# LCD I2C  MC42005A6W-BNMLWI-V2

Il display ha 4 righe per 20 colonne, e viene interfacciato con il microcontrollore, tramite lo standard I2C.

Di seguito la piedinatura del display e lo schema di collegamento, estratti dal suo datasheet.

Il display va alimentato a 5V, ed il contrasto viene regolato mediante una tensione variabile sul piedino 3, applicata in questo caso con un potenziometro.

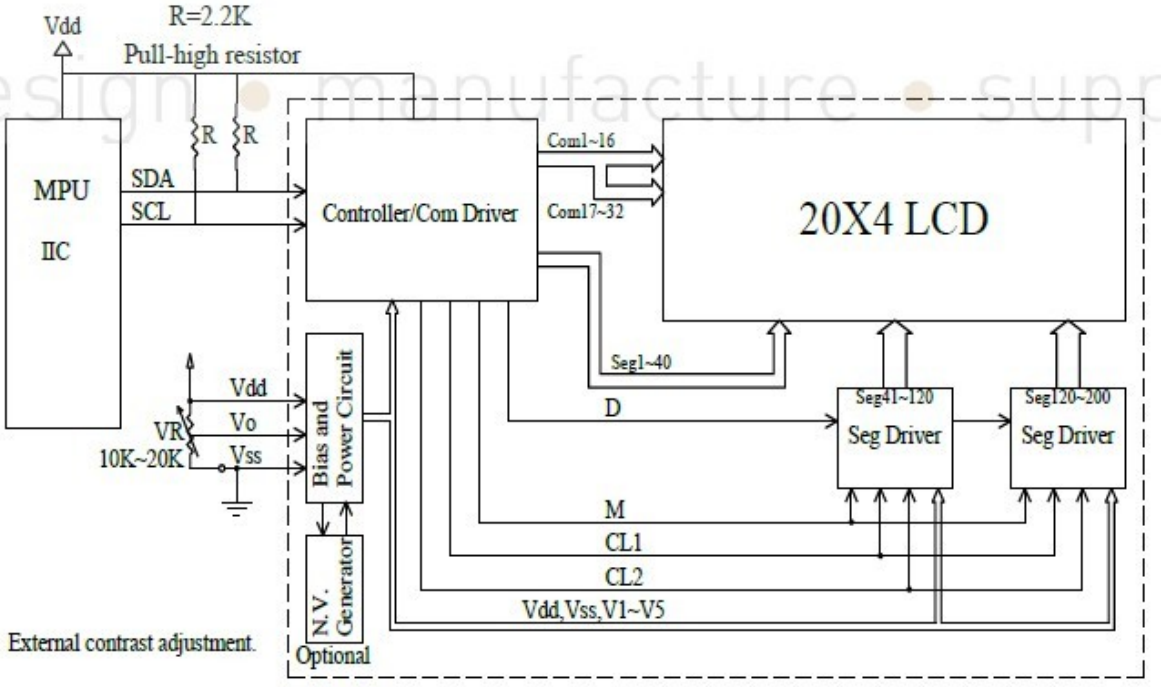
I pin 7 e 8, permettono di impostare l'indirizzo sul bus I2C.

Il pin 12 va collegato a GND per abilitare il display.

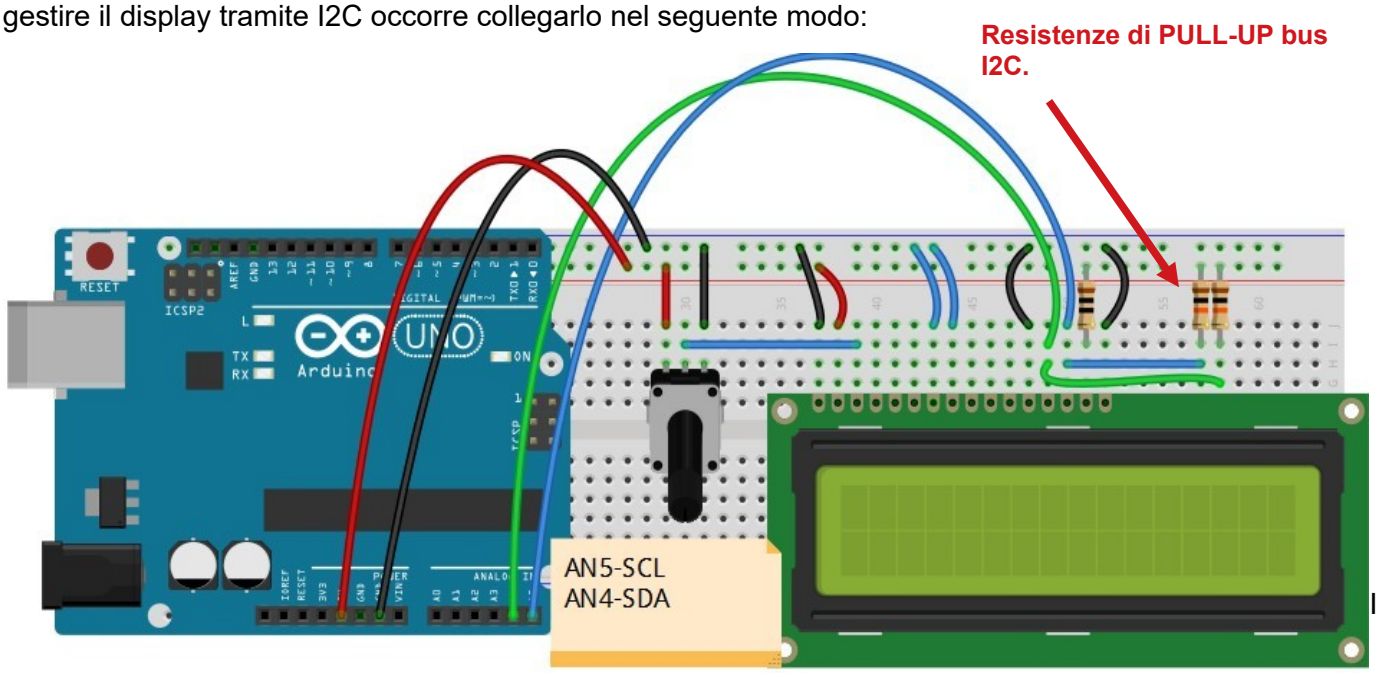
I pin 13 e 14, vengono utilizzati per il collegamento al BUS I2C.

I pin 15 e 16 invece vengono utilizzati per la retroilluminazione, è opportuno mettere una resistenza di 100 Ohm in serie per evitare rotture.

| Pin No. | Symbol | Level | Description |
|---|---|---|---|
| 1 | Vss | 0V | Ground |
| 2 | VDD | 5.0V | Supply Voltage for logic |
| 3 | VO | (Variable) | Operating voltage for LCD |
| 4 | NC | – | No connection |
| 5 | NC | – | No connection |
| 6 | NC | – | No connection |
| 7 | SA0 | H/L | In IIC interface ,DB1(SA1) and DB0(SA0) are used for Slave address, must be connect to VDD or VSS |
| 8 | SA1 | H/L | |
| 9 | NC | – | No connection |
| 10 | NC | – | No connection |
| 11 | NC | – | No connection |
| 12 | CSB | H/L | In IIC serial mode, used as chip selection input.<br>When CSB = "Low", selected<br>When CSB = "High", not selected.<br>( Low access enable ) |
| 13 | SDA | H/L | serial input data |
| 14 | SCL | H/L | serial clock input |
| 15 | A | – | Power supply for B/L + |
| 16 | K | – | Power supply for B/L - |



Per gestire il display tramite I2C occorre collegarlo nel seguente modo:

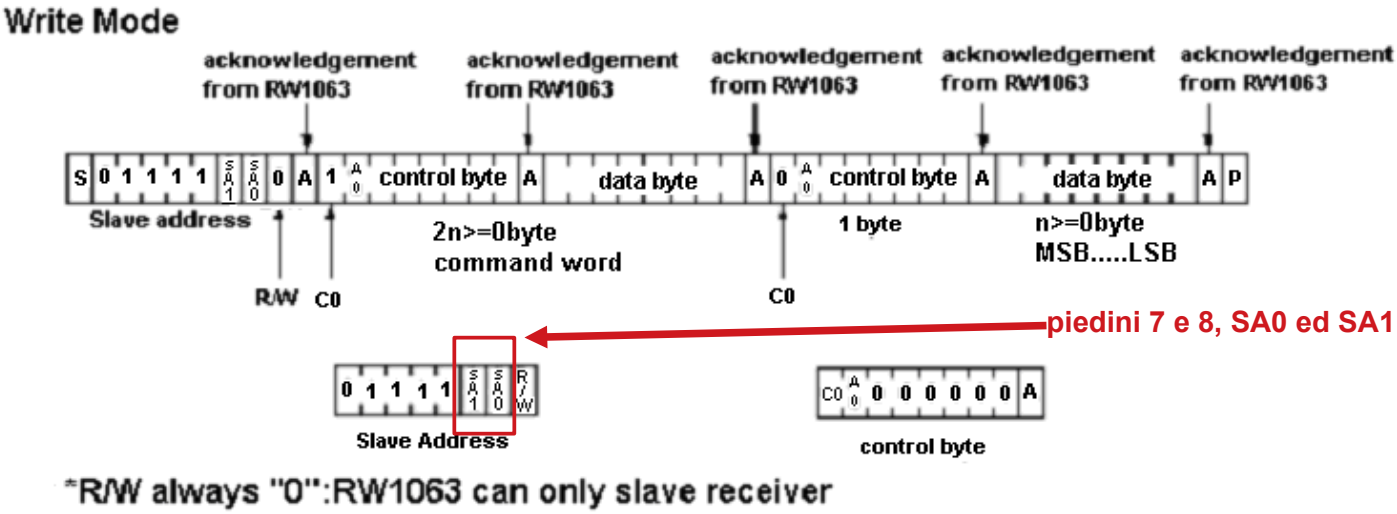



Il piedino 12 del display va collegato a GND, i piedini 7 e 8 SA0 ed SA1 servono per indirizzare il display se ci sono più display collegati sul bus I2C. Nel caso di un solo display vanno collegati a GND, l'indirizzo del display in questo caso è 0x78.

---

I comandi vanno inviati con la seguente sequenza:

1. I2Cstart
2. Invio indirizzo 0x78 (l'indirizzo dipende dallo stato dei pin 7-8)
3. Invio control_byte pari a 0x00 per comandi o pari a 0x40 per scrittura su schermo
4. Invio dato che può essere un comando o un carattere da visualizzare
5. I2Cstop



**Esempi:**

| | | | | |
|---|---|---|---|---|
| ***I2CStart*** | *Condizione di start* | | ***I2CStart*** | *Condizione di start* |
| ***I2Cwrite(0x78)*** | *Scrittura indirizzo LCD* | | ***I2Cwrite(0x78)*** | *Scrittura indirizzo LCD* |
| ***I2Cwrite(0x00)*** | *Control byte* | | ***I2Cwrite(0x40)*** | *Control byte* |
| ***I2Cwrite(0x01)*** | *Clear display* | | ***I2Cwrite(0x41)*** | *Codice ASCII carattere* |
| ***I2CStop*** | *Condizione di stop* | | ***I2CStop*** | *Condizione di stop* |

## ELENCO PRINCIPALI COMANDI

| COMANDO | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | Desrcizione | Durata |
|---|---|---|---|---|---|---|---|---|---|---|
| **Clear Display** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | Scrive 20H in tutta la DDRAM e setta successviamente l'indirizzo 00H (inizio display) | 0,7ms |
| **Return Home** | 0 | 0 | 0 | 0 | 0 | 0 | 1 | X | Setta l'indirizzo 00H della DDRAM(inizio display) il contenuto della DDRAM non viene modificato | 0,7ms |
| **Entry Mode Set** | 0 | 0 | 0 | 0 | 0 | 1 | I/D | S | Seleziona il movimento del cursore dopo la scrittura sulla DDRAM. I/D=1 incremento a dx | 18,5us |
| **Display ON/OFF** | 0 | 0 | 0 | 0 | 1 | D | C | B | D=1 display on D=0 display off<br>C=1 cursore on C=0 cursore off<br>B=1 lampeggio on B=0 lampeggio off | 18,5us |
| **Cursor or Display shift** | 0 | 0 | 0 | 1 | S/C | R/L | X | X | S/C=1 shift del display S/C=0 shift del cursore<br>R/L=1 shift a destra R/L=0 shift a sinistra | 18,5us |
| **Function Set** | 0 | 0 | 1 | DL | N | F | X | X | DL= 8-bit interface/ 4-bit interface<br>N = 2-line/1-line display<br>F= 5x8 Font Size / 5x11Font Size | 18,5us |
| **Set DDRAM Address** | | | | | | | | | | 18,5us |

## SEQUENZA INIZIALE

- ***I2CStart*** ***I2Cwrite(0x78)*** ***I2Cwrite(0x00)*** ***I2Cwrite(0x38)*** ***I2CStop*** *- impostazione 2/4 righe*
- ***I2CStart*** ***I2Cwrite(0x78)*** ***I2Cwrite(0x00)*** ***I2Cwrite(0x0F)*** ***I2CStop*** *- attivo display e cursore*
- ***I2CStart*** ***I2Cwrite(0x78)*** ***I2Cwrite(0x00)*** ***I2Cwrite(0x01)*** ***I2CStop*** *- clear display*

Dopo la sequenza iniziale, si può selezionare la posizione del cursore e scrivere il carattere desiderato.

Per posizionare il cursore sul display, occorre selezionare l'indirizzo della DDRAM interna, il control byte è 0x00 seguito dall'indirizzo della DDRAM che si può ricavare dalla seguente tabella.

**INDIRIZZO DDRAM IN ESADECIMALE - POSIZIONE SUL DISPLAY**

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ***80*** | ***81*** | ***82*** | ***83*** | ***84*** | ***85*** | ***86*** | ***87*** | ***88*** | ***89*** | ***8A*** | ***8B*** | ***8C*** | ***8D*** | ***8E*** | ***8F*** | ***90*** | ***91*** | ***92*** | ***93*** |
| ***C0*** | ***C1*** | ***C2*** | ***C3*** | ***C4*** | ***C5*** | ***C6*** | ***C7*** | ***C8*** | ***C9*** | ***CA*** | ***CB*** | ***CC*** | ***CD*** | ***CE*** | ***CF*** | ***D0*** | ***D1*** | ***D2*** | ***D3*** |
| ***94*** | ***95*** | ***96*** | ***97*** | ***98*** | ***99*** | ***9A*** | ***9B*** | ***9C*** | ***9D*** | ***9E*** | ***9F*** | ***A0*** | ***A1*** | ***A2*** | ***A3*** | ***A4*** | ***A5*** | ***A6*** | ***A7*** |
| ***D4*** | ***D5*** | ***D6*** | ***D7*** | ***D8*** | ***D9*** | ***DA*** | ***DB*** | ***DC*** | ***DD*** | ***DE*** | ***DF*** | ***E0*** | ***E1*** | ***E2*** | ***E3*** | ***E4*** | ***E5*** | ***E6*** | ***E7*** |

Dopo i comandi iniziali si può procedere alla visualizzazione del carattere posizionando il cursore e scrivendo il carattere in ASCII.

- ***I2CStart*** ***I2Cwrite(0x78)*** ***I2Cwrite(0x00)*** ***I2Cwrite(0xC0)*** ***I2CStop*** *- inizio seconda riga*
- ***I2CStart*** ***I2Cwrite(0x78)*** ***I2Cwrite(0x40)*** ***I2Cwrite(0x41)*** ***I2CStop*** *- scrivi 'A'*

## UTILIZZO CON ARDUINO

Utilizzando la piattaforma di ARDUINO, si può utilizzare la libreria **Wire.h** , con le sue funzioni.

In questa libreria troviamo le seguenti funzioni:

*La funzione* **Wire.begin()** *va lanciata una sola volta all'avvio*

*Con la funzione* **Wire.begintrasmission(indirizzo)** *si crea la condizione di start dell'I2C, e si invia sul bus l'indirizzo dello Slave.* **L'indirizzo va indicato senza considerare il bit R/W,** *perciò nel nostro caso l'indirizzo da scrivere non sarà 78 in esadecimale ma* **3C sempre in esadecimale**.

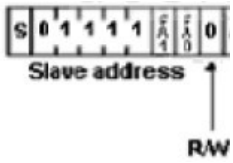
*Con la funzione* **Wire.endtrasmission()** *creiamo sul bus la condizione di Stop dell'I2C.*

*Con la funzione* **Wire.write()** *inviamo il dato o i dati, da scrivere dopo l'indirizzo.*

Functions

- begin()
- requestFrom()
- beginTransmission()
- endTransmission()
- write()
- available()
- read()
- onReceive()
- onRequest()

A differenza di altre librerie (come ad esempio quella utilizzata nell'ambiente per microcontrollori PIC MikroC) la funzione che invia l'indirizzo dello Slave, utilizza il valore a 7 bit, senza considerare il bit R/W.

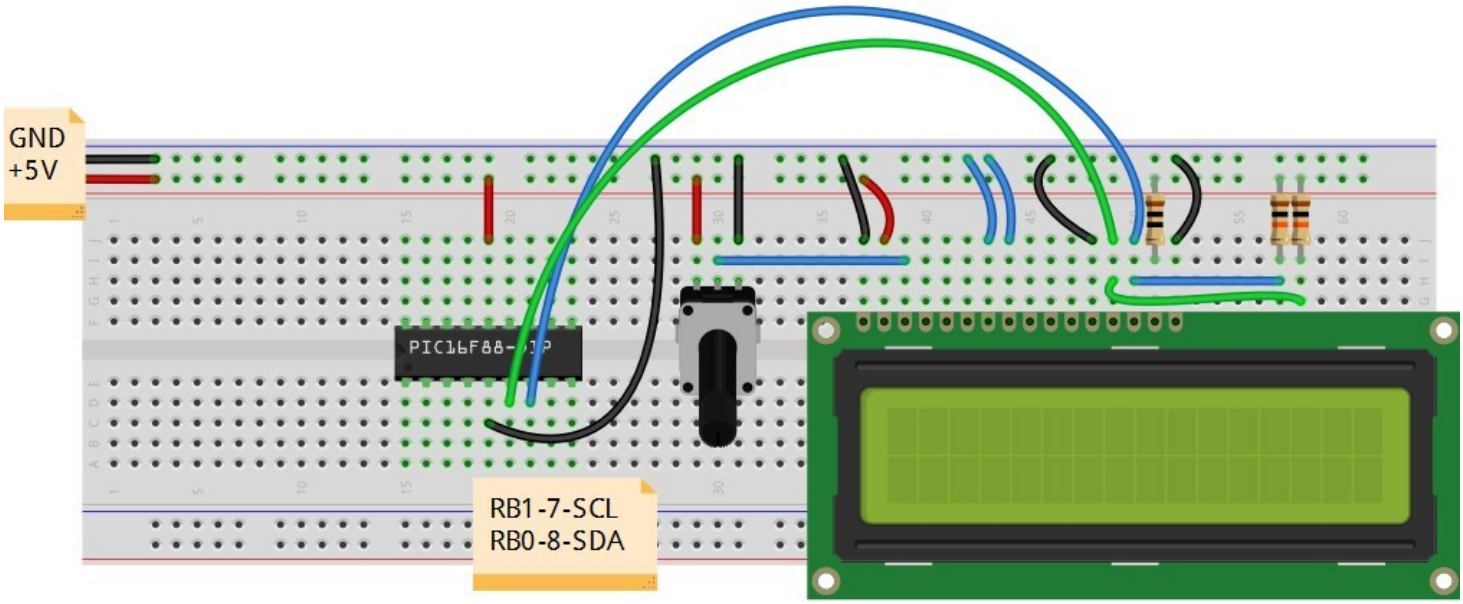



Perciò il valore da indicare sull'indirizzo con la funzione Wire.begintrasmission, non sarà 0111 1000=0x78 , ma **011 1100=0x3C.**

Il seguente programma di esempio scrive il carattere A all'inizio della seconda riga.

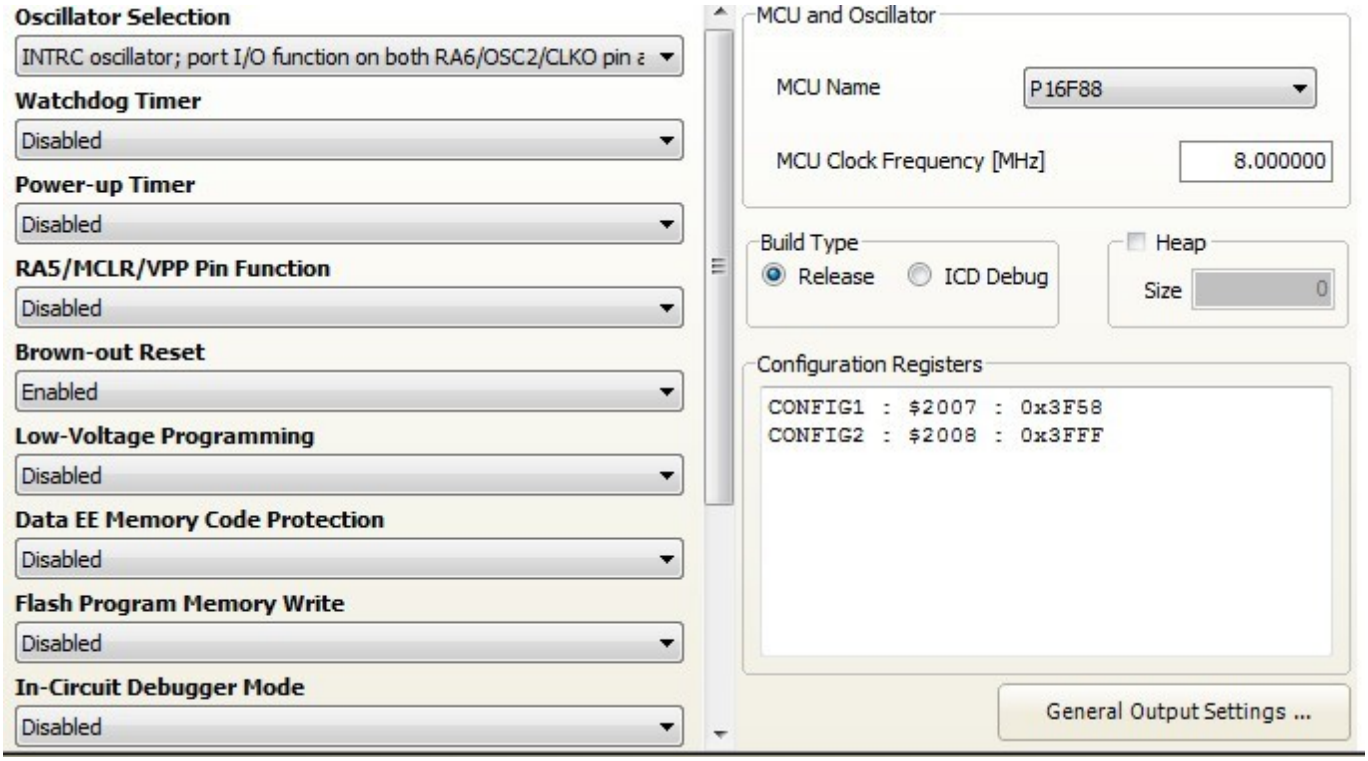
```
#include <Wire.h>

char address=0X3C;  //indirizzo display 0x78 in esadecimale shiftato di un bit

void setup() {
  Wire.begin();

  Wire.beginTransmission(address);//start ed indirizzo
  Wire.write(0);          // control byte
  Wire.write(56);         // dato=0x38 in esadecimale impostazione 4 righe
  Wire.endTransmission(); // stop
  delay(1);

  Wire.beginTransmission(address);//start ed indirizzo
  Wire.write(0);          // control byte
  Wire.write(15);         // dato=0x0F in esadecimale display e cursore attivo
  Wire.endTransmission(); // stop
  delay(1);

  Wire.beginTransmission(address);//start ed indirizzo
  Wire.write(0);          // control byte
  Wire.write(1);          // clear display
  Wire.endTransmission(); // stop
  delay(1);
}


void loop() {
  Wire.beginTransmission(address);//start ed indirizzo
  Wire.write(0);      // control byte 0
  Wire.write(192);     //dato=0xC0 in esadecimale cursore inizio seconda riga
  Wire.endTransmission();  // stop
  delayMicroseconds(20);

  Wire.beginTransmission(address);//start ed indirizzo
  Wire.write(64);           // control byte 0x40 in esadecimale
  Wire.write(65);           // scrivi il carattere inviato in ASCII
  Wire.endTransmission();  // stop
  delayMicroseconds(20);

  while(1){}  //ciclo infinito
}
```

**UTILIZZO CON MICROCONTROLLORE PIC16F88 E COMPILATORE MIKROC**




In questo caso verrà utilizzata la libreria I2C softwre, che consente di utilizzare il protocollo I2C su qualsiasi pin.

Di seguito le impostazioni del microcontrollore, clock interno e tutti i piedini disponibili.



```
sbit Soft_I2C_Scl           at RB1_bit;
sbit Soft_I2C_Sda           at RB0_bit;
sbit Soft_I2C_Scl_Direction at TRISB1_bit;
sbit Soft_I2C_Sda_Direction at TRISB0_bit;

//*************************************************************
//SELEZIONE DEL TIPO DI DISPLAY
//LCD 2 o 4 righe
void setup_display()
{
   Soft_I2C_Start();
   Soft_I2C_Write(0x78);
   Soft_I2C_Write(0x00);
   Soft_I2C_Write(0x38);
   Soft_I2C_Stop();
   delay_us(20);
}//setup_display
```

```
//*************************************************************
//CLEAR DEL DISPLAY
void clear_display()
{
   Soft_I2C_Start();
   Soft_I2C_Write(0x78);
   Soft_I2C_Write(0x00);
   Soft_I2C_Write(0x01);
   Soft_I2C_Stop();
   delay_ms(1);
}//clear_display

//*************************************************************
//RITORNO ALLA PRIMA POSIZIONE, HOME
void return_home()
{
   Soft_I2C_Start();
   Soft_I2C_Write(0x78);
   Soft_I2C_Write(0x00);
   Soft_I2C_Write(0x02);
   Soft_I2C_Stop();
   delay_ms(1);
}//return_home
```

……..continua……

```
//*************************************************************
//SHIFT DISPLAY O CURSORE
//type=1 shift display
//type=0 shift cursore
//dir=1 shift right
//dir=0 shift left
void shift(char type, char dir)
{
   char valore=0x10;

   if(type) valore=valore|0x08;
   else     valore=valore&0xF7;
   if(dir)  valore=valore|0x04;
   else     valore=valore&0xFB;
   Soft_I2C_Start();
   Soft_I2C_Write(0x78);
   Soft_I2C_Write(0x00);
   Soft_I2C_Write(valore);
   Soft_I2C_Stop();
   delay_us(20);
}//shift
//*************************************************************
//ACCENSIONE DISPLAY E SELEZIONE DEL TIPO DI CURSORE
//dusplay on/off, cursor on/off, blink on/off
void set_cursor_type(char display, char cursor, char blink)
{
   char valore=0x08;

   if(display) valore=valore|0x04;
   else        valore=valore&0x0B;
   if(cursor)  valore=valore|0x02;
   else        valore=valore&0x0D;
   if(blink)   valore=valore|0x01;
   else        valore=valore&0x0E;
   Soft_I2C_Start();
   Soft_I2C_Write(0x78);
   Soft_I2C_Write(0x00);
   Soft_I2C_Write(valore);
   Soft_I2C_Stop();
   delay_us(20);
}//set_cursor_type
```

……..continua……

```
//*************************************************************
//SELEZIONE TIPO DI INCREMENTO O DECREMENTO CURSORE
//direzione shift cursore 1=avanti 0=indietro
void set_cursor_dir(char direction)
{
   char valore;

   if(direction) valore=0x06;
   else          valore=0x04;
   Soft_I2C_Start();
   Soft_I2C_Write(0x78);
   Soft_I2C_Write(0x00);
   Soft_I2C_Write(valore);
   Soft_I2C_Stop();
   delay_us(20);
}//set_cursor_dir
```

```
//*************************************************************
//POSIZIONAMENTO CURSORE IN RIGHE E COLONNE
//DA 1-1 A 4-20
//posiziono il cursore
void set_cursor_pos(char linea, char colonna)
]{
  char valore;

  if(colonna) colonna--;
]  switch(linea){
    case 1: valore=128+colonna;  break;     //0x80 + colonna
    case 2: valore=192+colonna;  break;     //0xC0 + colonna
    case 3: valore=148+colonna;  break;     //0x94 + colonna
    case 4: valore=212+colonna;  break;     //0xD4 + colonna
  }//switch
  if((valore>=128)&&(valore<=231))
]   {
    Soft_I2C_Start();
    Soft_I2C_Write(0x78);
    Soft_I2C_Write(0x00);
    Soft_I2C_Write(valore);
    Soft_I2C_Stop();
    delay_us(20);
   }//if
}//set_cursor_pos
```

……..continua……

```
//*****************************************************************
//SCRITTURA DEL SINGOLO CARATTERE
void scrivi_char(char valore)
{
  Soft_I2C_Start();
  Soft_I2C_Write(0x78);
  Soft_I2C_Write(0x40);
  Soft_I2C_Write(valore);
  Soft_I2C_Stop();
  delay_us(20);
}//scrivi_char
```

```
void main() {
   ANSEL=0;       //disabilito i convertitori AD
   CMCON=0x07;    //disabilito i comparatori
   OSCCON=0xFC;   //imposto il clock interno a 8MHz

   Soft_I2C_Init();

   setup_display();
   clear_display();
   return_home();
   set_cursor_type(1,1,1);

   set_cursor_pos(2,1);
   scrivi_char('A');
   while(1){}
}
```

Partendo da quanto visto sopra, si possono sviluppare delle semplici funzioni sia per Arduino che per il microcontrollore per scrivere non solo il singolo carattere, ma anche una stringa o il valore di una variabile, come avviene con le librerie dei più comuni display paralelli.