包含HID 滑鼠功能並使用SPI Mode讀取SD卡的USB儲存裝置

NuMicro® 32 位系列微控制器範例代碼介紹

文件資訊

代碼簡述	此範例透過 SPI 介面讀取 SD 卡作為 USB 儲存裝置並支援 HID 滑鼠功能
BSP 版本	NUC200 Series BSP CMSIS V3.00.003
開發平台	NuTiny-EVB-NUC240-LQFP100 V1.00

1 功能介紹

1.1 簡介

此範例示範包含HID滑鼠功能的USB儲存裝置，透過SPI介面讀取SD Card，並使用GPIO模擬滑鼠輸入。

1.2 原理

1.2.1 SPI Mode SD Card

SD Card支援兩種運作模式：SD Card模式與SPI模式，相關訊號定義於表格 1。

Pin	SD Mode			SPI Mode		
	Name	Type	Description	Name	Type	Description
1	CD/DAT3	I/O/PP	Card Detect/ Data Line [Bit 3]	CS	I	Chip Select (neg true)
2	CMD	PP	Command/Response	DI	PP	Data In
3	VSS	S	Supply voltage ground	VSS	S	Supply voltage ground
4	VDD	S	Supply voltage	VDD	S	Supply voltage
5	CLK	I	Clock	SCLK	I	Clock
6	VSS	S	Supply voltage ground	VSS	S	Supply voltage ground
7	DAT0	I/O/PP	Data Line [Bit 0]	DO	O/PP	DataOut
8	DAT1	I/O/PP	Data Line [Bit 1]	RSV		
9	DAT2	I/O/PP	Data Line [Bit 2]	RSV		

表格 1 SD 卡運作模式訊號定義

此範例提供SPI Mode SD Card命令與初始化流程的原始碼。

1.2.2 USB 裝置

此範例程式複合裝置有2組介面，使用MSC(Mass Storage Device)與HID(Human Interface Devices)標準實作。一個USB裝置一定會有一個控制管線(端點0)，用於控制傳輸。其他端點可以根據應用而有不同的設定。此範例中的控制管線由裝置端硬體的端點0和端點1構成，從USB主機的角度看來，只會看到端點0；介面0包含一個批量的輸入端點與一個批量的輸出端點，用於批量傳輸(Bulk Transfer)；介面1包含一個中斷的輸入端點，此端點用於輸入方向的中斷傳輸

(Interrupt Transfer)。這些介面和端點的關系列於表格 2。

Function	USB Composite Device	NUC200 Hardware setting
Control Transfer	Control Pipe endpoint 0: Control IN/OUT	endpoint 0: Control IN endpoint 1: Control OUT
Mass Storage	Interface 0 endpoint 2: Bulk IN endpoint 3: Bulk OUT	endpoint 2: Bulk IN endpoint 3: Bulk OUT
HID Mouse	Interface 1 endpoint 4: Interrupt IN	endpoint 4: Interrupt IN

表格 2 複合裝置的介面和端點組態

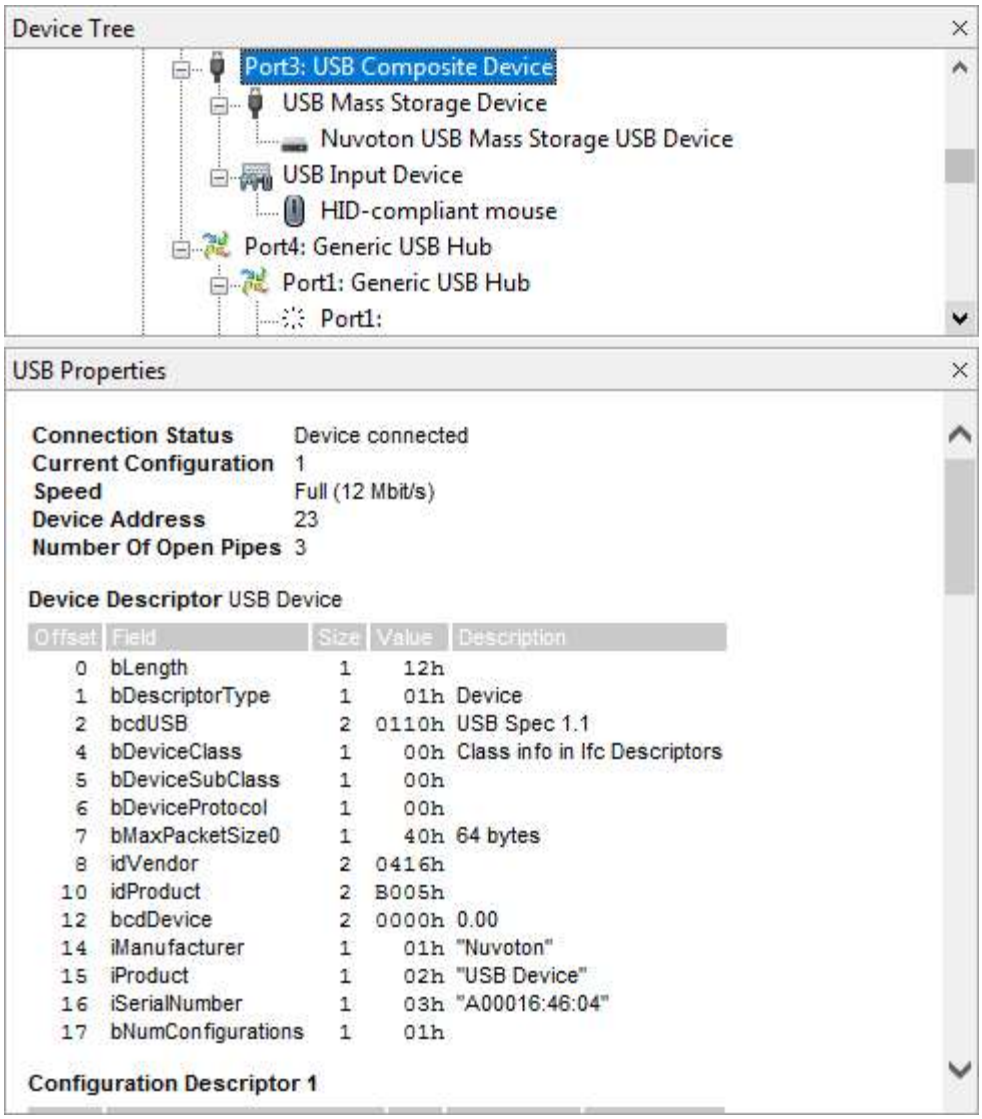
介面1是一個HID滑鼠裝置，使用表格 3的資料格式回傳資料。

Byte	Bits	Description
0	0~2	Button 1~3
0	3~7	Padding
1	0~7	X-axis
2	0~7	Y-axis

表格 3 HID 滑鼠報告格式

1.3 執行結果

此範例需與連接個人電腦才能運行。此範例程式執行時，使用USBLyzer我們可以獲得圖表 1的裝置清單。可以看到此複合裝置有2組介面。



圖表 1 裝置清單

2 代碼介紹

2.1 SPI Mode SD Card

設定 SPI 介面與 SD Card 初始化。

```
uint8_t udcFlashInit(void)
{
    RoughDelay(100000);

    if(DrvSDCARD_Open() == E_SUCCESS)
        printf("SDCard Open success\n");
    else
        printf("SDCard Open failed\n");

    if (!Flash_Identify())
        return 0;

    return 1;
}
```

讀取 SC Card。

```
void SpiRead(uint32_t addr, uint32_t size, uint8_t* buffer)
{
/* This is low level read function of USB Mass Storage */
    uint32_t response;
    if(addr>=LogicSector)
    {
        DBG_PRINTF("Read illegal Sector:0x%x\n",addr);
        return;
    }
    if(SDtype&SDBlock)
    {
        while(size >= PHYSICAL_BLOCK_SIZE)
        {
            while(MMC_Command_Exec(READ_SINGLE_BLOCK,addr,buffer,&response)==FALSE);
            addr   ++;
            buffer += PHYSICAL_BLOCK_SIZE;
            size   -= PHYSICAL_BLOCK_SIZE;
        }
```

```
    }
    else
    {
        addr*=PHYSICAL_BLOCK_SIZE;
        while(size >= PHYSICAL_BLOCK_SIZE)
        {
            while(MMC_Command_Exec(READ_SINGLE_BLOCK,addr,buffer,&response)==FALSE);
            addr   += PHYSICAL_BLOCK_SIZE;
            buffer += PHYSICAL_BLOCK_SIZE;
            size   -= PHYSICAL_BLOCK_SIZE;
        }
    }
}
```

寫入 SD Card。

```
void SpiWrite(uint32_t addr, uint32_t size, uint8_t* buffer)
{
    uint32_t response;
    if(addr>=LogicSector)
    {
        DBG_PRINTF("Write illegal Sector:0x%x\n",addr);
        return;
    }
    if(SDtype&SDBlock)
    {
        while(size >= PHYSICAL_BLOCK_SIZE)
        {
            while(MMC_Command_Exec(WRITE_BLOCK,addr,buffer,&response)==FALSE);
            addr   ++;
            buffer += PHYSICAL_BLOCK_SIZE;
            size   -= PHYSICAL_BLOCK_SIZE;
        }
    }
    else
    {
        addr*=PHYSICAL_BLOCK_SIZE;
        while(size >= PHYSICAL_BLOCK_SIZE)
        {
            while(MMC_Command_Exec(WRITE_BLOCK,addr,buffer,&response)==FALSE);
```

```
            addr   += (PHYSICAL_BLOCK_SIZE);
            buffer += PHYSICAL_BLOCK_SIZE;
            size   -= PHYSICAL_BLOCK_SIZE;
        }
    }
}
```

2.2 滑鼠

USB 主機會發出 IN Token 向端點 4 請求資料，當 IN Token 收到時，會執行端點 4 的中斷處理常式設定並回傳數據。以此範例為例，當 GPB15 電位為 0 時，設定 buf[2] (表格 3 的 X 軸欄位)為 0x1。

```
void EP4_Handler(void)
{
    uint8_t *buf;
    buf = (uint8_t *)(USBD_BUF_BASE + USBD_GET_EP_BUF_ADDR(EP4));

    buf[0] = 0x00;
    if(PB15 == 0)
        buf[1] = 0x01;
    else
        buf[1] = 0x00;
    buf[2] = 0x00;
    USBD_SET_PAYLOAD_LEN(EP4, 3);
}
```

2.3 MSC & HID 類別請求

MSC_ClassRequest()包含了 MSC 與 HID 類別請求。

```
void MSC_ClassRequest(void)
{
    uint8_t buf[8];
    USBD_GetSetupPacket(buf);

    if(buf[0] & EP_INPUT)    /* request data transfer direction */
    {
 /* Device to host */
        switch(buf[1])
```

```
        {
        case GET_MAX_LUN:
        {
            if((buf[4] == gsInfo.gu8ConfigDesc[LEN_CONFIG + 2]) &&
                (buf[2] + buf[3] + buf[6] + buf[7] == 1))
            {
                M8(USBD_BUF_BASE + USBD_GET_EP_BUF_ADDR(EP0)) = 0;
/* Data stage */
                USBD_SET_DATA1(EP0);
                USBD_SET_PAYLOAD_LEN(EP0, 1);
                /* Status stage */
                USBD_PrepareCtrlOut(0, 0);
            }
            else
                USBD_SET_EP_STALL(EP1); /* Stall when wrong parameter */
            break;
        }
        case GET_REPORT:
        case GET_IDLE:
        case GET_PROTOCOL:
        default:
        {
 /* Setup error, stall the device */
            USBD_SetStall(0);
            DBG_PRINTF("Unknow MSC req(0x%x). stall ctrl pipe\n", buf[1]);
            break;
        }
        }
    }
    else
    {
       /* Host to device */
        switch(buf[1])
        {
        case BULK_ONLY_MASS_STORAGE_RESET:
        {
 /* Check interface number with cfg descriptor and check wValue = 0,
              wLength = 0 */
            if((buf[4] == gsInfo.gu8ConfigDesc[LEN_CONFIG + 2]) &&
                (buf[2] + buf[3] + buf[6] + buf[7] == 0))
            {
```

```
                g_u32Length = 0; /* Reset all read/write data transfer */
                USBD_LockEpStall(0);

               /* Clear ready */
                USBD->EP[EP2].CFGP |= USBD_CFGP_CLRRDY_Msk;
                USBD->EP[EP3].CFGP |= USBD_CFGP_CLRRDY_Msk;

                /* Prepare to receive the CBW */
                g_u8EP3Ready = 0;
                g_u8BulkState = BULK_CBW;
                USBD_SET_DATA1(EP3);
                USBD_SET_EP_BUF_ADDR(EP3, g_u32BulkBuf0);
                USBD_SET_PAYLOAD_LEN(EP3, 31);
            }
            else
            {
                /* Stall when wrong parameter */
                USBD_SET_EP_STALL(EP1);
            }
            /* Status stage */
            USBD_SET_DATA1(EP0);
            USBD_SET_PAYLOAD_LEN(EP0, 0);
            break;
        }
        case SET_REPORT:
        {
            if(buf[3] == 3)
            {
                /* Request Type = Feature */
                USBD_SET_DATA1(EP1);
                USBD_SET_PAYLOAD_LEN(EP1, 0);
            }
            break;
        }
        case SET_IDLE:
        {
            /* Status stage */
            USBD_SET_DATA1(EP0);
            USBD_SET_PAYLOAD_LEN(EP0, 0);
            EP4_Handler();
            break;
```

```
        }
        case SET_PROTOCOL:
        default:
        {
            /* Setup error, stall the device */
            USBD_SetStall(0);
            DBG_PRINTF("Unknow MSC req (0x%x). stall ctrl pipe\n", buf[1]);
            break;
        }
        }
    }
}
```

2.4 裝置描述符

2.4.1 HID 報告描述符

HID_MouseReportDescriptor 是滑鼠的 HID 報告描述符元。此描述元定義了表格 3 中的資料格式。此資料格式內有，3 個按鈕、X 軸、Y 軸和滾輪。

```
const uint8_t HID_MouseReportDescriptor[] =
{
    0x05, 0x01,     /* Usage Page(Generic Desktop Controls) */
    0x09, 0x02,     /* Usage(Mouse) */
    0xA1, 0x01,     /* Collection(Application) */
    0x09, 0x01,     /* Usage(Pointer) */
    0xA1, 0x00,     /* Collection(Physical) */
    0x05, 0x09,     /* Usage Page(Button) */
    0x19, 0x01,     /* Usage Minimum(0x1) */
    0x29, 0x03,     /* Usage Maximum(0x3) */
    0x15, 0x00,     /* Logical Minimum(0x0) */
    0x25, 0x01,     /* Logical Maximum(0x1) */
    0x95, 0x03,     /* Report Count(0x3) */
    0x75, 0x01,     /* Report Size(0x1) */
    0x81, 0x02,     /* Input(3 button bit) */
    0x95, 0x01,     /* Report Count(0x1) */
    0x75, 0x05,     /* Report Size(0x5) */
    0x81, 0x01,     /* Input(5 bit padding) */
    0x05, 0x01,     /* Usage Page(Generic Desktop Controls) */
    0x09, 0x30,     /* Usage(X) */
    0x09, 0x31,     /* Usage(Y) */
```

```
    0x15, 0x81,     /* Logical Minimum(0x81)(-127) */
    0x25, 0x7F,     /* Logical Maximum(0x7F)(127) */
    0x75, 0x08,     /* Report Size(0x8) */
    0x95, 0x02,     /* Report Count(0x2) */
    0x81, 0x06,     /* Input(1 byte) */
    0xC0,           /* End Collection */
    0xC0,           /* End Collection */
};
```

2.4.2 組態描述元

USB 主機需要組態描述元(Configuration Descriptor)來辨識裝置。gu8ConfigDescriptor 是此範例使用的組態描述元，裡面含有 2 組介面的描述。 “bNumInterfaces” 欄位設定為 2，用來讓 USB 主機知道此複合裝置有 2 組介面。2 組介面描述元(Interface Descriptor)，HID 描述元(HID descriptor)和端點描述元(Endpoint Descriptor)，依序列於組態描述元之後。

```
/*!<USB Configure Descriptor */
const uint8_t gu8ConfigDescriptor[] =
{
   /* Configuration Descriptor */
    LEN_CONFIG,                     /* bLength             */
    DESC_CONFIG,                    /* bDescriptorType     */
    LEN_TOTAL & 0x00FF,             /* wTotalLength        */
    (LEN_TOTAL & 0xFF00) >> 8,
    0x02,                           /* bNumInterfaces      */
    0x01,                           /* bConfigurationValue */
    0x00,                           /* iConfiguration      */
    0xC0,                           /* bmAttributes        */
    0x32,                           /* MaxPower            */

    /* Interface Descriptor */
    LEN_INTERFACE,                  /* bLength             */
    DESC_INTERFACE,                 /* bDescriptorType     */
    0x00,                           /* bInterfaceNumber    */
    0x00,                           /* bAlternateSetting   */
    0x02,                           /* bNumEndpoints       */
    0x08,                           /* bInterfaceClass     */
    0x05,                           /* bInterfaceSubClass  */
    0x50,                           /* bInterfaceProtocol  */
    0x00,                           /* iInterface          */
```

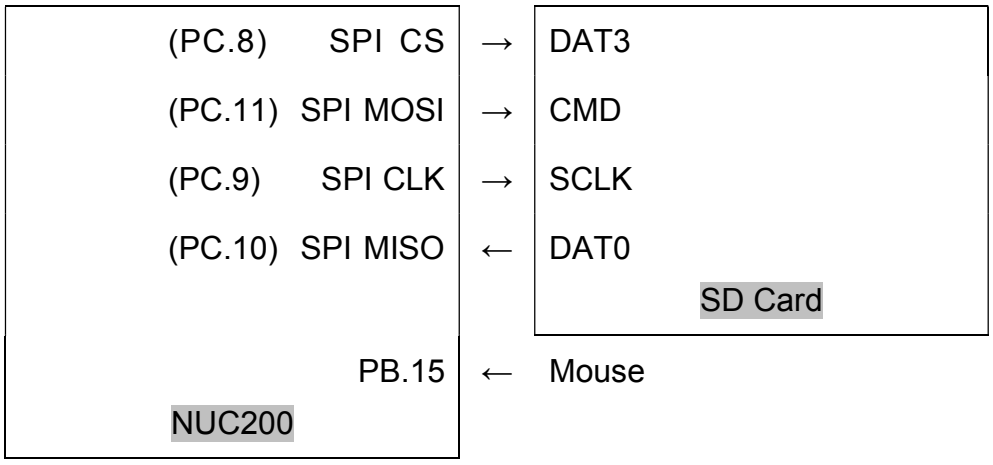
```
    /* Endpoint Descriptor */
    LEN_ENDPOINT,                   /* bLength             */
    DESC_ENDPOINT,                  /* bDescriptorType     */
    (BULK_IN_EP_NUM | EP_INPUT),    /* bEndpointAddress    */
    EP_BULK,                        /* bmAttributes        */
    EP2_MAX_PKT_SIZE, 0x00,         /* wMaxPacketSize      */
    0x00,                           /* bInterval           */

    /* Endpoint Descriptor */
    LEN_ENDPOINT,                   /* bLength             */
    DESC_ENDPOINT,                  /* bDescriptorType     */
    BULK_OUT_EP_NUM,                /* bEndpointAddress    */
    EP_BULK,                        /* bmAttributes        */
    EP3_MAX_PKT_SIZE, 0x00,         /* wMaxPacketSize      */
    0x00,                           /* bInterval           */

    /* Interface Descriptor */
    LEN_INTERFACE,                  /* bLength             */
    DESC_INTERFACE,                 /* bDescriptorType     */
    0x01,                           /* bInterfaceNumber    */
    0x00,                           /* bAlternateSetting   */
    0x01,                           /* bNumEndpoints       */
    0x03,                           /* bInterfaceClass     */
    0x01,                           /* bInterfaceSubClass  */
    HID_MOUSE,                      /* bInterfaceProtocol  */
    0x00,                           /* iInterface          */

    /* HID Descriptor */
    LEN_HID,                        /* bLength             */
    DESC_HID,                       /* bDescriptorType     */
    0x10, 0x01,                     /* bcdHID              */
    0x00,                           /* bCountryCode        */
    0x01,                           /* bNumDescriptors     */
    DESC_HID_RPT,                   /* DbDescriptorType    */
    LEN_REPORT_DESC & 0x00FF,       /* wDescriptorLength   */
    (LEN_REPORT_DESC & 0xFF00) >> 8,

  /* Endpoint Descriptor */
    LEN_ENDPOINT,                   /* bLength             */
    DESC_ENDPOINT,                  /* bDescriptorType     */
```

```
    (INT_IN_EP_NUM | EP_INPUT),      /* bEndpointAddress    */
    EP_INT,                          /* bmAttributes        */
    EP4_MAX_PKT_SIZE & 0x00FF,       /* wMaxPacketSize      */
    (EP4_MAX_PKT_SIZE & 0xFF00) >> 8,
    HID_DEFAULT_INT_IN_INTERVAL      /* bInterval           */
};
```

3 軟體與硬體環境

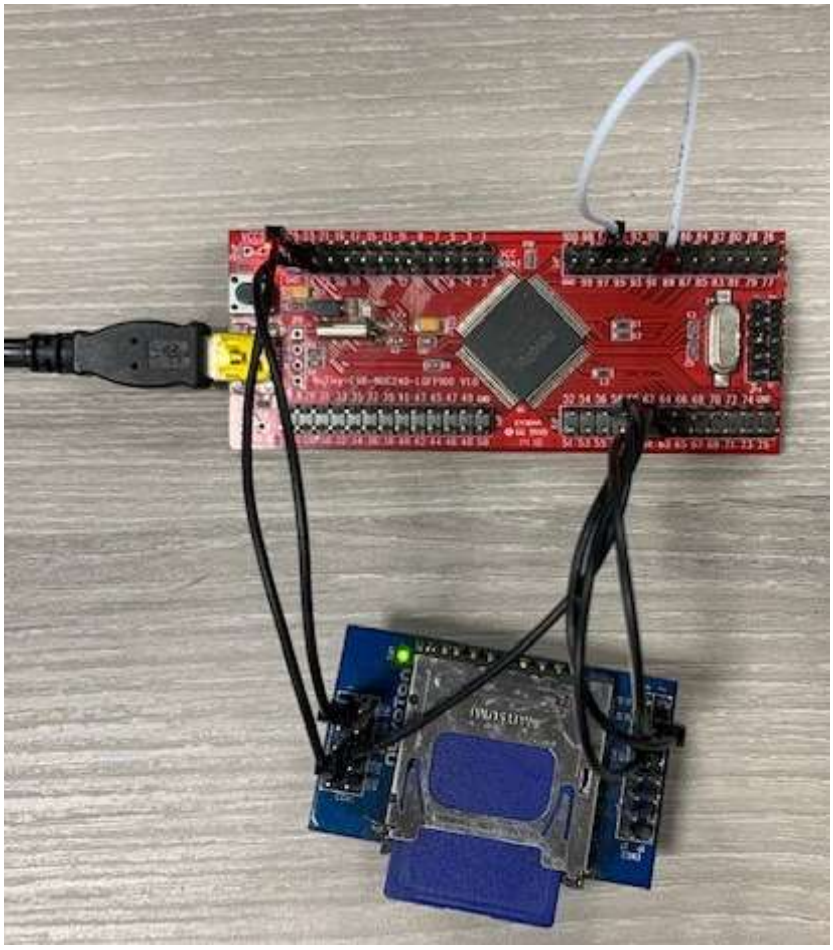
- **軟體環境**
  - BSP 版本
    - NUC200 Series BSP CMSIS V3.00.003
  - IDE 版本
    - Keil uVersion5.26
- **硬體環境**
  - 電路元件
    - NuTiny-EVB-NUC240-LQFP100 V1.00
    - SD card socket board
    - USB mini USB cable
  - 示意圖

NUC200 使用 SPI1 介面讀取 SD Card，並使用 GPB.15 模擬滑鼠輸入。以下是 NUC200 與 SD Card 的連接方式。



- SPI CS 是 Chip Select 連接 SD card 的 DAT3 pin

- SPI MOSI 是 NUC200 的資料輸出，連接 SD card 的 DAT0 pin (必須連接上拉電阻)。
- NUC200 SPI SCK 連接 SD card Clock。
- SPI MISO 是 NUC200 的資料輸入，連接 SD card 的 CMD pin (必須連接上拉電阻)。



4 目錄資訊

- EC_NUC200_USBD_MSD_SPI_Mode_SDCard_HID_Mouse_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

5 如何執行範例程式

1. 根據目錄資訊章節進入 ExampleCode 路徑中的 KEIL 資料夾，雙擊 USBD_MSD_SPI_Mode_SDCard_HID_Mouse.uvproj。
2. 進入編譯模式介面
   a. 編譯
   b. 下載代碼至記憶體
   c. 進入 / 離開除錯模式
3. 進入除錯模式介面
   a. 執行代碼

6 修訂紀錄

Date	Revision	Description
Nov.22, 2019	1.00	1. 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.