M480 Ethernet to USB Host Bridge

Example Code Introduction for 32-bit NuMicro® Family

Information

Application	This sample code implements a bridge function between Ethernet port and USB host port in the M487 microcontroller (MCU).
BSP Version	M480 Series BSP CMSIS V3.05.001
Hardware	NuMaker-PFM-M487 v3.0

The information described in this document is the exclusive intellectual property of Nuvoton Technology Corporation and shall not be reproduced without permission from Nuvoton.

Nuvoton is providing this document only for reference purposes of NuMicro microcontroller and microprocessor based system design. Nuvoton assumes no responsibility for errors or omissions.

All data and specifications are subject to change without notice.

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

1 Overview

This example code implements a USB host to Ethernet bridge function. The function is a part of the Ethernet connection solution. The solution utilizes USB host and device peripherals of M487 microcontroller (MCU) to implement Ethernet frame transfer function between two PC sites. A PC site can send Ethernet frames to the other PC site without considering factors of USB packets. Ethernet frames can be converted into USB packet format automatically by M487 MCU. Then USB packets are converted into Ethernet frames again and received by the other PC site. Figure 1-1 shows the architecture of the solution. This example code implements the part of the red rectangle and red arrows; green circles mean the conversion between USB data and Ethernet data.

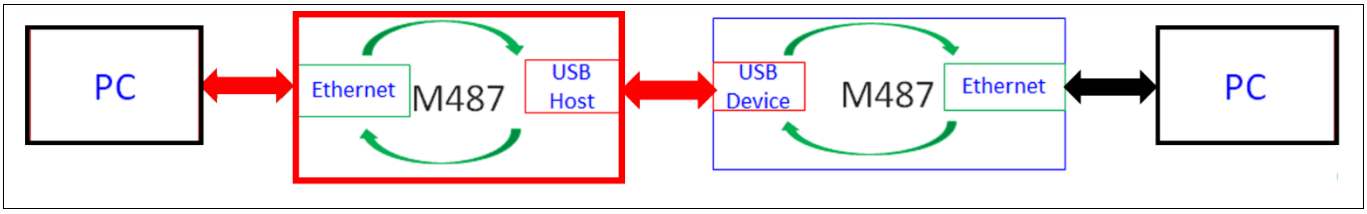

Figure 1-1 Architecture of Ethernet Connection

1.1 Network Connection

The IP addresses of PCs in the network connection solution need to be set in the same networking domain. Figure 1-2 shows an example of IP addresses configurations in the solution. The IP addresses of the two PCs are 192.168.10.200 and 192.168.10.150 respectively.

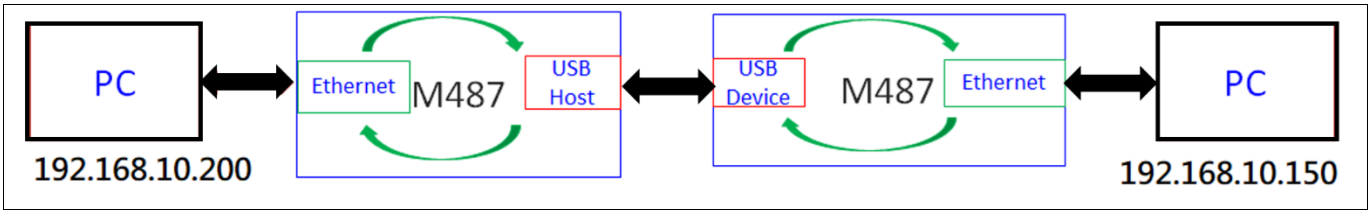

Figure 1-2 IP Addresses Example of the Ethernet Connection

1.2 USB Enumeration

In the Ethernet solution, the two PFM-M487 evaluation (EV) boards are connected through USB host and device interfaces. One of the PFM-M487 EV boards is configured to be a USB host that can deal with the enumeration of USB HID class devices. The other EV board is treated as a USB HID class device. In the enumeration phase, the USB host can recognize this USB HID class device and complete the enumerating configurations. Then the USB data transfer channel is ready for USB data transfer.

1.3 Packet Format

An Ethernet frame transferred from a PC site to the other PC site is shown in Figure 1-3. The

maximum length of the frame is 1518 Bytes.

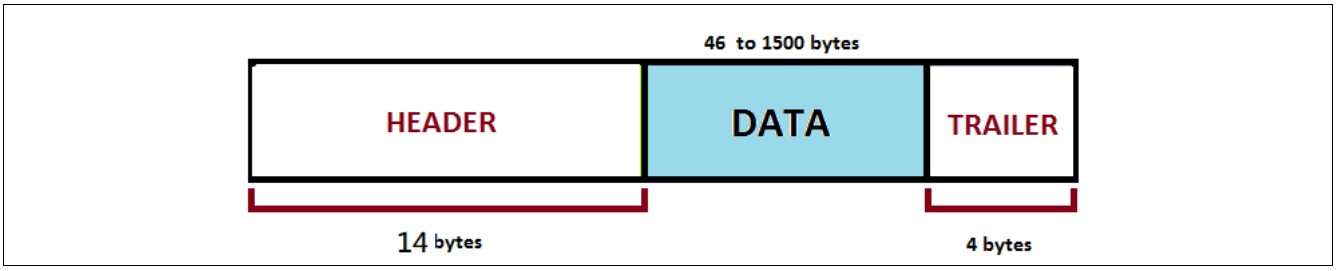

Figure 1-3 Ethernet Frame Structure

As long as the M487 receives an Ethernet frame, extra tags with actual frame length information will be appended to the frame. Figure 1-4 shows the updated frame.

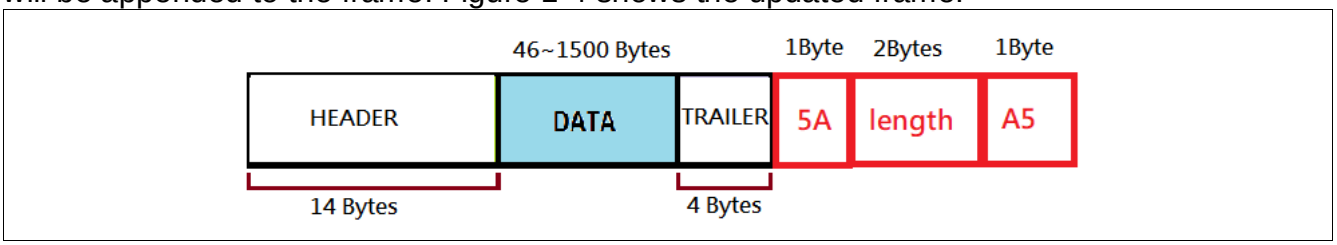

Figure 1-4 Ethernet Frame Structure with Extra Tags

Figure 1-5 presents an example to explain the data transfer flow. An Ethernet frame is divided into multiple USB packets that are sent from USB device side to USB host side. The USB host recombines these USB packets to be the frame with extra tags and checks whether the extra tags are received correctly or not. If the host receives the extra tags correctly, it will transfer the frame without extra tags through Ethernet port.

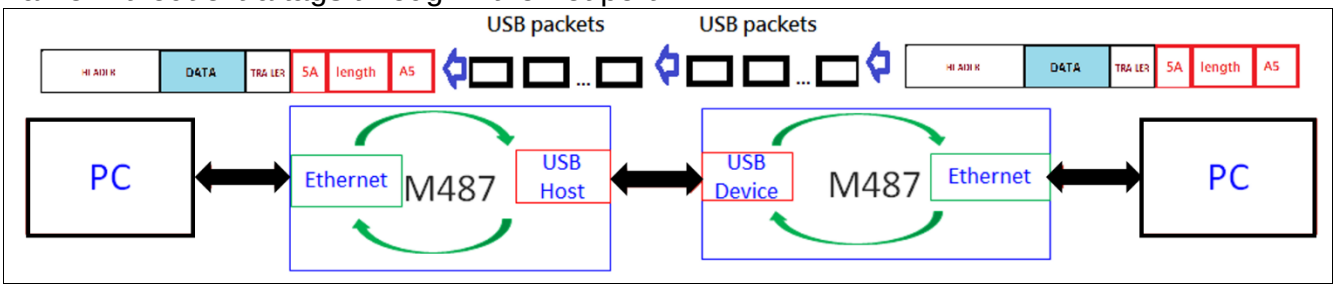

Figure 1-5 Data Transfer Example

The same method is applied to the data transfer from USB host side to USB device side.

1.4 Demo Result

In the demonstration, ping command is executed on both PC sites. Either of the PC sites can receive ping information from the other PC site. Figure 1-6 shows the result of ping execution.

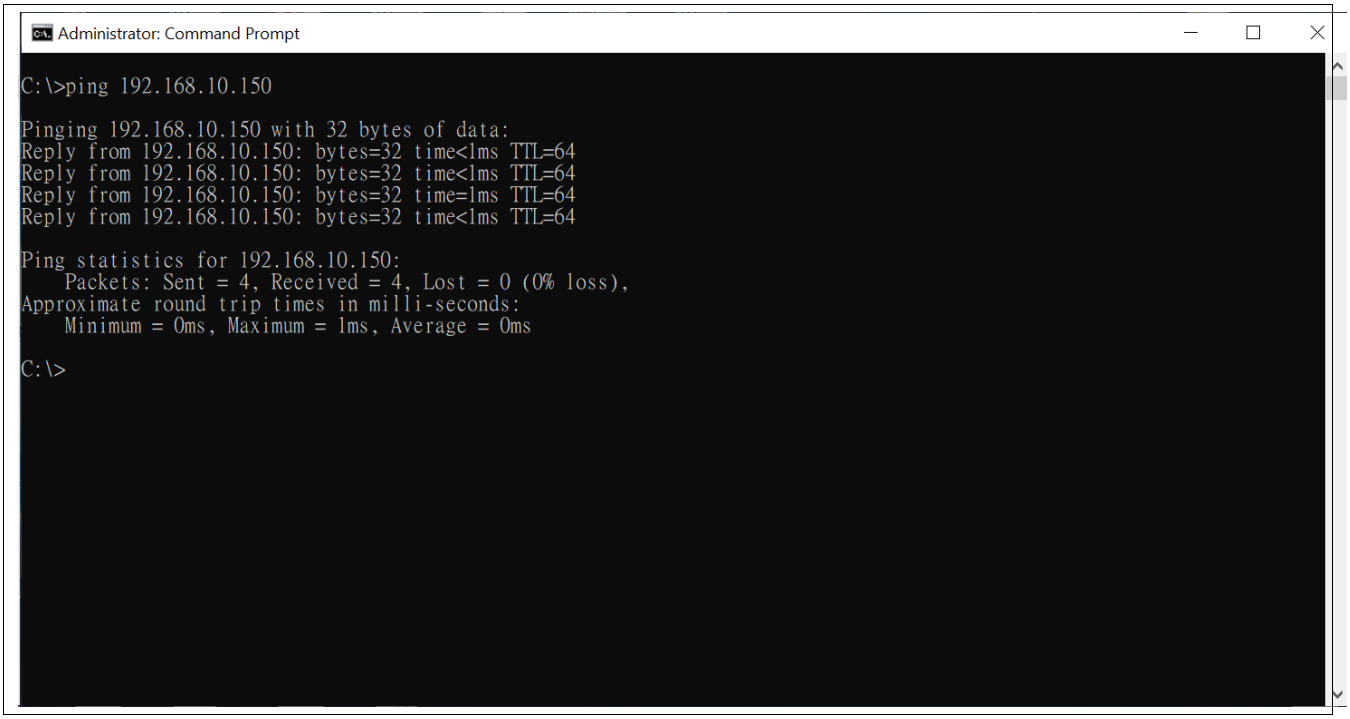

Figure 1-6 Ping Command Result

The demo result is shown in Figure 1-7.

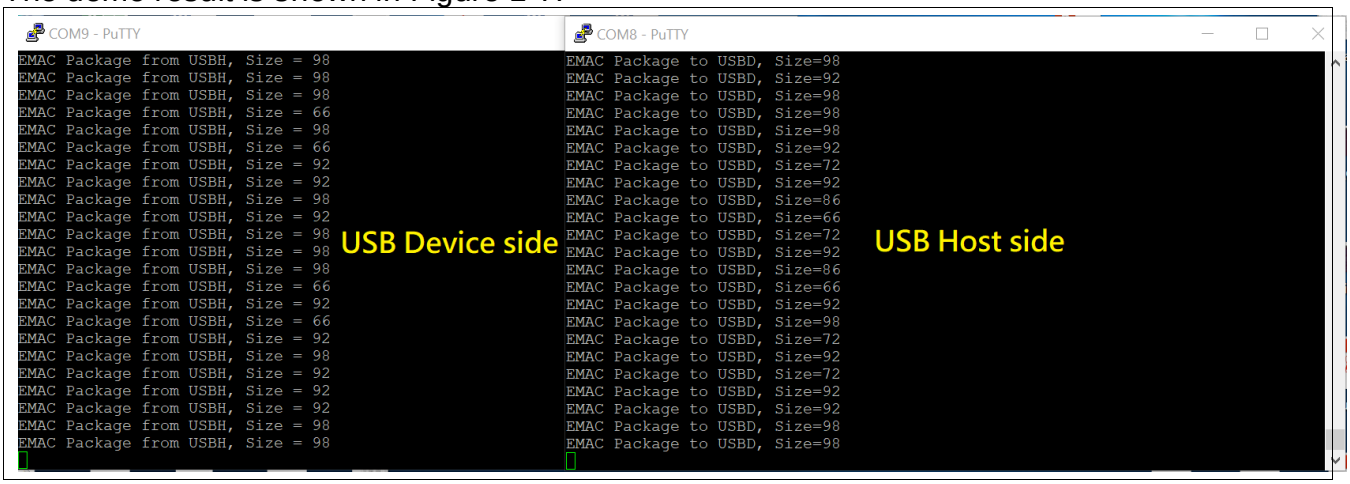

Figure 1-7 Demo Result

2 Code Description

First, initialize M480 in main.c, and initialize Ethernet RMII interface, debug port, and USB Host. Then, detect newly added HID devices.

```
int32_t main(void)
{
    HID_DEV_T    *hdev, *hdev_list;
    uint32_t     t0;

    /* Init System, IP clock and multi-function I/O */
    SYS_Init();
    /* Initialize UART0 */
    UART0_Init();

    enable_sys_tick(100);

    printf("+-----------------------------------------+\n");
    printf("|  USB Host to Ethernet Bridge sample code  |\n");
    printf("+-----------------------------------------+\n");


    /* Select RMII interface by default */
    EMAC_Open(g_au8MacAddr);

    /* Init phy */
    EMAC_PhyInit();

    NVIC_EnableIRQ(EMAC_TX_IRQn);
    NVIC_EnableIRQ(EMAC_RX_IRQn);

    EMAC_ENABLE_RX();
    EMAC_ENABLE_TX();

    /* Initialize USB host controller */
    usbh_core_init();
    usbh_hid_init();
    usbh_memory_used();

    memset(g_hid_list, 0, sizeof(g_hid_list));
    t0 = g_tick_cnt;

    while (1)
    {
        /* USB Host port detect polling and management */
        if (usbh_pooling_hubs())
        {
            /* print out USB memory allocating information */
            usbh_memory_used();

            printf("\n Has hub events.\n");
            hdev_list = usbh_hid_get_device_list();
            hdev = hdev_list;

            while (hdev != NULL)
            {
                if (is_a_new_hid_device(hdev))
                {
                    init_hid_device(hdev);
                    g_hdev = hdev;
                }

                hdev = hdev->next;
```

```
            }

            update_hid_device_list(hdev_list);
            usbh_memory_used();
        }

        if (g_tick_cnt - t0 >= 100)
        {
            t0 = g_tick_cnt;
            int_cnt = 0;
        }
    }
}
```

The int_read_callback function combines the USB packets received from USB host into a frame and then sends it to Ethernet port.

```
void  int_read_callback(HID_DEV_T *hdev, uint16_t ep_addr, int status, uint8_t *rdata,
uint32_t data_len)
{
    /* USB host HID driver notify user the transfer status via <status> parameter.*/
    if (status < 0)
    {
        printf("Interrupt in transfer failed! status: %d\n", status);
        return;
    }

    int_cnt++;

    if (data_len > 0)
    {

        if ((g_host_in_buf_len + data_len) >= 1522) g_host_in_buf_len = 0;

        HSUSB_MemCopy(&g_host_in_buf[g_host_in_buf_len], rdata, data_len);

        g_host_in_buf_len += data_len;

    }

    /* Check package size tag */
    if (g_host_in_buf[g_host_in_buf_len - 1] == 0xA5)
    {
        if (g_host_in_buf[g_host_in_buf_len - 4] == 0x5A)
        {
            int g_pkg_size = g_host_in_buf[g_host_in_buf_len - 3] << 8 |
g_host_in_buf[g_host_in_buf_len - 2];
            /*Send received package from USBD to EMAC */
            printf("EMAC Package from USBD, Size = %d\n", g_pkg_size);
            EMAC_SendPkt(g_host_in_buf, g_pkg_size);
            g_host_in_buf_len = 0;
        }
    }
}
```

The EMAC_RX_IRQHandler function adds extra tags to the frame received from Ethernet port, and then sends it to USB host.

```
void EMAC_RX_IRQHandler(void)
{
    while (1)
    {
        /* Check if there's any packets available */
        if (EMAC_RecvPkt(auPkt, &u32PktLen) == 0)
            break;
```

```
        /*Receive  data from PC and retransfer to USBD */
        printf("EMAC Package to USBD, Size=%d\n", u32PktLen);

        /* Store raw data buffer */
        g_pkg_len = u32PktLen;
        g_EMAC_res_len = u32PktLen;
        memcpy(g_EMAC_buf, auPkt, g_pkg_len);

        /* Add extra package size tag */
        g_EMAC_buf[g_pkg_len] = 0x5A;
        g_EMAC_buf[g_pkg_len + 1] = (g_pkg_len >> 8) & 0xFF;
        g_EMAC_buf[g_pkg_len + 2] = g_pkg_len & 0xFF;
        g_EMAC_buf[g_pkg_len + 3] = 0xA5;

        g_pkg_len += 4;
        g_EMAC_res_len += 4;

        /* Start USB transfer */
        if (!g_usb_wtn_en_flag && g_usb_ready)
        {
            g_usb_wtn_en_flag = TRUE;
        }

        /* Clean up Rx resource occupied by previous received packet */
        EMAC_RecvPktDone();
    }
}
```

3 Software and Hardware Requirements

3.1 Software Requirements

- BSP version
  - M480 Series BSP CMSIS V3.05.001
- IDE version
  - Keil uVersion 5.26

3.2 Hardware Requirements

- Circuit components
  - NuMaker-PFM-M487 v3.0

- Pin Connect

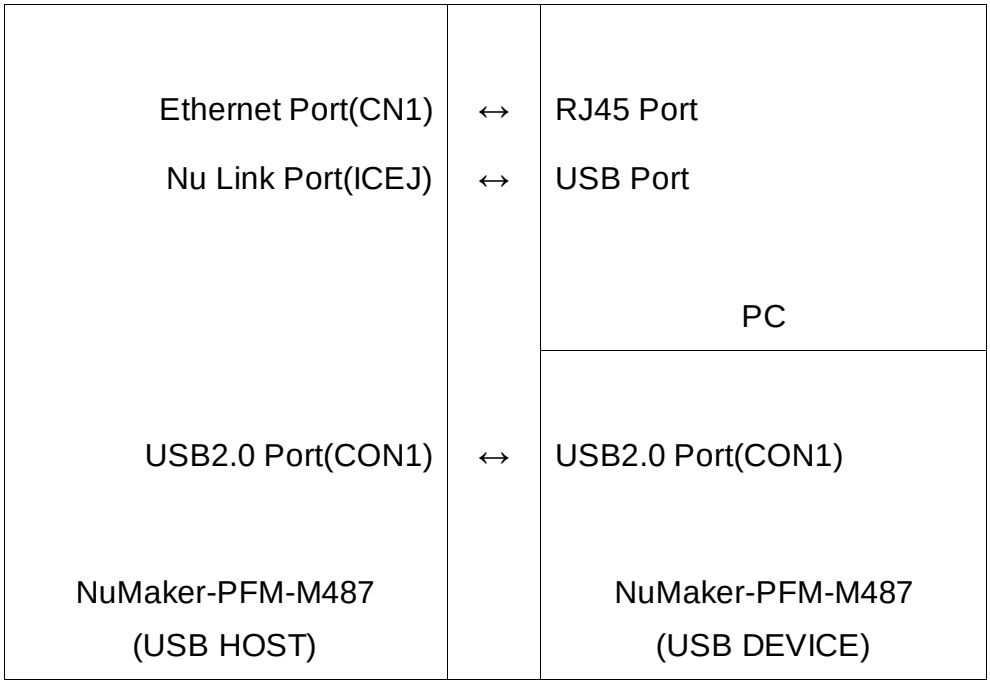

Figure 3-1 Pin Connect

4 Directory Information

The directory structure is shown below.

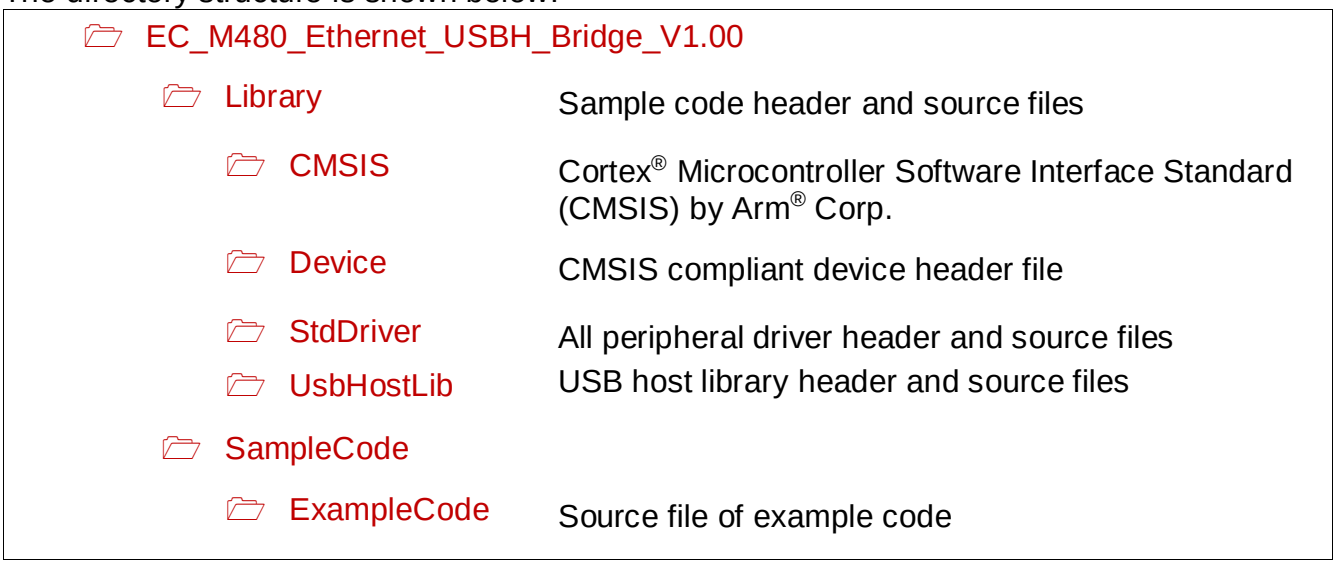

Figure 4-1 Directory Structure

5 Example Code Execution

1. Browse the sample code folder as described in the Directory Information section and double-click *HSUSBH_ETH_BRIDGE.uvproj*.
2. Enter Keil compile mode.
   - Build
   - Download
   - Start/Stop debug session
3. Enter debug mode.
   - Run

6 Revision History

Date	Revision	Description
2021.11.01	1.00	1. Initially issued.

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.

Please note that all data and specifications are subject to change without notice.
All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.