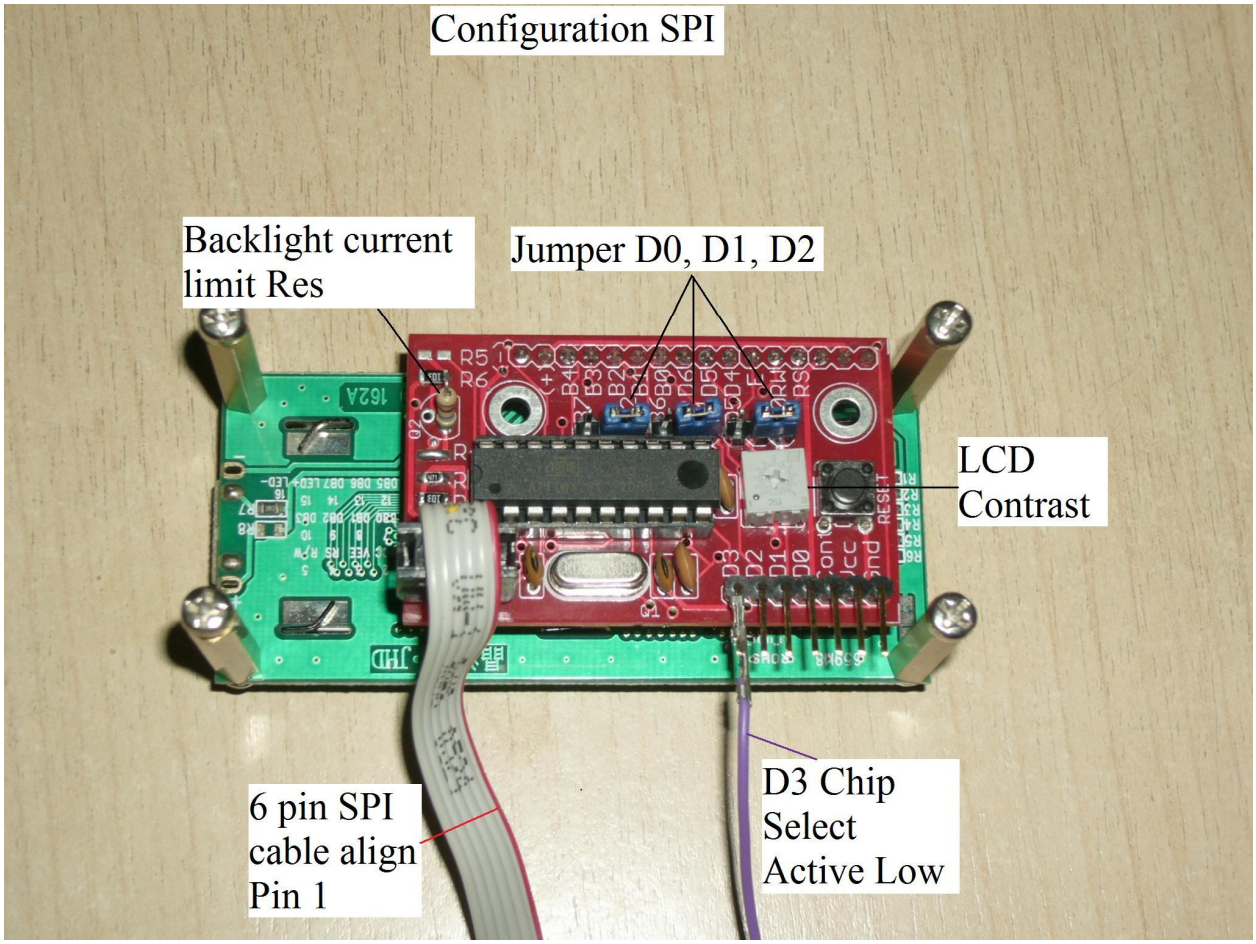
Index

Attiny2313 based Character LCD controller

Connecting LCD display is one of the messiest interconnection in a microcontroller project and taking up valuable I/O for small devices like Atmega168/328. This controller works in two modes as shown in the pictures below. The Serial mode can be connected using the Tx out of microcontroller only.

This controller works by using hex code 250 to differentiate between writing instruction register and data register. Using two consecutive bytes , hex 250 followed by instruction data byte will write to instruction register of the LCD. Every data to be sent to Instruction register must be constructed this way using two bytes. The display data (to data register) can be send as continuous string. This way you can also re-initialize the LCD display from your code as shown in the Arduino sketch below. The controller has been tested with many 16x2 LCD display from different manufacturer.

Important :- The configuration change must always be done in power down mode. The controller cannot change configuration in power up mode.

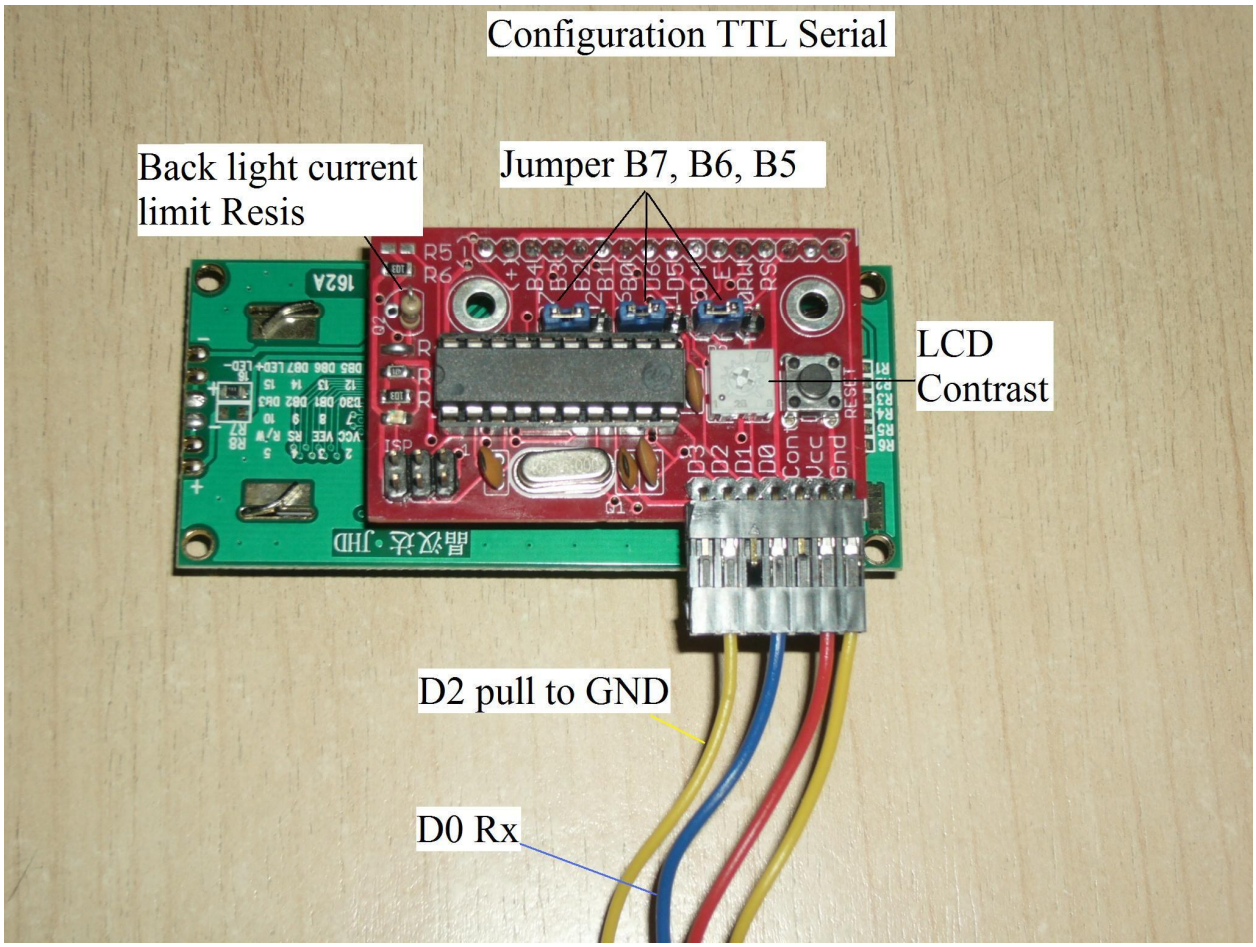


Notes

1) Move the three jumper to short D0, D1,D2
2) Use D3 as Chip select signal. If you do not have any other SPI device than this can be pulled low permanently.
3) The backlight resistor is 500 ohm and it works perfectly with new LCD with high efficiency LED backlight. However this can be sorted also. Infact this was designed for automatic backlight

control using BC547 transistor. Hence you can replace the resistor with BC547 and connect the base to AVR/PIC PWM output through 2K resistor.

4) The reset button will also reset the AVR device.



Notes

1) Move the three jumper to short B7, B6,B5
2) Pull D2 to ground as it is polled at the start to configure the controller to either serial or SPI interface.
3) The backlight resistor is 500 ohm and it works perfectly with new LCD with high efficiency LED backlight. However this can be sorted also . Infact this was designed for automatic backlight

control using BC547 transistor. Hence you can replace the resistor with BC547 and connect the base to AVR/PIC PWM output through 2K resistor.

4) Connect Tx out of microcontroller to D0.

WinAVR C Code

```
#define F_CPU 16000000UL

#define CR 250
#define CD 1

#define L1 0X80   //ROW 0 COL 0
#define L2 0XC0   //ROW 1 COL 0


#include <inttypes.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/sleep.h>
#include <avr/eeprom.h>
#include <math.h>
#include <stdio.h>
#include <util/delay.h>
#include <string.h>
#include <stdlib.h>
#include <avr/wdt.h>
//#include <uart.h>



void ADC_InitA (void);
void PWM_Init (void);
void WDT_off(void);
void WDT_Prescaler_Change(void);
int ADC_Meas(void);
void Comm_write( const char* myval, int b);
void Comm_Transmit( char myval, int b);
void SPI_MasterInit(void);
void SPI_MasterTransmit(char cData);
void USART_Init( );
void USART_Transmit( unsigned char data);



int main(void)
```

```
{
USART_Init();
SPI_MasterInit();

int d, b;
char c[30] ;
//************************
//b = 0 for serial
//b = 1 for SPI
b = 1;
//*****************************


//Clear Display
USART_Transmit( CR );
_delay_us(50);
USART_Transmit( CD );
_delay_us(50);

d = 0;
while(1){
d++;



Comm_Transmit( CR,b );
_delay_us(50);
Comm_Transmit( L1,b );
_delay_us(50);
sprintf(c,"Good welldone!!");
Comm_write( c,b );


Comm_Transmit( CR,b );
_delay_us(50);
Comm_Transmit( L2,b );
_delay_us(50);
sprintf(c,"The Value - %u%cC. ",d, 0xdf);
Comm_write( c,b );


if (d>9){
d = 0;
}

_delay_ms(500);


}

}


void USART_Init( )
{
      /* Set baud rate */
      UBRR0H = 0;
```

```
      UBRR0L = 103;
      UCSR0A &= !(1<<U2X0);//U2X0 = 0
      /* Enable Receiver and Transmitter */
      UCSR0B = (1<<RXEN0)|(1<<TXEN0);
      //UCSR0B = _BV(TXEN0) | _BV(RXEN0);
      /* Set frame format: 8data, 1stop bit */
      UCSR0C = (3<<UCSZ00);
      //OSCCAL = 0XB0;

}

void USART_Transmit( unsigned char data )
{
      /* Wait for empty transmit buffer */
      //! Not. And USCRA with the bit position UDR0E
      while ( !( UCSR0A & (1<<UDRE0)) );
      /* Put data into buffer, sends the data */
      UDR0 = data;
}

void Comm_write(const char* myval, int b)
{
int a = 0;
while (a<strlen(myval)){
if (b == 0){
    USART_Transmit(myval[a]);
      a++;}
if (b > 0){
    SPI_MasterTransmit(myval[a]);
      a++;}
}
}
void Comm_Transmit( char myval, int b)
{


if (b == 0){
    USART_Transmit(myval);
      }
if (b > 0){
    SPI_MasterTransmit(myval);
      }

}

void SPI_MasterInit(void)
{

/* Set MOSI and SCK output, all others input */
DDRB |=(1<<PORTB2)|(1<<PORTB3)|(1<<PORTB5);//Keep portb2 as output
PORTB = (1<<PORTB4);
/* Enable SPI, Master, set clock rate fck/16 - SPR0 = 1, fck/64- SPR1 = 1*/
SPCR = (1<<SPE)|(1<<MSTR)|(1<<SPR0);
}

void SPI_MasterTransmit(char cData)
{
```

```
/* Start transmission */
SPDR = cData;
/* Wait for transmission complete */
while(!(SPSR & (1<<SPIF)))
;
_delay_ms(1);
}
```

Arduino Sketch

Requirement:-

HEspi Library to setup the SPI communication. Please note that the library does not assert any chip_select . Hence the same library can control many SPI connected device by asserting the pins using Digital I/O.

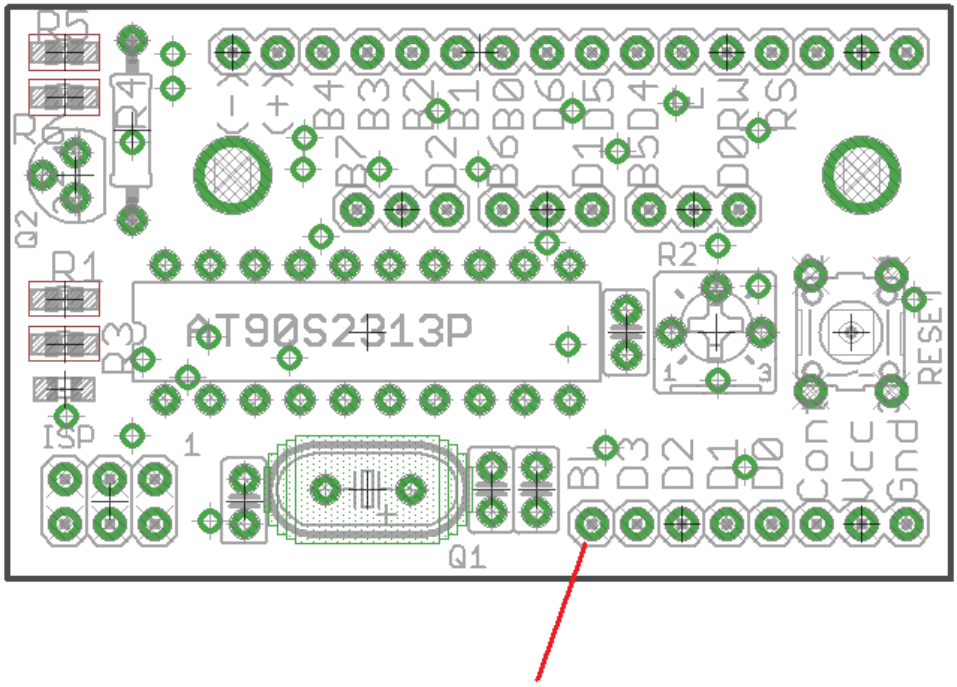
```
//*************************************************
#include <avr/io.h>
#include <stdlib.h>
#include <HEspi.h>

/*LCD CONTROL
CONTROL CHARACTER DECIDED BY INTERNAL OPERATION OF
ATTINY2313 SPI LCD / RS232 CONTROLLER
T0 SEND CONTROL CHARACTERS.
     Serial.write(CR);
     Serial.write(CD);
         OR
     Comm (CD,0)  (0 for SPI and 1 for RS232)

SEND ASCII 250 AND THEN ANY CONTROL CHARACTER  */

#define CR 250  //TO ACCESS CONTROL REGISTER

/*SEE PAGE 191,192,193 OF HITACHI LCD CONTROLLER HD44780 DATSHEET
0 0 0 0 0 0 0 1   */
#define CD 1    //CLEAR DISPLAY

//1 ADD ADD ADD ADD ADD ADD ADD
#define L1 0X80   //ROW 0 COL 0
#define L2 0XC0   //ROW 1 COL 0

// 0 0 0 0 1 D C B
#define CBY 0X0F   //CURSOR ON BLINK ON 00001DCB
#define CBN 0X0C   //CURSOR 0FF BLINK 0FF


int RCByte;
int TCByte = 0xff;
int mv = 0;
char c[30];
HEspi myspi;
void setup()
{
 Serial.begin(9600);
 pinMode(6,OUTPUT); //To SPI LCD
```

```
 pinMode(7,OUTPUT); //To SPI I/O Board.
 digitalWrite(7,HIGH);
 digitalWrite(6,LOW);
 LCD_Reset(0);
 LCD_Reset(1);


}

void loop()                     // run over and over again
{

   digitalWrite(7,HIGH);
   digitalWrite(6,LOW);

   myspi.write(CR);  //  | Alternate |
   myspi.write(CD);  //  | Comm(CD)  |


   Serial.write(CR);
   Serial.write(L1);

   myspi.print("HobbyElectronic");
   sprintf(c,"Trimmer Val %u%%  ",analogRead(0)/10);
   Serial.print(c);
   myspi.write(CR);
   myspi.write(L2);
   Serial.write(CR);
   Serial.write(L2);

   myspi.print("support LCD 1");
   //Serial.print("support LCD 2");
   //
   sprintf(c,"CPU Temp %u%cC  ",analogRead(8)/2,0xdf);
   Serial.print(c);



   delay(500);

   myspi.write(CR);
   myspi.write(CBN);


   //digitalWrite(6,HIGH);
   //digitalWrite(7,LOW);
   ////BLINK LED CONNECTED TO ATTINY2313 SPI OPTOISOLATED I/O BOARD
  //for (int i=0; i <= 5; i++){

   //TCByte = 0;
   //RCByte = myspi.Transmit(TCByte);
   //delay(500);

   //TCByte = 0XFF;
   //RCByte = myspi.Transmit(TCByte);

   //delay(500);
```

```
   //}



}

void LCD_Reset(uint8_t inter) //inter is for interface: 0 for SPI: 1 for
RS232
{
 Comm(0b00110000,inter);
 delay(5);
 Comm(0b00110000,inter);
 delayMicroseconds(100);
 Comm(0b00110000,inter);
 Comm(0b00111000,inter);
 Comm(0b00001100,inter);
 Comm(0b00000001,inter);
 Comm(0b00000110,inter);

}

void Comm(uint8_t mval, uint8_t inter){
   if (inter == 0){
  myspi.write(CR);
   myspi.write(mval);  }
  if (inter == 1){
   Serial.write(CR);
   Serial.write(mval);
}
}
```

Note

We are shipping New boards with following modification. These boards have a new pin input called BL to control the LCD backlight brightness.



1) Connect to uC PWM output for variable brightness
2) Connect to VCC for fixed brightness