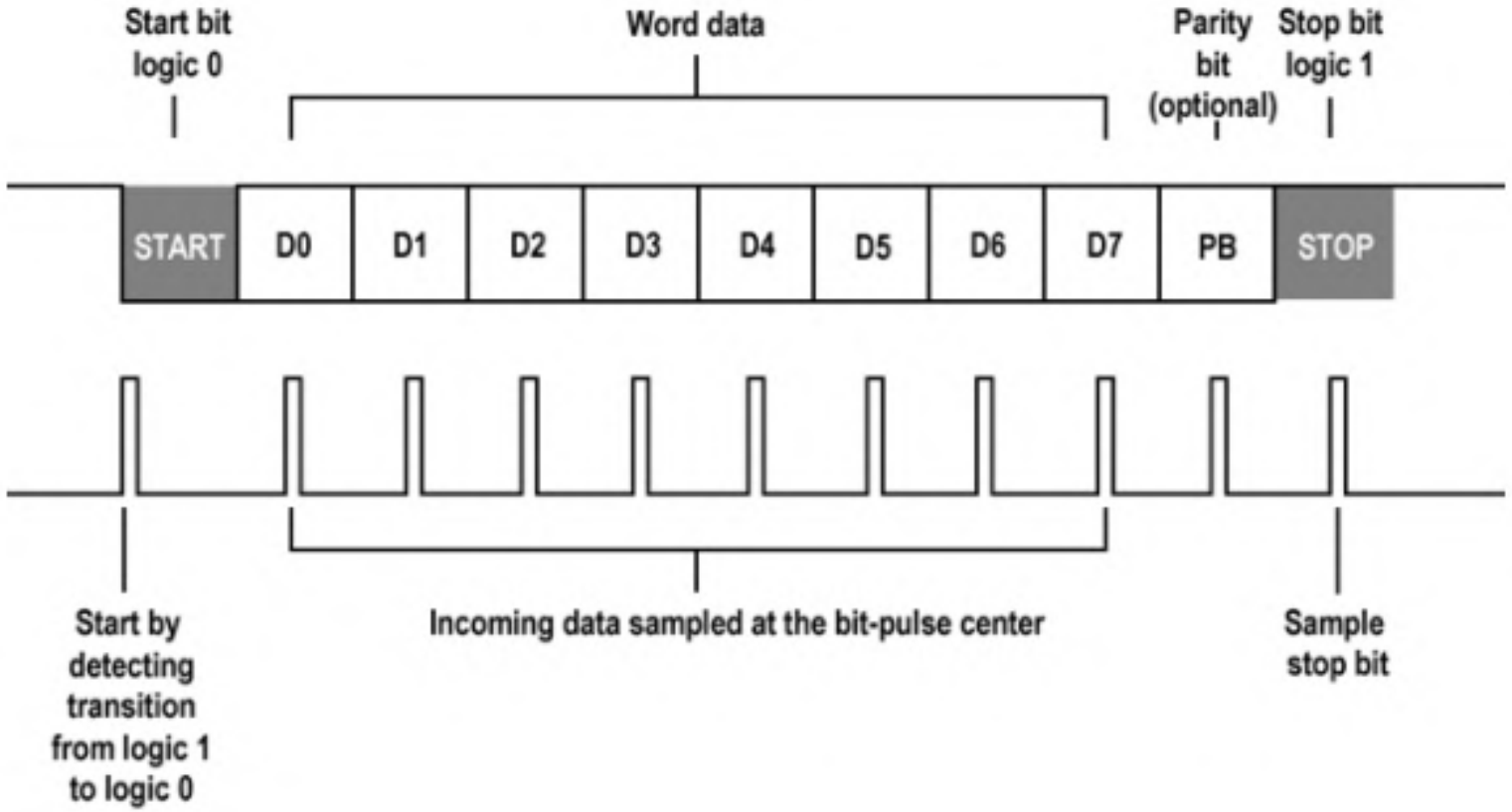
USART

By: Eng. R. Saahith Ahamed.
Department of Electrical and Information Engineering
Faculty of Engineering
University of Ruhuna

USART Frame



Frame consist of

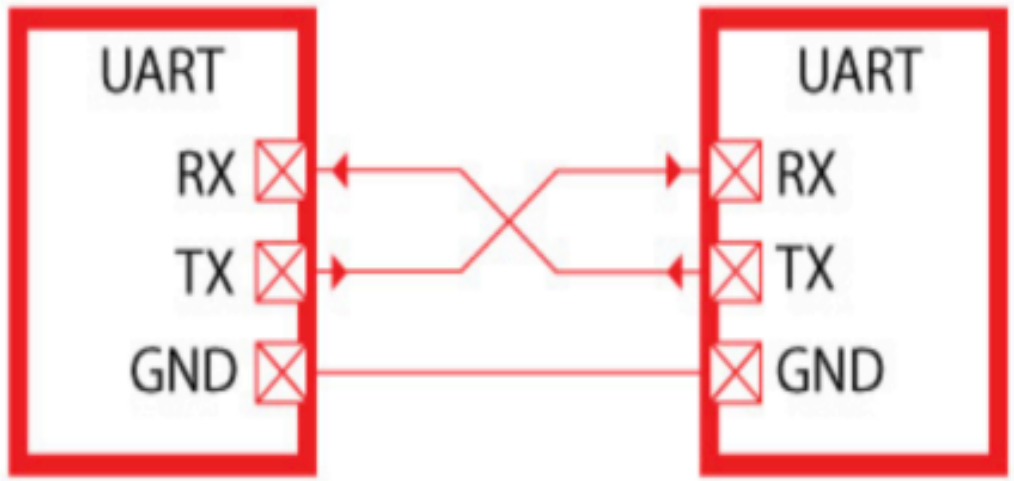
- One Start bit
- Data bits 5 to 8
- One or two stop bits
- Optional parity bit

Transmitter and receiver has to agree on

- BAUD rate
- Number of data bits
- Number of stop bits
- Number of parity bits

USART Connection

A two-wire connection Tx and Rx along with the Ground connection



SERIAL - UART

Rx and/or Tx of one device should be connected to Tx and/or Rx of the other device, respectively

Table 19-1 contains equations for calculating the baud rate (in bits per second) and for calculating the UBRRn value for each mode of operation using an internally generated clock source.

Table 19-1. Equations for Calculating Baud Rate Register Setting

Operating Mode	Equation for Calculating Baud Rate[1]	Equation for Calculating UBRRn Value
Asynchronous normal mode (U2Xn = 0)	$BAUD = \frac{f_{OSC}}{16(UBRRn + 1)}$	$UBRRn = \frac{f_{OSC}}{16BAUD} - 1$
Asynchronous double speed mode (U2Xn = 1)	$BAUD = \frac{f_{OSC}}{8(UBRRn + 1)}$	$UBRRn = \frac{f_{OSC}}{8BAUD} - 1$
Synchronous master mode	$BAUD = \frac{f_{OSC}}{8(UBRRn + 1)}$	$UBRRn = \frac{f_{OSC}}{2BAUD} - 1$

Note: 1. The baud rate is defined to be the transfer rate in bit per second (bps)

BAUD Baud rate (in bits per second, bps)

f_{OSC} System oscillator clock frequency

UBRRn Contents of the UBRRnH and UBRRnL registers, (0-4095)

Table 19-1 contains equations for calculating the baud rate (in bits per second) and for calculating the UBRRn value for each mode of operation using an internally generated clock source.

Table 19-1. Equations for Calculating Baud Rate Register Setting

Operating Mode	Equation for Calculating Baud Rate[1]	Equation for Calculating UBRRn Value
Asynchronous normal mode (U2Xn = 0)	$BAUD = \frac{f_{OSC}}{16(UBRRn + 1)}$	$UBRRn = \frac{f_{OSC}}{16BAUD} - 1$
Asynchronous double speed mode (U2Xn = 1)	$BAUD = \frac{f_{OSC}}{8(UBRRn + 1)}$	$UBRRn = \frac{f_{OSC}}{8BAUD} - 1$
Synchronous master mode	$BAUD = \frac{f_{OSC}}{8(UBRRn + 1)}$	$UBRRn = \frac{f_{OSC}}{2BAUD} - 1$

Note: 1. The baud rate is defined to be the transfer rate in bit per second (bps)

BAUD Baud rate (in bits per second, bps)

f_{OSC} System oscillator clock frequency

UBRRn Contents of the UBRRnH and UBRRnL registers, (0-4095)

UBRR register is a 16-bit Higher 8-bits are UBRR0H and Lower 8-bits are UBRR0L

Name: UBRR0L
Offset: 0xC4
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	UBRR0[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

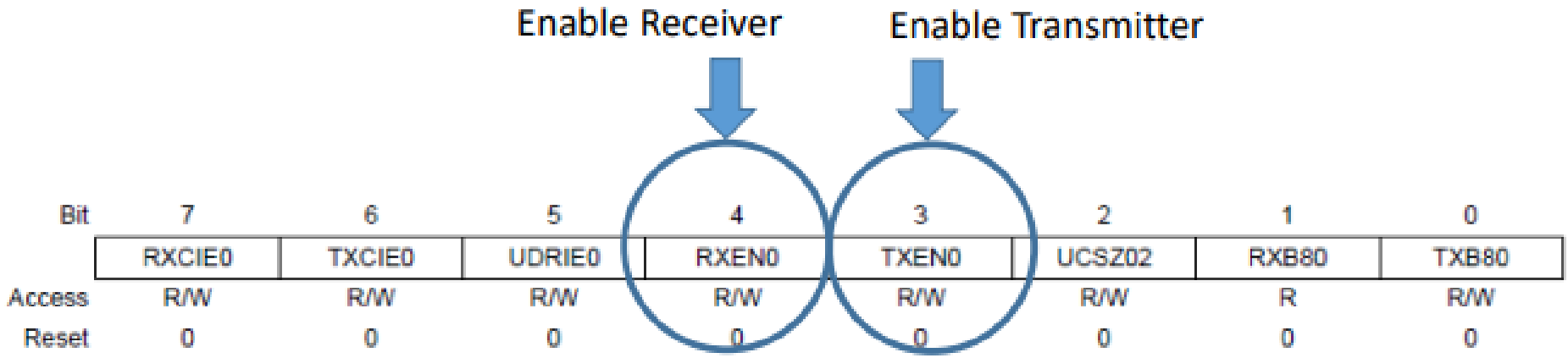
Name: UBRR0H
Offset: 0xC5
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
					UBRR0[3:0]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

USART Configuration : Enabling Tx/Rx

Transmitter and Receiver has to be separately enabled before use

Name: UCSR0B
Offset: 0xC1
Reset: 0x00
Property: -



Bit	7	6	5	4	3	2	1	0
	RXCIE0	TXCIE0	UDRIE0	RXEN0	TXEN0	UCSZ02	RXB80	TXB80
Access	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
Reset	0	0	0	0	0	0	0	0

USART Configuration : Frame Format

Number of bits in each transmit frame is configured by UCSZ02:0 bits

Name: UCSR0B
Offset: 0xC1
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	RXCIE0	TXCIE0	UDRIE0	RXEN0	TXEN0	UCSZ02	RXB80	TXB80
Access	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
Reset	0	0	0	0	0	0	0	0

Name: UCSR0C
Offset: 0xC2
Reset: 0x06
Property: -

Bit	7	6	5	4	3	2	1	0
	UMSEL01	UMSEL00	UPM01	UPM00	USBS0	UCSZ01 / UDORD0	UCSZ00 / UCPHA0	UCPOL0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0			0

UCSZ0[2:0]	Character Size
000	5-bit
001	6-bit
010	7-bit
011	8-bit
100	Reserved
101	Reserved
110	Reserved
111	9-bit

USART Configuration : Frame Format

Name: UCSR0C
Offset: 0xC2
Reset: 0x06
Property: -

Bit	7	6	5	4	3	2	1	0
	UMSEL01	UMSEL00	UPM01	UPM00	USBS0	UCSZ01 / UDORD0	UCSZ00 / UCPHA0	UCPOL0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	1	1	0

UMSEL0[1:0]	Mode
00	Asynchronous USART
01	Synchronous USART
10	Reserved
11	Master SPI (MSPIM)(1)

UPM0[1:0]	ParityMode
00	Disabled
01	Reserved
10	Enabled, Even Parity
11	Enabled, Odd Parity

USBS0	Stop Bit(s)
0	1-bit
1	2-bit

Mode of operation

Number of Parity bits in the USART frame.

Number of stop bits associated with the USART frame.

USART Transmitter

Name: UCSR0A
Offset: 0xC0
Reset: 0x20
Property: -

Bit	7	6	5	4	3	2	1	0
	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0
Access	R	R/W	R	R	R	R	R/W	R/W
Reset	0	0	1	0	0	0	0	0

Bit 5 – UDRE0: USART Data Register Empty

The UDRE0 Flag indicates if the transmit buffer (UDR0) is ready to receive new data. If UDRE0 is one, the buffer is empty, and therefore ready to be written. The UDRE0 Flag can generate a Data Register Empty interrupt (see description of the UDRIE0 bit). UDRE0 is set after a reset to indicate that the Transmitter is ready.

Name: UDR0
Offset: 0xC6
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	TXB / RXB[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

USART Transmitter : Task 01

Specification: You are required to write a program that should transmit ESD6304 via serial port and display the characters via serial monitor of Proteus simulator. Repeat the operation over and over.

Keep an LED attached to PORTB PIN5 ON throughout the operation.

USART Transmitter : Settings

Assuming the required baud rate to be 9600

Value set at the UBRR register = 16,000,000 / (16 x 9600) -1

MYUBRR = 16,000,000 / (16 x 9600) -1

USART Transmitter : Settings

Name: UCSR0C
Offset: 0xC2
Reset: 0x06
Property: -

Bit	7	6	5	4	3	2	1	0
	UMSEL01	UMSEL00	UPM01	UPM00	USBS0	UCSZ01 / UDORD0	UCSZ00 / UCPHA0	UCPOL0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0		1	0

UMSEL0[1:0]	Mode
00	Asynchronous USART
01	Synchronous USART
10	Reserved
11	Master SPI (MSPIM)(1)

UPM0[1:0]	ParityMode
00	Disabled
01	Reserved
10	Enabled, Even Parity
11	Enabled, Odd Parity

USBS0	Stop Bit(s)
0	1-bit
1	2-bit

UCSR0C &= 0B00000111

USART Transmitter : Settings

Name: UCSR0B
Offset: 0xC1
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	RXCIE0	TXCIE0	UDRIE0	RXEN0	TXEN0	UCSZ02	RXB80	TXB80
Access	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
Reset	0	0	0	0	0	0	0	0

Name: UCSR0C
Offset: 0xC2
Reset: 0x06
Property: -

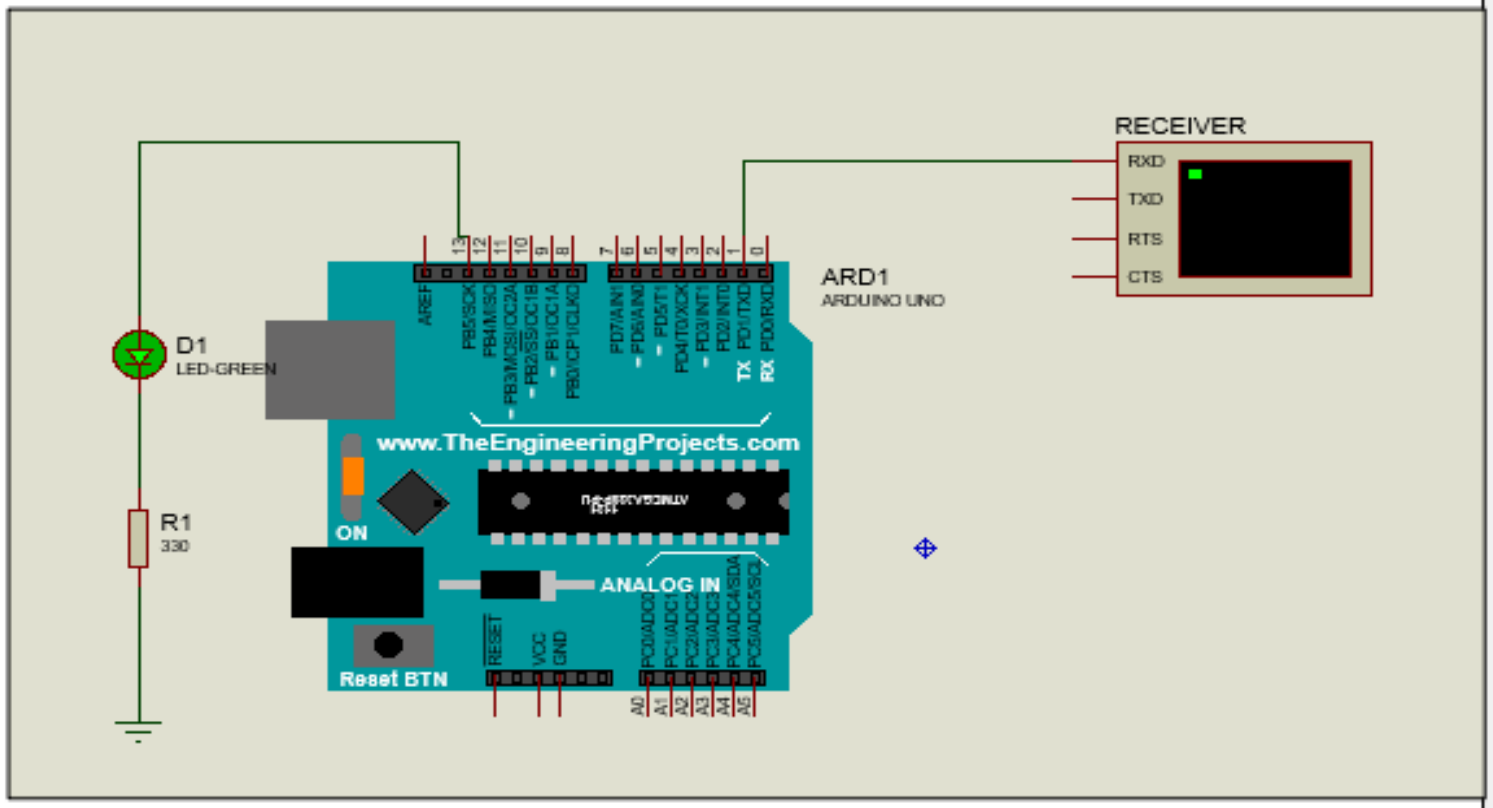
Bit	7	6	5	4	3	2	1	0
	UMSEL01	UMSEL00	UPM01	UPM00	USBS0	UCSZ01 / UDORD0	UCSZ00 / UCPHA0	UCPOL0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	1	1	0

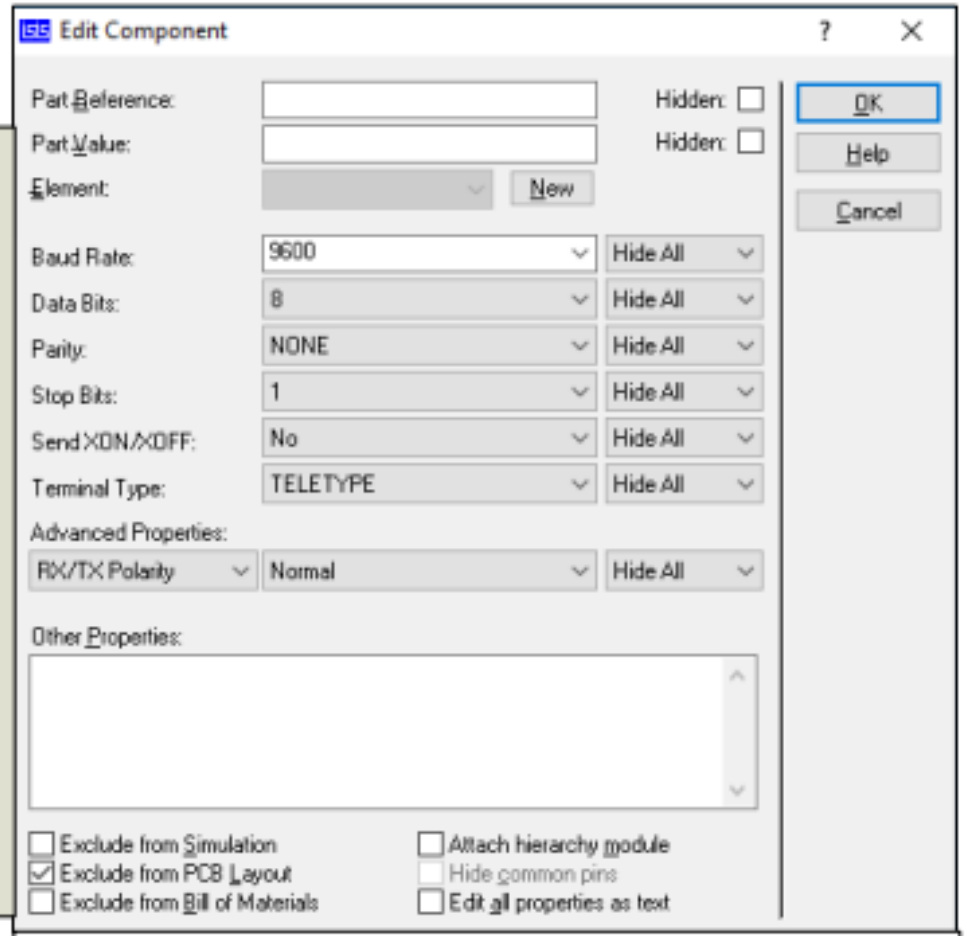
UCSZ0[2:0]	Character Size
000	5-bit
001	6-bit
010	7-bit
011	8-bit
100	Reserved
101	Reserved
110	Reserved
111	9-bit

USART Transmitter : Configuration Code

USART Transmitter : Transmit Code

Generate the Simulation Circuit in Proteus





Implement the circuit shown in the Figure

Make sure that the TX pin of Arduino board is connected to RX pin of the virtual terminal

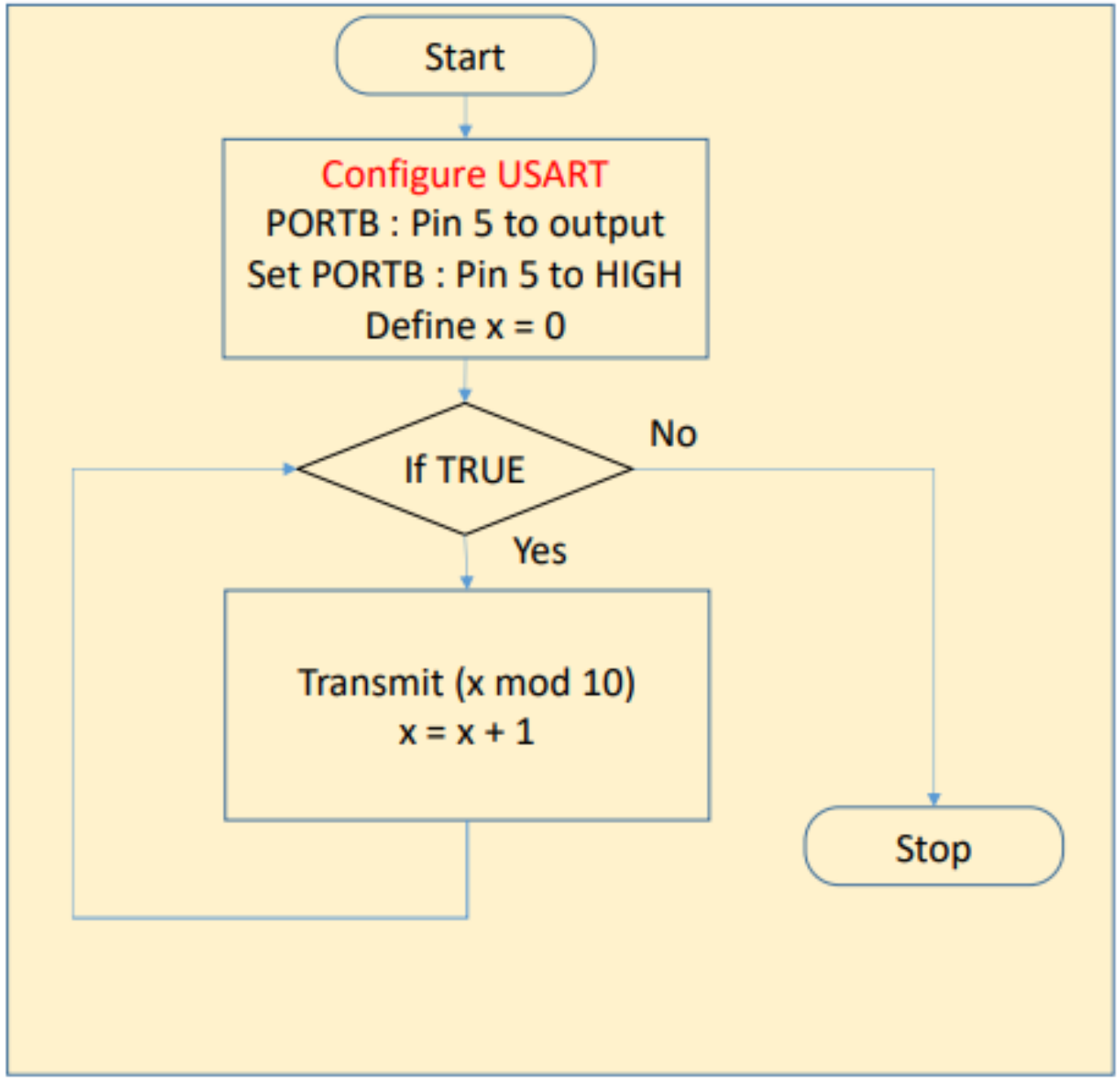
Right click on the virtual terminal and set properties similar to what you set in your code as shown in above figure

BAUD, PARITY, DATA BITS etc.

USART Transmitter : Task 02

Specification: You are required to write a program that should transmit integers from 0 to 9 via serial port and display the characters via serial monitor of Proteus simulator. Repeat the operation over and over.

Keep an LED attached to PORTB PIN 5 ON throughout the operation.



USART Transmitter : Configuration Code

USART Transmitter : Transmit Code

USART Transmitter : Transmit Code

```
void USART_Transmit(unsigned char data)
{



}

int main(void)
{
    Config_USART();
    DDRB  |= (1<<PORTB5); // PORTB:PIN5 to output
    PORTB |= (1<<PORTB5); // PORTB:PIN5 to  High


    int x = 0;
    while (1)
    {
        USART_Transmit(          );
        x++;
        _delay_ms(250);
    }
    return 0;
}
```

Ascii	Char	Ascii	Char
0	Null	32	Space
1	Start of heading	33	!
2	Start of text	34	"
3	End of text	35	#
4	End of transmit	36	$
5	Enquiry	37	%
6	Acknowledge	38	&
7	Audible bell	39	'
8	Backspace	40	(
9	Horizontal tab	41	)
10	Line feed	42	*
11	Vertical tab	43	+
12	Form feed	44	,
13	Carriage return	45	-
14	Shift in	46	.
15	Shift out	47	/
16	Data link escape	48	0
17	Device control 1	49	1
18	Device control 2	50	2
19	Device control 3	51	3
20	Device control 4	52	4
21	Neg. acknowledge	53	5
22	Synchronous idle	54	6
23	End trans. block	55	7
24	Cancel	56	8
25	End of medium	57	9
26	Substitution	58	:
27	Escape	59	;
28	File separator	60	<
29	Group separator	61	=
30	Record separator	62	>
31	Unit separator	63	?

0x30

0x39

Receive Data Through USART : Enabling the Receiver

Enabling the receiver

Name: UCSR0B
Offset: 0xC1
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	RXCIE0	TXCIE0	UDRIE0	RXEN0	TXEN0	UCSZ02	RXB80	TXB80
Access	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
Reset	0	0	0	0	0	0	0	0

```
UCSR0B |= (1 << RXEN0)
```

Receive Data Through USART : Identifying Data in Receiver

Name: UCSR0A
Offset: 0xC0
Reset: 0x20
Property: -

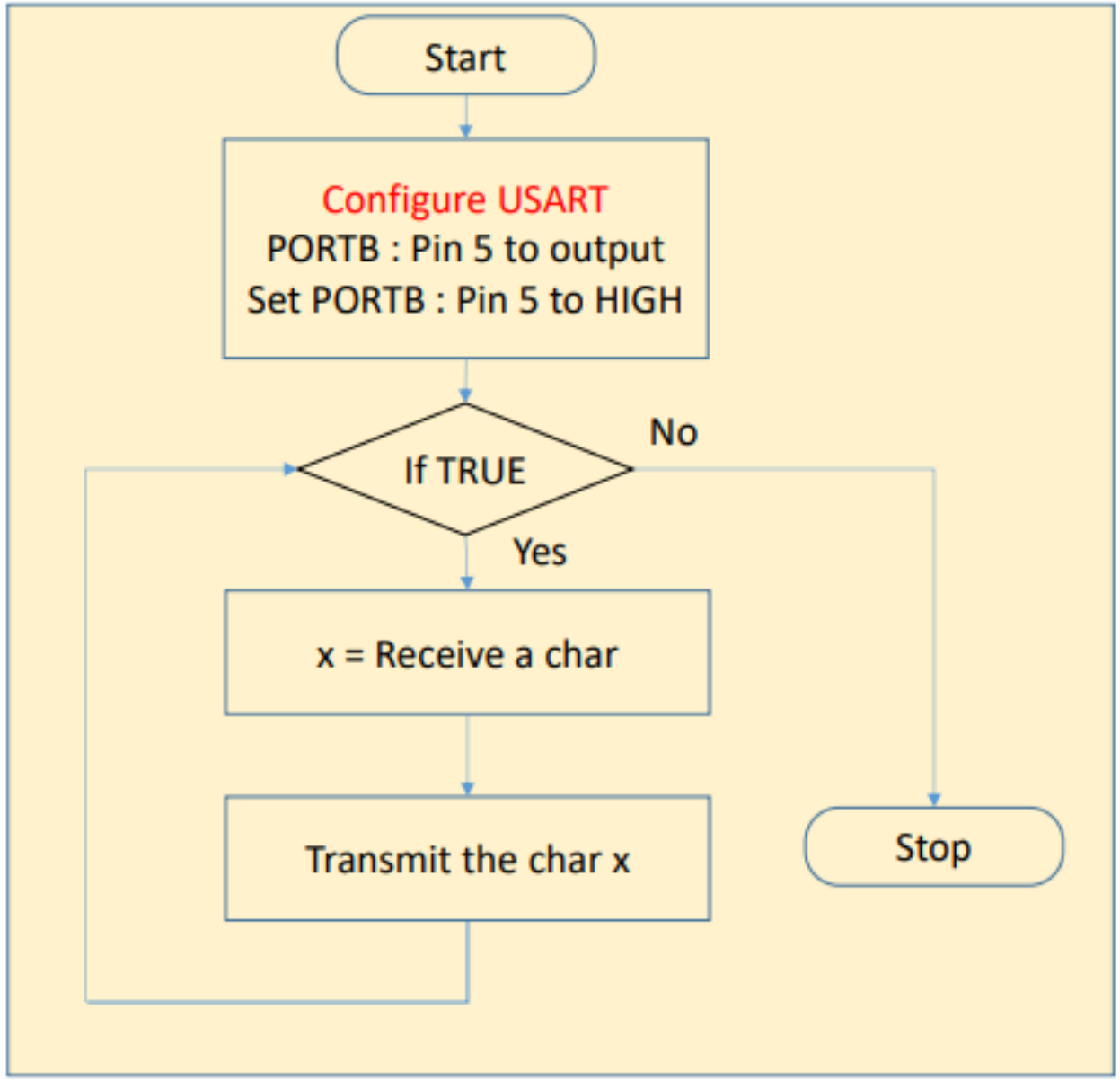
Bit	7	6	5	4	3	2	1	0
	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0
Access	R	R/W	R	R	R	R	R/W	R/W
Reset	0	0	1	0	0	0	0	0

Bit 7 – RXC0: USART Receive Complete

This flag bit is set when there are unread data in the receive buffer and cleared when the receive buffer is empty (i.e., does not contain any unread data). If the Receiver is disabled, the receive buffer will be flushed and consequently the RXC0 bit will become zero. The RXC0 Flag can be used to generate a Receive Complete interrupt (see description of the RXCIE0 bit).

USART Transmitter and Receiver : Task

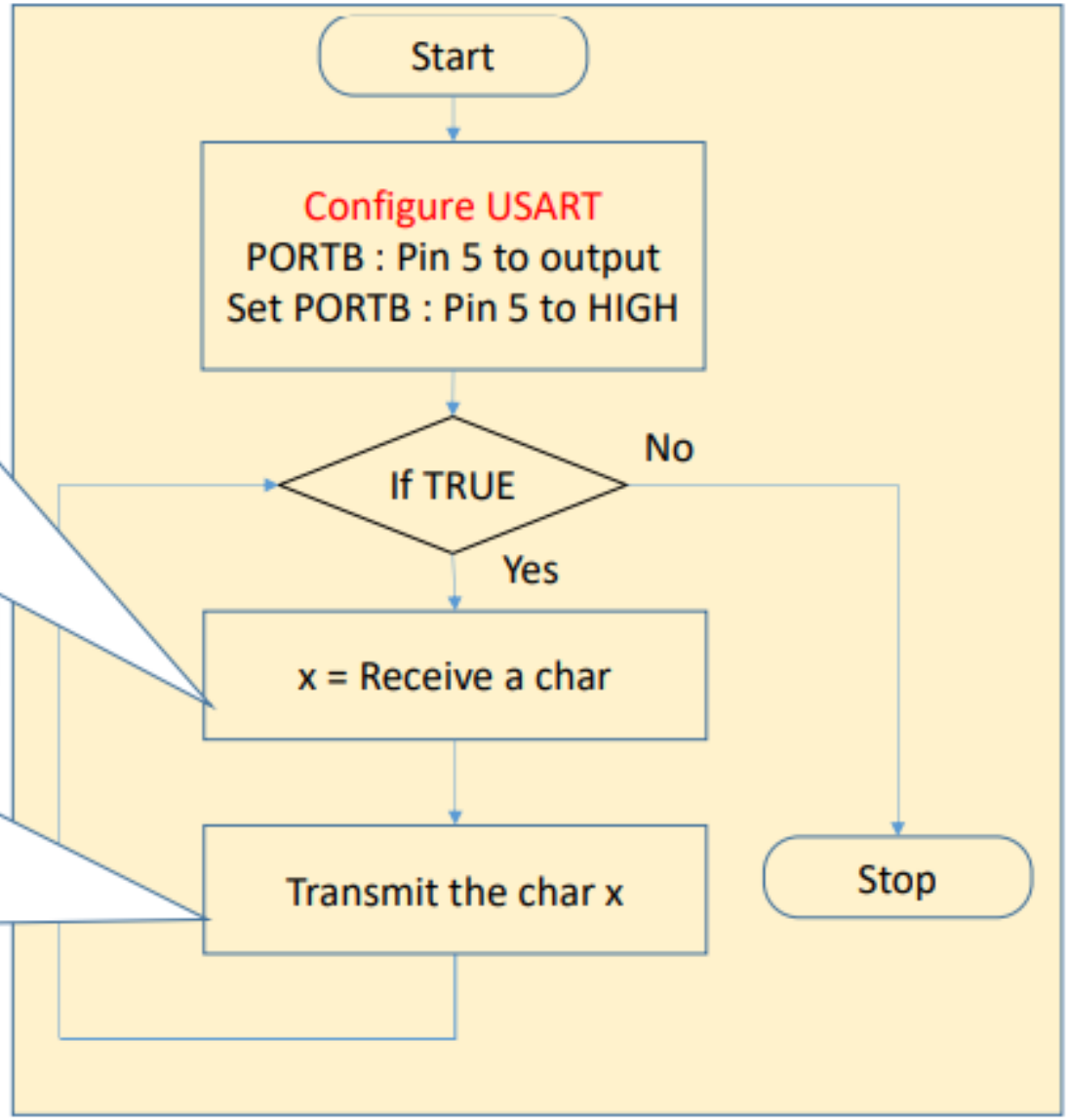
You are required to write a program that should receive data via USART RX pin, transmit the same via USART TX to the virtual terminal and display the characters via serial monitor



USART Transmitter and Receiver : Code

```
unsigned char USART_Receive( void )
{
    /* Wait for data to be received */

    ;
    /* Get and return received data from buffer */

}
```

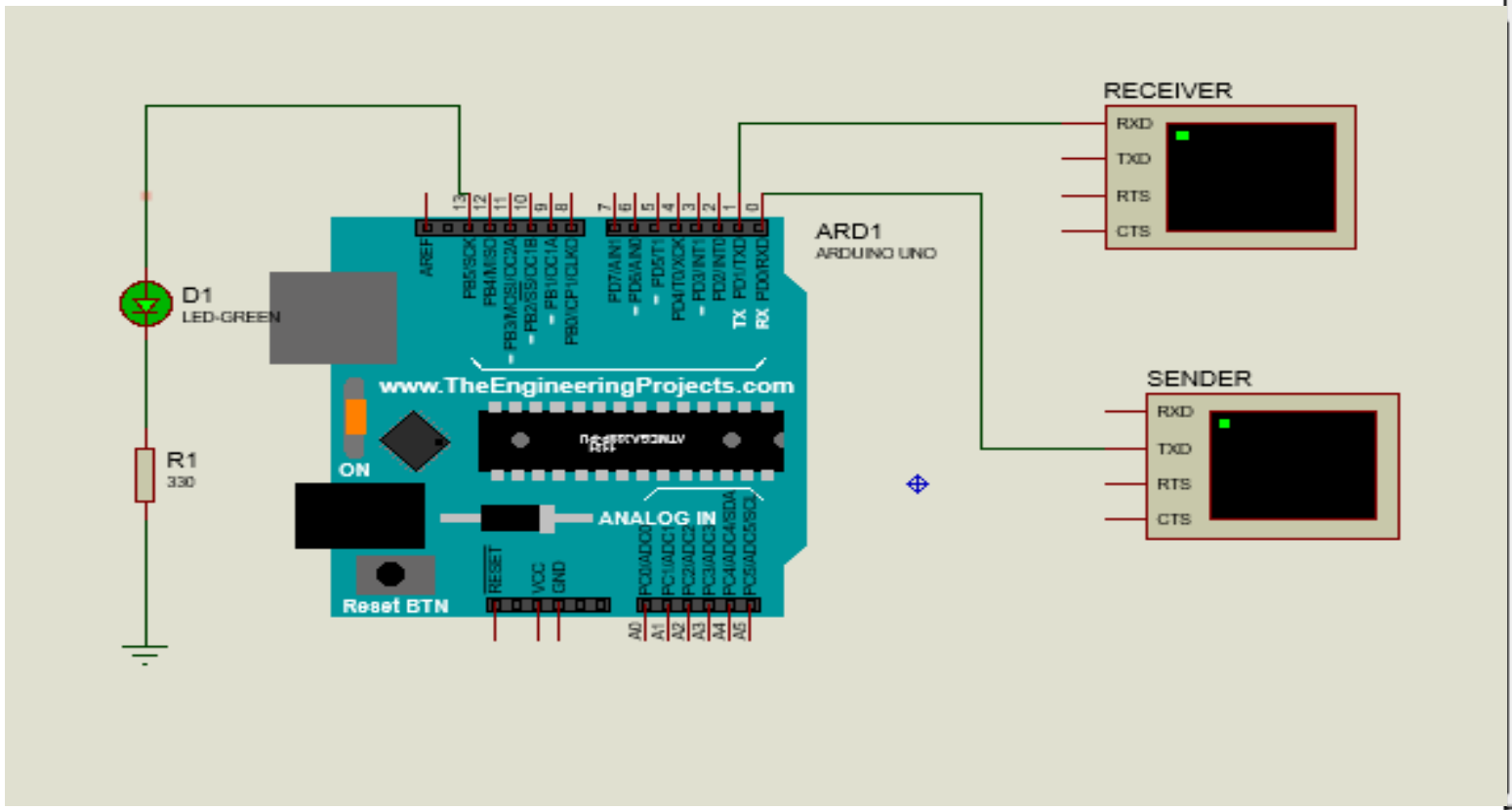
```
void USART_Transmit( unsigned char data )
{



}
```

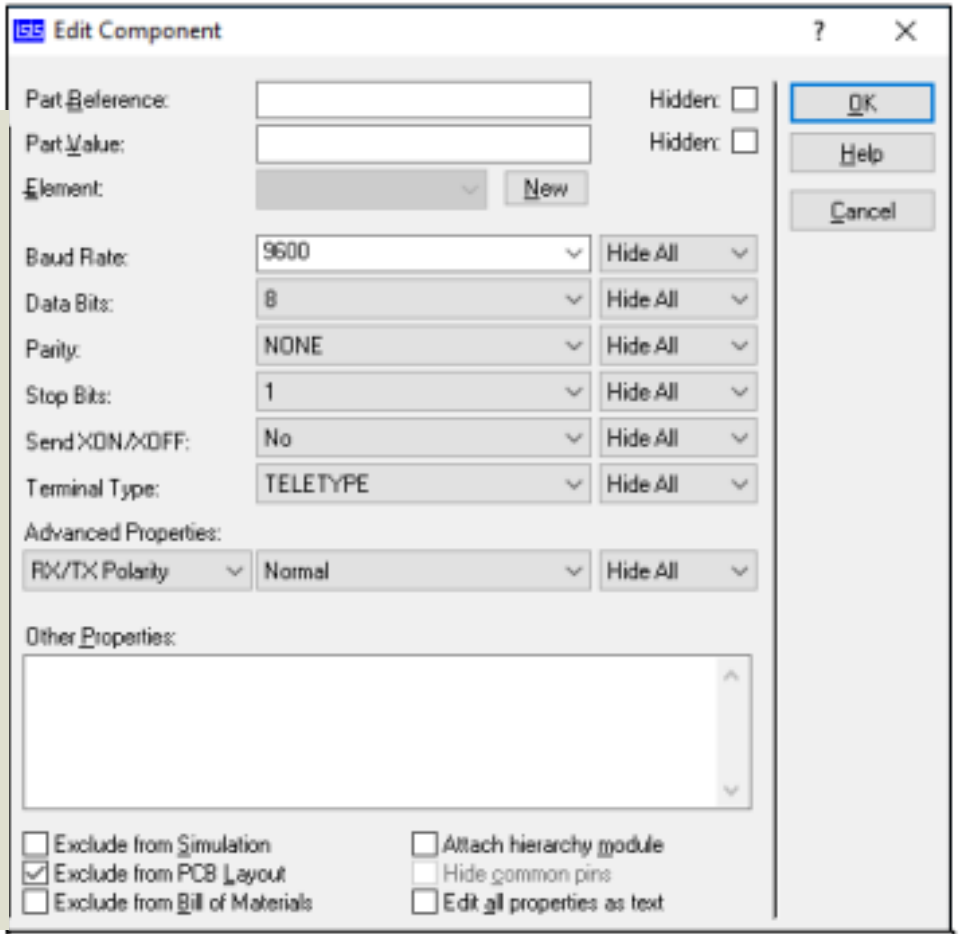


USART Transmitter and Receiver : Code

```
int main(void)
{
    Config_USART();
    DDRB  |= (1<<PORTB5); // PORTB:PIN5 to output
    PORTB |= (1<<PORTB5); // PORTB:PIN5 to  High


    unsigned char rx;
    while (1)
    {


    }
    return 0;
}
```

Generate the Simulation Circuit in Proteus





Implement the circuit shown in the Figure

Make sure that the TX pin of Arduino board is connected to RX pin of the virtual terminal

Right click on the virtual terminal and set properties similar to what you set in your code as shown in above figure

BAUD, PARITY, DATA BITS etc.

Data Receive and Transmit Using an Interrupt

In the previous exercise the receiver was waiting (polling) for data.

While waiting for data, the main program could not do anything else.

Rather than waiting for a data byte, main program can run independently and receive a data byte in an ISR.

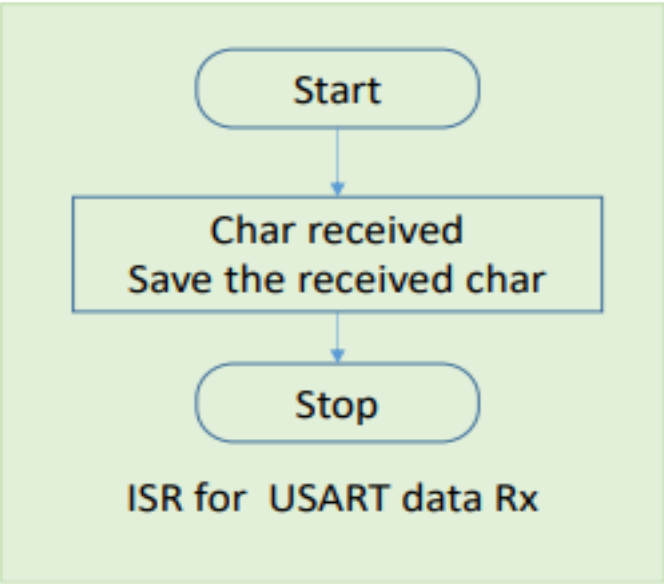
Name: UCSR0B
Offset: 0xC1
Reset: 0x00
Property: -

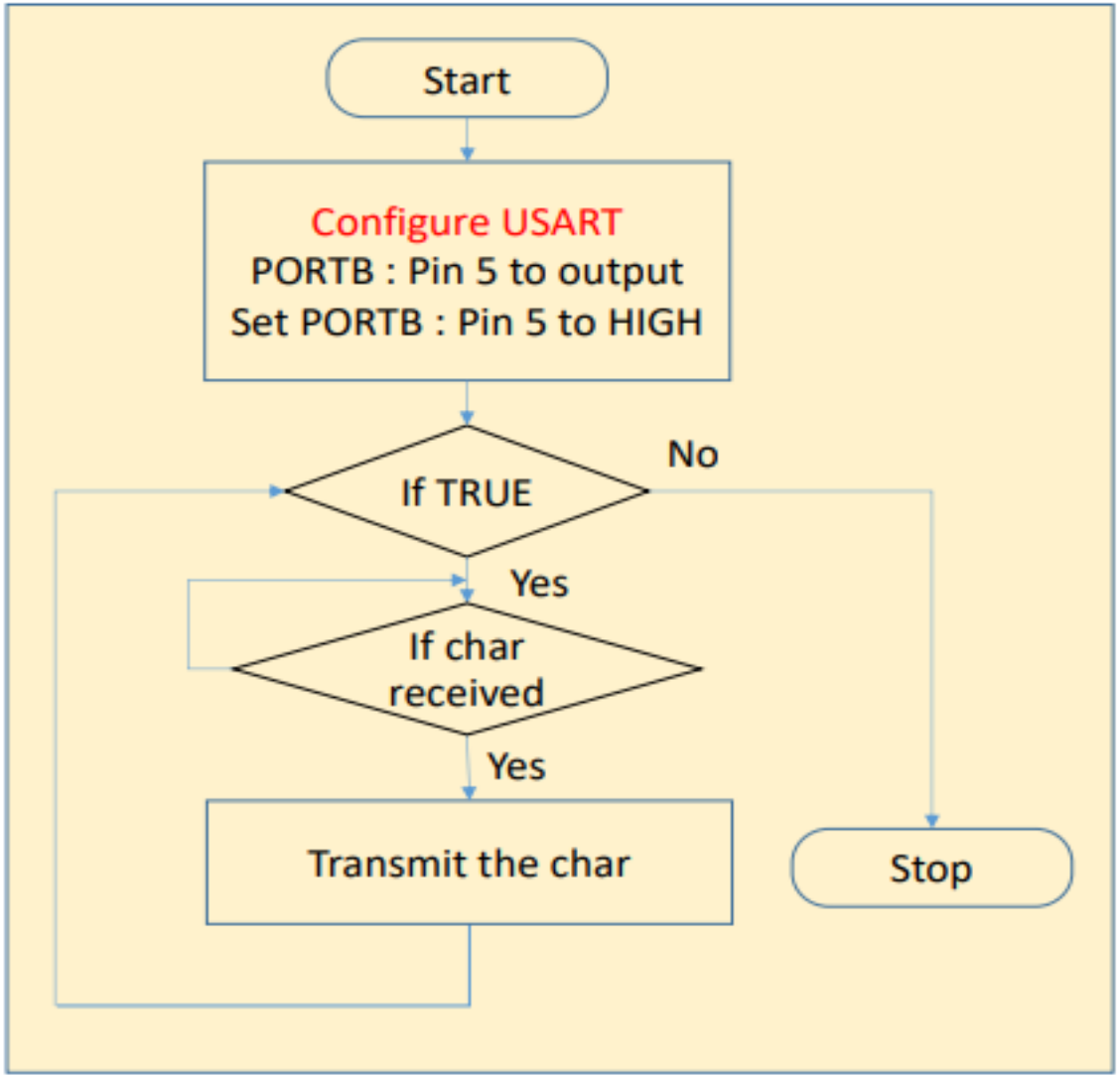
Bit	7	6	5	4	3	2	1	0
	RXCIE0	TXCIE0	UDRIE0	RXEN0	TXEN0	UCSZ02	RXB80	TXB80
Access	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
Reset	0	0	0	0	0	0	0	0

```
UCSR0B |= (1 << RXCIE0) | (1 << RXEN0)
```

USART Transmitter and Receiver Using an Interrupt: Task

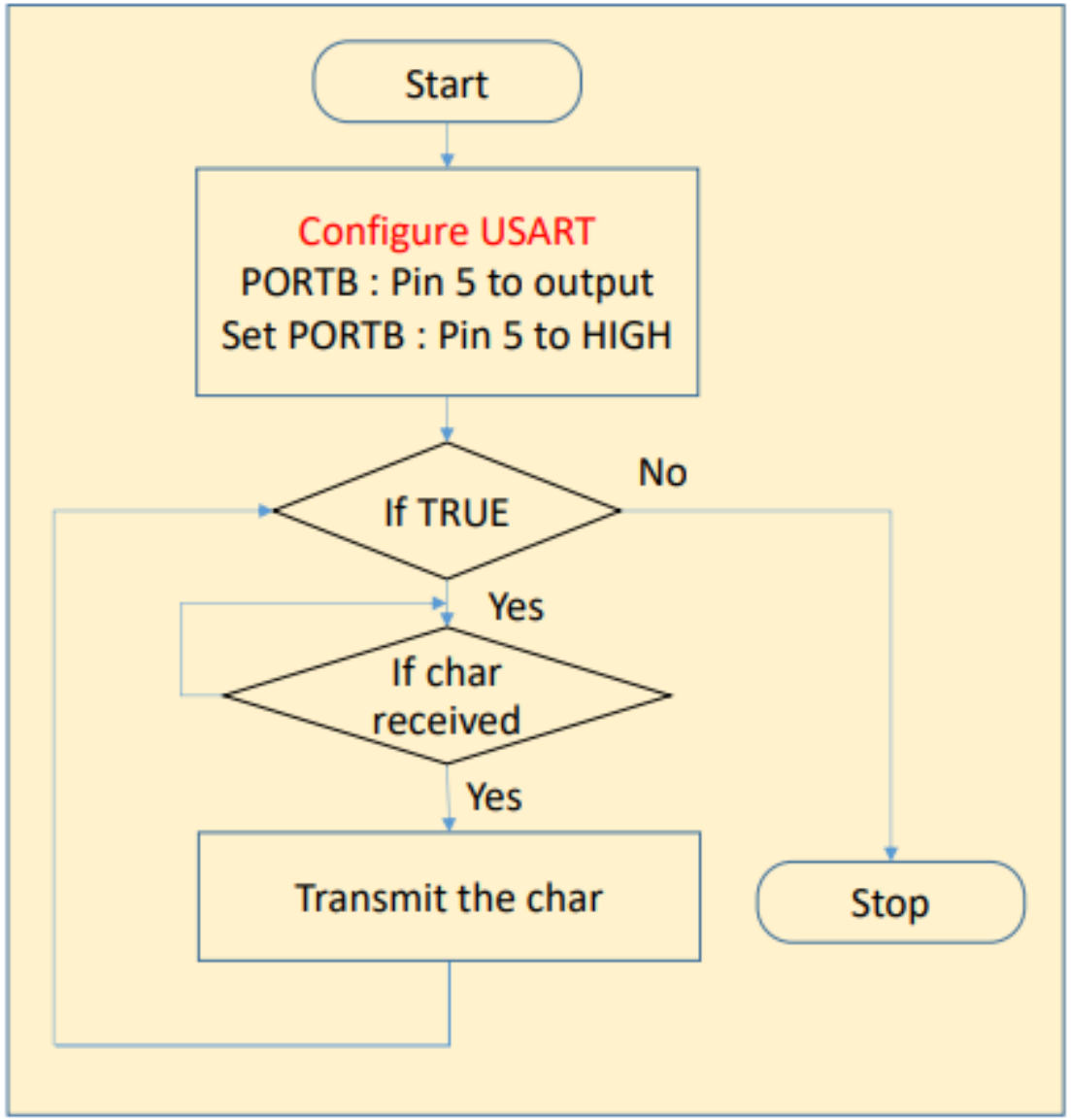
You are required to write a program that should receive data via USART RX pin, transmit the same via USART TX to the virtual terminal and display the characters via serial monitor





USART Transmitter and Receiver Using an Interrupt: Task

You are required to write a program that should receive data via USART RX pin, transmit the same via USART TX to the virtual terminal and display the characters via serial monitor



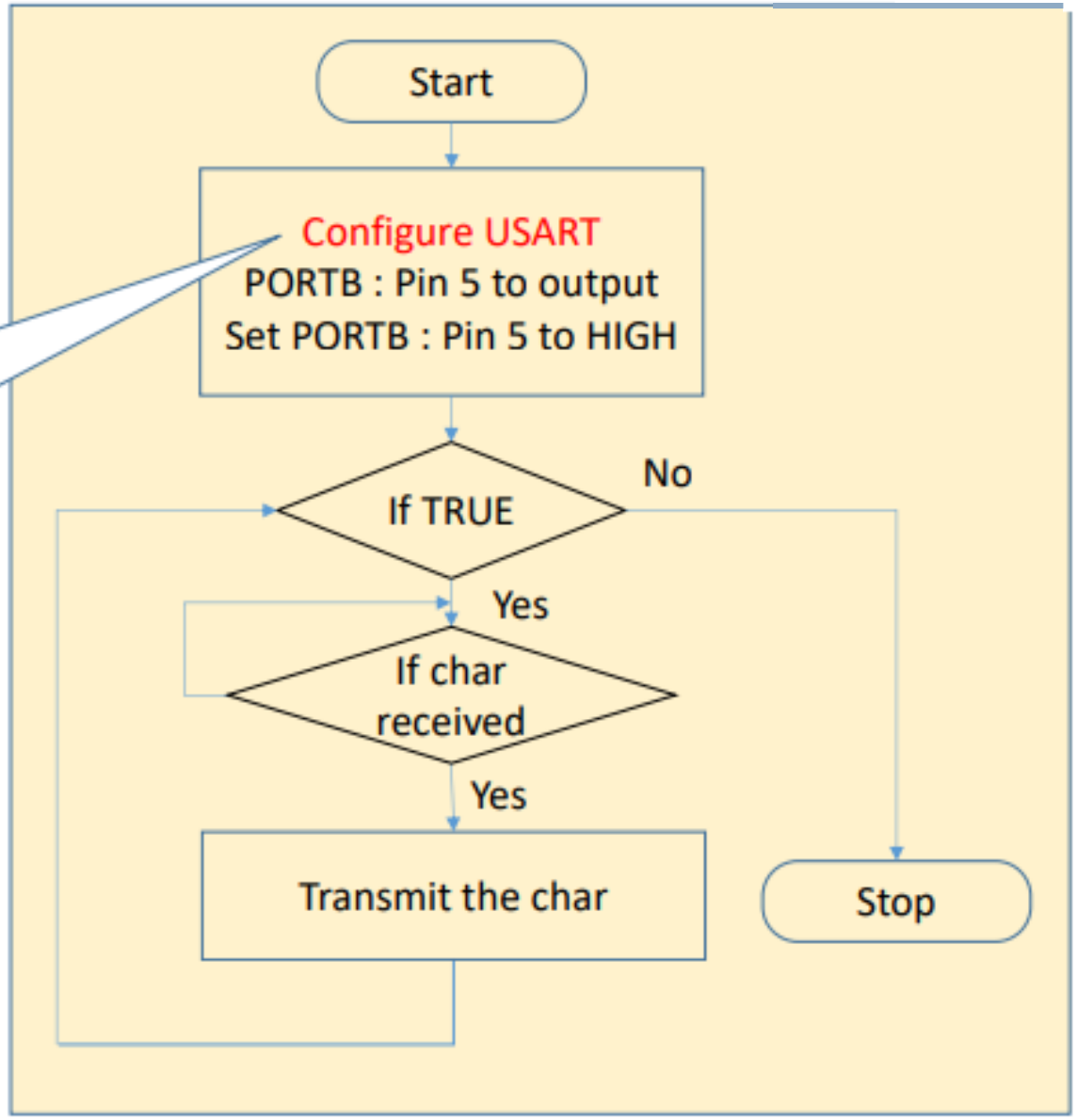
USART Transmitter and Receiver Using an Interrupt: Task

You are required to write a program that should receive data via USART RX pin, transmit the same via USART TX to the virtual terminal and display the characters via serial monitor

```
void Config_USART(void)
{
    UBRR0L = (unsigned char) UBRR_VALUE;          //Setting UBRR0 Lower byte
    UBRR0H = (unsigned char) (UBRR_VALUE>>8); //Setting UBRR0 Upper byte

    UCSR0B |= (1<<RXCIE0) |(1<<TXEN0) |(1<<RXEN0);  //Enable transmitter & Receiver
    UCSR0C |= (1<<UCSZ01) | (1<<UCSZ00);//Setting 8-bit data
    sei();
}
```

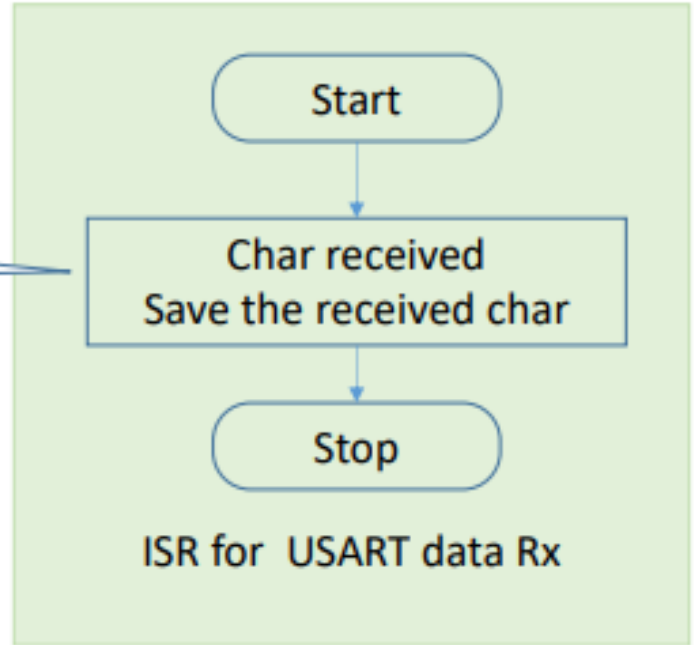
Enable Global interrupt

Enable Receive interrupt



USART Transmitter and Receiver Using an Interrupt: Task

```
ISR (USART_RX_vect){
    rxFlag = 1;
    rxData = UDR0;
}
```

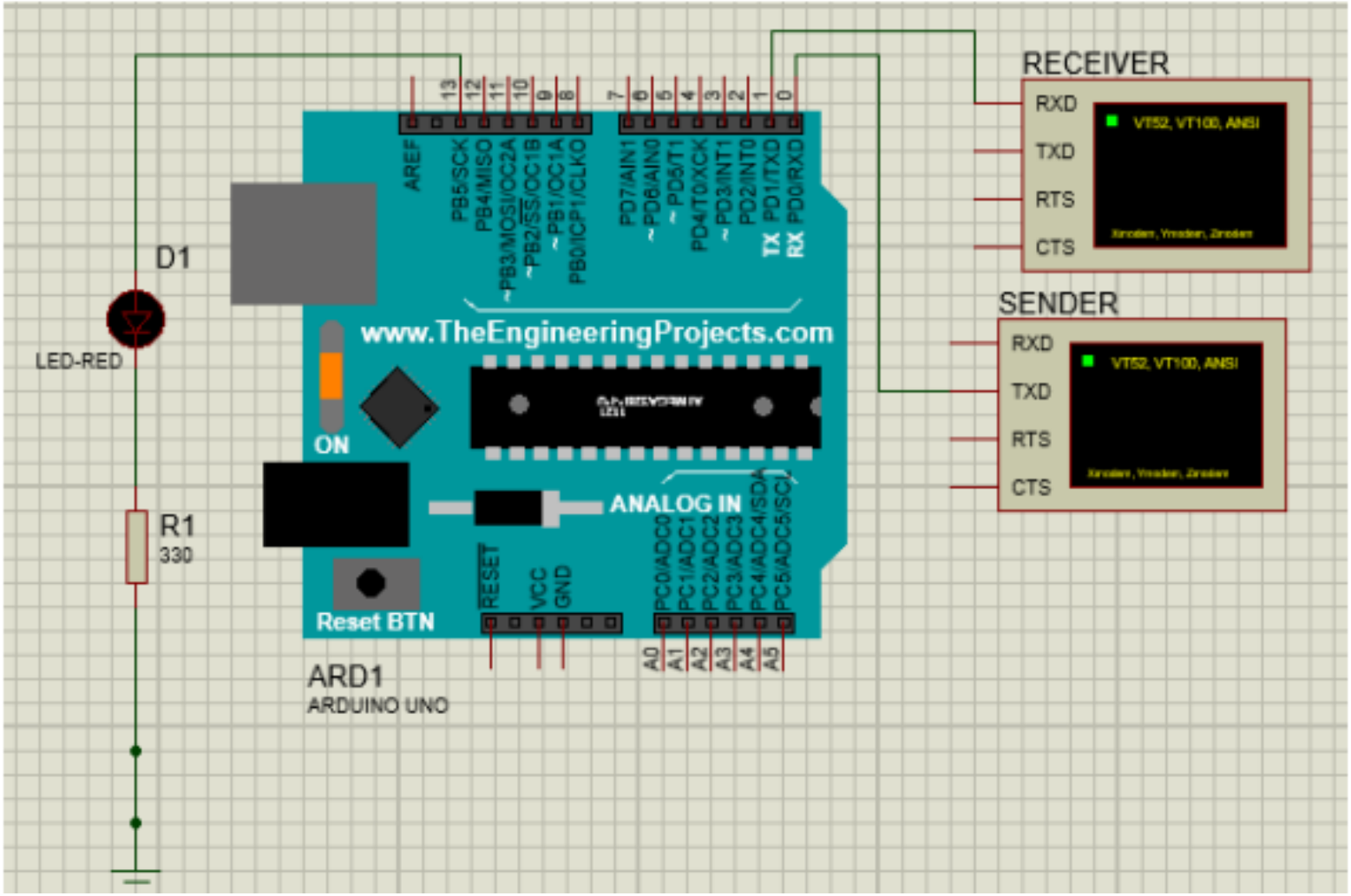


USART Transmitter and Receiver Using an Interrupt: Task

```
volatile int rx_flag = 0; // 1: data rx, 0: waiting for data
volatile char rx_data;
int main(void)
{
    Config_USART();
    DDRB  |= (1<<PORTB5); // PORTB:PIN5 to output
    PORTB |= (1<<PORTB5); // PORTB:PIN5 to  High

    while (1)
    {
        if (rx_flag)
        {
            USART_Transmit('r');
            USART_Transmit('x');
            USART_Transmit('>');
            USART_Transmit(rx_data);
            USART_Transmit(0x0A);USART_Transmit(0x0D);
            rx_flag = 0;
        }
    }
    return 0;
}
```

If the **rxFlag** set to 1 transmit **rxData**.

USART Transmitter and Receiver Using an Interrupt: Simulation



Now you are ready to transmit data from SENDER virtual terminal, receive at the Atmega 328P microcontroller

then transmit received data to the RECEIVER virtual terminal

Type in the SENDER terminal and see it appears on the RECEIVER terminal Run the simulation

USART Settings

Why USART is said to be asynchronous?

What is BAUD rate?

List the settings you should match between transmitter and receiver