Experiment 10 UART Protocol

The objective of this experiment is to establish a **serial communication between STM32 Microcontroller board and Arduino Uno**. The communication established here is using **UART** (Universal Asynchronous Receiver Transmitter) protocol, which is an internal module within all available microcontrollers. It's a serial communication with full-duplex mode, by this data can be shared between two microcontrollers. In this experiment we considered the STM32F303RE board as transmitter and Arduino Uno as receiver. The received data can be observed on the serial monitor of the Arduino IDE and also we are controlling the LED of an Arduino via UART.

Working Principle:

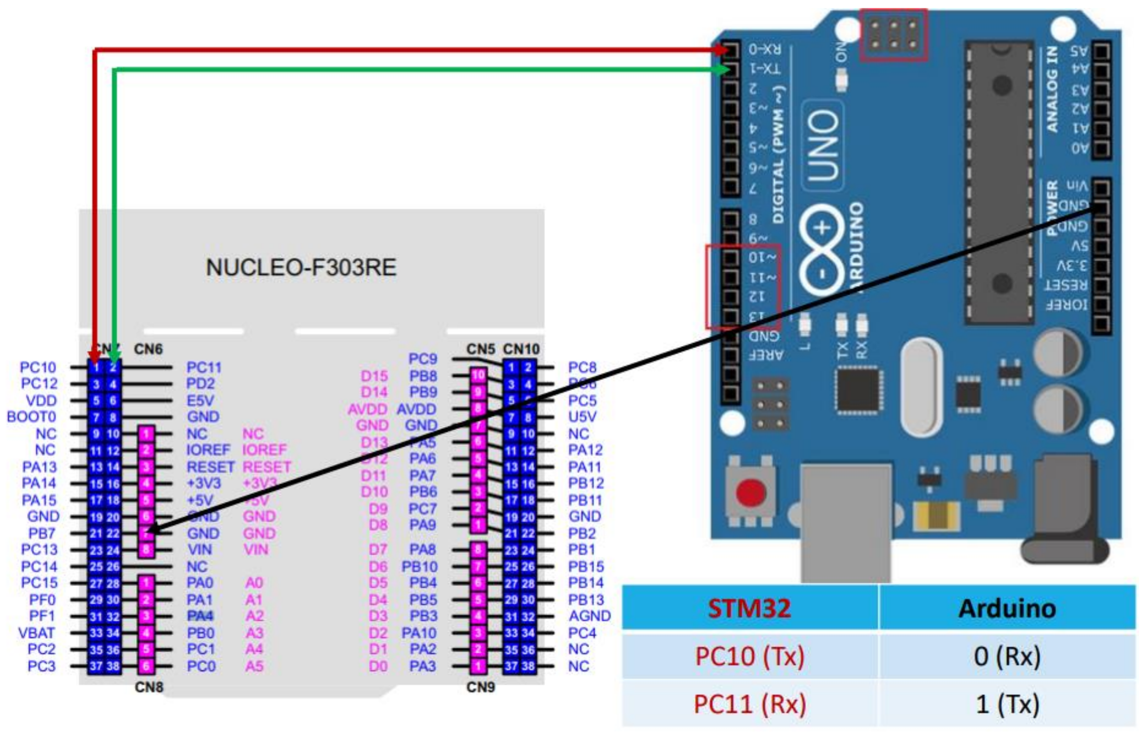
UART works by converting data into packets for sending or rebuilding data from packets received. Before the UART device can send data, the transmitting device converts the data bytes to bits. After converting the data into bits, the UART device then splits them into packets for transmission. Each packet contains a **start bit, a data frame, parity bit, and the stop bits**. The receiving UART device checks the received packet (via RX pin) for errors by calculating the number of 1's and comparing it with the value of the parity bit contained in the packet. If there are no errors in transmission, it will then proceed to strip the start bit, stop bits, and parity bit to get the data frame. It may need to receive several packets before it can rebuild the whole data byte from the data frames. After rebuilding the byte, it is stored in the UART buffer.

Components and Software applications required:

Components: Arduino Uno board, STM32F303RE board, LED, USB Cables and connecting wires.

Software applications: STM32CubeIDE and Arduino IDE.

Circuit Connection Diagram:

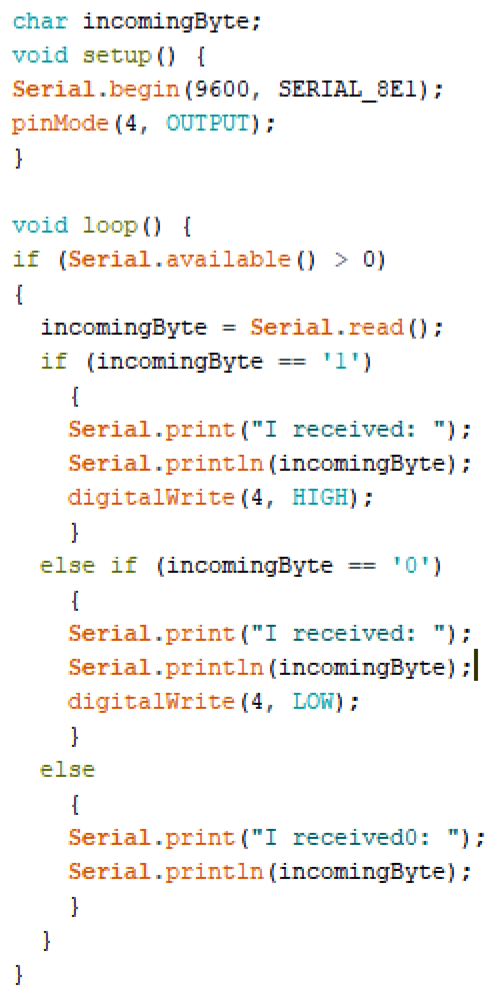


STM32	Arduino
PC10 (Tx)	0 (Rx)
PC11 (Rx)	1 (Tx)

Also connect a LED between digital Pin No. 4 and GND

Section: 01 Steps to program the Arduino Uno board

1. Open Arduino IDE and create the new file.
2. Select the serial port by clicking **Tools** → **Port** → **COMx (Arduino Uno)**
3. Write the program as pasted below.
4. Save the file and compile (with no errors) then upload.



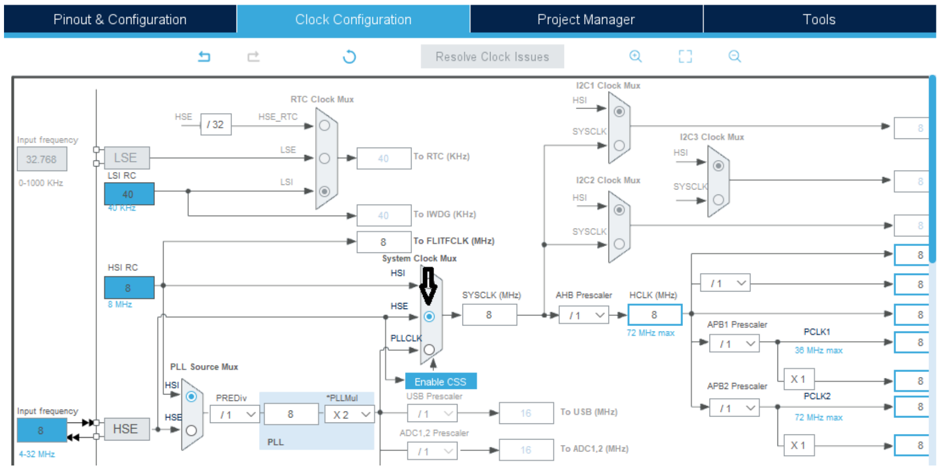
```
char incomingByte;
void setup() {
Serial.begin(9600, SERIAL_8E1);
pinMode(4, OUTPUT);
}

void loop() {
if (Serial.available() > 0)
{
  incomingByte = Serial.read();
  if (incomingByte == '1')
    {
    Serial.print("I received: ");
    Serial.println(incomingByte);
    digitalWrite(4, HIGH);
    }
  else if (incomingByte == '0')
    {
    Serial.print("I received: ");
    Serial.println(incomingByte);
    digitalWrite(4, LOW);
    }
  else
    {
    Serial.print("I received0: ");
    Serial.println(incomingByte);
    }
  }
}
```

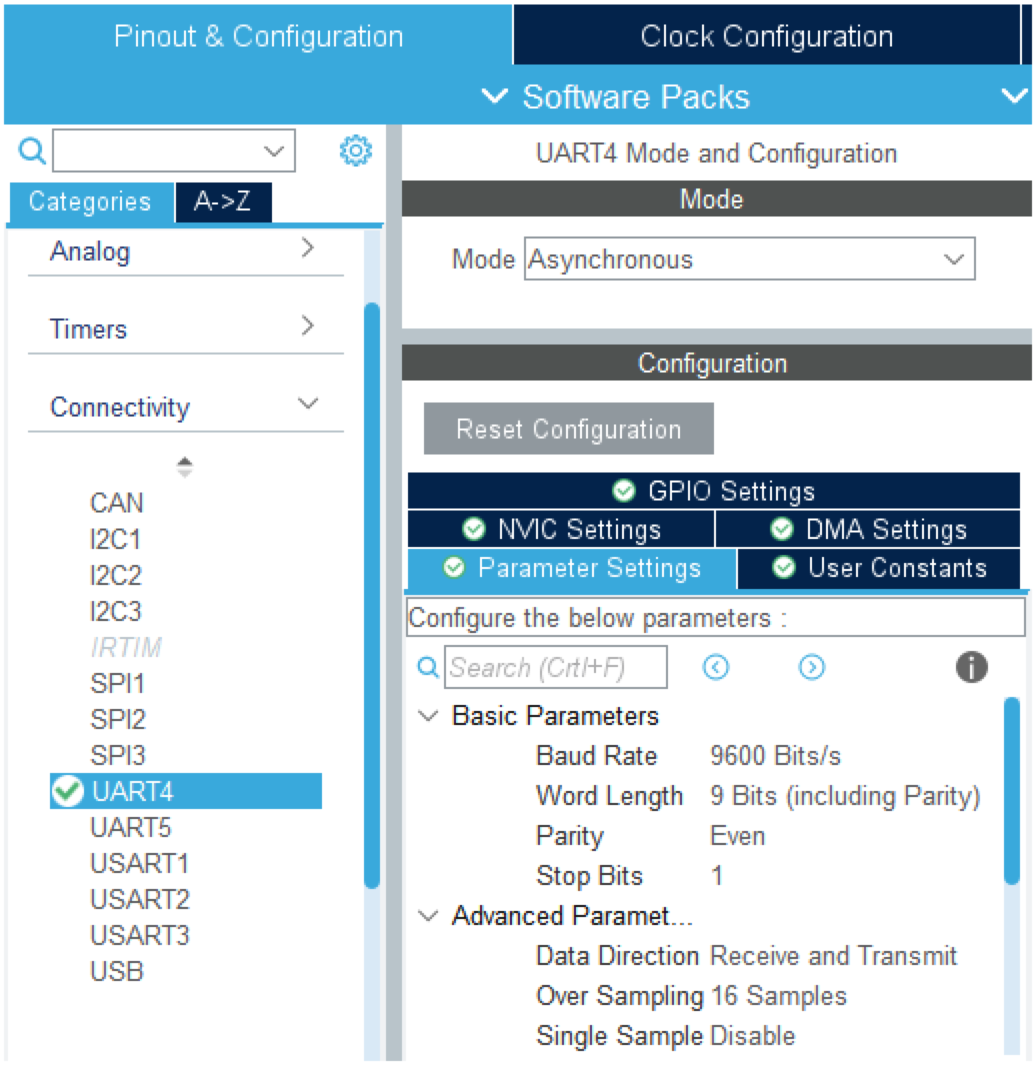
Section: 02 Steps to program the STM32F303RE board

1. Open the STM32CubeIDE and create the new STM32 project.
2. Select the appropriate STM32 board by entering the part number and name the project.
3. In the **Pinout and Configuration** window,
   - Select **RCC** as High Speed Clock (HSE): Crystal/Ceramic Resonator.
4. In the **Clock Configuration** window,

- Choose **HSE** from the multiplexer to provide the system clock (SYSCLK) as shown.



5. Select **UART4** from pinout and configuration pane and do the following settings as depicted in below image.



UART4 Mode and Configuration

Mode

Mode Asynchronous

Configuration

Configure the below parameters :

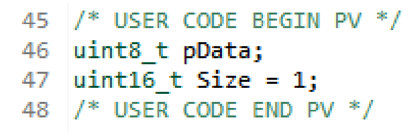
Basic Parameters

Baud Rate	9600 Bits/s
Word Length	9 Bits (including Parity)
Parity	Even
Stop Bits	1

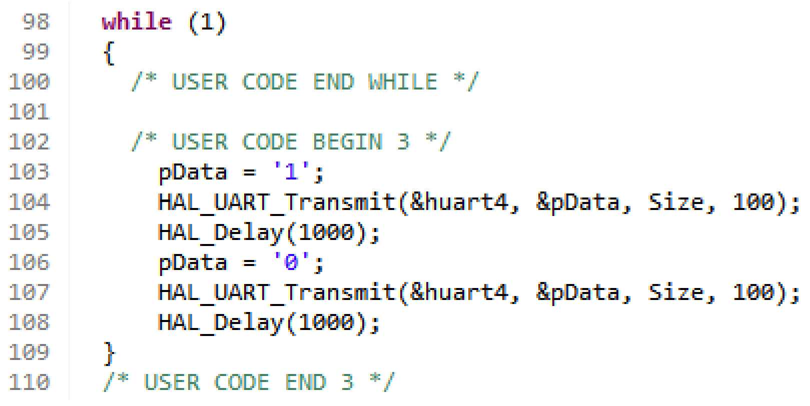
Advanced Paramet...

Data Direction	Receive and Transmit
Over Sampling	16 Samples
Single Sample	Disable

6. After doing the above mentioned settings, generate the code and add the followings in the main.c file. (Under PV and BEGIN 3)



```
/* USER CODE BEGIN PV */
uint8_t pData;
uint16_t Size = 1;
/* USER CODE END PV */
```



```
  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
      pData = '1';
      HAL_UART_Transmit(&huart4, &pData, Size, 100);
      HAL_Delay(1000);
      pData = '0';
      HAL_UART_Transmit(&huart4, &pData, Size, 100);
      HAL_Delay(1000);
  }
  /* USER CODE END 3 */
```

7. Save and debug the program then click on Run.
8. Open the Serial Montor window in the Arduino IDE and observe the received data.
9. Establish the UART communication for different **standard baud rates** (300, 1200, 2400, 4800, 19200, 38400, 57600, 74880, 115200, 230400, 250000, 500000, 1000000, 2000000) by changing the baud rate in Section-01 and Section-02 and write your observation.

NOTE:

- Baud rate, word length, parity, and stop bit must be same for transmitter and receiver microcontrollers.
- Common ground for both devices transmitter and receiver.