# 实作使用者自定义命令(Vendor Command)

NuMicro® 32 位系列微控制器范例代码介绍

## 文件信息

| | |
|---|---|
| 代码简述 | 示范如何扩充用户自定义指令(Vendor Command)。 |
| BSP 版本 | NUC123 Series BSP CMSIS V3.01.001 |
| 开发平台 | NuTiny-EVB-NUC123-LQFP64 v1.0 |

*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 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 功能介绍

## 1.1 简介

此范例以HID装置为例，示范如何扩充用户自定义指令(Vendor Command)。本范例会实作2个自定义指令，写入和读取的控制传输指令，并搭配Bus Hound软件观察结果，展示整个流程。

## 1.2 原理

Vendor Command建立在USB的控制传输之上。因此传输分为三个阶段，如图 1，Setup、Data和STATUS。其中SETUP和STATUS阶段是必备。DATA阶段是可选择的，视应用而定。

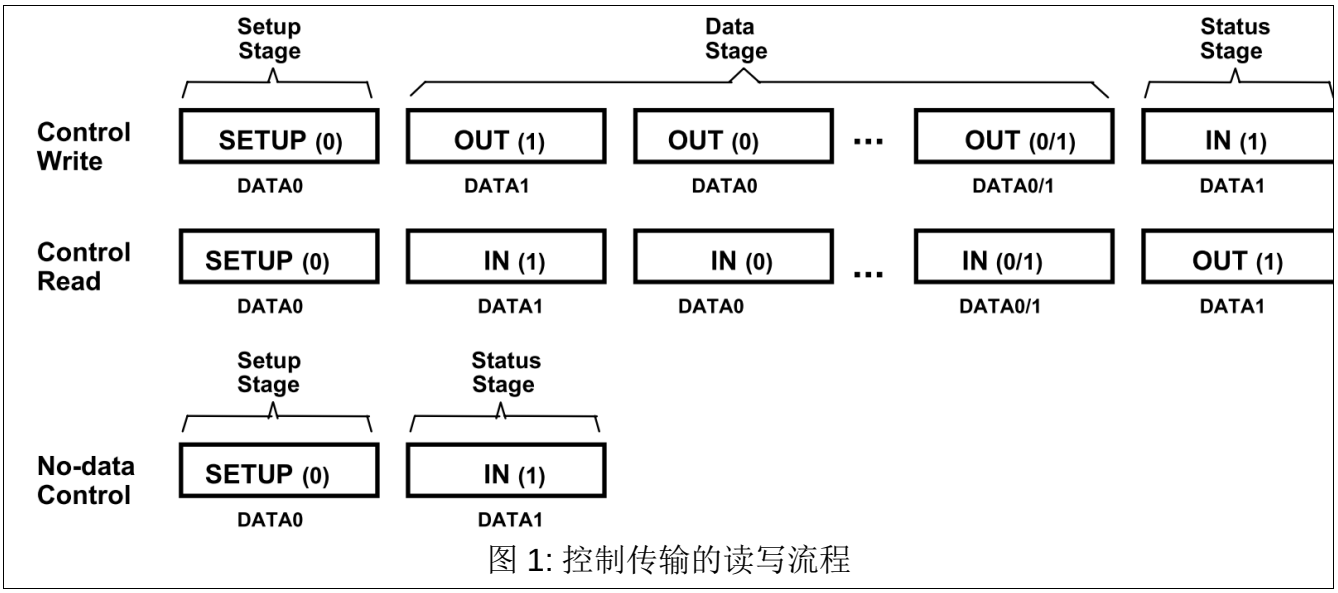



图 1: 控制传输的读写流程

| Offset | Field | Size | Value | Description |
|---|---|---|---|---|
| 0 | *bmRequestType* | 1 | Bitmap | Characteristics of request:<br>D7: Data transfer direction<br>0 = Host-to-device<br>1 = Device-to-host<br>D6...5: Type<br>0 = Standard<br>1 = Class<br>2 = Vendor<br>3 = Reserved<br>D4...0: Recipient<br>0 = Device<br>1 = Interface<br>2 = Endpoint<br>3 = Other<br>4...31 = Reserved |
| 1 | *bRequest* | 1 | Value | Specific request (refer to Table 9-3) |
| 2 | *wValue* | 2 | Value | Word-sized field that varies according to request |
| 4 | *wIndex* | 2 | Index or Offset | Word-sized field that varies according to request; typically used to pass an index or offset |
| 6 | *wLength* | 2 | Count | Number of bytes to transfer if there is a Data stage |

图 2: Setup 封包的数据格式

本次实验要实作读取和写入64 字节的数据。设定阶段的命令须符合USB控制传输的设定封包的格式，如图 2。因此我们定义了表 1的命令格式，其中的bmRequestType为Vendor Command的命令分类和传输方向；bRequest的内容即是这次的自定义Vendor Command的句柄，将写入代码设为0x1，读取代码设为0x2；wLength是要传输的数据长度。其他未提到的字段此次皆未使用，设为0。

| Vendor-Defined Command | bmRequestType | bRequest | wValue | wIndex | wLength |
|---|---|---|---|---|---|
| **VENDOR_WRITE_TEST** | 0x40 | 0x01 | 0 | 0 | 0x40 |
| **VENDOR_READ_TEST** | 0xC0 | 0x02 | 0 | 0 | 0x40 |

表 1: Vendor Command 的 SETUP 数据格式

## 1.3 执行结果

本范例使用Bus Hound来控制传输测试。本装置连接上PC后，打开Bus Hound，可以在Device Tree( 图 3)找到此装置，可以看到VID和PID分别是十六进制的0416和B001。照着图 3的操作步骤可进入指令控制(Bus Commander)窗口。

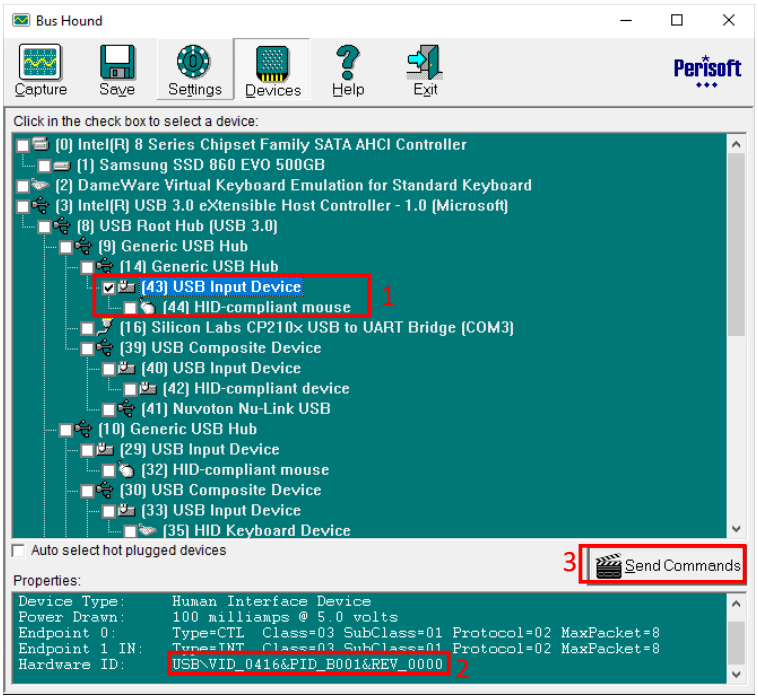



图 3: Bus Hound 界面

### 1.3.1 Read Command

在指令控制窗口，依照图 4的操作步骤，可观察指令的执行结果:

1. 在终端软件透过 UART 的输出，观察初始的缓冲区数值。按下任意键，可以显示缓冲区的初始值，0x0 ~ 0x3F。

2. 在 Bus Command 窗口输入表 1 的读取指令。

3. 按下”Run”按钮，执行指令。

4. 可以观察 Bus Hound 读取到的数值与步骤 1 显示的数值相符，代表传输完成。

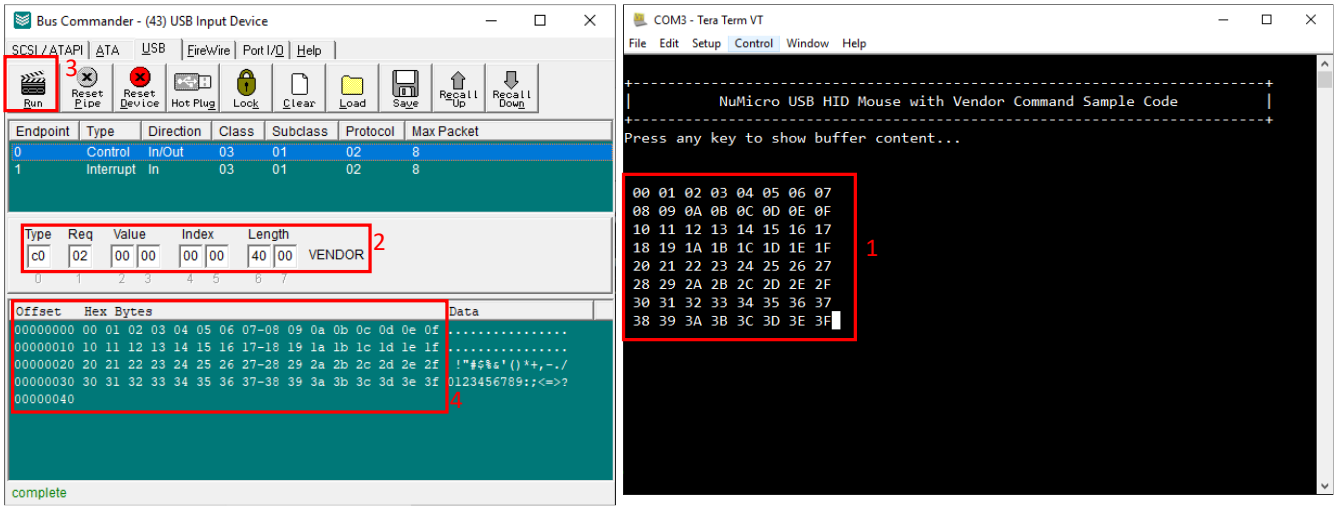



图 4: 读取指令的操作

### 1.3.2 Write Command

在指令控制窗口，依照图 5的操作步骤，观察指令的执行结果。

1. 对终端窗口按下任意键，显示目前的缓冲区数值为 0x0 ~ 0x3f。
2. 在 Bus Commander 窗口输入表 1 的写入指令。
3. 于 Bus Commander 窗口的数据区填入欲传输的数据。
4. 按下 Run 的按钮，USB 主机就会执行控制传输。
5. 在终端窗口按下任意键，可看到步骤 3 的数据已经被写入至装置端的缓冲区内。

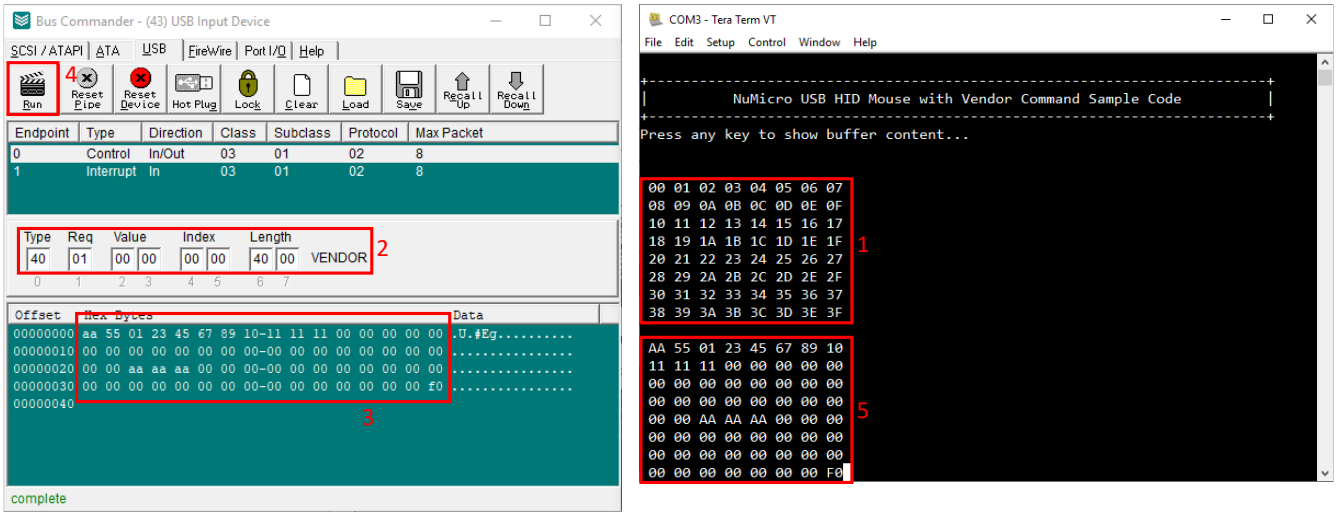



图 5:写入指令的操作

# 2 代码介绍

## 2.1 main.c

将USB装置驱动的回调函数 (callback function)，设为VENDOR_Request()。当收到用户自定义命令后，USB 装置驱动就会呼叫的处理函数。

```
    /* Set Callback function for Vendor Command */
    USBD_SetVendorRequest(VENDOR_Request);
```

## 2.2 hid_vendor_command.h, hid_vendor_command.c

hid_vendor_command.h里面定义”写入指令”为0x1，”读取指令”为0x2。

```
/*!<USB VENDOR CMD */
#define VENDOR_WRITE_TEST 0x1
#define VENDOR_READ_TEST  0x2
```

hid_vendor_command.c里的VENDOR_Request函数，实作了Vendor Command的处理。进入程序的开头，先把SETUP封包的数据储存，供后续判断使用。

```
void VENDOR_Request(void)
{
    uint8_t buf[8];
    uint32_t wLength;

    USBD_GetSetupPacket(buf);

    wLength = buf[7];
    wLength = buf[6] | (wLength << 8);
```

buf[0]是bmRequestType字段，见表 1，提供数据方向和指令种类。这里把数据分为两个方向处理。由于USBD 驱动已经处理完设定阶段，VENDOR_Request函数只需负责接下来的数据阶段和状态阶段。

数据阶段的处理，只需识别命令字段 (buf[1]) 是不是欲处理的命令，并呼叫USBD_PrepareCtrlIn()并设定存放数据的缓冲区地址和长度。

```
    if(buf[0] & 0x80)    /* request data transfer direction */
    {
        /* Device to host */
        switch(buf[1])
        {
            case VENDOR_READ_TEST:
             {
                wLength = Minimum(wLength, sizeof(g_au8temp));

                /* Data stage */
               USBD_PrepareCtrlIn((uint8_t *)g_au8temp, wLength);
```

状态阶段的处理，须设定TOGGLE FLAG为1，并呼叫USBD_PrepareCtrlOut();由于没有数据须要经由端点0回传至USB主机，因此参数皆设为0。

```
               /* Status stage */
               USBD_SET_DATA1(EP1);
               USBD_PrepareCtrlOut(0, 0);
```

```
                break;
             }

            default:
            {
                /* Setup error, stall the device */
                USBD_SetStall(EP0);
                USBD_SetStall(EP1);
                break;
            }
        }
    }
```

当收到写入指令(VENDOR_WRITE_TEST)后，呼叫USB driver的USBD_PrepareCtrlOut()，设定暂存数据的地址和长度，做数据阶段的处理。STATUS阶段不须回传数据，因此数据回传长度设为0。当传输结束后，收到的数据会存放在g_au8temp的数组内。

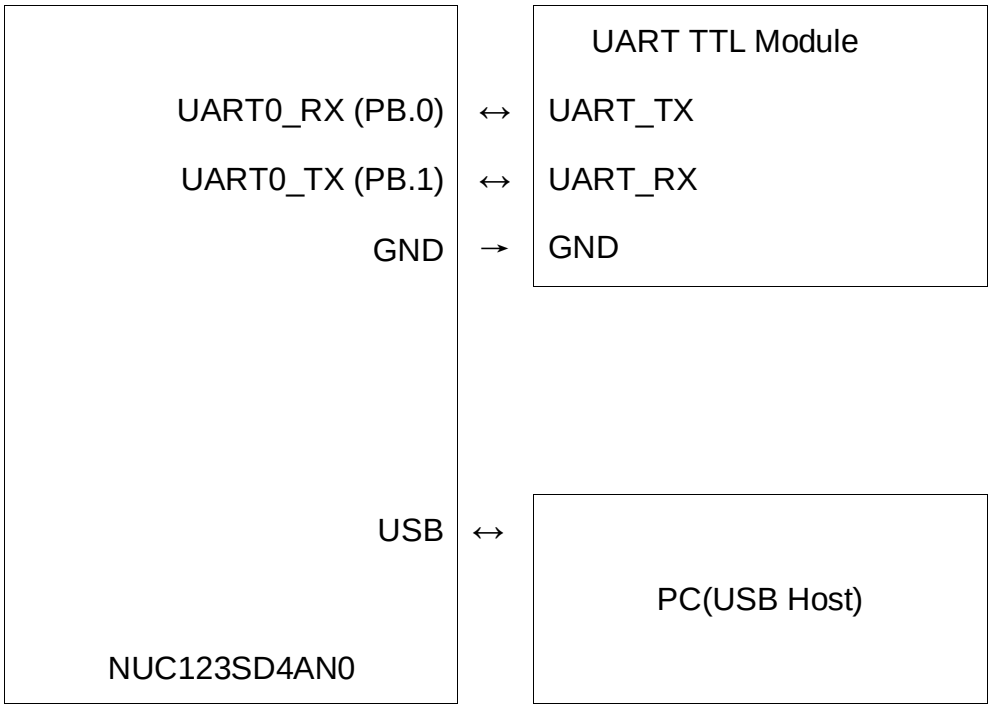
```
    else
    {
        /* Host to device */
        switch(buf[1])
        {
            case VENDOR_WRITE_TEST:
            {
                /* Data stage*/
                USBD_PrepareCtrlOut(g_au8temp, wLength);

                /* Status stage */
                USBD_SET_DATA1(EP0);
                USBD_SET_PAYLOAD_LEN(EP0, 0);
                break;
            }

            default:
            {
                /* Setup error, stall the device */
                USBD_SetStall(EP0);
                USBD_SetStall(EP1);
                break;
            }
        }
    }
}
```

# 3 软件与硬件环境

- 软件环境
  - BSP 版本
    - NUC123 Series BSP CMSIS V3.01.001
  - IDE 版本
    - Keil uVersion 5.26
  - 测试软件
    - 终端窗口软件，例如 putty(https://www.putty.org/)。
    - Bus Hound(http://www.perisoft.net/bushound/)
- 硬件环境
  - 电路组件
    - NuTiny-EVB-NUC123-LQFP64 v1.0 C
    - USB mini USB cable
    - USB-UART TTL
  - 示意图



# 4 目录信息

| | |
|---|---|
| EC_NUC123_USBD_HID_Vendor_Command_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 |
| KEIL | KEIL project file |

# 5 如何执行范例程序

1. 根据目录信息章节进入 ExampleCode 路径中的 KEIL 文件夹，双击 NUC123_USBD_HID_Vendor_Command.uvproj。
2. 进入编译模式接口
   a. 编译
   b. 下载代码至内存
   c. 进入 / 离开除错模式
3. 进入除错模式接口
   a. 执行代码，可按下 RESET 键直接执行，并从 Console 窗口观察结果。

# 6 修订纪录

| Date | Revision | Description |
|---|---|---|
| Aug. 1, 2019 | 1.00 | 1. 初始发布. |

**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.*