Design of USB Driver and Firmware based in USBTMC Protocol

Aastha Shrivastava[1] Mr. Rajesh Sola[2] Mrs. G. Sasikala[3]
[2]Project Engineer
[1]Department of Embedded [3]Department of Electronics & Communication Engineering
[1,3]Veltech University, Chennai, Tamil Nadu, India [2]C-DAC Acts, Pune, Maharashtra, India

Abstract— USB is the most widely used bus for transfer of data anywhere. Along with the ease of use, fast data transfer is also one of the basic needs these days. In this paper, we are trying to replace the GPIB (General Purpose Interface Bus) pins which are used in Instruments with the most common and faster technology i.e. USB. For this purpose we are using industrial level protocol – USBTMC (USB Test and measurement class). USBTMC is a protocol which is built on the top of USB and from user's point of view, the device works like a traditional GPIB device with full speed of USB. Our aim is to bring this protocol from industrial level to consumer level so that general users can enjoy the ease of full speed in their electronic gadgets and other instruments.

Key words: USBTMC Protocol, GPIB (General Purpose Interface Bus), USBTMC Communication Model

I. INTRODUCTION

Human beings always crave to getting things done easily, comfortably and fast. Compact and faster things have always fascinated us. Same is true when it comes to the transfer of data between devices. There are two ways in which data can be transferred, wired and wireless. Wired transmission of data is faster and more reliable as there are less chances of data loss. So, it is always preferred over the wireless transmission when it comes to reliability and speed. There have been an evolution in the field of this technology. People focus more on making things compact and faster with same reliability. Transfer of data among devices via cable is very common. So it is necessary that it should be generalized. There are different kinds of interface connectors being used for this purpose, like GPIO, Ethernet, RS-232, USB etc. USB is very common and widely used connector. So, we have given a special preference to this interface connector. USB (Universal Serial Bus) is very widely used communication link for devices and so it is important that it should be fast and dependable. In the way to fit the need of technology, USB has overcome many disadvantages of previously used communication interfaces[1]. It is a fast, bi-directional, isochronous, low-cost, dynamically attachable serial interface that is consistent with the requirements of the PC platform for now and future[7]. USB is being used for wide range of peripheral devices like mouse, keyboard, printers, disks, audio and video devices. It is also used for monitoring, control, data acquisition, multimedia and security purpose[7].

This paper shows the architecture of an embedded device that uses USB port as an interface to send and receive data to the Host. Here, we are using PC as the host. Main differential of our work is that we are using USBTMC protocol for this purpose. We are using USB communication in bulk mode as it is the preferred mode for USBTMC communication because of its fast data transfer. On the device side, we are using NGX - Xplorer LPC1830 board which has microUSB type B connector. The data is transferred between host and device through USB using USBTMC protocol.

II. USB

USB (Universal Serial Bus) is one of the leading components in connecting host/pc to devices like cameras, printers, mobile phones and other consumer devices, networking devices, medical devices, car navigation systems etc. Therefore, USB has become an integral part of embedded devices.USB standard was developed by industry leaders such as IBM, Intel, Microsoft and HP. The main Moto behind developing this device was to provide an open architecture to designers to connect different peripherals [3]. Over the years, its specifications have been improved and matured as an interface protocol. Now, more than 6 billion devices use USB interface. The main advantage of using USB is that other communication interfaces like serial and parallel ports do not allow the expansion of port in PC, whereas USB is flexible to this and port expansion is very easy [1]. Also USB has increased the speed of data transfer between the devices.

III. INTERCONNECTION OF USB HOST AND DEVICE

USB is divided into 3 major parts namely,

- USB device,
- USB host, and
- USB interconnect

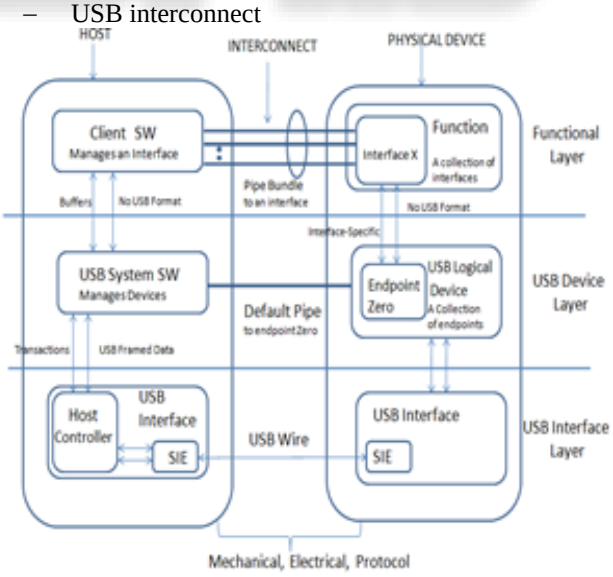

Fig. 1: USB Architecture.

USB device is the physical connection that performs the desired function, USB host is the controlling part, which controls the functionality of the device and USB interconnect is the communication media between the host and the device [2].

The whole USB system is made up of three layers – functional layer, the USB Device layer and USB bus

All rights reserved by www.ijsrd.com

interface layer which are common in both USB host and USB device.

IV. USBTMC PROTOCOL

USBTMC stands for USB test and measurement class. It is the protocol which is used for exchanging messages between host and devices. It also gives us the Idea about how to abort a pending I/O operation, and how to clear our device. This protocol allows the GPIB-like communication with USB devices. It allows a USB device to behave just like a GPIB device with all the advantages of USB, like high speed, large packet size and also it make the device compact in comparison to GPIB device. GPIB is General Purpose Bus Interface. It is common name of IEE-488, 8-bit parallel multi-master Interface Bus specification. It is being used in automated test equipments [3]. Use of USBTMC increases the speed of data transfer between the devices.

V. USBTMC COMMUNICATION MODEL

The general model of USBTMC interface is shown below [4]. The main thing is that USBTMC client software has to be able to support the endpoints of USBTMC.

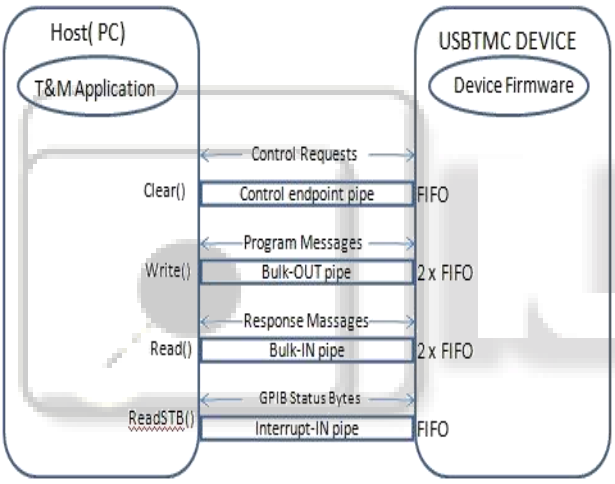

Fig. 2: USBTMC Communication Model.

Basically USBTMC have 4 endpoints, which are

1) Control endpoint - required by USB 2.0 specification.
2) Bulk-OUT endpoint – used by USBTMC for data transfer from Host to device. Host sends the USBTMC message commands to device through bulk-out endpoint and the device performs accordingly.
3) Bulk-IN endpoint – provides data path from device to host.
4) Interrupt –IN - used by device to send notifications to host. It is used by USBTMC subclass, so it is not necessary to be used by the USBTMC interface. If the interface descriptor has bInterfaceProtocol = 0, then no subclass specification is there. So USBTMC interface does not need to have an Interrupt – In endpoint.

VI. USBTMC SPECIFICATIONS

USBTMC specifications are parallel to USB 2.0 specifications. In this specification, the protocol requirements have been set to make the host multitasking and multiple implementations can be managed [5].

The USBTMC protocol needs the following –

- Minimal devices.
- Devices that communicate with IEEE 488 messages.
- Devices with sub addressable components.

VII. USBTMC DRIVER

Classic method of implementation for text based I/O on UNIX and Linux is through a character device driver. A character file allows the end user to use basic file I/O mechanisms for communication, i.e. no special API is needed to communicate with the device.

USBTMC driver use character device driver for USBTMC instruments. There are several benefits of this such asthe driver is independent of the programming language/environment, code is portable etc.

For starting the interaction, USBTMC driver needs to be registered with the USB core. It allows the USB core to understand the type of driver, services offered by the driver and the devices which need to be connected. After registration, the devices are filtered by various attributes like device's vendor ID, product ID or device class. For USBTMC, it is most appropriate to filter by device class (application specific) and USBTMC subclass. It makes the USBTMC driver universal. It means, USBTMC driver will get notified whenever a USBTMC- compatible device is attached, independent of its vendor or product code.

VIII. USBTMC FIRMWARE

The firmware is the combination of hardware device and it is permanent software which is programmed into the ROM. It gives the way in which device should work and communicate with other devices. It controls the entire communication process. When the device is plugged in, the micro-controller starts enumeration and downloads USB descriptors and firmware over the USB cable. Then, enumeration is done again and the device is defined by the downloaded information. The USBTMC firmware is implemented on USBTMC device. The firmware design should be on bulk transfer mode. It make the data transfer speed faster and easier as USBTMC allows fast bulk – IN and bulk – OUT rate.

IX. USBTMC HARDWARE INTERFACE

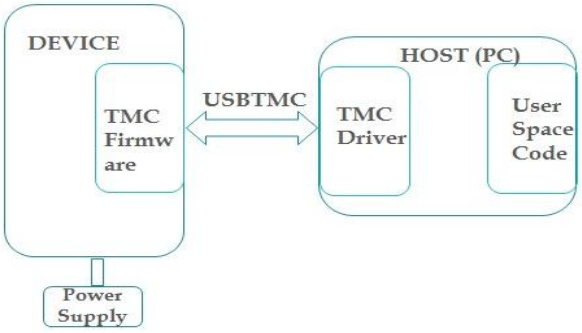

Fig. 3: Block Diagram

Figure 3 shows the block diagram of USBTMC based driver and firmware. The data is transmitted based on the USBTMC protocol. The transmission media is cable with

All rights reserved by *www.ijsrd.com*

USB 2.0 specifications and the transmission flow is between the host and device. We need to write a firmware on the device side from scratch. Also we need to write a driver program and configure both according to USBTMC specifications. After this, we have to test it through user space code and then we can get a confirmation that our setup is working properly according to USBTMC requirements.

X. CONCLUSION AND FUTURE WORK

The architecture of USB driver and firmware based on USBTMC protocol discussed in this paper enables the faster data transfer rate and it also makes the device more compact and reliable for the use. This architecture can totally remove the GPIB pins from the instruments. Using USB in place of GPIB can reduce the size of whole system. This architecture helps in bringing the USBTMC protocol from industrial level to the consumer level. As a future work, this protocol can be implemented in all devices. It will reduce the work of programmers and designers as driver will become universal and there will be no need to write different driver for different applications.

XI. ACKNOWLEDGEMENT

Any research work or any project is not the work done by a single person, this project also have contributions of many people who made this research survey to get completed. So the author would like to thank and give credential to Mr. Rajesh Sola, Mrs. G. Sasikala, Prof. Aaradhana A. Deshmukh and Mrs. Vaishali P. Maheshkar for their continuous encouragement and motivation. The author would also like to thank R&D Dept., C-DAC, Aundh Pune.

REFERENCES

[1] "Universal Serial Bus Specification", Revision 2.0, April 2007.
[2] Rajaram Regupathy, "Bootstrap Yourself with Linux-USB Stack: Design, Develop, Debug and Validate Embedded USB", Course Technology PTR: Stacy L. Hiquet, © 2012 Course Technology, a part of Cengage Learning.
[3] Axelson, Jan. "USB Complete", Lakeview Research LLC, Third Edition, 2005.
[4] Universal Serial Bus Test and Measurement Class Specification (USBTMC). USB Implementers Forum, Inc.2003.
[5] Universal Serial Bus Test and Measurement Class, Subclass USB488 Specification (USBTMC-USB488). USB Implementers Forum, Inc.2003.
[6] User Manual of LPC 18xx ARM Cortex-M3 microcontroller (UM10430), rev.2.6 – 17 February 2015.
[7] Ruben Posada-Gomez, Jose Jorge Enriquez-Rodriguez, Giner Alor-Hernandez,Albino Martinez-Sibaja, "USB bulk transfers between a PC and a PIC microcontroller for embedded Applications", 978-0-7695-3320-9/08 $25.00 © 2008 IEEE DOI 0.1109/CERMA.2008.21.
[8] Dong Zhanyong , Zhao Hao, "Data Transfer Principles and Implementation in USB Microwave Power Sensor", 2014 Seventh International Symposium on Computational Intelligence and Design, 978-1-4799-7005-6/14 $31.00 © 2014 IEEE DOI 10.1109/ISCID.2014.58

All rights reserved by *www.ijsrd.com*