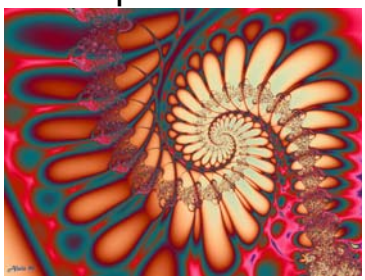
Serial Peripheral Interface



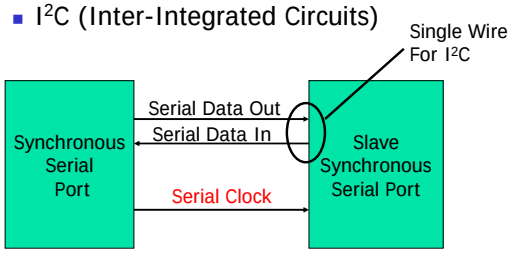
Embedded Systems Interfacing

Overview

- Interfaces for Communication
  - Synchronous Serial Interface
    - I²C (see handout)
    - SPI (this chapter)
  - Asynchronous Serial Interface (chapter 8)
  - Parallel Master Port (see handout)
- SPI Module
- External EEPROM

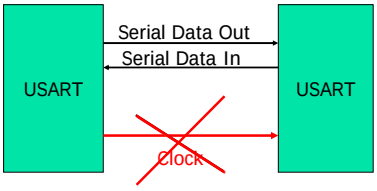
Synchronous Serial Standard

- SPI (Serial Peripheral Interface)
- I²C (Inter-Integrated Circuits)



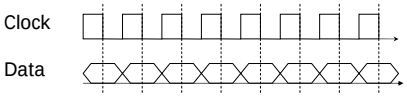
Asynchronous Serial Standard

- Universal Asynchronous/Synchronous Receiver Transmitter (USART or UART)



Synchronous Communication

- Valid data time determined by clock edge
  - Clock edge may be evenly spaced in time (SPI)
  - Clock edge may be unevenly spaced in time (I²C)



Three Primary Protocols

- Serial Peripheral Interface or SPI
  - Motorola (now Freescale)
- Microwire
  - National Semiconductor
  - Easiest to understand
- Inter-Integrated Circuit or I²C
  - Phillips (now NXP)
  - Hardest to understand

Summary

	Synchronous		Asynchronous
Peripheral	SPI	I²C	UART
Bit rate	10 Mbit/sec	1 Mbit/sec	500 kbit/sec
Bus Size	Number of pins	128	One (EIA-232) 256 (EIA-485)
Pins	3+n*CS	2	2
Pro	▪Simple ▪Low cost ▪High speed	▪Small pin count ▪Allows multiple masters	▪Longest distance ▪Improved noise immunity
Cons	▪Single master ▪Short distance	▪Slower than SPI ▪Short distance	▪Slower than Synchronous

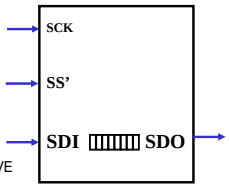
Serial Peripheral Interface (SPI)

- Four wires
  - Master Out Slave In[2] (MOSI) or Serial Data Out (SDOx)
  - Master In Slave Out[2] (MISO) or Serial Data In (SDIx)
  - Serial ClocK (SCKx)
  - Slave Select (SSx or FSYNCx)
- Data valid on clock transitions
  - May be changed from leading to trailing edge
- Multi master (single master at a time) and multi slaves

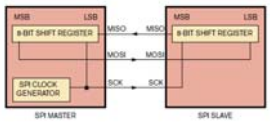
[2] Freescale Specification

SPI Slave

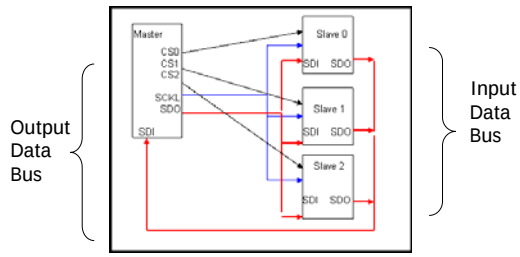
- 4-wire Interface
- Shift register to shift register transfer
- SDO_{MASTER} to SDI_{SLAVE} (MOSI)
- SDI_{MASTER} from SDO_{SLAVE} (MISO)



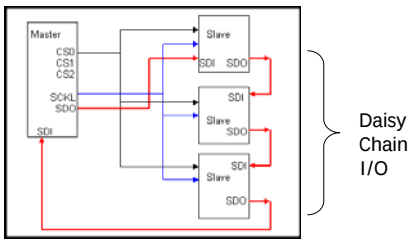
Shift Register Swap



Master & Multiple Slaves



Master & Multiple Slaves



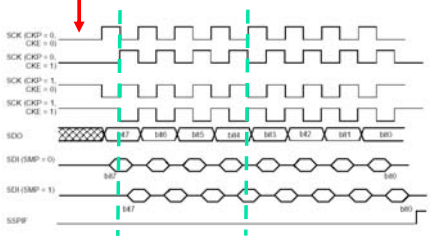
SPI MODE TIMING

SPI-mode	CPOL	CPHA
0	0	0
1	0	1
2	1	0
3	1	1

Standard SPI Modes

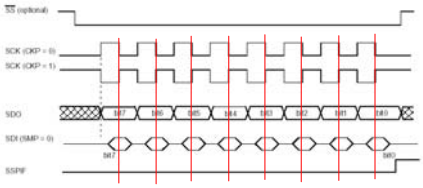
0, 1 →
0, 0 →
1, 1 →
1, 0 →

Idle State (Clock Phase – CKP)



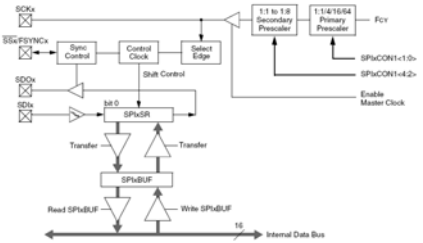
MASTER MODE

SPI MODE TIMING

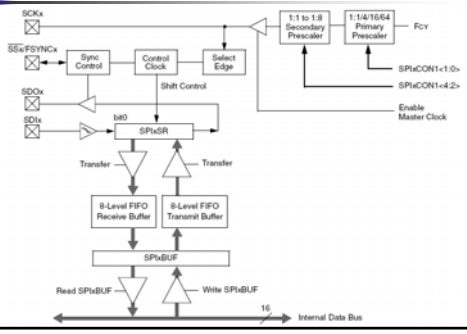


SLAVE MODE WITH CKE = 0

Standard SPI



Enhanced SPI



SCK for Various Primary/Secondary Prescaler

f_{OSC} = 8 MHz

	1	2	3	4	5	6	7	8
1	8000000	4000000	2666667	2000000	1600000	1333333	1142857	1000000
4	2000000	1000000	666666.7	500000	400000	333333.3	285714.3	250000
16	500000	250000	166666.7	125000	100000	83333.33	71428.57	62500
64	125000	62500	41666.67	31250	25000	20833.33	17857.14	15625

f_{OSC} = 32 MHz

	1	2	3	4	5	6	7	8
1	(invalid)	(invalid)	(invalid)	8000000	6400000	5333333	4571429	4000000
4	8000000	4000000	2666667	2000000	1600000	1333333	1142857	1000000
16	2000000	1000000	666666.7	500000	400000	333333.3	285714.3	250000
64	500000	250000	166666.7	125000	100000	83333.33	71428.57	62500

SCK above 10 MHz is invalid

SPI Transfers

- To write master data to slave
  - #define BUSY1 SPI1STATbits.SRMPT
  - while(BUSY1 == 0); //wait for previous transfer
  - SSP1BUF = datum; //send to slave
- To read master data from slave
  - SSP1BUF=0; //write dummy data, ignored by slave
  - while(BUSY1 == 0); //wait for transfer
  - A=SPI1BUF; //save slave's data

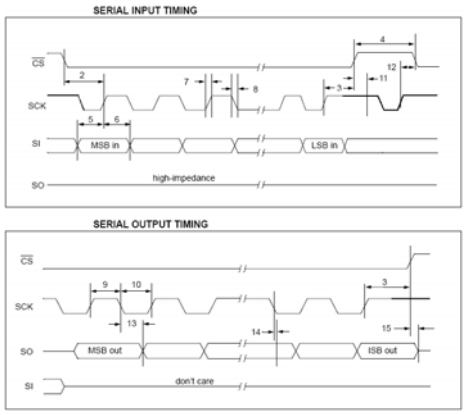
Transfer initiated by write to SSP1BUF.

Parallel Port Expander

- Parallel-Out with Double Buffering
  - 74HC595
  - Works similar to 74HC495 in Text
- Parallel-In (Load) Serial Out (Shift)
  - 74HC165
  - See Text

SEEPROM 25AA040A

- 10 Mbps transfer rate
- 512 by 8 organization
- Write Page mode (up to 16 bytes)
- Self-timed erase and write (5 ms)
- Endurance > 1,000,000 write cycles
- Data retention > 200 years

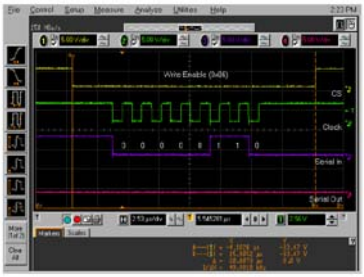


CKP = ? 1

CKE = ? 0

SMP = ? 1

Write Enable



From Microchip AN1000A

Definitions

```
#define CS_ RC7
#define OUTPUT 0
#define INPUT 1
#define DUMMY 0x00
#define WIP_MASK 0x01

#define WRSR 0x01
#define WRITE 0x02
#define READ 0x03
#define WRDI 0x04
#define RDSR 0x05
#define WREN 0x06
```

Instruction Set

Instruction Name	Instruction Format	Description
READ	0000 $A_8$011	Read data from memory array beginning at selected address
WRITE	0000 $A_8$010	Write data to memory array beginning at selected address
WRDI	0000 x100	Reset the write enable latch (disable write operations)
WREN	0000 x110	Set the write enable latch (enable write operations)
RDSR	0000 x101	Read STATUS register
WRSR	0000 x001	Write STATUS register

Note: A_8 is the 9th address bit, which is used to address the entire 512 byte array.
x = don't care.

Initialization

```
void spiInit(void){
// Configure I/O pins
        CS_ = 1;                        //Deassert chip select
        _TRISC7 = 0;                    //CS_ as output
//Configure SSPCON1
        SSP1CON1 = 0b000001000111101;
                                        //SCK & SDO used by module
                                        //Sample middle, SS1 not used,CKP high
                                        //Master mode 1:1 x 16:1
        SSP1CON2 = 0x0000               //Legacy mode
        //Configure SSPSTAT mode (1,1)
        SSP1STAT = 0;                   //Clear SSPSTAT
        _SPIEN = 1;                     //Enable SPI #1
        }
```

Support Functions

```
#define UCHAR unsigned char
#define BUSY1 SPI1STATbits.SRMPT

UCHAR spiInOut(UCHAR datum){
        CS_ = 0;                //Assert chip select
        SSP1BUF = datum;        //send data
        while(BUSY1 == 0);      //wait for data
        return(SSP1BUF);
        }

UCHAR spiReadStatus(void){
        UCHAR datum;
        spiInOut(RDSR);
        datum=spiInOut(DUMMY); CS_ = 1;
        return(datum);
        }
```

Init and Enable Write

```
#define UCHAR unsigned char

void main(void){
        UCHAR addrLo = 0x55;
        UCHAR addrHi = 0x00;
        UCHAR data = 0xAA;
        spiInit();
        for(;;){
                //Enable write
                spiInOut(WREN); CS_ = 1;
                spiReadStatus();
```

To finish command

Write Data and Wait

```
        //Write data at <addrHi:addrLo>
        spiInOut(WRITE);
#ifdef HI_DENSITY
        spiInOut(addrHi);
#endif
        spiInOut(addrLo);
        spiInOut(data); CS_ = 1;
        //wait on Write In Progress
        while((spiReadStatus()& WIP_MASK) == 1);
```

To finish command

Read Data

```
        //Read data at <addrHi:addrLo>
        spiInOut(READ);
#ifdef HI_DENSITY
        spiInOut(addrHi);
#endif
        spiInOut(addrLo);
        data=spiInOut(DUMMY); CS_ = 1;
```

To finish command

Enable and Disable Writes

```
        //Send the write enable sequence
        spiInOut(WREN); CS_ = 1;
        spiReadStatus();
        //Send the write disable sequence
        spiInOut(WRDI); CS_ = 1;
        spiReadStatus();
        }
}
```

To finish command

Synchronous Serial Port Peripherals

- Serial EEPROM
- Analog-to-Digital Converters
- Digital-to-Analog Converters
- Display Drivers
- Real Time Clocks
- Parallel I/O Ports
- MMC and SD Card memory (1-Wire Mode)
- Other special function devices

SPI Pros and Cons

- Pro
  - Very simple protocol
  - Supports multiple slave device
    - Star
    - Daisy-Chain
- Cons
  - No acknowledge (“Any body home?”)
  - Requires one Chip Select per slave in star
  - Must deal with all slaves when daisy-chain

SPI Modes

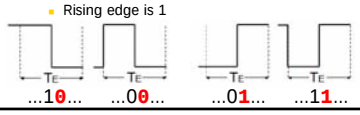
- PIC as Master
  - Discussed
  - See previous slides
- PIC as Slave
  - Not discussed
  - See Data Sheet or Family Reference Manual

Implementation in PIC24FJ128

- Hardware Modules
  - I²C (two each)
  - SPI (two each)
  - Microwire (See Microchip’s AN1023 for PIC18F Family)
- Bit Bang in Software (See Square-1’s Serial Communications)
  - I²C
  - SPI
  - Microwire
  - Dallas 1-Wire

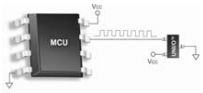
New Protocol

- UNI/O
  - Microchip protocol
  - Three pin interface
    - GND
    - Power
    - Data line with Manchester encoded data for clock extraction
      - Falling edge is 0
      - Rising edge is 1



New Protocol

- UNI/O
  - Currently only EEPROMs from Microchip available
  - I²C and SPI DIP-8 compatible pinout parts available
  - Bit Bang drivers available from Microchip
  - May not catch on



Homework #8

- Chapter 7 (Handout)
  - 1 Circular Buffer Read/Write as written with two buffers of 16 bytes each always keeping one byte open.
  - 2 (Enable 16-bit Mode)