Flash Memory on Firmware as Block Device

# Summary

In this project, the USB API is used to write and read into the flash memory of the LPC2148 microcontroller. First, the LPC2148 is converted into a USB Mass Storage device(MSD) by making changes in the device firmware and USB descriptors. Then a loadable USB driver module is built to communicate with the USB MSD. The module uses SCSI commands inside block function to transfer data over the bulk Endpoint. The read and write sector requests are conveyed through these commands.

SCSI commands are sent using wrapper function, i.e Command Block Wrapper(CBW) and the device acknowledges the same by sending back Command Status Wrapper(CSW). Interpretation of the incoming command is implemented on the firmware side, so as to access the flash memory for writing or reading into the sectors and return back the asked data from host . The firmware part includes mscuser.c which implements multiple states such as CBW, Data and CSW and depending on the values of the bulk\_stage. diskimg.c file contains the flash initialization data to write into MBR.

## Firmware

Steps to build:

Firmware files are uploaded in the respective directory on github.

* In order convert LPC to mass storage, Keil uVision software is required to build the files along with the library present in it.
* Flash the generated hex file using flash magic onto the LPC device.
* Disconnect the device and reconnect after 10 seconds.
* Go to Device Manager and verify that the device is recognized as a mass storage device.
* In Linux, the logs can also be used to verify the same. The VID and PID number of device can be extracted from the logs, generated using $ dmesg.

The main source files in the firmware are as follows:

* Usbdesc.c: this c files contains all the descriptors like device, configuration, interface and endpoint.
* Msc.h: all the SCSI commands supported in the device are listed in this header file along with the wrappers of CBW and CSW.
* Mscuser.c: the state diagram following the sequence of :

Get\_CBW

Trigger event from out endpoint,

Stage=Set\_CSW

Extract SCSI command

If READ(10):

Bulk\_Stage=Data\_IN

Go to Memory\_Read()

Perform Write\_endpoint

Stage= DATA\_IN\_LAST

Acknowledge by

Sending CSW

If WRITE(10):

Bulk\_Stage=Data\_OUT

Go to MSC\_MemoryWrite

Perform write on

Flash memory

Trigger event from out endpoint,

Stage=Set\_CSW

Acknowledge by

Sending CSW

* Kernel Space Driver and build Process:

To communicate with the device, driver is written as a loadable kernel block module. It uses URB’s to send the SCSI commands to the device in the command block wrapper.

In order to build the driver, use the makefile uploaded on the github.

$ make all

$ sudo insmod kernel\_driver.ko

Before inserting the above module remove the inbuilt drivers for USB devices:

$ sudo rmmod uas

$ sudo rmmod usb\_storage

Command to read a specific sector:

$ dd if=/dev/sdc bs=512 skip=5 count=1 | hexdump –R

Command to write to specific sector:

$ dd if=/sample.txt of=/dev/sdc skip=5 count=1