```c
#include <hidef.h>      /* common defines and macros */
#include "derivative.h"      /* derivative-specific definitions */

//   _____             _             _   _____           _
//  /  __ \           | |           | | /  ___|         | |
//  | /  \/ ___  _ __ | |_ _ __ ___ | | \ `--. _   _ ___| |_ ___ _ __ ___
//  | |    / _ \| '_ \| __| '__/ _ \| |  `--. \ | | / __| __/ _ \ '_ ` _ \
//  | \__/\ (_) | | | | |_| | | (_) | | /\__/ / |_| \__ \ ||  __/ | | | | |
//   \____/\___/|_| |_|\__|_|  \___/|_| \____/ \__, |___/\__\___|_| |_| |_|
//                                              __/ |
//                                             |___/

//*****************************************************************
//*                                                               *
//*    ECNS-414                                                   *
//*    12 November 2020                                           *
//*    LAB-9                                                      *
//*                                                               *
//*    The purpose of this program is to construct                *
//*    and demonstrate a closed loop control system               *
//*    using pulse width modulation in C.                         *
//*                                                               *
//*                                                               *
//*  Created by: Brandon Empie                                    *
//*                                                               *
//*****************************************************************

int myval;                      //used to store temp. reading for use in displaying current temp.
int Pval;                       //used to store temp. reading for use in control algorithm

unsigned char tens;             //Initilization, var used in seperate function
unsigned char ones;             //initilization  var used in seperate function

unsigned int const1 = 0xC350;  //TCNT = ((intTime/(1/BusCLK)) / Pre) = ((100ms/(1/4Mhz)) / 8) = 50,000 = $C350

unsigned char PRE = 0x70;        //PWM Prescaler
unsigned char PERIOD = 0xFA;     //PWM Period
unsigned char PWMCLOCK = 0x00;   //0 when scaler is not used, $08 when scaler is used

unsigned char display[10] = {0x5f,0x06,0x3b,0x2f,0x66,0x6d,0x7d,0x07,0x7F,0x67};

void SEPERATE(void);        //Function prototypes
void SHOW(void);
void DELAY(void);

int Spoint = 32;            //control algorithm variables
int Gain = 0x14;
int Adjust = 0x10;

unsigned int Q = 7;         //Delay function variable initilizations
unsigned int W = 65535;

unsigned int A = 0;         //Main variable initilizations
unsigned int B = 0;
unsigned int D = 0;
unsigned long E = 0;        //long used to deal with product values larger than 16-bit

int J = 0;                  //Timer function variable initilizations
int K = 0;
unsigned char L = 0;
//****************************************************************
//*                          MAIN                                *
//****************************************************************
void main(void)
{
  PWMCLK = PWMCLOCK;        //turn on scaler B for channel 3
  PWMPRCLK = PRE;           //set PWM PRE
  PWMPER3 = PERIOD;         //setting PWM period for channel 3
  PWME = 0x08;              //enable PWM channel 3
  PWMPOL = 0x08;            //setting polarity so 100% DTY = Max brightness
  PWMDTY3 = 0;              //clear PWMDTY3 (to ensure nothing is left in register)
  TSCR2 = 0x03;             //set onboard timer pre to 8

  TIOS = 0x01;              //set channel 0 for output compare
  TIE = 0x01;               //enable interrupt for compare 0

  TC0 = const1;             //TC0 = const1 (loading compare value for timer interrupt)

  TSCR1 = 0x80;             //Turn on Timer

  DDRA = 0xFF;              //Set PORTA as outouts
  ATD1CTL2 = 0x80;         //Turn on A/D
  ATD1CTL3 = 0x08;         //set A/D one conversion/cycle

        EnableInterrupts;

  for(;;)
  {

      ATD1CTL4 = 0x61;                //A/D 2mHz CLK in 10 bit mode

      ATD1CTL5 = 0x82;                //Start A/D channel 2
        while(!(ATD1STAT0 & 0x82));   //poll ATD1STAT0 until bit 0x82 changes
        B = ATD1DR0L;                 //B = 10-bit result (low byte)
        A = ATD1DR0H;                 //A = 10-bit reslult (high byte)
        A = A<<8;                     //A = (shift A 8 bits left)
        E = A + B;                    //E = A + B, combining the two results into one variable
        E = ((E * 489)/1000);         //E = ((E * 489)/1000)

        D = (E - 273);                //D = (E-273), D is now in celcius
        Pval = D;                     //Pval = D, for use in timer interrupt
        myval = D;                    //myval = D, for use in SEPERATE function

        SEPERATE();                   //Call "SEPERATE" function

        SHOW();                       //Call "SHOW" function

  }

}
//;*************************************************
//;*                                               *
//;*     FUNCTION:SEPERATE                         *
//;*     IN:Nothing                                *
//;*     OUT:Nothing                               *
//;*                                               *
//;*    The purpose of this function is to         *
//;*    seperate a decimal number into two         *
//;*    variables so they can be displayed one at  *
//;*    a time on a 7-segment display.             *
//;*                                               *
//;*                                               *
//;*    Created by Brandon Empie                   *
//;*                                               *
//;*************************************************
void SEPERATE(void)
  {
     tens = 0;                  //clear tens
     ones = 0;                  //clear ones

       while(myval != 0)        //is myval not equal to zero? if so run loop
        {
          myval--;              //myval = myval - 1
          ones++;               //ones = ones + 1
            if(ones == 10)      //is ones equal to 10? if so run statements
            {
              tens++;           //tens = tens + 1
              ones = 0;         //clear ones
            }
        }
     return;                    //return to main
  }
//;*************************************************
//;*                                               *
//;*     FUNCTION: SHOW                            *
//;*     IN:Nothing                                *
//;*     OUT:Nothing                               *
//;*                                               *
//;*    The purpose of this function is to show    *
//;*    the tempeture in celcius on the 7-segment  *
//;*    display                                    *
//;*                                               *
//;*    Created by Brandon Empie                   *
//;*                                               *
//;*************************************************
void SHOW(void)
  {
  unsigned char Z = 0;     //initilize local variable Z = 0
  Z = tens;                //Z = tens
  PORTA = display[Z];      //PORTA = display[Z]
  DELAY();                 //call 'DELAY'
  PORTA = 0;               //PORTA = 0
  DELAY();                 //call 'DELAY'
  Z = ones;                //Z = ones
  PORTA = display[Z];      //PORTA = display[Z]
  DELAY();                 //call 'DELAY'
  PORTA = 0;               //PORTA = 0
  DELAY();                 //call 'DELAY'
  return;                  //return to main
  }
//;**************************************************
//;*                                                *
//;*     Function: DELAY                            *
//;*     IN:Nothing                                 *
//;*     OUT:Nothing                                *
//;*                                                *
//;*    The purpose of this function is to create    *
//;*    A delay.                                    *
//;*                                                *
//;*    Created by Brandon Empie                    *
//;*                                                *
//;**************************************************
void DELAY(void)
  {
    while(Q != 0)                      //As long as Q is not zero
    {
      for(W=65535;W>0;W--);            //initilize W = 65535,decrement W by 1 as long as W is > 0
      Q--;                             //Q = Q - 1
    }
    Q = 7;                             //Q = 7 (resetting delay length)
  return;                              //return to main
  }
//;********************************************************
//;*                                                      *
//;*     ISR: TIMER                                       *
//;*     IN:Nothing                                       *
//;*     OUT:Nothing                                      *
//;*                                                      *
//;*    The purpose of this Interrupt is to update         *
//;*    lamp brightness (heat) based on proportion         *
//;*    control algorithm:                                *
//;*    Output = (setpoint - present value)*gain + offset  *
//;*    Or POT1 input when operated in manual mode         *
//;*                                                      *
//;*    Created by Brandon Empie                          *
//;*                                                      *
//;********************************************************
interrupt void TIMER(void)
  {
  J = 0;                      //clear J
  K = 0;                      //clear K
  L = PORTB;                  //L = PORTB (checking PORTB for 0,1 used in switch statement)

  switch(L)                   //swtich statement based on value of L (PORTB)
  {
    case 0:                           //if PORTB == 0      (control algorithm based on Spoint)
          K = Spoint - Pval;          //K = Spoint - Pval
          if(K <= 0)                  //if K <= 0
          {
            PWMDTY3 = Adjust;         //PWMDTY3 = Adjust (cool down bulb)
          }
          else if(K > 0)              //if K > 0
          {
           K = ((K * Gain) + Adjust); //K = ((K * Gain) + Adjust)
              if(K <= 0xFF)           //then if K is <= 0xFF
                PWMDTY3 = K;          //PWMDTY3 = K;
              else if(K > 0xFF)       //otherwise if K > 0xFF
                PWMDTY3 = 0xFF;       // PWMDTY3 = 0xFF
          }
          break;                      //break out of switch statement
    case 1:                           //if PORTB == 1 (manual mode)
          ATD1CTL4 = 0xE1;            //set A/D for 8-bit mode
          ATD1CTL5 = 0x80;            //Start A/D channel 0
          while(!(ATD1STAT0 & 0x80)); //poll A/D until a change is detected on channel 0
          J = ((((ATD1DR0L * 100)/255) * PERIOD) / 100);
          PWMDTY3 = J;                //PWMDTY3 = ((((A/D result * 100)/255) * PWM_period)/100)
          break;
  }
  TCNT = 0x0000;                      //clear TCNT
  TFLG1 = 0x01;                       //clear TFLG1 so next timer interrupt can occur
  return;
  }
```