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

Introduction

The serial port, which is the industry standard with RS-232 (and RS485) is a serial communication physical interface where information is transferred serially; one bit at a time in order to communicate with the external devices/peripherals connected to the computers or controllers. In industry, serial ports (RS232 and RS485) are used extensively to connect field devices to a controller. The controller could be a Programmable Logic Controller (PLC) or an industrial PC.

Figure 01: RS232 Serial Cable

The aims of the lab are,

1. To understand the way serial interfaces are connected together to communicate between two devices.
2. To develop an interface circuit that communicates with the field devices and the computer.

The lab will be done in 05 parts.
During the online lab session, we do not have access to computers with serial ports. Therefore in this Lab01 virtual serial ports are created. You can assume that each virtual serial port mimics a physical serial port in a computer.

Part 01: Establishing and testing a virtual RS232 serial port connection with the help of third-party tools

In this part, you will create two virtual serial ports on the same computer and establish a connection between the two with a certain wiring configuration. Finally, you will test and monitor the connection using a terminal emulator and a serial port monitoring tool respectively.

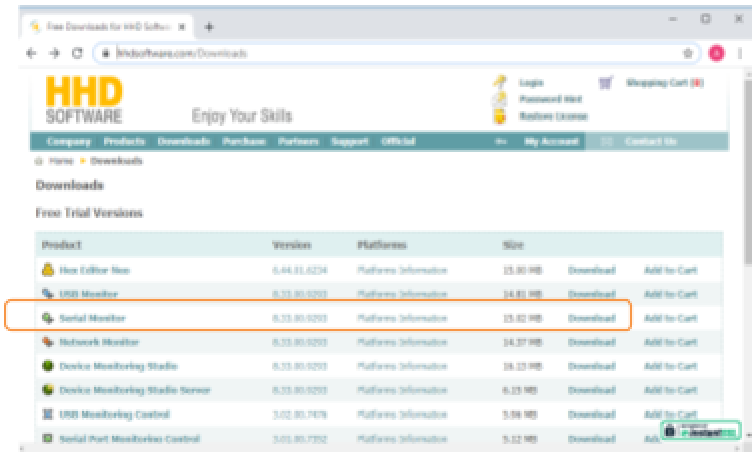
Required Tools

You will be completing this part of the lab using a few small-sized third-party tools. Therefore, you are required to download and set-up these tools before you start doing the lab. The list of tools required with a link to download each tool is given below.

1. Virtual Serial Ports (14 day trial)
   - https://www.hhdsoftware.com/Download/virtual-serial-port-tools.exe

2. Tera Term
   - https://osdn.net/projects/ttssh2/downloads/72009/teraterm-4.105.exe/

3. HHD Free Serial Port Monitor
   - https://www.hhdsoftware.com/Downloads



Downloads

Free Trial Versions

Product	Version	Platforms	Size		
Hex Editor Neo	[illegible]	Platforms Information	[illegible]	Download	Add to Cart
USB Monitor	[illegible]	Platforms Information	[illegible]	Download	Add to Cart
Serial Monitor	[illegible]	Platforms Information	[illegible]	Download	Add to Cart
Network Monitor	[illegible]	Platforms Information	[illegible]	Download	Add to Cart
Device Monitoring Studio	[illegible]	Platforms Information	[illegible]	Download	Add to Cart
Device Monitoring Studio Server	[illegible]	Platforms Information	[illegible]	Download	Add to Cart
USB Monitoring Control	[illegible]	Platforms Information	[illegible]	Download	Add to Cart
Serial Port Monitoring Control	[illegible]	Platforms Information	[illegible]	Download	[illegible]

Figure 02: Screenshot of HHD free serial port monitor downloads page

Steps to Complete the Lab

1. First, you need to create two virtual serial ports using "Virtual Serial Ports" software and establish a connection between them.

   Here, we will be creating both the ports on the same computer and establishing a local connection. Therefore, you need to choose the "Create Local Bridge" option (Figure 02).

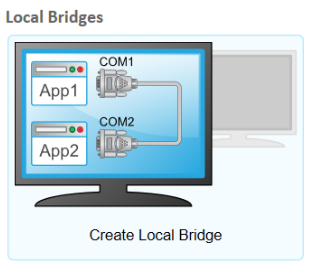


Figure 03: Screenshot of "Create Local Bridge" option

On the "Create Local Bridge" window, choose two COM port names (COM1, COM2, etc.) for the two virtual ports to be created.

Next, you can click on "Options..." button and specify the wiring configuration between the two ports being created. You can choose a pre-configured configuration such as full, loopback, etc. from the list under "Pinout scheme" (Figure 03). Or, you can create your own custom configuration by clicking on pins (remember that you should only create a connection that is supported by other hardware/software associated with the connection). Finally, click on "OK" and then "Create" to create the virtual serial port connection. On Windows, you will be able to see the new virtual serial ports getting listed in “Device Manager” under “Ports (COM & LPT)”.

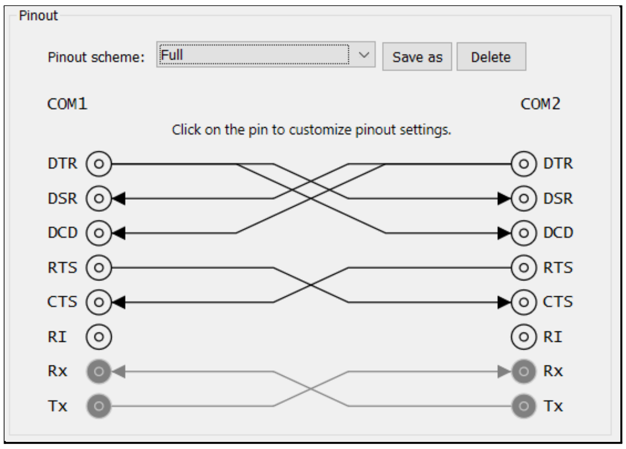
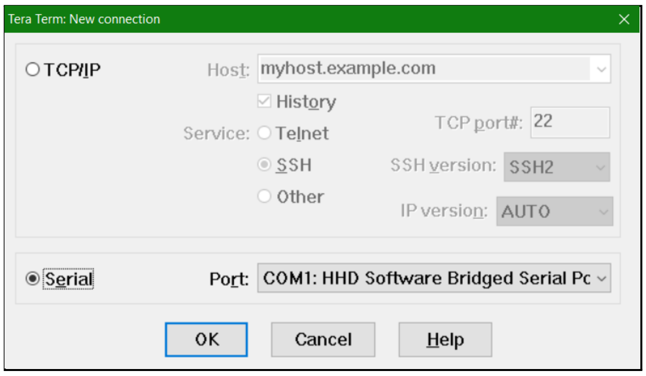


Figure 04: Screenshot of “Pinout” configuration

2. Now, you should test the connection by sending some data from one port and checking whether you receive the same at the other port. This can be done by the terminal emulator software - “Tera Term”.

   Open “Tera Term” and choose “Serial” from the starting window. Also, choose one of the two serial ports you have created above.



Tera Term: New connection

○ TCP/IP — Host: myhost.example.com
☑ History
Service: ○ Telnet — TCP port#: 22
● SSH — SSH version: SSH2
○ Other — IP version: AUTO

● Serial — Port: COM1: HHD Software Bridged Serial Pc

Figure 05: Screenshot of “Tera Term” starting window configuration

Now, you can send character data through the port you selected by typing letters on the terminal window you get. However, in order to see what is being received at the other port, you need to create another connection in Tera Term for the second port. You can do so by choosing File->New connection… and selecting the second virtual you have created.

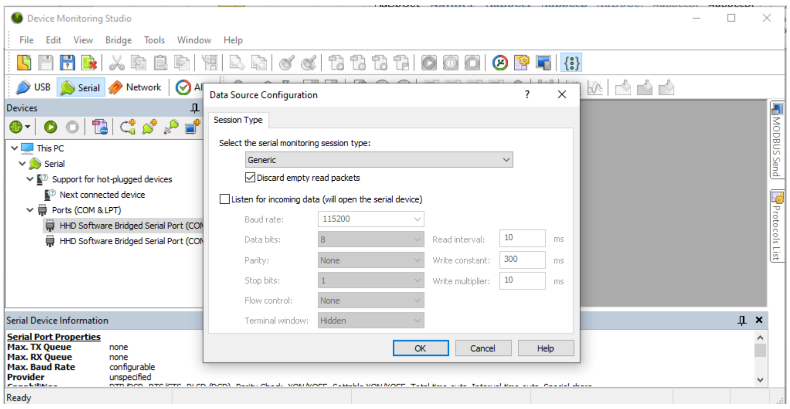
You now have a full setup to test the connection. You can type something on the terminal created for the first port and check whether what you typed is being received at the second terminal.

You can try entering a new line on the first terminal and see whether that is received properly at the second terminal. If not, you should change settings on the terminal windows to get the desired behavior (Hint: see what Carriage Return, Line Feed are).

3. Further, you can see what is happening underlying when you communicate between two serial ports by monitoring the serial port connections on your computer. You can do this using “HHD Free Serial Port Monitor” software.

    a. Open Device Monitoring Studio. (NOTE: As you have downloaded the serial port monitor, the software has the modules related to the serial port monitoring only. If you want USB and other network monitoring, the respective modules can be downloaded later)

    b. In the “Serial” section, select one port you’ve created (e.g. COM1) by double clicking on the port and select the monitoring session as “Generic” for a simple monitoring session and click OK.


Data Source Configuration

Session Type

Select the serial monitoring session type: Generic

☑ Discard empty read packets

☐ Listen for incoming data (will open the serial device)

Baud rate:	115200			
Data bits:	8	Read interval:	10	ms
Parity:	None	Write constant:	300	ms
Stop bits:	1	Write multiplier:	10	ms
Flow control:	None			
Terminal window:	Hidden			

Serial Device Information

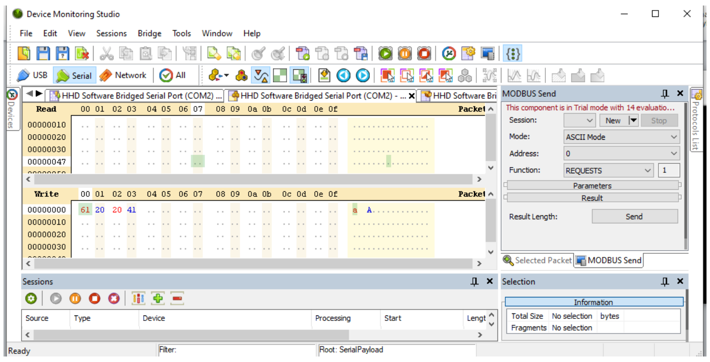
Serial Port Properties

Max. TX Queue	none
Max. RX Queue	none
Max. Baud Rate	configurable
Provider	unspecified

Figure 06: Screenshot of “Device Monitoring Studio” starting monitoring session

    c. Then select the monitoring views from the next window. There are a number of views which can be selected. Select “Raw data view” to see the data entering, “packet view“ to see the packet flow etc.

    d. Try entering data at the “Tera Term” terminal and observe the different views at the monitoring tool to identify the behavior of the data flow in the serial port communication.


HHD Software Bridged Serial Port (COM2)

MODBUS Send

This component is in Trial mode with 14 evaluatio...

Session:	
Mode:	ASCII Mode
Address:	0
Function:	REQUESTS 1

```
Write   00 01 02 03  04 05 06 07  08 09 0a 0b  0c 0d 0e 0f
00000000 61 20 20 41
```

Figure 07: Screenshot of “Device Monitoring Studio” Raw data view

Submission

Take screenshots of

I. Serial data (text) transfer between two virtual ports with two Tera Term terminals.
II. Output of the monitoring tool when the above data transfering occurs

---

Part 02: Using "Proteus" to simulate one serial port

In this part, you will be using "Proteus" software to simulate one serial port connected to a virtual terminal. You will be using the same virtual ports and the virtual connection established in Part 01 for this part as well. The connection will be tested using a "Tera Term" terminal connected to one serial port and the virtual terminal in Proteus.

Steps to Complete the Lab

1. As the initial step you have to implement the following setup in "Proteus". Use a COMPIM to model a serial port and connect a virtual terminal to that serial port.

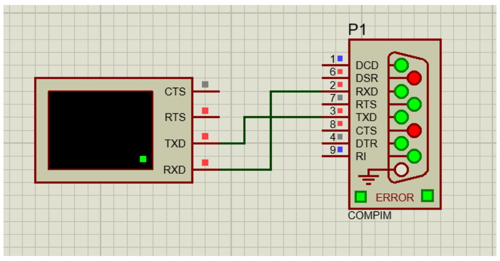
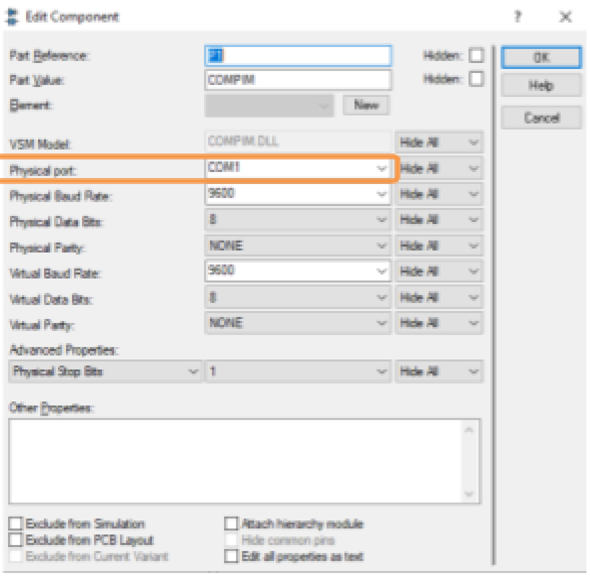


Figure 08: Screenshot of "Proteus" setup to be implemented.

2. Next, you have to make this serial port as one of the two virtual serial ports (e.g. COM1) you have created before. For that, using the component mode, edit the component COMPIM by double clicking on it. In the opening "Edit component" window you have to assign the physical port as one of the two ports you've created (e.g. COM1).

   When this setup is completed, you can run the simulation from debug -> run simulation. Once it is running, the virtual terminal corresponding to the serial port being simulated in Proteus will be opened.



Part Reference: P1 Hidden: ☐
Part Value: COMPIM Hidden: ☐
Element:
VSM Model: COMPIM.DLL Hide All
Physical port: COM1 Hide All
Physical Baud Rate: 9600 Hide All
Physical Data Bits: 8 Hide All
Physical Parity: NONE Hide All
Virtual Baud Rate: 9600 Hide All
Virtual Data Bits: 8 Hide All
Virtual Parity: NONE Hide All
Advanced Properties:
Physical Stop Bits 1 Hide All
Other Properties:

☐ Exclude from Simulation
☐ Exclude from PCB Layout
☐ Exclude from Current Variant
☐ Attach hierarchy module
☐ Hide common pins
☐ Edit all properties as text

Figure 09: Screenshot of "Edit Component" window of "Proteus"

3. Now, open a "Tera Term" terminal connected to your other virtual serial port (e.g. COM2).
4. You can test the connection between the two ports by sending some character data from the "Tera Term" terminal to the virtual terminal in Proteus and vice versa.

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

---

Part 03: Using "Proteus" to simulate both the serial ports

In this part, you have to make use of "Proteus" to simulate both the serial ports. You will be using the same virtual ports and the virtual connection established in Part 01 for this part as well. The connection will be tested using virtual terminals connected to each serial port being simulated in Proteus.

Steps to Complete the Lab

1. Using "Proteus" to create the following setup with two COMPIM devices as Serial ports and two virtual terminals connected to each of those serial ports.

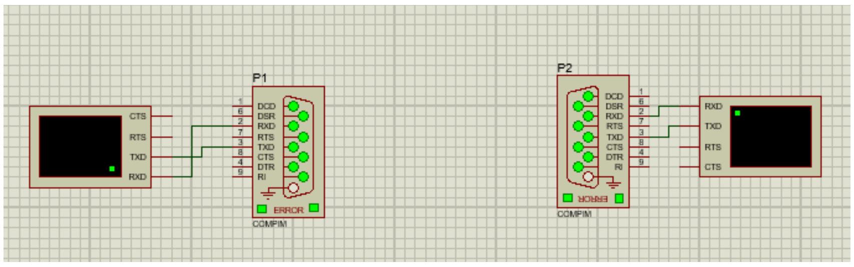


Figure 10: Screenshot of "Proteus" setup to be implemented

2. Then, assign one virtual serial port (e.g. COM1) to one serial port in Proteus and the other one (e.g. COM2) to the other port. Finally, run the simulation.
3. Now, you will see two virtual terminal windows being opened. You can use those two terminals to test the connection between the two ports by sending character data from one terminal and observing receiving data at the other terminal and vice versa.

Submission

Take screenshots of

I. The proteus setup
II. Serial data (text) transfer between two virtual ports performed with two virtual terminals in Proteus

---

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" 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.

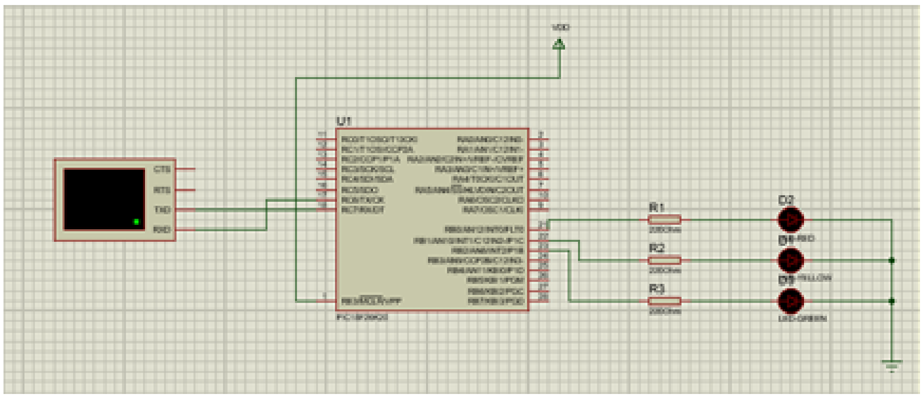


Figure 1: Screenshot of "Proteus" setup to be implemented

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.

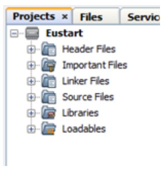


Figure 2: Screenshot the folder structure created for project

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.

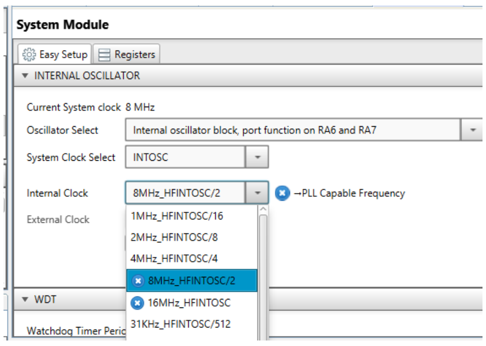
- Tools -> Embedded -> MPL @ Code Configurator V2.0

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
External Clock
Watchdog Timer Peric

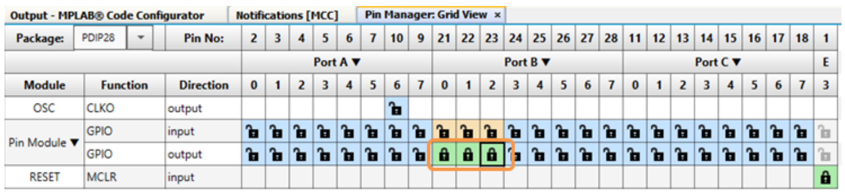
Figure 3: Screenshot of the System clock settings

- 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)

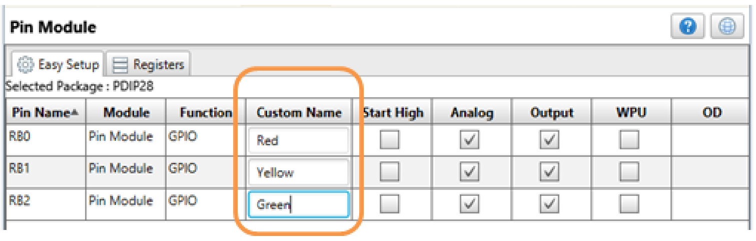


Output - MPLAB® Code Configurator | Notifications [MCC] | Pin Manager: Grid View ×

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

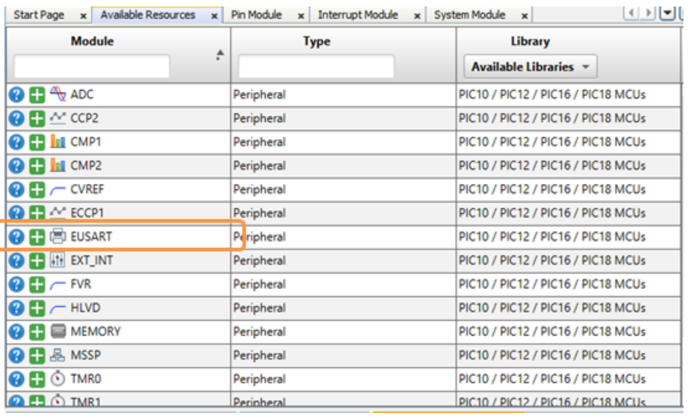
Easy Setup | Registers

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”

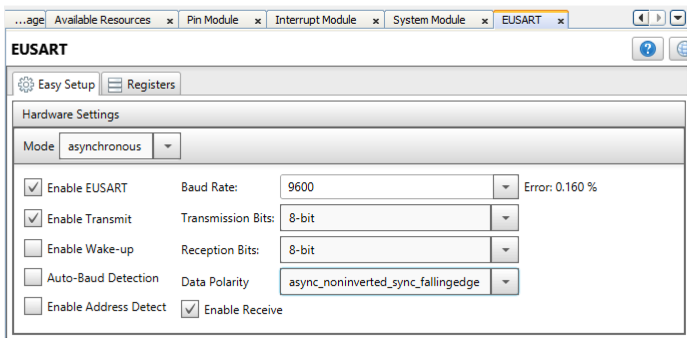
v. As the next step you have to configure the “EUSART” module.
   - Go to the “Available Resources” section in the “Composer Area”.
   - You will find “EUSART” in the Module column. Click on the add (+) button to add the module to your configuration.



Module	Type	Library
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
    1. Enable EUSART
    2. Enable Transmit
    3. Enable Receive



Mode: asynchronous

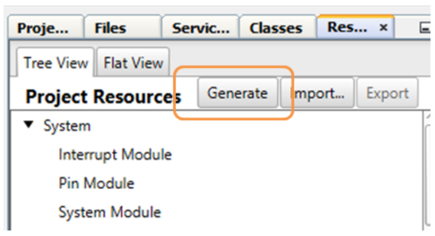
✓ Enable EUSART	Baud Rate: 9600	Error: 0.160 %
✓ Enable Transmit	Transmission Bits: 8-bit	
Enable Wake-up	Reception Bits: 8-bit	
Auto-Baud Detection	Data Polarity: async_noninverted_sync_fallingedge	
Enable Address Detect	✓ Enable Receive	

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.

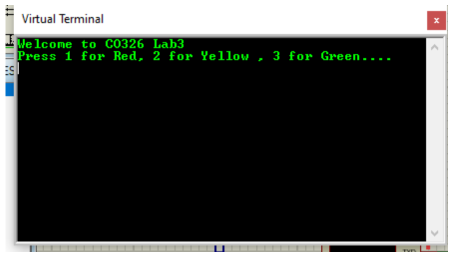
- First create a function called "Send_String".

```
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. 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.

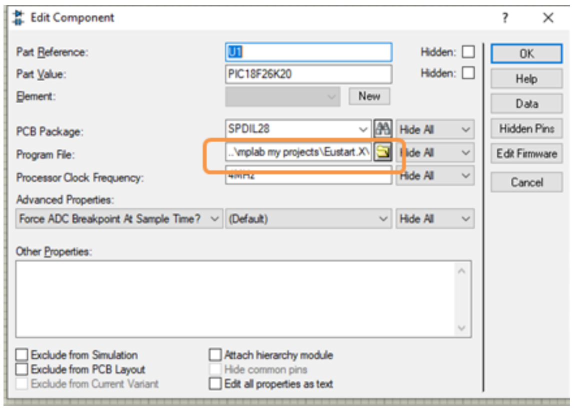
NOTE: You can refer to the youtube video in the following link for further clarifications.

- 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”



Edit Component

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

☐ Exclude from Simulation
☐ Exclude from PCB Layout
☐ Exclude from Current Variant
☐ Attach hierarchy module
☐ Hide common pins
☐ Edit all properties as text

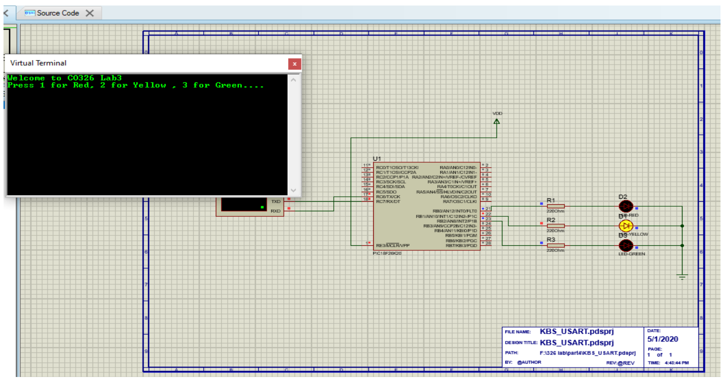
window.

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.



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".

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".

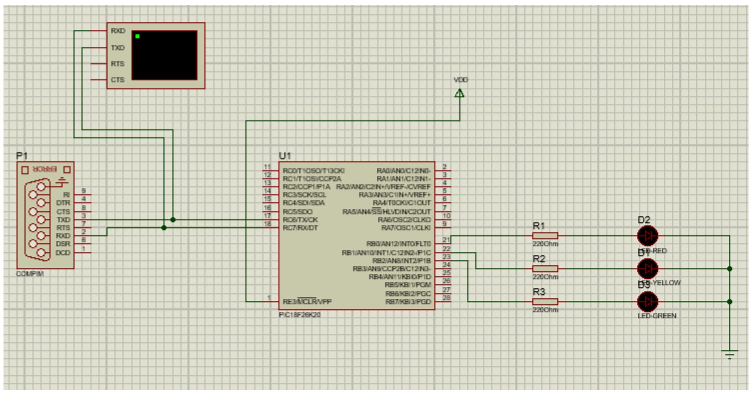


Figure 12: Screenshot of the "Proteus" setup to be implemented

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.

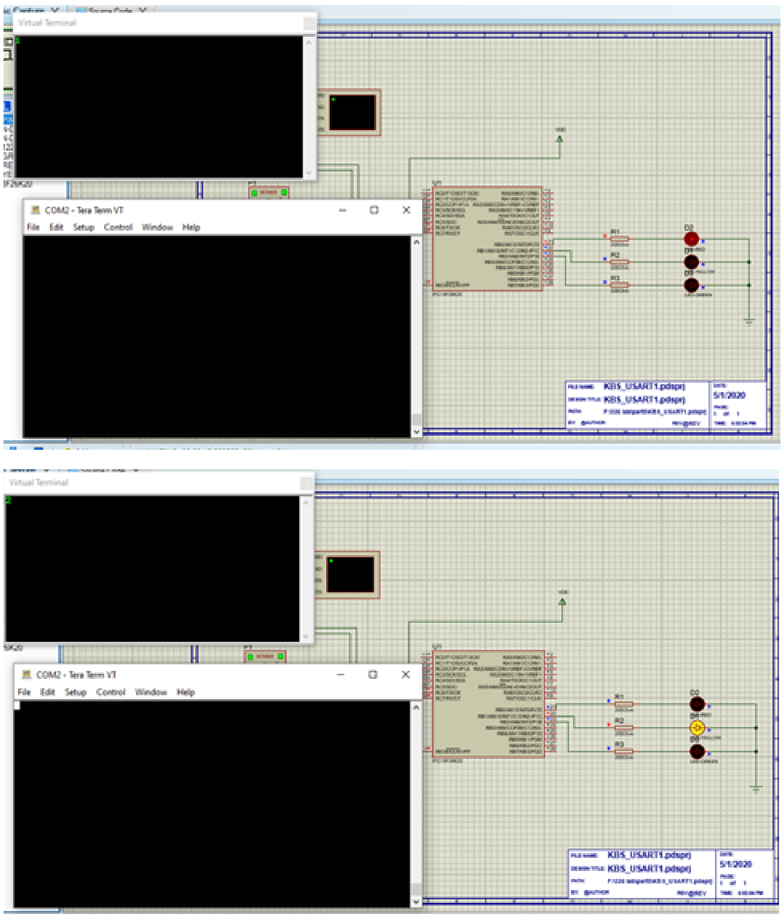


Figure 13: Screenshot showing the serial port communication in virtual terminal of "Proteus" and "Tera Term" terminal to light up LEDs

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 (.pdsprj)

Final Submission

Create a Report named Lab02_E17XXX.pdf where XXX is your E Number, including the screenshots for all the submissions under each part, from Part 01 to 05. Include explanations where necessary.

Submit a Folder named Lab02_E17XXX.zip, including the report and the project files before the deadline. Once submission per group is sufficient.