```
////////////////////////////////////////////////////////////////////////////////////////////////////////////
/////////

// this program enables SPI communication and

// Sets the AVR into Master mode



#include <avr/io.h>

#define F_CPU 1000000UL

#include <util/delay.h>


unsigned int higherByte = (unsigned int)(0x07); // 0x07 can be changed to any other byte value

unsigned int lowerByte = (unsigned int)(0xaa);  // 0xaa can also be changed to any other byte value


void SPI_init()

{

        DDRB |= (1<<PB2)|(1<<PB3)|(1<<PB5);    // SCK, MOSI and SS as outputs

        DDRB &= ~(1<<PB4);                 // MISO as input

        SPCR |= (1<<MSTR);                // Set as Master

        SPCR |= (1<<SPR0)|(1<<SPR1);       // divided clock by 128

        SPCR |= (1<<SPE);                // Enable SPI

        //SPCR |= (1<< DORD);           // Set the DORD bit high if you only want to transfer LSB first

}


void send_data()

{

        PORTB |=  (1<<PB2);                    //Set SS high to synchronize with slave
```

```
	PORTB &= ~(1<<PB2);                    //Clear SS
	SPDR = higherByte;                     // sending higher byte
	while(!(SPSR & (1<<SPIF)));            // wait until transmission is complete
	PORTB |=  (1<<PB2);                    //SS high - end of transmission
	////////////////////////////////////////
	//_delay_ms(1000);
	////////////////////////////////////////
	PORTB |=  (1<<PB2);                //Set SS high to synchronize with slave
	PORTB &= ~(1<<PB2);                //Clear SS
	SPDR = lowerByte;                  // sending lower byte
	while(!(SPSR & (1<<SPIF)));    // wait until transmission is complete
	PORTB |=  (1<<PB2);            //SS high - end of transmission




}

int main (void)
{

	SPI_init();


	while(1)
	{
		send_data();
```

```
        }
        return 0;
}

////////////////////////////////////////////////////////////////////////////////////////////////////
////////////

// this program enables SPI communication and
// Sets the AVR into Slave mode
#include <avr/io.h>
#define F_CPU 1000000UL
#include <avr/interrupt.h>
#include <util/delay.h>     // some of the Header files are not actually required,
                    // I included them because this was part of some other project
                                                        //  that I have been working on.
unsigned int result, result1, result2;
unsigned int higherByte, lowerByte;
//
//define functions to be used
void spi_init();
void chip_init();
unsigned int recieve_SPI_data(void);
//
//////////////////////////////
void spi_init()
{
        SPCR &= ~(1<<MSTR);             // Set as slave
```

```
        SPCR |= (1<<SPR0)|(1<<SPR1);      // divide clock by 128
        SPCR |= (1<<SPE);               // Enable SPI

}
//////////////////////////////
void chip_init()
{
        DDRB &= ~((1<<PB2)|(1<<PB3)|(1<<PB5));       // SCK, MOSI and SS as inputs
        DDRB |= (1<<PB4);                    // MISO as output
        DDRC |= (0x0f);                     // Configure PC0 - PC3 as output pins on PORTC
        DDRD |= (0xff);                     // Configure PORTD as an output PORT
}
//////////////////////////////
unsigned int recieve_SPI_data()
{
        //////////////////////////////

        //while((SPSR & (1<<SPIF)) == 0);     // wait until a byte is received
        while(!(SPSR & (1<<SPIF)));         // wait until a byte is received
        lowerByte = SPDR;                // Note: SPDR contains contents of the last byte sent

  //while((SPSR & (1<<SPIF)) == 0);   // wait until a byte is received
        while(!(SPSR & (1<<SPIF)));      // wait until a byte is received
        higherByte = SPDR;              //  Note: SPDR contains contents of first byte sent
```

```
	result1 = (higherByte << 8);
	result2 =(lowerByte);

	result = (uint16_t)(result1 | result2);   // combine contents of byte one and byte two.
	return result;
}
////////////////////////////////////
// the recieve_SPI_data(); function
// can be called in any other function
// like the main.
// For my case, i used it in void display_result(unsigned int result)
// as seen in my AVR Seven segment display.
// It the very first function that I call in that function
////////////////////////////////////
int main (void)
{
	chip_init();
	spi_init();
	//set_timer();


	while(1)
	{

	}
```

```
        return 0;
}
```