USB to USB Bridge for Computer Independent Data Transfer

Bhushan Kamble
Student of E&TC
Sinhgad College of Engg, Pune

Amey Priolkar
Student of E&TC
Sinhgad College of Engg, Pune

Ashitosh Kamble
Student of E&TC
Sinhgad College of Engg, Pune

Prof.S.O.Rajankr
Dept. of E&TC
Sinhgad College of Engg, Pune

ABSTRACT
Today, the need and the importance of the data in human life are well known. It has become an inevitable part of day to day life. This paper discusses a modern approach towards the way the data is transferred to and fro in the USB mass storage devices using a computer. The aim of the proposed system is to remove the every time need of the computer for the data transfer. To do so, the system uses an independent solution for the problem – USB to USB Bridge. The concept of the bridge will make it possible to carry out the mass data transfer anywhere, anytime. The bridge is built around the ARM9 processor to accommodate all the requirements that the end users may want. The portability and small foot print are the major advantages of the bridge. It is an embedded solution to a practical problem.

General Terms
ARM Based Embedded Systems, Portability.

Keywords: USB, Data Transfer, ARM9, Bridge.

1. INTRODUCTION
In general terms, the bridge is an architecture that carries the things across some gap. The USB to USB Bridge is also based on the same concept-only thing is that it carries the data between the devices. The current scenario is that you need to have a USB enabled computer to do the task of data transfer. It is difficult to take the computer always with you wherever you go for the sole purpose of data transfer. Rather it is inconvenient to do so.

So a solution is provided by means of implementation of the bridge. The small footprint and ease of portability makes it a choice for the data transfer. This bridge will help the user to select a particular data file from the mass storage device connected to one of the ports and transfers it to the other mass storage device using some controls like explore, list, copy provided on the front panel. The data onto the USB mass storage device will be explored using the USB host. The LCD displays the list of explored files from the mass storage device and the keypad does the selection of the operations to be performed.

The linux environment provides the basis of all functions for the embedded design. The USB host controller embedded into the ARM architecture does all of the major tasks. The interfaces provided for ease of access give the feasibility of selecting the operation to be performed. The DMA is activated when the data transfer is initiated. Bidirectional data transfer is possible by means of half duplex communication.

Objectives of the design:

- To enable the USB to USB data transfer by using an embedded system.
- To make speedy data transfers through the use of the DMA controller.
- Making the product more user-friendly.
- To develop the system as a product.

2. REQUIREMENTS OF SYSTEM
The motivation of the conceptual design of USB to USB Bridge was an answer to a question; what can be the solution to make the computer independent data transfer using USB? This question is of most importance for engineering design. After making a lot of explorations in search of answer to this question and by questioning to ourselves, the only available answer resulted is that develop a system that will perform the same task as that of the computer.

This answer satisfies the common reader but the real challenge begins here. Making a system that will do the tasks of a computer – made us to think a little because the answer directly suggests implementing a system that will handle USB protocol along with the other processes.

Thus, it has been the primary objective of this study to present as an answer - a realistic and practical concept of USB to USB Bridge.

As majority of the tasks are for completion of the operation of data transfer, some basic guidelines were formed. The requirements are based on the needs and prerequisite those are needed for the proposed system. This frame of guidelines is forming the basis for the design and implementation of the bridge. These are nothing the building pillars of the whole idea. The whole system has been build around these requirements.

Requirement 1: The first and very basic need was to have a look over the USB protocol. The protocol is easy to use but is somewhat difficult to implement in high and full speed modes. This requirement is to give realistic background of the study. [3]

Requirement 2: Commercially versatile technologies are needed for bridge. The present technology is the one that end users don't afford. One cannot use expensive technology in the development. More design efforts are required for application at lower end than using top end hardware.

Requirement 3: The data transfer speeds of the bridge should be commercially competitive. It is acceptable that cost of USB to USB Bridge can be lower than existing systems.

Requirement 4: The USB to USB Bridge will be placed on a low end of user systems by commercial launch. The product needs to survive the market demands.

Requirement 5: The customers for the USB to USB Bridge may be from the particular zone of the society. Adoption by all zones makes it possible to survive in the market. People using the bridge will take advantage of the low power

consumption of the system, even if it would be too small for industrialized zone. It is important for the design to know practical demand for this type of varied use.

Requirement 6: Design a basic model with minimum power allowing for system growth in the future. Once the USB to USB Bridge demonstrates technical and economic feasibility of implementation, larger and more stable systems such as the High Speed System will be expected to follow. In this respect, the design of the bridge should consider evolution to make a larger system from the first operating model.

Depending on the requirements it is very important to choose the suitable interfaces, hardware, and software from the many options available. Let's look forward to the same.

- **Interface**

Why USB when there are lot many interfaces available in the industry? The answer lies within the advantages that USB provides over the other interfaces and the number of USB devices in the digital world. More than 6 billion USB devices are sold in total. [9]

USB is an easy to use interface. From the developers as well as users point of view, it is has the advantages like ease of use and ease of accessibility. Also the speed, hot pluggability and low power consumption makes it a choice to be used in the portable embedded system like the bridge itself.

The simplicity that USB specifications implements in the hardware and software is of most importance when compared to the hardware complexity of the IEEE1394-Firewire bus. Thus it became an obvious choice for the use in the bridge as the main interface.

- **Processor/ Microcontroller**

In the proposed system, data transfer is the main task. Thus, the system needs a processor that will handle the data transfer and related processes speedily. The processor should have a USB host with drivers available for it.

Also the processor should be low power consuming, and should provide compatibility with the software and or firmware. The operating system used by the processor plays a major in the development of the system. The processors from ARM were standing above all considering the demands of the embedded product. Here, the processor that provides all above requirements is the ARM9. Its controller version is EP9302. The ARM processors are made for embedded design. The flexibility and choices in interfaces as well as robustness provided by the ARM is very high as compared to others. This makes it an appropriate selection for the design and building of the product.

3. IMPLEMENTATION

3.1 Hardware

As the proposed system is a specific application of a large system, the hardware selection and implementation is very important. The main hardware of the system is the processor. As the system will be used by different users, so along with the processor the other user interfaces will be needed. Here, in the system, LCD and the keypad are used to provide menu driven system that will put the options on LCD screen and the options will be selected with the help of a keypad.

- **USB2.0**

USB is a master-slave bus with one master and multiple slaves. The master is called a host and the slaves are the peripherals. Only the host has the ability to initiate the data transfers; the slaves only respond to the host's instructions- they never initiate transfers. A PC is a common host. But the proposed systems embedded host does not involve a PC, instead uses the hosts in the microcontroller.

The USB protocol is quite different from the other interfaces. Figure 1 shows the format of an USB Frame. It is based on the single host and multiple of downstream USB port. The USB devices are connected to it in a tiered star topology. The hubs make it possible to connect many devices in series. The root hub is in the host controller which acts as data controller.

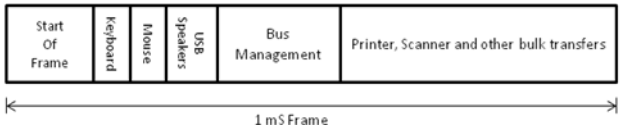

Figure 1. USB frame format

The communication made by USB is based on logical channels - known as 'pipes'. It connects the host controller to the device endpoint. The endpoint is a logical entity which resides onto the device. The connections established are 1 to 1 for the endpoints in pipes. A USB device can have 32 endpoints- two of which are reserved. So a total of 30 are present for normal use.

The data transfer is having four types:

1. *Interrupt transfers:* for the devices needing quick but guaranteed response(e.g. pointing device)
2. *Isochronous transfers*: For some fixed data rate but data loss may take place(e.g. audio, video)
3. *Control transfers*: used for simple status check.
4. *Bulk transfers*: uses available bandwidth with no fixed data rate (e.g. file transfer).

Depending on the type of data transfer, there are two types of pipes: stream and message. The stream pipe is connected to a unidirectional endpoint for the interrupt, isochronous and bulk data transfer modes. The message pipe is connected to bidirectional endpoint for control data transfer. The frame format of the USB communication is shown in the figure 2.

The speeds offered by the USB are defined in the USB2.0 specification:

1. Low speed: 1.5 Mbps
2. Full speed: 12 Mbps
3. High speed: 480 Mbps

- **ARM9**

Data transfer is the main task in the system. Thus, a processor that will handle the data transfer processes speedily is needed. The processor should have a USB host with USB device drivers installed in it for plugging in a mass storage device which will be used to transfer data from one device to other. Here, the processor that provides all above requirements is the ARM9.

- **EP9302 Development Board**

In the implementation of this system, the development platform used is the ARM9 development board- SBC9302. It is based on Cirrus Logic EP9302 processor. It has ARM920T at its

core. The EP9302 is an ARM920T based system-on-a-chip design with a large peripheral set targeted to a variety of applications. Various interfaces along with the connections in the ARM architecture are shown in figure 3.

The EP9302 features an advanced ARM920T processor design with an MMU that supports Linux, Windows CE and many other embedded operating systems.

The EP9302 has a dual port USB host which is very important for the system. The USB host Controller Interface provides full speed serial communication ports at a baud rate of 12 Mbits/sec. Figure 2 shows the detailed block diagram of EP9302

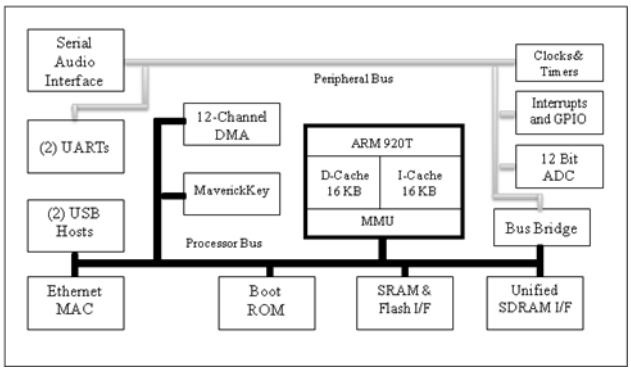

Figure 2. Block diagram of EP9302

This can support both low speed and full speed USB device connections. Root HUB is integrated with 2 downstream USB ports. It also supports power management. The open HCI host controller initializes the master DMA transfer with the AHB bus.

- **LCD**

To provide User interface, the bridge uses the LCD (Liquid Crystal Display). The contents of the mass storage device are displayed on the LCD. This helps the user to view and select the files or folders of interest from the USB device. Also the options like select, copy for data transfer are put on to the LCD.

- **Keypad**

The keypad is one of the User interfaces which are provided in the system. The main purpose of the keypad is for navigation through options and files. The contents displayed on the LCD can be selected by using this keypad. The user will be able to select any option from the options that are available on the LCD like COPY, PASTE, EXPLORE, etc. just by pressing the corresponding key. The system uses a 4x4 keypad which is easy to interface and work with.

3.2 Software

The hardware used supports operating systems like LINUX and WinCE. The OS is needed for the system as all the initializations, drivers , data and flow control, error handling, resource sharing as well as multitasking is possible only with the help of a good operating system. For EP9302, LINUX 2.4 operating system is employed.

A Linux system can actually be adapted to work with as little as 256 KB ROM and 512 KB RAM. So it's a lightweight operating system to bring to the embedded market. Drivers and other features can be either compiled in or added to the kernel at run-time as loadable modules. This provides a highly modular building-block approach to construct a custom embedded system. The Linux kernel provides an extended support to the drivers for the development boards.

The compiler used is 'gcc' (arm-linux-gcc). The gcc helps to develop the program as the system works with the linux environment.

4. ARCHITECTURE OF THE SYSTEM

As shown in figure 3, the architecture of the system mainly consists of a processor, a LCD and a keypad. The processor and the USB host work in the LINUX environment.

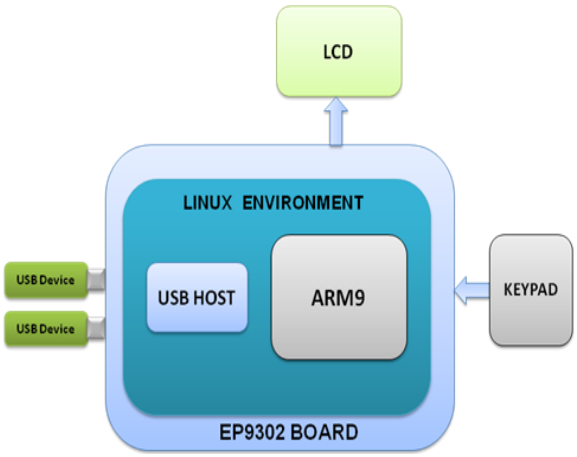

Figure 3. Architecture of the system

All the initializations for USB host and processor are done using the LINUX kernel. The USB host allows the processor to gain the control of the USB devices. Various LINUX commands [4] are used to access the information such as files, folders, total size and other system related information of the USB device. Once the USB mass storage device is connected, the contents of the USB device are displayed on the LCD.

4.1 Implementation Algorithm

i. Select the suitable development board.

ii. Check whether the OS is ported or not.

iii. Port the OS and install the USB device driver.

iv. Connect the USB device to check functionality of the USB device.

v. Interface the LCD and keypad as a User interfaces.

vi. Check the communication between the USB device and the board. (This is done by plugging in the mass storage device in the USB port and by looking the response on the HyperTerminal. We get all the information of the mass storage device which is connected to the board as a response.)

vii. Explore the device contents on LCD.

viii. Select a particular file, and by using the option COPY, copy that file to destination device using keypad.

ix. The selected file is then copied into destination USB device that is connected in one of the two USB ports.

x. If another copy operation is to be performed, go to step vi.

xi. Terminate the process.

5. WORKING OF THE SYSTEM

5.1 Hardware Initialization

To get the system start functioning, some set up controls like baud rate, flow control etc need to be done. As soon as the board is powered up, the linux starts booting. All the preliminary checks are done for check functionality of the board.

The USB device should be connected to the hardware after the system boots. When a USB device is connected to the hardware, the initialization starts. Normally it takes 1 or 2 seconds to initialize it. Consider a case when we connect the USB device to the hardware in between the boot process, then the error comes into picture as "USB device not recognized".

5.2 Execution of the task

Instead of showing the process in a descriptive manner, we can represent the flow of execution in flowchart format. The following flowchart shows the stepwise flow of the execution of the task. First step show the initialization of the system. Next step defines the initialization of the USB peripheral and system components. The steps ahead define the exploring and selecting the operation. The second last step shows the operation completion whereas the last step shows the termination of the process. The flowchart of operation is as shown in the figure 4 below.

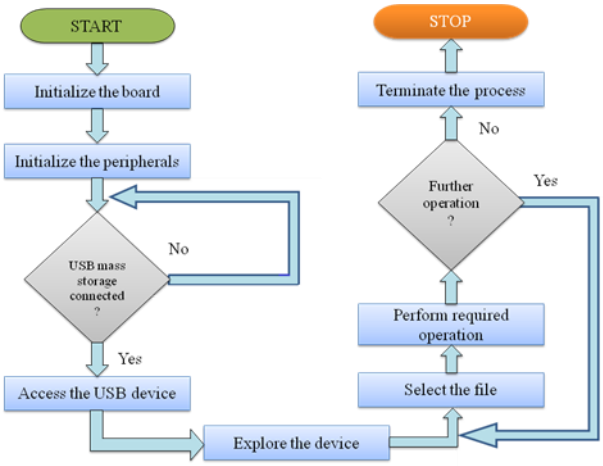

Figure 4. Flowchart of task execution

5.3 Termination

When the USB device remains unaccessed for more than 3msec, it (the USB device) goes into the sleep mode. It is one of the features of the USB2.0 specification. Once the USB device goes in to the sleep mode, the termination of the process can be completed i.e. the USB can now be removed or ejected.

6. TESTS AND RESULTS

The EP9302 hardware was tested to check whether the USB drivers are installed or not. The boot process of the embedded linux is shown in the figure 5. The next test performed was to check whether the hardware can recognize the USB device or not. This test shows the result by displaying "USB device is recognized" when USB is connected to the hardware.

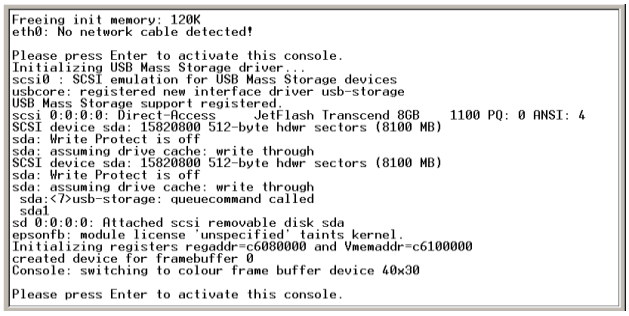

Figure 5. Linux boot window in the HyperTerminal

After confirming the above tests, the test was carried out to check whether a file can be copied from the PC to the USB. This test ran successfully. Going further, the proposed system was also tested to copy a file from one USB device connected at one end to a device at the other port using the linux commands like ls /media, cp. The "minicom" facility in the Linux kernel acts in the same manner as the "HyperTerminal" in the Windows.

The user interfaces (4x4keypad and 16x2 LCD) were also worked out. One important test was also carried out to check for the bidirectional data transfer in the half duplex mode. To do so, a single file from mass storage was copied to and fro using the HyperTerminal. The data transfer in half duplex mode is also possible.

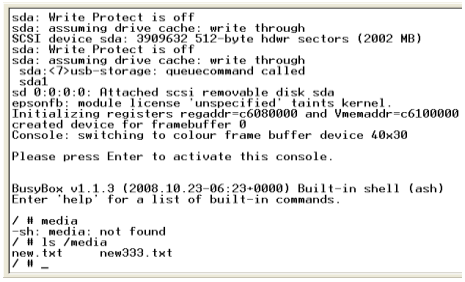

Figure 6. USB mass storage device explored

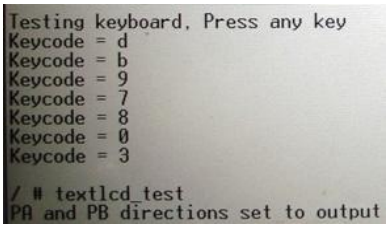

Figure 7. Keypad test output

7. ADVANTAGES

7.1 Battery operated

As the whole system operates on the 5 V supply and the core operates on 3.3V and also is very low power consuming so the system can be made to operate on the battery.

7.2 Portable

The proposed system can be made portable by means of making it a standalone platform. The processor along with the peripherals makes it to work independent of PC. One can carry out the data transfer anywhere, anytime.

7.3 Power Optimization

As both the processor and the USB2.0 specifications are designed to keep the lowest possible power consumption, the power optimization is done by using it only when it is necessary.

8. CONCLUSION

The USB to USB Bridge gives a concept to study the USB protocol as well as the working of the USB host along with the processor in group. A set of basic requirements were defined and used for design work of the USB bridge concept based on practical tests and results. The system - if mass produced will be costing very less as compared to the costs incurred in the development. The ease of use and portability makes it a powerful but unique tool to do the data transfer. The problems regarding the embedded OS independent solution can also be resolved using the linux kernel.

As the system is going to work as an application of the major system, the number of applications is limitless for the bridge. For example, the CAN bus implemented in automobiles can also be made to work with the USB protocol for faster data access. The systems like vehicle scanner, electronic testing platform for automobiles, CNC machines are also capable of deploying the bridge for their sole purpose of fulfilling the system specific goals. One more application to which the bridge looks forward is as the detachable device that will bridge many interfaces e.g. a tiny module that can be connected to the handheld and mobile gadgets or units.

As a result, a more generalized but unique concept of USB to USB Bridge will be developed. In this respect, this study has given us a remarkable insight into the future of development of high speed USB 3.0 Bridge.

9. REFERENCES

[1] ARM926EJ-S Datasheet.

[2] Datasheet of EP9302 from Cirrus Logic.

[3] Jan Axelson “USB Complete”-third edition.

[4] SPJ Embedded Technologies- “SBC9302 User's Manual”.

[5] William Shotts, Jr., “Linux bash shell programming tutorials”.

[6] Xiaoping, Shelei, “Study on System Verification of USB2.0 Interface Protocol Control Chip Hardware Design”, ICEE 2007.

[7] Jia Luo, YingQing Xia, AiSheng Li, Cheng Yang, “The design and realization of DSRC logic analyzer based on USB”, ICFCC 2010.

[8] Li Ying-lian, Hu Bing, “Design of transient recorder based on USB2.0”, ICECE 2010.

[9] Gong Yun, Sun Li-hua, “Analysis and Implementation of USB Driver Based on VxWorks”, ICECE 2010.

[10] "SuperSpeed USB 3.0: More Details Emerge". 6 Jan 2009. http://www.pcworld.com/article/156494/superspeed_usb_30_more_details_emerge.html.

[11] http://www.cirrus.com/en/products/ep9302.html

[12] http://www.linuxfordevices.com/c/a/Linux-For-Devices-Articles/