M480 Ethernet to USB Device Bridge

Example Code Introduction for 32-bit NuMicro® Family

Information

Application	This sample code implements a bridge function between Ethernet port and USB device 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 device 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.

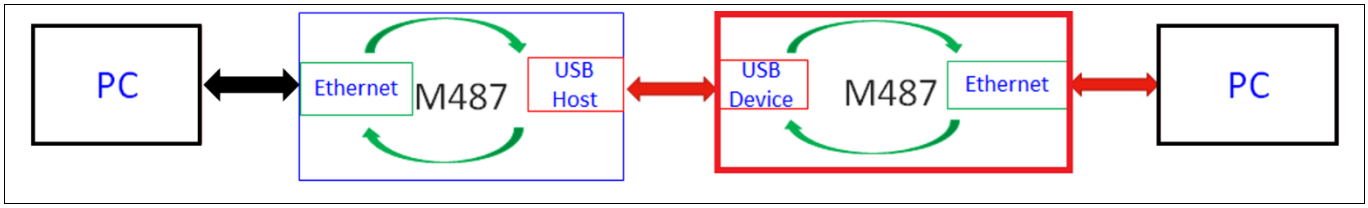

Figure 1-1 Architecture of Ethernet Connection

1.1 Network Connection

The IP addresses of PCs in the network connection solution have 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.

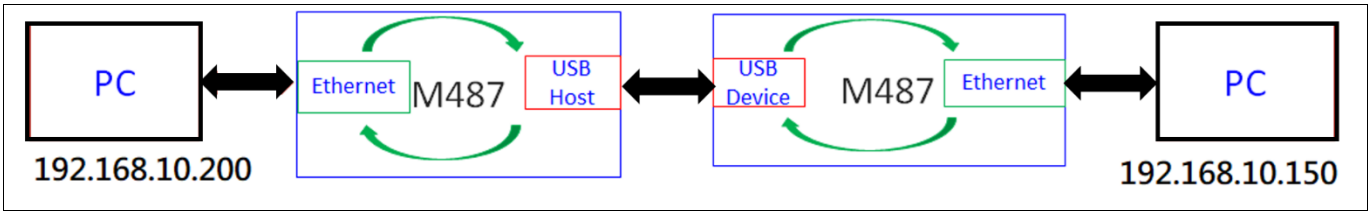

Figure 1-2 IP Addresses Example of the Ethernet Connection

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 board 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

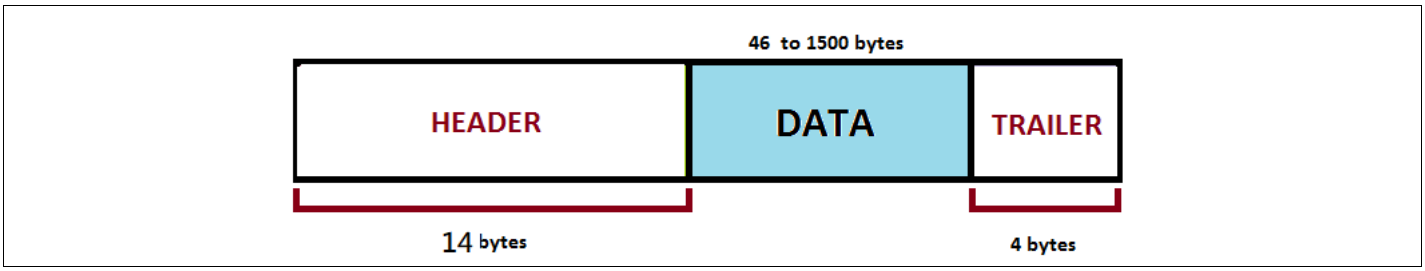

Figure 1-3. The maximum length of the frame is 1518 Bytes.

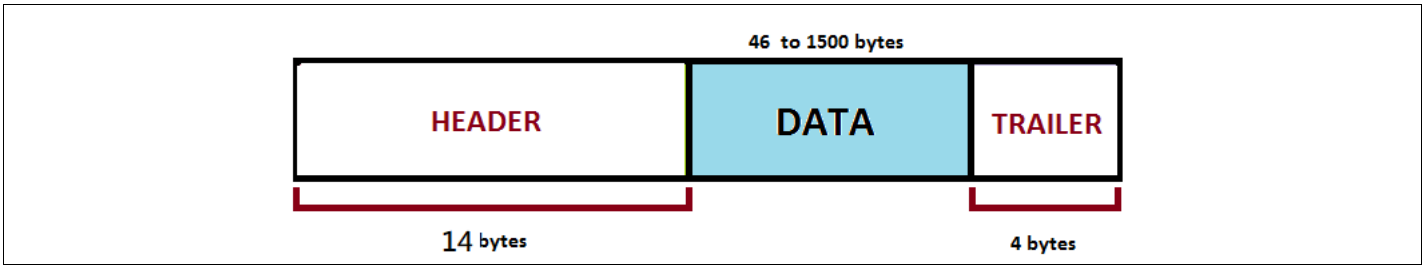

Figure 1-3 Ethernet Frame Structure

As long as the M487 receives an Ethernet frame, extra tags with actual frame length information will be appended to the frame.

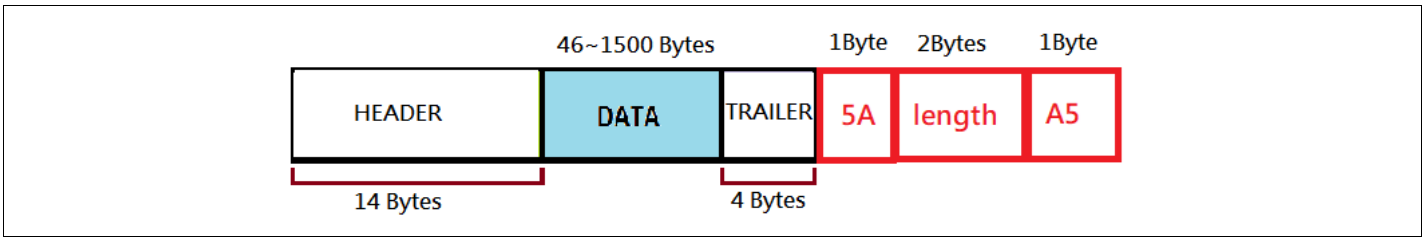

Figure 1-4 shows the updated frame.

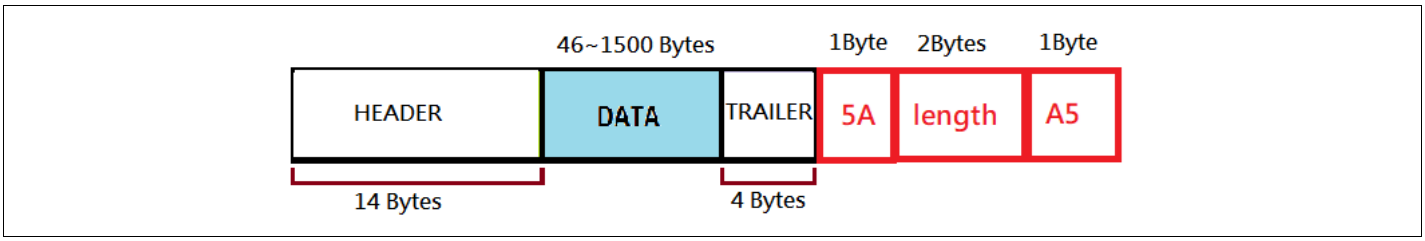

Figure 1-4 Ethernet Frame Structure with Extra Tags

Figure 1-5 presents an example to explain the data transfer flow. The 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.

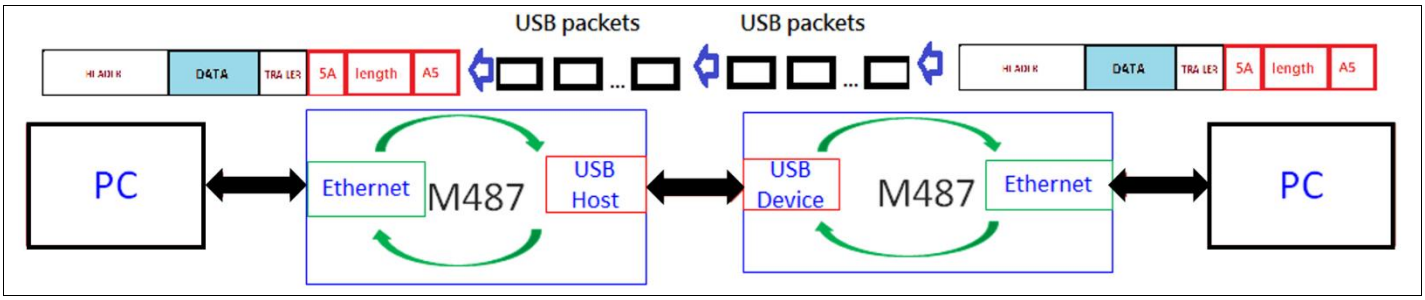

Figure 1-5 Data Transfer Example

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.

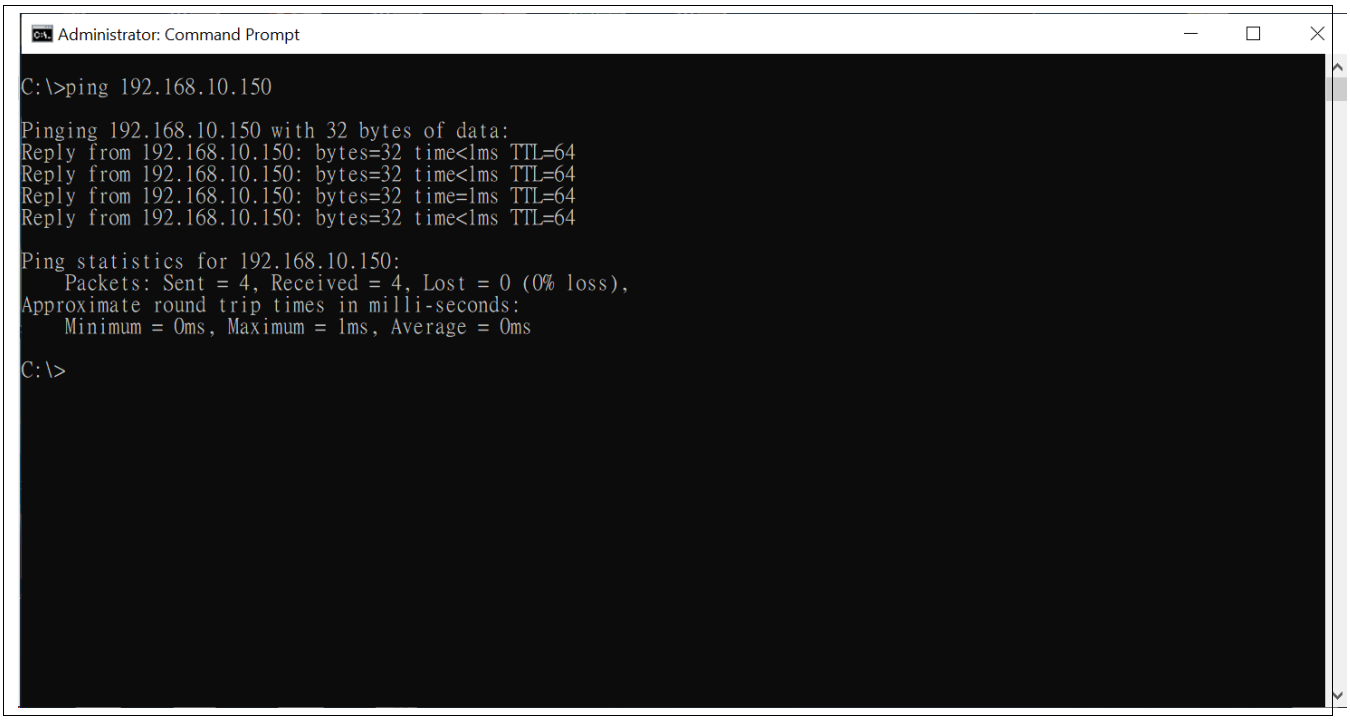

Figure 1-6 Ping Command Result

The demo result is shown in Figure 1-7.

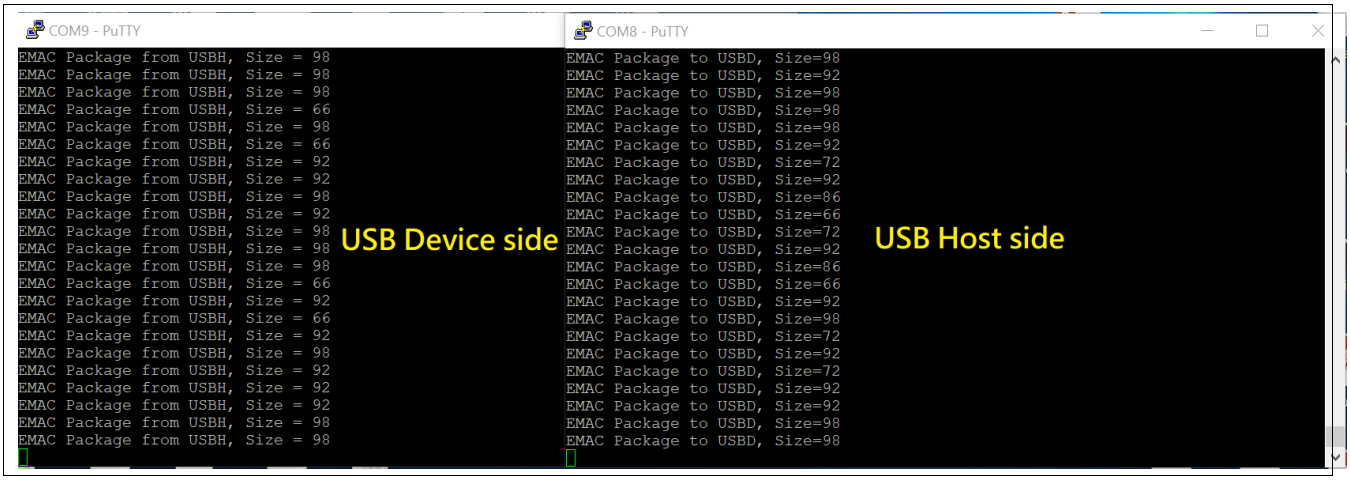

Figure 1-7 Demo Result

2 Code Description

First, initialize M480 in main.c, and initialize Ethernet RMII interface, debug port, and USB device. Then, the USB transaction is getting started.

```
int32_t main(void)
{
    /* Initialize System */
    SYS_Init();

    /* Init UART to 115200-8n1 for print message */
    UART_Open(UART0, 115200);

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

    /* Initialize phy */
    EMAC_PhyInit();

    NVIC_EnableIRQ(EMAC_TX_IRQn);
    NVIC_EnableIRQ(EMAC_RX_IRQn);

    EMAC_ENABLE_RX();
    EMAC_ENABLE_TX();

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

    HSUSBD_Open(&gsHSInfo, HID_ClassRequest, NULL);

    /* Endpoint configuration */
    HID_Init();

    /* Enable USBD interrupt */
    NVIC_EnableIRQ(USBD20_IRQn);

    g_u32BytesInPageBuf = 0;
    /* Start transaction */
    HSUSBD_Start();

    while (1)
    {
    }
}
```

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

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

        printf("EMAC Package to USBH, Size=%d\n", u32PktLen);
        /* Store raw data buffer */
        g_pkg_len = u32PktLen; //max 1518
        g_EMAC_res_len = u32PktLen;
        memcpy(g_EMAC_buf, auPkt, g_pkg_len);
```

```
        /* Add 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;

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

If the EMAC_RX_IRQHandler adds extra tags in the frame, HID_SetInReport function will be triggered to convert the frame into USB packets.

```
void HID_SetInReport(void)
{

    uint32_t len;
    int32_t i;

    /* Prepare the data for next HID IN transfer */
    if (g_pkg_len == 0)
    {
        h_out_start = 0;
    }
    else
    {
        len = Minimum(g_u32EpAMaxPacketSize, g_pkg_len);

        for (i = 0; i < len; i++)
        {
            HSUSBD->EP[EPA].EPDAT_BYTE = g_EMAC_buf[h_out_start + i];
        }

        h_out_start += len;
        HSUSBD->EP[EPA].EPTXCNT = len;
        HSUSBD_ENABLE_EP_INT(EPA, HSUSBD_EPINTEN_INTKIEN_Msk);
        g_pkg_len -= len;
    }
}
```

The HID_GetOutReport function combines all received USB packets and checks the extra tags information. Finally, the assembled frame will be sent.

```
void HID_GetOutReport(uint8_t *pu8EpBuf, uint32_t u32Size)
{
    uint32_t volatile i;


    /* Get data from HID OUT if data len is not zero */
    if (u32Size > 0)
    {


        if ((g_u32BytesOutPageBuf + u32Size) >= PKG_SIZE) g_u32BytesOutPageBuf = 0;

        HSUSBD_MemCopy(&g_u8PageOutBuff[g_u32BytesOutPageBuf], pu8EpBuf, u32Size);

        g_u32BytesOutPageBuf += u32Size;

        /* Check package size tag */
        if (g_u8PageOutBuff[g_u32BytesOutPageBuf - 1] == 0xA5)
```

```
        {
            if (g_u8PageOutBuff[g_u32BytesOutPageBuf - 4] == 0x5A)
            {
                g_pkg_size = g_u8PageOutBuff[g_u32BytesOutPageBuf - 3] << 8 |
g_u8PageOutBuff[g_u32BytesOutPageBuf - 2];
                printf("EMAC Package from USBH, Size = %d\n", g_pkg_size);
                /* Send received package to EMAC  without “A5 + lenginfo + 5A” */
                EMAC_SendPkt(g_u8PageOutBuff, g_pkg_size);
                g_u32BytesOutPageBuf = 0;
            }
        }
    }
}
```

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

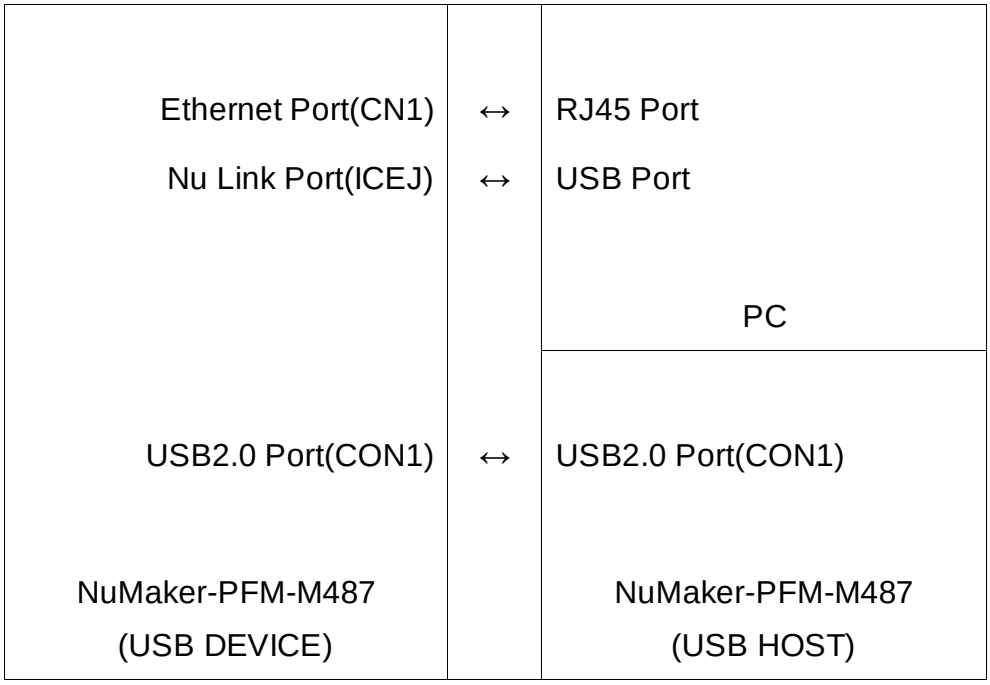

Figure 3-1 Pin Connect

4 Directory Information

The directory structure is shown below.

EC_M480_Ethernet_USBD_Bridge_V1.00	
Library	Sample code header and source files
CMSIS	Cortex® Microcontroller Software Interface Standard (CMSIS) by Arm® Corp.
Device	CMSIS compliant device header file
StdDriver	All peripheral driver header and source files
SampleCode	
ExampleCode	Source file of example code

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 *HSUSBD_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.