Lab 03 - USB

S.M.B.G. Janakantha

E/20/157

CO326 – Industrial Networks

2026-01-24

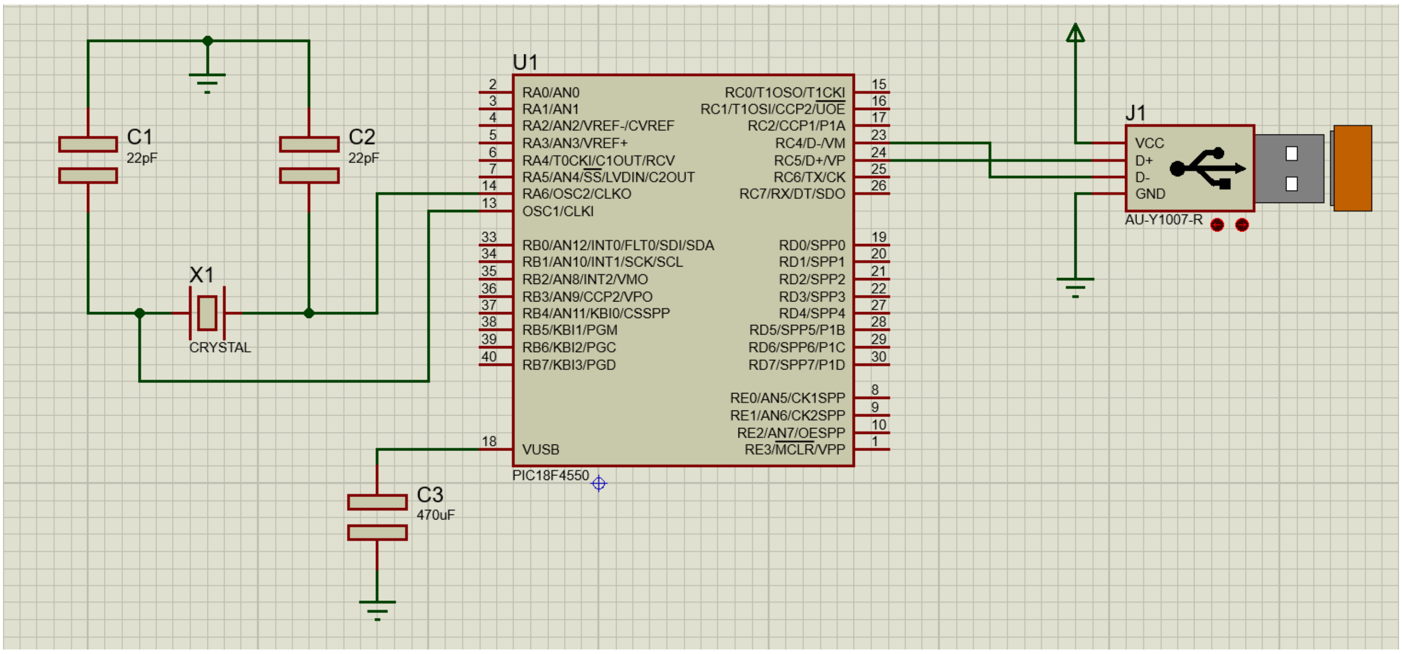
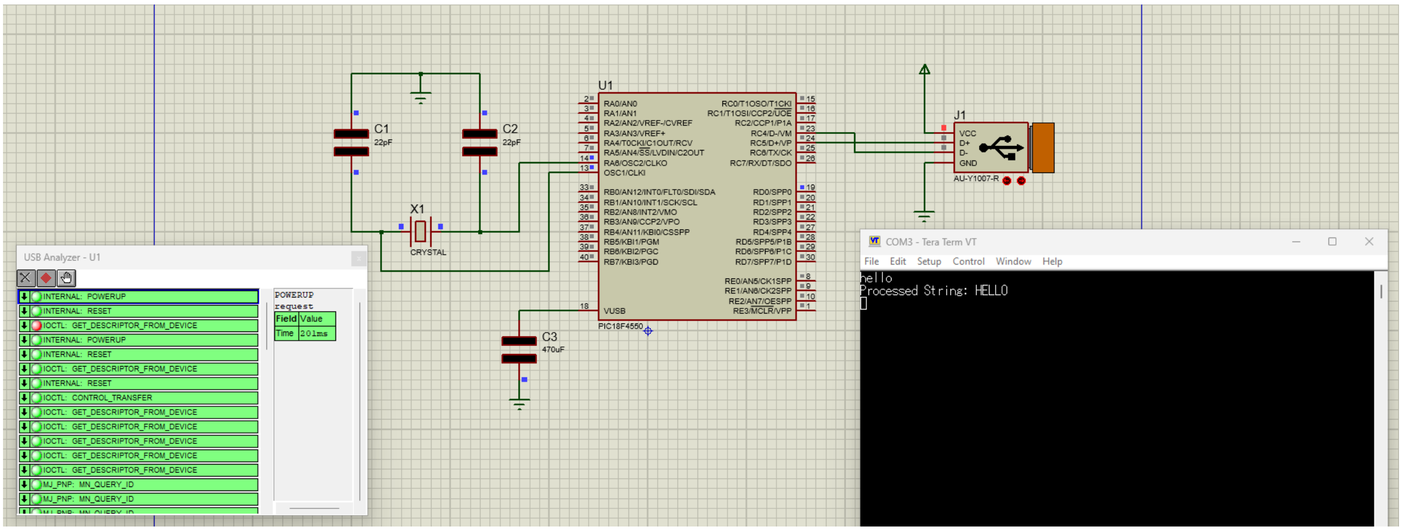


Figure 1: Proteus Setup



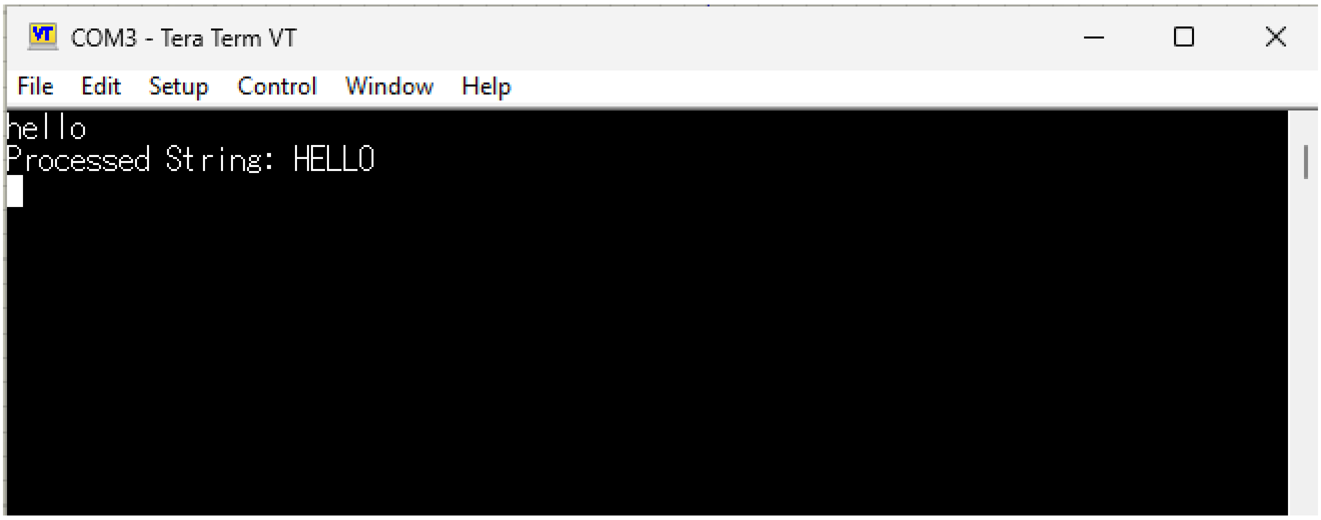
```
INTERNAL:  POWERUP
INTERNAL:  RESET
IOCTL:  GET_DESCRIPTOR_FROM_DEVICE
INTERNAL:  POWERUP
INTERNAL:  RESET
IOCTL:  GET_DESCRIPTOR_FROM_DEVICE
INTERNAL:  RESET
IOCTL:  CONTROL_TRANSFER
IOCTL:  GET_DESCRIPTOR_FROM_DEVICE
IOCTL:  GET_DESCRIPTOR_FROM_DEVICE
IOCTL:  GET_DESCRIPTOR_FROM_DEVICE
IOCTL:  GET_DESCRIPTOR_FROM_DEVICE
IOCTL:  GET_DESCRIPTOR_FROM_DEVICE
MJ_PNP:  MN_QUERY_ID
MJ_PNP:  MN_QUERY_ID
```

POWERUP
request

Field	Value
Time	201ms

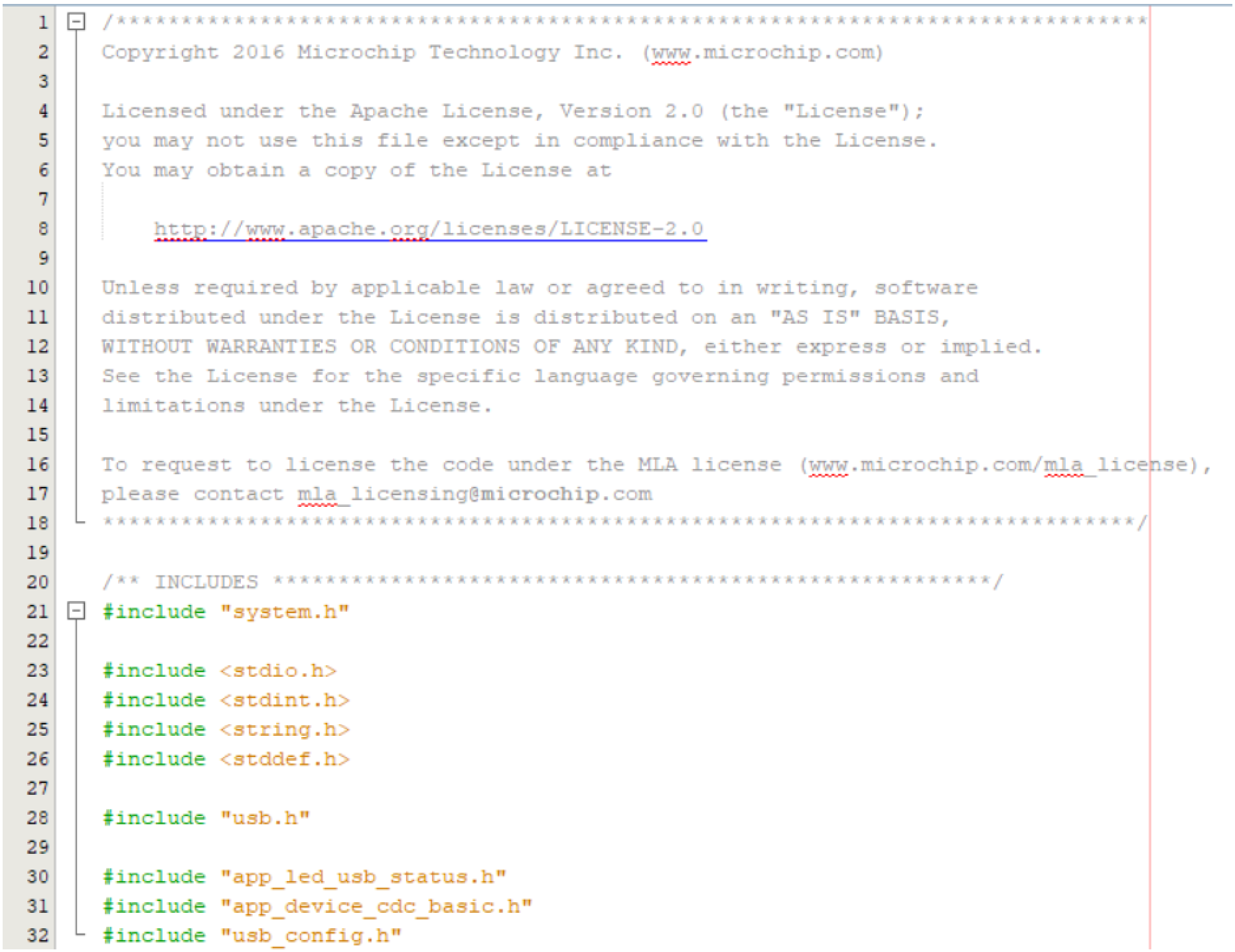
```
hello
Processed String: HELLO
```

Figure 2: Working Setup

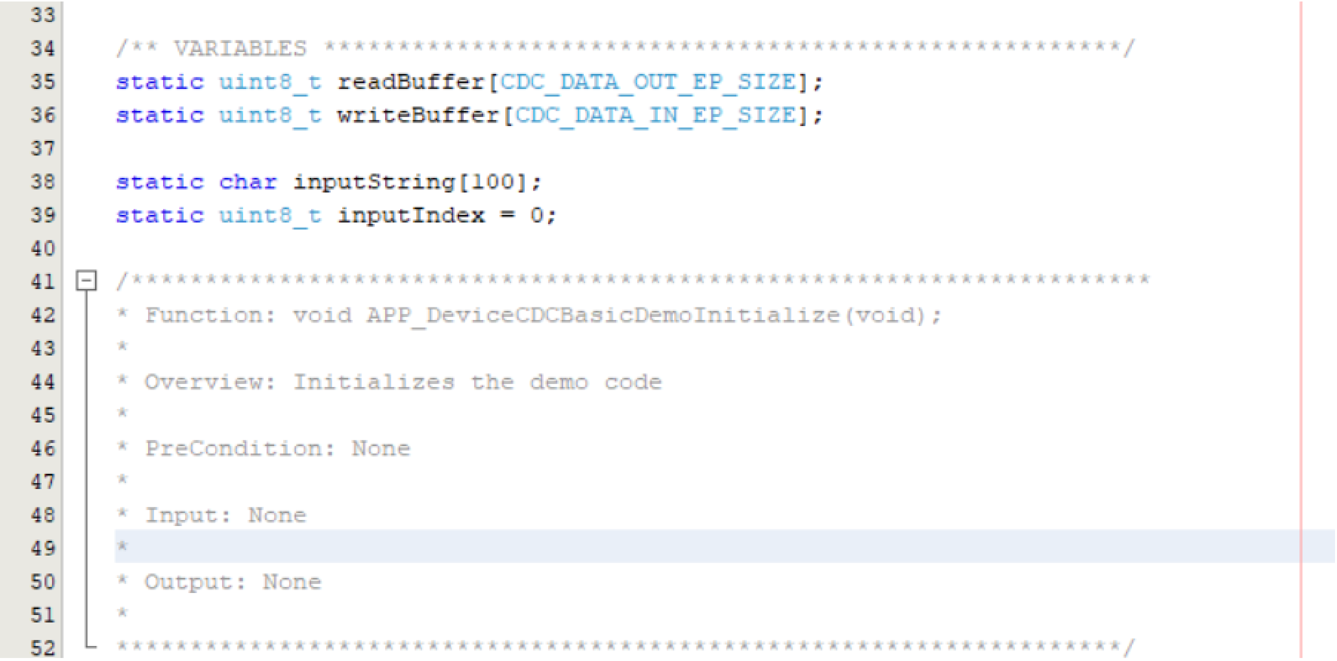


```
hello
Processed String: HELLO
```

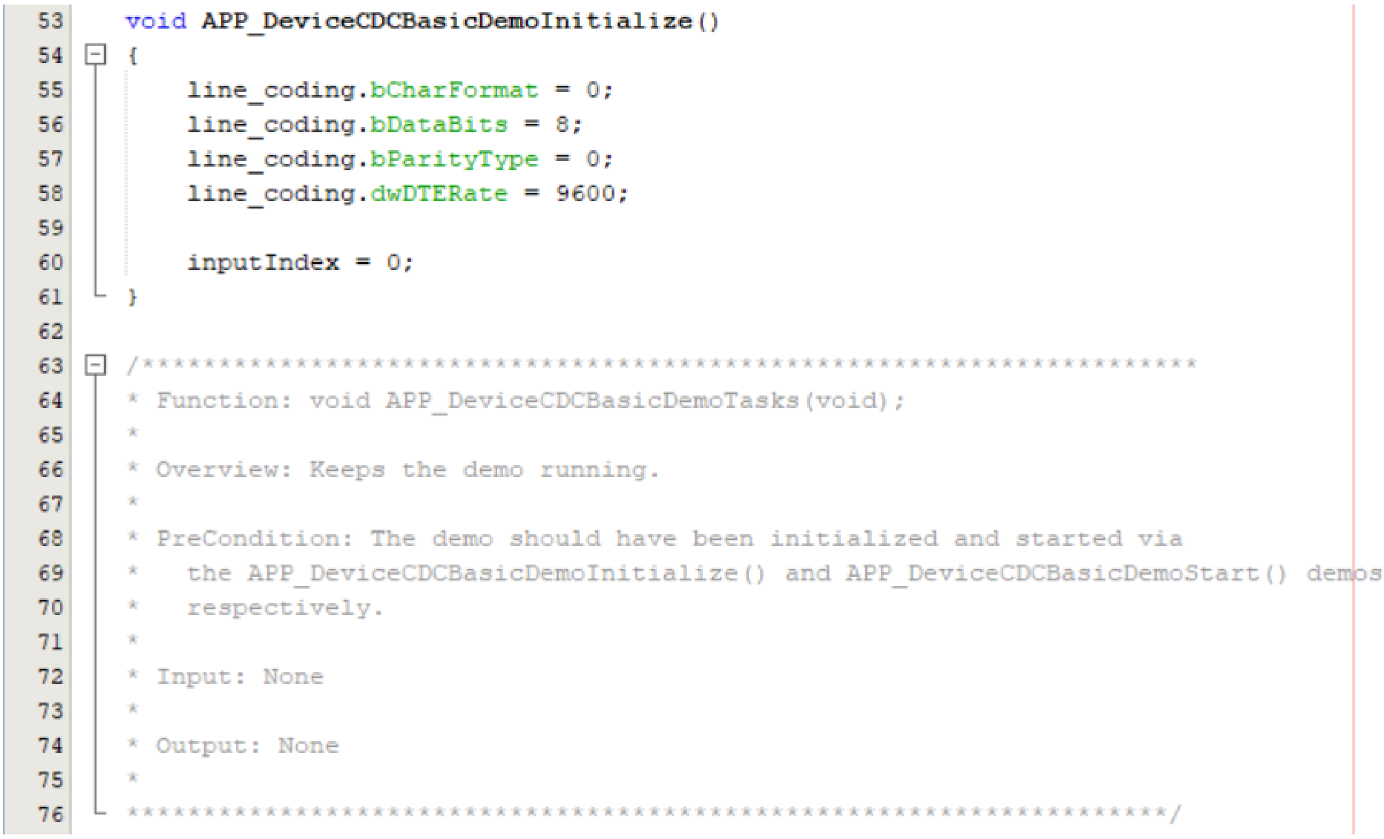
Figure 3: Tera Term Terminal



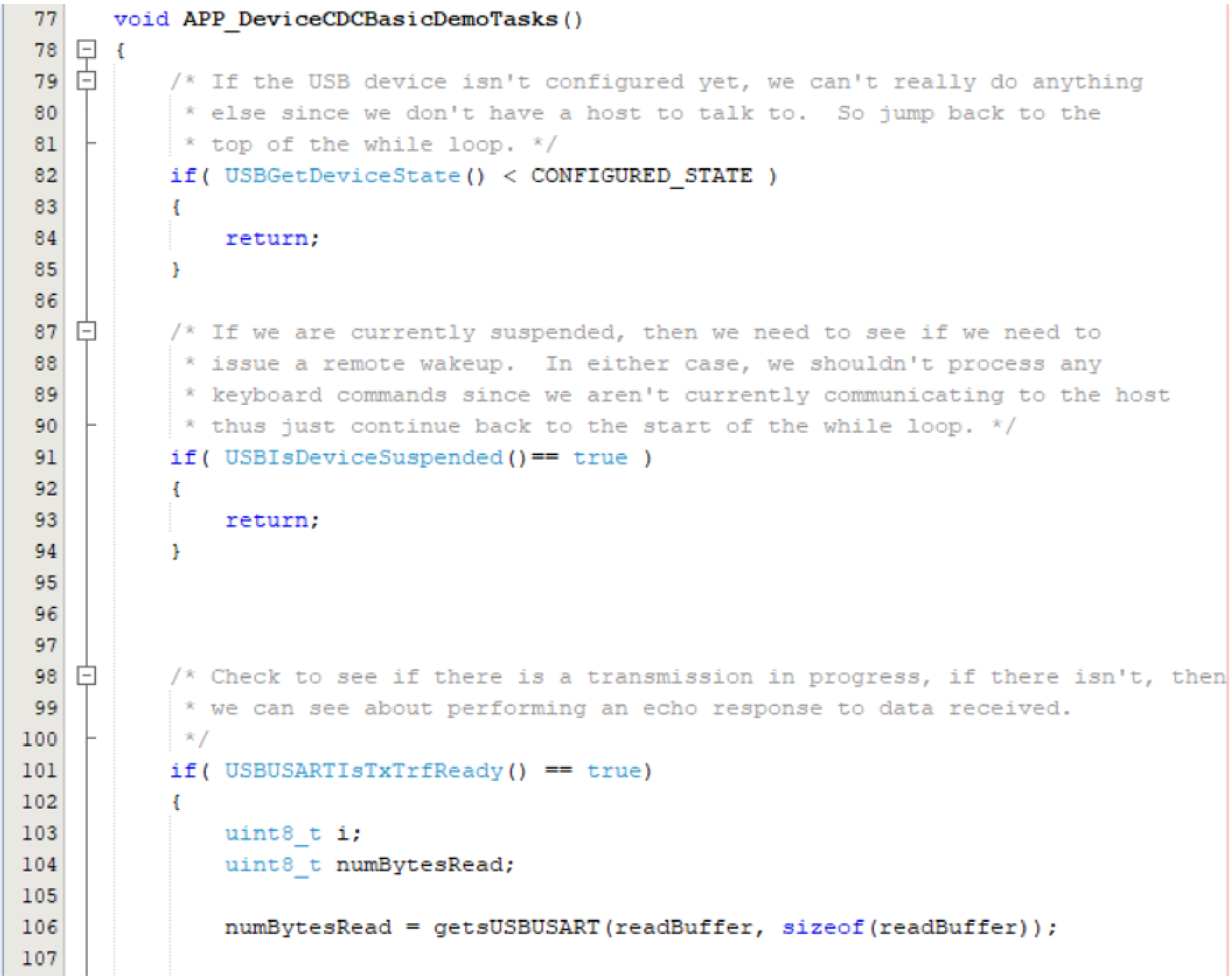
```
/*******************************************************************************
Copyright 2016 Microchip Technology Inc. (www.microchip.com)

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 <stdio.h>
#include <stdint.h>
#include <string.h>
#include <stddef.h>

#include "usb.h"

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



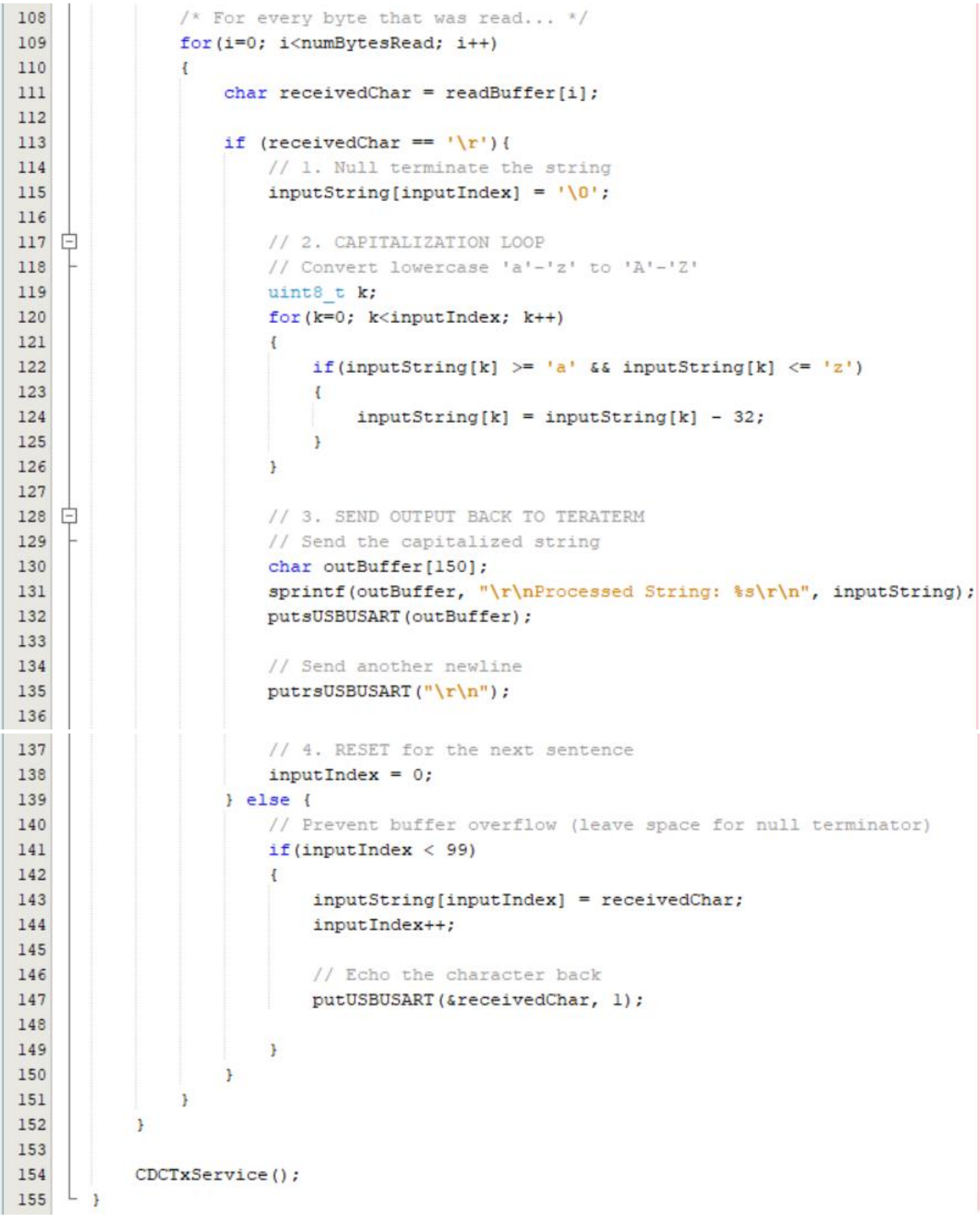
```

/** VARIABLES ******************************************************/
static uint8_t readBuffer[CDC_DATA_OUT_EP_SIZE];
static uint8_t writeBuffer[CDC_DATA_IN_EP_SIZE];

static char inputString[100];
static uint8_t inputIndex = 0;

/*********************************************************************
* Function: void APP_DeviceCDCBasicDemoInitialize(void);
*
* Overview: Initializes the demo code
*
* PreCondition: None
*
* Input: None
*
* Output: None
*
********************************************************************/
```



```c
void APP_DeviceCDCBasicDemoInitialize()
{
    line_coding.bCharFormat = 0;
    line_coding.bDataBits = 8;
    line_coding.bParityType = 0;
    line_coding.dwDTERate = 9600;

    inputIndex = 0;
}

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



```c
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;
    }



    /* 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));

```



```c
        /* For every byte that was read... */
        for(i=0; i<numBytesRead; i++)
        {
            char receivedChar = readBuffer[i];

            if (receivedChar == '\r'){
                // 1. Null terminate the string
                inputString[inputIndex] = '\0';

                // 2. CAPITALIZATION LOOP
                // Convert lowercase 'a'-'z' to 'A'-'Z'
                uint8_t k;
                for(k=0; k<inputIndex; k++)
                {
                    if(inputString[k] >= 'a' && inputString[k] <= 'z')
                    {
                        inputString[k] = inputString[k] - 32;
                    }
                }

                // 3. SEND OUTPUT BACK TO TERATERM
                // Send the capitalized string
                char outBuffer[150];
                sprintf(outBuffer, "\r\nProcessed String: %s\r\n", inputString);
                putsUSBUSART(outBuffer);

                // Send another newline
                putrsUSBUSART("\r\n");

                // 4. RESET for the next sentence
                inputIndex = 0;
            } else {
                // Prevent buffer overflow (leave space for null terminator)
                if(inputIndex < 99)
                {
                    inputString[inputIndex] = receivedChar;
                    inputIndex++;

                    // Echo the character back
                    putUSBUSART(&receivedChar, 1);

                }
            }
        }
    }

    CDCTxService();
}
```

Figure 4: MPLAB Code : app_device_cdc_basic.c