小川工作室编写，本书为 LM3S 的 USB 芯片编写，上传的均为草稿，还有没修改，可能还有很多地方不足，希望各位网友原谅！

QQ: 2609828265

TEL: 15882446438

E-mail: paulhyde@126.com

第十章 Composite 设备

10.1 Composite 设备介绍

USB 的 Composite 类是 USB 复合设备类，一个 USB 设备具有多种设备功能，比如一个 USB 设备同时具有鼠标和键盘功能。单一的 USB 设备开发相对简单，但在很多时候使用的 USB 设备具有多种功能。Composite 类可以满足这种要求。

10.2 Composite 数据类型

usbdcomp.h 中已经定义好 composite 设备类中使用的所有数据类型和函数。下面介绍 composite 设备类使用的数据类型。

```
typedef struct
{
    const tDeviceInfo *pDeviceInfo;
    const tConfigHeader *psConfigHeader;
    unsigned char ucIfaceOffset;
} tUSBDCompositeEntry;
```

tUSBDCompositeEntry，定义 composite 设备信息。定义在 usbdcomp.h。

```
typedef struct
{
    unsigned long ulUSBBase;
    tDeviceInfo *psDevInfo;
    tConfigDescriptor sConfigDescriptor;
    tDeviceDescriptor sDeviceDescriptor;
    tConfigHeader sCompConfigHeader;
    tConfigSection psCompSections[2];
    tConfigSection *ppsCompSections[2];
    unsigned long ulDataSize;
    unsigned char *pucData;
}
tCompositeInstance;
```

tCompositeInstance，设备类实例。定义了 Composite 设备类的 USB 基地址、设备信息、IN 端点、OUT 端点等信息。

```
typedef struct
```

```
{
    const tDeviceInfo *psDevice;
    void *pvInstance;
}
tCompositeEntry;
```

tCompositeEntry，Composite 各设备的设备信息。

```
typedef struct
{
    unsigned short usVID;
    unsigned short usPID;
    unsigned short usMaxPowermA;
    unsigned char ucPwrAttributes;
    tUSBCallback pfnCallback;
    const unsigned char * const *ppStringDescriptors;
    unsigned long ulNumStringDescriptors;
    unsigned long ulNumDevices;
    tCompositeEntry *psDevices;
    tCompositeInstance *psPrivateData;
}
tUSBDCompositeDevice;
```

tUSBDCompositeDevice，Composite 设备类，定义了 VID、PID、电源属性、字符串描述符等，还包括了 Composite 设备类实例。其它设备描述符、配置信息通过 API 函数储入 tCompositeInstance 定义的 Composite 设备实例中。

10.3 API 函数

在 Composite 设备类 API 库中定义了 2 个函数，完成 USB Composite 设备初始化、配置及数据处理。下面为 usbdcomp.h 中定义的 API 函数：

```
 void *USBDCompositeInit(unsigned long ulIndex,
                              tUSBDCompositeDevice *psCompDevice,
                              unsigned long ulSize,
                              unsigned char *pucData);
void USBDCompositeTerm(void *pvInstance);
```

```
void *USBDCompositeInit(unsigned long ulIndex,
                              tUSBDCompositeDevice *psCompDevice,
                              unsigned long ulSize,
                              unsigned char *pucData);
```

作用：初始化 Composite 设备硬件、协议，把其它配置参数填入 psCompDevice 实例中。

参数：ulIndex，USB 模块代码，固定值：USB_BASE0。psMSCDevice，MSC 设备类。

返回：指向配置后的 tUSBDCompositeDevice。

void USBDCompositeTerm(void *pvInstance);

作用：结束 Composite 设备。

参数：pvInstance，指向 tUSBDCompositeDevice。

返回：无。

在这些函数中 USBDCompositeInit 函数最重要，用于处理各子设备信息，保存所有子设备配置及其它数据。

9.4 Composite 设备开发

Composite 设备开发只需要 3 步就能完成：各子设备配置、完善接口函数；Composite 设备配置、协调；各子设备数据处理。如图 2 所示，

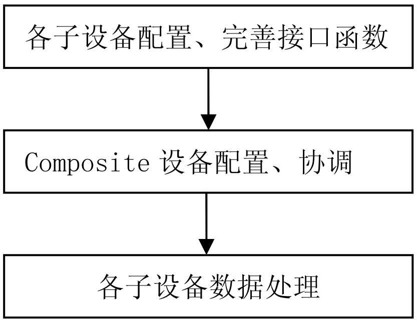


图 2

下面以“电子教鞭”实例说明使用 USB 库开发 USB Composite 设备过程，电子教鞭有两个重要功能，U 盘功能和控制功能。所以要做两个子类：大容量存储类与键盘类：

第一步：各子设备配置、完善接口函数：

```
#define DESCRIPTOR_DATA_SIZE    (COMPOSITE_DHID_SIZE + COMPOSITE_DMSC_SIZE)
unsigned char g_pucDescriptorData[DESCRIPTOR_DATA_SIZE];

//声明函数原型
unsigned long USBDMSCEventCallback(void *pvCBData, unsigned long ulEvent,
                                       unsigned long ulMsgParam,
                                       void *pvMsgData);
//声明函数原型
unsigned long KeyboardHandler(void *pvCBData,
                                     unsigned long ulEvent,
                                     unsigned long ulMsgData,
                                     void *pvMsgData);
unsigned long EventHandler(void *pvCBData, unsigned long ulEvent,
                                  unsigned long ulMsgData, void *pvMsgData);

const tUSBDMSCDevice g_sMSCDevice;
//msc 状态
volatile enum
{
    MSC_DEV_DISCONNECTED,
    MSC_DEV_CONNECTED,
    MSC_DEV_IDLE,
    MSC_DEV_READ,
    MSC_DEV_WRITE,
}
g_eMSCState;
//全局标志
#define FLAG_UPDATE_STATUS      1
```

```
static unsigned long g_ulFlags;
//DMA
tDMAControlTable sDMAControlTable[64] __attribute__ ((aligned(1024)));

//****************************************************************************
// 语言描述符
//****************************************************************************
const unsigned char g_pLangDescriptor[] =
{
    4,
    USB_DTYPE_STRING,
    USBShort(USB_LANG_EN_US)
};
//****************************************************************************
// 制造商 字符串 描述符
//****************************************************************************
const unsigned char g_pManufacturerString[] =
{
    (17 + 1) * 2,
    USB_DTYPE_STRING,
    ’T’, 0, ’e’, 0, ’x’, 0, ’a’, 0, ’s’, 0, ’ ’, 0, ’I’, 0, ’n’, 0, ’s’, 0,
    ’t’, 0, ’r’, 0, ’u’, 0, ’m’, 0, ’e’, 0, ’n’, 0, ’t’, 0, ’s’, 0,
};
//****************************************************************************
//产品 字符串 描述符
//****************************************************************************
const unsigned char g_pProductString[] =
{
    (19 + 1) * 2,
    USB_DTYPE_STRING,
    ’M’, 0, ’a’, 0, ’s’, 0, ’s’, 0, ’ ’, 0, ’S’, 0, ’t’, 0, ’o’, 0, ’r’, 0,
    ’a’, 0, ’g’, 0, ’e’, 0, ’ ’, 0, ’D’, 0, ’e’, 0, ’v’, 0, ’i’, 0, ’c’, 0,
    ’e’, 0
};
//****************************************************************************
//   产品 序列号 描述符
//****************************************************************************
const unsigned char g_pSerialNumberString[] =
{
    (8 + 1) * 2,
    USB_DTYPE_STRING,
    ’1’, 0, ’2’, 0, ’3’, 0, ’4’, 0, ’5’, 0, ’6’, 0, ’7’, 0, ’8’, 0
};
//*****************************************************************************
```

```
// 字符串描述符集合
//*****************************************************************************
const unsigned char * const g_pStringDescriptors[] =
{
    g_pLangDescriptor,
    g_pManufacturerString,
    g_pProductString,
    g_pSerialNumberString,
};
#define NUM_STRING_DESCRIPTORS (sizeof(g_pStringDescriptors) /                \
                                sizeof(unsigned char *))
//*****************************************************************************
//MSC 实例，配置并为设备信息提供空间
//*****************************************************************************
tMSCInstance g_sMSCInstance;
//*****************************************************************************
//msc 设备配置
//*****************************************************************************
const tUSBDMSCDevice g_sMSCDevice =
{
    USB_VID_STELLARIS,
    USB_PID_MSC,
    "TI      ",
    "Mass Storage    ",
    "1.00",
    200,
    USB_CONF_ATTR_SELF_PWR,
    g_pStringDescriptors,
    NUM_STRING_DESCRIPTORS,
    {
        USBDMSCStorageOpen,
        USBDMSCStorageClose,
        USBDMSCStorageRead,
        USBDMSCStorageWrite,
        USBDMSCStorageNumBlocks
    },
    USBDMSCEventCallback,
    &g_sMSCInstance
};
#define MSC_BUFFER_SIZE 512
//*****************************************************************************
//键盘实例，配置并为设备信息提供空间
//*****************************************************************************
tHIDKeyboardInstance g_KeyboardInstance;
```

```
//*****************************************************************************
//键盘设备配置
//*****************************************************************************
const tUSBDHIDKeyboardDevice g_sKeyboardDevice =
{
    USB_VID_STELLARIS,
     USB_VID_STELLARIS,
    200,
    USB_CONF_ATTR_SELF_PWR | USB_CONF_ATTR_RWAKE,
    KeyboardHandler,
    (void *)&g_sKeyboardDevice,
    0,
    0,
    &g_KeyboardInstance
};

//*****************************************************************************
//callback 函数
//*****************************************************************************
unsigned long USBDMSCEventCallback(void *pvCBData, unsigned long ulEvent,
                     unsigned long ulMsgParam, void *pvMsgData)
{
    switch(ulEvent)
    {
        // 正在写数据到存储设备.
        case USBD_MSC_EVENT_WRITING:
        {

            break;
        }
        //读取数据.
        case USBD_MSC_EVENT_READING:
        {
             GPIOPinWrite(GPIO_PORTF_BASE,0x10,0x10);
            break;
        }
         //空闲
        case USBD_MSC_EVENT_IDLE:
        default:
        {
              GPIOPinWrite(GPIO_PORTF_BASE,0x10,0x00);
            break;
        }
    }
    return(0);
}
//******************************************************************************
//键盘callback函数
//******************************************************************************
unsigned long KeyboardHandler(void *pvCBData, unsigned long ulEvent,
                              unsigned long ulMsgData, void *pvMsgData)
{
    switch (ulEvent)
    {
        case USB_EVENT_CONNECTED:
        {
              GPIOPinWrite(GPIO_PORTF_BASE,0x20,0x20);
            break;
        }
        case USB_EVENT_DISCONNECTED:
        {
              GPIOPinWrite(GPIO_PORTF_BASE,0x20,0x00);
            break;
        }
        case USB_EVENT_TX_COMPLETE:
        {

            break;
        }
        case USB_EVENT_SUSPEND:
        {

            break;
        }
        case USB_EVENT_RESUME:
        {
            break;
        }
        case USBD_HID_KEYB_EVENT_SET_LEDS:
        {
            break;
        }
        default:
        {
            break;
        }
    }
    return (0);
```

```
}
```

第二步：完成 Composite 设备配置、协调：

```
//*****************************************************************************
//复合设备配置
//*****************************************************************************
tCompositeEntry g_psCompDevices[]=
{
    {
        &g_sMSCDeviceInfo,
       (void  *)&g_sMSCDeviceInfo
    },
    {
        &g_sHIDDeviceInfo,
        (void  *)&g_sHIDDeviceInfo
    }
};
#define NUM_DEVICES          (sizeof(g_psCompDevices)/sizeof(tCompositeEntry))
tCompositeInstance g_CompInstance;
unsigned long xxx[10];
tUSBDCompositeDevice g_sCompDevice =
{
    USB_VID_STELLARIS,
    0x0123,
    500,
    USB_CONF_ATTR_BUS_PWR,
    EventHandler,
    g_pStringDescriptors,
    NUM_STRING_DESCRIPTORS,
    2,
    g_psCompDevices,
     xxx,
    &g_CompInstance
};
//*****************************************************************************
//复合设备 callback 函数
//*****************************************************************************
unsigned long EventHandler(void *pvCBData, unsigned long ulEvent, unsigned long ulMsgData,
             void *pvMsgData)
{
    unsigned long ulNewEvent;
    ulNewEvent = 1;
    switch(ulEvent)
    {
        case USB_EVENT_CONNECTED:
        {
            break;
        }
        case USB_EVENT_DISCONNECTED:
        {
            break;
        }
        case USB_EVENT_SUSPEND:
        {
            break;
        }
        case USB_EVENT_RESUME:
        {
            break;
        }

        default:
        {
            ulNewEvent = 0;
            break;
        }
    }
    if(ulNewEvent)
    {
    }
    return(0);
}
```

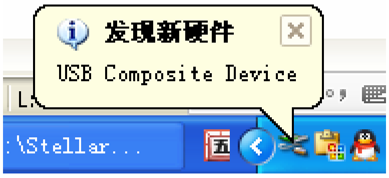
第三步：各子设备数据处理，主要是按键处理，U 盘功能自动调用底层驱动自动完成：

```
         //系统初始化。
         SysCtlLDOSet(SYSCTL_LDO_2_75V);
         SysCtlClockSet(SYSCTL_XTAL_8MHZ    |    SYSCTL_SYSDIV_8    |    SYSCTL_USE_PLL        |
SYSCTL_OSC_MAIN );
         SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
         GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE,0xf0);
         GPIOPinTypeGPIOInput(GPIO_PORTF_BASE,0x0f);
         HWREG(GPIO_PORTF_BASE+GPIO_O_PUR) |= 0x0f;
         // ucDMA 配置
        SysCtlPeripheralEnable(SYSCTL_PERIPH_UDMA);
        SysCtlDelay(10);
        uDMAControlBaseSet(&sDMAControlTable[0]);
        uDMAEnable();

         g_ulFlags = 0;
         g_eMSCState = MSC_DEV_IDLE;
```

```
    //复合设备初始化
    g_sCompDevice.psDevices[0].pvInstance =
      USBDMSCCompositeInit(0, &g_sMSCDevice);
   g_sCompDevice.psDevices[1].pvInstance =
       USBDHIDKeyboardInit(0, &g_sKeyboardDevice);
    USBDCompositeInit(0, &g_sCompDevice, DESCRIPTOR_DATA_SIZE,
                     g_pucDescriptorData);
    //初始化存储设备
    disk_initialize(0);
    while(1)
    {
       USBDHIDKeyboardKeyStateChange((void *)&g_sKeyboardDevice, HID_KEYB_CAPS_LOCK,
                                     HID_KEYB_USAGE_A,
                                     (GPIOPinRead(GPIO_PORTF_BASE, 0x0f) & GPIO_PIN_0)
                                     ? false : true);
       USBDHIDKeyboardKeyStateChange((void *)&g_sKeyboardDevice, 0,
                                     HID_KEYB_USAGE_DOWN_ARROW,
                                     (GPIOPinRead(GPIO_PORTF_BASE, 0x0f) & GPIO_PIN_1)
                                     ? false : true);
       USBDHIDKeyboardKeyStateChange((void *)&g_sKeyboardDevice, 0,
                                     HID_KEYB_USAGE_UP_ARROW,
                                     (GPIOPinRead(GPIO_PORTF_BASE, 0x0f) & GPIO_PIN_2)
                                     ? false : true);
       USBDHIDKeyboardKeyStateChange((void *)&g_sKeyboardDevice, 0,
                                     HID_KEYB_USAGE_ESCAPE,
                                     (GPIOPinRead(GPIO_PORTF_BASE, 0x0f) & GPIO_PIN_3)
                                     ? false : true);
       SysCtlDelay(SysCtlClockGet()/3000);
   }
```

使用上面三步就完成 Composite 设备开发。Composite 设备开发时要加入两个 lib 库函数：usblib.lib 和 DriverLib.lib，在启动代码中加入 USB0DeviceIntHandler 中断服务函数。以上 Composite 备开发完成，在 Win xp 下运行效果如下图所示：


发现新硬件

USB Composite Device

在电脑中可以发现多了 USB MSC 设备和 HID 设备，同时还多了一个 Composite 设备。Composite 设备开发源码较多，下面只列出一部分如下：

```
#include "inc/hw_ints.h"
#include "inc/hw_memmap.h"
#include "inc/hw_gpio.h"
#include "inc/hw_types.h"
```

```
#include "inc/hw_ints.h"
#include "driverlib/sysctl.h"
#include "driverlib/gpio.h"
#include "driverlib/interrupt.h"
#include "driverlib/rom.h"
#include "driverlib/systick.h"
#include "driverlib/usb.h"
#include "driverlib/udma.h"
#include "usblib/usblib.h"
#include "usblib/usb-ids.h"
#include "usblib/device/usbdevice.h"
#include "usblib/device/usbdmsc.h"
#include "diskio.h"
#include "usbdsdcard.h"
#include "usblib/usblib.h"
#include "usblib/usbhid.h"
#include "usblib/device/usbdhid.h"
#include "usblib/device/usbdcomp.h"
#include "usblib/device/usbdhidkeyb.h"

#define DESCRIPTOR_DATA_SIZE    (COMPOSITE_DHID_SIZE + COMPOSITE_DMSC_SIZE)
unsigned char g_pucDescriptorData[DESCRIPTOR_DATA_SIZE];

//声明函数原型
unsigned long USBDMSCEventCallback(void *pvCBData, unsigned long ulEvent,
                                       unsigned long ulMsgParam,
                                       void *pvMsgData);
//声明函数原型
unsigned long KeyboardHandler(void *pvCBData,
                                     unsigned long ulEvent,
                                     unsigned long ulMsgData,
                                     void *pvMsgData);
unsigned long EventHandler(void *pvCBData, unsigned long ulEvent,
                                  unsigned long ulMsgData, void *pvMsgData);

const tUSBDMSCDevice g_sMSCDevice;
//msc 状态
volatile enum
{
    MSC_DEV_DISCONNECTED,
    MSC_DEV_CONNECTED,
    MSC_DEV_IDLE,
    MSC_DEV_READ,
    MSC_DEV_WRITE,
}
g_eMSCState;
//全局标志
#define FLAG_UPDATE_STATUS      1
static unsigned long g_ulFlags;
//DMA
tDMAControlTable sDMAControlTable[64] __attribute__ ((aligned(1024)));

//*****************************************************************************
// 语言描述符
//*****************************************************************************
const unsigned char g_pLangDescriptor[] =
{
    4,
    USB_DTYPE_STRING,
    USBShort(USB_LANG_EN_US)
};
//*****************************************************************************
// 制造商 字符串 描述符
//*****************************************************************************
const unsigned char g_pManufacturerString[] =
{
    (17 + 1) * 2,
    USB_DTYPE_STRING,
    ’T’, 0, ’e’, 0, ’x’, 0, ’a’, 0, ’s’, 0, ’ ’, 0, ’I’, 0, ’n’, 0, ’s’, 0,
    ’t’, 0, ’r’, 0, ’u’, 0, ’m’, 0, ’e’, 0, ’n’, 0, ’t’, 0, ’s’, 0,
};
//*****************************************************************************
//产品 字符串 描述符
//*****************************************************************************
const unsigned char g_pProductString[] =
{
    (19 + 1) * 2,
    USB_DTYPE_STRING,
    ’M’, 0, ’a’, 0, ’s’, 0, ’s’, 0, ’ ’, 0, ’S’, 0, ’t’, 0, ’o’, 0, ’r’, 0,
    ’a’, 0, ’g’, 0, ’e’, 0, ’ ’, 0, ’D’, 0, ’e’, 0, ’v’, 0, ’i’, 0, ’c’, 0,
    ’e’, 0
};
//*****************************************************************************
//   产品 序列号 描述符
//*****************************************************************************
const unsigned char g_pSerialNumberString[] =
{
    (8 + 1) * 2,
    USB_DTYPE_STRING,
    '1’, 0, '2’, 0, '3’, 0, '4’, 0, '5’, 0, '6’, 0, '7’, 0, '8’, 0
};
//*****************************************************************************
// 字符串描述符集合
//*****************************************************************************
const unsigned char * const g_pStringDescriptors[] =
{
    g_pLangDescriptor,
    g_pManufacturerString,
    g_pProductString,
    g_pSerialNumberString,
};
#define NUM_STRING_DESCRIPTORS (sizeof(g_pStringDescriptors) /                \
                                sizeof(unsigned char *))
//*****************************************************************************
//MSC 实例，配置并为设备信息提供空间
//*****************************************************************************
tMSCInstance g_sMSCInstance;
//*****************************************************************************
//msc 设备配置
//*****************************************************************************
const tUSBDMSCDevice g_sMSCDevice =
{
    USB_VID_STELLARIS,
    USB_PID_MSC,
    "TI      ",
    "Mass Storage    ",
    "1.00",
    200,
    USB_CONF_ATTR_SELF_PWR,
    g_pStringDescriptors,
    NUM_STRING_DESCRIPTORS,
    {
        USBDMSCStorageOpen,
        USBDMSCStorageClose,
        USBDMSCStorageRead,
        USBDMSCStorageWrite,
        USBDMSCStorageNumBlocks
    },
    USBDMSCEventCallback,
    &g_sMSCInstance
};
#define MSC_BUFFER_SIZE 512
```

```
//*****************************************************************************
//键盘实例，配置并为设备信息提供空间
//*****************************************************************************
tHIDKeyboardInstance g_KeyboardInstance;
//*****************************************************************************
//键盘设备配置
//*****************************************************************************
const tUSBDHIDKeyboardDevice g_sKeyboardDevice =
{
    USB_VID_STELLARIS,
     USB_VID_STELLARIS,
    200,
    USB_CONF_ATTR_SELF_PWR | USB_CONF_ATTR_RWAKE,
    KeyboardHandler,
    (void *)&g_sKeyboardDevice,
    0,
    0,
    &g_KeyboardInstance
};
//*****************************************************************************
//复合设备配置
//*****************************************************************************
tCompositeEntry g_psCompDevices[]=
{
    {
        &g_sMSCDeviceInfo,
       (void  *)&g_sMSCDeviceInfo
    },
    {
        &g_sHIDDeviceInfo,
        (void  *)&g_sHIDDeviceInfo
    }
};
#define NUM_DEVICES          (sizeof(g_psCompDevices)/sizeof(tCompositeEntry))
tCompositeInstance g_CompInstance;
unsigned long xxx[10];
tUSBDCompositeDevice g_sCompDevice =
{
    USB_VID_STELLARIS,
    0x0124,
    500,
    USB_CONF_ATTR_BUS_PWR,
    EventHandler,
    g_pStringDescriptors,
    NUM_STRING_DESCRIPTORS,
    2,
    g_psCompDevices,
     xxx,
    &g_CompInstance
};

//****************************************************************************
//callback 函数
//****************************************************************************
unsigned long USBDMSCEventCallback(void *pvCBData, unsigned long ulEvent,
                     unsigned long ulMsgParam, void *pvMsgData)
{
    switch(ulEvent)
    {
        // 正在写数据到存储设备.
        case USBD_MSC_EVENT_WRITING:
        {

            break;
        }
        //读取数据.
        case USBD_MSC_EVENT_READING:
        {
             GPIOPinWrite(GPIO_PORTF_BASE,0x10,0x10);
            break;
        }
         //空闲
        case USBD_MSC_EVENT_IDLE:
        default:
        {
              GPIOPinWrite(GPIO_PORTF_BASE,0x10,0x00);
            break;
        }
    }
    return(0);
}
//****************************************************************************
//键盘 callback 函数
//****************************************************************************
unsigned long KeyboardHandler(void *pvCBData, unsigned long ulEvent,
                              unsigned long ulMsgData, void *pvMsgData)
{
    switch (ulEvent)
    {
        case USB_EVENT_CONNECTED:
        {
              GPIOPinWrite(GPIO_PORTF_BASE,0x20,0x20);
            break;
        }
        case USB_EVENT_DISCONNECTED:
        {
              GPIOPinWrite(GPIO_PORTF_BASE,0x20,0x00);
            break;
        }
        case USB_EVENT_TX_COMPLETE:
        {

            break;
        }
        case USB_EVENT_SUSPEND:
        {

            break;
        }
        case USB_EVENT_RESUME:
        {
            break;
        }
        case USBD_HID_KEYB_EVENT_SET_LEDS:
        {
            break;
        }
        default:
        {
            break;
        }
    }
    return (0);
}
//******************************************************************************
//复合设备 callback 函数
//******************************************************************************
unsigned long EventHandler(void *pvCBData, unsigned long ulEvent, unsigned long ulMsgData,
             void *pvMsgData)
{
    unsigned long ulNewEvent;
    ulNewEvent = 1;
```

```
        switch(ulEvent)
        {
            case USB_EVENT_CONNECTED:
            {
                break;
            }
            case USB_EVENT_DISCONNECTED:
            {
                break;
            }
            case USB_EVENT_SUSPEND:
            {
                break;
            }
            case USB_EVENT_RESUME:
            {
                break;
            }

            default:
            {
                ulNewEvent = 0;
                break;
            }
        }
        if(ulNewEvent)
        {
        }
        return(0);
    }
    //********************************************************************************
    //主函数
    //********************************************************************************
    int main(void)
    {
         //系统初始化。
         SysCtlLDOSet(SYSCTL_LDO_2_75V);
         SysCtlClockSet(SYSCTL_XTAL_8MHZ    |    SYSCTL_SYSDIV_8    |    SYSCTL_USE_PLL       |
SYSCTL_OSC_MAIN );
         SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
         GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE,0xf0);
         GPIOPinTypeGPIOInput(GPIO_PORTF_BASE,0x0f);
         HWREG(GPIO_PORTF_BASE+GPIO_O_PUR) |= 0x0f;
         // ucDMA 配置
    SysCtlPeripheralEnable(SYSCTL_PERIPH_UDMA);
    SysCtlDelay(10);
    uDMAControlBaseSet(&sDMAControlTable[0]);
    uDMAEnable();

     g_ulFlags = 0;
     g_eMSCState = MSC_DEV_IDLE;
     //复合设备初始化
     g_sCompDevice.psDevices[0].pvInstance =
       USBDMSCCompositeInit(0, &g_sMSCDevice);
    g_sCompDevice.psDevices[1].pvInstance =
        USBDHIDKeyboardInit(0, &g_sKeyboardDevice);
     USBDCompositeInit(0, &g_sCompDevice, DESCRIPTOR_DATA_SIZE,
                      g_pucDescriptorData);
     //初始化存储设备
     disk_initialize(0);
     while(1)
     {
        USBDHIDKeyboardKeyStateChange((void *)&g_sKeyboardDevice, HID_KEYB_CAPS_LOCK,
                                     HID_KEYB_USAGE_A,
                                     (GPIOPinRead(GPIO_PORTF_BASE, 0x0f) & GPIO_PIN_0)
                                     ? false : true);
        USBDHIDKeyboardKeyStateChange((void *)&g_sKeyboardDevice, 0,
                                     HID_KEYB_USAGE_DOWN_ARROW,
                                     (GPIOPinRead(GPIO_PORTF_BASE, 0x0f) & GPIO_PIN_1)
                                     ? false : true);
        USBDHIDKeyboardKeyStateChange((void *)&g_sKeyboardDevice, 0,
                                     HID_KEYB_USAGE_UP_ARROW,
                                     (GPIOPinRead(GPIO_PORTF_BASE, 0x0f) & GPIO_PIN_2)
                                     ? false : true);
        USBDHIDKeyboardKeyStateChange((void *)&g_sKeyboardDevice, 0,
                                     HID_KEYB_USAGE_ESCAPE,
                                     (GPIOPinRead(GPIO_PORTF_BASE, 0x0f) & GPIO_PIN_3)
                                     ? false : true);
        SysCtlDelay(SysCtlClockGet()/3000);
    }
}
```