UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
UNIDADE ACADÊMICA DE ENGENHARIA ELÉTRICA

LUIS GUSTAVO GUEDES PEREIRA DE CASTRO

SISTEMA PARA IMPLEMENTAÇÃO DE FALHAS INTERMITENTES NO CHAVEAMENTO DE INTERRUPTORES

TRABALHO DE CONCLUSÃO DE CURSO

CAMPINA GRANDE-PB
FEVEREIRO-2009

LUIS GUSTAVO GUEDES PEREIRA DE CASTRO

SISTEMA PARA IMPLEMENTAÇÃO DE FALHAS INTERMITENTES NO CHAVEAMENTO DE INTERRUPTORES

Trabalho de Conclusão de Curso submetida à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para obtenção da graduação em Engenharia Elétrica.

Orientador
Prof. Talvanes Meneses Oliveira

CAMPINA GRANDE - PB

2009

LUIS GUSTAVO GUEDES PEREIRA DE CASTRO

SISTEMA PARA IMPLEMENTAÇÃO DE FALHAS INTERMITENTES NO CHAVEAMENTO DE INTERRUPTORES

Trabalho de Conclusão de Curso aprovado como requisito parcial necessário para a obtenção da graduação em Engenharia Elétrica pela Universidade Federal de Campina Grande.

Data de Aprovação: 13 de fevereiro de 2009

BANCA EXAMINADORA:

_____

Prof. Talvanes Meneses Oliveira
Universidade Federal de Campina Grande
Orientador

_____

Prof. Eurico Bezerra de Souza Filho
Universidade Federal de Campina Grande

Agradecimentos

Ao meu orientador prof. Talvanes Meneses Oliveira, pela oportunidade de trabalho. A Euler Macedo pelo auxilio durante o projeto e à Kamilla Montenegro pela ajuda na revisão deste relatório.

Aos amigos pelo incentivo e apoio, e a todos que de alguma forma colaboraram com este trabalho.

Resumo

Este trabalho tem como objetivo o desenvolvimento de uma rotina para gerar falhas no chaveamento de comando de interruptores. Para tanto, foi utilizado o microcontrolador PIC18F452 para gerar sinal de comando e controlar o momento de ocorrência da falha e sua duração. O sistema foi montado em laboratório podendo auxiliar no estudo sobre falhas no comando de interruptores.

Palavras-chave: Microcontroladores, PIC18F452, Falhas no chaveamento de comando.

Abstract

This work has as objective the development of a routine to generate flaws on the switching of power switches. For that, the PIC18F452 microcontroller was used to generate the flaws and to control the moment it will occur and the time it will take. The system was fit up in laboratory and could be used to assist the study of flaws on the switching of power switches.

Key Words: Microcontrollers, PIC18F452, Flaws on the switching of power switches

Lista de Figuras

Lista de Tabelas

Capítulo 1 – Introdução

O objetivo deste trabalho é adquirir conhecimentos sobre os microcontroladores PIC e desenvolver uma rotina para implementar um sistema de falhas no chaveamento de comando de interruptores.

Os microcontroladores PIC surgiram na década de 80, estão divididos nas famílias de 8-bits, 16-bits e 32-bits e atualmente é uma ferramenta importante no projeto de sistemas de controle.

O sistema para geração de falhas a ser desenvolvido será controlado pelo microcontrolador PIC18F452 e permitirá ao usuário especificar a frequência de chaveamento, o momento e duração da falha.

1.1 – Organização do Trabalho

Este trabalho está organizado em quatro capítulos. No capítulo 1 encontra-se a introdução e organização do trabalho. No capítulo 2 encontra-se uma introdução sobre as diferentes famílias de microcontroladores PIC, assim como, uma revisão sobre a arquitetura e principais recursos do PIC18F452.

No capítulo 3 é apresentado o sistema para implementação de falhas intermitentes no chaveamento de comando de interruptores e sua rotina de controle é descrita.

No capítulo 4 é apresentado o sistema, o material utilizado no experimento para aquisição e os resultados.

As considerações finais e sugestões para trabalhos futuros estão no capítulo 5. O trabalho ainda possui um anexo onde se encontra a rotina desenvolvida.

Capítulo 2 – Microcontroladores PIC

2.1 – Introdução

A Microchip Technology Inc., fabricante dos microcontroladores PIC, surgiu na década de 80 e tornou-se, ao longo do tempo, uma das maiores fornecedoras de microcontroladores do mundo. Os microcontroladores PIC (*Programmable Interface Controller*) possuem arquitetura Harvard-RISC modificada e estão disponíveis no mercado de acordo com diferentes especificações como: memória, número de pinos, periféricos e funções especiais.

O número de periféricos é uma das características que mais diferencia os microcontroladores. Os principais periféricos dos PIC são: Timers, Conversores A/D, USART, Módulo CCP (Capture/Compare/PWM), Entrada/saída digital, Comparadores, Porta síncrona serial (SSP), I^2C, etc. Alguns PIC´s também possuem funções especiais como: USB, CAN, Ethernet, módulo para controle de motores, módulo para processamento de áudio, etc.

Nesse capítulo serão introduzidas as diferentes famílias de microcontroladores PIC, sendo em seguida realizado um estudo sobre os PICs de 8-bits e os periféricos importantes para o desenvolvimento da rotina de controle. O estudo realizado neste trabalho é focado no microcontrolador PIC18F452.

2.2 – Famílias

Desde 2007, com o lançamento do PIC32, a família de microcontroladores PIC pode ser subdividida em três, como ilustrado na figura 2.1.

Na família de 32-bits (PIC32) estão os PIC de maior desempenho e maior funcionalidade. Com frequência de operação de até 80MHz. Dispositivos com até 100 pinos e tamanho reduzido, os PIC32 são ideais para sistemas embarcados complexos. Suas principais características são:

- Operação em até 80MHz, 1.56DMIPS;
- Até 512KB de memória Flash;
- Até 32KB de memória RAM;

- 4 controladores DMA;
- Até 85 portas de entrada/saída digital;
- Operação em 3.3V, pinos de entrada tolerantes a 5V
- Conversor A/D de 10-bits;
- Módulos Capture/Compare/PWM;
- USB 2.0 OTG;
- Comunicação serial: USART, I2C, SPI;

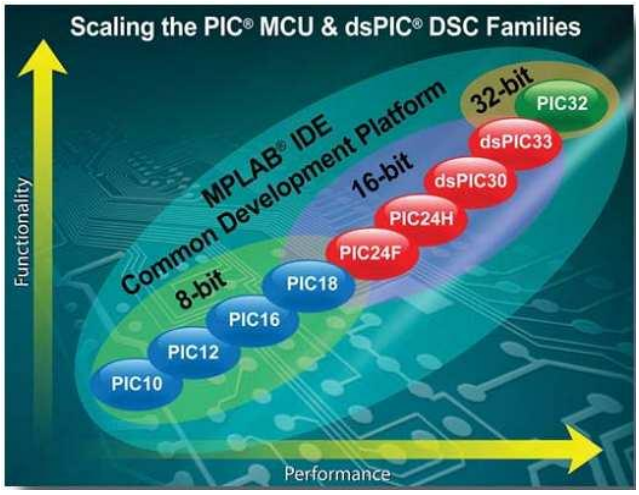

Figura 2.1: Famílias de Microcontroladores PIC

Os PIC24 e os dsPIC formam a família de controladores de 16-bits. Alguns dispositivos dessa família operam em até 80MHz e possuem módulos para controle de motor e processamento de áudio. Suas principais características são:

- Operação em até 80 MHz, 40MIPS;
- Até 512KB de memória Flash;
- Até 32KB de memória RAM;
- Até 85 portas de entrada/saída digital;
- Controlador DMA;
- RTC interno;
- Conversores A/D de 10/12 bits;
- Conversores D/A de 12/16 bits;
- Módulo PWM de 10/16 bits para controle de motor;
- Interface para encoder;

- USB 2.0;
- Até 9 timers/counters;
- Módulo Capture/Compare/PWM;
- Comunicação serial: USART, I2C, SPI.

A família de 8-bits pode ser subdividida em três categorias de acordo com o tamanho da palavra de instrução: Baseline, Mid-Range e High-Performance. A figura 2.2 ilustra a divisão dos PIC de 8-bits.

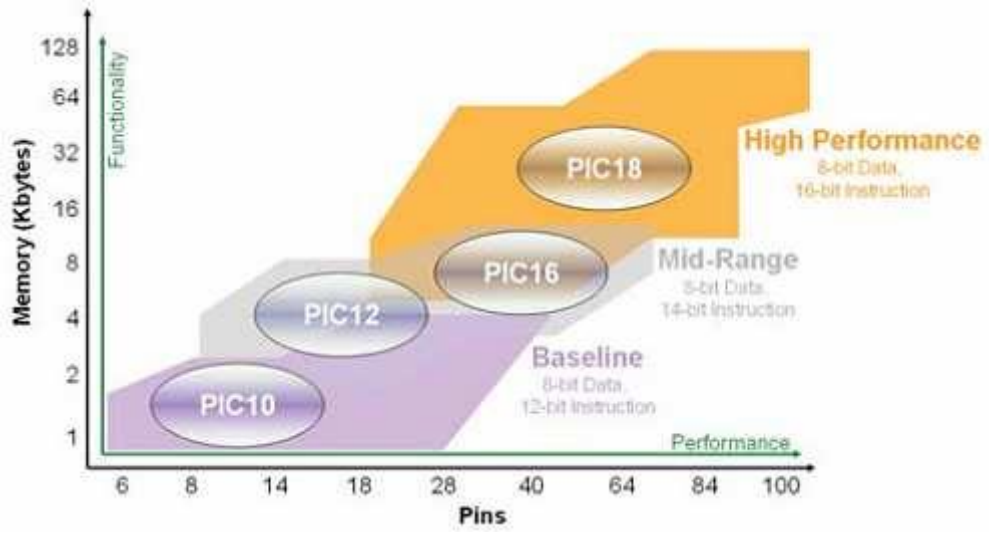

Figura 2.2: Categorias dos microcontroladores PIC de 8-bits

O PIC10, o PIC12 e alguns PIC16 formam a categoria Baseline. Eles possuem palavra de instrução de 12-bits e são os microcontroladores de menor custo e desempenho da Microchip.

A categoria Mid-Range é formada pelos PIC12 e PIC16, que possuem palavra de instrução de 14-bits, memória de até 14KB e diversos periféricos integrados. Na categoria Mid-Range podemos destacar os chamados Enhanced Mid-Range que possuem maior desempenho e maior capacidade de armazenamento que os Mid-Range comuns.

Os PICs com instrução de 16-bits formam a categoria High-Performance (PIC18). Os PIC18 são os microcontroladores de 8-bits de maior desempenho e com maior número de funções da Microchip. Possuem módulos especiais como CAN, ETHERNET, USB e Touch Sensing. Abaixo temos um quadro com as principais características de cada categoria:

	Baseline	Mid-Range	Enhanced Mid-Range	High-Performance
Nº de Pinos	6 – 40	8 – 64	8 – 64	18 – 100
Interrupções	não possui	Capacidade de Interrupção simples	Capacidade de Interrupção simples	Multiplas Interrupções
Desenpenho	5 MIPS	5 MIPS	8 MIPS	10 – 16 MIPS
Instruções	33, 12-bit	35, 14-bit	49, 14-bit	75 - 83, 16-bit
Memória de Programa	Até 3 KB	Até 14 KB	Até 56 KB	Até 128 KB
Memória de Dados	Até 138 bytes	Até 368 bytes	Até 4 KB	Até 4 KB
Recursos	•Tamanho reduzido •Baixo custo •Ideal para uso com bateria •Facil de aprender e usar	•Ótimo custo-benefício •Periféricos Integrados, incluindo: SPI, I2C, UART, LCD,ADC.	•Cógico em C otimizado •Mapa de memória simplificado •Endereçamento Indireto otimizado •Latência de interrupção reduzida	•Hardware multiplicador 8x8 •Código em C otimizado; •Periféricos avançados: CAN, USB, Ethernet,Touch Sensing, LCD.

Tabela 1: Comparação entre as famílias de 8-bits

2.3 – Arquitetura

Os microcontroladores PIC apresentam arquitetura Harvard, onde a memória de programa e a memória de dados estão separadas e são acessadas por barramentos diferentes, o que representa um aumento de velocidade comparado a arquitetura von-Neumann (figura 2.3), onde as memórias são acessados por um único barramento. A distinção entre os barramentos possibilita que as instruções sejam representadas por uma palavra maior que 8 bits (microcontroladores de 8-bits). Nos PICs de 8 bits, essa palavra pode ser de 12, 14 ou 16 bits, o que permite que todas as instruções ocupem uma só palavra.

As características que fazem do PIC um microcontrolador RISC (*Reduced Instruction Set Computer*) são:

- Arquitetura Harvard;
- Instruções em uma única palavra;
- Instruções executadas em um único ciclo de máquina;
- Pipeline de instruções;
- Número reduzido de instruções;

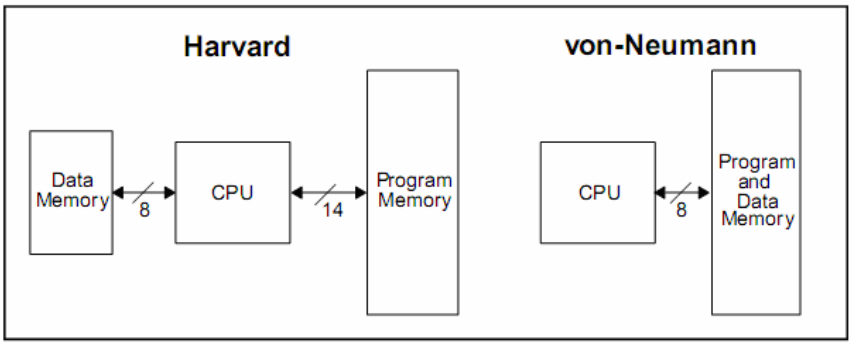

Figura 2.3: Arquiteturas Harvard versus Von-Neumann

2.3.1 – Ciclo de Instrução

O clock (relógio) de entrada do PIC é dividido internamente em quatro fases (Q1, Q2, Q3 e Q4) que não se sobrepõem. Internamente, o contador de programa, que armazena o endereço da próxima instrução a ser executada, é incrementado em Q1, quando a instrução é buscada da memória de programa e armazenada no registrador de instruções em Q4. A mesma instrução é decodificada e executada durante o próximo ciclo de Q1 a Q4. A figura 2.4 ilustra a relação entre o clock de entrada (OSC1), as fases Q1-Q4 e o contador de programa (PC):

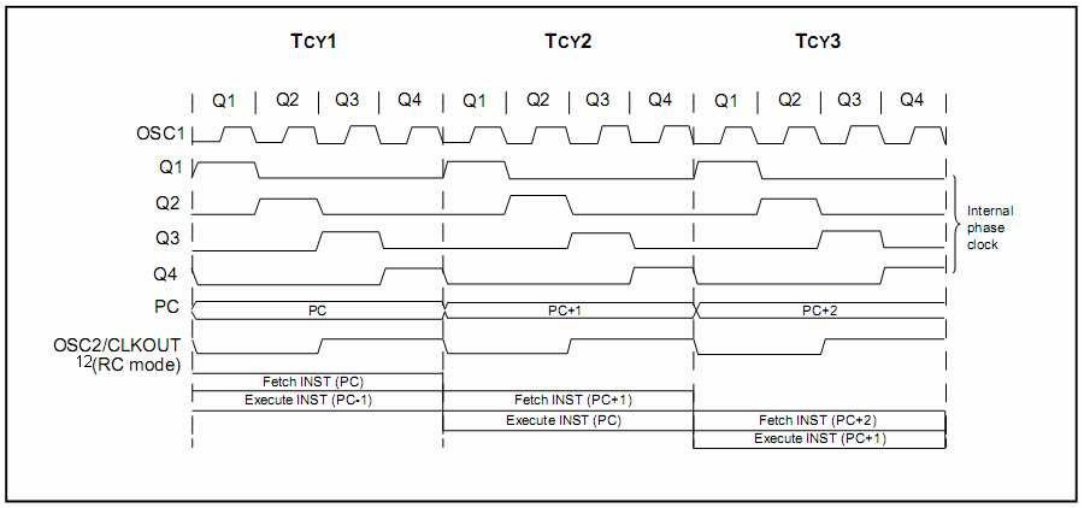

Figura 2.4: Ciclo de instruções do PIC

2.3.2 – Pipeline

Um ciclo de instrução possui as quatro fases Q1, Q2, Q3 e Q4. A busca de uma instrução é realiza em um ciclo, enquanto sua decodificação e execução são realizas no ciclo seguinte. Entretanto, devido ao pipeline, cada instrução é efetivamente executada em apenas um ciclo. Com o pipeline, ao mesmo tempo em que se executa uma instrução, o microcontrolador busca na memória o código da instrução seguinte. Se uma instrução modificar o valor de PC (ex: saltos), um ciclo extra é necessário para completar a instrução. Na fase Q1 a instrução é escrita no registrador de instruções (Instruction Register-IR), nos ciclos Q2, Q3 e Q4 ela é decodificada e executada. Na figura 2.5 é ilustrando o princípio de operação do pipeline do PIC, enquanto a instrução 1 está sendo executada, a instrução 2 está sendo buscada na memória e será executada no próximo ciclo.

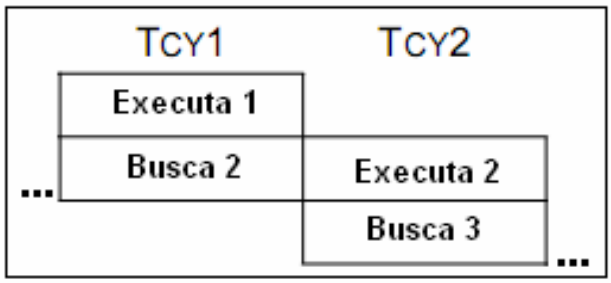

Figura 2.5: Pipeline do PIC

2.3.3 – Unidade Central de Processamento (CPU)

A CPU é o cérebro do microcontrolador. Ela é responsável por buscar a instrução correta, decodificá-la e executá-la. A CPU controla o barramento de endereços das memórias de programa e de dados, e endereços da pilha.

2.3.4 – Unidade Lógica e Aritmética (ULA)

Os microcontroladores PIC possuem ULA de 8-bits e um registrador de trabalho (*W register*) também de 8-bits. A ULA é capaz de realizar as operações aritméticas de soma e subtração, assim como as operações lógicas de

deslocamentos, *and*, *or*, *xor*, etc. Em operações que envolvem dois operandos, um deles é sempre o registrador *W* e o outro é um registrador de arquivo (*f register*) ou uma constante. Em operações com um único operando o registrador *W* ou um dos registradores de arquivo é utilizado. O registrador *W* não é endereçável e, dependendo da instrução executada, pode afetar o valor dos bits *Carry* (C), *Digit Carry* (DC) e *Zero* (Z) do registrador de Status (*STATUS Register*). O PIC18F452 possui ainda os bits *Overflow* (OV) e *Negative* (N).

Nos PIC18 um hardware multiplicador 8x8 está presente na ULA, tornando a multiplicação uma operação de hardware que pode ser executada em apenas um ciclo de instrução. O resultado da operação de multiplicação é armazenado no par de registradores (*PRODH:PRODL*) e não afeta o registrador de Status. O multiplicador aumenta o desempenho do microcontrolador e reduz o tamanho de código para operações de multiplicação.

2.3.5 – Registrador de Status

O Registrador de Status, figura 2.6, armazena o estado das operações aritméticas realizadas na ULA. Os bits desse registrador são “setados” ou apagados de acordo com o resultado da operação.

REGISTRADOR DE STATUS

U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
—	—	—	N	OV	Z	DC	C
bit 7							bit 0

Figura 2.6: Registrador de Status

2.3.6 – Registrador de Controle de Reset

O *RCON* (*Reset Controle Register*), figura 2.7, contém os bits (*flags*) que permitem diferenciar entre as fontes de reset do dispositivo. Podemos ler ou escrever no *RCON* e seus flags incluem os bits:

- $\overline{TO}$: Transbordo do Watchdog (Watchdog Timer-out);
- $\overline{PD}$: Detecção de queda de tensão (Power-down detection);
- $\overline{POR}$: Reset ao iniciar (Power-on Reset);

- $\overline{BOR}$: Queda de tensão abaixo de um valor pré-determinado (Brown-out Reset);
- $\overline{RI}$: Instrução de RESET;
- $IPEN$: Habilita prioridade de interrupções;

RCON

R/W-0	U-0	U-0	R/W-1	R-1	R-1	R/W-0	R/W-0
IPEN	—	—	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$
bit 7							bit 0

Figura 2.7: Registrador RCON

2.3.7 – Organização da memória

OS PIC18 possuem três blocos de memória separados, memória de programa (Flash), memória de dados RAM e memória de dados EEPROM. As memórias de dados e de programa são acessadas por barramentos distintos.

2.3.7.1 – Memória de programa

O PIC18F452 possui uma memória de programa de 32kB do tipo Flash. As memórias flash tornam possível a reprogramação do microcontrolador por diversas vezes de maneira simples.

Todos os PIC18F possuem um contador de programa (PC) de 21-bits, que são capazes de endereçar 2Mbytes de memória. O espaço de memória utilizado pelo PIC18F452 vai de 00000h até 7FFFh. Caso uma posição de memória não existente seja acessada (de 8000h até 1FFFFFh), um conjunto de zeros (000000h) é lido. O vetor de reset está endereçado em 0000h e os endereços 0008h e 0018h são reservados para os vetores de interrupção de alta e baixa prioridade, respectivamente. A figura abaixo ilustra a memória de programa do PIC18F452.

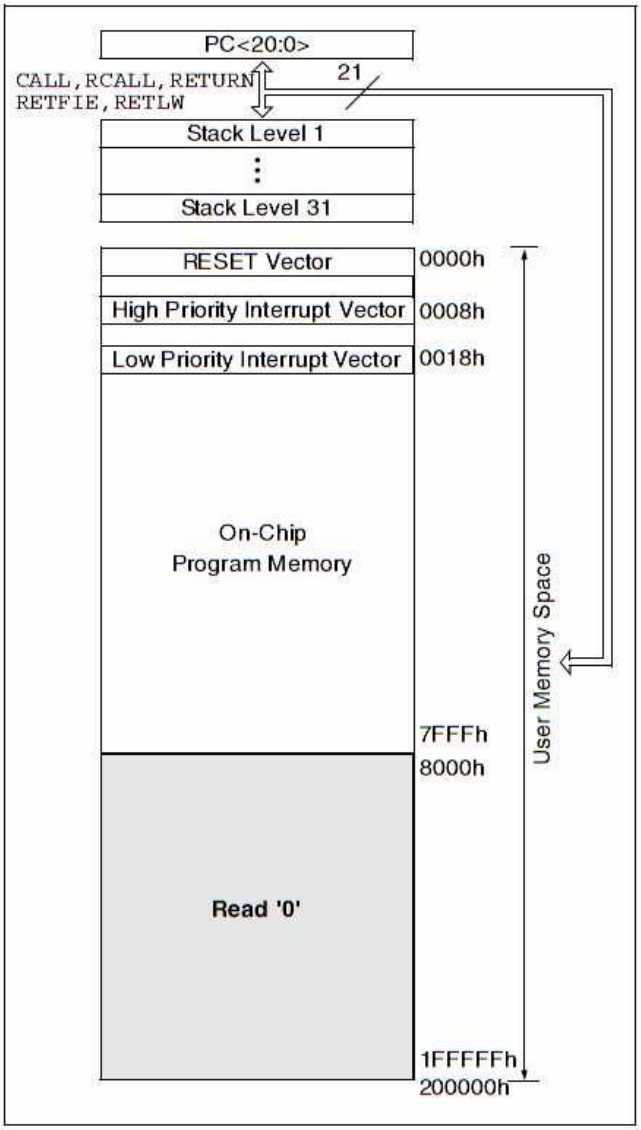

Figura 2.8: Mapa de memória do PIC18F452

A pilha (*stack*) nos PIC18F possui 31 posições de 21-bits e é responsável por armazenar o valor do contador de programa quando ocorre um salto do programa principal para uma subrotina. A pilha é controlada pelo registrador *STKPTR* e tem seu valor inicializado com 00000h após cada reset do microcontrolador.

2.3.7.2 – Memória de Dados

A memória de dados do PIC é formada por uma memória EEPROM e uma memória RAM. A memória EEPROM do PIC18F452 possui 256 bytes e não está mapeada diretamente no espaço de memória. Ela pode ser acessada

indiretamente através dos registradores *EECON1*, *EECON2, EEDATA* e *EEADR.* A *EEPROM* é utilizada para armazenar dados que devem ser mantidos mesmo após o desligamento do dispositivo.

A RAM é utilizada para armazenar temporariamente dados e resultados intermediários criados durante a operação do microcontrolador. Cada registro da RAM possui um endereço de 12-bits, possibilitando até 4096 bytes de memória de dados. O mapa de memória RAM é dividido em 16 bancos contendo 256 bytes cada. O Registrador de Seleção de Bancos (*Bank Select Register – BSR*) é utilizado para selecionar qual banco será acessado. A RAM é formada pelos Registradores de Propósito Geral (*General Purpose Registers – GPR*) e pelos Registradores de Funções Especiais (*Special Funcion Registers-SFR*). Os SFRs são utilizados para controlar os periféricos do microcontrolador enquanto os GPRs armazenam dados. A figura 2.9 ilustra o Mapa da RAM do PIC18F452.

2.3.8 – Modos de Endereçamento

A memória RAM pode ser acessada de duas formas: diretamente ou indiretamente.

2.3.8.1 – Endereçamento Direto

No endereçamento direto um dos 16 bancos da RAM é selecionado através do registrador BSR. Os bits <3:0> do BSR armazenam os quatro bits mais significativos do endereço de 12-bits, e os bits <7:4> não são utilizados. Os outros 8-bits para endereçamento são obtidos do endereço da instrução.

2.3.8.2 – Endereçamento Indireto

No endereçamento indireto o endereço da instrução não está fixado. O endereço é obtido através dos registradores FSR (*File Select Register*) e INDF (*Indirect File Operand*). Os três FSRs do PIC18 são divididos em quatro registradores de 8-bits e são responsáveis por armazenar o endereço de 12-bits da memória. Eles são usados como ponteiro podendo ser lidos ou escritos. O

endereçamento direto é possível se utilizarmos um dos 3 registradores INDF. Qualquer instrução utilizando os registradores INDF acessa o registro apontado pelo FSR correspondente.

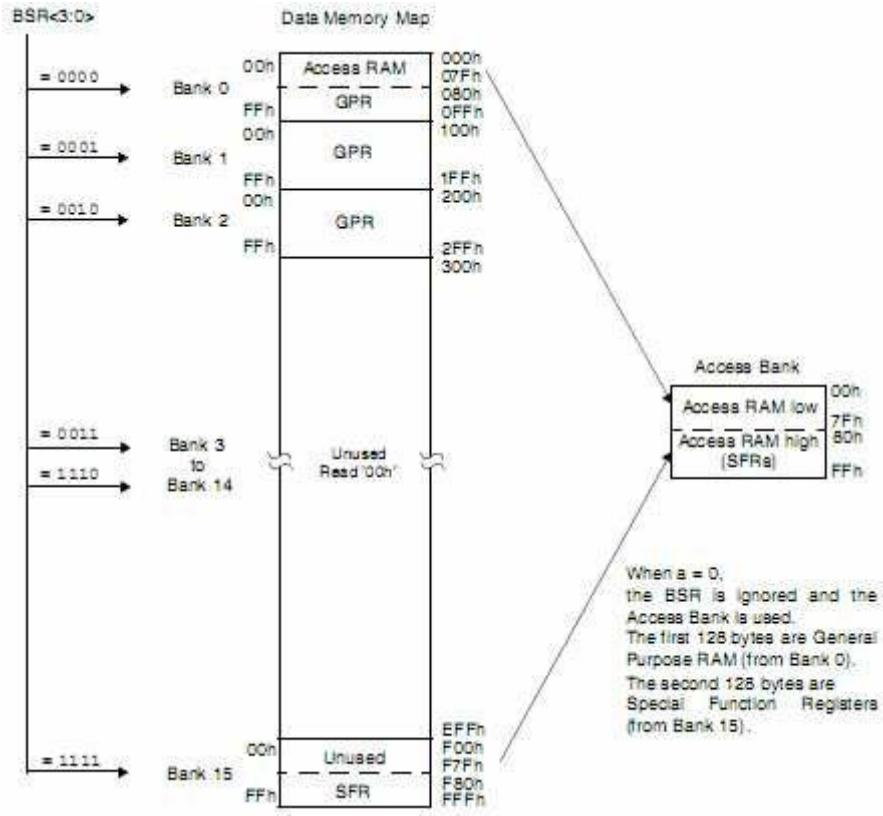

Figura 2.9: Mapa da Memória RAM do PIC18F452

2.3.9 – Interrupções

O PIC18F452 possui diversas fontes de interrupção e um recurso que possibilita especificar cada fonte de interrupção como sendo de alta ou baixa prioridade. O vetor de interrupções de alta prioridade está no endereço 000008h e o de baixa prioridade em 000018h. As interrupções marcadas como de alta prioridade podem ser atendidas quando a rotina de uma interrupção de baixa prioridade estiver em execução. As interrupções podem ser dividias em dois tipos: Interrupções externas e do Timer0, e Interrupções dos Periféricos. Dez registradores são utilizados para controlá-las:

- RCON;
- INTCON;

- INTCON2;
- INTCON3;
- PIR1, PIR2;
- PIE1, PIE2;
- IPR1, IPR2;

Os registradores INTCON e INTCON2 podem ser lidos ou escritos, ambos contêm os bits de ativação, de prioridade e flags das interrupções externas e do Timer 0. Para interrupção dos periféricos são usados os registradores PIR, PIE e IPR. Os registradores PIR contém os flags individuais, os PIE contém os bits de habilitação e os IPR os bits individuais de prioridade. O registrador RCON possui um bit que habilita o recurso de priorização de interrupções.

2.4 – O PIC18F452

O PIC18F452, figura 2.10, é um microcontrolador de 8-bits da Microchip que possui 75 instruções e palavra de instrução de 16-bits. Ele é enquadrado na categoria High-Performance e suas características principais são:

- 32KB de memória flash;
- 1526 bytes de memória RAM;
- 256 bytes de memória EEPROM;
- Tensão de operação entre 2.0V e 5.5V;
- Operação em até 10MIPS, 40 MHz;
- Oscilador com PLL ativo;
- Oscilador secundário;
- Hardware Multiplicador 8x8;
- Priorização de interrupções;
- 18 Fontes de Interrupção;
- 5 Portas de Entrada/Saída;
- 2 módulos Captura/Comparação/PWM;
- 3 pinos de interrupção externa;
- 4 Timers/Counters de 8/16 bits;

- Porta Serial Síncrona Mestre (MSSP) com SPI e I^2C;
- Módulo USART com suporte a: RS-485 e RS-232;
- 8 conversores A/D de 10-bits;
- Watchdog Timer, Power-up Timer, Proteção de código, SLEEP Mode, Programação In-Circuit, etc;

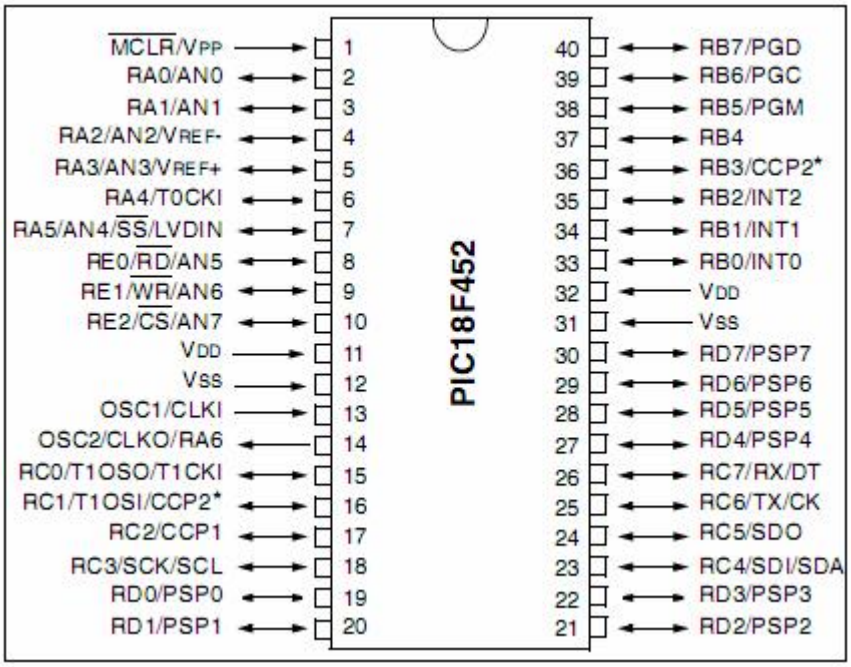

Figura 2.10: Diagrama de pinos do PIC18F452

2.4.1 – Oscilador

O oscilador do PIC18F452 pode operar em 8 modos diferentes. O modo de operação pode ser selecionado alterando o valor dos bits FOSC2, FOSC1 e FOSC0 no registrador de configuração CONFIG1H. Os modos são:

- LP: Cristal de baixa potência (*Low Power Crystal*);
- XT: Cristal ou Ressonador (*Crystal/Ressonator*);
- HS: Cristal/Ressonador de alta velocidade (*High Speed Crystal/Resonator*);
- HS+PLL: Cristal/Ressonador de alta velocidade com PLL habilitado (*High Speed Crystal/Resonator with PLL enabled*);
- RC: Resistor/Capacitor externo (*External Resistor/Capacitor*);

- RCIO: Resistor/Capacitor externo com pino de E/S habilitado (*External Resistor/Capacitor with I/O pin enabled*);
- EC: Clock externo (*External Clock*);
- ECIO: Clock externo com pino de E/S habilitado (*External Clock with I/O pin enabled*);

Nos modos LP, XT, HS ou HS+PLL, Figura 2.11, um cristal ou ressonador cerâmico é conectado aos pinos OSC1 e OSC2 para estabelecer a oscilação. No modo HS+PLL a frequência do cristal é multiplicada por 4, possibilitando a operação em 40MHz se o cristal utilizado for de 10MHz.

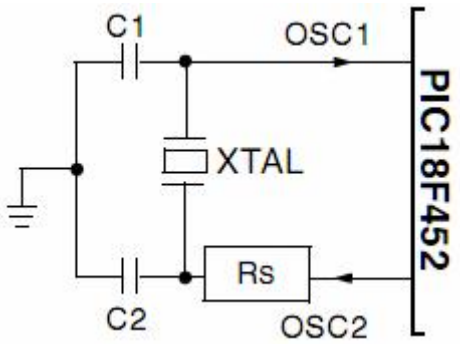

Figura 2.11: Oscilador no modo XT, HS ou LP

Os modos RC e RCIO utilizam um circuito RC como fonte de “clock” conectado ao pino OSC1. A frequência de “clock” é função da tensão de alimentação, do valor do resistor e do capacitor. Esse modo é utilizado em aplicações insensíveis ao tempo. No modo RC, o pino OSC2 fornece a frequência de oscilação dividida por 4. O modo RCIO é semelhante ao RC, porém, o pino OSC2 não fornece a frequência de oscilação e pode ser utilizado com entrada/saída. A figura 2.12 ilustra o oscilador no modo RC.

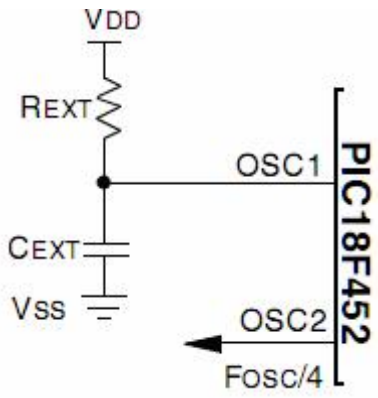

Figura 2.12: Oscilador no modo RC

Nos modos EC e ECIO um clock externo é a fonte de oscilação e deve ser conectado ao pino OSC1. No modo EC, assim como no RC, a frequência de entrada dividida por 4 está disponível no pino OSC2. No modo ECIO o pino OSC2 pode ser utilizado como entrada/saída.

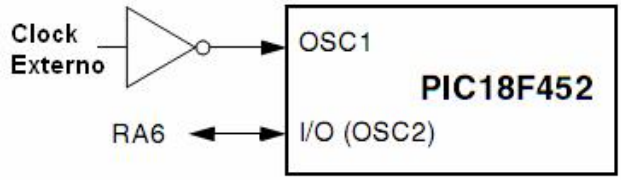

Figura 2.13: Oscilador no modo ECIO

2.4.2 – Portas de Entrada/Saída

O PIC18F452 possui 5 portas de entrada/saída. Alguns pinos das portas de E/S são multiplexados com funções dos periféricos do dispositivo. Quando um periférico está habilitado, esse pino não pode ser utilizado como E/S de propósito geral. Cada porta possui 3 registradores para sua operação. Esses registradores são:

- TRISx: Controla a direção dos dados (entrada ou saída);
- PORTx: Lê o valor dos pinos do dispositivo;
- LATx: Armazena o estado das portas;

2.4.3 – Timers

O PIC18F452 possui quatro Timers programáveis que podem gerar interrupções em intervalos de tempos específicos.

2.4.3.1 – Timer 0

O Timer 0 é similar ao da família PIC16, com exceção da possibilidade de operar no modo de 8-bits ou 16-bits. Ele pode operar como temporizador ou contador, suas características são:

- Operação em 8-bits ou 16-bits;
- Pré-escalonador (*prescaler*) programável de 8-bits;
- Fonte de clock externa ou interna;
- Interrupção no estouro de FFh para 00h (8-bits) e FFFFh para 0000h (16-bits);

O registrador *T0CON* controla a operação do Timer 0. Os seis bits menos significativos desse registrador configuram o prescaler e a fonte de clock. Os outros dois bits configuram o modo de operação (8 ou 16-bits) e habilitam o Timer.

T0CON

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0
bit 7							bit 0

Figura 2.14: Registrador *T0CON*

No modo 8-bits o valor carregado no registrador *TMR0* controla a contagem do Time 0 até que uma interrupção ocorra. A equação 2.1 pode ser usada para calcular o tempo necessário para que o Timer 0 estoure, podendo gerar interrupção.

$$\text{Tempo de Estouro} = 4 \times \text{Tosc} \times \text{Prescaler} \times (256 - \text{TMR0}) \qquad (2.1)$$

onde:

Tosc: é o período do oscilador em µs;

Prescaler: é o valor do pré-escalonador;

TMR0: é o valor carregado em TMR0;

No modo de 16-bits dois registradores são utilizados para armazenar o valor do Timer, *TMR0* e *TMR0L*. O byte menos significativo (TMR0L) é carregado diretamente do barramento de dados e o TMR0 é carregado através de um *buffer* chamado *TMR0H.* O tempo de estouro é calculado de maneira similar ao modo de 8-bits:

Tempo de Estouro (16-bits) = 4 x Tosc x Prescaler x (65536- *TMR0:TMR0L*) (2.2)

2.4.3.2 – Timer 1

O Timer 1 do PIC18F452 é de 16-bits, e é controlado pelos registradores *T1CON*, *TMR1H* e *TMR1L* os quais possuem as seguintes características:

- Contador/temporizador de 16-bits;
- Fonte de clock externa ou interna;
- Interrupção no estouro de FFFFh para 0000h;
- RESET em evento especial do módulo CCP;

O bit *TMR1CS* do registrador *T1CON* configura a fonte de "clock" para o Timer 1, se o *TMR1CS* for 0 será selecionada a operação como temporizador, caso contrário a operação como contador será selecionada. O bit RD16 configura para que a leitura/escrita dos registradores *TMR1H* e *TMR1L* seja no modo de 16-bits ou de 8-bits. Os demais registradores habilitam a operação do timer e configuram o pré-escalonador.

O tempo de estouro é calculado de maneira similar ao Timer 0 no modo 16-bits, com *TMR0:TMR0L* substituído por *TMR1H:TMR1L*. Se a interrupção do Timer 1 estiver habilitada, quando a contagem passar de FFFFh para 0000h uma interrupção será gerada.

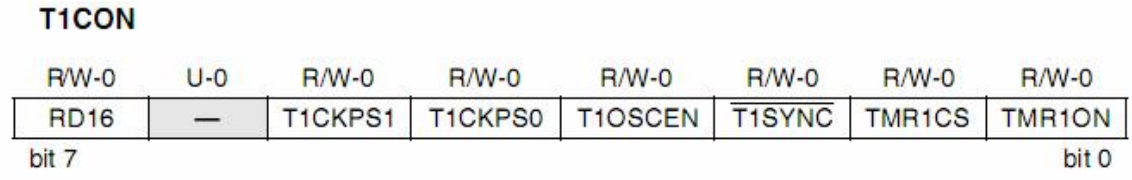

Figura 2.15: Registrador *T1CON*

2.4.3.3 – Timer 2

O Timer 2 possui as seguintes características:

- Timer de 8-bits (registrador *TMR2*);
- Registrador de período de 8-bits (*PR2*);
- Pré-escalonador programável (1:1, 1:4, 1:16);

- Pós-escalonador programável (1:1 até 1:16);
- Interrupção quando *TMR2* alcançar *PR2*;

O registrador T2CON, figura 2.16, controla a operação do Timer 2. Os bits *T2CKPS1: T2CKPS0* configuram o pré-escalonador e os bits *TOUTPS3:TOUTPS0* o pós-escalonador. O timer pode ser habilitado/desabilitado através do bit *TMR2ON*.

O Timer 2 pode ser utilizado como base de tempo para o PWM do módulo CCP e sua saída pode ser selecionada para o módulo SSP como clock.

T2CON

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
bit 7							bit 0

Figura 2.16: Registrador *T2CON*

2.4.3.4 – Timer 3

O Timer 3 possui a mesma estrutura e operação do Timer 1, com os registradores *TMR3H* e *TMR3L*. O registrador *T3CON* controla a operação do Timer 3.

2.4.4 – Módulo CCP

O microcontrolador PIC18F452 possui dois módulos Captura/Comparação/PWM (CCP), que operam em conjunto com os Timers 1, 2 e 3. Cada módulo possui um par de registradores de 8-bits que formam dois registradores de 16-bits: *CCPR1H:CCPR1L* (CCP1) e *CCPR2H:CCPR2L* (CCP2). Os registradores *CCP1CON* e *CCP2CON* controlam, respectivamente, os módulos CCP1 e CCP2. A figura 2.18 trás os timers utilizados por cada modo de operação CCP.

CCP1CON/CCP2CON

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	DCxB1	DCxB0	CCPxM3	CCPxM2	CCPxM1	CCPxM0
bit 7							bit 0

Figura 2.17: Registradores CCP1CON/CCP2CON

Modo CCP	Timer
Capture	Timer1 ou Timer3
Compare	Timer1 ou Timer3
PWM	Timer2

Figura 2.18: Modos CCP e Timers

2.4.4.1 – Captura

No modo Captura, os registradores *CCPRxH:CCPRxL* armazenam o valor dos registradores do TMR1 ou TMR3 quando ocorre um evento no pino RC2/CCP1 ou RC1/CCP2. O evento é selecionado pelos bits de controle *CCPxM3:CCPxM0* e podem ser:

- A cada borda de descida;
- A cada borda de subida;
- A cada quatro bordas de subida;
- A cada dezesseis bordas de subida;

Se a interrupção da captura estiver habilitada a ocorrência de um evento causará uma interrupção. Os Timers 1 ou 3 podem ser utilizados no modo captura e devem estar operando como timers ou como contadores sincronizados e selecionados através do registrador *T3CON*. A figura 2.19 ilustra o funcionamento do modo captura.

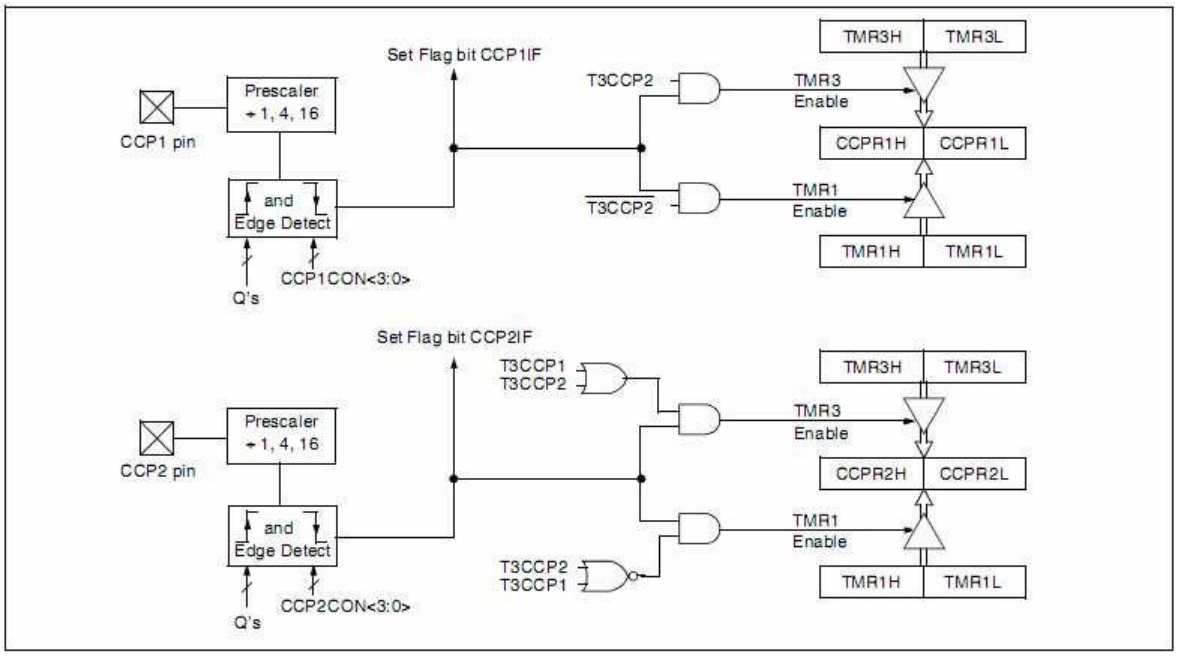

Figura 2.19: Diagrama de blocos do modo Captura

2.4.4.2 – Comparação

No modo Comparação, o valor do registrador de 16-bits CCPR1 (ou CCPR2) é constantemente comparado com o valor do par de registradores do Timer 1 ou do Timer 3. Quando o valor dos registradores são iguais o valor do pino CCP é modificado. A figura 2.20 ilustra o funcionamento do modo Comparação.

O valor do registrador *CCPRxH:CCPRxL* é continuamente comparado com o valor do Timer 1 ou do Timer 3 e quando são iguais, o estado do pino CCPx é modificado de acordo com a configuração dos bits *CCPxM2:CCPxM0* do registrador *CCPxCON,* e pode:

- Ser habilitado (nível lógico 1);
- Ser desabilitado (nível lógico 0);
- Trocar de nível lógico (1 para 0 ou 0 para 1);
- Não sofrer alteração;

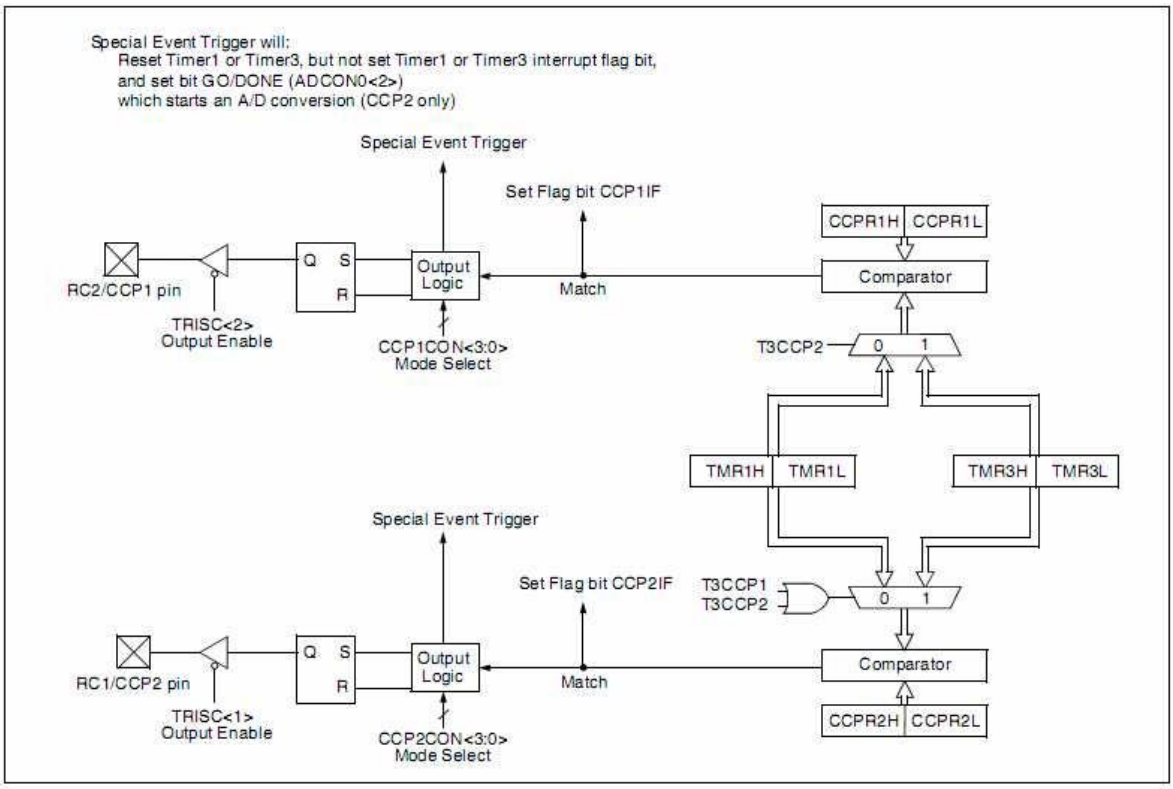

Figura 2.20: Diagrama de blocos do modo Comparação

A seleção do timer a ser usado no modo Comparação deve ser feita através do registrador *T3CON.* Um evento especial pode ser configurado através dos registradores *CCPxCON*. Esse evento pode resetar os Timers 1 ou 3 sem habilitar o flag de interrupção ou iniciar uma conversão A/D (apenas para o CCP2).

2.4.4.3 – PWM

No modo PWM (Modulação por Largura de Pulso), o pino CCPx produz uma saída PWM com resolução de até 10-bits, e é controlado pelo Timer 2. A saída PWM possui uma base de tempo (período) e um tempo em que a saída permanece alta (duty cycle).

O período do PWM é especificado quando escrevemos no registrador *PR2* e pode ser calculado usando a seguinte fórmula:

$$Período_{PWM} = [(PR2) + 1] \bullet 4 \bullet T_{OSC} \bullet (TMR2\ Prescaler) \qquad (2.3)$$

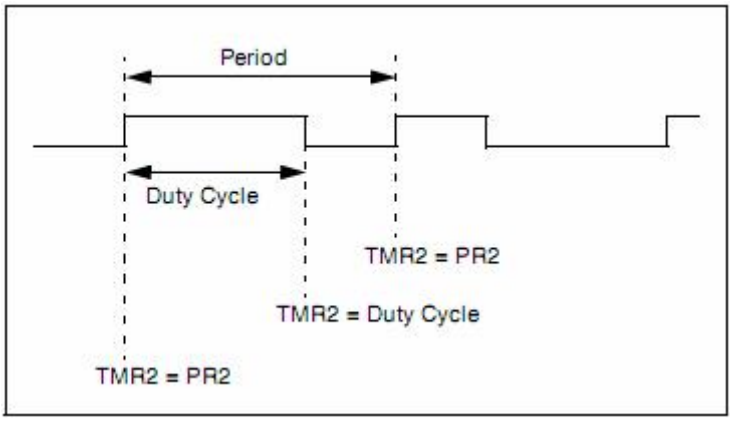

Figura 2.21: Saída PWM

A frequência do PWM é definida como o inverso do período. Quando o valor do Timer 2 é igual a PR2 três eventos ocorrem no próximo ciclo de clock:

- Timer 2 é apagado;
- O pino CCP1 é habilitado (exceto se o duty cycle for de 0%);
- O valor do Duty Cycle é armazenado de CCPR1L em CCPR1H.;

O Duty Cycle do PWM é especificado escrevendo no registrador *CCPRxL* e nos bits 4 e 5 do registrador *CCPxCON*. Uma resolução de até 10 bits está disponível. O registrador *CCPRxL* contém os oito bits menos significativos e os bits 4 e 5 do registador CCPxCON os dois mais significativos. A equação 2.4 é utilizada para calcular o Duty Cycle:

$$\text{Duty Cyle}_{PWM} = (\text{CCPR1L:CCP1CON<5:4>}) \bullet T_{OSC} \bullet (\text{TMR2 Prescaler}) \qquad (2.4)$$

O valor do Duty Cycle pode ser escrito a qualquer momento nos registradores *CCPRxL* e *CCPxCON,* porém, o valor só é armazenado em *CCPRxH* quando o valor do *TMR2* e de *PR2* forem iguais, ou seja, quando um novo período começar.

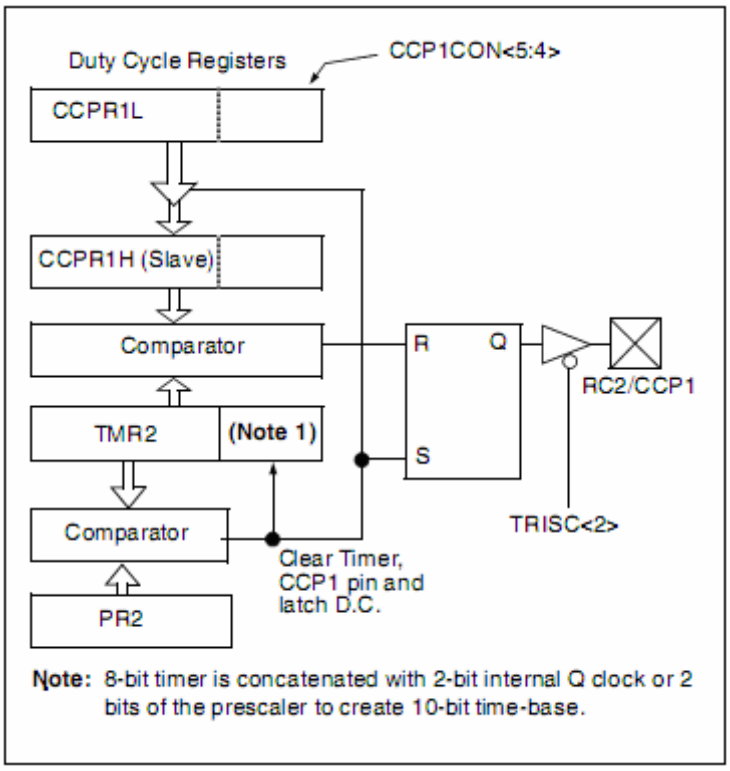

Figura 2.22: Diagrama de Blocos do modo PWM

A máxima resolução (bits) do PWM para uma dada frequência pode ser calculada usando a equação 2.5. A tabela 2 trás um exemplo de frequências e resoluções do PIC18F452 operando em 40 MHz.

$$\mathrm{Re}\,solução_{PWM} = \frac{\log(\frac{F_{osc}}{F_{PWM}})}{\log(2)} bits \qquad (2.5)$$

Para utilização do CCPx como PWM os seguintes passos devem ser seguidos:

- Configurar o período escrevendo no registrador *PR2*;
- Configurar o duty cyle escrevendo nos registradores *CCPR1L* e CCP1CON<5:4:>;
- Configurar o pino CCP1 como saída através do registrador TRISC;
- Configurar o pré-escalonador e habilitar o Timer 2 através do registrador *T2CON*;
- Configurar o módulo CCP para operação como PWM;

Frequência	2.44 kHz	9.77 kHz	39.06 kHz	156.25 kHz	312.50 kHz	416.67 kHz
Timer Prescaler (1, 4, 16)	16	4	1	1	1	1
PR2	0xFF	0xFF	0xFF	0x3F	0x1F	0x17
Máxima Resolução (bits)	14	12	10	8	7	6.58

Tabela 2: Exemplo de frequências e resolução em 40MHz

Capítulo 3 – Sistema para Implementação de Falhas

3.1 Introdução

O Sistema para Implementação de Falhas Intermitentes é composto por um microcontrolador PIC, um optoacoplador, um driver ir2125 e uma chave (IGBT ou MOSFET). O PIC é responsável por gerar o sinal de comando e a falha, o optoacoplador isola o circuito de baixa potência (microcontrolador) do circuito da chave e o driver ir2125 é responsável por converter o sinal de tensão do PIC em um sinal capaz de acionar as chaves.

Para implementação do sistema proposto, foi desenvolvida uma rotina para os microcontroladores PIC, que permite ao usuário configurar o PWM (sinal de comando), a duração e momento de ocorrência da falha.

Figura 3.1: Sistema de falhas de chaveamento

3.2 Rotina de Controle

A rotina de controle visa gerar o sinal de comando da chave e interromper esse sinal durante o tempo especificado pelo usuário. Três botões (ENTER, UP e DOWN) e um LCD foram utilizados como interface de entrada e saída, respectivamente.

No início da rotina, o Timer 0, o modo Captura e o PWM são inicializados. O PWM é configurado para uma frequência de saída de 20kHz e duty cyle de 50%. As interrupções do Timer 0 e do modo Captura foram habilitadas.

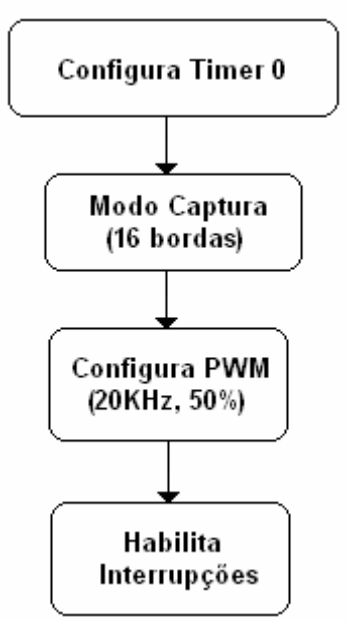

Figura 3.2 Sequência de Inicialização

Quatro menus estão disponíveis e podem ser selecionados através dos três botões. O botão ENTER é utilizado para confirmar a entrada e as modificações realizadas pelo usuário, sendo essas através das teclas UP e DOWN, em cada menu.

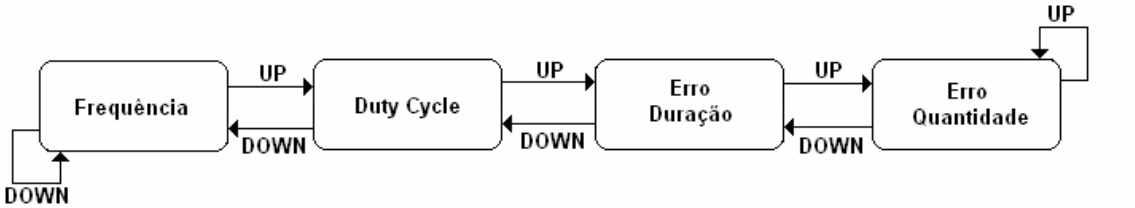

Figura 3.3 Menu Principal

O menu Frequência e o menu Duty Cycle permitem ao usuário modificar a frequência e o valor do duty cycle do PWM em tempo real. Nesses menus o Timer 0 é utilizado para aumentar a taxa de modificação da frequência e duty cycle, caso o usuário permaneça mais de 3 segundos com a tecla pressionada. A frequência pode variar entre 3 e 100kHz, e o duty entre 0% e 100%.

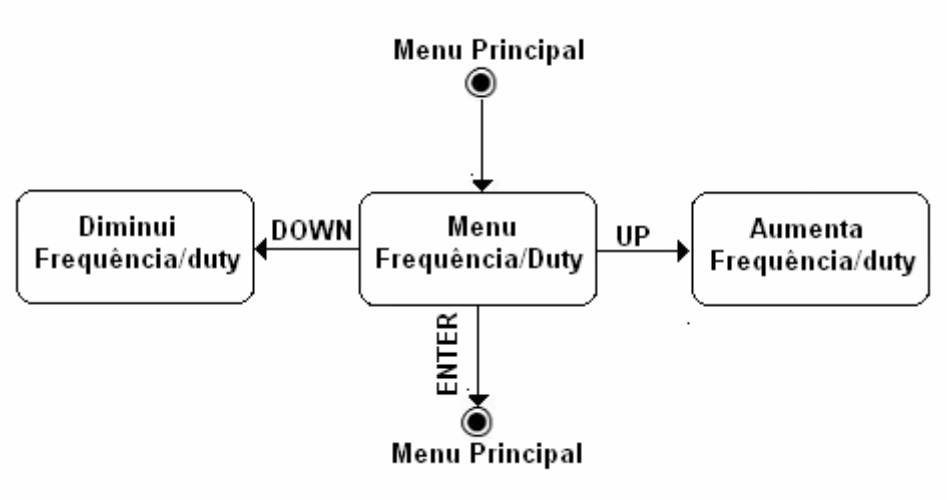

Figura 3.4: Menus Frequência/Duty Cycle

Nos menus Erro Duração e Erro Quantidade, podem ser configurados, respectivamente, a duração da falha e sua ocorrência. A duração da falha é especificada em microssegundos e a ocorrência em número de bordas de subida (múltiplos de 16). Após a configuração, uma mensagem para habilitação/desabilitação do erro é exibida nos dois menus.

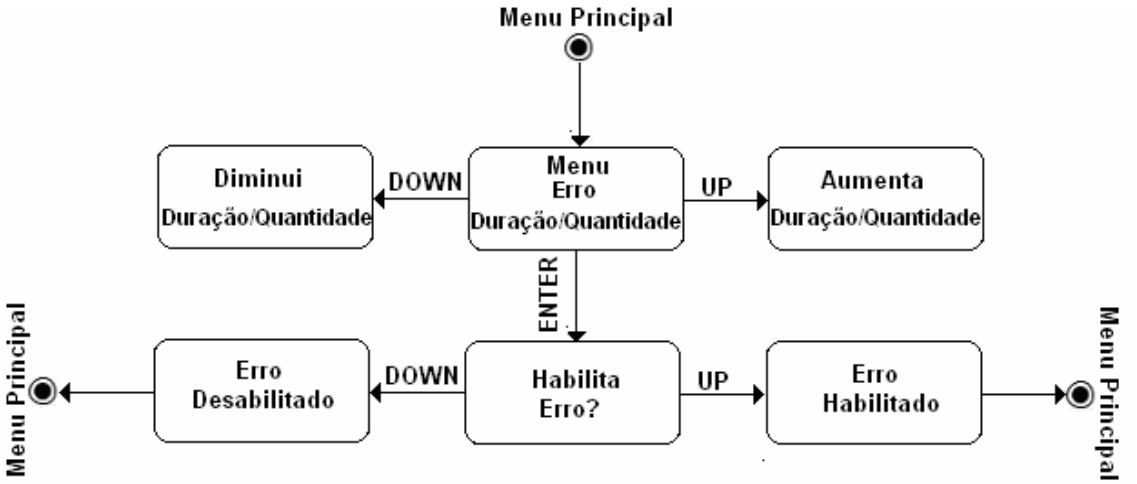

Figura 3.4 Menus Erro Duração/Erro Quantidade

Com a habilitação da falha, o PIC inicia a captura do PWM e, após a quantidade de bordas configuradas, a rotina para interromper o PWM é iniciada. Nessa rotina, é criado um atraso para desativar o PWM de acordo com o tempo especificado no menu Erro Duração. Para desativar o PWM, o registrador *CCP1CON* foi limpo. Após a desativação, o PWM é reativado em três quartos do

período, escrevendo 0x0C em *CCP1CON*. A falha pode ser desabilitada acessando novamente os menus de Erro. A figura abaixo ilustra o fluxograma para rotina de implementação da falha no PWM.

Figura 3.5: Fluxograma para implementação da falha no PWM

Capítulo 4 – Material, Métodos e Resultados

4.1 – Material e Métodos

Apenas a parte do sistema referente ao microcontrolador foi implementada, sendo o restante do sistema de menor importancia para a comprovação do funcionamento. O PIC18F452 foi o microcontrolador escolhido devido a sua disponibilidade, velocidade de operação e recursos disponíveis.

A rotina de controle foi desenvolvida com o compilador *PCWHD* v4.037 da *CCS Inc.*. O sistema foi montado em protoboard e alimentado por uma fonte de tensão PS-5000 da ICEL. Para a aquisição dos resultados, o *software Scope Connect* do osciloscópio DSO3202A da Agilent Technologies foi utilizado através de um microcomputador. A figura abaixo apresenta o sistema montado em laboratório.

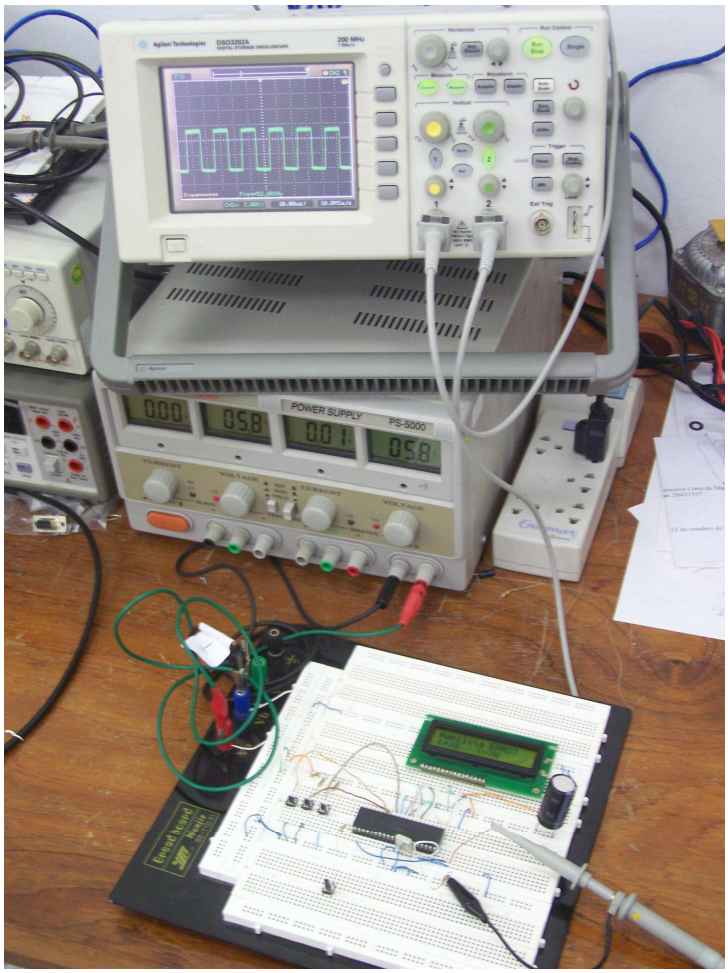

Figura 4.1: Sistema montado em protoboard

4.2 – Resultados

Os resultados obtidos foram para as frequências de 10, 20, 50, 75 e 100 KHz, com o momento de ocorrência da falha sempre iguais a ¼ do período. Foi observado que o atraso configurado para desativar o PWM tinha um acréscimo de 5µs. Esse atraso foi corrigido via software. Para frequências acima de 50 kHz, deve-se configurar a falha para que ocorra no período seguinte.

As figuras 4.2 e 4.3 ilustram os resultados de falhas no PWM em 10kHz, com duração de 25µs e ocorrência a cada 16 bordas.

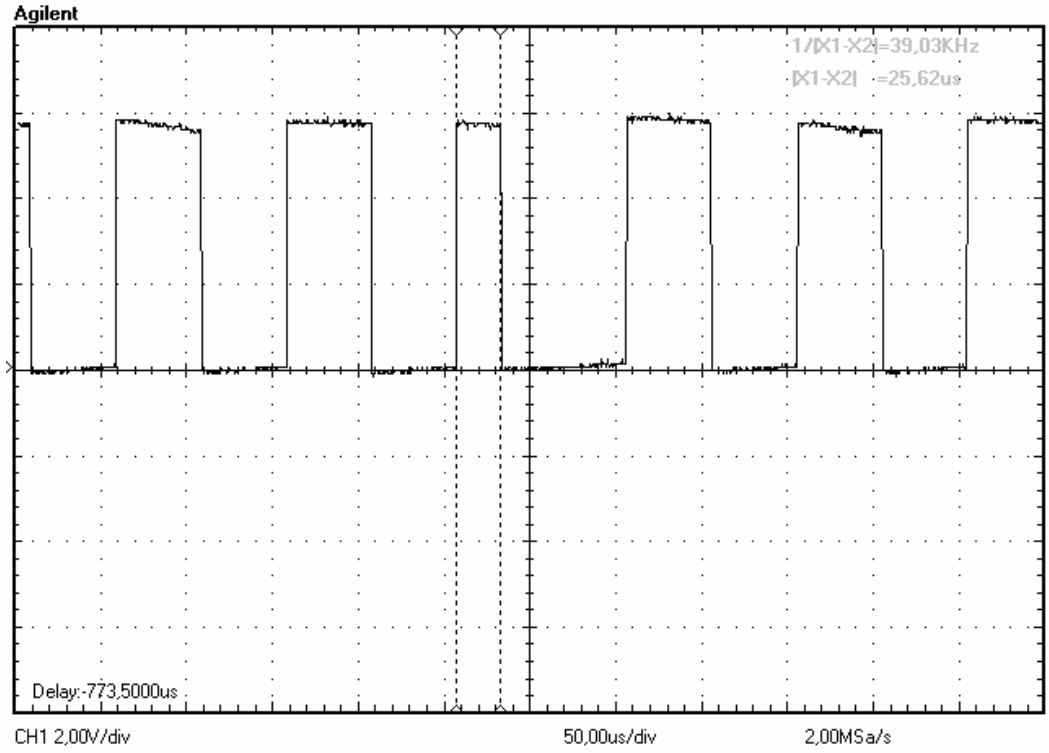

Figura 4.2: Falha de 25µs em 10kHz

Na figura 4.4 tem-se uma frequência de 20Khz e uma falha de 15µs. A partir de 39KHz, o PWM do PIC tem resolução menor que 10 bits e já observamos uma pequena diferença de frequência para um PWM em 50KHz, ver figura 4.5. Na figura 4.6, para a frequência de 50KHz uma falha de 5 µs é gerada com atraso de um período.

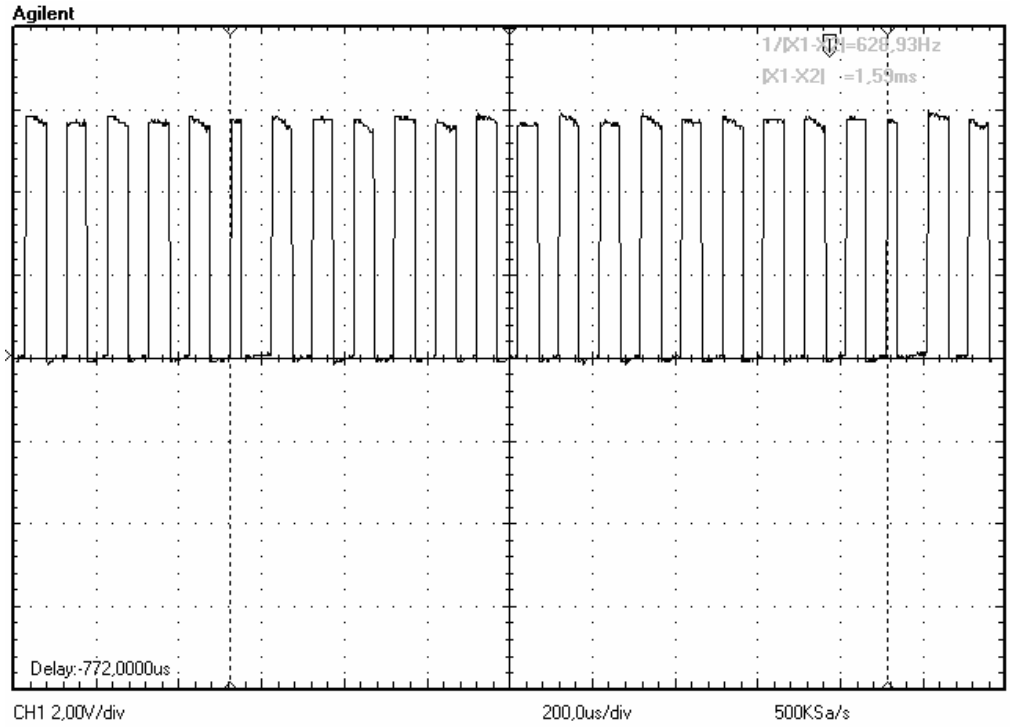

Figura 4.3: Ocorrência a cada 16 bordas em 10KHz

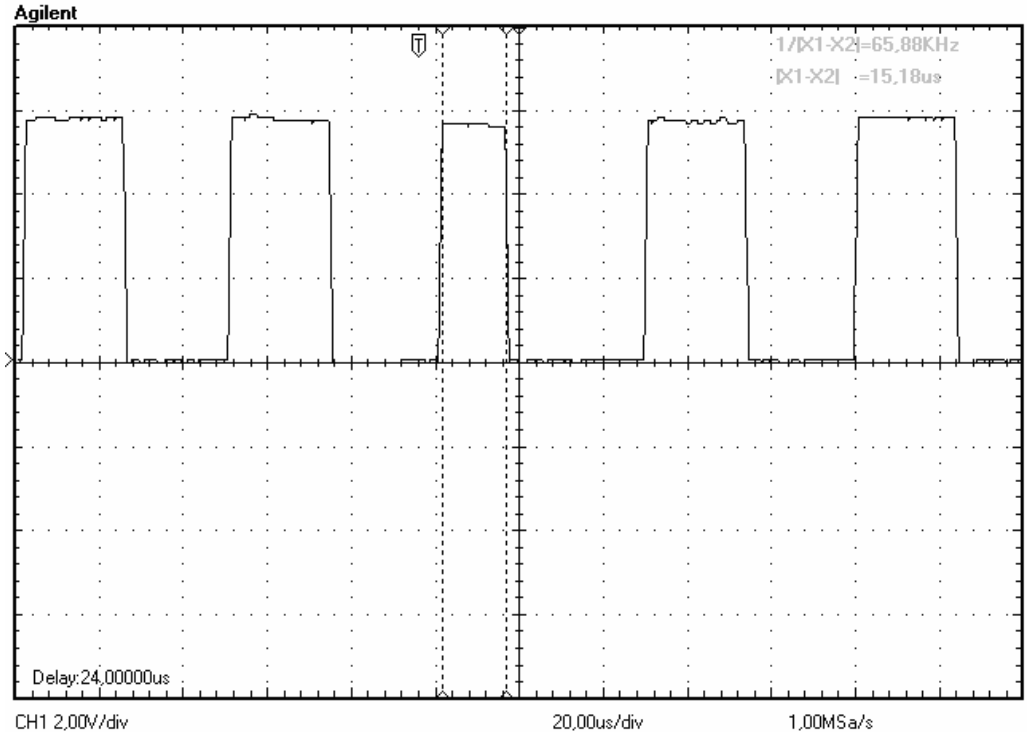

Figura 4.4: Falha de 15µs em 20KHz

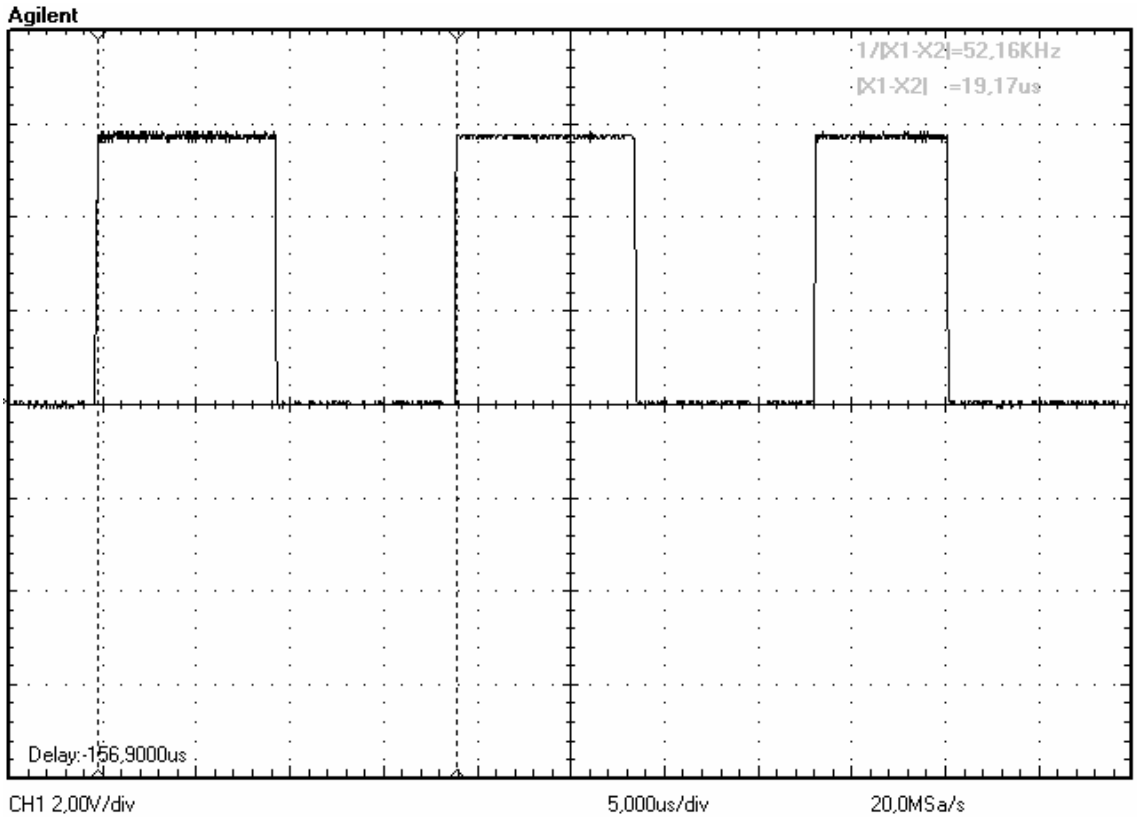

Figura 4.5: PWM em 50KHz

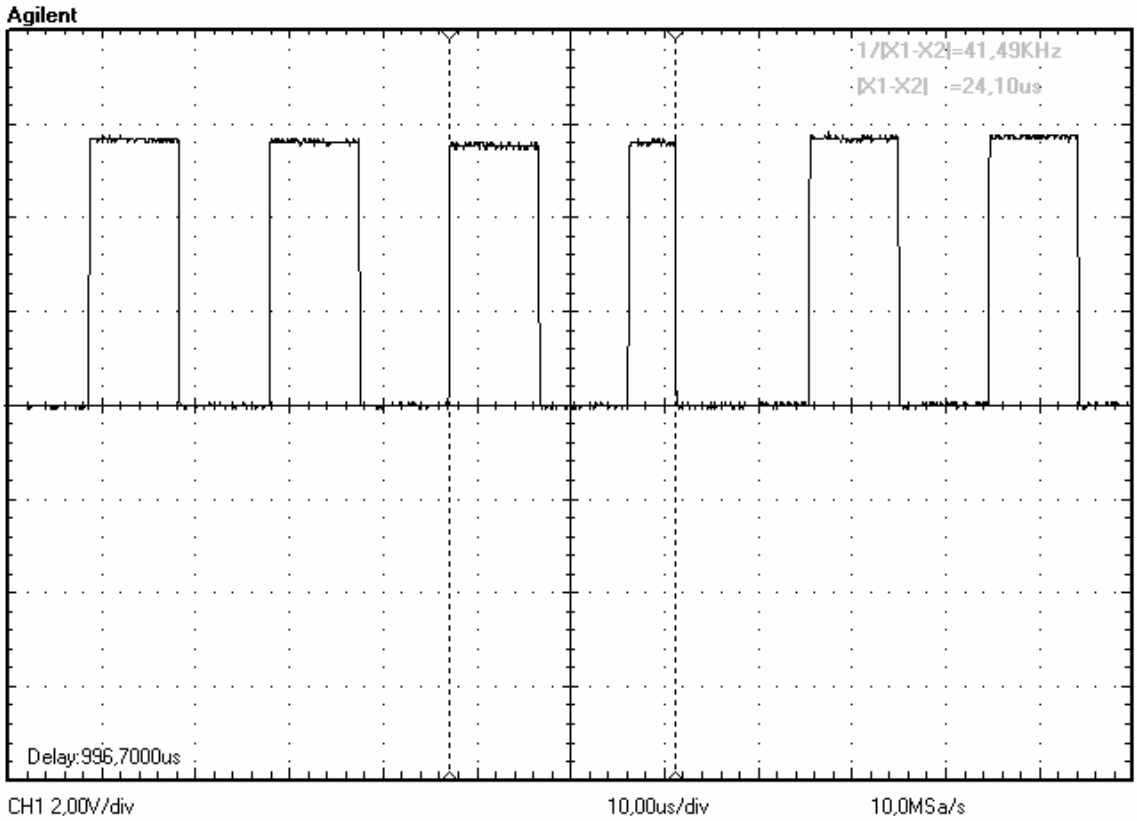

Figura 4.6: Falha de 5μs em 50Khz com atraso de um período

As figuras 4.7 e 4.8 ilustram as falhas geradas para as frequências de 75 kHz e 100 kHz, em 3 µs e 2 µs, respectivamente.

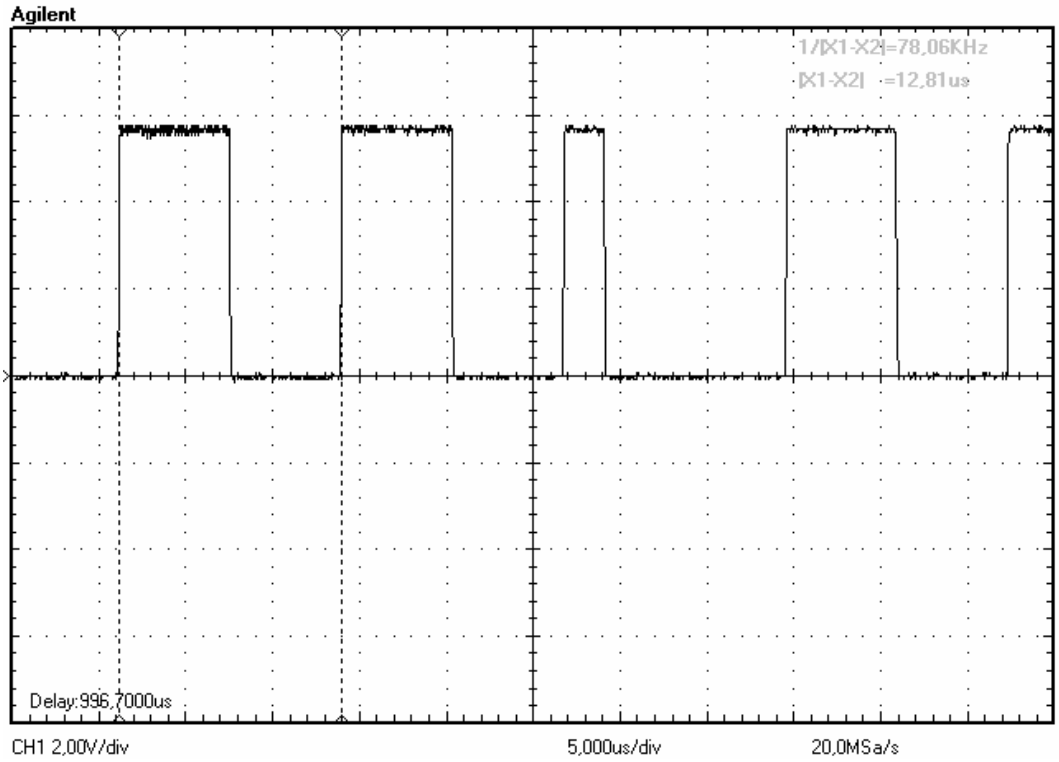

Figura 4.7: Falha de 3µs em 75kHz

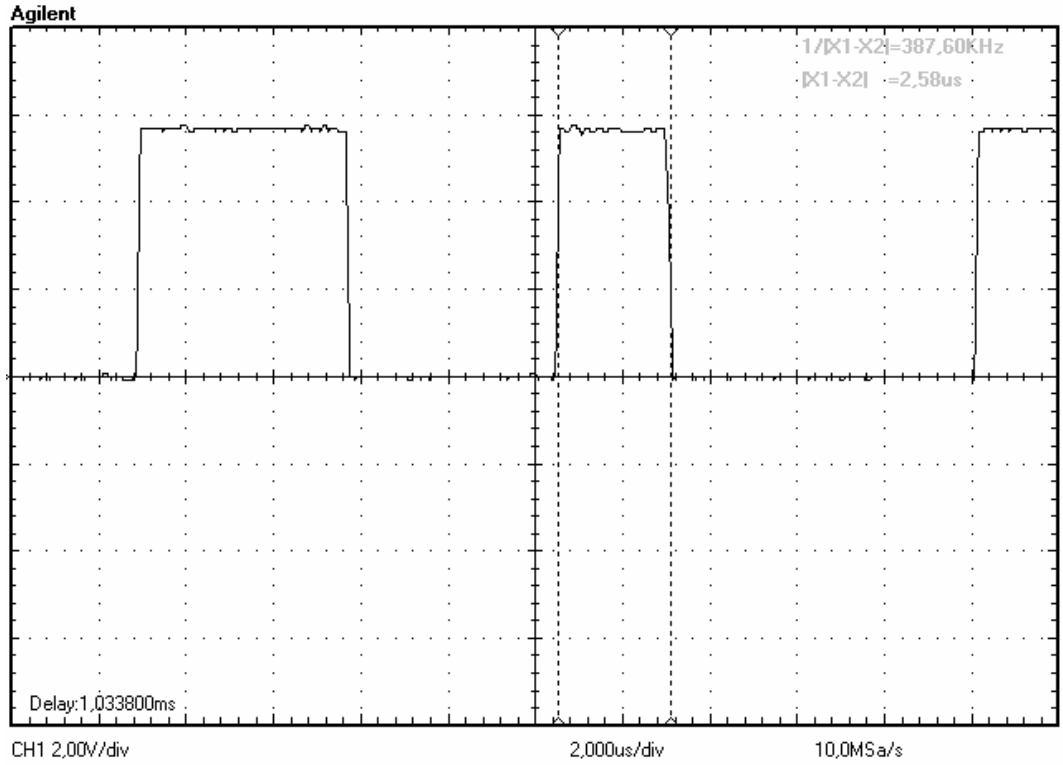

Figura 4.8: Falha de 2µs em 100kHz

Capítulo 5 – Considerações Finais e Sugestões Futuras

5.1 – Considerações Finais

Neste relatório foi apresentado um estudo sobre as diferentes famílias de microcontroladores PIC, assim como sobre a arquitetura e principais recursos do PIC18F452. Foi apresentado o sistema para geração de falhas intermitentes e sua rotina de controle.

Os resultados obtidos no experimento são todos para falhas de ¼ do período, porém, qualquer valor pode ser especificado, respeitando o limite de 5 µs.

O PIC18F452 mostrou um bom desempenho para frequências abaixo de 50KHz. Para frequências cada vez maiores, o erro na frequência torna-se relevante.

A rotina apresentada mostrou-se eficiente, podendo ser utilizada em estudos sobre falhas no comando de interruptores e pode ser facilmente adaptada a outros controladores PIC.

5.2 – Sugestões Futuras

Substituir o PIC18F452 por um microcontrolador de melhor desempenho, da família de PIC16 ou superior, melhorando a resposta do sistema para altas frequências.

Otimizar o código para que o tempo de reativação do PWM seja escolhido pelo usuário.

Implementar um sistema para detecção das falhas.

Anexo

1. Código principal: falha.c

```
/*****************************************************************************
                    Universidade Federal de Campina Grande
                  Centro de Engenharia Elétrica e Informática
                  Unidade Acadêmica de Engenharia Elétrica
                          Projeto de Engenharia Elétrica

                          Orientador: Talvanes Meneses Oliveira
                        Aluno: Luis Gustavo Guedes Pereira de Castro

                      Rotina para implementação de falhas intermitentes
                        no chaveamento de comando de interruptores
      *****************************************************************************/
#include "falha.h"
#include "gera_duty.c"
#include "gera_period.c"

#define CCP1CON 0xFBD
#priority ccp2,timer0

//Rotina de Interrupção do Módulo CCP2
#int_ccp2
void ccp2_isr()
{
   if(ERRO)
   {
   if (++cont_duracao==duracao)
   {

  #asm
  loop: decfsz tempo                //dura 3 ciclos
   goto loop
   clrf CCP1CON;                    //Desabilita o PWM

 #endasm

delay_us(tempo2);

#asm
  MOVLW   0X0C
  MOVWF   CCP1CON           //Habilita o PWM
  #endasm

cont_duracao=0;
  }
   }
}
```

```
#int_timer0
void timer0_isr()
{
   if (++cont_tecla == 500)
   {
   cont_tecla = 0;
   if (!input(UP_KEY))
   {
      variavel += 1;
      if (variavel >= 10)
         variavel = 10;
   }
   else variavel = 1;
   if (!input(DOWN_KEY))
   {
      variavel2 += 1;
      if (variavel2 >= 10)
         variavel2 = 1;
   }
   else variavel2 =1;

  }

}
//Função para leitura das teclas
int tecla (void)
{
 int kp =0;

 while(!kp)
 {
   if(!input(pin_B0)) kp = MENU_KEY;
   else if(!input(pin_B1)) kp = UP_KEY;
   else if(!input(pin_B2)) kp = DOWN_KEY;
   delay_ms(500);
 }
return kp;
}

void main()
{
int8 key;
inicialization();


   while(1)
   {
      if(EXT_FLAG)disable_interrupts(INT_ext);
      key = tecla();
```

```
      if (key == MENU_KEY)
        menu_enter();
      else if (key == UP_KEY)
        ++submenu;
      else if (key == DOWN_KEY)
        --submenu;
      if(key!=0)
      {
        menu_adjust();
        menu_show();
      }
   }

}
void inicialization (void)
{

   setup_adc_ports(NO_ANALOGS);
   setup_adc(ADC_OFF);
   setup_psp(PSP_DISABLED);
   setup_spi(SPI_SS_DISABLED);
   setup_wdt(WDT_OFF);
   setup_timer_0(RTCC_INTERNAL|RTCC_DIV_64|RTCC_8_BIT);
   setup_timer_1(T1_DISABLED);
   setup_timer_2(T2_DISABLED,0,1);
   setup_timer_3(T3_DISABLED|T3_DIV_BY_1);


  //Habilita interrupções
   enable_interrupts(INT_timer0);
   enable_interrupts(INT_CCP2);
   enable_interrupts(GLOBAL);

   set_tris_a(0);
   output_low(PIN_A0);

   //Seta o timer 0s para contagem
   //(4/40000000)*64=6.4us , 6.4us*157 = 1ms  => 255-157 = 98 (0x62)
  set_timer0(0x62);

 //Inicia LCD
   lcd_init();
   delay_ms(500);
   printf(LCD,"\f     TCC\n      LG");

  //Inicia PWM em 20kHz / 50%
   pr2 = ((FOSC)/(16*4*(float)frequencia*1000))-1;
   CCPCON = ((((float)dut/100)*(1/((float)frequencia*1000)))*FOSC)/16;
   setup_timer_2(pre[2],(int)pr2, 1);
   set_pwm1_duty((int16)CCPCON);
```

```
   tempo2 = (int)((3/4)*(1/frequencia)*1000);

   setup_ccp2(CCP_CAPTURE_DIV_16);              //seta CCP2 para CAPTURE
   setup_ccp1(ccp_pwm);                         //seta CCP1 para PWM
}
void menu_adjust (void)
{
   if(submenu >4)submenu=4;
   else if (submenu<=0)submenu=1;
}
void menu_show (void)
{
   if(submenu==0x01) printf(LCD,"\f       PWM\n  FREQUENCIA");
   else if(submenu==0x02) printf(LCD,"\f       PWM\n     DUTY");
   else if(submenu==0x03) printf(LCD,"\f   ERRO\nQuantidade");
   else if(submenu==0x04) printf(LCD,"\f   ERRO\nDuracao");
}
void menu_enter (void)
{
   if(submenu==0x01) pwm_freq();
   else if(submenu==0x02) pwm_duty();
   else if(submenu==0x03) erro_qnt();
   else if(submenu==0x04) erro_dur();
}
void pwm_freq (void)
{
  int key;
   printf(LCD,"\f FREQUENCIA");

   while(key!=MENU_KEY)
   {
      key = tecla();

      if (key == UP_KEY)
         frequencia=frequencia+variavel;
      if (key == DOWN_KEY)
         frequencia=frequencia-variavel2;
      if(frequencia < 3) frequencia =3;
      if (frequencia >100) frequencia = 100;
    gera_period();
    printf(LCD,"\n%dkHz",frequencia);
    tempo2 = (int)((3/4)*(1/frequencia)*1000);
    }
}
void pwm_duty (void)
{
   int key;
   printf(LCD,"\f DUTY");
   while(key!=MENU_KEY)
   {
```

```
      key = tecla();

      if (key == UP_KEY)
         dut=dut+variavel;
      if (key == DOWN_KEY)
         dut=dut-variavel2;
    gera_duty();
    printf(LCD,"\n%d%%",dut);
    }
}
void erro_qnt (void)
{
   int key;
   tempo = 50;
   printf(LCD,"\f TEMPO\n %li",tempo);
   while(key!=MENU_KEY)
   {
      key = tecla();

      if (key == UP_KEY)
         tempo=tempo+variavel;
      if (key == DOWN_KEY)
         tempo=tempo-variavel2;
      if(tempo < 4) tempo =4;
      if (tempo >1000) tempo = 1000;
      printf(LCD,"\n %li",tempo);
    }
   printf(LCD,"\f Habilita ERRO?\n(+)S   (-)N");
   key=0;
   while(!key)
   {
      key = tecla();

      if (key == UP_KEY)
      {
      printf(LCD,"\fERRO HABILITADO");
      printf(LCD,"\ntempo:%li dura:%d ",tempo,duracao);
      tempo = ((tempo-5)/0.3);
      ERRO_ON;
      delay_ms(1000);
      FLAG_ON;
      }
      if (key == DOWN_KEY)
         ERRO_OFF;
   }
}
void erro_dur (void)
{
   int key;
   printf(LCD,"\f DURACAO");
```

```
  while(key!=MENU_KEY)
  {
     key = tecla();

     if (key == UP_KEY)
        duracao=duracao+variavel;
     if (key == DOWN_KEY)
        duracao=duracao-variavel2;
     if(duracao < 1) duracao =1;
     if (duracao >20) duracao = 20;
     printf(LCD,"\n %d periodos",duracao);
   }
  printf(LCD,"\f Habilita ERRO?\n(+)S   (-)N");
  key=0;
  while(!key)
  {
     key = tecla();

     if (key == UP_KEY)
     {
     printf(LCD,"\fERRO HABILITADO\ntempo:%li dura:%d",tempo,duracao);
     tempo = ((tempo-5)/0.3);
     ERRO_ON;
     delay_ms(1000);
     FLAG_ON;
     }
     if (key == DOWN_KEY)
      ERRO_OFF;
  }
}
```

2. Arquivo de cabeçalho falhas.h

```
#include <18F452.h>

#FUSES NOWDT                 //No Watch Dog Timer
#FUSES H4                    //High speed osc with HW enabled 4X PLL
#FUSES NOBROWNOUT   //No brownout reset
#FUSES NOPUT                 //Power Up Timer
#FUSES NOLVP                //No low voltage prgming, B3(PIC16) or B5(PIC18)
used for I/O


#use delay(clock=10000000)
#use fast_io(A)

//Definição de constantes
#define MENU_KEY     PIN_B0
#define UP_KEY       PIN_B1
#define DOWN_KEY     PIN_B2
```

```
#define FOSC         40000000
#define FLAG_ON      FLAG=1
#define FLAG_OFF     FLAG=0
#define EXT_ON       EXT_FLAG=1
#define EXT_OFF      EXT_FLAG=0
#define ERRO_ON      ERRO=1
#define ERRO_OFF     ERRO=0

#include "LCD2.c"
#include <math.h>


//Declaração das variáveis utilizadas no programa
int         submenu=1;
char        pre[3] = {T2_DIV_BY_1,T2_DIV_BY_4,T2_DIV_BY_16};
int         frequencia=20;
float       CCPCON=78;
float       pr2=156;
int         dut=50;
int         prescaler = 16;
int         variavel = 1;
int         variavel2 = 1;
int         duracao=1;
int         quantidade=0;
int         cont_tecla=0;
int         cont_duracao=0;
int16       tempo=83;
int         tempo2;
boolean     FLAG;
boolean     EXT_FLAG;
boolean     ERRO=0;
int         x=0;
int        COUNTER=50;

//Declaração das funções
void inicialization (void);
void menu_enter     (void);
void menu_adjust    (void);
void menu_show      (void);
void pwm_freq      (void);
void pwm_duty      (void);
void erro_qnt      (void);
void erro_dur      (void);
```

3. Arquivo gera_duty.c

```
//A função realiza o calculo do valor de frequência a ser escrito em CCP1CON
void gera_duty () {

CCPCON = ((((float)dut/100)*(1/((float)frequencia*1000)))*FOSC)/prescaler;
```

```
set_pwm1_duty((int16)CCPCON);
}
```

4. Arquivo gera_period.c

```
//A função calcula o valor do período a ser escrito em PR2
{
   pr2 = ((FOSC)/(16*4*(float)frequencia*1000))-1;
   setup_timer_2(pre[2],(int)pr2, 1);
   prescaler = 16;
   if ((int)pr2 >= 255)
   {
      pr2 = ((FOSC)/(4*4*(float)frequencia*1000))-1;
      setup_timer_2(pre[1],(int)pr2, 1);
      prescaler = 4;
    }
  else if (pr2 >= 255)
   {
      pr2 = ((FOSC)/(1*4*(float)frequencia*1000))-1;
      setup_timer_2(pre[0],(int)pr2, 1);
      prescaler = 1;
  }
CCPCON = ((((float)dut/100)*(1/((float)frequencia*1000)))*FOSC)/prescaler;
set_pwm1_duty((int16)CCPCON);;
}
```

Referências Bibliográficas

[1]. OLIVEIRA, Talvanes Meneses; Hubert Razik. **Análise e Identificação de Falta no Comando de Interruptores Usando Wavelet**. Campina Grande, 2008.

[2]. MICROCHIP Technology Inc,. **PIC18FXX2 Datasheet**. Disponível em: http://ww1.microchip.com/downloads/en/DeviceDoc/39564c.pdf

[3]. WILMSHURT, Tim. **Designing Embedded Systems with PIC Microcontroller – Principles and Aplications**. Londres: Elsevier, 2007.

[4]. IBRAHIM, Dogan. **Advanced PIC Microcontroller Projects in C**. Estados Unidos: Elsevier, 2008.

[5]. MICROCHIP Technology Inc. **8-bit PIC Microcontrollers Brochure.** Disponível em: http://ww1.microchip.com/downloads/en/DeviceDoc/8bit%20brochure%20FINAL%2039630e.pdf

[6]. MICROCHIP Technology Inc. **PIC24 Microcontrollers Brochure**. Disponível em: http://www.microchip.com/stellent/groups/picmicro_sg/documents/devicedoc/en025311.pdf

[7]. MICROCHIP Technology Inc. **PIC32 Overview Brochure**. Disponível em: http://ww1.microchip.com/downloads/en/DeviceDoc/PIC32_Brochure_USB_39904f.pdf

[8]. PARLMER, Mark,. **Using the CCP Module(s) – AN594**. MICROCHIP Technology Inc 1997. Disponível em: http://ww1.microchip.com/downloads/en/AppNotes/00594B.PDF

[9]. PARLMER, Mark,. **Using the PWM – AN564**. MICROCHIP Technology Inc 1997. Disponível em: http://ww1.microchip.com/downloads/en/AppNotes/00564b.pdf

[10]. D´SOUZA, Stan; MITRA, Sumit. **Frequency and Resolution Options for PWM Outputs – AN539**. MICROCHIP Technology Inc 1997. Disponível em: http://ww1.microchip.com/downloads/en/AppNotes/00539c.pdf

[11]. MICROCHIP Technology Inc. **Complete Mid-Range Reference Manual.** 2004**.** Disponível em: http://ww1.microchip.com/downloads/en/DeviceDoc/33023a.pdf

[12]. PEREIRA, Fabio. **PIC Programação em C**. São Paulo: Érica, 1º Edição, 2003.