Experiment 8

Title: - Study of serial Communication between PC and Arduino

Objective

Write a program in Embedded C to transfer the message and receive serially at 9600 baud, 8 bit data, 1 stop bit.

THEORY

1. Serial Communication:

The microcontroller is parallel device that transfers eight bits of data simultaneously over eight data lines to parallel I/O devices. However, in many situations, parallel data transfer is impractical. For example, parallel data transfer over a long is very expensive. Hence, a serial communication is widely used in long distance communication.

In serial data communication, 8-bit data is converted to serial bits using a parallel in serial out shift register and then it is transmitted over a single data line. The data byte is always transmitted with least significant bit first.

Serial ports are a type of computer interface for serial communication that complies with the RS-232 standard. They are 9-pin connectors that relay information, incoming or outgoing, one byte at a time. Each byte is broken up into a *series* of eight bits, hence the term serial port. Serial ports are controlled by a special chip call a UART (Universal Asynchronous Receiver Transmitter).

1.1 Communication links:

Serial communication is classified into three types of communication.

a. Simplex communication link: In simplex transmission, the line is dedicated for transmission. The transmitter sends and the receiver receives the data.

b. Half duplex communication link: In half duplex, the communication link can be used for either transmission or reception. Data is transmitted in only one direction at a time.

c. Full duplex communication link: If the data is transmitted in both ways at the same time, it is a full duplex i.e. transmission and reception can proceed simultaneously. This communication link requires two wires for data, one for transmission and one for reception.

1.2 Types of Serial communication:

Serial data communication uses two types of communication.

a. Synchronous serial data communication: In this transmitter and receiver are synchronized. It uses a common clack to synchronize the receiver and the transmitter. First the synch character is sent and then the data is transmitted. This format is generally used for high speed transmission. In Synchronous serial data communication a block of data is transmitted at a time.

b. Asynchronous Serial data transmission: In this, different clock sources are used for transmitter and receiver. In this mode, data is transmitted with start and stop bits. A transmission begins with start bit, followed by data and then stop bit. For error

checking purpose parity bit is included just prior to stop bit. In Asynchronous serial data communication a single byte is transmitted at a time.

1.3 Baud rate:

The rate at which the bits are transmitted is called baud or transfer rate. The baud rate is the reciprocal of the time to send one bit. In asynchronous transmission, baud rate is not equal to number of bits per second. This is because; each byte is preceded by a start bit and followed by parity and stop bit. For example, in synchronous transmission, if data is transmitted with 9600 baud, it means that 9600 bits are transmitted in one second. For bit transmission time = 1 second/ 9600 = 0.104 ms.

2. RS-232 standards:

To allow compatibility among data communication equipment made by various manufactures, an interfacing standard called Rs232 was set by the Electronics Industries Association (EIA) in 1960. In 1963 it was modified and called RS232A. RS232B and RS232C were issued in 1965 and 1969, respectively. Today Rs232is the most widely used serial I/O interfacing standard. This standard is used in PCs and numerous equipments. However, since the standard was set long before the advent of logic family, its input and output voltage levels are not TTL compatible. In RS232, a logic one (1) is represented by -3 to -25V and referred as MARK while logic zero (0) is represented by +3 to +25V and referred as SPACE. For this reason to connect any RS232 to a microcontroller system we must use voltage converters such as MAX232 to convert the TTL logic level to RS232 voltage levels and vice-versa. MAX232 IC chips are commonly referred as line drivers.

2.1. Serial Data Format

The serial data format includes one start bit, between five and eight data bits, and one stop bit. A parity bit and an additional stop bit might be included in the format as well. The figure 4.1 below illustrates the serial data format.

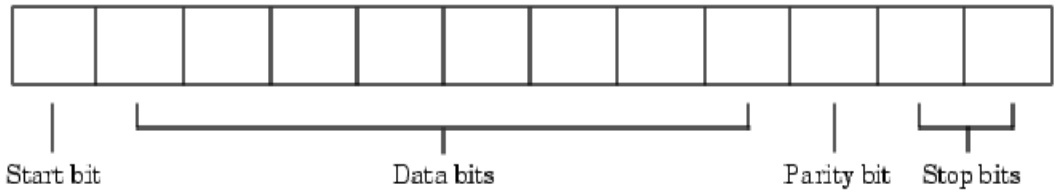

Figure 1.1: Serial data format

The format for serial port data is often expressed using the following notation: Number of data bits - parity type - number of stop bits.

For example, 8-N-1 is interpreted as eight data bits, no parity bit, and one stop bit, while 7-E-2 is interpreted as seven data bits, even parity, and two stop bits.

The data bits are often referred to as a *character* because these bits usually represent an ASCII character. The remaining bits are called *framing bits* because they frame the data bits.

2.2. Serial Port Pin Function

Pin No.	Abbreviation	Full Name	Function
Pin 3	TXD	Transmit Data	Serial Data Output
Pin 2	RXD	Receive Data	Serial Data Input
Pin 7	RTS	Request To Send	This line informs the Modem that the UART is ready to exchange data.
Pin 8	CTS	Clear To Send	This line indicates that the Modem is ready to exchange data.
Pin 6	DSR	Data Set Ready	This tells the UART that the modem is ready to establish a link.
Pin 5	SG	Signal Ground	Ground
Pin 1	DCD	Data Carrier Detect	When the modem detects a "Carrier" from the modem at the other end of the phone line, this Line becomes active.
Pin 4	DTR	Data Terminal Ready	This is the opposite to DSR. This tells the Modem that the UART is ready to link.
Pin 9	RI	Ring Indicator	Goes active when modem detects a ringing signal from the PSTN.

Table 9.1: D Type 9 Pin and Serial Port Pin Functions

2. Interfacing Diagram

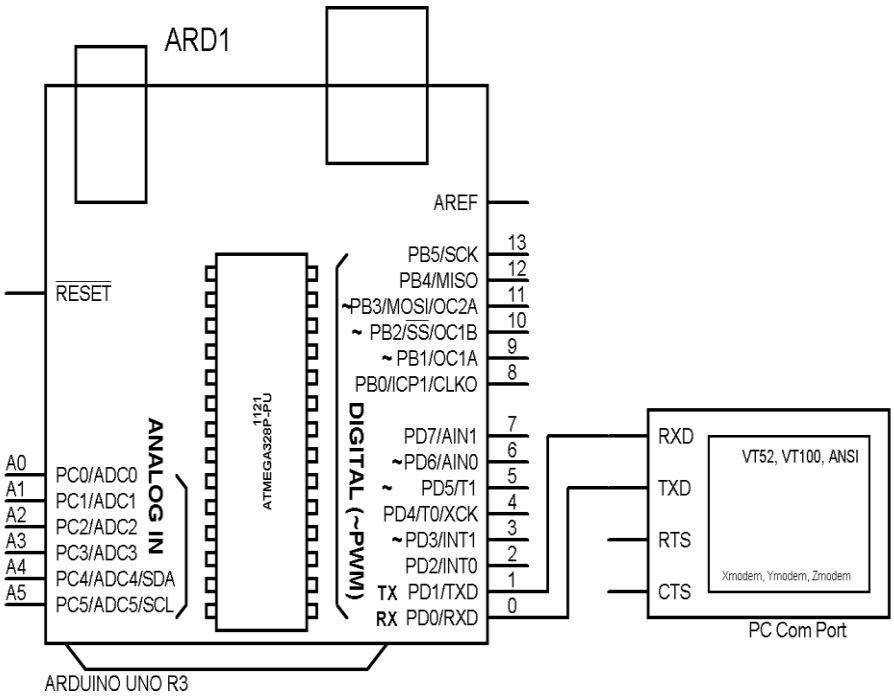

Figure 9.1 : Serial Communication

3. Algorithm

1. In Setup function
    a. Initialize the serial port with desired baud rate (e.g. 9600)
    b. Transmit the initial messages

2. In Loop function
    a. Declare the Variable with data type integer to hold the received data
    b. Check for any data is received on serial port
    c. If yes read the data into variable and retransmit the data on serial port

4. Source Code

```
void setup() {
   Serial.begin(9600); // initialize  serial ports:
   Serial.println("Smart Logic Technologies");
   Serial.println("Serial Communication Demo");
   Serial.println("Please send data");
}

void loop() {
  // read from com port, and send it again
  if (Serial.available()) {
    int inByte = Serial.read();
    Serial.write(inByte);
  }
}
```