Ambiente de Programação MPLAB

Roteiro Nº 02

Fundação Universidade Federal de Rondônia, Núcleo de Ciência e Tecnologia, Departamento de Engenharia - DEE
Curso de Bacharelado em Engenharia Elétrica - Disciplina de Sistemas Microprocessados
Elaboração: Ivan S. de Oliveira - Revisão: Prof. M.Sc. Ciro Egoavil
Laboratório de Sistemas Microprocessados

I. Introdução

O MPLAB é um programa para PC, que roda sobre a plataforma Windows, e serve como ambiente de desenvolvimento de programas para PICs. O MPLAB é um software desenvolvido pela Microchip e é um ambiente chamado de IDE (Intregrated Environment Development - Ambiente Integrado de Desenvolvimento), pois agrega em um só programa as funções de editor, compilador e gravador.

Ao iniciar o MPLAB, tem-se acesso ao ambiente de trabalho global. Trata-se de uma área para abertura das janelas de trabalho, um menu superior e uma barra de ferramentas com diversos ícones relativos a funções específicas.

Para que se possa trabalhar neste ambiente, não basta o arquivo de código-fonte, é necessário criar um projeto que guarda as informações básicas necessárias à compilação do sistema, tais como a relação arquivos-fonte, opções de compilação e algumas ferramentas de compilação.

O conceito de Workspace é ainda mais abrangente que o de Projeto, pois ele armazena todas as demais informações necessárias ao desenvolvimento de um sistema. No arquivo do Workspace serão armazenadas informações relacionadas a um ou mais projetos associados, qual está ativo no momento, quais as janelas abertas e suas posições na tela, configurações do ambiente de trabalho, etc.

O importante é saber que o MPLAB não funciona adequadamente se um Workspace/Projeto não for aberto.

II. Objetivos

Este trabalho tem por objetivo principal apresentar o ambiente MPLAB e como este pode ser utilizado na criação e compilação de programas para microcontroladores PIC. Este apresenta, também, o software PROTEUS como ambiente de simulação de dispositivos eletrônicos baseados em lógica programável.

III. Softwares Utilizados

- MPLAB
- PROTEUS

IV. Criando um Projeto no MPLAB

Após a instalação do MpLAB, inicialize-o. A tela da Figura 1 será apresentada.

Para iniciar a programação no MPLAB, existe a necessidade de se criar um projeto. No projeto estarão anexados os arquivos fontes e todas as informações relativas ao projeto. Para iniciar

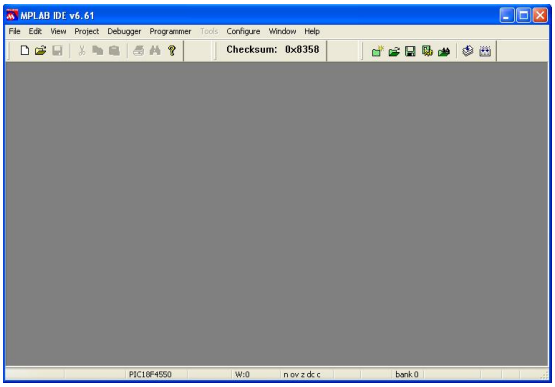

Figura 1. Tela inicial do MPLAB.

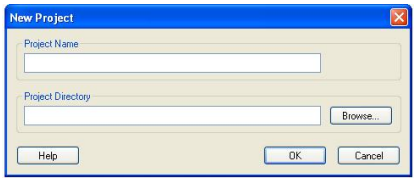

Figura 2. Tela New Project.

um projeto, vá ao menu Project > New. A tela da Figura 2 surgirá.

Em project Name, deve-se informar o nome do projeto e, em Project Directory, a pasta onde ele será salvo. Após a criação do projeto surgirá uma tela semelhante a da Figura 3. Agora será necessário associar um arquivo fonte a projeto para que ele possa ser compilado. O arquivo fonte é aquele onde será escrito o programa. Para isso, vá em File > New. Agora será necessário salvar este arquivo fonte. Note que a extensão do arquivo em *Assembly* será *.asm*.

Após salvar, será necessário associá-lo ao seu projeto. Para isto, clique com o botão direito no item Source Files, que esta presente na Figura 4, e em seguida no item Add Files.

Antes de começar a escrever o programa, vá no menu Configure > Select Device e selecione o microcontrolador que será utilizado no decorrer do projeto. Agora, basta clicar duas vezes sobre o arquivo fonte. Neste ponto, o programa em *Assembly* pode finalmente ser escrito.

V. Montando um Projeto

O processo de montagem consiste na tradução do código em *Assembly* para o código que o microcontrolador é capaz de entender. Para isto, após o programa ser digitado no

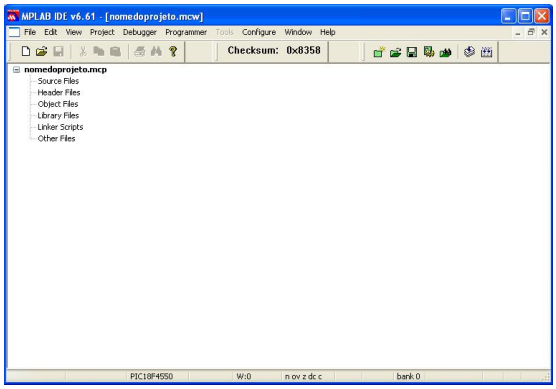

Figura 3. Tela do Projeto .

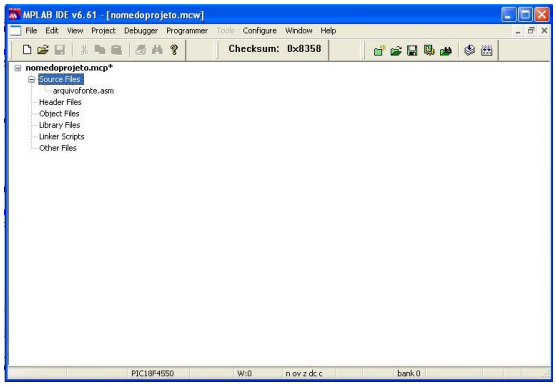

Figura 4. Associação do arquivo fonte.

arquivo fonte, pode-se iniciar a montagem e observar se houve algum erro de sintaxe e se o código já pode ser gravado no microcontrolador. Observe que o MPLAB é capaz de observar somente erros de sintaxe, e não de lógica, ficando estes sob responsabilidade do programador.

Após escrever um programa, para iniciar o processo de montagem, basta pressionar a tecla F10 ou ir no menu Project > Make. Neste momento, irá aparecer um janela como a apresentada na Figura 5.

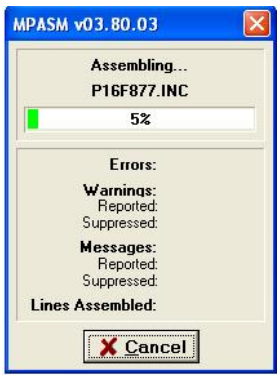

Figura 5. Tela de Compilação.

No final deste processo, caso não haja nenhum erro, aparecerá uma janela de output informando que a operação foi bem sucedida. Observe a Figura 6. No entanto, caso ocorra um erro, o compilador irá informar o erro e a linha onde ele está.

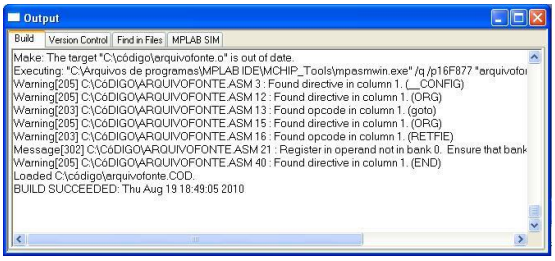

Figura 6. Apararência da tela Output.

No final da compilação, podemos saber quanto foi ocupado em código, tanto da RAM quando da Flash. Para visualizar esta informação, vá no menu View > Memory Usage Gauge.

VI. Programação no MPLAB

O esquema básico para o desenvolvimento de um programa em *Assembly* segue a forma apresentada abaixo:

```
#INCLUDE <P16FXXXX.INC>
CBLOCK 0x0000
ENDC
ORG 0x0000
ORG 0x0004
END
```

Em #INCLUDE <P16FXXX.INC>, é informado o microcontrolador utilizado pelo projeto. Esse arquivo é chamado de "arquivo de definições"e serve para que não se tenha que declarar os registradores do microcontrolador ao acessá-los, pois já estão declarados neste arquivo. Como o microcontrolador utilizado será o PIC16F877, a definição será a seguinte: #INCLUDE <16F877.INC>.

O microcontrolador PIC16F877 possui 368 bytes de memória RAM, divididos em 4 bancos. Portando, quando se escreve CBLOCK 0x0000, está se informando que irá se declarar uma variável (ou registrador) de 8 bits a partir deste endereço. Veja que foi declarada uma variável na posição 0x0000 da memória RAM do microcontrolador. Pode-se declarar várias variáveis utilizando este recurso, de acordo com a necessidade do programa. Verifique o código abaixo:

```
#INCLUDE <P16F877.INC>
CBLOCK 0x0000
        x       ;declara x
        y       ;declara y
ENDC
ORG 0x0000    ;vetor de reset
ORG 0x0004    ;vetor de interrupção
END           ;fim do programa
```

Observe que neste exemplo foi declarada uma variável chamada *x* e outra chamada *y*, estando a primeira alocada no endereço 0x0000 e a segunda no endereço 0x0001. Conforme esta declaração continuar, será preenchido todas as posições relativas a um banco de memória.

Neste exemplo, é possível observar, também, a diretiva **ORG**. Esta diretiva tem a função de determinar o ponto onde

começa a escrita do programa. **ORG** quer dizer origem. Note que as origens estão segmentadas de acordo com o vetor de reset e de interrupção.

O vetor reset refere-se ao primeiro endereço da mémoria de programa que será executado quando o PIC começar a rodar (após a alimentação ou um reset). Na maioria dos modelos, o reset aponta para o endereço 0x0000, mas em alguns modelos mais antigos ele pode apontar para o último endereço disponível.

O vetor de interrupção consiste em um endereço que é reservado para o início do tratamento de todas as interrupções, nos modelos de PIC que possuem esse recurso. O vetor interrupção é encontrado na posição 0x0004 da memória de programa.

Finalmente, o programa acaba com o **END**. Esta é a organização básica de qualquer programa em *Assembly* para microcontrolador.

A. Exemplo 1: Acendendo LED em Assembly

Neste exemplo, será mostrado como criar um programa que aciona um LED de acordo com o estado de um botão. O programa ficará em loop infinito e testará um botão a cada laço e caso este esteja pressionado um LED irá acender. O fluxograma do código é ilustrado na Figura 7.

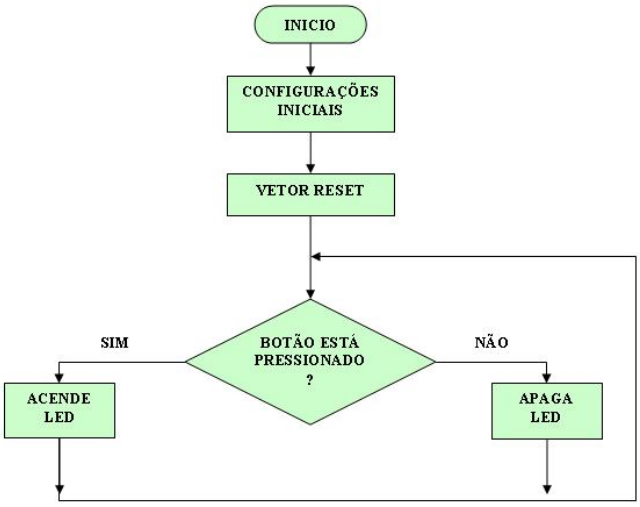

Figura 7. Fluxograma do programa .

O cógigo em *Assembly* e editado no MPLAB que realizará esta tarefa é apresentado abaixo.

```
#INCLUDE <P16F877.INC>

__CONFIG  _LVP_OFF&_PWRTE_ON&_WDT_OFF&_XT_OSC

#DEFINE  BANCO0  BCF STATUS,RP0
#DEFINE  BANCO1  BsF STATUS,RP0


#DEFINE BOTAO  PORTB,0
#DEFINE LED    PORTB,1

ORG 0x0000   ;vetor de reset
     GOTO INICIO

ORG 0x0004   ;vetor de interrupção
     RETFIE

INICIO
    BANCO1
    MOVLW B'00000001'
    MOVWF TRISB

    BANCO0
    CLRF PORTB

LOOP
    BTFSS BOTAO
    GOTO  LIGAR_LED
    GOTO  DESLIGAR_LED

LIGAR_LED
    BSF LED
    GOTO LOOP

DESLIGAR_LED
    BCF LED
    GOTO LOOP

END ;fim do programa
```

Como observado na primeira linha do código, o microcontrolador usado é o PIC16F877, neste ponto é incluído no programa o ficheiro com as características desse microcontrolador. Na segunda linha é realizada a configuração das opções de gravação através da diretriz __CONFIG. As configurações de gravação realizadas neste código são:

- _LVP_OFF = Desativa o sistema de programação em baixa tensão.
- _PWRTE_ON = Liga o Power Up Timer para que o PIC comece a operar cerca de 72 ms após ser alimentado.
- _WDT_OFF = Desliga o WatchDog Timer.
- _XT_OSC = Para osciladores externos tipo cristal ou ressoadores.

Nas linhas seguintes é definido através da diretiva #DEFINE o BANCO0 e o BANCO1 para os ajustes iniciais dos registradores de função especial. Também é definido o Botão no bit 0 e o LED no bit 1 do PORTB.

A diretiva ORG 0x0000 determina o ponto onde deve iniciar a execução do programa, e o comando **GOTO** desvia a execução para o rótulo INICIO.

Na rotina INICIO, o primeiro bit do PORTB é configurado como entrada e os demais como saída. E posteriormente o PORTB é zerado através da instrução **CLRF**.

Após as configurações iniciais, a execução do programa entrará na rotina LOOP onde o estado do botão será testado a cada laço através da instrução **BTFSS**. Caso o botão esteja pressionado (nível lógico baixo "0"), a execução é desviada para o rótulo LIGAR_LED e o LED é acionado através da instrução **BSF**. Caso o botão não esteja pressionado (nível lógico alto "1"), a execução é desviada para o rótulo DESLIGAR_LED e o LED é desligado através da instrução **BCF**.

Após a montagem do programa no MPLAB, um arquivo de extensão .HEX é gerado na mesma pasta onde foi salvo o projeto. O arquivo .HEX pode ser usado para simular o projeto utilizando o software PROTEUS.

O software PROTEUS permite elaborar circuitos com dispositivos de lógica programável como o PIC16F877 e simular seu funciona-

mento.

Para simular a execução do programa descrito anteriormente, monte o circuito da Figura 8. Utilizando os seguintes componentes:

- PIC16F877;
- LED;
- Resistor 10kΩ;
- Resistor 330Ω;
- Botão;
- Fonte 5 V ;

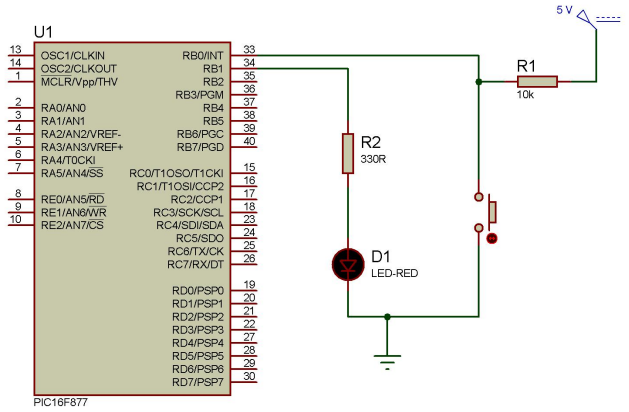

Figura 8. Esquema Elétrico do Circuito.

Após salvar o circuito no PROTEUS, dê um duplo clique sobre o PIC116F877, uma tela surgirá, selecione a caixa Program File e indique o diretório onde o arquivo .HEX, gerado pelo MPLAB, está localizado, como visto na Figura 9. Nesta Figura, ajuste também a caixa Processor Clock Frequency para 4 MHz. Agora simule o circuito, pressione o botão e observe que o comportamento do LED.

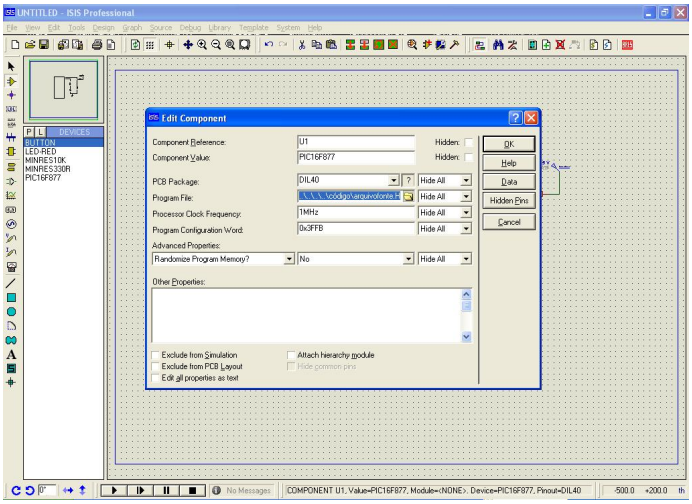

Figura 9. Simulação do programa no PROTEUS.

B. Exemplo 2: Pisca-Pisca com 3 LEDs em Assembly

A partir dos princípios ilustrados no exemplo anterior, é possível criar um programa mais elaborado onde o objetivo acionar 3 LED alternadamente a cada 200 ms, caso um Botão esteja pressionado, ou deixa-los ligados caso o Botão não esteja pressionado. Para que este programa seja possível, é necessário criar uma rotina que faça com que o programa aguarde por 200 ms.

O código que executa tal tarefa é apresentado a seguir:

```
#include <P16F877.INC>

__CONFIG  _LVP_OFF&_PWRTE_ON&_WDT_OFF&_XT_OSC

#DEFINE  BANCO0  BCF STATUS,RP0
#DEFINE  BANCO1  BsF STATUS,RP0

CBLOCK 0X0010
    TEMPO1
    TEMPO2
ENDC

#DEFINE BOTAO  PORTB,0
#DEFINE LED1   PORTB,1
#DEFINE LED2   PORTB,2
#DEFINE LED3   PORTB,3

ORG 0x0000   ;vetor de reset
    GOTO INICIO

ORG 0x0004   ;vetor de interrupção
    RETFIE

DELAY
    MOVLW .200
    MOVWF TEMPO2
DELAY_1
    MOVLW .200
    MOVWF TEMPO1
DELAY_2
    NOP
    NOP
    DECFSZ TEMPO1,F
    GOTO DELAY_2

    DECFSZ TEMPO2,F
    GOTO DELAY_1
RETURN

INICIO
    BANCO1
    MOVLW B'00000001'
    MOVWF TRISB

    BANCO0
    CLRF PORTB

LOOP
    BTFSS BOTAO
    GOTO  LIGAR_LED
    GOTO  DESLIGAR_LED

LIGAR_LED
    BSF LED1
    CALL DELAY
    BSF LED2
    CALL DELAY
    BSF LED3
    CALL DELAY
    BCF LED3
    CALL DELAY
    BCF LED2
    CALL DELAY
```

```
    BCF LED1
    CALL DELAY
    GOTO LOOP

DESLIGAR_LED
    BSF LED1
    BSF LED2
    BSF LED3
    GOTO LOOP

END ;fim do programa
```

Observe que o código é semelhante ao anterior, a diferença está na criação de um bloco de variáveis, onde é declarada a variável TEMPO1 e TEMPO2, e também na utilização de uma rotina de *DELAY*.

A rotina *DELAY* é responsável por paralisar a execução do programa por 200 ms. Isto é obtido pelo decrementando a variável TEMPO1, que tem valor inicial igual a 200. Isto fará com que a rotina DELAY_2 seja executada 200 vezes. Se a rotina DELAY_2 demora 5 microsegundos para ser totalmente executada, então tem-se um tempo total de 1 ms. Se a variável TEMPO2 tem valor inicial de 200, e sendo ela responsável pelo número de execuções da rotina DELAY_1, e como a rotina DELAY_1 demora 1 ms para ser executa, tem-se um tempo total de execução de 200 ms.

Na rotina LOOP o estado do Botão é testado e caso esteja pressionado os LEDS são acionados, um a cada 200 ms, e posteriormente são desligados, um a cada 200 ms. Caso o Botão não esteja pressionado, todos os LEDS são ligados quase simultaneamente, e permanecem neste estado até que o Botão seja pressionado.

Após a compilação do código no MPLAB, monte o circuito da Figura 10 no PROTEUS utilizando os seguintes componentes:

- PIC16F877;
- 3 LEDs;
- Resistor 10kΩ;
- 3 Resistores 330Ω;
- Botão;
- Fonte 5 V ;

Simule o programa, e com este em execução pressione o Botão e verifique o estado dos LEDS.

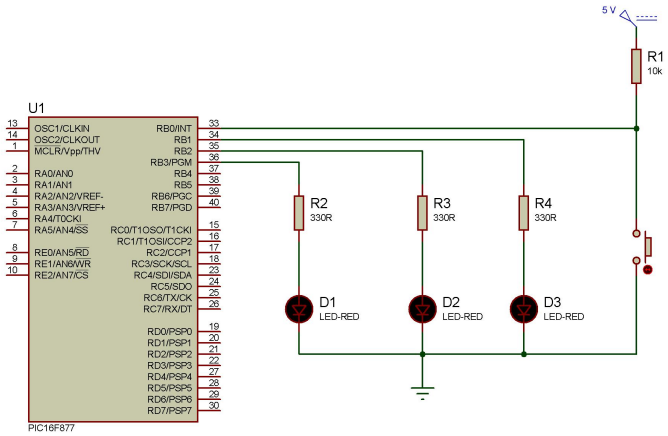

Figura 10. Esquema do Circuito no PROTEUS.

VII. Relatório

Crie um programa no MPLAB que utilize três LEDs, um verde, um amarelo e um vermelho e um Botão, conectados na PORTB do PIC16F877.

Se o Botão estiver pressionado os LEDs devem ser acionados na seguinte sequência:

- 1º LED verde aceso por 250 ms;
- 2º LED verde apagado;
- 3º LED amarelo aceso por 100 ms;
- 4º LED amarelo apagado;
- 5º LED vermelho aceso por 200 ms;
- 6º LED vermelho apagado;

Se o botão não estiver pressionado o LED amarelo de piscar a cada 50 ms.

As opções de gravação utilizadas no programa devem ser as seguintes:

- Oscilador externo tipo cristal (XT);
- Brown Out Detect (BOR) ligado;
- Código de Proteção (CP) desligado;
- WatchDog Timer (WDT) ligado;
- Sistema de programação em baixa tensão (LVP) desativado;

Simule o programa criado no PROTEUS e elabore um relatório detalhando o funcionamento do código e analisando o comportamento do circuito durante a simulação. Inclua no relatório um fluxograma que explique a lógica utilizada na elaboração do programa.

Crie outro programa utilizando os quesitos anteriormente citados, mas agora com os valores de tempo dez vezes maior.

Referências

[1] Souza, Vitor Amadeu, ”Projetando com os microcontroladores da família PIC 18: Uma nova percepção”, 1ª Ed., São Paulo: Ensino Profissional, 2007.

[2] Souza, David José de, ”Desbravando o PIC: ampliado e atualizado para PIC 16F628A”, 6ª Ed., São Paulo: Érica, 2003.