Application Note

W3150A+ Guide in SPI mode

Document History

Ver 1.0 (Sep 07, 2006)	First release

For more information, visit our website at http://www.wiznet.co.kr

This guide provides the information on the usage of W3150A+ in SPI mode.

1. Using W3150A+ in SPI Mode

For the data communication through the SPI (Serial Peripheral Interface), four pins are used – SCLK, /SS, MOSI, MISO.

In the W3150A+, one more pin is needed for enabling SPI Operation - SPI_EN pin.

By Asserting high SPI_EN pin, the A[14~11] pins turn to SCLK, /SS, MOSI, MISO.

Remember that /SS, SCLK, MOSI pins are for input, and MISO pin is for output.

Below Figure 1 shows the connection between SPI master(MCU) and SPI slave(W3150A+)

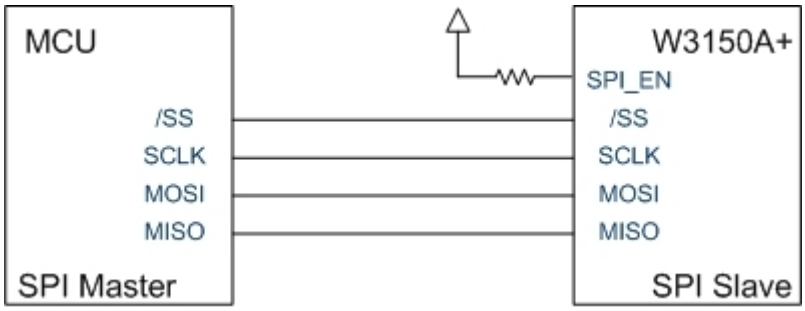


Figure 1. Connection between MCU and W3150A+

2. Device Operation

W3150A+ operates as a SPI slave device. It is controlled by a set of instruction sent by a host controller, commonly referred to as the SPI master

The SPI protocol defines four types of operation mode (Mode 0, 1, 2, 3). Each mode is differentiated by the SCLK polarity, phase, and their control method of data flow on the SPI data bus.

As a SPI Slave device, W3150A+ supports two types of operation mode – SPI Mode 0 and 3, the most common mode. The difference of SPI Mode 0 and 3 is the polarity of the SCLK signal in the inactive state. At the SPI Mode 0 and 3, data is always latched in on the rising edge of SCLK and output on the falling edge of SCLK.

Refer to below for W3150A+ SPI Mode 0 and 3.

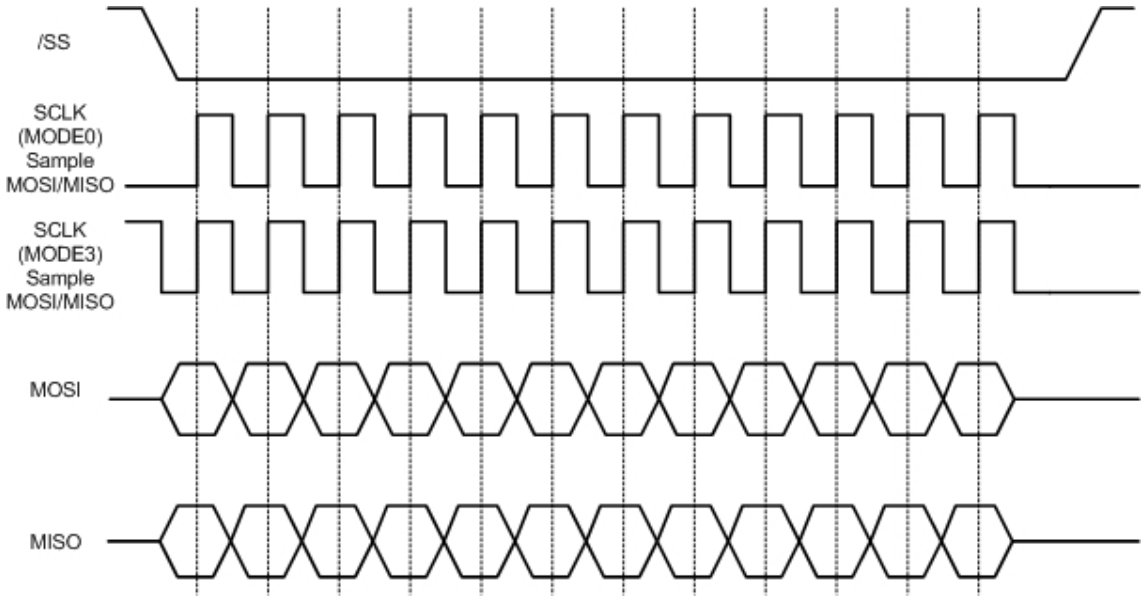


Figure 2. SPI Mode 0 and 3

3. W3150A+ SPI Commands

As like general SPI slave devices, W3150A+ uses two types of OP-Code – Read OP-Code and Write OP-Code. For other OP-Codes except for those two, W3150A+ will ignore the commands and there will be no operation.

In SPI Mode, W3150A+ operates in "unit of 32-bit stream".

The unit of 32-bit stream composed of 1 byte OP-Code Field, 2 bytes Address Field and 1 byte data Field.

OP-Code, Address and data bytes are transferred with the most significant bit(MSB) first and least significant bit(LSB) last. In other words, the first bit of SPI data is MSB of OP-Code Field and the last bit of SPI data is LSB of Data-Field.

W3150A+ SPI data format is below.

Command	OP-Code Field		Address Field	Data Field
Write operation	0xF0	1111 0000	2 bytes	1 byte
Read operation	0x0F	0000 1111	2 bytes	1 byte

4. Process of using W3150A+ in SPI mode

The process is generally introduced in SPI protocol.

Assume that SPDR is 8bit register.

1. Configure Input / Output direction on SPI Master device pins
2. Configure /SS as 'High'
3. Configure the registers on SPI Master device
4. Configure /SS as 'Low' (data transfer start)
5. Write value in OP-code Field & Wait for reception complete
6. Write value in upper address field & Wait for reception complete
7. Write value in lower address field & Wait for reception complete
8. Write value in data field & Wait for reception complete
9. The difference of writing and reading operations
   - 9-1. In case of writing operation
     - A. Write data value
   - 9-2. In case of reading operation
     - A. Write dummy value for normal SPI communication (Any value is OK)
     - B. If reception is complete, you can use SPDR register value.
10. configure /SS as 'High'

W3150A+ SPI Writing/Reading operation shows below.

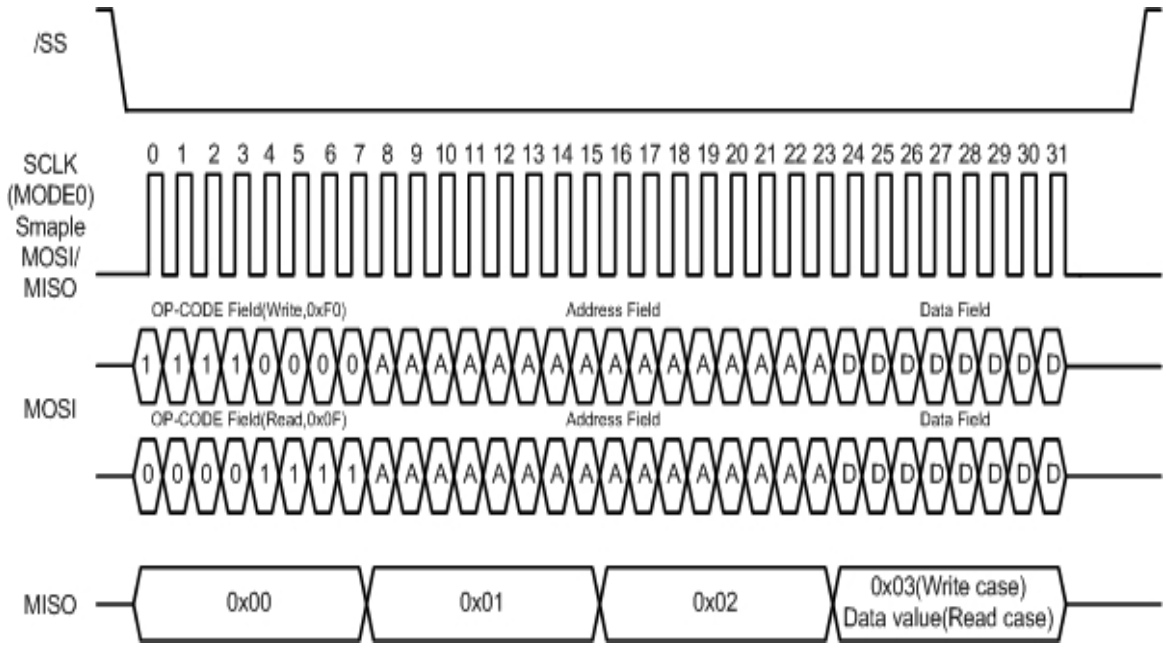


Figure 3. W3150A+ SPI Writing/Reading operation

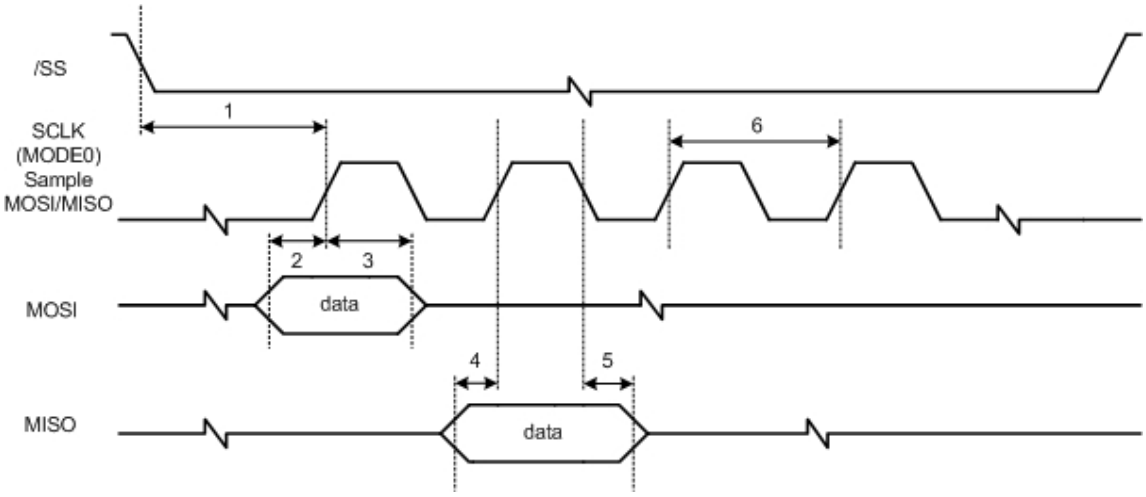
5. 3150A+ SPI sample code for AVR

Operation	Sample code based on WIZNET EVB board using AVR128
Writing Operation	/* 1. Define port direction (In case of using AVR, Port B is used for SPI communication) PB0 (/SS, Output, 'High'), PB1 (SCLK, Output, 'High') PB2 (MOSI, Output, 'High'), PB3 (MISO, Input , ' Low') */ DDRB = 0x07 ; /* 2. Standing for SPI communication, /SS = 'high' */ PORTB = 0x01 ; /* 3. Define SPI Enable bit , Master/Slave bit , SPI Mode bit , SPI Data rate bit */ SPCR = 0x50 ; /* SPI Control Register */ SPSR = 0x01 ; /* SPI State Register */ /* 4. start SPI communication, /SS = 'low' */ PORTB = 0x00 ; /* 5.1 Write desired value for transmission, OP-Code Field, Write OP-Code(0xF0) */ SPDR = 0xF0 ; /* 5.2 Wait for reception complete */ While ((SPSR&0x80)==0x00) ; /* 6.1 Write desired value for transmission, Address Field, Upper address value */ SPDR = (addr&0xFF00) ; /* 6.2 Wait for reception complete */

	While ((SPSR&0x80)==0x00) ; /* 7.1 Write desired value for transmission, Address Field, lower address value */ SPDR = (addr&0x00FF) ; /* 7.2 Wait for reception complete */ While ((SPSR&0x80)==0x00) ; /* 8.1 Write desired value for transmission, Data Field */ SPDR = data ; /* 8.2 Wait for reception complete */ While ((SPSR&0x80)==0x00) ; /* 9. data transmission ends, /SS='High' */ PORTB =0x01 ;
Read Operation	/* 1. Define port direction (In case of using AVR, Port B is used for SPI communication) PB0 (/SS , Output, 'High'), PB1 (SCLK, Output, 'High') PB2 (MOSI, Output, 'High'), PB3 (MISO, Input , ' Low') */ DDRB = 0x07 ; /* 2. standing for SPI communication ,/SS = 'high'*/ PORTB = 0x01 ; /* 3. Define SPI Enable bit, Master/Slave bit , SPI Mode bit , SPI Data rate bit */ SPCR = 0x50 ; /* SPI Control Register */ SPSR = 0x01 ; /* SPI State Register */ /* 4. start SPI communication, /SS = 'low' */ PORTB = 0x00 ; /* 5.1 Write desired value for transmission,

```
        OP-Code Field, Read OP-Code(0x0F) */
SPDR    = 0x0F ;
/* 5.2 Wait for reception complete */

While   ((SPSR&0x80)==0x00) ;
/* 6.1 Write desired value for transmission,
        Address Field, Upper address value */
SPDR    = (addr&0xFF00) ;
/* 6.2 Wait for reception complete */
While   ((SPSR&0x80)==0x00) ;

/* 7.1 Write desired value for transmission,
        Address Field, lower address value */
SPDR    = (addr&0x00FF) ;
/* 7.2 Wait for reception complete */
While   ((SPSR&0x80)==0x00) ;

/* 8.1 Write dummy data in the data Field
   SPI protocol simultaneously exchanges data between SPI Master and
   SPI Slave. When reading operation, there is no data transferred from
   SPI Master to SPI Slave in Data Field. But remember that Data Field
   should be filled with dummy data. Sample code uses 0x00 as dummy
   data. Any data is no matter. Omission of writing dummy data will
   can cause some errors. */
SPDR    = data ;
/* 8.2 Wait for reception complete */
While   ((SPSR&0x80)==0x00) ;

/* Copy receiving data to data register*/
Data= SPDR;

/*   /SS='High', data transmission ends */
PORTB =0x01 ;
```

6. SPI Timing



	Description	Mode	Min	Max
1	/SS low to SCLK	Slave	21 ns	-
2	Input setup time	Slave	7 ns	-
3	Input hold time	Slave	28 ns	-
4	Output setup time	Slave	7 ns	14 ns
5	Output hold time	Slave	21 ns	-
6	SCLK time	Slave	70 ns	