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

#include <p18f4550.h>
#define LCD_DEFAULT
#include <LCD_44780.h>

#include <ctype.h>
#include <delay.h>
#include <portb.h>
#include <math.h>
#include <string.h>
#include <i2c.h>
#include <i2cEEPROM.h>

#pragma config FOSC = HS
#pragma config WDT = OFF
#pragma config LVP = OFF
#pragma config PBADEN = OFF

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

char I2C_dato = 0;


//*************************************************************
//  Funzione di Inizializzazione
//*************************************************************

        void Inizializza (void) {

// Imposto PORTA tutti ingressi

        LATA = 0x00;
        TRISA = 0xFF;

// Imposto PORTB tutti ingressi

        LATB = 0x00;
        TRISB = 0xFF;

// Imposto PORTC tutti ingressi

        LATC = 0x00;
        TRISC = 0b11111101;
```

```
// Imposto PORTD tutte uscite (display)

        LATD = 0x00;
        TRISD = 0x00;

// Imposto PORTE tutti ingressi

        LATE = 0x00;
        TRISE = 0xFF;

        setQuartz (20);


//************************************************************
// Abilito i pulsanti per le interruzioni ad alta priorita'
//************************************************************

// Abilita i resistori di pull-up sulla PORTB
        EnablePullups();

// Abilito le interruzioni su PORTB (RBIE) come alta priorita' (RBIP)
        INTCONbits.RBIE = 1;
        INTCON2bits.RBIP = 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 ;


// Inizializzo il display LCD con quarzo a 20MHz

        OpenLCD (20);

        ClearLCD ();

// Inizializzo l' I2C a 400Khz con quarzo a 20MHz SSPAD = 12

        OpenI2C (MASTER, SLEW_ON);
        SSPADD = 12;
}

//************************************************************
//************************************************************
// 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) {


// Controllo che l'interrupt sia stato generato da PORTB
        if (INTCONbits.RBIF == 1 ) {


// Controllo la pressione di RB4
        if (PORTBbits.RB4 == 0) {
                delay_ms(100);
                if (PORTBbits.RB4 == 0) {
                IdleI2C ();
                StartI2C ();
                IdleI2C ();
                WriteI2C(0x11); // indirizzo D0 in Lettura
                AckI2C ();
           I2C_dato = ReadI2C(); //
                IdleI2C ();
                StopI2C ();
                }
        }
// Controllo la pressione di RB5
        if (PORTBbits.RB5 == 0) {
                delay_ms(100);
                if (PORTBbits.RB5 == 0) {
                IdleI2C ();
                StartI2C ();
                IdleI2C ();
                WriteI2C(0x10); // indirizzo D0 in scrittura
                IdleI2C ();
                WriteI2C(0b01010101);        // tipo
                IdleI2C ();
```

```
                StopI2C ();
                }

        }

// Controllo la pressione di RB6
        if (PORTBbits.RB6 == 0) {
                delay_ms(100);
                if (PORTBbits.RB6 == 0) {
                IdleI2C ();
                StartI2C ();
                IdleI2C ();
                WriteI2C(0x10); // indirizzo D0 in scrittura
                IdleI2C ();
                WriteI2C(0b10101010);        // tipo
                IdleI2C ();
                StopI2C ();
                }

        }

// Resetto il flag d'interrupt per permettere nuove interruzioni
                INTCONbits.RBIF = 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)
 {

}

//**************************************************************
// Inizio programma principale
//**************************************************************
void main (void){
        Inizializza ();
        // Ciclo infinito
                while (1)  {
                }
}
```