Understanding the UART Protocol from the STM32 Serial Communication Protocols:

UART (Universal Asynchronous Receiver/Transmitter) is a serial communication protocol used for one-to-one peer-to-peer communication. It involves a minimal 3-wire connection setup consisting of a transmit (TX) line, a receive (RX) line, and a ground (GND).

Key Features:

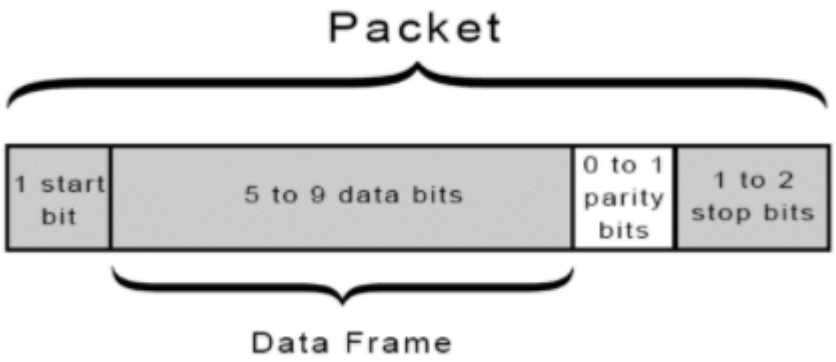
- **Word Length:** Configurable from 5 to 8 bits.
- **Parity:** Optional parity bit for error checking.
- **Start Bit:** Indicates the beginning of data transmission.
- **Stop Bit:** Indicates the end of data transmission.
- **Baud Rate:** Programmable baud rate for data transfer speed.

Example Configurations:

- **9600 baud rate, 8 data bits, No parity, 1 stop bit (9600 8N1):** Total 10 bits per frame.
- **115200 baud rate, 5 data bits, Even parity, 2 stop bits (115200 5E2):** Total 9 bits per frame.

UART Communication Protocol Format:

The format of UART communication is structured with:



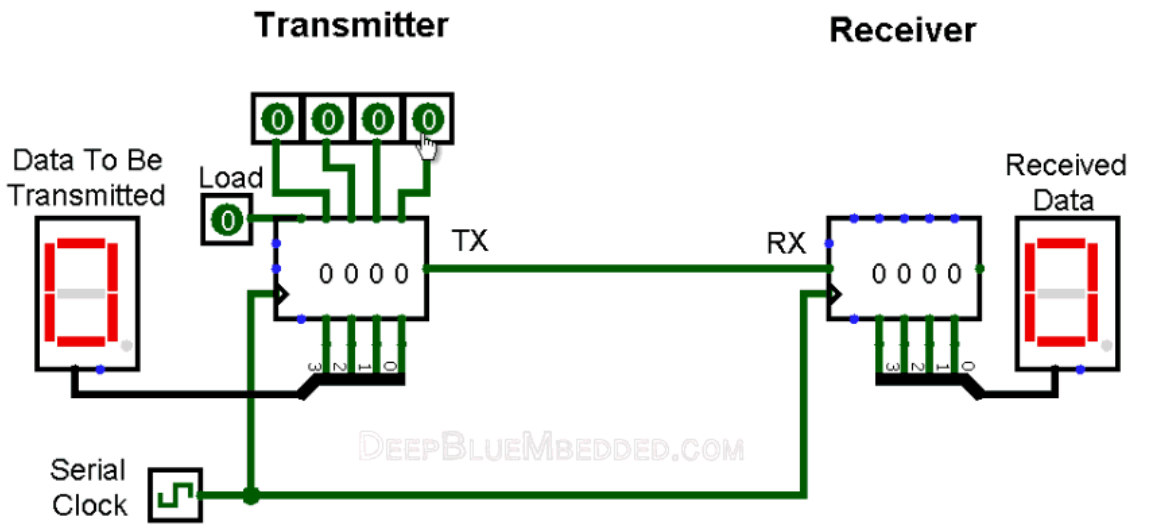
- Start Bit (1 bit)
- Data Bits (5-9 bits)
- Parity Bit (1 bit, optional)
- Stop Bit (1-2bits)

Example:

- For a configuration of 9600 baud rate, 8 data bits, no parity, and 1 stop bit (9600 8N1), the frame comprises 1 start bit, 8 data bits, and 1 stop bit, totaling 10 bits.

STM32 UART Features:

- **Full-Duplex Communication:** Allows simultaneous transmission and reception of data.
- **Configurable Oversampling:** Can be set to 16 or 8 times, providing flexibility between speed and clock tolerance.
- **Fractional Baud Rate Generator:** Allows fine-tuning of the baud rate for precise communication.
- **Programmable Data Word Length:** Supports data word lengths of 8 or 9 bits.
- **Configurable Stop Bits:** Can be set to 1 or 2 stop bits.
- **Separate Enable Bits for Transmitter and Receiver:** Allows independent enabling/disabling of the transmitter and receiver.



Initialization and Configuration:

To use UART on an STM32 microcontroller, you need to:

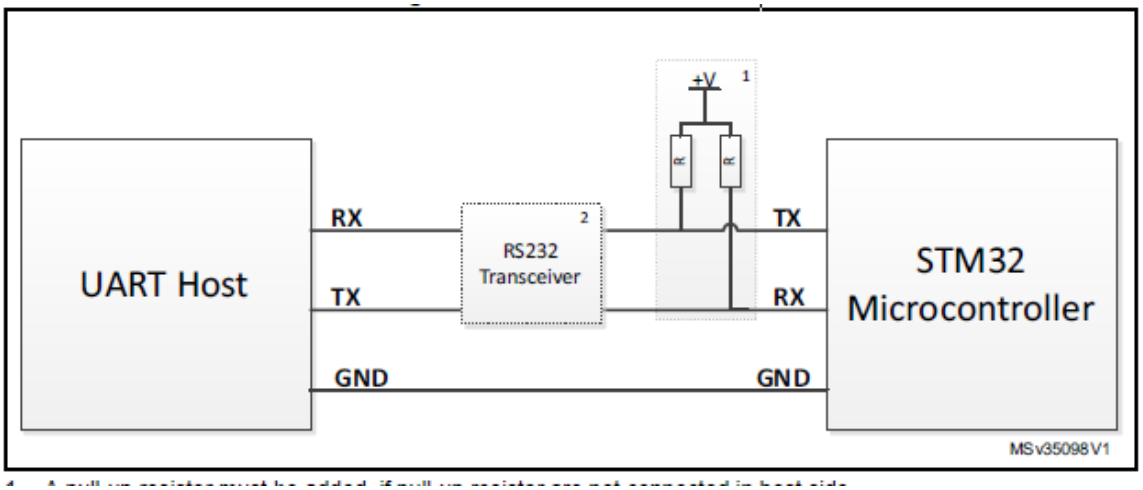
- **Enable the UART Peripheral:** Turn on the UART module.
- **Set the RX/TX Pins:** Configure the microcontroller pins for UART functionality.
- **Configure Baud Rate and Data Format:** Set the desired baud rate, data length, parity, and stop bits.
- **Configure Additional Features:** Enable interrupts and FIFO if necessary.

Sending and Receiving Data:

Data Transmission and Reception: Uses a single register for both transmit and receive operations, allowing for efficient data handling.

Blocking/Non-Blocking Functions: Provides functions for both blocking and non-blocking data transfers.

Interfacing UART with STM32F401RBT6 Microcontroller:



UART Interface on STM32F401RBT6:

The STM32F401RBT6 microcontroller comes equipped with multiple UART interfaces, making it highly versatile for serial communication. Here's how you can interface and configure UART on this microcontroller:

Steps to Interface UART:

Enable UART Peripheral Clock:

- Use the RCC (Reset and Clock Control) registers to enable the clock for the UART peripheral.

Configure GPIO Pins:

- Set up the GPIO pins for UART TX and RX functions.

Initialize UART:

- Configure the UART parameters such as baud rate, word length, stop bits, and parity.

Enable UART Interrupts (Optional):

- If using UART interrupts, configure and enable them.

The UART protocol provides a simple and efficient method for serial communication in embedded systems. With multiple UART interfaces, the STM32F401RBT6 microcontroller supports diverse applications ranging from consumer electronics to industrial automation.

Interrupts:

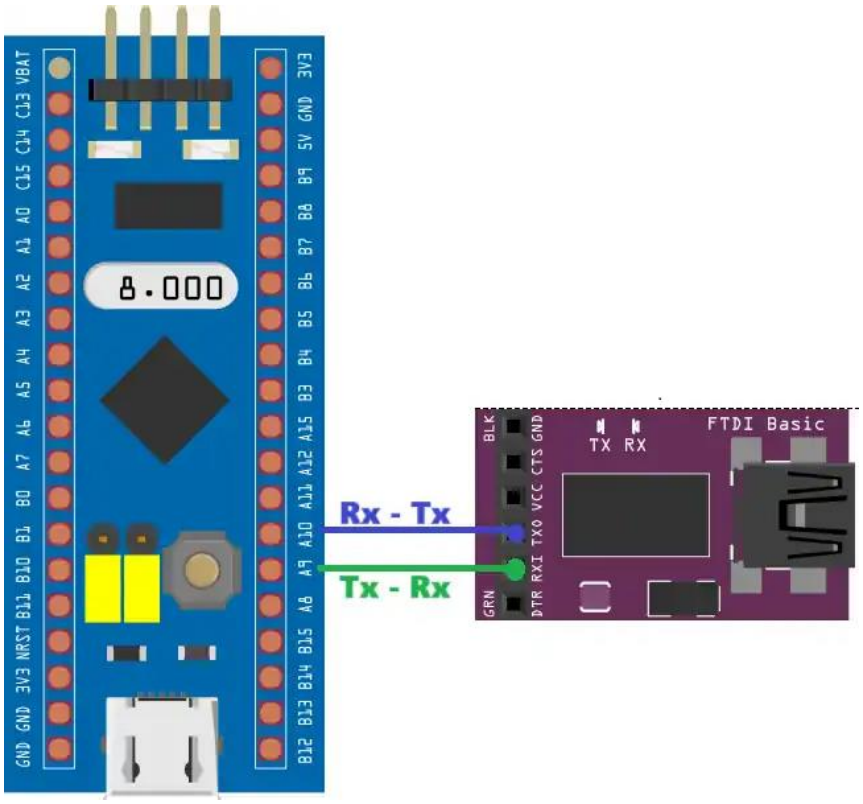
- **Overrun Error:** Indicates data loss due to buffer overflow.
- **Break Error:** Indicates a break condition on the line.
- **Parity Error:** Indicates a parity mismatch.
- **Framing Error:** Indicates incorrect frame structure.
- **Receive Timeout:** Indicates a timeout when no data is received over a period.
- **Transmit Interrupt:** Triggered when the transmit buffer is empty.
- **Receive Interrupt:** Triggered when data is received.

UART Interrupt Handling:

Single Interrupt per Module: Each UART module generates a single interrupt, which must be handled appropriately to determine the source and clear the interrupt flag.

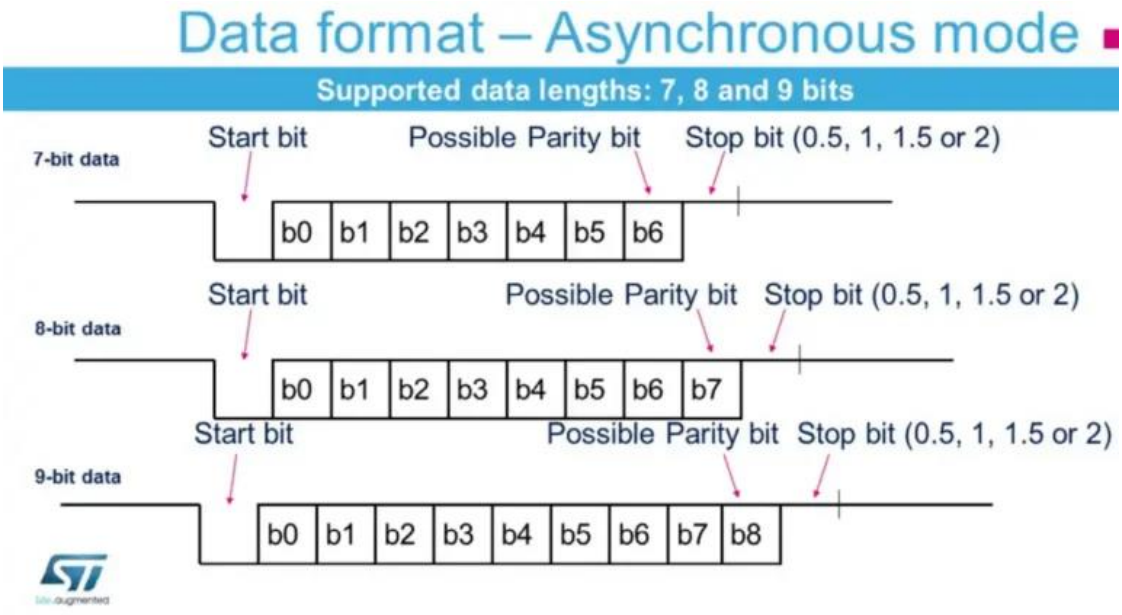
Conditions for Interrupts: Various conditions can trigger interrupts, such as data errors, buffer states, and timeouts.

Below is the image showing the connection between the MCU and the FT232 USB to UART converter.



The UART is always connected in the cross connection, connecting the TX pin of the MCU to the RX of the device and the RX to the TX of the device. The module then connects to the computer using the USB.

- We will use the Asynchronous Mode for the communication. Only 2 pins are used in the Asynchronous mode, TX and RX.
- The baud rate is set to 115200. We need to use the same baud rate for the receiver device also.
- Word length consists of the data bits and the parity bit. STM32 supports different word lengths of 7 bits, 8 bits and 9 bits. A typical UART frame is shown in the image below.



- The frame consists of a start bit, data bits and stop bits. we are using 8 data bits with no parity bit and 1 stop bit. We need to use the same configuration in the receiver device also. The data direction is set to Receive and Transmit.
- The oversampling is used to increase the tolerance of the receiver to the clock deviation. But it also reduces the maximum baud rate the device can achieve. The image below explains the oversampling and max baud rate.

Number of UARTs in STM32F401RBT6:

- **4 UART interfaces**: USART1, USART2, USART6, and UART4. This provides flexibility in connecting multiple serial devices.

UART Applications:

Communication with Sensors:

- UART is commonly used to interface with sensors that require serial communication, such as GPS modules, temperature sensors, and accelerometers.

Microcontroller to Microcontroller Communication:

- UART allows easy communication between two microcontrollers, which can be useful in complex systems where multiple microcontrollers share tasks.

Debugging:

- UART is often used for debugging purposes, allowing developers to send debug information to a terminal or PC.

Wireless Communication:

- UART is used in conjunction with wireless modules like Bluetooth and Wi-Fi to facilitate wireless communication.

Peripheral Interface:

- UART interfaces with various peripherals like RFID readers, GSM modules, and other serial devices.

Use Cases for UART Protocol:

Industrial Automation:

- Used for communication between controllers and sensors/actuators in an industrial environment.

Consumer Electronics:

- Interfaces between microcontrollers and peripherals in devices like home appliances, smartphones, and gaming consoles.

Medical Devices:

- Used in medical equipment for data transmission between microcontrollers and sensor modules.

Embedded Systems Development:

- Essential in development boards and kits for serial communication between the microcontroller and PC for programming and debugging.

The UART protocol remains a cornerstone in serial communication, providing a straightforward, reliable, and efficient method for data exchange between microcontrollers and peripherals. The STM32F401RBT6 microcontroller, with its multiple UART interfaces, stands out as a versatile choice for a wide range of applications, from industrial automation to consumer electronics.