UNIVERSIDADE ESTADUAL PAULISTA

Instituto de Geociências e Ciências Exatas

Campus de Rio Claro

IMPLEMENTAÇÃO DE DUAS ARQUITETURAS MICROCONTROLADAS TOLERANTES A FALHAS PARA CONTROLE DA TEMPERATURA

Gilson Fernando Botta

Orientadora: Profa. Dra. Deisy Piedade Munhoz Lopes

Dissertação de Mestrado elaborada junto ao Programa de Pós-Graduação em Física – Área de Concentração em Física Aplicada, para obtenção do Título de Mestre em Física.

Rio Claro (SP)

2002

004.22 Botta, Gilson Fernando
B751i Implementação de duas arquiteturas microcontroladas tolerantes a falhas para controle da temperatura / Gilson Fernando Botta. – Rio Claro : [s.n.], 2002

113 f. : il., gráfs., tabs., fots.

Dissertação (Mestrado) -Universidade Estadual Paulista,
Instituto de Geociências e Ciências exatas
Orientador: Deisy Piedade Munhoz Lopes

1. Arquitetura de computador. 2. Tolerância a falha (Computação). 3. Microcontroladores. 4. Arquitetura TMR. 5. Arquitetura em anel. 6. Sistema embarcado. I. Título.

Ficha catalográfica elaborada pela STATI – Biblioteca da UNESP
Campus de Rio Claro/SP

Comissão Examinadora

Profa. Dra. Deisy Piedade Munhoz Lopes
Instituição: IGCE/RC

Prof. Dr. Gerson Antonio Santarine
Instituição: IGCE/RC

Prof. Dr. Ronaldo Guimaraes Correa
Instituição: UFSCAR

Gilson Fernando Botta
Aluno

Rio Claro, 17 de Dezembro de 2002

Resultado: Aprovado – com “Distinção e Louvor”

Dedico este trabalho a minha esposa Karina, aos meus pais, Edilberto e Arlete, aos meus avós, Pedro e Irene, Cirvo (in memoriam) e Albina, e a todos os parentes e amigos que compreenderam minha ausência.

Agradecimentos

A Profa. Dra. Deisy Piedade Munhoz Lopes, pela orientação, amizade e incentivo, que foram fundamentais para realização deste trabalho.

Ao Prof. Dr. Gerson Antonio Santarine, pelo auxílio, apoio, amizade e pelas fotografias contidas no trabalho.

Ao Prof. Dr. Ronaldo Guimaraes Correa, pelas valiosas sugestões referentes ao trabalho.

A minha esposa Karina de Cássia Borges Botta, pelo incentivo e compreensão, durante todo o processo de desenvolvimento deste trabalho.

Aos Professores do Curso de Pós-Graduação em Física – Área de Concentração: Física Aplicada – da UNESP de Rio Claro, pelos ensinamentos transmitidos.

Ao Instituto de Geociências e ao Departamento de Física da UNESP de Rio Claro, pelo apoio e facilidades concedidos.

Ao José Roberto Locatelli Fonseca, Valter Rodrigues de Moraes, Danilo Carlos Pereira e Rodrigo Luiz Botta, pela cooperação, amizade e companheirismo.

Aos parentes e amigos que, direta ou indiretamente, me apoiaram no desenvolvimento deste trabalho.

“Também, assim como quereis que os homens façam a vós, fazei do mesmo modo a eles”.

Lucas 6:31

SUMÁRIO

ÍNDICE

Resumo

Os microcontroladores são empregados com freqüência crescente na automação e controle de processos. É fato, que em certos processos, a falha no sistema de controle é inadmissível. Nestes casos, é necessário o emprego de técnicas de tolerância à falhas. Diante disso, duas arquiteturas com base em microcontroladores, foram projetadas, construídas e submetidas a testes. Tanto a Arquitetura em Anel como a Arquitetura TMR implementadas, podem suportar falhas tanto nos nodos da estrutura, como nos arcos, que representam as ligações entre microcontroladores. Os nodos são controlados por protocolos implementados por programação, sem a necessidade de um circuito votante comum, presente na arquitetura TMR clássica, ou de qualquer outro circuito especial, para o controle da redundância dos circuitos. Os sistemas são modulares e podem operar sem um dos módulos ativado. Isso permite que um determinado módulo seja retirado para manutenção e posteriormente reinstalado, de maneira transparente a aplicação. Foram realizados testes nas arquiteturas desenvolvidas, com injeção de falhas físicas e lógicas. Ambas as arquiteturas responderam conforme o desejado, ou seja, detectaram e toleraram as falhas. As duas arquiteturas agregam características de confiabilidade e disponibilidade a sistemas de controle e apresentam-se como opções promissoras para a gestão de processos em tempo real.

Palavras Chave:

Tolerância à Falhas, Arquitetura em Anel, Arquitetura TMR, Microcontroladores, PIC, Arquitetura de Computadores, Arquiteturas Tolerantes a Falhas, Controle da Temperatura.

Abstract

The microcontrollers are frequently used in automation and process control. It's a fact that in certain processes the failure in the control system it is inadmissible. In these cases it is necessary to make use of the fault-tolerance techniques. Within this context two fault-tolerant architecture based in microcontrollers were project, built and submitted to extensive tests. The implemented Ring Architecture on the TMR Architecture can endure failure, either in the structure nodes (in the microcontrollers) or in the arches, which represent the connections between the microcontrollers. The nodes are controlled by protocols implemented by a program without the need of a common voting circuit, which is present in the classic TMR Architecture, on any other special circuit to control the redundancy of the circuits. Both system are modules and can operate without one been activated. These allow one module to be removed for maintenance and be reinstalled after words. In tests of physics fault and logic were made in the enveloped architecture and both reacted as expected such as detected and endure fault. Both architecture congregated characteristics of reliability and availability to the control systems.

Key Words:

Fault Tolerant, Ring Architecture, TMR Architecture, Microcontrollers, PIC, Computer Architecture, Fault Tolerant Architecture, Temperature Control.

LISTA DE FIGURAS

LISTA DE FOTOGRAFIAS

LISTA DE TABELAS

Tabela **pg**

Nomenclatura

Latinas

F	Freqüência	Hz
I	Corrente elétrica	A
P	Potência elétrica	W
R	Resistência elétrica	Ω
T	Temperatura	[ºC]
t	Tempo	s
U	Tensão elétrica	V

Abreviações

MCU	Microcontroller Unit – Unidade de Microcontrolador
MODEM	MOdulator/DEModulator – MOdulador/DEModulador
MTBF	Mean Time Between Failure – Tempo Médio Entre Falhas
MTTF	Mean Time To Failure – Tempo Médio Para Ocorrência de Falhas
MTTR	Mean Time To Repair – Tempo Médio Para Reparo
NMR	N Modular Redundancy – Redundância Modular Múltipla
PC	Program Counter – Contador de Programa
PIC	Programmable Integrated Circuit – Circuito Integrado Programável
PWM	Pulse Width Modulation – Modulação de Largura de Pulso
SCI	Serial Communication Interface – Interface de Comunicação Serial
TMR	Triple Modular Redundancy – Redundância Modular Tripla

Siglas

IGCE	Instituto de Geociências e Ciências Exatas
NASA	National Aeronautics and Space Administration
UNESP	Universidade Estadual Paulista
UFSCAR	Universidade Federal de São Carlos

CAPÍTULO 1

INTRODUÇÃO

A dependência humana relacionada aos sistemas computacionais tem sido cada vez maior. Nas telecomunicações, controle de tráfego aéreo e terrestre, armazenamento de dados sigilosos, previsão de tempo, usinas de geração de energia elétrica e em aplicações espaciais, militares, médicas e econômicas, computadores operam ativa e ininterruptamente. É, portanto, necessário que se possa confiar nos serviços prestados pelos mesmos.

Um alto grau de confiabilidade é imprescindível na gestão de tarefas críticas, como no controle de um reator nuclear (CLEMENTS, 1991). O emprego de técnicas de tolerância à falhas, permite agregar altos índices de confiabilidade e disponibilidade a sistemas de computação.

Confiabilidade é definida como a probabilidade de um sistema operar, sem incorrer em falha, até o tempo t, dado que no tempo t = 0, estava funcionando corretamente (HAYES, 1988). É a medida mais utilizada em sistemas em que mesmo curtos períodos de funcionamento incorreto são inaceitáveis, ou em sistemas em que o reparo nem sempre é possível, como em missões não tripuladas e sistemas de aviação (WEBER, 2002.a).

Como disponibilidade, define-se a probabilidade de um sistema estar trabalhando corretamente em um instante de tempo determinado (CLEMENTS, 1991). É a medida de alternância de períodos de operação e reparo de um sistema. Um sistema de computação pode possuir alto grau de disponibilidade, mesmo apresentando períodos de inoperância, desde que esses sejam breves e não comprometam a qualidade do serviço (LAPRIE, 1985).

Esses dois parâmetros são os mais usualmente empregados para mensurar a dependabilidade de um sistema. O termo dependabilidade, segundo Weber (2002.a), "é uma tradução literal do termo inglês **dependability**, que indica a qualidade de um serviço fornecido por um dado sistema e a confiança depositada no serviço fornecido". Outros parâmetros utilizados para medida de dependabilidade são: segurança de funcionamento (*safety*), segurança (*security*), mantenabilidade, testabilidade e comprometimento do desempenho (*performability*) (WEBER, 2002.a).

Existem basicamente duas técnicas utilizadas para implementação de sistemas altamente confiáveis:

1- Técnica de prevenção de falhas;
2- Técnica de tolerância à falhas.

A primeira procura eliminar as possíveis causas de falhas por construção, através de revisões de projeto e aplicação de métodos de controle de qualidade. A construção de circuitos 100% confiáveis, no entanto, é utopia. Já a segunda técnica, parte do princípio que falhas não podem ser evitadas. Um sistema computacional tolerante a falhas, portanto, pode ser definido como um sistema com alta dependabilidade, capaz de apresentar operação correta, mesmo na presença de falhas. Para se atingir tais metas, os projetos prevêem recursos redundantes, necessários para detectar e mascarar falhas, ou mesmo, isolar ou substituir um elemento com defeito, através de reconfiguração. A redundância – de hardware, software, informação e tempo – é o alicerce da técnica de tolerância à falhas (SIEWIOREK; SWARZ, 1992).

Modelagem de falhas, técnicas de redundância, injeção de falhas e técnicas de tolerância a defeitos para sistemas distribuídos, são áreas já bem desenvolvidas no estudo da tolerância à falhas. Atualmente, pesquisadores se dedicam à construção de sistemas embutidos com alta dependabilidade, através da aplicação de técnicas de tolerância à falhas (RENNELS; HWANG, 2001).

Sistemas embutidos, também conhecidos como sistemas embarcados, são sistemas computacionais dedicados a aplicações específicas, cujas principais características são (Silva Jr; et al., 2002):

- Alta integração de módulos de hardware e software;
- Voltados para aplicação específica;
- Interface com o mundo exterior bem definida;
- Restrições e requisitos fortes e bem definidos.

Exemplos de sistemas embutidos são terminais de caixa eletrônico, máquinas de refrigerante, controladores de temperatura, robôs móveis autônomos e aparelhos eletrônicos em geral. Estes sistemas podem ser implementados por meio da construção de hardware específico, baseado em portas lógicas, ou utilizando-se microcontroladores programáveis.

Atualmente, o segundo método é o mais comumente usado, por conferir vantagens como a otimização dos circuitos e custos e proporcionar maior flexibilidade aos sistemas, oferecendo uma melhor relação custo/benefício.

O presente trabalho descreve o projeto, construção e análise de duas arquiteturas microcontroladas tolerantes a falhas, para controle da temperatura de um determinado ambiente. Essas arquiteturas, próprias para utilização em sistemas embutidos, são bastante flexíveis e podem ser facilmente adaptadas para a gestão das mais diferentes tarefas. O trabalho aqui apresentado está estruturado como segue.

O capítulo 2 apresenta uma revisão bibliográfica com enfoque nos principais conceitos e definições relacionados ao estudo de técnicas de tolerância à falhas. A abordagem considera a caracterização de falha, erro e defeito e os vários tipos de

redundância utilizados para obtenção de alta confiabilidade e disponibilidade. As principais características de um microcontrolador são também consideradas neste capítulo.

No capítulo 3, encontra-se a descrição da parte experimental. A implementação do hardware e software das arquiteturas em anel e TMR é tratada, ressaltando-se as diferenças entre elas. Os circuitos e programas necessários para aplicação no controle da temperatura de um ambiente, são apresentados em detalhes.

Os resultados experimentais referentes ao desempenho dos sistemas construídos são apresentados no capítulo 4. Esses resultados são comparados com os obtidos em outras implementações.

As conclusões e sugestões para trabalhos futuros estão contidas no capítulo 5. Subseqüentemente, no final do trabalho, encontram-se as referências bibliográficas e os anexos.

CAPÍTULO 2

REVISÃO DA LITERATURA

2.1 SISTEMAS COMPUTACIONAIS TOLERANTES A FALHAS

2.1.1 Introdução

É fato que uma falha em uma máquina algorítmica (BROOKSHEAR, 2000) pode ocorrer a qualquer momento. Sendo assim, é desejável – às vezes imprescindível – que esta possa operar corretamente, mesmo na ocorrência de falhas (CHANDE et al., 1989) (KIMURA et al., 1980).

Com o intuito de se evitar os transtornos advindos de uma pane, são utilizadas basicamente duas técnicas (SIEWOREK; SWARZ, 1982):

1- Técnica de prevenção à falhas;
2- Técnica de tolerância à falhas.

Com a técnica de prevenção à falhas, que é onerosa, tenta-se prevenir a ocorrência destas, através de revisões de projeto, realização intensa de testes e aplicação de diversos métodos de controle de qualidade. Busca-se eliminar por construção, as possíveis causas de falhas. Todavia, pragmaticamente, não se consegue circuitos 100% confiáveis. Embora seja possível prever a confiabilidade de um lote de componentes, não há como assegurar a confiabilidade de um componente individual (FISCHER, 1990). Projetos que empregam esta técnica, portanto, procuram reduzir a taxa de ocorrência de falhas, a patamares ditos aceitáveis (FRAGA, 1987).

Na técnica de tolerância à falhas, parte-se do princípio que falhas não podem ser evitadas. É escopo desta técnica, reagir à ação da falha na máquina física, permitindo à máquina lógica continuar operando corretamente. Com ela, pode-se construir sistemas com altos índices de confiabilidade e disponibilidade (SIEWOREK; SWARZ, 1982). Em relação à técnica de prevenção à falhas, apresenta a vantagem de permitir a construção de sistemas robustos, com o uso de componentes de menor custo (FISCHER, 1990).

A seguir, são definidos os conceitos de erro, falha e defeito. Posteriormente, são analisadas as expressões de confiabilidade e disponibilidade.

2.1.1 Conceitos de erro, falha e defeito

Autores como Laprie (1985) e Anderson e Lee (1981), ocuparam-se da nomenclatura e dos conceitos básicos da área. Com base em seus trabalhos e concernente ao sucesso de um sistema computacional no atendimento de sua especificação, um defeito (*failure*), trata-se de um desvio da especificação. Não pode ser tolerado, mas deve ser evitado. Um sistema encontra-se em erro, ou em estado errôneo, se o próximo processamento a partir desse estado, pode levar a um defeito. A causa física ou algorítmica de erro, por fim, é denominada falha, conforme mostra a Figura 2.1.

2.1.2 Confiabilidade e disponibilidade

Confiabilidade e disponibilidade são medidas probabilísticas, uma vez que a falha é um fenômeno aleatório.

Na equação (2.1), o valor da confiabilidade (R) é dado em função da taxa de falhas (SIEWIOREK; SWARZ, 1992).

$$R(t) = e^{\int_0^t l\,(t)\,d(\,t)} \tag{2.1}$$

Nesta equação, λ (t) é a taxa de falhas no intervalo de tempo Δt. A função fornece uma avaliação quantitativa do desempenho de componentes ou sistemas.

A taxa de falhas de componentes é especificada pelo fabricante. Em ensaios laboratoriais, verificam-se quantos dos componentes submetidos aos testes sobreviveram, ou seja, quantos mantiveram operação correta, durante o período de testes. Usualmente, a taxa de falhas é expressa em FITS (uma falha por 10^9 horas) (SIEWIOREK; SWARZ, 1982).

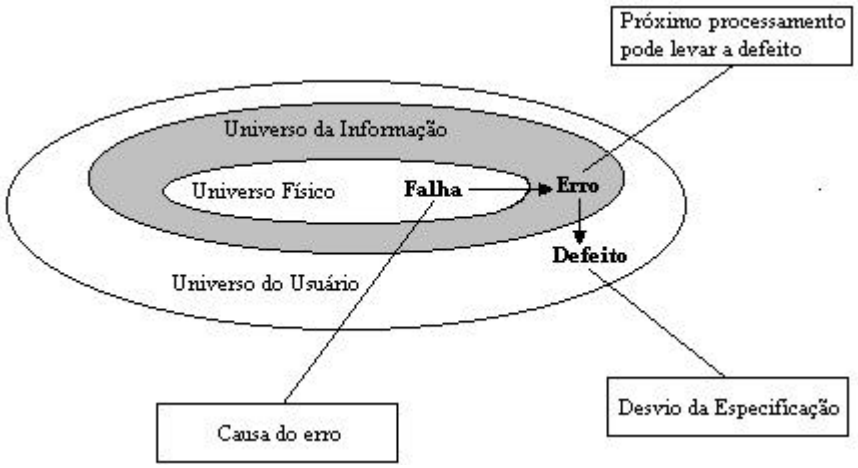


Figura 2.1 – Modelo dos universos de falha, erro e defeito.

A Figura 2.2 apresenta a variação típica da taxa de falhas com o tempo. Nela, pode-se observar três regiões distintas (HAYES, 1988):

I- Região de falhas prematuras, com taxa de falhas decrescente. Os componentes apresentam defeito logo após serem postos em funcionamento (mortalidade infantil) (CLEMENTS, 1991).

II- Região de operação normal, com taxa de falhas praticamente constante.

III- Região de desgaste, com taxa de falhas crescente. Isso se deve a degradação dos componentes com o decorrer do tempo. Esse período compreende o tempo de vida útil de componentes eletrônicos.

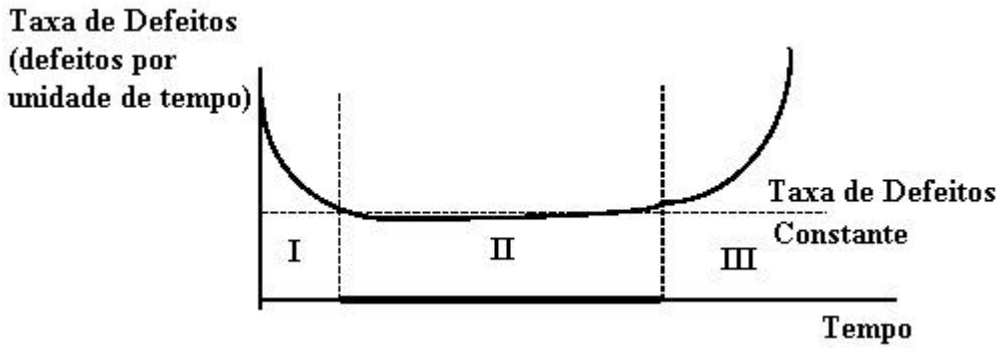


Figura 2.2 – Variação típica da taxa de falhas com o tempo.

Durante o período de vida útil, a confiabilidade dos componentes individuais é obtida através da equação (2.2).

$$R(T) = e^{-lt} \tag{2.2}$$

Circuitos integrados com baixa escala de integração (SSI), têm taxas de falhas de 10^3 a 10^1 FITS, enquanto os com alta escala de integração, da ordem de 10^4 a 10^3 FITS (FISCHER, 1990).

Outros parâmetros relacionados com a confiabilidade de um sistema são (HAYES, 1988):

- ✓ Tempo Médio entre Falhas (MTBF – Mean Time Between Failure): indica o tempo médio de funcionamento correto de um sistema reparável, entre sucessivas falhas.

- ✓ Tempo Médio para Ocorrência de Falhas (MTTF – Mean Time To Failure): determina, em sistemas não reparáveis, o tempo médio, até a ocorrência da primeira falha.

- ✓ Tempo Médio para Reparo (MTTR – Mean Time To Repair): especifica o grau de recuperação ou de conserto do sistema.

O tempo médio entre falhas pode ser obtido com auxílio da equação (2.3). Como se pode notar, o tempo médio entre falhas é igual ao tempo de reparo somado ao tempo médio para ocorrência de falhas.

$$MTBF = MTTR + MTTF \tag{2.3}$$

A disponibilidade traduz a probabilidade de um sistema executar uma tarefa, no instante em que esta é requerida. É usada para medir quão freqüentemente um sistema falha e quão rapidamente é restaurado (SIEWIOREK; SWARZ, 1992). Tanto a equação 2.4 (CLEMENTS, 1991), como a equação 2.5 (HAYES, 1988), podem ser usadas para especificação da disponibilidade (A) de um sistema.

$$A = \frac{MTBF}{MTBF + MTTR} \tag{2.4}$$

$$A = \frac{MTTF}{MTTF + MTTR} \tag{2.5}$$

Em suma, sistemas que devem estar prontos para atender a uma solicitação de serviço a qualquer momento, mas que podem ser paralisados para manutenção e reparo, como sistemas bancários e estações telefônicas, por exemplo, são sistemas onde a disponibilidade é um fator extremamente importante. A confiabilidade, por outro lado, é um requisito imprescindível em sistemas onde o reparo nem sempre é possível, como em satélites espaciais e computadores de bordo.

2.1.3 Utilização de recursos redundantes

A técnica de tolerância à falhas trata-se de um atributo que é designado ao sistema, para se atingir metas de projeto. Conforme já citado, sua função é reagir à ação da máquina física, permitindo a máquina lógica operar corretamente, creditando altos índices de confiabilidade e disponibilidade aos sistemas (AHUJA; MISHRA, 1997). Essa técnica envolve duas etapas (SIEWIOREK, 1992):

1- Detecção de erros ou falhas nos sistemas;
2- Recuperação do sistema, para que o mesmo continue sua operação normal.

Para que um sistema possa detectar erros ou falhas e tenha a habilidade de se reconfigurar, o mesmo deve possuir recursos redundantes. Esses recursos não são estritamente necessários para o funcionamento de um sistema, mas sim para detecção e tolerância de falhas e reconfiguração (BOTTA, LOPES; 2002). O uso de redundância, na verdade, é o alicerce da técnica de tolerância à falhas. Esta pode apresentar-se em uma das seguintes formas:

- Redundância de hardware;
- Redundância de software;
- Redundância de tempo;
- Redundância de informação.

A redundância de hardware baseia-se no uso de réplicas de componentes e circuitos. A redução no tamanho dos componentes eletrônicos e seu baixo custo são imensos incentivos para utilização desta técnica (ANISIMOV et al., 2002). No uso de repetição de componentes e circuitos, devem ser considerados fatores como desempenho, tamanho, peso e consumo de energia (CHENG et al., 2000).

A base da redundância de software é a inclusão de recursos de programação, que auxiliam na detecção de erros e recuperação de sistema. Um exemplo simples é incluir num programa, uma rotina para escrever e ler em posições aleatórias da memória, em intervalos de tempo predeterminados (DISHON; GEORGIOU, 1987).

Com a repetição de operações, que implicam na redundância de tempo, pode-se detectar uma possível diferença entre os resultados das mesmas. Isso permite a distinção entre falhas transientes e permanentes. O grande problema desta técnica é que dificilmente, se houver uma falha, ter-se-á todos os mesmos dados para um novo teste (SIEWIOREK; SWARZ, 1982).

A redundância de informação, por sua vez, envolve o uso de informação extra (códigos detectores de erros) na estrutura de dados básica, para detecção e em alguns casos, correção de erros (JOHNSON, 1984). Um dos códigos mais simples e conhecidos é a verificação da paridade (BROOKSHEAR, 2000).

2.1.4 Emprego de sistemas computacionais tolerantes a falhas

As aplicações para sistemas de computação tolerantes a falhas, podem ser classificadas basicamente em quatro categorias:

- Aplicações de longa duração;
- Tarefas críticas;
- Adiamento de manutenção;
- Alta disponibilidade.

Sondas, foguetes, satélites e missões não tripuladas são exemplos de aplicações de longa duração. Algumas devem ter uma probabilidade de operação confiável por períodos tão longos como dez anos. Estes sistemas são altamente redundantes e podem ser reconfigurados automaticamente ou através de estações remotas na Terra.

Aparentemente, as aplicações tolerantes a falhas mais conhecidas, são as referentes à gestão de tarefas críticas. Na execução destas, atrasos e falhas não são tolerados (WENSLEY, 1982). O controle de processos em tempo real, em aeronaves, sistemas militares e certos tipos de controles industriais, são exemplos de tarefas críticas (BUTLER et al., 1985).

Estações localizadas remotamente no espaço, controle de transportes coletivos, como o metrô, e aplicações em ambientes inóspitos, são representantes da categoria de aplicações que requerem adiantamento da manutenção. Neste caso, o sistema deve ser capaz de tolerar as possíveis falhas, que venham a ocorrer, entre uma visita e outra, da equipe responsável pelo reparo.

Um dos parâmetros que deve ser considerado em muitas aplicações, como em atividades comerciais e bancárias, é a disponibilidade. Os usuários destes serviços desejam, e realmente necessitam, de respostas num tempo reduzido, praticamente ao mesmo tempo em que são requisitadas (SIEWIOREK; SWARZ, 1992).

Certamente, quanto maior a dependência humana a sistemas computacionais, maior a necessidade de desenvolvimento de técnicas cada vez mais apuradas de controle de ocorrência de erros. Atualmente, vários pesquisadores, como Rennels e Hwang (2001), se dedicam ao estudo da aplicação de técnicas de tolerância à falhas a sistemas embutidos, baseados em microcontroladores. Subseqüentemente, são consideradas as principais características destes dispositivos.

2.2 MICROCONTROLADORES

2.2.1 Definição

O microcontrolador, também conhecido como computador em um só chip, vem sendo utilizado em escala cada vez maior na automação e controle de processos. Um microcontrolador pode ser definido como um circuito integrado, que contém um processador e seus periféricos típicos. Em virtude de suas dimensões reduzidas, pode ser alojado no próprio produto que gerência, como ocorre no mouse e no teclado, periféricos típicos de um microcomputador (Silva Jr et al., 2002).

Dependendo da aplicação, a Unidade de Microcontrolador (MCU – Micro Controler Unit), deve agregar características diferentes. Deste modo, a escolha de uma MCU deve ser feita com base nos requisitos do sistema (Vaglica; Gilmour, 1990).

Um microcontrolador pode ser classificado de diversas maneiras. A forma mais usual, diz respeito ao barramento de dados. De acordo com esta classificação, um microcontrolador pode se inserir em uma das quatro categorias:

- MCUs de 4 bits;
- MCUs de 8 bits;
- MCUs de 16 bits;
- MCUs de 32 bits.

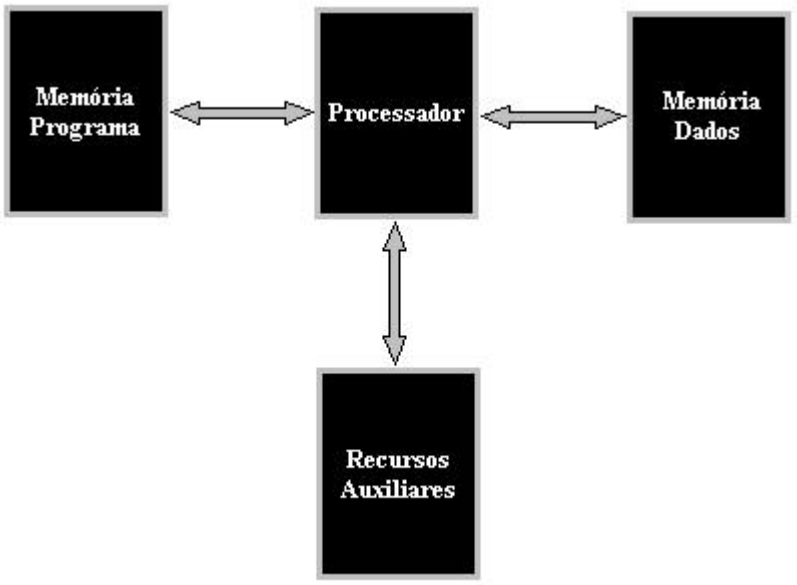


Figura 2.3 – Estrutura básica de um microcontrolador típico.

Atualmente, os microcontroladores de 8 bits, são os mais utilizados, devido a seu custo, diversidade e versatilidade. A Figura 2.3 apresenta a estrutura básica de um microcontrolador típico. Como se pode observar, este é composto de:

- Processador;
- Memória de dados;
- Memória de programa;
- Recursos auxiliares.

O comprimento da palavra armazenada na memória de dados não é necessariamente igual ao da memória de programa. Portanto, cabe ressaltar, que quando se diz que um microcontrolador é de 16 bits, por exemplo, faz-se referência à memória de dados, uma vez que, dependendo da arquitetura do microcontrolador, o tamanho da memória de dados e de programa pode ser diferente, conforme será visto adiante (MICROCHIP, 1997).

2.2.2 Memória de dados

É utilizada para armazenar registradores e variáveis. Permite a escrita e leitura de dados e pode ser de um dos seguintes tipos fundamentais (PEREIRA, 2002):

- **RAM (Ready Only Memory – Memória somente para leitura):** Trata-se de uma memória volátil, ou seja, quando o microcontrolador deixa de ser alimentado eletricamente, os dados armazenados nela, são perdidos. Em uma RAM, o processo de leitura e escrita é bastante rápido.

- **EEPROM (Eletrically Erasable Programable Read Only Memory – Memória apenas de leitura que se pode programar e apagar eletricamente):** A EEPROM é uma memória não volátil, portanto, os dados nela contidos, não são perdidos quando o microcontrolador é desenergizado. O processo de leitura e gravação em uma memória EEPROM é lento.

2.2.3 Memória de Programa

É uma memória não volátil (as informações não são perdidas quando o sistema é desligado) que acondiciona as instruções do programa que gerencia uma aplicação. Pode ser de um dos tipos abaixo (NICOLOSI, 2000):

- **ROM (Read Only Memory – Memória apenas de leitura) com máscara:** As instruções são gravadas nos chips, durante o processo de fabricação e não podem mais ser alteradas ou excluídas. Possuem baixo custo e são viáveis somente para grandes demandas.

- **OTP:** Dispositivos fabricados com memória PROM (Programmable Read Only Memory – memória apenas de leitura programável), podem ser gravados pelo usuário uma única vez, por intermédio de um gravador ligado a um microcomputador.

- **EPROM (Erasable Programable Read Only Memory – Memória apenas de leitura que se pode programar e apagar):** As instruções são gravadas como nos dispositivos OTP, com auxílio de um gravador e um microcomputador. Neste caso, porém, as informações podem ser excluídas e gravadas diversas vezes. A exclusão é feita submetendo o chip, que possui em sua face superior uma janela de cristal, a raios ultravioletas.

- **EEPROM (Eletrically Erasable Programable Read Only Memory – Memória apenas de leitura que se pode programar e apagar eletricamente):** As informações podem ser escritas e apagadas eletricamente, sem a necessidade de raios ultravioletas.

- **FLASH:** Como na EEPROM, as informações podem ser excluídas ou escritas eletricamente pelo usuário, utilizando-se um gravador conectado a um microcomputador.

2.2.4 Processador

O Processador ou Unidade Central de Processamento (CPU – Central Processing Unit) é o responsável pela interpretação e execução das instruções do programa (BROOKSHEAR, 2000).

Todo e qualquer processador digital é composto por duas partes, a saber:

1- **Unidade de Controle:** compreende os circuitos que gerenciam todas as atividades do sistema.

2- **Unidade de Lógica e Aritmética (ULA):** possui os circuitos que manipulam os dados.

2.2.5 Recursos Auxiliares

A capacidade de memória e a quantidade de recursos auxiliares determinam a potencialidade e a complexidade de um microcontrolador. Alguns dos recursos auxiliares mais conhecidos são (VAGLICA; GUILMOUR, 1990):

- Conversor Analógico/Digital (A/D) e Digital/Analógico (D/A);
- Modo de repouso ou de baixo consumo (SLEEP);
- Portas de I/O (Input/Output – Entrada/Saída);
- Cão de Guarda (Watchdog);
- Temporizadores;
- Canais Seriais;
- Interrupções.

2.2.6 Arquitetura

O desempenho de um microcontrolador depende muito de sua arquitetura interna. As arquiteturas de computadores mais populares são a Von Neumann, com conjunto de instruções complexas (CISC – Complex Instuction Set Computer) e a Harvard, com conjunto reduzido de instruções (RISC – Reduced Instruction Set Computer).

O popular microcontrolador 8051 (INTEL, 2002), fabricado pela Intel, é um exemplo de dispositivos CISC, enquanto que os microcontroladores PIC (MICROCHIP, 1997), fabricados pela Microchip, e os AVR (ATMEL, 2002.a) (ATMEL, 2002.b), fabricados pela Atmel, são exemplos de dispositivos RISC. A Atmel produz também, assim como outras empresas, dispositivos CISC, compatíveis com a família 8051 (SCHUNK, 2001) (ATMEL, 2002.c).

1º Ciclo Instrução	2º Ciclo Instrução	3º Ciclo Instrução	4º Ciclo Instrução
Fase de Busca 1ª Instrução	Fase de Execução 1ª Instrução	Fase de Busca 2ª Instrução	Fase de Execução 2ª Instrução

Figura 2.4 – Arquitetura Von Neumann: busca e execução de uma instrução.

1º Ciclo Instrução	2º Ciclo Instrução	3º Ciclo Instrução	4º Ciclo Instrução
Fase de Busca 1ª Instrução	Fase de Execução 1ª Instrução		
	Fase de Busca 2ª Instrução	Fase de Execução 2ª Instrução	
		Fase de Busca 3ª Instrução	Fase de Execução 3ª Instrução
			Fase de Busca 4ª Instrução

Figura 2.5 – Arquitetura Harvard: busca e execução de uma instrução.

Uma vantagem dos dispositivos RISC, em relação aos CISC, é a velocidade de execução de programas. Uma desvantagem, é que operações matemáticas, são mais facilmente executadas por dispositivos CISC (SCHUNK, 2001).

Na arquitetura creditada a John von Neumann (talvez incorretamente[1]), existe um único barramento entre memórias e CPU, por onde trafegam dados e instruções. Já na arquitetura Harvard, existem barramentos independentes, um para dados e outro para instruções. Isto permite que dados e instruções tenham comprimentos diferentes. Além disso, viabiliza o uso da técnica denominada pipelining, que possibilita que as fases de busca e execução de uma

[1] "Alguns reivindicam que esta idéia foi desenvolvida originalmente por J. P. Eckert, Jr., na Moore School, mas suas idéias resultaram de uma pesquisa em grupo, tendo sido, no final, indevidamente atribuída a von Neumann" (BROOKSHEAR, 2000).

instrução ocorram simultaneamente. Assim, enquanto na arquitetura Von Neumann, a busca e execução de uma instrução são realizadas seqüencialmente (Figura 2.4), na arquitetura Harvard, enquanto uma instrução é executada, outra é buscada na memória (Figura 2.5). Pragmaticamente, portanto, microcontroladores construídos com arquitetura Harvard e tecnologia RISC, são mais rápidos que os construídos com arquitetura Von Neumann e tecnologia CISC. Como exemplo, o popular 8051 da Intel, rodando a 12MHz, executa a maioria das instruções em 1µs, enquanto um PIC da Microchip, executa a maioria de suas instruções em 1µs rodando a 4MHz (VIDAL, 1997).

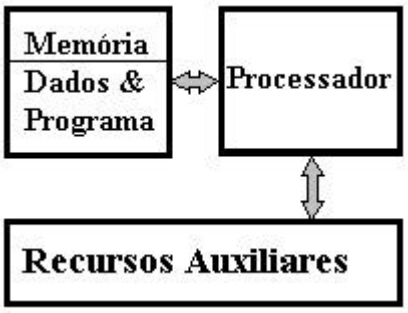


Figura 2.6

Arquitetura Von Neumann.

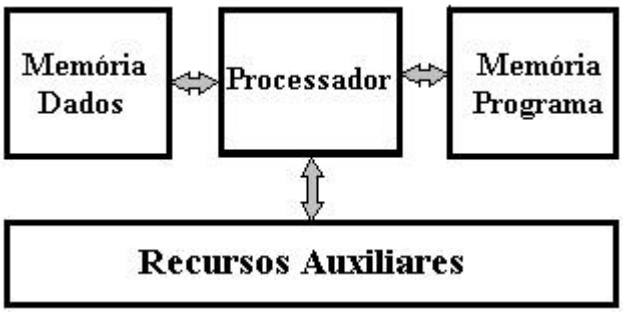


Figura 2.7

Arquitetura Harvard.

A Figura 2.7 mostra a estrutura básica de um microcontrolador construído com arquitetura Harvard, ao passo que a Figura 2.6, apresenta a de um microcontrolador com arquitetura Von Neumann. É conspícuo, que para um microcontrolador com arquitetura Von Neumann, existe uma memória única, onde são armazenados dados e instruções, ao passo que, em um microcontrolador com arquitetura Harvard, a memória onde são gravadas as instruções (memória de programa), é distinta da memória onde são armazenados os dados.

2.2.7 Seleção de um Microcontrolador

A maioria dos fabricantes de circuitos integrados possui sua linha de microcontroladores, tendo em vista a crescente demanda por tais dispositivos. Não existe, porém, uma MCU que possa ser considerada "a melhor entre todas". No momento da escolha, há que se levar em conta que para cada aplicação, existe um modelo que pode oferecer a melhor relação custo/benefício (VAGLICA; GUILMOUR, 1990). Abaixo estão alistados aspectos que devem ser observados:

- Velocidade de execução das instruções;
- Capacidade da memória de programa;
- Número de portas de entrada e saída;
- Tamanho da memória de dados;
- Periféricos inclusos;
- Restrições impostas pelo sistema, como consumo de energia, faixa de temperatura e tipo de encapsulamento.

Para as arquiteturas tolerantes a falhas descritas neste trabalho, foram usados microcontroladores PIC (Programmable Integrated Circuit – circuito integrado programável), fabricados pela Microchip.

Esta empresa, cuja matriz localiza-se em Chandler, Arizona, oferece atualmente, mais de uma centena de modelos de microcontroladores. Isso facilita encontrar modelos que atendam aos requisitos das mais diferentes aplicações. Outra vantagem, de se trabalhar com a família de microcontroladores PIC, é a facilidade encontrada na migração de um modelo para outro, uma vez que todos os modelos conservam certas características básicas e possuem um set de instruções bem parecido (MICROCHIP, 1997) (SOUZA, 2000). Hoje, em sua linha de produtos, encontram-se os controladores digitais de sinais (DSC – Digital Signal Controller), numa analogia aos processadores digitais de sinais (DSP – Digital Signal Processor). São dispositivos com memória de programa de 24 bits e longitude de dados de 16 bits.

Os microcontroladores PIC obtiveram ampla aceitação no mercado mundial. Uma evidência disso, é sua rápida ascensão no ranking mundial de vendas de microcontroladores de 8 bits. Conforme mostra a Figura 2.8, em 1990, a Microchip ocupava o vigésimo posto; em 1999, o segundo.

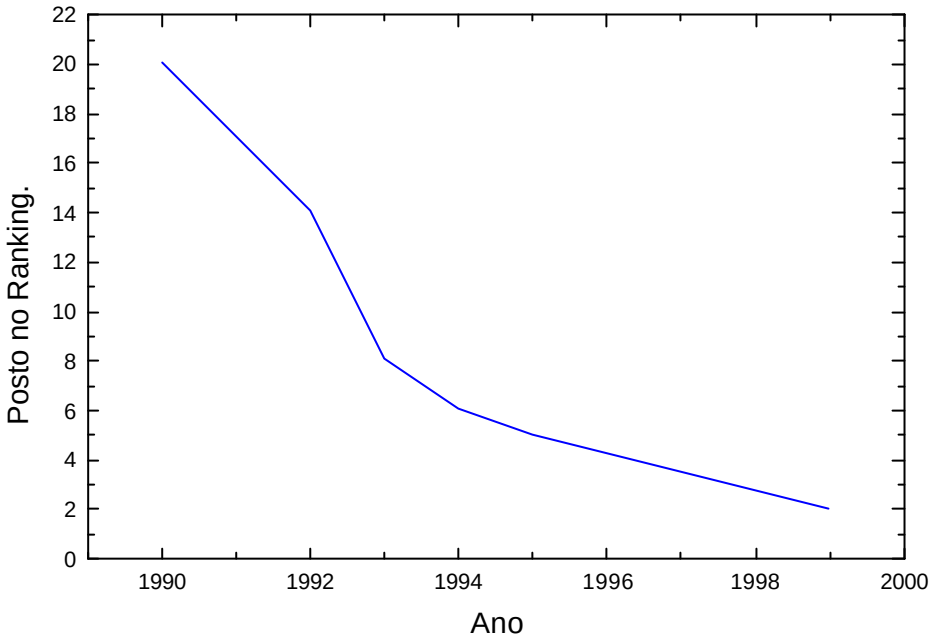


Figura 2.8 – Evolução de vendas.

Fatores que levaram a opção por tais dispositivos, estão alistados a seguir:

- Ampla variedade de modelos;
- Farta informação bibliográfica;
- Excelente relação custo/benefício;
- Facilidade de migração de um modelo para outro;
- Ferramentas de desenvolvimento econômicas e fáceis de manejar;
- Velocidade de execução de programas superior a dos principais concorrentes;
- O tamanho do código que seus programas geram é menor que dos principais concorrentes.

Dentre mais de setenta modelos que compõem a gama média (MICROCHIP, 1997), ou família intermediária, de microcontroladores da Microchip, o PIC 16F628 (MICROCHIP, 1999), que é analisado no Anexo I, foi eleito por possuir características tais, que atende aos requisitos da aplicação em questão (controle da temperatura), oferecendo excelente relação custo/benefício.

Algumas características do PIC 16F628 (MICROCHIP, 1999) são:

- Memória de programa FLASH com capacidade para 2048 instruções de 14 bits;
- Memória de dados EEPROM com capacidade para 128 palavras de 8 bits;
- Microcontrolador de 18 pinos, o que facilita a construção de protótipos;
- Memória de dados RAM com capacidade para 224 palavra x 8 bits;
- Comparadores analógicos com referência programável de tensão;
- Capacidade de corrente de 25mA por pino de I/O;
- Módulo de Captura, Comparação e PWM;
- Canal de comunicação serial USART;
- Dez fontes possíveis de interrupção;
- Possui 16 portas de entrada e saída;
- Temporizadores/Contadores;
- Oscilador interno;
- Watchdog.

O capítulo seguinte relata a construção das arquiteturas tolerantes a falhas para controle da temperatura, onde os microcontroladores PIC 16F628 foram usados.

CAPÍTULO 3

PARTE EXPERIMENTAL

3.1 MODELAGEM TEÓRICA

3.1.1 Introdução

Os microcontroladores são dispositivos eletrônicos largamente utilizados em aplicações de controle e automação, tais como: controle de motores, edifícios inteligentes, instrumentação eletrônica, visão computacional, sistemas de comunicação, controle de aplicações aeroespaciais, navegação por satélite, controle de periféricos de microcomputadores, sistemas de segurança, equipamentos hospitalares e robótica.

Levando em conta que na gestão de certos processos, a falha do sistema de controle é inadmissível, podendo implicar em conseqüências drásticas, foram projetadas duas arquiteturas tolerantes a falhas usando microcontroladores, a Arquiteturas TMR e a Anel. Ambas são capazes de tolerar falhas tanto nos nodos da estrutura, ou seja, nos microcontroladores, como nos arcos (FILHO; NUNES, 1995), que representam as ligações entre os mesmos.

A diferença entre a arquitetura em anel com três módulos e a TMR, aqui apresentadas, está no programa de gerenciamento. Deste modo, um mesmo circuito serve de base para operação de ambas as arquiteturas.

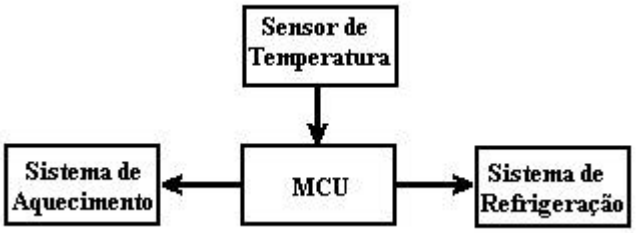


Figura 3.1 – Sistema para controle da temperatura de um ambiente.

A Figura 3.1 mostra um sistema simplex (arquitetura sem recursos de tolerância à falhas) para controlar a temperatura de um ambiente. Para manter a temperatura do ambiente dentro da faixa desejada, um sensor irá monitorá-la e informará constantemente as variações a um microcontrolador, responsável pelo controle do processo, para que este atue sobre um sistema de aquecimento ou de refrigeração, se necessário. É certo que uma falha no sensor ou no microcontrolador, pode levar a temperatura do ambiente a índices indesejados. Se o sistema, no entanto, for tolerante a falhas, a situação será outra. A Figura 3.2 apresenta um sistema tolerante à falhas, para controle da temperatura de um ambiente.

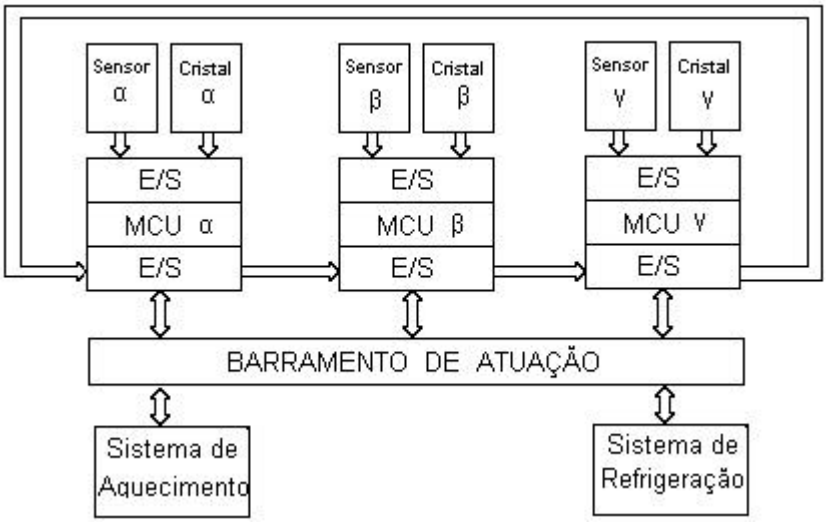


Figura 3.2 – Sistema tolerante a falhas para controle da temperatura de um ambiente.

Ao invés de um sensor para monitorar as mudanças de valor da temperatura, existem agora três. O mesmo ocorre com o número de MCUs. Os microcontroladores recebem informação de sensores diferentes e comunicam-se para verificar se há algum sensor com defeito e qual dos módulos será responsável pelo controle do sistema, ou seja, qual deles atuará sobre o sistema de refrigeração ou aquecimento. Desta forma, mesmo que ocorra falha em um dos microcontroladores ou em um dos sensores, o sistema de controle continuará operando corretamente. Logo, a técnica de tolerância à falhas agrega maior confiabilidade ao sistema de controle. Além disso, se ocorrer falha em um dos módulos, o mesmo poderá sofrer intervenção da manutenção sem implicar na desativação do sistema de controle, pois os demais módulos podem garantir a disponibilidade do serviço de controle da temperatura (BOTTA; LOPES, 2002.a).

3.1.2 Arquiteturas TMR e Anel

Três módulos trabalhando em sincronismo e executando as mesmas tarefas, com os resultados de cada módulo sendo submetidos a um único circuito votante por maioria (geralmente implementado por hardware), como mostra a Figura 3.3, constitui uma arquitetura TMR clássica. Uma saída permanece válida, desde que dois dos três módulos concordem e o circuito de votação esteja livre de falhas (HAYES, 1988). Os circuitos votantes podem ser implementados com a técnica de prevenção à falhas. Esta arquitetura suporta falhas em apenas um módulo. Uma extensão desta, é a arquitetura com redundância modular múltipla (NMR – N MODULAR REDUNDANCY), que pode suportar falhas em f módulos, onde N = 2f + 1 (SIEWIOREK; SWARZ, 1992). Na implementação aqui descrita, foi eliminado o circuito votante comum e a votação é executada por software. Quando um dos microcontroladores apresenta defeito, o sistema continua exercendo a tarefa de controle, desde que os outros dois módulos funcionem corretamente (BOTTA; LOPES, 2002.a).

A arquitetura em anel, (RODA, 1983), é um tipo de estrutura NMR, onde os módulos estão configurados num loop (anel) e sem circuito votante comum. Cada módulo recebe os dados de uma fonte, processa-os e envia o resultado do processamento para os outros módulos através de um canal de comunicação, havendo circulação dos dados através do anel. A circulação pode ser unidirecional ou bidirecional, podendo até

mesmo usar canais redundantes. Uma vez que os módulos têm dados suficientes, cada módulo executa a votação por programação, em concordância com um determinado algoritmo. O resultado da votação é indicado através de sinalizadores de erro que são utilizados para selecionar apenas um módulo (sem falhas), para dar saída ao sistema (FISCHER, 1990). A Figura 3.2 mostra uma estrutura em anel com três módulos e circulação unidirecional no sentido horário. Para votação dos dados podem ser usadas duas estratégias:

1- Votação majoritária: sobre todos os dados;
2- Votação sobre um número parcial de dados.

No primeiro caso, cada módulo recebe os dados de todos os outros para votação. No outro, a votação é feita sobre um número menor de dados, sendo que o limite, é a comparação de dados de dois módulos adjacentes.

A confiabilidade da arquitetura em anel foi analisada[1] levando-se em consideração:

1- Número de módulos;
2- Estratégia de votação.

Foram obtidas as seguintes conclusões:

1- Com o uso da estratégia de votação sobre a maioria dos dados, a confiabilidade decresce conforme aumenta o número de módulos.

2- Usando a estratégia de votação sobre um número parcial de dados, quando o número de dados é reduzido, há um aumento de confiabilidade.

[1] As limitações da arquitetura em anel com mais de três módulos e as expressões de confiabilidade para o sistema com estratégia de votação majoritária e parcial, podem ser encontradas, pelos leitores interessados, nos trabalhos realizados por Roda (1983) e Fischer (1990).

3- A arquitetura em anel, com três módulos, utilizando a estratégia de votação sobre um número parcial de dados, têm os mesmos valores de confiabilidade que a arquitetura TMR.

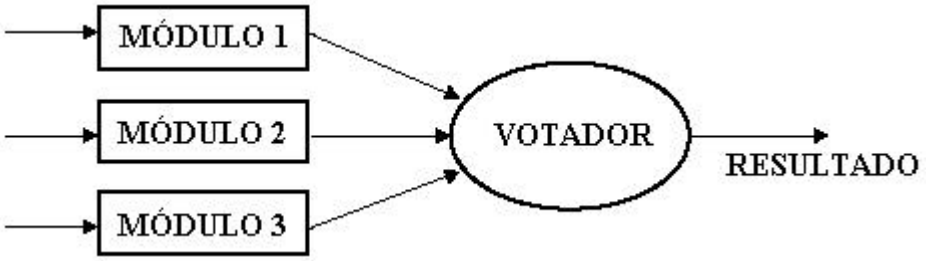


Figura 3.3 – Arquitetura TMR clássica.

Com base no acima exposto, para implementação da arquitetura em anel apresentada neste trabalho, foi usada a estratégia de votação sobre um número parcial de dados.

Subseqüentemente, serão consideradas as implementações de hardware e software, das arquiteturas tolerantes a falhas, para controle da temperatura de um determinado ambiente.

3.2 ANÁLISE EXPERIMENTAL

3.2.1 Hardware dos sistemas

Foi projetado e construído um circuito que pudesse suportar tanto a arquitetura em anel como a TMR, apresentadas nesse trabalho[2]. O circuito implementado é composto de três módulos idênticos, que se comunicam por intermédio das portas de entrada e saída dos microcontroladores. Cada módulo é constituído basicamente por fonte de alimentação ($5V_{CC}$), cristal oscilador (4MHz), sensor de temperatura e microcontrolador.

[2] Isso foi possível devido à arquitetura em anel implementada ser composta por três módulos. Caso fossem construídas com mais de três, o circuito seria diferente para o sistema NMR-anel e NMR-mascarável, devido à forma de comunicação entre os módulos. Para maiores detalhes, deve-se reportar aos trabalhos de Roda (1983) e Fischer (1990).

Para a aplicação destas arquiteturas, na gestão da temperatura de um determinado ambiente, foram utilizados microcontroladores PIC16F628, devido a este modelo possuir comparadores analógicos internos. Os sensores de precisão LM35 (NATIONAL, 2000), foram os selecionados para detecção das variações de temperatura. Estes dispositivos fornecem em um de seus terminais uma tensão elétrica, cujo valor, é de 10mV/ºC, linear em toda a faixa de operação, que vai de – 55 a +150ºC (NATIONAL, 2000). Desta forma, V_{OUT} é 300, –100, e 1200mV a 30, –10 e 120^{o}C, respectivamente. Sendo o consumo destes sensores de 60µA, o calor gerado no próprio dispositivo pode ser considerado desprezível, o que significa que praticamente não influi na medida. A Tabela 3.1 apresenta a faixa de operação de diferentes modelos de LM35. O modelo utilizado neste trabalho foi o LM35DZ (a última letra, no caso a letra Z, se refere ao tipo de encapsulamento do dispositivo).

Tabela 3.1 – Faixa de operação de diferentes modelos de LM35.

Modelo	Faixa de Operação
LM35/LM35A	-55 a +150^{o}C
LM35C/LM35CA	-40 a +110^{o}C
LM35D	0 a +100^{o}C

Um circuito bastante simples, com base no circuito integrado 7805, foi utilizado para implementação das fontes de alimentação estabilizadas. O diagrama da fonte pode ser visto na Figura 3.4.

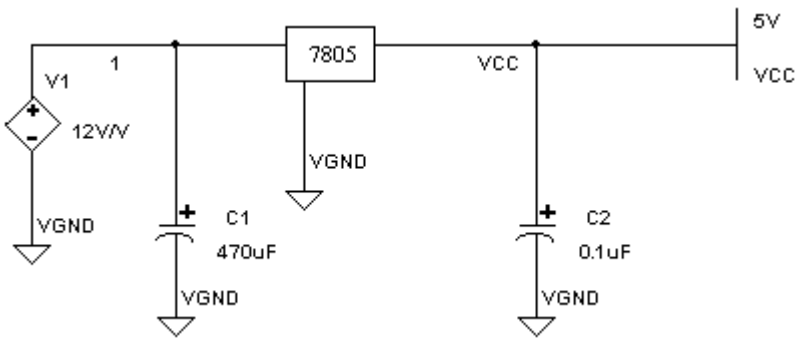


Figura 3.4 – Fonte de alimentação.

O diagrama do circuito de um módulo, das arquiteturas implementadas, é apresentado na Figura 3.5. Já nas Fotografias 3.1, 3.2, 3.3 e 3.4, é possível observar o protótipo das arquiteturas tolerantes a falhas, para gestão da temperatura de um determinado ambiente.

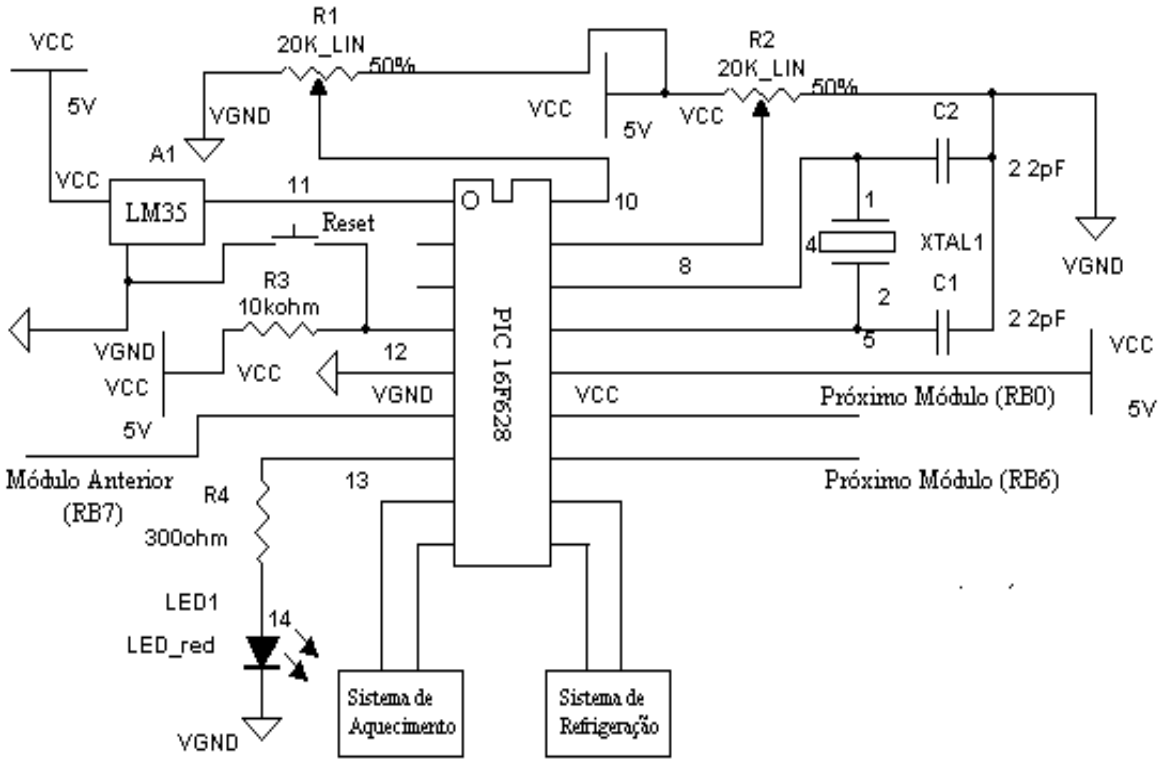
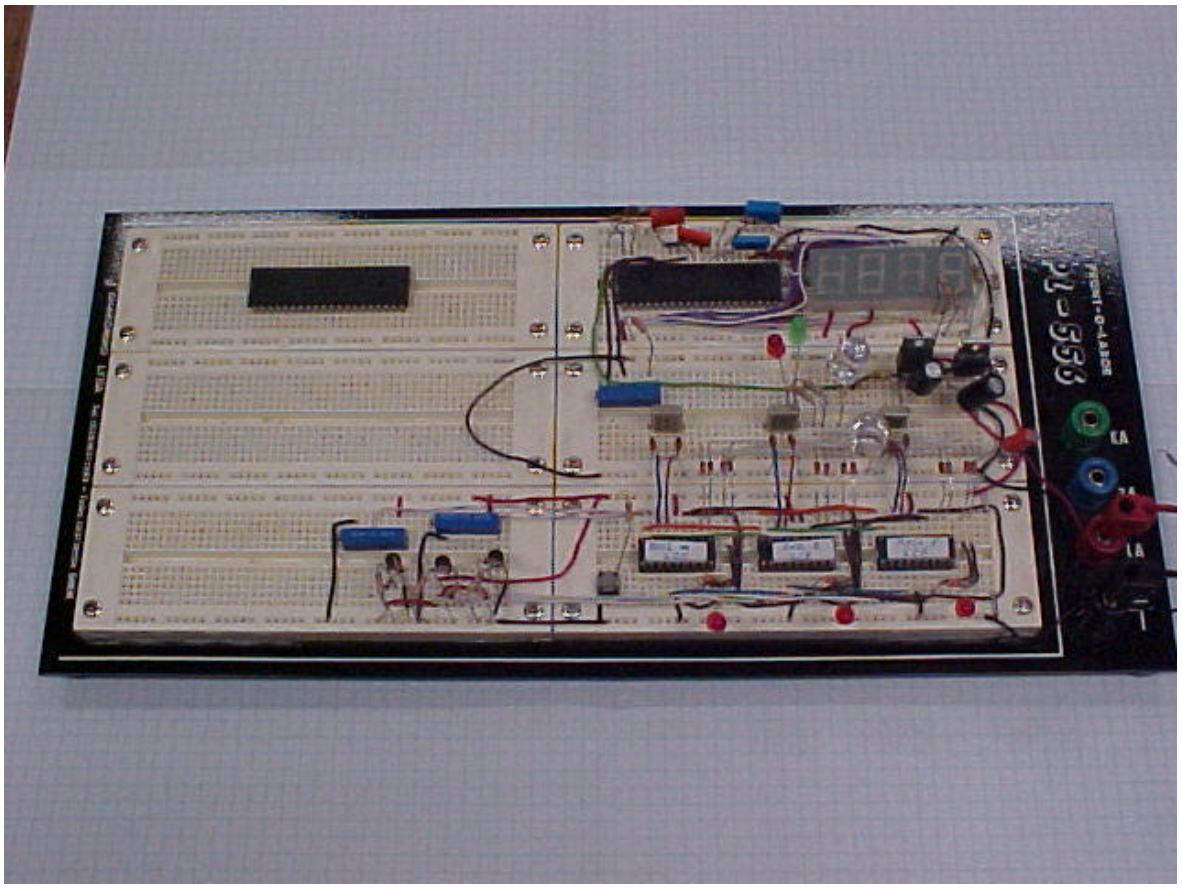
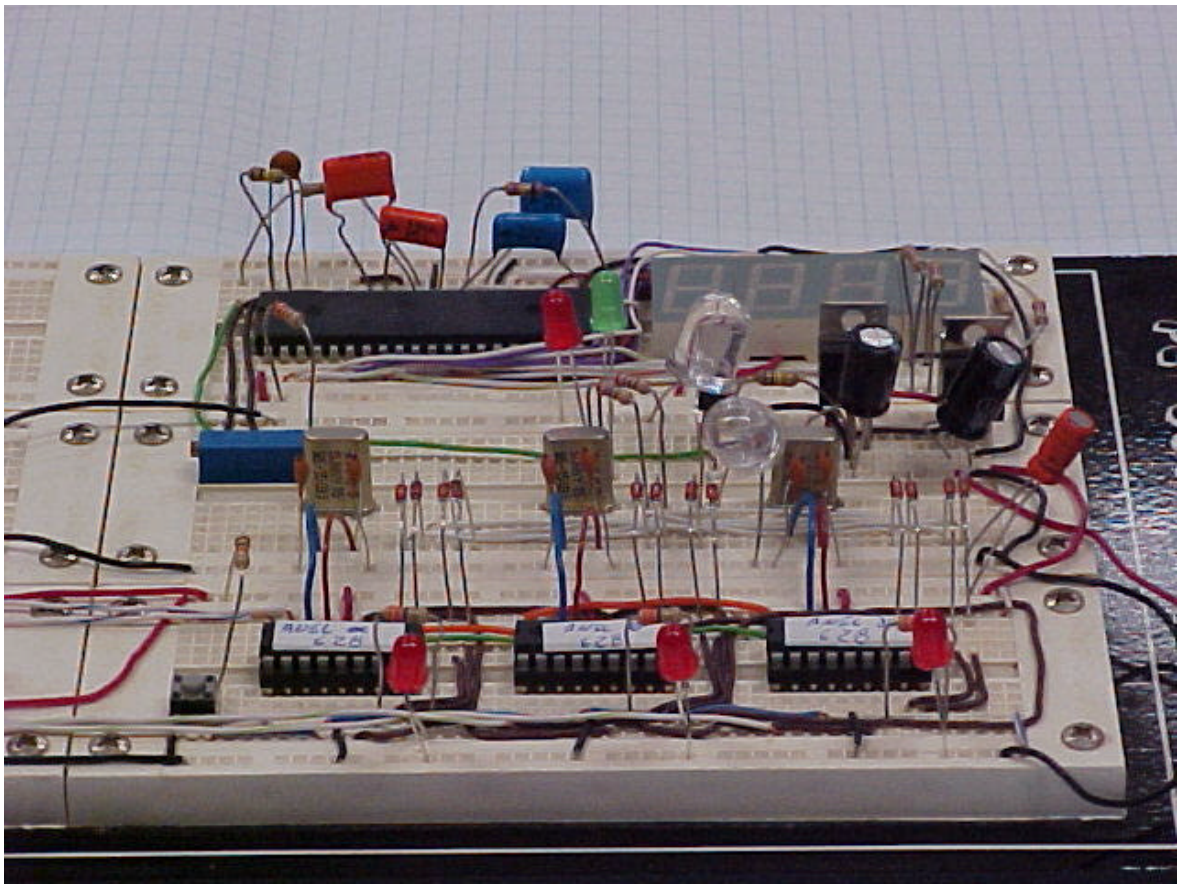


Figura 3.5 – Circuito de um módulo das arquiteturas.

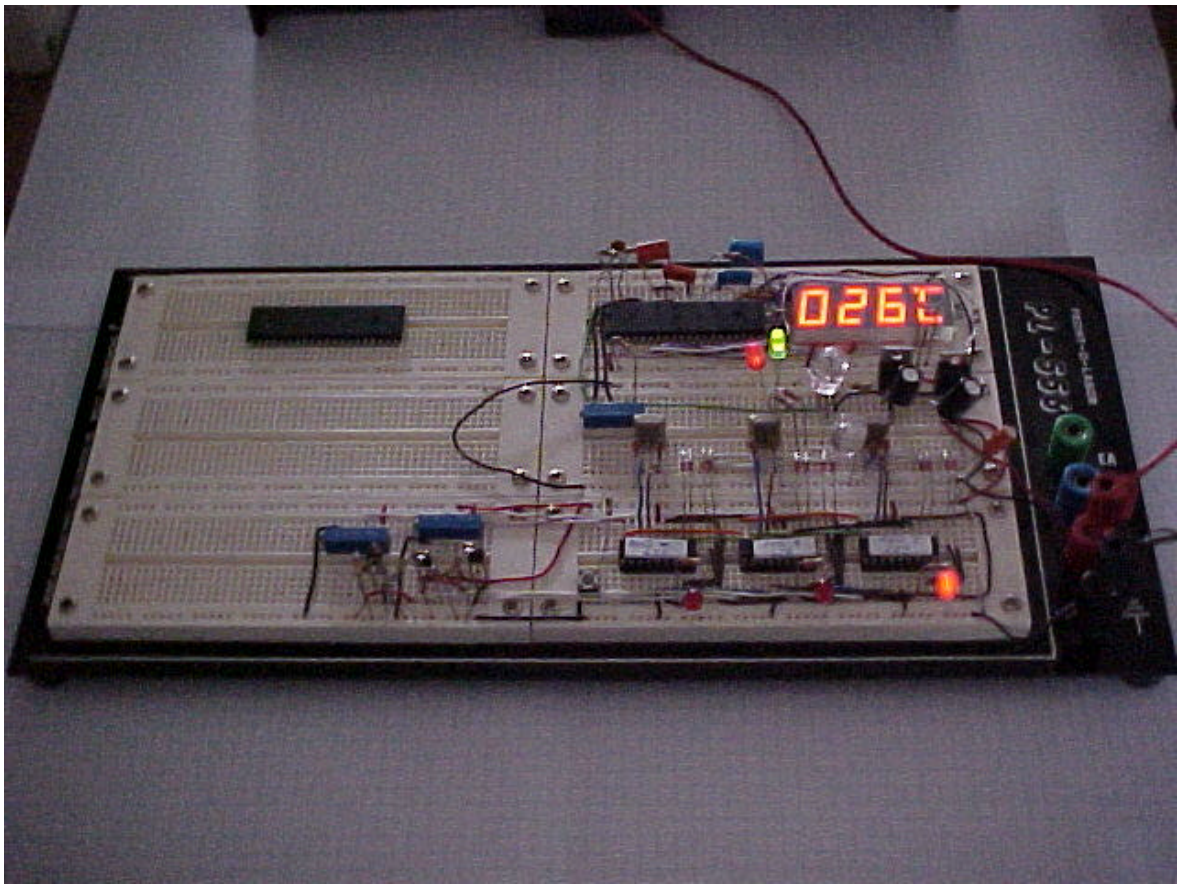
As arquiteturas aqui descritas foram originalmente construídas com MCUs PIC16F84A (MICROCHIP, 2001.b), como mostram as Fotografias 3.5 e 3.6. De início, foram construídas para propósito geral, sendo que, os microcontroladores recebiam 5 bits e representavam o valor destes por meio de LEDs. Embora já tenha sido mencionado, vale lembrar que uma grande vantagem de se trabalhar com a família de microcontroladores PIC, é que todos os modelos possuem um conjunto de instruções bem parecido, além de manterem muitas semelhanças entre suas características básicas (SOUZA, 2000). Isto facilita muita a migração de um modelo para outro.



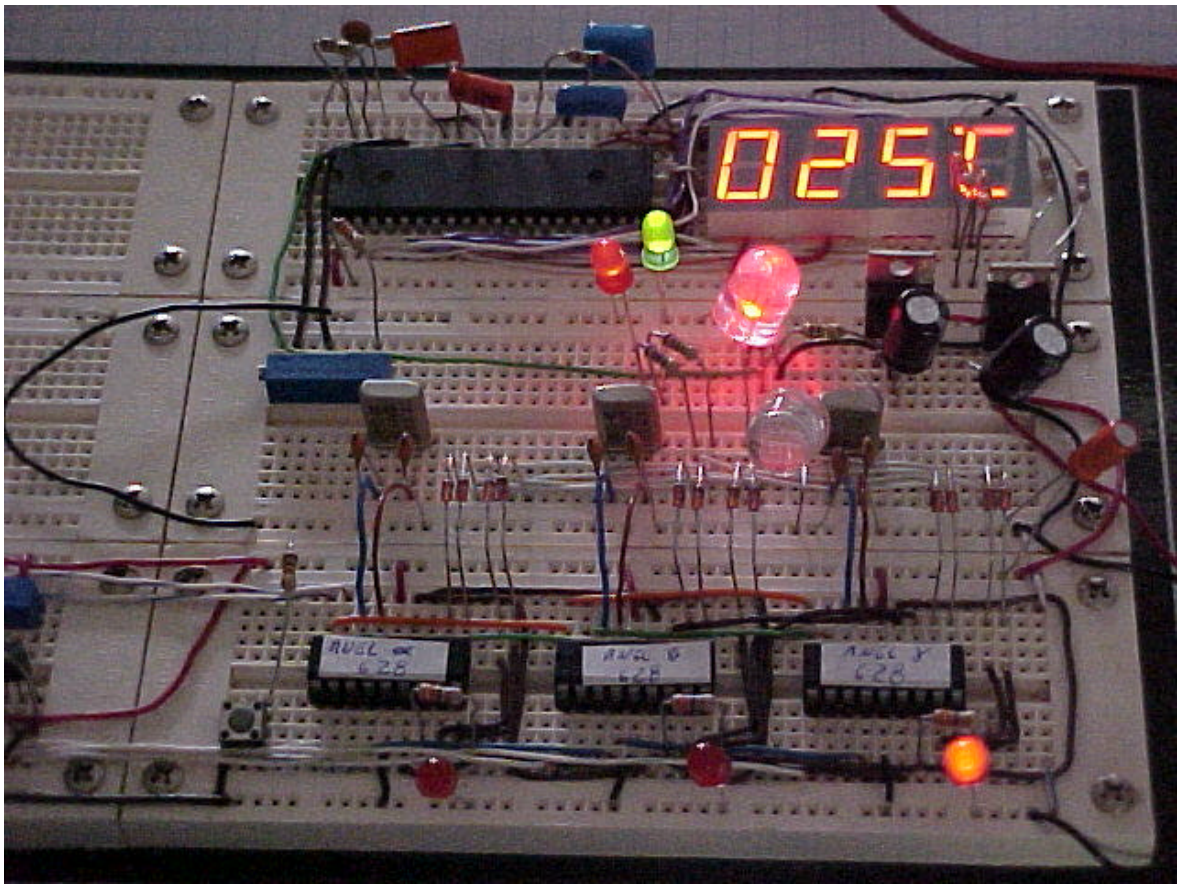
Fotografia 3.1 – Visão geral do protótipo do hardware das arquiteturas.



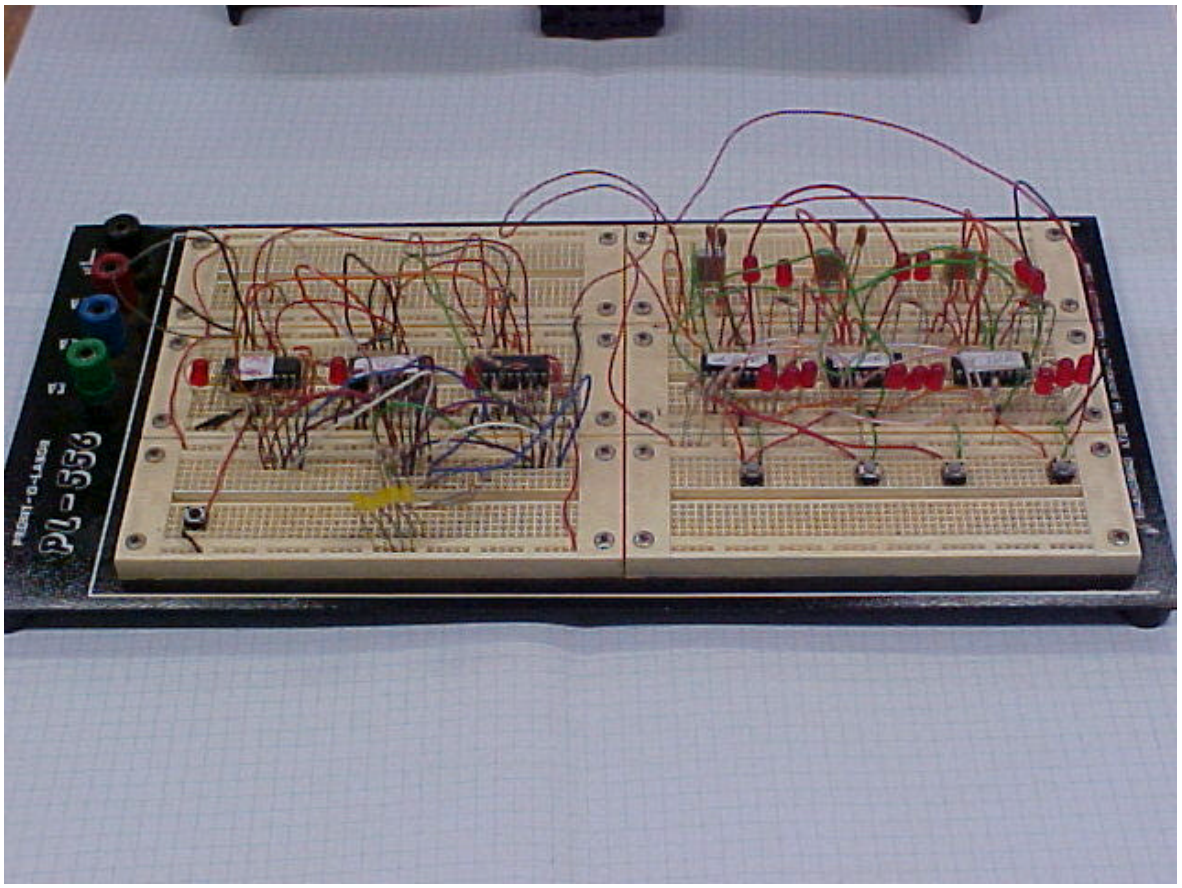
Fotografia 3.2 – Visão parcial do protótipo do hardware das arquiteturas.



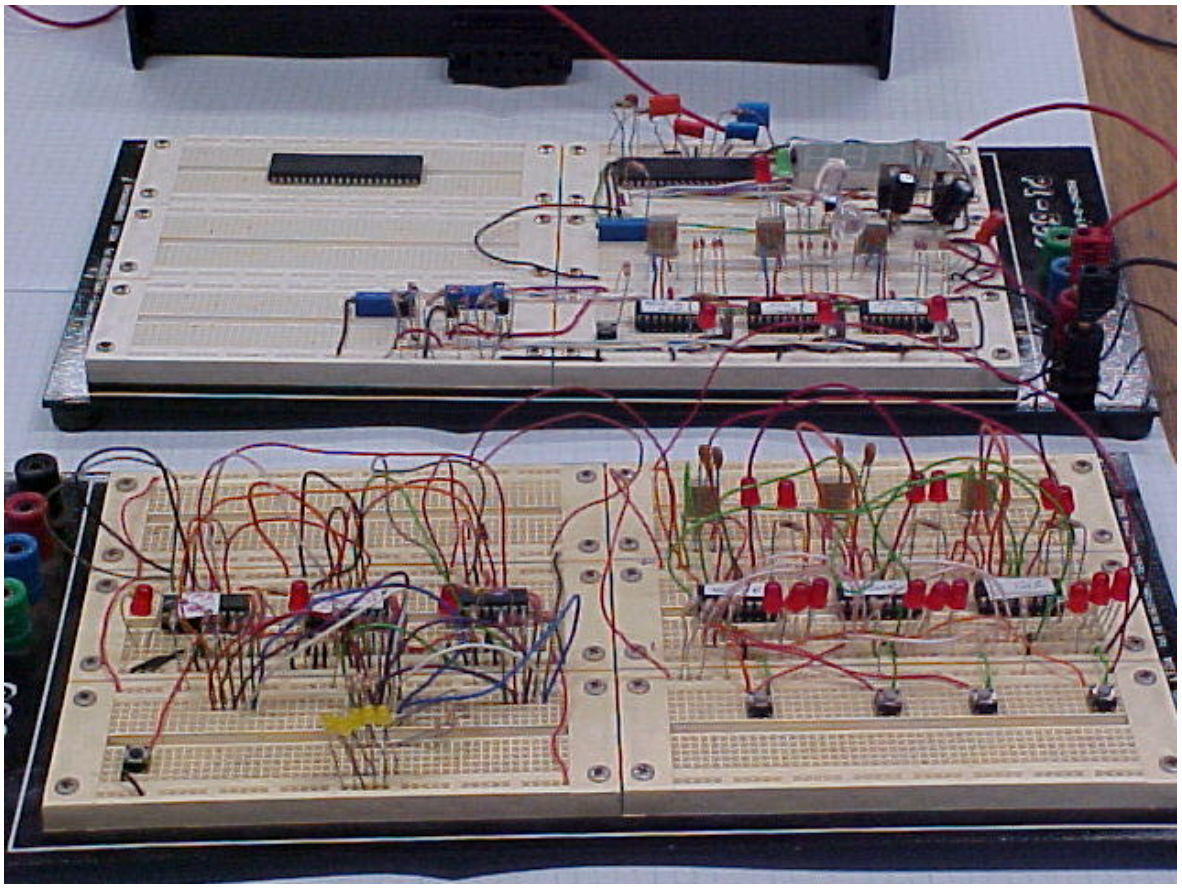
Fotografia 3.3 – Visão geral do protótipo das arquiteturas em operação.



Fotografia 3.4 – Visão parcial do protótipo das arquiteturas em operação.



Fotografia 3.5 – Primeiro protótipo construído, utilizando MCUs PIC 16F84A.



Fotografia 3.6 – Protótipo com PIC 16F84A e protótipo com PIC 16F628 e LM35DZ.

3.2.2 Software dos sistemas

Nos fluxogramas das Figuras 3.6 e 3.7, demonstra-se à maneira como funcionam os sistemas de controle com arquitetura em anel e TMR, respectivamente.

Os programas desenvolvidos foram escritos em linguagem assembly e podem ser alterados para a construção de sistemas com um maior números de módulos e/ou para o controle das mais diferentes tarefas. Para criação dos programas, foi utilizado o MPLAB (MICROCHIP, 2001), um ambiente integrado de programação (IDE), que contém módulos de edição, montagem e simulação de programas para microcontroladores PIC. Para uso da linguagem assembly, o MPLAB dispõe de um montador (assembler), denominado MPASM (PEREIRA, 2002). O ambiente MPLAB pode ser visto na Figura 3.8.

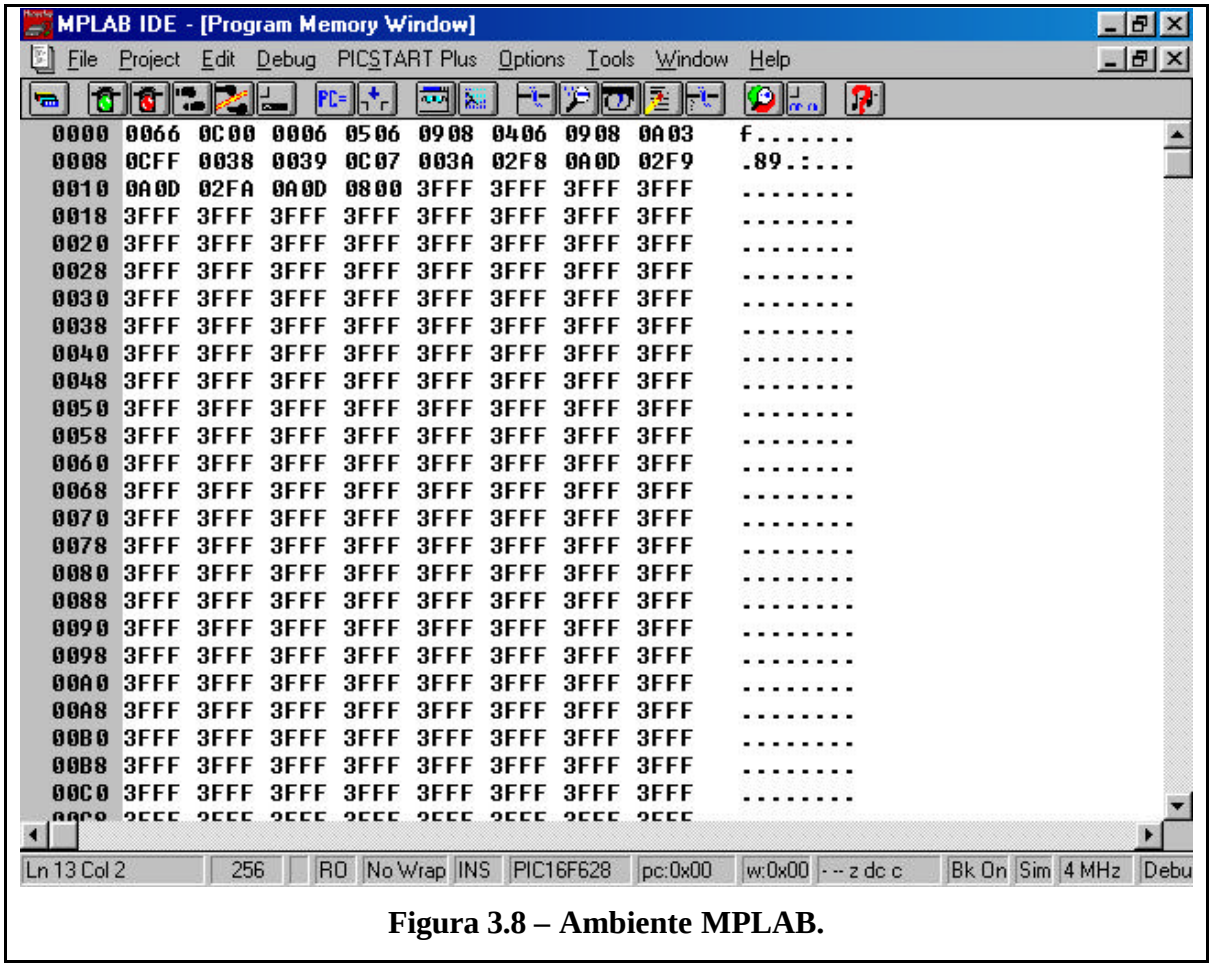


Figura 3.8 – Ambiente MPLAB.

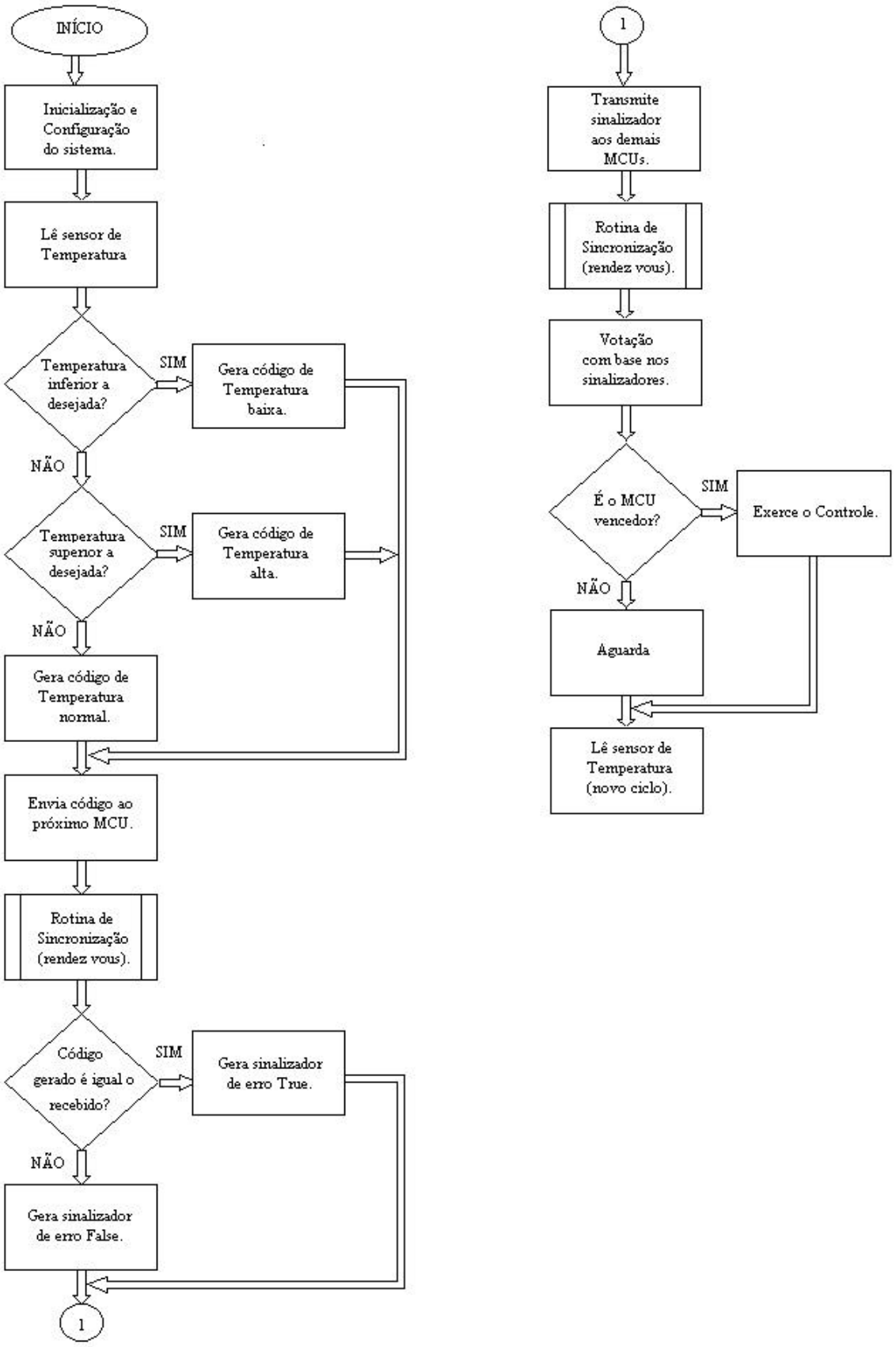


Figura 3.6 – Fluxograma do sistema de controle com arquitetura em anel.

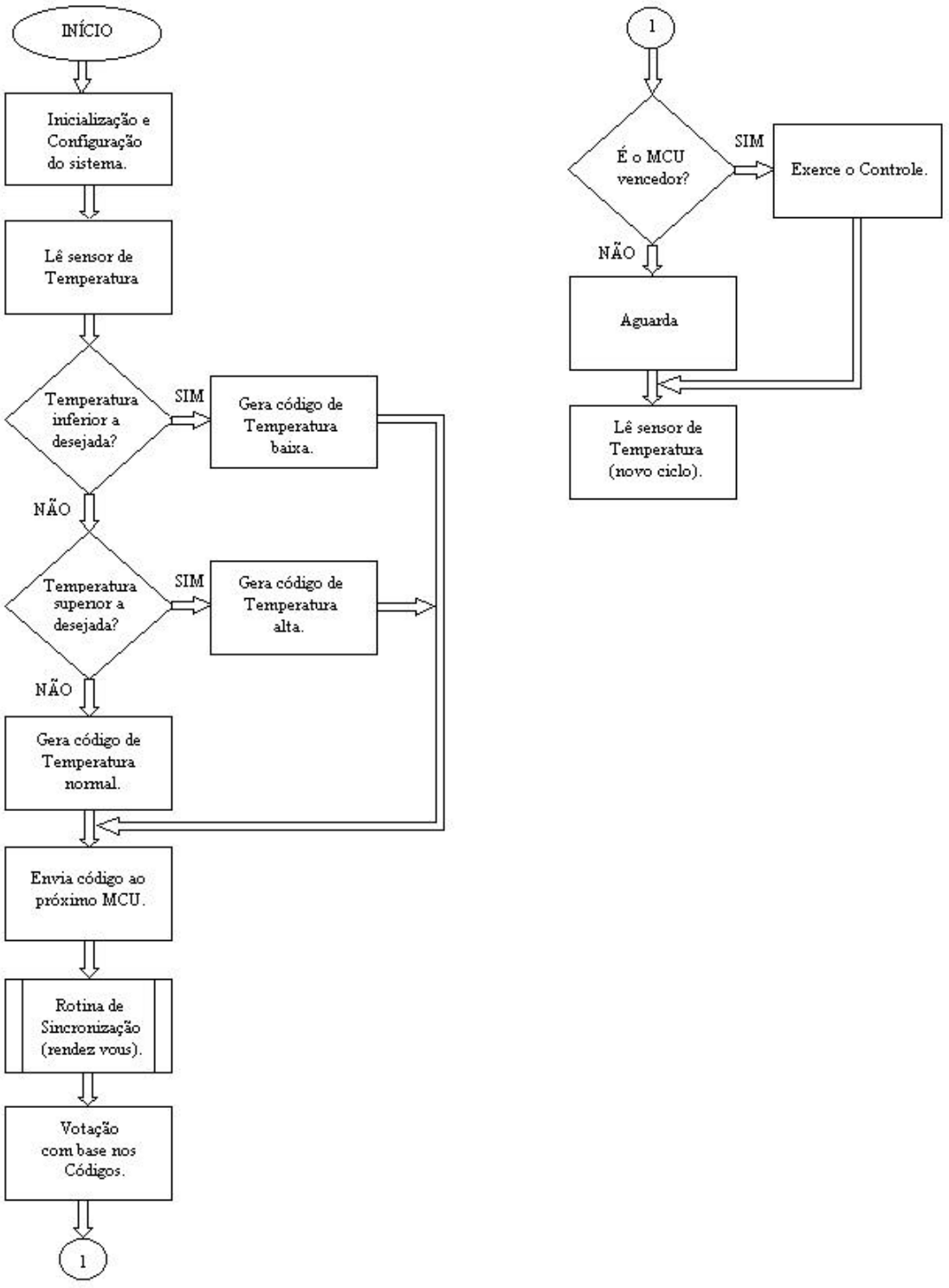


Figura 3.7 – Fluxograma do sistema de controle com arquitetura TMR.

Cada MCU possui uma cópia do programa desenvolvido, gravado em sua memória de programa. A seguir, é comentada a forma como os programas são estruturados para as diferentes arquiteturas.

3.2.3 Estrutura do software da arquitetura em anel

1) Inicialização: O PIC 16F628 dispõe de muitos recursos. Neste ponto é feita a configuração da maneira como ele irá trabalhar (utilizando o circuito oscilador interno ou um externo, quais portas serão utilizadas como entrada e quais serão utilizadas como saída, fontes de interrupção que estarão ativas, enfim, a configuração dos periféricos internos) e a alocação de registradores de propósito geral (GPR) para variáveis, constantes e flags que serão utilizados pelo sistema.

2) Aquisição e interpretação de dados: O sensor de temperatura é lido e é verificado se a temperatura está entre os limites desejados. Cada microcontrolador gera seu parecer.

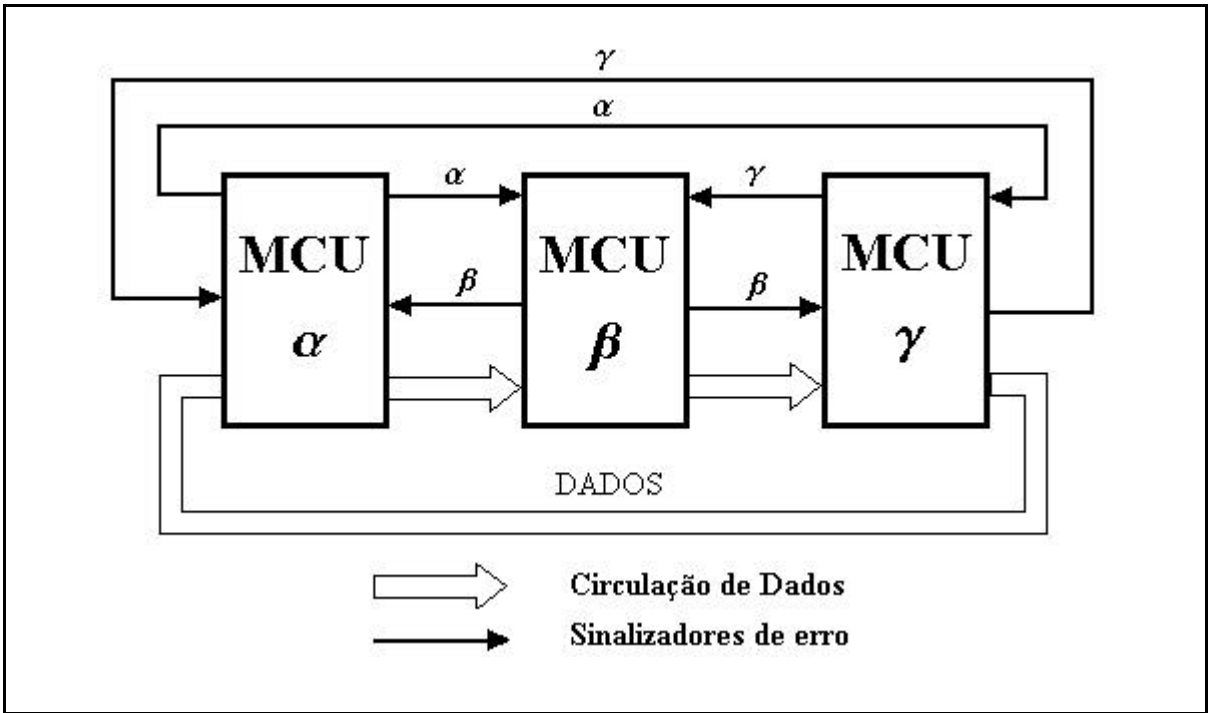


Figura 3.9 – Fluxo de dados e sinalizadores de erro na arquitetura em anel.

3) Transmissão de dados: A Figura 3.9 mostra a circulação dos dados no anel. Cada microcontrolador se comporta na transmissão de dados, em conformidade com a Tabela 3.2.

Tabela 3.2 – Transmissão de dados na arquitetura em anel.

Transmissão	MCUa	MCUß	MCU?
1º Ciclo	Transmite	Recebe	Aguarda
2º Ciclo	Aguarda	Transmite	Recebe
3º Ciclo	Recebe	Aguarda	Transmite

4) Comparação dos dados e geração do sinalizador de erro: Os microcontroladores comparam seu próprio parecer com o que recebeu. Com base na comparação, geram um sinalizador de erro.

5) Transmissão do sinalizador de erro: Cada microcontrolador recebe o sinalizador de erro dos outros dois, como mostra a Figura 3.9. Esse sinalizador é constituído de 8 bits e transmitido serialmente aos outros módulos.

6) Processo de votação: De posse do código de erro dos outros dois módulos, cada MCU consulta uma tabela (Tabela 3.3) para saber se é o vencedor, ou seja, se fará o controle do sistema. Para qualquer outra combinação não apresentada nesta tabela, nenhum microcontrolador exercerá a função de controle e os sistemas de aquecimento e refrigeração permanecerão desligados.

Tabela 3.3 – Processo de Votação.

Módulo a	Módulo ß	Módulo ?	Vencedor
Sem Falha	Sem Falha	Sem Falha	MCU?
Sem Falha	Sem Falha	Com Falha	MCUß
Sem Falha	Com Falha	Sem Falha	MCUa
Com Falha	Sem Falha	Sem Falha	MCU?

7) Controle pelo módulo vencedor: O módulo vencedor dá saída ao sistema, ou seja, atua sobre o sistema de refrigeração ou de aquecimento, conforme a necessidade.

3.3.3 Estrutura do software da arquitetura TMR

1) Inicialização: Análoga à arquitetura em anel.

2) Aquisição e interpretação de dados: Da mesma forma que a arquitetura em anel.

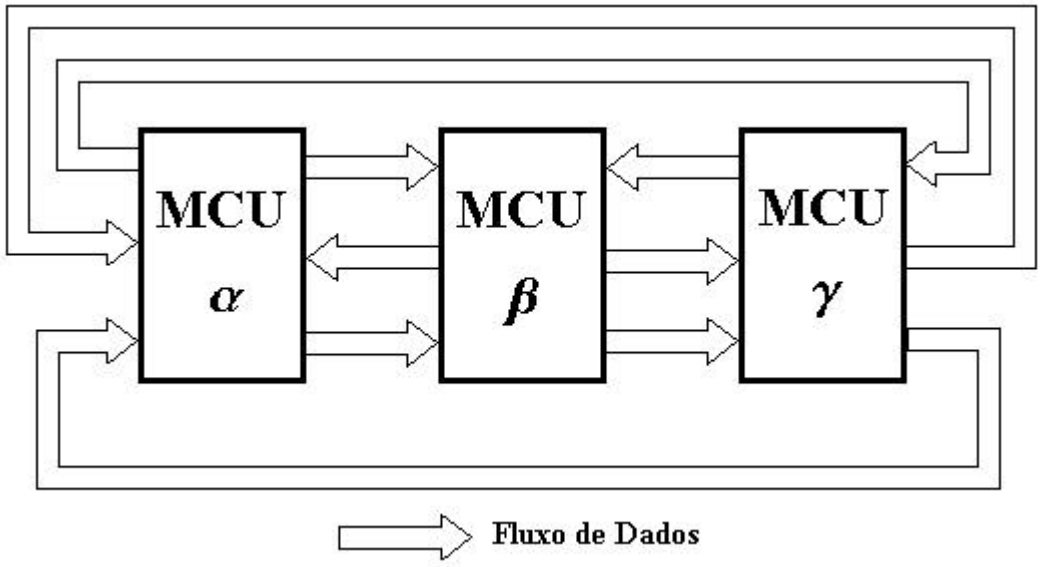


Figura 3.10 – Fluxo de dados na arquitetura TMR.

3) Transmissão de dados: A Figura 3.10 demonstra a comunicação entre MCUs na arquitetura TMR. Cada microcontrolador transmite seu parecer aos demais módulos, conforme a Tabela 3.4.

4) Comparação dos dados: Cada microcontrolador compara seu parecer com o dos outros dois e gera flags internos, que são utilizados no processo de votação.

5) Processo de votação: Com base nos flags gerados, cada MCU consulta uma tabela (Tabela 3.3) para saber se é o vencedor, ou seja, se fará o controle do

sistema. Para qualquer outra combinação não apresentada na Tabela 3.3, nenhum microcontrolador exercerá a função de controle e os sistemas de aquecimento e refrigeração ficarão desligados.

6) Controle pelo módulo vencedor: Mesmo procedimento da arquitetura em anel.

Tabela 3.4 – Transmissão de dados na arquitetura TMR.

Transmissão	MCUa	MCUß	MCU?
1º Ciclo	Transmite	Recebe	Recebe
2º Ciclo	Recebe	Transmite	Recebe
3º Ciclo	Recebe	Recebe	Transmite

3.3.4 Considerações sobre os sistemas

a) Os três microcontroladores operam de forma concorrente, pois, a votação deve ser sobre a mesma versão de dados. Para sincronizá-los, foi usada a técnica de "rendez-vous" (WENLEY, 1983.a), também chamada de "ponto comum de encontro". O ponto de encontro, neste caso, é o ponto no processo onde cada módulo apronta o dado para ser submetido à votação. Neste esquema, o módulo que chega primeiro ao "ponto de encontro", avisa os outros e fica esperando pela chegada dos demais. Após este encontro, todos executam a votação em paralelo (WENLEY, 1983.b).

b) Se um módulo é retirado do sistema, para reparo ou outro motivo, o sistema considera o módulo ausente como um módulo falho.

c) Após a obtenção de todos os resultados (leitura do sensor, interpretação e comparação de dados), o sistema primeiro verifica a presença de falhas para depois exercer a ação de controle (BOTTA; LOPES, 2002.a). Os chamados "testes de último momento", têm a vantagem de testar todas as atividades do sistema (LOMBARDI; RODA, 1982).

3.3.5 A diferença entre o software das arquiteturas implementadas

Sob este tópico, são expostas algumas peculiaridades dos programas desenvolvidos e a diferença existente entre os programas para a arquitetura em anel e TMR.

"Para que um programa seja escrito e funcione corretamente", segundo Souza (2000), "basta que as instruções certas sejam colocadas na ordem correta". Embora isso seja verdade, um programa que não é bem estruturado, mesmo sendo funcional, não pode ser considerado eficiente. De acordo com o próprio Souza (Souza, 2000), "isto porque ele não estará devidamente estruturado e padronizado, dificultando futuras alterações e/ou o entendimento por outros programadores". Assim sendo, houve a preocupação com a obtenção de uma estruturação adequada, separando e identificando nos programas, cada tarefa.

Falando ainda sobre padronização e otimização da programação, não se pode deixar de mencionar os arquivos de definições. Esses arquivos, denominados "Includes" pela Microchip, são arquivos de texto ou código fonte, que podem ser incluídos em um programa, com a finalidade de evitar digitações desnecessárias. A Microchip disponibiliza um arquivo de definições para cada modelo de microcontrolador. Nesses arquivos, encontra-se a definição dos nomes e endereços de cada SFR, além de outras definições necessárias para o uso de um microcontrolador. O arquivo include para o PIC 16F628 (PIC 16F628.INC) pode ser apreciado no Anexo II.

Com o intuito de facilitar ainda mais a compreensão dos programas, praticamente todas as linhas de comando foram comentadas, sendo os comentários encontrados logo após um ponto-e-vírgula (;). Cabe aqui a menção, que os comentários não ocupam espaço em um arquivo executável (.hex), somente no código fonte (.asm). Isso se deve ao fato, de os comentários serem ignorados pelo compilador no processo de geração do arquivo executável.

Uma observação atenta das listagens dos programas, para a arquitetura em anel e TMR, revela que a diferença entre os programas para as diferentes arquiteturas

encontra-se na rotina principal. Como já foi ressaltado, a diferença fundamental entre as arquiteturas, está no método de comunicação entre os módulos do sistema, conforme é possível notar com a ajuda das Figuras 3.9 e 3.10. Além da diferença entre a circulação de dados na arquitetura em anel e o fluxo de dados na TMR, existem os sinalizadores de erro. A arquitetura TMR não utiliza tais sinalizadores, ao passo que a arquitetura em anel emprega-os. Deste modo, para a arquitetura TMR, há apenas uma etapa de troca de informação, onde cada microcontrolador envia aos outros dois, seu parecer referente à temperatura. Após isto, inicia-se a fase de votação. Já na arquitetura em anel, existem duas etapas de troca de informação. Na primeira, cada microcontrolador envia o seu parecer referente à temperatura, somente a um dos outros dois. Com base na comparação do parecer gerado com o recebido, cada MCU gera um código de erro. Segue então a segunda etapa de troca de informação, onde cada MCU envia seu código de erro as outras duas. Somente então, inicia-se a fase de votação.

Neste ponto, é importante lembrar, que nem todos os modelos de microcontroladores PIC possuem uma interface de comunicação serial, como possui o PIC 16F628. O PIC 16F84A, por exemplo, não a possui. Uma vez que, no início, ele foi o modelo utilizado para implementação das arquiteturas tolerantes a falhas descritas neste trabalho, foi necessária a implementação de rotinas para recepção e transmissão de dados. Essas rotinas foram aproveitadas para o PIC 16F628 e estão inseridas nas listagens dos programas. A rotina de recepção elaborada é capaz de reconhecer a falha de um módulo, quando este, depois de decorrido um tempo predeterminado, não transmite dados. Uma outra vantagem do emprego destas rotinas, é que a migração para modelos que não possuem USART pode ser realizada de maneira mais ágil e cômoda.

3.3.6 Peculiaridades do software de cada módulo

Os programas implementados, para cada módulo de uma mesma arquitetura, embora similares, não são idênticos. A diferença fundamental, para ambas as arquiteturas, reside na tabela usada no processo de votação, para garantir que somente um microcontrolador exerça o controle, enquanto os demais aguardam.

A Tabela 3.5 mostra as linhas de comando para MCU γ. Esta MCU exerce o controle em duas situações distintas, a saber:

1- Quando o módulo posterior a ele (Módulo α) apresenta falha;
2- Sempre que os três módulos operam corretamente.

Tabela 3.5 – Linhas de comando do software para MCU g.

MOVF	**FLAGS,W**	**;COPIA VALOR DE FLAGS EM W**
ANDLW	**B'00000111'**	**;MASCARA VALOR, CONSIDERA SOMENTE ATÉ 7**
ADDWF	**PCL,F**	**;PCL = (PCL+W)**
GOTO	**AGUARDA**	**;0**
GOTO	**AGUARDA**	**;1**
GOTO	**AGUARDA**	**;2**
GOTO	**CONTROLA**	**;3**
GOTO	**AGUARDA**	**;4**
GOTO	**AGUARDA**	**;5**
GOTO	**AGUARDA**	**;6**
GOTO	**CONTROLA**	**;7**

Tabela 3.6 – Linhas de comando do software para MCU a e MCU b.

MOVF	**FLAGS,W**	**;COPIA VALOR DE FLAGS EM W**
ANDLW	**B'00000111'**	**;MASCARA VALOR, CONSIDERA SOMENTE ATÉ 7**
ADDWF	**PCL,F**	**;PCL = (PCL+W)**
GOTO	**AGUARDA**	**;0**
GOTO	**AGUARDA**	**;1**
GOTO	**AGUARDA**	**;2**
GOTO	**CONTROLA**	**;3**
GOTO	**AGUARDA**	**;4**
GOTO	**AGUARDA**	**;5**
GOTO	**AGUARDA**	**;6**
GOTO	**AGUARDA**	**;7**

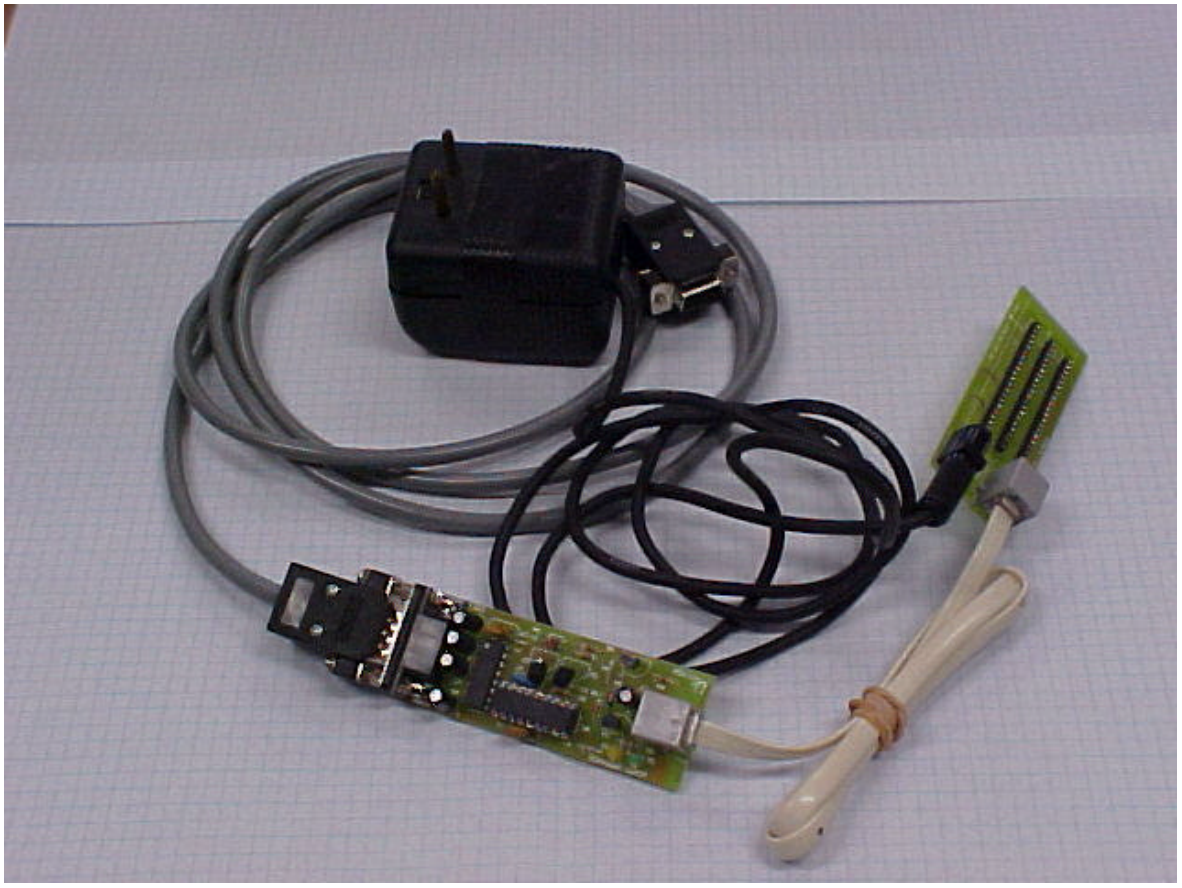
O Módulo α, por outro lado, exerce o controle somente quando o módulo posterior a ele (Módulo β) apresenta falha. Da mesma forma, o Módulo β também só exerce controle sobre o sistema, quando o módulo posterior a ele (Módulo γ) não opera

corretamente. Desta forma, as linhas de comando para estes dois módulos assumem a forma exibida na Tabela 3.6.

3.3.7 Gravação do Microcontrolador

Depois do software pronto e devidamente testado, uma cópia foi gravada em cada microcontrolador, sendo em seguida, realizados ensaios com a utilização de um protótipo.

Para gravação de um programa em um microcontrolador PIC, existem muitas ferramentas diferentes disponíveis no mercado. O projeto de algumas destas ferramentas, pode até mesmo ser obtido gratuitamente através da internet.



Fotografia 3.7 – Gravador μC Flash.

Neste trabalho, a ferramenta usada foi o μC Flash, comercializado pela Mosaico Engenharia (2002). Trata-se de uma plataforma de desenvolvimento de baixo

custo, compatível com o MPLAB. É capaz de gravar toda a gama de microcontroladores da linha Flash da Microchip, possuam estes 40, 28 ou 18 pinos.

O kit do µC Flash, que pode ser visto na Fotografia 3.7, é composto pelos seguintes itens:

- µC Flash – gravador de microcontroladores da linha Flash;
- µC Soc – placa auxiliar para leitura/gravação de MCUs com encapsulamento DIP;
- Cabo de comunicação entre o µC Flash e µC Soc;
- Cabo para comunicação Serial RS-232;
- Fonte de alimentação de 15V_{CC}.

No capítulo seguinte, encontram-se os resultados obtidos por meio de testes e discussões sobre os mesmos.

CAPÍTULO 4

RESULTADOS E DISCUSSÕES

4.1 Considerações iniciais

Antes da consideração da análise propriamente dita, vale lembrar as características compartilhadas por ambas as arquiteturas implementadas:

- A comparação dos dados entre módulos é implementada por programação.
- Cada módulo possui sincronização independente, não existindo, portanto, um gerador de tempo comum, que poderia constituir um ponto comum de falhas.
- Podem tolerar falhas em um módulo.
- Somente um dos módulos é o responsável pelo controle.
- Os sistemas são modulares e podem operar sem um dos módulos ativado, o que facilita a manutenção e aumenta a disponibilidade dos mesmos.
- É considerado como módulo falho, o que transmite dados incorretos ou é incapaz de enviar dados.

Tendo em mente tais características, o desempenho das duas arquiteturas foi avaliado, submetendo o protótipo construído, a testes de operação e simulação de falhas.

Testes da sincronização mostraram a necessidade de se usar cristais osciladores externos, pois o uso dos osciladores internos, próprios de cada microcontrolador, levava a perda de sincronização e travamento dos sistemas. Isto devido ao oscilador interno do componente ser do tipo RC, menos estável e preciso que um cristal oscilador externo. Cabe ressaltar, que a sincronização é mais crítica na arquitetura em anel, sendo eventualmente necessário no início da operação, o reset do sistema, para que os três módulos comecem a trabalhar concorrentemente.

Resolvido o problema da sincronização, o sistema foi submetido a uma série de testes. Antes da abordagem de tais testes, porém, convém esclarecer o funcionamento do módulo comparador analógico do PIC 16F628, na gestão da temperatura.

4.2 Utilização do módulo comparador na gestão da temperatura

Para esta aplicação, o módulo comparador analógico do PIC 16F628 foi configurado para o modo 1, com o bit CIS em nível lógico baixo (“0”), se apresentando portanto, conforme mostra a Figura 4.1.

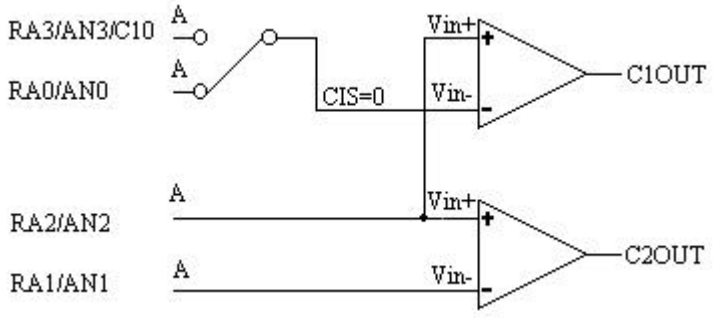


Figura 4.1 – Módulo comparador analógico no modo 1 e com o bit CIS = 0.

No pino RA2/AN2, do PIC 16F628, é aplicado o sinal referente à temperatura ambiente, diretamente pelo terminal Vout do sensor LM35DZ

(NATIONAL,2000), enquanto nos pinos RA0/AN0 e RA1/AN1, são aplicados os sinais referentes aos limites máximo e mínimo, respectivamente (Figura 4.2). Conforme mostra o diagrama da Figura 3.5, os limites máximo e mínimo podem ser facilmente ajustados através dos potenciômetros R1 e R2. De acordo com os valores de saída C1OUT e C2OUT, dos comparadores, será feito o controle dos sistemas de refrigeração e aquecimento.

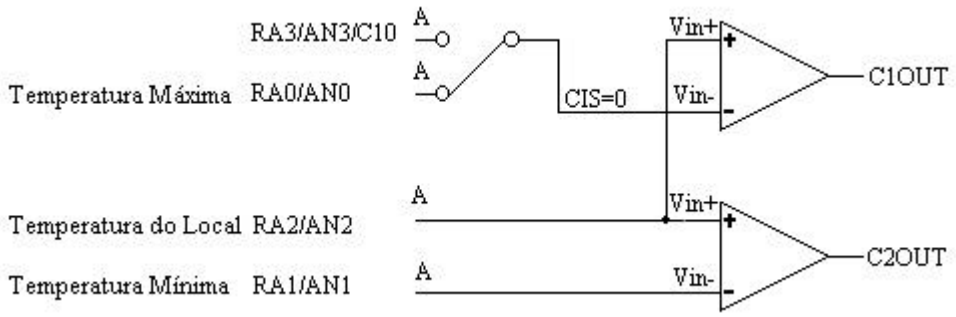


Figura 4.2 – Utilização do módulo comparador no controle da temperatura.

A temperatura, obviamente, poderá se encontrar em uma das seguintes condições:

1- Entre os limites desejados.
2- Acima do limite desejado.
3- Abaixo do limite desejado.

Tabela 4.1 – Relação entre temperatura e sistemas de aquecimento e refrigeração

Temperatura	Relação entre VRA0, VRA1 e VRA2.	Implicação
Entre os limites desejados.	VRA0 > VRA2 > VRA1	Desligar ou manter desligados os sistemas de refrigeração e aquecimento.
Acima do limite desejado.	VRA2 > VRA0 > VRA1	Ligar ou manter ligado sistema de refrigeração.
Abaixo do limite desejado.	VRA0 > VRA1 > VRA2	Ligar ou manter ligado sistema de aquecimento.

A Tabela 4.1 estabelece a relação entre a temperatura do ambiente e o controle dos sistemas de aquecimento e refrigeração. Por outro lado, a Tabela 4.2, exibe as combinações possíveis de C1OUT e C2OUT e o significado destas combinações, para os sistemas de gerenciamento da temperatura.

Tabela 4.2 – Combinações possíveis entre C1OUT e C2OUT.

C1OUT	C2OUT	REFRIGERAÇÃO	AQUECIMENTO	SITUAÇÃO
0	0	Desligado	Ligado	Temperatura Baixa
0	1	Desligado	Desligado	Temperatura Ideal
1	0	Ligado	Ligado	Proibida
1	1	Ligado	Desligado	Temperatura Alta

Nota: Para que a situação denominada proibida ocorresse, a tensão de referência (temperatura do ambiente) teria de ser ao mesmo tempo, maior que o limite máximo e menor que o limite mínimo.

4.3 Análise

O pino 1 (porta RA2/AN2 – Figura 4.2) dos microcontroladores foi destinada a receber informação do sensor de temperatura. Creditando sinal a esta porta, foi possível simular variações de temperatura. Ambas as arquiteturas responderam as simulações atuando adequadamente sobre os sistemas de aquecimento e refrigeração. A Tabela 4.3 apresenta a resposta das arquiteturas para as diferentes condições.

Tabela 4.3 – Controle da temperatura.

Condição	**Resposta do Sistema**
Temperatura acima da desejada.	Ligar ou manter ligado sistema de refrigeração.
Temperatura abaixo da desejada.	Ligar ou manter ligado sistema de aquecimento.
Temperatura dentro dos limites preestabelecidos.	Desligar ou manter desligado os sistemas de refrigeração e aquecimento.

Em seguida os sensores foram submetidos a variações reais de temperatura e as arquiteturas implementadas comportaram-se corretamente (de acordo com a Tabela 4.3).

Simulações de falhas foram realizadas de duas formas diferentes. Em uma delas, foram introduzidos erros nas leituras de um dos sensores. Em outro teste, um dos módulos foi retirado completamente, simulando falha total dos circuitos. Em ambos os

casos, os sistemas com arquitetura em anel e TMR continuaram a operar corretamente e de forma transparente à aplicação. As Figuras 4.4 e 4.5, registram momentos das etapas de realização de testes e simulações de falhas, com as arquiteturas implementadas.

Foi também avaliado o aumento de tempo de processamento (LOMBARDI; RODA, 1982) dos sistemas tolerantes a falhas, em relação a um sistema simplex. O MPLAB possui uma ferramenta denominada Stopwatch (Figura 4.3), que conta os ciclos de máquina e calcula o tempo baseado na freqüência do oscilador que foi configurado para uso. Um ciclo de máquina equivale a ¼ do sinal de clock externo utilizado. Portanto, empregando um cristal oscilador externo de 4MHz, o clock interno é de 1MHz e conseqüentemente um ciclo de máquina dura 1µs. Com o Stopwatch, foi mensurado o tempo de execução dos programas e de suas rotinas. As rotinas de transmissão/recepção (180µs) são as maiores responsáveis pelo aumento de tempo de processamento neste sistema, em relação a um sistema simplex. Caso haja interrupção no canal de comunicação entre dois módulos, ou caso um módulo seja incapaz de enviar dados, isso implicará num aumento de tempo um pouco maior. Isto se deve ao tempo que os módulos em bom funcionamento esperam pelos dados do módulo com defeito. O tempo de espera é determinado por programação, podendo ser ajustado para obtenção de um valor ótimo (FISCHER, 1990). Para as arquiteturas implementadas, o tempo de espera máximo foi configurado para ser de 250 e 500µs, para as arquiteturas TMR e anel, respectivamente. O incremento de tempo de processamento para a arquitetura TMR é menor que para a arquitetura em anel, porque a TMR não necessita transmitir/receber sinalizadores de erro. Se houver necessidade de um incremento de tempo menor, dos sistemas tolerantes a falhas aqui expostos em relação a um sistema simplex, esse problema pode ser facilmente resolvido. Uma maneira é utilizar uma freqüência de clock maior. Outra é empregar microcontroladores com um maior número de pinos e transmitir os dados de forma paralela ao invés de serial. Essa transmissão levaria menos de 10µs.

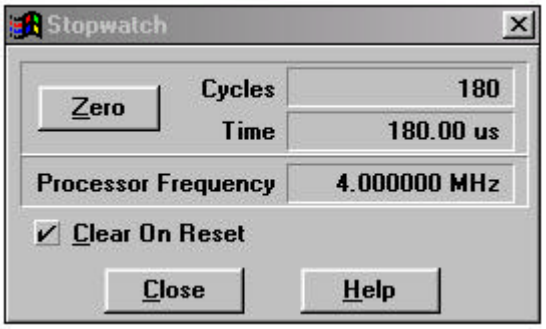
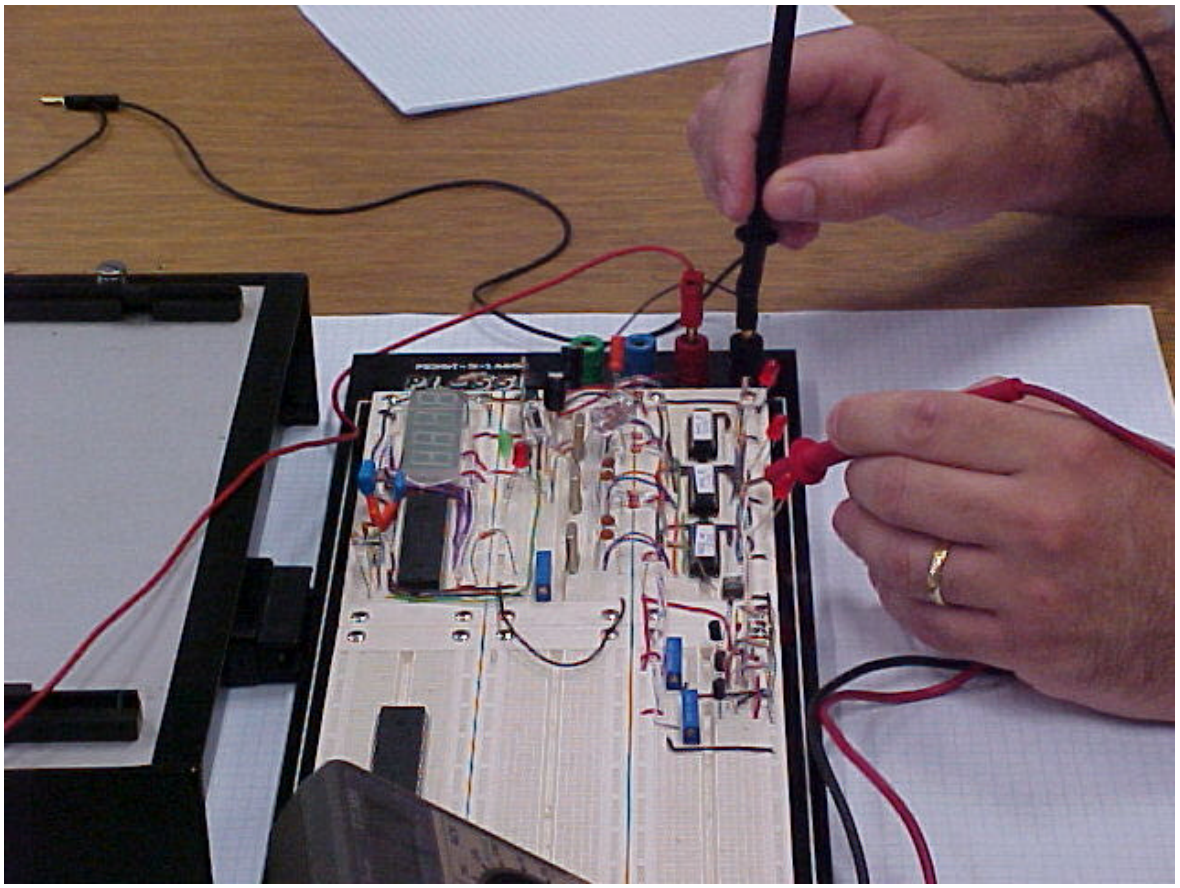
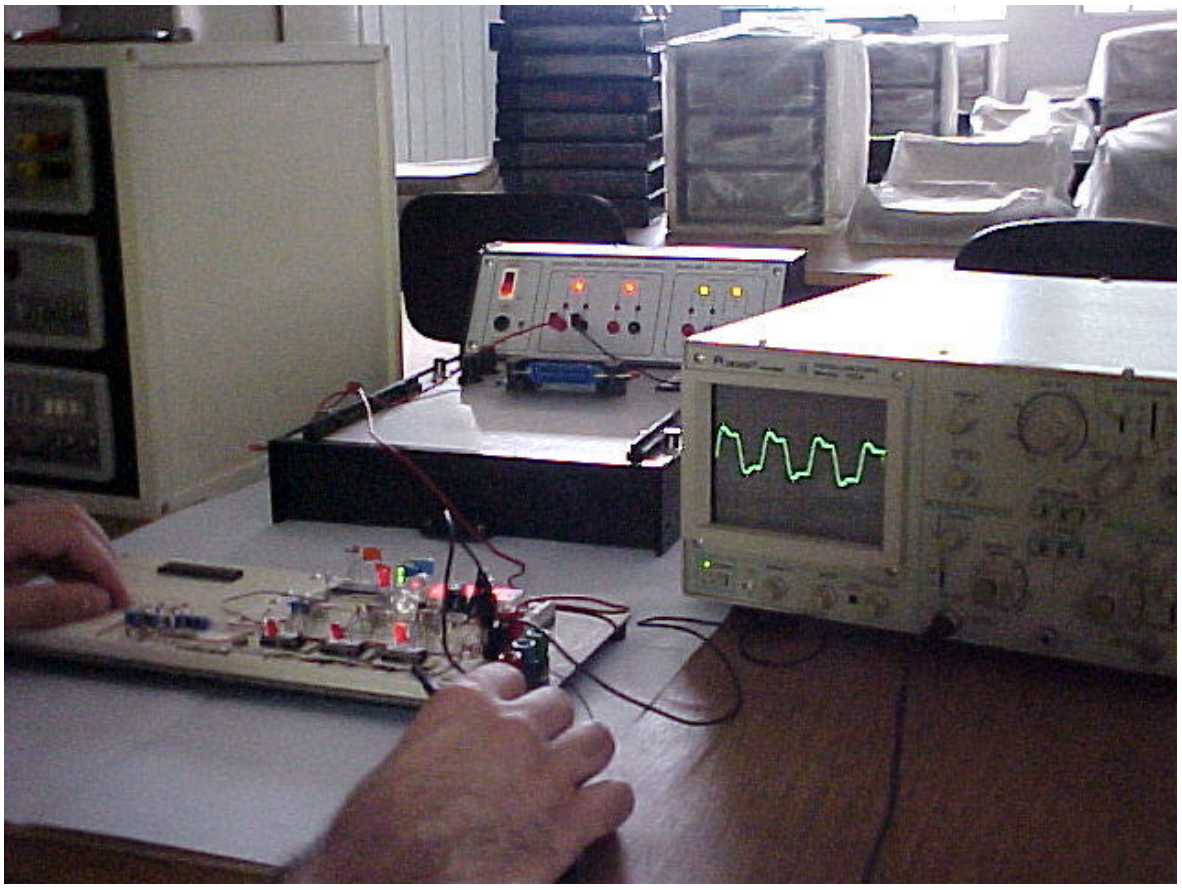


Figura 4.3 – Contagem de tempo com o MPLAB.



Fotografia 4.1 – Etapa de realização de testes.



Fotografia 4.2 – Etapa de simulação de falhas.

Os tempos medidos, no entanto, mostram que as arquiteturas implementadas são adequadas para controle em tempo real, se comparadas com outros projetos desenvolvidos. O sistema “SIFT”, para controle de aeronaves em tempo real, gasta tempo na rotina de sincronização e calculando o vetor de consistência. Embora o tempo gasto no processo de sincronização, não seja indicado nas publicações (WENSLEY; et al., 1978) (BUTLER, et al., 1985) sobre o “SIFT”, o sistema August-série 300 (WENSLEY; HARCLERODE, 1982) (WENSLEY, 1983.b), baseado no “SIFT”, indica 200μs (WENSLEY, 1981) como tempo de espera para que os módulos fiquem sincronizados. O August-300 emprega um microprocessador de 16 bits e relógio de 5MHz. As publicações (WENSLEY, 1981) (WENSLEY, 1982) (WENSLEY, 1983.b), no entanto, não citam o tempo gasto na rotina de sincronização. O primeiro sistema em anel estudado (RODA, 1983), usando microprocessador de 8 bits e relógio de 2MHz, gastava 45μs com a transmissão e 555μs para completar o reconhecimento e ciclo de espera.

Quanto ao processo de sincronização, o que foi utilizado nas arquiteturas descritas neste trabalho, é extremamente simples e flexível se comparado com outros processos. O processo utilizado permite que os módulos executem suas tarefas em tempos levemente diferentes. Isto proporciona a vantagem de tornar os sistemas menos sujeitos à falhas correlatas.

A maior parte dos sistemas descritos na literatura pesquisada, como por exemplo, os sistemas “SIFT” e August-300, utiliza-se de referência de tempo (uso de um único relógio), para sincronização das unidades de processamento, que pode constituir-se em um ponto comum de falhas (hardcore). Logo, o método de sincronização empregado, apresenta-se como uma boa alternativa em relação aos relógios, empregados para sincronização de sistemas NMR.

O sistema TMR proposto por Chande, não utiliza relógio no processo de sincronização, empregando também um ponto comum de encontro, através de sinalizadores (CHANDE, et al., 1989). O sistema de Chande, no entanto, usa uma memória global. Cada módulo do sistema tem que requisitar o acesso à memória, esperar que o controlador conceda o acesso, para somente então ler os sinalizadores de sincronização dos outros módulos. Só um módulo por vez escreve na memória. Esse processo é mais

complexo que o utilizado nas arquiteturas descritas nesse trabalho. Além disso, a memória global constitui-se um ponto vulnerável do sistema, mesmo que construída com circuitos de alta confiabilidade.

Uma grande vantagem de se usar microcontroladores, quando possível, em sistemas tolerantes a falhas, é que estes possuem em seu interior vários periféricos, reduzindo ao mínimo o número de componentes externos necessários para implementação dos circuitos. Um exemplo disso é a arquitetura baseada no sistema Mestre/Escravo (Máster/Slave), desenvolvida para NASA (National Aeronautics and Space Administration – Administração Nacional de Aeronáutica e Espaço), por Rennels e Hwang (2001). Essa arquitetura (Rennels; Hwang, 2001) foi construída com base no microcontrolador 87C196CA, fabricado pela Intel Corporation. Em um sistema Mestre/Escravo, porém, se a unidade Mestra falha, todo o sistema fica comprometido. Diferentemente, as arquiteturas TMR e anel, baseadas em microcontroladores, que são apresentadas neste trabalho, podem tolerar defeito em qualquer dos módulos, por não haver um módulo Mestre.

As arquiteturas, apresentadas nesse trabalho, podem ser consideradas excelentes opções, para agregação de confiabilidade e disponibilidade, a sistemas de controle embarcados. Ao contrário da arquitetura TMR clássica, não necessitam de um elemento votador, uma vez que o processo de votação, é implementado por programação. Diferente do sistema proposto por Chande (1989), que utiliza uma memória global, não necessitam de nenhum hardware especial para controle da redundância. Dispensam também, o uso de uma referência de tempo comum, para sincronização das unidades de processamento, utilizada em inúmeros sistemas, como por exemplo, o August-300. Em relação ao sistema desenvolvido por Rennels e Hwang (2001), as arquiteturas TMR e Anel, são mais robustas que a Mestre/Escravo.

CAPÍTULO 5

CONCLUSÕES E PERSPECTIVAS FUTURAS

Com o contínuo avanço de novas tecnologias, a dependência humana a tais, torna-se cada vez mais inexorável. Isto, por si só, serve como um agente motivador para as incessantes pesquisas no ramo da tolerância à falhas. É extremamente desejável a obtenção de sistemas de controle e automação, cada vez mais seguros e confiáveis.

Os avanços da microeletrônica (HONDA; PAIXÃO, 2000) possibilitaram a incorporação de um processador, memória, portas de entrada e saída e recursos auxiliares em uma única pastilha de silício (em um único chip). Esses dispositivos eletrônicos, os microcontroladores, cada vez mais potentes e baratos, são destinados com freqüência crescente ao controle de processos. Isto tem motivado a construção de arquiteturas tolerantes a falhas para sistemas embutidos, como a apresentada por Rennels e Hwang (2001).

Neste trabalho, discutiu-se a implementação de duas arquiteturas tolerantes a falhas baseadas em microcontroladores.

A arquitetura em anel, segundo Fischer (1990), trata-se de um “sistema que abre uma perspectiva diferente na pesquisa em arquiteturas tolerantes à falhas”. Isso,

por ser “uma arquitetura flexível, que facilita o procedimento de sincronização e a comunicação entre módulos, para o mascaramento, detecção e diagnose de falhas”.

Uma arquitetura TMR clássica prevê um elemento votador, (HAYES, 1988) que se constituí num ponto vulnerável do sistema. Algumas arquiteturas TMR que dispensam o elemento votador, por sua vez, utilizam-se de circuitos especiais para controle da redundância (CHANDE; et al., 1989). Na arquitetura TMR construída, foi usada estratégia de votação por software e não há necessidade de qualquer circuito especial para controle da redundância, uma vez que o controle é feito por protocolos implementados por programação.

De acordo com Wensley (1983.a; 1983.b), a sincronização de um módulo com o sistema, após o reparo, é um desafio para o projeto de sistemas tolerantes a falhas. Com base em dados empíricos, obtidos por meio de ensaios laboratoriais, é possível afirmar que após a recolocação de um módulo no sistema, a sincronização ocorre em ambas as arquiteturas, sem interferência na tarefa de controle. Isso ocorre, essencialmente, porque as arquiteturas implementadas podem ser dedicadas a aplicações específicas (como no controle da temperatura, por exemplo) e não usam uma base de tempo comum para sincronizar os módulos, evitando-se um ponto comum de falhas e com a vantagem de possibilitar que as tarefas sejam executadas em cada módulo, em tempos ligeiramente diferentes (BOTTA; LOPES, 2002.a).

Logo, as arquiteturas tolerantes a falhas implementadas, podem ser consideradas como opções promissoras, para uso no controle dedicado de processos, em que se requeira altos índices de confiabilidade e disponibilidade de operação.

As arquiteturas, aqui apresentadas, estão projetadas para propósito geral. Neste trabalho, foram designadas para o controle da temperatura de um ambiente, que por vezes, pode ser uma tarefa crítica, como em estufas de maternidade. Ilustra-se com isso, que os sistemas desenvolvidos podem ser facilmente modificados para diferentes aplicações, especialmente, levando-se em conta que a migração de um modelo de microcontrolador da família PIC, para outro, não é tarefa difícil. Projetos de instrumentação tolerante a falhas, biotecnologia ou robótica, podem ser realizados com base nas arquiteturas apresentadas. Estas podem também ser exploradas com outras linhas

de microcontroladores, como AVR (ATMEL, 2002.a; 2002.b), MCS 51/251 (INTEL, 2002), COP8 (NATIONAL, 2002) ou outra disponível no mercado. Além disso, pode-se estudar o desenvolvimento de outras arquiteturas microcontroladas tolerantes a falhas, como, por exemplo, arquiteturas com redundância dinâmica por circuitos ou com redundância híbrida (combinação da redundância estática com a dinâmica), que venham a propiciar maior confiabilidade e disponibilidade a sistemas embutidos.

REFERÊNCIAS BIBLIOGRÁFICAS

Ahuja, M.; Mishra, S. Units of computation in fault-tolerant distributed systems. **Journal of Parallel and Distributed Computing**, Los Angeles, no. 40, p. 194-209, 1997.

Anisimov, A. V.; Teplov, V. Y.; Silkin, N. I. System of thermostatic control on peltier thermopiles and microprocessor control for a portable NMR relaxometer. **Journal of Magnetic Resonance**, Kazan, no. 154, p. 176-180, Jan. 2002.

ATMEL CORPORATION. **8-bit AVR Microcontroller with 1K Byte of In-System Programmable Flash – AT90S1200**. [s.l.], 2002.a. Disponível em: **< http://www.atmel.com/atmel/acrobat/doc0839.pdf>** Acesso em: 07.nov.2002.

ATMEL CORPORATION. **8-bit AVR Microcontroller with 2K Byte of In-System Programmable Flash – AT90S2313**. [s.l.], 2002.b. Disponível em: **<http://www.atmel.com/atmel/postscript/first_page/doc0838a.gif>** Acesso em: 07.nov.2002.

ATMEL CORPORATION. **8051-Architecture – Data Sheets**. [s.l.], 2002.c. Disponível em: **< http://www.atmel.com/atmel/products/prod71.htm>** Acesso em: 07.nov.2002.

Botta, G. F.; Lopes, D. P. M. Controle da temperatura com um sistema tolerante a falhas usando microcontroladores. In: CONGRESSO TEMÁTICO DE DINÂMICA, CONTROLE E APLICAÇÕES DA SOCIEDADE BRASILEIRA DE MATEMÁTICA APLICADA E COMPUTACIONAL, 1., 2002.a, São José do Rio Preto, **Anais do 1º Congresso Temático de Dinâmica, Controle e Aplicações da Sociedade Brasileira de Matemática Aplicada e Computacional**. São José do Rio Preto: SBMAC, 2002. p. 899-903.

Botta, G. F.; Lopes, D. P. M. Implementação de duas arquiteturas microcontroladas tolerantes a falhas. In: ENCONTRO DE ATIVIDADES CIENTÍFICAS, 5., 2002, Londrina. **Resumos...** Londrina: SBPC, 2002.b. p. 15.

Brookshear, J. G. **Ciência da computação:** uma visão abrangente. 5. ed. Porto Alegre: Bookman. 2000. 499 p.

Butler, R.W.; Palumbo, D. L.; Johnson, S. C. Application of a clock synchronization validation methodology to the SIFT computer system. In: SYMPOSIUM ON FAULT-TOLERANT COMPUTING, 15, 1985, Arbor. **Proceedings…** Arbor, 1985. p. 194-199.

Chande, P.K.; Ramani, A.K.; Sharma, P.C. Modular TMR multiprocessor system, **IEEE Transactions on Industrial Electronics**, v. 36, n. 1, p. 34-41, Feb. 1989.

Cheng, S. T.; Chen, C. M.; Tripathi, S. K. A fault-tolerance model for multiprocessor real-time systems. **Journal of Computer and System Sciences**, no. 61, p. 457-477, 2000.

Clements, Alan. **The principles of computer hardware**. 2. ed. Malta: Oxford University Press. 1991. 682 p.

Coulouris, G. F.; Dollimore J. **Distributed systems:** concepts and design. 1. ed. Great Britain: Addison-Wesley Publishing Company. 1988. 366 p.

Dishon, Y.; Georgiou, C. J. A highly available storage system using the checksum method. In: FAULT-TOLERANT COMPUTING SYMPOSIUM, 17., 1987, Pittsburgh, **Digest of Paper...** Pittsburgh: The Computer Society of the IEEE, 1987. p. 176-181.

Fayman, J. A.; Rivlin, E.; Mossé, D. FT-AVS: a fault-tolerant architecture for real-time active vision. **Real-Time Imaging**, no. 4, p. 144-157, 1998.

Filho, H.S.; Nunes, C.N. Análise de duas técnicas de projeto de arquiteturas tolerantes a falhas. In: CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, 15., 1995, Canela, **Anais do 6º Simpósio de Computadores Tolerantes a Falhas**. Canela: SBC, 1995. p. 49-62.

Fischer, D.P.M. **Desenvolvimento de um sistema de computação tolerante a falhas com arquitetura em anel**. 1990. 171f. Tese (Doutorado) – Instituto de Física e Química de São Carlos, Universidade de São Paulo, São Carlos, 1990.

Fraga, J.; Lemos, R. Conceituação e terminologia em sistemas informáticos tolerantes a falhas e intrusões. SIMPÓSIO EM SISTEMAS DE COMPUTADORES TOLERANTES A FALHAS, 2., 1987, [s.l.]. **Anais do 2º Simpósio em Sistemas de Computadores Tolerantes a Falhas**, 1987. p. 177-188.

George, D. E.; Salari, E. Real-time pitch extraction of voiced speech. **Journal of Network and Computer Applications**, no. 20, p. 379-387, 1997.

Handler, W.E. A Multiprocessor Working as a Fault-Tolerant Cellular Automaton. **Computing**, no. 48, p. 5-20, 1992.

Hayes, J. P. **Computer architecture and organization**. 2. ed. Singapore: McGraw-Hill. 1988. 702 p.

Honda, R.; Paixão, R. R. **Processadores Intel**. 1. ed. São Paulo: Érica, 2000. 176 p.

Johnson, B.W. Fault Tolerant Microprocessor based Systems. **IEEE Micro**, p. 6-21, 1984.

INTEL CORPORATION. **MCS 51/251 Microcontrollers**. [s.l.], 2002. Disponível em: **<http://www.intel.com/design/MCS51/docs_mcs51.htm#Datasheets>**. Acesso em: 08.nov.2002.

Kimura, Y.; Hasegawa, K.; Sekiguchi, A. Microprocessor-Based System for Processing Redundant Instrumentation Signals, **IEEE Transactions on Industrial Electronics and Control Instrumentation**, v. 27, no. 3, p. 218-222, Aug. 1980.

Laprie, J. C. Dependable computing and fault tolerance: concepts e terminology. In: ANNUAL INTERNATIONAL SYMPOSIUM ON FAULT-TOLERANT COMPUTING, 15., Arbor. **Proceedings...** New York, IEEE, 1985. p. 2-11.

Laprie, J. C.; Arlat J.; Béounes, C.; Kanoun K. Definition and analysis of hardware – and software – faut-tolerant architectures. **IEEE Computer**, p. 39-51, July 1990.

Lombardi, F.; Roda, V. O. Software implemented fault tolerance – a methodology. **Microeletronics and Reliability**, v. 22, no. 3, p. 873-885, 1982.

Mahesh, K.; Manimaran, G.; Murthy, S.R. Scheduling algorithms with fault detection and location capabilities for real-time multiprocessor systems. **Journal of Parallel and Distributed Computing**, no. 51, p. 136-150, 1998.

MICROCHIP TECHNOLY INC. **PICmicro™ Mid-Range MCU Family Reference Manual**. Chandler, Arizona, 1997. Disponível em: < **http://www.microchip.com/download/lit/suppdoc/refernce/midrange/33023a.pdf**>. Acesso em 08.nov.2002.

MICROCHIP TECHNOLY INC. **PIC16F62X:** Flash-Based 8-Bit CMOS Microcontrollers. Chandler, Arizona, 1999. Disponível em: **<http://www.microchip.com/download/lit/pline/picmicro/families/16c62x/40300b.pdf**>. Acesso em 07.nov.2002.

MICROCHIP TECHNOLY INC. **MPLAB IDE, Simulator, Editor, User's Guide**. Chandler, Arizona, 2001.a. Disponível em: **<http://www.microchip.com/download/tools/picmicro/devenv/manual/51025e.pdf**>. Acesso em 07.nov.2002.

MICROCHIP TECHNOLY INC. **PIC16F84A Data Sheet:** 18-pin enhanced FLASH/EEPROM 8-bit microcontroller. Chandler, Arizona, 2001.b. Disponível em: < **http://www.microchip.com/download/lit/pline/picmicro/families/16f8x/35007b.pdf**>. Acesso em 07.nov.2002.

MICROCHIP TECHNOLY INC. **PIC16F62X:** 16F62X EEPROM Memory Programming Specification. Chandler, Arizona, 2002.a. Disponível em: < **http://www.microchip.com/download/lit/suppdoc/specs/30034d.pdf**>. Acesso em 07.nov.2002.

MICROCHIP TECHNOLY INC. **PIC16F62X:** Silicon/Data Sheet Errata. Chandler, Arizona, 2002.b. Disponível em: **<http://www.microchip.com/download/lit/suppdoc/errata/80073f.pdf**>. Acesso em 08.nov.2002.

MICROCHIP TECHNOLY INC. **SQTP Specification for PIC16/17**. Chandler, Arizona, 2002.c. Disponível em: < **http://www.microchip.com/download/lit/suppdoc/specs/30154f.pdf**>. Acesso em 07.nov.2002.

MOSAICO ENGENHARIA ELETRÔNICA S/C LTDA. **Mosaico Engenharia**. São Paulo, 2002. Disponível em: <www.mosaico-eng.com.br>. Acesso em: 15.nov.2002.

Merritt, B. Low-cost digital thermometer uses single-chip microcontroller. **Eletronic Desing**, p. 62-64, Feb. 2002.

NATIONAL SEMICONDUCTOR CORPORATION. **LM35:** precision centigrade temperature sensors. [s.l.], 2000. Disponível em: **<http://www.national.com/ads-cgi/viewer.pl/ds/LM/LM35.pdf**>. Acesso em 13.maio.2002.

NATIONAL SEMICONDUCTOR CORPORATION. **COP8AME9/COP8ANE9:** 8-Bit CMOS FLASH Microcontroller with 8K Memory, Dual Op Amps, Virtual EEPROM, Temperature Sensor, 10-Bit A/D and Brownout Reset. [s.l.], 2002. Disponível em: < **http://www.national.com/ads-cgi/viewer.pl/ds/CO/COP8AME9.pdf**>. Acesso em 08.nov.2002.

Nicolosi, D. E. C. **Microcontrolador 8051 Detalhado**. 2. ed. São Paulo: Érica, 2000. 234 p.

Ortega, C.; Tyrrell, A. Biologically inspired fault-tolerant architecture for real-time control applications, **Control Engineering Practice**, no. 7, p. 673-678, 1999.

Paula Jr., A. R. Aspectos de tolerância a defeitos do sistema de computação do primeiro microssatélite de aplicações científicas do INPE. In: CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, 15., 1995, Canela, **Anais do 6º Simpósio de Computadores Tolerantes a Falhas**. Canela: SBC, 1995. p. 63-75.

Pereira, Fábio. **Microcontroladores PIC:** técnicas avançadas. 1. ed. São Paulo: Érica. 2002. 358 p.

Pontarelli, S.; Cardarilli, G.C.; Malvoni, A.; Ottavi, M.; Salsano, A. System-on-Chip Oriented Fault-Tolerant Sequential Systems Implementation Methodology, 1., 2001, San Francisco. In: IEEE INTERNATIONAL SYMPOSIUM ON DEFECT AND FAULT TOLERANCE IN VLSI SYSTEMS, 2001, San Francisco. **Proceedings…** San Francisco: The Institute of Electrical and Electronics Engineers, Inc., 2001. p. 455-460.

Rennels, D. A.; Hwang, R. Recovery in fault-tolerant distributed microcontrollers. In: THE INTERNATIONAL CONFERENCE ON DEPENDABLE SYSTEM AND NETWORKS, 1., 2001, Goteborg. **Proceedings...** Goteborg: The Institute of Electrical and Electronics Engineers, Inc., 2001. p. 475-480.

Roda, V.O. **Techniques for Hight Integrity Intelligent Instrumentation**. 1983. Tese (Doutorado) – Department of Eletronic and Eletrical Engineering, University College London, London, 1983.

Roda, V. O.; Lin, T. T. Y. On the effect of spare positioning on the reconfigurability of two-dimensional processors arrayas. In: CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, 15., 1995, Canela, **Anais do 6º Simpósio de Computadores Tolerantes a Falhas**. Canela: SBC, 1995. p. 77-90.

Schlichting, R. D. A technique for estimating performance of fault-tolerant programs. In: SIMPOSIUM ON REALIABILITY IN DISTRIBUTED SOFTWARE AND DATABASE SYSTEMS, 4., 1984, Silver Spring, **Proceedings...** Silver Spring: IEEE Computer Society Press, 1984, p. 62-74.

Schunk, L. M.; Luppi, A. **Microcontroladores AVR: teoria e aplicações práticas**. 1. ed. São Paulo: Érica, 2001. 180 p.

Shen, H. **Fault-tolerant multicast with traffic-balancing in hipercubes**. [Beijing], 1996. Disponível em: **<http://computer.org/proceedings/ispan/7460/74600415abs.htm>** Acesso em: 07.nov.2002.

Shinghi, N. M.; Shriniwas, I; Antony, T. A. Combinatorial techniques and objects in computer science: fault-tolerance and other interesting applications. **Academic Press Limited**, no. 17, p. 97-111, 1996.

Siewiorek, D.P.; Swarz, R.S. **The Theory and practice of reliable system design**. 1. ed. Bedford,MA: Digital Press, 1982. 772 p.

Siewiorek, D.P.; Swarz, R.S. **Reliable computer systems**. 2. ed. Bedford,MA: Digital Press. 1992. 908 p.

Silva Jr., D. C.; Coelho Jr., C. J. N.; Fernandes, A. O.; Mata, J. M. **Conexão de sistemas embutidos a internet**. Belo Horizonte, 2002. Disponível em: **<http://www.lecom.dcc.ufmg.br/~dgns/pubs/1999/ein99.pdf>** Acesso em: 15.nov.2002.

Silva Jr., V. P. **Microcontroladores PIC: teoria e prática**. 1. ed. São Paulo: V. P. Silva Júnior, 1997. 140 p.

Souza, D.J. **Desbravando o PIC**. 1. ed. São Paulo: Érica, 2000. 202 p.

Vaglica, J. J.; Gilmour P. S. How to select a microcontroller. **IEEE Spectrum**, p. 106-109, Nov. 1990.

Weber, T. S. **Tolerância a falhas:** conceitos e exemplos. Porto Alegre, 2002.a. Disponível em:
<**http://www.inf.ufrgs.br/~taisy/disciplinas/textos/ConceitosDependabilidade.PDF**>
Acesso em: 14.nov.2002.

Weber, T. S. **Arquiteturas tolerantes a falhas**. Porto Alegre, 2002.b. Disponível em: <**http://www.inf.ufrgs.br/~taisy/disciplinas/textos/TFarqs.PDF**> Acesso em: 14.nov.2002.

Wensley, J. H.; Harclerode, C. S. Programmable control of a chemical reactor using a fault tolerant computer. **IEEE Transactions on industrial eletronics**, v. IE-29, no. 4, p. 258-264, Nov. 1982.

Wensley, J. H.; Lamport, L.; Goldberg, J.; Green, M.W.; Levitt, K.N.; Melliar-Smith, P.M.; Shostak, R.E.; Weristock, C. B. SIFT: the design and analisys of a fault-tolerant computer for aircraft control. **Proc. IEEE**, v. 66, no. 4, p. 1240-1255, Oct. 1978.

Wensley, J. H. Fault-tolerant computers ensure reliable industrial controls. **Eletronic Design**, p. 129-135, Jun. 1981.

Wensley, J. H.; Harclerod, C. S. Programable control of a chemical reactor using a fault tolerant computer. **IEEE Transactions on industrial eletronics**. v. IE-29, no. 4, p. 258-264, Nov. 1982.

Wensley, J.H. An operating system for a TMR fault-tolerant system, **13º International Symposium on Fault-Tolerant Computing**. Milão, 1983a, p. 452-455.

Wensley, J.H. Fault-tolerant computers: industrial-control system does things in threes for safety. **Eletronics**, v. 56, no. 2, p. 98-102, Jan. 1983b.

ANEXO I

MCU PIC 16F628

1 Principais Características

Para aplicação no controle da temperatura de um ambiente, as arquiteturas tolerantes a falhas, descritas neste trabalho, foram implementadas usando microcontroladores PIC 16F628 (MICROCHIP, 1999).

A família PIC 16F62X (Tabela I.1), da qual o PIC 16F628 é membro, pertence à gama média (Mid-range) de microcontroladores da Microchip (MICROCHIP, 1997) (PEREIRA, 2002), cujas principais características são:

- Baixo custo;
- Alto desempenho;
- Tecnologia CMOS;
- Memória de dados de 8 bits;
- Diversos periféricos internos;
- Memória de programa tipo FLASH;
- Conjunto de 35 instruções de 14 bits;
- Possibilidade de gravação no próprio circuito de aplicação.

Tabela I.1 – Família PIC 16F62X.

		PIC16F627	PIC16F628	PIC16LF627	PIC16LF628
Clock	Freqüência máxima de operação (MHz)	20	20	20	20
Memória	Memória de programa FLASH (palavras)	1024	2048	1024	2048
	Memória de dados RAM (bytes)	224	224	224	224
	Memória de dados EEPROM (bytes)	128	128	128	128
Periféricos	Módulos Temporizadores	TMRO, TMR1,TMR2	TMRO, TMR1,TMR2	TMRO, TMR1,TMR2	TMRO, TMR1,TMR2
	Comparadores	2	2	2	2
	Módulo de Captura/Comparação/PWM	1	1	1	1
	Comunicação Serial	USART	USART	USART	USART
Características	Tensão de referência interna	Sim	Sim	Sim	Sim
	Fontes de interrupção	10	10	10	10
	Pinos de I/O	16	16	16	16
	Faixa de tensão (Volts)	3.0-5.5	3.0-5.5	2.0-5.5	2.0-5.5
	Detecção de Brown-out	Sim	Sim	Sim	Sim
	Encapsulamento	18-Pin DIP, SOIC; 20-Pin SSOP	18-Pin DIP, SOIC; 20-Pin SSOP	18-Pin DIP, SOIC; 20-Pin SSOP	18-Pin DIP, SOIC; 20-Pin SSOP

A Figura I.1 apresenta a estrutura interna do PIC16F628 e a Tabela I.2 a descrição de seus terminais.

Tabela I.2 – Descrição dos terminais do PIC 16F628.

Pino	Nome	Tipo	Descrição
1	RA2/AN2/Vref	Entrada/Saída	Porta A bit 2 / Entrada comparador analógico / Saída da referência de tensão
2	RA3/AN3/CMP1	Entrada/Saída	Porta A bi 3 / Entrada comparador analógico / Saída comparador 1
3	RA4/T0CKI/CMP2	Entrada/Saída	Porta A bit 4 / Entrada de clock externo do timer 0 / Saída comparador 2
4	RA5/MCLR/THV	Entrada	Porta A bit5 / Reset / Tensão de programação
5	V_{SS}	Alimentação	GND
6	RB0/INT	Entrada/Saída	Porta B bit0 / Entrada de Interrupção externa
7	RB1/RX/DT	Entrada/Saída	Porta B bit 1 / Recepção USART (modo assíncrono) / Dados modo síncrono
8	RB2/TX/CK	Entrada/Saída	Porta B bit 2 / Transmissão USART (modo assíncrono) / Clock (modo síncrono)
9	RB3/CCP1	Entrada/Saída	Porta B bit 3 / Entrada/Saída do módulo CCP
10	RB4/PGM	Entrada/Saída	Porta B bit 4 / Entrada de programação LVP
11	RB5	Entrada/Saída	Porta B bit 5
12	RB6/T1OSO/T1CKI	Entrada/Saída	Porta B bit 6 / Saída oscilador TMR1 / Entrada clock TMR1

Pino	Nome	Tipo	Descrição
13	RB7/T1OSI	Entrada/Saída	Porta B bit 7 / Entrada oscilador TMR1
14	V_{DD}	Alimentação	5Vcc
15	RA6/OSC2/CLKOUT	Entrada/Saída	Porta A bit 6 / Entrada para cristal oscilador / Saída de clock
16	RA7/OSC1/CLKIN	Entrada/Saída	Porta A bit 7 / Entrada para cristal oscilador / Entrada de clock externo
17	RA0/AN0	Entrada/Saída	Porta A bit 0 / Entrada comparador analógico
18	RA1/AN1	Entrada/Saída	Porta A bit 1 / Entrada comparador analógico

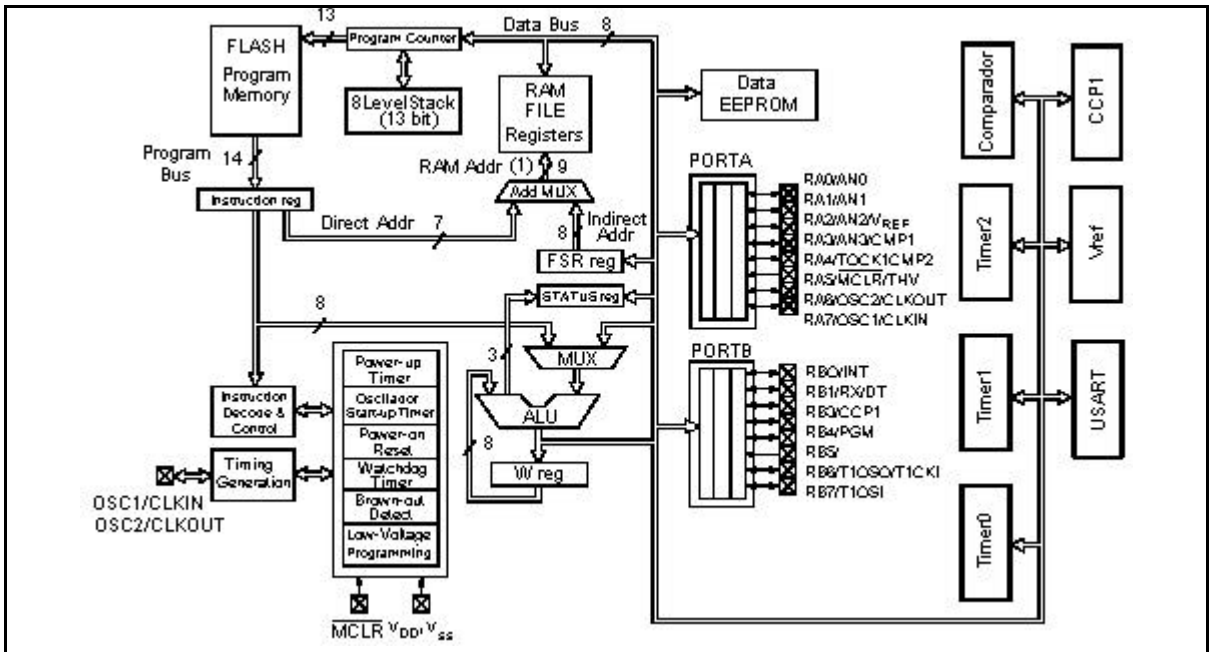


Figura I.1 – Diagrama de blocos da estrutura interna do PIC 16F628.

Como é possível notar na Figura I.1, o barramento de dados e instruções possuem comprimentos diferentes. O barramento de dados é de 8 bits, ao passo que o de instruções, é de 14 bits. Pode-se ver também, os diversos periféricos que o PIC 16F628 possui e que serão considerados adiante.

2 Memória de programa

A memória de programa (MICROCHIP, 2002.a) do PIC 16F628, que pode ser vista na Figura I.2, é do tipo FLASH e pode armazenar até 2048 instruções de 14 bits.

Referente a memória de programa, é interessante ressaltar que:

1- Quando o PIC é energizado, ou após um reset, a primeira instrução tratada é a que se encontra armazenada no vetor de reset.

2- Denomina-se vetor de interrupção, o endereço reservado para o tratamento inicial de toda e qualquer interrupção gerada.

As fontes de interrupção do PIC 16F628 serão consideradas oportunamente.

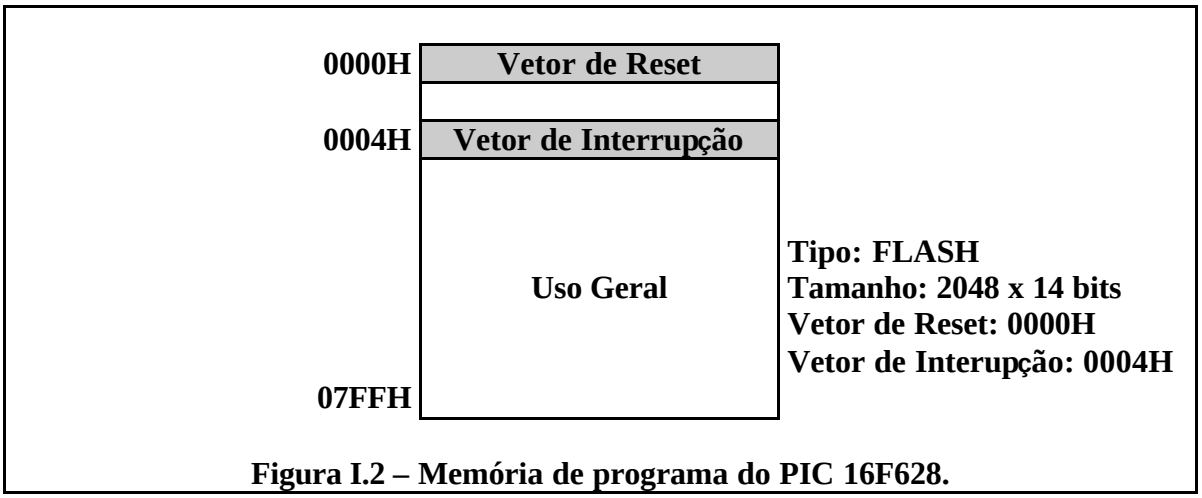


Figura I.2 – Memória de programa do PIC 16F628.

3 Registradores

Os registradores de um microcontrolador PIC estão divididos em dois grupos:

1- **SFR (Special Function Register – Registradores de Funções Especiais):** Encarregados do controle do microcontrolador e de seus periféricos.

2- **GPR (General Purpose Registers – Registradores de Propósito Geral):** São para o usuário a memória RAM, onde se armazena as variáveis e informações necessárias ao programa desenvolvido.

Na Figura I.3, encontra-se o mapa de memória do PIC 16F628, no qual pode-se visualizar todos os registradores, bem como a divisão entre os SFRs e GPRs.

Banco 0		Banco 1		Banco 2		Banco 3	
Acesso Indireto*	00H	Acesso Indireto*	80H	Acesso Indireto*	100H	Acesso Indireto*	180H
TMR0	01H	OPTION	81H	TMR0	101H	OPTION	181H
PCL	02H	PCL	82H	PCL	102H	PCL	182H
STATUS	03H	STATUS	83H	STATUS	103H	STATUS	183H
FSR	04H	FSR	84H	FSR	104H	FSR	184H
PORTA	05H	TRISA	85H		105H		185H
PORTB	06H	TRISB	86H	PORTB	106H	TRISB	186H
	07H		87H		107H		187H
	08H		88H		108H		188H
	09H		89H		109H		189H
PCLATH	0AH	PCLATH	8AH	PCLATH	10AH	PCLATH	18AH
INTCON	0BH	INTCON	8BH	INTCON	10BH	INTCON	18BH
PIR1	0CH	PIE1	8CH		10CH		18CH
	0DH		8DH		10DH		18DH
TMR1L	0EH	PCON	8EH		10EH		18EH
TMR1H	0FH		8FH		10FH		18FH
T1CON	10H		90H				
TMR2	11H		91H				
T2CON	12H	PR2	92H				
	13H		93H				
	14H		94H				
CCPIR1L	15H		95H				
CCPR1H	16H		96H				
CCP1CON	17H		97H				
RCSTA	18H	TXSTA	98H				
TXREG	19H	SPBRG	99H				
RCREG	1AH	EEDATA	9AH				
	1BH	EEADR	9BH				
	1CH	EECON1	9CH				
	1DH	EECON2	9DH				
	1EH		9EH				
CMCON	1FH	VRCON	9FH				
	20H		A0H		11FH		
					120H		
Registradores de Propósito Geral (GPR) 96 Bytes		Registradores de Propósito Geral (GPR) 80 Bytes		Registradores de Propósito Geral (GPR) 48 Bytes	14FH		
					150H		
			EFH		16FH		1EFH
		Acessa Endereços	E0H	Acessa Endereços	170H	Acessa Endereços	1E0H
	7FH	70H a 7FH	FFH	70H a 7FH	17FH	70H a 7FH	1FFH

Posição não implementada, lida como "0". * Não é um registrador físico.

Figura I.3 – Mapa de memória do PIC 16F628.

Ao longo das próximas páginas, todos os SFRs serão abordados. Primeiro os registradores responsáveis pelo controle da CPU e em seguida, os registradores dedicados ao controle de periféricos, que serão analisados juntamente com os periféricos que controlam. Para efeito de padronização, sempre que um bit puder ser lido e/ou escrito individualmente, será indicado pela letra “R” e/ou “W”, respectivamente. O número “0” ou “1”, após estas letras, indica o valor do bit depois de um reset POR (Power-on Reset). Quando um bit de um registrador não for implementado, será sempre lido como “0” e indicado pela letra “U”.

4 Registradores de Controle da CPU

São os registros STATUS, OPTION e PCON, analisados a seguir.

4.1 Registrador STATUS

Presente em todos os bancos de memória, armazena os flags (sinalizadores) da ULA, os bits de seleção do banco de memória e os flags relativos ao WATCHDOG e ao modo SLEEP. A Tabela I.3 exibe os bits do registrador STATUS.

Tabela I.3 – Bits do registrador STATUS.

STATUS (003H, 083H, 103H, 183H)							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
IRP	**RP1**	**RP0**	**TO**	**PD**	**Z**	**DC**	**C**
R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x

IRP É o bit seletor do banco de memória usado para endereçamento indireto. Deve ser mantido sempre em 0 (zero) no PIC 16F628.

0 = Banco 0 e 1 (100H – 1FFH)
1 = Banco 2 e 3 (000H – 0FFH)

RP1 Seletor do banco de memória (Tabela I.4), usado para endereçamento direto. No PIC 16F628, RP1 deve ser sempre mantido em 0 (zero).

RP0 Seletor do banco de memória (Tabela I.4), usado para endereçamento direto.

Tabela I.4 – Seleção do Banco de Memória.

RP1	RP0	BANCO SELECIONADO
0	0	Banco 0 (000H – 07FH)
0	1	Banco 1 (080H – 0FFH)
1	0	Banco 2 (100H – 17FH)
1	1	Banco 3 (180H – 1FFH)

TO Sinalizador de time-out ou estouro do WATCHDOG.

0 = Ocorreu estouro do WATCHDOG (WDT).

1 = Ocorreu POWER-UP ou execução da instrução CLRWDT ou SLEEP.

PD Sinalizador de power-down.

0 = Após execução da instrução SLEEP.

1 = Após execução da instrução CLRWDT ou ocorrência de um power-up.

Z Sinalizador de zero.

0 = O resultado da última operação, lógica ou aritmética, não foi zero.

1 = O resultado da última operação, lógica ou aritmética, foi zero.

DC Transbordo/Empréstimo de dígito (Digit Carry/Borrow). Refere-se a um conjunto de 4 bits (nibble) de um registrador. Dito de uma outra maneira, refere-se a um dígito hexadecimal, uma vez que um registrador de 8 bits armazena dois dígitos hexadecimais ou dois nibbles. Como exemplo:

- ✓ 000**0**1101 + 000**0**0101 = 000**1**0010 \ **DC** = 1
- ✓ 000**0**1000 + 000**0**0001 = 000**0**1001 \ **DC** = 0

0 = Não houve carry-out (estouro de dígito).

1 = Houve carry-out (estouro de dígito).

C Transporte/Empréstimo (Carry/Borrow). Sinaliza a condição de empréstimo ou transporte de um bit após operação matemática ou lógica. Exemplificando:

- ✓ 00001101 + 00000101 = 00010010 \ **C** = 0
- ✓ 11111110 + 00000011 = **1**00000001 \ **C** = 1

0 = Ocorreu carry-out do sétimo bit do registrador.
1 = Não ocorreu carry-out do sétimo bit do registrador.

4.2 Registrador OPTION

O registrador OPTION configura determinadas funções do microcontrolador. Devido ao fato de PICs mais antigos possuírem uma instrução denominada Option, esse registrador é muitas vezes referenciado pelos programadores como OPTION-REG. Na Tabela I.5, pode-se observar os bits do registrador OPTION.

Tabela I.5 – Bits do registrador OPTION.

OPTION (081H, 181H)							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
RPU	**INTEDG**	**T0CS**	**T0SE**	**PSA**	**PS2**	**PS1**	**PS0**
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1

RPU Habilitação dos pull-ups internos dos pinos do PORTB configurados como entrada.

0 = Pull-ups habilitados.
1 = Pull-ups desabilitados.

INTEDG Seleção da borda que gerará a interrupção externa RB0/INT.

0 = Borda descendente do sinal aplicado no pino RB0/INT.
1 = Borda ascendente do sinal aplicado no pino RB0/INT.

T0CS Fonte de clock do TMR0 (timer 0). O TMR0 pode ser incrementado por um sinal interno ao microcontrolador (Fosc/4) ou por um sinal externo, injetado no pino RA4/T0CKI.

0 = Fonte interna.
1 = Fonte externa.

T0SE Escolha da borda que incrementará o TMR0.

0 = Borda de descida do sinal aplicado ao pino RA4/T0CKI.
1 = Borda de subida do sinal aplicado ao pino RA4/T0CKI.

PSA Atribui prescaler ou ao TMR0 ou ao WDT.

0 = Prescaler aplicado ao TMR0.
1 = Prescaler aplicado ao WDT.

PS2, PS1 e PS0 O prescaler é um divisor programável que pode ser aplicado ao WDT ou ao TMR0. Com ele é possível alterar o tempo de estouro de um destes contadores (WDT ou TMR0). Os bits PS2, PS1 e PS0 são usados para ajustar a taxa de divisão do prescaler, conforme Tabela I.6.

Tabela I.6 – Ajuste da taxa de divisão do prescaler.

PS2	PS1	PS0	TMR0 (PSA=0)	WDT (PSA =1)
0	0	0	1:2	1:1
0	0	1	1:4	1:2
0	1	0	1:8	1:4
0	1	1	1:16	1:8
1	0	0	1:32	1:16
1	0	1	1:64	1:32
1	1	0	1:128	1:64
1	1	1	1:256	1:128

4.3 Registrador PCON

Localiza-se no Banco 1 de memória. Utilizado para armazenar Flags de sinalização de dois estados de reset, a saber:

- Reset por queda de tensão de alimentação (Brown-out).
- Reset de inicialização (Power-on reset).

Ainda, quando o microcontrolador opera no modo de oscilador interno ou resistor externo, esse registrador é responsável pelo controle do clock.

A Tabela I.7 exibe os bits do registrador PCON.

Tabela I.7 – Bits do registrador PCON.

PCON (08EH)							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
---	**---**	**---**	**---**	**OSCF**	**---**	**POR**	**BOD**
U	U	U	U	R/W-1	U	R/W-q	R/W-q

Legenda: q = Valor depende de uma condição.

OSCF Freqüência do oscilador de clock interno.

0 = Clock de 37KHz.

1 = Clock de 4MHz.

POR Sinaliza reset de inicialização (POR – Power-on reset).

0 = Houve reset POR.

1 = Não houve reset POR.

BOR Sinaliza reset por queda de tensão de alimentação (Brown-out).

0 = Ocorreu reset por queda de tensão de alimentação.

1 = Não ocorreu reset por queda de tensão de alimentação.

5 Portas de Entrada e Saída

O PIC 16F628 possui duas portas de I/O (Porta A e Porta B), ambas com oito pinos, sendo alguns destes, multiplexados com outras funções. Quando uma destas funções é habilitada, o pino deixa de ser simplesmente uma linha de entrada ou saída de propósito geral (MICROCHIP, 1999). Com o intuito de manter a compatibilidade com os manuais da Microchip, daqui em diante serão usados os termos PORTA (port A) e PORTB (port B).

Tabela I.8 – Registradores associados ao PORTA.

Nome	End.	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PORTA	05H	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0
TRISA	85H	TRISA7	TRISA6	---	TRISA6	TRISA7	TRISA6	TRISA7	TRISA6
CMCON	1FH	C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0
VRCON	9FH	V_{REN}	V_{ROE}	V_{RR}	---	V_{R3}	V_{R2}	V_{R1}	V_{R0}

Nota: Os bits sombreados não são usados pelo PORTA.

A Tabela I.8 mostra os registradores associados ao PORTA e a Tabela I.9 exibe os registradores associados ao PORTB.

Tabela I.9 – Registradores associados ao PORTB.

Nome	End.	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PORTB	06H	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
TRISB	86H	TRISB7	TRISB6	TRISB5	TRISB6	TRISB7	TRISB6	TRISB7	TRISB6
OPTION	81H	RBPU\	INTEDG	TOCS	TOSE	PSA	PS2	PS1	PS0

Nota: Os bits em células sombreadas não são usados pelo PORTB.

5.1 Registradores TRISA e TRISB

Para configuração do PORTA, deve-se usar o registrador TRISA, e para configuração do PORTB, deve-se usar o registrador TRISB.

Esses registradores configuram os pinos das portas como entrada ou como saída. Para configurar um pino como saída, deve-se escrever “1” no bit relacionado a este. Já, para se configurar um pino como entrada, deve-se escrever “0” no bit relacionado. Uma regra prática para memorização, é a associação do “0” ao “O” de Output (saída) e o “1” ao “I” de Input (entrada) (SOUZA, 2000).

5.2 Registradores PORTA e PORTB

Os registradores PORTA e PORTB são usados para se ler/escrever as linhas de entrada/saída digitais. Quando um pino de uma porta está configurado como saída, seu estado lógico pode ser alterado escrevendo-se no bit relacionado a este. Da mesma forma, se um pino está configurado como entrada, ao se ler o bit relacionado ao mesmo, será encontrado o nível lógico a ele aplicado.

6 Interrupções

Antes da análise das interrupções existentes no PIC 16F628, é importante tornar claro o conceito de interrupção.

Uma interrupção é gerada por um evento independente do software. Tratada diretamente pelo hardware, quando ocorre, paralisa o fluxo normal do programa e executa uma função predefinida pelo programador, voltando novamente a execução do programa, do ponto onde havia parado. Isto faz com que a CPU cientifique-se de eventos de alta prioridade para uma dada aplicação.

As dez fontes de interrupção (MICROCHIP, 1999) existentes no PIC 16F628 são:

Timer 0: Ocorre quando há um extravasamento (overflow) do Timer 0, ou seja, na transição do registrador TMR0 de 255 (FFH) para 0 (00H).

Timer 1: Acontece na transição do valor do módulo Timer 1 de 65535 (FFFFH) para 0 (0000H).

Timer 2: Após N coincidências entre o valor de contagem do registro TMR2 e do registrador de período PR2, é que se sucede a interrupção do Timer 2. O valor de N depende do fator de divisão do postscaler (pós-divisor) e é configurável.

Interrupção externa: Esta interrupção é gerada por um sinal externo, aplicado em um terminal específico do microcontrolador, neste caso, o terminal RB0/INT.

Comparador Analógico: É ocasionada pela mudança de estado na saída dos comparadores.

Recepção de dados na USART: A interrupção acontece sempre que um caractere é recebido e encontra-se disponível no registro RCREG.

Final de transmissão de dados na USART: Quando um caractere é passado para o registrador de deslocamento de transmissão (TSR), ocorre a interrupção.

Término de escrita na EEPROM: Essa interrupção permite a identificação do término de uma rotina de escrita na EEPROM interna. Embora a utilização dessa interrupção não seja obrigatória para o funcionamento da rotina de escrita, em muitos casos, ela se faz necessária, a fim de se evitar uma parada durante a escrita na EEPROM.

Mudança de nível lógico no PORTB: O evento de interrupção decorre da mudança do estado lógico em um dos seguintes terminais do PORTB: RB7, RB6, RB5 ou RB4 – desde que o terminal esteja configurado como entrada.

Comparação ou captura no módulo CCP: No modo captura, a interrupção ocorre sempre que uma captura é realizada. No modo comparação, sendo *VERDADE* o resultado de uma comparação efetuada, é gerado um sinal de interrupção.

Ocorrendo uma interrupção no PIC 16F628, o endereço da próxima instrução a ser executada, é armazenado na pilha (Stack) e o programa é desviado para o endereço 04H da memória de programação. Neste endereço, é que se deve escrever a rotina para reconhecimento e tratamento da interrupção ocorrida. Findo o tratamento da interrupção, o programa volta a ser executado do ponto onde havia parado.

Para o controle das interrupções no PIC 16F628, são utilizados três registradores, o INTCON (registrador que controlava todas as interrupções em modelos mais antigos) e os registradores responsáveis pelas chamadas interrupções periféricas, o PIR1 (para sinalização de interrupções periféricas) e o PIE1 (para controle das interrupções periféricas).

6.1 Registrador INTCON

O INTCON é o registrador responsável pelo controle de interrupções no PIC 16F628 e pode ser dividido em três grupos:

1- Controle geral (GIE).
2- Controle individual (PEIE, T0IE, INTE e RBIE).
3- Sinalizadores da ocorrência da interrupção (T0IF, INTF E RBIF).

Os bits que compõem o registrador INTCON podem ser observados na Tabela I.10.

Tabela I.10 – Bits do registrador INTCON.

INTCON (00BH)							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
GIE	**EEIE**	**TOIE**	**INTE**	**RBIE**	**T0IF**	**INTF**	**RBIF**
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x

GIE Habilitação global das interrupções (Global Interrupt Enable). Funciona como uma chave geral.

0 = Chave geral desligada. Nenhuma interrupção ocorrerá.
1 = Chave geral ligada. Pode ocorrer uma interrupção.

PEIE Bit de habilitação das interrupções periféricas (chave individual).

0 = Interrupção periférica habilitada.
1 = Interrupção periférica desabilitada.

T0IE Habilitação da interrupção de estouro (overflow) do TMR0 (chave individual).

0 = Interrupção de estouro do TMR0 desabilitada.
1 = Interrupção de estouro do TMR0 habilitada.

INTE Interrupção externa RB0/INT (chave individual).

0 = Interrupção externa desabilitada.
1 = Interrupção externa habilitada.

RBIE Interrupção por mudança de estado nos pinos RB4 – RB7 (chave individual).

0 = Desabilitada. 1 = Habilitada.

T0IF Sinaliza interrupção de estouro (overflow) do TMR0.

0 = Não ocorreu overflow do TMR0. 1 = Ocorreu overflow do TMR0.

INTF Sinaliza interrupção externa (RB0/INT).

0 = Não ocorreu esta interrupção. 1 = Ocorreu esta interrupção.

RBIF Sinaliza interrupção por mudança de estado nos pinos RB4 – RB7.

0 = Esta interrupção não ocorreu. 1 = Esta interrupção ocorreu.

6.2 Registrador PIE1

Esse registrador possui os bits de habilitação das chamadas interrupções periféricas. A tabela I.11 apresenta os bits do registrador PEI1.

Tabela I.11 – Bits do registrador PEI1.

PCON (08EH)							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EEIE	**CMIE**	**RCIE**	**TXIE**	**---**	**CCP1IE**	**TMR2IE**	**TMR1IE**
R/W-0	R/W-0	R/W-0	R/W-0	U	R/W-0	R/W-0	R/W-0

EEIE Habilita interrupção por término de escrita na EEPROM interna.

0 = Interrupção desabilitada. 1 = Interrupção habilitada.

CMIE Habilita interrupção por mudança de estado na saída dos comparadores.

0 = Interrupção desabilitada. 1 = Interrupção habilitada.

RCIE Habilita interrupção de recepção de dados USART.

0 = Interrupção desabilitada. 1 = Interrupção habilitada.

TXIE Habilita interrupção de transmissão USART.

0 = Interrupção desabilitada. 1 = Interrupção habilitada.

CCP1IE Habilita interrupção do módulo CCP.

0 = Interrupção desabilitada. 1 = Interrupção habilitada.

TMR2IE Habilita interrupção de estouro (overflow) do TMR1.

0 = Interrupção desabilitada. 1 = Interrupção habilitada.

TMR1IE Habilita interrupção de estouro (overflow) do TMR2.

0 = Interrupção desabilitada. 1 = Interrupção habilitada.

6.3 Registrador PIR1

Armazena os sinalizadores (flags) de interrupções periféricas e gera a interrupção, caso esta esteja habilitada. Os bits de sinalização, abaixo relacionados, são setados pelo evento gerador de interrupção, independente da interrupção individual estar ou não habilitada. A Tabela I.12 apresenta os bits do registrador PIR1.

Tabela I.12 – Bits do registrador PIR1.

PIR1 (00CH)							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EEIF	**CMIF**	**RCIF**	**TXIF**	**---**	**CCP1IF**	**TMR2IF**	**TMR1IF**
R/W-0	R/W-0	R-0	R-0	U	R/W-0	R/W-0	R/W-0

EEIF Sinaliza término de escrita na EEPROM.

0 = A escrita ainda não foi concluída ou não começou.

1 = A escrita foi concluída.

CMIF Sinaliza mudança de estado na saída dos comparadores analógicos.

0 = Não ocorreu mudança de estado na saída dos comparadores.

1 = Ocorreu mudança de estado na saída dos comparadores.

RCIF Sinaliza recepção de caractere na USART.

0 = Não houve recepção de caractere.

1 = Um novo caractere foi recebido.

TXIF Sinaliza transmissão na USART.

0 = Não ocorreu transmissão na USART ou o TSR está ocupado.

1 = Caractere armazenado no TXREG foi repassado para o TSR.

CCPIF Sinalizador do módulo CCP.

0 = Não houve comparação ou captura.

1 = Houve comparação ou captura.

TMR2IF Sinaliza estouro (overflow) no TMR2.

0 = Não ocorreu Overflow no TMR2.

1 = Ocorreu overflow no TMR2.

TMR1IF Sinaliza estouro (overflow) no TMR2.

0 = Não ocorreu Overflow no TMR1.

1 = Ocorreu overflow no TMR1.

7 Contador de programa – PC

O escopo do contador de programa (PC – Program Counter) é controlar a seqüência de execução das instruções. Trata-se de um registrador de 13 bits que aponta constantemente para a próxima instrução a ser tratada pela CPU. Portanto, é possível alterar o fluxo normal do programa, alterando o conteúdo do PC.

Os 8 bits menos significativos do PC, são alojados num registrador denominado PCL, enquanto os 5 bits mais significativos, são contidos pelo registrador chamado PCH.

O conteúdo do PCL é diretamente acessível e pode ser alterado pelo programa do usuário, diferentemente do conteúdo do PCH, controlado diretamente pelo hardware. Pode-se, no entanto, alterar o conteúdo do PCH por meio do registrador PCLATH.

Tabela I.13 – Bits do registrador PCL.

PCL (002H, 082H, 102H, 182H)							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

A Tabela I.13 apresenta os bits constituintes do registrador PCL e a Tabela I.14 mostra os bits do registrador PCLATH.

Tabela I.14 – Bits do registrador PCLATH.

PCLATH (00AH, 08AH, 10AH, 18AH)							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

8 Contador/Temporizador TMR0

O Timer 0 é um contador binário de 8 bits que incrementa seu valor a cada ciclo de instrução (Fosc/4), no modo temporizador, ou a cada pulso na entrada RA/T0CKI, no modo contador.

O registro TMR0, pode ser acessado diretamente tanto para escrita como para leitura e seu extravasamento pode gerar uma interrupção, se a mesma estiver habilitada. Abaixo estão alistadas suas principais características:

- Contador/temporizador de 8 bits.
- Pode-se ler ou escrever no registro TMR0.
- Sinal de contagem pode ser dividido por um prescaler programável.
- Seleção de clock interno ou externo.
- Contagem do sinal externo na borda de subida ou de descida.
- Geração de interrupção por overflow de FFH para 00H, se estiver habilitada.

9 Contador/Temporizador Timer 1

O Timer 1 é um contador/temporizador de 16 bits, constituído por um par de registradores de 8 bits, o TMR1H, e o TMR1L. O TMR1H armazena os 8 bits mais significativos enquanto o TMR1L os 8 bits menos significativos.

A freqüência das interrupções do Timer 1 pode ser calculada com o uso da equação (1.1).

$$F_{INT} = \frac{CLOCK / PRESCALER}{(65536 - TMR1)} \qquad (1.1)$$

Onde :

- ✓ **CLOCK:** Valor da freqüência de clock (interno ou externo).
- ✓ **PRESCALER:** Fator de divisão programado no prescaler do Timer 1.
- ✓ **TMR1:** Valor inicial de 16 bits contido no TMR1.
- ✓ **F_{INT}:** Freqüência das interrupções.

O módulo Timer 1 pode operar como temporizador ou contador (síncrono ou assíncrono), de acordo com o configurado no registro T1CON. No modo temporizador, o Timer 1 é incrementado a cada ciclo de instrução (a cada 1µs para F_{OSC} =

4MHz). No modo contador, é incrementado a cada borda de subida do sinal de clock externo.

Atuando como contador, o Timer 1 pode operar no modo assíncrono ou síncrono (em modo SLEEP pode operar somente em modo assíncrono). O que caracteriza a diferença entre o modo síncrono e assíncrono, é que no modo síncrono, circuitos internos realizam a sincronização entre o sinal de clock interno e externo, o que implica em atraso entre a chegada do pulso de clock na entrada RB6/T1OSO/T1CKI e sua aplicação ao contador. Por outro lado, no modo assíncrono, o incremento do contador é feito sem sincronização entre clock interno e externo, o que pode proporcionar certos inconvenientes, aos quais o programador deve estar atento.

Para aplicações de mensuração de tempo, como relógios de tempo real, o Timer 1 possui um oscilador interno que o possibilita trabalhar com cristais externos de freqüências baixas, até 200KHz. A Tabela I.15 apresenta os valores de capacitores, recomendados pela Microchip, para as freqüências mais usuais de cristais osciladores.

Tabela I.15 – Seleção de capacitores para o Timer 1.

Freqüência do oscilador tipo LP	C1	C2
32KHz	33pF	33pF
100KHz	15pF	15pF
200KHz	15pF	15pF

9.1 Registrador T1CON

O T1CON (Tabela I.16) é o registro de controle através do qual é possível configurar o modo de operação e o fator de divisão do prescaler do Timer 1. A Tabela I.17 mostra como configurar o Timer 1 para trabalhar com sinal de clock interno, sinal de clock externo assíncrono ou sinal de clock externo síncrono.

Tabela I.16 – Bits do registrador T1CON.

T1CON (010H)							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
---	---	**T1CKPS1**	**T1CKPS0**	**T1OSCEN**	**T1SYNC**	**TMR1CS**	**TMR1ON**
U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

T1CKPS1 Seleciona fator de divisão do prescaler do Timer 1. Veja Tabela I.18.

T1CKPS0 Seleciona fator de divisão do prescaler do Timer 1. Veja Tabela I.18.

Tabela I.17 – Seleção do sinal de clock para o Timer 1.

TMR1CS	T1SYNC\	Clock
0	0	Interno
0	1	Interno
1	0	Externo sincronizado
1	1	Externo não sincronizado

Tabela I.18 – Configuração do prescaler do Timer1.

T1CKPS1	T1CKPS0	Fator de Divisão
0	0	1:1
0	1	1:2
1	0	1:4
1	1	1:8

T1OSCEN Bit de controle para habilitação do oscilador do Timer 1.

0 = Oscilador desabilitado.

1 = Oscilador habilitado.

T1SYNC Seleciona modo de operação com sinal de clock externo.

0 = Modo síncrono.

1 = Modo assíncrono.

TMR1CS Opção de clock do Timer 1.

0 = Clock interno ($F_{OSC}/4$).

1 = Clock externo (pinos RB6/T1OSO/T1CKI e RB7/T1OSI).

TMR1ON Bit de habilitação da contagem do Timer 1.

0 = Paralisa contagem do Timer 1.

1 = Habilita contagem do Timer 1.

10 Módulo Timer 2

O Timer 2, trata-se de um temporizador de 8 bits com prescaler (pré-divisor) e postscaler (pós-divisor). Embora seu objetivo, a priori, seja fornecer a base de tempo para o módulo CCP, que será visto adiante, pode também ser empregado para outros propósitos, tal como na função de um temporizador de uso geral.

Na Figura I.4 pode-se observar o diagrama de blocos do Timer 2. É possível notar que o módulo CCP recebe o sinal de extravasamento diretamente do registro TMR2. O bit TMR2IF (PIR1), por sua vez, é setado somente após o sinal de extravasamento passar pelo postscaler. Também, é interessante verificar a presença de um registrador de período, denominado PR2, cuja função é permitir a alteração do valor de extravasamento do TMR2. O TMR2 volta a conter o valor 00H sempre após o comparador detectar coincidência entre o valor do TMR2 e do PR2. Após um reset, o valor acumulado no PR2 é 255 (FFH). Nesse caso, o extravasamento do TMR2 ocorre sempre que seu valor passar de 255 (FFH) para 0 (00H). Se, no entanto, PR2 for carregado com outro valor, pelo programa do usuário, por exemplo, 100 (64H), o TMR2 extravasará sempre que seu valor se igualar a este, no caso, na passagem de 100 (64H) para 0 (00H). Ainda, com base na Figura I.4, fica conspícuo que o Timer 2 não pode operar como um contador de eventos, uma vez que o sinal de clock pode provir apenas do sistema ($F_{OSC}/4$), não havendo possibilidade de seleção de sinal de clock externo, como no Timer 0 e no Timer 1.

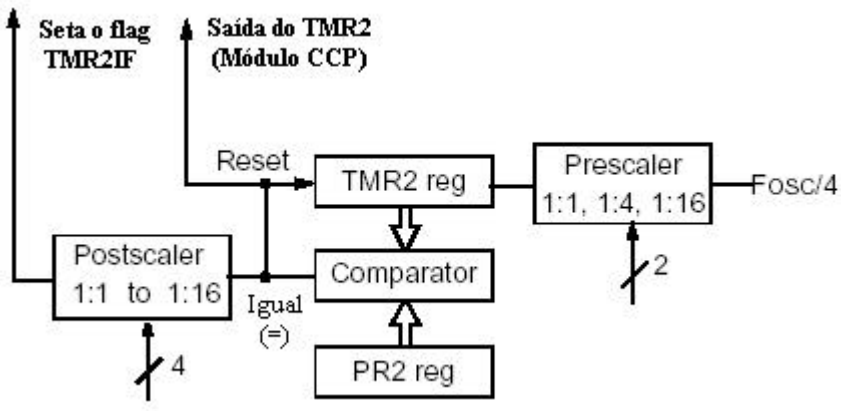


Figura I.4 – Diagrama de blocos do Timer 2.

10.1 Registrador TMR2CON

O registro de controle do Timer 2, visto na Tabela I.19, é denominado TMR2CON. Por meio dele, pode-se configurar tanto o prescaler, quanto o postscaler do Timer 2.

Tabela I.19 – Bits do registrador TMR2CON.

TMR2CON (012H)							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
---	**TOUTPS3**	**TOUTPS2**	**TOUTPS1**	**TOUTPS0**	**TMR2ON**	**T2CKPS1**	**T2CKPS0**
U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

TOUTPS3 Bit de configuração do postscaler do Timer 2 (Tabela I.20).

TOUTPS2 Bit de configuração do postscaler do Timer 2 (Tabela I.20).

TOUTPS1 Bit de configuração do postscaler do Timer 2 (Tabela I.20).

TOUTPS0 Bit de configuração do postscaler do Timer 2 (Tabela I.20).

Tabela I.20 – Bits de configuração do Postscaler.

TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	Postscaler
0	0	0	0	1:1
0	0	0	1	1:2
0	0	1	0	1:3
0	0	1	1	1:4
0	1	0	0	1:5
0	1	0	1	1:6
0	1	1	0	1:7
0	1	1	1	1:8
1	0	0	0	1:9
1	0	0	1	1:10
1	0	1	0	1:11
1	0	1	1	1:12
1	1	0	0	1:13
1	1	0	1	1:14
1	1	1	0	1:15
1	1	1	1	1:16

TMR2ON Bit de habilitação do Timer 2.

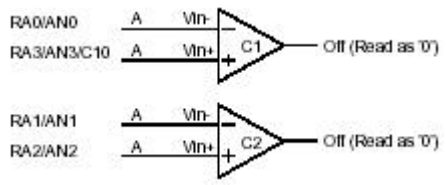
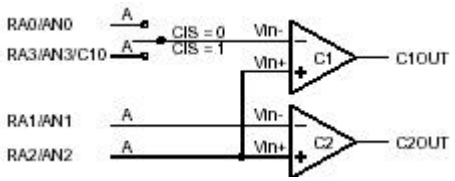
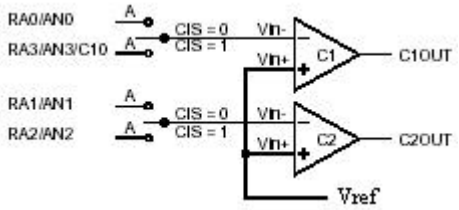
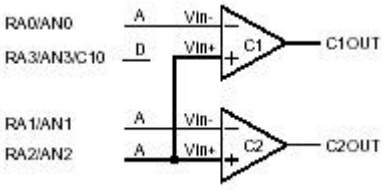
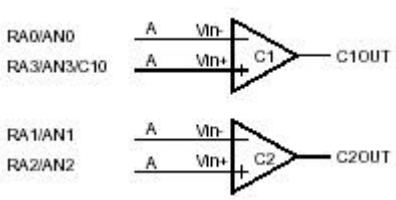
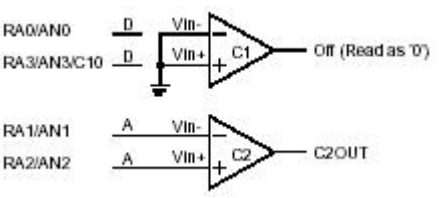
0 = Timer 2 ligado. 1 = Timer 2 desligado.

T2CKPS1 Bit de configuração do prescaler do Timer 2 (Tabela I.21).

T2CKPS0 Bit de configuração do prescaler do Timer 2 (Tabela I.21).

Tabela I.21 – Bits de Configuração do Prescaler.

T2CKPS1	T2CKPS0	Prescaler
0	0	1:1
0	1	1:4
1	0	1:16
1	1	1:16

Modo 0 (CM2,CM1,CM0) = (0,0,0)	Modo 1 (CM2,CM1,CM0) = (0,0,1)
	
Modo 2 **(CM2,CM1,CM0) = (0,1,0)**	**Modo 3** **(CM2,CM1,CM0) = (0,1,1)**
	
Modo 4 **(CM2,CM1,CM0) = (1,0,0)**	**Modo 5** **(CM2,CM1,CM0) = (1,0,1)**
	

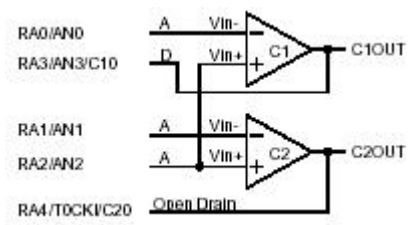
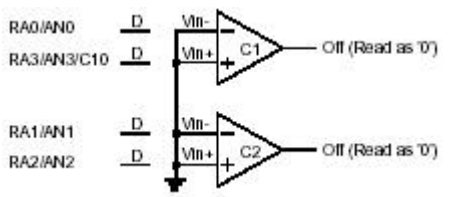
Modo 6 (CM2,CM1,CM0) = (1,1,0)	Modo 7 (CM2,CM1,CM0) = (1,1,1)
	

Figura I.5 – Configurações possíveis para o módulo comparador analógico.

11 Módulo Comparador

O módulo comparador do PIC 16F628 é composto por dois comparadores analógicos e pode ser configurado de sete modos diferentes, apresentados na Figura I.5. A configuração e o controle deste módulo são feitos através do registrador CMCON.

11.1 Registrador CMCON

Neste registrador, encontram-se os bits necessários para configuração e controle do módulo comparador. A Tabela I.22 permite visualizar os bits que compõem este registrador.

Tabela I.22 – Bits do registrador CMCON.

CMCON (01FH)							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
C2OUT	**C1OUT**	**C2INV**	**C1INV**	**CIS**	**CM2**	**CM1**	**CM0**
R-0	R-0	R/W-0	R/W-0	R/M-0	R/W-0	R/W-0	R/W-0

C2OUT Indica o estado da saída do comparador 2.

Para C2INV = 0	**Para C2INV = 1**
0 = Entrada Vin+ < Vin-	0 = Entrada Vin+ > Vin-
1 = Entrada Vin+ > Vin-	1 = Entrada Vin+ < Vin-

C1OUT Indica o estado de saída do comparador 1.

Para C1INV = 0	**Para C1INV = 1**
0 = Entrada Vin+ < Vin-	0 = Entrada Vin+ > Vin-
1 = Entrada Vin+ > Vin-	1 = Entrada Vin+ < Vin-

C2INV Seleciona inversão do sinal de saída do comparador 2.

0 = Saída não invertida.
1 = Saída invertida.

C1INV Seleciona inversão do sinal de saída do comparador 1.

0 = Saída não invertida.
1 = Saída invertida.

CIS Seleciona entrada dos comparadores nos modos 1 e 2.

No modo 1	**No modo 2**
0 = Entrada Vin- de C1 conectada em RA0	0 = Entrada Vin- de C1 conectada em RA0 Entrada Vin- de C2 conectada em RA1
No modo 1	**No modo 2**
1 = Entrada Vin- de C1 conectada em RA3	1 = Entrada Vin- de C1 conectada em RA3 Entrada Vin- de C2 conectada em RA2

NOTA: De acordo com o documento de errata DS80073E (MICROCHIP, 2002.b), existe um BUG interno no módulo comparador analógico quando configurado para o modo 1. Não se deve utilizar o modo 1 em conjunto bit CIS em nível lógico alto (“1”), pois o comparador 1 pode funcionar de maneira errática.

Tabela I.23 – Configuração do módulo comparador.

CM2	CM1	CM0	Modo
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

CM2 Seleção do modo de operação do módulo comparador (Tabela I.23).

CM1 Seleção do modo de operação do módulo comparador (Tabela I.23).

CM0 Seleção do modo de operação do módulo comparador (Tabela I.23).

12 Módulo de referência interna de tensão

O módulo de referência interna de tensão, cujo registro de controle é o VRCON, constitui-se basicamente num complemento ao módulo comparador analógico, fornecendo a este, uma referência programável de tensão.

12.1 Registrador VRCON

O registrador VRCON é constituído pelos bits empregados na configuração e controle do módulo de referência interna de tensão, conforme mostra a Tabela I.24.

Tabela I.24 – Bits do registrador VRCON.

VRCON (09FH)							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
V_{REN}	**V_{ROE}**	**V_{RR}**	**---**	**V_{R3}**	**V_{R2}**	**V_{R1}**	**V_{R0}**
R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0

V_{REN} Bit de habilitação do módulo de referência interna de tensão (VREF).

0 = Módulo VREF desligado.
1 = Módulo VREF ligado.

V_{ROE} Conexão da saída VREF ao pino RA2.

0 = Saída VREF desconectada do pino RA2.
1 = Saída VREF conectada ao pino RA2.

V_{RR} Bit de seleção de escala de tensão (Tabela X).

0 = Escala grande.
1 = Escala pequena.

V_{R3} Seleção do nível de tensão de saída do módulo VREF (Tabela 5.25).

V_{R2} Seleção do nível de tensão de saída do módulo VREF (Tabela 5.25).

V_{R1} Seleção do nível de tensão de saída do módulo VREF (Tabela 5.25).

V_{R0} Seleção do nível de tensão de saída do módulo VREF (Tabela 5.25).

Tabela I.25 – Valores de saída do módulo VREF para V_{DD} = 5V.

V_{R3}	V_{R2}	V_{R1}	V_{R0}	V_{RR} = 1	V_{RR} = 0
0	0	0	0	0,00V	1,25V
0	0	0	1	0,21V	1,41V
0	0	1	0	0,42V	1,56V
0	0	1	1	0,63V	1,72V
0	1	0	0	0,83V	1,88V
0	1	0	1	1,04V	2,03V
0	1	1	0	1,25V	2,19V
0	1	1	1	1,46V	2,34V
1	0	0	0	1,67V	2,50V
1	0	0	1	1,88V	2,66V
1	0	1	0	2,08V	2,81V
1	0	1	1	2,29V	2,97V
1	1	0	0	2,50V	3,13V
1	1	0	1	2,71V	3,28V
1	1	1	0	2,92V	3,44V
1	1	1	1	3,13V	3,59V

13 Módulo de Captura/Comparação/PWM

O módulo CCP (Capture/Compare/PWM – Captura/Comparação/PWM) pode ser empregado para:

- **Captura:** Mensura o período de um sinal administrado ao pino 9 (RB3/CCP1) do PIC 16F628.

- **Comparação:** O valor do registro de 16 bits do módulo CCP é comparado com o valor do Timer 1. Havendo coincidência, pode-se provocar uma interrupção (TMR1IF), setar ou ressetar o terminal

CCP1, ou ainda, ressetar o Timer 1. O modo comparação é indicado para geração de pulsos de largura controlada por software.

- **PWM:** A produção de um sinal PWM, que pode ser disponibilizado através terminal RB3/CCP1, é obtida pelo trabalho conjunto do Timer 2 e do módulo CCP.

A Tabela I.26 exibe as bases de tempo que podem ser utilizadas pelo módulo CCP.

Tabela I.26 – Bases de tempo para o módulo CCP.

Módulo CCP	Base de Tempo
Modo Captura	Timer 1
Modo Comparação	Timer 1
Modo PWM	Timer 2

Na seqüência, serão abordados os registradores do módulo CCP e em seguida, os modos de operação deste módulo.

13.1 Registrador CCP1CON

O CCP1CON (Tabela I.27) é o registrador de controle do módulo CCP.

Tabela I.27 – Bits do registrador CCP1CON.

CCP1CON (017H)							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
---	**---**	**CCP1X**	**CCP1Y**	**CCP1M3**	**CCP1M2**	**CCP1M1**	**CCP1M0**
U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

CCP1X

CCP1Y Quando em modo PWM, são os bits menos significativos do registrador de ciclo ativo. Em modo de Captura ou Comparação, estes bits não são utilizados. A Tabela I.28 apresenta exemplos hipotéticos de uso destes bits.

CCP1M3 Bit de seleção do modo de operação do módulo PWM (Tabela I.29).

CCP1M2 Bit de seleção do modo de operação do módulo PWM (Tabela I.29).

CCP1M1 Bit de seleção do modo de operação do módulo PWM (Tabela I.29).

CCP1M0 Bit de seleção do modo de operação do módulo PWM (Tabela I.29).

Tabela I.28 – Exemplos de uso dos bits CCP1X e CCP1Y.

CCPR1L	CCP1X	CCP1Y	Valor Decimal
00000000	0	0	0
00000000	0	1	1
00000001	1	1	7
10101010	1	0	682
11111111	1	1	1023

Tabela I.29 – Seleção do modo de operação do Módulo CCP.

CCP1M3	CCP1M2	CCP1M1	CCP1M0	MODO
0	0	0	0	Módulo CCP desligado, reset CCP.
0	1	0	0	Captura a cada borda descendente.
0	1	0	1	Captura a cada borda ascendente.
0	1	1	0	Captura a cada 4ª borda ascendente.
0	1	1	1	Captura a cada 16ª borda ascendente.
1	0	0	0	Comparação, CCP1 inicia em 0 e muda para 1 em caso de coincidência.
1	0	0	1	Comparação, CCP1 inicia em 1 e muda para 0 em caso de coincidência.
1	0	1	0	Comparação, provoca interrupção CCP1IF em caso de coincidência.
1	0	1	1	Comparação, provoca interrupção CCP1IF e reseta TMR1 em caso de coincidência.
1	1	X	X	Modo PWM.

13.2 Registrador CCPR1L

O registro CCPR1L, cujo endereço é 015H, no modo PWM, tem como escopo armazenar os 8 bits menos significativos usados na determinação do ciclo ativo do sinal. Já, nos outros dois modos, tem como objetivo armazenar os 8 bits menos significativos empregados na captura/comparação de um sinal.

13.3 Registrador CCPR1H

Os 8 bits menos significativos, resultantes da captura/comparação de sinais, são armazenados no registro CCPR1H.

No modo PWM, o registro CCPR1H, cujo endereço é 016H, é usado como um registrador escravo, sendo nele copiado o conteúdo do registro CCPR1L. “Isto permite uma operação livre de ruídos e interferência, que poderiam ser introduzidas no sinal, caso fosse utilizado o registrador CCPR1L diretamente” (Pereira, 2002).

13.4 Modo de Captura

Neste modo, pode-se mensurar o período de um sinal aplicado a entrada CCP1, sendo o valor guardado nos registradores CCPR1L e CCPR1H.

Toda vez que ocorrer uma borda ascendente ou descendente, dependendo do modo de operação, no sinal aplicado ao terminal CCP1, é produzido um sinal de captura, que providencia a cópia do valor dos registradores TMR1L e TMR1H do módulo Timer 1 para os registradores CCPR1L e CCPR1H. O sinal de captura é responsável também, por setar o bit CCP1IF do registrador PIR1, gerando uma interrupção, caso esta esteja habilitada (CCP1IE = 1, PEIE = 1 , GIE = 1).

13.5 Modo de Comparação

No modo de comparação, o valor contido pelo par de registradores do módulo Timer 1 (TMR1L e TMR1H), é constantemente comparado com o valor armazenado nos registradores CCPR1L e CCPR1H. Havendo coincidência e de acordo com o que foi previamente configurado, pode ocorrer de um dos seguintes eventos:

1- O terminal CCP1 é ressetado e é produzido um sinal de interrupção CCP1IF. Neste caso, o terminal CCP1 começa com nível lógico baixo (0) e passa para nível lógico alto (1) após uma comparação cujo resultado seja *VERDADE*.

2- O terminal CCP1 é setado e é produzido um sinal de interrupção CCP1IF. Neste caso, o terminal CCP1 começa com nível lógico alto (1) e passa para nível lógico baixo (0) após uma comparação cujo resultado seja *VERDADE*.

3- O estado lógico do terminal CCP1 não se modifica e um sinal de interrupção CCP1IF é gerado. Neste caso, ocorrendo uma comparação cujo resultado seja *VERDADE*, o bit CCP1IF é setado e a interrupção acontece, caso esteja habilitada (CCP1IE = 1).

4- O estado lógico do terminal CCP1 não se modifica, um sinal de interrupção CCP1IF é gerado e o Timer 1 é ressetado. Neste caso, ocorrendo uma comparação cujo resultado seja *VERDADE*, o bit CCP1IF é setado e a interrupção acontece, caso esteja habilitada (CCP1IE = 1). Além disso, o Timer 1 é ressetado.

13.6 Modo PWM

Sinais PWM (Pulse Width Modulation – Modulação de Largura de Pulso) são muito utilizados, entre outras coisas, para o controle da velocidade e do torque de motores de corrente contínua. Embora a modulação da largura de pulso de um sinal possa ser efetuada por software, o hardware específico para tal, como oferece o PIC 16F628, pode economizar tempo do projetista, além de poupar espaço na memória de programa.

No PIC 16F628, o período e a freqüência (conseqüentemente) de um sinal podem ser estabelecidos pela programação do Timer 2, uma vez que o Timer 2 serve de base de tempo para o módulo CCP, no modo PWM. A resolução máxima obtida é de 10 bits, conforme Tabela 5.26, podendo ser diminuída em função da freqüência máxima do sinal PWM de saída.

O período em segundos do sinal PWM é dado pela equação (1.2).

$$T_{PWM} = [(PR2) + 1] \times 4\,T_{OSC} \times (PRESCALERTMR2) \qquad (1.2)$$

Para cálculo do ciclo ativo do sinal PWM pode-se usar a equação (1.3).

$$T_{CICLOATIVO} = (CCPR1L + CCP1X + CCP1Y) \times T_{OSC} \times (PRESCALERTMR2) \qquad (1.3)$$

A resolução máxima em bits para uma dada freqüência é obtida pela equação (1.4).

$$\text{Re}sol._{(BITS)} = \frac{\log\left(\frac{F_{OSC}}{F_{PWM}}\right)}{\log(2)} \qquad (1.4)$$

14 Interface de Comunicação Serial

A Interface Serial Universal Síncrona/Assíncrona (USART – Universal Synchronous Asynchronous Receiver Transmitter), também chamada de Interface de comunicação Serial (SCI – Serial Communications Interface), possibilita a comunicação entre o microcontrolador e dispositivos externos, tais como: conversores A/D e D/A, computadores, sensores e modem (acrônimo de MOdulator/DEModulator – MOdulador/DEModulador).

No PIC 16F628, o módulo USART pode ser configurado de quatro modos diferentes:

1- Assíncrono Full-duplex sem detecção de endereço;
2- Assíncrono Full-duplex com detecção de endereço;
3- Síncrono Half-duplex com clock interno (modo mestre);
4- Síncrono Half-duplex com clock externo (modo mestre).

Os pinos empregados pelo microcontrolador para transmissão/recepção de sinais, são:

RB1 Entrada de dados quando a SCI está operando no modo assíncrono.
Entrada/Saída de dados quando a SCI está operando no modo síncrono.

RB2 Saída de dados quando a SCI está operando no modo assíncrono.
Saída de clock quando a SCI está operando no modo síncrono com clock interno.
Entrada de clock quando a SCI está operando no modo síncrono com clock externo.

A SCI do PIC 16F628, possui um gerador interno de clock, denominado BRG (Baud[1] Rade Generator – Gerador da Taxa da Velocidade de Transmissão), para seus dois registradores de deslocamento, cuja utilidade será vista a seguir. Posteriormente serão tratados os registradores responsáveis pela configuração e controle da SCI.

14.1 Transmissão/Recepção de dados no modo assíncrono

O PIC 16F628 possui dois registradores de deslocamento, um denominado TSR e outro RSR, utilizados na transmissão e recepção de dados, respectivamente. A existência desses dois registradores independentes permite que a transmissão e a recepção de dados possam ocorrer simultaneamente (Full-Duplex).

Na comunicação em modo assíncrono, empregam-se velocidades pré-estabelecidas para comunicação entre elemento receptor e transmissor, além de bits de início (START) e de fim (STOP), para sinalizar o começo e o final de uma comunicação.

14.2 Transmissão/Recepção de dados no modo síncrono

A comunicação síncrona emprega uma linha de clock para sincronização do elemento transmissor com o elemento receptor, diferentemente da comunicação assíncrona. Cada vez que um bit é disponibilizado na linha de dados pelo dispositivo transmissor, na linha de clock é provido um pulso indicativo para o dispositivo receptor, sincronizando assim a transmissão e recepção.

[1] Em homenagem ao engenheiro francês Jean Maurice Émile Baudot (1845–1903), que inventou o telégrafo múltiplo, foi cunhado o termo Baud, que originalmente, indicava a velocidade de transmissão do telégrafo.

14.3 Registrador TXSTA

O registrador TXSTA (Tabela I.30) é responsável pelo controle e configuração da transmissão da SCI.

Tabela I.30 – Bits do registrador TXSTA.

TXSTA (098H)							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CSRC	**TX9**	**TXEN**	**SYNC**	**---**	**BRGH**	**TRMT**	**TX9D**
R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0

CSRC Seletor da fonte de clock para o modo síncrono.

0 = Clock externo (modo escravo).
1 = Clock interno (modo mestre).

TX9 Habilita modo de transmissão de 9 bits.

0 = Modo de transmissão de 8 bits.
1 = Modo de transmissão de 9 bits.

TXEN Habilita Transmissão.

0 = Transmissão desabilitada.
1 = Transmissão habilitada.

SYNC Seletor do modo de operação da SCI.

0 = Modo assíncrono de transmissão.
1 = Modo síncrono de transmissão.

BRGH Seleciona modo de clock, tendo função somente no modo assíncrono.

0 = Modo de baixa velocidade.
1 = Modo de alta velocidade.

TRMT Indica o estado do registrador TSR.

0 = Registrador de transmissão (TSR) está cheio.
1 = Registrador de transmissão (TSR) está vazio.

TX9D Bit usado apenas no modo de 9 bits. Esse 9º bit pode ser usado, por exemplo, na detecção de paridade.

14.4 Registrador RCSTA

Este registrador, apresentado na Tabela I.31, é responsável pelo controle e configuração da recepção da SCI.

Tabela I.31 – Bits do registrador RCSTA.

RCSTA (018H)							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SPEN	**RX9**	**SREN**	**CREN**	**ADEN**	**FERR**	**OERR**	**RX9D**
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-0

SPEN Habilitação do módulo USART.

0 = USART desabilitada (RB1 e RB2 usados como I/O).
1 = USART habilitada (RB1 e RB2 dedicados a USART).

RX9 Habilita modo de recepção de 9 bits.

0 = Modo de recepção de 8 bits.
1 = Modo de recepção de 9 bits.

SREN Habilita recepção única. Após recepção de um caractere o bit volta a “0”.

0 = Recepção única desabilitada.
1 = Recepção única habilitada.

CREN Habilita recepção contínua.

0 = Recepção contínua desabilitada.
1 = Recepção contínua habilitada.

ADEN Habilita detecção de endereço (não deve ser usada no modo síncrono).

0 = Detecção de endereço desabilitada. O 9º bit pode ser usado para paridade.
1 = Detecção de endereço habilitada. A USART aguarda a chegada de um endereço.

FERR Indica erro de Frame.

0 = Sem erro de Frame (Bit STOP detectado no momento correto).
1 = Erro de Frame (Bit STOP não detectado ou detectado fora de hora).

OERR Indica erro de overrun (sobreposição).

0 = Não houve erro de overrun.
1 = Houve erro de overrun.

RX9D O 9º bit de dados pode ser recebido para detecção de paridade, desde que o transmissor esteja configurado para envia-lo.

0 = Recepção contínua desabilitada.
1 = Recepção contínua habilitada.

14.5 Registrador SPBRG

O SPBRG é o registrador responsável pelo controle do gerador interno de baud rate da ICS.

Diferentes taxas de transmissão podem ser obtidas para a ICS, bastando para isso, modificar os valores do registrador SPBRG.

A equação (1.5) viabiliza encontrar o valor a ser armazenado no registrador SPBRG, para se obter a taxa de transmissão desejada.

$$BaudeRate = \frac{F_{OSC}}{\left[Y\left(X+1\right)\right]} \qquad (1.5)$$

Na equação (1.5), X é o valor a ser armazenado no registrador SPBRG e o valor de Y varia de acordo com o modo de trabalho do módulo USART, conforme mostra a Tabela I.32.

Tabela I.32 – Valores de Y para cálculo do Baud Rate.

SYNC	Modo	BRGH = 0	BRGH = 1
0	Assíncrono	Y = 64	Y = 16
1	Síncrono	Y = 4	NA

15 Memória EEPROM interna

A memória EEPROM de um microcontrolador é extremamente útil para armazenar informações e parâmetros que não devem ser perdidos quando o dispositivo é desenergizado, como a memória de um telefone celular ou o ajuste de cor e brilho em um televisor, por exemplo.

Os 128 bytes de memória EEPROM que o PIC 16F628 possui, não são acessados diretamente, mas indiretamente, por meio de quatro SFRs, a saber:

- EEADR
- EEDATA
- EECON1
- EECON2

15.1 Registrador EEADR

Nesse registrador deve ser especificado o endereço da EEPROM em que se quer ler ou escrever.

Tabela I.33 – Bits do registrador EEADR.

EEADR (09BH)							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
---	**EADR6**	**EADR5**	**EADR4**	**EADR3**	**EADR2**	**EADR1**	**EADR0**
U	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Endereço de Acesso a EEPROM							

É digno de nota, que esse registro possui 8 bits (Tabela I.33) e permite escolher 256 diferentes posições, embora o PIC 16F628 possua apenas as 128 primeiras posições implementadas (00H – 7FH).

15.2 Registrador EEDATA

Na operação de escrita na EEPROM, o registrador EEDATA deve conter a informação a ser armazenada. Já na operação de leitura, o dado lido é armazenado nesse registrador.

15.3 Registrador EECON1

O registro EECON1, cuja composição pode ser vista na Tabela I.34, controla as operações de escrita e leitura na EEPROM.

Tabela I.34 – Bits do registrador EECON1.

EECON1 (018H)							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
---	**---**	**---**	**---**	**WRERR**	**WREN**	**WR**	**RD**
U-0	U-0	U-0	U-0	R/W-x	R/W-0	R-0	R-0

WRERR Indica erro na operação de escrita na EEPROM.

0 = Operação concluída com sucesso. 1 = Operação encerrada prematuramente (possivelmente por um reset).

WREN Permissão de escrita na EEPROM.

0 = Não permite escrita na EEPROM. 1 = Permite escrita na EEPROM.

WR Acionamento de escrita na EEPROM. Esse bit pode apenas ser setado pelo software, sendo zerado somente pelo hardware. O bit WR permanece em estado lógico “1” do início ao término da operação de escrita, quando então é posto em nível lógico “0”, automaticamente pela CPU.

0 = Indica que o ciclo de escrita foi concluído. 1 = Inicia operação de escrita na EEPROM.

RD Acionamento de leitura da EEPROM. O bit RD pode somente ser setado pelo software, sendo zerado apenas pelo hardware. Esse bit permanece em estado lógico “1” do início ao término da operação de leitura, quando então é posto em nível lógico “0”, automaticamente pela CPU.

0 = Indica que o ciclo de leitura foi concluído.
1 = Inicia operação de leitura da EEPROM.

15.4 Registrador EECON2

Este registrador é utilizado na inicialização do processo de escrita na EEPROM, por questão de segurança, e evita que o conteúdo da EEPROM seja alterado ou que nela ocorra acidentalmente uma escrita indesejada.

Para que algo possa ser escrito na EEPROM, deve-se primeiro armazenar o valor 55H e depois o valor AAH no registrador EECON2. Somente após ser efetuado isso, é que se pode executar a operação de escrita na EEPROM.

16 Palavra de Configuração

Os PICs da série 16 (MICROCHIP, 1997), possuem no endereço 2007H, após o final da memória de programa, um conjunto de bits que possibilitam o usuário configurar aspectos do modo de trabalho do microcontrolador. Esses bits, mostrados na Tabela I.35, são acessíveis somente no momento da gravação ou verificação do dispositivo. É importante salientar que o programador não precisa se preocupar com essa posição de memória, uma vez que os sistemas de gravação permitem a escolha das opções desejadas, como mostra a Figura I.6.

Tabela I.35 – Bits de configuração de aspectos do modo de trabalho do PIC.

Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CP1	CP0	CP1	CP0	---	CPD	LVP	BODEN	MCLRE	FOSC2	PWRTE	WDTE	FOSC1	FOSC0

CP1 Bit de proteção contra leitura de código (Tabela I.36).

CP0 Bit de proteção contra leitura de código (Tabela I.36).

Tabela I.36 – Proteção da memória de programa.

CP1	CP1	CP0	CP0	Área protegida da memória de programa
0	0	0	0	0000H – 07FFH
0	0	1	1	0200H – 07FFH
1	1	0	0	0400H – 07FFH
1	1	1	1	Proteção desabilitada

CPD Proteção contra leitura externa da EEPROM.

0 = Proteção desabilitada.

1 = Proteção habilitada.

LVP Bit de habilitação de programação com baixa tensão. A programação com baixa tensão é feita aplicando-se temsão no terminal RB4/PGM. Já a programação sem baixa tensão, é realizada com a aplicação de 13V no terminal RA5/THV/ MCLR\.

0 = Desabilita programação com baixa tensão.

1 = Habilita programação com baixa tensão.

BODEN Seleciona reset por Brown-out.

0 = Reset por Brown-out desabilitado.

1 = Reset por Brown-out habilitado.

MCLRE Bit de habilitação do MCLR.

0 = Terminal RA5/MCLR\ atua como entrada digital.

1 = Terminal RA5/MCLR\ atua como entrada de reset.

Tabela I.37 – Configuração do oscilador interno.

FOSC2	FOSC1	FOSC0	Tipo de Oscilador
1	1	1	ER
1	1	0	ER
1	0	1	INTRC
1	0	0	INTRC
0	1	1	EC
0	1	0	HS
0	0	1	XT
0	0	0	LP

FOSC2 Seleção do oscilador (Tabela I.37).

PWRTE Habilitação do temporizador de power-up (PWRT).

0 = Desabilita temporizador. 1 = Habilita temporizador.

WDTE Habilita Watchdog (WDT).

0 = Watchdog desabilitado. 1 = Watchdog habilitado.

FOSC1 Seleção do oscilador (Tabela I.37).

FOSC0 Seleção do oscilador (Tabela I.37).

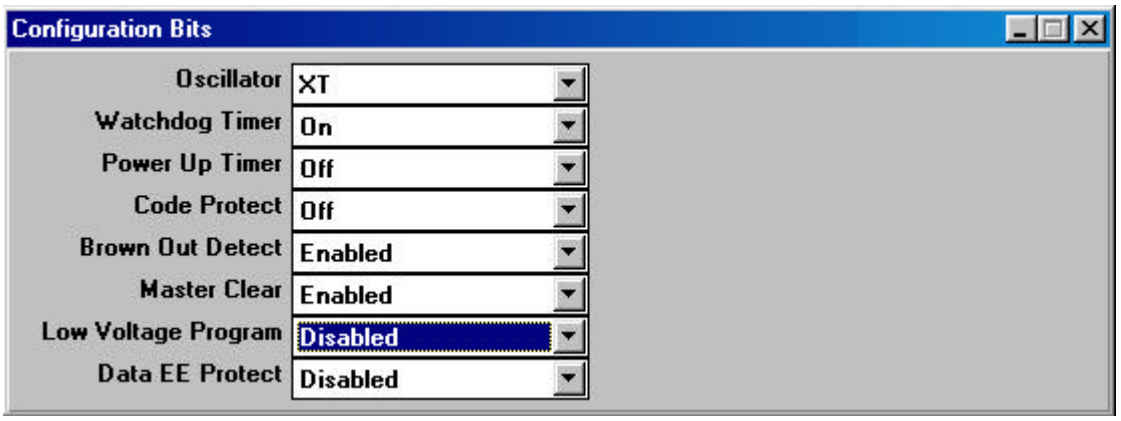


Figura I.6 – Formatação da Palavra de Configuração antes da gravação do chip.

17 Configuração do Oscilador do PIC 16F628

Os microcontroladores PIC 16F628 possuem um oscilador interno que pode funcionar de oito modos diferentes, a saber:

- Cristal/Ressoador cerâmico de alta freqüência (até 20MHz);
- Oscilador RC interno sem saída de clock;
- Oscilador RC interno com saída de clock;
- Cristal/Ressoador cerâmico (até 4MHz);

- Cristal de baixa potência (até 200KHz);
- Resistor externo com saída de clock;
- Resistor externo sem saída de clock;
- Clock externo.

17.1 MODO ER

Neste caso, um resistor com valor entre 38KO e 1MO, é conectado ao pino RA7, de acordo com a Figura I.7, e o oscilador é capaz de trabalhar com valores de 10 KHz até 8 MHz. O terminal RA6 pode ser configurado como I/O digital ou como saída de clock.

17.2 MODO EC

Conforme mostra a Figura I.8, no terminal OSC1 é injetado o sinal de clock proveniente de uma fonte externa autônoma, enquanto RA6 pode ser usado como I/O digital.

17.3 MODO INTRC

Neste modo, o clock da CPU provém de um circuito RC interno, com valor típico de 4 MHz (3,65 MHz – 4,28 MHz). A principal vantagem oferecida por este modo, além da econômica, é a liberação dos terminais RA6 e RA7, que podem ser empregados como I/Os digitais.

17.4 MODOS LP, XT E HS

A Figura I.9 mostra como utilizar um ressoador cerâmico ou um cristal de quartzo junto ao PIC 16F628. Os capacitores de desacoplamento, cujo valor da capacitância deve situar-se entre 15 e 33 pF (MICROCHIP, 1999), são usados para melhorar a estabilidade do oscilador. A configuração do modo do oscilador deve ser feita em função da freqüência, em consonância com a Tabela I.38.

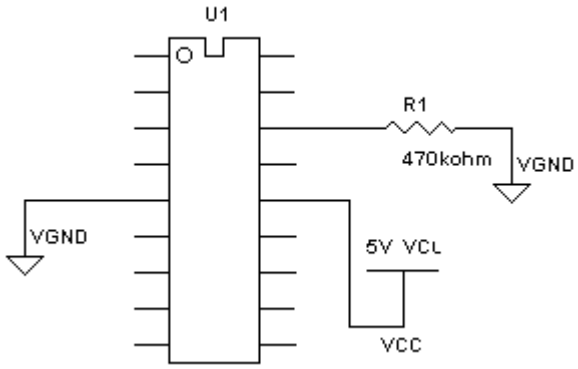


Figura I.7 – Modo ER.

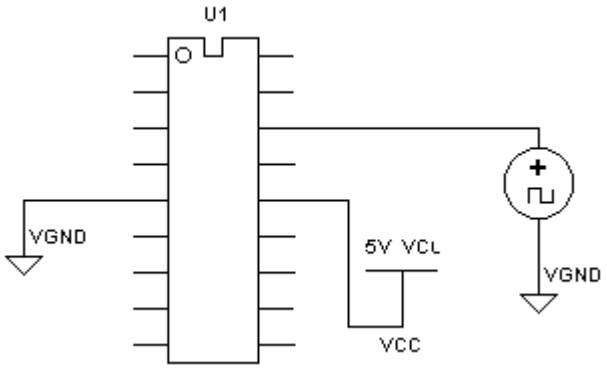


Figura I.8 – Modo EC.

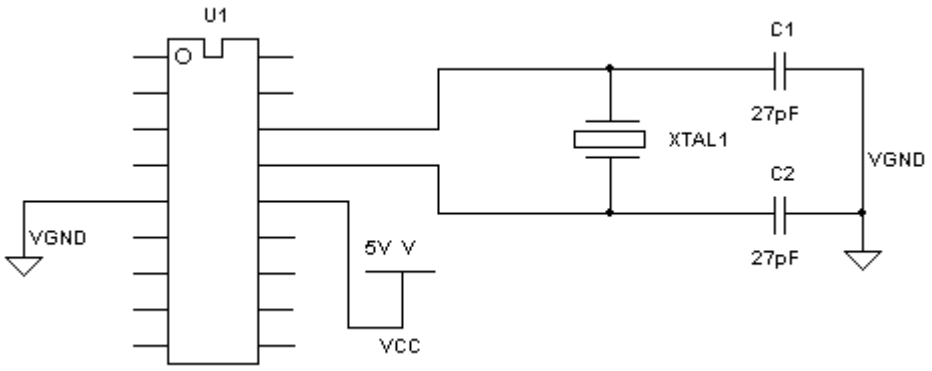


Figura I.9 – Modos LP, XT e HS.

Tabela I.38 – Configuração do oscilador com base na freqüência.

Modo	Freqüência	Observação
LP	35 a 250 KHz	Baixo consumo, indicado para sistemas alimentados por pilhas.
XT	100 KHz a 4 MHz	Usado na Implementação das arquiteturas descritas neste trabalho.
HS	> 4 MHz	Alta freqüência e consumo elevado

Em sistemas em que a precisão é requisito indispensável, como no caso das arquiteturas implementadas, deve-se optar pelo uso de cristais de quartzo.

ANEXO II

ARQUIVO INCLUDE PARA O PIC 16F628

```
                    LIST

; P16F628.INC  Standard Header File, Version 1.01    Microchip Technology, Inc.
      NOLIST

; This header file defines configurations, registers, and other useful bits of
; information for the PIC16F628 microcontroller.  These names are taken to match
; the data sheets as closely as possible.

; Note that the processor must be selected before this file is
; included.  The processor may be selected the following ways:

;       1. Command line switch:
;             C:\ MPASM MYFILE.ASM /PIC16F628
;       2. LIST directive in the source file
;             LIST   P=PIC16F628
;       3. Processor Type entry in the MPASM full-screen interface

;==========================================================================
;
;       Revision History
;
;==========================================================================

;Rev:   Date:    Reason:

;1.00   10 Feb 1999 Initial Release

;==========================================================================
;
;       Verify Processor
;
;==========================================================================

      IFNDEF __16F628
        MESSG "Processor-header file mismatch.  Verify selected processor."
      ENDIF

;==========================================================================
;
;       Register Definitions
;
;==========================================================================

W                                 EQU    H'0000'
F                                 EQU    H'0001'


;----- Register Files------------------------------------------------------

INDF                              EQU    H'0000'
TMR0                              EQU    H'0001'
PCL                               EQU    H'0002'
STATUS                            EQU    H'0003'
```

```
FSR                             EQU     H'0004'
PORTA                           EQU     H'0005'
PORTB                           EQU     H'0006'
PCLATH                          EQU     H'000A'
INTCON                          EQU     H'000B'
PIR1                            EQU     H'000C'
TMR1L                           EQU     H'000E'
TMR1H                           EQU     H'000F'
T1CON                           EQU     H'0010'
TMR2                            EQU     H'0011'
T2CON                           EQU     H'0012'
CCPR1L                          EQU     H'0015'
CCPR1H                          EQU     H'0016'
CCP1CON                         EQU     H'0017'
RCSTA                           EQU     H'0018'
TXREG                           EQU     H'0019'
RCREG                           EQU     H'001A'
CMCON                           EQU     H'001F'

OPTION_REG                      EQU     H'0081'
TRISA                           EQU     H'0085'
TRISB                           EQU     H'0086'
PIE1                            EQU     H'008C'
PCON                            EQU     H'008E'
PR2                             EQU     H'0092'
TXSTA                           EQU     H'0098'
SPBRG                           EQU     H'0099'
EEDATA                          EQU     H'009A'
EEADR                           EQU     H'009B'
EECON1                          EQU     H'009C'
EECON2                          EQU     H'009D'
VRCON                           EQU     H'009F'

;----- STATUS Bits --------------------------------------------------

IRP                             EQU     H'0007'
RP1                             EQU     H'0006'
RP0                             EQU     H'0005'
NOT_TO                          EQU     H'0004'
NOT_PD                          EQU     H'0003'
Z                               EQU     H'0002'
DC                              EQU     H'0001'
C                               EQU     H'0000'

;----- INTCON Bits --------------------------------------------------

GIE                             EQU     H'0007'
PEIE                            EQU     H'0006'
T0IE                            EQU     H'0005'
INTE                            EQU     H'0004'
RBIE                            EQU     H'0003'
T0IF                            EQU     H'0002'
INTF                            EQU     H'0001'
RBIF                            EQU     H'0000'
```

```
;----- PIR1 Bits ----------------------------------------------------

EEIF                            EQU     H'0007'
CMIF                            EQU     H'0006'
RCIF                            EQU     H'0005'
TXIF                            EQU     H'0004'
CCP1IF                          EQU     H'0002'
TMR2IF                          EQU     H'0001'
TMR1IF                          EQU     H'0000'

;----- T1CON Bits ---------------------------------------------------

T1CKPS1                         EQU     H'0005'
T1CKPS0                         EQU     H'0004'
T1OSCEN                         EQU     H'0003'
NOT_T1SYNC                      EQU     H'0002'
TMR1CS                          EQU     H'0001'
TMR1ON                          EQU     H'0000'

;----- T2CON Bits ---------------------------------------------------

TOUTPS3                         EQU     H'0006'
TOUTPS2                         EQU     H'0005'
TOUTPS1                         EQU     H'0004'
TOUTPS0                         EQU     H'0003'
TMR2ON                          EQU     H'0002'
T2CKPS1                         EQU     H'0001'
T2CKPS0                         EQU     H'0000'

;----- CCP1CON Bits -------------------------------------------------

CCP1X                           EQU     H'0005'
CCP1Y                           EQU     H'0004'
CCP1M3                          EQU     H'0003'
CCP1M2                          EQU     H'0002'
CCP1M1                          EQU     H'0001'
CCP1M0                          EQU     H'0000'

;----- RCSTA Bits ---------------------------------------------------

SPEN                            EQU     H'0007'
RX9                             EQU     H'0006'
SREN                            EQU     H'0005'
CREN                            EQU     H'0004'
ADEN                            EQU     H'0003'
FERR                            EQU     H'0002'
OERR                            EQU     H'0001'
RX9D                            EQU     H'0000'

;----- CMCON Bits ---------------------------------------------------

C2OUT                           EQU     H'0007'
C1OUT                           EQU     H'0006'
C2INV                           EQU     H'0005'
```

```
C1INV                           EQU     H'0004'
CIS                             EQU     H'0003'
CM2                             EQU     H'0002'
CM1                             EQU     H'0001'
CM0                             EQU     H'0000'


;----- OPTION Bits --------------------------------------------------

NOT_RBPU                        EQU     H'0007'
INTEDG                          EQU     H'0006'
T0CS                            EQU     H'0005'
T0SE                            EQU     H'0004'
PSA                             EQU     H'0003'
PS2                             EQU     H'0002'
PS1                             EQU     H'0001'
PS0                             EQU     H'0000'

;----- PIE1 Bits ----------------------------------------------------

EEIE                            EQU     H'0007'
CMIE                            EQU     H'0006'
RCIE                            EQU     H'0005'
TXIE                            EQU     H'0004'
CCP1IE                          EQU     H'0002'
TMR2IE                          EQU     H'0001'
TMR1IE                          EQU     H'0000'

;----- PCON Bits ----------------------------------------------------

OSCF                            EQU     H'0003'
NOT_POR                         EQU     H'0001'
NOT_BO                          EQU     H'0000'
NOT_BOR                         EQU     H'0000'
NOT_BOD                         EQU     H'0000'

;----- TXSTA Bits ---------------------------------------------------

CSRC                            EQU     H'0007'
TX9                             EQU     H'0006'
TXEN                            EQU     H'0005'
SYNC                            EQU     H'0004'
BRGH                            EQU     H'0002'
TRMT                            EQU     H'0001'
TX9D                            EQU     H'0000'

;----- EECON1 Bits --------------------------------------------------

WRERR                           EQU     H'0003'
WREN                            EQU     H'0002'
WR                              EQU     H'0001'
RD                              EQU     H'0000'
```

```
;----- VRCON Bits --------------------------------------------------------

VREN                            EQU     H'0007'
VROE                            EQU     H'0006'
VRR                             EQU     H'0005'
VR3                             EQU     H'0003'
VR2                             EQU     H'0002'
VR1                             EQU     H'0001'
VR0                             EQU     H'0000'

;==========================================================================
;
;       RAM Definition
;
;==========================================================================

    __MAXRAM H'01FF'
    __BADRAM H'07'-H'09', H'0D', H'13'-H'14', H'1B'-H'1E'
                    __BADRAM H'87'-H'89', H'8D', H'8F'-H'91', H'93'-H'97', H'9E'
                    __BADRAM H'105', H'107'-H'109', H'10C'-H'11F', H'150'-H'16F'
                    __BADRAM H'185', H'187'-H'189', H'18C'-H'1EF'

;==========================================================================
;
;       Configuration Bits
;
;==========================================================================

_BODEN_ON                       EQU     H'3FFF'
_BODEN_OFF                      EQU     H'3FBF'
_CP_ALL                         EQU     H'03FF'
_CP_75                          EQU     H'17FF'
_CP_50                          EQU     H'2BFF'
_CP_OFF                         EQU     H'3FFF'
_PWRTE_OFF                      EQU     H'3FFF'
_PWRTE_ON                       EQU     H'3FF7'
_WDT_ON                         EQU     H'3FFF'
_WDT_OFF                        EQU     H'3FFB'
_LVP_ON                         EQU     H'3FFF'
_LVP_OFF                        EQU     H'3F7F'
_MCLRE_ON                       EQU     H'3FFF'
_MCLRE_OFF                      EQU     H'3FDF'
_ER_OSC_CLKOUT                  EQU     H'3FFF'
_ER_OSC_NOCLKOUT                EQU     H'3FFE'
_INTRC_OSC_CLKOUT               EQU     H'3FFD'
_INTRC_OSC_NOCLKOUT             EQU     H'3FFC'
_EXTCLK_OSC                     EQU     H'3FEF'
_LP_OSC                         EQU     H'3FEC'
_XT_OSC                         EQU     H'3FED'
_HS_OSC                         EQU     H'3FEE'

                    LIST
```