包含**HID** 鼠标功能并使用**SPI Mode**读取**SD**卡的**USB**储存装置

NuMicro® 32 位系列微控制器范例代码介绍

文件信息

代码简述	此范例透过 SPI 接口读取 SD 卡作为 USB 储存装置并支持 HID 鼠标功能
BSP 版本	NUC200 Series BSP CMSIS V3.00.003
开发平台	NuTiny-EVB-NUC240-LQFP100 V1.00

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

此范例示范包含HID鼠标功能的USB储存装置，透过SPI接口读取SD Card，并使用GPIO仿真鼠标输入。

1.2 原理

1.2.1 SPI Mode SD Card

SD Card支持两种运作模式：SD Card模式与SPI模式，相关讯号定义于表格 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 卡运作模式讯号定义

此范例提供SPI Mode SD Card命令与初始化流程的原始码。

1.2.2 USB 装置

此范例程序复合装置有2组接口，使用MSC(Mass Storage Device)与HID(Human Interface Devices)标准实作。一个USB装置一定会有一个控制管线(端点0)，用于控制传输。其他端点可以根据应用而有不同的设定。此范例中的控制管线由装置端硬件的端点0和端点1构成，从USB主机的角度看来，只会看到端点0；接口0包含一个批量的输入端点与一个批量的输出端点，用于批量传输(Bulk Transfer)；接口1包含一个中断的输入端点，此端点用于输入方向的中断传输(Interrupt Transfer)。这些接口和端点的关系列于表格 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 2: Bulk IN
		endpoint 3: Bulk OUT

	endpoint 3: Bulk OUT	
HID Mouse	Interface 1 endpoint 4: Interrupt IN	endpoint 4: Interrupt IN

表格 2 复合装置的接口和端点组态

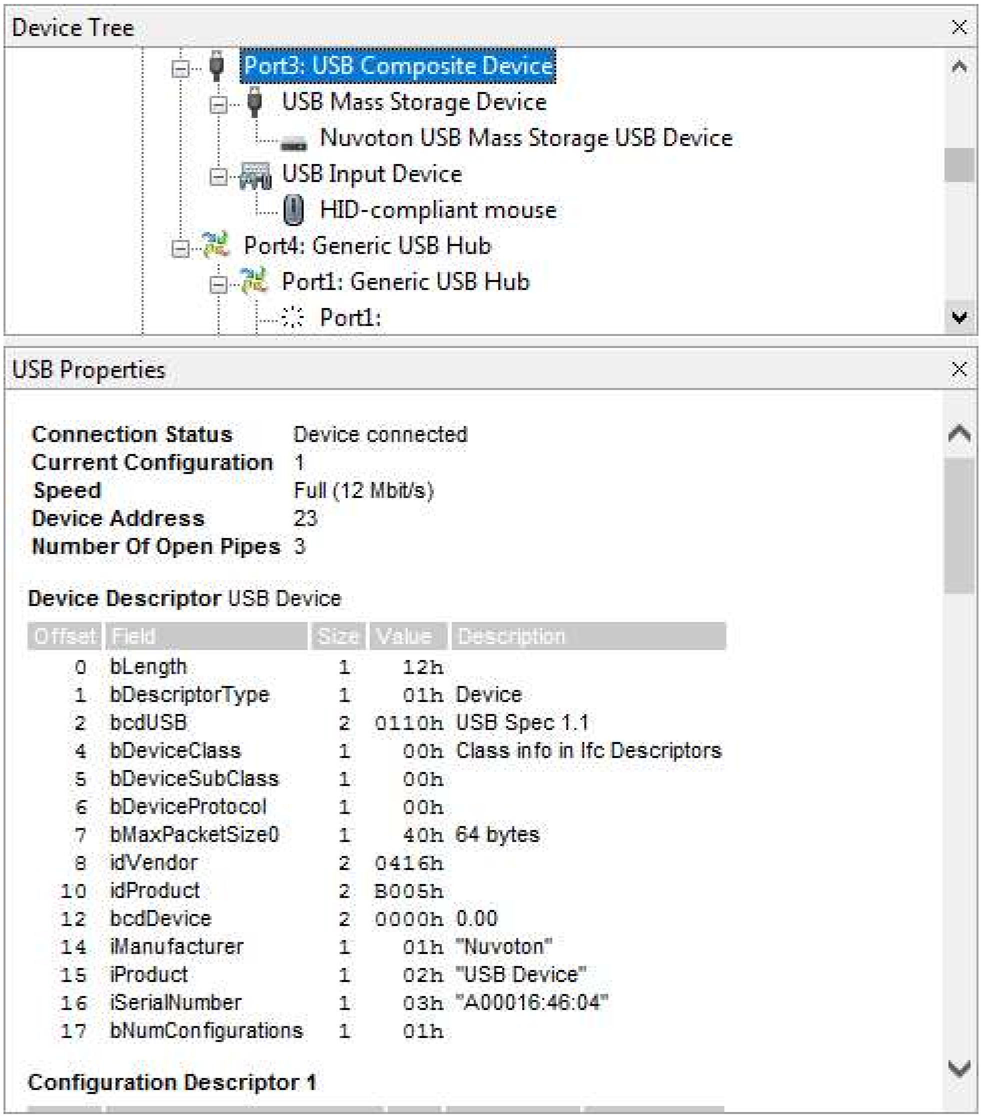
接口1是一个HID鼠标装置，使用表格 3的数据格式回传数据。

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 鼠标报告格式

1.3 执行结果

此范例需与连接个人计算机才能运行。此范例程序执行时，使用USBLyzer我们可以获得图表 1 的装置列表。可以看到此复合装置有2组接口。



- Port3: USB Composite Device
  - USB Mass Storage Device
    - Nuvoton USB Mass Storage USB Device
  - USB Input Device
    - HID-compliant mouse
- Port4: Generic USB Hub
  - Port1: Generic USB Hub
    - Port1:

Connection Status	Device connected
Current Configuration	1
Speed	Full (12 Mbit/s)
Device Address	23
Number Of Open Pipes	3

Device Descriptor USB Device

Offset	Field	Size	Value	Description
0	bLength	1	12h	
1	bDescriptorType	1	01h	Device
2	bcdUSB	2	0110h	USB Spec 1.1
4	bDeviceClass	1	00h	Class info in Ifc Descriptors
5	bDeviceSubClass	1	00h	
6	bDeviceProtocol	1	00h	
7	bMaxPacketSize0	1	40h	64 bytes
8	idVendor	2	0416h	
10	idProduct	2	B005h	
12	bcdDevice	2	0000h	0.00
14	iManufacturer	1	01h	"Nuvoton"
15	iProduct	1	02h	"USB Device"
16	iSerialNumber	1	03h	"A00016:46:04"
17	bNumConfigurations	1	01h	

Configuration Descriptor 1

图表 1 装置列表

2 代码介绍

2.1 SPI Mode SD Card

设定 SPI 接口与 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;
    }
```

读取 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;
        }
    }
}
```

写入 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 鼠标

USB 主机会发出 IN Token 向端点 4 请求数据，当 IN Token 收到时，会执行端点 4 的中断处理程序设定并回传数据。以此范例为例，当 GPB15 电位为 0 时，设定 buf[2] (表格 3 的 X 轴字段) 为 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 类别请求

MSC_ClassRequest()包含了 MSC 与 HID 类别请求。

```
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 装置描述符

2.4.1 HID 报告描述符

HID_MouseReportDescriptor 是鼠标的 HID 报告描述符元。此描述元定义了表格 3 中的数据格式。此数据格式内有，3 个按钮、X 轴、Y 轴和滚轮。

```
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 组态描述元

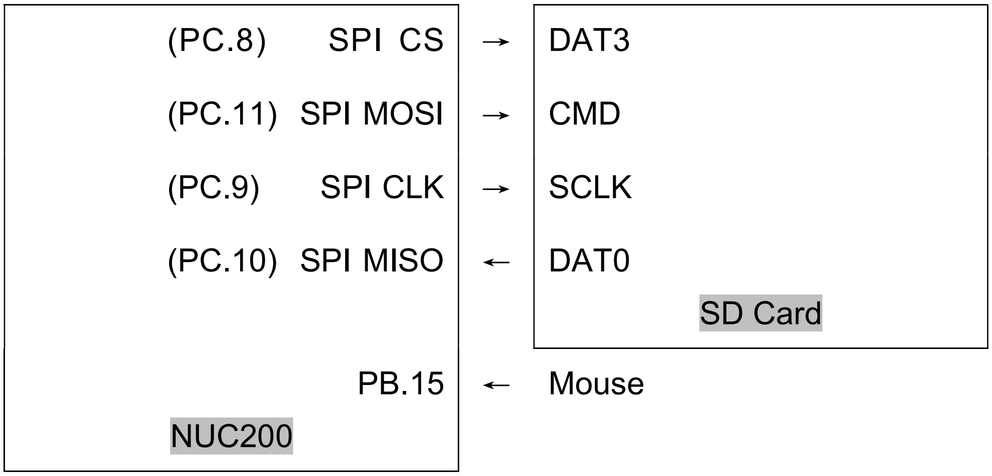
USB 主机需要组态描述元(Configuration Descriptor)来辨识装置。gu8ConfigDescriptor 是此范例使用的组态描述元，里面含有 2 组界面的描述。“bNumInterfaces”字段设定为 2，用来让 USB 主机知道此复合装置有 2 组接口。2 组界面描述元(Interface Descriptor)，HID 描述元(HID descriptor)和端点描述元(Endpoint Descriptor)，依序列于组态描述元之后。

```
/*!<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           */
};
```

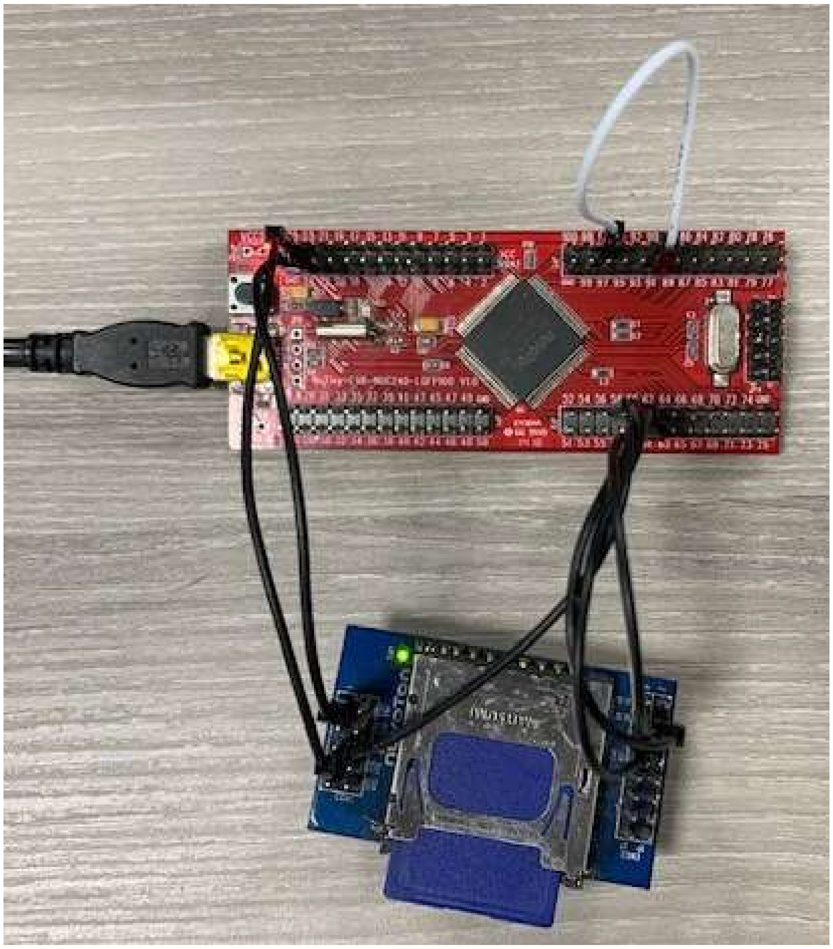
3 软件与硬件环境

- 软件环境
  - BSP 版本
    - NUC200 Series BSP CMSIS V3.00.003
  - IDE 版本
    - Keil uVersion5.26
- 硬件环境
  - 电路组件
    - NuTiny-EVB-NUC240-LQFP100 V1.00
    - SD card socket board
    - USB mini USB cable
  - 示意图

NUC200 使用 SPI1 接口读取 SD Card，并使用 GPB.15 仿真鼠标输入。以下是 NUC200 与 SD Card 的连接方式。



- SPI CS 是 Chip Select 连接 SD card 的 DAT3 pin
- SPI MOSI 是 NUC200 的数据输出，连接 SD card 的 DAT0 pin (必须连接上拉电阻)。
- NUC200 SPI SCK 连接 SD card Clock。
- SPI MISO 是 NUC200 的数据输入，连接 SD card 的 CMD pin (必须连接上拉电阻)。



4 目录信息

🗁 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 USBD_MSD_SPI_Mode_SDCard_HID_Mouse.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

6 修订纪录

Date	Revision	Description
Nov.22, 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.