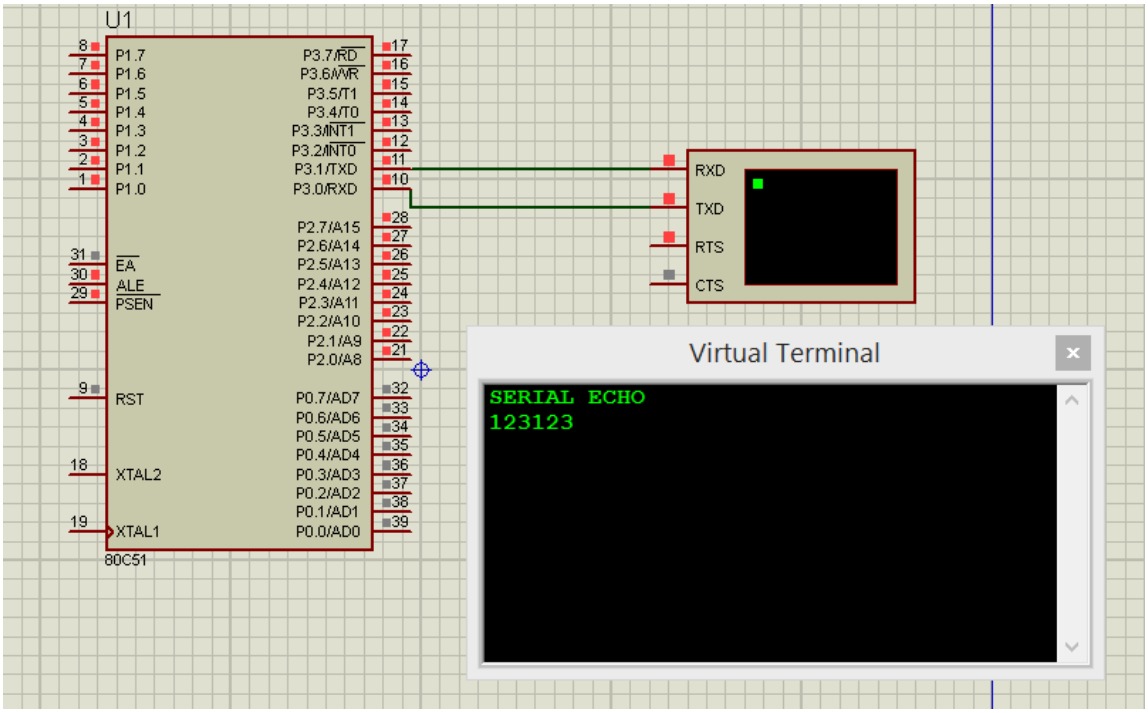
Problem 18



This project is a serial communication using TX and RX pins of the microcontroller (P3.1, P3.0).

Character received at RX pin is sent again through TX pin (This is called echo).

The internal Serial Controller of the 8051 must be programmed for a certain baud rate (bit rate/speed).

Main code

```
25 Start:
26    MOV SCON,#50H        ;Asynchronous mode, 8-bit data and 1-stop bit
27    MOV TMOD,#20H        ;Timer1 in Mode2.
28    MOV TH1,#253         ; // Load timer value for baudrate generation = 256 - (11059200)/(32*12*baudrate)
29    MOV TL1,#253
30    SETB TR1     ;      //Turn ON the timer for Baud rate generation
31
32    MOV DPTR,#WELCOME_MSG
33    CALL PRINT_MSG
34
35 LOOP:
36    LCALL RECEIVE_BYTE
37    LCALL SEND_BYTE
38
39    JMP LOOP
```

The baud rate is adjusted by setting Timer1 in mode 2 (8-bit auto-reload) (line 27).

This is the formula for calculating the value to be put in TH1

$$TH1 = 256 - \frac{crystal\ frequency}{12 \times 32 \times baud\ rate}$$

To achieve standard baud rate with high precision, a special value crystal is used → crystal frequency of 11.0592Mhz.

For a baud rate of 9600 bit/sec

$$TH1 = 256 - \frac{11059200}{12 \times 32 \times 9600} = 253 = 0FDH$$

As in line 28, we put 253 in TH1

The serial controller has more than one mode to be programmed for. In our example, we will program it in "Asynchronous mode" with 8 bits data, and 1 stop bit.

The register associated with serial controller is called "SCON"

For this mode we put a value of 50H in it as in line 26

We use three functions in our example.

Print_msg → print a message of multiple characters to the serial port (TX)

Receive_byte→ read byte from the serial port (RX)

Send_Byte → send/print a byte to the serial port (TX)

At start, the program will print a welcome message (32,33), then it enter an infinite loop that will read character from serial RX, and then print it to TX (36,37)

Function used

1- Receive_Byte

```
59 ;=========================================================
60 RECEIVE_BYTE:
61 WAIT_RX:
62    JNB RI,WAIT_RX       ; Wait till the data is received
63    CLR RI               ; Clear Receive Interrupt Flag for next cycle
64    MOV A,SBUF           ; return the received char
65 RET
66 ;=========================================================
```

RI → indicates that a byte was received by the serial controller, so we test it, and wait for it to become '1' (62)

Then we reset RI to enable reception of another byte (63)

Character received is stored in SBUF, so we put it on ACC (64)

2- Send_Byte

```
53  SEND_BYTE:
54     MOV SBUF,A           ; Load the data to be transmitted
55  WAIT_TX:
56     JNB TI, WAIT_TX      ; Wait till the data is trasmitted
57     CLR TI               ; Clear the Tx flag for next cycle.
58  RET
```

To send a byte over serial TX, we just put this byte on the SBUF→ automatically the serial controller will convert it to serial bits through TX

This function will wait until a character is sent by monitoring 'TI' and wait for TI to become '1' (56)

3- Print_msg

```
41  PRINT_MSG:
42  NEXT_CHAR:
43     CLR A
44     MOVC A,@A+DPTR
45     INC DPTR
46     CJNE A,#'$',PRINT_CHAR
47     RET
48  PRINT_CHAR:
49     CALL SEND_BYTE
50     JMP NEXT_CHAR
51  RET
```

This function will send a message stored within the code as follows

```
WELCOME_MSG:
   DB "SERIAL ECHO", 10,13,'$'
```

So before calling this function, we load the DPTR with the message address "WELCOME_MSG".

Message are ended by the "$", so the function will continuously read message character by character (43-45) and send them to serial using "SEND_BYTE" (49), till it found the "$" (46).