EE128: Data Acquisition, Instrumentation and Process Control
University of California, Riverside

Lab 7: Serial Communication (SPI and I2C)

Spring 2019

Objective

To get familiar with SPI and I2C interfacing and programming.

References

- Dragon12-JR Trainer User's Manual
- Freescale MC9S12DG256 Device User Guide
- Freescale HCS12 Core User Guide
- Reference Guide for D–Bug12 Version 4.x.x
- MCP23S08 Datasheet
- 24LC01B Datasheet
- HCS12 SPI Module Manual

Equipment

- PC running MS Windows
- Digital Multi-Meter (DMM)
- Power supply (+5V, +9V)
- Dragon12-JR 9S12DG256 EVB
- Breadboard (you need to bring one)

Parts

- Microchip MCP23S08 SPI I/O expander chip
- Microchip 24LC01B I2C 1 kB EEPROM
- 4 button DIP switch
- 8-bit LED bar
- Resistor Networks 7 x 4.7kΩ and 5 x 4.7kΩ
- 2 x 2.2 kΩ resistors

Software

- Freescale CodeWarrior for HC12 v5.1
- RealTerm

PART 1. SPI Based I/O Extender MCP23S08 – Interfacing and Programming

Serial Peripheral Interface (SPI) is a synchronous serial protocol, widely used for interfacing peripheral chips to a microcontroller. Under SPI, a receiving device (slave) can be implemented as a simple shift register as shown in **Figure 1**.

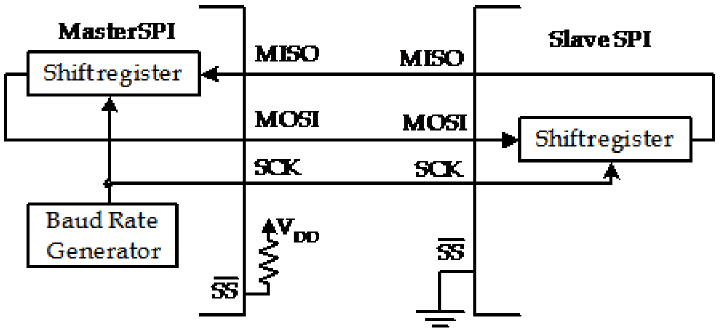


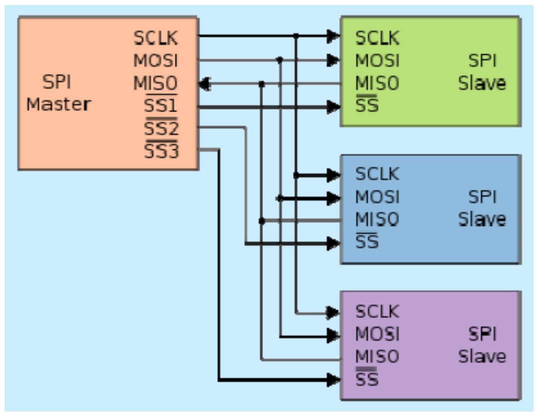
Figure 1. Serial Peripheral Interface (SPI)

SPI classifies devices into the master or slaves, and uses four wires to carry out the task of data communication:

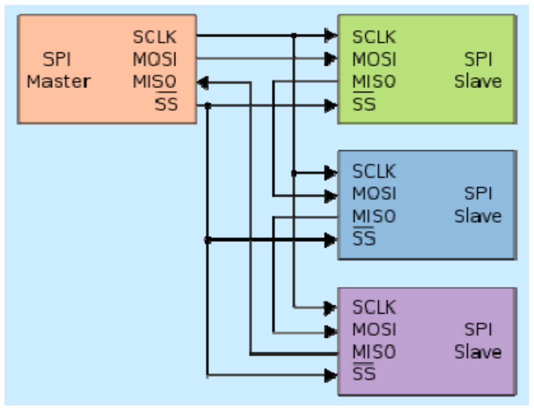
- MOSI: master out slave in
- MISO: master in slave out
- SCK: serial clock
- SS: slave select

An SPI data transfer is initiated by the master device. The master is responsible for generating the SCK signal to synchronize the data transfer.

SPI Topologies and Programming



(a) independent slaves



(b) daisy chain

Figure 2. SPI Topologies

- The operating parameters of each SPI module are controlled via two control registers:

  - SPIxCR1: (x = 0, 1, or 2)
  - SPIxCR2

- The baud rate of SPI transfer is controlled by the SPIxBR register.
- The operation status of the SPI operation is recorded in the SPIxSR register.

Microchip MCP23S08 8-Bit I/O Expander

General Purpose I/O (GPIO) expanders allow easy expansion of I/O using standard serial interfaces. The GPIO expanders can increase the availability of I/O pins on an MCU or provide remote I/O access using a serial interface.

The I/O port of MCP23S08 is highly configurable with flexibility. The port can either drive logic levels on the pin, or read logic levels from the pad. The level on the pad can be read at any time, regardless if the pin is configured as an input or output.

The IODIR register of MC23S08 controls the direction of the pins (input or output). More specifically, the IODIR register simply enables/disables the output driver. When the driver is activated (IODIR = 0), the pad is driven to the state in the latch register (OLAT). When deactivated (IODIR = 1), the driver is high impedance. The I/O port has multiple, individual configurations.

Each pin can …

- be configured as an input. The output driver is disabled (high impedance)
- be configured as an output. The output driver is enabled, and the value in the latch is driven on the pin.
- enable a weak pull-up resistor
- emulate an open-drain configuration. This is accomplished by clearing the output latch (OLAT) bit to zero and using the direction register (IODIR) to set the level on the pin. A pull-up resistor is required to pull the pin to voltage when the pin is an input
  - To drive a 0: configure the pin as an output (IODIR = 0), so the port drives whatever is in OLAT (logic 0 in this case)
  - To float a 1: set the pin as an input (IODIR = 1). The output driver is disabled and the pull-up resistor pulls the pin to a logic 1

For further information, refer to the MCP23S08 datasheet (see the resources tab on Piazza).

<u>Lab Procedure</u>

In this lab experiment, we will obtain a 3-bit information from a switch and display this info by using the lower 3 LEDs of an LED bar display after receiving an interrupt signal generated by the change of the switch status. The I/O expander will operate in the slave mode at all times.

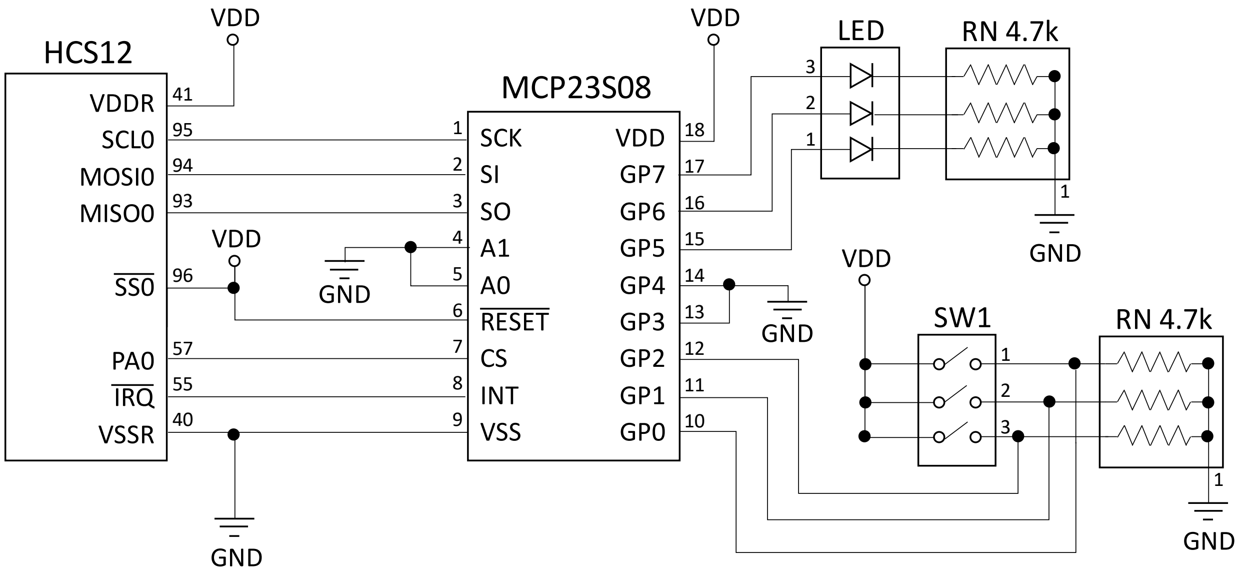


Figure 3. I/O Expender Experiment Setup

a) Assemble the circuit as shown in **Figure 3**.

b) Download the Lab7 SPI template code from Piazza.

c) Compile and run the SPI template code to verify the default functionality. In the template code, MCP23S08 is set up so that the lower 3 bits of the IODIR register are set to input and its upper three bits are set to output. The I/O information is contained in the GPIO register of MCP23S08. The desired input comes from switches and the output goes to the LED bar. Individual switches show bit information to be displayed in the LED bar.

d) Modify the template code so that the LED bar displays bits in reverse order (hint: code in the ISR function)

e) Setup the hardware (GPx pins) and MCP23S08 registers so that the input goes to the upper three bits of GPIO and the output to the lower three bits of GPIO.

PART 2. I^{2}C-based EEPROM Interfacing and Programming

I²C (Inter-Integrated Circuit) is a multimaster serial bus invented by Philips used for attaching low-speed peripherals to a motherboard, embedded system, cellphone, or other electronic device. Not to be confused with the term Two Wire Interface which only describes a compatible hardware interface.

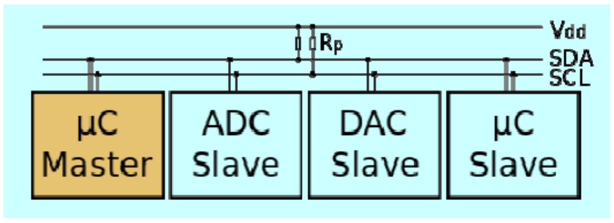


Figure 4. I2C Bus with multiple peripheral devices

- Uses the SCL signal to carry clock signal to synchronize data transfer
- Uses the SDA signal to carry data and address
- The SDA and SCL pins of I2C devices (masters and slaves) are open-drain and need external pull-up resistors.

MC9S12 I2C Interface

- Implements a subset of the I2C protocol
- Provides interrupts on start and stop bits in hardware to determine if the I2C bus is free
- Supports only 7-bit addressing
- Supports 100 Kbps baud rate but requires the user to limit the slow rate to no higher than 100 ns if the 400 Kbps baud is to be used
- Use PJ7 (SCL) and PJ6 (SDA) pins to support the I2C communication.
- Use five registers to support its operation:
    - I2C Control Register (IBCR)
    - I2C status Register (IBSR)
    - I2C data I/O register (IBDR)
    - I2C Frequency Divider Register (IBFD)
    - I2C Address Register (IBAD)

Microchip 24LC01B EEPROM

The 24LC01B device is a 1 Kbit EEPROM (Electrically Erasable PROM). The device is organized as one block of 128 x 8-bit memory with a 2-wire serial interface for I2C commnication. The 24XX01 also has a page write capability for up to 8 bytes of data.

The 24XX01 supports a bidirectional, 2-wire bus and data transmission protocol. A device that sends data onto the bus is defined as transmitter, while defining a device receiving data as a receiver. The bus has to be controlled by a master device which generates the serial clock (SCL), controls the bus access and generates the Start and Stop conditions, while the 24XX01 works as slave. Both master and slave can operate as transmitter or receiver, but the master device determines which mode is activated.

Lab Procedure

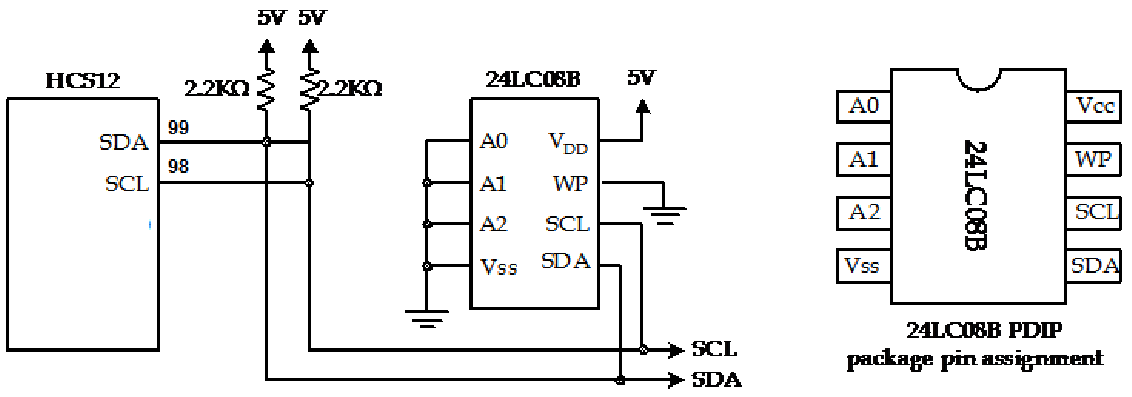


Figure 5. I^{2}C experiment setup

In this experiment, we will write and read 8-bit (byte) data to and from EEPROM.

a) Assemble the circuit as shown in **Figure 5**.

b) Download the Lab7 I2C template code from Piazza.

c) Compile and run the I2C template code to verify the default functionality. In the default application a number 0x10 is written to address 0x05, followed by reading it back to the microcontroller.

d) Modify the code so that you can write a page (8 bytes) of information in sequence with a single WRITE command.

Write-up Questions

1. Fill in the blanks to configure the SCI1 module of HCS12 with the following settings:

- 14400 baud (Bus clock is 24 MHz)
- SCI enabled in wait mode
- One start bit, 8 data bits, one stop bit
- Enable transmit and receive
- Enable TDRE (TX data register empty) interrupt
- Enable RDRF (RX data register full) interrupt
- No loop back
- Enable parity checking and use odd parity

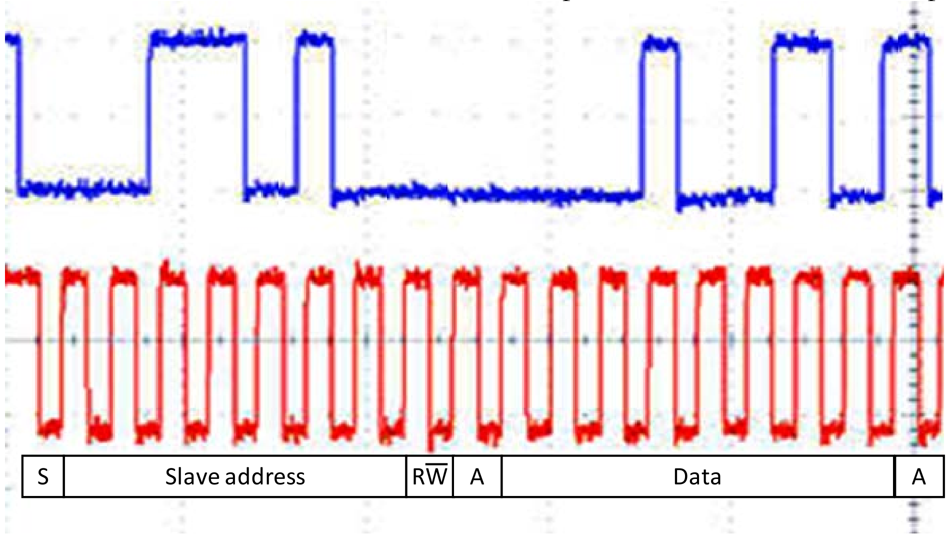
```
SCI1BD  = ___________;  // 14400 baud
SCI1CR1 = ___________;  // SCI enabled in wait mode; enable parity and use odd parity
SCI1CR2 = ___________;  // Enable TDRE and RDRF interrupts; enable TX and RX
```

2. Fill in the blanks to configure the HCS12 SPI module to operate with the following settings:

- Baud rate set to 500 KHz (Bus clock is 24 MHz)
- Master mode
- Enable TX/RX interrupts
- Clock pin idle high (active low) with data sampled on the rising edge of the clock
- Disable slave select output
- Shift least significant bit first
- SPI clock operates normally in stop mode

```
SPI0BR  = __________;
SPI0CR1 = __________;
SPI0CR2 = __________;
```

3. An I2C communication has been captured by a mixed signal oscilloscope. SDA is the blue waveform and SCL is the red one. The address and data portions are indicated in the picture.



a) What is the address of the slave?

b) What is the data value being transferred? Is the master reading or writing the data?

Lab Demo (50 points)

Demonstrate your working system to the TA and get a confirmation of completion.

Lab Report (50 points)

Make sure you include the following in your report:

1. Abstract
   - Short paragraph stating the objectives and accomplishments
2. Hardware design **(10 points)**
   - Detailed schematic diagram; do not copy and paste the block diagram given the handout

- Photos of your boards and circuits

3. Software design **(10 points)**
   - High level description of the software
   - Program code listing (including comments)
4. Discussion **(5 points)**
   - Technical challenges encountered and how they are addressed (e.g., took longer than expected due to some reasons, could not finish within the lab hours)
5. Answers to the questions **(15 points)**
6. Conclusion
   - A very short concluding remark
   - Summary of the contributions of each member