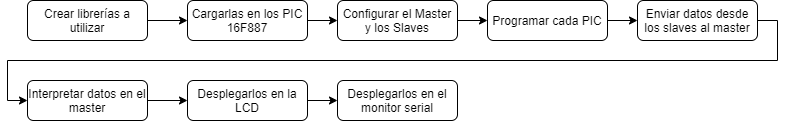
Mini Proyecto #1

SPI

**Link de GitHub:** <https://github.com/mon19379/DIGITAL2.git>

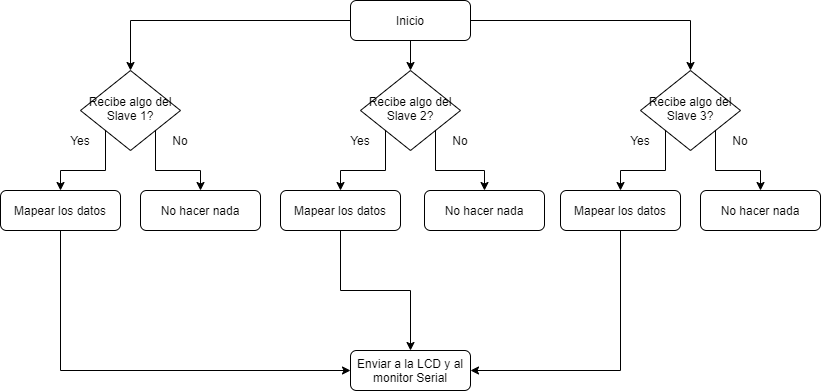
**Link Video:** [**https://youtu.be/eyKU6VjGp8Q**](https://youtu.be/eyKU6VjGp8Q)

**Diagrama de flujo:**

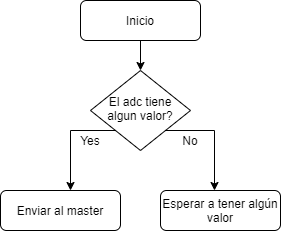


**Pseudocódigo:**

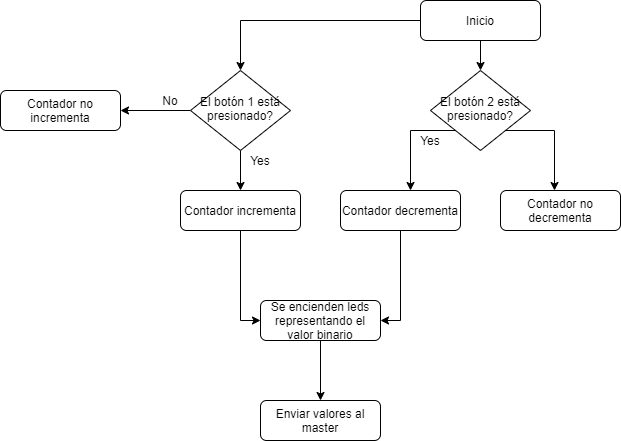
**Master:**



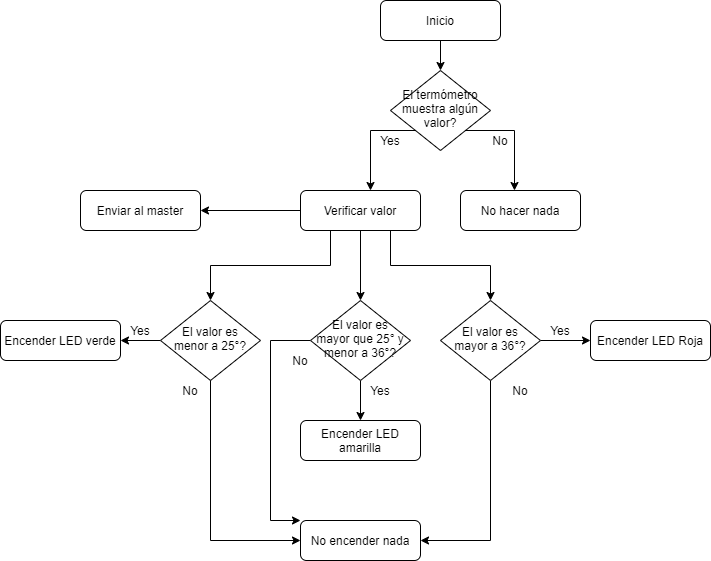
**Slave 1:**



**Slave 2:**



**Slave 3:**



**Código:**

**Librerías Master:**

**Headers:**

**#ifndef LCDM\_H**

**#define LCDM\_H**

**#include <xc.h> // include processor files - each processor file is guarded.**

**#include <stdint.h>**

**#define \_XTAL\_FREQ 4000000**

**#ifndef EN**

**#define EN PORTEbits.RE0**

**#endif**

**#ifndef RS**

**#define RS PORTEbits.RE1**

**#endif**

**#ifndef RW**

**#define RW**

**#endif**

**void Lcd\_Port(char a);**

**void Lcd\_Cmd(char a);**

**void Lcd\_Set\_Cursor(char a, char b);**

**void Lcd\_Init();**

**void Lcd\_Write\_Char(char a);**

**void Lcd\_Write\_String(char \*a);**

**void Lcd\_Shift\_Right();**

**void Lcd\_Shift\_Left();**

**#endif /\* LCDM\_H \*/**

**// This is a guard condition so that contents of this file are not included**

**// more than once.**

**#ifndef OSCM\_H**

**#define OSCM\_H**

**#include <xc.h> // include processor files - each processor file is guarded.**

**#include <stdint.h>**

**void initOscm(uint8\_t IRCF);**

**#endif /\* OSCM\_H \*/**

**/\***

**\* File : spi.h**

**\* Author : Ligo George**

**\* Company : electroSome**

**\* Project : SPI Library for MPLAB XC8**

**\* Microcontroller : PIC 16F877A**

**\* Created on April 15, 2017, 5:59 PM**

**\*/**

**// This is a guard condition so that contents of this file are not included**

**// more than once.**

**#ifndef \_\_SSP\_H**

**#define \_\_SSP\_H**

**#include <xc.h> // include processor files - each processor file is guarded.**

**#include <pic16f887.h>**

**typedef enum**

**{**

**SPI\_MASTER\_OSC\_DIV4 = 0b00100000,**

**SPI\_MASTER\_OSC\_DIV16 = 0b00100001,**

**SPI\_MASTER\_OSC\_DIV64 = 0b00100010,**

**SPI\_MASTER\_TMR2 = 0b00100011,**

**SPI\_SLAVE\_SS\_EN = 0b00100100,**

**SPI\_SLAVE\_SS\_DIS = 0b00100101**

**}Spi\_Type;**

**typedef enum**

**{**

**SPI\_DATA\_SAMPLE\_MIDDLE = 0b00000000,**

**SPI\_DATA\_SAMPLE\_END = 0b10000000**

**}Spi\_Data\_Sample;**

**typedef enum**

**{**

**SPI\_CLOCK\_IDLE\_HIGH = 0b00010000,**

**SPI\_CLOCK\_IDLE\_LOW = 0b00000000**

**}Spi\_Clock\_Idle;**

**typedef enum**

**{**

**SPI\_IDLE\_2\_ACTIVE = 0b00000000,**

**SPI\_ACTIVE\_2\_IDLE = 0b01000000**

**}Spi\_Transmit\_Edge;**

**void spiInit(Spi\_Type, Spi\_Data\_Sample, Spi\_Clock\_Idle, Spi\_Transmit\_Edge);**

**void spiWrite(char);**

**unsigned spiDataReady();**

**char spiRead();**

**#endif /\* SSP\_H \*/**

**//#ifndef SSP\_H**

**//#define SSP\_H**

**//**

**//#include <xc.h> // include processor files - each processor file is guarded.**

**//#include <stdint.h>**

**//**

**////void configSSP(uint8\_t sspm, uint8\_t ckp, uint8\_t cke, uint8\_t smp);**

**//**

**//**

**//#endif /\* SSP\_H \*/**

**#ifndef USARTM\_H**

**#define USARTM\_H**

**#include <xc.h> // include processor files - each processor file is guarded.**

**#include <stdint.h>**

**void usartm(void);**

**#endif /\* USARTM\_H \*/**

**C:**

**/\***

**\* File: LCD.c**

**\* Author: Extraído de electrosome.com**

**\***

**\* Created on 4 de febrero de 2021, 12:52 PM**

**\*/**

**#include <xc.h>**

**#include <stdint.h>**

**#include "LCDM.h"**

**void Lcd\_Port(char a) {**

**PORTA = a;**

**}**

**void Lcd\_Cmd(char a) {**

**Lcd\_Port(a);**

**RS = 0; // => RS = 0**

**EN = 1; // => E = 1**

**\_\_delay\_ms(5);**

**EN = 0; // => E = 0**

**}**

**Lcd\_Clear() {**

**Lcd\_Cmd(0);**

**Lcd\_Cmd(1);**

**}**

**void Lcd\_Set\_Cursor(char a, char b) {**

**char temp;**

**if (a == 1) {**

**temp = 0x80 + b - 1;**

**Lcd\_Cmd(temp);**

**} else if (a == 2) {**

**temp = 0xC0 + b - 1;**

**Lcd\_Cmd(temp);**

**}**

**}**

**void Lcd\_Init() {**

**/////////////////////////////////////////////////////**

**Lcd\_Cmd(0x38);**

**Lcd\_Cmd(0x0C);**

**Lcd\_Cmd(0x06);**

**Lcd\_Cmd(0x80);**

**}**

**void Lcd\_Write\_Char(char a) {**

**RS = 1; // => RS = 1**

**Lcd\_Port(a); //Data transfer**

**EN = 1;**

**\_\_delay\_us(40);**

**EN = 0;**

**RS = 0;**

**}**

**void Lcd\_Write\_String(char \*a) {**

**int i;**

**for(i=0;a[i]!='\0';i++)**

**Lcd\_Write\_Char(a[i]);**

**}**

**void Lcd\_Shift\_Right() {**

**Lcd\_Cmd(0x1C);**

**}**

**void Lcd\_Shift\_Left() {**

**Lcd\_Cmd(0x18);**

**}**

**#include <pic16f887.h>**

**#include <xc.h>**

**#include "oscm.h"**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**//Inicialización del oscilador interno pg. 62**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**void initOscm(uint8\_t IRCF){**

**switch (IRCF){**

**case 0: //OSCILADOR DE 31 kHz**

**OSCCONbits.IRCF2 = 0;**

**OSCCONbits.IRCF1 = 0;**

**OSCCONbits.IRCF0 = 0;**

**break;**

**case 1: //OSCILADOR DE 125 kHz**

**OSCCONbits.IRCF2 = 0;**

**OSCCONbits.IRCF1 = 0;**

**OSCCONbits.IRCF0 = 1;**

**break;**

**case 2: //OSCILADOR DE 250 kHz**

**OSCCONbits.IRCF2 = 0;**

**OSCCONbits.IRCF1 = 1;**

**OSCCONbits.IRCF0 = 0;**

**break;**

**case 3: //OSCILADOR DE 500kHz**

**OSCCONbits.IRCF2 = 0;**

**OSCCONbits.IRCF1 = 1;**

**OSCCONbits.IRCF0 = 1;**

**break;**

**case 4: //OSCILADOR DE 1MHz**

**OSCCONbits.IRCF2 = 1;**

**OSCCONbits.IRCF1 = 0;**

**OSCCONbits.IRCF0 = 0;**

**break;**

**case 5: //OSCILADOR DE 2MHz**

**OSCCONbits.IRCF2 = 1;**

**OSCCONbits.IRCF1 = 0;**

**OSCCONbits.IRCF0 = 1;**

**break;**

**case 6: //OSCILADOR DE 4MHz**

**OSCCONbits.IRCF2 = 1;**

**OSCCONbits.IRCF1 = 1;**

**OSCCONbits.IRCF0 = 0;**

**break;**

**case 7: //OSCILADOR DE 8MHz**

**OSCCONbits.IRCF2 = 1;**

**OSCCONbits.IRCF1 = 1;**

**OSCCONbits.IRCF0 = 1;**

**break;**

**default: //OSCILADOR DE 4MHz**

**OSCCONbits.IRCF2 = 1;**

**OSCCONbits.IRCF1 = 1;**

**OSCCONbits.IRCF0 = 0;**

**break;**

**}**

**OSCCONbits.SCS = 1; //SE VA A USAR EL OSCILADOR INTERNO**

**}**

**/\***

**\* File : spi.c**

**\* Author : Ligo George**

**\* Company : electroSome**

**\* Project : SPI Library for MPLAB XC8**

**\* Microcontroller : PIC 16F877A**

**\* Created on April 15, 2017, 5:59 PM**

**\*/**

**#include <pic16f887.h>**

**#include <xc.h>**

**#include "SSP.h"**

**void spiInit(Spi\_Type sType, Spi\_Data\_Sample sDataSample, Spi\_Clock\_Idle sClockIdle, Spi\_Transmit\_Edge sTransmitEdge)**

**{**

**TRISC5 = 0;**

**if(sType & 0b00000100) //If Slave Mode**

**{**

**SSPSTAT = sTransmitEdge;**

**TRISC3 = 1;**

**}**

**else //If Master Mode**

**{**

**SSPSTAT = sDataSample | sTransmitEdge;**

**TRISC3 = 0;**

**}**

**SSPCON = sType | sClockIdle;**

**}**

**static void spiReceiveWait()**

**{**

**while ( !SSPSTATbits.BF ); // Wait for Data Receive complete**

**}**

**void spiWrite(char dat) //Write data to SPI bus**

**{**

**SSPBUF = dat;**

**}**

**unsigned spiDataReady() //Check whether the data is ready to read**

**{**

**if(SSPSTATbits.BF){**

**return 1;**

**}**

**else{**

**return 0;**

**}**

**}**

**char spiRead() //REad the received data**

**{**

**spiReceiveWait(); // wait until the all bits receive**

**return(SSPBUF); // read the received data from the buffer**

**}**

**//#include <pic16f887.h>**

**//#include <xc.h>**

**//#include "SSP.h"**

**//**

**//void configSSP(uint8\_t sspm, uint8\_t ckp, uint8\_t cke, uint8\_t smp) {**

**// SSPCONbits.SSPEN = 1;**

**// switch (sspm) {**

**//**

**// case 0: //MASTER, FOSC/4**

**// SSPCONbits.SSPM3 = 0;**

**// SSPCONbits.SSPM2 = 0;**

**// SSPCONbits.SSPM1 = 0;**

**// SSPCONbits.SSPM0 = 0;**

**// switch (SMP) {**

**// case 0:**

**// SSPSTATbits.SMP = 0;**

**// break;**

**//**

**// case 1:**

**// SSPSTATbits.SMP = 1;**

**// break;**

**// }**

**// break;**

**//**

**// case 1: //MASTER, FOSC/16**

**// SSPCONbits.SSPM3 = 0;**

**// SSPCONbits.SSPM2 = 0;**

**// SSPCONbits.SSPM1 = 0;**

**// SSPCONbits.SSPM0 = 1;**

**// switch (SMP) {**

**// case 0:**

**// SSPSTATbits.SMP = 0;**

**// break;**

**//**

**// case 1:**

**// SSPSTATbits.SMP = 1;**

**// break;**

**// }**

**// break;**

**//**

**// case 2: //MASTER, FOSC/64**

**// SSPCONbits.SSPM3 = 0;**

**// SSPCONbits.SSPM2 = 0;**

**// SSPCONbits.SSPM1 = 1;**

**// SSPCONbits.SSPM0 = 0;**

**// switch (SMP) {**

**// case 0:**

**// SSPSTATbits.SMP = 0;**

**// break;**

**//**

**// case 1:**

**// SSPSTATbits.SMP = 1;**

**// break;**

**// }**

**// break;**

**//**

**// case 3: //MASTER, TMR2 OUTPUT/2**

**// SSPCONbits.SSPM3 = 0;**

**// SSPCONbits.SSPM2 = 0;**

**// SSPCONbits.SSPM1 = 1;**

**// SSPCONbits.SSPM0 = 1;**

**// switch (SMP) {**

**// case 0:**

**// SSPSTATbits.SMP = 0;**

**// break;**

**//**

**// case 1:**

**// SSPSTATbits.SMP = 1;**

**// break;**

**// }**

**// break;**

**//**

**// case 4: //SLAVE, SCK,SS ENABLED**

**// SSPCONbits.SSPM3 = 0;**

**// SSPCONbits.SSPM2 = 1;**

**// SSPCONbits.SSPM1 = 0;**

**// SSPCONbits.SSPM0 = 0;**

**// switch (SMP) {**

**// case 0:**

**// SSPSTATbits.SMP = 0;**

**// break;**

**// }**

**// break;**

**//**

**//**

**// case 5: //SLAVE, SCK, SS DISABLED**

**// SSPCONbits.SSPM3 = 0;**

**// SSPCONbits.SSPM2 = 1;**

**// SSPCONbits.SSPM1 = 0;**

**// SSPCONbits.SSPM0 = 1;**

**//**

**// switch (SMP) {**

**// case 0:**

**// SSPSTATbits.SMP = 0;**

**// break;**

**// }**

**// break;**

**//**

**// default: //MASTER, FOSC/64**

**// SSPCONbits.SSPM3 = 0;**

**// SSPCONbits.SSPM2 = 0;**

**// SSPCONbits.SSPM1 = 1;**

**// SSPCONbits.SSPM0 = 0;**

**// switch (SMP) {**

**// case 0:**

**// SSPSTATbits.SMP = 0;**

**// break;**

**//**

**// case 1:**

**// SSPSTATbits.SMP = 1;**

**// break;**

**// }**

**// break;**

**//**

**// }**

**//**

**// switch (ckp) {**

**// case 0:**

**// SSPCONbits.CKP = 0;**

**// switch (CKE) {**

**// case 0:**

**// SSPSTATbits.CKE = 1;**

**// break;**

**// case 1:**

**// SSPSTATbits.CKE = 0;**

**// }**

**//**

**// break;**

**// case 1:**

**// SSPCONbits.CKP = 1;**

**// switch (CKE) {**

**// case 0:**

**// SSPSTATbits.CKE = 1;**

**// break;**

**// case 1:**

**// SSPSTATbits.CKE = 0;**

**//**

**// }**

**// break;**

**//**

**// }**

**#include <pic16f887.h>**

**#include "usartm.h"**

**void usartm(void){**

**//CONFIG TX**

**TXSTAbits.TX9 = 0; //TRANSMISION DE 8 BITS**

**TXSTAbits.SYNC = 0; //ASINCRONO**

**TXSTAbits.BRGH = 1; //HIGH SPEED**

**BAUDCTLbits.BRG16 = 0; //BAUD RATE DE 8 BITS**

**SPBRGH = 0;**

**SPBRG = 25;**

**PIE1bits.TXIE = 1;**

**TXSTAbits.TXEN = 1;**

**//CONFIG RX**

**RCSTAbits.SPEN = 1;**

**RCSTAbits. RX9 = 0;**

**RCSTAbits.CREN = 1;**

**}**

**Código Master:**

**/\***

**\* File: newmain.c**

**\* Author: franc**

**\***

**\***

**\*/**

**#include <xc.h>**

**#include <stdint.h>**

**#include <pic16f887.h>**

**#include "LCDM.h"**

**#include "oscm.h"**

**#include "SSP.h"**

**#include "usartm.h"**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**// Palabra de configuración**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**// CONFIG1**

**#pragma config FOSC = INTRC\_NOCLKOUT // Oscillator Selection bits (XT oscillator: Crystal/resonator on RA6/OSC2/CLKOUT and RA7/OSC1/CLKIN)**

**#pragma config WDTE = OFF // Watchdog Timer Enable bit (WDT disabled and can be enabled by SWDTEN bit of the WDTCON register)**

**#pragma config PWRTE = OFF // Power-up Timer Enable bit (PWRT disabled)**

**#pragma config MCLRE = OFF // RE3/MCLR pin function select bit (RE3/MCLR pin function is digital input, MCLR internally tied to VDD)**

**#pragma config CP = OFF // Code Protection bit (Program memory code protection is disabled)**

**#pragma config CPD = OFF // Data Code Protection bit (Data memory code protection is disabled)**

**#pragma config BOREN = OFF // Brown Out Reset Selection bits (BOR disabled)**

**#pragma config IESO = OFF // Internal External Switchover bit (Internal/External Switchover mode is disabled)**

**#pragma config FCMEN = OFF // Fail-Safe Clock Monitor Enabled bit (Fail-Safe Clock Monitor is disabled)**

**#pragma config LVP = OFF // Low Voltage Programming Enable bit (RB3 pin has digital I/O, HV on MCLR must be used for programming)**

**// CONFIG2**

**#pragma config BOR4V = BOR40V // Brown-out Reset Selection bit (Brown-out Reset set to 4.0V)**

**#pragma config WRT = OFF // Flash Program Memory Self Write Enable bits (Write protection off)**

**#define \_XTAL\_FREQ 4000000 //SE CONFIGURA EL OSCILADOR EXTERNO**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**//Variables**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**uint8\_t desecho = 0;**

**uint8\_t pot1 = 0;**

**uint8\_t cont1 = 0;**

**uint8\_t CP1 = 0;**

**uint8\_t DP1 = 0;**

**uint8\_t UP1 = 0;**

**uint8\_t C1 = 0;**

**uint8\_t D1 = 0;**

**uint8\_t U1 = 0;**

**uint8\_t INDIC = 0;**

**uint8\_t CONTC = 0;**

**uint8\_t CONTD = 0;**

**uint8\_t CONTU = 0;**

**uint8\_t CO1 = 0;**

**uint8\_t CO2 = 0;**

**uint8\_t CO3 = 0;**

**uint8\_t SEND = 0;**

**uint8\_t term1 = 0;**

**uint8\_t TEMPC = 0;**

**uint8\_t TEMPD = 0;**

**uint8\_t TEMPU = 0;**

**uint8\_t TEMPND = 0;**

**uint8\_t TEMPNU = 0;**

**uint8\_t T1 = 0;**

**uint8\_t T2 = 0;**

**uint8\_t T3 = 0;**

**uint8\_t TN1 = 0;**

**uint8\_t TN2 = 0;**

**uint8\_t TEMP = 0;**

**uint8\_t TEMPN = 0;**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**// Prototipos de funciones**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**void Setup(void);**

**void map(void);**

**void map2(void);**

**void map3(void);**

**void map4(void);**

**void mandar(void);**

**void temperatura(void);**

**void temperatura2(void);**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**//Interrupción**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**void \_\_interrupt() ISR(void) {**

**if (PIR1bits.TXIF == 1) {**

**mandar();**

**SEND++;**

**PIE1bits.TXIE = 0;**

**PIR1bits.TXIF = 0;**

**}**

**}**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**//Ciclo pincipal**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**void main(void) {**

**Setup();**

**Lcd\_Set\_Cursor(1, 1);**

**Lcd\_Write\_String("ADC");**

**Lcd\_Set\_Cursor(1, 7);**

**Lcd\_Write\_String("CONT");**

**Lcd\_Set\_Cursor(1, 13);**

**Lcd\_Write\_String("TEMP");**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**// Loop principal**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**while (1) {**

**INDIC++;**

**map();**

**map2();**

**map3();**

**PIE1bits.TXIE = 1;**

**if (INDIC == 1) {**

**PORTCbits.RC0 = 0;**

**\_\_delay\_ms(1);**

**spiWrite(desecho);**

**pot1 = spiRead();**

**\_\_delay\_ms(1);**

**PORTCbits.RC0 = 1;**

**}**

**if (INDIC == 2) {**

**PORTCbits.RC1 = 0;**

**\_\_delay\_ms(1);**

**spiWrite(desecho);**

**cont1 = spiRead();**

**\_\_delay\_ms(1);**

**PORTCbits.RC1 = 1;**

**}**

**if (INDIC == 3) {**

**PORTCbits.RC2 = 0;**

**\_\_delay\_ms(1);**

**spiWrite(desecho);**

**term1 = spiRead();**

**\_\_delay\_ms(1);**

**PORTCbits.RC2 = 1;**

**INDIC = 0;**

**}**

**Lcd\_Set\_Cursor(2, 1);**

**Lcd\_Write\_Char(C1);**

**Lcd\_Set\_Cursor(2, 2);**

**Lcd\_Write\_String(".");**

**Lcd\_Write\_Char(D1);**

**Lcd\_Set\_Cursor(2, 4);**

**Lcd\_Write\_Char(U1);**

**Lcd\_Set\_Cursor(2, 7);**

**Lcd\_Write\_Char(CO1);**

**Lcd\_Set\_Cursor(2, 8);**

**Lcd\_Write\_Char(CO2);**

**Lcd\_Set\_Cursor(2, 9);**

**Lcd\_Write\_Char(CO3);**

**}**

**}**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**//Configuracion**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**void Setup(void) {**

**TRISA = 0;**

**TRISE = 0; //PUERTO E SALIDAS**

**initOscm(6);**

**usartm();**

**Lcd\_Init();**

**Lcd\_Cmd(0x8A);**

**spiInit(SPI\_MASTER\_OSC\_DIV4, SPI\_DATA\_SAMPLE\_MIDDLE, SPI\_CLOCK\_IDLE\_LOW, SPI\_IDLE\_2\_ACTIVE);**

**ANSEL = 0; // ENTRADAS DIGITALES Y BIT 0 ANALÓGICA**

**ANSELH = 0;**

**PORTA = 0; //PUERTO A EN 0**

**PORTB = 0; //PUERTO B EN 0**

**PORTC = 0; //PUERTO C EN 0**

**PORTD = 0; //PUERTO D EN 0**

**PORTE = 0; //PUERTO E EN 0**

**//PINES RA0 Y RA2 COMO ENTRADAS, LOS DEMAS COMO SALIDAS**

**TRISC = 0b00010000; //PUERTO C SALIDAS**

**TRISD = 0; //PUERTO D SALIDAS**

**TRISB = 0; //PUERTO B**

**OPTION\_REG = 0b10000111; //SE APAGAN LAS PULLUPS DEL PUERTO B**

**INTCONbits.GIE = 1; //SE HABILITAN LAS INTERRUPCIONES GLOBALES**

**// INTCONbits.T0IE = 1; //SE HABILITA LA INTERRUPCION DEL TIMER0**

**INTCONbits.PEIE = 1; //SE HABILITAN LAS INTERRUPCIONES PERIFERICAS**

**// PIE1bits.ADIE = 1; //SE HABILITA LA INTERRUPCION DEL ADC**

**// INTCONbits.T0IF = 0; // SE LIMPIA LA BANDERA DE INTERRUPCION DEL TIMER 0**

**// PIR1bits.ADIF = 0; //SE LIMPIOA LA BANDERA DE INTERRUPCION DEL ADC**

**PIR1bits.TXIF = 0;**

**PIE1bits.TXIE = 1;**

**}**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**// Subrutinas**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**void map(void) {**

**CP1 = ((pot1) / 51);**

**DP1 = (((pot1 \* 100) / 51)-(CP1 \* 100)) / 10;**

**UP1 = (((pot1 \* 100) / 51)-(CP1 \* 100)-(DP1 \* 10));**

**C1 = (CP1 + 0x30);**

**D1 = (DP1 + 0x30);**

**U1 = (UP1 + 0x30);**

**}**

**void map2(void) {**

**CONTC = (cont1 / 100);**

**CONTD = (cont1 - (CONTC \* 100)) / 10;**

**CONTU = (cont1 - (CONTC \* 100)-(CONTD \* 10));**

**CO1 = (CONTC + 0x30);**

**CO2 = (CONTD + 0x30);**

**CO3 = (CONTU + 0x30);**

**}**

**void map3(void) {**

**if (term1 >= 68) {**

**TEMP = (((term1 - 68)\*150) / 187);**

**TEMPC = (TEMP / 100);**

**TEMPD = (TEMP - (TEMPC \* 100)) / 10;**

**TEMPU = (TEMP - (TEMPC \* 100)-(TEMPD \* 10));**

**T1 = (TEMPC + 0x30);**

**T2 = (TEMPD + 0x30);**

**T3 = (TEMPU + 0x30);**

**temperatura();**

**} else if (term1 < 68) {**

**TEMPN = (((term1 \* (-55)) / 68) + 55);**

**TEMPND = (TEMPN / 10);**

**TEMPNU = (TEMPN - (TEMPND \* 10));**

**TN1 = (TEMPND + 0x30);**

**TN2 = (TEMPNU + 0x30);**

**temperatura2();**

**}**

**}**

**void mandar(void) {**

**switch (SEND) {**

**case 0:**

**TXREG = 0x20;**

**break;**

**case 1:**

**TXREG = 0x28;**

**break;**

**case 2:**

**TXREG = C1;**

**break;**

**case 3:**

**TXREG = 0x2E;**

**break;**

**case 4:**

**TXREG = D1;**

**break;**

**case 5:**

**TXREG = U1;**

**break;**

**case 6:**

**TXREG = 0x29;**

**break;**

**case 7:**

**TXREG = 0x2C;**

**break;**

**case 8:**

**TXREG = 0x20;**

**break;**

**case 9:**

**TXREG = 0x28;**

**break;**

**case 10:**

**TXREG = CO1;**

**break;**

**case 11:**

**TXREG = CO2;**

**break;**

**case 12:**

**TXREG = CO3;**

**break;**

**case 13:**

**TXREG = 0x29;**

**break;**

**case 14:**

**TXREG = 0x2C;**

**break;**

**case 15:**

**TXREG = 0x20;**

**break;**

**case 16:**

**TXREG = 0x28;**

**break;**

**case 17:**

**if (term1 >= 68) {**

**TXREG = T1;**

**} else if (term1 < 68) {**

**TXREG = 45;**

**}**

**break;**

**case 18:**

**if (term1 >= 68) {**

**TXREG = T2;**

**} else if (term1 < 68) {**

**TXREG = TN1;**

**}**

**break;**

**case 19:**

**if (term1 >= 68) {**

**TXREG = T3;**

**} else if (term1 < 68) {**

**TXREG = TN2;**

**}**

**break;**

**case 20:**

**TXREG = 0x29;**

**break;**

**case 21:**

**TXREG = 0x0D;**

**SEND = 0;**

**break;**

**}**

**}**

**void temperatura(void) {**

**Lcd\_Set\_Cursor(2, 12);**

**Lcd\_Write\_String("+");**

**Lcd\_Set\_Cursor(2, 13);**

**Lcd\_Write\_Char(T1);**

**Lcd\_Set\_Cursor(2, 14);**

**Lcd\_Write\_Char(T2);**

**Lcd\_Set\_Cursor(2, 15);**

**Lcd\_Write\_Char(T3);**

**}**

**void temperatura2(void) {**

**Lcd\_Set\_Cursor(2, 12);**

**Lcd\_Write\_String("-");**

**Lcd\_Set\_Cursor(2, 14);**

**Lcd\_Write\_Char(TN1);**

**Lcd\_Set\_Cursor(2, 15);**

**Lcd\_Write\_Char(TN2);**

**}**

**Librerías slave 1:**

**Headers:**

**/\***

**\* File:**

**\* Author:**

**\* Comments:**

**\* Revision history:**

**\*/**

**// This is a guard condition so that contents of this file are not included**

**// more than once.**

**#ifndef ADCS1\_H**

**#define ADCS1\_H**

**#include <xc.h> // include processor files - each processor file is guarded.**

**#include <stdint.h>**

**void configADC1(uint8\_t fosc, uint8\_t chan);**

**#endif /\* ADCS1\_H\*/**

**// This is a guard condition so that contents of this file are not included**

**// more than once.**

**#ifndef OSCS1\_H**

**#define OSCS1\_H**

**#include <xc.h> // include processor files - each processor file is guarded.**

**#include <stdint.h>**

**void initOscs1(uint8\_t IRCF);**

**#endif /\* OSCS1\_H \*/**

**//\***

**// \* File : spi.h**

**// \* Author : Ligo George**

**// \* Company : electroSome**

**// \* Project : SPI Library for MPLAB XC8**

**// \* Microcontroller : PIC 16F877A**

**// \* Created on April 15, 2017, 5:59 PM**

**// \*/**

**// This is a guard condition so that contents of this file are not included**

**// more than once.**

**#ifndef \_\_SSP1\_H**

**#define \_\_SSP1\_H**

**#include <xc.h> // include processor files - each processor file is guarded.**

**#include <pic16f887.h>**

**typedef enum**

**{**

**SPI\_MASTER\_OSC\_DIV4 = 0b00100000,**

**SPI\_MASTER\_OSC\_DIV16 = 0b00100001,**

**SPI\_MASTER\_OSC\_DIV64 = 0b00100010,**

**SPI\_MASTER\_TMR2 = 0b00100011,**

**SPI\_SLAVE\_SS\_EN = 0b00100100,**

**SPI\_SLAVE\_SS\_DIS = 0b00100101**

**}Spi\_Type;**

**typedef enum**

**{**

**SPI\_DATA\_SAMPLE\_MIDDLE = 0b00000000,**

**SPI\_DATA\_SAMPLE\_END = 0b10000000**

**}Spi\_Data\_Sample;**

**typedef enum**

**{**

**SPI\_CLOCK\_IDLE\_HIGH = 0b00010000,**

**SPI\_CLOCK\_IDLE\_LOW = 0b00000000**

**}Spi\_Clock\_Idle;**

**typedef enum**

**{**

**SPI\_IDLE\_2\_ACTIVE = 0b00000000,**

**SPI\_ACTIVE\_2\_IDLE = 0b01000000**

**}Spi\_Transmit\_Edge;**

**void spiInit(Spi\_Type, Spi\_Data\_Sample, Spi\_Clock\_Idle, Spi\_Transmit\_Edge);**

**void spiWrite(char);**

**unsigned spiDataReady();**

**char spiRead();**

**#endif /\* SSP1\_H \*/**

**C:**

**#include <pic16f887.h>**

**#include <xc.h>**

**#include "adcs1.h"**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**// CONFIGURACION DEL ADC**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**void configADC1(uint8\_t fosc, uint8\_t chan) {**

**switch (fosc) {**

**case 0:**

**ADCON0bits.ADCS = 0b00;**

**break;**

**case 1:**

**ADCON0bits.ADCS = 0b01;**

**break;**

**case 2:**

**ADCON0bits.ADCS = 0b10;**

**break;**

**case 3:**

**ADCON0bits.ADCS = 0b11;**

**break;**

**default:**

**ADCON0bits.ADCS = 0b00;**

**break;**

**}**

**switch (chan) {**

**case 0:**

**ADCON0bits.CHS = 0b0000;**

**break;**

**case 1:**

**ADCON0bits.CHS = 0b0001;**

**break;**

**case 2:**

**ADCON0bits.CHS = 0b0010;**

**break;**

**case 3:**

**ADCON0bits.CHS = 0b0011;**

**break;**

**case 4:**

**ADCON0bits.CHS = 0b0100;**

**break;**

**case 5:**

**ADCON0bits.CHS = 0b0101;**

**break;**

**case 6:**

**ADCON0bits.CHS = 0b0110;**

**break;**

**case 7:**

**ADCON0bits.CHS = 0b0111;**

**break;**

**case 8:**

**ADCON0bits.CHS = 0b1000;**

**break;**

**case 9:**

**ADCON0bits.CHS = 0b1001;**

**break;**

**case 10:**

**ADCON0bits.CHS = 0b1010;**

**break;**

**case 11:**

**ADCON0bits.CHS = 0b1011;**

**break;**

**case 12:**

**ADCON0bits.CHS = 0b1100;**

**break;**

**case 13:**

**ADCON0bits.CHS = 0b1101;**

**break;**

**case 14:**

**ADCON0bits.CHS = 0b1110;**

**break;**

**case 15:**

**ADCON0bits.CHS = 0b1111;**

**break;**

**default:**

**ADCON0bits.CHS = 0b0000;**

**break;**

**}**

**ADCON0bits.GO\_nDONE = 1;**

**ADCON0bits.ADON = 1;**

**ADCON1 = 0;**

**}**

**#include <pic16f887.h>**

**#include <xc.h>**

**#include "oscs1.h"**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**//Inicialización del oscilador interno pg. 62**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**void initOscs1(uint8\_t IRCF){**

**switch (IRCF){**

**case 0: //OSCILADOR DE 31 kHz**

**OSCCONbits.IRCF2 = 0;**

**OSCCONbits.IRCF1 = 0;**

**OSCCONbits.IRCF0 = 0;**

**break;**

**case 1: //OSCILADOR DE 125 kHz**

**OSCCONbits.IRCF2 = 0;**

**OSCCONbits.IRCF1 = 0;**

**OSCCONbits.IRCF0 = 1;**

**break;**

**case 2: //OSCILADOR DE 250 kHz**

**OSCCONbits.IRCF2 = 0;**

**OSCCONbits.IRCF1 = 1;**

**OSCCONbits.IRCF0 = 0;**

**break;**

**case 3: //OSCILADOR DE 500kHz**

**OSCCONbits.IRCF2 = 0;**

**OSCCONbits.IRCF1 = 1;**

**OSCCONbits.IRCF0 = 1;**

**break;**

**case 4: //OSCILADOR DE 1MHz**

**OSCCONbits.IRCF2 = 1;**

**OSCCONbits.IRCF1 = 0;**

**OSCCONbits.IRCF0 = 0;**

**break;**

**case 5: //OSCILADOR DE 2MHz**

**OSCCONbits.IRCF2 = 1;**

**OSCCONbits.IRCF1 = 0;**

**OSCCONbits.IRCF0 = 1;**

**break;**

**case 6: //OSCILADOR DE 4MHz**

**OSCCONbits.IRCF2 = 1;**

**OSCCONbits.IRCF1 = 1;**

**OSCCONbits.IRCF0 = 0;**

**break;**

**case 7: //OSCILADOR DE 8MHz**

**OSCCONbits.IRCF2 = 1;**

**OSCCONbits.IRCF1 = 1;**

**OSCCONbits.IRCF0 = 1;**

**break;**

**default: //OSCILADOR DE 4MHz**

**OSCCONbits.IRCF2 = 1;**

**OSCCONbits.IRCF1 = 1;**

**OSCCONbits.IRCF0 = 0;**

**break;**

**}**

**OSCCONbits.SCS = 1; //SE VA A USAR EL OSCILADOR INTERNO**

**}**

**/\***

**\* File : spi.c**

**\* Author : Ligo George**

**\* Company : electroSome**

**\* Project : SPI Library for MPLAB XC8**

**\* Microcontroller : PIC 16F877A**

**\* Created on April 15, 2017, 5:59 PM**

**\*/**

**#include <pic16f887.h>**

**#include <xc.h>**

**#include "SSP1.h"**

**void spiInit(Spi\_Type sType, Spi\_Data\_Sample sDataSample, Spi\_Clock\_Idle sClockIdle, Spi\_Transmit\_Edge sTransmitEdge)**

**{**

**TRISC5 = 0;**

**if(sType & 0b00000100) //If Slave Mode**

**{**

**SSPSTAT = sTransmitEdge;**

**TRISC3 = 1;**

**}**

**else //If Master Mode**

**{**

**SSPSTAT = sDataSample | sTransmitEdge;**

**TRISC3 = 0;**

**}**

**SSPCON = sType | sClockIdle;**

**}**

**static void spiReceiveWait()**

**{**

**while ( !SSPSTATbits.BF ); // Wait for Data Receive complete**

**}**

**void spiWrite(char dat) //Write data to SPI bus**

**{**

**SSPBUF = dat;**

**}**

**unsigned spiDataReady() //Check whether the data is ready to read**

**{**

**if(SSPSTATbits.BF){**

**return 1;**

**}**

**else{**

**return 0;**

**}**

**}**

**char spiRead() //REad the received data**

**{**

**spiReceiveWait(); // wait until the all bits receive**

**return(SSPBUF); // read the received data from the buffer**

**}**

**Código slave 1:**

**#include <xc.h>**

**#include <stdint.h>**

**#include <pic16f887.h>**

**#include "adcs1.h"**

**#include "oscs1.h"**

**#include "SSP1.h"**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**// Palabra de configuración**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**// CONFIG1**

**#pragma config FOSC = INTRC\_NOCLKOUT // Oscillator Selection bits (XT oscillator: Crystal/resonator on RA6/OSC2/CLKOUT and RA7/OSC1/CLKIN)**

**#pragma config WDTE = OFF // Watchdog Timer Enable bit (WDT disabled and can be enabled by SWDTEN bit of the WDTCON register)**

**#pragma config PWRTE = OFF // Power-up Timer Enable bit (PWRT disabled)**

**#pragma config MCLRE = OFF // RE3/MCLR pin function select bit (RE3/MCLR pin function is digital input, MCLR internally tied to VDD)**

**#pragma config CP = OFF // Code Protection bit (Program memory code protection is disabled)**

**#pragma config CPD = OFF // Data Code Protection bit (Data memory code protection is disabled)**

**#pragma config BOREN = OFF // Brown Out Reset Selection bits (BOR disabled)**

**#pragma config IESO = OFF // Internal External Switchover bit (Internal/External Switchover mode is disabled)**

**#pragma config FCMEN = OFF // Fail-Safe Clock Monitor Enabled bit (Fail-Safe Clock Monitor is disabled)**

**#pragma config LVP = OFF // Low Voltage Programming Enable bit (RB3 pin has digital I/O, HV on MCLR must be used for programming)**

**// CONFIG2**

**#pragma config BOR4V = BOR40V // Brown-out Reset Selection bit (Brown-out Reset set to 4.0V)**

**#pragma config WRT = OFF // Flash Program Memory Self Write Enable bits (Write protection off)**

**#define \_XTAL\_FREQ 4000000 //SE CONFIGURA EL OSCILADOR EXTERNO**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**//Variables**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**uint8\_t pot = 0;**

**uint8\_t CONTADC = 0;**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**// Prototipos de funciones**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**void Setup(void);**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**//Interrupción**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**void \_\_interrupt() ISR(void) {**

**if (INTCONbits.T0IF == 1) {**

**TMR0 = 236;**

**CONTADC++;**

**INTCONbits.T0IF = 0;**

**}**

**if (PIR1bits.SSPIF == 1) {**

**spiWrite(pot);**

**PIR1bits.SSPIF = 0;**

**}**

**if (PIR1bits.ADIF == 1) {**

**pot = ADRESH;**

**PIR1bits.ADIF = 0;**

**}**

**}**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**//Ciclo pincipal**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**void main(void) {**

**Setup();**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**// Loop principal**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**while (1) {**

**if (CONTADC > 20) {**

**ADCON0bits.GO\_nDONE = 1;**

**CONTADC = 0;**

**}**

**PORTD = pot;**

**}**

**}**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**//Configuracion**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**void Setup(void) {**

**configADC1(1, 12); //SE LLAMA LA CONFIG DEL ADC**

**initOscs1(6);**

**spiInit(SPI\_SLAVE\_SS\_EN, SPI\_DATA\_SAMPLE\_MIDDLE, SPI\_CLOCK\_IDLE\_LOW, SPI\_IDLE\_2\_ACTIVE);**

**ANSEL = 0; // ENTRADAS DIGITALES Y BIT 0 ANALÓGICA**

**ANSELH = 0b00000001;**

**PORTA = 0; //PUERTO A EN 0**

**PORTB = 0; //PUERTO B EN 0**

**PORTC = 0; //PUERTO C EN 0**

**PORTD = 0; //PUERTO D EN 0**

**PORTE = 0; //PUERTO E EN 0**

**//PINES RA0 Y RA2 COMO ENTRADAS, LOS DEMAS COMO SALIDAS**

**TRISC = 0b00001000; //PUERTO C SALIDAS**

**TRISA = 0b00100000; //PUERTO A SALIDAS**

**TRISB = 0b00000001; //PUERTO B**

**TRISD = 0;**

**TRISE = 0;**

**OPTION\_REG = 0b10000111; //SE APAGAN LAS PULLUPS DEL PUERTO B**

**INTCONbits.GIE = 1; //SE HABILITAN LAS INTERRUPCIONES GLOBALES**

**INTCONbits.PEIE = 1; //SE HABILITAN LAS INTERRUPCIONES PERIFERICAS**

**PIE1bits.ADIE = 1; //SE HABILITA LA INTERRUPCION DEL ADC**

**PIR1bits.ADIF = 0; //SE LIMPIOA LA BANDERA DE INTERRUPCION DEL ADC**

**PIR1bits.SSPIF = 0;**

**PIE1bits.SSPIE = 1;**

**INTCONbits.T0IE = 1;**

**INTCONbits.T0IF = 0;**

**}**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**// Subrutinas**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Librerías slave 2:**

**Headers:**

**#include <xc.h>**

**#include <stdint.h>**

**#include <pic16f887.h>**

**#include "adcs1.h"**

**#include "oscs1.h"**

**#include "SSP1.h"**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**// Palabra de configuración**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**// CONFIG1**

**#pragma config FOSC = INTRC\_NOCLKOUT // Oscillator Selection bits (XT oscillator: Crystal/resonator on RA6/OSC2/CLKOUT and RA7/OSC1/CLKIN)**

**#pragma config WDTE = OFF // Watchdog Timer Enable bit (WDT disabled and can be enabled by SWDTEN bit of the WDTCON register)**

**#pragma config PWRTE = OFF // Power-up Timer Enable bit (PWRT disabled)**

**#pragma config MCLRE = OFF // RE3/MCLR pin function select bit (RE3/MCLR pin function is digital input, MCLR internally tied to VDD)**

**#pragma config CP = OFF // Code Protection bit (Program memory code protection is disabled)**

**#pragma config CPD = OFF // Data Code Protection bit (Data memory code protection is disabled)**

**#pragma config BOREN = OFF // Brown Out Reset Selection bits (BOR disabled)**

**#pragma config IESO = OFF // Internal External Switchover bit (Internal/External Switchover mode is disabled)**

**#pragma config FCMEN = OFF // Fail-Safe Clock Monitor Enabled bit (Fail-Safe Clock Monitor is disabled)**

**#pragma config LVP = OFF // Low Voltage Programming Enable bit (RB3 pin has digital I/O, HV on MCLR must be used for programming)**

**// CONFIG2**

**#pragma config BOR4V = BOR40V // Brown-out Reset Selection bit (Brown-out Reset set to 4.0V)**

**#pragma config WRT = OFF // Flash Program Memory Self Write Enable bits (Write protection off)**

**#define \_XTAL\_FREQ 4000000 //SE CONFIGURA EL OSCILADOR EXTERNO**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**//Variables**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**uint8\_t pot = 0;**

**uint8\_t CONTADC = 0;**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**// Prototipos de funciones**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**void Setup(void);**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**//Interrupción**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**void \_\_interrupt() ISR(void) {**

**if (INTCONbits.T0IF == 1) {**

**TMR0 = 236;**

**CONTADC++;**

**INTCONbits.T0IF = 0;**

**}**

**if (PIR1bits.SSPIF == 1) {**

**spiWrite(pot);**

**PIR1bits.SSPIF = 0;**

**}**

**if (PIR1bits.ADIF == 1) {**

**pot = ADRESH;**

**PIR1bits.ADIF = 0;**

**}**

**}**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**//Ciclo pincipal**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**void main(void) {**

**Setup();**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**// Loop principal**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**while (1) {**

**if (CONTADC > 20) {**

**ADCON0bits.GO\_nDONE = 1;**

**CONTADC = 0;**

**}**

**PORTD = pot;**

**}**

**}**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**//Configuracion**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**void Setup(void) {**

**configADC1(1, 12); //SE LLAMA LA CONFIG DEL ADC**

**initOscs1(6);**

**spiInit(SPI\_SLAVE\_SS\_EN, SPI\_DATA\_SAMPLE\_MIDDLE, SPI\_CLOCK\_IDLE\_LOW, SPI\_IDLE\_2\_ACTIVE);**

**ANSEL = 0; // ENTRADAS DIGITALES Y BIT 0 ANALÓGICA**

**ANSELH = 0b00000001;**

**PORTA = 0; //PUERTO A EN 0**

**PORTB = 0; //PUERTO B EN 0**

**PORTC = 0; //PUERTO C EN 0**

**PORTD = 0; //PUERTO D EN 0**

**PORTE = 0; //PUERTO E EN 0**

**//PINES RA0 Y RA2 COMO ENTRADAS, LOS DEMAS COMO SALIDAS**

**TRISC = 0b00001000; //PUERTO C SALIDAS**

**TRISA = 0b00100000; //PUERTO A SALIDAS**

**TRISB = 0b00000001; //PUERTO B**

**TRISD = 0;**

**TRISE = 0;**

**OPTION\_REG = 0b10000111; //SE APAGAN LAS PULLUPS DEL PUERTO B**

**INTCONbits.GIE = 1; //SE HABILITAN LAS INTERRUPCIONES GLOBALES**

**INTCONbits.PEIE = 1; //SE HABILITAN LAS INTERRUPCIONES PERIFERICAS**

**PIE1bits.ADIE = 1; //SE HABILITA LA INTERRUPCION DEL ADC**

**PIR1bits.ADIF = 0; //SE LIMPIOA LA BANDERA DE INTERRUPCION DEL ADC**

**PIR1bits.SSPIF = 0;**

**PIE1bits.SSPIE = 1;**

**INTCONbits.T0IE = 1;**

**INTCONbits.T0IF = 0;**

**}**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**// Subrutinas**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**//\***

**// \* File : spi.h**

**// \* Author : Ligo George**

**// \* Company : electroSome**

**// \* Project : SPI Library for MPLAB XC8**

**// \* Microcontroller : PIC 16F877A**

**// \* Created on April 15, 2017, 5:59 PM**

**// \*/**

**// This is a guard condition so that contents of this file are not included**

**// more than once.**

**#ifndef \_\_SSP2\_H**

**#define \_\_SSP2\_H**

**#include <xc.h> // include processor files - each processor file is guarded.**

**#include <pic16f887.h>**

**typedef enum**

**{**

**SPI\_MASTER\_OSC\_DIV4 = 0b00100000,**

**SPI\_MASTER\_OSC\_DIV16 = 0b00100001,**

**SPI\_MASTER\_OSC\_DIV64 = 0b00100010,**

**SPI\_MASTER\_TMR2 = 0b00100011,**

**SPI\_SLAVE\_SS\_EN = 0b00100100,**

**SPI\_SLAVE\_SS\_DIS = 0b00100101**

**}Spi\_Type;**

**typedef enum**

**{**

**SPI\_DATA\_SAMPLE\_MIDDLE = 0b00000000,**

**SPI\_DATA\_SAMPLE\_END = 0b10000000**

**}Spi\_Data\_Sample;**

**typedef enum**

**{**

**SPI\_CLOCK\_IDLE\_HIGH = 0b00010000,**

**SPI\_CLOCK\_IDLE\_LOW = 0b00000000**

**}Spi\_Clock\_Idle;**

**typedef enum**

**{**

**SPI\_IDLE\_2\_ACTIVE = 0b00000000,**

**SPI\_ACTIVE\_2\_IDLE = 0b01000000**

**}Spi\_Transmit\_Edge;**

**void spiInit(Spi\_Type, Spi\_Data\_Sample, Spi\_Clock\_Idle, Spi\_Transmit\_Edge);**

**void spiWrite(char);**

**unsigned spiDataReady();**

**char spiRead();**

**#endif /\* SSP2\_H \*/**

**C:**

**#include <pic16f887.h>**

**#include <xc.h>**

**#include "osc2.h"**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**//Inicialización del oscilador interno pg. 62**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**void initOscs2(uint8\_t IRCF){**

**switch (IRCF){**

**case 0: //OSCILADOR DE 31 kHz**

**OSCCONbits.IRCF2 = 0;**

**OSCCONbits.IRCF1 = 0;**

**OSCCONbits.IRCF0 = 0;**

**break;**

**case 1: //OSCILADOR DE 125 kHz**

**OSCCONbits.IRCF2 = 0;**

**OSCCONbits.IRCF1 = 0;**

**OSCCONbits.IRCF0 = 1;**

**break;**

**case 2: //OSCILADOR DE 250 kHz**

**OSCCONbits.IRCF2 = 0;**

**OSCCONbits.IRCF1 = 1;**

**OSCCONbits.IRCF0 = 0;**

**break;**

**case 3: //OSCILADOR DE 500kHz**

**OSCCONbits.IRCF2 = 0;**

**OSCCONbits.IRCF1 = 1;**

**OSCCONbits.IRCF0 = 1;**

**break;**

**case 4: //OSCILADOR DE 1MHz**

**OSCCONbits.IRCF2 = 1;**

**OSCCONbits.IRCF1 = 0;**

**OSCCONbits.IRCF0 = 0;**

**break;**

**case 5: //OSCILADOR DE 2MHz**

**OSCCONbits.IRCF2 = 1;**

**OSCCONbits.IRCF1 = 0;**

**OSCCONbits.IRCF0 = 1;**

**break;**

**case 6: //OSCILADOR DE 4MHz**

**OSCCONbits.IRCF2 = 1;**

**OSCCONbits.IRCF1 = 1;**

**OSCCONbits.IRCF0 = 0;**

**break;**

**case 7: //OSCILADOR DE 8MHz**

**OSCCONbits.IRCF2 = 1;**

**OSCCONbits.IRCF1 = 1;**

**OSCCONbits.IRCF0 = 1;**

**break;**

**default: //OSCILADOR DE 4MHz**

**OSCCONbits.IRCF2 = 1;**

**OSCCONbits.IRCF1 = 1;**

**OSCCONbits.IRCF0 = 0;**

**break;**

**}**

**OSCCONbits.SCS = 1; //SE VA A USAR EL OSCILADOR INTERNO**

**}**

**/\***

**\* File : spi.c**

**\* Author : Ligo George**

**\* Company : electroSome**

**\* Project : SPI Library for MPLAB XC8**

**\* Microcontroller : PIC 16F877A**

**\* Created on April 15, 2017, 5:59 PM**

**\*/**

**#include <pic16f887.h>**

**#include <xc.h>**

**#include "SSP2.h"**

**void spiInit(Spi\_Type sType, Spi\_Data\_Sample sDataSample, Spi\_Clock\_Idle sClockIdle, Spi\_Transmit\_Edge sTransmitEdge)**

**{**

**TRISC5 = 0;**

**if(sType & 0b00000100) //If Slave Mode**

**{**

**SSPSTAT = sTransmitEdge;**

**TRISC3 = 1;**

**}**

**else //If Master Mode**

**{**

**SSPSTAT = sDataSample | sTransmitEdge;**

**TRISC3 = 0;**

**}**

**SSPCON = sType | sClockIdle;**

**}**

**static void spiReceiveWait()**

**{**

**while ( !SSPSTATbits.BF ); // Wait for Data Receive complete**

**}**

**void spiWrite(char dat) //Write data to SPI bus**

**{**

**SSPBUF = dat;**

**}**

**unsigned spiDataReady() //Check whether the data is ready to read**

**{**

**if(SSPSTATbits.BF){**

**return 1;**

**}**

**else{**

**return 0;**

**}**

**}**

**char spiRead() //REad the received data**

**{**

**spiReceiveWait(); // wait until the all bits receive**

**return(SSPBUF); // read the received data from the buffer**

**}**

**Código slave 2:**

**#include <xc.h>**

**#include <stdint.h>**

**#include <pic16f887.h>**

**#include "SSP2.h"**

**#include "osc2.h"**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**// Palabra de configuración**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**// CONFIG1**

**#pragma config FOSC = INTRC\_NOCLKOUT // Oscillator Selection bits (XT oscillator: Crystal/resonator on RA6/OSC2/CLKOUT and RA7/OSC1/CLKIN)**

**#pragma config WDTE = OFF // Watchdog Timer Enable bit (WDT disabled and can be enabled by SWDTEN bit of the WDTCON register)**

**#pragma config PWRTE = OFF // Power-up Timer Enable bit (PWRT disabled)**

**#pragma config MCLRE = OFF // RE3/MCLR pin function select bit (RE3/MCLR pin function is digital input, MCLR internally tied to VDD)**

**#pragma config CP = OFF // Code Protection bit (Program memory code protection is disabled)**

**#pragma config CPD = OFF // Data Code Protection bit (Data memory code protection is disabled)**

**#pragma config BOREN = OFF // Brown Out Reset Selection bits (BOR disabled)**

**#pragma config IESO = OFF // Internal External Switchover bit (Internal/External Switchover mode is disabled)**

**#pragma config FCMEN = OFF // Fail-Safe Clock Monitor Enabled bit (Fail-Safe Clock Monitor is disabled)**

**#pragma config LVP = OFF // Low Voltage Programming Enable bit (RB3 pin has digital I/O, HV on MCLR must be used for programming)**

**// CONFIG2**

**#pragma config BOR4V = BOR40V // Brown-out Reset Selection bit (Brown-out Reset set to 4.0V)**

**#pragma config WRT = OFF // Flash Program Memory Self Write Enable bits (Write protection off)**

**#define \_XTAL\_FREQ 4000000 //SE CONFIGURA EL OSCILADOR EXTERNO**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**//Variables**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**uint8\_t B1 = 0;**

**uint8\_t B2 = 0;**

**uint8\_t c1 = 0;**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**// Prototipos de funciones**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**void Setup(void);**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**//Interrupción**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**void \_\_interrupt() ISR(void) {**

**c1 = PORTD;**

**if (RBIF == 1) { //SE REVISA LA BANDERA DE INTERRUPCION DEL PUERTO B**

**if (PORTBbits.RB0 == 0) { //ANTIREBOTE, SI NO SE PRESIONA EL BOTON**

**B1 = 1; // SE ENCIENDE LA BANDERA DEL BOTON DE INCREMENTO**

**} else {**

**if (B1 == 1 && PORTBbits.RB0 == 1) { //SE PRESIONA EL BOTON**

**B1 = 0; //SE APAGA LA BANDERA DE BOTON DE INCREMENTO**

**PORTD++; // SE INCREMENTA EL PUERTOD**

**}**

**}**

**if (PORTBbits.RB1 == 0) { //ANTIREBOTE, SI NO SE PRESIONA EL BOTON**

**B2 = 1; // SE ENCIENDE LA BANDERA DEL BOTON DE DECREMENTO**

**} else {**

**if (B2 == 1 && PORTBbits.RB1 == 1) { //SE PRESIONA EL BOTON**

**B2 = 0; //SE APAGA LA BANDERA DE BOTON DE DECREMENTO**

**PORTD--; // SE DECREMENTA UN EL PUERTOD**

**}**

**}**

**INTCONbits.RBIF = 0; //SE APAGA LA BANDERA DE INTERRUPION DEL PUERTO B**

**}**

**if (PIR1bits.SSPIF == 1) {**

**spiWrite(c1);**

**PIR1bits.SSPIF = 0;**

**}**

**}**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**//Ciclo pincipal**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**void main(void) {**

**Setup();**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**// Loop principal**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**while (1) {**

**}**

**}**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**//Configuracion**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**void Setup(void) {**

**initOscs2(6);**

**spiInit(SPI\_SLAVE\_SS\_EN, SPI\_DATA\_SAMPLE\_MIDDLE, SPI\_CLOCK\_IDLE\_LOW, SPI\_IDLE\_2\_ACTIVE);**

**TRISD = 0;**

**TRISE = 0; //PUERTO E SALIDAS**

**ANSEL = 0; // ENTRADAS DIGITALES Y BIT 0 ANALÓGICA**

**ANSELH = 0;**

**PORTA = 0; //PUERTO A EN 0**

**PORTB = 0; //PUERTO B EN 0**

**PORTC = 0; //PUERTO C EN 0**

**PORTD = 0; //PUERTO D EN 0**

**PORTE = 0; //PUERTO E EN 0**

**//PINES RA0 Y RA2 COMO ENTRADAS, LOS DEMAS COMO SALIDAS**

**TRISC = 0b00001000; //PUERTO C SALIDAS**

**TRISA = 0b00100000; //PUERTO A SALIDAS**

**TRISB = 0b00000011; //PUERTO B**

**OPTION\_REG = 0b00000111; //SE ENCIENDEN LAS PULLUPS DEL PUERTO B**

**INTCONbits.GIE = 1; //SE HABILITAN LAS INTERRUPCIONES GLOBALES**

**INTCONbits.PEIE = 1; //SE HABILITAN LAS INTERRUPCIONES PERIFERICAS**

**INTCONbits.RBIE = 1; //SE HABILITA LA INTERRUPCION DEL PUERTO B**

**INTCONbits.RBIF = 0; //SE LIMPIA LA BANDERA DEL INTERRUPCION DEL PUERTO B**

**IOCB = 3; //SE HABILITA EL INTERRUPT ON CHANGE**

**WPUB = 0b0000011;**

**PIR1bits.SSPIF = 0;**

**PIE1bits.SSPIE = 1;**

**}**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**// Subrutinas**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Librerías slave 3:**

**Headers:**

**/\***

**\* File:**

**\* Author:**

**\* Comments:**

**\* Revision history:**

**\*/**

**// This is a guard condition so that contents of this file are not included**

**// more than once.**

**#ifndef ADCS3\_H**

**#define ADCS3\_H**

**#include <xc.h> // include processor files - each processor file is guarded.**

**#include <stdint.h>**

**void configADC3(uint8\_t fosc, uint8\_t chan);**

**#endif /\* ADCS3\_H\*/**

**// This is a guard condition so that contents of this file are not included**

**// more than once.**

**#ifndef OSCS3\_H**

**#define OSCS3\_H**

**#include <xc.h> // include processor files - each processor file is guarded.**

**#include <stdint.h>**

**void initOscs3(uint8\_t IRCF);**

**#endif /\* OSCS3\_H \*/**

**//\***

**// \* File : spi.h**

**// \* Author : Ligo George**

**// \* Company : electroSome**

**// \* Project : SPI Library for MPLAB XC8**

**// \* Microcontroller : PIC 16F877A**

**// \* Created on April 15, 2017, 5:59 PM**

**// \*/**

**// This is a guard condition so that contents of this file are not included**

**// more than once.**

**#ifndef \_\_SSP1\_H**

**#define \_\_SSP1\_H**

**#include <xc.h> // include processor files - each processor file is guarded.**

**#include <pic16f887.h>**

**typedef enum**

**{**

**SPI\_MASTER\_OSC\_DIV4 = 0b00100000,**

**SPI\_MASTER\_OSC\_DIV16 = 0b00100001,**

**SPI\_MASTER\_OSC\_DIV64 = 0b00100010,**

**SPI\_MASTER\_TMR2 = 0b00100011,**

**SPI\_SLAVE\_SS\_EN = 0b00100100,**

**SPI\_SLAVE\_SS\_DIS = 0b00100101**

**}Spi\_Type;**

**typedef enum**

**{**

**SPI\_DATA\_SAMPLE\_MIDDLE = 0b00000000,**

**SPI\_DATA\_SAMPLE\_END = 0b10000000**

**}Spi\_Data\_Sample;**

**typedef enum**

**{**

**SPI\_CLOCK\_IDLE\_HIGH = 0b00010000,**

**SPI\_CLOCK\_IDLE\_LOW = 0b00000000**

**}Spi\_Clock\_Idle;**

**typedef enum**

**{**

**SPI\_IDLE\_2\_ACTIVE = 0b00000000,**

**SPI\_ACTIVE\_2\_IDLE = 0b01000000**

**}Spi\_Transmit\_Edge;**

**void spiInit(Spi\_Type, Spi\_Data\_Sample, Spi\_Clock\_Idle, Spi\_Transmit\_Edge);**

**void spiWrite(char);**

**unsigned spiDataReady();**

**char spiRead();**

**#endif /\* SSP1\_H \*/**

**C:**

**#include <pic16f887.h>**

**#include <xc.h>**

**#include "adcs3.h"**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**// CONFIGURACION DEL ADC**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**void configADC3(uint8\_t fosc, uint8\_t chan) {**

**switch (fosc) {**

**case 0:**

**ADCON0bits.ADCS = 0b00;**

**break;**

**case 1:**

**ADCON0bits.ADCS = 0b01;**

**break;**

**case 2:**

**ADCON0bits.ADCS = 0b10;**

**break;**

**case 3:**

**ADCON0bits.ADCS = 0b11;**

**break;**

**default:**

**ADCON0bits.ADCS = 0b00;**

**break;**

**}**

**switch (chan) {**

**case 0:**

**ADCON0bits.CHS = 0b0000;**

**break;**

**case 1:**

**ADCON0bits.CHS = 0b0001;**

**break;**

**case 2:**

**ADCON0bits.CHS = 0b0010;**

**break;**

**case 3:**

**ADCON0bits.CHS = 0b0011;**

**break;**

**case 4:**

**ADCON0bits.CHS = 0b0100;**

**break;**

**case 5:**

**ADCON0bits.CHS = 0b0101;**

**break;**

**case 6:**

**ADCON0bits.CHS = 0b0110;**

**break;**

**case 7:**

**ADCON0bits.CHS = 0b0111;**

**break;**

**case 8:**

**ADCON0bits.CHS = 0b1000;**

**break;**

**case 9:**

**ADCON0bits.CHS = 0b1001;**

**break;**

**case 10:**

**ADCON0bits.CHS = 0b1010;**

**break;**

**case 11:**

**ADCON0bits.CHS = 0b1011;**

**break;**

**case 12:**

**ADCON0bits.CHS = 0b1100;**

**break;**

**case 13:**

**ADCON0bits.CHS = 0b1101;**

**break;**

**case 14:**

**ADCON0bits.CHS = 0b1110;**

**break;**

**case 15:**

**ADCON0bits.CHS = 0b1111;**

**break;**

**default:**

**ADCON0bits.CHS = 0b0000;**

**break;**

**}**

**ADCON0bits.GO\_nDONE = 1;**

**ADCON0bits.ADON = 1;**

**ADCON1 = 0;**

**ADCON1bits.VCFG0 = 1;**

**ADCON1bits.VCFG1 = 1;**

**}**

**#include <pic16f887.h>**

**#include <xc.h>**

**#include "oscs3.h"**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**//Inicialización del oscilador interno pg. 62**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**void initOscs3(uint8\_t IRCF){**

**switch (IRCF){**

**case 0: //OSCILADOR DE 31 kHz**

**OSCCONbits.IRCF2 = 0;**

**OSCCONbits.IRCF1 = 0;**

**OSCCONbits.IRCF0 = 0;**

**break;**

**case 1: //OSCILADOR DE 125 kHz**

**OSCCONbits.IRCF2 = 0;**

**OSCCONbits.IRCF1 = 0;**

**OSCCONbits.IRCF0 = 1;**

**break;**

**case 2: //OSCILADOR DE 250 kHz**

**OSCCONbits.IRCF2 = 0;**

**OSCCONbits.IRCF1 = 1;**

**OSCCONbits.IRCF0 = 0;**

**break;**

**case 3: //OSCILADOR DE 500kHz**

**OSCCONbits.IRCF2 = 0;**

**OSCCONbits.IRCF1 = 1;**

**OSCCONbits.IRCF0 = 1;**

**break;**

**case 4: //OSCILADOR DE 1MHz**

**OSCCONbits.IRCF2 = 1;**

**OSCCONbits.IRCF1 = 0;**

**OSCCONbits.IRCF0 = 0;**

**break;**

**case 5: //OSCILADOR DE 2MHz**

**OSCCONbits.IRCF2 = 1;**

**OSCCONbits.IRCF1 = 0;**

**OSCCONbits.IRCF0 = 1;**

**break;**

**case 6: //OSCILADOR DE 4MHz**

**OSCCONbits.IRCF2 = 1;**

**OSCCONbits.IRCF1 = 1;**

**OSCCONbits.IRCF0 = 0;**

**break;**

**case 7: //OSCILADOR DE 8MHz**

**OSCCONbits.IRCF2 = 1;**

**OSCCONbits.IRCF1 = 1;**

**OSCCONbits.IRCF0 = 1;**

**break;**

**default: //OSCILADOR DE 4MHz**

**OSCCONbits.IRCF2 = 1;**

**OSCCONbits.IRCF1 = 1;**

**OSCCONbits.IRCF0 = 0;**

**break;**

**}**

**OSCCONbits.SCS = 1; //SE VA A USAR EL OSCILADOR INTERNO**

**}**

**/\***

**\* File : spi.c**

**\* Author : Ligo George**

**\* Company : electroSome**

**\* Project : SPI Library for MPLAB XC8**

**\* Microcontroller : PIC 16F877A**

**\* Created on April 15, 2017, 5:59 PM**

**\*/**

**#include <pic16f887.h>**

**#include <xc.h>**

**#include "SSP3.h"**

**void spiInit(Spi\_Type sType, Spi\_Data\_Sample sDataSample, Spi\_Clock\_Idle sClockIdle, Spi\_Transmit\_Edge sTransmitEdge)**

**{**

**TRISC5 = 0;**

**if(sType & 0b00000100) //If Slave Mode**

**{**

**SSPSTAT = sTransmitEdge;**

**TRISC3 = 1;**

**}**

**else //If Master Mode**

**{**

**SSPSTAT = sDataSample | sTransmitEdge;**

**TRISC3 = 0;**

**}**

**SSPCON = sType | sClockIdle;**

**}**

**static void spiReceiveWait()**

**{**

**while ( !SSPSTATbits.BF ); // Wait for Data Receive complete**

**}**

**void spiWrite(char dat) //Write data to SPI bus**

**{**

**SSPBUF = dat;**

**}**

**unsigned spiDataReady() //Check whether the data is ready to read**

**{**

**if(SSPSTATbits.BF){**

**return 1;**

**}**

**else{**

**return 0;**

**}**

**}**

**char spiRead() //REad the received data**

**{**

**spiReceiveWait(); // wait until the all bits receive**

**return(SSPBUF); // read the received data from the buffer**

**}**

**Código slave 3:**

**#include <xc.h>**

**#include <stdint.h>**

**#include <pic16f887.h>**

**#include "oscs3.h"**

**#include "adcs3.h"**

**#include "SSP3.h"**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**// Palabra de configuración**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**// CONFIG1**

**#pragma config FOSC = INTRC\_NOCLKOUT // Oscillator Selection bits (XT oscillator: Crystal/resonator on RA6/OSC2/CLKOUT and RA7/OSC1/CLKIN)**

**#pragma config WDTE = OFF // Watchdog Timer Enable bit (WDT disabled and can be enabled by SWDTEN bit of the WDTCON register)**

**#pragma config PWRTE = OFF // Power-up Timer Enable bit (PWRT disabled)**

**#pragma config MCLRE = OFF // RE3/MCLR pin function select bit (RE3/MCLR pin function is digital input, MCLR internally tied to VDD)**

**#pragma config CP = OFF // Code Protection bit (Program memory code protection is disabled)**

**#pragma config CPD = OFF // Data Code Protection bit (Data memory code protection is disabled)**

**#pragma config BOREN = OFF // Brown Out Reset Selection bits (BOR disabled)**

**#pragma config IESO = OFF // Internal External Switchover bit (Internal/External Switchover mode is disabled)**

**#pragma config FCMEN = OFF // Fail-Safe Clock Monitor Enabled bit (Fail-Safe Clock Monitor is disabled)**

**#pragma config LVP = OFF // Low Voltage Programming Enable bit (RB3 pin has digital I/O, HV on MCLR must be used for programming)**

**// CONFIG2**

**#pragma config BOR4V = BOR40V // Brown-out Reset Selection bit (Brown-out Reset set to 4.0V)**

**#pragma config WRT = OFF // Flash Program Memory Self Write Enable bits (Write protection off)**

**#define \_XTAL\_FREQ 4000000 //SE CONFIGURA EL OSCILADOR EXTERNO**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**//Variables**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**uint8\_t term = 0;**

**uint8\_t CONTERM = 0;**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**// Prototipos de funciones**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**void Setup(void);**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**//Interrupción**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**void \_\_interrupt() ISR(void) {**

**if (INTCONbits.T0IF == 1) {**

**TMR0 = 236;**

**CONTERM++;**

**INTCONbits.T0IF = 0;**

**}**

**if (PIR1bits.SSPIF == 1) {**

**spiWrite(term);**

**PIR1bits.SSPIF = 0;**

**}**

**if (PIR1bits.ADIF == 1) {**

**term = ADRESH;**

**PIR1bits.ADIF = 0;**

**}**

**if (term < 100) {**

**PORTEbits.RE2 = 1;**

**PORTEbits.RE1 = 0;**

**PORTEbits.RE0 = 0;**

**}**

**if (term > 100 && term < 113) {**

**PORTEbits.RE2 = 0;**

**PORTEbits.RE1 = 1;**

**PORTEbits.RE0 = 0;**

**}**

**if (term > 113) {**

**PORTEbits.RE2 = 0;**

**PORTEbits.RE1 = 0;**

**PORTEbits.RE0 = 1;**

**}**

**}**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**//Ciclo pincipal**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**void main(void) {**

**Setup();**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**// Loop principal**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**while (1) {**

**if (CONTERM > 20) {**

**ADCON0bits.GO\_nDONE = 1;**

**CONTERM = 0;**

**}**

**}**

**}**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**//Configuracion**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**void Setup(void) {**

**configADC3(1, 10); //SE LLAMA LA CONFIG DEL ADC**

**initOscs3(6);**

**spiInit(SPI\_SLAVE\_SS\_EN, SPI\_DATA\_SAMPLE\_MIDDLE, SPI\_CLOCK\_IDLE\_LOW, SPI\_IDLE\_2\_ACTIVE);**

**ANSEL = 0; // ENTRADAS DIGITALES Y BIT 0 ANALÓGICA**

**ANSELH = 0b00000010;**

**PORTA = 0; //PUERTO A EN 0**

**PORTB = 0; //PUERTO B EN 0**

**PORTC = 0; //PUERTO C EN 0**

**PORTD = 0; //PUERTO D EN 0**

**PORTE = 0; //PUERTO E EN 0**

**//PINES RA0 Y RA2 COMO ENTRADAS, LOS DEMAS COMO SALIDAS**

**TRISC = 0b00001000; //PUERTO C SALIDAS**

**TRISA = 0b00100000; //PUERTO A SALIDAS**

**TRISB = 0b00000010; //PUERTO B**

**TRISD = 0;**

**TRISE = 0;**

**OPTION\_REG = 0b10000111; //SE APAGAN LAS PULLUPS DEL PUERTO B**

**INTCONbits.GIE = 1; //SE HABILITAN LAS INTERRUPCIONES GLOBALES**

**INTCONbits.PEIE = 1; //SE HABILITAN LAS INTERRUPCIONES PERIFERICAS**

**PIE1bits.ADIE = 1; //SE HABILITA LA INTERRUPCION DEL ADC**

**PIR1bits.ADIF = 0; //SE LIMPIOA LA BANDERA DE INTERRUPCION DEL ADC**

**PIR1bits.SSPIF = 0;**

**PIE1bits.SSPIE = 1;**

**INTCONbits.T0IE = 1;**

**INTCONbits.T0IF = 0;**

**}**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**// Subrutinas**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***