Prática 2:

Usando I/O com teclado e display de 7 segmentos

2.1 – Introdução e objetivos

Esta prática tem por objetivo trabalhar com entrada (input) e saída de dados (output) através da leitura ou envio de tensões nos pinos de I/O (input/output) exclusivamente dedicados para esta tarefa. No caso dos PIC, os pinos podem ser configurados para ler valores de tensão digitais ou para enviar sinais digitais. Para ilustrar este processo, esta prática usará para leitura (input) um teclado numérico matricial similar ao que se vê na Figura 2.1. Ao mesmo tempo, também usará um display de 7 segmentos para ilustrar o processo de output. Assim, o valor lido no teclado deve ser lido e processado pelo PIC e enviado para visualização no display.

O teclado matricial é um item comumente encontrado em dispositivos que necessitam de uma entrada de dados numérica tais como, por exemplo, forno microondas, telefones, sistemas de controle de acesso por senha etc. Ele é chamado matricial pois seus botões são dispostos de uma forma similar aos elementos de uma matriz organizados por linhas e colunas. Assim, quando uma de suas teclas é pressionada, um contato elétrico é fechado entre a linha e coluna que estão ligados este botão. A Figura 2.1b ilustra o diagrama elétrico deste teclado.

(a) (b)

Figura 3.1 - (a) Exemplo de teclado matricial. (b) Esquema elétrico do teclado matricial. Podem haver pequenas diferenças, porém a ideia de se fechar contatos entre linhas e colunas através do pressionamento de um determinado botão é mantida nestas versões de teclado matricial.

Assim, os objetivos desta prática são:

- ilustrar como acontece o processo de I/O em microcontroladores;
- manipular teclados matriciais e fazer a leitura de números;
- trabalhar com display de 7 segmentos.

2.2 – Exemplo de aplicação

Esta prática propõe um exemplo onde o usuário deve pressionar um botão do teclado e este valor deve ser lido no PIC que deverá exibir o valor lindo em um display.

2.2.1 - O circuito

O circuito elétrico da aplicação é mostrado na Figura 2.2. Neste esquemático é usado um teclado 3x4 (colunas x linhas). Destaca-se que podem haver outros modelos de teclado que apresentem significativas variações em relação ao apresentado neste diagrama elétrico. Daí pode haver a necessidade de pequenas modificações assim como no código fonte da aplicação.

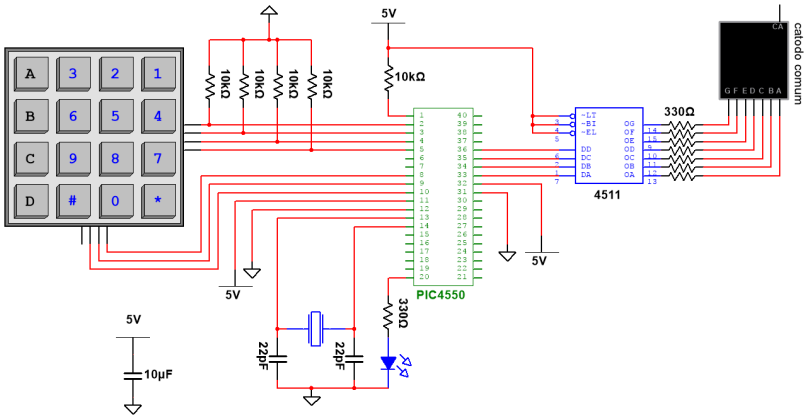

Figura 3.2 - Esquema elétrico da aplicação proposta que pretende varrer o teclado procurando por teclas pressionadas e mostrar seu correspondente valor em um display 7 segmentos acionado por um decodificador BCD para 7 segmentos.

No circuito da Figura 2.2, o PIC varre as colunas do teclado através dos pinos 8, 9 e 10 (portas E2 a E0, respectivamente) e lê as correspondentes linhas do teclado através dos pinos 2, 3, 4 e 5 (portas A3 a A0, respectivamente). Assim a porta E é de saída e a porta A é de entrada. As resistências de *pull up* ligadas aos pinos 2 a 5 servem para não deixar a entrada da porta A flutuando quando nenhuma tecla é pressionada. Sem elas muito ruído pode interferir na leitura da porta A.

2.2.2 – O código

Baseado no circuito elétrico apresentado, o seguinte código deve ser gravado na memória de programa do PIC. Nas suas linhas 4 a 10 são associadas as portas do PIC com os fios de teclado. Esta associação pode mudar segundo a aplicação e indica apenas uma sugestão para este exemplo e o circuito da Figura 2.2.

Código 3.1 - Código da prática para varredura do teclado e exibição do valor em display 7 segmentos

```
1   #include <main.h>
2
3   #define ESPERA_MS 250
4   #define COLUNA1 PIN_E0
5   #define COLUNA2 PIN_E1
6   #define COLUNA3 PIN_E2
7   #define LINHA1 PIN_A0
8   #define LINHA2 PIN_A1
9   #define LINHA3 PIN_A2
10  #define LINHA4 PIN_A3
11
12  signed int le_teclado(void);
13  int varre_coluna(int1, int1, int1);
14  ////////////////////////////////////////////////////////////////////////////
15  void main()
16  {
17     output_b(0);
18     signed int tecla_lida;
19     while(TRUE)
```

```
20      {
21         tecla_lida = le_teclado();
22         if(tecla_lida >= 0)
23         {
24            output_b(tecla_lida);
25            output_high(PIN_D1);  delay_ms(100);  output_low(PIN_D1);  //pisca led
26         }
27         delay_ms(ESPERA_MS);
28      }
29   }
30   ////////////////////////////////////////////////////////////////////////
31   signed int le_teclado(void)
32   {
33      int linha;
34
35      linha = varre_coluna(0b1, 0b0, 0b0); //varre a primeira coluna do teclado
36      if(linha > 0) //se achar alguma tecla, devolve seu correspondente int.
37      {
38         if(linha == 1)//se foi pressionada a linha 1 amarela
39            return 10;//tecla esteristico (dependendo do teclado pode ser outra!)
40         if(linha == 2)//se foi pressionada a linha 2 azul
41            return 1;//tecla 1 (dependendo do teclado pode ser outra!)
42         if(linha == 3)//se foi pressionada a linha 3 branco
43            return 4;//tecla 4 (dependendo do teclado pode ser outra!)
44         if(linha == 4)//se foi pressionada a linha 4 cinza
45            return 7;//tecla 7 (dependendo do teclado pode ser outra!)
46      }
47
48      linha = varre_coluna(0b0, 0b1, 0b0); //varre a segunda coluna do teclado
49      if(linha > 0) //se achar alguma tecla, devolve seu correspondente int.
50      {
51         if(linha == 1)//se foi pressionada a linha 1 amarela
52            return 12;//tecla cerquilha (dependendo do teclado pode ser outra!)
53         if(linha == 2)//se foi pressionada a linha 2 azul
54            return 3;//tecla 3 (dependendo do teclado pode ser outra!)
55         if(linha == 3)//se foi pressionada a linha 3 branco
56            return 6;//tecla 6 (dependendo do teclado pode ser outra!)
57         if(linha == 4)//se foi pressionada a linha 4 cinza
58            return 9;//tecla 9 (dependendo do teclado pode ser outra!)
59      }
60
61      linha = varre_coluna(0b0, 0b0, 0b1); //varre a terceira coluna do teclado
62      if(linha > 0) //se achar alguma tecla, devolve seu correspondente int.
63      {
64         if(linha == 1)//se foi pressionada a linha 1 amarela
65            return 0;//tecla 0 (dependendo do teclado pode ser outra!)
66         if(linha == 2)//se foi pressionada a linha 2 azul
67            return 2;//tecla 2 (dependendo do teclado pode ser outra!)
68         if(linha == 3)//se foi pressionada a linha 3 branco
69            return 5;//tecla 5 (dependendo do teclado pode ser outra!)
70         if(linha == 4)//se foi pressionada a linha 4 cinza
71            return 8;//tecla 8 (dependendo do teclado pode ser outra!)
72      }
73      return -1;  //se nao retornou nada até agora, devolve 13
74   }
75   ////////////////////////////////////////////////////////////////////////
76   int varre_coluna(int1 C1, int1 C2, int1 C3)
77   {
78      output_bit(COLUNA1, C1 );
79      output_bit(COLUNA2, C2 );
80      output_bit(COLUNA3, C3 );
81      delay_ms(100);//espera estabilizar o contato entre as chaves para ler
82      if(input(LINHA1))
83         return 1;
84      if(input(LINHA2))
85         return 2;
86      if(input(LINHA3))
87         return 3;
88      if(input(LINHA4))
89         return 4;
90      return 0;//se nenhum return anterior foi executado, este sera!
91   }
```

A função main tem um laço infinito que busca constantemente varrer o teclado procurando por teclas pressionadas através da chama a rotina "`le_teclado()`". Na linha 22 é conferido se alguma teclada foi detectada por esta função (ela retorna um inteiro correspondente ao valor teclado ou, se nada foi teclado, retorna o valor -1). Se alguma tecla foi pressionada (linha 22), seu valor é exibido na porta D (linha 24) e um LED ligado à porta D1 é piscado por um instante de 100ms. Uma vez varrido o teclado, o PIC é colocado em espera (linha 27) para que o usuário solte a tecla.

A função "`le_teclado`" chama a função "`varre_coluna`" para varrer cada uma das 3 colunas do teclado (linhas 35, 48 e 61). Esta última função retorna a linha do teclado pressionada (linhas 83, 85, 87 e 89). Se nenhuma linha é ativa, a função retorna 0 (linha 90). Note que nas linhas 78 a 80 uma das colunas é ativada por qualquer uma das chamas desta função que acontece nas linhas 35, 48 e 61.

O bloco de código 36 a 46, por exemplo, faz com que a primeira coluna do teclado seja alimentada por VCC e depois verifica qual linha foi ativada. Conforme se verifica nas linhas 39, 41, 43 e 45 é definido qual tecla foi pressionada sabendo que se está na coluna 1 e também identificando a linha do teclado pressionada. Estes valores podem variar de teclado para teclado e deve ser testados antes de produzir o código. Este processo se repete outras vezes para a segunda e terceira colunas do teclado. Por fim, na linha 81, é dado um atraso de 100ms para que o barramento elétrico seja estabilizado e o programa não leia ruídos causados, principalmente, pela trepidação das chaves (problema de *debounce* típicos de chaves mecânicas).