Application Note

Software SPI Emulation Using Z8 Encore! XP® Family of MCUs

AN025002-0608

Abstract

This application note discusses the software emulation of Serial Peripheral Interface (SPI) using Zilog's Z8 Encore! XP® Family of MCUs. The emulated SPI is used in applications where the MCUs do not have an on-chip hardware SPI block. This application note demonstrates the SPI emulation by interfacing Z8F042A MCU as a master device with temperature sensor DS1722 as a slave device.

The SPI emulation application discussed in this document provides application programming interface (APIs) to read and write on the slave device. This application works in master mode only. You can configure any General-Purpose Input/Output (GPIO) pin on the master device as Master-Out/Slave-In (MOSI), Master-In/Slave-Out (MISO), and Slave Select (SS) through the source code.

Note: *The source code is available as Z8 Encore! XP based Emulated Peripheral Code Library for ZDS II—Z8 Encore! at* http://www.zilog.com/library/.

Z8F042A MCU Overview

Zilog's Z8F042A device is a member of Z8 Encore! XP family of high-performance MCUs with extended peripherals. Demonstrating a high level of integration with overall low system cost, the Z8 Encore! XP MCU integrates a temperature sensor and transimpedance amplifier which process their outputs by an enhanced Sigma-Delta Analog-to-Digital Converter (ADC).

Key features of Z8F042A device include:

- 4 KB Flash Program Memory
- 128 B Non-Volatile Data Storage (NVDS)
- 1 KB register RAM
- Up to 8-channel 10-bit ADC
- On-chip Analog Comparator
- On-chip Temperature Sensor
- On-chip Transimpedance Amplifier
- Internal Precision Oscillator
- Two 16-bit timers with pulse width modulation (PWMs) and capture and compare
- Watchdog Timer (WDT)

Discussion

This section describes Serial Peripheral Interface and Temperature Sensor DS1722.

Serial Peripheral Interface

The SPI is a 4-wire serial communication interface for data transfer between master and slave devices. It consists of one master and one or more slave devices. The master device is defined as a device which generates the clock and slave device which receives the clock from master device for data transfer. This is a synchronous communication and data is transferred with respect to the clock. Figure 1 on page 2 displays the pin connections between master/slave devices in SPI interface.

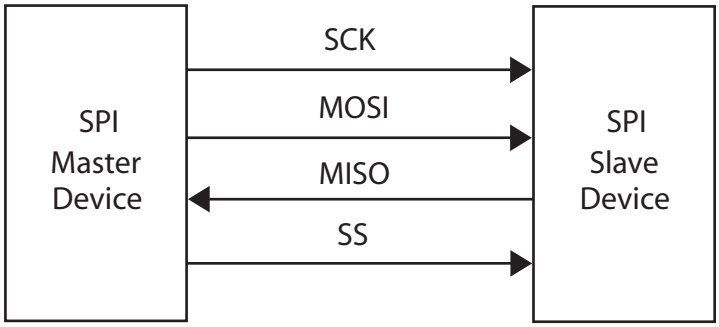


Figure 1. Pin Connections between Master/Slave Device in SPI Interface

Serial Clock Pin

The Serial Clock (SCK) pin generates the clock used for data transfer between master and slave devices.

Master-Out/Slave-In Pin

The MOSI pin transfers data from master device to the slave device in master mode and reads the data from slave device in slave mode.

Master-In/Slave-Out Pin

The MISO pin receives data from slave device in master mode and sends data to the slave device in slave mode.

Slave Select Pin

The SS pin selects and enables the SPI slave device to communicate with the master device.

Temperature Sensor DS1722

Temperature sensor DS1722 communicates either through a standard SPI interface or a 3-wire interface. It can measure temperature either in CONTINUOUS or in SINGLE-SHOT mode. It can also be configured for bit resolution required for temperature measurement, which ranges from 8-bit to 12-bit.

Configuration/Status Register Settings

To configure temperature sensor DS1722 for continuous measurement and 8-bit resolution mode, write configuration byte `0xE0` at address `0x80` of configuration register in DS1722. Figure 2 displays the bit configuration for Configuration/Status register.

1	1	1	0	0	0	0	0

MSB (Most significant bit) **LSB** (Least significant bit)

Figure 2. Configuration/Status Register Bit Configuration

Below are bit settings for Configuration/Status register:

- Bits 5, 6, and 7 are Read only bits.
- Bit 4 is used to configure DS1722 for CONTINUOUS or SINGLE-SHOT mode. By configuring Bit 4 to 0, DS1722 is set to CONTINUOUS mode.
- Bits 1, 2, and 3 are used to set the measurement resolution. By setting Bits 1, 2, and 3 to 0, temperature sensor DS1722 is set for 8-bit resolution.
- Bit 0 enables DS1722 to measure temperature. By default, Bit 0 is set to1.

After configuration, temperature sensor DS1722 starts measuring temperature. The temperature sensor DS1722 updates the temperature value at location 0x01 and 0x02. This measured temperature value is read from address 0x01 and 0x02 in DS1722. The emulated SPI software reads the measured temperature value from DS1722 and displays it on the HyperTerminal. For more details on Configuration Register Bit Settings, refer to Temperature Sensor DS1722 datasheet (DS1722) available for download at www.maxim-ic.com.

Theory of Operation

The Z8 Encore! XP® 28-pin Z8F042A MCU is configured for external crystal (20 MHz) as the clock source to achieve high data transfer rates. The emulated SPI code used in this application supports basic SPI communication with clock polarity (CPOL) set to 0 or 1 and clock phase (CPHA) set to 1 (see Table 1).

Before performing any data transfer on the SPI interface, the SPI slave device must be enabled. The slave device is enabled by pulling the chip SS line either to logic Low or logic High.

While writing on the SPI interface, 8 bits are transferred from master to slave and while reading 8 bits are read from slave by master. Depending on the polarity and phase of SCK, there are four possible combinations of data transfer between master and slave device. Table 1 lists these four data transfer combinations.

Table 1. Data Transfer Combinations

Clock Polarity (CPOL)	Clock Phase (CPHA)	Clock State (Idle)
0	0	Low
1	0	High
0	1	Low
1	1	High

Figure 3 displays the data transfer (1 byte) on SPI interface with CPOL set to 0 and 1 and CPHA set to 1.

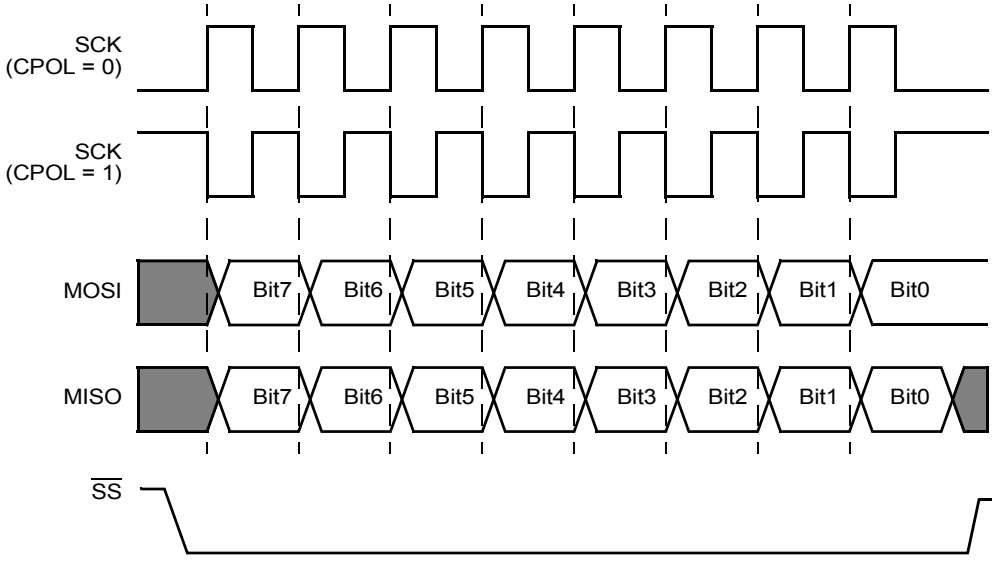


Figure 3. SPI Data Flow Diagram with CPHA Set to 1

In Figure 3, CPOL = 0 indicates that initial state or idle state of SPI clock line is logic Low. CPOL = 1 indicates that initial state or idle state of SPI clock line is logic High. CPHA = 1 indicates that slave places data on rising edge of the SPI clock. During SPI communication, the data is transferred from MSB to LSB. First, the master device transfers address or data on the MOSI line. In response to that data transfer, slave device places data on MISO line.

Developing Software Emulated SPI Using Z8F042A MCU

In this application, a Z8 Encore! XP® 28-pin Z8F042A MCU is configured as master device and temperature sensor DS1722 as a slave device.

This section describes the hardware architecture and software implementation of the emulated SPI using Z8F042A MCU.

Hardware Architecture

The hardware architecture for demonstrating emulated SPI uses Z8 Encore! XP Development Kit (Z8F04A28100KIT) and temperature sensor DS1722. The four signals (pins) MOSI/MISO/SS/SCK of temperature sensor DS1722 are connected to JP2 connector on the development board.

The power supply required for temperature sensor DS1722 is provided from the Z8 Encore! XP Development Kit. Figure 4 displays the hardware architecture for emulated SPI demonstration.

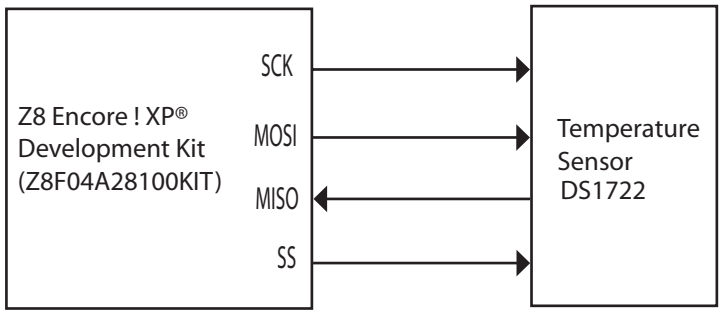


Figure 4. Emulated SPI Demonstration Block Diagram

Software Implementation

To initiate Read or Write process, chip select signal is applied to the slave device. In this application, chip select signal is made active High to enable the slave. The Timer1 peripheral is configured for continuous mode of operation and generates an interrupt on every re-load of counter. Whenever a Timer1 counter register reaches the count loaded into re-load register, an interrupt is generated and Timer1 output (PA7) is toggled. This generates the SPI clock. Initial status of T1OUT pin is Low. The data is transferred or received on rising edge of port pin T1OUT.

This modular software allows you to use the associated APIs directly in your application. The Z8 Encore! XP based Emulated Peripheral Code Library v1.0.0 consists of the following APIs:

- Init_Master_SPI
- Write_SPI
- Read_SPI

Init_Master_SPI

The `Init_Master_SPI()` API configures the Z8F042A MCU's port pin to function as SPI interface.

The Init_Master_SPI() API must be called before performing Read or Write operation on SPI interface. This API takes configuration input from a header file. You can select any available port pin and configure as MOSI/MISO/SS in this header file. The MOSI pin, SCK pin, and SS pin is configured as output and MISO pin is configured as input pin. This API also sets the initial condition of these port pins. It configures Timer1 in Continuous and interrupt mode to generate the clock.

Write_SPI

The Write_SPI API performs write operation on SPI interface.

The Write_SPI() API takes address and data to be written on the address as input parameters. It asserts chip select signal, writes address, writes data and then deasserts the chip select signal.

To write the data on SPI slave, first send the write address and then data to be written on the MOSI line. In this application, when data is required to be transmitted on MOSI line, MSB of the data byte is checked. If MSB bit is set, then port pin MOSI is kept High else it is made Low. After putting the status of MSB bit on MOSI line, data byte is shifted to left by 1 bit position. This process is repeated for 8 bit times.

Read_SPI

The Read_SPI API performs read operation on SPI interface.

The Read_SPI() API takes address as input and reads the data from that slave address. It asserts chip select signal, writes address onto MOSI line, writes dummy byte onto MOSI line, and reads the MISO line while writing dummy byte. After reading, this API deasserts the chip select signal. A dummy byte (0xFF) is shifted on the MOSI line so that the data is shifted by slave on MISO line. In this application, when data is required to be read on MISO line, status of port pin MISO is read.

If the port pin status is High, '1' is shifted left into a receive byte and if it is Low, '0' is shifted left into receive byte. This procedure is repeated for 8 bit times.

In SPI interface, to write the data onto slave, call Write_SPI API with appropriate address/data to be written. To read the data from slave, call Read_SPI API. This API returns data read from SPI slave. For emulated SPI software flow, see Appendix B—Flowcharts on page 11.

Testing Software Emulated SPI Using Z8F042A MCU

This section details the procedure designed to demonstrate the emulated SPI developed using Z8F042A MCU.

Hardware Required

The hardware required for the SPI emulation includes:

- Z8 Encore! XP® Development Kit (Z8F04A28100KIT)
- Temperature Sensor DS1722
- RS232-9 pin male/female cable

System Configuration

The system configuration for the SPI emulation includes:

- Windows XP with 2 COM port support
- HyperTerminal application
- Zilog Developer Studio (ZDS II)—Z8 Encore!
- Z8 Encore! XP based Emulated Peripheral Code Library

Equipment Used

The equipments used for demonstrating SPI emulation includes:

- Mixed Signal Oscilloscope (MSO)
- Logic Analyzer

Test Setup

The SPI temperature sensor DS1722 is soldered to the general purpose area on the Z8 Encore! XP® Development Kit (Z8F04A28100KIT). All port pins of Z8F042A MCU are available on connectors JP1 and JP2 of the Development Kit.

The port pin connections are given below:

- PC5 is configured as MISO
- PC6 is configured as MOSI
- PC7 is configured as SS
- T1OUT pin is configured as an alternate function to generate the clock

The MOSI and SS pins are configured as output and MISO pin is configured as input. You can configure any port pin to function as MOSI/MISO/SS pins except few (for example, debug, X1, X2, Vcc, etc.). Table 2 provides the actual pin numbers of Development Kit connected to temperature sensor DS1722.

Table 2. Z8F042A MCU and Temperature Sensor DS1722 Pin Connections

Z8F042A MCU (Default Configuration)	JP2 Connector (on Development Kit)	Temperature Sensor DS1722
PC5 (MISO)	Pin 12	MISO (Pin 5)
PC6 (MOSI)	Pin 14	MOSI (Pin 6)
PC7 (SS)	Pin 13	SS (Pin 2)
PA7 (SCK)	Pin 21	SCK (Pin 3)

Figure 5 displays the pin connections between Z8 Encore! XP Development Kit and temperature sensor DS1722.

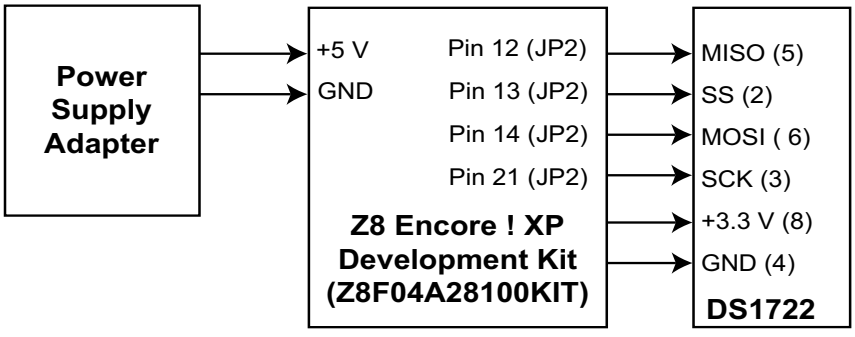


Figure 5. Development Kit and Temperature Sensor DS1722 Pin Connections

To print the temperature value read from SPI temperature sensor, connect the Z8 Encore! XP Development Kit (Z8F04A28100KIT) with a 9-pin standard serial cable to a standard PC running Windows XP. The MSO is connected to the Z8 Encore! XP Development Kit (Z8F04A28100KIT) to observe the SPI waveforms.

HyperTerminal Settings

The HyperTerminal settings are provided below:

1. Launch the HyperTerminal application
2. Select the communication port with following configurations:
   (a) Baud rate set to 57600 bps
   (b) 8-N-1 with no flow control

Test Procedure

Follow the steps below to execute the emulated SPI application:

1. Install the Z8 Encore! XP based Emulated Peripheral Code Library available for download at http://www.zilog.com/library/.
2. Powerup the Development Kit (Z8F04A28100KIT) using 5 V adapter.
3. Launch ZDS II by navigating from the **Start** menu to **Programs →ZDS II—Z8 Encore!**. (For version compatibility of ZDS II, refer to the source code available in the library).
4. Download the library to the Flash Memory using ZDS II.
5. Run the program using ZDS II.

The room temperature measured by DS1722 is displayed on HyperTerminal in hexadecimal format. The SPI waveforms are displayed on oscilloscope by connecting the probes on the SPI signals.

Test Results

Figure 6 displays the emulated SPI waveforms captured using Logic Analyzer. When SS is High, slave is active and when SS is Low slave becomes inactive. The data is transferred during high period of clock. In this instance, first address `0x02` is sent during first 8 clock cycles and then during next 8 clock cycles slave device sends data onto the MISO line. This data is read by the master device during every high period of clock cycles. Cursor 1 and Cursor 2 in Figure 6 displays the clock period.

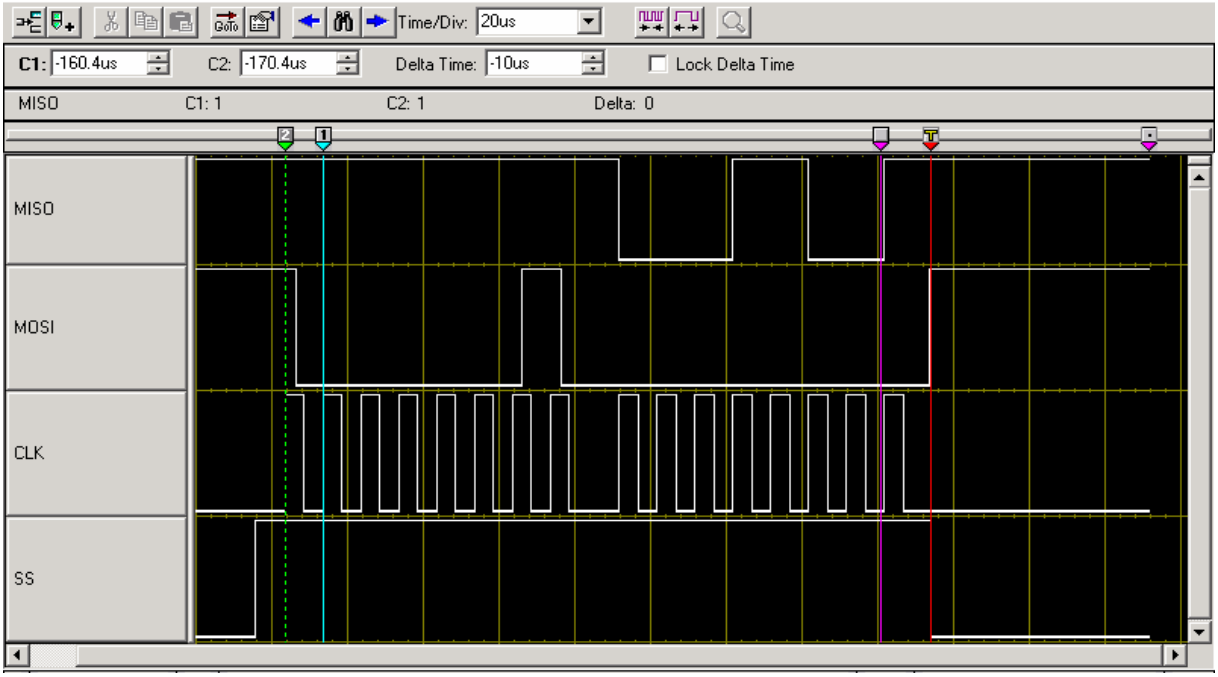


Figure 6. Emulated SPI Waveforms

Table 3 provides the code size and maximum data transfer rate.

Table 3. Code Size and Maximum Data Transfer Rate

Code Size		
Emulated SPI Application	**SPI APIs**	**Supporting Code**
900 Bytes	600 Bytes	300 Bytes
Data Transfer Rate		
Clock Source	**Value**	**Maximum Speed**
Internal Precision Oscillator	5.5296 MHz	27 Kbps
External Crystal	20.00 MHz	100 Kbps

Summary

This application note demonstrates the software Emulation of SPI using Z8 Encore! XP® (Z8F042A) MCU. As seen from the results, a maximum speed of 100 Kbps is achieved using an external crystal with a frequency of 20 MHz.

It demonstrates transfer of data bytes on SPI interface with CPOL set to 0 or 1 and CPHA set to 1. The code for software emulated SPI is developed in 'C' language. The emulated SPI APIs takes 600 Bytes of code. Total application code size is 900 Bytes, which includes UART code to print the data read from the slave device on the HyperTerminal. This application code is pin configurable for Z8 Encore! XP (Z8F042A) MCU and can be done with minimum changes.

References

The documents associated with Z8 Encore! XP , eZ8 CPU, ZDS, and Temperature Sensor DS1722 are provided below:

- Z8 Encore! XP® 4K Microcontrollers Product Specification (PS0228)
- Z8 Encore! XP® F042A Series Development Kit User Manual (UM0166)
- eZ8™ CPU User Manual (UM0128)
- ZDS II Zilog Developer Studio II — Z8 Encore!® User Manual (UM0130)
- Temperature Sensor DS1722 datasheet DS1722 (www.maxim-ic.com)

Appendix A—Glossary

Table 4 lists the terms and abbreviations used in this Application Note.

Table 4. Glossary

Abbreviation	Expansion
SPI	Serial Peripheral Interface
MOSI	Master-Out/Slave-In
MISO	Master-In/Slave-Out
SS	Slave Select
SCK	Serial Clock
CPOL	Clock Polarity
CPHA	Clock Phase
MSO	Mixed Signal Oscilloscope
ZDS	Zilog Developer Studio
NVDS	Non-volatile Data Storage

Appendix B—Flowcharts

Figure 7 displays the basic software flow of Emulated SPI.

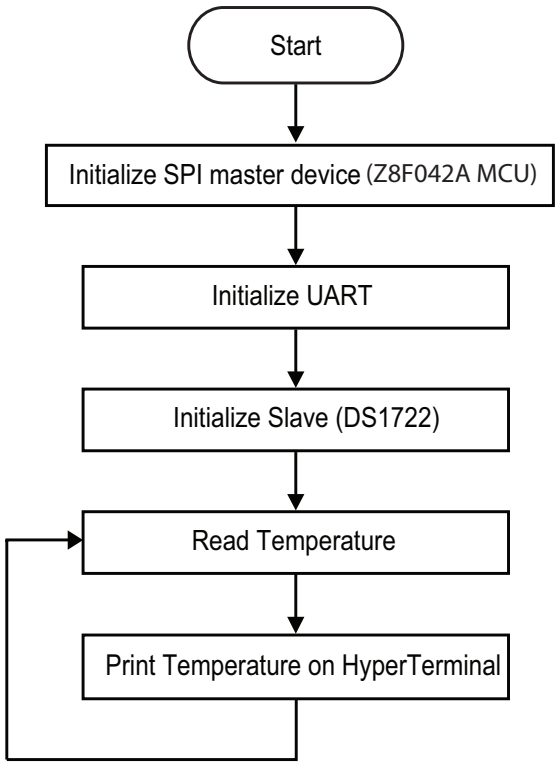


Figure 7. Emulated SPI Software Flow

Figure 8 displays the flowchart for performing Write operation on SPI.

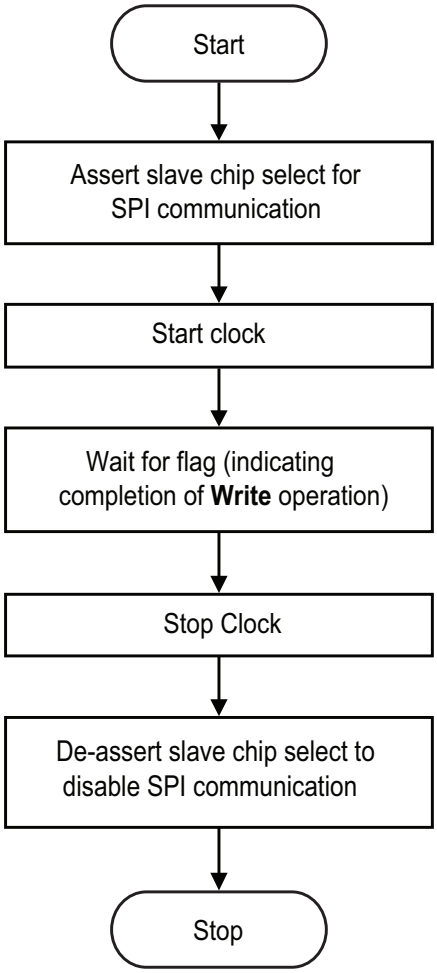


Figure 8. Write Operation

Figure 9 displays the flowchart for Read operation on SPI.

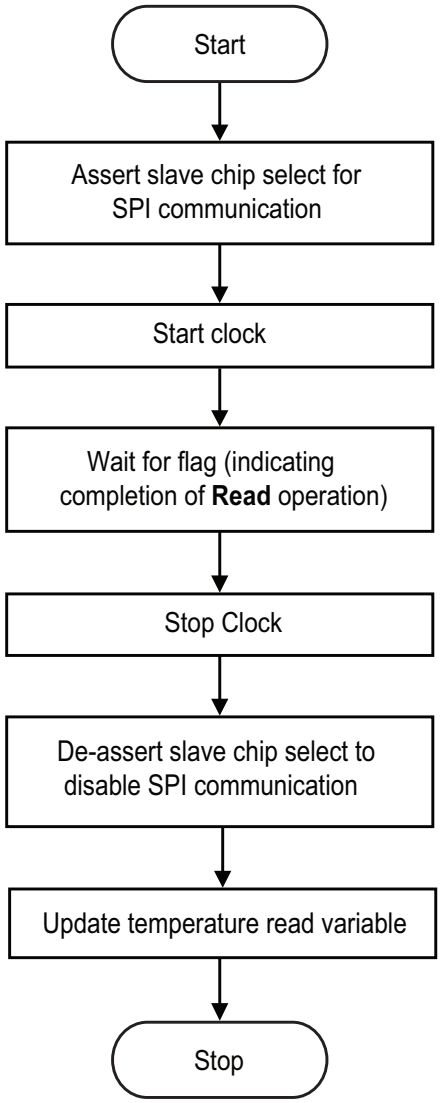
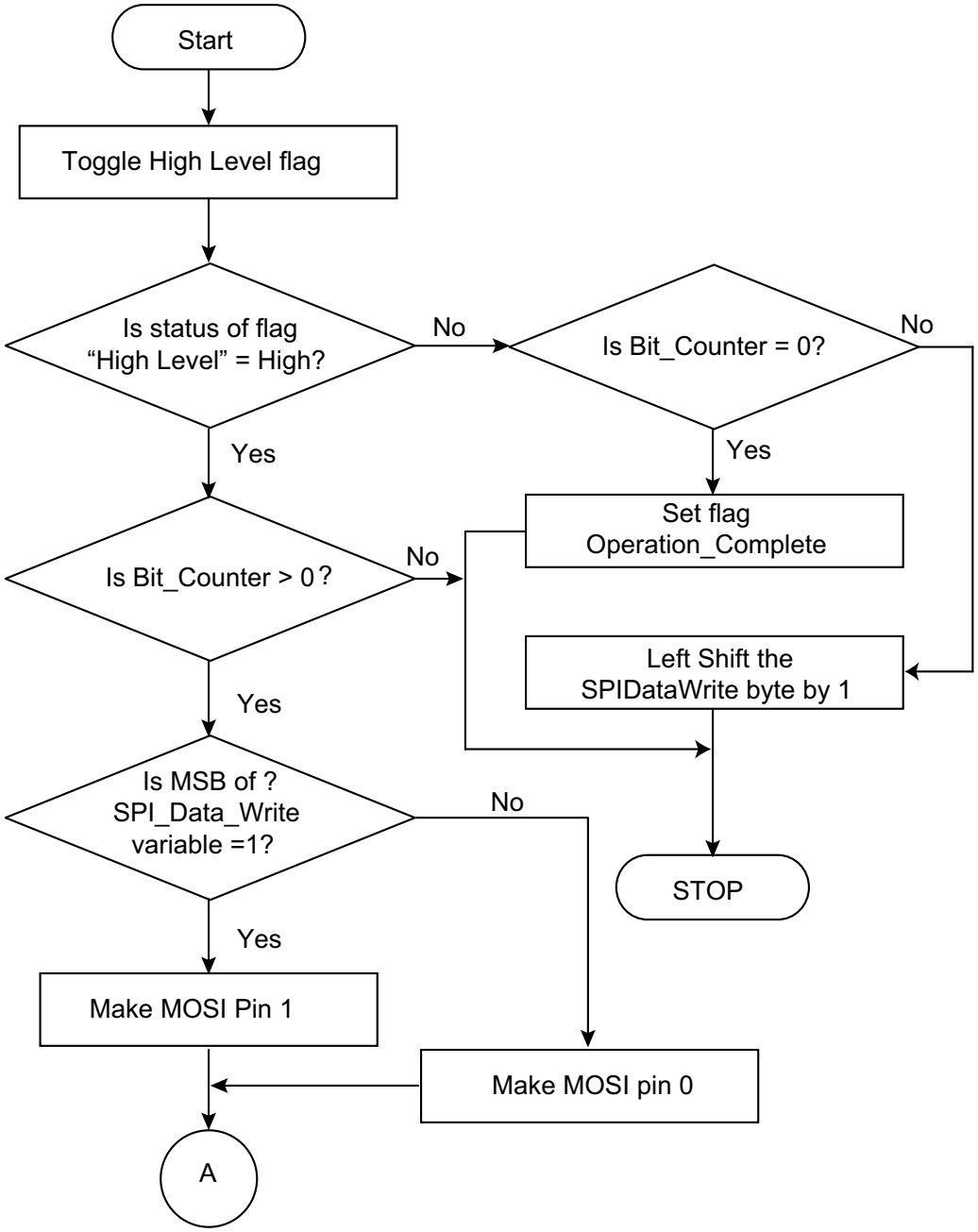


Figure 9. Read Operation

Figure 10 displays the Timer ISR flowchart.



Continued

Continued

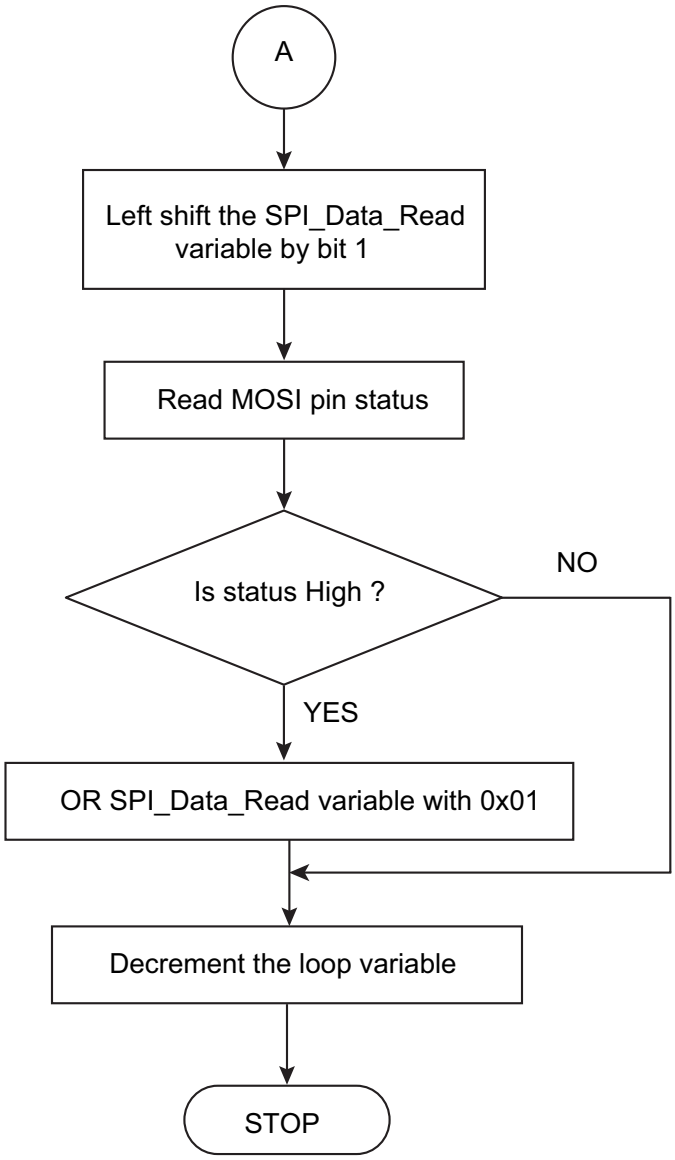


Figure 10. Timer ISR Flowchart

Warning: DO NOT USE IN LIFE SUPPORT

LIFE SUPPORT POLICY

ZILOG'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS PRIOR WRITTEN APPROVAL OF THE PRESIDENT AND GENERAL COUNSEL OF ZILOG CORPORATION.

As used herein

Life support devices or systems are devices which (a) are intended for surgical implant into the body, or (b) support or sustain life and whose failure to perform when properly used in accordance with instructions for use provided in the labeling can be reasonably expected to result in a significant injury to the user. A critical component is any component in a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system or to affect its safety or effectiveness.

Document Disclaimer