EEC 417/517
Embedded Systems
Cleveland State University

Lab 6

Serial Communications,

The Universal Synchronous-Asynchronous Receiver-Transmitter (USART) Module,

The RS232 Protocol Standard

Dan Simon

Rick Rarick

Spring 2018

Lab 6 Outline

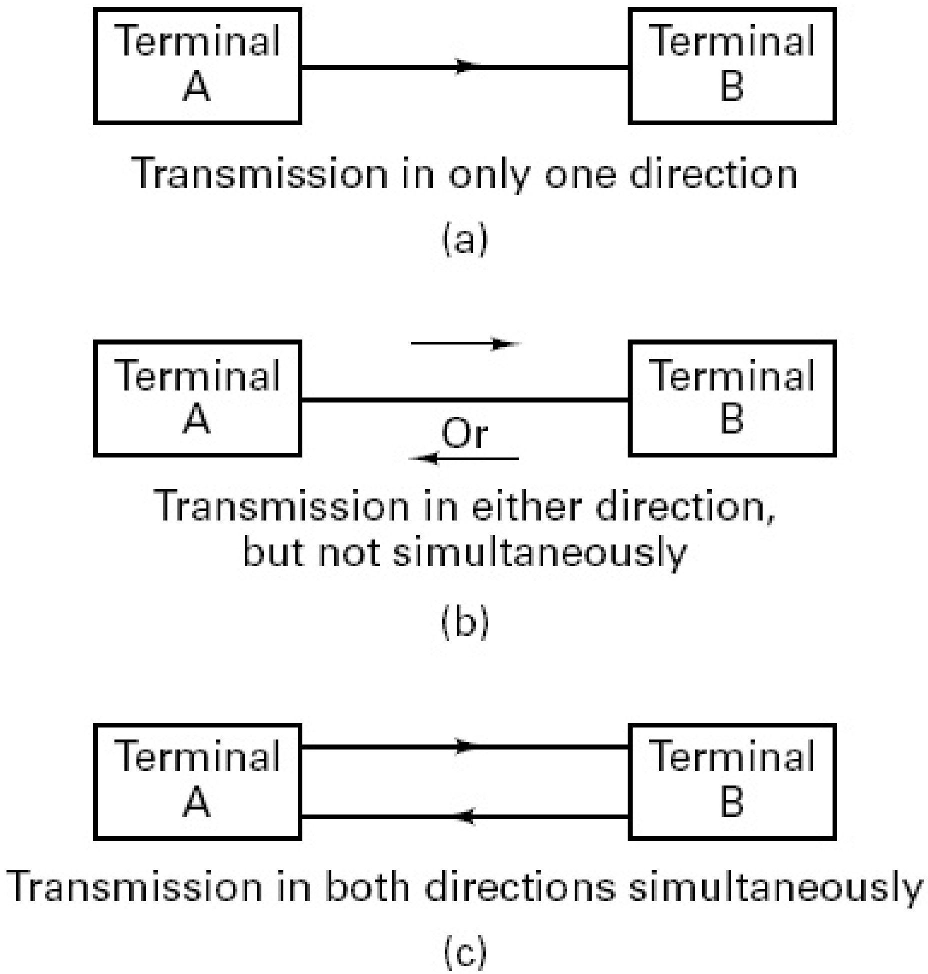
1. **Serial Communications Overview**
2. RS232 Communications Protocol Standard
3. Universal Synchronous Asynchronous Receiver Transmitter (USART) Module
4. Lab 6 Setup

Serial Communications

Simplex

Half Duplex

Full Duplex



Garage door opener

Walkie-Talkie

Telephones, cellphones

Serial Communications

Used for communication between electronic devices

a. Keyboard ↔ PC
b. Mouse ↔ PC
c. Modem ↔ PC
d. PC ↔ PC
e. Microcontroller ↔ PC
f. Microcontroller ↔ Microcontroller
g. Instrumentation ↔ PC
h. Microcontroller ↔ Instrumentation

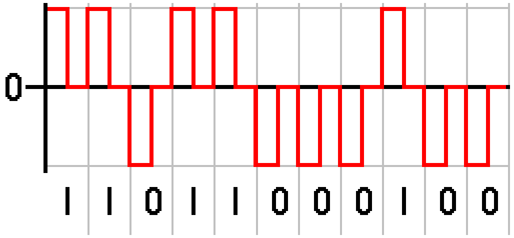
Serial Communications

1. Pulse-code modulation (PCM): Uses electrical pulse waveforms to represent binary digits.

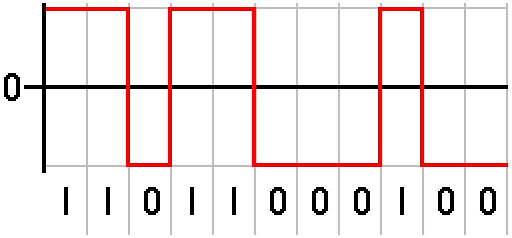
2. Many different types of PCM waveforms (e. g., RZ, NRZ below).

3. PCM waveforms often called bitstreams, digital baseband (< 3 MHz) signals, or linecodes (telecommunications).

Return-to-Zero
(RZ) waveform



Nonreturn-to-Zero
(NRZ) waveform

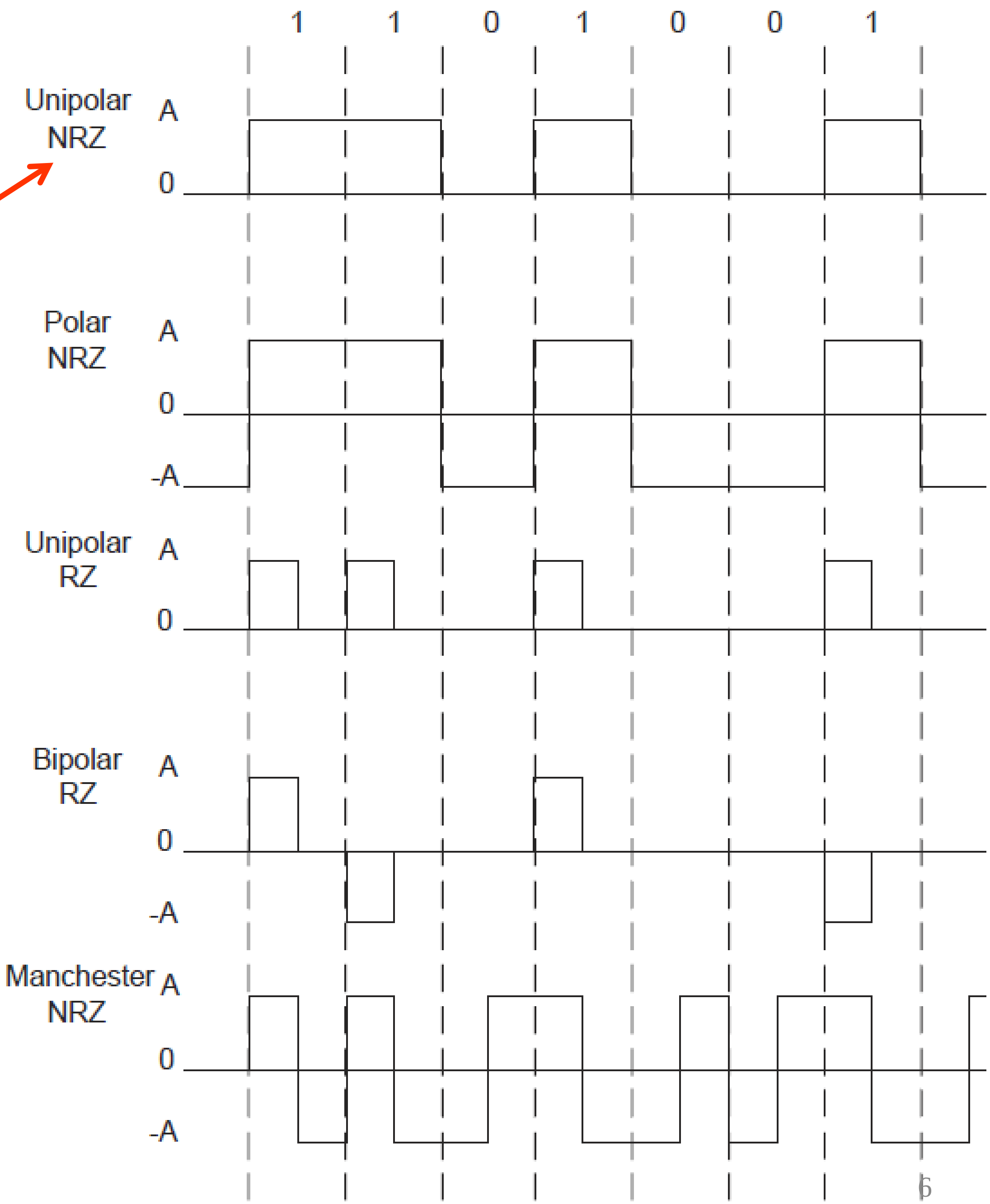


Serial Communications

PIC

Some common PCM waveforms

Ethernet



Serial Communications

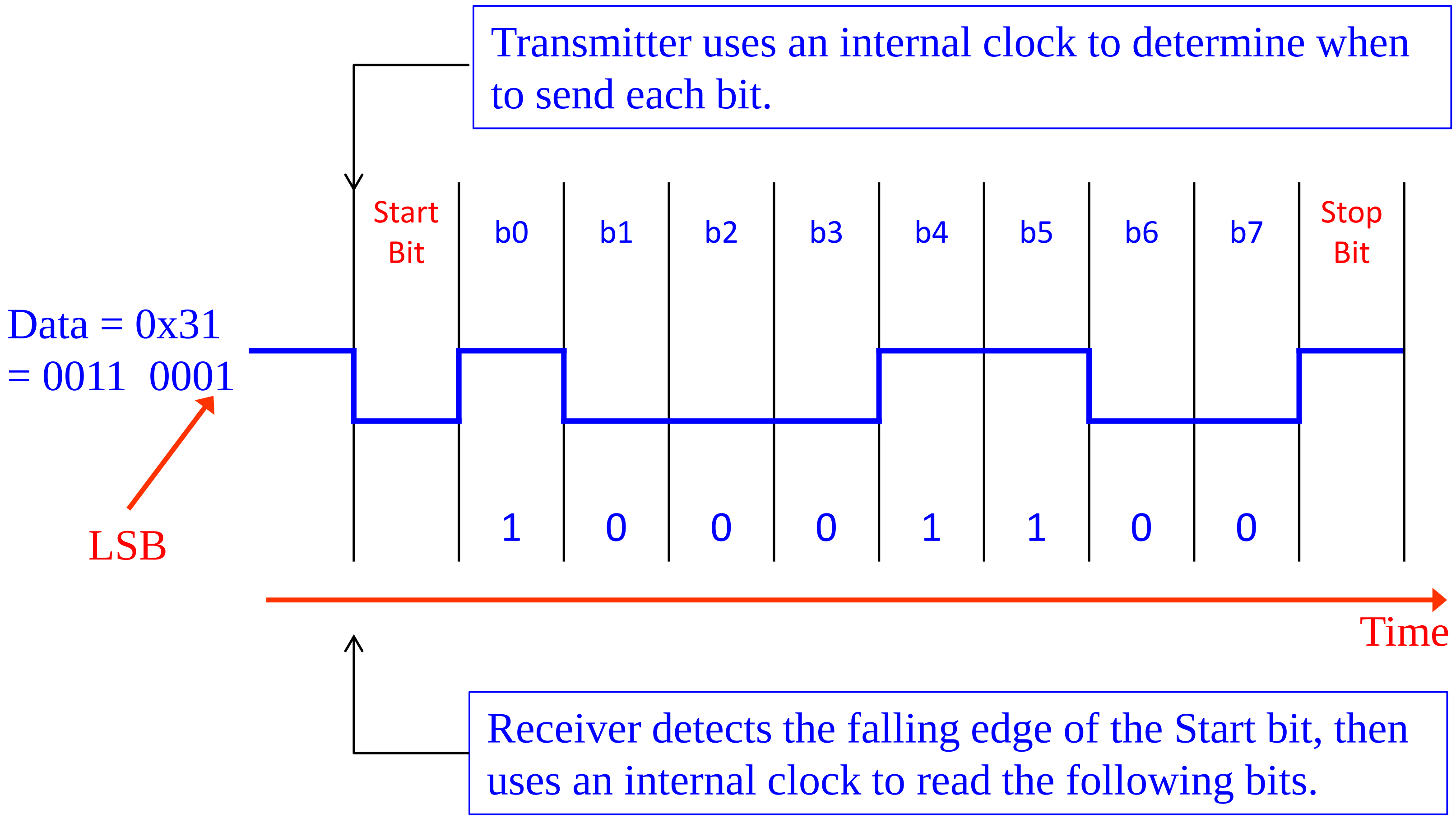
1. Asynchronous serial communication

a. Transmitter and receiver are not synchronized with a common clock signal.
b. Transmitter sends a START bit to inform receiver that data is coming, and a STOP bit to indicate that data transmission is complete.
c. Can transmit and receive simultaneously (full duplex).
d. Communication channel is idle when not transmitting or receiving.

2. Synchronous serial communication

a. Transmitter and receiver are synchronized with a clock signal.
b. Clock signal is always present on communication channel.
c. Can be duplex or half-duplex.

Asynchronous Serial Communications



1. Asynchronous protocols usually send the LSB (b0) first.

2. Internal clocks of transmitter and receiver must have same frequency (baud rates).

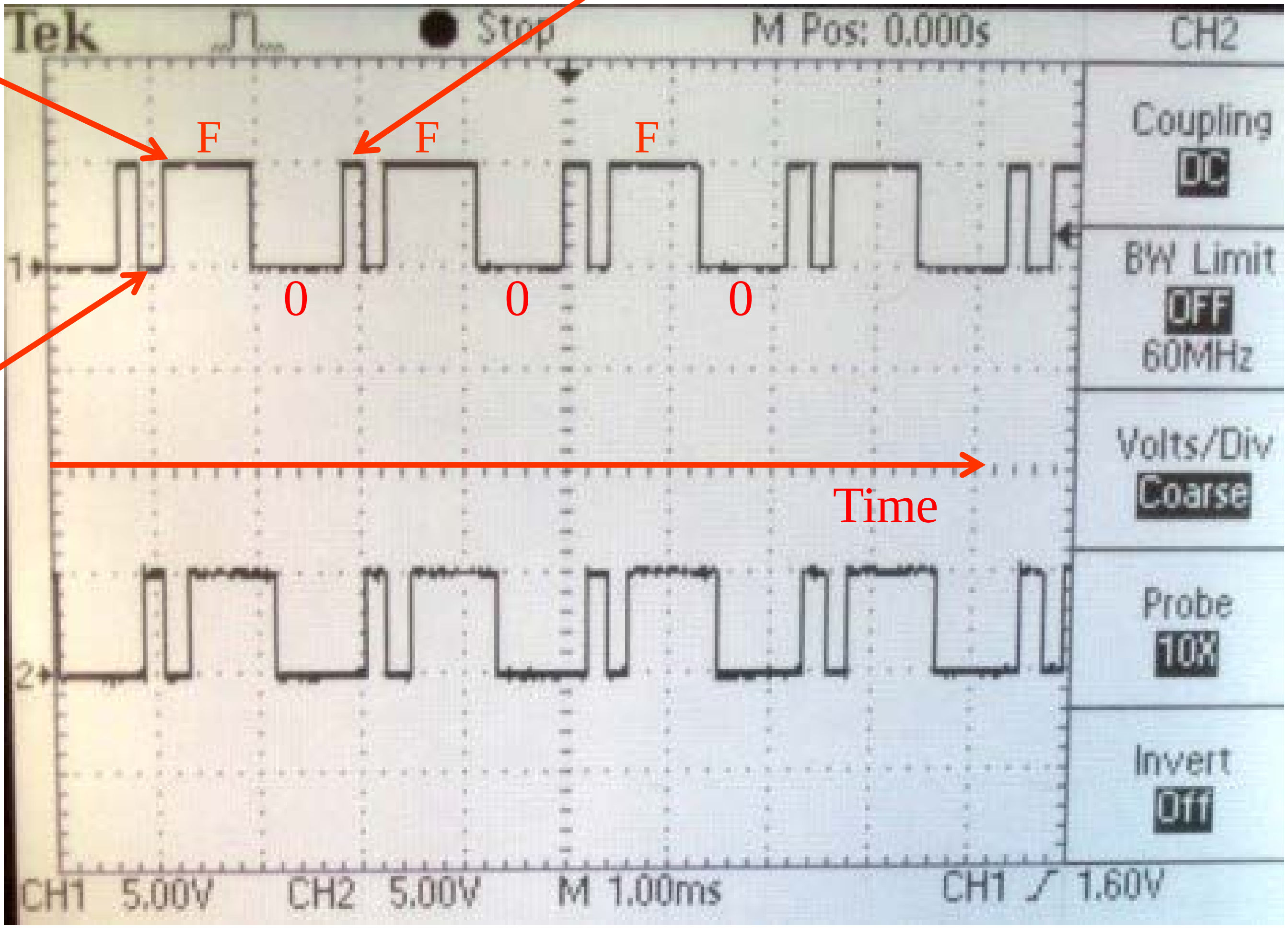
Asynchronous Serial Communications

LSB transmitted first

Stop bit

Transmitted waveform:
0x0F = 0000 1111

Start bit



Received waveform:
0x0F

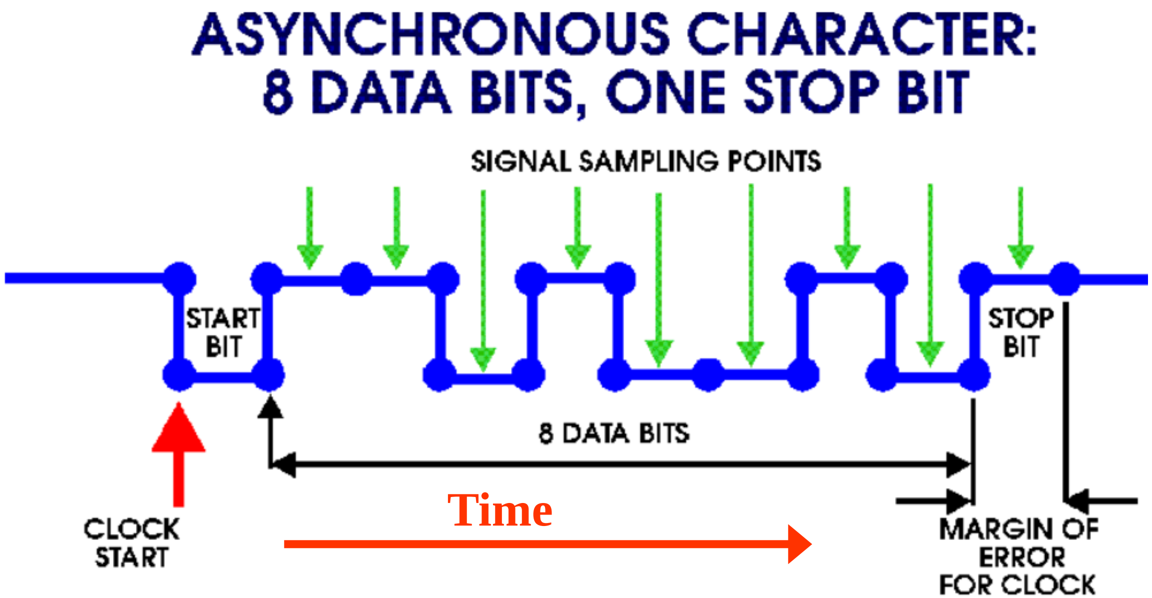
Asynchronous Serial Communications

Transmitter

Receiver

Transmit LSB first:

0x4B = 0100 1011

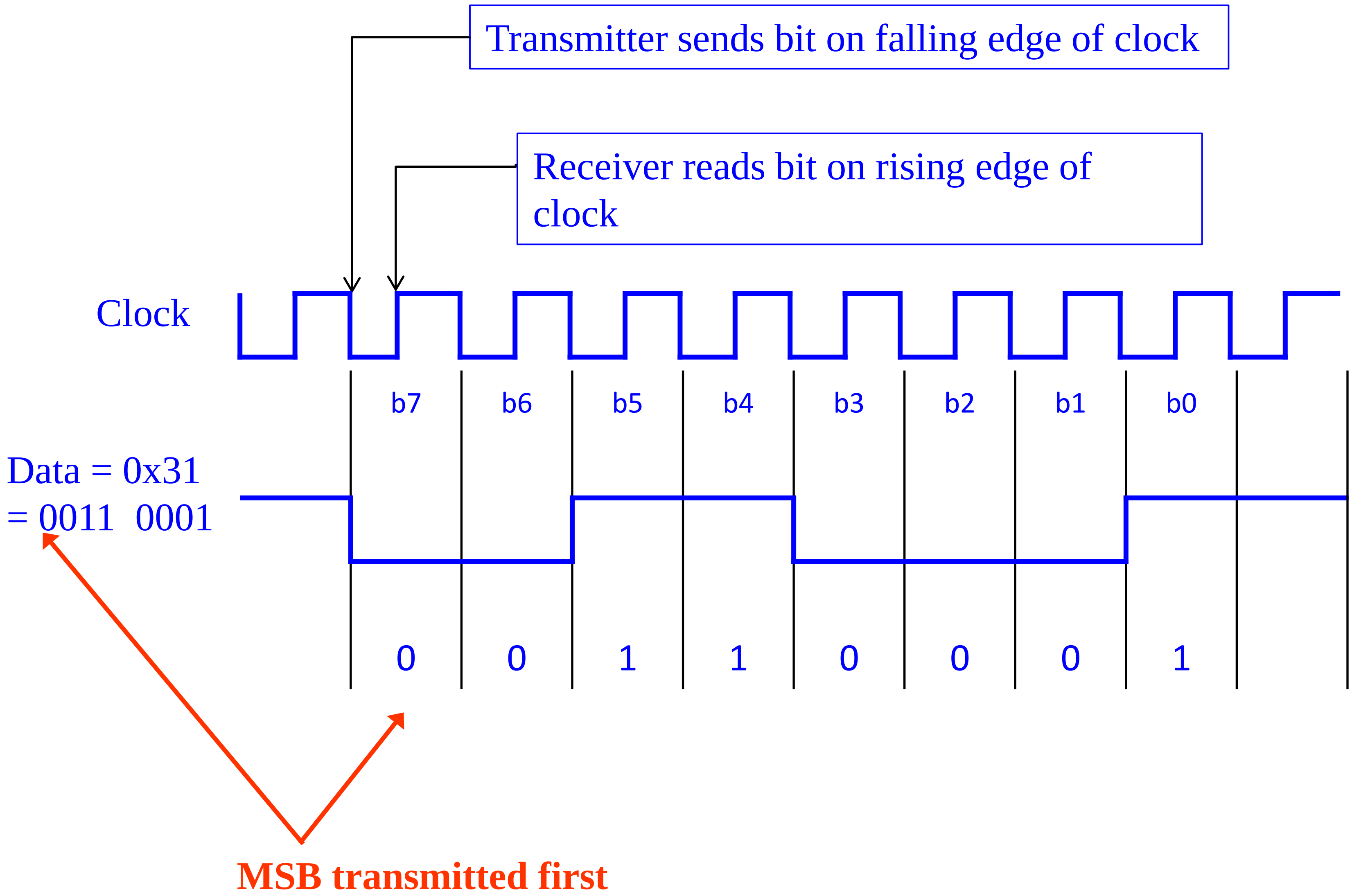


Receive LSB first, then shift right:

0 1 0 0 1 0 1 1

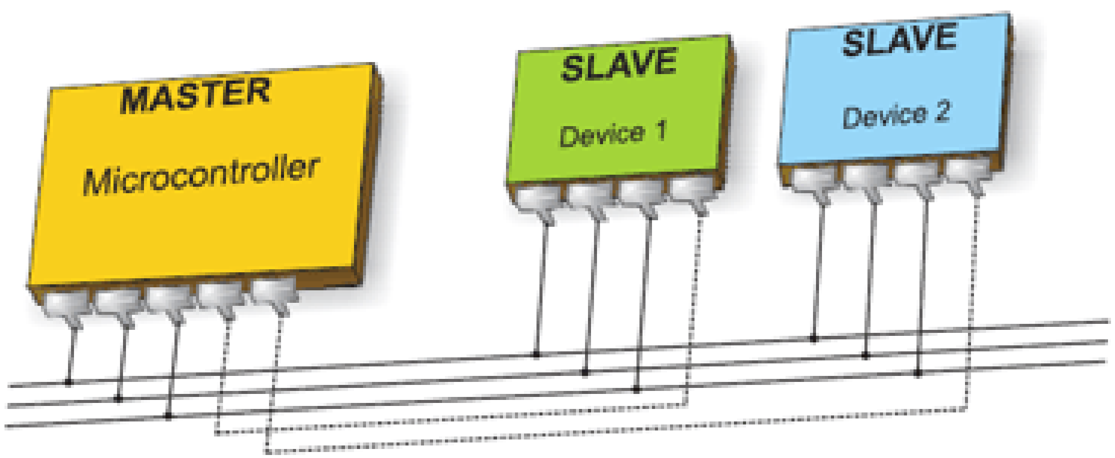
4 B

Synchronous Serial Communications



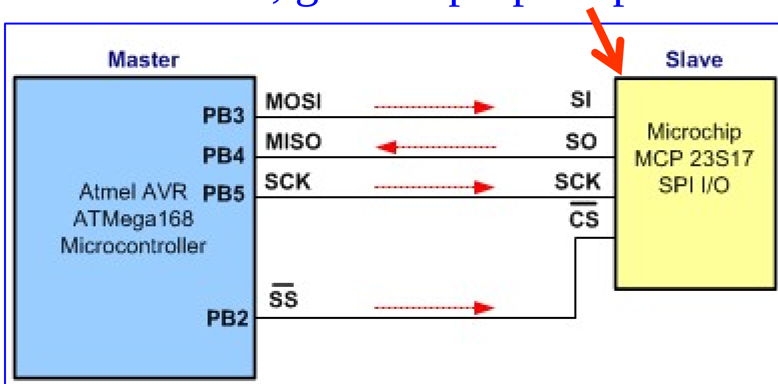
Synchronous Serial Communication

1. One of the communicating devices is designated as the **master** device.
2. The master device supplies the synchronizing clock signal.
3. The second device is designated as the **slave** device.
4. The slave device synchronizes with the master device using the master's clock.
5. Multiple slave devices possible.



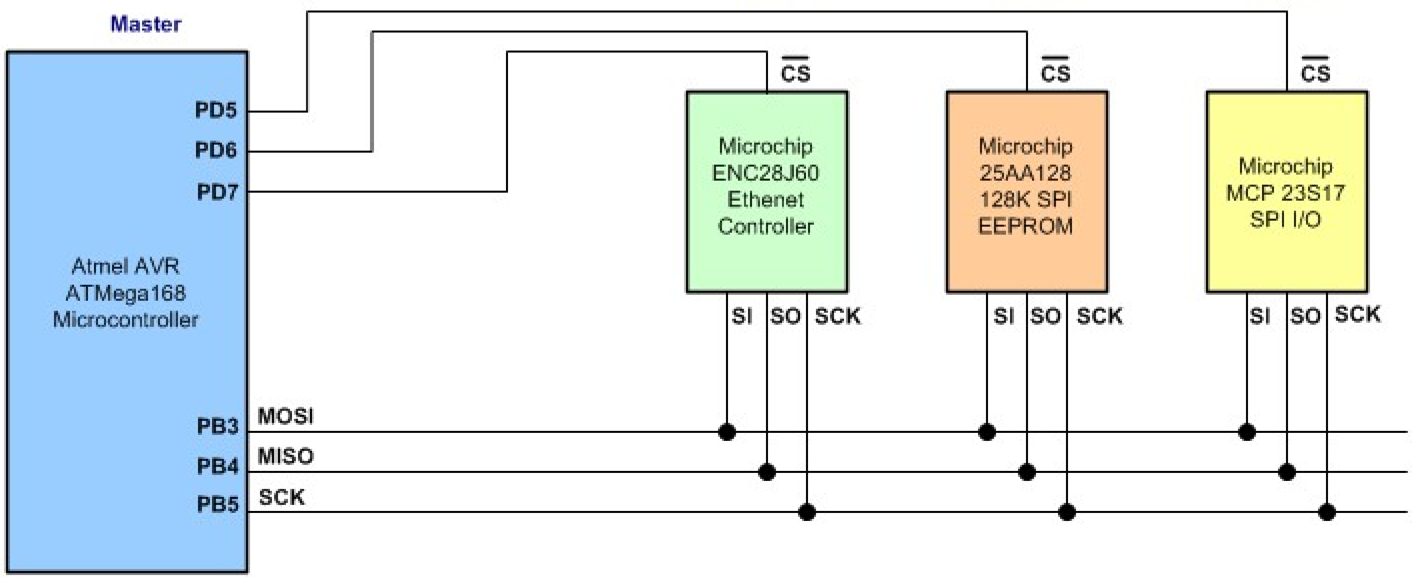
Synchronous Serial Communication

16-bit, general purpose parallel I/O expansion device



Pin Name	Description
MOSI	Master Out Slave In
MISO	Master In Slave Out
SCK	Synchronous Clock
SS	Slave Select (active Low)
CS	Chip Select (active low)
SI	Slave In
SO	Slave Out

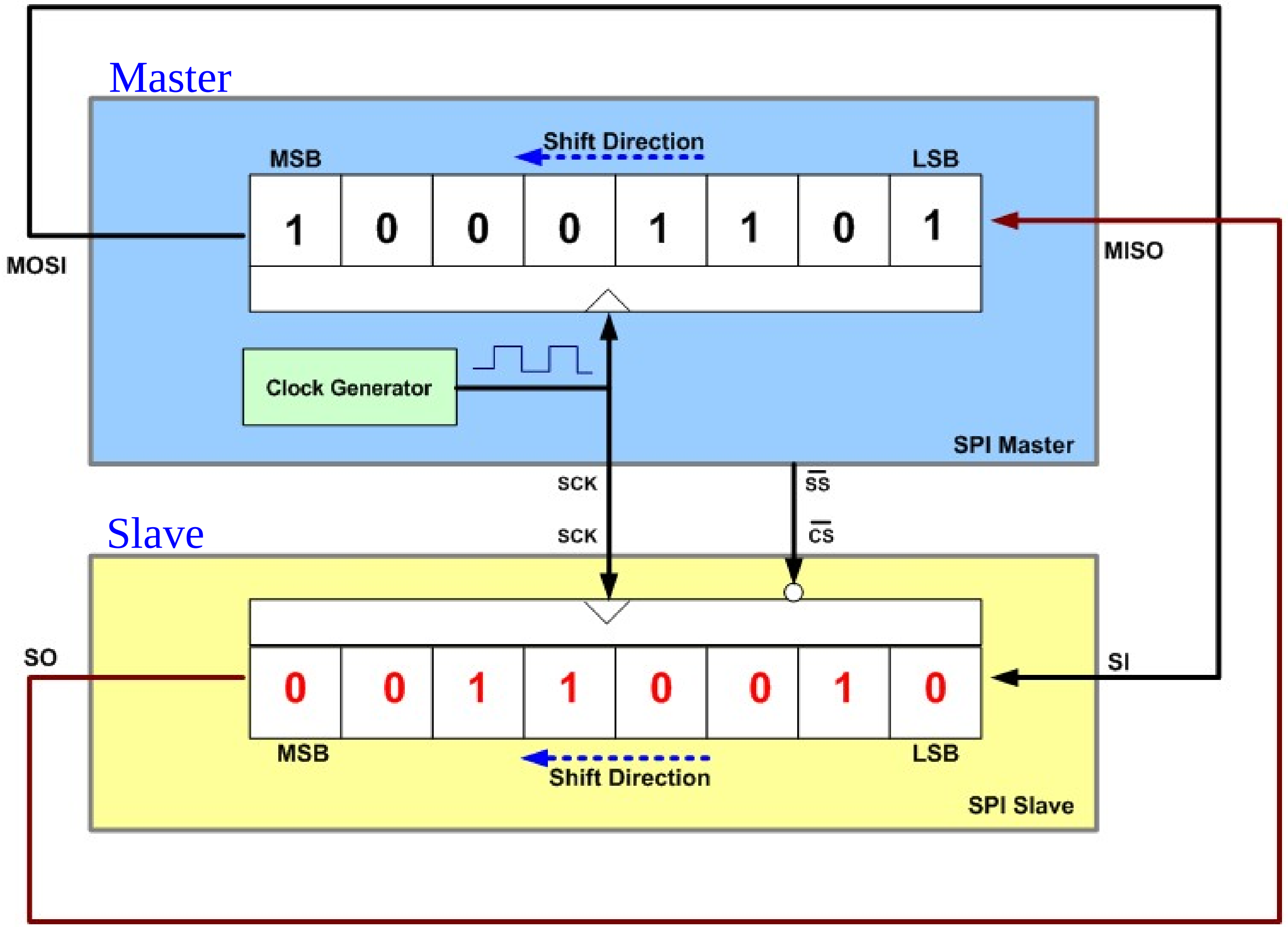
Typical SPI Master and SPI Slave Device Connection



Typical SPI Master with Multiple SPI Slave Device Connection

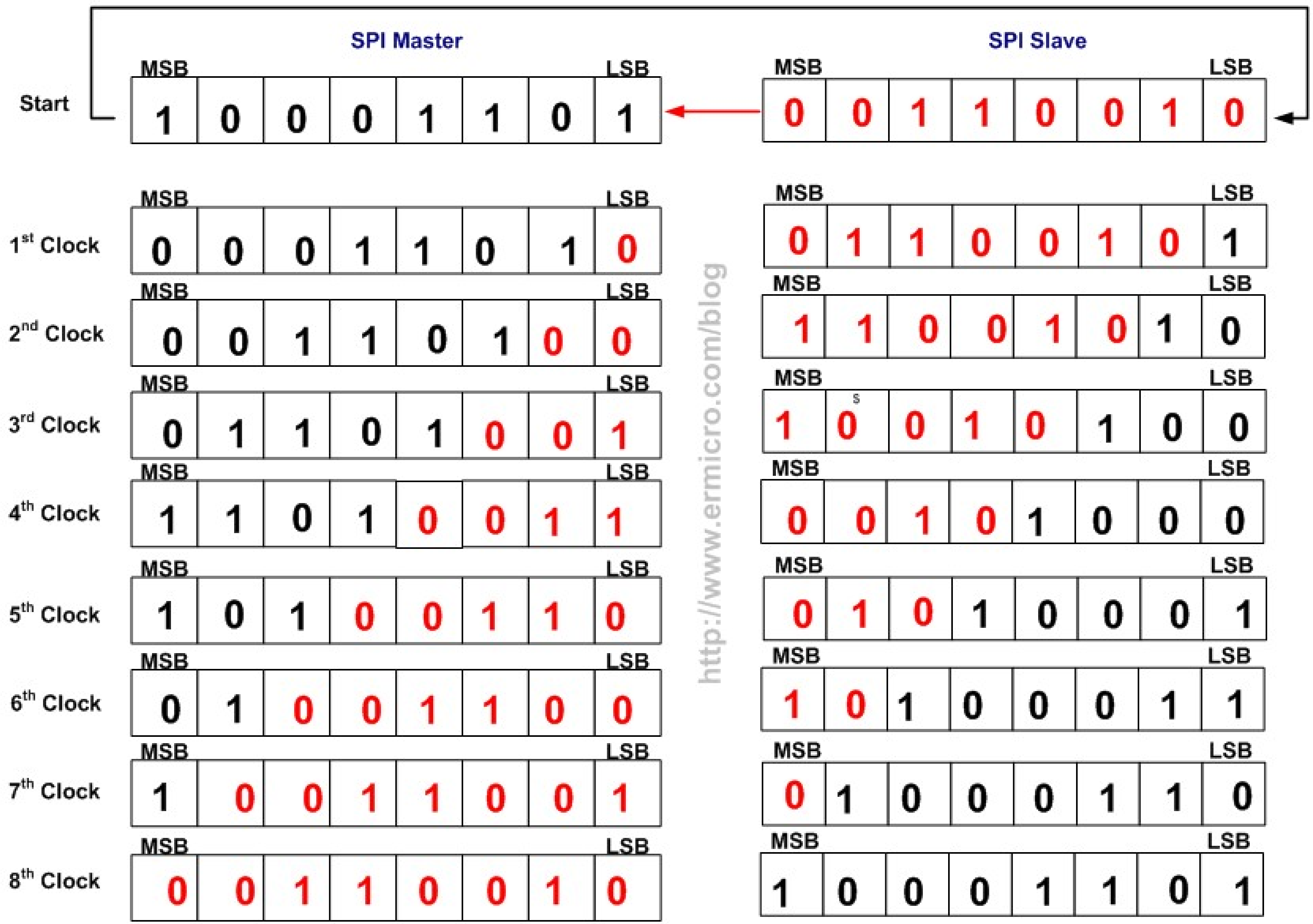
Synchronous Serial Communication

http://www.ermicro.com/blog



SPI Master and Slave Interconection

Synchronous Serial Communication



SPI Master and Slave Data Transfer Diagram

Serial Communications Standards (Partial List)

There are many more standards for serial communications

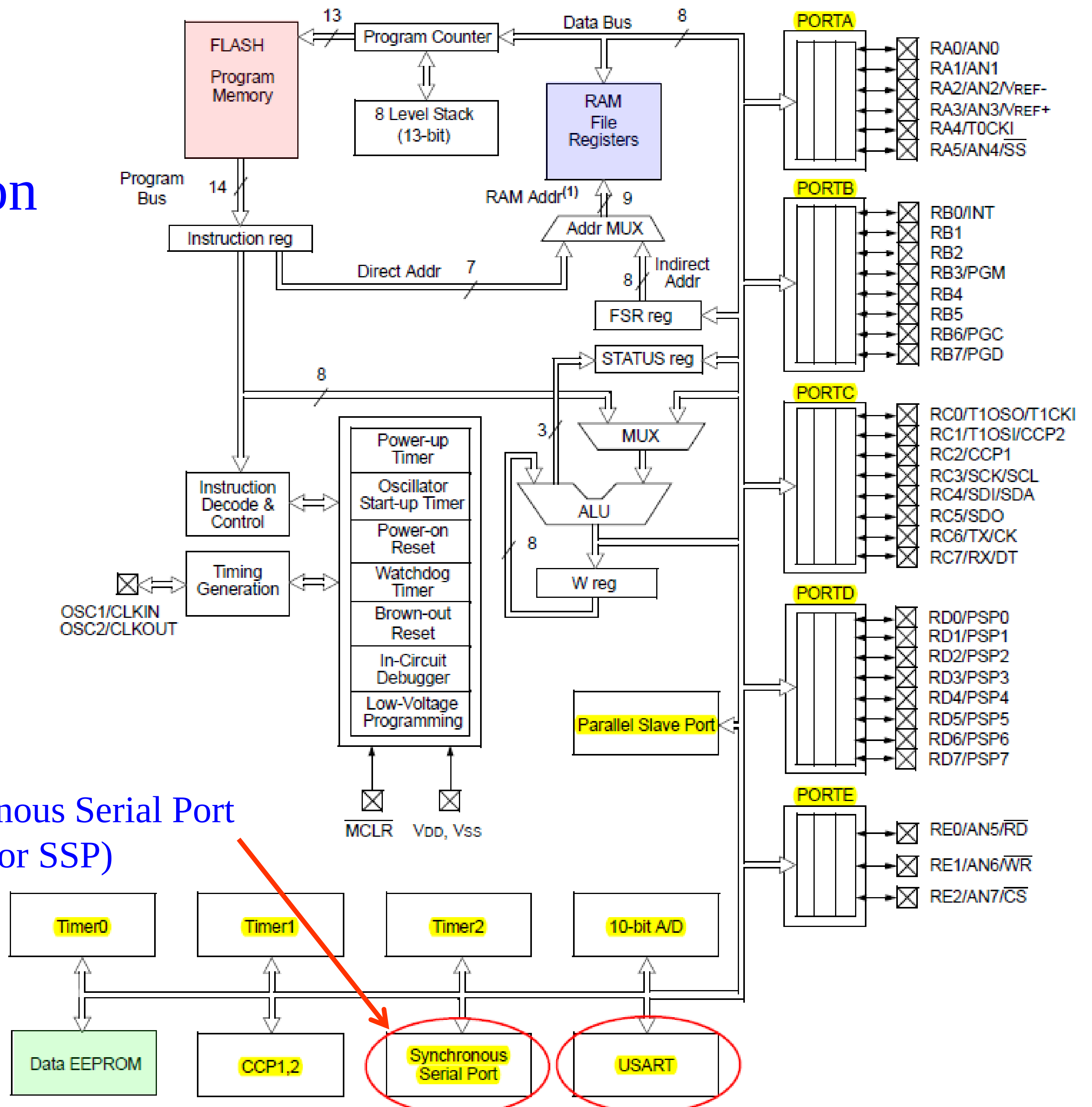
- Morse code telegraphy
- RS-232 (low-speed, implemented by serial ports)
- RS-422
- RS-423
- RS-485
- I²C
- SPI
- ARINC 818 Avionics Digital Video Bus
- Universal Serial Bus (moderate-speed, for connecting peripherals to computers)
- FireWire
- Ethernet
- Fibre Channel (high-speed, for connecting computers to mass storage devices)
- InfiniBand (very high speed, broadly comparable in scope to PCI)
- MIDI control of electronic musical instruments
- DMX512 control of theatrical lighting
- SDI-12 industrial sensor protocol
- Serial Attached SCSI
- Serial ATA
- SpaceWire Spacecraft communication network
- HyperTransport
- PCI Express
- SONET and SDH (high speed telecommunication over optical fibers)
- T-1, E-1 and variants (high speed telecommunication over copper pairs)
- MIL-STD-1553A/B

PIC Serial Hardware

The PIC has two hardware modules for
serial communications.

USART Module
Universal Synchronous-Asynchronous Receiver-Transmitter
Also called Serial Communications Interface (SCI)
MSSP Module
Master Synchronous Serial Port

PIC Serial Communication Modules



Master Synchronous Serial Port
(MSSP or SSP)

Hardware Configurations

The PIC serial communications modules
can be configured for various modes of operation.

<table>
<tr><th colspan="4">USART Module</th></tr>
<tr><td colspan="2">Asynchronous</td><td colspan="2">Synchronous</td></tr>
<tr><td colspan="2"></td><td>Master</td><td>Slave</td></tr>
<tr><th colspan="4">MSSP Module</th></tr>
<tr><td colspan="2">Serial Peripheral Interface
(SPI)</td><td colspan="2">Inter-Integrated Circuit
(I2C)</td></tr>
<tr><td>Master</td><td>Slave</td><td>Master</td><td>Slave</td></tr>
</table>

Lab 6 Outline

1. Serial Communications Overview
2. **RS232 Communications Protocol Standard**
3. Universal Synchronous Asynchronous Receiver Transmitter (USART) Module
4. Lab 6 Setup

RS232 Protocol

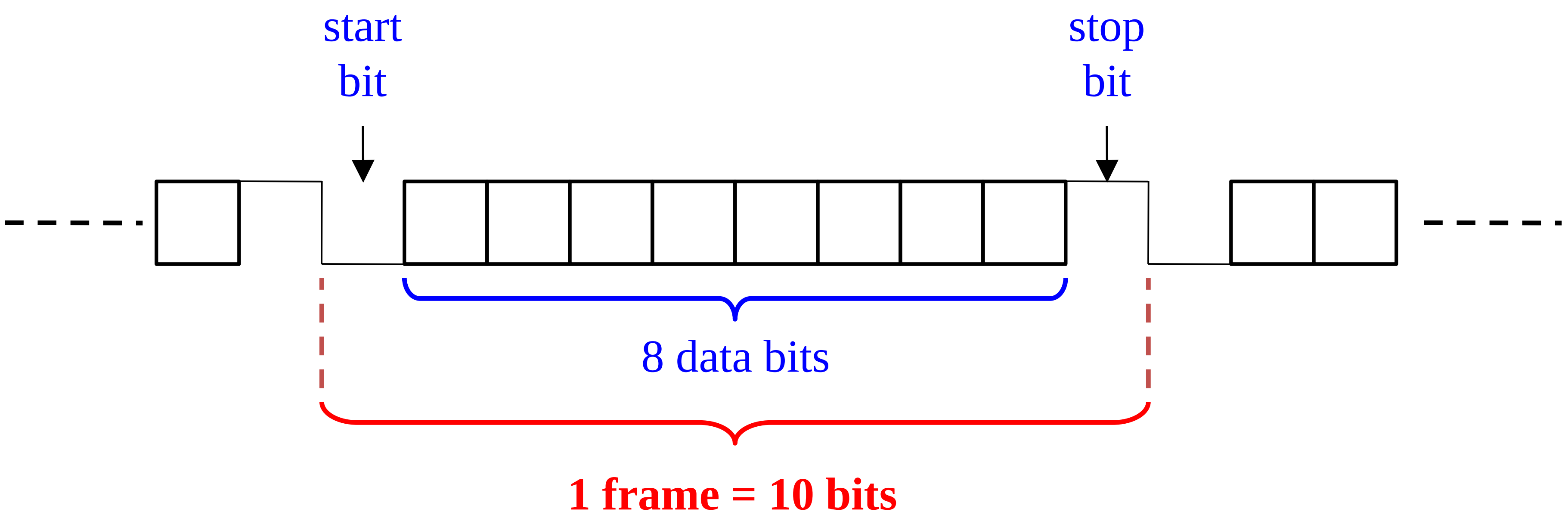
1. In order for two devices to communicate, they must use a common language (**standard** or **protocol**).

2. **RS-232 standard** (Recommended Standard 232)
   A software standard for serial transmission between computers and peripheral devices (modem, mouse, keyboard, etc).

3. The PIC uses the USART hardware module to implement the RS-232 software standard.

4. The RS-232 protocol is still used in many industrial and scientific applications.

RS232 Protocol



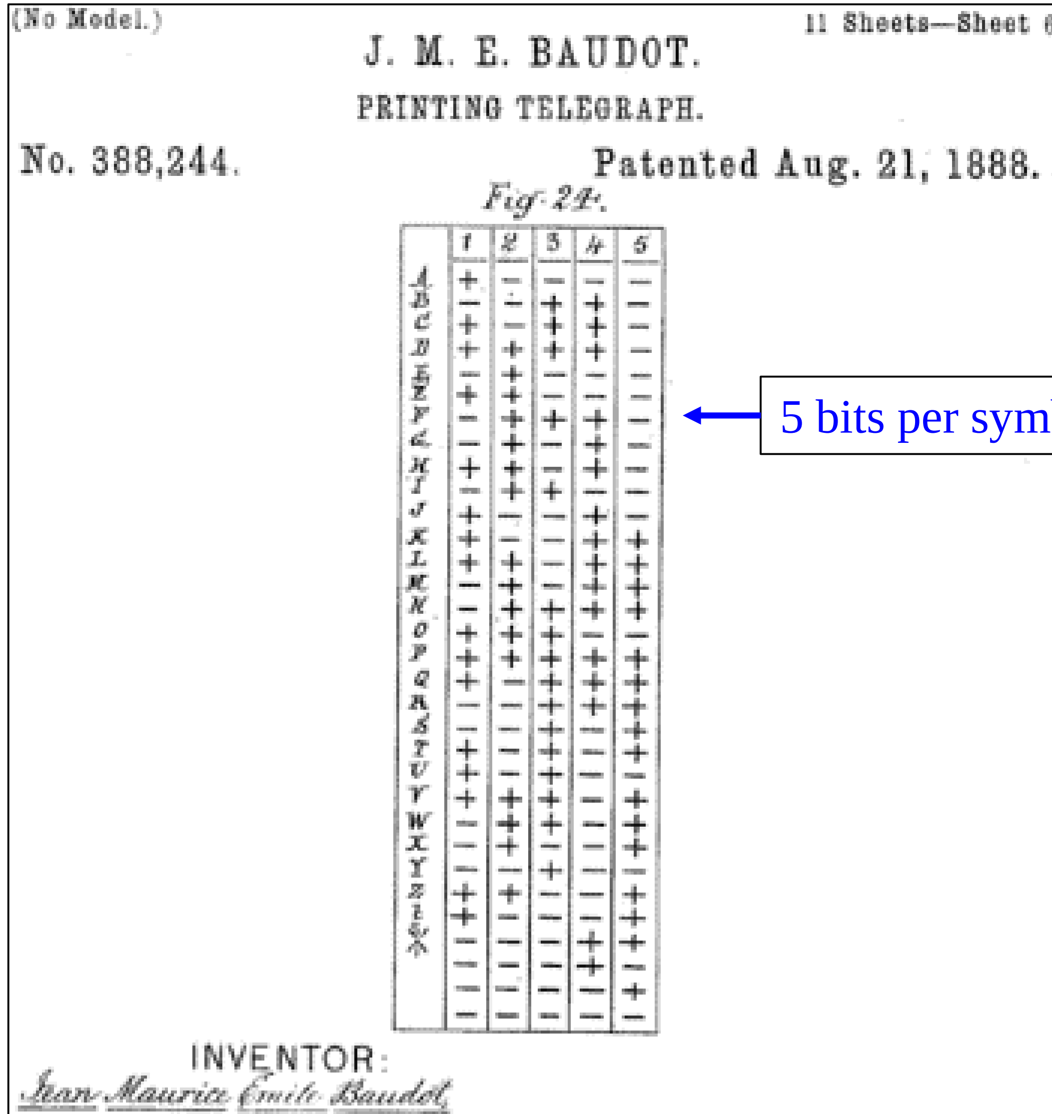
1. The PIC uses 1 start bit, 8 data bits, 1 stop bit, and no parity bits.

2. A 9-data-bit option is available on the 16F877. Other options are available with other devices.

Symbols and Bits

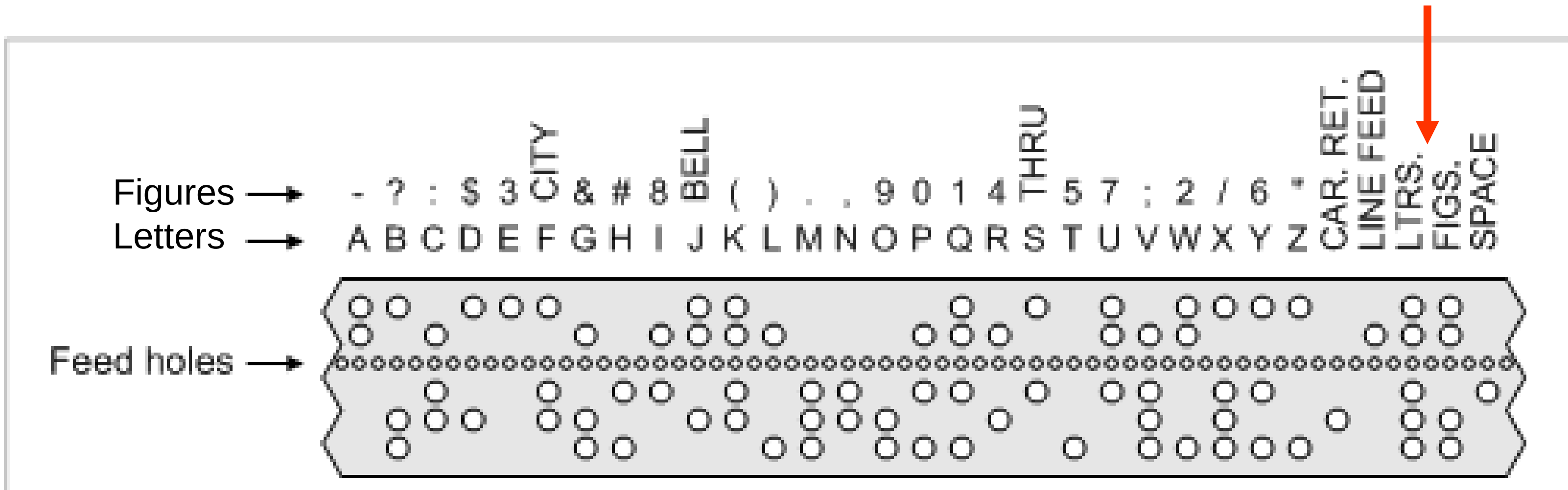
ASCII symbols encoded with the Baudot Code (1870): 5 bits per symbol (originally invented by Gauss and Weber in 1834).

Emile Baudot, 1845-1903, French telegraph engineer.



5 bits per symbol

Bit Rate and Baud Rate (or Symbol Rate)



Paper tape showing the five-bit Baudot Code

1. Tape used to mechanically print symbols.
2. FIGS (Figure Shift) character sent to indicate following symbols are figures until a LTRS (Letter Shift) sent.
3. bit rate (bits / sec) = (bits / symbol) × symbol rate (symbols / sec)
4. Baud rate = symbol rate

RS232 Protocol

1. **Symbol** = group of bits

2. **Baud Rate** (or symbol rate) = symbols per second

   (We will only consider 1-bit symbols)

3. In general, baud ≠ bits per second (bps)
   (But if 1 symbol = 1 bit, then baud = bps)

4. PIC Example: if 1 symbol = 1 bit, a 9600 baud rate gives the transmission time per frame of

$$\frac{10\text{ bits}}{1\text{ frame}} \times \frac{1\text{ s}}{9600\text{ bits}} = 1.04\ \frac{\text{ms}}{\text{frame}}$$

5. The most common RS232 format requires 10 bits to send each byte, so at 9600 baud you can send 960 bytes per second.

RS232 Protocol

7-bit ASCII Table

ASCII = American Standard Code for Information Interchange

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
32	20	Space	64	40	@	96	60	`
33	21	!	65	41	A	97	61	a
34	22	"	66	42	B	98	62	b
35	23	#	67	43	C	99	63	c
36	24	$	68	44	D	100	64	d
37	25	%	69	45	E	101	65	e
38	26	&	70	46	F	102	66	f
39	27	'	71	47	G	103	67	g
40	28	(	72	48	H	104	68	h
41	29	)	73	49	I	105	69	i
42	2A	*	74	4A	J	106	6A	j
43	2B	+	75	4B	K	107	6B	k
44	2C	,	76	4C	L	108	6C	l
45	2D	-	77	4D	M	109	6D	m
46	2E	.	78	4E	N	110	6E	n
47	2F	/	79	4F	O	111	6F	o
48	30	0	80	50	P	112	70	p
49	31	1	81	51	Q	113	71	q
50	32	2	82	52	R	114	72	r
51	33	3	83	53	S	115	73	s
52	34	4	84	54	T	116	74	t
53	35	5	85	55	U	117	75	u
54	36	6	86	56	V	118	76	v
55	37	7	87	57	W	119	77	w
56	38	8	88	58	X	120	78	x
57	39	9	89	59	Y	121	79	y
58	3A	:	90	5A	Z	122	7A	z
59	3B	;	91	5B	[	123	7B	{
60	3C	<	92	5C	\	124	7C	|
61	3D	=	93	5D	]	125	7D	}
62	3E	>	94	5E	^	126	7E	~
63	3F	?	95	5F	_	127	7F	□

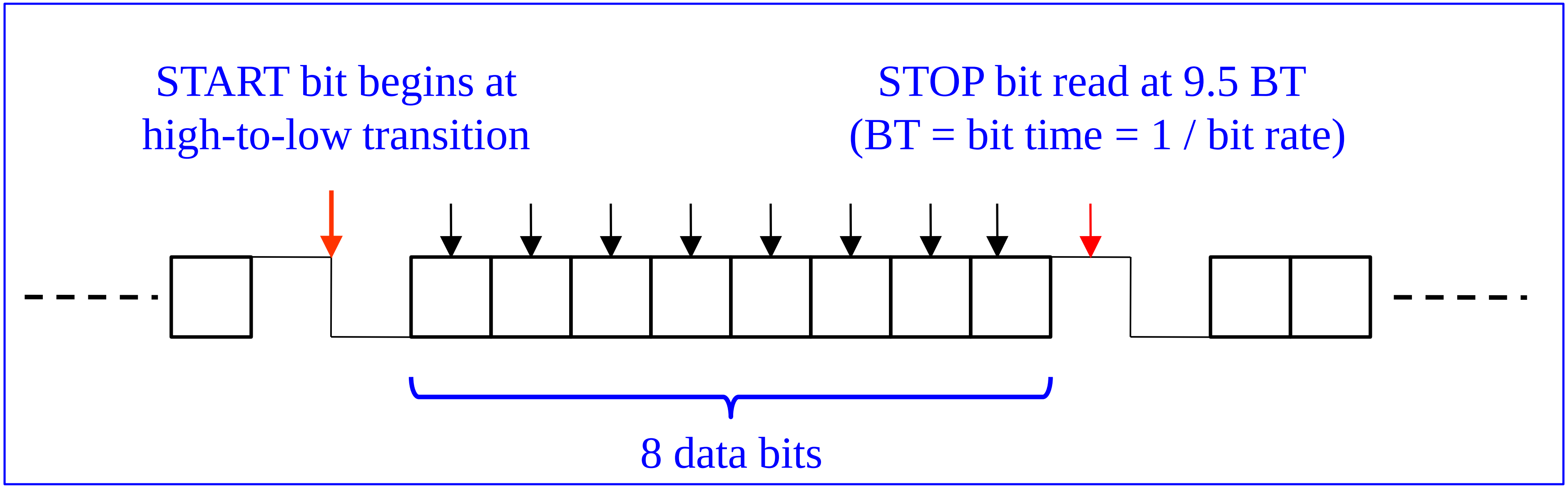
RS232 Protocol

The PIC transmits one ASCII character per frame. Therefore,

$$\text{Character Transmission Rate}\left(\frac{\text{characters}}{\text{s}}\right)=\frac{1}{\text{bits/frame}}\times\frac{1\text{ character}}{1\text{ frame}}\times\text{baud rate}\left(\frac{\text{bits}}{\text{s}}\right)$$

Standard baud rates supported by most serial ports:		
	110	300
	600	1200
	2400	4800
	9600	14400
	19200	28800
	38400	56000
	57600	115200

RS232 Protocol

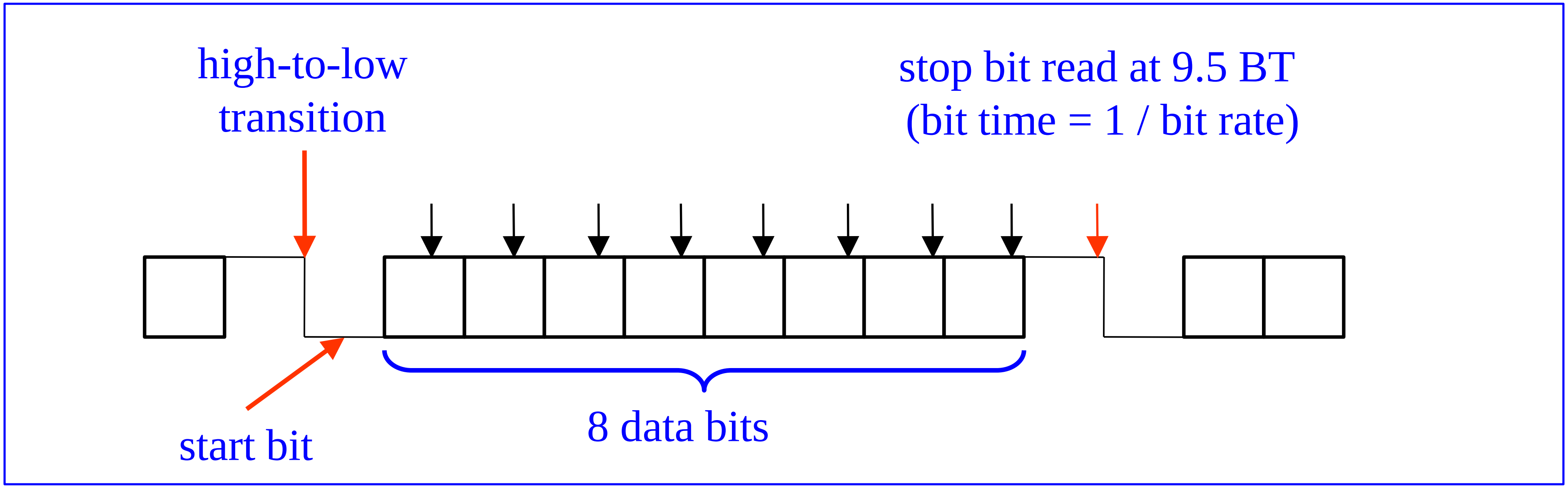


1. A high-to-low transition is the beginning of the START bit.

2. The PIC attempts to read the data bits and the stop bit at the middle of each bit time interval.

3. STOP bit = 0 $\rightarrow$ **framing error (FERR) has occurred**.

RS232 Protocol



1. Suppose the PIC baud generator clock is slow. How much can the clock be in error before communication is lost?
2. The PIC tries to read the stop bit at 9.5 BT. If it is off by more than 0.5 BT, then communication is lost.
3. Max allowable baud rate error = ±0.5 / 9.5 = ±5.3%
4. This is one reason for using a crystal oscillator rather than *RC* oscillator (more accurate).

Lab 6 Outline

1. Serial Communications Overview
2. RS232 Communications Protocol Standard
3. **Universal Synchronous Asynchronous Receiver Transmitter (USART) Module**
4. Lab 6 Setup

USART Modes

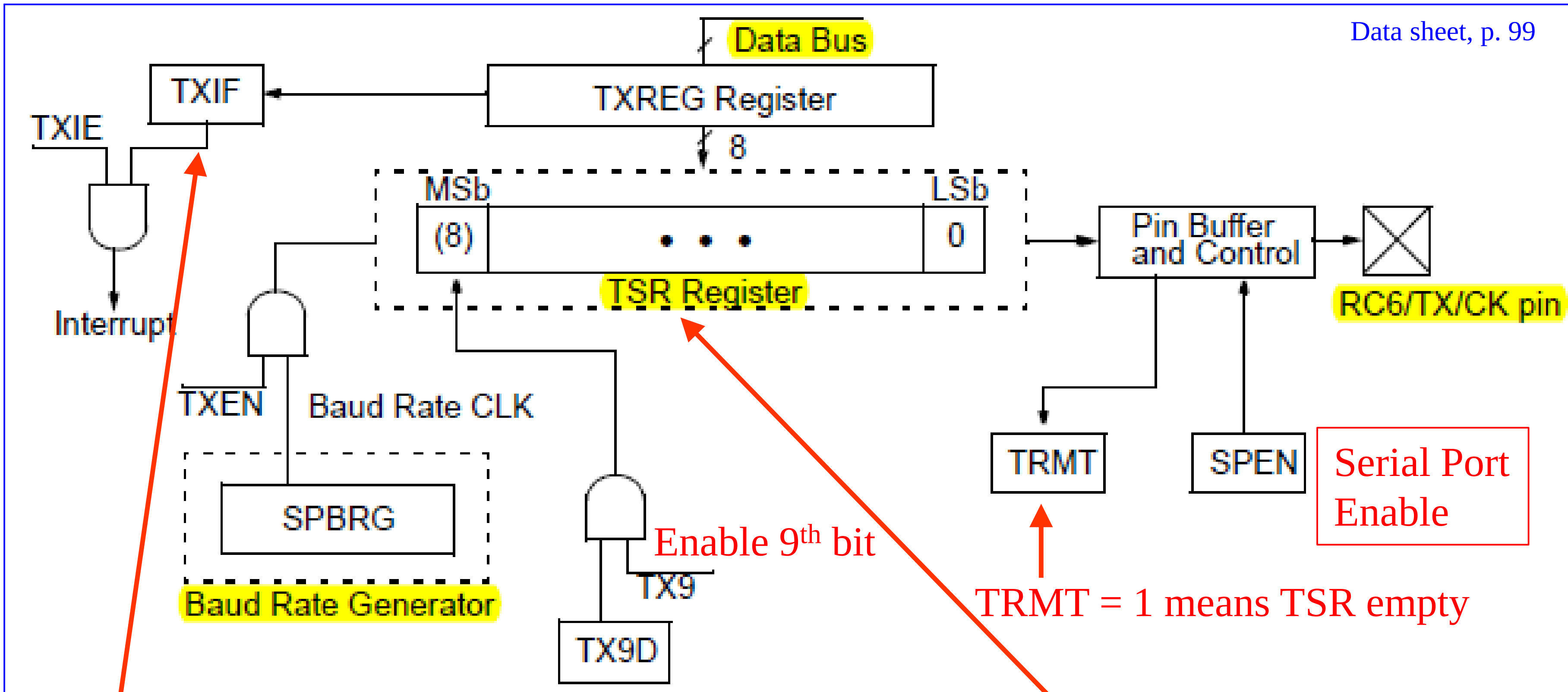
The USART can be configured in the following modes:

1. Asynchronous (full duplex)
2. Synchronous - Master (half duplex)
3. Synchronous - Slave (half duplex)

We will only use the asynchronous mode in lab06.

USART Asynchronous Transmission

Transmission begins automatically after writing to TXREG (provided TXEN is set).



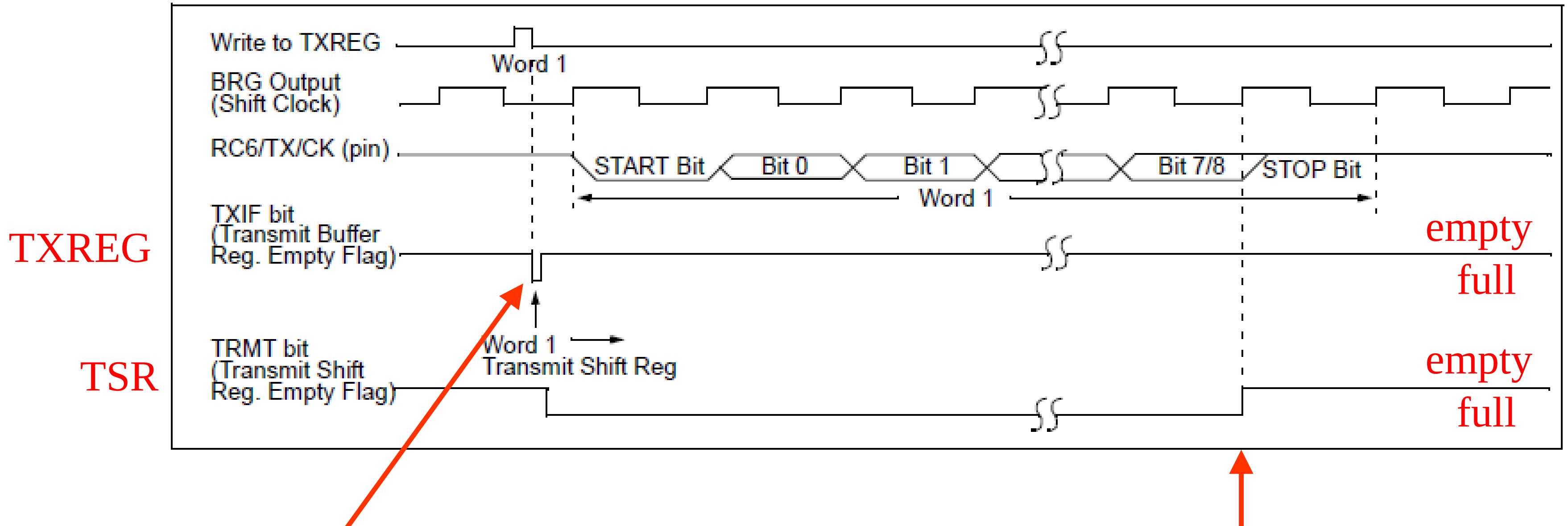
TXIF = 1 when TXREG is empty
TXIF = 0 when TXREG is full (data loaded into TXREG)

Transmit Shift Register

USART Asynchronous Transmission

Transmission begins after writing to TXREG (provided TXEN is set).

FIGURE 10-2: ASYNCHRONOUS MASTER TRANSMISSION



TXIF= 0 (full) for one instruction cycle.

TRMT = 1 means transmission complete

USART Registers

Bank 0	File Address
Indirect addr.(*)	00h
TMR0	01h
PCL	02h
STATUS	03h
FSR	04h
PORTA	05h
PORTB	06h
PORTC	07h
PORTD(1)	08h
PORTE(1)	09h
PCLATH	0Ah
INTCON	0Bh
PIR1	0Ch
PIR2	0Dh
TMR1L	0Eh
TMR1H	0Fh
T1CON	10h
TMR2	11h
T2CON	12h
SSPBUF	13h
SSPCON	14h
CCPR1L	15h
CCPR1H	16h
CCP1CON	17h
RCSTA	18h
TXREG	19h
RCREG	1Ah
CCPR2L	1Bh
CCPR2H	1Ch
CCP2CON	1Dh
ADRESH	1Eh
ADCON0	1Fh
General Purpose Register 96 Bytes	20h 32 – 6Fh
	70h – 7Fh 127

Bank 1	File Address
Indirect addr.(*)	128 80h
OPTION_REG	81h
PCL	82h
STATUS	83h
FSR	84h
TRISA	85h
TRISB	86h
TRISC	87h
TRISD(1)	88h
TRISE(1)	89h
PCLATH	8Ah
INTCON	8Bh
PIE1	8Ch
PIE2	8Dh
PCON	8Eh
	8Fh
	90h
SSPCON2	91h
PR2	92h
SSPADD	93h
SSPSTAT	94h
	95h
	96h
	97h
TXSTA	98h
SPBRG	99h
	9Ah
	9Bh
	9Ch
	9Dh
ADRESL	9Eh
ADCON1	9Fh
General Purpose Register 80 Bytes	A0h 160 – EFh
accesses 70h-7Fh	F0h – FFh 255

Bank 2	File Address
Indirect addr.(*)	256 100h
TMR0	101h
PCL	102h
STATUS	103h
FSR	104h
	105h
PORTB	106h
	107h
	108h
	109h
PCLATH	10Ah
INTCON	10Bh
EEDATA	10Ch
EEADR	10Dh
EEDATH	10Eh
EEADRH	10Fh
General Purpose Register 16 Bytes	110h – 11Fh
General Purpose Register 80 Bytes	120h 288 – 16Fh
accesses 70h-7Fh	170h – 17Fh 383

Bank 3	File Address
Indirect addr.(*)	384 180h
OPTION_REG	181h
PCL	182h
STATUS	183h
FSR	184h
	185h
TRISB	186h
	187h
	188h
	189h
PCLATH	18Ah
INTCON	18Bh
EECON1	18Ch
EECON2	18Dh
Reserved(2)	18Eh
Reserved(2)	18Fh
General Purpose Register 16 Bytes	190h – 19Fh
General Purpose Register 80 Bytes	1A0h 416 – 1EFh
accesses 70h - 7Fh	1F0h – 1FFh 511

REGISTER 10-2: RCSTA: RECEIVE STATUS AND CONTROL REGISTER (ADDRESS 18h)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
bit 7							bit 0

bit 7 **SPEN:** Serial Port Enable bit
1 = Serial port enabled (configures RC7/RX/DT and RC6/TX/CK pins as serial port pins)
0 = Serial port disabled

bit 6 **RX9**: 9-bit Receive Enable bit
1 = Selects 9-bit reception
0 = Selects 8-bit reception

bit 5 **SREN**: Single Receive Enable bit
Asynchronous mode:
Don't care
Synchronous mode - master:
1 = Enables single receive
0 = Disables single receive
This bit is cleared after reception is complete.
Synchronous mode - slave:
Don't care

Data sheet p. 96

bit 4 **CREN**: Continuous Receive Enable bit
Asynchronous mode:
1 = Enables continuous receive
0 = Disables continuous receive
Synchronous mode:
1 = Enables continuous receive until enable bit CREN is cleared (CREN overrides SREN)
0 = Disables continuous receive

bit 3 **ADDEN:** Address Detect Enable bit
Asynchronous mode 9-bit (RX9 = 1):
1 = Enables address detection, enables interrupt and load of the receive buffer when RSR<8> is set
0 = Disables address detection, all bytes are received, and ninth bit can be used as parity bit

bit 2 **FERR**: Framing Error bit
1 = Framing error (can be updated by reading RCREG register and receive next valid byte)
0 = No framing error

bit 1 **OERR**: Overrun Error bit
1 = Overrun error (can be cleared by clearing bit CREN)
0 = No overrun error

bit 0 **RX9D:** 9th bit of Received Data (can be parity bit, but must be calculated by user firmware)

REGISTER 10-1: TXSTA: TRANSMIT STATUS AND CONTROL REGISTER (ADDRESS 98h)

R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D
bit 7							bit 0

bit 7 **CSRC:** Clock Source Select bit

Asynchronous mode:
Don't care

Synchronous mode:
1 = Master mode (clock generated internally from BRG)
0 = Slave mode (clock from external source)

bit 6 **TX9**: 9-bit Transmit Enable bit
1 = Selects 9-bit transmission
0 = Selects 8-bit transmission

bit 5 **TXEN**: Transmit Enable bit
1 = Transmit enabled
0 = Transmit disabled

Note: SREN/CREN overrides TXEN in SYNC mode.

bit 4 **SYNC**: USART Mode Select bit
1 = Synchronous mode
0 = Asynchronous mode

Data sheet p. 95

bit 3 **Unimplemented:** Read as '0'

bit 2 **BRGH**: High Baud Rate Select bit

Asynchronous mode:
1 = High speed
0 = Low speed

Synchronous mode:
Unused in this mode

bit 1 **TRMT**: Transmit Shift Register Status bit
1 = TSR empty
0 = TSR full

bit 0 **TX9D:** 9th bit of Transmit Data, can be parity bit

Baud Rate Generator (BRG)

The Baud Rate Generator can run in two modes:

- Low speed: TXSTA<BRGH> = TXSTA<2> = 0

$$\boxed{\text{Baud Rate} = \frac{F_{osc}}{64(\text{SPBRG}+1)} \quad [\text{bits/sec}]}$$

- High speed: TXSTA<BRGH> = TXSTA<2> = 1

$$\boxed{\text{Baud Rate} = \frac{F_{osc}}{16(\text{SPBRG}+1)} \quad [\text{bits/sec}]}$$

The Baud Rate is determined by the value of the BRGH bit and the value we put in SPBRG register.

Baud Rate Example (Low Speed: BRGH = 0)

1. Example: F_{osc} = 10 MHz, desire 9600 baud.
2. $$\text{SPBRG} = \frac{F_{osc}}{64(\text{Baud Rate})} - 1 = \frac{10 \text{ MHz}}{64(9600)} - 1 = 15.3$$
3. Round to the nearest integer (do not truncate)
4. SPBRG = 15
5. Check baud rate error:
6. $$\text{Baud Rate} = \frac{10 \text{ MHz}}{64(15+1)} = 9766 \quad [\text{bits/sec}]$$
7. Error = (9766 – 9600) / 9600 = 1.7 %

Baud Rate Example (High Speed: BRGH = 1)

1. Example: F_{osc} = 10 MHz, desire 9600 baud
2. $$\text{SPBRG} = \frac{F_{osc}}{16(\text{Baud Rate})} - 1 = \frac{10\ \text{MHz}}{16(9600)} - 1 = 64.1$$
3. Round to the nearest integer (do not truncate)
4. SPBRG = 64
5. $$\text{Baud Rate} = \frac{10\ \text{MHz}}{16(64+1)} = 9615 \quad [\text{bits/sec}]$$
6. Error = (9615 – 9600) / 9600 = 0.16 %
7. **In this example, the high speed mode gives a more accurate baud rate.**

Asynchronous Transmission Steps

1. Set BRGH (high/low) and SPBRG (0 - 255) for desired baud rate.

2. RCSTA< SPEN > = 1 (serial port enable
   TXSTA< SYNC > = 0 (asynchronous mode)
   TXSTA< TXEN > = 1 (transmit enable)

3. TRISC< 6 > = 0 (RC6/TX pin = output)

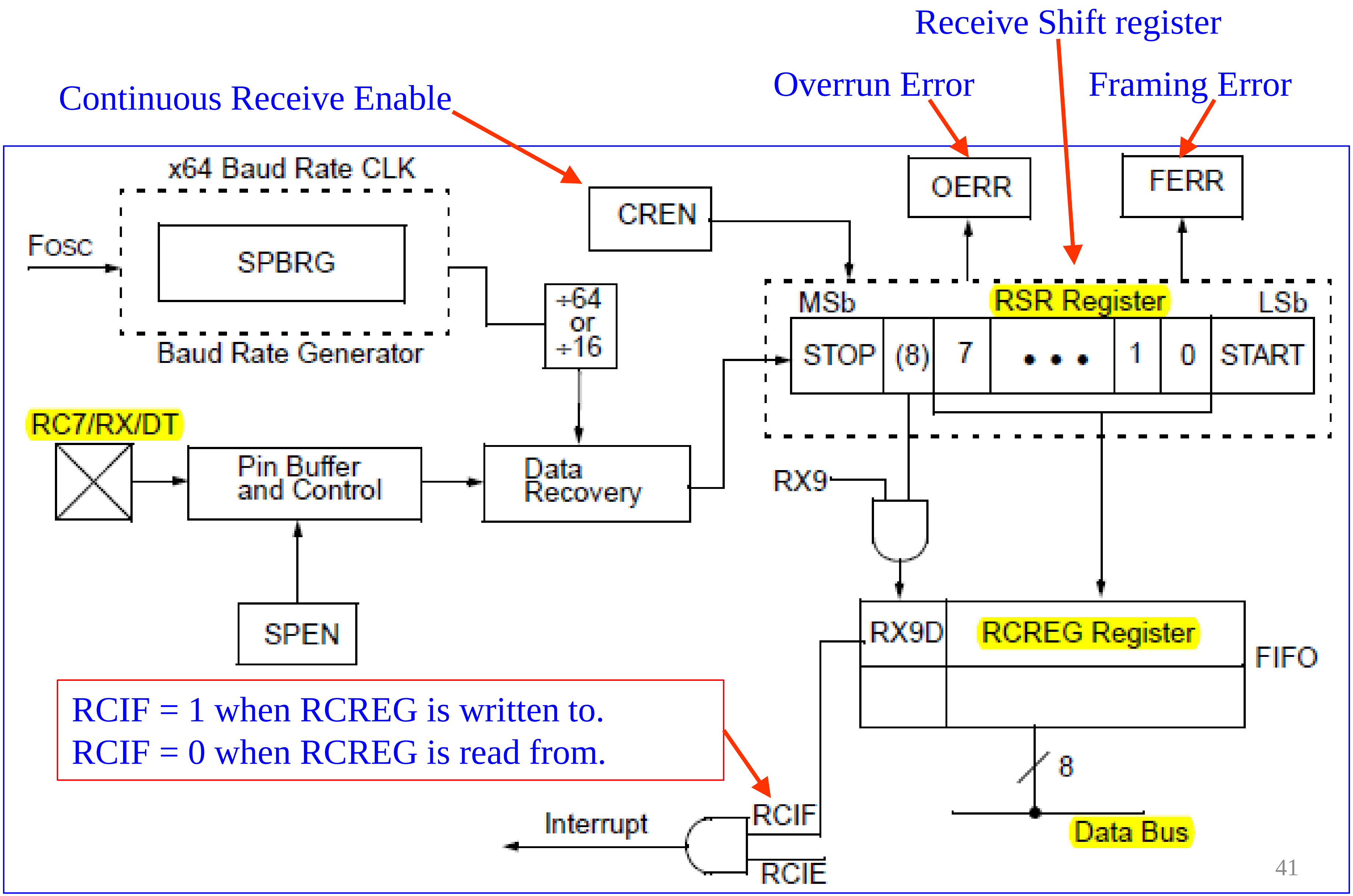
4. TXREG empty: TXIF = 1 (Ready for data)
   TSR empty: TRMT = 1 (Ready for data)

5. Load TXREG with data to transmit.
   TXREG full: TXIF = 0 for one instruction cycle

6. Data automatically moved from TXREG to TSR (TXREG empty: TXIF = 1)
   Transmission automatically starts (TSR not empty: TRMT = 0)

7. Transmission complete (TSR empty: TRMT = 1)

USART Asynchronous Reception



RCIF = 1 when RCREG is written to.
RCIF = 0 when RCREG is read from.

Asynchronous Reception Steps

1. Set BRGH and SPBRG for desired baud rate.

2. RCSTA< SPEN > = 1 (USART enable)
   TXSTA< SYNC > = 0 (asynchronous mode)
   RCSTA< CREN > = 1 (receive enable)

3. TRISC< 7 > = 1 (RC7/RX pin = input) (This is the default.)

4. If required, enable RX interrupt
   INTCON< GIE : PEIE > = 11, PEI1< RCIE > = 1

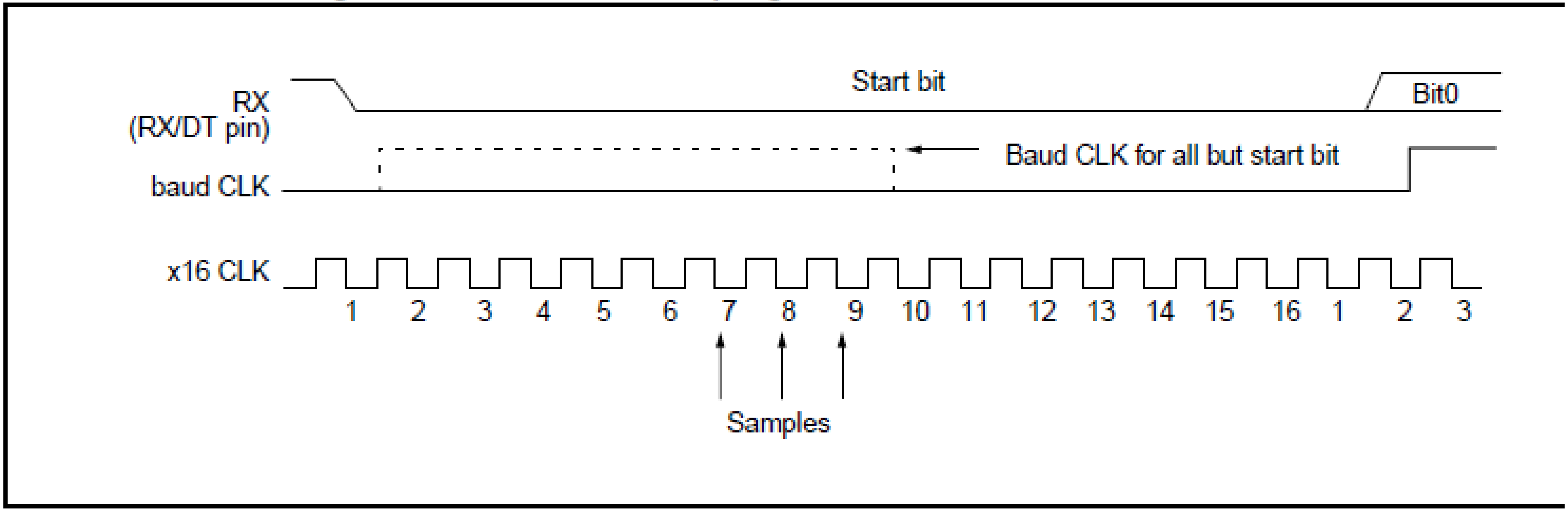
5. When data comes in RX pin, PIR1< RCIF > = 1 and RCREG contains the data.

6. After we read data from RCREG, RCIF is automatically cleared.

USART Asynchronous Reception

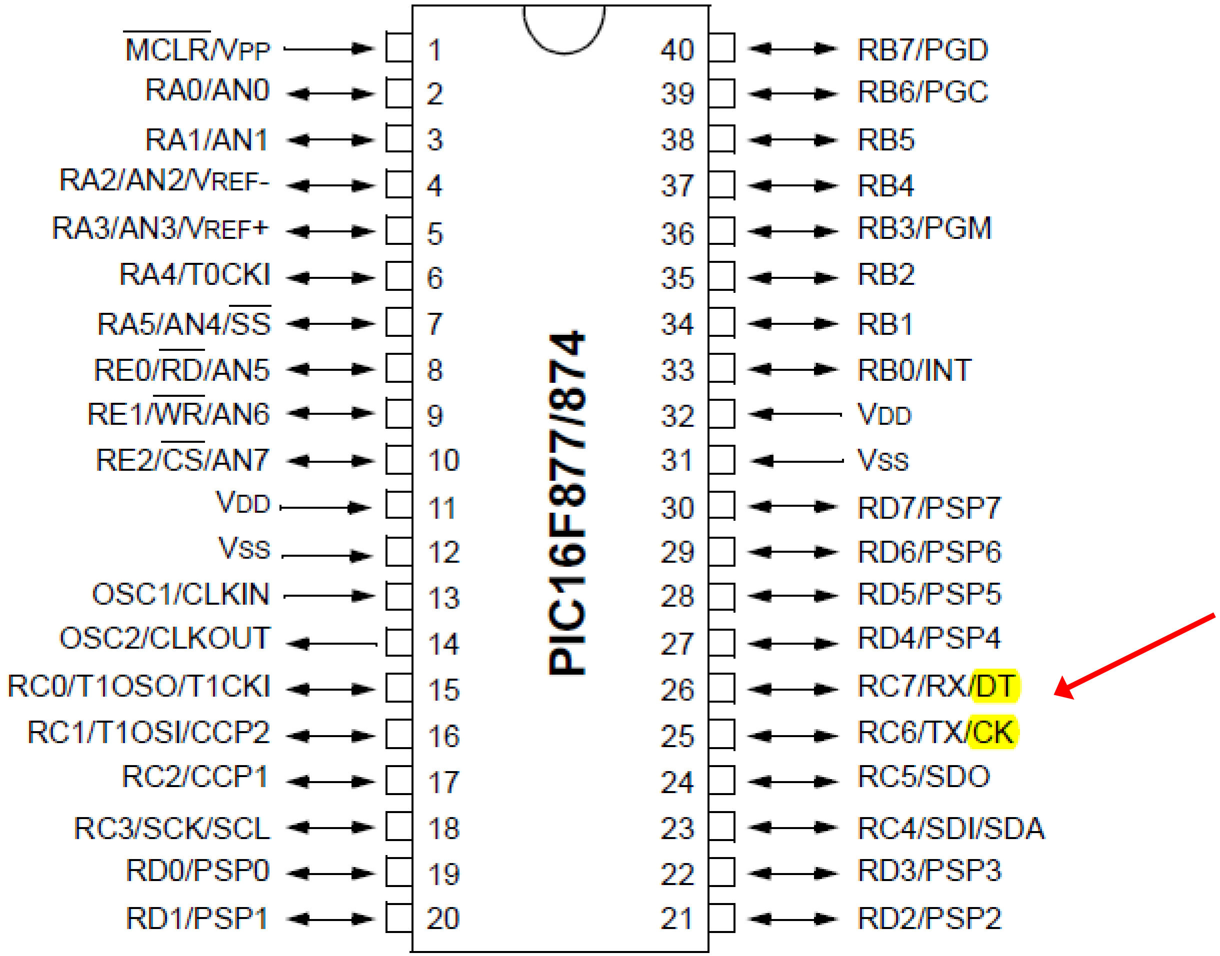
The data on the RX pin is sampled three times by a majority detect circuit to determine if a high or a low level is present on the RX pin.

Figure 18-6: RX Pin Sampling Scheme, BRGH = 0 or BRGH = 1



USART Synchronous Reception

The USART can be used for **synchronous** serial communications by using RC6 for the **clock** signal and RC7 for **data** (transmit or receive, half-duplex).



Lab 6 Outline

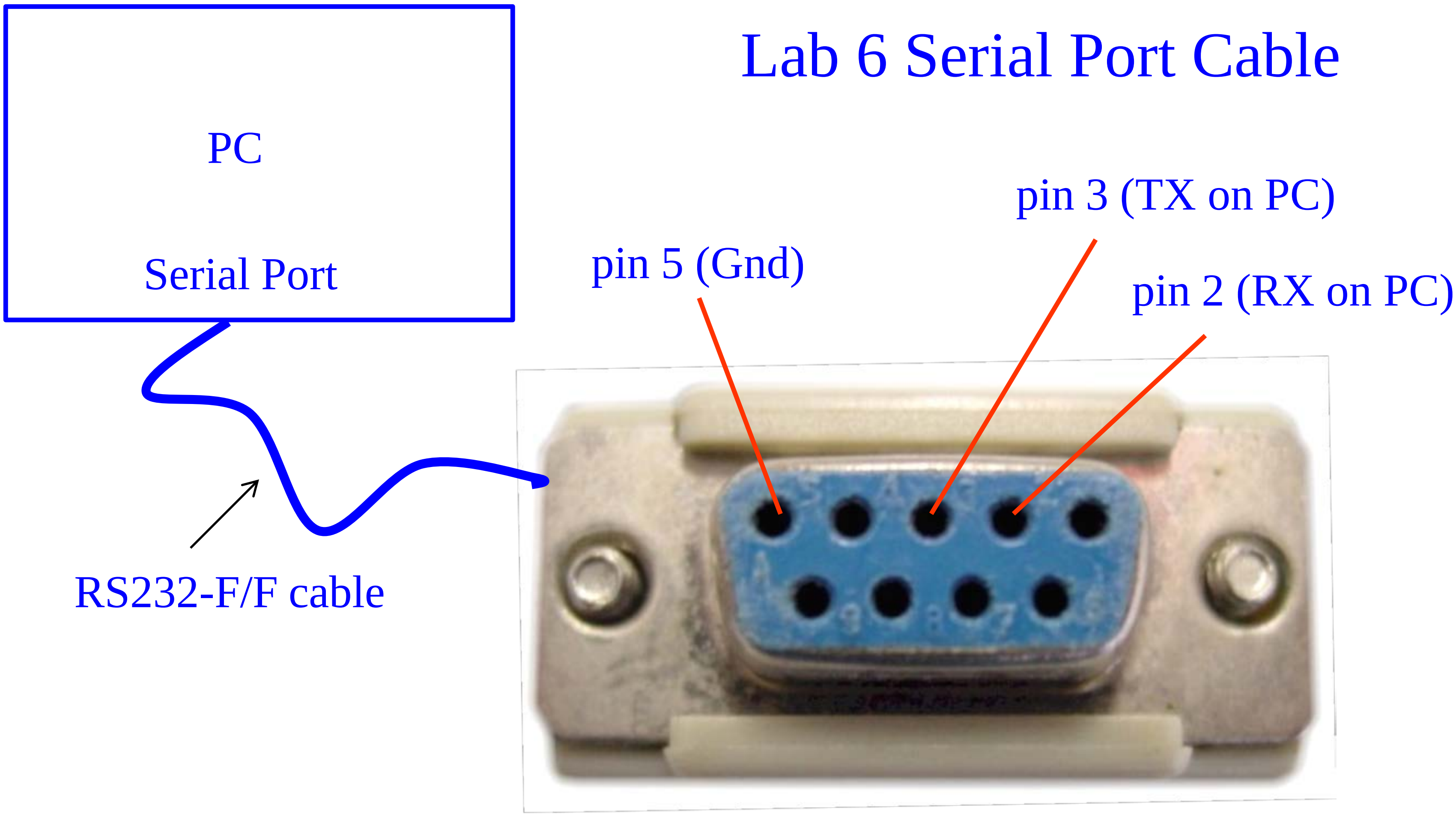
1. Serial Communications Overview
2. RS232 Communications Protocol Standard
3. Universal Synchronous Asynchronous Receiver Transmitter (USART) Module
4. **Lab 6 Setup**

Lab06a.asm

Asynchronous Transmission

Transmit characters A – Z from the PIC, one character per half second, to be received by the Serial Port Program on the PC.

Lab 6 Serial Port Cable



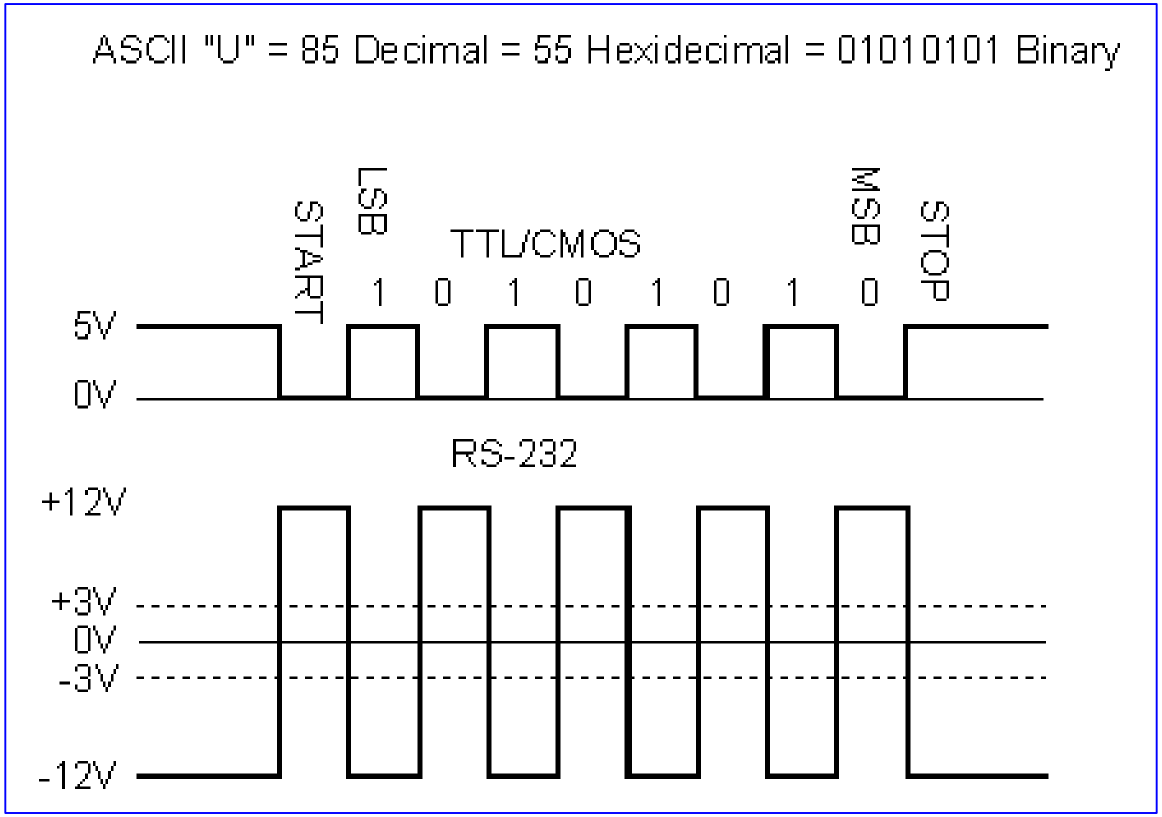
Use the Serial Port Program to run a loop-back test by connecting pins 2 and 3. This verifies that your cable is good and that your PC serial port is working correctly.

Lab 6 Setup

The USART on the PIC cannot communicate directly with the RS232 port on the computer because they operate at different voltage levels and polarities. **Damage will occur if you connect the PC serial port directly to the PIC.**

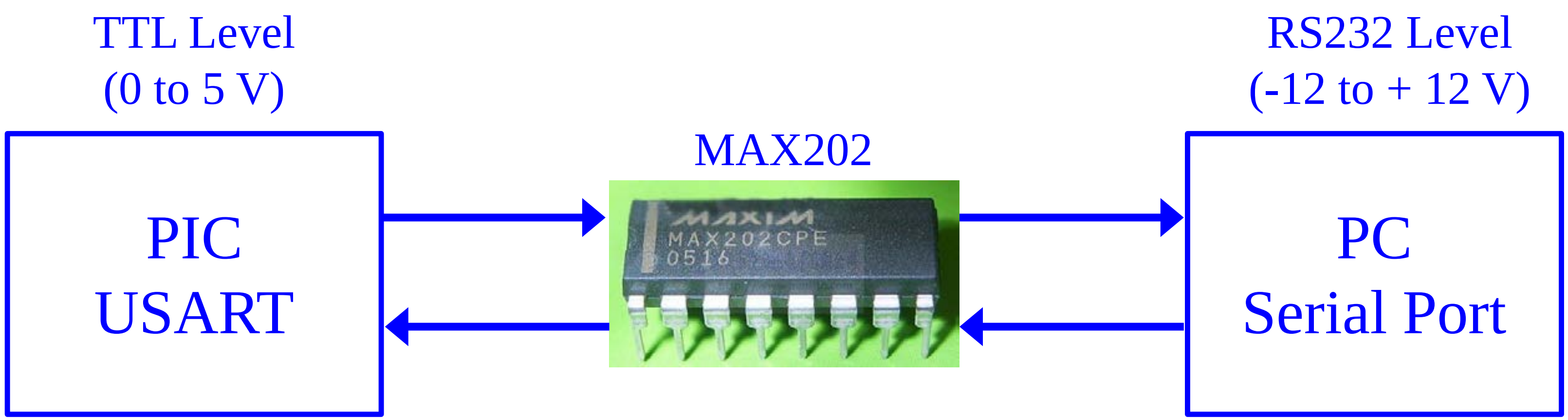
TTL Level

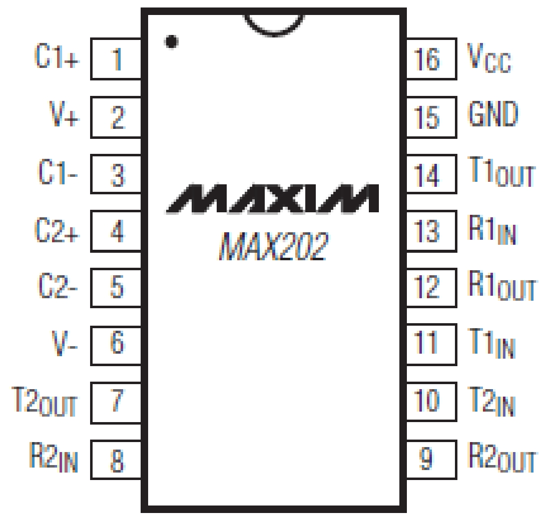
RS232 Level



Lab 6 Setup

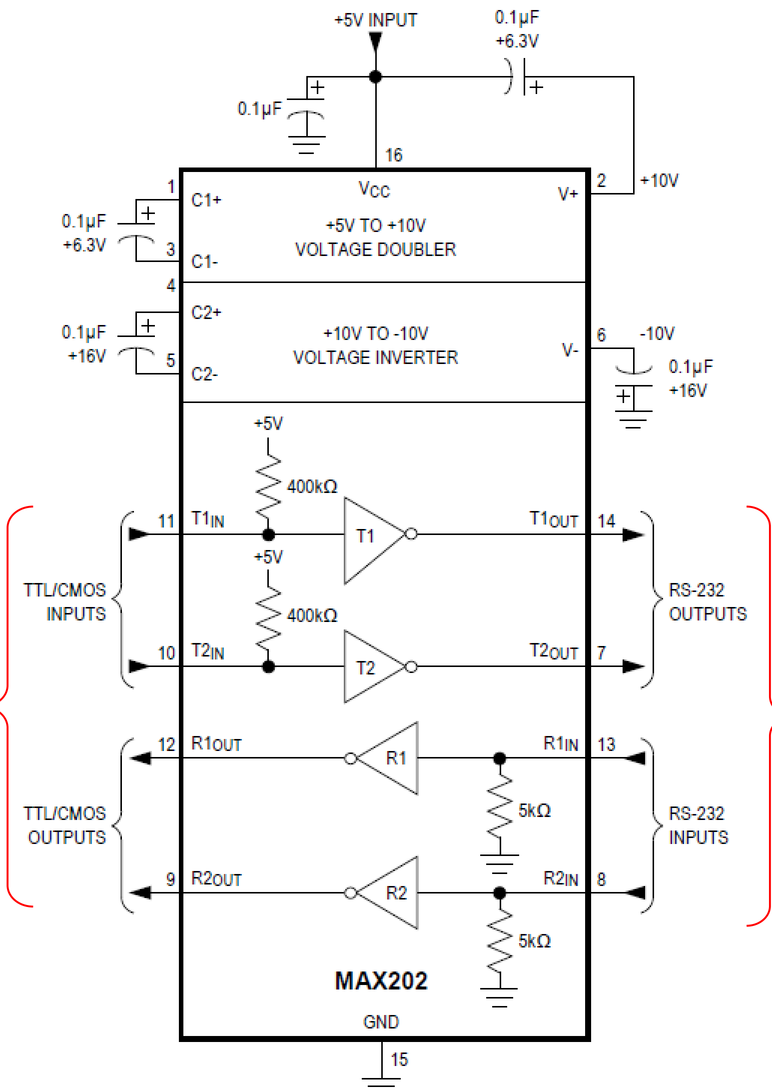
We need an interface between the two levels. The MAX202 transceiver (transmitter/receiver) is commonly used.





MAX202 Logical Circuit

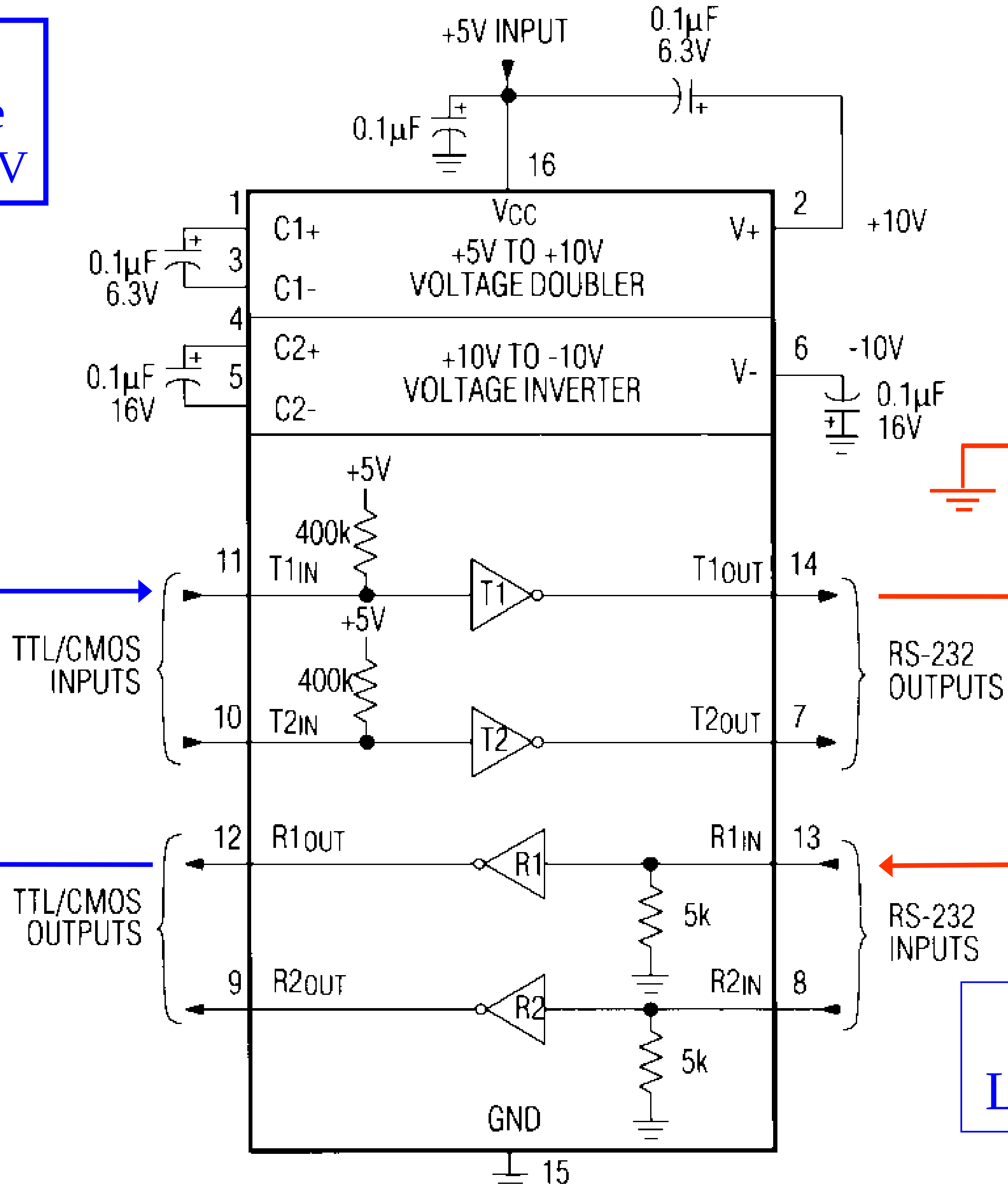
← To PIC



To PC →

PIC Side 0 - 5 V

PC Side ±12 V



From RC6/TX

To RC7/RX

Serial cable pin 5 (ground)

To serial cable pin 2 (RX on PC)

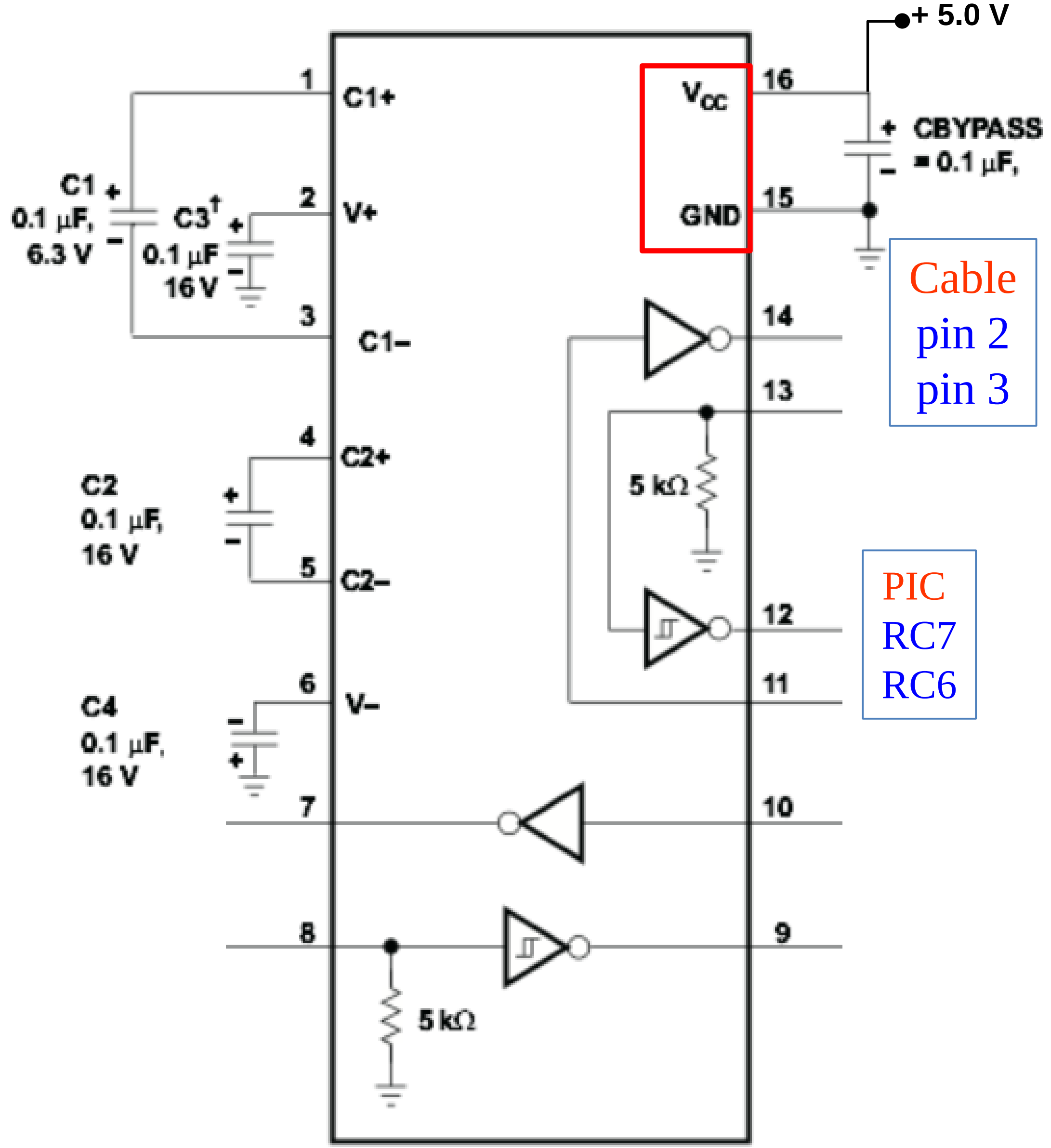
From serial cable pin 3 (TX on PC)

MAX202 Logical Circuit

1. The diagram on the previous slide is a **logical** representation of the circuit connections.

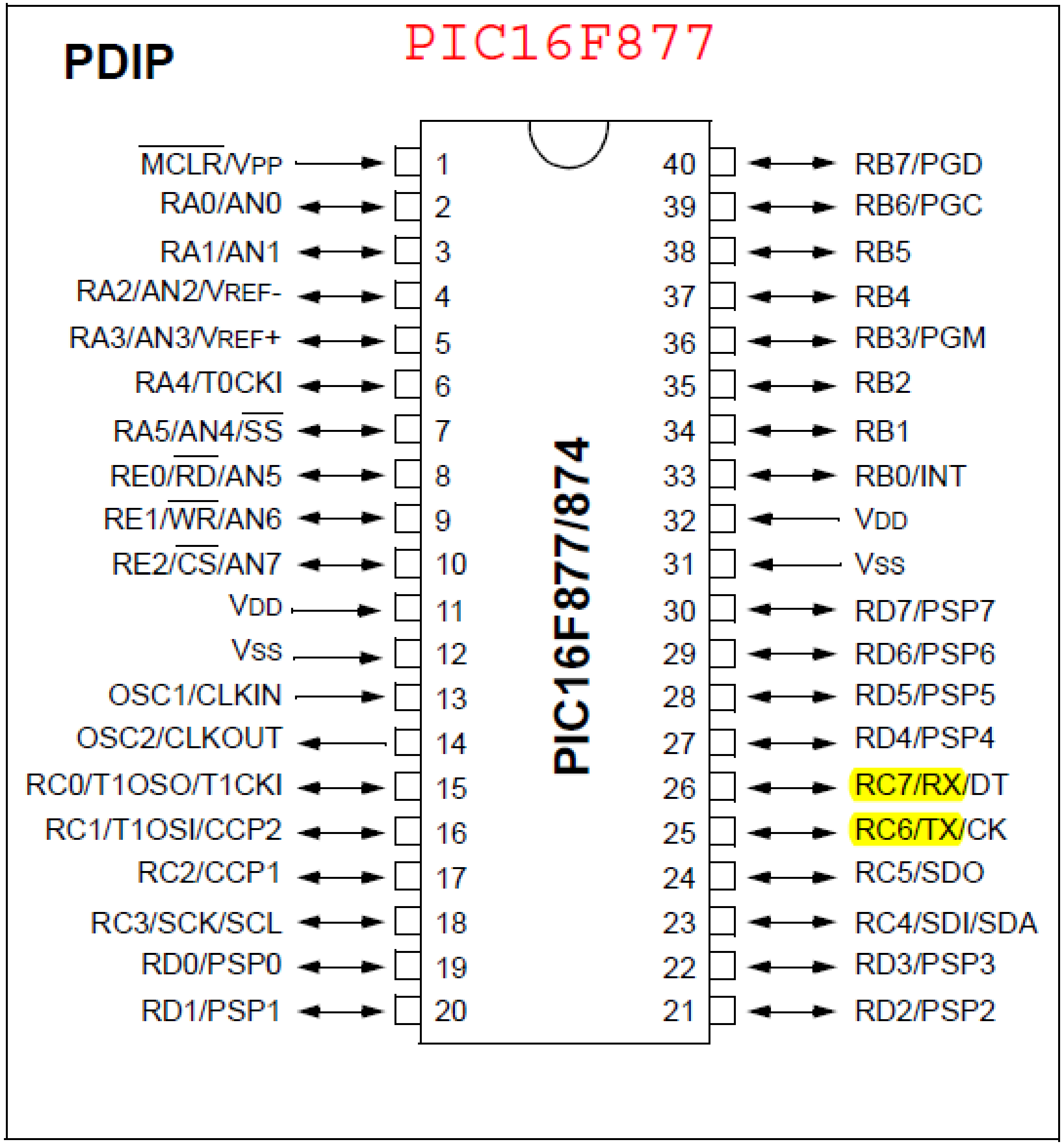
2. Use the diagram on the right for **physical** connections. Note the polarities of the capacitors.

3. **Caution**: The voltage from the PC serial port is ± 12 volts. Be sure your connections are correct before connecting the MAX202 to the PIC and connecting the PC serial port.



MAX202
Physical Circuit

USART TX and RX Pins



Lab 6 Setup

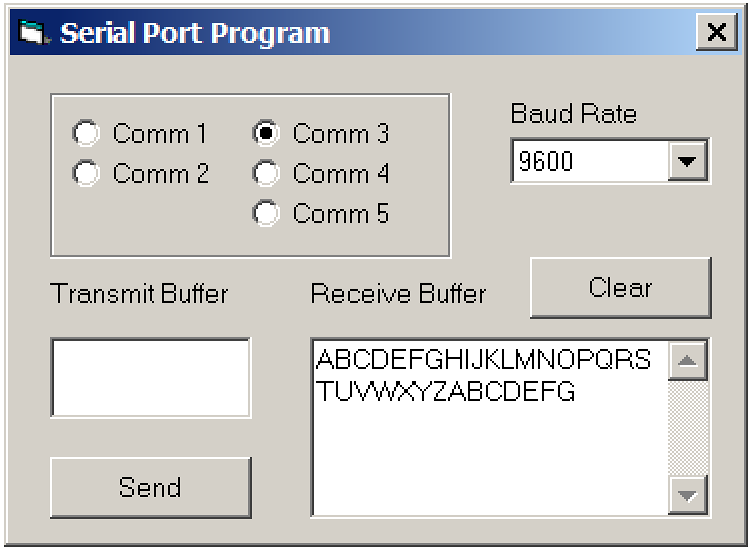
1. Download the SerialPortProgram.zip file from the Resources folder. This is a Windows application.

2. Unzip and run setup.exe to install the program shown below.

3. If the PC you are using does not have a serial port, you need a serial port or a USB-to-Serial adapter.

4. The Comm Port you select depends on your computer.

5. In order to run the Serial Port Program on the lab computers, double click on serial.exe



Lab06a.asm
Asynchronous Transmission

If the USART is enabled (SPEN = 1), TRISC<6> is automatically cleared when TXEN is set.

```
Init

    banksel RCSTA           ; Enable the USART serial port
    bsf     RCSTA, SPEN

    banksel TXSTA
    bcf     TXSTA, SYNC ; Set up the USART for asynchronous operation
    bsf     TXSTA, TXEN ; Transmit enabled. If the USART is enabled
                        ; (SPEN = 1), TRISC<RC6> is automatically
                        ; cleared when TXEN is set.
    bsf     TXSTA, BRGH ; High baud rate

    movlw   D'23'           ; This sets the baud rate to 9600
    banksel SPBRG           ; assuming BRGH = 1 and Fosc = 3.6864 MHz
    movwf   SPBRG           ; SPBRG = Fosc/(16*(Baud Rate)) - 1 = 23

                            ; TRISC<6> is automatically cleared when TXEN
                            ; is set, if the USART is enabled.

    banksel PIE1                ; Enable the Timer2 interrupt for the
    bsf     PIE1, TMR2IE        ; 1/2 sec delay.

    banksel INTCON          ; Enable global and peripheral interrupts
    bsf     INTCON, GIE
    bsf     INTCON, PEIE

    movlw   D'230'          ; Set up the Timer2 Period register
    banksel PR2             ; Timer2 period = Prescaler * (PR2 + 1) *
    movwf   PR2             ; Postscaler * 4 * Tosc = 4 * 231 * 2 *
                            ; 1.085 usec = 2.00 ms.

    movlw   B'00001101' ; Postscale = 2, Timer2 ON, prescaler = 4
    banksel T2CON
    movwf   T2CON

    movlw   D'65'           ; Initialize the serial port output to "A"
    movlw   "A"             ; This is another way to load "A" into the
    movwf   TX_temp         ; W register.
    return
```

Lab06a.asm

```
MainLoop

    call    Delay_500ms

    movf    TX_temp, W   ; W = TX_temp
    movwf   TXREG        ; Transmit TX_temp. When a byte is moved
                         ; into TXREG, the USART immediately
                         ; transmits the byte from the PIC's TX pin to
                         ; the RX pin on the serial port on the PC.
    addlw   1            ; W = TX_temp + 1
    movwf   TX_temp      ; TX_temp = TX_temp + 1

    sublw   "Z" + 1      ; W = "Z" + 1 - W
                         ; "Z" + 1 = 0x5A + 1 = 0x5B (See note below).

    btfss   STATUS, Z    ; If W = 0 (STATUS<Z> = 1), then
                         ; TX_temp = "Z" + 1, so the character just
                         ; sent was a "Z". Skip the next instruction
                         ; and reset TX_temp to "A".
    goto    MainLoop     ; Else goto MainLoop and send the next
                         ; character.
    movlw   "A"          ; Reset TX_temp to "A"
    movwf   TX_temp      ; Transmit "A"

    goto    MainLoop     ; Repeat indefinitely

    ; Note: The assembler can perform many operations that are not
    ; covered in this course. See Page 43 of the MPASM Assembler
    ; User Guide.
```

Lab06b.asm

Asynchronous Reception

Transmit 0, 1, 2, or X from the Serial Port Program on the PC to be received by the serial port on the PIC – turn on the LED connected to RC0, RC1, or RC2, or turn off (X) all LEDs

Lab06b.asm

```
Init

    banksel RCSTA
    bsf     RCSTA, SPEN ; Enable the USART serial port
    bsf     RCSTA, CREN ; Enable serial port reception

    banksel TXSTA
    bcf     TXSTA, SYNC ; Set up the USART for asynchronous operation
    bsf     TXSTA, BRGH ; High baud rate

    movlw   D'23'       ; This sets the baud rate to 9600
    banksel SPBRG       ; assuming BRGH = 1 and Fosc = 3.6864 MHz
    movwf   SPBRG       ; SPBRG = Fosc/(16*(Baud Rate)) - 1 = 23

    banksel PIE1        ; Enable the Serial Port Reception Interrupt
    bsf     PIE1, RCIE

    banksel INTCON      ; Enable global and peripheral interrupts
    bsf     INTCON, GIE
    bsf     INTCON, PEIE

    banksel TRISC       ; Set PortC bits 0, 1, and 2 as outputs
                        ; Set RC7/RX as an input pin

    movlw   B'11111000'
    movwf   TRISC

    banksel PORTC       ; Clear PortC bits 0, 1, and 2
    clrf    PORTC

    return
```

Lab06b.asm

```
Receive

    movf    RCREG, W    ; Read and empty the RCREG register.

    sublw   D'48'       ; W = 48 - W.  (ASCII "0" = 0x48)

    btfsc   STATUS, Z   ; Check if a "0" was received
    goto    LED0        ; If so (W = 0, Z = 1), don't skip.

    movf    RCREG, W    ; If not, read RCREG again.
    sublw   D'49'       ; Check if a "1" was received
    btfsc   STATUS, Z
    goto    LED1

    movf    RCREG, W
    sublw   D'50'       ; Check if a "2" was received
    btfsc   STATUS, Z
    goto    LED2

    movf    RCREG, W
    sublw   D'88'       ; Check if an "X" was received
    btfsc   STATUS, Z
    goto    LEDOff
    return
```

… Receive Routine (continued) …

Lab06b.asm

```
LED0                            ; Turn on RC0

        movlw   B'00000001'
        movwf   PORTC
        return

LED1                            ; Turn on RC1
        movlw   B'00000010'
        movwf   PORTC
        return

LED2                            ; Turn on RC2
        movlw   B'00000100'
        movwf   PORTC
        return

LEDOff                          ; Turn off RC0, RC1, and RC2

        clrf    PORTC
        return
```

PIC Serial Communications

For a narrative on PIC serial communications, see Chapter 9 of

Embedded Systems Programming with the Pic16F877

Timothy D. Green

End of Lab 6