**Department of Electrical Engineering and   
Computer Science**

**Faculty Member:** Dr. Arbab Latif  **Dated:** 09/05/2022

**Semester:** 4th **Section:** BEE 12C

**EE-222: Microprocessor Systems**

Lab 11: SPI Protocol

Group Members

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Reg. No** | **Report**  **Marks / 10** | **Viva**  **Marks / 5** | **Total**  **Marks / 15** |
| Danial Ahmad | 331388 |  |  |  |
| Muhammad Umer | 345834 |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# Experiment

## Objectives

1. Use SPI protocol to transmit/receive data on AVR ATmega

## Equipment

Software

* *Atmel Studio*

Hardware

* ATmega16A Microcontroller Unit
* Universal Programmer
* Seven Segment Display
* Resistance 47Ω
* LEDs (may use from trainer kit)
* USB-TTL Converter.

## Introduction

 Atmel Studio IDE is a free development environment for programming Atmel MCUs, sourced by Microchip Technology Inc. It provides us with the means to simulate assembly language codes on specific Microcontrollers and provides an easy and intuitive way of producing .HEX files, which are what makes burning the code on the hardware possible.

In this specific lab, we familiarize ourselves with Serial Peripheral Interface (SPI) Protocol of the ATmega16A. We brief ourselves over the initialization of the protocol itself and learn how to connect multiple ATmega16A’s with each other, enabling us to receive and send data from each, to each, serially.

# Lab Tasks

## Task A

Implement the code given in website, to establish SPI communication on ATmega16. Patch the Hardware and demonstrate.

**Master Code**

**#define *F\_CPU* 1000000UL /\* Define CPU Frequency 1MHz \*/**

**#include <avr/io.h> /\* Include AVR std. library file \*/**

**#include <util/delay.h> /\* Include Delay header file \*/**

**#include <stdio.h> /\* Include Std. i/p o/p file \*/**

**#include <string.h> /\* Include String header file \*/**

**#define MOSI 5 /\* Define SPI bus pins \*/**

**#define MISO 6**

**#define SCK 7**

**#define SS 4**

**#define SS\_Enable PORTB &= ~(1<<SS) /\* Define Slave enable \*/**

**#define SS\_Disable PORTB |= (1<<SS) /\* Define Slave disable \*/**

**void SPI\_Init() /\* SPI Initialize function \*/**

**{**

**DDRB |= (1<<MOSI)|(1<<SCK)|(1<<SS); /\* Make MOSI, SCK, SS**

**as Output pin \*/**

**DDRB &= ~(1<<MISO); /\* Make MISO pin**

**as input pin \*/**

**PORTB |= (1<<SS); /\* Make high on SS pin \*/**

**SPCR = (1<<SPE)|(1<<MSTR)|(1<<SPR0); /\* Enable SPI in master mode \*/**

**SPSR &= ~(1<<SPI2X); /\* Disable speed doubler \*/**

**}**

**void SPI\_Write(char data) /\* SPI write data function \*/**

**{**

**SPDR = data; /\* Write data to SPI data register \*/**

**while(!(SPSR & (1<<SPIF))); /\* Wait till transmission complete \*/**

**}**

**int main(void)**

**{**

**DDRD = 0xFF;**

**char count;**

**SPI\_Init();**

**count = 0;**

**while (1) /\* Send Continuous count \*/**

**{**

**SS\_Enable;**

**SPI\_Write(count);**

**count++;**

***\_delay\_ms*(500);**

**PORTD = count;**

**}**

**}**

**Slave Code**

**#define *F\_CPU* 1000000UL /\* Define CPU Frequency 1MHz \*/**

**#include <avr/io.h> /\* Include AVR std. library file \*/**

**#include <util/delay.h> /\* Include Delay header file \*/**

**#include <stdio.h> /\* Include Std. i/p o/p file \*/**

**#include <string.h> /\* Include String header file \*/**

**#define MOSI 6**

**#define SCK 8**

**#define MISO 7**

**#define SS 5**

**void SPI\_Init() /\* SPI Initialize function \*/**

**{**

**DDRB &= ~((1<<MOSI)|(1<<SCK)|(1<<SS)); /\* Make MOSI, SCK, SS as input pins \*/**

**DDRB |= (1<<MISO); /\* Make MISO pin as output pin \*/**

**SPCR = (1<<SPE)|(1<<SPR0); /\* Enable SPI in slave mode \*/**

**}**

**char SPI\_Receive() /\* SPI Receive data function \*/**

**{**

**while(!(SPSR & (1<<SPIF))); /\* Wait till reception complete \*/**

**return(SPDR); /\* Return received data \*/**

**}**

**int main(void)**

**{**

**DDRD = 0xFF;**

**SPI\_Init();**

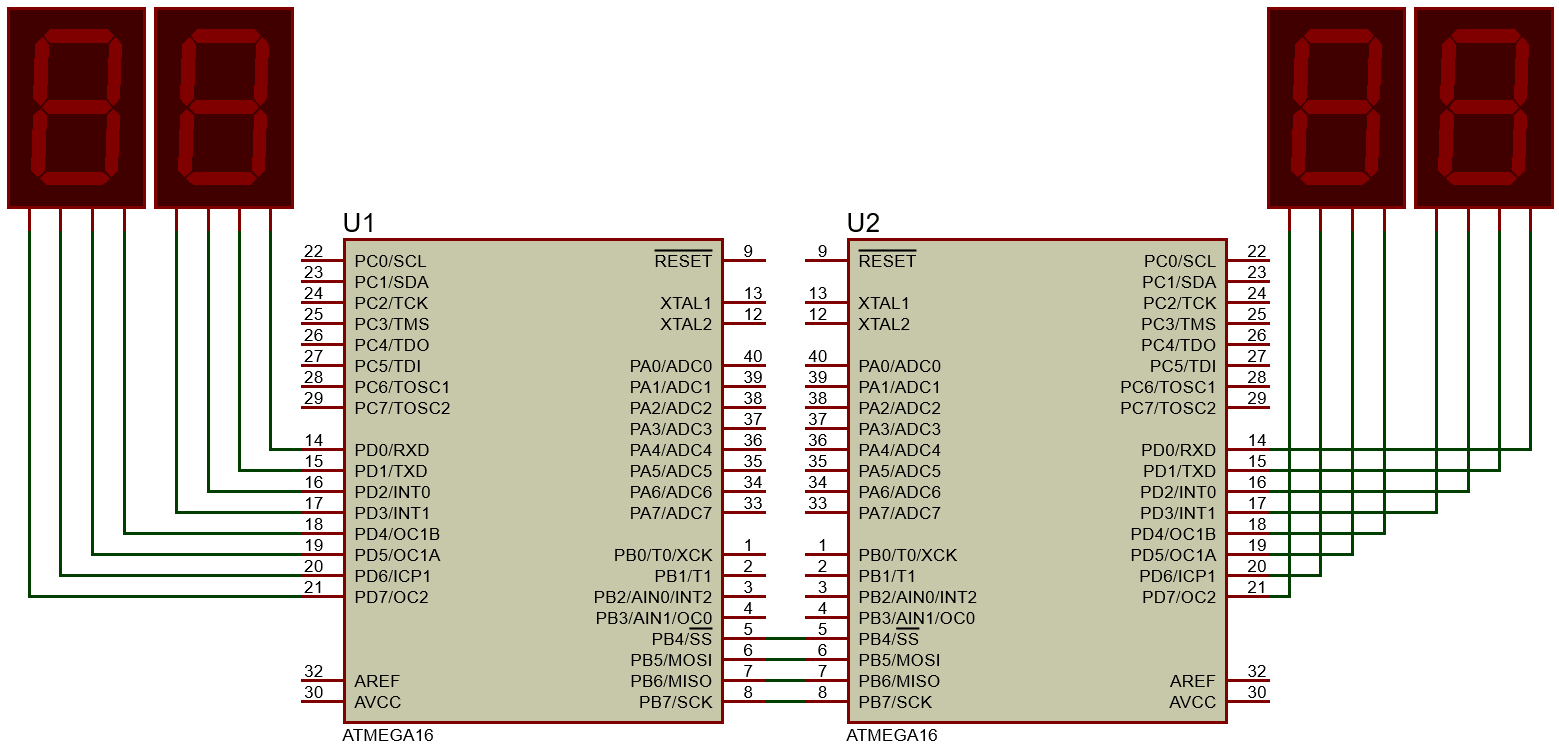
**while (1) /\* Send Continuous count \*/**

**{**

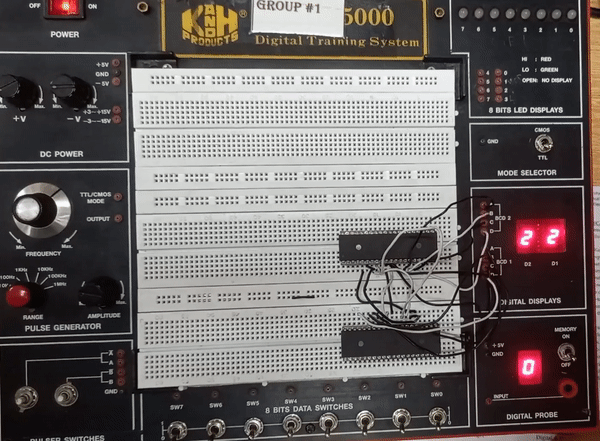
**PORTD = SPI\_Receive();**

**}**

**}**



**Simulation**



**Hardware Implementation**

## Task B

Use Proteus to implement the example 17-1 code using AVR C Programming.

**Master Code**

**#define *F\_CPU* 1000000UL /\* Define CPU Frequency 1MHz \*/**

**#include <avr/io.h> /\* Include AVR std. library file \*/**

**#include <util/delay.h> /\* Include Delay header file \*/**

**#include <stdio.h> /\* Include Std. i/p o/p file \*/**

**#include <string.h> /\* Include String header file \*/**

**#define MOSI 5 /\* Define SPI bus pins \*/**

**#define MISO 6**

**#define SCK 7**

**#define SS 4**

**#define SS\_Enable PORTB &= ~(1<<SS) /\* Define Slave enable \*/**

**#define SS\_Disable PORTB |= (1<<SS) /\* Define Slave disable \*/**

**void SPI\_Init() /\* SPI Initialize function \*/**

**{**

**DDRB |= (1<<MOSI)|(1<<SCK)|(1<<SS); /\* Make MOSI, SCK, SS**

**as Output pin \*/**

**DDRB &= ~(1<<MISO); /\* Make MISO pin as input pin \*/**

**PORTB |= (1<<SS); /\* Make high on SS pin \*/**

**SPCR = (1<<SPE)|(1<<MSTR)|(1<<SPR0); /\* Enable SPI in master mode \*/**

**SPSR &= ~(1<<SPI2X); /\* Disable speed doubler \*/**

**}**

**void SPI\_Transmit(char data) /\* SPI write data function \*/**

**{**

**SS\_Enable;**

**SPDR = data; /\* Write data to SPI data register \*/**

**while(!(SPSR & (1<<SPIF))); /\* Wait till transmission complete \*/**

**PORTA = SPDR;**

**SS\_Disable;**

**}**

**int main(void)**

**{**

**DDRA = 0xFF;**

**char data;**

**SPI\_Init();**

***\_delay\_ms*(200);**

**data = 'G';**

**while (1) /\* Send Continuous count \*/**

**{**

**SPI\_Transmit(data);**

***\_delay\_ms*(100);**

**}**

**}**

**Slave Code**

**#define *F\_CPU* 1000000UL /\* Define CPU Frequency 8MHz \*/**

**#include <avr/io.h> /\* Include AVR std. library file \*/**

**#include <util/delay.h> /\* Include Delay header file \*/**

**#include <stdio.h> /\* Include Std. i/p o/p file \*/**

**#include <string.h> /\* Include String header file \*/**

**#define MOSI 6**

**#define SCK 8**

**#define MISO 7**

**#define SS 5**

**void SPI\_Init() /\* SPI Initialize function \*/**

**{**

**DDRB &= ~((1<<MOSI)|(1<<SCK)|(1<<SS)); /\* Make MOSI, SCK, SS as input pins \*/**

**DDRB |= (1<<MISO); /\* Make MISO pin as output pin \*/**

**SPCR = (1<<SPE)|(1<<SPR0); /\* Enable SPI in slave mode \*/**

**}**

**char SPI\_Receive() /\* SPI Receive data function \*/**

**{**

**while(!(SPSR & (1<<SPIF))); /\* Wait till reception complete \*/**

**return(SPDR); /\* Return received data \*/**

**}**

**int main(void)**

**{**

**DDRA = 0xFF;**

**SPI\_Init();**

***\_delay\_ms*(200);**

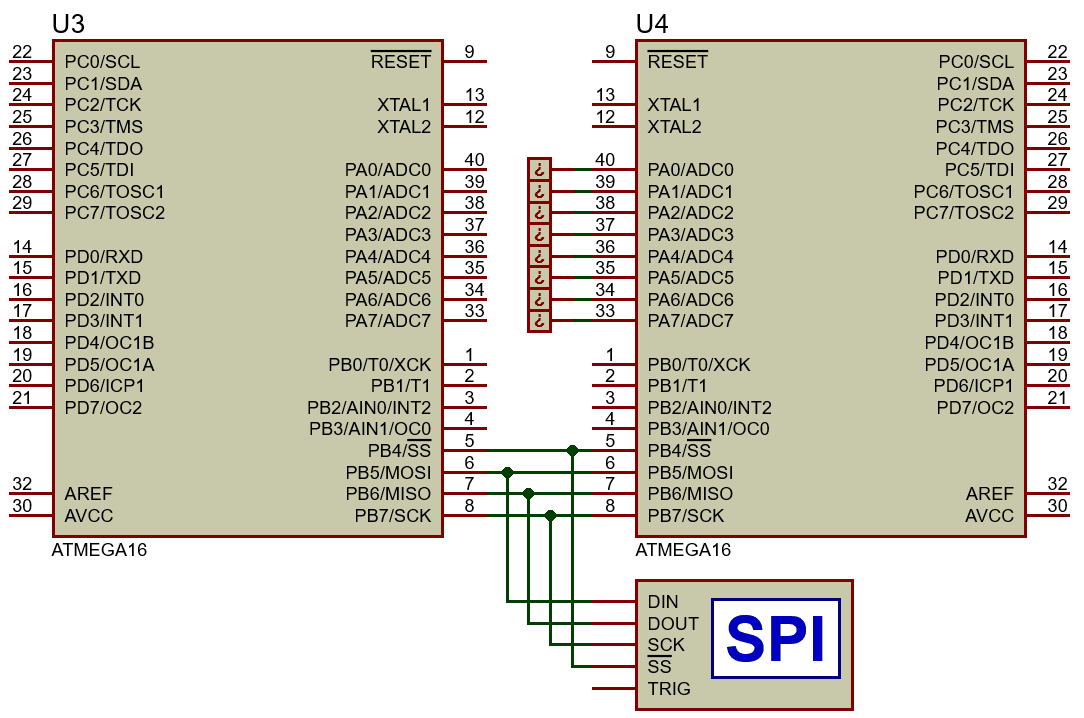
**while (1) /\* Send Continuous count \*/**

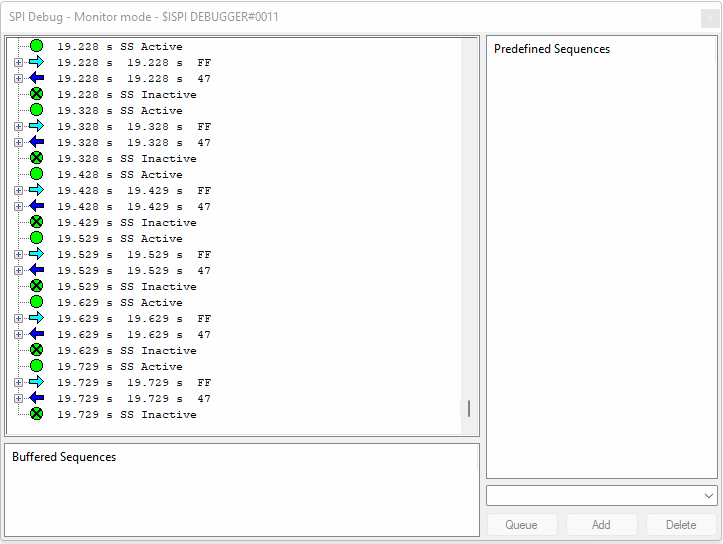
**{**

**PORTA = SPI\_Receive();**

**}**

**}**





**HEX 47 = ‘G’, which is being transmitted to the slave repeatedly.**

## Conclusion

After the conduction of this lab, we have learnt how to initialize and setup the SPI module of the ATmega16A. This protocol serves its purpose best when the master device Is used to communicate with multiple slave devices. In this lab, we initialized the SPI protocol to implement a simple one-to-one master – slave interface and verified that the data sent from the master is the same as the data received by the slave. We also used the SPI debugger present within the toolkit of Proteus to monitor the outflux of data from the master in conversion of an assembly language program to its C variant.