CO326: Industrial Networks
Lab 02B - Serial Port I/O

Introduction

In Lab 02B, you will be doing the remaining 2 parts of Lab 02. Here, you will be involved in developing an interface circuit using a PIC microcontroller that communicates serially with the field devices and the computer.

Part 04: Designing a serial interfacing device with a PIC microcontroller

This part of the lab will focus on creating an interfacing device that communicates via a serial port of a computer or a controller. We use a PIC microcontroller in this interface design where it's internal USART is used to communicate with the serial port of the PC/Controller. The input pins get digital signals from the sensors in the field and output pins give binary numbers to control the actuators in the field.

The lab experiment will be done using "Proteus" with a PIC microcontroller. The microcontroller is to be programmed with "MPLAB X IDE" which is an integrated development environment used for the development of embedded applications on PIC and dsPIC microcontrollers. With Proteus, the compiled program will be loaded onto the microcontroller and the functionality will be tested with a virtual terminal connected to the device.

Required Tools

To complete this part of the Lab you will need "Proteus" which was used in the previous lab and "MPLAB X IDE" along with some other components associated with "MPLAB". The list of required new tools is as follows,

1. MPLAB® X IDE v5.50 (https://www.microchip.com/content/dam/mchp/documents/DEV/ProductDocuments/SoftwareTools/MPLABX-v5.50-windows-installer.exe)
    - https://www.microchip.com/en-us/development-tools-tools-and-software/mplab-x-ide#tabs
2. MPLAB® XC8 Compiler v2.10 (https://www.microchip.com/mplab/compilers)
    - https://www.microchip.com/mplabxc8windows
3. Microchip Libraries for Applications - MLA (https://www.microchip.com/mplab/microchip-libraries-for-applications)
    - http://ww1.microchip.com/downloads/en/softwarelibrary/mla_v2017_03_06_windows_installer.exe
4. Code Configurator V3.0 on MPLAB X IDE

    Steps to set up

    i. Install and run MPLAB IDE
    ii. Tools -> Plugins -> Available Plugins

iii. Select "MPLAB @ Code Configurator" from the available list and click install.

Steps to Complete the Lab

1. As the first step, you have to create a firmware project in "Proteus" with PIC18F26K20 (choose Create Firmware Project in the last step of New Project Wizard and choose PIC18F26K20 as the controller) and implement the following setup using the PIC18F26K20 microcontroller, 3 LEDs of different colours and a virtual terminal. Note how RX, TX in a microcontroller are connected to RX, TX in a virtual terminal.

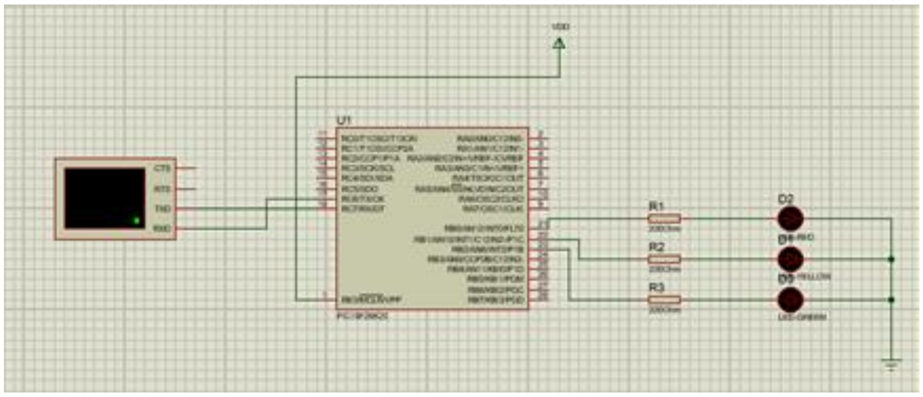


Figure 1: Screenshot of "Proteus" setup to be implemented

2. Now, we need to program the PIC18F26K20 microcontroller with "MPLAB X IDE" so that it can serially communicate with an external device (here, the virtual terminal). The steps you need to follow in order to generate a program code that can do this (using MPLAB) are as follows.

    a. Create a new project in "MPLAB"

        i. File -> New Project

        ii. Select "Standalone project" under "MicroChip Embedded" and click Next

        iii. Now, select the device "PIC18F26K20" from the list and click Next

        iv. From the "Compiler" window you have to select the "XC8" compiler which we already have installed

        v. Give a project name and a location and click "Finish" to create the project.

        vi. Created project will have the folder structure as in Figure 2.

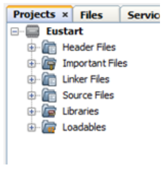


Figure 2: Screenshot the folder structure created for project

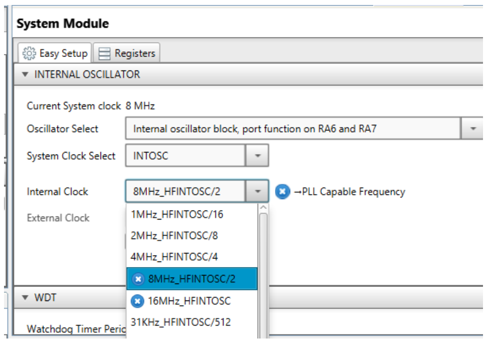
    b. Once the project is created, we'll be using "Code Configurator" in MPLAB to generate the source code needed to build our serial interface. You can start "Code Configurator" and change the configurations using the steps given below.

i. Start the “Code Configurator” as the first step.
   - Tools -> Embedded -> MPL @ Code Configurator V2.0

ii. Then it will open a window asking to save the configuration file (e.g. MyConig.mc3). Save it.

   (NOTE: If you find difficulty in identifying the components of the MCC interface, refer to the help documentation of the MCC.)

iii. Now you will see the MCC user interface within the “MPLAB IDE”. You have to change the following configurations as the initial step.
   - System Clock Settings
     - In the “Composer Area”, under the “Easy steps” you will find the clock settings. You have to change the settings as follows
       i. System Clock Settings: INTOSC
       ii. Internal Clock : 8MHz

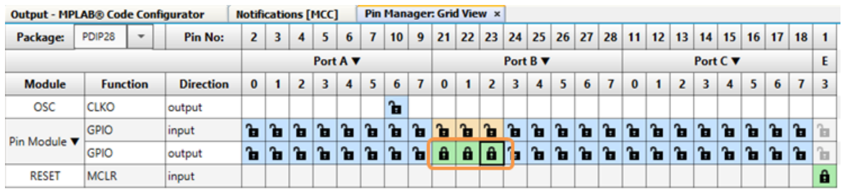


Current System clock 8 MHz
Oscillator Select: Internal oscillator block, port function on RA6 and RA7
System Clock Select: INTOSC
Internal Clock: 8MHz_HFINTOSC/2 →PLL Capable Frequency

Figure 3: Screenshot of the System clock settings

   - Register settings
     - Under the “Registers” section in the “Composer Area” do the following changes in respective registers.
       i. CONFIG1H
          - FCEMN: Disable the fail safe clock monitor
          - FOSC: Internal oscillator block, port function on RA6, RA7
          - IESO : Disable the Oscillator Switchover mode
       ii. CONFIG3H
          - PBADEN: Set portB pins to digital on reset

iv. As the next step you have to configure the input output pins which are to connect the LEDs. Use “Pin Manager Grid View Area”. Then follow the steps given below
   - Use the package as “PDIP28”.
   - As you are going to connect 3 LEDs, you will need 3 pins. Click on 3 pins to use them. As they are used as output pins, you have to click on the output part. (e.g. pin 0,1,2 in portB . These 3 ports should be the ports that you used in the “Proteus” design to connect the 3 LEDs)

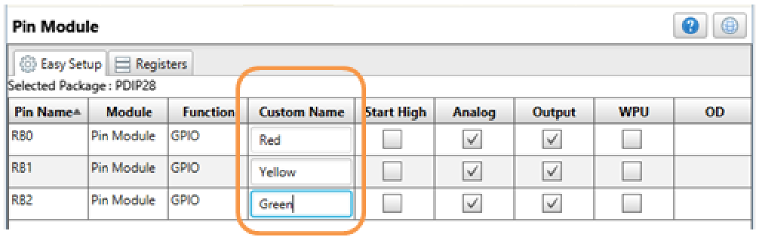


Output - MPLAB® Code Configurator | Notifications [MCC] | Pin Manager: Grid View ×

Package:	PDIP28	Pin No:	2	3	4	5	6	7	10	9	21	22	23	24	25	26	27	28	11	12	13	14	15	16	17	18	1
			Port A ▼								Port B ▼								Port C ▼								E
Module	Function	Direction	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	3
OSC	CLKO	output							🔓																		
Pin Module ▼	GPIO	input	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓
	GPIO	output	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔒	🔒	🔒	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓
RESET	MCLR	input																									🔒

Figure 4: Screenshot of the "Pin Manager Grid View Area"

- Go to the "Pin Module" in "Composer Area". Now you will see the selected ports in a table. You can change the visible names of the pins in the "custom names" column in the table (e.g. Red, Yellow, Green)



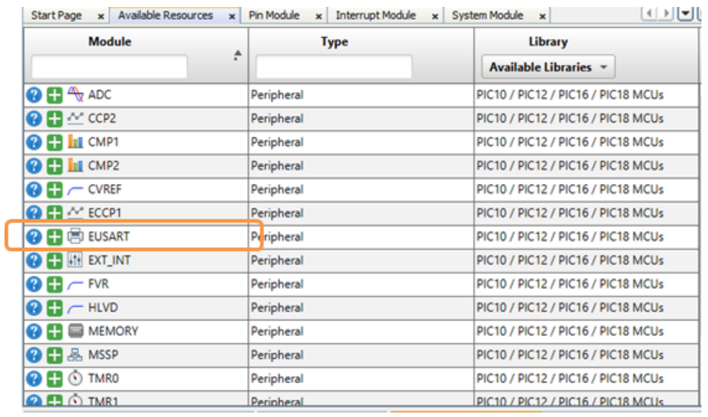
Pin Module

Selected Package : PDIP28

Pin Name▲	Module	Function	Custom Name	Start High	Analog	Output	WPU	OD
RB0	Pin Module	GPIO	Red	☐	✓	✓	☐	
RB1	Pin Module	GPIO	Yellow	☐	✓	✓	☐	
RB2	Pin Module	GPIO	Green	☐	✓	✓	☐	

Figure 5: Screenshot of the "Pin Module"

v. As the next step you have to configure the "EUSART" module.
   - Go to the "Available Resources" section in the "Composer Area".
   - You will find "EUSART" in the Module column. Click on the add (+) button to add the module to your configuration.

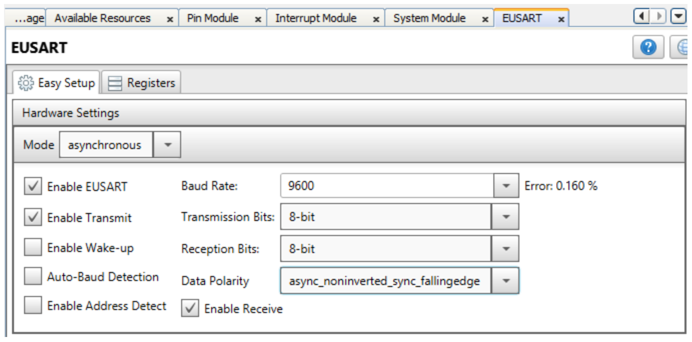


Module	Type	Library
		Available Libraries
ADC	Peripheral	PIC10 / PIC12 / PIC16 / PIC18 MCUs
CCP2	Peripheral	PIC10 / PIC12 / PIC16 / PIC18 MCUs
CMP1	Peripheral	PIC10 / PIC12 / PIC16 / PIC18 MCUs
CMP2	Peripheral	PIC10 / PIC12 / PIC16 / PIC18 MCUs
CVREF	Peripheral	PIC10 / PIC12 / PIC16 / PIC18 MCUs
ECCP1	Peripheral	PIC10 / PIC12 / PIC16 / PIC18 MCUs
EUSART	Peripheral	PIC10 / PIC12 / PIC16 / PIC18 MCUs
EXT_INT	Peripheral	PIC10 / PIC12 / PIC16 / PIC18 MCUs
FVR	Peripheral	PIC10 / PIC12 / PIC16 / PIC18 MCUs
HLVD	Peripheral	PIC10 / PIC12 / PIC16 / PIC18 MCUs
MEMORY	Peripheral	PIC10 / PIC12 / PIC16 / PIC18 MCUs
MSSP	Peripheral	PIC10 / PIC12 / PIC16 / PIC18 MCUs
TMR0	Peripheral	PIC10 / PIC12 / PIC16 / PIC18 MCUs
TMR1	Peripheral	PIC10 / PIC12 / PIC16 / PIC18 MCUs

Figure 6: Screenshot of the "Available Resources"

   - In the "EUSART" section appearing in the "Composer Area", change the following settings
     - Mode: Asynchronous
     - Baud Rate:9600
     - Transmission Bits: 8
     - Reception Bits: 8
     - Data Polarity: Non inverted

- Tick the following options
    i. Enable EUSART
    ii. Enable Transmit
    iii. Enable Receive



EUSART

Hardware Settings

Mode: asynchronous

- [✓] Enable EUSART
- [✓] Enable Transmit
- [ ] Enable Wake-up
- [ ] Auto-Baud Detection
- [ ] Enable Address Detect
- [✓] Enable Receive

Baud Rate: 9600 Error: 0.160 %

Transmission Bits: 8-bit

Reception Bits: 8-bit

Data Polarity: async_noninverted_sync_fallingedge

Figure 7: Screenshot of the "EUSART" settings

vi. As the final step of the code configurator you have to generate the code.

- Go to the "Resource Management MCC" tab in the "Resource Management Area" of the MCC interface and click on the "Generate" button.

- Then the changes you have done so far to the configurations will be saved to the configuration file which was created initially (e.g. MyConig.mc3).

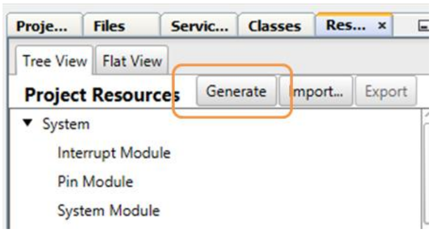


Figure 8: Screenshot of the "EUSART" settings

- Now you can close the MCC by following the same path you opened the MCC.

c. The next part of the project is that you have to code the three pins such that the LEDs are lighting on user input

i. The coding of the pins are done in the "Main.c" file. Open the "Main.c" file in the Editor section. You will find it in, Project -> Source files -> Main.c

ii. "MPLab" has already created the template for the main file. What you have to do is, insert the code between the created template such that the LEDs you used are lighted on the user inputs. Follow the instructions below.

You need to send a message through the EUSART module whenever the microcontroller is switched on.

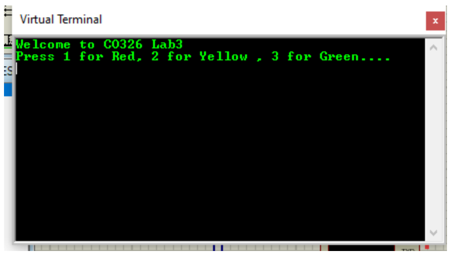
- First create a function called "Send_String".

```
void send_string(const char *x)
{
  while (*x)
  {
        EUSART_Write(*x++);
      }
    }
```

NOTE: EUSART_Write(unit8_t Data) is an inbuilt function which is used to write a single byte of data to EUSART module.

- Then you need to call the function you created to display the message. The message you need to display should be as follows. The second message should display after a 500ms delay to the first one.

  Welcome to CO326 Lab3
  Press 1 for Red, 2 for Yellow, 3 for Green ...



```
Welcome to CO326 Lab3
Press 1 for Red, 2 for Yellow , 3 for Green....
```

Figure 9: Screenshot of the virtual terminal in "Proteus" showing how messages should be displayed.

- As the second step you need to write the code to light up the bulbs on user input. In the "Main.c" template you will find a "while(1){}" code where your application code can go inside. You should code such that 3 numbers light up 3 LEDs respectively and any other key would light off all LEDs. (e.g.: 1 for Red, 2 for Yellow, 3 for Green, any other key - non will light up)

  Hints you may need to use in the coding are as follows.

  i. Use a switch case

  ii. To read the user input, you may use the inbuilt function EUSART_Read();,
     - E.g.: data=EUSART_Read();
       You need to create a character variable (e.g.: data) before the main method.

  iii. For lighting up, you can set the pins up using the inbuilt function (these are called "Pin Macros"), "SetHigh" and to light off, use "SetLow"
     - E.g: Red_SetHigh();
     - E.g: Green_SetLow();

  NOTE: The inbuilt functions or pin micros you need with regard to pins can be found at, Header Files -> MCC Generated Files -> pin_manager.h
  The inbuilt functions for EUSART module can be found at, Header Files -> MCC Generated Files -> eusart.h

d. After your coding is completed. Now you can compile the project. You need to clean and build the main project.

Production -> Clean and Build Main Project

NOTE: If you find any compilation problem, try changing the C standard from 99 to 90 in the compiler and the linker.

- Right click on the project -> properties -> XC8 Global Options
- Change C standard from c99 to c90 in CX8 Compiler, CX8 Linker.

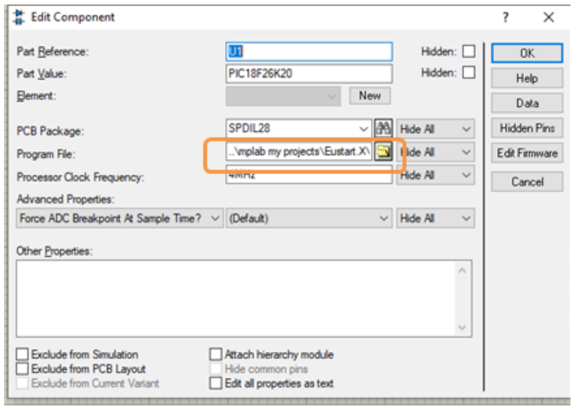
NOTE: You can refer to the youtube video in the following link for further clarifications.

- https://youtu.be/jLRTTHI9m94

3. Now you generated the program code (hex code) for the microcontroller. You have to use this code to program the microcontroller in the "Proteus".

a. In "Proteus" switch to "Component Mode" and double click on the PIC18F26K20 device.

b. Give the path of the hex file to the "Program File" in the "Edit Component" window.



Edit Component

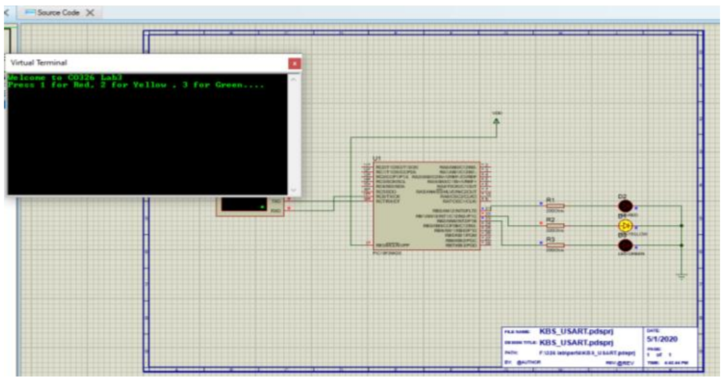
Part Reference: U1
Part Value: PIC18F26K20
Element:
PCB Package: SPDIL28
Program File: ..\mplab my projects\Eustart.X\
Processor Clock Frequency: 4MHz
Advanced Properties: Force ADC Breakpoint At Sample Time? (Default)
Other Properties:

Figure 10: Screenshot of the "Edit Component" window of PIC18F26K20

NOTE: You can find the hex file at the following path from the directory you saved your MPLAB project

..\<ProjectName>\dist\default\production\<ProjectName>.X.production.hex

c. Then you have to run the simulation and test the component by entering user inputs at the virtual terminal.



```
Welcome to CO326 Lab3
Press 1 for Red, 2 for Yellow , 3 for Green....
```

Figure 11: Screenshot of the "Proteus" with working setup

Submission

Take screenshots of

I. The proteus setup
II. The proteus setup all stages LEDs (lighting, not lighting)

Project files

I. Proteus File (.pdsprj)
II. Main.c file of the "MPLAB" project
III. .hex file generated from "MPLAB IDE".

Create a folder named "Lab02_Part4" and put the above screenshots and the project files in it and submit it along with the other submissions for Lab02B.

Part 05: Communicating with the serial device designed in "Proteus" using computer

In this part of the lab you will connect a serial port to the "Proteus" setup and that port will be connected to another serial port on the computer. Then with the virtual terminal of "Proteus" and a "Tera Term" terminal opened at the other serial port, you have to transmit data to light up the LEDs. (You will be using the same virtual ports and the virtual connection established in Part 01 for this part as well.)

Steps to Complete the Lab

1. As the initial step you have to implement the following setup in "Proteus".

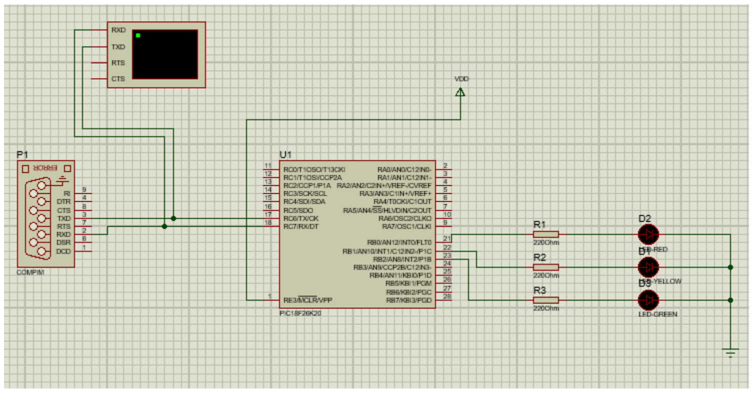


Figure 12: Screenshot of the "Proteus" setup to be implemented

2. Now you have to follow the steps done at part 03 and modify part 04 to create a communication between two serial ports to communicate with the microcontroller.

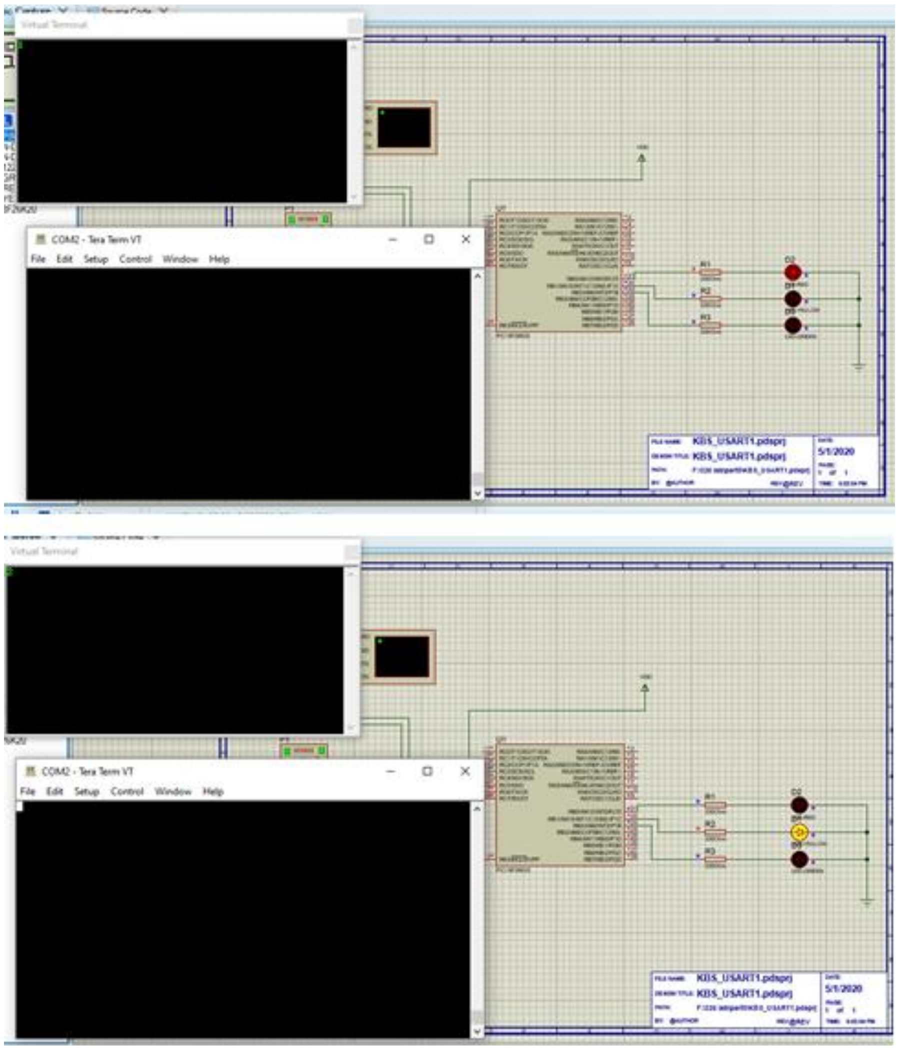


Figure 13: Screenshot showing the serial port communication in virtual terminal of "Proteus" and "Tera Term" terminal to light up LEDs

Submission

Take screenshots of

I. The proteus setup
II. Serial data (text) transfer between two virtual ports performed with Tera Term terminal and virtual terminal in Proteus for all stages of LEDs.

Project files

III. Proteus File

Create a folder named "Lab02_Part5" and put the above screenshots and project files in it and submit it along with the other submissions for Lab02B.