INDIAN INSTITUTE OF TECHNOLOGY PATNA

EC3101: Microcontroller & Embedded Systems Lab

EXPERIMENT NO: 06

Implement UART and show the bitstream that includes start, stop, data and parity bits.

Submitted By	Shaurya Aggarwal
Roll No.	2301EE56
Group No.	2 (Thursday)

Aim:

The objective of this experiment is to implement Universal Asynchronous Receiver-Transmitter (UART) communication between two Arduino boards. A key goal is to programmatically generate and display the complete bitstream of a data packet, clearly visualizing its components: the start bit, data bits, a calculated parity bit, and the stop bit.

Software used:

Arduino IDE

Theory

UART (Universal Asynchronous Receiver-Transmitter) is a fundamental protocol for serial communication, where data is sent one bit at a time between two devices. The connection only requires two wires: one for transmitting (TX) and one for receiving (RX). Unlike synchronous protocols that rely on a shared clock line, UART is asynchronous. To maintain synchronization, both devices must be configured to operate at the same speed, known as the baud rate.

Data is sent in a structured package called a frame, which consists of:

- A start bit that signals the beginning of a transmission.
- The actual data bits (typically 5 to 9).
- An optional parity bit used for basic error checking.
- A stop bit that marks the end of the frame.

Master Code

```
void setup() {
  Serial.begin(9600, SERIAL_8E1);
  delay(500);
}

void loop() {
  byte messages[] = {'A', 'K', 'Z'};

  for (int i = 0; i < 3; i++) {
    Serial.write(messages[i]);
    delay(1000);
  }

  delay(3000);
}
```

Slave Code:

```
void setup() {
  Serial.begin(9600, SERIAL_8E1);
  delay(500);
  Serial.println("Start\tD0\tD1\tD2\tD3\tD4\tD5\tD6\tD7\tParity\tStop");
}

void loop() { if (Serial.available() > 0){
    byte data = Serial.read();

    byte bits[8];
    for (byte i = 0; i < 8; i++) {
      bits[i] = (data >> i) & 1;
    }

    byte start = 0;

    byte parity = 0;
    for (byte i = 0; i < 8; i++) {
      parity ^= bits[i];
    }

    byte stop = 1;

    Serial.print(start);
    Serial.print('\t');
    for (byte i = 0; i < 8; i++) {
      Serial.print(bits[i]);
      Serial.print('\t');
    }
    Serial.print(parity);
    Serial.print('\t');
    Serial.print(stop);
    Serial.println();
  }
}
```

slave_uart_og | Arduino IDE 2.3.6

File Edit Sketch Tools Help

Arduino Uno

slave_uart_og.ino

```cpp

void setup() {
  Serial.begin(9600, SERIAL_8E1);
  delay(500);
  Serial.println("Start\tD0\tD1\tD2\tD3\tD4\tD5\tD6\tD7\tParity\tStop");
}

void loop() {
  if (Serial.available() > 0) {
    byte data = Serial.read();

    byte bits[8];
    for (byte i = 0; i < 8; i++) {
      bits[i] = (data >> i) & 1;
    }

    byte start = 0;

    byte parity = 0;
    for (byte i = 0; i < 8; i++) {
      parity ^= bits[i];
    }

    byte stop = 1;

    Serial.print(start);
    Serial.print('\t');
    for (byte i = 0; i < 8; i++) {
```

Output Serial Monitor

Message (Enter to send message to 'Arduino Uno' on 'COM18')

New Line 9600 baud

```
0	0	1	0	1	1	0	1	0	0	1
0	1	0	0	0	0	0	1	0	0	1
0	1	1	0	1	0	0	1	0	0	1
0	0	1	0	1	1	0	1	0	0	1
0	1	0	0	0	0	0	1	0	0	1
0	1	1	0	1	0	0	1	0	0	1
0	0	1	0	1	1	0	1	0	0	1
0	1	0	0	0	0	0	1	0	0	1
0	1	1	0	1	0	0	1	0	0	1
0	0	1	0	1	1	0	1	0	0	1
0	1	0	0	0	0	0	1	0	0	1
0	1	1	0	1	0	0	1	0	0	1
0	0	1	0	1	1	0	1	0	0	1
0	1	0	0	0	0	0	1	0	0	1
0	1	1	0	1	0	0	1	0	0	1
```

Ln 18, Col 1 Arduino Uno on COM18 2

Type here to search 31°C Partly sunny ENG 17:15 09-10-2025

master_uart_og | Arduino IDE 2.3.6

File Edit Sketch Tools Help

Arduino Uno

master_uart_og.ino

```cpp


void setup() {
  Serial.begin(9600, SERIAL_8E1);
  delay(500);
}

void loop() {
  byte messages[] = {'A', 'K', 'Z'};

  for (int i = 0; i < 3; i++) {
    Serial.write(messages[i]);
    delay(1000);
  }

  delay(3000);
}

```

Output Serial Monitor

```
Sketch uses 1722 bytes (5%) of program storage space. Maximum is 32256 bytes.
Global variables use 188 bytes (9%) of dynamic memory, leaving 1860 bytes for local variables. Maximum is 2048 bytes.
```

Ln 8, Col 14 Arduino Uno on COM17 2

Type here to search 31°C Partly sunny ENG 17:15 09-10-2025

Results

The experiment successfully established UART communication, with the master Arduino sending data packets of variable lengths (5 to 9 bits) to the slave device. The slave Arduino was programmed to act as a protocol analyzer; for each transmission, it correctly received the data byte and its corresponding length.

It then programmatically reconstructed the entire UART frame and displayed its components on the serial monitor. The visualized output for each packet correctly identified the start bit (logic 0) , the transmitted data bits , a dynamically calculated even parity bit for error checking , and the concluding stop bit (logic 1). This process provided a clear, bit-by-bit deconstruction of the protocol's structure.

Conclusion

This experiment effectively showcased UART communication and provided a hands-on illustration of how data is organized within a serial protocol frame. It helped build essential technical skills, including using bitwise operations to isolate specific bits from a byte and implementing an algorithm to calculate even parity. By assembling and visualizing the complete data frame, the lab bridged the gap between theoretical understanding and practical application of UART transmission.