

SPI-to-WiFi Guide

(WizFi 250 Application Notes V0.9)



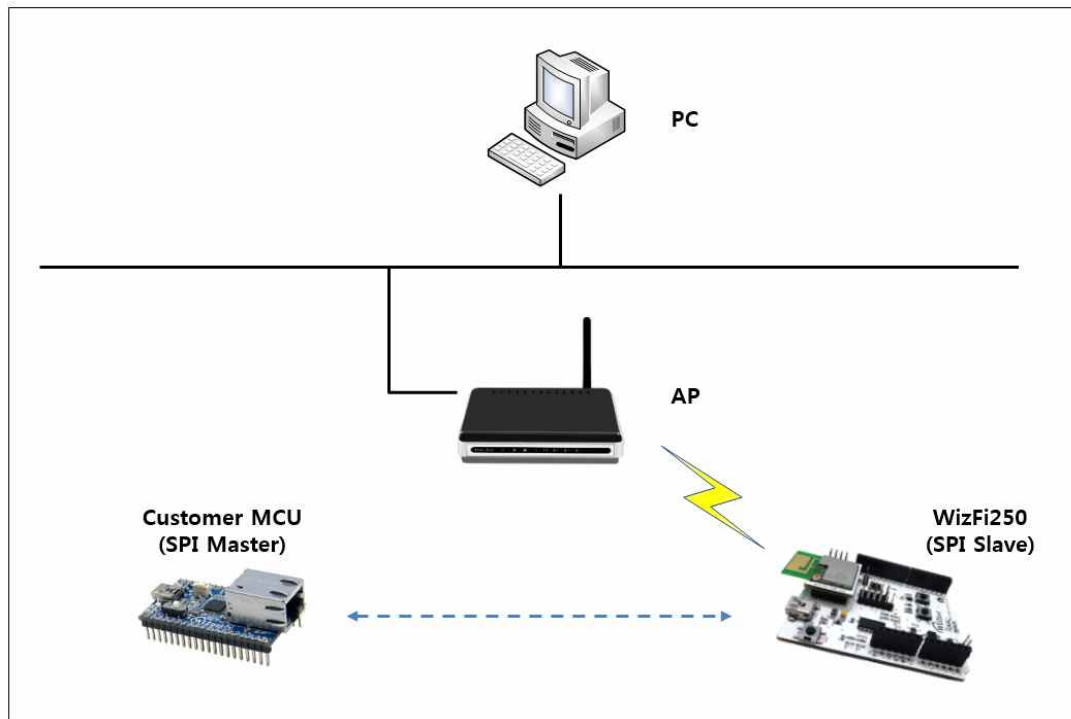
©2013 WIZnet Co., Ltd. All Rights Reserved.

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

Document Revision History

Date	Revision	Changes

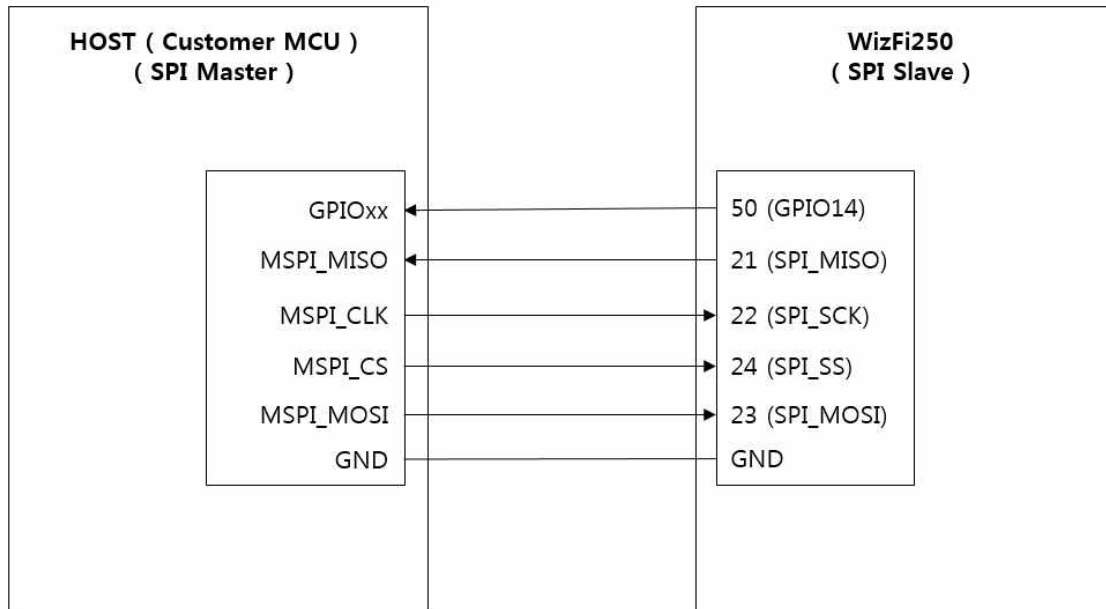
1. Diagram for example of SPI Interface



In the case of SPI interface, WizFi250 acts as slave and will communicate to master SPI controller. By default, SPI interface supports Motorola protocol with clock polarity 0 and clock phase 0.

Since SPI data transfer works in full duplex mode, it's required to make use of special octet to indicate idle data. Similarly if host MCU is sending data at higher rate flow control mechanism is required. In order differentiate these special control codes (such as idle pattern, flow control codes and other control octets) from user data, byte stuffing mechanism is incorporated.

2. Pin connections for SPI



Host App (SPI Master)	WizFi210 (SPI Slave)	Remarks
Allocate your GPIO	GPIO14 (50)	WizFi250 Data Ready
MSPI_MISO	SPI_MISO (21)	SPI Master In/Slave Out
MSPI_CLK	SPI_SCK (22)	SPI Clock
MSPI_CS	SPI_SS (24)	SPI Chip Select
MSPI_MOSI	SPI_MOSI (23)	SPI Master Out/Slave In
GND	GND	Command Ground

3. SPI interface details

In case of SPI interface additional task is required to handle SPI data transfer. SPI data transfer

- Only Motorola mode is supported
- Only 8 bit SPI data word size is supported
- By default SPI Mode#0 is selected (CPOL =0 and CPH=0)
- Under 562.5KHz master clock is supported

Motorola SPI Format with CPL=0, CPH=0

4. <WizFi250 Data Ready> Signal Handling

<WizFi250 Data Ready> signal is ACTIVE HIGH signal. Host controller must give the SPI clock, as long as <WizFi250 Data Ready> signal is HIGH.

Whenever WizFi250 wants to transfer the data it asserts (HIGH) <WizFi250 Data Ready> signal. Once all the data transferred from WizFi250 it again de-asserts (LOW) the signal.

Host controller will detect the <WizFi250 Data Ready> signal transition (LOW to HIGH) as edge triggered interrupt and process the incoming data.

5. SPI Transmit data handling

The SPI data transfer layer makes use of an octet (or byte) stuffing procedure. The Control Escape octet is defined as binary 11110001 (hexadecimal 0xF1), most significant bit first.

Each special control pattern is replaced by a two octet sequence consisting of the SPI Escape octet followed by the original octet.

Receiving implementations must correctly process all Control Escape sequences.

Escaped data is transmitted on the link as follows:

<i>Pattern</i>	<i>Description</i>
0xF0	SPI NULL
0xF1	SPI Escape
0xF1 0x00	F0 Data
0xF1 0x01	F1 Data
0xF1 0x02	SPI SYNC
0xF1 0x03	SPI XON
0xF1 0x04	SPI XOFF
0xF1 0x05	SPI ERROR

[Byte stuffing control codes]

One dedicated GPIO signal known as <WizFi250 Data Ready> is available for data ready indication from Slave WizFi250 to Master Host controller. Master host controller must provide clock as long as <WizFi250 Data Ready> signal is active. Host controller can make use of GPIO interrupt (edge triggered low-to-high transition) to receive the data from WizFi250.

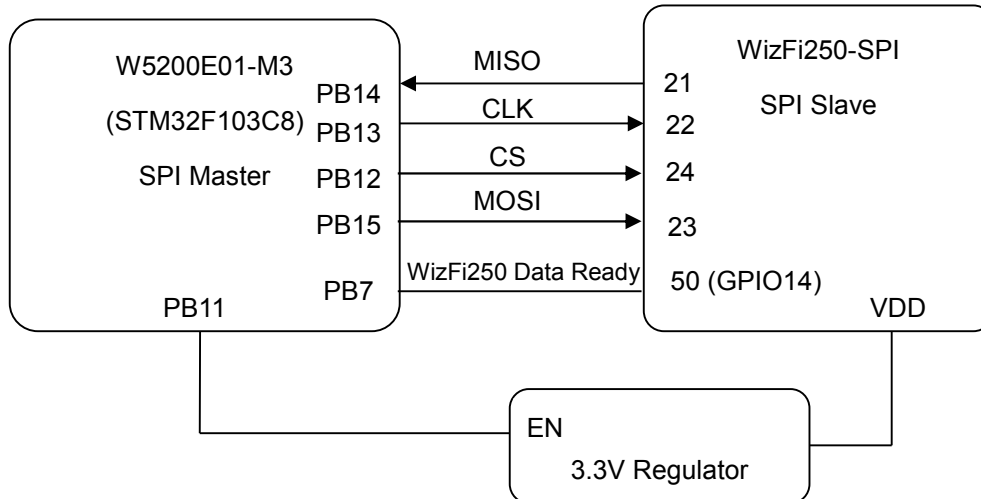
Since SPI data transfer works in full duplex mode, special fill character (**SPI NULL**) will be transmitted during idle period (if there is no more data to transmit). These idle fill pattern shall be dropped at receiving end.

6. SPI Receive data handling

Since byte stuffing is used, each Control Escape octet must be removed, and the next immediate octet is SPI control signal. (**0xF1 0x00** or **0xF1 0x01** are exception. In this case, **0xF1 0x00** is real data of **0xF0** and **0xF1 0x01** is real data of **0xF1**)

If receive buffer is reached upper water mark, then **SPI XOFF(0xF1 0x04)** character will be sent out informing the host to stop transmitting actual data. After receiving SPI XOFF character, host must stop transmitting actual data. Once the application starts processing received data and enough space available for further reception (reached lower water mark), **SPI XON** will be transmitted. Once host receives **SPI XON**, then it can resume the valid data transmission. Special control byte **SPI IDLE** will be dropped at receiver.

7. Implementation Example



- SPI Master : W5200E01-M3 (STM32F103C8) See the below web pages.
http://www.wiznet.co.kr/Sub_Modules/en/product/Product_Detail.asp?cate1=5&cate2=7&cate3=56&pid=1152
<http://www.st.com/internet/mcu/product/164476.jsp>
- PB7 : GPIO of SPI Master, <WizFi250 Data Ready> signal, Active High
- PB11 : To reset WizFi250, control the Enable signal of regulator(Optional)
- SPI Mode 0 (CPOL = 0 and CPHASE = 0)
- Only 8 bit SPI data word size is supported
- **SPI Clock Rate is 200 KHZ(See the SPI Master's source files)**
- WizFi250 send the <SPI NULL(0xF0)> periodically, when it is in idle state.
- SPI Master needs to send the <SPI NULL> to the WizFi250 after reboot to synchronize the SPI interface.
- When <WizFi250 Data Ready> signal is high, SPI Master needs to receive the data which WizFi250 is sending.
- SPI Master can check whether WizFi250 is the data mode or command mode by sending the dummy "AT" command.

- This is pin map between W5200E01-M3 and WizFi210-EVB

	W5200E01-M3	WizFi250-EVB
MSPI_MISO	PB14	JP13 MISO
MSPI_MOSI	PB15	JP13 MOSI
MSPI_CLK	PB13	JP13 SCK
MSPI_CS	PB12	JP13 SS
Allocate your GPIO	PB7	GPIO14
GND		