MICROCHIP

Regional Training Centers

COM 3101

Introduction to Full-Speed USB

Lab Manual

v1.0

Table of Contents

Lab Exercise 1a: Create an USB template project (Polling) 1-1
Lab Exercise 1b: Create an USB template project (Interrupts)..... 1-10
Lab Exercise 2: Sending and Receiving strings..... 2-1
Lab Exercise 3: Sending and Receiving raw data..... 3-1
Lab Exercise 4: Migrating Applications to USB from RS-232 UART 4-1
Lab Exercise 5: Adding a Serial Number String Descriptor 5-1
Appendix A: Removing Existing MCHPFSUSB Framework Drivers and .inf Files A-1

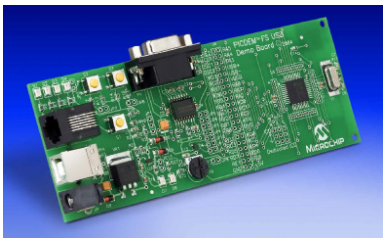
Installing the Class CD-ROM

1. Remove previously installed MCHPFSUSB framework drivers and .inf files (See Appendix A.)
2. Launch “COM3101 Restore.bat” from CD-ROM

Demo Board Setup

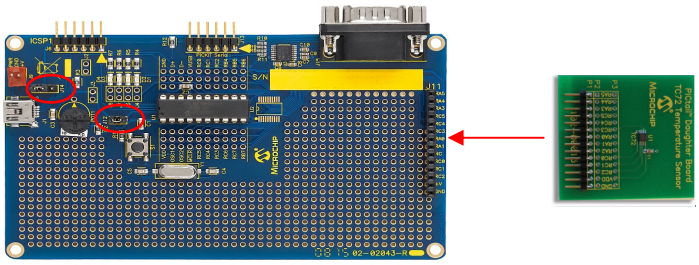
PICDEM™ FS USB Board

No hardware jumper setup required



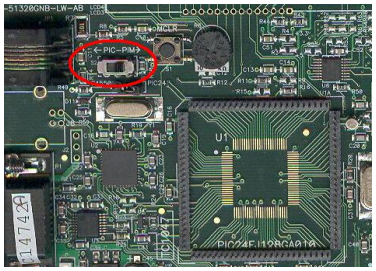
Low Pin Count Development Kit + TC72 Digital Temperature Sensor PICtail™ Demo Board

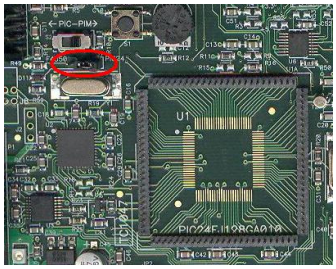
- Make sure pins 2 and 3 of J14 are shorted
- Make sure J12 is left open
- Connect TC72 PICtail Daughter Board

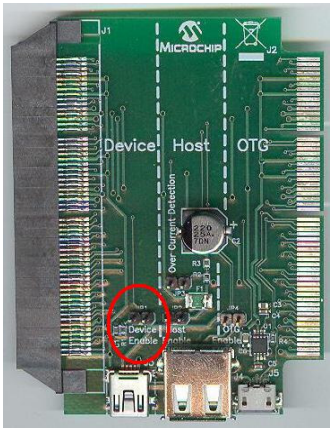


Explorer 16 + USB PICtail Plus Daughter Board + USB PIMs

- Make sure PIM is installed (PIC24FJ256GB110 or PIC32MX460F512)
  - Slider S2 set to “PIM”
  - Jumper J7 set to “PIC24”
- Make sure “Device Enable” Jumper is shorted







THIS PAGE INTENTIONALLY LEFT BLANK

Lab Exercise 1

Create an USB template project

Purpose

The Microchip USB Device Firmware Framework is a library that can be used to create new USB applications. It can be thought of as a reference design project, containing the necessary firmware code for USB operation and providing a placeholder for the user's code. With this first exercise you will learn how to configure the framework and which files need to be included in your project to achieve a fully functional USB device.

This exercise is split in two parts:

a) Create an USB project from scratch, implementing a Polling scheme
b) Modify the previous part to implement an Interrupts scheme

Requirements

Development Environment:	MPLAB® IDE v8.40 or later
C Compiler:	MPLAB® C18 v3.30 or later, C30 v3.12 or later, C32 v1.05 or later
Hardware Tools:	PICDEM FS USB Board or Low Pin Count USB Development Kit + TC72 Digital Temperature Sensor PICtail Demo Board or Explorer 16 + USB PICtail Plus Daughter Board + (PIC24F USB PIM or PIC32 USB PIM) MPLAB REAL ICE™, MPLAB ICD 3 or PICkit™ 3
Lab files on class PC:	C:\RTC\COM3101\Lab1a C:\RTC\COM3101\Lab1b

Objective

Write a short program that will turn on LED 1 while switch sw is pressed, whilst the USB Framework is running on the board. The LED should turn off only when the switch is released

Procedure

Part a

1

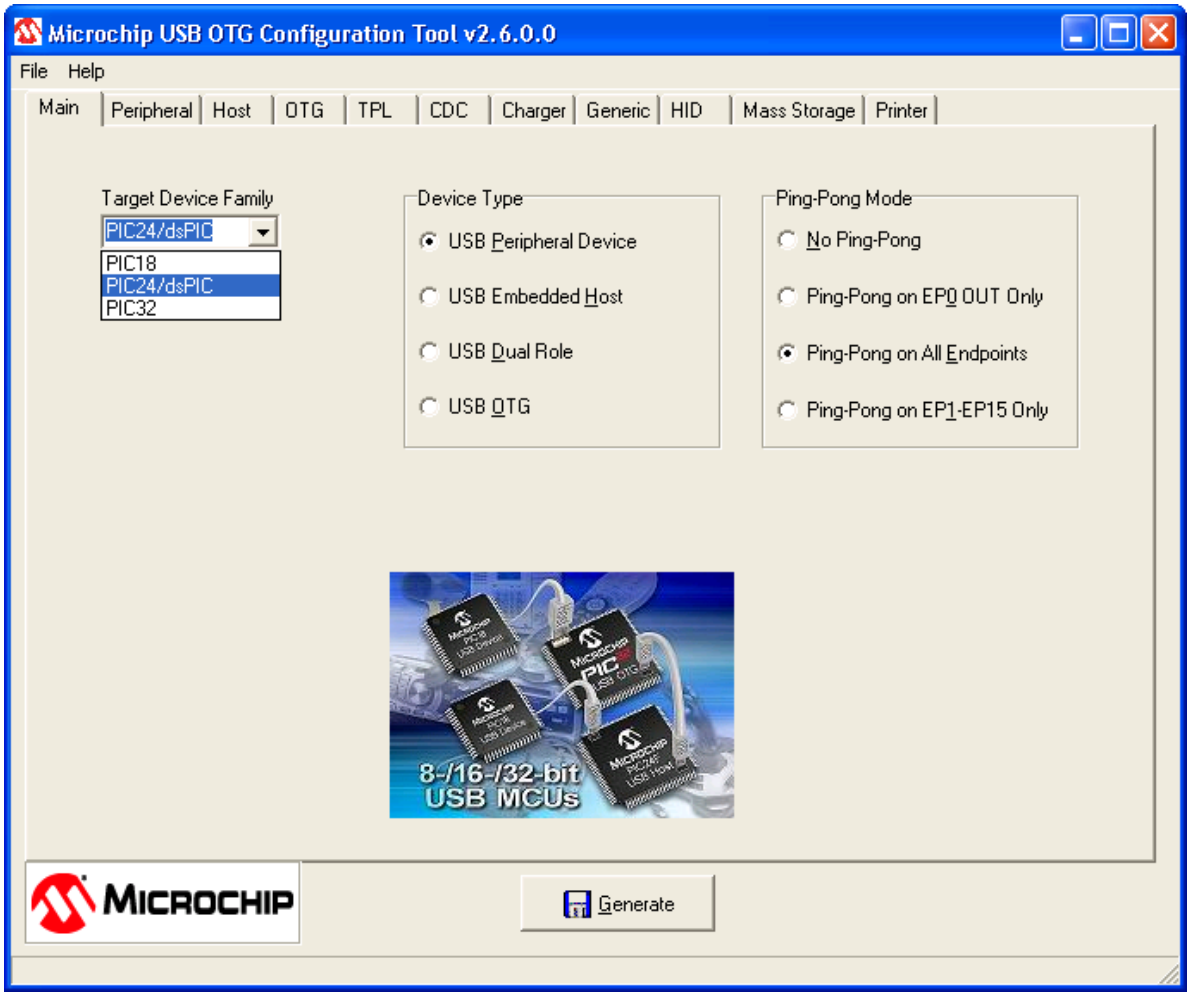
If you still have **MPLAB IDE** running, you must first close it by selecting from the menu:
File ▸ **Exit**

Launch the **Microchip USB OTG Configuration Tool** to create the **usb_config.h** file.
Start ▸ **All Programs** ▸ **Microchip** ▸ **COM3101** ▸ **USB Configuration Tool**

2

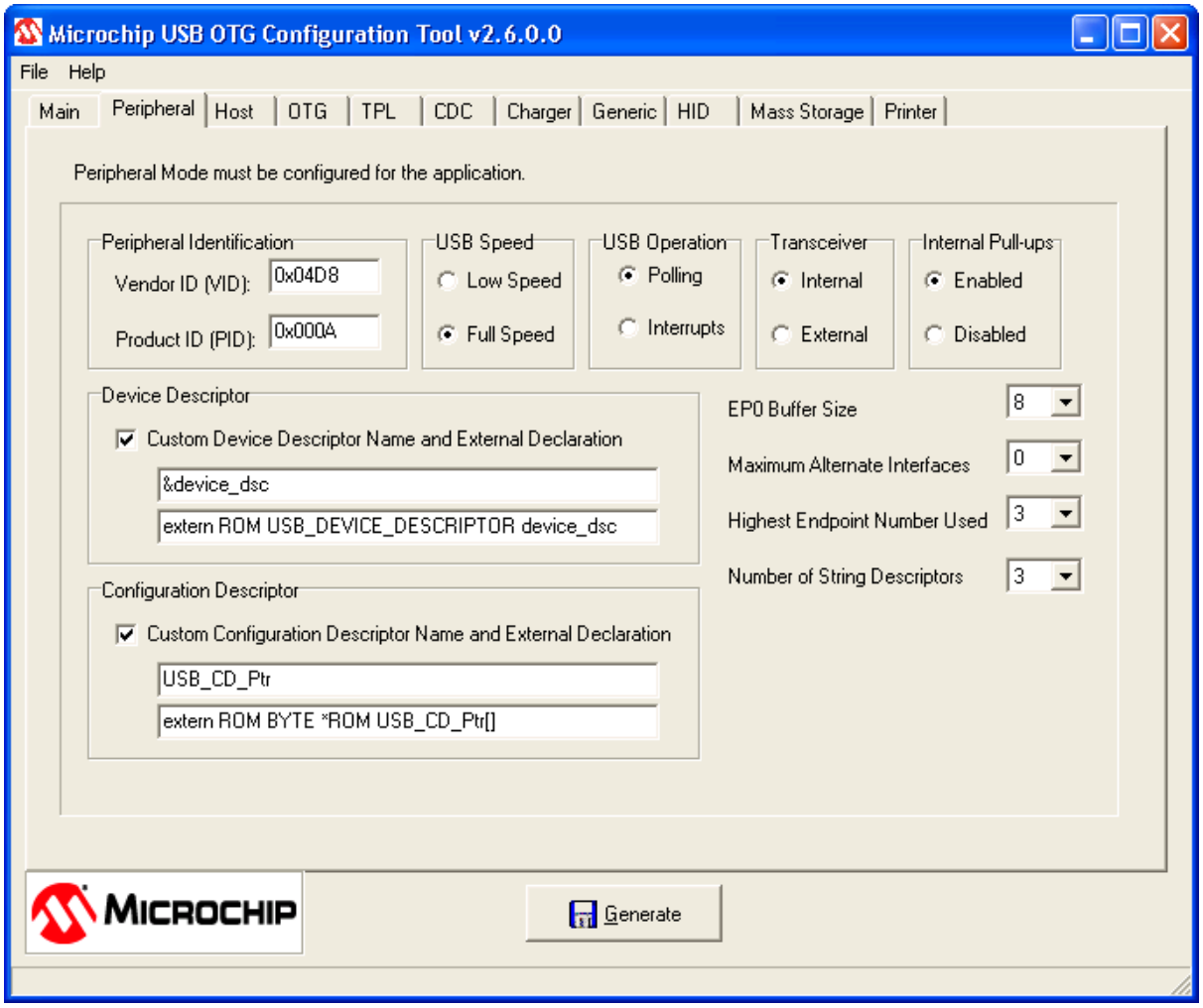
In the **Main** tab:

- Select the right **Target Device Family**
  - **PIC18** → **Low Pin Count USB Development Kit** or **PICDEM FS USB Board**
  - **PIC24/dsPIC** → **Explorer 16 + PIC24F USB PIM**
  - **PIC32** → **Explorer 16 + PIC32 USB PIM**
- Select **USB Peripheral Device** as **Device Type**
- Select **Ping-Pong on All Endpoints** as **Ping-Pong Mode**

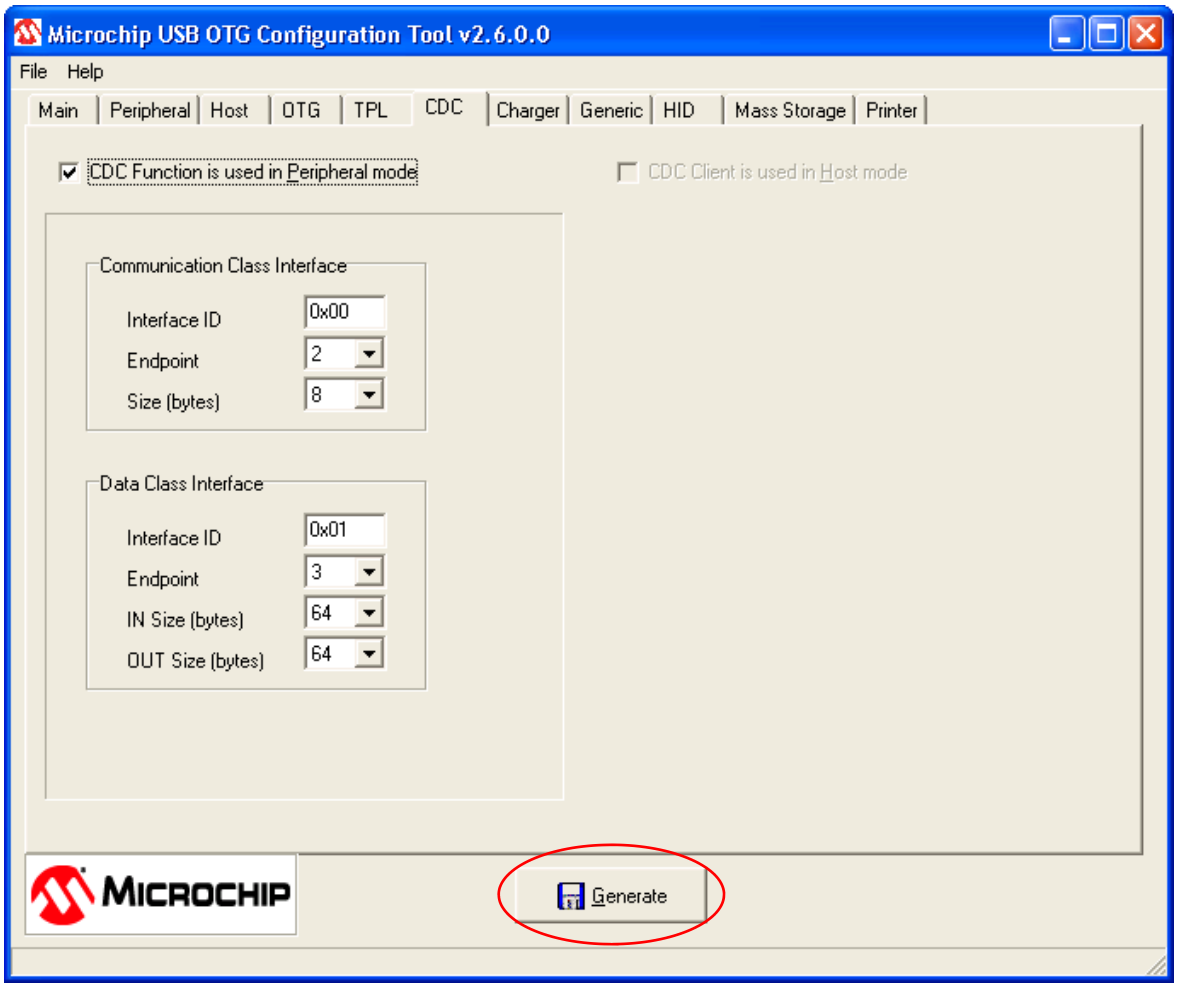


3 In the **Peripheral** tab:

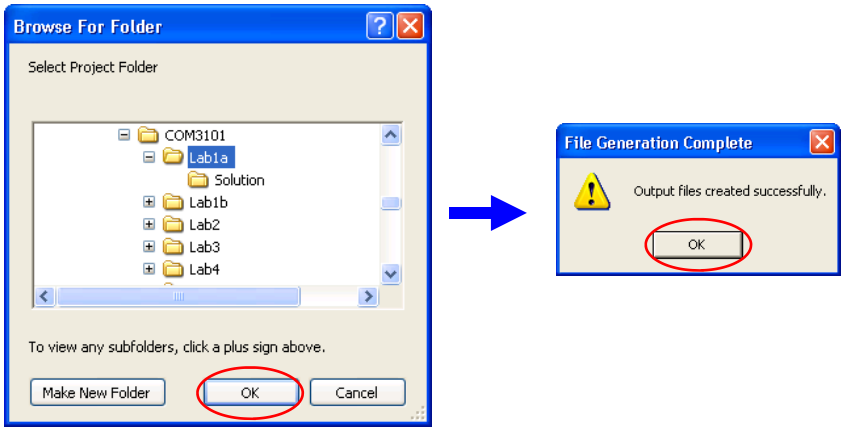
- Set **Product ID (PID)** as **0x000A**
- Select **Full Speed** as **USB Speed**
- Select **Polling** as **USB Operation**
- Select **Internal** as **Transceiver**
- Select **Enabled** as **Internal Pull-ups**
- Check the **Custom Device Descriptor Name and External Declaration** box
- Check the **Custom Configuration Descriptor Name and External Declaration** box
- Select **8** as **EP0 Buffer Size**
- Select **0** as **Maximum Alternate Interfaces**
- Select **3** as **Highest Endpoint Number Used**
- Select **3** as **Number of String Descriptors**



4 In the **CDC** tab, check the **CDC Function is used in Peripheral mode** box, let the default parameters as they are and click **Generate**



5 Select **Lab1a** folder and click **OK** and **OK** on the pop-up window



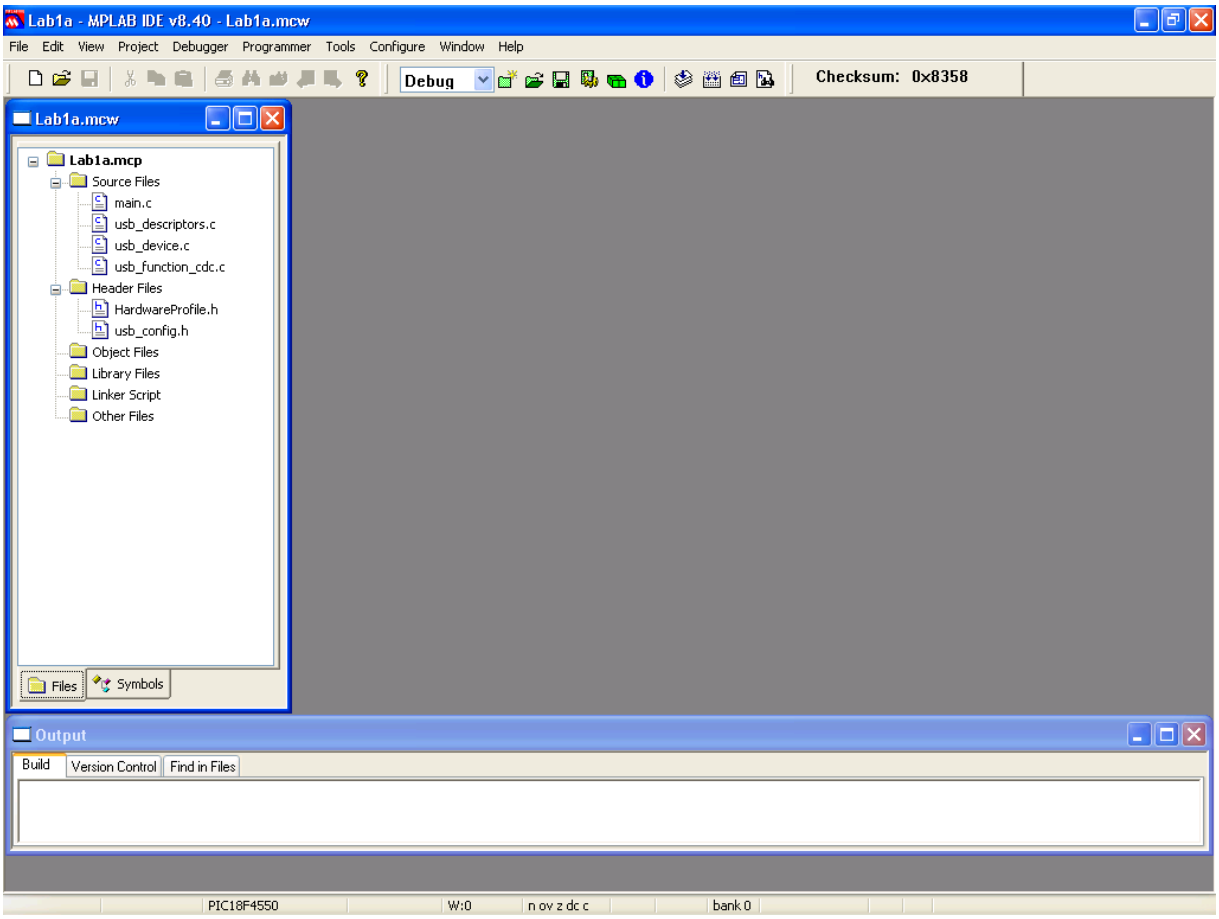
C:\RTC\COM3101\Lab1a

6 Open **MPLAB IDE** and create an empty project, named **C:\RTC\COM3101\Lab1a\Lab1a.mcp**, for your development board. Select the device as follow:

- **PIC18F4550 → PICDEM FS USB Board**
- **PIC18F14K50 → Low Pin Count USB Development Kit**
- **PIC24FJ256GB110 → Explorer 16 + PIC24F USB PIM**
- **PIC32MX460F512L → Explorer 16 + PIC32 USB PIM**

7 Add the following files to the project:
Project ▸ Add Files to Project...

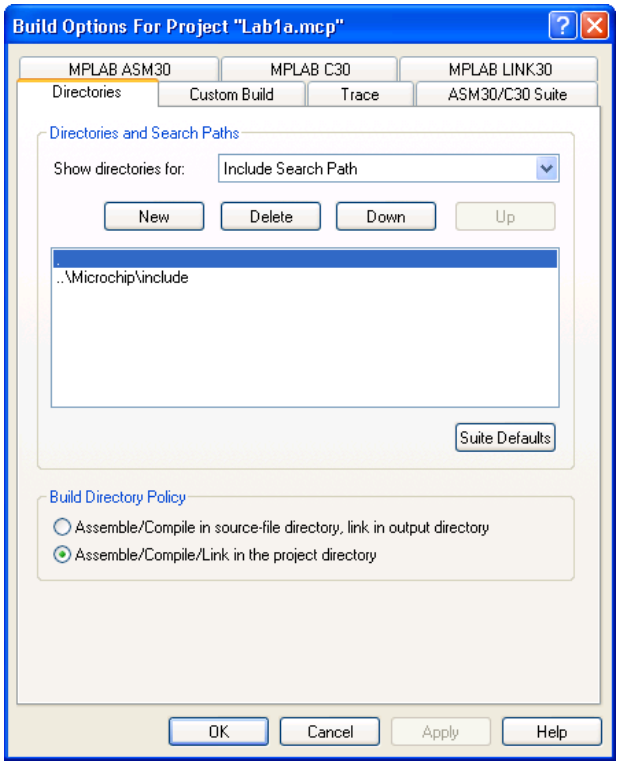
- **C:\RTC\COM3101\Lab1a\main.c**
- **C:\RTC\COM3101\Lab1a\usb_descriptors.c**
- **C:\RTC\COM3101\Lab1a\HardwareProfile.h**
- **C:\RTC\COM3101\Lab1a\usb_config.h**
- **C:\RTC\COM3101\Microchip\USB\usb_device.c**
- **C:\RTC\COM3101\Microchip\USB\CDC Device Driver\usb_function_cdc.c**



If you are using the **Low Pin Count USB Development Kit**, you also need to add the file:
rm18F14K50.lkr

8 Add **"..\Microchip\Include"** and **"."** to the **Include Search Path** into the **Build Options** for the project:

Project ▸ Build Options... ▸ Project



If you are using the **MPLAB C18** as Language Tool suite, you also need to add **"C:\MCC18\lib"** to the **Library Search Path** into the **Build Options**

9 Find the structure **sd002** in **usb_descriptors.c** and modify the product string descriptor to put your name. Remember to adjust the dimension of the array string, with the right number of characters you will use for your name.

usb_descriptors.c

```
//Product string descriptor
ROM struct{BYTE bLength; BYTE bDscType; WORD string[22];}sd002={
sizeof(sd002),USB_DESCRIPTOR_STRING,
{'#','#','#',' ','Y','o','u','r',' ','n','a',
'm','e',' ','h','e','r','e',' ','#','#','#'}
};
```

Update this number

Write your "unicode" name here

10 Find the include section (line 45) in **main.c** and include the following files:

- **Compiler.h**
- **HardwareProfile.h**
- **USB\usb.h**
- **USB\usb_function_cdc.h**

main.c

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

//### Add your code here ###

/** CONFIGURATION **************************************************/
```

11 Find function **UserInit()** in **main.c** and add the calls to the following macros:

- **mInitAllLEDs()**
- **mInitAllSwitches()**

main.c

```
void UserInit(void)
{
    //### Add your code here ###
}//end UserInit
```

12 Find function **ProcessIO()** in **main.c**
Add the code to turn on LED 1 while switch sw is pressed. The LED should turn off only when the switch is released.
TIP: Use a state machine! You can use LED 1 as status variable and change the status if it is equal to the value of the variable **sw** (mLED_1 = !sw)

main.c

```
void ProcessIO(void)
{
  // User Application USB tasks
  if ((USBDeviceState < CONFIGURED_STATE)||(USBSuspendControl==1)) return;

  //### Add your code here ###

  CDCTxService();
}//end ProcessIO
```

Task 1a.12 Reference Information

`mLED_1` Definition

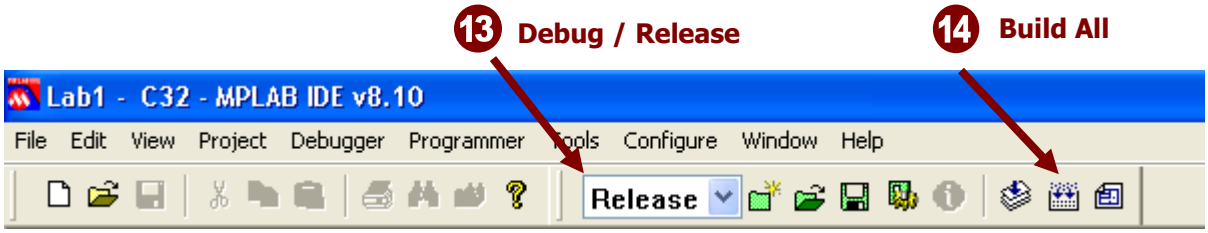
```
#define mLED_1 LATxbits.LATxx;
```

- `mLED_1` refers to LED 1. The PIN connected to LED 1 depends on the development board.
  - `mLED_1 = 0` LED 1 off
  - `mLED_1 = 1` LED 1 on

`sw` Definition

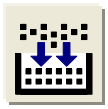
```
#define sw swX;
```

- `sw` refers to a board switch. The number of the switch depends on the development board.
  - `sw = 0` the switch is pressed
  - `sw = 1` the switch is not pressed



13 Select **Release** mode.

14 Click on the **Build All** button.



15 Program the device using **MPLAB Real ICE™/ MPLAB ICD 3/ PICkit™ 3**

1. **Programmer** ▸ **Select Programmer** ▸ **REAL ICE/MPLAB ICD 3/PICkit 3**
2. **Programmer** ▸ **Program** or click on 

If you are not using the AC164114 adapter, **DO NOT** program the Low Pin Count USB Development Kit while the USB cable is connected. Disconnect the USB cable and program the PIC18F14K50 powering the target from PICkit 3 at 3.3V

16 Connect the board to an available USB port (if not already done).
Select the correct .inf file if required by **Windows®** (Found New Hardware Wizard)

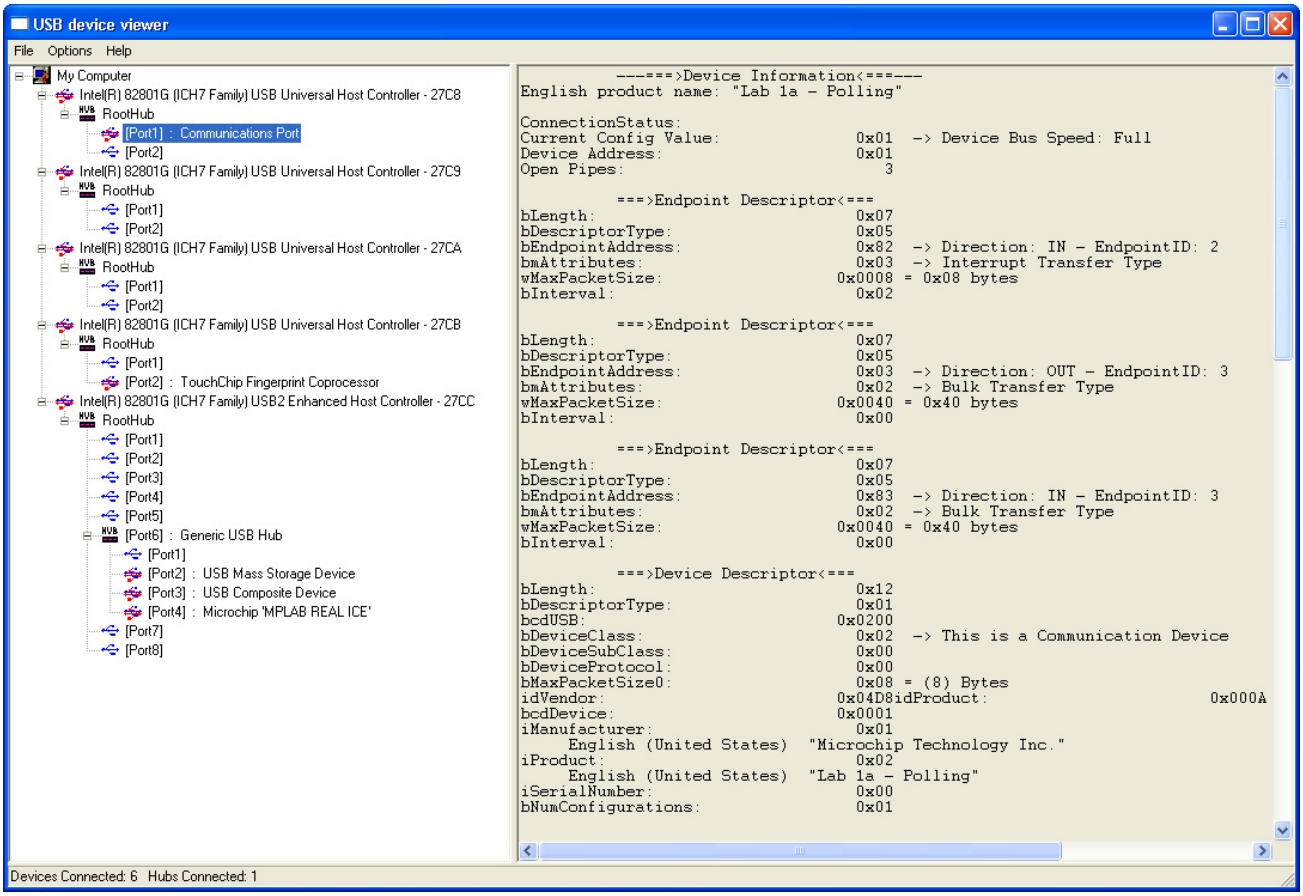
C:\RTC\COM3101\USB Tools\USB CDC Serial Demo\inf\mchpcdc.inf

17 Test the LED 1 functionality:
Low Pin Count USB Development Kit : press Switch S1
PICDEM FS USB: press Switch S2
Explorer 16: press Switch S3

18 Launch **USB Device Viewer**
Start ▸ All Programs ▸ Microchip ▸ COM3101 ▸ USB Device Viewer

Results

After clicking on the right device instance, you will see



Conclusions - Part a

This lab has shown you how to create a CDC-class device project from scratch.

The lab 1a demonstrated the coding style that is required for implementing an application within the device framework - Cooperative multitasking. You wrote non-blocking code in function ProcessIO() using a state machine with no blocking functions.

Part b

1

If you still have the previous project open, you must first close it by selecting from the menu:
File ▸ Close Workspace

Then, open Lab 1b by selecting from the menu:
File ▸ Open Workspace... and opening the workspace file located at:

C:\RTC\COM3101\Lab1b*.mcw

Select the workspace related to your development tool:

Lab 1b - C18 - Low Pin Count USB Development Kit.mcw	for the Low Pin Count USB Development Kit
Lab 1b - C18 - PICDEM FS USB.mcw	for the PICDEM FS USB Board
Lab 1b - C30.mcw	for the Explorer 16 + PIC24F USB PIM
Lab 1b - C32.mcw	for the Explorer 16 + PIC32 USB PIM

2

Find **#define USB_POLLING** in **usb_config.h** and change it to **USB_INTERRUPT**

main.c

```
#define MY_VID                      0x04D8
#define MY_PID                      0x000A
#define USB_SPEED_OPTION            USB_FULL_SPEED
#define USB_POLLING
#define USB_PULLUP_OPTION           USB_PULLUP_ENABLE
#define USB_TRANSCEIVER_OPTION      USB_INTERNAL_TRANSCEIVER
#define USB_EP0_BUFF_SIZE           8
#define USB_MAX_NUM_INT             (0+1)
#define USB_MAX_EP_NUMBER           3
#define USB_NUM_STRING_DESCRIPTORS  3
```

3 Find function **main()** in **main.c** and call function **USBDeviceAttach()** before the **while(1)** loop

main.c

```
#if defined(__18CXX)
void main(void)
#else
int main(void)
#endif
{
    InitializeSystem();

    while(1)
    {
```

Call the function here

Task 1b.3 Reference Information

USBDeviceAttach() Function Prototype

```
void USBDeviceAttach(void);
```

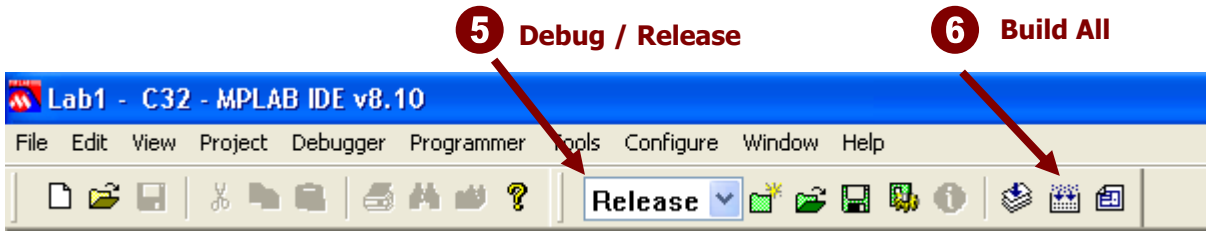
This function indicates to the USB module that the USB device has been attached to the bus. This function needs to be called in order for the device to start to enumerate on the bus. Should only be called when USB_INTERRUPT is defined.

4 Remove the call to function **USBDeviceTasks()** from function **main()**.

main.c

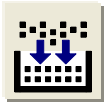
```
#if defined(__18CXX)
void main(void)
#else
int main(void)
#endif
{
    InitializeSystem();

    while(1)
    {
        // Check bus status and service USB interrupts.
        USBDeviceTasks(); // Using polling, must call this function
                          // periodically. This function will take care
                          // of processing and responding to SETUP
                          // transactions.
```

Remove this function



5 Select **Release** mode.

6 Click on the **Build All** button.



7 Program the device using **MPLAB Real ICE™/ MPLAB ICD 3/ PICkit™ 3**

1. **Programmer** ▸ **Select Programmer** ▸ **REAL ICE/MPLAB ICD 3/PICkit 3**
2. **Programmer** ▸ **Program** or click on 

If you are not using the AC164114 adapter, **DO NOT** program the Low Pin Count USB Development Kit while the USB cable is connected. Disconnect the USB cable and program the PIC18F14K50 powering the target from PICkit 3 at 3.3V

8 Connect the board to an available USB port (if not already done).
Select the correct .inf file if required by **Windows®** (Found New Hardware Wizard)

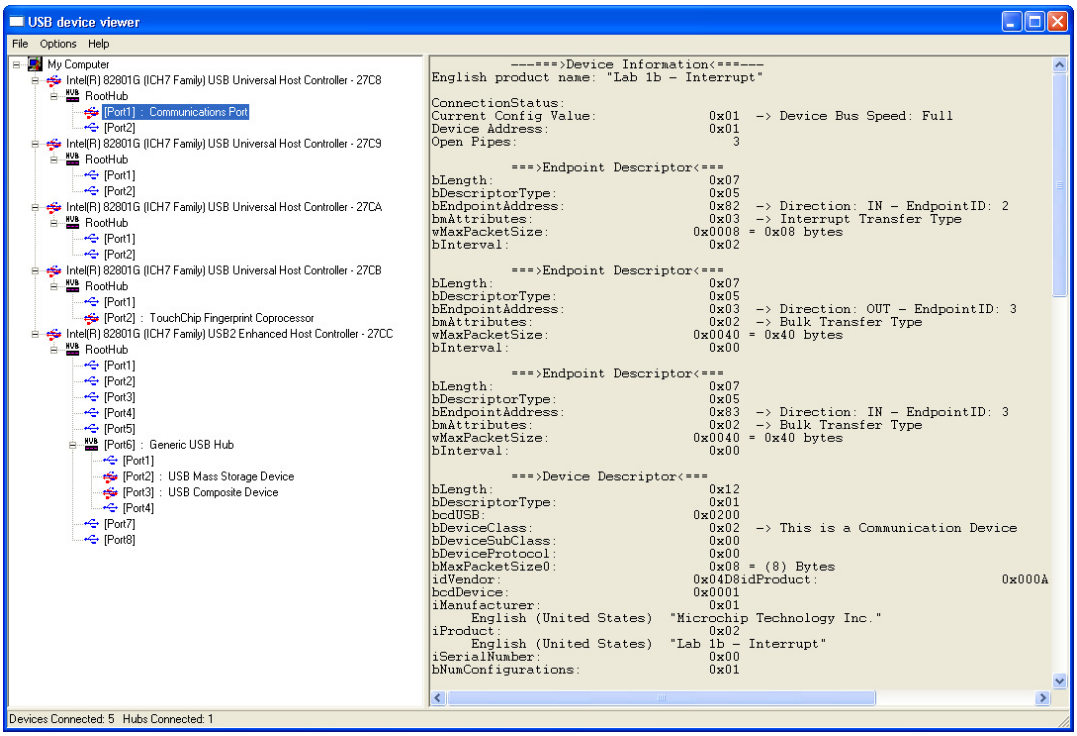
C:\RTC\COM3101\USB Tools\USB CDC Serial Demo\inf\mchpcdc.inf

9 Test the LED 1 functionality:
Low Pin Count USB Development Kit : press Switch S1
PICDEM FS USB: press Switch S2
Explorer 16: press Switch S3

10 Launch **USB Device Viewer**
Start ▸ **All Programs** ▸ **Microchip** ▸ **COM3101** ▸ **USB Device Viewer**

Results

After clicking on the right device instance, you will see



Conclusions - Part b

This lab has shown you how to configure a CDC-class device project Interrupt driven.

The same result can be achieve using Lab 1a procedure but selecting **Interrupts** as **USB Operation** in **Peripheral** tab of the **USB OTG Configuration Tool**.

This first Lab just shows how to create an USB project from scratch. The device is enumerated as CDC Class peripheral, but there's no interaction between USB stack and User's Application. The following exercises will show how to use the CDC function driver to send and receive data to and from the PC.

THIS PAGE INTENTIONALLY LEFT BLANK

Lab Exercise 2

Sending and Receiving strings

Purpose

In a typical communication, the data exchange is from device to PC and vice versa.
In this second lab, you will learn how to send and receive strings from PC and make the device perform a certain task, based on the received string.

Requirements

Development Environment:	MPLAB® IDE v8.40 or later
C Compiler:	MPLAB® C18 v3.30 or later, C30 v3.12 or later, C32 v1.05 or later
Hardware Tools:	PICDEM FS USB Board or Low Pin Count USB Development Kit + TC72 Digital Temperature Sensor PICtail Demo Board or Explorer 16 + USB PICTail Plus Daughter Board + (PIC24F USB PIM or PIC32 USB PIM) MPLAB Real ICE™, ICD 3 or PICkit™ 3
Lab files on class PC:	C:\RTC\COM3101\Lab2
Software Tools:	Tera Term (http://ttssh2.sourceforge.jp)

Objective

Write a short program to toggle LED 1 when the ASCII string “Microchip” is received from the terminal application and send to it the ASCII string “Hello World!\r\n” when the switch sw is pressed.

Procedure

1

If you still have the previous project open, you must first close it by selecting from the menu:
File ▸ **Close Workspace**

Then, open Lab 2 by selecting from the menu:
File ▸ **Open Workspace...** and opening the workspace file located at:

C:\RTC\COM3101\Lab2*.mcw

Select the workspace related to your development tool:

Lab 2 - C18 - Low Pin Count USB Development Kit.mcw	for the Low Pin Count USB Development Kit
Lab 2 - C18 - PICDEM FS USB.mcw	for the PICDEM FS USB Board
Lab 2 - C30.mcw	for the Explorer 16 + PIC24F USB PIM
Lab 2 - C32.mcw	for the Explorer 16 + PIC32 USB PIM

2

Find function **Lab2a()** in **main.c**

main.c

```
/*********************************************************************
 * Function:        void Lab2a(void)
 *
 * PreCondition:    None
 *
 * Input:           None
 *
 * Output:          None
 *
 * Side Effects:    None
 *
 * Overview:        In this function you should write the code to send a
 *                  string ("Hello World!\n\r") to HyperTerminal when
 *                  sw is pressed
 *
 * Note:            None
 ********************************************************************/
void Lab2a(void)
{
      //### Write your code here
}
```

3 Write code in this function that sends a literal null-terminated string of text (**"Hello World!\r\n"**) to the PC when switch sw on the development board.

Task 2.3 Reference Information

`SwitchIsPressed()` Function Prototype

```
BOOL SwitchIsPressed(void);
```

- Returns 1 if switch sw was pressed
- Returns 0 if switch sw was not pressed

`USBUSARTIsTxTrfReady()` Macro

```
BOOL USBUSARTIsTxTrfReady(void);
```

- Returns 1 when cdc_trf_state is CDC_TX_READY
- Returns 0 when cdc_trf_state is not CDC_TX_READY

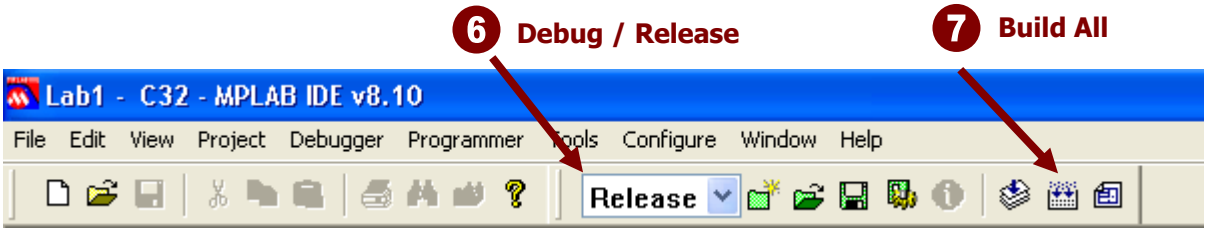
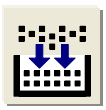
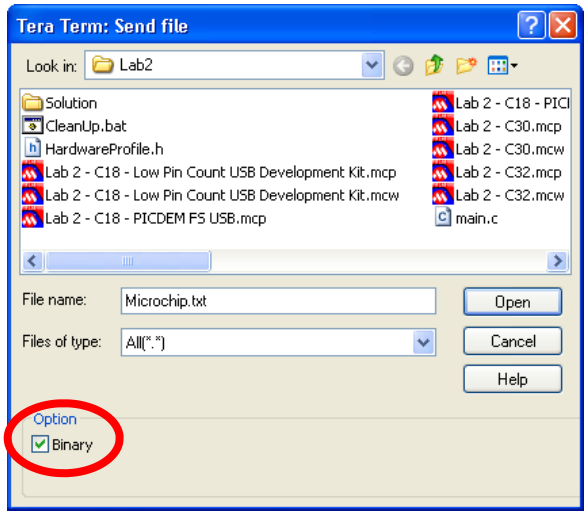
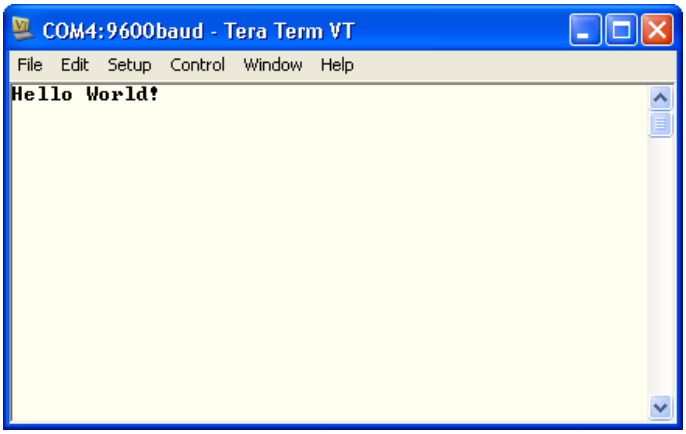
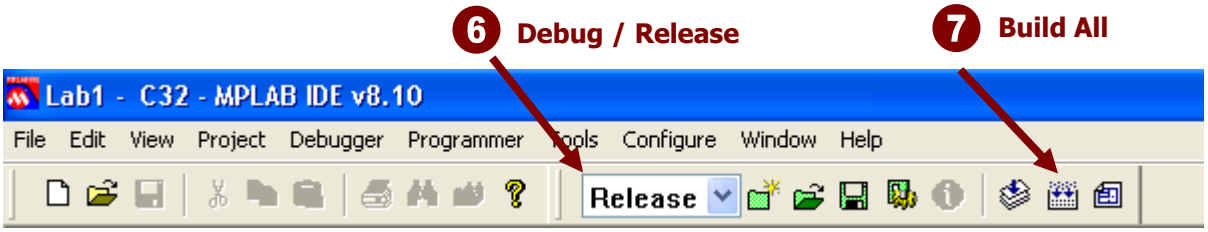
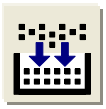
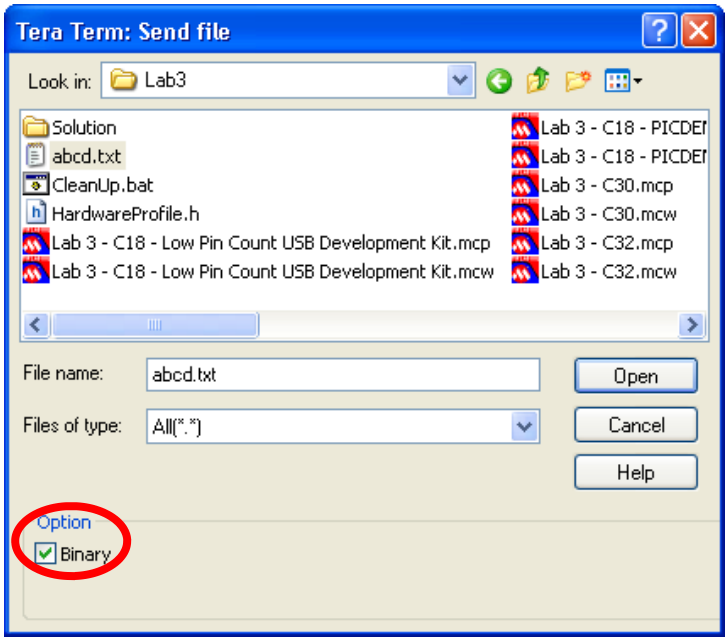
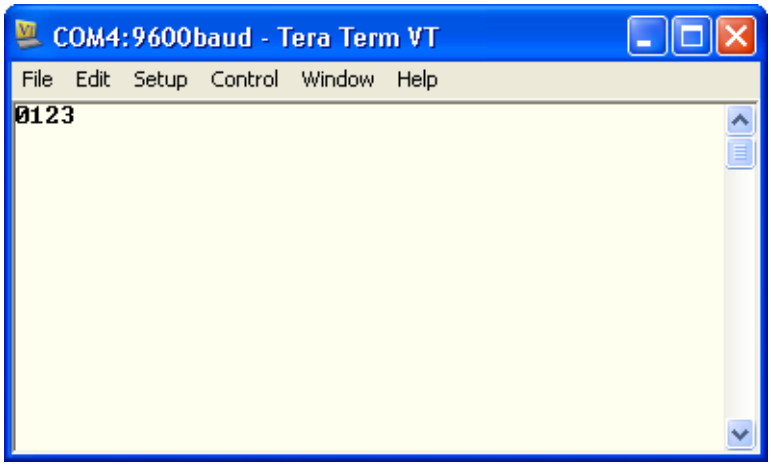
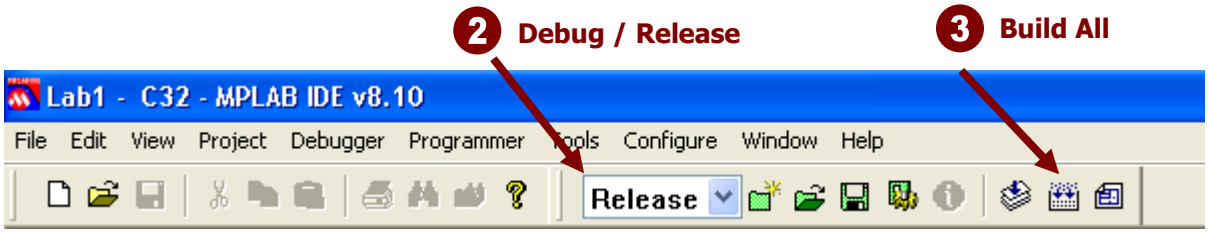
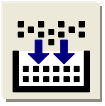
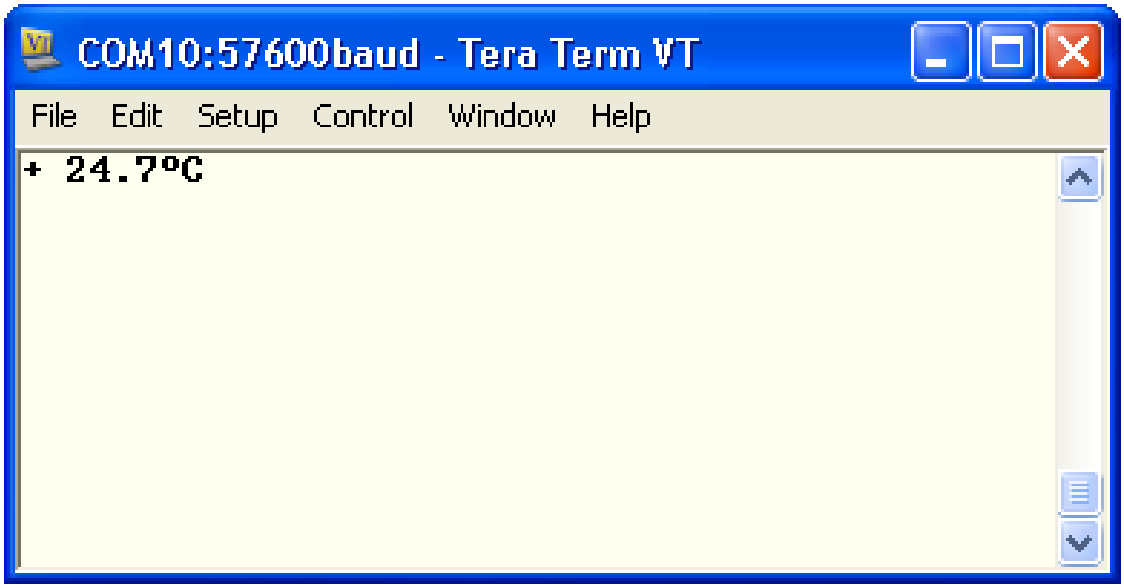
`putrsUSBUSART()` Function Prototype

```
void putrsUSBUSART(const ROM char *data);
```

- `data` is a pointer to string stored in ROM. Usage examples:
  - `putrsUSBUSART("Hello");`
  - ```
    ROM char Welcome[]="Hello";
    putrsUSBUSART(Welcome);
    ```

4 Find function **Lab2b()** in **main.c**

main.c

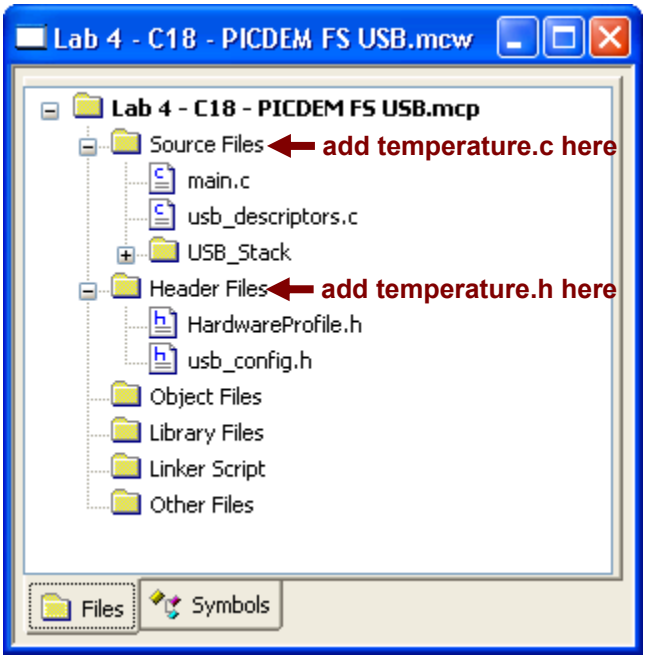
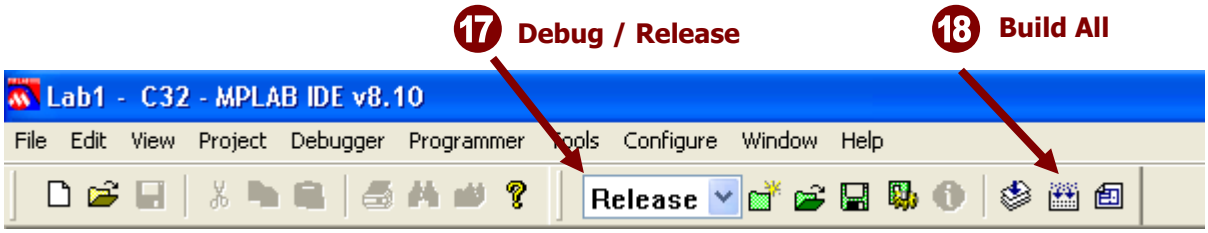
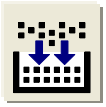
```
/*********************************************************************
 * Function:        void Lab2b(void)
 *
 * PreCondition:    None
 *
 * Input:           None
 *
 * Output:          None
 *
 * Side Effects:    None
 *
 * Overview:        In this function you should write the code to
 *                  receive a string from HyperTerminal and toggle LED1
 *                  if the received string is "Microchip"
 *
 * Note:            None
 ********************************************************************/
void Lab2b(void)
{
      //### Write your code here
}
```
```

5 Write code in this function that will check if data was received from the PC Host, then evaluate the content of it. If the received ASCII string is “Microchip”, toggle LED 1

Task 2.5 Reference Information

`getsUSBUSART()` Function Prototype

```
BYTE getsUSBUSART(char *buffer, BYTE len);
```

- `buffer` is a pointer to user buffer of size equals to or larger than `len`.
- `len` is the expected number of input bytes.

`mLED_1_Toggle()` Macro

```
mLED_1_Toggle();
```

- Toggle Led 1 (LED D1 on PICDEM FS USB Board and Low Pin Count USB Development Kit, LED D3 on Explorer 16 + PIM24F USB PIM or PIC32 USB PIM)

`USB_Out_Buffer[ ]` Array

```
char USB_Out_Buffer[64];
```

- OUT Buffer that can be used to hold data sent from the PC Host

`strcmppgm2ram()` Function Prototype

```
signed char strcmppgm2ram(const char *s1, const MEM_MODEL rom char *s2);
```

- Return 0 if string s1 and string s2 are identical.
- `s1` is a pointer to string in data memory.
- `s2` is a pointer to string in program memory
- Usage example:
  strcmppgm2ram(pStr, "Hello") return 0 if the string pointed by pStr is “Hello”



6 Select **Release** mode.

7 Click on the **Build All** button.



8 Program the device using **MPLAB REAL ICE/ MPLAB ICD 3/ PICkit 3**

1. **Programmer** ▸ **Select Programmer** ▸ **REAL ICE/MPLAB ICD 3/PICkit 3**
2. **Programmer** ▸ **Program** or click on 

If you are not using the AC164114 adapter, **DO NOT** program the Low Pin Count USB Development Kit while the USB cable is connected. Disconnect the USB cable and program the PIC18F14K50 powering the target from PICkit 3 at 3.3V

9 Connect the board to an available USB port (if not already done).
Select the correct .inf file if required by **Windows®** (Found New Hardware Wizard)

C:\RTC\COM3101\USB Tools\USB CDC Serial Demo\inf\mchpcdc.inf

10 Launch **Tera Term** and select the **Virtual COM Port** assigned to your demo board.

11 Send the string to **Tera Term**:

Low Pin Count USB Development Kit : press Switch S1
PICDEM FS USB: press Switch S2
Explorer 16: press Switch S3

Send the "**Microchip.txt**" file to the board, in **binary** format:
File ▸ **Send File...**

C:\RTC\COM3101\Lab2\Microchip.txt



Results

After pressing switch sw on the demo board, you will see



After sending the string “Microchip” (Microchip.txt), you will see the LED 1 changing its state

Conclusions

This lab has shown you how to use the CDC-class device framework API to send and receive Strings from the PC Host to your device via USB and vice versa.

The CDC-class application implemented on the device simplifies USB communications with a PC Host by appearing as a device connected to a COM port. This allows your application to exchange data with existing PC applications, such as HyperTerminal or Tera Term, which communicate using the PCs COM ports.

You wrote code in function Lab2a() and you checked the status of USB Stack before sending the data. In this way, you are sure that the stack can satisfy the USB protocol specification and your data are sent correctly.

You wrote code in function Lab2b() to check if some data arrived from the PC Host. Based on the value, you wrote code to make the device perform a specific task. In this way, you have learned how to implement a simple command/response protocol via USB.

Lab Exercise 3

Sending and Receiving raw data

Purpose

Sending and receiving data from your device are not limited to ASCII characters or strings.
In a typical embedded environment, device and host need to exchange raw data and the message information depends on the implemented application protocol. In this third lab, you will learn how to send and receive generic data to and from the Terminal application.

Requirements

Development Environment:	MPLAB® IDE v8.40 or later
C Compiler:	MPLAB® C18 v3.30 or later, C30 v3.12 or later, C32 v1.05 or later
Hardware Tools:	PICDEM FS USB Board or Low Pin Count USB Development Kit + TC72 Digital Temperature Sensor PICtail Demo Board or Explorer 16 + USB PICTail Plus Daughter Board + (PIC24F USB PIM or PIC32 USB PIM) MPLAB Real ICE™, ICD 3 or PICkit™ 3
Lab files on class PC:	C:\RTC\COM3101\Lab3
Software Tools:	Tera Term (http://ttssh2.sourceforge.jp)

Objective

Write a short program to toggle LED 1 when the array {0x61, 0x62, 0x63, 0x64} is received from the terminal application and send to it the array {0x30, 0x31, 0x32, 0x33} when the switch sw is pressed.

Procedure

1

If you still have the previous project open, you must first close it by selecting from the menu:
File ▸ **Close Workspace**

Then, open Lab 3 by selecting from the menu:
File ▸ **Open Workspace...** and opening the workspace file located at:

C:\RTC\COM3101\Lab3*.mcw

Select the workspace related to your development tool:

Lab 3 - C18 - Low Pin Count USB Development Kit.mcw	for the Low Pin Count USB Development Kit
Lab 3 - C18 - PICDEM FS USB.mcw	for the PICDEM FS USB Board
Lab 3 - C30.mcw	for the Explorer 16 + PIC24F USB PIM
Lab 3 - C32.mcw	for the Explorer 16 + PIC32 USB PIM

2

Find function **Lab3a()** in **main.c**

main.c

```
/*********************************************************************
 * Function:        void Lab3a(void)
 *
 * PreCondition:    None
 *
 * Input:           None
 *
 * Output:          None
 *
 * Side Effects:    None
 *
 * Overview:        In this function you should write the code to send
 *                  an array {0x30, 0x31, 0x32, 0x33} to HyperTerminal
 *                  sw is pressed
 *
 * Note:            None
 ********************************************************************/
void Lab3a(void)
{
      //### Write your code here
}
```

3 Write code in function **Lab3a()** that sends 4 bytes to the PC when switch sw is pressed

Task 3.3 Reference Information

`SwitchIsPressed()` Function Prototype

```
BOOL SwitchIsPressed(void);
```

- Returns 1 if switch sw was pressed
- Returns 0 if switch sw was not pressed

`USBUSARTIsTxTrfReady()` Macro

```
BOOL USBUSARTIsTxTrfReady(void);
```

- Returns 1 when cdc_trf_state is CDC_TX_READY
- Returns 0 when cdc_trf_state is not CDC_TX_READY

`putUSBUSART()` Function Prototype

```
void putUSBUSART(char *data, BYTE Length);
```

- `data` is a pointer to user buffer of size equals to or larger than `Length`.
- `Length` is the number of bytes to send

`USB_In_Buffer[ ]` Array

```
char USB_In_Buffer[64];
```

- IN Buffer that can be used to hold data to send to the Host

4 Find function **Lab3b()** in **main.c**

main.c

```
/*********************************************************************
 * Function:        void Lab3b(void)
 *
 * PreCondition:    None
 *
 * Input:           None
 *
 * Output:          None
 *
 * Side Effects:    None
 *
 * Overview:        In this function you should write the code to
 *                  receive an array from HyperTerminal and toggle LED1
 *                  if the received array is {0x61, 0x62, 0x63, 0x64}
 *
 * Note:            None
 ********************************************************************/
void Lab2b(void)
{
      //### Write your code here
}
```

5 Write code in this function that will check if data was received from the PC Host, then evaluate the content of it. If the received array is {0x61, 0x62, 0x63, 0x64}, toggle LED 1

Task 3.5 Reference Information

`getsUSBUSART()` Function Prototype

```
void getsUSBUSART(char *buffer, BYTE len);
```

- `buffer` is a pointer to user buffer of size equals to or larger than `len`.
- `len` is the expected number of input bytes.

`mLED_1_Toggle()` Macro

```
mLED_1_Toggle();
```

- Toggle Led 1 (LED D1 on PICDEM FS USB Board and Low Pin Count USB Development Kit, LED D3 on Explorer 16 + PIM24F USB PIM or PIC32 USB PIM)

`USB_Out_Buffer[ ]` Array

```
char USB_Out_Buffer[64];
```

- OUT Buffer that can be used to hold data sent from the PC Host



6 Select **Release** mode.

7 Click on the **Build All** button.



8 Program the device using **MPLAB Real ICE™/ MPLAB ICD 3/ PICkit™ 3**

1. **Programmer** ▸ **Select Programmer** ▸ **REAL ICE/MPLAB ICD 3/PICkit 3**
2. **Programmer** ▸ **Program** or click on 

If you are not using the AC164114 adapter, **DO NOT** program the Low Pin Count USB Development Kit while the USB cable is connected. Disconnect the USB cable and program the PIC18F14K50 powering the target from PICkit 3 at 3.3V

9 Connect the board to an available USB port (if not already done).
Select the correct .inf file if required by **Windows®** (Found New Hardware Wizard)

C:\RTC\COM3101\USB Tools\USB CDC Serial Demo\inf\mchpcdc.inf

10 Launch **Tera Term** and select the **Virtual COM Port** assigned to your demo board.

11 Send the array to **Tera Term**:

Low Pin Count USB Development Kit : press Switch S1
PICDEM FS USB: press Switch S2
Explorer 16: press Switch S3

Send the "**abcd.txt**" file to the board, in **binary** format:
File ▸ Send File...

C:\RTC\COM3101\Lab3\abcd.txt



Results

After pressing switch sw on the demo board, you will see



After sending the array {0x60, 0x61, 0x62, 0x63, 0x64} (abcd.txt), you will see the LED 1 changing its state

Conclusions

This lab has shown you how to use the CDC-class device framework API to send and receive raw data from the PC Host to your device via USB and vice versa.

The CDC-class application implemented on the device simplifies USB communications with a PC Host by appearing as a device connected to a COM port. This allows your application to exchange data with existing PC applications, such as HyperTerminal or Tera Term, which communicate using the PCs COM ports.

You wrote code in function Lab3a() and you checked the status of USB Stack before sending the data. In this way, you are sure that the stack can satisfy the USB protocol specification and your data are sent correctly.

You wrote code in function Lab3b() to check if some data arrived from the PC Host. Based on the value, you wrote code to make the device perform a specific task. In this way, you have learned how to implement a simple command/response protocol via USB.

Lab Exercise 4

Migrating Applications to USB from RS-232 UART

Purpose

You may already have an existing application which communicates to the PC via an RS-232 UART, and you need to migrate the design over to use USB communications with minimal effect on the PC application. This lab will show you how to integrate an existing application into the MCHPFSUSB CDC firmware framework, using some of the APIs we have used in the previous labs.

Requirements

Development Environment:	MPLAB® IDE v8.40 or later
C Compiler:	MPLAB® C18 v3.30 or later, C30 v3.12 or later, C32 v1.05 or later
Hardware Tools:	PICDEM FS USB Board or Low Pin Count USB Development Kit + TC72 Digital Temperature Sensor PICtail Demo Board or Explorer 16 + USB PICTail Plus Daughter Board + (PIC24F USB PIM or PIC32 USB PIM) MPLAB Real ICE™, ICD 3 or PICkit™ 3
Lab files on class PC:	C:\RTC\COM3101\Lab4
Software Tools:	Tera Term (http://ttssh2.sourceforge.jp)

Objective

Merge an existing application that sends an ASCII temperature measurement to the PC when switch sw is pressed and the MCHPFSUSB CDC firmware framework.

Procedure—Running the existing application

1

If you still have the previous project open, you must first close it by selecting from the menu:
File ▸ **Close Workspace**

Then, open Temperature Monitor project by selecting from the menu:
File ▸ **Open Workspace...** and opening the workspace file located at:

C:\RTC\COM3101\Lab4*.mcw

Select the workspace related to your development tool:

TempMon - C18 - Low Pin Count USB Development Kit.mcw	for the Low Pin Count USB Development Kit
TempMon - C18 - PICDEM FSUSB.mcw	for the PICDEM FS USB Board
TempMon - C30.mcw	for the Explorer 16 + PIC24F USB PIM
TempMon - C32.mcw	for the Explorer 16 + PIC32 USB PIM



2 Select **Release** mode.

3 Click on the **Build All** button.



4 Program the device using **MPLAB Real ICE™/ MPLAB ICD 3/ PICkit™ 3**

1. **Programmer** ▸ **Select Programmer** ▸ **REAL ICE/MPLAB ICD 3/PICkit 3**
2. **Programmer** ▸ **Program** or click on 

5 Connect the USB-to-RS232 adapter to an available USB port (if not already done)
Connect the board to the USB-to-RS232 adapter (if not already done)
Select the correct .inf file if required by **Windows®** (Found New Hardware Wizard)

6 Launch **Tera Term** and select the **Virtual COM Port** assigned to your USB-to-RS232 adapter. Configure the port with the following settings:
Setup ▸ Serial port...

- Baud Rate = ***57600***
- Data bits = ***8***
- Parity = ***None***
- Stop bits = ***1***
- Flow control = ***None***

7 Send the temperature string to **Tera Term**:

Low Pin Count USB Development Kit : press Switch S1
PICDEM FS USB: press Switch S2
Explorer 16: press Switch S3

8 After pressing the switch, you will see



Procedure—Migrating to the CDC Framework

9 Close HyperTerminal

If you still have the previous project open, you must first close it by selecting from the menu:
File ▸ Close Workspace

Then, open the copy of Lab 1b project by selecting from the menu:
File ▸ Open Workspace... and opening the workspace file located at:

C:\RTC\COM3101\Lab4*.mcw

Select the workspace related to your development tool:

Lab 4 - C18 - Low Pin Count USB Development Kit.mcw	for the Low Pin Count USB Development Kit
Lab 4 - C18 - PICDEM FSUSB.mcw	for the PICDEM FS USB Board
Lab 4 - C30.mcw	for the Explorer 16 + PIC24F USB PIM
Lab 4 - C32.mcw	for the Explorer 16 + PIC32 USB PIM

10 Find function **ProcessIO()** in **main.c**
Remove the **switch** statement .

main.c

```
void ProcessIO(void)
{
    // User Application USB tasks
    if((USBDeviceState < CONFIGURED_STATE)||(USBSuspendControl==1)) return;

    switch (mLED_1)
    {
        case 1:
            if (sw) mLED_1=0;
            break;
        case 0:
            if (!sw) mLED_1=1;
            break;
        default:
            mLED_1=0;
    }

    CDCTxService();
}//end ProcessIO
```

Remove this

11 Open the file **TempMon_main.c** by selecting from the menu:
File ▸ Open

C:\RTC\COM3101\Lab4\TempMon_main.c

12 Find function **main()** in **TempMon_main.c**
Copy and paste the content of the while loop into **ProcessIO()** in **main.c** where the **switch** was.

TempMon_main.c

```
void main(void)
{
  UserInit();

  while(1)
  {
    while(!SwitchIsPressed());
    AcquireTemperature(); // Perform temperature reading from
                          // sensor using "AcquireTemperature()"
    UpdateCelsiusASCII(); // Convert to ASCII string using
                          // "UpdateCelsiusASCII()"
    #if defined(__C30__)
      putsUART2((unsigned int*)tempString);
    #elif defined(__C32__)
      putsUART2(tempString);
    #else
      putsUSART(tempString);
    #endif
  }//end while

}//end main
```

Copy and paste these lines

13 Find the Include section in **main.c**
Then, include the **temperature.h** header file into **main.c**

main.c

```
/** INCLUDES *******************************************************/
#include "Compiler.h"
#include "HardwareProfile.h"
#include "USB/usb.h"
#include "USB/usb_function_cdc.h"

/** CONFIGURATION **************************************************/
```

← Include temperature.h here

14 Find function **UserInit()** in **main.c**
Then, call function **InitTempSensor()** from **UserInit()**

main.c

```
void UserInit(void)
{
        mInitAllLEDs();
        mInitAllSwitches();
                                    ← Call InitTempSensor() here
}//end UserInit
```

15 Add file **temperature.h** and **temperature.c** to the project



If you are using the **Low Pin Count USB Development Kit**, you also need to add the files:

- **SoftwareSPI.c**
- **SoftwareSPI.h**

16 Remove all blocking code in function **ProcessIO()** and send **tempString[]** to the Terminal via **USB**

Task 4.16 Reference Information

`SwitchIsPressed()` Function Prototype

```
BOOL SwitchIsPressed(void);
```

- Returns 1 if switch sw was pressed
- Returns 0 if switch sw was not pressed

`USBUSARTIsTxTrfReady()` Macro

```
BOOL USBUSARTIsTxTrfReady(void);
```

- Returns 1 when cdc_trf_state is CDC_TX_READY
- Returns 0 when cdc_trf_state is not CDC_TX_READY

`putsUSBUSART()` Function Prototype

```
void putsUSBUSART(char *data);
```

- **data** is a pointer to string stored in RAM. Usage examples:
  - ```
    char Welcome[]="Hello";
    putsUSBUSART(Welcome);
    ```



17 Select **Release** mode.

18 Click on the **Build All** button.



19 Program the device using **MPLAB Real ICE™/ MPLAB ICD 3/ PICkit™ 3**

1. **Programmer** ▸ **Select Programmer** ▸ **REAL ICE/MPLAB ICD 3/PICkit 3**
2. **Programmer** ▸ **Program** or click on 

If you are not using the AC164114 adapter, **DO NOT** program the Low Pin Count USB Development Kit while the USB cable is connected.
Disconnect the USB cable and program the PIC18F14K50 powering the target from PICkit 3 at 3.3V
    ```

20 Connect the board to an available USB port (if not already done).
Select the correct .inf file if required by **Windows®** (Found New Hardware Wizard)

C:\RTC\COM3101\USB Tools\USB CDC Serial Demo\inf\mchpcdc.inf

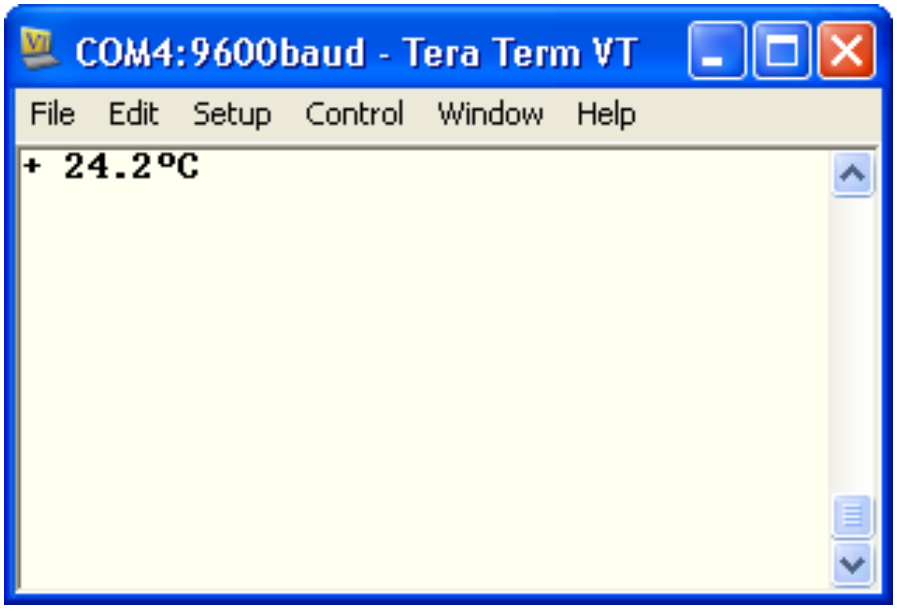
21 Launch **Tera Term** and select the **Virtual COM Port** assigned to your demo board.

22 Send the temperature string to **Tera Term**:

Low Pin Count USB Development Kit : press Switch S1
PICDEM FS USB: press Switch S2
Explorer 16: press Switch S3

Results

After pressing the switch sw on the demo board, you will see



Conclusions

This lab has shown you how to migrate an existing RS-232 UART application to USB.
The CDC class implemented on the device allows communication with HyperTerminal using a Virtual COM Port on the PC, so no Windows® programming is needed in order to send data from the device to the host.
You also saw how easy it can be to migrate an existing application using Microchip CDC API due to the fact that the CDC API function names are very similar to standard UART API function names.

Lab Exercise 5

Adding a Serial Number String Descriptor

Purpose

A USB CDC device provided with a serial number string descriptor prevents unwanted "COM port proliferation", meaning that it retains its COM port number if it is moved to a different USB port on a Windows® PC. A USB CDC device that doesn't contain a serial number gets a new port number on each attachment to a different port on a PC.
In this last lab, the attendee will learn how to modify string descriptors of a CDC device to add a Serial Number string.

Requirements

Development Environment:	MPLAB® IDE v8.40 or later
C Compiler:	MPLAB® C18 v3.30 or later, C30 v3.12 or later, C32 v1.05 or later
Hardware Tools:	PICDEM FS USB Board or Low Pin Count USB Development Kit + TC72 Digital Temperature Sensor PICtail Demo Board or Explorer 16 + USB PICTail Plus Daughter Board + (PIC24F USB PIM or PIC32 USB PIM) MPLAB Real ICE™, ICD 3 or PICkit™ 3
Lab files on class PC:	C:\RTC\COM3101\Lab5

Objective

Show how to add a USB serial number string descriptor to an existing USB device project.

Procedure

1

If you still have the previous project open, you must first close it by selecting from the menu:
File ▸ Close Workspace

Then, open Lab 5 project by selecting from the menu:
File ▸ Open Workspace... and opening the workspace file located at:

C:\RTC\COM3101\Lab5*.mcw

Select the workspace related to your development tool:

Lab 5 - C18 - Low Pin Count USB Development Kit.mcw	for the Low Pin Count USB Development Kit
Lab 5 - C18 - PICDEM FS USB.mcw	for the PICDEM FS USB Board
Lab 5 - C30.mcw	for the Explorer 16 + PIC24F USB PIM
Lab 5 - C32.mcw	for the Explorer 16 + PIC32 USB PIM

2

Find “Device Descriptor” section in **usb_descriptors.c**
Then, change the index value of serial number string to 3

usb_descriptors.c

```
/* Device Descriptor */
ROM USB_DEVICE_DESCRIPTOR device_dsc=
{
    0x12,                   // Size of this descriptor in bytes
    USB_DESCRIPTOR_DEVICE,  // DEVICE descriptor type
    0x0200,                 // USB Spec Release Number in BCD format
    CDC_DEVICE,             // Class Code
    0x00,                   // Subclass code
    0x00,                   // Protocol code
    USB_EP0_BUFF_SIZE,      // Max packet size for EP0, see usb_config.h
    MY_VID,                 // Vendor ID
    MY_PID,                 // Product ID: CDC RS-232 Emulation Demo
    0x0001,                 // Device release number in BCD format
    0x01,                   // Manufacturer string index
    0x02,                   // Product string index
    0x00,  <- Change this   // Device serial number string index
    0x01                    // Number of possible configurations
};
```

3 Find the Product string descriptor in **usb_descriptors.c:**

usb_descriptors.c

```
//Product string descriptor
ROM struct{BYTE bLength;BYTE bDscType;WORD string[16];}sd002={
sizeof(sd002),USB_DESCRIPTOR_STRING,
{'L','a','b',' ','5',' ','-',' '
'C','O','M',' ','3','1','0','1'}
};

//Array of configuration descriptors
```

← **Add serial string here**

Then, add the following serial string descriptor immediately below it

```
ROM struct {BYTE bLength;BYTE bDscType;WORD string[4];} sd003={
sizeof(sd003),USB_DESCRIPTOR_STRING,
{'1', '2', '3', '4'}
};
```

4 Find the string descriptor address array in **usb_descriptors.c**
Add a comma after **&sd002**, followed by: **(ROM BYTE *ROM)&sd003**

usb_descriptors.c

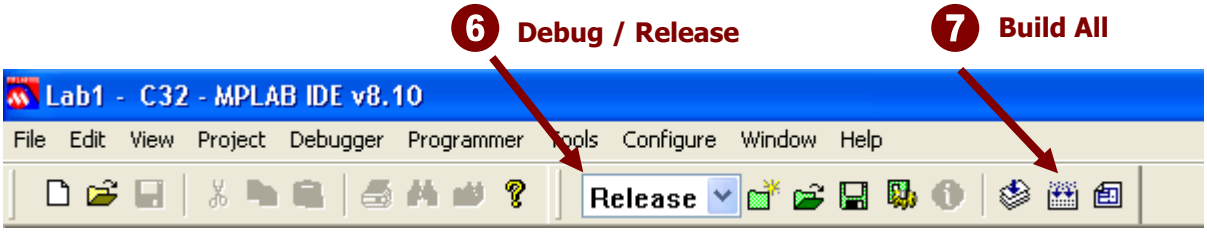
```
//Array of string descriptors
ROM BYTE *ROM USB_SD_Ptr[]=
{
    (ROM BYTE *ROM)&sd000,
    (ROM BYTE *ROM)&sd001,
    (ROM BYTE *ROM)&sd002
};
```

← **Add the new item here**

5 Find the define **USB_NUM_STRING_DESCRIPTORS** in **usb_config.h**
Then, change the value to 4

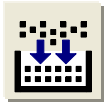
usb_config.h

```
// Peripheral Configuration

#define MY_VID                          0x04D8
#define MY_PID                          0x000A
#define USB_SPEED_OPTION                USB_FULL_SPEED
#define USB_INTERRUPT
#define USB_PULLUP_OPTION               USB_PULLUP_ENABLE
#define USB_TRANSCEIVER_OPTION          USB_INTERNAL_TRANSCEIVER
#define USB_EP0_BUFF_SIZE               8
#define USB_MAX_NUM_INT                 (0+1)
#define USB_MAX_EP_NUMBER               3
#define USB_NUM_STRING_DESCRIPTORS      3   <- Change the value here

#define USB_ENABLE_ALL_HANDLERS
```



6 Select **Release** mode.

7 Click on the **Build All** button.



8 Program the device using **MPLAB Real ICE™/ MPLAB ICD 3/ PICkit™ 3**

1. **Programmer** ▸ **Select Programmer** ▸ **REAL ICE/MPLAB ICD 3/PICkit 3**
2. **Programmer** ▸ **Program** or click on 

If you are not using the AC164114 adapter, **DO NOT** program the Low Pin Count USB Development Kit while the USB cable is connected.
Disconnect the USB cable and program the PIC18F14K50 powering the target from PICkit 3 at 3.3V

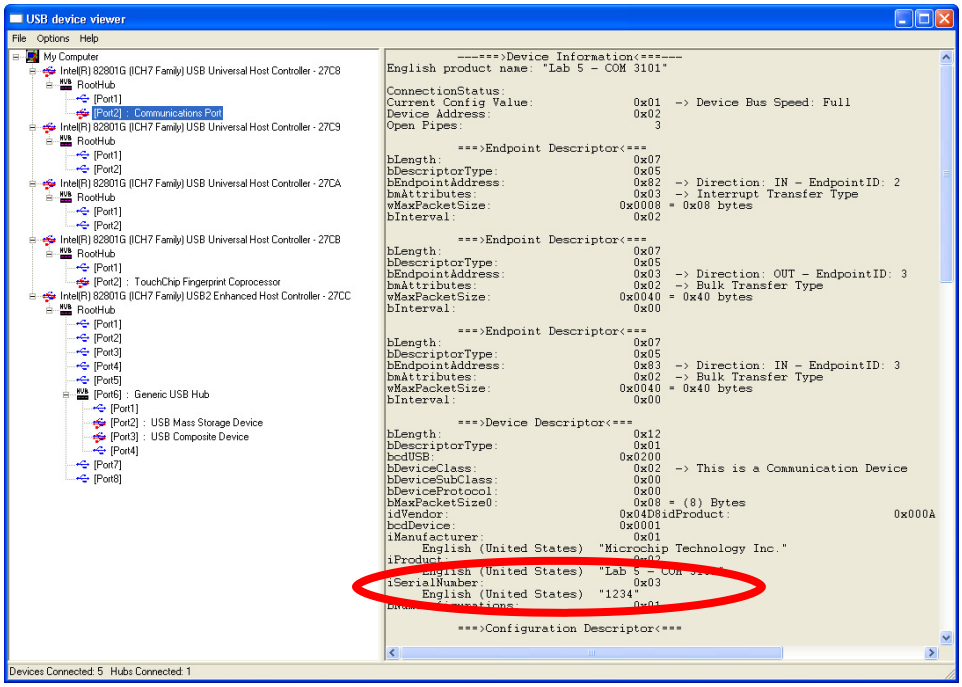
9 Connect the board to an available USB port (if not already done).
Select the correct .inf file if required by **Windows**® (Found New Hardware Wizard)

C:\RTC\COM3101\USB Tools\USB CDC Serial Demo\inf\mchpcdc.inf

10 Launch **USB Device Viewer**
Start ▸ **All Programs** ▸ **Microchip** ▸ **COM3101** ▸ **USB Device Viewer**

Results

After clicking on the right device instance, you will see



Conclusions

This lab has shown you how to add a serial number string descriptor to a USB device project.
You modified the USB descriptors in the CDC project in order to add a new string which is used as serial number to avoid COM Port Proliferation on the PC. Serial number string descriptors enable PC applications to distinguish between multiple devices having identical VID/PID.

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix A

Removing Existing MCHPFSUSB Framework Drivers and .inf Files

Objective

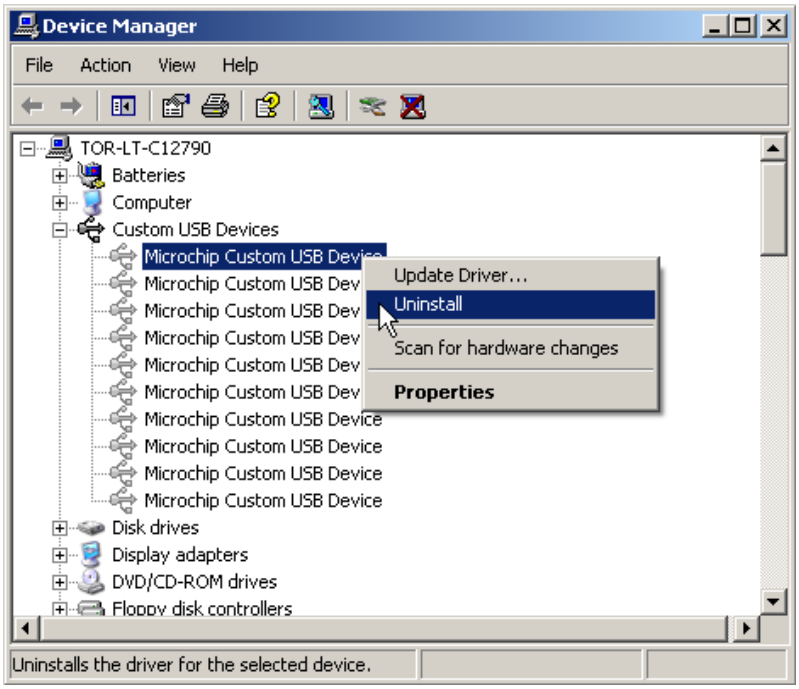
To get an authentic “out-of-the-box” experience each time you run through this class (or before using a newer MCHPFSUSB release), you should un-install all .inf and .sys files.

Procedure

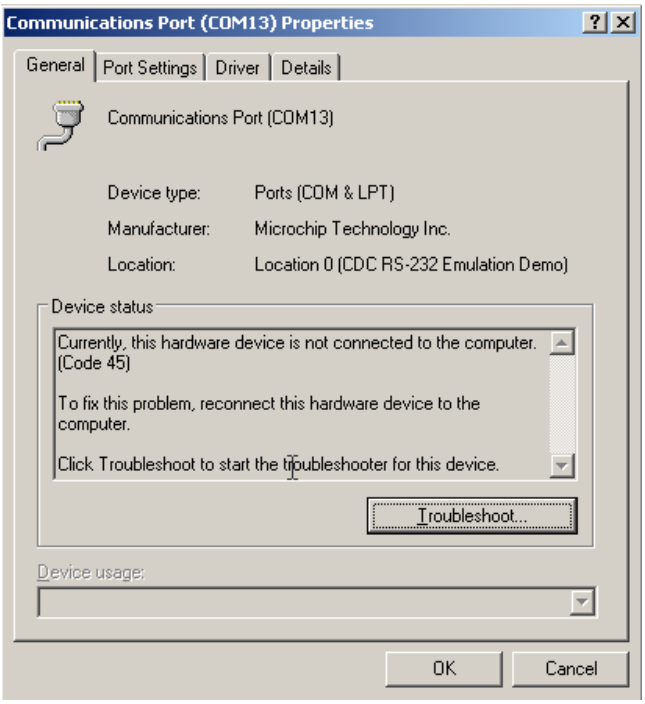
1. Launch **Windows® Device Manager Console**
   Start ▸ **All Programs** ▸ **Microchip** ▸ **COM3101** ▸ **Super Device Manager**

2. In the **Device Manager Console**, select:
   View ▸ **Show hidden devices**

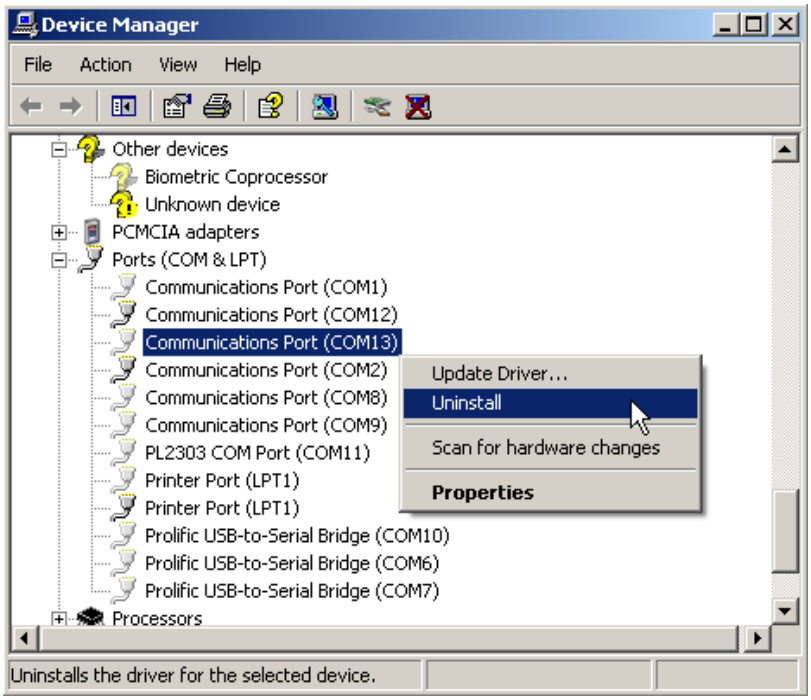
3 The installed custom class drivers are represented by the USB Trident symbol ( ) and appear under **Custom USB Devices** .
Right-click/Uninstall all of the **Custom USB Devices** ▸ **Microchip Custom USB Device** as shown:



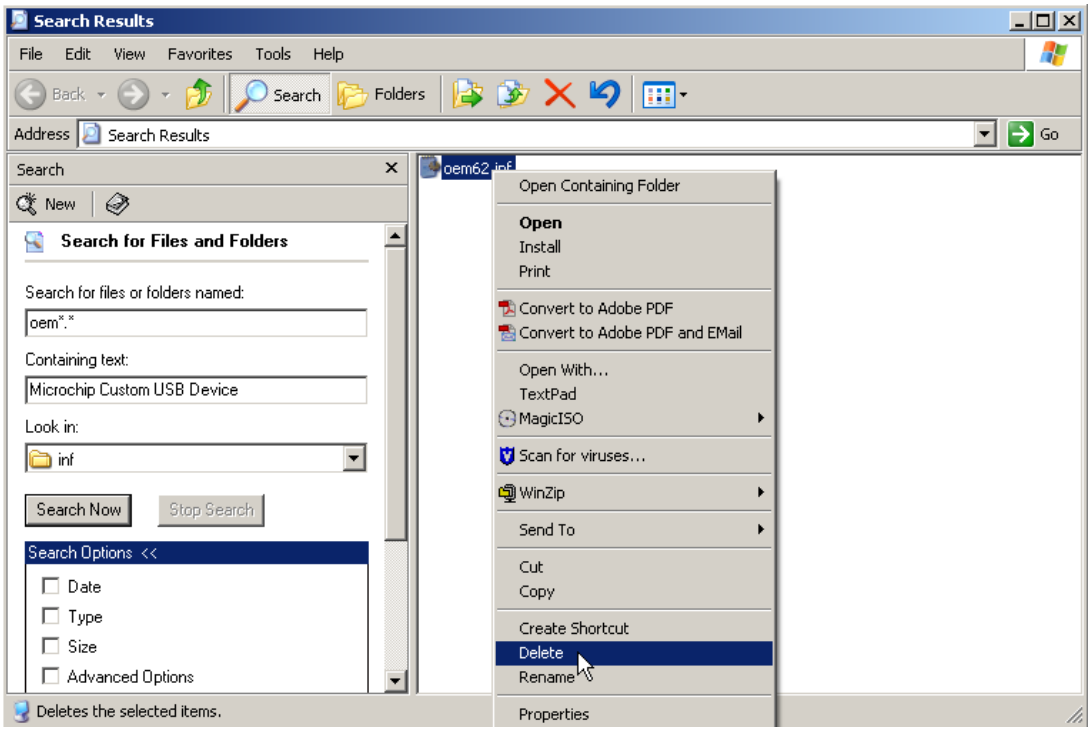
4 The installed **CDC** drivers are represented by the Serial Cable symbol ( ) and appear under **Ports (COM & LPT)**. The choice of **Communication Port** as port name by the Factory USB team is unfortunate, in that it also refers to physical RS-232 COM port drivers, which we **DO NOT** wish to delete. Right-click on each instance of **Communication Port**, select **Properties** and verify that it is the emulated COM port **Microchip CDC RS-232 Emulation Demo**:



Now it is safe to right-click/Un-install this instance (COM13 in this example) of **Ports (COM & LPT)** ▸ **Communications Port (COM13)** as shown:



5 Open the **Windows® Search** tool:
Start ▸ **Search**
Search for files/folders named **oem*.***, containing the text **Microchip Custom USB Device** (used in **mchpusb.inf**) in **C:\windows\inf**. Now delete all of the **oemxx.inf** files found by the search tool. Repeat for CDC .inf files (**mchpcdc.inf**) by searching for files containing the text **USB RS-232 Emulation Driver**



THIS PAGE INTENTIONALLY LEFT BLANK