Recent Lectures

Timer capabilities and configuration

- Counters
- Pre/post scalers
- Interrupts

Capture/Compare/PWM Module

- Configuration
- Timer use

Today: Synchronous Serial Communication

Two standards:

- SPI
- I^2C

Administrivia

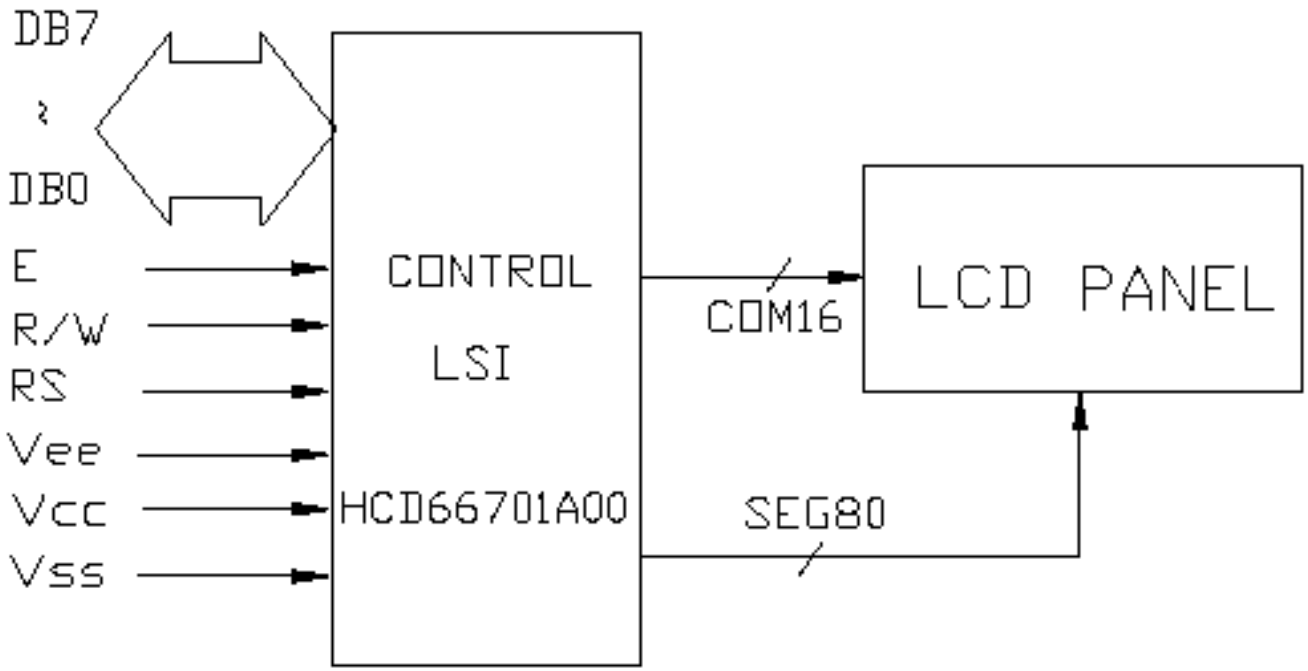
Lab:

- Lab exercise 3 must be completed by Friday
- Milestone 3 (PWM control) due close of lab on Friday
- Milestone 4 due in 2 weeks: synchronous serial communication
- Extended lab hours

Actuator of the Day...

Actuator of the Day
LCD Display

Parallel data interface for control and character display



LCD Display

Character Set

New Device Arrivals

Synchronous serial devices:

- SRAM: 8Kx8
- EEPROM: 16Kx8
- Digital output (shift register)
- Digital potentiometers

Microphones

Speakers

SPI Support in the PICs

- Only the 16f876 and the 16f88 chips support SPI (the 628s do not)
- Pins of interest:
  - SCK: clock in/out
  - SDO: serial data out
  - SDI: serial data in
  - SS: slave select (slaves only). For a master, you must allocate some digital output(s) to generate the select signals.

SPI Master in CCSC

Initialization:

```
#define SELECT PIN_A1
// Configure SDO, SCK, and SELECT as outputs
// Configure SDI as an input


// Init the SPI interface
setup_spi(SPI_MASTER | SPI_H_TO_L |
  SPI_CLK_DIV_16);
output_high(SELECT);
```

SPI Master

Simultaneous read and write:

```
int data_from_slave, data_to_slave;


output_low(SELECT);
data_from_slave =
  spi_read(data_to_slave);
output_high(SELECT);
```

SPI Master

Simultaneous read and write (a different way):

```
int data_from_slave, data_to_slave;


output_low(SELECT);
spi_write(data_to_slave);
// This use of spi_read() does not
//   generate a clock signal
data_from_slave = spi_read();
output_high(SELECT);
```

SPI Master

Read from slave:

```
int data_from_slave;


output_low(SELECT);
// This use of spi_read() forces a
//   a clock signal
data_from_slave = spi_read(0);
output_high(SELECT);
```

SPI Master

Write only:

```
int data_to_slave;


output_low(SELECT);
spi_write(data_to_slave);
output_high(SELECT);
```

SPI Slave

Initialization:

```
// Configure SDO as an output
// Configure SDI, SCK, and SS as inputs

// Init the SPI interface
setup_spi(SPI_SLAVE |SPI_H_TO_L);
```

SPI Slave Write/Read

```
int data_from_master,
 data_to_master;


// Load the data into the buffer
spi_write(data_to_master);


// Wait for the master to initiate
while(!spi_data_is_in()){
    //  Do other things
};
data_from_master = spi_read();
```

An Alternative to Busy-Waiting

When a slave receives a byte, the SSP interrupt will be raised (if configured)

Master/Slave Interaction

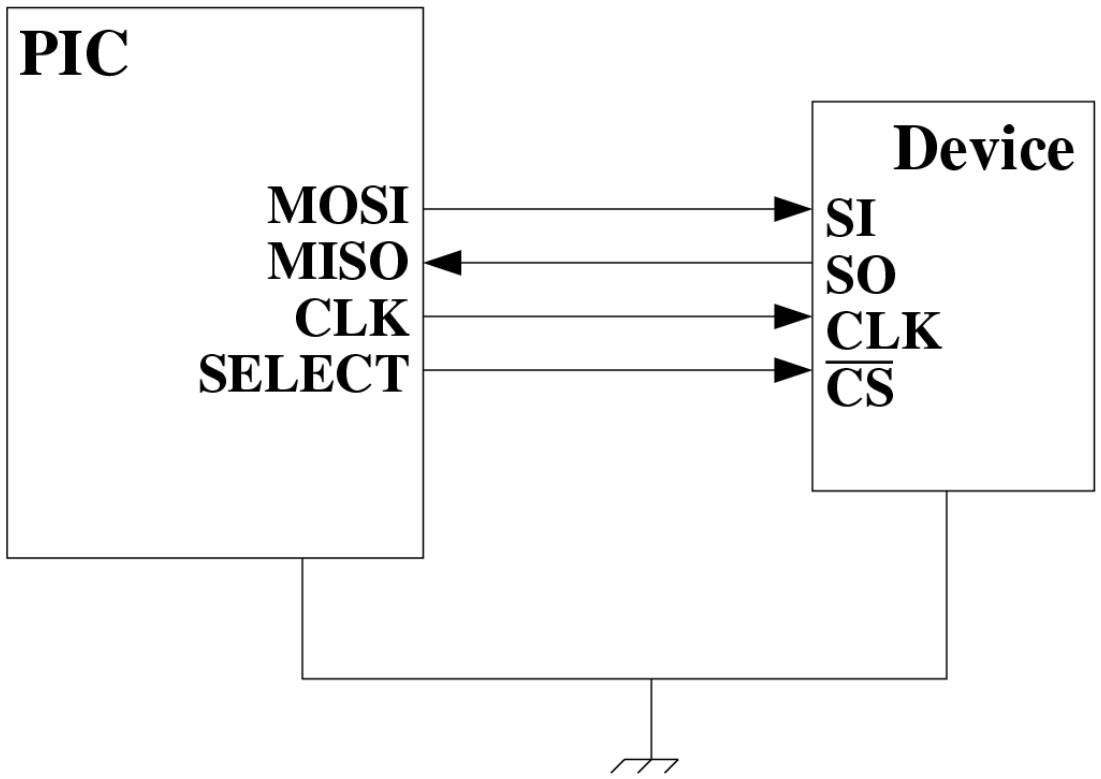
- In some cases, a master/slave transaction will involve more than one exchange of bytes
  - For example, a command byte followed by a data byte
  - Or a command byte to which the slave must respond with a data byte
- The master must give the slave an appropriate amount of time after the first exchange to prepare for the next exchange
  - This timing will depend on how you design your slave

SPI Clock Configuration

- The clock can be related to the main system clock:
  - div_4
  - div_16
  - div_64
- Or can be tied to Timer 2:
  - div_2

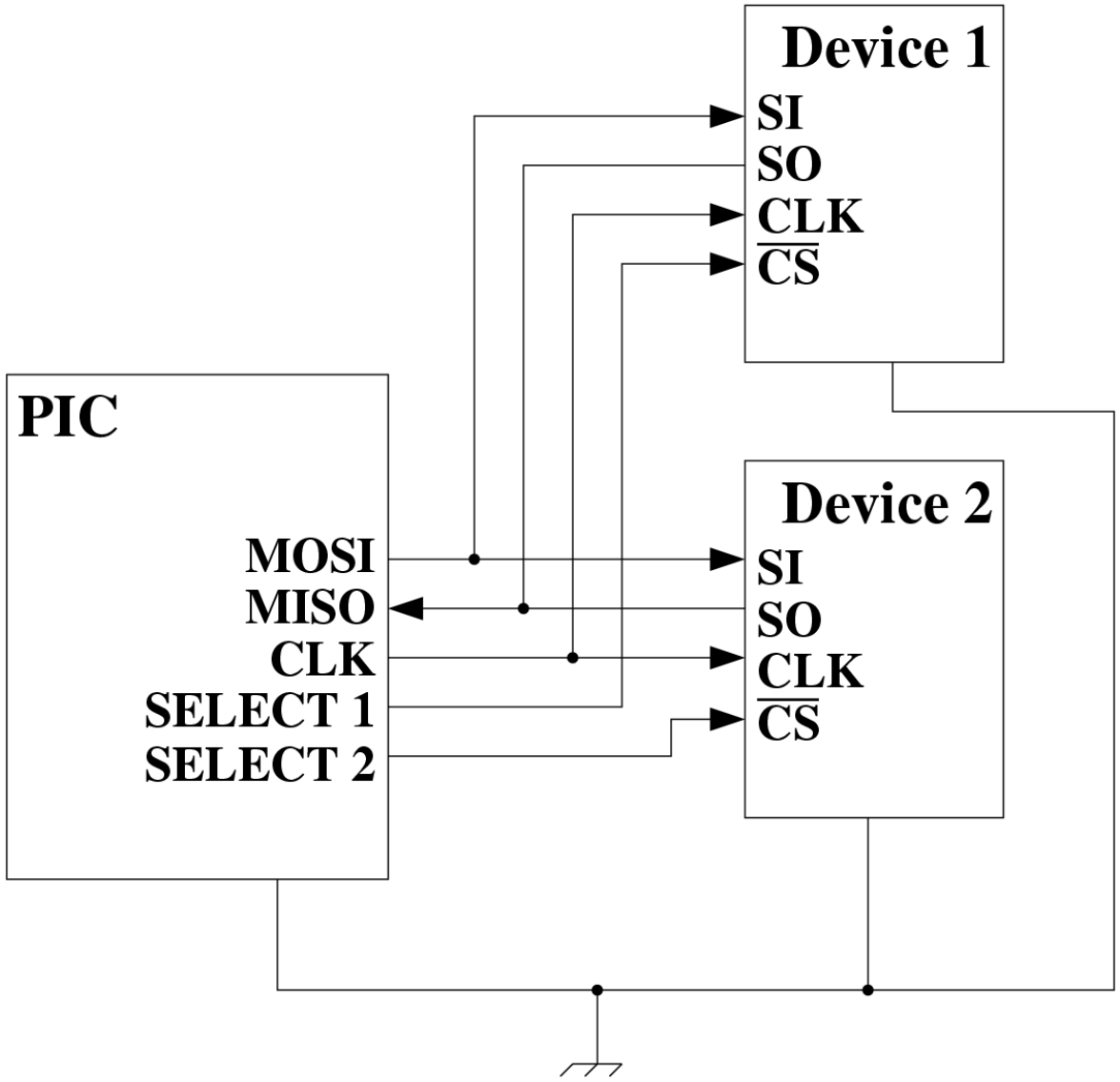
Unidirectional SPI

- If the devices is only a sensor, it does not need to receive an input signal line
- Likewise, if the device is an actuator, it does not require an output signal line

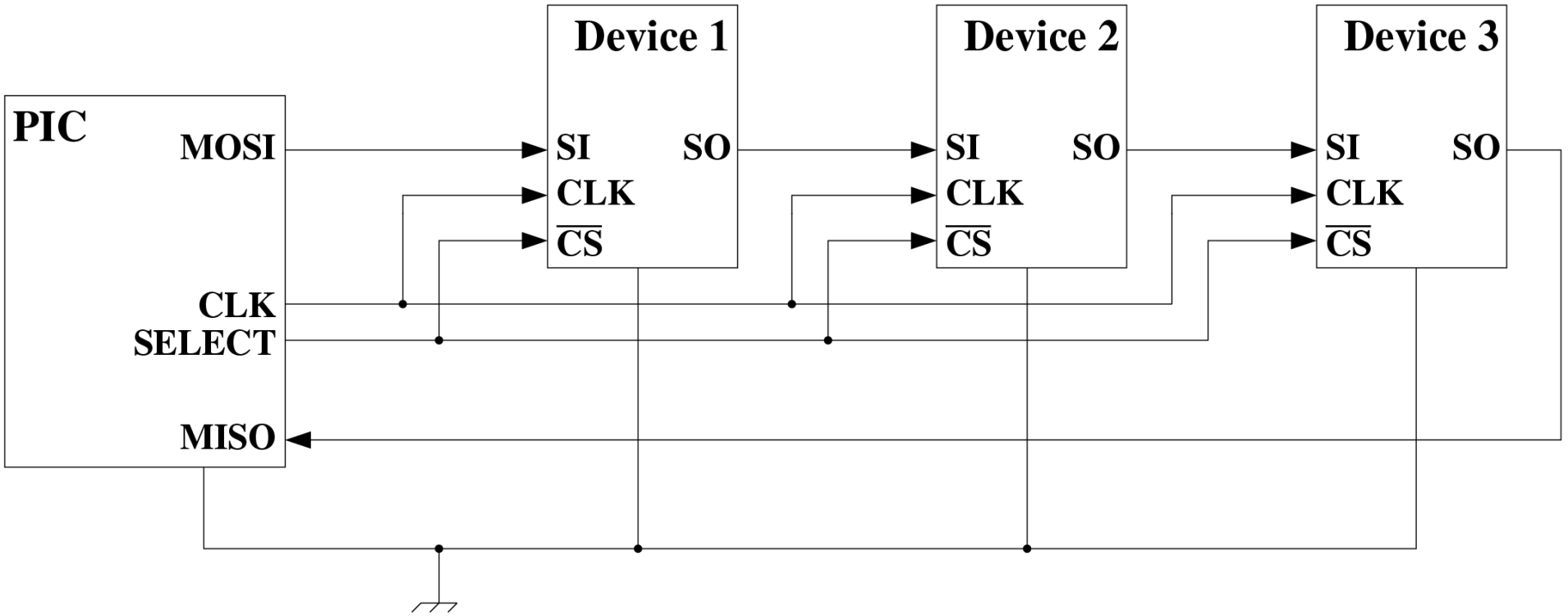


SPI With Multiple Slaves

- Serial bus: MOSI/MISO/CLK are shared
- Separate select lines for each device
- Both devices can drive MISO (which one is determined by the select state)



A Serial Set of Slaves



How do we do communication in this case?

Serial Slaves

- The slave buffers must act as shift registers
- The master first writes out the byte for device 3
- ... then the bytes for 2 and 1, respectively

- As the subsequent bytes are written, the byte for device 3 is shifted to device 2 and then to device 3

- The chip select signal tells the slaves when they can latch a valid byte

Serial Slaves

- As the three writes are performed by the master, the slave bytes arrive in the order: 3, 2, and 1

- Note that not all SPI devices can be serialized in this way

What are the pros and cons of this scheme?

Serial Slaves Pros/Cons

- Pros:
  - Only need one chip select line for a set of devices
- Cons:
  - More work to read data from a single device
  - Lower data rates per device

Serial Communication with I^2C (Inter Integrated Circuit)

- Also a master/slave system, but with potentially many masters
- Unidirectional communication: only one device may write to the bus at any one time
- Only two signal lines:
  - SDA: data transmission
  - SCL: clock
- Up to 400kps

What is missing?

Next Time

- Lab Friday:
  - Milestone 3 is due
  - Lab exercise 3 must be completed
- Lecture Tuesday:
  - Finish up I^2C
  - Come prepared to design our collaborative project
    - What do the devices look like (sensors and actuators)
    - What are the communication standards?
    - What algorithms sit between the sensors and actuators?