Application Note

Getting Started with ESPI Interface Using the Z8 Encore! XP® F1680

AN027301-0308

Abstract

This application note demonstrates how to use the Enhanced Serial Peripheral Interface (ESPI) in Master and Slave mode on Z8 Encore! XP F1680 microcontrollers. The software implementation provides several functions that enable users to initialize, send and receive a byte, and send and receive a block of up to 256 bytes.

Note: The source code (AN0273-SC01) associated with this application note is available for download from *www.zilog.com*.

Discussion

The Serial Peripheral Interface bus is a synchronous, serial communication link that operates in full duplex, meaning that a device transmits and receives data simultaneously. The devices communicate as a Master/Slave, where the Master initiates communication by selecting a Slave device with a hardware line and also provides the synchronous clock used to shift data bits in and out of the Slave. The signals required for communication are the Slave Select (SS), Master In Slave Out (MISO), Master Out Slave In (MOSI), and Serial Clock (SCK). The advantages of SPI over other communication protocols is that the addressing is performed in hardware with the SS line, making it faster to address a device, and that communication is full duplex, allowing for faster transfers of data.

SPI communication begins with the Master asserting the SS line. Depending upon the device, the SS line might be active high or active low. The Master must then wait at least one clock period before starting communication. Much like the active polarity of the SS line, the waiting period after SS activation varies from device to device. As an example, an analog-to-digital converter might require that the Master wait for a conversion to be completed after its SS line has been asserted. Next, the Master will begin shifting data out of the MOSI line, and it will shift data in on the MISO. Data is always transferred as full duplex, even when that data is not meaningful. As an example, for a Master to receive 24 bits of data from a Slave device it must also transmit 24 bits to the Slave device. See Figures 1 and 2 on page 2.

Phase 0 Timing

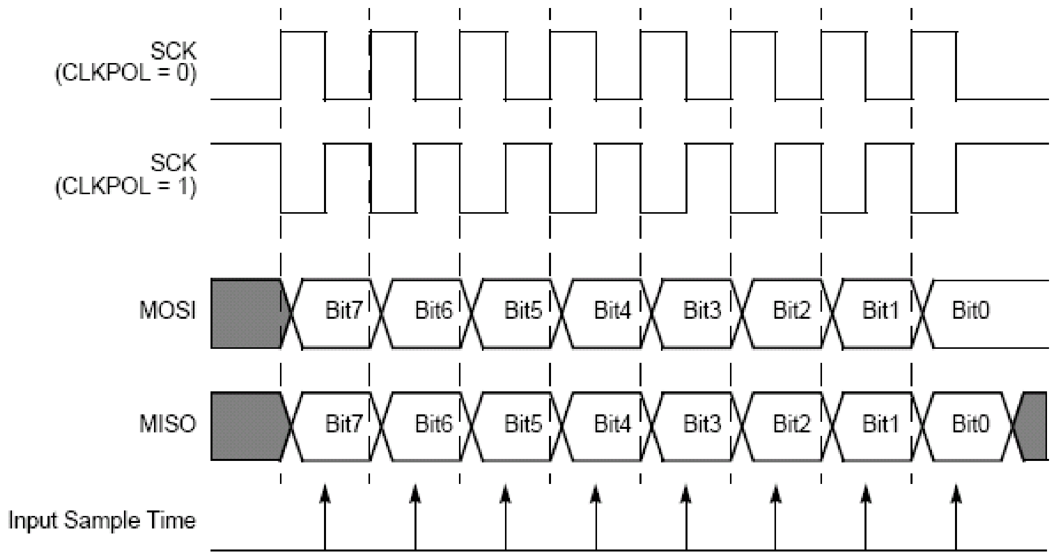

Figure 1. Phase 0 Timing

Phase 1 Timing

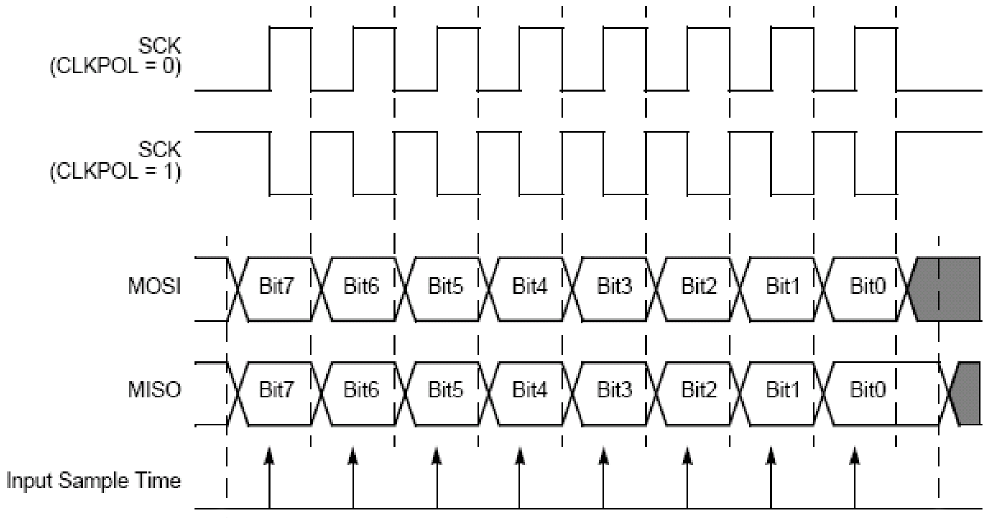

Figure 2. Phase 1 Timing

There is no standard for which clock edges are used for transmitting and receiving data, which leaves four possible modes of operation, depending on clock polarity and clock phasing. See the following table.

SPI Modes

MODE	SCK Polarity	SCK Phase	SCK Transmit Edge	SCK Receive Edge	SCK Idle State
0	0	0	Falling	Rising	Low
1	0	1	Rising	Falling	Low
2	1	0	Rising	Falling	High
3	1	1	Falling	Rising	High

Using Mode 1 as an example, the Master would idle the bus with the SCK line low. When the Master pushes the SCK line high, it will also place the most significant bit first on the MOSI line. At the same time, the Slave will place the most significant first on the MISO line. Next, the Master pulls the SCK line and reads the stable bit from the Slave on the MISO line. At the same time, the Slave reads a stable bit generated from the Master on the MOSI line. Communication is terminated when the SS line is deactivated, so it must remain active during the entire communication frame.

Testing the SPI Master/Slave

Sample Hardware

The Z8 Encore! XP F1680 microcontroller does have a hardware peripheral for SPI support. To test the software provided with this application note, two Z8 Encore! XP F1680 28-Pin Development Kits are required.

Hardware Preparation

One development kit will be used as a Master device, and the second development kit will be used as a Slave device. Along with a common ground, the SS, SCK, MOSI, and MISO lines will need to be jumpered together between the two devices.

1. Run a wire from the Master's J2-16 to the Slave's J2-16. This will serve as ground.
2. Run a wire from the Master's JP2-12 to the Slave's JP2-12. This will serve as the SCK line.
3. Run a wire from the Master's J2-11 to the Slave's J2-11. This will serve as the MISO line.
4. Run a wire from the Master's J2-13 to the Slave's J2-13. This will serve as the SS line
5. Run a wire from the Master's JP2-11 to the Slave's JP2-11. This will serve as the MOSI line.

Software APIs

The software APIs required for using the Master and Slave driver are located in the spi.c file. Three functions are available to drive the SPI hardware interface:

- SPI_Init
- SPI_SendReceive
- SPI_SendReceiveBlock

Definitions

- DATASIZE
  Determines the size of the input buffer. Because the Master determines how much data is exchanged on a transfer, DATASIZE needs to be at least as large as the number of bytes exchanged by the Master.

Globals

No global variable is defined in the SPI module. However, the main file contains one global, which is used for the input buffer, **DataIn[DATASIZE]**.

APIs

- `void SPI_Init(unsigned short freq, char  mode, char phase, char clkpol)`
  This function will initialize the SPI interface.
  The parameter freq is used to set the baud rate (value from 0 to 0xFFFF, 0 = slowest baud rate, 0x0001 = fastest baud rate), the parameter mode is used to determine if the SPI interface is configured as Master or as Slave (value = MASTER or SLAVE), the parameter phase is used to determine the phase (value = TRUE or FALSE), and the parameter clkpol is used to determine the clock polarity (value = TRUE or FALSE).
- `char SPI_SendReceive(char data, char mode)`
  This function will transmit the byte in data and will return the byte received. The parameter mode is used to determine if the SPI interface is configured as Master or as Slave (value = MASTER or SLAVE).
- `void SPI_SendReceiveBlock(char* p_out, char* p_in, unsigned char length, char mode)`
  This function will transmit length bytes (value from 0 to 0xFF, 0 = 256 bytes) pointed to by p_out and will store length bytes received in the buffer pointed to by p_in. The parameter mode is used to determine if the SPI interface is configured as Master or as Slave (value = MASTER or SLAVE).

Usage

In addition to calling the SPI_Init function, your software must include code to set the alternate function register in order to route the SPI pins to the IO pins. This code can be:

```
PCADDR = 0x02;      // Select alternate function

PCCTL = 0x36;       // Enable SPI interface
```

To perform a data transfer in your application, use the SPI_SendReceive function for a single byte or SPI_SendReceiveBlock with the parameter p_out pointing to the n data (n is passed in the parameter length) to send and the parameter p_in pointing to the buffer where the data received will be stored.

Software Preparation

1. Build the Master Zilog Developer Studio (ZDS) project and flash the code into the Master development board.
2. Build the Slave ZDS project and flash the code into the Slave development board.
3. Reset the Slave then the Master development boards by pressing SW1 or power cycling the boards.

Testing the Hardware SPI Master/Slave

Sample Software

Master

The Master software will send the string “Hello ZiLOG!” to the Slave; then it will wait until the switch SW2 is pressed. The Slave will respond with the same string to the Master and will light D3 if the string received is correct. If this response is not received by the Master, then the Master will light LED D4 to signal that an error has occurred. If no error occurs, the Master lights LED D2.

Then, with each press of switch SW2, the Master will increase the value of the byte sent to the Slave until the value reaches 128, and then the Master will start over again sending a value of 1. If the response sent by the slave is equal to 0xAA, the Master will light LED D2. If it is not, the Master will light LED D4.

At power-up, the Master will call SPI_Init to configure the SPI interface. Next, the Master calls IdlePorts() to configure switch SW2 and LEDs D2, D4 for the correct direction and configure the alternate function register. Next, the Master will send the string “Hello ZiLOG!” and verify that it is also received from the Slave. In this case, it will light the LED D2. If the string received is not correct, it will light the LED D4.

The Main Loop waits 62.5 mS to help debounce the switch, and then the switch is read. If the switch is closed, a byte is increased for use by the Slave’s PWM. Next, the Master starts the exchange by calling SPI_SendReceive. After SPI_SendReceive completes, the Master compares the data received to verify that the Slave responded 0xAA. If the Slave responded, then LED D4 is turned off, and LED D2 is turned on. If the Slave did not respond, then LED D2 is turned off, and LED D4 is turned on.

Slave

At power-up, the Slave will call SPI_Init to configure the SPI interface. Next, the Slave will call IdlePorts to initialize the PWM on LED D3 and configure the alternate register. Next, the Slave will call SPI_SendReceiveBlock to send the string “Hello ZiLOG!” and will turn on LED D3 if the same string is received from the Master.

In the Main loop, the Slave will wait for an exchange from the Master and perform a PWM on LED D3 based upon the value received by the Master. The Slave always responds to the Master with the value 0xAA.

Testing the Hardware SPI Master/Software SPI Slave
Testing the Software SPI Master/Hardware SPI Slave

It is possible to connect the F1680 development board to the F083A development board in order to have hardware SPI communicating with software SPI.

Hardware Preparation

One development kit will be used as a Master device, and the second development kit will be used as a Slave device. Along with a common ground, the SS, SCK, MOSI, and MISO lines will need to be jumpered together between the two devices.

1. Run a wire from the F1680's J2-16 to the F083A's J2-16. This will serve as ground.
2. Run a wire from the F1680's JP2-12 to the F083A's J2-9. This will serve as the SCK line.
3. Run a wire from the F1680's J2-11 to the F083A's J2-11. This will serve as the MISO line.
4. Run a wire from the F1680's J2-13 to the F083A's J2-13. This will serve as the SS line
5. Run a wire from the F1680's JP2-11 to the F083A's J2-15. This will serve as the MOSI line.

Software Preparation

The source code (AN0267-SC01) available from www.zilog.com will be used for the software SPI.

1. Build the Master ZDS project and flash the code into the Master development board.
2. Build the Slave ZDS project and flash the code into the Slave development board.
3. Reset the Slave then the Master development boards by pressing SW1 or power cycling the boards.

Summary

This application note demonstrates how to use the hardware Serial Peripheral Interface Master and Slave functionality on Z8 Encore! XP F1680 microcontrollers.

Appendix A—References

Topic	**Document Name**
Z8 Encore!® MCU	*Z8 Encore! XP F1680 Series Product Specification* (PS0250)
	Z8 Encore! MCU User Manual (UM0128)
ZDS II	*Zilog Developer Studio II—Z8 Encore! User Manual* (UM0130)
SPI Master	*Z8 Encore! XP F64xx Series Product Specification* (PS0199)
Development Kit	*Z8 Encore! XP F1680 Series Development Kit User Manual* (UM0203)
SPI Protocol	http://www.embedded.com/story/OEG20020124S0116

Warning: DO NOT USE IN LIFE SUPPORT