```
1   #include<p18f452.h>
2   #include<delays.h>
3   #include<pwm.h>
4   #include<compare.h>
5   #include<timers.h>
6   #include<math.h>
7   #include<portb.h>
8
9   #pragma config WDT = OFF, LVP = OFF, OSC = XT, PWRT = ON, BOR = ON, BORV = 42
10
11  //Protótipos de funções
12  void ISR_High_Priority(void);
13  void ISR_Low_Priority(void);
14  void inicia_lcd(void);          //Função de inicialização do LCD
15  void escreve_comando(char c);   //Envia comando para o LCD
16  void escreve_dado(char d);      //Escrita de uma dado no LCD
17
18  //Definição de variáveis globais
19  unsigned int dutyPWM = 0;
20  unsigned char W_temp, BSR_temp, STATUS_temp;
21  unsigned char div_freq_tmr0 = 20;
22  unsigned char decod[] = {'0','1','2','3','4','5','6','7','8','9'};
23
24  //-----------------------------------------------------------------
25  #pragma code vec_int_high_priority = 0x08
26  void vec_int_high_priority(void)
27  { _asm GOTO ISR_High_Priority _endasm }
28
29  #pragma code vec_int_low_priority = 0x18
30  void vec_int_low_priority(void)
31  { _asm GOTO ISR_Low_Priority _endasm }
32
33  #pragma code
34  void ISR_High_Priority(void)
35  {
36      INTCONbits.TMR0IF = 0;
37      //WriteTimer0();
38      //..
39      _asm
40      RETFIE 1
41      _endasm
42  }
43
44  //-----------------------------------------------------------------
45  void ISR_Low_Priority(void)
46  {
47      _asm
48      MOVFF   WREG,W_temp
49      MOVFF   STATUS, STATUS_temp
50      MOVFF   BSR, BSR_temp
51      _endasm
52
53      //;
54
55      _asm
56      MOVFF   W_temp, WREG
57      MOVFF   STATUS_temp, STATUS
58      MOVFF   BSR_temp, BSR
59      RETFIE 0
60      _endasm
61  }
62
63  //-----------------------------------------------------------------
64  void main(void)
65  {
66      ADCON1 = 0x07;      // PORTA como pinos digitais
67
68      TRISA = 0xFF;
69      TRISB = 0b00001100;
70      TRISC = 0b11111001;
71      TRISD = 0x00;
72      TRISE = 0b00000000;
73
74      PORTB = 0x00;
75      PORTD = 0x00;
76
```

```
77      OpenTimer1(TIMER_INT_OFF & T1_16BIT_RW & T1_SOURCE_EXT
78              & T1_PS_1_1 & T1_OSC1EN_OFF & T1_SOURCE_CCP);
79      WriteTimer1(0);
80
81      OpenCompare1(COM_INT_ON & COM_TRIG_SEVNT, 7);   //CCP1 como COMPARE e disparo de
82                                                      // TMR1 = CCP1H/L
83                                                      // e seta valor de comparação
84
85      IPR1bits.CCP1IP = 0;        //COMPARE é de baixa prioridade
86      IPR2 = 0;                   //Int. de periféricos é de baixa prioridade
87
88      INTCON2 = 0b00000100;       //Int. TMR0 é de alta prioridade
89      OpenTimer0(TIMER_INT_ON &               //Liga interrupção
90                 T0_16BIT & T0_SOURCE_INT &  //Clock interno
91                 T0_PS_1_1);                 //Prescaler 1:32
92
93      WriteTimer0(15536);     //15536 = 65536 - 50000 (50000 ciclos de p/ estouro)
94
95      PIR1bits.CCP1IF = 0;
96      INTCONbits.TMR0IF = 0;      //Inicializa flags de interrupções
97
98      RCONbits.IPEN = 1;          //Habilita prioridade de interrupções
99      INTCONbits.GIE = 1;         //Habilita interrupções de alta prioridade
100     INTCONbits.PEIE = 1;        //Habilita interrupções de baixa prioridade
101
102     OpenPWM2(24);   //Tpwm = 25us => maxPWM = 100
103     SetDCPWM2(dutyPWM);
104
105     DisablePullups();
106     inicia_lcd();
107
108     while(1)    //Loop principal
109     {
110         while((PORTBbits.RB2 == 1) && (PORTBbits.RB3 == 1))
111         {
112             // Adicione aqui o que deve ser executado quando
113             // as teclas estiverem soltas.
114         }
115
116         Delay1KTCYx(20);    //Delay para debounce das teclas
117
118         if((PORTBbits.RB2 == 0) && (PORTBbits.RB3 == 1))
119         {   //RB2 pressionado
120             // Adicione aqui o que deve ser executado quando
121             // as tecla RB2 for acionada.
122
123         }
124         else if((PORTBbits.RB2 == 1) && (PORTBbits.RB3 == 0))
125         {   //RB3 pressionado
126             // Adicione aqui o que deve ser executado quando
127             // as tecla RB3 for acionada.
128         }
129
130         while(((PORTBbits.RB2 == 0) || (PORTBbits.RB3 == 0))){}
131         Delay1KTCYx(20);
132         while(((PORTBbits.RB2 == 0) || (PORTBbits.RB3 == 0))){}
133     }
134
135 }
136
137 //-----------------------------------------------------------------
138 //Função de inicialização do LCD
139 void inicia_lcd(void)
140 {
141     escreve_comando(0x38);
142     Delay1KTCYx(3);
143     escreve_comando(0x38);
144     escreve_comando(0x06);
145     escreve_comando(0x0C);
146     escreve_comando(0x01);
147     Delay1KTCYx(1);
148     escreve_comando(0x80);  //Posiciona cursor
149
150     escreve_dado('X');
151 }
152
```

```
153 //---------------------------------------------------------------
154 //Envia comando para o LCD
155 void escreve_comando(char c)
156 {
157     PORTEbits.RE0=0;
158     PORTD=c;
159     Delay10TCYx(1);
160     PORTEbits.RE1=1;
161     Delay1TCY();
162     Delay1TCY();
163     Delay1TCY();
164     PORTEbits.RE1=0;
165     Delay1KTCYx(1);
166 }
167
168 //---------------------------------------------------------------
169 //Escrita de uma dado no LCD
170 void escreve_dado(char d)
171 {
172 PORTEbits.RE0=1;
173 PORTD=d;
174 Delay10TCYx(1);
175 PORTEbits.RE1=1;
176 Delay1TCY();
177 Delay1TCY();
178 Delay1TCY();
179 PORTEbits.RE1=0;
180 Delay1KTCYx(1);
181 PORTEbits.RE0=1;
182 }
```