UEFI Development Exploration 86- YIE002USB Development Board (09 UEFI Support for USB 1)

原创 luobing4365 Posted on 2021-05-04 16:08:28 Views: 962 Collection 4 Likes

Category Column: UEFI Development Article Tags: uefi Low-level application development USB bios UEFI Programming Practice

YIE002USB development board UEFI support for USB1

Through the previous blogs, we have made a USB HID device with communication capabilities. Using the test tools under Windows written previously , the device works well.

Similarly, in the UEFI environment, you can also write a host computer program to access USB HID devices. This article mainly introduces UEFI's support for USB, including UEFI's USB driver architecture and the provided USB Protocol.

1 USB driver protocol architecture under UEFI

In the UEFI system, there are USB host controller drivers, USB bus drivers, and USB device drivers, which work together to build the USB driver protocol stack of the UEFI platform. The model of the USB driver protocol stack is shown in Figure 1, which demonstrates the relationship between USB drivers and the protocols used.

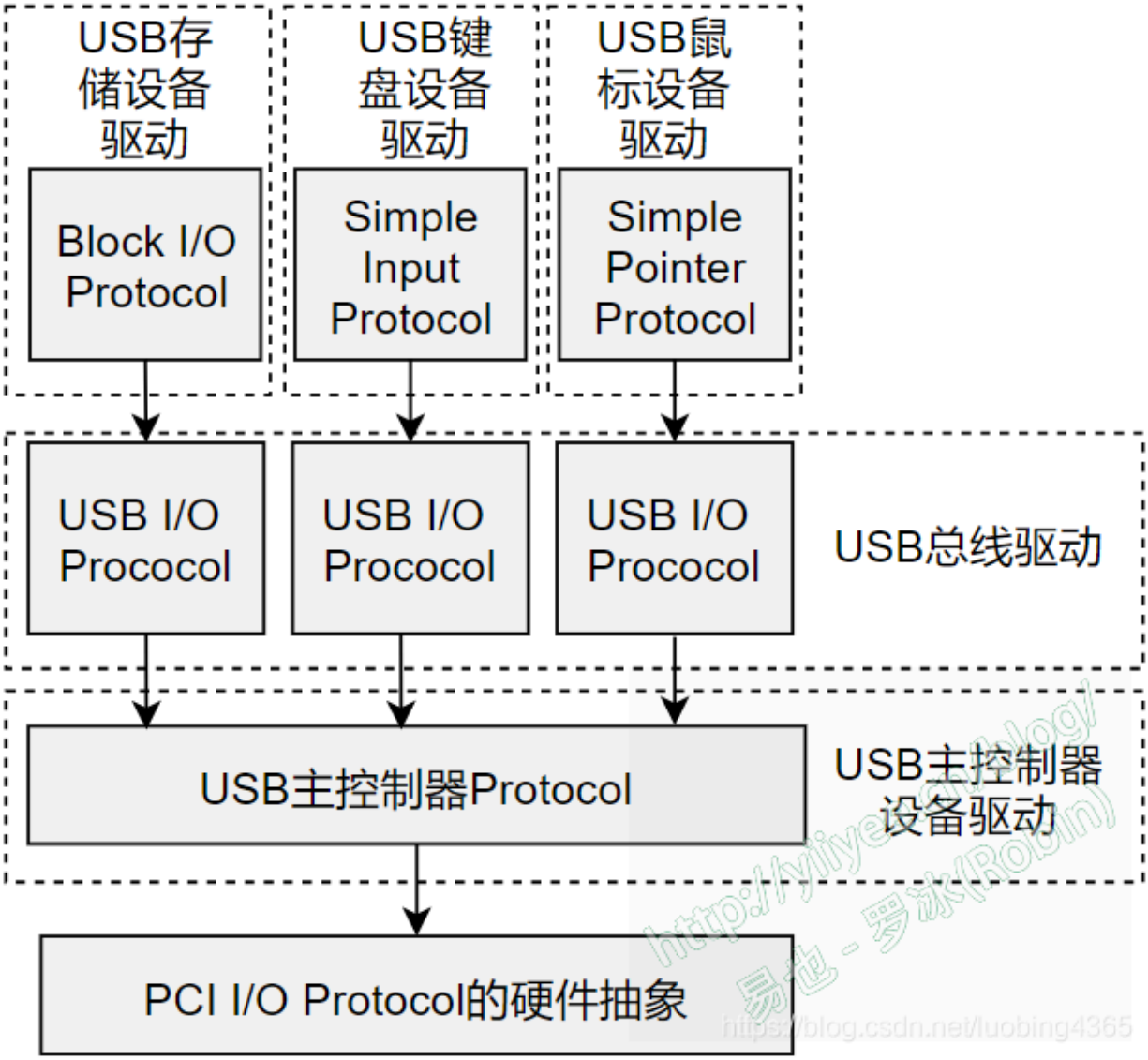

Figure 1 USB protocol stack in UEFI

A separate USB controller is provided on the PCI bus of the platform hardware . The PCI bus driver installs protocols for the USB host controller handle, including EFI_DEVICE_PATH_PROTOCOL and EFI_PCI_IO_PROTOCOL. The USB host controller uses EFI_PCI_IO_PROTOCOL and installs EFI_USB2_HC_PROTOCOL on its handle.

The USB bus driver uses the services provided by EFI_USB2_HC_PROTOCOL to communicate with USB devices on the USB bus. In Figure 1, the USB bus driver detects three devices, a USB keyboard, a USB mouse, and a USB mass storage device. The USB bus driver creates three child handles for these three devices and installs protocols for them, namely EFI_DEVICE_PATH_PROTOCOL and EFI_USB_IO_PROTOCL.

These devices will use the EFI_USB_IO_PROTOCOL of their handles and generate the corresponding Protocol. Take the USB mouse driver as an example, it will use EFI_USB_IO_PROTOCOL and generate EFI_SIMPLE_POINTER_PROTOCOL.
The processing of other devices is similar, and you can see the Protocols they generate from the figure.

2 EFI_USB2_HC_PROTOCOL

The USB host controller driver is a device driver that follows the UEFI driver model. In the MdeModulePkg of EDK2, various USB host controller drivers are prepared. The supported USB host controllers include:

1. Open Host Controller Interface (OHCI) (USB 1.0 and USB 1.1)
2. Universal Host Controller Interface (UHCI) (USB 1.0 and USB 1.1)
3. Enhanced Host Controller Interface (EHCI) (USB 2.0)
4. Extended Host Controller Interface (XHCI) (USB 3.0)

The USB host controller driver uses the services provided by EFI_PCI_IO_PROTOCOL and installs EFI_USB2_HC_PROTOCOL on the host controller handle. As we know, the USB2 host controller is a hardware component connected to the Universal Serial Bus (USB) . It generates transfer events on the USB and transfers data between the system memory and the device. Therefore, this protocol is generally used by the USB bus driver to manage the USB root hub and various USB devices.

The function interface of EFI_USB2_HC_PROTOCOL is shown below.

```
typedef struct _EFI_USB2_HC_PROTOCOL {
  EFI_USB2_HC_PROTOCOL_GET_CAPABILITY GetCapability; //获取USB主控制器的属性
  EFI_USB2_HC_PROTOCOL_RESET Reset;           //软重启USB主控制器
  EFI_USB2_HC_PROTOCOL_GET_STATE GetState;  //获取当前USB主控制的状态
  EFI_USB2_HC_PROTOCOL_SET_STATE SetState;  //设置USB主控制器状态
  EFI_USB2_HC_PROTOCOL_CONTROL_TRANSFER ControlTransfer;
//向目标USB设备发送控制传输
  EFI_USB2_HC_PROTOCOL_BULK_TRANSFER BulkTransfer;//向目标USB设备发送批量传输
  EFI_USB2_HC_PROTOCOL_ASYNC_INTERRUPT_TRANSFER \
                            AsyncInterruptTransfer;     //异步中断传输
  EFI_USB2_HC_PROTOCOL_SYNC_INTERRUPT_TRANSFER  \
                            SyncInterruptTransfer;      //同步中断传输
  EFI_USB2_HC_PROTOCOL_ISOCHRONOUS_TRANSFER      \
                            IsochronousTransfer;         //实时传输
  EFI_USB2_HC_PROTOCOL_ASYNC_ISOCHRONOUS_TRANSFER
                             AsyncIsochronousTransfer; //异步实时传输
  EFI_USB2_HC_PROTOCOL_GET_ROOTHUB_PORT_STATUS
                            GetRootHubPortStatus;       //获得根USB Hub端口状态
  EFI_USB2_HC_PROTOCOL_SET_ROOTHUB_PORT_FEATURE
                            SetRootHubPortFeature;      //设置根USB Hub端口状态
  EFI_USB2_HC_PROTOCOL_CLEAR_ROOTHUB_PORT_FEATURE
                            ClearRootHubPortFeature;   //清除特征
  UINT16 MajorRevision;    //主版本号
  UINT16 MinorRevision;   //次版本号
} EFI_USB2_HC_PROTOCOL;
```

In the EDK2 directory MdeModulePkg/Bus/Usb/UsbBusDxe, you can view how the USB bus driver uses this protocol. We mainly use EFI_USB2_HC_PROTOCOL to enumerate the USB controller. The interface functions to be used include GetCapability() and GetState(). The following describes the usage of these two functions.

The interface function GetCapability() is used to obtain the properties of the main controller. Its function prototype is:

```
typedef EFI_STATUS (EFIAPI *EFI_USB2_HC_PROTOCOL_GET_CAPABILITY) (
  IN EFI_USB2_HC_PROTOCOL *This,  //Protocol实例
  OUT UINT8 *MaxSpeed,              //最大传输速度
  OUT UINT8 *PortNumber,           //根hub端口号
  OUT UINT8 *Is64BitCapable       //是否支持64位内存地址
);
#define EFI_USB_SPEED_FULL  0x0000   //全速，12Mb/s
#define EFI_USB_SPEED_LOW   0x0001   //低速，1.5Mb/s
#define EFI_USB_SPEED_HIGH  0x0002   //高速，480Mb/s
#define EFI_USB_SPEED_SUPER 0x0003  //超高速，4.8GMb/s
```

This function can obtain the properties of the host controller through the EFI_USB2_HC_PROTOCOL instance. Among them, MaxSpeed is the maximum transfer rate of the controller, which can be one of the four types: low speed, full speed, high speed and super speed. After PortNumber is the root Hub port, the USB bus driver needs this parameter when performing bus enumeration. Is64BitCapable is used to show whether the controller supports 64-bit memory access. The host controller software can use this to determine whether to use more than 4G memory for data transmission.

The interface function GetState() is used to obtain the current USB host controller status. Its prototype is as follows:

```
typedef EFI_STATUS (EFIAPI *EFI_USB2_HC_PROTOCOL_GET_STATE) (
  IN EFI_USB2_HC_PROTOCOL *This, //Protocol实例
  OUT EFI_USB_HC_STATE *State //指向EFI_USB_HC_STATE数据类型，USB主控制器状态
);
typedef enum {
  EfiUsbHcStateHalt,            //停止状态
  EfiUsbHcStateOperational,    //运行中的状态
  EfiUsbHcStateSuspend,         //挂起状态
  EfiUsbHcStateMaximum
} EFI_USB_HC_STATE;
```

The USB host controller can be in one of three states: Stop, Run, and Suspend. Only in the Run state can the host controller perform bus communication. When no bus communication occurs for more than 3 seconds, the host controller will enter the Suspend state, which can also be set by software. When the host controller hardware is reset, or a fatal error occurs, such as a consistency check error, the host controller will enter the Stop state. Of course, the Stop state can also be set by software.

EFI_USB2_HC_PROTOCOL can be used to obtain the USB devices in the system and their related information. In UEFI Shell, the "pci" command is provided, but no information about USB devices is provided. You can follow its idea and use EFI_USB2_HC_PROTOCOL to write and obtain information about USB devices.

To develop USB device access, you need to use another Protocol-EFI_USB_IO_PROTOCOL. Due to space limitations, this part will be described in the next blog.
