UEFI Development Exploration 87- YIE002USB Development Board (10 UEFI Support for USB 2)

原创 luobing4365 Posted on 2021-05-04 16:13:06 Read 893 Collection 2 Likes copyright
Category Column: UEFI Development Article Tags: uefi USB Low-level application development bios UEFI Programming Practice

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 UEFI support for USB2

3 EFI_USB_IO_PROTOCOL

EFI_USB_IO_PROTOCOL is generated by the USB bus driver and can be used by UEFI applications and drivers to access various USB devices, such as USB keyboards, mice, and mass storage devices. The interface provided by EFI_USB_IO_PROTOCOL can provide four types of transmission methods to communicate with USB devices, namely the control transmission, interrupt transmission, bulk transmission, and real-time transmission introduced earlier.

The function interface of EFI_USB_IO_PROTOCOL is as follows:

```
typedef struct _EFI_USB_IO_PROTOCOL {
  EFI_USB_IO_CONTROL_TRANSFER UsbControlTransfer;    //控制传输
  EFI_USB_IO_BULK_TRANSFER UsbBulkTransfer;           //批量传输
  EFI_USB_IO_ASYNC_INTERRUPT_TRANSFER  \
                        UsbAsyncInterruptTransfer;    //异步中断传输
  EFI_USB_IO_SYNC_INTERRPUT_TRANSFER UsbSyncInterruptTransfer //同步中断传输
  EFI_USB_IO_ISOCHRONOUS_TRANSFER UsbIsochronousTransfer;       //实时传输
  EFI_USB_IO_ASYNC_ISOCHRONOUS_TRANSFER \
                        UsbAsyncIsochronousTransfer; //异步实时传输
  EFI_USB_IO_GET_DEVICE_DESCRIPTOR UsbGetDeviceDescriptor; //获取设备描述符
  EFI_USB_IO_GET_CONFIG_DESCRIPTOR UsbGetConfigDescriptor; //获取配置描述符
  EFI_USB_IO_GET_INTERFACE_DESCRIPTOR \

                        UsbGetInterfaceDescriptor;               //获取接口描述符
EFI_USB_IO_GET_ENDPOINT_DESCRIPTOR UsbGetEndpointDescriptor;
//获取端点描述符
  EFI_USB_IO_GET_STRING_DESCRIPTOR UsbGetStringDescriptor;//获取字符串描述符
  EFI_USB_IO_GET_SUPPORTED_LANGUAGES UsbGetSupportedLanguages;//获取支持语言
  EFI_USB_IO_PORT_RESET UsbPortReset;                         //重启USB控制器
} EFI_USB_IO_PROTOCOL;
```

With the knowledge of USB protocol introduced before, it is easy to understand the various interface functions provided by EFI_USB_IO_PROTOCOL. The following mainly introduces two commonly used interface functions, UsbGetDeviceDescriptor() and UsbControlTransfer(). For the description of other interface functions, please refer to UEFI Spec.

The interface function UsbGetDeviceDescriptor() is used to obtain the device descriptor of the USB device. Its prototype is shown below.

```
typedef EFI_STATUS (EFIAPI *EFI_USB_IO_GET_DEVICE_DESCRIPTOR) (
  IN EFI_USB_IO_PROTOCOL *This,  //Protocol实例
  OUT EFI_USB_DEVICE_DESCRIPTOR *DeviceDescriptor //设备描述符指针变量
);
typedef struct {
  UINT8 Length;              //描述符长度(0x12)
  UINT8 DescriptorType;    //描述符类型(0x01)
  UINT16 BcdUSB;             //USB设备支持的协议版本号
  UINT8 DeviceClass;        //类代码
  UINT8 DeviceSubClass;    //子类代码
  UINT8 DeviceProtocol;    //协议码
  UINT8 MaxPacketSize0;    //端点0最大包长度
  UINT16 IdVendor;          //厂商ID
  UINT16 IdProduct;         //产品ID
  UINT16 BcdDevice;         //设备发行号
  UINT8 StrManufacturer;   //厂商信息
  UINT8 StrProduct;         //产品信息
  UINT8 StrSerialNumber;   //设备序列号
  UINT8 NumConfigurations;//配置描述符数目
} EFI_USB_DEVICE_DESCRIPTOR;
```

The device descriptor obtained by UsbGetDeviceDescriptor() has member variables that are exactly the same as the descriptor introduced in Chapter 82. It is easy to understand by comparing them.

The interface function UsbControlTransfer() is used to perform control transfer, including USB standard commands and class commands, which are all performed through control transfer. Its function interface prototype is:

```
typedef EFI_STATUS (EFIAPI *EFI_USB_IO_CONTROL_TRANSFER) (
  IN EFI_USB_IO_PROTOCOL *This,        //Protocol实例
  IN EFI_USB_DEVICE_REQUEST *Request, //USB命令（标准命令、类命令、厂商命令）
  IN EFI_USB_DATA_DIRECTION Direction,//方向
  IN UINT32 Timeout,                      //超时时间，单位为毫秒
  IN OUT VOID *Data OPTIONAL,           //发往或接收自USB设备的数据缓冲区
  IN UINTN DataLength OPTIONAL,         //数据缓冲区的长度
  OUT UINT32 *Status                      //USB传输的结果
);
typedef enum {
  EfiUsbDataIn,     //接收自USB设备，即从USB设备往USB主机发送
  EfiUsbDataOut,    //发往USB设备，即从USB主机发往USB设备
  EfiUsbNoData
} EFI_USB_DATA_DIRECTION;
typedef struct {
  UINT8 RequestType;   //命令类型
  UINT8 Request;        //命令序号
  UINT16 Value;         //不同的命令，含义不同
  UINT16 Index;         //不同的命令，含义不同
  UINT16 Length;        //如果有数据阶段，此字段为数据的字节数
} EFI_USB_DEVICE_REQUEST;   //USB命令结构体
```

UsbControlTransfer() builds a transmission channel between the USB device driver and the USB device. As can be seen from the function prototype, its main function is to execute various USB commands, including USB standard commands, class commands, and manufacturer commands. The data structure of its parameter Request is exactly the same as the structure of the USB command given in the 84th article of the UEFI exploration series blog, and the meaning is exactly the same.

The parameter Status gives the result of USB transmission, which is different from the function execution return value. The function return value is used to reflect the function execution status, such as whether the parameter is wrong, whether the execution timeout, etc.

```
#define EFI_USB_NOERROR         0x0000
#define EFI_USB_ERR_NOTEXECUTE 0x0001
#define EFI_USB_ERR_STALL       0x0002
#define EFI_USB_ERR_BUFFER      0x0004
#define EFI_USB_ERR_BABBLE      0x0008
#define EFI_USB_ERR_NAK          0x0010
#define EFI_USB_ERR_CRC          0x0020
#define EFI_USB_ERR_TIMEOUT     0x0040
#define EFI_USB_ERR_BITSTUFF    0x0080
#define EFI_USB_ERR_SYSTEM      0x0100
```

4 Using USB Protocol

The usage of EFI_USB2_HC_PROTOCOL and EFI_USB_IO_PROTOCOL is actually similar to the PCI Protocol and SMBUS Protocol introduced in the previous blog. The basic usage process includes:

1. Add header files to support USB access, including UsbHostController.h and UsbIo.h;
2. Add the GUIDs of the supported USB access protocols to the [Protocols] Section of the INF file, including gEfiUsbIoProtocolGuid and gEfiUsb2HcProtocolGuid;

3. Write a function to get a Protocol instance.

The following is an example. The arrays gUsb2HC[] and gUsbIO[] include instances of EFI_USB2_HC_PROTOCOL and EFI_USB_IO_PROTOCOL, which can be used directly to obtain USB host controller information and USB device information.

Example 1: Get USB host controller information

```
VOID lsUsb2HC(void)
{
  EFI_STATUS Status;
  UINTN i;
  UINT8 maxSpeed,portNumber,is64BC;
  EFI_USB_HC_STATE state;
  CHAR16 *speed[]={L"FULL ",L"LOW  ",L"HIGH ",L"SUPER"};
  CHAR16 *hcState[]={L"Halt",L"Operational",L"Suspend"};
  if(gUsb2HCCount == 0)   //没有Protocol实例，直接退出
    return;
  Print(L"Usb HC: %d \n",gUsb2HCCount);
  Print(L"No. MaxSpeed PortNumber Is64BitCapable State\n");
  for(i=0;i<gUsb2HCCount;i++)
  {
    Print(L"%03d ",i);
    Status = gUsb2HC[i]->GetCapability(gUsb2HC[i],&maxSpeed, \
&portNumber,&is64BC); //获取属性
    if(EFI_ERROR(Status))
      Print(L"???");
    else
      Print(L"%*S %*d %*d ",8,speed[maxSpeed],10,portNumber,14,is64BC);
    Status = gUsb2HC[i]->GetState(gUsb2HC[i],&state);   //获取状态
    if(EFI_ERROR(Status))
      Print(L" ???\n");
    else
      Print(L"%S \n",hcState[state]);
  }
}
```

The logic of the function is relatively simple. It mainly calls the EFI_USB2_HC_PROTOCOL interface functions GetCapability() and GetState() to obtain the properties and status of the USB host controller.

The method of obtaining USB device information is similar. Use the interface function UsbGetDeviceDescriptor() of EFI_USB_IO_PROTOCOL to obtain the device descriptor of the USB device. Print out the class, subclass, manufacturer ID and product ID in the device descriptor. The implementation code is shown in Example 2.

Example 2: Get USB device information

```
VOID lsUsbIO(void)
{
  EFI_STATUS Status;
  UINTN i;
  EFI_USB_DEVICE_DESCRIPTOR UsbDevDesc;
  if(gUsbIOCount == 0)   //没有Protocol实例，直接退出
    return;
  Print(L"Usb device: %d \n",gUsbIOCount);
  Print(L"No. DevClass SubClass IdVendor IdProduct \n");
  for(i=0;i<gUsbIOCount;i++)
  {
    Print(L"%03d ",i);
    Status = gUsbIO[i]->UsbGetDeviceDescriptor(gUsbIO[i], &UsbDevDesc);
    if(EFI_ERROR(Status))
      Print(L"Get Device Descriptor Error!\n");
    else
    {
      Print(L"     %03d      %03d ", \
UsbDevDesc.DeviceClass,UsbDevDesc.DeviceSubClass);
      Print(L"  0x%04x    0x%04x\n", \
UsbDevDesc.IdVendor,UsbDevDesc.IdProduct);
    }
  }
}
```

At this point, the introduction to USB support in UEFI system is complete.

In the next article, we will use this knowledge to build a routine to access USB HID devices.