Serial Interconnect Buses—I²C (SMB) and SPI

55:036

Embedded Systems and Systems Software

Purpose of Serial Interconnect Buses

- Provide low-cost—i.e low wire/pin count—connection between IC devices
- There are lots of serial bus "standards"
  - I²C
  - SMB
  - SPI
  - Microwire
  - Maxim 3-wire
  - Maxim/Dallas 1-wire
  - etc.

Purpose of Serial Interconnect Buses

- Provide low-cost—i.e low wire/pin count—connection between IC devices
- There are lots of serial bus "standards"
  - **I²C**
  - SMB
  - **SPI**
  - Microwire
  - Maxim 3-wire
  - Maxim/Dallas 1-wire
  - etc.

We'll focus on these.

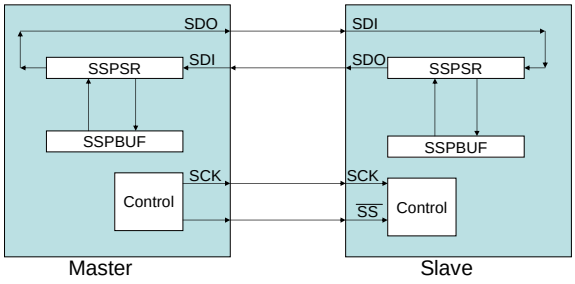
Serial Peripheral Interface (SPI)

- Originally developed by Motorola
- Synchronous, serial protocol
  - Data timing is controlled by an explicit clock signal (SCK)
- Master-slave
  - Master device controls the clock
- Bi-directional data exchange
  - data clocked into and out-of device at same time

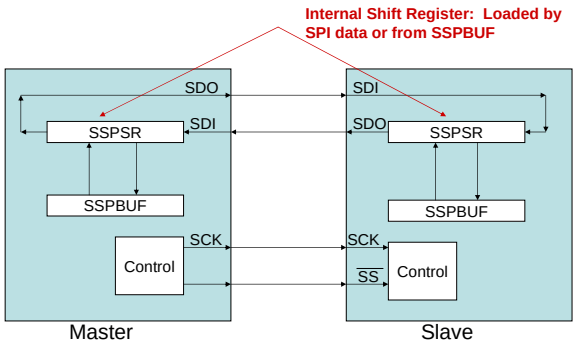
SPI signals

- $\overline{SS}$ ($\overline{CS}$) (Slave Select, Chip Select)
  - When SS is low the slave is enabled
- SCK (Serial Clock)
  - Controls the sending and reading of data
- SD0 (Serial Data Out)
  - Carries data OUT of the device
- SDI (Serial Data In)
  - Carries data INTO the device

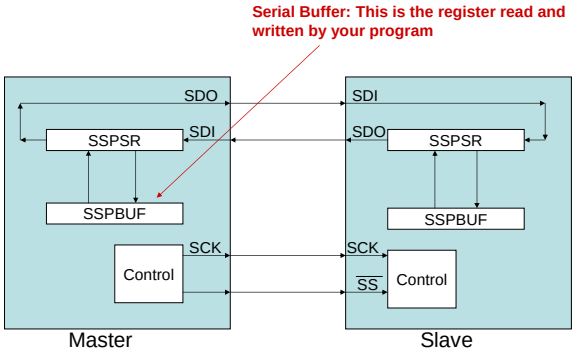
SPI Data Loop



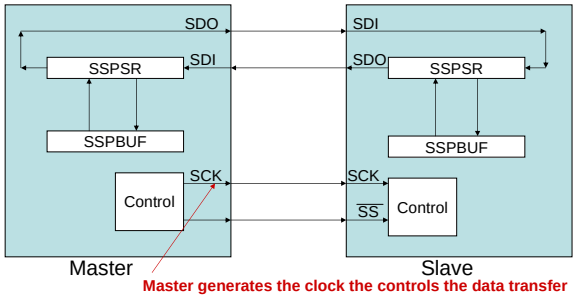
SPI Data Loop



SPI Data Loop

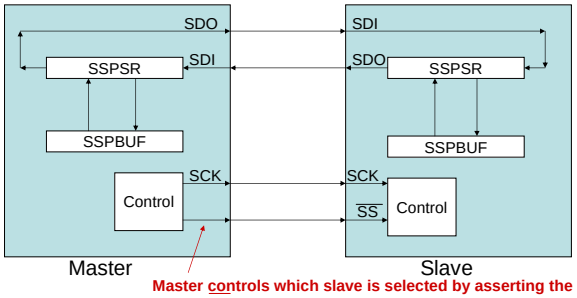


SPI Data Loop



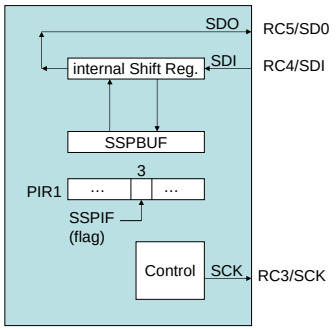
Master generates the clock the controls the data transfer

SPI Data Loop



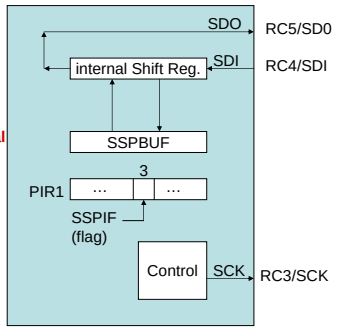
Master controls which slave is selected by asserting the slave's SS

PIC 18F452 SPI Module

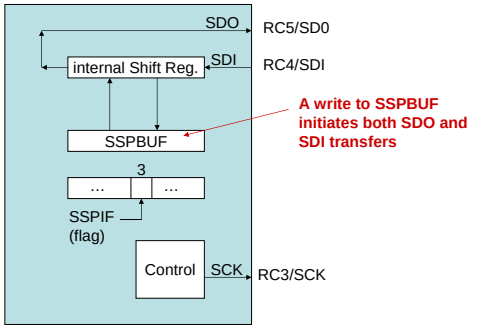


PIC 18F452 SPI Module

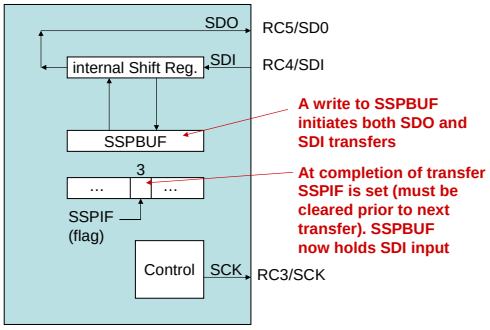
Actually the SPI Interface is part of a multifunctional PIC module called MSSP (Master Synchronous Serial Port) that supports both SPI and I²C



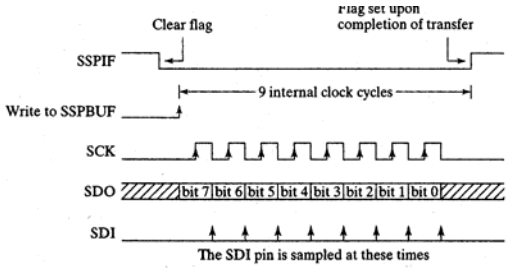
PIC 18F452 SPI Module



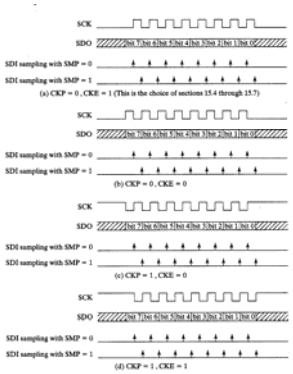
PIC 18F452 SPI Module



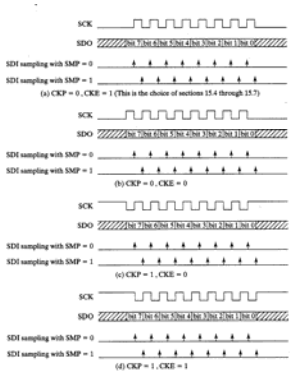
SPI Timing



SPI Modes



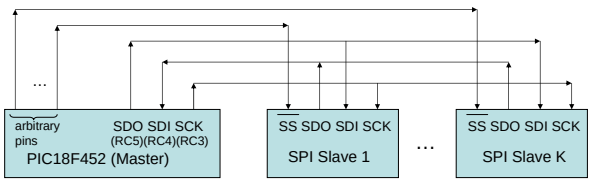
SPI Modes



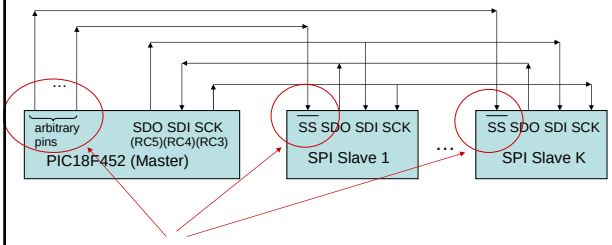
Mode selection is controlled by three bits:
CKP (Clock Polarity)
CKE (Clock edge select)
SMP (SPI sample time)

Determining the right mode to use for a given device can be tricky (see section 15.3 in the text)

Connecting Multiple SPI Devices

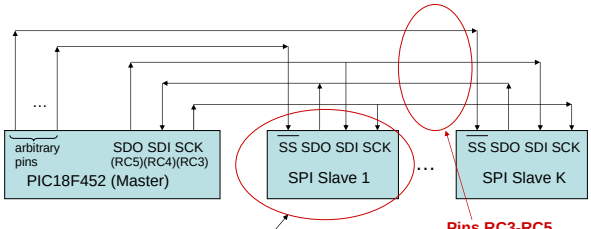


Connecting Multiple SPI Devices



Slave selection is NOT handled by the MSSP Unit

Connecting Multiple SPI Devices



Note: On the QwikFlash Boards the MAX522 DAC is permanently connected to the SPI interface (RC0 is used for SS)

Pins RC3-RC5 (SCK, SDI, SDO) are also brought out to the H2 expansion header to allow connection of additional SPI devices

Using SPI with C18 C

- Setting up the SPI Unit

Function: Initialize the SSP module.
Include: spi.h
Prototype: void OpenSPI(unsigned char ***sync_mode***,
unsigned char ***bus_mode***,
unsigned char ***smp_phase***);

Using SPI with C18 C

Setting up the SPI Unit (continued)

Arguments: *sync_mode*
One of the following values, defined in spi.h:
SPI_FOSC_4 SPI Master mode, clock = FOSC/4
SPI_FOSC_16 SPI Master mode, clock = FOSC/16
SPI_FOSC_64 SPI Master mode, clock = FOSC/64
SPI_FOSC_TMR2 SPI Master mode, clock = TMR2 output/2
SLV_SSON SPI Slave mode, /SS pin control enabled
SLV_SSOFF SPI Slave mode, /SS pin control disabled
bus_mode
One of the following values, defined in spi.h:
MODE_00 Setting for SPI bus Mode 0,0
MODE_01 Setting for SPI bus Mode 0,1
MODE_10 Setting for SPI bus Mode 1,0
MODE_11 Setting for SPI bus Mode 1,1

Using SPI with C18 C

Setting up the SPI Unit (continued)

smp_phase
One of the following values, defined in spi.h:
SMPEND Input data sample at end of data out
SMPMID Input data sample at middle of data out

Remarks: This function sets up the SSP module for use with a SPI bus device.

File Name: spi_open.c

Code Example: OpenSPI(SPI_FOSC_16, MODE_00, SMPEND);

Using SPI with C18 C

Writing to the SPI bus:

WriteSPI
putcSPI
Function: Write a byte to the SPI bus.
Include: spi.h
Prototype: unsigned char WriteSPI(unsigned char ***data_out***);
unsigned char putcSPI(unsigned char ***data_out***);

Arguments: *data_out* Value to be written to the SPI bus.
Remarks: This function writes a single data byte out and then checks for a write collision. putcSPI is defined to be WriteSPI in spi.h.

Return Value: 0 if no write collision occurred
-1 if a write collision occurred

File Name: spi_writ.c
#define in spi.h

Code Example: WriteSPI('a');

Using SPI with C18 C

Reading from the SPI bus:

ReadSPI
getcSPI
Function: Read a byte from the SPI bus.
Include: spi.h
Prototype: unsigned char ReadSPI(void);
unsigned char getcSPI(void);
Remarks: This function initiates a SPI***x*** bus cycle for the acquisition of a byte of data. getcSPI is defined to be ReadSPI in spi.h.

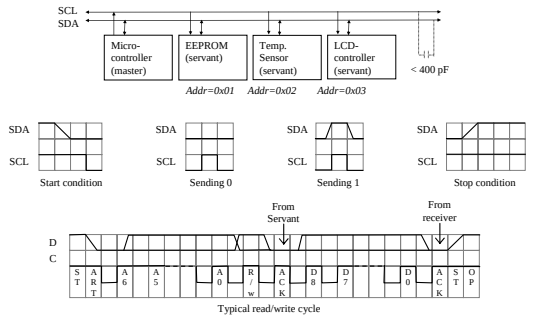
Return Value: This function returns a byte of data read during a SPI read cycle.
File Name: spi_read.c
#define in spi.h

Code Example: char x;
x = ReadSPI();

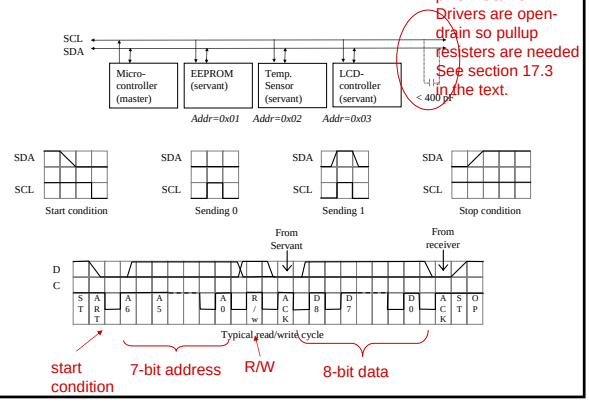
Another Serial Bus

- I²C (Inter-IC)
  - Two-wire serial bus protocol developed by Philips Semiconductors nearly 20 years ago
  - Enables peripheral ICs to communicate using simple communication hardware
  - Data transfer rates up to 100 kbits/s and 7-bit addressing possible in normal mode
  - 3.4 Mbits/s and 10-bit addressing in fast-mode
  - Common devices capable of interfacing to I²C bus:
    - EPROMS, Flash, and some RAM memory, real-time clocks, watchdog timers, and microcontrollers

I2C bus structure



I2C bus structure



PIC18f452 uses pins RC3/RC4. Drivers are open-drain so pullup resisters are needed See section 17.3 in the text.