UART

Universal Asynchronous Receiver/Transmitter

Motivation

- Suppose 2 digital systems need to communicate
- Communication can be
  - serial ← one bit at a time

    or
  - parallel ← several bits simultaneously

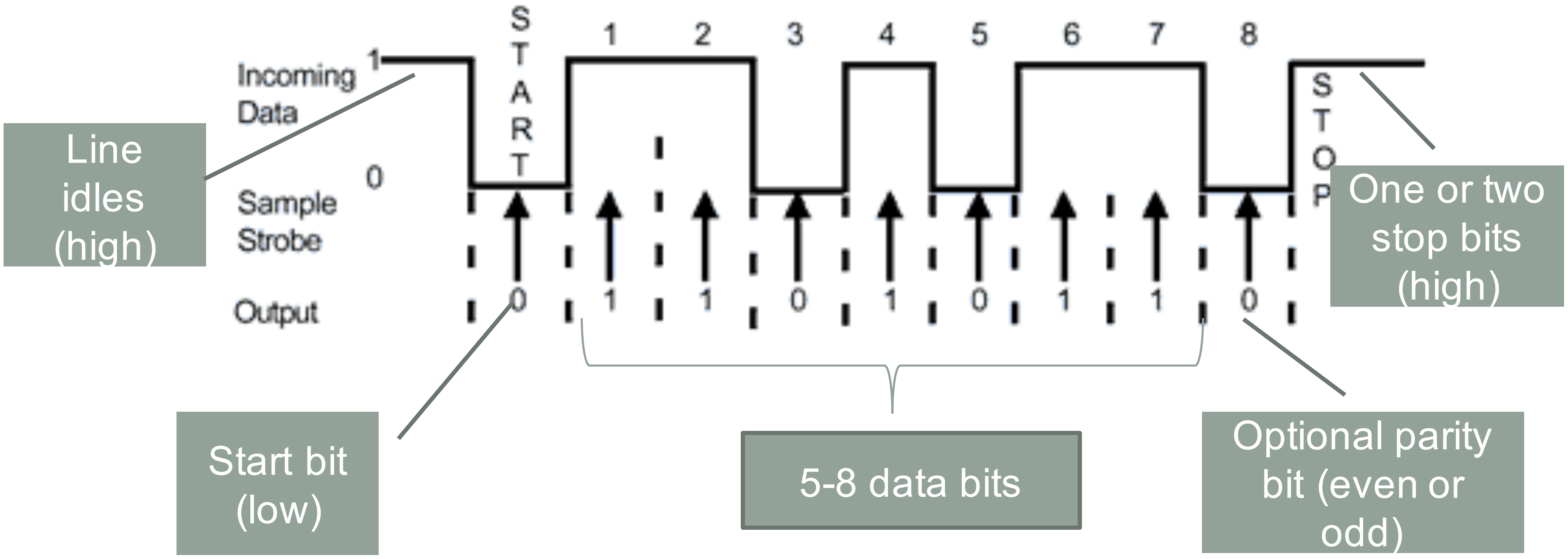
- Which alternative is faster?
- Which alternative is cheaper to implement?

Serial Transmission

- Two primary forms of serial transmission
  - Synchronous
    - A clock signal is shared by sender and receiver
    - E.g., sender provides the timing signal
  - Asynchronous
    - Sender and receiver do <u>not</u> share a clock signal
    - Often used when devices are not local to one another
    - Why might this be a good idea?

Data Frame of Asynchronous Serial Communications

- "start bit" (typically logic 0) provides a falling edge to tell receiver that data is coming
- "stop bit" (typically logic 1) marks the end of transmission and makes sure there will be a rising edge



Data Frame of Asynchronous Serial Communications (cont.)

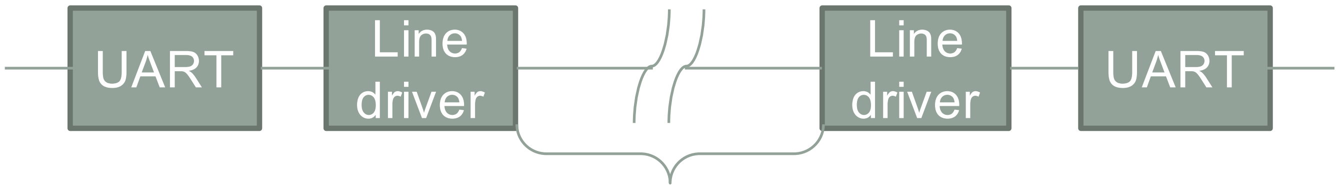
- Protocol
  - baud rate, # of data bits per frame, # of stop bits, type of parity, etc.
  - All devices must agree to it in advance
- How to receive a frame without sharing a clock?
  - The receiver has an internal clock that is much faster than the data transmission rate
  - The receiver knows when a frame is being received because of the start bit
  - The receiver “looks” at the signal at approximately halfway through the period assigned to each bit to determine if the bit is a 1or a 0.
  - Example: receiver clock is 16x faster than bit rate
    - After the falling start edge, receiver counts off 8 clocks and samples the line. If low, assume valid start and begins to counting 16 clocks to sample next bit
    - Sometimes sample bits multiple times & vote

UART controller

- UART (Universal Asynchronous Receiver/Transmitter)
  - A device that handles the receive and transmit functions of Async serial communication
  - Converts data from parallel to serial for transmission, and from serial to parallel on reception
  - Provides additional signals to indicate the state of the transmission media
  - Regulates the flow of data in the event that the remote device is not prepared to accept more data

Other concepts related to serial comm.

- A UART typically operates on computer logic voltage (e.g., 3.3 V), but . . .
- Serial interfaces to other devices frequently require a line driver
- Baud rate: Number of bits transmitted per second, including start, stop and parity
- RS-232: a very common standard (from the 1960s!) for asynchronous serial communication



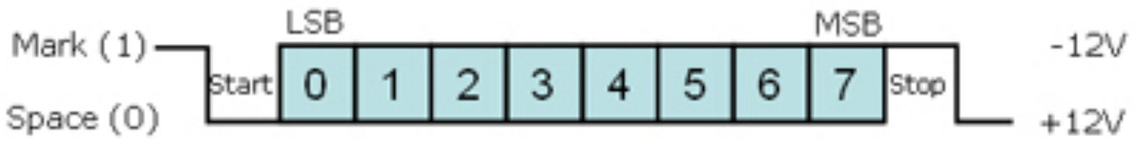
Based on standard such as RS-232

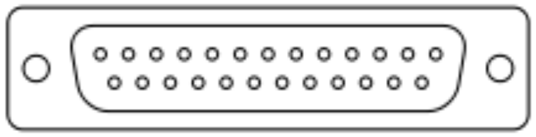
RS-232

- Defines 25 signals, but you can often get by with as few as 3: TxD, RxD, and Signal Ground

- Logic 0 = “space” =+12V
- Logic 1 = “mark” = -12 V

Voltages used in computer serial port
Cannot direct connect to 3.3V or 5V logic





A DB-25 connector as described in the RS-232 standard

(Today, DB-9 is more common)

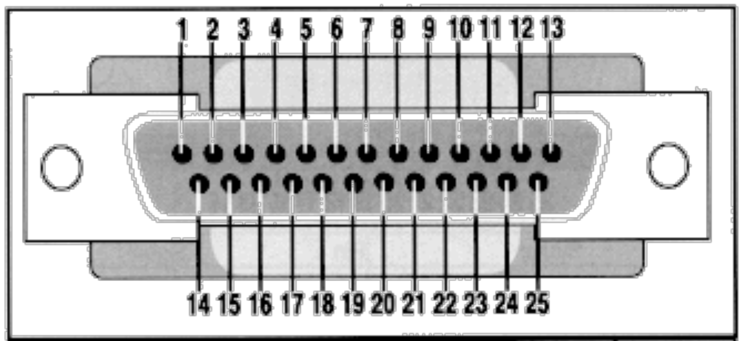
RS-232 connectors

- Data Terminal Equipment (DTE)
    - Usually, terminals and computers have male connectors with DTE pin functions
- Data Communication Equipment (DCE)
    - modems have female connectors with DCE pin functions

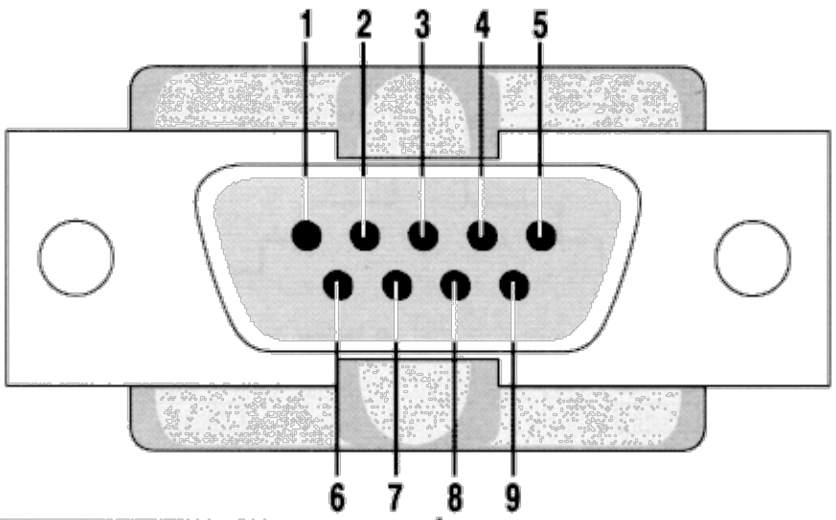
- History: originally designed to connect teletypewriters and modems.

RS-232 Interface

RS-232 (EIA Std.) applicable to the 25 pin interconnection of Data Terminal Equipment (DTE) and Data Communications Equipment (DCE) using serial binary data



Pin	Description	EIA CKT	From DCE	To DCE
1	Frame Ground	AA		
2	Transmitted Data	BA		D (Data)
3	Received Data	BB	D	
4	Request to Send	CA		C (Control)
5	Clear to Send	CB	C	
6	Data Set Ready	CC	C	
7	Signal Gnd/Common Return	AB		
8	Rcvd. Line Signal Detector	CF	C	
11	Undefined			
12	Secondary Rcvd. Line Sig. Detector	SCF	C	
13	Secondary Clear to Send	SCB	C	
14	Secondary Transmitted Data	SBA		D
15	Transmitter Sig. Element Timing	DB	T (Timing)	
16	Secondary Received Data	SBB	D	
17	Receiver Sig. Element Timing	DD	T	
18	Undefined			
19	Secondary Request to Send	SCA		C
20	Data Terminal Ready	CD		C
21	Sig. Quality Detector	CG		C
22	Ring Indicator	CE	C	
23	Data Sig. Rate Selector (DCE)	CI	C	
23	Data Sig. Rate Selector (DTE)	CH		C
24	Transmitter Sig. Element Timing	DA		T
25	Undefined			

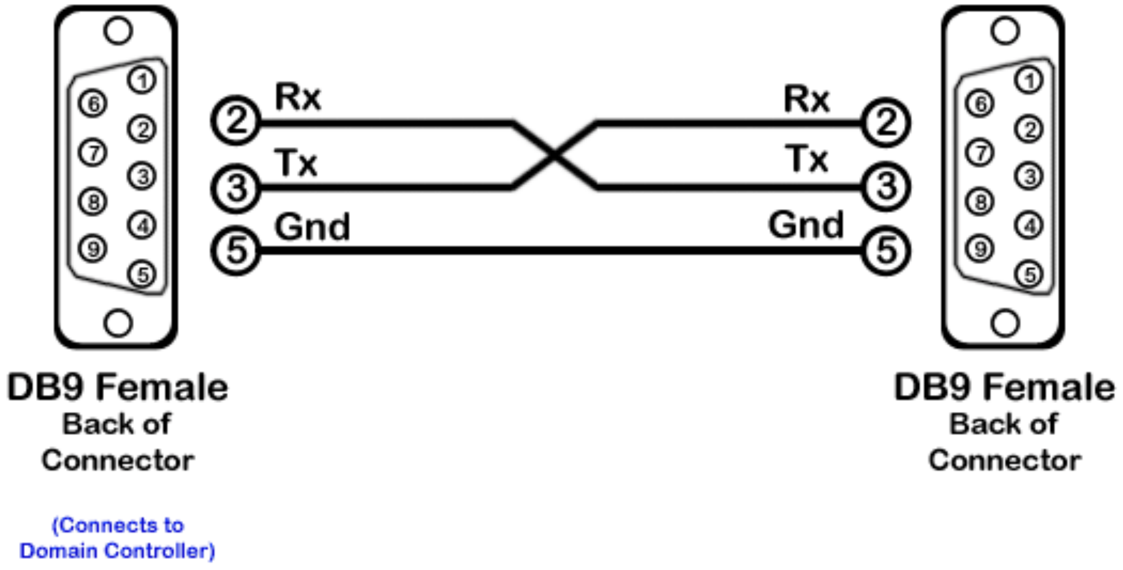


Pin	Signal	Pin	Signal
1	Data Carrier Detect	6	Data Set Ready
2	Received Data	7	Request to Send
3	Transmitted Data	8	Clear to Send
4	Data Terminal Ready	9	Ring Indicator
5	Signal Ground		

DB-9	DB-25	Function	
1	8	DCD	Data Carrier Detect
2	3	RxD	Receive Data
3	2	TxD	Transmit Data
4	20	DTR	Data Terminal Ready
5	7	GND	Ground (Signal)
6	6	DSR	Data Set Ready
7	4	RTS	Request to Send
8	5	CTS	Clear to Send
9	22	RI	Ring Indicator

RS-232 connections

- A straight-through cable is used to connect a DTE to a DCE
- A crossover (null-modem) cable is used to connect two DTE directly, without a modem in between



Flow control (handshaking)

- Purpose: prevent receiver overloading
- Types:
  - Software handshaking
    - Control bytes are sent to the sender using the standard communication lines.
  - Hardware handshaking
    - additional lines are present in the communication cable to signal handshaking conditions
    - E.g., RTS = Request To Send, CTS = Clear To Send

Software Handshaking

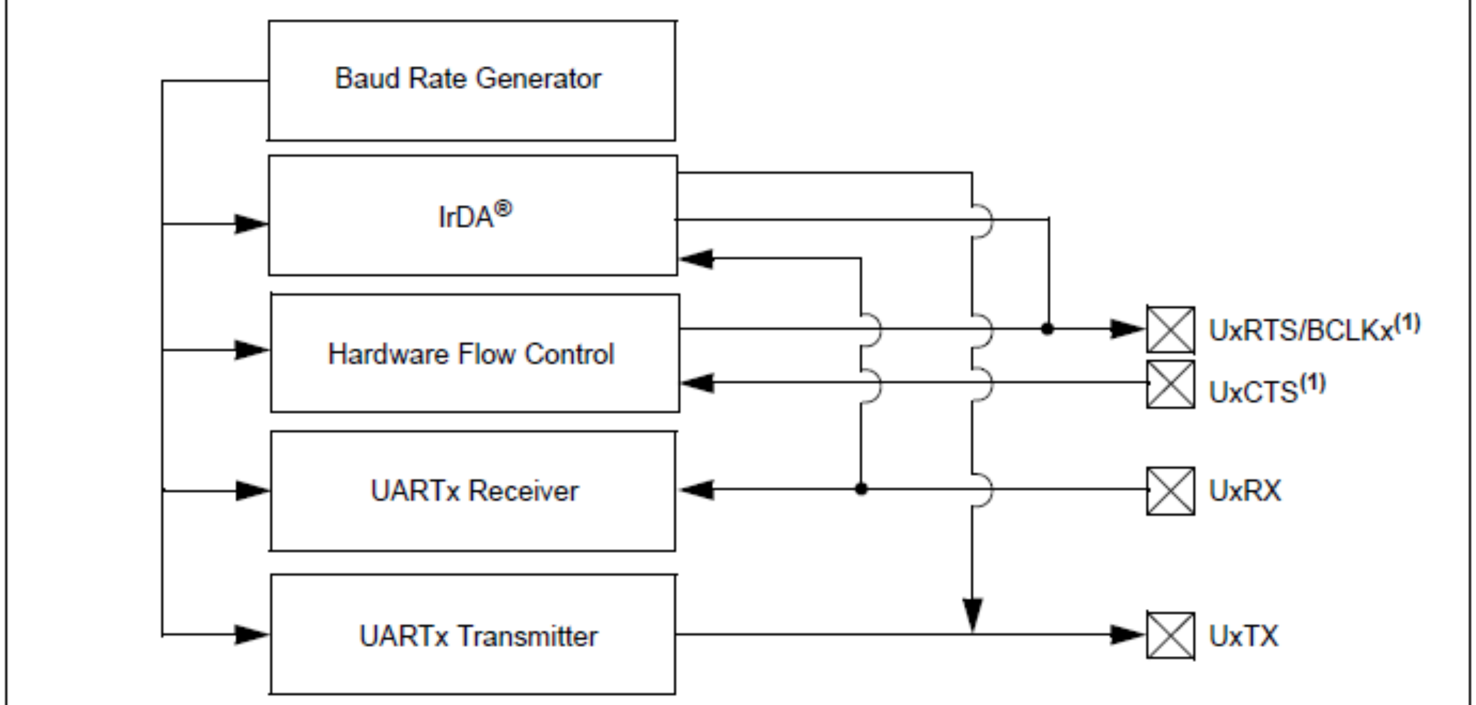
- *Software handshaking*: Control characters from receiver to transmitter
  - XON and XOFF
    - XON: start transmitting
    - XOFF: Pause transmitting
  - ACK & NAK
    - ACK: Acknowledge blocks received correctly
    - NAK: Blocks not received correctly

Hardware Handshaking

- Before sending a data, DTE set its RTS output to high. No data will be sent until DCE set CTS line high.
- DTE set DTR and DCE set DSR high to indicate they are ready to accept information respectively
- DTR/DSR are normally ON or OFF for the whole connection session, while RTS/CTS are ON or OFF for each data transmission.
- DCD (Data Carrier Ready) is used by the modem when a connection has been established with remote equipment
- RI (Ring Indicator) is used by the modem to indicate a ring signal from telephone line

UART modules on the PIC32

Figure 21-1: UART Simplified Block Diagram



Note 1: These pins are not available on some of the UART modules. Refer to the **"UART"** chapter in the specific device data sheet for more information on availability of these pins in different UART modules.

CONTROL & STATUS REGISTERS

- **UxMODE**: UARTx Mode Register
  - This register sets the operation mode of UART
- **UxSTA**: UARTx Status and Control Register
  - This register sets the interrupt mode, controls the UART transmission, and indicates various status conditions, such as transmit and receive buffer state, parity error, framing error and overflow error
- **UxTXREG**: UARTx Transmit Register
  - This register provides the data to be transmitted.
- **UxRXREG**: UARTx Receive Register
  - This register stores the received data.
- **UxBRG**: UARTx Baud Rate Register
  - Stores the baud rate value of the transmitted or received data.
- Each UART module also has associated bits for interrupt control (transmit interrupt, recv interrupt, error interrupt, interrupt priority)

UART BAUD RATE GENERATOR

- 16-bit Baud Rate Generator (BRG).
  - UxBRG register controls the period of a free-running 16-bit timer
  - **BRGH bit (UxMODE <3>) :** High Baud Rate Enable bit
    - 1 = High-Speed mode – 4x baud clock enabled
    - 0 = Standard Speed mode – 16x baud clock enabled

UART Baud Rate with BRGH=0

$$Baud\ Rate = \frac{F_{PB}}{16 \cdot (UxBRG + 1)}$$

$$UxBRG = \frac{F_{PB}}{16 \cdot Baud\ Rate} - 1$$

UART Baud Rate with BRGH=1

$$Baud\ Rate = \frac{F_{PB}}{4 \cdot (UxBRG + 1)}$$

$$UxBRG = \frac{F_{PB}}{4 \cdot Baud\ Rate} - 1$$

F_{PB} denotes the PBCLK frequency.

How to know the PBCLK frequency

- PBLCK frequency: is a division (FPBDIV=1, 2, 4, or 8) of the SYSCLK frequency

    #pragma config FPBDIV

- *SYSCLK frequency: is* external clock frequency (80Mhz) first divided by the input divider value selected (FPLLIDIV), then multiplied by the selected multiplier value (FPLLMUL) and finally divided by the selected output divider (FPLLODIV)

```
#pragma config FPLLIDIV
#pragma config FPLLMUL
#pragma config FPLLODIV
```

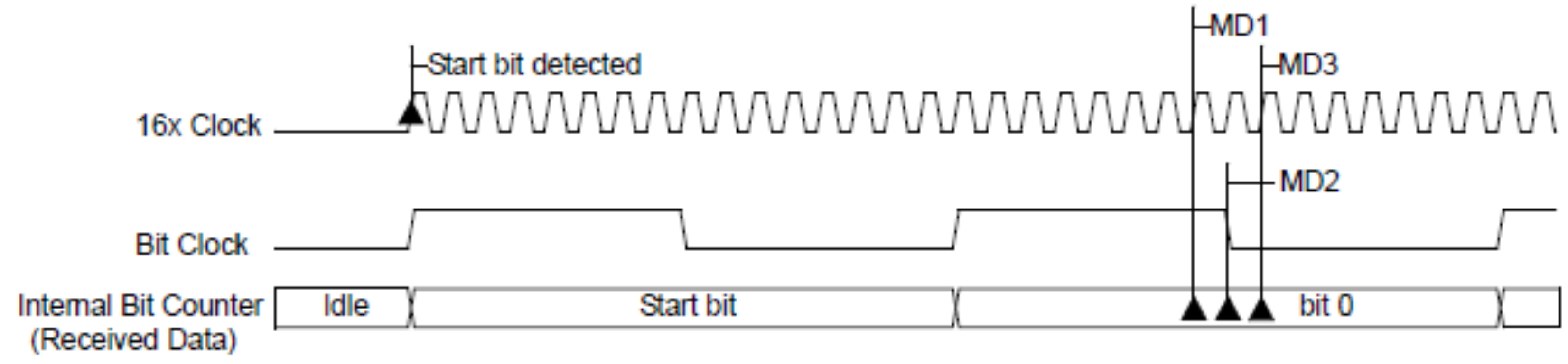
- Our Digilent board is designed to operate in either EC oscillator option, or XT oscillator option. Both are 80MHz.

```
#pragma config FNOSC = PRIPLL
#pragma config POSCMOD = EC
```

Baud clock rate and data bit detection

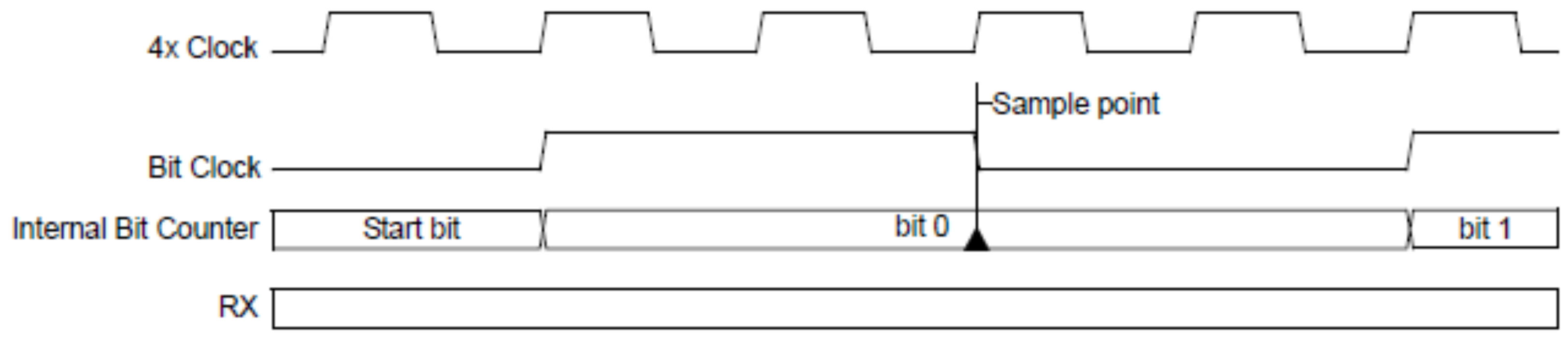
- 16x Clock mode (more robust)
  - each bit of the received data is 16 clock pulses wide.
  - for each incoming data bit, the bit is sampled at 7th, 8th and 9th rising edges of the clock and majority vote is used.
- 4x Clock mode
  - each bit is four clock pulses wide.
  - each received bit is sampled at the half-bit point in time

Figure 21-5: **16x Clock Mode with Majority Detection**



Note: In 16x Clock mode, each bit is sampled at 7th, 8th and 9th rising edges of the clock.

Figure 21-6: **4x Clock Mode without Majority Detection**



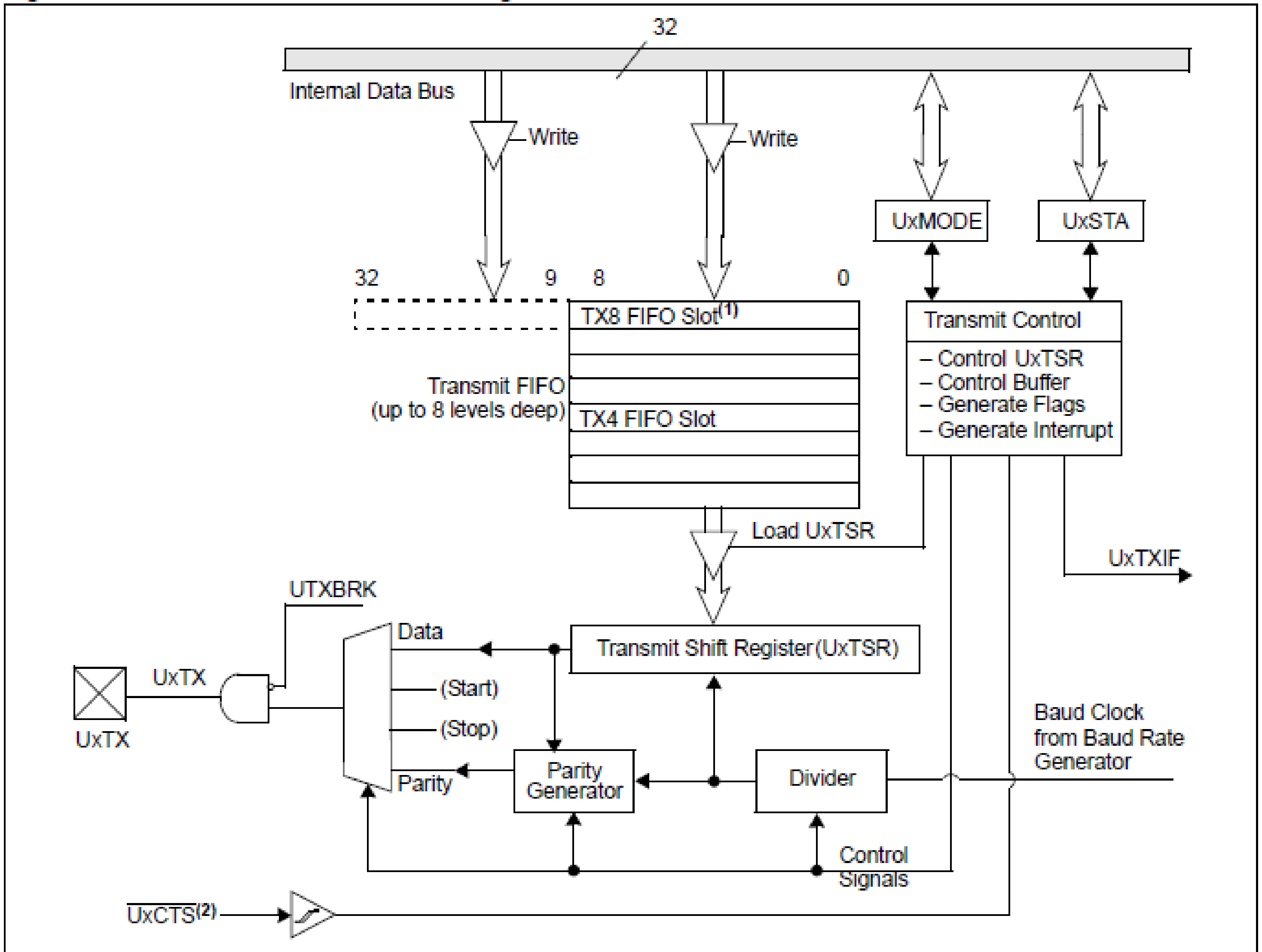
Note: In 4x Clock mode, the sampling occurs only at the one-half bit width.

Enable/Disable UART

- Enable UART
  - ON bit (UxMODE<15>) =1 enable UART module
  - UTXEN (UxSTA<10>)=1 enable transmitter
  - URXEN(UxSTA<12>)=1 enable receiver
- Disable UART
  - ON bit (UxMODE<15>) =0
  - Default state after reset

UART Transmitter Block Diagram

Figure 21-3: UART Transmitter Block Diagram(1)



Note 1: Refer to the **"UART"** chapter in the specific device data sheet for availability of the 8-level-deep FIFO.

2: Refer to the **"Pin Diagrams"** section in the specific device data sheet for availability of the $\overline{\text{UxCTS}}$ pin.

UART Transmitter

- **Transmit Shift register (UxTSR):**
  - Obtains its data from the transmit FIFO buffer, UxTXREG.
  - is not loaded until the Stop bit is transmitted from the previous load.
- **Transmit Buffer (UxTXREG):**
  - 9 bits wide and 8 levels deep.
  - The UTXBF status bit (UxSTA<9>) is set whenever UxTXREG is full.
- **UTXEN bit (UxSTA<10>)**: Enable transmission
- **PDSEL<1:0> bits (UxMODE<2:1>):**
  - 11 = 9-bit data, no parity
  - 10 = 8-bit data, odd parity
  - 01 = 8-bit data, even parity
  - 00 = 8-bit data, no parity

UART Transmitter

- PDSEL<1:0>(UxMODE<2:1>: Parity and Data Selection bits
  - 11 = 9-bit data, no parity
  - 10 = 8-bit data, odd parity
  - 01 = 8-bit data, even parity
  - 00 = 8-bit data, no parity
- STSEL(UxMODE<0>): Stop Selection bit
  - 1 = 2 Stop bits
  - 0 = 1 Stop bit

Transmit Interrupt Mode

- **UTXISEL<1:0>: (**UxSTA<15:14>). TX Interrupt Mode Selection bits
  - For 8-level deep FIFO UART modules:
    - 11 = Reserved, do not use
    - 10 = Interrupt is generated and asserted while the transmit buffer is empty
    - 01 = Interrupt is generated and asserted when all characters have been transmitted
    - 00 = Interrupt is generated and asserted while the transmit buffer contains at least one empty space
- **TRMT bit (UxSTA<8>):** is set when the UxTSR register is empty. No interrupt logic is tied to this bit, so the user has to poll this bit

UART Transmission Setup

1. Initialize the UxBRG register for the appropriate baud rate

2. Set the number of data and Stop bits, and parity selection by writing to the PDSEL<1:0> bits (UxMODE<2:1>) and STSEL bit (UxMODE<0>).

3. If transmit interrupts are desired, enable interrupt, specify priority/subpriority. Also, select the Transmit Interrupt mode by writing to the UTXISEL bits (UxSTA<15:14>).

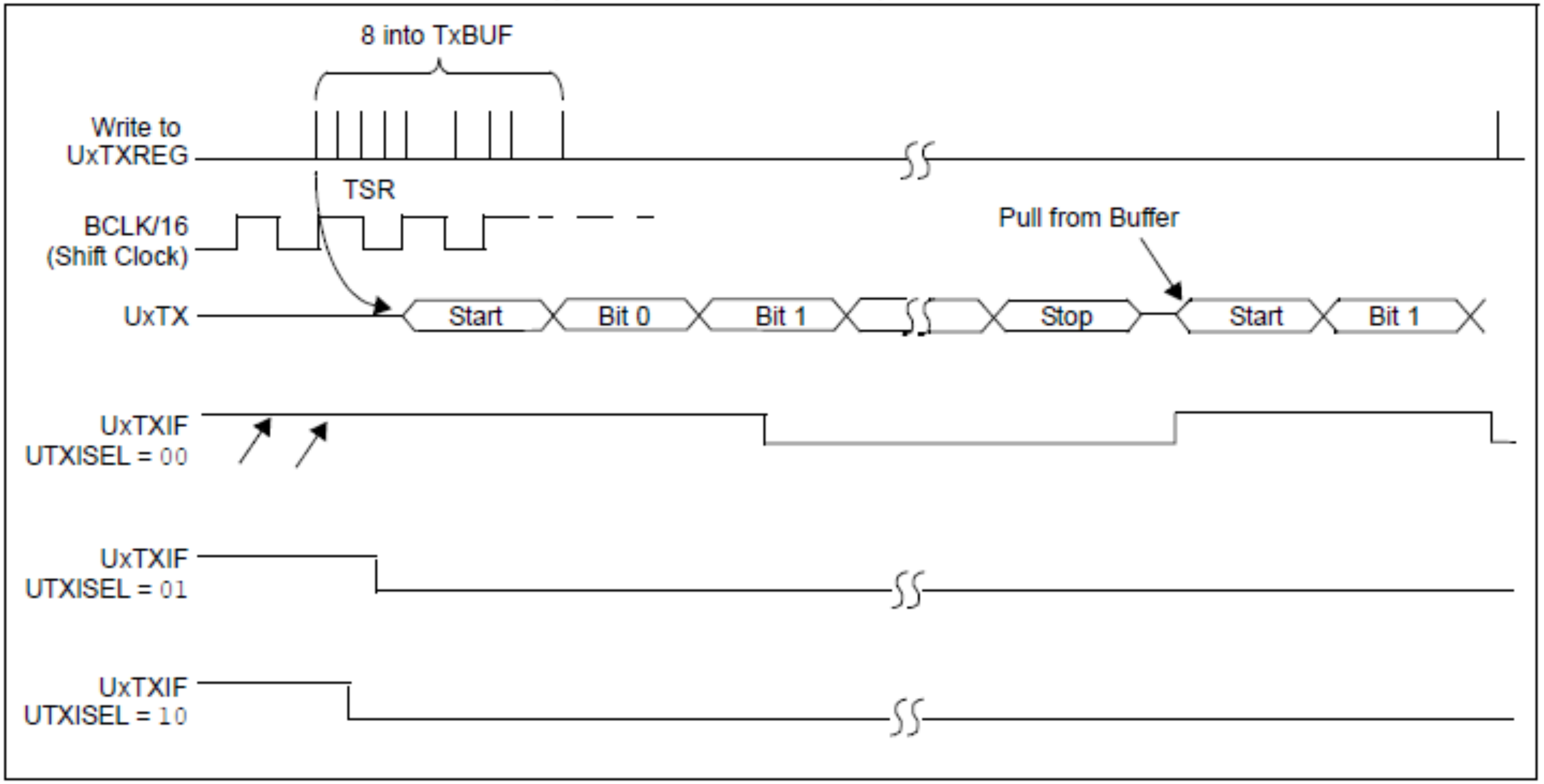
4. Enable the transmission by setting the UTXEN bit (UxSTA<10>).

5. Enable the UART module by setting the ON bit (UxMODE<15>).

6. Load data to the UxTXREG register (starts transmission).

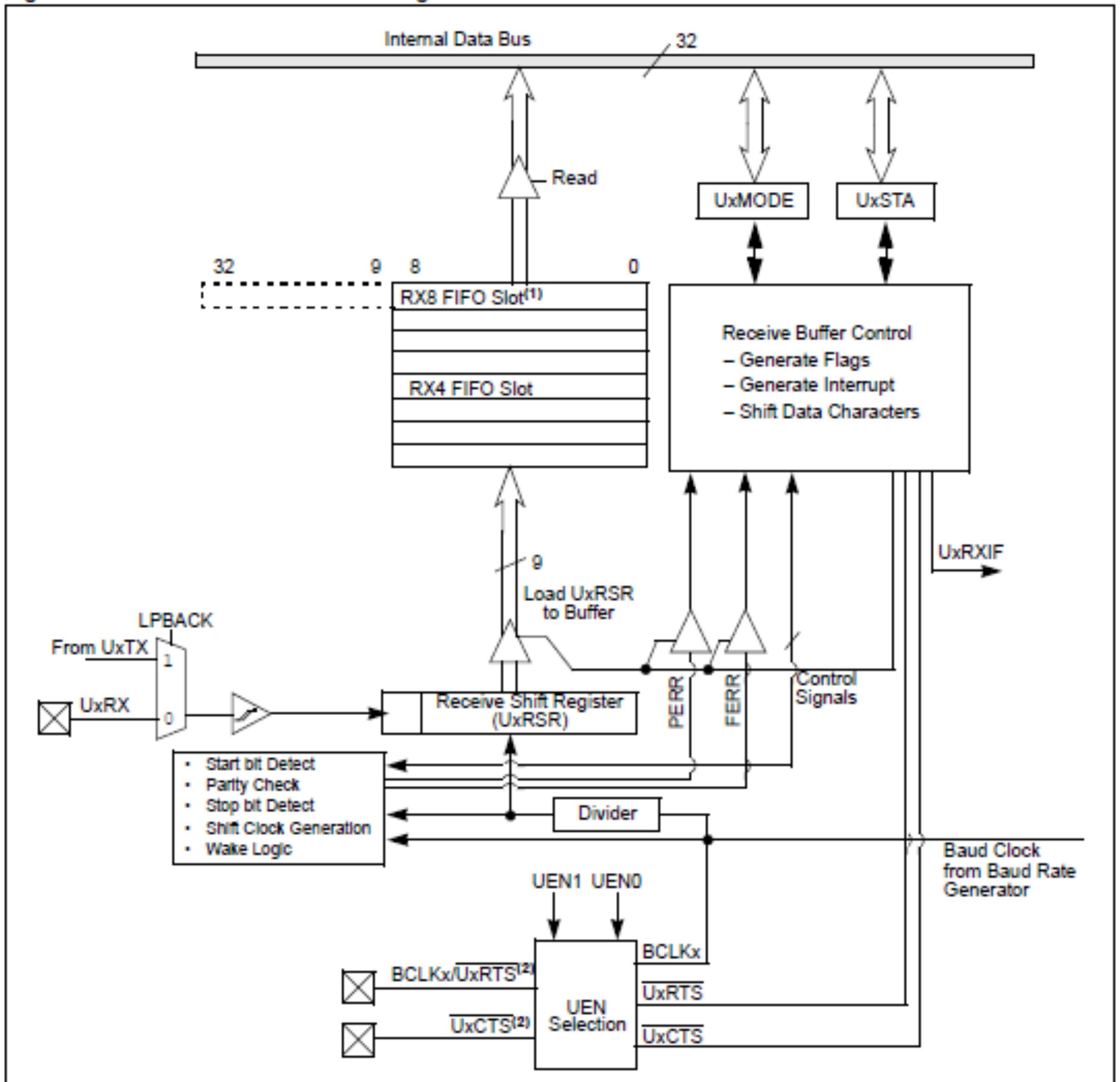
UART Transmission

FIGURE 19-3: TRANSMISSION (8-BIT OR 9-BIT DATA)



UART Receiver

Figure 21-7: UART Receiver Block Diagram[1]



Note 1: Refer to the "**UART**" chapter in the specific device data sheet for availability of the 8-level-deep FIFO.

2: Refer to the "**Pin Diagrams**" section of the specific device data sheet for availability of the $\overline{\text{UxRTS}}$ and $\overline{\text{UxCTS}}$ pins.

UART Receiver

- **Receive (Serial) Shift register (UxRSR)**:
  - received data on UxRX is sampled and stored in UxRSR
  - After stop bit is sampled, data in UxRSR is transferred to the receive FIFO
- **URXEN bit (UxSTA<12>):**
  - set to enable reception
- **Receive Buffer (UxRXREG):**
  - 9 bit wide, 8 levels deep FIFO
- **Overrun error bit OERR (UxSTA<1>):**
  - Set when FIFO is full and UxRSR receives a new character
  - OERR bit must be cleared to enable data transfer to FIFO
  - The data in the receive FIFO should be read prior to clearing the OERR bit. The FIFO is reset when the OERR bit is cleared, which causes all data in the buffer to be lost.
- **Framing Error Status bit, FERR (UxSTA<2>):**
  - set when the received state of the Stop bit is incorrect.
- **Parity Error Status bit, PERR (UxSTA<3>):**
  - set if a parity error exists in the data word at the top of the FIFO buffer

UART Receiver

- PDSEL<1:0>(UxMODE<2:1>: Parity and Data Selection bits
  - 11 = 9-bit data, no parity
  - 10 = 8-bit data, odd parity
  - 01 = 8-bit data, even parity
  - 00 = 8-bit data, no parity
- STSEL(UxMODE<0>): Stop Selection bit
  - 1 = 2 Stop bits
  - 0 = 1 Stop bit

UART Receive Interrupt

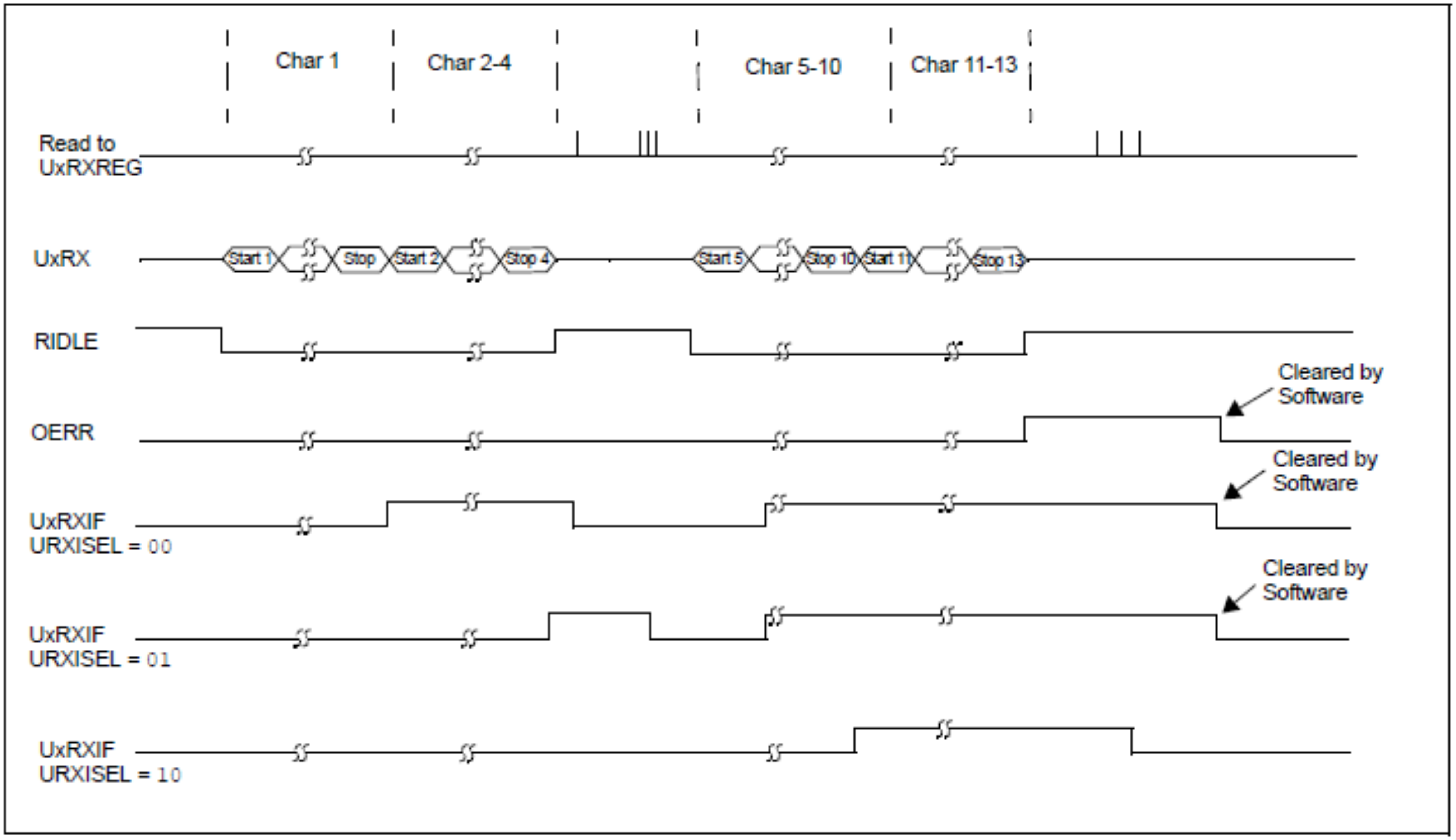
- URXISEL<1:0> Receive Interrupt Mode Selection bit (UxSTA<7:6>):
  - For 8-level deep FIFO UART modules:
    - 11 = Reserved; do not use
    - 10 = Interrupt flag bit is asserted while receive buffer is 3/4 or more full
    - 01 = Interrupt flag bit is asserted while receive buffer is 1/2 or more full
    - 00 = Interrupt flag bit is asserted while receive buffer is not empty
- RIDLE bit (UxSTA<4>)
  - Is set when UxRSR register is empty. No tied interrupt with this bit
- URXDA bit (UxSTA<0>)
  - is set when receive buffer is not empty

UART Reception Setup

1. Initialize the UxBRG register for the appropriate baud rate
2. Set the number of data and Stop bits, and parity selection by writing to the PDSEL<1:0> (UxMODE<2:1>) and STSEL (UxMODE<0>) bits.
3. If interrupts are desired, enable interrupt, specify priority/subpriority. Also, select the Receive Interrupt mode by writing to the URXISEL<1:0> bits (UxSTA<7:6>).
4. Enable the UART receiver by setting the URXEN bit (UxSTA<12>).
5. Enable the UART module by setting the ON bit (UxMODE<15>).
6. Receive interrupts are dependent on the URXISEL<1:0> bit settings. If receive interrupts are not enabled, the user can poll the URXDA bit (UxSTA<0>). The UxRXIF bit should be cleared in the software routine that services the UART receive interrupt.
7. Read data from the receive buffer. If 9-bit transmission is selected (for multiprocessor case), read a word; otherwise, read a byte. The URXDA bit is set whenever data is available in the buffer.

UART Reception

FIGURE 19-2: UART RECEPTION



Initialization sample

Example 21-2: 8-bit Transmit/Receive (UART1)

```
U1BRG   =   BaudRate;           // Set Baud rate

U1STA   =   0;
U1MODE  =   0x8000;             // Enable UART for 8-bit data
                                // No Parity, 1 Stop bit
U1STASET = 0x1400;              // Enable Transmit and Receive
```

UxCTS & UxRTS

- **UEN<1:0>:** UARTx Enable bits
  - 11 = UxTX, UxRX and UxBCLK pins are enabled and used; UxCTS pin is controlled by corresponding bits in the PORTx register
  - **10 = UxTX, UxRX, UxCTS and UxRTS pins are enabled and used**
  - **01 = UxTX, UxRX and UxRTS pins are enabled and used; UxCTS pin is controlled by corresponding bits in the PORTx register**
  - 00 = UxTX and UxRX pins are enabled and used; UxCTS and UxRTS/UxBCLK pins are controlled by corresponding bits in the PORTx register

- UxCTS pin:
  - UxCTS is an input pin
  - UxCTS = 1, the transmitter loads data in the Transmit Shift register, but will not initiate a transmission.
  - Transmission begins when the UxCTS pin is low

UxRTS

- UxRTS pin is always an output pin
- **RTSMD:** Mode Selection for UxRTS Pin bit(2)
  - 1 = UxRTS pin is in Simplex mode:
    - UxRTS pin is asserted (driven low) whenever the data is available to transmit (TRMT = 0). UxRTS pin is deasserted (driven high) when the transmitter is empty (TRMT = 1).
  - 0 = UxRTS pin is in Flow Control mode
    - The UxRTS pin is driven low whenever the receiver is ready to receive data

Figure 21-12: **$\overline{\text{UxRTS}}$/$\overline{\text{UxCTS}}$ Handshake for DTE-DCE (RTSMD = 1, Simplex Mode)**

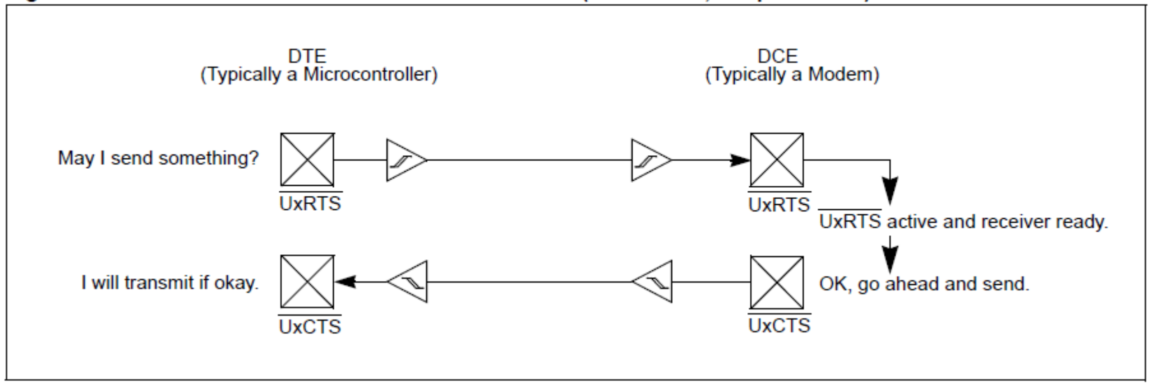
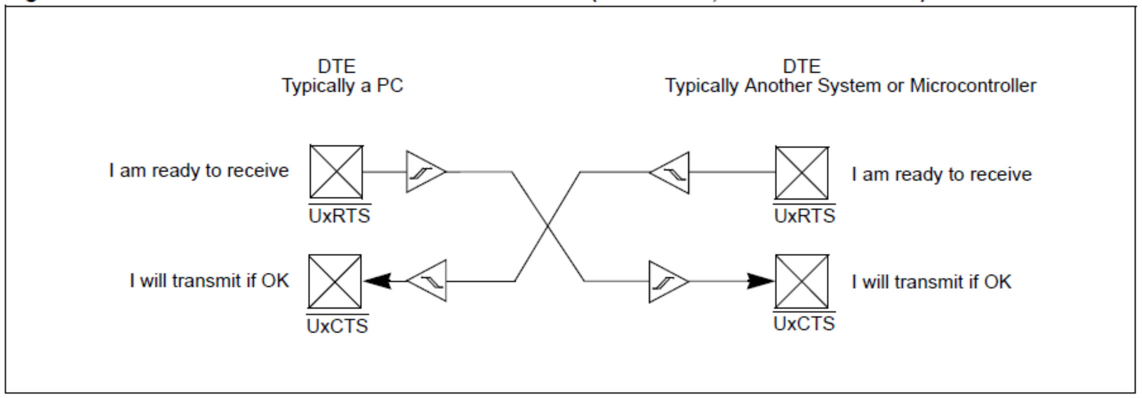


Figure 21-11: **$\overline{\text{UxRTS}}$/$\overline{\text{UxCTS}}$ Flow Control for DTE-DTE (RTSMD = 0, Flow Control Mode)**



UARTs on the Digilent board

- Our Digilent board allows use of two UART interfaces out of the six supported by PIC32MX795
  - 2-wire interface: receive (RxD) and transmit (TxD) pins.
  - 4-wire interface: RxD, TxD, RTS, CTS

- UART1 on Pmod connector JE
  - U1CTS JE-01
  - U1TX JE-02
  - U1RX JE-03
  - U1RTS JE-04

- UART2 on Pmod connector JF
  - U2CTS JF-01
  - U2TX JF-02
  - U2RX JF-03
  - U2RTS JF-04

UARTs on the Digilent board

- IMPORTANT:
  UART1 can be also accessed through a USB connector

- Near the board’s power switch is the “USB UART serial port”, called “USB connector J2” in the reference manual

- A USB-to-UART chip allows connection to external USB devices, such as your laptop

Plib functions – described in uart.h
(Plib manual is out of date)

- void **UARTConfigure** (UART_MODULE id, UART_CONFIGURATION flags)
- void **UARTSetFifoMode** (UART_MODULE id, UART_FIFO_MODE mode)
- void **UARTSetLineControl** (UART_MODULE id,
  UART_LINE_CONTROL_MODE mode)
- UINT32 **UARTSetDataRate** (UART_MODULE id, UINT32 sourceClock,
  UINT32 dataRate)
- UINT32 UARTGetDataRate(UART_MODULE id, UINT32 sourceClock)
- void UARTSetAddress (UART_MODULE id, BYTE address, BOOL watch)
- void UARTWatchForAddress (UART_MODULE id, BOOL watch)
- void **UARTEnable** (UART_MODULE id, UART_ENABLE_MODE mode)

Plib functions – described in uart.h

- BOOL **UARTTransmitterIsReady** (UART_MODULE id)
- void **UARTSendDataByte** (UART_MODULE id, BYTE data)
- BOOL **UARTTransmissionHasCompleted** (UART_MODULE id)

- BOOL UARTReceivedDataIsAvailable (UART_MODULE id)
- BYTE UARTGetDataByte (UART_MODULE id)

SUMMARY

- Serial communication between digital devices is popular because ________?_________.
- RS-232 is a well-known serial comm standard
- A "UART" is a peripheral device that interfaces between a serial comm link and the (parallel) data bus of a CPU
- Our PIC32 chip contains 6 "UART" modules, and our Digilent board interfaces 2 of them to the outside world (one of those through a USB connector)