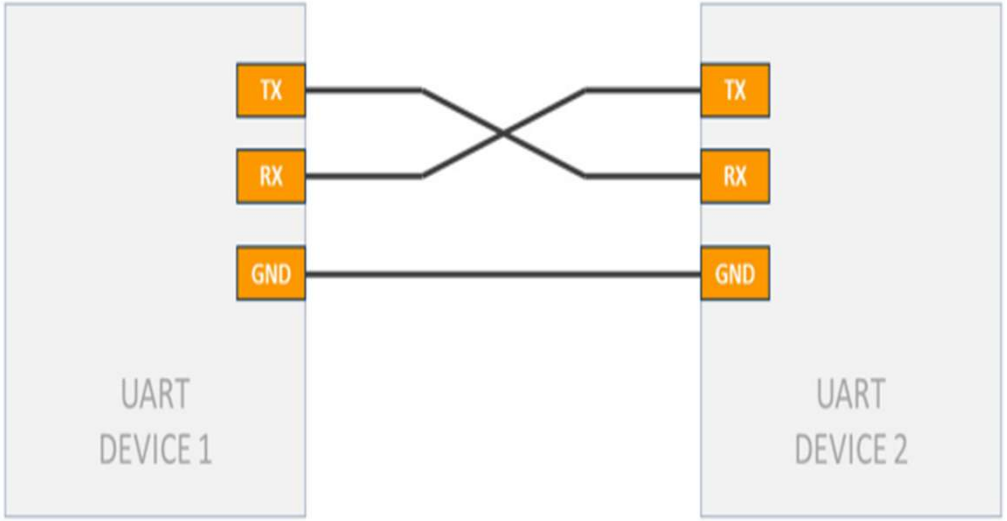
Universal Asynchronous Receive and Transmit(UART)

- **1. What is UART?**
- UART = **Universal Asynchronous Receiver/Transmitter**
  It is a hardware block inside microcontrollers used for **serial communication** without a clock line.
- Key characteristics:
- **Asynchronous** (no clock)
- Uses **start bit**, **data bits**, **optional parity**, **stop bits**
- Works on **TX** (transmit) and **RX** (receive) lines
- **Full-duplex** (send & receive at same time)



USART vs UART

✓ UART = Asynchronous communication

No clock line, uses start/stop bits. Simple 2-wire TX/RX communication.

✓ USART = UART + Synchronous mode

Supports both asynchronous (UART) and synchronous communication (uses an extra clock line).

UART Frame Format

Why?

Start bit = tells receiver that transmission begins

Stop bit = allows resynchronization

Parity = simple error check

Baud Rate

Baud rate = speed of communication (bits/sec)

Examples:

9600

115200 (most common for debugging)

Both devices must use the same baud rate.

Start Bit	Data Frame	Parity Bits	Stop Bits
1 bit	5 to 9 bits	1bit	1 or 2 bits

Figure 10.2: The UART data packet

Bit Rate vs Baud Rate

Bit Rate = number of bits transmitted per second (bps).

Baud Rate = number of signal changes (symbols) per second.

If 1 symbol = 1 bit → baud rate = bit rate, but if 1 symbol carries multiple bits, then bit rate > baud rate.

UART Registers (The Core of Hardware Programming)

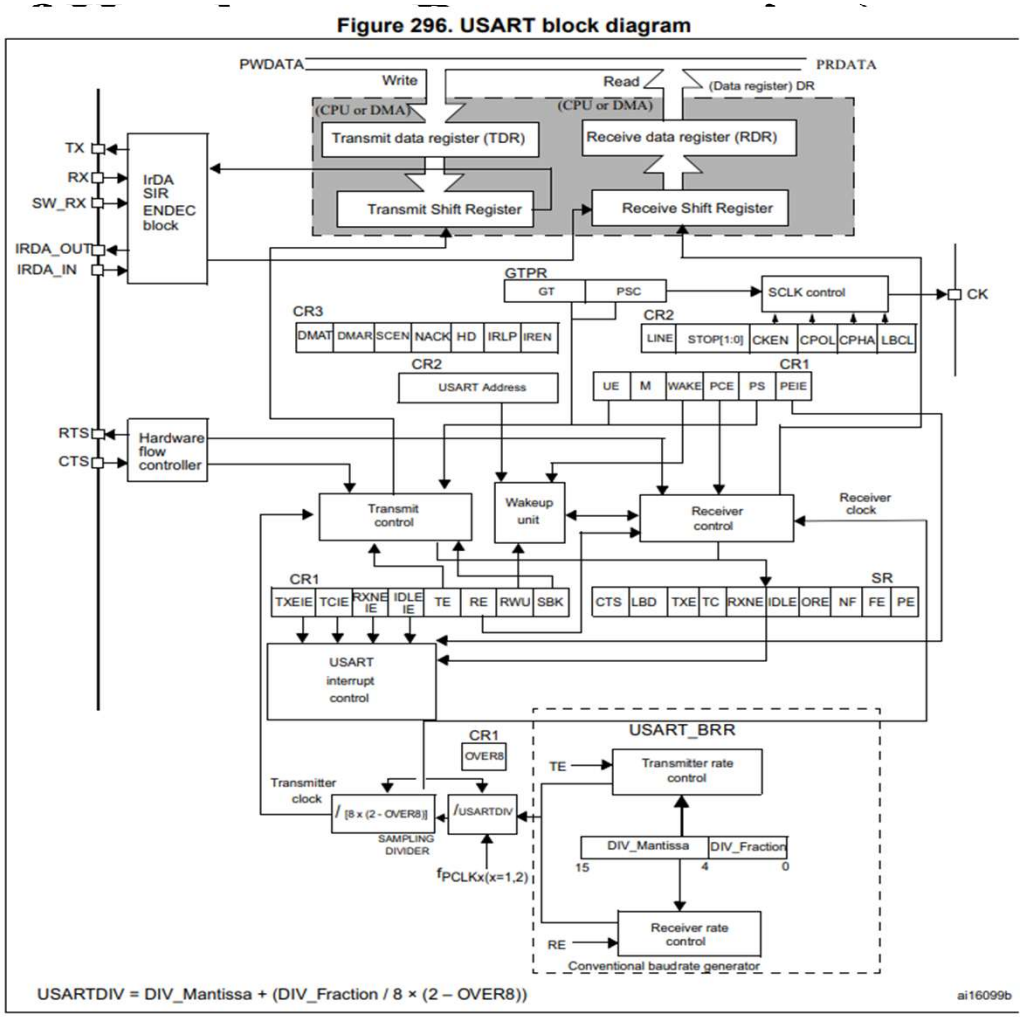
- When you use UART in STM32 (or any MCU), you control it by writing to specific **registers**.
- For STM32, UART is part of the **USART peripheral**
  (USART = synchronous + asynchronous; UART = only asynchronous).
- There are **4 main registers** to understand:
- 🔶 **1. USART_SR — Status Register**
- Contains status flags.
- Important bits:
- **TXE (bit 7)** → Transmit Data Register Empty
  1 = ready to send next byte
- **RXNE (bit 5)** → Receive Data Ready
  1 = new data available
- **TC (bit 6)** → Transmission complete
- **ORE (bit 3)** → Overrun error

Reset value: 0x000C0 00C0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CTS	LBD	TXE	TC	RXNE	IDLE	ORE	NF	FE	PE
						rc_w0	rc_w0	r	rc_w0	rc_w0	r	r	r	r	r

UART Registers (The Core

- **2. USART_DR — Data Register**
- Writing to DR → sends data out TX pin
- Reading DR → gets received data
- Only lower 8 bits used for 8-bit data.
- 🔶 **3. USART_BRR — Baud Rate Register**
- Sets the baud rate.
- Formula:
- USARTDIV = Fclk / (16 × BaudRate)
- You compute:
- Mantissa (integer part)
- Fraction (decimal part)
- and write to BRR.



UART Registers (The Core of Hardware Programming)

- **4 .USART_CR1 — Control Register 1**
- Controls most UART functions.
- Important bits:
- **UE (bit 13)** → USART Enable
- **TE (bit 3)** → Transmitter Enable
- **RE (bit 2)** → Receiver Enable
- **M (bit 12)** → Word length (8/9 bit)
- **PCE (bit 10)** → Parity enable
- **PS (bit 9)** → Parity type (even/odd)
- Minimum to enable UART TX:
- UE = 1 (USART ON)
- TE = 1 (Transmitter ON)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OVER8	Reserved	UE	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	RWU	SBK
rw	Res.	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Simulated Registers

USART_SR = Status register
USART_DR = Data register
Used to mimic real UART hardware behavior.

✓ Key Flags (Bits)

TXE (bit 7): Transmit buffer empty → ready to send

RXNE (bit 5): Data received → ready to read

TC (bit 6): Transmission complete

CTS, **LBD**, **IDLE**: Special hardware events

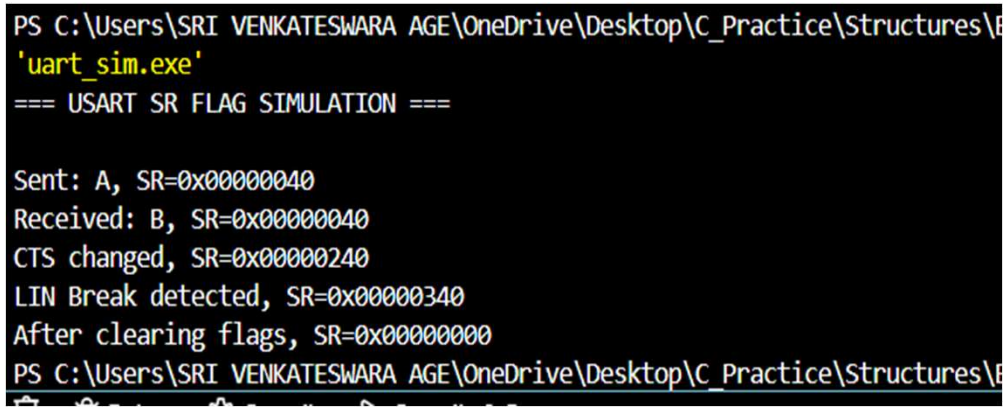
✓ Driver Logic Practiced

uart_write() waits for TXE → sends data → clears TXE → sets TC

uart_read() waits for RXNE → reads data → clears RXNE

✓ Simulation Functions

simulate_tx_ready(), simulate_rx(), simulate_cts_change(), simulate_lin_break()
→ emulate hardware setting flags.



UART vs SPI vs I²C

Feature	UART	SPI	I²C
Type	Asynchronous serial	Synchronous serial	**Synchronous serial**
Wires Required	2 wires (TX, RX)	4+ wires (MOSI, MISO, SCK, CS)	**2 wires (SDA, SCL)**
Clock Line	❌ No clock	✔ Yes	✔ **Yes**
Speed	Low–Medium (up to ~1 Mbps)	**Very High** (10+ Mbps)	**Medium (100k–3.4M)**
Number of Devices	Only 2 (point-to-point)	1 master + many slaves	**Many masters + many slaves**
Addressing	❌ No	❌ No	✔ **Yes (7/10-bit address)**
Duplex Mode	**Full-Duplex**	**Full-Duplex**	**Half-Duplex**
Protocol Complexity	Very Simple	Simple	**Complex**
Data Framing	Start/Stop/Parity bits	No framing	**Start/Stop, ACK/NACK**
Reliability	Good for short distances	High	**Good but noise sensitive**
Use Cases	Debugging, GPS, Bluetooth	SD cards, Displays, ADCs	**Sensors, EEPROMs, IMUs**
Hardware Overhead	Very Low	Medium	**Medium**
Pull-up Resistors	Not required	Not required	**Required on SDA/SCL**
Pin Efficiency	Moderate	**Worst** (many pins)	**Best (only 2 pins for many devices)**