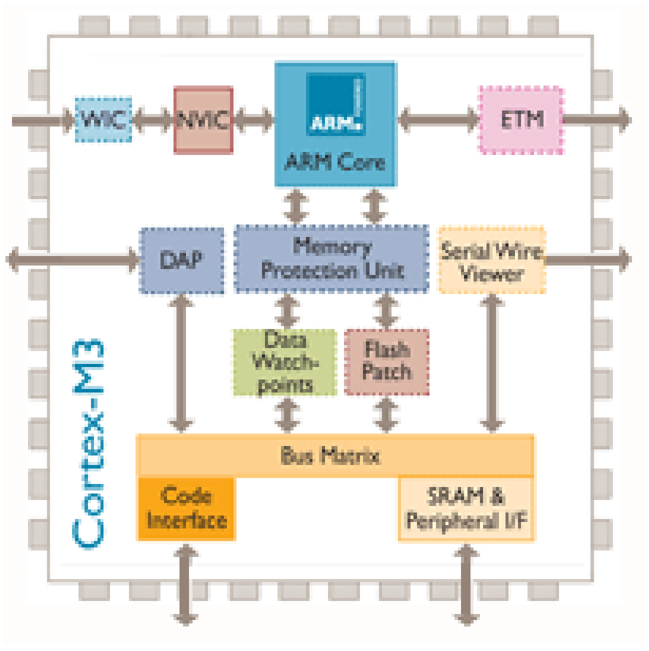
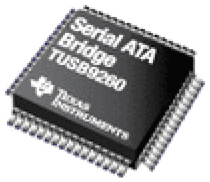


Design Document



USB 3.0 write blocker

Senior Design Group Dec 1108

Team members

Chen Zhao

Elphas Sang

Yan Fang

Advisor

Dr. Zhao Zhang

Iowa State University

October, 27 2011

Contents

Executive summary

A write blocker is used to protect a hard drive or other massive storage devices from data contamination during data communication with a computer. There are currently many USB 2.0 write blocker products in the market, but none for USB 3.0. The project's goal is to work out a prototype of such a USB 3.0 write blocker to protect SATA hard drives from accidental data writing by a USB 3.0 PC.

This project requires at least one programmable hardware development board supporting USB 3.0 standard and a SATA interface. The project also requires a corresponding software platform to perform firmware design.

The basic technique of the project is to conduct firmware level embedded system programming on a USB3.0 bridge board. The processor on this bridge board is to be programmed in such a way that it captures and filters all writing packets to keep the SATA hard drive free from data modification. There are two main issues in this project. The first step is to set up both hardware and software environment which consists of a working development board and a code development and compiling tool. The second step is to implement firmware design by trying various versions of firmware to achieve the goal.

The major difficulty for this project is lack of documentation and embedded development environment, which includes no sufficient programming documents for the development kit itself and difficulty finding a proper and efficient code compiler.

The expected product for the project is a simple device. It would be a small blocker device with a well-programmed, self-booted microprocessor acting as a filter between a USB 3.0 PC and SATA hard drive.

Acknowledgement

Electronic Crime Institute

Iowa State University

Texas Instrument

I. High Level Design

Problem Statement

General problem statement

1. USB 3.0 to SATA data adapter
2. Correct firmware design

General solution approach

1. The development kit for this project is the TI TUSB9261USB 3.0 to SATA bridge board. What we are supposed to do is study the development environment and figure out various ways of performing I/O control which may not be present on the product's instruction manual from the manufacturer. Then we prepare software environment supporting the hardware device for code development.

2. Blocking is realized in firmware. How blocking is realized depends on the way of C programming firmware design for this device. What we do is perform our new design by changing the default firmware of this product, which is originally designed simply as an adapter between USB3.0 and SATA.

Functional decomposition

1. USB 3.0 to SATA adapter

2. Support for Mass-Storage Devices Compatible With the ATA/ATAPI-8 Specification

3. UASP protocol compliant mass storage device suitable for bridging hard disk drives (HDD)

4. Auto-load firmware (boot code and program we write)

System analysis (Design tradeoffs)

1. Maintainability prefers adaptability. The program we write works as self-booted code for the ARM processor and it is maintainable under any circumstances. Even after the final product is finished, the firmware is still open to change. But the

product is functionally fixed to mainstream massive storage devices on USB PCs. It may not work for some consumer storage devices not complying to SCSI standards.

Technology platforms

Development kit:

TUSB9261bridge board

Code composer studio

Flash burner utility (for firmware writing)

Features

1. SuperSpeed USB 3.0 Compliant
2. Integrated development environment for C programming
3. Separate software tool for firmware writing on the device
4. Integrated ARM Cortex M3 Core

II. Detailed Design

System architecture

This development board we use is an ARM cortex M3 based microcontroller based USB 3.0 to SATA bridge board. It provides various necessary firmware to implement bridging for hard drives, solid disk drives, optical drives and other SATA 3.0G standard drives.

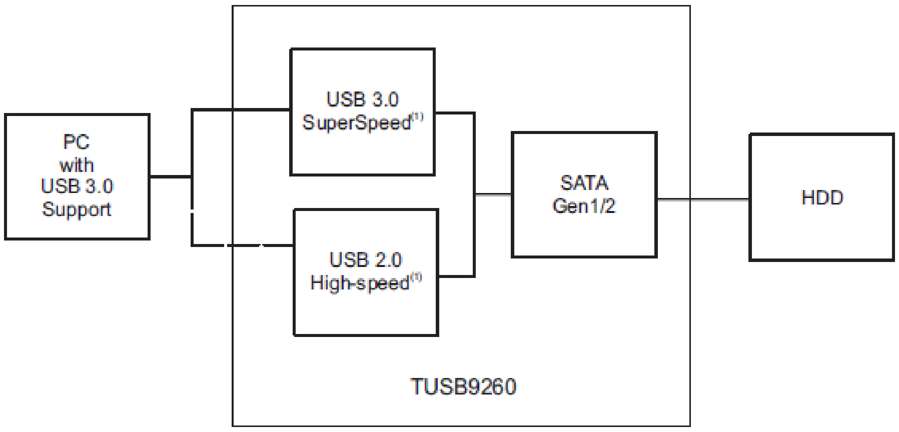


Figure 2.1 clarifies the connections that require our consideration on the board.

What we are studying is the connection between the host (USB 3.0 PC) and massive storage device, in super speed transfer mode.

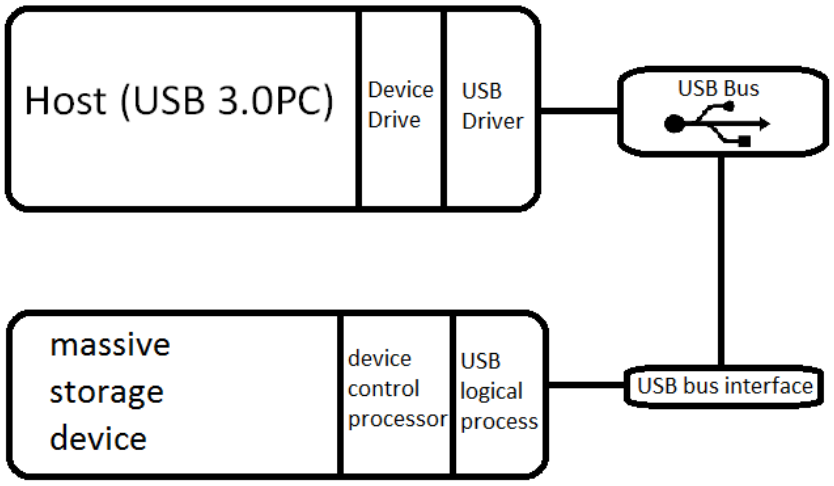


Figure 2.2 Host/ Device Conceptual View

Design modules

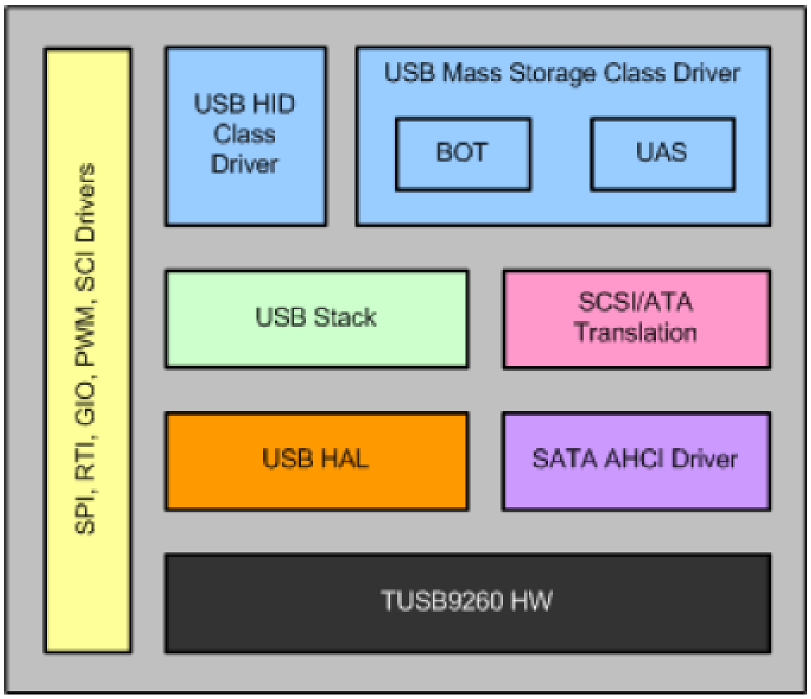


Figure 2.3 software architecture

The firmware layout of the original product TUSB9261 is shown in figure 2.3. SCSI commands for mass storage devices are translated into ATA commands.

The project is focused on learning SCSI command handling and control in the SCSI/ATA layer shown in figure 2.3

In SCSI command architecture, the firmware considers four different forms of writing operations, each with a specific operation code, as shown below.

SCSI_WRITE6	0x0A
SCSI_WRITE10	0x2A
SCSI_WRITE12	0xAA
SCSI_WRITE16	0x8A

The SCSI module in the firmware catches these writing commands by OP code identification. To disable these operations, we modified the return value of the routine that reports command status.

Functional module integration

Disabling a writing command can be realized in two different ways. One way is to simulate a writing error on the board, so that the USB driver on the host PC receives the error and report to the operation system. In this way, the user will receive a non-writable error message from the OS.

The other way is to cheat the host by returning invalid parameter bits. In this case when the user is using the device, he/she gets no response from the PC to check if the writing is unsuccessful.

The method we implemented was that we made the board ignore writing operations. An invalid status parameter would be returned in that case so that the command handler cannot process it as a valid writing command. The behavior of the device under this implementation may vary depending on different operating system.

In Mac, the operating system complains about not being able to write into a storage device, which is what’s expected to see.

In Windows, due to device writing buffer, the operating system may not complain the writing disability but it reports delayed writing error and may get into not-responding status. In this case, additional system configuration may be required when this device is used in Windows.

III. Testing and Evaluation plan

Unit testing

The team is working on developing on a USB 3.0 write blocker prototype that meets the specification required by the client.

Integration Testing

Currently there exists a USB 2.0 write Blocker and USB 3.0 does not exist. USB 3.0 should allow Bi-Directional Data Transfer compared to USB 2.0 which is one directional at a given instance.

USB 3.0 has less Power Consumption rate has opposed to USB 2.0

Faster transfer rate should be maintained in USB 3.0.

System testing

Intel X58 SATA 6 GB/s USB 3.0 ATX Intel Motherboard: This board provides newest features and the latest technologies. The motherboard also supports the latest Intel® Core™ processors.

Intel® Core™2 Duo processor E8000: The processor is efficient in that it reduces the amount of power need by the CPU therefore less CPU heat and prolonged system's life.

Hard Disk: Hitachi 7K3000 7200RPM SATA HD:

Big storage capacity: The hard Disk drive provides an enormous three terabytes of storage capacity and a 7200 RPM performance in a standard 3.5 –inch form factor

Excellent performance and energy efficiency: The device high performance and is power efficient.

SATA BRIDGE IC

SATA Bridge IC is backward compatible with USB 2.0.

The bridge has support for data rate of up to 5Gbps.

The also provides with commands that does filtering. This enables it to drops writing commands and allows reading commands. In this case the no data is contaminated by modification of information contained in the external HDD.