Daniel Rodrigues de Sousa
David José de Souza
Nicolás César Lavinia

DESBRAVANDO O MICROCONTROLADOR PIC18

PIC18F4520

Recursos Avançados

Características, Interrupções, Watchdog Timer

Manipulação de I/O, Timers, SPI, I²C, USART, módulo CCP (Capture, Compare, PWM), conversor AD, comparadores de tensão, tensão de referência ajustável, recursos avançados, técnicas de hardware e de software.

ONLINE

Códigos-fonte dos exemplos apresentados, respostas dos exercícios, data sheets dos componentes adicionais (LCD e memória), listagem dos operadores e diretivas do MPASM e um software para testes da comunicação serial via RS232, desenvolvido pelos autores disponíveis em www.editoraerica.com.br para download.

Desbravando o Microcontrolador PIC18

Recursos Avançados

Editora Érica - Desbravando o Microcontrolador PIC18 Recursos Avançados - Daniel R. de Sousa, David José de Souza e Nicolás César Lavinia - 1ª Edição

Editora Érica - Desbravando o Microcontrolador PIC18 Recursos Avançados - Daniel R. de Sousa, David José de Souza e Nicolás César Lavinia - 1ª Edição

Daniel Rodrigues de Sousa
David José de Souza
Nicolás César Lavinia

Desbravando o Microcontrolador PIC18

Recursos Avançados

1ª Edição

érica | Saraiva

Dados Internacionais de Catalogação na Publicação (CIP)
(Câmara Brasileira do Livro, SP, Brasil)

Sousa, Daniel Rodrigues de

Desbravando o Microcontrolador PIC18: Recursos Avançados / Daniel Rodrigues de Sousa, David José de Souza, Nicolás César Lavinia. – 1. ed. -- São Paulo : Érica, 2010.

Bibliografia.
ISBN 978-85-365-1966-1

1. Microcontroladores I. Souza, David José de. II. Lavinia, Nicolás César. III. Título.

09-13112 Editado também como livro impresso CDD-621.3916

Índice para catálogo sistemático:
1. Microcontroladores: PIC18: Engenharia eletrônica 621.3916

Copyright © 2010 da Editora Érica Ltda.
Todos os direitos reservados. Nenhuma parte desta publicação poderá ser reproduzida por qualquer meio ou forma sem prévia autorização da Editora Érica. A violação dos direitos autorais é crime estabelecido na Lei nº 9.610/98 e punido pelo Artigo 184 do Código Penal.

Coordenação Editorial: Rosana Arruda da Silva
Capa: Maurício S. de França
Editoração e Finalização: Rosana Ap. A. dos Santos
Flávio Eugenio de Lima
Carla de Oliveira Morais

Os Autores e a Editora acreditam que todas as informações aqui apresentadas estão corretas e podem ser utilizadas para qualquer fim legal. Entretanto, não existe qualquer garantia, explícita ou implícita, de que o uso de tais informações conduzirá sempre ao resultado desejado. Os nomes de sites e empresas, porventura mencionados, foram utilizados apenas para ilustrar os exemplos, não tendo vínculo nenhum com o livro, não garantindo a sua existência nem divulgação. Eventuais erratas estarão disponíveis para download no site da Editora Érica.

Conteúdo adaptado ao Novo Acordo Ortográfico da Língua Portuguesa, em execução desde 1º de janeiro de 2009.

A Ilustração de capa e algumas imagens de miolo foram retiradas de <www.shutterstock.com>, empresa com a qual se mantém contrato ativo na data de publicação do livro. Outras foram obtidas da Coleção MasterClips/MasterPhotos© da IMSI, 100 Rowland Way, 3rd floor Novato, CA 94945,USA, e do CorelDRAW X5 e X6, Corel Gallery e Corel Corporation Samples. Copyright© 2013 Editora Érica, Corel Corporation e seus licenciadores. Todos os direitos reservados.

Todos os esforços foram feitos para creditar devidamente os detentores dos direitos das imagens utilizadas neste livro. Eventuais omissões de crédito e copyright não são intencionais e serão devidamente solucionadas nas próximas edições, bastando que seus proprietários contatem os editores.

Seu cadastro é muito importante para nós

Ao preencher e remeter a ficha de cadastro constante no site da Editora Érica, você passará a receber informações sobre nossos lançamentos em sua área de preferência.
Conhecendo melhor os leitores e suas preferências, vamos produzir títulos que atendam suas necessidades.

Contato com o editorial: editorial@editoraerica.com.br

Editora Érica Ltda. | Uma Empresa do Grupo Saraiva
Rua São Gil, 159 - Tatuapé
CEP: 03401-030 - São Paulo - SP
Fone: (11) 2295-3066 - Fax: (11) 2097-4060
www.editoraerica.com.br

» Distribuidores

Mosaico Engenharia Eletrônica S/C Ltda.

Projetos eletrônicos com especialização em PIC, consultoria e treinamento.
Rua Galeão Carvalhal, 125
Bairro Jardim Bela Vista - Santo André - SP
Tel.: (11) 4992-8775
Tel./Fax: (11) 4992-8775
e-mail: comercial@mosaico.com.br
Site: www.mosaico.com.br

Microchip Technology Inc.

Fabricante do PIC.
2355, W. Chandler Blvd.
Chandler - Arizona - USA
Tel.: (480) 768-7200
Fax: (480) 899-9210
Site: www.microchip.com

Aplicações Eletrônicas Artimar Ltda.

Representante exclusivo Microchip no Brasil.
Rua Bela Cintra, 746 - 3º andar - São Paulo - SP
Tel.: (11) 3231-0277
Fax: (11) 3255-0511
e-mail: artimar@artimar.com.br
Site: www.artimar.com.br

Bevian Comercial e Importadora Ltda.

Revendedor autorizado.
Rua Lord Cokrane, 616 - São Paulo - SP
Tel.: (11) 3437-7443
e-mail: bevian@bevian.com.br
Site: www.bevian.com.br

Farnell-Newark

Revendedor autorizado.
Rua Emir Macedo Nogueira, 240 - Diadema - SP
Tel.: (11) 4066-9400
Fax: (11) 4066-9410
e-mail: vendas@farnell-newarkinone.com.br
Site: www.farnell.com

Future Electronics

Revendedor autorizado.
Rua Luzitana, 740 - 10º andar - conj. 103/104
Campinas - SP
Tel.: (19) 3737-4100
e-mail: future.saopaulo@future.ca
Site: www.futureeletronics.com

Hitech

Revendedor autorizado.
Av. Adolfo Pinheiro, 1000 - São Paulo - SP
Tel.: (11) 2182-4300
Fax: (11) 2182-4360
e-mail: microchip@hitech.com.br
Site: www.hitech.com.br

Avnet do Brasil

Revendedor autorizado.
Rua Luis Góes, 1205 - 2º andar - São Paulo - SP
Tel.: (11) 5079-2150
Site: em.avnet.com/mcc

Intertek Componentes Eletrônicos Ltda.

Revendedor autorizado.
Rua Miguel Casagrande, 200 - São Paulo - SP
Tel.: (11) 3186-2922
Fax: (11) 3186-2924
e-mail: intertek@intertek.com.br
Site: www.intertek.com.br

» Requisitos básicos de hardware e de software

Hardware

- ✓ Microcomputador Pentium 200 MHz ou superior;
- ✓ Monitor com resolução 800x600;
- ✓ 128 MB de memória RAM;
- ✓ 800 MB livres de HD;
- ✓ Acesso à Internet.

Softwares

- ✓ Sistema Operacional Windows 98 SE ou superior;
- ✓ MPLAB 8.40 ou superior.

  Link MPLAB IDE:

  http://www.microchip.com/mplab

Kit didático utilizado no livro

Todas as experiências do livro foram feitas com a placa didática McLab2 da LabTools, divisão da Mosaico High Performance Solutions, todavia o apêndice apresenta um hardware proposto para a realização das experiências.

A placa de desenvolvimento McLab2 é dotada de um microcontrolador PIC de 8 bits, que consiste no PIC18F4520 da Microchip.Veja todos os recursos que a placa oferece:

- ✓ LEDs (quatro LEDs);
- ✓ Varredura de display (quatro displays);
- ✓ Multiplexação de portas;
- ✓ Memória interna/externa;
- ✓ AD e AD através de RC;
- ✓ Comunicação com PC (RS-232);
- ✓ Buzzer;
- ✓ Sistema completo de temperatura (sensor, aquecedor e resfriador);
- ✓ Medição de rotação (óptico);
- ✓ Conector de expansão;
- ✓ Compatibilidade com os gravadores McFlash+ e ICD2[BR].

Observação: Esta placa didática é de inteira responsabilidade do fabricante.

Dedicatória

Dedico este trabalho à minha querida Andréa, meu amor de ontem, hoje e sempre. Também não posso deixar de dedicá-lo às minhas filhas Rebeca e Meirieny. Que Deus nos abençoe sempre!

Daniel Rodrigues de Sousa

"Portanto, não vos entristeçais, porque
a alegria do Senhor é a vossa força."

Neemias 8:10

Agradecimentos

Em primeiro lugar a Deus, pela minha vida cheia de bênçãos e, principalmente, por tudo que me trouxe. Ele sempre é fiel às suas promessas. Que Deus seja louvado sempre!

A todas as pessoas que, direta e indiretamente, ajudaram a concretizar este livro e, ainda, à minha querida filhinha Rebeca, pela alegria que trouxe a um pai coruja. Amo todos vocês!

Ao pessoal da Mosaico pelo apoio e pela participação do livro. Especialmente ao Gil pelo tempo da Mosaico que usei para a concretização desta obra, ao David e ao Nicolás pelo conhecimento da obra Conectando o PIC utilizado neste livro.

Ao professor, amigo e irmão em Cristo Jesus, Walter Varella, por tudo.

À Editora Érica por possibilitar a concretização do livro.

Por último, como nos demais livros, não poderia deixar de agradecer a você, leitor, que está se preparando para passar horas ao lado deste livro, enfrentando os mesmos desafios que enfrentamos quando começamos a explorar esse vasto universo denominado PIC.

Daniel Rodrigues de Sousa

Índice analítico

CAPÍTULO 1 - INTRODUÇÃO 17
Nossos objetivos 17
Pré-requisitos sugeridos 17
A didática do sistema 18
Introdução 18
Prática 18
Projeto final 19
Apêndices 19
Padrões textuais 19

CAPÍTULO 2 - O PIC18F4520 20
Pinagem 21
Nomenclatura dos pinos 21
A estruturação interna 23
Os ciclos de máquina 25
A memória de programa 26
Vetor de reset 26
Vetores de interrupção 26
Tamanho da memória 26
Pilha (stack) 27
Mapa da memória de programa 27
A memória de dados volátil (RAM) 28
Registradores especiais 28
Registradores de uso geral 28
Tamanho da memória e bancos 28
Mapa da memória de dados 30
Mapa da memória de dados (SFRs) 31
Memórias não voláteis (E^2PROM e FLASH) 31
As interrupções 32
Interrupções externas 32
Interrupção por mudança de estado 32
Interrupção do Timer 0 32
Interrupção do Timer 1 33
Interrupção do Timer 2 33
Interrupção do Timer 3 33
Interrupção da porta paralela (PSP) 33
Interrupção dos conversores AD 33
Interrupção de recepção da USART 33
Interrupção de transmissão da USART 33
Interrupção da comunicação serial (SPI e I^2C) 33
Interrupção do CCP1 (Capture/Compare/PWM) 34
Interrupção do CCP2 (Capture/Compare/PWM) 34
Interrupção do FSCM (Fail Save Clock Monitor) 34
Interrupção de colisão de dados (Bus Collision) 34

Interrupção dos comparadores 34
Interrupção de fim de escrita na E²PROM/FLASH 34
Interrupção do HLVD (High/Low Voltage Detector) 34
Operação com as interrupções 34
Demais recursos e periféricos 36
Características elétricas 38

CAPÍTULO 3 - RESUMO DO SET DE INSTRUÇÕES 39
Os termos utilizados 39
A construção dos nomes das instruções 41
O resumo das instruções 41

CAPÍTULO 4 - AS PRIMEIRAS EXPLORAÇÕES (I/Os E TIMERS) 44
Introdução 44
Teoria e recursos do PIC 44
Estudo das PORTAS 44
Estudo dos timers 49
Maximização de pino 57
Lógica do exemplo 59
Esquema elétrico 60
Fluxograma 61
Código 62
Dicas e comentários 69
Exercícios propostos 69

CAPÍTULO 5 - VARREDURA DE DISPLAY DE QUATRO DÍGITOS 70
Introdução 70
Teoria e recursos do PIC 70
Lógica do exemplo 72
Esquema elétrico 74
Fluxograma 75
Código 78
Dicas e comentários 88
Exercícios propostos 88

CAPÍTULO 6 - OPERAÇÃO COM DISPLAY DE CRISTAL LÍQUIDO (LCD) 89
Introdução 89
Teoria e recursos do PIC 89
Inicialização do LCD 91
Comandos do LCD 93
Lógica do exemplo 96
Esquema elétrico 97
Fluxograma 98
Código 102
Dicas e comentários 110
Exercícios propostos 110

CAPÍTULO 7 - CONVERSOR ANALÓGICO-DIGITAL INTERNO 111
Introdução 111
Teoria 111
Recursos do PIC 113
Adequação do capacitor 116
Desligamento do capacitor 116
Conversão 116
Religamento do capacitor 116
Nova adequação do capacitor 117
Lógica do exemplo 120
Esquema elétrico 120
Fluxograma 121
Código 122
Dicas e comentários 130
Exercícios propostos 130

CAPÍTULO 8 - CONVERSOR ANALÓGICO-DIGITAL POR RC 131
Introdução 131
Teoria e recursos do PIC 131
Lógica do exemplo 133
Esquema elétrico 135
Fluxograma 136
Código 137
Dicas e comentários 144
Exercícios propostos 144

CAPÍTULO 9 - OS MÓDULOS CCP (CAPTURE/COMPARE/PWM) 145
Introdução 145
Teoria e recursos do PIC 145
Modo Capture 146
Modo Compare 148
Modo PWM 149
Módulo PWM com capacidade aumentada (ECCP) 153
Lógica do exemplo 156
Esquema elétrico 157
Fluxograma 158
Código 160
Dicas e comentários 168
Exercícios propostos 168

CAPÍTULO 10 - TRABALHO COM AS MEMÓRIAS NÃO VOLÁTEIS 169
Introdução 169
Teoria e recursos do PIC 169
Escrita na E^2PROM (dados) 171
Leitura da E^2PROM (dados) 171
Escrita na FLASH (programa) 172

Leitura da FLASH (programa) 173
Tratamento da interrupção de final de escrita na E²PROM e FLASH 173
Lógica do exemplo 174
Esquema elétrico 175
Fluxograma 176
Código 178
Dicas e comentários 189
Exercícios propostos 190

CAPÍTULO 11 - COMUNICAÇÃO SERIAL 1 - SPI e I²C 191
Introdução 191
Teoria e recursos do PIC para SPI 191
Teoria para I²C 197
Condição de Start 198
Condição de Stop 198
Condição de Re-Start 199
Condição de Acknowledge (ACK) 199
Transmissão de endereço 199
Transmissão de dados 200
Pausas 200
Diagramas de tempo 200
Recursos do PIC para I²C 203
Modo Slave 203
Modo Master 209
Lógica do exemplo 215
Esquema elétrico 216
Fluxograma 217
Código 222
Dicas e comentários 232
Exercícios propostos 233

CAPÍTULO 12 - COMUNICAÇÃO SERIAL 2 - USART 234
Introdução 234
Teoria 234
Modo assíncrono 234
Modo síncrono 236
Recursos do PIC 237
Modo assíncrono 241
Modo síncrono 242
Funções adicionais na USART 244
Lógica do exemplo 244
Esquema elétrico 245
Fluxograma 246
Código 248
Dicas e comentários 256
Exercícios propostos 256
CAPÍTULO 13 - OUTRAS CARACTERÍSTICAS 257

Introdução 257
Comunicação paralela (PSP) 257
High/Low Voltage Detect (HLVD) 259
Comparadores analógicos 260
Módulo gerador de tensão de referência 261
Watchdog Timer (WDT) 262
Power-on Reset (POR) 263
Power-up Timer (PWRT) 263
Oscilator Start-up Timer (OST) 264
Brown-out Reset (BOR) 264
SLEEP (Power-down Mode) 264
Controle de resets 266
Oscilador 266
Oscilador primário 267
Oscilador secundário 267
Bloco oscilador interno 267
Configuração do sistema de clock 267
Fail Save Clock Monitor (FSCM) 269
Dicas para o oscilador externo 269
Sistema de proteção do código (Code Protection) 270
Registradores de identificação (IDs) 270
Sistema de emulação In-Circuit (Debugger Mode) 271
Proteção de escrita interna da FLASH 271
Gravação In-Circuit (ICSP) 271
Gravação em baixa tensão (Low Voltage Programming) 272

CAPÍTULO 14 - IMPLEMENTAÇÃO DE UM SISTEMA DE MEDIÇÃO DE TEMPERATURA 273
Introdução 273
O sistema 273
O sensor de temperatura 273
O aquecimento 274
O resfriamento 274
Comunicação serial 274
Considerações gerais 275
Esquema elétrico 275
Fluxograma 277
Código 281

APÊNDICE A - DETALHAMENTO DOS REGISTRADORES ESPECIAIS (SFRs) 297

APÊNDICE B - DETALHAMENTO DOS CONFIGURATIONS BITS 323

APÊNDICE C - ESQUEMA ELÉTRICO DA PLACA PROPOSTA (McLab2) 328

BIBLIOGRAFIA 331

ÍNDICE REMISSIVO 333

» Sobre o material disponível na Internet

O material disponível no site da Editora Érica contém códigos-fonte dos exemplos apresentados, respostas dos exercícios, data sheets dos componentes adicionais (LCD e memória), listagem dos operadores e diretrizes do MPASM e um software para testes da comunicação serial via RS232, desenvolvido pelos autores. É necessário ter instalado em sua máquina Adobe Acrobat 8.0 e MPLAB 8.40 ou superior.

PIC18F.exe - 1.050KB

Procedimento para download

Acesse o site da Editora Érica Ltda.: www.editoraerica.com.br. A transferência do arquivo disponível pode ser feita de duas formas:

- ✓ **Por meio do módulo pesquisa**. Localize o livro desejado, digitando palavras-chave (nome do livro ou do autor). Aparecerão os dados do livro e o arquivo para download. Com um clique o arquivo executável é transferido.
- ✓ **Por meio do botão "Download"**. Na página principal do site, clique no item "Download". É exibido um campo no qual devem ser digitadas palavras-chave (nome do livro ou do autor). Aparecem o nome do livro e o arquivo para download. Com um clique o arquivo executável é transferido.

Procedimento para descompactação

Primeiro passo: após ter transferido o arquivo, verifique o diretório em que se encontra e dê um duplo-clique nele. Aparece uma tela do programa WINZIP SELF-EXTRACTOR que conduz ao processo de descompactação. Abaixo do Unzip To Folder há um campo que indica o destino dos arquivos que serão copiados para o disco rígido do seu computador.

C:\ PIC18F

Segundo passo: prossiga a instalação, clicando no botão Unzip, o qual se encarrega de descompactar o arquivo. Logo abaixo dessa tela, aparece a barra de status que monitora o processo para que você acompanhe. Após o término, outra tela de informação surge, indicando que os arquivos foram descompactados com sucesso e estão no diretório criado. Para sair dessa tela, clique no botão OK. Para finalizar o programa WINZIP SELF-EXTRACTOR, clique no botão Close.

» Prefácio

Ao receber o convite do Daniel para escrever o prefácio de seu novo livro, *Desbravando o PIC18 - Recursos Avançados*, fiquei muito feliz e ao mesmo tempo surpreso, pois utilizo também o livro, Desbravando o PIC - Ampliado e Atualizado para PIC 16F628A, em minhas aulas no Instituto Federal de Ciência e Tecnologia de São Paulo, no curso de Tecnologia em Automação Industrial. De verdade, se não fosse o livro e a forma como ele foi organizado, eu teria muita dificuldade em apresentar não só o microcontrolador, mas também em organizar os conceitos de forma a aplicá-los nos diversos projetos com que os alunos se envolvem e como gostam de criar.

Ao ler este novo livro, logo decidi adotá-lo em minhas aulas, pois segue a mesma maneira de ensinar.

Apresenta o hardware do microcontrolador PIC18F4520 e o conjunto de instruções de forma clara e objetiva, o que facilita as aulas iniciais de todo curso de microcontroladores que devem ser o mais dinâmicas possível, a fim de motivar os alunos a gostarem do que leem e, deste modo, buscar cada vez mais aprender como se fossem mesmo "desbravar" o desconhecido.

Seguindo o pensamento e os anseios de muitos professores que trabalham com o ensino de microcontroladores, podemos encontrar nos capítulos deste livro os projetos básicos que são desenvolvidos a partir de uma situação problema, o escopo de como resolvê-la, o fluxograma para a solução, o circuito elétrico e, finalmente, o código muito bem documentado, o que possibilita diversas atividades, e se pensarmos um pouco, podemos tornar o livro dinâmico, propondo diversos desafios a partir dele. Com esta maneira de ensinar os alunos se desenvolvem rápido na proposta de cada capítulo e, ao final, incrementam seus próprios projetos.

Os circuitos principais estão contemplados neste livro, e ao término da leitura e do aprendizado a maioria dos alunos começa a pensar em desenvolver seu próprio kit para as mais diversas aplicações de sistemas embarcados.

Este livro nos guia por algumas das tarefas mais comuns e importantes do dia a dia dos desenvolvedores e nós, como profissionais da educação tecnológica, vamos preparando os alunos para o mercado de sistemas embarcados, tão promissor para a engenharia nacional na geração de empregos e riqueza.

Na minha época de faculdade, um professor aconselhou a classe a nunca ler livros bons, e isso causou muito espanto, mas logo o sábio mestre completou: leia apenas os melhores.

Ao ler este novo livro, percebi mais uma vez o cuidado com os conteúdos e a forma didática como foi escrito. Parabenizo a você, Daniel, por mais esta obra que trata como se fosse um filho.

Prof. MSc. Walter Augusto Varella

Instituto Federal de Ciência e Tecnologia de São Paulo - Unidade de Cubatão
Unipaulistana - Centro Universitário Paulistano
UNIBAN - Universidade Bandeirante de São Paulo
Faculdade Anchieta - São Bernardo do Campo
Consultor em Tecnologias de Redes e Segurança - Wave Consultoria

» Sobre os autores

Daniel Rodrigues de Sousa é formado em Engenharia Elétrica pela Universidade Cruzeiro do Sul, em São Paulo, SP. Atua desde 1998 na área de hardware e software. Desenvolveu diversos trabalhos na área de reabilitação, engenharia clínica, instrumentação industrial e automação comercial. É autor e coautor de diversos artigos acadêmicos na área de eletrônica de potência (motores de relutância chaveados) e inteligência artificial (redes neurais e lógica fuzzy), alguns publicados em congressos internacionais. Possui experiência com microcontroladores PIC, microcontroladores ARM7 (LPC2xxx), linguagens de programação Assembly e C, além de hardware em geral. Atualmente, trabalha na Mosaico, dedicando-se a projetos na área educacional, engenharia, produtos e suporte técnico. Ministra palestras e cursos de microcontroladores PIC e LPC213x. Possui o blog www.portalmcu.com.br, dedicado a dicas e projetos envolvendo microcontroladores e circuitos em geral.

David José de Souza é formado em Engenharia Mecânica pela Universidade Santa Cecília, trabalhando há quatro anos no setor. Trabalhou durante três anos no segmento de informática pela Canal 1 Informática. Especializou-se nos microcontroladores da Microchip em virtude de sua grande utilização nos laboratórios da Mosaico. Foi responsável pelas especificações de projetos, com as quais adquiriu experiência necessária para poder ministrar palestras e cursos sobre o assunto.

Nicolás César Lavinia é engenheiro eletricista com especialização em Eletrônica, formado pela Escola de Engenharia Mauá. Cursou mestrado no Laboratório de Automação e Controle da Escola Politécnica da Universidade de São Paulo. Possui treinamento Master em Microcontroladores PIC, promovido pela própria Microchip em Arizona, nos EUA. Trabalha na área de desenvolvimento de projetos e consultoria há mais de quinze anos. Atualmente, é sócio e diretor técnico da Mosaico, responsável pela coordenação do laboratório de desenvolvimento. Como coautor deste livro, expõe a vasta experiência adquirida no desenvolvimento de projetos eletrônicos.

1

Introdução

» Nossos objetivos

Quando criamos nossa primeira obra sobre este tema, o livro Desbravando o PIC - Ampliado e Atualizado para PIC 16F628A, dedicamos muitas horas de trabalho para que tudo saísse conforme o intento e que o sucesso alcançado por essa literatura seja o reconhecimento da qualidade do nosso trabalho. Entretanto, era intenção, desde o começo, que a literatura disponível para o PIC fosse muito mais abrangente.

O objetivo deste livro é exatamente este: complementar o conhecimento dos usuários do PIC. Enquanto o primeiro destina-se ao aprendizado da linguagem assembly, utilizando-se para isso dos recursos disponíveis no PIC16F628A, é intenção agora aprimorar seus conhecimentos nos demais recursos da família PIC, além da integração com periféricos práticos e totalmente necessários nos dias de hoje, como varredura de displays, LCD e comunicação RS-232, apresentando a família PIC18.

Embasado no título Conectando o PIC 16F877A - Recursos Avançados, publicado pela Editora Érica, a mesma metodologia utilizada para a família PIC18 é uma forma de abranger todos os periféricos e ao mesmo tempo estudar a nova família de microcontroladores PIC de 8 bits de alto desempenho tanto em velocidade quanto em novos periféricos e facilidade de operação.

Com esses conhecimentos seus projetos podem dar um salto evolutivo em relação aos recursos empregados, possibilitando uma melhor interface com o usuário e o mundo exterior de uma forma geral. Com os novos recursos do PIC seus programas ficam menores e mais eficientes, seus circuitos mais enxutos e robustos, seus sistemas muito mais avançados e poderosos. A partir de agora seus limites são sua dedicação e criatividade.

» Pré-requisitos sugeridos

Tudo isso exige um conhecimento prévio. Por isso, para um bom desempenho neste treinamento, é necessário que você já tenha conhecimento da linguagem assembly do PIC, assim como as ferramentas de trabalho: o MpLab e um sistema de gravação. Como já dissemos, nossa intenção não é o aprendizado da programação básica, mas sim a exploração de recursos avançados e os sistemas complementares. Caso você ainda não domine a linguagem, recomendamos a leitura da obra Desbravando o PIC, também publicada pela Editora Érica.

Editora Érica - Desbravando o Microcontrolador PIC18 Recursos Avançados - Daniel R. de Sousa, David José de Souza e Nicolás César Lavinia - 1ª Edição

» A didática do sistema

Mais uma vez houve, de nossa parte, grande empenho em relação à didática e ordem cronológica aplicada ao texto. Consideramos essa sistemática muito importante para o seu desempenho, principalmente numa aplicação autodidata.

Desta vez resolvemos dividir o conhecimento em quatro grandes partes:

Introdução

Esta parte é destinada ao esclarecimento das características do PIC utilizado como base no estudo. Antes de nos aprofundarmos nos recursos, devemos conhecer bem todas as facetas do PIC em questão, tais como seus registradores especiais, características elétricas, pinagem etc.

Prática

Todas as funções e recursos estudados estão divididos em dez capítulos práticos. Cada um desses capítulos possui uma estruturação padronizada, baseada nos seguintes tópicos:

- ✓ **Introdução**: explicações gerais sobre os assuntos estudados.
- ✓ **Teoria**: quando necessário, detalha o embasamento teórico obrigatório para o entendimento do assunto em questão.
- ✓ **Recursos do PIC**: detalhamento dos recursos do PIC que serão utilizados nos exemplos expostos, com a explicação do funcionamento, registradores envolvidos etc. Muitas vezes esse tópico pode estar agrupado ao anterior.
- ✓ **Lógica do exemplo**: explica a lógica utilizada para a montagem do programa de exemplo.
- ✓ **Esquema elétrico**: apresenta o esquema elétrico necessário à implementação do programa. Esse esquema é totalmente compatível com o hardware apresentado no Apêndice C (McLab2), porém apresenta somente os elementos relacionados com o capítulo em questão.
- ✓ **Fluxograma**: esse tópico apresenta o fluxograma de operação para complementar a lógica descrita anteriormente e facilitar o entendimento do exemplo.
- ✓ **Código**: o código propriamente dito, em assembly, para o exemplo do capítulo.
- ✓ **Dicas e comentários**: dicas, comentários importantes e demais desdobramentos resultantes da utilização do exemplo apresentado.
- ✓ **Exercícios propostos**: outros problemas e questões que podem ser discutidos e resolvidos com os recursos já aprendidos.

Um fato importante é que, com essa sistemática, todos os tópicos relacionados a um determinado assunto encontram-se agrupados no mesmo capítulo, desde o embasamento teórico até o exemplo estudado e os exercícios propostos. Desta maneira, cada capítulo torna-se independente dos demais, podendo ser estudado separadamente.

Projeto final

Ao término dos capítulos práticos é apresentado um projeto real que envolve a medição de temperatura, com atuadores para aquecimento e resfriamento. Trata-se de um exemplo capaz de utilizar os recursos adicionais da placa apresentada no Apêndice C (McLab2), baseando-se nas práticas aprendidas durante o treinamento.

Apêndices

Como já conhecemos as características marcantes de nossos leitores, os apêndices tornaram-se obrigatórios em uma boa literatura didática. Eles serão de grande valia quando você se transformar em um programador dedicado, pois tornam-se fontes de consulta permanente. Desta forma, tentamos colocar nos apêndices informações úteis ao dia a dia do programador, seguindo, sempre que possível, a formatação e os termos utilizados nas literaturas técnicas da própria Microchip.

» Padrões textuais

Para facilitar o entendimento do livro, alguns padrões textuais são utilizados:

- ✓ Todos os nomes de registradores, bits ou pinos são apresentados em negrito com a seguinte formatação: **STATUS**, **STATUS<Z>**, **STATUS<3:0>**, **Pino<1>**.
- ✓ Quando no nome do registrador houver um **x** (letra x minúscula), é porque existe mais de um registrador com o mesmo nome, diferenciados por um número que substitui o **x**: **ICxBUF** = **IC1BUF**, **IC2BUF** ... **IC5BUF**.
- ✓ Nomes de funções e comandos de programação estão em letra minúscula e fonte diferenciada: `while, if()`.
- ✓ Números em hexadecimal são finalizados com a letra h: 0001FEh.
- ✓ Devido ao grande número de termos em língua estrangeira, não adotaremos nenhuma padronização diferenciada para eles.

2

O PIC18F4520

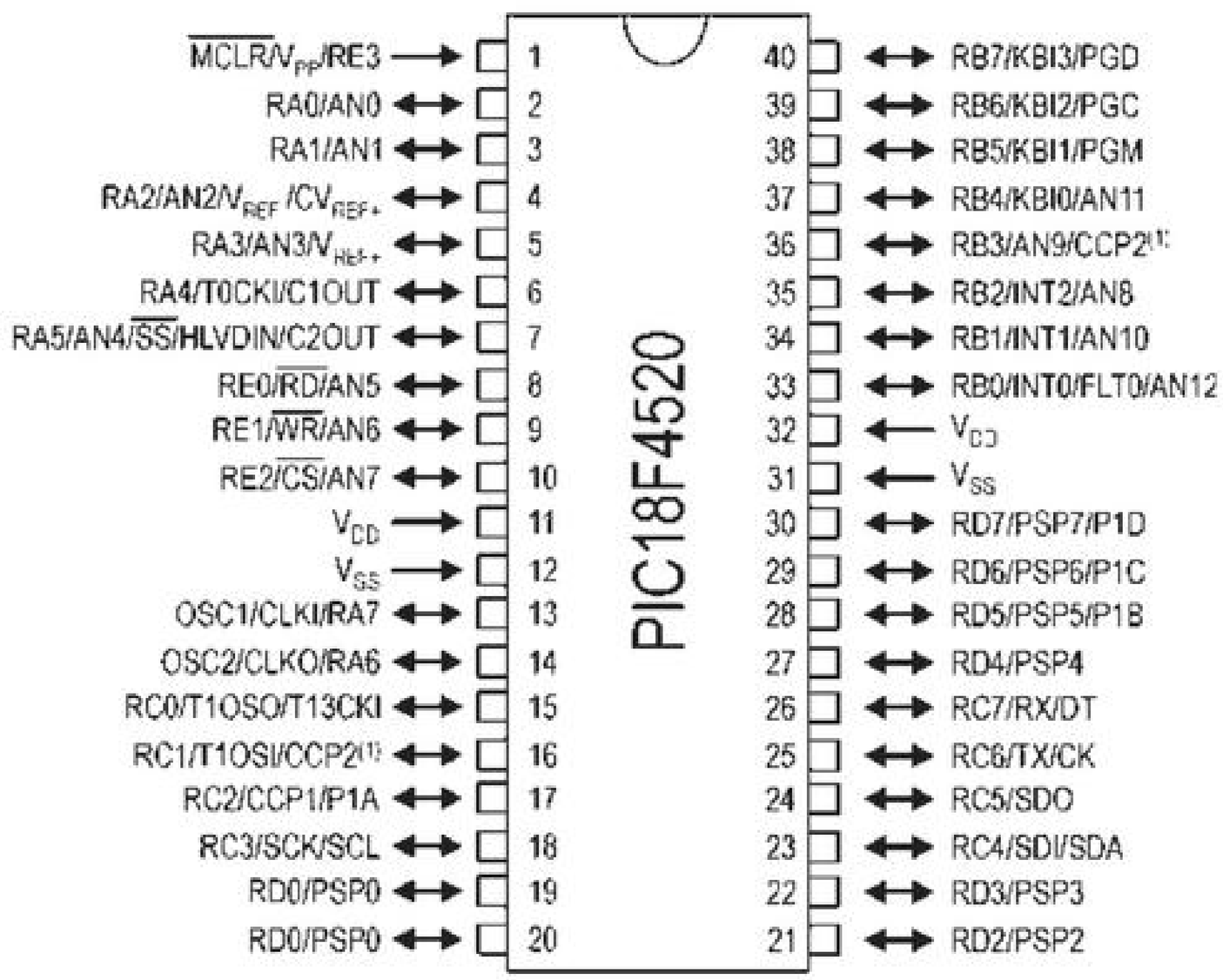
Para podermos conhecer os demais recursos da família PIC, devemos adotar como base de estudo um modelo extremamente poderoso que agrupe de uma só vez o maior número possível de recursos disponíveis. Desta vez escolhemos trabalhar com o modelo 18F4520. Vejamos agora o porquê:

- ✓ Microcontrolador de 40 pinos, o que possibilita a montagem de um hardware complexo e capaz de interagir com diversos recursos e funções ao mesmo tempo;
- ✓ 75 instruções (83 instruções contando o set de instruções estendido);
- ✓ 36 portas configuráveis como entrada ou saída;
- ✓ 20 interrupções disponíveis;
- ✓ Memória de programação E^2PROM FLASH;
- ✓ Memória de programa com 32 kbytes, com capacidade de escrita e leitura pelo próprio código interno;
- ✓ Memória E^2PROM (não volátil) interna com 256 bytes;
- ✓ Memória RAM com 1536 bytes;
- ✓ Quatro timers (1x8 bits e 3x16 bits);
- ✓ Comunicações seriais: SPI, I^2C e USART;
- ✓ Conversores analógicos de 10 bits (13x) e comparadores analógicos (2x);
- ✓ Dois módulos CCP: Capture, Compare e PWM;
- ✓ Programação in-circuit (alta e baixa tensão);
- ✓ Power-on Reset (POR) interno;
- ✓ Brown-out Reset (BOR) interno;
- ✓ Diversos recursos adicionais.

A grande vantagem da família PIC é que todos os modelos possuem um set de instruções bem parecido, assim como mantêm muitas semelhanças entre suas características básicas. Desta maneira, ao conhecermos e estudarmos o PIC18F4520, familiarizamo-nos com todos os microcontroladores da Microchip, o que torna a migração para outros modelos muito mais simples. Os recursos estudados encontram-se disponíveis em diversos outros modelos.

Editora Érica - Desbravando o Microcontrolador PIC18 Recursos Avançados - Daniel R. de Sousa, David José de Souza e Nicolás César Lavinia - 1ª Edição

» Pinagem



Legenda: ↔ Entrada ou saída
→ Somente entrada
← Somente saída

(1) Função CCP2 configurável nos Configurations Bits

» Nomenclatura dos pinos

Para entendermos melhor o significado de cada nomenclatura utilizada para identificação dos pinos, montamos uma tabela que descreve detalhes da sua utilização.

Nome do Pino	Núm. Pino	I/O/P	Tipo	Descrição
V_{SS}	12/31	P	-	GND.
V_{DD}	11/32	P	-	Alimentação positiva.
				PORTA (I/Os digitais bidirecionais e sistema analógico):
RA0 / AN0	2	I/O	TTL	RA0: I/O digital ou entrada analógica AN0.
RA1 / AN1	3	I/O	TTL	RA1: I/O digital ou entrada analógica AN1.
RA2 / AN2 / V_{REF}– / CV_{REF}	4	I/O	TTL	RA2: I/O digital ou entrada analógica AN2 ou tensão negativa de referência analógica.
RA3 / AN3 / V_{REF}+	5	I/O	TTL	RA3: I/O digital ou entrada analógica AN3 ou tensão positiva de referência analógica.
RA4 / T0CKI / C1OUT	6	I/O	ST	RA4: I/O digital ou entrada externa do contador TMR0 ou saída do comparador 1.
RA5 / SS / HLVDIN / AN4 / C2OUT	7	I/O	TTL	RA5: I/O digital ou entrada analógica AN4 ou habilitação externa (slave select) para comunicação SPI ou tensão de referência para o módulo HLVD ou saída do comparador 2.

Nome do Pino	Núm. Pino	I/O/P	Tipo	Descrição
OSC2 / CLKO / RA6	14	I/O	TTL	Saída para cristal. Os cristais ou ressonadores devem ser ligados aos pinos OSC1 e OSC2. Saída com onda quadrada em ¼ da frequência imposta em OSC1 quando em modo RC. Essa frequência equivale aos ciclos de máquina internos. RA6: I/O digital.
OSC1 / CLKIN / RA7	13	I/O	TTL/ CMOS(4)	Entrada para cristal. Entrada para osciladores externos (híbridos ou RC). RA7: I/O digital.
				PORTB (I/Os digitais bidirecionais). Todos os pinos desse PORT possuem pull-up interno que pode ser ligado/desligado pelo software:
RB0 / INT0 / FLT0 / AN12	33	I/O	TTL/ST(1)	RB0: I/O digital com interrupção externa ou entrada analógica AN12.
RB1 / INT1 / AN10	34	I/O	TTL/ST(1)	RB1: I/O digital com interrupção externa ou entrada analógica AN10.
RB2 / INT2 / AN8	35	I/O	TTL/ST(1)	RB2: I/O digital com interrupção externa ou entrada analógica AN8.
RB3 / AN9 / CCP2	36	I/O	TTL/ST(1)	RB3: I/O digital ou entrada analógica AN9 ou entrada do Capture2 ou saídas para Compare2/PWM2.
RB4 / KBI0 / AN11	37	I/O	TTL	RB4: I/O digital com interrupção por mudança de estado.
RB5 / KBI1 / PGM	38	I/O	TTL	RB5: I/O digital com interrupção por mudança de estado ou entrada para programação em baixa tensão (5V).
RB6 / KBI2 / PGC	39	I/O	TTL/ST(2)	RB6: I/O digital com interrupção por mudança de estado ou clock da programação serial ou pino de in-circuit debugger.
RB7 / KBI3 / PGD	40	I/O	TTL/ST(2)	RB7: I/O digital com interrupção por mudança de estado ou data da programação serial ou pino de in-circuit debugger.
				PORTC (I/Os digitais bidirecionais):
RC0 / T1OSO / T1CKI	15	I/O	ST	RC0: I/O digital ou saída do oscilador externo para TMR1 ou entrada de incremento para TMR1.
RC1 / T1OSI / CCP2	16	I/O	ST	RC1: I/O digital ou entrada do oscilador externo para TMR1 ou entrada do Capture2 ou saídas para Compare2/PWM2.
RC2 / CCP1 / P1A	17	I/O	ST	RC2: I/O digital ou entrada do Capture1 ou saídas para Compare1/PWM1 ou saída do ECCP, canal A.
RC3 / SCK / SCL	18	I/O	ST	RC3: I/O digital ou entrada/saída de clock para comunicação serial SPI / I²C.
RC4 / SDI / SDA	23	I/O	ST	RC4: I/O digital ou entrada de dados para SPI ou via de dados (entrada/saída) para I²C.
RC5 / SDO	24	I/O	ST	RC5: I/O digital ou saída de dados para SPI.
RC6 / TX / CK	25	I/O	ST	RC6: I/O digital ou TX (transmissão) para comunicação USART assíncrona ou clock para comunicação síncrona.

Nome do Pino	Núm. Pino	I/O/P	Tipo	Descrição
				PORTD (I/Os digitais bidirecionais) ou porta de comunicação paralela:
RD0 / PSP0	19	I/O	TTL/ST(3)	RD0: I/O digital ou dado 0 (comunicação paralela).
RD1 / PSP1	20	I/O	TTL/ST(3)	RD1: I/O digital ou dado 1 (comunicação paralela).
RD2 / PSP2	21	I/O	TTL/ST(3)	RD2: I/O digital ou dado 2 (comunicação paralela).
RD3 / PSP3	22	I/O	TTL/ST(3)	RD3: I/O digital ou dado 3 (comunicação paralela).
RD4 / PSP4	27	I/O	TTL/ST(3)	RD4: I/O digital ou dado 4 (comunicação paralela).
RD5 / PSP5 / P1B	28	I/O	TTL/ST(3)	RD5: I/O digital ou dado 5 (comunicação paralela) ou saída do ECCP, canal B.
RD6 / PSP6 / P1C	29	I/O	TTL/ST(3)	RD6: I/O digital ou dado 6 (comunicação paralela) ou saída do ECCP, canal C.
RD7 / PSP7 / P1D	30	I/O	TTL/ST(3)	RD7: I/O digital ou dado 7 (comunicação paralela) ou saída do ECCP, canal D.
				PORTE (I/Os digitais bidirecionais e sistema analógico):
RE0 / RD / AN5	8	I/O	TTL/ST(3)	RE0: I/O digital ou controle de leitura da porta paralela ou entrada analógica AN5.
RE1 / WR / AN6	9	I/O	TTL/ST(3)	RE1: I/O digital ou controle de escrita da porta paralela ou entrada analógica AN6.
RE2 / CS / AN7	10	I/O	TTL/ST(3)	RE2: I/O digital ou habilitação externa da porta paralela ou entrada analógica AN7.
MCLR / V_{PP} /RE3	1	I/P	ST	Master Clear (reset) externo. O microcontrolador só funciona quando esse pino encontra-se em nível alto. Entrada para tensão de programação (13V). RE0: I/O digital (somente entrada)

Legenda:

I	=	Input (entrada)
O	=	Output (saída)
I/O	=	Input/Output (entrada ou saída)
P	=	Power (alimentação)
-	=	Não utilizado
TTL	=	Entrada TTL
ST	=	Entrada Schmitt Trigger

Notas:

(1) Esta entrada é ST, somente quando configurado como interrupção externa.

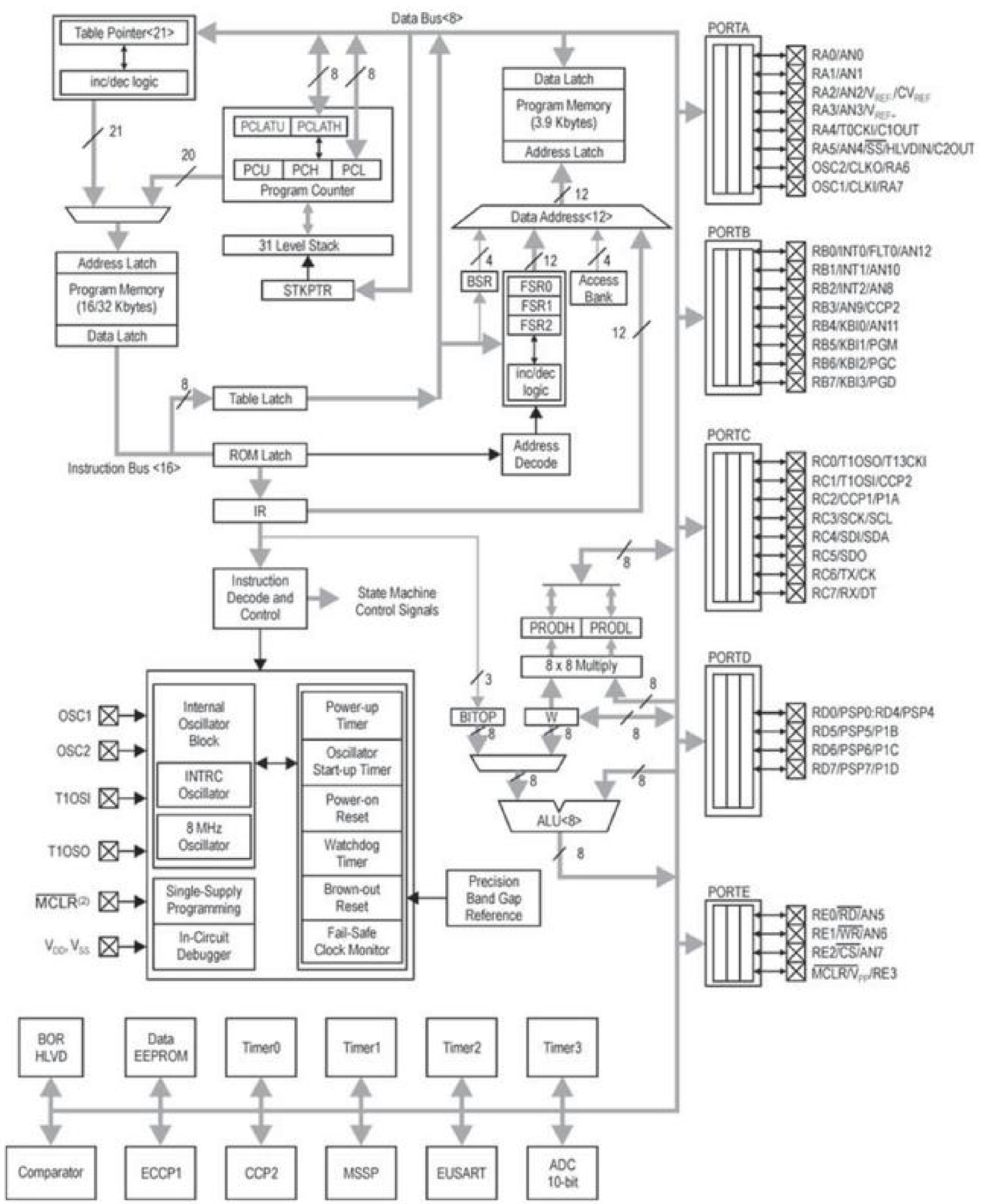
(2) Esta entrada é ST, somente durante o modo de programação serial.

(3) Esta entrada é ST, quando configurado como I/O de uso geral e TTL quando usado em modo de porta paralela.

(4) Esta entrada é ST quando em modo RC e CMOS nos demais casos.

» A estruturação interna

Já que estamos nos aprofundando nas características do PIC18F4520, que tal darmos uma olhadinha em seu interior? O que há dentro desse componente? Como é sua estrutura? O diagrama de blocos mostrado a seguir detalha todos os periféricos e comunicações que compõem esse tão poderoso microcontrolador:



No diagrama em blocos do PIC18F4520 podemos visualizar algumas diferenças em relação à família PIC16. No centro do diagrama vemos a unidade lógica aritmética (ALU) e próximo a ela temos um hardware de multiplicação 8 x 8 bits com resultado em 16 bits, disposto em dois registradores próprios (PRODL e PRODH).

Totalmente ao canto superior esquerdo há um novo recurso que facilita e muito o processo de leitura e escrita da memória Flash, que é o Table Pointer. Com o uso de registradores apropriados a essa finalidade temos o acesso total da memória de programa, tanto para leitura quanto para escrita.

Ainda no canto superior esquerdo existe o contador de programa (Program Counter). O acesso a ele é feito em três registradores (PCU, PCH e PCL) e é atualizado o valor durante a execução do programa. Temos ainda 31 níveis de pilha (Level Stack) e a memória de programa (Flash) de 32 kbytes.

Na parte central superior há a memória de dados (RAM) de 3,9 kbytes (máximo), em que temos os registradores especiais (SFR - Special Function Register). Para acesso indireto a essa memória existem três registradores FSRs e para acesso direto existe o BSR.

Totalmente à direita temos os cinco ports (PORTA, PORTB, PORTC, PORTD e PORTE) e na parte inferior do diagrama de blocos estão os periféricos como timers, EEPROM, USART etc. Ainda na parte inferior temos um bloco que controla o oscilador, Watchdog Timer, Brown-out Reset e outros circuitos auxiliares.

» Os ciclos de máquina

Para quem já trabalha com microcontroladores, possivelmente este assunto já é conhecido, mas, devido à sua grande importância, achamos melhor revisá-lo e, quem sabe, aperfeiçoar seus conhecimentos.

O básico que deve ser conhecido é que, neste microcontrolador (assim como na maioria dos modelos da linha PIC), o clock interno (CK_{INT}) é equivalente ao clock externo (CK_{EXT} ou F_{OSC}) dividido por 4. Com isso teremos:

$$CK_{INT} = \frac{CK_{EXT}}{4}$$

Desta forma, quando trabalhamos, por exemplo, com um cristal de 4 MHz, o PIC trabalha internamente com uma frequência de 1 MHz. Para nós, mais importante que o clock interno é o período dessa frequência (o inverso dela), que é equivalente ao tempo de duração de um ciclo de máquina, popularmente chamado de CM ou T_{CY}. Assim sendo, teremos:

$$T_{CY} = \frac{1}{CK_{INT}}$$

Continuando o exemplo de 4 MHz externo, o ciclo de máquina (CM) será de 1 µs.

Mas, afinal de contas, qual o motivo da necessidade da divisão do clock externo por 4? Isso pode ser visualizado e explicado quando entendermos o funcionamento interno do processador.

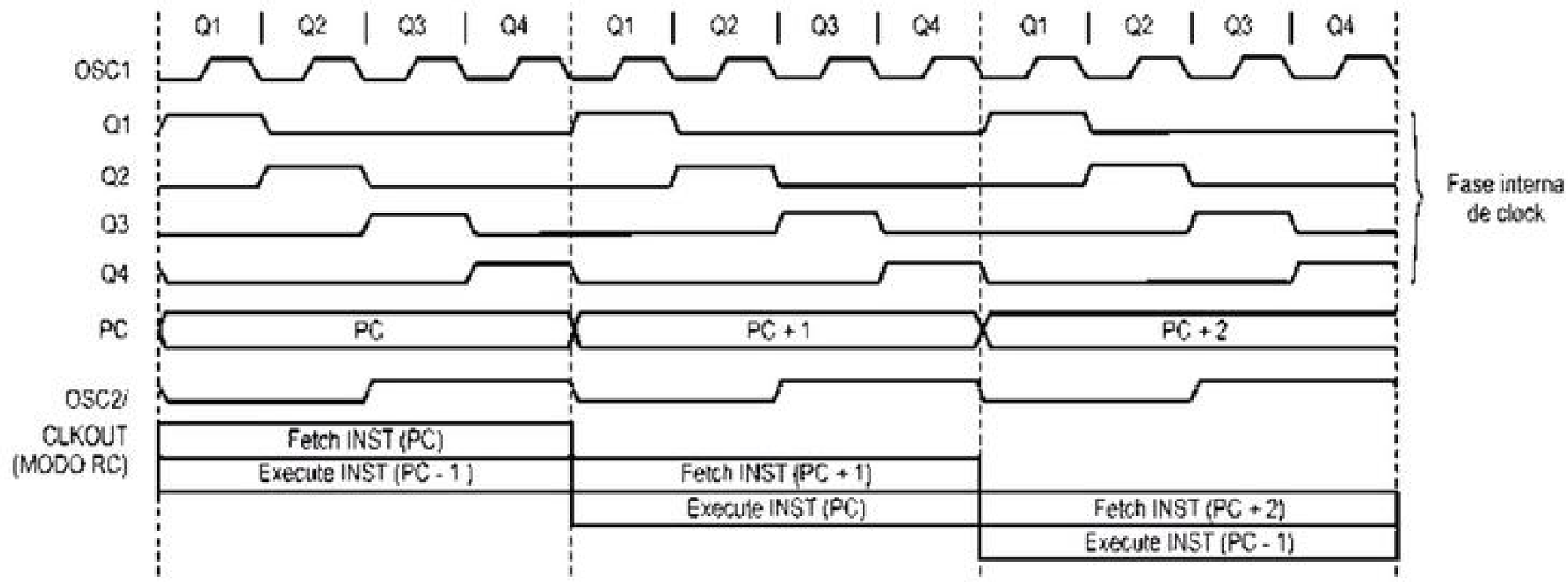
Para a execução de uma única instrução pelo processador, várias operações precisam ser realizadas. Como não existe capacidade de processamento paralelo, essas operações são executadas em subciclos do ciclo de máquina, originados pela divisão do clock externo. Esses subciclos são chamados de Q1, Q2, Q3 e Q4.

O PC (contador de programa) é incrementado automaticamente no início de Q1. Durante o decorrer dos quatro tempos (Q1 a Q4), a instrução previamente carregada para dentro da ULA é executada, sendo trocadas informações com a memória de dados e o registrador Work sempre que necessário. Por último, ao final do tempo Q4, a próxima instrução (lembre-se de que o PC já foi incrementado) é buscada da memória de programa e armazenada na ULA.

Essa característica de buscar a informação em um ciclo de máquina e executá-la no próximo é conhecida como Pipeline. Ela permite que quase todas as instruções sejam executadas em apenas um ciclo, gastando assim 1 µs (continuando com o exemplo de 4 MHz) e tornando o sistema muito mais rápido. As únicas exceções referem-se às instruções que geram "saltos" no program counter (PC), como chamadas de rotinas e retornos. Ao executar essas instruções, o Pipeline deve ser primeiramente limpo para depois poder ser carregado novamente com o endereço correto, consumindo para isso dois ciclos de máquina.

Editora Érica - Desbravando o Microcontrolador PIC18 Recursos Avançados - Daniel R. de Sousa, David José de Souza e Nicolás César Lavinia - 1ª Edição

O diagrama seguinte ilustra os quatro subciclos (Q1 a Q4) e o conceito de Pipeline.



» A memória de programa

A memória de programa do PIC18F4520 é do tipo FLASH, regravável eletronicamente com escrita rápida. Uma outra característica muito importante da memória de programa desse PIC é que ela pode ser acessada por software, possibilitando que o programa seja reescrito dinamicamente, ou, ainda, que ela seja usada como expansão da memória E²PROM de dados. Vamos conhecer, então, as demais características dessa memória:

Vetor de reset

Trata-se do primeiro endereço da memória de programa que será executado após um Start-up ou Reset. Neste modelo, o **vetor de reset** encontra-se no endereço 0000h.

Vetores de interrupção

Esse PIC possui 20 tipos de interrupções diferentes. Entretanto, quando qualquer uma delas acontece (se todas as condições necessárias forem favoráveis), o programa é desviado para um ponto específico, que é denominado **vetor de interrupção**. Esses vetores encontram-se nas posições 0008h (alta prioridade) e 0018h (baixa prioridade).

Tamanho da memória

Esse PIC possui uma memória de 32 kbytes de memória de programa, isto é, capacidade de 16.384 instruções de dois bytes cada. Diferentemente do que ocorre com os microcontroladores da família PIC16, em que cada instrução ocupa um único espaço de 14 bits, na família PIC18 podemos ter instruções que ocupam dois ou quatro bytes.

Essa nova disposição das instruções nesse microcontrolador resolve um problema antigo de gerenciamento da família PIC16 nas instruções CALL e GOTO, na qual era necessário fazer um acerto de página de memória antes da execução da instrução. Como essas instruções na família PIC18 possuem o tamanho de quatro bytes, ela pode acessar qualquer ponto da memória de programa sem a necessidade de acerto de página. Isso facilita o desenvolvimento do software.

Editora Érica - Desbravando o Microcontrolador PIC18 Recursos Avançados - Daniel R. de Sousa, David José de Souza e Nicolás César Lavinia - 1ª Edição

Pilha (stack)

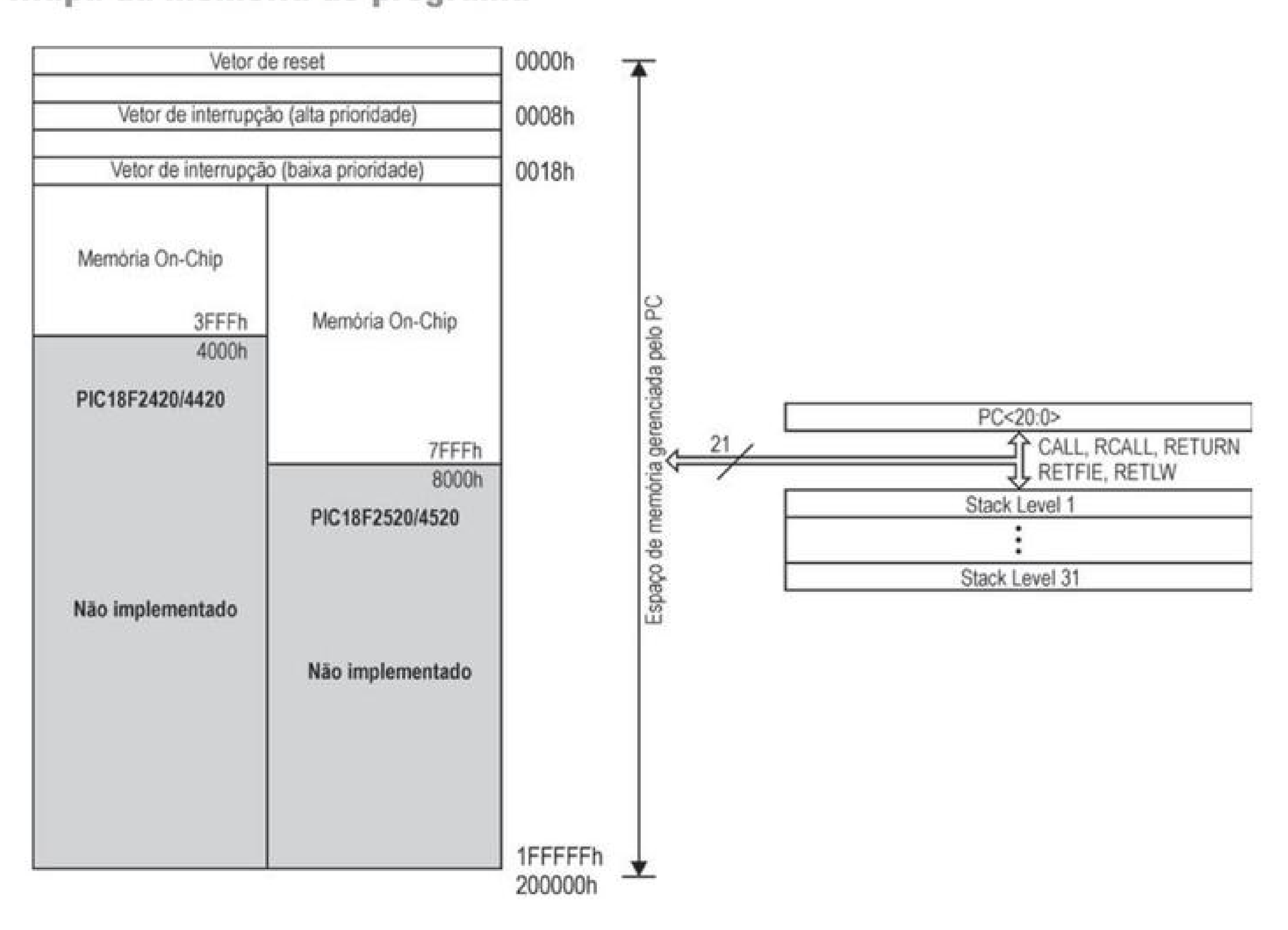
A pilha é um local, fisicamente separado da memória de programa, onde serão armazenados os endereços de retorno quando utilizarmos instruções de desvio para rotinas de chamada. Quando o programa é desviado para o começo de uma rotina através da instrução correta (CALL ou interrupção), o endereço seguinte ao ponto que estava sendo executado é armazenado na pilha para que, ao fim da rotina, o programa possa retornar novamente ao ponto em que estava.

Esse PIC possui uma pilha de 31 níveis, isto é, é possível o armazenamento de 31 endereços de retorno, possibilitando 31 desvios consecutivos. Caso se tente chamar um número de rotinas maior que o tamanho da pilha, o endereço de retorno mais antigo é perdido. É possível configurar nos Configurations Bits do microcontrolador para que ocorra um reset por estouro da pilha.

Esse microcontrolador possui as instruções PUSH e POP e com o uso dos registradores adequados é possível ter uma parte da pilha na memória de dados, aumentando a pillha.

Outra característica que devemos saber sobre a pilha é o seu tamanho em bits, que representa a quantidade de endereços que o sistema de retorno automático pode gerenciar. No nosso caso, a pilha armazena endereços de 21 bits, o que é suficiente para o gerenciamento de até 2 M de memória de programa. Como o PIC possui 32 kbytes, os retornos através da pilha não apresentam nenhum tipo problema.

Mapa da memória de programa



» A memória de dados volátil (RAM)

A memória de dados também é conhecida como RAM e serve para guardar as variáveis e os registradores utilizados pelo programa. Essa memória armazena dados de 8 bits e é volátil, ou seja, quando o PIC é desenergizado, ela é automaticamente perdida. Podemos dividi-la em dois grupos que são estudados a seguir: **registradores especiais** e **registradores de uso geral**.

Registradores especiais

Nesta região da memória encontram-se todos os registradores especiais, denominados SFRs (Special Functions Registers) e que são utilizados pelo microcontrolador para a execução do programa e processamentos da ULA. Esses registradores podem ser escritos e lidos tanto pelo usuário quanto pelo hardware e servem também para a configuração de muitas funções e para a utilização de todos os periféricos. O mapa da memória apresenta a posição de todos os registradores especiais e seus devidos nomes. Para um melhor detalhamento de suas propriedades e funções específicas, o Apêndice A deve ser consultado.

Registradores de uso geral

Trata-se de uma área destinada ao armazenamento de variáveis definidas pelo usuário para serem escritas e lidas pelo programa. O PIC18F4520 possui 1.536 bytes disponíveis para uso geral.

Tamanho da memória e bancos

A arquitetura do PIC está preparada para operar com uma RAM de até 4096 bytes. Entretanto, verificaremos que nem toda memória está disponível ao usuário. São 1.536 bytes para uso geral, 109 bytes para registradores especiais e 2.451 endereços indisponíveis.

Quanto aos endereços indisponíveis, trata-se de posições não implementadas pela Microchip e que, no caso de uma leitura, retornam somente zeros (0).

Muito bem, ao detalharmos o tamanho da memória de dados, surgimos com o termo **Banco**. Mas, afinal, o que isso significa?

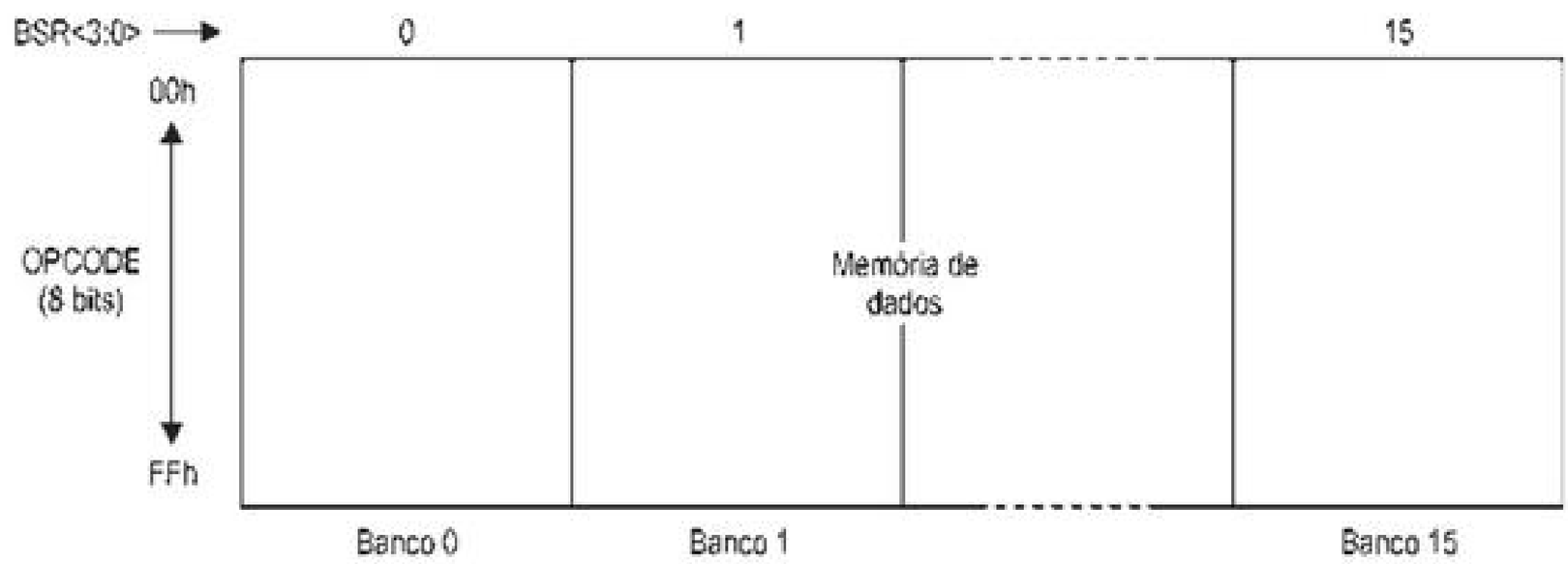
O mesmo tipo de problema de gerenciamento de endereços que enfrentamos ao trabalhar com a memória de programas surge novamente para a memória de dados. Vejamos os Opcodes de algumas instruções que trabalham com a RAM:

Instrução	Opcode	Observações
MOVF	`0001 00da ffff ffff`	d = destino a = bit de acesso à RAM ffff ffff = endereço na RAM
MOVWF	`0110 111a ffff ffff`	

Fica fácil perceber que só existem 8 bits para endereçamento de todos os registradores. Acontece que, com esse número de bits, só podemos gerenciar 256 endereços diferentes. Por isso, a Microchip dividiu a memória de dados em grupos de 256 bytes, aos quais deu o nome de **Banco**.

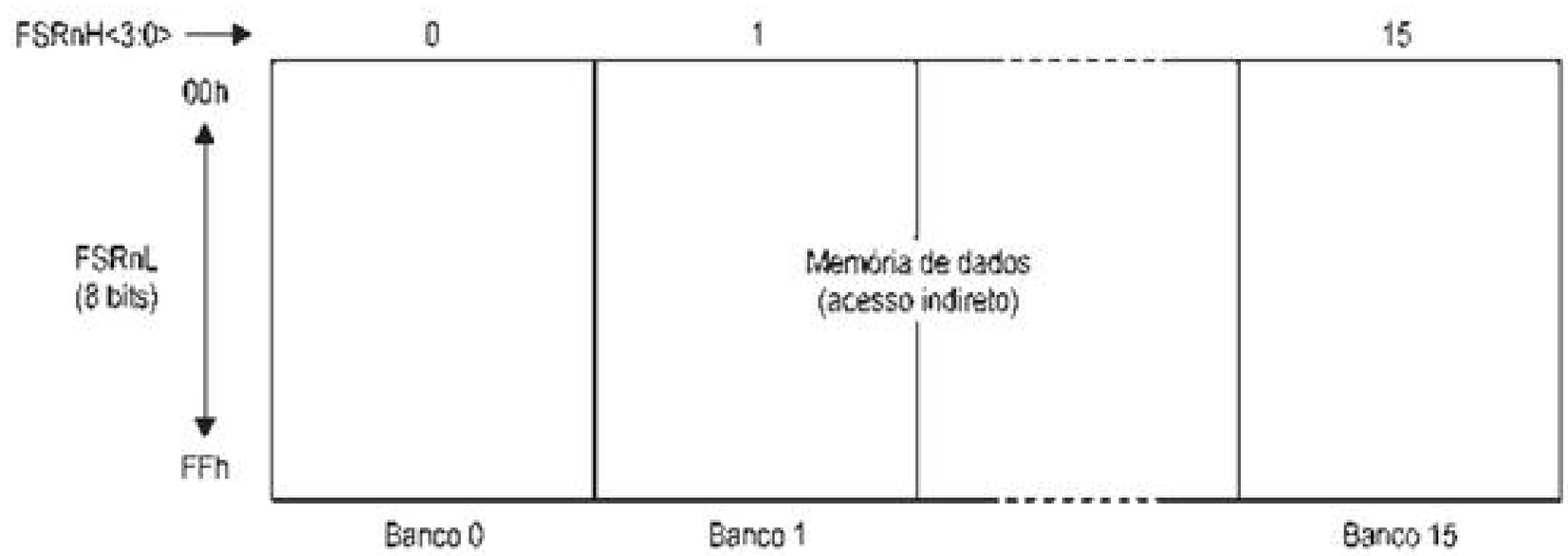
No caso do nosso PIC, a memória de dados é composta por 16 bancos. Para a seleção do banco de memória temos o registrador **BSR**.

Devido a essa estruturação, antes de utilizarmos qualquer comando que trabalhe com os registradores (RAM) devemos comutar para o banco onde ele se encontre.



Para a seleção do banco de memória podemos escrever o valor correspondente ao banco desejado no registrador **BSR** ou utilizar a instrução MOVLB (mover literal ao registrador **BSR**).

Já que estamos falando dos bancos de memória, existe outro caso em que a seleção do banco é feita de maneira diferente. Trata-se do acesso indireto através dos registradores **FSRn** (em que **n** é um número de 0 a 2) para apontar para um endereço da memória de dados e **INDFn** para leitura ou escrita do endereço apontado pelo **FSRn**. Temos três conjuntos de registradores para acesso indireto, facilitando assim a programação. Note que no caso do **FSRn** há dois registradores correspondentes ao byte inferior (os oito primeiros bits) e o byte superior (os quatro últimos bits).



Além do registrador **INDFn** para acesso ao conteúdo apontado pelo **FSRn**, existem ainda os seguintes registradores que complementam o acesso indireto à memória de dados:

- ✓ **POSTDECn**: acessa o dado apontado pelo **FSRn** e depois decrementa **FSRn**.
- ✓ **POSTINCn**: acessa o dado apontado pelo **FSRn** e depois incrementa **FSRn**.
- ✓ **PREINCn**: incrementa **FSRn** e depois acessa o dado do **FSRn** incrementado.
- ✓ **PLUSWn**: acessa o dado apontado pelo **FSRn** com um offset acertado anteriormente em **W** (faixa de –127 até 128). Não altera o **FSRn**.

Ainda falando em acesso à memória de dados, temos outra novidade: o **Access Bank**.

Esse recurso facilita o acesso de informação na memória de dados. É um "banco virtual" que reúne os primeiros 128 bytes do banco 0 (memória de uso geral) e os últimos 128 bytes do banco 15 (memória com os registradores especiais). Para programas que não utilizam muitas variáveis ou para acesso a registradores especiais não há necessidade de mudança de banco utilizando o registrador **BSR** ou pela instrução MOVLB. Nas instruções em Assembly orientada a byte da família PIC18 existe um novo operando que informa se o acesso será pelo Access Bank ou via banco de memória. Vejamos algumas instruções e exemplos de utilização:

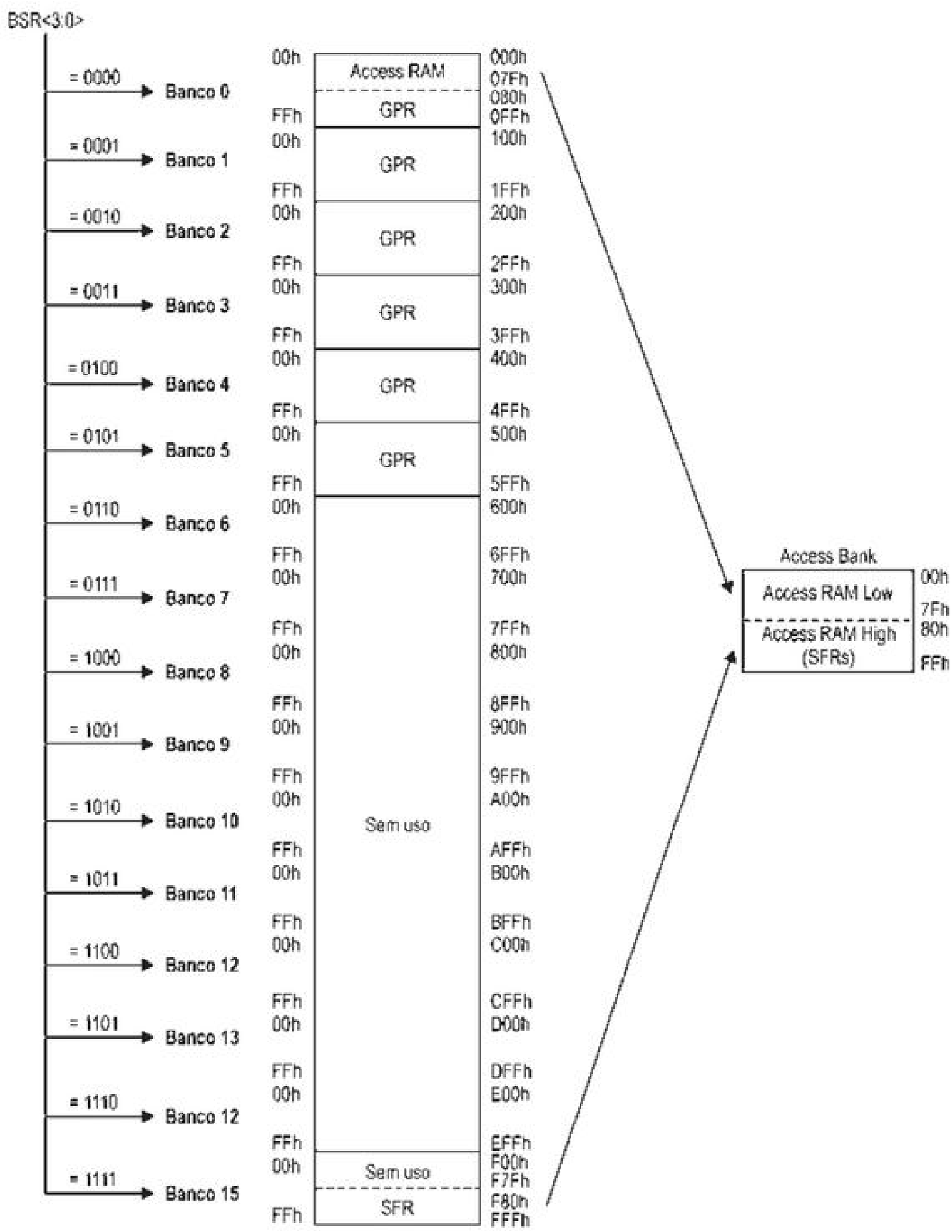
Editora Érica - Desbravando o Microcontrolador PIC18 Recursos Avançados - Daniel R. de Sousa, David José de Souza e Nicolás César Lavinia - 1ª Edição

Instrução	Sintaxe	Exemplo	Observações
MOVF	`MOVF f, d, a`	`MOVF STATUS, W, ACCESS`	d = destino (W ou F) a = bit de acesso à RAM (ACCESS - Access Bank BANKED - Acesso por banco) f = endereço na RAM
MOVWF	`MOVWF f, a`	`MOVWF PORTB, BANKED`	

Caso o programador opte por não utilizar o argumento **a** (bit de acesso a RAM), por padrão esse bit aponta para o Access Bank, facilitando assim a programação. Por exemplo:

`MOVF STATUS, W` é o mesmo que `MOVF STATUS, W, ACCESS`

Mapa da memória de dados



Mapa da memória de dados (SFRs)

Endereço	Nome	Endereço	Nome	Endereço	Nome	Endereço	Nome
FFFh	TOSU	FDFh	INDF2(1)	FBFh	CCPR1H	F9Fh	IPR1
FFEh	TOSH	FDEh	POSTINC2(1)	FBEh	CCPR1L	F9Eh	PIR1
FFDh	TOSL	FDDh	POSTDEC2(1)	FBDh	CCP1CON	F9Dh	PIE1
FFCh	STKPTR	FDCh	PREINC2(1)	FBCh	CCPR2H	F9Ch	__(2)
FFBh	PCLATU	FDBh	PLUSW2(1)	FBBh	CCPR2L	F9Bh	OSCTUNE
FFAh	PCLATH	FDAh	FSR2H	FBAh	CCP2CON	F9Ah	__(2)
FF9h	PCL	FD9h	FSR2L	FB9h	__(2)	F99h	__(2)
FF8h	TBLPTRU	FD8h	STATUS	FB8h	BAUDCON	F98h	__(2)
FF7h	TBLPTRH	FD7h	TMR0H	FB7h	PWM1CON(3)	F97h	__(2)
FF6h	TBLPTRL	FD6h	TMR0L	FB6h	ECCP1ASN(3)	F96h	TRISE(3)
FF5h	TABLAT	FD5h	T0CON	FB5h	CVRCON	F95h	TRISD(3)
FF4h	PRODH	FD4h	__(2)	FB4h	CMCON	F94h	TRISC
FF3h	PRODL	FD3h	OSCCON	FB3h	TMR3H	F93h	TRISB
FF2h	INTCON	FD2h	HLVDCON	FB2h	TMR3L	F93h	TRISA
FF1h	INTCON2	FD1h	WDTCON	FB1h	T3CON	F91h	__(2)
FF0h	INTCON3	FD0h	RCON	FB0h	SPBRGH	F90h	__(2)
FEFh	INDF0(1)	FCFh	TMR1H	FAFh	SPBRG	F8Fh	__(2)
FEEh	POSTINC0(1)	FCEh	TMR1L	FAFh	RCBRG	F8Eh	__(2)
FEDh	POSTDEC0(1)	FCDh	T1CON	FADh	TXREG	F8Dh	LATE(3)
FECh	PREINC0(1)	FCCh	TMR2	FACh	TXSTA	F8Ch	LATD(3)
FEBh	PLUSW0(1)	FCBh	PR2	FABh	RCSTA	F8Bh	LATC
FEAh	FSR0H	FCAh	T2CON	FAAh	__(2)	F8Ah	LATB
FE9h	FSR0L	FC9h	SSPBUF	FA9h	EEADR	F89h	LATA
FE8h	WREG	FC8h	SSPADD	FA9h	EEDATA	F88h	__(2)
FE7h	INDF1(1)	FC7h	SSPSTAT	FA9h	EECON2(1)	F87h	__(2)
FE6h	POSTINC1(1)	FC6h	SSPCON1	FA9h	EECON1	F86h	__(2)
FE5h	POSTDEC1(1)	FC5h	SSPCON2	FA9h	__(2)	F85h	__(2)
FE4h	PREINC1(1)	FC4h	ADRESH	FA9h	__(2)	F84h	PORTE(3)
FE3h	PLUSW1(1)	FC3h	ADRESL	FA9h	__(2)	F83h	PORTD(3)
FE2h	FSR1H	FC2h	ADCON0	FA9h	IPR2	F82h	PORTC
FE1h	FSR1L	FC1h	ADCON1	FA9h	PIR2	F81h	PORTB
FE0h	BSR	FC0h	ADCON2	FA9h	PIE2	F80h	PORTA

Nota:

(1) Não é um registrador físico.

(2) Não implementado.

(3) Não implementado nos modelos de 28 pinos.

O detalhamento de todos os SFRs pode ser encontrado no Apêndice A.

» Memórias não voláteis (E^2PROM e FLASH)

A memória não volátil até poderia ser tratada diretamente no tópico anterior, no qual falávamos da memória de dados. Entretanto, para o PIC18F4520 o acesso a essa memória é mais poderoso. Em alguns modelos de PIC, quando falamos dos recursos internos para manipulação dessa memória, nos referimos somente a um bloco de bytes disponíveis para o usuário como uma memória de dados não volátil, conhecida como E^2PROM. Esse PIC possui esse bloco e seu acesso é feito como nos demais modelos. O que ele tem a mais então? É que o mesmo sistema de manipulação da E^2PROM também pode ser utilizado para escrever e ler na memória de programa (FLASH).

Editora Érica - Desbravando o Microcontrolador PIC18 Recursos Avançados - Daniel R. de Sousa, David José de Souza e Nicolás César Lavinia - 1ª Edição

Com esse poderoso recurso podemos utilizar a memória de programa como uma expansão da E²PROM. Outro benefício desse sistema é que podemos escrever na memória de programa, possibilitando alterar o próprio software de controle. Isso permite técnicas e recursos extremamente avançados. Mas cuidado! Esse benefício também pode ser uma arma letal, pois podemos alterar o programa erroneamente, travando o sistema permanentemente.

As técnicas e os exemplos para acessarmos essas memórias encontram-se no Capítulo 10. No momento, basta conhecermos as características de ambas:

Características	Dados	Programa
Tipo	8 bits	8 bits
Tamanho	256 bytes	32 k (menos o programa)

» As interrupções

Esse PIC possui um total de 20 interrupções diferentes. Diferentemente da família PIC16, a família PIC18 possui dois vetores de interrupção, um de alta prioridade e outra de baixa prioridade. Com esse recurso podemos ter uma interrupção dentro de outra, interessante quando existem periféricos cuja interrupção deve ser executada de forma rápida.

Na sequência acompanhe as fontes de interrupção e como elas ocorrem.

Interrupções externas

Essa interrupção é gerada por um sinal externo ligado aos pinos **RB0**, **RB1** ou **RB2**, caso ele esteja configurado como entrada. Desta maneira, podemos identificar e processar imediatamente um sinal externo. Ela é utilizada para diversas finalidades, como, por exemplo, para a comunicação entre micros, garantindo o sincronismo, para o reconhecimento de botão ou outro sinal do sistema que necessite de uma ação imediata. Essas interrupções podem ser configuradas para a borda de subida ou para a borda de descida.

Interrupção por mudança de estado

Ao contrário da interrupção externa, a interrupção por mudança de estado acontece em ambas as bordas (subida e descida). Na verdade, essa interrupção é sensível à diferença de nível existente entre o pino e o latch interno. Está relacionada às portas **RB4**, **RB5**, **RB6** e **RB7** simultaneamente. Por isso, se essas portas forem configuradas como entradas, a mudança de estado em qualquer uma delas vai gerar a interrupção. Como o latch só é atualizado quando a porta é lida, a leitura é obrigatória para que o evento da interrupção pare de ocorrer. Esse tipo de interrupção pode ser utilizado, por exemplo, para criar um sincronismo com a rede de 60 Hz, para o controle de um triac ou outro sistema semelhante.

Interrupção do Timer 0

Essa interrupção acontece sempre que o contador **TMR0** (Timer 0) estoura, ou seja, como ele é um contador de 8 bits, sempre que ele passar de FFh para 00h (modo 8 bits) ou FFFFh para 0000h (modo de 16 bits). É utilizada normalmente para a contagem de tempo. Como ela pode acontecer a qualquer momento, a contagem de tempo fica precisa, não dependendo de análises constantes durante o programa para garantir que o tempo seja contado. Os Timers são estudados no Capítulo 4.

Interrupção do Timer 1

Acontece sempre que o Timer 1 (**TMR1**) estoura. Este é um contador de 16 bits. Os Timers são estudados no Capítulo 4.

Interrupção do Timer 2

Acontece sempre que o Timer 2 (**TMR2**) acrescido do seu Postscaler estoura. Este é um contador de 8 bits que possui um Prescaler para incrementar o registrador e um Postscaler para gerar a interrupção. Os Timers são estudados no Capítulo 4.

Interrupção do Timer 3

Essa interrupção acontece sempre que o Timer 3 (**TMR3**) estourar. Este é um contador de 16 bits. Os Timers são estudados no Capítulo 4.

Interrupção da porta paralela (PSP)

Está diretamente ligada à porta paralela do tipo escravo (PSP - Parallel Slave Port) e acontece sempre que uma operação de escrita ou leitura dessa porta é completada. O estudo completo desse recurso é feito no Capítulo 13.

Interrupção dos conversores AD

Acontece quanto uma conversão AD (Analógica/Digital) é completada. O estudo completo desse recurso é realizado no Capítulo 7.

Interrupção de recepção da USART

Essa interrupção indica o término da recepção de um dado pela USART (Universal Synchronous Asynchronous Receiver Transmitter). O estudo completo desse recurso é realizado no Capítulo 12.

Interrupção de transmissão da USART

Essa interrupção indica o esvaziamento do buffer relacionado à transmissão de um dado pela USART (Universal Synchronous Asynchronous Receiver Transmitter). O estudo completo desse recurso é feito no Capítulo 12.

Interrupção da comunicação serial (SPI e I^2C)

O segundo sistema de comunicação serial desse PIC, além da USART, denomina-se MSSP (Master Synchronous Serial Port), e possui dois modos de comunicação: SPI e I^2C. Sempre que um dado é transmitido ou recebido por essa porta serial, essa interrupção acontece. Ela também acontece em muitas outras situações, estudadas no Capítulo 11.

Editora Érica - Desbravando o Microcontrolador PIC18 Recursos Avançados - Daniel R. de Sousa, David José de Souza e Nicolás César Lavinia - 1ª Edição

Interrupção do CCP1 (Capture/Compare/PWM)

Está vinculada ao primeiro sistema CCP, que engloba os recursos Capture1, Compare1 e PWM1. O estudo completo desse recurso é feito no Capítulo 9.

Interrupção do CCP2 (Capture/Compare/PWM)

Essa interrupção está vinculada ao segundo sistema CCP, que engloba os recursos Capture2, Compare2 e PWM2. O estudo completo desse recurso é realizado no Capítulo 9.

Interrupção do FSCM (Fail Save Clock Monitor)

Essa interrupção está vinculada à falha do oscilador principal, podendo assim mudar de oscilador para não parar o programa. O estudo completo desse recurso é feito no Capítulo 13.

Interrupção de colisão de dados (Bus Collision)

Serve para informar o sistema sobre colisões de dados na comunicação I^2C. O estudo completo desse recurso é feito no Capítulo 11.

Interrupção dos comparadores

Essa interrupção está diretamente ligada à mudança de estado dos comparadores internos do PIC. O estudo completo desse recurso é realizado no Capítulo 13.

Interrupção de fim de escrita na E^2PROM/FLASH

Como já foi visto anteriormente, alguns PICs possuem uma memória E^2PROM interna. Essa interrupção serve para detectarmos o final de uma rotina de escrita nessa memória. A utilização da interrupção não é obrigatória para que a escrita funcione, mas como a E^2PROM é lenta na hora de escrever, em alguns sistemas sua utilização pode ser necessária para evitar que o programa pare durante a escrita na E^2PROM. A interrupção também é válida para a memória FLASH (programa).

Interrupção do HLVD (High/Low Voltage Detector)

Essa interrupção serve para informar se houve uma queda de tensão ou uma subida de tensão no pino de alimentação do microcontrolador. O estudo completo desse recurso é realizado no Capítulo 13.

Operação com as interrupções

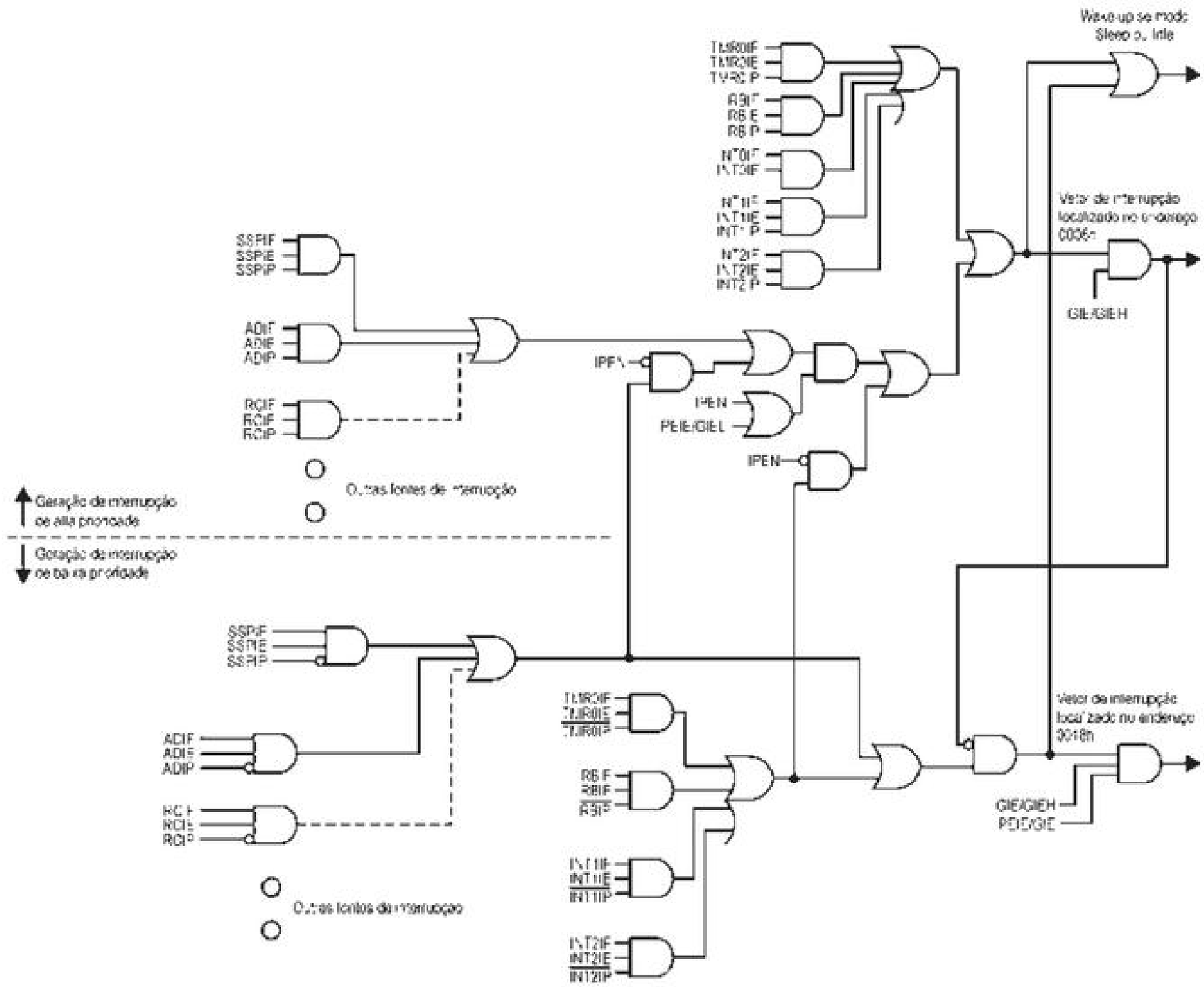
O primeiro conceito que devemos entender para o uso das interrupções é que, para elas realmente acontecerem, isto é, paralisarem o programa e desviarem para o vetor de interrupção, algumas questões devem ser verificadas:

1. Uma ação (ou evento) relacionada à interrupção deve acontecer.
2. Quando a ação acontece, o flag da interrupção é marcado.

3. Caso as chaves de habilitação da interrupção (geral e individual) estejam ligadas, o sistema é paralisado e desviado para o vetor de interrupção. O vetor de interrupção está relacionado à prioridade, baixa prioridade (endereço 0018h) e alta prioridade (endereço 0008h). Essa configuração é feita em bits específicos para o controle de prioridade.

Para habilitar o sistema de prioridade de interrupção, o bit **RCON<IPEN>** deve estar em 1. Por padrão, o bit **RCON<IPEN>** é igual a 0. Neste caso não existe prioridade de interrupção e qualquer pedido de interrupção é desviado para o endereço 0008h, correspondente ao vetor de alta prioridade.

A ação propriamente dita é exatamente o que queremos monitorar. Mas e as chaves de acesso? Trata-se de bits específicos que servem como chaves ON/OFF para ligar e desligar as interrupções, individualmente ou em grupo. O próximo diagrama ilustra todas as chaves, flags e bits de prioridade relacionados às interrupções.



Chaves	SFR	Bit
GIE/GIEH	INTCON	7
PEIE/GIEL	INTCON	6
TMR0IE	INTCON	5
INT0IE	INTCON	4
RBIE	INTCON	3
INT1IE	INTCON3	3
INT2IE	INTCON3	4
PSPIE	PIE1	7
ADIE	PIE1	6
RCIE	PIE1	5
TXIE	PIE1	4
SSPIE	PIE1	3
CCP1IE	PIE1	2
TMR2IE	PIE1	1
TMR1IE	PIE1	0
OSCFIE	PIE2	7
CMIE	PIE2	6
EEIE	PIE2	4
BLCIE	PIE2	3
HLVDIE	PIE2	2
TMR3IE	PIE2	1
CCP2IE	PIE2	0

Flags	SFR	Bit
-	-	-
-	-	-
TMR0IF	INTCON	2
INT0IF	INTCON	1
RBIF	INTCON	0
INT1IF	INTCON3	0
INT2IF	INTCON3	1
PSPIF	PIR1	7
ADIF	PIR1	6
RCIF	PIR1	5
TXIF	PIR1	4
SSPIF	PIR1	3
CCP1IF	PIR1	2
TMR2IF	PIR1	1
TMR1IF	PIR1	0
OSCFIF	PIR2	7
CMIF	PIR2	6
EEIF	PIR2	4
BLCIF	PIR2	3
HLVDIF	PIR2	2
TMR3IF	PIR2	1
CCP2IF	PIR2	0

Prioridade	SFR	Bit
-	-	-
-	-	-
TMR0IP	INTCON2	2
-	-	-
RBIP	INTCON2	0
INT1IP	INTCON3	6
INT2IP	INTCON3	7
PSPIP	IPR1	7
ADIP	IPR1	6
RCIP	IPR1	5
TXIP	IPR1	4
SSPIP	IPR1	3
CCP1IP	IPR1	2
TMR2IP	IPR1	1
TMR1IP	IPR1	0
OSCFIP	IPR2	7
CMIP	IPR2	6
EEIP	IPR2	4
BLCIP	IPR2	3
HLVDIP	IPR2	2
TMR3IP	IPR2	1
CCP2IP	IPR2	0

Agora que todas as chaves e flags foram demonstradas, vejamos mais alguns detalhes do tratamento de interrupções da família PIC18.

- ✓ Ao utilizar somente o vetor de baixa prioridade, devemos obrigatoriamente ativar as chaves **INTCON<GIE/GIEH>** e **INTCON<PEIE/GIEL>** para que a interrupção ocorra.
- ✓ Na interrupção de alta prioridade o bit **INTCON<PEIE/GIEL>** = 0 habilita apenas as interrupções externas, mudança de estado e timer 0.
- ✓ A interrupção externa 0 não possui bit de prioridade e a prioridade dele é sempre alta.
- ✓ A interrupção de alta prioridade pode interromper uma interrupção de baixa prioridade. Uma interrupção de baixa prioridade não interrompe uma interrupção de alta prioridade.
- ✓ Se utilizada apenas a interrupção de alta prioridade, temos ainda um recurso chamado shadow em que automaticamente o hardware salva e restaura os registradores **BSR**, **STATUS** e **WREG** (acumulador). Para isso, devemos utilizar a instrução `RETFIE` com a opção `FAST`. Não utilizar esse recurso quando usar a interrupção de alta e baixa prioridades simultaneamente; sendo assim, cada vetor deve ter sua rotina de salva contexto e recupera contexto.

» Demais recursos e periféricos

Todos os demais recursos e periféricos que estão disponíveis no PIC18F4520 são detalhados e estudados no decorrer dos capítulos (de 4 a 13). A tabela seguinte relaciona todos os itens a serem estudados e os respectivos capítulos em que eles aparecem detalhadamente.

Editora Érica - Desbravando o Microcontrolador PIC18 Recursos Avançados - Daniel R. de Sousa, David José de Souza e Nicolás César Lavinia - 1ª Edição

Recursos Internos do PIC										
Assunto	Capítulos									
	4	5	6	7	8	9	10	11	12	13
Portas	✔									
Porta paralela (PSP)										✔
E^2PROM (dados)							✔			
FLASH (programa)							✔			
Timer 0	✔									
Timer 1	✔									
Timer 2	✔									
Timer 3	✔									
CCP (Capture/Compare/PWM)						✔				
MSSP (SPI e I^2C)								✔		
USART									✔	
Conversor AD				✔						
Interrupção de Timer 0	✔									
Interrupção de Timer 1	✔									
Interrupção de Timer 2	✔									
Interrupção de Timer 3	✔									
Interrupção externas										
Interrupção de mudança de estado (RB4 - RB7)										
Interrupção da porta paralela (PSP)										✔
Interrupção dos conversores AD				✔						
Interrupção de recepção da USART									✔	
Interrupção de transmissão da USART									✔	
Interrupção da comunicação MSSP (SPI e I^2C)								✔		
Interrupção de CCP1						✔				
Interrupção de CCP2						✔				
Interrupção de fim de escrita na E^2PROM/FLASH							✔			
Interrupção de colisão de dados (Bus Collision)								✔		
Osciladores										✔
Sistemas de resets (POR, PWRT e BOR)										✔
Watchdog Timer (WDT)										✔
HLVD										✔
FSCM										✔
SLEEP										✔
Code Protection										✔
Gravação In-circuit										✔

Outros Recursos Estudados										
Assunto	Capítulos									
	4	5	6	7	8	9	10	11	12	13
Varredura de display de sete segmentos (quatro dígitos)		✔								
Operação com LCD padrão			✔							
Comunicação com memória E²PROM externa								✔		
Simulação de AD por meio de RC					✔					
Acionamento de buzzer	✔									
Entrada e saída no mesmo pino (botões e LEDs)	✔									
Acionamento de motor DC (ventilador)						✔				

» Características elétricas

Temperatura de trabalho ..... –40°C até +125°C

Temperatura de armazenamento ..... –65°C até 150°C

Tensão de trabalho ..... 4.2 V a 5.5 V

Tensão máxima no pino V_{DD} (em relação ao V_{SS}) ..... –0.3 V até 7.5 V

Tensão máxima no pino MCRL (em relação ao V_{SS}) ..... 0 até 13,25V

Tensão máxima nos demais pinos (em relação ao V_{SS}) ..... –0.3 V até (V_{DD} + 0.3 V)

Dissipação máxima de energia ..... 1.0 W

Corrente máxima de saída no pino V_{SS} ..... 300 mA

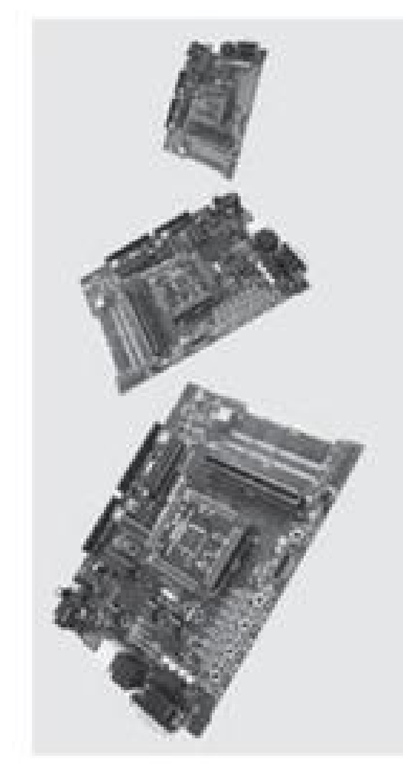
Corrente máxima de entrada no pino V_{DD} ..... 250 mA

Corrente máxima de entrada de um pino (quando em V_{SS}) ..... 25 mA

Corrente máxima de saída de um pino (quando em V_{DD}) ..... 25 mA

Corrente máxima de entrada do PORTA, PORTB, PORTC, PORTD e PORTE ..... 200 mA

Corrente máxima de saída do PORTA, PORTB, PORTC, PORTD e PORTE ..... 200 mA



3

Resumo do Set de Instruções

» Os termos utilizados

Para facilitar o aprendizado do set de instruções do PIC, é conveniente entendermos corretamente os termos utilizados na construção dos nomes das instruções e seus argumentos.

- ✓ **Work**: trata-se de um registrador temporário para as operações da ULA. No assembly do PIC ele é conhecido como **W**. Também é comum chamá-lo de acumulador.
- ✓ **File**: referência a um registrador (posição de memória) propriamente dito. Utilizamos a letra **F** para sua representação nos nomes de instruções e **f** nos argumentos delas.
- ✓ **Access**: referência ao acesso da RAM via Access Bank ou via banco através do registrador **BSR**. Utilizamos a letra **a** nos argumentos das instruções.
- ✓ **Literal**: um número qualquer, que pode ser escrito na forma decimal, hexadecimal ou binária. Utilizamos a letra **L** para sua representação nos nomes de instruções e **k** nos argumentos delas.
- ✓ **Destino**: o local onde deve ser armazenado o resultado da operação. Existem somente dois destinos possíveis: **f**, que guarda o resultado no próprio registrador passado como argumento ou **w**, que coloca o resultado em Work. Na verdade, na sintaxe das instruções o destino deve ser expressado pelos números 0 (w) e 1 (f). No entanto, como veremos mais adiante, as letras **f** e **w** são definidas no include para facilitar a programação.
- ✓ **Não**: refere-se a uma operação ou um estado oposto. Utilizamos a letra **N** para sua representação nos nomes das instruções.
- ✓ **Borrow**: refere-se ao bit Borrow (ou o "vem 1" de uma operação de subtração) do registrador STATUS. Utilizamos a letra **B** para sua representação nos nomes das instruções.
- ✓ **Carry**: refere-se ao bit Carry (ou o "vai 1" de uma operação de soma) do registrador STATUS. Utilizamos a letra **C** para sua representação nos nomes das instruções.
- ✓ **Overflow**: refere-se ao bit Overflow do registrador STATUS. Utilizamos as letras **OV** para sua representação nos nomes das instruções.
- ✓ **Bit**: refere-se a um bit específico dentro de um byte. Utilizamos a letra **B** para sua representação nos nomes das instruções e **b** nos argumentos delas.
- ✓ **Teste**: quando queremos testar o estado de um bit, para descobrirmos se ele é zero ou um. Utilizamos a letra **T** para representá-lo nos nomes das instruções.

Editora Érica - Desbravando o Microcontrolador PIC18 Recursos Avançados - Daniel R. de Sousa, David José de Souza e Nicolás César Lavinia - 1ª Edição

- ✓ **Skip**: significa "pulo", e é utilizado para criar desvios, pulando a próxima linha. Utilizamos a letra **S** para representá-lo nos nomes das instruções.
- ✓ **Set**: refere-se ao ato de setar um bit, isto é, torná-lo equivalente a UM. Utilizamos a letra **S** para representá-lo nos nomes das instruções.
- ✓ **Clear**: refere-se ao clear de um bit, isto é, torná-lo equivalente a ZERO. Utilizamos a letra **C** para representá-lo nos nomes das instruções.
- ✓ **Toggle**: refere-se ao ato de inverter o estado de um bit. Utilizamos a letra **T** para representá-lo nos nomes das instruções.
- ✓ **Zero**: algumas instruções podem gerar desvios se o resultado da operação efetuada for zero. Neste caso utilizamos a letra **Z** para indicar tal condição.
- ✓ **ADD**: somatória.
- ✓ **AND**: lógica "E".
- ✓ **CLR**: limpar, zerar (Clear).
- ✓ **COM**: complemento.
- ✓ **CPF**: comparar registrador.
- ✓ **DEC**: decremento de uma unidade.
- ✓ **EQ**: igual.
- ✓ **GT**: maior que (>).
- ✓ **INC**: incremento de uma unidade.
- ✓ **IOR**: lógica "OU".
- ✓ **LT**: menor que (<).
- ✓ **MOV**: mover, transferir para algum lugar.
- ✓ **MUL**: multiplicar.
- ✓ **RD**: leitura.
- ✓ **RL**: rotacionar 1 bit para a esquerda (rotation left).
- ✓ **RR**: rotacionar 1 bit para a direita (rotation right).
- ✓ **SET**: setar todos os bits de um registrador.
- ✓ **SUB**: subtração.
- ✓ **SWAP**: inversão entre as partes alta e baixa de um registrador.
- ✓ **TST**: teste.
- ✓ **TBL**: tabela.
- ✓ **XOR**: lógica "OU exclusivo".
- ✓ **WT**: escrita.

» A construção dos nomes das instruções

Com base nos termos demonstrados é muito mais fácil entender o significado de uma instrução por intermédio do seu nome, pois ele é composto pela junção desses termos. Por exemplo, digamos que você deseja decrementar o valor de um determinado registrador. A instrução que faz isso é composta pelos termos referentes à ação que você quer fazer:

Decrementar (DEC) um registrador (F) = DECF

Agora vamos fazer a análise ao contrário, isto é, partindo do nome de uma instrução, e descobrir para que ela serve:

DECFSZ = Decrementa (DEC) o registrador (F) e pula (S) se o resultado for zero (Z)

» O resumo das instruções

O Set de instrução do PIC18F4520 é exatamente o mesmo para todos os modelos de 16 bits, e é composto de 75 instruções divididas em cinco grupos: operações com registradores, operações com literais, operações com bits, controles e operações de leitura e escrita de dados na memória de programa.

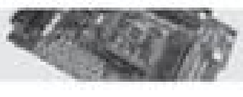
Operações com Registradores				
Instrução	**Argumentos**	**Ciclos**	**Status Afetado**	**Descrição**
ADDWF	f,d,a	1	C, DC, Z, OV, N	Soma W e f, guardando o resultado em d.
ADDWFC	f,d,a	1	C, DC, Z, OV, N	Soma W e f com Carry, guardando o resultado em d.
ANDWF	f,d,a	1	Z,N	Lógica "E" entre W e f, guardando o resultado em d.
CRLF	f,a	1	Z	Limpa f (todos os bit em 0).
COMF	f,d,a	1	Z,N	Calcula o complemento de f, guardando o resultado em d.
CPFSEQ	f,a	1 ou 2	Nenhum	Compara f com W, pula se igual (=).
CPFSGT	f,a	1 ou 2	Nenhum	Compara f com W, pula se maior (>).
CPFSLT	f,a	1 ou 2	Nenhum	Compara f com W, pula se menor (<).
DECF	f,d,a	1	C, DC, Z, OV, N	Decrementa f, guardando o resultado em d.
DECFSZ	f,d,a	1 ou 2	Nenhum	Decrementa f, guardando o resultado em d, e pula a próxima linha se o resultado for zero.
DCFSNZ	f,d,a	1 ou 2	Nenhum	Decrementa f, guardando o resultado em d, e pula a próxima linha se o resultado não for zero.
INCF	f,d,a	1	C, DC, Z, OV, N	Incrementa f, guardando o resultado em d.
INCFSZ	f,d,a	1 ou 2	Nenhum	Incrementa f, guardando o resultado em d, e pula a próxima linha se o resultado for zero.
INCFNZ	f,d,a	1 ou 2	Nenhum	Incrementa f, guardando o resultado em d, e pula a próxima linha se o resultado não for zero.
IORWF	f,d,a	1	Z, N	Lógica "OU" entre W e f, guardando o resultado em d.
MOVF	f,d,a	1	Z, N	Move f para d.
MOVFF	fs,fd	2	Nenhum	Move de fs (origem) para fd (destino).
MOVWF	f,a	1	Nenhum	Move W para f.
MULWF	f,a	1	Nenhum	Multiplica W e f, guardando o resultado em d.
NEGF	f,a	1	C, CD, Z, OV, N	Efetua o complemento de 2 de f
RLCF	f,d,a	1	C, Z, N	Rotaciona f 1 bit para a esquerda, guardando o resultado em d.

Operações com Registradores				
Instrução	**Argumentos**	**Ciclos**	**Status Afetado**	**Descrição**
RLNCF	f,d,a	1	Z, N	Rotaciona f 1 bit para a esquerda (sem o Carry), guardando o resultado em d.
RRCF	f,d,a	1	C, Z, N	Rotaciona f 1 bit para a direita, guardando o resultado em d.
RRNCF	f,d,a	1	Z, N	Rotaciona f 1 bit para a direita (sem o Carry), guardando o resultado em d.
SETF	f,d,a	1	Nenhum	Seta f (todos os bit em 1).
SUBFWB	f,d,a	1	C, DC, Z, OV, N	Subtrai f de W (W - f) com Borrow, guardando o resultado em d.
SUBWF	f,d,a	1	C, DC, Z, OV, N	Subtrai W de f (f - W), guardando o resultado em d.
SUBWFB	f,d,a	1	C, DC, Z, OV, N	Subtrai W de f (f - W) com Borrow, guardando o resultado em d.
SWAPF	f,d,a	1	Nenhum	Executa uma inversão entre as partes alta e baixa de f, guardando o resultado em d.
TSTFSZ	f,a	1 ou 2	Nenhum	Testa f, pula se 0.
XORWF	f,d,a	1	Z, N	Lógica "OU exclusivo" entre W e f, guardando o resultado em d.

Operações com Literais				
Instrução	**Argumentos**	**Ciclos**	**Status Afetado**	**Descrição**
ADDLW	k	1	C, DC, Z, OV, N	Soma k com W, guardando o resultado em W.
ANDLW	k	1	Z, N	Lógica "E" entre k e W, guardando o resultado em W.
IORLW	k	1	Nenhum	Lógica "OU" entre k e W, guardando o resultado em W.
LFSR	f,k	2	Nenhum	Move k para **FSRx**.
MOVLB	k	1	Nenhum	Move k para **BSR<3:0>**.
MOVLW	k	1	Nenhum	Move k para W.
MULLW	k	1	Nenhum	Multiplica k com W, guardando o resultado em W.
RETLW	k	2	Nenhum	Retorna de uma rotina, com k em W.
SUBLW	k	1	C, DC, Z, OV, N	Subtrai W de k (k - W), guardando o resultado em W.
XORLW	k	1	Z, N	Lógica "OU exclusivo" entre k e W, guardando o resultado em W.

Operações com Bits				
Instrução	**Argumentos**	**Ciclos**	**Status Afetado**	**Descrição**
BCF	f,b,a	1	Nenhum	Impõe 0 (zero) no bit b do registrador f.
BSF	f,b,a	1	Nenhum	Impõe 1 (um) no bit b do registrador f.
BTFSC	f,b,a	1 ou 2	Nenhum	Testa o bit b do registrador f, e pula a próxima linha se ele for 0 (zero).
BTFSS	f,b,a	1 ou 2	Nenhum	Testa o bit b do registrador f, e pula a próxima linha se ele for 1 (um).
BTG	f,b,a	1	Nenhum	Inverte o bit b do registrador f.

Controles				
Instrução	**Argumentos**	**Ciclos**	**Status Afetado**	**Descrição**
BC	n	1 ou 2	Nenhum	Desvia se Carry.
BN	n	1 ou 2	Nenhum	Desvia se Negado.
BNC	n	1 ou 2	Nenhum	Desvia se não Carry.
BNN	n	1 ou 2	Nenhum	Desvia se não Negado.



Controles				
Instrução	**Argumentos**	**Ciclos**	**Status Afetado**	**Descrição**
BNOV	n	1 ou 2	Nenhum	Desvia se não Overrun.
BNZ	n	1 ou 2	Nenhum	Desvia se não Zero.
BOV	n	1 ou 2	Nenhum	Desvia se Overrun.
BRA	n	2	Nenhum	Desvio incondicional.
BZ	n	1 ou 2	Nenhum	Desvia se Zero.
CALL	n,s	2	Nenhum	Executa a rotina n. Em s use a opção FAST para usar a SHADOW.
CLRWDT	-	1	Nenhum	Limpa o registrador WDT para não acontecer o reset.
DAW	-	1	C	Efetua o ajuste decimal de W.
GOTO	n	2	Nenhum	Vai para o ponto n, mudando o PC.
NOP	-	-	Nenhum	Gasta um ciclo de máquina sem fazer absolutamente nada.
POP	-	1	Nenhum	Restaura o endereço de retorno gravado em **TOS**.
PUSH	-	1	Nenhum	Grava o endereço de retorno em **TOS**.
RCALL	n	2	Nenhum	Executa a rotina n (relativo).
RETFIE	s	2	Nenhum	Retorna de uma interrupção. Em s use a opção FAST para usar a SHADOW.
RESET	-	1	C, DC, Z, OV, N	Provoca reset no microcontrolador.
RETURN	s	2	Nenhum	Retorna de uma rotina, sem afetar W. Em s use a opção FAST para usar a SHADOW.
SLEEP	-	1	Nenhum	Coloca o PIC em modo sleep (dormindo) para economia de energia.

Operações de leitura e escrita de dados na memória de programa				
Instrução	**Argumentos**	**Ciclos**	**Status Afetado**	**Descrição**
TBLRD*	-	2	Nenhum	Leitura de dado na memória de programa.
TBLRD*+	-	2	Nenhum	Leitura de dado na memória de programa e depois incrementa ponteiro (registrador **TBLPTR**).
TBLRD*-	-	2	Nenhum	Leitura de dado na memória de programa e depois decrementa ponteiro (registrador **TBLPTR**).
TBLRD+*	-	2	Nenhum	Incrementa ponteiro (registrador **TBLPTR**) e depois leitura de dado na memória de programa.
TBLWT*	-	2	Nenhum	Escrita de dado na memória de programa.
TBLWT*+	-	2	Nenhum	Escrita de dado na memória de programa e depois incrementa ponteiro (registrador **TBLPTR**).
TBLWT*-	-	2	Nenhum	Escrita de dado na memória de programa e depois decrementa ponteiro (registrador **TBLPTR**).
TBLWT+*	-	2	Nenhum	Incrementa ponteiro (registrador **TBLPTR**) e depois escrita de dado na memória de programa.

Nota: Para facilitar o seu estudo não foi listado o Set de Instruções estendido.
Mais informações você pode encontrar no data sheet do componente.

4

As Primeiras Explorações (I/Os e Timers)

» Introdução

O objetivo deste capítulo é pegar um pouco mais de intimidade com o PIC18F4520. Quem aprendeu o assembly da Microchip com um PIC menor, provavelmente um de 18 pinos (16F84 ou 16F628, por exemplo), já deve ter percebido que, apesar da diferença de tamanho e quantidade de recursos, o set de instruções de todos esses modelos é exatamente o mesmo. Assim sendo, todo o seu conhecimento adquirido com a programação de outros PICs é de grande valia neste treinamento. Por enquanto, vamos começar com o básico. E nada mais básico do que os I/Os do microcontrolador. Mas convenhamos que, apesar de o 18F4520 possuir 36 I/Os, um capítulo inteiro dedicado a eles é muita coisa. Para aproveitarmos melhor este estudo, vamos aprender também como maximizar esses I/Os, utilizando um mesmo pino ora como entrada, ora como saída.

Vamos aproveitar também a oportunidade para nos aprofundarmos nos quatro Timers existentes nesse modelo.

» Teoria e recursos do PIC

O capítulo é dividido em dois tópicos principais: I/Os e Timers. Cada um desses assuntos merece considerações particulares, tanto na parte teórica quanto na parte prática, já que utilizaremos o mesmo exemplo para demonstrar os dois.

Estudo das PORTAS

A primeira coisa que devemos saber sobre os I/Os de um microcontrolador é que eles são agrupados por PORTs. Esse agrupamento se dá geralmente por características elétricas particulares e para facilitar o gerenciamento da máquina. Por exemplo, como toda a memória de dados interna é de 8 bits, os SFRs também são. Como existem SFRs específicos para controlar os I/Os, é melhor agrupá-los em conjuntos de oito. Desta forma, o 18F4520 possui seus I/Os divididos da seguinte forma:

- ✓ Quatro PORTs de oito I/Os (PORTA, PORTB, PORTC e PORTD);
- ✓ Um PORT de quatro I/Os (PORTE).

Ao total temos então 36 I/Os que podem ser configurados como entrada ou saída pelo programador, conforme as necessidades do projeto.

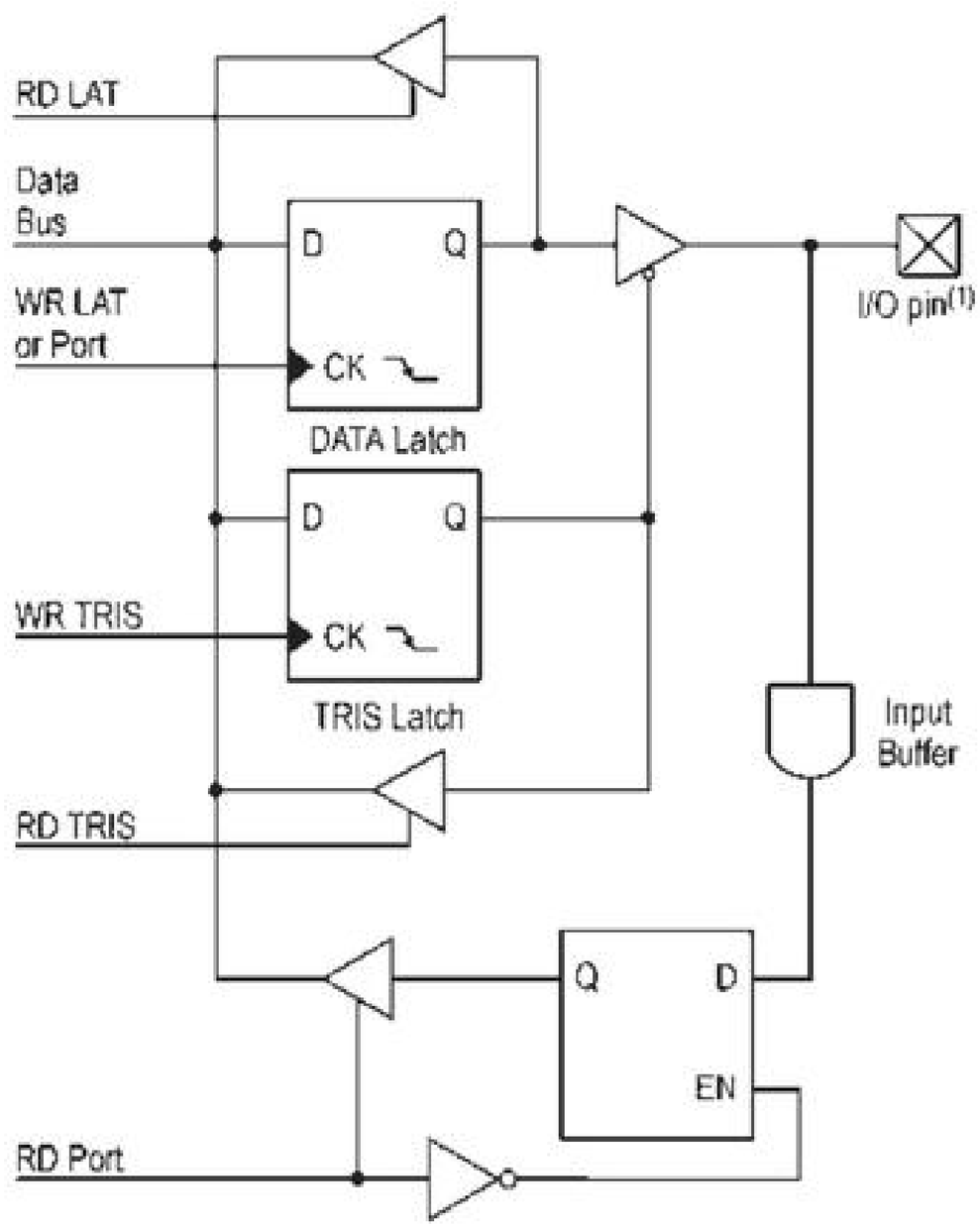
Editora Érica - Desbravando o Microcontrolador PIC18 Recursos Avançados - Daniel R. de Sousa, David José de Souza e Nicolás César Lavinia - 1ª Edição

Outras características importantes que devemos saber sobre essas portas dizem respeito à sua operação elétrica. Internamente esses pinos são ligados de formas diferentes, principalmente pela sobrecarga de recursos aplicada a cada um deles. Vamos conhecer melhor cada uma das portas.

PORTA

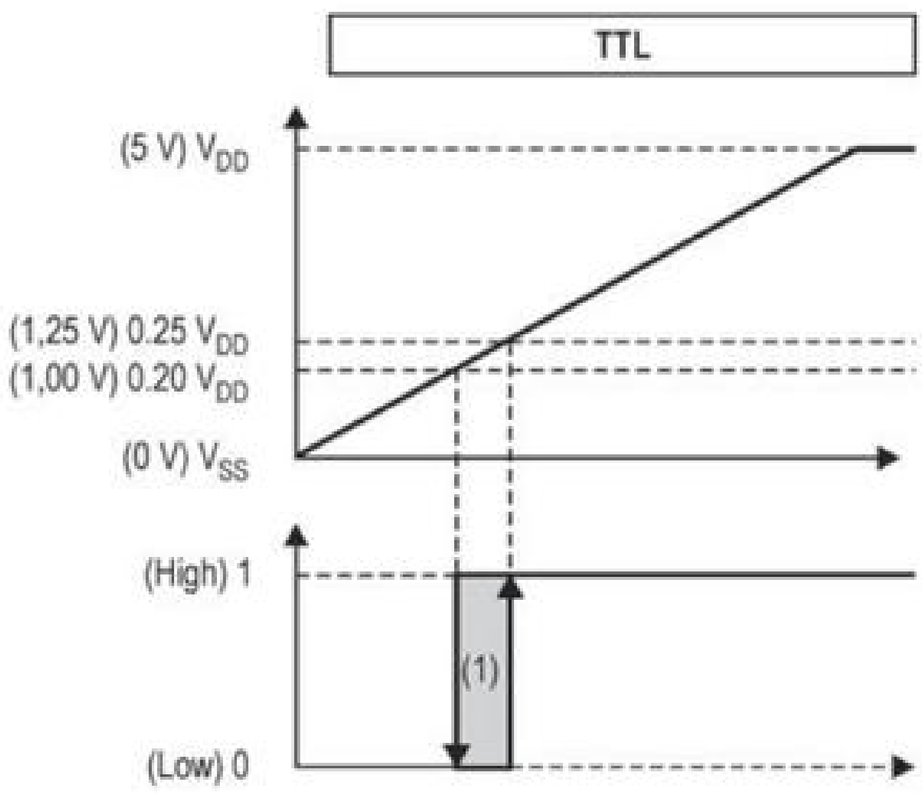
Analisando os diagramas de blocos das ligações internas de um I/O genérico, conhecemos melhor cada uma das características importantes:

Nos diagramas, a WR representa a escrita (Write) e a RD representa a leitura (Read).

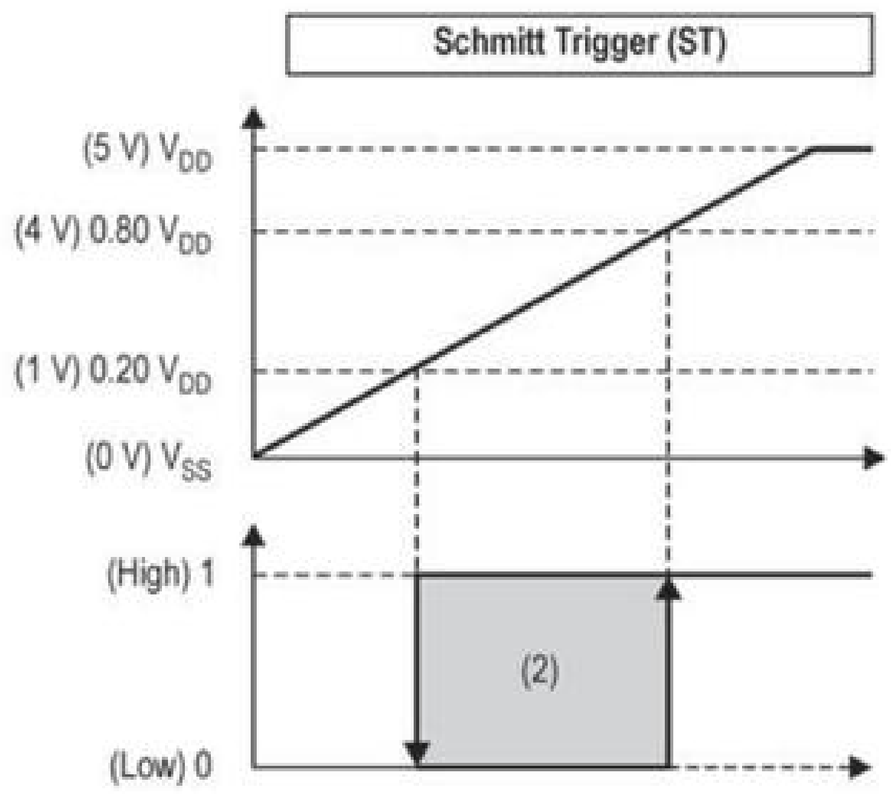


Nota 1: Todos os pinos são protegidos por diodos ligados no V_{DD} e V_{SS}

O primeiro ponto a ser observado é que algumas portas são TTL e outras são Schmitt Trigger. Esse dado é muito relevante quando estamos operando com uma porta como entrada, porque interfere diretamente nos níveis de tensão interpretados pelo PIC como 0 (zero) e 1 (um). Vejamos como isso acontece:



Nota 1: Região incerta. Mantém nível anterior.



Nota 2: Região incerta. Mantém nível anterior.

Repare, então, que nas entradas ST necessitamos de um nível de tensão bem mais elevado para que o PIC interprete a mudança de estado. Isso é muito útil para enquadrarmos uma senoide, por exemplo. Por outro lado, é ruim quando possuímos um hardware que gera níveis de tensão menores, como pode acontecer quando sobrecarregamos o mesmo pino com diversas funções. Essa explicação sobre TTL e ST é aplicável a todas as portas.

Outra característica comum a todos os I/Os é quanto as diferenças existentes entre sua escrita e leitura. Repare que a leitura é feita diretamente sobre o pino, enquanto a escrita passa por um Latch. Isso pode ocasionar um atraso entre o comando de escrita e a efetiva alteração da saída. É por esse motivo que não é recomendável efetuar uma operação de leitura imediatamente após a alteração de uma saída. O certo é aguardarmos pelo menos um ciclo de máquina entre essas duas operações. Esse latch possibilita também que uma operação de escrita seja executada mesmo que o pino esteja configurado como entrada, através do TRIS. O latch é alterado e, quando o estado do pino for modificado para saída, o valor atual do latch lhe será imposto. Lembre-se de que a configuração das entradas e saídas desse port é feita pelo registrador **TRISA**, enquanto as operações de escrita e leitura são executadas pelo registrador **PORTA**.

Um outro cuidado importante é em relação às instruções orientadas a bit. Elas realizam operações de leitura-modificação-escrita (R-M-W - Read-Modify-Write), em que para alterar o estado de um bit internamente é feita a leitura do port, é alterado o estado de um bit e finalmente é escrito o valor no mesmo port. O problema ocorre principalmente quando as vias de saída estão ligadas em um mesmo port e a alteração do port é feita de forma sequencial, bit a bit, sem um interlavo entre uma mudança e outra. Por exemplo, se todos os pinos de um port forem setados de forma sequencial, somente o último bit muda de estado. Esse efeito ocorre principalmente quando temos uma carga capacitiva no pino de saída, ou problema de layout da placa (distâncias das ligações, plano de terra etc.) e a velocidade do microcontrolador.

Para resolver este problema foram criados os registradores **LATx** (sendo x o port correspondente), que atua diretamente na saída do microcontrolador. Com isso temos uma diferenciação conceitual entre registradores **PORTx** e **LATx**:

- ✓ Leitura do registrador **PORTx** corresponde à leitura no port do microcontrolador;
- ✓ Escrita do registrador **PORTx** corresponde à escrita no port do microcontrolador;
- ✓ Leitura do registrador **LATx** corresponde à leitura do último estado no port antes de configurar como entrada pelo registrador **TRISx**;
- ✓ Escrita do registrador **LATx** corresponde à escrita no port do microcontrolador.

Como sugestão, podemos concluir que:

- ✓ **Operações de leitura do port**: usar os registrados PORTx.
- ✓ **Operações de escrita do port**: usar os registradores LATx.

Para os pinos de **RA0** a **RA3** e o **RA5** têm-se uma ligação para o conversor analógico/digital (AD). Um capítulo será completamente dedicado ao sistema de conversão, mas a seleção entre portas digitais ou analógicas é realizada pelo registrador **ADCON1**.

Em contrapartida, o pino **RA4** não possui circuito analógico, mas é ligado internamente ao sistema de incremento do Timer 0 (**TMR0**). Isso possibilita que um clock imposto a esse pino incremente automaticamente o **TMR0**, independentemente do clock real da máquina. O registrador onde se encontra a configuração para incremento do **TMR0** é o **T0CON**.

Os pinos **RA6** e **RA7** são compartilhados com o circuito oscilador externo. Quando configurado para trabalhar com cristal ou ressonador, esses pinos não ficam disponíveis, assim como a configuração para circuito oscilador externo ou RC externo. Para a opção do oscilador interno temos disponíveis os pinos **RA6** e **RA7** para serem utilizados como I/O.

Resumo dos registradores associados ao PORTA

Endereço	Nome	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
F80h	PORTA	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0
F92h	TRISA	Configuração como Entrada (I) ou Saída (0)							
F89h	LATA	LATA7	LATA6	LATA5	LATA4	LATA3	LATA2	LATA1	LATA0
FC1h	ADCON1	-	-	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0
FD5h	T0CON	TMR0ON	T08BIT	T0CS	T0SE	PSA	PS2	PS1	PS0

Não usado para essa finalidade. Para obter mais informações, consulte o Apêndice A.

PORTB

O **PORTB** possui oito I/Os, estando sete deles associados também às interrupções: externa e mudança de estado. Além disso, existem cinco entradas analógicas.

Uma característica elétrica bem interessante para todos os pinos do **PORTB** é a ligação do pull-up interno. Essa ligação para V_{DD} pode ser ligada ou não (em todos os pinos ao mesmo tempo) através da chave **/RBPU** localizada em **INTCON2**. Além da ativação dessa chave, o pino deve ser configurado como entrada para poder ter seu pull-up ativado. A resistência interna para V_{DD} é de aproximadamente 12,5 kΩ, para uma corrente máxima de 400 μA. Na verdade, o pull-up é feito por intermédio de um FET e não de um resistor, e sua corrente típica é de 250 μA, o que aumenta a resistividade.

Todo o **PORTB** como I/O digital é do tipo TTL, exceto os pinos **RB0**, **RB1** e **RB2** quando configurados para utilizarem a interrupção externa, os pinos **RB5**, **RB6** e **RB7** quando em modo de programação serial e **RB4** quando utilizado o módulo CCP. Neste caso, esses pinos tornam-se ST.

Os pinos de **RB4** a **RB7** são interligados à interrupção de mudança de estado e os pinos **RB0**, **RB1** e **RB2** são associados à interrupção externa, desde que configurados como entrada. As chaves e flags dessas interrupções encontram-se em **INTCON**, **INTCON2** e **INTCON3**.

Para os pinos de **RB0** a **RB4** tem-se uma ligação para o conversor analógico/digital (AD). A seleção entre portas digitais ou analógicas é realizada pelo registrador **ADCON1**.

Temos ainda no pino **RB3** a possibilidade de trabalhar com o módulo CCP2. Por meio dos Configurations Bits do microcontrolador podemos selecionar o CCP2 no pino **RB3** ou **RC1**. Essa possibilidade de mudança do pino do módulo CCP2 se deve ao fato de evitar conflitos de hardware entre os pinos entrada do cristal para o Timer 1 e o módulo CCP2. Se trabalharmos com o Timer 1 como contador, não há problemas, mas com um cristal para gerar uma base de tempo diferenciado o módulo CCP2 ficaria inutilizado. Este é um problema dos microcontroladores da família PIC16, como o PIC16F877A.

Resumo dos registradores associados ao PORTB

Endereço	Nome	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
F81h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
F93h	TRISB	Configuração como Entrada (I) ou Saída (0)							
F8Ah	LATB	LATB7	LATB6	LATB5	LATB4	LATB3	LATB2	LATB1	LATB0
FF2h	INTCON	GIE/ GIEH	PEIE/ GIEL	T0IE	INTE	RBIE	TMR0IF	INTF	RBIF
FF1h	INTCON2	RBPU	INTEDG0	INTEDG1	INTEDG2	-	TMR0IP	-	RBIP
FF0h	INTCON3	INT2IP	INT1IP	-	INT2IE	INT1IE	-	INT2IF	INT1IF
FC1h	ADCON1	-	-	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0

Não usado para essa finalidade. Para obter mais informações, consulte o Apêndice A.

PORTC

Nesse PORT, além dos oito I/Os digitais, existe uma série de periféricos ligados aos seus pinos. Cada um desses periféricos será estudado em um capítulo específico.

Todos os pinos desse PORT são do tipo ST quando configurados como entrada.

Resumo dos registradores associados ao PORTC

Endereço	Nome	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
F82h	PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0
F94h	TRISC	Configuração como Entrada (I) ou Saída (0)							
F8Bh	LATC	LATC7	LATC6	LATC5	LATC4	LATC3	LATC2	LATC1	LATC0

Não usado para essa finalidade. Para obter mais informações, consulte o Apêndice A.

PORTD

O **PORTD** é um dos mais simples de todos, pois seus oito pinos são I/Os digitais. Além disso, esses pinos só são ligados ao sistema de porta paralela (Parallel Slave Port) estudado no Capítulo 13.

Todos os pinos do **PORTD** são do tipo ST quando configurados para entrada. Entretanto, para o sistema de porta paralela, esses pinos operam em modo TTL.

A chave de seleção do sistema de porta paralela encontra-se em **TRISE<PSPMODE>**.

Os pinos **RD5** a **RD7** fazem parte do módulo ECCP, que será estudado no Capítulo 9.

Resumo dos registradores associados ao PORTD

Endereço	Nome	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
F83h	PORTD	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0
F95h	TRISD	Configuração como Entrada (I) ou Saída (0)							
F96h	TRISE	IBF	OBF	IBOV	PSPMODE	-	Configuração I/Os		
F8Ch	LATD	LATD7	LATD6	LATD5	LATD4	LATD3	LATD2	LATD1	LATD0
FBDh	CCP1CON	P1M1	P1M0	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0

Não usado para essa finalidade. Para obter mais informações, consulte o Apêndice A.

PORTE

Esse PORT é o menor de todos, possuindo somente quatro pinos.

Observe que todos os pinos do **PORTE** são do tipo ST quando configurados para entrada. Entretanto, para o sistema de porta paralela (**/RD**, **/RW** e **/CS**), os pinos **RE0** a **RE2** operam em modo TTL.

A chave de seleção do sistema de porta paralela encontra-se em **TRISE<PSPMODE>**.

Note que o pino **MCLR** pode ser desabilitado nos Configurations Bits, dando lugar ao **RE3**, que é somente entrada.

Os pinos **RE0** a **RE2** também operam como entradas analógicas, ligadas ao conversor interno.

Resumo dos registradores associados ao PORTE

Endereço	Nome	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
F84h	PORTE	-	-	-	-	RE3	RE2	RE1	RE0
F96h	TRISE	IBF	OBF	IBOV	PSPMODE	-	Configuração dos I/Os		
F8Dh	LATE	LATE7	LATE6	LATE5	LATE4	LATE3	LATE2	LATE1	LATE0
FC2h	ADCON1	-	-	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0

Não usado para essa finalidade. Para obter mais informações, consulte o Apêndice A.

Estudo dos timers

Vamos aprender agora os recursos dos quatro timers disponíveis no PIC18F4520 e como utilizá-los.

Cada um desses contadores internos possui características próprias, como o limite de contagem, o tipo de incremento, os pre e postscalers, a geração de interrupções, periféricos associados etc. Vamos, então, explorá-los individualmente.

Timer0

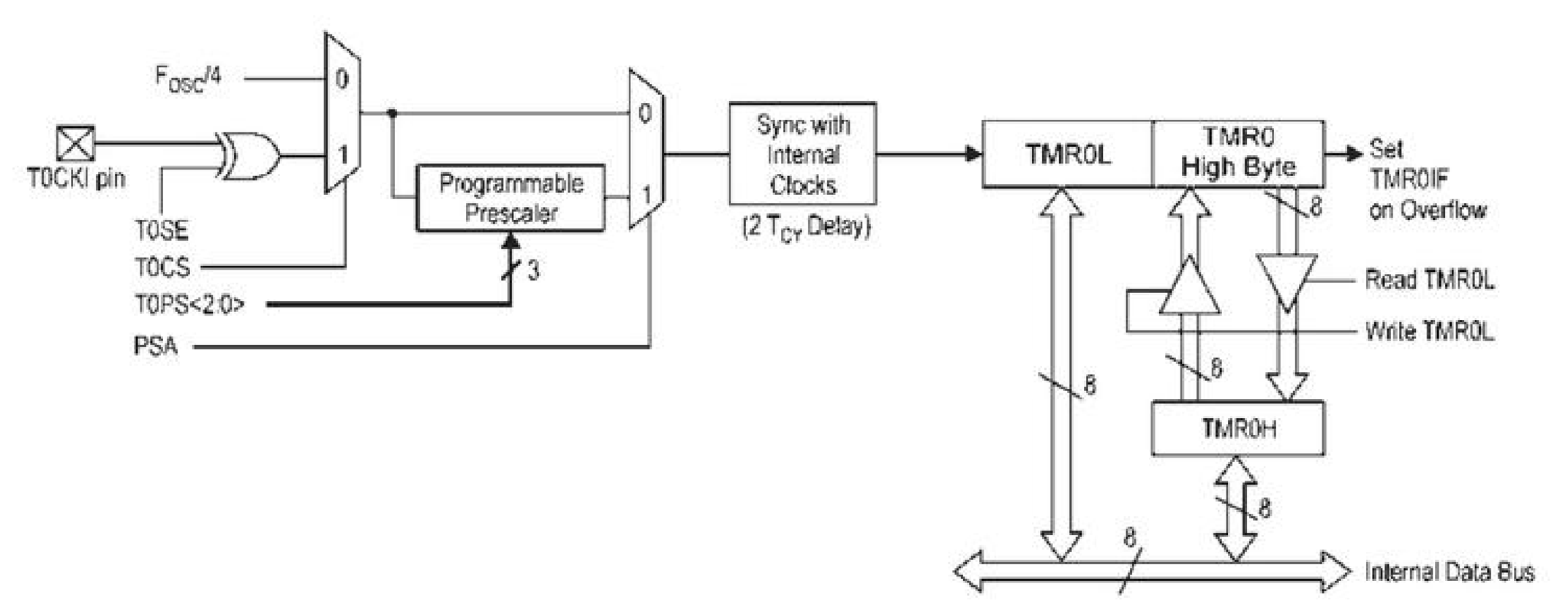
É um contador de 16 bits, composto de dois registradores de 8 bits (**TMR0L** e **TMR0H**). Esses registradores podem ser tanto lidos quanto escritos, possibilitando a inicialização do contador. Outra característica interessante do Timer0 é que apesar de ele ser somente incremental, seus incrementos podem ocorrer de duas maneiras distintas, através da chave **T0CON<T0CS>**:

- ✓ **T0CS = 1**: incremento a cada transição no pino **RA4 / T0CKI**;
- ✓ **T0CS = 0**: incremento a cada ciclo de máquina.

Quando selecionado o incremento pelo pino **T0CKI**, isto é, um pulso ou clock externo, é possível ainda selecionar se o incremento acontecerá na borda de subida ou na borda de descida do sinal de entrada, através de **T0CON<T0SE>**:

- ✓ **T0SE = 1**: incremento na borda de descida;
- ✓ **T0SE = 0**: incremento na borda de subida.

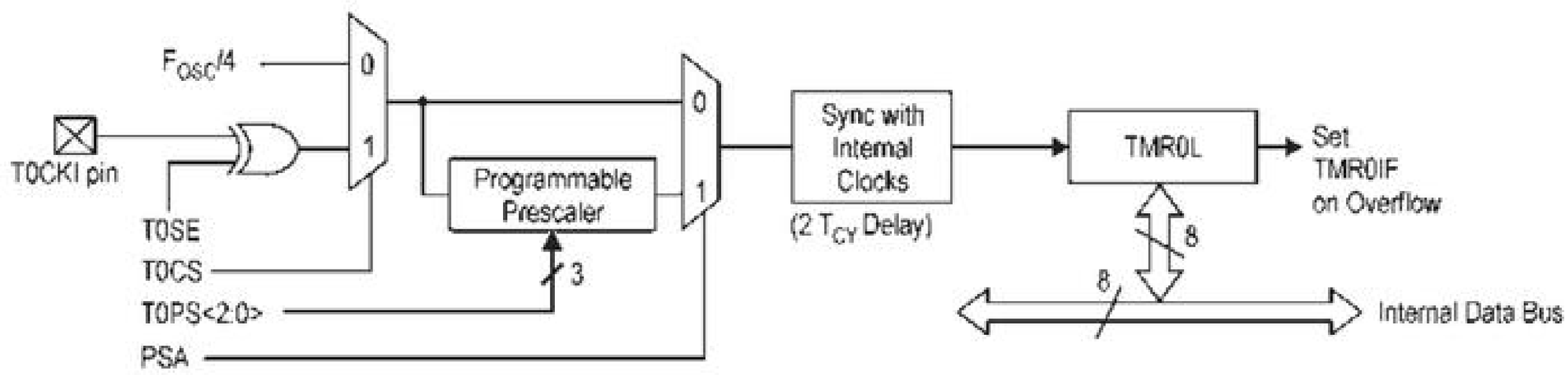
Entretanto, para incremento externo ou interno, é aplicado um prescaler (PS) antes de o contador do Timer0 ser realmente alterado. Desta forma, caso o prescaler esteja configurado em 1:4, por exemplo, são necessários quatro ciclos de máquina (ou pulsos externos) para que o contador do Timer0 seja incrementado. O contador interno do PS é de 8 bits, mas não está disponível nem para leitura nem para escrita. Vale lembrar que toda vez que algum dado é escrito no contador do Timer0, o contador de PS é zerado.



Note na figura anterior que existe uma ordem de acesso aos registradores **TMR0L** e **TMR0H** para operações de escrita e leitura. Para operação de escrita devemos escrever primeiro no **TMR0H** e depois no **TMR0L**. Para a operação de leitura devemos ler primeiro no **TMR0L** e depois no **TMR0H**. Veja que essa melhoria no acesso a um contador de 16 bits elimina alguns problemas de leitura e escrita de dados no contador, como ocorre nos microcontroladores PIC16.

O Timer0 pode trabalhar com contador no modo 8 bits por intermédio do bit **T0CON<T0BIT>**:

- ✓ **T0BIT = 1**: contador modo 8 bits;
- ✓ **T0BIT = 0**: contador modo 16 bits.



Para a utilização do prescaler é necessário habilitar o divisor. Isso é possível por intermédio do **T0CON<PSA>**:

- ✓ **PSA = 1**: prescaler ligado;
- ✓ **PSA = 0**: prescaler desligado.

Depois disso, é necessário ainda configurar o valor do PS, conforme ajustes em **T0CON<PS2:PS0>**:

PS2	PS1	PS0	Contador TMR0
0	0	0	1:2
0	0	1	1:4
0	1	0	1:8
0	1	1	1:16
1	0	0	1:32
1	0	1	1:64
1	1	0	1:128
1	1	1	1:256

Desta forma, a única maneira de incrementarmos o **TMR0** com uma relação de 1:1 é limpar o bit **T0CON<PSA>**.

Para aumentar o poder de operação do Timer0, associado a ele existe uma interrupção acontecendo toda vez que o contador do Timer0 estourar, isto é, sempre que o limite do contador for ultrapassado (255 ou FFh no modo 8 bits, 65535 ou FFFFh no modo 16 bits). Quando isso acontecer, o flag **INTCON<TMR0IF>** será setado. Para utilizar essa interrupção, a chave **INTCON<TMR0IE>** deve ser ligada, assim como as chaves **INTCON<GIEH>** e **INTCON<GIEL>**. Não se esquecer de limpar o flag (**TMR0IF**) manualmente durante o tratamento da interrupção.

Resumo dos registradores associados ao Timer0

Endereço	Nome	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
FD6h	TMR0L	Contador - byte inferior							
FD7h	TMR0H	Contador - byte superior							
FF2h	INTCON	GIE/ GIEH	PEIE/ GIEL	T0IE	INTE	RBIE	TMR0IF	INTF	RBIF
FD5h	T0CON	TMR0ON	T08BIT	T0CS	T0SE	PSA	PS2	PS1	PS0
F92h	TRISA	TRISA7	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0

Não usado para essa finalidade. Para obter mais informações, consulte o Apêndice A.

Timer1

Assim como o Timer0, o Timer1 é um timer de 16 bits, composto de dois registradores de 8 bits (**TMR1H** e **TMR1L**) que podem ser escritos e lidos pelo programador. Além disso, como no Timer0, também pode operar com um sinal externo, mas com uma vantagem: existe um circuito interno que possibilita ligar diretamente um cristal aos pinos **RC0/T1OSO** e **RC1/T1OSI**. Outra forma de operar com sinal externo é ligando um sinal pulsado qualquer ao pino **RC0/T1CKI**, de forma similar ao Timer0.

Para escolher entre incremento interno ou externo, a chave **T1CON<TMR1CS>** deve ser configurada:

- ✓ **TMR1CS = 1**: incremento externo, através de sinal **RC0/T0CKI** ou oscilador em **RC0/T1OSO** e **RC1/T1OSI**;
- ✓ **TMR1CS = 0**: incremento interno através dos ciclos de máquina.

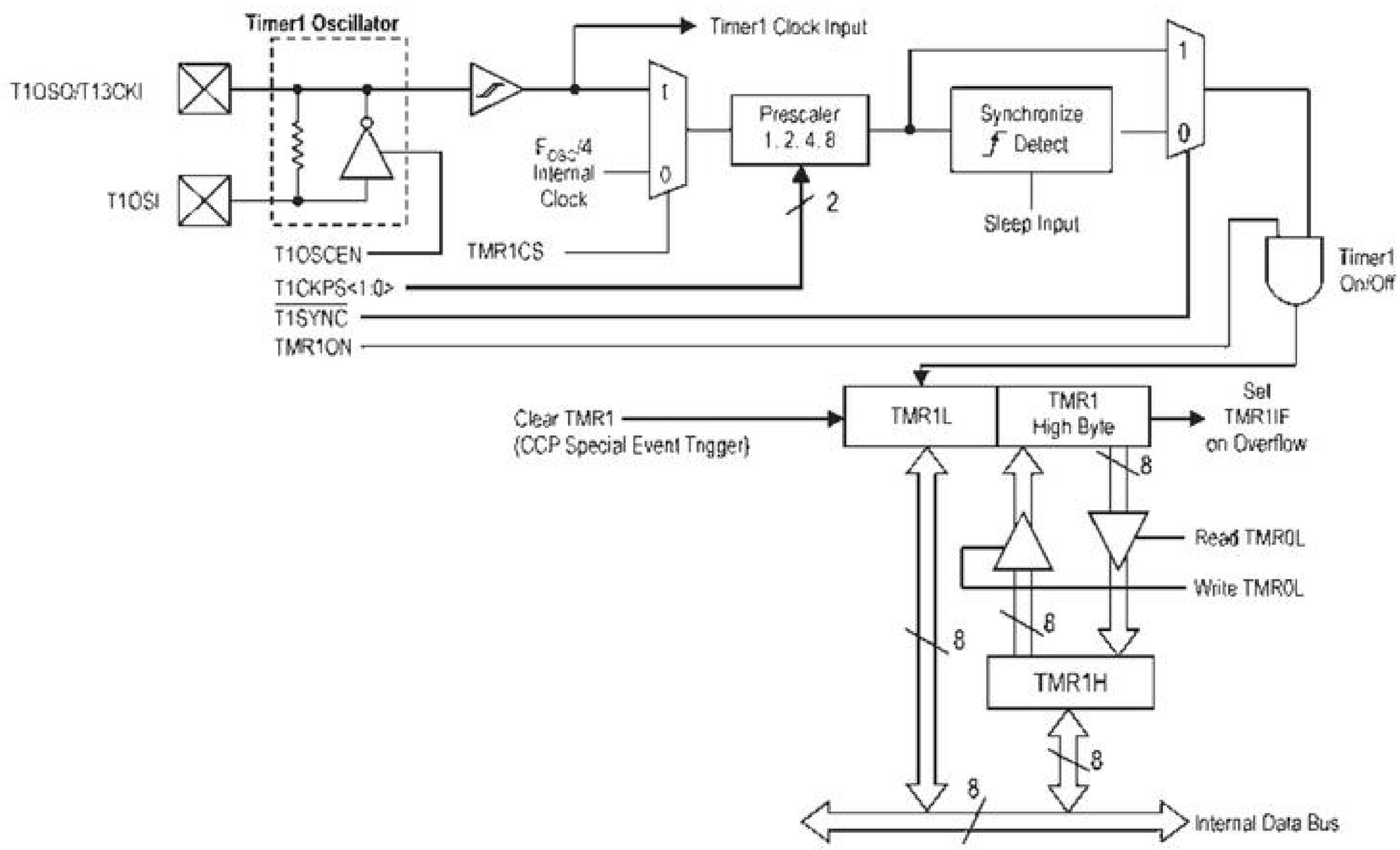
A chave **T1CON<T1OSCEN>** é a responsável pela habilitação ou não do circuito de oscilação interna:

- ✓ **T1OSCEN = 1**: oscilador ligado. **RC0** e **RC1** são configurados como entrada, independentemente do estado do **TRISC**. Esses pinos são perdidos como I/Os;
- ✓ **T1OSCEN = 0**: circuito oscilador desligado. **RC0** opera como **T13CKI** (sinal externo para contagem) e **RC1** opera como I/O normal.

Esse circuito de oscilação é idêntico ao utilizado pelo próprio cristal do PIC quando na opção LP. Desta forma, ele é de baixo consumo, mas também tem sua frequência de trabalho limitada em 32,768 kHz. Note que esse oscilador pode ser utilizado como oscilador secundário, e podemos utilizar para baixar momentaneamente o consumo do microcontrolador ou no caso de falha do oscilador principal. O bit **T1C0N<T1RUN>** monitora se o Timer1 está recebendo clock de outra fonte (oscilador principal, oscilador interno ou pulsos externos) ou do oscilador dedicado do módulo:

- ✓ **T1RUN = 1**: clock do Timer1 originado do oscilador dedicado;
- ✓ **T1RUN = 0**: clock do Timer1 originado de outra fonte.

Para melhorar o sinal externo utilizado para incrementar o contador do Timer1, ele passa por uma porta Schmitt Trigger. Isso pode ser observado no diagrama de blocos desse timer.



Depois de selecionada a origem do sinal, o sistema possui um prescaler próprio para efetivar o incremento. Esse prescaler deve ser configurado através de **T1CON<T1CKPS1 :T1CKPS0>**:

T1CKPS1	T1CKPS0	Prescale
0	0	1:1
0	1	1:2
1	0	1:4
1	1	1:8

De forma análoga ao Timer0, esse timer também possui um sistema de sincronismo do clock externo com o clock interno. Entretanto, ao contrário do Timer0, esse sincronismo pode ser desligado através da chave **T1CON</T1SYNC>**:

- ✓ **/T1SYNC = 1**: sincronismo desligado;
- ✓ **/T1SYNC = 0**: sincronismo ligado.

Esse recurso é muito interessante, pois se mantivermos o sincronismo desligado, podemos continuar operando com o Timer1 mesmo com o PIC em modo SLEEP, desde que seu incremento seja externo. Quando o sincronismo está ligado, como no caso do Timer0, isso não é possível, pois o sistema de sincronismo para de operar no SLEEP, uma vez que o clock interno é desligado.

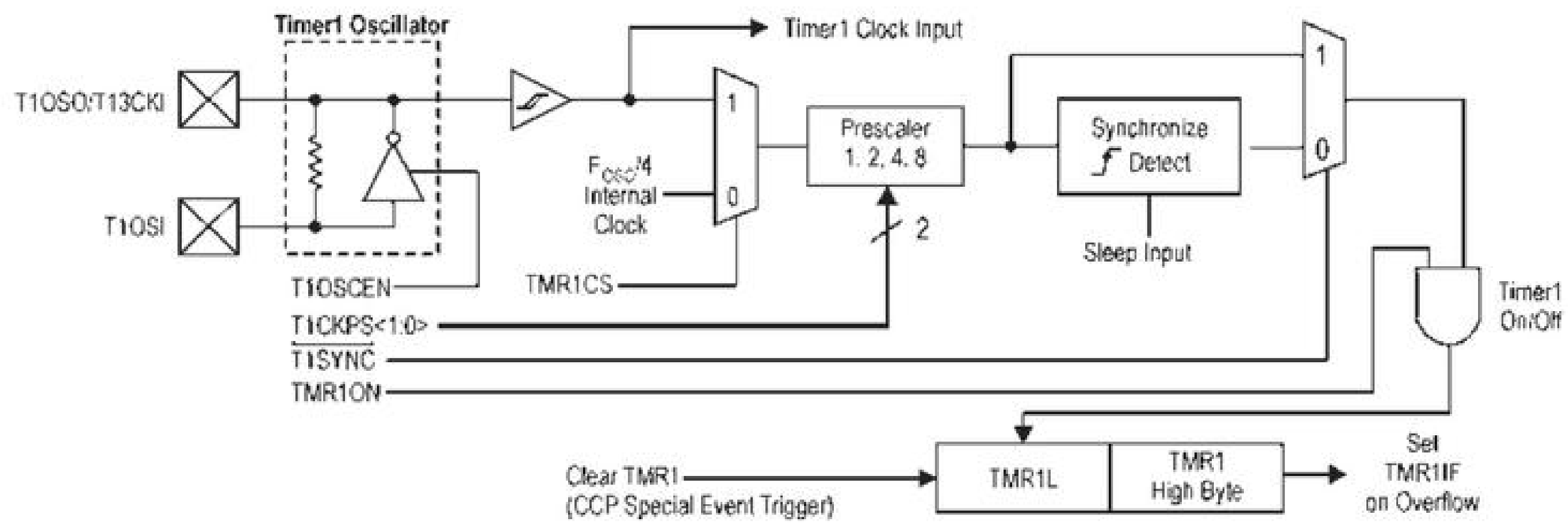
Quando estamos trabalhando com um sinal externo, o incremento do **TMR1** é efetuado na borda de subida do sinal. Entretanto, após o timer ser ligado, é necessária uma borda de descida antes da primeira borda de subida que será considerada.

Esse timer possui ainda uma chave de habilitação que pode ser utilizada para ligar e desligar os incrementos de **TMR1**. Essa chave localiza-se em **T1CON<TMR1ON>**:

- ✓ **TMR1ON = 1**: incremento de TMR1 habilitado;
- ✓ **TMR1ON = 0**: incremento de TMR1 desabilitado.

No Timer1, assim como no Timer0, existe uma melhoria no hardware de acesso do contador na leitura e na escrita, como visto na figura anterior. Todavia, por questão de compatibilidade de software da família PIC16, o bit **T1CON<RD16>** muda a característica do hardware do contador:

- ✓ **RD16 = 1**: contador modo 16 bits de leitura e escrita;
- ✓ **RD16 = 0**: contador modo 8 bits de leitura e escrita.



No caso de leitura e escrita no modo de 8 bits, a maneira mais convencional seria desligarmos o timer antes de executarmos essas operações, mas nem sempre isso é possível. Outra forma é lermos os valores (primeiro **TMR1H** e depois **TMR1L**) guardando-os em variáveis temporárias. Depois da leitura, comparamos o valor de **TMR1H**. Caso ele tenha aumentado, lemos ambos os valores novamente. Para a escrita, devemos primeiro limpar **TRM1L**, depois carregamos **TMR1H** e **TMR1L**. Assim como no TMR0, a escrita nesses registradores reseta o prescaler. A figura anterior mostrou o hardware do Timer1 com o contador modo de 8 bits de leitura e escrita.

Independente do modo de trabalho do contador, ele pode ser reiniciado por um evento do módulo CCP. Em um capítulo dedicado a esse módulo vamos detalhar essa possibilidade.

Associado ao estouro do Timer1 também temos uma interrupção, que deve ser habilitada por **PIE1<TMR1IE>**, além de **INTCON<GIE/GIEH>** e **INTCON<PEIE/GIEL>**. O flag de indicação para essa interrupção é o **PIR1<TMR1IF>**.

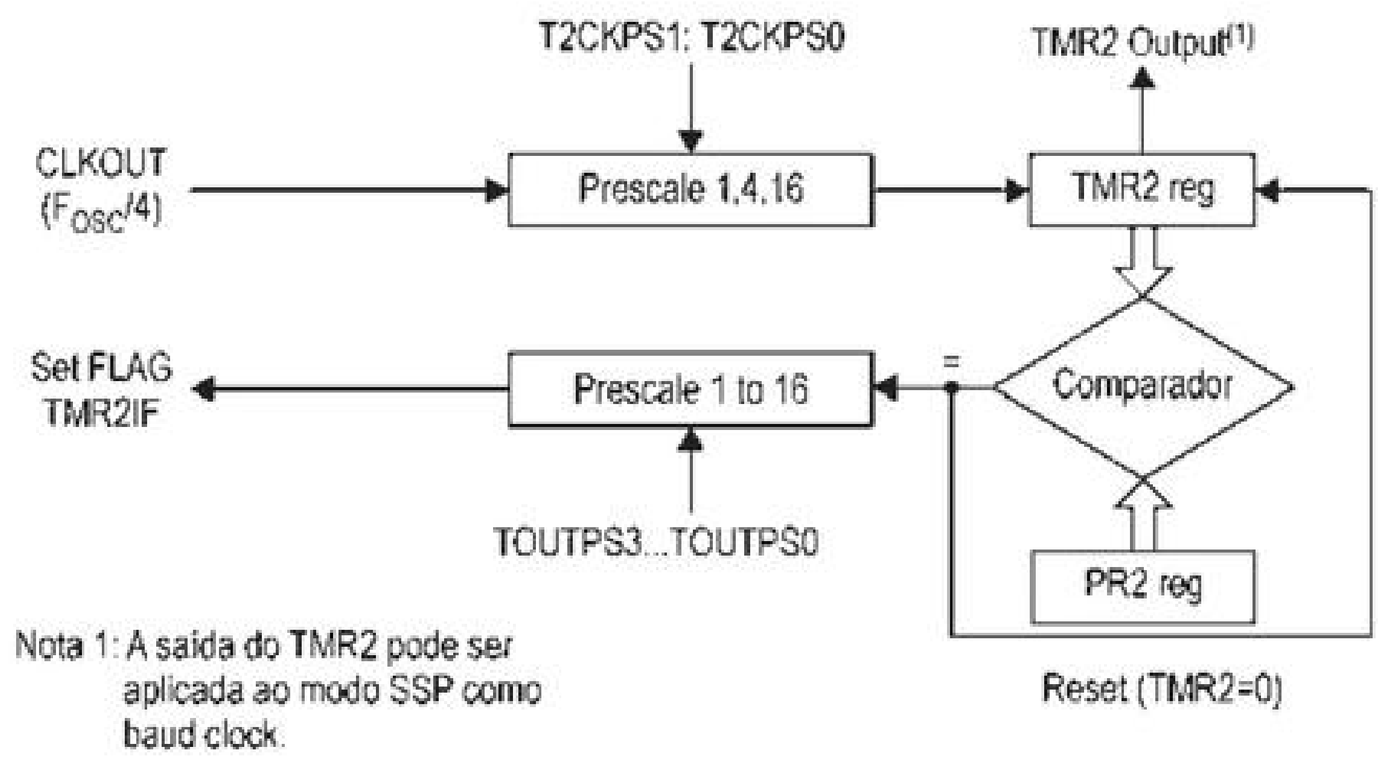
Resumo dos registradores associados ao Timer1

Endereço	Nome	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
FF2h	INTCON	GIE/ GIEH	PEIE/ GIEL	T0IE	INTE	RBIE	TMR0IF	INTF	RBIF
F9Eh	PIR1	PSPIF	ADIF	RCIF	RXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
F9Dh	PIE1	PSPIE	ADIE	RCIE	RXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
F9Fh	IPR1	PSPIP	ADIP	RCIP	RXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP
FCEh	TMR1L	Contador - 8 bits (parte menos significativa)							
FCFh	TMR1H	Contador - 8 bits (parte mais significativa)							
FCDh	T1CON	RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	/T1SYNC	TMR1CS	TMR1ON

Não usado para essa finalidade. Para obter mais informações, consulte o Apêndice A.

Timer2

O Timer2 é um contador de 8 bits, relacionado somente ao clock interno, que possui um prescaler e um postscale, sendo ambos configuráveis.



Uma característica que o difere bastante dos demais Timers é que ele não conta de 0 (zero) até o limite imposto pelos 8 bits. Na verdade, quem impõe o limite de incremento é o valor escrito no registrador **PR2**. Desta forma, sempre que **TMR2 = PR2**, o timer é resetado, voltando a zero. Nesse mesmo momento, o contador de postscale é incrementado. Quando o postscale terminar, a interrupção associada ao Timer2 será gerada.

O ajuste do prescaler deve ser efetuado através de **T2CON<T2CKPS1:T2CKPS0>**:

T2CKPS1	T2CKPS0	Prescale
0	0	1:1
0	1	1:4
1	0	1:16
1	1	1:16

Apesar de parecer um erro, as duas últimas opções são realmente idênticas.

Quanto ao postscale, sua função está diretamente ligada à interrupção de Timer2. Isso porque ele nada mais é que um contador de estouros do **TMR2** (não se esqueça de que esses estouros são em comparação com **PR2**, e não no limite de 8 bits). Somente quando a quantidade de estouros for igual ao valor configurado no postscale é que a interrupção vai acontecer.

Uma característica interessante para o postscale é que ele pode ser ajustado entre 1 e 16, com incrementos unitários. Em relação aos SFRs, sua configuração deve ser feita em **T2CON<TOUTPS3:TOUTPS0>**:

TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	Postscale
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

TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	Postscale
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

Observe que tanto o prescaler quanto o postscale são zerados sempre que uma das seguintes operações acontecer:

- ✓ Uma operação de escrita envolvendo o registrador **TMR2**;
- ✓ Uma operação de escrita envolvendo o registrador **T2CON**;
- ✓ Qualquer tipo de reset do PIC.

Entretanto, o TMR2 é zerado somente durante algum tipo de reset, diferentemente dos demais timers que, nos resets, ou mantêm seu valor ou possuem um valor desconhecido.

Apesar de o diagrama simplificado de blocos não mostrar, o Timer2 possui ainda uma chave de habilitação idêntica à existente no Timer1. Essa chave encontra-se em **T2CON<TMR2ON>**:

- ✓ **TMR2ON = 1**: incremento de TMR2 habilitado;
- ✓ **TMR2ON = 0**: incremento de TMR2 desabilitado.

Resumo dos registradores associados ao Timer2

Endereço	Nome	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
FF2h	INTCON	GIE/ GIEH	PEIE/ GIEL	T0IE	INTE	RBIE	TMR0IF	INTF	RBIF
F9Eh	PIR1	PSPIF	ADIF	RCIF	RXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
F9Dh	PIE1	PSPIE	ADIE	RCIE	RXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
F9Fh	IPR1	PSPIP	ADIP	RCIP	RXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP
FCCh	TMR2	Contador - 8 bits							
FCAh	T2CON	-	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
FCBh	PR2	Limite de incremento para TMR2							

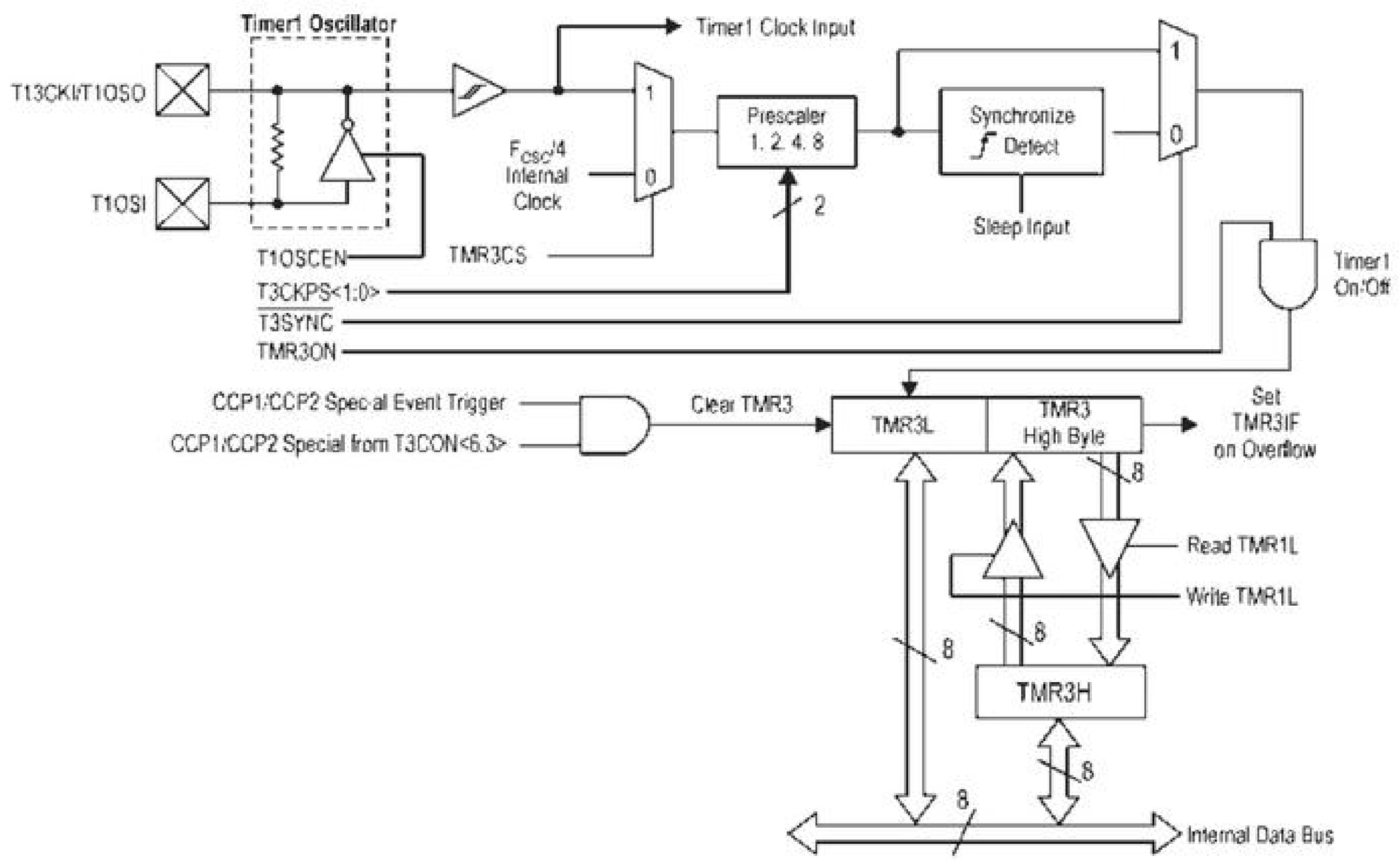
Não usado para essa finalidade. Para obter mais informações, consulte o Apêndice A.

Timer3

Podemos dizer que o Timer3 é uma cópia do Timer1, onde é timer de 16 bits, composto de dois registradores de 8 bits (**TMR3H** e **TMR3L**) que podem ser escritos e lidos pelo programador. Pode trabalhar com um cristal independente ligado em **RC0/T1OSO** e **RC1/T1OSI** ou com sinal externo ligando um sinal pulsado qualquer ao pino **RC0/T1CKI**, de forma similar ao Timer0. Note que o circuito oscilador dedicado e o pino que recebe os pulsos externos são compartilhados ao Timer1, e toda a configuração foi vista anteriormente.

Para escolher entre incremento interno ou externo, a chave **T3CON<TMR3CS>** deve ser configurada:

- ✓ **TMR3CS = 1**: incremento externo, através de sinal **RC0/T0CKI** ou oscilador em **RC0/T1OSO** e **RC1/T1OSI**;
- ✓ **TMR3CS = 0**: incremento interno através dos ciclos de máquina.



Depois de selecionada a origem do sinal, o sistema possui um prescaler próprio para efetivar o incremento. Esse prescaler deve ser configurado através de **T3CON<T3CKPS1 :T3CKPS0>**:

T3CKPS1	T3CKPS0	Prescale
0	0	1:1
0	1	1:2
1	0	1:4
1	1	1:8

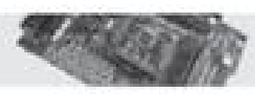
De forma análoga ao Timer0, esse timer também possui um sistema de sincronismo do clock externo com o clock interno. Entretanto, ao contrário do Timer0, esse sincronismo pode ser desligado pela chave **T3CON</T3SYNC>**:

- ✓ **/T3SYNC = 1**: sincronismo desligado;
- ✓ **/T3SYNC = 0**: sincronismo ligado.

Esse timer possui ainda uma chave de habilitação que pode ser utilizada para ligar e desligar os incrementos de **TMR3**. Ela se localiza em **T3CON<TMR3ON>**:

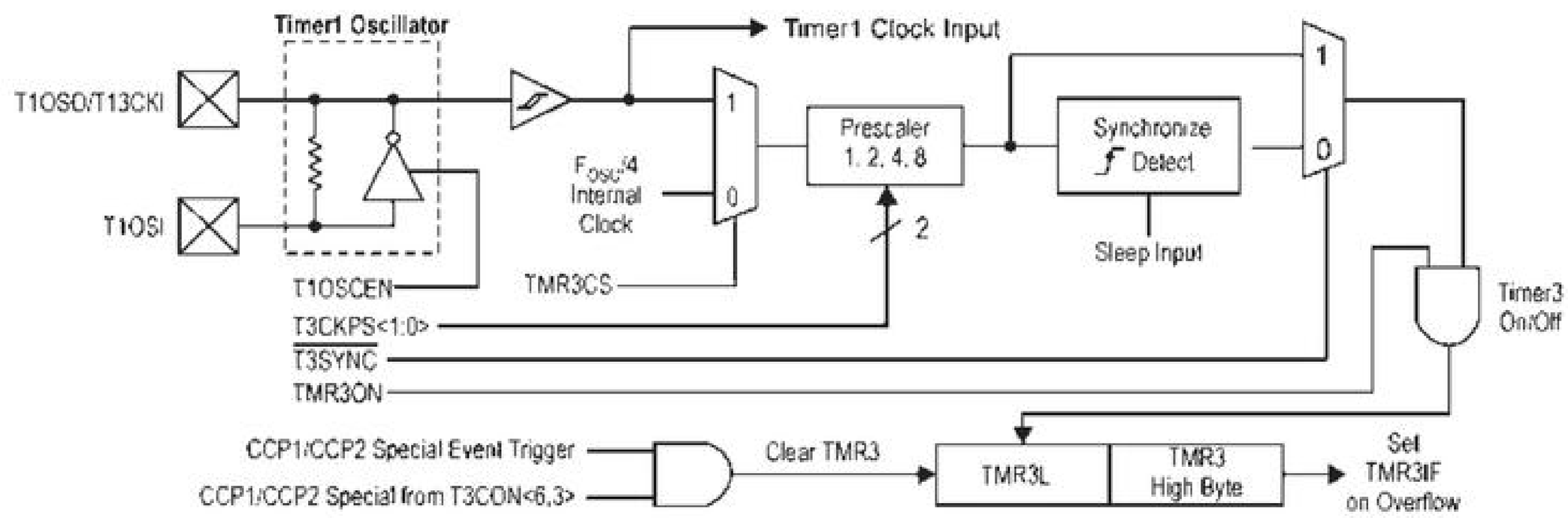
- ✓ **TMR3ON = 1**: incremento de TMR3 habilitado;
- ✓ **TMR3ON = 0**: incremento de TMR3 desabilitado.

Assim como no Timer1, no Timer3 existe a configuração do contador modo 8 bits ou 16 bits, em que o bit **T3CON<RD16>** muda a característica do hardware do contador:



- ✓ **RD16 = 1**: contador modo 16 bits de leitura e escrita;
- ✓ **RD16 = 0**: contador modo 8 bits de leitura e escrita.

A figura seguinte mostra o hardware do Timer3 com o contador modo 8 bits de leitura e escrita.



Independente do modo do de trabalho do contador, ele pode ser reiniciado por um evento do módulo CCP. Em um capítulo dedicado a esse módulo vamos detalhar essa possibilidade.

Associada ao estouro do Timer3 também temos uma interrupção, que deve ser habilitada por **PIE2<TMR3IE>**, além de **INTCON<GIE/GIEH>** e **INTCON<PEIE/GIEL>**. O flag de indicação para essa interrupção é o **PIR2<TMR3IF>**.

Resumo dos registradores associados ao Timer3

Endereço	Nome	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
FF2h	INTCON	GIE/ GIEH	PEIE/ GIEL	T0IE	INTE	RBIE	TMR0IF	INTF	RBIF
FA1h	PIR2	OSCFIF	CMIF	-	EEIF	BCLIF	HLVDIF	TMR3IF	CCP2IF
FA0h	PIE2	OSCFIE	CMIE	-	EEIE	BCLIE	HLVDIE	TMR3IE	CCP2IE
FA2h	IPR2	OSCFIP	CMIP	-	EEIP	BCLIP	HLVDIP	TMR3IP	CCP2IP
FB2h	TMR3L	Contador - 8 bits (parte menos significativa)							
FB3h	TMR3H	Contador - 8 bits (parte mais significativa)							
FCDh	T1CON	RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	/T1SYNC	TMR1CS	TMR1ON
FB1h	T3CON	RD16	T3CCP2	T1CKPS1	T1CKPS0	T3CCP1	/T3SYNC	TMR3CS	TMR3ON

Não usado para essa finalidade. Para obter mais informações, consulte o Apêndice A.

Maximização de pino

Este assunto é muito solicitado e necessário nos projetos atuais. Isso porque é de grande valia economizarmos recursos de hardware, seja para utilizar um PIC menor, seja para viabilizar o projeto ou ainda para obter I/Os de sobra para futuras expansões. O importante é lembrarmos que o investimento em um projeto mais aprimorado é feito uma única vez, enquanto o custo de um PIC maior é inerente a toda a vida do produto.

A maximização mais comum que podemos efetuar é o reaproveitamento de um mesmo I/O para mais de uma função, muitas vezes extremamente diferenciadas. Mas como isso é possível? Uma maneira de realizarmos isso é, por exemplo, pelo chaveamento de um MUX externo, fazer com que hardwares diferentes possam ser interligados no mesmo pino do PIC, um de cada vez.

Outra forma de maximização é a varredura, em que trabalhamos com matrizes do tipo linhas versus colunas. Essa técnica é comumente utilizada para teclados com grande número de teclas, LEDs ou displays de segmentos. O próximo capítulo dedica-se ao sistema de varredura de displays.

Por fim, outra técnica muito utilizada é a ligação de dois hardwares diferentes no mesmo pino, ao mesmo tempo, sem a separação por MUX. Mas como isso é possível? Obviamente nem sempre isso é real, mas existem diversos casos em que podemos ligar um hardware equivalente a um sinal de entrada a outro hardware responsável pelo controle de uma saída.

Está parecendo um pouco confuso, não é mesmo? Mas, na verdade, é muito simples. Vamos pegar o modelo mais utilizado para essa aplicação, com o qual vamos trabalhar em nosso exemplo. Trata-se da ligação de um botão e um LED ao mesmo pino do PIC.

Para que isso funcione corretamente, projetamos um hardware capaz de operar com o botão normalmente, informando ao PIC dois níveis de tensão diferenciados (alto e baixo). Conjuntamente, acrescentamos ao esquema um LED que será controlado quando o PIC impor o nível de tensão (alto e baixo). Para que isso funcione, o segredo está na operação conjunta entre esse hardware e o controle efetuado pelo software, pois o pino que está sendo utilizado deve ficar constantemente chaveado como saída (para controlar o LED) e entrada, para ler o valor do botão.

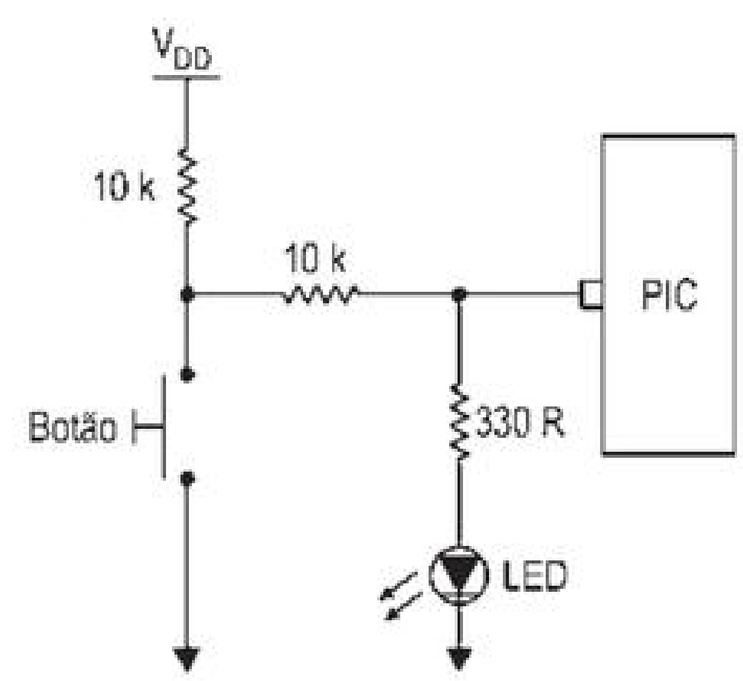
Vamos, então, nos aprofundar um pouco mais nesta teoria. Comecemos pelo esquema elétrico do hardware necessário à aplicação.

Do lado esquerdo temos a ligação padrão para um botão, isto é, um pull-up para garantir o nível alto quando o botão está liberado (aberto) e o botão propriamente dito, ligado ao V_{SS}, que garante o nível baixo quando pressionado (fechado).

Depois temos a ligação de um LED para terra com um resistor para o PIC, limitando a corrente máxima permitida.

Entre o circuito do botão e do LED existe um terceiro resistor para evitarmos um curto durante o momento em que queremos acender o LED e pressionamos o botão.

Vamos agora entender a lógica do sistema.



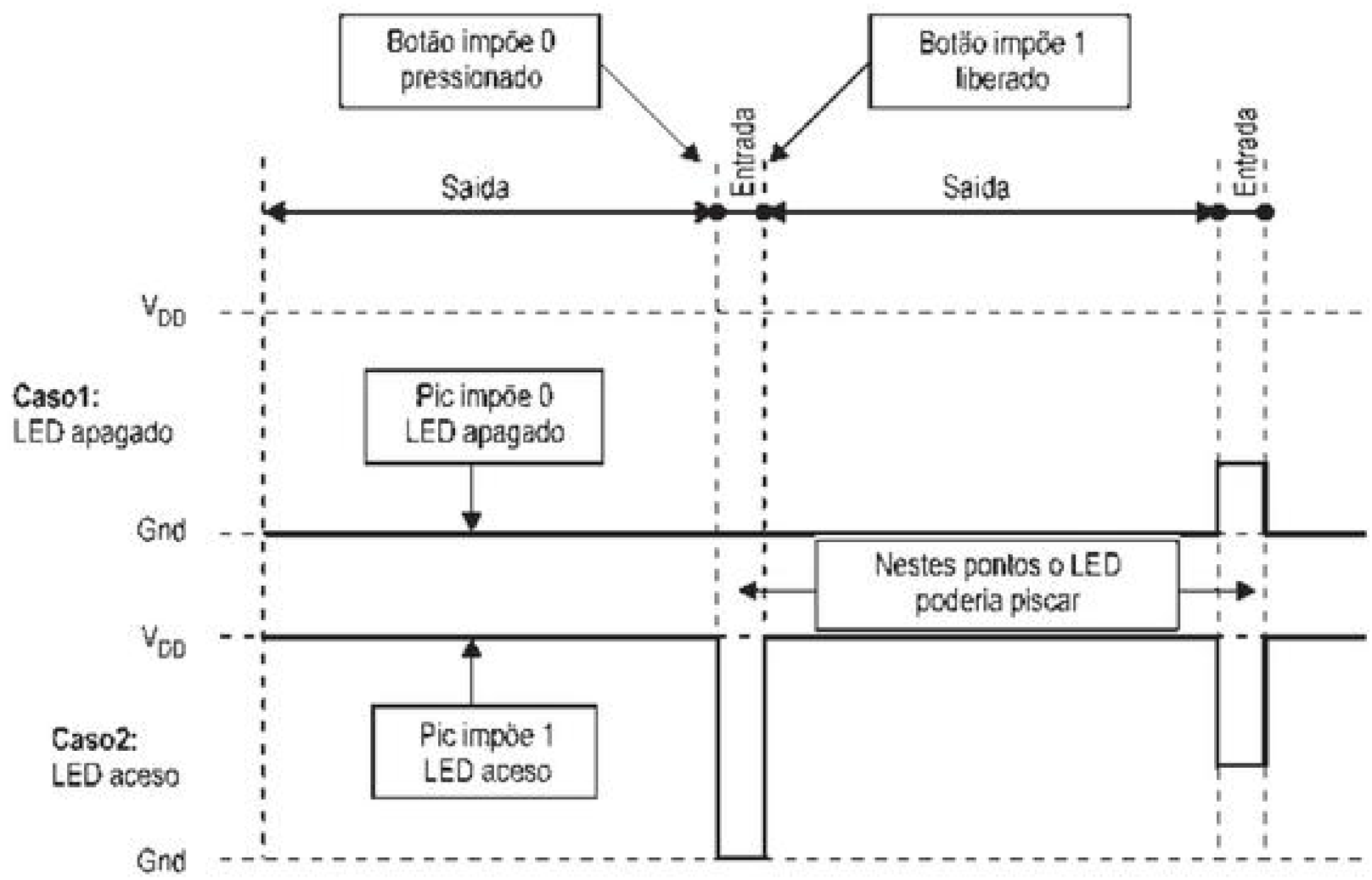
Na maior parte do tempo, o pino do PIC é configurado como saída, já que, se ele estiver impondo 0 (V_{SS}), o LED permanece apagado. Se ele impor 1 (V_{DD}), o LED será setado. Devido ao resistor de proteção, o pressionamento ou não do botão não interfere no funcionamento do LED.

Até aí tudo bem, mas como vamos manter o LED controlado e ler o botão ao mesmo tempo? A questão é muito mais simples do que parece. Vamos fazer uso da técnica de configurar o pino como entrada para lermos o estado do botão. Acontece que faremos isso tão rápido que será imperceptível para o LED. Assim sendo, na maior parte do tempo o pino permanece como saída, e periodicamente ele é convertido em entrada, o botão é lido e em seguida, vira saída novamente.

A figura seguinte apresenta claramente o sinal em relação ao pino do PIC. Repare que a relação entre o tempo de entrada e saída é muito mais propícia ao controle do LED (saída). Como isso é feito em uma frequência relativamente alta para o olho humano (no mínimo 100 Hz), a piscada do LED durante o estado de entrada é imperceptível.

Desta forma, o botão não é lido o tempo inteiro, mas somente uma vez dentro de cada período da frequência de chaveamento. Por isso, para que possamos implementar bons filtros para o debounce do botão, é necessário uma boa amostragem do estado do botão. Isso nos obriga a trabalharmos com frequências mais altas (em torno de 1 kHz).

Editora Érica - Desbravando o Microcontrolador PIC18 Recursos Avançados - Daniel R. de Sousa, David José de Souza e Nicolás César Lavinia - 1ª Edição



Outro ponto importante que deve ser observado nesse diagrama é o fato de que, durante a leitura do botão, o nível de tensão sobre o pino do PIC não é muito alto. Isso se dá devido a uma certa polarização do LED quando o botão não está pressionado. Por isso, o cálculo dos resistores deve ser feito para garantir um nível mínimo que será interpretado pelo PIC como 1 (nível alto). Isso é ainda mais crítico se estivermos utilizando uma entrada Schmitt Trigger (ST).

» Lógica do exemplo

Chegamos agora ao primeiro exemplo prático. Está na hora de visualizarmos as primeiras linhas de código para este treinamento. Comecemos, então, entendendo a lógica empregada no exemplo para que possamos ver na prática os conceitos já estudados na parte teórica deste capítulo.

O exemplo será composto de um programa capaz de ler quatro botões e tocar o buzzer com uma frequência diferente para cada combinação de botões. Para cada botão existe um LED ligado ao mesmo pino, que será utilizado para indicar os botões pressionados no momento. Utilizaremos os timers e duas interrupções para controlarmos a frequência de leitura dos botões e a frequência do buzzer:

- ✓ **Timer0**: controla a frequência de varredura dos botões;
- ✓ **Timer2**: controla a frequência do som. O som será feito excitando o buzzer com uma onda quadrada de frequência variável. Vamos variar essa frequência alterando o valor de **PR2**, que controla o estouro desse timer.

Desta forma, o programa principal será um loop infinito sem nenhum efeito. Somente vamos esperar o acontecimento das interrupções.

Para a varredura dos botões, ajustaremos a interrupção de **TMR0** para aproximadamente 500 Hz:

Ciclo de Maq.	Prescale	Conta TMR0	Auxiliar	Período	Frequência
1 µs	8	256	-	2048 µs	488 Hz

Além disso, o sistema possui um filtro, regulado pela constante FILTRO_BOTAO, para evitar o debaunce da tecla. Esse filtro garante que a tecla fique pressionada pelo tempo de FILTRO_BOTAO x 2048 µs.

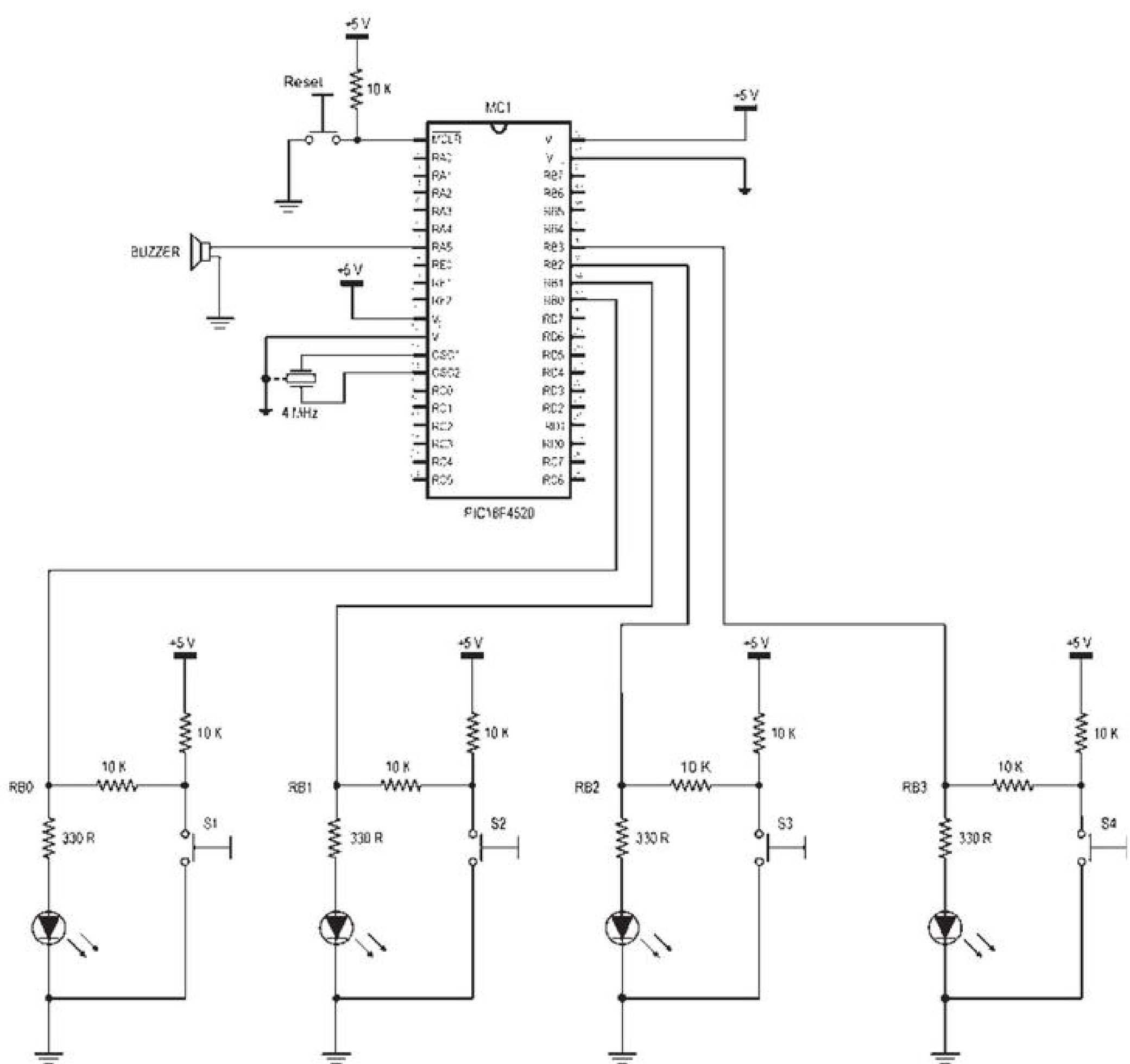
Quanto à frequência do buzzer, será controlada por **TMR2**. Calibramos os valores de pre e postscale para que a frequência da interrupção do **TMR2** varie entre 100 Hz e 2 kHz, com a variação de **PR2** entre 16 e 240:

Ciclo de Maq.	Prescale	Postscale	PR2	Período	Frequência
1 µs	16	2	16	512 µs	1953 Hz
1 µs	16	2	240	7680 µs	130 Hz

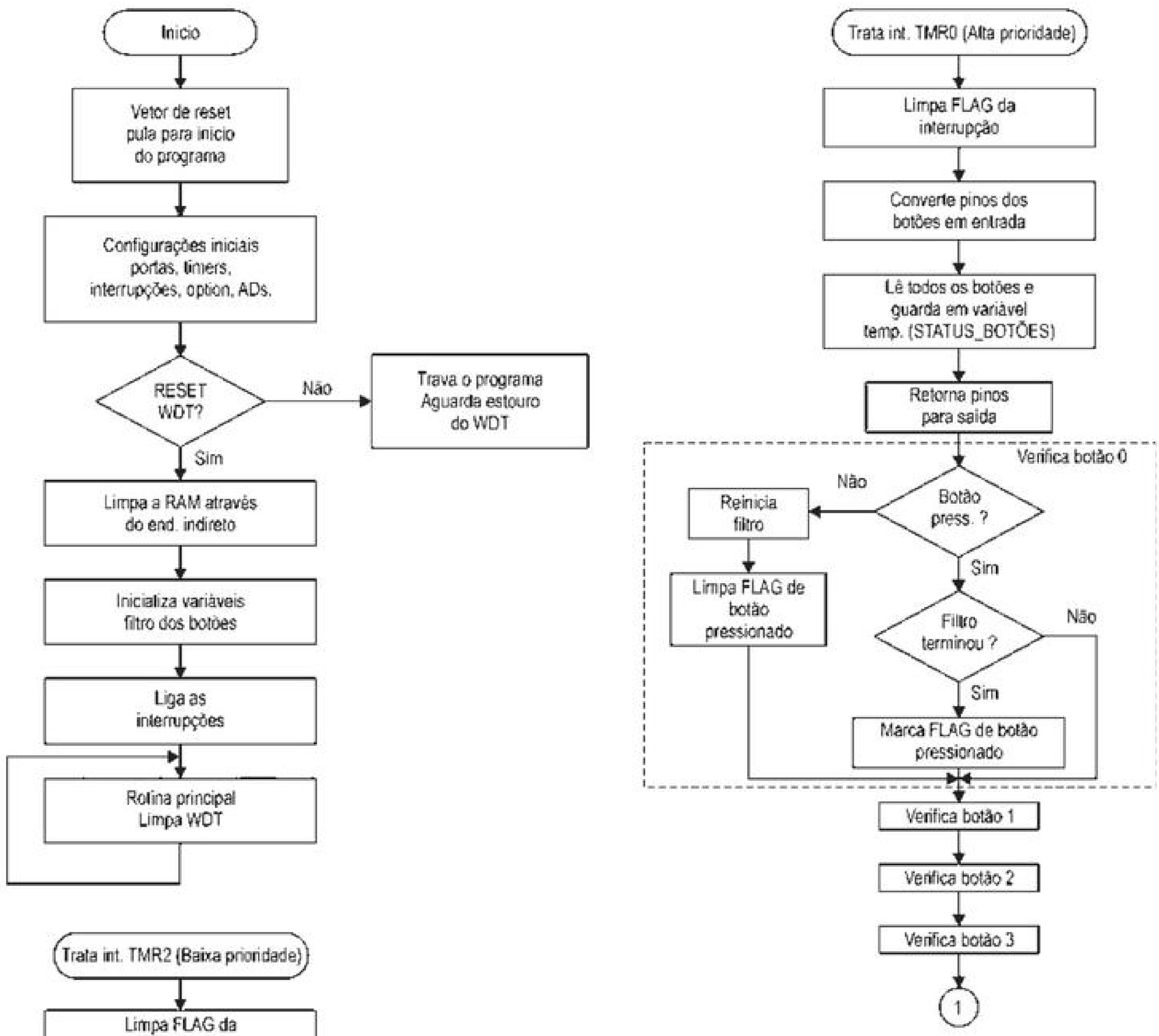
Para facilitar a implementação, a cada interrupção invertemos o estado do pino de acionamento do buzzer. Desta forma, a frequência deste será equivalente à metade da frequência da interrupção do **TMR2**.

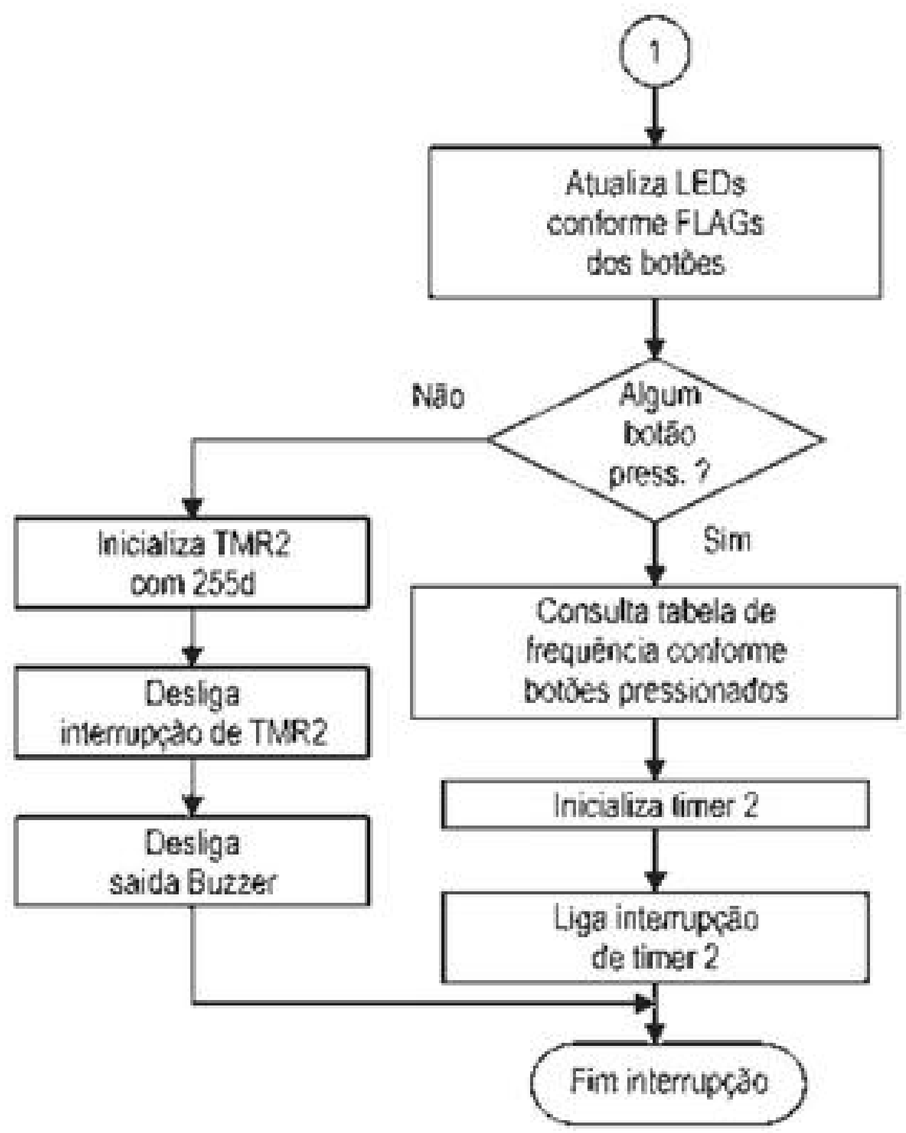
Neste exemplo também vemos o sistema de prioridades de interrupções. O Timer0 será uma interrupção de alta prioridade e o Timer2 de baixa prioridade.

» Esquema elétrico



» Fluxograma


Inicio
Vetor de reset pula para inicio do programa
Configurações iniciais portas, timers, interrupções, option, ADs.
RESET WDT?
Não
Trava o programa Aguarda estouro do WDT
Sim
Limpa a RAM através do end. indireto
Inicializa variáveis filtro dos botões
Liga as interrupções
Rotina principal Limpa WDT
Trata int. TMR2 (Baixa prioridade)
Limpa FLAG da interrupção
Saída buzzer ligada?
Não
Sim
Desliga saída
Liga saída
Fim interrupção
Trata int. TMR0 (Alta prioridade)
Limpa FLAG da interrupção
Converte pinos dos botões em entrada
Lê todos os botões e guarda em variável temp. (STATUS_BOTÕES)
Retorna pinos para saída
Verifica botão 0
Não
Reinicia filtro
Botão press. ?
Sim
Limpa FLAG de botão pressionado
Filtro terminou ?
Não
Sim
Marca FLAG de botão pressionado
Verifica botão 1
Verifica botão 2
Verifica botão 3
1



» Código

```
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                DESBRAVANDO O PIC18 - RECURSOS AVANÇADOS                 *
; *                                EXEMPLO 1                                *
; *                                                                         *
; * DANIEL RODRIGUES DE SOUSA, DAVID JOSÉ DE SOUZA e  NICOLÁS CÉSAR LAVINIA *
; *                                                                         *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *     DATA : 03/08/2009                                                   *
; *   VERSÃO : 1.0                                                          *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                              DESCRIÇÃO GERAL                            *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTE SOFTWARE ESTÁ PREPARADO PARA LER QUATRO BOTÕES E TOCAR O BUZZER COM
;  DURAÇÃO VARIÁVEL CONFORME A TECLA PRESSIONADA, ALÉM DE ACENDER O LED
;  INDICANDO A ÚLTIMA TECLA PRESSIONADA.
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                          CONFIGURAÇÕES PARA GRAVAÇÃO                    *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
    CONFIG  OSC = XT, FCMEN = OFF, IESO = OFF, PWRT = ON, BOREN = ON, BORV = 0
    CONFIG  WDT = ON, WDTPS = 128, MCLRE = ON, LPT1OSC = OFF, PBADEN = OFF
    CONFIG  CCP2MX = PORTC, STVREN = ON, LVP = OFF, DEBUG = OFF, XINST = OFF
    CONFIG  CP0 = OFF, CP1 = OFF, CP2 = OFF, CP3 = OFF, CPB = OFF, CPD = OFF
    CONFIG  WRT0 = OFF, WRT1 = OFF, WRT2 = OFF, WRT3 = OFF, WRTB = OFF
    CONFIG  WRTC = OFF, WRTD = OFF, EBTR0 = OFF, EBTR1 = OFF, EBTR2 = OFF
    CONFIG  EBTR3 = OFF, EBTRB = OFF

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                           DEFINIÇÃO DAS VARIÁVEIS                       *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
    CBLOCK  0X00            ; BANK 0 --> 0X00..0X5F --> ACCESS BANK
```

```
BOTOES_TEMP
STATUS_BOTOES          ; ARMAZENA O ESTADO DOS BOTÕES
STATUS_LEDS            ; ARMAZENA O ESTADO DOS LEDS

FILTRO_BT0             ; FILTRO PARA BOTAO 0
FILTRO_BT1             ; FILTRO PARA BOTAO 1
FILTRO_BT2             ; FILTRO PARA BOTAO 2
FILTRO_BT3             ; FILTRO PARA BOTAO 3

WREG_TEMP_H            ; VARIÁVEIS TEMPORÁRIAS PARA INTERRUPÇÃO
STATUS_TEMP_H          ; DE ALTA PRIORIDADE
BSR_TEMP_H             ; (SALVA E RECUPERA CONTEXTO)

WREG_TEMP_L            ; VARIÁVEIS TEMPORÁRIAS PARA INTERRUPÇÃO
STATUS_TEMP_L          ; DE BAIXA PRIORIDADE
BSR_TEMP_L             ; (SALVA E RECUPERA CONTEXTO)

ENDC

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                DEFINIÇÃO DAS VARIÁVEIS INTERNAS DO PIC                *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  O ARQUIVO DE DEFINIÇÕES DO PIC UTILIZADO DEVE SER REFERENCIADO PARA QUE
;  OS NOMES DEFINIDOS PELA MICROCHIP POSSAM SER UTILIZADOS, SEM A NECESSIDADE
;  DE REDIGITAÇÃO.

#INCLUDE <P18F4520.INC> ; MICROCONTROLADOR UTILIZADO

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                           CONSTANTES INTERNAS                         *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  A DEFINIÇÃO DE CONSTANTES FACILITA A PROGRAMAÇÃO E A MANUTENÇÃO.

FILTRO_BOTAO    EQU     .20 ; FILTRO P/ EVITAR RUIDOS DOS BOTÕES

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                    DECLARAÇÃO DOS FLAGs DE SOFTWARE                   *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  A DEFINIÇÃO DE FLAGs AJUDA NA PROGRAMAÇÃO E ECONOMIZA MEMÓRIA RAM.

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                                 ENTRADAS                              *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  AS ENTRADAS DEVEM SER ASSOCIADAS A NOMES PARA FACILITAR A PROGRAMAÇÃO E
;  FUTURAS ALTERAÇÕES DO HARDWARE.

#DEFINE BOTOES      PORTB           ; ENTRADA DOS BOTÕES (RB0 ATÉ RB3)

#DEFINE BOTAO_0     BOTOES_TEMP,0   ; ESTADO DO BOTÃO 0
                                    ; 1 -> LIBERADO
                                    ; 0 -> PRESSIONADO

#DEFINE BOTAO_1     BOTOES_TEMP,1   ; ESTADO DO BOTÃO 1
                                    ; 1 -> LIBERADO
                                    ; 0 -> PRESSIONADO

#DEFINE BOTAO_2     BOTOES_TEMP,2   ; ESTADO DO BOTÃO 2
                                    ; 1 -> LIBERADO
                                    ; 0 -> PRESSIONADO

#DEFINE BOTAO_3     BOTOES_TEMP,3    ; ESTADO DO BOTÃO 3
                                    ; 1 -> LIBERADO
                                    ; 0 -> PRESSIONADO
```

```
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                                SAÍDAS                                   *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  AS SAÍDAS DEVEM SER ASSOCIADAS A NOMES PARA FACILITAR A PROGRAMAÇÃO E
;  FUTURAS ALTERAÇÕES DO HARDWARE.

#DEFINE LEDS        LATB            ; SAÍDA PARA LEDS (RB0 ATÉ RB3)

#DEFINE BUZZER      LATA,5          ; SAÍDA PARA BUZZER

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                  VETOR DE RESET DO MICROCONTROLADOR                     *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  POSIÇÃO INICIAL PARA EXECUÇÃO DO PROGRAMA

    ORG     0X0000                  ; ENDEREÇO DO VETOR DE RESET
    BRA     CONFIGURACAO            ; PULA PARA CONFIG DEVIDO A REGIÃO
                                    ; DESTINADA ÀS INTERRUPÇÕES

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *               VETOR DE INTERRUPÇÃO DO MICROCONTROLADOR                  *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  POSIÇÃO DE DESVIO DO PROGRAMA QUANDO UMA INTERRUPÇÃO ACONTECE

    ORG    0X0008            ; ENDEREÇO DO VETOR DE INTERRUPÇÃO (ALTA PRIORIDADE)

    MOVFF   WREG,WREG_TEMP_H
    MOVFF   STATUS,STATUS_TEMP_H
    MOVFF   BSR,BSR_TEMP_H
    BRA     INT_HIGH

    ORG    0X0018            ; ENDEREÇO DO VETOR DE INTERRUPÇÃO (ALTA PRIORIDADE)

    MOVFF   WREG,WREG_TEMP_L
    MOVFF   STATUS,STATUS_TEMP_L
    MOVFF   BSR,BSR_TEMP_L
    BRA     INT_LOW

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *         TRATAMENTO DA INTERRUPÇÃO DE BAIXA PRIORIDADE (TIMER 2)         *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ROTINA PARA TRATAMENTO DA INTERRUPÇÃO DE TIMER 2.
;  INVERTE O ESTADO DO PINO DO BUZZER.

INT_LOW
    BCF     PIR1,TMR2IF     ; LIMPA FLAG DA INTERRUPÇÃO

    BTFSS   BUZZER          ; BUZZER LIGADO?
    BRA     LIGA_BUZZER     ; NÃO -  ENTÃO LIGA
                            ; SIM
    BCF     BUZZER          ; DESLIGA O BUZZER
    BRA     SAI_INT_LOW     ; SAI DA INTERRUPÇÃO

LIGA_BUZZER
    BSF     BUZZER          ; LIGA O BUZZER
    BRA     SAI_INT_LOW     ; SAI DA INTERRUPÇÃO

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *         TRATAMENTO DA INTERRUPÇÃO DE ALTA PRIORIDADE (TIMER 0)          *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  TRATAMENTO DA INTERRUPÇÃO DE TIMER 0. RESPONSÁVEL POR CONVERTER OS PINOS
;  DOS BOTÕES EM ENTRADA, SALVAR A SITUAÇÃO DOS MESMOS NUMA VARIÁVEL
;  TEMPORÁRIA, CONVERTER NOVAMENTE OS PINOS PARA SAÍDA E ATUALIZAR OS LEDS.
```

```
INT_HIGH
    BCF     INTCON,T0IF     ; LIMPA FLAG DA INTERRUPÇÃO

; ***************** FOTOGRAFA O ESTADO DOS BOTÕES ****************************

    MOVLW   B'00001111'     ; PREPARA MASCARA
    IORWF   TRISB,F         ; EXECUTA MASCARA (RB0...RB3 ENTRADA)

    BRA     $+.2            ; DELAY(ESTABILIZAÇÃO ENTR.=4 CICLOS)
    BRA     $+.2

    MOVF    BOTOES,W        ; CARREGA NO WORK O ESTADO DOS BOTÕES
    XORLW   0XFF            ; INVERTE TODOS OS BITS DEVIDO À
                            ; LÓGICA INVERTIDA DOS BOTÕES
    ANDLW   B'00001111'     ; LIMPA A PARTE ALTA, QUE NÃO É USADA
    MOVWF   BOTOES_TEMP     ; SALVA EM BOTOES_TEMP

    MOVLW   B'11110000'     ; PREPARA MASCARA
    ANDWF   TRISB,F         ; EXECUTA MASCARA (RB0...RB3 SAÍDA)

; ********* ATUALIZA STATUS_BOTOES CONFORME BOTÕES PRESSIONADOS **************

TESTA_BT0
    BTFSS   BOTAO_0         ; BOTÃO 0 PRESSIONADO?
    BRA     BT0_LIB         ; NÃO - TRATA BOTÃO COMO LIBERADO
                            ; SIM
    DECFSZ  FILTRO_BT0,F    ; DECREMENTA FILTRO DO BOTÃO. ACABOU?
    BRA     TESTA_BT1       ; NÃO - TESTA PRÓXIMO BOTÃO
    BSF     STATUS_BOTOES,0 ; SIM - MARCA BOTÃO COMO PRESSIONADO
    BRA     TESTA_BT1       ; TESTA PRÓXIMO BOTÃO

BT0_LIB
    MOVLW   FILTRO_BOTAO
    MOVWF   FILTRO_BT0      ; REINICIALIZA FILTRO
    BCF     STATUS_BOTOES,0 ; MARCA BOTÃO COMO LIBERADO

TESTA_BT1
    BTFSS   BOTAO_1         ; BOTÃO 1 PRESSIONADO?
    BRA     BT1_LIB         ; NÃO - TRATA BOTÃO COMO LIBERADO
                            ; SIM
    DECFSZ  FILTRO_BT1,F    ; DECREMENTA FILTRO DO BOTÃO. ACABOU?
    BRA     TESTA_BT2       ; NÃO - TESTA PRÓXIMO BOTÃO
    BSF     STATUS_BOTOES,1 ; SIM - MARCA BOTÃO COMO PRESSIONADO
    BRA     TESTA_BT2       ; TESTA PRÓXIMO BOTÃO

BT1_LIB
    MOVLW   FILTRO_BOTAO
    MOVWF   FILTRO_BT1      ; REINICIALIZA FILTRO
    BCF     STATUS_BOTOES,1 ; MARCA BOTÃO COMO LIBERADO

TESTA_BT2
    BTFSS   BOTAO_2         ; BOTÃO 2 PRESSIONADO?
    BRA     BT2_LIB         ; NÃO - TRATA BOTÃO COMO LIBERADO
                            ; SIM
    DECFSZ  FILTRO_BT2,F    ; DECREMENTA FILTRO DO BOTÃO. ACABOU?
    BRA     TESTA_BT3       ; NÃO - TESTA PRÓXIMO BOTÃO
    BSF     STATUS_BOTOES,2 ; SIM - MARCA BOTÃO COMO PRESSIONADO
    BRA     TESTA_BT3       ; TESTA PRÓXIMO BOTÃO

BT2_LIB
    MOVLW   FILTRO_BOTAO
    MOVWF   FILTRO_BT2      ; REINICIALIZA FILTRO
    BCF     STATUS_BOTOES,2 ; MARCA BOTÃO COMO LIBERADO
```

```
TESTA_BT3
    BTFSS   BOTAO_3          ; BOTÃO 3 PRESSIONADO?
    BRA     BT3_LIB          ; NÃO - TRATA BOTÃO COMO LIBERADO
                             ; SIM
    DECFSZ  FILTRO_BT3,F     ; DECREMENTA FILTRO DO BOTÃO. ACABOU?
    BRA     CONTINUA         ; NÃO - CONTINUA EXECUÇÃO DO PROGRAMA
    BSF     STATUS_BOTOES,3  ; SIM - MARCA BOTÃO COMO PRESSIONADO
    BRA     CONTINUA         ; E CONTINUA EXECUÇÃO DO PROGRAMA

BT3_LIB
    MOVLW   FILTRO_BOTAO
    MOVWF   FILTRO_BT3       ; REINICIALIZA FILTRO
    BCF     STATUS_BOTOES,3  ; MARCA BOTÃO COMO LIBERADO

CONTINUA
    MOVFF   STATUS_BOTOES,LEDS  ; ATUALIZA LEDS CONFORME BOTÕES

    MOVF    STATUS_BOTOES,F  ; TODOS OS BOTÕES SOLTOS?
    BNZ     MUDA_FREQ        ; NÃO - DEVE ALT. A FREQ.
                             ; SIM
    MOVLW   .255
    MOVWF   PR2              ; PR2 = 255
    BCF     PIE1,TMR2IE      ; DESLIGA INT. TIMER2

    BCF     BUZZER           ; GARANTE PINO DO BUZZER EM 0
    BRA     SAI_INT_HIGH     ; SAI DA INTERRUPÇÃO

MUDA_FREQ
    RCALL   ACERTA_FREQ      ; CHAMA TABELA DE FREQ.

    MOVWF   PR2              ; ACERTA VALOR DE PR2 CONFORME
                             ; RETORNO DA TABELA
    BSF     PIE1,TMR2IE      ; LIGA INT. TIMER2

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                    FIM DA ROTINA DE INTERRUPÇÃO                       *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  RESTAURAR OS VALORES DE "W" E "STATUS" ANTES DE RETORNAR.

SAI_INT_HIGH
    MOVFF   STATUS_TEMP_H,STATUS
    MOVFF   BSR_TEMP_H,BSR
    MOVFF   WREG_TEMP_H,WREG

    RETFIE                   ; RETORNA DA INTERRUPÇÃO

SAI_INT_LOW
    MOVFF   STATUS_TEMP_L,STATUS
    MOVFF   BSR_TEMP_L,BSR
    MOVFF   WREG_TEMP_L,WREG

    RETFIE                   ; RETORNA DA INTERRUPÇÃO

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                 TABELA DE ACERTO DA FREQUÊNCIA DO BUZZER              *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

ACERTA_FREQ
    MOVF    STATUS_BOTOES,W  ; COLOCA STATUS_BOTOES EM W
    RLNCF   WREG,W           ; DOBRA O VALOR DO PONTEIRO DA TABELA
    ADDWF   PCL,F            ; SOMA STATUS_BOTOES AO PCL
                             ; CRIANDO UMA SELEÇÃO TIPO "CASE"
                             ; CONFORME TABELA ABAIXO:
                             ; - -> LIBERADO
```

```
                                ; X -> PRESSIONADO
                                ; BT3   BT2   BT1   BT0
    RETLW   .255                ;  -     -     -     -
    RETLW   .16                 ;  -     -     -     X
    RETLW   .32                 ;  -     -     X     -
    RETLW   .48                 ;  -     -     X     X
    RETLW   .64                 ;  -     X     -     -
    RETLW   .80                 ;  -     X     -     X
    RETLW   .96                 ;  -     X     X     -
    RETLW   .112                ;  -     X     X     X
    RETLW   .128                ;  X     -     -     -
    RETLW   .144                ;  X     -     -     X
    RETLW   .160                ;  X     -     X     -
    RETLW   .176                ;  X     -     X     X
    RETLW   .192                ;  X     X     -     -
    RETLW   .208                ;  X     X     -     X
    RETLW   .224                ;  X     X     X     -
    RETLW   .240                ;  X     X     X     X

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                CONFIGURAÇÕES INICIAIS DE HARDWARE E SOFTWARE          *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  NESTA ROTINA SÃO INICIALIZADAS AS PORTAS DE I/O DO MICROCONTROLADOR E AS
;  CONFIGURAÇÕES DOS REGISTRADORES ESPECIAIS (SFR). A ROTINA INICIALIZA A
;  MÁQUINA E AGUARDA O ESTOURO DO WDT.

CONFIGURACAO
    CLRF    PORTA               ; GARANTE TODAS AS SAÍDAS EM ZERO
    CLRF    PORTB
    CLRF    PORTC
    CLRF    PORTD
    CLRF    PORTE

    CLRF    LATA
    CLRF    LATB
    CLRF    LATC
    CLRF    LATD
    CLRF    LATE

    MOVLW   B'11011111'
    MOVWF   TRISA               ; CONFIGURA I/O DO PORTA

    MOVLW   B'11110000'
    MOVWF   TRISB               ; CONFIGURA I/O DO PORTB

    MOVLW   B'11111111'
    MOVWF   TRISC               ; CONFIGURA I/O DO PORTC

    MOVLW   B'11111111'
    MOVWF   TRISD               ; CONFIGURA I/O DO PORTD

    MOVLW   B'00000111'
    MOVWF   TRISE               ; CONFIGURA I/O DO PORTE

    MOVLW   B'11000010'         ; CONFIGURA TIMER0
    MOVWF   T0CON               ; TIMER0 HABILITADO
                                ; TIMER0 MODO 8 BITS
                                ; TIMER0 INCREM. PELO CICLO DE MÁQUINA
                                ; TIMER0: PRESCALE - 1:8

    MOVLW   B'00001111'         ; TIMER2: PRESCALE - 1:16
    MOVWF   T2CON               ; POSTSCALE - 1:2

    MOVLW   B'01100000'         ; CONFIGURA INTERRUPÇÕES
```

```
MOVWF   INTCON          ; HABILITA AS INT. DE TMR0 E PERIF.

MOVLW   B'10000100'     ; ALTA PRIORIDADE PARA INT. DO TIMER0
MOVWF   INTCON2         ; PULL-UPs DESABILITADOS

MOVLW   B'10000000'
MOVWF   RCON            ; ATIVA SISTEMA DE PRIORIDADE DAS INTERRUPÇÕES

MOVLW   B'00000000'
MOVWF   IPR1            ; BAIXA PRIORIDADE PARA INT. DO TIMER2

MOVLW   B'00000000'     ; CONFIGURA INTERRUPÇÕES
MOVWF   PIE1            ; DESABILITA AS INT. DE TMR1 E TMR2

MOVLW   B'00001111'     ; CONFIGURA CONVERSOR AD
MOVWF   ADCON1          ; CONFIGURA PORTA, PORTB E PORTE COMO I/O DIGITAL

;  AS INSTRUÇÕES A SEGUIR FAZEM COM QUE O PROGRAMA TRAVE QUANDO HOUVER UM
;  RESET OU POWER-UP, MAS PASSE DIRETO SE O RESET FOR POR WDT. DESTA FORMA,
;  SEMPRE QUE O PIC É LIGADO, O PROGRAMA TRAVA, AGUARDA UM ESTOURO DE WDT
;  E COMEÇA NOVAMENTE. ISTO EVITA PROBLEMAS NO START-UP DO PIC.

BTFSC   RCON,NOT_TO     ; RESET POR ESTOURO DE WDT?
BRA     $               ; NÃO - AGUARDA ESTOURO DO WDT
                        ; SIM

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                       INICIALIZAÇÃO DA RAM                            *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTA ROTINA IRÁ LIMPAR TODA A RAM, DO BANCO 0 ATÉ AO BANCO 15

LFSR    FSR0,0X00       ; APONTA FSR0 PARA ENDEREÇO ZERO

LIMPA_RAM
CLRF    POSTINC0        ; LIMPA A POSIÇÃO E INCREMENTA FSR0
CLRWDT                  ; LIMPA WATCHDOG TIMER
MOVLW   0X0F            ; TESTA SE JÁ CHEGOU AO BANCO 15
SUBWF   FSR0H,W
BNZ     LIMPA_RAM       ; SE NÃO, CONTINUA LIMPANDO

MOVLW   FILTRO_BOTAO    ; INICIALIZA OS FILTROS DOS BOTÕES
MOVWF   FILTRO_BT0
MOVWF   FILTRO_BT1
MOVWF   FILTRO_BT2
MOVWF   FILTRO_BT3

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                          LOOP PRINCIPAL                               *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTA ROTINA PRINCIPAL SIMPLESMENTE LIMPA O WDT, POIS TODA A LÓGICA DO
;  PROGRAMA É TRATADA DENTRO DAS INTERRUPÇÕES.

BSF     INTCON,GIEL
BSF     INTCON,GIEH     ; LIGA AS INTERRUPÇÕES

LOOP
CLRWDT                  ; LIMPA WATCHDOG TIMER
BRA     LOOP            ; VOLTA AO LOOP

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                          FIM DO PROGRAMA                              *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

END                     ; FIM DO PROGRAMA
```

Dicas e comentários

Observe que, pela lógica e funcionalidade do sistema, não seria necessário o tratamento do filtro dos botões, uma vez que o som só é gerado enquanto o botão é pressionado. Entretanto, esse sistema já foi implementado desta forma para fornecer uma lógica eficiente de tratamento de debaunce. Lembre-se também de que em muitos sistemas em que será necessário filtrar corretamente uma entrada pode ser preciso aplicar esse filtro duas vezes, uma para cada mudança de estado, isto é, quando o botão é pressionado e quando é liberado.

Exercícios propostos

Agora que o exemplo foi estudado e esclarecido, aproveite para gerar novos problemas e soluções, seguindo os exercícios propostos:

1. Fazer o sistema trabalhar em outra faixa de frequência, alterando os valores de ajuste do TMR2.
2. Acrescentar um delay enquanto os pinos (LEDs/botões) estão configurados como entrada e verificar o efeito sobre os LEDs.
3. Inibir o buzzer pelo desligamento do Timer2 no lugar da desabilitação da interrupção.
4. Inverter o estado do pino do buzzer por intermédio do XOR.
5. Inverter a escala de frequências, trocando a mais alta pela mais baixa.

5

Varredura de Display de Quatro Dígitos

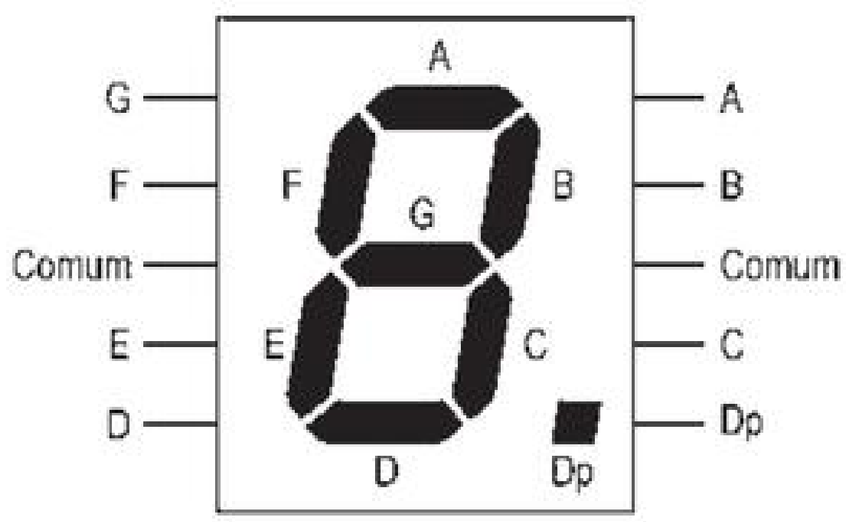
» Introdução

Atualmente, um dos pontos mais importantes de qualquer projeto pode ser considerado a interface com o usuário, já que ela é composta de entradas e saídas. No capítulo anterior começamos a estudar essas interfaces, mas, como ponto de saída, utilizamos somente LEDs e o buzzer. Para aprimorarmos mais a interface de saída, podemos fazer uso de displays. Neste capítulo estudamos os displays de LEDs, que apesar de possuírem recursos limitados, ainda são os mais utilizados no mercado devido ao seu baixo custo e fácil visualização.

» Teoria e recursos do PIC

Os displays com os quais trabalharemos são conhecidos como displays de LEDs de sete segmentos, pois os números são compostos por sete traços. Esses componentes possuem ainda o ponto decimal e são considerados displays numéricos, por não possuírem traços suficientes para a exibição de todas as letras do alfabeto.

Para facilitar a vida do projetista, o mercado padronizou uma nomenclatura para todos os traços do display, possibilitando que tratemos cada um deles individualmente:



Desta forma, temos um pino para controlar cada um dos segmentos (A...G) e mais o ponto decimal (Dp). Os dois pinos adicionais são os comuns, que podem ser ligados a todos os catodos ou anodos dos LEDs internos. Por causa disso, esses displays são fornecidos em dois tipos: catodo comum ou anodo comum. Neste caso, os displays utilizados são catodo comum, isto é, o pino comum deve ser ligado ao terra e os segmentos devem ser ligados ao V_{DD} para acenderem.

Editora Érica - Desbravando o Microcontrolador PIC18 Recursos Avançados - Daniel R. de Sousa, David José de Souza e Nicolás César Lavinia - 1ª Edição

Outra observação importante é que a pinagem descrita no desenho é válida para o tipo de display utilizado na placa proposta (McLab2). Existem displays de outros tamanhos que possuem uma disposição de pinos diferente.

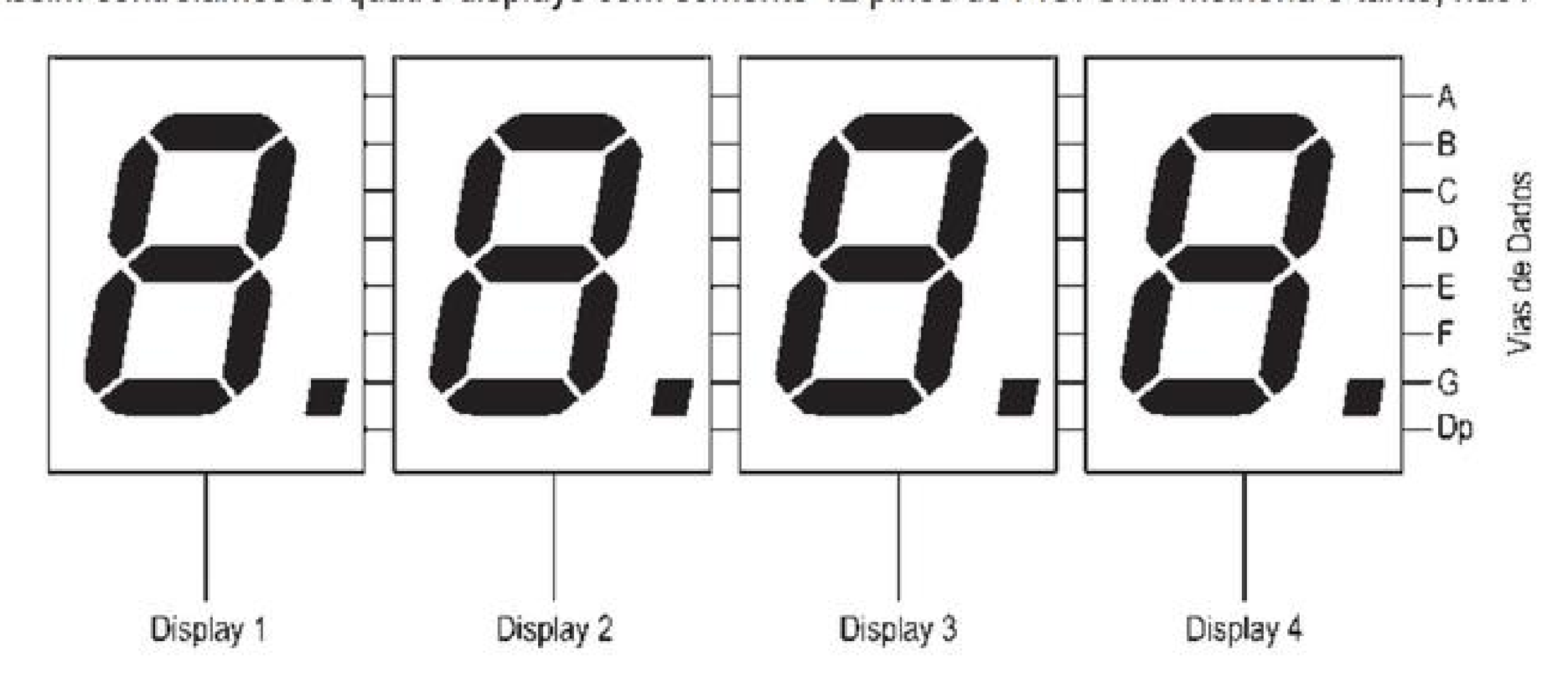
Bem, agora que já conhecemos melhor o componente, vejamos como operá-lo.

Como cada segmento é um LED individual, precisaremos de um pino do PIC para controlar cada um deles. Desta forma, precisaremos de oito pinos para acionar os sete segmentos e mais o ponto decimal. Até aí tudo bem. Acontece que na placa (e na maioria dos projetos existentes) possuímos não um único display, mas um conjunto deles. Por isso, uma maneira de controlarmos vários displays é disponibilizar um conjunto de oito pinos para cada um. Como na placa possuímos quatro displays, são necessários então 32 pinos no total (8x4). Usaríamos o PIC inteiro somente para controlar os quatro displays. Mas qual a solução, então?

O segredo para minimizarmos a quantidade de pinos utilizados é o emprego de um conceito denominado varredura. Interligamos todos os pinos de um mesmo segmento (para todos os displays), criando assim oito vias de dados, de A a Dp. Depois, utilizamos um pino para controlar o comum de cada um dos displays (total de quatro pinos). Assim, quando quisermos escrever em um dos displays, basta informar os segmentos a serem acionados nas vias de dados e ligar o comum do display desejado.

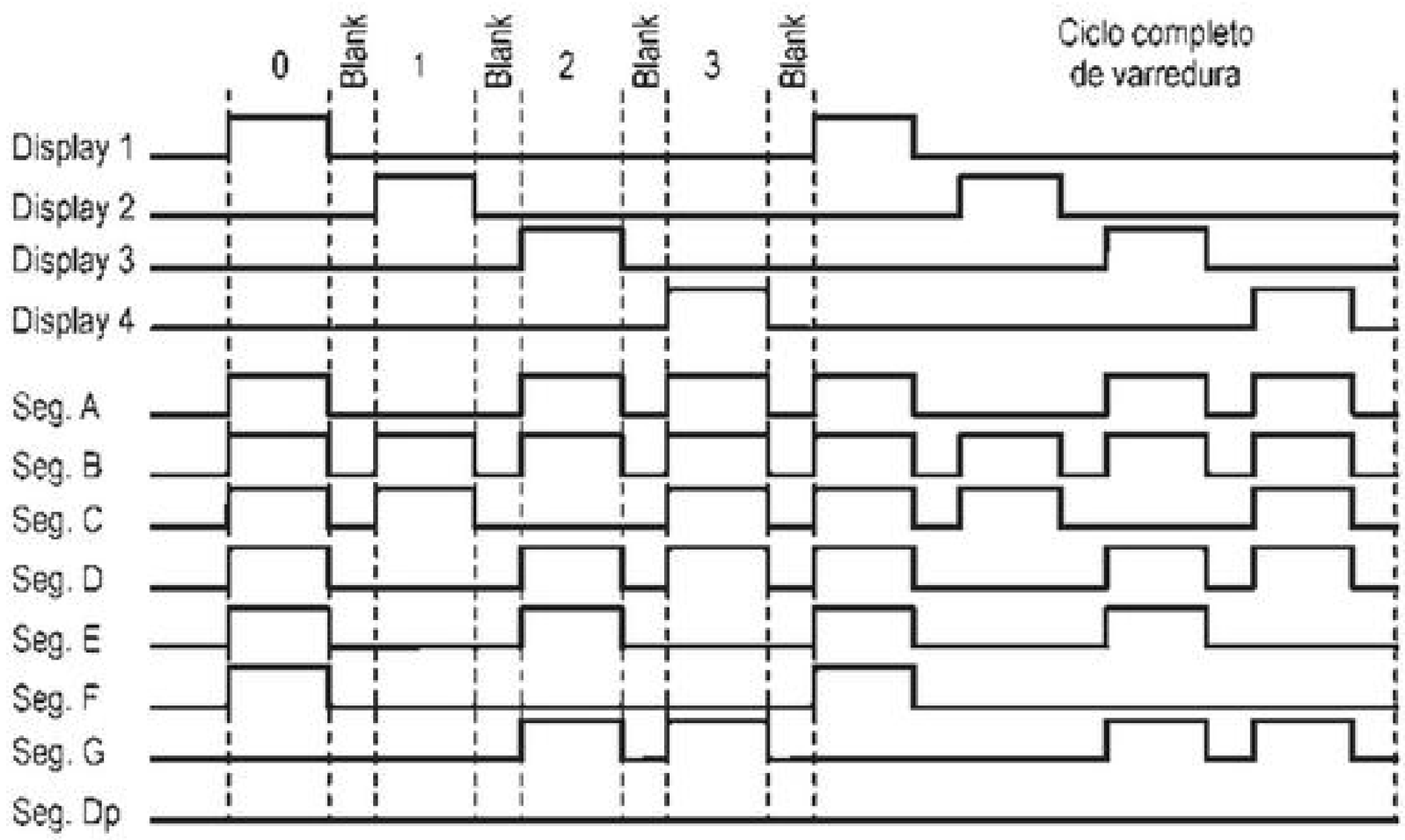
Mas, então, só podemos escrever em um display de cada vez? Na verdade, sim. Acontece que faremos a escrita na sequência, isto é, primeiro escrevemos no display 1, depois no 2, no 3 e no 4, voltando novamente para o 1 e repetindo o ciclo infinitamente. Desta forma "varremos" todos os displays. Faremos isso numa frequência tão rápida que os olhos não serão capazes de perceber que somente um display está realmente aceso de cada vez.

Assim controlamos os quatro displays com somente 12 pinos do PIC. Uma melhoria e tanto, não?



Para que não seja possível percebermos a varredura dos displays, sugerimos que a frequência do ciclo completo (chaveamento de todos os displays) seja de no mínimo 100 Hz. Frequências bem superiores a esta podem ser utilizadas para facilitar a integração da varredura com as demais funções do sistema. Por outro lado, frequências mais altas comprometem a velocidade dos demais componentes utilizados, como os transistores ligados aos pinos comuns dos displays. Caso esses transistores não estejam respondendo corretamente à frequência aplicada, começa a surgir um problema denominado sombreamento, que nada mais é que a exibição do valor de um dígito no display seguinte. Além disso, como o tempo de resposta dos LEDs também não é muito rápido, o problema do sombreamento pode tornar-se crítico.

Para evitarmos esse sombreamento, além da velocidade dos transistores, utilizamos também o recurso de desligar todos os segmentos durante o chaveamento de um display para o outro. Um display com todos os segmentos apagados é chamado de blank.



Outro ponto importante a ser observado na utilização da varredura é que o tempo de permanência de um segmento aceso é o inverso da quantidade de display. No caso, cada display permanece aceso aproximadamente 25% (1/4) do tempo. Por isso, a intensidade luminosa do LED durante a varredura é bem menor que se ele fosse controlado individualmente. Para compensar isso, a corrente aplicada ao LED é normalmente maior. No caso de a varredura ser feita com muitos displays, talvez seja necessária a utilização de displays de alto brilho.

» Lógica do exemplo

O exemplo desenvolvido para este capítulo faz muito mais que simplesmente implementar a varredura dos displays. Trata-se de um contador regressivo de segundos, ou seja, um temporizador capaz de contar até 9.999 segundos.

Para isso, utilizaremos os displays para indicar o valor atual do temporizador. A primeira tecla (S1) não possui nenhuma função. Por outro lado, o LED relacionado a ela (L1) será utilizado para indicar o estado do temporizador:

L1	Descrição
Aceso	Temporizador em contagem regressiva
Apagado	Temporizador paralisado

Os demais botões apresentam as funções de controle do temporizador:

Botão	Descrição
S2	Incrementa o valor inicial em 1 segundo
S3	Decrementa o valor inicial em 1 segundo
S4	Inicia e paralisa o temporizador

Os botões de incremento e decremento operam de forma rotativa, isto é, comutam automaticamente entre 0000 e 9999. Outra característica desses botões é que eles executam suas funções repetidamente quando mantidos pressionados e só funcionam quando o temporizador está paralisado. Ao atingir o valor zero (0000), o temporizador é automaticamente paralisado, desligando-se o LED indicativo (L1).

Editora Érica - Desbravando o Microcontrolador PIC18 Recursos Avançados - Daniel R. de Sousa, David José de Souza e Nicolás César Lavinia - 1ª Edição

Para o sistema de varredura foram criadas quatro variáveis para armazenamento dos dígitos mostrados nos respectivos displays: UNIDADE, DEZENA, CENTENA e MILHAR. Essas variáveis representam o valor atual do temporizador e são incrementadas e decrementadas através dos botões. Na verdade, os botões alteram diretamente o valor da unidade. A lógica do sistema compara esse valor com os limites (0 e 9) para alterar ou não os demais dígitos.

A frequência de varredura é controlada pela interrupção de Timer0. Ajustamos seus parâmetros para que a comutação entre displays (tempo da interrupção) seja de aproximadamente 4 kHz:

Ciclo de Maq.	Prescale	Conta TMR0	Auxiliar	Período	Frequência
1 µs	1	256	-	256 µs	3900 Hz

A frequência de varredura será a frequência de comutação dividida pelo número de displays, que no caso é de aproximadamente 1 kHz. Dentro do tratamento da interrupção de **TMR0** são desligados todos os segmentos (blank), depois é comutado para o próximo display (desligado o atual e ligado o seguinte), dando um delay (cerca de 11 µs no total) para resposta do transistor e só depois é que será carregado o valor do dígito correspondente ao display atualmente ativo.

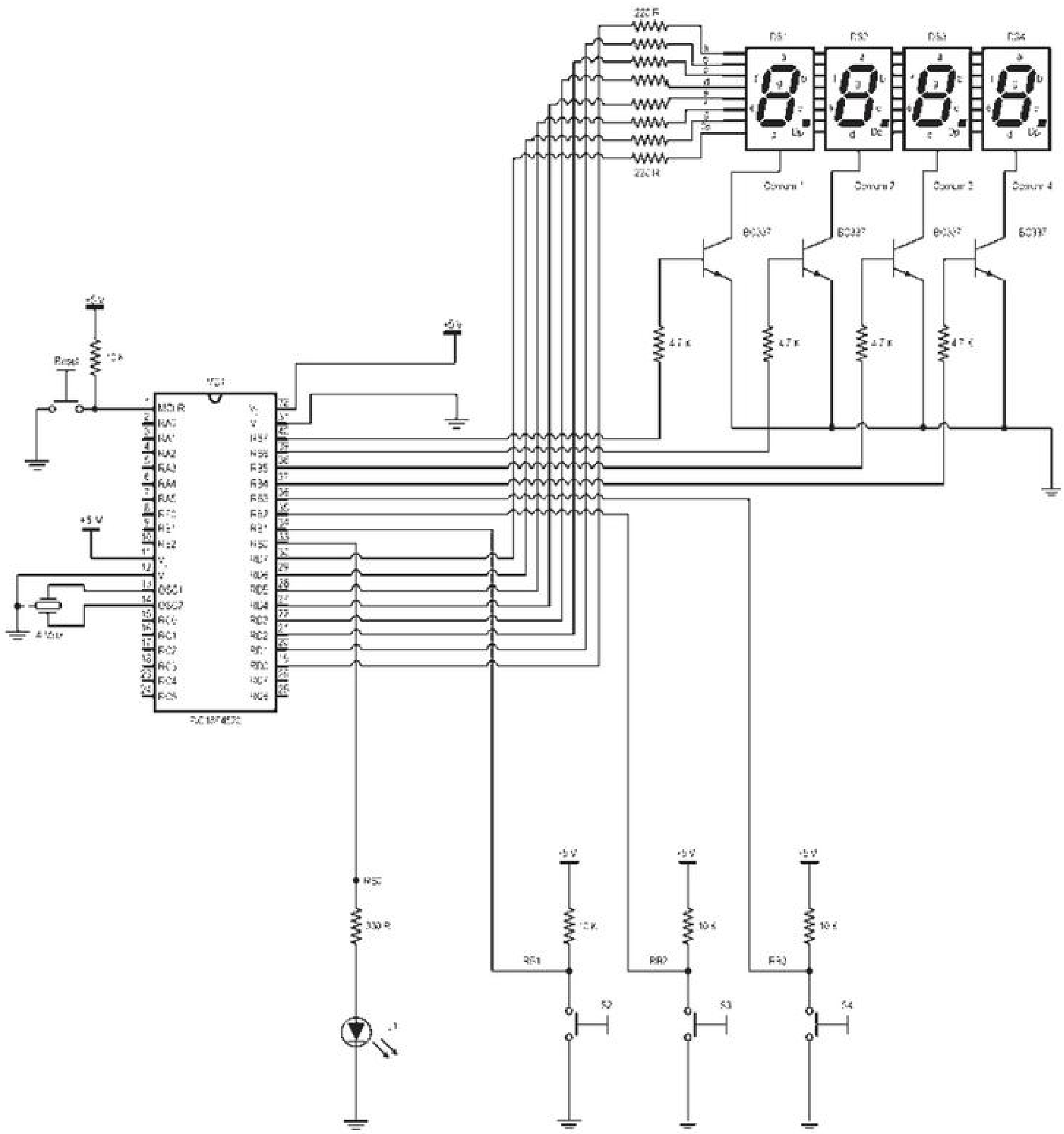
Quanto à contagem dos segundos, utilizamos a interrupção de Timer1 para essa finalidade. Veja os ajustes dos parâmetros para essa interrupção:

Ciclo de Maq.	Prescale	Conta TMR1	Auxiliar	Período	Frequência
1 µs	8	62500	2	1.000.000 µs	1 Hz

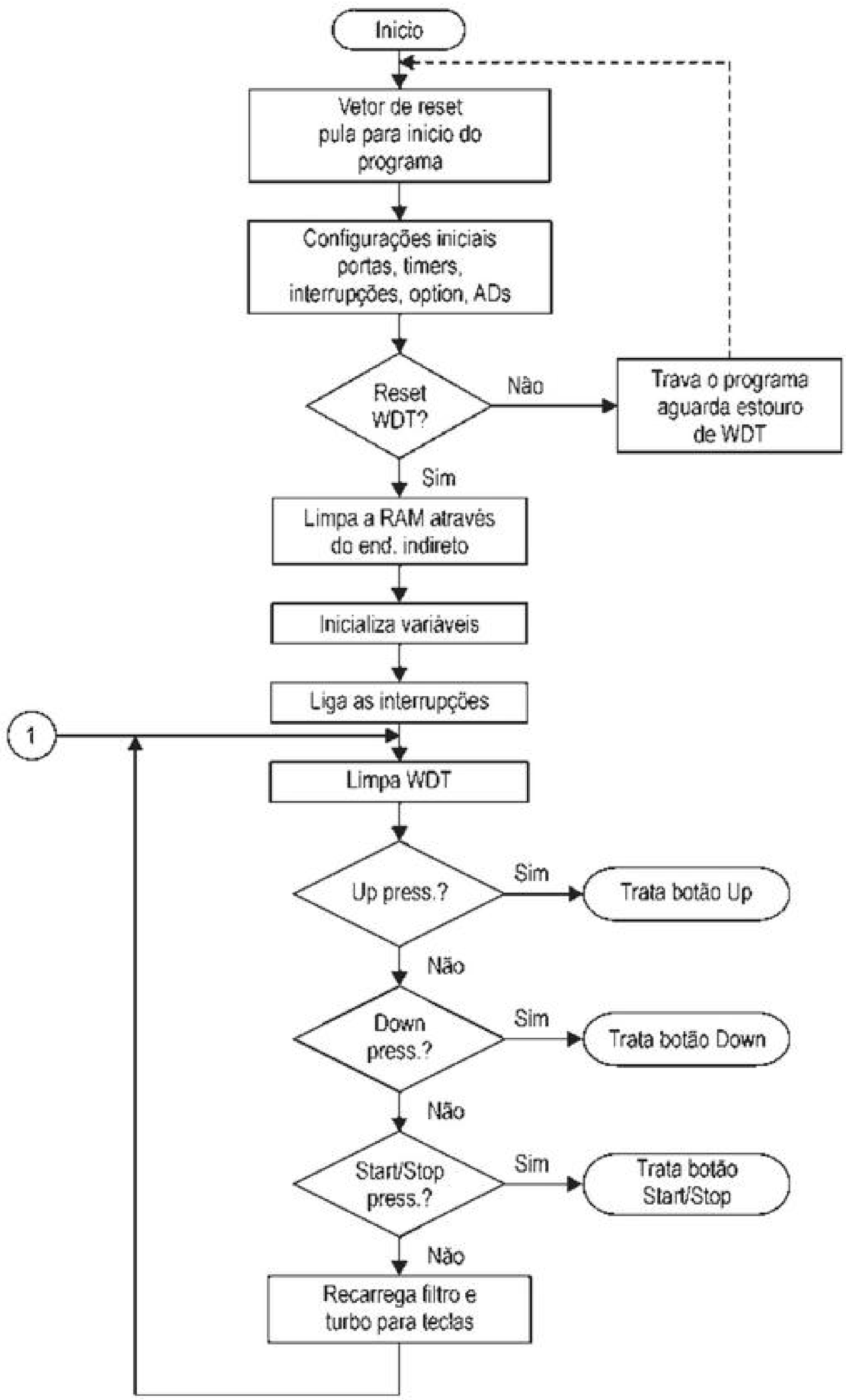
Para isso, configuramos o prescaler de **TMR1** em 1:8 e iniciamos o contador com o valor total menos o desejado para a contagem (65.536 - 62.500). Como esse valor é de 16 bits, a constante de inicialização foi dividida em TMR1_HIGH e TMR1_LOW. Desta maneira, a interrupção acontece a cada 0,5 segundo. Para poder contar um segundo foi criada uma variável auxiliar denominada DIVISOR_TMR1.

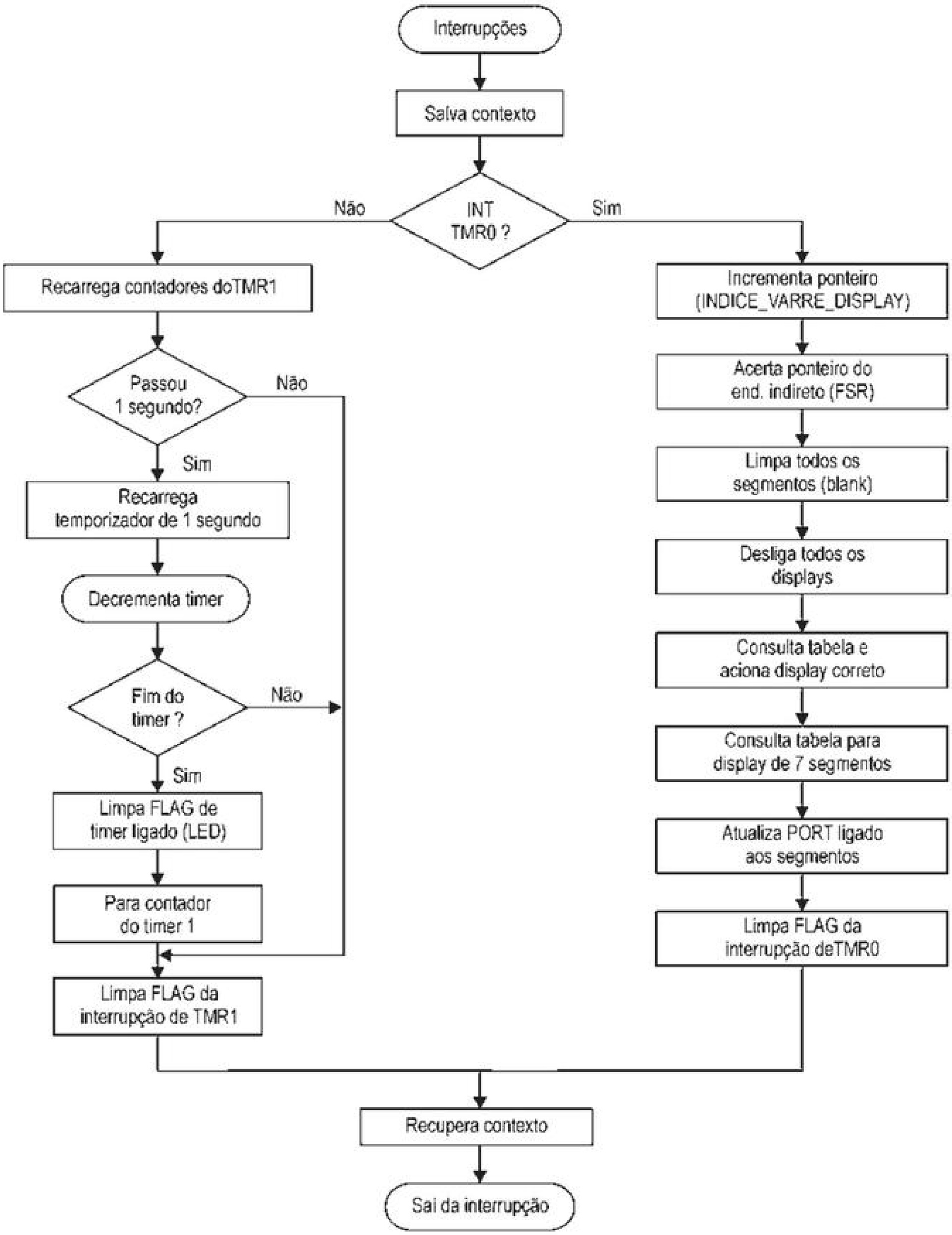
Cada vez que o sistema entrar na interrupção de TMR1 e o contador auxiliar (DIVISOR_TMR1) terminar, o tempo é decrementado, começando pela unidade e chegando até a milhar, se for necessário. Quando o tempo termina (0000), tanto o LED quanto o TMR1 são desligados.

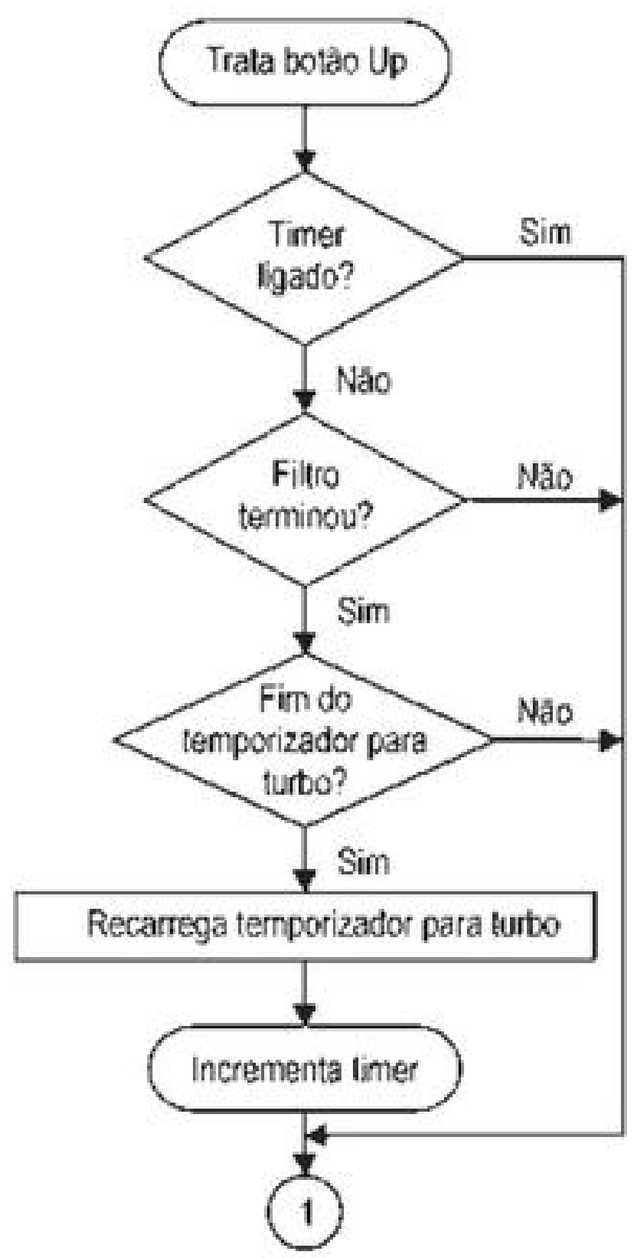
» Esquema elétrico



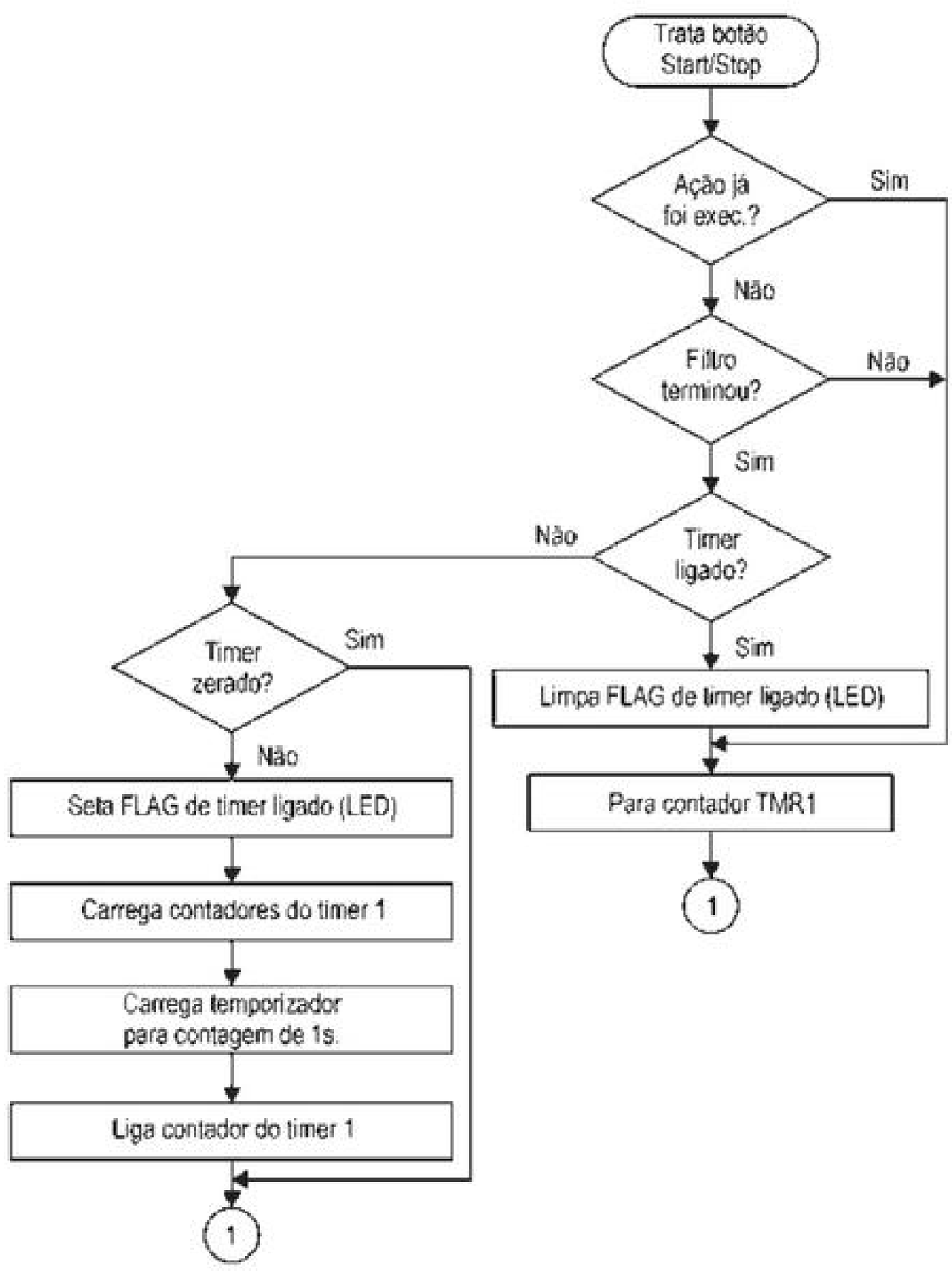
» Fluxograma

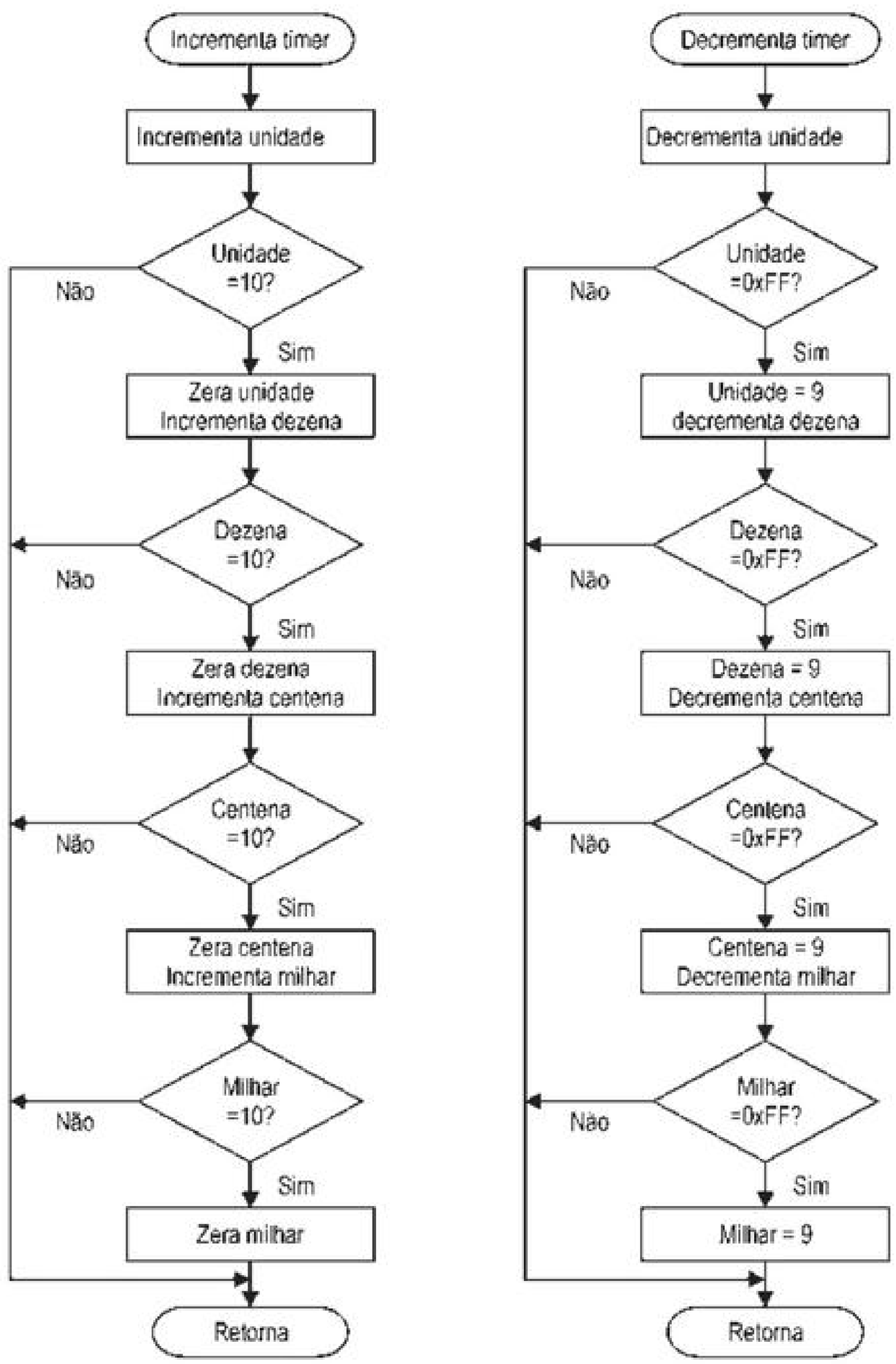

Inicio
Vetor de reset pula para inicio do programa
Configurações iniciais portas, timers, interrupções, option, ADs
Reset WDT?
Não
Trava o programa aguarda estouro de WDT
Sim
Limpa a RAM através do end. indireto
Inicializa variáveis
Liga as interrupções
1
Limpa WDT
Up press.?
Sim
Trata botão Up
Não
Down press.?
Sim
Trata botão Down
Não
Start/Stop press.?
Sim
Trata botão Start/Stop
Não
Recarrega filtro e turbo para teclas


Interrupções
Salva contexto
INT TMR0 ?
Não
Sim
Recarrega contadores doTMR1
Passou 1 segundo?
Não
Sim
Recarrega temporizador de 1 segundo
Decrementa timer
Fim do timer ?
Não
Sim
Limpa FLAG de timer ligado (LED)
Para contador do timer 1
Limpa FLAG da interrupção de TMR1
Incrementa ponteiro (INDICE_VARRE_DISPLAY)
Acerta ponteiro do end. indireto (FSR)
Limpa todos os segmentos (blank)
Desliga todos os displays
Consulta tabela e aciona display correto
Consulta tabela para display de 7 segmentos
Atualiza PORT ligado aos segmentos
Limpa FLAG da interrupção deTMR0
Recupera contexto
Sai da interrupção


Trata botão Up
Timer ligado?
Sim
Não
Filtro terminou?
Não
Sim
Fim do temporizador para turbo?
Não
Sim
Recarrega temporizador para turbo
Incrementa timer
1

Trata botão Down
Timer ligado?
Sim
Não
Filtro terminou?
Não
Sim
Fim do temporizador para turbo?
Não
Sim
Recarrega temporizador para turbo
Decrementa timer
1


Trata botão Start/Stop
Ação já foi exec.?
Sim
Não
Filtro terminou?
Não
Sim
Timer ligado?
Não
Sim
Timer zerado?
Sim
Não
Seta FLAG de timer ligado (LED)
Carrega contadores do timer 1
Carrega temporizador para contagem de 1s.
Liga contador do timer 1
1
Limpa FLAG de timer ligado (LED)
Para contador TMR1
1



» Código

```
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                DESBRAVANDO O PIC18 - RECURSOS AVANÇADOS                *
; *                              EXEMPLO 2                                 *
; *                                                                        *
; * DANIEL RODRIGUES DE SOUSA, DAVID JOSÉ DE SOUZA e  NICOLÁS CÉSAR LAVINIA *
; *                                                                        *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *     DATA : 03/08/2009                                                  *
; *   VERSÃO : 1.0                                                         *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                            DESCRIÇÃO GERAL                             *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

```
;  ESTE EXEMPLO FOI PREPARADO PARA DEMONSTRAR O FUNCIONAMENTO DO TIMER DE
;  16 BITS DO PIC (TMR1) E DA VARREDURA DE DISPLAYS.
;  CONSISTE NUM TEMPORIZADOR DE SEGUNDOS. DOIS BOTÕES FORAM UTILIZADOS PARA
;  PROGRAMAR O TEMPO DA CONTAGEM. UM OUTRO BOTÃO FOI UTILIZADO PARA DISPARAR
;  O CONTADOR. O TEMPORIZADOR CONSEGUE CONTAR ATÉ 9999 SEGUNDOS, DE FORMA QUE
;  OS 4 DISPLAYS DE 7 SEGMENTOS FORAM NECESSÁRIOS. A CONTAGEM É REGRESSIVA.
;  UM LED INDICA QUE O TEMPORIZADOR ESTÁ OPERANDO. QUANDO O SISTEMA CHEGA
;  A 0000 (ZERO), O LED É DESLIGADO AUTOMATICAMENTE.
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                          CONFIGURAÇÕES PARA GRAVAÇÃO                  *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
   CONFIG  OSC = XT, FCMEN = OFF, IESO = OFF, PWRT = ON, BOREN = ON, BORV = 0
   CONFIG  WDT = ON, WDTPS = 128, MCLRE = ON, LPT1OSC = OFF, PBADEN = OFF
   CONFIG  CCP2MX = PORTC, STVREN = ON, LVP = OFF, DEBUG = OFF, XINST = OFF
   CONFIG  CP0 = OFF, CP1 = OFF, CP2 = OFF, CP3 = OFF, CPB = OFF, CPD = OFF
   CONFIG  WRT0 = OFF, WRT1 = OFF, WRT2 = OFF, WRT3 = OFF, WRTB = OFF
   CONFIG  WRTC = OFF, WRTD = OFF, EBTR0 = OFF, EBTR1 = OFF, EBTR2 = OFF
   CONFIG  EBTR3 = OFF, EBTRB = OFF

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                           DEFINIÇÃO DAS VARIÁVEIS                     *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
   CBLOCK  0X00            ; BANK 0 --> 0X00..0X5F --> ACCESS BANK

       UNIDADE             ; (LSD)
       DEZENA              ;
       CENTENA             ;
       MILHAR              ; (MSD)

       FILTRO_BOTOES       ; FILTRO PARA RUIDOS

       TEMPO_TURBO         ; TEMPORIZADOR P/ TURBO DAS TECLAS

       INDICE_VARRE_DISPLAY; INDEXADOR P/ VARREDURA DOS DISPLAYS

       DIVISOR_TMR1        ; CONTADOR AUXILIAR P/ SEGUNDOS

   ENDC

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                DEFINIÇÃO DAS VARIÁVEIS INTERNAS DO PIC                *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  O ARQUIVO DE DEFINIÇÕES DO PIC UTILIZADO DEVE SER REFERENCIADO PARA QUE
;  OS NOMES DEFINIDOS PELA MICROCHIP POSSAM SER UTILIZADOS, SEM A NECESSIDADE
;  DE REDIGITAÇÃO.
   #INCLUDE <P18F4520.INC> ; ARQUIVO DE DEFINIÇÕES DO PIC ATUAL

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                            CONSTANTES INTERNAS                        *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  A DEFINIÇÃO DE CONSTANTES FACILITA A PROGRAMAÇÃO E A MANUTENÇÃO.

FILTRO_TECLA    EQU .200                ; FILTRO P/ EVITAR RUIDOS DOS BOTÕES
TURBO_TECLA     EQU .70                 ; TEMPORIZADOR P/ TURBO DAS TECLAS

TMR1_HIGH       EQU HIGH (.65536-.62500)    ; VALOR PARA CONTAGEM DE 62500
TMR1_LOW        EQU LOW  (.65536-.62500)    ; CICLOS DE CONTAGEM DO TMR1
                                            ; (PROGRAMADO P/PRESCALER DE 1:8)

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                     DECLARAÇÃO DOS FLAGs DE SOFTWARE                  *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  A DEFINIÇÃO DE FLAGs AJUDA NA PROGRAMAÇÃO E ECONOMIZA MEMÓRIA RAM.
```

```
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                                ENTRADAS                                 *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  AS ENTRADAS DEVEM SER ASSOCIADAS A NOMES PARA FACILITAR A PROGRAMAÇÃO E
;  FUTURAS ALTERAÇÕES DO HARDWARE.

#DEFINE BT_UP           PORTB,1     ; ESTADO DO BOTÃO 1
                                    ; 1 -> LIBERADO
                                    ; 0 -> PRESSIONADO

#DEFINE BT_DOWN         PORTB,2       ; ESTADO DO BOTÃO 2
                                    ; 1 -> LIBERADO
                                    ; 0 -> PRESSIONADO

#DEFINE BT_START_STOP   PORTB,3     ; ESTADO DO BOTÃO 3
                                    ; 1 -> LIBERADO
                                    ; 0 -> PRESSIONADO

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                               SAÍDAS                                   *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  AS SAÍDAS DEVEM SER ASSOCIADAS A NOMES PARA FACILITAR A PROGRAMAÇÃO E
;  FUTURAS ALTERAÇÕES DO HARDWARE.

#DEFINE ESTADO_TIMER    LATB,0      ; LED DE ESTADO DO TIMER
                                    ; (FUNCIONA TAMBÉM COMO FLAG)
                                    ; 1 -> TIMER CONTANDO
                                    ; 0 -> TIMER PARADO

#DEFINE MUX             LATB        ; MUX PARA ACIONAMENTO DOS DISPLAYS
                                    ; (DE RB4 ATÉ RB7)

#DEFINE SEGMENTOS       LATD        ; SEGMENTOS DOS DISPLAYS

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                  VETOR DE RESET DO MICROCONTROLADOR                     *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  POSIÇÃO INICIAL PARA EXECUÇÃO DO PROGRAMA

    ORG     0X0000          ; ENDEREÇO DO VETOR DE RESET
    BRA     CONFIGURACAO    ; PULA PARA CONFIG DEVIDO A REGIÃO
                            ; DESTINADA ÀS INTERRUPÇÕES


; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                           VETOR DE INTERRUPÇÃO                          *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  POSIÇÃO DE DESVIO DO PROGRAMA QUANDO UMA INTERRUPÇÃO ACONTECE

    ORG         0X0008            ; ENDEREÇO DO VETOR DE INTERRUPÇÃO

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                 TESTA QUAL INTERRUPÇÃO FOI SOLICITADA                   *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  TESTA O FLAG DAS INTERRUPÇÕES PARA SABER PARA QUAL ROTINA DESVIAR.

    BTFSS   INTCON,T0IF     ; FOI INTERRUPÇÃO DE TMR0 ?
    BRA     INT_TMR1        ; NÃO - ENTÃO PULA P/ INT_TMR1
                            ; SIM

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                   TRATAMENTO DA INTERRUPÇÃO DE TIMER 0                  *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ROTINA PARA EXECUTAR AS AÇÕES NECESSÁRIAS SEMPRE QUE A INTERRUPÇÃO
```

```
;  ACONTECE. NESTE CASO, A INTERRUPÇÃO ESTA SENDO UTILIZADA PARA GERAR A
;  FREQUÊNCIA DE VARREDURA DOS DISPLAYS. POR ISSO, CADA VEZ QUE ELA ACONTECER,
;  O PRÓXIMO DISPLAY SERÁ ACIONADO.

INT_TMR0
    INCF    INDICE_VARRE_DISPLAY,F  ; INCR. O ÍNDICE DE VAR. DOS DISPLAYS

    MOVLW   B'00000011'
    ANDWF   INDICE_VARRE_DISPLAY,F  ; LIMITA CONTAGEM DE 0 A 3

    MOVF    INDICE_VARRE_DISPLAY,W  ; CARREGA NO WORK O VALOR DO ÍNDICE
    ADDLW   UNIDADE                 ; SOMA ENDEREÇO DO PRIMEIRO DÍGITO
    MOVWF   FSR0L                   ; SALVA RESULTADO NO FSR, APONTANDO
                                    ; PARA O ENDEREÇO DO DÍGITO ATUAL.
                                    ; (ENDEREÇAMENTO INDIRETO)

    CLRF    SEGMENTOS               ; LIMPA OS SEGMENTOS (BLANK)
                                    ; UTILIZADO P/ EVITAR SOMBRAS NOS
                                    ; DISPLAYS

    MOVLW   B'00001111'             ; PREPARA MÁSCARA
    ANDWF   MUX,F                   ; EXECUTA MÁSCARA (DESLIGA OS DISP.)

    MOVF    INDICE_VARRE_DISPLAY,W  ; SALVA NO WORK O VALOR DO ÍNDICE
    RCALL   TABELA_MUX              ; CONSULTA TABELA MUX
    IORWF   MUX,F                   ; ATUALIZA MUX, SELECIONANDO O
                                    ; DISPLAYS CORRETO PARA O MOMENTO

    BRA     $+.2                    ; DELAY DE 2US
                                    ; (TEMPO DE RESPOSTA DO TRANSISTOR)

    MOVF    INDF0,W                 ; RECUPERA NO WORK O VALOR DO DÍGITO
    RCALL   TABELA_DISPLAY_7_SEG    ; CONSULTA TABELA P/ DISPLAYS
    MOVWF   SEGMENTOS               ; ATUALIZA OS SEGMENTOS, ESCREVENDO
                                    ; O VALOR DO DÍGITO CORRETO (PORTD)

SAI_INT_TMR0
    BCF     INTCON,T0IF             ; LIMPA FLAG DA INTERRUPÇÃO DE TMR0
    RETFIE  FAST                    ; RETORNA DA INTERRUPÇÃO

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                    TRATAMENTO DA INTERRUPÇÃO DE TIMER 1               *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

;  ROTINA PARA EXECUTAR AS AÇÕES NECESSÁRIAS SEMPRE QUE A INTERRUPÇÃO
;  ACONTECE. NESTE CASO, A INTERRUPÇÃO ESTA SENDO UTILIZADA PARA CONTAR O
;  TEMPO DO TEMPORIZADOR. POR ISSO, CADA VEZ QUE ELA ACONTECER O VALOR DO
;  TIMER SERÁ DECREMENTADO, CASO JÁ TENHA SE PASSADO 1SEG.
;  PERIODO DA INTERRUPÇÃO: 1US (CICLO DE MAQUINA) * 8 (PRESCALER DO TMR1) *
;  62500 (CONTAGEM DO TMR1) = 0,5SEG.

INT_TMR1
    MOVLW   TMR1_HIGH       ; RECARREGA CONTADOR DO TMR1
    MOVWF   TMR1H           ; PERIODICIDADE DE 0,5SEG.
    MOVLW   TMR1_LOW
    MOVWF   TMR1L

    DECFSZ  DIVISOR_TMR1,F  ; PASSOU-SE 1 SEGUNDO ?
    BRA     SAI_INT_TMR1    ; NÃO - ENTÃO SAI DA INTERRUPÇÃO
                            ; SIM
    MOVLW   .2
    MOVWF   DIVISOR_TMR1    ; RECARREGA CONTADOR DE 1SEG.

    RCALL   DECREMENTA_TIMER; DECREMENTA O VALOR DO TIMER
```

```
    MOVF    UNIDADE,F
    BNZ     SAI_INT_TMR1

    MOVF    DEZENA,F
    BNZ     SAI_INT_TMR1

    MOVF    CENTENA,F
    BNZ     SAI_INT_TMR1

    MOVF    MILHAR,F        ; FINAL DA CONTAGEM ? (TIMER=0?)
    BNZ     SAI_INT_TMR1    ; NÃO - SAI DA INTERRUPÇÃO

                            ; SIM
    BCF     ESTADO_TIMER    ; DESLIGA LED DE TIMER OPERANDO
    BCF     T1CON,TMR1ON    ; PARALIZA CONTADOR DO TMR1

SAI_INT_TMR1
    BCF     PIR1,TMR1IF     ; LIMPA FLAG DA INTERRUPÇÃO DE TMR1

    RETFIE  FAST            ; RETORNA DA INTERRUPÇÃO (HABILITA GIE)

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                TABELA PARA OS DISPLAYS DE 7 SEGMENTOS               *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ROTINA PARA CONVERSÃO DO VALOR NÚMÉRICO DO DÍGITO EM RELAÇÃO AOS SEGMENTOS
;  QUE DEVEM SER ACESOS E APAGADOS NO DISPLAY

TABELA_DISPLAY_7_SEG
    ANDLW   B'00001111'     ; EXECUTA MASCARA P/ EVITAR PULOS ERRADOS
    RLNCF   WREG,W          ; DOBRA O VALOR DO PONTEIRO DA TABELA
    ADDWF   PCL,F           ; SOMA DESLOCAMENTO AO PROGRAM COUNTER,
                            ; GERANDO UMA TABELA DO TIPO "CASE".
;             PGFEDCBA      ; POSIÇÃO RELATIVA AOS SEGMENTOS
    RETLW   B'00111111'     ; 0H - 0
    RETLW   B'00000110'     ; 1H - 1
    RETLW   B'01011011'     ; 2H - 2
    RETLW   B'01001111'     ; 3H - 3
    RETLW   B'01100110'     ; 4H - 4
    RETLW   B'01101101'     ; 5H - 5
    RETLW   B'01111101'     ; 6H - 6
    RETLW   B'00000111'     ; 7H - 7
    RETLW   B'01111111'     ; 8H - 8
    RETLW   B'01101111'     ; 9H - 9
    RETLW   B'00000000'     ; AH - BLANK
    RETLW   B'00000000'     ; BH - BLANK
    RETLW   B'00000000'     ; CH - BLANK
    RETLW   B'00000000'     ; DH - BLANK
    RETLW   B'00000000'     ; EH - BLANK
    RETLW   B'00000000'     ; FH - BLANK

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                TABELA PARA ACIONAMENTO DOS DISPLAYS                 *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ROTINA PARA CONVERTER O DÍGITO ATUAL EM RELAÇÃO AO PORT QUE DEVE SER
;  LIGADO PARA ACIONAMENTO DO DISPLAY RELACIONADO.

TABELA_MUX
    RLNCF   WREG,W          ; DOBRA O VALOR DO PONTEIRO DA TABELA
    ADDWF   PCL,F           ; SOMA DESLOCAMENTO AO PROGRAM COUNTER
                            ; GERANDO UMA TABELA DO TIPO "CASE".
    RETLW   B'00010000'     ; 0 - ACIONA DISPLAY 0
    RETLW   B'00100000'     ; 1 - ACIONA DISPLAY 1
    RETLW   B'01000000'     ; 2 - ACIONA DISPLAY 2
```

```
    RETLW   B'10000000'     ; 2 - ACIONA DISPLAY 3

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *              ROTINA PARA INCREMENTAR O VALOR DO TIMER (BCD)            *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ROTINA UTILIZADA PARA INCREMENTAR O VALOR DOS REGISTRADORES UNIDADE,
;  DEZENA, CENTENA E MILHAR, QUE SÃO OS CONTADORES DO TIMER. A CONTAGEM É
;  FEITA DIRETAMENTE EM BCD.

INCREMENTA_TIMER
    INCF    UNIDADE,F       ; INCREMENTA UNIDADE

    MOVLW   .10
    XORWF   UNIDADE,W
    BTFSS   STATUS,Z        ; UNIDADE = 10 ?
    RETURN                  ; NÃO - RETORNA
                            ; SIM
    CLRF    UNIDADE         ; ZERA A UNIDADE
    INCF    DEZENA,F        ; INCREMENTA A DEZENA

    MOVLW   .10
    XORWF   DEZENA,W
    BTFSS   STATUS,Z        ; DEZENA = 10 ?
    RETURN                  ; NÃO - RETORNA
                            ; SIM
    CLRF    DEZENA          ; ZERA A DEZENA
    INCF    CENTENA,F       ; INCREMENTA A CENTENA

    MOVLW   .10
    XORWF   CENTENA,W
    BTFSS   STATUS,Z        ; CENTENA = 10 ?
    RETURN                  ; NÃO - RETORNA
                            ; SIM
    CLRF    CENTENA         ; ZERA A CENTENA
    INCF    MILHAR,F        ; INCREMENTA O MILHAR

    MOVLW   .10
    XORWF   MILHAR,W
    BTFSC   STATUS,Z        ; MILHAR = 10 ?
    CLRF    MILHAR          ; SIM - ZERA MILHAR

    RETURN                  ; NÃO - RETORNA


; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *              ROTINA PARA DECREMENTAR O VALOR DO TIMER (BCD)            *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ROTINA UTILIZADA PARA DECREMENTAR O VALOR DOS REGISTRADORES UNIDADE,
;  DEZENA, CENTENA E MILHAR, QUE SÃO OS CONTADORES DO TIMER. A CONTAGEM É
;  FEITA DIRETAMENTE EM BCD.

DECREMENTA_TIMER
    DECF    UNIDADE,F       ; INCREMENTA UNIDADE

    MOVLW   0XFF
    XORWF   UNIDADE,W
    BTFSS   STATUS,Z        ; UNIDADE = 0XFF ?
    RETURN                  ; NÃO - RETORNA
                            ; SIM
    MOVLW   .9
    MOVWF   UNIDADE         ; CARREGA UNIDADE COM 9
    DECF    DEZENA,F        ; DECREMENTA A DEZENA

    MOVLW   0XFF
```

```
XORWF   DEZENA,W
BTFSS   STATUS,Z          ; DEZENA = 0XFF ?
RETURN                    ; NÃO - RETORNA
                          ; SIM
MOVLW   .9
MOVWF   DEZENA            ; CARREGA A DEZENA COM 9
DECF    CENTENA,F         ; DECREMENTA A CENTENA

MOVLW   0XFF
XORWF   CENTENA,W
BTFSS   STATUS,Z          ; CENTENA = 0XFF ?
RETURN                    ; NÃO - RETORNA
                          ; SIM
MOVLW   .9
MOVWF   CENTENA           ; CARREGA CENTENA COM 9
DECF    MILHAR,F          ; DECREMENTA O MILHAR

MOVLW   0XFF
XORWF   MILHAR,W
BTFSS   STATUS,Z          ; MILHAR = 0XFF ?
RETURN                    ; NÃO - RETORNA
                          ; SIM
MOVLW   9
MOVWF   MILHAR            ; CARREGA O MILHAR COM 9
RETURN                    ; RETORNA

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                CONFIGURAÇÕES INICIAIS DE HARDWARE E SOFTWARE          *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  NESTA ROTINA SÃO INICIALIZADAS AS PORTAS DE I/O DO MICROCONTROLADOR E AS
;  CONFIGURAÇÕES DOS REGISTRADORES ESPECIAIS (SFR). A ROTINA INICIALIZA A
;  MÁQUINA E AGUARDA O ESTOURO DO WDT.

CONFIGURACAO
CLRF    PORTA             ; GARANTE TODAS AS SAÍDAS EM ZERO
CLRF    PORTB
CLRF    PORTC
CLRF    PORTD
CLRF    PORTE

CLRF    LATA
CLRF    LATB
CLRF    LATC
CLRF    LATD
CLRF    LATE

MOVLW   B'11111111'
MOVWF   TRISA             ; CONFIGURA I/O DO PORTA

MOVLW   B'00001110'
MOVWF   TRISB             ; CONFIGURA I/O DO PORTB

MOVLW   B'11111111'
MOVWF   TRISC             ; CONFIGURA I/O DO PORTC

MOVLW   B'00000000'
MOVWF   TRISD             ; CONFIGURA I/O DO PORTD

MOVLW   B'00000111'
MOVWF   TRISE             ; CONFIGURA I/O DO PORTE

MOVLW   B'11011000'       ; CONFIGURA TIMER0
MOVWF   T0CON             ; TIMER0 INCREM. PELO CICLO DE MÁQUINA
                          ; TIMER0- 1:1
```

```
MOVLW   B'01100000'     ; CONFIGURA INTERRUPÇÕES
MOVWF   INTCON          ; HABILITADA A INTERRUPÇÃO DE TIMER0
                        ; HABILITA AS INTERRUPÇÕES DE PERIFÉRICO

MOVLW   B'10000100'     ; ALTA PRIORIDADE PARA INT. DO TIMER0
MOVWF   INTCON2         ; PULL-UPs DESABILITADOS

MOVLW   B'10000000'
MOVWF   RCON            ; ATIVA SISTEMA DE PRIORIDADE DAS INTERRUPÇÕES

MOVLW   B'00000001'
MOVWF   IPR1            ; ALTA PRIORIDADE PARA INT. DO TIMER1

MOVLW   B'00000001'     ; CONFIGURA INTERRUPÇÕES DE PERIFIÉRICOS
MOVWF   PIE1            ; HABILITADA A INTERRUPÇÃO DE TMR1

MOVLW   B'00001111'     ; CONFIGURA CONVERSOR AD
MOVWF   ADCON1          ; CONFIGURA PORTA, PORTB E PORTE COMO I/O DIGITAL

MOVLW   B'10110000'     ; CONFIGURA TMR1
MOVWF   T1CON           ; PRESCALER -> 1:8
                        ; INCREMENTADO PELO CICLO DE MÁQUINA

;  AS INSTRUÇÕES A SEGUIR FAZEM COM QUE O PROGRAMA TRAVE QUANDO HOUVER UM
;  RESET OU POWER-UP, MAS PASSE DIRETO SE O RESET FOR POR WDT. DESTA FORMA,
;  SEMPRE QUE O PIC É LIGADO, O PROGRAMA TRAVA, AGUARDA UM ESTOURO DE WDT
;  E COMEÇA NOVAMENTE. ISTO EVITA PROBLEMAS NO START-UP DO PIC.

BTFSC   RCON,NOT_TO     ; RESET POR ESTOURO DE WATCHDOG TIMER?
BRA     $               ; NÃO - AGUARDA ESTOURO DO WDT
                        ; SIM

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                        INICIALIZAÇÃO DA RAM                           *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTA ROTINA IRÁ LIMPAR TODA A RAM, DO BANCO 0 ATÉ AO BANCO 15

LFSR    FSR0,0X00       ; APONTA FSR0 PARA ENDEREÇO ZERO

LIMPA_RAM
CLRF    POSTINC0        ; LIMPA A POSIÇÃO E INCREMENTA FSR0
CLRWDT                  ; LIMPA WATCHDOG TIMER
MOVLW   0X0F            ; TESTA SE JÁ CHEGOU AO BANCO 15
SUBWF   FSR0H,W
BNZ     LIMPA_RAM       ; SE NÃO, CONTINUA LIMPANDO

LFSR    FSR0,0X00       ; APONTA FSR0 PARA ENDEREÇO ZERO

BCF     ESTADO_TIMER    ; INICIA COM ESTADO EM OFF

MOVLW   .2
MOVWF   DIVISOR_TMR1    ; CARREGA CONTADOR DE 1SEG.

MOVLW   .9              ; INICIALIZA TIMER
MOVWF   UNIDADE
MOVWF   DEZENA
MOVWF   CENTENA
MOVWF   MILHAR

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                      VARREDURA DOS BOTÕES                             *
; *                         LOOP PRINCIPAL                                *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

```
;  A ROTINA PRINCIPAL FICA VERIFICANDO O ESTADO DOS BOTÕES. CASO ALGUM SEJA
;  PRESSIONADO, A ROTINA DE TRATAMENTO DO BOTÃO É CHAMADA.

    BSF     INTCON,GIEL      ; LIGA AS INTERRUPÇÕES
    BSF     INTCON,GIEH      ; USADA INT. TMR0 PARA VARREDURA
                             ; DOS DISPLAYS
VARRE
    CLRWDT                   ; LIMPA WATCHDOG TIMER

    BTFSS   BT_UP            ; O BOTÃO DE UP ESTÁ PRESSIONADO?
    BRA     TRATA_BT_UP      ; SIM - PULA P/ TRATA_BT_UP
                             ; NÃO
    BTFSS   BT_DOWN          ; O BOTÃO DE DOWN ESTÁ PRESSIONADO?
    BRA     TRATA_BT_DOWN    ; SIM - PULA P/ TRATA_BT_DOWN
                             ; NÃO
    BTFSS   BT_START_STOP        ; O BOTÃO START/STOP ESTÁ PRESSIONADO?
    BRA     TRATA_BT_START_STOP ; SIM - PULA P/ TRATA_BT_START_STOP
                                 ; NÃO
    MOVLW   FILTRO_TECLA     ; CARREGA NO WORK O VALOR DE FILTRO_TECLA
    MOVWF   FILTRO_BOTOES    ; SALVA EM FILTRO_BOTOES
                             ; RECARREGA FILTRO P/ EVITAR RUIDOS
    MOVLW   .1
    MOVWF   TEMPO_TURBO      ; CARREGA TEMPO DO TURBO DAS TECLAS
                             ; COM 1 - IGNORA O TURBO A PRIMEIRA
                             ; VEZ QUE A TECLA É PRESSIONADA
    BRA     VARRE            ; VOLTA PARA VARRER TECLADO

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                          TRATAMENTO DOS BOTÕES                          *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

; ************************** TRATAMENTO DO BOTÃO DE UP ************************

TRATA_BT_UP
    BTFSC   ESTADO_TIMER     ; TIMER ESTÁ PARADO ?
    BRA     VARRE            ; NÃO - VOLTA P/ VARRE
                             ; SIM

    DECFSZ  FILTRO_BOTOES,F  ; FIM DO FILTRO ? (RUIDO?)
    BRA     VARRE            ; NÃO - VOLTA P/ VARRE
                             ; SIM - BOTÃO PRESSIONADO

    DECFSZ  TEMPO_TURBO,F    ; FIM DO TEMPO DE TURBO ?
    BRA     VARRE            ; NÃO - VOLTA P/ VARRE
                             ; SIM
    MOVLW   TURBO_TECLA
    MOVWF   TEMPO_TURBO      ; RECARREGA TEMPORIZADOR DO TURBO
                             ; DAS TECLAS

    CALL    INCREMENTA_TIMER; INCREMENTA O VALOR DO TIMER

    BRA     VARRE            ; VOLTA P/ VARREDURA DOS BOTÕES

; ************************** TRATAMENTO DO BOTÃO DE DOWN **********************

TRATA_BT_DOWN
    BTFSC   ESTADO_TIMER     ; TIMER ESTÁ PARADO ?
    BRA     VARRE            ; NÃO - VOLTA P/ VARRE
                             ; SIM

    DECFSZ  FILTRO_BOTOES,F  ; FIM DO FILTRO ? (RUIDO?)
    BRA     VARRE            ; NÃO - VOLTA P/ VARRE
                             ; SIM - BOTÃO PRESSIONADO
```

```
    DECFSZ  TEMPO_TURBO,F   ; FIM DO TEMPO DE TURBO ?
    BRA     VARRE           ; NÃO - VOLTA P/ VARRE
                            ; SIM
    MOVLW   TURBO_TECLA
    MOVWF   TEMPO_TURBO     ; RECARREGA TEMPORIZADOR DO TURBO
                            ; DAS TECLAS

    CALL    DECREMENTA_TIMER; DECREMENTA O VALOR DO TIMER

    BRA     VARRE           ; VOLTA P/ VARREDURA DOS BOTÕES

; ************************** TRATAMENTO DO BOTÃO START / STOP *****************

TRATA_BT_START_STOP
    MOVF    FILTRO_BOTOES,F ; FILTRO JÁ IGUAL A ZERO?(FUNÇÃO JÁ FOI EXECUTADA?)
    BZ      VARRE           ; SIM - VOLTA P/ VARREDURA DO TECLADO
                            ; NÃO

    DECFSZ  FILTRO_BOTOES,F ; FIM DO FILTRO ? (RUIDO?)
    BRA     VARRE           ; NÃO - VOLTA P/ VARRE
                            ; SIM - BOTÃO PRESSIONADO

    BTFSS   ESTADO_TIMER    ; TIMER ESTA LIGADO ?
    BRA     LIGA_TIMER      ; NÃO - ENTÃO LIGA
                            ; SIM - ENTÃO DESLIGA

DESLIGA_TIMER
    BCF     ESTADO_TIMER    ; DESLIGA LED E FLAG DO ESTADO DO TIMER
    BCF     T1CON,TMR1ON    ; PARA CONTADOR DO TMR1
    BRA     VARRE           ; VOLTA P/ VARREDURA DOS BOTÕES

LIGA_TIMER
    MOVF    UNIDADE,F       ; UNIDADE ESTÁ ZERADA ?
    BNZ     LIGA_TIMER_2    ; NÃO - PULA P/ LIGA_TIMER_2
                            ; SIM - TESTA DEZENA

    MOVF    DEZENA,F        ; DEZENA ESTÁ ZERADA ?
    BNZ     LIGA_TIMER_2    ; NÃO - PULA P/ LIGA_TIMER_2
                            ; SIM - TESTA CENTENA

    MOVF    CENTENA,F       ; CENTENA ESTÁ ZERADA ?
    BNZ     LIGA_TIMER_2    ; NÃO - PULA P/ LIGA_TIMER_2
                            ; SIM - TESTA MILHAR

    MOVF    MILHAR,F        ; MILHAR ESTÁ ZERADO ?
    BNZ     LIGA_TIMER_2    ; NÃO - PULA P/ LIGA_TIMER_2
    BRA     VARRE           ; SIM - VOLTA P/ VARRER TECLADO
                            ; SEM LIGAR O TIMER

LIGA_TIMER_2
    BSF     ESTADO_TIMER    ; LIGA LED E FLAG DO ESTADO DO TIMER

    MOVLW   TMR1_HIGH
    MOVWF   TMR1H
    MOVLW   TMR1_LOW
    MOVWF   TMR1L           ; INICIALIZA CONTADORES

    MOVLW   .2
    MOVWF   DIVISOR_TMR1    ; INICIALIZA DIVISOR

    BSF     T1CON,TMR1ON    ; LIGA CONTAGEM DO TMR1

    BRA     VARRE           ; VOLTA P/ VARREDURA DOS BOTÕES
```

```
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                                FIM DO PROGRAMA                            *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

    END                        ; FIM DO PROGRAMA
```

» Dicas e comentários

Observe que neste exemplo estamos utilizando apenas a interrupção de alta prioridade. Neste caso empregamos a shadow para salvar e recuperar o contexto automaticamente, por hardware.

Como a varredura dos displays não precisa ter uma frequência 100% ajustada, ela poderia ser feita sem o emprego de interrupção, isto é, através do loop principal, como foi o caso da verificação dos botões. O único cuidado a ser tomado neste caso é que determinadas ações podem atrapalhar o tempo desse loop, interferindo na frequência de varredura. Por isso, é muito comum encontrarmos certos projetos em que o display cintila quando pressionamos uma tecla.

» Exercícios propostos

Agora que o exemplo foi estudado e esclarecido, aproveite para gerar novos problemas e soluções, seguindo os exercícios propostos:

1. Implementar o quarto botão, para resetar o temporizador (voltar a zero). Esse botão também só deve funcionar quando o temporizador estiver parado.
2. Implementar uma segunda velocidade para os botões de incremento e decremento, de forma que facilite o ajuste de valores maiores.
3. Em vez de fazer um timer somente de segundos, utilizar os dois dígitos da esquerda para mostrar o tempo em minutos e os da direita para mostrar o tempo em segundos. O ponto do display DS2 pode ser usado para marcar a separação. Não esquecer que agora os displays da direita devem contar somente de 0 a 59 e não mais de 0 a 99.

6

Operação com Display de Cristal Líquido (LCD)

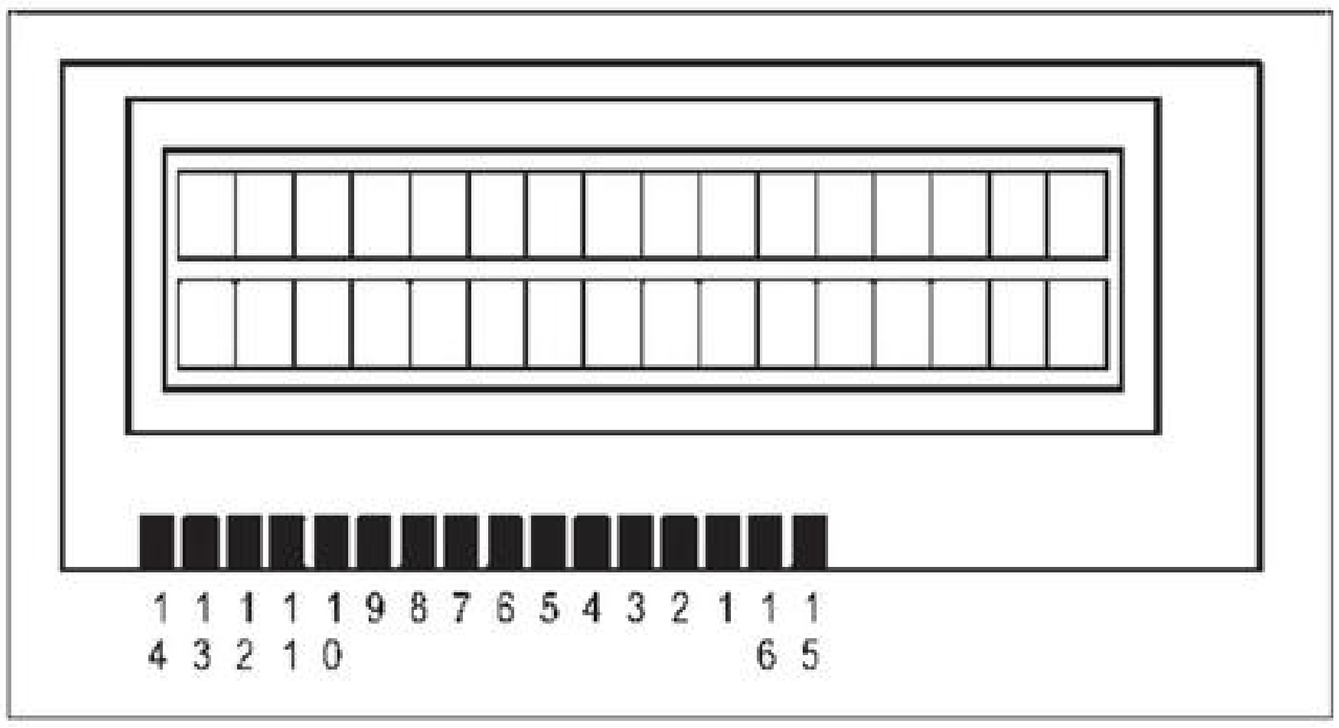
» Introdução

Dando continuidade às melhorias no sistema de interface com o usuário, veremos agora a operação com display de cristal líquido. Esses displays são muito mais poderosos que os displays de segmentos, pois possuem muito mais caracteres e são alfanuméricos. Entretanto, são também mais caros e com uma visualização inferior, pois os caracteres não possuem iluminação própria e são de tamanho bem reduzido.

» Teoria e recursos do PIC

Existem muitos displays de cristal líquido (LCD) no mercado, mas trabalharemos com o modelo padrão de todos. Trata-se de um display de duas linhas com 16 caracteres cada uma, bem conhecido por ser o modelo utilizado nos telefones públicos (orelhões). Outra característica importante do display que vamos estudar é que ele já possui um drive de controle interno. Desta forma, nos comunicaremos com ele através de uma comunicação paralela, passando comandos e os caracteres que desejamos escrever diretamente em código ASCII.

Vejamos agora o formato do LCD que vamos estudar:



Este LCD possui 16 pinos para sua ligação ao projeto. A tabela seguinte identifica cada um desses pinos:

Pino	Função
1	V_{SS}
2	V_{DD}
3	Vo
4	RS
5	R/W
6	E
7	DB0
8	DB1

Pino	Função
9	DB2
10	DB3
11	DB4
12	DB5
13	DB6
14	DB7
15	A
16	K

Os dois primeiros pinos (V_{SS} e V_{DD}) são relativos à alimentação do componente e devem ser ligados a uma tensão nominal de 5 V_{DC}. As tensões mínima e máxima para a alimentação podem variar conforme o fabricante, mas na maioria dos casos ficam entre 4,75 e 5,25 V_{DC}.

O pino **Vo** é utilizado para controle do contraste e normalmente o ligamos ao centro de um potenciômetro de 10 k com as extremidades ligadas ao V_{SS} e ao V_{DD}. Na verdade, esse pino deve possuir uma tensão variável ou fixa entre V_{SS} e V_{DD}.

O pino **RS** (Register Select) é utilizado para definirmos o tipo de informação passada através da comunicação paralela:

RS	Descrição
0	A informação é um comando ou instrução.
1	A informação é um dado.

O pino **R/W** muda o estado do LCD entre Leitura (Read) e Escrita (Write). Essa mudança pode ser feita para escrevermos um comando ou dado e verificarmos quando o LCD terminou a operação e está pronto para darmos continuidade ao processo.

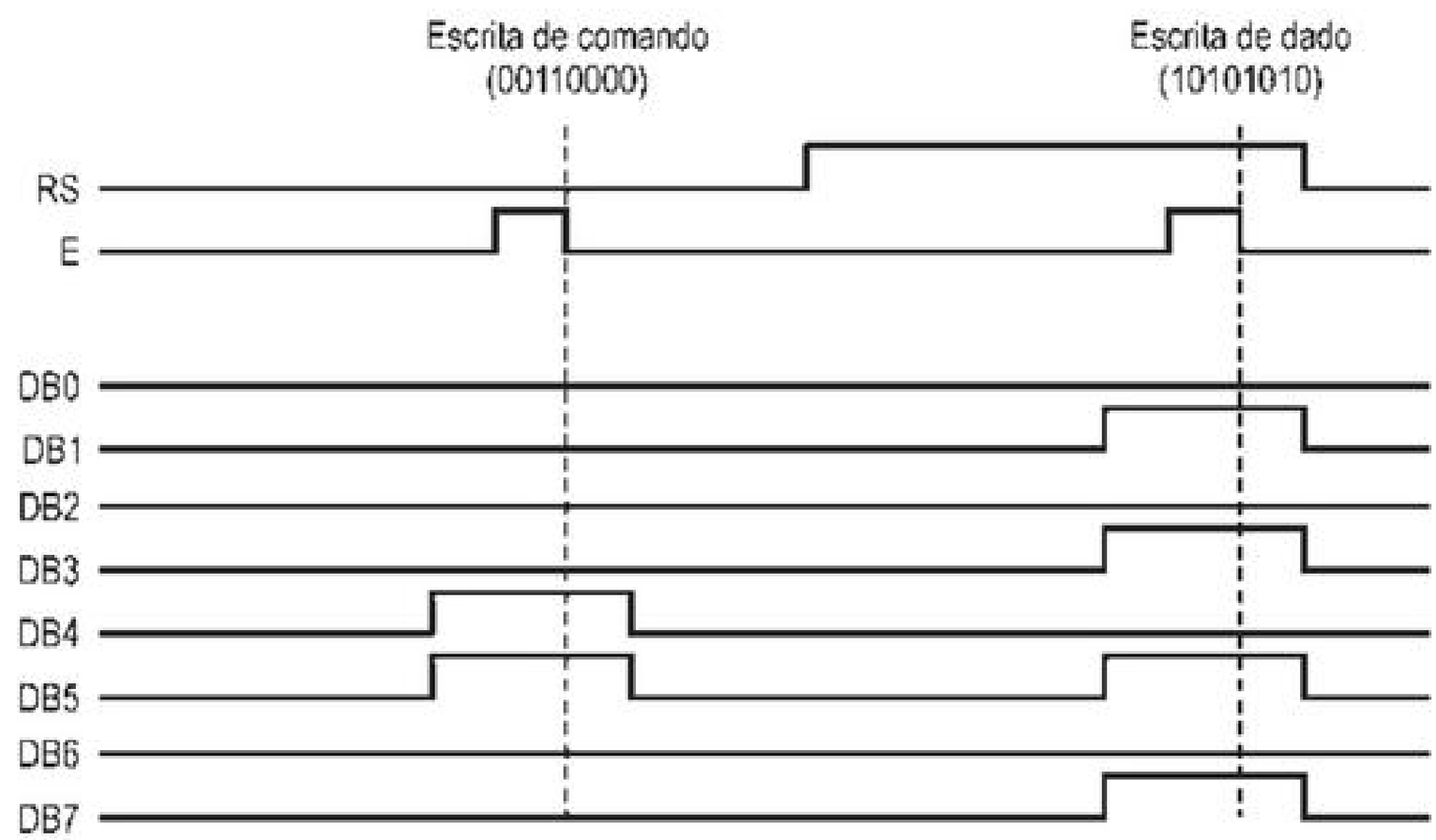
R/W	Descrição
0	Operação de escrita.
1	Operação de leitura.

Em muitos projetos, como no caso da placa proposta (McLab2), esse pino não é utilizado, ficando permanentemente ligado ao V_{SS}. Neste caso, o LCD só opera em modo de escrita. Por isso, precisamos garantir o término das operações internas do módulo de LCD através de tempos preestabelecidos, que são descritos adiante.

O pino **E** (Enable) é utilizado para efetivar a leitura da informação escrita no barramento de dados. Essa leitura é efetuada na borda de descida desse sinal.

Os pinos de **DB0** a **DB7** equivalem ao barramento de dados paralelo. Apesar de existirem oito vias de dados, esses displays também podem operar com quatro vias (**DB4** a **DB7**), já que as demais vias ficam sem funções. Neste caso, as informações são enviadas em dois pacotes de 4 bits cada um. Os pinos **A** (Anode) e **K** (Katode) são usados para ligação do Backlight (iluminação de fundo). O fato é que, apesar da existência dos pinos, nem todos os displays possuem essa iluminação.

Para nós, então, vale a comunicação em oito vias de dados, acrescida dos controles **RS** e **E**, dando um total de dez pinos interligando o PIC ao LCD. Desta forma, para enviarmos uma informação ao LCD precisamos primeiramente ajustar **RS** para informarmos se é um comando ou um dado. Em seguida, devemos escrever a informação no barramento de dados. O próximo passo é darmos um pulso em **E**.



Inicialização do LCD

A primeira ação a ser efetuada no display é sua inicialização, garantindo a comunicação em oito vias. Devemos criar uma rotina baseada no seguinte roteiro:

1. Aguarde pelo menos 15 ms após a energização do LCD para garantir que ele já está operando corretamente.
2. Envie o comando 30h para o display.

RS	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	1	1	0	0	0	0

3. Aguarde pelo menos 4 ms para garantir o término da operação.
4. Envie novamente o comando 30h para o display.

RS	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	1	1	0	0	0	0

5. Aguarde pelo menos 100 µs.
6. Envie novamente o comando 30h para o display.

RS	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	1	1	0	0	0	0

7. Aguarde pelo menos 40 µs.
8. Estabeleça as condições de utilização. Neste caso, comunicação em oito vias, display de duas linhas e matriz de 7x5.

RS	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	1	1	1	0	0	0

9. Aguarde pelo menos 40 µs.

10. Comando para limpar o display e posicionar o cursor na primeira linha, primeira coluna (esquerda).

RS	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	0	0	1

11. Aguarde pelo menos 1,8 ms.

12. Envie o comando para ligar o display sem cursor.

RS	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	1	1	0	0

13. Aguarde pelo menos 40 µs.

14. Envie o comando para estabelecer o modo de operação. Por exemplo, deslocamento automático do cursor para a direita.

RS	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	1	1	0

15. Aguarde pelo menos 40 µs.

Para o caso de utilização com quatro vias, o roteiro é ligeiramente diferente.

1. Aguarde pelo menos 15 ms após a energização do LCD para garantir que ele já esteja operando corretamente.

2. Envie o comando 30h para o display.

RS	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	1	1	-	-	-	-

3. Aguarde pelo menos 4 ms para garantir o término da operação.

4. Envie novamente o comando 30h para o display.

RS	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	1	1	-	-	-	-

5. Aguarde pelo menos 100 µs.

6. Envie novamente o comando 30h para o display.

RS	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	1	1	-	-	-	-

7. Aguarde pelo menos 40 µs.

8. Estabeleça a comunicação em quatro vias.

RS	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	1	0	-	-	-	-

9. Aguarde pelo menos 40 µs.

10. Estabeleça as condições de utilização. Neste caso, comunicação em quatro vias, display de duas linhas e matriz de 7x5.

RS	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	1	0	-	-	-	-
0	1	0	0	0	-	-	-	-

11. Aguarde pelo menos 40 µs.

12. Envie o comando para limpar o display e posicionar o cursor na primeira linha, primeira coluna (esquerda).

RS	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	-	-	-	-
0	0	0	0	1	-	-	-	-

13. Aguarde pelo menos 1,8 ms.

14. Envie o comando para ligar o display sem cursor.

RS	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	-	-	-	-
0	1	1	0	0	-	-	-	-

15. Aguarde pelo menos 40 µs.

16. Envie o comando para estabelecer o modo de operação. Por exemplo, deslocamento automático do cursor para a direita.

RS	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	-	-	-	-
0	0	1	1	0	-	-	-	-

17. Aguarde pelo menos 40 µs.

Comandos do LCD

Vejamos agora quais são realmente os comandos existentes no display. As nomenclaturas utilizadas respeitam a maioria dos data sheets existentes para esse tipo de componente. Para informações mais detalhadas, recomendamos a consulta direta ao manual do modelo em uso no seu projeto. Os tempos apresentados também podem variar de um modelo para outro.

Limpeza do display

RS	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	0	0	1

Esse comando apaga todo o display, escrevendo um espaço em branco (ASCII 20h) em todas as 32 posições disponíveis na memória interna do display (DDRAM). Depois disso, o cursor é posicionado no primeiro caractere (lado esquerdo) da primeira linha. O comando dura cerca de 1,64 ms.

Retorno do cursor

RS	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	0	1	(1/0)

Esse comando faz com que o cursor retorne à posição inicial (primeira linha/primeira coluna) sem afetar a memória do display. Ele dura cerca de 1,64 ms.

Modo de operação

RS	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	1	I/D	S

Esse comando possui dois bits de configuração que podem ser ajustados para deslocamento automático do cursor e da mensagem:

Bit	Descrição
I/D = 0	O cursor desloca-se automaticamente para a esquerda (decrementado) após uma operação de escrita ou leitura.
I/D = 1	O cursor desloca-se automaticamente para a direita (incrementado) após uma operação de escrita ou leitura.
S = 0	Desliga o deslocamento da mensagem.
S = 1	Liga o deslocamento da mensagem. A mensagem desloca-se para a direita ou para a esquerda, conforme o valor do bit I/D. Serve para implementar as funções Insert e Backspace.

Esse comando dura cerca de 40 µs.

Controle do display e cursor

RS	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	1	D	C	B

Esse comando possui 3 bits para a configuração da visualização dos caracteres e do cursor:

Bit	Descrição
D = 0	Inibe a visualização dos caracteres no display. A memória interna do LCD permanece inalterada.
D = 1	Habilita a visualização dos caracteres no display, conforme os dados existentes na memória interna.
C = 0	O cursor não é visível.
C = 1	O cursor é visível como uma linha embaixo do caractere (oitava linha da matriz), como se fosse um sublinhado.
B = 0	Desativa o cursor piscante.
B = 1	Ativa o cursor piscante. Na verdade, nesse modo o caractere da posição atual é alternado com um bloco negro em intervalos de 0,4 s. Pode ser combinado com o cursor do tipo sublinhado quando C=1.

Esse comando dura cerca de 40 µs.

Deslocamento do cursor ou da mensagem

RS	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	1	S/C	R/L	-	-

Esse comando possui dois bits de configuração para que o cursor ou a mensagem sejam deslocados para a direita ou para a esquerda, sem a necessidade de uma operação de escrita ou de leitura:

S/C	R/L	Descrição
0	0	O cursor desloca-se para a esquerda. Decrementa o endereço da memória interna DDRAM.
0	1	O cursor desloca-se para a direita. Incrementa o endereço da memória interna DDRAM.
1	0	Desloca a mensagem e o cursor para a esquerda, em relação à posição atual do cursor. Função Insert.
1	1	Desloca a mensagem e o cursor para a direita, em relação à posição atual do cursor. Função Backspace.

Esse comando dura cerca de 40 µs.

Configuração para utilização

RS	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	1	DL	N	F	-	-

Esse comando possui um bit para configuração da quantidade de vias de comunicação:

DL	Descrição
1	Comunicação feita pelas oito vias de dados, de DB0 a DB7.
0	Comunicação feita em quatro vias de dados, de DB4 a DB7. Inicialmente deve ser enviada a parte alta do byte e, em seguida, a parte baixa.

Existem ainda mais dois bits de configuração para definir a quantidade de linhas e o tamanho da matriz do caractere:

N	F	Descrição
0	0	Uma linha com matriz de 7x5 + cursor
0	1	Uma linha com matriz de 10x5 + cursor
1	-	Duas linhas com matriz de 7x5 + cursor (McLab2)

Esse comando dura cerca de 40 µs.

Posicionamento do cursor (DDRAM)

RS	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	1	A	A	A	A	A	A	A

Para escrever um caractere em qualquer local do display, basta posicionar corretamente o cursor antes da operação de escrita do dado. Para isso, cada posição possui um endereço representado por AAAAAAA. Para ficar ainda mais fácil, vamos montar uma tabela com os 32 caracteres do display e seus endereços absolutos (em hexadecimal), já considerando DB7=1:

Coluna	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Linha 0	80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
Linha 1	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF

Desta forma, fica fácil posicionar o cursor em qualquer local. Basta enviar ao LCD o comando relativo à posição desejada.

Esse comando dura cerca de 40 µs.

Escrita de um caractere

RS	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
1	D	D	D	D	D	D	D	D

Posicionado o cursor corretamente, basta enviar o caractere que deseja escrever. Para isso devemos manter **RS=1** e colocar o código ASCII do caractere desejado nas vias de dados. Como no caso dos comandos, um pulso em **E** também deve ser executado. Esse comando dura cerca de 40 µs.

A tabela ASCII interna do display, gravada em ROM, pode alterar de um modelo para outro. Por isso, fornecemos somente a tabela ASCII não estendida (primeiros 127 caracteres) que permanece inalterada para a maioria dos modelos disponíveis. Quanto à parte alta, os displays mais comuns encontrados no Brasil possuem os caracteres na língua japonesa nessas posições.

Observe que as primeiras posições são definidas como oito endereços da memória CGRAM duplicados. Esses endereços acessam os caracteres especiais que podem ser desenhados na RAM dos módulo de LCD. Para definição e uso desses caracteres especiais, consulte o manual do LCD.

Valores em hexadecimal		Dígito mais significativo (primeiro)							
		0	1	2	3	4	5	6	7
Dígito menos significativo (segundo)	0	CGRAM (1)			0	@	P	`	p
	1	CGRAM (2)		!	1	A	Q	a	q
	2	CGRAM (3)		"	2	B	R	b	r
	3	CGRAM (4)		#	3	C	S	c	s
	4	CGRAM (5)		$	4	D	T	d	t
	5	CGRAM (6)		%	5	E	U	e	u
	6	CGRAM (7)		&	6	F	V	f	v
	7	CGRAM (8)		'	7	G	W	g	w
	8	CGRAM (1)		(	8	H	X	h	x
	9	CGRAM (2)		)	9	I	Y	i	y
	A	CGRAM (3)		*	:	J	Z	j	z
	B	CGRAM (4)		+	;	K	[	k	{
	C	CGRAM (5)		,	<	L	¥	l	|
	D	CGRAM (6)		-	=	M	]	m	}
	E	CGRAM (7)		.	>	N	^	n	→
	F	CGRAM (8)		/	?	O	_	o	←

» Lógica do exemplo

O exemplo para demonstração do LCD é bem reduzido. Simplesmente utilizamos esse display para informar ao usuário qual botão foi pressionado. Para isso elaboramos uma rotina chamada ESCREVE que envia a informação passada pelo W para o display. Para evitarmos problemas de temporização, essa rotina já garante um tempo de espera de 1 ms. Para os comandos que exigem tempo maior, um delay adicional será dado após a chamada dessa rotina.

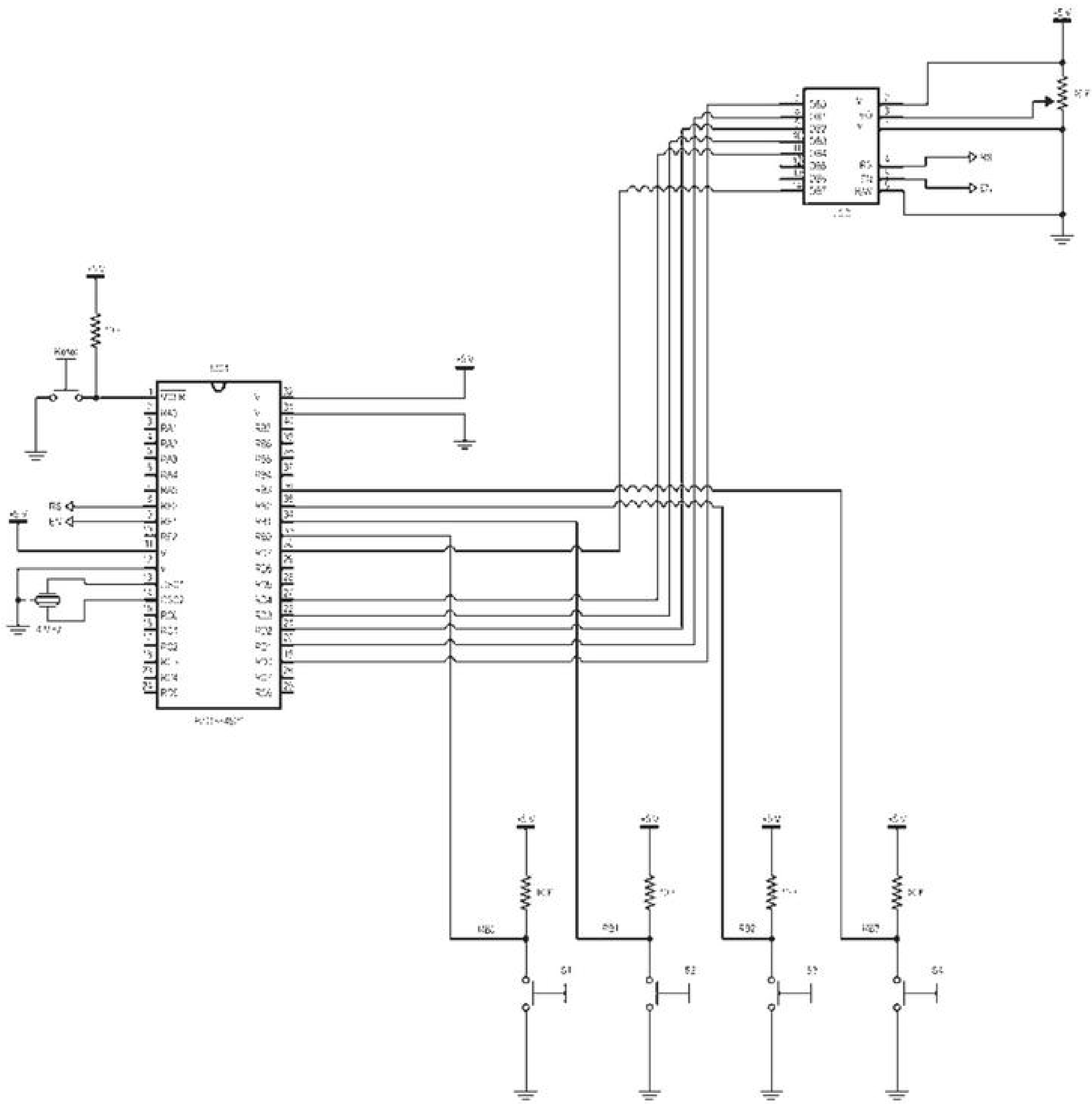
A rotina ESCREVE pode ser usada tanto para enviar comandos quanto dados para o display. Acontece que, como essa rotina não verifica ou altera o valor de RS, essa saída deve ser configurada antes de a rotina ser chamada.

Seguindo o roteiro descrito na parte teórica deste capítulo, implementamos também uma rotina de preparação do LCD, chamada INICIALIZACAO_DISPLAY. Essa rotina configura o sistema para comunicação com oito vias, duas linhas, sem cursor visível e com movimento automático do cursor para a direita. Além disso, ela já limpa a tela e posiciona o cursor na primeira linha, primeiro caractere da esquerda.

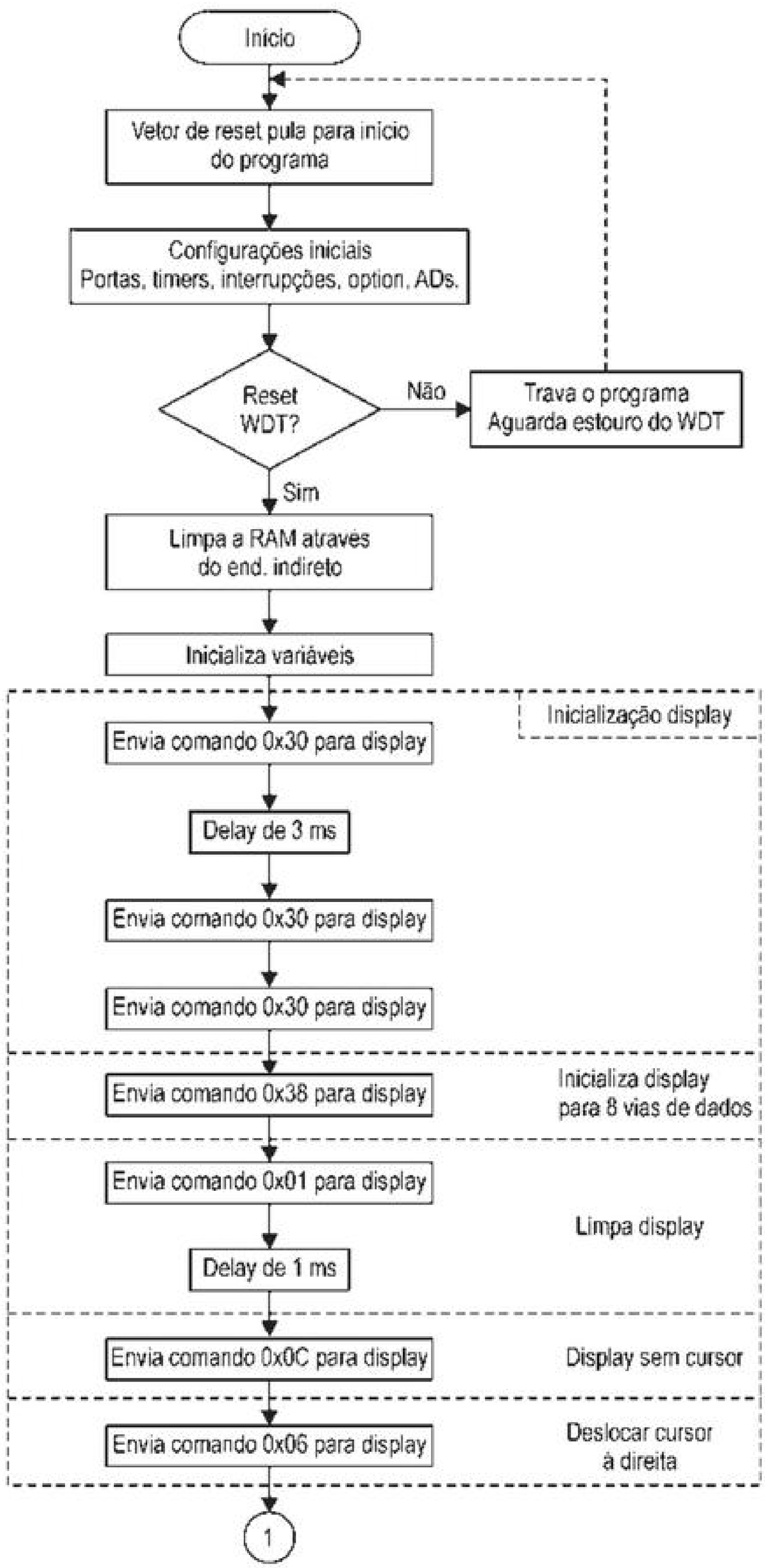
Para cada botão pressionado posicionamos o cursor em um local diferente da tela e escrevemos o referido número do botão. Após a liberação, uma tela padrão é utilizada.

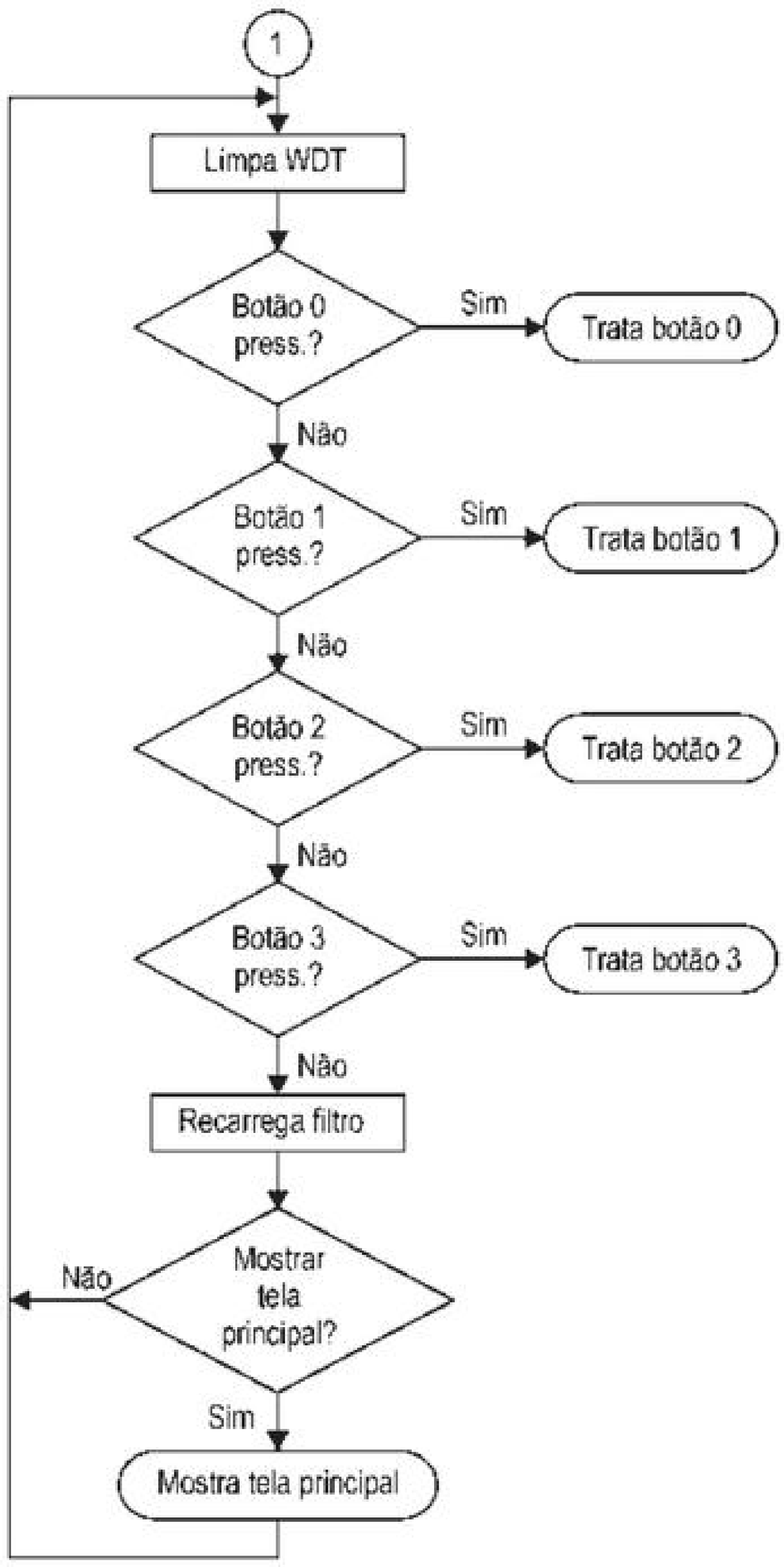
Este exemplo não utiliza nenhuma interrupção.

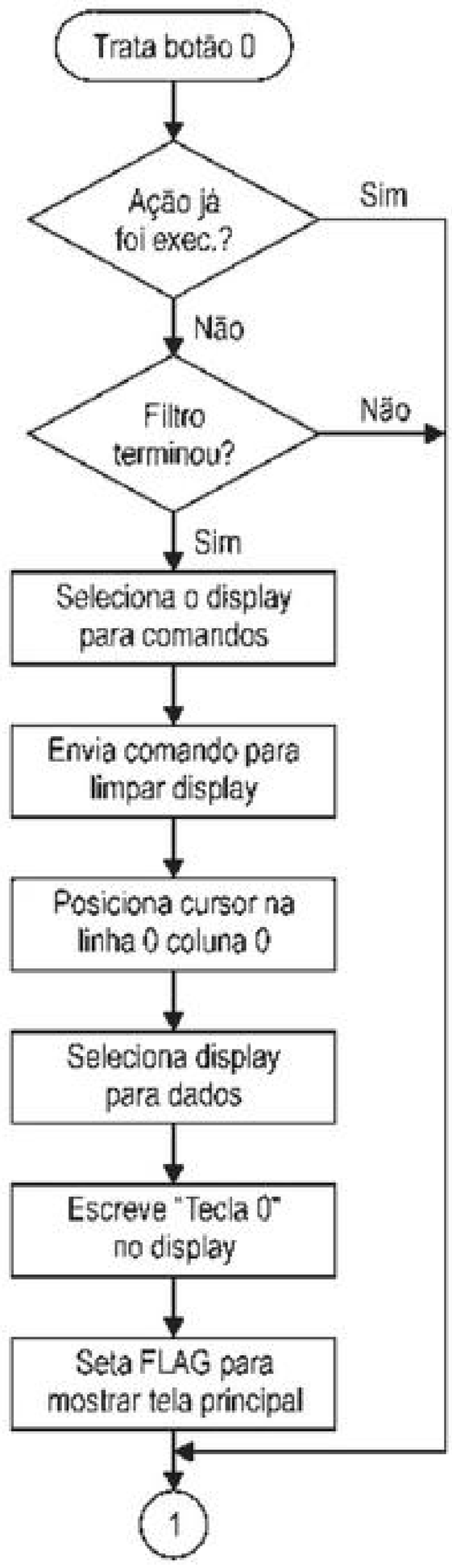
» Esquema elétrico

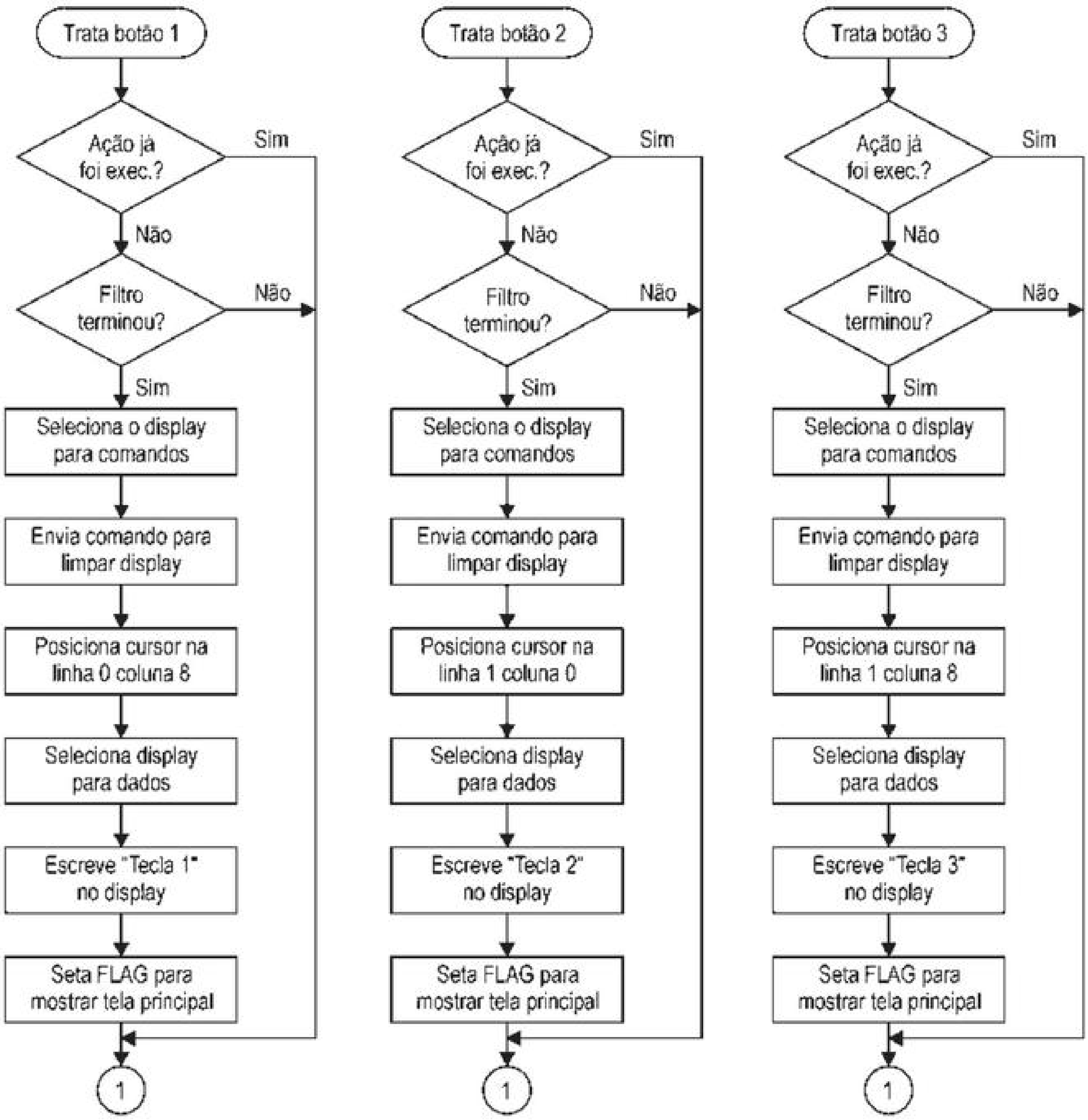


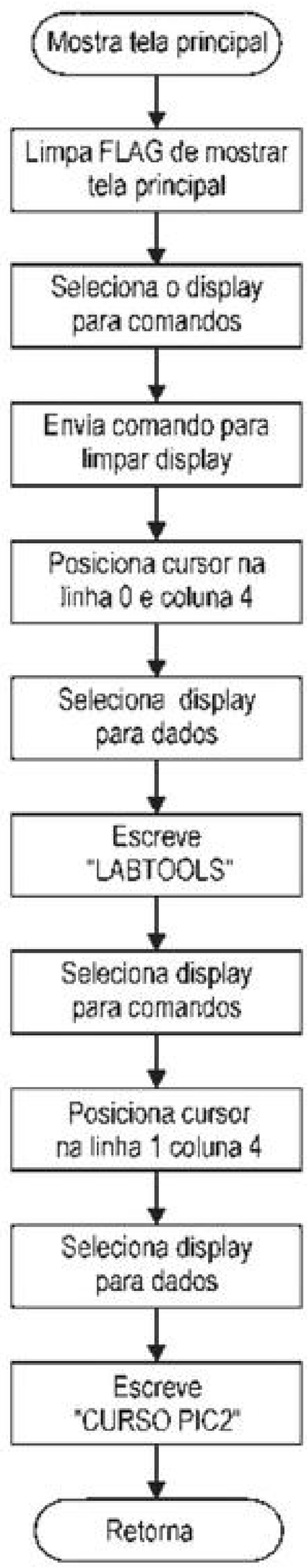
» Fluxograma

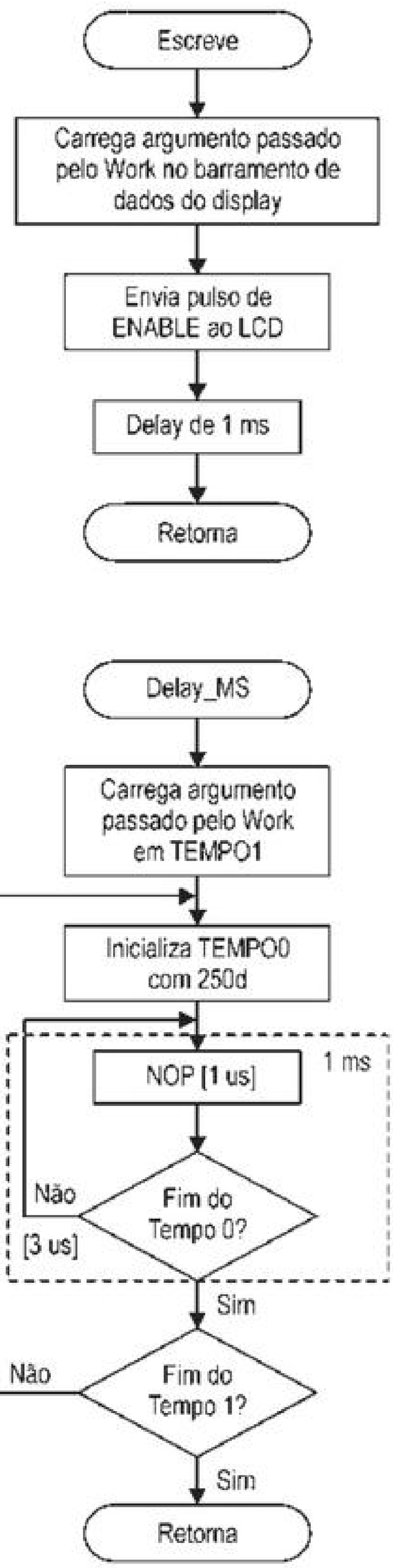

Início
Vetor de reset pula para início do programa
Configurações iniciais
Portas, timers, interrupções, option, ADs.
Reset WDT?
Não
Trava o programa
Aguarda estouro do WDT
Sim
Limpa a RAM através do end. indireto
Inicializa variáveis
Inicialização display
Envia comando 0x30 para display
Delay de 3 ms
Envia comando 0x30 para display
Envia comando 0x30 para display
Inicializa display para 8 vias de dados
Envia comando 0x38 para display
Envia comando 0x01 para display
Limpa display
Delay de 1 ms
Envia comando 0x0C para display
Display sem cursor
Envia comando 0x06 para display
Deslocar cursor à direita
1


1
Limpa WDT
Botão 0 press.?
Sim
Trata botão 0
Não
Botão 1 press.?
Sim
Trata botão 1
Não
Botão 2 press.?
Sim
Trata botão 2
Não
Botão 3 press.?
Sim
Trata botão 3
Não
Recarrega filtro
Mostrar tela principal?
Não
Sim
Mostra tela principal


Trata botão 0
Ação já foi exec.?
Sim
Não
Filtro terminou?
Não
Sim
Seleciona o display para comandos
Envia comando para limpar display
Posiciona cursor na linha 0 coluna 0
Seleciona display para dados
Escreve "Tecla 0" no display
Seta FLAG para mostrar tela principal
1

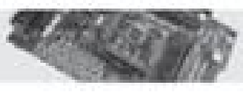

Trata botão 1
Ação já foi exec.?
Sim
Não
Filtro terminou?
Não
Sim
Seleciona o display para comandos
Envia comando para limpar display
Posiciona cursor na linha 0 coluna 8
Seleciona display para dados
Escreve "Tecla 1" no display
Seta FLAG para mostrar tela principal
1
Trata botão 2
Ação já foi exec.?
Sim
Não
Filtro terminou?
Não
Sim
Seleciona o display para comandos
Envia comando para limpar display
Posiciona cursor na linha 1 coluna 0
Seleciona display para dados
Escreve "Tecla 2" no display
Seta FLAG para mostrar tela principal
1
Trata botão 3
Ação já foi exec.?
Sim
Não
Filtro terminou?
Não
Sim
Seleciona o display para comandos
Envia comando para limpar display
Posiciona cursor na linha 1 coluna 8
Seleciona display para dados
Escreve "Tecla 3" no display
Seta FLAG para mostrar tela principal
1


Mostra tela principal
Limpa FLAG de mostrar tela principal
Seleciona o display para comandos
Envia comando para limpar display
Posiciona cursor na linha 0 e coluna 4
Seleciona display para dados
Escreve "LABTOOLS"
Seleciona display para comandos
Posiciona cursor na linha 1 coluna 4
Seleciona display para dados
Escreve "CURSO PIC2"
Retorna


Escreve
Carrega argumento passado pelo Work no barramento de dados do display
Envia pulso de ENABLE ao LCD
Delay de 1 ms
Retorna
Delay_MS
Carrega argumento passado pelo Work em TEMPO1
Inicializa TEMPO0 com 250d
NOP [1 us]
1 ms
Não
Fim do Tempo 0?
[3 us]
Sim
Não
Fim do Tempo 1?
Sim
Retorna

» Código

```
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *              DESBRAVANDO O PIC18 - RECURSOS AVANÇADOS                   *
; *                              EXEMPLO 3                                  *
; *                                                                         *
; * DANIEL RODRIGUES DE SOUSA, DAVID JOSÉ DE SOUZA e  NICOLÁS CÉSAR LAVINIA *
; *                                                                         *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *     DATA : 03/08/2009                                                   *
; *   VERSÃO : 1.0                                                          *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                            DESCRIÇÃO GERAL                              *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTE EXEMPLO FOI ELABORADO PARA EXPLICAR O FUNCIONAMENTO DO MÓDULO DE LCD.
;  FOI CRIADA UMA ROTINA PARA ESCREVER COMANDOS OU CARACTERES NO LCD. EXISTE
;  TAMBÉM UMA ROTINA DE INICIALIZAÇÃO NECESSÁRIA PARA A CORRETA CONFIGURAÇÃO
;  DO LCD. OS BOTÕES CONTINUAM SENDO MONITORADOS. UMA MENSAGEM É ESCRITA
;  NO LCD PARA CADA UM DOS BOTÕES, QUANDO ELE É PRESSIONADO.
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                        CONFIGURAÇÕES PARA GRAVAÇÃO                      *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
    CONFIG  OSC = XT, FCMEN = OFF, IESO = OFF, PWRT = ON, BOREN = ON, BORV = 0
    CONFIG  WDT = ON, WDTPS = 128, MCLRE = ON, LPT1OSC = OFF, PBADEN = OFF
    CONFIG  CCP2MX = PORTC, STVREN = ON, LVP = OFF, DEBUG = OFF, XINST = OFF
    CONFIG  CP0 = OFF, CP1 = OFF, CP2 = OFF, CP3 = OFF, CPB = OFF, CPD = OFF
    CONFIG  WRT0 = OFF, WRT1 = OFF, WRT2 = OFF, WRT3 = OFF, WRTB = OFF
    CONFIG  WRTC = OFF, WRTD = OFF, EBTR0 = OFF, EBTR1 = OFF, EBTR2 = OFF
    CONFIG  EBTR3 = OFF, EBTRB = OFF

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                          DEFINIÇÃO DAS VARIÁVEIS                        *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
    CBLOCK  0X00            ; BANK 0 --> 0X00..0X5F --> ACCESS BANK
        TEMPO1
        TEMPO0              ; CONTADORES P/ DELAY

        FILTRO_BOTOES       ; FILTRO PARA RUIDOS

        FLAG                ; FLAG DE USO GERAL
    ENDC

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *               DEFINIÇÃO DAS VARIÁVEIS INTERNAS DO PIC                   *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  O ARQUIVO DE DEFINIÇÕES DO PIC UTILIZADO DEVE SER REFERENCIADO PARA QUE
;  OS NOMES DEFINIDOS PELA MICROCHIP POSSAM SER UTILIZADOS, SEM A NECESSIDADE
;  DE REDIGITAÇÃO.

    #INCLUDE <P18F4520.INC> ; MICROCONTROLADOR UTILIZADO

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                           CONSTANTES INTERNAS                           *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  A DEFINIÇÃO DE CONSTANTES FACILITA A PROGRAMAÇÃO E A MANUTENÇÃO.

FILTRO_TECLA EQU .200       ; FILTRO P/ EVITAR RUIDOS DOS BOTÕES
```

```
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                    DECLARAÇÃO DOS FLAGs DE SOFTWARE                     *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  A DEFINIÇÃO DE FLAGs AJUDA NA PROGRAMAÇÃO E ECONOMIZA MEMÓRIA RAM.

#DEFINE  TELA_PRINCIPAL   FLAG,0     ; FLAG P/ INDICAR QUE DEVE MOSTRAR
                                     ; A TELA PRINCIPAL
                                     ; 1-> MOSTRA TELA PRINCIPAL
                                     ; 0-> TELA PRINCIPAL JÁ FOI MOSTRADA

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                                 ENTRADAS                               *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  AS ENTRADAS DEVEM SER ASSOCIADAS A NOMES PARA FACILITAR A PROGRAMAÇÃO E
;  FUTURAS ALTERAÇÕES DO HARDWARE.

#DEFINE BOTAO_0   PORTB,0            ; ESTADO DO BOTÃO 0
                                     ; 1 -> LIBERADO
                                     ; 0 -> PRESSIONADO

#DEFINE BOTAO_1   PORTB,1            ; ESTADO DO BOTÃO 1
                                     ; 1 -> LIBERADO
                                     ; 0 -> PRESSIONADO

#DEFINE BOTAO_2   PORTB,2            ; ESTADO DO BOTÃO 2
                                     ; 1 -> LIBERADO
                                     ; 0 -> PRESSIONADO

#DEFINE BOTAO_3   PORTB,3            ; ESTADO DO BOTÃO 3
                                     ; 1 -> LIBERADO
                                     ; 0 -> PRESSIONADO

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                                 SAÍDAS                                 *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  AS SAÍDAS DEVEM SER ASSOCIADAS A NOMES PARA FACILITAR A PROGRAMAÇÃO E
;  FUTURAS ALTERAÇÕES DO HARDWARE.

#DEFINE DISPLAY     LATD             ; BARRAMENTO DE DADOS DO DISPLAY

#DEFINE RS          LATE,0           ; INDICA P/ O DISPLAY UM DADO OU COMANDO
                                     ; 1 -> DADO
                                     ; 0 -> COMANDO

#DEFINE ENABLE      LATE,1           ; SINAL DE ENABLE P/ DISPLAY
                                     ; ATIVO NA BORDA DE DESCIDA

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                   VETOR DE RESET DO MICROCONTROLADOR                   *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  POSIÇÃO INICIAL PARA EXECUÇÃO DO PROGRAMA

    ORG     0X0000                   ; ENDEREÇO DO VETOR DE RESET
    BRA     CONFIGURACAO             ; PULA PARA CONFIG DEVIDO A REGIÃO
                                     ; DESTINADA AS ROTINAS SEGUINTES

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                     ROTINA DE DELAY (DE 1MS ATÉ 256MS)                 *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTA É UMA ROTINA DE DELAY VARIÁVEL, COM DURAÇÃO DE 1MS X O VALOR PASSADO
;  EM WORK (W).

DELAY_MS
```

```
    MOVWF   TEMPO1          ; CARREGA TEMPO1 (UNIDADES DE MS)
    MOVLW   .250
    MOVWF   TEMPO0          ; CARREGA TEMPO0 (P/ CONTAR 1MS)

    CLRWDT                  ; LIMPA WDT (PERDE TEMPO)
    DECFSZ  TEMPO0,F        ; FIM DE TEMPO0 ?
    BRA     $-.4            ; NÃO - VOLTA 2 INSTRUÇÕES
                            ; SIM - PASSOU-SE 1ms
    DECFSZ  TEMPO1,F        ; FIM DE TEMPO1 ?
    BRA     $-.12           ; NÃO - VOLTA 6 INSTRUÇÕES
    RETURN                  ; SIM - RETORNA

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                ROTINA DE ESCRITA DE UM CARACTERE NO DISPLAY             *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTA ROTINA ENVIA UM CARACTERE PARA O MÓDULO DE LCD. O CARACTERE A SER
;  ESCRITO DEVE SER COLOCADO EM WORK (W) ANTES DE CHAMAR A ROTINA.

ESCREVE
    MOVWF   DISPLAY         ; ATUALIZA DISPLAY (PORTD)
    NOP                     ; PERDE 1US PARA ESTABILIZAÇÃO
    BSF     ENABLE          ; ENVIA UM PULSO DE ENABLE AO DISPLAY
    BRA     $+.2            ; .
    BCF     ENABLE          ; .

    MOVLW   .1
    RCALL   DELAY_MS        ; DELAY DE 1ms
    RETURN                  ; RETORNA

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                ROTINA DE ESCRITA DA TELA PRINCIPAL                      *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTA ROTINA ESCREVE A TELA PRINCIPAL DO PROGRAMA, COM AS FRASES:
;  LINHA 1 - " LABTOOLS "
;  LINHA 2 - "CURSO PIC2"

MOSTRA_TELA_PRINCIPAL

    BCF     TELA_PRINCIPAL  ; LIMPA FLAG DE MOSTRAR TELA PRINCIPAL

    BCF     RS              ; SELECIONA O DISPLAY P/ COMANDOS
    MOVLW   0X01            ; ESCREVE COMANDO PARA
    RCALL   ESCREVE         ; LIMPAR A TELA
    MOVLW   .1
    RCALL   DELAY_MS        ; DELAY DE 1MS

    MOVLW   0X84            ; COMANDO PARA POSICIONAR O CURSOR
    RCALL   ESCREVE         ; LINHA 0 / COLUNA 4
    BSF     RS              ; SELECIONA O DISPLAY P/ DADOS

    MOVLW   'L'             ; COMANDOS PARA ESCREVER AS
    RCALL   ESCREVE         ; LETRAS DE "LABTOOLS."
    MOVLW   'A'
    RCALL   ESCREVE
    MOVLW   'B'
    RCALL   ESCREVE
    MOVLW   'T'
    RCALL   ESCREVE
    MOVLW   'O'
    RCALL   ESCREVE
    MOVLW   'O'
    RCALL   ESCREVE
    MOVLW   'L'
    RCALL   ESCREVE
```

```
MOVLW   'S'
RCALL   ESCREVE

BCF     RS              ; SELECIONA O DISPLAY P/ COMANDOS
MOVLW   0XC3            ; COMANDO PARA POSICIONAR O CURSOR
RCALL   ESCREVE         ; LINHA 1 / COLUNA 3
BSF     RS              ; SELECIONA O DISPLAY P/ DADOS

MOVLW   'C'             ; COMANDOS PARA ESCREVER AS
RCALL   ESCREVE         ; LETRAS DE "CURSO PIC2"
MOVLW   'U'
RCALL   ESCREVE
MOVLW   'R'
RCALL   ESCREVE
MOVLW   'S'
RCALL   ESCREVE
MOVLW   'O'
RCALL   ESCREVE
MOVLW   ' '
RCALL   ESCREVE
MOVLW   'P'
RCALL   ESCREVE
MOVLW   'I'
RCALL   ESCREVE
MOVLW   'C'
RCALL   ESCREVE
MOVLW   '2'
RCALL   ESCREVE
RETURN

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                CONFIGURAÇÕES INICIAIS DE HARDWARE E SOFTWARE        *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  NESTA ROTINA SÃO INICIALIZADAS AS PORTAS DE I/O DO MICROCONTROLADOR E AS
;  CONFIGURAÇÕES DOS REGISTRADORES ESPECIAIS (SFR). A ROTINA INICIALIZA A
;  MÁQUINA E AGUARDA O ESTOURO DO WDT.

CONFIGURACAO
    CLRF    PORTA           ; GARANTE AS SAÍDAS EM ZERO
    CLRF    PORTB
    CLRF    PORTC
    CLRF    PORTD
    CLRF    PORTE

    CLRF    LATA
    CLRF    LATB
    CLRF    LATC
    CLRF    LATD
    CLRF    LATE

    MOVLW   B'11011111'
    MOVWF   TRISA           ; CONFIGURA I/O DO PORTA

    MOVLW   B'11111111'
    MOVWF   TRISB           ; CONFIGURA I/O DO PORTB

    MOVLW   B'11111111'
    MOVWF   TRISC           ; CONFIGURA I/O DO PORTC

    MOVLW   B'00000000'
    MOVWF   TRISD           ; CONFIGURA I/O DO PORTD

    MOVLW   B'00000100'
    MOVWF   TRISE           ; CONFIGURA I/O DO PORTE
```

```
    MOVLW   B'11011000'     ; CONFIGURA TIMER0
    MOVWF   T0CON           ; TIMER0 INCREM. PELO CICLO DE MÁQUINA
                            ; TIMER0- 1:1

    MOVLW   B'00000000'     ; CONFIGURA INTERRUPÇÕES
    MOVWF   INTCON          ; DESABILITADAS TODAS AS INTERRUPÇÕES

    MOVLW   B'00001111'     ; CONFIGURA CONVERSOR AD
    MOVWF   ADCON1          ; CONFIGURA PORTA, PORTB E PORTE COMO I/O DIGITAL

;  AS INSTRUÇÕES A SEGUIR FAZEM COM QUE O PROGRAMA TRAVE QUANDO HOUVER UM
;  RESET OU POWER-UP, MAS PASSE DIRETO SE O RESET FOR POR WDT. DESTA FORMA,
;  SEMPRE QUE O PIC É LIGADO, O PROGRAMA TRAVA, AGUARDA UM ESTOURO DE WDT
;  E COMEÇA NOVAMENTE. ISTO EVITA PROBLEMAS NO START-UP DO PIC.

    BTFSC   RCON,NOT_TO     ; RESET POR ESTOURO DE WATCHDOG TIMER?
    BRA     $               ; NÃO - AGUARDA ESTOURO DO WDT
                            ; SIM

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                        INICIALIZAÇÃO DA RAM                             *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTA ROTINA IRÁ LIMPAR TODA A RAM, DO BANCO 0 ATÉ AO BANCO 15

    LFSR    FSR0,0X00       ; APONTA FSR0 PARA ENDEREÇO ZERO

LIMPA_RAM
    CLRF    POSTINC0        ; LIMPA A POSIÇÃO E INCREMENTA FSR0
    CLRWDT                  ; LIMPA WATCHDOG TIMER
    MOVLW   0X0F            ; TESTA SE JÁ CHEGOU AO BANCO 15
    SUBWF   FSR0H,W
    BNZ     LIMPA_RAM       ; SE NÃO, CONTINUA LIMPANDO

    BSF     TELA_PRINCIPAL  ; INICIALIZA MOSTRANDO TELA PRINCIPAL

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                  CONFIGURAÇÕES INICIAIS DO DISPLAY                      *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; ESTA ROTINA INICIALIZA O DISPLAY P/ COMUNICAÇÃO DE 8 VIAS, DISPLAY PARA 2
; LINHAS, CURSOR APAGADO E DESLOCAMENTO DO CURSOR À DIREITA.

INICIALIZACAO_DISPLAY
    BCF     RS              ; SELECIONA O DISPLAY P/ COMANDOS

    MOVLW   0X30            ; ESCREVE COMANDO 0X30 PARA
    RCALL   ESCREVE         ; INICIALIZAÇÃO

    MOVLW   .3
    RCALL   DELAY_MS        ; DELAY DE 3MS (EXIGIDO PELO DISPLAY)

    MOVLW   0X30            ; ESCREVE COMANDO 0X30 PARA
    RCALL   ESCREVE         ; INICIALIZAÇÃO

    MOVLW   0X30            ; ESCREVE COMANDO 0X30 PARA
    RCALL   ESCREVE         ; INICIALIZAÇÃO

    MOVLW   B'00111000'     ; ESCREVE COMANDO PARA
    RCALL   ESCREVE         ; INTERFACE DE 8 VIAS DE DADOS

    MOVLW   B'00000001'     ; ESCREVE COMANDO PARA
    RCALL   ESCREVE         ; LIMPAR TODO O DISPLAY

    MOVLW   .1
    RCALL   DELAY_MS        ; DELAY DE 1MS
```

```
    MOVLW   B'00001100'     ; ESCREVE COMANDO PARA
    RCALL   ESCREVE         ; LIGAR O DISPLAY SEM CURSOR

    MOVLW   B'00000110'     ; ESCREVE COMANDO PARA INCREM.
    RCALL   ESCREVE         ; AUTOMÁTICO À DIREITA

    BSF     RS              ; SELECIONA O DISPLAY P/ DADOS

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                        VARREDURA DOS BOTÕES                          *
; *                           LOOP PRINCIPAL                             *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  A ROTINA PRINCIPAL FICA VERIFICANDO O ESTADO DOS BOTÕES. CASO ALGUM SEJA
;  PRESSIONADO, A ROTINA DE TRATAMENTO DO BOTÃO É CHAMADA.

VARRE
    CLRWDT                  ; LIMPA WATCHDOG TIMER

    BTFSS   BOTAO_0         ; O BOTÃO 0 ESTÁ PRESSIONADO ?
    BRA     TRATA_BOTAO_0   ; SIM - PULA P/ TRATA_BOTAO_0
                            ; NÃO
    BTFSS   BOTAO_1         ; O BOTÃO 1 ESTÁ PRESSIONADO ?
    BRA     TRATA_BOTAO_1   ; SIM - PULA P/ TRATA_BOTAO_1
                            ; NÃO
    BTFSS   BOTAO_2         ; O BOTÃO 2 ESTÁ PRESSIONADO ?
    BRA     TRATA_BOTAO_2   ; SIM - PULA P/ TRATA_BOTAO_2
                            ; NÃO
    BTFSS   BOTAO_3         ; O BOTÃO 3 ESTÁ PRESSIONADO ?
    BRA     TRATA_BOTAO_3   ; SIM - PULA P/ TRATA_BOTAO_3
                            ; NÃO
    MOVLW   FILTRO_TECLA    ; CARREGA NO WORK O VALOR DE FILTRO_TECLA
    MOVWF   FILTRO_BOTOES   ; SALVA EM FILTRO_BOTOES
                            ; RECARREGA FILTRO P/ EVITAR RUIDOS
    BTFSS   TELA_PRINCIPAL  ; DEVE MOSTRAR TELA PRINCIPAL ?
    BRA     VARRE           ; NÃO - VOLTA P/ VARRE
                            ; SIM
    RCALL   MOSTRA_TELA_PRINCIPAL
    BRA     VARRE           ; VOLTA PARA VARRER TECLADO

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                        TRATAMENTO DOS BOTÕES                         *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

; ************************* TRATAMENTO DO BOTÃO 0 ****************************
TRATA_BOTAO_0
    MOVF    FILTRO_BOTOES,F ; FILTRO JÁ IGUAL A ZERO?(FUNÇÃO JÁ FOI EXECUTADA?)
    BZ      VARRE           ; SIM - VOLTA P/ VARREDURA DO TECLADO
                            ; NÃO
    DECFSZ  FILTRO_BOTOES,F ; FIM DO FILTRO ? (RUIDO?)
    BRA     VARRE           ; NÃO - VOLTA P/ VARRE
                            ; SIM - BOTÃO PRESSIONADO

;  OS COMANDOS A SEGUIR SÃO PARA ESCREVER A FRASE RELACIONADA AO BOTÃO 0

    BCF     RS              ; SELECIONA O DISPLAY P/ COMANDOS
    MOVLW   0X01
    RCALL   ESCREVE         ; COMANDO P/ LIMPAR A TELA
    MOVLW   .1
    RCALL   DELAY_MS        ; DELAY DE 1MS

    MOVLW   0X80            ; COMANDO PARA POSICIONAR O CURSOR
    RCALL   ESCREVE         ; LINHA 0 / COLUNA 0
    BSF     RS              ; SELECIONA O DISPLAY P/ DADOS
```

```
MOVLW   'T'                 ; COMANDOS PARA ESCREVER AS
RCALL   ESCREVE             ; LETRAS DE "TECLA 0"
MOVLW   'E'
RCALL   ESCREVE
MOVLW   'C'
RCALL   ESCREVE
MOVLW   'L'
RCALL   ESCREVE
MOVLW   'A'
RCALL   ESCREVE
MOVLW   ' '
RCALL   ESCREVE
MOVLW   '0'
RCALL   ESCREVE

    BSF     TELA_PRINCIPAL  ; SETA FLAG P/ MOSTRAR TELA PRINCIPAL

    BRA     VARRE           ; VOLTA P/ VARREDURA DOS BOTÕES

; ************************** TRATAMENTO DO BOTÃO 1 ****************************
TRATA_BOTAO_1
    MOVF    FILTRO_BOTOES,F ; FILTRO JÁ IGUAL A ZERO?(FUNÇÃO JÁ FOI EXECUTADA?)
    BZ      VARRE           ; SIM - VOLTA P/ VARREDURA DO TECLADO
                            ; NÃO
    DECFSZ  FILTRO_BOTOES,F ; FIM DO FILTRO ? (RUIDO?)
    BRA     VARRE           ; NÃO - VOLTA P/ VARRE
                            ; SIM - BOTÃO PRESSIONADO

;  OS COMANDOS A SEGUIR SÃO PARA ESCREVER A FRASE RELACIONADA AO BOTÃO 1

    BCF     RS              ; SELECIONA O DISPLAY P/ COMANDOS
    MOVLW   0X01
    RCALL   ESCREVE         ; COMANDO P/ LIMPAR A TELA
    MOVLW   .1
    RCALL   DELAY_MS        ; DELAY DE 1MS

    MOVLW   0X88            ; COMANDO PARA POSICIONAR O CURSOR
    RCALL   ESCREVE         ; LINHA 0 / COLUNA 8
    BSF     RS              ; SELECIONA O DISPLAY P/ DADOS

    MOVLW   'T'             ; COMANDOS PARA ESCREVER AS
    RCALL   ESCREVE         ; LETRAS DE "TECLA 1"
    MOVLW   'E'
    RCALL   ESCREVE
    MOVLW   'C'
    RCALL   ESCREVE
    MOVLW   'L'
    RCALL   ESCREVE
    MOVLW   'A'
    RCALL   ESCREVE
    MOVLW   ' '
    RCALL   ESCREVE
    MOVLW   '1'
    RCALL   ESCREVE

    BSF     TELA_PRINCIPAL  ; SETA FLAG P/ MOSTRAR TELA PRINCIPAL

    BRA     VARRE           ; VOLTA P/ VARREDURA DOS BOTÕES

; ************************** TRATAMENTO DO BOTÃO 2 ****************************
TRATA_BOTAO_2
    MOVF    FILTRO_BOTOES,F ; FILTRO JÁ IGUAL A ZERO?(FUNÇÃO JÁ FOI EXECUTADA?)
    BZ      VARRE           ; SIM - VOLTA P/ VARREDURA DO TECLADO
                            ; NÃO
```

```
    DECFSZ  FILTRO_BOTOES,F ; FIM DO FILTRO ? (RUIDO?)
    BRA     VARRE           ; NÃO - VOLTA P/ VARRE
                            ; SIM - BOTÃO PRESSIONADO

;  OS COMANDOS A SEGUIR SÃO PARA ESCREVER A FRASE RELACIONADA AO BOTÃO 2

    BCF     RS              ; SELECIONA O DISPLAY P/ COMANDOS
    MOVLW   0X01
    RCALL   ESCREVE         ; COMANDO P/ LIMPAR A TELA
    MOVLW   .1
    RCALL   DELAY_MS        ; DELAY DE 1MS

    MOVLW   0XC0            ; COMANDO PARA POSICIONAR O CURSOR
    RCALL   ESCREVE         ; LINHA 1 / COLUNA 0
    BSF     RS              ; SELECIONA O DISPLAY P/ DADOS

    MOVLW   'T'             ; COMANDOS PARA ESCREVER AS
    RCALL   ESCREVE         ; LETRAS DE "TECLA 2"
    MOVLW   'E'
    RCALL   ESCREVE
    MOVLW   'C'
    RCALL   ESCREVE
    MOVLW   'L'
    RCALL   ESCREVE
    MOVLW   'A'
    RCALL   ESCREVE
    MOVLW   ' '
    RCALL   ESCREVE
    MOVLW   '2'
    RCALL   ESCREVE

    BSF     TELA_PRINCIPAL  ; SETA FLAG P/ MOSTRAR TELA PRINCIPAL

    BRA     VARRE           ; VOLTA P/ VARREDURA DOS BOTÕES

; ************************* TRATAMENTO DO BOTÃO 3 ****************************
TRATA_BOTAO_3
    MOVF    FILTRO_BOTOES,F ; FILTRO JÁ IGUAL A ZERO?(FUNÇÃO JÁ FOI EXECUTADA?)
    BZ      VARRE           ; SIM - VOLTA P/ VARREDURA DO TECLADO
                            ; NÃO
    DECFSZ  FILTRO_BOTOES,F ; FIM DO FILTRO ? (RUIDO?)
    BRA     VARRE           ; NÃO - VOLTA P/ VARRE
                            ; SIM - BOTÃO PRESSIONADO

;  OS COMANDOS A SEGUIR SÃO PARA ESCREVER A FRASE RELACIONADA AO BOTÃO 3

    BCF     RS              ; SELECIONA O DISPLAY P/ COMANDOS
    MOVLW   0X01
    RCALL   ESCREVE         ; COMANDO P/ LIMPAR A TELA
    MOVLW   .1
    RCALL   DELAY_MS        ; DELAY DE 1MS

    MOVLW   0XC8            ; COMANDO PARA POSICIONAR O CURSOR
    RCALL   ESCREVE         ; LINHA 1 / COLUNA 8
    BSF     RS              ; SELECIONA O DISPLAY P/ DADOS

    MOVLW   'T'             ; COMANDOS PARA ESCREVER AS
    RCALL   ESCREVE         ; LETRAS DE "TECLA 3"
    MOVLW   'E'
    RCALL   ESCREVE
    MOVLW   'C'
    RCALL   ESCREVE
    MOVLW   'L'
```

```
    RCALL   ESCREVE
    MOVLW   'A'
    RCALL   ESCREVE
    MOVLW   ' '
    RCALL   ESCREVE
    MOVLW   '3'
    RCALL   ESCREVE

    BSF     TELA_PRINCIPAL  ; SETA FLAG P/ MOSTRAR TELA PRINCIPAL

    BRA     VARRE           ; VOLTA P/ VARREDURA DOS BOTÕES

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                              FIM DO PROGRAMA                          *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

    END                         ; FIM DO PROGRAMA
```

» Dicas e comentários

Apesar de a estrutura do sistema ficar muito simples com a implementação da rotina ESCREVE, nunca se esqueça de confirmar o estado da saída RS antes de utilizá-la. Conforme foi empregado neste exemplo, recomendamos deixar RS=1 como valor padrão, alterando-o para 0 sempre que desejar escrever algum comando. Em seguida, retornar RS=1 (padrão) para enviar dados.

Para enviar um caractere ao LCD, não esqueça que deve ser especificado seu código ASCII. Para facilitar, o MpLab efetua essa conversão quando você digita um caractere entre aspas simples (exemplo: 'A'). Neste caso, existe diferença entre maiúsculas e minúsculas. Para caracteres não "printáveis", seu código pode ser diretamente definido (exemplo: 0x35). O MpLab, assim como a maioria dos programas para Windows, também aceita a especificação de um caractere que não aparece no teclado através da combinação da tecla ALT seguida do código ASCII, imprimindo o resultado na tela. O problema é que nem sempre o código ASCII é equivalente ao mesmo símbolo para o LCD e para a fonte do MpLab.

Apesar de a placa proposta (McLab2) possuir ligação com o módulo de LCD através de oito vias de dados, é possível utilizá-la para testar e implementar a comunicação com quatro vias. Basta modificar a rotina de inicialização e a de escrita de um byte.

» Exercícios propostos

Agora que o exemplo foi estudado e esclarecido, aproveite para gerar novos problemas e soluções, seguindo os exercícios propostos:

1. Alterar a comunicação para quatro vias.
2. Manter a tela principal disponível, somente quando o sistema é ligado. Após alguns segundos, mostrar uma tela com o nome das quatro teclas e indicar a tecla pressionada através de um caractere de seta (←) ou outro qualquer.

7

Conversor Analógico-Digital Interno

» Introdução

Como já dissemos mais de uma vez, vivemos em um mundo analógico. Sendo assim, por mais digital que seja nosso sistema, é quase impossível não trabalharmos com variáveis analógicas. Seja o valor de uma tensão, ou de uma temperatura, seja uma posição ou o sinal de um sensor qualquer. Conforme aumenta a complexidade do sistema, maior o número de entradas analógicas que devem ser monitoradas pelo microcontrolador. Mas como isso é possível? Uma vez que o PIC só pode processar dados digitais, é necessário converter qualquer sinal analógico em valores digitais. Este é o assunto tratado neste capítulo.

» Teoria

Para começarmos a entender melhor a teoria da conversão dos sinais analógicos em dados digitais, nada melhor que partir para um exemplo prático: os sensores de temperatura normalmente fornecem uma informação analógica (como, por exemplo, uma tensão) proporcional à temperatura e, portanto, para que esta possa ser analisada pelo microcontrolador, necessitamos de um conversor analógico digital (CAD ou simplesmente AD).

O menor passo, ou resolução, de um CAD é dado diretamente pelo seu número de bits e pode ser expresso por:

$$\text{resolução} = \frac{V_{ref}}{2^n}$$

Em que V_{REF} é uma tensão de referência e **n** é o número de bits do conversor.

Cada um dos **n** bits que compõem a informação digital representa uma parcela do valor da tensão analógica a ser convertida, de forma que a soma de todas as contribuições de cada um dos n bits forma a tensão de entrada do conversor AD. Assim, a parcela de tensão proporcional ao bit m do conversor AD é dada por:

$$V_{entrada} = \frac{b_m 2^{(m-1)}}{2^n} V_{ref}$$

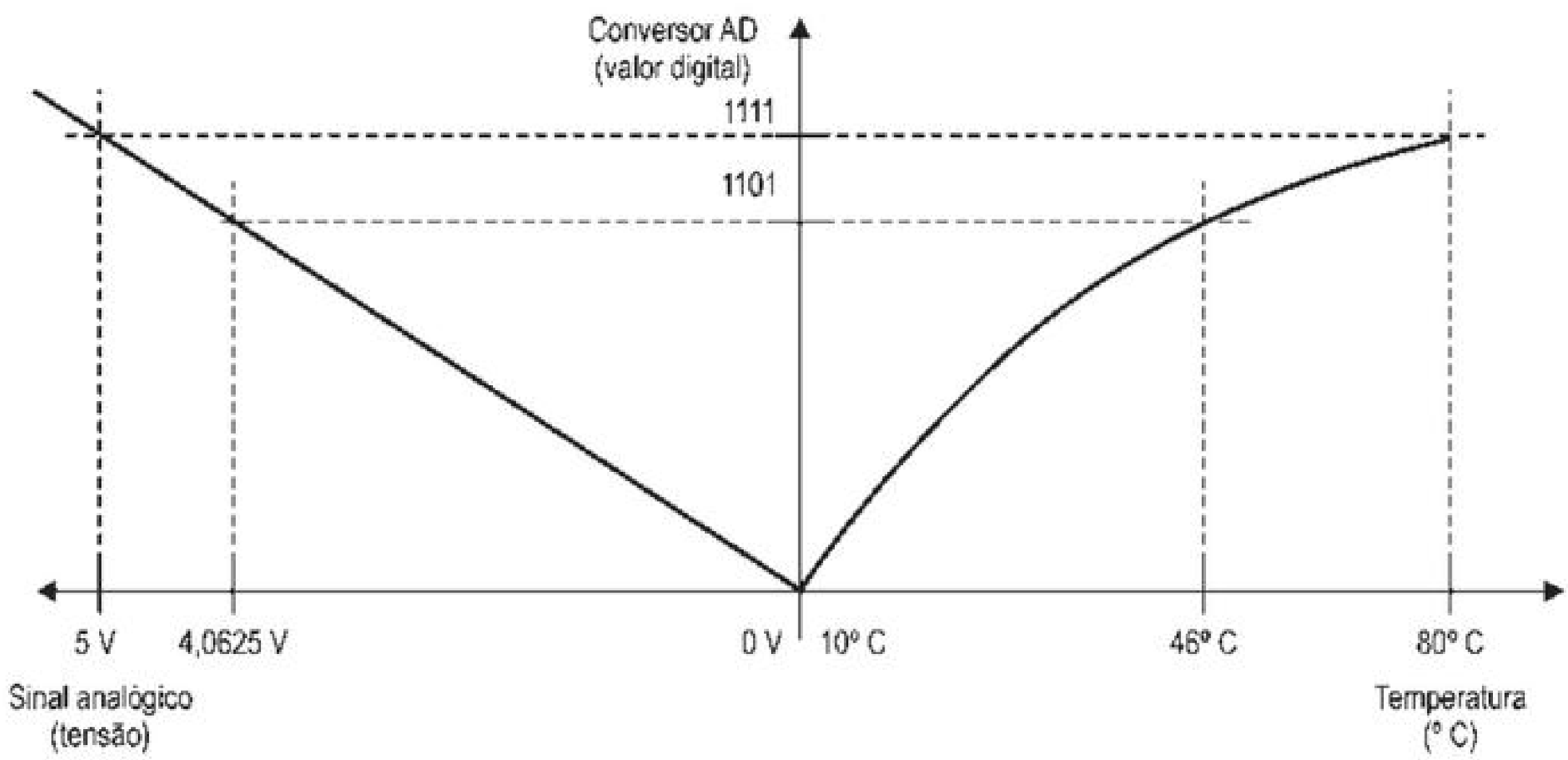
Em que b_m é o valor do bit **m**, ou seja, 0 ou 1.

Veja que apenas os bits em 1 representam algum valor em termos de tensão analógica, uma vez que os bits em zero não contribuem para formar a tensão de entrada. Quanto maior a quantidade de bits, maiores a resolução e a precisão do conversor. O PIC18F4520 possui um conversor interno de 10 bits, mas existem outros modelos com 8 ou 12 bits.

Vamos supor que o CAD para o exemplo da temperatura seja de 4 bits, a tensão de referência seja de 5 V e o valor da conversão em binário seja 1101. A tensão de entrada, então, é:

$$V_{entrada} = \frac{\left(1x2^3 + 1x2^2 + 0x2^1 + 1x2^0\right)}{2^4}(5) = 4{,}0625V$$

Esse valor de tensão é equivalente a uma dada temperatura, e por isso o valor convertido (1101) é equivalente à mesma temperatura. Obviamente, para o exemplo da temperatura, assim como para a maioria dos demais casos práticos, não será necessária somente a conversão. Temos também de nos preocupar com uma equação ou tabela para a adequação dos valores convertidos na unidade desejada. Muitas vezes essa equação ou tabela é a responsável também pela linearização. Assim sendo, temos então duas conversões a serem feitas: a primeira de sinal analógico em valor digital e a segunda de valor digital na unidade realmente desejada, como, por exemplo, °C.



Existem diversas maneiras de implementarmos um conversor analógico/digital. Porém, como este livro trata do microcontrolador PIC18F4520, faremos um estudo aprofundado do sistema de conversão adotado por ele, denominado conversor de aproximação sucessiva.

Nesse tipo de conversor, a conversão é realizada do bit mais significativo para o menos significativo. Uma vez que o bit mais significativo (Msb) representa metade da tensão de referência, conhecer o estado desse bit (0 ou 1) já significa saber se a tensão de entrada é maior ou menor que a metade da referência. Conhecido o bit mais significativo, passa-se ao próximo bit, que representa a metade da metade da tensão de referência, ou seja, ¼ da tensão de referência. A conversão segue assim até o bit menos significativo (Lsb).

Editora Érica - Desbravando o Microcontrolador PIC18 Recursos Avançados - Daniel R. de Sousa, David José de Souza e Nicolás César Lavinia - 1ª Edição

Vamos supor um CAD de quatro bits com tensão de referência de 5 V:

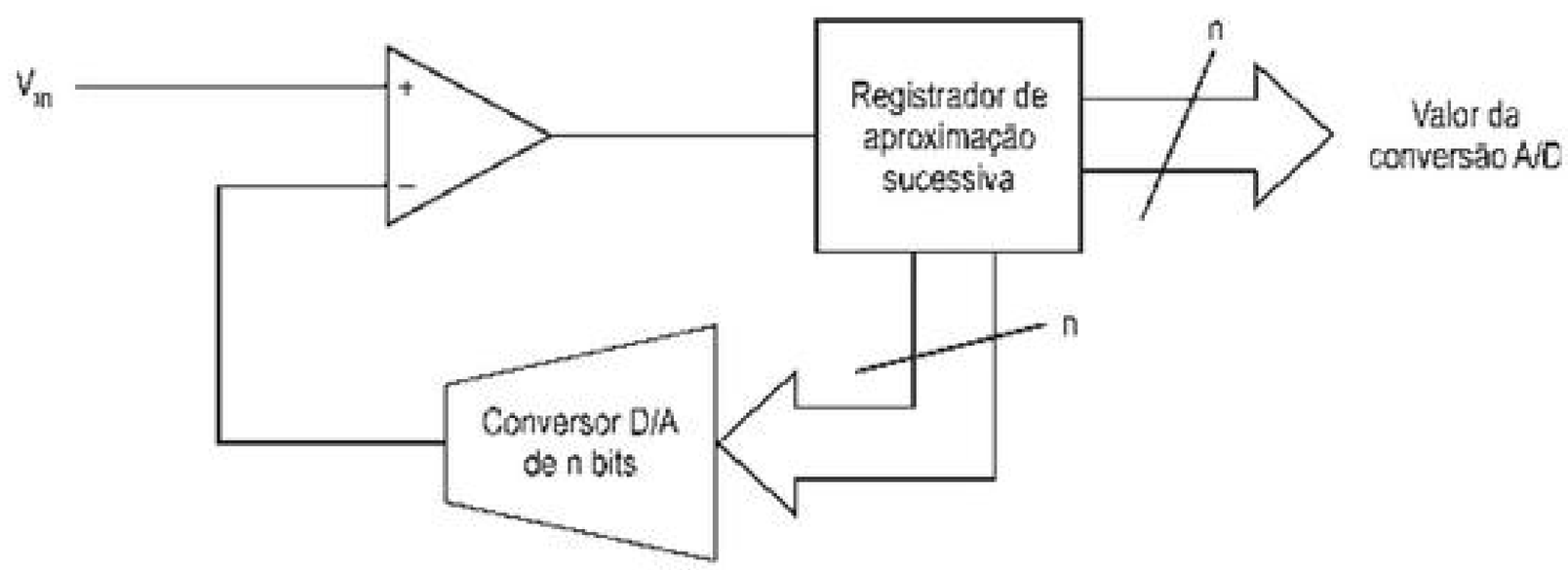
Bit	Tensão
4 (Msb)	2,5000 V
3	1,2500 V
2	0,6250 V
1 (Lsb)	0,3125 V

Suponha que a tensão de entrada seja de 3,3 V. Neste caso, o procedimento de conversão seria assim:

1. Testa-se o bit mais significativo, ou seja, a tensão de entrada é maior do que 2,5 V? Sim, portanto, esse bit vale 1.
2. Testa-se o próximo bit, ou seja, a tensão de entrada é maior do que 3,75 V (2,5 V + 1,25 V)? Não, portanto esse bit é 0.
3. Testa-se o próximo bit, ou seja, a tensão de entrada é maior do que 3,125 V (2,5 V + 0,625 V)? Sim, portanto esse bit é 1.
4. Finalmente testa-se o bit menos significativo, ou seja, a tensão de entrada é maior do que 3,4375 V (2,5 V + 0,625 V + 0,3125 V)? Não, portanto esse bit é 0 e o valor final da conversão em binário é 1010.

Essa forma de conversão é muito rápida, pois veja que para um conversor de n bits são necessárias n interações, independente do valor a ser convertido.

Em termos de hardware (diagrama de blocos), esse tipo de conversor pode ser representado por:



» Recursos do PIC

Vamos agora estudar os modos de operação do conversor interno do PIC18F4520, conhecendo seus recursos e os registradores de configuração e trabalho.

O primeiro item que precisamos saber são as características desse conversor:

- ✓ Conversor interno de 10 bits, dando um total de 1.024 pontos;
- ✓ Até 13 canais de conversão, com diversas configurações entre analógicas e digitais;
- ✓ Quatro tipos de referência: V_{DD} (interna), V_{SS} (interna), V_{REF+} (externa) e V_{REF-} (externa);
- ✓ Frequência de conversão baseada no clock da máquina ou em RC dedicado, possibilitando o funcionamento em modo SLEEP;
- ✓ Três ajustes de frequência (divisores) para o clock de máquina;

- ✓ Dois tipos de justificação para o resultado da conversão: direita e esquerda;
- ✓ Ajuste do tempo de carga do capacitor interno;
- ✓ Um interrupção para término da conversão.

Bem, agora que já sabemos o resumo dos recursos analógicos do PIC, aprenderemos a utilizá-los.

O primeiro conceito que deve ser entendido é que apesar de o microcontrolador possuir diversos canais analógicos, internamente só existe um sistema de conversão. Por isso, somente um canal pode ser utilizado de cada vez. Vamos começar, então, aprendendo como configurar os canais corretamente.

Inicialmente devemos definir, conforme as necessidades do projeto, a quantidade de canais analógicos necessária. Como esse PIC possui até oito canais analógicos, podemos utilizar todos ou só parte deles, deixando os demais pinos configurados como I/Os digitais. O problema é que, infelizmente, não é possível configurar individualmente cada canal, conforme nosso desejo. Existem valores padrão de configuração que devem ser respeitados.

Para configurarmos os canais analógicos precisamos alterar o valor dos bits existentes em **ADCON1<PCFG3:PCFG0>**:

PCFG3: PCFG0	AN12 RB0	AN11 RB4	AN10 RB1	AN9 RB3	AN8 RB2	AN7 RE2	AN6 RE1	AN5 RE0	AN4 RA5	AN3 RA3	AN2 RA2	AN1 RA1	AN0 RA0
0000	A	A	A	A	A	A	A	A	A	A	A	A	A
0001	A	A	A	A	A	A	A	A	A	A	A	A	A
0010	A	A	A	A	A	A	A	A	A	A	A	A	A
0011	D	A	A	A	A	A	A	A	A	A	A	A	A
0100	D	D	A	A	A	A	A	A	A	A	A	A	A
0101	D	D	D	A	A	A	A	A	A	A	A	A	A
0110	D	D	D	D	A	A	A	A	A	A	A	A	A
0111	D	D	D	D	D	A	A	A	A	A	A	A	A
1000	D	D	D	D	D	D	A	A	A	A	A	A	A
1001	D	D	D	D	D	D	D	A	A	A	A	A	A
1010	D	D	D	D	D	D	D	D	A	A	A	A	A
1011	D	D	D	D	D	D	D	D	D	A	A	A	A
1100	D	D	D	D	D	D	D	D	D	D	A	A	A
1101	D	D	D	D	D	D	D	D	D	D	D	A	A
1110	D	D	D	D	D	D	D	D	D	D	D	D	A
1111	D	D	D	D	D	D	D	D	D	D	D	D	D

Pinos configurados como analógicos: A = Analógico; D = Digital.

Além da configuração dos pinos analógicos, conforme tabela apresentada, não se pode esquecer de configurar corretamente os registradores **TRISA** e **TRISE**, de forma que os canais utilizados estejam ajustados para entrada (TRIS = 1). Caso um canal analógico esteja como saída, o sistema de conversão continua funcionando, mas os valores convertidos serão equivalentes aos níveis alto (1 = V_{DD}) e baixo (0 = V_{SS}).

Outro ponto importante a ser observado nessa tabela diz respeito às tensões de referência. A conversão sempre será feita comparando-se a tensão no pino em relação às tensões de referência V_{REF+} e V_{REF-}. Desta forma, quando a tensão de entrada for igual à tensão V_{REF+}, a conversão resultará no valor máximo (1024) e quando a tensão de entrada for igual à V_{REF-}, então o resultado da conversão será zero (0). Como

já foi dito anteriormente, podemos ter quatro tipos de referências, sendo duas internas (V_{SS} e V_{DD}) e duas externas, ligadas aos pinos **RA2** (V_{REF-}) e **RA3** (V_{REF+}). A entrada analógica pode operar, por exemplo, com uma tensão variável de 0 a 5 V. Para isso podemos utilizar as referências internas como $V_{REF+} = V_{DD}$(5 V) e $V_{REF-} = V_{SS}$(0 V). Em uma outra aplicação, o sensor pode variar de 1 a 4 V. Para que não percamos resolução do AD, podemos trabalhar com tensões de referência externas: V_{REF+} (RA3) = 4 V e V_{REF-} (**RA2**) = 1 V. O importante é respeitarmos os limites elétricos impostos a essas referências:

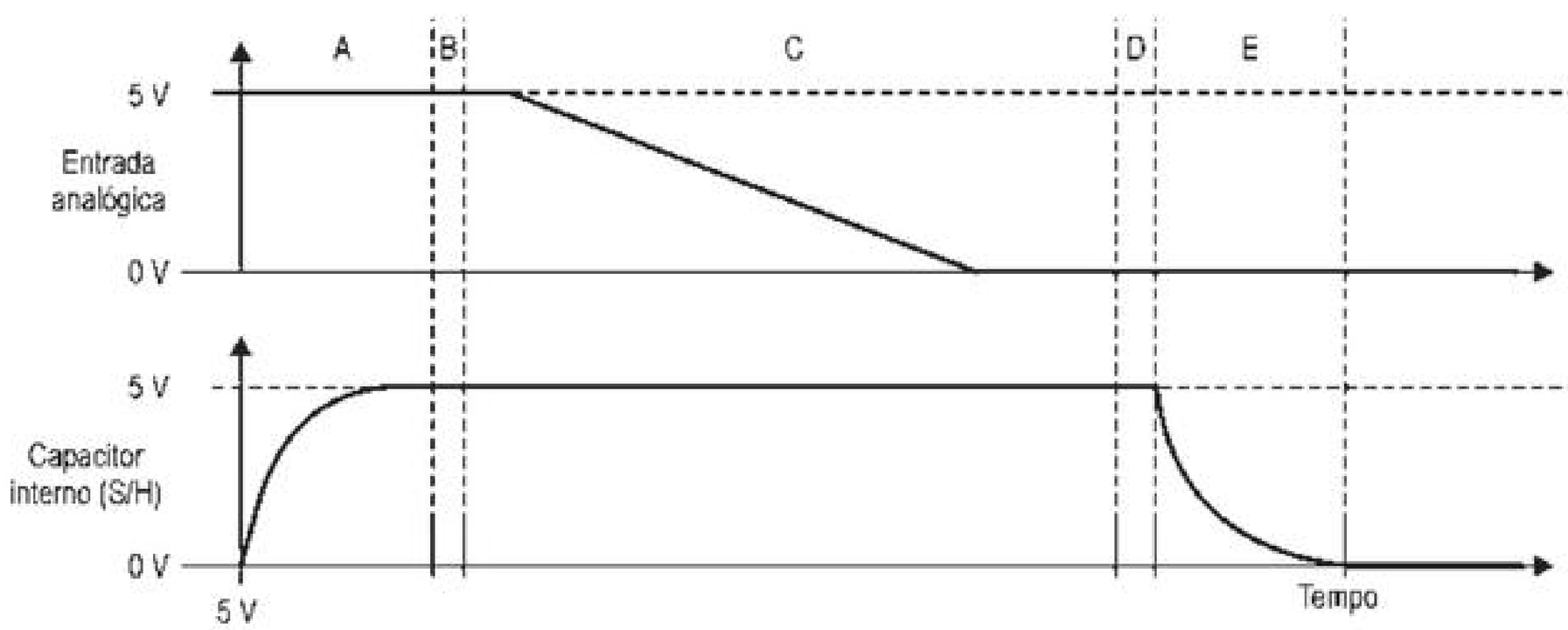
Referência	Mínimo (V)	Máximo (V)
V_{REF+}	$V_{DD} - 2,5$	$V_{DD} + 0,3$
V_{REF-}	$V_{SS} - 0,3$	$V_{REF+} - 2,0$
$(V_{REF+} - V_{REF-})$	2,0	$V_{DD} + 0,3$

Para configurarmos os canais analógicos **AN2** e **AN3**, como entrada de tensão de referência precisamos alterar o valor dos bits existentes em **ADCON1<VCFG1:VCFG0>**:

VCFG1:VCFG0		V_{REF+}	V_{REF-}
0	0	VDD	VSS
0	1	AN3	VSS
1	0	VDD	AN2
1	1	AN3	AN2

O próximo ponto a ser aprendido diz respeito à velocidade e, consequentemente, aos tempos de amostragem para a conversão AD. Vamos começar entendendo como o sistema de conversão funciona internamente.

Para evitarmos problemas de ruído e variações da entrada analógica durante o processo de conversão (não podemos esquecer que nada é absolutamente instantâneo), o PIC utiliza internamente um processo denominado Sample and Hold (S/H). Esse nome poderia ser traduzido como amostra e congela. Mas como isso é feito? Muito simples. Internamente o PIC possui um capacitor (25pF) que é ligado ao canal analógico em uso. Sendo assim, ele fica carregado com a tensão existente no canal (amostra). Quando o processo de conversão é iniciado, esse capacitor é desligado, automaticamente, do canal analógico, mantendo a tensão sobre o capacitor constante (congela). Por isso, durante todo o processo de conversão, a tensão utilizada é a existente no capacitor e não mais a do canal analógico. Mesmo que a tensão externa no canal varie, a conversão não é afetada.



Agora que já sabemos como o sistema funciona no âmbito do hardware, vamos conhecer os tempos a serem respeitados de forma a não gerarmos erros durante a conversão. A figura anterior ilustrou o processo, de um gráfico resumido dos tempos de conversão:

Conforme o gráfico mostrado, podemos agora analisar cada um dos tempos envolvidos no sistema de conversão:

Código	Nome
A	Adequação do capacitor
B	Desligamento do capacitor
C	Conversão
D	Religamento do capacitor
E	Nova adequação do capacitor

Adequação do capacitor

Internamente o capacitor nunca fica desligado, exceto durante a conversão. Ele sempre se encontra ligado ao sistema de entradas analógicas. Mas imaginemos que no momento o canal ativo esteja ligado ao GND. Desta forma, o capacitor fica totalmente descarregado. No momento seguinte desejamos medir outra entrada analógica (outro canal) e por isso o sistema será chaveado internamente (veremos como isso é possível mais à frente). Caso o outro canal esteja com uma tensão de 5 V, o capacitor interno terá de ser completamente carregado antes que possamos começar a efetuar a conversão. Obviamente essa carga não é instantânea.

Para evitar problemas, sugerimos que o tempo padrão deixado para garantir a carga do capacitor seja de pelo menos 40 µs. Obviamente, este é um tempo que garante a pior condição. Na verdade, o tempo de adequação pode ser bem menor que isso, chegando no mínimo a 1 µs. Essa variação depende basicamente da diferença de tensão entre a entrada e o capacitor, da temperatura e da impedância de entrada. Essa impedância deve ser de no mínimo 50 Ω e no máximo 2,5 kΩ. Quanto menor a impedância, menor também o tempo de adequação. Para o cálculo exato desse tempo consulte o data sheet do PIC.

Desligamento do capacitor

Depois de iniciada a conversão, o capacitor é desligado internamente. Isso é feito em aproximadamente 100 ns, sendo um valor praticamente desprezível.

Conversão

O tempo de conversão já não é tão desprezível assim. Para entendermos o tempo de conversão precisaremos entender melhor a frequência de trabalho do AD. Para que o sistema de conversão funcione corretamente, um clock deve ser aplicado a ele. Cada período desse clock é chamado de T_{AD} e é equivalente ao tempo de conversão de 1 bit. Como nosso conversor é de 10 bits, o tempo total da conversão é de 10 T_{AD} + 2 T_{AD}. Os dois períodos adicionais são para a adequação e o início da conversão. O valor de T_{AD} depende da frequência empregada, e para isso o sistema possui um RC interno ou divisores para o oscilador da própria máquina, mas veremos isso mais adiante. Assim sendo, o tempo de conversão é 12 T_{AD}.

Religamento do capacitor

Ao final da conversão, o valor dos registradores de resultado é atualizado, o flag da interrupção é marcado é o capacitor religado. O tempo mínimo recomendado para que tudo isso aconteça é de 2 T_{AD}.

Nova adequação do capacitor

Durante o tempo da conversão, o valor da tensão no capacitor interno não foi alterado; porém, ao término da conversão, ele será religado à entrada analógica, que pode ter sofrido uma variação brusca (como no exemplo), seja por imposição do próprio sistema, seja por uma troca de canal. Por isso, entre uma conversão e outra é necessário haver nova adequação da carga do capacitor. Recomendamos novamente um tempo de 40 µs. As observações do item "Adequação do capacitor" também são válidas para a diminuição desse tempo.

Esse microcontrolador permite ajustar o tempo de aquisição por hardware através dos bits **ADCON2<ACQT2:ACQT0>**:

ACQT2	ACQT1	ACQT0	Tempo de aquisição
0	0	0	0 T_{AD}(1)
0	0	1	2 T_{AD}
0	1	0	4 T_{AD}
0	1	1	6 T_{AD}
1	0	0	8 T_{AD}
1	0	1	12 T_{AD}
1	1	0	16 T_{AD}
1	1	1	20 T_{AD}
Nota: 1 - Tempo a ser definido por software.			

Agora que já entendemos um pouco mais as características e conceitos do AD, vejamos como realmente devemos proceder para operar com ele.

Depois de ter configurado quais serão os canais utilizados (**ADCON1** e **TRIS**), temos de configurar também a frequência de trabalho. Para isso devemos ajustar 2 bits em **ADCON2<ADCS2:ADCS0>**:

ADCS2	ADCS1	ADCS0	Frequência
0	0	0	F_{osc} / 2
0	0	1	F_{osc} / 8
0	1	0	F_{osc} / 32
0	1	1	RC Interno
1	0	0	F_{osc} / 4
1	0	1	F_{osc} / 16
1	1	0	F_{osc} / 64
1	1	1	RC Interno

Para os três primeiros ajustes, a frequência de trabalho do AD será a frequência do oscilador externo do PIC (F_{OSC}) dividida por uma das opções (2, 4, 8, 16, 32 ou 64). Desta forma, obtemos o valor de T_{AD}. Por exemplo, para um cristal externo de 4 MHz e uma opção de F_{osc} / 8 teremos:

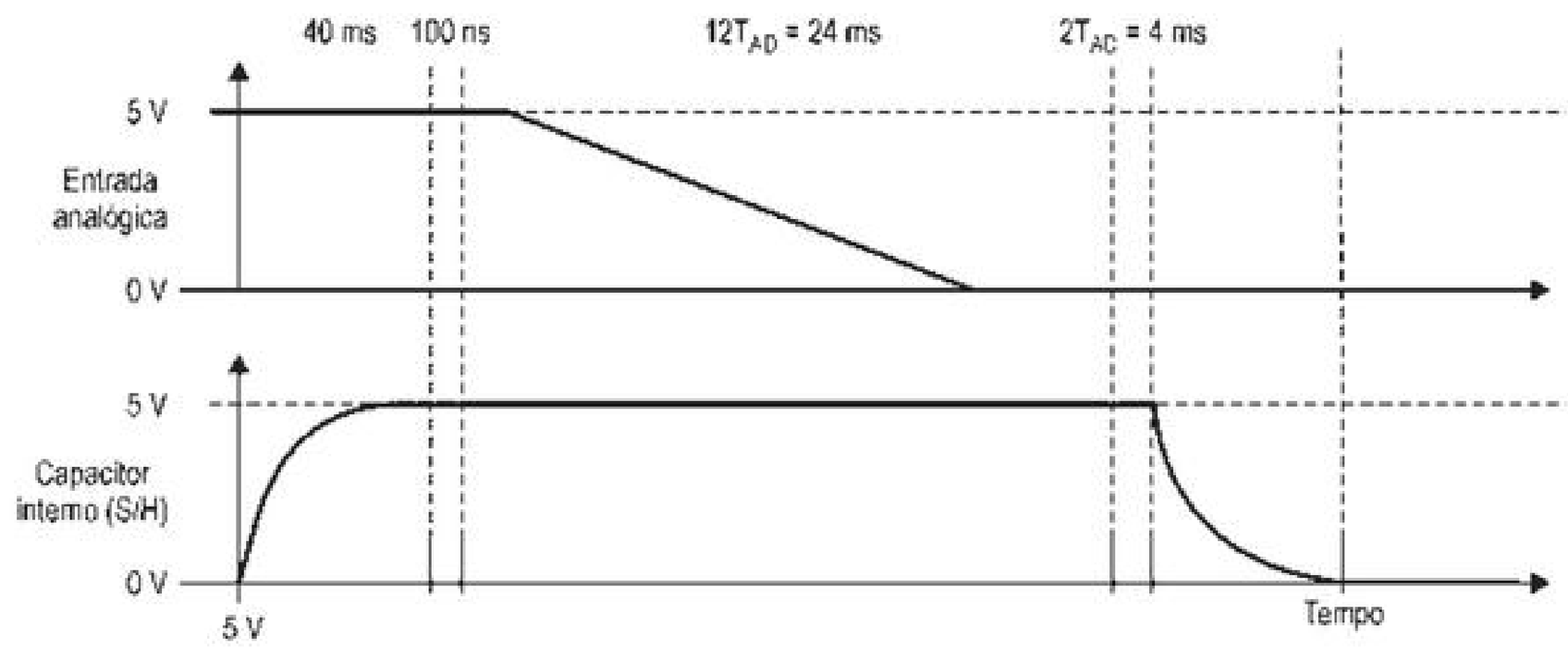
$$T_{AD} = 8 / 4.000.000$$

$$T_{AD} = 0{,}000002 \text{ s ou } 2 \text{ µs}$$

Neste caso, o T_{AD} é de 2 µs. O importante na escolha do cristal e da opção de divisão da frequência é respeitar os valores mínimos de T_{AD} aceitos pelo PIC. No caso do PIC18F4520 (Standard) esse valor deve

ser maior que 0,7 µs. Valores muito altos para o T_{AD} também não são muito aconselháveis, por isso recomendamos que não seja ultrapassado o limite de 25 µs.

Continuando o exemplo dado, verifiquemos a frequência máxima de amostragem para a comutação entre canais:



Desconsiderando-se o tempo de desligamento (desprezível), teríamos um tempo total para a conversão de 68 µs (40 + 24 + 4). Sendo assim, nossa frequência de amostragem máxima seria de 14,7 kHz.

A última opção de escolha para o clock do AD refere-se a um RC interno dedicado para essa finalidade. Neste caso o valor nominal para T_{AD} é de 4 µs (pode variar de 2 a 6 µs). Essa opção pode ser utilizada para que o sistema de conversão continue funcionando mesmo em modo SLEEP. Na verdade, esse RC pode ser usado a qualquer momento, mas é recomendado que para sistemas que operem com frequências superiores a 4 MHz esse oscilador seja usado somente no modo SLEEP.

A última configuração necessária quanto às características de funcionamento do AD diz respeito à forma como o resultado é armazenado nos registradores específicos. Como a conversão gera um resultado de 10 bits, são necessários dois registradores para armazenar este valor. Para isso existem os registradores **ADRESH** e **ADRESL**, equivalentes às partes alta e baixa do resultado. Acontece que a soma desses dois registradores resulta 16 bits, por isso o resultado pode ser armazenado neles justificando pela esquerda ou direita. O bit de configuração **ADCON2<ADFM>** ajusta essa orientação:

ADFM	Frequência
1	Justificado pela direita. Utiliza todos os bits de **ADRESL<7:0>** e somente dois bits de **ADRESH<1:0>.**
0	Justificado pela esquerda. Utiliza todos os bits de **ADRESH<7:0>** e somente dois bits de **ADRESL<7:6>.**

Essa justificação é válida quando queremos, por exemplo, trabalhar com somente 8 bits do conversor. Neste caso, podemos justificar pela esquerda e acessar somente a parte mais significativa através de **ADRESH**. Com isso criamos um filtro, em que jogamos fora os dois bits menos significativos, que se encontram em **ADRESL**.

Agora que os ajustes da configuração já foram feitos, podemos finalmente efetuar uma conversão.

Comecemos ligando o sistema de conversão. Isso é feito através de **ADCON0<ADON>**:

ADON	Estado do AD
0	Sistema desligado
1	Sistema ligado

O sistema pode ser mantido desligado sempre que não estiver sendo utilizado para redução do consumo.

A próxima ação é a escolha do canal a ser utilizado, entre os oito disponíveis. Essa escolha é feita através de **ADCON0<CHS3:CHS0>**:

CHS3:CHS0	Canal Selecionado
0000	Canal 0
0001	Canal 1
0010	Canal 2
0011	Canal 3
0100	Canal 4
0101	Canal 5
0110	Canal 6
0111	Canal 7

CHS3: CHS0	Canal Selecionado
1000	Canal 8
1001	Canal 9
1010	Canal 10
1011	Canal 11
1100	Canal 12
1101	Não implementado
1110	Não implementado
1111	Não implementado

O último passo é iniciar a conversão por meio do bit **ADCON0<GO/DONE>**:

GO/DONE	Estado do AD
1	Inicia a conversão.
0	Indica que a conversão terminou. Caso seja forçado manualmente, cancela a conversão atual.

Depois de iniciada a conversão, todo o processo é executado. O capacitor interno é desconectado. A conversão é efetuada (12 T_{AD}) e logo em seguida (no próximo ciclo de máquina), os registradores **ADRESH** e **ADRESL** são atualizados. Nesse momento, o bit **ADCON0<GO/DONE>** volta automaticamente para 0 (zero) e poder ser monitorado pelo programa para verificar o término da conversão. Além disso, o flag da interrupção **PIR1<ADIF>** também é ativado. Caso a conversão seja abortada manualmente através de **ADCON0<GO/DONE>=0**, os registradores **ADRESH** e **ADRESL** não são alterados.

Resumo dos registradores associados ao AD

Endereço	Nome	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
FF2h	INTCON	INTCON	GIE/ GIEH	PEIE/ GIEL	T0IE	INTE	RBIE	TMR0IF	INTF
F9Eh	PIR1	PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
F9Dh	PIE1	PSPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
F9Fh	IPR1	PSPIP	ADIP	RCIP	RXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP
FC4h	ADRESH	Resultado da conversão (parte alta)							
FC3h	ADRESL	Resultado da conversão (parte baixa)							
FC2h	ADCON0	-	-	CHS3	CHS2	CHS1	CHS0	GO/DONE	ADON
FC1h	ADCON1	-	-	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0
FC0h	ADCON2	ADFM	-	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0
F92h	TRISA	Configuração I/Os							
F93h	TRISB	Configuração I/Os							
F96h	TRISE	IBF	OBF	IBOV	PSPMODE	-	Configuração I/Os		

Não usado para essa finalidade. Para obter mais informações, consulte o Apêndice A.

Para fazer uso da interrupção de AD, não se esqueça de efetuar os seguintes ajustes antes de iniciar a conversão:

- ✓ Limpar o flag **PIR1<ADIF>=0**;
- ✓ Ligar a interrupção de AD em **PIE1<ADIE>=1**;

- ✓ Ligar as interrupções de periféricos em **INTCON<PEIE/GIEL>=1**;
- ✓ Ligar a chave geral das interrupções em **INTCON<GIE/GIEH>=1**.

» Lógica do exemplo

Neste exemplo efetuamos a leitura da tensão regulada pelo potenciômetro P2 da placa proposta (McLab2).

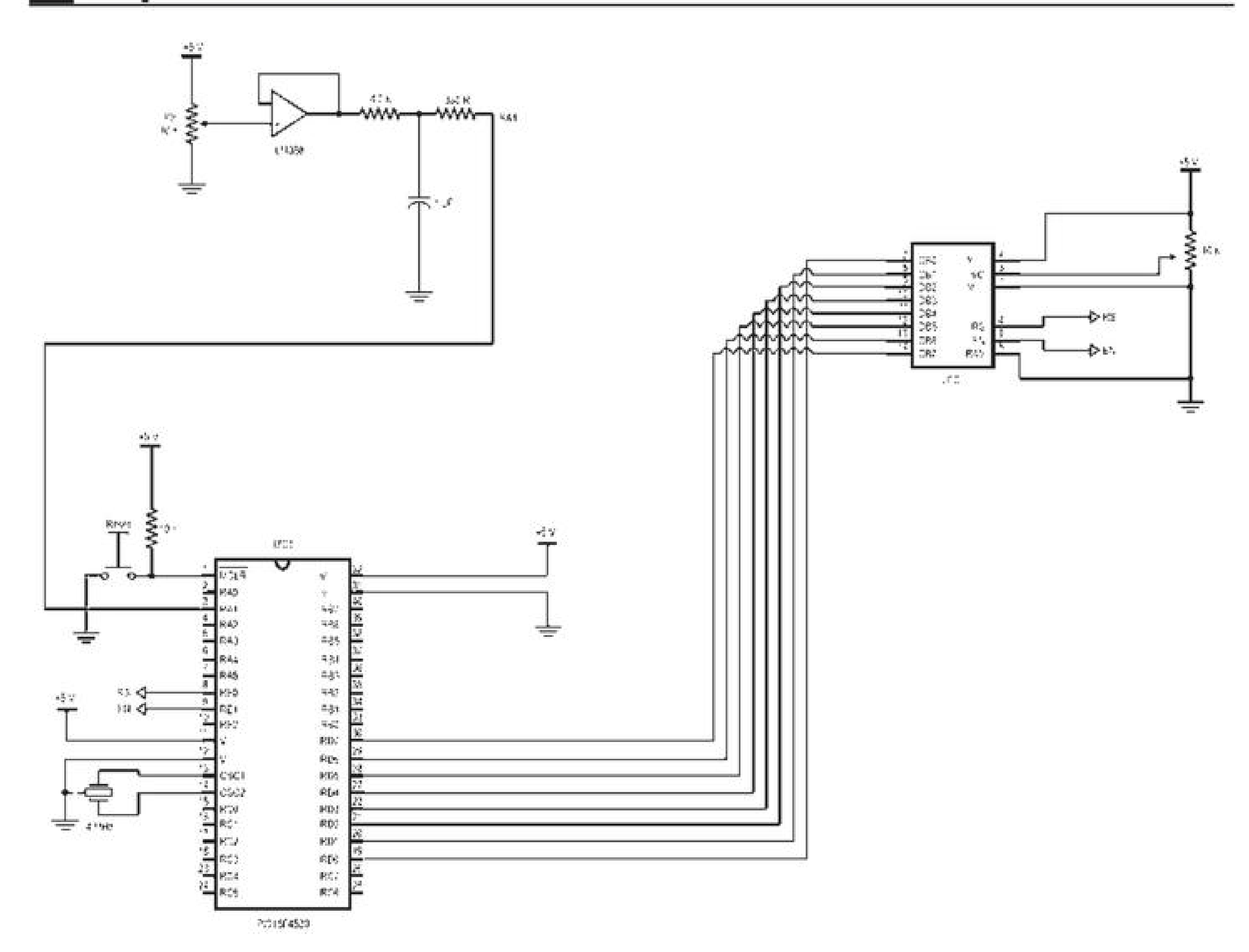
A conversão é feita diretamente no loop principal, sem a utilização de nenhuma interrupção, nem para verificar o término da conversão, nem para definir uma frequência de amostragem. Desta forma, a conversão será feita uma após a outra, na frequência definida pelo período do loop principal.

Terminada a conversão, descartamos os dois bits menos significativos e consideramos somente o resultado armazenado em **ADRESH**. Com isso já executamos uma espécie de filtragem, evitando assim que o valor final fique oscilando.

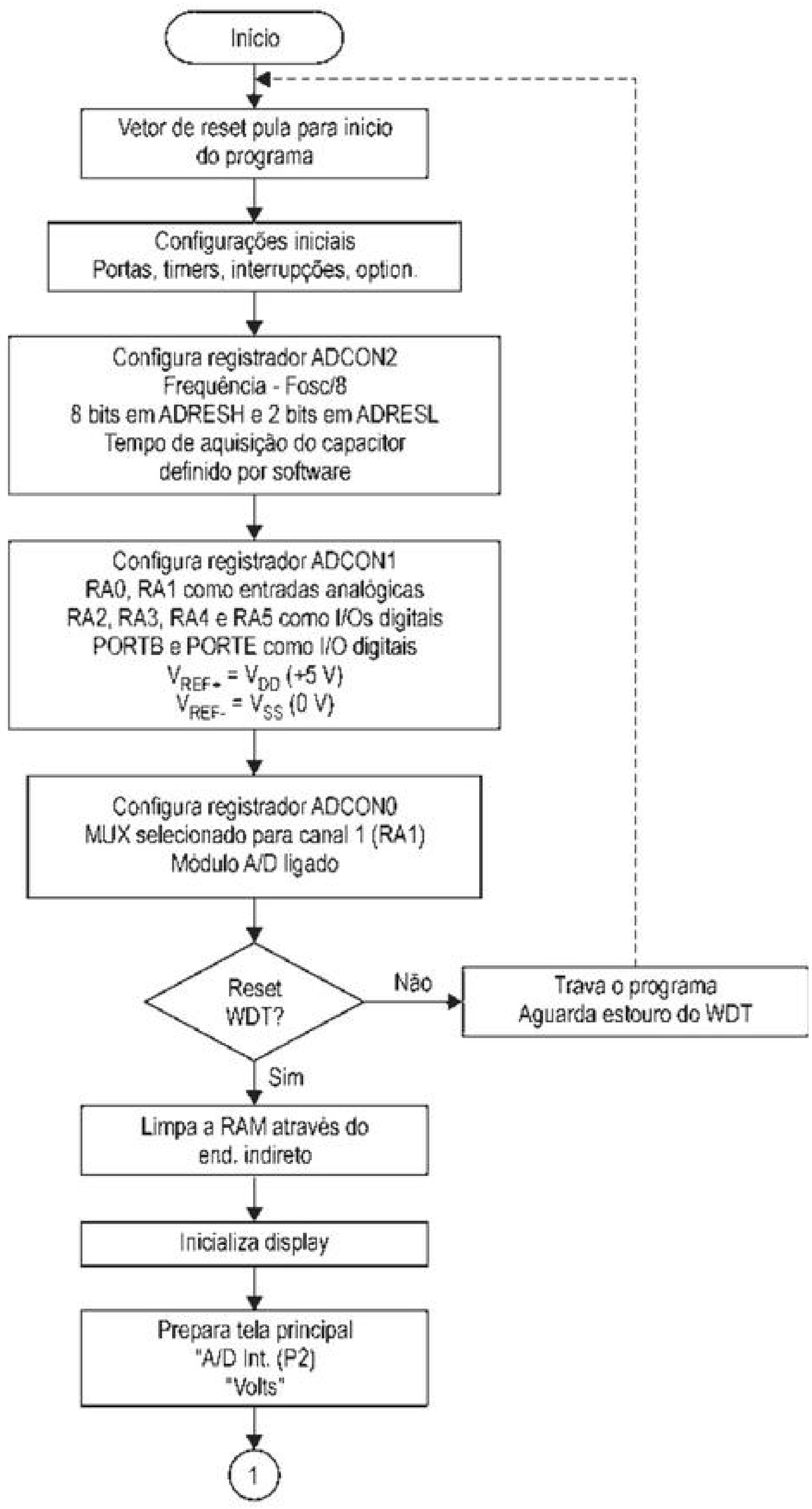
Aplicaremos então uma regra de três básica para converter o valor do AD na unidade desejada: volts. Conseguimos isso considerando que, quando o AD resultar em 0 (zero), a entrada possui 0,0 V, e quando o AD resultar em 255, a entrada é equivalente a 5,0 V. Essa conta também diminui a resolução, aumentando ainda mais a estabilidade.

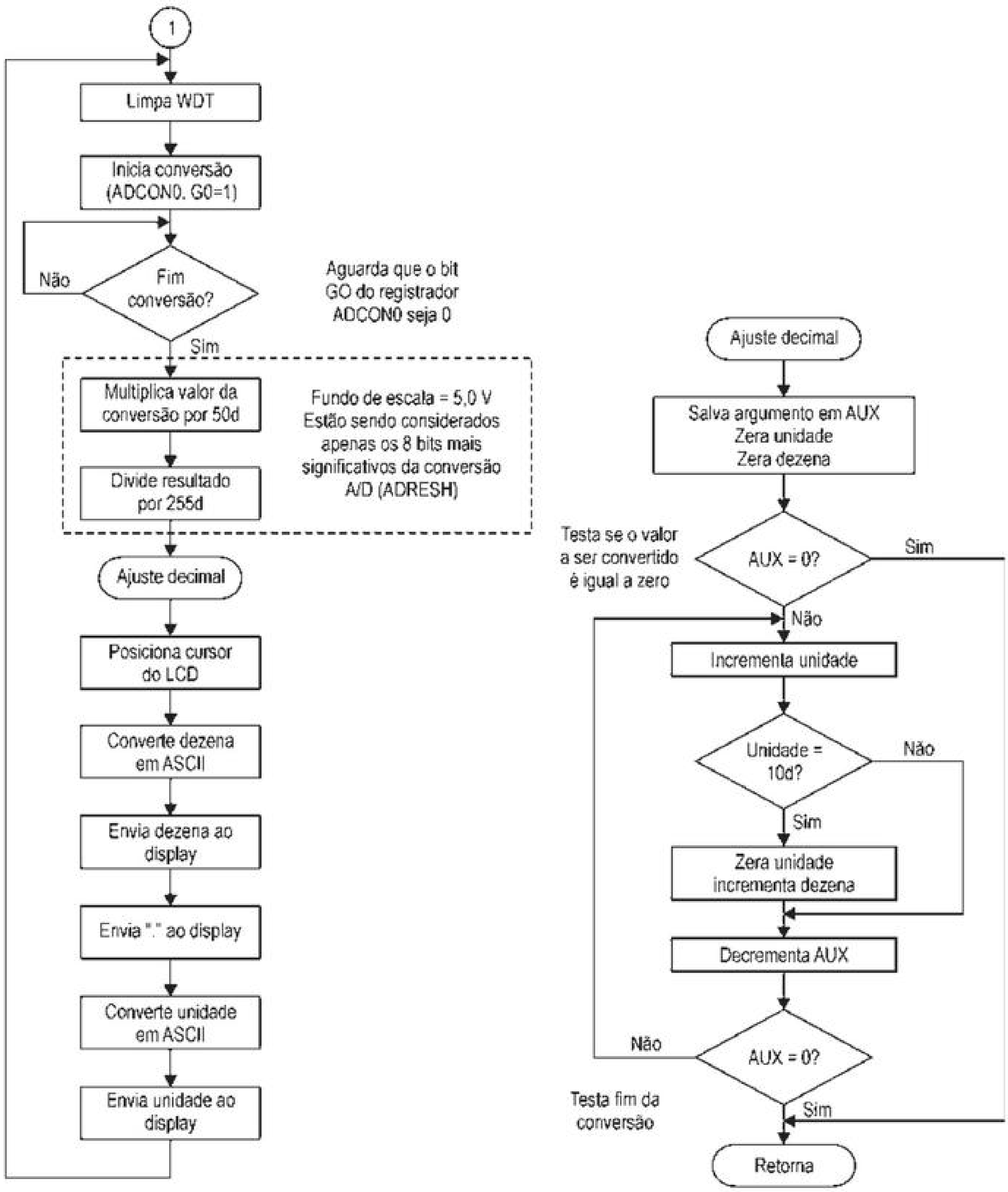
Por último escrevemos o valor, já em volts, no LCD.

» Esquema elétrico



» Fluxograma


Início
Vetor de reset pula para início do programa
Configurações iniciais
Portas, timers, interrupções, option.
Configura registrador ADCON2
Frequência - Fosc/8
8 bits em ADRESH e 2 bits em ADRESL
Tempo de aquisição do capacitor definido por software
Configura registrador ADCON1
RA0, RA1 como entradas analógicas
RA2, RA3, RA4 e RA5 como I/Os digitais
PORTB e PORTE como I/O digitais
$V_{REF+} = V_{DD}$ (+5 V)
$V_{REF-} = V_{SS}$ (0 V)
Configura registrador ADCON0
MUX selecionado para canal 1 (RA1)
Módulo A/D ligado
Reset WDT?
Não
Trava o programa
Aguarda estouro do WDT
Sim
Limpa a RAM através do end. indireto
Inicializa display
Prepara tela principal
"A/D Int. (P2)
"Volts"
1



Código

```
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  *              DESBRAVANDO O PIC18 - RECURSOS AVANÇADOS                  *
;  *                              EXEMPLO 4                                 *
;  *                                                                        *
;  * DANIEL RODRIGUES DE SOUSA, DAVID JOSÉ DE SOUZA e  NICOLÁS CÉSAR LAVINIA *
;  *                                                                        *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```


Editora Érica - Desbravando o Microcontrolador PIC18 Recursos Avançados - Daniel R. de Sousa, David José de Souza e Nicolás César Lavinia - 1ª Edição

```
; *     DATA : 03/08/2009                                                *
; *   VERSÃO : 1.0                                                       *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                          DESCRIÇÃO GERAL                            *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTE EXEMPLO FOI ELABORADO PARA EXPLICAR O FUNCIONAMENTO DO MÓDULO DE
;  CONVERSÃO ANALÓGICO DIGITAL INTERNO DO PIC. É CONVERTIDO O VALOR ANALÓGICO
;  PRESENTE NO PINO RA2 DO MICROCONTROLADOR, SENDO QUE ESTE VALOR PODE SER
;  ALTERADO ATRAVÉS DO POTENCIÔMETRO P2 DA PLACA MCLAB2. O VALOR DA CONVERSÃO
;  AD É AJUSTADO NUMA ESCALA DE 0 A 5V E MOSTRADO NO LCD.
;  FOI UTILIZADA A ROTINA DE DIVISÃO DE 16x16. ESTAS ROTINAS FORAM RETIRADAS
;  DE APLICATION NOTES DA PRÓPRIA MICROCHIP.
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                      CONFIGURAÇÕES PARA GRAVAÇÃO                    *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
    CONFIG  OSC = XT, FCMEN = OFF, IESO = OFF, PWRT = ON, BOREN = ON, BORV = 0
    CONFIG  WDT = ON, WDTPS = 128, MCLRE = ON, LPT1OSC = OFF, PBADEN = OFF
    CONFIG  CCP2MX = PORTC, STVREN = ON, LVP = OFF, DEBUG = OFF, XINST = OFF
    CONFIG  CP0 = OFF, CP1 = OFF, CP2 = OFF, CP3 = OFF, CPB = OFF, CPD = OFF
    CONFIG  WRT0 = OFF, WRT1 = OFF, WRT2 = OFF, WRT3 = OFF, WRTB = OFF
    CONFIG  WRTC = OFF, WRTD = OFF, EBTR0 = OFF, EBTR1 = OFF, EBTR2 = OFF
    CONFIG  EBTR3 = OFF, EBTRB = OFF

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                        DEFINIÇÃO DAS VARIÁVEIS                      *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
    CBLOCK  0X00            ; BANK 0 --> 0X00..0X5F --> ACCESS BANK

        ACCaHI              ; ACUMULADOR a DE 16 BITS UTILIZADO
        ACCaLO              ; NA ROTINA DE DIVISÃO

        ACCbHI              ; ACUMULADOR b DE 16 BITS UTILIZADO
        ACCbLO              ; NA ROTINA DE DIVISÃO

        ACCcHI              ; ACUMULADOR c DE 16 BITS UTILIZADO
        ACCcLO              ; NA ROTINA DE DIVISÃO

        ACCdHI              ; ACUMULADOR d DE 16 BITS UTILIZADO
        ACCdLO              ; NA ROTINA DE DIVISÃO

        temp                ; CONTADOR TEMPORÁRIO UTILIZADO
                            ; NA ROTINA DE DIVISÃO

        TEMPO0
        TEMPO1              ; TEMPORIZADORES P/ ROTINA DE DELAY

        AUX                 ; REGISTRADOR AUXILIAR DE USO GERAL

        UNIDADE             ; ARMAZENA VALOR DA UNIDADE DA TENSÃO
        DEZENA              ; ARMAZENA VALOR DA DEZENA DA TENSÃO

    ENDC

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                 DEFINIÇÃO DAS VARIÁVEIS INTERNAS DO PIC             *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  O ARQUIVO DE DEFINIÇÕES DO PIC UTILIZADO DEVE SER REFERENCIADO PARA QUE
;  OS NOMES DEFINIDOS PELA MICROCHIP POSSAM SER UTILIZADOS, SEM A NECESSIDADE
;  DE REDIGITAÇÃO.

    #INCLUDE <P18F4520.INC> ; MICROCONTROLADOR UTILIZADO
```

```
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                          CONSTANTES INTERNAS                            *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  A DEFINIÇÃO DE CONSTANTES FACILITA A PROGRAMAÇÃO E A MANUTENÇÃO.

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                   DECLARAÇÃO DOS FLAGs DE SOFTWARE                      *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  A DEFINIÇÃO DE FLAGs AJUDA NA PROGRAMAÇÃO E ECONOMIZA MEMÓRIA RAM.

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                              ENTRADAS                                   *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  AS ENTRADAS DEVEM SER ASSOCIADAS A NOMES PARA FACILITAR A PROGRAMAÇÃO E
;  FUTURAS ALTERAÇÕES DO HARDWARE.

;  ESTE PROGRAMA UTILIZA APENAS UMA ENTRADA P/ O CONVERSOR AD.
;  ESTA ENTRADA NÃO PRECISA SER DECLARADA, POIS O SOFTWARE NUNCA FAZ
;  REFERÊNCIA A ELA DE FORMA DIRETA, POIS O CANAL AD A SER CONVERTIDO É
;  SELECIONADO NO REGISTRADOS ADCON0 DE FORMA BINÁRIA E NÃO ATRAVÉS DE
;  DEFINES. PORÉM PARA FACILITAR O ENTENDIMENTO DO HARDWARE VAMOS DECLARAR
;  ESTA ENTRADA NORMALMENTE.

#DEFINE CAD_P2      PORTA,1          ; ENTRADA AD P/ O POTENCIÔMETRO P2

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                              SAÍDAS                                     *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  AS SAÍDAS DEVEM SER ASSOCIADAS A NOMES PARA FACILITAR A PROGRAMAÇÃO E
;  FUTURAS ALTERAÇÕES DO HARDWARE.

#DEFINE DISPLAY     LATD             ; BARRAMENTO DE DADOS DO DISPLAY

#DEFINE RS          LATE,0           ; INDICA P/ O DISPLAY UM DADO OU COMANDO
                                     ; 1 -> DADO
                                     ; 0 -> COMANDO

#DEFINE ENABLE      LATE,1           ; SINAL DE ENABLE P/ DISPLAY
                                     ; ATIVO NA BORDA DE DESCIDA

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                  VETOR DE RESET DO MICROCONTROLADOR                     *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  POSIÇÃO INICIAL PARA EXECUÇÃO DO PROGRAMA

    ORG     0X0000                   ; ENDEREÇO DO VETOR DE RESET
    BRA     CONFIGURACAO             ; PULA PARA CONFIG DEVIDO A REGIÃO
                                     ; DESTINADA AS ROTINAS SEGUINTES

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                   ROTINA DE DELAY (DE 1MS ATÉ 256MS)                    *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTA É UMA ROTINA DE DELAY VARIÁVEL, COM DURAÇÃO DE 1MS X O VALOR PASSADO
;  EM WORK (W).

DELAY_MS
    MOVWF   TEMPO1               ; CARREGA TEMPO1 (UNIDADES DE MS)
    MOVLW   .250
    MOVWF   TEMPO0               ; CARREGA TEMPO0 (P/ CONTAR 1MS)

    CLRWDT                       ; LIMPA WDT (PERDE TEMPO)
    DECFSZ  TEMPO0,F             ; FIM DE TEMPO0 ?
    BRA     $-.4                 ; NÃO - VOLTA 2 INSTRUÇÕES
```

```
                                ; SIM - PASSOU-SE 1ms
    DECFSZ  TEMPO1,F            ; FIM DE TEMPO1 ?
    BRA     $-.12               ; NÃO - VOLTA 6 INSTRUÇÕES
    RETURN                      ; SIM - RETORNA

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                ROTINA DE ESCRITA DE UM CARACTERE NO DISPLAY           *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTA ROTINA ENVIA UM CARACTERE PARA O MÓDULO DE LCD. O CARACTERE A SER
;  ESCRITO DEVE SER COLOCADO EM WORK (W) ANTES DE CHAMAR A ROTINA.

ESCREVE
    MOVWF   DISPLAY             ; ATUALIZA DISPLAY (PORTD)
    NOP                         ; PERDE 1US PARA ESTABILIZAÇÃO
    BSF     ENABLE              ; ENVIA UM PULSO DE ENABLE AO DISPLAY
    BRA     $+.2                ; .
    BCF     ENABLE              ; .

    MOVLW   .1
    RCALL   DELAY_MS            ; DELAY DE 1ms
    RETURN                      ; RETORNA

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                                AJUSTE DECIMAL                         *
; *                W [HEX] =  DEZENA [DEC] ; UNIDADE [DEC]                *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTA ROTINA RECEBE UM ARGUMENTO PASSADO PELO WORK E RETORNA NAS VARIÁVEIS
;  DEZENA E UNIDADE O NÚMERO BCD CORRESPONDENTE AO PARÂMETRO PASSADO.

AJUSTE_DECIMAL
    MOVWF   AUX                 ; SALVA VALOR A CONVERTER EM AUX
    CLRF    UNIDADE
    CLRF    DEZENA              ; RESETA REGISTRADORES

    MOVF    AUX,F               ; VALOR A CONVERTER = 0 ?
    BZ      $+.18               ; SIM - RETORNA
    INCF    UNIDADE,F           ; NÃO - INCREMENTA UNIDADE

    MOVF    UNIDADE,W
    XORLW   0X0A                ; UNIDADE = 10d ?
    BNZ     $+.6                ; NÃO
    CLRF    UNIDADE             ; SIM - RESETA UNIDADE
    INCF    DEZENA,F            ; INCREMENTA DEZENA

    DECFSZ  AUX,F               ; FIM DA CONVERSÃO ?
    BRA     $-.14               ; NÃO - VOLTA P/ CONTINUAR CONVERSÃO
    RETURN                      ; SIM

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                              ROTINA DE DIVISÃO                        *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;*********************************************************************
;                        Double Precision Division
;*********************************************************************
;   Division : ACCb(16 bits) / ACCa(16 bits) -> ACCb(16 bits) with
;                                               Remainder in ACCc (16 bits)
;       (a) Load the Denominator in location ACCaHI & ACCaLO ( 16 bits )
;       (b) Load the Numerator in location ACCbHI & ACCbLO ( 16 bits )
;       (c) RCALL D_divF
;       (d) The 16 bit result is in location ACCbHI & ACCbLO
;       (e) The 16 bit Remainder is in locations ACCcHI & ACCcLO
;*********************************************************************

D_divF
```

```
    MOVLW   .16
    MOVWF   temp            ; CARREGA CONTADOR PARA DIVISÃO

    MOVFF   ACCbHI,ACCdHI
    MOVFF   ACCbLO,ACCdLO   ; SALVA ACCb EM ACCd

    CLRF    ACCbHI
    CLRF    ACCbLO          ; LIMPA ACCb

    CLRF    ACCcHI
    CLRF    ACCcLO          ; LIMPA ACCc

DIV
    BCF     STATUS,C
    RLCF    ACCdLO,F
    RLCF    ACCdHI,F
    RLCF    ACCcLO,F
    RLCF    ACCcHI,F
    MOVF    ACCaHI,W
    SUBWF   ACCcHI,W        ;check if a>c
    BTFSS   STATUS,Z
    BRA     NOCHK
    MOVF    ACCaLO,W
    SUBWF   ACCcLO,W        ;if msb equal then check lsb
NOCHK
    BTFSS   STATUS,C        ;carry set if c>a
    BRA     NOGO
    MOVF    ACCaLO,W        ;c-a into c
    SUBWF   ACCcLO,F
    BTFSS   STATUS,C
    DECF    ACCcHI,F
    MOVF    ACCaHI,W
    SUBWF   ACCcHI,F
    BSF     STATUS,C        ;shift a 1 into b (result)
NOGO
    RLCF    ACCbLO,F
    RLCF    ACCbHI,F

    DECFSZ  temp,F          ; FIM DA DIVISÃO ?
    BRA     DIV             ; NÃO - VOLTA P/ DIV
    RETURN                  ; SIM - RETORNA

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                CONFIGURAÇÕES INICIAIS DE HARDWARE E SOFTWARE          *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  NESTA ROTINA SÃO INICIALIZADAS AS PORTAS DE I/O DO MICROCONTROLADOR E AS
;  CONFIGURAÇÕES DOS REGISTRADORES ESPECIAIS (SFR). A ROTINA INICIALIZA A
;  MÁQUINA E AGUARDA O ESTOURO DO WDT.

CONFIGURACAO
    CLRF    PORTA           ; GARANTE AS SAÍDAS EM ZERO
    CLRF    PORTB
    CLRF    PORTC
    CLRF    PORTD
    CLRF    PORTE

    CLRF    LATA
    CLRF    LATB
    CLRF    LATC
    CLRF    LATD
    CLRF    LATE

    MOVLW   B'11111111'
    MOVWF   TRISA           ; CONFIGURA I/O DO PORTA
```

```
MOVLW   B'11111111'
MOVWF   TRISB           ; CONFIGURA I/O DO PORTB

MOVLW   B'11111111'
MOVWF   TRISC           ; CONFIGURA I/O DO PORTC

MOVLW   B'00000000'
MOVWF   TRISD           ; CONFIGURA I/O DO PORTD

MOVLW   B'00000100'
MOVWF   TRISE           ; CONFIGURA I/O DO PORTE

MOVLW   B'11011000'     ; CONFIGURA TIMER0
MOVWF   T0CON           ; TIMER0 INCREM. PELO CICLO DE MÁQUINA
                        ; TIMER0- 1:1

MOVLW   B'00000000'     ; CONFIGURA INTERRUPÇÕES
MOVWF   INTCON          ; DESABILITADAS TODAS AS INTERRUPÇÕES

MOVLW   B'00000001'     ; CONFIGURA CONVERSOR AD
MOVWF   ADCON2          ; VELOCIDADE -> Fosc/8
                        ; JUSTIFICADO À ESQUERDA
                        ; 8 BITS EM ADRESH E 2 BITS EM ADRESL
                        ; TEMPO DE AQUISIÇÃO DO CAPACITOR
                        ; DEFINIDO POR SOFTWARE

MOVLW   B'00001101'     ; CONFIGURA CONVERSOR AD
MOVWF   ADCON1          ; RA0 E RA1 COMO ANALÓGICO
                        ; RA2, RA3, RA4 E RA5 COMO I/O DIGITAL
                        ; PORTB E PORTE COMO I/O DIGITAL
                        ; Vref+ = VDD (+5V)
                        ; Vref- = GND ( 0V)

MOVLW   B'00000101'     ; CONFIGURA CONVERSOR AD
MOVWF   ADCON0          ; CANAL 1
                        ; MÓDULO LIGADO

;  AS INSTRUÇÕES A SEGUIR FAZEM COM QUE O PROGRAMA TRAVE QUANDO HOUVER UM
;  RESET OU POWER-UP, MAS PASSE DIRETO SE O RESET FOR POR WDT. DESTA FORMA,
;  SEMPRE QUE O PIC É LIGADO, O PROGRAMA TRAVA, AGUARDA UM ESTOURO DE WDT
;  E COMEÇA NOVAMENTE. ISTO EVITA PROBLEMAS NO START-UP DO PIC.

BTFSC   RCON,NOT_TO     ; RESET POR ESTOURO DE WATCHDOG TIMER?
BRA     $               ; NÃO - AGUARDA ESTOURO DO WDT
                        ; SIM

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                       INICIALIZAÇÃO DA RAM                            *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTA ROTINA IRÁ LIMPAR TODA A RAM, DO BANCO 0 ATÉ AO BANCO 15

LFSR    FSR0,0X00       ; APONTA FSR0 PARA ENDEREÇO ZERO

LIMPA_RAM
    CLRF    POSTINC0    ; LIMPA A POSIÇÃO E INCREMENTA FSR0
    CLRWDT              ; LIMPA WATCHDOG TIMER
    MOVLW   0X0F        ; TESTA SE JÁ CHEGOU AO BANCO 15
    SUBWF   FSR0H,W
    BNZ     LIMPA_RAM   ; SE NÃO, CONTINUA LIMPANDO
```

```
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *              CONFIGURAÇÕES INICIAIS DO DISPLAY                          *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; ESTA ROTINA INICIALIZA O DISPLAY P/ COMUNICAÇÃO DE 8 VIAS, DISPLAY PARA 2
; LINHAS, CURSOR APAGADO E DESLOCAMENTO DO CURSOR À DIREITA.

INICIALIZACAO_DISPLAY
    BCF     RS              ; SELECIONA O DISPLAY P/ COMANDOS

    MOVLW   0X30            ; ESCREVE COMANDO 0X30 PARA
    RCALL   ESCREVE         ; INICIALIZAÇÃO

    MOVLW   .3
    RCALL   DELAY_MS        ; DELAY DE 3MS (EXIGIDO PELO DISPLAY)

    MOVLW   0X30            ; ESCREVE COMANDO 0X30 PARA
    RCALL   ESCREVE         ; INICIALIZAÇÃO

    MOVLW   0X30            ; ESCREVE COMANDO 0X30 PARA
    RCALL   ESCREVE         ; INICIALIZAÇÃO

    MOVLW   B'00111000'     ; ESCREVE COMANDO PARA
    RCALL   ESCREVE         ; INTERFACE DE 8 VIAS DE DADOS

    MOVLW   B'00000001'     ; ESCREVE COMANDO PARA
    RCALL   ESCREVE         ; LIMPAR TODO O DISPLAY

    MOVLW   .1
    RCALL   DELAY_MS        ; DELAY DE 1MS

    MOVLW   B'00001100'     ; ESCREVE COMANDO PARA
    RCALL   ESCREVE         ; LIGAR O DISPLAY SEM CURSOR

    MOVLW   B'00000110'     ; ESCREVE COMANDO PARA INCREM.
    RCALL   ESCREVE         ; AUTOMÁTICO À DIREITA

    BSF     RS              ; SELECIONA O DISPLAY P/ DADOS

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *              ROTINA DE ESCRITA DA TELA PRINCIPAL                        *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTA ROTINA ESCREVE A TELA PRINCIPAL DO PROGRAMA, COM AS FRASES:
;  LINHA 1 - " AD Int.  (P2) "
;  LINHA 2 - "        Volts   "

    MOVLW   0X81            ; COMANDO PARA POSICIONAR O CURSOR
    RCALL   ESCREVE         ; LINHA 0 / COLUNA 1

    BSF     RS              ; SELECIONA O DISPLAY P/ DADOS

    MOVLW   'A'             ; COMANDOS PARA ESCREVER AS
    RCALL   ESCREVE         ; LETRAS DE "AD Int.  (P2)"
    MOVLW   '/'
    RCALL   ESCREVE
    MOVLW   'D'
    RCALL   ESCREVE
    MOVLW   ' '
    RCALL   ESCREVE
    MOVLW   'I'
    RCALL   ESCREVE
    MOVLW   'n'
    RCALL   ESCREVE
    MOVLW   't'
    RCALL   ESCREVE
```

```
MOVLW   '.'
RCALL   ESCREVE
MOVLW   ' '
RCALL   ESCREVE
MOVLW   ' '
RCALL   ESCREVE
MOVLW   '('
RCALL   ESCREVE
MOVLW   'P'
RCALL   ESCREVE
MOVLW   '2'
RCALL   ESCREVE
MOVLW   ')'
RCALL   ESCREVE

    BCF     RS              ; SELECIONA O DISPLAY P/ COMANDOS
    MOVLW   0XC7            ; COMANDO PARA POSICIONAR O CURSOR
    RCALL   ESCREVE         ; LINHA 1 / COLUNA 7
    BSF     RS              ; SELECIONA O DISPLAY P/ DADOS

    MOVLW   'V'             ; COMANDOS PARA ESCREVER AS
    RCALL   ESCREVE         ; LETRAS DE "Volts"
    MOVLW   'o'
    RCALL   ESCREVE
    MOVLW   'l'
    RCALL   ESCREVE
    MOVLW   't'
    RCALL   ESCREVE
    MOVLW   's'
    RCALL   ESCREVE

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                          LOOP PRINCIPAL                                 *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  A ROTINA PRINCIPAL FICA CONVERTENDO O CANAL AD, CALCULANDO O VALOR EM
;  VOLTS E MOSTRANDO NO DISPLAY.

LOOP
    CLRWDT                  ; LIMPA WATCHDOG TIMER

    BSF     ADCON0,GO       ; INICIA CONVERSÃO AD
    BTFSC   ADCON0,GO       ; FIM DA CONVERSÃO ?
    BRA     $-.2            ; NÃO - VOLTA 1 INSTRUÇÃO
                            ; SIM
    MOVF    ADRESH,W        ; SALVA VALOR DA CONVERSÃO NO WORK
    MULLW   .50             ; MULTIPLICA VALOR DO AD POR 50

    MOVFF   PRODH,ACCbHI    ; CARREGA ACCb COM O RESULTADO DA MULTIPLICAÇÃO
    MOVFF   PRODL,ACCbLO    ; (NUMERADOR)

    CLRF    ACCaHI          ; CARREGA ACCa COM 255d (FUNDO DE ESCALA DO
    MOVLW   .255            ; CONVERSOR AD - 8 BITS, JUSTIFICADO A ESQUERDA)
    MOVWF   ACCaLO          ; (DENOMINADOR)

    RCALL   D_divF          ; CHAMA ROTINA DE DIVISÃO

    MOVF    ACCbLO,W        ; FAZ O AJUSTE DECIMAL DO RESULTADO DA DIVISÃO
    RCALL   AJUSTE_DECIMAL  ; PARA MOSTRAR NO DISPLAY (LCD)

    BCF     RS              ; SELECIONA O DISPLAY P/ COMANDOS
    MOVLW   0XC3            ; COMANDO PARA POSICIONAR O CURSOR
    RCALL   ESCREVE         ; LINHA 1 / COLUNA 3
    BSF     RS              ; SELECIONA O DISPLAY P/ DADOS
```

```
    MOVF    DEZENA,W
    ADDLW   0X30            ; CONVERTE BCD DA DEZENA EM ASCII
    RCALL   ESCREVE         ; ENVIA AO LCD

    MOVLW   ','
    RCALL   ESCREVE         ; ESCREVE UMA VIRGULA NO LCD

    MOVF    UNIDADE,W
    ADDLW   0X30            ; CONVERTE BCD DA UNIDADE EM ASCII
    RCALL   ESCREVE         ; ENVIA AO LCD

    BRA     LOOP            ; VOLTA PARA LOOP

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                          FIM DO PROGRAMA                              *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

    END                     ; FIM DO PROGRAMA
```

» Dicas e comentários

Inicialmente podemos comentar que toda a estrutura e as rotinas utilizadas para a escrita no LCD são as mesmas já aplicadas no exemplo do Capítulo 6.

Observe também que, conforme foi comentado anteriormente, não foi utilizada nenhuma interrupção nesse programa. Por isso, o programa permanece parado em um pequeno loop enquanto a conversão não termina. Isso é verificado através do bit **ADCON0<GO/DONE>**. Outro ponto interessante é que nenhum delay é especificado antes do início da conversão. Isso não é necessário, pois o sistema analógico é lento (potenciômetro), só utilizamos um canal e, ainda por cima, o resto do loop principal dura mais de 40 µs, garantindo o tempo mínimo entre uma conversão e outra.

A rotina de divisão (16 bits) foi obtida dos Aplication Notes da própria Microchip e pode ser utilizada em outros projetos.

Outra rotina bem interessante que aparece nesse sistema é a de conversão de um número qualquer (limitado entre 0 e 99) em dois dígitos separados, para a composição de números decimais, facilitando a escrita no LCD. Essa rotina devolve os dígitos nas variáveis UNIDADE e DEZENA. Não se esqueça de que, antes de transmitir uma variável decimal para o LCD, deve-se convertê-la em ASCII, bastando para isso somar o valor 30h (Hex).

Para facilitar as contas e não utilizar números fracionários, a conversão em volts é feita considerando 50 no lugar de 5,0. Na hora de enviar para o LCD, é simplesmente colocada uma vírgula entre os dois dígitos.

» Exercícios propostos

Agora que o exemplo foi estudado e esclarecido, aproveite para gerar novos problemas e soluções, seguindo os exercícios propostos:

1. Simular que a entrada analógica é um sensor de temperatura linear que deve marcar de 10 a 80°C.
2. Alterar o exemplo para indicar a tensão entre 0 e 2,50 V, utilizando 10 bits de resolução. Para isso, fazer uso da tensão de referência externa existente na placa proposta (pino RA3).

8

Conversor Analógico-Digital por RC

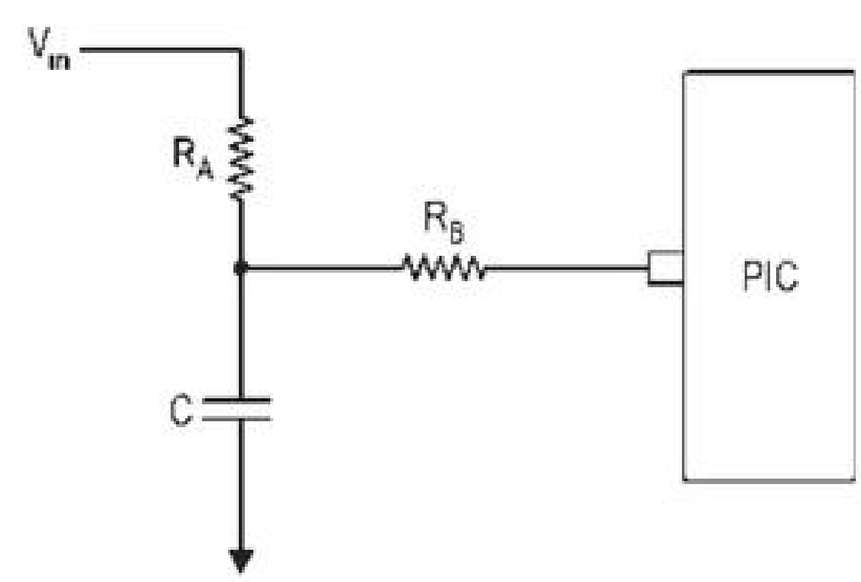
» Introdução

Quando o microcontrolador não dispõe de um CAD interno, é possível utilizarmos conversores externos que efetuam a conversão e disponibilizam o dado convertido através de uma comunicação serial ou paralela.

Mas existem alguns casos em que o custo ou os recursos do projeto não possibilitam a escolha de um PIC com conversor interno e muito menos o uso de um conversor externo. Neste caso, estamos destinados a operar com valores meramente digitais? Nem sempre, pois existe uma solução que, apesar de não ser muito precisa, pode ser a salvação.

» Teoria e recursos do PIC

Às vezes estamos interessados em quantizar apenas alguns níveis de tensão analógica, criando patamares referenciais. Por exemplo, em um sistema alimentado a bateria é interessante sabermos se ela se encontra carregada, normal ou descarregada e, portanto, apenas três níveis comparativos são necessários. O custo de um conversor AD convencional para esse tipo de aplicação não se justifica. Nessas ocasiões pode-se recorrer a um sistema de conversão baseado na carga e descarga de um capacitor. A ideia consiste em medir, através do microcontrolador, o tempo de carga de um capacitor em um circuito RC. Veja o exemplo de hardware colocado ao lado.



Admitindo-se que o pino do PIC está configurado como entrada, o tempo de carga do capacitor **C** está relacionado com o valor de entrada (V_{in}) do resistor R_A e do próprio capacitor C. O resistor R_B não interfere no tempo de carga, pois o pino do PIC está em alta impedância (entrada). Já se o pino do PIC estiver configurado como saída em nível lógico 0, o capacitor tende a descarregar-se pelo resistor R_B e carregar pelo resistor R_A. Porém, vamos admitir que o valor do resistor R_B seja muito menor do que o de R_A e, portanto, nessa configuração, podemos desprezar a carga proveniente do resistor R_A e admitir que o capacitor C apenas se descarrega através de R_B. Em resumo, o capacitor se carrega através de R_A (pino como entrada) e se

descarrega através de R_B (pino como saída em 0), foi que o tempo de carga/descarga depende do próprio valor do capacitor, da tensão de entrada (V_{in}) e do resistor em questão.

Como funciona então a conversão AD?

1. O software deve configurar o pino do PIC como saída em 0.
2. Esperar o tempo de descarrega do capacitor C. Esse tempo deve ser garantido por software, conforme os valores dos componentes utilizados.
3. Configurar o pino como entrada, ou seja, permitir a carga do capacitor.
4. Contar o tempo que o capacitor leva para que o PIC entenda nível lógico 1, ou seja, contar o tempo de carga do capacitor.
5. Repetir o processo para uma nova conversão.

Como os valores do resistor e do capacitor são fixos e conhecidos, o tempo de carga do capacitor é proporcional à tensão de entrada V_{in}.

Admitindo-se que a tensão de entrada V_{in} não varia durante a conversão AD, o modelo matemático aproximado para a curva de descarga do capacitor é

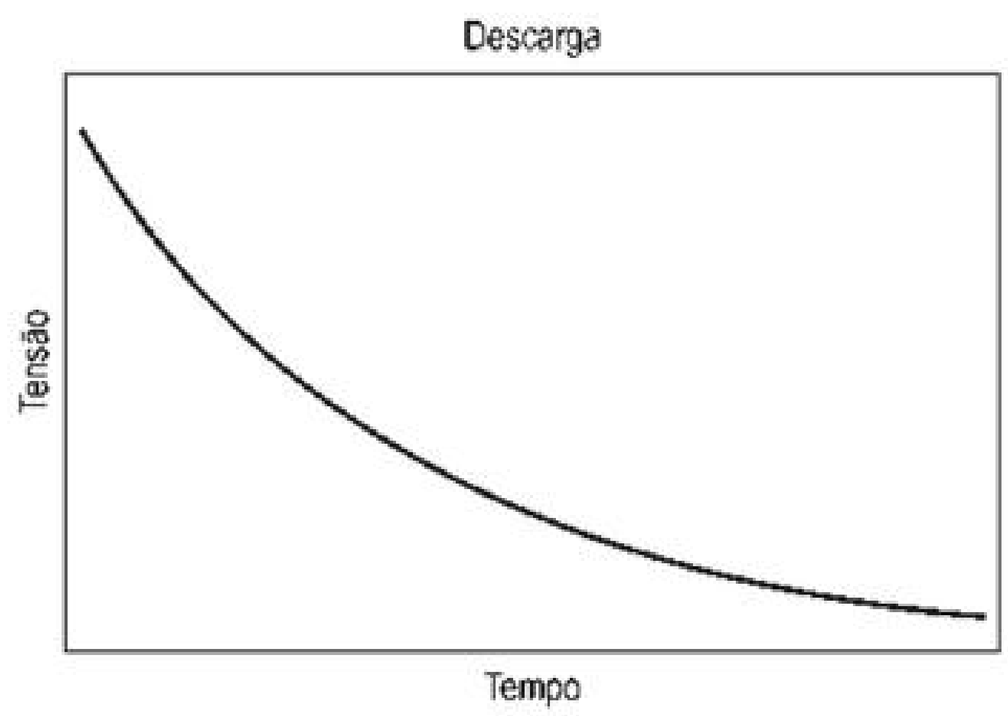
$$V_{cap}(t) = V_{in}\, e^{-t/R_B C},$$

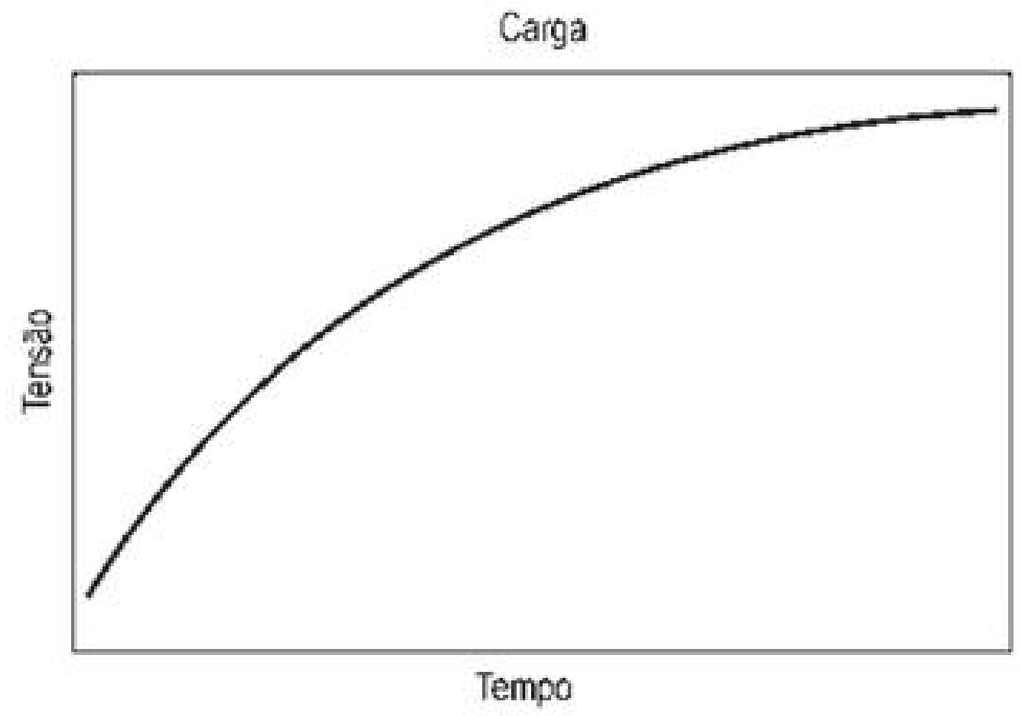
enquanto a curva de carga segue o modelo

$$V_{cap}(t) = V_{in}\left(1 - e^{-t/R_A C}\right).$$

Para a curva de carga anterior, temos o tempo de carga:

$$t = -R_A C\left(\ln\left(1 - \frac{V_{cap}(t)}{V_{in}}\right)\right)$$





Conforme foi comentado no início, esse tipo de conversor não apresenta uma boa precisão, além de possuir uma série de inconvenientes:

- ✓ Nota-se nos gráficos que a tensão não varia linearmente no tempo, portanto esse tipo de conversor AD não é linear.

- ✓ O valor da conversão, ou seja, o tempo de carga do capacitor está sujeito às variações dos componentes envolvidos. Para o caso do resistor, pode-se utilizar resistores de 1%, porém para o capacitor o problema torna-se mais grave, já que geralmente existe muita variação de um lote de componentes para o outro.
- ✓ Normalmente o capacitor é muito suscetível a variações térmicas.
- ✓ A tensão de entrada deve ser suficientemente alta para que o PIC entenda nível lógico 1, por isso esse conversor não funciona no range completo de 0 a 5 V.
- ✓ O valor de tensão necessário para que o PIC entenda nível lógico 1 pode variar em função da pastilha, da tensão da fonte (alimentação do PIC) e do tipo de pino (TTL/ST). Reavalie a teoria do Capítulo 4.

Como dica, podemos sugerir:

- ✓ Utilizar R_B pelo menos dez vezes menor que R_A;
- ✓ Não utilizar capacitores maiores do que 1 μF;
- ✓ Dar preferência ao uso de capacitores de tântalo ou cerâmico;
- ✓ Não discretizar mais do que oito níveis.

» Lógica do exemplo

O exercício proposto para este capítulo mostra no LCD o tempo de carga de um circuito RC disponível na placa do Apêndice C (McLab2). Vamos colocar em prática a teoria vista anteriormente.

O pino de entrada do PIC (**RA1**) é do tipo TTL, portanto o nível mínimo de tensão para indicar nível lógico 1 é de aproximadamente 1,25 V. Vamos admitir que a tensão de entrada varie entre 1,5 V e 5 V. A escolha de 1,5 V e não 1,25 V deve-se ao fato de a curva do capacitor ser exponencial e, portanto, o capacitor tende a carregar-se com uma tensão um pouco abaixo da tensão de entrada. Se a tensão de entrada fosse de 1,25 V, nunca o PIC atingiria nível lógico 1.

Para o circuito da placa temos:

- ✓ $R_A = 4{,}7\ k\Omega$
- ✓ $R_B = 330\ \Omega$
- ✓ $C = 1\ \mu F$

O tempo de carga do capacitor para uma tensão de entrada de 1,5 V, considerando que o PIC entenda nível lógico 1 com uma tensão de entrada de 1,25 V, será então:

$$t = -(4{,}7\ k)(1\ \mu)\left(\ln\left(1 - \frac{(1{,}25)}{(1{,}5)}\right)\right)$$

$$t = 8{,}4\ ms$$

Editora Érica - Desbravando o Microcontrolador PIC18 Recursos Avançados - Daniel R. de Sousa, David José de Souza e Nicolás César Lavinia - 1ª Edição

Já o tempo de carga do capacitor para uma tensão de entrada de 5 V será:

$$t = -(4{,}7\ k)(1\ \mu)\left(\ln\left(1 - \frac{(1{,}25)}{(5)}\right)\right)$$

$$t = 1{,}4\ ms$$

Verifica-se, então, que para o range de tensão de entrada proposto (1,5 V a 5,0 V) o tempo de carga do capacitor varia entre 1,4 ms e 8,4 ms. Admitindo que a rotina que verifica se o capacitor já está carregado tem um tempo de execução de 50 µs e que a variável que armazenará o tempo (em múltiplos de 50 µs) é de 8 bits (256 níveis), temos que o tempo máximo de carga do capacitor não pode ser maior do que:

$$t_{máx} = 256 * 50\ \mu s = 12{,}8\ ms$$

Veja que esse valor está acima dos 8,4 ms calculados, o que garante que mesmo com um tempo de carga lento, o contador da rotina não vai estourar, a não ser no caso em que a tensão de entrada está abaixo da mínima admissível interpretada como nível lógico 1.

O tempo mínimo de carga do capacitor, ou seja, tensão máxima de entrada, foi calculado em 1,4 ms. Como a rotina do software proposto executa uma contagem a cada 50 µs, nesta condição, o valor mínimo medido pelo conversor AD deve ser de:

$$A/D = \frac{1{,}4\ ms}{50\ \mu s} = 28\ d$$

Esses dados estão próximos aos reais e podem ser comprovados no software de exemplo. As diferenças encontradas podem ser explicadas levando-se em conta todos os comentários já discutidos anteriormente.

Além do tempo de carga do capacitor, devemos calcular também o tempo de descarga, uma vez que o ciclo de conversão deve ser iniciado com o capacitor descarregado. Considerando que o capacitor está totalmente carregado, ou seja, com 5 V e que ele será considerado descarregado quando sua tensão for menor do que 50 mV, podemos calcular o tempo máximo de descarga em:

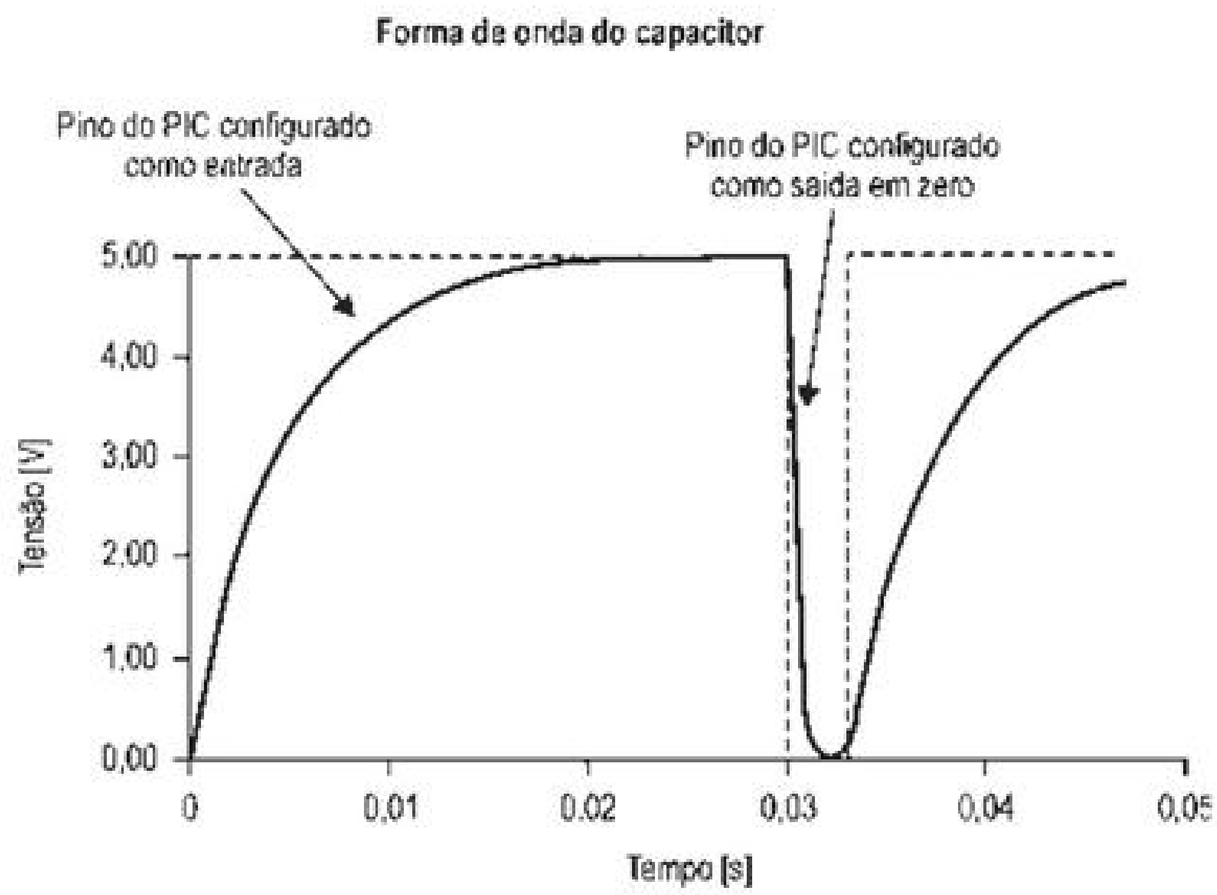
$$t = -R_B C \ln\left(\frac{V_{cap}(t)}{V_{in}}\right)$$

$$t = -(330)(1\ \mu)\ \ln\left(\frac{0{,}05}{5}\right)$$

$$t = 1{,}5\ ms$$

Na realidade, o capacitor nunca se descarregará tanto (50 mV), pois o resistor R_A (ligado à tensão de entrada) forma em conjunto com o resistor R_B (ligado ao terra através do PIC) um divisor de tensão que não permite a descarga do capacitor até uma tensão tão baixa. De qualquer forma, o cálculo deve ser utilizado para estimar a ordem de grandeza do tempo de descarga do capacitor.

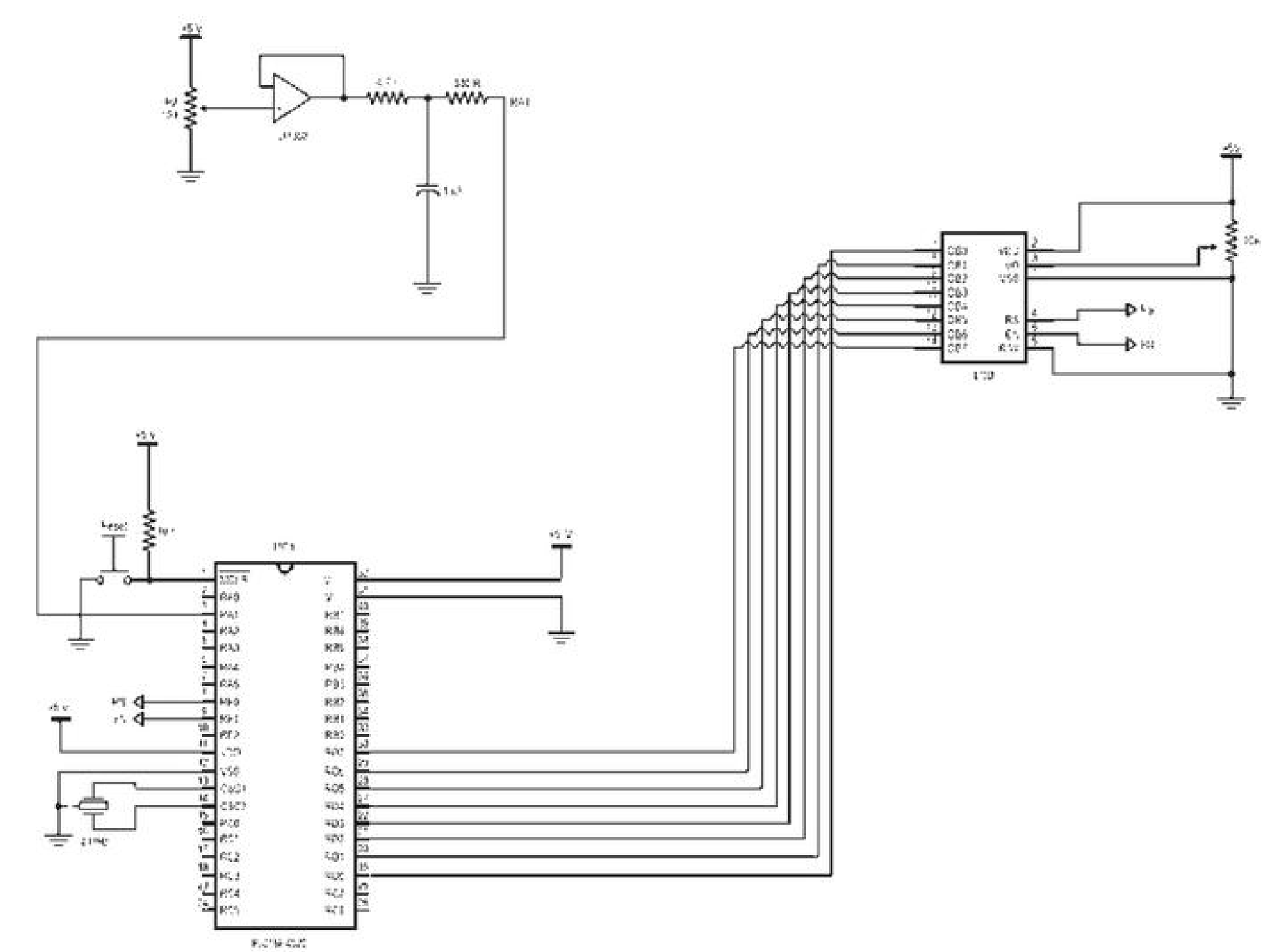
Para efeitos ilustrativos, segue o exemplo plotado da forma de onda no capacitor para uma tensão de entrada de 5,0 V:



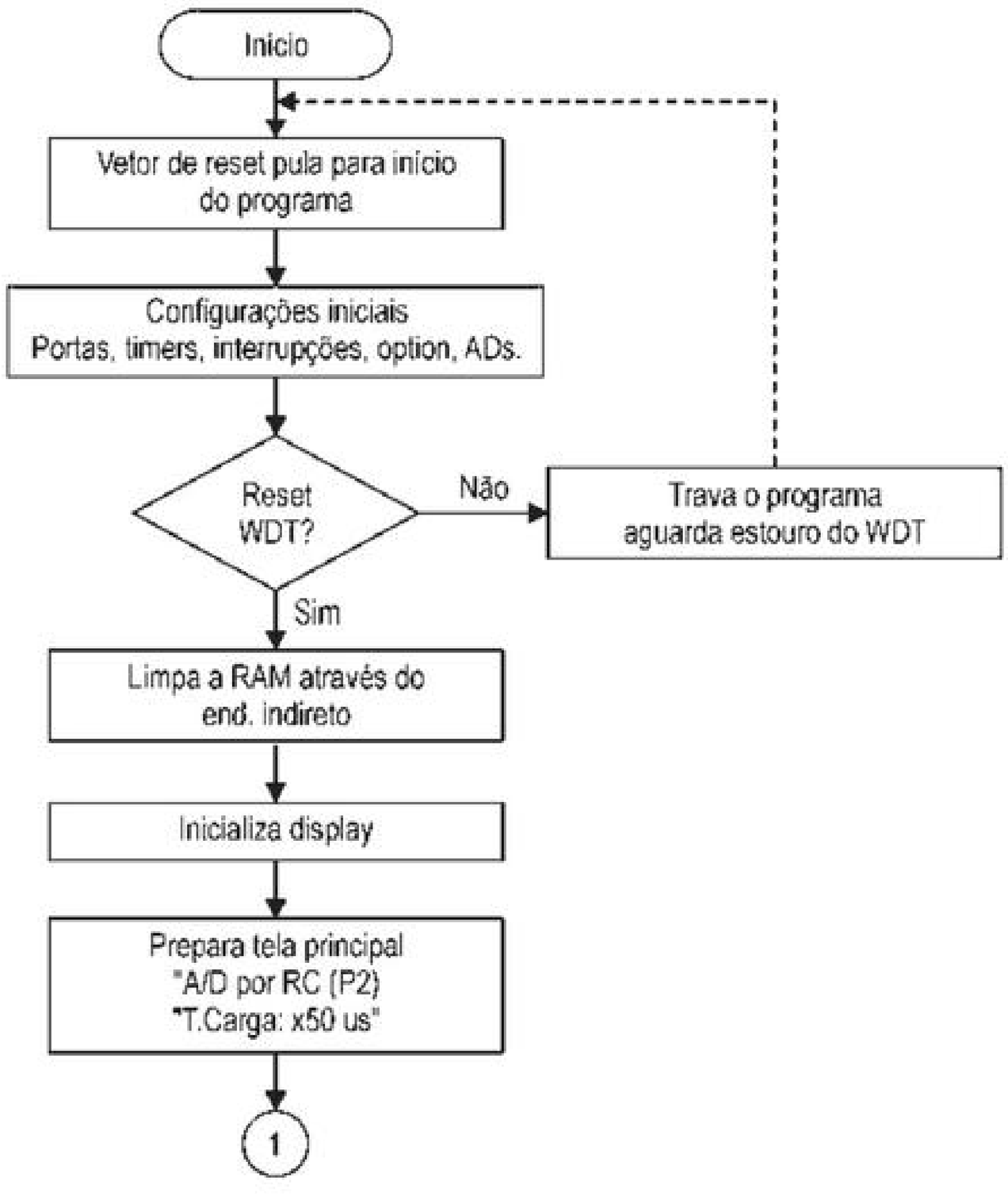
Levando-se em conta os cálculos apresentados, no software de exemplo foi criado um loop de 50 µs para a contagem do tempo de carga do capacitor. O valor do contador foi armazenado na variável denominada CONTADOR_AD. Após a carga do capacitor, o valor desse contador é mostrado no LCD. Para garantir a descarga do capacitor foi deixado um delay de 3 ms.

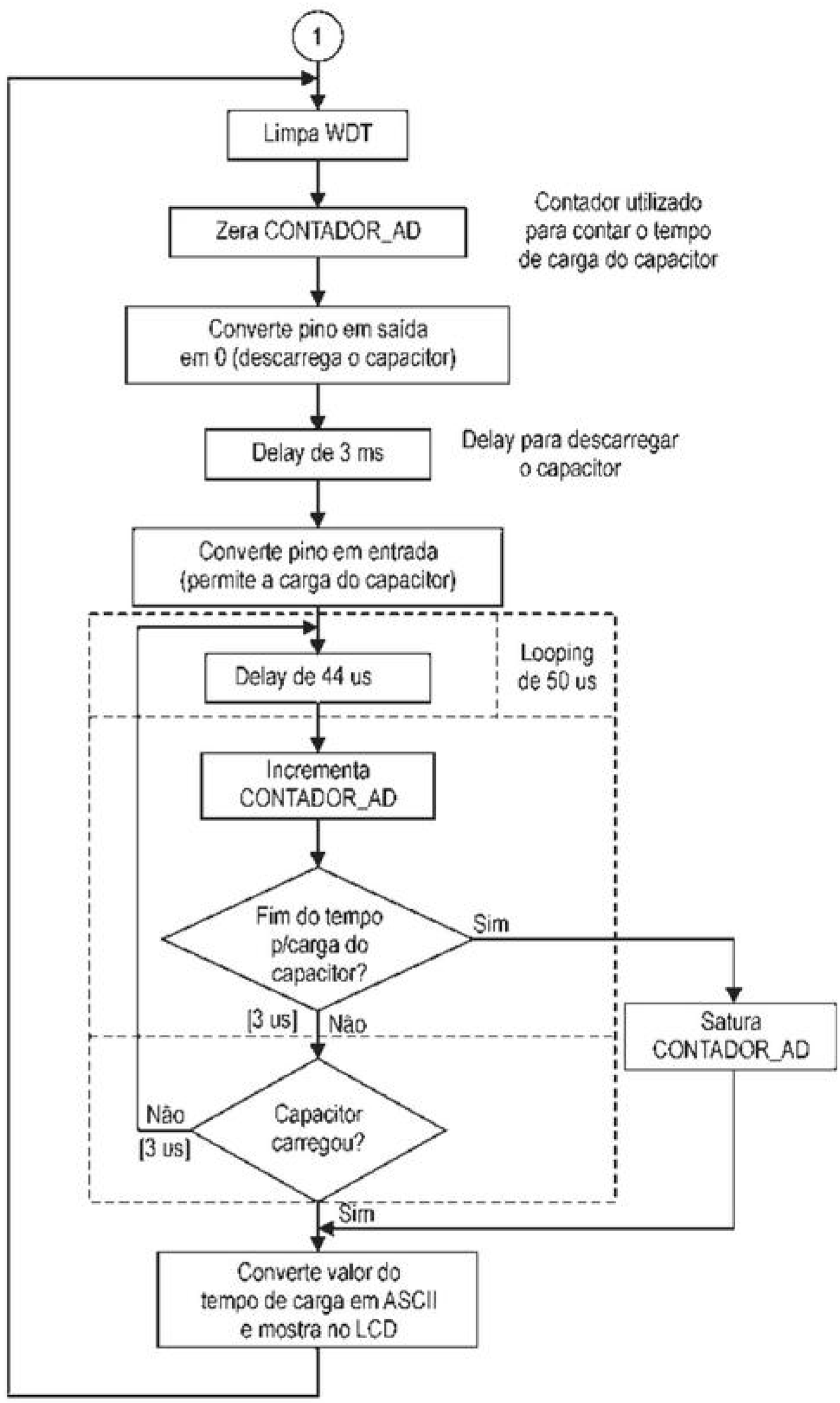
Nenhuma interrupção foi empregada neste exemplo.

» Esquema elétrico



» Fluxograma


Início
Vetor de reset pula para início do programa
Configurações iniciais
Portas, timers, interrupções, option, ADs.
Reset WDT?
Não
Trava o programa aguarda estouro do WDT
Sim
Limpa a RAM através do end. indireto
Inicializa display
Prepara tela principal
"A/D por RC (P2)
"T.Carga: x50 us"
1



» Código

```
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *              DESBRAVANDO O PIC18 - RECURSOS AVANÇADOS                 *
; *                              EXEMPLO 5                                *
; *                                                                       *
; * DANIEL RODRIGUES DE SOUSA, DAVID JOSÉ DE SOUZA e  NICOLÁS CÉSAR LAVINIA *
; *                                                                       *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

```
; *     DATA : 03/08/2009                                                *
; *   VERSÃO : 1.0                                                       *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                          DESCRIÇÃO GERAL                           *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTE EXEMPLO FOI ELABORADO PARA EXPLICAR O FUNCIONAMENTO DE UM TIPO DE
;  CONVERSOR AD FUNDAMENTADO NO TEMPO DE CARGA DE UM CAPACITOR. O TEMPO DE
;  CARGA DO CAPACITOR É MOSTRADO NO LCD E É INVERSAMENTE PROPORCIONAL À
;  TENSÃO APLICADA ATRAVÉS DO POTENCIÔMETRO P2.
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                       CONFIGURAÇÕES PARA GRAVAÇÃO                  *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
    CONFIG  OSC = XT, FCMEN = OFF, IESO = OFF, PWRT = ON, BOREN = ON, BORV = 0
    CONFIG  WDT = ON, WDTPS = 128, MCLRE = ON, LPT1OSC = OFF, PBADEN = OFF
    CONFIG  CCP2MX = PORTC, STVREN = ON, LVP = OFF, DEBUG = OFF, XINST = OFF
    CONFIG  CP0 = OFF, CP1 = OFF, CP2 = OFF, CP3 = OFF, CPB = OFF, CPD = OFF
    CONFIG  WRT0 = OFF, WRT1 = OFF, WRT2 = OFF, WRT3 = OFF, WRTB = OFF
    CONFIG  WRTC = OFF, WRTD = OFF, EBTR0 = OFF, EBTR1 = OFF, EBTR2 = OFF
    CONFIG  EBTR3 = OFF, EBTRB = OFF

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                        DEFINIÇÃO DAS VARIÁVEIS                     *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
    CBLOCK  0X00            ; BANK 0 --> 0X00..0X5F --> ACCESS BANK

        TEMPO1
        TEMPO0              ; CONTADORES P/ DELAY

        CONTADOR_AD         ; CONTADOR PARA CONVERSOR AD

        AUX                 ; REGISTRADOR AUXILIAR DE USO GERAL

    ENDC

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *              DEFINIÇÃO DAS VARIÁVEIS INTERNAS DO PIC               *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  O ARQUIVO DE DEFINIÇÕES DO PIC UTILIZADO DEVE SER REFERENCIADO PARA QUE
;  OS NOMES DEFINIDOS PELA MICROCHIP POSSAM SER UTILIZADOS, SEM A NECESSIDADE
;  DE REDIGITAÇÃO.

    #INCLUDE <P18F4520.INC> ; MICROCONTROLADOR UTILIZADO

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                          CONSTANTES INTERNAS                       *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  A DEFINIÇÃO DE CONSTANTES FACILITA A PROGRAMAÇÃO E A MANUTENÇÃO.

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                     DECLARAÇÃO DOS FLAGs DE SOFTWARE               *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  A DEFINIÇÃO DE FLAGs AJUDA NA PROGRAMAÇÃO E ECONOMIZA MEMÓRIA RAM.

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                                ENTRADAS                            *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  AS ENTRADAS DEVEM SER ASSOCIADAS A NOMES PARA FACILITAR A PROGRAMAÇÃO E
;  FUTURAS ALTERAÇÕES DO HARDWARE.

#DEFINE PINO_AD TRISA,1     ; PINO P/ LEITURA DO RC
                            ; 0 -> FORÇA A DESCARGA DO CAPACITOR
```

```
                                ; 1 -> LIBERA A CARGA DO CAPACITOR

#DEFINE CAD     PORTA,1         ; PINO P/ LEITURA DO CONV. AD
                                ; 0 -> CAPACITOR DESCARREGADO
                                ; 1 -> CAPACITOR CARREGADO

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                                SAÍDAS                                *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  AS SAÍDAS DEVEM SER ASSOCIADAS A NOMES PARA FACILITAR A PROGRAMAÇÃO E
;  FUTURAS ALTERAÇÕES DO HARDWARE.

#DEFINE DISPLAY     LATD            ; BARRAMENTO DE DADOS DO DISPLAY

#DEFINE RS          LATE,0          ; INDICA P/ O DISPLAY UM DADO OU COMANDO
                                    ; 1 -> DADO
                                    ; 0 -> COMANDO

#DEFINE ENABLE      LATE,1          ; SINAL DE ENABLE P/ DISPLAY
                                    ; ATIVO NA BORDA DE DESCIDA

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                    VETOR DE RESET DO MICROCONTROLADOR                 *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  POSIÇÃO INICIAL PARA EXECUÇÃO DO PROGRAMA

    ORG     0X0000          ; ENDEREÇO DO VETOR DE RESET
    BRA     CONFIGURACAO    ; PULA PARA CONFIG DEVIDO A REGIÃO
                            ; DESTINADA AS ROTINAS SEGUINTES

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                     ROTINA DE DELAY (DE 1MS ATÉ 256MS)                *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTA É UMA ROTINA DE DELAY VARIÁVEL, COM DURAÇÃO DE 1MS X O VALOR PASSADO
;  EM WORK (W).

DELAY_MS
    MOVWF   TEMPO1          ; CARREGA TEMPO1 (UNIDADES DE MS)
    MOVLW   .250
    MOVWF   TEMPO0          ; CARREGA TEMPO0 (P/ CONTAR 1MS)

    CLRWDT                  ; LIMPA WDT (PERDE TEMPO)
    DECFSZ  TEMPO0,F        ; FIM DE TEMPO0 ?
    BRA     $-.4            ; NÃO - VOLTA 2 INSTRUÇÕES
                            ; SIM - PASSOU-SE 1ms
    DECFSZ  TEMPO1,F        ; FIM DE TEMPO1 ?
    BRA     $-.12           ; NÃO - VOLTA 6 INSTRUÇÕES
    RETURN                  ; SIM - RETORNA

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                ROTINA DE ESCRITA DE UM CARACTERE NO DISPLAY           *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTA ROTINA ENVIA UM CARACTERE PARA O MÓDULO DE LCD. O CARACTERE A SER
;  ESCRITO DEVE SER COLOCADO EM WORK (W) ANTES DE CHAMAR A ROTINA.

ESCREVE
    MOVWF   DISPLAY         ; ATUALIZA DISPLAY (PORTD)
    NOP                     ; PERDE 1US PARA ESTABILIZAÇÃO
    BSF     ENABLE          ; ENVIA UM PULSO DE ENABLE AO DISPLAY
    BRA     $+.2            ; .
    BCF     ENABLE          ; .

    MOVLW   .1
    RCALL   DELAY_MS        ; DELAY DE 1ms
```

```
    RETURN                      ; RETORNA

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                CONFIGURAÇÕES INICIAIS DE HARDWARE E SOFTWARE          *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  NESTA ROTINA SÃO INICIALIZADAS AS PORTAS DE I/O DO MICROCONTROLADOR E AS
;  CONFIGURAÇÕES DOS REGISTRADORES ESPECIAIS (SFR). A ROTINA INICIALIZA AS
;  VARIÁVEIS DE RAM E AGUARDA O ESTOURO DO WDT.

CONFIGURACAO
    CLRF    PORTA               ; GARANTE AS SAÍDAS EM ZERO
    CLRF    PORTB
    CLRF    PORTC
    CLRF    PORTD
    CLRF    PORTE

    CLRF    LATA
    CLRF    LATB
    CLRF    LATC
    CLRF    LATD
    CLRF    LATE

    MOVLW   B'11111111'
    MOVWF   TRISA               ; CONFIGURA I/O DO PORTA

    MOVLW   B'11111111'
    MOVWF   TRISB               ; CONFIGURA I/O DO PORTB

    MOVLW   B'11111111'
    MOVWF   TRISC               ; CONFIGURA I/O DO PORTC

    MOVLW   B'00000000'
    MOVWF   TRISD               ; CONFIGURA I/O DO PORTD

    MOVLW   B'00000100'
    MOVWF   TRISE               ; CONFIGURA I/O DO PORTE

    MOVLW   B'11011000'         ; CONFIGURA TIMER0
    MOVWF   T0CON               ; TIMER0 INCREM. PELO CICLO DE MÁQUINA
                                ; TIMER0- 1:1

    MOVLW   B'00000000'         ; CONFIGURA INTERRUPÇÕES
    MOVWF   INTCON              ; DESABILITADAS TODAS AS INTERRUPÇÕES

    MOVLW   B'00001111'         ; CONFIGURA CONVERSOR AD
    MOVWF   ADCON1              ; CONFIGURA PORTA, PORTB E PORTE COMO I/O DIGITAL

;  AS INSTRUÇÕES A SEGUIR FAZEM COM QUE O PROGRAMA TRAVE QUANDO HOUVER UM
;  RESET OU POWER-UP, MAS PASSE DIRETO SE O RESET FOR POR WDT. DESTA FORMA,
;  SEMPRE QUE O PIC É LIGADO, O PROGRAMA TRAVA, AGUARDA UM ESTOURO DE WDT
;  E COMEÇA NOVAMENTE. ISTO EVITA PROBLEMAS NO START-UP DO PIC.

    BTFSC   RCON,NOT_TO         ; RESET POR ESTOURO DE WATCHDOG TIMER ?
    BRA     $                   ; NÃO - AGUARDA ESTOURO DO WDT
                                ; SIM

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                          INICIALIZAÇÃO DA RAM                         *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTA ROTINA IRÁ LIMPAR TODA A RAM, DO BANCO 0 ATÉ O BANCO 15

    LFSR    FSR0,0X00           ; APONTA FSR0 PARA ENDEREÇO ZERO

LIMPA_RAM
```

```
    CLRF    POSTINC0          ; LIMPA A POSIÇÃO E INCREMENTA FSR0
    CLRWDT                    ; LIMPA WATCHDOG TIMER
    MOVLW   0X0F              ; TESTA SE JÁ CHEGOU AO BANCO 15
    SUBWF   FSR0H,W
    BNZ     LIMPA_RAM         ; SE NÃO, CONTINUA LIMPANDO

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                    CONFIGURAÇÕES INICIAIS DO DISPLAY                  *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; ESTA ROTINA INICIALIZA O DISPLAY P/ COMUNICAÇÃO DE 8 VIAS, DISPLAY PARA 2
; LINHAS, CURSOR APAGADO E DESLOCAMENTO DO CURSOR À DIREITA.

INICIALIZACAO_DISPLAY
    BCF     RS                ; SELECIONA O DISPLAY P/ COMANDOS

    MOVLW   0X30              ; ESCREVE COMANDO 0X30 PARA
    RCALL   ESCREVE           ; INICIALIZAÇÃO

    MOVLW   .3
    RCALL   DELAY_MS          ; DELAY DE 3MS (EXIGIDO PELO DISPLAY)

    MOVLW   0X30              ; ESCREVE COMANDO 0X30 PARA
    RCALL   ESCREVE           ; INICIALIZAÇÃO

    MOVLW   0X30              ; ESCREVE COMANDO 0X30 PARA
    RCALL   ESCREVE           ; INICIALIZAÇÃO

    MOVLW   B'00111000'       ; ESCREVE COMANDO PARA
    RCALL   ESCREVE           ; INTERFACE DE 8 VIAS DE DADOS

    MOVLW   B'00000001'       ; ESCREVE COMANDO PARA
    RCALL   ESCREVE           ; LIMPAR TODO O DISPLAY

    MOVLW   .1
    RCALL   DELAY_MS          ; DELAY DE 1MS

    MOVLW   B'00001100'       ; ESCREVE COMANDO PARA
    RCALL   ESCREVE           ; LIGAR O DISPLAY SEM CURSOR

    MOVLW   B'00000110'       ; ESCREVE COMANDO PARA INCREM.
    RCALL   ESCREVE           ; AUTOMÁTICO À DIREITA

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                ROTINA DE ESCRITA DA TELA PRINCIPAL                    *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTA ROTINA ESCREVE A TELA PRINCIPAL DO PROGRAMA, COM AS FRASES:
;  LINHA 1 - "AD por RC  (P2)"
;  LINHA 2 - "T.CARGA:   x50us"

    MOVLW   0X80              ; COMANDO PARA POSICIONAR O CURSOR
    RCALL   ESCREVE           ; LINHA 0 / COLUNA 0

    BSF     RS                ; SELECIONA O DISPLAY P/ DADOS

    MOVLW   'A'               ; COMANDOS PARA ESCREVER AS
    RCALL   ESCREVE           ; LETRAS DE "AD por RC (P2)"
    MOVLW   '/'
    RCALL   ESCREVE
    MOVLW   'D'
    RCALL   ESCREVE
    MOVLW   ' '
    RCALL   ESCREVE
    MOVLW   'p'
    RCALL   ESCREVE
```

```
MOVLW   'o'
RCALL   ESCREVE
MOVLW   'r'
RCALL   ESCREVE
MOVLW   ' '
RCALL   ESCREVE
MOVLW   'R'
RCALL   ESCREVE
MOVLW   'C'
RCALL   ESCREVE
MOVLW   ' '
RCALL   ESCREVE
MOVLW   ' '
RCALL   ESCREVE
MOVLW   '('
RCALL   ESCREVE
MOVLW   'P'
RCALL   ESCREVE
MOVLW   '2'
RCALL   ESCREVE
MOVLW   ')'
RCALL   ESCREVE

BCF     RS              ; SELECIONA O DISPLAY P/ COMANDOS

MOVLW   0XC0            ; COMANDO PARA POSICIONAR O CURSOR
RCALL   ESCREVE         ; LINHA 1 / COLUNA 0

BSF     RS              ; SELECIONA O DISPLAY P/ DADOS

MOVLW   'T'             ; COMANDOS PARA ESCREVER AS
RCALL   ESCREVE         ; LETRAS DE "T.CARGA:   x50us"
MOVLW   '.'
RCALL   ESCREVE
MOVLW   'C'
RCALL   ESCREVE
MOVLW   'A'
RCALL   ESCREVE
MOVLW   'R'
RCALL   ESCREVE
MOVLW   'G'
RCALL   ESCREVE
MOVLW   'A'
RCALL   ESCREVE
MOVLW   ':'
RCALL   ESCREVE
MOVLW   ' '
RCALL   ESCREVE
MOVLW   ' '
RCALL   ESCREVE
MOVLW   ' '
RCALL   ESCREVE
MOVLW   'x'
RCALL   ESCREVE
MOVLW   '5'
RCALL   ESCREVE
MOVLW   '0'
RCALL   ESCREVE
MOVLW   'u'
RCALL   ESCREVE
MOVLW   's'
RCALL   ESCREVE
```

```
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *      ROTINA PARA DESCARREGAR O CAPACITOR DE LEITURA DO CONVERSOR AD     *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; ESTA ROTINA CONVERTE O PINO DO MICROCONTROLADOR EM SAÍDA COM NÍVEL LÓGICO 0
; E AGUARDA QUE O CAPACITOR SE DESCARREGUE. EM SEGUIDA O PINO É CONVERTIDO
; NOVAMENTE EM ENTRADA PARA PERMITIR QUE O CAPACITOR SE CARREGUE.

DESCARGA_CAPACITOR
    CLRWDT                  ; LIMPA WATCHDOG TIMER

    CLRF    CONTADOR_AD     ; ZERA O CONTADOR DE TEMPO DE CARGA
                            ; DO CAPACITOR
    BCF     PINO_AD         ; TRANSFORMA PINO EM SAIDA
    BCF     CAD             ; DESCARREGA O CAPACITOR

    MOVLW   .3              ; CHAMA ROTINA DE DELAY (3ms)
    RCALL   DELAY_MS        ; TEMPO NECESSÁRIO P/ DESCARGA
                            ; DO CAPACITOR
    BSF     PINO_AD         ; TRANSFORMA PINO EM ENTRADA

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                    LOOP P/ ESPERAR CARGA DO CAPACITOR                   *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; O TEMPO CONTA O TEMPO QUE O CAPACITOR LEVA PARA ATINGIR UM NÍVEL DE TENSÃO
; SUFICIENTE PARA QUE O MICROCONTROLADOR ENTENDA NÍVEL LÓGICO 1 NA ENTRADA TTL
; DO PINO RA1. CASO O CAPACITOR NUNCA SE DEMORE MAIS DO QUE 256 CICLOS DESTE
; LOOP, A ROTINA DESVIA PARA UMA ROTINA DE SATURAÇÃO.
; O LOOP DA ROTINA É DE 50us (CRISTAL DE 4MHz).

LOOP_CAD
    NOP                     ; [1us]
    MOVLW   .14             ; [2us]
    MOVWF   AUX             ; [3us] - CARREGA AUX COM 14d
    DECFSZ  AUX,F
    BRA     $-.2            ; [4us] A [44us] - DELAY

    INCFSZ  CONTADOR_AD,F   ; INCREM. CONTADOR E VERIFICA ESTOURO
    BRA     $+.4            ; NÃO HOUVE ESTOURO - PULA 1 INSTRUÇÃO
    BRA     SATURACAO       ; HOUVE ESTOURO - PULA P/ SATURAÇÃO

    BTFSS   CAD             ; CAPACITOR JÁ CARREGOU ?
    BRA     LOOP_CAD        ; NÃO - VOLTA P/ LOOP_CAD
    BRA     MOSTRA_CONTADOR ; SIM - MOSTRA TEMPO DE CARGA

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *              MOSTRA O TEMPO DE CARGA DO CAPACITOR NO LCD                *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; ESTA ROTINA MOSTRA O TEMPO DE CARGA DO CAPACITOR EM HEXADECIMAL NO LCD.
; CASO O CAPACITOR NÃO TENHA SE CARREGADO, A ROTINA DE SATURAÇÃO GARANTE
; UM VALOR MÁXIMO PARA O TEMPO DE CARGA (0xFF).

SATURACAO
    MOVLW   0XFF
    MOVWF   CONTADOR_AD     ; SATURA O CONTADOR
                            ; (CAPACITOR NÃO CARREGOU)
MOSTRA_CONTADOR
    MOVLW   0XC9            ; COMANDO PARA POSICIONAR O CURSOR
    RCALL   ESCREVE         ; LINHA 1 / COLUNA 9

    BSF     RS              ; SELECIONA O DISPLAY P/ DADOS

    SWAPF   CONTADOR_AD,W   ; INVERTE NIBLE DO CONTADOR_AD
    ANDLW   B'00001111'     ; MASCARA BITS MAIS SIGNIFICATIVOS
    MOVWF   AUX             ; SALVA EM AUXILIAR
```

```
MOVLW   0X0A
SUBWF   AUX,W           ; AUX - 10d (ATUALIZA FLAG DE CARRY)
MOVLW   0X30            ; CARREGA WORK COM 30h
BTFSC   STATUS,C        ; RESULTADO E POSITIVO? (É UMA LETRA?)
MOVLW   0X37            ; SIM - CARREGA WORK COM 37h
                        ; NÃO - WORK FICA COM 30h (NÚMERO)
ADDWF   AUX,W           ; SOMA O WORK AO AUXILIAR
                        ; (CONVERSÃO ASCII)
RCALL   ESCREVE         ; MOSTRA NO DISPLAY

MOVF    CONTADOR_AD,W   ; CARREGA NO WORK O CONTADOR_AD
ANDLW   B'00001111'     ; MASCARA BITS MAIS SIGNIFICATIVOS
MOVWF   AUX             ; SALVA EM AUXILIAR

MOVLW   0X0A
SUBWF   AUX,W           ; AUX - 10d (ATUALIZA FLAG DE CARRY)
MOVLW   0X30            ; CARREGA WORK COM 30h
BTFSC   STATUS,C        ; RESULTADO E POSITIVO? (É UMA LETRA?)
MOVLW   0X37            ; SIM - CARREGA WORK COM 37h
                        ; NÃO - WORK FICA COM 30h (NÚMERO)
ADDWF   AUX,W           ; SOMA O WORK AO AUXILIAR
                        ; (CONVERSÃO ASCII)
RCALL   ESCREVE         ; MOSTRA NO DISPLAY

BCF     RS              ; SELECIONA O DISPLAY P/ COMANDOS

BRA     DESCARGA_CAPACITOR  ; VOLTA P/ DESCARREGAR O CAPACITOR

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                          FIM DO PROGRAMA                              *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

END                     ; FIM DO PROGRAMA
```

» Dicas e comentários

Observe que foi feita uma verificação para garantir que, se o tempo de carga do capacitor ultrapassar o limite da variável CONTADOR_AD, essa variável terá seu valor assegurado em 0xFF através da rotina SATURACAO.

Toda a estrutura e rotinas para escrita no LCD são as mesmas utilizadas no exemplo do Capítulo 6.

» Exercícios propostos

Agora que o exemplo foi estudado e esclarecido, aproveite para gerar novos problemas e soluções, seguindo os exercícios propostos:

1. Seguindo o exemplo dado no começo do capítulo em relação ao monitoramento de uma bateria, estabelecer três níveis de comparação e alterar a tela do LCD para mostrar as mensagens de bateria Fraca, Normal e Carregada.

2. A placa proposta (Apêndice C) possui um outro RC ligado ao PIC, só que com a tensão constante e o R variável. Trata-se dos jumpers A, B e C. A combinação desses jumpers cria sete níveis de resistência aplicados ao RC que está ligado no pino **RA4**. Alterar o programa para ler esses jumps e indicar seus estados no LCD. Os detalhes do esquema de ligação e valores dos componentes podem ser encontrados no Apêndice C.

9

Os Módulos CCP (Capture/Compare/PWM)

» Introdução

Vamos estudar agora os módulos denominados CCPs, cujo nome é originado dos três tipos de recursos oferecidos por eles: Capture, Compare e PWM. Cada um desses recursos é empregado para uma finalidade diferente, que será conhecida a partir de agora.

» Teoria e recursos do PIC

O PIC18F4520 possui dois módulos de CCP, denominados CCP1 e CCP2, que são praticamente idênticos e, por isso, serão explicados ao mesmo tempo, sendo feitos comentários necessários quando houver algum tipo de diferença. Para facilitar o entendimento, cada um dos recursos (Capture, Compare e PWM) será descrito separadamente.

Para começar, é bom esclarecermos a nomenclatura padrão que será utilizada para descrevermos características comuns aos dois módulos existentes:

Padrão	CCP1	CCP2	Descrição
CCPxCON	CCP1CON	CCP2CON	Registrador de configuração
CCPRxH	CCPR1H	CCPR2H	Parte alta do valor de controle
CCPRxL	CCPR1L	CCPR2L	Parte baixa do valor de controle
CCPx	CCP1 (RC2)	CCP2 (RC1)	Pino relacionado

Outro dado interessante que devemos informar neste momento é quanto ao uso dos dois módulos conjuntamente. Como eles utilizam recursos compartilhados para suas bases de tempo (Timer 1 / Timer 2 e Timer 3), podem existir algumas limitações ou conflitos, conforme a combinação de recursos desejada:

Modo	Base de Tempo
Capture	Timer 1 / Timer 3
Compare	Timer 1 / Timer 3
PWM	Timer 2

Recursos Desejados		Observações
Capture	Capture	Sem conflitos, entretanto ambos utilizam a mesma base de tempo (Timer 1 / Timer 3) e, por isso, são sincronizados.
Capture	Compare	Caso o Compare esteja configurado para zerar o Timer 1 / Timer 3, isso pode acarretar conflito com o outro modo.
Compare	Compare	Caso o Compare esteja configurado para zerar o Timer 1 / Timer 3, isso pode acarretar conflito com o outro modo.
PWM	PWM	Ambos os PWM têm a mesma frequência e são sincronizados, devido ao uso da mesma base de tempo. Os Duty Cycles possuem controles independentes.
PWM	Capture	Ambos os modos são completamente independentes.
PWM	Compare	Ambos os modos são completamente independentes.

Note que no caso do Capture e Compare o conflito pode existir quando é selecionada uma mesma base de tempo para CCP1 e CCP2. A seleção da base de tempo para os modos Capture e Compare é feita nos bits **T3CON<T3CCP2>** e **T3CON<T3CCP1>**:

T3CCP2:T3CCP1		Base de Tempo CCP1	Base de Tempo CCP2
0	0	Timer 1	Timer 1
0	1	Timer 1	Timer 3
1	x	Timer 3	Timer 3

Modo Capture

Esse módulo tem como objetivo a contagem de tempo entre dois eventos ocorridos em um dos pinos CCPx. Para isso será utilizado como base de tempo o Timer 1 / Timer 3 e no momento do evento seu valor será capturado (daí o nome Capture). Como Timer 1 / Timer 3 são de 16 bits, a captura será feita em dois registradores: **CCPRxH** e **CCPRxL**.

Com esse recurso é possível então criar um periodímetro, contando o tempo gasto, por exemplo, entre duas bordas de subida da onda ligada ao pino **CCPx**. É importante observarmos, entretanto, que a captura do valor de Timer 1 / Timer 3 não reseta o timer, e por isso, para definirmos o tempo real entre duas leituras é necessária uma conta de subtração entre a última leitura e a anterior. Essa conta deve ser implementada pelo software.

Vejamos, então, como configurar o Capture para que possamos utilizá-lo corretamente.

O modo Capture opera com os pinos como entrada, mas essa configuração não é feita automaticamente. Por isso, antes de mais nada configure, através do **TRISC**, o pino **CCPx** como entrada. Caso este esteja configurado como saída, operações de escrita nesse pino podem ser consideradas como mudança de borda, ativando a captura.

Quanto ao Timer 1 / Timer 3, que será usado como base de tempo para esse modo, ele não pode estar configurado para operação assíncrona, isto é, confirme a condição **TxCON</TxSYNC>=0** (sendo **x** o timer selecionado).

O Capture possui também quatro diferentes configurações (fora o desligamento), que podem ser escolhidas através de **CCPxCON<CCPxM3:CCPxM0>**:

CCPxM3: CCPxM0	Descrição
0000	CCPx desligado
0100	Capture ligado para borda de descida Prescale de 1:1
0101	Capture ligado para borda de subida

CCPxM3: CCPxM0	Descrição
	Prescale de 1:1
0110	Capture ligado para borda de subida Prescale de 1:4
0111	Capture ligado para borda de subida Prescale de 1:16
Observação: As demais opções de configuração dizem respeito aos outros modos (Compare/PWM).	

As diferenças básicas entre essas configurações dizem respeito à borda que vai gerar o evento e ao prescaler adotado, que é um contador interno (não acessível) de bordas. Por exemplo, caso seja escolhida a opção 0111, a captura do timer acontece a cada 16 bordas de subida. A finalidade desse prescaler é o aumento da precisão do sistema. Quando optamos por trabalhar com prescaler de 1:1, o erro máximo em um período é de um ciclo de máquina (T_{CY}). Quando aumentamos o prescaler, esse erro diminui proporcionalmente, e temos para 1:16 um erro máximo de $T_{CY}/16$.

Toda vez que o evento de Capture acontecer, o flag **CCPxIF** será ativado, e caso essa interrupção esteja ligada, ela vai acontecer. Esta é uma maneira fácil de implementarmos a conta de subtração dos tempos absolutos para chegarmos ao período correto. Não se esqueça de que, quando uma nova captura acontece, ela é gravada em **CCPRxH** e **CCPRxL**, sobrescrevendo os valores anteriores.

Para alterar as opções de configuração do modo, alguns cuidados devem ser tomados. Como o prescaler é um contador interno que não pode ser zerado manualmente, a alteração de configuração pode gerar uma interrupção. Uma maneira de evitarmos isso seria o desligamento da interrupção (**CCPxIE**) antes dessa operação. Outra maneira, mais prática, é desligarmos o modo Capture (limpando o registrador **CCPxCON**). Isso reseta o modo, limpando também o contador de prescaler. Depois, basta escolhermos a nova configuração, carregando **CCPxCON** com o valor desejado.

Quanto ao funcionamento do Capture em modo SLEEP, a história é um pouco confusa, pois nessa situação ou o Timer 1 / Timer 3 não está funcionando ou está em modo assíncrono. Por isso, a interrupção de CCP pode até acontecer, acordando o PIC, mas os registradores **CCPRxH** e **CCPRxL** não são atualizados.

Resumo dos registradores associados ao Capture

Endereço	Nome	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
FF2h	INTCON	INTCON	GIE/ GIEH	PEIE/ GIEL	T0IE	INTE	RBIE	TMR0IF	INTF
F9Eh	PIR1	PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
F9Dh	PIE1	PSPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
F9Fh	IPR1	PSPIP	ADIP	RCIP	RXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP
FA1h	PIR2	OSCFIF	CMIF	-	EEIF	BCLIF	HLVDIF	TMR3IF	CCP2IF
FA0h	PIE2	OSCFIE	CMIE	-	EEIE	BCLIE	HLVDIE	TMR3IE	CCP2IE
FA2h	IPR2	OSCFIP	CMIP	-	EEIP	BCLIP	HLVDIP	TMR3IP	CCP2IP
FBDh	CCP1CON	P1M1	P1M0	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0
FBFh	CCPR1H	Captura de TMRxH (parte alta)							
FBEh	CCPR1L	Captura de TMRxL (parte baixa)							
FBAh	CCP2CON	-	-	DC2B1	DC2B0	CCP2M3	CCP2M2	CCP2M1	CCP2M0
FBCh	CCPR2H	Captura de TMRxH (parte alta)							
FBBh	CCPR2L	Captura de TMRxL (parte baixa)							
F94h	TRISC	Configuração do PORTC com Entrada(1) ou Saída(0)							

Não usado para essa finalidade. Para obter mais informações, consulte o Apêndice A.

Modo Compare

Enquanto no modo Capture o valor de Timer 1 / Timer 3 era capturado e armazenado, aqui ele será constantemente comparado (olhe a origem do nome novamente) com os valores dos registradores **CCPRxH** e **CCPRxL**. Sempre que essa comparação de 16 bits resultar numa igualdade, o flag **CCPxIF** será ativado e a interrupção pode acontecer, caso ela esteja devidamente ligada. Além disso, se desejado, podemos alterar automaticamente o estado do pino CCPx.

Também vale a observação quanto ao funcionamento do Timer 1 / Timer 3, que deve obrigatoriamente estar ajustado em modo síncrono (**TxCON</TxSYNC>=0**). Não se esqueça também de configurar o pino **CCPx** como saída, através do **TRISC**.

A ativação do Compare e as opções para mudança no pino podem ser configuradas em **CCPxCON<CCPxM3:CCPxM0>**:

CCPxM3:CCPxM0	Descrição
0000	CCPx desligado
1000	Inicia com o pino em 0 (baixo) e altera para 1 (alto) quando a comparação for bem-sucedida.
1001	Inicia com o pino em 1 (alto) e altera para 0 (baixo) quando a comparação for bem-sucedida.
1010	Não altera o pino.
1011	Não altera o pino, mas reseta Timer 1 / Timer 3.
Observação: As demais opções de configuração dizem respeitos aos outros modos (Compare/PWM).	

Em todas as opções, o flag da interrupção sempre é ativado.

A última opção (1011) é chamada no Data Sheet do PIC de Special Event Trigger, e apesar de ela não alterar o estado do pino **CCPx**, uma outra alteração muito importante acontece. Os registradores do Timer 1 / Timer 3 (**TMRxH** e **TMRxL**) são zerados. Com isso podemos utilizar o Timer 1 / Timer 3 de forma similar ao Timer 2 com o registrador de limite **PR2**. A vantagem é que estamos trabalhando com 16 bits, e não mais com 8. Este é o único caso de diferença entre CCP1 e CCP2, pois o CCP2, além de resetar o Timer 1 / Timer 3, também inicia uma conversão analógica (**ADCON0<GO/DONE>**) se o conversor estiver ligado. Atenção ao escolher essa configuração quando os dois modos CCP1 e CCP2 estiverem em uso, pois tanto o Compare quanto o Capture utilizam o Timer 1 / Timer 3 como base de tempo.

No modo SLEEP, ou Timer 1 / Timer 3, está em modo assíncrono ou está paralisado. Em nenhum dos dois casos o Compare pode funcionar. Por isso, esse modo não opera durante o SLEEP. Só não se esqueça de que a saída **CCPx** é um pino como outro qualquer e, por isso, o seu nível de tensão é garantido mesmo em SLEEP.

Resumo dos registradores associados ao Compare

Endereço	Nome	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
FF2h	INTCON	INTCON	GIE/ GIEH	PEIE/ GIEL	T0IE	INTE	RBIE	TMR0IF	INTF
F9Eh	PIR1	PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
F9Dh	PIE1	PSPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
F9Fh	IPR1	PSPIP	ADIP	RCIP	RXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP
FA1h	PIR2	OSCFIF	CMIF	-	EEIF	BCLIF	HLVDIF	TMR3IF	CCP2IF
FA0h	PIE2	OSCFIE	CMIE	-	EEIE	BCLIE	HLVDIE	TMR3IE	CCP2IE
FA2h	IPR2	OSCFIP	CMIP	-	EEIP	BCLIP	HLVDIP	TMR3IP	CCP2IP
FBDh	CCP1CON	P1M1	P1M0	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0

Endereço	Nome	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
FBFh	CCPR1H	Comparação de TMRxH (parte alta)							
FBEh	CCPR1L	Comparação de TMRxL (parte baixa)							
FBAh	CCP2CON	-	-	DC2B1	DC2B0	CCP2M3	CCP2M2	CCP2M1	CCP2M0
FBCh	CCPR2H	Comparação de TMRxH (parte alta)							
FBBh	CCPR2L	Comparação de TMRxL (parte baixa)							
F94h	TRISC	Configuração do PORTC com Entrada(1) ou Saída(0)							

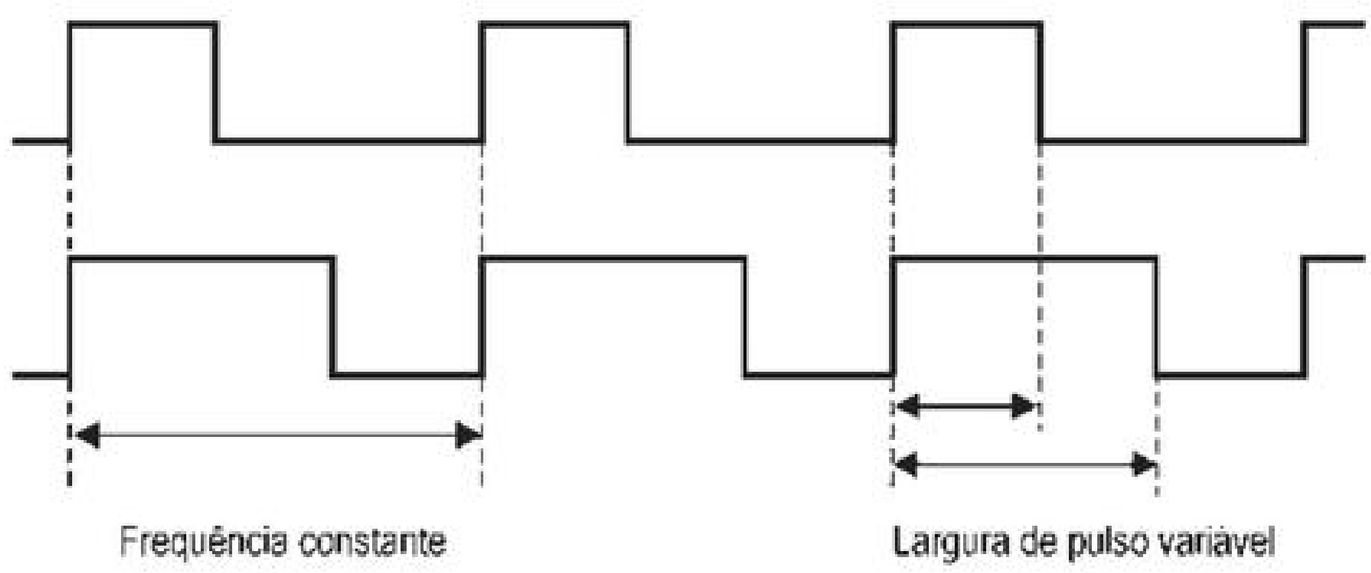
Não usado para essa finalidade. Para obter mais informações, consulte o Apêndice A.

Modo PWM

O modo PWM é provavelmente o recurso mais poderoso dos módulos CCPs, pois com ele podemos obter uma tensão analógica a partir de um sinal digital. Na verdade, essa saída é meramente digital, isto é, somente pode assumir os estados 0 e 1. Porém, pelo conceito aplicado ao PWM, podemos transformá-la em uma tensão variável. Obviamente isso exige um hardware complementar (filtros) depois da saída do PIC, mas é uma outra história. Por enquanto, vamos entender melhor o que é um PWM.

O nome PWM tem origem no inglês Pulse Width Modulation que em português pode ser considerado como Modulação por Largura de Pulso. Mas o que exatamente significa isso? Trata-se de uma onda com frequência constante (período fixo) e largura de pulso (duty cycle) variável.

Na figura seguinte temos a representação de duas formas de onda PWM, cada uma delas com uma largura de pulso diferente:



Esse tipo de sinal é particularmente importante, já que a partir dele é possível implementar um conversor digital analógico com um único pino do microcontrolador, uma vez que controlando a largura do pulso, é possível obter uma tensão analógica variável.

Vejamos a teoria. Genericamente, a tensão média de uma forma de onda é dada por:

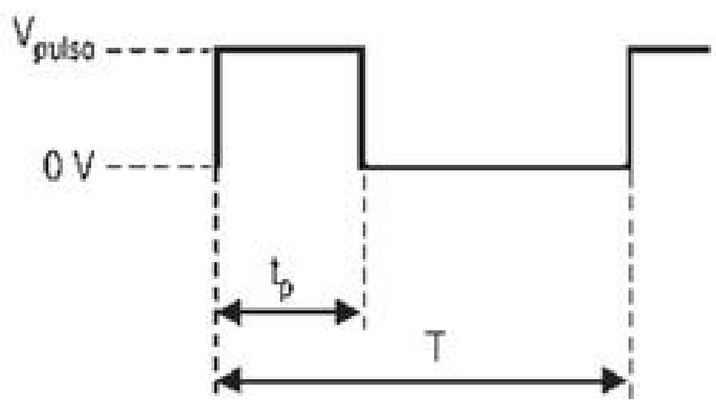
$$V_{dc} = \frac{1}{T}\int_0^T V(t)dt$$

Em que T é o período da forma de onda e V(t) é a função da tensão no tempo.

Para o caso do PWM temos que: $V(t) = \begin{cases} V_{pulso} \rightarrow 0 \leq t \leq t_p \\ 0 \rightarrow t_p < t \leq T \end{cases}$

Sendo: t_p a duração do pulso em nível lógico 1

V_{pulso} a tensão de pulso do sinal PWM



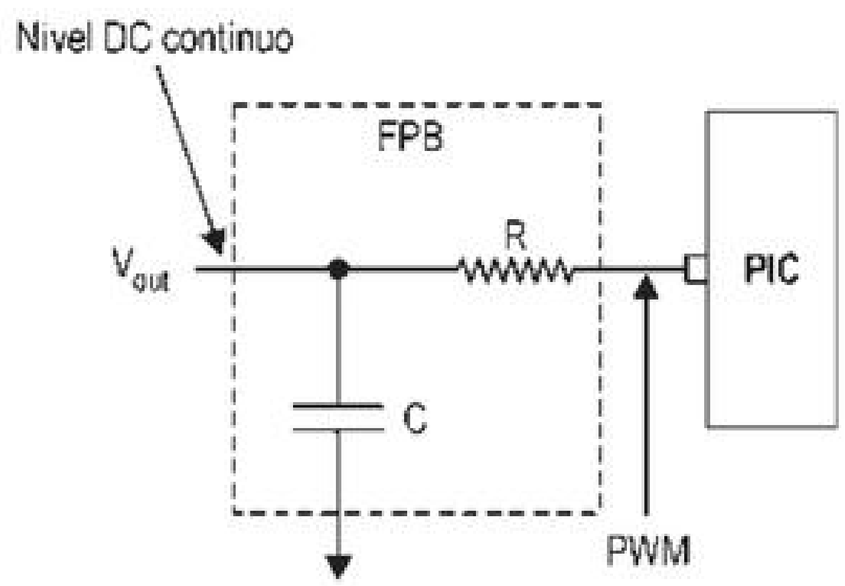
Então,

$$V_{dc} = \frac{1}{T}\left(\int_0^{t_p} V_{pulso}\,dt + \int_{t_p}^{T} 0\,dt\right)$$

$$V_{dc} = \frac{t_p}{T} V_{pulso}$$

A razão entre a largura de pulso e o período da forma de onda recebe o nome de duty cycle, ou em português, ciclo ativo. O pulso da onda PWM apresenta tensão fixa, porém o valor médio da tensão dessa forma de onda varia em função do duty cycle. A tensão média (V_{dc}) é diretamente proporcional ao duty cycle e como este varia entre 0 (quando $t_p = 0$) e 1 (quando $t_p = T$), temos que a tensão média de onda pode variar entre 0 e V_{pulso}. No caso a variação será de V_{SS} a V_{DD}, ou seja, de 0 a 5 V.

Assim, para obtermos um conversor digital analógico a partir do pino **CCPx**, basta implementar um sinal PWM e adicionar à saída um filtro que passa baixa frequência de corte menor que a própria frequência do PWM.



Cálculo da frequência de corte do filtro (fc):

$$F_c = \frac{1}{2\pi RC}$$

Adotar:

$$F_c \leq \frac{f_{PWM}}{10}$$

Quando não é necessário obter uma tensão média contínua, a implementação do filtro é descartada, como nos casos da placa proposta. Tanto o resistor de aquecimento quanto o ventilador trabalham com PWMs sem filtro, pois a função desses componentes faz com que eles atuem como filtros, desde que a frequência do PWM não seja muito baixa.

Assim sendo, a teoria continua válida, o que significa que podemos, através do PWM, regular a taxa de aquecimento do resistor e a velocidade do ventilador, variando a tensão média aplicada a eles.

Editora Érica - Desbravando o Microcontrolador PIC18 Recursos Avançados - Daniel R. de Sousa, David José de Souza e Nicolás César Lavinia - 1ª Edição

Vamos aprender agora um pouco mais sobre o funcionamento dos PWMs dentro do 18F4520.

Esse PIC possui dois canais de PWMs (CCP1 e CCP2), cada um com resolução máxima de dez bits. Isso significa que o duty cycle pode ser regulado de 0 a 100% com uma resolução máxima de 1.024 pontos. No entanto, dependendo da configuração adotada, essa resolução não é atingida.

Vamos estudar como os tempos do PWM (pulso e período) são controlados internamente para entender melhor esse problema.

O período do PWM (T) é controlado diretamente pelo Timer 2, através do registrador **PR2**. Como já foi visto no Capítulo 4, sempre que **TMR2 = PR2**, o timer é zerado. Nesse momento, um novo período do PWM é iniciado. Desta forma, podemos definir o período e a frequência do PWM pelas seguintes fórmulas:

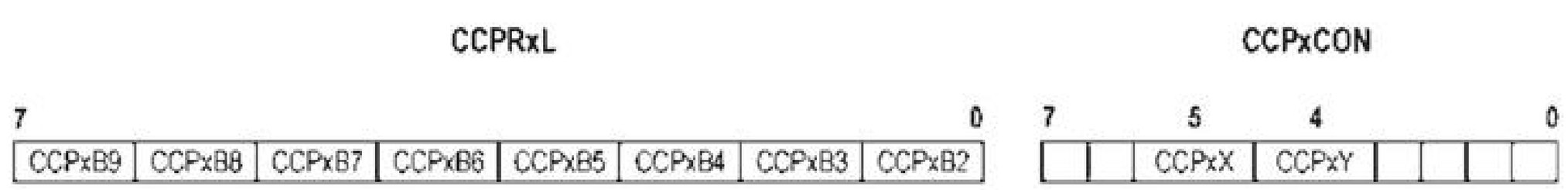
$$T = [(\mathbf{PR2}) + 1] \times 4 \times T_{OSC} \times (\text{Prescale do TMR2})$$

$$PWM_{Freq} = 1 / T$$

Tudo bem quanto ao período, mas como definimos o duty cycle? Na realidade, no PIC não definimos o valor do duty cycle e sim o tempo do pulso em nível alto. Desta forma, o tempo do pulso pode ser calculado por:

$$tp = \mathbf{CCPRxL{:}CCPxCON<CCPxX{:}CCPxY>} \times T_{OSC} \times (\text{Prescale do TMR2})$$

Repare que a largura do pulso é ajustada em dois registradores: **CCPRxL**, que armazena os 8 bits mais significativos, e **CCPxCON**, que armazena os dois bits menos significativos. Assim, temos os 10 bits que controlam o duty cycle do PWM alocados da seguinte maneira:



Para ficarmos de acordo com a teoria, calcularemos efetivamente o duty cycle dividindo o tempo do pulso em nível alto pelo período total do PWM.

$$\frac{t_p}{T} = \frac{CCPRxL : CCPxCON < DCxB1 : xB0 > \times T_{OSC} \times (\text{Prescale do TMR2})}{[(PR2) + 1] \times 4 \times T_{OSC} \times (\text{Prescale do TMR2})}$$

$$\frac{t_p}{T} = \frac{CCPRxL : CCPxCON < DCxB1 : dc \times b0 >}{[(PR2) + 1] \times 4}$$

Verifica-se então que apesar de o período e o tempo de pulso dependerem do cristal (T_{OSC}) e do ajuste do prescaler do Timer 2, o duty cycle depende única e exclusivamente dos valores ajustados nos registradores **PR2**, **CCPRxL** e **CCPxCON** (bits 5 e 4).

Veja que o registrador **PR2** (8 bits) que controla o período do PWM é multiplicado por quatro para poder igualar-se aos 10 bits que controlam o duty cycle. É justamente este o problema da resolução máxima atingida. Se o registrador **PR2** for ajustado com um valor menor que 8 bits, ou seja, menor do que 255, são necessários menos do que 10 bits para atingir um PWM com 100% de duty cycle. Portanto, o número de pontos para ajuste do duty cycle é quatro vezes maior do que o valor ajustado em (**PR2**+1). Em termos de bits, podemos dizer que a resolução do duty cycle é 2 bits maior do que o número de bits que formam o valor ajustado em **PR2**. Repare também que, caso **PR2** seja ajustado com 255, nunca será atingido um duty

cycle de 100%, pois o período atinge o valor máximo de 1024 ([**PR2**+1]x4), enquanto o tempo do pulso em nível alto (**<DCxB9:DCxB0>**) será no máximo 1023.

É fácil notar também que a resolução para o ajuste do período depende do prescaler do Timer 2, assim:

Prescale	Tempo do Menor Passo (Resolução)
1	T_{OSC}
4	$4T_{OSC}$ ou T_{CY}
16	$16T_{OSC}$ ou $4T_{CY}$

De qualquer forma, a menor resolução para o tempo do pulso (duty cycle) será sempre quatro vezes menor que a do período.

Note também que o postscale do Timer 2 não é utilizado para a construção dos tempos envolvidos no PWM.

Uma forma de calcular a quantidade máxima de bits que define a quantidade máxima de passos do PWM é:

$$\frac{\log\left(\frac{F_{OSC}}{F_{PWM}}\right)}{\log(2)}$$

Vamos a um exemplo prático.

Calculemos os valores para um PWM de 78,125 kHz, com um PIC rodando a 20 MHz e ajuste do prescaler do Timer2 em 1:1.

$$1 / 78{,}125\ \text{kHz} = [(PR2) + 1) \times 4 \times (1 / 20\ \text{MHz}) \times 1$$

$$12{,}8\ \mu s = [(PR2) + 1] \times 4 \times 50\ \text{ns} \times 1$$

$$\mathbf{PR2 = 63}$$

Partimos agora para a conta da resolução:

$$1 / 78{,}125\ \text{kHz} = 2^{PWMResolução} \times (1 / 20\ \text{MHz}) \times 1$$

$$12{,}8\ \mu s = 2^{PWMResolução} \times 50\ \text{ns} \times 1$$

$$2^{PWMResolução} = 256$$

$$PWM_{Resolução} \times \log(2) = \log(256)$$

$$PWM_{Resolução} = 8\ (\text{bits})$$

Quanto à operação prática do PWM, já ficou claro que antes de mais nada é necessário definirmos a frequência de trabalho, com base na frequência de funcionamento do PIC e na resolução desejada. Com isso calculamos o valor para ser colocado em **PR2**.

Depois devemos configurar o pino **CCPx** para ser utilizado como saída. Essa configuração não é automática e deve ser feita através do **TRISC**. Em seguida, devemos calcular a largura de pulso desejada. O

resultado deve ser armazenado em dois registradores, sendo os oito bits mais significativos em **CCPRxL** e os outros dois bits restantes em **CCPxCON<DCxB1:DCxB0>**.

Para o PWM, o registrador **CCPRxH** é do tipo somente leitura e utilizado pelo sistema para armazenar uma cópia do **CCPRxL**. Essa cópia é utilizada para possibilitar que a largura de pulso seja alterada durante o funcionamento do PWM. Assim sendo, a nova largura é adotada automaticamente no próximo período do PWM. Os dois bits adicionais também são armazenados internamente pelo sistema.

Quando **TMR2 = PR2**, as seguintes ações acontecem:

- ✓ **TMR2 = 0**, iniciando o próximo período;
- ✓ O pino **CCPx** é colocado em 1 (alto), a menos que a largura do pulso esteja definida para 0 (zero);
- ✓ O valor do registrador **CCPRxL** é copiado para **CCPRxH**, atualizando a largura de pulso.

O sistema passa, então, a monitorar o término do pulso, quando **TMR2 = CCPRxH**. Essa comparação considera ainda os dois bits menos significativos. Neste momento, a saída **CCPx** é colocada em 0 (baixo), até que um novo período comece. Caso o tamanho do pulso seja especificado como sendo maior que o período total, a saída **CCPx** nunca será colocada em 0 (baixo), mas o sistema funciona normalmente, como ajustado para 100% do PWM.

Para que, finalmente, a saída comece a operar, é necessário ainda ajustar o prescaler do Timer 2 e ligá-lo, através do registrador **T2CON**. Por último, ligue também o módulo de PWM, através dos bits **CCPxCON<CCPxM3:CCPxM0>**:

CCPxM3:CCPxM0	Descrição
0000	CCPx desligado.
110x	Ativa a saída PWM.

No caso do CCP1, os bits **CCP1CON<P1M1:P1M0>** devem estar em 0, como é visto no próximo tópico.

Os PWMs não funcionam em modo SLEEP nem geram interrupções, porém não se esqueça de que a interrupção de Timer2 pode continuar acontecendo.

Resumo dos registradores associados ao PWM

Endereço	Nome	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
FBDh	CCP1CON	P1M1	P1M0	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0
FBFh	CCPR1H	Cópia de CCPR1L (somente leitura)							
FBEh	CCPR1L	Largura de pulso, bits de 9 a 2 (parte baixa)							
FBAh	CCP2CON	-	-	DC2B1	DC2B0	CCP2M3	CCP2M2	CCP2M1	CCP2M0
FBCh	CCPR2H	Cópia de CCPR2L (somente leitura)							
FBBh	CCPR2L	Largura de pulso, bits de 9 a 2 (parte baixa)							
F94h	TRISC	Configuração do PORTC com Entrada(1) ou Saída(0)							

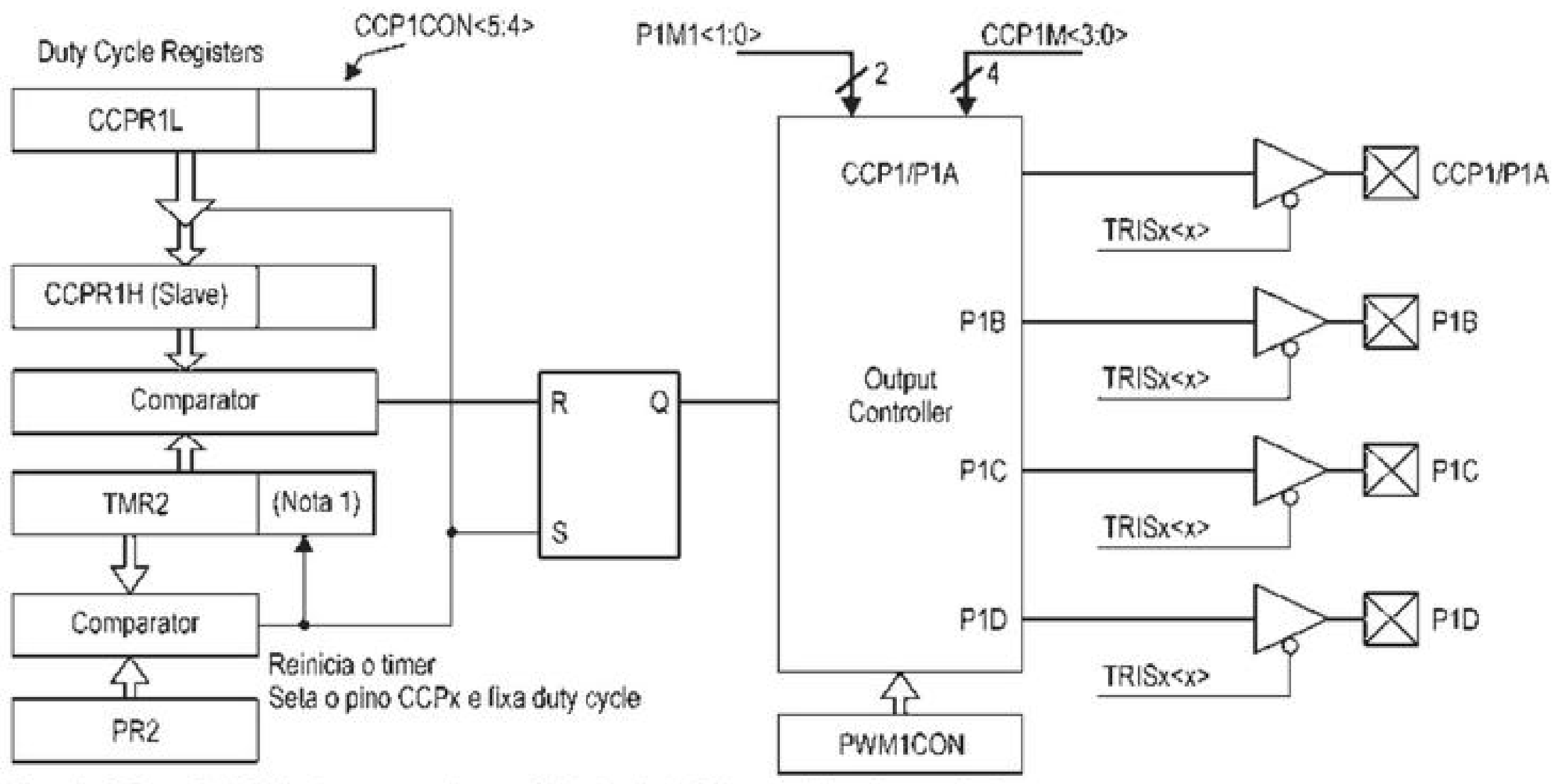
Não usado para essa finalidade. Para obter mais informações, consulte o Apêndice A.

Módulo PWM com capacidade aumentada (ECCP)

O módulo CCP1 possui uma nova função que é o PWM compatível com pontes inversoras monofásicas em modo half-bridge, full-bridge, com diversas configurações e ajuste do dead band (tempo morto). É um

módulo ideal para controle de motores de corrente contínua, inversores de frequência, motores de corrente alternada monofásicos, entre outras aplicações.

Na figura seguinte visualizam-se algumas características: podemos trabalhar com uma saída simples (módulo PWM padrão em **P1A**), duas saídas (meia ponte em **P1A** e **P1B**) ou quatro saídas (ponte completa em **P1A**, **P1B**, **P1C** e **P1D**), onde as saídas devem ser configuradas no registrador **TRISx** (em que **x** se refere ao **PORTC** ou **PORTD**).



Nota 1: O Timer 2 de 8 bits é concatenado com 2 bits do clock Q interno (ciclo de máquina), para formar um contador de 10 bits.

Note que nesse módulo a base de tempo é a mesma do módulo CCP comum quando configurado como PWM, o Timer 2. O cálculo do período é o mesmo apresentado no item Módulo PWM. Para os modos PWM com duas e quatro saídas existe uma configuração referente ao dead band (tempo morto), que é o tempo entre as transições de uma(s) chave(s) e outra(s) nos modos PWM com duas (meia ponte) e quatro saídas (ponte completa). Esse ajuste é feito no registrador **PWM1CON<PDC6:PDC0>** e é calculado pela equação:

$$Tdb = \frac{4}{F_{osc}} \times PCD \qquad PCD = \frac{(Tdb \times F_{osc})}{4}$$

Em que **Tdb** é o tempo do dead band (tempo morto), **Fosc** é a frequência do oscilador e **PCD** é um valor entre 0 e 127, correspondente aos bits **PWM1CON<PDC6:PDC0>**.

Os bits **CCP1CON<P1M1:P1M0>** e **CCP1CON<CCP1M3:CCP1M0>** determinam o modo de trabalho (PWM padrão, PWM com duas saídas - meia ponte e PWM com quatro saídas - ponte completa) e o sentido (direto e inverso).

P1M1: P1M0	CCP1M3: CCP1M0	Descrição
00	110x	Seleciona módulo PWM padrão. O sinal modulado (PWM) sai em **P1A**, ativo em nível alto. Pinos **P1B**, **P1C** e **P1D** operam como I/O.
00	111x	Seleciona módulo PWM padrão. O sinal modulado (PWM) sai em **P1A**, ativo em nível baixo. Pinos **P1B**, **P1C** e **P1D** operam como I/O.
10	1100	Seleciona módulo PWM com duas saídas (meia ponte em **P1A** e **P1B**). O sinal modulado (PWM) sai em **P1A**, ativo em nível alto e **P1B**, ativo em nível alto. Pinos **P1C** e **P1D** operam como I/O.

P1M1: P1M0	CCP1M3: CCP1M0	Descrição
10	1101	Seleciona módulo PWM com duas saídas (meia ponte em **P1A** e **P1B**). O sinal modulado (PWM) sai em **P1A**, ativo em nível alto e **P1B**, ativo em nível baixo. Pinos **P1C** e **P1D** operam como I/O.
10	1110	Seleciona módulo PWM com duas saídas (meia ponte em **P1A** e **P1B**). O sinal modulado (PWM) sai em **P1A**, ativo em nível baixo e **P1B**, ativo em nível alto. Pinos **P1C** e **P1D** operam como I/O.
10	1111	Seleciona módulo PWM com duas saídas (meia ponte em **P1A** e **P1B**). O sinal modulado (PWM) sai em **P1A**, ativo em nível baixo e **P1B**, ativo em nível baixo. Pinos **P1C** e **P1D** operam como I/O.
01	1100	Seleciona módulo PWM com quatro saídas (ponte completa em **P1A**, **P1B**, **P1C** e **P1D**), sentido direto. O sinal modulado (PWM) sai em **P1D**, ativo em nível alto. Saída **P1A** é colocada em nível alto e **P1B** e **P1C** são colocados em nível baixo.
01	1101	Seleciona módulo PWM com quatro saídas (ponte completa em **P1A**, **P1B**, **P1C** e **P1D**), sentido direto. O sinal modulado (PWM) sai em **P1D**, ativo em nível baixo. Saída **P1C** é colocada em nível baixo e **P1A** e **P1B** são colocados em nível alto.
01	1110	Seleciona módulo PWM com quatro saídas (ponte completa em **P1A**, **P1B**, **P1C** e **P1D**), sentido direto. O sinal modulado (PWM) sai em **P1D**, ativo em nível alto. Saída **P1C** é colocado em nível alto e **P1A** e **P1B** são colocados em nível baixo.
01	1111	Seleciona módulo PWM com quatro saídas (ponte completa em **P1A**, **P1B**, **P1C** e **P1D**), sentido direto. O sinal modulado (PWM) sai em **P1D**, ativo em nível baixo. Saída **P1A** é colocada em nível baixo e **P1B** e **P1C** são colocados em nível alto.
11	1100	Seleciona módulo PWM com quatro saídas (ponte completa em **P1A**, **P1B**, **P1C** e **P1D**), sentido inverso. O sinal modulado (PWM) sai em **P1B**, ativo em nível alto. Saída **P1C** é colocada em nível alto e **P1A** e **P1D** são colocados em nível baixo.
11	1101	Seleciona módulo PWM com quatro saídas (ponte completa em **P1A**, **P1B**, **P1C** e **P1D**), sentido inverso. O sinal modulado (PWM) sai em **P1B**, ativo em nível baixo. Saída **P1A** é colocado em nível baixo e **P1C** e **P1D** são colocados em nível alto.
11	1110	Seleciona módulo PWM com quatro saídas (ponte completa em **P1A**, **P1B**, **P1C** e **P1D**), sentido inverso. O sinal modulado (PWM) sai em **P1B**, ativo em nível alto. Saída **P1A** é colocada em nível alto e **P1C** e **P1D** são colocados em nível baixo.
11	1111	Seleciona módulo PWM com quatro saídas (ponte completa em **P1A**, **P1B**, **P1C** e **P1D**), sentido inverso. O sinal modulado (PWM) sai em **P1B**, ativo em nível baixo. Saída **P1C** é colocada em nível baixo e **P1A** e **P1D** são colocados em nível alto.

Os pinos colocados como saída podem ser configurados para auto-shutdown, em que podemos proteger a eletrônica de potência (transistor, MOSFET, IGBT, GTO etc.) contra queima em decorrência da elevação de corrente consumida pela ponte. Podemos utilizar o pino **FLT0** ou os comparadores de tensão internos do microcontrolador.

A seleção das entradas é feita nos bits **ECCP1AS<ECCPAS2:ECCPAS0>**.

ECCPAS2:ECCPAS0	Descrição
111	FLT0 ou comparador 1 ou comparador 2.
110	FLT0 ou comparador 2.
101	FLT0 ou comparador 1.
100	FLT0.
011	Comparador 1 ou comparador 2.
010	Saída do comparador 2.
001	Saída do comparador 1.
000	Desabilita o auto-shutdown.

Em seguida devemos configurar o estado dos pinos após evento que dispara o auto-shutdown. Os pinos **P1A** e **P1C** são configurados nos bits **ECCP1AS<PSSAC1:PSSAC0>** e os pinos **P1B** e **P1D** são configurados nos bits **ECCP1AS<PSSBD1:PSSBD0>**.

PSSAC1:PSSAC	Descrição
1x	P1A e P1C em modo tri-state.
01	P1A e P1C em nível alto.
00	P1A e P1C em nível baixo.

PSSBD1:PSSBD0	Descrição
1x	P1B e P1D em modo tri-state.
01	P1B e P1D em nível alto.
00	P1B e P1D em nível baixo.

Na ocorrência do auto-shutdown o bit **ECCP1AS<ECCPASE>** é setado. O bit **PWM1CON<PRSEN>** configura o reinício do módulo de forma automática após evento de auto-shutdown:

- ✓ **PWM1CON<PRSEN>** = 1: o bit **ECCP1AS<ECCPASE>** é limpo automaticamente e o PWM é reiniciado.
- ✓ **PWM1CON<PRSEN>** = 0: o bit **ECCP1AS<ECCPASE>** deve ser limpo manualmente para reiniciar o PWM.

Resumo dos registradores associados ao ECCP

Endereço	Nome	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
FBDh	CCP1CON	P1M1	P1M0	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0
FBFh	CCPR1H	Cópia de CCPR1L (somente leitura)							
FBEh	CCPR1L	Largura de pulso, bits de 9 a 2 (parte baixa)							
FB6h	ECCP1AS	ECCPASE	ECCPAS2	ECCPAS1	ECCPAS0	PSSAC1	PSSAC0	PSSBD1	PSSBD0
FB7h	PWM1CON	PRSEN	PDC6	PDC5	PDC4	PDC3	PDC2	PDC1	PDC0
F94h	TRISC	Configuração do PORTC com Entrada(1) ou Saída(0)							
F95h	TRISD	Configuração do PORTD com Entrada(1) ou Saída(0)							

» Lógica do exemplo

Infelizmente não é possível construirmos um exemplo que utilize todos os modos do CCP ao mesmo tempo. Por isso, optamos pela criação de um único exemplo que trabalhe com o recurso que consideramos o mais importante: o PWM.

Neste exemplo ativaremos a saída do módulo CCP2 para podermos controlar a rotação do ventilador que está ligado ao pino RC1.

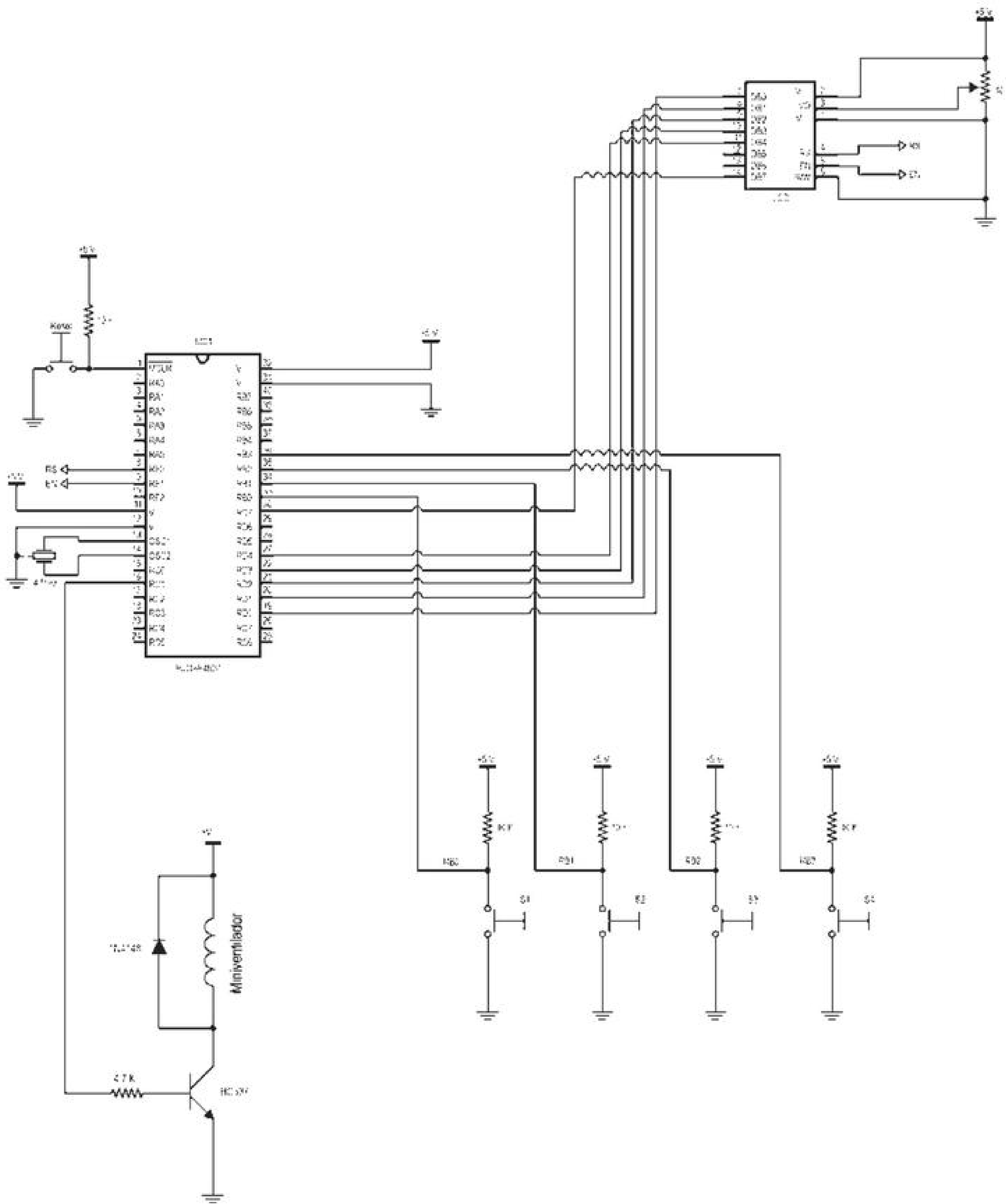
Optamos por trabalhar com PR2 no valor máximo (255) e o prescaler em 16. Com isso a frequência do PWM será de 244,14 Hz ($PWM_{Período}$ = 4,096 ms), considerando-se que na placa proposta o PIC está rodando a 4 MHz. Essa conta foi feita utilizando a fórmula dada na aula teórica.

Para ajuste do duty cycle, optamos pela implementação de quatro botões:

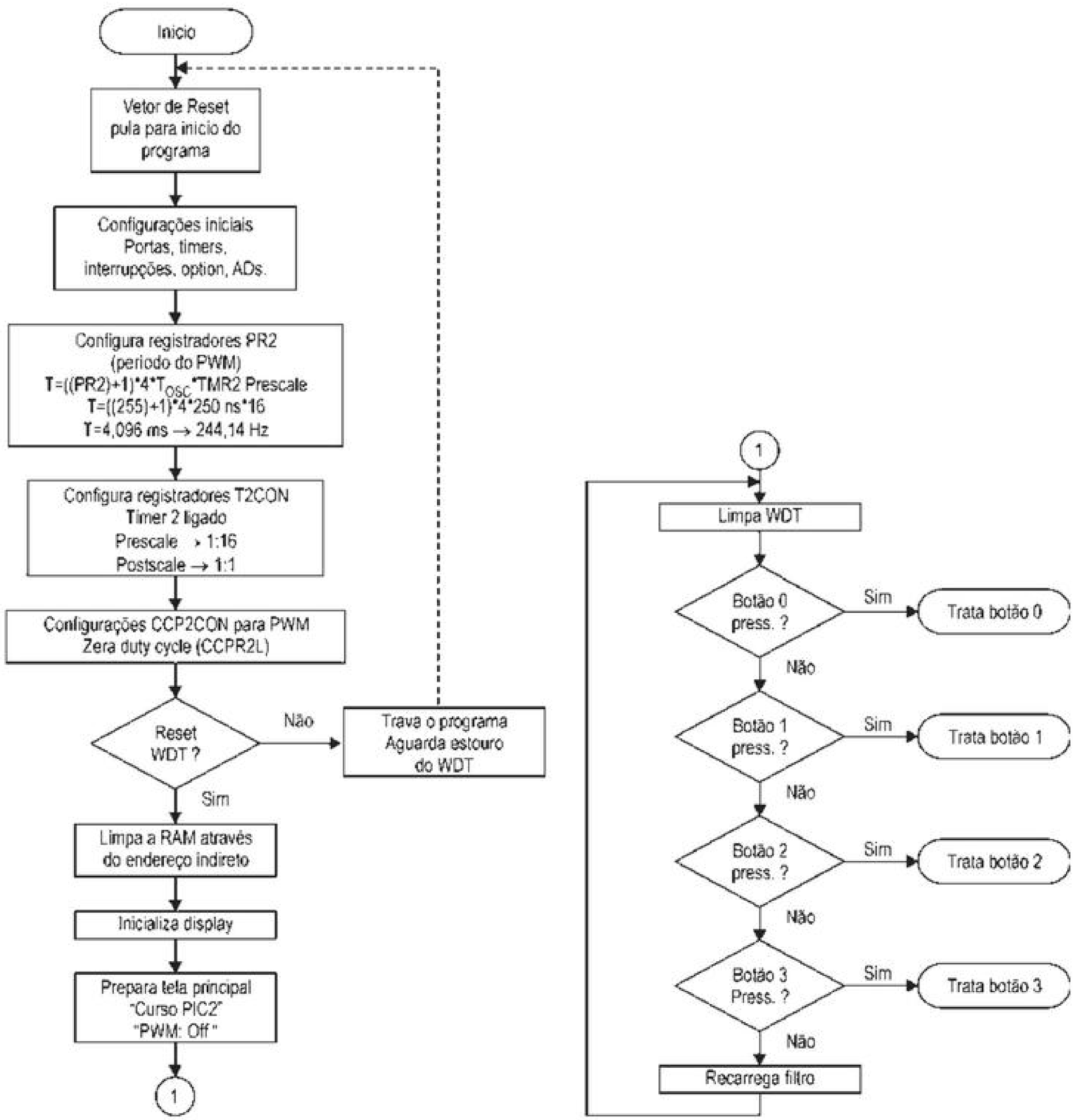
Botão	Duty Cycle
S1	0%
S2	50%
S3	75%
S4	100%

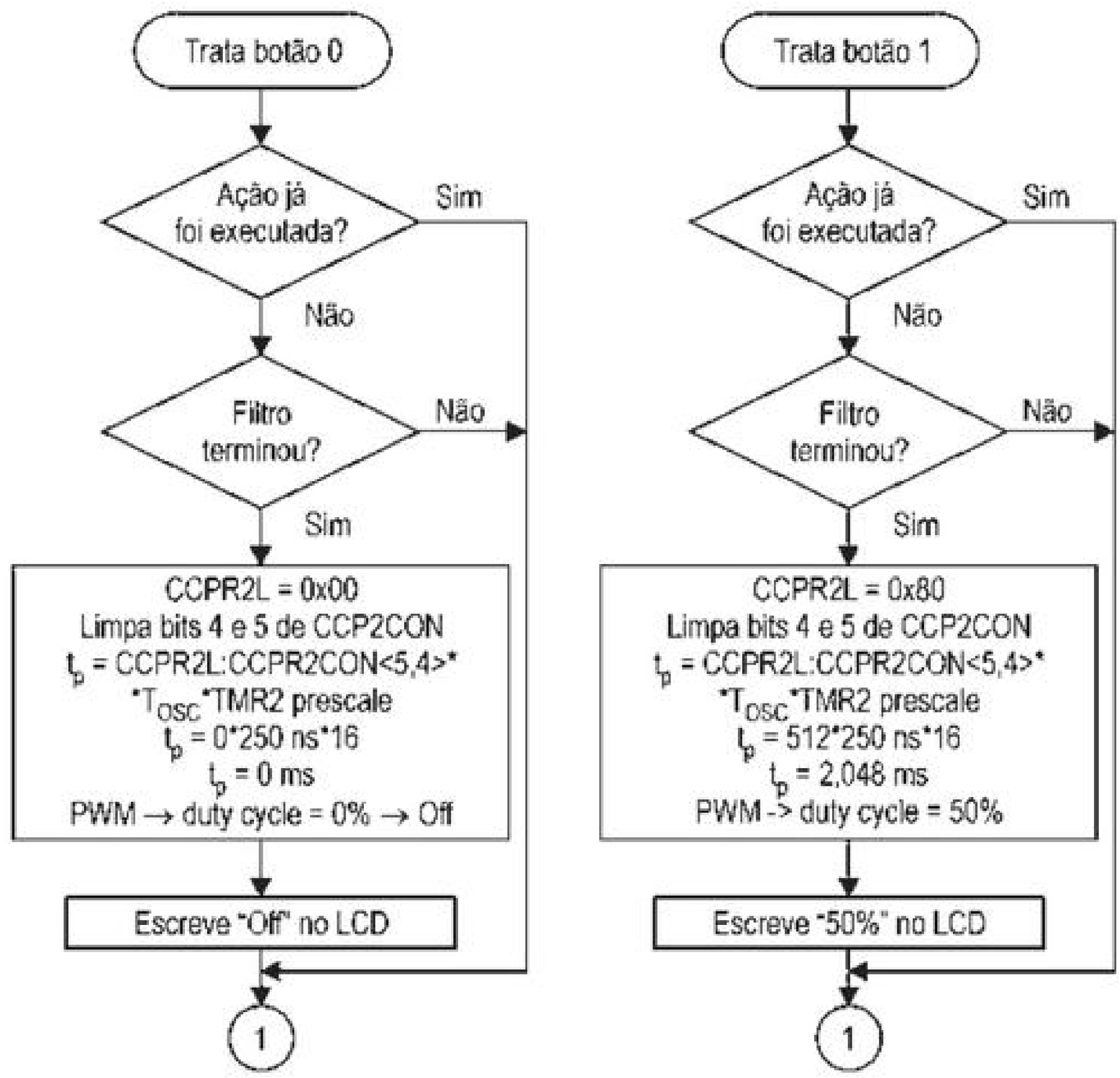
A fim de deixarmos o sistema mais interativo, utilizamos o LCD para mostrar o valor atual ajustado para o PWM.

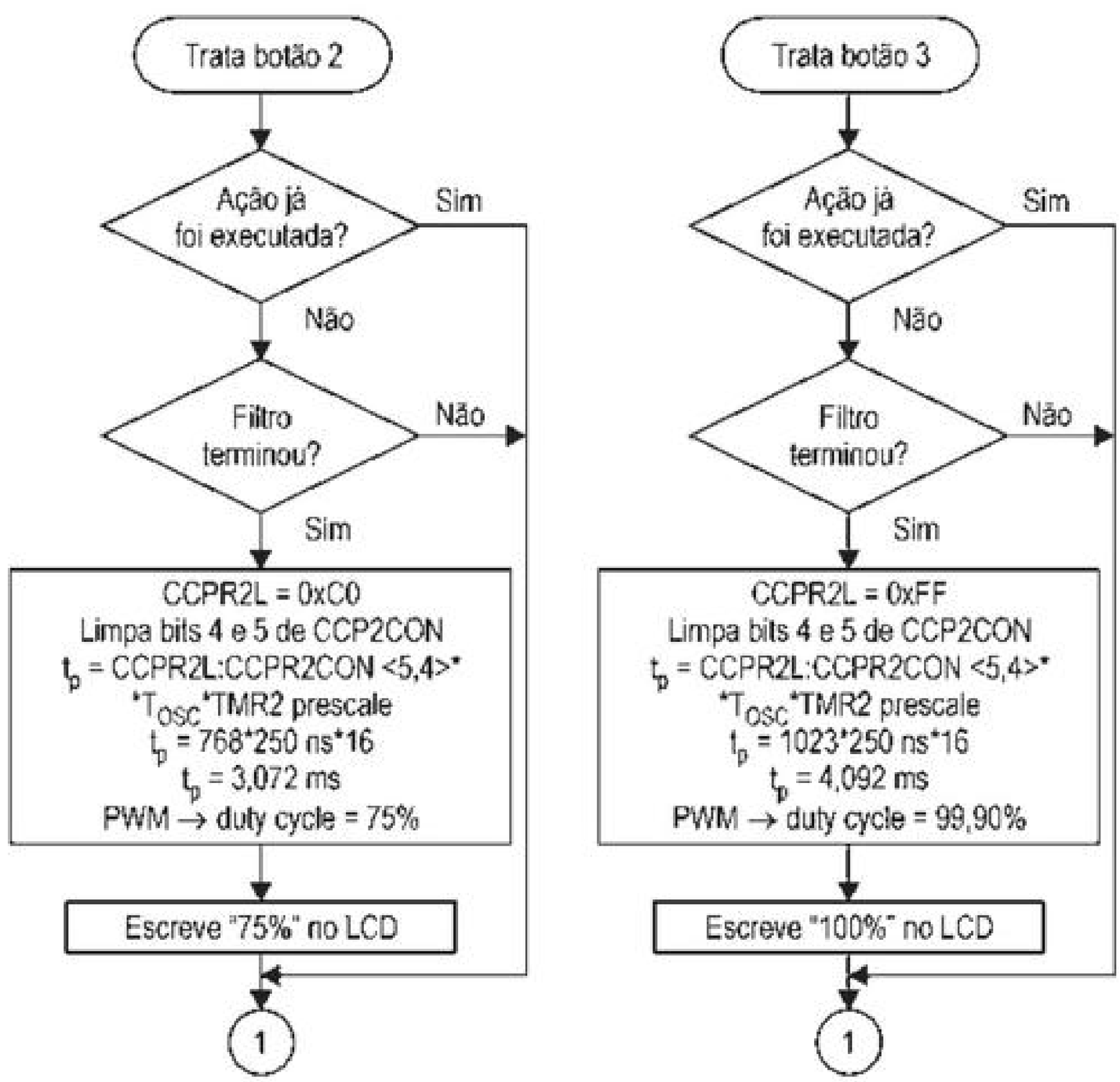
Esquema elétrico



» Fluxograma

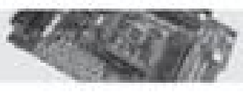

Inicio
Vetor de Reset pula para inicio do programa
Configurações iniciais Portas, timers, interrupções, option, ADs.
Configura registradores PR2 (periodo do PWM) T=((PR2)+1)*4*T_OSC*TMR2 Prescale T=((255)+1)*4*250 ns*16 T=4,096 ms → 244,14 Hz
Configura registradores T2CON Timer 2 ligado Prescale → 1:16 Postscale → 1:1
Configurações CCP2CON para PWM Zera duty cycle (CCPR2L)
Reset WDT ?
Não
Trava o programa Aguarda estouro do WDT
Sim
Limpa a RAM através do endereço indireto
Inicializa display
Prepara tela principal "Curso PIC2" "PWM: Off"
1
1
Limpa WDT
Botão 0 press. ?
Sim
Trata botão 0
Não
Botão 1 press. ?
Sim
Trata botão 1
Não
Botão 2 press. ?
Sim
Trata botão 2
Não
Botão 3 Press. ?
Sim
Trata botão 3
Não
Recarrega filtro


Trata botão 0
Ação já foi executada?
Sim
Não
Filtro terminou?
Não
Sim
CCPR2L = 0x00
Limpa bits 4 e 5 de CCP2CON
t_p = CCPR2L:CCPR2CON<5,4>*
*T_{osc}*TMR2 prescale
t_p = 0*250 ns*16
t_p = 0 ms
PWM → duty cycle = 0% → Off
Escreve "Off" no LCD
1
Trata botão 1
Ação já foi executada?
Sim
Não
Filtro terminou?
Não
Sim
CCPR2L = 0x80
Limpa bits 4 e 5 de CCP2CON
t_p = CCPR2L:CCPR2CON<5,4>*
*T_{osc}*TMR2 prescale
t_p = 512*250 ns*16
t_p = 2,048 ms
PWM -> duty cycle = 50%
Escreve "50%" no LCD
1


Trata botão 2
Ação já foi executada?
Sim
Não
Filtro terminou?
Não
Sim
CCPR2L = 0xC0
Limpa bits 4 e 5 de CCP2CON
t_p = CCPR2L:CCPR2CON <5,4>*
*T_{osc}*TMR2 prescale
t_p = 768*250 ns*16
t_p = 3,072 ms
PWM → duty cycle = 75%
Escreve "75%" no LCD
1
Trata botão 3
Ação já foi executada?
Sim
Não
Filtro terminou?
Não
Sim
CCPR2L = 0xFF
Limpa bits 4 e 5 de CCP2CON
t_p = CCPR2L:CCPR2CON <5,4>*
*T_{osc}*TMR2 prescale
t_p = 1023*250 ns*16
t_p = 4,092 ms
PWM → duty cycle = 99,90%
Escreve "100%" no LCD
1

» Código

```
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *              DESBRAVANDO O PIC18 - RECURSOS AVANÇADOS                   *
; *                              EXEMPLO 6                                  *
; *                                                                         *
; * DANIEL RODRIGUES DE SOUSA, DAVID JOSÉ DE SOUZA e  NICOLÁS CÉSAR LAVINIA *
; *                                                                         *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *     DATA : 03/08/2009                                                   *
; *   VERSÃO : 1.0                                                          *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                           DESCRIÇÃO GERAL                               *
; *                                                                         *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTE EXEMPLO FOI ELABORADO PARA EXPLICAR O FUNCIONAMENTO DO MÓDULO PWM.
;  ELE MONITORA OS QUATRO BOTÕES E CONFORME O BOTÃO SELECIONADO
;  APLICA UM VALOR DIFERENTE NO PWM, FAZENDO ASSIM UM CONTROLE SOBRE A
;  VELOCIDADE DO VENTILADOR. NO LCD É MOSTRADO O VALOR ATUAL DO DUTY CYCLE.
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                        CONFIGURAÇÕES PARA GRAVAÇÃO                      *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
    CONFIG  OSC = XT, FCMEN = OFF, IESO = OFF, PWRT = ON, BOREN = ON, BORV = 0
    CONFIG  WDT = ON, WDTPS = 128, MCLRE = ON, LPT1OSC = OFF, PBADEN = OFF
    CONFIG  CCP2MX = PORTC, STVREN = ON, LVP = OFF, DEBUG = OFF, XINST = OFF
    CONFIG  CP0 = OFF, CP1 = OFF, CP2 = OFF, CP3 = OFF, CPB = OFF, CPD = OFF
    CONFIG  WRT0 = OFF, WRT1 = OFF, WRT2 = OFF, WRT3 = OFF, WRTB = OFF
    CONFIG  WRTC = OFF, WRTD = OFF, EBTR0 = OFF, EBTR1 = OFF, EBTR2 = OFF
    CONFIG  EBTR3 = OFF, EBTRB = OFF

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                          DEFINIÇÃO DAS VARIÁVEIS                        *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
   CBLOCK  0X00            ; BANK 0 --> 0X00..0X5F --> ACCESS BANK

        TEMPO1
        TEMPO0              ; CONTADORES P/ DELAY

        FILTRO_BOTOES       ; FILTRO PARA RUIDOS DOS BOTÕES

    ENDC

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *               DEFINIÇÃO DAS VARIÁVEIS INTERNAS DO PIC                   *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  O ARQUIVO DE DEFINIÇÕES DO PIC UTILIZADO DEVE SER REFERENCIADO PARA QUE
;  OS NOMES DEFINIDOS PELA MICROCHIP POSSAM SER UTILIZADOS, SEM A NECESSIDADE
;  DE REDIGITAÇÃO.

    #INCLUDE <P18F4520.INC> ; MICROCONTROLADOR UTILIZADO

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                           CONSTANTES INTERNAS                           *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  A DEFINIÇÃO DE CONSTANTES FACILITA A PROGRAMAÇÃO E A MANUTENÇÃO.

FILTRO_TECLA    EQU .200    ; FILTRO P/ EVITAR RUIDOS DOS BOTÕES
```

```
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                    DECLARAÇÃO DOS FLAGs DE SOFTWARE                     *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  A DEFINIÇÃO DE FLAGs AJUDA NA PROGRAMAÇÃO E ECONOMIZA MEMÓRIA RAM.

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                                ENTRADAS                                 *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  AS ENTRADAS DEVEM SER ASSOCIADAS A NOMES PARA FACILITAR A PROGRAMAÇÃO E
;  FUTURAS ALTERAÇÕES DO HARDWARE.

#DEFINE BOTAO_0   PORTB,0            ; ESTADO DO BOTÃO 0
                                     ; 1 -> LIBERADO
                                     ; 0 -> PRESSIONADO

#DEFINE BOTAO_1   PORTB,1            ; ESTADO DO BOTÃO 1
                                     ; 1 -> LIBERADO
                                     ; 0 -> PRESSIONADO

#DEFINE BOTAO_2   PORTB,2            ; ESTADO DO BOTÃO 2
                                     ; 1 -> LIBERADO
                                     ; 0 -> PRESSIONADO

#DEFINE BOTAO_3   PORTB,3            ; ESTADO DO BOTÃO 3
                                     ; 1 -> LIBERADO
                                     ; 0 -> PRESSIONADO

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                               SAÍDAS                                   *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  AS SAÍDAS DEVEM SER ASSOCIADAS A NOMES PARA FACILITAR A PROGRAMAÇÃO E
;  FUTURAS ALTERAÇÕES DO HARDWARE.

#DEFINE DISPLAY     LATD             ; BARRAMENTO DE DADOS DO DISPLAY

#DEFINE RS          LATE,0           ; INDICA P/ O DISPLAY UM DADO OU COMANDO
                                     ; 1 -> DADO
                                     ; 0 -> COMANDO

#DEFINE ENABLE      LATE,1           ; SINAL DE ENABLE P/ DISPLAY
                                     ; ATIVO NA BORDA DE DESCIDA

#DEFINE VENTILADOR  LATC,1           ; SAÍDA P/ O VENTILADOR
                                     ; 1 -> VENTILADOR LIGADO
                                     ; 0 -> VENTILADOR DESLIGADO

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                  VETOR DE RESET DO MICROCONTROLADOR                    *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  POSIÇÃO INICIAL PARA EXECUÇÃO DO PROGRAMA

    ORG     0X0000          ; ENDEREÇO DO VETOR DE RESET
    BRA     CONFIGURACAO    ; PULA PARA CONFIG DEVIDO A REGIÃO
                            ; DESTINADA AS ROTINAS SEGUINTES

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                    ROTINA DE DELAY (DE 1MS ATÉ 256MS)                   *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTA É UMA ROTINA DE DELAY VARIÁVEL, COM DURAÇÃO DE 1MS X O VALOR PASSADO
;  EM WORK (W).

DELAY_MS
    MOVWF   TEMPO1          ; CARREGA TEMPO1 (UNIDADES DE MS)
    MOVLW   .250
```

```
    MOVWF   TEMPO0          ; CARREGA TEMPO0 (P/ CONTAR 1MS)

    CLRWDT                  ; LIMPA WDT (PERDE TEMPO)
    DECFSZ  TEMPO0,F        ; FIM DE TEMPO0 ?
    BRA     $-.4            ; NÃO - VOLTA 2 INSTRUÇÕES
                            ; SIM - PASSOU-SE 1ms
    DECFSZ  TEMPO1,F        ; FIM DE TEMPO1 ?
    BRA     $-.12           ; NÃO - VOLTA 6 INSTRUÇÕES
    RETURN                  ; SIM - RETORNA

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *              ROTINA DE ESCRITA DE UM CARACTERE NO DISPLAY             *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTA ROTINA ENVIA UM CARACTERE PARA O MÓDULO DE LCD. O CARACTERE A SER
;  ESCRITO DEVE SER COLOCADO EM WORK (W) ANTES DE CHAMAR A ROTINA.

ESCREVE
    MOVWF   DISPLAY         ; ATUALIZA DISPLAY (PORTD)
    NOP                     ; PERDE 1US PARA ESTABILIZAÇÃO
    BSF     ENABLE          ; ENVIA UM PULSO DE ENABLE AO DISPLAY
    BRA     $+.2            ; .
    BCF     ENABLE          ; .

    MOVLW   .1
    RCALL   DELAY_MS        ; DELAY DE 1ms
    RETURN                  ; RETORNA

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *               CONFIGURAÇÕES INICIAIS DE HARDWARE E SOFTWARE           *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  NESTA ROTINA SÃO INICIALIZADAS AS PORTAS DE I/O DO MICROCONTROLADOR E AS
;  CONFIGURAÇÕES DOS REGISTRADORES ESPECIAIS (SFR). A ROTINA INICIALIZA A
;  MÁQUINA E AGUARDA O ESTOURO DO WDT.

CONFIGURACAO
    CLRF    PORTA           ; GARANTE AS SAÍDAS EM ZERO
    CLRF    PORTB
    CLRF    PORTC
    CLRF    PORTD
    CLRF    PORTE

    CLRF    LATA
    CLRF    LATB
    CLRF    LATC
    CLRF    LATD
    CLRF    LATE

    MOVLW   B'11111111'
    MOVWF   TRISA           ; CONFIGURA I/O DO PORTA

    MOVLW   B'11111111'
    MOVWF   TRISB           ; CONFIGURA I/O DO PORTB

    MOVLW   B'11111101'
    MOVWF   TRISC           ; CONFIGURA I/O DO PORTC

    MOVLW   B'00000000'
    MOVWF   TRISD           ; CONFIGURA I/O DO PORTD

    MOVLW   B'00000100'
    MOVWF   TRISE           ; CONFIGURA I/O DO PORTE

    MOVLW   B'11011000'     ; CONFIGURA TIMER0
    MOVWF   T0CON           ; TIMER0 INCREM. PELO CICLO DE MÁQUINA
```

```
                                ; TIMER0- 1:1

        MOVLW   B'00000000'     ; CONFIGURA INTERRUPÇÕES
        MOVWF   INTCON          ; DESABILITADAS TODAS AS INTERRUPÇÕES

        MOVLW   B'00001111'     ; CONFIGURA CONVERSOR AD
        MOVWF   ADCON1          ; CONFIGURA PORTA, PORTB E PORTE COMO I/O DIGITAL

        MOVLW   .255            ; CONFIGURA PERÍODO DO PWM
        MOVWF   PR2             ; T=((PR2)+1)*4*Tosc*TMR2 Prescale
                                ; T=((255)+1)*4*250ns*16
                                ; T=4,096ms -> 244,14Hz

        MOVLW   B'00000111'     ; CONFIGURA TMR2
        MOVWF   T2CON           ; TIMER 2 LIGADO
                                ; PRESCALE  - 1:16
                                ; POSTSCALE - 1:1

        MOVLW   B'00001100'     ; CONFIGURA CCP2CON PARA PWM
        MOVWF   CCP2CON         ; (PINO RC1)

        CLRF    CCPR2L          ; INICIA COM DUTY CYCLE EM ZERO

;  AS INSTRUÇÕES A SEGUIR FAZEM COM QUE O PROGRAMA TRAVE QUANDO HOUVER UM
;  RESET OU POWER-UP, MAS PASSE DIRETO SE O RESET FOR POR WDT. DESTA FORMA,
;  SEMPRE QUE O PIC É LIGADO, O PROGRAMA TRAVA, AGUARDA UM ESTOURO DE WDT
;  E COMEÇA NOVAMENTE. ISTO EVITA PROBLEMAS NO START-UP DO PIC.

        BTFSC   RCON,NOT_TO     ; RESET POR ESTOURO DE WATCHDOG TIMER ?
        BRA     $               ; NÃO - AGUARDA ESTOURO DO WDT
                                ; SIM

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                        INICIALIZAÇÃO DA RAM                             *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTA ROTINA IRÁ LIMPAR TODA A RAM, DO BANCO 0 ATÉ O BANCO 15

        LFSR    FSR0,0X00       ; APONTA FSR0 PARA ENDEREÇO ZERO

LIMPA_RAM
        CLRF    POSTINC0        ; LIMPA A POSIÇÃO E INCREMENTA FSR0
        CLRWDT                  ; LIMPA WATCHDOG TIMER
        MOVLW   0X0F            ; TESTA SE JÁ CHEGOU AO BANCO 15
        SUBWF   FSR0H,W
        BNZ     LIMPA_RAM       ; SE NÃO, CONTINUA LIMPANDO

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                    CONFIGURAÇÕES INICIAIS DO DISPLAY                    *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; ESTA ROTINA INICIALIZA O DISPLAY P/ COMUNICAÇÃO DE 8 VIAS, DISPLAY PARA 2
; LINHAS, CURSOR APAGADO E DESLOCAMENTO DO CURSOR À DIREITA.

INICIALIZACAO_DISPLAY
        BCF     RS              ; SELECIONA O DISPLAY P/ COMANDOS

        MOVLW   0X30            ; ESCREVE COMANDO 0X30 PARA
        RCALL   ESCREVE         ; INICIALIZAÇÃO

        MOVLW   .3
        RCALL   DELAY_MS        ; DELAY DE 3MS (EXIGIDO PELO DISPLAY)

        MOVLW   0X30            ; ESCREVE COMANDO 0X30 PARA
        RCALL   ESCREVE         ; INICIALIZAÇÃO
```

```
MOVLW   0X30            ; ESCREVE COMANDO 0X30 PARA
RCALL   ESCREVE         ; INICIALIZAÇÃO

MOVLW   B'00111000'     ; ESCREVE COMANDO PARA
RCALL   ESCREVE         ; INTERFACE DE 8 VIAS DE DADOS

MOVLW   B'00000001'     ; ESCREVE COMANDO PARA
RCALL   ESCREVE         ; LIMPAR TODO O DISPLAY

MOVLW   .1
RCALL   DELAY_MS        ; DELAY DE 1MS

MOVLW   B'00001100'     ; ESCREVE COMANDO PARA
RCALL   ESCREVE         ; LIGAR O DISPLAY SEM CURSOR

    MOVLW   B'00000110'     ; ESCREVE COMANDO PARA INCREM.
    RCALL   ESCREVE         ; AUTOMÁTICO À DIREITA

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                ROTINA DE ESCRITA DA TELA PRINCIPAL                      *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTA ROTINA ESCREVE A TELA PRINCIPAL DO PROGRAMA, COM AS FRASES:
;  LINHA 1 - "  CURSO PIC2  "
;  LINHA 2 - "   PWM:  xx%  "

MOSTRA_TELA_PRINCIPAL
    MOVLW   0X83            ; COMANDO PARA POSICIONAR O CURSOR
    RCALL   ESCREVE         ; LINHA 0 / COLUNA 3
    BSF     RS              ; SELECIONA O DISPLAY P/ DADOS

    MOVLW   'C'             ; COMANDOS PARA ESCREVER AS
    RCALL   ESCREVE         ; LETRAS DE "CURSO PIC2"
    MOVLW   'U'
    RCALL   ESCREVE
    MOVLW   'R'
    RCALL   ESCREVE
    MOVLW   'S'
    RCALL   ESCREVE
    MOVLW   'O'
    RCALL   ESCREVE
    MOVLW   ' '
    RCALL   ESCREVE
    MOVLW   'P'
    RCALL   ESCREVE
    MOVLW   'I'
    RCALL   ESCREVE
    MOVLW   'C'
    RCALL   ESCREVE
    MOVLW   '2'
    RCALL   ESCREVE

    BCF     RS              ; SELECIONA O DISPLAY P/ COMANDOS
    MOVLW   0XC3            ; COMANDO PARA POSICIONAR O CURSOR
    RCALL   ESCREVE         ; LINHA 1 / COLUNA 3
    BSF     RS              ; SELECIONA O DISPLAY P/ DADOS

    MOVLW   'P'             ; COMANDOS PARA ESCREVER AS
    RCALL   ESCREVE         ; LETRAS DE "   PWM:  OFF    "
    MOVLW   'W'
    RCALL   ESCREVE
    MOVLW   'M'
    RCALL   ESCREVE
    MOVLW   ':'
    RCALL   ESCREVE
```

```
    MOVLW   ' '
    RCALL   ESCREVE
    MOVLW   ' '
    RCALL   ESCREVE
    MOVLW   'O'
    RCALL   ESCREVE
    MOVLW   'F'
    RCALL   ESCREVE
    MOVLW   'F'
    RCALL   ESCREVE

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                          VARREDURA DOS BOTÕES                          *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; ESTA ROTINA VERIFICA SE ALGUM BOTÃO ESTÁ PRESSIONADO E CASO AFIRMATIVO
; DESVIA PARA O TRATAMENTO DELE.

VARRE
    CLRWDT                      ; LIMPA WATCHDOG TIMER

; ***************** VERIFICA ALGUM BOTÃO PRESSIONADO ***************************

    BTFSS   BOTAO_0             ; O BOTÃO 0 ESTÁ PRESSIONADO ?
    BRA     TRATA_BOTAO_0       ; SIM - PULA P/ TRATA_BOTAO_0
                                ; NÃO
    BTFSS   BOTAO_1             ; O BOTÃO 1 ESTÁ PRESSIONADO ?
    BRA     TRATA_BOTAO_1       ; SIM - PULA P/ TRATA_BOTAO_1
                                ; NÃO
    BTFSS   BOTAO_2             ; O BOTÃO 2 ESTÁ PRESSIONADO ?
    BRA     TRATA_BOTAO_2       ; SIM - PULA P/ TRATA_BOTAO_2
                                ; NÃO
    BTFSS   BOTAO_3             ; O BOTÃO 3 ESTÁ PRESSIONADO ?
    BRA     TRATA_BOTAO_3       ; SIM - PULA P/ TRATA_BOTAO_3
                                ; NÃO
    MOVLW   FILTRO_TECLA        ; CARREGA NO WORK O VALOR DE FILTRO_TECLA
    MOVWF   FILTRO_BOTOES       ; SALVA EM FILTRO_BOTOES
                                ; RECARREGA FILTRO P/ EVITAR RUIDOS

; ***************************** FILTRO P/ EVITAR RUIDOS ************************

    MOVLW   FILTRO_TECLA        ; CARREGA NO WORK O VALOR DE FILTRO_TECLA
    MOVWF   FILTRO_BOTOES       ; SALVA EM FILTRO_BOTOES
                                ; RECARREGA FILTRO P/ EVITAR RUIDOS
                                ; NOS BOTÕES
    BRA     VARRE               ; VOLTA PARA VARRER TECLADO

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                          TRATAMENTO DOS BOTÕES                         *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; NESTE TRECHO DO PROGRAMA ESTÃO TODOS OS TRATAMENTOS DOS BOTÕES

; ************************** TRATAMENTO DO BOTÃO 0 *****************************

TRATA_BOTAO_0
    MOVF    FILTRO_BOTOES,F     ; FILTRO JÁ IGUAL A ZERO?(FUNÇÃO JÁ FOI EXECUTADA?)
    BZ      VARRE               ; SIM - VOLTA P/ VARREDURA DO TECLADO
                                ; NÃO
    DECFSZ  FILTRO_BOTOES,F     ; FIM DO FILTRO ? (RUIDO?)
    BRA     VARRE               ; NÃO - VOLTA P/ VARRE
                                ; SIM - BOTÃO PRESSIONADO
    CLRF    CCPR2L              ; ZERA CCPR2L
    BCF     CCP2CON,5           ; ZERA OS BITS 5 e 4
    BCF     CCP2CON,4           ; (LSB DO DUTY CYCLE)
                                ; Tp = CCPR2L:CCP2CON<5,4>*Tosc*TMR2 Prescale
```

```
                                ; Tp = 0 * 250ns * 16
                                ; Tp = 0
                                ; PWM -> DUTY CYCLE = 0% -> OFF
BCF     RS                      ; SELECIONA O DISPLAY P/ COMANDOS
MOVLW   0XC8                    ; COMANDO PARA POSICIONAR O CURSOR
RCALL   ESCREVE                 ; LINHA 1 / COLUNA 8
BSF     RS                      ; SELECIONA O DISPLAY P/ DADOS

MOVLW   ' '                     ; COMANDOS PARA ESCREVER AS
RCALL   ESCREVE                 ; LETRAS DE " OFF"
MOVLW   'O'
RCALL   ESCREVE
MOVLW   'F'
RCALL   ESCREVE
MOVLW   'F'
RCALL   ESCREVE

BRA     VARRE                   ; VOLTA P/ VARREDURA DOS BOTÕES

; ************************** TRATAMENTO DO BOTÃO 1 ****************************

TRATA_BOTAO_1
    MOVF    FILTRO_BOTOES,F     ; FILTRO JÁ IGUAL A ZERO?(FUNÇÃO JÁ FOI EXECUTADA?)
    BZ      VARRE               ; SIM - VOLTA P/ VARREDURA DO TECLADO
                                ; NÃO
    DECFSZ  FILTRO_BOTOES,F     ; FIM DO FILTRO ? (RUIDO?)
    BRA     VARRE               ; NÃO - VOLTA P/ VARRE
                                ; SIM - BOTÃO PRESSIONADO
    MOVLW   0X80
    MOVWF   CCPR2L              ; CARREGA CCPR2L COM 0X80
    BCF     CCP2CON,5           ; LIMPA OS BITS 5 e 4
    BCF     CCP2CON,4           ; LSB DO DUTY CYCLE
                                ; Tp = CCPR2L:CCP2CON<5,4>*Tosc*TMR2 Prescale
                                ; Tp = 512 * 250ns * 16
                                ; Tp = 2,048ms
                                ; PWM -> DUTY CYCLE = 50%
    BCF     RS                  ; SELECIONA O DISPLAY P/ COMANDOS
    MOVLW   0XC8                ; COMANDO PARA POSICIONAR O CURSOR
    RCALL   ESCREVE             ; LINHA 1 / COLUNA 8
    BSF     RS                  ; SELECIONA O DISPLAY P/ DADOS

    MOVLW   ' '                 ; COMANDOS PARA ESCREVER AS
    RCALL   ESCREVE             ; LETRAS DE " 50%"
    MOVLW   '5'
    RCALL   ESCREVE
    MOVLW   '0'
    RCALL   ESCREVE
    MOVLW   '%'
    RCALL   ESCREVE

    BRA     VARRE                ; VOLTA P/ VARREDURA DOS BOTÕES

; ************************** TRATAMENTO DO BOTÃO 2 ****************************

TRATA_BOTAO_2
    MOVF    FILTRO_BOTOES,F     ; FILTRO JÁ IGUAL A ZERO?(FUNÇÃO JÁ FOI EXECUTADA?)
    BZ      VARRE               ; SIM - VOLTA P/ VARREDURA DO TECLADO
                                ; NÃO
    DECFSZ  FILTRO_BOTOES,F     ; FIM DO FILTRO ? (RUIDO?)
    BRA     VARRE               ; NÃO - VOLTA P/ VARRE
                                ; SIM - BOTÃO PRESSIONADO
    MOVLW   0XC0
    MOVWF   CCPR2L              ; CARREGA CCPR2L COM 0XC0
    BCF     CCP2CON,5           ; LIMPA OS BITS 5 e 4
```

```
BCF     CCP2CON,4         ; LSB DO DUTY CYCLE
                          ; Tp = CCPR2L:CCP2CON<5,4>*Tosc*TMR2 Prescale
                          ; Tp = 768 * 250ns * 16
                          ; Tp = 3,072ms
                          ; PWM -> DUTY CYCLE = 75%
BCF     RS                ; SELECIONA O DISPLAY P/ COMANDOS
MOVLW   0XC8              ; COMANDO PARA POSICIONAR O CURSOR
RCALL   ESCREVE           ; LINHA 1 / COLUNA 8
BSF     RS                ; SELECIONA O DISPLAY P/ DADOS

MOVLW   ' '               ; COMANDOS PARA ESCREVER AS
RCALL   ESCREVE           ; LETRAS DE "75%"
MOVLW   '7'
RCALL   ESCREVE
MOVLW   '5'
RCALL   ESCREVE
MOVLW   '%'
RCALL   ESCREVE

BRA     VARRE             ; VOLTA P/ VARREDURA DOS BOTÕES

; ************************* TRATAMENTO DO BOTÃO 3 ****************************

TRATA_BOTAO_3
    MOVF    FILTRO_BOTOES,F ; FILTRO JÁ IGUAL A ZERO?(FUNÇÃO JÁ FOI EXECUTADA?)
    BZ      VARRE           ; SIM - VOLTA P/ VARREDURA DO TECLADO
                            ; NÃO
    DECFSZ  FILTRO_BOTOES,F ; FIM DO FILTRO ? (RUIDO?)
    BRA     VARRE           ; NÃO - VOLTA P/ VARRE
                            ; SIM - BOTÃO PRESSIONADO
    MOVLW   0XFF
    MOVWF   CCPR2L          ; CARREGA CCPR2L COM 0XFF
    BSF     CCP2CON,5       ; SETA OS BITS 5 e 4
    BSF     CCP2CON,4       ; LSB DO DUTY CYCLE
                            ; Tp = CCPR2L:CCP2CON<5,4>*Tosc*TMR2 Prescale
                            ; Tp = 1023 * 250ns * 16
                            ; Tp = 4,092ms
                            ; PWM -> DUTY CYCLE = 99,90%
BCF     RS                  ; SELECIONA O DISPLAY P/ COMANDOS
MOVLW   0XC8                ; COMANDO PARA POSICIONAR O CURSOR
RCALL   ESCREVE             ; LINHA 1 / COLUNA 8
BSF     RS                  ; SELECIONA O DISPLAY P/ DADOS

MOVLW   '1'                 ; COMANDOS PARA ESCREVER AS
RCALL   ESCREVE             ; LETRAS DE "100%"
MOVLW   '0'
RCALL   ESCREVE
MOVLW   '0'
RCALL   ESCREVE
MOVLW   '%'
RCALL   ESCREVE

BRA     VARRE               ; VOLTA P/ VARREDURA DOS BOTÕES

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                          FIM DO PROGRAMA                                *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

    END                     ; FIM DO PROGRAMA
```

» Dicas e comentários

Como foi explicado na lógica do exemplo, deixamos o valor de **PR2** em 255 e o prescaler do Timer2 ajustado para 16. Com isso o PWM possui um período de 4,096 ms. Para podermos operar com o PWM em diversos níveis (0, 50 75 e 100%) foi necessário calcular os valores corretos a serem carregados em **CCPR2**. Desta forma, foi efetuada a seguinte conta:

CCPR2 = [(**PR2**)+1] x 4 x Porcentagem desejada

Neste caso:

- ✓ Porcentagem = 0%, 50%, 75% e 100%
- ✓ **PR2** = 255

Com isso obtivemos os seguintes valores:

- ✓ **Para 0%**: **CCPR2** = 0, sem arredondamento e sem erro;
- ✓ **Para 50%**: **CCPR2** = 512. Largura do pulso em 2,048 ms, resultando uma porcentagem final de 50,0%, também sem arredondamento e sem erro;
- ✓ **Para 75%**: **CCPR2** = 768. Largura do pulso em 3,072 ms, resultando novamente uma porcentagem final de 75,0% sem erros de arredondamento;
- ✓ **Para 100%**: **CCPR2** = 1023. Já neste caso, a largura do pulso ficou em 4,092 ms, resultando uma porcentagem final de 99,90%. Para 100%, **CCPR2** deve ser arredondado para cima, ultrapassando o valor do período e garantindo o pulso sempre em 1, porém isso não é possível, pois já estamos no limite máximo de 1023.

» Exercícios propostos

Agora que o exemplo foi estudado e esclarecido, aproveite para gerar novos problemas e soluções, seguindo os exercícios propostos:

1. Corrigir o problema encontrado no nível 100%, evitando que a saída seja colocada em zero, mesmo que por um período de tempo muito curto.
2. Em vez de trabalhar com somente quatro níveis de PWM, alterar o sistema para que um botão ligue e desligue a saída e outros dois botões incrementem e decrementem o PWM, de 50 a 100% com passos de 5%. Não recomendamos valores muito baixos para essa saída, pois o ventilador possui uma tensão mínima de trabalho.
3. Ativar as duas saídas PWMs ao mesmo tempo, uma para o ventilador e outra para a resistência. Utilizar dois botões para controlar o ajuste de cada uma delas.

10

Trabalho com as Memórias Não Voláteis

» Introdução

O estudo já está bem avançado e provavelmente o assunto deste capítulo você já conhece, pois os PICs 16F84 ou 16F628A (focos do livro Desbravando o PIC) também possuem esse tipo de memória. Mas não custa nada relembrarmos a matéria e explicarmos as características diferentes do modelo 18F4520, sendo a principal delas o acesso direto à memória de programa.

» Teoria e recursos do PIC

Quando uma memória é do tipo não volátil, significa que ela é capaz de continuar armazenando seus dados mesmo quando não energizada, ao contrário da RAM, que perde todas as informações quando o PIC é desligado. Isso é muito utilizado para sistemas que precisam manter dados ou configurações programadas pelo usuário. Uma discadora telefônica é um bom exemplo disso, pois o usuário só precisa programar os números que serão chamados uma única vez.

A grande diferença do PIC18F4520 em relação a outros modelos mais simples, como o 16F84 ou 16F62X, é que esse modelo possui acesso a dois tipos de memórias não voláteis:

Tipo	Tamanho	Observações
Dados (E^2PROM)	256 x 8 bits	A memória de dados é o tipo de memória convencional para armazenamento de informações do usuário. Ela é conhecida como E^2PROM.
Programa (FLASH)	32k x 8 bits	A memória de programa pode ser usada para alterar o próprio software ou mesmo como expansão da memória de dados. Por ser um tipo especial de E^2PROM, ela é conhecida como FLASH.

Muitas vezes uma memória E^2PROM externa pode ser necessária, principalmente quando os 256 bytes disponíveis não são suficientes. A vantagem é que a memória de programa também pode ser utilizada como expansão da memória de dados. O acesso à memória externa existente na placa proposta é estudado no próximo capítulo.

Muito interessante, não é? Quer dizer que podemos sair escrevendo nossos dados na memória de programa indefinidamente? Não é bem assim; alguns cuidados devem ser observados:

Editora Érica - Desbravando o Microcontrolador PIC18 Recursos Avançados - Daniel R. de Sousa, David José de Souza e Nicolás César Lavinia - 1ª Edição

- ✓ A memória de dados (E^2PROM) possui uma vida útil bem maior que a memória de programa (FLASH). Enquanto a primeira suporta 100.000 ciclos de escrita, a segunda possui uma vida útil assegurada de somente 10.000 ciclos. Quanto à velocidade, ambas respondem igualmente: 4 ms nominais para um ciclo de escrita (máximo 8 ms).
- ✓ Não podemos esquecer que a memória FLASH continua sendo uma "memória de programa". Isso significa que seu tamanho útil será somente a sobra em relação ao tamanho ocupado pelo programa propriamente dito.
- ✓ Por último, e o mais importante, é que o acesso à memória FLASH é irrestrito. Com isso, podemos acabar escrevendo onde não se deve. Caso seja utilizada, para armazenar um dado, uma posição da memória ocupada pelo programa, todo o sistema pode travar ou executar ações completamente aleatórias. Por isso, muita atenção ao utilizar essa memória, principalmente porque uma vez escrita, não adianta resetar o sistema para voltar ao normal.

Como já foi dito, a última observação é considerada a mais importante de todas. Por outro lado, ela possibilita também recursos extremamente avançados. Como assim? Bem, se temos acesso irrestrito a toda a área de programa, então podemos ler e escrever nela à vontade, certo? Isso mesmo. E com isso podemos criar sistemas capazes de se autoatualizarem. Podemos, por exemplo, elaborar um projeto que recebe um comando via serial (ligação com PC ou modem) e com isso entrar em modo de reconfiguração. Depois, ele recebe, pela mesma via, partes do programa que devem ser alteradas. O sistema pode então sofrer um upgrade pela porta serial. A grande vantagem disso é que não estamos limitados a alterar somente parâmetros gravados na E^2PROM de dados. Podemos alterar as funções do sistema.

Voltemos agora ao assunto principal: como ter acesso a essas memórias, sejam elas de dados ou de programa.

As diferenças para as rotinas existentes para os PICs são:

- ✓ Com a existência de um novo módulo interno (Table Pointer) a informação para a memória de programa agora é acessada pelos registradores **TBLAT**, onde podemos ler o dado disponível ou colocar o dado a ser escrito, e os registradores **TBLPTRU**, **TBLPTRH** e **TBLPTRL**, em que passamos o endereço da memória de programa onde vamos ler ou escrever o dado.
- ✓ Há novas instruções dedicadas para a operação de leitura e escrita da memória de programa **TBLRD** usada para operações de leitura e **TBLWT** para operações de escrita. Existem ainda operadores de ponteiro que auxiliam na operação de leitura ou escrita:

Instrução	Operação no Ponteiro (TBLPTR)
TBLRD* TBLWT*	Ponteiro não modificado.
TBLRD*+ TBLWT*+	Ponteiro é incrementado depois de uma operação de leitura/escrita.
TBLRD*- TBLWT*-	Ponteiro é decrementado depois de uma operação de leitura/escrita.
TBLRD+* TBLWT+*	Ponteiro é incrementado antes de uma operação de leitura/escrita.

- ✓ Os registradores **EECON1** e **EECON2** continuam existindo e possuem as mesmas funções: controlar as operações de escrita e leitura. Foram criados novos bits para controle das operações de leitura e escrita:

Bit	Descrição	Configuração
EECON1<EEPGD>	Seleção de acesso ao tipo de memória.	0 = acesso à memória EEPROM.
		1 = acesso à memória FLASH.
EECON1<CFGS>	Seleção de configuração do tipo de memória.	0 = acesso à memória FLASH ou EEPROM.
		1 = acesso aos registradores de configuração.
EECON1<FREE>	Apagar um bloco de memória (bulk erase).	0 = operação normal.
		1 = apaga um bloco de memória no próximo comando de escrita. O início do bloco é apontado pelo **TBLPTR**.

Escrita na E²PROM (dados)

A primeira ação que faremos em relação à E²PROM é a operação de escrita. Na verdade, ela é ligeiramente mais complexa que a operação de leitura, mas cronologicamente é mais interessante aprendermos primeiro a escrever. Afinal, como podemos ler algo se não escrevermos nada?

A complexidade da escrita é necessária para garantirmos a proteção do sistema e evitarmos escritas acidentais na memória. Por isso, a inicialização da escrita parece um pouco confusa e desnecessária, mas isso torna o sistema robusto e seguro.

A escrita da E²PROM deve seguir o roteiro:

1. O endereço para a escrita deve ser colocado em **EEADR**. Como existem 256 bytes disponíveis, esse endereço deve estar entre 0 e 255.
2. O dado a ser escrito deve ser colocado em **EEDATA**. Só podemos escrever um byte de cada vez.
3. Em seguida devemos ajustar a opção de trabalho para memória de dados (E²PROM) através de **EECON1<EEPGD>=0** e o acesso nos bits **EECON1<CFGS> = 0** .
4. A escrita deve ser habilitada através de **EECON1<WREN>=1**.
5. As interrupções devem ser desligadas para evitarmos conflitos, por meio de **INTCON<GIEH>=0**.
6. O registrador **EECON2** deve ser carregado com os valores **55h** e **AAh** sequencialmente. Esse procedimento é obrigatório e utilizado para a proteção da escrita.
7. A escrita deve ser iniciada através dos bits **EECON1<WR>=1** e **EECON1<WREN>=0**, nesta ordem.
8. As interrupções podem ser novamente ligadas com **INTCON<GIEH>=1**.
9. A operação de escrita é um pouco demorada e só termina quando o bit **EECON1<WR>** tiver sido limpo automaticamente pelo hardware. Por isso, normalmente ficamos esperando que essa ação aconteça. No caso de não podermos esperar pelo fim da escrita, é possível ligar a interrupção relacionada a esse evento através do bit **PIE2<EEIE>** e esperar que ela aconteça para considerarmos finalizada a escrita.
10. Caso algum erro ocorra durante a operação de escrita, o bit **EECON1<WRERR>** é setado (1). No caso de sucesso na operação esse bit é mantido em zero (0).

Leitura da E²PROM (dados)

Agora que você já sabe escrever algo nos endereços disponíveis da E²PROM, podemos ler essa informação de volta por meio de uma rotina de leitura. A leitura é muito mais simples, pois não necessita de tanta proteção. Também é muito mais rápida.

O roteiro para a criação da rotina de leitura é o seguinte:

1. O endereço para a escrita deve ser colocado em **EEADR**. Como existem 256 bytes disponíveis, esse endereço deve estar entre 0 e 255.
2. Em seguida devemos ajustar a opção de trabalho para memória de dados (E^2PROM) através de **EECON1<EEPGD>=0** e o acesso nos bits **EECON1<CFGS> = 0**.
3. A leitura deve ser ligada através do bit **EECON1<RD>=1**.
4. O dado lido é colocado em **EEDATA**.

Escrita na FLASH (programa)

Essa operação é um pouco mais complicada que a gravação da memória EEPROM. Assim como na versão 16F877A, possui uma limitação em relação à escrita na memória FLASH. Agora, os dados só podem ser escritos em blocos de 32 bytes, começando obrigatoriamente nos endereços terminados em 000000 (binário). Por causa disso, para escrever em um único endereço é necessário antes ler e armazenar temporariamente o valor existente nos outros 31 endereços para que eles não sejam perdidos. A complexidade é a mesma para garantir a segurança em relação à escrita acidental.

O roteiro para essa operação fica então da seguinte maneira:

1. O endereço para a escrita deve ser colocado em **TBLPTRU** (parte alta), **TBLPTRH** (parte central) e **TBLPTRL** (parte baixa). Como existem 32 k disponíveis, esse endereço deve estar entre 0 e 16.383 (3FFFh), mas deve ser utilizado obrigatoriamente um endereço terminado em 000000 (binário).
2. Devemos ajustar a opção de trabalho para memória de programa através de **EECON1<EEPGD>=1**.
3. Devemos ajustar a opção de configuração para memória de programa através de **EECON1<CFGS>=0**.
4. A escrita deve ser habilitada através de **EECON1<WREN>=1**.
5. Devemos ajustar a opção de apagamento do bloco de memória em **EECON1<FREE>=1**.
6. As interrupções devem ser desligadas para evitarmos conflitos, através de **INTCON<GIEH>=0**.
7. O registrador **EECON2** deve ser carregado com os valores **55h** e **AAh** sequencialmente. Esse procedimento é obrigatório e utilizado para a proteção da escrita.
8. O apagamento do bloco de memória deve então ser iniciado pelos bits **EECON1<WR>=1** e **EECON1<WREN>=0**, nesta ordem.
9. As interrupções podem ser novamente ligadas com **INTCON<GIEH>=1**.
10. Deve-se executar a instrução TBLRD*-. Note que não vamos ler o dado, é só para decrementar o endereço apontado em **TBLPTR**.
11. Devemos carregar o dado no registrador **TABLAT**.
12. Deve-se executar a instrução TBLWT+*. Note que primeiro é incrementado o **TBLPTR** e depois é realizado o envio do dado para finalização da escrita. Neste ponto devemos repetir os passos 11 e 12 até completar os 32 bytes máximos do bloco.
13. Devemos ajustar a opção de trabalho para memória de programa através de **EECON1<EEPGD>=1**.
14. Devemos ajustar a opção de configuração para memória de programa através de **EECON1<CFGS>=0**.
15. A escrita deve ser habilitada através de **EECON1<WREN>=1**.

16. As interrupções devem ser desligadas para evitarmos conflitos, através de **INTCON<GIEH>=0**. Aqui não existe problema quanto ao banco de memória.

17. O registrador **EECON2** deve ser carregado com os valores **55h** e **AAh** sequencialmente. Esse procedimento é obrigatório e utilizado para a proteção da escrita.

18. A escrita deve ser iniciada pelos bits **EECON1<WR>=1** e **EECON1<WREN>=0**, nesta ordem.

19. As interrupções podem ser novamente ligadas com **INTCON<GIEH>=1.**

20. A operação de escrita é um pouco demorada e só termina quando o bit **EECON1<WR>** tiver sido limpo automaticamente pelo hardware. Por isso, normalmente ficamos esperando que essa ação aconteça. No caso de não podermos esperar pelo fim da escrita, é possível ligar a interrupção relacionada a esse evento através do bit **PIE2<EEIE>** e esperar que ela aconteça para considerarmos finalizada a escrita.

21. Caso algum erro ocorra durante a operação de escrita, o bit **EECON1<WRERR>** é setado (1). No caso de sucesso na operação esse bit é mantido em zero (0).

Leitura da FLASH (programa)

A relação entre a leitura da memória de dados e a de programa é a mesma que a comentada para as operações de escrita, porém no caso da leitura podemos ler qualquer posição de memória.

O roteiro para a criação da rotina de leitura é da seguinte maneira:

1. O endereço para a escrita deve ser colocado em TBLPTRU (parte alta), TBLPTRH (parte central) e TBLPTRL (parte baixa). Como existem 32 k disponíveis, esse endereço deve estar entre 0 e 16.383 (3FFFh).

2. Deve-se executar a instrução TBLRD*.

3. O dado lido é colocado em **TABLAT**.

Tratamento da interrupção de final de escrita na E²PROM e FLASH

Essa interrupção deve ser usada em sistemas que não podem ficar parados esperando a operação de escrita terminar. Por isso, tão logo a operação de escrita seja iniciada, a rotina deve ser finalizada (não deve possuir o teste do bit **EECON1<WR>**, como no exemplo anterior). Quando a escrita for terminada, uma interrupção ocorre e o sistema pode tomar as ações pertinentes.

Resumo dos registradores associados à E²PROM/FLASH

Endereço	Nome	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
FF2h	INTCON	INTCON	GIE/ GIEH	PEIE/ GIEL	T0IE	INTE	RBIE	TMR0IF	INTF
F9Eh	PIR2	OSCFIF	CMIF	-	EEIF	BCLIF	HLVDIF	TMR3IF	CCP2IF
F9Dh	PIE2	OSCFIE	CMIE	-	EEIE	BCLIE	HLVDIE	TMR3IE	CCP2IE
FA2h	IPR2	OSCFIP	CMIP	-	EEIP	BCLIP	HLVDIP	TMR3IP	CCP2IP
FA6h	EECON1	EEPGD	CFGS	-	FREE	WRERR	WREN	WR	RD
FA7h	EECON2	Registrador de segurança para operações de escrita na E²PROM / FLASH							
FA8h	EEDATA	Dado a ser escrito na E²PROM							
FA9h	EEADR	Endereço de escrita da E²PROM							

Não usado para essa finalidade. Para obter mais informações, consulte o Apêndice A.

Para que essa interrupção possa ocorrer, sua chave individual **PIE2<EEIE>** deve estar ligada (1). Dentro da rotina de tratamento devemos testar o flag **PIR2<EEIF>** para sabermos se foi essa interrupção que ocorreu. Antes de sairmos do tratamento devemos limpar esse mesmo flag manualmente.

» Lógica do exemplo

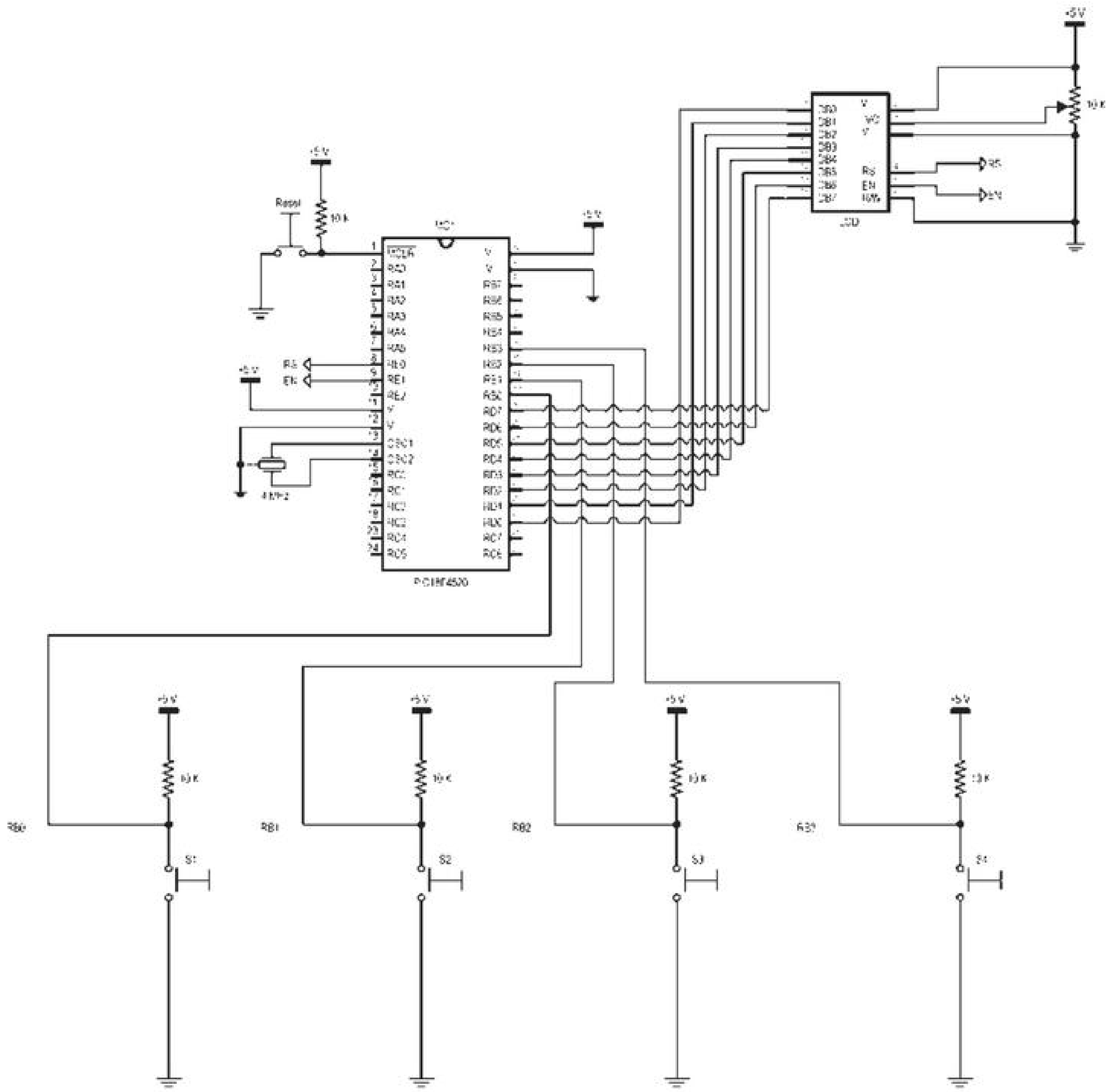
Este sistema divide o LCD em duas partes. Do lado esquerdo temos um valor relativo à memória de dados, variável de 0 a FFh (8 bits), com incremento e decremento rotativo através dos botões S2 e S3. Do lado direito o valor é para a memória de programa, também com incremento e decremento rotativo através dos botões S2 e S3, podendo ir de 0 a FFh (8 bits).

Para alterar o controle dos botões S2 e S3 entre o lado esquerdo e o lado direito deve ser usado o botão S1. Para o lado ativo, no momento, o valor é indicado entre os sinais > e <.

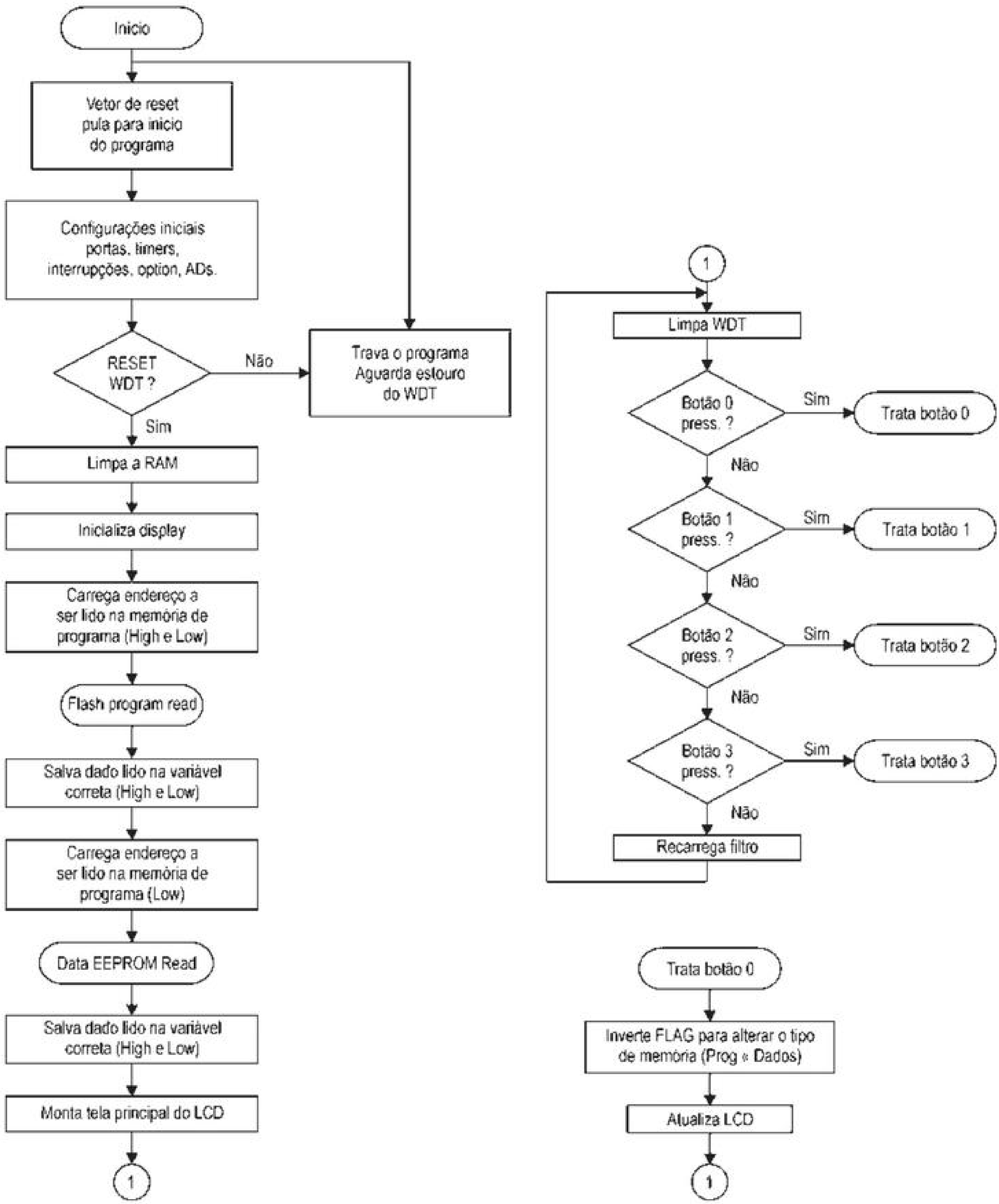
Depois de ajustados os valores desejados, basta pressionar o botão S4 para que ambos sejam gravados, cada um na memória correspondente.

Para verificar a gravação, altere os valores e resete o sistema. Os valores gravados são recuperados na inicialização e mostrados no LCD.

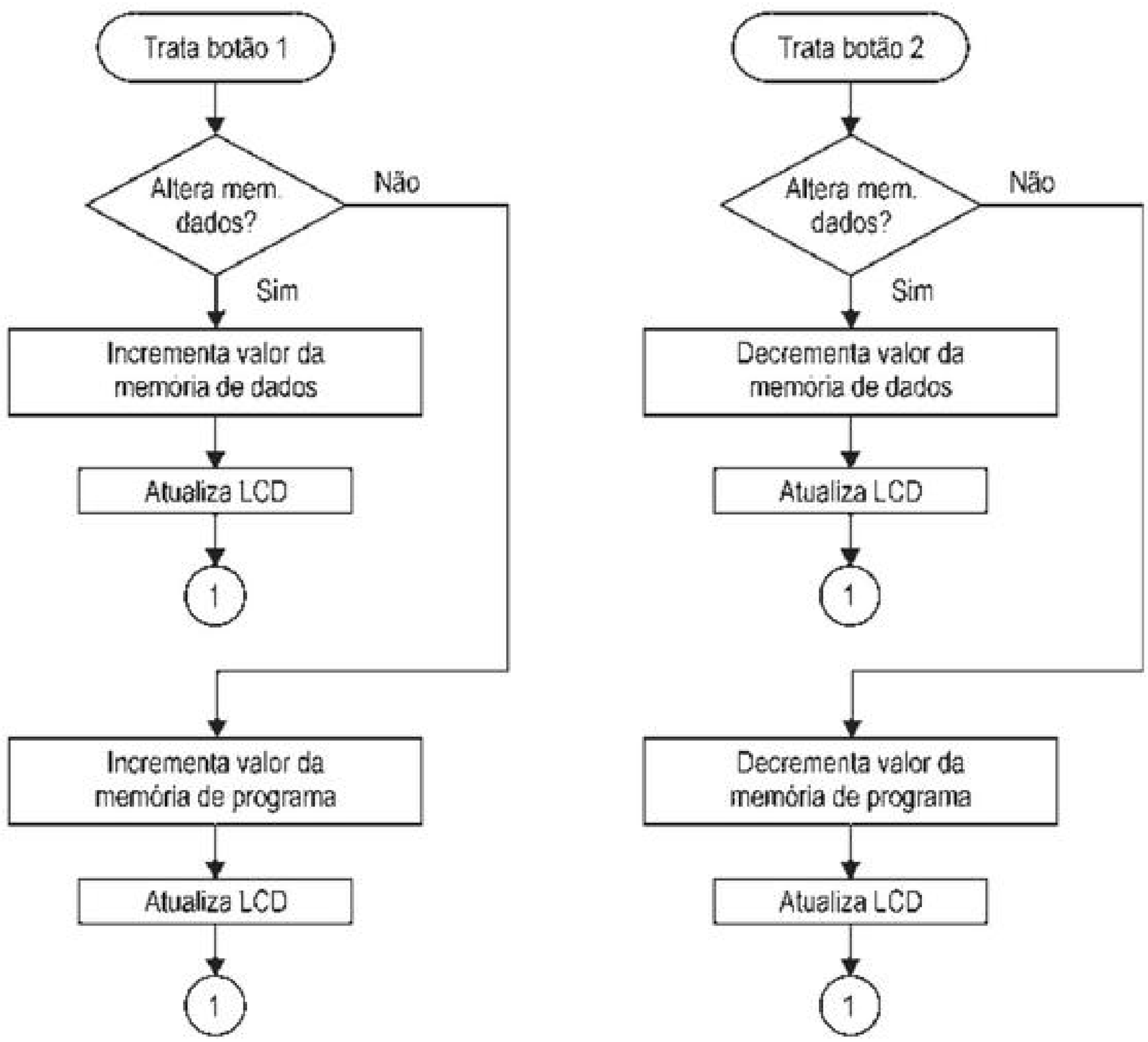
» Esquema elétrico



» Fluxograma



Editora Érica - Desbravando o Microcontrolador PIC18 Recursos Avançados - Daniel R. de Sousa, David José de Souza e Nicolás César Lavinia - 1ª Edição


Trata botão 1
Altera mem. dados?
Não
Sim
Incrementa valor da memória de dados
Atualiza LCD
1
Incrementa valor da memória de programa
Atualiza LCD
1
Trata botão 2
Altera mem. dados?
Não
Sim
Decrementa valor da memória de dados
Atualiza LCD
1
Decrementa valor da memória de programa
Atualiza LCD
1

Trata botão 3
Carrega endereço onde o dado será salvo na memória de programa (High e Low)
Carrega dado a ser salvo na memória de programa (High e Low)
Flash program write
Carrega endereço onde o dado será salvo na memória de programa (Low)
Carrega dado a ser salvo na memória de programa (Low)
Data EEPROM write
1

» Código

```
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *              DESBRAVANDO O PIC18 - RECURSOS AVANÇADOS                   *
; *                              EXEMPLO 7                                  *
; *                                                                         *
; * DANIEL RODRIGUES DE SOUSA, DAVID JOSÉ DE SOUZA e  NICOLÁS CÉSAR LAVINIA *
; *                                                                         *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *     DATA : 03/08/2009                                                   *
; *   VERSÃO : 1.0                                                          *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                            DESCRIÇÃO GERAL                              *
; *                                                                         *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTE EXEMPLO FOI ELABORADO PARA EXPLICAR O FUNCIONAMENTO DA LEITURA/ESCRITA
;  TANTO NA MEMÓRIA DE DADOS QUANTO NA MEMÓRIA DE PROGRAMA.
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                        CONFIGURAÇÕES PARA GRAVAÇÃO                      *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
    CONFIG  OSC = XT, FCMEN = OFF, IESO = OFF, PWRT = ON, BOREN = ON, BORV = 0
    CONFIG  WDT = ON, WDTPS = 128, MCLRE = ON, LPT1OSC = OFF, PBADEN = OFF
    CONFIG  CCP2MX = PORTC, STVREN = ON, LVP = OFF, DEBUG = OFF, XINST = OFF
    CONFIG  CP0 = OFF, CP1 = OFF, CP2 = OFF, CP3 = OFF, CPB = OFF, CPD = OFF
    CONFIG  WRT0 = OFF, WRT1 = OFF, WRT2 = OFF, WRT3 = OFF, WRTB = OFF
    CONFIG  WRTC = OFF, WRTD = OFF, EBTR0 = OFF, EBTR1 = OFF, EBTR2 = OFF
    CONFIG  EBTR3 = OFF, EBTRB = OFF

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                         DEFINIÇÃO DAS VARIÁVEIS                         *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
    CBLOCK  0X00            ; BANK 0 --> 0X00..0X5F --> ACCESS BANK

        FILTRO_BOTOES       ; FILTRO PARA RUIDOS
        TEMPO_TURBO         ; TEMPORIZADOR P/ TURBO DAS TECLAS

        TEMPO1
        TEMPO0              ; CONTADORES P/ DELAY

        FLAG                ; FLAG DE USO GERAL

        VALOR_DADOS         ; VALOR ARMAZENADO NA MEMÓRIA
                            ; DE DADOS

        VALOR_PROG          ; VALOR ARMAZENADO NA MEMÓRIA
                            ; DE PROGRAMA

        AUX                 ; REGISTRADOR AUXILIAR DE USO GERAL

        ENDERECO_HIGH       ; REGISTRADORES DE ENDEREÇO PARA
        ENDERECO_LOW        ; ACESSO À MEMÓRIA DE DADOS E PROGRAMA
                            ; MAPEADOS NO BANCO 0 DA RAM
        CONTADOR

        DADO_00

    ENDC

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                DEFINIÇÃO DAS VARIÁVEIS INTERNAS DO PIC                  *
```

```
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  O ARQUIVO DE DEFINIÇÕES DO PIC UTILIZADO DEVE SER REFERENCIADO PARA QUE
;  OS NOMES DEFINIDOS PELA MICROCHIP POSSAM SER UTILIZADOS, SEM A NECESSIDADE
;  DE REDIGITAÇÃO.

    #INCLUDE <P18F4520.INC> ; MICROCONTROLADOR UTILIZADO

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                          CONSTANTES INTERNAS                           *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  A DEFINIÇÃO DE CONSTANTES FACILITA A PROGRAMAÇÃO E A MANUTENÇÃO.

FILTRO_TECLA    EQU .200            ; FILTRO P/ EVITAR RUIDOS DOS BOTÕES

TURBO_TECLA     EQU .60             ; TEMPORIZADOR P/ TURBO DAS TECLAS

END_MEM_DADO    EQU    0X10         ; ENDEREÇO P/ LEITURA E GRAVAÇÃO
                                    ; NA MEMÓRIA DE DADOS

END_MEM_PROG_H  EQU HIGH  (0x1000)  ; ENDEREÇO P/ LEITURA E GRAVAÇÃO
END_MEM_PROG_L  EQU LOW   (0x1000)  ; NA MEMÓRIA DE PROGRAMA

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                   DECLARAÇÃO DOS FLAGs DE SOFTWARE                     *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  A DEFINIÇÃO DE FLAGs AJUDA NA PROGRAMAÇÃO E ECONOMIZA MEMÓRIA RAM.

#DEFINE TIPO_MEMORIA    FLAG,0  ; DEFINE A MEMORIA QUE ESTA SENDO
                                ; UTILIZADA
                                ; 1 -> MEMORIA DE PROGRAMA
                                ; 0 -> MEMORIA DE DADOS

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                              ENTRADAS                                  *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  AS ENTRADAS DEVEM SER ASSOCIADAS A NOMES PARA FACILITAR A PROGRAMAÇÃO E
;  FUTURAS ALTERAÇÕES DO HARDWARE.

#DEFINE BOTAO_0   PORTB,0           ; ESTADO DO BOTÃO 0
                                    ; 1 -> LIBERADO
                                    ; 0 -> PRESSIONADO

#DEFINE BOTAO_1   PORTB,1           ; ESTADO DO BOTÃO 1
                                    ; 1 -> LIBERADO
                                    ; 0 -> PRESSIONADO

#DEFINE BOTAO_2   PORTB,2           ; ESTADO DO BOTÃO 2
                                    ; 1 -> LIBERADO
                                    ; 0 -> PRESSIONADO

#DEFINE BOTAO_3   PORTB,3           ; ESTADO DO BOTÃO 3
                                    ; 1 -> LIBERADO
                                    ; 0 -> PRESSIONADO

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                              SAÍDAS                                    *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  AS SAÍDAS DEVEM SER ASSOCIADAS A NOMES PARA FACILITAR A PROGRAMAÇÃO E
;  FUTURAS ALTERAÇÕES DO HARDWARE.

#DEFINE DISPLAY     LATD            ; BARRAMENTO DE DADOS DO DISPLAY

#DEFINE RS          LATE,0          ; INDICA P/ O DISPLAY UM DADO OU COMANDO
                                    ; 1 -> DADO
                                    ; 0 -> COMANDO
```

```
#DEFINE ENABLE      LATE,1          ; SINAL DE ENABLE P/ DISPLAY
                                    ; ATIVO NA BORDA DE DESCIDA

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                    VETOR DE RESET DO MICROCONTROLADOR                 *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  POSIÇÃO INICIAL PARA EXECUÇÃO DO PROGRAMA

    ORG     0X0000          ; ENDEREÇO DO VETOR DE RESET
    BRA     CONFIGURACAO    ; PULA PARA CONFIG DEVIDO A REGIÃO
                            ; DESTINADA AS ROTINAS SEGUINTES

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                     ROTINA DE DELAY (DE 1MS ATÉ 256MS)                *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTA É UMA ROTINA DE DELAY VARIÁVEL, COM DURAÇÃO DE 1MS X O VALOR PASSADO
;  EM WORK (W).

DELAY_MS
    MOVWF   TEMPO1          ; CARREGA TEMPO1 (UNIDADES DE MS)
    MOVLW   .250
    MOVWF   TEMPO0          ; CARREGA TEMPO0 (P/ CONTAR 1MS)

    CLRWDT                  ; LIMPA WDT (PERDE TEMPO)
    DECFSZ  TEMPO0,F        ; FIM DE TEMPO0 ?
    BRA     $-.4            ; NÃO - VOLTA 2 INSTRUÇÕES
                            ; SIM - PASSOU-SE 1ms
    DECFSZ  TEMPO1,F        ; FIM DE TEMPO1 ?
    BRA     $-.12           ; NÃO - VOLTA 6 INSTRUÇÕES
    RETURN                  ; SIM - RETORNA

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                ROTINA DE ESCRITA DE UM CARACTERE NO DISPLAY           *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTA ROTINA ENVIA UM CARACTERE PARA O MÓDULO DE LCD. O CARACTERE A SER
;  ESCRITO DEVE SER COLOCADO EM WORK (W) ANTES DE CHAMAR A ROTINA.

ESCREVE
    MOVWF   DISPLAY         ; ATUALIZA DISPLAY (PORTD)
    NOP                     ; PERDE 1US PARA ESTABILIZAÇÃO
    BSF     ENABLE          ; ENVIA UM PULSO DE ENABLE AO DISPLAY
    BRA     $+.2            ; .
    BCF     ENABLE          ; .

    MOVLW   .1
    RCALL   DELAY_MS        ; DELAY DE 1ms
    RETURN                  ; RETORNA

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                ROTINA DE ESCRITA LINHA 1 DO LCD                       *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTA ROTINA ESCREVE A LINHA 1 DA TELA PRINCIPAL DO LCD, COM A FRASE:
;  LINHA 1 - "M.DADOS  M.PROG."

ATUALIZA_TELA_LINHA_1
    BCF     RS              ; SELECIONA O DISPLAY P/ COMANDOS
    MOVLW   0X80            ; COMANDO PARA POSICIONAR O CURSOR
    RCALL   ESCREVE         ; LINHA 0 / COLUNA 0
    BSF     RS              ; SELECIONA O DISPLAY P/ DADOS

    MOVLW   'M'             ; COMANDOS PARA ESCREVER AS
    RCALL   ESCREVE         ; LETRAS DE "M.DADOS  M.PROG."
    MOVLW   '.'
    RCALL   ESCREVE
    MOVLW   'D'
```

```
    RCALL   ESCREVE
    MOVLW   'A'
    RCALL   ESCREVE
    MOVLW   'D'
    RCALL   ESCREVE
    MOVLW   'O'
    RCALL   ESCREVE
    MOVLW   'S'
    RCALL   ESCREVE
    MOVLW   ' '
    RCALL   ESCREVE
    MOVLW   ' '
    RCALL   ESCREVE
    MOVLW   'M'
    RCALL   ESCREVE
    MOVLW   '.'
    RCALL   ESCREVE
    MOVLW   'P'
    RCALL   ESCREVE
    MOVLW   'R'
    RCALL   ESCREVE
    MOVLW   'O'
    RCALL   ESCREVE
    MOVLW   'G'
    RCALL   ESCREVE
    MOVLW   '.'
    RCALL   ESCREVE
    RETURN                  ; RETORNA

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *               ROTINA DE ESCRITA LINHA 2 DO LCD                      *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTA ROTINA ESCREVE A LINHA 2 DA TELA PRINCIPAL DO LCD.
;  A ROTINA LEVA EM CONTA TODAS AS VARIÁVEIS PERTINENTES P/ FORMAR A LINHA 2.

ATUALIZA_TELA_LINHA_2
    BCF     RS              ; SELECIONA O DISPLAY P/ COMANDO
    MOVLW   0XC1            ; COMANDO PARA POSICIONAR O CURSOR
    RCALL   ESCREVE         ; LINHA 1 / COLUNA 1

    BSF     RS              ; SELECIONA O DISPLAY P/ DADOS

    MOVLW   '>'
    BTFSC   TIPO_MEMORIA    ; ESTÁ UTILIZANDO A MEMÓRIA DE DADOS ?
    MOVLW   ' '             ; NÃO - ESCREVE ESPAÇO EM BRANCO
    RCALL   ESCREVE         ; SIM - ESCREVE ">" NO DISPLAY

    SWAPF   VALOR_DADOS,W   ; INVERTE NIBLE DO VALOR_DADOS
    ANDLW   B'00001111'     ; MASCARA BITS MAIS SIGNIFICATIVOS
    MOVWF   AUX             ; SALVA EM AUXILIAR

    MOVLW   0X0A
    SUBWF   AUX,W           ; AUX - 10d (ATUALIZA FLAG DE CARRY)
    MOVLW   0X30            ; CARREGA WORK COM 30h
    BTFSC   STATUS,C        ; RESULTADO É POSITIVO? (É UMA LETRA?)
    MOVLW   0X37            ; SIM - CARREGA WORK COM 37h
                            ; NÃO - WORK FICA COM 30h (NÚMERO)
    ADDWF   AUX,W           ; SOMA O WORK AO AUXILIAR
                            ; (CONVERSÃO ASCII)
    RCALL   ESCREVE         ; ENVIA CARACTERE AO DISPLAY LCD

    MOVF    VALOR_DADOS,W   ; CARREGA WORK COM VALOR_DADOS
    ANDLW   B'00001111'     ; MASCARA BITS MAIS SIGNIFICATIVOS
    MOVWF   AUX             ; SALVA EM AUXILIAR
```

```
MOVLW   0X0A
SUBWF   AUX,W           ; AUX - 10d (ATUALIZA FLAG DE CARRY)
MOVLW   0X30            ; CARREGA WORK COM 30h
BTFSC   STATUS,C        ; RESULTADO É POSITIVO? (É UMA LETRA?)
MOVLW   0X37            ; SIM - CARREGA WORK COM 37h
                        ; NÃO - WORK FICA COM 30h (NÚMERO)
ADDWF   AUX,W           ; SOMA O WORK AO AUXILIAR
                        ; (CONVERSÃO ASCII)
RCALL   ESCREVE         ; ENVIA CARACTERE AO DISPLAY LCD

MOVLW   'h'
RCALL   ESCREVE         ; ESCREVE "h" NO DISPLAY

MOVLW   '<'
BTFSC   TIPO_MEMORIA    ; ESTÁ UTILIZANDO A MEMÓRIA DE DADOS ?
MOVLW   ' '             ; NÃO - ESCREVE ESPAÇO EM BRANCO
RCALL   ESCREVE         ; SIM - ESCREVE "<" NO DISPLAY

MOVLW   ' '
RCALL   ESCREVE
MOVLW   ' '
RCALL   ESCREVE
MOVLW   ' '
RCALL   ESCREVE         ; ESPAÇOS EM BRANCO

MOVLW   '>'
BTFSS   TIPO_MEMORIA    ; ESTÁ UTILIZANDO A MEMÓRIA DE PROGRAMA?
MOVLW   ' '             ; NÃO - ESCREVE ESPAÇO EM BRANCO
RCALL   ESCREVE         ; SIM - ESCREVE ">" NO DISPLAY

SWAPF   VALOR_PROG,W    ; INVERTE NIBLE DO VALOR_PROG_HIGH
ANDLW   B'00001111'     ; MASCARA BITS MAIS SIGNIFICATIVOS
MOVWF   AUX             ; SALVA EM AUXILIAR
MOVLW   0X0A
SUBWF   AUX,W           ; AUX - 10d (ATUALIZA FLAG DE CARRY)
MOVLW   0X30            ; CARREGA WORK COM 30h
BTFSC   STATUS,C        ; RESULTADO É POSITIVO? (É UMA LETRA?)
MOVLW   0X37            ; SIM - CARREGA WORK COM 37h
                        ; NÃO - WORK FICA COM 30h (NÚMERO)
ADDWF   AUX,W           ; SOMA O WORK AO AUXILIAR
                        ; (CONVERSÃO ASCII)
RCALL   ESCREVE         ; ENVIA CARACTERE AO DISPLAY LCD

MOVF    VALOR_PROG,W    ; CARREGA WORK COM VALOR_PROG_HIGH
ANDLW   B'00001111'     ; MASCARA BITS MAIS SIGNIFICATIVOS
MOVWF   AUX             ; SALVA EM AUXILIAR

MOVLW   0X0A
SUBWF   AUX,W           ; AUX - 10d (ATUALIZA FLAG DE CARRY)
MOVLW   0X30            ; CARREGA WORK COM 30h
BTFSC   STATUS,C        ; RESULTADO É POSITIVO? (É UMA LETRA?)
MOVLW   0X37            ; SIM - CARREGA WORK COM 37h
                        ; NÃO - WORK FICA COM 30h (NÚMERO)
ADDWF   AUX,W           ; SOMA O WORK AO AUXILIAR
                        ; (CONVERSÃO ASCII)
RCALL   ESCREVE         ; ENVIA CARACTERE AO DISPLAY LCD

MOVLW   'h'
RCALL   ESCREVE         ; ESCREVE "h" NO DISPLAY

MOVLW   '<'
BTFSS   TIPO_MEMORIA    ; ESTÁ UTILIZANDO A MEMÓRIA DE PROGRAMA?
MOVLW   ' '             ; NÃO - ESCREVE ESPAÇO EM BRANCO
RCALL   ESCREVE         ; SIM - ESCREVE "<" NO DISPLAY
RETURN
```

```
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                ROTINA DE ESCRITA NA MEMÓRIA DE DADOS                      *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; ESTA ROTINA ESCREVE UM DADO (8 BITS) NA MEMÓRIA DE DADOS (E2PROM).
; O DADO A SER GRAVADO DEVE SER PASSADO PELO REGISTRADOR DADO_00.
; O ENDEREÇO DEVE SER PASSADO PELO REGISTRADOR ENDERECO_LOW.
; O REGISTRADOR ENDERECO_HIGH NÃO É UTILIZADO, POIS A MEMÓRIA TEM 256 ENDER.

DATA_EEPROM_WRITE
    MOVFF   DADO_00,EEDATA       ; SALVA DADO A SER GRAVADO EM EEDATA
    MOVFF   ENDERECO_LOW,EEADR   ; SALVA ENDERECO EM EEADR

    BCF     EECON1,EEPGD         ; APONTA P/ MEMÓRIA DE DADOS
    BCF     EECON1,CFGS          ; ACESSA EEPROM

    BSF     EECON1,WREN          ; HABILITA ESCRITA

    MOVLW   0X55
    MOVWF   EECON2               ; ESCREVE 0X55 EM EECON2 (OBRIGATÓRIO)
    MOVLW   0XAA
    MOVWF   EECON2               ; ESCREVE 0XAA EM EECON2 (OBRIGATÓRIO)
    BSF     EECON1,WR            ; INICIA ESCRITA

    BTFSC   EECON1,WR            ; ACABOU ESCRITA ?
    BRA     $-.2                 ; NÃO - AGUARDA FIM DA ESCRITA
                                 ; SIM
    BCF     EECON1,WREN          ; DESABILITA ESCRITAS NA MEMÓRIA
    RETURN                       ; RETORNA

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                ROTINA DE LEITURA NA MEMÓRIA DE DADOS                      *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; ESTA ROTINA LÊ UM DADO (8 BITS) DA MEMÓRIA DE DADOS (E2PROM).
; O DADO A SER LIDO É RETORNADO NO REGISTRADOR DADO_00.
; O ENDEREÇO DEVE SER PASSADO PELO REGISTRADOR ENDERECO_LOW.
; O REGISTRADOR ENDERECO_HIGH NÃO É UTILIZADO, POIS A MEMÓRIA TEM 256 ENDER.

DATA_EEPROM_READ
    MOVFF   ENDERECO_LOW,EEADR   ; SALVA ENDERECO EM EEADR

    BCF     EECON1,EEPGD         ; APONTA P/ MEMÓRIA DE DADOS
    BCF     EECON1,CFGS          ; ACESSA EEPROM
    BSF     EECON1,RD            ; HABILITA LEITURA

    MOVFF   EEDATA,DADO_00       ; SALVA DADO LIDO EM DADO_LOW_00
    RETURN                       ; RETORNA

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                ROTINA DE ESCRITA NA MEMÓRIA DE PROGRAMA                   *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; ESTA ROTINA ESCREVE UM DADO (8 BITS) NA MEMÓRIA DE PROGRAMA (FLASH).
; O DADO A SER GRAVADO DEVE SER PASSADO PELO REGISTRADOR DADO_00.
; O ENDEREÇO DEVE SER PASSADO PELO REGISTRADOR ENDERECO_LOW E ENDERECO_HIGH.
; ESTA MEMÓRIA DEVE SER ESCRITA EM BLOCOS (32 BITS MÁXIMO). PRIMEIRO DEVE-SE
; FAZER O BULK ERASE (BLOCO DE 64 BITS) E DEPOIS A GRAVAÇÃO DO DADO

FLASH_PROGRAM_WRITE
    MOVFF   ENDERECO_HIGH,TBLPTRH   ; CARREGA TBLPTR COM O ENDEREÇO DO BLOCO
    MOVFF   ENDERECO_LOW,TBLPTRL
    MOVLW   0X40                    ; GARANTE O INÍCIO DE UM BLOCO DE 64 BYTES
    ANDWF   TBLPTRL,F
```

```
;-----------------------------------------------------
; SEQUÊNCIA DE APAGAMENTO DE UM BLOCO (BULK ERASE)
;-----------------------------------------------------

    BSF     EECON1,EEPGD        ; APONTA P/ MEMÓRIA DE DADOS
    BCF     EECON1,CFGS         ; ACESSA MEMÓRIA FLASH
    BSF     EECON1,WREN         ; HABILITA ESCRITA DA MEMÓRIA
    BSF     EECON1,FREE         ; HABILITA OPERAÇÃO DE APAGAMENTO DA MEMÓRIA
    BCF     INTCON,GIE          ; DESABILITA INTERRUPÇÕES

    MOVLW   0X55
    MOVWF   EECON2              ; ESCREVE 0X55 EM EECON2 (OBRIGATÓRIO)
    MOVLW   0XAA
    MOVWF   EECON2              ; ESCREVE 0XAA EM EECON2 (OBRIGATÓRIO)
    BSF     EECON1,WR           ; INICIA APAGAMENTO DA MEMÓRIA
    BSF     INTCON,GIE          ; HABILITA INTERRUPÇÕES

    TBLRD*-                     ; FAZ UMA LEITURA DO DADO APONTADO EM TBLPTR
                                ; (O DADO É DESPREZADO) E DECREMENTA ENDEREÇO

;-----------------------------------------
; SEQUÊNCIA DE CARREGAMENTO DOS DADOS
;-----------------------------------------

    LFSR    FSR0,DADO_00        ; APONTA PARA BUFFER (ENDEREÇO DE DADO_00)

WRITE_BUFFER_BACK

    MOVLW   .1                  ; NÚMERO DE BYTES A SER CARREGADO PARA A
    MOVWF   CONTADOR            ; GRAVAÇÃO NA FLASH (ATÉ 32 BYTES)

WRITE_BYTE_TO_HREGS

    MOVFF   POSTINC0,WREG       ; CARREGA DADO APONTADO EM FSR0, GRAVA
    MOVWF   TABLAT              ; EM TABLAT E INCREMENTA FSR0
    TBLWT+*                     ; EXECUTA COMANDO DE ESCRITA PARA OS LATCH
                                ; INTERNO E INCREMENTA TBLPTR
    DECFSZ  CONTADOR            ; DECREMENTA CONTADOR. CONATDOR = 0?
    BRA     WRITE_BYTE_TO_HREGS ; NÃO, CONTINUA CARREGANDO OS DADOS

;-----------------------------------------
; SEQUÊNCIA DE ESCRITA DO BLOCO DE DADOS
;-----------------------------------------

PROGRAM_MEMORY                  ; SIM, EXECUTA PROCEDIMENTO PARA GRAVAÇÃO
    BSF     EECON1,EEPGD        ; APONTA P/ MEMÓRIA DE DADOS
    BCF     EECON1,CFGS         ; ACESSA MEMÓRIA FLASH
    BSF     EECON1,WREN         ; HABILITA ESCRITA DA MEMÓRIA
    BCF     INTCON,GIE          ; DESABILITA INTERRUPÇÕES

    MOVLW   0X55
    MOVWF   EECON2              ; ESCREVE 0X55 EM EECON2 (OBRIGATÓRIO)
    MOVLW   0XAA
    MOVWF   EECON2              ; ESCREVE 0XAA EM EECON2 (OBRIGATÓRIO)
    BSF     EECON1,WR           ; INICIA APAGAMENTO DA MEMÓRIA
    BSF     INTCON,GIE          ; HABILITA INTERRUPÇÕES
    BCF     EECON1,WREN         ; DESABILITA ESCRITA DA MEMÓRIA
    RETURN                      ; RETORNA DA SUBROTINA

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *              ROTINA DE LEITURA NA MEMÓRIA DE DADOS                    *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; ESTA ROTINA LÊ UM DADO DA MEMÓRIA DE PROGRAMA.
; O DADO LIDO É RETORNADO NO REGISTRADOR DADO_00
; O ENDEREÇO DEVE SER PASSADO PELOS REGIST. ENDERECO_HIGH E ENDERECO_LOW.
```

```
FLASH_PROGRAM_READ
    MOVFF   ENDERECO_HIGH,TBLPTRH   ; CARREGA TBLPTR COM O ENDEREÇO DO BLOCO
    MOVFF   ENDERECO_LOW,TBLPTRL

 READ_WORD

    TBLRD*                          ; EXECUTA COMANDO DE LEITURA
    MOVF    TABLAT,W                ; LÊ DADO APONTADO EM TBLPTR
    MOVWF   DADO_00                 ; CARREGA DADO EM DADO_00
    RETURN                          ; RETORNA

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                CONFIGURAÇÕES INICIAIS DE HARDWARE E SOFTWARE          *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  NESTA ROTINA SÃO INICIALIZADAS AS PORTAS DE I/O DO MICROCONTROLADOR E AS
;  CONFIGURAÇÕES DOS REGISTRADORES ESPECIAIS (SFR). A ROTINA INICIALIZA A
;  MÁQUINA E AGUARDA O ESTOURO DO WDT.

CONFIGURACAO
    CLRF    PORTA               ; GARANTE AS SAÍDAS EM ZERO
    CLRF    PORTB
    CLRF    PORTC
    CLRF    PORTD
    CLRF    PORTE

    CLRF    LATA
    CLRF    LATB
    CLRF    LATC
    CLRF    LATD
    CLRF    LATE

    MOVLW   B'11111111'
    MOVWF   TRISA               ; CONFIGURA I/O DO PORTA

    MOVLW   B'11111111'
    MOVWF   TRISB               ; CONFIGURA I/O DO PORTB

    MOVLW   B'11111101'
    MOVWF   TRISC               ; CONFIGURA I/O DO PORTC

    MOVLW   B'00000000'
    MOVWF   TRISD               ; CONFIGURA I/O DO PORTD

    MOVLW   B'00000100'
    MOVWF   TRISE               ; CONFIGURA I/O DO PORTE

    MOVLW   B'11011000'         ; CONFIGURA TIMER0
    MOVWF   T0CON               ; TIMER0 INCREM. PELO CICLO DE MÁQUINA
                                ; TIMER0- 1:1

    MOVLW   B'00000000'         ; CONFIGURA INTERRUPÇÕES
    MOVWF   INTCON              ; DESABILITADAS TODAS AS INTERRUPÇÕES

    MOVLW   B'00001111'         ; CONFIGURA CONVERSOR AD
    MOVWF   ADCON1              ; CONFIGURA PORTA, PORTB E PORTE COMO I/O DIGITAL

;  AS INSTRUÇÕES A SEGUIR FAZEM COM QUE O PROGRAMA TRAVE QUANDO HOUVER UM
;  RESET OU POWER-UP, MAS PASSE DIRETO SE O RESET FOR POR WDT. DESTA FORMA,
;  SEMPRE QUE O PIC É LIGADO, O PROGRAMA TRAVA, AGUARDA UM ESTOURO DE WDT
;  E COMEÇA NOVAMENTE. ISTO EVITA PROBLEMAS NO START-UP DO PIC.

    BTFSC   RCON,NOT_TO         ; RESET POR ESTOURO DE WATCHDOG TIMER ?
    BRA     $                   ; NÃO - AGUARDA ESTOURO DO WDT
                                ; SIM
```

```
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                          INICIALIZAÇÃO DA RAM                            *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTA ROTINA IRÁ LIMPAR TODA A RAM, DO BANCO 0 ATÉ O BANCO 15

    LFSR    FSR0,0X00        ; APONTA FSR0 PARA ENDEREÇO ZERO

LIMPA_RAM
    CLRF    POSTINC0         ; LIMPA A POSIÇÃO E INCREMENTA FSR0
    CLRWDT                   ; LIMPA WATCHDOG TIMER
    MOVLW   0X0F             ; TESTA SE JÁ CHEGOU AO BANCO 15
    SUBWF   FSR0H,W
    BNZ     LIMPA_RAM        ; SE NÃO, CONTINUA LIMPANDO

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                     CONFIGURAÇÕES INICIAIS DO DISPLAY                    *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; ESTA ROTINA INICIALIZA O DISPLAY P/ COMUNICAÇÃO DE 8 VIAS, DISPLAY PARA 2
; LINHAS, CURSOR APAGADO E DESLOCAMENTO DO CURSOR À DIREITA.

INICIALIZACAO_DISPLAY
    BCF     RS               ; SELECIONA O DISPLAY P/ COMANDOS

    MOVLW   0X30             ; ESCREVE COMANDO 0X30 PARA
    RCALL   ESCREVE          ; INICIALIZAÇÃO

    MOVLW   .3
    RCALL   DELAY_MS         ; DELAY DE 3MS (EXIGIDO PELO DISPLAY)

    MOVLW   0X30             ; ESCREVE COMANDO 0X30 PARA
    RCALL   ESCREVE          ; INICIALIZAÇÃO

    MOVLW   0X30             ; ESCREVE COMANDO 0X30 PARA
    RCALL   ESCREVE          ; INICIALIZAÇÃO

    MOVLW   B'00111000'      ; ESCREVE COMANDO PARA
    RCALL   ESCREVE          ; INTERFACE DE 8 VIAS DE DADOS

    MOVLW   B'00000001'      ; ESCREVE COMANDO PARA
    RCALL   ESCREVE          ; LIMPAR TODO O DISPLAY

    MOVLW   .1
    RCALL   DELAY_MS         ; DELAY DE 1MS

    MOVLW   B'00001100'      ; ESCREVE COMANDO PARA
    RCALL   ESCREVE          ; LIGAR O DISPLAY SEM CURSOR

    MOVLW   B'00000110'      ; ESCREVE COMANDO PARA INCREM.
    RCALL   ESCREVE          ; AUTOMÁTICO À DIREITA

    BSF     RS               ; SELECIONA O DISPLAY P/ DADOS

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                          INICIALIZAÇÃO DA RAM                            *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTE TRECHO DO PROGRAMA LÊ OS DADOS DAS MEMÓRIAS (E2PROM E FLASH) E
;  ATUALIZA A RAM.

LE_MEMORIA_PROGRAMA
    MOVLW   END_MEM_PROG_H
    MOVWF   ENDERECO_HIGH
    MOVLW   END_MEM_PROG_L
    MOVWF   ENDERECO_LOW         ; CARREGA ENDERECO P/ LEITURA

    RCALL   FLASH_PROGRAM_READ   ; CHAMA ROTINA P/ LER DADO
```

```
    MOVFF   DADO_00,VALOR_PROG  ; SALVA DADO LIDO EM VALOR_PROG

LE_MEMORIA_DADOS
    MOVLW   END_MEM_DADO
    MOVWF   ENDERECO_LOW        ; CARREGA ENDERECO P/ LEITURA

    RCALL   DATA_EEPROM_READ    ; CHAMA ROTINA P/ LER DADO

    MOVFF   DADO_00,VALOR_DADOS ; SALVA DADO LIDO EM VALOR_DADOS

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                ROTINA DE ESCRITA DA TELA PRINCIPAL                    *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTA ROTINA ESCREVE A TELA PRINCIPAL DO PROGRAMA, COM AS FRASES:
;  LINHA 1 - "M.DADOS  M.PROG."
;  LINHA 2 - " >xxh<    xxh "

    RCALL   ATUALIZA_TELA_LINHA_1   ; ATUALIZA TELA LINHA 1 DO LCD

    RCALL   ATUALIZA_TELA_LINHA_2   ; ATUALIZA TELA LINHA 2 DO LCD

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                          VARREDURA DOS BOTÕES                         *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; ESTA ROTINA VERIFICA SE ALGUM BOTÃO ESTÁ PRESSIONADO E CASO AFIRMATIVO
; DESVIA PARA O TRATAMENTO DELE.

VARRE
    CLRWDT                      ; LIMPA WATCHDOG TIMER

; **************** VERIFICA ALGUM BOTÃO PRESSIONADO **************************

VARRE_BOTOES
    BTFSS   BOTAO_0             ; O BOTÃO 0 ESTÁ PRESSIONADO ?
    BRA     TRATA_BOTAO_0       ; SIM - PULA P/ TRATA_BOTAO_0
                                ; NÃO
    BTFSS   BOTAO_1             ; O BOTÃO 1 ESTÁ PRESSIONADO ?
    BRA     TRATA_BOTAO_1       ; SIM - PULA P/ TRATA_BOTAO_1
                                ; NÃO
    BTFSS   BOTAO_2             ; O BOTÃO 2 ESTÁ PRESSIONADO ?
    BRA     TRATA_BOTAO_2       ; SIM - PULA P/ TRATA_BOTAO_2
                                ; NÃO
    BTFSS   BOTAO_3             ; O BOTÃO 3 ESTÁ PRESSIONADO ?
    BRA     TRATA_BOTAO_3       ; SIM - PULA P/ TRATA_BOTAO_3
                                ; NÃO

; ***************************** FILTRO P/ EVITAR RUIDOS ************************

    MOVLW   FILTRO_TECLA        ; CARREGA NO WORK O VALOR DE FILTRO_TECLA
    MOVWF   FILTRO_BOTOES       ; SALVA EM FILTRO_BOTOES
                                ; RECARREGA FILTRO P/ EVITAR RUIDOS
                                ; NOS BOTÕES
    MOVLW    .1
    MOVWF    TEMPO_TURBO        ; CARREGA TEMPO DO TURBO DAS TECLAS
                                ; COM 1 - IGNORA O TURBO A PRIMEIRA
                                ; VEZ QUE A TECLA É PRESSIONADA

    BRA     VARRE               ; VOLTA PARA VARRER TECLADO

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                          TRATAMENTO DOS BOTÕES                        *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; NESTE TRECHO DO PROGRAMA ESTÃO TODOS OS TRATAMENTOS DOS BOTÕES
```

```
; ************************** TRATAMENTO DO BOTÃO 0 ****************************

TRATA_BOTAO_0
    MOVF    FILTRO_BOTOES,F ; FILTRO JÁ IGUAL A ZERO?(FUNÇÃO JÁ FOI EXECUTADA?)
    BZ      VARRE           ; SIM - VOLTA P/ VARREDURA DO TECLADO
                            ; NÃO
    DECFSZ  FILTRO_BOTOES,F ; FIM DO FILTRO ? (RUIDO?)
    BRA     VARRE           ; NÃO - VOLTA P/ VARRE
                            ; SIM - BOTÃO PRESSIONADO

    BTG     TIPO_MEMORIA    ; INVERTE FLAG, ALTERA A MEMÓRIA UTILIZADA

    RCALL   ATUALIZA_TELA_LINHA_2   ; CHAMA ROTINA P/ ATUALIZAR LCD

    BRA     VARRE           ; VOLTA P/ VARREDURA DOS BOTÕES

; ************************** TRATAMENTO DO BOTÃO 1 ****************************

TRATA_BOTAO_1
    DECFSZ  FILTRO_BOTOES,F ; FIM DO FILTRO ? (RUIDO?)
    BRA     VARRE           ; NÃO - VOLTA P/ VARRE
                            ; SIM - BOTÃO PRESSIONADO
    DECFSZ  TEMPO_TURBO,F   ; FIM DO TEMPO DE TURBO ?
    BRA     VARRE           ; NÃO - VOLTA P/ VARRE
                            ; SIM
    MOVLW   TURBO_TECLA
    MOVWF   TEMPO_TURBO     ; RECARREGA TEMPORIZADOR DO TURBO
                            ; DAS TECLAS

    BTFSC   TIPO_MEMORIA    ; ESTÁ UTILIZANDO MEMÓRIA DE DADOS ?
    BRA     INC_MEM_PROG    ; NÃO - ENTÃO PULA P/ INC_MEM_PROG
                            ; SIM
INC_MEM_DADOS
    INCF    VALOR_DADOS,F   ; INCREMENTA VALOR_DADOS

    RCALL   ATUALIZA_TELA_LINHA_2   ; CHAMA ROTINA P/ ATUALIZAR LCD

    BRA     VARRE           ; VOLTA P/ VARREDURA DOS BOTÕES

INC_MEM_PROG
    INCF    VALOR_PROG,F    ; INCREMENTA VALOR_PROG_LOW

    RCALL   ATUALIZA_TELA_LINHA_2   ; CHAMA ROTINA P/ ATUALIZAR LCD

    BRA     VARRE           ; VOLTA P/ VARREDURA DOS BOTÕES

; ************************** TRATAMENTO DO BOTÃO 2 ****************************

TRATA_BOTAO_2
    DECFSZ  FILTRO_BOTOES,F ; FIM DO FILTRO ? (RUIDO?)
    BRA     VARRE           ; NÃO - VOLTA P/ VARRE
                            ; SIM - BOTÃO PRESSIONADO
    DECFSZ  TEMPO_TURBO,F   ; FIM DO TEMPO DE TURBO ?
    BRA     VARRE           ; NÃO - VOLTA P/ VARRE
                            ; SIM
    MOVLW   TURBO_TECLA
    MOVWF   TEMPO_TURBO     ; RECARREGA TEMPORIZADOR DO TURBO
                            ; DAS TECLAS

    BTFSC   TIPO_MEMORIA    ; ESTÁ UTILIZANDO MEMÓRIA DE DADOS ?
    BRA     DEC_MEM_PROG    ; NÃO - ENTÃO PULA P/ DEC_MEM_PROG
                            ; SIM
DEC_MEM_DADOS
    DECF    VALOR_DADOS,F   ; DECREMENTA VALOR_DADOS
```

```
    RCALL   ATUALIZA_TELA_LINHA_2   ; CHAMA ROTINA P/ ATUALIZAR LCD

    BRA     VARRE                   ; VOLTA P/ VARREDURA DOS BOTÕES

DEC_MEM_PROG
    DECF    VALOR_PROG,F            ; DECREMENTA VALOR_PROG_LOW

    RCALL   ATUALIZA_TELA_LINHA_2   ; CHAMA ROTINA P/ ATUALIZAR LCD

    BRA     VARRE                   ; VOLTA P/ VARREDURA DOS BOTÕES

; ************************** TRATAMENTO DO BOTÃO 3 ****************************

TRATA_BOTAO_3
    MOVF    FILTRO_BOTOES,F ; FILTRO JÁ IGUAL A ZERO?(FUNÇÃO JÁ FOI EXECUTADA?)
    BZ      VARRE           ; SIM - VOLTA P/ VARREDURA DO TECLADO
                            ; NÃO
    DECFSZ  FILTRO_BOTOES,F ; FIM DO FILTRO ? (RUIDO?)
    BRA     VARRE           ; NÃO - VOLTA P/ VARRE
                            ; SIM - BOTÃO PRESSIONADO

; ********TRECHO DO PROGRAMA PARA GRAVAR DADOS DA RAM NA MEMÓRIA *************

GRAVA_MEMORIA_PROGRAMA
    MOVLW   END_MEM_PROG_H
    MOVWF   ENDERECO_HIGH
    MOVLW   END_MEM_PROG_L
    MOVWF   ENDERECO_LOW        ; CARREGA ENDERECO ONDE O DADO SERÁ SALVO

    MOVFF   VALOR_PROG,DADO_00  ; EM DADO_HIGH_00 E DADO_LOW_00

    RCALL   FLASH_PROGRAM_WRITE ; CHAMA ROTINA DE GRAVAÇÃO

GRAVA_MEMORIA_DADOS
    MOVLW   END_MEM_DADO
    MOVWF   ENDERECO_LOW        ; CARREGA ENDERECO ONDE O DADO SERÁ SALVO

    MOVFF   VALOR_DADOS,DADO_00 ; CARREGA DADO A SER SALVO EM DADO_LOW_00

    RCALL   DATA_EEPROM_WRITE   ; CHAMA ROTINA DE GRAVAÇÃO

    BRA     VARRE               ; VOLTA P/ VARREDURA DOS BOTÕES

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                          FIM DO PROGRAMA                                *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

    END                         ; FIM DO PROGRAMA
```

» Dicas e comentários

Para a definição das posições de memória ser gravada, tanto para dados quanto para programa, foram especificadas três constantes no sistema com as seguintes inicializações:

- ✓ **END_MEM_DADOS** = 10h: grava a informação na memória de dados na posição 10h;
- ✓ **END_MEM_PROG_H** = 08h e **END_MEM_PROG_L** = 00h: gravam a informação na memória de programa na posição 0800h.

Para o uso das memórias foram criadas quatro rotinas, já que elas trabalham com quatro variáveis, todas elas definidas no banco 0:

Rotina	Função	Variáveis (Bank0)
DATA_EEPROM_READ	Lê um byte da memória de dados.	O endereço deve ser passado em ENDERECO_LOW. O dado lido é retornado em DADO_LOW.
DATA_EEPROM_WRITE	Escreve um byte na memória de dados.	O endereço deve ser passado em ENDERECO_LOW e o dado a ser escrito em DADO_LOW.
FLASH_PROGRAM_READ	Lê uma word (14 bits) da memória de programa.	O endereço deve ser passado em ENDERECO_HIGH e ENDERECO_LOW. O dado lido é retornado em DADO_HIGH e DADO_LOW.
FLASH_PROGRAM_WRITE	Escreve uma word (14 bits) na memória de programa.	O endereço deve ser passado em ENDERECO_HIGH e ENDERECO_LOW e o dado a ser escrito em DADO_HIGH e DADO_LOW.

» Exercícios propostos

Agora que o exemplo foi estudado e esclarecido, aproveite para gerar novos problemas e soluções, seguindo os exercícios propostos:

1. Criar um sistema no qual do lado esquerdo você continua informando o dado para a E^2PROM, no centro você escolhe a posição, de 0 a 255, e do lado direito é informada o tipo de operação: E para escrita e L para leitura. O botão S1 continua alterando entre os parâmetros a serem ajustados e os botões S2 e S3 alteram o parâmetro atual. O botão S4 efetua a operação: escreve na posição de memória especificada o valor ajustado ou leitura da posição especificada para indicação do valor encontrado. Depois, utilizar o gravador para ler a E^2PROM e verificar se a operação foi executada com sucesso.
2. Repetir o exercício anterior para a memória FLASH.

11

Comunicação Serial 1 - SPI e I²C

» Introdução

Este capítulo começa o estudo de um assunto muito procurado pelos profissionais atuais: a comunicação serial. Dissemos que começa o estudo porque o PIC possui tantos recursos voltados a ele que utilizamos dois capítulos para completá-lo. Nesta primeira parte, estudamos dois tipos distintos de comunicação serial, denominados SPI e I²C. Esses protocolos podem ser utilizados para a comunicação do PIC com outros microcontroladores, com memórias externas, drives de LCD, conversores, sensores e uma infinidade de outros periféricos.

Os dois protocolos mencionados (SPI e I²C) fazem parte de um sistema comum do PIC denominado MSSP (Master Synchronous Serial Port). Como esse sistema é comum, ele compartilha os mesmos recursos (pinos, hardware interno e registradores) entre os dois casos. Por isso só podemos utilizar um sistema de cada vez. Eles serão estudados separadamente.

Comecemos, então, conhecendo as diferenças e semelhanças básicas entre eles:

SPI	I²C
Foi criado pela Motorola e seu nome significa Serial Peripherall Interface	Foi criado pela Philips e seu nome significa Inter-Integrade Circuit
Opera em modo Master e Slave	Opera em modo Master e Slave
Comunicação síncrona (com Clock)	Comunicação síncrona (com Clock)
A comunicação é feita por três vias: ✓ Clock (SCK) ✓ Data In (SDI) ✓ Data Out (SDO)	A comunicação é feita por duas vias: ✓ Clock (SCL) ✓ Data In/Out (SDA)
Não endereçável	Endereçável (7 ou 10 bits)
8 bits de dados	8 bits de dados

» Teoria e recursos do PIC para SPI

Antes de mais nada precisamos saber alguns conceitos básicos da comunicação serial síncrona. Como o próprio nome diz, essa comunicação exige uma via de sincronismo entre os dois componentes envolvidos (receptor e transmissor) para que haja a marcação do momento exato da transmissão/recepção de cada bit.

Editora Érica - Desbravando o Microcontrolador PIC18 Recursos Avançados - Daniel R. de Sousa, David José de Souza e Nicolás César Lavinia - 1ª Edição

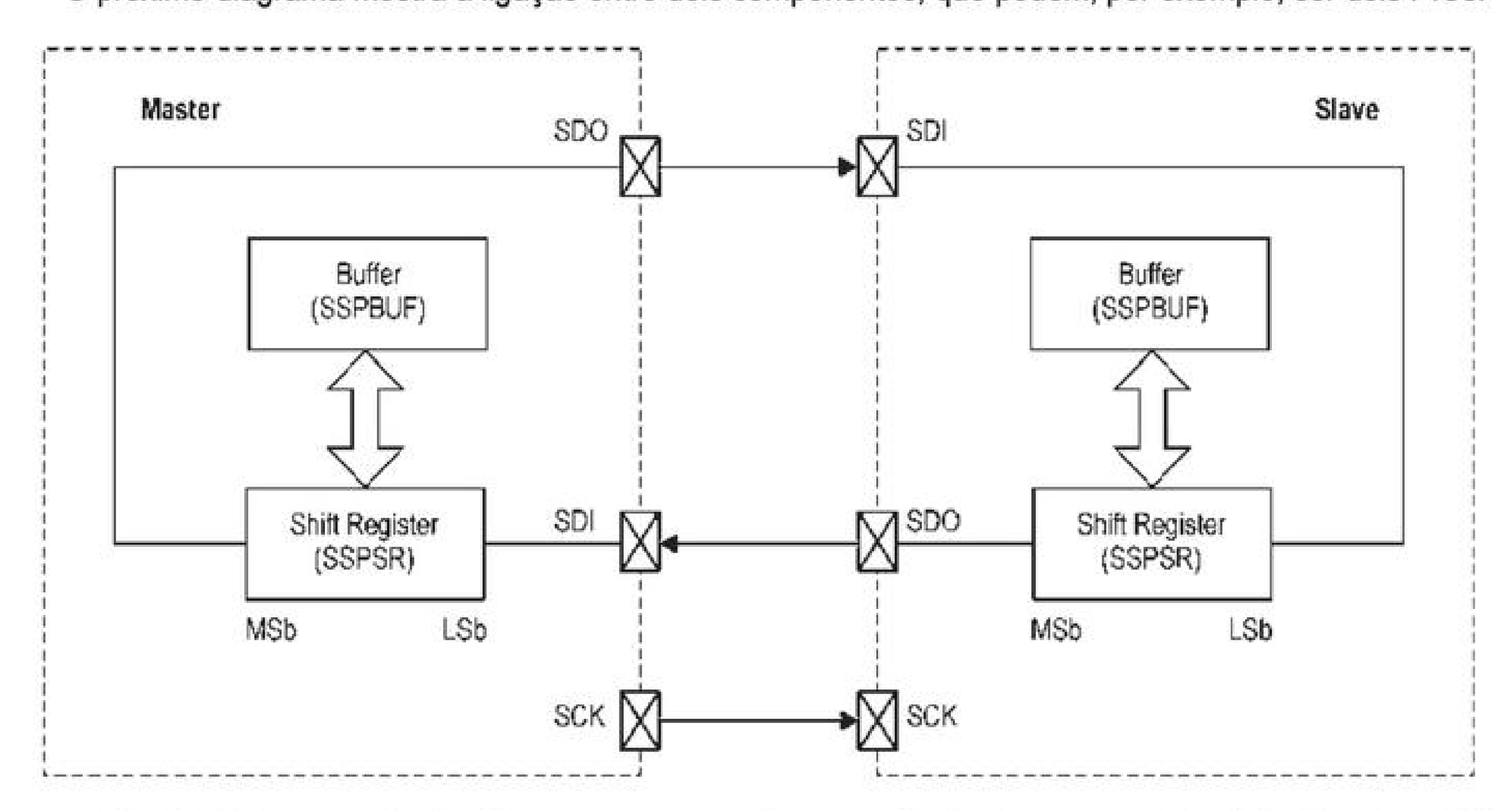
Isso é feito através do clock, que nada mais é que uma onda quadrada para sincronizar o sistema. Acontece que só pode existir um clock, e por isso não podemos deixar ambos os lados da comunicação responsáveis por ele. Para resolver esse problema foi criado o conceito de Master e Slave. O componente denominado Master (Mestre) é responsável pelo controle da comunicação, gerando o clock de sincronismo. Do outro lado, o componente chama-se Slave (Escravo) e fica sempre aguardando o clock enviado pelo mestre.

Como a SPI não permite o endereçamento, a comunicação só pode ser feita entre dois pontos, sendo um deles o Master e o outro o Slave.

Como é observado na tabela comparativa apresentada anteriormente, nesse modo de comunicação a ligação entre os componentes deve ser feita por três vias:

- ✓ **Clock**: trata-se da via de clock, que pode ser uma entrada (Slave) ou saída (Master). O SPI aceita os quatro tipos de clocks diferentes especificados para o padrão SPI. A descrição e a configuração desses tipos são vistas posteriormente. No PIC18F4520 é chamado de **SCK** e está localizado no pino 18 (RC3).
- ✓ **Data In**: é a entrada de dados, ou seja, a via de recepção. No PIC18F4520 é chamada de **SDI** e está localizada no pino 23 (**RC4**).
- ✓ **Data Out**: trata-se da saída de dados, ou seja, a via de transmissão. No PIC18F4520 é chamada de **SDO** e está localizada no pino 24 (**RC5**).

O próximo diagrama mostra a ligação entre dois componentes, que podem, por exemplo, ser dois PICs:



O PIC do lado esquerdo é o Master e, por isso, é responsável pela geração do clock. Seu pino **SCK** deve ser configurado como saída. Já do lado direito o PIC é definido como Slave, ficando o pino de clock configurado como entrada.

Pelo diagrama podemos também começar a entender melhor o funcionamento interno do SPI. Esse protocolo de comunicação trabalha com um registrador interno (não acessível) denominado **SSPSR**, que é do tipo rotativo (Shift Register). A cada ciclo do clock, um bit lido da porta **SDI** é escrito nesse registrador, entrando pelo bit menos significativo e rotacionando todos os bits para a esquerda. Ao mesmo tempo, o bit mais significativo é enviado para a porta **SDO**. Por isso, enquanto estamos recebendo um byte estamos também transmitindo outro. O registrador **SSPBUF** é o responsável pelo valor a ser transmitido e também pelo byte recebido.

Editora Érica - Desbravando o Microcontrolador PIC18 Recursos Avançados - Daniel R. de Sousa, David José de Souza e Nicolás César Lavinia - 1ª Edição

A comunicação funciona da seguinte maneira:

1. Quem manda na comunicação é o Master. Por isso, um valor qualquer é escrito no registrador **SSPBUF** desse PIC.
2. O valor é movido para o registrador **SSPSR**.
3. Em seguida oito pulsos são gerados na saída de clock **SCK** (cada pulso transmite 1 bit).
4. Ao término dos oito pulsos o valor que estava no $\mathbf{SSPSR}_{\mathbf{Master}}$ é trocado com o valor que estava no $\mathbf{SSPSR}_{\mathbf{Slave}}$.
5. Tanto de um lado quanto do outro o valor de **SSPSR** é movido novamente para o registrador **SSPBUF**.
6. Também de ambos os lados flags são ativados para informar ao sistema o fim da comunicação. Os dados podem então ser acessados novamente em **SSPBUF**.

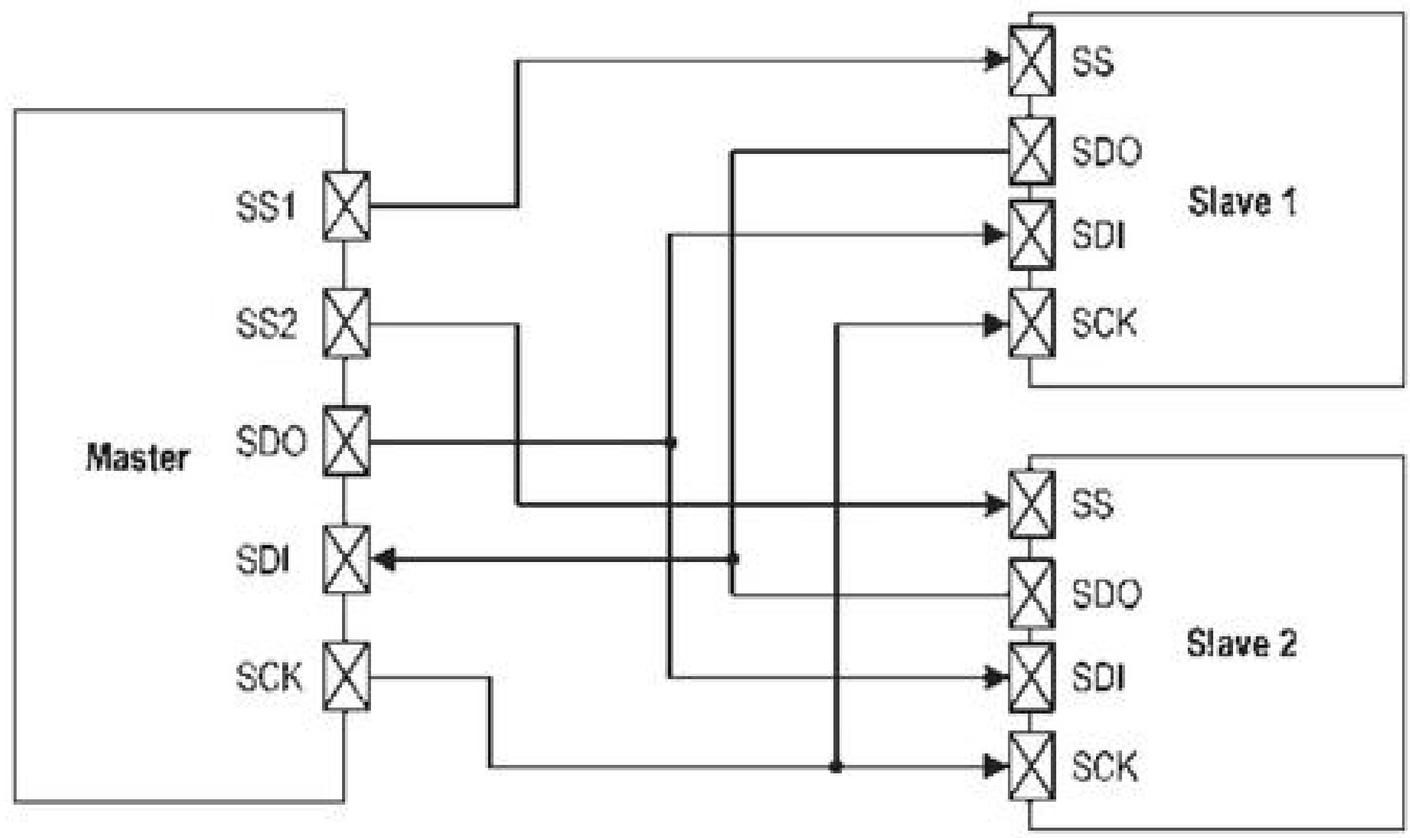
Bem simples, não é mesmo? Desta forma, a comunicação é feita sempre em sentido duplo, ao mesmo tempo. Assim, três situações diferentes podem acontecer:

- ✓ **Master envia dado válido/Slave envia dado não válido**: neste caso o Master está enviando algum dado para o Slave, porém não interessa a informação que é enviada pelo Slave nesse mesmo momento. É muito comum quando o Master está fazendo uma pergunta ao Slave que terá de ser processada antes de a resposta ser transmitida de volta. Neste caso, o dado recebido pelo Master ao final da transmissão deve ser descartado.
- ✓ **Master envia dado não válido/Slave envia dado válido**: este caso é complementar ao anterior. Depois que o Master efetuou uma pergunta e o Slave a processou, uma resposta deve ser retornada. Acontece que como é o Master quem manda no clock, ele precisa disparar uma nova transmissão para poder receber o dado do Slave.
- ✓ **Master envia dado válido/Slave envia dado válido**: pode ser o caso mais raro, mas é usado para aumento da velocidade. Por exemplo, o Master faz uma pergunta e ao mesmo tempo recebe a resposta da pergunta anterior.

Obviamente, apesar de o sistema de comunicação ser de transmissão e recepção ao mesmo tempo, nem sempre isso é necessário. Caso a comunicação seja em um só sentido, a via de transmissão do Slave (**SDO**) pode ser desativada configurando-se esse pino como entrada, a recepção do Master (**SDI**) sem uso.

Existe também um outro pino denominado **/SS** (**RA5**) que pode ser usado do lado Slave. Na maioria dos casos, esse pino é opcional e serve como um sinal de Chip Select. Sua função é muito simples. Quando esse pino está em nível baixo (0), o sistema de comunicação funciona normalmente, com o pino **SDO** funcionando como saída de dados a cada pulso do clock. No entanto, se esse pino for colocado em nível alto (1), o sistema de comunicação é desligado e o pino **SDO** fica flutuante. Com esse recurso podemos montar uma rede de comunicação com um Master e vários Slaves, desde que o Master controle individualmente os pinos **/SS** de cada um dos Slaves. Esse controle deve ser implementado manualmente no software e garantir que somente um Slave está ativado de cada vez, evitando conflitos nas saídas **SDO**.

Quando configurado para Master, o pino **/SS** opera como um I/O normal (RA5).



Vejamos agora como configurar corretamente o sistema para efetuar uma transmissão.

O primeiro passo a ser feito diz respeito à configuração correta dos pinos, como entradas ou saídas:

Pino	Configuração
SDI	Esse pino é configurado automaticamente pelo sistema como **entrada.**
SDO	Esse pino deve ser configurado manualmente como **saída** através do **TRISC**.
SCK	Esse pino deve ser configurado como **saída** no Master e **entrada** do Slave, através do **TRISC**.
/SS	Quando habilitado no Slave, deve ser configurado manualmente como **entrada**.

Depois devemos configurar a comunicação como Slave ou Master. Existe mais de uma opção para cada tipo, devendo ser configuradas em **SSPCON1<SSPM3:SSPM0>**:

SSPM3: SSPM0	Descrição
0000	SPI Master, com clock = F_{osc} / 4
0001	SPI Master, com clock = F_{osc} / 16
0010	SPI Master, com clock = F_{osc} / 64
0011	SPI Master, com clock = TMR2 / 2
0100	SPI Slave, com **/SS** habilitado
0101	SPI Slave, com **/SS** desabilitado.
Observação: As demais opções de configurações dizem respeito aos outros modos (I^2C).	

As quatro primeiras opções dizem respeito à escolha do modo Master, cada uma com uma frequência diferente para o clock. Nas três primeiras, o clock é uma divisão (4, 16 ou 64) da frequência do oscilador do próprio PIC. Com isso, podemos deduzir que na frequência máxima de 40 MHz a taxa de transmissão é de 10,0 Mbps. Na outra opção, o clock é gerado com base na metade do **TMR2**. Assim, a base de tempo será **PR2** / 2. O postscale não é usado para esse caso.

As duas últimas opções devem ser escolhidas quando estamos configurando um PIC em modo Slave. Neste caso devemos escolher entre trabalhar ou não com o pino **/SS**. Mais à frente se comenta um caso em que a ativação desse pino é obrigatória.

Outra configuração ajusta a condição de recebimento. Assim, o sistema pode efetuar a leitura da porta **SDI** em dois pontos distintos, conforme o valor imposto em **SSPSTAT<SMP>**:

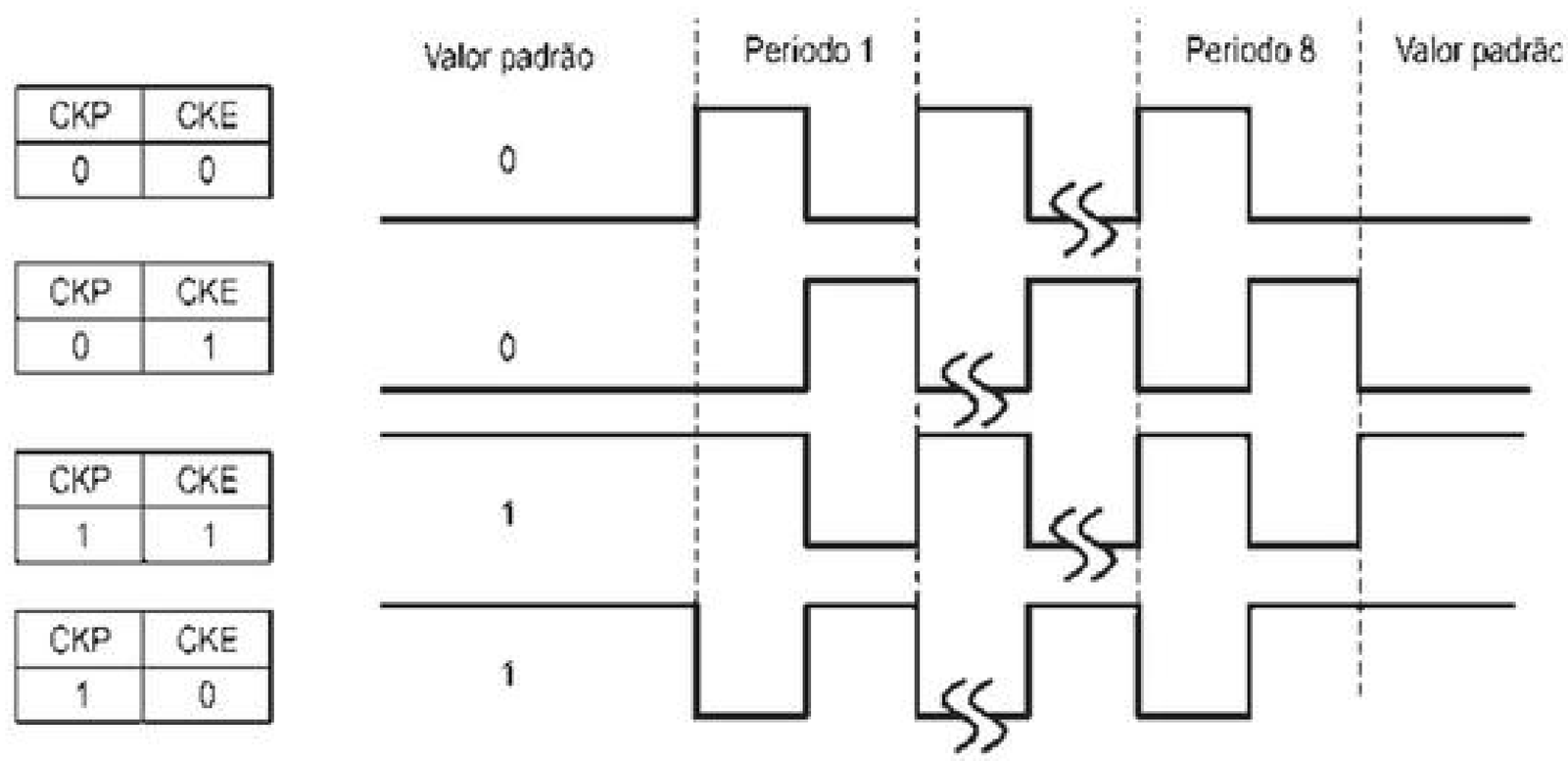
SMP	Descrição
0	A recepção é feita na borda do meio do período. Pode ser usada tanto no Master quanto no Slave.
1	A recepção é feita na borda do final do período. Só pode ser usada no Master.

O próximo passo diz respeito à escolha da forma de onda do clock. O padrão SPI disponibiliza quatro opções que podem ser ajustadas em dois bits distintos, **SSPCON1<CKP>** e **SSPSTAT<CKE>**:

CKP	Descrição
0	Clock normalmente em nível baixo (0)
1	Clock normalmente em nível alto (1)

CKE	Descrição
0	Pulso do clock no começo do período
1	Pulso do clock no final do período

Graficamente fica muito mais fácil percebermos as diferenças entre os quatro modos resultantes da combinação desses 2 bits:



Para o Master não existe limitação quanto ao uso de qualquer um dos quatro tipos. Para o Slave, a questão é um pouco mais complicada.

Como os pinos de clock estão diretamente ligados, então o valor padrão da via (nível alto ou baixo) deve ser o mesmo. Desta forma o valor de **CKP** para o Slave deve ser obrigatoriamente o mesmo adotado para o Master.

O outro problema está em relação à transmissão/recepção do dado pelo Slave. Para o Master, o primeiro bit (o mais significativo) é colocado em **SDO** logo depois do começo do período, permanecendo lá até o final dele. Desta forma, podemos considerar que o bit é transmitido no meio do período. Como ele sabe o momento exato de início do período, fica fácil também controlar a leitura da entrada **SDI** (no começo ou no fim). Para o Slave, entretanto, um problema acontece quando **CKE=1**, pois não há como ele saber o início do período, uma vez que não existe nenhuma borda nesse ponto. Por este motivo, para utilizarmos o Slave com **CKE=1**, obrigatoriamente temos de habilitar o pino **/SS**. Quando o pino de **/SS** for colocado em nível baixo, o primeiro bit é colocado em **SDO** (até este momento ele estava flutuante) e o sistema fica aguardando o clock para dar continuidade à transmissão. Quanto à recepção, como o Slave só pode operar com leitura no meio do período (**SMP=0**), a entrada **SDI** será lida logo na primeira borda do clock.

Para ligar a comunicação SPI, basta ativar o sistema SSP através do bit **SSPCON1<SSPEN>**:

SSPEN	Descrição
0	Desabilita a comunicação.
1	Habilita a comunicação. Nesse momento, os pinos **SCK**, **SDO**, **SDI**, e em alguns casos o **/SS**, deixam de operar como I/Os convencionais.

Por último, basta escrever o dado que deseja transmitir no registrador **SSPBUF**. Se isso for feito no Master, a transmissão será iniciada imediatamente após a escrita. Para o Slave, o sistema fica aguardando o clock. No término dos oito pulsos do clock, o dado terá sido enviado do Master para o Slave e vice-versa, como já explicado. Os registradores **SSPBUF** são, então, atualizados e o flag **SSPSTAT<BF>** indica que o buffer está cheio:

BF	Descrição
0	Recebimento em operação. **SSPBUF** está vazio.
1	Recebimento terminado. **SSPBUF** carregado.

Esse bit é limpo automaticamente toda vez que o registrador **SSPBUF** é lido, indicando que o programa já processou a informação recebida.

Nesse mesmo momento o bit da interrupção **PIR1<SSPIF>** é ativado. A interrupção só acontece no caso de as suas chaves estarem ligadas.

Caso seja escrito outro valor em **SSPBUF** antes do término da transmissão/recepção, ele será ignorado, não afetando a operação. Entretanto, um bit é setado informando uma colisão de informações. Esse bit é denominado **SSPCON1<WCOL>** e deve ser limpo manualmente pelo programa. Por outro lado, caso o Slave receba um novo valor antes de o **SSPBUF** ser lido (**BF=1**), esse novo valor é perdido e o flag de overflow ativado (**SSPCON1<SSPOV>=1**). Esse bit também deve ser limpo manualmente. Para o Master esse flag não possui função.

Quanto ao modo SLEEP, o resultado é diferente quando aplicado ao Master ou ao Slave. No caso do Master, a comunicação é completamente interrompida, pois não é possível a geração do clock sem o oscilador estar funcionando. Por outro lado, para o Slave, a comunicação continua funcionando e o PIC pode acordar caso a interrupção de SSP esteja ligada.

Resumo dos registradores associados a SPI

Endereço	Nome	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
FF2h	INTCON	INTCON	GIE/ GIEH	PEIE/ GIEL	T0IE	INTE	RBIE	TMR0IF	INTF
F9Eh	PIR1	PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
F9Dh	PIE1	PSPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
F9Fh	IPR1	PSPIP	ADIP	RCIP	RXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP
FC7h	SSPSTAT	SMP	CKE	D/A	P	S	R/W	UA	BF
FC6h	SSPCON1	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
FC9h	SSPBUF	Buffer de transmissão/recepção							
F92h	TRISA	Configuração do PORTA com Entrada(1) ou Saída(0)							
F94h	TRISC	Configuração do PORTC com Entrada(1) ou Saída(0)							

Não usado para essa finalidade. Para obter mais informações, consulte o Apêndice A.

» Teoria para I²C

O padrão de comunicação I²C possui características bem diferentes do SPI. Isso o torna muito mais poderoso, mas, em contrapartida, muito mais complexo também. Vejamos suas principais características:

- ✓ Esse protocolo continua com o conceito de Master/Slave, entretanto ele permite o endereçamento de diversos pontos da rede por meio das próprias vias de comunicação. Com isso podemos ter um Master com diversos Slaves sem necessitarmos de pinos de controle adicionais. Na verdade, é possível também a estruturação de uma rede com diversos Mestres.
- ✓ A comunicação é feita somente em duas vias: Clock e Data. Como também se trata de uma comunicação síncrona, a via de clock continua sendo indispensável. Quanto aos dados, eles transitam em uma única via, tanto para a transmissão quanto para a recepção.
- ✓ Tanto o Master quanto o Slave podem transmitir e receber dados, mas o controle é sempre do Master.
- ✓ Para evitar conflitos na via de dados (**SDA**), os pinos são chaveados como entrada ou saída (impõe somente o nível baixo, forçando o GND), conforme a necessidade imposta pelo protocolo. Por esse fato, um resistor de pull-up é necessário, sendo recomendados valores de 10 kΩ a 1 kΩ. Os valores mais baixos (até 1 kΩ) podem ser necessários para velocidades muito elevadas ou uma rede com muitos periféricos.
- ✓ O clock (**SCL**) continua sendo totalmente controlado pelo Mestre. Acontece que, nesse protocolo, pelo compartilhamento de uma única via de dados e pela possibilidade de mais de um Mestre, o clock necessário para a resposta do Escravo deve ser dado no momento correto, evitando conflitos no processamento da informação. Para que isso seja possível, a via de clock também é alterada como entrada e saída (impõe somente o nível baixo, forçando o GND) durante a transmissão, ficando em determinados momentos em um estado flutuante. Por isso, é necessário que essa linha possua um resistor de pull-up para evitar esse estado flutuante e não gerar problemas no protocolo. Esse resistor deve ser de 10 kΩ a 1 kΩ, adotando-se os valores mais baixos quando a velocidade de transferência é maior. O detalhamento desse funcionamento será visto mais à frente.
- ✓ Fica claro, então, que o valor padrão tanto para **SCL** quanto para **SDA** é o nível alto (1), com esses pinos em alta impedância (entradas). Aproveitamos para comentar também que os bits são lidos sempre da borda de descida do **SCL**.
- ✓ Além de esse protocolo permitir a especificação do endereço, existem dois modos diferentes para que isso seja feito: 7 bits ou 10 bits. No primeiro caso é possível acessar até 128 endereços diferentes. Para o segundo caso, esse número eleva-se para 1.024 endereços.

Com essas explicações básicas, podemos concluir que na comunicação existem quatro situações distintas do protocolo que precisamos conhecer:

- ✓ Transmissão do Master/Recepção do Slave, com endereço de 7 bits;
- ✓ Recepção do Master/Transmissão do Slave, com endereço de 7 bits;
- ✓ Transmissão do Master/Recepção do Slave, com endereço de 10 bits;
- ✓ Recepção do Master/Transmissão do Slave, com endereço de 10 bits.

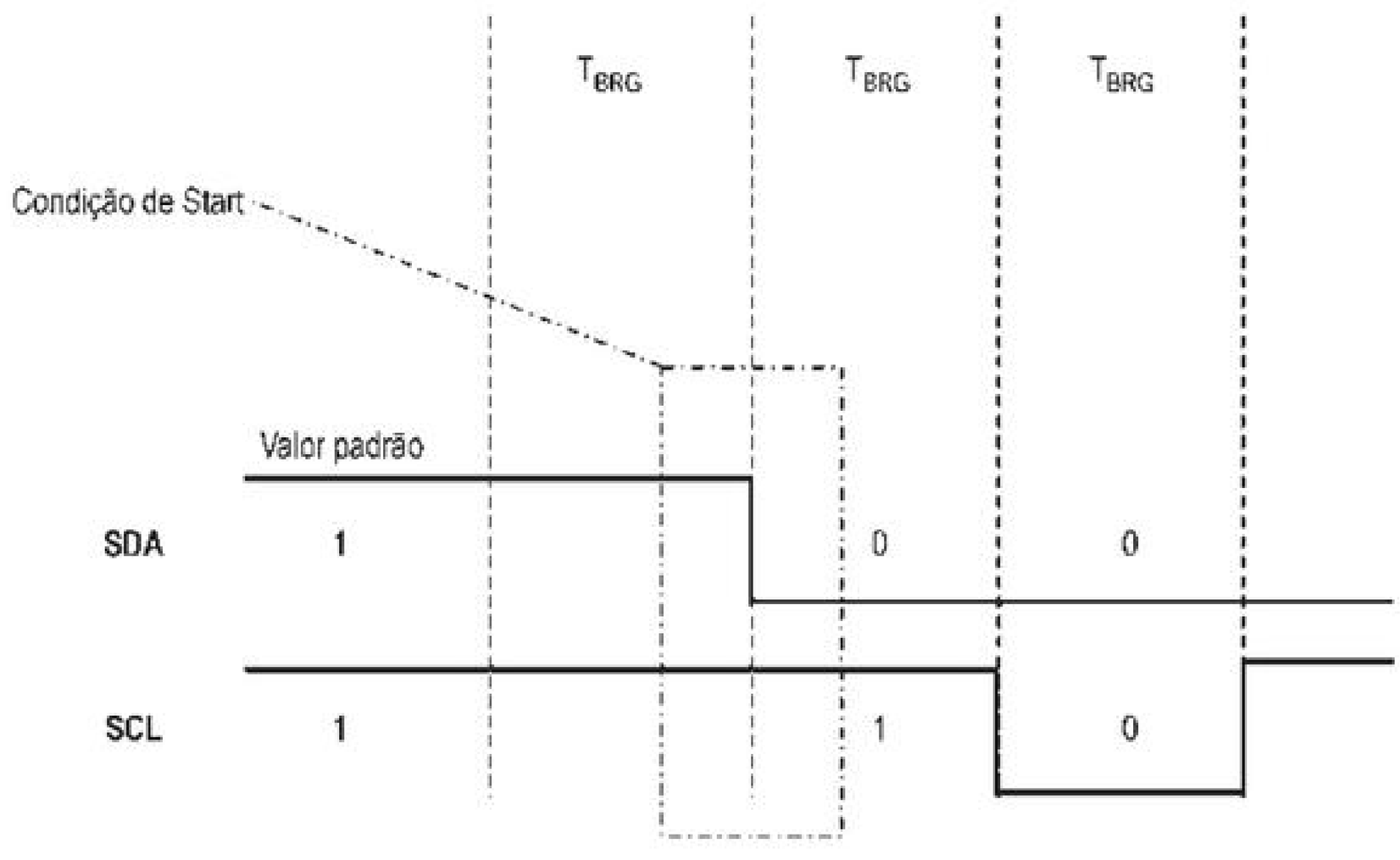
A seguir são mostradas quatro cartas de tempo completas, para cada uma das situações apresentadas. Antes, porém, vamos conhecer outros pontos importantes desse protocolo que aparecem nos diagramas mencionados.

Editora Érica - Desbravando o Microcontrolador PIC18 Recursos Avançados - Daniel R. de Sousa, David José de Souza e Nicolás César Lavinia - 1ª Edição

Condição de Start

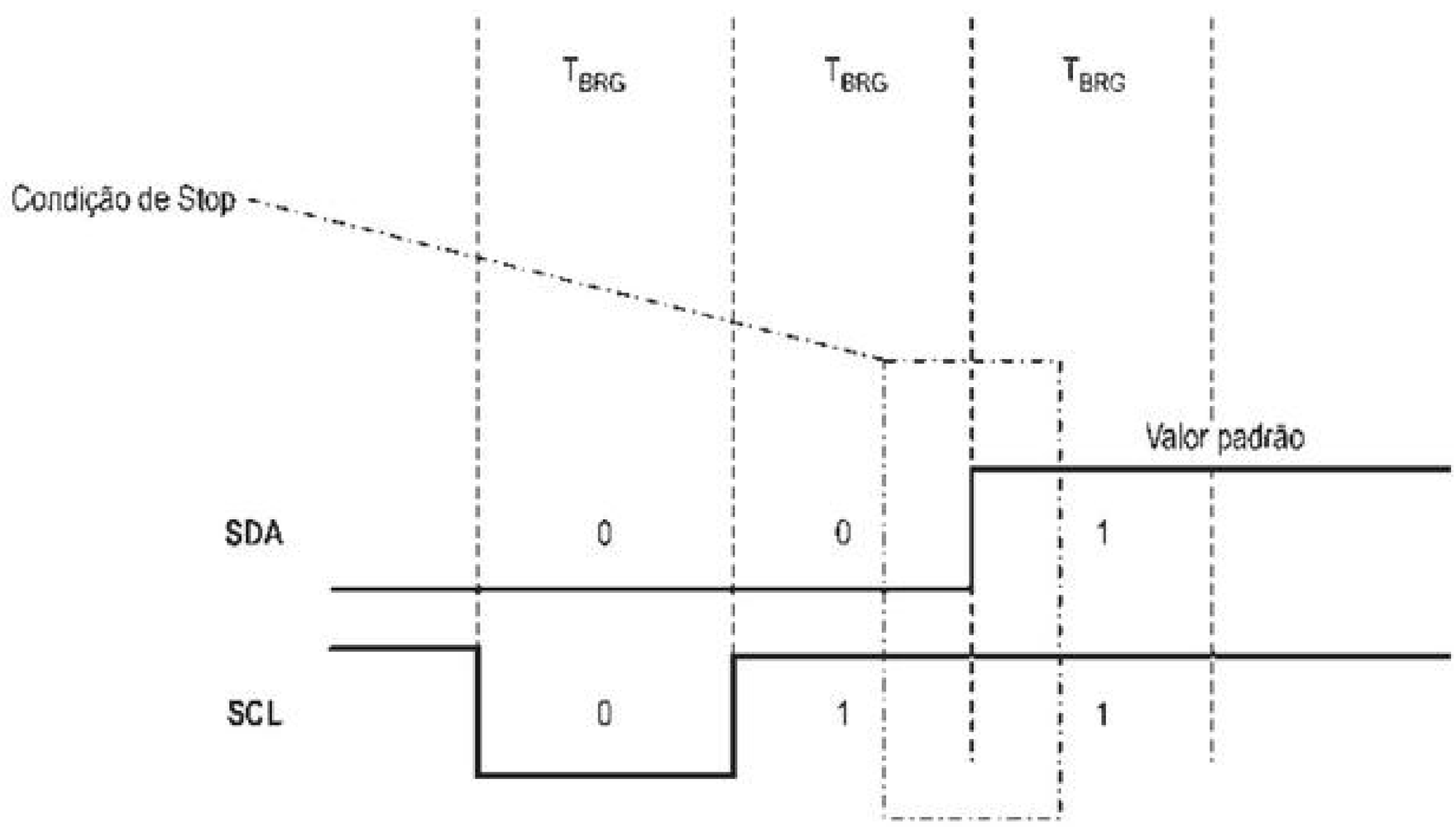
Trata-se de uma condição especial entre as vias **SDA** e **SCL** para informar à rede que uma transmissão vai começar. Essa condição é controlada pelo Mestre e todos os Escravos podem ficar prontos, esperando um dado ser transmitido. Em rede de múltiplos mestres, essa condição serve também para informar aos demais Mestres que a linha está ocupada.

Essa condição é atingida respeitando-se o seguinte diagrama: em que T_{BRG} é o tempo equivalente a um pulso de clock.



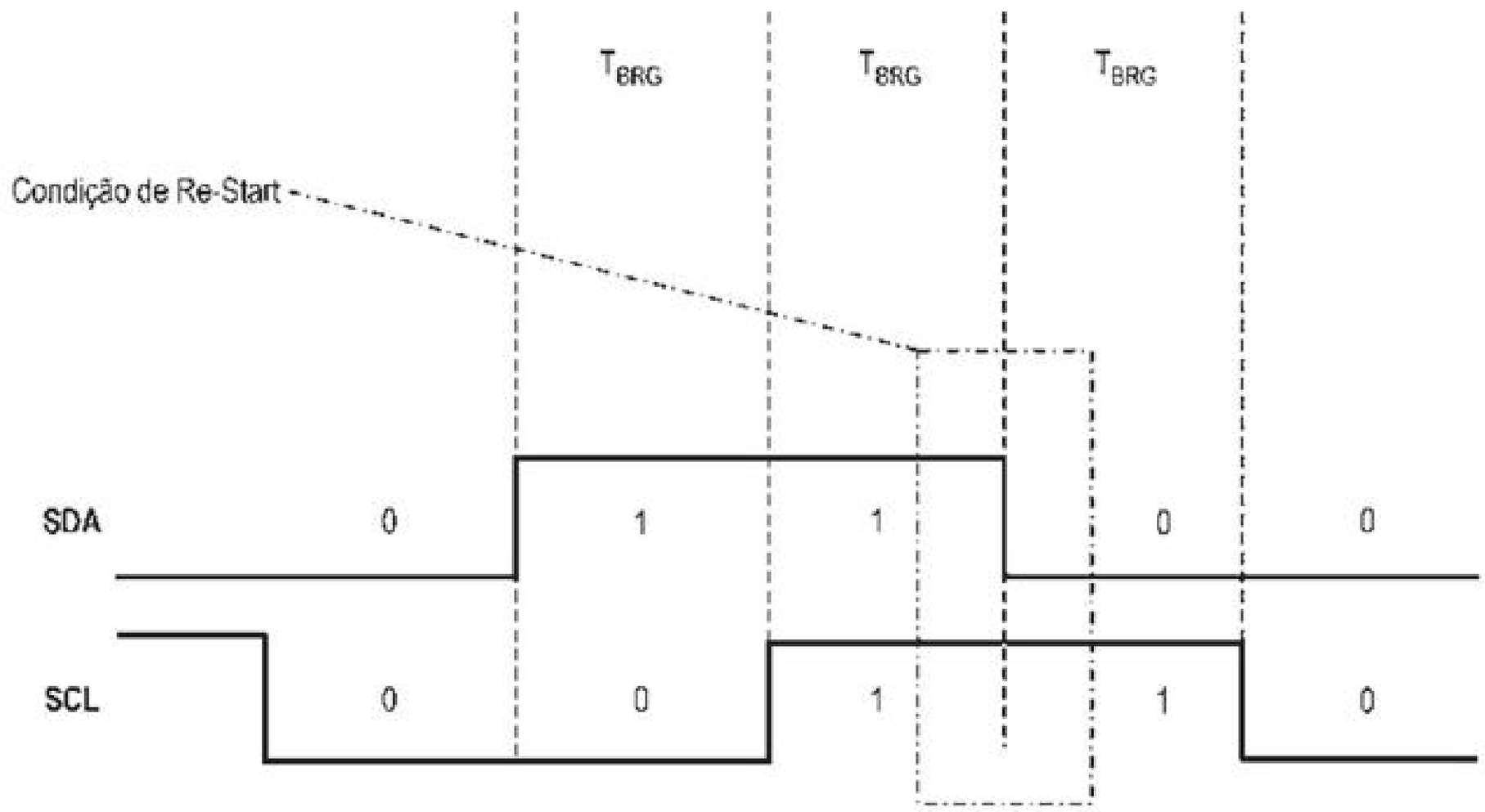
Condição de Stop

Outro estado especial dos pinos **SDA** e **SCL**, com efeito contrário à condição de Start, pois informa a toda a rede que a transmissão terminou, ficando ela disponível. Uma condição de Start obriga uma posterior condição de Stop, para que o sistema não fique travado.



Condição de Re-Start

É uma condição muito semelhante à de Start, mas que pode ser executada antes do Stop. Isso é necessário, por exemplo, para a implementação da comunicação com endereço de 10 bits ou nos casos em que precisamos enviar novamente o primeiro byte antes de finalizarmos a comunicação através de um Stop.



Condição de Acknowledge (ACK)

Na verdade, isso é uma resposta dada pelo Mestre ou pelo Escravo, no lugar do nono bit, informando se o dado foi corretamente recebido ou não. Um acknowledge alto (1) representa um erro, enquanto um acknowledge baixo (0) representa OK. No final de uma transmissão de dados do Escravo para o Mestre, isto é, após o último byte, o Mestre deve sempre responder ACK=1.

Transmissão de endereço

Independentemente da informação a ser transmitida (endereço de 7 bits, endereço de 10 bits ou dados de 8 bits), a comunicação sempre é feita em pacotes de 8 bits e mais um retorno de acknowledge. Desta forma, para cada informação transmitida são necessários nove pulsos no clock. O último bit do primeiro byte não faz parte da informação, pois é utilizado como um flag de marcação para escrita (W-Write) ou leitura (R-Read). Quando o Mestre deseja enviar dados ao Escravo, esse bit recebe o valor 0 (zero). Quando ele deseja receber um valor do Escravo, ele começa perguntando com esse bit em 1 (um). É por isso que o endereço só comporta 7 bits: são exatamente os sete que sobraram nesse mesmo byte.

Para o endereçamento de 10 bits, é necessária a transmissão inicial de 2 bytes só para o endereço. No primeiro byte, o bit menos significativo continua válido como R/W. Dos outros 7 bits, entretanto, somente dois são considerados como a parte alta do endereço. Trata-se dos bits 2 e 1. Os bits de 7 a 3 devem ser escritos com o valor 11110. Após a transmissão do primeiro byte, **SDA** é transformado em entrada para verificar o acknowledge ao ser dado o nono pulso do clock. Se a resposta for afirmativa, temos ACK=0.

Para a geração de um pulso no clock, o nível baixo é conseguido impondo-se o aterramento do pino **SCL** (como se ele fosse uma saída). Em seguida, para o nível alto, desliga-se o aterramento e o nível é conseguido pelo pull-up da linha. Durante todo o tempo do pulso, o pino **SCL** deve ser monitorado pelo

Editora Érica - Desbravando o Microcontrolador PIC18 Recursos Avançados - Daniel R. de Sousa, David José de Souza e Nicolás César Lavinia - 1ª Edição

Mestre e, caso ele desça, é porque algum Escravo está pedindo uma pausa no processo. O tempo do clock é, então, paralisado e o Mestre só volta a responder quando a linha de **SCL** estiver liberada.

Quando o clock estiver liberado, o Mestre inicia a transmissão do segundo byte de endereço, no qual estão os 8 bits menos significativos que faltam para completar o endereço de 10 bits. Ao término, um novo ACK é aguardado e conferido.

Transmissão de dados

Depois do término da transferência da parte relacionada ao endereço, um dos dois lados (Mestre ou Escravo) deve transmitir um ou mais bytes de dados. Caso seja o Escravo que esteja transmitindo para o Mestre, para finalizar a comunicação o Mestre deve responder um ACK=1 ao término do último byte recebido, para só depois enviar uma condição de Stop. Isso se faz necessário para que o Escravo seja informado de que o Mestre não mais deseja receber nenhum dado, evitando conflitos quando a condição de Stop for gerada. Nessa condição o Escravo deve ter sua comunicação paralisada e resetada, ficando no aguardo de uma nova condição de Start. Com isso o protocolo fica mais robusto, evitando erros na contagem do número de bytes enviados.

Pausas

Sempre que algum módulo Slave (ou até outro Master) necessitar de uma pausa na comunicação, por exemplo para processar a informação recebida, ela será conseguida mantendo-se a linha de clock em nível baixo, isto é, em zero.

Diagramas de tempo

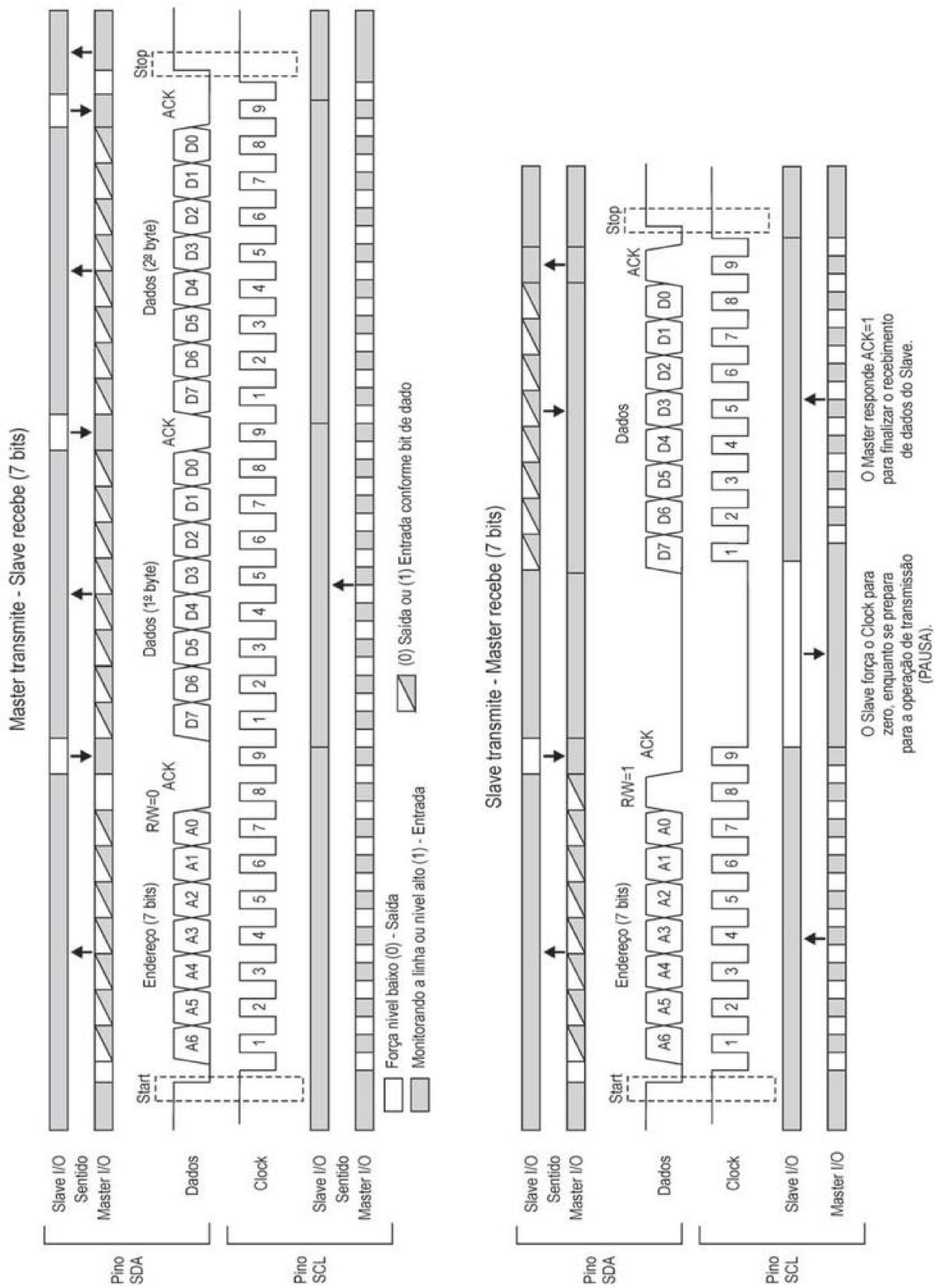
Os diagramas apresentados nas páginas seguintes demonstram graficamente o protocolo I^2C para os quatro casos possíveis já comentados:

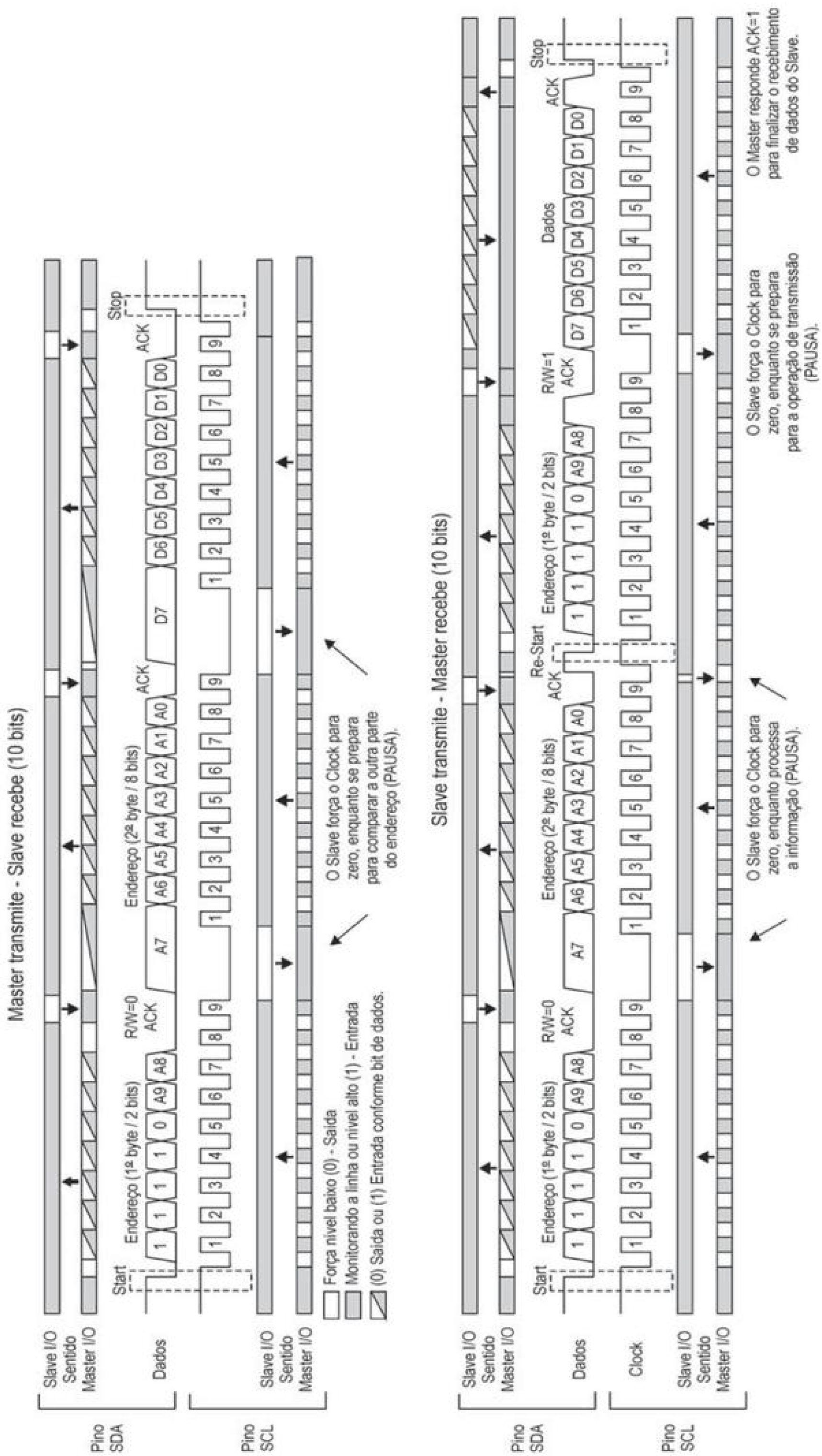
- ✓ Transmissão do Master/Recepção do Slave, com endereço de 7 bits;
- ✓ Recepção do Master/Transmissão do Slave, com endereço de 7 bits;
- ✓ Transmissão do Master/Recepção do Slave, com endereço de 10 bits;
- ✓ Recepção do Master/Transmissão do Slave, com endereço de 10 bits.

Em cada um deles são mostradas as informações que trafegam pelas vias de dados e de clock. Além disso, acima e abaixo dos desenhos das formas de onda existem também barras que representam qual dos dois lados está controlando a via. São duas barras para cada pino (**SDA** e **SCL**), uma representando o lado do Master e a outra o lado do Slave. Quando a barra está cinza, significa que o pino está configurado como entrada e quando a barra está branca, é porque o pino está impondo o nível 0 (baixo). Desta forma fica mais fácil visualizarmos o sentido de tráfego da informação:

- ✓ Master para Slave;
- ✓ Slave para Master.

Observando essas barras para o pino de clock (**SCL**), podemos ver claramente quando o Slave força pausas na comunicação, impondo o nível baixo (zero) nessa via.


Master transmite - Slave recebe (7 bits)
Slave transmite - Master recebe (7 bits)
Start
Endereço (7 bits)
R/W=0
R/W=1
ACK
Dados (1º byte)
Dados (2º byte)
Dados
Stop
Slave I/O
Sentido
Master I/O
Dados
Clock
Pino SDA
Pino SCL
(0) Saida ou (1) Entrada conforme bit de dado
Força nivel baixo (0) - Saida
Monitorando a linha ou nivel alto (1) - Entrada
O Slave força o Clock para zero, enquanto se prepara para a operação de transmissão (PAUSA).
O Master responde ACK=1 para finalizar o recebimento de dados do Slave.


Master transmite - Slave recebe (10 bits)
Slave transmite - Master recebe (10 bits)
Pino SDA
Pino SCL
Slave I/O
Sentido
Master I/O
Dados
Clock
Start
Re-Start
Stop
ACK
R/W=0
R/W=1
Endereço (1º byte / 2 bits)
Endereço (2º byte / 8 bits)
O Slave força o Clock para zero, enquanto se prepara para comparar a outra parte do endereço (PAUSA).
O Slave força o Clock para zero, enquanto processa a informação (PAUSA).
O Slave força o Clock para zero, enquanto se prepara para a operação de transmissão (PAUSA).
O Master responde ACK=1 para finalizar o recebimento de dados do Slave.
Força nivel baixo (0) - Saida
Monitorando a linha ou nivel alto (1) - Entrada
(0) Saida ou (1) Entrada conforme bit de dados.

» Recursos do PIC para I²C

Agora que as formas gerais do protocolo já estão explicadas, vamos ver como ativar e utilizar esse recurso no PIC18F4520.

Antes de mais nada, vamos explicar melhor como o PIC controla os pinos **SCL** e **SDA**. Como já foi dito na teoria do I²C, para que o protocolo funcione corretamente, evitando conflitos na linha, cada um dos módulos ligados à rede deve somente impor nível baixo (0), sendo o nível alto imposto pelos resistores de pull-up. O hardware interno do PIC respeita essas condições. Para isso, entretanto, é necessário que ambos os pinos (**SCL** e **SDA**) sejam configurados inicialmente como entrada através do **TRISC**, para que fiquem em alta impedância, possibilitando que o controle da I²C possa forçar o nível baixo quando necessário. A imposição do GND na linha é feita por intermédio de um transistor interno do tipo N. Este fato possibilita a verificação de colisão de dados na linha **SDA**, explicada posteriormente.

Caso um desses pinos, por exemplo o **SCL**, seja configurado como saída em nível alto, o sistema pode continuar funcionando, pois quando a I²C não estiver impondo zero, a própria porta vai impor 1. Isso não gera problemas internos para o PIC, mas pode gerar problemas de conflito na linha, caso seja utilizado mais de um Master ou caso algum Slave solicite uma pausa através da linha de clock.

Para facilitar o entendimento, dividimos essas explicações em duas partes: uma dedicada ao Slave e outra ao Master, descritas a seguir.

Modo Slave

Inicie sempre configurando os pinos **SCL** e **SDA** como entrada por meio do **TRISC**.

Para configurarmos um PIC para trabalhar em modo Slave, devemos ajustar os bits de **SSPCON1<SSPM3:SSPM0>**:

SSPM3: SSPM0	Descrição
0110	I²C Slave, endereço de 7 bits.
0111	I²C Slave, endereço de 10 bits.
Observação: As demais opções de configuração dizem respeito aos outros modos (SPI e I²C Master).	

Depois é necessário também ajustar o bit **SSPSTAT<SMP>** que, neste caso, serve para habilitar um sistema interno de tratamento das bordas quando operando em frequência de 400 kHz:

SMP	Descrição
0	Tratamento desabilitado.
1	Tratamento habilitado, quando usando 400 kHz.

O bit **SSPSTAT<CKE>** serve para configurar o tipo de Schmitt Trigger utilizado nos pinos SCL e SDA:

CKE	Descrição
0	Entradas conforme especificações I²C.
1	Entradas conforme especificações SMBUS.

Por último é necessário ainda habilitar o sistema de comunicação através do bit **SSPCON1<SSPEN>**:

SSPEN	Descrição
0	Desabilita a comunicação.
1	Habilita a comunicação. Nesse momento os pinos SCL e SDA deixam de operar como I/Os convencionais.

Feito isso, vejamos quais tarefas serão feitas automaticamente e quais devem ser implementadas pelo programa. Vamos tratar separadamente os possíveis casos de transferência de informações.

Recepção para endereçamento de 7 bits

1. Tanto **SCL** quanto **SDA** estão em sua configuração padrão, isto é, como entradas. A linha recebe então uma condição de Start.
2. A condição de Start é reconhecida e o bit **SSPSTAT<S>** setado; porém nenhuma interrupção acontece.
3. O sistema aguarda, então, um byte com os 7 bits de endereço e mais o bit **R/W=0**, com oito pulsos de clock. Teremos **SSPSTAT<R/W>=0**. Como o byte recebido é equivalente a um endereço, então **SSPSTAT<D/A>=0.**
4. Nesse momento, o endereço recebido é comparado automaticamente com o registrador **SSPADD**. Caso não seja o mesmo, a comunicação do lado Slave é resetada (ACK=1) e ele fica aguardando um novo Start. Caso o endereço seja idêntico, o bit **SSPSTAT<BF>** é setado, informando que um valor foi transferido para o buffer (**SSPBUF**), o pino **SDA** é transformado automaticamente em saída com nível baixo (ACK=0) e fica aguardando o nono pulso de clock. Logo depois **SDA** é liberado, tornando-se novamente entrada. Nesse momento também o flag **PIR1<SSPIF>** é setado e a interrupção pode acontecer (se as chaves estiverem corretamente ligadas). É importante que a interrupção aconteça para que seja feita uma leitura do **SSPBUF**, forçando novamente **BF=0.**
5. O sistema aguarda então um byte de dado, com oito novos pulsos de clock. Como agora o byte recebido é equivalente a um dado, então **SSPSTAT<D/A>=1**. Nesse momento, também temos **BF=1** e o dado será colocado em **SSPBUF**.
6. Mais uma vez o pino **SDA** impõe um nível baixo para responder ACK=0 no nono pulso do clock. **SDA** volta em seguida à condição de entrada.
7. Uma nova interrupção acontece **(SSPIF=1)** para que o dado recebido possa ser tratado.
8. Novos bytes de dados podem continuar sendo recebidos até que uma condição de Stop seja imposta na linha. Nesse momento, a condição é reconhecida, setando o bit **SSPSTAT<P>** e finalizando a comunicação. A interrupção não é gerada.
9. Caso seja recebida uma condição de Re-Start (entre um Start e um Stop), o Slave fica esperando um novo byte de endereço, com o **R/W**, e não mais um byte de dados.

Devido ao buffer duplo de recepção, isto é, o dado entra em **SSPSR** e depois é colocado em **SSPBUF**, a informação recebida pode então ser lida mesmo durante o recebimento do próximo byte. Depois que lemos **SSPBUF**, o flag **BF** é limpo automaticamente. Entretanto, caso o **SSPBUF** não seja lido antes da nova recepção terminar, o dado mais novo é perdido (**SSPBUF** não é atualizado) e o bit **SSPCON1<SSPOV>** setado, indicando um overlflow. **SSPOV** deve ser limpo manualmente pelo programa. Dependendo das condições de **BF** e **SSPOV**, a resposta automática do acknowledge pode ser negativa (ACK=1).

BF	SSPOV	Atualiza SSPBUF	ACK	Interrupção SSPIF=1
0	0	Sim	0 (Ok)	Sim
1	0	Não	1 (Erro)	Sim
1	1	Não	1 (Erro)	Sim
0 (Nota)	1 (Nota)	Sim	1 (Erro)	Sim
Nota: Esta condição representa um erro de programação, pois o SSPBUF foi limpo e o bit de SSPOV não.				

Transmissão para endereçamento de 7 bits

1. Tanto **SCL** quanto **SDA** estão em sua configuração padrão, isto é, como entradas. A linha recebe, então, uma condição de Start.
2. A condição de Start é reconhecida e o bit **SSPSTAT<S>** setado; porém nenhuma interrupção acontece.
3. O sistema aguarda então um byte com os 7 bits de endereço e mais o bit **R/W=1**, com oito pulsos de clock. Teremos **SSPSTAT<R/W>=1**. Como o byte recebido é equivalente a um endereço, então **SSPSTAT<D/A>=0**.
4. O endereço recebido é comparado automaticamente com o registrador **SSPADD**. Caso não seja o mesmo, a comunicação do lado Slave é resetada (ACK=1) e ele fica aguardando um novo Start. Caso o endereço seja idêntico, o pino **SDA** é transformado automaticamente em saída com nível baixo (ACK=0) e o sistema aguarda o nono pulso de clock. Logo depois **SDA** é transformado novamente em entrada.
5. O sistema automaticamente altera **SSPCON1<CKP>=0**, desabilitando o clock (pausa). Isso é feito impondo-se nível baixo (0) em **SCL** (saída).
6. O flag **PIR1<SSPIF>** é setado e a interrupção pode acontecer (se as chaves estiverem corretamente ligadas). É importante que a interrupção aconteça para que seja escrito o valor a ser transmitido em **SSPBUF**, o que vai fazer com que **BF=1**.
7. Depois do buffer atualizado, para que a comunicação proceda, basta impor **CKP=1**. Nesse momento, **SCL** é novamente liberado (entrada).
8. O clock é liberado e o byte existente em **SSPBUF** enviado nos próximos oito pulsos. Ao término teremos **BF=0**.
9. Mais uma vez o pino **SDA** fica aguardando um ACK no nono pulso do clock. Caso seja recebido um ACK=1 é porque a comunicação será encerrada, sendo aguardada, então, uma condição de Stop. Nesse momento a condição é reconhecida, setando-se o bit **SSPSTAT<P>** e finalizando a comunicação. A interrupção não é gerada. Caso ACK=0, o sistema volta ao item 5 para que um novo byte de dado seja transmitido.
10. Durante a transferência de dados não pode acontecer uma situação de Re-Start. Para reiniciar completamente o sistema é necessário terminar a transferência com um ACK=1 por parte do Master, seguido de um Stop. Só então uma nova condição de Start pode ser gerada.

Caso se tente escrever um valor em **SSPBUF** enquanto uma transmissão está em progresso (**BF=1**), o valor de **SSPBUF** não é atualizado e o bit **SSPCON1<WCOL>** será setado, indicando uma colisão na escrita. Esse bit deve ser limpo manualmente pelo programador.

Editora Érica - Desbravando o Microcontrolador PIC18 Recursos Avançados - Daniel R. de Sousa, David José de Souza e Nicolás César Lavinia - 1ª Edição

Recepção para endereçamento de 10 bits

1. Tanto **SCL** quanto **SDA** estão em sua configuração padrão, isto é, como entradas. A linha recebe, então, uma condição de Start.

2. A condição de Start é reconhecida e o bit **SSPSTAT<S>** setado; porém nenhuma interrupção acontece.

3. O sistema aguarda um byte com os 2 bits de endereço (A9:A8) e mais o bit **R/W=0**, com oito pulsos de clock. Teremos, então, **SSPSTAT<R/W>=0**. Como o byte recebido é equivalente a um endereço, temos também **SSPSTAT<D/A>=0**.

4. Nesse momento o endereço recebido é comparado automaticamente com o registrador **SSPADD**. Caso não seja o mesmo, a comunicação do lado Slave é resetada (ACK=1) e ele fica aguardando um novo Start. Caso essa parte do endereço esteja correta, o bit **SSPSTAT<BF>** é setado, informando que um valor foi transferido para o buffer (**SSPBUF**) e teremos **SSPSTAT<UA>=1**, indicando que o endereço deve ser atualizado. O pino **SDA** é transformado automaticamente em saída com nível baixo (ACK=0) e fica aguardando o nono pulso de clock. Logo depois, **SDA** é transformado novamente em entrada. Nesse momento, o flag **PIR1<SSPIF>** é setado e a interrupção pode acontecer (se as chaves estiverem corretamente ligadas). É importante que a interrupção aconteça para que seja feita uma leitura do **SSPBUF**, forçando novamente **BF=0**. O registrador **SSPADD** deve, então, ser atualizado com a parte baixa do endereço (A7:A0), o que força automaticamente **UA=0**. Depois do ACK e até o momento em que o **SSPADD** é atualizado, o Slave trava a comunicação (pausa), impondo um nível baixo (0) em **SCL**. Esse controle é totalmente automático.

5. É aguardado, então, um byte com a segunda parte do endereço (A7:A0) para os próximos oito pulsos do clock. Como o byte recebido ainda é equivalente a um endereço, continuamos com **SSPSTAT<D/A>=0**.

6. Todo o processo descrito para o item 4 é repetido. No final, o sistema entra em nova pausa até que **SSPADD** seja novamente atualizado com a parte alta do endereço (A9:A8).

7. O sistema aguarda um byte de dado, com mais oito pulsos de clock. Como o byte recebido é equivalente a um dado, então **SSPSTAT<D/A>=1**. Nesse momento também teremos **BF=1** e o dado é colocado em **SSPBUF**.

8. O pino **SDA** é transformado automaticamente em saída com nível baixo (ACK=0) e o sistema aguarda o nono pulso do clock. Logo depois **SDA** é transformado novamente em entrada.

9. Uma nova interrupção acontece (**SSPIF=1**) para que o dado recebido possa ser tratado.

10. Novos bytes de dados podem continuar sendo recebidos até que uma condição de Stop seja imposta na linha. Nesse momento, a condição é reconhecida, setando-se o bit **SSPSTAT<P>** e finalizando a comunicação. A interrupção não é gerada.

11. Caso seja recebida uma condição de Re-Start (entre um Start e um Stop), o Slave fica esperando um novo byte de endereço, com o bit de **R/W** e não mais um byte de dados.

Os comentários feitos a respeito do buffer e dos flags **BF** e **SSPOV** para a comunicação de 7 bits também são válidos aqui.

Transmissão para endereçamento de 10 bits

1. Tanto **SCL** quanto **SDA** estão em sua configuração padrão, isto é, como entradas. A linha recebe, então, uma condição de Start.

2. A condição de Start é reconhecida e o bit **SSPSTAT<S>** setado; porém nenhuma interrupção acontece.

3. O sistema aguarda um byte com os 2 bits de endereço (A9:A8) e mais o bit **R/W=0**, com oito pulsos de clock. Teremos, então, **SSPSTAT<R/W>=0**. Como o byte recebido é equivalente a um endereço, temos também **SSPSTAT<D/A>=0**.

4. Nesse momento, o endereço recebido é comparado automaticamente com o registrador **SSPADD**. Caso não seja o mesmo, a comunicação do lado Slave é resetada (ACK=1) e fica aguardando um novo Start. Caso essa parte do endereço esteja correta, o bit **SSPSTAT<BF>** é setado, informando que um valor foi transferido para o buffer (**SSPBUF**) e teremos **SSPSTAT<UA>=1**, indicando que o endereço deve ser atualizado. O pino **SDA** é transformado automaticamente em saída com nível baixo (ACK=0) enquanto aguarda o nono pulso do clock. Depois **SDA** é transformado novamente em entrada. Nesse momento, o flag **PIR1<SSPIF>** é setado e a interrupção pode acontecer (se as chaves estiverem corretamente ligadas). É importante que a interrupção aconteça para que seja feita uma leitura do **SSPBUF**, forçando novamente **BF=0**. O registrador **SSPADD** deve, então, ser atualizado com a parte baixa do endereço (A7:A0), o que força automaticamente **UA=0**. Depois do ACK e até o momento em que o **SSPADD** é atualizado, o Slave trava a comunicação (pausa), impondo um nível baixo (0) em **SCL**. Esse controle é totalmente automático.

5. É aguardado um byte com a segunda parte do endereço (A7:A0) nos próximos oito pulsos do clock. Como o byte recebido ainda é equivalente a um endereço, continuamos com **SSPSTAT<D/A>=0**.

6. Todo o processo descrito para o item 4 é repetido. No final o sistema entra em pausa até que **SSPADD** seja novamente atualizado com a parte alta do endereço (A9:A8).

7. A linha fornece, então, uma condição de Re-Start, informando que o próximo byte é novamente uma informação de controle.

8. É aguardado outro byte com a parte alta do endereço e com **R/W=1**, para os próximos oito pulsos do clock. Como o byte recebido ainda é equivalente a um endereço, continuamos com **SSPSTAT<D/A>=0**. O valor recebido é colocado em **SSPBUF** e **BF=1**.

9. O pino **SDA** é transformado automaticamente em saída com nível baixo (ACK=0), enquanto o sistema aguarda o nono pulso do clock. Logo depois, **SDA** é transformado novamente em entrada.

10. O sistema automaticamente altera **SSPCON1<CKP>=0**, desabilitando o clock (pausa). Isso é feito transformando **SCL** em saída com nível baixo (0).

11. O flag **PIR1<SSPIF>** é setado e a interrupção pode acontecer (se as chaves estiverem corretamente ligadas). É importante que a interrupção aconteça para que **SSPBUF** seja lido, forçando **BF=0** novamente. Depois devemos escrever o valor a ser transmitido em **SSPBUF**, o que faz com que **BF=1**.

12. Depois do buffer atualizado, para que a comunicação proceda, basta impor **CKP=1**. Nesse momento, **SCL** é novamente liberado.

13. O clock é liberado e o byte existente em **SSPBUF** enviado nos próximos oito pulsos. Ao término temos **BF=0**.

14. Mais uma vez o pino **SDA** fica aguardando um ACK no nono pulso do clock. Caso seja recebido um ACK=1, é porque a comunicação será encerrada, sendo aguardada então uma condição de Stop. Nesse momento, a condição é reconhecida, setando-se o bit **SSPSTAT<P>** e finalizando-se a comunicação. A interrupção não é gerada. Caso ACK=0, o sistema volta ao item 5 para que um novo byte de dado seja transmitido.

Editora Érica - Desbravando o Microcontrolador PIC18 Recursos Avançados - Daniel R. de Sousa, David José de Souza e Nicolás César Lavinia - 1ª Edição

15. Durante a transferência de dados não pode acontecer uma nova situação de Re-Start. Para reiniciar completamente o sistema é necessário terminar a transferência com um ACK=1 por parte do Master, seguido de um Stop. Só então uma nova condição de Start pode ser gerada.

A condição de colisão (**WCOL**) descrita na comunicação de 7 bits também é valida para a de 10 bits.

Tratamento da interrupção SSP

Como só existe uma interrupção de porta serial (SSP) e diversas condições, os bits adicionais dessa porta devem ser verificados para sabermos exatamente que atitude deve ser tomada. Veja algumas condições:

BF	UA	D/A	SSPOV	Situação
1	X	X	X	Sempre que a interrupção acontecer **BF**=1, é porque alguma informação foi colocada em **SSPBUF**. Para o caso da transmissão, esse flag pode ser verificado para sabermos se o processo já terminou, evitando uma colisão (**WCOL**).
1	X	0	X	Foi recebido um endereço. Como o endereço é verificado automaticamente, a interrupção deve somente ler **SSPBUF** para limpar **BF**.
1	1	0	X	Somente quando configurada comunicação em 10 bits. A interrupção deve atualizar o endereço, primeiro com o valor baixo e da segunda vez com o valor alto.
1	0	0	X	Essa situação só acontece para 7 bits ou quando recebido o segundo byte de controle para a transmissão quando operando em 10 bits, em que também teremos R/W=1.
1	X	X	1	Significa que a última informação recebida foi perdida, pois o **SSPBUF** não havia sido limpo.

Endereçamento global

O Slave possui ainda um último recurso, que podemos chamar de endereço global. Isso significa que o endereço 0 (zero) pode ser aceito automaticamente por todos os PICs. Desta forma, cada unidade da rede possui um endereço específico (**SSPADD**) e um endereço global (Zero).

Para habilitar o uso de endereço global em um módulo Slave, basta ativar o bit **SSPCON2<GCEN>**:

GCEN	Descrição
0	Endereço global (Zero) desabilitado.
1	Endereço global (Zero) habilitado.

Quando o endereço global é inserido na rede, todos os módulos Slave recebem a informação e todos aqueles com **GCEN** devem responder com um ACK. Isso lembra um comentário muito importante. Quando ACK=0, o Slave coloca **SDA** como saída em nível baixo. Porém, quando ACK=1, **SDA** continua como entrada e o nível alto é imposto pelo resistor de pull-up. Isso possibilita que diversos módulos respondam ACKs diferentes, tendo sempre a preferência o erro, ou seja, ACK=0.

Com a habilitação do endereço global a interrupção é gerada e o tratamento ou não das informações será uma escolha do programador.

Resumo dos registradores associados a I²C Slave

Endereço	Nome	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
FF2h	INTCON	INTCON	GIE/ GIEH	PEIE/ GIEL	T0IE	INTE	RBIE	TMR0IF	INTF
F9Eh	PIR1	PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
F9Dh	PIE1	PSPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
F9Fh	IPR1	PSPIP	ADIP	RCIP	RXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP
FC7h	SSPSTAT	SMP	CKE	D/A	P	S	R/W	UA	BF
FC6h	SSPCON1	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
FC5h	SSPCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN
FC9h	SSPBUF	Buffer de transmissão / recepção							
FC8h	SSPADD	Endereço (parte alta ou parte baixa)							
F94h	TRISC	Configuração do PORTC com Entrada(1) ou Saída(0)							

Não é usado para essa finalidade. Para obter mais informações, consulte o Apêndice A.

Modo Master

O Master não diferencia automaticamente o tipo de comunicação em 7 ou 10 bits. Ele somente transmite e recebe informações de 8 bits. Por isso, toda a lógica da comunicação deve ser feita por intermédio do programa. Internamente, o PIC possui controle de hardware para os seguintes eventos:

- ✓ Gerar/monitorar condição de Start;
- ✓ Gerar/monitorar condição de Stop;
- ✓ Gerar/monitorar condição de Re-Start;
- ✓ Gerar/monitorar condição de Pausa;
- ✓ Gerar/monitorar Acknowledge;
- ✓ Gerar/monitorar o clock;
- ✓ Transmitir informação de 8 bits;
- ✓ Receber informação de 8 bits.

Inicie sempre configurando os pinos **SCL** e **SDA** como entrada através do **TRISC**.

Para configurar um PIC para trabalhar em modo Master, ajustam-se os bits de **SSPCON1<SSPM3:SSPM0>**:

SSPM3: SSPM0	Descrição
1000	I²C Master, controle por hardware com clock= F_{osc} / (4 x (SSPADD+1))
1001	Reservado
1010	Reservado
1011	Reservado
1100	Reservado
1101	Reservado
1110	Reservado
1111	Reservado
Observação: As demais opções de configuração dizem respeito aos outros modos (SPI e I²C Slave).	

Depois, é necessário também ajustar o bit **SSPSTAT<SMP>** que, neste caso, serve para habilitar um sistema interno de tratamento das bordas, quando operando em frequência de 400 kHz:

SMP	Descrição
0	Tratamento desabilitado.
1	Tratamento habilitado, quando usando 400 kHz.

O bit **SSPSTAT<CKE>** serve para configurar o tipo de Schmitt Trigger utilizado nos pinos **SCL** e **SDA**:

CKE	Descrição
0	Entradas conforme especificações I^2C.
1	Entradas conforme especificações SMBUS.

A frequência do clock deve ser configurada pelo registrador **SSPADD**, respeitando a seguinte fórmula:

$$SSPADD = \frac{\left(\frac{F_{osc}}{Freq.}\right)}{4} - 1$$

As frequências padronizadas para o protocolo I^2C são as de 100 kHz e 400 kHz. Com o PIC, entretanto, é possível configurar outras frequências. O fato é que todos os periféricos da rede devem estar aptos a responder na frequência escolhida. Por isso, o recomendável é verificar a frequência mais alta suportada pelos periféricos e escolher o valor mais próximo possível dos padronizados.

Por último, é necessário ainda habilitar o sistema de comunicação através do bit **SSPCON1<SSPEN>**:

SSPEN	Descrição
0	Desabilita a comunicação.
1	Habilita a comunicação. Nesse momento os pinos SCL e SDA deixam de operar como I/Os convencionais.

Feito isso, vejamos que tarefas serão realizadas automaticamente e quais devem ser implementadas pelo programa. Vamos tratar separadamente os possíveis casos de transferência de informações.

Transmissão para endereçamento de 7 bits

1. Tanto **SCL** quanto **SDA** estão em sua configuração padrão, isto é, como entradas.
2. Deve ser imposta uma condição de Start na linha. Para isso, basta ativar o bit **SSPCON2<SEN>**. A condição de Start será, então, gerada automaticamente.
3. Ao término do Start, o flag da interrupção é ativado (**SSPIF=1**) e **SEN=0**. Deve-se, então, escrever em **SSPBUF** o endereço do periférico com o qual desejamos nos comunicar. O endereço deve ser escrito nos bits de 7 a 1. O bit 0 deve ser considerado como o valor **R/W=0**.
4. Automaticamente, logo após a escrita no registrador **SSPBUF**, temos **SSPSTAT<BF>=1**, **SSPSTAT<R/W>=1** e o byte é transmitido através de oito pulsos gerados em **SCL**. No término da transmissão teremos novamente **BF=0**.
5. O pino **SDA** aguarda um ACK (deve ser 0) no nono pulso do clock. O valor recebido é armazenado no bit **SSPCON2<ACKSTAT>**. O valor de ACK não é verificado automaticamente.
6. Nesse momento, **R/W=0** e **SSPIF=1**, podendo gerar uma nova interrupção.
7. O dado a ser transmitido é, então, colocado em **SSPBUF**. Teremos **BF=1** e **R/W=1**.

8. Mais uma vez teremos a transmissão do byte através de oito pulsos em **SCL**, a partir do momento em que nenhum periférico esteja solicitando uma pausa. No término **BF=0**.

9. Novamente o pino **SDA** aguarda um ACK (deve ser 0) no nono pulso do clock. O valor recebido é armazenado no bit **SSPCON2<ACKSTAT>**. O valor de ACK não é verificado automaticamente.

10. Nesse momento **R/W=0** e **SSPIF=1**, podendo gerar outra interrupção.

11. Se um novo byte deve ser transmitido, o processo é reiniciado no item 7. Para interromper a transmissão, uma condição de Stop deve ser gerada. Isso é conseguido pelo bit **SSPCON2<PEN>=1**.

12. No final da condição de Stop teremos automaticamente **PEN=0** e **SSPIF=1**.

Recepção para endereçamento de 7 bits

1. Tanto **SCL** quanto **SDA** estão em sua configuração padrão, isto é, como entradas.

2. Deve ser imposta uma condição de Start na linha. Para isso, basta ativar o bit **SSPCON2<SEN>**. A condição de Start será, então, gerada automaticamente.

3. Ao término do Start, o flag da interrupção é ativado (**SSPIF=1**) e **SEN=0**. Deve-se escrever em **SSPBUF** o endereço do periférico com o qual desejamos nos comunicar. O endereço deve ser escrito nos bits de 7 a 1. O bit 0 deve ser considerado como o valor **R/W=1**.

4. Automaticamente, logo após a escrita no registrador **SSPBUF**, teremos **SSPSTAT<BF>=1**, **SSPSTAT<R/W>=1** e o byte é transmitido através de oito pulsos gerados em **SCL**. No término da transmissão teremos novamente **BF=0**.

5. O pino **SDA** aguarda um ACK (deve ser 0) no nono pulso do clock. O valor recebido é armazenado no bit **SSPCON2<ACKSTAT>**. O valor de ACK não é verificado automaticamente.

6. Nesse momento, **R/W=0** e **SSPIF=1**, podendo gerar uma nova interrupção.

7. O Master deve, então, ser colocado em modo de recepção, ajustando-se **SSPCON2<RCEN>=1**.

8. Assim que a linha de clock não estiver em pausa, é aguardado um byte de dados para os próximos oito pulsos do clock. Como o byte recebido é equivalente a um dado, teremos **SSPSTAT<D/A>=1**. O valor recebido é colocado em **SSPBUF** e **BF=1**.

9. Nesse momento, **RCEN=0** (automaticamente), desligando-se o modo de recepção. Também teremos **SSPIF=1**, gerando uma nova interrupção.

10. Se um novo byte deve ser recebido, o Master deve responder ACK=0. Caso seja o último byte desejado, então ACK=1. Para responder o ACK, o valor desejado deve ser colocado no bit **SSPCON2<ACKDT>**. Liga-se a geração do sinal de ACK através de **SSPCON2<ACKEN>=1.**

11. Ao término do ACK temos **ACKEN=0** e **SSPIF=1**, gerando outra interrupção. Caso tenha sido respondido ACK=0, o processo deve retornar ao item 5 para que um novo byte seja recebido. Se ACK=1, o processo deve ser finalizado através de uma condição de Stop que será gerada com **SSPCON2<PEN>**.

12. No final da condição de Stop teremos automaticamente **PEN=0** e **SSPIF=1**.

Existem duas situações que podem gerar uma transmissão:

- ✓ **O Master efetua uma pergunta e deseja uma resposta**: neste caso, primeiramente é feita uma recepção pelo Slave (pergunta) e depois uma transmissão (resposta). Entre a pergunta e a resposta pode ser gerada uma condição de Re-Start, não sendo necessário liberar a linha (Stop). Um exemplo disso são as memórias E^2PROM externas. Primeiro devemos informar qual endereço desejamos acessar para que depois ela responda o valor contido no endereço em questão.
- ✓ **O Master só deseja uma resposta**: em casos de Slaves mais "burros" é provável que somente com seu endereço já seja possível responder corretamente. Por exemplo, para um módulo de teclado, basta que ele transmita o número da tecla pressionada, não precisando processar perguntas diferentes.

Transmissão para endereçamento de 10 bits

1. Tanto **SCL** quanto **SDA** estão em sua configuração padrão, isto é, como entradas.
2. Deve ser imposta uma condição de Start na linha. Para isso, basta ativar o bit **SSPCON2<SEN>**. A condição de Start é, então, gerada automaticamente.
3. Ao término do Start, o flag da interrupção é ativado (**SSPIF=1**) e **SEN=0**. Deve-se, então, escrever em **SSPBUF** a parte alta do endereço do periférico com o qual desejamos nos comunicar. Os 2 bits do endereço (A9:A8) devem ser escritos nos bits de 2 e 1. O bit 0 deve ser considerado como o valor **R/W=0**. Os demais bits (7:3) devem conter os valores 11110.
4. Automaticamente, logo após a escrita no registrador **SSPBUF**, teremos **SSPSTAT<BF>=1**, **SSPSTAT<R/W>=1** e o byte é transmitido através de oito pulsos gerados em **SCL**. No término da transmissão teremos novamente **BF=0**.
5. O pino **SDA** aguarda um ACK (deve ser 0) no nono pulso do clock. O valor recebido é armazenado no bit **SSPCON2<ACKSTAT>**. O valor de ACK não é verificado automaticamente.
6. Nesse momento, **R/W=0** e **SSPIF=1**, podendo gerar uma nova interrupção.
7. A segunda parte do endereço (A7:A0) é, então, escrita em **SSPBUF**.
8. Automaticamente, logo após a escrita no registrador **SSPBUF**, teremos **SSPSTAT<BF>=1**, **SSPSTAT<R/W>=1** e o byte é transmitido através de oito pulsos gerados em **SCL**. No término da transmissão teremos novamente **BF=0**.
9. O pino **SDA** aguarda um ACK (deve ser 0) no nono pulso do clock. O valor recebido é armazenado no bit **SSPCON2<ACKSTAT>**. O valor de ACK não é verificado automaticamente.
10. Nesse momento, **R/W=0** e **SSPIF=1**, podendo gerar uma nova interrupção.
11. O dado a ser transmitido é colocado em **SSPBUF**. Teremos **BF=1** e **R/W=1**.
12. Mais uma vez temos a transmissão do byte através de oito pulsos em **SCL**, a partir do momento em que nenhum periférico esteja solicitando uma pausa. No término **BF=0**.
13. Novamente o pino **SDA** aguarda um ACK (deve ser 0) no nono pulso do clock. O valor recebido é armazenado no bit **SSPCON2<ACKSTAT>**. O valor de ACK não é verificado automaticamente.
14. Nesse momento, **R/W=0** e **SSPIF=1**, podendo gerar outra interrupção.

15. Se um novo byte deve ser transmitido, o processo é reiniciado no item 11. Para interromper a transmissão, uma condição de Stop deve ser gerada. Isso é conseguido por intermédio do bit **SSPCON2<PEN>=1**.

16. No final da condição de Stop temos automaticamente **PEN=0** e **SSPIF=1**.

Recepção para endereçamento de 10 bits

1. Tanto **SCL** quanto **SDA** estão em sua configuração padrão, isto é, como entradas.

2. Deve ser imposta uma condição de Start na linha. Para isso, basta ativar o bit **SSPCON2<SEN>**. A condição de Start será, então, gerada automaticamente.

3. Ao término do Start, o flag da interrupção é ativado (**SSPIF=1**) e **SEN=0**. Deve-se, então, escrever em **SSPBUF** a parte alta do endereço do periférico com o qual desejamos nos comunicar. Os 2 bits do endereço (A9:A8) devem ser escritos nos bits de 2 e 1. O bit 0 deve ser considerado como o valor **R/W=0**. Os demais bits (7:3) devem conter os valores 11110.

4. Automaticamente, logo após a escrita no registrador **SSPBUF**, teremos **SSPSTAT<BF>=1**, **SSPSTAT<R/W>=1** e o byte é transmitido através de oito pulsos gerados em **SCL**. No término da transmissão temos novamente **BF=0**.

5. O pino **SDA** aguarda um ACK (deve ser 0) no nono pulso do clock. O valor recebido é armazenado no bit **SSPCON2<ACKSTAT>**. O valor de ACK não é verificado automaticamente.

6. Nesse momento, **R/W=0** e **SSPIF=1**, podendo gerar uma nova interrupção.

7. A segunda parte do endereço (A7:A0) é escrita em **SSPBUF**.

8. Automaticamente, logo após a escrita no registrador **SSPBUF**, teremos **SSPSTAT<BF>=1**, **SSPSTAT<R/W>=1** e o byte é transmitido através de oito pulsos gerados em **SCL**. No término da transmissão temos novamente **BF=0**.

9. O pino **SDA** aguarda um ACK (deve ser 0) no nono pulso do clock. O valor recebido é armazenado no bit **SSPCON2<ACKSTAT>**. O valor de ACK não é verificado automaticamente.

10. Nesse momento, **R/W=0** e **SSPIF=1**, podendo gerar uma nova interrupção.

11. Deve ser gerada uma condição de Re-Start. Para isso, basta ativar o bit **SSPCON2<RSEN>=1**.

12. Ao término do Re-Start, o flag da interrupção é ativado (**SSPIF=1**) e **RSEN=0**.

13. Mais uma vez devemos colocar em **SSPBUF** o valor 11110, seguido dos bits A9:A8 e o último bit como **R/W=1**. Teremos **BF=1** e **R/W=1**.

14. Temos novamente a transmissão do byte através de oito pulsos em **SCL**, a partir do momento em que nenhum periférico esteja solicitando uma pausa. No término **BF=0**.

15. Novamente o pino **SDA** aguarda um ACK (deve ser 0) no nono pulso do clock. O valor recebido é armazenado no bit **SSPCON2<ACKSTAT>**. O valor de ACK não é verificado automaticamente.

16. Nesse momento, **R/W=0** e **SSPIF=1**, podendo gerar outra interrupção.

17. O Master deve, então, ser colocado em modo de recepção, ajustando-se **SSPCON2<RCEN>=1**.

18. Assim que a linha de clock não estiver em pausa, será aguardado um byte de dados para os próximos oito pulsos do clock. Como o byte recebido é equivalente a um dado, temos **SSPSTAT<D/A>=1**. O valor recebido é colocado em **SSPBUF** e **BF=1**.

19. Nesse momento, **RCEN=0** (automaticamente), desligando-se o modo de recepção. Também teremos **SSPIF=1**, gerando uma nova interrupção.

20. Se um novo byte deve ser recebido, o Master deve responder ACK=0. Caso seja o último byte desejado, então ACK=1. Para responder o ACK, o valor desejado deve ser colocado no bit **SSPCON2<ACKDT>**. Liga-se a geração do sinal de ACK através de **SSPCON2<ACKEN>=1**.

21. Ao término do ACK temos **ACKEN=0** e **SSPIF=1**, gerando outra interrupção. Caso tenha sido respondido ACK=0, o processo deve retornar ao item 13 para que um novo byte seja recebido. Se ACK=1, o processo deve ser finalizado por intermédio de uma condição de Stop que é gerada com **SSPCON2<PEN>.**

22. No final da condição de Stop temos automaticamente **PEN=0** e **SSPIF=1**.

Outras considerações

Existem outros bits e flags envolvidos com o Master que não foram comentados até o momento.

Como um sistema pode possuir mais de um Master, as condições de Start e Stop geradas na linha acarretam a ativação de **SSPIF**, podendo gerar uma interrupção em qualquer Master. Os flags **SSPSTAT<S>** e **SSPSTAT<P>** informam, respectivamente, as condições de Start e Stop. Desta forma é possível que outros Masters percebam quando a linha está ou não ocupada.

Aqui valem também os comentários para os flags **SSPCON<WCOL>** e **SSPCON1<SSPOV>** que indicam, respectivamente, uma escrita inválida (na hora errada) em **SSPBUF** e o recebimento de um novo byte sem a leitura do byte anterior.

As condições de Start, Stop, Re-Start e ACK não podem ser geradas fora dos momentos corretos, quando o modo de comunicação não está em stand-by. A situação de stand-by acontece sempre que o hardware da I^2C acaba o controle de um evento qualquer.

Detecção de colisão na linha (Bus colision)

Outro recurso que o Master possui está relacionado à colisão de informações na linha de dados (**SDA**). Vejamos como isso funciona.

Para que o PIC possa impor o nível baixo da linha (normalmente consideramos uma condição de saída) existe um transistor interno conectando esse pino ao GND. Quando essa conexão está feita, a linha realmente está em nível baixo e não há o que monitorar. No outro caso, entretanto, quando esse transistor está em aberto, o pino é considerado uma entrada e o nível alto é imposto pelo resistor de pull-up (externo). Nessa situação, o PIC monitora a linha. Caso seja necessário transmitir um bit em 1 (nível alto/transistor aberto) e a linha esteja em zero, é porque algum outro ponto da rede está em conflito de comunicação.

Quando isso acontecer, o bit **PIR2<BCLIF>** é setado, e se **PIE2<BCLIE>=1**, a interrupção é gerada. Temos ainda o reset do sistema de comunicação, voltando ao estado de stand-by.

As colisões podem acontecer durante a transmissão de dados e as condições de Start, Stop, Re-Start e ACK.

Resumo dos registradores associados a I²C Master

Endereço	Nome	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
FF2h	INTCON	INTCON	GIE/ GIEH	PEIE/ GIEL	T0IE	INTE	RBIE	TMR0IF	INTF
F9Eh	PIR1	PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
F9Dh	PIE1	PSPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
F9Fh	IPR1	PSPIP	ADIP	RCIP	RXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP
FC7h	SSPSTAT	SMP	CKE	D/A	P	S	R/W	UA	BF
FC6h	SSPCON1	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
FC5h	SSPCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN
FC9h	SSPBUF	Buffer de transmissão/recepção							
FC8h	SSPADD	Ajuste da frequência							
F94h	TRISC	Configuração do PORTC com Entrada(1) ou Saída(0)							

Não é usado para essa finalidade. Para obter mais informações, consulte o Apêndice A.

» Lógica do exemplo

O exemplo para este capítulo usa somente uma parte da teoria apresentada. Vamos usar o protocolo I²C, operando como Master, para comunicação com uma memória E²PROM externa. A memória em questão é a 24C04, a mesma indicada no esquema do Apêndice C.

Implementamos, então, um sistema muito simples que mostra um valor no LCD, podendo ser alterado através de dois botões (S1 e S2). Esse valor é apresentado em hexadecimal e pode variar de 00 a FF. Com o botão S3 salvamos o valor atual em uma posição fixa da memória. O valor pode, então, ser novamente alterado pelos botões. Para recuperar o valor salvo anteriormente, basta pressionar o botão S4.

Essa memória trabalha com o protocolo I²C de 7 bits. No entanto, ela distorce um pouco a interpretação desse protocolo. Para que ela funcione, devemos respeitar o seguinte:

O endereço (posição dentro da memória) no qual queremos escrever foi dividido em 2 bytes: ENDERECO_HIGH (P10:P8) e ENDERECO_LOW (P7:P0), pois essa memória possui 0,5 kbyte disponível e, portanto, mais de 8 bits são necessários para endereçar toda a memória. Assim sendo, para escrever um byte nessa memória devemos primeiramente enviar a posição (endereço interno) e depois o dado.

Esse tipo de memória não aceita endereçamento na rede, isto é, aquele endereçamento suportado pelo protocolo I²C. Por isso, ela substitui o byte de endereçamento por outro chamado de controle. O byte de controle é composto por:

1	0	1	0	P10	P9	P8	R/W

Em que P10:P8 é a parte alta da posição onde desejamos escrever dentro da memória e R/W é o flag que indica se vamos efetuar uma operação de escrita (R/W=0) ou leitura (R/W=1). No caso dessa memória, só é usado o bit P8 (512 bytes -> 9 bits).

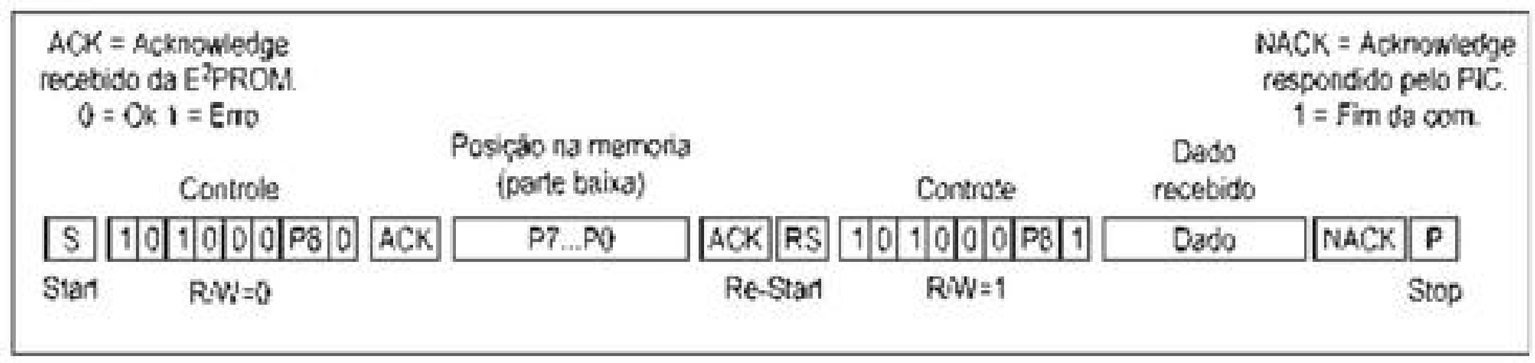
O segundo byte enviado (byte de dado no protocolo I²C) é equivalente ao complemento do endereço, isto é, sua parte baixa:

P7	P6	P5	P4	P3	P2	P1	P0

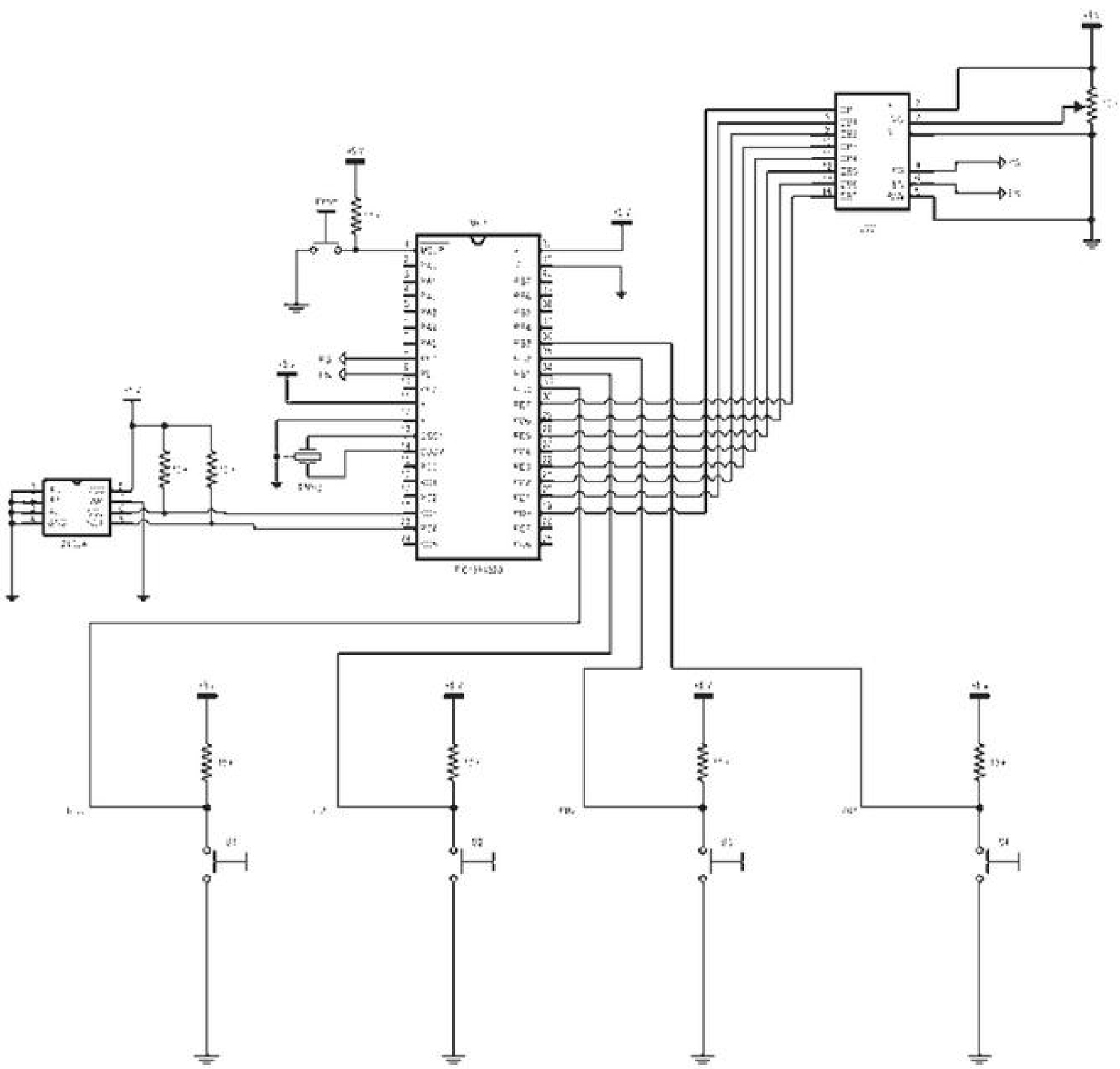
Desta forma, temos operações diferenciadas para a escrita e leitura da E²PROM externa.

Escrita

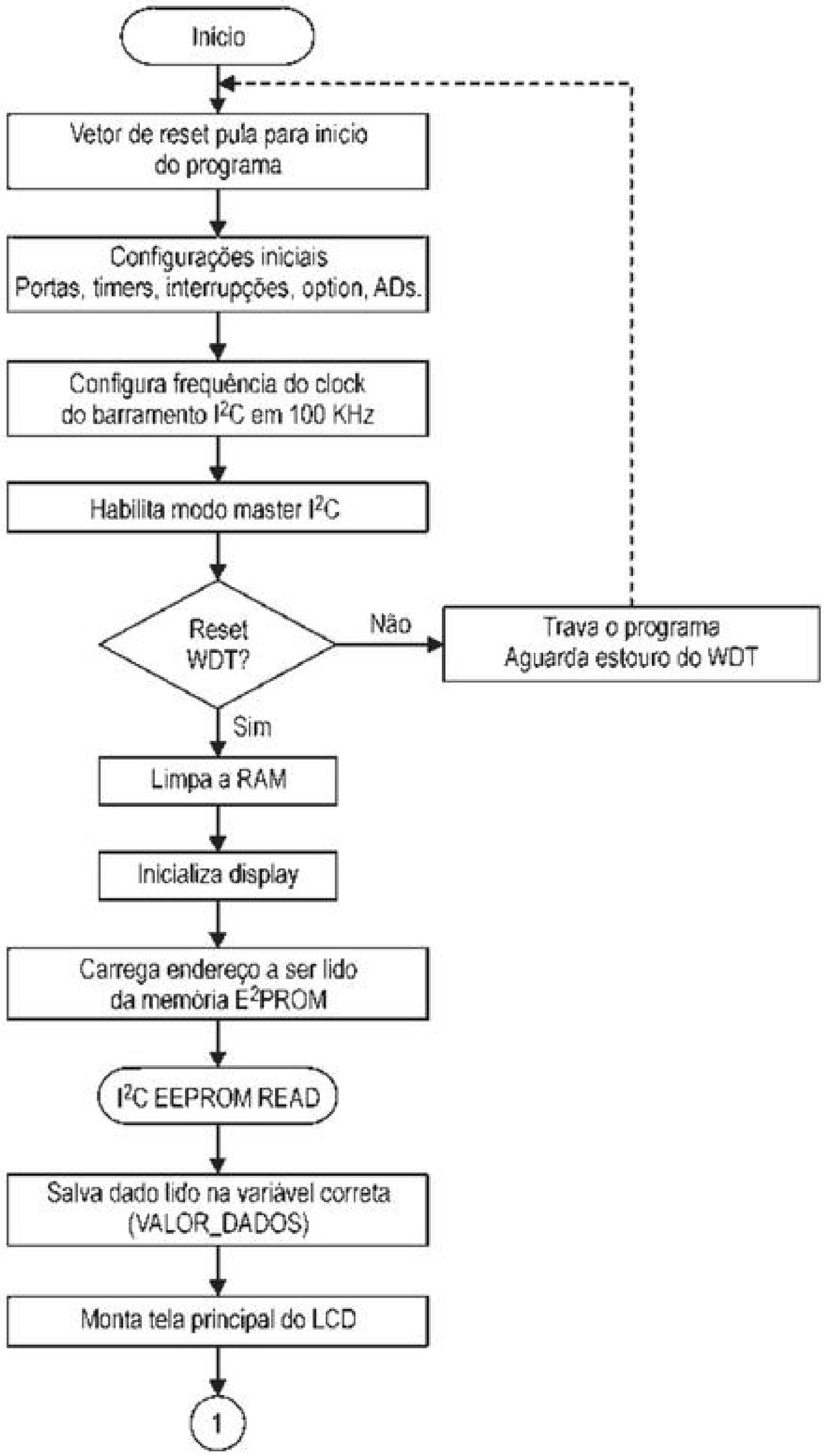
Leitura

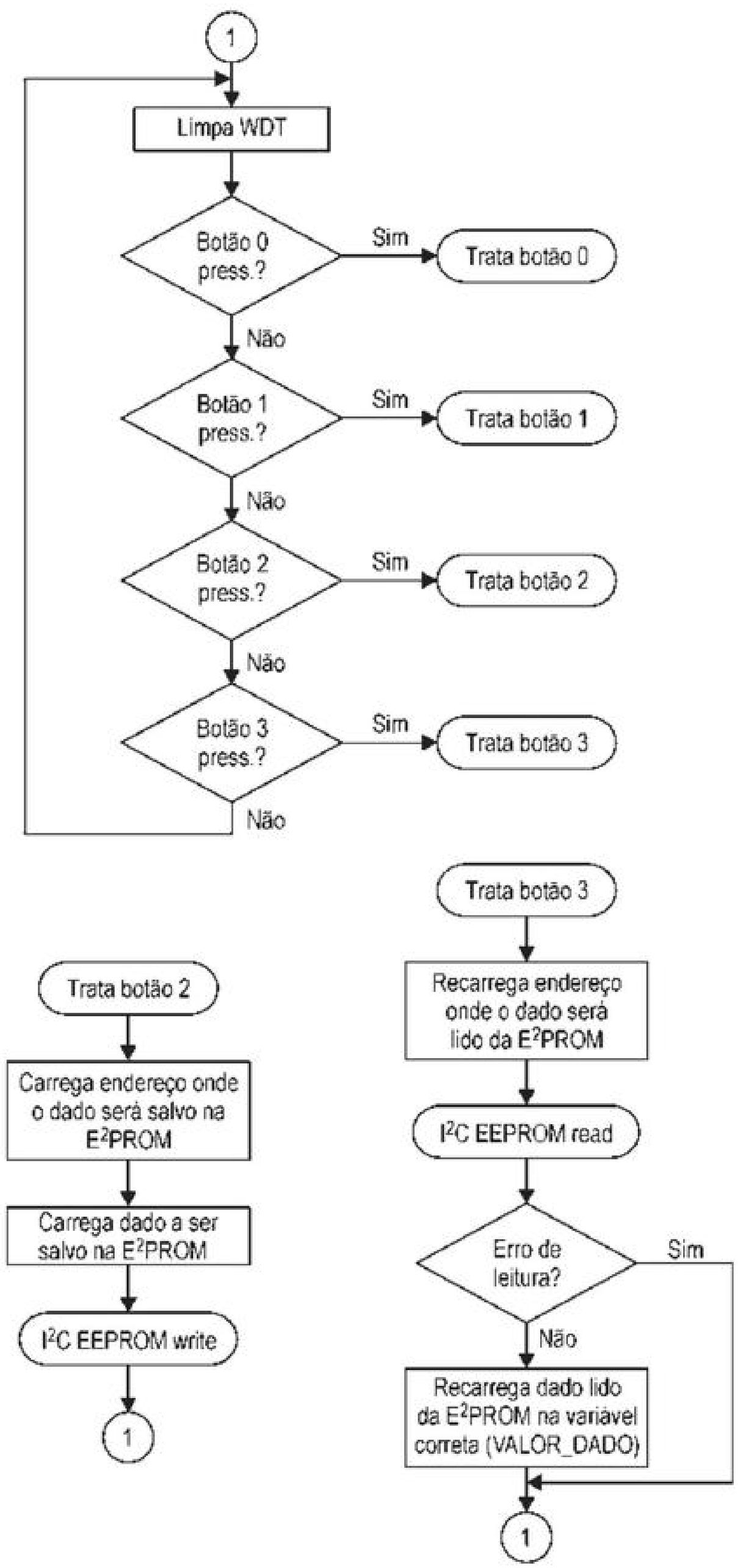


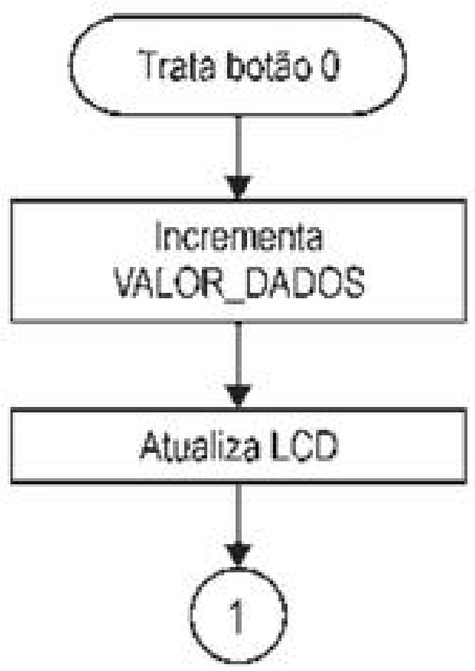
» Esquema elétrico

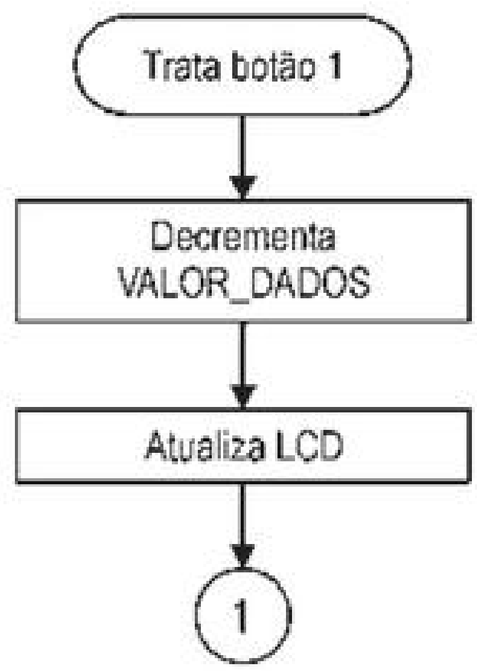


» Fluxograma

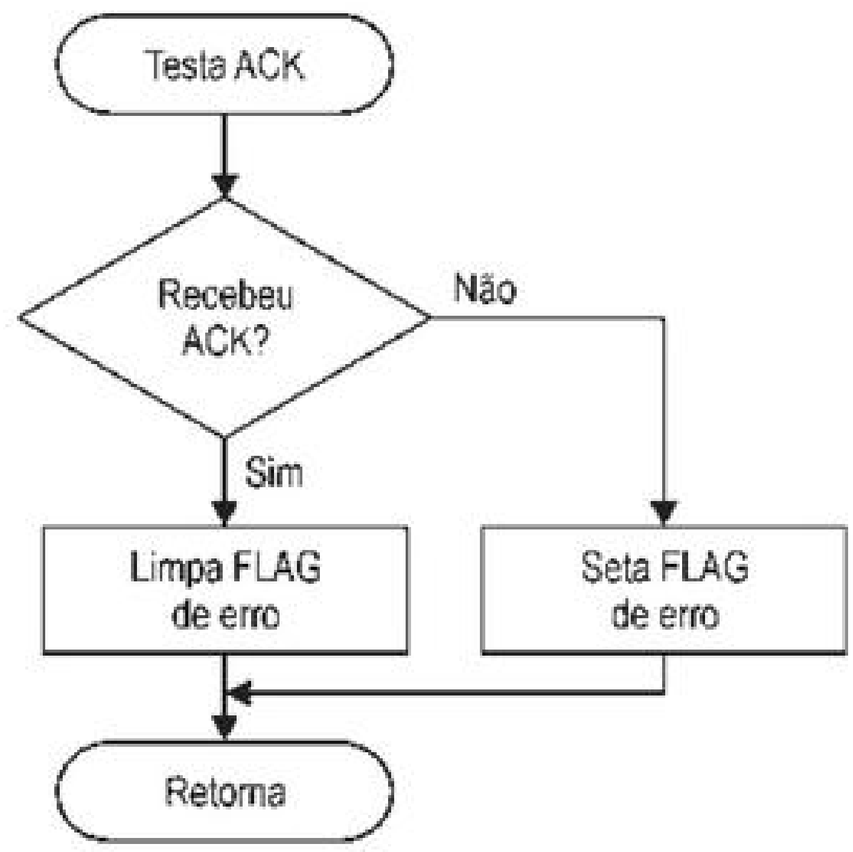

Inicio
Vetor de reset pula para inicio do programa
Configurações iniciais Portas, timers, interrupções, option, ADs.
Configura frequência do clock do barramento I²C em 100 KHz
Habilita modo master I²C
Reset WDT?
Não
Trava o programa Aguarda estouro do WDT
Sim
Limpa a RAM
Inicializa display
Carrega endereço a ser lido da memória E²PROM
I²C EEPROM READ
Salva dado lido na variável correta (VALOR_DADOS)
Monta tela principal do LCD
1

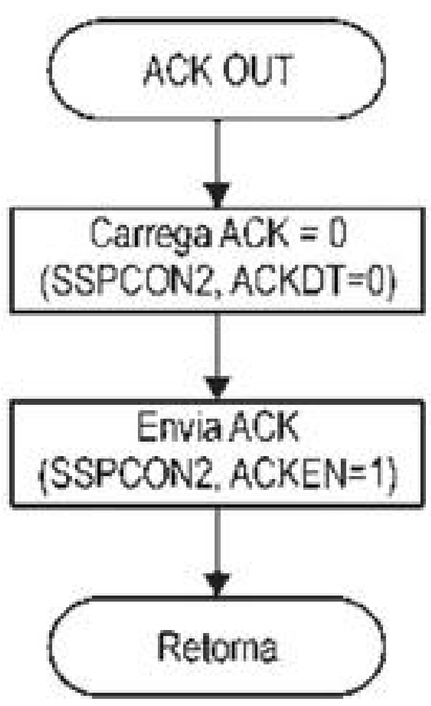

1
Limpa WDT
Botão 0 press.?
Sim
Trata botão 0
Não
Botão 1 press.?
Sim
Trata botão 1
Não
Botão 2 press.?
Sim
Trata botão 2
Não
Botão 3 press.?
Sim
Trata botão 3
Não
Trata botão 2
Carrega endereço onde o dado será salvo na E2PROM
Carrega dado a ser salvo na E2PROM
I2C EEPROM write
1
Trata botão 3
Recarrega endereço onde o dado será lido da E2PROM
I2C EEPROM read
Erro de leitura?
Sim
Não
Recarrega dado lido da E2PROM na variável correta (VALOR_DADO)
1

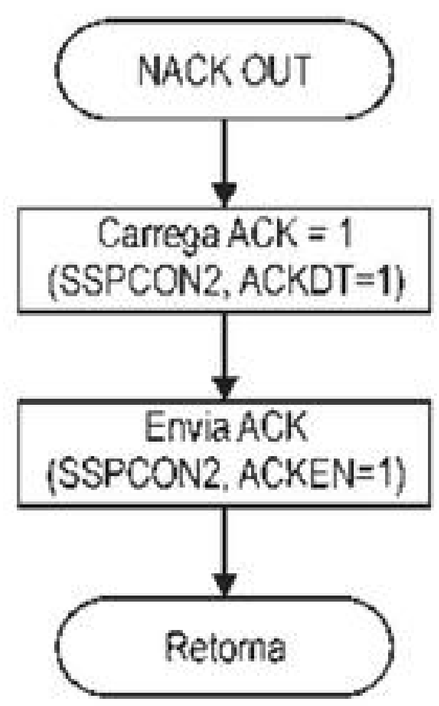

Trata botão 0
Incrementa VALOR_DADOS
Atualiza LCD
1

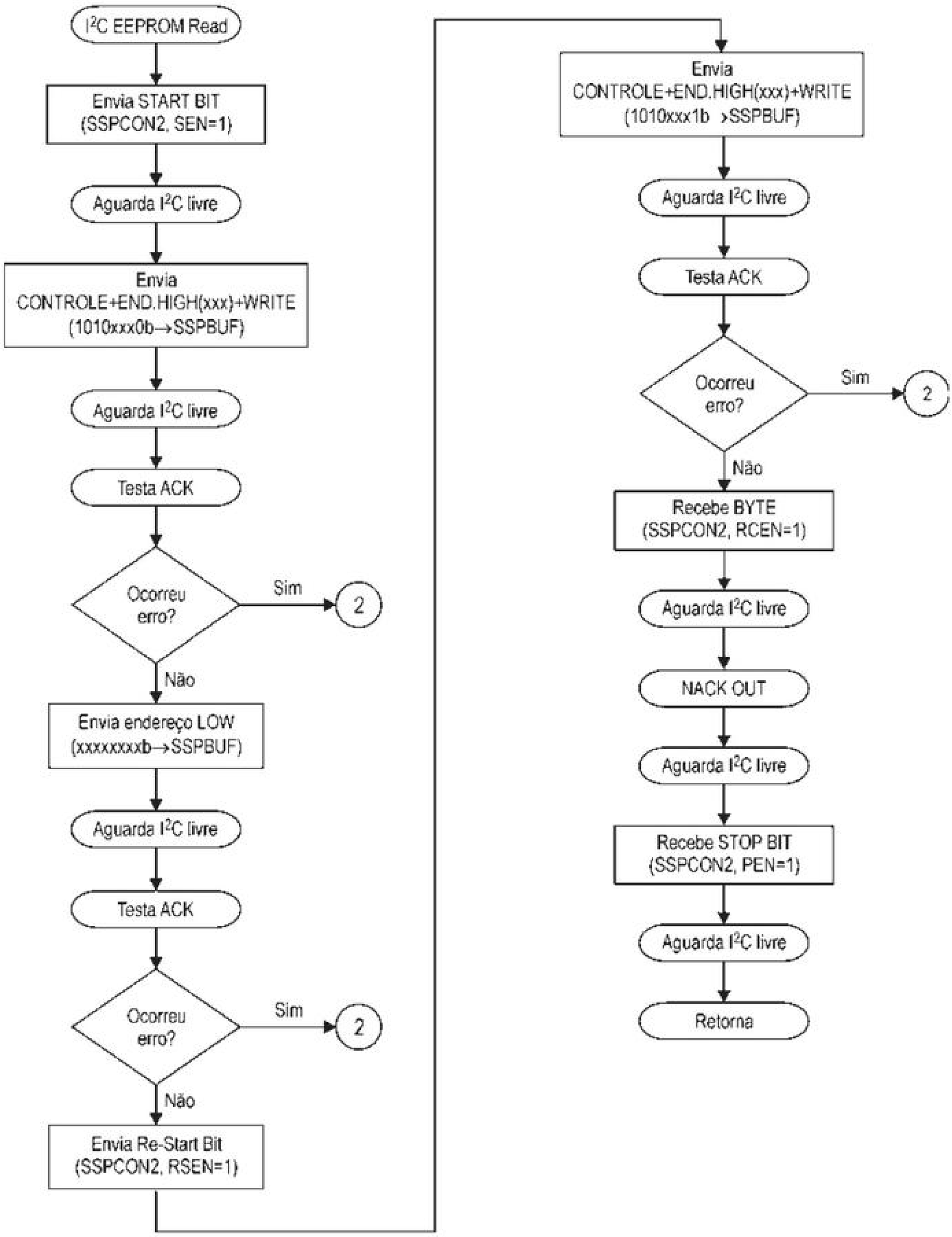

Trata botão 1
Decrementa VALOR_DADOS
Atualiza LCD
1

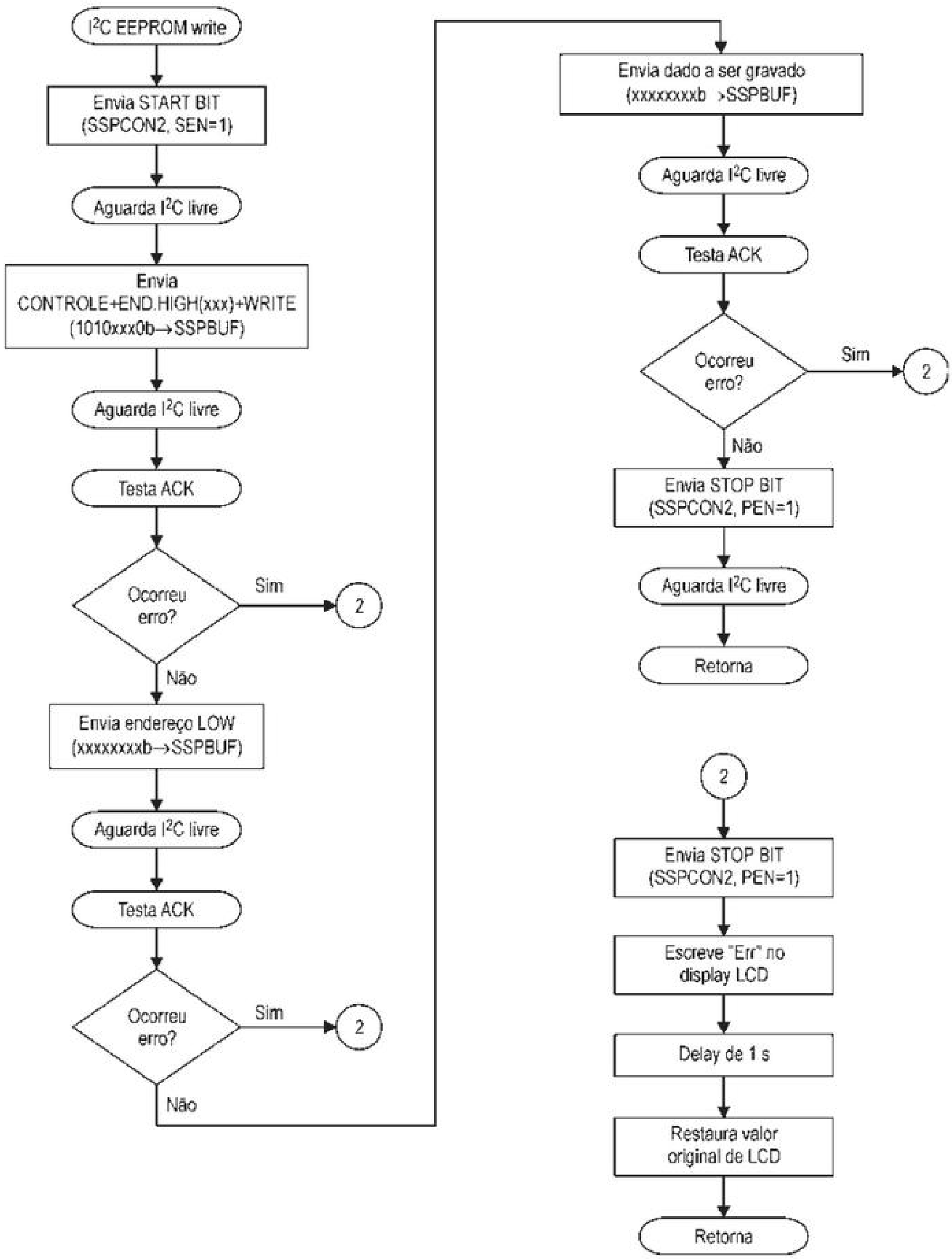
Aguarda I²C livre
Sim
Evento I²C em andamento?
Não
Não
Todos os eventos I²C finalizados?
Sim
Retorna


Testa ACK
Recebeu ACK?
Não
Sim
Limpa FLAG de erro
Seta FLAG de erro
Retorna


ACK OUT
Carrega ACK = 0 (SSPCON2, ACKDT=0)
Envia ACK (SSPCON2, ACKEN=1)
Retorna


NACK OUT
Carrega ACK = 1 (SSPCON2, ACKDT=1)
Envia ACK (SSPCON2, ACKEN=1)
Retorna


I²C EEPROM Read
Envia START BIT (SSPCON2, SEN=1)
Aguarda I²C livre
Envia CONTROLE+END.HIGH(xxx)+WRITE (1010xxx0b→SSPBUF)
Aguarda I²C livre
Testa ACK
Ocorreu erro?
Sim
2
Não
Envia endereço LOW (xxxxxxxxb→SSPBUF)
Aguarda I²C livre
Testa ACK
Ocorreu erro?
Sim
2
Não
Envia Re-Start Bit (SSPCON2, RSEN=1)
Envia CONTROLE+END.HIGH(xxx)+WRITE (1010xxx1b ›SSPBUF)
Aguarda I²C livre
Testa ACK
Ocorreu erro?
Sim
2
Não
Recebe BYTE (SSPCON2, RCEN=1)
Aguarda I²C livre
NACK OUT
Aguarda I²C livre
Recebe STOP BIT (SSPCON2, PEN=1)
Aguarda I²C livre
Retorna


I²C EEPROM write
Envia START BIT (SSPCON2, SEN=1)
Aguarda I²C livre
Envia CONTROLE+END.HIGH(xxx)+WRITE (1010xxx0b→SSPBUF)
Aguarda I²C livre
Testa ACK
Ocorreu erro?
Sim
2
Não
Envia endereço LOW (xxxxxxxxb→SSPBUF)
Aguarda I²C livre
Testa ACK
Ocorreu erro?
Sim
2
Não
Envia dado a ser gravado (xxxxxxxxb →SSPBUF)
Aguarda I²C livre
Testa ACK
Ocorreu erro?
Sim
2
Não
Envia STOP BIT (SSPCON2, PEN=1)
Aguarda I²C livre
Retorna
2
Envia STOP BIT (SSPCON2, PEN=1)
Escreve "Err" no display LCD
Delay de 1 s
Restaura valor original de LCD
Retorna

» Código

```
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *              DESBRAVANDO O PIC18 - RECURSOS AVANÇADOS                   *
; *                              EXEMPLO 8                                  *
; *                                                                         *
; * DANIEL RODRIGUES DE SOUSA, DAVID JOSÉ DE SOUZA e  NICOLÁS CÉSAR LAVINIA *
; *                                                                         *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *     DATA : 03/08/2009                                                   *
; *   VERSÃO : 1.0                                                          *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                            DESCRIÇÃO GERAL                              *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTE EXEMPLO FOI ELABORADO PARA EXPLICAR O FUNCIONAMENTO DA LEITURA/ESCRITA
;  NA MEMÓRIA E2PROM SERIAL EXTERNA, UTILIZANDO O MASTER I2C.
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                        CONFIGURAÇÕES PARA GRAVAÇÃO                      *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
    CONFIG  OSC = XT, FCMEN = OFF, IESO = OFF, PWRT = ON, BOREN = ON, BORV = 0
    CONFIG  WDT = ON, WDTPS = 128, MCLRE = ON, LPT1OSC = OFF, PBADEN = OFF
    CONFIG  CCP2MX = PORTC, STVREN = ON, LVP = OFF, DEBUG = OFF, XINST = OFF
    CONFIG  CP0 = OFF, CP1 = OFF, CP2 = OFF, CP3 = OFF, CPB = OFF, CPD = OFF
    CONFIG  WRT0 = OFF, WRT1 = OFF, WRT2 = OFF, WRT3 = OFF, WRTB = OFF
    CONFIG  WRTC = OFF, WRTD = OFF, EBTR0 = OFF, EBTR1 = OFF, EBTR2 = OFF
    CONFIG  EBTR3 = OFF, EBTRB = OFF

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                          DEFINIÇÃO DAS VARIÁVEIS                        *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
    CBLOCK  0X00            ; BANK 0 --> 0X00..0X5F --> ACCESS BANK

        FILTRO_BOTOES       ; FILTRO PARA RUIDOS
        TEMPO_TURBO         ; TEMPORIZADOR P/ TURBO DAS TECLAS

        TEMPO1
        TEMPO0              ; CONTADORES P/ DELAY

        FLAG                ; FLAG DE USO GERAL

        AUX                 ; REGISTRADOR AUXILIAR DE USO GERAL

        ENDERECO_HIGH       ; REGISTRADORES DE ENDEREÇO PARA
        ENDERECO_LOW        ; ACESSO À MEMÓRIA EEPROM SERIAL EXTERNA
                            ; MAPEADOS NO BANCO 0 DA RAM
        BUFFER              ; REGISTRADOR PARA LEITURA/GRAVAÇÃO NA EEPROM SERIAL
                            ; EXTERNA
        VALOR_DADOS         ; REGISTRADOR DE DADO PARA EEPROM SERIAL EXTERNA
                            ; MAPEADO NO BANCO 0 DA RAM
    ENDC

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                DEFINIÇÃO DAS VARIÁVEIS INTERNAS DO PIC                  *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  O ARQUIVO DE DEFINIÇÕES DO PIC UTILIZADO DEVE SER REFERENCIADO PARA QUE
;  OS NOMES DEFINIDOS PELA MICROCHIP POSSAM SER UTILIZADOS, SEM A NECESSIDADE
;  DE REDIGITAÇÃO.

    #INCLUDE <P18F4520.INC> ; MICROCONTROLADOR UTILIZADO
```

```
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                          CONSTANTES INTERNAS                            *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  A DEFINIÇÃO DE CONSTANTES FACILITA A PROGRAMAÇÃO E A MANUTENÇÃO.

FILTRO_TECLA    EQU .200    ; FILTRO P/ EVITAR RUIDOS DOS BOTÕES

TURBO_TECLA     EQU .60     ; TEMPORIZADOR P/ TURBO DAS TECLAS

END_EEPROM_H    EQU 0X00    ; ENDEREÇO P/ LEITURA E GRAVAÇÃO
END_EEPROM_L    EQU 0X00    ; NA MEMÓRIA EEPROM SERIAL

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                   DECLARAÇÃO DOS FLAGs DE SOFTWARE                      *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  A DEFINIÇÃO DE FLAGs AJUDA NA PROGRAMAÇÃO E ECONOMIZA MEMÓRIA RAM.

#DEFINE F_ERRO  FLAG,0      ; 1 --> ERRO NA LEITURA DA EEPROM SERIAL

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                              ENTRADAS                                   *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  AS ENTRADAS DEVEM SER ASSOCIADAS A NOMES PARA FACILITAR A PROGRAMAÇÃO E
;  FUTURAS ALTERAÇÕES DO HARDWARE.

#DEFINE BOTAO_0   PORTB,0           ; ESTADO DO BOTÃO 0
                                    ; 1 -> LIBERADO
                                    ; 0 -> PRESSIONADO

#DEFINE BOTAO_1   PORTB,1           ; ESTADO DO BOTÃO 1
                                    ; 1 -> LIBERADO
                                    ; 0 -> PRESSIONADO

#DEFINE BOTAO_2   PORTB,2           ; ESTADO DO BOTÃO 2
                                    ; 1 -> LIBERADO
                                    ; 0 -> PRESSIONADO

#DEFINE BOTAO_3   PORTB,3           ; ESTADO DO BOTÃO 3
                                    ; 1 -> LIBERADO
                                    ; 0 -> PRESSIONADO

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                              SAÍDAS                                     *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  AS SAÍDAS DEVEM SER ASSOCIADAS A NOMES PARA FACILITAR A PROGRAMAÇÃO E
;  FUTURAS ALTERAÇÕES DO HARDWARE.

#DEFINE DISPLAY     LATD            ; BARRAMENTO DE DADOS DO DISPLAY

#DEFINE RS          LATE,0          ; INDICA P/ O DISPLAY UM DADO OU COMANDO
                                    ; 1 -> DADO
                                    ; 0 -> COMANDO

#DEFINE ENABLE      LATE,1          ; SINAL DE ENABLE P/ DISPLAY
                                    ; ATIVO NA BORDA DE DESCIDA

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                  VETOR DE RESET DO MICROCONTROLADOR                     *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  POSIÇÃO INICIAL PARA EXECUÇÃO DO PROGRAMA

    ORG     0X0000                  ; ENDEREÇO DO VETOR DE RESET
    BRA     CONFIGURACAO            ; PULA PARA CONFIG DEVIDO A REGIÃO
                                    ; DESTINADA AS ROTINAS SEGUINTES
```

```
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                    ROTINA DE DELAY (DE 1MS ATÉ 256MS)                   *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTA É UMA ROTINA DE DELAY VARIÁVEL, COM DURAÇÃO DE 1MS X O VALOR PASSADO
;  EM WORK (W).

DELAY_MS
    MOVWF   TEMPO1          ; CARREGA TEMPO1 (UNIDADES DE MS)
    MOVLW   .250
    MOVWF   TEMPO0          ; CARREGA TEMPO0 (P/ CONTAR 1MS)

    CLRWDT                  ; LIMPA WDT (PERDE TEMPO)
    DECFSZ  TEMPO0,F        ; FIM DE TEMPO0 ?
    BRA     $-.4            ; NÃO - VOLTA 2 INSTRUÇÕES
                            ; SIM - PASSOU-SE 1ms
    DECFSZ  TEMPO1,F        ; FIM DE TEMPO1 ?
    BRA     $-.12           ; NÃO - VOLTA 6 INSTRUÇÕES
    RETURN                  ; SIM - RETORNA

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *               ROTINA DE ESCRITA DE UM CARACTERE NO DISPLAY              *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTA ROTINA ENVIA UM CARACTERE PARA O MÓDULO DE LCD. O CARACTERE A SER
;  ESCRITO DEVE SER COLOCADO EM WORK (W) ANTES DE CHAMAR A ROTINA.

ESCREVE
    MOVWF   DISPLAY         ; ATUALIZA DISPLAY (PORTD)
    NOP                     ; PERDE 1US PARA ESTABILIZAÇÃO
    BSF     ENABLE          ; ENVIA UM PULSO DE ENABLE AO DISPLAY
    BRA     $+.2            ; .
    BCF     ENABLE          ; .

    MOVLW   .1
    RCALL   DELAY_MS        ; DELAY DE 1ms
    RETURN                  ; RETORNA

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *               ROTINA DE ESCRITA LINHA 1 DO LCD                          *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTA ROTINA ESCREVE A LINHA 1 DA TELA PRINCIPAL DO LCD, COM A FRASE:
;  LINHA 1 - "   MASTER I2C   "

ATUALIZA_TELA_LINHA_1
    BCF     RS              ; SELECIONA O DISPLAY P/ COMANDOS
    MOVLW   0X83            ; COMANDO PARA POSICIONAR O CURSOR
    RCALL   ESCREVE         ; LINHA 0 / COLUNA 3
    BSF     RS              ; SELECIONA O DISPLAY P/ DADOS

    MOVLW   'M'             ; COMANDOS PARA ESCREVER AS
    RCALL   ESCREVE         ; LETRAS DE "MASTER I2C"
    MOVLW   'A'
    RCALL   ESCREVE
    MOVLW   'S'
    RCALL   ESCREVE
    MOVLW   'T'
    RCALL   ESCREVE
    MOVLW   'E'
    RCALL   ESCREVE
    MOVLW   'R'
    RCALL   ESCREVE
    MOVLW   ' '
    RCALL   ESCREVE
    MOVLW   'I'
    RCALL   ESCREVE
    MOVLW   '2'
```

```
    RCALL   ESCREVE
    MOVLW   'C'
    RCALL   ESCREVE
    RETURN                      ; RETORNA

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *               ROTINA DE ESCRITA LINHA 2 DO LCD                        *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTA ROTINA ESCREVE A LINHA 2 DA TELA PRINCIPAL DO LCD.
;  A ROTINA LEVA EM CONTA A VARIÁVEL VALOR_DADOS PARA FORMAR A LINHA 2.

ATUALIZA_TELA_LINHA_2
    BCF     RS              ; SELECIONA O DISPLAY P/ COMANDO
    MOVLW   0XC6            ; COMANDO PARA POSICIONAR O CURSOR
    RCALL   ESCREVE         ; LINHA 1 / COLUNA 6

    BSF     RS              ; SELECIONA O DISPLAY P/ DADOS

    SWAPF   VALOR_DADOS,W   ; INVERTE NIBLE DO VALOR_DADOS
    ANDLW   B'00001111'     ; MASCARA BITS MAIS SIGNIFICATIVOS
    MOVWF   AUX             ; SALVA EM AUXILIAR

    MOVLW   0X0A
    SUBWF   AUX,W           ; AUX - 10d (ATUALIZA FLAG DE CARRY)
    MOVLW   0X30            ; CARREGA WORK COM 30h
    BTFSC   STATUS,C        ; RESULTADO É POSITIVO? (É UMA LETRA?)
    MOVLW   0X37            ; SIM - CARREGA WORK COM 37h
                            ; NÃO - WORK FICA COM 30h (NÚMERO)
    ADDWF   AUX,W           ; SOMA O WORK AO AUXILIAR
                            ; (CONVERSÃO ASCII)
    RCALL   ESCREVE         ; ENVIA CARACTERE AO DISPLAY LCD

    MOVF    VALOR_DADOS,W   ; CARREGA WORK COM VALOR_DADOS
    ANDLW   B'00001111'     ; MASCARA BITS MAIS SIGNIFICATIVOS
    MOVWF   AUX             ; SALVA EM AUXILIAR

    MOVLW   0X0A
    SUBWF   AUX,W           ; AUX - 10d (ATUALIZA FLAG DE CARRY)
    MOVLW   0X30            ; CARREGA WORK COM 30h
    BTFSC   STATUS,C        ; RESULTADO É POSITIVO? (É UMA LETRA?)
    MOVLW   0X37            ; SIM - CARREGA WORK COM 37h
                            ; NÃO - WORK FICA COM 30h (NÚMERO)
    ADDWF   AUX,W           ; SOMA O WORK AO AUXILIAR
                            ; (CONVERSÃO ASCII)
    RCALL   ESCREVE         ; ENVIA CARACTERE AO DISPLAY LCD

    MOVLW   'h'
    RCALL   ESCREVE         ; ESCREVE "h" NO DISPLAY
    RETURN                  ; RETORNA

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *               ROTINA DE CHECAGEM DE EVENTOS I2C LIBERADOS             *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; ESTA ROTINA AGUARDA ATÉ QUE TODOS OS EVENTOS DA I2C ESTEJAM LIBERADOS.

AGUARDA_I2C_LIVRE
    BTFSC   SSPSTAT,R_W     ; ESTÁ OCORRENDO ALGUM EVENTO I2C?
    BRA     $-.2            ; SIM - ESPERA TERMINAR

    MOVF    SSPCON2,W
    ANDLW   B'00011111'     ; MASCARA SSPCON2 (ATUALIZA FLAG ZERO)
    BNZ     $-.4            ; BITS DE EVENTOS LIBERADOS? NÃO - AGUARDA

    RETURN                  ; SIM - RETORNA
```

```
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                                ACK OUT                                  *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; ESTA ROTINA ENVIA UM ACK OUT PARA O BARRAMENTO I2C.

ACK_OUT
    BCF     SSPCON2,ACKDT   ; CARREGA ACK
    BSF     SSPCON2,ACKEN   ; TRANSMITE
    RETURN                  ; RETORNA

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                               NACK OUT                                  *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; ESTA ROTINA ENVIA UM NACK OUT PARA O BARRAMENTO I2C.

NACK_OUT
    BSF     SSPCON2,ACKDT   ; CARREGA NACK
    BSF     SSPCON2,ACKEN   ; TRANSMITE
    RETURN                  ; RETORNA

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                  ROTINA PARA TESTAR SE O ACK FOI RECEBIDO               *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; ESTA ROTINA TESTA O BIT DE ACK RECEBIDO NO REGISTRADOR SSPCON2. PARA
; FACILITAR O RESTANTE DO SOFTWARE, A ROTINA COPIA ESTE FLAG NO FLAG F_ERRO
; PRESENTE NO BANCO 0 DA RAM, POIS O REGISTRADOR SSPCON2 ENCONTRA-SE NO BANK1.

TESTA_ACK
    BTFSC   SSPCON2,ACKSTAT ; RECEBEU ACK ?
    BRA     RECEBEU_NACK    ; NÃO - SINALIZA ERRO
    BCF     F_ERRO          ; SIM - LIMPA FLAG DE ERRO
    RETURN                  ; RETORNA

RECEBEU_NACK
    BSF     F_ERRO          ; SETA FLAG P/ INDICAR ERRO
    RETURN                  ; RETORNA

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                  LEITURA DA EEPROM SERIAL EXTERNA                       *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; ESTA ROTINA LÊ A MEMÓRIA SERIAL EXTERNA. O ENDEREÇO DEVE SER PASSADO PELAS
; VARIÁVEIS ENDERECO_HIGH E ENDERECO_LOW. O VALOR LIDO É RETORNADO EM BUFFER.
; CASO ALGUM ERRO DE LEITURA OCORRA, A ROTINA DESVIA P/ I2C_ERRO.

I2C_EEPROM_READ
    BSF     SSPCON2,SEN         ; INICIA START BIT
    RCALL   AGUARDA_I2C_LIVRE   ; AGUARDA FIM DO EVENTO

    RLNCF   ENDERECO_HIGH,W     ; ROTACIONA ENDERECO_HIGH
    IORLW   B'10100000'         ; JUNTA AO BYTE DE CONTROLE
    MOVWF   SSPBUF              ; TRANSMITE CONTROLE + END_HIGH
    RCALL   AGUARDA_I2C_LIVRE   ; AGUARDA FIM DO EVENTO
    RCALL   TESTA_ACK           ; CHAMA ROTINA P/ TESTAR ACK
    BTFSC   F_ERRO              ; OCORREU ERRO DE ACK ?
    BRA     I2C_ERRO            ; SIM - PULA P/ I2C_ERRO
                                ; NÃO
    MOVFF   ENDERECO_LOW,SSPBUF ; TRANSMITE ENDEREÇO BAIXO
    RCALL   AGUARDA_I2C_LIVRE   ; AGUARDA FIM DO EVENTO
    RCALL   TESTA_ACK           ; CHAMA ROTINA P/ TESTAR ACK
    BTFSC   F_ERRO              ; OCORREU ERRO DE ACK ?
    BRA     I2C_ERRO            ; SIM - PULA P/ I2C_ERRO

    BSF     SSPCON2,RSEN        ; NÃO - REINICIA START BIT
    RCALL   AGUARDA_I2C_LIVRE   ; AGUARDA FIM DO EVENTO
```

```
RLNCF   ENDERECO_HIGH,W      ; ROTACIONA ENDERECO_HIGH
IORLW   B'10100001'          ; JUNTA AO BYTE DE CONTROLE
MOVWF   SSPBUF               ; TRANSMITE CONTROLE + END_HIGH
RCALL   AGUARDA_I2C_LIVRE    ; AGUARDA FIM DO EVENTO
RCALL   TESTA_ACK            ; CHAMA ROTINA P/ TESTAR ACK
BTFSC   F_ERRO               ; OCORREU ERRO DE ACK ?
BRA     I2C_ERRO             ; SIM - PULA P/ I2C_ERRO

BSF     SSPCON2,RCEN         ; NÃO - INICIA LEITURA DO BYTE
RCALL   AGUARDA_I2C_LIVRE    ; AGUARDA FIM DO EVENTO
MOVFF   SSPBUF,BUFFER        ; SALVA DADO EM BUFFER
RCALL   NACK_OUT             ; ENVIA NACK --> FIM
RCALL   AGUARDA_I2C_LIVRE    ; AGUARDA FIM DO EVENTO

BSF     SSPCON2,PEN          ; NÃO - INICIA STOP BIT
RCALL   AGUARDA_I2C_LIVRE    ; AGUARDA FIM DO EVENTO
RETURN                       ; RETORNA

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                    ESCRITA NA EEPROM SERIAL EXTERNA                   *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; ESTA ROTINA GRAVA UM DADO NA MEMÓRIA SERIAL EXTERNA. O ENDEREÇO DEVE SER
; PASSADO PELAS VARIÁVEIS ENDERECO_HIGH E ENDERECO_LOW. O VALOR A SER GRAVADO
; DEVE SER PASSADO EM BUFFER.
; CASO ALGUM ERRO DE GRAVAÇÃO OCORRA, A ROTINA DESVIA P/ I2C_ERRO.

I2C_EEPROM_WRITE
    BSF     SSPCON2,SEN      ; INICIA START BIT
    RCALL   AGUARDA_I2C_LIVRE ; AGUARDA FIM DO EVENTO

    RLNCF   ENDERECO_HIGH,W  ; ROTACIONA ENDERECO_HIGH
    IORLW   B'10100000'      ; JUNTA AO BYTE DE CONTROLE
    MOVWF   SSPBUF           ; TRANSMITE CONTROLE + END_HIGH
    RCALL   AGUARDA_I2C_LIVRE ; AGUARDA FIM DO EVENTO
    RCALL   TESTA_ACK        ; CHAMA ROTINA P/ TESTAR ACK
    BTFSC   F_ERRO           ; OCORREU ERRO DE ACK ?
    BRA     I2C_ERRO         ; SIM - PULA P/ I2C_ERRO

    MOVFF   ENDERECO_LOW,SSPBUF ; NÃO - TRANSMITE ENDEREÇO BAIXO
    RCALL   AGUARDA_I2C_LIVRE ; AGUARDA FIM DO EVENTO
    RCALL   TESTA_ACK        ; CHAMA ROTINA P/ TESTAR ACK
    BTFSC   F_ERRO           ; OCORREU ERRO DE ACK ?
    BRA     I2C_ERRO         ; SIM - PULA P/ I2C_ERRO

    MOVFF   BUFFER,SSPBUF    ; NÃO - GRAVA DADO
    RCALL   AGUARDA_I2C_LIVRE ; AGUARDA FIM DO EVENTO
    RCALL   TESTA_ACK        ; CHAMA ROTINA P/ TESTAR ACK
    BTFSC   F_ERRO           ; OCORREU ERRO DE ACK ?
    BRA     I2C_ERRO         ; SIM - PULA P/ I2C_ERRO

    BSF     SSPCON2,PEN      ; NÃO - INICIA STOP BIT
    RCALL   AGUARDA_I2C_LIVRE ; AGUARDA FIM DO EVENTO
    RETURN                   ; RETORNA

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                    ROTINA P/ SINALIZAR ERRO NA I2C                    *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; ESTA ROTINA SOMENTE É EXECUTADA CASO ALGUM ERRO DE LEITURA/GRAVAÇÃO OCORRA
; COM A MEMÓRIA SERIAL.
; A ROTINA ENVIA UM STOP BIT PARA FINALIZAR A COMUNICAÇÃO COM A MEMÓRIA
; SERIAL, ENVIA UMA MENSAGEM DE ERRO AO DISPLAY E APÓS 1s RETORNA À TELA
; PRINCIPAL.

I2C_ERRO
    BSF     SSPCON2,PEN      ; INICIA STOP BIT
```

```
BCF     RS                      ; SELECIONA O DISPLAY P/ COMANDO
MOVLW   0XC6                    ; COMANDO PARA POSICIONAR O CURSOR
RCALL   ESCREVE                 ; LINHA 1 / COLUNA 6
BSF     RS                      ; SELECIONA O DISPLAY P/ DADOS

MOVLW   'E'
RCALL   ESCREVE
MOVLW   'r'
RCALL   ESCREVE
MOVLW   'r'
RCALL   ESCREVE                 ; ESCREVE "Err" NO LCD

MOVLW   .250
RCALL   DELAY_MS
MOVLW   .250
RCALL   DELAY_MS
MOVLW   .250
RCALL   DELAY_MS                ; DELAY DE 1seg.

RCALL   ATUALIZA_TELA_LINHA_2   ; ATUALIZA TELA PRINCIPAL
RETURN                          ; RETORNA

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                CONFIGURAÇÕES INICIAIS DE HARDWARE E SOFTWARE          *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  NESTA ROTINA SÃO INICIALIZADAS AS PORTAS DE I/O DO MICROCONTROLADOR E AS
;  CONFIGURAÇÕES DOS REGISTRADORES ESPECIAIS (SFR). A ROTINA INICIALIZA AS
;  VARIÁVEIS DE RAM E AGUARDA O ESTOURO DO WDT.

CONFIGURACAO
    CLRF    PORTA           ; GARANTE AS SAÍDAS EM ZERO
    CLRF    PORTB
    CLRF    PORTC
    CLRF    PORTD
    CLRF    PORTE

    CLRF    LATA
    CLRF    LATB
    CLRF    LATC
    CLRF    LATD
    CLRF    LATE

    MOVLW   B'11111111'
    MOVWF   TRISA           ; CONFIGURA I/O DO PORTA

    MOVLW   B'11111111'
    MOVWF   TRISB           ; CONFIGURA I/O DO PORTB

    MOVLW   B'11111111'
    MOVWF   TRISC           ; CONFIGURA I/O DO PORTC

    MOVLW   B'00000000'
    MOVWF   TRISD           ; CONFIGURA I/O DO PORTD

    MOVLW   B'00000100'
    MOVWF   TRISE           ; CONFIGURA I/O DO PORTE

    MOVLW   B'11011000'     ; CONFIGURA TIMER0
    MOVWF   T0CON           ; TIMER0 INCREM. PELO CICLO DE MÁQUINA
                            ; TIMER0- 1:1

    MOVLW   B'00000000'     ; CONFIGURA INTERRUPÇÕES
    MOVWF   INTCON          ; DESABILITADAS TODAS AS INTERRUPÇÕES
```

```
    MOVLW   B'00001111'     ; CONFIGURA CONVERSOR A/D
    MOVWF   ADCON1          ; CONFIGURA PORTA, PORTB E PORTE COMO I/O DIGITAL

    MOVLW   B'00101000'     ; HABILITA I2C - MASTER MODE
    MOVWF   SSPCON1         ; CONFIGURA PINOS COMO DA I2C

    MOVLW   B'10000000'
    MOVWF   SSPSTAT         ; DESABILITA SLEW-RATE CONTROL (100 KHz)

    MOVLW   B'00001001'
    MOVWF   SSPADD          ; VELOCIDADE: 100KHz @ 4MHz

;  AS INSTRUÇÕES A SEGUIR FAZEM COM QUE O PROGRAMA TRAVE QUANDO HOUVER UM
;  RESET OU POWER-UP, MAS PASSE DIRETO SE O RESET FOR POR WDT. DESTA FORMA,
;  SEMPRE QUE O PIC É LIGADO, O PROGRAMA TRAVA, AGUARDA UM ESTOURO DE WDT
;  E COMEÇA NOVAMENTE. ISTO EVITA PROBLEMAS NO START-UP DO PIC.

    BTFSC   RCON,NOT_TO     ; RESET POR ESTOURO DE WATCHDOG TIMER ?
    BRA     $               ; NÃO - AGUARDA ESTOURO DO WDT
                            ; SIM

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                         INICIALIZAÇÃO DA RAM                          *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTA ROTINA IRÁ LIMPAR TODA A RAM, DO BANCO 0 ATÉ AO BANCO 15

    LFSR    FSR0,0X00       ; APONTA FSR0 PARA ENDEREÇO ZERO

LIMPA_RAM
    CLRF    POSTINC0        ; LIMPA A POSIÇÃO E INCREMENTA FSR0
    CLRWDT                  ; LIMPA WATCHDOG TIMER
    MOVLW   0X0F            ; TESTA SE JÁ CHEGOU AO BANCO 15
    SUBWF   FSR0H,W
    BNZ     LIMPA_RAM       ; SE NÃO, CONTINUA LIMPANDO

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                   CONFIGURAÇÕES INICIAIS DO DISPLAY                   *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; ESTA ROTINA INICIALIZA O DISPLAY P/ COMUNICAÇÃO DE 8 VIAS, DISPLAY PARA 2
; LINHAS, CURSOR APAGADO E DESLOCAMENTO DO CURSOR À DIREITA.

INICIALIZACAO_DISPLAY
    BCF     RS              ; SELECIONA O DISPLAY P/ COMANDOS

    MOVLW   0X30            ; ESCREVE COMANDO 0X30 PARA
    RCALL   ESCREVE         ; INICIALIZAÇÃO

    MOVLW   .3
    RCALL   DELAY_MS        ; DELAY DE 3MS (EXIGIDO PELO DISPLAY)

    MOVLW   0X30            ; ESCREVE COMANDO 0X30 PARA
    RCALL   ESCREVE         ; INICIALIZAÇÃO

    MOVLW   0X30            ; ESCREVE COMANDO 0X30 PARA
    RCALL   ESCREVE         ; INICIALIZAÇÃO

    MOVLW   B'00111000'     ; ESCREVE COMANDO PARA
    RCALL   ESCREVE         ; INTERFACE DE 8 VIAS DE DADOS

    MOVLW   B'00000001'     ; ESCREVE COMANDO PARA
    RCALL   ESCREVE         ; LIMPAR TODO O DISPLAY

    MOVLW   .1
    RCALL   DELAY_MS        ; DELAY DE 1MS
```

```
    MOVLW   B'00001100'     ; ESCREVE COMANDO PARA
    RCALL   ESCREVE         ; LIGAR O DISPLAY SEM CURSOR

    MOVLW   B'00000110'     ; ESCREVE COMANDO PARA INCREM.
    RCALL   ESCREVE         ; AUTOMÁTICO À DIREITA

    BSF     RS              ; SELECIONA O DISPLAY P/ DADOS

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                        INICIALIZAÇÃO DA RAM                            *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTE TRECHO DO PROGRAMA LÊ O DADOS DA MEMÓRIAS E2PROM EXTERNA E
;  ATUALIZA A RAM.

LE_MEMORIA_EEPROM
    MOVLW   END_EEPROM_H
    MOVWF   ENDERECO_HIGH
    MOVLW   END_EEPROM_L
    MOVWF   ENDERECO_LOW        ; CARREGA ENDEREÇO P/ LEITURA

    RCALL   I2C_EEPROM_READ     ; CHAMA ROTINA P/ LER DADO

    MOVFF   BUFFER,VALOR_DADOS  ; SALVA DADO LIDO EM VALOR_DADOS

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *              ROTINA DE ESCRITA DA TELA PRINCIPAL                       *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTA ROTINA ESCREVE A TELA PRINCIPAL DO PROGRAMA, COM AS FRASES:
;  LINHA 1 - "   MASTER I2C   "
;  LINHA 2 - "      xxh       "

    RCALL   ATUALIZA_TELA_LINHA_1   ; ATUALIZA TELA LINHA 1 DO LCD

    RCALL   ATUALIZA_TELA_LINHA_2   ; ATUALIZA TELA LINHA 2 DO LCD

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                          VARREDURA DOS BOTÕES                          *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; ESTA ROTINA VERIFICA SE ALGUM BOTÃO ESTÁ PRESSIONADO E CASO AFIRMATIVO
; DESVIA PARA O TRATAMENTO DELE.

VARRE
    CLRWDT                  ; LIMPA WATCHDOG TIMER

; **************** VERIFICA ALGUM BOTÃO PRESSIONADO **************************

VARRE_BOTOES
    BTFSS   BOTAO_0         ; O BOTÃO 0 ESTÁ PRESSIONADO ?
    BRA     TRATA_BOTAO_0   ; SIM - PULA P/ TRATA_BOTAO_0
                            ; NÃO
    BTFSS   BOTAO_1         ; O BOTÃO 1 ESTÁ PRESSIONADO ?
    BRA     TRATA_BOTAO_1   ; SIM - PULA P/ TRATA_BOTAO_1
                            ; NÃO
    BTFSS   BOTAO_2         ; O BOTÃO 2 ESTÁ PRESSIONADO ?
    BRA     TRATA_BOTAO_2   ; SIM - PULA P/ TRATA_BOTAO_2
                            ; NÃO
    BTFSS   BOTAO_3         ; O BOTÃO 3 ESTÁ PRESSIONADO ?
    BRA     TRATA_BOTAO_3   ; SIM - PULA P/ TRATA_BOTAO_3
                            ; NÃO

; ***************************** FILTRO P/ EVITAR RUIDOS ************************

    MOVLW   FILTRO_TECLA    ; CARREGA NO WORK O VALOR DE FILTRO_TECLA
    MOVWF   FILTRO_BOTOES   ; SALVA EM FILTRO_BOTOES
```

```
                                ; RECARREGA FILTRO P/ EVITAR RUIDOS
                                ; NOS BOTÕES
    MOVLW    .1                 ; CARREGA TEMPO DO TURBO DAS TECLAS
    MOVWF    TEMPO_TURBO        ; COM 1 - IGNORA O TURBO A PRIMEIRA
                                ; VEZ QUE A TECLA É PRESSIONADA

    BRA      VARRE              ; VOLTA PARA VARRER TECLADO

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                            TRATAMENTO DOS BOTÕES                          *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; NESTE TRECHO DO PROGRAMA ESTÃO TODOS OS TRATAMENTOS DOS BOTÕES

; ************************** TRATAMENTO DO BOTÃO 0 ****************************

TRATA_BOTAO_0
    DECFSZ   FILTRO_BOTOES,F ; FIM DO FILTRO ? (RUIDO?)
    BRA      VARRE           ; NÃO - VOLTA P/ VARRE
                             ; SIM - BOTÃO PRESSIONADO
    DECFSZ   TEMPO_TURBO,F   ; FIM DO TEMPO DE TURBO ?
    BRA      VARRE           ; NÃO - VOLTA P/ VARRE
                             ; SIM
    MOVLW    TURBO_TECLA
    MOVWF    TEMPO_TURBO     ; RECARREGA TEMPORIZADOR DO TURBO
                             ; DAS TECLAS

    INCF     VALOR_DADOS,F   ; INCREMENTA VALOR_DADOS

    RCALL    ATUALIZA_TELA_LINHA_2   ; CHAMA ROTINA P/ ATUALIZAR LCD

    BRA      VARRE           ; VOLTA P/ VARREDURA DOS BOTÕES

; ************************** TRATAMENTO DO BOTÃO 1 ****************************

TRATA_BOTAO_1
    DECFSZ   FILTRO_BOTOES,F ; FIM DO FILTRO ? (RUIDO?)
    BRA      VARRE           ; NÃO - VOLTA P/ VARRE
                             ; SIM - BOTÃO PRESSIONADO
    DECFSZ   TEMPO_TURBO,F   ; FIM DO TEMPO DE TURBO ?
    BRA      VARRE           ; NÃO - VOLTA P/ VARRE
                             ; SIM
    MOVLW    TURBO_TECLA
    MOVWF    TEMPO_TURBO     ; RECARREGA TEMPORIZADOR DO TURBO
                             ; DAS TECLAS

    DECF     VALOR_DADOS,F   ; DECREMENTA VALOR_DADOS

    RCALL    ATUALIZA_TELA_LINHA_2   ; CHAMA ROTINA P/ ATUALIZAR LCD

    BRA      VARRE           ; VOLTA P/ VARREDURA DOS BOTÕES

; ************************** TRATAMENTO DO BOTÃO 2 ****************************

TRATA_BOTAO_2
    MOVF     FILTRO_BOTOES,F ; FILTRO JÁ IGUAL A ZERO?(FUNÇÃO JÁ FOI EXECUTADA?)
    BZ       VARRE           ; SIM - VOLTA P/ VARREDURA DO TECLADO
                             ; NÃO
    DECFSZ   FILTRO_BOTOES,F ; FIM DO FILTRO ? (RUIDO?)
    BRA      VARRE           ; NÃO - VOLTA P/ VARRE
                             ; SIM - BOTÃO PRESSIONADO

; ********TRECHO DO PROGRAMA PARA GRAVAR DADO DA RAM NA MEMÓRIA *************

GRAVA_MEMORIA_EEPROM
    MOVLW    END_EEPROM_H
```

```
    MOVWF   ENDERECO_HIGH    ; CARREGA ENDERECO ONDE O DADO SERÁ SALVO
    MOVLW   END_EEPROM_L     ; END. -> 0x0000
    MOVWF   ENDERECO_LOW     ; PRIMEIRA POSIÇÃO DA EEPROM

    MOVFF   VALOR_DADOS,BUFFER  ; CARREGA DADO A SER SALVO EM BUFFER

    RCALL   I2C_EEPROM_WRITE; CHAMA ROTINA DE GRAVAÇÃO

    MOVLW   .10
    RCALL   DELAY_MS         ; GARANTE TEMPO DE ESCRITA (10ms)

    BRA     VARRE            ; VOLTA P/ VARREDURA DOS BOTÕES

; ************************** TRATAMENTO DO BOTÃO 3 ****************************

TRATA_BOTAO_3
    MOVF    FILTRO_BOTOES,F ; FILTRO JÁ IGUAL A ZERO?(FUNÇÃO JÁ FOI EXECUTADA?)
    BZ      VARRE            ; SIM - VOLTA P/ VARREDURA DO TECLADO
                             ; NÃO
    DECFSZ  FILTRO_BOTOES,F ; FIM DO FILTRO ? (RUIDO?)
    BRA     VARRE            ; NÃO - VOLTA P/ VARRE
                             ; SIM - BOTÃO PRESSIONADO

; ******** TRECHO DO PROGRAMA PARA LER DADO DA MEMÓRIA E ATUALIZAR RAM ***********

LER_MEMORIA_EEPROM
    MOVLW   END_EEPROM_H
    MOVWF   ENDERECO_HIGH    ; CARREGA ENDEREÇO DE LEITURA
    MOVLW   END_EEPROM_L     ; END. -> 0x0000
    MOVWF   ENDERECO_LOW     ; PRIMEIRA POSIÇÃO DA EEPROM

    RCALL   I2C_EEPROM_READ ; CHAMA ROTINA DE LEITURA

    BTFSC   F_ERRO
    BRA     $+.6
    MOVFF   BUFFER,VALOR_DADOS       ; ATUALIZA RAM COM O VALOR LIDO

    RCALL   ATUALIZA_TELA_LINHA_2    ; CHAMA ROTINA P/ ATUALIZAR LCD

    BRA     VARRE            ; VOLTA P/ VARREDURA DOS BOTÕES

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                              FIM DO PROGRAMA                              *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

    END                      ; FIM DO PROGRAMA
```

» Dicas e comentários

As constantes END_EEPROM_H e END_EEPROM_L representam a posição a ser gravada/lida da memória externa.

Esse programa não utiliza as interrupções e possui uma rotina (AGUARDA_I2C_LIVRE) para saber se o sistema está liberado para a próxima ação. Ele também testa o ACK e gera uma mensagem de erro, caso algo saia fora do padrão.

Exercícios propostos

Agora que o exemplo foi estudado e esclarecido, aproveite para gerar novos problemas e soluções, seguindo os exercícios propostos:

1. Fazer três modificações no primeiro exercício proposto no Capítulo 10.
   - ✓ Utilizar a memória externa;
   - ✓ Limitar os dados mostrados no display entre 0x41 e 0x5A;
   - ✓ Mostrar os dados em ASCII, ou seja, entre A (0x41) e Z (0x5A).
2. Utilizando o exercício anterior, gravar na memória uma mensagem de até 16 caracteres. Depois, criar um programa que, ao ser inicializado, leia os 16 caracteres da memória e mostre a mensagem lida no LCD.

Editora Érica - Desbravando o Microcontrolador PIC18 Recursos Avançados - Daniel R. de Sousa, David José de Souza e Nicolás César Lavinia - 1ª Edição

12

Comunicação Serial 2 - USART

» Introdução

Esta é a segunda parte sobre comunicação serial e vamos estudar agora outro módulo interno do PIC destinado a esse recurso. Trata-se da USART. A grande vantagem da separação dos recursos de comunicação em dois módulos é que eles são independentes, podendo ser utilizados simultaneamente.

O nome USART significa Universal Synchronous Asynchronous Receiver Transmitter. Com um nome desse, o negócio parece um tanto complicado, não é mesmo? Mas não é. Acontece que este é um protocolo universal e possui dois modos distintos de trabalho: o sincronizado e o não sincronizado. Mas vamos logo ao que interessa e tratemos de conhecer melhor esse recurso.

» Teoria

Como estávamos dizendo, a USART, que também é conhecida como SCI (Serial Communications Interface), possui dois modos de funcionamento. Vejamos as características de cada um deles a seguir.

Modo assíncrono

A comunicação é feita somente com duas vias, entretanto, como esse modo não é sincronizado, essas duas vias são utilizadas para dados. Uma delas para transmissão (TX) e a outra para recepção (RX). Isso possibilita que as informações sejam enviadas e recebidas ao mesmo tempo, cada qual na sua via. Esse recurso é conhecido como Full Duplex. Esse modo é o utilizado, por exemplo, na porta serial dos computadores, para implementar o padrão RS-232, mas pode ser utilizado para acesso a outros sistemas também.

Mas como é possível os dados serem transmitidos entre dois pontos se não há sincronismo entre eles? Quando estudamos a comunicação SSP, vimos que uma via era perdida exatamente para essa função. Era definida como clock do sistema e servia para informar os dois lados (Master e Slave) do momento correto de transmissão de cada bit. Como aqui não há essa via, a sincronização deve ser feita pela própria via de dados. Isso é conseguido pelo Baud Rate ou velocidade de transmissão. Entenda como funciona.

Comecemos definindo exatamente o que é Baud Rate. Para que o sistema funcione, o tamanho dos dados (intervalo de cada bit) deve ser completamente padronizado, e ambos os lados devem estar ajustados para o mesmo valor. Como essa comunicação trabalha sempre com base nos bits, essa velocidade é

Editora Érica - Desbravando o Microcontrolador PIC18 Recursos Avançados - Daniel R. de Sousa, David José de Souza e Nicolás César Lavinia - 1ª Edição

normalmente indicada em bits por segundo, ou bps. Com ela somos capazes de calcular o tempo de duração de cada bit.

$$T_{BIT} = \frac{1}{\text{Baud Rate}}$$

Com isso, existe somente um sincronismo de tempo feito para a transmissão/recepção de cada byte. Esse sincronismo é conseguido através do Start bit.

Devemos entender também que ambas as vias devem ser tratadas igualmente, pois o TX de um lado deve estar conectado ao RX do outro e vice-versa. Em ambos os lados, TX é sempre saída e RX é sempre entrada. Desta forma, quando falarmos de transmissão ou recepção, serve para qualquer uma das vias.

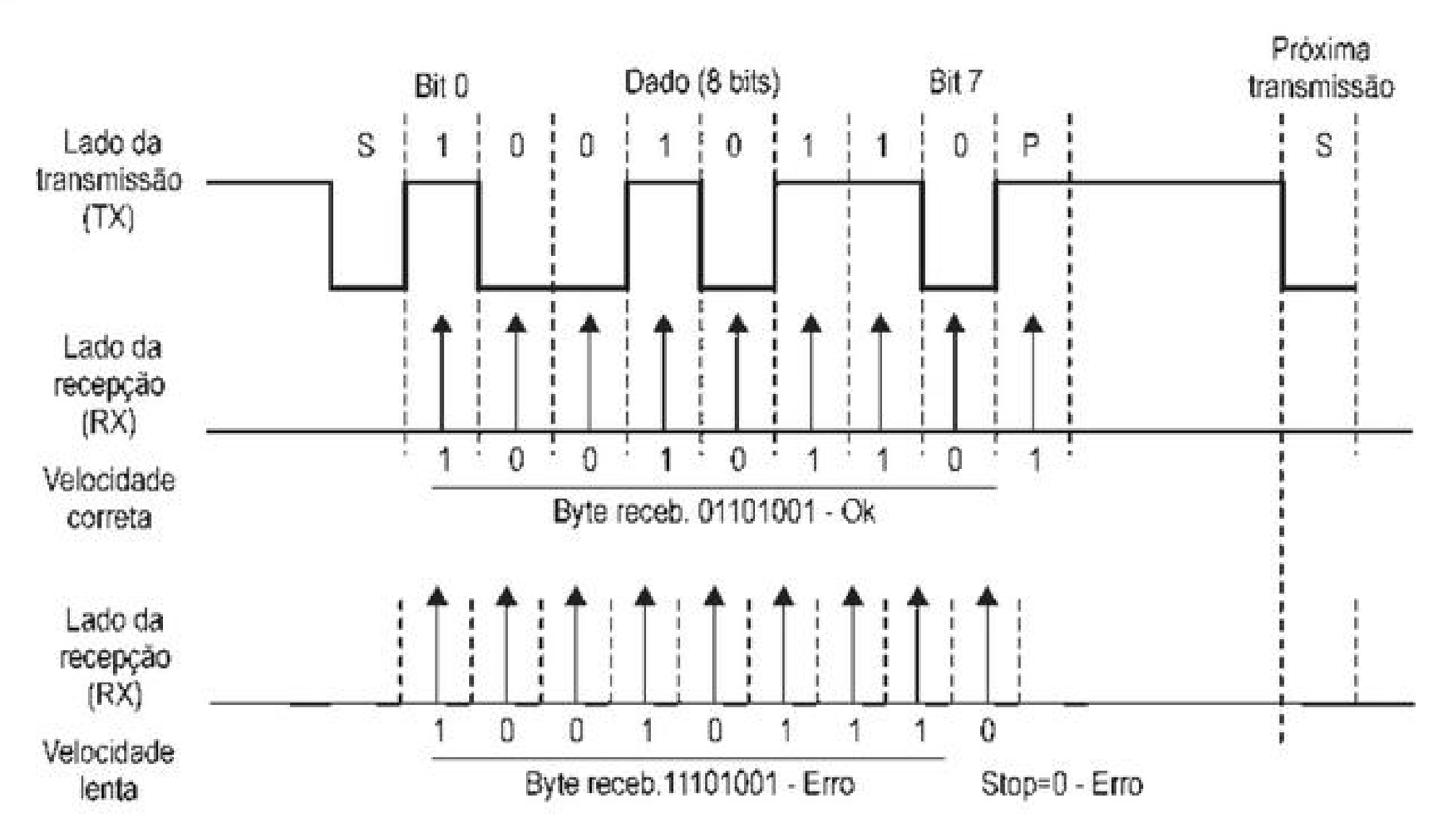
As vias possuem seu estado padrão como sendo o nível alto. Temos, então, uma situação de stand-by. Quando um lado inicia uma transmissão, ele força seu TX para nível baixo, mantendo-o assim pelo tempo T_{BIT}. Essa borda de descida é reconhecida pelo outro lado (em RX) e é suficiente para iniciar o processo de sincronização para recebimento desse byte. Esse pulso em nível baixo é chamado de Start Bit.

Depois disso, os dois lados já sabem o que fazer. TX envia então os 8 bits de dados, todos eles com o mesmo tamanho do Start Bit. Como RX soube exatamente o momento de início do Start Bit, ele deixa passar o tempo e depois coleta os 8 bits, pegando o dado mais ou menos no meio do tempo do bit (T_{BIT} / 2).

Por último, para garantir o processo, TX envia um Stop Bit, que nada mais é que outro bit com valor fixo em 1, garantindo assim que a linha voltará ao seu estado padrão e o sistema ao stand-by, ficando apto ao próximo dado. O lado RX deve considerar a leitura do Stop Bit para garantir que nenhum erro grosseiro aconteça com a recepção. Caso o Stop Bit seja 0 (zero), pode ter acontecido um erro de temporização, e no seu lugar foi lido o bit 8 ou, então, o próximo Start.

Repare também que o erro desse processo é acumulativo. Por exemplo, caso o lado TX esteja com sua velocidade no limite superior do erro, e o RX com a velocidade menor do que devia, cada bit será lido mais perto do começo do pulso. Como existem 10 bits ao total (Start + Dado + Stop), é possível que, no final, aconteça a leitura errada de um bit.

Vejamos o processo graficamente:



O primeiro caso de recepção está com a velocidade bem próxima à velocidade do transmissor, por isso não houve erro na recepção. Já no outro caso a velocidade está mais lenta, e houve erro no último bit e no Stop Bit. Por isso, é importante o acerto e a precisão da velocidade em ambos os lados. Quanto maior o Baud Rate, mais crítica é a situação.

Observe também que a ordem de transmissão dos bits é a inversa das comunicações SSP. Aqui, o bit menos significativo é enviado primeiro.

Esse padrão aceita também a comunicação com 9 bits, sendo que o bit adicional poder ser utilizado para dado, paridade ou endereçamento. A paridade nada mais é que uma confirmação matemática dos 8 bits de dados. Somando-se a quantidade de bits em 1 (incluindo dados e paridade), o resultado correto deve ser um número par (quando utilizando paridade PAR) ou ímpar (quando utilizando paridade ÍMPAR). Quanto ao endereçamento, respeita-se o seguinte critério: 0 para dado e 1 para endereço.

O importante é que ambos os lados (TX e RX) estejam configurados para operar com a mesma quantidade de bits. A configuração mais comum é a de 8 bits de dado (sem paridade) com 1 bit de Stop e é normalmente chamada de padrão 8N1.

No caso de comunicações padronizadas, o Baud Rate (BR) também obedece a valores pré-ajustados, tais como 300, 1.200, 2.400, 9.600, 19.200 bps e muitos outros.

Modo síncrono

Esse modo pode ser considerado como uma certa mistura entre os padrões SPI e I^2C.

Assim como no modo assíncrono, aqui também trabalhamos com somente duas vias, só que neste caso uma é destinada ao clock (CK) e a outra aos dados (DT). Desta forma, os dados devem trafegar em uma única via, impossibilitando a transmissão e recepção simultâneas. É o mesmo conceito utilizado no padrão I^2C. Essa comunicação é chamada de Half Duplex e pode ser utilizada para a troca de dados com outros microcontroladores ou diversos periféricos existentes no mercado, tais como ADs, DAs, memórias etc.

Quanto à forma como a informação trafega na linha, é bem mais simples que o padrão I^2C, não possuindo o sistema de endereçamento e parecendo-se mais com o formato SPI. Para cada pulso (borda de descida) é transmitido um bit.

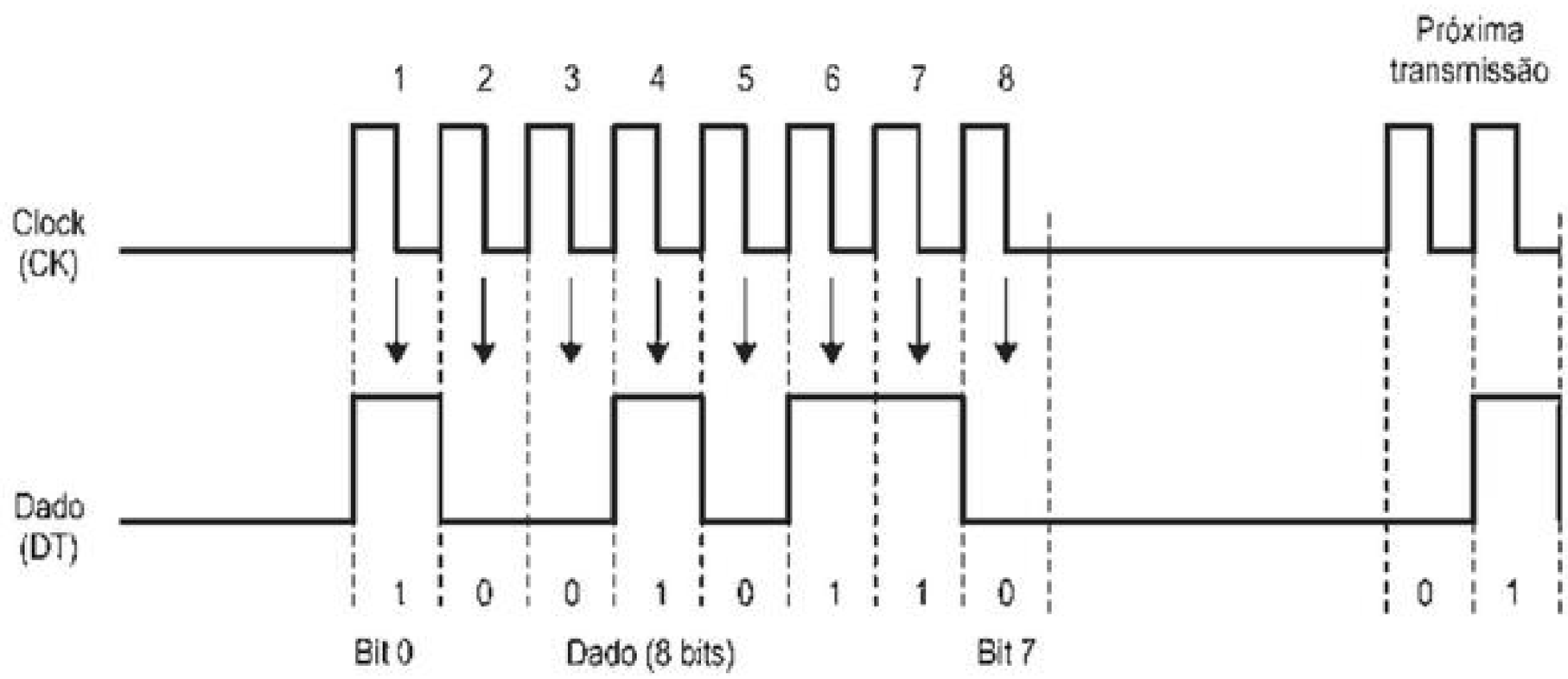
Esse modo também opera com Mestre e Escravo, sendo o clock sempre gerenciado pelo Mestre. Para o Mestre, a via CK é sempre uma saída e para o Escravo ela é sempre uma entrada. Quanto à via de dados, ela muda constantemente de sentido, ora para a transmissão, ora para a recepção. Assim sendo, para qualquer uma das pontas, a via DT é saída para a transmissão e entrada para recepção.

O tempo de duração de um bit também define o Baud Rate, do mesmo modo descrito na comunicação assíncrona.

Neste caso, o nível baixo (0) é o padrão para a via CK. Para a via DT, não existe um padrão obrigatório, pois ela não opera sem CK. Porém, recomendamos mantê-la também em nível baixo quando não está sendo usada.

Observe que, nesse padrão, também o bit menos significativo (bit 0) é enviado primeiro.

Para o modo síncrono também é aceita a comunicação com 9 bits e, neste caso, o Mestre sempre gera pacotes de nove pulsos de clock. O importante é que ambos os lados da comunicação (Mestre e Escravo) estejam configurados para operar com 9 bits.



» Recursos do PIC

Vamos estudar agora como operar com a USART do PIC18F4520. Para continuarmos com a mesma linha de raciocínio, dividiremos mais uma vez parte do assunto entre os dois modos de operação, mas antes podemos explicar a maioria dos conceitos que são compartilhados por ambos.

Os pinos da comunicação (**RC6/TX/CK**) e (**RC7/RX/DT**) são os mesmos nos dois modos e controlados diretamente pelo sistema da USART. Somente para evitarmos possíveis conflitos, caso a USART seja desabilitada, é recomendável que ambos sejam ajustados como entrada através do **TRISC**.

A definição entre os modos de operação é feita através de **TXSTA<SYNC>**:

SYNC	Descrição
0	Modo assíncrono
1	Modo síncrono

O ajuste do Baud Rate (BR) é feito por meio do registrador denominado **SPBRG**, do bit **TXSTA<BRGH>** e do bit **BAUDCON<BRG16>**. A combinação desses parâmetros e do modo de operação define o cálculo do BR:

BRGH	Descrição
0	Ajuste para baixa velocidade
1	Ajuste para alta velocidade

Para **BAUDCON<BRG16> = 0**:

SYNC	BRGH=0	BRGH=1
0	BR = F_{osc}/(64x(**SPBRG**+1))	BR = F_{osc}/(16x(**SPBRG**+1))
1	BR = F_{osc}/(4x(**SPBRG**+1))	Não válido

Para **BAUDCON<BRG16> = 1**:

SYNC	BRGH=0	BRGH=1
0	BR = F_{osc}/(16x(**SPBRG**+1))	BR = F_{osc}/(4x(**SPBRG**+1))
1	BR = F_{osc}/(4x(**SPBRG**+1))	Não válido

Note que **SPBRG** é composto de dois registradores: **SPBRG** (parte baixa) e **SPBRGH** (parte alta).

Primeiramente observe que o ajuste de **BRGH** não tem efeito quando estamos trabalhando no modo síncrono. Neste caso, o tempo de cada bit (T_{BIT}) será múltiplo do tempo de ciclo de máquina (T_{CY}). Para o menor ajuste possível (**SPBRG=0**), teremos $T_{BIT}=T_{CY}$. Nesta situação, a parte alta do clock é gerada nos subtempos Q4 e Q1, ficando Q2 e Q3 para a parte baixa.

Por exemplo, para uma F_{OSC} de 4 MHz e **BRG16=0**, teremos:

- ✓ $BR_{MÁX}$ = 1.000.000 bps
- ✓ $BR_{MÍN}$ = 3.906 bps

Para o modo síncrono, **BRGH** surte efeito alterando a fórmula para o cálculo de BR. Observe que, quando **BRGH=0**, T_{BIT} será múltiplo de 64 T_{CY}. Isso diminui a velocidade máxima e aumenta o erro para nos aproximarmos das velocidades padronizadas. Para o segundo caso (**BRGH=1**), temos um ajuste mais preciso, pois T_{BIT} será múltiplo de 16 T_{CY}. Agora temos um erro menor e aumento da velocidade máxima. Por outro lado, perdemos na velocidade mínima.

Peguemos o mesmo exemplo dado para o caso do modo síncrono (F_{OSC} = 4 MHz), primeiro para **BRGH=0**:

- ✓ $BR_{MÁX}$ = 62.500 bps
- ✓ $BR_{MÍN}$ = 244 bps

E agora para **BRGH=1**:

- ✓ $BR_{MÁX}$ = 250.000 bps
- ✓ $BR_{MÍN}$ = 976 bps

Vamos agora calcular o valor de **SPBRG** para uma velocidade de 9.600 bps, com cristal de 4 MHz, **BRGH=0** e **BRG16=0**:

$$BR = F_{OSC}/(64 \times (SPBRG + 1))$$

$$SPBRG = (F_{OSC}/(64 \times BR)) - 1$$

$$SPBRG = (4.000.000 / (64 \times 9.600)) - 1$$

$$SPBRG = 5{,}510 \rightarrow \mathbf{5}$$

Como houve um arredondamento, existe um erro que deve ser calculado:

$$BR = 4.000.000 / (64 \times (5 + 1))$$

$$BR = 10.416 \text{ bps}$$

$$Erro = (10.416 - 9.600) / 9.600$$

$$Erro = \mathbf{8{,}5\%} \text{ (muito elevado)}$$

Vejamos o mesmo caso para **BRGH=1**:

$$BR = F_{osc}/(16x(SPBRG + 1))$$

$$SPBRG = (F_{osc}/(16 \times BR)) - 1$$

$$SPBRG = (4.000.000 / (16 \times 9600)) - 1$$

$$SPBRG = 25{,}042 \rightarrow \mathbf{25}$$

$$BR = 4.000.000 / (16 \times (25 + 1))$$

$$BR = 9.615 \text{ bps}$$

$$Erro = (9.615 - 9.600) / 9.600$$

Erro = **0,16%** (muito bom!)

Devido à grande diferença de erro, devemos adotar o segundo cálculo. Existem casos, principalmente quando BR é elevado, que para melhorarmos a situação é necessário alterar o valor do oscilador (F_{osc}), tentando chegar a números mais precisos. Não esqueça que existe ainda a tolerância de funcionamento do sistema de oscilação. O uso de cristais pode também ser obrigatório para as velocidades mais altas.

Para ativar o sistema da USART, configurando os pinos corretamente, deve-se ajustar o bit **RCSTA<SPEN>**:

SPEN	Descrição
0	USART desabilitada. Pinos como I/Os convencionais.
1	USART habilitada. Pinos controlados automaticamente.

A partir deste ponto o usuário deve escolher entre as operações de transmissão e/ou recepção, mas como elas são bem diferentes para os modos assíncronos e síncronos, serão explicadas na divisão dos tópicos. Por enquanto, vejamos somente quais são e os bits que as controlam:

Operação	Bit	Observações
Transmissão	**TXSTA<TXEN>**	Ativa o sistema de transmissão de bytes.
Recepção unitária	**RCSTA<SREN>**	Ativa o sistema de recepção de somente 1 byte. Somente para modo síncrono.
Recepção contínua	**RCSTA<CREN>**	Ativa o sistema de recepção contínua.

O que devemos explicar, que ainda é comum aos modos de operação, diz respeito ao dado que será transmitido e/ou recebido.

Primeiramente devemos optar pela comunicação com 8 ou 9 bits. Isso é feito separadamente para a transmissão e para a recepção, através dos bits **TXSTA<TX9>** e **RCSTA<RX9>**:

TX9	Descrição
0	Transmissão feita em 8 bits.
1	Transmissão feita em 9 bits.

RX9	Descrição
0	Recepção feita em 8 bits.
1	Recepção feita em 9 bits.

Esse nono bit deve ser escrito em **TXSTA<TX9D>** para a transmissão e recebido em **RCSTA<RX9D>** do lado da recepção. O PIC não possui sistema automático para implementação desse bit como sendo a paridade. Se o uso da paridade for necessário, essa implementação deve ser feita através do software. Existe um auxílio para uso desse bit como endereçamento, mas isso será visto posteriormente.

Quanto ao dado propriamente dito, para o caso da transmissão, ele deve ser escrito no registrador **TXREG**. Caso o sistema de transmissão esteja ligado, a simples escrita nesse registrador vai iniciá-la. Esse dado será, então, transferido a um registrador interno denominado **TSR** (sem acesso pelo programa) para que seja enviado pela porta serial. Neste momento, **TXREG** fica vazio e liberado para uma nova escrita. Desta forma, o buffer de saída é duplo, podendo haver um byte em **TXREG** e outro em **TSR**. O segundo byte só será passado de **TXREG** para **TSR** quando o último bit (7 ou 8) do byte anterior for transmitido, deixando **TSR** vazio.

Toda vez que o valor for passado de **TXREG** para **TSR**, o bit **PIR1<TXIF>** será setado, podendo gerar a interrupção. O interessante é que esse bit não precisa ser limpo manualmente como os demais flags de interrupção. Ele é limpo automaticamente sempre que **TXREG** for escrito. Isso serve para sabermos quando **TXREG** está liberado para uma nova escrita. O problema é que, se não desejamos transmitir nada, não escreveremos em **TXREG** e não limparemos **TXIF**. Isso trava o programa, entrando sem parar na interrupção. A solução, para este caso, é desligar a interrupção ou mesmo a transmissão.

O bit **TXIF** não pode ser utilizado para sabermos que uma transmissão já foi completada, pois, depois de **TXREG** ser colocado em **TSR**, o dado ainda precisa ser enviado pela porta serial. Para saber quando realmente a transmissão foi finalizada, devemos saber quando **TSR** está vazio. O bit **TXSTA<TRMT>** possui essa função:

TRMT	Descrição
1	Transmissão finalizada. TSR liberado.
0	Transmissão em curso. TSR ocupado.

Para a recepção, o dado recebido é primeiramente armazenado em um registrador interno não acessível (**RSR**). Quando esse registrador está completo, a informação é então passada para o registrador **RCREG**. O mais interessante é que esse registrador possui dois níveis de pilha, podendo ser escrito e lido duas vezes. Com isso o buffer de entrada é triplo, podendo haver um dado sendo recebido em **RSR** e mais dois já recebidos em **RCREG**. O registrador **RCREG** trabalha com o sistema FIFO (First In / First Out), isto é, o primeiro a ser recebido é o primeiro a ser lido.

Cada vez que o valor de **RSR** é transferido para **RCREG** o bit **PIR1<RCIF>** é setado, podendo gerar a interrupção. Como no caso da transmissão, esse bit é limpo automaticamente pelo hardware sempre que **RCREG** estiver vazio. Isso significa que, se foram recebidos 2 bytes antes de ser efetuada uma conferência, **RCREG** terá de ser lido duas vezes (pilha) para que o bit **RCIF** seja limpo. Caso o terceiro byte seja completado antes da leitura das duas já existentes em **RCREG**, um erro de overflow acontece e o bit **RCSTA<OERR>** é setado. O terceiro byte que se encontra em **RSR** será perdido. Esse bit deve ser limpo manualmente. Isso também pode ser conseguido desativando-se o modo de recepção.

Para trabalhar com o nono bit, é necessário que ele seja escrito (**TX9D**) antes do dado em **TXREG**, possibilitando a atualização correta do **TSR**.

O sistema de recepção verifica também o Stop Bit, mas não toma nenhuma atitude automática em caso de erro. A situação do Stop Bit é armazenada em **RCSTA<FERR>**:

FERR	Descrição
0	O Stop Bit foi recebido corretamente (Stop Bit = 1).
1	O Stop Bit foi recebido errado (Stop Bit = 0).

Para que esse bit possa ser verificado pelo programa, assim como o nono (quando usado), uma leitura deve ser feita antes em **RCREG** para que eles sejam atualizados.

Por último devemos comentar sobre o sistema de endereçamento, que só funciona quando ajustada a comunicação para 9 bits, o que é controlado através do bit **RCSTA<ADDEN>**:

ADDEN	Descrição
0	Sistema de endereçamento desativado.
1	Sistema de endereçamento ativado.

Quando esse sistema está ativo, o dado recebido só é transferido de **RSR** para **RCREG** quando o nono for 1, podendo então gerar a interrupção. Caso seja recebida uma informação com o nono em 0, ela é reconhecida como dado e descartada. Para que o processo funcione corretamente é necessário, então, começarmos com o endereçamento ativo para recebermos um endereço que será tratado e comparado como o endereço da própria unidade. O endereço será o byte recebido, possibilitando 256 valores diferentes. Caso o endereço recebido corresponda à unidade em questão, o sistema de endereçamento deve ser desativado para que o próximo valor possa ser recebido como um dado válido.

Vejamos agora as particularidades e observações do roteiro de trabalho de cada modo.

Modo assíncrono

Para o modo assíncrono, quando habilitamos a USART através do bit **RCSTA<SPEN>**, o pino **TX** é transformado em saída com nível alto (1) e o pino **RX** é transformado em entrada.

Para ativar o sistema de recepção, basta tornar **RCSTA<CREN>=1**. Para esse modo não é possível escolher o sistema de recepção unitária e, por isso, **RCSTA<SREN>** não possui função.

A partir desse momento cada byte recebido é colocado em **RCREG** e a interrupção ativada através de **RCIF**. A lógica do sistema é, então, tratar das informações recebidas. É recomendável também a implementação da verificação do Stop Bit (**FERR**) e do estouro de recepção (**OERR**).

Para ativar o sistema de transmissão, basta tornar **TXSTA<TXEN>=1**. O importante é lembrar que os sistemas de transmissão e recepção são totalmente independentes (só compartilham o mesmo Baud Rate) e, por isso, cada um deles pode ser ativado e desativado, conforme a necessidade.

Depois da ativação, basta escrever um valor em **TXREG** e ele será enviado automaticamente pela porta serial, com a geração do Start Bit e do Stop Bit. O importante é a verificação do bit **TXIF** para que o sistema não sobre-escreva **TXREG** antes de **TSR** ser atualizado.

Quando um dos sistemas é desativado (transmissão ou recepção), o pino relacionado a ele é mantido como entrada.

Esse modo não pode operar em SLEEP.

Essa USART possui um recurso de auto wake-up, ou seja, acordar do modo SLEEP ao receber um dado. Setando o bit **BAUDCON<WUE>**, ao receber um novo dado o microcontrolador sai do modo SLEEP. Para que o microcontrolador não tenha erros ao receber o primeiro dado é recomendado que o primeiro caractere seja uma sequência de zeros (8 zeros no RS-232 padrão ou 12 zeros no barramento LIN).

Editora Érica - Desbravando o Microcontrolador PIC18 Recursos Avançados - Daniel R. de Sousa, David José de Souza e Nicolás César Lavinia - 1ª Edição

O bit **BAUDCON<RCIDL>** monitora a recepção de dados. Caso ocorra recebimento de dados, o bit **BAUDCON<RCIDL>** é setado.

O bit **BAUDCON<RXDTP>** configura a polaridade do dado recebido. O bit **BAUDCON<RXDTP>** deve estar setado se o dado recebido for ativo-baixo; caso contrário, o bit **BAUDCON<RXDTP>** deve estar limpo.

O bit **BAUDCON<TXCKP>** configura o estado normal do pino de transmissão. Se o bit **BAUDCON<TXCKP>** estiver setado, o estado normal do pino de transmissão será em nível 0. Caso o bit **BAUDCON<TXCKP>** esteja limpo, o estado normal do pino de transmissão será em nível 1.

Resumo dos registradores associados à USART assíncrona

Endereço	Nome	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
FF2h	INTCON	INTCON	GIE/ GIEH	PEIE/ GIEL	T0IE	INTE	RBIE	TMR0IF	INTF
F9Eh	PIR1	PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
F9Dh	PIE1	PSPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
F9Fh	IPR1	PSPIP	ADIP	RCIP	RXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP
FACh	TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D
FABh	RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
FB8h	BAUDCON	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	-	WUE	ABDEN
FADh	TXREG	Buffer de transmissão							
FAEh	RCREG	Buffer de recepção							
FAFh	SPBRG	Gerador de baudrate, parte baixa							
FB0h	SPBRGH	Gerador de baudrate, parte alta							
F94h	TRISC	Configuração do PORTC com Entrada(1) ou Saída(0)							

Não é usado para essa finalidade. Para obter mais informações, consulte o Apêndice A.

Modo síncrono

Para esse modo existe mais um bit de configuração relacionado ao sistema Master/Slave. Trata-se do **TXSTA<CSRC>**:

CSRC	Descrição
0	Slave (CK como entrada).
1	Master (CK como saída).

Com essa seleção, o sistema configura corretamente o estado do pino de clock (**CK**). O estado do pino de dados (**DT**) é variável, conforme o sistema se encontre em recepção ou transmissão. Devemos, então, escolher qual operação desejamos efetuar:

Operação	Bit	Observações
Transmissão	TXSTA<TXEN>	O pino DT é colocado como saída e o sistema está pronto para transmitir o dado escrito em TXREG. A escolha de um modo de recepção tem prioridade sobre este modo.
Recepção unitária	RCSTA<SREN>	O pino DT é colocado como entrada e o sistema de recepção de somente 1 byte é ativado.
Recepção contínua	RCSTA<CREN>	O pino DT é colocado como entrada e o sistema de recepção contínua é ativado.

Transmissão

No caso do Master, o sistema funciona de forma muito parecida com o descrito anteriormente para o modo assíncrono, operando da mesma forma com **TXREG**, **TXIF** e **TRMT**. A única diferença é que, quando escrevemos algum dado em **TXREG**, ele é transmitido por **DT**, com pulsos sendo gerados em **CK** e sem a presença de Start ou Stop bit. **CK** pode gerar oito ou nove pulsos, dependendo do estado de **TXSTA<TX9>**.

Ao término da transmissão, o sistema fica em stand-by, aguardando um novo dado em **TXREG**.

Para o Slave, a diferença é que, depois de escrevermos em **TXREG**, o sistema fica aguardando os clocks enviados pelo Master.

O bit **BAUDCON<TXCKP>** configura o estado normal do pino de transmissão. Se o bit **BAUDCON<TXCKP>** estiver setado, o estado normal do pino de transmissão será em nível 0. Caso o bit **BAUDCON<TXCKP>** esteja limpo, o estado normal do pino de transmissão será em nível 1.

Recepção

A recepção para o Master e para o Slave também é idêntica, exceto pela geração do clock. Quando ativada uma operação de recepção, a operação de transmissão é desligada imediatamente e o pino **DT** torna-se entrada.

A recepção contínua mantém a unidade recebendo dados até que o bit **RCSTA<CREN>** seja limpo manualmente. Já a recepção unitária (**RCSTA<SREN>=1**) recebe somente um byte e depois desliga-se (**RCSTA<SREN>=0**). Se por acaso forem ligadas as duas maneiras de recepção, o modo contínuo tem prioridade. No modo contínuo, o clock não é desativado, gerando pulsos constantes. Isso torna a operação nesse sistema mais difícil.

Ao receber um dado, ele é enviado a **RCREG** e o flag **PIR1<RCIF>** é ativado, podendo gerar a interrupção.

O bit **BAUDCON<RCIDL>** monitora a recepção de dados. Caso ocorra recebimento de dados, o bit **BAUDCON<RCIDL>** é setado.

O bit **BAUDCON<RXDTP>** configura a polaridade do dado recebido. O bit **BAUDCON<RXDTP>** deve estar setado se o dado recebido for ativo-baixo; caso contrário, o bit **BAUDCON<RXDTP>** deve estar limpo.

Operação em SLEEP

Somente as unidades Slaves do modo síncrono podem operar em SLEEP.

Quando um dado é recebido durante o SLEEP, ao final da recepção a informação é transportada de **RSR** para **RCREG**, ativando o flag de interrupção **PIR1<RCXIF>**. Deixando essa interrupção habilitada, o PIC é acordado e o dado recebido pode ser tratado.

Uma transmissão também é possível durante o SLEEP. O primeiro dado escrito em **TXREG** é imediatamente escrito em **TSR**. O segundo byte (se houver) é mantido em **TXREG**. O sistema é, então, colocado em SLEEP. Quando chegar o primeiro pacote de pulsos, o dado de **TSR** será transmitido. Ao final, o segundo dado é colocado em **TSR** e o flag **TXIF** é setado. Se a interrupção estiver habilitada, o PIC acorda.

Editora Érica - Desbravando o Microcontrolador PIC18 Recursos Avançados - Daniel R. de Sousa, David José de Souza e Nicolás César Lavinia - 1ª Edição

Resumo dos registradores associados à USART síncrona

Endereço	Nome	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
FF2h	INTCON	INTCON	GIE/ GIEH	PEIE/ GIEL	T0IE	INTE	RBIE	TMR0IF	INTF
F9Eh	PIR1	PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
F9Dh	PIE1	PSPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
F9Fh	IPR1	PSPIP	ADIP	RCIP	RXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP
FACh	TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D
FABh	RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
FB8h	BAUDCON	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	-	WUE	ABDEN
FADh	TXREG	Buffer de transmissão							
FAEh	RCREG	Buffer de recepção							
FAFh	SPBRG	Gerador de baudrate, parte baixa							
FB0h	SPBRGH	Gerador de baudrate, parte alta							
F94h	TRISC	Configuração do PORTC com Entrada(1) ou Saída(0)							

Não é usado para essa finalidade. Para obter mais informações, consulte o Apêndice A.

Funções adicionais na USART

O PIC18F4520, assim como outros microcontroladores, possui funções adicionais na USART para ser utilizado no barramento LIN, com a inclusão do envio de caractere de break e recepção do caractere de sincronismo (auto baudrate).

Um caractere de break consiste em um start bit, 12 zeros e um stop bit. Para o envio do caractere de break devemos setar o bits **TXSTA<SENDB>** e **TXSTA<TXEN>**. Ao setar **TXSTA<SENDB>**, o hardware de transmissão aguarda a escrita de um dado qualquer no **TXREG**, sendo o dado ignorado e o caractere de break enviado. Depois o bit **TXSTA<SENDB>** é automaticamente limpo pelo hardware de transmissão no final do envio do stop bit.

Um caractere de sincronismo consiste no envio de 55h (caractere **'U'**) após o envio do caractere de break. O hardware de recepção pode receber o caractere de sincronismo para o ajuste automático do baud rate. O bit **BAUDCON<ABDEN>**, quando setado, faz o hardware de recepção aguardar o caractere de sincronismo. Antes de receber o caractere de sincronismo é recomendado ler o registrador **RCREG**.

Ao receber esse caractere, há uma contagem de cinco bordas de subida, os registradores **SPBRG** e **SPBRGH** são atualizados com um novo valor e o bit **BAUDCON<ABDEN>** é limpo automaticamente. Note que durante a sequência de auto baudrate são utilizados os registradores **SPBRG** e **SPBRGH** independente da configuração do bit **BAUDCON<BRG16>**.

Caso ocorra um estouro no incremento dos registradores **SPBRG** e **SPBRGH** (valor maior que 65535), o bit **BAUDCON<ABDOVF>** é setado.

» Lógica do exemplo

Neste exemplo implementamos uma comunicação assíncrona Full Duplex, isto é, ativamos tanto a transmissão quanto a recepção.

Para tornar o sistema versátil e simples, vamos montar um programa capaz de operar o hardware proposto interligando-o ao PC por meio do conector DB9 (RS-232).

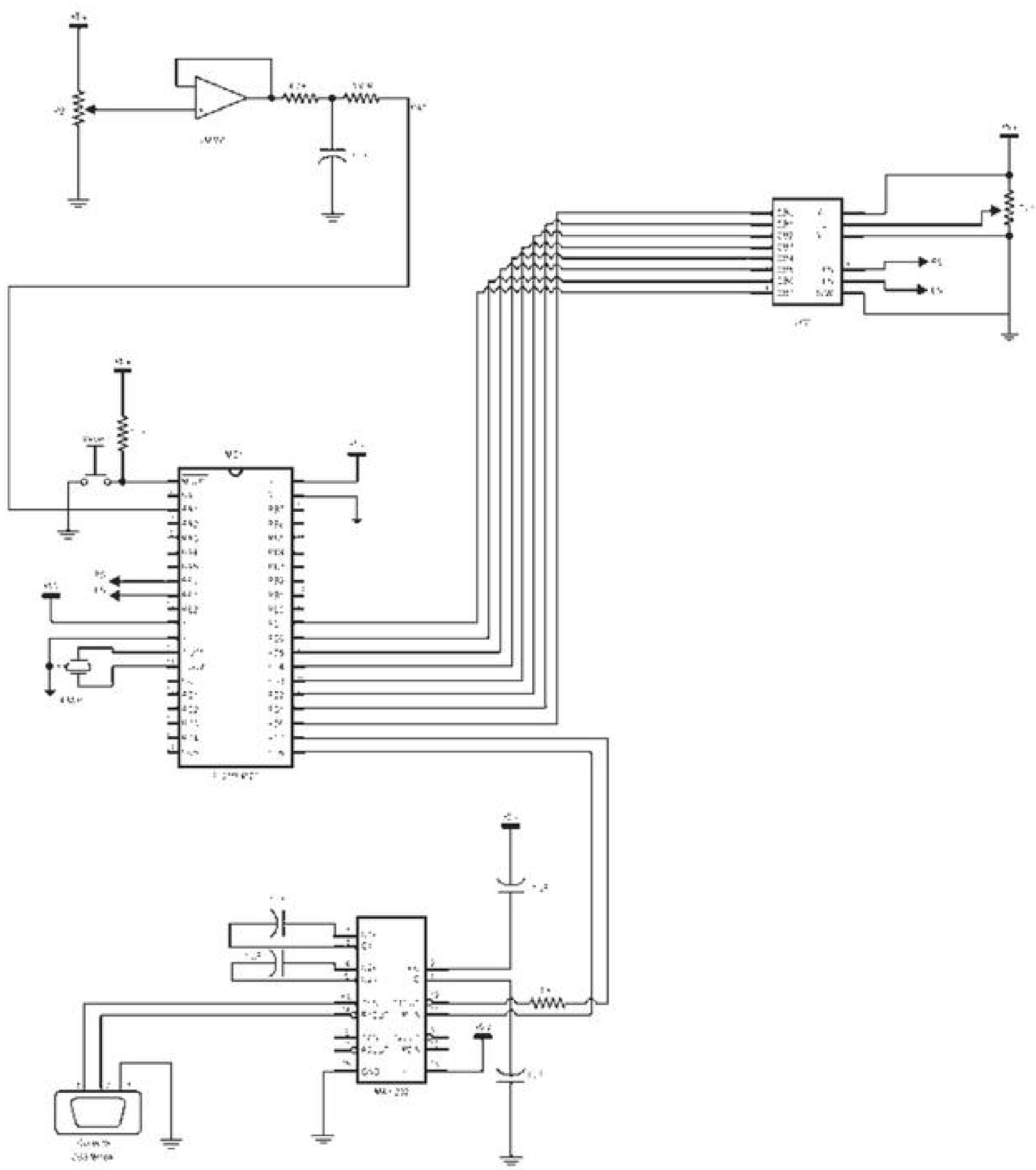
A transmissão é feita lendo-se a tensão do potenciômetro P2 através do AD, limitando os valores entre 0 e 255 (8 bits) e enviando esse resultado à porta serial e ao LCD. Desta forma, é possível visualizarmos o dado transmitido. Para facilitar ainda mais ao usuário, vamos mostrar o valor em decimal (d) e em hexadecimal (h). A transmissão é realizada no padrão 8N1 com uma velocidade de 9.600 bps.

Quanto à recepção, o valor obtido pela porta serial será diretamente impresso no display de LCD, através do código ASCII.

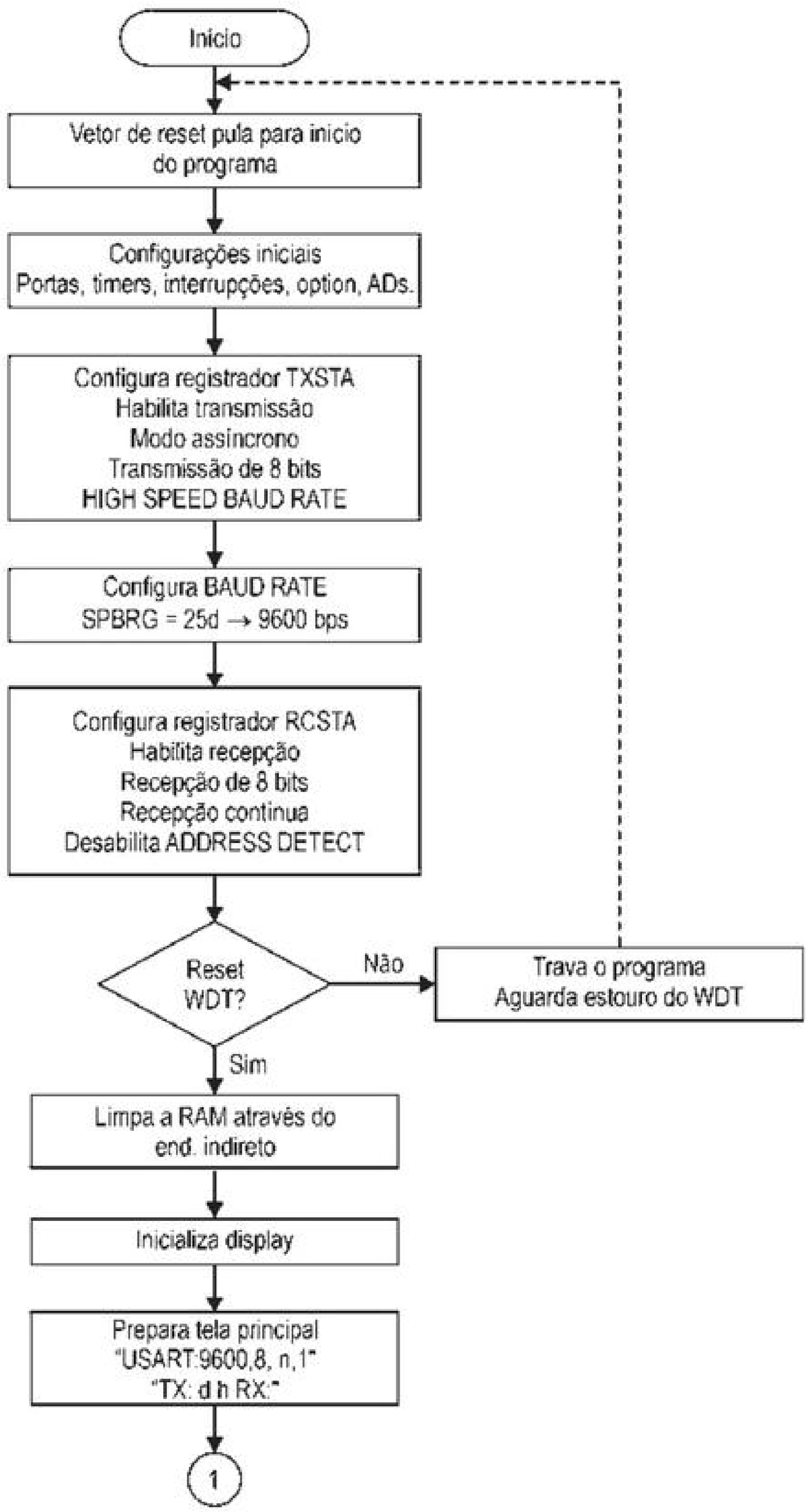
Para que o sistema funcione sem o PC, basta interligar os pinos 2 e 3 do conector DB9. Isso faz com que tudo que for transmitido por TX seja imediatamente recebido em RX. Tanto a transmissão quanto a recepção são contínuas.

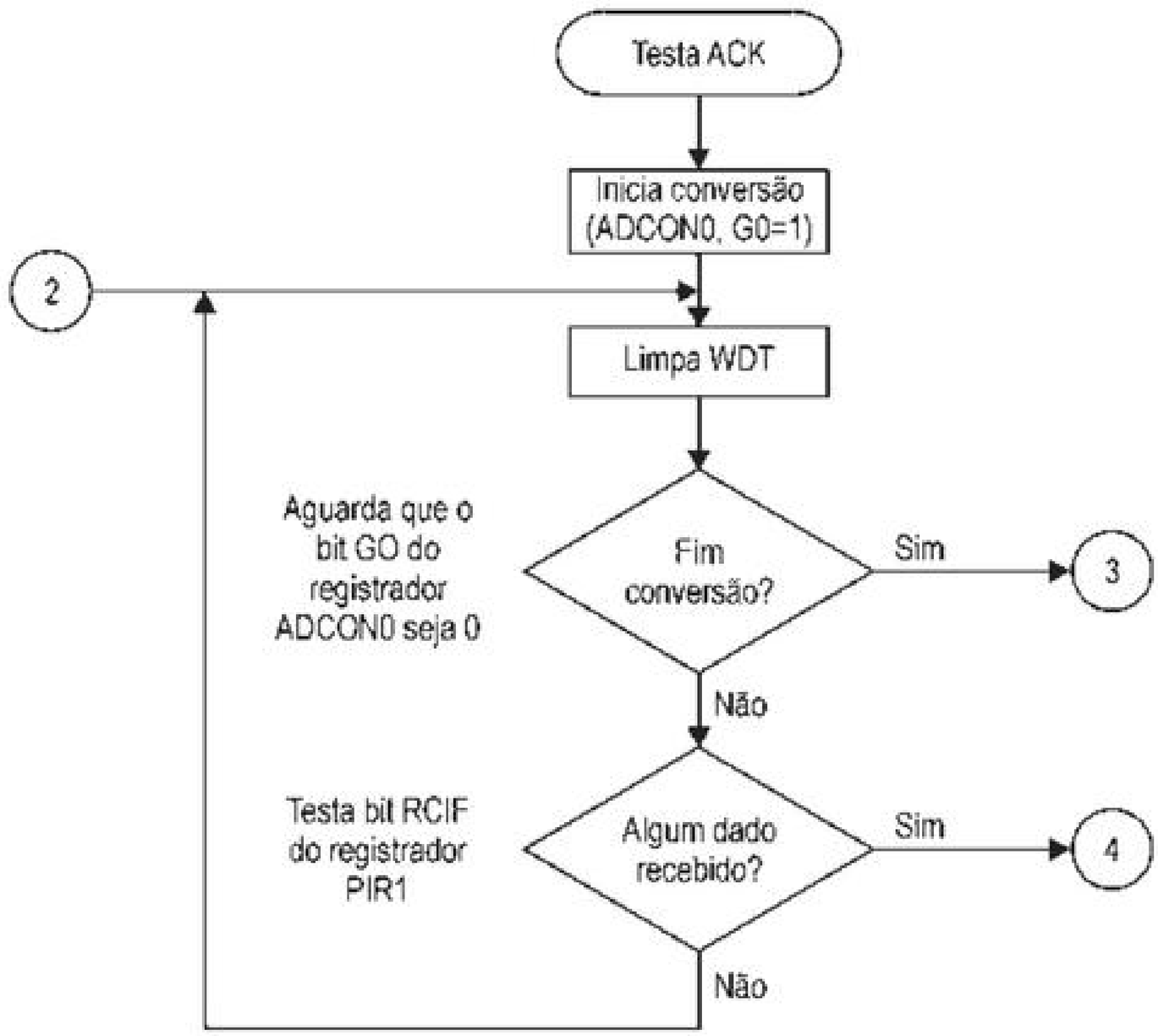
Como a comunicação está regulada para 9.600 bps e a placa opera em 4 MHz, o exemplo de cálculo para **SPBRG** já apresentado pode ser considerado.

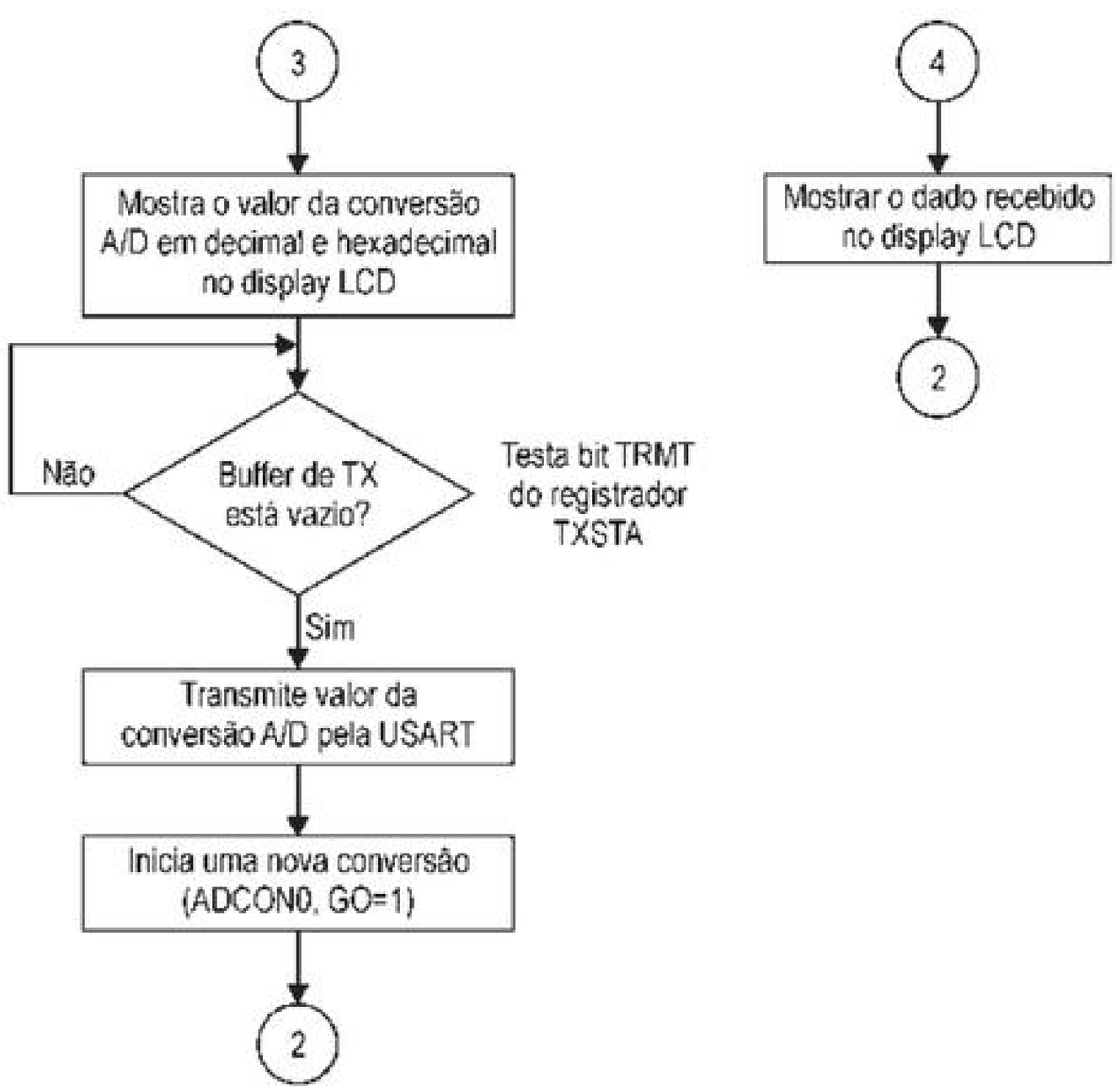
» Esquema elétrico



» Fluxograma


Início
Vetor de reset pula para inicio do programa
Configurações iniciais
Portas, timers, interrupções, option, ADs.
Configura registrador TXSTA
Habilita transmissão
Modo assincrono
Transmissão de 8 bits
HIGH SPEED BAUD RATE
Configura BAUD RATE
SPBRG = 25d → 9600 bps
Configura registrador RCSTA
Habilita recepção
Recepção de 8 bits
Recepção continua
Desabilita ADDRESS DETECT
Reset WDT?
Não
Trava o programa
Aguarda estouro do WDT
Sim
Limpa a RAM através do end. indireto
Inicializa display
Prepara tela principal
"USART:9600,8, n,1"
"TX: d h RX:"
1


Testa ACK
Inicia conversão (ADCON0, G0=1)
2
Limpa WDT
Aguarda que o bit GO do registrador ADCON0 seja 0
Fim conversão?
Sim
3
Não
Testa bit RCIF do registrador PIR1
Algum dado recebido?
Sim
4
Não


3
Mostra o valor da conversão A/D em decimal e hexadecimal no display LCD
Não
Buffer de TX está vazio?
Testa bit TRMT do registrador TXSTA
Sim
Transmite valor da conversão A/D pela USART
Inicia uma nova conversão (ADCON0, GO=1)
2
4
Mostrar o dado recebido no display LCD
2

» Código

```
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *              DESBRAVANDO O PIC18 - RECURSOS AVANÇADOS                   *
; *                              EXEMPLO 9                                  *
; *                                                                         *
; * DANIEL RODRIGUES DE SOUSA, DAVID JOSÉ DE SOUZA e  NICOLÁS CÉSAR LAVINIA *
; *                                                                         *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *     DATA : 03/08/2009                                                   *
; *   VERSÃO : 1.0                                                          *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                            DESCRIÇÃO GERAL                              *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTE EXEMPLO FOI ELABORADO PARA EXPLICAR O FUNCIONAMENTO DA USART DO PIC.
;  O SOFTWARE CONVERTE O CANAL 1 DO CONVERSOR AD (POTENCIÔMETRO P2) E MOSTRA
;  NO DISPLAY O VALOR CONVERTIDO EM DECIMAL E HEXADECIMAL.
;  ALÉM DE MOSTRAR O VALOR NO DISPLAY, O SOFTWARE TRANSMITE PELA USART O VALOR
;  DA CONVERSÃO. OS VALORES RECEBIDOS PELA USART TAMBÉM SÃO MOSTRADOS NO LCD
;  COMO CARACTERES ASCII.
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                        CONFIGURAÇÕES PARA GRAVAÇÃO                      *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
    CONFIG  OSC = XT, FCMEN = OFF, IESO = OFF, PWRT = ON, BOREN = ON, BORV = 0
    CONFIG  WDT = ON, WDTPS = 128, MCLRE = ON, LPT1OSC = OFF, PBADEN = OFF
    CONFIG  CCP2MX = PORTC, STVREN = ON, LVP = OFF, DEBUG = OFF, XINST = OFF
    CONFIG  CP0 = OFF, CP1 = OFF, CP2 = OFF, CP3 = OFF, CPB = OFF, CPD = OFF
    CONFIG  WRT0 = OFF, WRT1 = OFF, WRT2 = OFF, WRT3 = OFF, WRTB = OFF
    CONFIG  WRTC = OFF, WRTD = OFF, EBTR0 = OFF, EBTR1 = OFF, EBTR2 = OFF
    CONFIG  EBTR3 = OFF, EBTRB = OFF

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                          DEFINIÇÃO DAS VARIÁVEIS                        *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
    CBLOCK  0X00            ; BANK 0 --> 0X00..0X5F --> ACCESS BANK

        TEMPO1
        TEMPO0              ; CONTADORES P/ DELAY

        AUX                 ; REGISTRADOR AUXILIAR DE USO GERAL

        UNIDADE             ; ARMAZENA VALOR DA UNIDADE
        DEZENA              ; ARMAZENA VALOR DA DEZENA
        CENTENA             ; ARMAZENA VALOR DA CENTENA

    ENDC

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *              DEFINIÇÃO DAS VARIÁVEIS INTERNAS DO PIC                    *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  O ARQUIVO DE DEFINIÇÕES DO PIC UTILIZADO DEVE SER REFERENCIADO PARA QUE
;  OS NOMES DEFINIDOS PELA MICROCHIP POSSAM SER UTILIZADOS, SEM A NECESSIDADE
;  DE REDIGITAÇÃO.

   #INCLUDE <P18F4520.INC>   ; MICROCONTROLADOR UTILIZADO

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                          CONSTANTES INTERNAS                            *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  A DEFINIÇÃO DE CONSTANTES FACILITA A PROGRAMAÇÃO E A MANUTENÇÃO.
```

```
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                    DECLARAÇÃO DOS FLAGs DE SOFTWARE                     *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  A DEFINIÇÃO DE FLAGs AJUDA NA PROGRAMAÇÃO E ECONOMIZA MEMÓRIA RAM.

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                                ENTRADAS                                 *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  AS ENTRADAS DEVEM SER ASSOCIADAS A NOMES PARA FACILITAR A PROGRAMAÇÃO E
;  FUTURAS ALTERAÇÕES DO HARDWARE.

;  ESTE PROGRAMA UTILIZA UMA ENTRADA P/ O CONVERSOR AD.
;  ESTA ENTRADA NÃO PRECISA SER DECLARADA, POIS O SOFTWARE NUNCA FAZ
;  REFERÊNCIA A ELA DE FORMA DIRETA, POIS O CANAL AD A SER CONVERTIDO É
;  SELECIONADO NO REGISTRADOR ADCON0 DE FORMA BINÁRIA E NÃO ATRAVÉS DE
;  DEFINES. PORÉM PARA FACILITAR O ENTENDIMENTO DO HARDWARE VAMOS DECLARAR
;  ESTA ENTRADA NORMALMENTE.

#DEFINE CAD_P2  PORTA,1      ; ENTRADA AD P/ O POTENCIÔMETRO P2

;  ALÉM DA ENTRADA DO CONVERSOR AD, TEMOS A ENTRADA DA USART (RECEPÇÃO).
;  NOVAMENTE ESTA ENTRADA NÃO NECESSITA SER DECLARADA, PORÉM, PARA
;  FACILITAR O ENTENDIMENTO DO HARDWARE VAMOS DECLARAR ESTA ENTRADA
;  NORMALMENTE.

#DEFINE RXUSART PORTC,7      ; ENTRADA DE RX DA USART

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                                SAÍDAS                                   *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  AS SAÍDAS DEVEM SER ASSOCIADAS A NOMES PARA FACILITAR A PROGRAMAÇÃO E
;  FUTURAS ALTERAÇÕES DO HARDWARE.

#DEFINE DISPLAY     LATD             ; BARRAMENTO DE DADOS DO DISPLAY

#DEFINE RS          LATE,0           ; INDICA P/ O DISPLAY UM DADO OU COMANDO
                                     ; 1 -> DADO
                                     ; 0 -> COMANDO

#DEFINE ENABLE      LATE,1           ; SINAL DE ENABLE P/ DISPLAY
                                     ; ATIVO NA BORDA DE DESCIDA

;  TEMOS TAMBÉM A SAÍDA DE TX DA USART.
;  NOVAMENTE ESTA SAÍDA NÃO NECESSITA SER DECLARADA, PORÉM, PARA FACILITAR O
;  ENTENDIMENTO DO HARDWARE VAMOS DECLARAR ESTA SAÍDA NORMALMENTE.

#DEFINE TXUSART     PORTC,6          ; SAÍDA DE TX DA USART

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                  VETOR DE RESET DO MICROCONTROLADOR                     *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  POSIÇÃO INICIAL PARA EXECUÇÃO DO PROGRAMA

    ORG     0X0000           ; ENDEREÇO DO VETOR DE RESET
    BRA     CONFIGURACAO     ; PULA PARA CONFIG DEVIDO A REGIÃO
                             ; DESTINADA AS ROTINAS SEGUINTES

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                    ROTINA DE DELAY (DE 1MS ATÉ 256MS)                   *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTA É UMA ROTINA DE DELAY VARIÁVEL, COM DURAÇÃO DE 1MS X O VALOR PASSADO
;  EM WORK (W).

DELAY_MS
    MOVWF   TEMPO1           ; CARREGA TEMPO1 (UNIDADES DE MS)
```

```
    MOVLW   .250
    MOVWF   TEMPO0          ; CARREGA TEMPO0 (P/ CONTAR 1MS)

    CLRWDT                  ; LIMPA WDT (PERDE TEMPO)
    DECFSZ  TEMPO0,F        ; FIM DE TEMPO0 ?
    BRA     $-.4            ; NÃO - VOLTA 2 INSTRUÇÕES
                            ; SIM - PASSOU-SE 1ms
    DECFSZ  TEMPO1,F        ; FIM DE TEMPO1 ?
    BRA     $-.12           ; NÃO - VOLTA 6 INSTRUÇÕES
    RETURN                  ; SIM - RETORNA

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *               ROTINA DE ESCRITA DE UM CARACTERE NO DISPLAY            *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTA ROTINA ENVIA UM CARACTERE PARA O MÓDULO DE LCD. O CARACTERE A SER
;  ESCRITO DEVE SER COLOCADO EM WORK (W) ANTES DE CHAMAR A ROTINA.

ESCREVE
    MOVWF   DISPLAY         ; ATUALIZA DISPLAY (PORTD)
    NOP                     ; PERDE 1US PARA ESTABILIZAÇÃO
    BSF     ENABLE          ; ENVIA UM PULSO DE ENABLE AO DISPLAY
    BRA     $+.2            ; .
    BCF     ENABLE          ; .

    MOVLW   .1
    RCALL   DELAY_MS        ; DELAY DE 1ms
    RETURN                  ; RETORNA

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                              AJUSTE DECIMAL                           *
; *          W [HEX] =  CENTENA [DEC] : DEZENA [DEC] ; UNIDADE [DEC]      *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTA ROTINA RECEBE UM ARGUMENTO PASSADO PELO WORK E RETORNA NAS VARIÁVEIS
;  CENTENA, DEZENA E UNIDADE O NÚMERO BCD CORRESPONDENTE AO PARÂMETRO PASSADO.

AJUSTE_DECIMAL
    MOVWF   AUX             ; SALVA VALOR A CONVERTER EM AUX
    CLRF    UNIDADE
    CLRF    DEZENA          ; RESETA REGISTRADORES
    CLRF    CENTENA

    MOVF    AUX,F           ; VALOR A CONVERTER = 0 ?
    BZ      $+.28           ; SIM - RETORNA
    INCF    UNIDADE,F       ; NÃO - INCREMENTA UNIDADE

    MOVF    UNIDADE,W
    XORLW   0X0A            ; UNIDADE = 10d ?
    BNZ     $+.6            ; NÃO
    CLRF    UNIDADE         ; SIM - RESETA UNIDADE
    INCF    DEZENA,F        ; INCREMENTA DEZENA

    MOVF    DEZENA,W
    XORLW   0X0A            ; DEZENA = 10d ?
    BNZ     $+.6            ; NÃO
    CLRF    DEZENA          ; SIM - RESETA DEZENA
    INCF    CENTENA,F       ; INCREMENTA CENTENA

    DECFSZ  AUX,F           ; FIM DA CONVERSÃO ?

    BRA     $-.24           ; NÃO - VOLTA P/ CONTINUAR CONVERSÃO
    RETURN                  ; SIM

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *             CONFIGURAÇÕES INICIAIS DE HARDWARE E SOFTWARE             *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

```
;  NESTA ROTINA SÃO INICIALIZADAS AS PORTAS DE I/O DO MICROCONTROLADOR E AS
;  CONFIGURAÇÕES DOS REGISTRADORES ESPECIAIS (SFR). A ROTINA INICIALIZA A
;  MÁQUINA E AGUARDA O ESTOURO DO WDT.

CONFIGURACAO
    CLRF    PORTA           ; GARANTE AS SAÍDAS EM ZERO
    CLRF    PORTB
    CLRF    PORTC
    CLRF    PORTD
    CLRF    PORTE

    CLRF    LATA
    CLRF    LATB
    CLRF    LATC
    CLRF    LATD
    CLRF    LATE

    MOVLW   B'11111111'
    MOVWF   TRISA           ; CONFIGURA I/O DO PORTA

    MOVLW   B'11111111'
    MOVWF   TRISB           ; CONFIGURA I/O DO PORTB

    MOVLW   B'10111111'
    MOVWF   TRISC           ; CONFIGURA I/O DO PORTC

    MOVLW   B'00000000'
    MOVWF   TRISD           ; CONFIGURA I/O DO PORTD

    MOVLW   B'00000100'
    MOVWF   TRISE           ; CONFIGURA I/O DO PORTE

    MOVLW   B'11011000'     ; CONFIGURA TIMER0
    MOVWF   T0CON           ; TIMER0 INCREM. PELO CICLO DE MÁQUINA
                            ; TIMER0- 1:1

    MOVLW   B'00000000'     ; CONFIGURA INTERRUPÇÕES
    MOVWF   INTCON          ; DESABILITA TODAS AS INTERRUPÇÕES

    MOVLW   B'00000001'     ; CONFIGURA CONVERSOR AD
    MOVWF   ADCON2          ; VELOCIDADE -> Fosc/8
                            ; JUSTIFICADO À ESQUERDA
                            ; 8 BITS EM ADRESH E 2 BITS EM ADRESL
                            ; TEMPO DE AQUISIÇÃO DO CAPACITOR
                            ; DEFINIDO POR SOFTWARE
                            ; Vref+ = VDD (+5V)
                            ; Vref- = GND ( 0V)

    MOVLW   B'00001101'     ; CONFIGURA CONVERSOR AD
    MOVWF   ADCON1          ; RA0 E RA1 COMO ANALÓGICO
                            ; RA2, RA3, RA4 E RA5 COMO I/O DIGITAL
                            ; PORTB E PORTE COMO I/O DIGITAL
                            ; JUSTIFICADO À ESQUERDA
                            ; 8 BITS EM ADRESH E 2 BITS EM ADRESL
                            ; Vref+ = VDD (+5V)
                            ; Vref- = GND ( 0V)

    MOVLW   B'00000101'     ; CONFIGURA CONVERSOR AD
    MOVWF   ADCON0          ; CANAL 1
                            ; MÓDULO LIGADO

    MOVLW   B'00000000'     ; AUTOBAUD DESABILITADO, WAKE-UP DESABILITADO
    MOVWF   BAUDCON         ; GERADOR DE BAUD RATE EM 8 BITS
```

```
                                ;  PINO  TX  NORMAL  EM  1,  RECEBIMENTO  DE  DADOS  SEM
INVERSÃO

    MOVLW   B'00100100'         ; CONFIGURA USART
    MOVWF   TXSTA               ; HABILITA TX
                                ; MODO ASSINCRONO
                                ; TRANSMISSÃO DE 8 BITS
                                ; HIGH SPEED BAUD RATE
    MOVLW   .25
    MOVWF   SPBRG               ; ACERTA BAUD RATE -> 9600bps

    MOVLW   B'10010000'         ; CONFIGURA USART
    MOVWF   RCSTA               ; HABILITA RX
                                ; RECEPÇÃO DE 8 BITS
                                ; RECEPÇÃO CONTÍNUA
                                ; DESABILITA ADDRESS DETECT

;  AS INSTRUÇÕES A SEGUIR FAZEM COM QUE O PROGRAMA TRAVE QUANDO HOUVER UM
;  RESET OU POWER-UP, MAS PASSE DIRETO SE O RESET FOR POR WDT. DESTA FORMA,
;  SEMPRE QUE O PIC É LIGADO, O PROGRAMA TRAVA, AGUARDA UM ESTOURO DE WDT
;  E COMEÇA NOVAMENTE. ISTO EVITA PROBLEMAS NO START-UP DO PIC.

    BTFSC   RCON,NOT_TO         ; RESET POR ESTOURO DE WATCHDOG TIMER ?
    BRA     $                   ; NÃO - AGUARDA ESTOURO DO WDT
                                ; SIM

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                           INICIALIZAÇÃO DA RAM                            *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTA ROTINA IRÁ LIMPAR TODA A RAM, DO BANCO 0 ATÉ O BANCO 15

    LFSR    FSR0,0X00           ; APONTA FSR0 PARA ENDEREÇO ZERO

LIMPA_RAM
    CLRF    POSTINC0            ; LIMPA A POSIÇÃO E INCREMENTA FSR0
    CLRWDT                      ; LIMPA WATCHDOG TIMER
    MOVLW   0X0F                ; TESTA SE JÁ CHEGOU AO BANCO 15
    SUBWF   FSR0H,W
    BNZ     LIMPA_RAM           ; SE NÃO, CONTINUA LIMPA

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                     CONFIGURAÇÕES INICIAIS DO DISPLAY                     *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; ESTA ROTINA INICIALIZA O DISPLAY P/ COMUNICAÇÃO DE 8 VIAS, DISPLAY PARA 2
; LINHAS, CURSOR APAGADO E DESLOCAMENTO DO CURSOR À DIREITA.

INICIALIZACAO_DISPLAY
    BCF     RS                  ; SELECIONA O DISPLAY P/ COMANDOS

    MOVLW   0X30                ; ESCREVE COMANDO 0X30 PARA
    RCALL   ESCREVE             ; INICIALIZAÇÃO

    MOVLW   .3
    RCALL   DELAY_MS            ; DELAY DE 3MS (EXIGIDO PELO DISPLAY)

    MOVLW   0X30                ; ESCREVE COMANDO 0X30 PARA
    RCALL   ESCREVE             ; INICIALIZAÇÃO

    MOVLW   0X30                ; ESCREVE COMANDO 0X30 PARA
    RCALL   ESCREVE             ; INICIALIZAÇÃO

    MOVLW   B'00111000'         ; ESCREVE COMANDO PARA
    RCALL   ESCREVE             ; INTERFACE DE 8 VIAS DE DADOS
```

```
    MOVLW   B'00000001'     ; ESCREVE COMANDO PARA
    RCALL   ESCREVE         ; LIMPAR TODO O DISPLAY

    MOVLW   .1
    RCALL   DELAY_MS        ; DELAY DE 1MS

    MOVLW   B'00001100'     ; ESCREVE COMANDO PARA
    RCALL   ESCREVE         ; LIGAR O DISPLAY SEM CURSOR

    MOVLW   B'00000110'     ; ESCREVE COMANDO PARA INCREM.
    RCALL   ESCREVE         ; AUTOMÁTICO À DIREITA

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                 ROTINA DE ESCRITA DA TELA PRINCIPAL                   *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTA ROTINA ESCREVE A TELA PRINCIPAL DO PROGRAMA, COM AS FRASES:
;  LINHA 1 - "USART:9600,8,n,1"
;  LINHA 2 - "TX:   d   h RX: "

    MOVLW   0X80            ; COMANDO PARA POSICIONAR O CURSOR
    RCALL   ESCREVE         ; LINHA 0 / COLUNA 0
    BSF     RS              ; CONFIGURA CONVERSOR AD; SELECIONA O DISPLAY P/ DADOS

    MOVLW   'U'             ; COMANDOS PARA ESCREVER AS
    RCALL   ESCREVE         ; LETRAS DE "USART:9600,8,n,1"
    MOVLW   'S'
    RCALL   ESCREVE
    MOVLW   'A'
    RCALL   ESCREVE
    MOVLW   'R'
    RCALL   ESCREVE
    MOVLW   'T'
    RCALL   ESCREVE
    MOVLW   ':'
    RCALL   ESCREVE
    MOVLW   '9'
    RCALL   ESCREVE
    MOVLW   '6'
    RCALL   ESCREVE
    MOVLW   '0'
    RCALL   ESCREVE
    MOVLW   '0'
    RCALL   ESCREVE
    MOVLW   ','
    RCALL   ESCREVE
    MOVLW   '8'
    RCALL   ESCREVE
    MOVLW   ','
    RCALL   ESCREVE
    MOVLW   'n'
    RCALL   ESCREVE
    MOVLW   ','
    RCALL   ESCREVE
    MOVLW   '1'
    RCALL   ESCREVE

    BCF     RS              ; SELECIONA O DISPLAY P/ COMANDO
    MOVLW   0XC0            ; COMANDO PARA POSICIONAR O CURSOR
    RCALL   ESCREVE         ; LINHA 1 / COLUNA 0
    BSF     RS              ; SELECIONA O DISPLAY P/ DADOS

    MOVLW   'T'             ; COMANDOS PARA ESCREVER AS
    RCALL   ESCREVE         ; LETRAS DE "TX:   d   h RX: "
    MOVLW   'X'
```

```
    RCALL   ESCREVE
    MOVLW   ':'
    RCALL   ESCREVE
    MOVLW   ' '
    RCALL   ESCREVE
    MOVLW   ' '
    RCALL   ESCREVE
    MOVLW   ' '
    RCALL   ESCREVE
    MOVLW   'd'
    RCALL   ESCREVE
    MOVLW   ' '
    RCALL   ESCREVE
    MOVLW   ' '
    RCALL   ESCREVE
    MOVLW   ' '
    RCALL   ESCREVE
    MOVLW   'h'
    RCALL   ESCREVE
    MOVLW   ' '
    RCALL   ESCREVE
    MOVLW   'R'
    RCALL   ESCREVE
    MOVLW   'X'
    RCALL   ESCREVE
    MOVLW   ':'
    RCALL   ESCREVE

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                                LOOP PRINCIPAL                         *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  A ROTINA PRINCIPAL FICA AGUARDANDO O FINAL DA CONVERSÃO AD E VERIFICANDO
;  SE ALGUM DADO FOI RECEBIDO PELA USART

    BSF     ADCON0,GO         ; INICIA CONVERSÃO AD
                              ; EXECUTADA APENAS UMA VEZ
LOOP
    CLRWDT                    ; LIMPA WATCHDOG TIMER

    BTFSS   ADCON0,GO         ; FIM DA CONVERSÃO ?
    BRA     FIM_CONVERSAO_AD; SIM
                              ; NÃO
    BTFSC   PIR1,RCIF         ; RECEBEU ALGUM DADO NA SERIAL ?
    BRA     DADO_RECEBIDO     ; SIM
                              ; NÃO
    BRA     LOOP              ; VOLTA P/ LOOP

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                     MOSTRA AD NO DISPLAY E TRANSMITE                 *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTA ROTINA MOSTRA O VALOR DA CONVERSÃO AD NO DISPLAY LCD TANTO EM DECIMAL
;  COMO EM HEXADECIMAL. O VALOR DA CONVERSÃO TAMBÉM É TRANSMITIDO PELA USART.
;  AO FINAL, A ROTINA REQUISITA UMA NOVA CONVERSÃO AD.

FIM_CONVERSAO_AD

; ************* MOSTRA VALOR DA CONVERSÃO AD EM DECIMAL *********************

    MOVF    ADRESH,W          ; CARREGA WORK COM VALOR DO AD
    RCALL   AJUSTE_DECIMAL    ; CHAMA ROTINA DE AJUSTE DECIMAL

    BCF     RS                ; SELECIONA O DISPLAY P/ COMANDO
    MOVLW   0XC3              ; COMANDO PARA POSICIONAR O CURSOR
    RCALL   ESCREVE           ; LINHA 1 / COLUNA 3
```

```
BSF     RS                  ; SELECIONA O DISPLAY P/ DADOS

MOVF    CENTENA,W
ADDLW   0X30                ; CONVERTE BCD DA CENTENA EM ASCII
RCALL   ESCREVE             ; ENVIA AO LCD

MOVF    DEZENA,W
ADDLW   0X30                ; CONVERTE BCD DA DEZENA EM ASCII
RCALL   ESCREVE             ; ENVIA AO LCD

MOVF    UNIDADE,W
ADDLW   0X30                ; CONVERTE BCD DA UNIDADE EM ASCII
RCALL   ESCREVE             ; ENVIA AO LCD

; ************* MOSTRA VALOR DA CONVERSÃO AD EM HEXADECIMAL ****************

    BCF     RS              ; SELECIONA O DISPLAY P/ COMANDO
    MOVLW   0XC8            ; COMANDO PARA POSICIONAR O CURSOR
    RCALL   ESCREVE         ; LINHA 1 / COLUNA 8
    BSF     RS              ; SELECIONA O DISPLAY P/ DADOS

    SWAPF   ADRESH,W        ; INVERTE NIBLE DO ADRESH
    ANDLW   B'00001111'     ; MASCARA BITS MAIS SIGNIFICATIVOS
    MOVWF   AUX             ; SALVA EM AUXILIAR

    MOVLW   0X0A
    SUBWF   AUX,W           ; AUX - 10d (ATUALIZA FLAG DE CARRY)
    MOVLW   0X30            ; CARREGA WORK COM 30h
    BTFSC   STATUS,C        ; RESULTADO É POSITIVO? (É UMA LETRA?)
    MOVLW   0X37            ; SIM - CARREGA WORK COM 37h
                            ; NÃO - WORK FICA COM 30h (NÚMERO)
    ADDWF   AUX,W           ; SOMA O WORK AO AUXILIAR
                            ; (CONVERSÃO ASCII)
    RCALL   ESCREVE         ; ENVIA CARACTERE AO DISPLAY LCD

    MOVF    ADRESH,W        ; CARREGA WORK COM ADRESH
    ANDLW   B'00001111'     ; MASCARA BITS MAIS SIGNIFICATIVOS
    MOVWF   AUX             ; SALVA EM AUXILIAR

    MOVLW   0X0A
    SUBWF   AUX,W           ; AUX - 10d (ATUALIZA FLAG DE CARRY)
    MOVLW   0X30            ; CARREGA WORK COM 30h
    BTFSC   STATUS,C        ; RESULTADO É POSITIVO? (É UMA LETRA?)
    MOVLW   0X37            ; SIM - CARREGA WORK COM 37h
                            ; NÃO - WORK FICA COM 30h (NÚMERO)
    ADDWF   AUX,W           ; SOMA O WORK AO AUXILIAR
                            ; (CONVERSÃO ASCII)
    RCALL   ESCREVE         ; ENVIA CARACTERE AO DISPLAY LCD

; ************* TRANSMITE VALOR DA CONVERSÃO AD PELA USART *****************

    BTFSS   TXSTA,TRMT      ; O BUFFER DE TX ESTÁ VAZIO ?
    BRA     $-.2            ; NÃO - AGUARDA ESVAZIAR
    MOVFF   ADRESH,TXREG    ; ENVIA O VALOR DO AD EM TXREG (INICIA TX)

; ************************ INICIA UMA NOVA CONVERSÃO ***************************

    BSF     ADCON0,GO       ; PEDE UMA NOVA CONVERSÃO AD
    BRA     LOOP            ; VOLTA PARA LOOP

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                    ROTINA DE RECEPÇÃO DE DADOS NA USART                 *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTA ROTINA É EXECUTADA TODA VEZ QUE UM NOVO DADO É RECEBIDO PELA USART.
```

```
;  O DADO RECEBIDO É MOSTRADO NO LCD (EM ASCII).

DADO_RECEBIDO
    BCF     RS              ; SELECIONA O DISPLAY P/ COMANDO
    MOVLW   0XCF            ; COMANDO PARA POSICIONAR O CURSOR
    RCALL   ESCREVE         ; LINHA 1 / COLUNA 15
    BSF     RS              ; SELECIONA O DISPLAY P/ DADOS

    MOVF    RCREG,W         ; CARREGA DADO RECEBIDO NO WORK
    RCALL   ESCREVE         ; ENVIA AO LCD
                            ; AO LER O REGISTRADOR RCREG O BIT
                            ; RCIF DA INTERRUPÇÃO É LIMPO
                            ; AUTOMATICAMENTE.
    BRA     LOOP            ; VOLTA P/ LOOP PRINCIPAL

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                              FIM DO PROGRAMA                          *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

    END                     ; FIM DO PROGRAMA
```

» Dicas e comentários

A rotina de conversão Hex >>> Decimal deste exemplo é mais completa, pois trabalha com três dígitos (CENTENA, DEZENA e UNIDADE). Desta forma, ela pode converter todo o range do argumento de entrada (W) que vai de 0 a 255.

O sistema de conversão AD é o mesmo apresentado no Capítulo 7, onde utilizamos o conversor interno e só consideramos os 8 bits mais significativos. Com isso o valor já fica limitado a um byte.

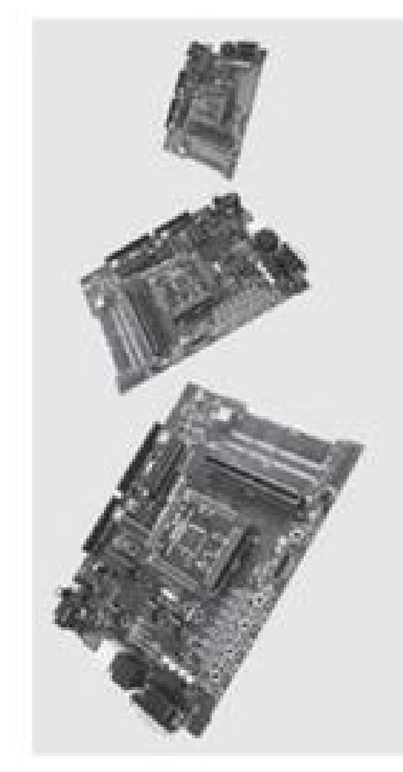
Devido à simplicidade do sistema, não foi necessário o uso das interrupções, deixando-as desabilitadas. Para o caso da recepção, o bit **RCIF** é testado toda vez dentro do loop principal. Quanto à transmissão, sempre que um novo valor foi convertido, verificamos se o buffer de saída está vazio para podermos escrever o novo valor.

» Exercícios propostos

Agora que o exemplo foi estudado e esclarecido, aproveite para gerar novos problemas e soluções, seguindo os exercícios propostos:

1. Ativar o uso da interrupção de recebimento. Quanto à transmissão, em vez de deixá-la contínua, criar uma interrupção de timer como base de tempo. Por exemplo, transmitir o valor atual convertido a cada um segundo.

2. Criar um programa para ligar e desligar o ventilador da placa através da USART. Ao receber o caractere 'L', deve ligar o ventilador e ao receber o caractere 'D', deve desligar o ventilador. Depois fazer a mesma implementação para os LEDs de placa.



13

Outras Características

Introdução

Este capítulo é destinado à apresentação de alguns recursos ou sistemas que não foram abordados nos capítulos anteriores. Estes assuntos são divididos em tópicos para facilitar a didática e a compreensão.

Comunicação paralela (PSP)

O sistema de comunicação paralela em 8 bits do PIC18F4520 é muito simples, por isso nem mereceu um capítulo dedicado a ele.

Sua primeira limitação é que ele só trabalha em modo Slave, isto é, controlado por alguma outra unidade. Por isso, esse sistema recebe o nome de PSP (Parallel Slave Port).

O controle da ativação da PSP encontra-se em **TRISE<PSPMODE>**:

PSPMODE	Descrição
0	PSP desativada. Pinos como I/Os convencionais.
1	PSP ativada. Pinos controlados internamente.

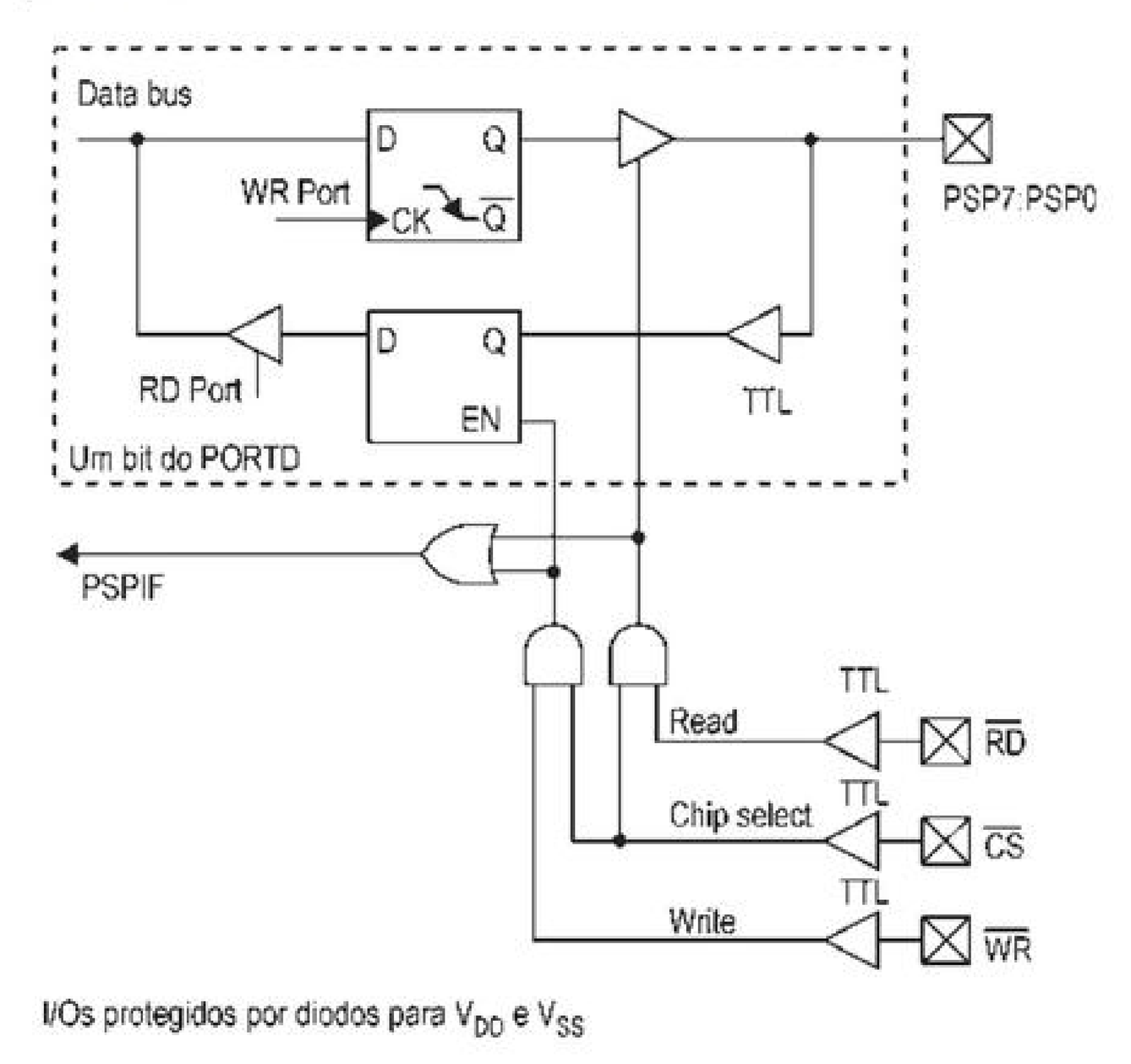
Os pinos utilizados pela porta paralela são:

Pino	Descrição
RD7... RD0	Barramento de dados (8 bits). É lido/escrito diretamente nos latches do PORTD.
RE0 / RD	Efetua uma leitura (Read). Configurado como entrada.
RE1 / WR	Efetua uma escrita (Write). Configurado como entrada.
RE2 / CS	Ativa/desativa a PSP externamente. Configurado como entrada.

Apesar do controle interno da porta paralela, os pinos **/RD**, **/WR** e **/CS** devem ser configurados anteriormente com entrada (**TRISE**) e com funcionamento digital (desligar o AD nesses pinos).

Editora Érica - Desbravando o Microcontrolador PIC18 Recursos Avançados - Daniel R. de Sousa, David José de Souza e Nicolás César Lavinia - 1ª Edição

O **PORTD** possui dois Latches, um para a saída e outro para a entrada. Por causa disso, o estado do **TRISD** não é considerado. Na figura seguinte, pode-se visualizar o hardware interno do PIC para o **PORTD** quando configurado para a PSP.



O pino **/CS** funciona como um Chip Select, e a porta somente opera quando esse pino estiver em nível baixo (0). Quando ele está em nível alto (1), os pinos do **PORTD** ficam como entrada, o latch de entrada não é alterado e nenhuma interrupção é gerada.

/CS	Descrição
0	PSP ativada externamente.
1	PSP desativada externamente.

Para uma operação de escrita, o Mestre deve manter **/CS=0** e **/WR=0**. Neste caso, o latch de entrada é atualizado para todo o barramento de dados (8 bits). O bit **PIR1<PSPIF>** é setado e a interrupção ativada (se as chaves estiverem corretamente ligadas). O bit **TRISE<IBF>** também é ativado, indicando que existe um dado recebido em **PORTD**. Esse bit é limpo automaticamente quando **PORTD** for lido. Caso seja efetuada uma nova operação de escrita sem o sistema ter lido o **PORTD** anteriormente, temos ainda **IBF=1** e, então, um erro de overflow ocorre, sendo indicado através do bit **TRISE<IBOV>=1**. Neste caso, o dado anterior terá sido perdido. O bit **IBOV** deve ser limpo manualmente pelo programa.

/WR	Descrição
0	Operação de escrita.
1	Situação normal.

IBF	Descrição
0	Nenhum dado novo escrito no latch de entrada
1	Um novo dado foi escrito no latch de entrada. Aconteceu uma operação de escrita.

IBOV	Descrição
0	Sem erro de overflow.
1	Houve um erro de overflow.

Uma operação de leitura é feita através da situação **/CS=0** e **/RD=0**. Todo o **PORTD** é transformado em saída e o valor previamente escrito no registrador **PORTD** (latch de saída) é imposto ao barramento. Aqui a interrupção também pode acontecer, pois o flag **PIR1<PSPIF>** é setado. Sempre que algum dado é escrito no registrador PORTD (latch de saída), temos **TRISE<OBF>=1**. Após uma operação de leitura da porta paralela, esse bit é limpo automaticamente.

/RD	Descrição
0	Operação de leitura.
1	Situação normal.

OBF	Descrição
0	Aconteceu uma operação de leitura.
1	Aconteceu uma escrita interna no latch de saída.

Os pinos **/WR** e **/RD** precisam receber somente um pulso para ativar as operações de escrita e leitura. O tempo mínimo desse pulso é de (1 / F_{OSC}), mas recomendamos a utilização de T_{CY}. O tempo máximo é indefinido e depende da lógica do sistema. Caso **/CS** não precise ser controlado pelo Master (só existe um Slave no barramento), então esse pino pode ser diretamente aterrado.

» High/Low Voltage Detect (HLVD)

O módulo High/Low Voltage Detect (HLVD) permite a monitoração da tensão de alimentação do microcontrolador. É um periférico que podemos utilizar para detectar uma queda de tensão de um sistema alimentado por baterias, monitorar uma carga de baterias ou até mesmo a falta de energia elétrica.

O direcionamento do evento é dado pelo bit **HLVDCON<VDIRMAG>**:

HLVDCON<VDIRMAG>	Evento
0	Tensão de entrada (V_{CC}) ≤ tensão de referência
1	Tensão de entrada (V_{CC}) ≥ tensão de referência

O registrador **HLVDCON** configura esse módulo. Podemos configurar a tensão de comparação nos bits **HLVDCON<HLVD3:HLVD0>**:

HLVDCON<HLVD3:HLVD0>	Tensão de ref. (V)	HLVDCON<HLVD3:HLVD0>	Tensão de ref. (V)
0000	2,17	1000	3,39
0001	2,23	1001	3,55
0010	2,36	1010	3,71
0011	2,44	1011	3,90
0100	2,60	1100	4,11
0101	2,79	1101	4,33
0110	2,89	1110	4,59
0111	3,12	1111	Externo por meio do pino HLVDIN

Uma vez o módulo habilitado, recomenda-se a verificação do bit **HLVDCON<IRVST>**. Quando o bit estiver setado, o módulo está pronto para gerar interrupção.

Resumo dos registradores associados ao HLVD

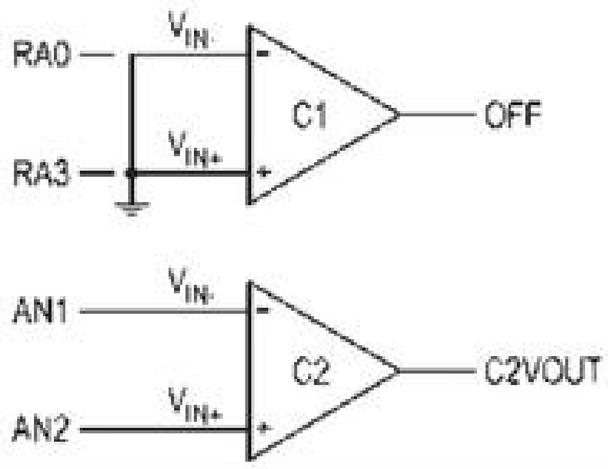
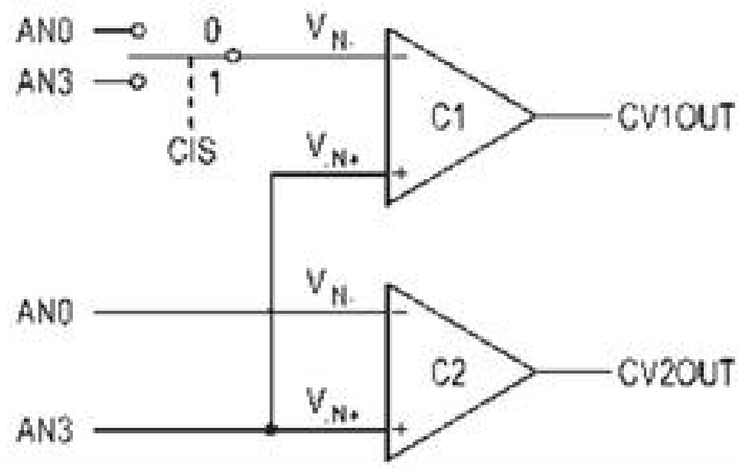
Endereço	Nome	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
FD2h	HLVDCON	VDIRMAG	-	IRVST	HLVDEN	HLVD3	HLVD2	HLVD1	HLVD0

» Comparadores analógicos

O PIC18F4520 possui internamente dois comparadores analógicos, em que temos oito opções diferentes de configuração. Podemos ligar as entradas dos comparadores nos pinos dedicados para essa função, ou uma das entradas ligada a um módulo gerador de tensão de referência que veremos mais adiante, monitorar a saída do comparador logicamente ou externar a saída em um outro pino.

A configuração de hardware dos comparadores de tensão é feita nos bits **CMCON<CM2:CM0>** e pelo bit **CMCON<CIS>**, como mostrado a seguir:

CM	Esquema	CM	Esquema
000	Reset dos comparadores AN0, AN3, C1, OFF AN1, AN2, C2, OFF	111	Comparadores desligados RA0, RA3, C1, OFF RA1, RA2, C2, OFF
100	2 comparadores independentes AN0, AN3, C1, C1VOUT AN1, AN2, C2, C2VOUT	010	4 entradas multiplexadas AN0, AN3, 0, 1, C1, CV1OUT AN0, AN3, 0, 1, C2, CV2OUT, CIS, para V_{REF}
011	2 comparadores com ref. comum AN0, RA3, C1, C1VOUT AN1, AN2, C2, C2VOUT	110	2 comparadores com ref. comum e saídas AN0, CMP1, C1, C1VOUT AN1, AN2, CMP2, C2, C2VOUT, Open Drain

CM	Esquema	CM	Esquema
101	1 comparador independente 	001	3 entradas multiplexadas 

Os bits **CMCON<CxOUT>**,**CMCON<CxINV>** (sendo x o comparador: 1 ou 2) determinam a resposta da saída do comparador; **CMCON<CxINV>** configura a inversão da resposta da saída do comparador e **CMCON<CxOUT>** o bit de monitoração da saída após um evento de interrupção, como mostrado na tabela seguinte:

CMCON<CxINV>	CMCON<CxOUT>	Situação
0	0	Cx V_{IN+} > Cx V_{IN-}
0	1	Cx V_{IN+} < Cx V_{IN-}
1	0	Cx V_{IN+} < Cx V_{IN-}
1	1	Cx V_{IN+} > Cx V_{IN-}

Resumo dos registradores associados aos comparadores analógicos

Endereço	Nome	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
FB4h	CMCON	C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0

» Módulo gerador de tensão de referência

O módulo gerador de tensão de referência pode ser utilizado em conjunto com os comparadores analógicos ou simplesmente na geração de uma tensão externa. Para compreender o funcionamento desse módulo vejamos a próxima figura.

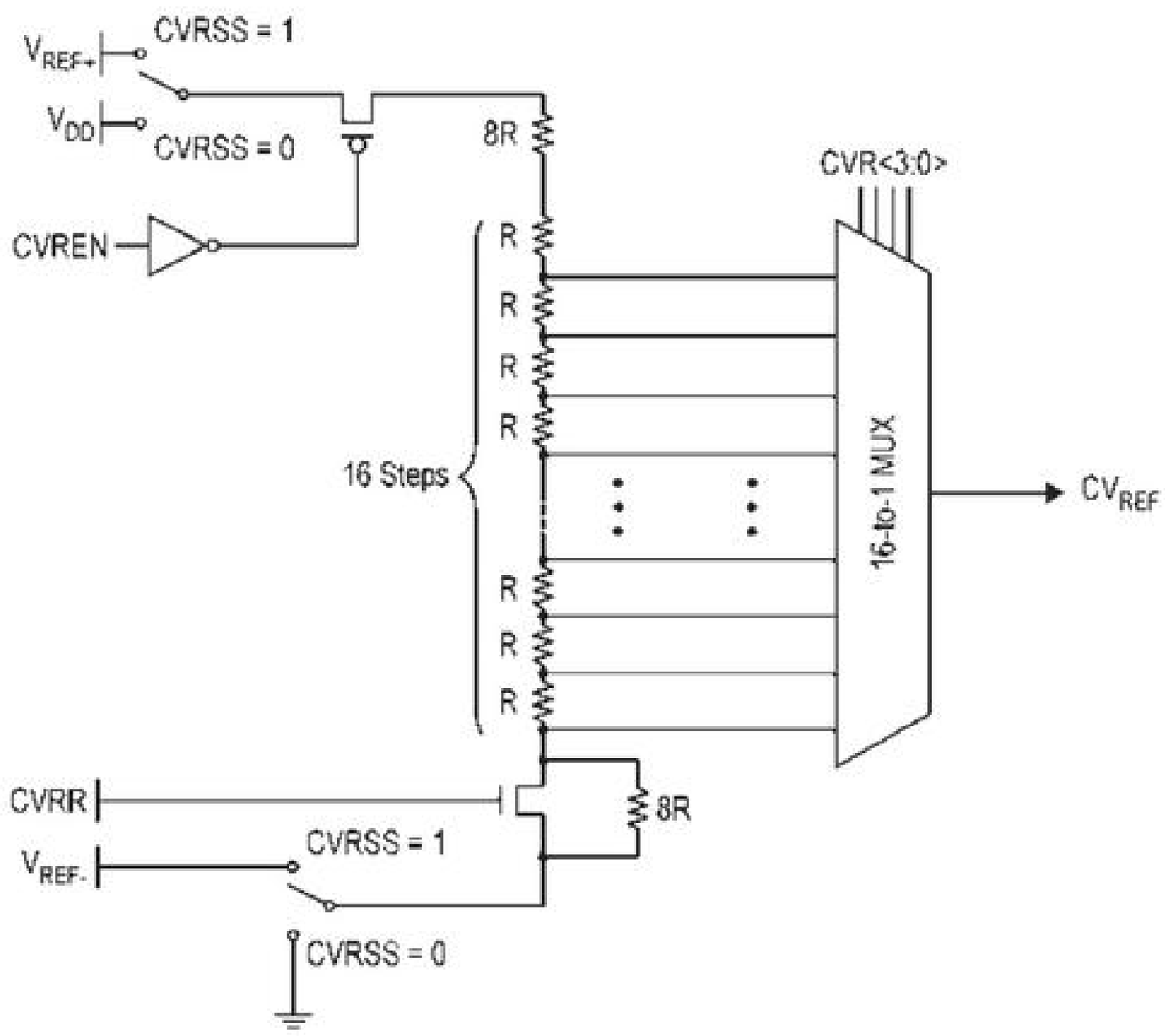
O registrador **CVRCON** configura todo o módulo. O bit **CVRCON<CVREN>** setado habilita o módulo. Já o bit **CVRCON<CVRSS>** configura a tensão a ser utilizada pelo módulo, seja interna (V_{DD} e V_{SS}) ou externa (V_{REF+} e V_{REF-}).

Os bits **CVRCON<CVR3:CVR0>** e o bit **CVRCON<CVRR>** configuram a tensão de saída do módulo.

Para **CVRCON<CVRR>** = 0:

$$C_{VREF} = \frac{(PASSO \times CV_{rscs})}{24}$$

Sendo PASSO um valor de 0 a 15, definido em **CVRCON<CVR3:CVR0>**, e CV_{RSCS} a diferença entre as tensões de entrada do módulo gerador de tensão de referência (se **CVRCON<CVRSS>** = 0, $CV_{RSCS} = V_{DD}$; caso contrário, $CV_{RSCS} = V_{REF+} - V_{REF-}$).



Já para **CVRCON<CVRR>** = 1:

$$C_{VREF} = \frac{CV_{rscs}}{4} + \frac{(PASSO \times CV_{rscs})}{32}$$

O bit **CVRCON<CROE>** configura se o nível de tensão C_{Vref} estará conectado ao pino C_{Vref}. Setando o bit **CVRCON<CROE>**, a tensão gerada pelo módulo fica conectada ao pino C_{vref}.

Resumo dos registradores associados ao módulo de tensão de referência

Endereço	Nome	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
FB5h	CVRCON	CVREN	CVROE	CVRR	CVRSS	CVR3	CVR2	CVR1	CVR0

» Watchdog Timer (WDT)

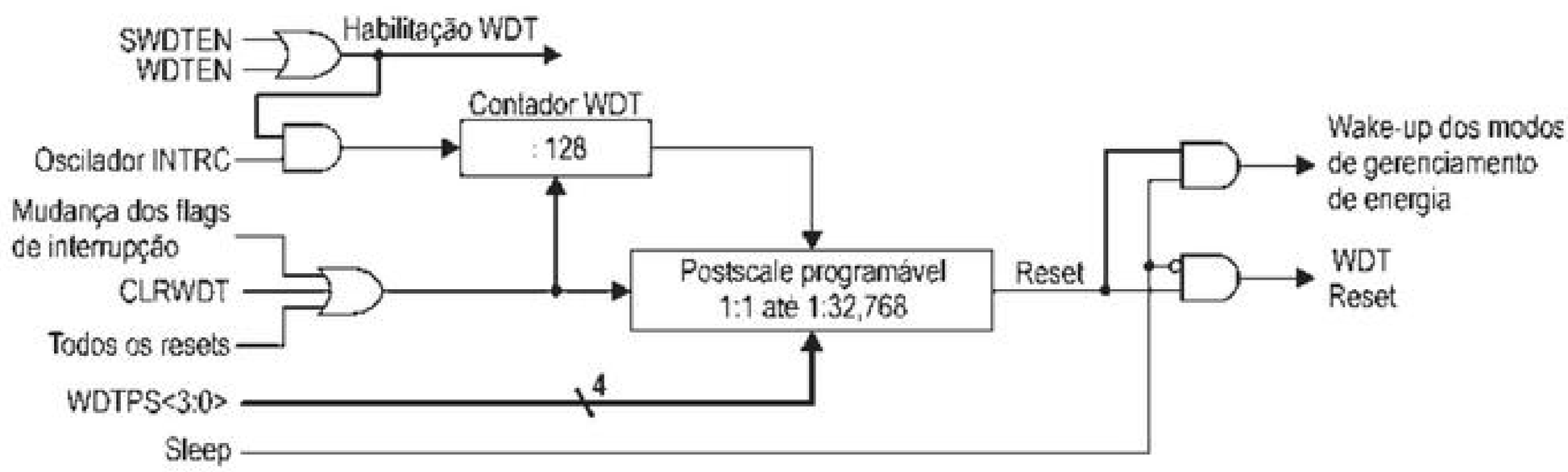
O Watchdog Timer é uma espécie de temporizador que, de tempos em tempos, reseta o microcontrolador. É muito utilizado para evitar que o sistema trave, seja por algum tipo de falha ou porque o programa entrou em um loop infinito. Para que o PIC não seja resetado, o contador do WDT deve ser periodicamente zerado. Essa função é tão importante que existe uma instrução especialmente criada para isso (CLRWDT).

O contador do WDT é independente do ciclo de máquina do PIC. Existe dentro da pastilha um circuito oscilador do tipo RC que incrementa o contador do WDT, de forma que ele trabalha independentemente do oscilador do PIC. Isso significa que o WDT continua operando mesmo em modo SLEEP.

O período de estouro do WDT é de aproximadamente 4 ms. Esse tempo pode ser aumentado até 131072 ms (2 minutos, 11 segundos e 72 milissegundos), configurando os configurations bits **WDTPS = xxx** (sendo xxx o fator de divisão do postscaler programável do módulo, que vai de 1:1 até 1:32768).

Editora Érica - Desbravando o Microcontrolador PIC18 Recursos Avançados - Daniel R. de Sousa, David José de Souza e Nicolás César Lavinia - 1ª Edição

Os bits **RCON</TO>** e **RCON </PD>** são setados sempre que a instrução CLRWDT for executada. O bit **/TO** é zerado sempre que ocorrer o estouro do Watchdog Timer.



O ideal é que apenas uma instrução CLRWDT seja utilizada no software todo. Por isso, geralmente, essa instrução é encontrada apenas no loop principal do programa. Dependendo da aplicação, em algumas rotinas de delay também se faz necessário o uso da instrução CLRWDT, apesar de isso não ser muito aconselhável.

Uma vez habilitado o Watchdog Timer, podemos ligar ou desligar momentaneamente através do bit **WDTCON<SWDTEN>**. O bit **WDTCON<SWDTEN>** setado mantém o Watchdog Timer ligado.

Resumo dos registradores associados ao Watchdog Timer

Endereço	Nome	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
FD0h	RCON	IPEN	SBOREN	-	/RI	/TO	/PD	/POR	/BOR
FD1h	WDTCON								SWDTEN

Não é usado para essa finalidade. Para obter mais informações, consulte o Apêndice A.

» Power-on Reset (POR)

Um pulso de Power-on Reset é gerado internamente no PIC sempre que a borda de subida do V_{DD} for detectada, ou seja, somente na energização do microcontrolador. O POR é utilizado para garantir que o microcontrolador opere corretamente e não sofra problemas durante a energização. Esse pulso dispara o temporizador do Power-up Timer, caso este esteja habilitado.

Quando um POR ocorre, o bit **RCON</POR>** é limpo (zero).

» Power-up Timer (PWRT)

O Power-up Timer é um temporizador utilizado para garantir que o microcontrolador somente comece a operar depois que a fonte de alimentação estiver estabilizada. O PWRT faz com que o PIC permaneça em estado de reset durante um intervalo de 72 ms. Esse tempo é contabilizado após o pulso do POR e pode variar em função da pastilha e da temperatura.

O PWRT opera com um circuito oscilador RC interno dedicado que somente pode ser habilitado no momento da gravação do PIC. Deve obrigatoriamente estar habilitado caso se deseje utilizar o Brown-out Reset.

» Oscilator Start-up Timer (OST)

O Oscilator Start-up Timer fornece um delay de 1.024 ciclos do cristal (ou do oscilador ligado ao pino **OSC1**) após o final do PWRT para iniciar a execução do software. É importante para garantir que o oscilador já esteja operando e estável quando o software começar a ser executado.

Ele é habilitado automaticamente quando o PIC estiver utilizando osciladores LP, XT ou HS e seu delay (1.024 ciclos do oscilador) é gerado sempre em três diferentes situações: na energização (POR), após um Brown-out Reset e após um wake-up ("acordar") do modo SLEEP.

» Brown-out Reset (BOR)

O Brown-out Reset força um reset do PIC sempre que a tensão de alimentação cair abaixo de 4 V (típico) durante um intervalo de tempo maior que 100 µs (típico). Caso a queda de tensão seja durante um intervalo de tempo menor que 100 µs, o reset não é gerado. A duração do reset é de 72 ms (lembra-se do PWRT?) e, enquanto a tensão de alimentação não for superior ao valor programado nos configurations bits, esse temporizador não é inicializado.

Quando um BOR ocorre, o bit **RCON</BOR>** é limpo (zero).

Para o PIC18F4520 o valor de tensão do BOR pode ser configurado nos configurations bits:

BORV	Tensão (Volts)
0	2,11
1	2,79
2	4,33
3	4,59

Em alguns modelos esse valor é fixo e não pode ser (outros permitem valores programáveis), além de somente poder ser habilitado na gravação do microcontrolador.

Habilitado o Brown-out Reset pelo configurations bits (**BOREN = ON**), podemos habilitar ou desabilitar o recurso por software através do bit **RCON<SBOREN>**, bastando setar o bit para habilitar o módulo. Alterando o configuration bits para **BOREN = SBORDIS**, desabilita-se o controle do Brown-out Reset por software.

» SLEEP (Power-down Mode)

O PIC entra em SLEEP (Power-down Mode) sempre que a instrução SLEEP for executada. Neste modo de operação, o oscilador é desligado, os pinos de I/Os mantêm o estado anterior à instrução SLEEP e o WDT é resetado, embora continue operando.

Para minimizar o consumo, deve-se desligar o conversor AD, desabilitar clocks externos, colocar todos os pinos em entrada e desligar os pull-ups do **PORTB**. Para casos extremos, deve-se levar em conta também o uso do Watchdog Timer (o circuito oscilador RC permanece consumindo), Power-up Timer (também utiliza um oscilador RC) e Brown-out Reset.

Para que o PIC volte a operar e "acorde" da situação de SLEEP existem três formas:

- ✓ por meio de um reset externo na entrada **/MCLR**;
- ✓ pelo estouro do WDT, se habilitado;
- ✓ ou pelo evento de alguma interrupção.

Um reset externo na entrada **/MCLR** causa o reset do PIC (PC=0000h). Todas as outras situações são consideradas uma continuação do programa e causam um wake-up na linha seguinte à instrução SLEEP. Os bits **RCON</TO>** e **RCON</PD>** podem ser testados para descobrir a causa do reset.

/TO	Descrição
0	Indica que ocorreu um estouro do WDT.
1	Indica que ocorreu um Power-up ou foram executadas as instruções CLRWDT ou SLEEP.

/PD	Descrição
0	Indica que a instrução SLEEP foi executada.
1	Indica que ocorreu um Power-up ou foi executada a instrução CLRWDT.

Quando a instrução SLEEP for executada, a próxima instrução (PC + 2) já estará sendo processada (lembre-se do conceito do Pipeline). Para que o PIC "acorde" por um evento de interrupção, a chave individual da interrupção deve estar ligada. Ao "acordar" nessa situação, existem duas possibilidades. Se o bit **INTCON<GIEH>** estiver desabilitado, o programa continua a ser executado no ponto seguinte à instrução SLEEP. Se o bit **INTCON<GIEH>** estiver habilitado, a instrução seguinte à instrução SLEEP é processada e, em seguida, o PC é desviado para o endereço 0008h ou 0018h (vetores de interrupção). Por este motivo é recomendado que logo após a instrução SLEEP seja adicionado um NOP.

Se a chave geral de interrupções (**INTCON<GIEH>**=0) estiver desabilitada e caso nenhuma interrupção habilitada individualmente tenha ocorrido, ao ser executada a instrução SLEEP duas situações podem ocorrer:

- ✓ Se o evento da interrupção ocorrer antes da execução da instrução SLEEP, a instrução SLEEP é executada como um NOP. Entretanto, o WDT não é resetado, o bit **RCON</TO>** não é setado e o bit **RCON</PD>** não é limpo.
- ✓ Se o evento da interrupção ocorrer durante ou após a execução da instrução SLEEP, a instrução é executada normalmente, porém o microcontrolador imediatamente "acorda" do SLEEP. Neste caso, o WDT é resetado, o bit **RCON</TO>** é setado e o bit **RCON</PD>** é limpo.

Desta maneira, não adianta testar esses bits antes da execução da instrução SLEEP, pois eles podem ser alterados enquanto a instrução está sendo executada. Para saber se a instrução foi corretamente executada, deve-se testar o bit **RCON</PD>**. Se ele estiver setado, a instrução foi executada como um NOP. Quanto ao WDT, recomenda-se executar um CLRWDT antes da instrução SLEEP.

» Controle de resets

Como pode ser observado, o PIC18F4520 apresenta diversas formas de reset. A combinação de alguns bits/flags especiais pode informar precisamente qual dessas formas ocorreu. Vejamos a tabela seguinte:

/POR	/BOR	/TO	/PD	/RI	Causa
0	x	1	1	1	Ocorreu um Power-on Reset.
0	x	0	x	x	Não pode ocorrer. /TO é setado em POR.
0	x	x	0	x	Não pode ocorrer. /PD é setado em POR.
1	0	1	1	1	Ocorreu um Brown-out Reset (BOR).
1	1	0	1	1	Ocorreu um WDT Reset.
1	1	0	0	1	Ocorreu um WDT Wake-up (acordou).
1	1	u	u	1	Ocorreu um MCRL Reset durante operação normal.
1	1	1	0	1	Ocorreu um MCRL Reset durante SLEEP ou acordou SLEEP por interrupção.
1	1	1	1	0	Ocorreu a execução da instrução RESET.
1	1	1	1	1	Ocorreu um reset por Stack Full/Underflow.

Resumo dos registradores associados aos resets

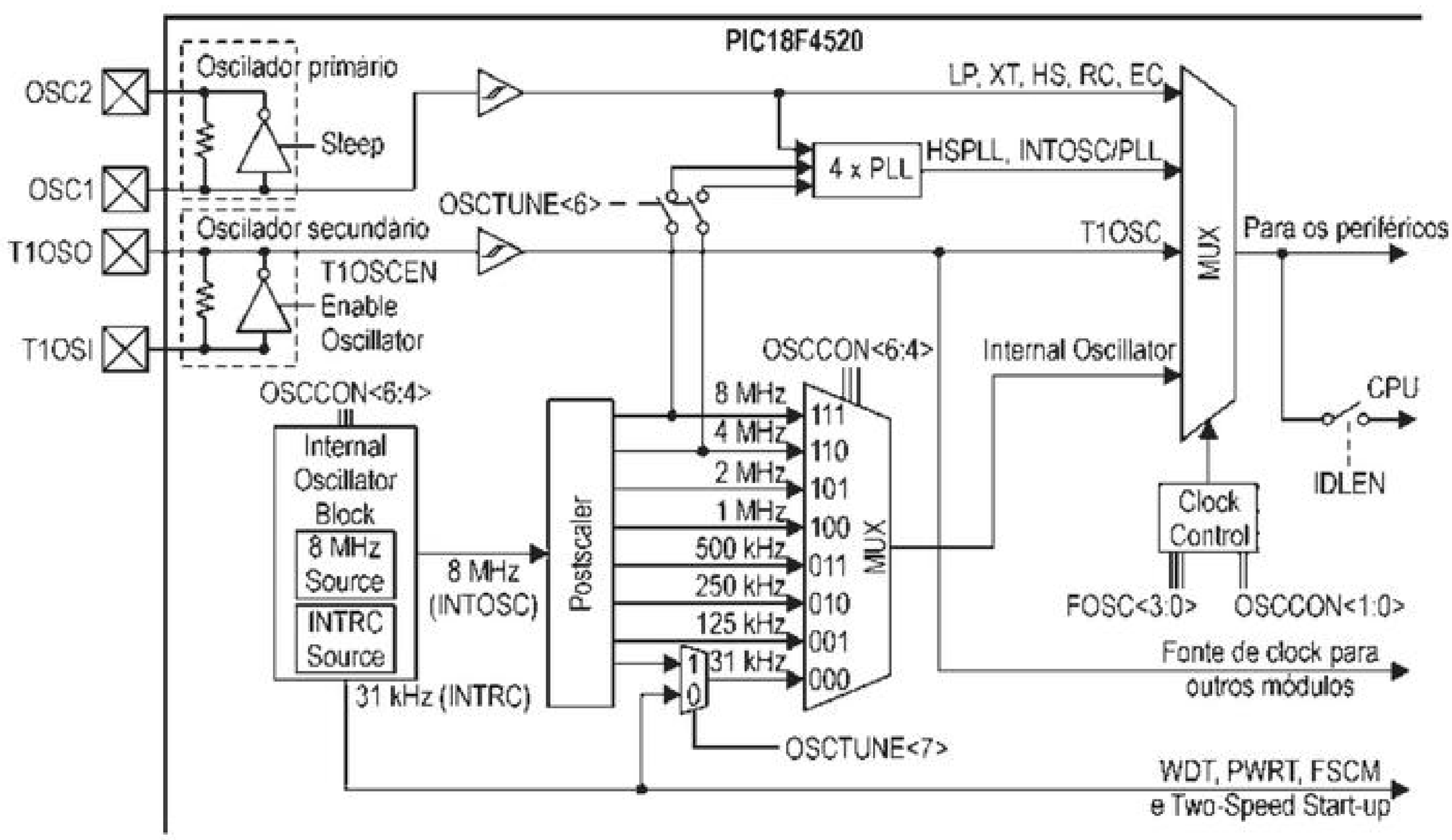
Endereço	Nome	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
FD0h	RCON	IPEN	SBOREN	-	/RI	/TO	/PD	/POR	/BOR

Não é usado para essa finalidade. Para obter mais informações, consulte o Apêndice A.

» Oscilador

O PIC18F4520 pode operar com dez tipos diferentes de osciladores:

- ✓ LP - cristal de baixa potência.
- ✓ XT - cristal/ressonador.
- ✓ HS - cristal/ressonador de alta frequência.
- ✓ HSPLL - cristal/ressonador de alta frequência com o PLL habilitado.
- ✓ RC - oscilador RC, RA6 é saída de clock, Fosc\4.
- ✓ RCIO - oscilador RC, RA6 pino de I/O.
- ✓ INTIO1 - oscilador RC interno, RA6 saída de clock Fosc\4 e RA7 pino de I/O.
- ✓ INTIO2 - oscilador RC interno, RA6 e RA7 pino de I/O.
- ✓ EC - RA7 é uma entrada de clock externa e RA6 saída de clock Fosc\4.
- ✓ ECIO - RA7 é uma entrada de clock externa e RA6 pino de I/O.



O sistema de clock permite utilizar o oscilador interno de 31 kHz ou 8 MHz, ou o Timer1 ou um cristal ou ressonador para geração de clock do microcontrolador.

Esse sistema possui essencialmente três fontes de clock, que são oscilador primário, oscilador secundário, bloco de oscilador interno.

Oscilador primário

O oscilador primário é formado pelo oscilador externo que pode ser um cristal ou ressonador (LP, XT, HS e HSPLL), pelo oscilador interno do tipo RC, pelo sistema de clock externo (EC) ou pelo oscilador externo do tipo RC.

Oscilador secundário

O módulo secundário é formado pelo oscilador do Timer1.

Bloco oscilador interno

A terceira fonte de clock faz parte do sistema para baixo consumo. Esse sistema pode ser utilizado na falha do oscilador primário, garantindo a continuidade de funcionamento do sistema.

É possível trabalhar com 31 kHz ou 8 MHz. Trabalhando em 8 MHz, podemos utilizar o postscale e selecionar frequências mais baixas, diminuindo, desta forma, o consumo do microcontrolador.

Configuração do sistema de clock

O registrador **OSCCON** é responsável pela seleção das fontes de clock do sistema, primário, secundário e bloco oscilador interno. Através dos bits **OSCCON<SCS1:SCS0>** podemos selecionar a fonte de clock. Com os bits **OSCCON<IRCF2:IRCF0>** selecionamos o postscale do oscilador interno RC, que pode ser ajustado de 31,25 kHz a 8 MHz, desde que o RC esteja configurado para 8 MHz.

Note que o bit **OSCTUNE<INTSRC>** também faz parte da seleção da frequência do oscilador interno, como mostra a tabela seguinte:

OSCTUNE<INTSRC>	OSCCON<IRCF2:IRCF0>	Frequência
0	000	31 kHz (INTRC)
1	000	31,25 kHz (8 MHz/256)
x	001	125 kHz (8 MHz/64)
x	010	250 kHz (8 MHz/32)
x	011	500 kHz (8 MHz/16)
x	100(1)	1 MHz (8 MHz/8)
x	101	2 MHz (8 MHz/4)
x	110(2)	4 MHz (8 MHz/2)
x	111(2)	8 MHz (8 MHz/1)

Nota

(1) - Ao selecionar o oscilador interno, a frequência padrão será de 1 MHz, sendo necessário o ajuste conforme aplicação.

(2) - Os valores de 4 MHz e 8 MHz podem ser conectados ao PLL após setar o bit **OSCTUNE<PLLEN>**.

No oscilador de 8 MHz (INTOSC) pode ser ajustada a frequência nos bits **OSCTUNE<TUN4:TUN0>** dentro de uma faixa de +/– 10% da frequência central de 8 MHz:

OSCTUNE<TUN4:TUN0>	Ajuste
01111	Máxima frequência
...	.
00001	.
00000	Frequência central
11111	.
...	.
10000	Mínima frequência

Ao configurar o oscilador externo com a opção do PLL, devemos lembrar que o PLL multiplica a frequência de entrada por quatro e que a frequência máxima de saída do PLL é de 40 MHz. Portanto, para trabalhar com a frequência máxima devemos ter um cristal ou ressonador de 10 MHz.

Já a seleção do tipo de oscilador pode ser feita em tempo de execução do programa, como mostrado na tabela seguinte:

OSCCON<SCS1:SCS0>	Fonte de clock
00	Oscilador primário
01	Oscilador secundário
1x	Oscilador interno

Os bits **OSCCON<OSTS>**, **OSCCON <IOFS>** e **T1CON <T1RUN>** indicam a fonte de clock do microcontrolador, sendo que:

- ✓ **OSCCON<OSTS>** monitora o oscilador primário. Quando setado, indica que o oscilador primário está funcionando.
- ✓ **OSCCON <IOFS>** monitora o oscilador INTOSC. Quando setado, indica que o oscilador INTOSC está funcionando.

- ✓ **T1CON <T1RUN>** monitora o oscilador do Timer1. Quando setado, indica que o oscilador do Timer1 está sendo utilizado como oscilador secundário.

O bit **OSCCON<IDLEN>** determina se o microcontrolador está no modo SLEEP ou no modo **IDLE**. Esse modo difere do SLEEP, no qual os periféricos e a CPU deixam de receber clock após a execução da instrução SLEEP. Já no modo IDLE somente os periféricos recebem clock após a execução da instrução SLEEP, e com isso qualquer periférico que gere interrupção nesse modo pode acordar o microcontrolador.

Fail Save Clock Monitor (FSCM)

É um recurso que monitora o clock do sistema. Ocorrendo algum problema no oscilador principal (oscilador externo), o microcontrolador continua funcionando e é gerada uma interrupção para que seja feita a troca da fonte de clock. No momento da falha, é trocado o oscilador para INTRC e o Watchdog Timer reinicia.

Dicas para o oscilador externo

Junto com o cristal ou ressonador devem ser colocados dois capacitores cerâmicos para terra. Porém, alguns modelos de ressonadores já incorporam esses capacitores internamente. Como regra geral, podemos dizer que ressonadores de três pinos já possuem os capacitores.

O tipo de cristal deve ser configurado antes de o PIC ser gravado. A tabela a seguir mostra o tipo de configuração do oscilador em função da frequência de trabalho, além do valor sugerido para os capacitores.

Tipo de Oscilador	Frequência do Cristal	Capacitor
LP	32 kHz	33 pF
	200 kHz	15 pF
XT	200 kHz	47 – 68 pF
	1 MHz	15 pF
	4 MHz	15 pF
HS ou HSPLL	4 MHz	15 pF
	8 MHz	15 – 33 pF
	20 MHz	15 – 33 pF

Para o caso do ressonador deve ser acompanhada a seguinte tabela:

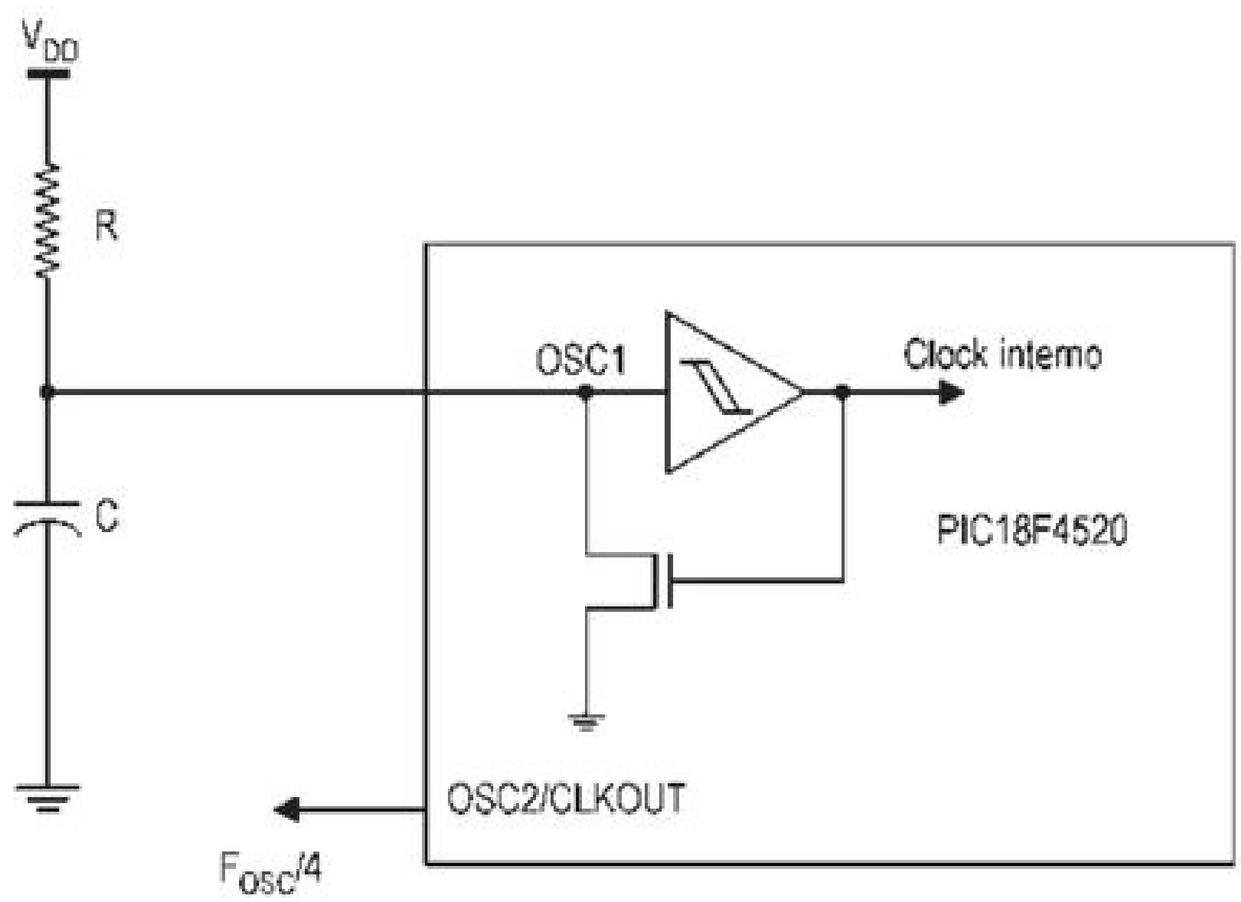
Tipo de Oscilador	Frequência do Ressonador	Capacitor
XT	455 kHz	68 – 100 pF
	2 MHz	15 – 68 pF
	4 MHz	15 – 68 pF
HS ou HSPLL	8 MHz	10 – 6 8pF
	16 MHz	10 – 22 pF

Para aplicações em que a precisão não é crítica, pode-se recorrer ao oscilador do tipo RC.

Recomenda-se utilizar resistor entre 3 kΩ e 100 kΩ. O capacitor deve ser maior do que 20 pF.

A frequência de trabalho do oscilador RC depende da tensão de alimentação e da temperatura ambiente. Por exemplo, com um resistor de 4,7 kΩ, um capacitor de 100 pF e uma temperatura de 25°C, o PIC deve trabalhar próximo a 1 MHz. O data sheet apresenta gráficos auxiliares para cálculo do RC.

Editora Érica - Desbravando o Microcontrolador PIC18 Recursos Avançados - Daniel R. de Sousa, David José de Souza e Nicolás César Lavinia - 1ª Edição



» Sistema de proteção do código (Code Protection)

Depois que seu sistema estiver pronto e um protótipo e/ou produto for entregue a terceiros, é muito importante que essa opção seja ativada, pois ela impede que qualquer pessoa consiga ter acesso ao programa gravado no PIC. Para o PIC18F4520, que é do tipo regravável (FLASH), o sistema de proteção impede a leitura, mas não a regravação. Entretanto, toda vez que uma nova gravação for feita quando o code protection estiver ativado, primeiramente toda a memória de programa será limpa, evitando assim que qualquer tipo de alteração indevida seja efetuado no software. Esse sistema de proteção é interno ao PIC, não dependendo dos programas externos utilizados para a gravação/leitura. Isso garante a sua eficiência.

Durante a configuração da gravação, existem diversas opções para proteção da memória de programa, considerando uma proteção total ou parcial.

» Registradores de identificação (IDs)

O PIC possui oito registradores especiais que se encontram nos endereços de 200000h a 200007h e não podem ser acessados diretamente pelo programa. Entretanto, esses registradores podem ser lidos e escritos por meio do sistema de gravação. Podemos utilizá-los, por exemplo, para a identificação da versão do programa gravado. A grande vantagem dos IDs é que podem ser lidos mesmo quando o Code Protection está ativado, possibilitando a rastreabilidade do software em casos de problemas futuros.

Essas posições podem ser lidas e escritas pelas instruções TBLRD e TBLWT ou na programação e verificação do programa durante a gravação do microcontrolador via programador.

O valor a ser gravado nos IDs pode ser escolhido diretamente durante as configurações de gravação. O sistema do MpLab possui incremento automático para a geração de número de série.

» Sistema de emulação In-Circuit (Debugger Mode)

O PIC18F4520 possui um sistema interno para operar com debugadores de baixo custo. Quando essa função está habilitada (pelas configurações de gravação), alguns recursos do PIC são perdidos:

Pinos de I/Os	RB6 e RB7 não operam mais como I/Os convencionais.
Pilha (Stack)	Perde-se um nível da pilha para o sistema de emulação.
Memória de programa	Não podem ser utilizados os endereços 7DC0h a 7FFFh.
Memória de dados	Não podem ser utilizados os endereços 5F4h a 5FFh.

Além dessas precauções, o sistema deve ainda disponibilizar um sistema de interligação (conector) para um emulador externo. A própria Microchip possui uma família de emuladores para operar com o sistema In-Circuit Debugger Mode chamado MpLab ICD. No Brasil, a LabTools possui uma ferramenta licenciada pela Microchip chamada ICD2BR. A conexão entre o ICD e o PIC é feita por cinco vias: MCLR/V_{PP}, V_{DD}, V_{SS}, RB6 e RB7. Mais detalhes sobre esse sistema podem ser adquiridos na documentação do MpLab ICD e no manual do ICD2BR.

» Proteção de escrita interna da FLASH

Esse sistema é conhecido como Flash Program Memory Write Enable e serve para proteger a E^2PROM interna, impedindo eventuais escritas.

» Gravação In-Circuit (ICSP)

O PIC18F4520, assim como a maioria dos modelos da Microchip, pode ser gravado diretamente nos circuitos, isto é, já montado na placa final. Essa gravação é conhecida como in-circuit e possibilita muito ganho de produtividade.

Como a gravação é feita de forma serial, utilizando somente cinco pinos, é possível a instalação de um soquete na placa para introduzir os sinais necessários à gravação. Obviamente uma série de cuidados especiais no projeto do hardware deve ser tomada a fim de que as condições de gravação não prejudiquem os demais circuitos na placa.

Está completamente fora do objetivo deste livro entrar nos detalhes do protocolo de gravação (a Microchip possui um data sheet específico para isso), mas um exemplo de aplicação é a própria placa McLab2, que possui um conector para gravação in-circuit por meio dos gravadores McFlash+ ou ICD2BR (LabTools).

Como pode ser observado no esquema elétrico dessa placa, os pinos necessários à gravação são:

- ✓ **V_{DD}**: alimentação convencional do PIC (5 V).
- ✓ **V_{SS}**: GND.
- ✓ **MCRL/V_{PP}**: tensão de programação. Para que o PIC entre em modo de programação, a tensão nesse pino deve ser elevada a aproximadamente 13 V. Por isso, o resto do circuito ligado a ele deve ser protegido.

- ✓ **RB6**: clock da comunicação serial imposto pelo gravador.
- ✓ **RB7**: dados da comunicação serial, que podem ser impostos pelo gravador (escrita) ou pelo próprio PIC (leitura).

» Gravação em baixa tensão (Low Voltage Programming)

Além do sistema de gravação in-circuit descrito anteriormente, o PIC18F4520 possui um segundo modo de gravação. No sistema convencional pudemos observar que a gravação só pode ser efetuada por intermédio de uma tensão especial em torno de 13 V. Isso impossibilita a gravação em nível TTL. Com o sistema de gravação em baixa tensão (LVP), o PIC pode ser colocado em modo de programação com apenas 5 V. Isso torna possível, por exemplo, que um PIC seja diretamente gravado por outro PIC.

O único problema é que, para poder utilizar o LVP, o pino RB5 é perdido como I/O convencional. Por isso, a habilitação desse sistema é feita por meio das configurações de gravação.

14

Implementação de um Sistema de Medição de Temperatura

» Introdução

Parabéns, você chegou ao final do estudo. Isso significa que a batalha foi ganha. Não deve ter sido muito fácil, mas esperamos que também não tenha sido tão difícil. O fato é que agora você possui muito mais conhecimento do que quando começou, e esse conhecimento será capaz de lançá-lo a um desafio muito maior, o desenvolvimento de projetos cada vez mais complexos. A batalha foi ganha, mas a luta está apenas começando.

Para ajudar nessa nova jornada, vamos apresentar a solução de um pequeno problema, utilizando os recursos da placa do Apêndice C (McLab2) que não foram vistos nos capítulos anteriores. A partir de agora, montaremos um pequeno sistema de medição de temperatura.

» O sistema

O monitoramento da temperatura é um dos problemas mais clássicos enfrentado pela maioria dos projetistas. Como nossa placa possui um sensor de temperatura (diodo) e dois atuadores: aquecimento (resistência) e resfriamento (ventilador), nada melhor do que implementarmos um sistema capaz de obter a temperatura atual para mostrá-la no LCD.

O sensor de temperatura

Para podermos obter a temperatura ambiente, usaremos um diodo. Como o diodo é um componente que apresenta queda de tensão proporcional à sua temperatura, vamos monitorar a tensão para encontrarmos a temperatura.

Para isso, o circuito eletrônico faz uso de um diodo de sinal convencional (1N4148) ligado a um amplificador e a uma porta analógica do PIC. Ligamos ao amplificador também um potenciômetro para poder alterar o offset da curva, ajustando assim a temperatura com uma referência externa.

Internamente, o sistema trabalha com uma conversão AD de 8 bits, gerando 256 possíveis valores de tensão para o diodo. Para cada valor obtido temos uma temperatura relacionada. A rotina TABELA_TEMP, que se encontra no final do código apresentado neste capítulo, efetua a conversão entre a tensão lida (unidades de AD) e a temperatura real. Nada mais é que uma tabela de conversão/linearização.

Editora Érica - Desbravando o Microcontrolador PIC18 Recursos Avançados - Daniel R. de Sousa, David José de Souza e Nicolás César Lavinia - 1ª Edição

Essa tabela foi construída com base na curva de resposta do diodo utilizado em função da temperatura. Caso seja construído um sensor de temperatura com outro tipo de componente que gere uma tensão variável, basta refazer a tabela de conversão.

Uma vez convertida, a temperatura é, então, mostrada no LCD, na unidade de °C (lado direito).

O aquecimento

Vamos possibilitar que o usuário aumente a temperatura sobre o diodo por meio do controle manual da resistência existente na placa. Isso é feito por intermédio de um dos PWMs do PIC, que se encontra ligado ao resistor.

Por intermédio dos botões S1 e S2 podemos aumentar e diminuir o duty cycle do PWM, variando de 0 a 100%. Mantendo os botões pressionados, o incremento/decremento é automático. O valor atual para o aquecimento é mostrado no LCD (lado esquerdo).

O resfriamento

Inversamente ao aquecimento, vamos possibilitar também o resfriamento do sistema por meio do ventilador, que é controlado pelo outro canal de PWM do PIC. Obviamente só seremos capazes de obter temperaturas levemente abaixo do valor ambiente, mas a intenção é poder criar variáveis diferentes de aquecimento e resfriamento.

Controlamos o ventilador pelo duty cycle do PWM, mas para o sistema ficar um pouco mais completo, mostramos no LCD (centro) o valor da rotação do motor, em RPS (rotações por segundo). Isso é feito através do sensor óptico que se encontra instalado na base das hélices do ventilador. Os botões S3 e S4 são usados para aumentar e diminuir o duty cycle do PWM, variando de 0 a 100%. Mantendo os botões pressionados, o incremento/decremento é automático.

Cada vez que uma das pás da hélice passa em frente ao sensor óptico, um pulso é transmitido ao PIC. Como esse sinal está ligado ao pino RC1, utilizamos o TMR1 com incremento externo para contabilizar a quantidade de pulsos gerados. A cada segundo (base de tempo gerada pela interrupção de TMR2), o total de pulsos é transferido para a variável CONT_VENT. Antes de mostrarmos o valor correto no LCD, devemos dividir o total de pulsos durante um segundo (CONT_VENT) pelo número de paletas (pulsos por volta). Neste caso CONT_VENT é dividido por sete.

Comunicação serial

O sistema possibilita ainda que maiores implementações sejam feitas com o tratamento externo da temperatura, pois os valores obtidos são enviados automaticamente pela porta serial, a cada segundo. Com isso é possível criar um pequeno programa de computador capaz de plotar essa temperatura no decorrer do tempo. Os dados também podem ser armazenados para cálculos ou consultas futuras. Use a imaginação.

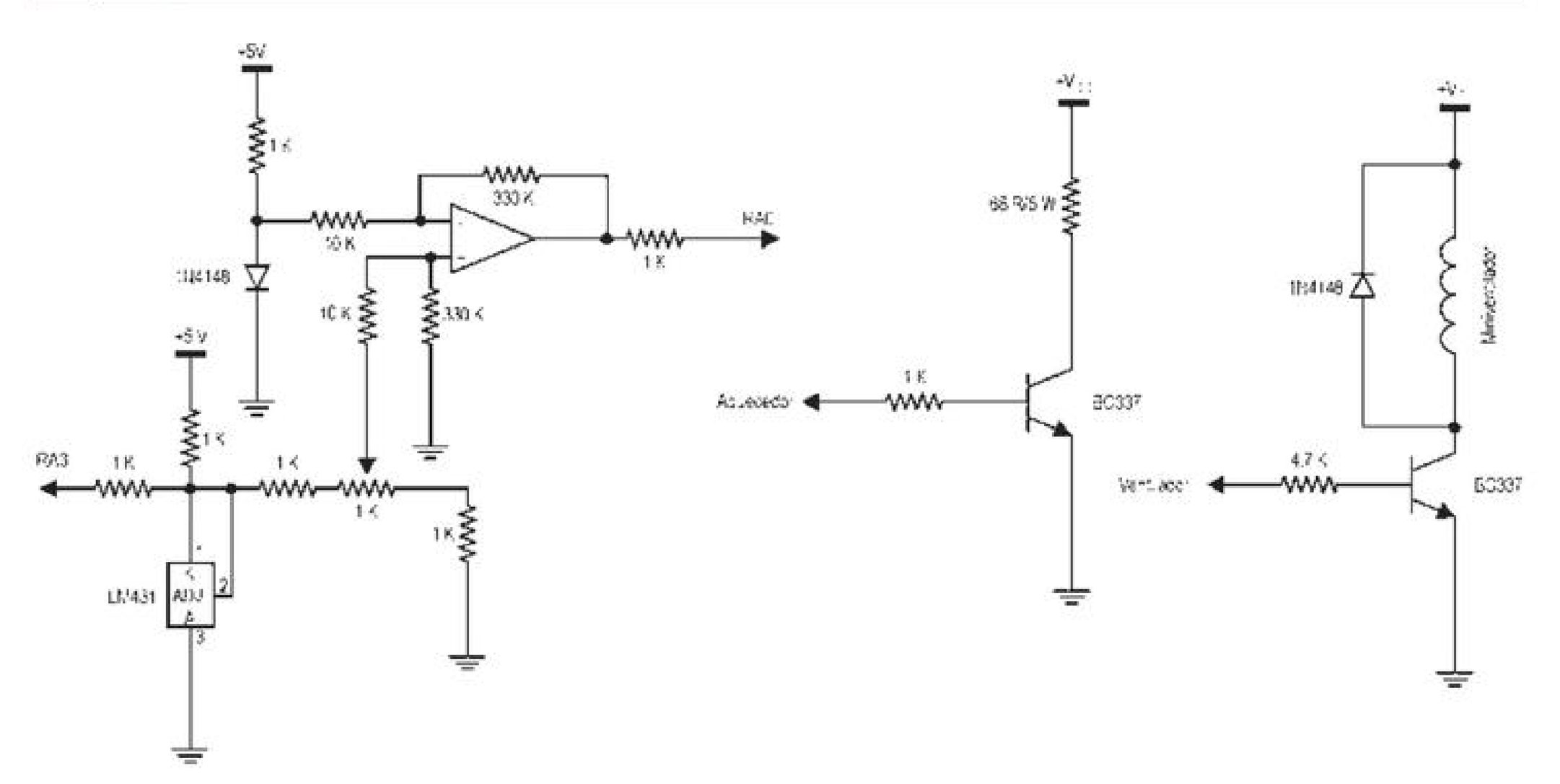
Os dados são transmitidos pela USART por meio do conector DB9, respeitando o padrão RS-232 com 8N1 e baud rate de 9.600 bps. A cada segundo é transmitido um byte com o valor da temperatura já convertido em °C.

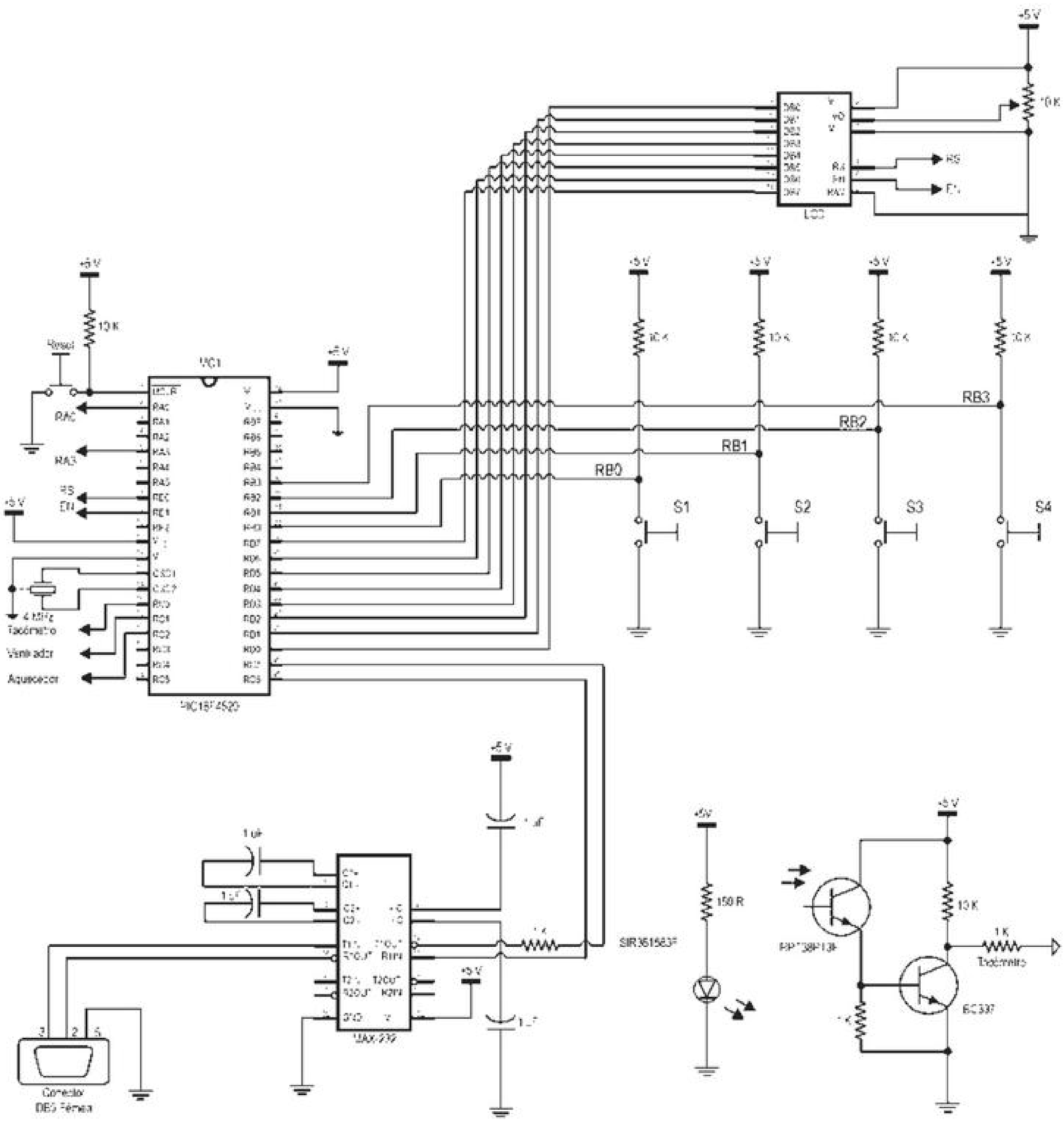
Considerações gerais

Ambos os PWMs são controlados pelo TMR2, que está regulado para 1 ms. Por isso, o período dos 2 PWMs é de 1 ms, ou seja, eles operam a 1 kHz.

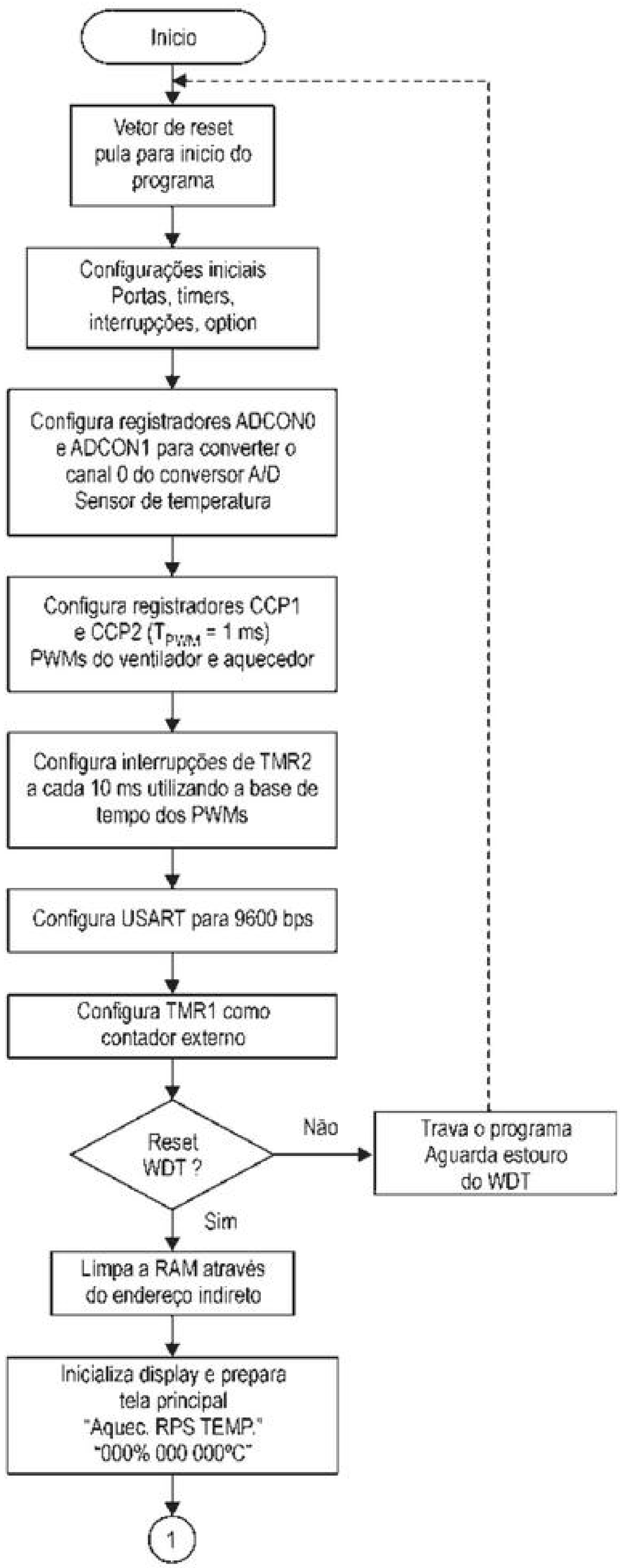
O postscale do TMR2 foi regulado em 1:10, gerando uma interrupção a cada 10 ms. Utilizamos um contador auxiliar (TEMPO_1S) para contabilizar cem interrupções, gerando a base de tempo de um segundo. Essa base é utilizada para capturar a rotação do ventilador, efetuar uma conversão de temperatura e transmitir dados pela USART.

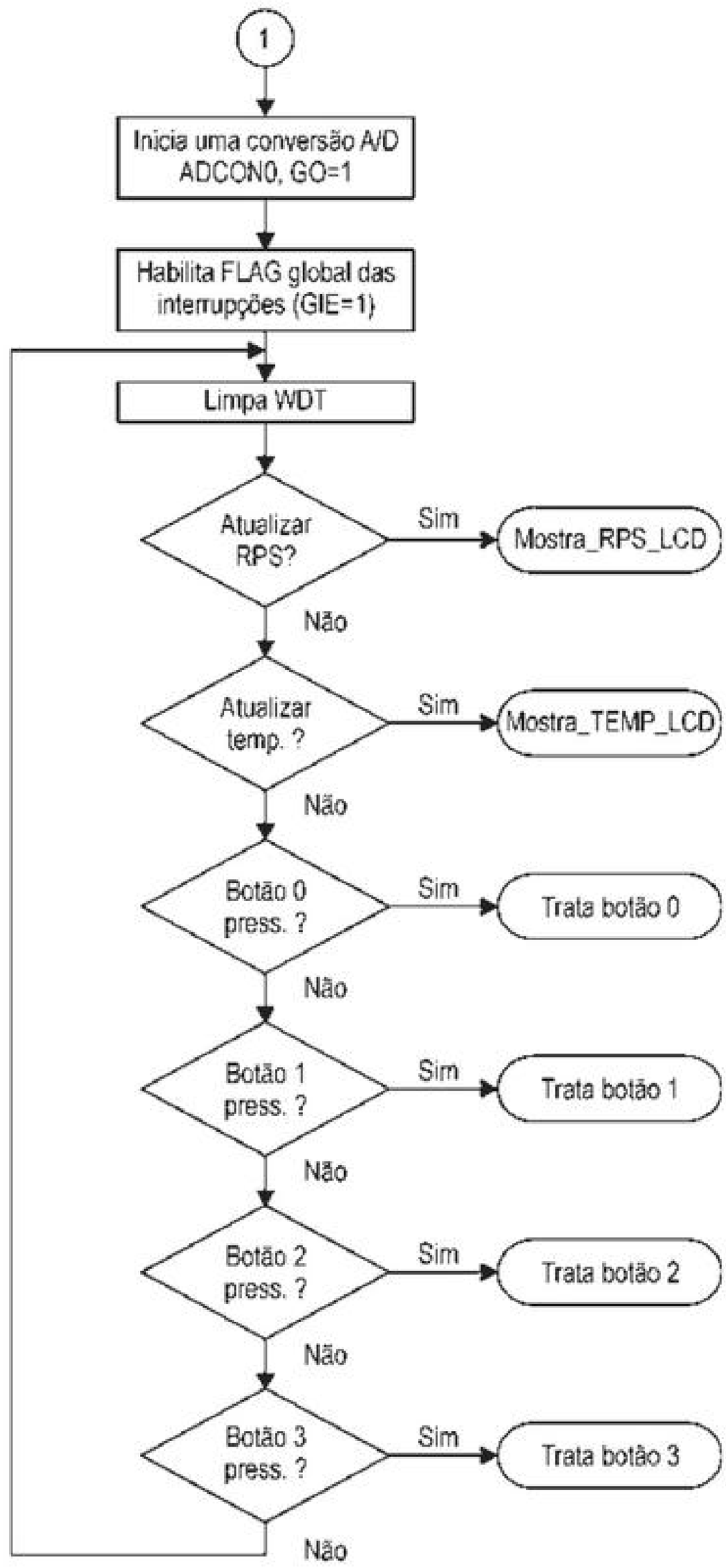
» Esquema elétrico

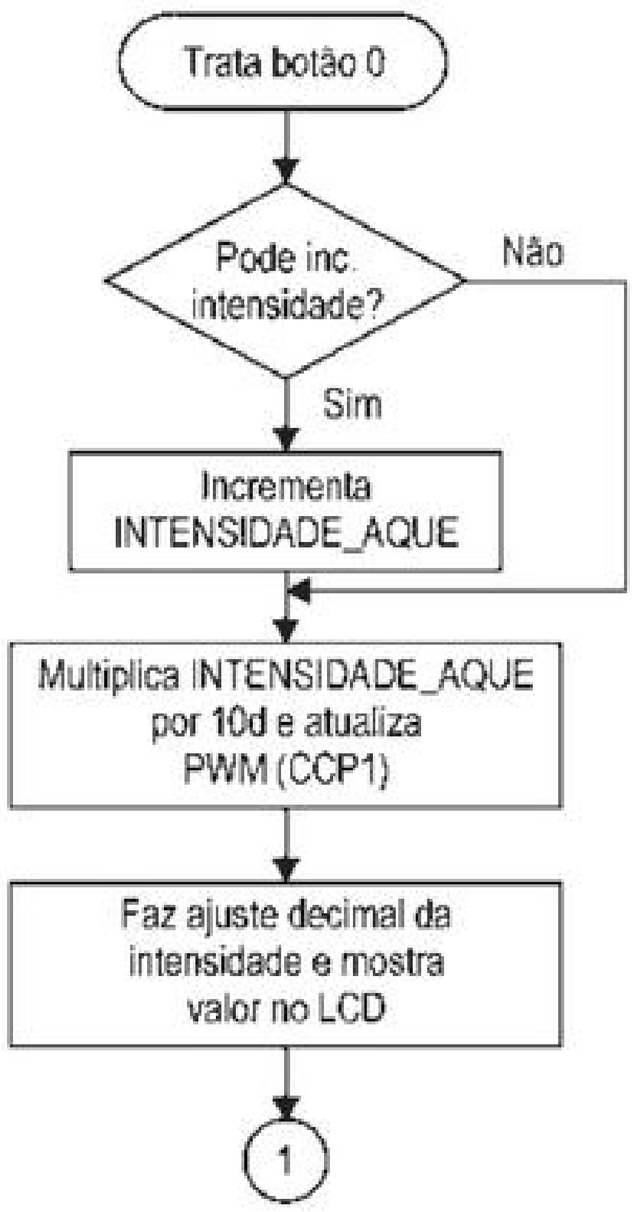


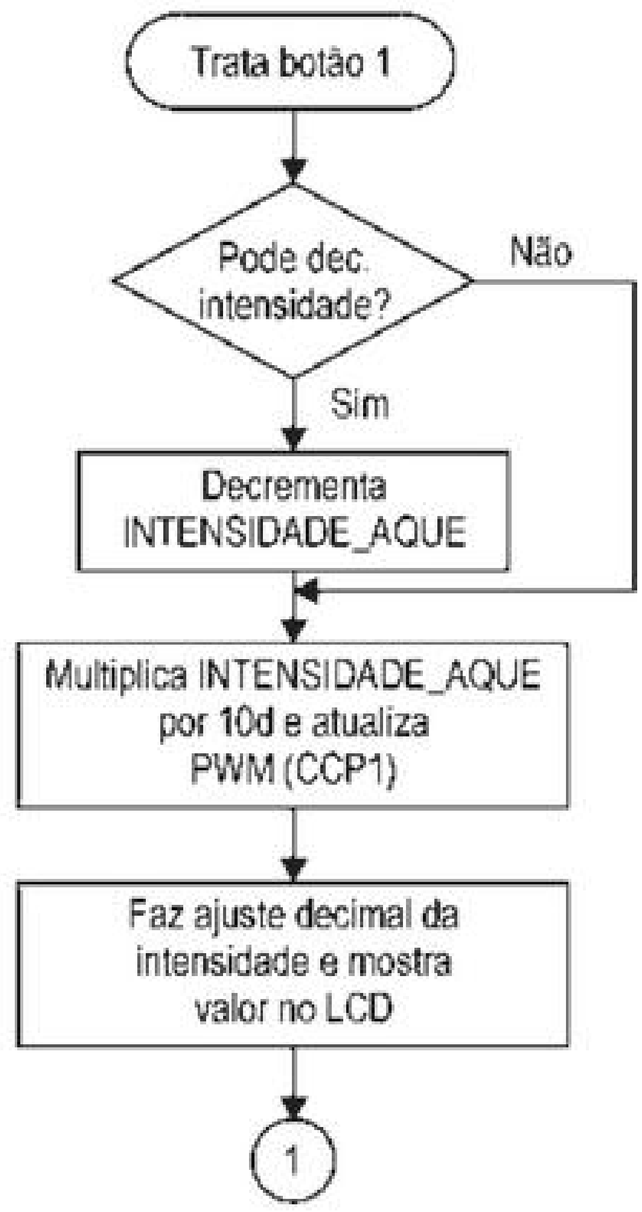

+5V
10K
Reset
RA0
RA3
RS
EN
Tacómetro
Ventilador
Aquecedor
PIC18F4520
RB0
RB1
RB2
RB3
S1
S2
S3
S4
LCD
MAX232
Conector
DB9 Fêmea
150 R
SIR361563F
BC337
1K
Tacómetro

» Fluxograma


Inicio
Vetor de reset pula para inicio do programa
Configurações iniciais Portas, timers, interrupções, option
Configura registradores ADCON0 e ADCON1 para converter o canal 0 do conversor A/D Sensor de temperatura
Configura registradores CCP1 e CCP2 (T_{PWM} = 1 ms) PWMs do ventilador e aquecedor
Configura interrupções de TMR2 a cada 10 ms utilizando a base de tempo dos PWMs
Configura USART para 9600 bps
Configura TMR1 como contador externo
Reset WDT ?
Não
Trava o programa Aguarda estouro do WDT
Sim
Limpa a RAM através do endereço indireto
Inicializa display e prepara tela principal "Aquec. RPS TEMP." "000% 000 000ºC"
1

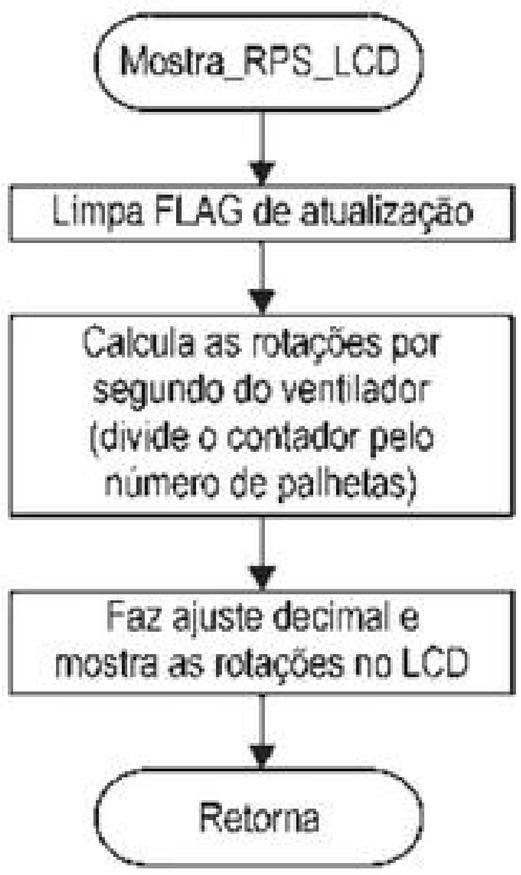

1
Inicia uma conversão A/D
ADCON0, GO=1
Habilita FLAG global das
interrupções (GIE=1)
Limpa WDT
Atualizar
RPS?
Sim
Mostra_RPS_LCD
Não
Atualizar
temp. ?
Sim
Mostra_TEMP_LCD
Não
Botão 0
press. ?
Sim
Trata botão 0
Não
Botão 1
press. ?
Sim
Trata botão 1
Não
Botão 2
press. ?
Sim
Trata botão 2
Não
Botão 3
press. ?
Sim
Trata botão 3
Não

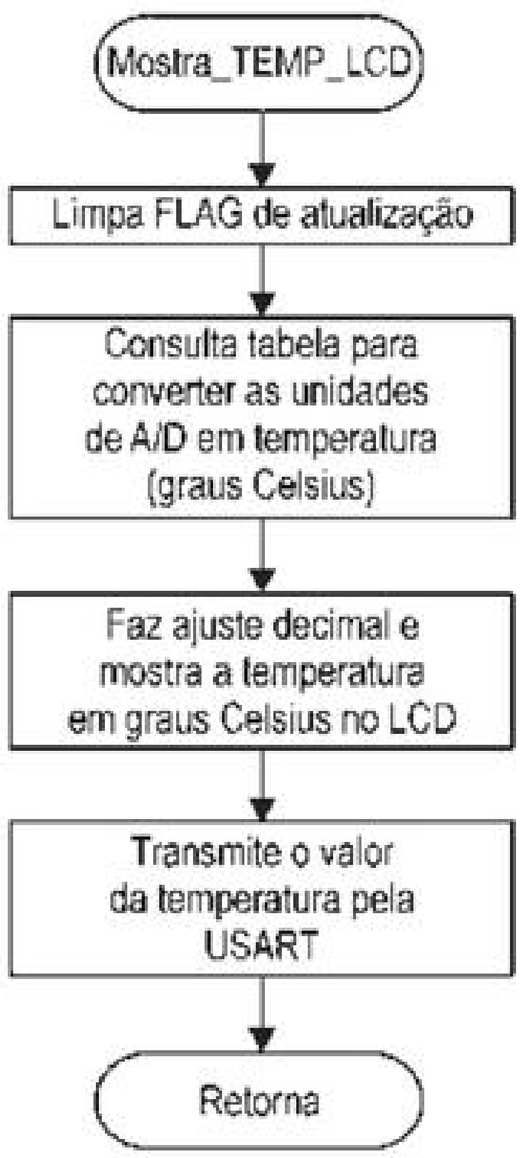

Trata botão 0
Pode inc. intensidade?
Não
Sim
Incrementa INTENSIDADE_AQUE
Multiplica INTENSIDADE_AQUE por 10d e atualiza PWM (CCP1)
Faz ajuste decimal da intensidade e mostra valor no LCD
1


Trata botão 1
Pode dec. intensidade?
Não
Sim
Decrementa INTENSIDADE_AQUE
Multiplica INTENSIDADE_AQUE por 10d e atualiza PWM (CCP1)
Faz ajuste decimal da intensidade e mostra valor no LCD
1

Trata botão 2
Pode inc. intensidade?
Não
Sim
Incrementa INTENSIDADE_VENT
Multiplica INTENSIDADE_VENT por 10d e atualiza PWM (CCP2)
1

Trata botão 3
Pode dec. intensidade?
Não
Sim
Decrementa INTENSIDADE_VENT
Multiplica INTENSIDADE_VENT por 10d e atualiza PWM (CCP2)
1


Mostra_RPS_LCD
Limpa FLAG de atualização
Calcula as rotações por segundo do ventilador (divide o contador pelo número de palhetas)
Faz ajuste decimal e mostra as rotações no LCD
Retorna


Mostra_TEMP_LCD
Limpa FLAG de atualização
Consulta tabela para converter as unidades de A/D em temperatura (graus Celsius)
Faz ajuste decimal e mostra a temperatura em graus Celsius no LCD
Transmite o valor da temperatura pela USART
Retorna

Interrupção
Salva contexto
Limpa FLAG da interrupção
Passou 1 segundo ?
Não
Sim
Recarrega contador de 1 segundo
Salva o valor do contador do TMR1 em CONT_VENT
Zera contadores do TMR1
Seta FLAG para atualizar LCD
Salva o valor da conversão A/D em TEMPERATURA
Inicia uma nova conversão A/D
Seta FLAG para atualizar LCD
Recupera contexto
Retorna

» Código

```
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *              DESBRAVANDO O PIC18 - RECURSOS AVANÇADOS                   *
; *                              EXEMPLO 10                                 *
; *                                                                         *
; * DANIEL RODRIGUES DE SOUSA, DAVID JOSÉ DE SOUZA e  NICOLÁS CÉSAR LAVINIA *
; *                                                                         *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *     DATA : 03/08/2009                                                   *
; *   VERSÃO : 1.0                                                          *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                            DESCRIÇÃO GERAL                              *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTE EXEMPLO FOI ELABORADO PARA EXPLICAR O FUNCIONAMENTO DO TMR1 COMO
;  CONTADOR, UTILIZADO NA PLACA MCLAB2 PARA CONTAR AS ROTAÇÕES DO VENTILADOR.
;  O SOFTWARE CONVERTE O CANAL 0 DO CONVERSOR AD (SENSOR DE TEMPERATURA).
;  DOIS PWMs FORAM UTILIZADOS, UM PARA MODULAR A RESISTÊNCIA DE AQUECIMENTO
;  E OUTRO PARA A VELOCIDADE DO VENTILADOR.
;  COM AS TECLAS S1 E S2 PODE-SE VARIAR O PWM DO AQUECEDOR E COM AS TECLAS
;  S3 E S4 O PWM DO VENTILADOR.
;  NO LCD SÃO MOSTRADOS OS VALORES DO PWM DO AQUECEDOR, O NÚMERO DE ROTAÇÕES
;  POR SEGUNDO DO VENTILADOR E A TEMPERATURA DO DIODO JÁ CONVERTIDA EM GRAUS
;  CELSIUS. ALÉM DISSO, O VALOR ATUAL DA TEMPERATURA DO DIODO É TRANSMITIDO
;  PERIODICAMENTE ATRAVÉS DA USART.
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                       CONFIGURAÇÕES PARA GRAVAÇÃO                       *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
    CONFIG  OSC = XT, FCMEN = OFF, IESO = OFF, PWRT = ON, BOREN = ON, BORV = 0
    CONFIG  WDT = ON, WDTPS = 128, MCLRE = ON, LPT1OSC = OFF, PBADEN = OFF
    CONFIG  CCP2MX = PORTC, STVREN = ON, LVP = OFF, DEBUG = OFF, XINST = OFF
    CONFIG  CP0 = OFF, CP1 = OFF, CP2 = OFF, CP3 = OFF, CPB = OFF, CPD = OFF
    CONFIG  WRT0 = OFF, WRT1 = OFF, WRT2 = OFF, WRT3 = OFF, WRTB = OFF
    CONFIG  WRTC = OFF, WRTD = OFF, EBTR0 = OFF, EBTR1 = OFF, EBTR2 = OFF
    CONFIG  EBTR3 = OFF, EBTRB = OFF

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                          DEFINIÇÃO DAS VARIÁVEIS                        *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTE BLOCO DE VARIÁVEIS ESTÁ LOCALIZADO LOGO NO INÍCIO DO BANCO 0

    CBLOCK  0X00            ; BANK 0 --> 0X00..0X5F --> ACCESS BANK

        TEMPO0
        TEMPO1              ; TMEPORIZADORES P/ ROTINA DE DELAY

        AUX                 ; REGISTRADOR AUXILIAR DE USO GERAL

        UNIDADE             ; ARMAZENA VALOR DA UNIDADE
        DEZENA              ; ARMAZENA VALOR DA DEZENA
        CENTENA             ; ARMAZENA VALOR DA CENTENA

        INTENSIDADE_VENT    ; INTENSIDADE DO VENTILADOR
        INTENSIDADE_AQUE    ; INTENSIDADE DO AQUECEDOR

        ACCaHI              ; ACUMULADOR a DE 16 BITS UTILIZADO
        ACCaLO              ; NA ROTINA DE DIVISÃO
```

```
        ACCbHI                  ; ACUMULADOR b DE 16 BITS UTILIZADO
        ACCbLO                  ; NA ROTINA DE DIVISÃO

        ACCcHI                  ; ACUMULADOR c DE 16 BITS UTILIZADO
        ACCcLO                  ; NA ROTINA DE DIVISÃO

        ACCdHI                  ; ACUMULADOR d DE 16 BITS UTILIZADO
        ACCdLO                  ; NA ROTINA DE DIVISÃO

        temp                    ; CONTADOR TEMPORÁRIO UTILIZADO
                                ; NA ROTINA DE DIVISÃO
        TEMPERATURA             ; TEMPERATURA DO DIODO EM UNIDADES DE AD
        TEMP_CELSIUS            ; TEMPERATURA DO DIODO JÁ CONVERTIDA
                                ; EM GRAUS CELSIUS
        FILTRO_BOTOES           ; FILTRO P/ DEBOUNCE DOS BOTOÕES
        TEMPO_TURBO             ; TEMPORIZADOR P/ TUBO DO TECLADO

        TEMPO_1S                ; TEMPORIZADOR DE 1 SEGUNDO

        CONT_VENT_HIGH
        CONT_VENT_LOW           ; CONTADORES PARA ROTAÇÃO DO VENTILADOR

        FLAG                    ; FLAG DE USO GERAL

    ENDC

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                DEFINIÇÃO DAS VARIÁVEIS INTERNAS DO PIC                *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  O ARQUIVO DE DEFINIÇÕES DO PIC UTILIZADO DEVE SER REFERENCIADO PARA QUE
;  OS NOMES DEFINIDOS PELA MICROCHIP POSSAM SER UTILIZADOS, SEM A NECESSIDADE
;  DE REDIGITAÇÃO.

    #INCLUDE <P18F4520.INC>  ; MICROCONTROLADOR UTILIZADO

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                            CONSTANTES INTERNAS                        *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  A DEFINIÇÃO DE CONSTANTES FACILITA A PROGRAMAÇÃO E A MANUTENÇÃO.

FILTRO_TECLA    EQU .200    ; FILTRO P/ EVITAR RUIDOS DOS BOTÕES

TURBO_TECLA     EQU .70     ; TEMPORIZADOR P/ TURBO DO TECLADO

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                    DECLARAÇÃO DOS FLAGs DE SOFTWARE                   *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  A DEFINIÇÃO DE FLAGs AJUDA NA PROGRAMAÇÃO E ECONOMIZA MEMÓRIA RAM.

#DEFINE MOSTRA_RPS  FLAG,0  ; FLAG PARA MOSTRAR A ROTAÇÃO NO LCD
                            ; 1 -> DEVE MOSTRAR A ROTAÇÃO
                            ; 0 -> NAO DEVE MOSTRAR A ROTAÇÃO

#DEFINE MOSTRA_TEMP FLAG,1  ; FLAG PARA MOSTRAR A TEMPERATURA NO LCD
                            ; 1 -> DEVE MOSTRAR A TEMPERATURA
                            ; 0 -> NAO DEVE MOSTRAR A TEMPERATURA

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                                 ENTRADAS                              *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  AS ENTRADAS DEVEM SER ASSOCIADAS A NOMES PARA FACILITAR A PROGRAMAÇÃO E
;  FUTURAS ALTERAÇÕES DO HARDWARE.
```

```
#DEFINE BOTAO_0   PORTB,0            ; ESTADO DO BOTÃO 0
                                     ; 1 -> LIBERADO
                                     ; 0 -> PRESSIONADO

#DEFINE BOTAO_1   PORTB,1            ; ESTADO DO BOTÃO 1
                                     ; 1 -> LIBERADO
                                     ; 0 -> PRESSIONADO

#DEFINE BOTAO_2   PORTB,2            ; ESTADO DO BOTÃO 2
                                     ; 1 -> LIBERADO
                                     ; 0 -> PRESSIONADO

#DEFINE BOTAO_3   PORTB,3            ; ESTADO DO BOTÃO 3
                                     ; 1 -> LIBERADO
                                     ; 0 -> PRESSIONADO

;  ESTE PROGRAMA UTILIZA UMA ENTRADA P/ O CONVERSOR AD.
;  ESTA ENTRADA NÃO PRECISA SER DECLARADA, POIS O SOFTWARE NUNCA FAZ
;  REFERÊNCIA A ELA DE FORMA DIRETA, POIS O CANAL AD A SER CONVERTIDO É
;  SELECIONADO NO REGISTRADOR ADCON0 DE FORMA BINÁRIA E NÃO ATRAVÉS DE
;  DEFINES. PORÉM PARA FACILITAR O ENTENDIMENTO DO HARDWARE VAMOS DECLARAR
;  ESTA ENTRADA NORMALMENTE.

#DEFINE CAD_TEMP    PORTA,0          ; ENTRADA AD PARA TEMPERATURA

;  ALÉM DA ENTRADA DO CONVERSOR AD, TEMOS A ENTRADA DA USART (RECEPÇÃO).
;  NOVAMENTE ESTA ENTRADA NÃO NECESSITA SER DECLARADA, PORÉM, PARA
;  FACILITAR O ENTENDIMENTO DO HARDWARE VAMOS DECLARAR ESTA ENTRADA
;  NORMALMENTE.

#DEFINE RXUSART     PORTC,7          ; ENTRADA DE RX DA USART

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                                SAÍDAS                                *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  AS SAÍDAS DEVEM SER ASSOCIADAS A NOMES PARA FACILITAR A PROGRAMAÇÃO E
;  FUTURAS ALTERAÇÕES DO HARDWARE.

#DEFINE DISPLAY     LATD             ; BARRAMENTO DE DADOS DO DISPLAY

#DEFINE RS          LATE,0           ; INDICA P/ O DISPLAY UM DADO OU COMANDO
                                     ; 1 -> DADO
                                     ; 0 -> COMANDO

#DEFINE ENABLE      LATE,1           ; SINAL DE ENABLE P/ DISPLAY
                                     ; ATIVO NA BORDA DE DESCIDA

;  TEMOS TAMBÉM AS SAÍDAS DE TX DA USART, PWM1 E PWM2.
;  PARA FACILITAR O ENTENDIMENTO DO HARDWARE VAMOS DECLARAR ESTAS SAÍDAS
;  NORMALMENTE APESAR DE NÃO SEREM UTILIZADAS.

#DEFINE TXUSART     PORTC,6          ; SAÍDA DE TX DA USART

#DEFINE VENTILADOR  PORTC,1          ; SAÍDA P/ VENTILADOR

#DEFINE AQUECEDOR   PORTC,2          ; SAÍDA P/ AQUECEDOR

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                   VETOR DE RESET DO MICROCONTROLADOR                 *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  POSIÇÃO INICIAL PARA EXECUÇÃO DO PROGRAMA

    ORG     0X0000                   ; ENDEREÇO DO VETOR DE RESET
```

```
    BRA     CONFIGURACAO            ; PULA PARA CONFIG DEVIDO A REGIÃO
                                    ; DESTINADA AS ROTINAS SEGUINTES

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                          VETOR DE INTERRUPÇÃO                         *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  POSIÇÃO DE DESVIO DO PROGRAMA QUANDO UMA INTERRUPÇÃO ACONTECE

    ORG     0x0008                  ; ENDEREÇO DO VETOR DE INTERRUPÇÃO

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                TRATAMENTO DA INTERRUPÇÃO DE TIMER 2                   *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; A INTERRUPÇÃO DE TMR2 É UTILIZADA PARA FORNECER A BASE DE TEMPO PARA AS
; MEDIDAS DAS ROTAÇÕES POR SEGUNDO DO VENTILADOR E DA TEMPERATURA DO DIODO.
; ALÉM DISSO, ELA SETA OS FLAGS PARA QUE ELES SEJAM ATUALIZADOS NO LCD.
; O TMR2 ESTÁ CONFIGURADO PARA POSTSCALE DE 1:10 E PORTANTO A CADA 10ms A
; INTERRUPÇÃO É GERADA.
; O CONVERSOR AD É LIDO A CADA INTERRUPÇÃO, OU SEJA, A CADA 10ms.
; A CADA 100 INTERRUPÇÕES, OU SEJA, A CADA 1 SEGUNDO, O VALOR DO CONTADOR DO
; TMR1 É SALVO NA VARIÁVEL CONT_VENT (HIGH E LOW), DESTA FORMA, O VALOR DE
; CONT_VENT É O NÚMERO DE ROTAÇÕES DO VENTILADOR POR SEGUNDO. NA VERDADE ESTE
; VALOR ENCONTRA-SE MULTIPLICADO PELO NÚMERO DE PALHETAS DO VENTILADOR.

INT_TMR2
    BCF     PIR1,TMR2IF             ; LIMPA FLAG DA INTERRUPÇÃO

    DECFSZ  TEMPO_1S,F              ; FIM DO 1 SEGUNDO ?
    BRA     INT_TMR2_2              ; NÃO - PULA P/ INT_TMR2_2
                                    ; SIM
    MOVLW   .100
    MOVWF   TEMPO_1S                ; RECARREGA TEMPORIZADOR DE 1 SEGUNDO

    BCF     T1CON,TMR1ON            ; PARALIZA CONTADOR DO TMR1
    MOVFF   TMR1H,CONT_VENT_HIGH
    MOVFF   TMR1L,CONT_VENT_LOW     ; SALVA VALOR DO TMR1 EM CONT_VENT

    CLRF    TMR1H
    CLRF    TMR1L                   ; RESETA CONTADORES

    BSF     T1CON,TMR1ON            ; LIBERA CONTADORES DO TMR1
    BSF     MOSTRA_RPS              ; SETA FLAG P/ MOSTRAR VALOR
                                    ; DAS RPS DO VENTILADOR
INT_TMR2_2
    MOVFF   ADRESH,TEMPERATURA      ; SALVA VALOR DA CONVERSÃO AD
                                    ; NA VARIÁVEL TEMPERATURA
    BSF     ADCON0,GO               ; INICIA UMA NOVA CONVERSÃO
    BSF     MOSTRA_TEMP             ; SETA FLAG P/ ATUALIZAR VALOR
                                    ; DA TEMPERATURA NO LCD
    RETFIE  FAST                    ; RETORNA DA INTERRUPÇÃO

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                           ROTINA DE DIVISÃO                           *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;*********************************************************************
;                        Double Precision Division
;*********************************************************************
;   Division : ACCb(16 bits) / ACCa(16 bits) -> ACCb(16 bits) with
;                                               Remainder in ACCc (16 bits)
;       (a) Load the Denominator in location ACCaHI & ACCaLO ( 16 bits )
;       (b) Load the Numerator in location ACCbHI & ACCbLO ( 16 bits )
;       (c) RCALL D_divF
;       (d) The 16 bit result is in location ACCbHI & ACCbLO
;       (e) The 16 bit Remainder is in locations ACCcHI & ACCcLO
```

```
;*****************************************************************************

D_divF
    MOVLW   .16
    MOVWF   temp            ; CARREGA CONTADOR PARA DIVISÃO

    MOVFF   ACCbHI,ACCdHI
    MOVFF   ACCbLO,ACCdLO   ; SALVA ACCb EM ACCd

    CLRF    ACCbHI
    CLRF    ACCbLO          ; LIMPA ACCb

    CLRF    ACCcHI
    CLRF    ACCcLO          ; LIMPA ACCc

DIV
    BCF     STATUS,C
    RLCF    ACCdLO,F
    RLCF    ACCdHI,F
    RLCF    ACCcLO,F
    RLCF    ACCcHI,F
    MOVF    ACCaHI,W
    SUBWF   ACCcHI,W        ;check if a>c
    BTFSS   STATUS,Z
    BRA     NOCHK
    MOVF    ACCaLO,W
    SUBWF   ACCcLO,W        ;if msb equal then check lsb
NOCHK
    BTFSS   STATUS,C        ;carry set if c>a
    BRA     NOGO
    MOVF    ACCaLO,W        ;c-a into c
    SUBWF   ACCcLO,F
    BTFSS   STATUS,C
    DECF    ACCcHI,F
    MOVF    ACCaHI,W
    SUBWF   ACCcHI,F
    BSF     STATUS,C        ;shift a 1 into b (result)
NOGO
    RLCF    ACCbLO,F
    RLCF    ACCbHI,F

    DECFSZ  temp,F          ; FIM DA DIVISÃO ?
    BRA     DIV             ; NÃO - VOLTA P/ DIV
    RETURN                  ; SIM - RETORNA

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                    ROTINA DE DELAY (DE 1MS ATÉ 256MS)                  *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTA É UMA ROTINA DE DELAY VARIÁVEL, COM DURAÇÃO DE 1MS X O VALOR PASSADO
;  EM WORK (W).

DELAY_MS
    MOVWF   TEMPO1          ; CARREGA TEMPO1 (UNIDADES DE MS)
    MOVLW   .250
    MOVWF   TEMPO0          ; CARREGA TEMPO0 (P/ CONTAR 1MS)

    CLRWDT                  ; LIMPA WDT (PERDE TEMPO)
    DECFSZ  TEMPO0,F        ; FIM DE TEMPO0 ?
    BRA     $-.4            ; NÃO - VOLTA 2 INSTRUÇÕES
                            ; SIM - PASSOU-SE 1ms
    DECFSZ  TEMPO1,F        ; FIM DE TEMPO1 ?
    BRA     $-.12           ; NÃO - VOLTA 6 INSTRUÇÕES
    RETURN                  ; SIM - RETORNA
```

```
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                ROTINA DE ESCRITA DE UM CARACTERE NO DISPLAY              *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTA ROTINA ENVIA UM CARACTERE PARA O MÓDULO DE LCD. O CARACTERE A SER
;  ESCRITO DEVE SER COLOCADO EM WORK (W) ANTES DE CHAMAR A ROTINA.

ESCREVE
    MOVWF   DISPLAY         ; ATUALIZA DISPLAY (PORTD)
    NOP                     ; PERDE 1US PARA ESTABILIZAÇÃO
    BSF     ENABLE          ; ENVIA UM PULSO DE ENABLE AO DISPLAY
    BRA     $+.2            ; .
    BCF     ENABLE          ; .

    MOVLW   .1
    RCALL   DELAY_MS        ; DELAY DE 1ms
    RETURN                  ; RETORNA

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                              AJUSTE DECIMAL                             *
; *          W [HEX] =  CENTENA [DEC] : DEZENA [DEC] ; UNIDADE [DEC]        *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTA ROTINA RECEBE UM ARGUMENTO PASSADO PELO WORK E RETORNA NAS VARIÁVEIS
;  CENTENA, DEZENA E UNIDADE O NÚMERO BCD CORRESPONDENTE AO PARÂMETRO PASSADO.

AJUSTE_DECIMAL
    MOVWF   AUX             ; SALVA VALOR A CONVERTER EM AUX
    CLRF    UNIDADE
    CLRF    DEZENA          ; RESETA REGISTRADORES
    CLRF    CENTENA

    MOVF    AUX,F           ; VALOR A CONVERTER = 0 ?
    BZ      $+.28           ; SIM - RETORNA
    INCF    UNIDADE,F       ; NÃO - INCREMENTA UNIDADE

    MOVF    UNIDADE,W
    XORLW   0X0A            ; UNIDADE = 10d ?
    BNZ     $+.6            ; NÃO
    CLRF    UNIDADE         ; SIM - RESETA UNIDADE
    INCF    DEZENA,F        ; INCREMENTA DEZENA

    MOVF    DEZENA,W
    XORLW   0X0A            ; DEZENA = 10d ?
    BNZ     $+.6            ; NÃO
    CLRF    DEZENA          ; SIM - RESETA DEZENA
    INCF    CENTENA,F       ; INCREMENTA CENTENA

    DECFSZ  AUX,F           ; FIM DA CONVERSÃO ?

    BRA     $-.24           ; NÃO - VOLTA P/ CONTINUAR CONVERSÃO
    RETURN                  ; SIM

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *            ROTINA PARA MOSTRAR A ROTAÇÃO DO VENTILADOR NO LCD          *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; ESTA ROTINA ATUALIZA O VALOR DAS ROTAÇÕES POR SEGUNDO DO VENTILADOR NO LCD.

MOSTRA_RPS_LCD
    BCF     MOSTRA_RPS      ; LIMPA FLAG DE ATUALIZAÇÃO DA RPS

    BCF     RS              ; SELECIONA O DISPLAY P/ COMANDOS
    MOVLW   0XC7            ; COMANDO PARA POSICIONAR O CURSOR
    RCALL   ESCREVE         ; LINHA 1 / COLUNA 7
    BSF     RS              ; SELECIONA O DISPLAY P/ DADOS
```

```
MOVFF   CONT_VENT_HIGH,ACCbHI   ; CARREGA ACCb COM VALOR DO CONTADOR
MOVFF   CONT_VENT_LOW,ACCbLO

CLRF    ACCaHI
MOVLW   .7                  ; CARREGA ACCa COM NÚMERO DE PALHETAS
MOVWF   ACCaLO              ; DO VENTILADOR

RCALL   D_divF              ; CHAMA ROTINA DE DIVISÃO

MOVF    ACCbLO,W
RCALL   AJUSTE_DECIMAL      ; FAZ O AJUSTE DECIMAL DO RESULTADO
                            ; (ROTAÇÕES POR SEGUNDO)
MOVF    CENTENA,W
ADDLW   0X30                ; CONVERTE CENTENA EM ASCII
RCALL   ESCREVE             ; ESCREVE VALOR NO LCD

MOVF    DEZENA,W
ADDLW   0X30                ; CONVERTE DEZENA EM ASCII
RCALL   ESCREVE             ; ESCREVE VALOR NO LCD

MOVF    UNIDADE,W
ADDLW   0X30                ; CONVERTE UNIDADE EM ASCII
RCALL   ESCREVE             ; ESCREVE VALOR NO LCD
RETURN                      ; RETORNA

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                  ROTINA PARA MOSTRAR A TEMPERATURA NO LCD           *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; ESTA ROTINA CONSULTA UMA TABELA P/ CONVERTER O VALOR DO CANAL AD DO SENSOR
; DE TEMPERATURA EM GRAUS CELSIUS, MOSTRA-O NO LCD E TRANSMITE PELA USART.

MOSTRA_TEMP_LCD
    BCF     MOSTRA_TEMP         ; LIMPA FLAG DE ATUALIZAÇÃO DA TEMP.

    RCALL   TABELA_TEMPERATURA  ; CONVERTE AD EM GRAUS CELSIUS
    MOVWF   TEMP_CELSIUS        ; SALVA VALOR EM TEMP_CELSIUS
    RCALL   AJUSTE_DECIMAL      ; FAZ O AJUSTE DECIMAL

    BCF     RS                  ; SELECIONA O DISPLAY P/ COMANDOS
    MOVLW   0XCB                ; COMANDO PARA POSICIONAR O CURSOR
    RCALL   ESCREVE             ; LINHA 1 / COLUNA 11
    BSF     RS                  ; SELECIONA O DISPLAY P/ DADOS

    MOVF    CENTENA,W
    ADDLW   0X30                ; CONVERTE CENTENA EM ASCII
    RCALL   ESCREVE             ; ESCREVE VALOR NO LCD

    MOVF    DEZENA,W
    ADDLW   0X30                ; CONVERTE DEZENA EM ASCII
    RCALL   ESCREVE             ; ESCREVE VALOR NO LCD

    MOVF    UNIDADE,W
    ADDLW   0X30                ; CONVERTE UNIDADE EM ASCII
    RCALL   ESCREVE             ; ESCREVE VALOR NO LCD

    BTFSS   TXSTA,TRMT          ; O BUFFER DE TX ESTÁ VAZIO ?
    BRA     $-.2                ; NÃO - AGUARDA ESVAZIAR

    MOVFF   TEMP_CELSIUS,TXREG  ; TRANSMITE A TEMPERATURA EM GRAUS
                                ; CELSIUS PELA USART
    RETURN                      ; RETORNA
```

```
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                CONFIGURAÇÕES INICIAIS DE HARDWARE E SOFTWARE            *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  NESTA ROTINA SÃO INICIALIZADAS AS PORTAS DE I/O DO MICROCONTROLADOR E AS
;  CONFIGURAÇÕES DOS REGISTRADORES ESPECIAIS (SFR). A ROTINA INICIALIZA A
;  MÁQUINA E AGUARDA O ESTOURO DO WDT.

CONFIGURACAO
    CLRF    PORTA           ; GARANTE TODAS AS SAÍDAS EM ZERO
    CLRF    PORTB
    CLRF    PORTC
    CLRF    PORTD
    CLRF    PORTE

    CLRF    LATA
    CLRF    LATB
    CLRF    LATC
    CLRF    LATD
    CLRF    LATE

    MOVLW   B'11111111'
    MOVWF   TRISA           ; CONFIGURA I/O DO PORTA

    MOVLW   B'11111111'
    MOVWF   TRISB           ; CONFIGURA I/O DO PORTB

    MOVLW   B'10111001'
    MOVWF   TRISC           ; CONFIGURA I/O DO PORTC

    MOVLW   B'00000000'
    MOVWF   TRISD           ; CONFIGURA I/O DO PORTD

    MOVLW   B'00000100'
    MOVWF   TRISE           ; CONFIGURA I/O DO PORTE

    MOVLW   B'11011000'     ; CONFIGURA TIMER0
    MOVWF   T0CON           ; TIMER0 INCREM. PELO CICLO DE MÁQUINA
                            ; TIMER0- 1:1

    MOVLW   B'01000000'     ; CONFIGURA INTERRUPÇÕES
    MOVWF   INTCON          ; HABILITA INTER. DE PERIFÉRICOS

    MOVLW   B'10000000'
    MOVWF   RCON            ; ATIVA SISTEMA DE PRIORIDADE DAS INTERRUPÇÕES

    MOVLW   B'00000010'
    MOVWF   IPR1            ; ALTA PRIORIDADE PARA INT. DO TIMER2

    MOVLW   B'00000010'     ; CONFIGURA INTER. DE PERIFÉRICOS
    MOVWF   PIE1            ; HABILITA A INTERRUPÇÃO DE TMR2

    MOVLW   B'00000001'     ; CONFIGURA CONVERSOR AD
    MOVWF   ADCON2          ; VELOCIDADE -> Fosc/8
                            ; JUSTIFICADO À ESQUERDA
                            ; 8 BITS EM ADRESH E 2 BITS EM ADRESL
                            ; TEMPO DE AQUISIÇÃO DO CAPACITOR
                            ; DEFINIDO POR SOFTWARE
                            ; Vref+ = VDD (+5V)
                            ; Vref- = GND ( 0V)

    MOVLW   B'00001101'     ; CONFIGURA CONVERSOR AD
    MOVWF   ADCON1          ; RA0 E RA1 COMO ANALÓGICO
                            ; RA2, RA3, RA4 E RA5 COMO I/O DIGITAL
                            ; PORTB E PORTE COMO I/O DIGITAL
```

```
                                ; JUSTIFICADO À ESQUERDA
                                ; 8 BITS EM ADRESH E 2 BITS EM ADRESL
                                ; Vref+ = VDD (+5V)
                                ; Vref- = GND ( 0V)

MOVLW   B'00000001'             ; CONFIGURA CONVERSOR AD
MOVWF   ADCON0                  ; CANAL 0
                                ; MÓDULO LIGADO

MOVLW   .249                    ; CONFIGURA PERÍODO DO PWM
MOVWF   PR2                     ; T=((PR2)+1)*4*Tosc*TMR2 Prescale
                                ; T=((249)+1)*4*250ns*4
                                ; T=1,000ms -> 1.000Hz = 1KHz

    MOVLW   B'00000000'         ; AUTOBAUD DESABILITADO, WAKE-UP DESABILITADO
    MOVWF   BAUDCON             ; GERADOR DE BAUD RATE EM 8 BITS
                                ; PINO TX NORMAL EM 1, RECEBIMENTO DE DADOS SEM
INVERSÃO

    MOVLW   B'00100100'         ; CONFIGURA USART
    MOVWF   TXSTA               ; HABILITA TX
                                ; MODO ASSINCRONO
                                ; TRANSMISSÃO DE 8 BITS
                                ; HIGH SPEED BAUD RATE
    MOVLW   .25
    MOVWF   SPBRG               ; ACERTA BAUD RATE -> 9600bps

    MOVLW   B'10010000'         ; CONFIGURA USART
    MOVWF   RCSTA               ; HABILITA RX
                                ; RECEPÇÃO DE 8 BITS
                                ; RECEPÇÃO CONTÍNUA
                                ; DESABILITA ADDRESS DETECT
    CLRF    TMR1L
    CLRF    TMR1H               ; ZERA CONTADOR DO TMR1

MOVLW   B'00000111'             ; CONFIGURA TMR1
MOVWF   T1CON                   ; PRESCALE DE 1:1
                                ; TMR1 INCREM. PELO PINO T1CKI (RC0)
                                ; NÃO SINCRONIZADO COM CLOCK INTERNO
                                ; CONTADOR HABILITADO

MOVLW   B'01001101'             ; CONFIGURA TMR2
MOVWF   T2CON                   ; TMR2 HABILITADO
                                ; POSTSCALE 1:10
                                ; PRESCALE 1:4

CLRF    CCPR2L                  ; ZERA PWM DO CCP2 (RC1 - VENTILADOR)

MOVLW   B'00001100'
MOVWF   CCP2CON                 ; CONFIGURA CCP2 P/ PWM

CLRF    CCPR1L                  ; ZERA PWM DO CCP1 (RC2 - AQUECEDOR)

    MOVLW   B'00001100'
    MOVWF   CCP1CON             ; CONFIGURA CCP1 P/ PWM

;   AS INSTRUÇÕES A SEGUIR FAZEM COM QUE O PROGRAMA TRAVE QUANDO HOUVER UM
;   RESET OU POWER-UP, MAS PASSE DIRETO SE O RESET FOR POR WDT. DESTA FORMA,
;   SEMPRE QUE O PIC É LIGADO, O PROGRAMA TRAVA, AGUARDA UM ESTOURO DE WDT
;   E COMEÇA NOVAMENTE. ISTO EVITA PROBLEMAS NO START-UP DO PIC.

    BTFSC   RCON,NOT_TO         ; RESET POR ESTOURO DE WATCHDOG TIMER ?
    BRA     $                   ; NÃO - AGUARDA ESTOURO DO WDT
                                ; SIM
```

```
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                          INICIALIZAÇÃO DA RAM                          *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTA ROTINA IRÁ LIMPAR TODA A RAM, DO BANCO 0 ATÉ O BANCO 15

    LFSR    FSR0,0x00        ; APONTA FSR0 PARA ENDEREÇO ZERO

LIMPA_RAM
    CLRF    POSTINC0         ; LIMPA A POSIÇÃO E INCREMENTA FSR0
    CLRWDT                   ; LIMPA WATCHDOG TIMER
    MOVLW   0x0F             ; TESTA SE JÁ CHEGOU AO BANCO 15
    SUBWF   FSR0H,W
    BNZ     LIMPA_RAM        ; SE NÃO, CONTINUA LIMPANDO

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                    CONFIGURAÇÕES INICIAIS DO DISPLAY                   *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; ESTA ROTINA INICIALIZA O DISPLAY P/ COMUNICAÇÃO DE 8 VIAS, DISPLAY PARA 2
; LINHAS, CURSOR APAGADO E DESLOCAMENTO DO CURSOR À DIREITA.

INICIALIZACAO_DISPLAY
    BCF     RS               ; SELECIONA O DISPLAY P/ COMANDOS

    MOVLW   0X30             ; ESCREVE COMANDO 0X30 PARA
    RCALL   ESCREVE          ; INICIALIZAÇÃO

    MOVLW   .3
    RCALL   DELAY_MS         ; DELAY DE 3MS (EXIGIDO PELO DISPLAY)

    MOVLW   0X30             ; ESCREVE COMANDO 0X30 PARA
    RCALL   ESCREVE          ; INICIALIZAÇÃO

    MOVLW   0X30             ; ESCREVE COMANDO 0X30 PARA
    RCALL   ESCREVE          ; INICIALIZAÇÃO

    MOVLW   B'00111000'      ; ESCREVE COMANDO PARA
    RCALL   ESCREVE          ; INTERFACE DE 8 VIAS DE DADOS

    MOVLW   B'00000001'      ; ESCREVE COMANDO PARA
    RCALL   ESCREVE          ; LIMPAR TODO O DISPLAY

    MOVLW   .1
    RCALL   DELAY_MS         ; DELAY DE 1MS

    MOVLW   B'00001100'      ; ESCREVE COMANDO PARA
    RCALL   ESCREVE          ; LIGAR O DISPLAY SEM CURSOR

    MOVLW   B'00000110'      ; ESCREVE COMANDO PARA INCREM.
    RCALL   ESCREVE          ; AUTOMÁTICO À DIREITA

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                 ROTINA DE ESCRITA DA TELA PRINCIPAL                    *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTA ROTINA ESCREVE A TELA PRINCIPAL DO PROGRAMA, COM AS FRASES:
;  LINHA 1 - "AQUEC. RPS TEMP."
;  LINHA 2 - " 000%  000 000°C"

    MOVLW   0X80             ; COMANDO PARA POSICIONAR O CURSOR
    RCALL   ESCREVE          ; LINHA 0 / COLUNA 0
    BSF     RS               ; SELECIONA O DISPLAY P/ DADOS

    MOVLW   'A'              ; COMANDOS PARA ESCREVER AS
    RCALL   ESCREVE          ; LETRAS DE "AQUEC. RPS TEMP."
    MOVLW   'Q'
```

```
RCALL   ESCREVE
MOVLW   'U'
RCALL   ESCREVE
MOVLW   'E'
RCALL   ESCREVE
MOVLW   'C'
RCALL   ESCREVE
MOVLW   '.'
RCALL   ESCREVE
MOVLW   ' '
RCALL   ESCREVE
MOVLW   'R'
RCALL   ESCREVE
MOVLW   'P'
RCALL   ESCREVE
MOVLW   'S'
RCALL   ESCREVE
MOVLW   ' '
RCALL   ESCREVE
MOVLW   'T'
RCALL   ESCREVE
MOVLW   'E'
RCALL   ESCREVE
MOVLW   'M'
RCALL   ESCREVE
MOVLW   'P'
RCALL   ESCREVE
MOVLW   '.'
RCALL   ESCREVE

BCF     RS              ; SELECIONA O DISPLAY P/ COMANDOS
MOVLW   0XC1            ; COMANDO PARA POSICIONAR O CURSOR
RCALL   ESCREVE         ; LINHA 1 / COLUNA 1
BSF     RS              ; SELECIONA O DISPLAY P/ DADOS

MOVLW   '0'             ; COMANDOS PARA ESCREVER AS
RCALL   ESCREVE         ; LETRAS DE "000%"
MOVLW   '0'
RCALL   ESCREVE
MOVLW   '0'
RCALL   ESCREVE
MOVLW   '%'
RCALL   ESCREVE


BCF     RS              ; SELECIONA O DISPLAY P/ COMANDOS
MOVLW   0XC7            ; COMANDO PARA POSICIONAR O CURSOR
RCALL   ESCREVE         ; LINHA 1 / COLUNA 7
BSF     RS              ; SELECIONA O DISPLAY P/ DADOS

MOVLW   '0'             ; COMANDOS PARA ESCREVER AS
RCALL   ESCREVE         ; LETRAS DE "000"
MOVLW   '0'
RCALL   ESCREVE
MOVLW   '0'
RCALL   ESCREVE

BCF     RS              ; SELECIONA O DISPLAY P/ COMANDOS
MOVLW   0XCB            ; COMANDO PARA POSICIONAR O CURSOR
RCALL   ESCREVE         ; LINHA 1 / COLUNA 11
BSF     RS              ; SELECIONA O DISPLAY P/ DADOS
```

```
    MOVLW   '0'                 ; COMANDOS PARA ESCREVER AS
    RCALL   ESCREVE             ; LETRAS DE "000°C"
    MOVLW   '0'
    RCALL   ESCREVE
    MOVLW   '0'
    RCALL   ESCREVE
    MOVLW   0XDF
    RCALL   ESCREVE
    MOVLW   'C'
    RCALL   ESCREVE

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                              LOOP PRINCIPAL                           *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  A ROTINA PRINCIPAL FICA AGUARDANDO O MOMENTO DE ESCREVER O VALOR DAS
;  ROTAÇÕES DO VENTILADOR E A TEMPERATURA NO LCD ALÉM DE VARRER O TECLADO
;  PARA MANIPULAR O VALOR DO PWM.

    BSF     ADCON0,GO           ; INICIA CONVERSÃO AD
                                ; EXECUTADA APENAS UMA VEZ
    BSF     INTCON,GIE          ; HABILITA FLAG GLOBAL DAS
                                ; INTERRUPÇÕES
VARRE
    CLRWDT                      ; LIMPA WATCHDOG TIMER

    BTFSC   MOSTRA_RPS          ; DEVE MOSTRAR RPS NO LCD ?
    RCALL   MOSTRA_RPS_LCD      ; SIM - CHAMA ROTINA P/ ATUALIZAR RPS
                                ; NÃO
    BTFSC   MOSTRA_TEMP         ; DEVE MOSTRAR A TEMP. NO LCD ?
    RCALL   MOSTRA_TEMP_LCD     ; SIM - CHAMA ROTINA P/ ATUALIZAR TEMP.
                                ; NÃO
    BTFSS   BOTAO_0             ; O BOTÃO 0 ESTÁ PRESSIONADO ?
    BRA     TRATA_BOTAO_0       ; SIM - PULA P/ TRATA_BOTAO_0
                                ; NÃO
    BTFSS   BOTAO_1             ; O BOTÃO 1 ESTÁ PRESSIONADO ?
    BRA     TRATA_BOTAO_1       ; SIM - PULA P/ TRATA_BOTAO_1
                                ; NÃO
    BTFSS   BOTAO_2             ; O BOTÃO 2 ESTÁ PRESSIONADO ?
    BRA     TRATA_BOTAO_2       ; SIM - PULA P/ TRATA_BOTAO_2
                                ; NÃO
    BTFSS   BOTAO_3             ; O BOTÃO 3 ESTÁ PRESSIONADO ?
    BRA     TRATA_BOTAO_3       ; SIM - PULA P/ TRATA_BOTAO_3
                                ; NÃO

    MOVLW   FILTRO_TECLA        ; CARREGA NO WORK O VALOR DE FILTRO_TECLA
    MOVWF   FILTRO_BOTOES       ; SALVA EM FILTRO_BOTOES
                                ; RECARREGA FILTRO P/ EVITAR RUIDOS
    MOVLW   .1
    MOVWF   TEMPO_TURBO         ; CARREGA TEMPO DO TURBO DAS TECLAS
                                ; COM 1 - IGNORA O TURBO A PRIMEIRA
                                ; VEZ QUE A TECLA É PRESSIONADA

    BRA     VARRE               ; VOLTA PARA VARRER TECLADO

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                          TRATAMENTO DOS BOTÕES                        *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

; ************************* TRATAMENTO DO BOTÃO 0 ****************************

TRATA_BOTAO_0
    DECFSZ  FILTRO_BOTOES,F ; FIM DO FILTRO ? (RUIDO?)
    BRA     VARRE           ; NÃO - VOLTA P/ VARRE
                            ; SIM - BOTÃO PRESSIONADO
```

```
DECFSZ  TEMPO_TURBO,F          ; FIM DO TEMPO DE TURBO ?
BRA     VARRE                  ; NÃO - VOLTA P/ VARRE
                               ; SIM
MOVLW   TURBO_TECLA
MOVWF   TEMPO_TURBO            ; RECARREGA TEMPORIZADOR DO TURBO
                               ; DAS TECLAS
MOVLW   .100
XORWF   INTENSIDADE_AQUE,W
BTFSS   STATUS,Z               ; PODE INCREMENTAR PWM DO AQUECEDOR ?
INCF    INTENSIDADE_AQUE,F     ; SIM - INCREMENTA INTENSIDADE_AQUE
                               ; NÃO
MOVF    INTENSIDADE_AQUE,W     ; CARREGA INTENSIDADE_AQUE NO WORK
MULLW   .10                    ; MULTIPLICA VALOR POR 10

SWAPF   PRODL,W
ANDLW   B'00110000'
IORLW   B'00001100'
RRCF    PRODH,F
RRCF    PRODL,F
RRCF    PRODH,F
MOVWF   CCP1CON
RRCF    PRODL,W                ; ATUALIZA REGISTRADORES DO DUTY CYCLE
MOVWF   CCPR1L                 ; DO MÓDULO CCP1 - PWM DO AQUECEDOR

MOVF    INTENSIDADE_AQUE,W     ; FAZ O AJUSTE DECIMAL DA INTENSIDADE
RCALL   AJUSTE_DECIMAL         ; DO PWM DO AQUECEDOR

BCF     RS                     ; SELECIONA O DISPLAY P/ COMANDOS
MOVLW   0XC1                   ; COMANDO PARA POSICIONAR O CURSOR
RCALL   ESCREVE                ; LINHA 1 / COLUNA 1
BSF     RS                     ; SELECIONA O DISPLAY P/ DADOS

MOVF    CENTENA,W
ADDLW   0X30                   ; FAZ AJUSTE ASCII DA CENTENA
RCALL   ESCREVE                ; ESCREVE VALOR NO LCD

MOVF    DEZENA,W
ADDLW   0X30                   ; FAZ AJUSTE ASCII DA DEZENA
RCALL   ESCREVE                ; ESCREVE VALOR NO LCD

MOVF    UNIDADE,W
ADDLW   0X30                   ; FAZ AJUSTE ASCII DA UNIDADE
RCALL   ESCREVE                ; ESCREVE VALOR NO LCD

BRA     VARRE                  ; VOLTA P/ VARREDURA DOS BOTÕES

; ************************** TRATAMENTO DO BOTÃO 1 ****************************

TRATA_BOTAO_1
    DECFSZ  FILTRO_BOTOES,F    ; FIM DO FILTRO ? (RUIDO?)
    BRA     VARRE              ; NÃO - VOLTA P/ VARRE
                               ; SIM - BOTÃO PRESSIONADO
    DECFSZ  TEMPO_TURBO,F      ; FIM DO TEMPO DE TURBO ?
    BRA     VARRE              ; NÃO - VOLTA P/ VARRE
                               ; SIM
    MOVLW   TURBO_TECLA
    MOVWF   TEMPO_TURBO        ; RECARREGA TEMPORIZADOR DO TURBO
                               ; DAS TECLAS
    MOVF    INTENSIDADE_AQUE,F
    BTFSS   STATUS,Z           ; PODE DECREMENTAR PWM DO AQUECEDOR ?
    DECF    INTENSIDADE_AQUE,F ; SIM - DECREMENTA INTENSIDADE_AQUE
                               ; NÃO
    MOVF    INTENSIDADE_AQUE,W ; CARREGA INTENSIDADE_AQUE NO WORK
    MULLW   .10                ; MULTIPLICA VALOR POR 10
```

```
SWAPF   PRODL,W
ANDLW   B'00110000'
IORLW   B'00001100'
RRCF    PRODH,F
RRCF    PRODL,F
RRCF    PRODH,F
MOVWF   CCP1CON
RRCF    PRODL,W                 ; ATUALIZA REGISTRADORES DO DUTY CYCLE
MOVWF   CCPR1L                  ; DO MÓDULO CCP1 - PWM DO AQUECEDOR

MOVF    INTENSIDADE_AQUE,W      ; FAZ O AJUSTE DECIMAL DA INTENSIDADE
RCALL   AJUSTE_DECIMAL          ; DO PWM DO AQUECEDOR

BCF     RS                      ; SELECIONA O DISPLAY P/ COMANDOS
MOVLW   0XC1                    ; COMANDO PARA POSICIONAR O CURSOR
RCALL   ESCREVE                 ; LINHA 1 / COLUNA 1
BSF     RS                      ; SELECIONA O DISPLAY P/ DADOS

MOVF    CENTENA,W
ADDLW   0X30                    ; FAZ AJUSTE ASCII DA CENTENA
RCALL   ESCREVE                 ; ESCREVE VALOR NO LCD

MOVF    DEZENA,W
ADDLW   0X30                    ; FAZ AJUSTE ASCII DA DEZENA
RCALL   ESCREVE                 ; ESCREVE VALOR NO LCD

MOVF    UNIDADE,W
ADDLW   0X30                    ; FAZ AJUSTE ASCII DA UNIDADE
RCALL   ESCREVE                 ; ESCREVE VALOR NO LCD

BRA     VARRE                   ; VOLTA P/ VARREDURA DOS BOTÕES

; ************************** TRATAMENTO DO BOTÃO 2 ****************************

TRATA_BOTAO_2
    DECFSZ  FILTRO_BOTOES,F     ; FIM DO FILTRO ? (RUIDO?)
    BRA     VARRE               ; NÃO - VOLTA P/ VARRE
                                ; SIM - BOTÃO PRESSIONADO
    DECFSZ  TEMPO_TURBO,F       ; FIM DO TEMPO DE TURBO ?
    BRA     VARRE               ; NÃO - VOLTA P/ VARRE
                                ; SIM
    MOVLW   TURBO_TECLA
    MOVWF   TEMPO_TURBO         ; RECARREGA TEMPORIZADOR DO TURBO
                                ; DAS TECLAS
    MOVLW   .100
    XORWF   INTENSIDADE_VENT,W
    BTFSS   STATUS,Z            ; PODE INCREMENTAR PWM DO VENTILADOR ?
    INCF    INTENSIDADE_VENT,F  ; SIM - INCREMENTA INTENSIDADE_VENT
                                ; NÃO
    MOVF    INTENSIDADE_VENT,W  ; CARREGA INTENSIDADE_AQUE NO WORK
    MULLW   .10                 ; MULTIPLICA VALOR POR 10

    SWAPF   PRODL,W
    ANDLW   B'00110000'
    IORLW   B'00001111'
    RRCF    PRODH,F
    RRCF    PRODL,F
    RRCF    PRODH,F
    MOVWF   CCP2CON
    RRCF    PRODL,W             ; ATUALIZA REGISTRADORES DO DUTY CYCLE
    MOVWF   CCPR2L              ; DO MÓDULO CCP2 - PWM DO VENTILADOR

    BRA     VARRE               ; VOLTA P/ VARREDURA DOS BOTÕES
```

```
; ************************* TRATAMENTO DO BOTÃO 3 ****************************

TRATA_BOTAO_3
    DECFSZ  FILTRO_BOTOES,F       ; FIM DO FILTRO ? (RUIDO?)
    BRA     VARRE                 ; NÃO - VOLTA P/ VARRE
                                  ; SIM - BOTÃO PRESSIONADO
    DECFSZ  TEMPO_TURBO,F         ; FIM DO TEMPO DE TURBO ?
    BRA     VARRE                 ; NÃO - VOLTA P/ VARRE
                                  ; SIM
    MOVLW   TURBO_TECLA
    MOVWF   TEMPO_TURBO           ; RECARREGA TEMPORIZADOR DO TURBO
                                  ; DAS TECLAS
    MOVF    INTENSIDADE_VENT,F
    BTFSS    STATUS,Z             ; PODE DECREMENTAR PWM DO VENTILADOR ?
    DECF    INTENSIDADE_VENT,F    ; SIM - DECREMENTA INTENSIDADE_VENT
                                  ; NÃO
    MOVF    INTENSIDADE_VENT,W    ; CARREGA INTENSIDADE_AQUE NO WORK
    MULLW   .10                   ; MULTIPLICA VALOR POR 10

    SWAPF   PRODL,W
    ANDLW   B'00110000'
    IORLW   B'00001111'
    RRCF    PRODH,F
    RRCF    PRODL,F
    RRCF    PRODH,F
    MOVWF   CCP2CON
    RRCF    PRODL,W               ; ATUALIZA REGISTRADORES DO DUTY CYCLE
    MOVWF   CCPR2L                ; DO MÓDULO CCP2 - PWM DO VENTILADOR

    BRA     VARRE                 ; VOLTA P/ VARREDURA DOS BOTÕES

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                TABELAS DE CONVERSAO P/ TEMPERATURA DO DIODO             *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

    ORG     0X400                 ; POSICIONA O INICIO DA TABELA EM 0X400
    RADIX   DEC                   ; CONFIGURA RADIX EM DECIMAL

TABELA_TEMPERATURA

    MOVF    TEMPERATURA,W
    ADDLW   LOW     (TABELA_TEMP)
    MOVWF   TBLPTRL

    MOVLW   HIGH    (TABELA_TEMP)
    MOVWF   TBLPTRH
    BNC     $+.4
    INCF    TBLPTRH,F

    TBLRD*
    MOVF    TABLAT,W
    RETURN

TABELA_TEMP
    DB  000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000 ;15
    DB  000,000,000,000,000,000,000,000,000,000,000,000,000,000,001,001 ;31
    DB  002,002,003,003,004,004,005,005,006,006,007,007,008,008,009,009 ;47
    DB  010,010,011,011,012,012,013,013,014,014,015,015,016,016,017,017 ;63

    DB  018,018,019,019,020,020,021,021,022,022,023,023,023,024,024,025 ;79
    DB  025,026,026,027,027,028,028,029,029,030,030,031,031,032,032,033 ;95
    DB  033,034,034,035,035,036,036,037,037,038,038,039,039,040,040,041 ;111
    DB  041,042,042,043,043,044,044,045,045,046,046,047,047,048,048,049 ;127
    DB  049,050,050,051,051,052,052,053,053,054,054,055,055,056,056,057 ;143
```

```
    DB  057,058,058,059,059,060,060,061,061,062,062,063,063,064,064,065 ;159
    DB  065,066,066,067,067,068,068,069,069,070,070,071,071,072,072,073 ;175
    DB  073,074,074,075,075,076,076,077,077,078,078,079,079,080,080,081 ;191
    DB  081,082,082,083,083,084,084,085,085,086,086,087,087,088,088,089 ;207
    DB  089,090,090,091,091,092,092,093,093,094,094,095,095,096,096,097 ;223
    DB  097,098,098,099,099,100,100,101,101,102,102,103,103,104,104,104 ;239
    DB  105,105,106,106,107,107,108,108,109,109,110,110,111,111,112,112 ;255

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                              FIM DO PROGRAMA                        *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

    END                         ; FIM DO PROGRAMA
```

A

Detalhamento dos Registradores Especiais (SFRs)

» Introdução

Este apêndice destina-se a apresentar um detalhamento de todos os registradores especiais (SFRs) disponíveis no PIC18F4520, para auxiliá-lo na programação dos seus sistemas.

A coluna Reset representa o valor do bit após um reset do microcontrolador e a coluna End. especifica um endereço na memória. A coluna Cap. refere-se ao capítulo que trata do recurso relacionado ao SFR e a coluna Pág. indica a página onde se encontra o detalhamento dos bits (quando houver).

Os registradores que possuem bits internos são detalhados caso a caso. Para efeito de padronização, cada bit dentro desses registradores recebe um nome, sendo também especificado se esse bit pode ser lido (R-Read) e/ou escrito (W-Write). Caso o bit não esteja implementado (U-unimplemented), será lido com zero.

» Agrupamento e localizações

Grupo	Cap.	Item	Descrição	Pág.	End.	Reset
Gerais	2	STATUS	Registrador de Status	301	FD8	---x xxxx
Gerais	13	RCON	Monitoração do Reset	301	FD0	0q-1 11q0
Gerais	13	OSCCON	Seleção e configuração do clock	302	FD3	0100 q000
Gerais	13	OSCTUNE	Seleção e configuração do clock	302	F9B	0--0 0000
Gerais	13	WDTCON	Configuração do Watchdog Timer	303	FD1	---- ---0
Gerais	2	WREG	Acumulador	-	FE8	xxxx xxxx
Gerais	2	TOSU	Endereço apontado por STKPTR (byte 2)	-	FFF	---0 0000
Gerais	2	TOSH	Endereço apontado por STKPTR (byte 1)	-	FFE	0000 0000
Gerais	2	TOSL	Endereço apontado por STKPTR (byte 0)	-	FFD	0000 0000
Gerais	2	STKPTR	Ponteiro para a pilha	-	FFC	00-0 0000
Gerais	2	PCLATU	Valor do contador de programa (byte 2)	-	FFB	---0 0000
Gerais	2	PCLATH	Valor do contador de programa (byte 1)	-	FFA	0000 0000
Gerais	2	PCL	Valor do contador de programa (byte 0)	-	FF9	0000 0000

Grupo	Cap.	Item	Descrição	Pág.	End.	Reset
Gerais	2	PRODH	Resultado das instruções de multiplicação (byte 1)	-	FF4	xxxx xxxx
Gerais	2	PRODL	Resultado das instruções de multiplicação (byte 0)	-	FF3	xxxx xxxx
Gerais	2	INDF0	Acessa o dado apontado pelo FSR0	-	FEF	---- ----
Gerais	2	POSTINC0	Acessa o dado apontado pelo FSR0 e depois incrementa FSR0	-	FEE	---- ----
Gerais	2	POSTDEC0	Acessa o dado apontado pelo FSR0 e depois decrementa FSR0	-	FED	---- ----
Gerais	2	PREINC0	Incrementa FSR0 e depois acessa o dado do FSR0 incrementado	-	FEC	---- ----
Gerais	2	PLUSW0	Acessa o dado apontado pelo FSR0 com um offset acertado anteriormente em W (faixa de –127 até 128). Não altera o FSR0	-	FEB	---- ----
Gerais	2	FSR0H	Ponteiro para memória de dados (byte 1)	-	FEA	---- 0000
Gerais	2	FSR0L	Ponteiro para memória de dados (byte 0)	-	FE9	xxxx xxxx
Gerais	2	INDF1	Acessa o dado apontado pelo FSR1	-	FE7	---- ----
Gerais	2	POSTINC1	Acessa o dado apontado pelo FSR1 e depois incrementa FSR1	-	FE6	---- ----
Gerais	2	POSTDEC1	Acessa o dado apontado pelo FSR1 e depois decrementa FSR1	-	FE5	---- ----
Gerais	2	PREINC1	Incrementa FSR1 e depois acessa o dado do FSR1 incrementado	-	FE4	---- ----
Gerais	2	PLUSW1	Acessa o dado apontado pelo FSR1 com um offset acertado anteriormente em W (faixa de –127 até 128). Não altera o FSR1	-	FE3	---- ----
Gerais	2	FSR1H	Ponteiro para memória de dados (byte 1)	-	FE2	---- 0000
Gerais	2	FSR1L	Ponteiro para memória de dados (byte 0)	-	FE1	xxxx xxxx
Gerais	2	BSR	Seleção de banco de memória (memória de dados)	-	FE0	---- 0000
Gerais	2	INDF2	Acessa o dado apontado pelo FSR2	-	FDF	---- ----
Gerais	2	POSTINC2	Acessa o dado apontado pelo FSR2 e depois incrementa FSR2	-	FDE	---- ----
Gerais	2	POSTDEC2	Acessa o dado apontado pelo FSR2 e depois decrementa FSR2	-	FDD	---- ----
Gerais	2	PREINC2	Incrementa FSR2 e depois acessa o dado do FSR2 incrementado	-	FDC	---- ----
Gerais	2	PLUSW2	Acessa o dado apontado pelo FSR2 com um offset acertado anteriormente em W (faixa de –127 até 128). Não altera o FSR2	-	FDB	---- ----
Gerais	2	FSR2H	Ponteiro para memória de dados (byte 1)	-	FDA	---- 0000
Gerais	2	FSR2L	Ponteiro para memória de dados (byte 0)	-	FD9	xxxx xxxx
Portas	4	PORTA	Acesso aos pinos de I/O	-	F80	xx0x xxxx
Portas	4	TRISA	Configura a direção dos pinos de I/O	-	F92	1111 1111
Portas	4	LATA	Acesso aos pinos de I/O	-	F89	xxxx xxxx
Portas	4	PORTB	Acesso aos pinos de I/O	-	F81	xxxx xxxx
Portas	4	TRISB	Configura a direção dos pinos de I/O	-	F93	1111 1111
Portas	4	LATB	Acesso aos pinos de I/O	-	F8A	xxxx xxxx

Grupo	Cap.	Item	Descrição	Pág.	End.	Reset
Portas	4	PORTC	Acesso aos pinos de I/O	-	F82	xxxx xxxx
Portas	4	TRISC	Configura a direção dos pinos de I/O	-	F94	1111 1111
Portas	4	LATC	Acesso aos pinos de I/O	-	F8B	xxxx xxxx
Portas	4	PORTD	Acesso aos pinos de I/O	-	F83	xxxx xxxx
Portas	4	TRISD	Configura a direção dos pinos de I/O	-	F95	1111 1111
Portas	4	LATD	Acesso aos pinos de I/O	-	F8C	xxxx xxxx
Portas	4	PORTE	Acesso aos pinos de I/O	-	F84	---- xxxx
Portas	4	TRISE	Configura a direção dos pinos de I/O	-	F96	---- -111
Portas	4	LATE	Acesso aos pinos de I/O	-	F8D	---- -xxx
TMR	4	T0CON	Configuração do Timer0	303	FD5	1111 1111
TMR	4	TMR0L	Contador do Timer0 (byte 0)	-	FD6	xxxx xxxx
TMR	4	TMR0H	Contador do Timer0 (byte 1)	-	FD7	0000 0000
TMR	4	T1CON	Configuração do Timer1	304	FCD	0000 0000
TMR	4	TMR1L	Contador do Timer1 (byte 0)	-	FCE	xxxx xxxx
TMR	4	TMR1H	Contador do Timer1 (byte 1)	-	FCF	xxxx xxxx
TMR	4	T2CON	Configuração do Timer2	304	FCC	0000 0000
TMR	4	PR2	Registrador de período do Timer2	-	FCB	1111 1111
TMR	4	TMR2	Contador do Timer2	-	FCA	0000 0000
TMR	4	T3CON	Configuração do Timer3	305	FB3	0000 0000
TMR	4	TMR3L	Contador do Timer3 (byte 0)	-	FB2	xxxx xxxx
TMR	4	TMR3H	Contador do Timer3 (byte 1)	-	FB1	xxxx xxxx
INT	2	INTCON	Configuração das interrupções	305	FF2	0000 000x
INT	2	INTCON2	Configuração das interrupções	306	FF1	1111 -1-1
INT	2	INTCON3	Configuração das interrupções	306	FF0	11-0 0-00
INT	2	PIE1	Configuração das interrupções (chaves)	307	F9D	0000 0000
INT	2	PIR1	Configuração das interrupções (flags)	308	F9E	0000 0000
INT	2	IPR1	Configuração das interrupções (prioridade)	308	F9F	1111 1111
INT	2	PIE2	Configuração das interrupções (chaves)	309	FA0	00-0 0000
INT	2	PIR2	Configuração das interrupções (flags)	309	FA1	00-0 0000
INT	2	IPR2	Configuração das interrupções (prioridade)	310	FA2	11-1 1111
AD	7	ADCON0	Configuração do conversor AD	311	FC2	--00 0000
AD	7	ADCON1	Configuração do conversor AD	311	FC1	--00 0qqq
AD	7	ADCON2	Configuração do conversor AD	312	FC0	0-00 0000
AD	7	ADRESH	Resultado da conversão do AD (byte 1)	-	FC4	xxxx xxxx
AD	7	ADRESL	Resultado da conversão do AD (byte 0)	-	FC3	xxxx xxxx
CCP	9	CCP1CON	Configuração do módulo CCP1	313	FBD	0000 0000
CCP	9	CCPR1L	Configuração/resultado do módulo CCP1	-	FBE	xxxx xxxx
CCP	9	CCPR1H	Configuração/resultado do módulo CCP1	-	FBF	xxxx xxxx
CCP	9	CCP2CON	Configuração do módulo CCP2	313	FBA	--00 0000
CCP	9	CCPR2L	Configuração/resultado do módulo CCP2	-	FBB	xxxx xxxx

Grupo	Cap.	Item	Descrição	Pág.	End.	Reset
CCP	9	CCPR2H	Configuração/resultado do módulo CCP2	-	FBC	xxxx xxxx
CCP	9	PWM1CON	Configuração do módulo ECCP1	314	FB7	0000 0000
CCP	9	ECCP1AS	Configuração do módulo ECCP1	314	FB6	0000 0000
E²PROM	10	EECON1	Configuração de gravação/leitura da E²PROM/Flash	314	FA6	xx-0 x000
E²PROM	10	EECON2	Configuração de gravação/leitura da E²PROM/Flash	-	FA7	0000 0000
E²PROM	10	EEDATA	Leitura/escrita do dado da E²PROM	-	FA8	0000 0000
E²PROM	10	EEADR	Endereço da E²PROM	-	FA9	xxxx xxxx
E²PROM	10	TBLPTRU	Ponteiro para memória de programa (byte 2)	-	FF8	--00 0000
E²PROM	10	TBLPTRH	Ponteiro para memória de programa (byte 1)	-	FF7	0000 0000
E²PROM	10	TBLPTRL	Ponteiro para memória de programa (byte 0)	-	FF6	0000 0000
E²PROM	10	TABLAT	Acessa o dado da memória de programa	-	FF5	0000 0000
MSSP	11	SSPCON1	Configuração do módulo MSSP	315	FC6	0000 0000
MSSP	11	SSPCON2	Configuração do módulo MSSP	316	FC5	0000 0000
MSSP	11	SSPSTAT	Configuração do módulo MSSP	317	FC7	0000 0000
MSSP	11	SSPADD	Configuração do módulo MSSP	-	FC8	0000 0000
MSSP	11	SSPBUF	Leitura/escrita do dado no módulo MSSP	-	FC9	xxxx xxxx
USART	12	TXSTA	Configuração do módulo USART (TX)	318	FAC	0000 0010
USART	12	RCSTA	Configuração do módulo USART (RX)	318	FAB	0000 000x
USART	12	TXREG	Buffer de transmissão do dado da USART	-	FAD	0000 0000
USART	12	RCREG	Buffer de recepção do dado da USART	-	FAE	0000 0000
USART	12	SPBRG	Configuração do baud rate	-	FAF	0000 0000
USART	12	SPBRGH	Configuração do baud rate	-	FB0	0000 0000
USART	12	BAUDCON	Configuração do EUSART	319	FB8	01-0 0-00
COMP	13	CMCON	Configuração dos comparadores	319	FB4	0000 0111
COMP	13	CVRCON	Configuração do módulo gerador de tensão de referência.	321	FB5	0000 0000
HLVD	13	HLVDCON	Configuração do High/Low Voltage Detect	322	FD2	0-00 0101

Legenda para coluna Reset

- ✓ **-:** não implementado
- ✓ **x:** estado incerto (pode ser 0 ou 1)
- ✓ **q:** valor depende de condição do hardware relacionado

STATUS				
Bit	**R/W**	**Reset**	**Nome**	**Descrição**
7 - 5	U	0	-	Não implementado.
4	R/W	0	N	Negativo: 0 = Indica que resultado de uma operação aritmética sinalizada é negativo. 1 = Indica que resultado de uma operação aritmética sinalizada é positivo.
3	R/W	0	OV	Overflow: 0 = Indica que ocorreu overflow em operação aritmética sinalizada. 1 = Indica que não ocorreu overflow em operação aritmética sinalizada.
2	R/W	0	Z	Indicação de Zero: 0 = Indica que o resultado da última operação (lógica ou aritmética) não resultou em zero. 1 = Indica que o resultado da última operação (lógica ou aritmética) resultou em zero
1	R/W	0	DC	Digit Carry/borrow: 0 = A última operação da ULA não ocasionou um estouro de dígito. 1 = A última operação da ULA ocasionou um estouro (carry) entre o bit 3 e 4, isto é, o resultado ultrapassou os 4 bits menos significativos. Utilizado quando se trabalha com números de 4 bits.
0	R/W	0	C	Carry/borrow: 0 = A última operação da ULA não ocasionou um estouro (carry). 1 = A última operação da ULA ocasionou um estouro (carry) no bit mais significativo, isto é, o resultado ultrapassou os 8 bits disponíveis.

RCON				
Bit	**R/W**	**Reset**	**Nome**	**Descrição**
7	R/W	0	IPEN	Habilitação do sistema de prioridade de interrupção: 0 =Desabilita o sistema de prioridades (compatibilidade com a família PIC16). 1 = Habilita o sistema de prioridades.
6	R/W	0	SBOREN	Habilitação do BOR: 0 = BOR desabilitado. 1 = BOR habilitado.
5	R/W	0	-	Não implementado.
4	R/W	1	/RI	Monitoração de reset pela execução da instrução RESET: 0 = A instrução RESET foi executada. 1 = A instrução RESET não foi executada.
3	R/W	1	/TO	Monitoração de reset pela execução por estouro do WDT: 0 = Ocorreu o estouro do WDT. 1 = Power-up ou execução da instrução CLRWDT ou SLEEP.
2	R/W	1	/PD	Monitoração de Power Down: 0 = Execução da instrução CLRWDT. 1 = Power-up ou execução da instrução CLRWDT.
1	R/W	0	/POR	Monitoração de reset por Power On Reset: 0 = Ocorreu um POR. 1 = Não ocorreu um POR (setado apenas pelo firmware).
0	R/W	0	/BOR	Monitoração de reset por Brown Out Reset: 0 = Ocorreu um BOR. 1 = Não ocorreu um BOR (setado apenas pelo firmware).

OSCCON				
Bit	**R/W**	**Reset**	**Nome**	**Descrição**
7	R/W	0	IDLEN	Habilitação do modo Idle: 0 = Microcontrolador entra no modo Sleep após execução da instrução SLEEP. 1 = Microcontrolador entra no modo Idle após execução da instrução SLEEP.
6	R/W	1	IRCF2	Seleção da frequência do oscilador interno: 111 = 8 MHz (frequência principal do INTOSC) 110 = 4 MHz 101 = 2 MHz 100 = 1 MHz 011 = 500 kHz 010 = 250 kHz 001 = 125 kHz 000 = 31,25 kHz ou 31 kHz (veja descrição do bit OCSTUNE<INTSRC>)
5	R/W	0	IRCF1	
4	R/W	0	IRCF2	
3	R	0	OSTS	Oscillator Start-up Time-out: 0 = Oscillator Start-up time em funcionamento. Oscilador principal está parado. 1 = Oscillator Start-up time terminou. Oscilador principal está rodando.
2	R	0	IOFS	Monitoração de estabilidade de frequência do INTOSC: 0 = Frequência do INTOSC instável (INTOSC parado). 1 = Frequência do INTOSC estável (INTOSC rodando).
1	R/W	0	SCS1	Seleção de fonte de clock: 1x = Oscilador interno 01 = Oscilador secundário (Timer1) 00 = Oscilador primário
0	R/W	0	SCS0	

OSCTUNE				
Bit	**R/W**	**Reset**	**Nome**	**Descrição**
7	R/W	0	INTSRC	Seleção do clock de baixa frequência: 0 = 31 kHz derivado do INTRC. 1 = 31,25 kHz derivado do INTOSC (8 MHz/256).
6	R/W	0	PLLEN	Habilitação do PLL para INTOSC: 0 = PLL desabilitado. 1 = PLL habilitado (somente para frequências de 8 MHz e 4 MHz).
5	R/W	0	-	Não implementado.
4	R/W	0	TUN4	Ajuste de frequência do oscilador INTOSC: 01111 = Máxima frequência ... 00001 = ... 00000 = Frequência central (8 MHz) 11111 = ... 10000 = Mínima frequência
3	R/W	0	TUN3	
2	R/W	0	TUN2	
1	R/W	0	TUN1	
0	R/W	0	TUN0	

WDTCON				
Bit	**R/W**	**Reset**	**Nome**	**Descrição**
0	R/W	0	SWDTEN	Habilitação do Watchdog Timer: 0 = Watchdog Timer desligado. 1 = Watchdog Timer ligado.

T0CON				
Bit	**R/W**	**Reset**	**Nome**	**Descrição**
7	R/W	1	TMR0ON	Habilitação do Timer0: 0 = Desabilita o Timer0. 1 = Habilita o Timer0.
6	R/W	1	T08BIT	Configuração do contador: 0 = Contador do Timer0 em 16 bits. 1 = Contador do Timer0 em 8 bits.
5	R/W	1	T0CS	Seleção de fonte de clock: 0 = Clock derivado do oscilador principal. 1 = Clock derivado do pino T0CKI.
4	R/W	1	T0SE	Seleção de borda do clock externo (T0CKI): 0 = Incremento na borda de subida. 1 = Incremento na borda de descida.
3	R/W	1	PSA	Habilitação do prescaler: 0 = Prescaler não habilitado (fator de divisão 1:1). 1 = Prescaler habilitado.
2	R/W	1	T0PS2	Configuração do prescaler: 111 = 1:256 110 = 1:128 101 = 1:64 100 = 1:32 011 = 1:16 010 = 1:8 001 = 1:4 000 = 1:2
1	R/W	1	T0PS1	
0	R/W	1	T0PS0	

T1CON				
Bit	**R/W**	**Reset**	**Nome**	**Descrição**
7	R/W	0	RD16	Configuração do modo de leitura/escrita do contador: 0 = Contador do Timer1 em modo de leitura/escrita em 8 bits. 1 = Contador do Timer1 em modo de leitura/escrita em 16 bits.
6	U	0	-	Não implementado.
5	R/W	0	T1CKPS1	Configuração do prescaler: 11 = 1:8 10 = 1:4 01 = 1:2 00 = 1:1
4	R/W	0	T1CKPS0	
3	R/W	0	T1OSCEN	Habilitação do circuito oscilador: 0 = Oscilador do Timer1 desabilitado. 1 = Oscilador do Timer1 habilitado.
2	R/W	0	/T1SYNC	Habilitação do sincronismo de clock: 0 = Habilita sincronismo do clock externo. 1 = Desabilita sincronismo do clock externo.
1	R/W	0	TMR1CS	Seleção de fonte de clock: 0 = Clock derivado do oscilador principal. 1 = Clock derivado do pino T1OSO/T13CKI.
0	R/W	0	TMR1ON	Habilitação do Timer1: 0 = Desabilita o Timer1. 1 = Habilita o Timer1.

T2CON				
Bit	**R/W**	**Reset**	**Nome**	**Descrição**
7	U	0	-	Não implementado.
6	R/W	0	TOUTPS3	Configuração do postscaler 0000 = 1:1 1000 = 1:9 0001 = 1:2 1001 = 1:10 0010 = 1:3 1010 = 1:11 0011 = 1:4 1011 = 1:12 0100 = 1:5 1100 = 1:13 0101 = 1:6 1101 = 1:14 0110 = 1:7 1110 = 1:15 0111 = 1:8 1111 = 1:16
5	R/W	0	TOUTPS2	
4	R/W	0	TOUTPS1	
3	R/W	0	TOUTPS0	
2	R/W	0	TMR2ON	Habilitação do Timer2: 0 = Desabilita o Timer2. 1 = Habilita o Timer2.
1	R/W	0	T2CKPS1	Configuração do prescaler: 00 = 1:1 01 = 1:4 1x = 1:16
0	R/W	0	T2CKPS0	

T3CON				
Bit	**R/W**	**Reset**	**Nome**	**Descrição**
7	R/W	0	RD16	Configuração do modo de leitura/escrita do contador: 0 = Contador do Timer3 em modo de leitura/escrita em 8 bits. 1 = Contador do Timer3 em modo de leitura/escrita em 16 bits.
6	R/W	0	T3CCP2	**Vide descrição do bit T3CCP1**
5	R/W	0	T3CKPS1	Configuração do prescaler: 11 = 1:8 10 = 1:4 01 = 1:2 00 = 1:1
4	R/W	0	T3CKPS0	
3	R/W	0	T3CCP1	Seleção de base de tempo para o módulo CCP1 e CCP2 (bits T3CCP2:T3CCP1): 1x = Timer3 como base de tempo para CCP1 e CCP2 (Capture e Compare) 01 = Timer3 como base de tempo para CCP2 (Capture e Compare) Timer1 como base de tempo para CCP1 (Capture e Compare) 00 = Timer1 como base de tempo para CCP1 e CCP2 (Capture e Compare)
2	R/W	0	/T3SYNC	Habilitação do sincronismo de clock: 0 = Habilita sincronismo do clock externo. 1 = Desabilita sincronismo do clock externo.
1	R/W	0	TMR3CS	Seleção de fonte de clock: 0 = Clock derivado do oscilador principal. 1 = Clock derivado do pino T1OSO/T13CKI.
0	R/W	0	TMR3ON	Habilitação do Timer3: 0 = Desabilita o Timer3. 1 = Habilita o Timer3.

INTCON				
Bit	**R/W**	**Reset**	**Nome**	**Descrição**
7	R/W	0	GIE/GIEH	Habilitação geral das interrupções (chave geral): 0 = Nenhuma interrupção será tratada. 1 = As interrupções habilitadas individualmente serão tratadas.
6	R/W	0	PEIE/GIEL	Habilitação das interrupções de periféricos (chave geral para periféricos): 0 = As interrupções de periféricos não serão tratadas. 1 = As interrupções de periféricos habilitadas individualmente serão tratadas.
5	R/W	0	TMR0IE	Habilitação da interrupção de estouro de TMR0 (chave individual): 0 = Interrupção de TMR0 desabilitada. 1 = Interrupção de TMR0 habilitada.
4	R/W	0	INT0IE	Habilitação da interrupção externa no pino RB0 (chave individual): 0 = Interrupção externa desabilitada. 1 = Interrupção externa habilitada.
3	R/W	0	RBIE	Habilitação da interrupção por mudança de estado no pinos RB4 a RB7 (chave individual): 0 = Interrupção por mudança de estado desabilitada. 1 = Interrupção por mudança de estado habilitada.
2	R/W	0	TMR0IF	Identificação de estouro do TMR0: 0 = Não ocorreu estouro do TMR0. 1 = Ocorreu estouro do TMR0 (este bit deve ser limpo por software).

INTCON				
Bit	**R/W**	**Reset**	**Nome**	**Descrição**
1	R/W	0	INT0IF	Identificação da interrupção externa no pino RB0: 0 = Não ocorreu evento da interrupção. 1 = Ocorreu evento da interrupção (este bit deve ser limpo por software).
0	R/W	x	RBIF	Identificação da interrupção por mudança de estado nos pinos RB4 a RB7: 0 = Não ocorreu evento da interrupção. 1 = Ocorreu evento da interrupção (este bit deve ser limpo por software).

INTCON2				
Bit	**R/W**	**Reset**	**Nome**	**Descrição**
7	R/W	0	/RBPU	Habilita pull-ups internos para o PORTB: 0 = Pull-ups habilitados para todos os pinos do PORTB configurados como saída. 1 = Pull-ups desabilitados.
6	R/W	0	INTEDG0	Configuração da borda que gerará a interrupção externa no RB0: 0 = A interrupção ocorrerá na borda de descida. 1 = A interrupção ocorrerá na borda de subida.
5	R/W	0	INTEDG1	Configuração da borda que gerará a interrupção externa no RB1: 0 = A interrupção ocorrerá na borda de descida. 1 = A interrupção ocorrerá na borda de subida.
4	R/W	0	INTEDG2	Configuração da borda que gerará a interrupção externa no RB2: 0 = A interrupção ocorrerá na borda de descida. 1 = A interrupção ocorrerá na borda de subida.
3	U	0	-	Não implementado.
2	R/W	0	TM0IP	Prioridade da interrupção do TMR0: 0 = Baixa prioridade. 1 = Alta prioridade.
1	U	0	-	Não implementado.
0	R/W	0	RBIP	Prioridade da interrupção por mudança de estado nos pinos RB4 a RB7: 0 = Baixa prioridade. 1 = Alta prioridade.

INTCON3				
Bit	**R/W**	**Reset**	**Nome**	**Descrição**
7	R/W	0	INT2IP	Prioridade da interrupção externa no RB2: 0 = Baixa prioridade. 1 = Alta prioridade.
6	R/W	0	INT1IP	Prioridade da interrupção externa no RB1: 0 = Baixa prioridade. 1 = Alta prioridade.
5	U	0	-	Não implementado.
4	R/W	0	INT2IE	Habilitação da interrupção externa no pino RB2 (chave individual): 0 = Interrupção externa desabilitada. 1 = Interrupção externa habilitada.

INTCON3				
Bit	**R/W**	**Reset**	**Nome**	**Descrição**
3	R/W	0	INT1IE	Habilitação da interrupção externa no pino RB1 (chave individual): 0 = Interrupção externa desabilitada. 1 = Interrupção externa habilitada.
2	U	0	-	Não implementado.
1	R/W	0	INT2IF	Identificação da interrupção externa no pino RB2: 0 = Não ocorreu evento da interrupção. 1 = Ocorreu evento da interrupção (este bit deve ser limpo por software).
0	R/W	0	INT1IF	Identificação da interrupção externa no pino RB1: 0 = Não ocorreu evento da interrupção. 1 = Ocorreu evento da interrupção (este bit deve ser limpo por software).

PIE1				
Bit	**R/W**	**Reset**	**Nome**	**Descrição**
7	R/W	0	PSPIE	Habilitação da interrupção da porta paralela (chave individual): 0 = Interrupção da porta paralela desabilitada. 1 = Interrupção da porta paralela habilitada.
6	R/W	0	ADIE	Habilitação da interrupção do conversor A/D (chave individual): 0 = Interrupção do conversor A/D desabilitada. 1 = Interrupção do conversor A/D habilitada.
5	R/W	0	RCIE	Habilitação da interrupção de recepção da USART (chave individual): 0 = Interrupção de recepção da USART desabilitada. 1 = Interrupção de recepção da USART habilitada.
4	R/W	0	TXIE	Habilitação da interrupção de transmissão da USART (chave individual): 0 = Interrupção de transmissão da USART desabilitada. 1 = Interrupção de transmissão da USART habilitada.
3	R/W	0	SSPIE	Habilitação da interrupção da porta serial MSSP (chave individual): 0 = Interrupção da porta serial SSP desabilitada. 1 = Interrupção da porta serial SSP habilitada.
2	R/W	0	CCP1IE	Habilitação da interrupção do módulo CCP1 (chave individual): 0 = Interrupção de CCP1 desabilitada. 1 = Interrupção de CCP1 habilitada.
1	R/W	0	TMR2IE	Habilitação da interrupção do Timer 2 (chave individual): 0 = Interrupção de Timer 2 desabilitada. 1 = Interrupção de Timer 2 habilitada.
0	R/W	0	TMR1IE	Habilitação da interrupção de estouro do Timer 1 (chave individual): 0 = Interrupção de Timer 1 desabilitada. 1 = Interrupção de Timer 1 habilitada.

PIR1				
Bit	**R/W**	**Reset**	**Nome**	**Descrição**
7	R/W	0	PSPIF	Identificação da interrupção da porta paralela (chave individual): 0 = Não ocorreu evento de leitura/escrita na porta paralela. 1 = Ocorreu evento de leitura/escrita na porta paralela (este bit deve ser limpo por software).
6	R/W	0	ADIF	Identificação da interrupção do conversor A/D (chave individual): 0 = Conversão A/D não foi completada. 1 = Conversão A/D foi completada.
5	R/W	0	RCIF	Identificação da interrupção de recepção da USART (chave individual): 0 = Buffer de recepção da USART está vazio. 1 = Buffer de recepção da USART está cheio.
4	R/W	0	TXIF	Identificação da interrupção de transmissão da USART (chave individual): 0 = Buffer de transmissão da USART está cheio. 1 = Buffer de transmissão da USART está vazio.
3	R/W	0	SSPIF	Identificação da interrupção da porta serial MSSP (chave individual): 0 = Não ocorreu condição de interrupção no módulo MSSP. 1 = Ocorreu condição de interrupção no módulo MSSP (este bit deve ser limpo por software).
2	R/W	0	CCP1IF	Identificação da interrupção do módulo CCP1 (chave individual): 0 = Não ocorreu condição de interrupção no módulo CCP1. 1 = Ocorreu condição de interrupção no módulo CCP1 (este bit deve ser limpo por software).
1	R/W	0	TMR2IF	Identificação da interrupção de Timer 2 (chave individual): 0 = Não ocorreu evento da interrupção de Timer 2. 1 = Ocorreu evento da interrupção de Timer 2 (este bit deve ser limpo por software).
0	R/W	0	TMR1IF	Identificação da interrupção de estouro do Timer 1 (chave individual): 0 = Não ocorreu estouro do Timer 1. 1 = Ocorreu estouro do Timer 1 (este bit deve ser limpo por software).

IPR1				
Bit	**R/W**	**Reset**	**Nome**	**Descrição**
7	R/W	0	PSPIP	Prioridade da interrupção da porta paralela (chave individual): 0 = Baixa prioridade. 1 = Alta prioridade.
6	R/W	0	ADIP	Prioridade da interrupção do conversor A/D (chave individual): 0 = Baixa prioridade. 1 = Alta prioridade.
5	R/W	0	RCIP	Prioridade da interrupção de recepção da USART (chave individual): 0 = Baixa prioridade. 1 = Alta prioridade.
4	R/W	0	TXIP	Prioridade da interrupção de transmissão da USART (chave individual): 0 = Baixa prioridade. 1 = Alta prioridade.
3	R/W	0	SSPIP	Prioridade da interrupção da porta serial MSSP (chave individual): 0 = Baixa prioridade. 1 = Alta prioridade.

IPR1				
Bit	**R/W**	**Reset**	**Nome**	**Descrição**
2	R/W	0	CCP1IP	Prioridade da interrupção do módulo CCP1 (chave individual): 0 = Baixa prioridade. 1 = Alta prioridade.
1	R/W	0	TMR2IP	Prioridade da interrupção de Timer 2 (chave individual): 0 = Baixa prioridade. 1 = Alta prioridade.
0	R/W	0	TMR1IP	Prioridade da interrupção de estouro do Timer 1 (chave individual): 0 = Baixa prioridade. 1 = Alta prioridade.

PIE2				
Bit	**R/W**	**Reset**	**Nome**	**Descrição**
7	R/W	0	OSCFIE	Habilitação da interrupção de falha do oscilador (chave individual): 0 = Interrupção de falha do oscilador desabilitada. 1 = Interrupção de falha do oscilador habilitada.
6	R/W	0	CMIE	Habilitação da interrupção dos comparadores de tensão (chave individual): 0 = Interrupção dos comparadores de tensão desabilitada. 1 = Interrupção dos comparadores de tensão habilitada.
5	R/W	0	-	Não implementado.
4	R/W	0	EEIE	Habilitação da interrupção de fim de escrita na E^2PROM (chave individual): 0 = Interrupção de fim de escrita na E^2PROM desabilitada. 1 = Interrupção de fim de escrita na E^2PROM habilitada.
3	R/W	0	BLCIE	Habilitação da interrupção de colisão na linha I^2C (chave individual): 0 = Interrupção de colisão na linha I^2C desabilitada. 1 = Interrupção de colisão na linha I^2C habilitada.
2	R/W	0	HLVDIE	Habilitação da interrupção de High/Low Voltage Detect (chave individual): 0 = Interrupção de High/Low Voltage Detect desabilitada. 1 = Interrupção de High/Low Voltage Detect habilitada.
1	R/W	0	TMR3IE	Habilitação da interrupção de estouro do Timer 3 (chave individual): 0 = Interrupção de Timer 3 desabilitada. 1 = Interrupção de Timer 3 habilitada.
0	R/W	0	CCP2IE	Habilitação da interrupção do módulo CCP2 (chave individual): 0 = Interrupção de CCP2 desabilitada. 1 = Interrupção de CCP2 habilitada.

PIR2				
Bit	**R/W**	**Reset**	**Nome**	**Descrição**
7	R/W	0	OSCFIF	Identificação da interrupção de falha do oscilador (chave individual): 0 = Não ocorreu falha do oscilador. 1 = Ocorreu falha do oscilador (este bit deve ser limpo por software).
6	R/W	0	CMIF	Identificação da interrupção dos comparadores de tensão (chave individual): 0 = Não ocorreu a troca de estado na saída de um dos comparadores. 1 = Ocorreu a troca de estado na saída de um dos comparadores (este bit deve ser limpo por software)

PIR2				
Bit	**R/W**	**Reset**	**Nome**	**Descrição**
5	U	0	-	Não implementado.
4	R/W	0	EEIF	Identificação da interrupção de fim de escrita na E²PROM (chave individual): 0 = Operação de escrita na E²PROM não foi completada e nova escrita não pode ser iniciada. 1 = Operação de escrita na E²PROM foi completada (este bit deve ser limpo por software).
3	R/W	0	BLCIF	Identificação da interrupção de colisão na linha I²C (chave individual): 0 = Não ocorreu colisão na linha I²C. 1 = Ocorreu colisão na linha I²C.
2	R/W	0	HLVDIF	Identificação da interrupção de High/Low Voltage Detect (chave individual): 0 = Não ocorreu a subida (ou descida) da tensão acima (ou abaixo) do valor programado. 1 = Ocorreu a subida (ou descida) da tensão acima (ou abaixo) do valor programado (este bit deve ser limpo por software).
1	R/W	0	TMR3IF	Identificação da interrupção de estouro do Timer 3 (chave individual): 0 = Não ocorreu estouro do Timer 3. 1 = Ocorreu estouro do Timer 3 (este bit deve ser limpo por software).
0	R/W	0	CCP2IF	Identificação da interrupção do módulo CCP2 (chave individual): 0 = Não ocorreu condição de interrupção no módulo CCP2. 1 = Ocorreu condição de interrupção no módulo CCP2 (este bit deve ser limpo por software).

IPR2				
Bit	**R/W**	**Reset**	**Nome**	**Descrição**
7	R/W	0	OSCFIP	Prioridade da interrupção de falha do oscilador (chave individual): 0 = Baixa prioridade. 1 = Alta prioridade.
6	R/W	0	CMIP	Identificação da interrupção dos comparadores de tensão (chave individual): 0 = Baixa prioridade. 1 = Alta prioridade.
5	U	0	-	Não implementado.
4	R/W	0	EEIP	Prioridade da interrupção de fim de escrita na E²PROM (chave individual): 0 = Baixa prioridade. 1 = Alta prioridade.
3	R/W	0	BLCIP	Prioridade da interrupção de colisão na linha I²C (chave individual): 0 = Baixa prioridade. 1 = Alta prioridade.
2	R/W	0	HLVDIP	Prioridade da interrupção de High/Low Voltage Detect (chave individual): 0 = Baixa prioridade. 1 = Alta prioridade.
1	R/W	0	TMR3IP	Prioridade da interrupção de estouro do Timer 3 (chave individual): 0 = Baixa prioridade. 1 = Alta prioridade.
0	R/W	0	CCP2IP	Prioridade da interrupção do módulo CCP2 (chave individual): 0 = Baixa prioridade. 1 = Alta prioridade.

ADCON0				
Bit	**R/W**	**Reset**	**Nome**	**Descrição**
7	U	0	-	Não implementado.
6	U	0	-	Não implementado.
5	R/W	0	CHS3	Seleção do canal para conversão: 0000 = Canal 0 (RA0/AN0) 1000 = Canal 8 (RB2/AN8) 0001 = Canal 1 (RA1/AN1) 1001 = Canal 9 (RB3/AN9) 0010 = Canal 2 (RA2/AN2) 1010 = Canal 10 (RB1/AN10) 0011 = Canal 3 (RA3/AN3) 1011 = Canal 11 (RB4/AN11) 0100 = Canal 4 (RA5/AN4) 1100 = Canal 12 (RB0/AN12) 0101 = Canal 5 (RE0/AN5) 1101 = Não implementado. 0110 = Canal 6 (RE1/AN6) 1110 = Não implementado. 0111 = Canal 7 (RE2/AN7) 1111 = Não implementado.
4	R/W	0	CHS2	
3	R/W	0	CHS1	
2	R/W	0	CHS0	
1	R/W	0	GO/DONE	Situação da conversão A/D. Válido somente quando **ADON**=1: 0 = Conversão não sendo efetuada. Este bit é limpo automaticamente quando a conversão termina. 1 = Conversão em andamento. Este bit deve ser setado para iniciar a conversão.
0	R/W	0	ADON	Ativa/Desativa o sistema de conversão A/D: 0 = Conversor desativado para economizar energia. 1 = Conversor ativado.

ADCON1				
Bit	**R/W**	**Reset**	**Nome**	**Descrição**
7	U	0	-	Não implementado.
6	U	0	-	Não implementado.
5	R/W	0	VCFG1	Configuração da tensão de referência (V_{REF-}) 0 = V_{SS} 1 = V_{REF-} (AN2)
4	R/W	0	VCFG0	Configuração da tensão de referência (V_{REF+}) 0 = V_{DD} 1 = Vr_{REF+} (AN3)
3	R/W	0	PCFG3	Configuração dos pinos analógicos/digitais.
2	R/W	0	PCFG2	
1	R/W	0	PCFG1	
0	R/W	0	PCFG0	

PCFG3: PCFG0	AN12 RB0	AN11 RB4	AN10 RB1	AN9 RB3	AN8 RB2	AN7 RE2	AN6 RE1	AN5 RE0	AN4 RA5	AN3 RA3	AN2 RA2	AN1 RA1	AN0 RA0
0000	A	A	A	A	A	A	A	A	A	A	A	A	A
0001	A	A	A	A	A	A	A	A	A	A	A	A	A
0010	A	A	A	A	A	A	A	A	A	A	A	A	A
0011	D	A	A	A	A	A	A	A	A	A	A	A	A
0100	D	D	A	A	A	A	A	A	A	A	A	A	A
0101	D	D	D	A	A	A	A	A	A	A	A	A	A
0110	D	D	D	D	A	A	A	A	A	A	A	A	A
0111	D	D	D	D	D	A	A	A	A	A	A	A	A
1000	D	D	D	D	D	D	A	A	A	A	A	A	A
1001	D	D	D	D	D	D	D	A	A	A	A	A	A
1010	D	D	D	D	D	D	D	D	A	A	A	A	A
1011	D	D	D	D	D	D	D	D	D	A	A	A	A
1100	D	D	D	D	D	D	D	D	D	D	A	A	A
1101	D	D	D	D	D	D	D	D	D	D	D	A	A
1110	D	D	D	D	D	D	D	D	D	D	D	D	A
1111	D	D	D	D	D	D	D	D	D	D	D	D	D

Pinos configurados como analógicos: A = Analógico / D = Digital.

ADCON2				
Bit	**R/W**	**Reset**	**Nome**	**Descrição**
7	R/W	0	ADFM	Justificação para o resultado da conversão A/D de 10 bits: 1 = Justificado à direita. Os 6 bits mais significativos de ADRESH são lidos em 0 (zero). 0 = Justificado à esquerda. Os 6 bits menos significativos de ADRESL são lidos em 0 (zero).
6	U	0	-	Não implementado.
5	R/W	0	ACQT2	Tempo de adequação do capacitor do S/H: 000 = 0 TAD (Tempo de adequação deve ser implementado por software) 001 = 2 TAD 010 = 4 TAD 011 = 6 TAD 100 = 8 TAD 101 = 12 TAD 110 = 16 TAD 111 = 20 TAD
4	R/W	0	ACQT1	
3	R/W	0	ACQT0	
2	R/W	0	ADSC2	Especificação do clock para o conversor A/D: 000 = F_{OSC} / 2 001 = F_{OSC} / 8 010 = F_{OSC} / 32 011 = FRC. Usado oscilador RC interno, específico para o conversor A/D. 100 = F_{OSC} / 4 101 = F_{OSC} / 16 110 = F_{OSC} / 64 111 = FRC. Usado oscilador RC interno, específico para o conversor A/D.
1	R/W	0	ADSC1	
0	R/W	0	ADCS0	

CCP1CON				
Bit	**R/W**	**Reset**	**Nome**	**Descrição**
7	R/W	0	P1M1	Configuração do modo ECCP (PWM): 00 = Saída simples: P1A modulado; P1B, P1C, P1D como pinos de I/O 01 = Saída ponte completa (Full-bridge), saída direta: P1D modulado; P1A ativo; P1B, P1C inativo 10 = Saída meia ponte (Half-bridge): P1A, P1B modulado com controle do tempo morto (dead-band); P1C, P1D como pinos de I/O 11 = Saída ponte completa (Full-bridge), saída reversa: P1B modulado; P1C ativo; P1A, P1D inativo
6	R/W	0	P1M0	
5	R/W	0	DC1B1	Parte baixa do PWM de 10 bits. A parte alta fica em **CCPR1L**. Válido somente quando em PWM.
4	R/W	0	DC1B0	
3	R/W	0	CCP1M3	Seleção do modo CCP1 - Compare/Capture/PWM: 0000 = Modo desligado. 0100 = Capture ligado para borda de descida com prescale de 1:1. 0101 = Capture ligado para borda de subida com prescale de 1:1. 0110 = Capture ligado para borda de subida com prescale de 1:4. 0111 = Capture ligado para borda de subida com prescale de 1:16. 1000 = Compare ligado. Pino de saída (RC2) será setado (1) quando o compare ocorrer. 1001 = Compare ligado. Pino de saída (RC2) será zerado (0) quando o compare ocorrer. 1010 = Compare ligado. Pino de saída (RC2) não será afetado. 1011 = Compare ligado. Pino de saída (RC2) não será afetado. TMR1/TMR3 será resetado. 1100 = PWM ligado, P1A, P1C ativo-alto; P1B, P1D ativo-alto 1101 = PWM ligado, P1A, P1C ativo-alto; P1B, P1D ativo-baixo 1110 = PWM ligado, P1A, P1C ativo-baixo; P1B, P1D ativo-alto 1111 = PWM ligado, P1A, P1C ativo-baixo; P1B, P1D ativo-baixo
2	R/W	0	CCP1M2	
1	R/W	0	CCP1M1	
0	R/W	0	CCP1M0	

CCP2CON				
Bit	**R/W**	**Reset**	**Nome**	**Descrição**
7	U	0	-	Não implementado.
6	U	0	-	Não implementado.
5	R/W	0	DC2B1	Parte baixa do PWM de 10 bits. A parte alta fica em **CCPR2L** Válido somente quando em PWM.
4	R/W	0	DC2B0	
3	R/W	0	CCP2M3	Seleção do modo CCP2 - Compare/Capture/PWM: 0000 = Modo desligado. 0100 = Capture ligado para borda de descida com prescale de 1:1. 0101 = Capture ligado para borda de subida com prescale de 1:1. 0110 = Capture ligado para borda de subida com prescale de 1:4. 0111 = Capture ligado para borda de subida com prescale de 1:16. 1000 = Compare ligado. Pino de saída (RC1) será setado (1) quando o compare ocorrer. 1001 = Compare ligado. Pino de saída (RC1) será zerado (0) quando o compare ocorrer. 1010 = Compare ligado. Pino de saída (RC1) não será afetado. 1011 = Compare ligado. Pino de saída (RC1) não será afetado. TMR1/TMR3 será resetado. Uma conversão A/D será iniciada. 1100 = PWM ligado 1101 = PWM ligado 1110 = PWM ligado 1111 = PWM ligado
2	R/W	0	CCP2M2	
1	R/W	0	CCP2M1	
0	R/W	0	CCP2M0	

PWM1CON				
Bit	**R/W**	**Reset**	**Nome**	**Descrição**
7	R/W	0	PRSEN	Configuração do auto-shutdown: 1 = O bit **ECCP1AS<ECCPASE>** é limpo automaticamente e o PWM é reiniciado. 0 = O bit **ECCP1AS<ECCPASE>** deve ser limpo manualmente para reiniciar o PWM.
6	R/W	0	PDC6	Configuração do tempo morto (dead band): $PDC = \frac{(Tdb \times Fosc)}{4}$ sendo PDC um valor entre 0 e 127 a ser carregado em **PWM1CON(PDC6:PDC0)**
5	R/W	0	PDC5	
4	R/W	0	PDC4	
3	R/W	0	PDC3	
2	R/W	0	PDC2	
1	R/W	0	PDC1	
0	R/W	0	PDC0	

ECCP1AS				
Bit	**R/W**	**Reset**	**Nome**	**Descrição**
7	R/W	0	ECCPASE	Indicação da ocorrência do auto-shutdown: 1 = Ocorreu o auto-shutdown 0 = Funcionamento normal
6	R/W	0	ECCPAS2	Seleção do pino para detecção do auto-shutdown: 111 = FLT0 ou comparador 1 ou comparador 2. 110 = FLT0 ou comparador 2. 101 = FLT0 ou comparador 1. 100 = FLT0. 011 = Comparador 1 ou comparador 2. 010 = Saída do comparador 2. 001 = Saída do comparador 1. 000 = Desabilita o auto-shutdown.
5	R/W	0	ECCPAS1	
4	R/W	0	ECCPAS0	
3	R/W	0	PSSAC1	Evento após auto-shutdown: 1x = P1A e P1C em modo tri-state. 01 = P1A e P1C em nível alto. 00 = P1A e P1C em nível baixo.
2	R/W	0	PSSAC0	
1	R/W	0	PSSBD1	Evento após auto-shutdown: 1x = P1B e P1D em modo tri-state. 01 = P1B e P1D em nível alto. 00 = P1B e P1D em nível baixo.
0	R/W	0	PSSBD0	

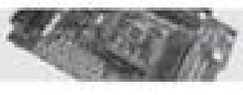
EECON1				
Bit	**R/W**	**Reset**	**Nome**	**Descrição**
7	R/W	x	EEPGD	Seleção do tipo de memória: 0 = Seleciona a memória de dados. 1 = Seleciona a memória de programa.
6	R/W	X	CFGS	Seleção do acesso à memória: 0 = Acessa a memória de dados. 1 = Acessa a memória de programa.

EECON1				
Bit	**R/W**	**Reset**	**Nome**	**Descrição**
5	U	0	-	Não implementado.
4	R/W	0	FREE	Limpeza de um bloco da memória de programa: 0 = mantém o último estado. 1 = Limpa a memória de programa (bulk erase).
3	R/W	0	WERR	Identificação de erro durante a escrita na E^2PROM: 0 = Não ocorreu erro, a escrita foi completada. 1 = Em erro ocorreu por uma escrita não terminada (um reset pode ter ocorrido).
2	R/W	0	WREN	Habilitação de escrita na E^2PROM (bit de segurança): 0 = Não habilita a escrita na E^2PROM. 1 = Habilita a escrita na E^2PROM.
1	R/W	0	WR	Ciclo de escrita na E^2PROM: 0 = Este bit será zerado pelo hardware quando o ciclo de escrita terminar (não pode ser zerado por software). 1 = Inicia o ciclo de escrita (deve ser setado por software).
0	R/W	0	RD	Ciclo de leitura da E^2PROM: 0 = Este bit será zerado pelo hardware quando o ciclo de leitura terminar (não pode ser zerado por software). 1 = Inicia o ciclo de leitura (deve ser setado por software).

SSPCON1				
Bit	**R/W**	**Reset**	**Nome**	**Descrição**
7	R/W	0	WCOL	Colisão na escrita do registrador SSPBUF: Master: 0 = Sem colisão. 1 = Ocorreu uma escrita em momento indevido. Slave: 0 = Sem colisão. 1 = Ocorreu uma escrita enquanto ainda estava transmitindo (deve ser limpo em software).
6	R/W	0	SSPOV	Erro de overflow na recepção: 0 = Sem erro. 1= Um novo byte foi recebido antes de o registrador SSPBUF ser lido (deve ser limpo em software).
5	R/W	0	SSPEN	Habilitação da porta SSP: SPI: 0 = SSP desabilitada. Pinos SCK, SDO, SDI e /SS como I/Os convencionais. 1 = SSP habilitada. Pinos SCK, SDO, SDI e /SS controlados internamente. I^2C: 0 = SSP desabilitada. Pinos SCL e SDA como I/Os convencionais. 1 = SSP habilitada. Pinos SCL e SDA controlados internamente.
4	R/W	0	CKP	Controle do Clock: SPI: 0 = Clock padrão em nível alto (1). 1 = Clock padrão em nível baixo (0). I^2C Slave: 0 = Clock habilitado. 1 = Clock travado (Pausa). Pino SCL mantido em nível baixo (0). I^2C Master: Sem uso

SSPCON1				
Bit	**R/W**	**Reset**	**Nome**	**Descrição**
3	R/W	0	SSPM3	Seleção do modo de trabalho da porta SSP: 0000 = SPI Master. Clock = F_{OSC} / 4. 0001 = SPI Master. Clock = F_{OSC} / 16. 0010 = SPI Master. Clock = F_{OSC} / 64. 0011 = SPI Master. Clock = TMR2 / 2. 0100 = SPI Slave. Pino /SS habilitado. 0101 = SPI Slave. Pino /SS desabilitado (pode ser usado como I/O). 0110 = I²C Slave. Endereçamento de 7 bits. 0111 = I²C Slave. Endereçamento de 10 bits. 1000 = I²C Master. Clock = F_{OSC} / (4 x (SSPADD+1)) Outros = Reservados
2	R/W	0	SSPM2	
1	R/W	0	SSPM1	
0	R/W	0	SSPM0	

SSPCON2				
Bit	**R/W**	**Reset**	**Nome**	**Descrição**
7	R/W	0	GCEN	Habilitação do endereço global (somente para I²C Slave): 0 = Endereço global desativado. 1 = Endereço global ativado. Quando recebido endereço 0 (zero), o Slave aceita como endereço válido.
6	R/W	0	ACKSTAT	Valor recebido como ACK após a transmissão de um dado (somente para I²C Master): 0 = ACK recebido do Slave. 1 = ACK não recebido do Slave.
5	R/W	0	ACKDT	Valor a ser respondido como ACK após uma recepção (somente para I²C Master): 0 = Continua recebendo mais bytes do Slave. 1 = O Slave deve parar de transmitir dados para o Master.
4	R/W	0	ACKEN	Gera uma condição de acknowledge com o valor de ACKDT (somente para I²C Master): 0 = Não gera condição ou condição já finalizada. 1 = Inicia condição e indica condição ainda em execução. Limpo automaticamente pelo hardware.
3	R/W	0	RCEN	Ativa o sistema de recepção de dados (somente para I²C Master): 0 = Master não pode receber dados. 1 = Master pronto para receber dados do Slave.
2	R/W	0	PEN	Gera uma condição de Stop na linha (somente para I²C Master): 0 = Não gera condição ou condição já finalizada. 1 = Inicia condição e indica condição ainda em execução. Limpo automaticamente pelo hardware.
1	R/W	0	RSEN	Gera uma condição de Restart na linha (somente para I²C Master): 0 = Não gera condição ou condição já finalizada. 1 = Inicia condição e indica condição ainda em execução. Limpo automaticamente pelo hardware.
0	R/W	0	SEN	Gera uma condição de Start na linha (somente para I²C Master): 0 = Não gera condição ou condição já finalizada. 1 = Inicia condição e indica condição ainda em execução. Limpo automaticamente pelo hardware.



SSPSTAT				
Bit	**R/W**	**Reset**	**Nome**	**Descrição**
7	R/W	0	SMP	Sistema de amostragem de dados: SPI: 0 = A recepção é feita na borda do meio do período. Modo obrigatório para Slave. 1 = A recepção é feita na borda do fim do período. Pode ser usado no Master. I²C: 0 = Habilita controle de sinal para alta velocidade (400 kHz). 1 = Desabilita controle de sinal para alta velocidade (100 kHz).
6	R/W	0	CKE	Sistema de amostragem de dados: SPI: 0 = Pulso do clock no início do período. 1 = Pulso do clock no final do período. I²C: 0 = Entradas conforme especificações I²C. 1 = Entradas conforme especificações SMBUS.
5	R	0	D/A	Indicação de Dados ou Endereço (somente para I²C): 0 = Último byte recebido ou transmitido foi um endereço. 1 = Último byte recebido ou transmitido foi um dado.
4	R	0	P	Indicação de Stop (somente para I²C): 0 = Não ocorreu uma condição de Stop. 1 = Ocorreu uma condição de Stop. Limpo quando SSP desabilitada.
3	R	0	S	Indicação de Start (somente para I²C): 0 = Não ocorreu uma condição de Start. 1 = Ocorreu uma condição de Start. Limpo quando SSP desabilitada.
2	R	0	R/W	Indicação de Leitura/Escrita (somente para I²C): Slave: 0 = Escrita. 1 = Leitura. Master: 0 = Nenhuma transmissão em progresso. 1 = Trans. em progresso. Limpo automaticamente pelo hardware no final.
1	R	0	UA	Indicação de atualização de endereço (somente para I²C com 10 bits): 0 = Não precisa atualizar o endereço. 1 = Deve atualizar o endereço (parte alta ou parte baixa) para nova comparação.
0	R	0	BF	Indicação de Buffer cheio: Recebe: 0 = Recepção ainda não foi completada. Buffer **SSPBUF** vazio. 1 = Recepção terminada. Buffer **SSPBUF** cheio. Transmite (I²C): 0 = Transmissão terminada. Buffer **SSPBUF** vazio. 1 = Transmissão em progresso. Buffer **SSPBUF** ainda cheio.

TXSTA				
Bit	**R/W**	**Reset**	**Nome**	**Descrição**
7	R/W	0	CSRC	Seleção entre Master/Slave (somente modo Síncrono): 0 = Slave. 1 = Master.
6	R/W	0	TX9	Habilitação da comunicação em 9 bits para a transmissão: 0 = Transmissão em 8 bits. 1 = Transmissão em 9 bits.
5	R/W	0	TXEN	Habilitação da transmissão: 0 = Transmissão desabilitada. 1 = Transmissão habilitada. No modo síncrono, a recepção tem prioridade sobre este bit.

TXSTA				
Bit	**R/W**	**Reset**	**Nome**	**Descrição**
4	R/W	0	SYNC	Seleção entre modo Assíncrono/Síncrono: 0 = Assíncrono. 1 = Síncrono.
3	R/W	0	SENDB	Envio do caractere de Break (somente modo Assíncrono): 0 = Caractere de Break enviado. 1 = Envia caractere de Break na próxima transmissão (bit limpo por hardware).
2	R/W	0	BRGH	Seleção para Baud Rate (somente modo Assíncrono): 0 = Baud Rate baixo. 1 = Baud Rate alto.
1	R	1	TRMT	Situação do registrador interno de transmissão (TSR): 0 = TSR cheio. 1 = TSR vazio.
0	R/W	0	TX9D	Valor a ser transmitido como 9º bit. Pode ser usado como paridade ou endereçamento.

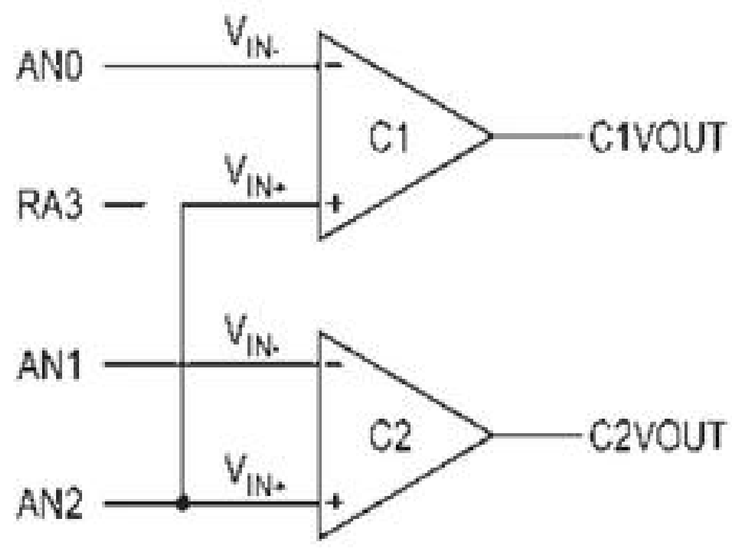
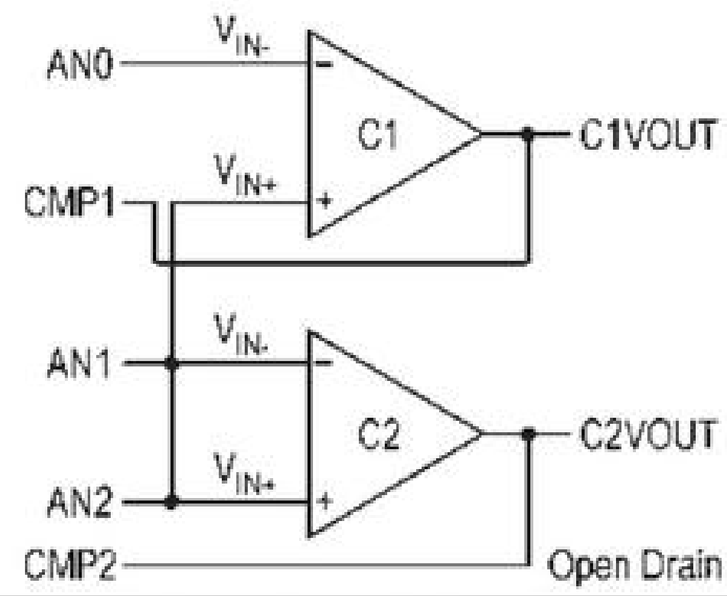
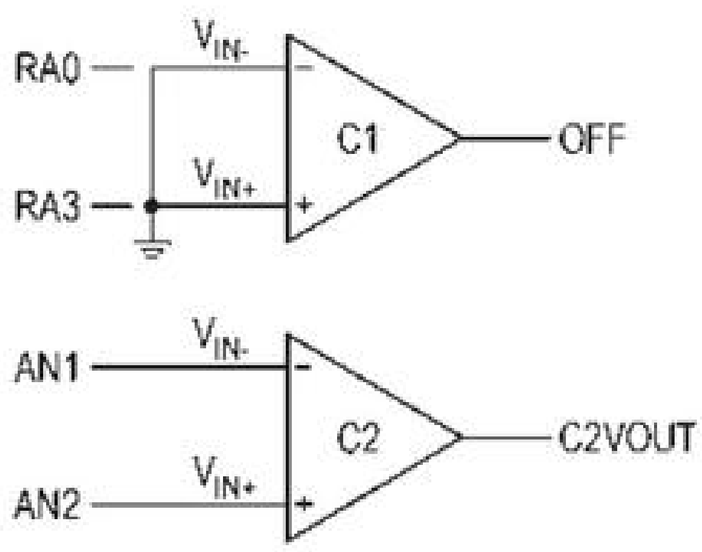
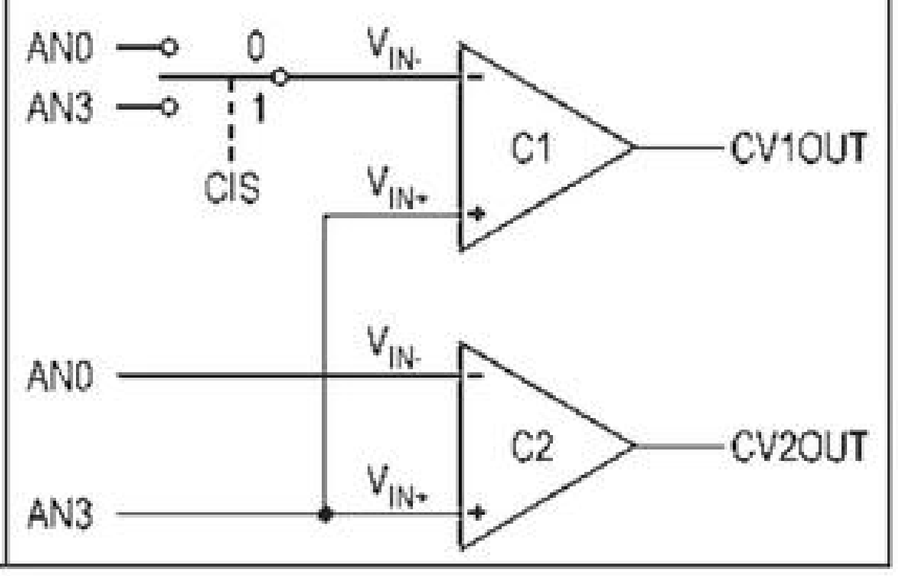
RCSTA				
Bit	**R/W**	**Reset**	**Nome**	**Descrição**
7	R/W	0	SPEN	Habilitação da USART: 0 = USART desabilitada. 1 = USART habilitada.
6	R/W	0	RX9	Habilitação da comunicação em 9 bits para a recepção: 0 = Recepção em 8 bits. 1 = Recepção em 9 bits.
5	R/W	0	SREN	Habilitação da recepção unitária (somente para modo Síncrono em Master): 0 = Recepção unitária desabilitada. 1 = Recepção unitária habilitada. Depois de receber um dado, desliga-se automaticamente.
4	R/W	0	CREN	Habilitação da recepção contínua: 0 = Recepção contínua desabilitada. 1 = Recepção contínua habilitada.
3	R/W	0	ADDEN	Habilitação do sistema de endereçamento (somente modo Assíncrono de 9 bits): 0 = Desabilita sistema de endereçamento. 1 = Habilita sistema de endereçamento.
2	R	0	FERR	Erro de Stop bit (somente modo Assíncrono): 0 = Não ocorreu erro. Stop bit = 1. 1 = Ocorreu um erro. Stop bit = 0 (deve ser atualizado lendo o registrador RCREG e recebendo o próximo dado válido).
1	R	0	OERR	Erro de muitos bytes recebidos sem nenhuma leitura: 0 = Não houve problemas de estouro do limite. 1 = Estouro do limite de 3 bytes recebidos antes da leitura de RCREG (para limpar deve-se zerar o bit CREN).
0	R	x	RX9D	Valor recebido no 9º bit. Pode ser usado como paridade ou endereçamento.

BAUDCON				
Bit	**R/W**	**Reset**	**Nome**	**Descrição**
7	R/W	0	ABDOV	Monitoramento do estouro do contador de Baud Rate: 0 = Não ocorreu estouro do contador. 1 = Ocorreu estouro do contador (valor maior que 65535).
6	R	1	RCIDL	Monitoramento da recepção de dados: 0 = Recepção ativa. 1 = Aguardando dados para ser recebido.
5	R/W	0	RXDTP	Configuração da polaridade do dado recebido: Modo Assíncrono: 0 = Pino RX ativo alto 1 = Pino RX ativo baixo Modo Síncrono: 0 = Pino DT ativo alto 1 = Pino DT ativo baixo
4	R/W	0	TXCKP	Configuração do estado normal do pino de transmissão: Modo Assíncrono: 0 = Estado normal do pino TX em 1 1 = Estado normal do pino TX em 0 Modo Síncrono: 0 = Estado normal do pino CK em 1 1 = Estado normal do pino Ck em 0
3	R/W	0	BRG16	Habilitação do contador de Baud Rate em 16 bits: 0 = Contador de Baud Rate em 8 bits. 1 = Contador de Baud Rate em 16 bits.
2	U	0	-	Não implementado.
1	R/W	0	WUE	Configuração do auto wake-up (somente modo Assíncrono): 0 = Não acorda microcontrolador no modo Sleep. 1 = O próximo dado tira microcontrolador no modo Sleep (bit limpo automaticamente).
0	R/W	0	ABDEN	Habilitação do recebimento do caractere de sincronismo: 0 = Sincronismo desabilitado ou completado. 1 = Aguarda caractere de sincronismo (bit limpo automaticamente após recebimento do caractere).

CMCON				
Bit	**R/W**	**Reset**	**Nome**	**Descrição**
7	R/W	0	C2OUT	Valor da saída do comparador 2: Normal (**C2INV**=0): 0 = C2 V_{IN+} < C2 V_{IN-} 1 = C2 V_{IN+} > C2 V_{IN-} Inversa (**C2INV**=1): 0 = C2 V_{IN+} > C2 V_{IN-} 1 = C2 V_{IN+} < C2 V_{IN-}
6	R/W	0	C1OUT	Valor da saída do comparador 1: Normal (**C1INV**=0): 0 = C1 V_{IN+} < C1 V_{IN-} 1 = C1 V_{IN+} > C1 V_{IN-} Inversa (**C1INV**=1):

CMCON				
Bit	**R/W**	**Reset**	**Nome**	**Descrição**
				0 = C1 V_{IN+} > C1 V_{IN-} 1 = C1 V_{IN+} < C1 V_{IN-}
5	R/W	0	C2INV	Tipo de saída do comparador 2: 0 = Normal. 1 = Inversa.
4	R/W	0	C1INV	Tipo de saída do comparador 1: 0 = Normal. 1 = Inversa.
3	R/W	0	CIS	Chave seletora de entrada do comparador: Quando **CM2:CM0** = 001 0 = RA0 conectado a C1 V_{IN-}. 1 = RA3 conectado a C1 V_{IN-}. Quando **CM2:CM0** = 010 0 = RA0 conectado a C1 V_{N-}, RA1 conectado a C2 V_{N-}. 1 = RA3 conectado a C1 V_{N-}, RA2 conectado a C2 V_{IN-}.
2	R/W	0	CM2	Configura a pinagem dos comparadores (modo de operação):
1	R/W	0	CM1	
0	R/W	0	CM0	

CM	Esquema	CM	Esquema
000	Reset dos comparadores AN0, AN3, V_{IN-}, V_{IN+}, C1, OFF AN1, AN2, V_{IN-}, V_{IN+}, C2, OFF	111	Comparadores desligados RA0, RA3, RA1, RA2, V_{IN-}, V_{IN+}, C1, C2, OFF
100	2 comparadores independentes AN0, AN3, V_{IN-}, V_{IN+}, C1, C1VOUT AN1, AN2, V_{IN-}, V_{IN+}, C2, C2VOUT	010	4 entradas multiplexadas AN0, AN3, 0, 1, CIS, V_{IN-}, V_{IN+}, C1, CV1OUT AN0, AN3, 0, 1, V_{IN-}, V_{IN+}, C2, CV2OUT para V_{REF}

CM	Esquema	CM	Esquema
011	2 comparadores com ref. comum  	110	2 comparadores com ref. comum e saídas 
011	1 comparador independente  	110	3 entradas multiplexadas 

CVRCON				
Bit	**R/W**	**Reset**	**Nome**	**Descrição**
7	R/W	0	CVREN	Energização do sistema de tensão de referência: 0 = Circuito de V_{REF} desenergizado. 1 = Circuito de V_{REF} energizado.
6	R/W	0	CVRON	Habilitação da saída de V_{REF}: 0 = Tensão de referência desligada. 1 = Tensão de referência ligada ao pino RA2.
5	R/W	0	CVRR	Seleção do range de operação do sistema de V_{REF}: 0 = Range baixo. 1 = Range alto.
4	U	0	-	Não implementado.
3	R/W	0	CVR3	Seleção do valor da tensão de V_{REF}: Se **CVRR** = 1: $V_{REF} = (VR/24) * V_{DD}$ Se **CVRR** = 0: $V_{REF} = ¼ * V_{DD} + (VR/32) * V_{DD}$
2	R/W	0	CVR2	
1	R/W	0	CVR1	
0	R/W	0	CVR0	

HLVDCON				
Bit	**R/W**	**Reset**	**Nome**	**Descrição**
7	R/W	0	VDIRMAG	Direção de detecção do evento: 0 = Tensão de entrada (Vcc) ≤ tensão de referência. 1 = Tensão de entrada (Vcc) ≥ tensão de referência.
6	U	0	-	Não implementado.
5	R/W	0	IRVST	Monitoração da estabilidade da referência de tensão do módulo: 0 = Tensão não estabilizada. 1 = Tensão estabilizada.
4	R/W	0	HLVDEN	Habilitação do módulo High/Low Voltage Detect 0 = Módulo desabilitado. 1 = Módulo habilitado.
3	R/W	0	HLVDL3	Tensão de comparação do módulo: 0000 = 2,17 V 1000 = 3,39 0001 = 2,23 V 1001 = 3,55 0010 = 2,36 V 1010 = 3,71 0011 = 2,44 V 1011 = 3,90 0100 = 2,60 V 1100 = 4,11 0101 = 2,79 V 1101 = 4,33 0110 = 2,89 V 1110 = 4,59 0111 = 3,12 V 1111 = Externo por meio do pino HLVDIN
2	R/W	1	HLVDL2	
1	R/W	0	HLVDL1	
0	R/W	1	HLVDL0	

B

Detalhamento dos Configurations Bits

» Introdução

Este apêndice apresenta um detalhamento de todos os Configurations Bits do PIC18F4520.

Oscilador	
Lista	**Descrição**
OSC = LP	Oscilador LP
OSC = XT	Oscilador XT
OSC = HS	Oscilador HS
OSC = RC	Oscilador RC externo
OSC = EC	Oscilador EC
OSC = ECIO6	Oscilador EC, pino RA6 como I/O
OSC = HSPLL	Oscilador HS com PLL
OSC = RCIO6	Oscilador RC externo, pino RA6 como I/O
OSC = INTIO67	Oscilador RC interno, pino RA6 e RA7 como I/O
OSC = INTIO7	Oscilador RC interno, pino RA7 como I/O

Fail-Safe Clock Monitor (FSCM)	
Lista	**Descrição**
FCMEN = OFF	FSCM desligado
FCMEN = ON	FSCM ligado

Chaveamento do Oscilador Interno/Externo	
Lista	**Descrição**
IESO = OFF	Chaveamento desligado
IESO = ON	Chaveamento ligado

Habilitação do Power-Up Timer	
Lista	**Descrição**
PWRT = OFF	Power-up Timer desabilitado
PWRT = ON	Power-up Timer habilitado

Habilitação do Brown-Out Reset	
Lista	**Descrição**
BOREN = OFF	Brown-out Reset desabilitado
BOREN = ON	Brown-out Reset habilitado por hardware e controlado por software
BOREN = NOSLP	Brown-out Reset habilitado por hardware e desabilitado no modo SLEEP
BOREN = SDORDIS	Brown-out Reset habilitado por hardware

Tensão do Brown-Out Reset	
Lista	**Descrição**
BORV = 0	Valor máximo
BORV = 1	...
BORV = 2	...
BORV = 3	Valor mínimo

Habilitação do Watchdog Timer	
Lista	**Descrição**
WDT = OFF	Watchdog Timer desabilitado
WDT = ON	Watchdog Timer habilitado

Potscaler do Watchdog Timer	
Lista	**Descrição**
WDTPS = 1	1:1
WDTPS = 2	1:2
WDTPS = 4	1:4
WDTPS = 8	1:8
WDTPS = 16	1:16
WDTPS = 32	1:32
WDTPS = 64	1:64
WDTPS = 128	1:128
WDTPS = 256	1:256
WDTPS = 512	1:512
WDTPS = 1024	1:1024
WDTPS = 2048	1:2048
WDTPS = 4096	1:4096
WDTPS = 8192	1:8192
WDTPS = 16384	1:16384
WDTPS = 32768	1:32768

Habilitação do Master Clear	
Lista	**Descrição**
MCLR = OFF	Master Clear desabilitado
MCLR = ON	Master Clear habilitado

Oscilador do Timer1	
Lista	**Descrição**
LPT1OSC = OFF	Timer 1 configurado como oscilador de consumo normal
LPT1OSC = ON	Timer 1 configurado como oscilador de consumo baixo

Habilitação dos Canais A/D do PORTB	
Lista	**Descrição**
PBADEN = OFF	Pinos RB0 ao RB4 como I/O após reset
PBADEN = ON	Pinos RB0 ao RB4 como analógico após reset

MUX CCP2	
Lista	**Descrição**
CCP2MX = PORTBE	Pinos RB3 como CCP2
CCP2MX = PORTC	Pinos RC1 como CCP2

Reset por Stack Full/Underflow	
Lista	**Descrição**
STVREN = OFF	Reset por Stack Full/Underflow desabilitado
STVREN = ON	Reset por Stack Full/Underflow habilitado

Gravação em Baixa Tensão	
Lista	**Descrição**
LVP = OFF	Gravação em baixa tensão desabilitada
LVP = ON	Gravação em baixa tensão habilitada

Habilitação do Set de Instruções em Modo Estendido	
Lista	**Descrição**
XINST = OFF	Modo estendido desabilitado
XINST = ON	Modo estendido habilitado

Modo de Depuração	
Lista	**Descrição**
DEBUG = OFF	Modo de depuração desabilitado
DEBUG = ON	Modo de depuração habilitado

Proteção de Código no Bloco 0	
Lista	**Descrição**
CP0 = ON	Bloco 0 (800h – 1FFFh)
CP0 = OFF	Bloco 0 (800h – 1FFFh)

Proteção de Código no Bloco 1	
Lista	**Descrição**
CP1 = ON	Bloco 1 (2000h – 3FFFh)
CP1 = OFF	Bloco 1 (2000h – 3FFFh)

Proteção de Código no Bloco 2	
Lista	**Descrição**
CP2 = ON	Bloco 2 (4000h – 5FFFh)
CP2 = OFF	Bloco 2 (4000h – 5FFFh)

Proteção de Código no Bloco 3	
Lista	**Descrição**
CP3 = ON	Bloco 3 (6000h – 7FFFh)
CP3 = OFF	Bloco 3 (6000h – 7FFFh)

Proteção de Código no Bloco Boot	
Lista	**Descrição**
CPB = ON	Bloco boot (0h – 7FFh)
CPB = OFF	Bloco boot (0h – 7FFh)

Proteção de Dados na EEPROM	
Lista	**Descrição**
CPD = ON	Dados da EEPROM protegidos
CPD = OFF	Dados da EEPROM não protegidos

Proteção de Escrita no Bloco 0	
Lista	**Descrição**
WRT0 = ON	Bloco 0 (800h – 1FFFh)
WRT0 = OFF	Bloco 0 (800h – 1FFFh)

Proteção de Escrita no Bloco 1	
Lista	**Descrição**
WRT1 = ON	Bloco 1 (2000h – 3FFFh)
WRT1 = OFF	Bloco 1 (2000h – 3FFFh)

Proteção de Escrita no Bloco 2	
Lista	**Descrição**
WRT2 = ON	Bloco 2 (4000h – 5FFFh)
WRT2 = OFF	Bloco 2 (4000h – 5FFFh)

Proteção de Escrita no Bloco 3	
Lista	**Descrição**
WRT3 = ON	Bloco 3 (6000h – 7FFFh)
WRT3 = OFF	Bloco 3 (6000h – 7FFFh)

Proteção de Escrita no Bloco Boot	
Lista	**Descrição**
WRTB = ON	Bloco boot (0h – 7FFh)
WRTB = OFF	Bloco boot (0h – 7FFh)

Proteção de Escrita na EEPROM	
Lista	**Descrição**
WRTD = ON	Dados da EEPROM protegidos contra escrita
WRTD = OFF	Dados da EEPROM não protegidos contra escrita

Proteção de Leitura de Tabela no Bloco 0	
Lista	**Descrição**
EBTR0 = ON	Bloco 0 (800h – 1FFFh)
EBTR0 = OFF	Bloco 0 (800h – 1FFFh)

Proteção de Leitura de Tabela no Bloco 1	
Lista	**Descrição**
EBTR1 = ON	Bloco 1 (2000h – 3FFFh)
EBTR1 = OFF	Bloco 1 (2000h – 3FFFh)

Proteção de Leitura de Tabela no Bloco 2	
Lista	**Descrição**
EBTR2 = ON	Bloco 2 (4000h – 5FFFh)
EBTR2 = OFF	Bloco 2 (4000h – 5FFFh)

Proteção de Leitura de Tabela no Bloco 3	
Lista	**Descrição**
EBTR3 = ON	Bloco 3 (6000h – 7FFFh)
EBTR3 = OFF	Bloco 3 (6000h – 7FFFh)

Proteção de Leitura de Tabela no Bloco Boot	
Lista	**Descrição**
EBTRB = ON	Bloco boot (0h – 7FFh)
EBTRB = OFF	Bloco boot (0h – 7FFh)

C

Esquema Elétrico da Placa Proposta (McLab2)

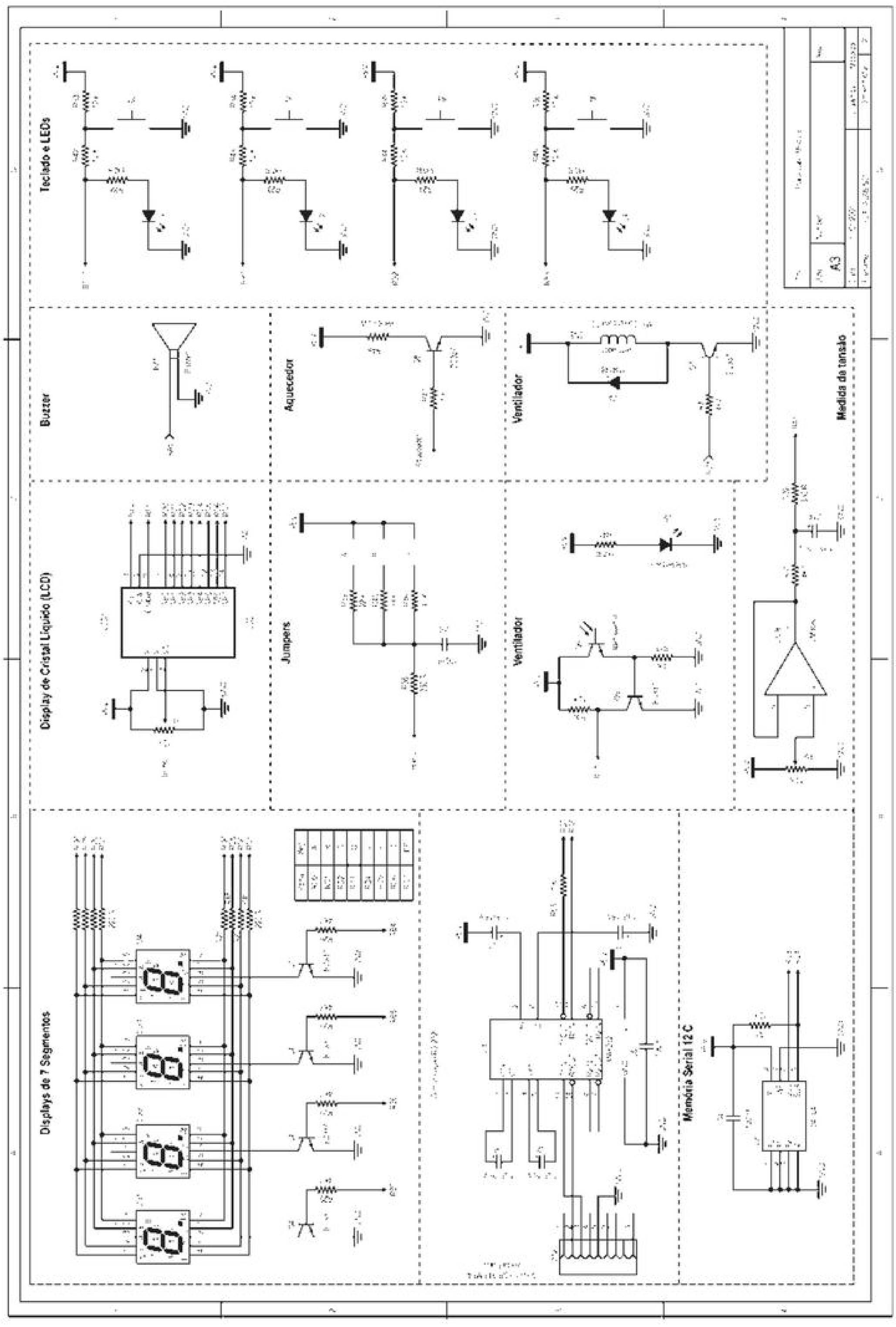
Sugerimos um hardware para a realização dos treinamentos e projetos presentes neste livro. Com o esquema elétrico apresentado nas próximas páginas, é possível a montagem de uma placa capaz de realizar todas as experiências deste livro e muitos outros experimentos com o PIC18F4520 (ou qualquer outro de 40 pinos).

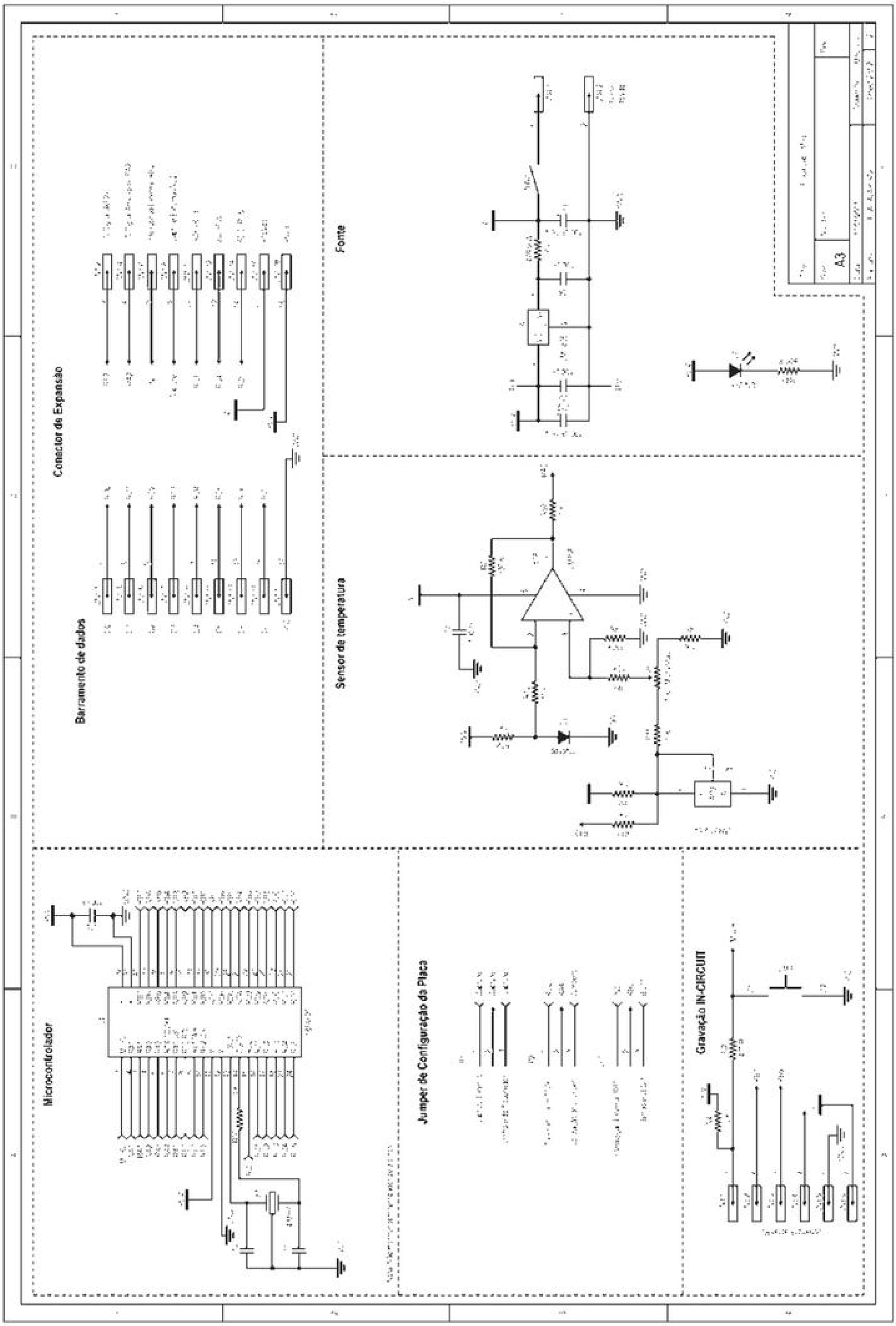
Sua placa vai possuir então os seguintes recursos:

- ✓ Soquete para montagem do PIC de 40 pinos;
- ✓ Quatro displays de sete segmentos com controle por varredura;
- ✓ Display de cristal líquido (LCD) com duas linhas e 16 colunas;
- ✓ Quatro teclas e quatro LEDs para entradas e saídas;
- ✓ Buzzer e jumpers;
- ✓ Memória E^2PROM serial externa do tipo 24C04 (ou outra similar);
- ✓ Comunicação RS-232;
- ✓ Conector para gravação In-Circuit;
- ✓ Sensor de temperatura;
- ✓ Aquecedor via resistor de potência;
- ✓ Miniventilador para resfriamento da resistência ou experiências com sensor óptico;
- ✓ Sensor (transmissor e receptor) óptico infravermelho;
- ✓ Conector de expansão.

Para facilitar ainda mais o seu trabalho, este esquema é o mesmo de um produto denominado McLab2 (LabTools), que pode ser facilmente encontrado no mercado, evitando a necessidade de montar o hardware proposto para poder executar todos os exercícios apresentados neste livro.

Editora Érica - Desbravando o Microcontrolador PIC18 Recursos Avançados - Daniel R. de Sousa, David José de Souza e Nicolás César Lavinia - 1ª Edição


Teclado e LEDs
Buzzer
Aquecedor
Ventilador
Medida de tensão
Display de Cristal Líquido (LCD)
Jumpers
Ventilador
Displays de 7 Segmentos
Memória Serial I2C
A3


Conector de Expansão
Barramento de dados
Fonte
Sensor de temperatura
Microcontrolador
Jumper de Configuração da Placa
Gravação IN-CIRCUIT
A3

» Bibliografia

SOUZA, D. J. **Desbravando o PIC**. São Paulo: Érica, 2000.

SOUZA, D. J.; LAVINIA, N. C. **Conectando o PIC 16F877A:** *Recursos Avançados*. São Paulo: Érica, 2003.

MICROCHIP TECHNOLOGY. **Data Sheet PIC16F877A**. USA, 2001.

_____. **Data Sheet PIC18F4520**. USA, 2008.

_____. **PICMICRO 18C FAMILY REFERENCE MANUAL**. USA, 2000.

_____. **MPASM Assembler User's Guide**. USA, 2001.

_____. **MPLAB User's Guide V6.22**. USA, 2003.

_____. **Home page (Internet) e Technical Library (CD-ROM).** USA, 2003.

_____. **Product Line Card**. Catálogo.USA, 2003.

Sites

http://www.microchip.com/mplap (link com informações e download do MPLAB)

http://www.microchip.com/pic18 (link com informações sobre a família PIC18)

http://www.microchip.com/memory (link com informações sobre memórias seriais)

http://www.microchip.com/applicationnotes (link com notas de aplicação Microchip)

» Marcas registradas

Os nomes PIC, PICmicro, Microchip, In-Circuit Serial Programming, MpLab e MPASM são propriedades da Microchip Technology Inc. nos Estados Unidos e em outros países.

Todos os demais nomes registrados, marcas registradas ou direitos de uso citados neste livro pertencem aos respectivos proprietários.

» Índice remissivo

A

A/D · 33, 111, 131
Access Bank · 29
Acknowledge · 199
ACON0 · 119
ADCON1 · 47, 49, 119
ADDEN · 242, 244
ADIE · 119, 147, 148, 196, 208, 214, 242, 244
ADIF · 119, 147, 148, 196, 208, 214, 242, 244
ADON · 119
ADRESH · 119

B

Banco · 28
Baud Rate · 234
Bit · 39
BRGH · 242, 244
Bronw-out · 264
  Reset · 264
BSR · 25, 28
Buzzer · 59

C

CAD · 111
CALL · 27
Capture · 34, 145, 146
Caractere de
  break · 244
  sincronismo · 244
CCP · 34
CCP1CON · 147, 153, 156
CCP1IE · 36
CCP2CON · 147
CCPR1H · 147, 153, 156
CCPR1L · 147, 153, 156
CCPR2H · 147
CCPR2L · 147
CCPs · 145
CGRAM · 96
Ciclos de máquina · 25
Circuito RC · 131
CKE · 196, 209, 214
Clear · 40
Clock · 195
  externo · 25
  interno · 25
CLRWDT · 262
Code Protection · 270
Comparadores analógicos · 260
Compare · 34, 145
Comunicação paralela · 257
Conversor
  analógico · 111
  analógico-digital · 111
CREN · 242, 244
Cristal · 266
CS · 48

D

Destino · 39
Displays numéricos · 70
Duty cycle · 274

E

E²PROM · 26, 31, 169
ECCP · 153
ECCP1AS · 156
EEADR · 173
EECON · 173
EECON1 · 170
EECON2 · 170, 171
EEPGD · 173
Escravo · 192

F

Fail Save Clock Monitor · 269
FERR · 242, 244
File · 39

FLASH · 26, 169, 172
FSR · 29
Full Duplex · 234

G

Gerador de tensão de referência · 261
GIE · 36
GO/DONE · 119
Gravação In-Circuit · 271

H

Half Duplex · 236
High/Low Voltage Detect (HLVD) · 259
HLVDCON · 259

I

I²C · 33, 191, 197
IDLE · 269
IDs · 270
In-Circuit · 271
INDF · 29
INTCON · 36
Interrupções · 26, 32

L

LCD · 89
LED · 59
LIN · 244
Literal · 39
Low Voltage Programming · 272

M

Máquina · 25
Master · 191, 195, 243
Maximização de pino · 57
Memória de
  dados · 25, 28
  programa · 26
Modo síncrono · 238
MSSP · 33, 191
Multiplicação · 24

O

OERR · 242, 244
OPTION_REG · 47, 263
OPTION_REG<T0SE>: · 49
Oscilador · 267
  Start-up Timer · 264
Overflow · 259

P

Paralela · 257
Paridade · 236
PCFG0 · 47, 49
PCFG1 · 47, 49
PCFG2 · 47, 49
PCFG3 · 47, 49
PCH · 24
PCL · 24
PCU · 24
PIE1 · 36
Pilha (stack) · 27
Pipeline · 25
POR · 263
PORTA · 21, 38, 47
PORTB · 22, 38, 47
PORTC · 22, 48, 147, 149, 153, 156, 196, 209, 214, 242, 244
PORTD · 23, 48
PORTE · 44, 48
Postscale · 54, 152
Power-down
  Mode · 264
  Reset · 263
  Timer · 263
Prescale · 49
Program Counter · 24
PSA · 51
PSP · 33, 257
PSPMODE · 49
Pull-up · 47
PWM · 34, 145, 149, 274
PWM1CON · 156
PWRT · 263

R

RA0 · 47
RA1 · 47
RA2 · 47
RA3 · 47
RA4 · 47
RA5 · 47
RAM · 25, 28
RB4 · 32
RB5 · 32
RB6 · 32
RB7 · 32
RC · 131
RC0 · 48
RC1 · 48
RC2 · 48
RC3 · 48
RC4 · 48
RC5 · 48
RC6 · 48
RC7 · 48
RCIE · 36
RCREG · 244
RCSTA · 242, 244
RD · 48, 173
Registradores especiais · 28
Reset · 55
Ressonador · 266
Re-Start · 199
RS-232 · 274
RW · 48
RX · 234

S

Schmitt Trigger · 59
Set · 40
    de instrução · 41
SFRs · 31
Skip · 40
Slave · 191, 194, 243
SLEEP · 147, 153, 241, 264, 269
SPBRG · 244
SPBRGH · 244
SPI · 33, 192
SSPBUF · 196
SSPCON · 196, 209, 214
SSPIE · 36
SSPM1 · 196, 209, 214
SSPM2 · 196, 209, 214
SSPSTAT · 196, 209, 214
Start · 198
Start Bit · 235
STATUS · 263, 266
Stop · 198
Stop Bit · 235

T

T0CKI · 49
T0CS · 47, 51
T0IF · 36
T0SE · 51
Tabela ASCII · 96
Table Pointer · 24
TBLAT · 170
TBLPTRH · 170
TBLPTRL · 170
TBLPTRU · 170
TBLRD · 170
TBLWT · 170
Teste · 39
Timer0 · 49
Timer1 · 51
Timer2 · 54
Timer3 · 55
TMR0 · 32, 49
TMR1 · 73
TMR1IE · 36
TMR2IE · 36
Trigger · 45
TRISA · 47, 119
TRISC · 147, 149, 153, 156, 196, 209, 214, 242, 244
TX · 234
TX9D · 242, 244
TXREG · 244

U

Unidade lógica aritmética (ALU) · 24
USART · 274

V

Vetor de
interrupção · 26
reset · 26

W

Wake-up · 264
Watchdog Timer · 262
WCOL · 196, 209, 214
WDT · 262
Work · 39
WR · 173
WREN · 173
WRERR · 173

Z

Zero · 40