

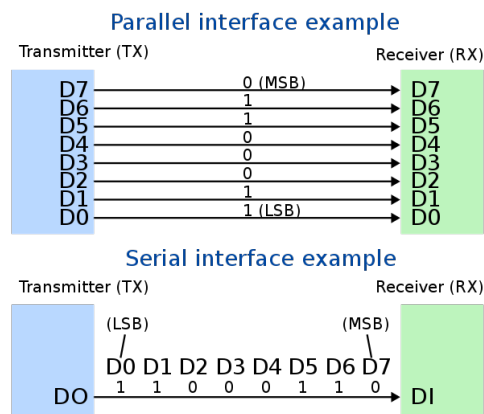
VE373 Recitation Class

Week 7

2022.06.25

L11 — Serial Communication: UART

1. Serial & Parallel Communication



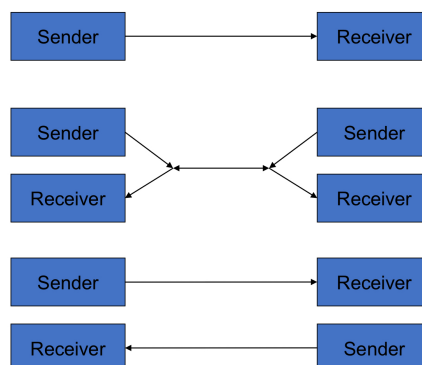
Why not parallel communication?

- Parallel I/O uses many signal pins - expensive
- Synchronization problem over longer distance
- High speed, but many applications do not need high data rate

2. Serial Modules

- UART (Universal Asynchronous Receiver and Transmitter)
- SPI (Synchronous Peripheral Interface)
- I2C (Inter-Integrated Circuit)
- USB (Universal Serial Bus)

3. Simplex and Duplex Transmission



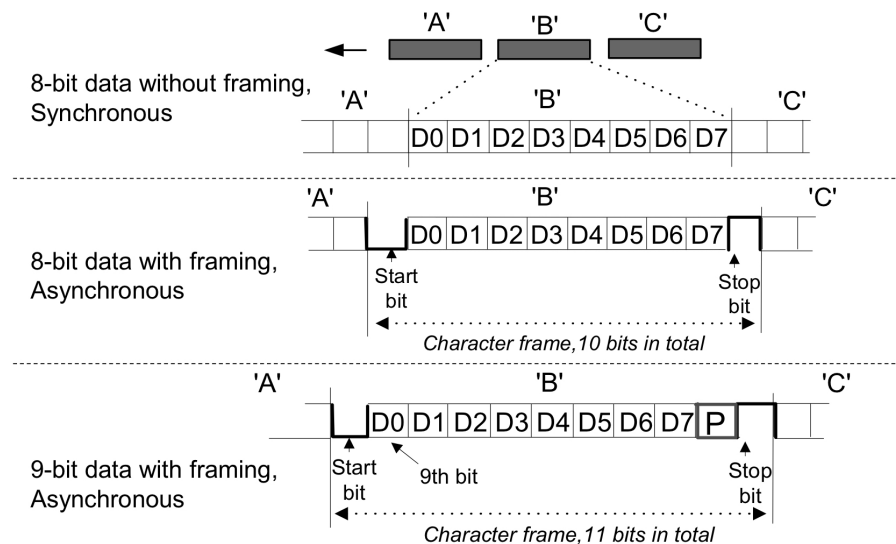
- **Simplex**: sender can send the data but the sender can't receive the data. It is a unidirectional communication.
- **Half-Duplex**: sender can send the data and also can receive the data one at a time. It is two-way directional communication but one at a time.

- **Full-Duplex:** sender can send the data and also can receive the data simultaneously. It is two-way directional communication simultaneously.

4. Transmission Timing

- **Asynchronous:** clock signal not transmitted, receiver and transmitter have their own clocks.
- **Synchronous:** clock signal transmitted.

5. Data Framing

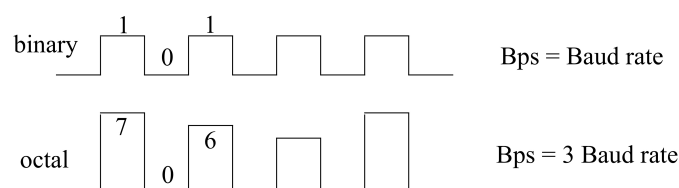


6. Data Transfer Rate

Two measurements:

- **Bps:** bits per second
- **Baud rate:** signaling event per second

One signaling event is possible to transmit multiple bits, i.e., $\text{Bps} \geq \text{Baud rate}$.



7. UART in PIC32

- Full-duplex, 8-bit or 9-bit data transmission
- Even, Odd or No Parity options (for 8-bit data)
- One or two Stop bits
- 8-level deep First-In-First-Out (FIFO) transmit data buffer
- 8-level deep FIFO receive data buffer
- Up to 6 UART modules
 - UART1A, 1B, 2A, 2B, 3A, 3B
 - UARTxB only available when UARTxA is in simple operation mode
- Supports multiple communication protocols
 - RS-232, RS-485, IrDA, ...

10. Baud Rate Generator

The transmit and receive clock for the UART is generated by the baud rate generator (UxBRG). UxBRG specifies the period of a free running 16-bit timer.

- **Standard Speed Mode** (BRGH = 0)

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

- **High-Speed Mode** (BRGH = 1)

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

11. Transmit Buffer (UxTXREG)

- 9 bits wide, and up to 8 levels deep.
- Together with the Transmit Shift registers (UxTSR), the user can have up to a 9-level-deep buffer.
- The UTXBF status bit (UxSTA<9>) is set whenever the buffer is full.
- If a user attempts to write to a full buffer, the new data will not be accepted into the FIFO.

12. Transmission Interrupt

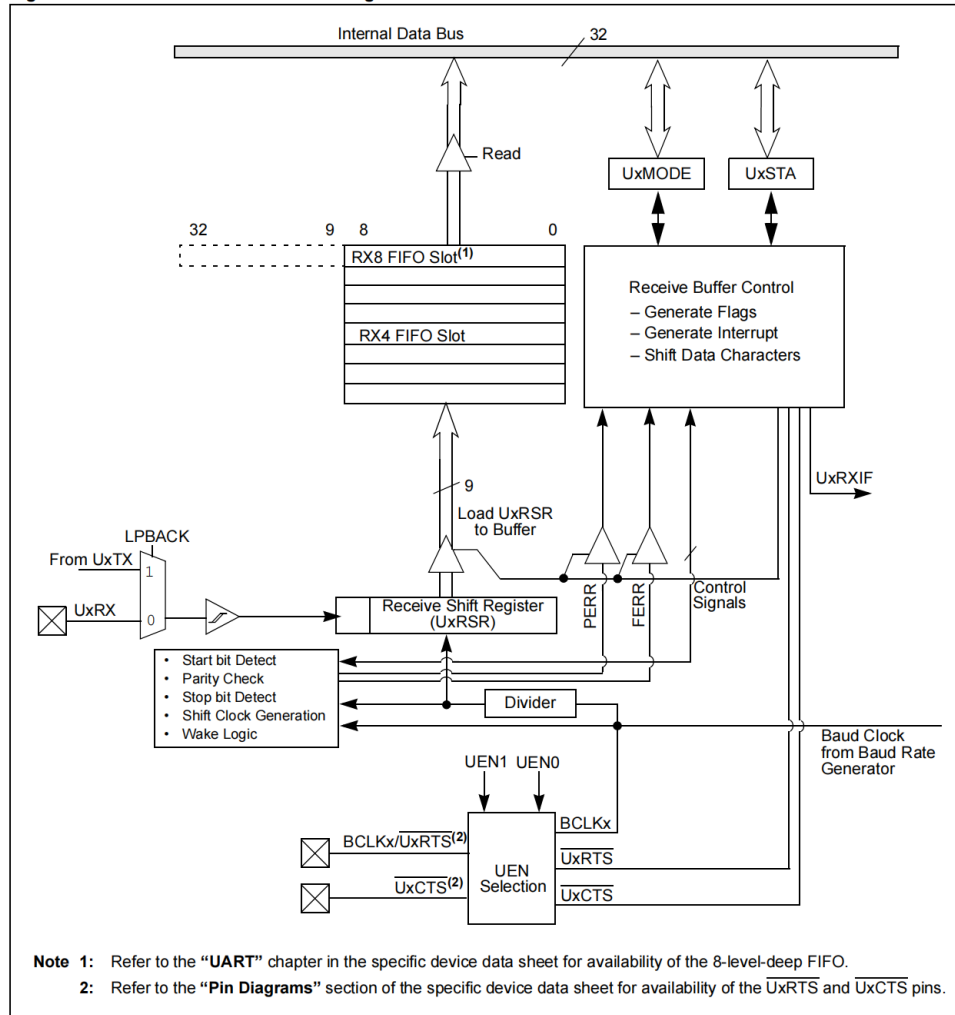
- UxTXIF is generated as soon as the UART module is enabled
- UTXISEL (UxSTA<15:14>) determines when UxTXIF flags
- UxTXIF for status of the UxTXREG
- TRMT bit (UxSTA<8>) for status of the UxTSR register
- Persistent interrupt: data must be read from, or written to, the UxRXREG or UxTXREG registers to obtain the number of data to receive/transmit below the level specified by the URXISEL<1:0> and UTXISEL<1:0> bits.

13. 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
 1. Set UxTXIE.
 2. Specify the interrupt priority and subpriority.
 3. 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>), which also sets the UxTXIF bit. The UxTXIF bit should be cleared in the software routine that services the UART transmit interrupt.
5. Enable the UART module by setting the ON bit (UxMODE<15>).
6. Load data to the UxTXREG register (starts transmission).

14. Receiver block diagram

Figure 21-7: UART Receiver Block Diagram⁽¹⁾



15. Receiver Error Handling (Not covered)

If the FIFO is full and a new character is fully received into the UxRSR register, the overrun error bit, OERR (UxSTA<1>), is set. The word in UxRSR register is not kept, and further transfers to the receive FIFO are inhibited as long as the OERR bit is set. The user must clear the OERR bit in software to allow further data to be received.

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.

16. Reception Interrupt

- UxRXIF, determined by RXISEL<1:0> (UxSTA<7:6>)
- Before clearing the corresponding UxRXIF flag bit, the user application must ensure that the interrupt condition specified by the URXISEL control bits is no longer true.
- RXDA (and UxRXIF) indicate the status of the UxRXREG
- RXDA (UxSTA<0>) indicates whether the receive buffer has data or is empty. Read-only
- RIDLE (UxSTA<4>) shows the status of the UxRSR register. Read-only

17. 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

1. Set the `UxRXIE` bit.
2. Specify the priority and subpriority for the interrupt.
3. 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>`).
7. Read data from the receive buffer.
 - If 9-bit transmission is selected, read a word; otherwise, read a byte.
 - The `URXDA` bit is set whenever data is available in the buffer.