1. **Problem 1 - To make the clock continuous**

**You suggested** - write multiple times in SPI transmit buffer using array of characters

**Our configuration** – we have separated the write operation and receive operation so that the transmit buffer will be called multiple times sequentially

**Result** – So that now the clock is continuous. The transmission process working fine checked in oscilloscope also the result is good as expected

**Code configuration** –

Void spi\_xmit(unsigned char sdata){

//Yes I know here we have to check the TXFFST FIFO overflow for writing but not checked since am writing only 4 bytes for every one second once for testing purpose and the FIFO size is 16 hence it may never overflow.

 SpibRegs.SPITXBUF = sdata<<8;

}

Unsigned char spi\_rec(unsigned char sdata){

 SpibRegs.SPITXBUF = sdata<<8;

 While(SpibRegs.SPIFFRX.bit.RXFFST == 0);

 rdata = SpibRegs.SPIRXBUF;

}

1. **Problem 2 – Received slave data not found in the receive buffer.**

As checking in the oscilloscope, the transmission is working and the slave is also responding back with proper timing structure as expected.

But when checking in the receive buffer(SpibRegs.SPIRXBUF) am getting 0xFF for the first four reading operation from the fifth read operation am getting the right data as expected

**Debugging results checked in IDE:**

First read operation: @ after 1 second from power ON and write operation

 SpibRegs.SPIRXBUF = 0xFF

 SpibRegs.SPIDAT = 0x6F //expected result

 SpibRegs.SPIFFRX.bit.RXFFST =2

SpibRegs.SPISTS.bit.OVERUN\_FLAG =0

Second read operation: @ after 2 seconds from power ON and write operation

 SpibRegs.SPIRXBUF = 0xFF

 SpibRegs.SPIDAT = 0x6F //expected result

 SpibRegs.SPIFFRX.bit.RXFFST =6

SpibRegs.SPISTS.bit.OVERUN\_FLAG =0

Third read operation: @ after 3 seconds from power ON and write operation

 SpibRegs.SPIRXBUF = 0xFF

 SpibRegs.SPIDAT = 0x6F //expected result

 SpibRegs.SPIFFRX.bit.RXFFST =10

SpibRegs.SPISTS.bit.OVERUN\_FLAG =0

Fourth read operation: @ after 2 seconds from power ON and write operation

 SpibRegs.SPIRXBUF = 0xFF

 SpibRegs.SPIDAT = 0x6F //expected result

 SpibRegs.SPIFFRX.bit.RXFFST =14

SpibRegs.SPISTS.bit.OVERUN\_FLAG =0

Fifth read operation: @ after 2 seconds from power ON and write operation

 SpibRegs.SPIRXBUF = 0x6F

 SpibRegs.SPIDAT = 0x6F //expected result

 SpibRegs.SPIFFRX.bit.RXFFST =16

 SpibRegs.SPISTS.bit.OVERUN\_FLAG =1

**Debug operation 2**:

Since RXFFST count is incrementing in four, I thought it is transferring the data to receive buffer after filling the FIFO register. Hence, I have changed the RXFFIL=1 so that FIFO interrupt will request when there is more than one data. Before it was RXFFIL=4

**Result:**

But this configuration also didn’t work. I mean the above results not changed at the time of fifth read operation only data received in receive buffer.

**Debug operation 3:**

we have tried by disabling the FIFO registers and to receive the data all the time by checking the SPISTS.bit.INT\_FLAG

i.e.,

initialization:

SpiaRegs.SPIFFTX.all =0;

SpiaRegs.SPIFFRX.all =0;

SpiaRegs.SPIFFCT.all =0;

Inside timer every one second once:

SpiaRegs.SPITXBUF = data;

**While(SpiaRegs.SPISTS.bit.INT\_FLAG != 1);**

Received\_data = SpiaRegs.SPIRXBUF;

**Result:**

MCU get hanged in the ‘while(SpiaRegs.SPISTS.bit.INT\_FLAG !=1);

Since the flag register is always 0.

**My Question is**

1. Why am not getting data in receive buffer (Spibregs.SPIRXBUF) for the first four read operation? Even though the data is available in SPIDAT register from the first read operation itself.
2. If I try to read the received data from SPIDAT register am not getting the data in my local variable. The data in SPIDAT is 0x6F whereas the data moved in my local variable is 0x300. So how to read this SPIDAT register?
3. why overrun flag bit is set even though am reading the receive buffer in all read operation?
4. Why FIFO buffer(RXFFST) incrementing in count of two and four since I have configured in byte mode it has to increment in size of one right?
5. Is changing the RXFFIL as like in debug operation 3 is correct or not? If so, why it is not setting the interrupt for buffer fill?
6. Why SPISTS.bit.INT\_FLAG is not getting 1 (in debug operation 3)?

**Oscilloscope results:**



Please help me to resolve this issue...

Thank you.