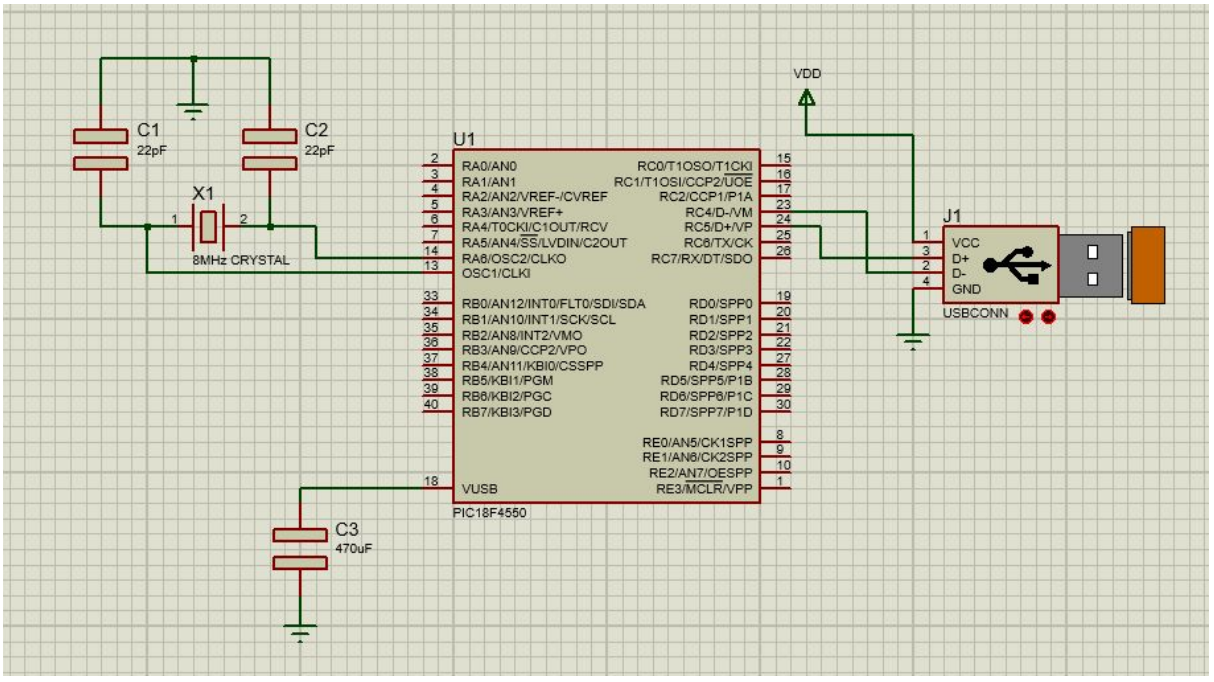
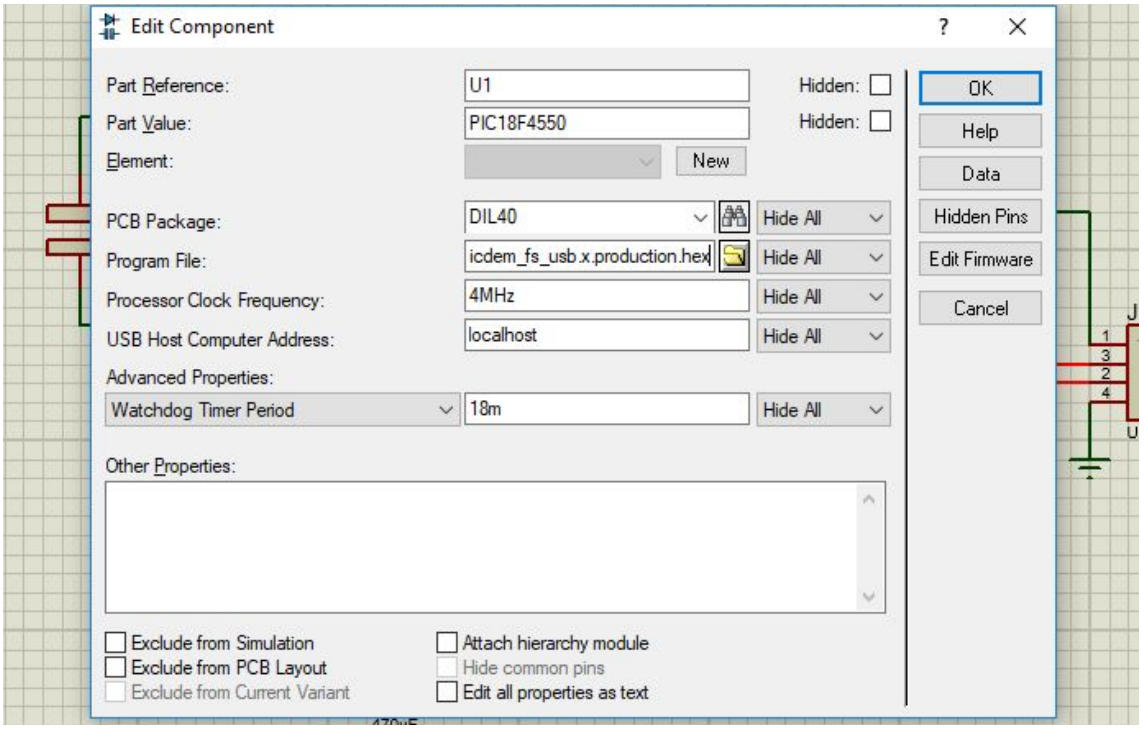
CO326 - Computer Systems Engineering & Industrial Networks

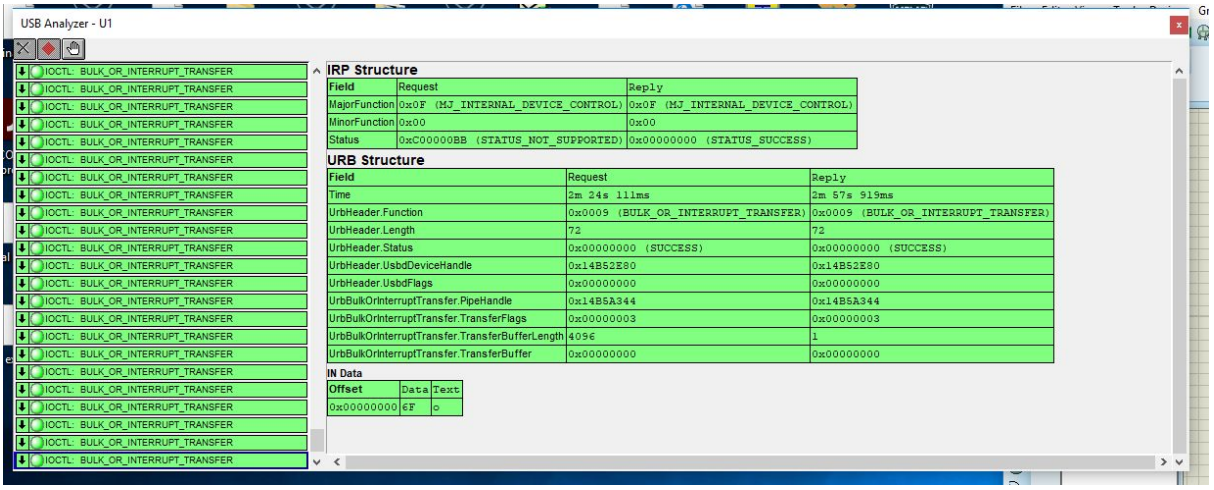
Lab: 04
Topic: USB Port I/O
Name: M.M.M Irfan
Reg no: E/15/138

Proteus Setup





USB Analyser from Proteus



Code from MPLAB for the Lab Task (app_device_cdc_basic.c)

```
/*******************************************************************
********
Copyright 2016 Microchip Technology Inc. (www.microchip.com)

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

To request to license the code under the MLA license
(www.microchip.com/mla_license),
please contact mla_licensing@microchip.com
*******************************************************************
*******/

/** INCLUDES *******************************************************/
#include "system.h"

#include <stdint.h>
#include <string.h>
```

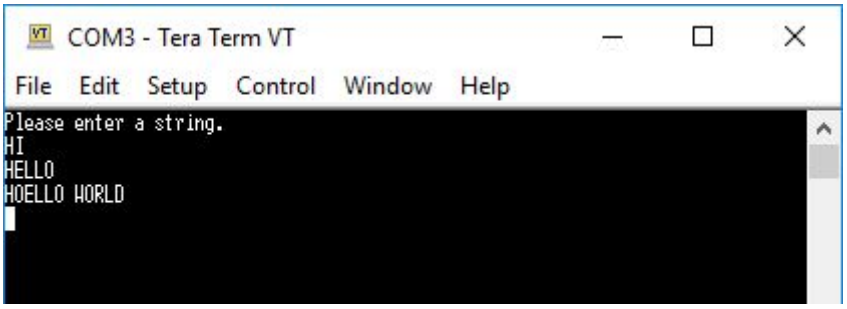
```c
#include <stddef.h>

#include "usb.h"

#include "app_led_usb_status.h"
#include "app_device_cdc_basic.h"
#include "usb_config.h"

/** VARIABLES ******************************************************/

static bool buttonPressed;
static char buttonMessage[] = "Please enter a string.\r\n";
static uint8_t readBuffer[CDC_DATA_OUT_EP_SIZE];
static uint8_t writeBuffer[CDC_DATA_IN_EP_SIZE];
static uint8_t count = 0;

/*********************************************************************
* Function: void APP_DeviceCDCBasicDemoInitialize(void);
*
* Overview: Initializes the demo code
*
* PreCondition: None
*
* Input: None
*
* Output: None
*
*********************************************************************/
void APP_DeviceCDCBasicDemoInitialize()
{
    line_coding.bCharFormat = 0;
    line_coding.bDataBits = 8;
    line_coding.bParityType = 0;
    line_coding.dwDTERate = 9600;

    buttonPressed = false;
}

/*********************************************************************
* Function: void APP_DeviceCDCBasicDemoTasks(void);
*
* Overview: Keeps the demo running.
*
* PreCondition: The demo should have been initialized and started via
*                   the      APP_DeviceCDCBasicDemoInitialize()      and
```

```
APP_DeviceCDCBasicDemoStart() demos
*   respectively.
*
* Input: None
*
* Output: None
*
*******************************************************************/
void APP_DeviceCDCBasicDemoTasks()
{
      /* If the USB device isn't configured yet, we can't really do
anything
     * else since we don't have a host to talk to.  So jump back to the
     * top of the while loop. */
    if( USBGetDeviceState() < CONFIGURED_STATE )
    {
        return;
    }

    /* If we are currently suspended, then we need to see if we need to
     * issue a remote wakeup.  In either case, we shouldn't process any
      * keyboard commands since we aren't currently communicating to the
host
     * thus just continue back to the start of the while loop. */
    if( USBIsDeviceSuspended()== true )
    {
        return;
    }

     /* If the user has pressed the button associated with this demo,
then we
     * are going to send a "Button Pressed" message to the terminal.
     */
    if(BUTTON_IsPressed(BUTTON_DEVICE_CDC_BASIC_DEMO) == true)
    {
         /* Make sure that we only send the message once per button press
and
         * not continuously as the button is held.
         */
        if(buttonPressed == false)
        {
                    /* Make sure that the CDC driver is ready for a
transmission.
             */
            if(mUSBUSARTIsTxTrfReady() == true)
```

```
            {
                putrsUSBUSART(buttonMessage);
                buttonPressed = true;
            }
        }
    }
    else
    {
         /* If the button is released, we can then allow a new message to
be
         * sent the next time the button is pressed.
         */
        buttonPressed = false;
    }

     /* Check to see if there is a transmission in progress, if there
isn't, then
     * we can see about performing an echo response to data received.
     */
    if( USBUSARTIsTxTrfReady() == true)
    {
        uint8_t i;
        uint8_t numBytesRead;

        numBytesRead = getsUSBUSART(readBuffer, sizeof(readBuffer));

        /* For every byte that was read... */
        for(i=0; i<numBytesRead; i++)
        {
            switch(readBuffer[i])
            {
                /*
                 * for newline or \r
                 */
                case 0x0A:
                case 0x0D:
                    writeBuffer[count] = readBuffer[i];
                    // add a newline character at the end
                    writeBuffer[count+1] = '\n';
                    count +=2;
                    // send the write buffer
                    putUSBUSART(writeBuffer,count);
                    count = 0;
                    break;
```

```
                default:
                    if(readBuffer[i] >= 'a' && readBuffer[i] <= 'z'){
                    /* Make the letter to upper case one */
                        writeBuffer[count] = readBuffer[i] + 'A' - 'a';
                        count++;
                    }else{
                    /* these are not lowercase letters so no change */
                        writeBuffer[count] = readBuffer[i];
                        count++;
                    }
                    break;

            }
        }
    }

    CDCTxService();
}
```

Screenshot of Tera Term Terminal after the labtask



Problems and issues you encountered and how you solved them

Initially, when I followed the steps and did a clean build in MPLAB editor it dit not compile successfully and gave the following errors

```
                                                                    ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ ^~~~~
../../../../../../framework/usb/src/usb_device.c:2920:53: warning: implicit conversion from enumeration type 'USB_DEVICE_STACK_EVENTS
                    USB_TRANSFER_TERMINATED_HANDLER(EVENT_TRANSFER_TERMINATED,p,sizeof(p));
                    ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
../../../../../../framework/usb/src/usb_device_local.h:392:113: note: expanded from macro 'USB_TRANSFER_TERMINATED_HANDLER'
    #define USB_TRANSFER_TERMINATED_HANDLER(event,pointer,size)                 USER_USB_CALLBACK_EVENT_HANDLER(event,pointer,size)
                                                                                ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ ^~~~~
5 warnings and 3 errors generated.
(908) exit status = 1
make[2]: *** [nbproject/Makefile-PICDEM_FSUSB.mk:206: build/PICDEM_FSUSB/production/_ext/838585624/usb_device_cdc.p1] Error 1
make[2]: *** Waiting for unfinished jobs....
make[2]: *** [nbproject/Makefile-PICDEM_FSUSB.mk:198: build/PICDEM_FSUSB/production/_ext/838585624/usb_device.p1] Error 1
(908) exit status = 1
make[1]: *** [nbproject/Makefile-PICDEM_FSUSB.mk:91: .build-conf] Error 2
make: *** [nbproject/Makefile-impl.mk:39: .build-impl] Error 2
make[2]: Leaving directory 'C:/microchip/mla/v2017_03_06/apps/usb/device/cdc_basic/firmware/picdem_fs_usb.x'
make[1]: Leaving directory 'C:/microchip/mla/v2017_03_06/apps/usb/device/cdc_basic/firmware/picdem_fs_usb.x'

BUILD FAILED (exit value 2, total time: 7s)
```

It seems like I've not followed the steps, in ***LAB03 part B***, there was an instruction as bellow

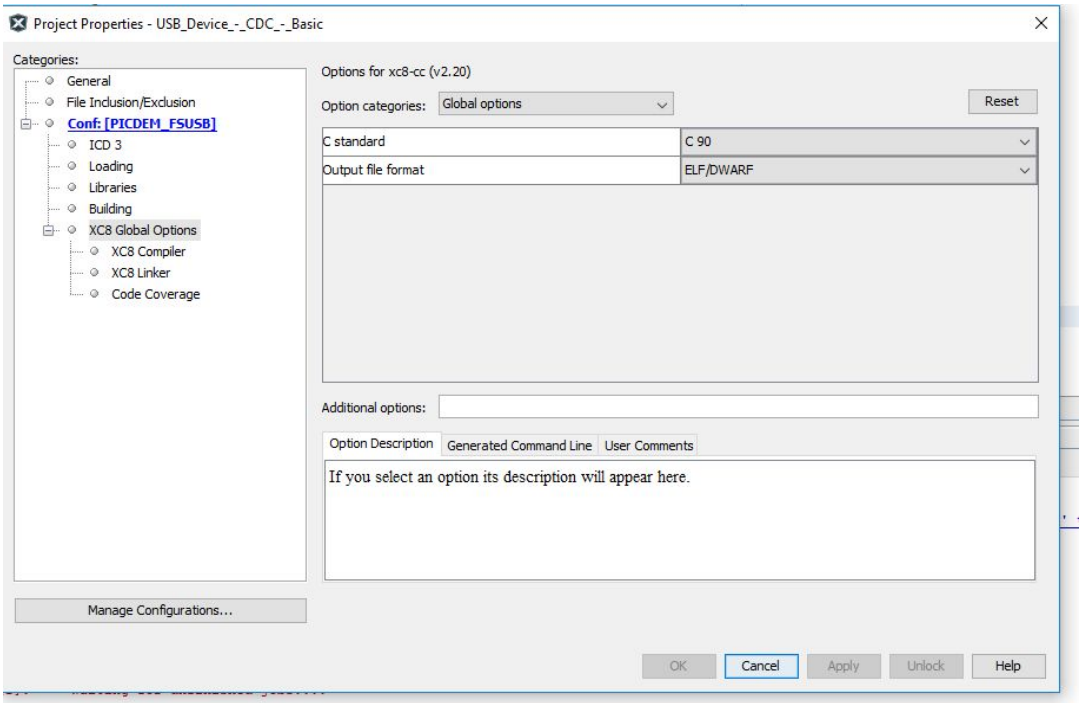
d. After your coding is completed. Now you can compile the project. You need to clean and build the main project.

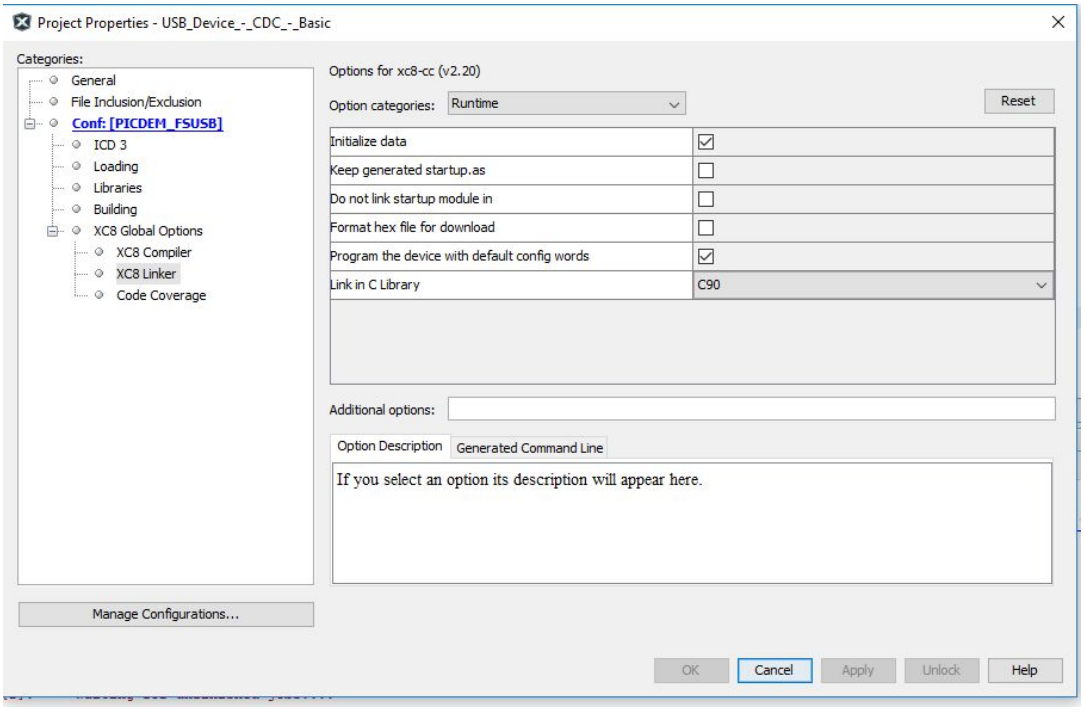
Production -> Clean and Build Main Project

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.

So I changed the C standard to C90 in both Global options and the linker.

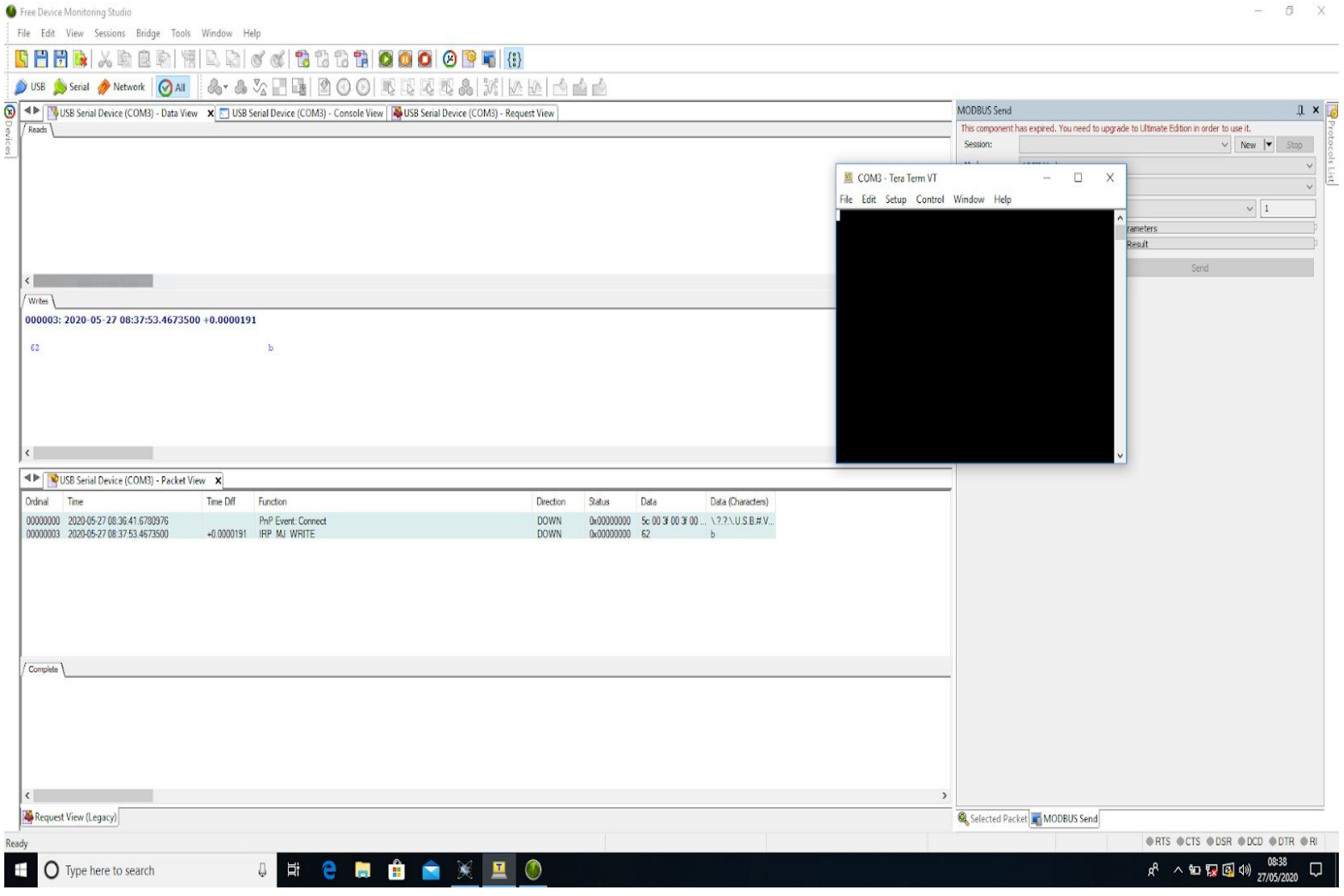




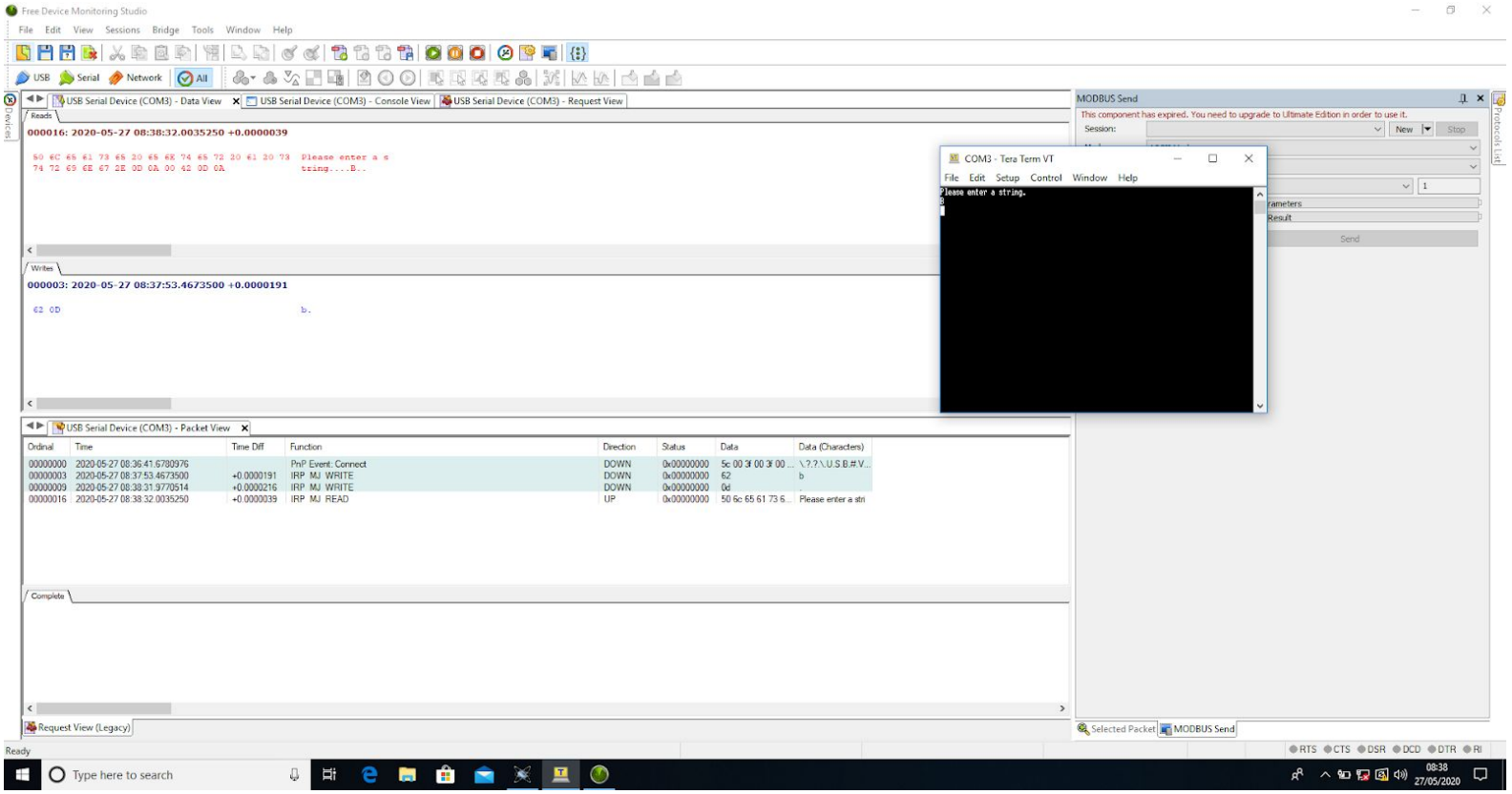
Explain followings

- Give a letter you typed and what is observed on the Tera Term

When a letter is pressed it did not display anything in the terminal, the reason for that to happen is that we do not immediately send the result, we wait for a carriage return or newline character to send it to the terminal through `putUSBUSART()` function.



- Give screenshots of the USB monitor relevant to the letter you type and the letter displayed on the Tera Term.



- One type of packet is IN and the other is OUT. Explain each case discussing why they become IN and OUT packets.

USB has four different packet types.

- ***Token packets*** indicate the type of transaction to follow,
- ***data packets*** contain the payload,
- ***handshake packets*** are used for acknowledging data or reporting errors and,
- ***The start of frame packets*** indicates the start of a new frame.

These IN and OUT packets comes under the token packets which have the following types

- **In** - Informs the USB device that the host wishes to read information.
- **Out** - Informs the USB device that the host wishes to send information.
- **Setup** - Used to begin control transfers.

In this scenario,
Host: Tera Term Terminal

USB Device: PIC microcontroller

So when we input data in the tera term terminal the host wishes to send data to the USB device which is the PIC microcontroller so it will be an ***OUT*** packet, on the other hand, PIC microcontroller sends back the data to the host when it gets an '\r' or '\n' so when the host is ready to read the data it will be an IN packet.

RESOURCES
Please find the images and the resources used in this report in the following link

https://github.com/irfanm96/CO326/tree/master/lab04/resources

REFERENCES

- https://www.beyondlogic.org/usbnutshell/usb3.shtml