

E/19/129

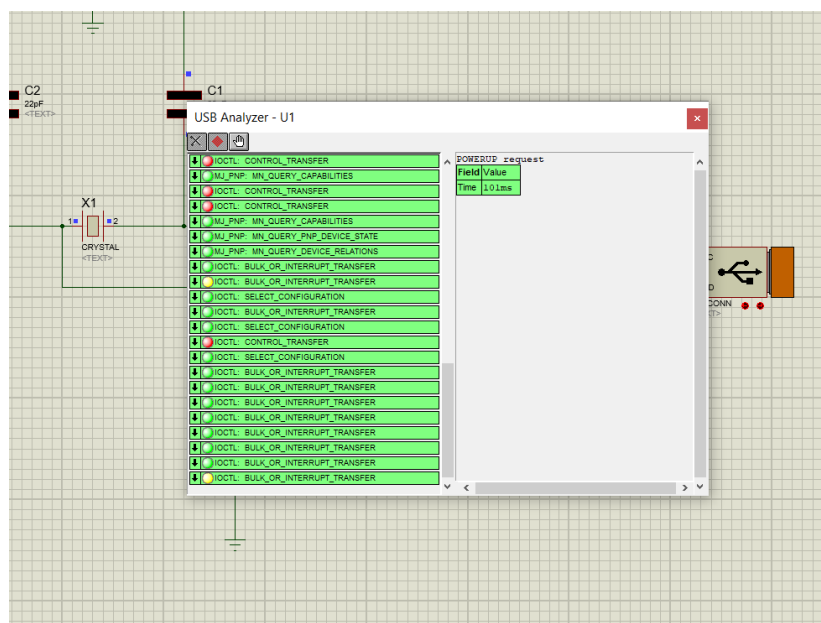
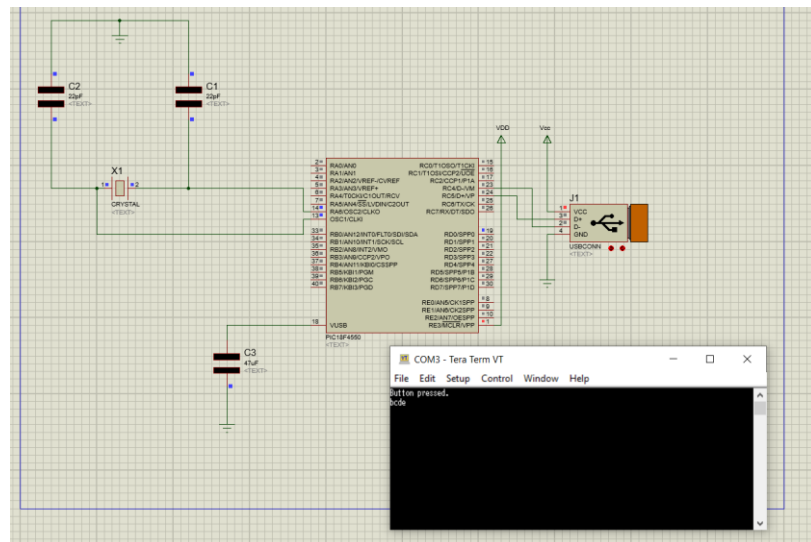
K.H. Gunawardana

## CO326 – Lab03

### USB Port I/O

## Example

Design:



HDD Monitor:

USB Serial Device (COM3) - Data ViewUSB Serial Device (COM3) - Packet View

Reads

000010: 2024-04-07 09:30:35.5208241 +0.0000016  
62b  
000034: 2024-04-07 09:30:47.1936303 +0.0000060  
63c

Writes

000003: 2024-04-07 09:30:35.5151477 +0.0000093  
61a  
000026: 2024-04-07 09:30:47.1473274 +0.0000171  
62b

USB Serial Device (COM3) - Data ViewUSB Serial Device (COM3) - Packet View

Ordinal	Time	Time Diff	Function	Direction	Status	Data	Data (Charac...
000000	2024-04-07 09:30:29.6992874		PnP Event Connect	DOWN	0x000000	5c 00 3f 00 3f 0...	\??.\USB#...
000000	2024-04-07 09:30:35.5151477	+0.0000093	IRP_MJ_WRITE	DOWN	0x000000	61	a
000000	2024-04-07 09:30:35.5208241	+0.0000016	IRP_MJ_READ	UP	0x000000	62	b
000000	2024-04-07 09:30:47.1473274	+0.0000171	IRP_MJ_WRITE	DOWN	0x000000	62	b
000000	2024-04-07 09:30:47.1936303	+0.0000060	IRP_MJ_READ	UP	0x000000	63	c
000000	2024-04-07 09:30:47.4856818	+0.0000181	IRP_MJ_WRITE	DOWN	0x000000	63	c
000000	2024-04-07 09:30:47.5090554	+0.0000048	IRP_MJ_READ	UP	0x000000	64	d
000000	2024-04-07 09:30:47.8744277	+0.0000242	IRP_MJ_WRITE	DOWN	0x000000	64	d
000000	2024-04-07 09:30:47.8860620	+0.0000051	IRP_MJ_READ	UP	0x000000	65	e

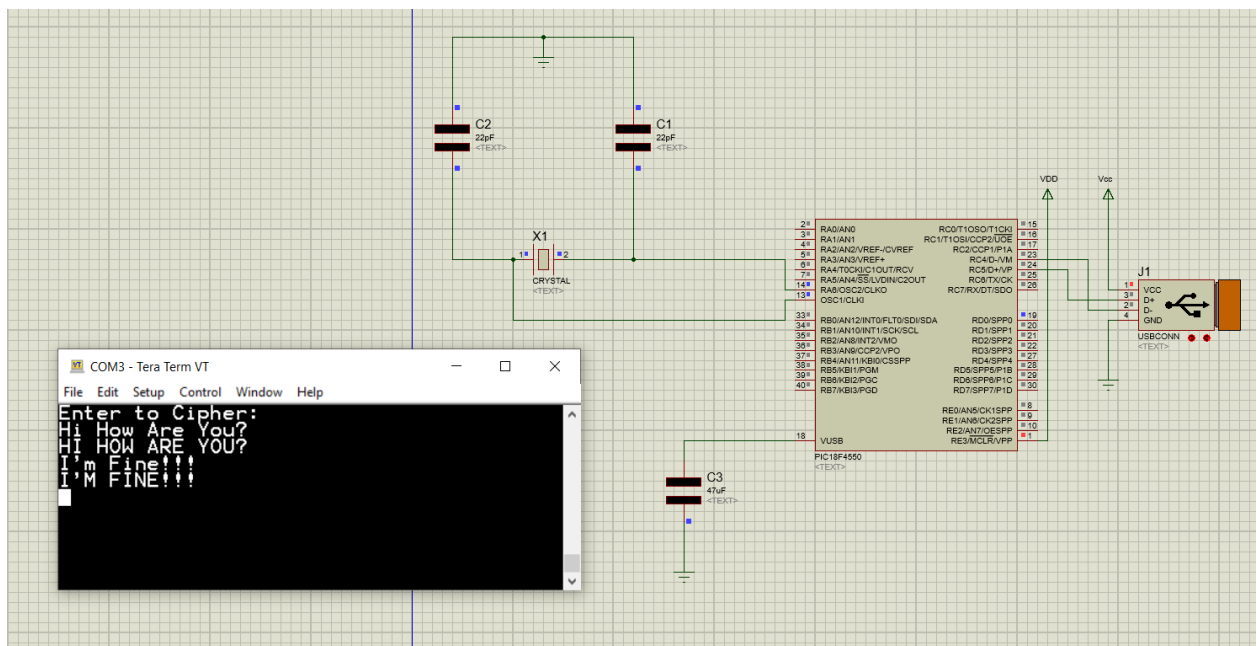
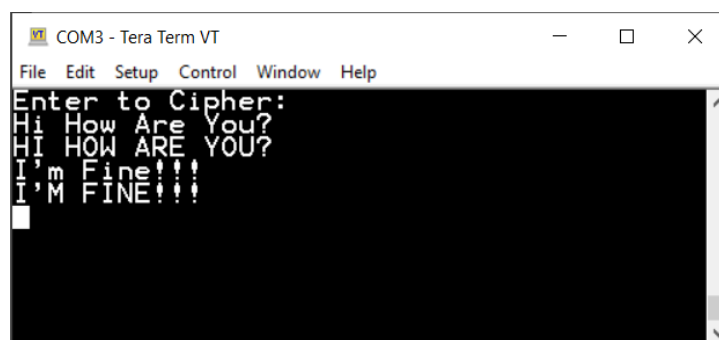
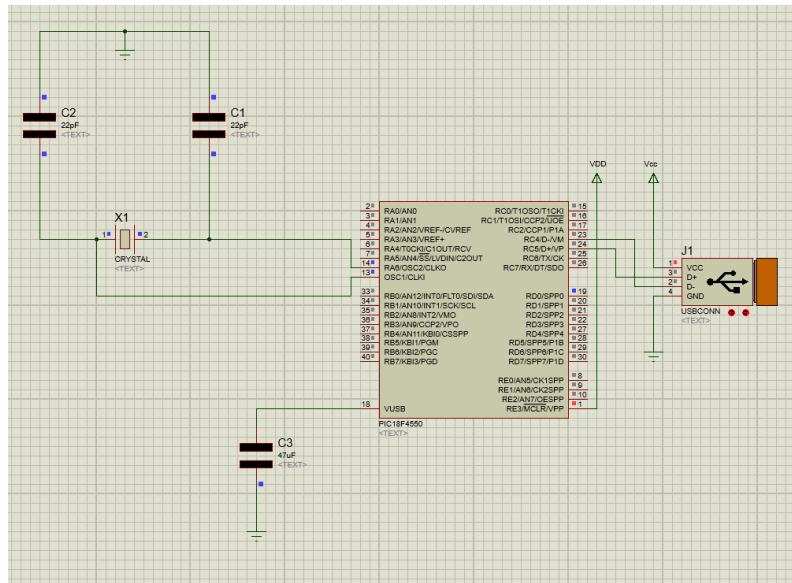
Complete

Buffer size: 0x1 bytes  
Status: 0x00000000  
65e

Request View (Legacy)

## Lab Task

### Screenshots:



## Code: app\_device\_cdc\_bacis.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](mailto:mla_licensing@microchip.com)

```
*****/
```

```
/** INCLUDES *****/
```

```
#include "system.h"
```

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

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

```
#include <stddef.h>
```

```
#include "usb.h"
```

```
#include "app_led_usb_status.h"
```

```
#include "app_device_cdc_bacis.h"
```

```
#include "usb_config.h"
```

```
/** VARIABLES *****/
```

```
static bool buttonPressed;
```

```
static bool end = false;
```

```
static char buttonMessage[] = "Enter to Cipher:\r\n";
```

```
static uint8_t readBuffer[CDC_DATA_OUT_EP_SIZE];
```

```
static uint8_t writeBuffer[CDC_DATA_IN_EP_SIZE];
```

```
static uint8_t myWriteBuffer[CDC_DATA_IN_EP_SIZE];
```

```
static uint8_t count = 0;
```

```
// Function to convert lowercase English letters to uppercase
```

```
void convertToUpperCase(char *str) {
```

```

while (*str) {

    if (*str >= 'a' && *str <= 'z') {

        *str = *str - 'a' + 'A';

    }

    str++;

}

}

/*****

* 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)
        {
            putsUSART(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;
}

```

```

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])
        {
            case 0x0D:
                end = true;
                break;

            default:
                writeBuffer[i] = readBuffer[i];
                // store in a buffer
                myWriteBuffer[count] = readBuffer[i];
                count++;
                break;
        }

    }

    if(numBytesRead > 0 && end != true)
    {
        /* After processing all of the received data, we need to send out
        * the "echo" data now.
        */
        putUSBUSART(writeBuffer, numBytesRead);
    }
}

// print the cipher
if (end && mUSBUSARTIsTxTrfReady() == true){
    if(count > 0)
    {

```

```

// Add newline and carriage return characters at the end of myWriteBuffer
myWriteBuffer[count] = '\r';

count++;

myWriteBuffer[count] = '\n';

count++;

myWriteBuffer[count] = '\0';

count++;


// Convert received bytes to uppercase
convertToUpperCase(myWriteBuffer);


// Shift the characters in myWriteBuffer to right by 2 positions
for (int j = count - 1; j >= 0; j--) {
    myWriteBuffer[j+2] = myWriteBuffer[j];
}


// Add newline and carriage return characters at the beginning of myWriteBuffer
myWriteBuffer[0] = '\r';
myWriteBuffer[1] = '\n';


// display the cipher text
putsUSBUSART(myWriteBuffer);
}


// initiate
end = false;

count = 0;
}

CDCTxService();
}

```



HDD Monitor:

COM3 - Tera Term VT

File Edit Setup Control Window Help

Enter to Cipher:  
Hi How Are You?  
HI HOW ARE YOU?  
I'm Fine!!!  
I'M FINE!!!  
abcd  
ABCD

USB Serial Device (COM3) - Data View

USB Serial Device (COM3) - Packet View

Reads

000082: 2024-04-07 15:18:54.7295697 +0.0000035

64 d

000106: 2024-04-07 15:18:56.0187156 +0.0000061

0D 0A 41 42 43 44 0D 0A 00 ..ABCD...

Writes

000074: 2024-04-07 15:18:54.7221637 +0.0000135

64 d

000098: 2024-04-07 15:18:56.0154834 +0.0000127

0D .

USB Serial Device (COM3) - Data View

USB Serial Device (COM3) - Packet View

Ordinal	Time	Time Diff	Function	Direction	Status	Data	Data (Charac...
000000...	2024-04-07 15:18:42.6505293		PnP Event: Connect	DOWN	0x000000...	5c 00 3f 00 3f 0...	{??}\USB #...
000000...	2024-04-07 15:18:53.7462823	+0.0000137	IRP_MJ_WRITE	DOWN	0x000000...	61	a
000000...	2024-04-07 15:18:54.0143301	+0.0000035	IRP_MJ_READ	UP	0x000000...	61	a
000000...	2024-04-07 15:18:54.0964608	+0.0000165	IRP_MJ_WRITE	DOWN	0x000000...	62	b
000000...	2024-04-07 15:18:54.1116677	+0.0000031	IRP_MJ_READ	UP	0x000000...	62	b
000000...	2024-04-07 15:18:54.5357571	+0.0000125	IRP_MJ_WRITE	DOWN	0x000000...	63	c
000000...	2024-04-07 15:18:54.5472702	+0.0000030	IRP_MJ_READ	UP	0x000000...	63	c
000000...	2024-04-07 15:18:54.7221637	+0.0000135	IRP_MJ_WRITE	DOWN	0x000000...	64	d
000000...	2024-04-07 15:18:54.7295697	+0.0000035	IRP_MJ_READ	UP	0x000000...	64	d
000000...	2024-04-07 15:18:56.0154834	+0.0000127	IRP_MJ_WRITE	DOWN	0x000000...	0d	.
000001...	2024-04-07 15:18:56.0187156	+0.0000061	IRP_MJ_READ	UP	0x000000...	0d 0a 41 42 43	ABCD.

Complete

000106: Read Request (UP), 2024-04-07 15:18:56.0187156 +0.0000061 (1. Device: USB Serial Device (COM3))

Buffer size: 0x9 bytes

Status: 0x00000000

0D 0A 41 42 43 44 0D 0A 00 ..ABCD...

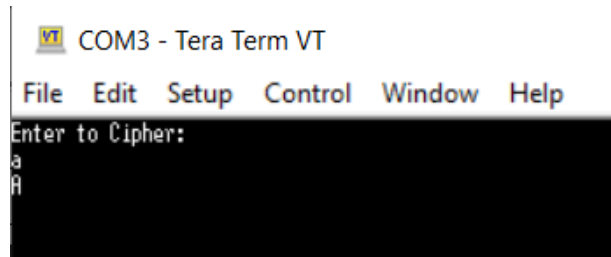
Problems and Solutions:

- Terminal Connection Issue:** Initially, there was an issue with connecting the Tera Terminal to the application. This problem was resolved by restarting the computer, which likely refreshed the USB connections and allowed the terminal to establish a connection with the application.
- Printing Characters Overlapping:** Characters were being printed on top of each other, causing readability issues. This was addressed by appending carriage return ('\r') and newline ('\n') characters at the end of each message and also at the beginning. This ensured that each message started on a new line, maintaining readability.

- **Echoing First Character Repeatedly:** Another issue encountered was the echoing of the first character repeatedly. To resolve this, a temporary array was introduced to store the full line of input before processing. This allowed for the complete line to be processed and echoed back without repetition of the first character.

### Explanations:

Consider 'a' as the input, thus using the Tera Terminal we can see, that first 'a' is echo and after pressing the enter 'A' gives back as the output.



### USB Analyzer:

The screenshot shows the USB Analyzer - U1 interface. The left pane lists various IRP structures, with 'IOCTL\_BULK\_OR\_INTERRUPT\_TRANSFER' selected. The right pane displays the details for this IRP.

Field	Request	Reply
MajorFunction	0x0F (MJ_INTERNAL_DEVICE_CONTROL)	0x0F (MJ_INTERNAL_DEVICE_CONTROL)
MinorFunction	0x00	0x00
Status	0x00000000 (STATUS_SUCCESS)	0x00000000 (STATUS_SUCCESS)

Field	Request	Reply
Time	23s 749ms	23s 751ms
UrbHeader.Function	0x0009 (BULK_OR_INTERRUPT_TRANSFER)	0x0009 (BULK_OR_INTERRUPT_TRANSFER)
UrbHeader.Length	72	72
UrbHeader.Status	0x00000000 (SUCCESS)	0x00000000 (SUCCESS)
UrbHeader.UsbdDeviceHandle	0x09FD33E0	0x09FD33E0
UrbHeader.UsbdFlags	0x00000000	0x00000000
UrbBulkOrInterruptTransfer.PipeHandle	0x09FDA5A0	0x09FDA5A0
UrbBulkOrInterruptTransfer.TransferFlags	0x00000000	0x00000000
UrbBulkOrInterruptTransfer.TransferBufferLength	1	1
UrbBulkOrInterruptTransfer.TransferBuffer	0x00000000	0x00000000

Offset	Data	Text
0x00000000	61	a

The screenshot shows the USB Analyzer - U1 interface. The left pane lists various IRP structures, with 'IOCTL\_BULK\_OR\_INTERRUPT\_TRANSFER' selected. The right pane displays the details for this IRP.

Field	Request	Reply
MajorFunction	0x0F (MJ_INTERNAL_DEVICE_CONTROL)	0x0F (MJ_INTERNAL_DEVICE_CONTROL)
MinorFunction	0x00	0x00
Status	0xC00000BB (STATUS_NOT_SUPPORTED)	0x00000000 (STATUS_SUCCESS)

Field	Request	Reply
Time	23s 769ms	24s 433ms
UrbHeader.Function	0x0009 (BULK_OR_INTERRUPT_TRANSFER)	0x0009 (BULK_OR_INTERRUPT_TRANSFER)
UrbHeader.Length	72	72
UrbHeader.Status	0x00000000 (SUCCESS)	0x00000000 (SUCCESS)
UrbHeader.UsbdDeviceHandle	0x09FD33E0	0x09FD33E0
UrbHeader.UsbdFlags	0x00000000	0x00000000
UrbBulkOrInterruptTransfer.PipeHandle	0x09FDA5B4	0x09FDA5B4
UrbBulkOrInterruptTransfer.TransferFlags	0x00000003	0x00000003
UrbBulkOrInterruptTransfer.TransferBufferLength	4096	6
UrbBulkOrInterruptTransfer.TransferBuffer	0x00000000	0x00000000

Offset	Data	Text
0x00000000	0D 0A 41 0D 0A 00	.A..

HDD Monitor:

The top screenshot displays the 'Reads' and 'Writes' sections of a USB Serial Device (COM3) monitoring tool. The 'Reads' section shows two read operations: one at 2024-04-07 16:01:00.3981285 with data '61' and another at 2024-04-07 16:01:01.4497031 with data '0D 0A 41 0D 0A 00'. The 'Writes' section shows two write operations: one at 2024-04-07 16:01:00.3802350 with data '61' and another at 2024-04-07 16:01:01.4230270 with data '0D'.

The bottom screenshot shows a detailed packet list table and a 'Complete' section. The table has columns: Ordinal, Time, Time Diff, Function, Direction, Status, Data, and Data (Charac...). The 'Complete' section shows a 'Read Request (UP)' at 2024-04-07 16:01:01.4497031 with a buffer size of 0x6 bytes and status 0x00000000.

Ordinal	Time	Time Diff	Function	Direction	Status	Data	Data (Charac...)
000000	2024-04-07 15:59:52.9816586		PnP Event Connect	DOWN	0x000000	5c 00 3f 00 3f 0	\??\USB#...
000000	2024-04-07 16:01:00.3802350	+0.0000160	IRP_MJ_WRITE	DOWN	0x000000	61	a
000000	2024-04-07 16:01:00.3981285	+0.0000089	IRP_MJ_READ	UP	0x000000	61	a
000000	2024-04-07 16:01:01.4230270	+0.0000133	IRP_MJ_WRITE	DOWN	0x000000	0d	.
000000	2024-04-07 16:01:01.4497031	+0.0000037	IRP_MJ_READ	UP	0x000000	0d 0a 41 0d 0	.A...

**Complete**

**000034: Read Request (UP), 2024-04-07 16:01:01.4497031 +0.0000037 (1. Device: USB Serial Device (COM3))**  
Buffer size: 0x6 bytes  
Status: 0x00000000  
0D 0A 41 0D 0A 00 ...A...

When observing the USB analyser, two types of packets can be identified: IN and OUT.

- **IN Packets:** These packets are sent into the USB port from the PIC18F4550 microcontroller. They represent data read by the USB port from the device. In the application context, IN packets contain data sent from the device to the host, such as characters typed on the keyboard or sensor readings.
- **OUT Packets:** Conversely, OUT packets are sent out through the USB port to the microcontroller. These packets contain commands or data from the host computer to be processed by the device. In the application, OUT packets could include commands or configuration settings sent from Tera Term to the USB device, such as baud rate adjustments or other control commands.