Display LCD com PIC

Roteiro Nº 03

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 Display de cristal líquido, acrônimo de LCD (em inglês *liquid crystal display*), é um painel fino usado para exibir informações por via eletrônica, como texto, imagens e vídeos.

Um LCD consiste de um líquido polarizador da luz, eletricamente controlado, que se encontra comprimido dentro de celas entre duas lâminas transparentes polarizadoras. Os eixos polarizadores das duas lâminas estão alinhados perpendicularmente entre si. Cada cela é provida de contatos elétricos que permitem que um campo elétrico possa ser aplicado ao líquido no interior.

Entre as suas principais características está a sua leveza, sua portabilidade, e sua capacidade de ser produzido em quantidades muito maiores do que os tubos de raios catódicos (CRT). Seu baixo consumo de energia elétrica lhe permite ser utilizado em equipamentos portáteis, alimentados por bateria eletrônica.

É um dispositivo eletrônico-óptico modulado, composto por um determinado número de pixels, preenchidos com cristais líquidos e dispostos em frente a uma fonte de luz para produzir imagens em cores ou preto e branco.

Os módulos LCD são interfaces de saída muito útil em sistemas microprocessados. Estes módulos podem ser gráficos e a caractere e são projetados para conectar-se com a maioria das CPU's disponíveis no mercado, bastando para isso que esta CPU atenda as temporizações de leitura e escrita de instruções e dados, fornecido pelo fabricante do módulo.

No entanto, ocorre que o módulo LCD quando alimentado necessita de algumas instruções de inicialização que identificará qual a forma de transmissão de dados que será estabelecida entre a CPU e o módulo.

No apêndice deste trabalho são apresentados informações sobre a pinagem, instruções de inicialização e endereços de memória para um display LCD 16x2.

II. Objetivos

Neste roteiro é realizado uma introdução ao controle de display LCD com microcontroladores PIC, utilizando as linguagens *Assembly*, no ambiente de programação MPLAB e, também, a linguagem *C*, no ambiente *CCS C Compiler*. É apresentado também, a utilização de recursos disponíveis no microcontroladores como: Interrupções e Timers.

III. Softwares Utilizados

- MPLAB
- CCS C Compiler
- PROTEUS Professional

IV. Ambiente MPLAB

O controle do didplay LCD 16x2 pode ser realizado utilizando a linguagem *Assembly* para microcontrolador PIC conforme o código abaixo, onde se propôs a escrita de duas frases; "ENGENHARIA ELETRICA"na primeira linha e "Sistemas Microprocessados"na segunda. As frases movimentam-se uma coluna a cada interrupção gerada pelo estouro do *Timer0*.

```
#include <P16F877A.INC>

 __CONFIG _CP_OFF&_PWRTE_ON& _WDT_OFF & _XT_OSC

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


 CBLOCK 0X020
   TEMPO1
   TEMPO0
 ENDC

 #DEFINE DISPLAY PORTD
 #DEFINE ENABLE PORTE,1
 #DEFINE RS PORTE,0


 ORG 0x0000
   GOTO CONFIGU

 ORG 0X0004
   BCF INTCON,T0IF
   MOVLW 0X1C
   CALL ESCREVE
   RETFIE


DELAY_MS
  MOVWF TEMPO1
DELAY_MSB
  MOVLW .250
  MOVWF TEMPO0
DELAY_MSA
  NOP
  DECFSZ TEMPO0,F
  GOTO DELAY_MSA

  DECFSZ TEMPO1,F
  GOTO DELAY_MSB
  RETURN
```

```
ESCREVE
  MOVWF DISPLAY
  NOP
  BSF ENABLE
  MOVLW .1
  CALL DELAY_MS
  BCF ENABLE
  NOP
  RETURN


CONFIGU
  BANCO1
  MOVLW B'00000000'
  MOVWF TRISD

  MOVLW B'00000000'
  MOVWF TRISE


  MOVLW B'10001110'
  MOVWF ADCON1

  MOVLW B'10000111'
  MOVWF OPTION_REG

  BANCO0
  MOVLW B'10000000'
  MOVWF INTCON

  MOVLW B'01000000'
  MOVWF ADCON0

  CLRF TEMPO0
  CLRF TEMPO1
  CLRF PORTD
  CLRF PORTE

  MOVLW .1
  CALL DELAY_MS


INICIALIZACAO_DISPLAY
  BCF RS
  MOVLW 0x38
  CALL ESCREVE

  MOVLW .2
  CALL DELAY_MS

  MOVLW 0X38
  CALL ESCREVE

  MOVLW 0X06
  CALL ESCREVE

  MOVLW 0x0C
  CALL ESCREVE

  MOVLW 0X01
  CALL ESCREVE

  MOVLW .1
  CALL DELAY_MS


  BCF     RS
  MOVLW 0X92
  CALL ESCREVE
  BSF RS
  MOVLW 'E'
  CALL ESCREVE
  MOVLW 'N'
  CALL ESCREVE
  MOVLW 'G'
  CALL ESCREVE
  MOVLW 'E'
  CALL ESCREVE
  MOVLW 'N'
  CALL ESCREVE
  MOVLW 'H'
  CALL ESCREVE
  MOVLW 'A'
  CALL ESCREVE
  MOVLW 'R'
  CALL ESCREVE
  MOVLW 'I'
  CALL ESCREVE
  MOVLW 'A'
  CALL ESCREVE
  MOVLW ' '
  CALL ESCREVE
  MOVLW 'E'
  CALL ESCREVE
  MOVLW 'L'
  CALL ESCREVE
  MOVLW 'E'
  CALL ESCREVE
  MOVLW 'T'
  CALL ESCREVE
  MOVLW 'R'
  CALL ESCREVE
  MOVLW 'I'
  CALL ESCREVE
  MOVLW 'C'
  CALL ESCREVE
  MOVLW 'A'
  CALL ESCREVE


  BCF     RS
  MOVLW 0XCF
  CALL ESCREVE
  BSF RS
  MOVLW 'S'
  CALL ESCREVE
  MOVLW 'i'
  CALL ESCREVE
  MOVLW 's'
  CALL ESCREVE
  MOVLW 't'
  CALL ESCREVE
  MOVLW 'e'
  CALL ESCREVE
  MOVLW 'm'
  CALL ESCREVE
  MOVLW 'a'
  CALL ESCREVE
  MOVLW 's'
  CALL ESCREVE
  MOVLW ' '
  CALL ESCREVE
  MOVLW 'M'
  CALL ESCREVE
  MOVLW 'i'
```

```
  CALL ESCREVE
  MOVLW 'c'
  CALL ESCREVE
  MOVLW 'r'
  CALL ESCREVE
  MOVLW 'o'
  CALL ESCREVE
  MOVLW 'p'
  CALL ESCREVE
  MOVLW 'r'
  CALL ESCREVE
  MOVLW 'o'
  CALL ESCREVE
  MOVLW 'c'
  CALL ESCREVE
  MOVLW 'e'
  CALL ESCREVE
  MOVLW 's'
  CALL ESCREVE
  MOVLW 's'
  CALL ESCREVE
  MOVLW 'a'
  CALL ESCREVE
  MOVLW 'd'
  CALL ESCREVE
  MOVLW 'o'
  CALL ESCREVE
  MOVLW 's'
  CALL ESCREVE
  BCF RS
  BCF INTCON,T0IF
  BSF INTCON,5


LOOP
  GOTO LOOP
  END
```

No início do código é definido o PORTD o responsável pela comunicação com a entrada de dados do LCD e os bits 0 e 1 do PORTE responsável pelo envio dos sinais de controle *Enable* e *RS*. Logo após, é realizada a configuração dos registradores de funções especiais.

O registrador ADCON1 é configurado de forma a permitir que os bits da PORTE sejam utilizados como I/O digital e o ADCON0 está configurado para desligar a conversão AD.

O registrador OPTION_REG é responsável pela configuração de algumas funções internas do PIC. No código em questão, este registrador determina que o clock do *Timer0* será proveniente do clock interno do sistema (Fosc/4) e determina, também, o prescaler conectado a o *Timer0* para 1:256.

O INTCON é responsável pelo controle de interrupções no PIC. Este foi configurado para habilitar a interrupção proveniente do estouro do *Timer0*.

Na sequência do código e realizada a inicialização do display, a função de cada comandos em hexadecimal enviado é apresentada na Tabela 1 do Apêndice. Após a inicialização os caracteres a serem apresentados no display são enviados um a um, e a cada novo caractere o cursor é deslocado para a direita.

O programa permanece em loop até que o estouro do *Timer0* ocorra e provoque uma interrupção, então, a execução é desviada para o vetor de interrupções ORG 0x0004, onde é enviado o comando para o display deslocar a mensagem para a esquerda.

A Figura 1 apresenta a simulação deste programa no software PROTEUS.

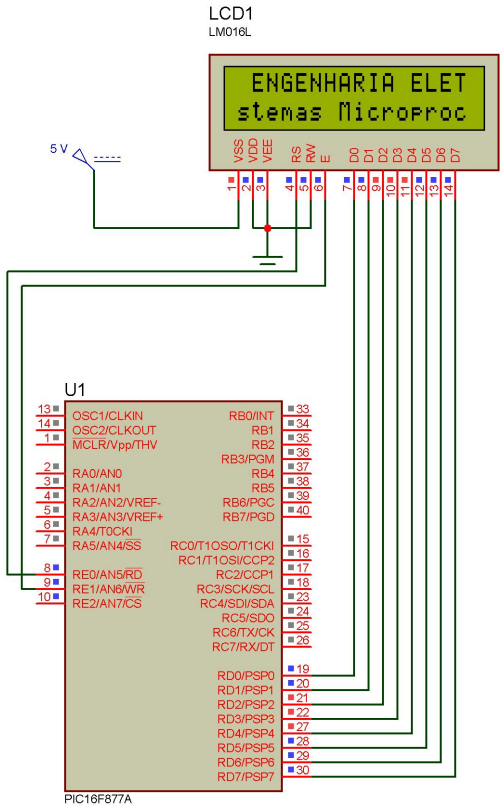

Figura 1. Simulação no PROTEUS.

V. CCS C Compiler

A principal vantagem de se usar linguagens de alto nível, no caso a linguagem *C*, esta na menor interação do programador com o hardware, no que diz respeito ao controle do mesmo (ajuste de bancos de registradores, sequências de inicialização, etc.). Desta forma o programador dedica seu tempo basicamente à lógica do problema e não aos detalhes do microcontrolador.

A linguagem *C* é estruturada, cuja característica e a *compartimentalização* do código e dos dados. Trata-se da habilidade de uma linguagem seccionar e esconder do resto do programa todas as informações necessárias para se realizar uma tarefa específica. Umas das maneiras de conseguir essa compartimentalização é pelo uso de sub-rotinas que empregam variáveis locais (temporárias). Com uso de variáveis locais é possível escrever sub-rotinas de forma que os eventos que ocorram dentro delas não causem nenhum efeito inesperado nas outras partes do programa. Essa capacidade permite que os programas em *C* compartilhem facilmente seções de código. Portanto, nesta parte do roteiro será realizada uma introdução

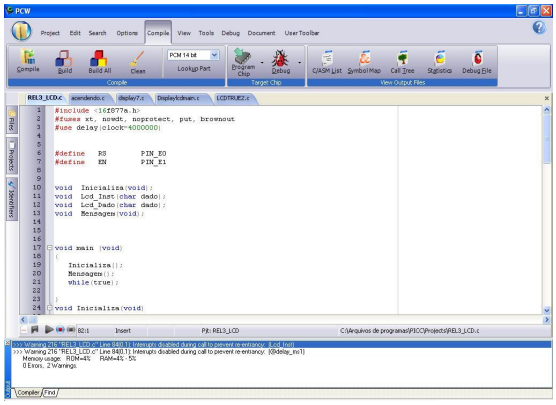

Figura 2. Interface do CCS C Compiler.

a programação na linguagem *C* para microcontroladores PIC no ambiente CCS C Compiler, cuja interface é apresentada na Figura 2.

Para exemplificar a utilização desta linguagem na programação, abaixo é apresentado um código que realização a mesma função do código, anteriormente apresentado, escrito em *Assembly*.

```
#include <16f877a.h>
#fuses xt, nowdt, noprotect, put, brownout
#use delay(clock=4000000)


#define   RS        PIN_E0
#define   EN        PIN_E1

void  Inicializa(void);
void  Lcd_Inst(char dado);
void  Lcd_Dado(char dado);
void  Mensagem(void);


void main (void)
{
  Inicializa();
  Mensagem();
  while(true);
}

void Inicializa(void)
{
  Setup_ADC(ADC_OFF);
  Set_Timer0(0);
  Setup_Timer_0(RTCC_INTERNAL | RTCC_DIV_256);
  Enable_Interrupts(INT_TIMER0);
  Enable_Interrupts(GLOBAL);
  Lcd_Inst(0x38);
  Delay_ms(2);
  Lcd_Inst(0x38);
  Lcd_Inst(0x0C);
  Lcd_Inst(0x06);
  Lcd_Inst(0x01);
  Delay_ms(1);
}

#int_timer0
void Ist_Timer0(void)
{
  Lcd_Inst(0x1C);
}

void  Mensagem(void)
{
  Lcd_Inst(0x92);
  printf(lcd_dado, "ENGENHARIA ELETRICA");
  Lcd_Inst(0xCF);
  printf(lcd_dado, "Sistemas Microprocessados");
}

void  Lcd_Inst(char dado)
{
  Disable_Interrupts(GLOBAL);
  output_low(RS);
  output_d(dado);
  delay_cycles(2);
  output_high(EN);
  delay_ms(1);
  output_low(EN);
  Enable_Interrupts(INT_TIMER0);
}

void  Lcd_Dado(char dado)
{
  Disable_Interrupts(GLOBAL);
  output_high(RS);
  output_d(dado);
  delay_cycles(2);
  output_high(EN);
  delay_ms(1);
  output_low(EN);
  Enable_Interrupts(GLOBAL);
}
```

O código é iniciado com as instruções reproduzidas abaixo:

```
#include <16f877a.h>
#fuses xt, nowdt, noprotect, put
#use delay(clock=4000000)


#define   RS        PIN_E0
#define   EN        PIN_E1


void  Inicializa(void);
void  Lcd_Inst(char dado);
void  Lcd_Dado(char dado);
void  Mensagem(void);
```

Neste trecho, é realizado a inclusão do ficheiro com as características do PIC16F877A, microcontrolador utilizado no projeto. Posteriormente, os fusíveis ou opções de gravação são ajustados da seguinte forma:

- Oscilador externo tipo cristal (xt);
- Power-up timer (put) ligado;
- Código de Proteção (noprotect) desligado;
- WatchDog Timer (nowdt) desligado;

Logo após,é definido a frequência do clock e são definidos os bits de comando *Enable* como o bit 0 do PORTE e o *RS* o bit "1"do PORTE. Em seguida são declaradas as funções que

serão utilizadas no corpo do programa.

Assim como na programação em *Assembly*, a diretiva do vetor reset ORG 0X0000 corresponde ao endereço de inicio para a execução do programa, na linguagem *C*, a função *main*, determina, também, o ponto inicial para a execução.

```
void main (void)
{
  Inicializa();
  Mensagem();
  while(true);
}
```

Na função $main$ são chamadas duas outras funções; a função $Inicializa()$ responsável por realizar a configuração inicial do PIC e, também, do display e função $Mensagem()$ responsável por definir a mensagem que será enviada ao display. Na função $main$, o comando $while(true)$ cria um laço que será executado até que uma interrupção ocorra.

Na função $Inicializa()$, e desligada a conversão AD (Setup_ADC(ADC_OFF)), é determinado valor inicial do *Timer0* (Set_Timer0(0)) e, também, são habilitadas a interrupção do $Timer0$ (Enable_Interrupts(INT_TIMER0)) e a interrupção global (Enable_Interrupts(GLOBAL)).

```
void Inicializa(void)
{
  Setup_ADC(ADC_OFF);
  Set_Timer0(0);
  Setup_Timer_0(RTCC_INTERNAL | RTCC_DIV_256);
  Enable_Interrupts(INT_TIMER0);
  Enable_Interrupts(GLOBAL);

  Lcd_Inst(0x38);
  Delay_ms(2);
  Lcd_Inst(0x38);
  Lcd_Inst(0x0C);
  Lcd_Inst(0x06);
  Lcd_Inst(0x01);
  Delay_ms(1);
}
```

Em seguida, são enviados os comandos de inicialização do display de LCD. A função de cada comando é apresentada na Tabela 1 do Apêndice.

Na função $Mensagem()$ são definidos os caracteres que serão apresentados no display e, também a posição em que deve ser iniciada a escrita.

```
void  Mensagem(void)
{
  Lcd_Inst(0x92);
  printf(lcd_dado, "ENGENHARIA ELETRICA");
  Lcd_Inst(0xCF);
  printf(lcd_dado, "Sistemas Microprocessados");
}
```

Neste trecho, são utilizadas as funções $Lcd\_Inst()$, para enviar os comandos para o display, neste caso informando a posição inicial, e a função $Lcd\_Dado()$ responsável por enviar cada caractere da frase a ser apresentada no display. A função $printf()$ faz parte da biblioteca de funções de I/O da linguagem *C* e realiza a saída de dados formatada, isto é, pode escrever dados em vários formatos que estão sob seu controle.

VI. Relatório

Altere os programas, em *Assembly* e em *C*, mostrados anteriormente, para que apresente na primeira linha do display a frase "Sistemas Microprocessados"e na segunda linha os nomes de todos os integrante do grupo. As frases devem se deslocar para a direita a cada 200 ms, sendo que o tempo de deslocamento deve ser determinado pelo estouro do *Timer0*.

Elabore um relatório apresentado ambos os códigos e as simulações no PROTEUS. Relate as diferença entre as linguagens no que diz respeito a facilidade em programar e quantidade de memória ocupada no microcontrolador.

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.

[3] Pereira, Fábio, "Micrcontroladores PIC: Técnicas Avançadas", 6ªEd., São Paulo: Érica, 2007.

[4] Apostila de Linguagem C para PIC16F877A com base no CCS - Cerne Tecnologia e Treinamento LTDA.

[5] Schildt, Herbert, C, Completo e Total, 3ª Ed., São Paulo: Pearson Markron Books, 1997.

APÊNDICE

Hexadecimal	**Binário**	**Descrição**
0x38	00111000	Comunicação do módulo com 8 bits
0x06	00000110	Sentido do deslocamento do cursor: para a direita.
0x0C	00001100	Liga display sem cursor.
0x01	00000001	Limpa display com Home cursor.
0x18	00011000	Deslocamento da mensagem sem entrada de caractere: para a esquerda.
0x1C	00011100	Deslocamento da mensagem sem entrada de caractere: para a direita.

Tabela I
COMANDOS PARA INICIALIZAÇÃO DO DISPLAY

LCD 16X2	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Linha 1	80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
Linha 2	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF

Tabela II
ENDEREÇOS DE CARACTERES NO DISPLAY.

Pino	**Nome**	**Função**	**Descrição**
1	Vss	Alimentação	Terra ou GND
2	Vdd	Alimentação	VCC ou +5V
3	Vee	V0	Tensão de ajuste de contraste.
4	RS	Register select	1-Dado, 0-Instrução.
5	R/W	Read/Write	1-Leitura, 0-Escrita.
6	E	Chip select	1 ou (1→0) - Habilita, 0 - Desabilita.
7	D0	Dado	Bit 0
8	D1	Dado	Bit 1
9	D2	Dado	Bit 2
10	D3	Dado	Bit 3
11	D4	Dado	Bit 4
12	D5	Dado	Bit 5
13	D6	Dado	Bit 6
14	D7	Dado	Bit 7

Tabela III
PINAGEM DO MÓDULO LCD