Intermediate Electrical and Computer Engineering Design Experience

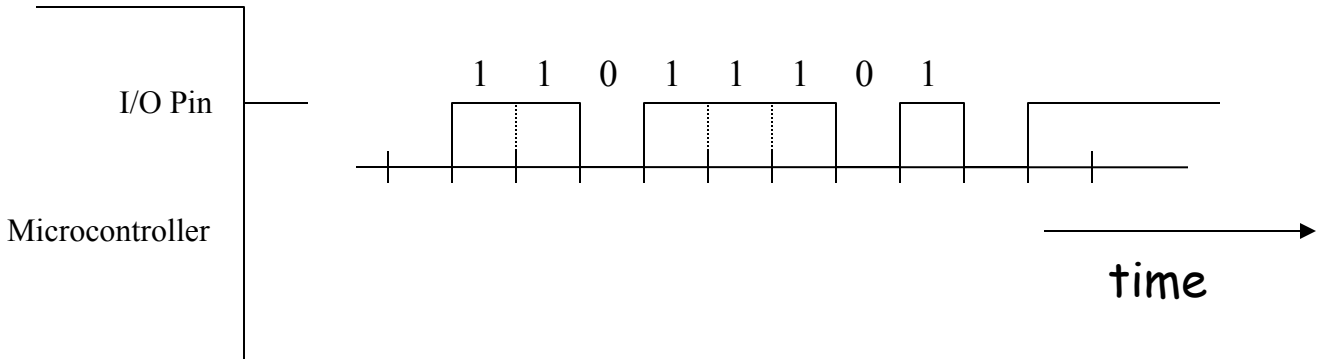
# *Serial Communication*

*EE395A*
*Winter 2011/2012*

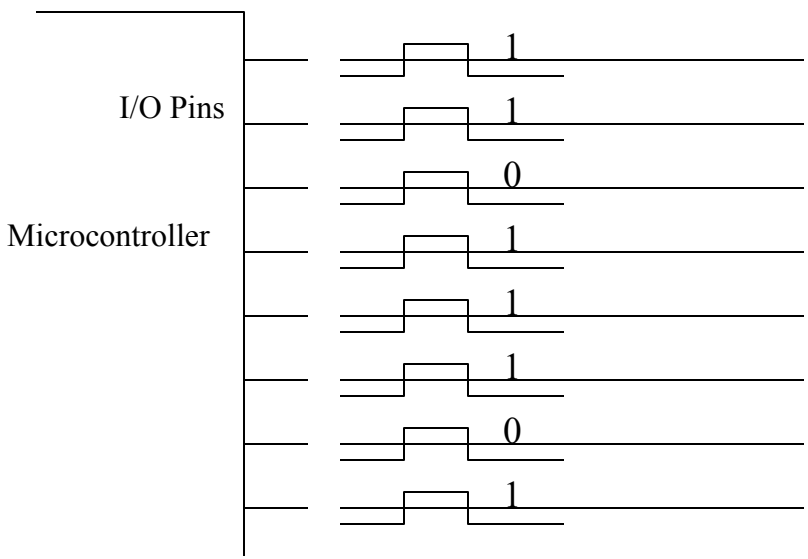
*by*
*Maarten Uijt de Haag, Tim Bambeck*

# I/O - Communications

- **Serial I/O:**




- **Parallel I/O:**




**<u>Advantage serial</u>**:
Only need one data line and ground.

**<u>Advantage parallel</u>**:
Multiple bits are sent or received at the same time.

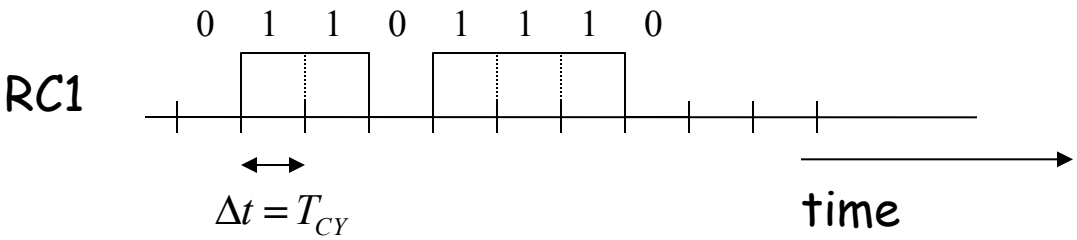
# How ?

## *Would you implement serial Communication*

Suppose you want to send ‘01110110’ on pin RC1.

```
bcf     PORTC,1     ; ‘0’
bsf     PORTC,1     ; ‘1’
bsf     PORTC,1     ; ‘1’
bcf     PORTC,1     ; ‘0’
bsf     PORTC,1     ; ‘1’
bsf     PORTC,1     ; ‘1’
bsf     PORTC,1     ; ‘1’
bcf     PORTC,1     ; ‘0’
```

**Bit-Banging**




Note: sent LSB first here

# How ?

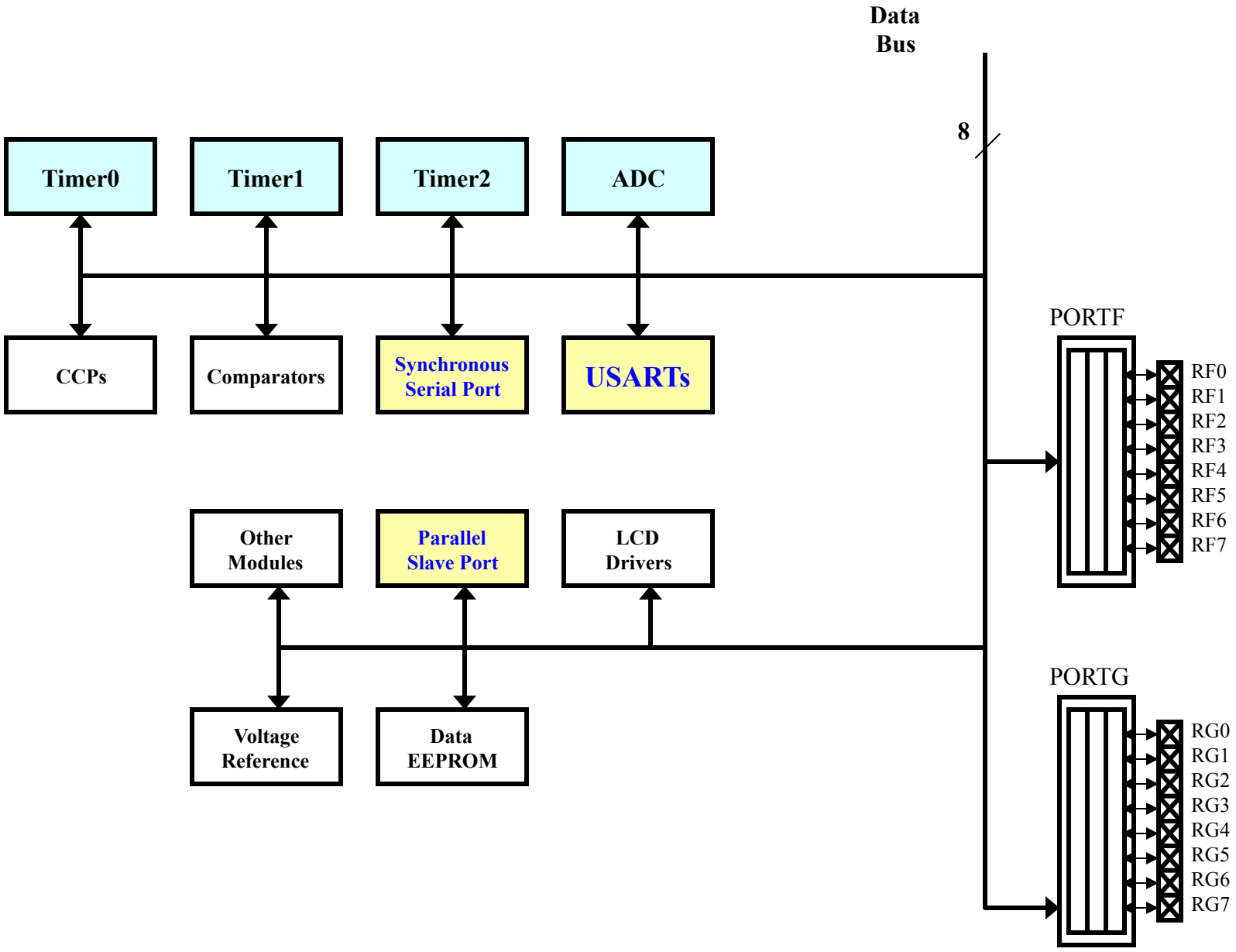
## *Would you implement serial Communication*

Suppose you want to send '01110110' on pin RC1 at a rate of ~100Hz
( T = 1/100 = 0.01 seconds)

```
bcf     PORTC,1     ; '0'
call    DELAY       ; 0.01 second delay
bsf     PORTC,1     ; '1'
call    DELAY       ; 0.01 second delay
bsf     PORTC,1     ; '1'
call    DELAY       ; 0.01 second delay
bcf     PORTC,1     ; '0'
bsf     PORTC,1     ; '1'
call    DELAY       ; 0.01 second delay
bsf     PORTC,1     ; '1'
call    DELAY       ; 0.01 second delay
bsf     PORTC,1     ; '1'
call    DELAY       ; 0.01 second delay
bcf     PORTC,1     ; '0'
call    DELAY       ; 0.01 second delay
```

# But …

- Bit-Banging is not required because:
  - The PIC microcontrollers have build-in serial communication and parallel communication devices/peripherals:
- Serial:
  - SPI, $I^2C$, USART
- Parallel:
  - PSP

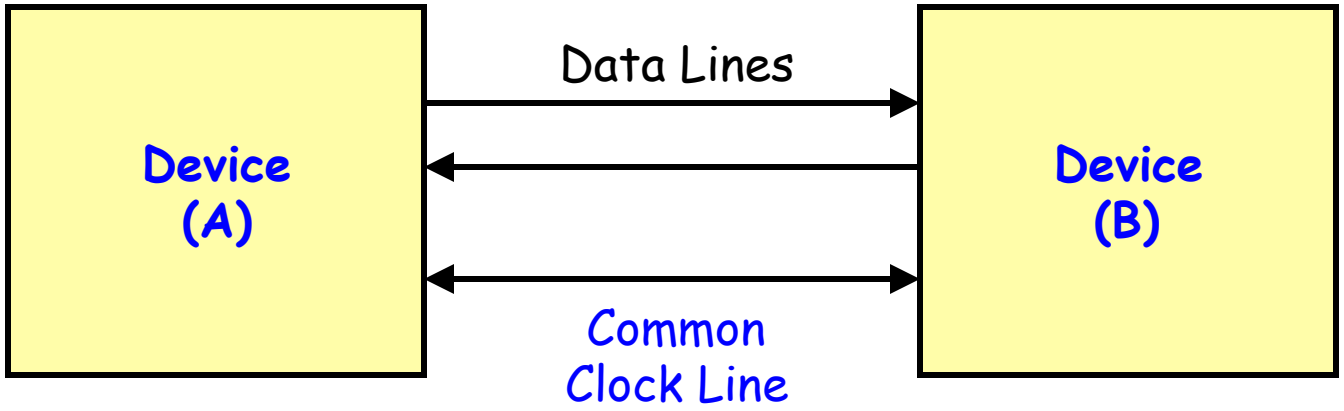



SerialComm.6

*Figure 4-2 in the Reference Manual*

# Serial Communications

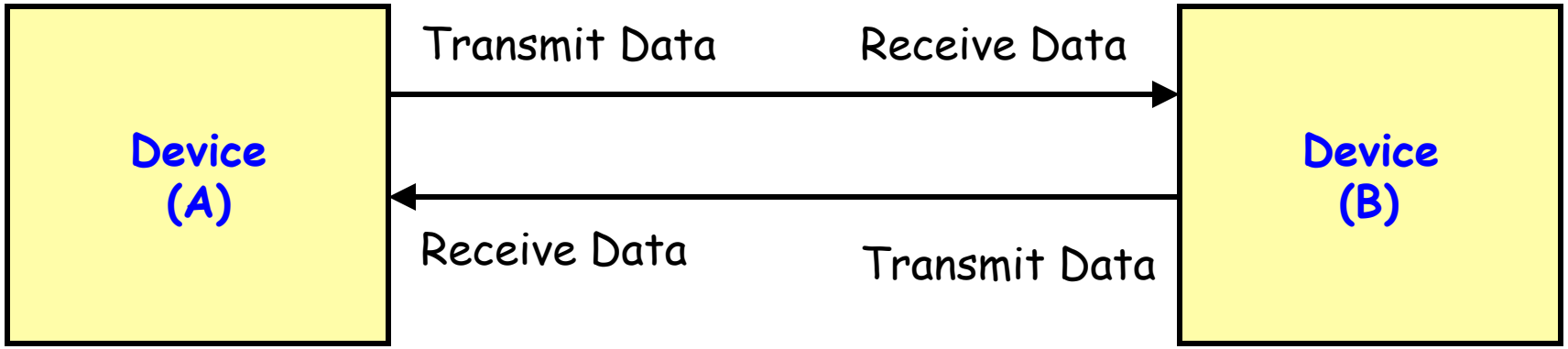
## *Asynchronous vs. Synchronous*




# Synchronous

# Serial Communications

## *Asynchronous vs. Synchronous*




# Asynchronous

# Synchronous I/O

- Synchronous Serial Port (SSP)
  - Serial Peripheral Interface (SPI)
    - RM: Section 15.3, 16.3, 17.3
    - DS: Section 9.1
  - Inter-Integrated Circuit ($I^2C$)
    - RM: Section 15.4, 16.4, 17.4
    - DS: Section 9.2
- Universal Synchronous / Asynchronous Receiver Transmitter (USART)
  - Synchronous Mode
    - RM: Section 18.5, 18.6
    - DS: Section 10.3, 10.4

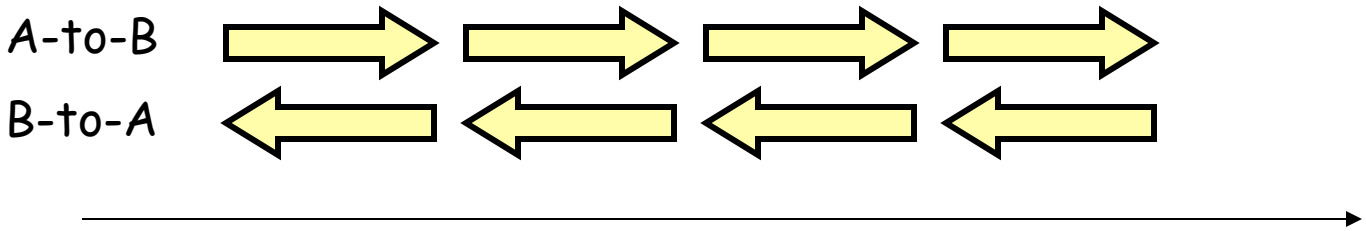
# Asynchronous I/O

- Universal Synchronous / Asynchronous Receiver Transmitter (USART)
  - Asynchronous Mode
    - RM: Section 18.4
    - DS: Section 10.2

# Serial Communications

## *Full Duplex vs. Half Duplex*

**Full Duplex:**

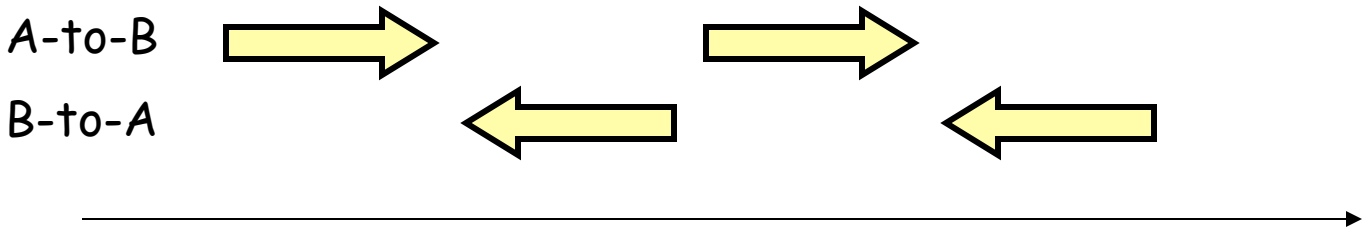



Data can be transmitted in both directions on a signal carrier, <u>at the same time</u>.

# Serial Communications

## *Full Duplex vs. Half Duplex*

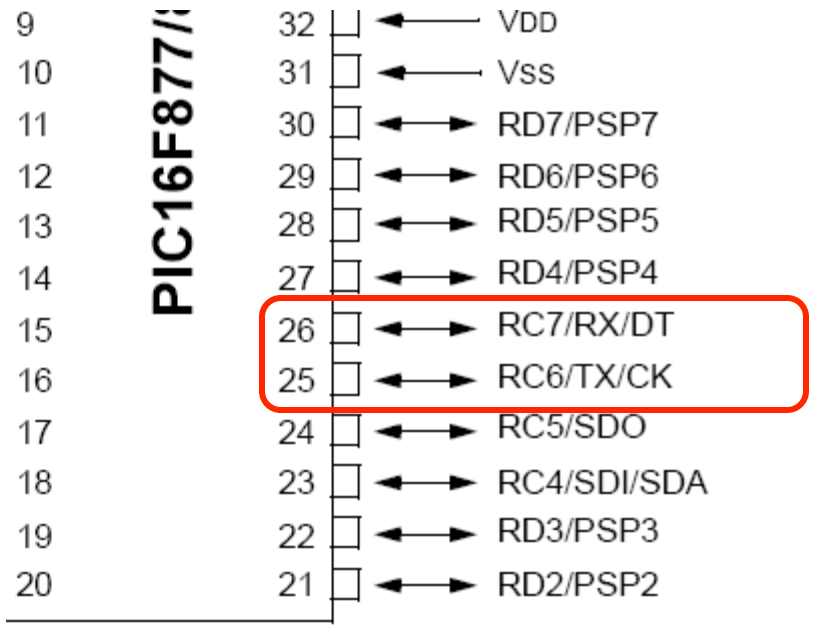
**Half Duplex:**




Data can be transmitted in both directions on a signal carrier, but not at the same time.

# Asynchronous I/O

## *USART – Full Duplex*

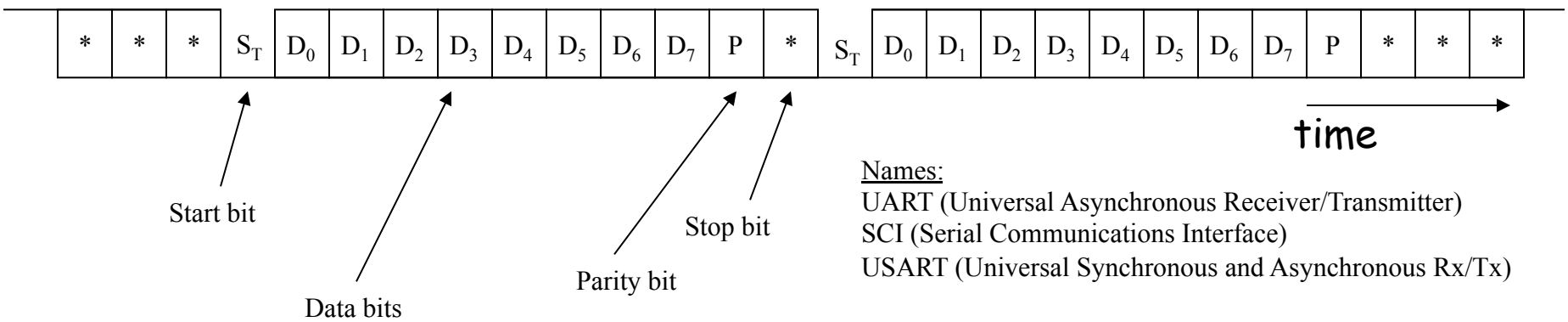



USART

**RX – Receive Line**
**TX – Transmit Line**

Multiplexing of I/O Pins

# Asynchronous I/O
## *USART*




Names:
UART (Universal Asynchronous Receiver/Transmitter)
SCI (Serial Communications Interface)
USART (Universal Synchronous and Asynchronous Rx/Tx)

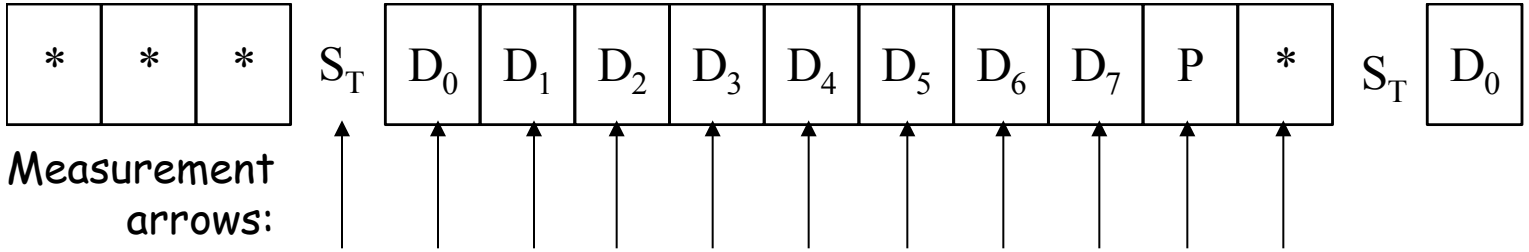
**Baud rate: number of bits transferred per second**

Example: - Packet consists of 8 data bits, 1 parity bit, 1 start, and 1 stop bit
- Baud rate = 115,200 baud

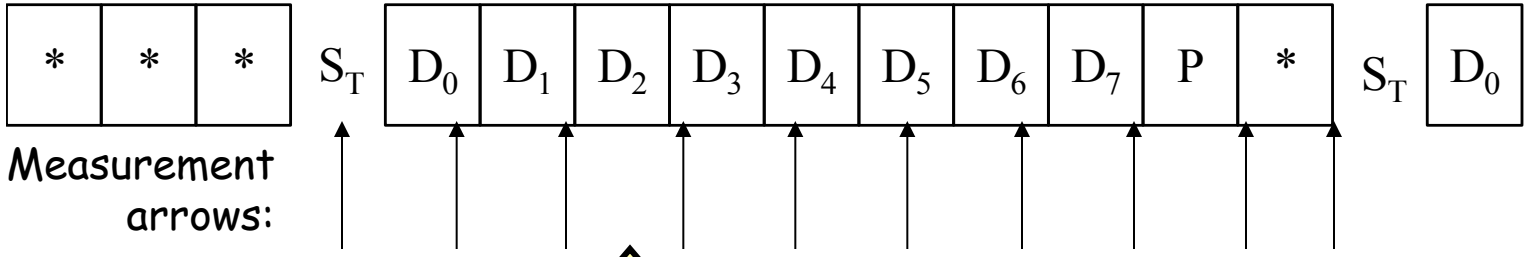
- 115,200 bits/second and 11 bits per word => ~10,473 words/second
- 8 data bits per word => ~10,473 bytes/second
- (~10,473*8) = ~83,782 bits/second

# USART

## *Baud Rate Problems*



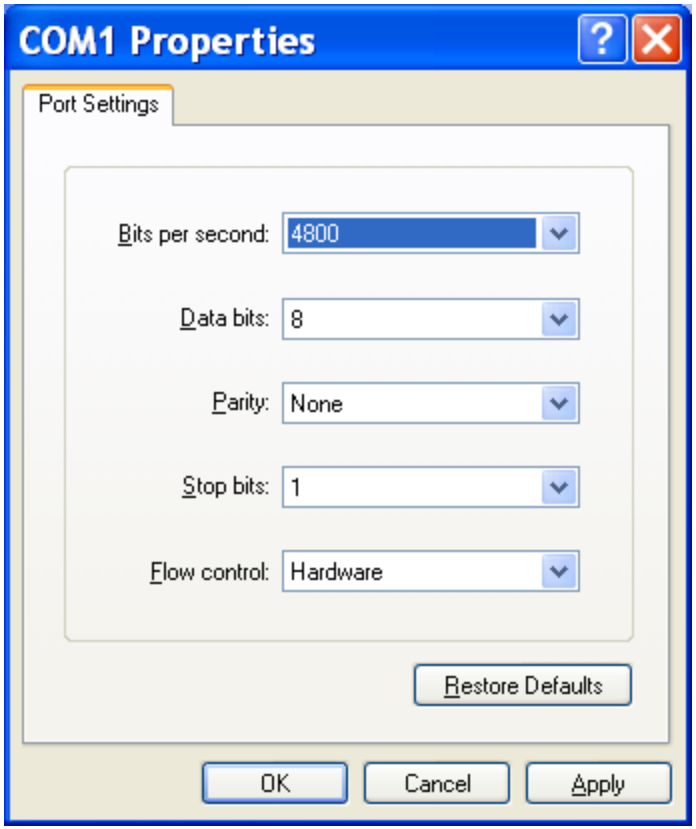

No problem Occurs

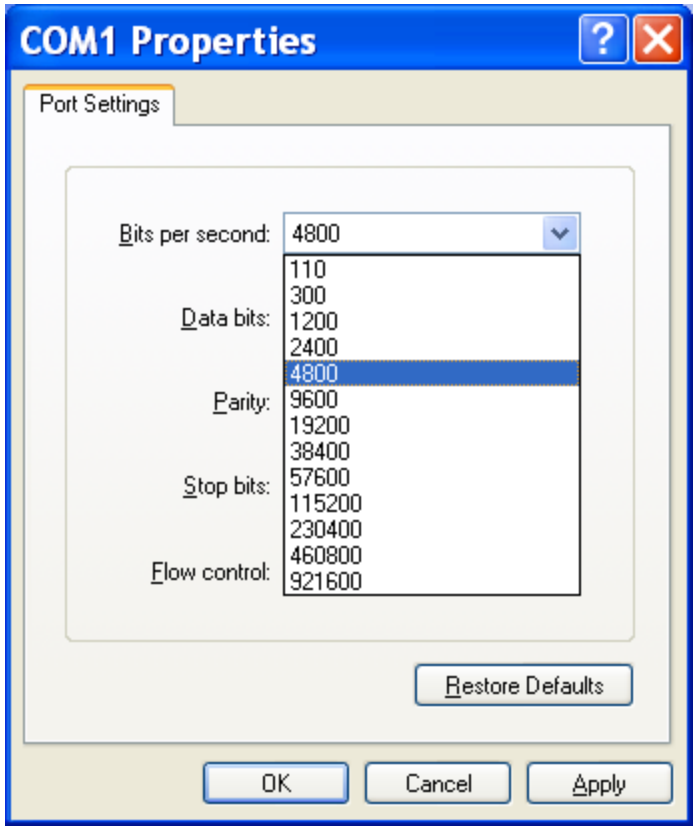



Oops, screwed up the $D_2$ bit

<u>So, it is important that both units use the same baud rates!!!!!</u>

# Common BAUD Rates for Personal Computers:




From “Hyperterminal” program in Microsoft Windows.

# USART

## *Configuration – Baud Rate*

**Table 18-2: Registers Associated with Baud Rate Generator**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on: POR, BOR | Value on all other resets |
|---|---|---|---|---|---|---|---|---|---|---|
| TXSTA | CSRC | TX9 | TXEN | SYNC | — | BRGH | TRMT | TX9D | 0000 -010 | 0000 -010 |
| RCSTA | SPEN | RX9 | SREN | CREN | — | FERR | OERR | RX9D | 0000 -00x | 0000 -00x |
| SPBRG | Baud Rate Generator Register | | | | | | | | 0000 0000 | 0000 0000 |

Legend: x = unknown, - = unimplemented read as '0'. Shaded cells are not used by the BRG.

TX setup

RX setup

**Tab**

| Name | Bit 7 | Bit |
|---|---|---|
| TXSTA | CSRC | TX |
| RCSTA | SPEN | RX |
| SPBRG | Baud Rate Ge | |

Legend: x = unknown, -

To determine the baud rate

**Register 18-1: TXSTA: Transmit Status and Control Register**

| 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

For both receiver & transmitter

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

For both receiver & transmitter

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

Transmission complete?

bit 0 **TX9D:** 9th bit of transmit data. Can be parity bit.

**Register 18-2: RCSTA: Receive Status and Control Register**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | U-0 | R-0 | R-0 | R-0 |
|---|---|---|---|---|---|---|---|
| SPEN | RX9 | SREN | CREN | — | FERR | OERR | RX9D |
| bit 7 | | | | | | | bit 0 |

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

For both receiver & transmitter

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
Unused in this mode

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 **Unimplemented:** Read as '0'

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

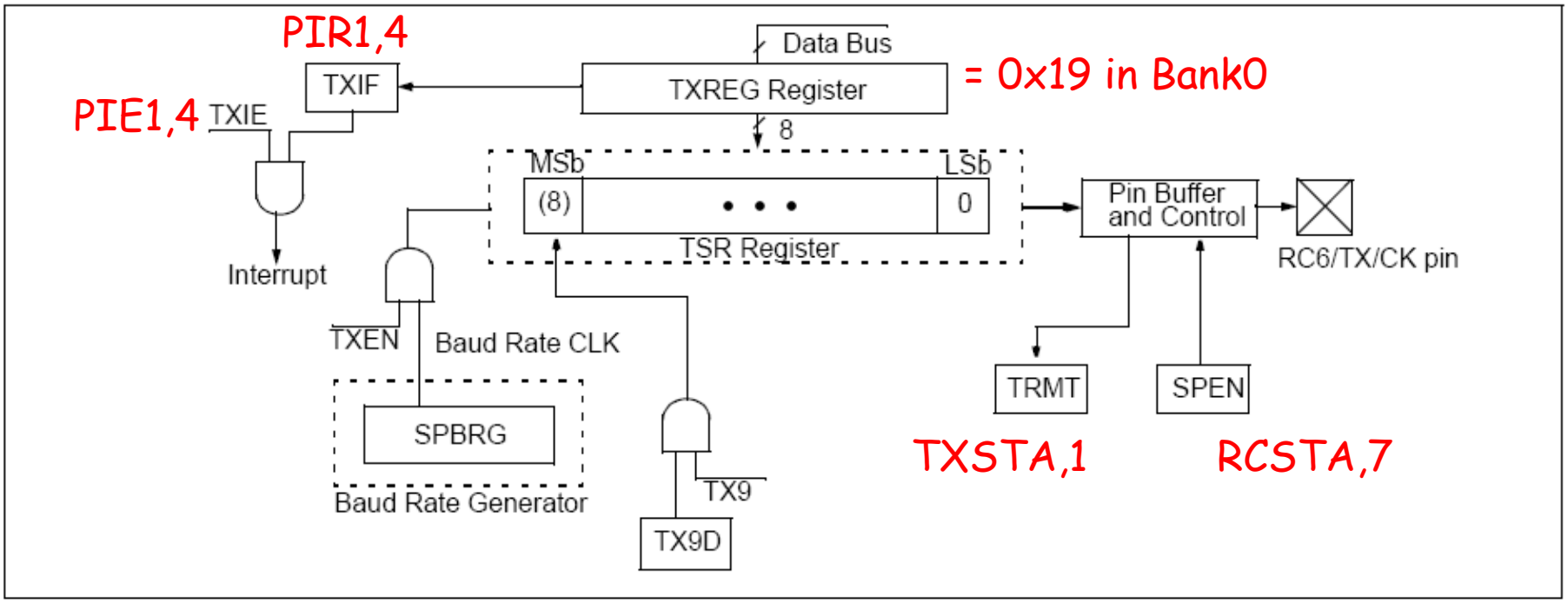
bit 1 **OERR**: Overrun Error bit
1 = Overrun error (Can be cleared by clearing bit CREN)
0 = No overrun error

bit 0 **RX9D:** 9th bit of received data, can be parity bit.

# USART

## *Inside the Transmitter*

**FIGURE 10-1: USART TRANSMIT BLOCK DIAGRAM**



# USART

## *Baud Rate Generator Byte*

| | | | |
|---|---|---|---|
| SPBRG | Baud Rate Generator Register | 0000 0000 | 0000 0000 |

Legend: x = unknown, - = unimplemented read as '0'. Shaded cells are not used by the BRG.

X

BRGH is: Bit 2 in TXSTA

**Table 18-1: Baud Rate Formula**

| SYNC | BRGH = 0 (Low Speed) | BRGH = 1 (High Speed) |
|---|---|---|
| 0 | (Asynchronous) Baud Rate = Fosc/(64(X+1)) | Baud Rate= Fosc/(16(X+1)) |
| 1 | (Synchronous) Baud Rate = Fosc/(4(X+1)) | NA |

X = value in SPBRG (0 to 255)

# USART - *Baud Rate Computation*

## Given the following parameters:

$f_{osc} = 16MHz$

Desired baud rate $= 57{,}600$ bps

(set SYNC $= 0$ for asynchronous mode)

If we select BRGH = 0

Desired baud rate $= 57{,}600$

$$57{,}600 = f_{osc}/(64(X+1))$$

$$57{,}600 = 16 \cdot 10^6 / (64(X+1)) \Rightarrow$$

$$64(X+1) = 16 \cdot 10^6 / 57{,}600 \Rightarrow$$

$$X = round\left(16 \cdot 10^6 / (57{,}600 \cdot 64)) - 1\right)$$

$$X = round(3.34)$$

$$X = 3 = SPBRG$$

Calculated baud rate $= f_{osc}/(64(X+1))$

$$= 16 \cdot 10^6 / (64(3+1))$$

$$= 62{,}500$$

Percent Error $= [(calculated - desired) / desired] * 100$

$$= [(62{,}500 - 57{,}600) / 57{,}600] * 100$$

$$= +8.5\%$$

# USART - *Baud Rate Computation*

## Given the following parameters:

$f_{osc} = 16MHz$

Desired baud rate $= 57{,}600$ bps

(set SYNC $= 0$ for asynchronous mode)

If we select BRGH = 1

Desired baud rate $= 57{,}600$

$$57{,}600 = f_{osc}/(16(X+1))$$

$$57{,}600 = 16 \cdot 10^6 / (16(X+1)) \Rightarrow$$

$$16(X+1) = 16 \cdot 10^6 / 57{,}600 \Rightarrow$$

$$X = round\left(16 \cdot 10^6 / (57{,}600 \cdot 16)) - 1\right)$$

$$X = round(16.36)$$

$$X = 16 = SPBRG$$

Calculated baud rate $= f_{osc}/(16(X+1))$

$$= 16 \cdot 10^6 / (16(16+1))$$

$$= 58{,}824$$

Percent Error $= [(calculated - desired) / desired] * 100$

$$= [(58{,}824 - 57{,}600) / 57{,}600] * 100$$

$$= +2.1\%$$

# Set BRGH and SPBRG to BEST Values:

Recall the design parameters:

$$f_{osc} = 16MHz$$

$$\text{Desired baud rate} = 57{,}600\ \text{bps}$$

$$(\text{set SYNC} = 0 \text{ for asynchronous mode})$$

If BRGH = 0

Then SPBRG = 3

Baud Error = + 8.5 %

If BRGH = 1

Then SPBRG = 16

Baud Error = + 2.1 %

So use this one for best choice:

## TABLE 10-3: BAUD RATES FOR ASYNCHRONOUS MODE (BRGH = 0)

| BAUD RATE (K) | Fosc = 20 MHz | | | Fosc = 16 MHz | | | Fosc = 10 MHz | | |
|---|---|---|---|---|---|---|---|---|---|
| | KBAUD | % ERROR | SPBRG value (decimal) | KBAUD | % ERROR | SPBRG value (decimal) | KBAUD | % ERROR | SPBRG value (decimal) |
| 0.3 | - | - | - | - | - | - | - | - | - |
| 1.2 | 1.221 | 1.75 | 255 | 1.202 | 0.17 | 207 | 1.202 | 0.17 | 129 |
| 2.4 | 2.404 | 0.17 | 129 | 2.404 | 0.17 | 103 | 2.404 | 0.17 | 64 |
| 9.6 | 9.766 | 1.73 | 31 | 9.615 | 0.16 | 25 | 9.766 | 1.73 | 15 |
| 19.2 | 19.531 | 1.72 | 15 | 19.231 | 0.16 | 12 | 19.531 | 1.72 | 7 |
| 28.8 | 31.250 | 8.51 | 9 | 27.778 | 3.55 | 8 | 31.250 | 8.51 | 4 |
| 33.6 | 34.722 | 3.34 | 8 | 35.714 | 6.29 | 6 | 31.250 | 6.99 | 4 |
| 57.6 | 62.500 | 8.51 | 4 | 62.500 | 8.51 | 3 | 52.083 | 9.58 | 2 |
| HIGH | 1.221 | - | 255 | 0.977 | - | 255 | 0.610 | - | 255 |
| LOW | 312.500 | - | 0 | 250.000 | - | 0 | 156.250 | - | 0 |

| BAUD RATE (K) | Fosc = 4 MHz | | | Fosc = 3.6864 MHz | | |
|---|---|---|---|---|---|---|
| | KBAUD | % ERROR | SPBRG value (decimal) | KBAUD | % ERROR | SPBRG value (decimal) |
| 0.3 | 0.300 | 0 | 207 | 0.3 | 0 | 191 |
| 1.2 | 1.202 | 0.17 | 51 | 1.2 | 0 | 47 |
| 2.4 | 2.404 | 0.17 | 25 | 2.4 | 0 | 23 |
| 9.6 | 8.929 | 6.99 | 6 | 9.6 | 0 | 5 |
| 19.2 | 20.833 | 8.51 | 2 | 19.2 | 0 | 2 |
| 28.8 | 31.250 | 8.51 | 1 | 28.8 | 0 | 1 |
| 33.6 | - | - | - | - | - | - |
| 57.6 | 62.500 | 8.51 | 0 | 57.6 | 0 | 0 |
| HIGH | 0.244 | - | 255 | 0.225 | - | 255 |
| LOW | 62.500 | - | 0 | 57.6 | - | 0 |

Low-speed Mode
(from Datasheet)

## TABLE 10-4: BAUD RATES FOR ASYNCHRONOUS MODE (BRGH = 1)

| BAUD RATE (K) | Fosc = 20 MHz | | | Fosc = 16 MHz | | | Fosc = 10 MHz | | |
|---|---|---|---|---|---|---|---|---|---|
| | KBAUD | % ERROR | SPBRG value (decimal) | KBAUD | % ERROR | SPBRG value (decimal) | KBAUD | % ERROR | SPBRG value (decimal) |
| 0.3 | - | - | - | - | - | - | - | - | - |
| 1.2 | - | - | - | - | - | - | - | - | - |
| 2.4 | - | - | - | - | - | - | 2.441 | 1.71 | 255 |
| 9.6 | 9.615 | 0.16 | 129 | 9.615 | 0.16 | 103 | 9.615 | 0.16 | 64 |
| 19.2 | 19.231 | 0.16 | 64 | 19.231 | 0.16 | 51 | 19.531 | 1.72 | 31 |
| 28.8 | 29.070 | 0.94 | 42 | 29.412 | 2.13 | 33 | 28.409 | 1.36 | 21 |
| 33.6 | 33.784 | 0.55 | 36 | 33.333 | 0.79 | 29 | 32.895 | 2.10 | 18 |
| 57.6 | 59.524 | 3.34 | 20 | 58.824 | 2.13 | 16 | 56.818 | 1.36 | 10 |
| HIGH | 4.883 | - | 255 | 3.906 | - | 255 | 2.441 | - | 255 |
| LOW | 1250.000 | - | 0 | 1000.000 | | 0 | 625.000 | - | 0 |

| BAUD RATE (K) | Fosc = 4 MHz | | | Fosc = 3.6864 MHz | | |
|---|---|---|---|---|---|---|
| | KBAUD | % ERROR | SPBRG value (decimal) | KBAUD | % ERROR | SPBRG value (decimal) |
| 0.3 | - | - | - | - | - | - |
| 1.2 | 1.202 | 0.17 | 207 | 1.2 | 0 | 191 |
| 2.4 | 2.404 | 0.17 | 103 | 2.4 | 0 | 95 |
| 9.6 | 9.615 | 0.16 | 25 | 9.6 | 0 | 23 |
| 19.2 | 19.231 | 0.16 | 12 | 19.2 | 0 | 11 |
| 28.8 | 27.798 | 3.55 | 8 | 28.8 | 0 | 7 |
| 33.6 | 35.714 | 6.29 | 6 | 32.9 | 2.04 | 6 |
| 57.6 | 62.500 | 8.51 | 3 | 57.6 | 0 | 3 |
| HIGH | 0.977 | - | 255 | 0.9 | - | 255 |
| LOW | 250.000 | - | 0 | 230.4 | - | 0 |

High-speed Mode
(from Datasheet)

# USART

## *Configuration - Asynchronous Transmission*

1. Initialize SPBRG (Baud Rate) & BRGH bit,
2. Enable Asynchronous Serial Port, → SPEN = 1 ,SYNC = 0
3. Setup interrupts if desired, → TXIE = 1, GIE = 1, PIE = 1
4. If 9bits, set TX9 bit,
5. Enable transmission, → TXEN = 1
6. If 9 bits, load 9th bit in TX9D,
7. Test if OK to load TXREG. → TXIF = 0, not OK; TXIF = 1, then OK
8. Load data in TXREG. → Starts transmission automatically

# USART

## *Registers - Asynchronous Transmission*

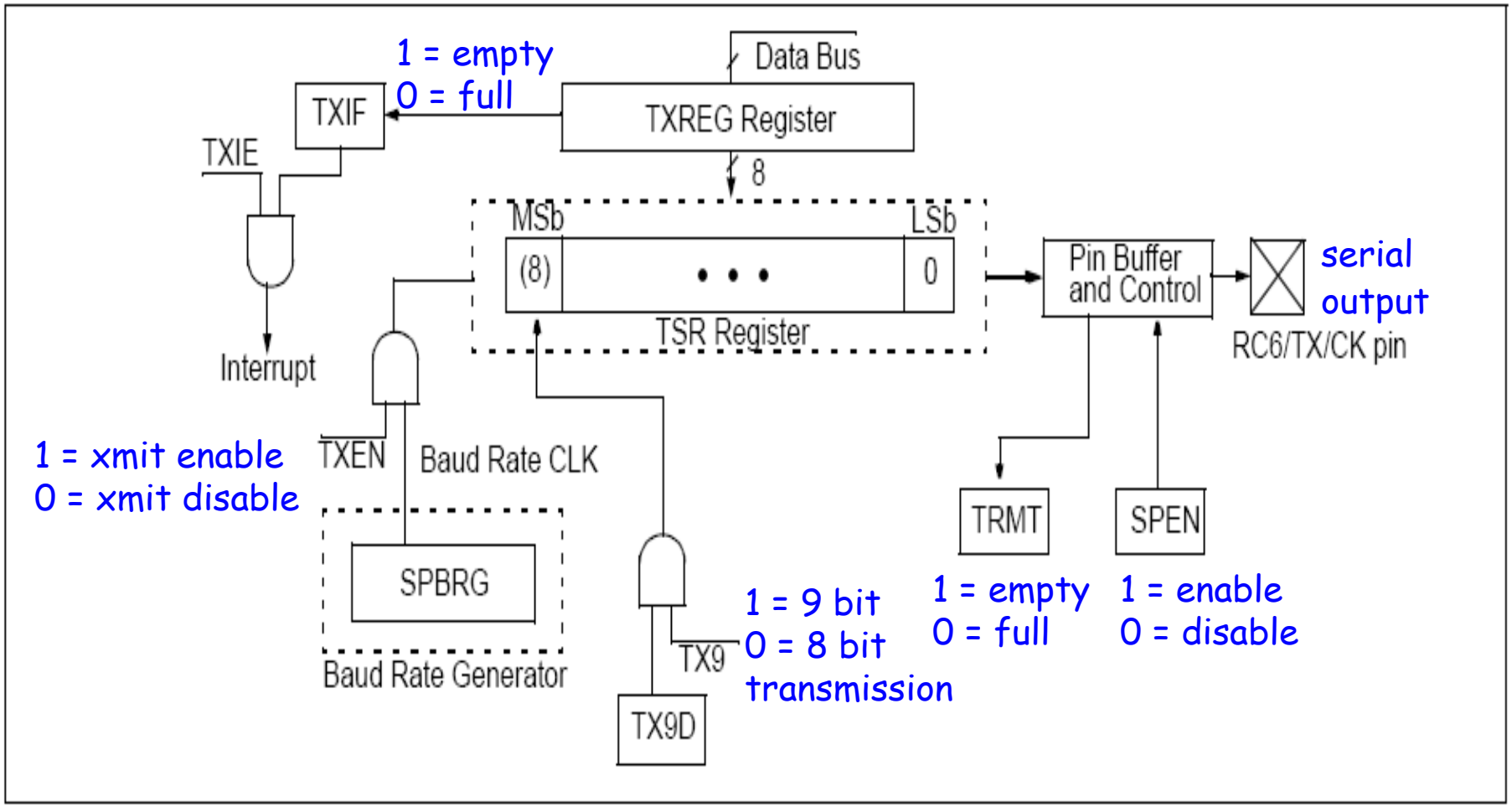
**TABLE 10-5: REGISTERS ASSOCIATED WITH ASYNCHRONOUS TRANSMISSION**

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on: POR, BOR | Value on all other RESETS |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0Bh, 8Bh, 10Bh,18Bh | INTCON | GIE | PEIE | T0IE | INTE | RBIE | T0IF | INTF | R0IF | 0000 000x | 0000 000u |
| 0Ch | PIR1 | PSPIF(1) | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 0000 0000 | 0000 0000 |
| 18h | RCSTA | SPEN | RX9 | SREN | CREN | — | FERR | OERR | RX9D | 0000 -00x | 0000 -00x |
| 19h | TXREG | USART Transmit Register | | | | | | | | 0000 0000 | 0000 0000 |
| 8Ch | PIE1 | PSPIE(1) | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 0000 0000 | 0000 0000 |
| 98h | TXSTA | CSRC | TX9 | TXEN | SYNC | — | BRGH | TRMT | TX9D | 0000 -010 | 0000 -010 |
| 99h | SPBRG | Baud Rate Generator Register | | | | | | | | 0000 0000 | 0000 0000 |

Legend: x = unknown, - = unimplemented locations read as '0'. Shaded cells are not used for asynchronous transmission.
**Note 1:** Bits PSPIE and PSPIF are reserved on the PIC16F873/876; always maintain these bits clear.

# USART - *Inside the Transmitter*

**FIGURE 10-1: USART TRANSMIT BLOCK DIAGRAM**



# USART

## *Code Example - Asynchronous Transmission*

Setup as follows:
Asynchronous transmission, baud rate 57,600 baud, no interrupts, SPBRG=d'16', 16MHz oscillator

```
bsf     STATUS, RP0        ; access bank 1
movlw   B'10111111'        ; RC6: TX (output); RC7: RX (input)
movwf   TRISC              ; set the port's inputs/outputs
movlw   B'00100110'        ; setup the transmission
movwf   TXSTA              ; 8-bits, asynch., etc.
movlw   d'16'              ; set baud rate to 57,600
movwf   SPBRG
bcf     STATUS, RP0        ; access bank 0
movlw   B'10010000'        ; SPEN enable, CREN enable
movwf   RCSTA              ; enable the serial port
movlw   A'M'               ; transmit a capital letter 'M'
call    xmit               ; call the transmit subroutine.
```

# Example Addendum

**Register 18-1: TXSTA: Transmit Status and Control Register**

| 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 |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |

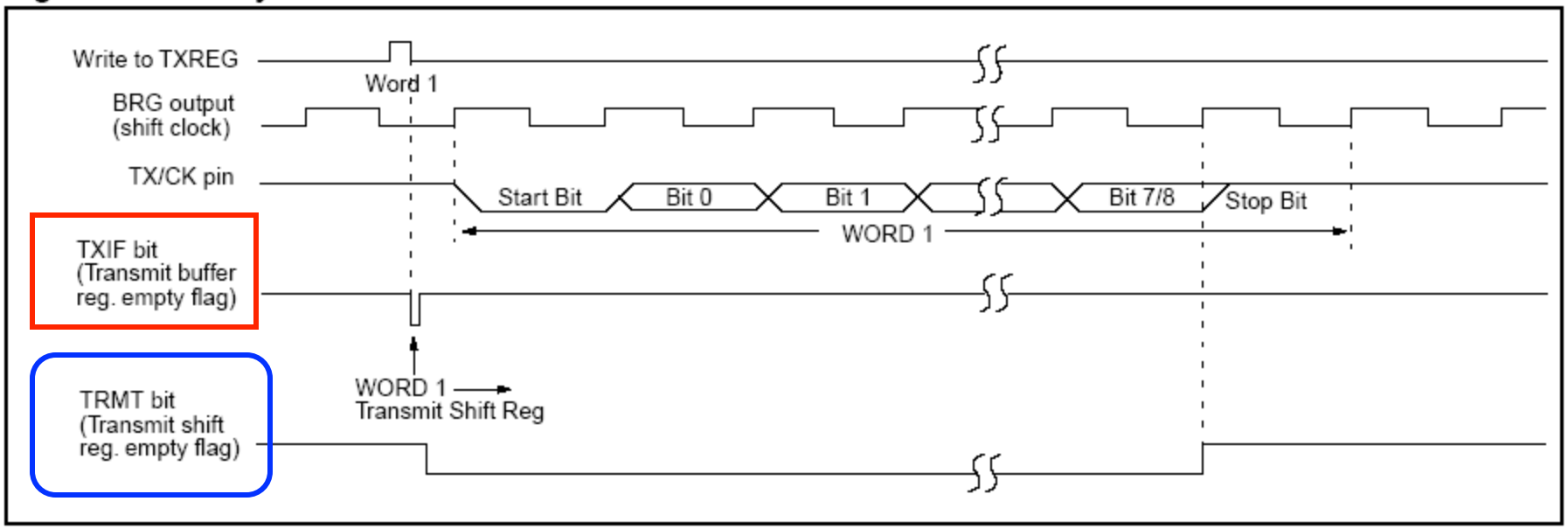
**Register 18-2: RCSTA: Receive Status and Control Register**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | U-0 | R-0 | R-0 | R-0 |
|---|---|---|---|---|---|---|---|
| SPEN | RX9 | SREN | CREN | — | FERR | OERR | RX9D |
| bit 7 | | | | | | | bit 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

# USART

## *Sending Bytes - Asynchronous Transmission*

**Figure 18-2: Asynchronous Master Transmission**




Polling Method: We can use the TRMT bit (TXSTA<1>) to check if the transmission is completed,

Or we can check the TXIF Flag to see if TXREG is empty.

When empty, write a Byte to the TXREG.

# USART

## *Subroutine TXMIT:*
## *Poll TRMT, send data when clear:*

TRMT bit gives the status of the TSR shift register.
1 = empty, 0 = shift in progress.

; Assume data to send is in 'W' before subroutine is called.

```
TXMIT: bsf     STATUS, RP0      ; TXSTA is in bank 1
LOOP:  btfss   TXSTA,TRMT       ; is TSR shift reg. empty?
       goto    LOOP             ; if not, test again
       bcf     STATUS, RP0      ; go back to bank 0
       movwf   TXREG            ; send the data
       return                   ; return to calling progam.
```

Polling

# USART

## *Subroutine TXMIT:*
## *Poll TXIF, send data when clear:*

TXIF bit gives the status of the TXREG register.
1 = empty, 0 = buffer full

```
; Assume data to send is in 'W' before subroutine is called.

TXMIT: btfss   PIR1,TXIF        ; is TXREG empty?
       goto    TXMIT            ; if not, test again
       movwf   TXREG            ; send the data
       return                   ; return to calling progam.
```

Polling

# USART

## *Other Example*

**Suppose we want to send a list of values in memory locations 0x30 through 0x59 via the serial port configured following the previous example**

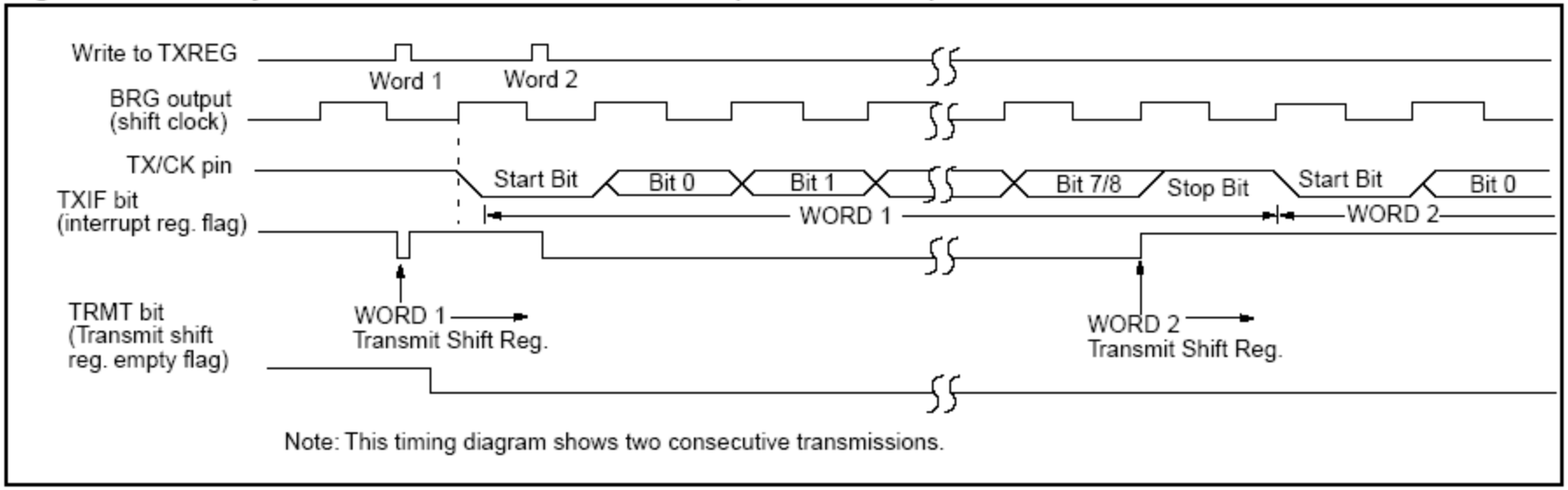
Use indirect addressing!!!!!

```
        movlw   0x30
        movwf   FSR             ; setup for indirect addressing
NEXT:   movf    INDF,W          ; get element from list
        movwf   TXREG           ; move to transmission register
        bsf     STATUS, RP0
TXPOLL: btfss   TXSTA,TRMT      ; is the buffer empty ?
        goto    TXPOLL
        bcf     STATUS, RP0
        incf    FSR             ; point to the next element
        movlw   0x60            ; end-of-list reached
        subwf   FSR,W
        btfss   STATUS, Z
        goto    NEXT            ; next element
```

# USART

## *Configuration - Asynchronous Transmission*

**Figure 18-3: Asynchronous Master Transmission (Back to Back)**




You can send two words back-to-back !

# USART

## *Configuration - Asynchronous Reception*

1. Initialize SPBRG and BRGH (Baud Rate),
2. Enable Asynchronous Serial Port, → SYNC = 0, SPEN = 1
3. Setup interrupts if desired, → RCIE = 1, GIE = 1, PIE = 1
4. If 9bits, set RX9 bit,
5. Enable reception, → CREN = 1
6. RCIF set if word received; interrupt generated if RCIE set,
7. Read RCSTA to get 9th bit,
8. If an error occurred, clear error by clearing CREN, then set CREN again.
9. Read 8-bit data from RCREG,

# USART

## *Registers - Asynchronous Reception*

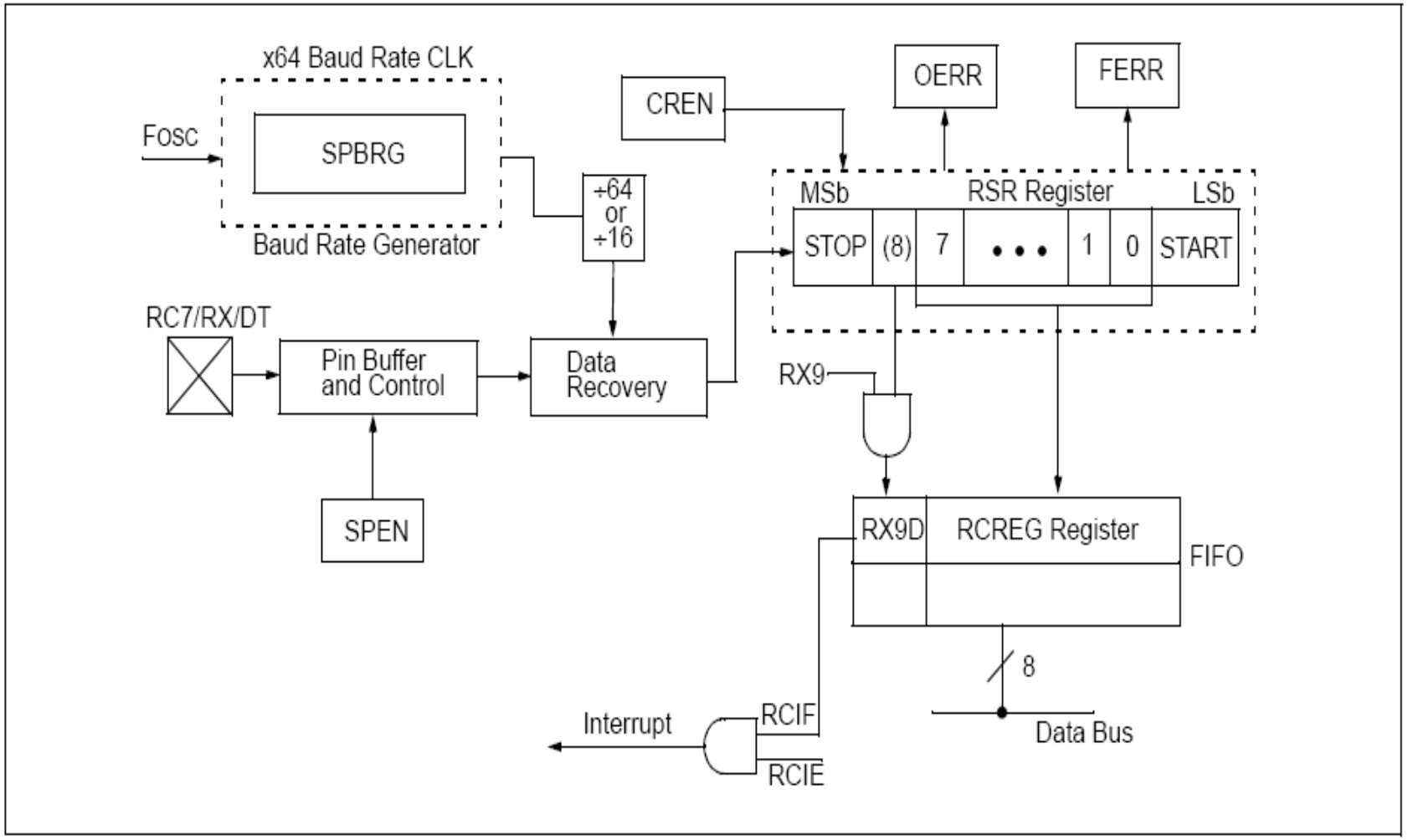
**TABLE 10-6: REGISTERS ASSOCIATED WITH ASYNCHRONOUS RECEPTION**

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on: POR, BOR | Value on all other RESETS |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0Bh, 8Bh, 10Bh,18Bh | INTCON | GIE | PEIE | T0IE | INTE | RBIE | T0IF | INTF | R0IF | 0000 000x | 0000 000u |
| 0Ch | PIR1 | PSPIF(1) | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 0000 0000 | 0000 0000 |
| 18h | RCSTA | SPEN | RX9 | SREN | CREN | — | FERR | OERR | RX9D | 0000 -00x | 0000 -00x |
| 1Ah | RCREG | USART Receive Register | | | | | | | | 0000 0000 | 0000 0000 |
| 8Ch | PIE1 | PSPIE(1) | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 0000 0000 | 0000 0000 |
| 98h | TXSTA | CSRC | TX9 | TXEN | SYNC | — | BRGH | TRMT | TX9D | 0000 -010 | 0000 -010 |
| 99h | SPBRG | Baud Rate Generator Register | | | | | | | | 0000 0000 | 0000 0000 |

Legend: x = unknown, - = unimplemented locations read as '0'. Shaded cells are not used for asynchronous reception.
**Note 1:** Bits PSPIE and PSPIF are reserved on PIC16F873/876 devices; always maintain these bits clear.

# USART - *Inside the Receiver*

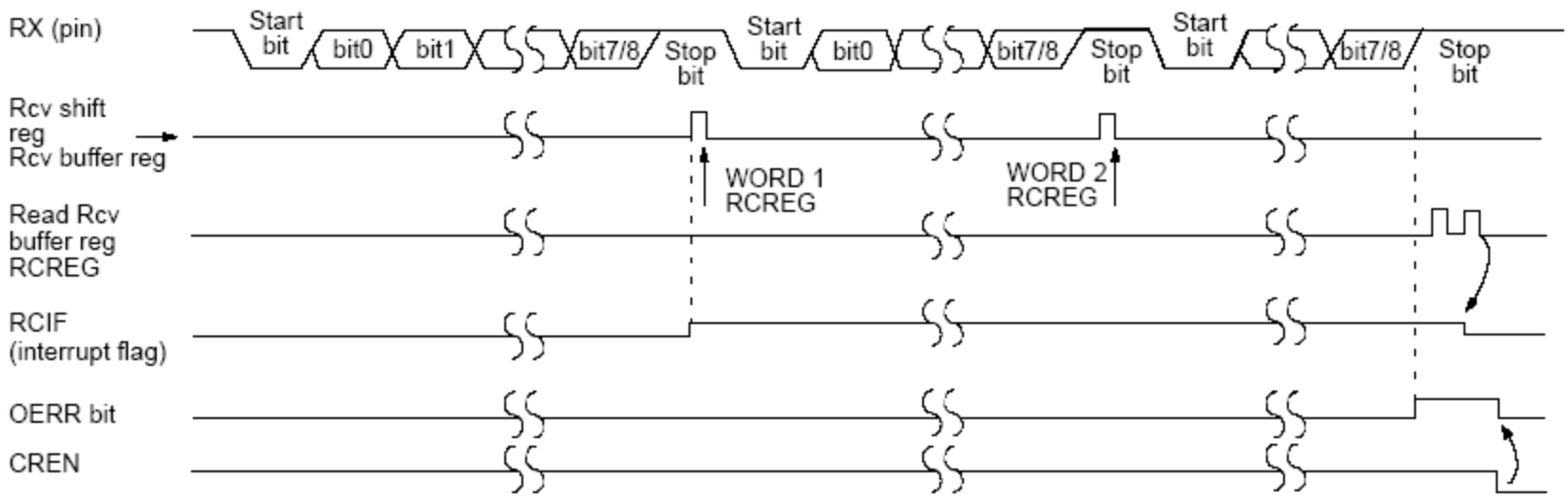
**FIGURE 10-4: USART RECEIVE BLOCK DIAGRAM**



# USART

## *Configuration - Asynchronous Reception*

**Figure 18-5: Asynchronous Reception**




Note: This timing diagram shows three words appearing on the RX input. The RCREG (receive buffer) is read after the third word, causing the OERR (overrun) bit to be set.

# USART

## *Code Example - Asynchronous Reception*

Setup as follows:
Asynchronous reception, baud rate 57,600 baud, no interrupts, 4MHz oscillator, wait until you receive a byte.

```
        bsf     STATUS, RP0         ; access bank 1
        movlw   B'10000000'         ; RC6: TX ;RC7: RX
        movwf   TRISC               ; set the port's inputs/outputs
        movlw   B'00000100'         ; setup the transmission
        movwf   TXSTA               ; 8-bits, asynch., etc.
        movlw   d'3'                ; set baud rate to 57,600
        movwf   SPBRG
        bcf     STATUS, RP0         ; access bank 0
        movlw   B'10010000'
        movwf   RCSTA               ; enable the serial port
RXWAIT: btfss   PIR1, RCIF          ; received a word yet?
        goto    RXWAIT
        movf    RCREG, W            ; read the word
        movwf   TEMP                ; store it in a temporary
                                    ; location
```

# Example Addendum

**Register 18-1: TXSTA: Transmit Status and Control Register**

| 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 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

**Register 18-2: RCSTA: Receive Status and Control Register**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | U-0 | R-0 | R-0 | R-0 |
|---|---|---|---|---|---|---|---|
| SPEN | RX9 | SREN | CREN | — | FERR | OERR | RX9D |
| bit 7 | | | | | | | bit 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

# USART

## *Code Example – Checking for Errors*

Check for Errors:
Check if an OVERRUN error or a FRAME error occurred!

```
                …
RCV:            btfss   PIR1, RCIF      ; is data in RCREG ?
                goto    RCV              ; no data yet, so try again
                btfsc   RCSTA, OERR      ; did overrun error occur?
                goto    CLROR            ; if so, go clear it.
                btfsc   RCSTA, FERR      ;  Frame error ? (FERR = 2)
                goto    CLRFE
                movf    RCREG, W         ; clears FERR, Rcvd data is lost
                return                   ; return to calling program
CLROR:          bcf     RCSTA, CREN      ; clear the cont. receive bit
                bsf     RCSTA, CREN      ; turn on cont. receive bit
                goto    RCV              ; OERR is clear, check again
CLRFE:          movf    RCREG, W         ; Clear the frame error this way
                goto    RCV              ; OERR is clear, check again
```

# USART

## *Reception – Use of Interrupts*

**TABLE 10-6: REGISTERS ASSOCIATED WITH ASYNCHRONOUS RECEPTION**

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on: POR, BOR | Value on all other RESETS |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0Bh, 8Bh, 10Bh,18Bh | INTCON | GIE | PEIE | T0IE | INTE | RBIE | T0IF | INTF | R0IF | 0000 000x | 0000 000u |
| 0Ch | PIR1 | PSPIF(1) | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 0000 0000 | 0000 0000 |
| 18h | RCSTA | SPEN | RX9 | SREN | CREN | — | FERR | OERR | RX9D | 0000 -00x | 0000 -00x |
| 1Ah | RCREG | USART Receive Register | | | | | | | | 0000 0000 | 0000 0000 |
| 8Ch | PIE1 | PSPIE(1) | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 0000 0000 | 0000 0000 |
| 98h | TXSTA | CSRC | TX9 | TXEN | SYNC | — | BRGH | TRMT | TX9D | 0000 -010 | 0000 -010 |
| 99h | SPBRG | Baud Rate Generator Register | | | | | | | | 0000 0000 | 0000 0000 |

Legend: x = unknown, - = unimplemented locations read as '0'. Shaded cells are not used for asynchronous reception.
**Note 1:** Bits PSPIE and PSPIF are reserved on PIC16F873/876 devices; always maintain these bits clear.

**IMPORTANT:** you do not have to clear the RCIF flag,
this is done automatically upon reading RCREG

# USART

## *Previous Example with Interrupts*

```
SER_INIT:   bsf     STATUS, RP0         ; access bank 1
            movlw   B'10000000'         ; RC6: TX ;RC7: RX
            movwf   TRISC               ; set the port's inputs/outputs
            movlw   B'00000100'         ; setup the transmission
            movwf   TXSTA               ; 8-bits, asynch., etc.
            movlw   d'3'                ; set baud rate to 57,600
            movwf   SPBRG
            bcf     STATUS, RP0         ; access bank 0
            movlw   B'10010000'
            movwf   RCSTA               ; enable the serial port
IRQ_INIT:   bsf     STATUS, RP0         ; access bank 1
            movlw   B'00100000'
            movwf   PIE1                ; enable serial interrupt
            bcf     STATUS, RP0         ; access bank 0
            movlw   B'11000000'         ; enable GIE & PEIE
            movwf   INTCON
MAIN:       ...
```

# USART

## *Code Example - Asynchronous Reception*

Setup as follows: Asynchronous reception, baud rate 57,600 baud, with interrupts, 4MHz oscillator

```
SER_ISR:        ; INSERT: saving of context

                btfsc   RCSTA, OERR     ; did overrun error occur?
                goto    CLROR           ; if so, go clear it.
                btfsc   RCSTA, FERR     ;  Frame error ? (FERR = 2)
                goto    CLRFE
                movf    RCREG, W        ; read the word
                movwf   TEMP            ; store it in a temporary
                goto    DONE            ; location
CLROR:          bcf     RCSTA, CREN     ; clear the cont. receive bit
                bsf     RCSTA, CREN     ; turn on cont. receive bit
                goto    DONE            ; OERR is clear, check again
CLRFE:          movf    RCREG, W        ; Clear the frame error this way
                goto    DONE

DONE:           ; INSERT: restoring of context
                retfie
```

# USART

## *Combining Transmission and Reception*

**Register 18-1: TXSTA: Transmit Status and Control Register**

| 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 |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |

Transmission-specific

Transmission-specific

**Register 18-2: RCSTA: Receive Status and Control Register**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | U-0 | R-0 | R-0 | R-0 |
|---|---|---|---|---|---|---|---|
| SPEN | RX9 | SREN | CREN | — | FERR | OERR | RX9D |
| bit 7 | | | | | | | bit 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

Reception-specific

Reception-specific

Reception-specific