Design of USB Composite Device and Host Driver Framework

Ranjitsinh Rathod
Embedded System Design
GTU PG SCHOOL, Gujarat Technological University
Ahmedabad, India
Ranjitsinhrathod1991@gmail.com

Rajesh Sola
Advance Computer Training School
C-DAC Acts
Pune, India
srajesh@cdac.in

***Abstract* -** The Universal Serial Bus (USB) has become a most popular communication interface among the personal computer and embedded devices because of its ease of use, low cost, data bandwidth and availability in most computing systems. There are many USB device are available of different USB class like bulk transfer, mass storage, USB CDC, USB HID and many other. All the devices have their own specific device driver for communicating with the USB host system. In the USB device each USB driver is communicate to each interface. So, this dissertation work is based on the implementation of the multifunctional firmware to support the multiple interfaces through a USB driver. Here on the host side any embedded development board or a Linux PC will be used and the USB driver will be develop to support the multiple interface. On the device side, any microcontroller board that supports the USB device will be used. So, between the USB host and device the communication is done using the different class like USB HID, bulk transfer, USB CDC, etc. At the end the outcome will be multifunctional USB firmware to support the two or three functionality.

***Keywords* -** *USB, Composite device.*

*****

I. INTRODUCTION

The Universal Serial Bus (USB) has been developed to overcome disadvantages of previously available communication interfaces; it is a fast, bi-directional, isochronous, low-cost, dynamically attachable serial interface that is consistent with the requirements of the PC platform now and future [1].

This work is induced from need of the multifunctional devices called Composite USB devices to support multiple independent interfaces in a device. In USB driver normally each device has one or more configuration and each configuration has one or many interfaces. Now for each interface we need driver to communicate with device. So here in composite device there is only one driver for two or more interface to communicating with composite device. Taking this need into consideration, this work describes flow from the specification of the USB host and the composite device for the design of it.

In this work, I present the development of the composite USB device and the host driver framework to support that composite USB device. On the hardware side, the design is based on the BeagleBone Black which will work as a composite USB device and a Personal computer with Linux which will act as a host. I used the bulk mode and CDC mode for USB communication and made a composite device. It communicates with PC through USB.

This paper is structured as follows. In section II I present the development of the hardware device, including the firmware design of the composite USB device. I cover the system architecture of all over the system in section III. In section IV I present the results of the communication through USB composite device and PC and then after finally I conclude my work in section V.

II. COMPOSITE USB DEVICE

The USB bolsters a wide choice of devices that range from lower-speed devices to higher-speed devices. Lower speed devices such as joysticks, keyboards and mice and higher speed devices such as digital cameras and scanners. The specification lists a number of device classes that each defines a set of expected device behaviors [6].

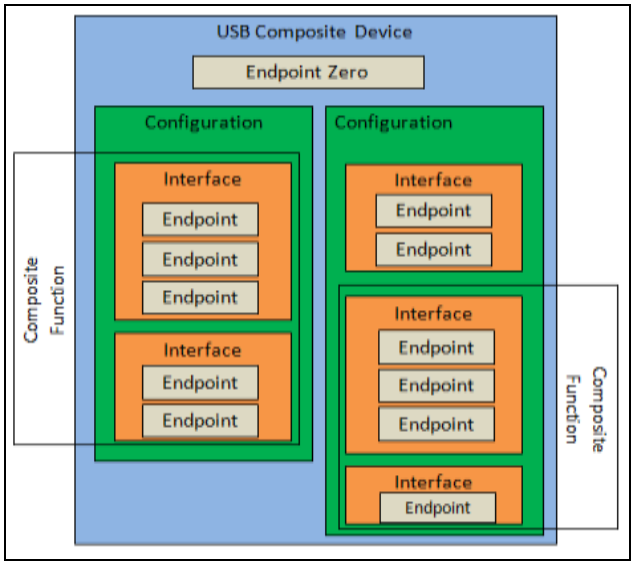

Figure 1. Composite USB Device

USB composite device is a multi-functional device with multiple, independent interfaces. The interfaces are defined by interface descriptor stored in the device. A composite device

 ISSN: 2321-8169
2520 - 2523

has one address on the bus but each interface has a different function and specifies its own device driver on the host [6].

Composite device have two or more interface like audio, video, mass storage or HID as an independent interface. The USB particular characterizes a composite class device as a device whose device-descriptor fields for device class and device subclass both have the 0 value [6]. A case of a composite class device is a multifunction device that performs printing, scanning, and faxing. In a device like this, each function is represented by a separate interface.

III. SYSTEM ARCHITECTURE

As shown in figure 2, the whole work is divided into three portions. In the first portion, developing a composite USB driver to support two functionalities for USB host in Linux. In the second portion, designing a firmware for beagleBone Black using TI StarterWare C-based non-OS platform to work as a composite device to control its functionality. In third portion, designing the user application to control the USB composite device so that it can read and write the composite device.

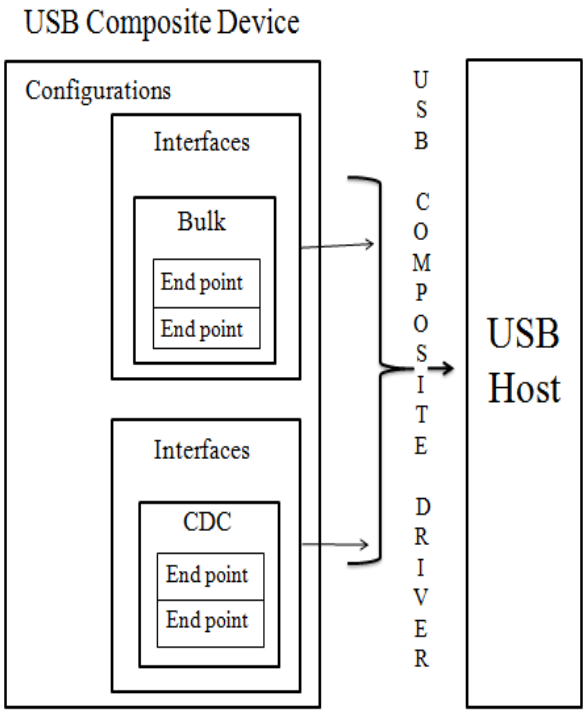

Figure 1. USB Composite Device and Host Architecture

A. Hardware Model

The basic idea for designing a proposed system is shown in the Fig. 2. Basic hardware that I used is BeagleBone Black with USB power cable and the standard serial FTDI cable. The connection of these two cables with the Linux PC and the BeagleBone Black for effective communication.

B. Software Model

In BeagleBone Black I used the TI StarterWare non-OS C-based platform to build the firmware of the composite USB device. So that the BeagleBone Black can work as a composite USB device when I attach this to the Linux PC.

For building host driver on Linux I use the C language and a USB device driver framework in Linux. The user application is also using the C.

Using the USB skeleton from Linux, we have developed USB driver. USB framework is as shown in Fig. 3.

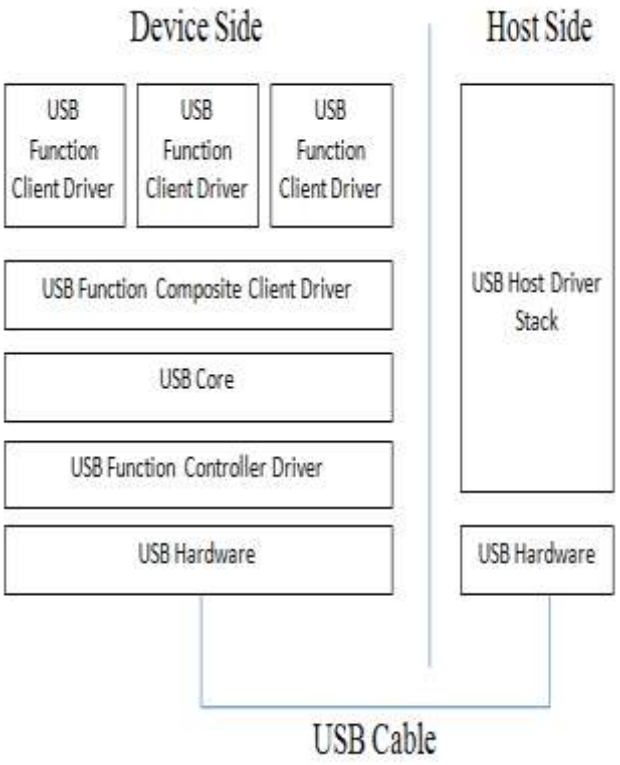

Figure 2. USB Host and Device Stack

IV. RESULTS

As discussed in the Proposed System the component are connected as shown in Figure 4.

Figure 3. BeagleBone Connected to Linux PC

A. *Device Side Communication*

Fig. 5 shows the communication between Device and the host are done. Here I controlled a GPIO of the BeagleBone Black that is inbuilt LED over USB.

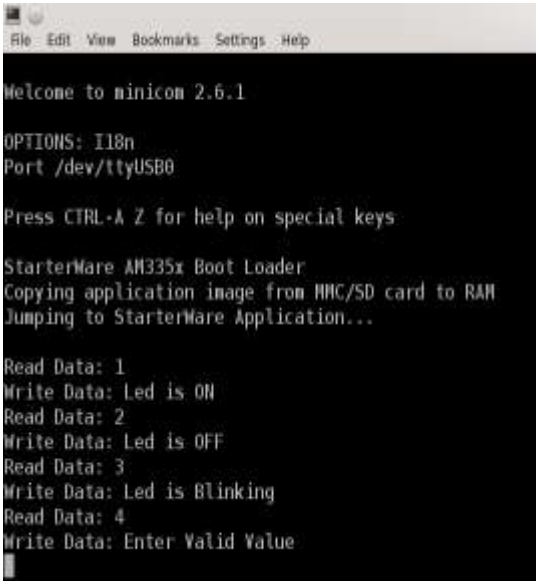

Figure 4. Serial Port minicom Output of BeagleBone Black

B. *Host Side Communication*

Here as shown in Fig. 6, host side from one user application I write the device file and according to data sent LED got controlled like on, off and blinking.

As shown in fig. 7 installation of the USB composite driver is done on a Linux system and a user application for controlling a device is also running on host side. After installation of the driver when I attach BeagleBone Black to the system it connects to the system using that USB driver via device VID and PID.

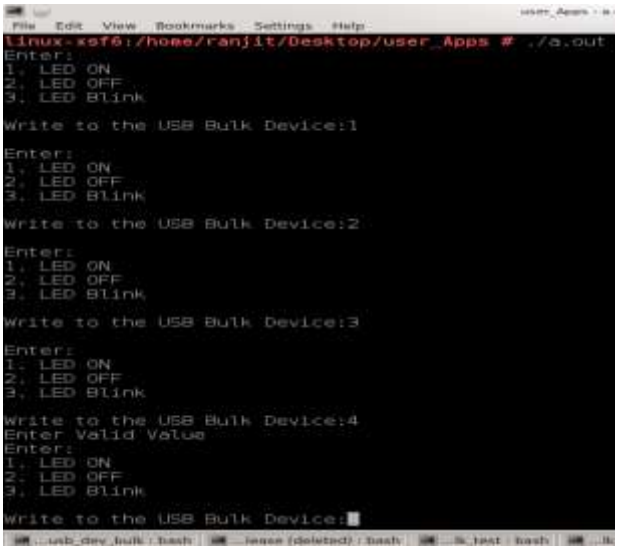

Figure 5. Writing to a Device for Controlling Device

After connection establish, I can write the device file generated in the /dev directory in Linux system and control the device as we want. Here, for example, I am doing the controlling of the inbuilt LED of BeagleBone Black but we can control all the GPIO pins of the BeagleBone Black.

V. CONCLUSION

I can conclude that the system of the USB host driver and the Composite device with the multiple interfaces can be done. For the composite device bulk transfer, CDC and mass storage USB profiles are use in the implementation of the USB device. But here I used only the bulk transfer and CDC for the composite device. Hence the USB host drivers have to build for the support that composite device to communicate with the device. Firmware for the specific embedded board has to build to support multiple functionalities as a composite device. BeagleBone Black is working as composite device and we can control its GPIO and other functionality from the host side through user application.

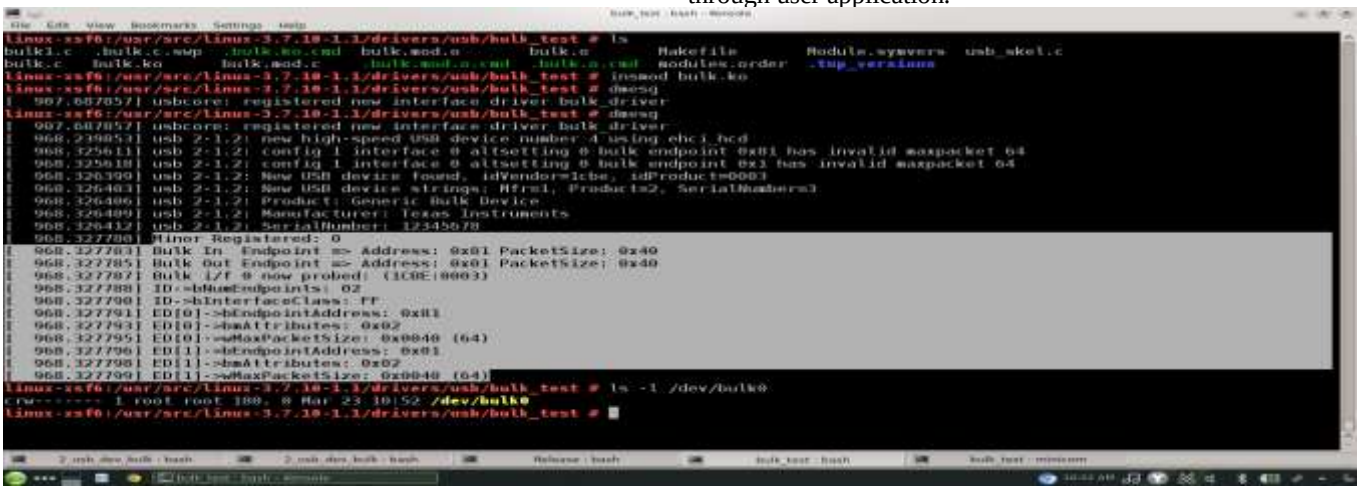

Figure 7. Driver Installation

REFERENCES

[1] "Universal Serial Bus Specification", Revision 3.1, July 26, 2013.

[2] Ruben Posada-Gomez, Jose Jorge Enriquez-Rodriguez, GinerAlor-Hernandez, Albino Martinez-Sibaja, "USB bulk transfers between a

PC and a PIC microcontroller for embedded applications" in CERMA '08Electronics, Robotics and Automotive Mechanics Conference, pp 559 – 564, 2008.

[3] Jodeit M., Johns M, "USB Device Drivers: A Stepping Stone into your Kernel" European Conference on Computer Network Defense (EC2ND), pp 46 – 52, 2010.

[4] Jan Axelson, "USB Complete: The Developer's Guide", 3rd Edition.

[5] RajaramRegupathy, "Bootstrap Yourself with Linux-USB Stack: Design, Develop, Debug, and Validate Embedded USB Systems", 1st Edition.

[6] "USB Class information", https://developer.apple.com/library/mac/documentation/DeviceDrivers/Conceptual/USBBook/USBOverview/USBOverview.html

[7] "The USB Composite Framework",

[8] http://lwn.net/Articles/395712/