C:\Documents and Settings\rromano\Meus documentos\EEM\Disciplines\ECA207\1oBim\encoder\sw.c

```
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
8   #pragma config WDT = OFF, LVP = OFF, OSC = XT, PWRT = ON, BOR = ON, BORV = 42
9
10  //Protótipos de funções
11  void ISR_High_Priority(void);
12  void ISR_Low_Priority(void);
13  void refresh_lcd(int x);
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 flags = 0, ct = 0, div_freq_tmr0 = 20;
22  unsigned char decod[] = {'0','1','2','3','4','5','6','7','8','9'};
23  volatile unsigned int freq, medida = 0;
24
25  //-----------------------------------------------------------------
26  #pragma code vec_int_high_priority = 0x08
27  void vec_int_high_priority(void)
28  { _asm GOTO ISR_High_Priority _endasm }
29
30  #pragma code vec_int_low_priority = 0x18
31  void vec_int_low_priority(void)
32  { _asm GOTO ISR_Low_Priority _endasm }
33
34  #pragma code
35  void ISR_High_Priority(void)
36  {
37      INTCONbits.TMR0IF = 0;
38      WriteTimer0(15536);
39
40      if (!(div_freq_tmr0--))
41      {
42          div_freq_tmr0 = 20;
43          freq = medida;
44          medida = 0;             //Zera var. medida para o próximo ciclo
45          flags = (flags | 0b00000001);
46      }
47      _asm
48      RETFIE 1
49      _endasm
50  }
51
52  //-----------------------------------------------------------------
53  void ISR_Low_Priority(void)
54  {
55      _asm
56      MOVFF   WREG,W_temp
57      MOVFF   STATUS, STATUS_temp
58      MOVFF   BSR, BSR_temp
59      _endasm
60      PIR1bits.CCP1IF = 0;
61      medida++;
62      _asm
63      MOVFF   W_temp, WREG
64      MOVFF   STATUS_temp, STATUS
65      MOVFF   BSR_temp, BSR
66      RETFIE 0
67      _endasm
68  }
69
70  //-----------------------------------------------------------------
71  void main(void)
72  {
73      TRISA = 0xFF; TRISB = 0b00001111;
74      TRISC = 0b10111001; TRISD = 0x00; TRISE = 0x00;
75
76      PORTB = 0x00; PORTD = 0x00;
```

```
77
78      OpenTimer1(TIMER_INT_OFF & T1_16BIT_RW & T1_SOURCE_EXT
79              & T1_PS_1_1 & T1_OSC1EN_OFF & T1_SOURCE_CCP);
80      WriteTimer1(0);
81
82      OpenCompare1(COM_INT_ON & COM_TRIG_SEVNT, 7);   //CCP1 como COMPARE e disparo de
83                                                      // TMR1 = CCP1H/L
84                                                      // e seta valor de comparação
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 1:1
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     DisablePullups();
105
106     inicia_lcd();
107
108     while(1)    //Loop principal
109     {
110         while((PORTBbits.RB2 == 1) && (PORTBbits.RB3 == 1))
111         {
112             if(flags & 0b00000001)
113             {
114                 int freq_disp = freq;
115                 refresh_lcd(freq_disp);
116                 flags = flags & 0b11111110;
117             }
118         }
119         Delay1KTCYx(20);    //Delay para debounce das teclas
120
121         if((PORTBbits.RB2 == 0) && (PORTBbits.RB3 == 1))
122         {   //RB2 pressionado
123             if(dutyPWM < 100)
124             {
125                 dutyPWM = dutyPWM + 10;
126                 escreve_comando(0x85 + dutyPWM/10);
127                 escreve_dado(0xFF);
128             }
129         }
130         else if((PORTBbits.RB2 == 1) && (PORTBbits.RB3 == 0))
131         {   //RB3 pressionado
132             if(dutyPWM)
133             {
134                 escreve_comando(0x85 + dutyPWM/10);
135                 escreve_dado(' ');
136                 dutyPWM = dutyPWM - 10;
137             }
138         }
139
140         SetDCPWM2(dutyPWM); //"Seta" novo duty-cycle
141         while(((PORTBbits.RB2 == 0) || (PORTBbits.RB3 == 0))){
142         Delay1KTCYx(20);
143         while(((PORTBbits.RB2 == 0) || (PORTBbits.RB3 == 0))){}
144         }
145     }
146 }
147
148 //-----------------------------------------------------------------
149 void refresh_lcd(int x)
150 {
151     unsigned short centena, dezena, unidade;
152     int aux;
```

```
153
154     escreve_comando(0xC6);
155     centena = ceil(x/100);
156     escreve_dado(decod[centena]);
157     aux = fmod(x,100);
158     dezena = ceil(aux/10);
159     escreve_dado(decod[dezena]);
160     unidade = fmod(aux,10);
161     escreve_dado(decod[unidade]);
162 }
163
164 //---------------------------------------------------------------------
165 //Função de inicialização do LCD
166 void inicia_lcd(void)
167 {
168     escreve_comando(0x38);
169     Delay1KTCYx(3);
170     escreve_comando(0x38);
171     escreve_comando(0x06);
172     escreve_comando(0x0C);
173     escreve_comando(0x01);
174     Delay1KTCYx(1);
175     escreve_comando(0x80);  //Posiciona cursor
176
177     escreve_dado(' ');
178     escreve_dado('P');
179     escreve_dado('W');
180     escreve_dado('M');
181     escreve_dado(':');
182     escreve_dado(' ');
183
184     escreve_comando(0xC0);
185     escreve_dado('R');
186     escreve_dado('o');
187     escreve_dado('t');
188     escreve_dado('.');
189     escreve_dado(':');
190     escreve_dado(' ');
191     escreve_dado(' ');
192     escreve_dado(' ');
193     escreve_dado(' ');
194     escreve_dado('H');
195     escreve_dado('z');
196 }
197
198 //---------------------------------------------------------------------
199 //Envia comando para o LCD
200 void escreve_comando(char c)
201 {
202     PORTEbits.RE0=0;
203     PORTD=c;
204     Delay10TCYx(1);
205     PORTEbits.RE1=1;
206     Delay1TCY();
207     Delay1TCY();
208     Delay1TCY();
209     PORTEbits.RE1=0;
210     Delay1KTCYx(1);
211 }
212
213 //---------------------------------------------------------------------
214 //Escrita de uma dado no LCD
215 void escreve_dado(char d)
216 {
217 PORTEbits.RE0=1;
218 PORTD=d;
219 Delay10TCYx(1);
220 PORTEbits.RE1=1;
221 Delay1TCY();
222 Delay1TCY();
223 Delay1TCY();
224 PORTEbits.RE1=0;
225 Delay1KTCYx(1);
226 PORTEbits.RE0=1;
227 }
```