Advances in Intelligent Systems Research, volume 147
International Conference on Network, Communication, Computer Engineering (NCCE 2018)

HID Architecture and Design in Embedded USB Host System

Xin Li[1, a)], Junchen You[2, b)], Xinyi Zhou[1, c)], Rentai Chen[1, d)] and Ci Tang[1]

[1]*Network and Information Management Center, Chengdu Normal University, Chengdu 611130, China.*
[2]*School of Computer Science, Chengdu Normal University, Chengdu 611130, China.*

[a)] lixin_uestc@163.com
[b)] jcyou@163.com
[c)]408615355@qq.com
[d)]6793838@qq.com

Abstract. Universal Serial Bus (USB) is a personal computer interconnection protocol, developed to make the connection of peripheral devices to a computer easier and more efficient. USB has become the dominant standard for plug-and-play connections between personal computers and peripherals. USB connectivity has moved rapidly into the embedded space. Usually, PCs take the role of USB host in USB system. However, any embedded system has the potential to be an embedded USB host. People can connect human interface devices (HID) or some other USB devices to their embedded products. In this paper we present HID architecture and design in an embedded USB host system.

Keywords: HID, USB Host, Embedded.

INTRODUCTION

There are two roles in USB systems, USB host and USB device. USB works as a Master/Slave bus, where the USB Host is the Master and the devices are the Slaves. All data transfers are initiated by the host and there is only one host in any USB system.

Today, USB has been largely used to connect various peripherals to PCs. And PC must be the host in USB system.

However, an embedded system has the potential to be a USB host. We have implemented the USB Host and HID in Motorola LTD radio. The HID class is a standard device classification for the USB and it consists primarily of devices that are used by humans to control the operation of computer systems. In this paper we will focus on the host-side HID software architecture and design in embedded systems.

USB OVERVIEW

USB is a cable bus that supports data exchange between a host computer and a wide range of simultaneously accessible peripherals. The attached peripherals share USB bandwidth through a host scheduled, token-based protocol. The bus allows peripherals to be attached, configured, used, and detached while the host and other peripherals are in operation.

USB devices can be functional or hubs. Each device has a number Endpoints which are indirectly accessed by the device drivers for data exchange. Devices with similar functions are grouped into classes in order to share common features and even use the same device drivers. Each class can define their own descriptors (class-specific descriptors), as for example, HID (Human Interface Device) Class Descriptors and Report Descriptors.

USB Host detect the attachment and removal of USB devices, manage USB standard control and data flow between the host and USB devices, collect status and activity statistics, control the electrical interface between the Host Controller and USB devices, including the provision of a limited amount of power.

The USB architecture comprehends four basic types of data transfers: Control, Interrupt, Isochronous and Bulk. Control transfers are used to configure a device and can be used for other device-specific purposes, including control of other pipes on the device. Interrupt transfers are typically used for devices that need to transfer data at regular period of time, and consequently must be polled periodically. Isochronous transfers need a guaranteed transfer rate and Bulk transfers are for large blocks.

SYSTEM ARCHITECTURE

Fig. 1 shows the architecture of the system.

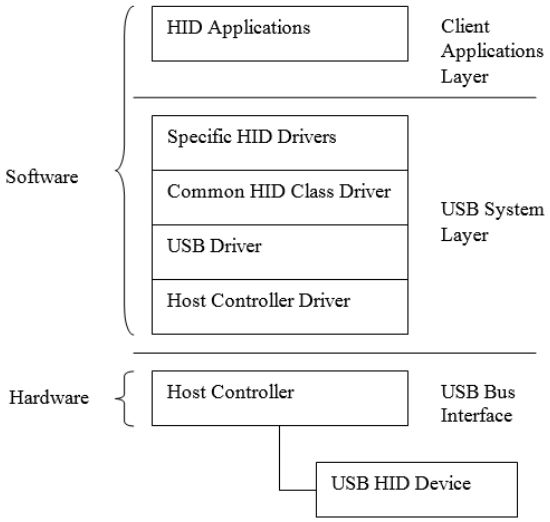

FIG. 1. System Architecture more References

There are three layers in the system, USB Bus Interface, USB System Layer, and Client Application Layer.

USB Bus Interface is hardware layer. The USB System uses the Host Controller to manage data transfers between the host and USB devices. The interface between the USB System and the Host Controller is dependent on the hardware definition of the Host controller.

USB System Layer has four software components: Host Controller Driver, USB Driver, HID Class Driver, and Class Driver Abstract.

Host Controller Driver is specific to the host controller used in the system. It abstracts the details of the host controller so that the higher layer software can interact with its USB device without knowing which Host Controller is in use. This encapsulation allows system to support different host controllers.

USB Driver handles the USB protocol specific operations and uses the lower host controller driver to communicate with USB host Hardware. It encapsulates the details of the USB protocol and presents to the higher layer an abstraction of a USB device.

Common HID Class Driver follows the HID Class Specification for communicating between the HID device and higher layer software. It uses the API provided by the USB Driver and provides common USB HID functionality. It makes the upper layer focus only on the unique additional functionality required for the specific devices.

There are a variety of different devices belong to HID class, such as keyboards, mouse, front panel controls. Specific HID drivers support the communication with different HID USB peripherals.

HID SOFTWARE DESIGN

Host Controller Driver

There are three industry standard USB host controller interfaces, EHCI (Enhanced Host Controller Interface, OHCI (Open Host Controller Interface), UHCI (Universal Host Controller Interface) and xHCI (extensible Host Controller Interface). The implementation of Host Controller Driver shall follow the specific host controller interface specification. In this paper we only implement OHCI.

Host Controller Driver manages the operation of Host Controller by communicating to the operational registers in the Host Controller and establishing the interrupt Endpoint Descriptor list head pointers in the HCCA [2].

HCCA is a very important data structure, in which interrupt Endpoint Descriptors are organized into a tree structure with the head pointers being the leaf nodes as the Fig. 2 shows.

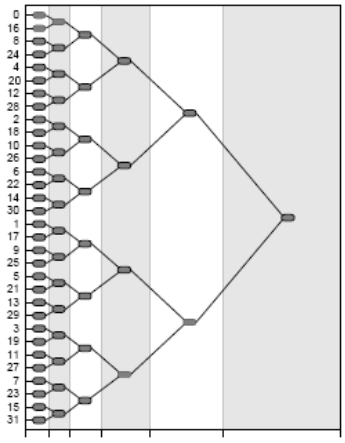

FIG. 2. HCCA

The Host Controller Driver provides many interfaces to USB Driver to make it communicate with Host Controller easily, such as hc_transfer (), hc_powermanage (), hc_getframeNumber () etc.

USB Driver

USB Driver component handles USB protocol operations and provides APIs to the class driver layer. Another major responsibility of the USB Driver component is to handle the plug and play capability to allow devices to dynamically insert onto the USB.

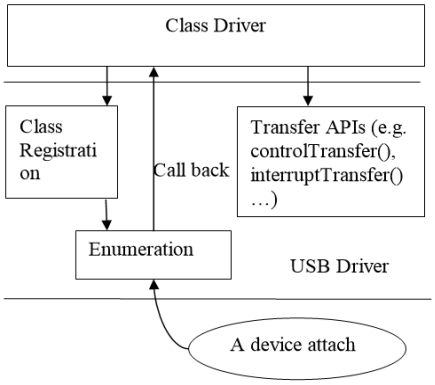

FIG. 3. USB Driver and Class Driver Interaction

Fig. 3 illustrates the three parts in USB Driver and the interaction between Class Driver and it. Class Driver will register with the underlying USB Driver, to indicate which types of USB devices should use the registered driver. This class registration method contributes much to the extensibility of the system.

The registered class driver provides several call back functions. These call back functions are invoked when a matching USB device is attached or detached.

When attach or detach a USB device, the host shall perform enumeration process. The enumeration module in USB Driver is designed to do this.

Common and Specific HID Class Driver

A HID class uses the following class-specific descriptors: HID, Report, Physical.

The Report descriptor in HID is unlike other descriptors. It is made up of items that provide information about the device.

As is known, a transfer is one or more transactions creating a set of data that is meaningful to the device. In HID, a transfer is synonymous with a report.

Common HID Class Driver component contains a parser used to analyze items found in the Report descriptor. The parser gets information as it walks through the report descriptor. The parser works as the Fig. 4 shows:

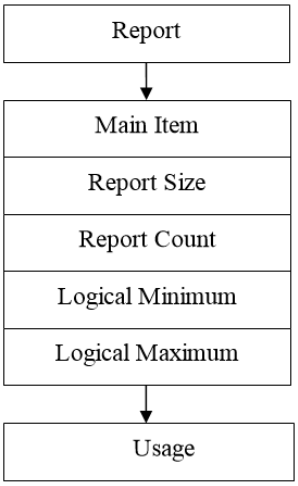

FIG. 4. HID Report Parser Process

Usages are part of the Report descriptor and supply an application developer with information about what a control is actually measuring. Usage tag indicates the device vendor's suggested use. It is used to choose the specific HID class driver to provide the functionality that common class driver not provide.

HID Applications

HID applications don't know any low level detailed transfers and protocols. When there is data from HID device, specific HID Class Driver can notify the application. And applications only use the interfaces provided by specific HID Class Driver to send data to device.

SUMMARY

With the growth of embedded market, the ease of data interconnection between various embedded devices is needed. Thus, USB continues to be the answer to connectivity for the embedded systems. So implementing USB in an embedded world has been a hot topic in the embedded and USB fields. In this paper, we have described the

method to implement a USB Host in embedded system and give out the architecture and design of HID class. In the future, we will enhance the extensibility of the design and implement other USB classes based on this architecture.

REFERENCES

1. USB Implementers Forum. Universal Serial Bus 3.2 Specification, Revision 1.0 [S/OL]. http://www.usb.org. Sep. 22, 2017.
2. USB Implementers Forum. Universal Serial Bus Specification, Revision 2.0 [S/OL]. http://www.usb.org. April 27, 2000.
3. USB Implementers Forum. Device Class Definition for Human Interface Devices. Revision 1.11 [S/OL]. http://www.usb.org. June 27, 2001.
4. Jiang Zou. Air mouse and keyboard combo pack based on USBD12 and accelerometer [J]. Modern Electronics Technique. Vol. 40 (2017) No.18, p. 101-103. (In Chinese).
5. Liu Zhiwu, Wang Jianyu, Zhang Lihui. Design of USB host device in embedded VxWorks based on PCI bus [J]. Application of Electronic Technique. Vol. 40 (2014) No. 2, p. 5-7. (In Chinese).
6. Xiaojing Wang, Design and realization of USB HID device based on C8051F MCU Family [D]. China University of Geosciences (Beijing). 2017. (In Chinese).
7. Cunkui Ye. Design and Implementation of USB Device stack [D]. The Huazhong University of scienceof Science and Technology, 2011. (In Chinese).
8. Mindshare, Inc., D. Anderson. Universal Serial Bus System Architecture. Addison-Wesley Developers Press, U.S., 1997.