Yanick Palacios i Arnau Cullell Grup: 13 Subgrup: E

CODI L6b PREVI

```
#include <xc.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "config.h"
#include "GLCD.h"
#include <math.h>
#define _XTAL_FREQ 8000000

/*
Aquest termometre té la capacitat de marcar -55 graus celcsius fins a 400 graus celsius.
Entre cada petita barra negra la diferencia es de 18 graus, i llavors, per veure el progres
de la barra cal augmentar en 9 graus la temperatura (ja que hi ha un espai entre barra negra
i barra negra)
*/

char buff[15]; //amb 15 caràcters tenim espai suficient
char prev[3] = {0, 0, 0}; //serviran per indicar l'estat previ de RC6 i RC7 i RC0 respectivament, char ja serveix perquè només serà 1 o 0
char duty = 66; //indica el percentatge de duty que es vol, char perquè el valor només arribarà fins a 100, no cal més
char duty_prev = -1; //indica el duty abans d'haver ser modificat
int segons = 10;
int decimes = 0;
int num0 = 0;
int temp;
int prev_temp;
int puja_o_baixa = 0;

float beta = 4050;
float A = 0.0059257;
```

```
float R2 = 4751;
float VCC = 5;

char first = 1; //Servirà per dibuixar la barra per primer cop
char estat = 0;
char prev_estat = -1;

char* red = "Ready";
char* run = "Run  ";
char* sto = "Stop! ";

//          PSI =    0   1   2   3   4   5   6   7   8   9
int press2bit[101] = {63, 63, 62, 62, 61, 61, 60, 60, 59, 59,
//               10  11  12  13  14  15  16  17  18  19
                      58, 58, 57, 57, 56, 56, 55, 55, 54, 54,
//               20  21  22  23  24  25  26  27  28  29
                      53, 53, 52, 52, 51, 51, 50, 50, 49, 49,
//               30  31  32  33  34  35  36  37  38  39
                      48, 48, 47, 47, 46, 46, 45, 45, 44, 44,
//                    40  41  42  43  44  45  46  47  48  49
                      43, 43, 42, 42, 41, 41, 40, 40, 39, 39,
//               50  51  52  53  54  55  56  57  58  59
                      38, 38, 37, 37, 36, 36, 35, 35, 34, 34,
//               60  61  62  63  64  65  66  67  68  69
                      33, 33, 32, 32, 31, 31, 30, 30, 29, 29,
//               70  71  72  73  74  75  76  77  78  79
                      28, 28, 27, 27, 26, 26, 25, 25, 24, 24,
//               80  81  82  83  84  85  86  87  88  89
                      23, 23, 22, 22, 21, 21, 20, 20, 19, 19,
//               90  91  92  93  94  95  96  97  98  99
                      18, 18, 17, 17, 16, 16, 15, 15, 14, 14,
```

```
//                  100
                   13};


void writeTxt(byte page, byte y, char * s) {
  int i=0;
  while (*s!='\n' && *s!='\0')
  {
    putchGLCD(page, y+i, *(s++));
    i++;
  };
}


void dibuixar_barra(char y, char x) { //en el dibuix, entre cada barreta la diferència és de 4 psi, i les barres grans marquen: 0, 20, 40, 60, 80, 100
  //nivell 0
  writeByte(y, x, 0x80);
  writeByte(y, x+1, 0x80);
  writeByte(y, x+2, 0xAA);
  writeByte(y, x+3, 0xAA);
  writeByte(y, x+4, 0x1F);
  writeByte(y, x+5, 0x20);
  writeByte(y, x+6, 0x40);
  for (int i = 7; i < 21; ++i) writeByte(y, x+i, 0x80);
  writeByte(y, x+21, 0x40);
  writeByte(y, x+22, 0x20);
  writeByte(y, x+23, 0x1F);

  //nivell 1
  writeByte(y-1, x, 0x20);
  writeByte(y-1, x+1, 0x20);
  writeByte(y-1, x+2, 0xAA);
```

```
  writeByte(y-1, x+3, 0xAA);
  writeByte(y-1, x+4, 0xFF);
  writeByte(y-1, x+23, 0xFF);

  //nivell 2
  writeByte(y-2, x, 0x08);
  writeByte(y-2, x+1, 0x08);
  writeByte(y-2, x+2, 0xAA);
  writeByte(y-2, x+3, 0xAA);
  writeByte(y-2, x+4, 0xFF);
  writeByte(y-2, x+23, 0xFF);

  //nivell 3
  writeByte(y-3, x, 0x02);
  writeByte(y-3, x+1, 0x02);
  writeByte(y-3, x+2, 0xAA);
  writeByte(y-3, x+3, 0xAA);
  writeByte(y-3, x+4, 0xFF);
  writeByte(y-3, x+23, 0xFF);

  //nivell 4
  writeByte(y-4, x+2, 0xAA);
  writeByte(y-4, x+3, 0xAA);
  writeByte(y-4, x+4, 0xFF);
  writeByte(y-4, x+23, 0xFF);

  //nivell 5
  writeByte(y-5, x, 0x80);
  writeByte(y-5, x+1, 0x80);
  writeByte(y-5, x+2, 0xAA);
  writeByte(y-5, x+3, 0xAA);
```

```
  writeByte(y-5, x+4, 0xFF);
  writeByte(y-5, x+23, 0xFF);

  //nivell 6
  writeByte(y-6, x, 0x20);
  writeByte(y-6, x+1, 0x20);
  writeByte(y-6, x+2, 0xA0);
  writeByte(y-6, x+3, 0xA0);
  writeByte(y-6, x+5, 0x80);
  writeByte(y-6, x+6, 0x40);
  for (int i = 7; i < 21; ++i) writeByte(y-6, x+i, 0x20);
  writeByte(y-6, x+21, 0x40);
  writeByte(y-6, x+22, 0x80);

  //omplir barra inicialment
  int ref = (int)(((float)(temp + 55) / 455) * 100); //passar d'un valor entre [-55, 400] a [0, 100]
  int aux = press2bit[ref] -1;
  for (int i = 63; i != aux; --i) {
    if (i == 63 || i == 12) {} // ja està dibuixat, no cal fer res

    else if (i == 13 || i == 62) { //part de la forma irregular
          for (int j = 7; j < 21; ++j) SetDot(i, x+j);
    }

    else if (i == 14 || i == 61) { //part de la forma irregular
          SetDot(i,x+6);
          SetDot(i,x+21);
          for (int j = 7; j < 21; ++j) SetDot(i, x+j);
    }
    else for (int j = 4; j < 23; ++j) SetDot(i, x+j); //tota la resta del mig
  }
```

```
}

void tic() {
  decimes -= 1;
  if (decimes == -1) {
    decimes = 9;
    --segons;
  }

  if (segons == -1) {
    segons = 0;
  }
}

void interrupt service_routine_HighP (){

  if (TMR0IE && TMR0IF) {
    INTCONbits.TMR0IF = 0; //baixem el flag
    TMR0H = 0x3C; //reiniciem registres per tornar a fer 0.1 seg
    TMR0L = 0xAF;
    tic();
  }

  if (PIR1bits.ADIF && PIE1bits.ADIE) { //ENTRA QUAN ACABA CONVERSIO
        num0 = (ADRESH << 8) + ADRESL;

        PIR1bits.ADIF = 0;
  }
}

int calcular_temp(int num0) {
```

```
  float Vout = (num0 * VCC) / 1023.0;
  float C = R2 / A;
  float lnValue = log(C * (VCC - Vout) / Vout);
  return (beta / lnValue) - 273.15;
}

unsigned int adjustPWM(float temperature, unsigned int pwmSelected) {
   int pwmCorrected = pwmSelected + 0.4 * (25.0 - temperature);
   if (pwmCorrected > 100) pwmCorrected = 100;
   if (pwmCorrected < 0) pwmCorrected = 0;
   return pwmCorrected;
}

void canviar_prog(char x) {
  int ref = (int)(((float)(temp + 55) / 455) * 100);
  int y = press2bit[ref]; //ens donarà la y corresponent del GLCD y:[0,63]

  //pujar
  if (puja_o_baixa == 9) {
    puja_o_baixa = 0;
    if (y == 63 || y == 12) {} // ja està dibuixat, no cal fer res

    else if (y == 13 || y == 62) { //part de la forma irregular
          for (int j = 7; j < 21; ++j) SetDot(y, x+j);
    }

    else if (y == 14 || y == 61) { //part de la forma irregular
          SetDot(y,x+6);
          SetDot(y,x+21);
          for (int j = 7; j < 21; ++j) SetDot(y, x+j);
    }
```

```
    else for (int j = 4; j < 23; ++j) SetDot(y, x+j); //tota la resta del mig
  }

  //baixar
  else if (puja_o_baixa == -9) {
    puja_o_baixa = 0;
    if (y == 14) {} //no cal esborrar perquè sinó treuria el sostre

    else if (y == 63 || y == 15) { //part de la forma irregular
          for (int j = 7; j < 21; ++j) ClearDot(y-1, x+j);
    }

    else if (y == 62 || y == 16) { //part de la forma irregular
          ClearDot(y-1,x+6);
          ClearDot(y-1,x+21);
          for (int j = 7; j < 21; ++j) ClearDot(y-1, x+j);
    }

    else for (int j = 5; j < 23; ++j) ClearDot(y-1, x+j); //tota la resta del mig
  }
}

void updateGLCD() {
  if (prev_temp < temp) ++puja_o_baixa;
  else if (prev_temp > temp) --puja_o_baixa;

  prev_temp = temp;
  temp = calcular_temp(num0);
  sprintf(buff, "Temperatura: %d  ", temp);
  writeTxt(7, 0, buff);
```

```
  sprintf(buff, "Duty:", duty);
  writeTxt(0, 7, buff);
  int PWM = adjustPWM(temp, duty);
  sprintf(buff, "%d%% ", PWM);
  writeTxt(2, 7, buff);

  //configurar el duty cycle
  if (duty_prev != duty) {
    float valor_float = ((float)duty/100.0) * (PR2+1) * 4;
    int valor = (int)valor_float;
    CCPR3L = valor >> 2; //part alta del duty cycle
    CCP3CONbits.DC3B = valor & 0b11; //part baixa
  }

  if (prev_estat != estat) {
    if (estat == 0) {
          writeTxt(0, 0, red); //ready
          sprintf(buff, "10.0 "); //resetegem
          writeTxt(2, 0, buff);
    }

    else if (estat == 1) {
          CCP3CONbits.CCP3M = 0b1100; //diem que funcioni per el PWM
          TMR0ON = 1; // toca encendre el cronometre
          TMR0IE = 1;
          sprintf(buff, "0%d.%d ", segons, decimes); //actualitzem cronòmetre
          writeTxt(2, 0, buff);
          if (estat != prev_estat) { //si just canvia d'estat es posa running, sino no cal
            writeTxt(0, 0, run);
          }
```

```
            if (segons == 0 && decimes == 0) {
              TMR0ON = 0;
              estat = 2;
            }
        }

        else { //estat == 2
            CCP3CONbits.CCP3M = 0b0000; //diem que no funcioni per el PWM
            TMR0ON = 0; // toca apagar el cronometre
            writeTxt(0, 0, sto);
            segons = 10; //resetegem
            decimes = 0;
        }
    }

    if (first) {
        dibuixar_barra(7, 95); //barra a posició y = 7 i x = 95
        first = 0;
    }

    //pujar o baixar la barra
    if (prev_temp != temp) canviar_prog(95);

}

void actualitzar_flancs() {
    prev[0] = PORTCbits.RC6; //RC6
    prev[1] = PORTCbits.RC7; //RC7
    prev[2] = PORTCbits.RC0; //RC0
}
```

```
void config_PIC() {
  ANSELA=0x00;
  ANSELB=0x00;
  ANSELC=0x00;
  ANSELD=0x00;
  ANSELEbits.ANSE0 = 0;
  ANSELEbits.ANSE1 = 1; //analogic

  TRISD=0x00;
  TRISB=0x00;
  TRISC=0xC1; //RC7 i RC6 i RC0 són inputs
  TRISEbits.TRISE0 = 0; //output per a el PWM
  TRISEbits.TRISE1 = 1; //input per a el RE1

  PORTD=0x00;
  PORTB=0x00;
  PORTC=0x00;

  //configuració interrupcions
  RCONbits.IPEN = 1;
  INTCONbits.GIEL = 1;
  INTCONbits.GIEH = 1; //activem interrupcions alta prioritat
  INTCONbits.TMR0IF = 0; //flag a 0
  INTCON2bits.TMR0IP = 1; //alta prioritat
  PIE1bits.ADIE = 1; //activem el ADIE
  PIR1bits.ADIF = 0; //flag a 0
  IPR1bits.ADIP = 1; //perquè sigui alta prioritat

  //configurar el PWM
  CCPTMRS0bits.C3TSEL = 0; //seleccionem el timer2 pel cpp3
```

```
  //configurar TMR0
  T08BIT = 0; //16 bits pel TMR0
  T0CS = 0; //Fosc/4
  PSA = 0; //amb preescalar
  T0CONbits.T0PS = 0b001; // PRE = 4
  TMR0H = 0x3C; //posem TMR0 amb valor adequat
  TMR0L = 0xAF;

  //configurar TMR2
  T2CONbits.T2CKPS = 0b10; //preescaler de 16, perquè el valor ha d'estar entre 0 i 255
  PR2 = 124; // periode del PWM (1000Hz amb 8Mhz, que es 1ms, el que volem)
  T2CONbits.TMR2ON = 1; //activem el TMR2

  //configurar converter
  ADCON0 = 0x19; //seleccionem el canal AN6 (CHS3:CHS0 = 00110), AD encés  (ADON = 1)
  ADCON1 = 0x00; //referencia VDD y VSS
  ADCON2 = 0xA9; //ADFM=1 (justificació a la dreta), ACQT = 101 (12 Tad), ADCS = 001 (Fosc/8)
}

void main(void) {
  config_PIC();
  ADCON0bits.GO = 1;
  GLCDinit();                   //Inicialitzem la pantalla
  clearGLCD(0,7,0,127);   //Esborrem pantalla
  setStartLine(0);       //Definim linia d'inici

  while (1) {
    if (ADCON0bits.GO == 0) ADCON0bits.GO = 1;
    if (prev[0] == 1 && PORTCbits.RC6 == 0) { //suma el duty a flanc de baixada
          if (duty == 100) {}
```

```
            else {
              duty_prev = duty;
              ++duty;
            }
        }

        if (prev[1] == 1 && PORTCbits.RC7 == 0) { //resta el duty a flanc de baixada
            if (duty == 0) {}
            else {
              duty_prev = duty;
              --duty;
            }
        }

        if (prev[2] == 1 && PORTCbits.RC0 == 0) { //canvi l'estat del cronometre
            prev_estat = estat;
            ++estat;
            if (estat == 3) estat = 0;
        }

        if (duty != duty_prev || prev_estat != estat) updateGLCD();
        actualitzar_flancs();
        __delay_ms(75);
      }
    }
```