Abu Dhabi University

Embedded Networks

Lab Report 1
USART and SPI

Author:
Muhammad Obaidullah 1030313
Sifat Sultan 1029887
Mohamed Hassan Mohamed Al Abasiri 10005290

Supervisor:
Dr. Mohammed Assad Ghazal

Section 1

October 14, 2012

Abstract

In this Lab we communicated with AtMega16 micro-controller using USART hardware subsystem and we also used a SPI protocol to get two micro-controllers to communicate with each other to control LEDs.

1 Introduction

1.1 USART - Universal asynchronous receiver/transmitter

USART (Universal asynchronous receiver/transmitter) allows micro-controllers to receive and send data to and from computer. In this type of communication the hardware has to agree on using the same rate of data transfer on both ends. This is called Baud Rate. Our lab assignment requires us to communicate with computer so we will also need MAX232 chip, which is basically a logic level converter and converts logic 0 = 0V of micro-controller to + 13V of serial port and logic 1 = 5V of micro-controller to -13V of serial port.

The USART pins of the AtMega 16 micro-controller are pins:

- Pin 14 = PD0 is Receive Data.
- Pin 15 = PD1 is Transmit Data.

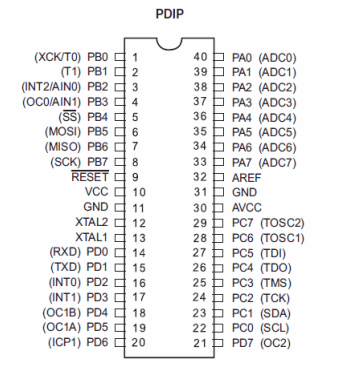

Figure 1: USART and SPI pins of AtMega 16

1.2 SPI - Serial Peripheral Interface

SPI (Serial Peripheral Interface) is a protocol developed by Motorola and is considered as the fastest synchronous interfaces with full duplex serial data transfer. Serial data transfer means that the data transferred one bit at a time and Duplex means that there are separate lines for read and write. This protocol is used by the micro-controllers to communicate with other micro-controllers.

The SPI pins of the AtMega 16 micro-controller are pins:

- Pin 5 = PB4 is Slave Select.
- Pin 6 = PB5 is Master Output Slave Input.
- Pin 7 = PB6 is Master Input Slave Output.
- Pin 8 = PB7 is System Clock.

2 Experiment Set-up

The ATMega16 chip was already mounted on a safety bracket of the master micro- controller. We had to place the bracket with the micro-controller on to the breadboard. Then we connected the micro-processor to each segment and also connected VCC to the common pin. We have connected the USART connection to the master, and connected the SPI connections between the master and the slave micro-controller. Also we have the slave micro-controller connected to the 8 LEDs. *Figure 3*

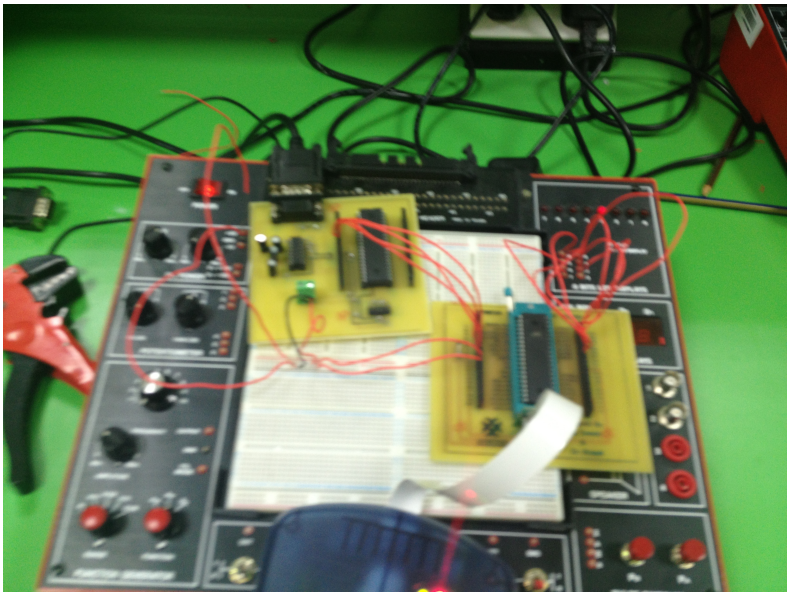

Figure 2: Connecting JTAG MKII to the ATMega 16

Figure 3: Connecting JTAG MKII to the ATMega 16

3 List of Equipment used

- 2 ATMega16 micro-controller chips.
- JTAG MKII programmer.
- Wires.
- Breadboard.
- Mounting bracket for micro-controller.
- 5V power supply.
- AVR Studio IDE.
- 8 LEDs.
- Serial female port.
- Serial cable.

4 Procedure

First, we download the code of the master and the slave using AVR studio and the programmer.Then the master and the slave micro-controllers will establish the master slave relationship via the predefined methods in their code which you will see below, and the slave will wait for a string to be sent from the master.Second, we input the led number that we wanted to light up, for example 5 in the hyper-terminal. The the number that we typed will be sent through the USART connection to the master, this number is represented in Unicode value of the number. At last, the master sent this Unicode to the slave through the SPI connection then the slave converted subtracted this Unicode by 48 to get the real value which we already put then it will shift a 1 positive value to this value on portA which the LEDs are connected to

We lit the LEDs specified by the user input using which was sent to the micro-controller by the USART connection, and the*Figure 3*

4.1 Writing Code for SLAVE.

- Start AVR Studio and click on File/New/New Project.
- Write the following code into the AVR .c file.

```
#include <avr/io.h>

char data,x,z;
void slavemode()
{
        DDRB = DDRB | 0b01001111 ; //SCK, MOSI and SS as inputs
        DDRB = DDRB & 0b01000000 ; //MISO as output
        SPCR &= ~(7<<MSTR) ;           //Set as as Slave
        SPCR |= (1<<SPR0)| (1<<SPR1); //Divide clock by 128
        SPCR |= (1<<SPE) ;            // Enable SPI
}
```

```
int main ()
{
        DDRA = 0XFF;
        PORTA = 0X00;
        x = 0X00;
        data = 0X00;
        slavemode();

        while(1)
        {
                z = PINB & 0b00010000;

                if (z == 0X00) //is SS enabled
                {
                while(!(SPSR & (1<<SPIF)));
                        data = SPDR;
                }
                else{}


                PORTA = 1<<(data-48);


        }
}
```

4.2 Writing Code FOR master.

- Start AVR Studio and click on File/New/New Project.
- Write the following code into the AVR .c file.

```
//Code for the MAster MCU
#include <avr/io.h>
#define USART_BAUDRATE 9600
#define F_CPU 8000000
#define BAUD_PRESCALE ((( F_CPU / ( USART_BAUDRATE * 16UL))) -1)

char test;

void mastermode()
{
        DDRB = DDRB | 0b10110000 ; //SCK, MOSI and SS as outputs
        DDRB = DDRB & 0b10111111 ; //MISO as output
        SPCR |= (1<<MSTR) ;           //Set as a Master
        SPCR |= (1<<SPR0)| (1<<SPR1); //Divide clock by 128
        SPCR |= (1<<SPE) ;            // Enable SPI

}

void SetupUSART()
```

```
{
        UCSRB |= (1<<RXEN)|(1<<TXEN);
        UCSRC |= (1<<URSEL)|(1<<UCSZ0)|(1<<UCSZ1);
        UBRRH = (BAUD_PRESCALE>>8);
        UBRRL = BAUD_PRESCALE;
}



char USART_Receive()
{
        while((UCSRA&(1<<RXC))==0){};
        char Data = UDR;
        return Data;
}


void USART_Transmit(char Data)
{
    while((UCSRA & (1<<UDRE))==0)
        {};
        UDR = Data;//Echo back the received data back to the computer
}

int main(void)
{
        SetupUSART();

        mastermode();

        while(1)
        {
                test = USART_Receive();
                PORTB = PORTB & 0b11101111; //Enable Slave
                SPDR = test;
                while(! (SPSR & (1<<SPIF)));

                PORTB = PORTB | 0b00010000;//Disable Slave
                USART_Transmit(test);

        }
}
```

4.3 Uploading the code to ATMega16.

- Connect JTAG to the computer through a USB cable and connect the JTAG Pins to the micro-controller.
- connect the Seven segment to the Atmega port B.
- Click build and compile in AVR Studio.
- Run the code.

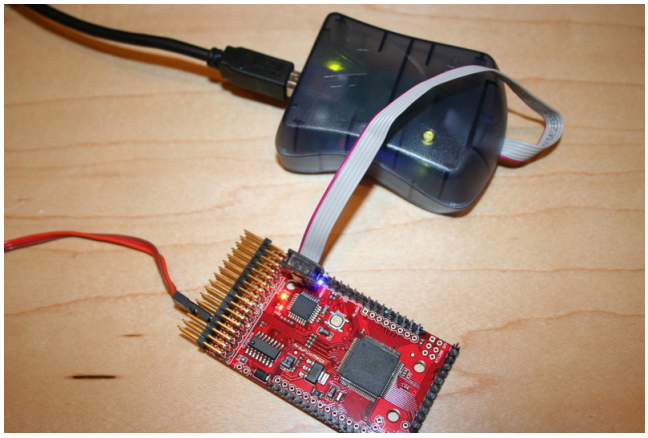

Figure 4: Connecting JTAG MKII to the ATMega 16

5 Results and Discussions

At the end of these exercises we got the following results:-

Figure 5: LED Lighting

- Successful operation of seven segment display.
- If we use timer0, which is a hardware clock, it keeps the micro-controller less busy and allows it to perform other functions.
- For enabling interrupts we have to call a function sei()
- For using interrupts we have to include a header file “avr/interrupt.h”.
- TCCR0 stands for Timer Counter Control Register, which controls the timer clock and also enables it.

- Every digit incremented after 1 second.

6 Conclusion

- The reason that a serial port has so much difference between its High an Low voltages is because of reason of data transfer. When data is transferred over a wire, the wire has some resistance so there is some voltage drop over it. Also bending of the wire, cracks in the wire, impurity of the material may cause some significant noise in the signal transferred. To prevent such in-accuracies the logic voltages in serial port are kept apart and then regained on the other side.
- If a pin is low, ie. grounded, it would light up.
- The ATMEGA16 has 3 hardware timers for performing several operations.
- SPI is much advanced and convenient way to communicate between micro-controllers than USART interms of hardware and software as it involves minimum pin and minimum code.