Experiment - 7 **UART** **Monsoon 2018**

Communication between two Arduino boards using usart protocol via software serial pins.

UART:

- UART stands for Universal Asynchronous Receiver Transmitter.
- It is a microcontroller peripheral that converts incoming and outgoing bytes of data into a serial bit stream.
- All Arduino boards have at least one serial port (also known as a UART): Serial. It communicates on digital pins 0 (RX) and 1 (TX) as well as with the computer via USB. Thus, if you use these functions, you cannot also use pins 0 and 1 for digital input or output.

In UART communication, two UARTs communicate directly with each other. Only two wires are needed to transmit data between two UARTs(Fig. 1). Data flows from the Tx pin of the transmitting UART to the Rx pin of the receiving UART:

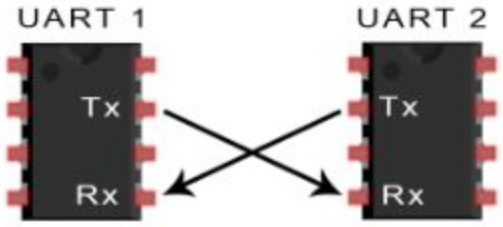

Fig. 1

UARTs transmit data asynchronously, which means there is no clock signal to synchronize the output of bits from the transmitting UART to the sampling of bits by the receiving UART. Instead of a clock signal, the transmitting UART adds start and stop bits to the data packet being transferred. These bits define the beginning and end of the data packet so the receiving UART knows when to start reading the bits. When the receiving UART detects a start bit, it starts to read the incoming bits at a specific frequency known as the baud rate. Baud rate is a measure of the speed of data transfer, expressed in bits per second (bps). Both UARTs must operate at about the same baud rate. The baud rate between the transmitting and receiving UARTs can only differ by about 10% before the timing of bits gets too far off. Both UARTs must also be configured to transmit and receive the same data packet structure.

UART working:

The UART that is going to transmit data receives the data from the data bus. The data bus is used to send data to UART by another device like a CPU, memory, or microcontroller. Data is transferred from the data bus to the transmitting UART in parallel form. After the transmitting UART gets the parallel data from the data bus, it adds a start bit, parity bit, and a stop bit, creating the data packet. Next, the data packet is output Serially, bit by bit at the Tx pin. The receiving UART then converts the data back into parallel form and removes the start bit, parity bit, and stop

bits(Fig. 2). Finally the receiving UART transfers data packer in parallel to data bus on the receiving end:

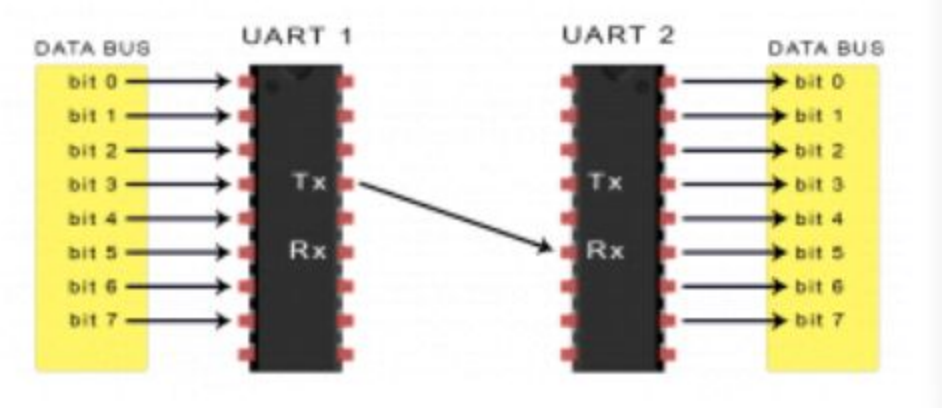

Fig. 2

- Start bit initiates the serial bit stream and a stop bit (or two) completes the data word.
- A UART also has the option of adding a parity bit to the stream to assist in detecting if a bit error occurs during transmission.
- Following figure(Fig. 3) shows a standard example of data transmitted through a UART.

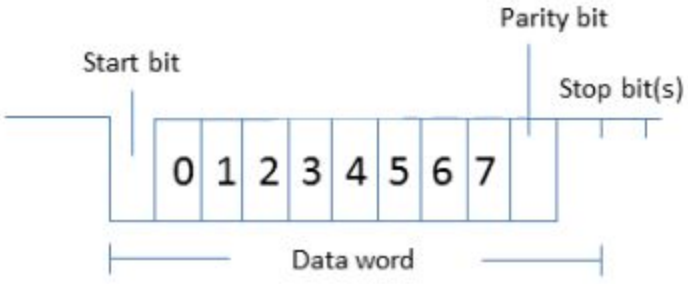

Fig. 3

Problem Statement:

Program 1:

Take input from 'Serial Monitor' of first arduino and send the number to second arduino via Software serial pins and display it on 'Serial Monitor' of the second arduino. Send an acknowledgement from Second arduino (send a signal High) to first arduino on receiving a number and print "acknowledgement received" on serial monitor of first arduino when you actually receive acknowledgement from second. Pseudo code for First Arduino and Second Arduino is given below.

First Arduino :

```
void loop() {

  read_form_serial_monitor();
  send_info_to_secondAr_via_softwareserial();
  wait_for_ack();
  if(ack){
    //print ack_recevied on serial_monitor;
  }
}
```

Second Arduino:

```
void loop() {
  read_from_Software_serial();
  //print info_recevied on serial_monito
  Send_ack_for_received_info();

}
```

Program 2:

Take two inputs multiplier and multiplicand from 'Serial Monitor' of first arduino and send to Second Arduino (via software serial pins) one after the other on receiving acknowledgement for previous sent data. Display multiplication result on Serial monitor of Second arduino and halt the communication for 5 seconds. Restart the communication and this process should go on indefinitely.