UART

(UNIVERSAL ASYNCHRONOUS RECEIVER/TRANSMITTER)

UART PROTOCOL – OVERVIEW

- It is a serial communication protocol used for data transfer between two devices.
- It sends data bits one by one, from the least significant to the most significant, framed by start and stop bits so that precise timing is handled by the communication channel.
- It was one of the earliest computer communication devices, used to attach teletypewriters for an operator console.

PRACTICAL APPLICATIONS

- Data Transmission:

The microcontroller reads the temperature from the sensor (e.g., via an analog-to-digital converter or a digital sensor like DS18B20) and formats it into a serial data stream (e.g., "Temperature: 25°C").

- Human Interface:

The PC receives this data through a terminal, allowing you to observe and log temperature changes in real time, which is useful for monitoring or logging purposes.

- System Integration:

It enables a simple way to connect the embedded system to a more powerful computing device for display or further processing.

KEY FEATURES

Supports three communication modes:

- Simplex: One-way data transfer.
- Half-Duplex: Two-way, but not at the same time.
- Full-Duplex: Two-way, simultaneous transfer.

Asynchronous communication:

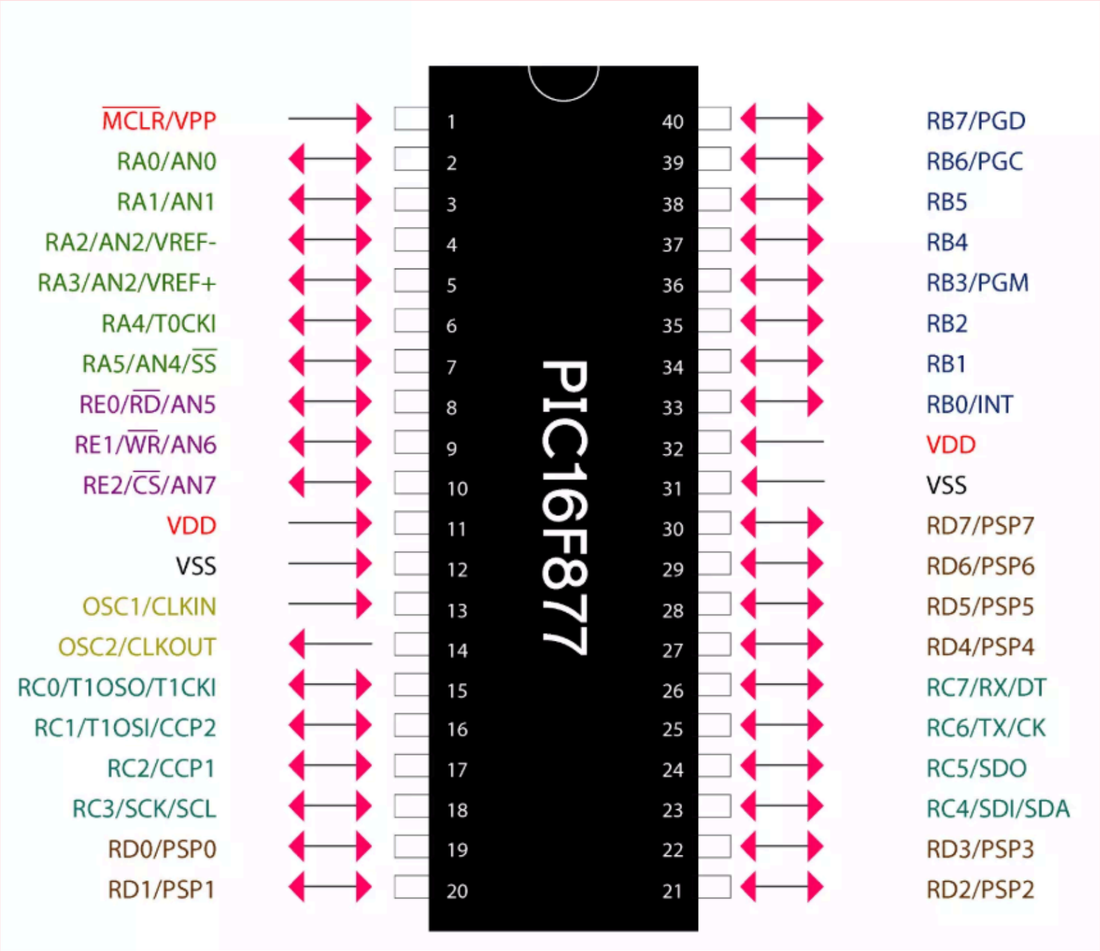
No shared clock between transmitter and receiver.

- UART uses a serial method where data bits are sent sequentially over a single channel using two wires (Tx and Rx) and a common ground.

- Data timing is controlled by the baud rate, which defines transmission speed in bits per second (bps). Both devices must share the same baud rate (e.g., 9600, 19200, 38400, 115200 bps) for proper communication.

- UART uses baud rate (bps) to control data timing instead of a clock signal.

PIN CONFIGURATION



- RC6 (TX) →
  RX (Receive pin of external device)
- RC7 (RX) ←
  TX (Transmit pin of external device)

RC6 / Pin 25 - Transmit (TX) - This pin sends (transmits) serial data from the microcontroller to an external device (like a PC, GPS, or Bluetooth module). The data is sent bit-by-bit (asynchronous serial communication).

RC7 / Pin 26 - Receive (RX) - This pin receives serial data from an external device into the microcontroller. It listens for incoming serial data (one bit at a time)

HARDWARE REQUIREMENT

Pull-up / Pull-down

- Optional pull-ups on RX line if using open-drain configuration.
- Usually, PIC16F877A's TX/RX are push-pull, so external resistors are not required.

Decoupling capacitors

- (0.1 μF) near VDD/VSS for stable operation.

ESD protection diodes

- for long or exposed UART lines.

Short cables

- (<1 m) recommended to reduce noise.

REGISTERS ON PIC16F877A :

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

- BIT 7 - CSRC: CLOCK SOURCE SELECT BIT
- BIT 6 - TX9: 9-BIT TRANSMIT ENABLE BIT
- BIT 5 - TXEN: TRANSMIT ENABLE BIT
- BIT 4 - SYNC: USART MODE SELECT BIT
- BIT 3 - UNIMPLEMENTED: READ AS ‘0’
- BIT 2 - BRGH: HIGH BAUD RATE SELECT BIT
- BIT 1 - TRMT: TRANSMIT SHIFT REGISTER STATUS BIT
- BIT 0 - TX9D: 9TH BIT OF TRANSMIT DATA, CAN BE PARITY BIT

RCSTA: RECEIVE STATUS AND CONTROL REGISTER (ADDRESS 18H)

- BIT 7 - SPEN: SERIAL PORT ENABLE BIT
- BIT 6 - RX9: 9-BIT RECEIVE ENABLE BIT
- BIT 5 - SREN: SINGLE RECEIVE ENABLE BIT
- BIT 4 - CREN: CONTINUOUS RECEIVE ENABLE BIT
- BIT 3 - ADDEN: ADDRESS DETECT ENABLE BIT
- BIT 2 - FERR: FRAMING ERROR BIT
- BIT 1 - OERR: OVERRUN ERROR BIT
- BIT 0 - RX9D: 9TH BIT OF RECEIVED DATA (CAN BE PARITY BIT BUT MUST BE CALCULATED BY USER FIRMWARE)

UART INITIALIZATION STEPS

- Select Mode - Choose Asynchronous or Synchronous operation.
- Set Baud Rate - Define data speed (e.g., 9600 bps) using baud rate generator.
- Configure Frame Format - Set data bits, parity, and stop bits.
- Enable Transmitter (TX) - Activate transmit circuitry for serial output.

- Enable Receiver (RX) - Activate receive circuitry for incoming data.
- Enable UART Module - Turn on the UART hardware block.
- Enable Interrupts (Optional) - Allow automatic TX/RX event handling.
- Check Status Flags - Monitor TX empty, RX full, and error flags.
- Start Communication - Begin serial data transmission and reception.

UART CONFIGURATION CODE STEPS

1.**SET UART PINS**

Configure TX pin as Output and RX pin as Input.

Example: TRISC6 = 0; and TRISC7 = 1;

2.SELECT COMMUNICATION MODE

Choose Asynchronous Mode for standard UART operation.

Example: SYNC = 0;

3.SET BAUD RATE

Calculate and load value into SPBRG register.

Example: SPBRG = 129; for 9600 bps at 20 MHz.

4. CONFIGURE TRANSMIT CONTROL (TXSTA REGISTER)

TXEN = 1 → Enable Transmitter
BRGH = 1 → High-Speed Mode
SYNC = 0 → Asynchronous Mode
Example: TXSTA = 0b00100100;

5. CONFIGURE RECEIVE CONTROL (RCSTA REGISTER)

SPEN = 1 → Enable Serial Port
CREN = 1 → Enable Continuous Reception
Example: RCSTA = 0b10010000;

6. ENABLE UART INTERRUPTS (OPTIONAL)

RCIE = 1 → Enable Receive Interrupt
PEIE = 1, GIE = 1 → Enable Peripheral & Global Interrupts.

7. TRANSMIT DATA FUNCTION

Wait until TX buffer is empty, then load data.

```c
while(!TXIF);
TXREG = data;
```

8. RECEIVE DATA FUNCTION

Wait until data is available, then read from receive register.

Example:

```c
while(!RCIF);
data = RCREG;
```

9. START COMMUNICATION

UART is configured and ready to send and receive serial data.

DEBUGGING AND TESTING

- Check Power Supply: Ensure correct voltage and stable power to all components.
- Verify Connections: Inspect wiring or circuit connections for loose or incorrect links.
- Confirm Pin Configuration: Make sure all I/O pins are properly assigned and not conflicting.
- Review Code/Logic: Check initialization, loops, and conditions for logical or syntax errors.
- Test Individual Modules: Verify each section (sensors, communication, display, etc.) separately before full integration.

- Monitor Serial Output:Use UART or serial monitor to print debug messages for program flow tracking.
- Observe Indicators: Check LEDs or signals for expected behavior during operation.
- Simulate Before Hardware Test: Run the circuit in simulation to detect logical or timing errors early.
- Check Baud Rate and Communication Settings: Ensure sender and receiver settings match for UART or other serial protocols.
- Validate Final Output: Compare actual output with expected results to confirm system correctness.

THANK YOU