```
/*

  ### Dimmer microcontrolado com PIC ###

  uC: PIC 16f876A
  Ciclo de máquina: 200ns
  Plataforma de desenvolvimento do software: MikroC PRO for PIC v.4.15.0.0

  Data de inicio: 12 Outubro de 2019
  Última atualização: 05 Outubro de 2019

  OBS: Não é recomendado fazer a chamada de uma função dentro da pilha de
  interrupção.

*/

// ###############################################################################
// --- Mapeamento do Display LCD ---
// LCD module connections
sbit LCD_RS at RC2_bit;
sbit LCD_EN at RC3_bit;
sbit LCD_D7 at RC7_bit;
sbit LCD_D6 at RC6_bit;
sbit LCD_D5 at RC5_bit;
sbit LCD_D4 at RC4_bit;

// Pin direction
sbit LCD_RS_Direction at TRISC2_bit;
sbit LCD_EN_Direction at TRISC3_bit;
sbit LCD_D7_Direction at TRISC7_bit;
sbit LCD_D6_Direction at TRISC6_bit;
sbit LCD_D5_Direction at TRISC5_bit;
sbit LCD_D4_Direction at TRISC4_bit;

// ###############################################################################
// --- Flags ---
#define inc      flagsA.B0       // Flag do botão de incremento
#define dec      flagsA.B1       // Flag do botão de decremento
#define GO       flagsA.B2       // Flag de entrada nos menus
#define Func_up  flagsA.B3       // Flag de incremento
#define ClearLCD flagsA.B4       // Flag de limpeza do LCD
#define Fade_M   flagsA.B5       // Flag do modo Fade
#define Flash_M  flagsA.B6       // Flag do modo Flash
#define Timer_M  flagsA.B7       // Flag do modo Timer
#define Func_dw  flagsB.B0       // Flag para decremento
#define _fFS     flagsB.B1       // Flag inicio fade
#define _fTM     flagsB.B2       // Flag timer mais
#define _fTm     flagsB.B3       // Flag timer menos
#define _fIF     flagsB.B4       // Flag inicio fade
#define _fFO     flagsB.B5       // Flag usada p/ indicar que está no modo flash
#define entTim   flagsB.B6       // Flag de sinalização de enter dentro de timer
  (Usada para acionar o timer)
#define changeSt flagsB.B7       // Flag p/ mudança do estado da Lampada no Flash
#define flinc    flagsC.B0       // Flag de incremento tempo flash
#define fldec    flagsC.B1       // Flag de decremento tempo flash
#define statusF  flagsC.B2       // Flag de status do Flash
```

```
#define ModoFade flagsC.B3       // Flag de status do Fade
#define atualiza flagsC.B3       // Flag de atualização
#define hab_f    flagsC.B4       // Flag que habilita o flash
#define ldr1     flagsC.B5
#define ldr2     flagsC.B6
#define leitura  flagsC.B7


// ###############################################################################
// --- Mapeamento de Hardware ---
#define nmenus   5
#define butinc   RB7_bit    // Pino 28
#define butdec   RB6_bit    // Pino 27
#define ON       RB5_bit    // Pino 26
#define OFF      RB4_bit    // Pino 25
#define ENTER    RB3_bit    // Pino 24
#define BACK     RB2_bit    // Pino 23
#define triac    RB1_bit    // Pino 22
#define Desl     32000      // ### Valores para ligar e desligar a lâmpada ###
#define Lig      62200      // valor display = (T_ajust - Desl)/((Lig - Desl)/100)


// ###############################################################################
// --- Variáveis Globais ---
char auxcont0 = 0x00,       // Variável auxiliar de contagem Timer0
     sel = 0x01,            // Variável para seleção dos menus
     estado = 0x00,         // Variável de estado Fade
     _butcontp = 0x00,      // Variável usada na rotina de incremento rápido
     _butcontm = 0x00,      // Variável usada na rotina de decremento rápido
     _1seg,                 // Variável usada para contar 01 segundos (TMR0)
     contseg,               // Variável usada para contar o tempo no modo Timer
     ContFlash,             // Variável usada para contar o tempo no modo Flash
     auxflash,              // Variável de decremento flash
     comparadora,           // Variave que armazena tempo flash
     step,                  // Armazena o número de passos modo fade
     flagsA, flagsB,        // Registradores de Flags
     flagsC;                // (Mapeados acima)

unsigned int  ajust_tmr1 = Desl,        // Valor da lampada apagada
              ajust_fade = Desl;
         int  LDR_Read, setpoint;       // Variaveis usadas no modo fade
// ###############################################################################
// --- Vetor de Interrupção ---
void interrupt ()                       // Função de interrupção
{
     // --- Interrupção externa ---
     if (INTF_bit)                              // Testa se houve interrupção externa
     {
      T1CON.F0 = 0x01;                          // Habilita a contagem do TMR1
      triac = 0x00;                             // Desliga o pulso
      INTF_bit = 0x00;                          // Limpa a flag de interrupção
        externa
     } // end INTF if

     // --- Timer 1 ---
     if (TMR1IF_bit)                            // Testa se houve estouro do TMR1
```

```
    {

    triac = 0x01;                           // Liga o pulso no pino triac

    TMR1L = (ajust_tmr1 & 0x00FF);          // Recarrega para contagem
    TMR1H = (char)(ajust_tmr1 >> 8);

    T1CON.F0 = 0x00;                        // Desliga o Timer 1 até a prox.
      interrupção externa
    TMR1IF_bit = 0x00;                      // Limpa a flag de overflow TMR1
    } // end TMR1IF if

    // --- Timer 0 ---
    if (TMR0IF_bit)                         // Testa se houve estouro do timer0
    {
       // === Base de tempo de 10ms ===
       auxcont0++;                          // Incrementa a variável auxiliar de
         contagem do TMR0




       // === Base de tempo 100 ms ===
          leitura = 1;
          if (auxcont0 == 10)
          {
             _1seg++;                       // Incrementa a variavel de contagem
               de segundo

             auxcont0 = 0x00;               // Zera a variável de contagem

       // === Rotina para implementação do botão com incremento inteligente ===
       // Para a parte de controle...
             if (!_fIF)                     // Teste para aproveitar a variável
               de
             {                              // contagem rápida para modo controle
                e fade

                if (Func_up)    _butcontp++; // Incrementa a variável se flag
                  setada (botão inc pressionado)
                if (Func_dw)    _butcontm++; // Incrementa a variável se flag
                  setada (botão dec pressionado)

                if (_butcontp > 15)         // Verifica se a variável de contagem
                   chegou a quinze sem o botão ser solto por 1,5s
                {
                   _butcontp = 15;          // Garante que não haja overflow da
                    variável
                   ajust_tmr1 += 960;       // Incrementa a variavel de ajuste
                    rapidamente
                } // ent _butcontp if

                if (_butcontm > 15)         // Verifica se a variável de contagem
                   chegou a quinze sem o botão ser solto por 1,5s
                {
                   _butcontm = 15;          // Garante que não haja overflow da
```

```
                    variável
                   ajust_tmr1 -= 960;        // Decrementa a variavel de ajuste
                    rapidamente
                  } // ent _butcontm if
               } // end !Fade_M if
         // === Fim da rotina do botão com incremento inteligente ===

         // === Rotina para implementação do botão com incremento inteligente ===
         // Para a parte do fade...
               if (_fIF)                       // Se esta flag está setada,
               {                               // o programa principal está em fade

                  if (Func_up)    _butcontp++; // Exatamente a mesma explicação
                  if (Func_dw)    _butcontm++; // da parte logo acima desta

                  if (_butcontp > 15)
                  {
                     _butcontp = 15;
                     ajust_fade += 960;
                  } // ent _butcontp if

                  if (_butcontm > 15)
                  {
                     _butcontm = 15;
                     ajust_fade -= 960;
                  } // ent _butcontm if
               } // end !Fade_M if
         // === Fim da rotina do botão com incremento Fade ===

         // === Teste do pressionar de botão de entrada e saída dos menus ===
               if (!ENTER)                     // Botão enter pressionado? Sim...
               {
                GO = 0x01;                     // Seta a flag de entrada do menu

                if (entTim)      Timer_M = 0x01;       // Verifica se o programa
                  está na função timer
                else             Timer_M = 0x00;       // caso esteja e enter seja
                  pressionado seta a flag Timer_M

                if (_fFO)
                {
                        Flash_M = 0x01;                // Verifica se o programa
                      está na função flash
                        statusF = 0x01;
                }
                else             Flash_M = 0x00;       // caso esteja e enter seja
                  pressionado seta a flag Flash_M

                if (_fFS && estado == 0)        estado = 1;

                if (Flash_M && _fFO && comparadora > 0)  // Lógica de acionamento
                  flash
                {
                   auxflash = comparadora;
                   hab_f = 1;
                }
```

```
            } // end if ENTER

            else if (!BACK)                         // Botão se saida pressionado?
               Sim...
            {
             GO = 0x00;                             // Limpa a flag de entrada do
               menu
             _fFS = 0x00;
             ajust_tmr1 = 32000;                    // Garante a lampada apagada
               no menu
             hab_f = 0;                             // Desliga o modo flash na
               saída
             if (sel == 5 && setpoint > 0)           ClearLCD = 1;
             if (sel == 2 && estado == 1)            ClearLCD = 1;
            } // end if BACK

      // === Teste do pressionar de botões de ON e OFF ===
            if (!ON)
            {
                ajust_tmr1 = Lig;
                GO = 0x00;
            }
            else if (!OFF)
            {
                 ajust_tmr1 = Desl;
                 GO = 0x00;
            }

      // === Do efeito fade 100ms ===
                //if (estado == 2)
                atualiza = 0x01;


      // === Base de tempo 1 segundo ===
            if (_1seg == 10)                // Testa se já passaram 1 segundo
            {
                _1seg = 0x00;               // Zera o valor da contagem

                if (Timer_M)                // Testa se a flag que indica incio
                  da contagem
                {
                   ClearLCD = 0x01;         // Reutiliza a flag para ligar e
                   apagar a lâmpada
                   contseg--;               // Decrementa a variável de contagem
                   if (contseg == 0x00)     // Desliga a lampada...
                   {
                    Timer_M = 0x00;
                    ClearLCD = 0x00;
                   }
                }
                if (hab_f)                  // Lógica modo flash
                {
                   auxflash--;
                   if (auxflash == 0)
                   {
```

```
                        auxflash = comparadora;
                        changeSt = !changeSt;
                    }
                }
            } // end _1seg if
        } // end base 100 ms

        TMR0 = 0x3C;                        // Carrega o timer0 com o valor 60
          novamente
        TMR0IF_bit = 0x00;                  // Limpa a flag do timer0 após o fim
          do processamento da interrupção
} // end TMR0IF if


} // end interrupt

// ###############################################################################
// --- Declaração de Funções Auxiliares ---
void registradores ();                      // Função que configurará os
  regitradores
void testabotoes ();                        // Função que testa os botoes
void controlight ();                        // Função que controla a luminosidade
void fade ();                               // Função responsável pelo fade
void flash ();                              // Função responsável pelo flash
void timer ();                              // Função responsável pelo timer
void calcDisplay ();                        // Função de envio de dados LCD
void ldr ();                                // Função responsável pelo
                                            // tratamento do ldr (Extra!!)

// ###############################################################################
// --- Função Principal ---
void main()
{
     GO       = 0x00;                       // Inicializa todas as flags
     inc      = 0x00;                       // mais importantes limpas...
     dec      = 0x00;
     ClearLCD = 0x00;
     Func_up  = 0x00;
     Func_dw  = 0x00;
     Timer_M  = 0x00;
     entTim   = 0x00;
     Fade_M   = 0x00;
     Flash_M  = 0x00;

     registradores ();                      // Faz chamada da função que
       configura os registradores

     Lcd_Init();                            // Inicia o display LCD
     Lcd_Cmd (_LCD_CLEAR);                  // Limpa o display LCD
     Lcd_Cmd(_LCD_CURSOR_OFF);              // Desliga o cursor do display LCD
     Lcd_Out (1, 6,"DIMMER");               // Imprime mensagem de inicialização
       no display LCD
     Lcd_Out (2, 2,"MICROCONTROLADO");      // (Mensagem generica)

     delay_ms (1000);                       // Aguarda 1 segundo com a mensagem
       na tela
```

```
     Lcd_Cmd (_LCD_CLEAR);

     while (1)
    {
        testabotoes();                          // Chama a função que testa os botões
           de incremento e decremento

        switch (sel)                            // Entra no menu de funcionalidades
          do dimmer
        {
         case 0x01:                             // Menu Controle

              if (ClearLCD)
              {
                 LCD_Cmd (_LCD_CLEAR);
                 ClearLCD = 0x00;
              } // end if Clear LCD

              Lcd_Chr (1,1, '<');
              Lcd_Chr (1,16, '>');
              Lcd_Chr (1,3, 'C');
              Lcd_Chr_Cp ('O');
              Lcd_Chr_Cp ('N');
              Lcd_Chr_Cp ('T');
              Lcd_Chr_Cp ('R');
              Lcd_Chr_Cp ('O');
              Lcd_Chr_Cp ('L');
              Lcd_Chr_Cp ('E');
              Lcd_Chr (1,11, ' ');
              Lcd_Out (2,1, "                ");

              _fIF = 0x00;
              controlight ();
              break;

         case 0x02:                             // Menu Fade

              if (ClearLCD)
              {
                 LCD_Cmd (_LCD_CLEAR);
                 ClearLCD = 0x00;
              } // end if Clear LCD

              Lcd_Chr (1,1, '<');
              Lcd_Chr (1,16, '>');
              Lcd_Chr (1,3, 'F');
              Lcd_Chr_Cp ('A');
              Lcd_Chr_Cp ('D');
              Lcd_Chr_Cp ('E');
              Lcd_Chr (1,7, ' ');
              Lcd_Out (2,1, "                ");


              estado = 0x00;                    // Reseta o modo fade
              Fade_M        = 0x00;
              _fIF          = 0x01;             // Flag que indica o modo fade
```

```
            _fFS          = 0x00;
            ajust_fade    = 0x00;
            atualiza      = 0x00;            // Variavel que indica contagem do
              tmr0
            ajust_fade    = Desl;

            contseg       = 0x00;

            fade ();
            break;

        case 0x03:                           // Menu Flash

            if (ClearLCD)
            {
               LCD_Cmd (_LCD_CLEAR);
               ClearLCD = 0x00;
            } // end if Clear LCD

            Lcd_Chr (1,1, '<');
            Lcd_Chr (1,16, '>');
            Lcd_Chr (1,3, 'F');
            Lcd_Chr_Cp ('L');
            Lcd_Chr_Cp ('A');
            Lcd_Chr_Cp ('S');
            Lcd_Chr_Cp ('H');
            Lcd_Out (2,1, "                ");

            comparadora = 0;                 // Reseta o modo
            changeSt = 0;
            hab_f    = 0;
            Flash_M = 0;
            _fFO     = 0;
            flash ();                        // Faz a chamada da função flash
            break;

        case 0x04:                           // Menu Timer

            if (ClearLCD)
            {
               LCD_Cmd (_LCD_CLEAR);
               ClearLCD = 0x00;
            } // end if Clear LCD

            Lcd_Chr (1,1, '<');
            Lcd_Chr (1,16, '>');
            Lcd_Chr (1,3, 'T');
            Lcd_Chr_Cp ('I');
            Lcd_Chr_Cp ('M');
            Lcd_Chr_Cp ('E');
            Lcd_Chr_Cp ('R');
            Lcd_Chr_Cp (' ');
            Lcd_Out (2,1, "                ");

            contseg = 0x00;                  // Garante que a contagem sempre
              inicia em zero
```

```
              entTim  = 0x00;               // Garante que a flag entTim esteja
                limpa ao entrar na função timer
              timer ();
              break;

         case 0x05:                         // Menu Sinc

              if (ClearLCD)
              {
                 LCD_Cmd (_LCD_CLEAR);
                 ClearLCD = 0x00;
              } // end if Clear LCD

              Lcd_Chr (1,1, '<');
              Lcd_Chr (1,16, '>');
              Lcd_Chr (1,4, 'L');
              Lcd_Chr_Cp ('D');
              Lcd_Chr_Cp ('R');
              Lcd_Chr_Cp (' ');
              Lcd_Chr_Cp (' ');
              Lcd_Out (2,1, "                ");

              setpoint = 0;
              ldr ();
              break;

        } // end switch case sel

    } // end while
  } // end main

// ###############################################################################
// --- Desenvolvimento de Funções Auxiliares ---
void registradores ()
{

// --- Configuração TMR0 e External Interrupt ---
     INTCON = 0xF0;     // 1111 0000
     // Habilita as interrupções globais para configuração (INTCON.F7)
     // Habilita as interrupções por periféricos (INTCON.F6)
     // Habilita interrupção do timer 0 (INTCON.F5)
     // Habilita interrupção externa no pino RB0 (INTCON.F4)
     // Desabilita interrução por mudança do PORTB (INTCON.F3)
     // Limpa a flag de interrupção do Timer0 (INTCON.F2)
     // Limpa a flag de interrupção externa (INTCON.F1)
     // Limpa a flag de mudança do PORTB (INTCON.F0)

     OPTION_REG = 0x87; // 1100 0111 | 1000 0111 (0x87 - Testar se é melhor com
       borda de subida ou descida)
     // Desabilita os PULLUPS do PORTB (OPTION_REG.F7)
     // Habilita interrupção Externa por borda de descida (OPTION_REG.F6)
     // Associa o clock do timer0 ao ciclo de máquina (OPTION_REG.F5)
     // Associa o preescaler ao timer0 (OPTION_REG.F3)
     // Configura o preescaler em 1:256 (OPTION_REG <F2:F0>)

// --- Configuração TMR1 ---
```

```
     T1CON = 0x00;      // 0000 0000
     // Não são implementados (T1CON <F7:F6>)
     // Habilita o prescale em 1:1 (T1CON <F5:F4>)
     // Desabilita o oscilador do TMR1 (T1CON.F3)
     // Bit de controle de sincronia do TMR1, don't care pois T1CON.F1 é setado
       (T1CON.F2)
     // Config. o incremento do TMR pelo ciclo de máquina (T1CON.F1)
     // Inicia o TMR1 desligado, deve ser ligado após interrupção externa
       (T1CON.F0)

     PIE1 = PIE1 | 0x01;// 0000 0001
     // Habilita a interrupção do TMR1 por overflow

     PIR1 = PIR1 & 0xFE;// 1111 1110
     // Limpa a flag de overflow do TMR1 (garantia!)

     TMR1H = 0x7D;      // Configura o TMR1 para contagem do periodo desligado
     TMR1L = 0x00;

// --- Configuração dos demais periféricos ----
     CMCON = 0x07;      // Desabilita os comparadores
     CVREN_bit = 0;
     CVROE_bit = 0;

     TMR0 = 0x3C;       // Carrega o timer0 com o valor 60 inicialmente

     ADON_bit = 0x00;   // Desabilita o modulo de conversão AD
     ADCON1 = 0x0E;     // Configura os pinos do PORTA como digitais

     TRISA = 0b11111101;
     TRISB = 0b11111101;
     TRISC = 0xF0;      // 1111 0000


     PORTA = 0x00;      // Inicia o PORTB em LOW
     PORTB = 0xFF;      // Inicia o PORTB em HIGH
     PORTC = 0x00;      // Inicia o PORTC em LOW

} // end registradores

void testabotoes ()
{
        if (!butinc)        inc = 0x01;      // Se botão de mais pressionado, seta
          a flag inc
        if (!butdec)        dec = 0x01;      // Se botão de menos pressionado, seta
           a flag dec

        if (butinc && inc)                   // Botão+ solto e flag inc setada?
          Sim...
        {
                inc = 0x00;                  // Limpa a flag
                sel++;                       // Incrementa a seleção de menus
                ClearLCD = 0x01;
        } // end but+ && inc

        if (butdec && dec)                   // Botão- solto e flag dec setada?
```

```
          Sim...
        {
                dec = 0x00;                     // Limpa a flag
                sel--;                          // Decrementa a seleção de menus
                ClearLCD = 0x01;
        } // end if but- && dec

        if (sel > nmenus)       sel = 0x01;   // Cria o efeito de menu ciclico...
        if (sel < 0x01)         sel = nmenus;

} // end testabotoes

void controlight ()
/*
  1. Controle de luminosidade: A luminosidade deverá aumentar ou diminuir por
  meio do acionamento de dois botões (+ e -). Deve ser possível escolher no
  mínimo entre 5 níveis diferentes de luminosidade, que deverão ser indicados
  no display e visíveis pela luminosidade da lâmpada. Deve existir, também,
  um botão ON e um botão OFF, que ligam (100% de luminosidade) ou
  desligam (0% de luminosidade) instantaneamente a lâmpada.
*/
{

     while (GO)
     {

      Lcd_Chr (1,1, ' ');
      Lcd_Chr (1,16, ' ');
      Lcd_Chr (1,11,':');

        if (!butinc)        Func_up = 0x01;     // Se botão de mais pressionado,
          seta a flag Func_up
        if (!butdec)        Func_dw = 0x01;     // Se botão de menos pressionado,
          seta a flag Func_dw

        if (butinc && Func_up)                  // Botão+ solto e flag inc setada?
          Sim...
        {
                Func_up = 0x00;                 // Limpa a flag
                _butcontp = 0x00;               // Limpa a variavel de contagem do
                  incremento inteligente
                _butcontm = 0x00;               // Redundancia para segurança do
                  código
                ajust_tmr1 += 302;              // Incrementa o valor de ajuste
        }

        if (butdec && Func_dw)                  // Botão- solto e flag Func_dw
          setada? Sim...
        {
                Func_dw = 0x00;                 // Limpa a flag
                _butcontm = 0x00;               // Limpa a variavel de contagem do
                  incremento inteligente
                _butcontp = 0x00;               // Redundancia para segurança do
                  código
                ajust_tmr1 -= 302;              // Decrementa a variável de ajuste
        }
```

```
        if (ajust_tmr1 > Lig)        ajust_tmr1 = 62200;
        if (ajust_tmr1 < Desl)       ajust_tmr1 = 32000;

        calcDisplay ();                         // Faz chamada da função que
          calcula valor do display
     } // end while
} // end controlight

void fade ()
/*
  2. Efeito fade: Este modo, que deve ser escolhido pelo usuário por meio de
  um botão e indicado que está ativo através de um led (ou via display), deve
  apagar ou acender (até o nível de luminosidade já programado) de forma
  gradual e ininterrupta.
*/
{
  while (GO)
  {

        switch (estado)
        {
               case 0:


                     Lcd_Chr (1,3, 'F');
                     Lcd_Chr_Cp ('A');
                     Lcd_Chr_Cp ('D');
                     Lcd_Chr_Cp ('E');
                     Lcd_Chr (1,1, ' ');
                     Lcd_Chr (1,16, ' ');
                     Lcd_Chr (1, 7,':');
                     //Lcd_Out (2,1, "                ");
                     Lcd_Chr (2,1, ' ');
                     Lcd_Chr (2,2, ' ');
                     Lcd_Chr (2,3, ' ');
                     Lcd_Chr (2,8, ' ');
                     Lcd_Chr (2,9, ' ');

                     if (!butinc)        Func_up = 0x01;     // Se botão de mais
                      pressionado, seta a flag Func_up
                     if (!butdec)        Func_dw = 0x01;     // Se botão de menos
                      pressionado, seta a flag Func_dw

                     if (butinc && Func_up)                  // Botão+ solto e flag
                      inc setada? Sim...
                     {
                        Func_up = 0x00;                 // Limpa a flag
                        _butcontp = 0x00;               // Limpa a variavel de
                      contagem do incremento inteligente
                        _butcontm = 0x00;               // Redundancia para
                      segurança do código
                        _fFS      = 0x01;               // Garante o inicio com um
                      pressionar de botão ao menos
                        ajust_fade += 302;              // Incrementa o valor de
                      ajuste do fade
```

```
                }

                if (butdec && Func_dw)              // Botão- solto e flag      ↪
                 Func_dw setada? Sim...
                {
                   Func_dw = 0x00;                  // Limpa a flag
                   _butcontm = 0x00;                // Limpa a variavel de      ↪
                 contagem do incremento inteligente
                   _butcontp = 0x00;                // Redundancia para         ↪
                 segurança do código
                   _fFS      = 0x01;
                   ajust_fade -= 302;               // Decrementa a variável de ↪
                 ajuste do fade
                }

                if (ajust_fade > Lig)        ajust_fade = Lig;
                if (ajust_fade < Desl)       ajust_fade = Desl;
                break;

            case 1:                                 // Neste caso o botão de enter ↪
              foi pressionado

                Lcd_Chr (1, 1, 'A');
                Lcd_Chr_Cp ('C');
                Lcd_Chr_Cp ('E');
                Lcd_Chr_Cp ('N');
                Lcd_Chr_Cp ('D');
                Lcd_Chr_Cp ('E');
                Lcd_Chr_Cp ('R');
                Lcd_Chr_Cp (':');
                Lcd_Chr_Cp ('B');
                Lcd_Chr_Cp ('+');

                Lcd_Chr (2, 1, 'A');
                Lcd_Chr_Cp ('P');
                Lcd_Chr_Cp ('A');
                Lcd_Chr_Cp ('G');
                Lcd_Chr_Cp ('A');
                Lcd_Chr_Cp ('R');
                Lcd_Chr_Cp (':');
                Lcd_Chr_Cp ('B');
                Lcd_Chr_Cp ('-');

                if (!butinc)     inc = 0x01;  // De desligado -> ligado
                if (!butdec)     dec = 0x01;  // De ligado -> desligado

                if (butinc && inc)            // Modo que liga
                {
                   inc = 0x00;
                   ModoFade = 1;
                   ajust_tmr1 = Desl;
                   step = ((ajust_fade - Desl)/302);
                   estado = 2;
                   Lcd_Cmd (_LCD_CLEAR);
                   Lcd_Chr (1,3, 'F');
                   Lcd_Chr_Cp ('A');
```

```
                    Lcd_Chr_Cp ('D');
                    Lcd_Chr_Cp ('I');
                    Lcd_Chr_Cp ('N');
                    Lcd_Chr_Cp ('G');
                    Lcd_Chr_Cp ('.');
                    Lcd_Chr_Cp ('.');
                    Lcd_Chr_Cp ('.');
                }

                if (butdec && dec)            // Modo que desliga
                {
                    dec = 0x00;
                    ModoFade = 1;
                    estado = 3;
                    ajust_tmr1 = Lig;
                    step = 100 -((ajust_fade - Desl)/302);
                    Lcd_Cmd (_LCD_CLEAR);
                    Lcd_Cmd (_LCD_CLEAR);
                    Lcd_Chr (1,3, 'F');
                    Lcd_Chr_Cp ('A');
                    Lcd_Chr_Cp ('D');
                    Lcd_Chr_Cp ('I');
                    Lcd_Chr_Cp ('N');
                    Lcd_Chr_Cp ('G');
                    Lcd_Chr_Cp ('.');
                    Lcd_Chr_Cp ('.');
                    Lcd_Chr_Cp ('.');
                }
                break;

            case 2:

                 if (atualiza)           // Ligando (Desl >> ajust_fade)
                 {
                  ajust_tmr1 += 302;
                  step--;
                  atualiza = 0;
                 } // end if atualiza
                //} // end if ModoFade

                if (step == 1)
                {
                 Lcd_Cmd (_LCD_CLEAR);
                 Lcd_Chr (1,3, 'F');
                 Lcd_Chr_Cp ('A');
                 Lcd_Chr_Cp ('D');
                 Lcd_Chr_Cp ('E');
                 Lcd_Chr (1,7, ':');
                 estado = 0;
                }
                calcDisplay();
                break;

            case 3:

                 if (atualiza)    // Desligado (Lig >> ajust_fade)
```

```
                    {
                     ajust_tmr1 = ajust_tmr1 - 302;
                     step--;
                     atualiza = 0;
                    } // end if atualiza
                   //} // end else if

                   if (step == 1)
                   {
                    Lcd_Cmd (_LCD_CLEAR);
                    Lcd_Chr (1,3, 'F');
                    Lcd_Chr_Cp ('A');
                    Lcd_Chr_Cp ('D');
                    Lcd_Chr_Cp ('E');
                    Lcd_Chr (1,7, ':');
                    estado = 0;
                   }

                   calcDisplay();
                   break;

        } // end switch estado

        if (estado != 1)   calcDisplay ();


   }// end while
} // end fade

void flash ()
/*
  3. Efeito flash: No efeito flash, quando acionado, a lâmpada deve piscar
  ininterruptamente com um tempo selecionado. Deve ser considerada uma faixa
  selecionável entre 1 e 5 segundos.
*/
{
        while (GO)
        {
              Flash_M = 0x01;

              Lcd_Chr (1,1, ' ');
              Lcd_Chr (1,16, ' ');
              Lcd_Chr (1, 8, ':');

              if (!butinc)   flinc = 0x01;
              if (!butdec)   fldec = 0x01;

              if (butinc && flinc)                  // Botão+ solto e flag flinc
                setada? Sim...
              {
                flinc = 0x00;
                _fFO = 0x01;
                comparadora++;
                if (comparadora > 5)               comparadora = 5;
              }
```

```
              if (butdec && fldec)                    // Botão- solto e flag fldec
                setada? Sim...
              {
                fldec = 0x00;
                _fFO = 0x01;
                comparadora--;
                if (comparadora < 0)                comparadora = 0;
              }

             // Rever p/ sexta
              if (changeSt)       ajust_tmr1 = Lig;
              else if (!changeSt) ajust_tmr1 = Desl;

              calcDisplay();

        } // end while
        _fFO = 0x00;
} // end flash

void timer ()
/*
  4. Modo timer: O usuário deve inserir o tempo, em segundos, via botões em que
     a lâmpada deverá ficar ligada. Deve ser considerada uma faixa selecionável
     entre 5 e 60 segundos, após esse tempo a mesma se apaga.
*/
{
     while (GO)
     {


              Lcd_Chr (1,1, ' ');
              Lcd_Chr (1,16, ' ');
              Lcd_Chr (1, 8, ':');
              Lcd_Chr (2, 7, 'S');
              Lcd_Chr_Cp ('e');
              Lcd_Chr_Cp ('g');

              if (!butinc)     _fTM = 0x01;   // Botão pressionado? Seta a flag

              if (!butdec)     _fTm = 0x01;   // Botão pressionado? Seta a flag


              if (butinc && _fTM && !Timer_M)             // Botão solto e flag
                setada?
              {
                 _fTM = 0x00;                 // Limpa a flag
                 contseg+= 5;                 // Incrementa o tempo de contagem
                 entTim  = 0x01;
                 //_StartTimer = 0x00;
              }

              if (butdec && _fTm)             // Botão solto e flag setada?
              {
                 _fTm = 0x00;                 // Limpa a flag
                 contseg--;                   // Decrementa o tempo de contagem
                 entTim = 0x01;
```

```
                //_StartTimer = 0x00;
             }

             if (contseg < 0x00)            contseg = 0x05; // Prende o valor
             else if (contseg > 0x3C)       contseg = 0x3C; // de contagem
                                                            // no intervalo
                                                            // [5,60]

             calcDisplay();

             if (ClearLCD)                    ajust_tmr1 = 59000;     //65530; //
                Inverti os valores
             else                             ajust_tmr1 = 0x7D00;

      } // end while
      Timer_M = 0x00;
} // end timer

void ldr ()
{
 ldr1 = 0;    // Flags dos botão
 ldr2 = 0;
             while (GO)
             {

              Lcd_Chr (1,1, ' ');
              Lcd_Chr (1,16, ' ');
              Lcd_Chr (1, 7, ':');

              Lcd_Chr (1, 12, 'A');
              Lcd_Chr_Cp ('J');
              Lcd_Chr_Cp ('T');
              Lcd_Chr_Cp (':');

             if (!butinc)   ldr1 = 1;
             if (!butdec)   ldr2 = 1;

             if (butinc && ldr1)
             {
               ldr1 = 0;
               setpoint += 100;
               if (setpoint >= 1000) setpoint = 1000;
             }

             if (butdec && ldr2)
             {
               ldr2 = 0;
               setpoint -= 100;
               if (setpoint <= 0)   setpoint = 0;
             }



              if (leitura)
              {
               leitura  = 0;
```

```
                LDR_Read = Adc_Read(0);
               }

               if (LDR_Read < setpoint)        ajust_tmr1 = Lig;
               else if (LDR_Read >= setpoint)  ajust_tmr1 = Desl;

                calcDisplay();
              } // end while
} // end ldr

void calcDisplay()
{
      char _unidade,
           _dezena,
           _centena,
           _milhar,
           set1, set2, set3, set4;
      unsigned Cont0_100;

           switch (sel)
           {
              case 0x01:
                   Cont0_100 = (ajust_tmr1 - Desl)/302;

                   _unidade = ((Cont0_100/1)%10);
                   _dezena  = ((Cont0_100/10)%10);
                   _centena = ((Cont0_100/100)%10);

                   Lcd_Chr (2,4, (_centena + 0x30));
                   Lcd_Chr_Cp ((_dezena + 0x30));
                   Lcd_Chr_Cp ((_unidade + 0x30));
                   break;


              case 0x02:
                   Cont0_100 = ((ajust_fade - Desl)/302);

                   _unidade = ((Cont0_100/1)%10);
                   _dezena  = ((Cont0_100/10)%10);
                   _centena = ((Cont0_100/100)%10);

                   Lcd_Chr (2,4, (_centena + 0x30));
                   Lcd_Chr_Cp ((_dezena + 0x30));
                   Lcd_Chr_Cp ((_unidade + 0x30));
                   break;

              case 0x03:
                   _unidade = ((comparadora/1)%10);
                   Lcd_Chr (2, 6,(_unidade+0x30));
                   break;

              case 0x04:

                  _unidade = ((contseg/1)%10);
                  _dezena = ((contseg/10)%10);
```

```
                Lcd_Chr (2,4, (_dezena + 0x30));
                Lcd_Chr_Cp ((_unidade + 0x30));
                break;

            case 0x05:
                 _unidade = ((LDR_Read/1)%10);
                 _dezena  = ((LDR_Read/10)%10);
                 _centena = ((LDR_Read/100)%10);
                 _milhar  = ((LDR_Read/1000)%10);

                 set1 = ((setpoint/1)%10);
                 set2 = ((setpoint/10)%10);
                 set3 = ((setpoint/100)%10);
                 set4 = ((setpoint/1000)%10);

                 Lcd_Chr (2,4, (_milhar + 0x30));
                 Lcd_Chr_Cp (_centena + 0x30);
                 Lcd_Chr_Cp (_dezena + 0x30);
                 Lcd_Chr_Cp (_unidade + 0x30);

                 Lcd_Chr (2, 12, (set4 + 0x30));
                 Lcd_Chr_Cp ((set3 + 0x30));
                 Lcd_Chr_Cp (set2 + 0x30);
                 Lcd_Chr_Cp (set1 + 0x30);


        } // end switch sel
} // end calcDisplay
```