CS/ECE 6780/5780

Al Davis

Today's topics:

- Serial I/O
  - general concepts in preparation for Lab 8

Introduction to Serial I/O

Serial communication transmits of one bit of information at a time.

One bit is sent, a time delay occurs, next bit is sent.

Used to interface to printers, keyboards, scanners, etc.

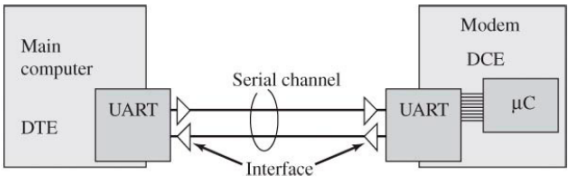
Universal asynchronous receiver/transmitter (UART) is the interface chip that implements the transmission.

A *serial channel* is collection of signals (or wires) that implement the communication.

Data terminal equipment (DTE) is the computer.

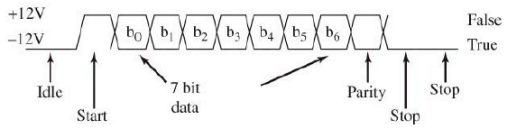
Data communication equipment (DCE) is the modem.

A Serial Channel



	CMOS Level	RS232 Level	RS422 Level
True/Mark	+5 V	TxD = -12 V	$(TxD^+ - TxD^-) = -3$ V
False/Space	+0.1 V	TxD = +12 V	$(TxD^+ - TxD^-) = +3$ V

Definitions



A *frame* is a complete and nondivisble packet of bits.

Includes both information (e.g. data, characters) and overhead (start bit, error checking, and stop bits).

Parity is generated at the transmitter and checked at the receiver to help detect errors in transmission.

Even parity makes number of 1s even (data+parity).

Odd parity makes number of 1s odd (data+parity).

Bit time is the time between each bit.

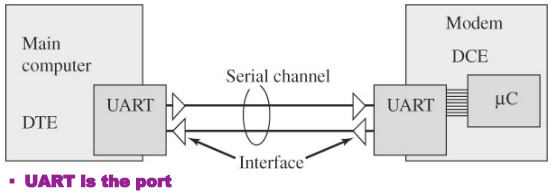
Bandwidth

Baud rate is total number bits transmitted per time.
Information is data user wishes to transmit:
Characters to be printed.
Overhead is bits added to achieve transmission:
Start bit(s), stop bit(s), parity, etc.

$$\text{Bandwidth} = \frac{\text{information bits/frame}}{\text{total bits/frame}} \times \text{baud rate}$$

More Basics

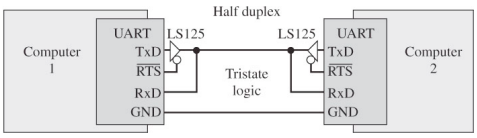
- **Classic UART: DTE & DCE communications**



  - **UART is the port**
    - » for Freescale this is SCI (serial communications interface)
      - this is just one instance
      - as usual signaling levels are TTL
      - interface logic can be used to convert to RS232 levels
        - e.g. MAX232, MC145407 chips
    - » parity is generated by Tx side and checked by Rx side

Half Duplex Signalling

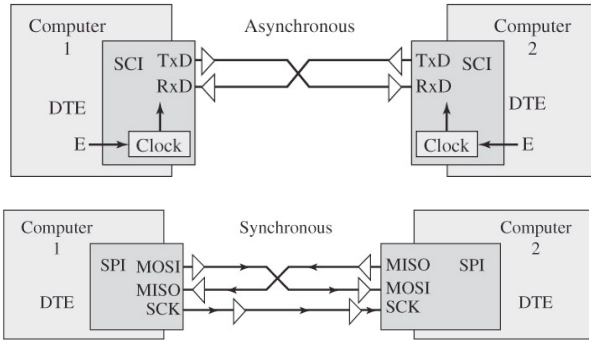
- **Half Duplex**
  - **normal usage**
    - » fixed Tx and Rx side – a.k.a. Simplex signalling
  - **expanded version**
    - » send can come from either side but only in one direction at a time
    - » problem = collisions
      - solution same as with Ethernet CDMA
      - transmit & receive + compare
        - If Tx & Rx values aren't the same then collision & retry



Other Issues

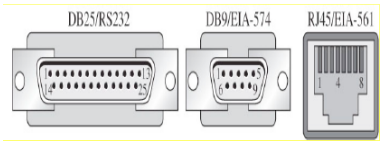
- **Full Duplex**
  - **more wires but 2 independent communication channels**
    - » concurrent send and receive buffers
- **Timing**
  - **send and receive baud rates must be the same**
- **Asynchronous (e.g. SCI)**
  - **separate send and receive clocks**
    - » start sequence is used to synch clocks for the frame
      - model is that drift won't be enough to cause errors intra-frame
      - In high speed signaling (e.g. HT, QPI, etc.)
        - this is a big problem and requires complex and energy hungry circuitry
        - long transmission paths also require significant pre- and post-emphasis circuits
- **Synchronous: multiple options**
  - **common clock (e.g. SPI)**
  - **Tx side clock – source synchronous signaling**

SCI & SPI Illustrated



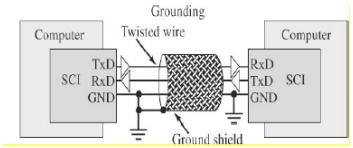
Protocols & Specifications

- **There are many**
- **Each one has specifications**
  - **electrical**
    - **» what voltage levels mean what logical value**
    - **» current sink and source requirements**
  - **cables**
    - **» often limited to some max length**
  - **mechanical**
    - **» what does the connector look like & pin function**

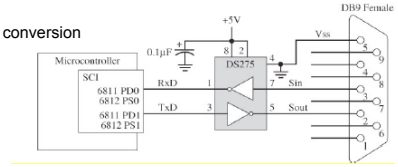


2 Common Freescale Options

Simple SCI

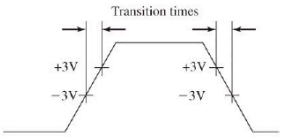


SCI to RS232 conversion

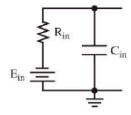


RS232 Output Specifications

Must withstand
  Short to ground
  Short to any other wire
Operating range
  True $-15 \le V_{out} \le -5V$
  False $+5 \le V_{out} \le +15V$
Maximum output voltage
  $-25 \le V_{out} \le +25V$
Short circuit current
  $I_{out} \le 0.5$ A
Transition (−3 to 3) range time ≤ 4%



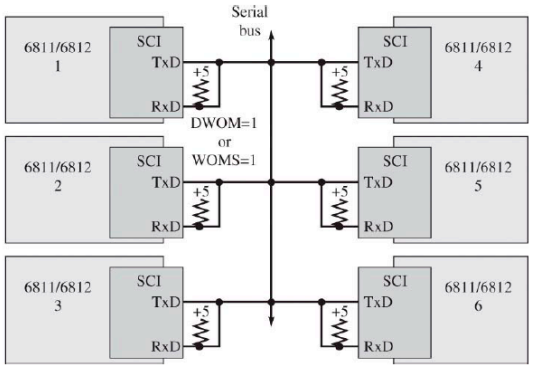
Maximum Slew Rate
  $dV_{in}/dt \le 30V/\mu s$
Operating Range
  True $-15 \le V_{in} \le -3V$
  Translation $-3 < V_{in} < +3V$
  False $+3 \le V_{in} \le +15V$
Input Resistance
  $3000\Omega \le R_{in} \le 7000\Omega$
Input Capacitance including cable
  $C_{in} \le 2500pF$
Input open circuit voltage
  $E_{in} \le 2V$



RS232 DB9 Pin Assignments

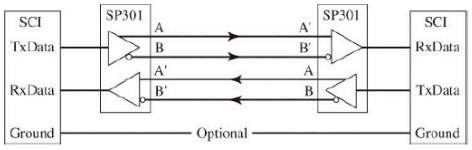
Pin	Signal	Description	True	DTE	DCE
1	DCD	Data Carrier Detect	+12	In	Out
2	RxD	Receive Data	−12	In	Out
3	TxD	Transmit Data	−12	Out	In
4	DTR	Data Terminal Rdy	+12	Out	In
5	SG	Signal Ground			
6	DSR	Data Set Ready	+12	In	Out
7	RTS	Request to Send	+12	Out	In
8	CTS	Clear to Send	+12	In	Out
9	RI	Ring Indicator	+12	In	Out

A Simple Serial Network

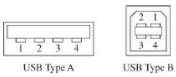


RS422/RS423/RS485 Specifications

Specification	RS232D	RS423A	RS422	RS485
Mode of operation	Single-ended	Single-ended	Diff.	Diff.
Drivers on one line	1	1	1	32
Receivers on one line	1	10	10	32
Max distance (ft)	50	4,000	4,000	4,000
Max data rate	20kb/s	100kb/s	10Mb/s	10Mb/s
Max driver output	±25V	±6V	-0.25/+6V	-7/+12V
Receiver input	±15V	±12V	±7V	-7/+12V



Universal Serial Bus (USB)

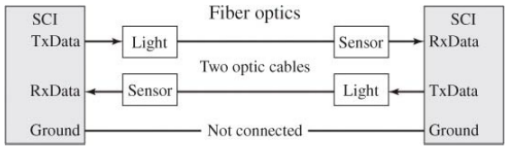


USB Type A USB Type B

Single host computer controls the USB.
Host controls all transactions using a token-based protocol.
Uses a tiered star topology with host at the center.
Up to 127 devices can be connected to one USB bus.
Plug'n'play implemented with dynamically loadable drivers.
Host detects new devices and loads appropriate driver.
Can operate at high (480Mb/s), full (12Mb/s), or low (1.5Mb/s) speeds.

Pin	Color	Function
1	Red	VBUS (5V)
2	White	D-
3	Green	D+
4	Black	Ground

Optical SCI Channel



Where & Why would you want to do this?

SCI

Most embedded microcomputers support SCI.

Common features include:

- A baud rate control register used to select transmission rate.
- A mode bit M used to select 8-bit (M=0) or 9-bit (M=1) data frames.

Each device can create its own serial port clock with period that is integer multiple of the E clock period.

Transmitting in Asynchronous Mode

Common features in the transmitter:

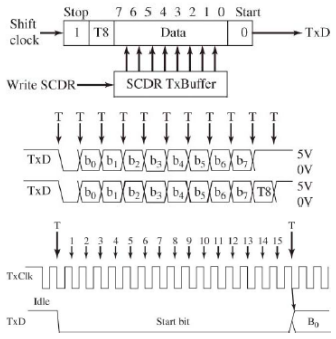
- TxD data output pin, with TTL voltage levels.
- 10- or 11-bit shift register, not directly accessible.
- Serial communications data register (SCDR), write only, separate from receive reg. though same address.
- T8 data bit for 9-bit data mode.

Control Bits for the Transmitter

Transmit Enable (TE), set to 1 to enable transmitter.
Send Break (SBK), set to 1 to send blks of 10 or 11 0s.
Transmit Interrupt Enable (TIE), set to arm TDRE flag.
Transmit Complete Enable (TCIE), set to arm TC flag.

NOTE: Tx Data Reg Empty (TDRE) flag signals that the SCDR register is empty, TDRE is cleared by reading it. Different from previous flag clearing methods where you had to write a 1 to the flag.
Read of TC flag (transmit complete) similarly clears it
Then write to the SCDR

Transmission Illustrated



Pseudo Code for Transmission Process

```
TRANSMIT Set TxD=0            Output start bit
         Wait 16 clock times  Wait 1 bit time
         Set n=0              Bit counter
TLOOP    Set TxD=bn           Output data bit
         Wait 16 clock times  Wait 1 bit time
         Set n=n+1
         Goto TLOOP if n<=7
         Set TxD=T8           Output T8 bit
         Wait 16 clock times  Wait 1 bit time
         Set TxD=1            Output a stop bit
         Wait 16 clock times  Wait 1 bit time
```


Receiving in Asynchronous Mode

Common features in the receiver:

- RxD data input pin, with TTL voltage levels.
- 10- or 11-bit shift register, not directly accessible.
- Serial communications data register (SCDR), read only, separate from transmit reg. though same address.
- R8 data bit for 9-bit data mode.

Control Bits for the Receiver

- Receiver Enable (RE), set to 1 to enable receiver.
- Receiver Wakeup (RWU), set to 1 to allow a receiver input to wakeup the computer.
- Receiver Interrupt Enable (RIE), set to arm RDRF flag.
- Idle Line Interrupt Enable (ILIE), set to arm IDLE flag.

Status Bits Generated by the Receiver

Receive Data Register Full flag (RDRF), set when new data available, clear by reading RDRF and SCDR.

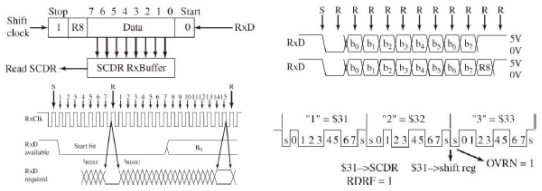
Receiver Idle flag (IDLE), set when receiver line is idle, clear by reading IDLE, then reading SCDR.

Overrun flag (OR), set when input data lost because previous frame not read, clear by reading OR and SCDR.

Noise flag (NF), set when input is noisy, clear by reading NF flag, then reading SCDR.

Framing error (FE), set when stop bit is incorrect, clear by reading FE, then reading SCDR.

Receiving Illustrated



Pseudo Code for Receive Process

```
RECEIVE Goto RECEIVE if RxD=1  Wait for start bit
        Wait 8 clock times     Wait half a bit time
        Goto RECEIVE if RxD=1  False start?
        Set n=0
RLOOP   Wait 16 clock times    Wait 1 bit time
        Set bn=RxD             Input data bit
        Set n=n+1
        Goto RLOOP if n<=7
        Wait 16 clock times    Wait 1 bit time
        Set R8=RxD             Read R8 bit
        Wait 16 clock times    Wait 1 bit time
        Set FE=1 if RxD=0      Framing error if
                               no stop bit
```


9S12C32 SCI Details

One SCI port using Port S bits 1 and 0.

Least significant 13 bits of **SCIBD** register determine baud rate.

SCICR2 register contains control bits for SCI (see Table 7.11).

SCICR1 register contains other miscellaneous SCI control bits.

- **LOOPS**: disconnects receiver from RxD pin.
- **RSRC**: when **LOOPS=1**, **RSRC=0** connects receiver to transmitter internally while **RSRC=1** connects receiver to TxD.
- **WAKE**: 0 wakeup on IDLE line, 1 wakeup on address mark.
- **ILT**: determines if idle line count starts from start or stop bit.
- **SWAI**: setting this bit causes SCI to shutdown and pause any communication.
- **PE**: setting this bit enables parity checking.
- **PT**: 0 is even parity while 1 is odd parity.

More SCI Details

Flags in **SCISR1** register can be read but not modified by software.

The error conditions are also reported in the **SCISR1** register including parity flag, **PF**, set on parity errors.

The **SCISR2** register contains two mode control and one status bit.

- **BRK13**: break character is 13 or 14 bits when 1 and 10 or 11 bits when 0.
- **TXDIR**: specifies direction of the TxD pin in single-wire mode.
- **RAF**: 1 when frame is being received.

The **SCIDRL** register contains data transmitted and received.

The **SCIDRH** register contains the 9th data bits.

SCI I/O Interrupts

Simultaneous input and output requires two FIFOs.

RxFifo passes data from InSCI handler to main thread.

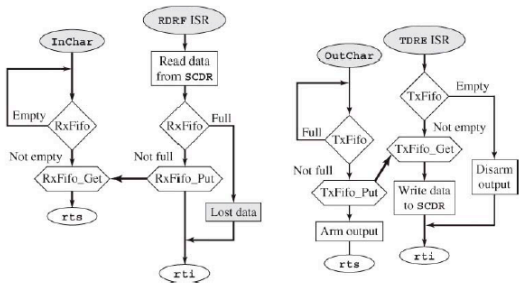
TxFifo passes data from main thread to OutSCI handler.

Since TxFifo initially empty, transmit interrupts initially disarmed.

When main thread calls OutChar, transmit interrupts armed.

Interrupt handler disarms transmit interrupts when TxFifo becomes empty.

SCI Rx, Tx, & ISR's



TxFifo full ➔ wait until there is space
RxFifo full ➔ data was lost due to Rx buffer over run

SCI Interface Ritual

```
void SCI_Init(void){
asm sei
  RxFifo_Init(); // empty FIFOs
  TxFifo_Init();
  SCIBD = 26;    // 9600 bits/sec
  SCICR1 = 0;    // M=0, no parity
  SCICR2 = 0x2C; // enable, arm RDRF
asm cli          // enable interrupts
}
```

SCI Interface ISR

```
// RDRF set on new receive data
// TDRE set on empty transmit register
interrupt 20 void SciHandler(void){
char data;
  if(SCISR1 & RDRF){
    RxFifo_Put(SCIDRL); // clears RDRF
  }
  if((SCICR2&TIE)&&(SCISR1&TDRE)){
    if(TxFifo_Get(&data)){
      SCIDRL = data;   // clears TDRE
    }
    else{
      SCICR2 = 0x2c;   // disarm TDRE
    }
  }
}
```


SCI In/Out Character

```
// Input ASCII character from SCI
// spin if RxFifo is empty
char SCI_InChar(void){ char letter;
  while (RxFifo_Get(&letter) == 0){};
  return(letter);
}
// Output ASCII character to SCI
// spin if TxFifo is full
void SCI_OutChar(char data){
  while (TxFifo_Put(data) == 0){};
  SCICR2 = 0xAC; // arm TDRE
}
```


Serial Port Printer Interfaces

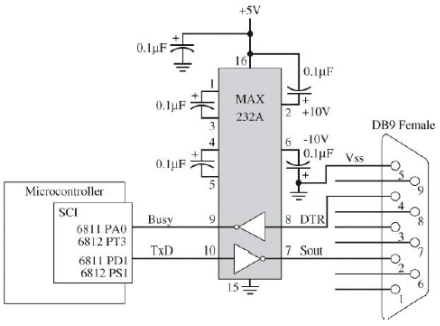
Printer bandwidth may be less than the maximum bandwidth supported by the serial channel.

- Special characters may require more time to print.
- Most printers have internal FIFOs that could get full.
- The printer may be disconnected.
- The printer may be deselected.
- The printer power may be off.

Flow control is needed to synchronize computer with variable rate output device

- DTR: data terminal ready
- XON/XOFF
  - Note: 2 approaches
  - DTR is a handshake saying send me another frame
  - Xoff is a shut up signal – more efficient for larger buffers but some timing complexity for on the fly & response time issues

SCI Simplex Printer Interface (w/ DTR handshake)



Serial Output w/ DTR

```
void Printer_Init(void){
asm sei
  TxFifo_Init(); // empty FIFOs
  SCIBD = 52;    // 9600 bits/sec
  SCICR1 = 0;    // M=0, no parity
  SCICR2 = 0x0C; // enable disarm TDRE
  TIOS &=~0x08;  // PT3 input capture
  DDRT &=~0x08;  // PT3 is input
  TSCR1 = 0x80;  // enable TCNT
  TCTL4 |= 0xC0; // both rise and fall
  TIE |= 0x08;   // Arm IC3
  TFLG1 = 0x08;  // initially clear
asm cli          // enable interrupts
}
```

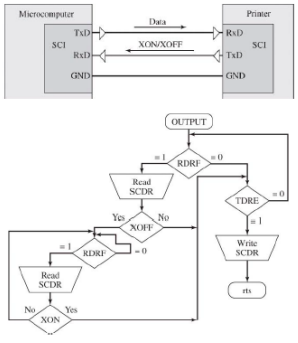

DTR Handshake ISR

```
// IC3 interrupt on any change of DTR
void interrupt 11 IC3Han(void) {
  TFLG1 = 0x08; // Ack, clear C3F
  checkIC3(); // Arm SCI if DTR=+12
}
```


Serial Output to Printer

```
void checkIC3(void){
  if(PTT&0x08)     // PT3=1 if DTR=-12
    SCICR2 = 0x0C; // busy, so disarm
  else
    SCICR2 = 0x8C; // not busy, so arm
}
// Output ASCII character to Printer
// spin if TxFifo is full
void Printer_OutChar(char data){
  while (TxFifo_Put(data) == 0){};
  checkIC3();
}
```


Using XON/XOFF (busy waiting)



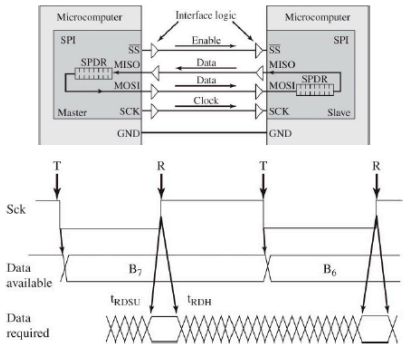
Synchronous = SPI (3 options)

Two devices communicating with SCI operate at same frequency but have 2 separate (not synchronized) clocks.

Two devices communicating with SPI operate using the same (synchronized) clock.

Master device creates the clock while slave device(s) use the clock to latch data in or out.

SPI Master/Slave Example



SPI Fundamentals

Motorola SPI includes four I/O lines:

- $\overline{SS}$ - slave select, used by master to indicate the channel is active.
- SCK - 50% duty cycle clock generated by the master.
- MOSI (master-out slave-in) - data line driven by master.
- MISO (master-in slave-out) - data line driven by slave.

Transmitting device uses one edge of clock to change data, and receiving device uses other edge to accept data.

When data transfer occurs combined 16-bit register is serially shifted eight bit positions (data exchanged).

More SPI Fundamentals

Common control features of the SPI module include:

- A baud rate control register
- A mode bit in the control register to select master versus slave, clock polarity, clock phase.
- Interrupt arm bit
- Ability to make outputs open-drain.

Common status bits for the SPI module include:

- SPIF, transmission complete
- WCOL, write collision
- MODF, mode fault

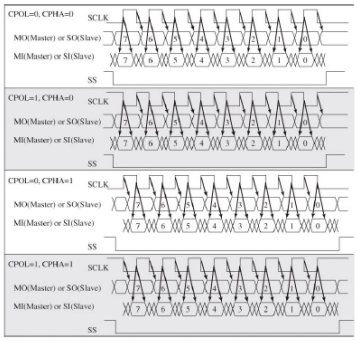
Mode fault occurs when master and slave synchronization is wrong – e.g. 2 masters

SPI Pseudo Code

```
TRANSMIT  Set n=7                       Bit counter
TLOOP     On fall of Sck, set Data=bn   Output bit
          Set n=n-1
          Goto TLOOP if n>=0
          Set Data=1                    Idle output

RECEIVE   Set n=7                       Bit counter
RLOOP     On rise of Sck, read data
          Set bn=Data                   Input bit
          Set n=n-1
          Goto RLOOP if n>=0
```


SPI Modes



CPOL sets SCLK polarity – e.g. what is IDLE

CPHA sets even or odd clock edges for the receiver shift register

9S12C32 SPI Details (Port M)

Uses four pins, $PM3 = \overline{SS}$, $PM5 = SCLK$, $PM4 = MOSI$, and $PM2 = MISO$.

If 6812 is master, set **DDRM** to make PM5, PM4, and PM3 outputs.

Can be in **run**, **wait**, or **stop** mode.

				4 MHz		24 MHz	
SPR2	SPR1	SPR0	Div	Freq	Bit Time	Freq	Bit Time
0	0	0	2	2 MHz	500 ns	12 MHz	83.3 ns
0	0	1	4	1 MHz	1 μs	6 MHz	166.7 ns
0	1	0	8	500 kHz	2 μs	3 MHz	333.3 ns
0	1	1	16	250 kHz	4 μs	1.5 MHz	666.7 ns
1	0	0	32	125 MHz	8 μs	750 kHz	1.33 μs
1	0	1	64	62.5 kHz	16 μs	375 kHz	2.67 μs
1	1	0	128	31.25 kHz	32 μs	187.5 kHz	5.33 μs
1	1	1	256	15.625kHz	64 μs	93.75 kHz	10.67 μs

SPIBR register

SPI Control Registers

SPIDR is 8-bit register used for both input and output.

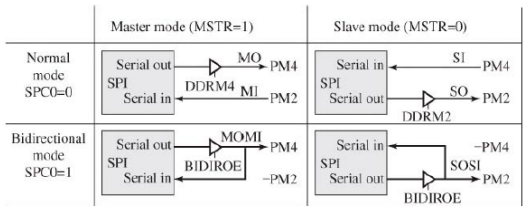
SPICR1 register species SPI mode of operation.

- **SPE**: enables the SPI system.
- **SPIE**: arms interrupts on the **SPIF** flag.
- **SPTIE**: arms interrupts on the **SPTEF** flag.
- **LSBF**: if 1 transmits least significant bit first.

SPISR register contains flags for the SPI system.

- **SPIF**: indicates that new data is available to be read.
- **SPTEF**: indicates that SPI data register can accept new data.
- **MODF**: mode error interrupt status flag.

SPI Modes



SPI Mode Selections

MODFEN	SSOE	Master Mode (MSTR=1)	Slave Mode (MSTR=0)
0	0	PM3 not used with SPI	PM3 is $\overline{SS}$ input
0	1	PM3 not used with SPI	PM3 is $\overline{SS}$ input
1	0	PM3 is $\overline{SS}$ input w/MODF	PM3 is $\overline{SS}$ input
1	1	PM3 is $\overline{SS}$ output	PM3 is $\overline{SS}$ input

Pin Mode	MSTR	SPC0	BIDIROE	MISO	MOSI
Normal	1	0	X	Master In	Master Out
Bidirectional	1	1	0	MISO not used	Master In
			1		Master I/O
Normal	0	0	X	Slave Out	Slave In
Bidirectional	0	1	0	Slave In	MOSI not used
			1	Slave I/O	

Concluding Remarks

- Serial I/O is very common
  - USB is obviously everywhere
  - SPI & SCI are more prevalent in embedded systems
    - » primarily because it's low cost
    - » most controllers support this
      - your kits support both
    - » difference is synch vs. asynch
- Too much detail already
  - but advise that you take a look at the DAC application
  - we'll go through the full SCI ritual next lecture
    - » In prep for Lab 8
- SPRING BREAK
  - hope you have some fun
  - hope I can catch up