Serial-to-Peripheral Interface (SPI)

1 Project Objective

The objective of this project is to introduce you to the Serial-to-Peripheral Interface (SPI) serial bus standard by using it to interface to serial Flash memory.

2 Project Requirements

The SPI bus is a synchronous serial communications bus that uses four wires for communication between a "master" device and a "slave" device. This standard operates in full-duplex mode and can accommodate multiple slaves. The four signal wires are

- Master Out Slave In (MOSI) is sent by the master and received by the slave.
- Master In Slave Out (MISO) is sent by the slave and received by the master.
- Serial Clock (SCLK) is generated by the master for synchronizing data transfers.
- Slave Select (SS) is generated by the master to select an individual slave.

Other names are sometimes assigned to these signals such as Serial Data In (SDI) for MOSI and Serial Data Out (SDO) for MISO. Figure 1 shows a master with a single slave.

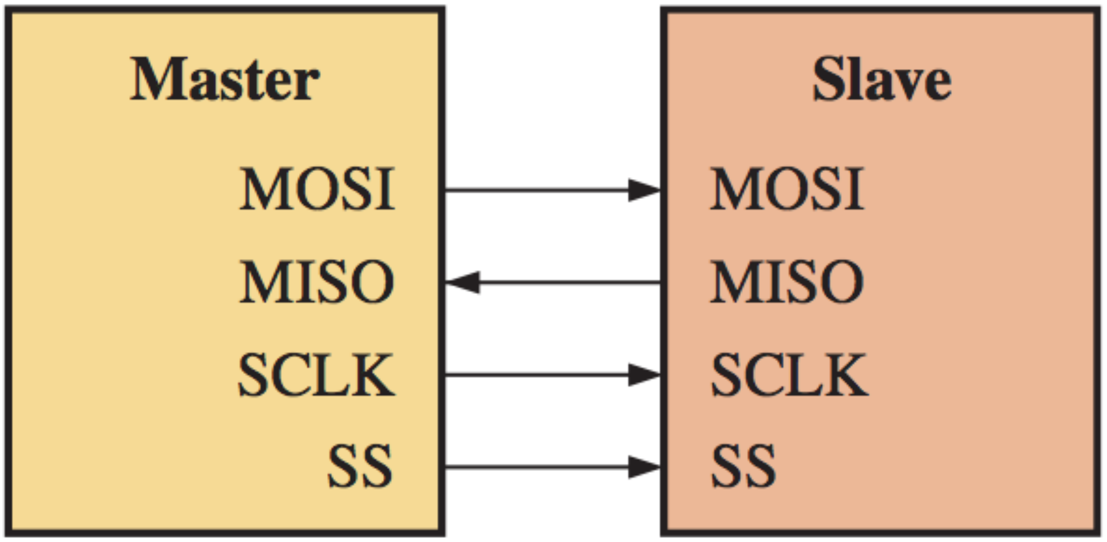

Figure 1: Basic Master/Slave SPI Configuration.

For this project you will design C code that implements the functionality of the four SPI signals in software to communicate with serial Flash memory contained on an external General-Purpose Input/Output/Memory (GPIO/MEM) Printed Circuit

Board (PCB) that connects to the Launchpad using the posts that are connected to the port pins (see Figure 2). On the GPIO/MEM, most of the port pins are connected to Test Points (TPs) that provide relatively easy access for monitoring a given port pin with an oscilloscope probe. In addition, the board contains two serial Flash memory components (U2 and U3 – see Figure 2), each of which can be accessed when the jumpers, J1 and J2, are in place.

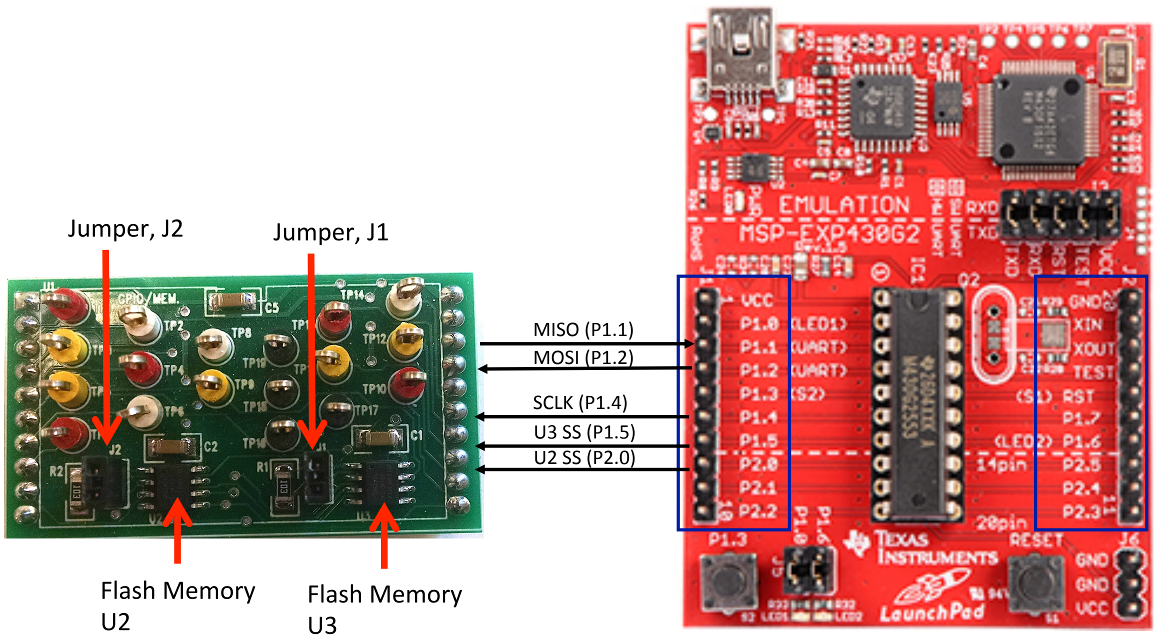

Figure 2: Launchpad and External GPIO/Memory Board.

2.1 Milestone 1

For the 1st milestone, you will develop a function to send a byte to the serial Flash memory on the GPIO/MEM board using the SPI signals, as shown in **Error! Reference source not found.**.

Note that the signals and their associated port pins shown in Figure 2 are not intended to be comprehensive, and as such there are other signals and corresponding port pins that you will need to configure and use in order to be able to access the serial Flash memory. You will need to consult the GPIO/MEM schematic and the datasheet for the serial Flash memory for a complete set of the signals and their associated port pins.

2.2 Milestone 2

Once you have verified a successful send over SPI, your next milestone is to develop a function to receive information from the serial Flash memory and verify its correct operation. A relatively simple way to verify a correctly functioning receive function is to read the two bytes in each Flash memory component that contain the Manufacturer's ID and the Device ID – 0xBF48. See the Flash memory data sheet for more details.

2.3 Final Verification

Finally, you will verify the remaining operations for the serial Flash. To do so, you will be provided with a software testbench, along with an include file for the serial flash that defines the function prototypes for the remaining functionality. You are required to write C code for all of the remaining functions defined in the include file, using the software testbench provided to verify correct functionality.

3 Project Grading

Requirement	**Point Value**
Milestone 1: Correctly functioning SPISendByte() function	5
Milestone 1: Verification	5
Milestone 2: Correctly functioning SPIReceiveByte() function	10
Milestone 2: Verification	5
Final Verification	75
Total points	**100**
- ReadFlashMemoryStatusRegister()	5
- WriteFlashMemoryStatusRegister()	10
- ReadFlashMemory()	5
- SetBlockProtection()	5
- ByteProgramFlashMemory()	10
- AAIProgramFlashMemory()	20
- ChipEraseFlashMemory()	5
- SectorBlockEraseFlashMemory()	10
- FlashMemoryBusy()	5

Note that if you are unable to verify all of the Flash memory operations, then individual functions will be graded according to the point values show (assuming you provide verification of correct functionality).