Sistema de Seguridad

Adrian Sanchez-Kevin Vernaza-Camilo Castro
Versión 3

Índice de archivos

Lista de archivos

Lista de todos los archivos con breves descripciones:

Documentación de archivos

Referencia del archivo D:/Trabajos MICRO/ADC.X/adc_port.c

```
#include <pic16f887.h>
#include "config.h"
#include <math.h>
#include "LCD.h"
#include <stdio.h>
#include <xc.h>
```

defines

- #define **LIGHT_MIN** 0
- #define **LIGHT_MAX** 1023
- #define **beta** 4090

Funciones

- void **leer_temperatura** (void)
  Lee el sensor de temperatura y calcula la temperatura en grados Celsius.

- void **leer_luz** (void)
  Lee el sensor de luz y calcula el porcentaje de luz ambiente.

- void **leer_micro** (void)
  Lee el sensor de micrófono y obtiene el valor de entrada analógica.

- void **leer_potenciometro** (void)
  Lee el valor del potenciómetro y guarda el resultado en la variable global `potenciometro_valor` .

- void **comparacion** (void)
  Compara la temperatura con un umbral y activa los pines de salida correspondientes.

- void **main** (void)

Variables

- float **temp** = 0.0
- int **pot** = 0
- int **luz** = 0
- int **mic** = 0
- float **tempC**
- float **A** = 0.001129148
- float **B** =0.000234125
- float **C** =0.0000000076741
- float **Var1** = 0.04058

Documentación de «define»

#define beta 4090

Definición en la línea **31** del archivo **adc_port.c**.

#define LIGHT_MAX 1023

Definición en la línea **30** del archivo **adc_port.c**.

#define LIGHT_MIN 0

Definición en la línea **29** del archivo **adc_port.c**.

Documentación de funciones

void comparacion (void)

Compara la temperatura con un umbral y activa los pines de salida correspondientes.

Si la temperatura es mayor o igual a 23°C, activa el pin RA3. Si la temperatura es menor o igual a 22°C, activa el pin RA4. En otros casos, activa el pin RA5.

Definición en la línea **115** del archivo **adc_port.c**.

void leer_luz (void)

Lee el sensor de luz y calcula el porcentaje de luz ambiente.

Utiliza el canal 6 del ADC para leer la tensión del sensor de luz y la convierte en un porcentaje de luz ambiente.

Definición en la línea **165** del archivo **adc_port.c**.

void leer_micro (void)

Lee el sensor de micrófono y obtiene el valor de entrada analógica.

Utiliza el canal 10 del ADC para leer el valor de entrada analógica del sensor de micrófono.

Definición en la línea **185** del archivo **adc_port.c**.

void leer_potenciometro (void)

Lee el valor del potenciómetro y guarda el resultado en la variable global `potenciometro_valor` .

Utiliza el canal 13 del ADC para leer el valor del potenciómetro y guarda el resultado en la variable global `potenciometro_valor` .

Definición en la línea **206** del archivo **adc_port.c**.

void leer_temperatura (void)

Lee el sensor de temperatura y calcula la temperatura en grados Celsius.

Utiliza el canal 0 del ADC para leer la tensión del sensor de temperatura y la convierte en temperatura en grados Celsius utilizando la ecuación de Steinhart-Hart.

Definición en la línea **144** del archivo **adc_port.c**.

void main (void)

Ciclo infinito donde se llaman las funciones y se hace la conversión a cadena de texto

Definición en la línea **43** del archivo **adc_port.c**.

Documentación de variables

float A = 0.001129148

Definición en la línea **42** del archivo **adc_port.c**.

float B =0.000234125

Definición en la línea **42** del archivo **adc_port.c**.

float C =0.0000000076741

Definición en la línea **42** del archivo **adc_port.c**.

int luz = 0

Definición en la línea **38** del archivo **adc_port.c**.

int mic = 0

Definición en la línea **39** del archivo **adc_port.c**.

int pot = 0

Definición en la línea **37** del archivo **adc_port.c**.

float temp = 0.0

Definición en la línea **36** del archivo **adc_port.c**.

float tempC

Definición en la línea **41** del archivo **adc_port.c**.

float Var1 = 0.04058

Definición en la línea **42** del archivo **adc_port.c**.

adc_port.c

Ir a la documentación de este archivo.

```
00001
00015 #include <pic16f887.h>
00016 #include "config.h"
00017 #include <math.h>
00018 #include "LCD.h"
00019 #include <stdio.h>
00020 #include <xc.h>
00021 //#include <string.h>
00022
00023 void leer_temperatura(void);
00024 void leer_luz(void);
00025 void leer_micro(void);
00026 void leer_potenciometro(void);
00027 void comparacion(void);
00028
00029 #define LIGHT_MIN 0
00030 #define LIGHT_MAX 1023
00031 #define beta 4090
00032
00033
00034
00035
00036 float temp = 0.0;
00037 int pot = 0;
00038 int luz = 0;
00039 int mic = 0;
00040
00041 float tempC;
00042 float A= 0.001129148, B=0.000234125, C=0.0000000076741, Var1= 0.04058;
00043 void main(void) {
00044     OSCCON = 0x71; // reloj interno de 8M - Configura el oscilador
00045
00046     TRISA0 = 1; // Configura RA0 como entrada
00047     TRISE1 = 1; // Configura E1 como entrada
00048     TRISB1 = 1; // Configura RB1 como entrada
00049     TRISB5 = 1; // Configura RB5 como entrada
00050
00051     TRISA3 = 0;
00052     TRISA4 = 0;
00053     TRISA5 = 0;
00054
00055
00056
00057     ANSEL = 0x41; // Configura el Puerto como Entrada Analógica.
00058     ANSELH = 0x24;
00059
00060
00061
00062     ADCON1bits.ADFM = 0; // Justificación Izquierda (modo-8bits).
00063     ADCON1bits.VCFG0 = 0; // Selecciona Voltajes de Referencia (5v-0v).
00064     ADCON1bits.VCFG1 = 0; // Selecciona Voltajes de Referencia (5v-0v).
00065     ADCON0bits.ADCS = 0b01; // Tiempo de Conversión Fosc/8.
00066
00067
00068     LCD_Init(); //Inicializa el LCD
00069     LCD_String_xy(0,0,"Bievenido");
00070     __delay_ms(2000);
00071
00077     while (1) {
00078         leer_temperatura();
00079         leer_luz();
00080         leer_micro();
00081         leer_potenciometro();
00082         comparacion();
00083
00084         char buferT[7];
```

```
00085         sprintf(buferT, "T:%.2f", tempC);//convertir a cadena de texto
00086
00087         char buferL[7];
00088         sprintf(buferL, "L:%d", luz);
00089
00090         char buferM[7];
00091         sprintf(buferM, "M:%d", mic);
00092
00093         char buferP[7];
00094         sprintf(buferP, "P:%d", pot);
00095
00096         LCD_Clear();
00097         LCD_String_xy(0,0,buferT);
00098         LCD_String_xy(0,10,buferL);
00099         LCD_String_xy(2,0,buferM);
00100         LCD_String_xy(2,10,buferP);
00101         __delay_ms(1000);
00102
00103     }
00104 }
00105
00106
00115 void comparacion(void) {
00116         if (tempC >= 23.0) {
00117               PORTAbits.RA3 = 1;
00118               PORTAbits.RA4 = 0;
00119               PORTAbits.RA5 = 0;
00120         }
00121         else if(tempC <= 22.0) {
00122               PORTAbits.RA3 = 0;
00123               PORTAbits.RA4 = 1;
00124               PORTAbits.RA5 = 0;
00125         }
00126         else {
00127               PORTAbits.RA3 = 0;
00128               PORTAbits.RA4 = 0;
00129               PORTAbits.RA5 = 1;
00130         }
00131 }
00132
00133
00134
00144 void leer_temperatura(void) {
00145     ADCON0bits.CHS = 0b0000; // Selecciona el Canal Analógico AN0.
00146     ADCON0bits.ADON = 1; // Habilita el Módulo AD.
00147     __delay_us(30);
00148     ADCON0bits.GO = 1; // Inicia la Conversión AD.
00149     while (ADCON0bits.GO); // Espera a que termine la conversión AD.
00150
00151     unsigned int Vo = ADRESH; // Lee el valor del conversor AD
00152     Vo= 1023.0 - (Vo*4);
00153     tempC= Vo*(Var1);
00154
00155 }
00156
00165 void leer_luz (void){
00166     ADCON0bits.CHS = 0b0110; // Selecciona el Canal Analógico
00167     ADCON0bits.ADON = 1; // Habilita el Módulo AD
00168     __delay_us(30);
00169     ADCON0bits.GO = 1; // Inicia la Conversión AD.
00170     while (ADCON0bits.GO); // Espera a que termine la conversión AD.
00171
00172     unsigned int luzx = ADRESH * 4; // Lee el valor del conversor AD para el sensor de
luz
00173     float porcentaje = ((float)(luzx - LIGHT_MIN) / (LIGHT_MAX - LIGHT_MIN)) * 100.0;
00174     // Calcula el porcentaje de luz
00175     luz = (int)porcentaje;
00176 }
00177
00185 void leer_micro (void){
00186
```

```
00187     ADCON0bits.CHS = 0b1010; // Selecciona el Canal Analógico
00188     ADCON0bits.ADON = 1; // Habilita el Módulo AD.
00189     __delay_us(30);
00190     ADCON0bits.GO = 1; // Inicia la Conversión AD
00191     while (ADCON0bits.GO); // Espera a que termine la conversión AD.
00192 
00193     unsigned int microfono = ADRESH;
00194     ADCON0bits.ADON = 0; // apaga el Módulo AD.
00195     mic = microfono;
00196 }
00197 
00198 
00199 
00206 void leer_potenciometro (void){
00207     ADCON0bits.CHS = 0b1101; // Selecciona el Canal Analógico
00208     ADCON0bits.ADON = 1; // Habilita el Módulo AD.
00209     __delay_us(30);
00210     ADCON0bits.GO = 1; // Inicia la Conversión AD.
00211     while (ADCON0bits.GO); // Espera a que termine la conversión AD.
00212 
00213     unsigned int potenciometro = ADRESH;
00214 
00215     ADCON0bits.ADON = 0; // apaga el Módulo AD.
00216     pot = potenciometro;
00217 }
00218 
00219 
```

Referencia del archivo D:/Trabajos MICRO/ADC.X/config.h

Configurations bits Settings.

defines

- #define **_XTAL_FREQ** 8000000

Descripción detallada

Configurations bits Settings.

Fecha

2021-09-13

Autor

Fulvio Vivas `fyvivas@unicauca.edu.co`

Copyright

Definición en el archivo **config.h**.

Documentación de «define»

#define _XTAL_FREQ 8000000

Definición en la línea **46** del archivo **config.h**.

config.h

Ir a la documentación de este archivo.

```
00001
00021 #ifndef CONFIG_H
00022 #define CONFIG_H
00023
00024 #ifdef  __cplusplus
00025 extern "C" {
00026 #endif
00027
00028 // CONFIG1
00029 #pragma config FOSC = INTRC_CLKOUT// Oscillator Selection bits (INTOSC oscillator: CLKOUT
function on RA6/OSC2/CLKOUT pin, I/O function on RA7/OSC1/CLKIN)
00030 #pragma config WDTE = OFF       // Watchdog Timer Enable bit (WDT disabled and can be
enabled by SWDTEN bit of the WDTCON register)
00031 #pragma config PWRTE = OFF      // Power-up Timer Enable bit (PWRT disabled)
00032 #pragma config MCLRE = OFF      // RE3/MCLR pin function select bit (RE3/MCLR pin function
is digital input, MCLR internally tied to VDD)
00033 #pragma config CP = OFF         // Code Protection bit (Program memory code protection
is disabled)
00034 #pragma config CPD = OFF        // Data Code Protection bit (Data memory code protection
is disabled)
00035 #pragma config BOREN = OFF      // Brown Out Reset Selection bits (BOR disabled)
00036 #pragma config IESO = OFF       // Internal External Switchover bit (Internal/External
Switchover mode is disabled)
00037 #pragma config FCMEN = OFF      // Fail-Safe Clock Monitor Enabled bit (Fail-Safe Clock
Monitor is disabled)
00038 #pragma config LVP = OFF        // Low Voltage Programming Enable bit (RB3 pin has digital
I/O, HV on MCLR must be used for programming)
00039
00040 // CONFIG2
00041 #pragma config BOR4V = BOR40V   // Brown-out Reset Selection bit (Brown-out Reset set
to 4.0V)
00042 #pragma config WRT = OFF        // Flash Program Memory Self Write Enable bits (Write
protection off)
00043
00044 // #pragma config statements should precede project file includes.
00045 // Use project enums instead of #define for ON and OFF.
00046 #define _XTAL_FREQ 8000000//frecuencia del oscilador
00047
00048
00049 #ifdef  __cplusplus
00050 }
00051 #endif
00052
00053 #endif  /* CONFIG_H */
```

Referencia del archivo D:/Trabajos MICRO/ADC.X/LCD.c

```
#include <xc.h>
#include "LCD.h"
#include "config.h"
```

Funciones

- void **LCD_Init** (void)
- void **LCD_Command** (unsigned char cmd)
- void **LCD_Char** (unsigned char dat)
- void **LCD_String** (const char *msg)
- void **LCD_String_xy** (char row, char pos, const char *msg)
- void **LCD_Clear** (void)

Documentación de funciones

void LCD_Char (unsigned char *dat*)

Definición en la línea **43** del archivo **LCD.c**.

void LCD_Clear (void)

Definición en la línea **88** del archivo **LCD.c**.

void LCD_Command (unsigned char *cmd*)

Definición en la línea **27** del archivo **LCD.c**.

void LCD_Init (void)

Definición en la línea **15** del archivo **LCD.c**.

void LCD_String (const char * *msg*)

Definición en la línea **58** del archivo **LCD.c**.

void LCD_String_xy (char *row*, char *pos*, const char * *msg*)

Definición en la línea **68** del archivo **LCD.c**.

LCD.c

Ir a la documentación de este archivo.

```
00001
00011 #include <xc.h>
00012 #include "LCD.h"
00013 #include "config.h"
00014
00015 void LCD_Init(void)
00016 {
00017     LCD_Port = 0;          /*PORT as Output Port*/
00018     __delay_ms(15);           /* 15 ms, Power-On delay*/
00019     LCD_Command(0x02);     /*send for initialization of LCD with nibble method */
00020     LCD_Command(0x28);     /*use 2 line and initialize 5*7 matrix in (4-bit mode)*/
00021     LCD_Command(0x01);     /*clear display screen*/
00022     LCD_Command(0x0c);     /*display on cursor off*/
00023     LCD_Command(0x06);     /*increment cursor (shift cursor to right)*/
00024 }
00025
00026
00027 void LCD_Command(unsigned char cmd )
00028 {
00029     ldata = (ldata & 0x0f) |(0xF0 & cmd);  /*Send higher nibble of command first to PORT*/
00030     RS = 0;  /*Command Register is selected i.e.RS=0*/
00031     EN = 1;  /*High-to-low pulse on Enable pin to latch data*/
00032     NOP();
00033     EN = 0;
00034     __delay_ms(1);
00035     ldata = (unsigned char)((ldata & 0x0f) | (cmd<<4));  /*Send lower nibble of command
to PORT */
00036     EN = 1;
00037     NOP();
00038     EN = 0;
00039     __delay_ms(3);
00040 }
00041
00042
00043 void LCD_Char(unsigned char dat)
00044 {
00045     ldata = (ldata & 0x0f) | (0xF0 & dat);  /*Send higher nibble of data first to PORT*/
00046     RS = 1;  /*Data Register is selected*/
00047     EN = 1;  /*High-to-low pulse on Enable pin to latch data*/
00048     NOP();
00049     EN = 0;
00050     __delay_ms(1);
00051     ldata = (unsigned char)((ldata & 0x0f) | (dat<<4));  /*Send lower nibble of data
to PORT*/
00052     EN = 1;  /*High-to-low pulse on Enable pin to latch data*/
00053     NOP();
00054     EN = 0;
00055     __delay_ms(3);
00056 }
00057
00058 void LCD_String(const char *msg)
00059 {
00060     while((*msg)!=0)
00061     {
00062       LCD_Char(*msg);
00063       msg++;
00064     }
00065 }
00066
00067
00068 void LCD_String_xy(char row,char pos,const char *msg)
00069 {
00070     char location=0;
00071     if(row<=1)
00072     {
```

```
00073         location=(0x80) | ((pos) & 0x0f);  /*Print message on 1st row and desired
location*/
00074         LCD_Command(location);
00075     }
00076     else
00077     {
00078         location=(0xC0) | ((pos) & 0x0f);  /*Print message on 2nd row and desired
location*/
00079         LCD_Command(location);
00080     }
00081
00082
00083     LCD_String(msg);
00084
00085 }
00086
00087
00088 void LCD_Clear(void)
00089 {
00090     LCD_Command(0x01);
00091     __delay_ms(3);
00092 }
```

Referencia del archivo D:/Trabajos MICRO/ADC.X/LCD.h

defines

- #define **RS** PORTDbits.RD0 /*PIN 0 of PORTD is assigned for register select Pin of LCD*/
- #define **EN** PORTDbits.RD1 /*PIN 1 of PORTD is assigned for enable Pin of LCD */
- #define **ldata** PORTD /*PORTD(PD4-PD7) is assigned for LCD Data Output*/
- #define **LCD_Port** TRISD /*define macros for PORTD Direction Register*/

Funciones

- void **LCD_Init** (void)
- void **LCD_Command** (unsigned char)
- void **LCD_Char** (unsigned char x)
- void **LCD_String** (const char *)
- void **LCD_String_xy** (char, char, const char *)
- void **LCD_Clear** (void)

Documentación de «define»

#define EN PORTDbits.RD1 /*PIN 1 of PORTD is assigned for enable Pin of LCD */

Definición en la línea **12** del archivo **LCD.h**.

#define LCD_Port TRISD /*define macros for PORTD Direction Register*/

Definición en la línea **14** del archivo **LCD.h**.

#define ldata PORTD /*PORTD(PD4-PD7) is assigned for LCD Data Output*/

Definición en la línea **13** del archivo **LCD.h**.

#define RS PORTDbits.RD0 /*PIN 0 of PORTD is assigned for register select Pin of LCD*/

Definición en la línea **11** del archivo **LCD.h**.

Documentación de funciones

void LCD_Char (unsigned char *x*)

Definición en la línea **43** del archivo **LCD.c**.

void LCD_Clear (void)

Definición en la línea **88** del archivo **LCD.c**.

void LCD_Command (unsigned char *cmd*)

Definición en la línea **27** del archivo **LCD.c**.

void LCD_Init (void)

Definición en la línea **15** del archivo **LCD.c**.

void LCD_String (const char * *msg*)

Definición en la línea **58** del archivo **LCD.c**.

void LCD_String_xy (char *row*, char *pos*, const char * *msg*)

Definición en la línea **68** del archivo **LCD.c**.

LCD.h

Ir a la documentación de este archivo.

```
00001
00002
00003 #ifndef LCD_H
00004 #define LCD_H
00005
00006 #ifdef  __cplusplus
00007 extern "C" {
00008 #endif
00009
00010
00011 #define RS PORTDbits.RD0  /*PIN 0 of PORTD is assigned for register select Pin of LCD*/
00012 #define EN PORTDbits.RD1  /*PIN 1 of PORTD is assigned for enable Pin of LCD */
00013 #define ldata PORTD  /*PORTD(PD4-PD7) is assigned for LCD Data Output*/
00014 #define LCD_Port TRISD  /*define macros for PORTD Direction Register*/
00015
00016 void LCD_Init(void);                    /*Initialize LCD*/
00017 void LCD_Command(unsigned char );  /*Send command to LCD*/
00018 void LCD_Char(unsigned char x);    /*Send data to LCD*/
00019 void LCD_String(const char *);     /*Display data string on LCD*/
00020 void LCD_String_xy(char, char , const char *);
00021 void LCD_Clear(void);                   /*Clear LCD Screen*/
00022
00023
00024 #ifdef  __cplusplus
00025 }
00026 #endif
00027
00028 #endif  /* LCD_H */
```

Índice

INDEX