INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SUDESTE DE MINAS GERAIS
Campus Juiz de Fora

2017

Apostila de Práticas de Sistemas Digitais para Mecatrônica

Prof. Thiago da Silva Castro

27/3/2017

Sumário

Introdução a Sistemas Microcontrolados

Sistemas digitais é uma cadeira de grande importância dentro da área de engenharia. Na engenharia mecatrônica ela tem um papel primordial dada ao foco eletrônico do curso. Entender os microcontroladores dá ao engenheiro um maior conhecimento sobre os processos industriais, sobre os sistemas eletromecânicos, lhe dando condição para desenvolver seus métodos e soluções.

São inúmeros os tipos de microcontroladores existentes no mundo. Apesar dessa variedade, grande parte deles compartilham as mesmas características, a mesma arquiterura e forma de contrução. Logo o estudo de microcontroladores se foca em alguns tipos, e uma vez entendido os conceitos básicos, se torna simples a migração para outros modelos.

Essa apostila está direcionada para o microcontrolador da Microchip, modelo PIC18F4550. Como compilador utilizou-se o MPLAB IDE versão 8.76, compatível com a placa de desenvolvimento integrado (IDE) da empresa Mosaico. Entretanto a metodologia utilizada poderia servir para qualquer outro modelo.

Todas as práticas foram desenvolvidas pelo professor Thiago da Silva Castro, exeto prática 10, desenvolvida pelo professor Márcio do Carmo Barbosa Poncílio Rodrigues.

Prática 1 – Ambiente Integrado do MPLAB

1) Introdução

O MPLAB IDE é um software desenvolvido para Windows para o desenvolvimento de aplicações para os Microcontroladores e Controladores Digitais de Sinais (DSC) da Microchip. Ele é chamado de um Ambiente Integrado de Desenvolvimento, ou IDE *(Integrated Development Environment*), porque fornece um único "ambiente" para desenvolver, compilar, programar e debugar um código para microcontroladores.

Para realizar a programação em um microcontrolador PIC, alguns programas deverão ser configurados no microcontrolador. Dentre eles, um programa importante é o MPLAB C18, um software responsável por realizar a compilação do código em C para a linguagem de máquina a ser gravado no PIC.

Ambos os programas podem ser obtidos no site do fabricante, em versão de estudante que possui limitações, mas não serão problemas para os trabalhos propostos na disciplina.

2) Desenvolvimento Teórico

O software MPLAB IDE é uma poderosa ferramenta para criação de programas para o microcontrolador, que permite trabalhar com uma grande quantidade de chips desenvolvidos pela microchip, e é bastante versátil. Entretanto, tal versatilidade requer conhecimentos do programador para configurar exatamente o modelo que ele quiser trabalhar, compiladores e gravador.

Todos os programas desenvolvidos nessa IDE deverão estar dispostos em forma de “projeto”. Um projeto é associação de grupos de arquivos com as ferramentas da linguagem de programação. De maneira prática, é arquivo com extensão .MCP que fará a ligação entre os demais arquivos do programa. Esse arquivo de projeto terá informações relacionadas ao mirocontrolador escolhido, o caminho que os compiladores estão instalados, e o caminho dos códigos em C. o produto final de um projeto, ou seja a saída, será um arquivo com extensão .HEX, que contém as instruções a serem gravadas no microcontrolador.

Aconselha-se que para cada projeto seja criada uma nova pasta, e dentro dela seja armazenado todos os arquivos do projeto (os códigos fonte, objeto, headers do projeto). Como o projeto cria links para os arquivos, a chance de arquivos fontes ficarem espalhados pelo computador é muito grande, e pode-se gerar confusão na hora de editar esses arquivos.

3) Objetivos

- Criar um projeto utilizando a IDE MPLAB e configurá-lo de maneira adequada.
- Configurar o gravador ICD2 para esse projeto.
- Desenvolver um arquivo principal "padrão" (*template*) a ser utilizado como início nos demais programas do curso

4) Desenvolvimento

- Criando um projeto

Aconselha-se em um primeiro momento a ferramenta *Project Wizard* para a criação de um projeto no MPLAB IDE. Na Figura 1 está apresentado o caminho "*Project*", onde está localizado a opção "*Project Wizard...*"

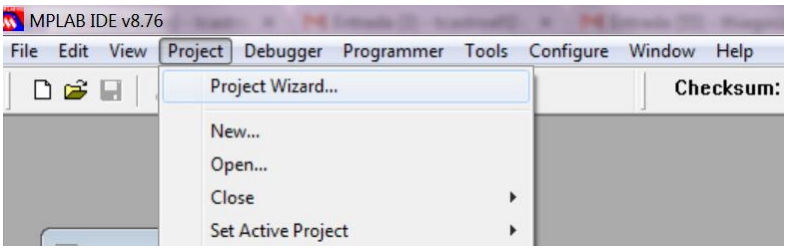

Figura 1 - Iniciando o Project Wizard

Após selecionar a opção destacada na Figura 1, inicia-se o processo de criação de projeto, que irá guiar-lo através dos passos para a criação de um projeto inicial. Na metodologia descrita a seguir, será realizada a criação e configuração do projeto, e arquivos fontes serão adicionados em momento posterior a sua criação.

Na Figura 2 é apresentada a tela de saudações para a criação de projeto, e al clicar em <avançar> será encaminhado para a tela de seleção do microcontrolador, conforme indicado pela Figura 3. Abrindo aparecerá todos os microcontroladores compatíveis com o MPLAB IDE, e deve-se verificar o microcontrolador PC18F4550 seja o escolhido (caso utilize outro microcontrolador ele deve ser escolhido de forma adequada)

Figura 2 - Tela Inicial

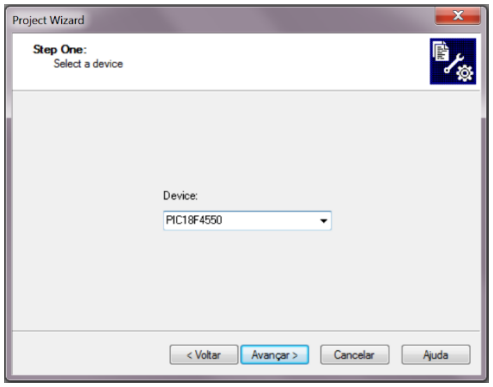

Figura 3- Escolha do microcontrolador

Um passo importante é a escolha da ferramenta responsável pela compilação do código. A Figura 4 apresenta a janela de seleção dos programas que serão utilizados na compilação do código. Observe que a barra "Active Toolsute" está selecionado a ferramenta *Microchip C18 Toolsuit*, que é a ferramenta responsável pela compilação do código em C.

! Caso tal ferramenta não seja selecionada, os arquivos fontes criados utilizarão a linguagem Assembler que é a linguagem padrão do MPLAB IDE.

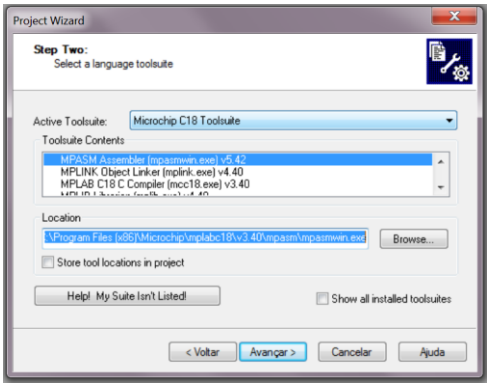

Figura 4 - Escolher a linguagem de programação

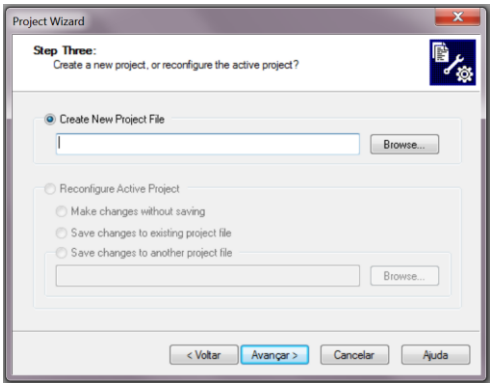

Figura 5- Selecionar a pasta do projeto

Na Figura 5 mostra a janela de seleção da pasta onde o projeto será salvo. Um projeto não faz diferença de onde os arquivos fontes do trabalho estejam localizados, uma vez que ele irá salvar o caminho relativo de cada um deles. Entretanto é importante e aconselhável que todos os arquivos pertencentes a um mesmo projeto esteja na mesma pasta. Certifique-se sempre que os arquivos fontes estejam nessa pasta de projeto.

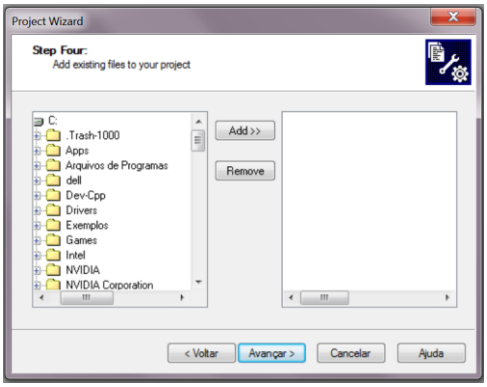

Figura 6 - Associar o programa fonte ao projeto

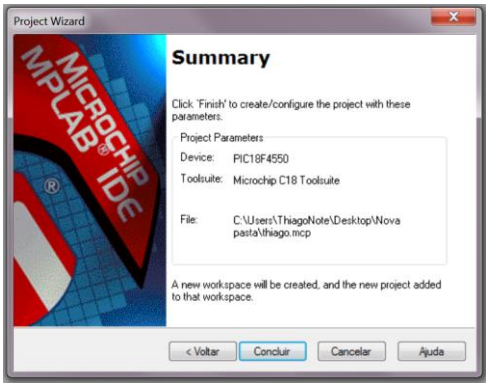

Figura 7 - Sumário de criação do projeto

A Figura 6 apresenta a janela que permite associar o arquivo-fonte ao projeto. Para fazer isso basta selecionar o esse arquivo na janela da direita, clicar no botão Add. Para maior comodidade iremos associar os arquivos fontes ao final do processo de criação, logo pode-se clicar em Avançar.

Com o programa configurado, o MCLAB IDE irá apresentar a janela de resumo com as informações relativas ao projeto, conforme pode ser visualizada na Figura 7. Clicando em Concluir o projeto será criado, e o MPLAB o levará para o ambiente de desenvolvimento.

- Ambiente de Desenvolvimento

O ambiente de desenvolvimento será o ambiente de trabalho no qual toda a programação será realizada. Na Figura 8 está apresentada essa janela de projetos. A janela que aparece a esquerda é chamada de com o título "*Projeto1.mcw*" é chamada de Área de Trabalho do Projeto. Nela estarão relacionados TODOS os arquivos que pertencem a esse projeto. Caso essa janela não aparece, clique no menu superior em **View\Project**

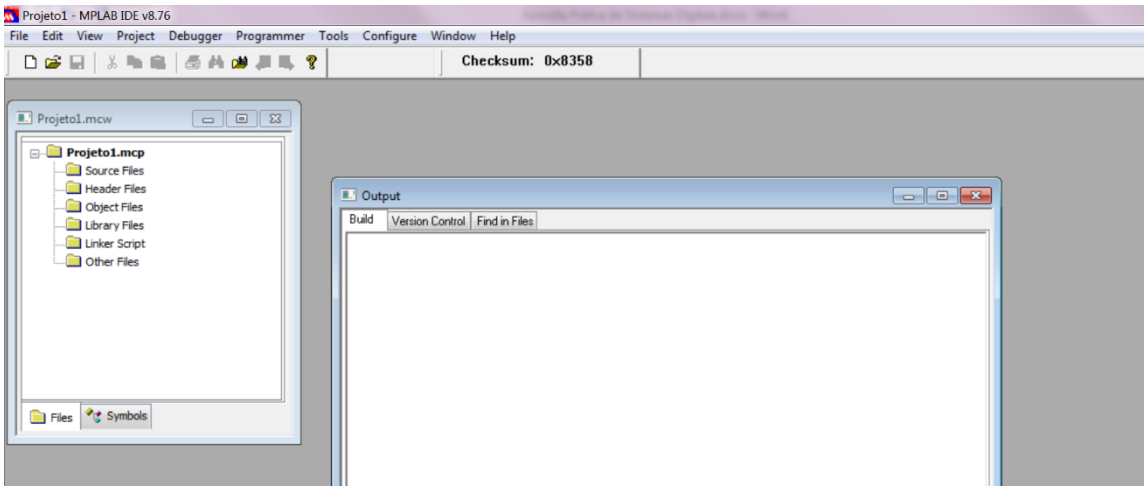

Figura 8 - Tela Principal do MPLAB IDE

Nessa janela encontram-se também pastas para a organização dos programas conforme o seu tipo. A pasta "*Source Files*" é onde ficará organizado todos os arquivos fontes, com extensão ".C" (ou .ASM caso o programa esteja codificado em Assembler). Para adicionar um arquivo ao projeto deve-se clicar com o botão direito do mouse sobre a pasta "*Source Files*", clicar e "*Add Files*" e apontar para o arquivo.

! As pastas existentes na Área de Trabalho do Projeto representam apenas ligações entre os arquivos nesse mesmo projeto. Esse link irá apontar para o arquivo na pasta que ele estiver, reforçando a necessidade de que, para melhorar a organização do projeto deve-se manter todos os arquivos na mesma pasta.

- Configurando um Gravador para esse projeto

Existem diversos modelos de gravadores que responsáveis por transferir um programa compilado para o microcontrolador. Além disso, o microcontrolador PIC18F4550 tem ainda a vantagem de realizar a gravação via bootloader, sem a necessidade de cortar a alimentação ou desligar o chip.

Um gravador que será utilizado é o MPLAB ICD2, construído pela Empresa Mosaico conforme pode ser visto na Figura 9. Ele possui conexão USB para a ligação com o computador, e possui conexão RJ11 para ligação da placa de desenvolvimento e o gravador. Ela também pode ser usada para realizar o DEBUG do programa gravado, passando a controlar o microcontrolador e executar passo a passo as instruções programadas. Além disso, ela possui extensão para realizar a gravação de qualquer chip compatível PIC (ver especificações).

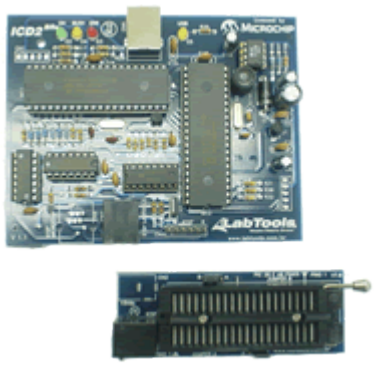

Figura 9 - Gravador MPLAB ICD2

Quando conectada a placa de desenvolvimento, o microcontrolador interrompe sua tarefa e entra em modo de programação. Ao desconectar o cabo RJ11 dessa placa, o microcontrolador se reinicializa e entra em modo de execução.

! A conexão entre o gravador e a placa de desenvolvimento é realizada através do cabo RJ11. Não confundir e tentar conectá-lo pela porta USB.

É necessário informar ao MPLAB IDE qual será a placa de gravação utilizada, para isso deve-se clicar no item da barra de menus **Programmer\Select Programmer** e escolher a opção MPLAB ICD2 conforme mostrado na Figura 10.

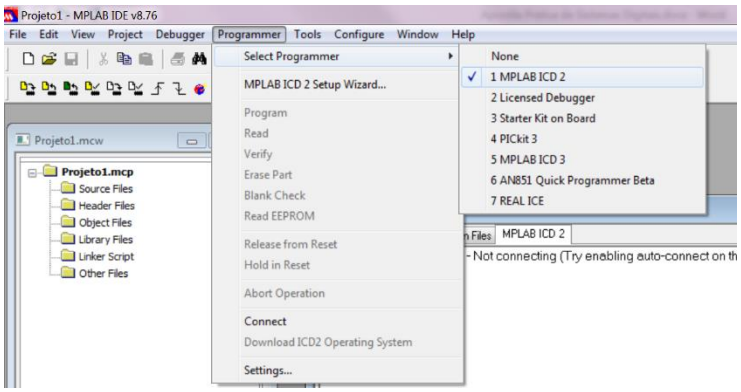

Figura 10 - Configurando o gravador

Para esse gravador, existe a opção “MPLAB ICD 2 Setup Wizard...”, uma ferramenta de configuração automática. Ao clicar nesse comando a janela de inicialização apresentada na Figura 11 será apresentada. Clique em avançar.

Figura 11 - Tela Inicial gravador

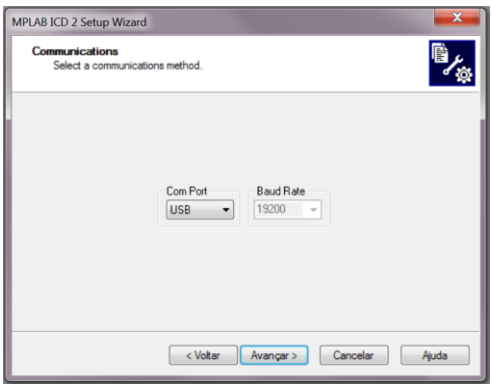

Figura 12 - Método de Comunicação

Na Figura 12 é mostrado a janela de configuração do método de comunicação. Basta selecionar o cabo USB para a comunicação (se ele não for o único) e clicar em avançar.

A janela apresentada na Figura 13 é uma parte crucial para a programação, que pode queimar o gravador caso ela seja configurada de forma incorreta. *Target has own power supply.*

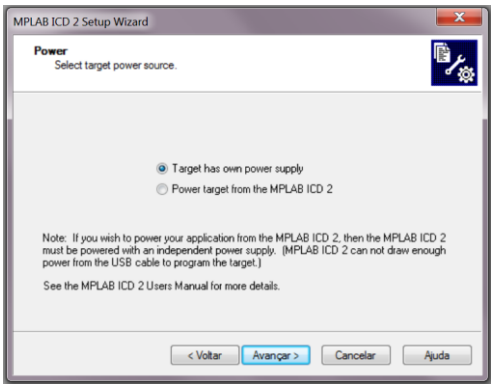

Figura 13 - Fonte de energia do gravador

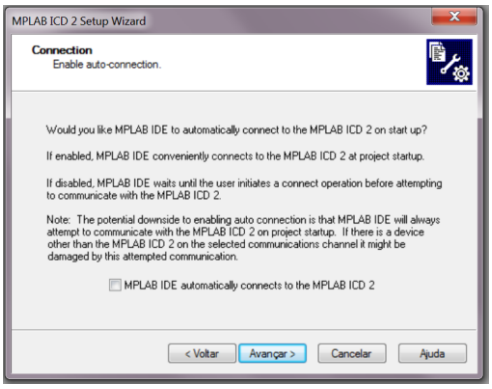

Figura 14 - Autoconexão com o gravador

! O gravador possui sua própria alimentação, então essa opção deve ser marcada na Figura 13. A outra opção pode acarretar em curto circuito no gravador e por consequência queimá-lo.

A autoconexão mostrada na Figura 14 pode ser deixada sem marcar. Dessa forma o gravador somente irá testar a conectividade no momento de realizar a gravação do código.

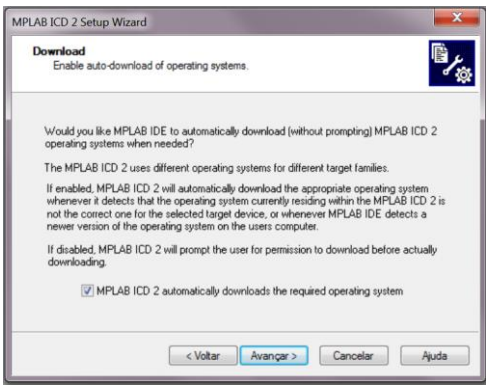

Figura 15 - Download de drivers automático

Figura 16 - Sumário

A Figura 15 trata de configurar o MPLAB IDE para realizar o download do driver caso o mesmo não se encontre no sistema operacional, logo pode deixar marcado o checkbox. Por sim Figura 16 irá mostrar o resumo das configurações do gravador.

Ao final desse procedimento, o gravador estará CONFIGURADO, e poderá ser conectado ao computador. Para fazer isso clica-se no menu **Programmer\Connect.** Uma mensagem de conexão será apresentada informando se houve algum erro, ou se o gravador está conectado conforme apresentado na Figura 17.

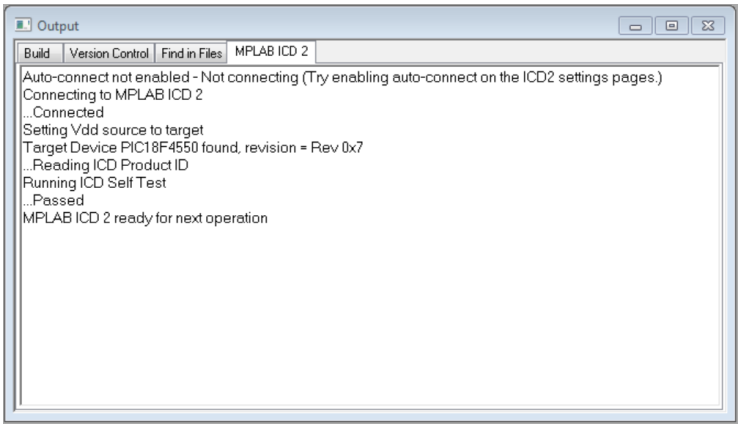

Figura 17 - Comunicação com gravador

5) Conclusão e Discussões

- O que é um arquivo fonte?
- O que significa compilar um arquivo fonte?
- Maneiras de controlar a versão que se está desenvolvendo um projeto.
- A importância de um arquivo principal para um projeto.
- Como criar um novo projeto baseado em um template?
- Instalação do MPLAB IDE e do MPLAB C18.
- O que é indentação de um código fonte, e qual a sua importância

6) Bibliografia

Zanco, W. S. (2013). *Microcontroladores PIC18 com Linguagem C uma abordagem prática e objetiva.* São Paulo: Érica.

Prática 2 – Piscando um LED (Programa Blink)

1) Introdução

O programa “Hello World!”, é um famoso programa de computador que imprime essa mensagem na tela. Como é tipicamente um dos programas mais simples a serem implementados em uma linguagem de programação, tornou-se uma tradição seu uso para mostrar a iniciantes o mais básico de uma sintaxe de linguagem de programação.

Quando programando microcontroladores, ou componentes software embarcado, o programa “Hello Word!” se modifica e se transforma no programa Blink!. Esse programa tem por objetivo acender um led ligado a uma porta de saída desse microcontrolador (e um resistor em série ligado ao terra), e mantê-lo piscando intermitentemente.

Independente do sistema que se esteja trabalhando, programas simples tem a função de testar o hardware, as ligações elétricas, e apresentar um primeiro contato. Usuários mais experientes podem querer pular esse passo, mas seja por saudosismo, ou qualquer outro motivo, pode-se afirmar sem sombra de dúvida que grandes microcontroladores, multi-núcleos, que funcionam a giga-hertz, tiveram seus primeiros testes acendendo e apagando um led.

2) Objetivos

- Criação de projetos utilizando o software MPLAB.
- Entender a criação de projetos utilizando o Project Wizard.
- Entender como realizar a gravação de um programa no PIC utilizando o ICD2
- Familiarizar com os componentes da placa de desenvolvimento.

3) Desenvolvimento

- Escrever o diagrama de blocos para um programa que pisque um LED.
- De posse do diagrama de blocos, escreva as rotinas em C e grave-as no PIC.
- Utilize a saída da placa para ligar um LED externo ao circuito. (atenção para o circuito na prot-o-board)

4) Conclusão e Discussões

- Explique o funcionamento dos registradores utilizados, e as funções compartilhadas nesses pinos.
- Quais recursos da placa foram utilizados. Quais foram as configurações necessárias de hardware.
- Qual a função do bloco de repetição

```
while(1)
{
     ....... código
}
```

- Explique o que é identação e qual a sua função em um programa em C?
- Qual a corrente de saída na porta do LED? O PIC suporta essa corrente?

Prática 3 – Botão acionando um LED

1) Introdução

O dispositivo que faz a interligação do microcontrolador com o mundo é chamado de unidade de I/O. Uma porta de I/O é controlada por dois registradores. O registrador o TRISx responsável pelo "SENTIDO" da porta (entrada ou saída) enquanto que o registrador PORTx é reponsável pelo "ESTADO" lógico dos pinos ("1" para alto e "0" para baixo). Uma porta possui 8 bits e a cada um é dado o nome de "Rxy" (exemplo? RB0, RA1, RE7). Cada pino pode ser configurado individualmente como entrada ou saída.

Para alterar o "SENTIDO" do pino atribui-se ao vetor TRISx, valores binários onde "1" será entrada (1nput) e "0" será saída (0utput). A posição do bit dentro do registrador está relacionado com o cada pino.

EX: TRISB =

RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
1	1	1	1	0	0	0	0

Independente da função assumida (I/O), o valor do registrador PORTx está relacionado com cada pino da associado a aquela porta:

Saída: o pino irá assumir uma tensão relacionada ao nível lógico que consta no registrador PORTx.

Entrada: o microcontrolador irá identificar a tensão no pino (normalmente níveis TTL) e atualizar o registrador PORTx com o nível lógico.

EX: PORTB =

RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
1	0	1	1	0	0	0	0

Para setar ou tomar o valor de apenas um bit de uma porta, pode-se usar os comandos:

TRISxbits.Rxy

Exemplo: TRISBbits.RB0 = 1 (configura o pino RB0 como entrada)

PORTxbits.Rxy

Exemplo: PORTDbits.RD5 = 1 (atribui ao pino RD5 o nível lógico alto)

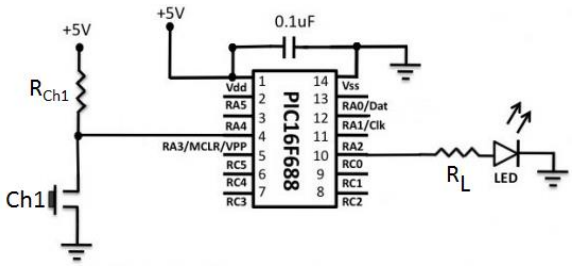

Figura 18 - Microcontrolador ligado a um LED e uma chave

Para as ligações elétricas no microcontrolador, pode-se observar um exemplo pela Figura 18 como será disponibilizado a entrada (chave no pino RA3) e a saída (LED no pino RC0). Os registradores TRISA e TRISC devem estar configurador de forma apropriada.

2) Objetivos

- Identificar registradores que controlam o dispositivo de I/O de um microcontrolador.
- Verificar a diferença entre entrada e saída de dados de uma porta e a sua ação sobre o registrador PORT
- Implementar um botão simples que ofereça entradas elétricas compatíveis com TTL ao microcontrolador.
- Controlar o estado lógico de uma porta através de um sinal de entrada.

3) Desenvolvimento

a) Escrever o diagrama de blocos para um programa que controle o estado de um LED ao pressionar um botão.
b) De posse do diagrama de blocos, escreva as rotinas em C e grave-as no PIC.
c) Utilize a saída da placa para ligar um LED externo ao circuito. (atenção para o circuito na prot-o-board) e controlar seu estado com o botão.

4) Conclusão e Discussões

- Qual será o valor da posição RB0 de um registrador PORTB setado todo como entrada e liga
- Como construir um circuito com botão de modo a oferecer uma tensão de entrada TTL a uma porta lógica? Qual a consequência ao deixar o pino de entrada sem ligação elétrica (“flutuando”).
- O que irá acontecer se somente o pino RB4 estiver setado como entrada (os outros saídas), ligado ao GND, e atribuir PORTB = 255?
- Como escolher o resistor para limitar a corrente de acionamento do LED?

Prática 4 – Matriz de botões

1) Introdução

Quano e necessário adquirir dados vindos de diversos botões (por exemplo um teclado de computador), uma forma de se tratar é através de um processo de varredura matricial de botões. Pode-se dispor os botões na forma de matriz, e através da varredura das linhas e colunas, construir a leitura de todo o teclado.

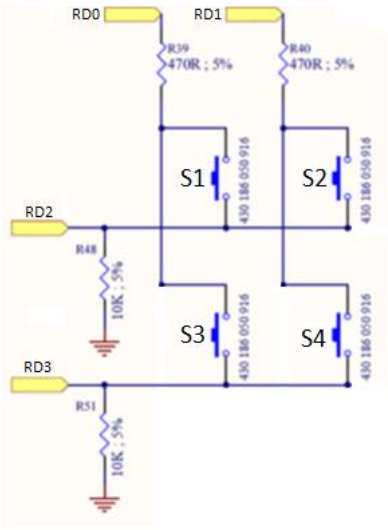

Figura 19 - Modelo de um teclado matrixial 2x2

O princípio de funcionamento do teclado matricial, pode ser exemplificado na Figura 19, onde é apresentado um teclado 2x2:

1. Serão necessárias 4 portas de I/O para a leitura de 4 chaves.
2. A varredura consiste no procedimento de leitura das teclas, horizontal ou verticalmente distribuídas.
3. Para o caso de uma varredura horizontal, segue-se o seguinte procedimento
   a. Portas RD0 e RD1 definidas como SAÍDA e Portas RD2 e RD3 definidas como ENTRADA (TRISD = 0b00001100 – bits 0 e 1 setados como Entrada “1”, e bits 2 e 3 setados como Saída “0”)
   b. Liga-se a primeira coluna (a RD0 é atribuído o valor lógico “1” e a porta RD1 atribui-se o valor “0”);
   c. Realizando a leitura das portas RD2 e RD3, pode-se verificar a condição das chaves S1 e S3 (se estão apertadas ou soltas, caso os valores em cada porta sejam “1” ou “0”)

d. Desliga-se a primeira coluna, e liga-se agora a segunda coluna (a RD0 agora será atribuído o valor lógico “0” e a RD1 o valor “1”)
e. Realizando a leitura nas mesmas portas RD2 e RD3 pode-se perceber que agora a leitura representará a condição das chaves S2 e S4.

2) Objetivos

- Entender o funcionamento de um teclado matricial, considerando 16 teclas.
- Realizar a leitura das teclas de um teclado matricial.
- Acionar os leds relacionando-os com as chaves pressionadas.
- Controlar o estado lógico de uma porta através de um sinal de entrada.

3) Desenvolvimento

a) Escrever o diagrama de blocos para um programa que faça a leitura dos valores de um teclado matricial.
b) De posse do diagrama de blocos, escreva as rotinas em C e grave-as no PIC.
c) Utilize a PORTB como saída dos LEDS e apresente o valor da tecla pressionada.
d) Implemente o mesmo exemplo utilizando o PIC Simulator ou o PICSimLab.

4) Conclusão e Discussões

- Explique o funcionamento do teclado matricial 4x4.
- Quais as diferenças entre o sistema simulado e o sistema físico?
- Qual a frequência de amostragem ideal para a varredura do teclado?
- Descreva como ficaria um teclado matricial para realizar a leitura de 64 teclas, quais portas seriam usadas, e qual a configuração adotada. Admita que esse tecla está disposta de forma quadrada.

Prática 5 – Display de 7 Segmentos

1) Introdução

O display de 7 segmentos é um dispositivo formado por vários LEDs, ou LCDs que permite visualizar números e alguns caracteres. Utilizado em relógios, balanças eletrônicas, medidores de temperatura, etc. Na Figura 20 está apresentado a disposição dos segmentos, e as ligações elétricas de cada um.

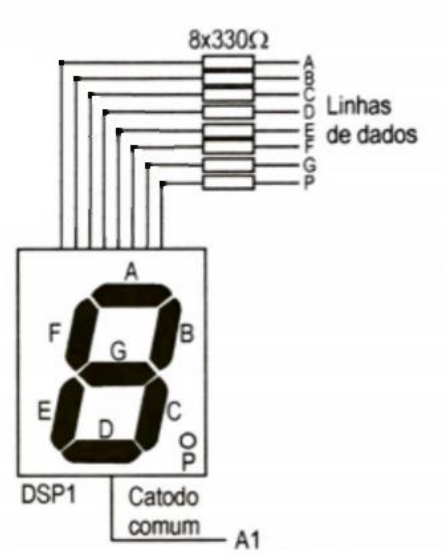

Figura 20 - Modelo de display de 7 semento Catodo Comum

Existem dois tipos de display: catodo comum e anôdo comum. Nessa prática será levando em conta o modelo catodo comum, onde os catodos do display estarão ligados ao terra, e um sinal “alto” nas entradas irá acender o display correspondente.

É importante se atentar para a composição dos elementos. Se forem LEDs pode ser necessário limitar a corrente utilizando um resistor.

Normalmente liga-se o display a uma porta lógica do microcontrolador, de modo que:

DIS-PLAY	P	G	F	E	D	C	B	A
PORTD	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0

Para visualizar corretamente os valores no display, uma determinada sequencia deve ser enviada a porta D. Pode-se criar um vetor de dados de forma a acionar diretamente a codificação de cada elemento:

```
const char tabela[] = { 0x3F // número 0
                        0x06 // número 1
                        0x5B // número 2
                        .....}
```

2) Objetivos

- Identificar as pinagens para a placa utilizada, e realizar a configuração para o acionamento de 1 display.
- Entender o funcionamento de um display de 7 segmentos.
- Realizar a leitura das teclas de um teclado matricial e acionar o display um display de 7 segmentos (use códigos hexadecimais para as teclas).
- Fazer a comutação entre o display e o teclado.

3) Desenvolvimento

a) Escrever um diagrama de blocos que faça a escrita de um digito no display de unidades.
b) Faça a programação em C do diagrama desenvolvido
c) Escrever agora um diagrama de blocos para um programa que faça a leitura dos valores de um teclado matricial e o acionamento do display.
d) De posse do diagrama de blocos, escreva as rotinas em C e grave-as no PIC.
e) Utilize a PORTB como saída dos LEDS e apresente o valor da tecla pressionada.
f) Implemente o mesmo exemplo utilizando o PIC Simulator ou o PICSimLab.

4) Conclusão e Discussões

- Explique o sistema de acionamento de cada display na placa, quais portas utilizar.
- Explique o processo de comutação entre teclado e display.
- Qual a frequência de amostragem do display de 7 segmentos?

Prática 6 – Contador com LED e Display

1) Introdução

Quando é necessário fazer a interface de mais de um display em um microcontrolador pode-se utilizar a técnia de multiplexação de displays. As portas de de todos os displays são ligados a um barramento e é realizado o controle de qual display será ligado através do acionamento de seu catodo.

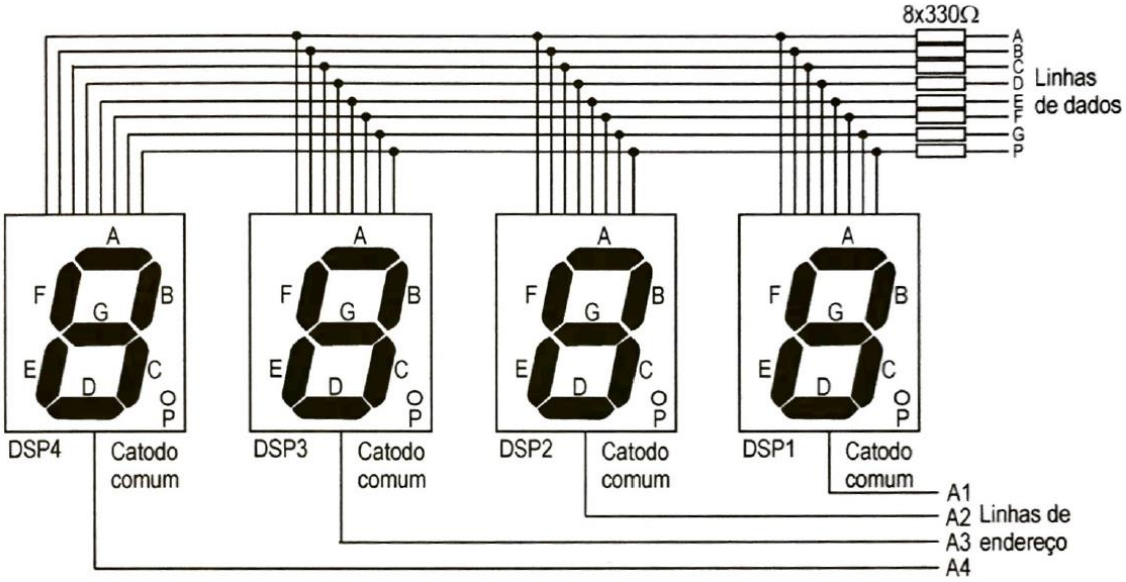

Figura 21 – Multiplexação de displays de 7 sementos - Catodo Comum

Na Figura 21 está apresentado um conjunto com 4 displays multiplexados. Estão ligados todos a um único barramento (linha de dados) e os catodos podem ser conectados individualmente ao terra através das linhas de endereço.

Essa multiplexação irá otimizar o hardware uma vez que diminui a quantidade de pinos dedicados. Cada display irá emitir um número de cada vez, e enxergamos todos acessos quando a frequência que cada display acende e apaga fica aproximadamente acima dos 30Hz. Frequências maiores irão melhorar também o resultado até um ponto.

Considerando que cada display fica acesso durante 4ms, teríamos um ciclo total de 16ms para acender e apagar os 4 digitos. Colocando isso em um diagrama temporal, temos o valor dos sinais de controle para cada display. Na Figura 22 está apresentado esse diagrama considerando alto quando o catodo está ligado ao nível lógico alto, e baixo quando ele está ligado ao terra.

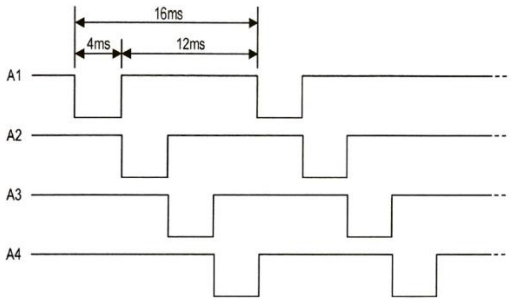

Figura 22 -Diagrama de tempo das linhas de endereço do display multiplexado

2) Objetivos

- Identificar as pinagens para a placa utilizada, e realizar a configuração para o acionar um 4 displays multiplexados
- Montar um contador.

3) Desenvolvimento

a) Escrever o diagrama de blocos para um programa que faça a varredura dos displays.
b) De posse do diagrama de blocos, escreva as rotinas em C e grave-as no PIC.
c) Montar um contador de 0 a 10.
d) Montar um contador de 0 a 9999.

4) Conclusão e Discussões

- Explique o sistema de acionamento de cada display na placa, quais portas utilizar.
- Explique o processo de multiplexação e a importância do tempo da frequencia de amostragem do display
- Complete a tabela abaixo

Númedo de Displays	Período	Tempo que cada display se mantem acesso	Frequência
2			
3			
4	16ms	4ms	62,5 Hz
5			
6			
7			
8			

Prática 7 – Conversor A/D

1) Introdução

O conversor analógico-digital é um dispositivo eletrônico capaz de gerar uma representação digital a partir de uma grandeza analógica. Utiliza-se normalmente um sinal representado por um nível de tensão ou intensidade de corrente elétrica.

Um conversor A/D de 10 bits, convertendo um sinal de entrada analógica com tensão variável entre 0V e 5V pode assumir os valores binários de 0 (0000000000) a 1023 (1111111111), capturaando 1024 níveis discretos de um determinado sinal. Se o sinal de entrada do suposto conversor A/D estiver em 2,5V, por exemplo, o valor binário gerado será 512.

O conversor A/D do PIC18F4550 é do tipo aproximação sucessiva, e sua entrada é multiplexada em 13 pinos. Dessa forma a configuração de quais portas utilizar deve ser realizada via registradores específicos. Na Figura 23 é apresentado o esquemático desse conversor com os flags de registradores a serem utilizados.

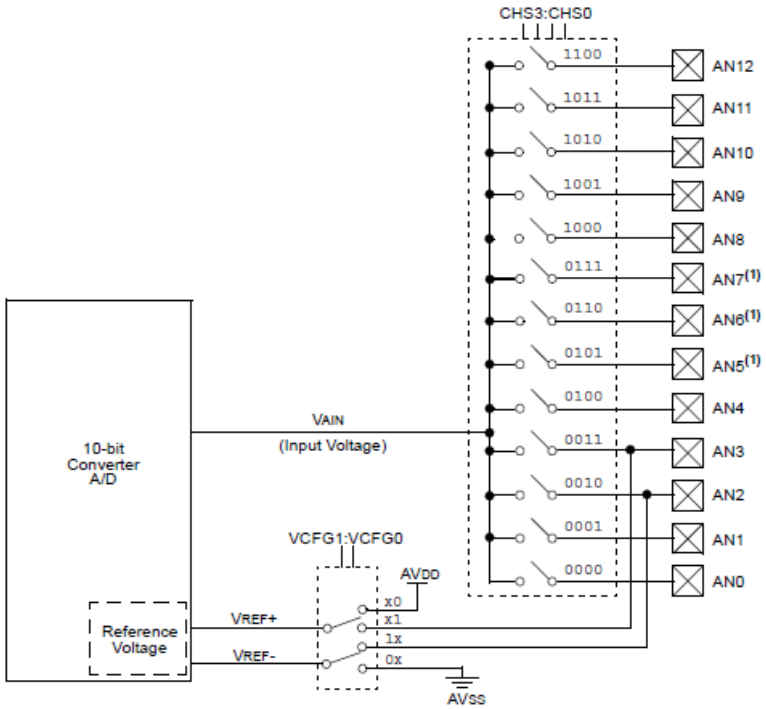

Figura 23 – Diagrama de blocos simplificado do conversor A/D

A operação do módulo ADC é bastante simples e intuitiva:

(1) Selecione o clock a ser utilizado no conversor (ADCON2).

(2) Configure a entrada do conversor (ADCON1).

(3) Selecione a tensão de referencia que será utilizada pelo conversor.

(4) Configure o tempo de aquisição desejado.

(5) Selecione o canal de entrada a ser convertido.

(6) A conversão inicia quando o bit $GO/\overline{DONE}$ do registrador ADCON0 é setado para 1. (ele vai pra 0 quando terminar a conversão)

2) Objetivos

- Identificar as pinagens para a placa utilizada, o potenciômetro de referência, os pinos para a entrada de sinais externos.
- Entender a criação de funções no C.
- Utilizar o conversor A/D para realizar medições de tensão analógica.

3) Desenvolvimento

a) Escrever um diagrama de blocos que faça a leitura do conversor A/D.
b) Criar funções para a leitura de tensão analógica.
c) Apresentar nos LEDS ligados a PORTB os valores relacionados com os 8 digitos mais significativos da conversão A/D
d) Adaptar a leitura de um segundo canal de conversão.

4) Conclusão e Discussões

- Explique o métod de conversão utilizado pelo PIC18F4550 (aproximação sucessiva).
- Apresente um modelo esquemático da placa para a leitura de 4 portas analógicas, de forma sequencial. Escreva o diagrama de blocos do programa.

Prática 8 – Display de Cristal Líquido (LCD)

1) Introdução

Um display de cristal líquido, ou LCD (*liquid crystal display*), é um painel fino usado para exibir informações por via eletrônica, como texto, imagens e vídeos. Seu uso inclui monitores para computadores, televisores, painéis de instrumentos. Cada elemento de um LCD tipicamente consiste de uma camada de moléculas alinhadas entre dois eletrodos transparentes e dois filtros polarizadores. Na presença de um campo eletromagnético essas moléculas adquirem um padrão estrutural que impedem a passagem da luz visível. Os componentes então são organizados de forma a construir os caracteres e os pixels.

Existem uma grande quantidade de LCDs no mercado, e será estudado o modelo com 2 linhas e 16 caracteres. Ele já possui um driver de controle interno, logo os textos ou comandos podem ser passados diretamente utilizando uma comunicação paralela.

O conversor A/ d do PIC18F4550 é do tipo aproximação sucessiva, e sua entrada é multiplexada em 13 pinos. Dessa forma a configuração de quais portas utilizar deve ser realizada via registradores específicos. Na Figura 23 é apresentado o esquemático desse conversor com os flags de registradores a serem utilizados.

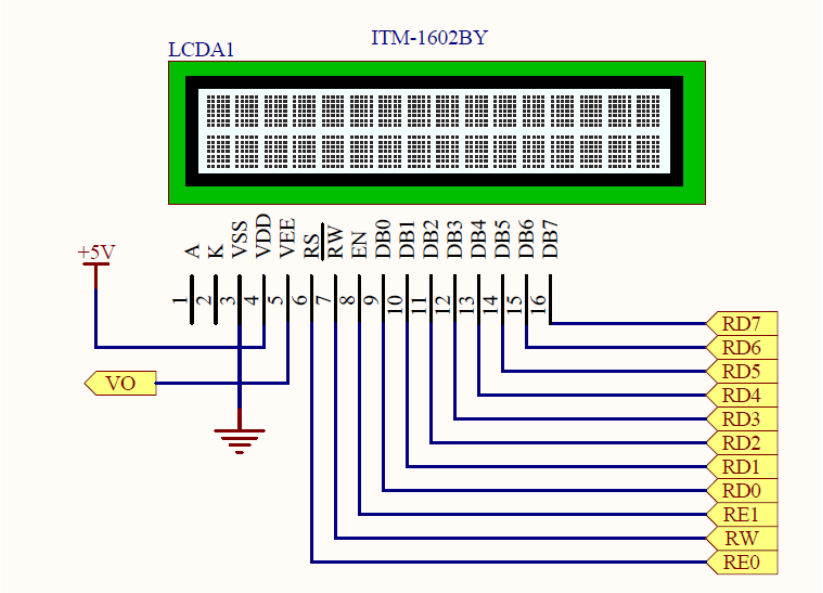

Figura 24 - Display LCD

Na Figura 24 está apresentado a ligação eletrônica entre o LCD e a placa MCLab3. Pode-se perceber que é utilizada a porta D para realizar a leitura ou escrita, e nesse projeto utilizaremos uma biblioteca que leva em consideração essa configuração.

Existem dois titpos de informação que podem ser enviadas a um display: Dados ou Comandos. Para cada uma delas será utilizada a função específica da biblioteca.

- **Dados**: Utiliza-se o padrão da tabela ASCII para escolher um caracter. Na linguagem C a forma de passar um código em ASCII de um caracter é utilizar aspas 'A'

Exemplo: `>> escreve_lcd('S');` *Escreve e avança um espaço*

`>> escreve_lcd('D');`

`>> escreve_lcd('M');` *Ao final estará apresentado SDM no display*

- **Comandos**: Existem uma série de comandos a serem utilizados no LCD. Os mais comuns são descritos a seguir:

`Limpar display`	`0x01`
`Ligar Cursor`	`0x0E`
`Desligar Cursor`	`0x0C`
`Cursor Piscante`	`0x0D`
`Cursor com Alternância`	`0x0F`
`Posicionar cursor`	Ver tabela

coluna	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
linha 0	0x80	0x81	0x82	0x83	0x84	0x85	0x86	0x87	0x88	0x89	0x8A	0x8B	0x8C	0x8D	0x8E	0x8F
linha 1	0xC0	0xC1	0xC2	0xC3	0xC4	0xC5	0xC6	0xC7	0xC8	0xC9	0xCA	0xCB	0xCC	0xCD	0xCE	0xCF

Exemplo: `>> comando_lcd(0x0D);` *Desligar o cursor*

`>> comando_LCD(0xC0);` *Posicionar no inicio da segunda linha*

Para ter acesso a essa biblioteca deve-se adcionar dois arquivos a pasta que contem o projeto e adcionar no gerenciador de projeto:

- LCD_Lib.h – arquivo de cabeçalho, contem o link com as funções.
- LCD_Lib.o – arquivo objeto, contem o programa em compilado.

Além desses comandos é necessário realizar uma rotina de inicialização no LCD. Essa rotina deve ser utilizada para configurar o número de linhas do display, o tipo de interface, e posicionar o cursor no início do bloco. Para isso pode-se utilizar uma vez a função: `>> inicializa_lcd();`

2) Objetivos

- Identificar as pinagens para a placa utilizada, o tipo de LCD, a forma de comunicação.
- Escrever uma mensagem padronizada no LCD.
- Incrementar um contador a cada ciclo e mostrar seu valor no LCD.

3) Desenvolvimento

a) Escrever um diagrama de blocos explicando o controle de fluxo no
b) Utilizando as funções do LCD escrever uma mensagem padronizada no display de LCD.
c) Incrementar uma variável e apresentar seu valor no LCD. (conferir a tabela ASCII para enviar facilmente um número ao LCD)
d) Colocar uma mensagem “CONTAGEM” na primeira linha e o valor da contagem sendo incrementada na segunda linha.

4) Conclusão e Discussões

- Identifique outros displays de LCD, de diferentes quantidade de linhas e colunas.
- Como funcionaria o display de LCD no caso de televisores que possuem a imagem composta por 3 cores básicas?

ASCII Table

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(	72	48	110	H	104	68	150	h
9	9	11		41	29	51	)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[	123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	|
29	1D	35		61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

Prática 9 – Interrupção / Módulo de Temporização

1) Introdução

Interrupções são definidas como eventos assíncronos, gerados por unidades de *hardware* internas ou externas ao sistema microprocessado. Este evento provoca um desvio na sequência normal de execução do programa, levando a CPU a executar as instruções de uma localidade específica da memória de programa (vetor de interrupção), que inicia uma rotina responsável por tratar o evento ocorrido. Após a execução desta **rotina de tratamento da interrupção (*interrupt service routine*, ISR)**, o programa principal volta a ser executado, sendo retomado do ponto onde havia sido interrompido. Variáveis que são manipuladas dentro do programa principal e dentro da ISR devem ser definidas como "voláteis". Ex.:

```
volatile unsigned char x = 0;
```

O sistema de interrupção do PIC18F4550 possui múltiplas fontes de interrupção independentes, provenientes de seus periféricos, tais como, os módulos de temporização (Timer0 a Timer3), o conversor A/D, o módulo PWM, o módulo de comunicação serial (USART), entre outros. Há dois níveis de prioridade de ISR: alta (*high priority*) baixa (*low priority*). Dez registradores estão relacionados ao sistema de interrupção:

RCON, INTCON, INTCON2, INTCON3, PIR1, PIR2, PIE1, PIE2, IPR1, IPR2

Basicamente, para habilitar uma interrupção de um periférico, deve-se:

i) Ligar a chave geral de interrupções GIE (INTCON<7>)
ii) Ligar a chave de interrupção de periféricos PEIE (INTCON<6>)
iii) Ativar a interrupção desejada
iv) Definir a prioridade da interrupção
v) Efetuar configurações específicas do periférico

Ao ocorrer uma interrupção, um bit de sinalização (***flag***), correspondente ao periférico que provocou o evento, é **setado**. Para que novas interrupções deste periférico sejam habilitadas, este *flag* deve ser limpo na ISR.

Nesta aula prática, será utilizado como exemplo do uso de interrupção em um programa a aplicação do módulo Timer0 do PIC18F4550 operando como temporizador. O módulo Timer0 é um contador/temporizador que pode operar com 8 bits ou 16 bits, mapeado na memória de dados através dos registradores TMR0H e TMR0L, que permite

leitura ou escrita de seu valor. O incremento do Timer0 é realizado a cada ciclo de seu sinal de *clock*, que pode ser proveniente do *clock* do sistema (Timer0 operando como temporizador, com incremento a cada ciclo de instrução) ou de uma fonte externa (Timer0 operando como contador, com incremento a cada pulso do sinal aplicado ao pino 6 – RA4/T0CKI), podendo ainda, ser dividido por meio de um "*prescaler*" programável. Quando ocorrer o estouro (*overflow*) da contagem do Timer0 pode ser requisitada uma interrupção do sistema. Neste caso, o *flag* que indica interrupção por estouro de Timer0 é bit TMR0IF do registrador INTCON (INTCON<2>).

A configuração do Timer0, ativando interrupção por estouro de contagem, pode ser realizada com os seguintes passos:

1. Ligar a chave geral de interrupções GIE (setar INTCON<7>)
2. Ligar a chave de interrupção de periféricos PEIE (setar INTCON<6>)
3. Habilitar interrupção de estouro de Timer0: setar TMR0IE (INTCON<5>)
4. Ativar o módulo Timer0: setar TMR0ON (T0CON<7>)
5. Definir modo de operação (8 bits ou 16 bits): T08BIT (T0CON<6>)
6. Selecionar origem do clock: T0CS (T0CON<5>). No caso de clock externo (RA4/T0CKI), deve-se definir borda de ativação: T0SE (T0CON<4>)
7. Ativar (ou não) e definir o *prescaler* (T0CON<3:0>)
8. Inicializar o temporizador: Carregar TMR0L e TMR0H.

O tempo total entre cada interrupção de estouro de Timer0, para operação no modo de 8 bits, pode ser calculado por:

$$\text{Tempo total} = T_{CLK,TMR0} \times \textit{prescaler} \times (\text{número de estados da contagem}),$$

sendo que o registrador TMROL deve ser inicializado (carregado) com:

TMR0L=256-(número de estados da contagem)

2) Objetivos

- Entender como habilitar e efetuar o tratamento de uma interrupção;
- Realizar a configuração do módulo de temporização Timer0;
- Utilizar o módulo Timer0 na implementação de um cronômetro digital.

3) Desenvolvimento

a) Esboçar fluxograma de tratamento de interrupção de Timer0 e criar as funções relacionadas;

b) Elaborar programa com uso do Timer0 com incremento a cada 0,1 s. Neste caso, utilize a comutação do estado de um dos LEDs conectados à porta de I/O PORTB para verificar a operação de temporização.
c) Implementar um cronômetro digital, utilizando o Timer0 e o módulo LCD, capaz de apresentar medição de tempo no seguinte formato: “00:00.0”.

4) Conclusão e Discussões

- O que é uma interrupção? O que a diferencia de uma chamada à função? O que é uma ISR?
- Explique a diferença entre a medição de tempo em um microcontrolador baseada em rotinas de *delay* (ex.: função “Delay1KTCTx()”) e módulos de temporização.
- Defina uma possível configuração do módulo Timer0 da PIC18F4550 da placa McLAB3 para temporização em períodos de 1 ms?

Prática 10 – Comunicação Serial Assíncrona

5) Introdução

A comunicação serial assíncrona UART (do inglês *Universal Asynchronous Receiver/Transmitter*) foi desenvolvida na década de 1960 para permitir comunicação entre *mainframes* e terminais de computadores remotos, tendo-se tornado amplamente difundida e sendo utilizada até os dias atuais em diversas aplicações. A comunicação entre dois dispositivos é dita serial quando o envio da informação (códigos dos caracteres) é realizado por meio de uma única linha, onde os bits enviados são encadeados um a um. O termo comunicação assíncrona indica que não é necessário sinal adicional de sincronismo (*clock*) para transmissão ou recepção, ou seja, o próprio dado transporta a informação necessária para sincronizar transmissor e receptor. Na comunicação UART, transmissão (TX) e recepção (RX) de dados de um dispositivo são feita por meio de linhas dedicadas. No caso do PIC18F4550, tais sinais são associados aos pinos RC6 e RC7, respectivamente. A comunicação UART entre diferentes dispositivos é padronizada pelo protocolo RS-232 (EIA-RS-232C). Um sinal típico de comunicação UART RS-232 é exibido na figura abaixo.

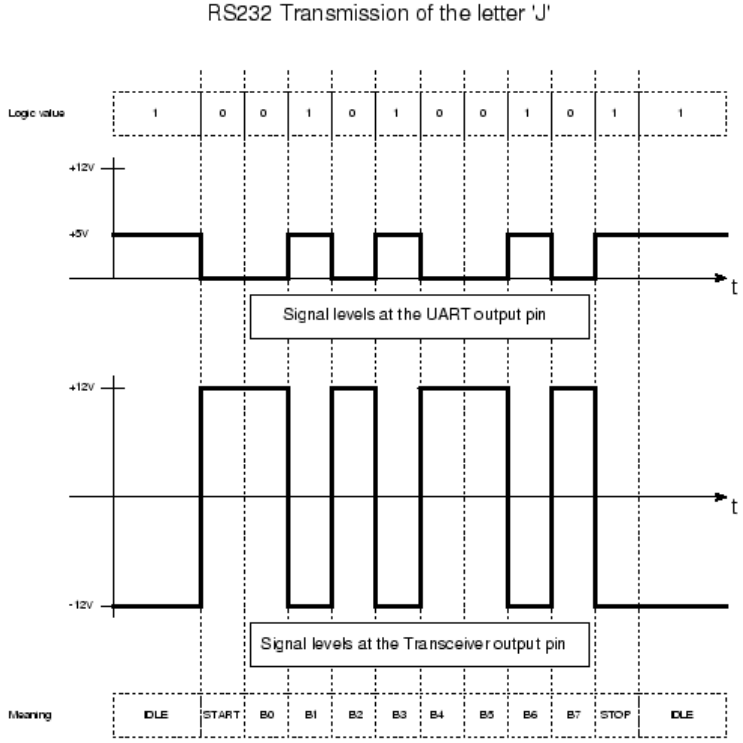

Figura 25 – Sinais típicos – Comunicação serial assíncrona (Fonte: http://www.best-microcontroller-projects.com/image-files/how-rs232-works-tx-logic-rs232-diag.png).

O PIC18F4550 possui, entre seus periféricos, um módulo de comunicação serial síncrona e assíncrona (USART, do inglês *Universal Synchronous/Asynchronous Receiver/Transmitter*). Na placa McLab 3 é possível utilizar apenas a operação no modo assíncrono (ou seja, UART). A configuração e acesso ao módulo de comunicação USART do PIC18F4550 pode ser realizada por meio do acesso direto aos registradores relacionados (TXSTA, RCSTA e BAUDCON) ou utilizando funções de periféricos disponibilizadas no compilador Microchip C18 (abordagem adotada nesta prática). As principais funções a serem utilizadas nesta prática são descritas na tabela a seguir (para utilizá-las deve-se incluir o arquivo de cabeçalho "***usart.h***" no projeto).

Tabela X – Funções de periférico (USART)

Função de periférico	Descrição
`OpenUSART()`	Configura USART
`BusyUSART()`	USART está transmitindo? (1: USART ocupado)
`WriteUSART()`	Escreve byte em USART
`ReadUSART()`	Lê byte de USART
`putrsUSART()`	Escreve string da memória de programa em USART

Alguns exemplos de uso destas funções de periféricos são exibidos abaixo:

```
while(BusyUSART());     //AGUARDA O BUFFER DE TRANSMISSÃO FICAR VAZIO

WriteUSART(0x0A);       //transmite dado de "nova linha"
WriteUSART(0x30+x);     //transmite dado do contador "x"
putrsUSART( "Hello World!" ); //transmite string da mem. de programa
```

Para configuração do módulo USART, deve-se ajustar os pinos RC6 (TX) e RC7 (RX) como saída e entrada, respectivamente, além de utilizar os parâmetros adequados na função `OpenUSART()`. Nesta prática, será utilizada a seguinte configuração:

```
TRISC = 0b10111101;         //CONFIG DIREÇÃO DOS PINOS PORTC

//CONFIGURAÇÃO DA USART 9600BPS
OpenUSART(USART_TX_INT_OFF & USART_RX_INT_OFF & USART_ASYNCH_MODE &
          USART_EIGHT_BIT  & USART_CONT_RX    & USART_BRGH_HIGH,25);
```

A descrição dos parâmetros de configuração da USART é apresentada abaixo:

Interrupt on Transmission:	
USART_TX_INT_ON	Transmit interrupt ON
USART_TX_INT_OFF	Transmit interrupt OFF
Interrupt on Receipt:	
USART_RX_INT_ON	Receive interrupt ON
USART_RX_INT_OFF	Receive interrupt OFF
USART Mode:	
USART_ASYNCH_MODE	Asynchronous Mode
USART_SYNCH_MODE	Synchronous Mode
Transmission Width:	
USART_EIGHT_BIT	8-bit transmit/receive
USART_NINE_BIT	9-bit transmit/receive
Slave/Master Select*:	
USART_SYNC_SLAVE	Synchronous Slave mode
USART_SYNC_MASTER	Synchronous Master mode
Reception mode:	
USART_SINGLE_RX	Single reception
USART_CONT_RX	Continuous reception
Baud rate:	
USART_BRGH_HIGH	High baud rate
USART_BRGH_LOW	Low baud rate

Para definir a taxa de transmissão (*baud rate*), dada em bits/s (bps), para comunicação assíncrona (modo de alta velocidade: USART_BRGH_HIGH):

$$Baud\ \ rate = \frac{f_{clock}}{16(spbrg+1)},$$

onde *spbrg* é o último dos parâmetros de entrada da função `OpenUSART()`, igual a 25 no exemplo considerado. Vale observar que o valor definido não é exato.

6) Objetivos

- Entender como configurar o módulo USART;
- Realizar comunicação serial assíncrona entre PIC18F4550 e computador;
- Combinar o uso do módulo USART com teclado matricial e LCD.

7) Desenvolvimento

a) Esboçar fluxograma de configuração e transmissão assíncrona de dados utilizando o módulo USART do PIC18F4550;
b) Elaborar programa que transmita uma informação fixa, verificando a recepção por meio de software executado em computador (HyperTerminal, Putty, ou similares);
c) Fazer a transmissão de uma contagem (0 a 9) para o microcomputador, verificando o valor transmitido por meio do LCD da placa McLab3;
d) Utilizando o programa de reconhecimento de teclas (teclado matricial), anteriormente desenvolvido, elaborar um programa que identifique a tecla pressionada do teclado matricial da placa McLab3, apresente seu valor no LCD, além de transmitir sua identificação para o microcomputador.

8) Conclusão e Discussões

- Qual a diferença entre comunicação serial assíncrona e síncrona? Pesquise as características destes dois modos do padrão RS-232.
- Pesquise sobre o CI MAX-232 (sua função e formas de utilização).