```
// ***************************************************************
// ***************************************************************
// **      PROGGETTO          di   ALBERTO DALLA VAL            **
// ***************************************************************
// ***************************************************************
// **                                Slave                        **
// ***************************************************************
// ***************************************************************

        #include <p18f2431.h>
        #include <ctype.h>
        #include <delay.h>
        #define  LCD_DEFAULT
        #include <LCD_44780_A.h>

        #include <string.h>
        #include <i2c.h>
        #include <portb.h>

        //#include <stdlib.h>
        //#include <pwm.h>
        //#include <adc.h>
        //#include <timers.h>



        #pragma config OSC = HS
        #pragma config WINEN =OFF
        #pragma config WDTEN = OFF
        #pragma config PWRTEN = ON
        #pragma config BOREN = ON
        #pragma config WDPS = 128
        #pragma config LVP = OFF
        #pragma config DEBUG = OFF


//OSC = HS    Impostato per lavorare ad alta frequenza
//WDT = OFF           Disabilito il watchdog timer
//LVP = OFF   Disabilito programmazione LVP


//*************************************************************
//                                      Inizializza
//*************************************************************


        void Inizializza (void) {


// Imposto PORTA tutti ingressi
                LATA = 0x00;
                TRISA = 0b11111100;

// Imposto PORTB 0-1-2-3 come uscite e 4-5-6-7 come ingressi
```

```
                LATB = 0x00;
                TRISB = 0b11110000;

// Imposto PORTC tutti ingressi, RC1 come output
                LATC = 0x00;
                TRISC = 0b00111100;
                PORTC = 0x00;
// Disattivo tutti gli interrupt

                INTCON = 0;


// Inizializzo la I2C

                SSPSTAT = 0b00000000;
                SSPADD        = 0x10;
                SSPCON  = 0b00111110;
                SSPBUF  = 0x00;

// Abilitazione interruzioni del modulo I2C SSPIE 1-0 come alta-bassa priorita' SSPIP 1-0
                PIE1bits.SSPIE = 1;
                IPR1bits.SSPIP = 1;


//******************************************************
// Abilito le interruzioni
//******************************************************

// Abilito modalita' interruzione a due livelli alta e bassa
                RCONbits.IPEN = 1;

// Abilito gli interrupt ad alta priorita'
                INTCONbits.GIEH = 1;

// Abilito gli interrupt a bassa priorita'
                INTCONbits.GIEL = 1 ;

// Azzero interrupt I2C
                PIR1bits.SSPIF = 0;
                SSPCON  = SSPCON & 0b00111110;

}
//************************************************************
// Dichiarazione Prototipi di funzione
//************************************************************


// Prototipo di funzione per bassa priorita'
        void Low_Int_Event (void);

// Prototipo di funzione per alta priorita'
        void High_Int_Event (void);



// Impostazione e Gestione priorita' alta
```

```
	#pragma code high_vector = 0x08

	void high_interrupt (void) {

		// Salto per la gestione dell'interrupt ad alta priorita'
	  _asm GOTO High_Int_Event _endasm
	}

	#pragma code

	#pragma interrupt High_Int_Event



//************************************************************
// Funzione per la gestione dell'interruzione ad alta priorita'
//************************************************************

	void High_Int_Event (void) {

	char I2C_dato = 0;

	if (PIR1bits.SSPIF == 1) {

   {if(!SSPSTATbits.D_A && !SSPSTATbits.R_W && SSPSTATbits.BF && SSPSTATbits.S) { //
An address from master: write to slave
	SSPBUF = SSPBUF;	// Svuoto buffer e cancello il registro BF
		if(SSPCONbits.SSPOV){		// Controllo overflow
		   //...		// Gestione overflow
		   SSPCONbits.SSPOV = 0;
		   }
		 }


	 else if(SSPSTATbits.D_A && !SSPSTATbits.R_W && SSPSTATbits.BF){	//
Ricezione dati dal Master
	    I2C_dato = SSPBUF;


		if(SSPCONbits.SSPOV){	// Controllo overflow
		   //...			// Gestione overflow
		   SSPCONbits.SSPOV = 0;
		   }
		 }

	 else if(!SSPSTATbits.D_A && SSPSTATbits.R_W && !SSPSTATbits.BF) {		// An
address from master: read from slave


	    SSPBUF = 10;			// Scrivo un dato per il Master
	SSPCONbits.CKP = 1;			// Rilascia il clock (dato pronto per essere letto dal
Master)

		}
```

```
        else if(SSPSTATbits.D_A && SSPSTATbits.R_W && !SSPSTATbits.BF &&
SSPSTATbits.S)  { // Read a byte from slave (next byte)
                SSPBUF = 29;                            // Scrivo un dato per il Master
        SSPCONbits.CKP = 1;       // Rilascia il clock (dato pronto per essere letto dal Master)
                }

        else if(SSPSTATbits.D_A && !SSPSTATbits.R_W && !SSPSTATbits.BF)         // NAK
from master (complete transmission has occurred)
           {
                }

        }
                PIR1bits.SSPIF = 0;
                }
        }

//***********************************************************
// Impostazione e Gestione priorita' bassa
//***********************************************************

        #pragma code low_vector = 0x18

        void low_interrupt (void) {

                // Salto per la gestione dell'interrupt a bassa priorita'
          _asm GOTO Low_Int_Event _endasm
        }

        #pragma code

        #pragma interruptlow Low_Int_Event


// Funzione per la gestione dell'interruzione ad alta priorita'
        void Low_Int_Event (void) {



        }

//***********************************************************
//***********************************************************
        void main (void){


                Inizializza ();

                while (1) {
                }


        }
```