UNIVERSIDADE FEDERAL DE LAVRAS - UFLA

Laboratório de Sistemas Inteligentes e Embarcados - LABSINE

Guia de Aulas Práticas de Sistemas Embarcados

versão 1.2

Autores:
Wilian Soares Lacerda
João Paulo Fernandes de Cerqueira César

Lavras - MG
fevereiro de 2019

Ficha catalográfica elaborada pela Coordenadoria de Processos Técnicos da Biblioteca Universitária da UFLA

Lacerda, Wilian Soares.

Guia de aulas práticas de sistemas embarcados : versão 1.2 / Wilian Soares Lacerda, João Paulo Fernandes de Cerqueira César. – Lavras : UFLA, 2019.

86p. : il.

Guia elaborado para orientação da Disciplina Introdução aos Sistemas Embarcados e Microcontroladores do Departamento de Automação da Universidade Federal de Lavras.

1. Eletrônica. 2. Microcontroladores. 3. Sistemas embarcados. I. César, João Paulo Fernandes de Cerqueira. II. UniversidadeFederal de Lavras, Departamento de Automática. III. Título.

CDD-629.89

Sumário

Introdução

Este guia de aulas práticas serve às aulas da disciplina de *Introdução aos Sistemas Embarcados e Microcontroladores*, oferecida pelo Departamento de Automática da Universidade Federal de Lavras.

Este guia contém as aulas práticas a serem seguidas e cumpridas pelos alunos por semana, além de exercícios para fixação do conhecimento adquirido nas aulas. Em cada aula, é especificado o projeto de hardware a ser estudado, e o software correspondente para ser executado como exemplo. A seguir, o aluno é encorajado a desenvolver um projeto de hardware/software para praticar o que foi aprendido.

As aulas práticas são executadas utilizando o kit educacional PICGenios v3.0 fabricado pela empresa Microgenios (*http://www.microgenios.com.br*). Para o desenvolvimento do software a ser executado no kit, é utilizado a suite MikroC PRO v6.4 desenvolvida pela empresa Elektrica Inc. (*http://https://www.mikroe.com/mikroc-pic*). Este software contém, além do compilador C, várias ferramentas como um poderoso editor de texto e *help*.

Inicialmente são apresentadas experiências bem simples, como exemplo fazer um LED piscar, para que o aluno adquira a capacidade de utilizar o kit (hardware) com o software de desenvolvimento. Ao longo do curso, as experiências se tornam mais complexas a medida que outros periféricos do kit são utilizados. Ao final, o aluno aprende a lidar com o sistema de interrupção do microcontrolador.

Não é intenção deste guia substituir o livro texto que acompanha a disciplina, mas somente complementar os conhecimentos e apresentar na prática a implementação da teoria aprendida nas aulas teóricas.

Recomenda-se aos alunos adquirirem este guia de forma impressa (impressão A4, frente/verso, colorido ou preto), para que possam aproveitar os espaços em branco completando-os com suas próprias anotações.

Bom proveito!

Prof. Wilian Soares Lacerda

Normas e Conduta do Laboratório

Não é permitido fumar, comer ou beber dentro do Laboratório. Deve-se manter a bancada limpa e organizada. Durante a aula deve-se evitar falar alto. Não é permitido instalar ou desinstalar algum software, assim como mudar alguma configuração do computador. Trazer sempre um *pen-drive* para a aula prática, assim como este guia. Desligar o computador e todos equipamentos adequadamente ao final da aula.

1 Aula 01

Nessa primeira aula prática da disciplina, serão abordados os passos necessários para criação de um projeto de software no *software* “MikroC PRO”. Além disso, um exemplo básico será codificado e testado na placa física disponível no laboratório: o PICGenios v3.0 com o microcontrolador PIC 18F452.

O esquemático do circuito eletrônico que será utilizado nesta aula é mostrado na Figura 1. Nessa configuração será utilizado um cristal oscilador de 8MHz (*clock*), dois *pushbuttons* (“RB0” e “RB1”), além de outro botão *reset* do circuito. Por último, os oito pinos “RD” serão utilizados como saídas, conectados a oito LEDs.

Ao final dessa aula o microcontrolador será programado para piscar todos os oito LEDs enquanto o *pushbutton* “RB0” estiver pressionado. O código utilizado como exemplo é mostrado na seção 1.1.

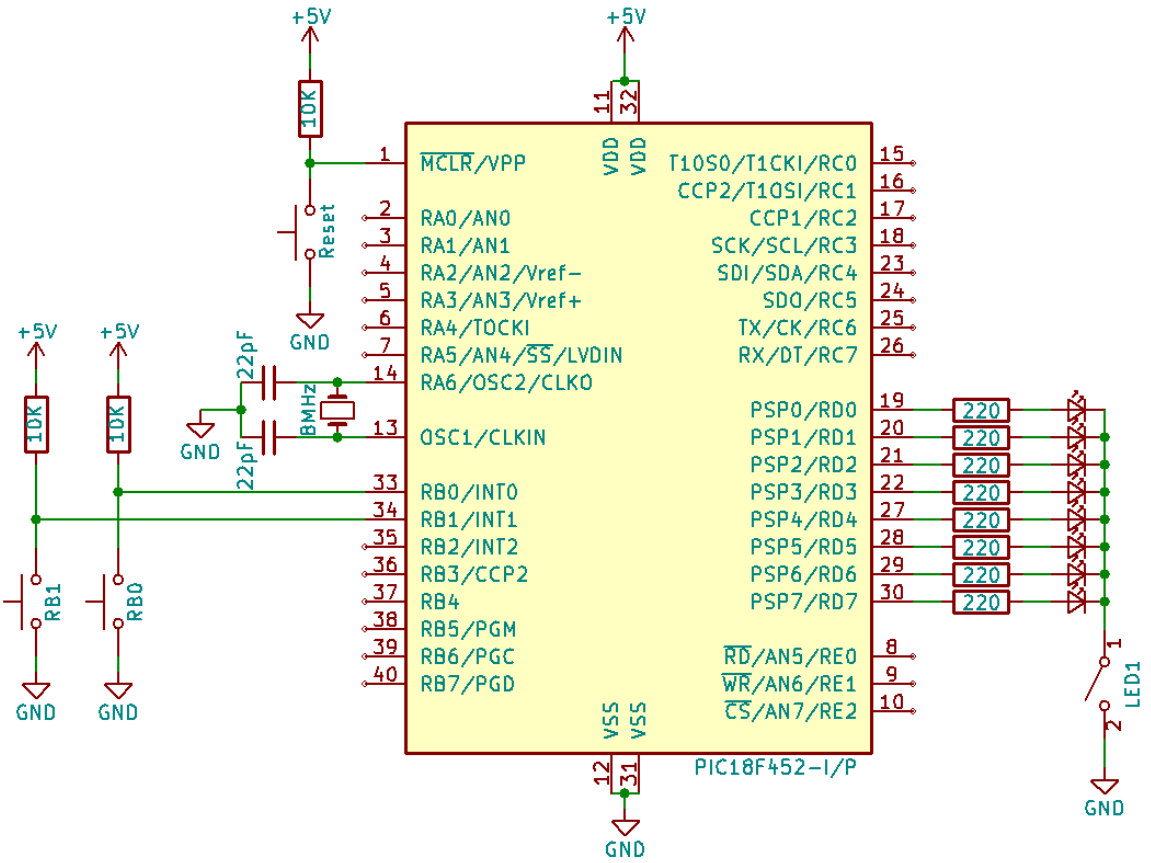

Figura 1: Esquemático do projeto.

O passo a passo de como criar um novo projeto de software, adicionar um código em linguagem C já existente a esse e compilá-lo, é mostrado na seção 1.2. A seção 1.3 mostra como ligar e conectar o kit PICGenios ao computador e, por fim, a

seção 1.4 mostra como gravar o projeto nesse kit.

1.1 Código do Projeto

```
/*****************************************************************
**                          Exemplo 01                          **
**      Exemplo para utilizacao de LEDs e pushbuttons do        **
**      kit PIC 18F452. Enquanto pushbutton RB0 estiver         **
**      presisonado, os LEDs (LED1 PORTD) piscarao alternando   **
**      o estado entre ligado e desligado.                      **
**                                                              **
** Arquivo:     piscapisca.c                                    **
** Compilador:  MikroC PRO PIC v.6.4.0                          **
** Data:        02/2019                                         **
*****************************************************************/
void main()
{
    // Ativa somente o pino RB0 como entrada.
    // os demais pinos sao configurados como saida.
    trisb = 1;  // 0b00000001 (bin) = 1 (dec)

    // Todos os pinos RD sao configurados como saida.
    trisd = 0;  // 0b0000000 (bin) = 0 (dec)

    while(1)    // Loop infinito
    {
        // Le o estado do push button RB0
        // Caso esteja pressionado (valor = 0), executa laco.
        while(portb.rb0 == 0)
        {
            // Desliga todos os LEDs: 0b00000000 (bin) ou 0 (dec)
            portd = 0;
            // Aguarda 1 segundo.
            delay_ms(1000);
            // Liga todos os LEDs - 0b11111111 (bin) ou 255 (dec)
            portd = 255;
            // Aguarda 1 segundo.
            delay_ms(1000);
        }
    }
}
```

1.2 Criando e Compilando um Projeto

- **1º Passo:** Crie uma pasta de nome ‘‘`Temp`’’ no caminho “`C:\`” ou utilize uma outra pasta vazia. Voce pode criar uma pasta vazia em seu *pen-drive*, assim seus arquivos criados estarão sempre salvos com voce.

- **2º Passo:** Dentro dessa pasta ‘‘`Temp`’’ serão salvos todos os projetos desse guia de aulas. Para cada projeto é importante que seja criada uma nova pasta vazia. Nessa primeira aula, crie uma pasta chamada ‘‘`piscapisca`’’.

- **3º Passo:** Salve o código mostrado na seção 1.1 na pasta ‘‘`piscapisca`’’. Esse arquivo deverá ser salvo com o nome ‘‘`piscapisca.c`’’.

- **4º Passo:** Execute o *software* ‘‘`mikroC PRO for PIC`’’[1]. Será então mostrada uma tela igual à exibida na Figura 2, essa é a tela principal do *software*.

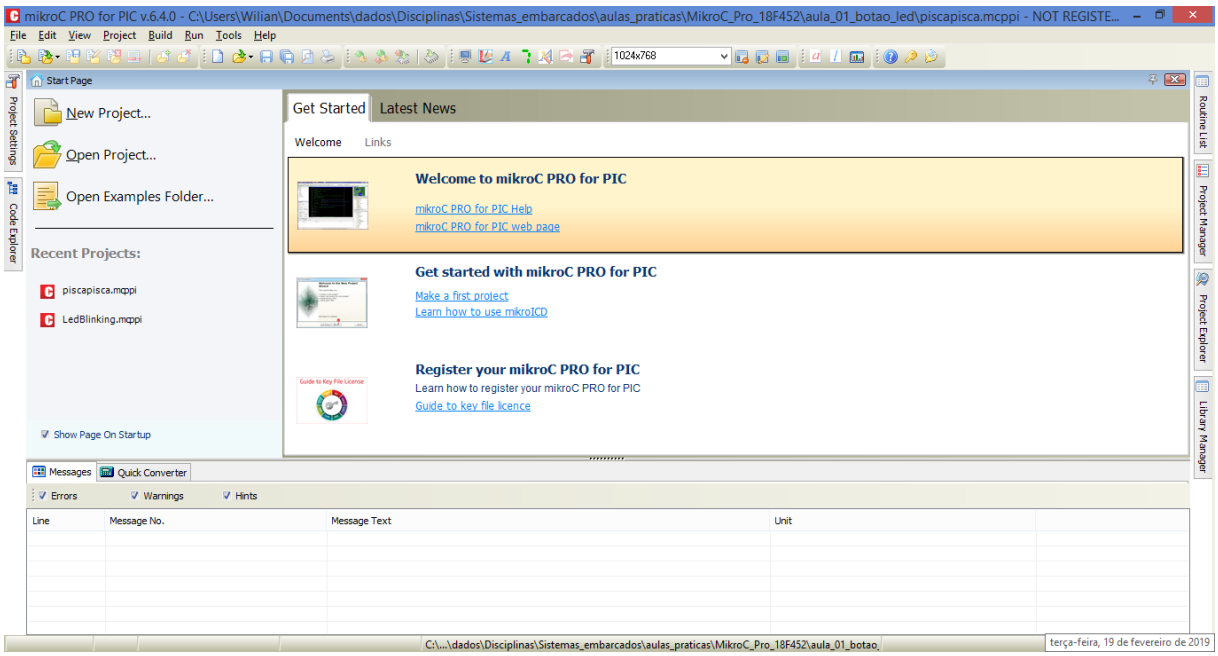

Figura 2: 4º Passo.

- **5º Passo:** O próximo passo é a criação de um novo projeto. Para isso, selecione o menu ‘‘`File`’’, em seguida a opção ‘‘`New`’’, e então por último o item ‘‘`New Project...`’’. A sequência de passos é mostrada na Figura 3.

[1] Neste guia a versão utilizada é a 6.4

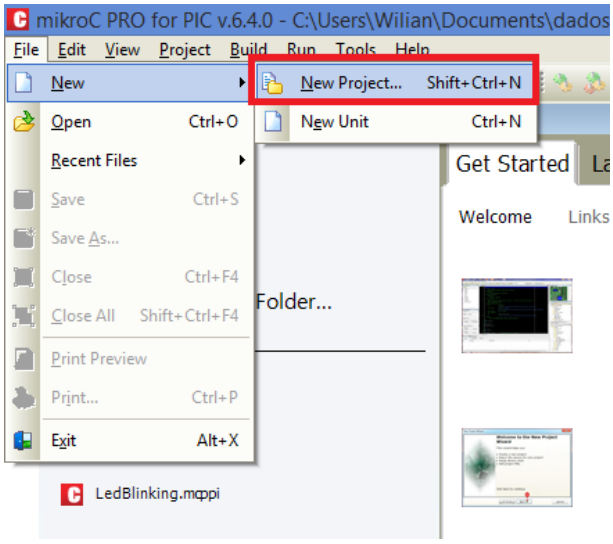

Figura 3: 5º Passo.

- **6º Passo:** Será exibida uma tela assim como a mostrada na Figura 4. Selecione a opção ‘‘`Next`’’.

Figura 4: 6º Passo.

- **7º Passo:** A próxima tela exibida é a de configuração do projeto que está sendo criado como mostrado na Figura 5. Nela preencha os campos da seguinte maneira:

  – **Project Name:** é o nome do projeto que está sendo criado, e deve ser igual ao nome do arquivo .c que contem a função *“main”*. No projeto da primeira aula preencha esse campo com o nome ‘‘`piscapisca`’’.

  – **Project Folder:** local onde o projeto será criado. Clique no botão ‘‘`Browse`’’ e navegue até a pasta criada anteriormente no 2º Passo.

– **Device Name:** modelo do microcontrolador PIC que será utilizado. Nesse caso selecione a opção ‘‘P18F452’’.

– **Device Clock:** oscilador de *clock* que será utilizado no microcontrolador. Nesse projeto preencha esse campo com o valor ‘‘8.000000 MHz’’.

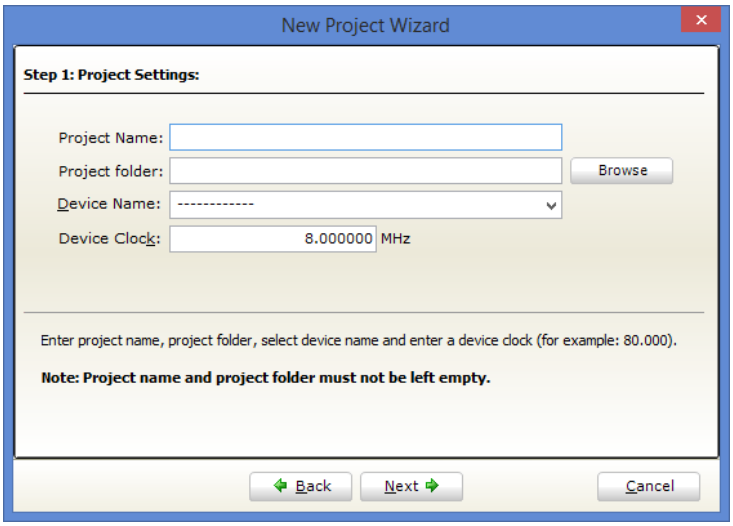

Figura 5: 7º Passo.

Em seguida, selecione a opção ‘‘Next’’.

- **8º Passo:** Nesse passo, o arquivo “.c” salvo anteriormente será adicionado ao projeto. Para isso, na janela exibida (Figura 6) selecione o ícone de pasta, será aberta então uma outra janela para seleção de arquivos. Navegue até a pasta do projeto e selecione o arquivo ‘‘piscapisca.c’’. Em seguida clique no botão ‘‘Add’’, o arquivo então será carregado para o projeto e estará pronto para ser processado.

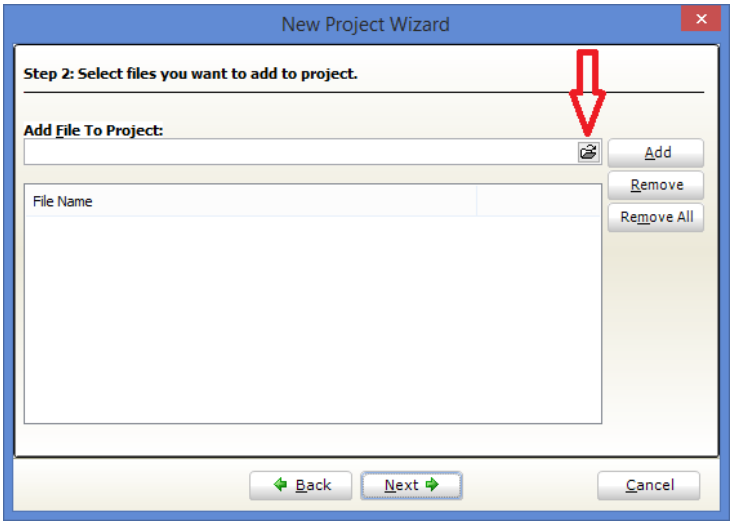

Figura 6: 8º Passo.

Em seguida, selecione a opção ‘‘`Next`’’.

- **9º Passo:** Em seguida é exibida uma tela, como a mostrada na Figura 7. Um arquivo de projeto escrito em linguagem “.c” pode utilizar diversas funções já prontas e que são disponibilizadas em forma de bibliotecas. Para que o usuário não tenha que adicionar manualmente cada uma das bibliotecas que implementam as funções utilizadas no código, deve-se selecionar a opção ‘‘`Include All (Default)`’’. Em seguida deve-se clicar no botão ‘‘`Next`’’.

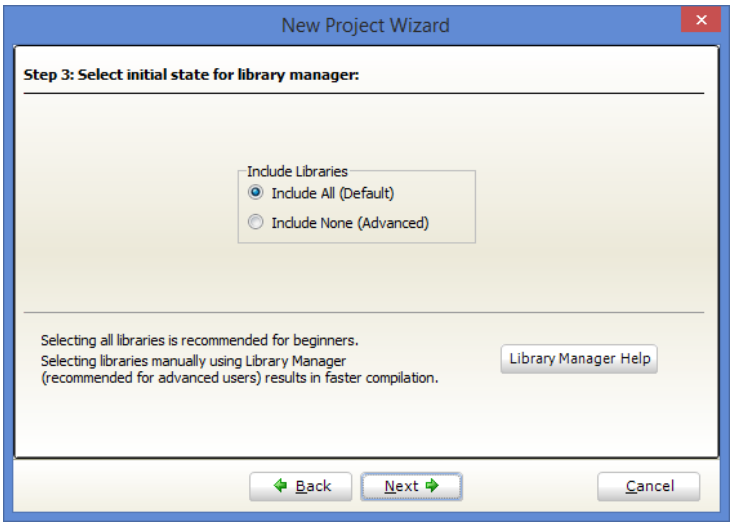

Figura 7: 9º Passo.

- **10º Passo:** Em seguida habilite a opção ‘‘`Open Edit Project window to set Configuration bits`’’ na janela mostrada na Figura 8. Em seguida selecione o botão ‘‘`Finish`’’ e uma outra janela para configurações avançadas de compilação abrirá.

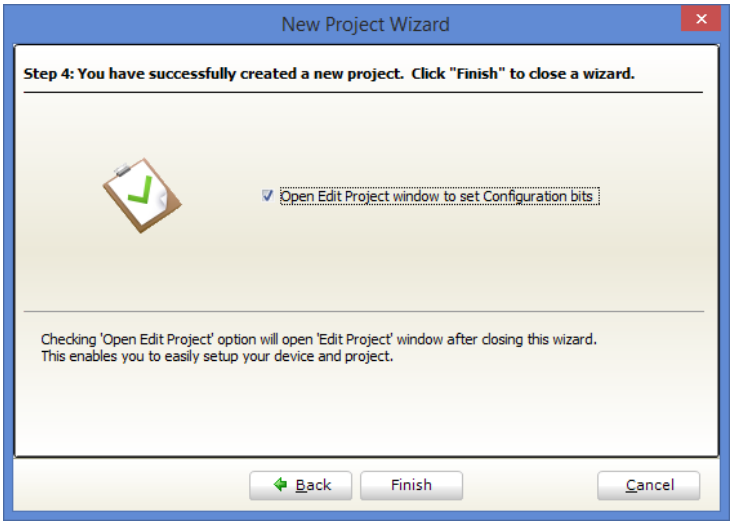

Figura 8: 10º Passo.

- **11º Passo:** Após realizar as configurações básicas no projeto, o próximo passo é ajustar as configurações avançadas. Na Figura 9 é mostrada a tela exibida no *software* durante esse passo. Nas opções à disposição deve-se atentar para o item ‘‘`Oscilator Selection`’’ que deve ser configurado para utilizar HS oscillator para habilitar o funcionamento do sistema com o cristal de *clock* de 8MHz. As demais opções deverão ser alteradas para ‘‘`Disabled`’’. Feito isso deve-se clicar no botão ‘‘`OK`’’.

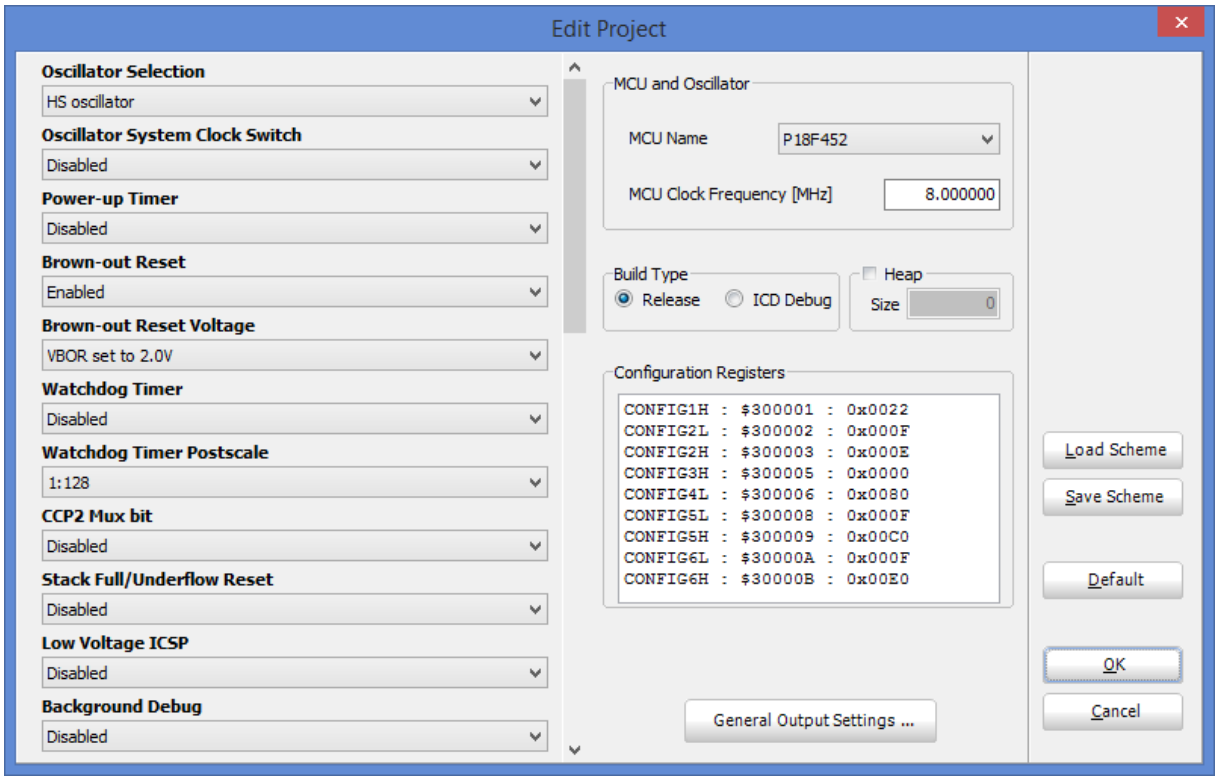

Figura 9: 11º Passo.

- **12º Passo:** Caso todo o processo de criação do projeto tenha sido feito corretamente, a tela mostrada na Figura 10, deverá ser exibida. Nela será possível visualizar o código “.c” do exemplo.

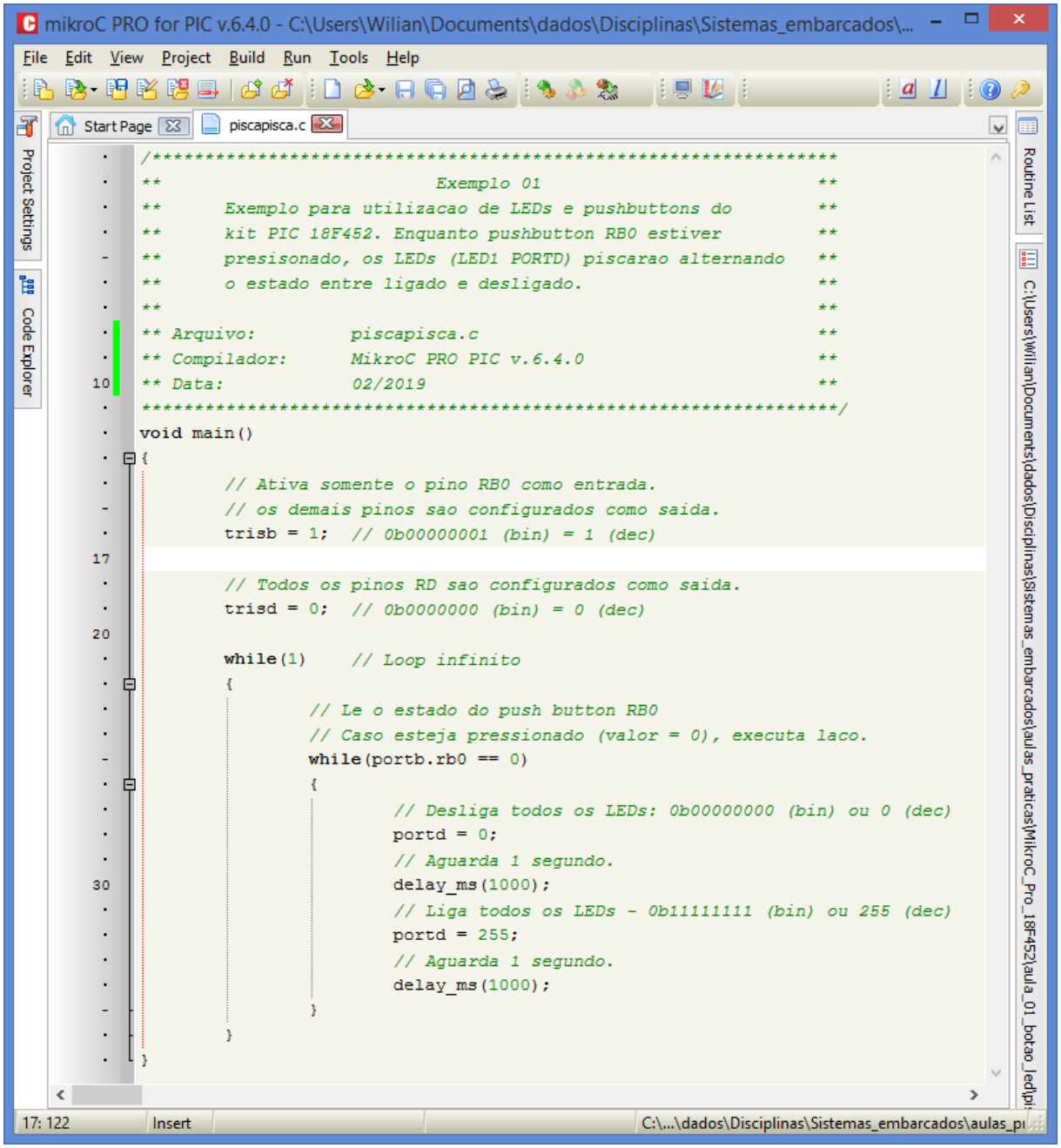

Figura 10: 12º Passo.

- **13º Passo:** O próximo passo a ser executado é a compilação do código ".c" importado no projeto. Para isso deve-se selecionar o menu "Build" e em seguida clicar no botão ''`Build`'' (esse processo pode ser substituído pela combinação de teclas de atalho ''`Ctrl+F9`''). Essa etapa é mostrada na Figura 11.

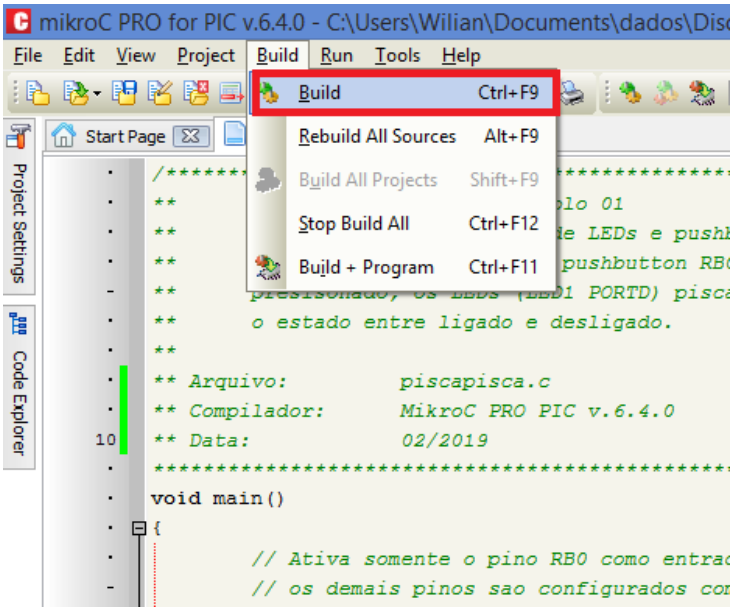

Figura 11: 13º Passo.

- **14º Passo:** Para verificar se a compilação foi bem sucedida, o usuário pode abrir o janela de mensagens. Para isso, basta acessar o menu ‘‘`View`’’, e em seguida habilitando a opção ‘‘`Messages`’’. Caso a compilação tenha sido executada com sucesso será possível visualizar a seguinte mensagem: ‘‘`Finish sucessfully`’’ e também o texto ‘‘`Compiled`’’ no rodapé da janela, como mostrado na Figura 12.

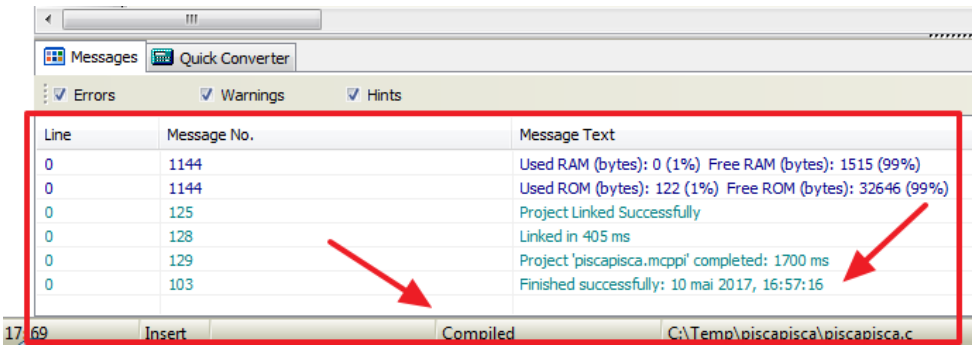

Figura 12: 14º Passo.

- **15º Passo:** Abra a pasta onde o projeto foi criado utilizando um visualizador de arquivos (Windows Explorer). Nesse momento, existirão diversos arquivos gerados na fase de criação e compilação do projeto. Entre eles, destaca-se o arquivo com extensão “.hex”, o qual contém o código com as instruções em linguagem de máquina que será executado no microcontrolador. Utilizando o “Bloco de Notas” do Windows, abra esse arquivo e observe como o código em linguagem C é diferente do código compilado (veja Figura 13).

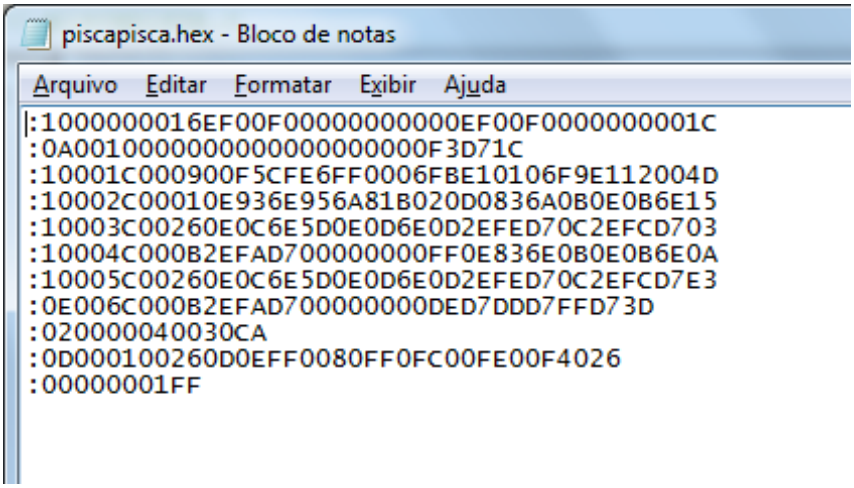

Figura 13: 15º Passo.

1.3 Preparando o Kit

- **1º Passo:** Solicite o kit ao professor e o coloque na bancada com cuidado, sem tocar em nenhuma parte dos circuitos eletrônicos. Veja Figura 14.

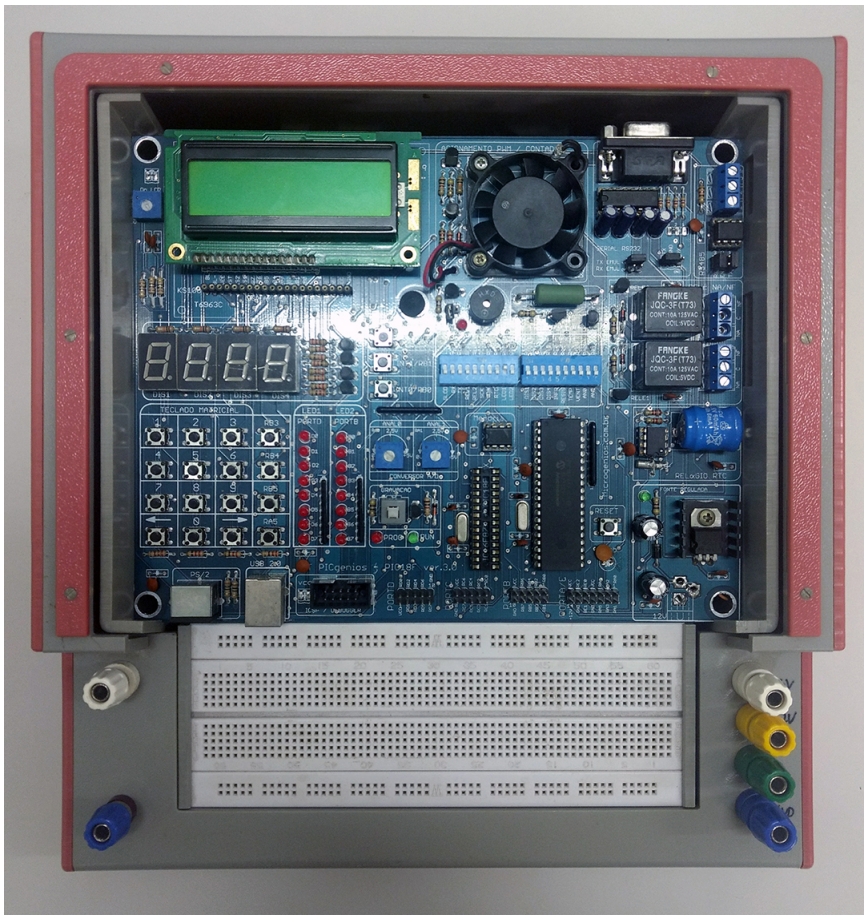

Figura 14: 1º Passo.

- **2º Passo:** Conecte o programador à porta "`ICSP/Debugger`". Coloque o programador em cima da *protoboard* sem que este toque em nenhuma parte metálica do kit (Figura 15) - **CUIDADO: risco de danos ao equipamento.**

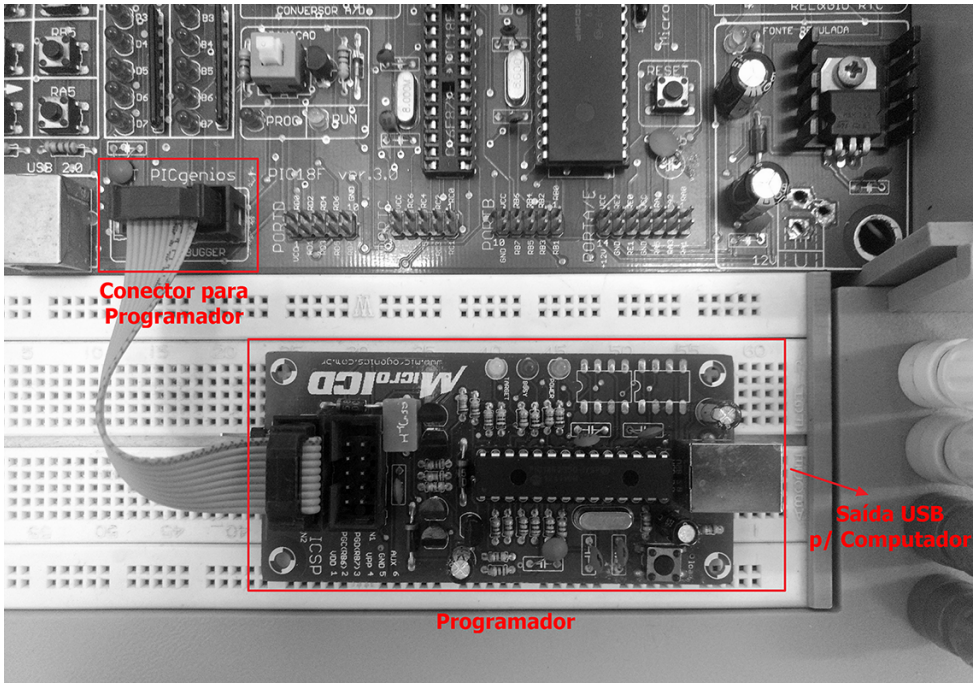

Figura 15: 2º Passo.

- **3º Passo:** Conecte a outra extremidade do programador ao cabo USB fornecido, e em seguida conecte este ao seu computador de trabalho (Figura 16).

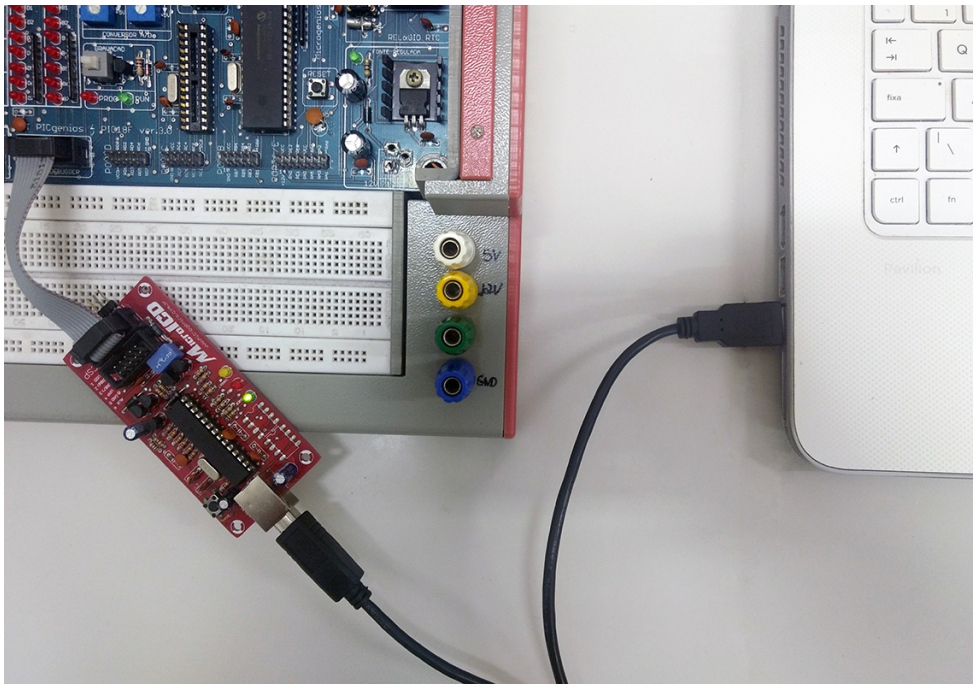

Figura 16: 3º Passo.

- **4º Passo:** O último passo na montagem do kit é a ligação deste à eletricidade. Portanto, conecte o cabo de alimentação ao kit e em seguida conecte este à tomada da bancada.

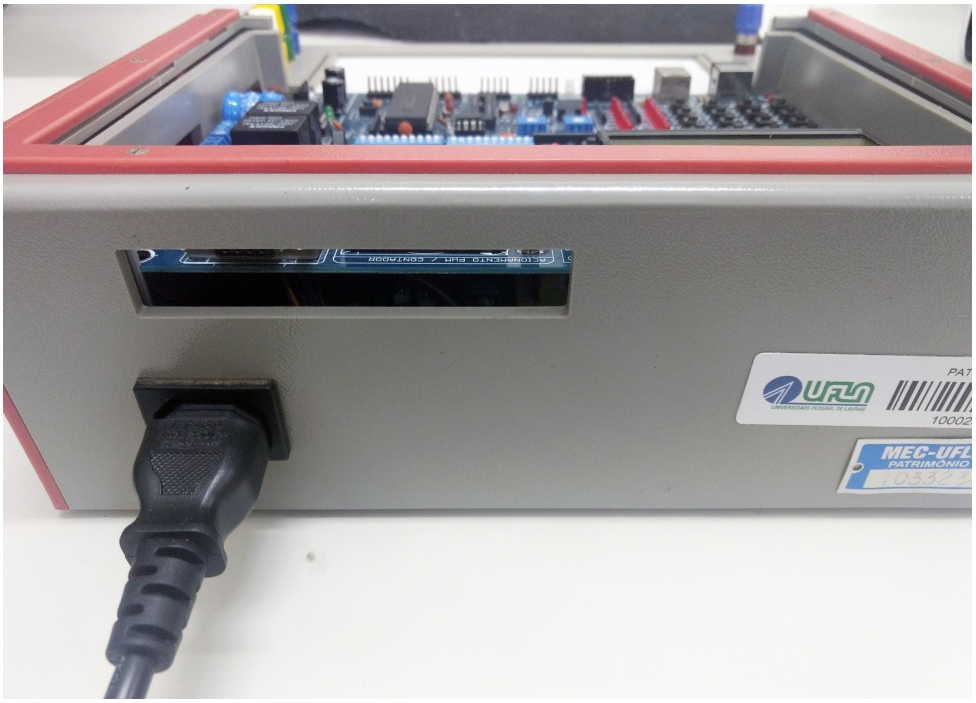

Figura 17: 4º Passo.

- **5º Passo:** Atente-se para que as chaves DIP do kit estejam configuradas de acordo com as interfaces que serão utilizadas nos projetos. No caso do projeto da primeira aula, é necessário que a chave ‘‘`LED1 - PORTD`’’ esteja ligada (posição “ON”) para que os LEDs da fileira “1” operem. A Figura 18 exibe a configuração das chaves.

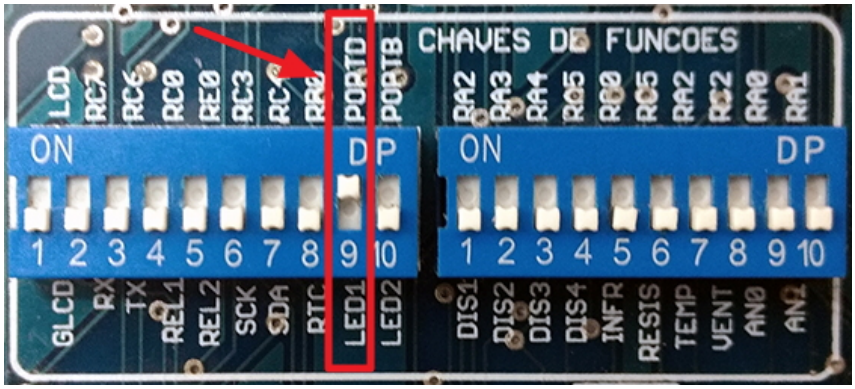

Figura 18: 5º Passo.

1.4 Gravando o Projeto

- **1º Passo:** Compilado o projeto e montado o kit, o próximo passo é enviar o arquivo ".hex" para o microcontrolador. Para isso, execute o *software* ''`PICkit 2`''. Será então mostrada uma tela igual a exibida na Figura 19, essa é a tela principal do *software* que realizará a programação do kit. **Atenção: o kit já deve estar conectado no PC pelo gravador.**

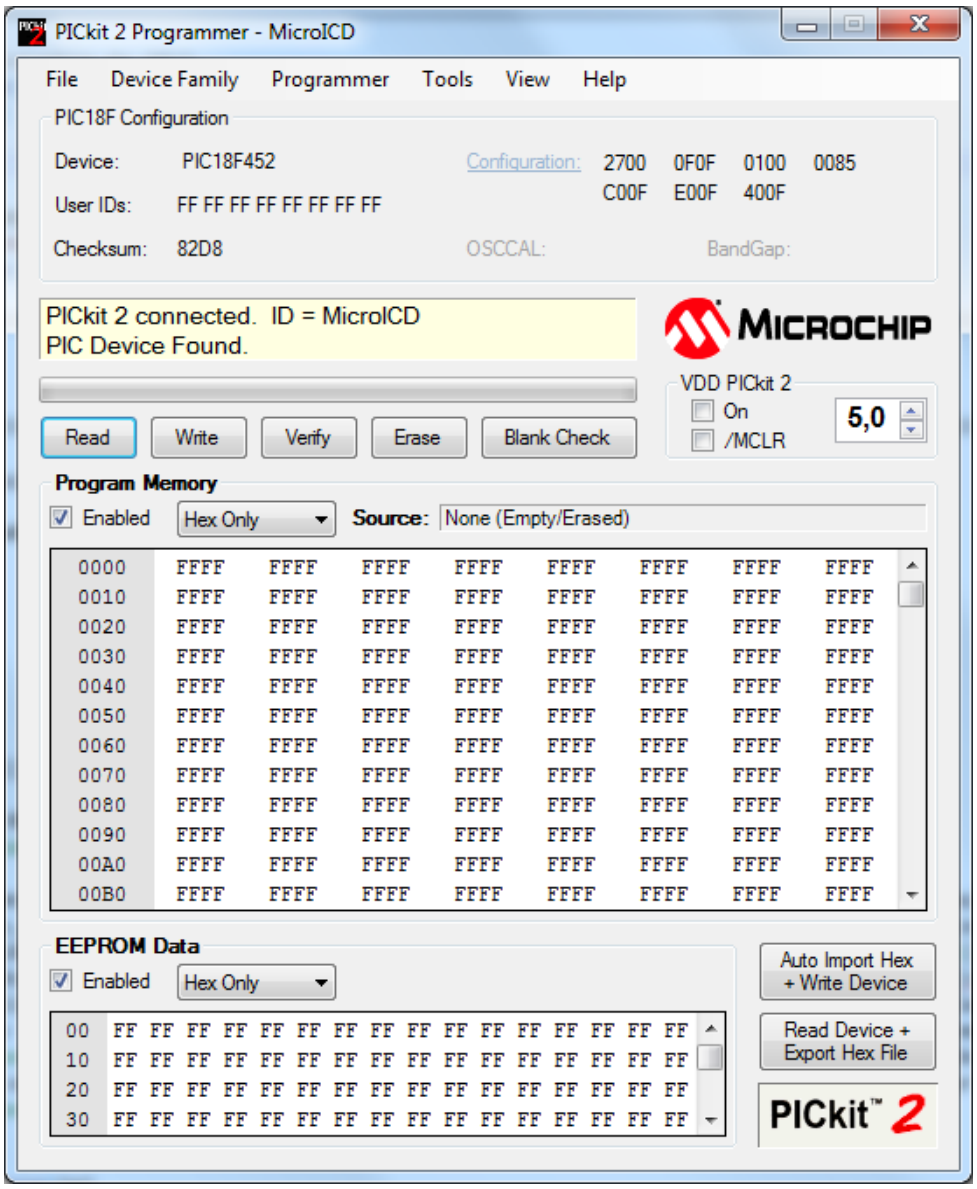

Figura 19: 1º Passo.

- **2º Passo:** Em seguida deve-se importar o arquivo ".hex" para o *software*. Para isso, selecione o menu ''`File`'' e em seguida a opção ''`Import Hex`'', como mostrado na Figura 20. Será então aberta uma janela para seleção do arquivo, navegue até a pasta onde o projeto foi salvo, selecione o arquivo ".hex" e em seguida

clique no botão ‘‘Abrir’’.

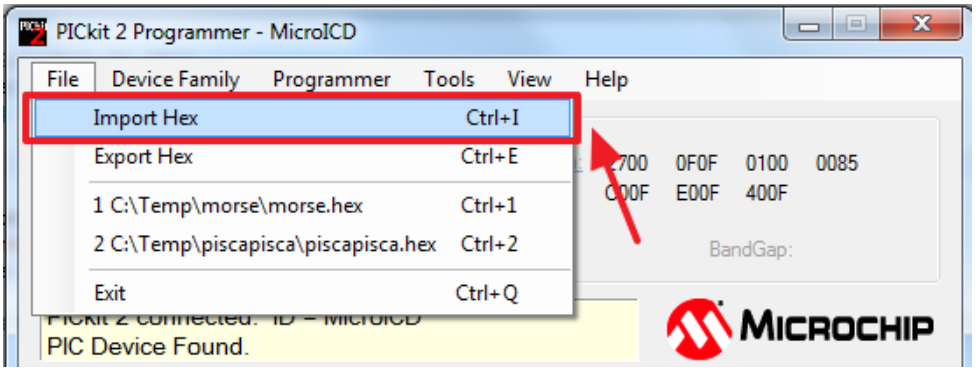

Figura 20: 2º Passo.

- **3º Passo:** Caso a importação do arquivo tenha sido bem sucedida, será exibida uma mensagem no *software* como a mostrada na Figura 21. Em seguida clique no botão ‘‘Write’’ para que o código seja carregado no microcontrolador.

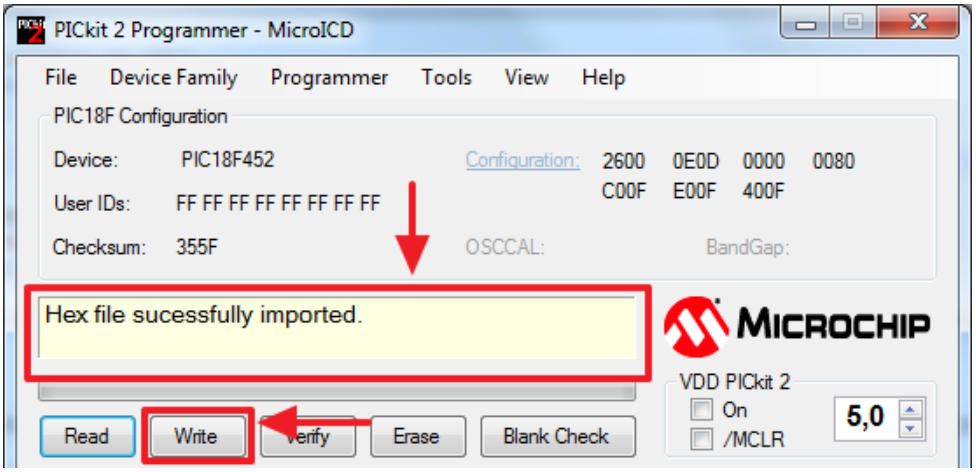

Figura 21: 3º Passo.

- **4º Passo:** Se a gravação ocorrer sem problemas, será exibida a mensagem ‘‘Programming Succesfull’’, como mostrado na Figura 22.

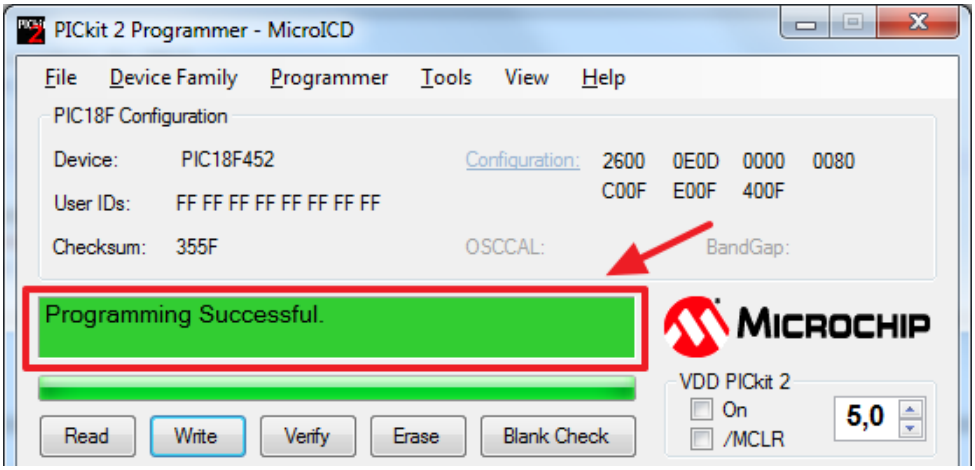

Figura 22: 4º Passo.

Após a execução de todos esses passos, o kit PICGenios está pronto para a execução do código implementado. Dessa forma, enquanto o *pushbutton* "RB0" estiver pressionado, todos os oito LEDs da fileira "LED1" alternarão entre ligados e desligados a cada 1 segundo. Caso o *pushbutton* não seja pressionado, os LEDs ficarão acesos.

1.5 Atividade Prática

Utilizando como base o código desenvolvido nessa aula e a Figura 23, crie um novo projeto seguindo os passos exemplificados. Por fim, teste a implementação utilizando o kit PICGenios.

A ·−	J ·−−−	S ···	2 ··−−−
B −···	K −·−	T −	3 ···−−
C −·−·	L ·−··	U ··−	4 ····−
D −··	M −−	V ···−	5 ·····
E ·	N −·	W ·−−	6 −····
F ··−·	O −−−	X −··−	7 −−···
G −−·	P ·−−·	Y −·−−	8 −−−··
H ····	Q −−·−	Z −−··	9 −−−−·
I ··	R ·−·	1 ·−−−−	0 −−−−−

Figura 23: Código Morse.

O projeto a ser desenvolvido deve realizar as seguintes funções:

- Quando o *pushbutton* "RB0" for acionado, um LED emitirá o código Morse das iniciais do nome do aluno. Ex: para o aluno de nome José de Souza Martins, deverá ser exibido o código morse para as letras "JSM".

- Quando o outro *pushbutton* (“RB1”) for acionado, o mesmo LED emitirá o código Morse do número de matrícula do aluno.

São sugeridos os intervalos de tempo apresentados na Figura 24 para gerar os símbolos do código, tomando como exemplo as letras “A” e “B”.

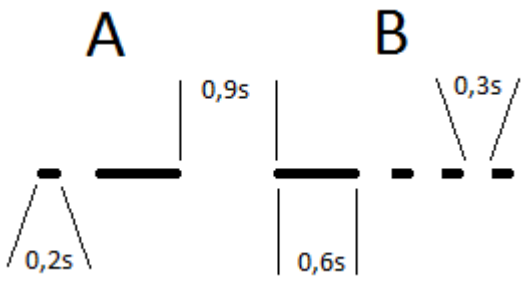

Figura 24: Temporização do Código Morse.

Observações:

- Exercício individual, mas pode ser feito em grupo. Cada aluno entrega o seu exercício.
- Postar um arquivo “.pdf” com o programa fonte em linguagem C contendo os devidos comentários na sala virtual da disciplina, até a data combinada. Não compactar o arquivo postado.
- Apresentar o exercício funcionando ao professor durante a aula prática. Isto é essencial para contabilizar a avaliação do exercício.
- Não será permitido a postagem do exercício depois do prazo combinado.

2 Aula 02

Na segunda aula prática do curso será mostrado como operar o *buzzer* que está instalado no kit PICGenios. O esquemático mostrado na Figura 25 é semelhante ao utilizado na primeira aula, a principal alteração é adição do componente *buzzer*. O exemplo mostrado nessa aula, e que pode ser conferido no código ".c" da seção 2.1, simplesmente ligará e desligará o *buzzer* do kit durante um intervalo de tempo predefinido (nesse exemplo utiliza-se 1 segundo). Para criação de projeto, configuração do kit e gravação do projeto, siga os passos mostrados anteriormente na Aula 01.

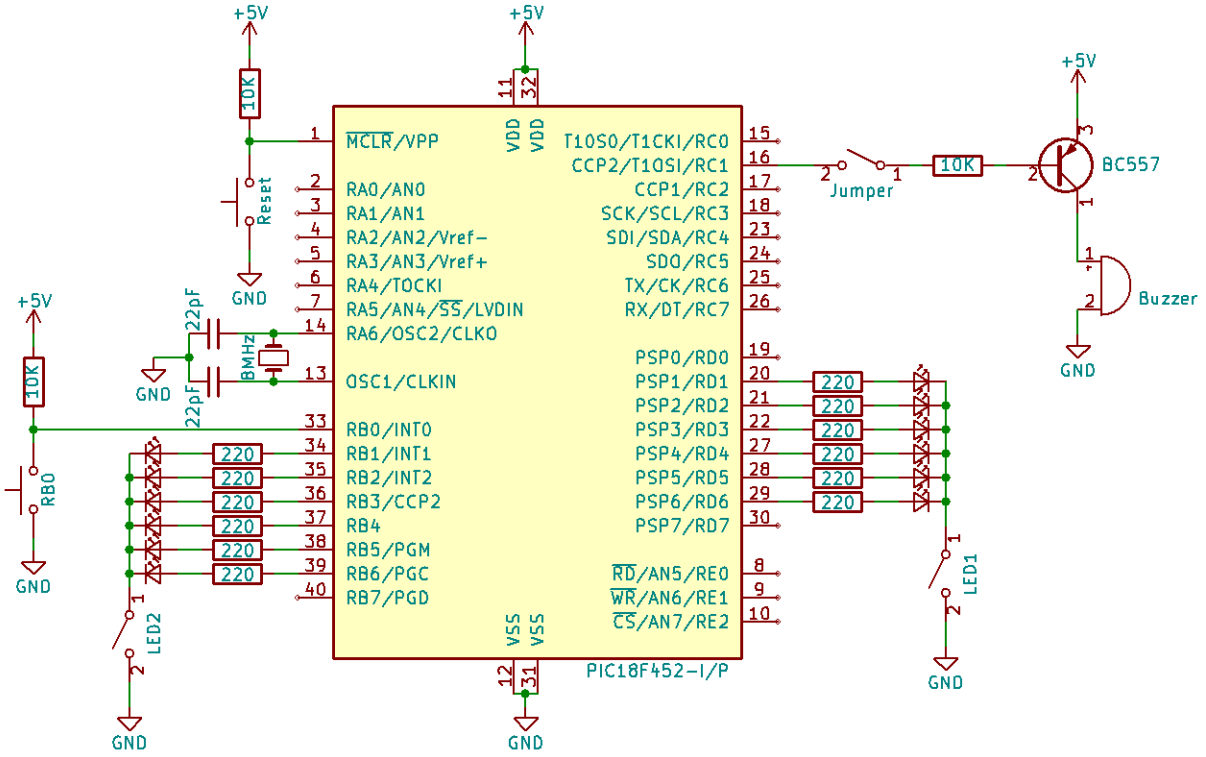

Figura 25: Esquemático do projeto.

O *buzzer* possui dois terminais com polaridade positiva e negativa já soldados no kit. Sua forma de funcionamento depende de seu fabricante. Alguns utilizam componentes eletromecânicos (indutor, membrana, etc...), e outros circuitos eletrônicos para gerar um oscilador e assim emitir sinal sonoro. O *buzzer* é conectado ao microcontrolador por meio de um transistor `PNP BC557`, atuando como uma carga deste. A base do transistor é conectada ao pino "`RC1`" em série com um resistor de $10k\Omega$ para limitar a corrente de base. Por fim, o coletor é ligado ao *buzzer* e o emissor ao $+5V$.

O pino "`RC1`" já é por padrão utilizado no kit para acionamento do *buzzer*. Quando o pino "`RC1`" (configurado com saída digital) está com nível lógico "1", então $+5V$ é aplicado na base (N) do transistor que é igual a tensão do emissor (P). Assim,

não existe diferença de potencial entre emissor e base, não gerando corrente de base, e consequentemente corrente de coletor - $I_c = \beta * I_b$. Ou seja, não existe corrente passando pelo *buzzer* e este não gera sinal sonoro (silêncio).

Quando o pino ‘‘`RC1`’’ (configurado com saída digital) está com nível lógico “0”, então um potencial de $0V$ é aplicado na base (N) do transistor. Assim, existe uma diferença de potencial entre emissor e base, polarizando a junção base-emissor diretamente, permitindo uma corrente de base, e consequentemente corrente de coletor - $I_c = \beta * I_b$. Ou seja, existe corrente passando pelo *buzzer* e este gera um sinal sonoro (apito).

Dessa forma o *buzzer* é ativo em nível lógico baixo (“0”) pois o valor de $0V$ permite que exista corrente passando no coletor do transistor e consequentemente no *buzzer*.

2.1 Código do Projeto

```
/*****************************************************************
**                          Exemplo 02                          **
**                                                              **
**      Exemplo para utilizacao de buzzer do kit PIC 18F452.    **
**      Ao carregar o programa no kit, o buzzer desse sera      **
**      acionado por um segundo e aguardara mais um segundo     **
**      ate o proximo acionamento.                              **
**                                                              **
** Arquivo:     buzzer.c                                        **
** Compilador:  MikroC PRO PIC v.6.4.0                          **
**                                                              **
** UFLA - Lavras/MG - 24/05/2017                                **
*****************************************************************/

void main()
{
    // trisc e uma posicao (registrador) na memoria de dados
    // do microcontrolador. Configura o pino RC1 como saida para
    // comunicacao com o buzzer.
    trisc = 0xFD; //0b11111101

    // Inicia com buzzer desligado.
    portc.rc1 = 1;

    // Loop infinito.
    while(1)
    {
        // Alterna estado do buzzer (ligado -> desligado -> ligado)
        portc.rc1 = ~portc.rc1;

        // Aguarda 1s ate a proxima execucao do laco.
        delay_ms(1000);
    }
}
```

OBS 1: Para que o *buzzer* funcione, tanto no código de exemplo quanto no exercício prático, é necessário que o *jumper* localizado ao seu lado esteja presente. Esse componente é mostrado na Figura 26.

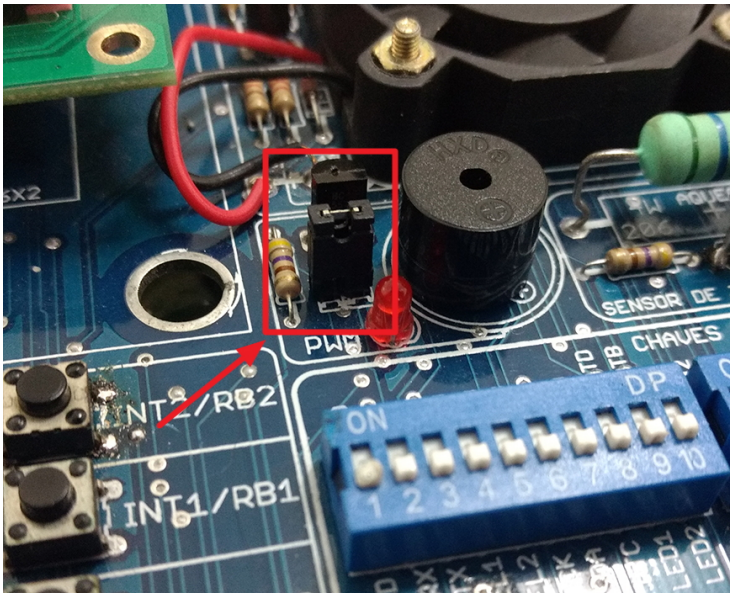

Figura 26: Jumper para conexão do *buzzer*.

2.2 Atividade Prática

Utilizando como base o código desenvolvido nessa aula, crie um novo projeto seguindo os passos dados na Aula 01. Por fim, teste a implementação utilizando o kit PICGenios. O projeto a ser desenvolvido deve realizar as seguintes funções para simular o lançamento de dois dados de 6 faces numeradas:

- Enquanto um *pushbutton* for acionado, 6 LEDs da porta D e 6 LEDs da porta B do PIC acenderão ou piscarão em alta frequência.
- Assim que o *pushbutton* for liberado permanecerão acesos apenas um LED da porta B e um LED da porta D, correspondendo aos "dados" de números 1 a 6 sorteados.
- Se os números sorteados forem 6 e 6, então um alarme sonoro será ativado durante um intervalo de tempo.
- O sistema aguarda acionar novamente o *pushbutton* para repetir o processo.

OBS 2: Para execução do programa do exercício prático atente-se para que as chaves DIP do kit estejam configuradas da seguinte maneira: ligue a chave ''`LED1 - PORTD`'' e a chave ''`LED2 - PORTB`'' (posição "ON") para que os LEDs das fileiras "1" e "2" operem. A Figura 27 exibe a configuração das chaves.

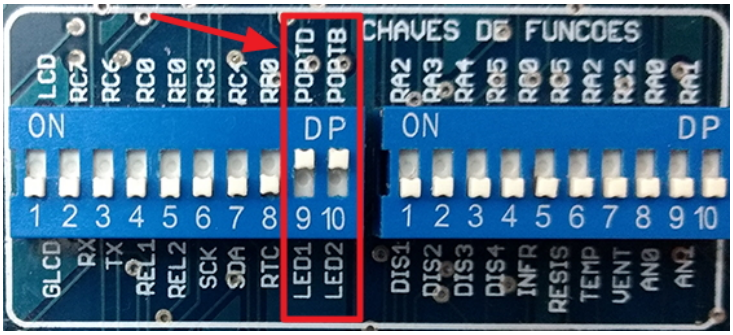

Figura 27: Configuração das chaves DIP.

Observações:

- Exercício individual, mas pode ser feito em grupo. Cada aluno entrega (posta) o seu exercício.
- Postar um arquivo ".pdf" com o programa fonte em linguagem C contendo os devidos comentários na sala virtual da disciplina, até a data combinada. Não compactar o arquivo postado.
- Apresentar o exercício funcionando ao professor durante a aula prática. Isto é essencial para contabilizar a avaliação do exercício.
- Não será permitido a postagem do exercício depois do prazo combinado.

Uma forma de implementação desse exercício é exemplificado pelo fluxograma mostrado na Figura 28. Nessa abordagem, na etapa ''`Inicialização`'' as variáveis (Dado1 e Dado2) são inicializadas com valor ''`1`''. Em seguida, no estado ''`Mostra`'' são exibidos os valores atuais de cada dado através dos LEDs. Feito isso, verifica-se se o "pushbutton" foi pressionado, caso não tenha sido, e os dados não são ambos iguais a ''`6`'', retorna e aguarda novamente o pressionar do "pushbutton". Caso nessa etapa verifica-se que os dois dados sejam iguais a 6, apita o *buzzer*. Se o "pushbutton" está pressionado no estado de decisão ''`Botão?`'', incrementa valor do Dado1. Caso esse valor seja maior que ''`6`'' atualiza seu valor para ''`1`'', e então segue para realizar o mesmo procedimento para o Dado2. Porém, caso o Dado1 não seja maior que ''`6`'' aguarda "pushbutton" novamente.

Nessa abordagem é praticamente impossível que o usuário consiga soltar o *"pushbutton"* quando os dados estiverem nos valores desejados, evitando fraude. Outra maneira para implementar esse exercício é utilizando a função ''`rand`'' disponível na biblioteca do compilador C. Caso opte por utilizar essa biblioteca, consulte o *help* do compilador utilizado.

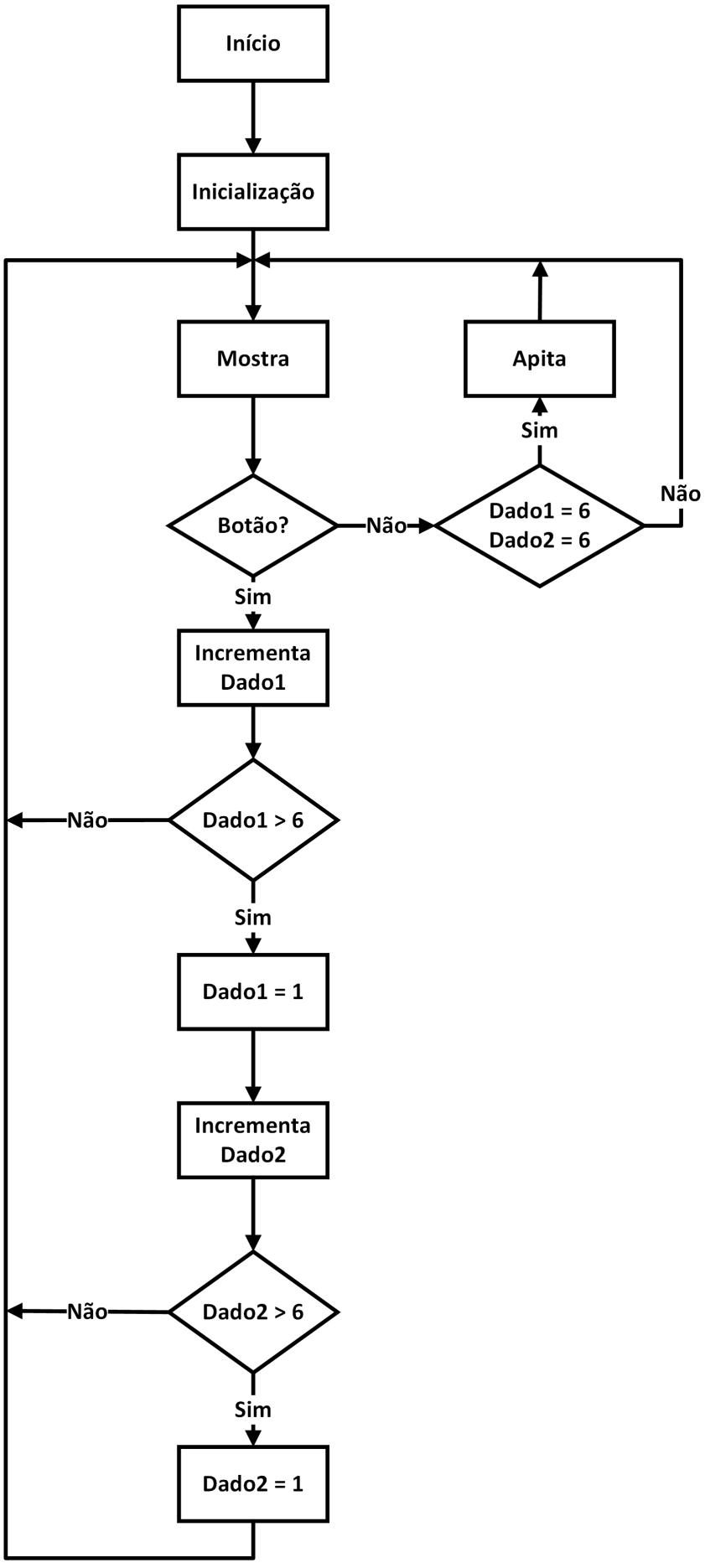

Figura 28: Fluxograma auxiliar para implementação.

Após a execução de todos esses passos, o kit PICGenios está pronto para a execução do código implementado. Dessa forma, enquanto o *pushbutton* “RB0” estiver pressionado, os seis LEDs da fileira “LED1” e os seis LEDs da fileira “LED2” piscarão aleatoriamente em alta frequência. Ao soltar o *pushbutton*, os dois valores sorteados deverão ser exibidos nos LEDs, que permanecerão acessos até a próxima execução. Caso os dois valores sejam iguais a “6”, o *buzzer* é ativado.

3 Aula 03

Na terceira aula prática do curso será mostrado como operar o *display* LCD que está instalado no kit PICGenios. O esquemático mostrado na Figura 29 descreve os componentes utilizados no projeto, onde a principal novidade nesse é a inclusão do *display*. O exemplo mostrado nessa aula, e que pode ser conferido no código ".c" da seção 3.1, simplesmente realizará a configuração do *display* e em seguida mostrará a frase ''`Hello World`''. Para criação de projeto, configuração do kit e gravação do projeto, siga os passos mostrados anteriormente na Aula 01.

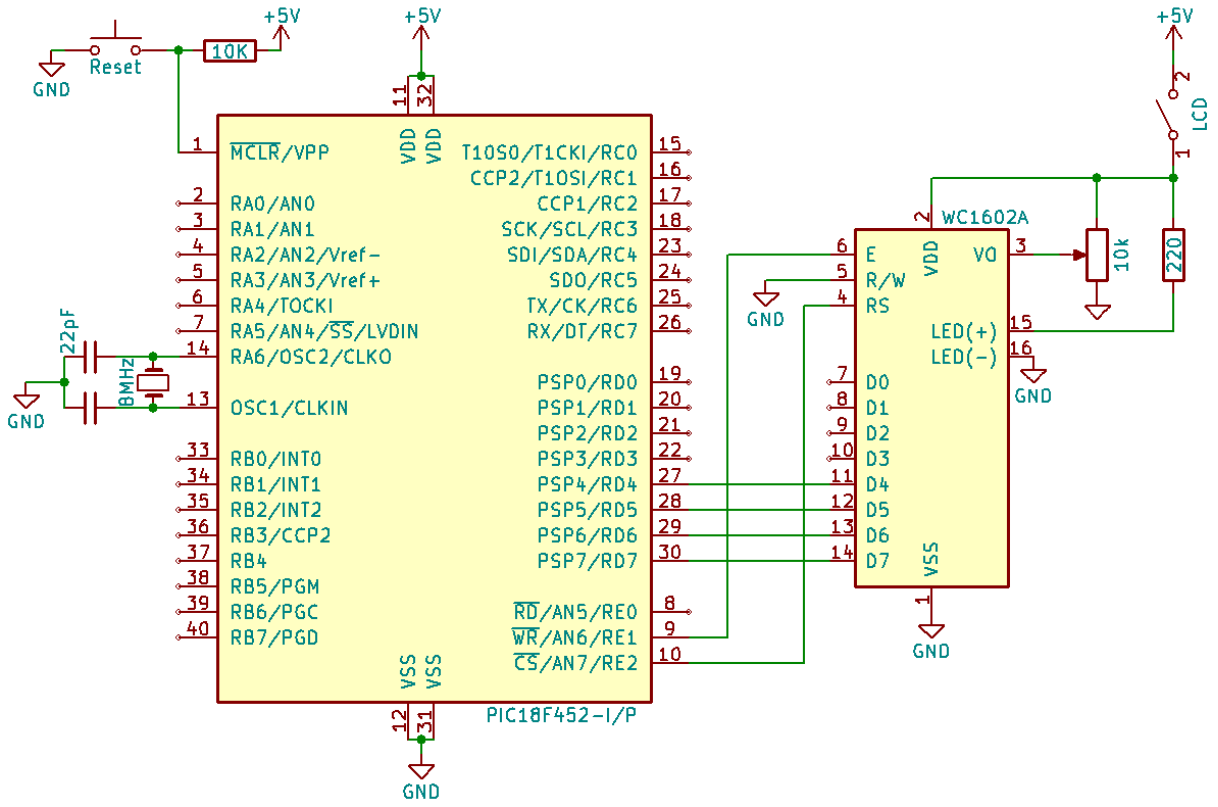

Figura 29: Esquemático do projeto.

3.1 Código do Projeto

```
/************************************************************
**                          Exemplo 03                     **
**     Exemplo para utilizacao do LCD presente no kit PIC  **
**     18F452. O programa inicializa o display e em seguida **
**     escreve a frase "Hello World".                       **
**                                                          **
** Arquivo:    lcd.c                                        **
** Compilador: MikroC PRO PIC v.6.4.0                       **
** Data:       06/2017                                      **
************************************************************/

// Conexoes LCD do kit PICGenios com 18F
sbit LCD_RS at RE2_bit;
sbit LCD_EN at RE1_bit;
sbit LCD_D4 at RD4_bit;
sbit LCD_D5 at RD5_bit;
sbit LCD_D6 at RD6_bit;
sbit LCD_D7 at RD7_bit;
sbit LCD_RS_Direction at TRISE2_bit;
sbit LCD_EN_Direction at TRISE1_bit;
sbit LCD_D4_Direction at TRISD4_bit;
sbit LCD_D5_Direction at TRISD5_bit;
sbit LCD_D6_Direction at TRISD6_bit;
sbit LCD_D7_Direction at TRISD7_bit;

void main()
{
    ADCON0  = 0x00; // Configura todos pinos das portas para
       digital
    ADCON1  = 0x06; // e desabilita o conversor A/D

    Lcd_Init();                     //Inicializa o Display
    Lcd_Cmd(_LCD_CLEAR);            //Limpa o Display
    Lcd_Cmd(_LCD_CURSOR_OFF);       //Desabilita o cursor
    Lcd_Out(1,4,"Hello World!");    // Escreve no LCD na linha 1
       coluna 4

    while(1);
} // fim do programa
```

OBS 1: Para projetos que utilizem o módulo LCD do kit, deve-se, ao criar o projeto, alterar a opção "`Oscillator Selection`" para a configuração "`HS oscillator`". Esse passo é mostrado na Figura 30

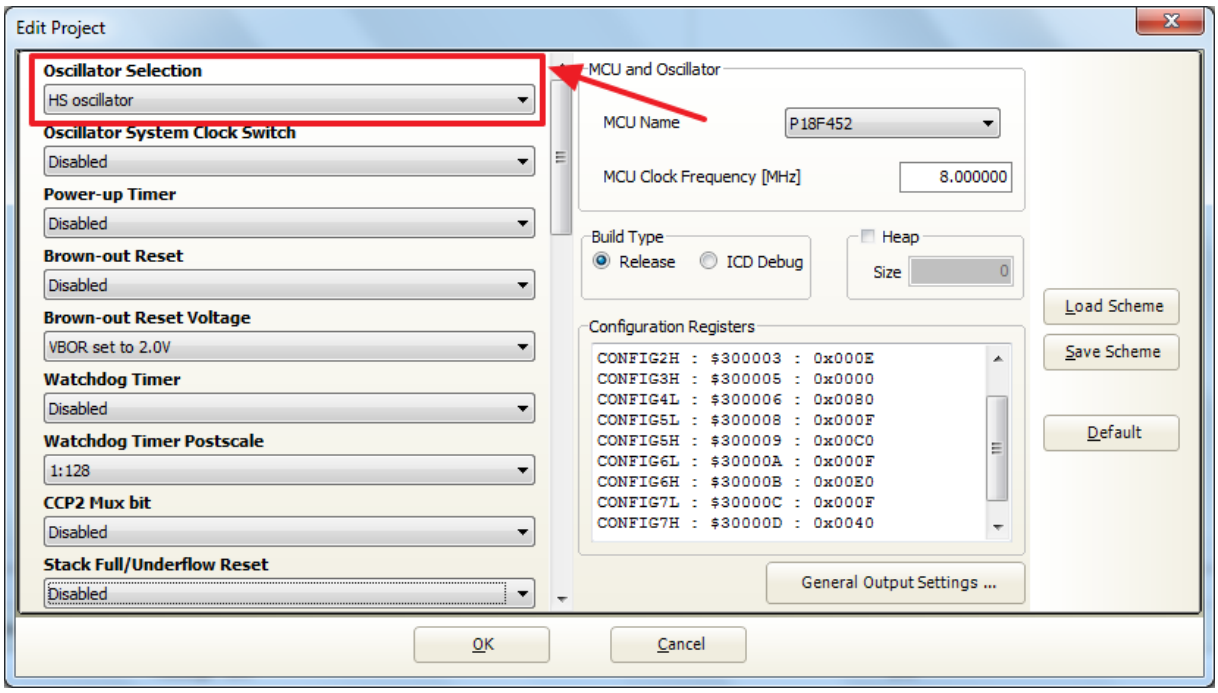

Figura 30: Alteração no modo oscilador.

3.2 Atividade Prática

Utilizando como base o código desenvolvido nessa aula, crie um novo projeto seguindo os passos dados na Aula 01. Por fim, teste a implementação utilizando o kit PICGenios. O projeto a ser desenvolvido deve realizar as seguintes funções:

- Conta e mostra a hora local no display LCD com: horas, minutos e segundos no formato 24 horas.
- Quando um *pushbutton* for acionado, a velocidade de *clock* aumenta para ajustar a hora correta e quando o *pushbutton* for solto, volta a velocidade de contagem de tempo normal.

OBS 2: Para execução do exercício prático atente-se para que as chaves DIP do kit estejam configuradas da seguinte maneira: ligue a chave "`GLCD-LCD`" (posição "ON") para que o *display* LCD possa funcionar corretamente. A Figura 31 exibe a configuração das chaves.

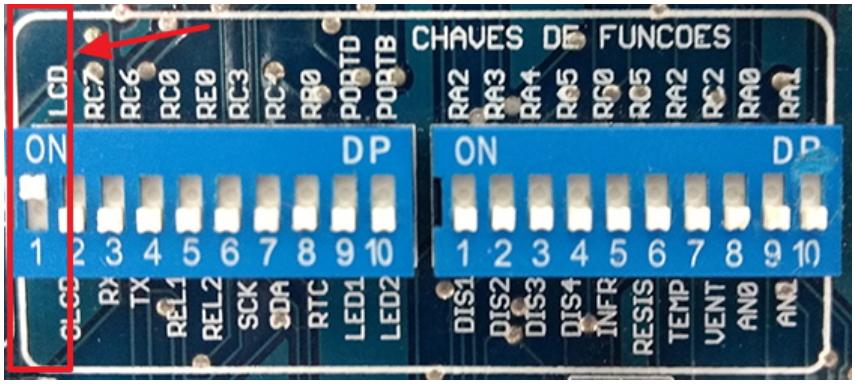

Figura 31: Configuração das chaves DIP.

Observações:

- Exercício individual, mas pode ser feito em grupo. Cada aluno posta o seu exercício.
- **Dica:** usar função `WordToStr()`, ou outra função de conversão para *string*, antes de mostrar a hora no LCD.
- Postar um arquivo ".pdf" com o programa fonte em linguagem C contendo os devidos comentários na sala virtual da disciplina, até a data combinada. Não compactar o arquivo postado.
- Apresentar o exercício funcionando ao professor durante a aula prática. Isto é essencial para contabilizar a avaliação do exercício.
- Não será permitido a postagem do exercício depois do prazo combinado.
- Uma forma de implementação desse problema é a mostrado no fluxograma da Figura 32.

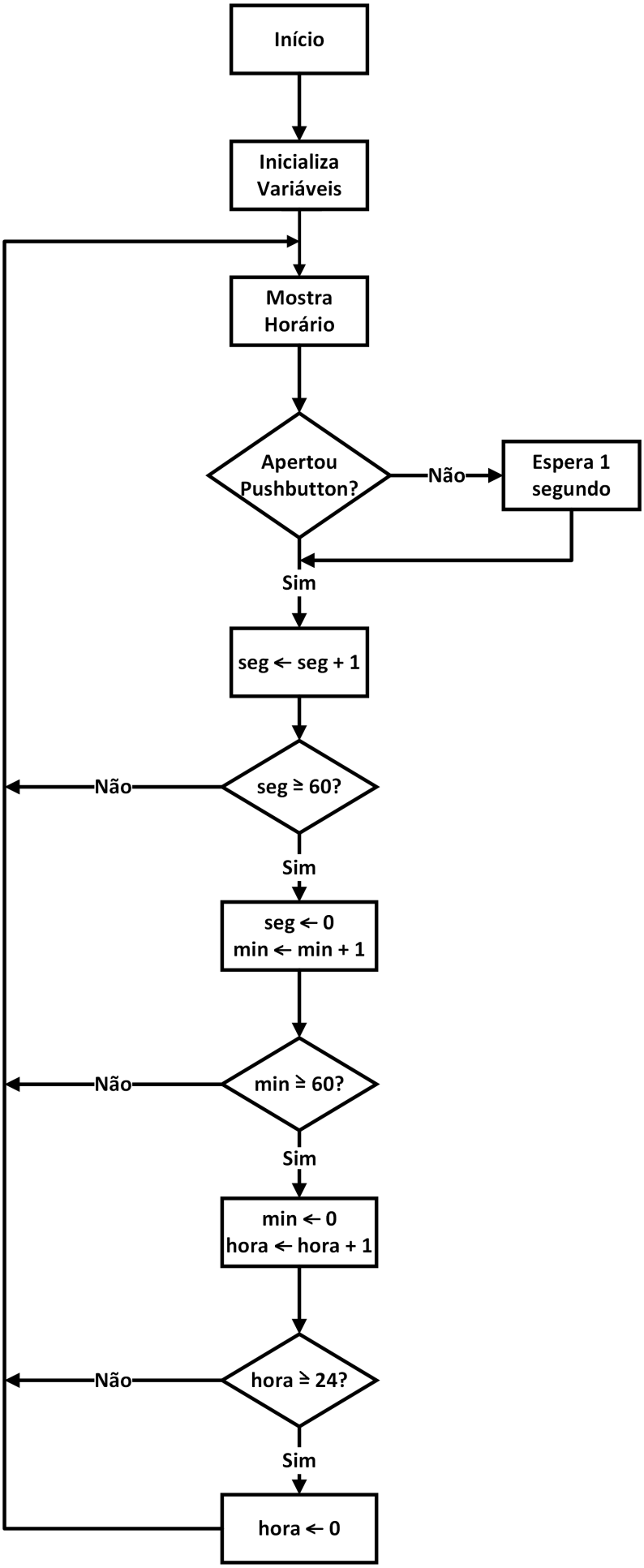

Figura 32: Fluxograma auxiliar para implementação.

Após a execução de todos esses passos, o kit PICGenios está pronto para a execução do código implementado. Assim, o relógio deverá mostrar a hora, atualizando o seu valor a cada segundo, exceto quando o *pushbutton* configurado for acionado. Quando isto ocorrer, a velocidade de atualização do relógio será realizada em alta velocidade, até que o usuário solte o *pushbutton* quando for a hora certa.

4 Aula 04

Na quarta aula prática do curso será mostrado como operar o teclado matricial instalado no kit PICGenios. O esquemático mostrado na Figura 33 descreve os componentes utilizados no projeto.

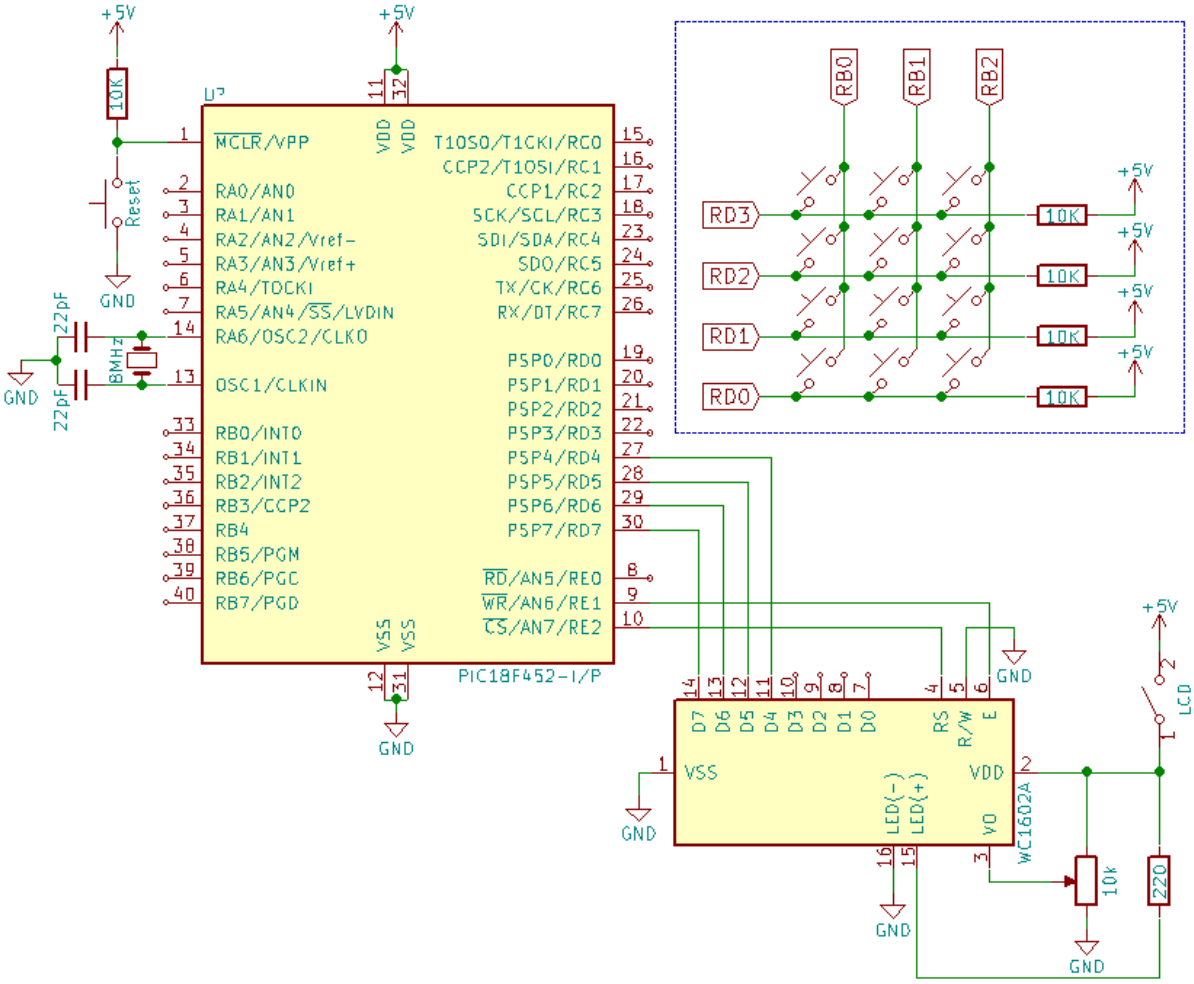

Figura 33: Esquemático do projeto.

O teclado matricial é um conjunto de *pushbuttons* que são conectados de forma matricial para economizar pinos do microcontrolador. A conexão é feita por meio de três linhas verticais e quatro linhas horizontais. As linhas horizontais são conectadas aos pinos da porta D (RD3 à RD0) configuradas como entrada no microcontrolador. Existem resistores de *pull-up* de $10k\Omega$ proporcionando nível lógico válido (“1”) quando nenhum botão estiver sendo pressionado. As linhas verticais são conectadas à 3 pinos da porta B (RB2 à RB0) configurados como saída, de tal forma que a cada momento existirá um nível digital “0” ou “1” nesses pinos definido pelo software. Na implementação adotada, as saídas possuem nível lógico padrão igual a “1”.

O funcionamento do circuito ocorre da seguinte maneira: quando um dos pinos de saída em `portB` é colocado em nível lógico “0”, ocorre a varredura desta coluna do teclado. Nesse momento caso algum usuário pressione um botão pertencente à essa coluna, o sistema reconhecerá essa ação e será capaz de identificar qual é o botão pressionado. O *software* é implementado de tal maneira que a varredura de cada uma das três colunas seja executada em *loop*. Esse passo ocorre de maneira muita rápida de tal forma que a identificação de botão pressionado ocorra com sucesso.

A Figura 34 exibe um esquemático demonstrando o funcionamento do teclado matricial. Nela é mostrado o momento em que a varredura (RB1) ocorre no mesmo momento em que uma tecla (número 5) dessa linha vertical está pressionada, assim, o pino RD2 reconhece o pressionamento dessa tecla.

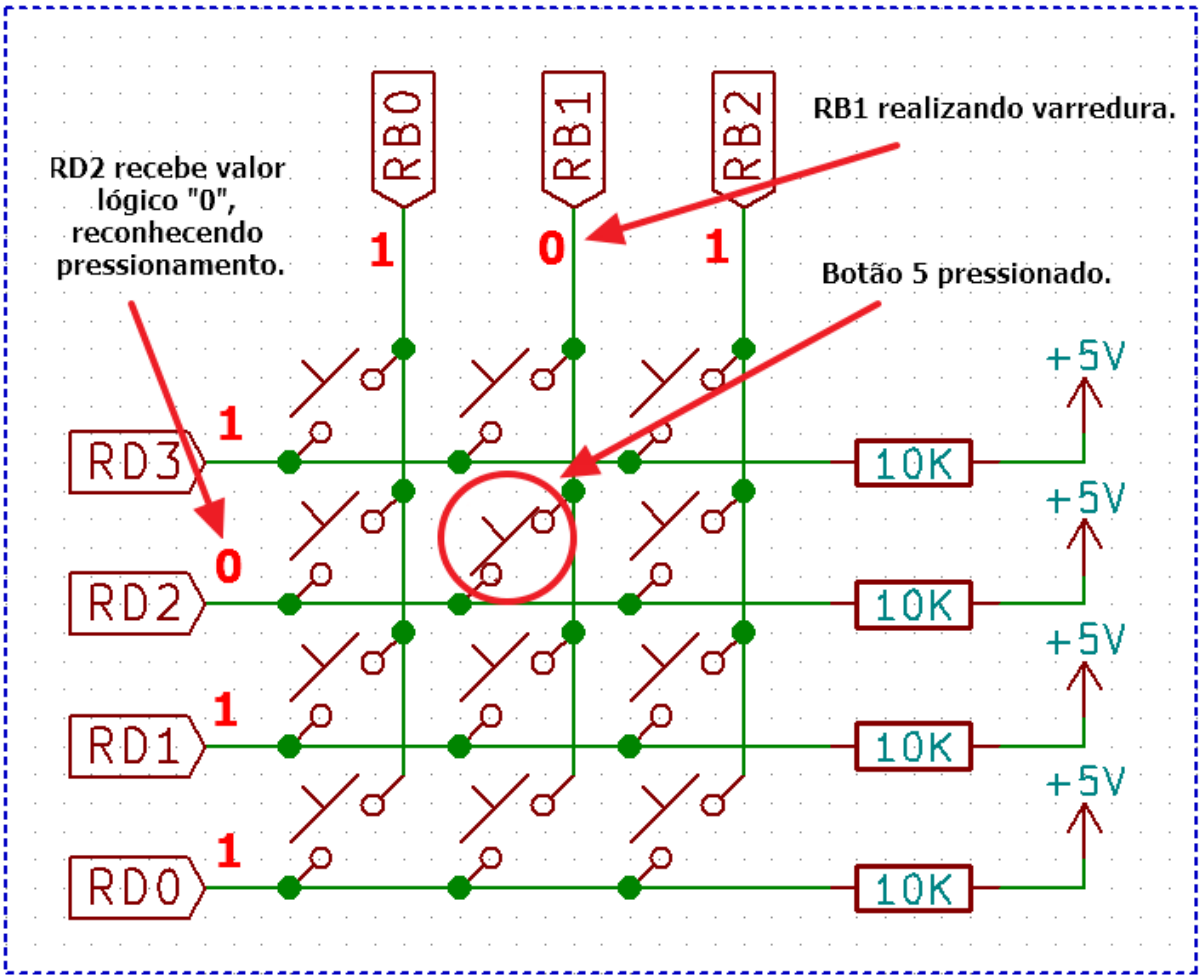

Figura 34: Funcionamento teclado matricial.

4.1 Código do Projeto

```
/*************************************************************************
**                            Exemplo 04                                **
**                                                                      **
**     Exemplo para utilizacao do LCD e do teclado matricial            **
**     presente no kit PIC 18F452. De acordo com a tecla ativa          **
**     no teclado, o valor correspondente e' escrito no LCD.            **
**                                                                      **
** Arquivo: LCD.c                                                       **
** Compilador: MikroC PRO PIC v.6.4.0                                   **
**                                                                      **
** UFLA - Lavras/MG - 28/06/2017                                        **
*************************************************************************/

// Conexoes LCD do kit PICGenios com PIC18F452
sbit LCD_RS at RE2_bit;
sbit LCD_EN at RE1_bit;
sbit LCD_D4 at RD4_bit;
sbit LCD_D5 at RD5_bit;
sbit LCD_D6 at RD6_bit;
sbit LCD_D7 at RD7_bit;

sbit LCD_RS_Direction at TRISE2_bit;
sbit LCD_EN_Direction at TRISE1_bit;
sbit LCD_D4_Direction at TRISD4_bit;
sbit LCD_D5_Direction at TRISD5_bit;
sbit LCD_D6_Direction at TRISD6_bit;
sbit LCD_D7_Direction at TRISD7_bit;
// Fim das conexoes do LCD

// Conexoes do teclado
// Saidas do PIC:
//    RB0 - primeira coluna
//    RB1 - segunda coluna
//    RB2 - terceira coluna
// Entradas do PIC:
//    RD3 - primeira linha
//    RD2 - segunda linha
//    RD1 - terceira linha
//    RD0 - quarta linha

```

```
41 void LCD(char texto[16])
42 {
43   Lcd_Cmd(_LCD_CLEAR);     // Limpa o Display
44   Lcd_Out(1,4,texto);
45 }
46
47 void main()
48 {
49   ADCON0 = 0X00;  // Desabilita o conversor A/D
50   ADCON1  = 0x06; // Configura pinos da porta A para digital
51
52   trisd = 0x0F;   // Configura os pinos RD0,RD1,RD2,RD3 da PORTD
53                   // como entradas (teclado), e RD4,RD5,RD6,RD7
54                   // como saida (LCD)
55
56   portb = 0xFF;
57   trisb = 0xF8;   // Configura pinos RB0,RB1,RB2 da PORTB como
58                   // saida (teclado) e o restante como entrada
59
60   Lcd_Init();                 // Inicializa o Display
61   Lcd_Cmd(_LCD_CLEAR);        // Limpa o Display
62   Lcd_Cmd(_LCD_CURSOR_OFF);   // Desabilita o cursor
63
64   do
65   {
66     portb.rb0 = 0;      // Habilita primeira coluna  do teclado
67
68     if (portd.rd0 == 0)                LCD("<---");
69     else if (portd.rd1 == 0)           LCD("7");
70          else if (portd.rd2 == 0)      LCD("4");
71               else if (portd.rd3 == 0) LCD("1");
72
73     portb.rb0 = 1;      // Desabilita primeira coluna do teclado
74     //-----------------------------------------------------
75
76     portb.rb1 = 0;      // Habilita segunda coluna do teclado
77
78     if (portd.rd0 == 0)                LCD("0");
79     else if (portd.rd1 == 0)           LCD("8");
80          else if (portd.rd2 == 0)      LCD("5");
81               else if (portd.rd3 == 0) LCD("2");
```

```
82
83      portb.rb1 = 1;      // Desabilita segunda coluna do teclado
84
85      //-----------------------------------------------------
86
87      portb.rb2 = 0;      // Habilita terceira coluna do teclado
88
89      if (portd.rd0 == 0)                 LCD("--->");
90      else if (portd.rd1 == 0)            LCD("9");
91           else if (portd.rd2 == 0)       LCD("6");
92                else if (portd.rd3 == 0)  LCD("3");
93
94      portb.rb2 = 1;      // Desabilita terceira coluna do teclado
95
96    } while(1);
97
98  } // fim do programa
```

- **OBS 1:** Para projetos que utilizem o módulo LCD do kit, deve-se, ao criar o projeto, alterar a opção ‘‘`Oscillator Selection`’’ para a configuração ‘‘`HS oscillator`’’. Esse passo é mostrado na Figura 35

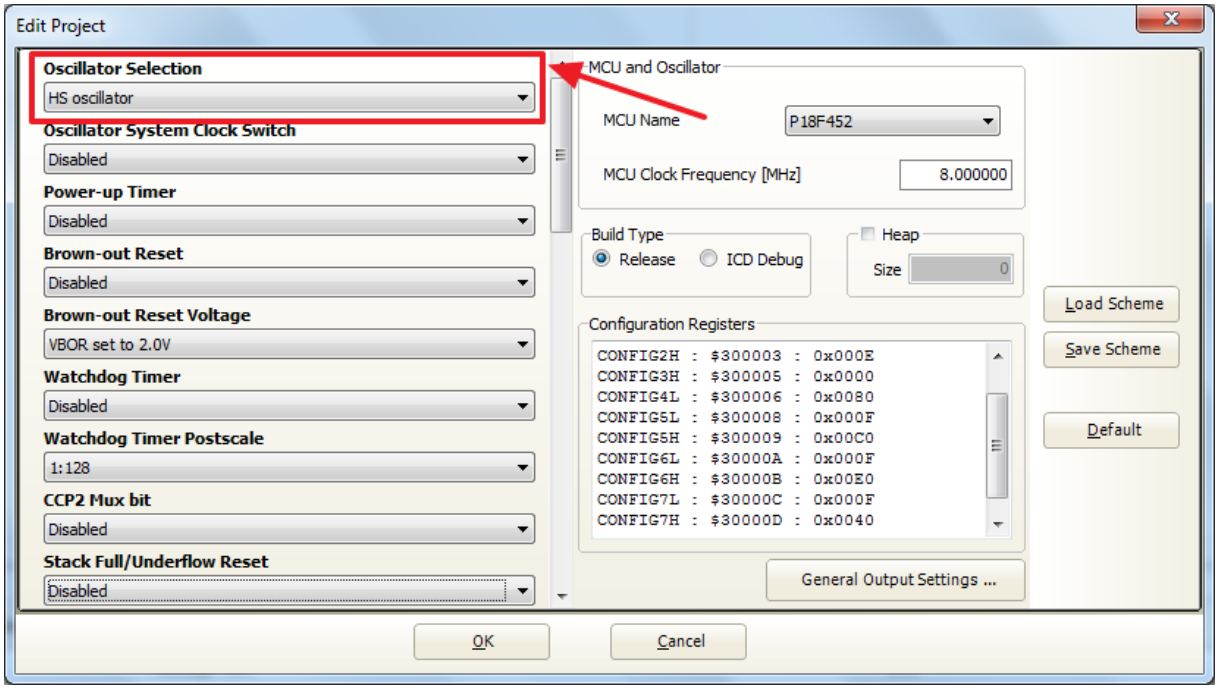

Figura 35: Alteração no modo oscilador.

- **OBS 2:** Para execução do exemplo desta aula e do exercício prático, atente-se para que as chaves DIP do kit estejam configuradas da seguinte maneira: ligue a chave ‘‘`GLCD-LCD`’’ (posição “ON”) para que o *display* LCD possa funcionar corretamente. A Figura 36 exibe a configuração das chaves.

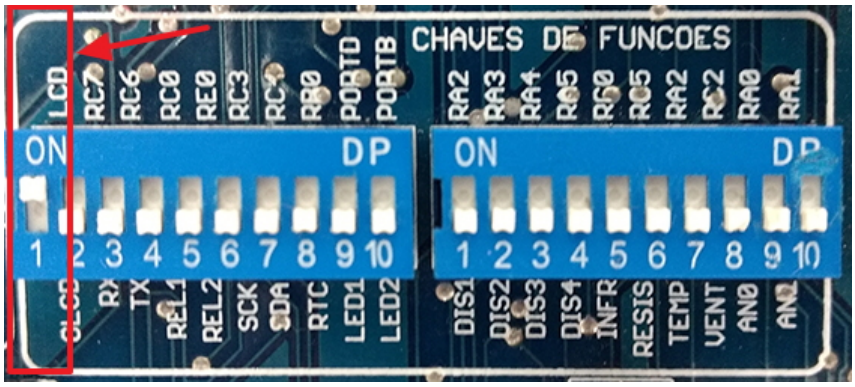

Figura 36: Configuração das chaves DIP.

4.2 Atividade Prática

Utilizando como base o código desenvolvido nessa aula, crie um novo projeto seguindo os passos dados na Aula 01. Por fim, teste a implementação utilizando o kit PICGenios.

Nesta tarefa, implemente um identificador de senha utilizando o teclado e LCD com as seguintes funções:

- Solicitar que o usuário digite uma senha de 4 números.
- Se a senha digitada for a correta, alguns LEDs piscam durante um intervalo de tempo.
- Se a senha digitada for errada, toca-se uma buzina durante um intervalo de tempo.
- O programa repete do início.

Observações:

- Exercício individual, mas pode ser feito em grupo. Cada aluno posta o seu exercício.
- Postar um arquivo ".pdf" com o programa fonte em linguagem C contendo os devidos comentários na sala virtual da disciplina, até a data combinada. Não compactar o arquivo postado.
- Apresentar o exercício funcionando ao professor durante a aula prática. Isto é essencial para contabilizar a avaliação do exercício.
- Não será permitido a postagem do exercício depois do prazo combinado.

5 Aula 05

Na quinta aula prática do curso será mostrado como operar os *displays* de sete segmentos instalados no kit PICGenios. O esquemático mostrado na Figura 37 descreve os componentes utilizados no projeto que inclui 4 displays de 7 segmentos de catodo comum.

O acionamento dos displays é baseado no processo de multiplexação. Esta técnica permite que por uma única porta paralela do microcontrolador seja possível acionar vários displays, economizando pinos do microcontrolador.

O circuito opera da seguinte maneira:

1. O software do microcontrolador coloca nos 8 bits da porta D o código do número que se deseja visualizar num dos displays.
2. Todos os displays de 7 segmentos receberão o mesmo código binário ao mesmo tempo, pois estão em paralelo. Entretanto, nenhum display acenderá pois os respectivos catodos comuns não estão aterrados (transistores em corte).
3. O software do microcontrolador aciona apenas um display colocando nível lógico 1 em apenas um dos pinos da porta A (pinos RA2 a RA5), durante um curto espaço de tempo (5ms).
4. Apenas o display que tem seu catodo comum aterrado pelo transistor irá acender, mostrando o número desejado para visualizar. Os outros displays ficarão apagados.
5. Em seguida, o software desliga o display aceso e repete o processo com outro display. Fazendo isto repetidamente com todos os displays em alta frequência, o usuário não perceberá que os displays estão piscando, mas enchergará os 4 displays acesos ao "mesmo tempo".

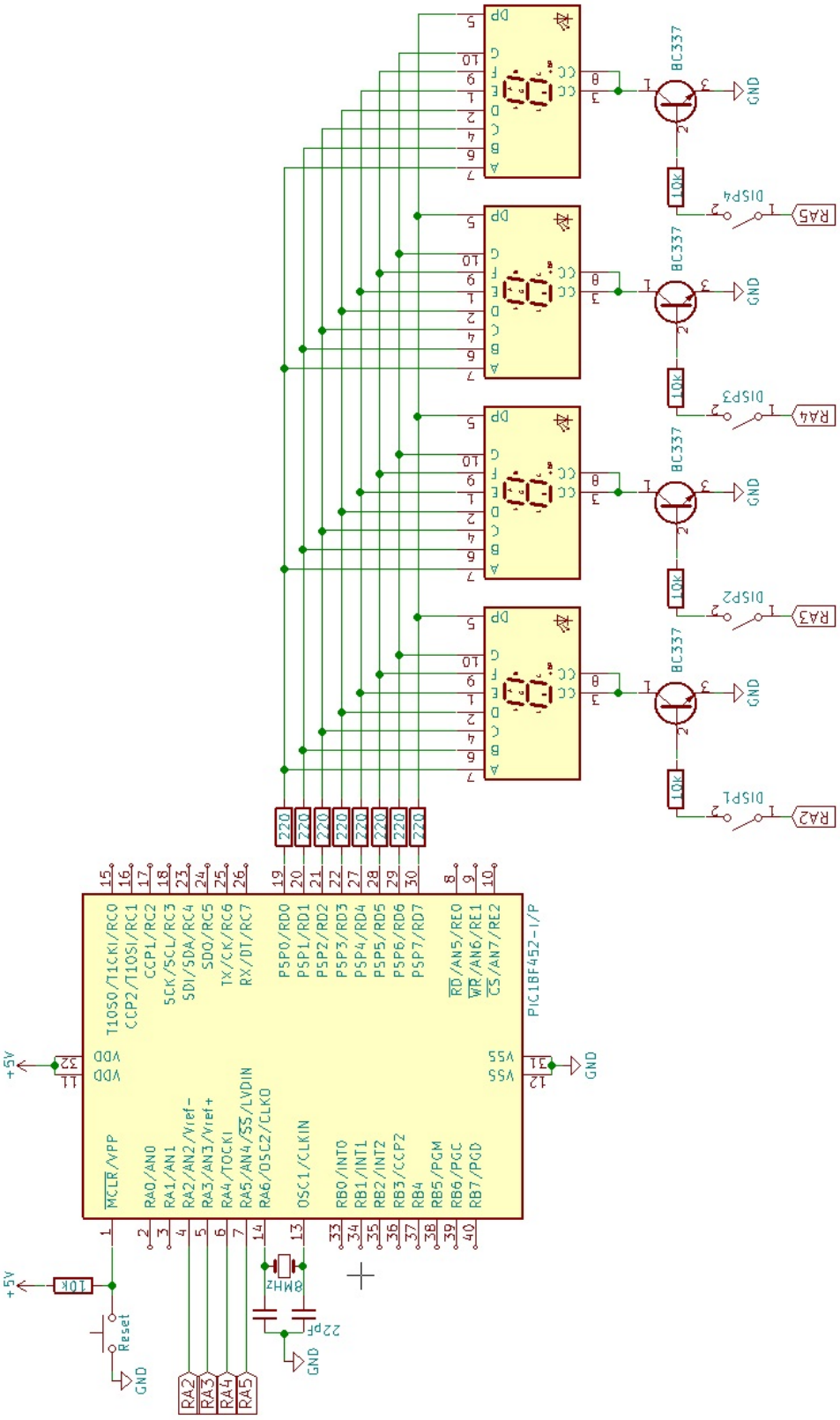

Figura 37: Esquemático do projeto.

5.1 Código do Projeto

Um exemplo de programa para acionamento dos displays multiplexados é apresentado a seguir:

```
/*****************************************************************
**                          Exemplo 05                          **
**                                                              **
**       Exemplo para utilizacao do display de 7 segmentos      **
**       presente no kit PIC 18F452. Cada um dos 4 displays     **
**       sao acessos, um de cada, vez, exibindo seu indice      **
**       Ex: display 1 = 1, display 2 = 2, etc...               **
**                                                              **
** Arquivo: display7seg.c                                       **
** Compilador: MikroC PRO PIC v.6.4.0                           **
**                                                              **
** Obs: Ativar dips switchs: DISP1 a DISP4                      **
**                                                              **
** UFLA - Lavras/MG - 12/07/2017                                **
*****************************************************************/

// Define o tempo de acendimento do display em ms.
#define tempo 5

// Converte valor numerico decimal para codigo 7 segmentos
unsigned short mask(unsigned short num)
{
    switch (num)
    {
        case 0 : return 0x3F;
        case 1 : return 0x06;
        case 2 : return 0x5B;
        case 3 : return 0x4F;
        case 4 : return 0x66;
        case 5 : return 0x6D;
        case 6 : return 0x7D;
        case 7 : return 0x07;
        case 8 : return 0x7F;
        case 9 : return 0x6F;
    }
}

```

```
38 void main(void)
39 {
40     ADCON0 = 0X00;
41     ADCON1 = 0X06;       // desabilita conversor A/D.
42     INTCON = 0;          // desabilita interrupcoes.
43     TRISA  = 0;          // configura portA como saida.
44     PORTA  = 0;
45     TRISD  = 0;          // configura portD como saida.
46     PORTD  = 0;
47     while(1)             // inicio do loop infinito.
48     {
49         // Escreve valor 1 no display 1 em codigo 7 segmentos.
50         PORTD = mask(1);
51         porta.f2 = 1;      // Ativa display 1.
52         Delay_ms(tempo);
53         porta.f2 = 0;      // Desativa display 1.
54
55         // Escreve valor 2 no display 2 em codigo 7 segmentos.
56         PORTD = mask(2);
57         porta.f3 = 1;      // Ativa display 2.
58         Delay_ms(tempo);
59         porta.f3 = 0;      // Desativa display 2.
60
61         // Escreve valor 3 no display 3 em codigo 7 segmentos
62         PORTD = mask(3);
63         porta.f4 = 1;      // Ativa display 3.
64         Delay_ms(tempo);
65         porta.f4 = 0;      // Desativa display 3.
66
67         // Escreve valor 4 no display 4 em codigo 7 segmentos.
68         PORTD = mask(4);
69         porta.f5 = 1;      // Ativa display 4.
70         Delay_ms(tempo);
71         porta.f5 = 0;      // desativa display 4.
72     } // Fim do loop infinito
73 }   // Fim do programa principal.
```

Na execução desse código, o kit PICGenios deverá mostrar os dígitos de 1 a 4 nos respectivos *displays*.

5.2 Atividade Prática

Utilizando como base o código desenvolvido nessa aula, crie um novo projeto seguindo os passos dados na Aula 01. Por fim, teste a implementação utilizando o kit PICGenios.

Nesta tarefa, implemente um temporizador digital utilizando os *displays* de 7 segmentos com as seguintes funções:

- O usuário escolhe um período de tempo em segundos utilizando 2 *push-bottons* e mostra no *displays* de 7 segmentos;
- Inicia a contagem regressiva por meio de outro *push-botton* e mostra nos *displays* de 7 segmentos o tempo restante;
- Ao final da contagem, um alarme sonoro toca indicando o final do tempo;
- Para reiniciar, deve-se apertar o botão de reset.

Observações:

- Exercício individual, mas pode ser feito em grupo. Cada aluno posta o seu exercício.
- Postar um arquivo ".pdf" com o programa fonte em linguagem C contendo os devidos comentários na sala virtual da disciplina, até a data combinada. Não compactar o arquivo postado.
- Apresentar o exercício funcionando ao professor durante a aula prática. Isto é essencial para contabilizar a avaliação do exercício.
- Não será permitido a postagem do exercício depois do prazo combinado.

6 Aula 06

Na sexta aula prática do curso será mostrado o funcionamento do encoder (sensor ótico) instalado no kit PICGenios. O esquemático eletrônico mostrado na Figura 38 descreve os componentes utilizados no projeto.

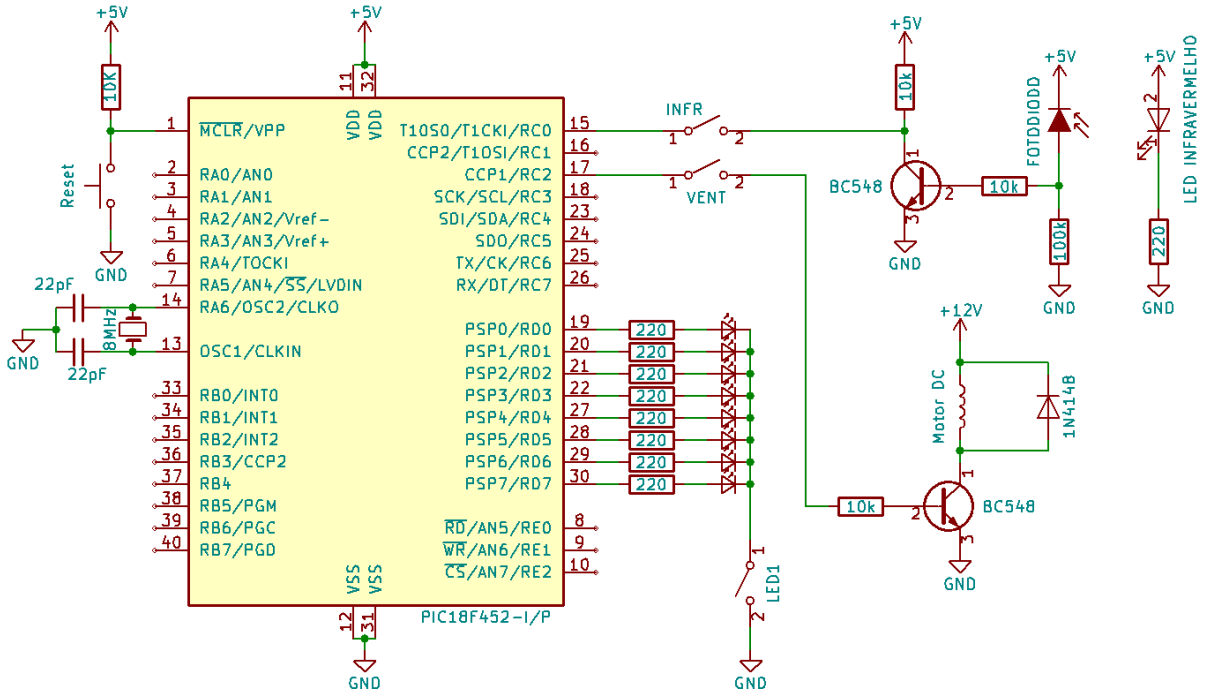

Figura 38: Esquemático do projeto.

O LED infravermelho está alimentado constantemente, fornecendo luz infravermelha em direção ao sensor fotodiodo. Entre o LED e o sensor estão as pás da ventoinha (motor DC). Quando a pá da ventoinha interrompe a luz que chega no fotodiodo, este não conduz, gerando tensão zero na base do transistor (BC548). Não havendo corrente na base do transistor, este corta fazendo com que a tensão de coletor (em relação ao terra) seja a tensão de alimentação do circuito (5V). O 5V aplicado no pino RC0 (configurado como entrada digital) é interpretado como nível lógico 1.

Quando a pá da ventoinha não interrompe a luz infravermelha que chega no fotodiodo, este conduz inversamente, gerando uma tensão nos terminais do resistor de $100k\Omega$. Esta tensão injeta uma corrente de base no transistor fazendo ele conduzir e saturar. O transistor saturado impõe uma tensão de quase 0V no pino RC0 do microcontrolador (configurado como entrada digital), que é interpretado como nível lógico zero.

O software que é executado no microcontrolador interpreta o sinal de entrada do pino RC0, identificando que o rotor do motor se move pela mudança do estado do

pino RC0. A partir disto é possível contar o número de voltas do rotor, calcular a sua velocidade e aceleração.

A ventoinha do kit é acionada pelo pino RC2 do microcontrolador (configurado como saída digital), que controla o transistor BC548 que alimenta a ventoinha. Nível lógico um no pino RC2 faz o transistor conduzir, ligando a ventoinha. Nível lógico zero no pino RC2 faz o transistor cortar, desligando a ventoinha.

6.1 Código do Projeto

```
/*****************************************************************
**                          Exemplo 06                          **
**       Exemplo para utilizacao do sensor otico                **
**       presente no kit PICGenios PIC18F452.                   **
**                                                              **
** Arquivo:    sensor_infra.c                                   **
** Compilador: MikroC PRO PIC v.6.4.0                           **
** Data:       julho 2017                                       **
** Obs: Ativar dips switchs INFR e VENT                         **
*****************************************************************/

void main(){

   char contador = 0;
   ADCON0  = 0x00;     // Configura todos pinos para digital e
   ADCON1  = 0x06;     // desabilita o conversor A/D
   trisd = 0; // configura porta D como saida
   portd = 0; // apaga todos os LEDs da porta D
   trisc = 1; // Entrada: RC0 ; os outros pinos sao saidas
   portc.rc2 = 1;  // liga a ventoinha

   while(1) {
        while (portc.rc0 == 0)
          delay_ms(1);
        while (portc.rc0 == 1)
          delay_ms(1);
        contador = contador + 1;
        portd = contador;
   }
} // fim do programa
```

6.2 Atividade Prática

Utilizando como base o código desenvolvido nessa aula, crie um novo projeto seguindo os passos dados na Aula 01. Por fim, teste a implementação utilizando o kit PICGenios.

Nesta tarefa, implemente um contador de voltas da ventoinha contida no kit PICGenios e mostre no LCD. Um push-botton será responsável por zerar a contagem. O motor DC (ventoinha) fica ligado o tempo todo.

Observações:

- Exercício individual, mas pode ser feito em grupo. Cada aluno posta o seu exercício.
- Postar um arquivo ".pdf" com o programa fonte em linguagem C contendo os devidos comentários na sala virtual da disciplina, até a data combinada. Não compactar o arquivo postado.
- Apresentar o exercício funcionando ao professor durante a aula prática. Isto é essencial para contabilizar a avaliação do exercício.
- Não será permitido a postagem do exercício depois do prazo combinado.

7 Aula 07

Na sétima aula prática do curso será mostrado como operar os relés do kit PICGenios utilizando o microcontrolador. O esquemático mostrado na Figura 39 descreve os componentes utilizados no projeto.

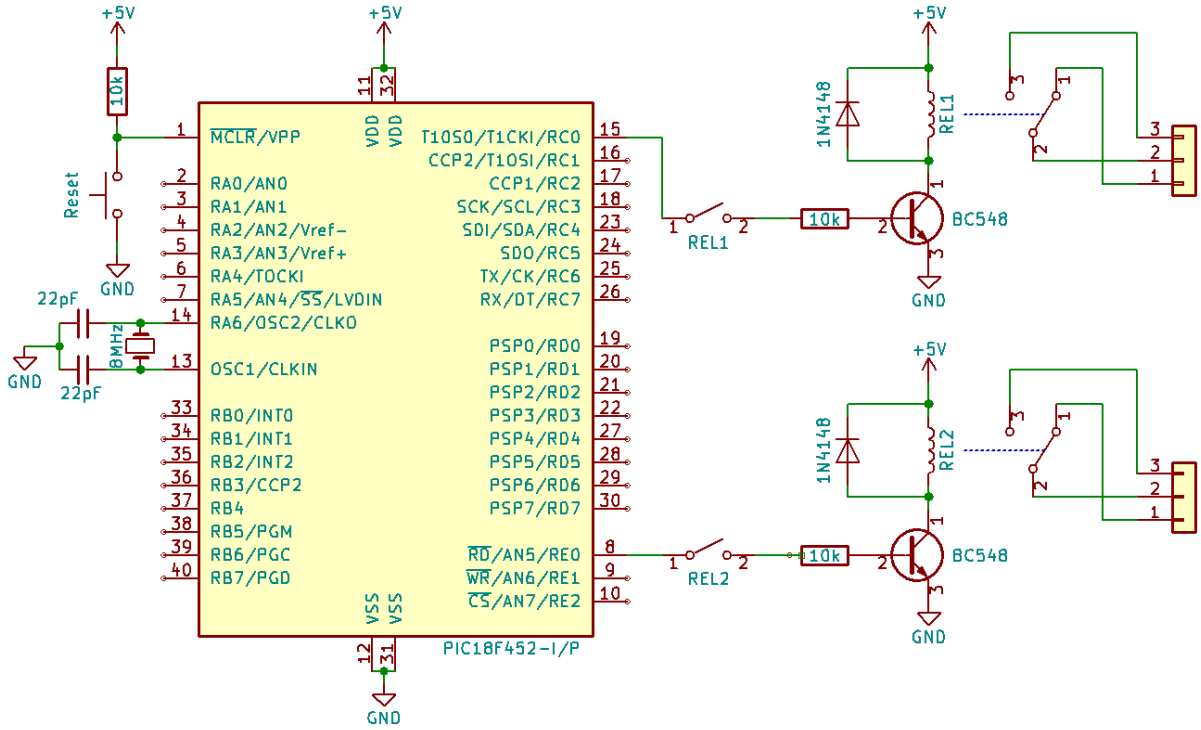

Figura 39: Esquemático do projeto.

Cada bobina (eletroíma) dos relés utilizados no kit necessita de aproximadamente 50mA para energizá-lo. Como cada pino (configurado como saída digital) do microcontrolador da família PIC pode fornecer somente 10mA de corrente, então é necessário um transistor para acionamento do relé.

Nível lógico 1 (+5V) na saída do pino RC0 (configurado como saída digital) impõe uma corrente de base no transistor correspondente que é suficiente para saturá-lo, energizando a bobina do relé e chaveando seus contatos mecânicos. Uma carga (lâmpada) ligada na saída do relé será energizada por uma fonte de tensão alternada externa (127Vrms - 60Hz) ligada em série.

Nível lógico 0 (0V) na saída do pino RC0 faz com que o transistor corte devido a falta de corrente de base. Com o transistor cortado, a bobina do relé fica desenergizada fazendo com que sua chave mecânica volte a posição normal, desligando a carga.

7.1 Código do Projeto

No código abaixo é demonstrado um exemplo de programação para o projeto em questão. O programa simplesmente ativa e desativa os dois relés do kit PICGenios em intervalos de 1 segundo. Uma lâmpada (40W - 127V), conectada em série com os contatos mecânicos de um dos relés e com uma fonte alternada, irá piscar conforme o programa é executado.

```
/*****************************************************************
**                          Exemplo 07                          **
**       Exemplo para utilizacao dos reles no kit PIC Genios.   **
**                                                              **
** Arquivo:    reles.c                                          **
** Compilador: MikroC PRO PIC v.6.4.0                           **
** Data:       12/2017                                          **
** Obs: Ativar dip-switchs: REL1 e REL2                         **
*****************************************************************/

void main( ) {

  adcon0 = 0x00;   // desliga CAD
  adcon1 = 0x06;   // configura todos os pinos como E/S
  trisc.rc0 = 0;   // configura pino RC0 como saida - rele 1
  trise.re0 = 0;   // configura pino RE0 como saida - rele 2

  while(1){ // loop infinito

    porte.re0 = 1;  // aciona rele 2
    delay_ms(1000); // atraso de 1000 milisegundos (1 segundo)
    portc.rc0 = 1;  // aciona rele 1
    delay_ms(1000); // atraso de 1000 milisegundos (1 segundo)
    porte.re0 = 0;  // desliga rele 2
    delay_ms(1000); // atraso de 1000 milisegundos (1 segundo)
    portc.rc0 = 0;  // desliga rele 1
    delay_ms(1000); // atraso de 1000 milisegundos (1 segundo)

  }
} // fim
```

8 Aula 08

Na oitava aula prática do curso será mostrado como utilizar o conversor analógico para digital (CAD) do microcontrolador. Para isto, serão utilizados dois projetos.

8.1 Aula 08 - parte A

O esquemático mostrado na Figura 40 descreve os componentes utilizados no primeiro projeto.

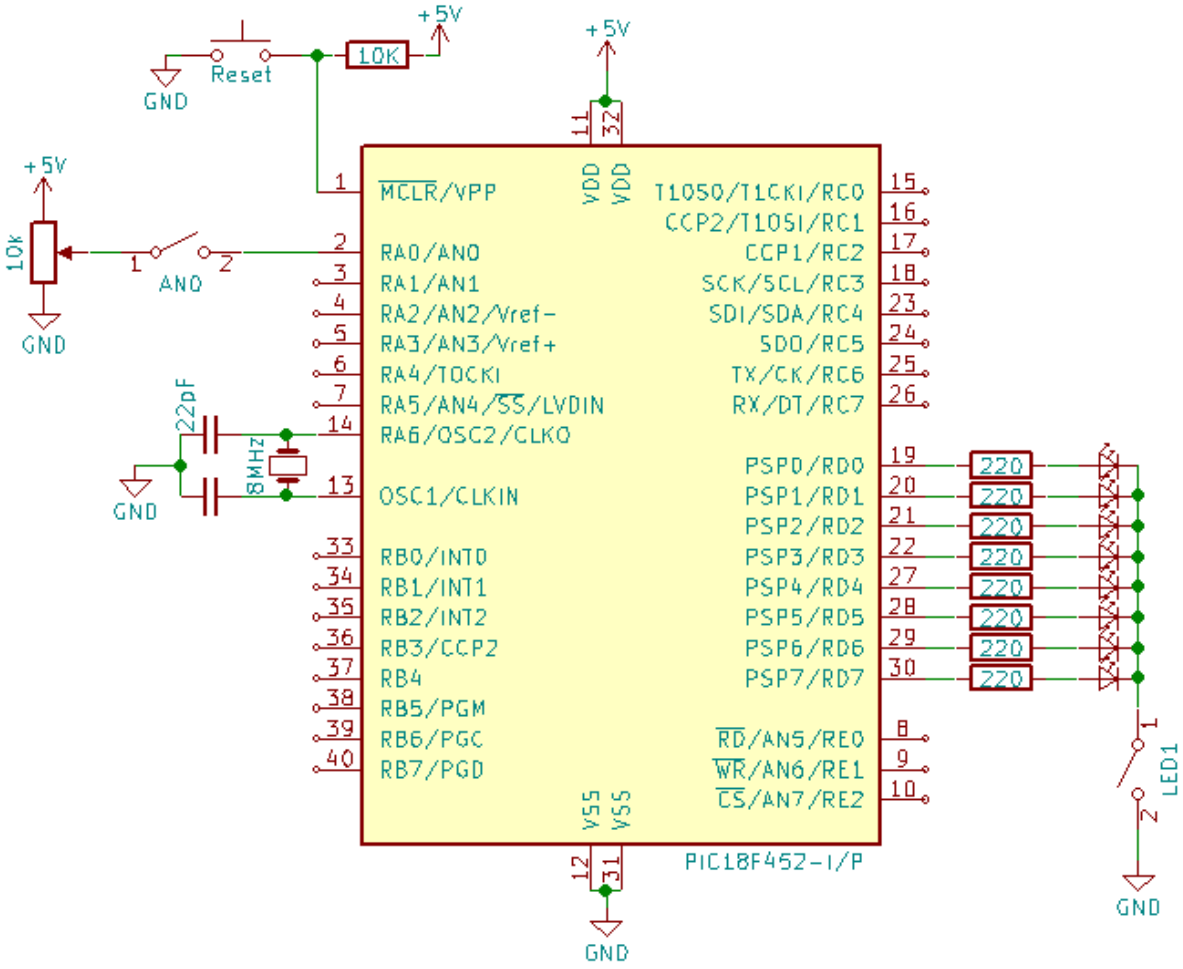

Figura 40: Esquemático do primeiro projeto.

Neste projeto, na entrada do canal AN0 do conversor A/D do microcontrolador PIC é colocado uma tensão por meio do *dip-switch* AN0. Assim, a tensão selecionada no trimpot AN0 é convertida para um valor digital de 10 bits pelo conversor A/D.

O grupo de 8 LEDs (LED1) conectado na porta D é utilizado para apresentar

o resultado da conversão A/D eliminando 2 bits menos significativos.

8.2 Código do Primeiro Projeto

O código a seguir demonstra um exemplo de utilização do conversor A/D presente no microcontrolador.

```
1  //*************************************************************
2  // Placa: KIT PICGENIOS com microcontrolador 18F452
3  // Objetivo: Ler o canal AN0 (RA0 - potenciometro) e mostrar
4  // nos LEDs
5  // Compilador: MikroC PRO v6.4
6  // Data:       agosto 2017
7  // Obs.: fechar os dip-switchs LED1 e AN0
8  //*************************************************************
9
10 //**************** definicao de variaveis ******************
11 unsigned int valor_AD = 0;     //define variavel
12 //*************************************************************
13
14 //************** programa principal ************************
15 void main() {
16
17   ADCON0 = 0X00;    // Desliga CAD
18   // todos os pinos do port A (AN0-AN4) e port E (AN5-AN7)
19   // como entrada/saida de uso geral
20   ADCON1 = 0x06;
21
22   trisd = 0x00;     //define portd como saida - LEDs
23   trisa = 0x01;     // define RA0/AN0 como entrada, outros pinos
        como saida
24
25   /*reconfigura e define pinos de A/D - ADCON1
26   B7 - 1: justificado a direita
27        0: justificado a esquerda
28   B6 - 0: fonte de clock oscilador RC interno do ADC
29   B5 - NC
30   B4 - NC
31   B3:B0 = 0b1110 = apenas AN0, Vref+ = VDD, Vref- = GND  */
32   ADCON1 = 0b10001110;
33
```

```
34    /*reconfigura A/D - ADCON0
35    B7:B6 - 0b11: fonte de clock oscilador RC interno do ADC
36    B5:B3 - 0b000: canal 0 - AN0
37    B2 - status do CAD
38    B1 - NC
39    B0 - 1: ADC ligado
40         0: ADC desligado                                    */
41    ADCON0 = 0b11000001;
42
43    do
44    {
45       valor_AD = adc_read(0)/4;   //le canal 0 e descarta 2 bits
46       portd = valor_AD;
47       delay_ms(100);              //delay de 100ms
48    } while(1);
49 }
```

8.3 Aula 08 - parte B

No segundo projeto, o sensor de temperatura LM35 é utilizado para medir a temperatura ambiente e apresentar no LCD. O esquemático mostrado na Figura 41 descreve os componentes utilizados neste projeto.

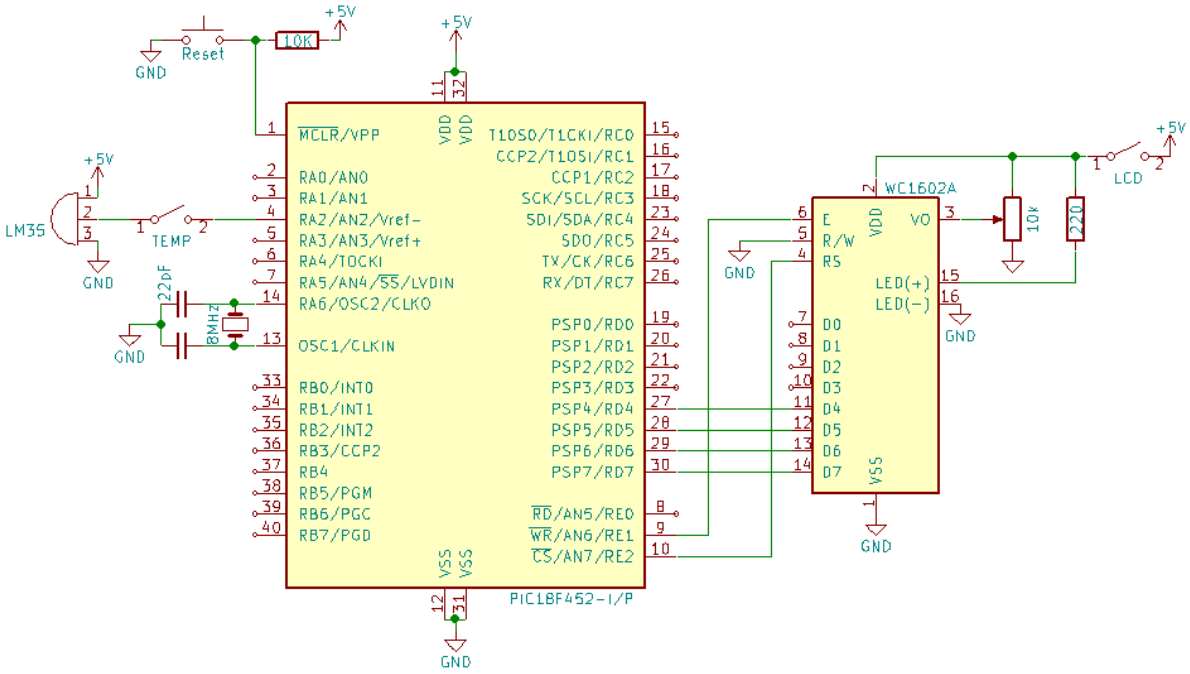

Figura 41: Esquemático do segundo projeto.

8.4 Código do Segundo Projeto

O código a seguir demonstra um exemplo de utilização do conversor A/D presente no microcontrolador para ler a temperatura do sensor LM35 e apresentar o valor no LCD.

```
/*********************************************************
Programa: Leitura CAD do sensor LM35
Placa: KIT PICGENIOS com microcontrolador 18F452
Objetivo: Ler o canal AN2 (RA2 - sensor temperatura LM35)
          e escrever no LCD.
Compilador: MikroC Pro 6.4
Data:       agosto 2017
Obs.: fechar os dip-switchs TEMP e LCD
*********************************************************/

// Conexoes LCD do kit PICGenios
sbit LCD_RS at RE2_bit;
sbit LCD_EN at RE1_bit;
sbit LCD_D4 at RD4_bit;
sbit LCD_D5 at RD5_bit;
sbit LCD_D6 at RD6_bit;
sbit LCD_D7 at RD7_bit;
sbit LCD_RS_Direction at TRISE2_bit;
sbit LCD_EN_Direction at TRISE1_bit;
sbit LCD_D4_Direction at TRISD4_bit;
sbit LCD_D5_Direction at TRISD5_bit;
sbit LCD_D6_Direction at TRISD6_bit;
sbit LCD_D7_Direction at TRISD7_bit;
// Fim das conexoes

//*************** definicao de variaveis ***************
char          texto[16];        //ponteiro texto
unsigned int  valor_AD;         //define variavel
float         temperatura;      // temperatura em graus Celsius
//******************************************************

//************** programa principal **********************
void main() {
   ADCON0 = 0x00;    // Desliga CAD
   ADCON1 = 0x06;
   // Todos os pinos de PortA (AN0-AN4) e PortE (AN5-AN7) como
   // entrada/saida de uso geral
```

```
38
39    trisd = 0x00;      // define PortE como saida - LCD
40    trise = 0x00;      // define PortE como saida - LCD
41    trisa = 0xFF;      // define RA2/AN2 como entrada, assim
42                       // como os demais pinos de PortA
43
44    // Inicializa o LCD
45    Lcd_Init();                        //Inicializa o Display
46    Delay_ms(100);
47    Lcd_Cmd(_LCD_CLEAR);               //limpa o Display
48    Delay_ms(100);
49    Lcd_Cmd(_LCD_CURSOR_OFF);          //Desabilita o cursor
50    Delay_ms(100);
51    Lcd_Out(1,1,"Temperatura[oC]:");
52    Delay_ms(100);
53
54    /*reconfigura e define pinos de A/D - ADCON1
55    B7 - 1: justificado a direita
56         0: justificado a esquerda
57    B6 - 0: fonte de clock oscilador RC interno do ADC
58    B5 - NC
59    B4 - NC
60    B3:B0 = 0b0010 = canais AN0 a AN4, Vref+ = VDD, Vref- = GND */
61    ADCON1 = 0b10000010;
62
63    /*reconfigura A/D - ADCON0
64    B7:B6 - 0b11: fonte de clock oscilador RC interno do ADC
65    B5:B3 - 0b010: canal 2 - AN2
66    B2 - status do CAD
67    B1 - NC
68    B0 - 1: ADC ligado
69         0: ADC desligado    */
70    ADCON0 = 0b11010001;
71
72    do
73    {
74       valor_AD =  adc_read(2);        //le canal 2 e salva
75
76       //converte para oC
77       temperatura = (float)valor_AD*((float)500/(float)1024);
78
```

```
79        //converte valor temperatura em string
80        FloatToStr(temperatura, texto);
81
82        //envia para o lcd o valor string da conversao A/D
83        Lcd_Out(2,1,texto);
84
85        Delay_ms(100);       //delay de 100ms
86    } while(1);
87 }
```

8.5 Atividade Prática

Utilizando como base o código desenvolvido nessa aula, crie um novo projeto seguindo os passos dados na Aula 01.

Nesta tarefa, implemente um programa em C para o kit PICGenios que utilize o sensor de temperatura LM35 e apresente no display LCD o valor da temperatura em graus Celsius, Fahrenheit ou Kelvin. A seleção da unidade de medida é feita por um *push-botton*. Dica: calcule a temperatura como valor inteiro para economizar memória do microcontrolador.

Por fim, teste a implementação utilizando o kit PICGenios.

Observações:

- Exercício individual, mas pode ser feito em grupo. Cada aluno posta o seu exercício.
- Postar um arquivo ".pdf" com o programa fonte em linguagem C contendo os devidos comentários na sala virtual da disciplina, até a data combinada. Não compactar o arquivo postado.
- Apresentar o exercício funcionando ao professor durante a aula prática. Isto é essencial para contabilizar a avaliação do exercício.
- Não será permitido a postagem do exercício depois do prazo combinado.

9 Aula 09

Na nona aula prática do curso será mostrado como utilizar o Timer1 do microcontrolador e também o gerador de PWM. O esquemático eletrônico mostrado na Figura 42 descreve os componentes utilizados no projeto.

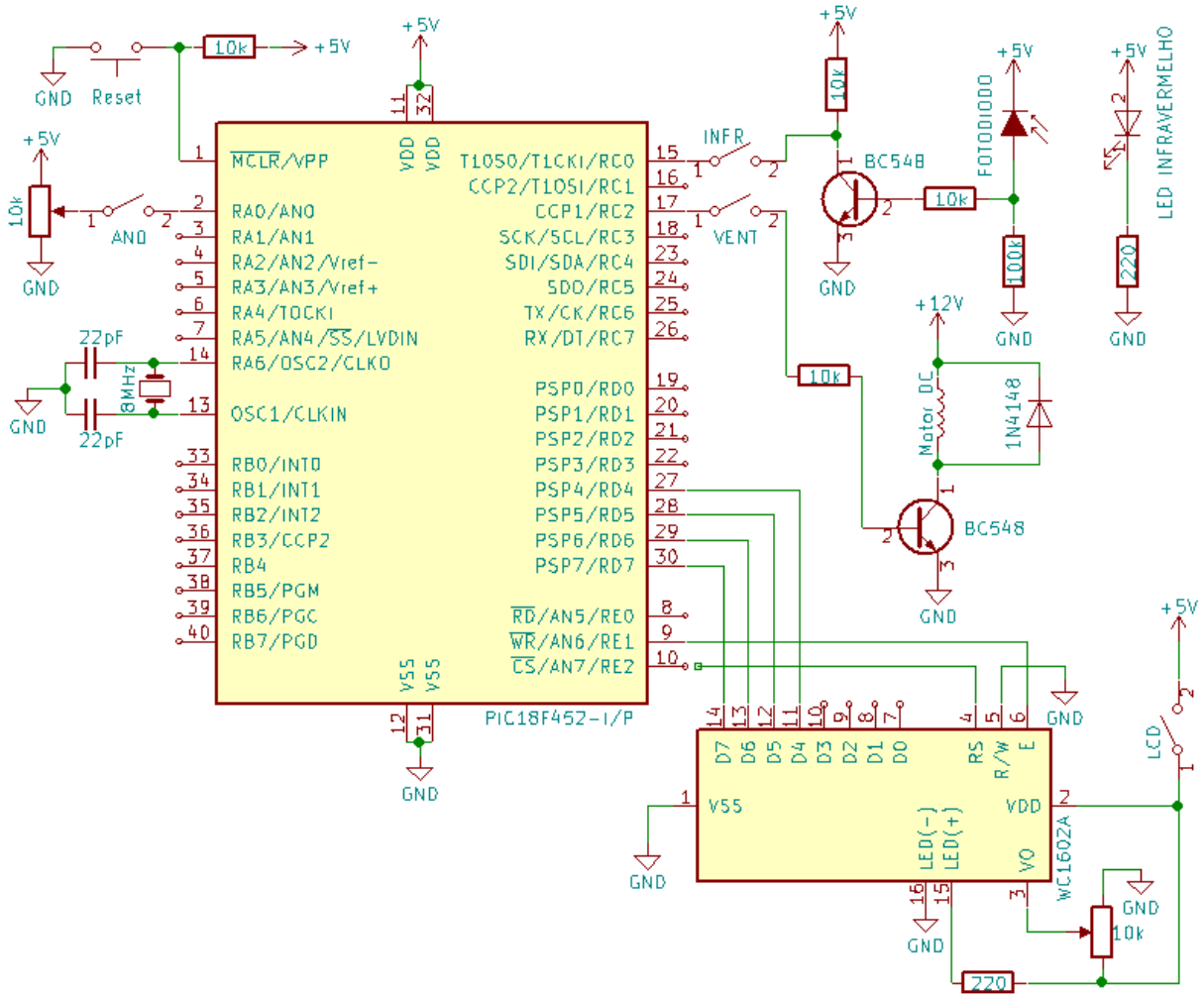

Figura 42: Esquemático do projeto.

Neste projeto, na entrada do canal AN0 do conversor A/D do microcontrolador PIC é colocado uma tensão por meio do *dip-switch* AN0. Assim, a tensão selecionada no trimpot AN0 pode ser convertida para um valor digital de 10 bits pelo conversor A/D.

Na entrada digital RC0 é conectado a saída do circuito sensor infravermelho pelo *dip-switch* INFR, que gera pulsos digitais a medida que a ventoinha (motor DC) funciona. Este pino estará funcionando como entrada de clock do Timer1, portanto a função deste pino será T1CKI.

O pino digital RC2, configurado como saída digital, controla o circuito de acio-

namento da ventoinha (motor DC), por meio do *dip-switch* VENT.

O LCD é conectado as portas D e E do microcontrolador e é alimentado pelo *dip-switch* LCD.

A seguir serão apresentados 2 exemplos de códigos de programa para utilização neste projeto.

9.1 Primeiro exemplo de Código

O código a seguir demonstra um exemplo de utilização do Timer1 presente no microcontrolador para contar o número de pulsos gerados pelo sensor infravermelho. O resultado da leitura do Timer1 é apresentado no LCD.

```
/************************************************************
                      Programa exemplo 09 - A
Este programa tem por objetivo contar o numero de pulsos recebidos
em T1CKI (gerados pela ventoinha) e enviar o valor para o LCD.
O pino RC0 e' a entrada de clock do contador TIMER1 (T1CKI).
No Kit PicGenios RC0 e' conectado ao sensor infravermelho.

Arquivo:    contador_lcd.c
Compilador: MikroC PRO PIC v.6.40
Placa:      Kit PICGenios v3 com microcontrolador PIC18F452
Data:       2017
Obs.: fechar os dip swtchs: LCD, INFR, VENT
************************************************************/

// Conexoes LCD do kit PICGenios com 18F452
sbit LCD_RS at RE2_bit;
sbit LCD_EN at RE1_bit;
sbit LCD_D4 at RD4_bit;
sbit LCD_D5 at RD5_bit;
sbit LCD_D6 at RD6_bit;
sbit LCD_D7 at RD7_bit;
sbit LCD_RS_Direction at TRISE2_bit;
sbit LCD_EN_Direction at TRISE1_bit;
sbit LCD_D4_Direction at TRISD4_bit;
sbit LCD_D5_Direction at TRISD5_bit;
sbit LCD_D6_Direction at TRISD6_bit;
sbit LCD_D7_Direction at TRISD7_bit;
// Fim das conexoes
```

```
29
30 // Declaracao das variaveis
31 char *texto[16];
32 unsigned int contador = 0;
33
34 // Programa principal
35 void main() {
36
37   ADCON0 = 0X00;     // Desliga CAD
38   // todos os pinos do port A (AN0-AN4) e port E
39   // (AN5-AN7) como entrada/saida de uso geral
40   ADCON1 = 0x06;
41
42   TRISD = 0;         // configura PortD como saida
43   TRISE = 0;         // configura PortE como saida
44
45   // Configura PortC como entrada no pino RC0
46   // (Sensor Infra-vermelho) e saida nos demais pinos
47   TRISC = 0x01;
48   portc.rc2 = 1;    // liga a ventoinha do Kit PicGenios
49
50   // Inicializa o LCD
51   Lcd_Init();                        //Inicializa o Display
52   Delay_ms(100);                     // delay de 100 milisegundos
53   Lcd_Cmd(_LCD_CLEAR);
54   Delay_ms(100);                     // delay de 100 milisegundos
55   Lcd_Cmd(_LCD_CURSOR_OFF);          // Desabilita o cursor
56   Delay_ms(100);                     // delay de 100 milisegundos
57
58   // Configura Timer 1:
59   // liga TIMER1, prescaler 1:1, modo 16bits, clock externo
60   T1CON = 0b10000011;     /*
61   Bit 7: modo de contagem
62              0 - 8 bits
63              1 - 16 bits
64   Bit 6: NC
65   Bit 5 e 4: Prescaler
66              11 - 1:8
67              10 - 1:4
68              01 - 1:2
69              00 - 1:1
```

```
  Bit 3: Oscilador
              1 - habilita
              0 - desabilita
  Bit 2: Sincronizacao de clock
              1 - nao sincroniza
              0 - sincroniza
  Bit 1: Fonte de clock
              1 - pino externo RC0
              0 - interno (Fosc/4)
  Bit 0: Timer1
              1 - habilita
              0 - desabilita
  */

  //carrega valor de contagem inicial do TIMER1
  TMR1H = 0x00;
  TMR1L = 0x00;

  Lcd_Out(1, 1,"Contagem: ");
  Delay_ms(100);                    // delay de 100 milisegundos

  do {
       contador = TMR1L;                // le o valor do timer1,
                                        // parte baixa
       contador = contador + TMR1H*256; // le o valor do timer1,
                                        // parte alta
       WordToStr(contador, texto);      //converte para string
       Lcd_Out(2, 1,texto);             // escreve variavel texto
       Delay_ms(100);
  } while(1);
} // Fim
```

9.2 Segundo exemplo de Código

O código a seguir demonstra um exemplo de utilização do gerador de PWM para acionar a ventoinha, controlando a sua velocidade por meio do potenciômetro conectado ao canal AN0 do CAD do microcontrolador.

```
/*************************************************************
                    Programa exemplo 09 - B
Este programa tem por objetivo variar a velocidade da ventoinha
a partir da variacao da tensao do trimpot conectado ao pino AN0.
Arquivo:    PWM.c
Compilador: MikroC PRO PIC v.6.40
Placa:      Kit PICGenios v3 com microcontrolador PIC18F452
Data:       2017
Obs.: fechar os dip swtchs: VENT, AN0
*************************************************************/
void main() {
  int p;

  /* reconfigura e define pinos de A/D - ADCON1
  B7 - 1: justificado a direita   0: justificado a esquerda
  B6 - 0: fonte de clock  oscilador RC interno do ADC
  B5:B4 - NC
  B3:B0 = 0b1110 = apenas AN0, Vref+ = VDD, Vref- = GND  */
  ADCON1 = 0b10001110;

  /* reconfigura A/D - ADCON0
  B7:B6 - 0b11: fonte de clock e' o oscilador RC interno do ADC
  B5:B3 - 0b000: canal 0 - AN0
  B2 - status do CAD
  B1 - NC
  B0 - 1: ADC ligado    0: ADC desligado                */
  ADCON0 = 0b11000001;

  trisa.ra0 = 1;     //programa pino RA0 como entrada - AN0
  PWM1_Init(2000);  //inicializa PWM, canal CCP1, com 2kHz
  PWM1_Set_Duty(0); //inicializa duty-cicle com 0
  PWM1_start();     //inicializa PWM

  while(1) {
     p = ADC_Read(0);  //le o canal AN0 e salva em P
     p = p>>2;         //converte o valor do AN0 em byte
     PWM1_Set_Duty(p); //atribui o valor de p ao dutycicle do PWM
     Delay_ms(100);    //aguarda delay de 100ms
  }
  PWM1_Stop();
}
```

10 Aula 10

Na décima aula prática do curso será mostrado como utilizar o circuito de interrupção do PIC. Serão utilizados dois exemplos. No primeiro exemplo, a interrupção será gerada pelo acionamento de um *push-botton*. No segundo exemplo, a interrupção será gerada pelo Timer0.

10.1 Primeiro exemplo de projeto

O esquemático mostrado na Figura 43 descreve os componentes utilizados no primeiro exemplo de projeto.

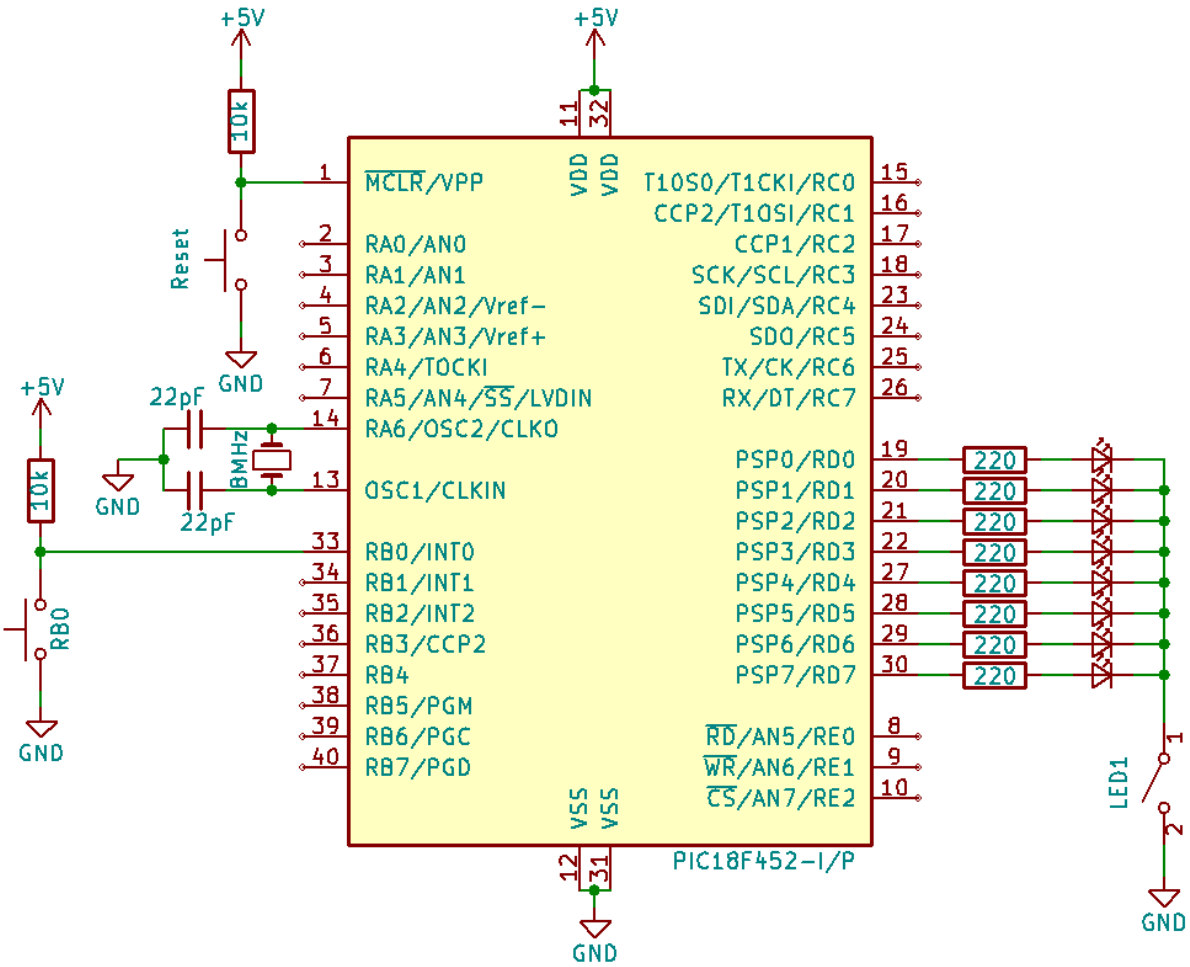

Figura 43: Esquemático do primeiro projeto.

Neste projeto, ao acionar o *push-botton* RB0, o pino de entrada INT0 do microcontrolador irá para nível lógico 0, ativando o pedido de interrupção do controlador de interrupção interno do microcontrolador. O software que está sendo executado é

o responsável por configurar o controlador de interrupção para aceitar o pedido vindo do pino INT0. Uma vez que o pedido de interrpção é aceito pela unidade de controle geral do microcontrolador, este para de executar o programa corrente e executa uma subrotina de interrupção previamente gravada na memória de programa. Após ser executada a subrotina de interrupção, o microcontrolador volta a continuar a executar o programa antes do pedido de interrupção.

A seguir é apresentado um exemplo de código.

10.2 Primeiro exemplo de Código

O código a seguir demonstra um exemplo de utilização da entrada do pino INT0 como fonte de pedido de interrupção. A função em linguagem C com o nome de *interrupt* é a responsável por conter as instruções que serão executadas para atender ao pedido de interrupção do periférico conectado no pino INT0. Neste exemplo, a função de interrupção simplesmente incrementa um contador em memória e escreve o seu valor na porta D onde o valor será apresentado de forma binária nos LEDs. Ao final da subrotina de interrupção é obrigatório ter o comando para limpar o *flag* do registrador de interrupção. O programa principal (função *main*) apenas configura o controlador de interrupção do microcontrolador por meio dos registradores apropriados e depois entra em *loop* infinito que não faz nada.

O efeito produzido pelo sistema será que: toda vez que se pressiona o *pushn-botton* RB0, o valor mostrado nos LEDs conectados na porta D incrementa de uma unidade. Repare que o *loop* infinito do programa principal (*main*) não contém nenhuma instrução para modificar o valor escrito na porta D, e não há nenhuma chamada à subrotina de interrupção por software.

```
1  /*********************************************************
2  Programa: Interrupcao por push-botton
3  Objetivo: incrementar uma unidade no portd a cada interrupcao
4  externa INT0, pino RB0/INT0 conectado ao pushbotton
5  Placa: Kit PICGenios v3 com microcontrolador PIC 18F452
6  Chaves de funcoes do kit: LED1
7  Compilador: MikroC PRO v6.4
8  Data: marco/2017
9  *********************************************************/
10
11 // declaracao das variaveis do programa
12 unsigned contador = 0;
13
```

```
14 //************** rotina de interrupcao ***********************
15 void interrupt(){  //vetor de interrupcao
16
17   contador++;
18   portd = contador;
19   delay_ms(500);
20   intcon.int0if = 0;  // apaga flag sinalizador de interrupcao
21                       // externa INT0 para que uma nova
22                       // interrupcao ocorra
23 }
24
25 //******************* programa principal *********************
26
27 void main() {  //funcao principal do programa
28
29   trisb.rb0 = 1; // configura pino RB0 como entrada:push-botton
30   trisd = 0;     // configura portd como saida - LEDs
31   portd = 0;     // resseta todos os pinos do portd:apaga LEDs
32
33   intcon2.rbpu = 0; // programa resistor de pull up interno do pic
34                     // habilitado em 0
35   intcon2.intedg0 = 0;  //aciona interrupcao por borda de descida
36
37   // Habilita interrupcao pelo pino externo INT0/RB0
38   intcon = 0b11010000;
39   //bit 7: habilita interrupcao geral
40   //bit 6: habilita interrupcao de perifericos
41   //bit 5: habilita interrupcao por overflow do TIMER0
42   //bit 4: habilita interrupcao externa INT0
43   //bit 3: habilita interrupcao por troca dos bits porta B
44   //bit 2: flag da interrupcao por overflow no TIMER0
45   //bit 1: flag da interrupcao EXTERNA INT0
46   //bit 0: flag da interrupcao por troca dos bits porta B
47
48   while(1);    // loop infinito aguardando a interrupcao
49
50 }  // fim
```

10.3 Segundo exemplo de projeto

O esquemático mostrado na Figura 44 descreve os componentes utilizados no segundo exemplo de projeto.

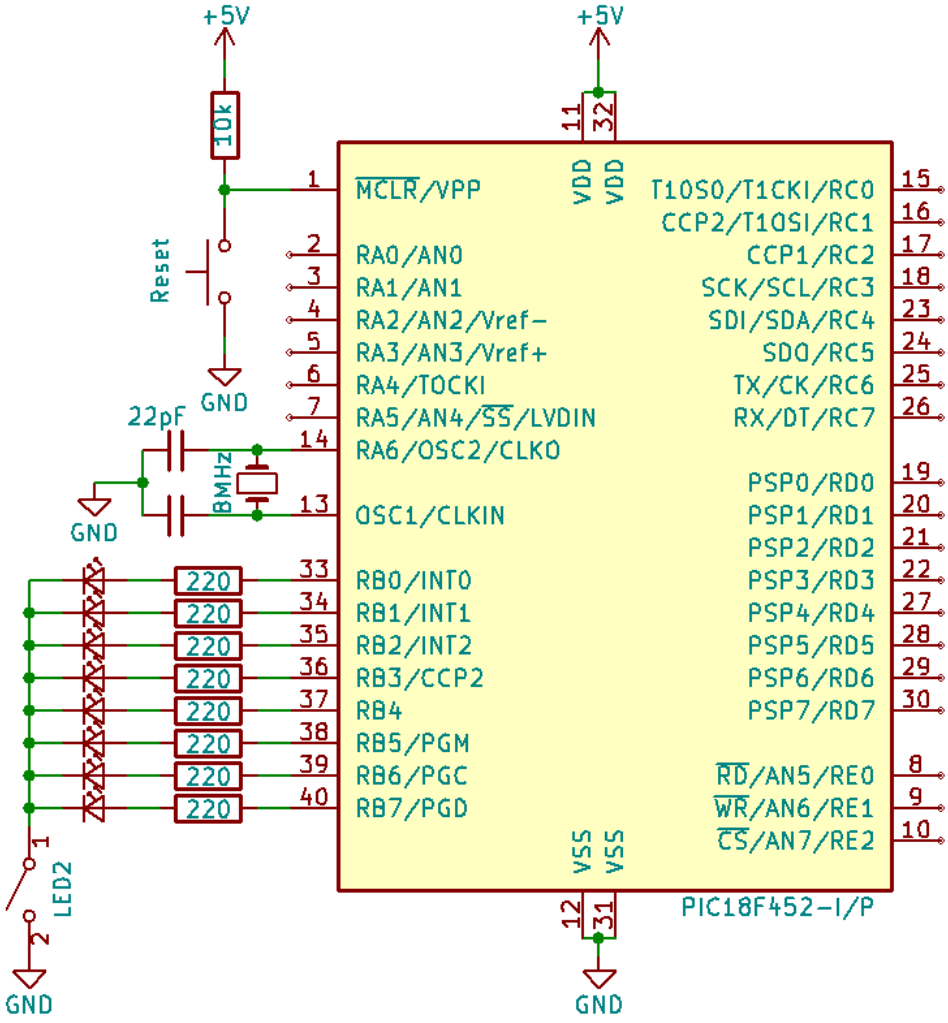

Figura 44: Esquemático do segundo projeto.

Neste projeto, o software gravado no microcontrolador configurará o contador interno Timer0 para contar uma quantidade de pulsos de clock do gerador a cristal. Ao final da contagem, o Timer0 irá gerar um pedido de interrupção ao controlador de interrupção interno do microcontrolador, ao mesmo tempo em que o Timer0 reinicia a contagem. Ao atender o pedido de interrupção, o microcontrolador executará uma subrotina específica na memória de programa.

A seguir é apresentado um exemplo de código.

10.4 Segundo exemplo de Código

O código a seguir demonstra um exemplo de utilização do contador Timer0 para solicitar interrupção automaticamente do microcontrolador, em uma frequência pré-estabelecida (1Hz). A subrotina de interrupção (*interrupt*), ao ser executada, incrementa um contador em software, reinicia o Timer0 com um valor pré-estabelecido, e encerra a subrotina resetando o *flag* de interrupção correspondente.

O programa principal configura a porta B (LEDs), inicializa o Timer0, configura o controlador de interrupção interno do microcontrolador por meio do registrador apropriado, e entra num *loop* infinito. No *loop* infinito, o programa executa a comparação do contador em software com o valor 5, e se for verdadeiro então inverte o estado dos LEDs da porta B, zerando o contador em software.

O efeito produzido pelo sistema será que os LEDs da porta B irão piscar a cada 5 segundos. Repare que não há nenhuma instrução dentro do *loop* infinito para modificar o valor do contador em software, a não ser a instrução para zerar o contador.

```
/*************************************************************
Programa: Interrupcao com Timer0
Objetivo: Este programa tem por objetivo  piscar os leds do
PortB em intervalo de 5 segundos (ligar e desligar)
utilizando o TIMER0 do PIC que gera o pedido de interrupcao
para contagem a cada um segundo.
Placa: Kit PICGenios v3 com PIC18F452
Chaves de funcoes do kit: LED2
Compilador: MikroC PRO v6.4 - usar: "HS oscilator PLL enable"
Data: marco/2017
*************************************************************/

unsigned contagem;

// Esta funcao e' executada toda vez que o Timer0 muda de
// 65535 para 0.
// Para clock do sistema de 8MHz, prescaler = 256, entao:
// 8MHz/256 = 31250Hz
// O Timer0 conta de 34286 ate 65535 (31250 pulsos de clock),
// e entao gera uma interrupcao a cada segundo.
void interrupt(void) {
  contagem++;           // incrementa variavel de contagem

  // reinicia o valor de contagem com 0d34286 = 0x85EE
  TMR0H = 0x85;
  TMR0L = 0xEE;

  INTCON.TMR0IF = 0;  // limpa flag de interrupcao do Timer0
}

void main() {

  TRISB = 0;            // configura PORTB como saida
  PORTB = 0xFF;         // Inicializa PORTB

  // Habilita o Timer0 (TMR0) para clock de 31250Hz = 8MHz/256
  T0CON = 0b10000111;
  // bit 7: 0 - desabilita Timer0
  //        1 - habilita Timer0
  // bit 6: 0 - configura contagem em 16 bits
  //        1 - configura contagem em 8 bits
```

```
42    // bit 5: 0 - fonte de clock do sistema
43    //        1 - fonte de clock do pino T0CKI
44    // bit 4: 0 - clock de T0CKI na transicao positiva
45    //        1 - clock de T0CKI na transicao negativa
46    // bit 3: 0 - clock vem do prescaler
47    //        1 - clock nao passa pelo prescaler
48    // bit 2-0: 000 - prescaler 1:2
49    //          111 - prescaler 1:256
50
51    // Inicia Timer0 com 0d34286 = 0x85EE
52    TMR0H = 0x85;
53    TMR0L = 0xEE;
54
55    INTCON = 0b11100000; // Configura a interrupcao pelo TMR0
56    //bit 7: habilita interrupcao geral
57    //bit 6: habilita interrupcao de perifericos
58    //bit 5: habilita interrupcao por overflow do TIMER0
59    //bit 4: habilita interrupcao externa INT0
60    //bit 3: habilita interrupcao por troca dos bits porta B
61    //bit 2: flag da interrupcao por overflow no TIMER0
62    //bit 1: flag da interrupcao EXTERNA INT0
63    //bit 0: flag da interrupcao por troca dos bits porta B
64
65    contagem = 0;          // Inicializa contador em software
66
67    do {
68      if (contagem == 5) {
69        PORTB = ~PORTB;    // inverte estado dos LEDs
70        contagem = 0;      // Reinicia contagem
71      }
72    } while(1);            // Fim do loop
73  }                        // Fim do programa
```

11 Aula 11

Na décima primeira aula prática do curso será mostrado como utilizar o circuito de comunicação serial do PIC (USART - Transmissor/Receptor Universal Síncrono/Assíncrono).

11.1 Projeto

O esquemático eletrônico mostrado na Figura 45 descreve os componentes utilizados neste exemplo de projeto.

Neste projeto, uma vez configurado e programado, o microcontrolador envia um dado serial (sequência de bits) pelo pino de saída TX (pino RC6), ao mesmo tempo em que pode receber um dado serial pelo pino RX (pino RC7). O circuito interno do microcontrodador (registradores de deslocamento que compõem a USART) é o responsável por converter os dados de forma paralela (8 bits) para serial e vice-versa.

Para compatibilizar a entrada e saída digital do microcontrolador, com nível de tensão 0V (dígito 0) e 5V (dígito 1), com os níveis de tensão padrões da interface RS232 do computador PC, é utilizado um chip MAX232. Esta interface converte os níveis 0V e 5V nas tensões +12V e -12V respectivamente. Repare que o chip MAX232 é alimentado com tensão de apenas +5V. Os capacitores de $1\mu F$ conectados em seus terminais geram as tensões -12V e +12V.

Para conectar o kit PICGenios com o computador, é utilizado um cabo serial padrão RS232 com conectores DB9 (macho e fêmea). Na Figura 46 é apresentada a configuração do cabo.

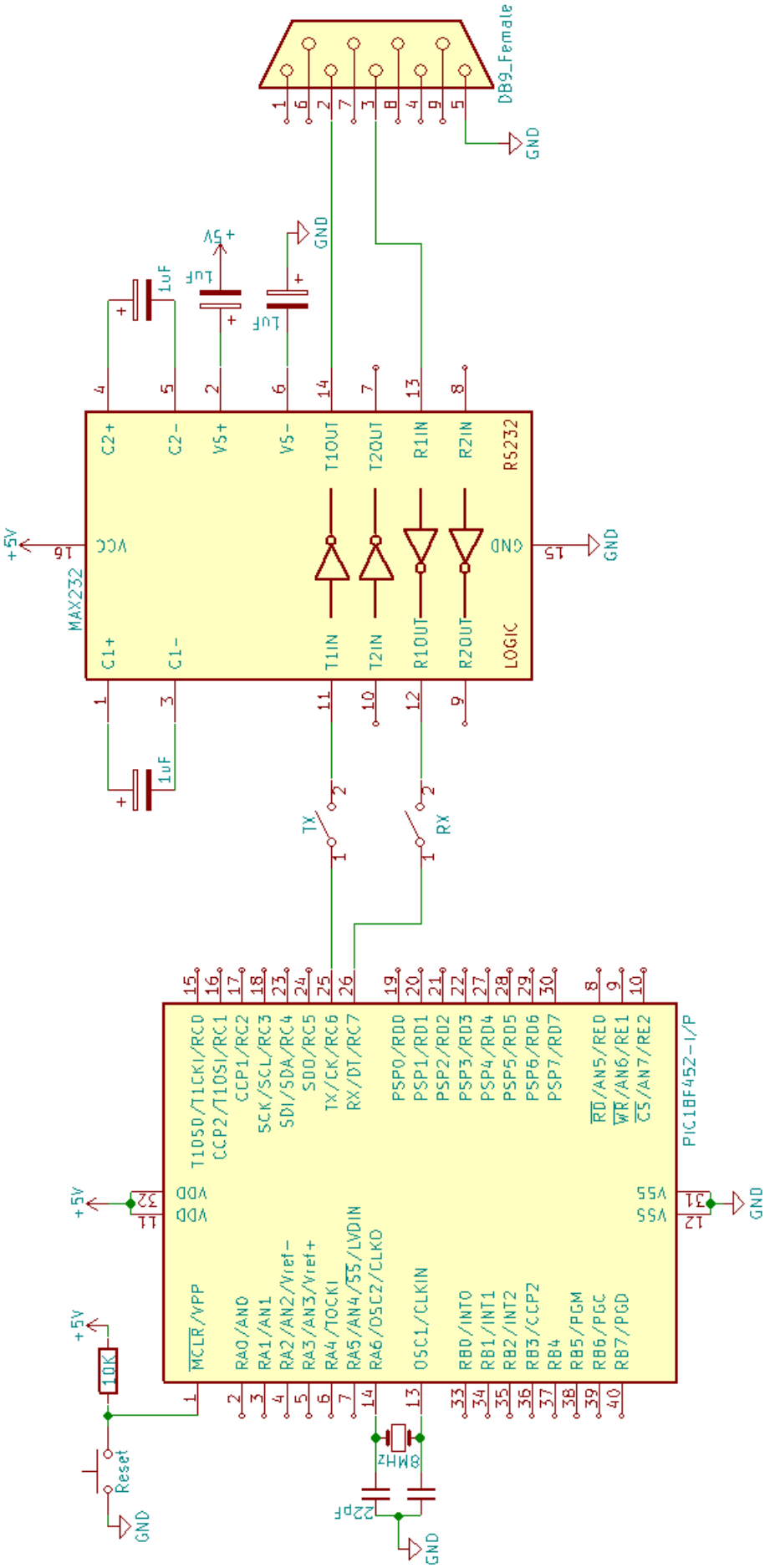

Figura 45: Esquemático do projeto.

Figura 46: Cabo RS232.

Pino	Sinal	Descrição do sinal
1	DCD	Data Carrier Detect
2	RXD	Receive Data
3	TXD	Transmit Data
4	DTR	Data Terminal Ready
5	GND	Signal Ground (Terra)
6	DSR	Data Set Ready
7	RTS	Request do Send
8	CTS	Clear to Send
9	RI	Ring Indicator

Figura 47: Pinagem do conector DB9 padrão RS232.

Na Figura 47 são apresentadas as funções de cada pino do conector DB9 padrão RS232. Neste projeto são utilizados apenas os pinos: TX, RX, e GND.

A seguir é apresentado um exemplo de código para ser executado no projeto acima.

11.2 Exemplo de Código

O código a seguir demonstra um exemplo de utilização da USART interna do microcontrolador PIC. São utilizadas funções disponíveis na biblioteca do compilador MikroC. O kit PICGenios deve estar conectado a interface serial do computador PC por meio do cabo apropriado. No PC, deverá ser executado um programa monitor de serial e configurada a sua interface serial (COM) apropriadamente.

```
1  //*************************************************************
2  // Programa de teste da interface serial do PIC
3  // Compilador: Mikro C PRO 6.4
4  // Placa: Kit PICGenios v3.0 - PIC18F452
5  // Chaves: RX, TX
6  // Conectar o Kit com a porta serial RS232 (COM1) do PC por meio
7  // de cabo serial e configure a porta serial do PC: 9600bps, 8
8  // bits, 1 stop bit, sem paridade
9  // Data: marco - 2017
10 //*************************************************************
11
12 unsigned short string[10] = {'O','l','a','!','\0'};
13 char uart_rd;
14
15 void main() {
16
17   ADCON0 = 0x00;
18   ADCON1 = 0x06;   // Configura todos pinos AN como digital
19
20   UART1_Init(9600);   // Inicializa modulo UART em 9600 bps
21   Delay_ms(100);      // Espera para estabilizar modulo UART
22
23   UART1_Write_Text(string);  // Envia string de caracteres
24   UART1_Write(13);           // retorno de carro
25   UART1_Write(10);           // alimentacao de linha
26
27   while(1) {                      // loop sem fim
28     if (UART1_Data_Ready()) {     // se dado foi recebido,
29       uart_rd = UART1_Read();     // leia o dado
30       UART1_Write(uart_rd);       // envia dado via UART
31     }
32   }
33 } // fim
```

Após algumas instruções de inicialização do programa, a chamada a função *UART1_Init()* inicializa a interface serial do microcontrolador para uma determinada taxa de transmissão, que no exemplo é de 9600bps (bits por segundo). Em seguida, as chamadas da função UART1_Write() enviam dados pela interface serial do microcontrolador. O computador PC receberá estes dados pela sua interface serial e apresentará na sua tela por meio de software monitor de serial.

Logo após, o programa entra em loop infinito. Neste loop, o programa verifica (por meio da função *UART1_Data_Ready()*) se há algum dado recebido serialmente pelo microcontrolador e que esteja pronto para ser lido (já foi convertido de serial para paralelo). Caso haja, então é feita a leitura deste dado pela função *UART1_Read()*. Então, o mesmo dado recebido é retransmitido pela interface serial do microcontrolador.

O efeito produzido pelo sistema será que na tela do computador aparecerá a mensagem 'Ola!', e logo em seguida aparecerão os caracteres que o PC transmite ao microcontrolador, que por sua vez devolve ao PC.

Referências

[1] **Martins**, Nardênio Almeida (2005). Sistemas microcontrolados. São Paulo: Novatec Editora, 2005, 263 p.

[2] **Moreno**, Edward David Moreno; **Penteado**, Cesar Giacomini; **Rodrigues**, Alexandre César (2005). Microcontroladores e FPGAs. São Paulo: Novatec Editora, 2005, 378 p.

[3] **Matic**, Nebojsa; **Andric**, Dragan (2000). Microcontroladores PIC. São Paulo: MikroElektronika, 2000. 252p.

[4] **Oliveira**, André Schneider de; **Andrade**, Fernando Souza de (2010). Sistemas Embarcados - Hardware e Firmware na Prática. São Paulo: Editora Érica, 320 p.

[5] **Pereira**, Fábio (2002). Microcontroladores PIC - Técnicas Avançadas. São Paulo: Érica, 2002. 368 p.

[6] **Pereira**, Fábio (2005). Microcontroladores PIC – Programação em C. São Paulo: Érica, 2005. 360 p.

[7] **Silva Junior**, Vidal Pereira da (1997). Microcontroladores PIC: teoria e prática. São Paulo: Érica, 1997. 139 p.

[8] **Silva Junior**, Vidal Pereira da (1988). Microcontroladores. São Paulo: Érica, 1988. 187p.

[9] **Souza**, David José de (2003); **Lavinia**, Nicolás César. Conectando o PIC - Recursos Avançados. São Paulo: Érica, 2003. 384 p.

[10] **Souza**, David José de (2004). Desbravando o PIC - Ampliado e Atualizado para PIC 16F628A. São Paulo: Érica, 2004. 272 p.

[11] **Wilmshurst**, Tim (2001). An Introduction to the design of small-scale embedded systems. New York: Palgrave, 2001. 411 p.

[12] **Wilmshurst**, Tim (2007). Designing Embedded Systems with PIC Microcontrollers: Principles and applications. United Kingdom: Elsevier Ltd., 2007, 556 p.

[13] **Wolf**, Wayne (2001). Computer as Components: Principles of embedded computing system design. San Francisco: Morgan Kaufmann Publishers, CA. 2001.

[14] **Zanco**, Wagner da Silva (2005). Microcontroladores PIC16F628A/648A - Uma abordagem prática e objetiva. São Paulo: Érica, 2005. 368 p.

[15] **Zanco**, Wagner da Silva (2006). Microcontroladores PIC - Técnicas de Software e Hardware para Projetos de Circuitos Eletrônicos. São Paulo: Érica, 2006. 392 p.