Yanick Palacios i Arnau Cullell Grup: 13 Subgrup: E

CODI LAB6a

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

char buff[15]; //amb 15 caràcters tenim espai suficient
char prev[2] = {0, 0}; //serviran per indicar l'estat previ de RC6 i RC7 respectivament, char ja serveix perquè només serà 1 o 0
char duty = 50; //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

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

void updateGLCD() {
  sprintf(buff, "Duty: %d%% ", duty);
  writeTxt(0, 9, buff);

  //configurar el duty cycle
  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
}

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

void config_PIC() {
  ANSELA=0x00;
  ANSELB=0x00;
  ANSELC=0x00;
  ANSELD=0x00;
  ANSELEbits.ANSE0 = 0;

  TRISD=0x00;
  TRISB=0x00;
  TRISC=0xC0; //RC7 i RC6 són inputs
  TRISEbits.TRISE0 = 0; //output per a el PWM

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

  //configurar el PWM
  CCP3CONbits.CCP3M = 0b1100; //diem que funcioni per el PWM
  CCPTMRS0bits.C3TSEL = 0; //seleccionem el timer2 pel cpp3

  //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
}

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

  while (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 (duty != duty_prev) updateGLCD();
    actualitzar_flancs();
    __delay_ms(75);
  }
}
```