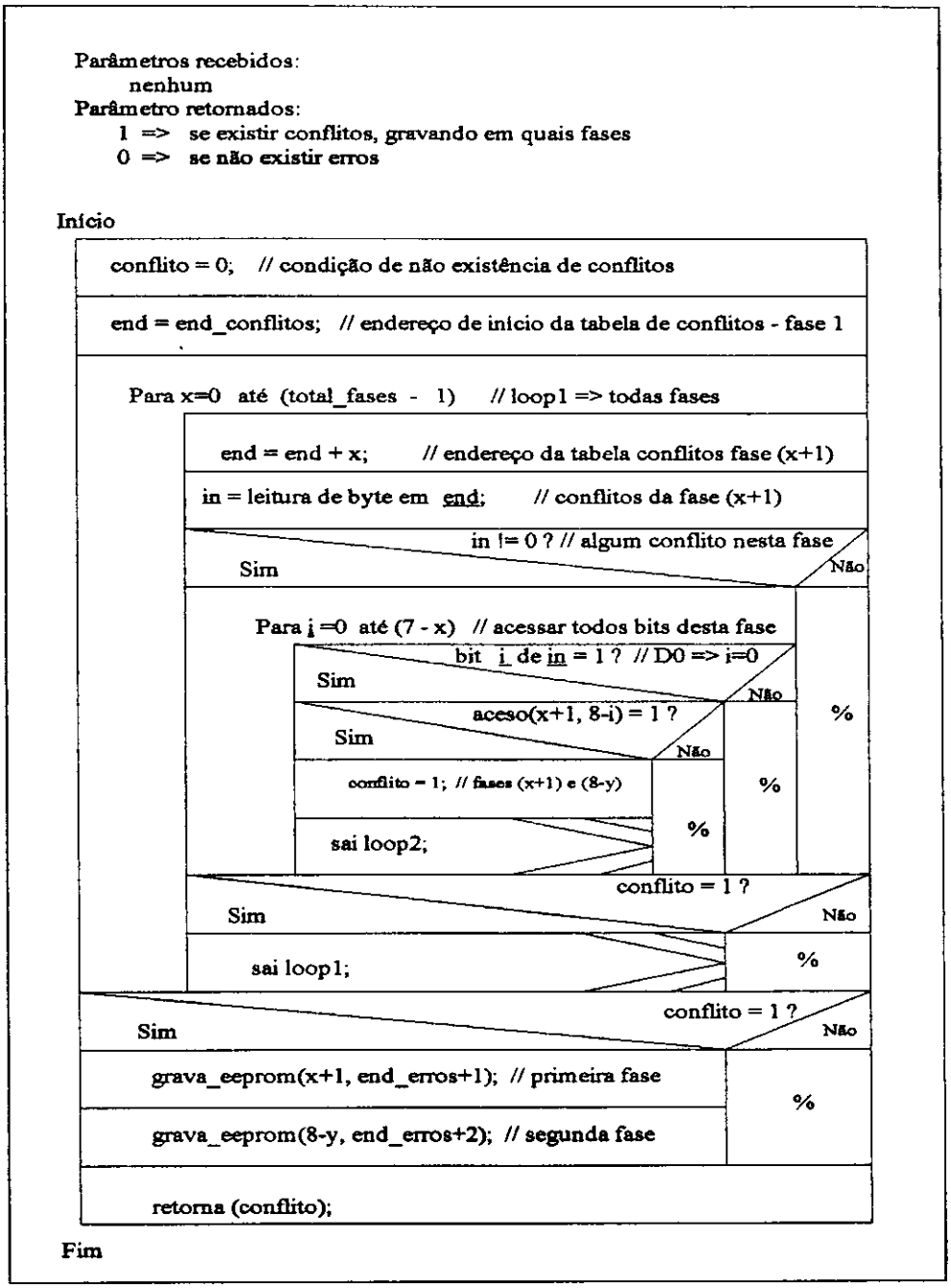


Figura 3.23 - Rotina verdes_confl()

A rotina *verdes_confl()* faz uso da rotina *aceso(fase1, fase2)* para verificar se as fases *fases1* e *fase2* estão acesas simultaneamente. O diagrama da rotina *aceso(fase1,fase2)* é mostrada na Figura 3.24.

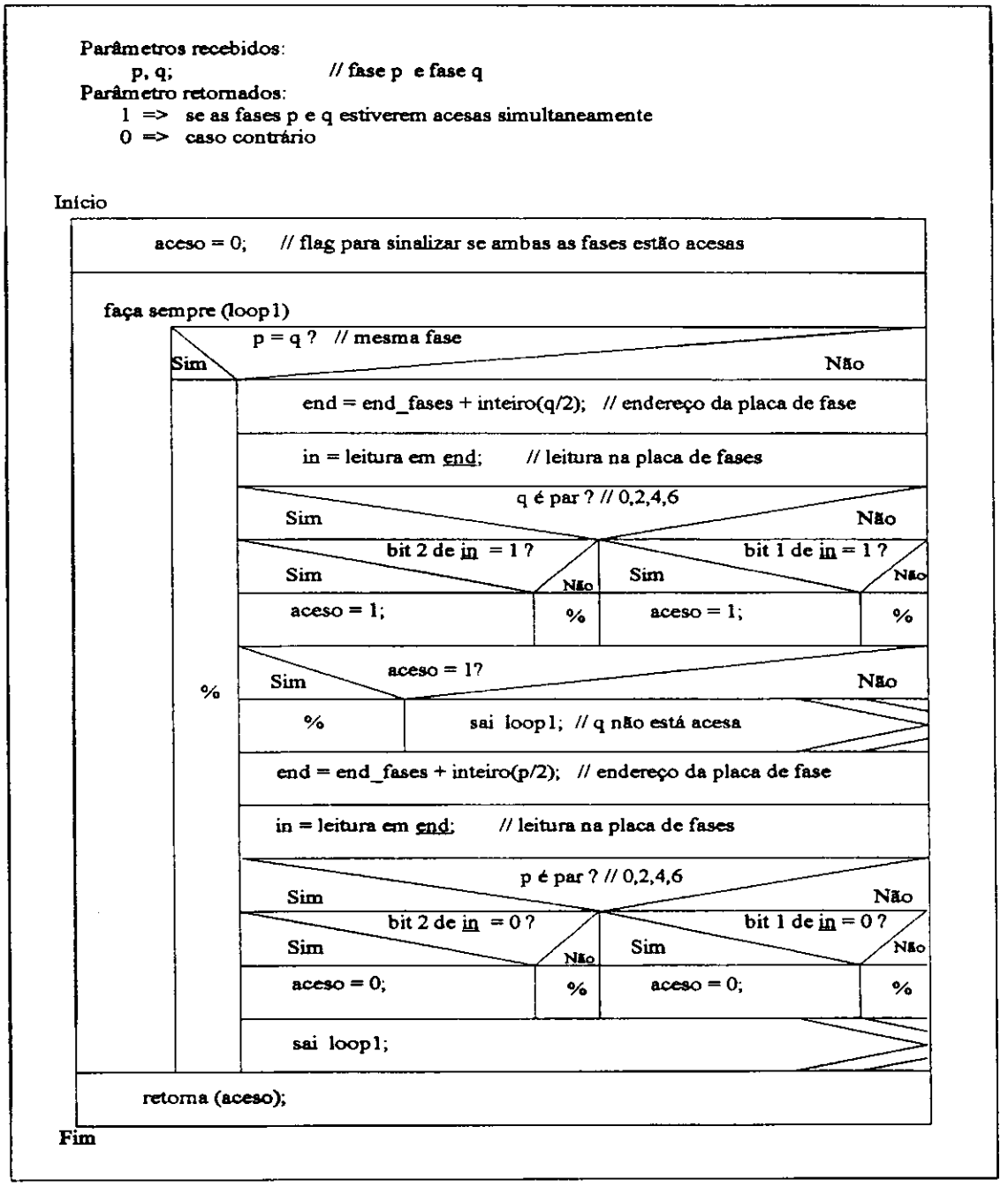


Figura 3.24 - Rotina aceso()

A verificação da ocorrência de erros na leitura das cores na eeprom é feita através da rotina *a_int1()*, toda vez que é escrito uma nova cor nos focos. Esta rotina faz uso da rotina *cor_existe()* para verificar se o byte lido corresponde a uma das cores válidas. A Figura 3.25 mostra o diagrama desta rotina. A combinação de 4 bits que formam a cor para as fases é definida via hardware, conforme Figura 2.2.

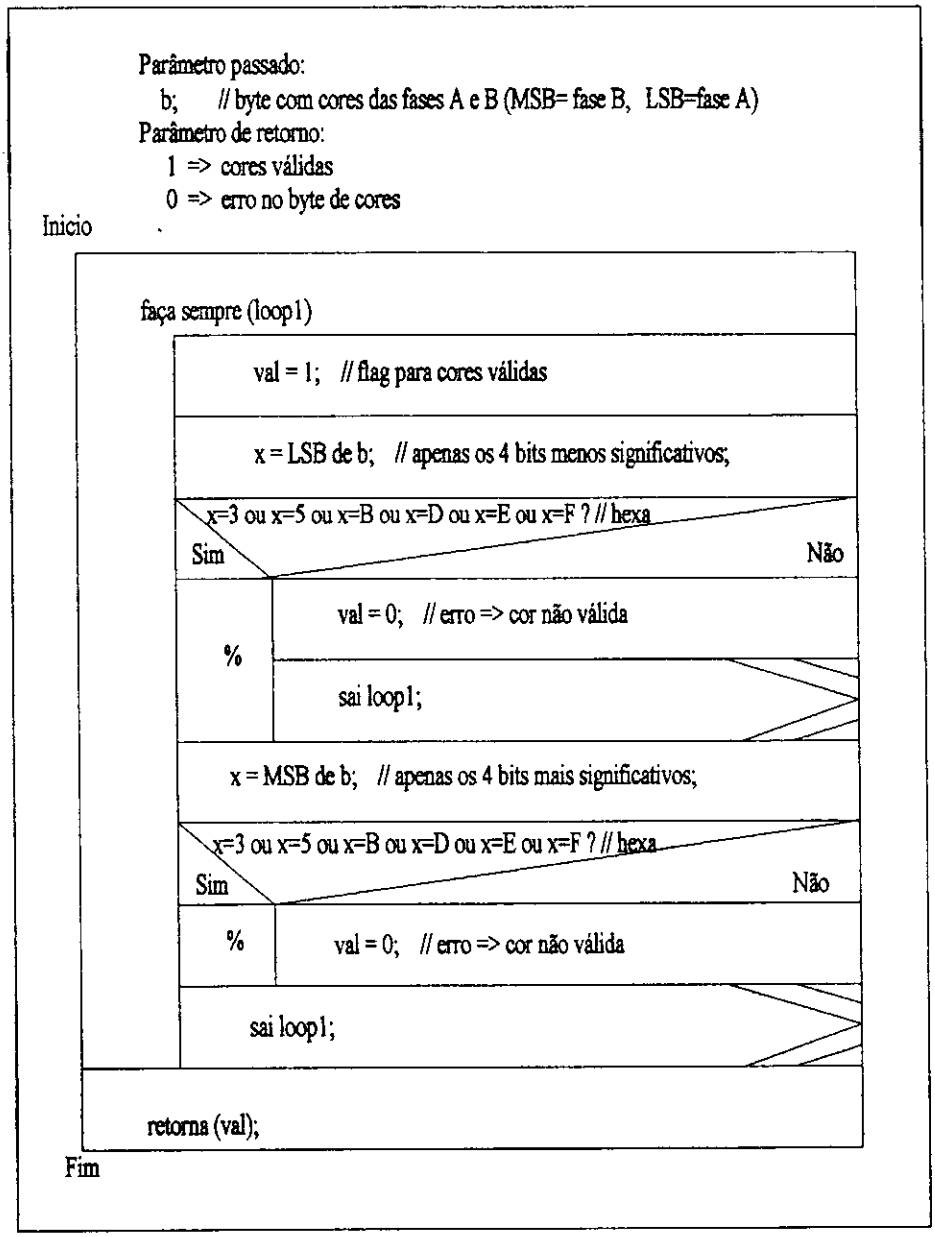


Figura 3.25 - Rotina cor_existe()

SOFTWARE DA INTERFACE

Apresentaremos neste capítulo o software de interface do equipamento. Descreveremos sua funcionalidade dentro do sistema, a metodologia empregada, sua arquitetura e estrutura de dados, os recursos mínimos exigidos para execução e, por fim, a interface com o usuário.

4.1 - Funcionalidade e Metodologia Empregada

A função deste software é fazer a aquisição dos dados relativos a uma programação completa de um sistema de controle de tráfego a tempo fixo. O software faz toda a consistência destes dados, armazenamento em disco, com opção de visualização gráfica e impressão. Outra opção deste software é fazer a comunicação serial com o equipamento de controle, com opções de envio e recebimento de dados, atendendo as opções descritas no capítulo anterior.

As características principais exigidas para este software foram: confiabilidade, facilidade de operação, facilidade de manutenção e reutilização do código. A confiabilidade é obtida após exaustivos testes por especialistas na área. A facilidade de operação é relacionada a interface homem-máquina, onde neste caso, adotamos as facilidades *janela-menu* em POP-UP com uso de cores e elementos estruturais de modo a tornar o uso do software confortável e fácil. Os ítens facilidade de manutenção e reutilização do código, são obtidos utilizando-se conceitos da Programação Orientada a Objeto - POO [Khoshafian, 1990], e por isto adotamos esta metodologia para o desenvolvimento deste software.

O cenário base do sistema consiste de objetos elementares como: janelas elementares com acessórios (sombreamento, títulos, movimentação e

redirecionamento), menus de opções com barra de seleção e teclas quentes e página gráfica. As classes de objetos com sua hierarquia são mostradas na Figura 4.1.

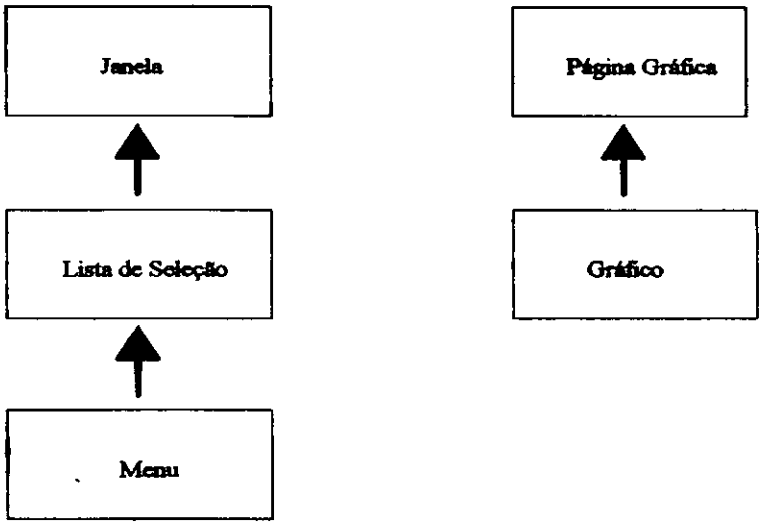


Figura 4.1 - Classes e hierarquia

Os métodos e atributos das classes criadas, foram concebidos para os objetivos deste trabalho.

4.2 - Arquitetura e Estrutura de Dados

A arquitetura do software é fundamentada em módulos que são acessíveis ao usuário pelo menu principal de opções. A Figura 4.2 mostra os módulos subordinados, que estarão disponíveis de acordo com a opção escolhida nos menus de opções.

O módulo *Programação* faz a aquisição de uma programação completa de um controle de tráfego a tempo fixo, que inclui: cadastro, alteração, consulta, armazenamento e impressão.

O módulo *relógio* faz a atualização e leitura do relógio do controlador de tráfego, através da porta de comunicação serial, padrão RS232.

O módulo *utilitário* gerencia, via comunicação serial, o restante das operações do controlador de tráfego, que inclui: transferência e leitura da programação do controlador de tráfego, leitura de erros de hardware quando existir, forçamento de planos e testes de comunicação.

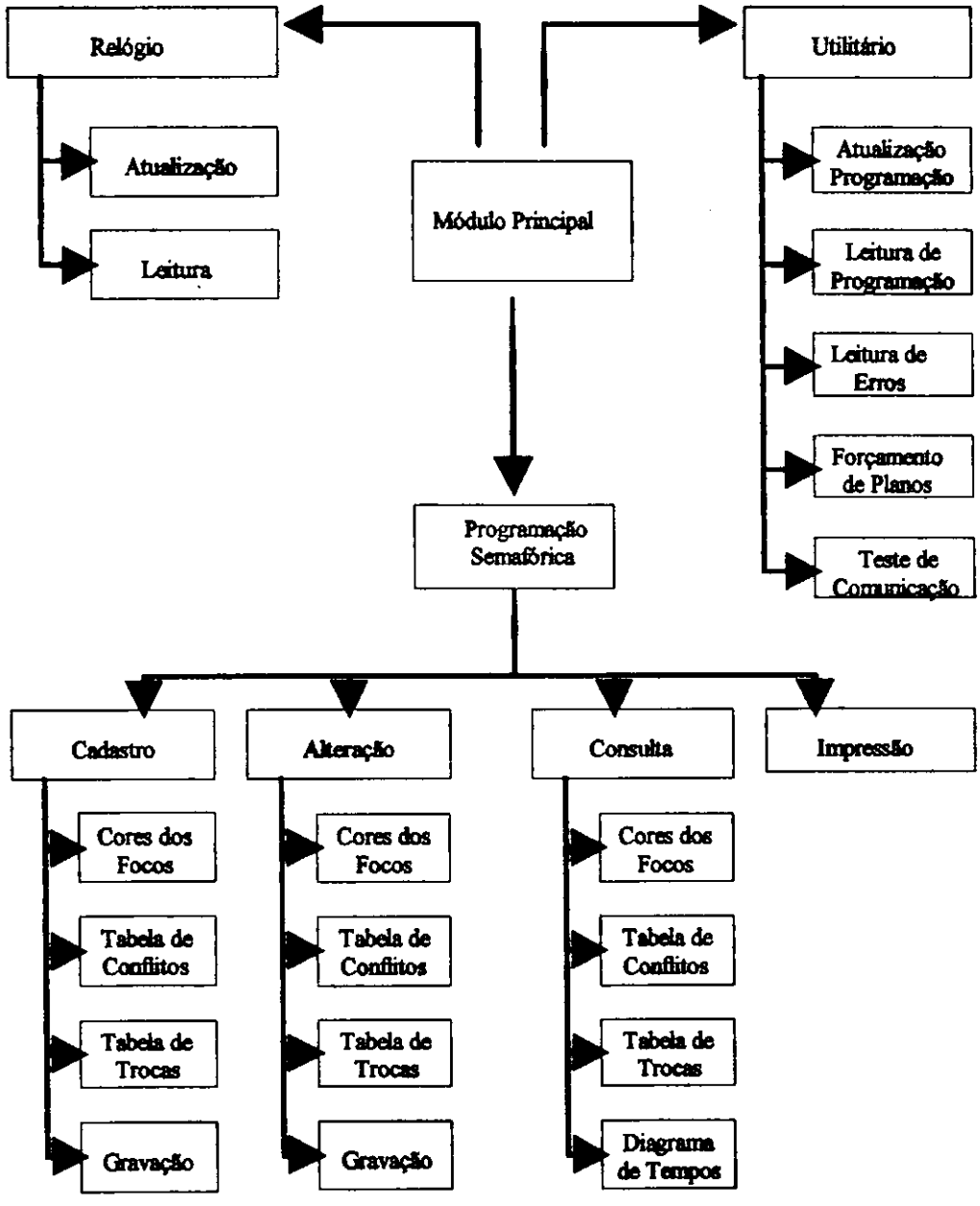


Figura 4.2 - Arquitetura dos módulos

Os protocolos para comunicação serial entre os diversos módulos com o controlador seguem os diversos diagramas detalhados no capítulo anterior, agora vistos pelo lado da interface.

A estrutura do arquivo gerenciado pelo módulo *Programação Semafórica* é composta pelos seguintes campos ou atributos:

```
struct horario{                 // horário com trocas previstas
  unsigned char w;              // 0DSTQQSS → 1=sim 0=nao
  int       horas;              // horas
  int       min;                // minutos
  int       seg;                // segundos
  int       nplano;             // Numero do plano a entrar em operação
};

struct plano{
  int       num_trechos;        // total de trechos programados
  int       num_fases;          // total de fases semafóricas
  int       num_planos;         // total de planos programados
  int       num_trocas;         // total de trocas previstas
  char      focos[8][8][40];    // cores por plano/fases/trecho
  int       tempos[8][40];      // tempos para cada trecho, por plano
  horario   h[50];              // tabela para troca de planos
  unsigned char conflitos[8];   // tabela de verdes conflitantes
  unsigned char plano9[8];      // cores dos focos para plano piscante
};
```

Neste caso, como teremos para cada objeto *plano* uma programação semafórica completa, esta corresponderá a um único arquivo para armazenamento. A opção de trabalharmos com apenas um registro por arquivo é possível pelo fato de termos vinculado aos nomes dos arquivos de dados sua própria identificação. O nome dos arquivos estão relacionados ao número do controle ou cruzamento e ao número da programação semafórica para aquele controle.

Nome de arquivo = **Pxxx_yyy. PLA**

onde:

xxx..... corresponde ao número do controle (000 .. 999)

yyy....: corresponde ao número da programação (000 .. 999)

Esta metodologia facilitará o manuseio do software pelos usuários, uma vez que nos grandes centros urbanos os controles são enumerados para facilitar sua operação e manutenção.

4.3 - Recursos Utilizados

Neste tópico vamos considerar os recursos utilizados durante o desenvolvimento do sistema, bem como, os recursos mínimos necessários para executar o software.

Entre as principais linguagens disponíveis para implementação dos conceitos orientados a objetos podemos enumerar as seguintes: SmalTalk, Objlog, Modula e C++. A linguagem C++ é considerada um remendo da linguagem C padrão com enxertos de alguns conceitos de orientação a objeto, pois não foi projetada especificamente para atender a esta nova metodologia. Entretanto, devido ao grande número de usuários C aliado a rapidez dos compiladores C++, e ao grande número de fornecedores, esta é sem dúvida a opção adotada pela maioria dos usuários.

Pelo exposto, adotamos a linguagem C++ como ferramenta de desenvolvimento do código. O produto empregado foi o compilador e ambiente de desenvolvimento integrado (IDE) da Borland Inc., Turbo C++, versão 2.0, para sistema operacional MS-DOS. Este ambiente inclui editor de texto, compilador, ligador, debugador e bibliotecas diversas.

Para execução do software de interface o usuário deve possuir equipamentos com no mínimo a seguinte configuração:

1 - CPU compatível com Intel 8088 / 8086, Ram mínima de 640K;

2 - Espaço em disco rígido livre de 1 Mbyte;

3 - Monitor de Vídeo com resolução VGA;

4 - Porta serial RS232;

5 - Impressora matricial de 80 colunas; e

6 - Sistema Operacional MS-DOS vs. 3.3 ou superior;

A plataforma utilizada para desenvolvimento e testes preliminares do software foi a seguinte:

1 - CPU 80486DX, 33 MHZ, 8 Mbyte de Ram, 2 (duas) Portas Seriais;

2 - Disco Rígido 240 Mbyte, 12 ms;

3 - monitor SVGA Color (1024 x 768 Pixels);

4 - impressora matricial 24 agulhas, 80 colunas;

Pelas características do projeto é recomendado a utilização de equipamentos *note-book* ou *Lap-top* para facilitar a operação e manutenção em campo dos controladores. Desta forma, estes requisitos mínimos serão totalmente atendidos, tendo em vista as características atuais destes equipamentos.

4.4 - Interface com o Usuário

O sistema utiliza o terminal de vídeo, o teclado e a impressora como modo de interação com o usuário, possibilitando ao mesmo ter acesso para consulta, alteração e inclusão de novos dados relativos a programação semafórica. Os dados requeridos para uma programação semafórica completa seguem alguns princípios, existindo uma seqüência de operações monitoradas pelo software.

A seguir descreveremos sobre as possíveis opções a partir do menu principal, mostrado na Figura 4.3.

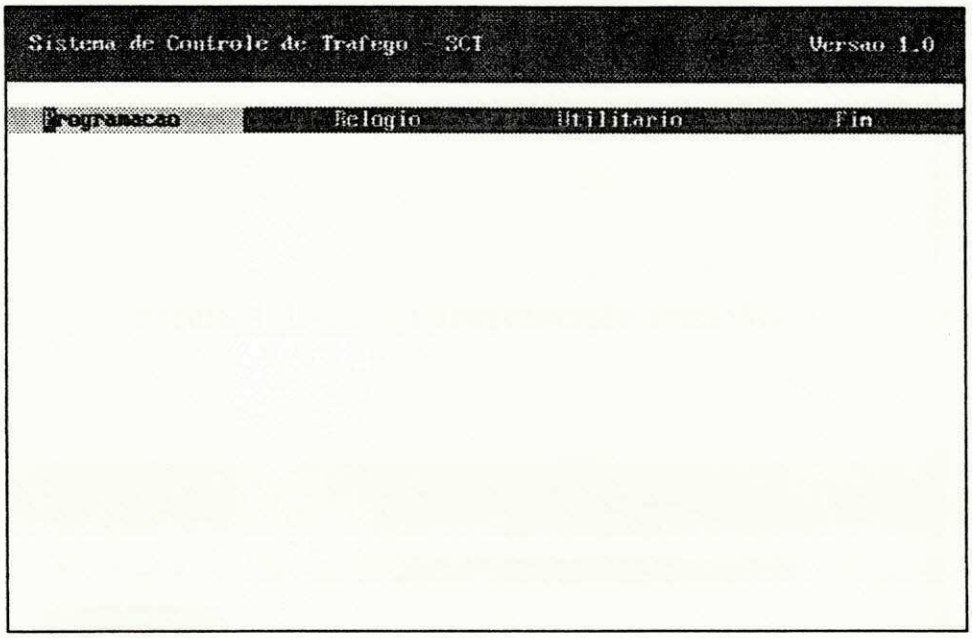


Figura 4.3 - Menu principal

A partir do menu principal, o usuário possui 04 (quatro) opções, disponíveis diretamente através das teclas quentes **P**, **R**, **U** e **F** ou através do movimento do cursor através das teclas → e ← (setas para direita e esquerda), seguido da tecla **[enter]**. Estas opções correspondem respectivamente a *Programação*, *Relógio*, *Utilitário* e *Fim*.

Na opção *Programação* o usuário poderá cadastrar, alterar, consultar e imprimir uma programação semafórica, sem ter acesso ao controlador através da porta serial. As Figuras 4.4, 4.5 e 4.6 mostram as novas opções decorrentes desta opção.

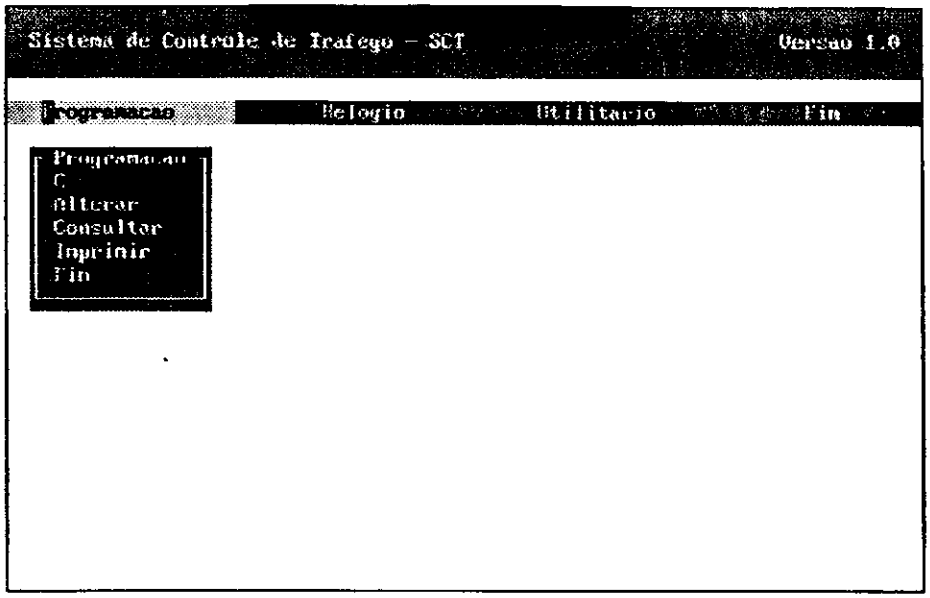


Figura 4.4 - Menu programação semafórica

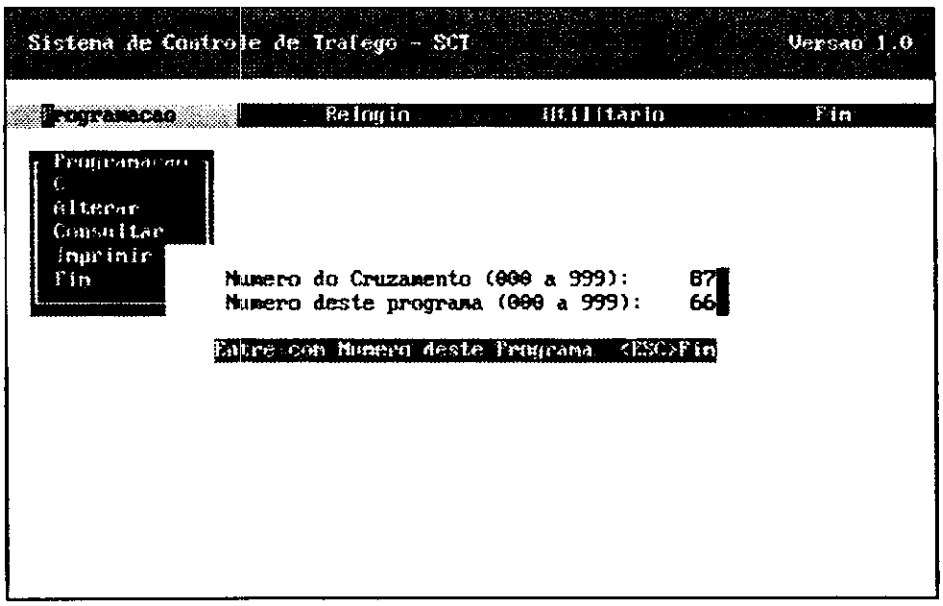


Figura 4.5 - Menu identificação do arquivo

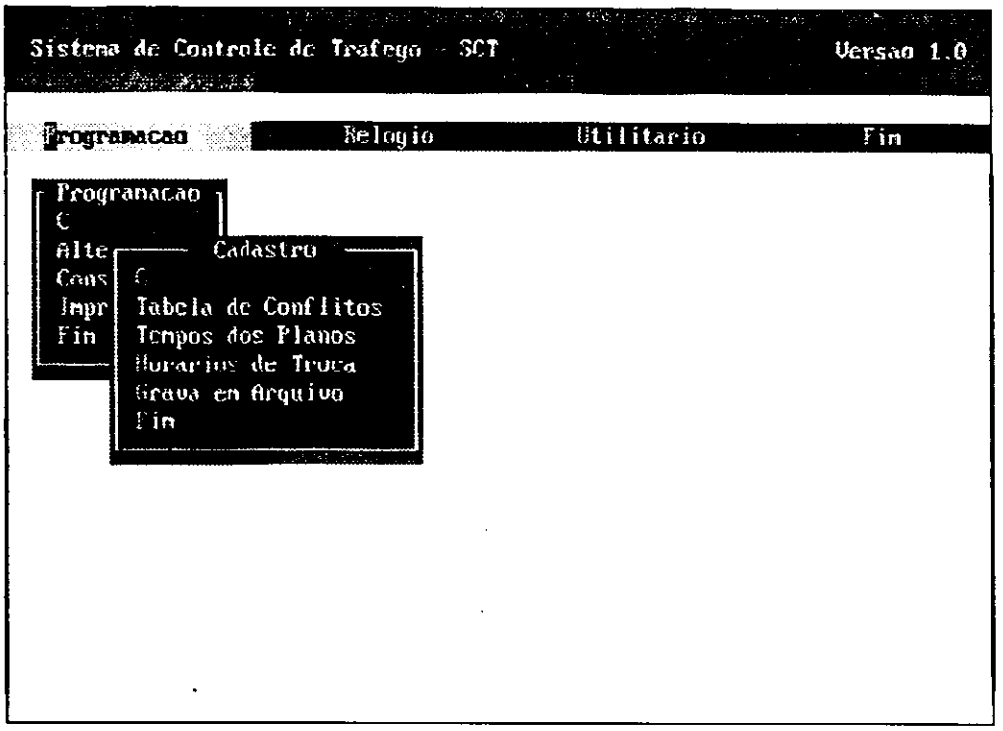


Figura 4.6 - Menu cadastro de planos

As Figuras 4.4, 4.5 e 4.6 mostram a trajetória percorrida durante o cadastro de uma programação semafórica. Inicialmente é solicitado a identificação desta programação, através do número do cruzamento e do número do programa. Se ainda não existir esta programação, o sistema através do menu plano novo, possibilitará ao usuário ao cadastro dos planos de tráfego através de suas tabelas de cores, tempos, conflitos de verdes, e da tabela de horários para troca de planos. Finalmente, pode ser feito o armazenamento desta programação, se os dados estiverem consistentes com as limitações do hardware do controlador.

Para a opção alteração e consulta o procedimento é semelhante, porém é necessário que já exista a programação semafórica (verificada pelo sistema procurando-se pelo nome do arquivo). No caso da opçao consulta, existe a alternativa de mostrar o diagrama de tempo de cada plano de tráfego programado, como mostrado nas Figuras 4.7 e 4.8, respectivamente.

Para a opção impressão o usuário basta fornecer a identificação da programação semafórica e colocar a impressora em linha. Apresentamos o exemplo de uma impressão no Anexo 3.

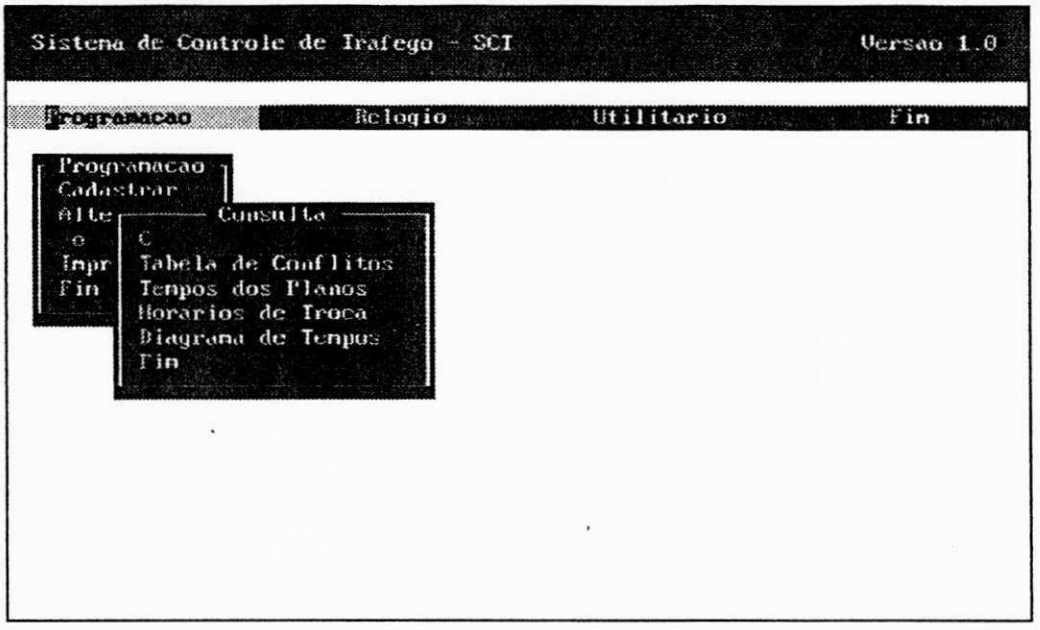


Figura 4.7 - Menu consulta

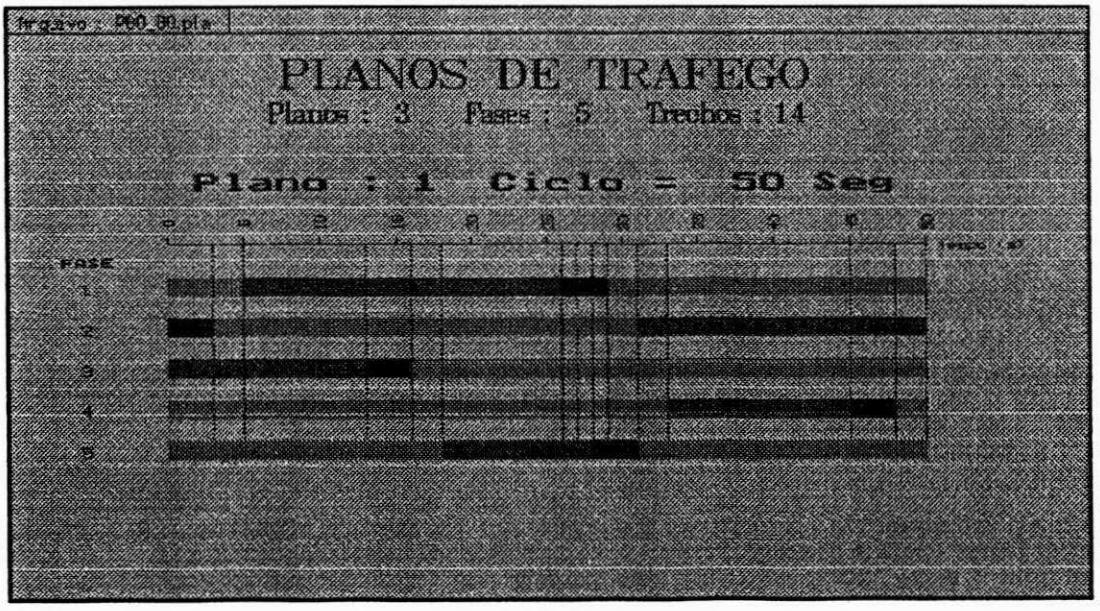


Figura 4.8 - Diagrama de tempos

A opção *utilitário*, a partir do menu principal, abre um novo menu de opções conforme mostrado na Figura 4.9, com 8 (oito) novas opções.

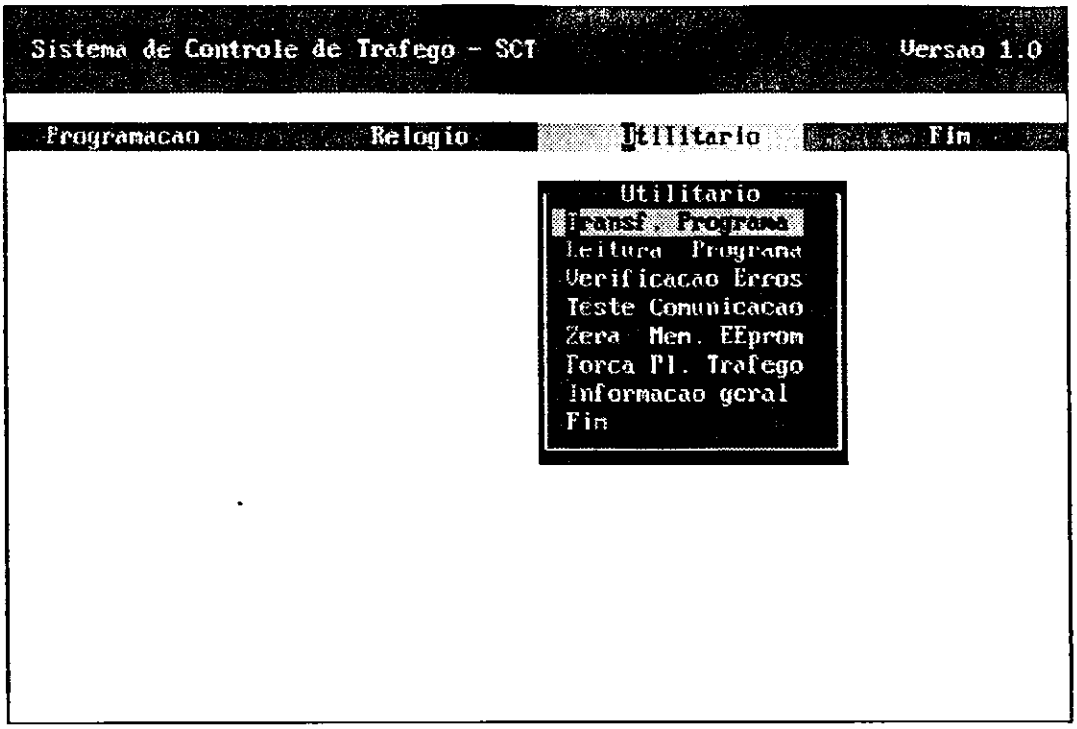


Figura 4.9 - Menu utilitário

A opção *Transf. Programa* corresponde a gravação de uma programação semafórica na memória *EEPROM* do controlador eletrônico, através da porta de comunicação serial. Ao escolher esta opção o sistema solicita a identificação do programa a ser transferido via serial e em seguida faz a transferência. Conforme visto no capítulo anterior, após ter sido concluído a transferência de programa é forçado um reinício no controlador para entrar em operação esta nova programação.

A opção *Leitura Programa* corresponde a uma leitura da memória *EEPROM* do controlador através da porta de comunicação serial, retornando ao usuário a identificação do controlador e os dados relativos a programação semafórica em operação.

A opção *Verificação Erros* corresponde a uma leitura na memória *EEPROM* do controlador em endereços correspondentes ao registro de ocorrência de eventuais erros, através da porta de comunicação serial.

A opção *Teste Comunicação* deve ser utilizada para verificar se a interface está comunicando corretamente com o controlador através da porta serial.

A opção *Zera Memória* corresponde ao envio de um código da interface para o controlador, através do canal serial, para limpar toda a memória *EEPROM* do controlador. Esta opção foi concebida originalmente durante a debugação do software, permanecendo até o momento.

A opção *Forçar Pl. Tráfego* corresponde a um comando da interface para o controlador, através da porta serial, forçando-o a operar com um determinado plano por um determinado período de tempo independentemente da tabela de trocas de planos. O número do plano e o período de tempo são fornecidos através do canal serial.

Na opção *Informação geral* é apresentado ao usuário dados relativos ao autor deste projeto e seus objetivos.

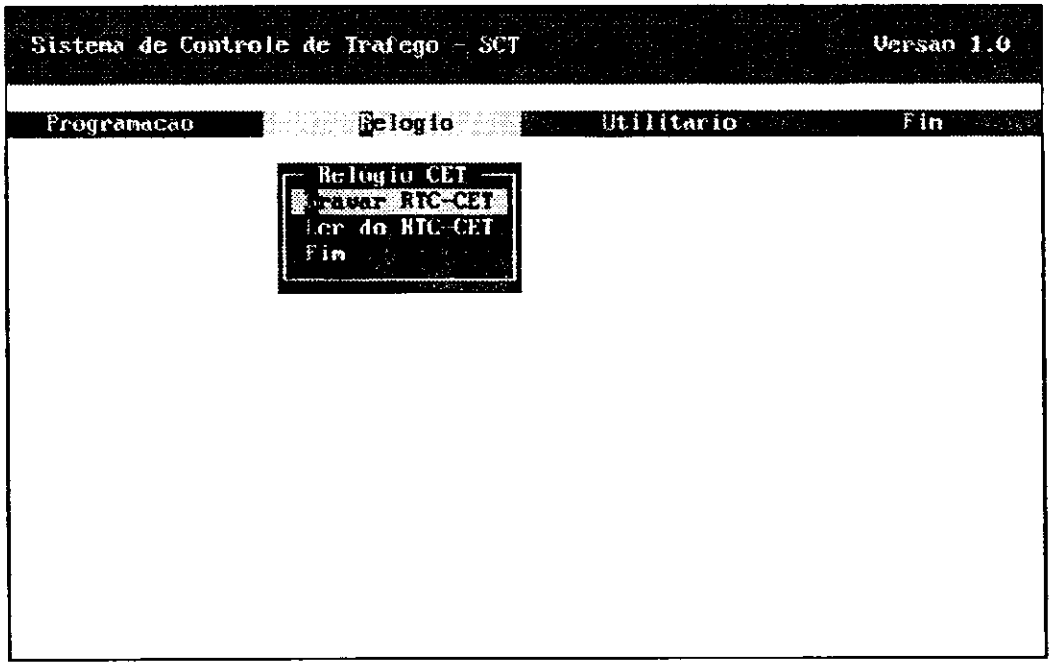


Figura 4.10 - Menu relógio

A opção *relógio*, a partir do menu principal, corresponde ao acesso dos dados do relógio do controlador através do canal serial, para atualização ou gravação e leitura.

A Figura 4.10 mostra o menu de opção relógio. Durante a gravação é enviado o horário do sistema operacional para atualização do controlador.

Todas estas opções que envolvem comunicação serial com o controlador semafórico atendem a um protocolo específico cujas rotinas correspondentes do outro lado do canal serial, foram detalhadas no capítulo anterior. Desta forma, os diagramas correspondentes a cada uma destas rotinas são imagens de suas rotinas correspondentes no controlador. Para efetivamente estabelecermos a comunicação serial com o controlador de tráfego, fizemos uso da *função bioscom()*, presente na biblioteca do compilador C++. A comunicação foi setada para saída COM1, 1200 bauds, utilizando 8 bits, sem paridade em cada transmissão.

IMPLEMENTAÇÃO E TESTES

Descreveremos neste capítulo os detalhes de implementação e os testes preliminares realizado com o protótipo em bancada. Todos os testes aconteceram em bancada com auxílio de equipamentos de medições para a parte de hardware e da interface para a parte de software. Foram desenvolvidos também alguns softwares de testes para rodar no microcontrolador para auxiliar na debugação.

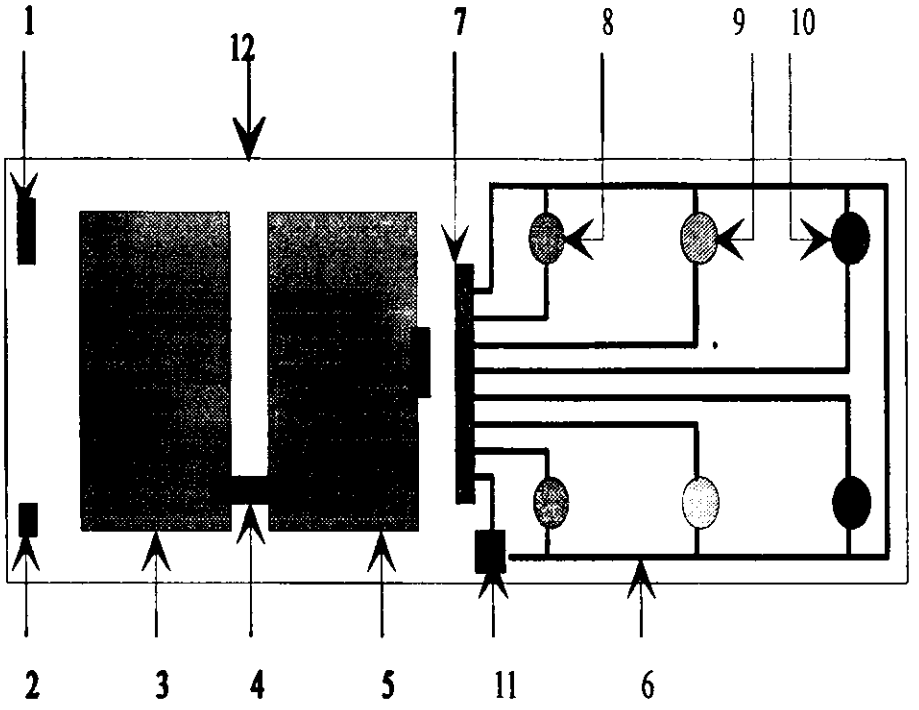
5.1 - Detalhes de Implementações do CET

A Figura 2.1 mostra o diagrama de bloco do controlador. Podemos dizer que o controlador é composto da fonte, da placa principal ou placa da CPU e de uma a quatro placa de fases. Da placa da CPU sai o cabo de 20 vias que é conectado a todas as placas de fases e o cabo de comunicação serial que é conectado a saída serial do microcomputador IBM-PC. Cada placa de fases possui um conector com 8 (oito) posições para os cabos de alimentação dos focos (fase e neutro, verde1, amarelo 1, vermelho1, verde2, amarelo2 e vermelho2), conforme visto nas Figura 2.2 e 2.3.

O protótipo implementado é composto da fonte, placa da CPU e apenas uma placa de fases. Foi improvisado também dois porta-focos rústicos com lâmpadas de cores e os conectores para as ligações à placa de fases e à alimentação da rede de energia elétrica (220V-60 Hz). Desta forma foi possível simular um cruzamento simples com duas fases semafóricas, como mostrado no Exemplo 1.1.

Os Anexo 1 e 2 mostram os digramas elétricos das placas da CPU e de fases. As duas placas e dois porta-focos rústicos foram afixadas em uma prancha de madeira, conforme mostrado na Figura 5.1.

Para o projeto definitivo sugerimos a utilização de um gabinete com porta e fechadura na parte frontal e com chassis interno conforme mostrado na Figura 5.2.



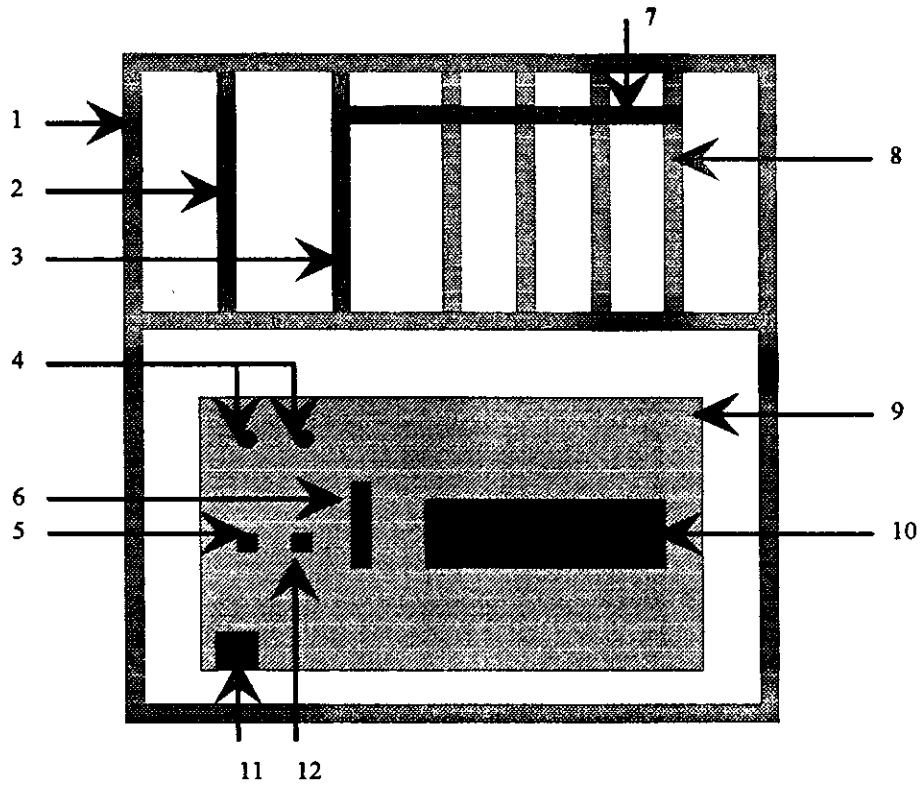
Legenda:

1 - conector para fonte (+5 V, GND, +12V, -12V)
2 - conector para comunicação serial
3 - Placa da CPU
4 - Cabo de 20 vias (conecta placa da CPU com placa de fases)
5 - Placa de fases
6 - Fio Neutro 220v
7 - barra de borne (conectores da placa de fases as lâmpadas)
8 - soquete e lâmpada verde 220v
9 - soquete e lâmpada amarela 220v
10-soquete e lâmpada vermelha 220v
11-conector para alimentação 220v-60Hz
12-prancha de madeira

Figura 5.1 - Detalhes de implementação

5.2 - Testes para Validação do CET

O sistema completo é composto das placas de hardware, o software de controle do controlador e o sotware da interface. Após efetuarmos a interligações entre as partes, passamos através da interface a efetuar diversas simulações, com várias programações semafóricas sendo enviadas ao controlador através da porta serial.



Legenda:

1 - Chassis
2 - Placa da Fonte
3 - Placa da CPU
4 - Led's
5 - Pino para teste de lâmpadas
6 - Disjuntor de entrada (10 A)
7 - Cabo com 20 vias p/ interligação das placas de fases
8 - Placa de Fases
9 - Painel de controle
10-Barra de Conectores de cabos de energia
11-Conector para saída serial
12-Botão de reset

Figura 5.2 - Proposta para projeto definitivo

Durante a etapa de depuração, foram desenvolvidos programas específicos para testes das partes, como por exemplo: teste de comunicação serial, teste de gravação da memória EEPROM, teste para programação de RTC, teste para simulação de erros devido a ocorrência de verdes conflitantes, etc. A medida que se comprovava o funcionamento correto das partes do programa, corrigindo eventuais falhas, tanto na parte de software como no hardware, eram integralizados ao sistema aquelas rotinas. Ao final, foram feitos exaustivos testes, executando-se o programa definitivo passo a passo e observando visualmente os resultados.

Nos testes em laboratório, utilizando-se a implementação mostrada na Figura 5.1, o sistema respondeu a todas as características propostas inicialmente. Entretanto, devemos considerar que estes testes foram feitos a temperatura ambiente, diferente da proposta definitiva (Figura 5.2), cujo gabinete deverá ficar ao tempo sujeito a sol, chuvas, portanto com variações continuas de umidade e temperatura.

CONSIDERAÇÕES FINAIS

Sinopse do Trabalho

Apresentou-se neste trabalho, as considerações e a configuração para um sistema de controle de tráfego a tempo fixo, com a finalidade de otimizar o funcionamento dos controladores semafóricos, utilizando-se o conceito de *trechos* na programação dos planos de tráfego.

No sistema proposto, adotou-se o uso do microcontrolador 8031, com a interface com usuário através de um microcomputador padrão IBM-PC, comunicando-se através da comunicação serial. Para isso, desenvolveu-se um programa escrito em linguagem de alto nível, para realizar a *programação semafórica*, armazená-la em memória de massa e posteriormente transmiti-la via serial, para o controlador.

Conclusões Finais

O conceito de *trecho* abordado neste trabalho pode ser estendido para equipamentos *atuados*, introduzindo-se alguns parâmetros adicionais em sua caracterização. Neste projeto, cada trecho é caracterizado pelo seu *tempo de duração* e pelas *cores de cada fase semafórica*. Para os equipamentos atuados ou equipamentos com sensores de veículos, podemos caracterizar cada trecho com os seguintes parâmetros:

1 - tipo de trecho: fixo, variável e optativo;

2 - tempos dos trechos: mínimo, máximo e extensões, quando varável;

3 - cores dos focos, por fases semafóricas;

O tipo de trecho quando *fixo*, terá apenas uma duração de tempo, no caso caracterizado em tempo mínimo. Quando tratar-se de tipo *variável*, teremos tempo mínimo e tempo máximo, além de extensão de tempo, contido no cabeçalho de dados, que será adicionado em função dos sensores de veículos. O tipo *optativo* é um tipo de trecho fixo que poderá ser pulado na execução do plano de tráfego, também em função de algum sensor de veículo ou para ajustar tempos de sincronismos.

O objetivo inicial deste trabalho era obtermos ao final, um sistema completo de controle de tráfego a tempo fixo, com vários equipamentos coordenados por um sistema central de controle. Ao final, foi implementado uma unidade completa, com apenas uma placa de fases, que, permitiu efetuar todos os testes requeridos. Esta unidade poderá funcionar *isolada* ou em *rede aberta*. Para efetuarmos a sua interligação com um controle central, será necessário rotinas adicionais de controle de comunicação, de acordo com o protocolo a ser adotado. Esta unidade mostrou-se segura, com interface amigável, com placa de fases modulares e principalmente com baixo custo.

A inexistência de publicações científicas nesta linha de pesquisa foi a principal barreira em que deparamos. Acreditamos estar iniciando uma nova linha de pesquisa junto a Coordenação de Pós-Graduação desta Universidade, que esperamos em breve, traduzir em conforto e economia para os usuários de trânsito.

Sugestões para Continuidade do Trabalho

A partir deste trabalho podemos enumerar diversas alternativas de pesquisas que poderão acontecer para melhorar e principalmente integrar o equipamento proposto em rede. Abaixo enumeramos algumas opções:

1 - Sistema para detecção de veículos integrado ao controlador;

2 - Sistema para digitalização de imagens acoplado ao controlador;

3 - Sistema de integração de controladores em rede;

4 - Programador portátil, etc.

REFERÊNCIAS BIBLIOGRÁFICAS

[AMD, 1991]

Advanced Micro Devices; "CMOS Memory Products", Data Book, Sunnyvale - Ca, USA, 1991.

[Arakaki, 1990]

Arakaki, R., et al, "Fundamentos de Programação em C - Técnicas Avançadas", 1990

[Berry, 1989]

Berry, J. T.; "The Waite Group's C++ Programming", Howard W. Sams & Company, Indiana, USA, 1989.

[CONTRAN, 1979]

Conselho Nacional de Trânsito - CONTRAN, "Serviços de Engenharia - Manual de Semáforos", Brasília - DF, Brasil, 1979.

[Esquivel, 1978]

Esquivel, Marco A. Vásquez; "Projeto e Construção de um Controlador de Sinais de Trânsito Adaptativo Isolados", Dissertação de Mestrado, DEE/PUC - RJ, Rio de Janeiro - RJ, Brasil, 1978.

[Khoshafian, 1990]

Khoshafian, S. et Razmik, A.; "Object Orientation - Concepts, Languages, Databases, User's Interface", John Wiley & Sons Inc., USA, 1990.

[Schinupp, 1978]

Schinupp. P. et al, "Software - Programmentwicklung und Projektooganisation", Walter de Gruyter, Berlin,1978

[Silva Jr, 1988]

Silva Jr, V. P. da; "Microcontroladores", Editora Érica, São Paulo, Brasil, 1988.

[Silva, 1990]

Silva Jr, V. P. da; "Microcontrolador 8051", Editora Érica, São Paulo, Brasil, 1990.

[Stroustrup, 1987]

Stroustrup, B.; "The C++ Programming Language", Addison-Wesley Publishing Company, New Jersey, USA, 1987.

[TEXAS, 1985]

TEXAS INSTRUMENTS; "The TTL Data Book - Standard TTL, Schottky, Low-Power Schottky Circuits", Data Book, v. 2, Texas Instruments, USA, 1985.

[XICOR, 1989]

Xicor Data Book Supplement; "Memory Products", Xicor Inc., Milpitas - Ca, USA, 1989.

[Weiskamp, 1991]

Weiskamp, K. et al; "Programação Orientada para Objeto com Turbo C++", Makron Books do Brasil Editora Ltda (tradução), São Paulo, Brasil, 1991.

[WEBSTER,1966]

WEBSTER, F. V. e COBRE, B. M. "Traffic Signals". Road research technical paper, nº 56, London, HMSO, 1966

Consultada:

1. "8051 Family of Single Chip Microcomputers", Microcomputer Components, Users Manual, Intel Corporation, Santa Clara - CA, USA, 1981.

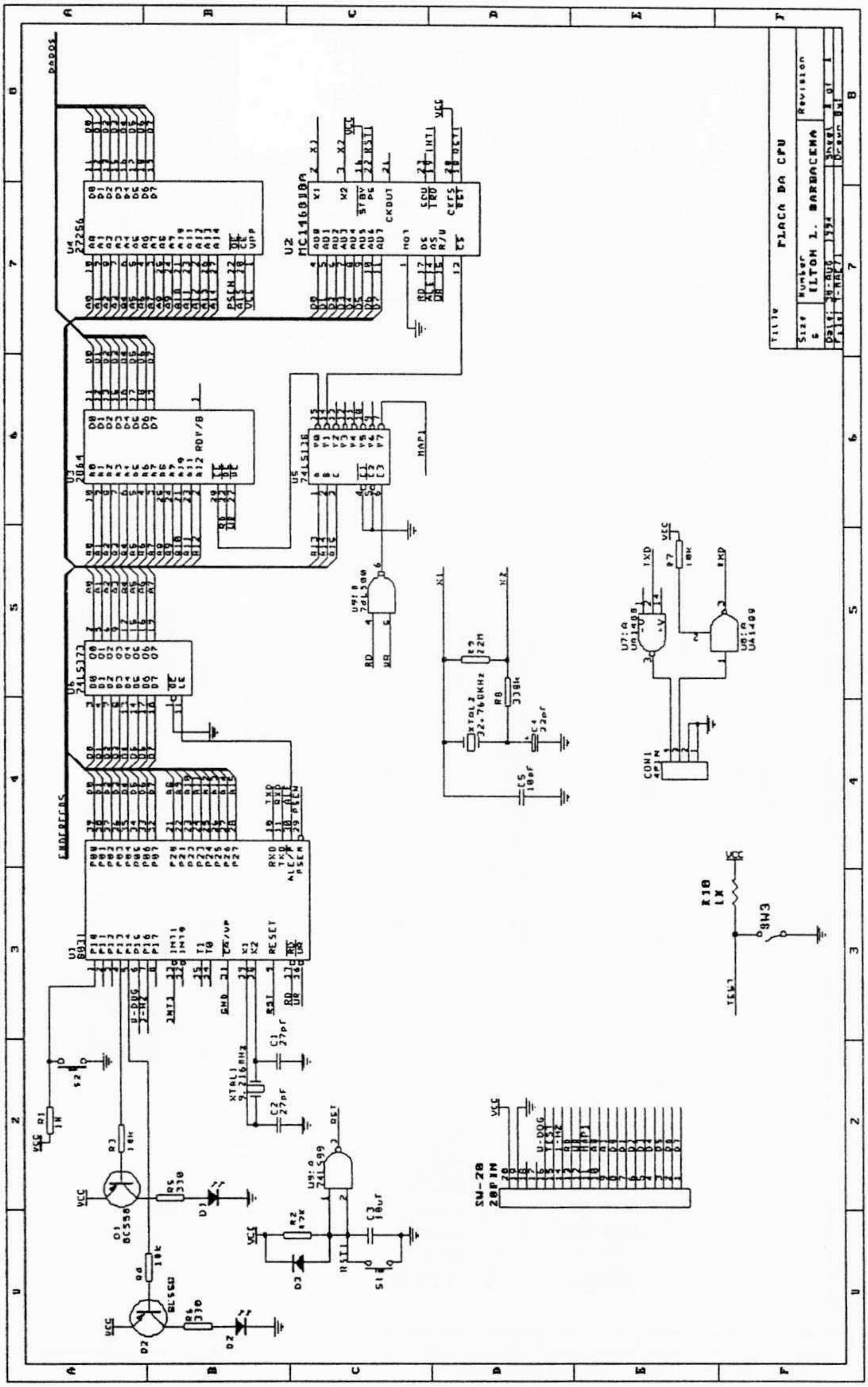
2. Ezzel, B.; "Programação Gráfica em Turbo C++", Ciência Moderna Ltda, Rio de Janeiro, Brasil, 1991.

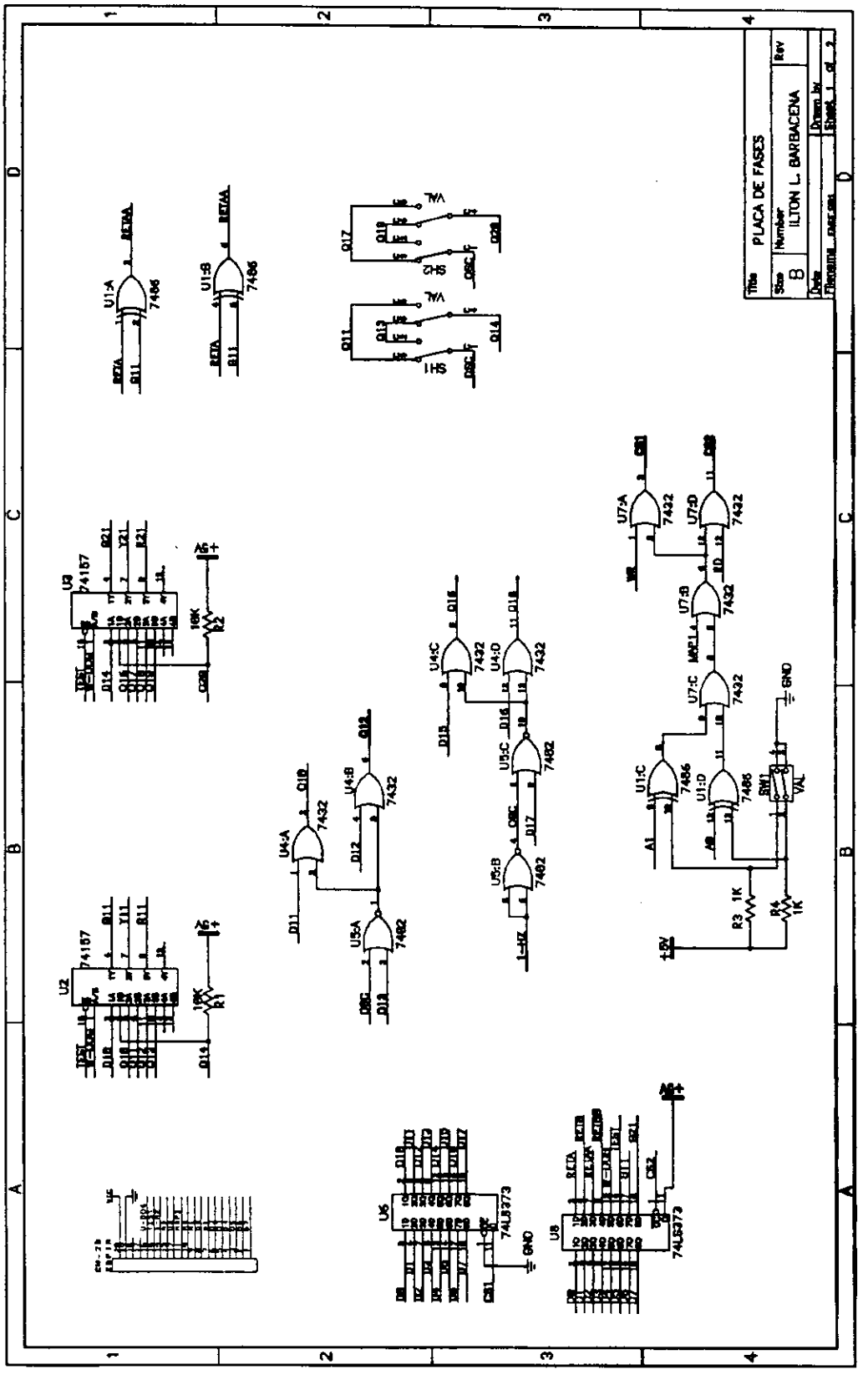
3. Ferraro, R. F.; "Guia do Programador Para as Placas EGA e VGA", Ciência Moderna Ltda, Rio de Janeiro, Brasil, 1990.

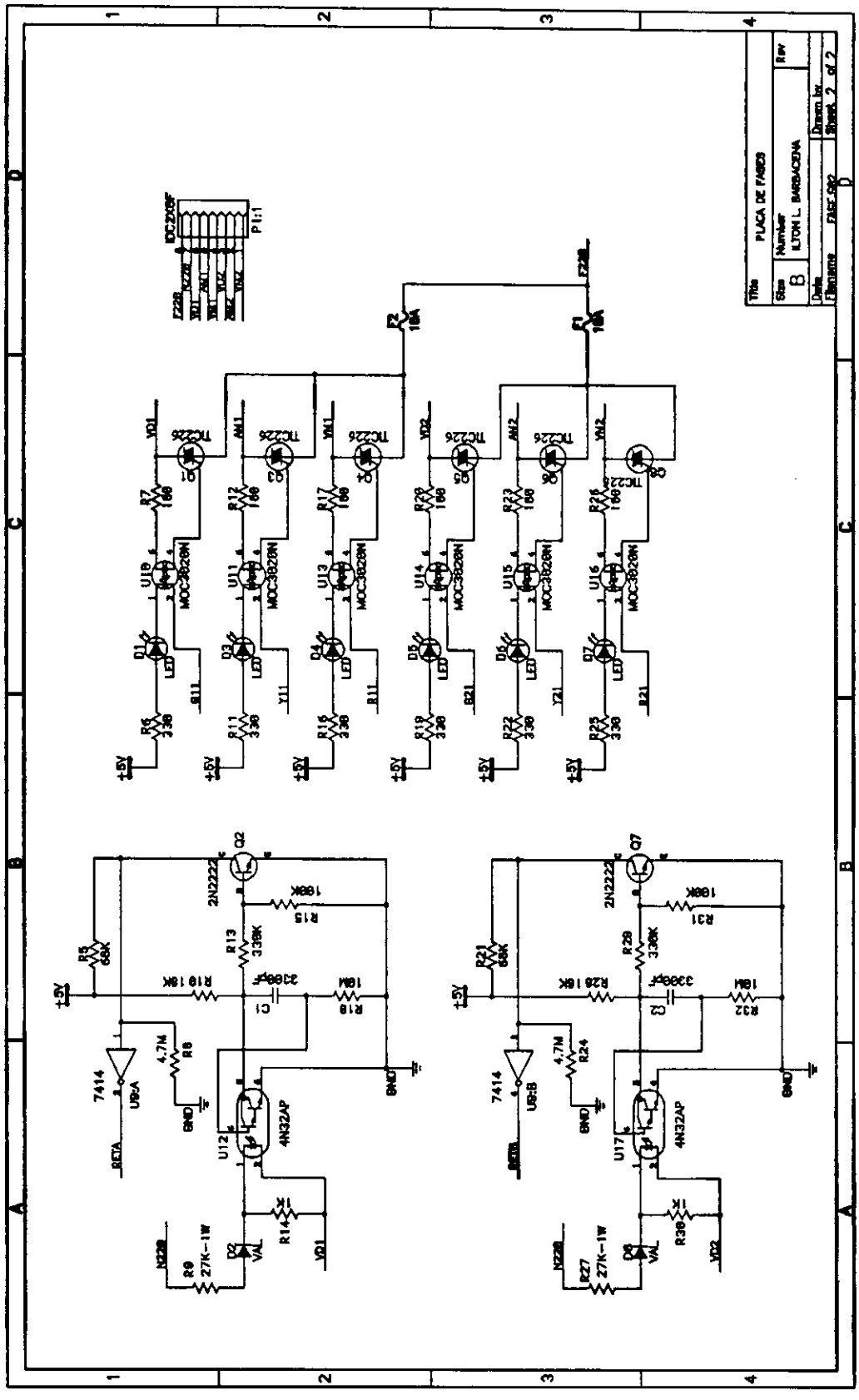
4. Lancaster, D.; "CMOS Cookbook", Howard W. Sams & Co. Inc., Indianapolis - ID, USA, 1977.

5. Norton, P.; "Guia do Programador para IBM-PC", Editora Campus Ltda, Rio de Janeiro, Brasil, 1989.

6. Pacitti, T.; "Programação e Métodos Computacionais", Livros Técnicos e Científicos, Rio de Janeiro, Brasil, 1977.




PLACA DE FASES
ILTON L. BARBACENA
U1:A
U1:B
U1:C
U1:D
U2
U3
U4:A
U4:B
U4:C
U4:D
U5:A
U5:B
U5:C
U5:D
U6
U7:A
U7:B
U7:C
U7:D
U8
R1 15K
R2 15K
R3 1K
R4 1K
RN1
VAL
SH1
SH2
GND
+5V


PLACA DE FASES
ILTON L. BARBACENA
P1:1
F2
10A
F1
10A
MOC3020M
2N2222
4N32AP
7414
+5V
BETA
BIND

```
================================================================================
UNIVERSIDADE FEDERAL DA PARAIBA - CCT                              Pag. :  1
COORDENACAO DE POS-GRADUACAO EM ENG. ELETRICA - COPELE
ORIENTADOR:  Prof. Misael Elias de Morais - Dr. Ing.
ALUNO      :  Ilton L. Barbacena

         RELATORIO DE PROGRAMACAO SEMAFORICA em 10/10/1994
================================================================================

                      Arquivo: P80_80.pla


          Total de Fases  :  5          Total de Trechos: 14
          Total de Planos :  3          Total de Trocas :  9
```

1 - Tabela de Conflitos das Fases:

	Fases: 1	2	3	4	5
Fase 1:	_	X	_	_	_
Fase 2:	_	_	_	_	_
Fase 3:	_	_	_	X	X
Fase 4:	_	_	_	_	X
Fase 5:	_	_	_	_	_

2 - Planos de Trafego:

Plano: 1 Ciclo: 50 Seg

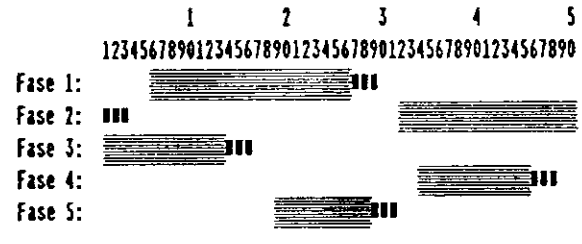
Cores dos Focos por Trecho:

Fase 1:	R	R	G	G	G	G	Y	Y	Y	R	R	R	R
Fase 2:	Y	R	R	R	R	R	R	R	R	G	G	G	G
Fase 3:	G	G	G	Y	R	R	R	R	R	R	R	R	R
Fase 4:	R	R	R	R	R	R	R	R	R	R	G	Y	R
Fase 5:	R	R	R	R	R	G	G	G	Y	Y	R	R	R

Tempos por Trecho:

Trecho[1] =	3 Seg	Trecho[2] =	2 Seg	Trecho[3] =	8 Seg
Trecho[4] =	3 Seg	Trecho[5] =	2 Seg	Trecho[6] =	8 Seg
Trecho[7] =	1 Seg	Trecho[8] =	1 Seg	Trecho[9] =	1 Seg
Trecho[10] =	2 Seg	Trecho[11] =	2 Seg	Trecho[12] =	12 Seg
Trecho[13] =	3 Seg	Trecho[14] =	2 Seg		

Diagrama de Tempo:




```
================================================================================
```

```
================================================================================
UNIVERSIDADE FEDERAL DA PARAIBA - CCT                          Pag. :  2
COORDENACAO DE POS-GRADUACAO EM ENG. ELETRICA - COPELE
ORIENTADOR:  Prof. Nisael Elias de Morais - Dr. Ing.
ALUNO      :  Ilton L. Barbacena

        RELATORIO DE PROGRAMACAO SEMAFORICA em 10/10/1994
================================================================================
```

Plano: 2 Ciclo: 65 Seg

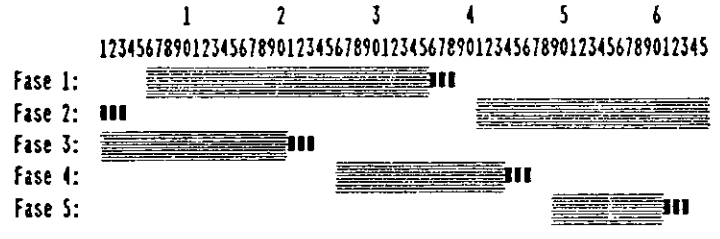
Cores dos Focos por Trecho:

```
        Fase 1:  R R G G G G Y R R R R R R R
        Fase 2:  Y R R R R R R R G G G G G G
        Fase 3:  G G G Y R R R R R R R R R R
        Fase 4:  R R R R R G G G G Y R R R R
        Fase 5:  R R R R R R R R R R G Y R
```

Tempos por Trecho:

```
Trecho[ 1] =  3 Seg    Trecho[ 2] =  2 Seg    Trecho[ 3] = 15 Seg
Trecho[ 4] =  3 Seg    Trecho[ 5] =  2 Seg    Trecho[ 6] = 10 Seg
Trecho[ 7] =  3 Seg    Trecho[ 8] =  2 Seg    Trecho[ 9] =  3 Seg
Trecho[10] =  3 Seg    Trecho[11] =  2 Seg    Trecho[12] = 12 Seg
Trecho[13] =  3 Seg    Trecho[14] =  2 Seg
```

Diagrama de Tempo:



Plano: 3 Ciclo: 75 Seg

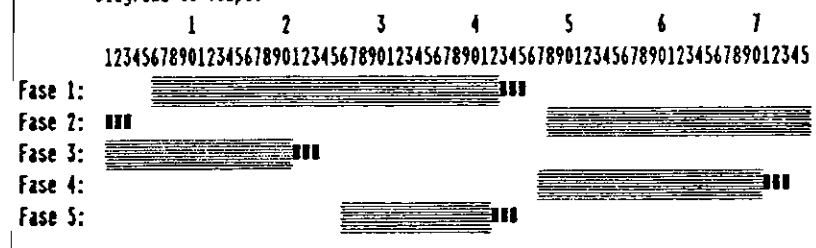
Cores dos Focos por Trecho:

```
        Fase 1:  R R G G G G G Y Y R R R R R
        Fase 2:  Y R R R R R R R R R G G G
        Fase 3:  G G G Y R R R R R R R R R
        Fase 4:  R R R R R R R R R G G Y R
        Fase 5:  R R R R R G Y Y R R R R R
```

Tempos por Trecho:

```
Trecho[ 1] =  3 Seg    Trecho[ 2] =  2 Seg    Trecho[ 3] = 15 Seg
Trecho[ 4] =  3 Seg    Trecho[ 5] =  2 Seg    Trecho[ 6] = 16 Seg
Trecho[ 7] =  1 Seg    Trecho[ 8] =  2 Seg    Trecho[ 9] =  1 Seg
Trecho[10] =  1 Seg    Trecho[11] =  1 Seg    Trecho[12] = 23 Seg
Trecho[13] =  3 Seg    Trecho[14] =  2 Seg
```

Diagrama de Tempo:




```
================================================================================
```

UNIVERSIDADE FEDERAL DA PARAIBA - CCT Pag. : 3
COORDENACAO DE POS-GRADUACAO EM ENG. ELETRICA - COPELE
ORIENTADOR: Prof. Nisael Elias de Morais - Dr. Ing.
ALUNO : Ilton L. Barbacena

RELATORIO DE PROGRAMACAO SEMAFORICA em 10/10/1994

3 - Tabela de Troca de Horarios:

Troca	Plano	Horario de Entrada	Dias da Semana DSTQQSS
1	1	21h 0m 0seg	XXXXXXX
2	1	8h 0m 0seg	X_____
3	2	6h30m 0seg	_XXXXXX
4	2	11h 0m 0seg	_XXXXXX
5	3	9h 0m 0seg	_XXXXXX
6	3	12h 0m 0seg	_XXXXXX
7	3	18h 0m 0seg	_XXXXX_
8	3	12h30m 0seg	_____X
9	9	23h59m59seg	XXXXXXX

Cores do Plano Especial (9) => y r A A y