Lab 9: Serial Peripheral Interface

ECE 3720

Preview

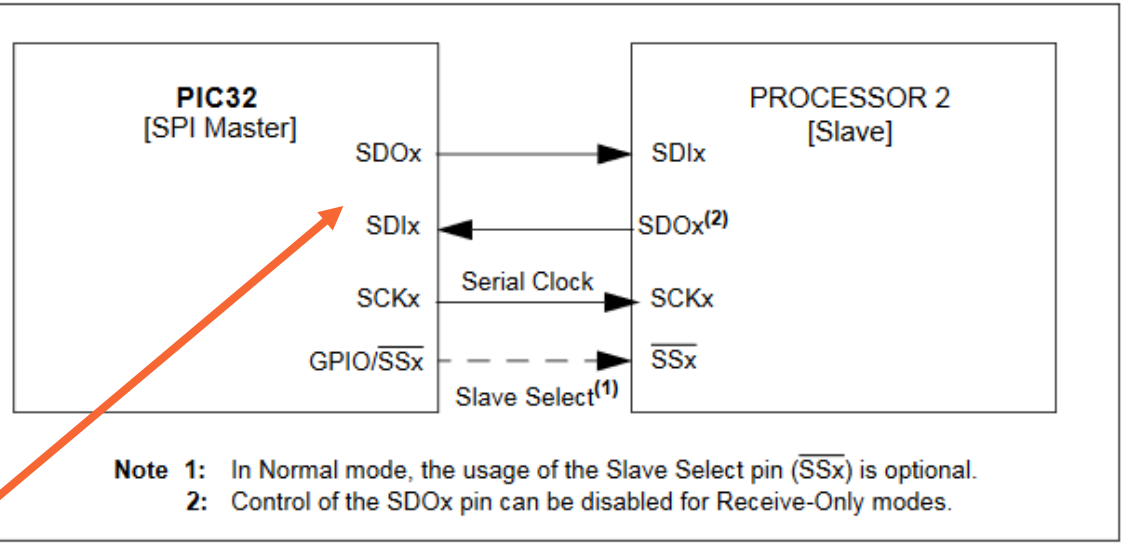
The PIC32's SPI module will be used to send data serially to the AD2, which will display its value on a logic analyzer. Each time a button is pressed, the MC will send a new value to the AD2.

Serial Peripheral Interface (SPI)

- SPI is a communication protocol often used for short-distance communication in embedded systems.
  - Serial communication sends data one bit at a time over a single channel.
- Master-slave architecture
  - Each can send data to the other.
  - Master provides the clock and slave select signals (slave select tells that particular device to listen).
  - PIC32 can operate as master or slave. We'll use it in master mode.

The terms **MOSI** (Master Out, Slave In) and **MISO** (Master In, Slave Out) are often used to refer to the two data channels.

Figure 23-2: Typical SPI Master-to-Slave Device Connection Diagram



Note 1: In Normal mode, the usage of the Slave Select pin ($\overline{SSx}$) is optional.
2: Control of the SDOx pin can be disabled for Receive-Only modes.

PIC32 FRM – Section 23. SPI, pg. 4

The SPIx serial interface consists of four pins:

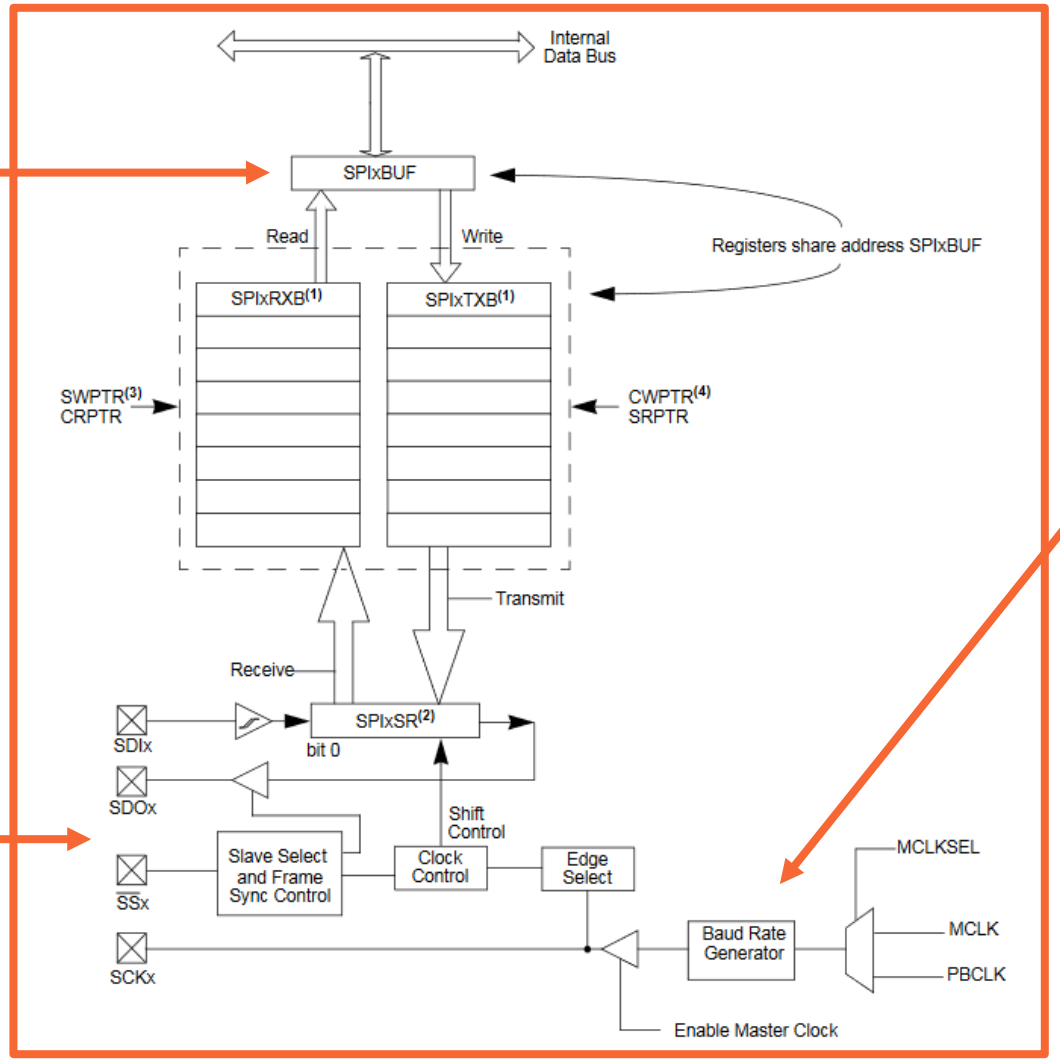
- SDIx: Serial Data Input
- SDOx: Serial Data Output
- SCKx: Shift Clock Input or Output
- $\overline{SSx}$: Active-Low Slave Select or Frame Synchronization I/O Pulse

PIC32 FRM – Section 23. SPI, pg. 2

SPI Module Diagram

Write transmit data and read received data from SPIxBUF

- Data propagates automatically to SPIxTXB (transmit) or from SPIxRXB (receive)
- **You will only write to SPIxBUF**



Output pins

- SDOx and SSx must be mapped with PPS
- SCKx is hard mapped to a specific pin

Baud Rate Generator

- Baud rate refers to rate of data transfer (bits/s)
- Here, the BRG divides the clock to achieve desired rate

PIC32 FRM – Section 23. SPI, pg. 3

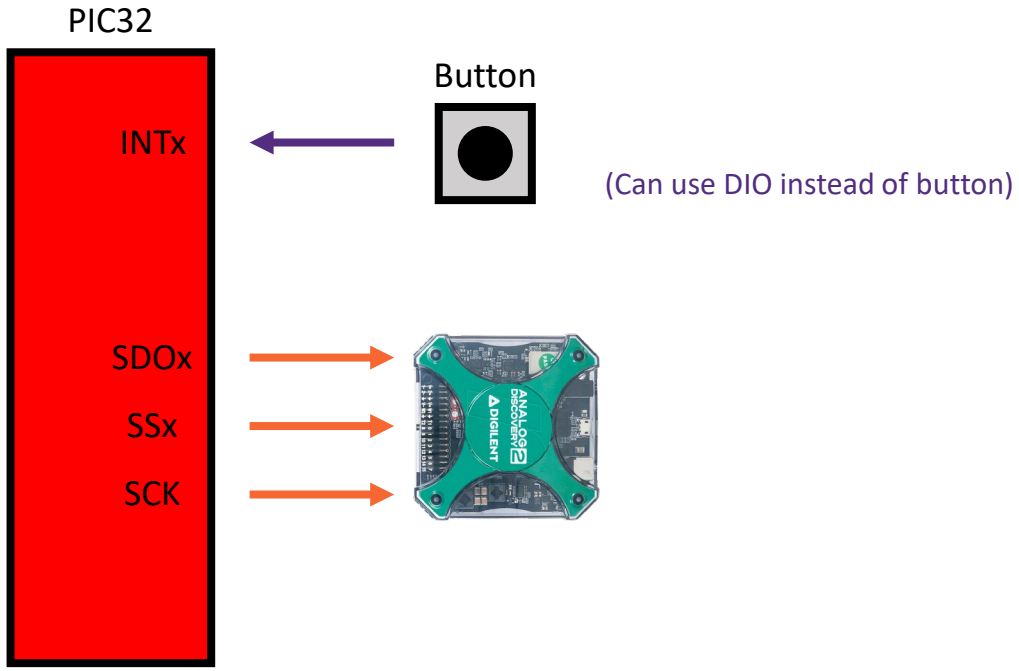
Using SPI

- The document ***Section 23 – Serial Peripheral Interface*** is available on Canvas in the Lab 9 module.
  - Contains all the information about the PIC32's SPI capabilities
- Follow the steps in **Section 23.3.3.1 Master Mode Operation.**
  - This covers the majority of what you need to do in your code.
  - Study pages 18-20 for a better understanding.
  - See the notes at the end of these slides for details on what values you will set.
- Registers of interest
  - SPIxBUF //write or read data
  - SPIxCON //configure SPI module (pg. 8)
  - SPIxSTAT //SPI module status (pg. 13)
  - SPIBRG //divisor for baud rate generator

Lab Goals

- Use the PIC32 in SPI master mode to send the values from the given array to the AD2.
- Connect the AD2 inputs as shown in the diagram.
- Set up an external interrupt. Each time it's triggered, the next value in the array should be written to SPIxBUF and appear on the LEDs.

Simple Diagram

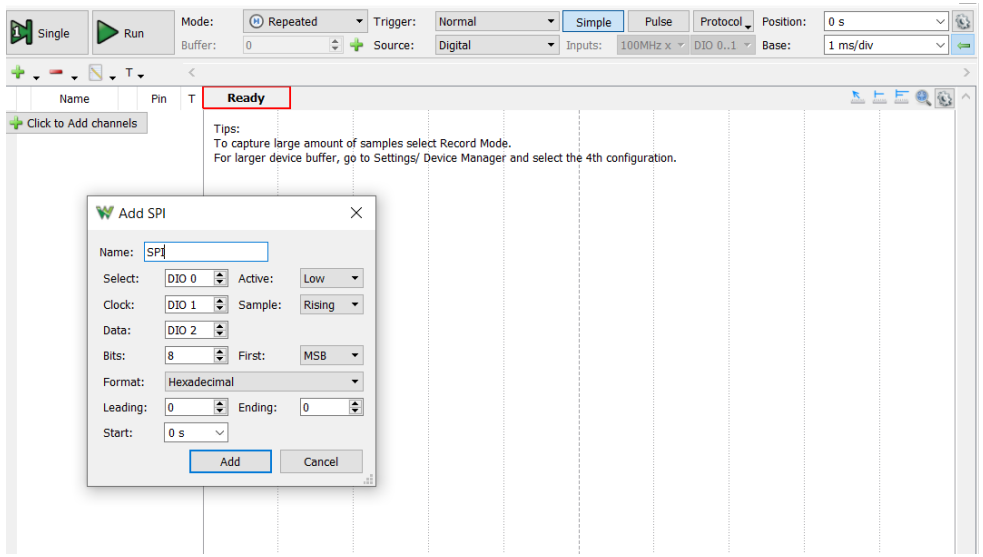


Array of values to display:

```
char spiChars[18]  = {0, 1, 2, 4, 8, 16, 32, 64, 128, 255, 254, 253, 251, 247, 239, 223, 191, 127};
```

Logic Analyzer

- The logic analyzer allows you to see output of a specific protocol.
- We will be using the SPI protocol.
- Select what DIO pins you want to use, and how you would like to see the data.
- Set the base to >100 ms/div



Notes

- The SPI module can transmit 8-, 16-, or 32-bit bit data. What data width will be needed for this lab, and how is that mode selected? (see section 23.3.1)
- The following is a checklist of values to set (located in SPI1CON, unless otherwise specified)
  - FRMPOL = 1 // Make SSx active-high to use with SRCLR (so shift register gets cleared when data is NOT transmitted)
  - IEC1bits.SPIRX = 0
  - IEC1bits.SPITX = 0 // Disable SPI interrupts
  - ON = 0 // Disable SPI module during setup
  - SPI1BUF
  - ENHBUF = 0 // Don’t want enhanced buffer mode
  - CKP, CKE // See figure 23-9 in Section 23 document (pg. 20)
  - SPIBRG = 2000 // Should result in baud rate slow enough to observe transmission with oscilloscope
  - SPIROV // In SPI1STAT
  - MSTEN, MSSEN
  - ON = 1 // Enable SPI module after setup is complete