Application Report
SLOA140–April 2009

Using the SPI Interface With TRF7960

ShreHarsha Rao

ABSTRACT

One of the microcontroller interfacing options available for the TRF7960 is the SPI with SS*. This is also known as 4-wire hardware SPI mode. The TRF7960 device acts as the slave device and the microcontroller/DSP is the master in this configuration.

Contents

List of Figures

List of Tables

1 TRF7960 - SPI With SS* Mode Errata

It is important to note that there are some non-standard conditions when the TRF7960 is operated in the SPI mode. These are listed below and are fixed by software patches to work around them.

NON-STANDARD CONDITIONS	SOFTWARE FIX
SCLK clock polarity switch needed when read operation (single or continuous) is executed.	Firmware fix to switch clock polarity between writes and reads (see Section 1.1).
IRQ Status Register is not automatically cleared after reading.	Dummy read is needed to clear the contents of IRQ status register and hence drive the IRQ pin low (see Section 1.2).
All stand-alone (single-byte) direct commands need additional clock cycle to work. An example is the slot markers (EOF) for ISO 15693 do not work in SPI mode.	All direct command functions need to have this additional SW fix. Direct commands like “Transmit Next Slot” needs to have additional SCLK cycle before SS* goes high (see Section 1.3).
Some of the registers (RX wait time, RX no response wait time) do not take default values when the appropriate protocol is chosen in the ISO control register.	Manually program these defaults again in the initialization routine (see Section 1.4).
Transmitting one byte through the FIFO.	Split the command (See Section 1.5).

The serial interface is in reset while the SS* signal is high. Serial Data-In (MOSI) changes on the falling edge, and are validated in the reader on the rising edge, as shown in Figure 1. Communication is terminated when SS* signal goes inactive (high). All words must be 8-bits long with the MSB transmitted first.

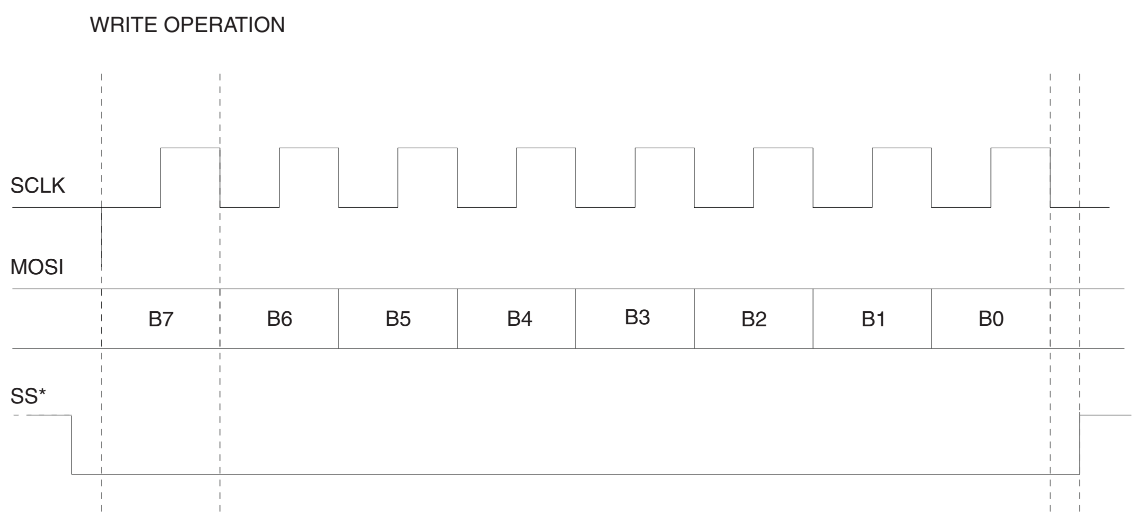


Figure 1. Serial - SPI Interface Communication (Write Mode)

1.1 SCLK Polarity Switch

The SPI read operation is shown in Figure 2 below.

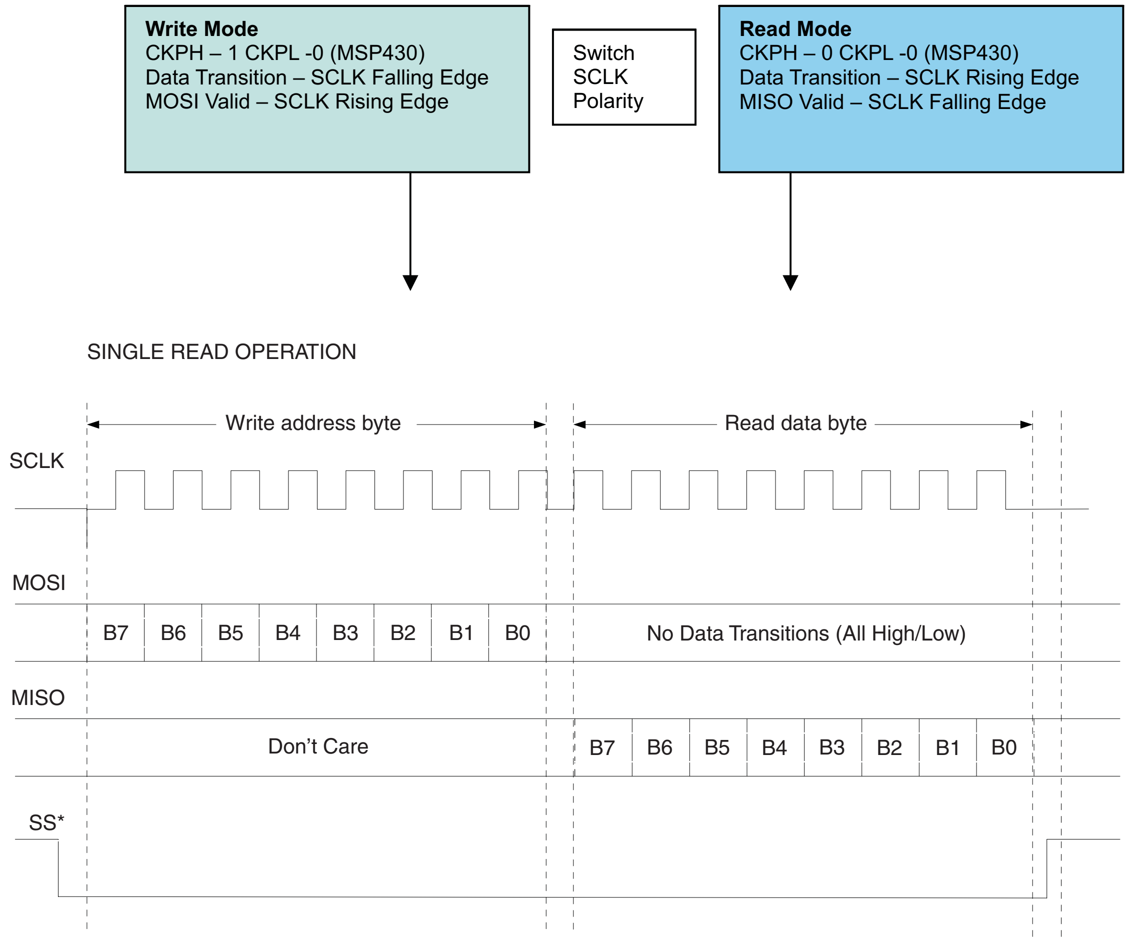


Figure 2. Serial - SPI Interface Communication (Read Mode)

The read command is sent out on the MOSI pin, MSB first in the first 8 clock cycles. MOSI data changes on the falling edge, and is validated in the reader on the rising edge, as shown in Figure 2. During the write cycle the serial data out (MISO) is not valid. After the last read command bit (B0) is validated at the 8th rising edge of SCLK, after half a clock cycle, valid data can be read on the MISO pin at the falling edge of SCLK. It takes 8 clock edges to read out the full byte (MSB first).

Note: When using the hardware SPI (for example, a MSP430 hardware SPI) to implement the above feature, care must be taken to switch the SCLK polarity after write phase for proper read operation. The example clock polarity for the MSP430-specific environment is shown in the box above. Refer to the USARTSPI chapter for any specific microcontroller family for further information on the setting the appropriate clock polarity.

This clock polarity switch NEEDS to be done for all read (single, continuous) operations.

The MOSI (serial data out) should not have any transitions (all high or all low) during the read cycle. Also, the SS* should be low during the whole write and read operation.

The clock polarity switch is illustrated by the following pseudo code. This code refers specifically to the MSP430 platform. Please refer to the datasheet of the relevant microcontroller for your design.

```
        *pbuf = (0x40 |  *pbuf);             // address, read, single
         *pbuf = (0x5f &*pbuf);             //  register address
         while (!(IFG2 & UCB0TXIFG));      // USCI_B0 TX buffer ready?
        UCB0TXBUF = *pbuf;                 // Previous data to TX, RX
       //while (!(IFG2 & UCB0RXIFG));
        temp=UCB0RXBUF;
       UCB0CTL0 &= ~UCCKPH;                          //Switch Clock Polarity

    SPIStartCondition();          //SCLK High/Low to complete the cycle
    P3SEL |=  BIT3;
    while (!(IFG2 & UCB0TXIFG));       // USCI_B0 TX buffer ready?
   UCB0TXBUF = 0x00;             //Receive initiated by a dummy TX write???

    while (!(IFG2 & UCB0RXIFG));
    _NOP();
    _NOP();
    *pbuf = UCB0RXBUF;
     pbuf++;
    lenght--;
   UCB0CTL0 |= UCCKPH;       //revert Back to the Original Clock Polarity
```

The continuous read operation is illustrated in Figure 3.

Continuous Read Operation

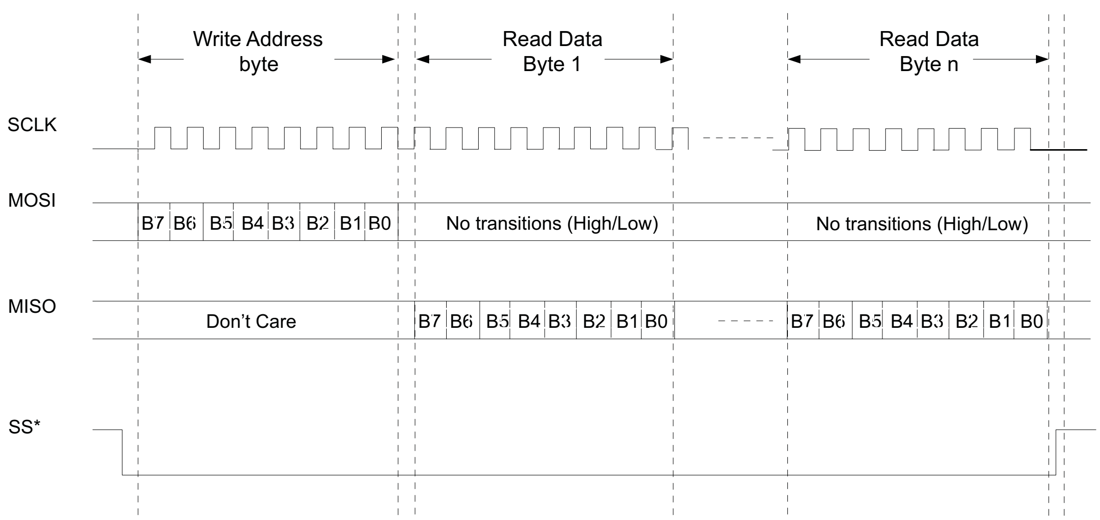


Figure 3. SPI Interface Communication (Continuous Read Mode)

1.2 IRQ Status Register Read

Note: Special steps are needed when you read the TRF796x IRQ status register (register address 0x0C) in SPI mode. The status of the bits in this register are cleared after a “dummy read”. The following steps need to be followed when reading the IRQ status register.

1. Write in command 0x6C: read 'IRQ status' register in continuous mode (8 clocks).
2. Read out the data in register 0x0C (8 clocks).
3. Generate another 8 clocks (as you were reading the data in register 0x0D) but ignore the MISO data line.

This is shown in Figure 4.

Special Case – IRQ Status Register Read

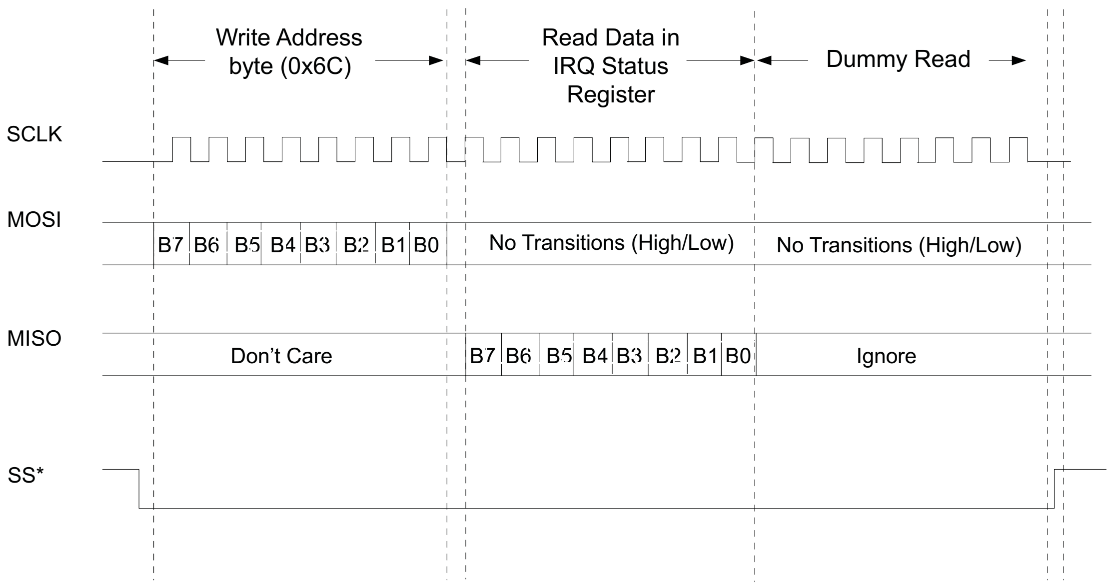


Figure 4. SPI Interface Communication (IRQ Status Register Read)

1.3 Direct Command Processing

The following are the direct commands supported by the TRF7960/61.

Table 1. Command Codes

COMMAND CODE (hex)	COMMAND	COMMENTS
00	Idle	
03	Software initialization	Software initialization, same as power on reset
0F	Reset	
10	Transmission without CRC	
11	Transmission wtih CRC	
12	Delayed transmission without CRC	EPC
13	Delayed transmission with CRC	EPC
14	Transmit next time slot	ISO15693, Tag-it™
16	Block receiver	
17	Enable receiver	
18	Test internal RF (RSSI at RX input with TX off)	
19	Test internal RF (RSSI at RX input with TX on)	
1A	Receiver gain adjust	

Of these the following are the direct commands that needs to have the software fix when using SPI with SS* mode. These are the direct commands that are executed stand-alone (direct commands with just one byte).

Table 2. Direct Commands

DIRECT COMMAND	COMMAND CODE	NEED DUMMY CLOCK
Idle	0x03	Yes
Software initialization	0x00	Yes
Reset	0x0f	Yes
Transmit next time slot	0x14	Yes
Block receiver	0x16	Yes
Enable receiver	0x17	Yes
Test internal RF	0x18	Yes
Test external RF	0x19	Yes
Receiver gain adjust	0x1A	Yes
Transmit without CRC	0x10	No
Transmit with CRC	0x11	No
Delayed transmit without CRC	0x12	No
Delayed transmit with CRC	0x13	No

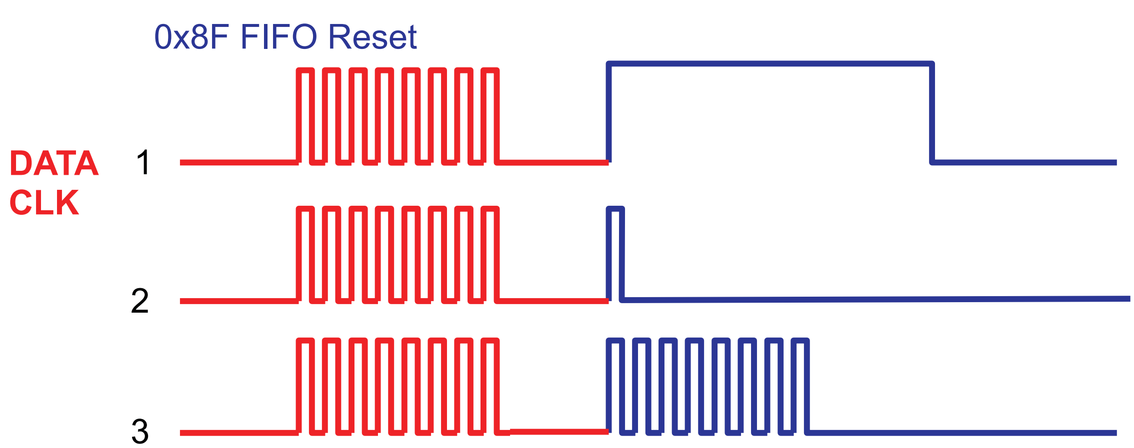
It is recommended to have this software fix written as part of a direct command function. An example of a direct command is the slot markers (EOF) for ISO 15693. This will not work in SPI mode.

This is solved by a software fix by implementing the direct command (for example transmit next slot") in one of two ways:

1. Have an additional SCLK cycle (low/high) before SS* goes high.
   or
2. Send a dummy TX write (8 SCLK cycles) before SS* goes high.

This is shown in the diagram below.

Any one of the three dummy clock options work.



The pseudo code shown below in the DirectCommand function implements a SCLK low/high/low transition.

```
SlaveSelectLOW; //Start SPI Mode
*pbuf = (0x80 | *pbuf); /* command */
*pbuf = (0x9f &*pbuf);     /* command code */
while (!(IFG2 & UCB0TXIFG));       // USCI_B0 TX buffer ready?
UCB0TXBUF = *pbuf;                  // Previous data to TX, RX

// while (!(IFG2 & UCB0RXIFG));
temp=UCB0RXBUF;

SPIStartCondition(); //SCLK Low/High cycle implemented
SlaveSelectHIGH; //Stop SPI Mode

P3SEL |=  BIT3;      //Revert Back
```

1.4 Initialization of Derivative Registers

Some of the registers (RX wait time, RX no response wait time) do not take default values when the Tag-it™ protocol is chosen in the ISO control register. This is solved by manually programming the timing related registers in the Initialization routine as shown in the pseudo code of the TIInventoryRequest function below.

```
//added code
buf[0] = RXNoResponseWaitTime;
buf[1] = 0x14;
buf[2] = ModulatorControl;
buf[3] = 0x21;
WriteSingle(buf, 4);
  //end added code
```

It is also recommended that the modulator and system clock register (register 0x09) be re-initialized when the inventory request (15693) or REQB (14443B) or REQA (14443A) is issued.

1.5 Transmitting One Byte Through the FIFO

When transmitting one byte to the TRF7960 using SPI with SS* mode, a special firmware fix is needed. This method involves splitting the writes into two operations as shown in the pseudo code below.

```
buf[0] = 0x8f;
buf[1] = 0x91;
buf[2] = 0x3d;
buf[3] = 0x00;
buf[4] = 0x10;
RAWwrite(&buf[0], 5);

buf[5] = 0x3F;
buf[6] = "one byte data to be transmitted";
buf[7] = 0x00;
RAWwrite(&buf[5], 3);
```

Each RAW Write function takes the SS low and high. Please refer to the TRF7960 firmware for definition of RAWwrite function. (file name parallel.c)