UEFI Development Exploration 88- YIE002USB Development Board (11 Accessing HID Devices under UEFI)

原创 luobing4365 Posted on 2021-05-13 10:38:40 Read 1.1k Collection 1 Likes 1 copyright

Category Column: UEFI Development Article Tags: uefi Low-level application development usb hid USB

UEFI Development This column includes this content 503 Subscribe 104 articles Subscribe to our column

(Please keep it-> Author: Luo Bing https://blog.csdn.net/luobing4365)

YIE002USB development board access HID device under UEFI

Through the blogs in the previous chapters, the USB HID device was made, and the working status of the device was tested using the host computer tool UsbHID under Windows .

Finally, we can build a project to access USB HID devices under UEFI system.

The USB HID device we made can communicate successfully under the Windows system, which also means that we can also communicate with it in the UEFI environment.

First, use the lsusb tool under Linux to view the USB HID device we implemented earlier:

```
robin@robin-virtual-machine:~$ sudo lsusb -v -d 0x8765:4321
Bus 002 Device 004: ID 8765:4321
Device Descriptor:
  bLength                18
  bDescriptorType         1
  bcdUSB               2.00
  bDeviceClass            0 (Defined at Interface level)
  bDeviceSubClass         0
  bDeviceProtocol         0
  bMaxPacketSize0        64
  idVendor           0x8765
  idProduct          0x4321

```

```
bcdDevice            2.00
iManufacturer           1 Robin
iProduct                2 Robin's UEFI Explorer
iSerial                 3 My123
bNumConfigurations      1
Configuration Descriptor:
  bLength                 9
  bDescriptorType         2
  wTotalLength           41
  bNumInterfaces          1
  bConfigurationValue     1
  iConfiguration          0
  bmAttributes         0xe0
    Self Powered
    Remote Wakeup
  MaxPower              100mA
  Interface Descriptor:
    bLength                 9
    bDescriptorType         4
    bInterfaceNumber        0
    bAlternateSetting       0
    bNumEndpoints           2
    bInterfaceClass         3 Human Interface Device
    bInterfaceSubClass      0 No Subclass
    bInterfaceProtocol      0 None
    iInterface              0
      HID Device Descriptor:
        bLength                 9
        bDescriptorType        33
        bcdHID               1.10
        bCountryCode            0 Not supported
        bNumDescriptors         1
        bDescriptorType        34 Report
        wDescriptorLength      33
       Report Descriptors:
         ** UNAVAILABLE **
    Endpoint Descriptor:
      bLength                 7
      bDescriptorType         5
      bEndpointAddress     0x81  EP 1 IN
      bmAttributes            3
        Transfer Type            Interrupt
        Synch Type               None
        Usage Type               Data
      wMaxPacketSize     0x0040  1x 64 bytes
      bInterval              32
    Endpoint Descriptor:
      bLength                 7
      bDescriptorType         5
      bEndpointAddress     0x01  EP 1 OUT

```

```
        bmAttributes            3
          Transfer Type            Interrupt
          Synch Type               None
          Usage Type               Data
        wMaxPacketSize     0x0040  1x 64 bytes
        bInterval              32
Device Status:     0x0003
  Self Powered
  Remote Wakeup Enabled
```

It can be seen that the homemade USB HID device can communicate using interrupt transmission at endpoint 1. In addition, the report descriptor is not listed in the above information. This device supports communication through Output report & Input report, Feature report, and the transmission byte is 16 bytes.

This article introduces how to implement the codes corresponding to these three communication methods under the UEFI system. The main implementation steps are as follows.

1 Add library and header files for accessing USB HID devices

In EDK2's MdePkg, a library called UefiUsbLib is provided to support USB HID device access. Its library functions are defined in the header file \MdePkg\Include\ Library \UefiUsbLib.h.

In UefiUsbLib, the corresponding functions of HID standard commands and class commands are provided. For example, the function corresponding to the standard command Get_Descriptor is UsbGetDescriptor(), and the function corresponding to the class command Get_Report is UsbGetReportRequest(). For the rest of the USB commands, you can know the corresponding functions from the function names.

When accessing USB HID devices, we can directly use these functions for communication. Therefore, it is necessary to add library declarations and header file declarations to the sample project. Add the following declarations to the INF file of the sample project:

```
[Packages]
  MdePkg/MdePkg.dec
  ......            //其他Package
[LibraryClasses]
  UefiUsbLib      //添加支持USB HID设备访问的函数库
  ......           //其他库
```

And in the header file Common.h, add the include header file declaration:

```
#include <Library/UefiUsbLib.h>
```

After completing the above work, you can call the function to access the USB HID device in the code.

2 Locating USB HID devices

Similar to the host computer test tool UsbHID, we locate the device through the manufacturer ID and product ID of the HID device. When
the manufacturer ID is 0x8765 and the product ID is 0x4321, it means that the device found is the HID device we made. The implementation code is shown in Example 1.

Example 1: Locating your own HID device

```
BOOLEAN findMyHidDevice(OUT INT16 *index,IN UINT16 MyVID,IN UINT16 MyPID)
{
  EFI_STATUS Status;
  INT16 i;
  EFI_USB_DEVICE_DESCRIPTOR     UsbDevDesc;
  if(gUsbIOCount == 0)  //没有USB设备
    return FALSE;
  for(i=0;i<gUsbIOCount;i++) //轮询是否为指定的设备
  {
    Status = gUsbIO[i]->UsbGetDeviceDescriptor(gUsbIO[i], &UsbDevDesc);
    if(Status == EFI_SUCCESS)
    {
      if((UsbDevDesc.IdVendor == MyVID) && (UsbDevDesc.IdProduct == MyPID))
      {
        *index = i;
        return TRUE;
      }
    }
  }
  return FALSE;
}
```

The function findMyHidDevice() in Example 1 finds the USB device with product ID MyVID and product ID MyPID from the global array gUsbIO[]. The array gUsbIO[] is an array of pointers of type EFI_USB_IO_PROTOCOL, and each element is equivalent to an interface of a USB device. The USB HID device we made has only one interface, so it only occupies one element in the array.

After finding the corresponding device, the function will return its corresponding array index (parameter INT16 *index). Thus, we get the EFI_USB_IO_PROTOCOL Protocol instance corresponding to the USB HID device, and we can call its interface function to communicate with the HID device.

3 Communicating with USB HID devices

After getting the Protocol instance of the USB device, you can use the functions corresponding to the class commands Set_Report and Get_Report to
send data to and receive data from the HID device. The implementation code is shown in Example 2.

Example 2: Communicating with HID device (Output report & Input report)

```
VOID Output_Input_Report (IN INT16 index)
{
  EFI_STATUS Status;
  UINT8   ReportId, myBuffer[16];
  INTN i;
  gBS->SetMem(myBuffer,16,0xA0);
  ReportId = 0;
  Status = UsbSetReportRequest(
    gUsbIO[index],  //Protocol实例
    0,                //接口
    ReportId,       //报告ID
    HID_OUTPUT_REPORT, //报告类型
    16,                   //缓冲区长度
    myBuffer             //缓冲区
  );
  if(EFI_ERROR(Status)) return;
  gBS->SetMem(myBuffer,16,0x00);
  Status = UsbGetReportRequest(
    gUsbIO[index], //Protocol实例
    0,               //接口
    ReportId,       //报告ID
    HID_INPUT_REPORT, //报告类型
    16,                  //缓冲区长度
    myBuffer            //缓冲区
  );
  if(EFI_ERROR(Status)) return;
  Print(L"Get data from MyHidDevice:\n");
  for(i=0;i<16;i++)
    Print(L"0x%02x ",myBuffer[i]);
  Print(L"\n");
}
```

In the function Output_Input_Report() in Example 2, we called UsbSetReportRequest() and UsbGetReportRequest() to communicate with the HID device through the Output report and Input report. It should be noted that when calling these two functions, the report ID (also known as ReportID) is set to 0. This is because when we designed the HID device, there was no item in the report descriptor to set the report ID, so we only needed to set it to 0.

The library functions UsbSetReportRequest() and UsbGetReportRequest() are implemented in the EDK2 source file \MdePkg\Library\UefiUsbLib\Hid.c. These two functions call the interface function UsbControlTransfer() of EFI_USB_IO_PROTOCOL to communicate with the HID device.

At this point, we have completed the core code for communicating with the HID device. In the main function, directly call findMyHidDevice() and Output_Input_Report() to access the HID device.

The use of endpoint 1 (interrupt transfer) and feature report is also implemented in the sample code. The implementation code of feature report is no different from the above Output_Input_Report(), except that the report type is modified.

Communication using endpoint 1, which is similar to Windows' ReadFile() & Write() communication method, is implemented using the UEFI USB Protocol interface function UsbSyncInterruptTransfer(). For the usage of this interface function, see the UEFI specification USB section.

The code for endpoint 1 communication is as follows:

Example 3 Communicating with HID device (endpoint 1 interrupt transfer)

```
VOID Endpoint_OutIn(IN INT16 index)
{
  EFI_STATUS Status;
  // UINT8   ReportId;
  UINT8 myBuffer[16];
  UINTN lenBuffer;
  UINTN i;
  UINT32 result;

  gBS->SetMem(myBuffer,16,0xA0);
  lenBuffer=16;
  Status = gUsbIO[index]->UsbSyncInterruptTransfer(
    gUsbIO[index],
    0x01, //EP1 OUT
    myBuffer,
    &lenBuffer,
    32,
    &result
  );
  if(EFI_ERROR(Status))
  {
    Print(L"OUT:UsbSyncInterruptTransfer Error!\n");
    Print(L"Status:%r\n",Status);
    return;
  }
  if(EFI_ERROR(result))
  {
    Print(L"UsbSyncInterruptTransfer result:%r\n",result);
    Print(L"\n");
    return;
  }

  gBS->SetMem(myBuffer,16,0x00);
  Status = gUsbIO[index]->UsbSyncInterruptTransfer(
    gUsbIO[index],
    0x81, //EP1 IN
    myBuffer,
    &lenBuffer,
    32,
    &result
  );
  if(EFI_ERROR(Status))
```

```
    {
      Print(L"IN:UsbSyncInterruptTransfer Error!\n");
      Print(L"Status:%r\n",Status);
      return;
    }
    if(EFI_ERROR(result))
    {
      Print(L"UsbSyncInterruptTransfer result:%r\n",result);
      Print(L"\n");
      return;
    }
    else
    {
      Print(L"Endpoint_OutIn,data from MyHidDevice=%d:\n",lenBuffer);
      for(i=0;i<lenBuffer;i++)
        Print(L"0x%02x ",myBuffer[i]);
      Print(L"\n");
    }
```

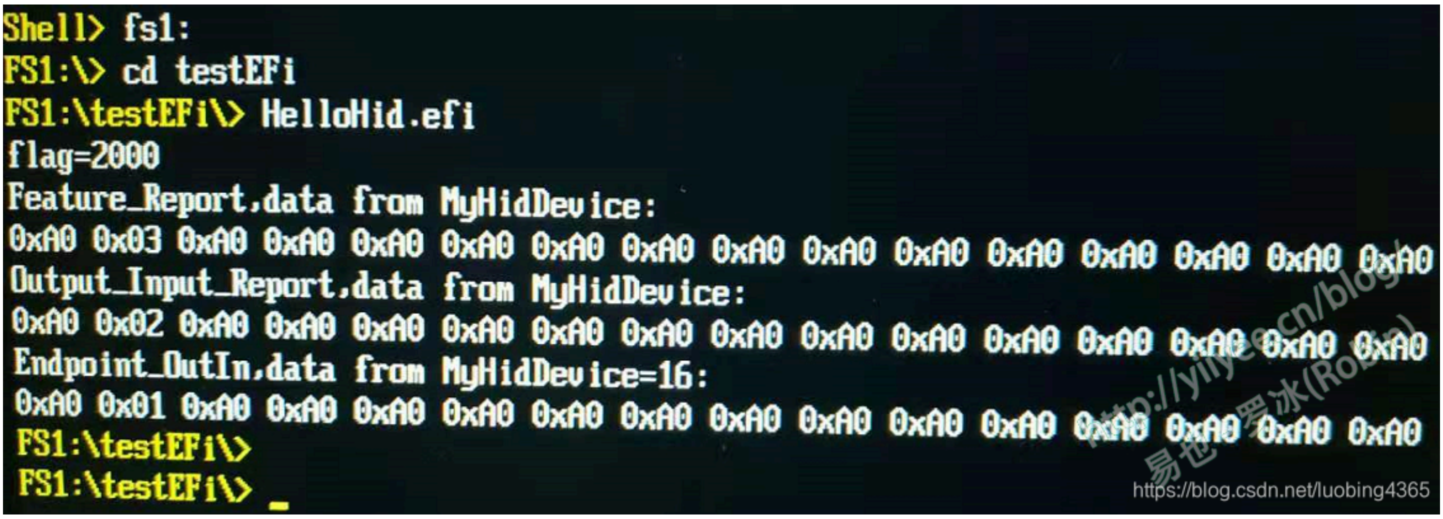
It should be noted that this chapter writes code for endpoint 1 because endpoint 1 is set as the communication endpoint in the homemade HID device. If the USB HID device uses other endpoints for communication, the code must be modified accordingly.

Compile the examples provided in this article:

```
C:\UEFIWorkspace\edk2\build -p RobinPkg\RobinPkg.dsc \
-m RobinPkg\Applications\ListUSB\HelloHid.inf -a X64
```

The HelloHid program can only be run on an actual machine. You can insert a homemade HID device into the computer, enter the UEFI Shell environment, run HelloHid, and observe the results of the communication with the HID device.

It should be noted that some UEFI BIOS do not support USB Protocol well. I conducted the experiment on Intel NUC (NUC6CAYHC) and the result was quite good, as shown in Figure 1.



```
Shell> fs1:
FS1:\> cd testEFi
FS1:\testEFi\> HelloHid.efi
flag=2000
Feature_Report,data from MyHidDevice:
0xA0 0x03 0xA0 0xA0 0xA0 0xA0 0xA0 0xA0 0xA0 0xA0 0xA0 0xA0 0xA0 0xA0 0xA0 0xA0
Output_Input_Report,data from MyHidDevice:
0xA0 0x02 0xA0 0xA0 0xA0 0xA0 0xA0 0xA0 0xA0 0xA0 0xA0 0xA0 0xA0 0xA0 0xA0 0xA0
Endpoint_OutIn,data from MyHidDevice=16:
0xA0 0x01 0xA0 0xA0 0xA0 0xA0 0xA0 0xA0 0xA0 0xA0 0xA0 0xA0 0xA0 0xA0 0xA0 0xA0
FS1:\testEFi\>
FS1:\testEFi\> _
```

Figure 1 Testing HelloHid

Gitee address: https://gitee.com/luobing4365/uefi-explorer

Project code is located in: /FF RobinPkg/RobinPkg/Applications/HelloHid