M032 USB HID to GPIO Bridge

NuMicro® 32 位元系列微控制器範例代碼介紹

文件資訊

應用簡述	本範例程式碼使用 M032 實現 USB HID to GPIO bridge 的功能
BSP 版本	M031_Series_BSP_CMSIS_V3.06.000
開發平台	NuMaker-M032TC_V1.2

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. 概述

本範例程式碼展示如何利用 NuMicro® M032 系列微控制器實現 USB HID 裝置，並透過自定義的 HID Report 協議與 PC 端溝通，同時提供一個虛擬序列 COM 埠供使用者收發資料。

PC 端透過 USB HID 傳送寫入或讀取命令，控制 M032 晶片上的 PB.0、PB.1、PB.2、PB.3 引腳模式（輸入/輸出）及電位狀態（High/Low）。

1.1 原理

本範例透過 USB HID 的 Interrupt OUT (接收 PC 命令) 與 Interrupt IN (回傳資料給 PC) 端點來進行資料交換。由於 HID 類別在 Windows 下不需額外驅動，本範例也提供了使用者控制介面 M032_HID_Controller.exe，透過 HID 通道發送報告控制裝置，並即時顯示 GPIO 狀態，可方便開發與驗證。

GPIO 配置：範例中使用 PB.0 ~ PB.3 共四個腳位對應到 HID 傳輸：PB.0(Pin0)、PB.1(Pin1)、PB.2(Pin2)、PB.3(Pin3)。程式在啟動時將 PB.0~PB.3 設為輸出模式。

當 USB 接收到 HID HID_CMD_WRITE 或 HID_CMD_READ 命令時，會觸發 HID_GetOutReport 並解析資料封包，進而呼叫 USB_to_Write_GPIO 或 USB_to_Read_GPIO 函式。HID 資料說明如圖 1-1 所示。

HID Command Write

	Report ID	CMD_WRITE	CMD_SIZE	StartPage	Pages	Signature	Checksum	Padding	
(MSB)	0x00	0xC3	0x0E	0x20	0x01	0x43444948	0x020A	0x00	(LSB)

HID Command Read

	Report ID	CMD_WRITE	CMD_SIZE	StartPage	Pages	Signature	Checksum	Padding	
(MSB)	0x00	0xD2	0x0E	0x20	0x01	0x43444948	0x0219	0x00	(LSB)

HID Data

	Pin0_Conf	Pin1_Conf	Pin2_Conf	Pin3_Conf	Pin0_WR	Pin1_WR	Pin2_WR	Pin3_WR	
(MSB)	0x00 : GPIO	0x00 : GPIO	0x00 : GPIO	0x00 : GPIO	0x00 : Read 0x01 : Write	0x00 : Read 0x01 : Write	0x00 : Read 0x01 : Write	0x00 : Read 0x01 : Write	
	PIN0_DATA	**PIN1_DATA**	**PIN2_DATA**	**PIN3_DATA**	**PIN0_Write**	**PIN1_Write**	**PIN2_Write**	**PIN3_Write**	
	0x00 : Low 0x01 : High	0x00 : Low 0x01 : High	0x00 : Low 0x01 : High	0x00 : Low 0x01 : High	0x00 : Low 0x01 : High	0x00 : Low 0x01 : High	0x00 : Low 0x01 : High	0x00 : Low 0x01 : High	(LSB)

圖 1-1 HID 資料說明

1.2 執行結果

本範例程式碼執行編譯和燒錄後再搭配使用者控制介面 M032_HID_Controller.exe，如圖 1-2 所示，按下 **Search** 鈕搜尋可用的 HID 裝置，找到本範例的裝置(VID=0x0416，PID=0xDC00)後點選，然後按 **Open** 建立連線。

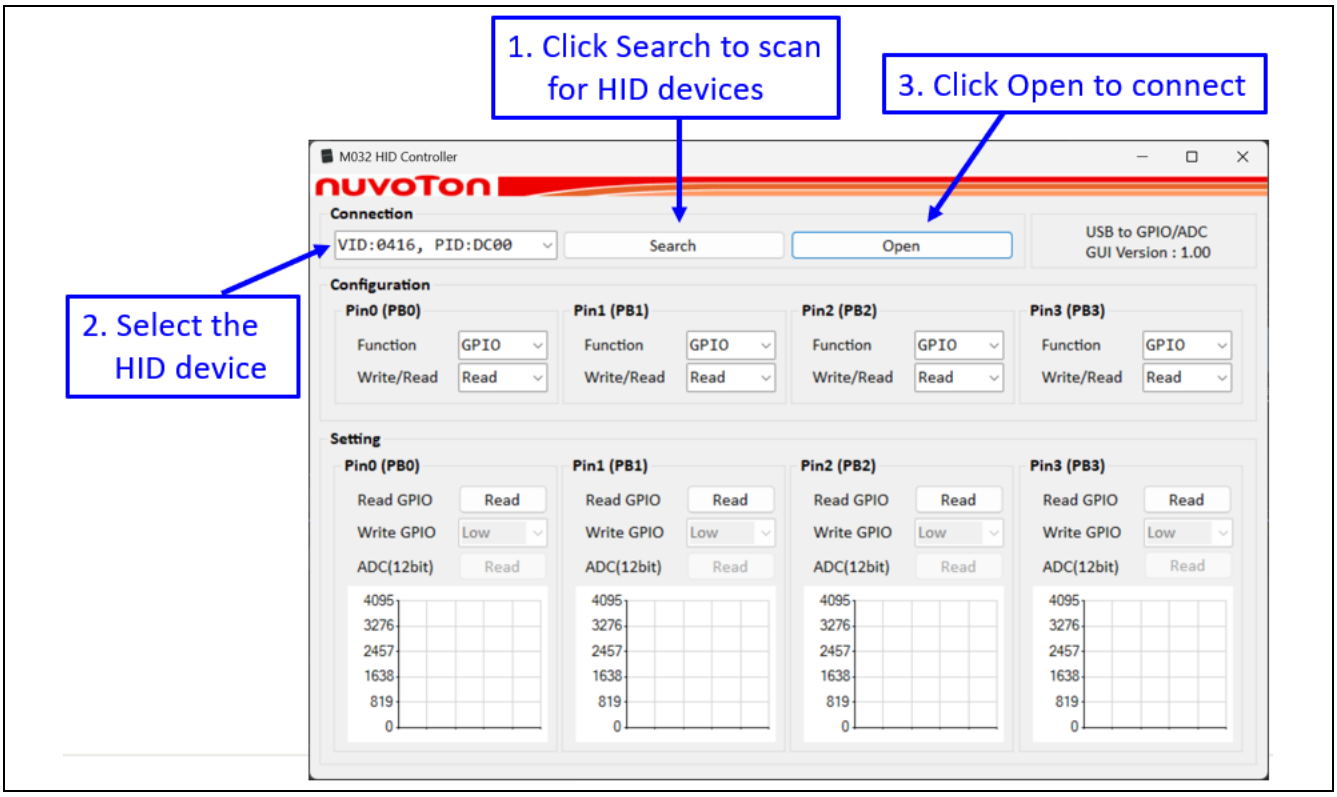

圖 1-2 M032 HID Controller 連線說明

圖 1-3 說明 GPIO 寫入 (GPIO Write)：在 GUI 中選擇 Write 功能，並在 High/Low 下拉選單中選擇要寫入的電位，即可將該腳位設為高或低電位。

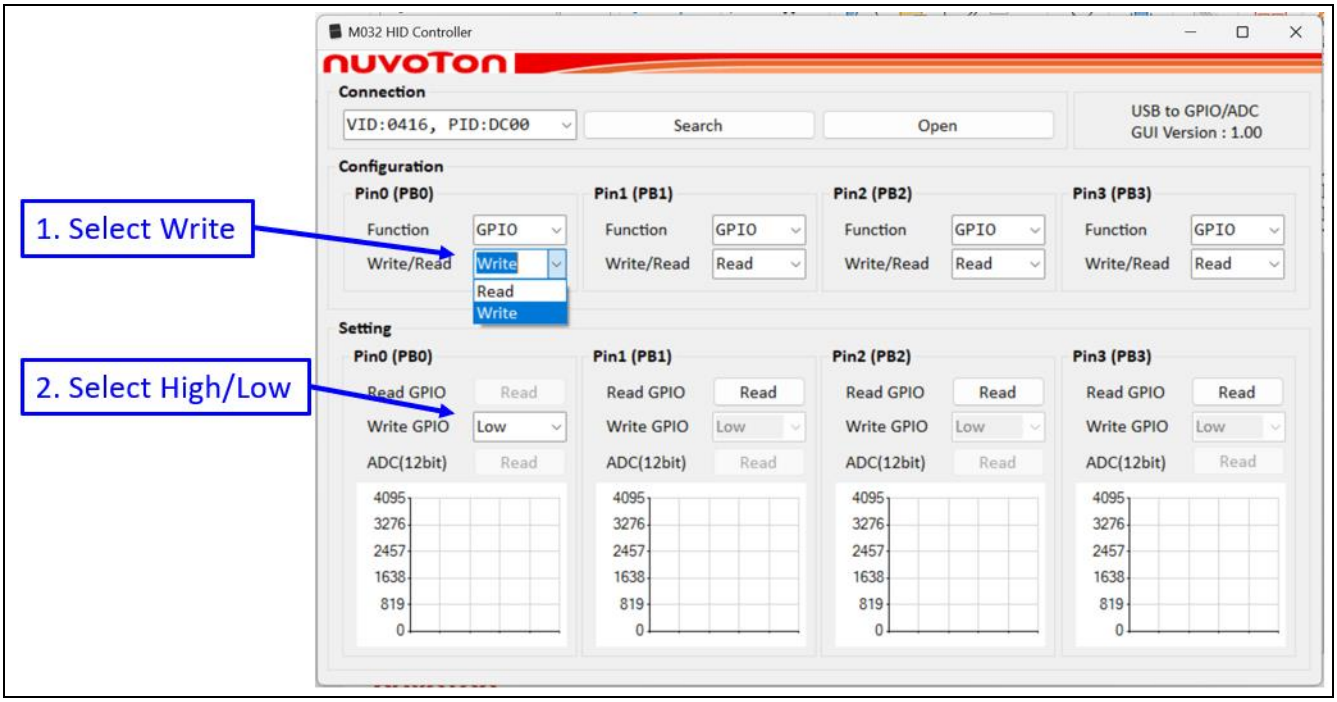

圖 1-3 GPIO 寫入 (GPIO Write)

如圖 1-4 所說明 GPIO 讀取 (GPIO Read)：選擇 **Read** 功能，點選 Read 鈕即可讀取所選腳位的當前電位，讀取結果會即時顯示在介面上。

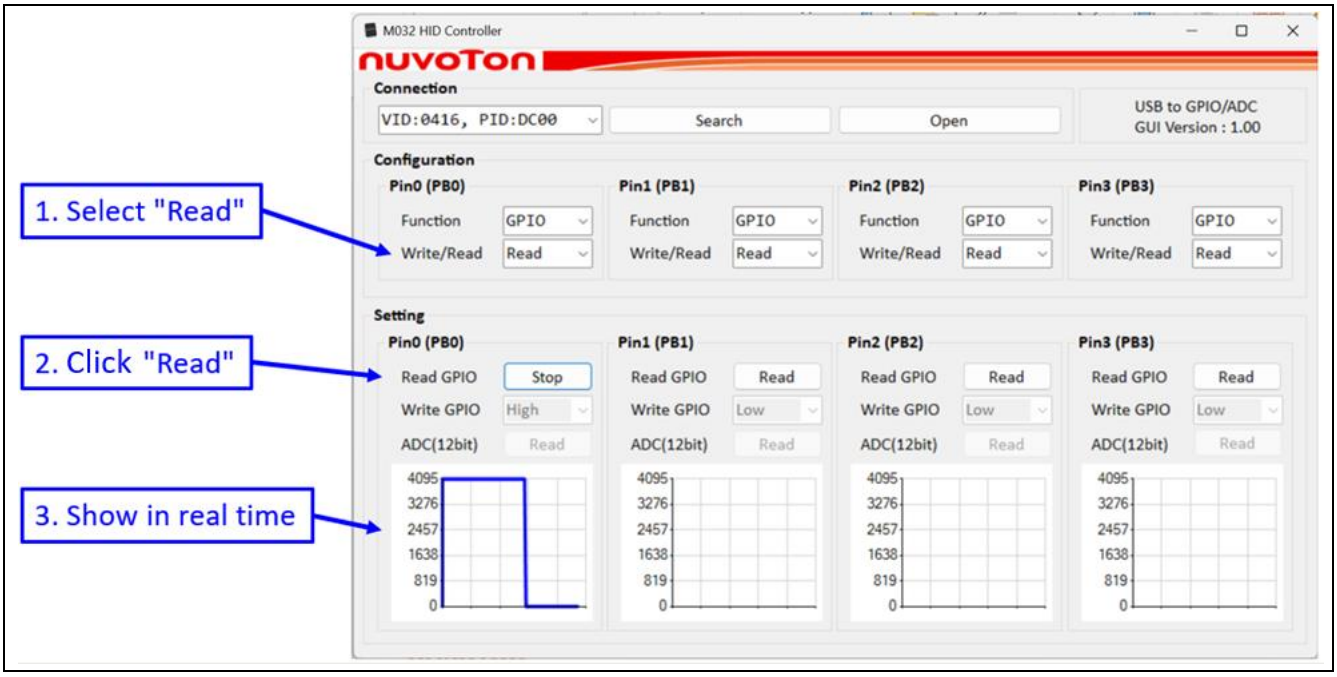

圖 1-4 GPIO 讀取 (GPIO Read)

2. 程式碼介紹

系統初始化與 GPIO 及 USB 裝置的啟動，程式碼位於 *main.c*。

```
/*---------------------------------------------------------------------------------------------------------*/
/*  Main Function                                                                                          */
/*---------------------------------------------------------------------------------------------------------*/
int32_t main(void)
{
#if CRYSTAL_LESS
    uint32_t u32TrimInit;
#endif

    /* Unlock protected registers */
    SYS_UnlockReg();

    /* Init System, peripheral clock and multi-function I/O */
    SYS_Init();

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

    printf("+------------------------------------------------------+\n");
    printf("|     NuMicro USB HID and GPIO Transfer Sample Code     |\n");
    printf("+------------------------------------------------------+\n");
    printf("PID is 0xDC00\n");

    /* Configure PB.0 ~ PB.3 as Output mode */
    GPIO_SetMode(PB, BIT0, GPIO_MODE_OUTPUT);
    GPIO_SetMode(PB, BIT1, GPIO_MODE_OUTPUT);
    GPIO_SetMode(PB, BIT2, GPIO_MODE_OUTPUT);
    GPIO_SetMode(PB, BIT3, GPIO_MODE_OUTPUT);

    /* Open USB controller */
    USBD_Open(&gsInfo, HID_ClassRequest, NULL);

    /* Endpoint configuration */
    HID_Init();

    /* Start USB device */
    USBD_Start();

    NVIC_EnableIRQ(USBD_IRQn);

    NVIC_EnableIRQ(UART02_IRQn);

#if CRYSTAL_LESS
    /* Backup default trim */
    u32TrimInit = M32(TRIM_INIT);
#endif

    /* Clear SOF */
    USBD->INTSTS = USBD_INTSTS_SOFIF_Msk;
```

```
    while (1)
    {
#if CRYSTAL_LESS

        /* Start USB trim if it is not enabled. */
        if ((SYS->HIRCTRIMCTL & SYS_HIRCTRIMCTL_FREQSEL_Msk) != 1)
        {
            /* Start USB trim only when SOF */
            if (USBD->INTSTS & USBD_INTSTS_SOFIF_Msk)
            {
                /* Clear SOF */
                USBD->INTSTS = USBD_INTSTS_SOFIF_Msk;

                /* Re-enable crystal-less */
                SYS->HIRCTRIMCTL = 0x01;
                SYS->HIRCTRIMCTL |= SYS_HIRCTRIMCTL_REFCKSEL_Msk;
            }
        }

        /* Disable USB Trim when error */
        if (SYS->HIRCTRIMSTS & (SYS_HIRCTRIMSTS_CLKERIF_Msk |
             SYS_HIRCTRIMSTS_TFAILIF_Msk))
        {
            /* Init TRIM */
            M32(TRIM_INIT) = u32TrimInit;

            /* Disable crystal-less */
            SYS->HIRCTRIMCTL = 0;

            /* Clear error flags */
            SYS->HIRCTRIMSTS = SYS_HIRCTRIMSTS_CLKERIF_Msk | SYS_HIRCTRIMSTS_TFAILIF_Msk;

            /* Clear SOF */
            USBD->INTSTS = USBD_INTSTS_SOFIF_Msk;
        }

#endif

        /* Enter power down when USB suspend */
        if (g_u8Suspend)
            PowerDown();

        /* USB HID data transfer */
        USB_HID_TransferData();
    }
}
```

此函式將目前的 GPIO 引腳狀態填入緩衝區，隨後透過 USB 回傳給 PC，程式碼位於 *GPIO_and_HID_Transfer.c*。

```
void USB_to_Read_GPIO(uint8_t *data)
{
    /* GPIO Input */
    data[8]  = PIN0;
```

```
    data[9]  = PIN1;
    data[10] = PIN2;
    data[11] = PIN3;
}
```

此函式負責解析來自 USB 的數據包，並根據內容設定 GPIO 的模式與輸出狀態，程式碼位於 *GPIO_and_HID_Transfer.c*。

```
void USB_to_Write_GPIO(uint8_t *data)
{
    /* Unlock protected registers */
    SYS_UnlockReg();

    /* Pin Config */
    CONF_S.Pin0_Conf = data[0];
    CONF_S.Pin1_Conf = data[1];
    CONF_S.Pin2_Conf = data[2];
    CONF_S.Pin3_Conf = data[3];

    /* Write & Read */
    WR_S.Pin0_WR = data[4];
    WR_S.Pin1_WR = data[5];
    WR_S.Pin2_WR = data[6];
    WR_S.Pin3_WR = data[7];

    /* High & Low */
    HL_S.Pin0_HL = data[12];
    HL_S.Pin1_HL = data[13];
    HL_S.Pin2_HL = data[14];
    HL_S.Pin3_HL = data[15];

    /* Pin 0 config (PB0) */
    switch (CONF_S.Pin0_Conf)
    {
        case 0:
            switch (WR_S.Pin0_WR)
            {
                case 0: // GPIO & Read
                    GPIO_SetMode(PB, BIT0, GPIO_MODE_INPUT);
                    break;

                case 1: // GPIO & Write
                    GPIO_SetMode(PB, BIT0, GPIO_MODE_OUTPUT);

                    switch (HL_S.Pin0_HL)
                    {
                        case 0: // Low
                            PIN0 = 0;
                            break;

                        case 1: // High
                            PIN0 = 1;
                            break;
                    }
                    break;
```

```
            }
            break;

        case 1:
            break;
    }

    /* Pin 1 config (PB1) */
    switch (CONF_S.Pin1_Conf)
    {
        case 0:
            switch (WR_S.Pin1_WR)
            {
                case 0: // GPIO & Read
                    GPIO_SetMode(PB, BIT1, GPIO_MODE_INPUT);
                    break;

                case 1: // GPIO & Write
                    GPIO_SetMode(PB, BIT1, GPIO_MODE_OUTPUT);

                    switch (HL_S.Pin1_HL)
                    {
                        case 0: // Low
                            PIN1 = 0;
                            break;

                        case 1: // High
                            PIN1 = 1;
                            break;
                    }
                    break;
            }
            break;

        case 1:
            break;
    }

    /* Pin 2 config (PB2) */
    switch (CONF_S.Pin2_Conf)
    {
        case 0:
            switch (WR_S.Pin2_WR)
            {
                case 0: // GPIO & Read
                    GPIO_SetMode(PB, BIT2, GPIO_MODE_INPUT);
                    break;

                case 1: // GPIO & Write
                    GPIO_SetMode(PB, BIT2, GPIO_MODE_OUTPUT);

                    switch (HL_S.Pin2_HL)
                    {
                        case 0: // Low
                            PIN2 = 0;
                            break;
```

```
                        case 1: // High
                            PIN2 = 1;
                            break;
                    }
                    break;
            }
            break;

        case 1:
            break;
    }

    /* Pin 3 config (PB3) */
    switch (CONF_S.Pin3_Conf)
    {
        case 0:
            switch (WR_S.Pin3_WR)
            {
                case 0: // GPIO & Read
                    GPIO_SetMode(PB, BIT3, GPIO_MODE_INPUT);
                    break;

                case 1: // GPIO & Write
                    GPIO_SetMode(PB, BIT3, GPIO_MODE_OUTPUT);

                    switch (HL_S.Pin3_HL)
                    {
                        case 0: // Low
                            PIN3 = 0;
                            break;

                        case 1: // High
                            PIN3 = 1;
                            break;
                    }
                    break;
            }
            break;

        case 1:
            break;
    }

    /* Lock protected registers */
    SYS_LockReg();
}
```

此函式負責函式接收來自 PC 的 USB HID 數據，判斷命令類型並執行動作，程式碼位於 *GPIO_and_HID_Transfer.c*。

```
void HID_GetOutReport(uint8_t *pu8EpBuf, uint32_t u32Size)
{
    uint32_t i;
    uint8_t  u8Cmd;
```

```
    uint32_t u32StartPage;
    uint32_t u32Pages;
    uint32_t u32PageCnt;

    /* Get command information */
    u8Cmd        = gCmd.u8Cmd;
    u32StartPage = gCmd.u32Arg1;
    u32Pages     = gCmd.u32Arg2;
    u32PageCnt   = gCmd.u32Signature; /* The signature word is used to count pages */

    /* Check if it is in the data phase of write command */
    if ((u8Cmd == HID_CMD_WRITE) && (u32PageCnt < u32Pages))
    {
        /* Process the data phase of write command */

        /* Get data from HID OUT */
        USBD_MemCopy(&g_u8PageBuff[g_u32BytesInPageBuf], pu8EpBuf, EP6_MAX_PKT_SIZE);
        g_u32BytesInPageBuf += EP6_MAX_PKT_SIZE;

        /* The HOST must make sure the data is PAGE_SIZE alignment */
        if (g_u32BytesInPageBuf >= PAGE_SIZE)
        {
            printf("Writing page %d\n", u32StartPage + u32PageCnt);
            /* TODO: We should program received data to storage here */
            memcpy(g_u8TestPages + u32PageCnt * PAGE_SIZE, g_u8PageBuff,
                   sizeof(g_u8PageBuff));
            u32PageCnt++;

            /* Write command complete! */
            if (u32PageCnt >= u32Pages)
            {
                u8Cmd = HID_CMD_NONE;
                printf("Write command complete.\n");

                /* Process the data phase of write command */
                USB_to_Write_GPIO(&g_u8TestPages[0]);
            }
            g_u32BytesInPageBuf = 0;
        }

        /* Update command status */
        gCmd.u8Cmd        = u8Cmd;
        gCmd.u32Signature = u32PageCnt;
    }
    else
    {
        /* Check and process the command packet */
        if (ProcessCommand(pu8EpBuf, sizeof(gCmd)))
        {
            printf("Unknown HID command!\n");
        }
    }
}
```

3. 軟體與硬體需求

3.1 軟體需求

- BSP 版本
  - M031_Series_BSP_CMSIS_V3.06.000
- IDE 版本
  - Keil uVision 5.41

3.2 硬體需求

- 電路元件
  - NuMaker-M032TC_V1.2
- 線路示意圖
  - 將 NuMaker-M032TC_V1.2 的 USB 連接到 PC 的 USB，以顯示範例程式碼的執行結果。

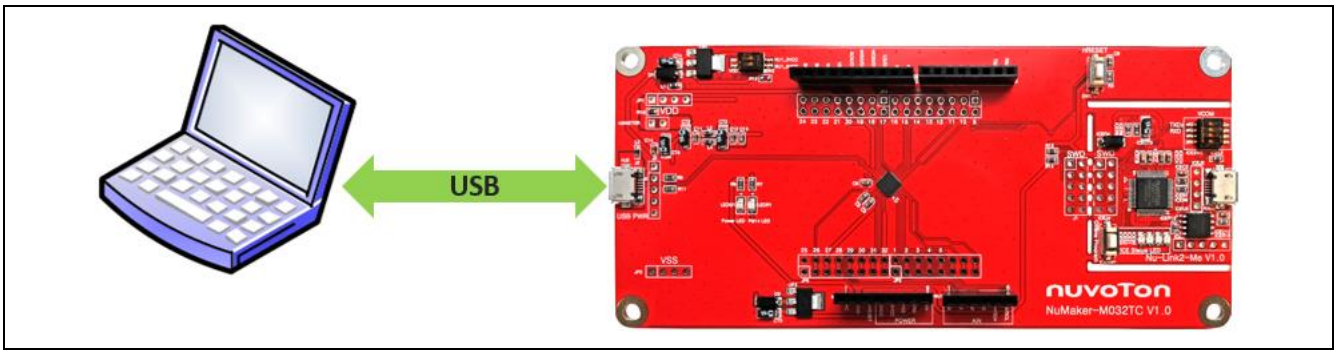

圖 3-1 線路示意圖

4. 目錄資訊

目錄結構如下圖所示。

EC_M032_USB_HID_to_GPIO_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	
Project	Source file of example code
WindowsTool	User Control GUI Tool

圖 4-1 目錄資訊

5. 範例程式執行

1. 根據目錄資訊章節進入 ExampleCode 路徑中的 KEIL 資料夾，雙擊 *USB_HID_to_GPIO_Bridge.uvprojx*。
2. 進入編譯模式介面
   - 編譯
   - 下載程式碼至記憶體
   - 進入 / 離開除錯模式
3. 進入除錯模式介面
   - 執行程式碼

6. 修訂紀錄

Date	Revision	Description
2026.01.27	1.00	初始發布。

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.