Application Note

32-bit Cortex™-M0 MCU
NuMicro® Family

USB MassStorage 应用注解和示例程序

目录

1 简介

本文利用***AN_1020_EN.ZIP***中的示例程序 “Smpl_UDC” 项目，阐述了用相关如何USB库和源代码，创建一个USB MassStorage 设备的办法。

1.1 目标读者

本文向程序员阐述了如何实现一个USB应用的过程，文章假定读者熟悉USB(Universal Serial Bus) 1.1协议，并熟悉MassStorage相关的USB类定义。

1.2 例子功能

- 支持自定义VID (Vendor ID) 和 PID (Product ID).
- 支持自定义用作存储的设备

2 代码片断

2.1 主函数main (见 Smpl_UDC.c)

（请参考调用过程一节）

在主函数里，首先设定PLL 时钟到48MHz，这样USB PHY才可以工作。

在硬件相关设定完成后，需要调用udcInit() 重置并且初始化USB的端点状态。udcInit同时也会检查USB连接状态并将全局变量g_u8UsbState设定到对应的状态值。

在USB初始化完成后，调用udcFlashInit() 初始化相关的存储设备。用户如果要用自定义的存储设备，可以在这个函数里加入初始化自定义设备的代码。

最后一步将调用udcMassBulk()。udcMassBulk()通过不断的轮询USB的状态发现USB事件，一旦发现有USB事件，就调用对应的事件处理函数进行处理。

```
int32_t main(void)
{
      ...

      /* Initialize USB Device function */
      udcInit();
      /* Initialize mass storage device */
      udcFlashInit();
      /* Start USB Mass Storage */
      udcMassBulk();

      return 0;
}
```

2.2 轮询流程 (见MassStorage.c)

正如您看到的，函数udcMassBuld() 中只有一个循环不断地调用函数UsbIsr()。UsbIsr() 并不是真正的中断处理寒暑。将UsbIsr()放在主进程中运行，看起来简单易懂，又能实现功能。（更复杂的应用场景，也有可能将类似于UsbIsr()功能的函数，放到终端中运行。）

```
void udcMassBulk(void)
{
    /* Handler the USB ISR by polling */
      while(1)
      {
```

```
            UsbIsr();
        }
}


void UsbIsr(void)
{
        uint32_t u32EVF = _DRVUSB_GET_EVF();
        if (u32EVF & EVF_FLD)
        {
            /* Handle the USB attached/detached event */
               UsbFdt();
        }
        else if(u32EVF & EVF_BUS)
        {
            /* Handle the USB bus event: Reset, Suspend, and Resume */
               UsbBus();
        }
        else if(u32EVF & EVF_USB)
        {
            /* Handle the USB Protocol/Clase event */
               UsbUsb(u32EVF);
    }
}
```

在上面的代码中， UsbIsr() 检查USB的事件标志，如果发现有插入 / 拔除事件、或者USB总线上的事件（比如重置、休眠、唤起等）、或者USB协议定义的数据交互事件，就会跳转到对应的响应函数进行处理。

在所有的响应函数中，有两个函数尤其重要，它们是UsbBulkOutAck和UsbBulkInAck，这两个函数会被UsbUsb()调用到。这两个函数完成了和主机USB的大部分数据通信工作。作为一个USB MassStorage设备需要支持的SCSI命令，也是由UsbBulkOutAck和UsbBulkInAck实现的。在我们的例子代码里，这些 SCSI命令，通过结构成员struct CBW::u8OPCode来描述。

(更进一步，如果用户可以尝试跟踪 “u8OPCode” 的值，并加入更多地或自定义的SCSI命令的支持。Windows用户要获知如何在程序里发送SCSI命令的办法，请到网络上搜索关键字DeviceIOControl和SCSI_IOCTL_DATA_OUT。)

USB事件的响应函数，有调用到另外两个函数SpiRead和SpiWrite。这两个函数提供给客户，用来定制

存储设备的输入输出操作。用户可以重写这两个函数，以便支持新的存储设备。原型函数如下：

```
void SpiRead(uint32_t addr, uint32_t size, uint32_t buffer)
{
    /* This is low level read function of USB Mass Storage */
}

void SpiWrite(uint32_t addr, uint32_t size, uint32_t buffer)
{
    /* This is low level write function of USB Mass Storage */
}
```

2.3 如何使用新的存储设备

要使用新的存储设备，仅需重写以下三个函数：

- uint8_t udcFlashInit(void)

  初始化存储设备

- void SpiRead(uint32_t addr, uint32_t size, uint32_t buffer)

  从设备读数据

- void SpiWrite(uint32_t addr, uint32_t size, uint32_t buffer)

  向设备写数据

3 调用过程

3.1 USB设备的运行

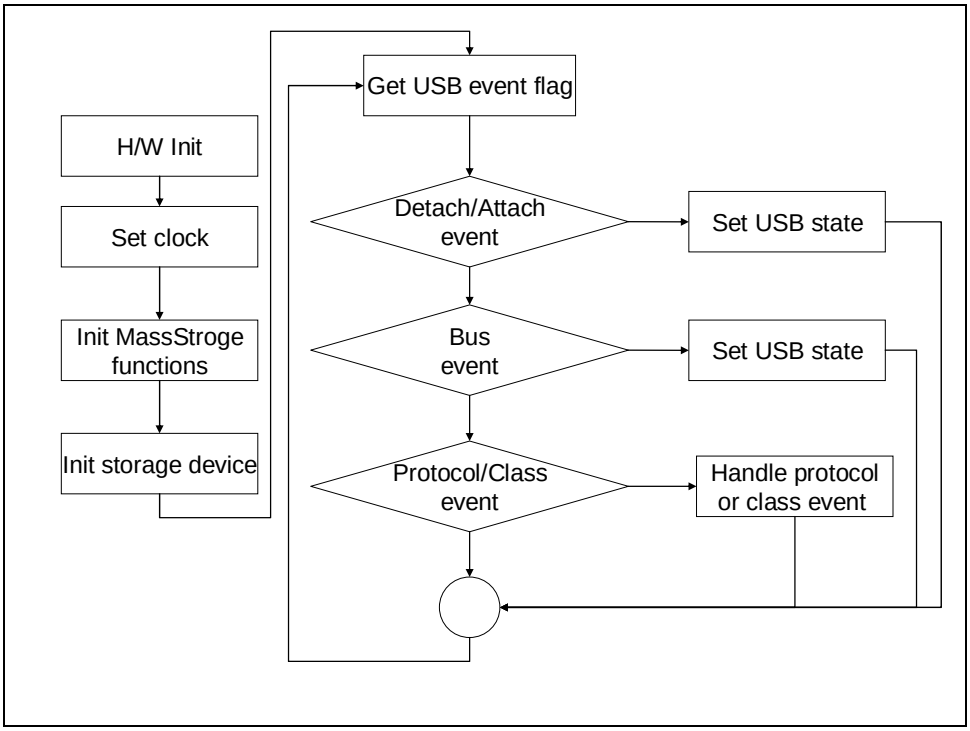

Figure 3-1 运行 USB 设备的流程

1. 初始化硬件
2. 为USB设备设定好时钟
3. 初始化USB MassStorage 功能函数
4. 初始化存储设备
5. 查询USB 设备状态
   1) 查询并处理USB插入/拔出事件
   2) 查询并处理USB总线事件，例如重置、休眠、唤起。
   3) 查询并处理USB协议定义的USB数据交互事件
6. 转到步骤5，继续查询USB设备状态

3.2 定制存储设备

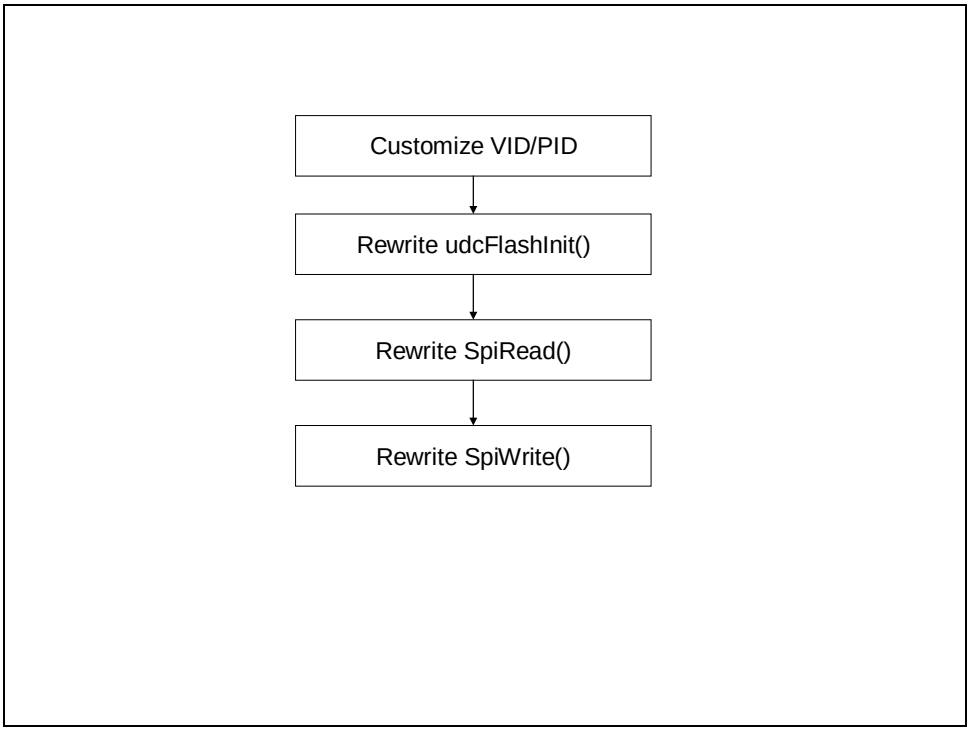

Figure 3-2 定义一个 USB MassStorage 设备所使用的存储设备的过程

1. 如有必要，可以自定义VID / PID
2. 为新的设备重新实现函数udcFlashInit(), SpiRead() 和SpiWrite()
3. 请参考 2.3 章节 查看上面三个函数的描述。

3.3 API 用法参考

- USB Driver Reference Guide.doc

4 运行环境和结果

4.1 测试 Smpl_UDC

您可以用Keil MDK编译并下载运行该USB MassStorage的示例代码(Smpl_UDC)。在Keil MDK下运行的时候，需要一块NUC1xx 系列开发板和一个ICE。.如果程序已经烧录在Flash中了，用户重启动开发板后，也能直接看到示例程序运行的结果。

4.2 结果

程序运行的时候，开发板将被模拟成一只U盘。一旦USB插入装有Windows操作系统的PC上，资源管理器的窗口里将可以看到一个新的盘符。

版本历史

REV.	DATE	DESCRIPTION
0.01	March 11, 2010	1. Initially issued.

Important Notice

Nuvoton products are not designed, intended, authorized or warranted for use as components in systems or equipment intended for surgical implantation, atomic energy control instruments, airplane or spaceship instruments, transportation instruments, traffic signal instruments, combustion control instruments, or for other applications intended to support or sustain life. Further more, Nuvoton products are not intended for applications wherein failure of Nuvoton products could result or lead to a situation wherein personal injury, death or severe property or environmental damage could occur.

Nuvoton customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify Nuvoton for any damages resulting from such improper use or sales.

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.