Project Log 2

Project Title: USB MicroSD Card Reader
EEE G512 Embedded System Design
October 2018

Submitted to:
Dr. Devesh Samaiya

Submitted by:
Joy Parikh | 2016A3PS0136P
Rutwik Narendra Jain | 2015A3PS0726P

2 LPC2148 USB Bootloader

The LPC2148 USB bootloader performs three steps:

1. The bootloader checks to see if a USB cable has been plugged in. If the LPC2148 detects the presence of a USB cable then it initiates a USB Mass Storage system. This will cause the target board to appear on any computer platform as a removable flash drive. The user can then seamlessly transfer files to the flash drive. In the background, the LPC2148 moves the user's files onto the SD card using the FAT16 file system.
2. The next thing the bootloader does is look for a firmware file (a file named FW.SFE). This file contains the desired operating firmware (in a binary file format) for the LPC2148 microprocessor. If the bootloader finds this file on the FAT16 system then it programs the contents of this file to the flash memory of the LPC2148. In this way, the bootloader acts as a "programmer" for the LPC2148; and we can upgrade the firmware on the LPC2148 simply by loading a new file onto the micro SD card.
3. After performing the first two checks, the bootloader calls the main firmware. The main code should not even know that the bootloader was used and will run normally.

2.1 Details

The USB device class used is MSCD (Mass Storage Class Device). The MSCD presents easy integration with PC's operating systems. This class allows the embedded system's flash memory space be represented as a folder in Windows/Linux, and the user can update the flash with the binary image using drag and drop (eg, using Windows Explorer) [5].

To make the LPC214x appear like a folder (or disk drive), we need a FAT (File Allocation table). The LPC2000 on chip Code Flash will appear as one single entity (file name: firmware.bin).

In order to run this program, the WinARM programming environment is used. The version used is WinARM 20060606. It's an open source IDE, similar to Keil. Given that it was a text editor, programmer and compiler before more advanced IDEs like Keil came along, WinARM was later found to be incompatible with Windows 10 OS. All MakeFiles to generate main.hex were executed on a laptop running Windows7.

LPC2148 USB Bootloader Source Code (from [4]) contains a whole bunch of source code, including two subdirectories: LCPUSB and System, along with the startup assembly code (crt.S) and the linker file (lpc2138.cmd).

The USB interface on the UM10139 is reproduced in figure 1 and the connections to the SD Card SPI pins is shown in figure 2.

We use the SPI0 interface of the LPC to connect to the microSD card.

The USB Bootloader code, especially portions such as the USB driver firmware is predominantly drawn from

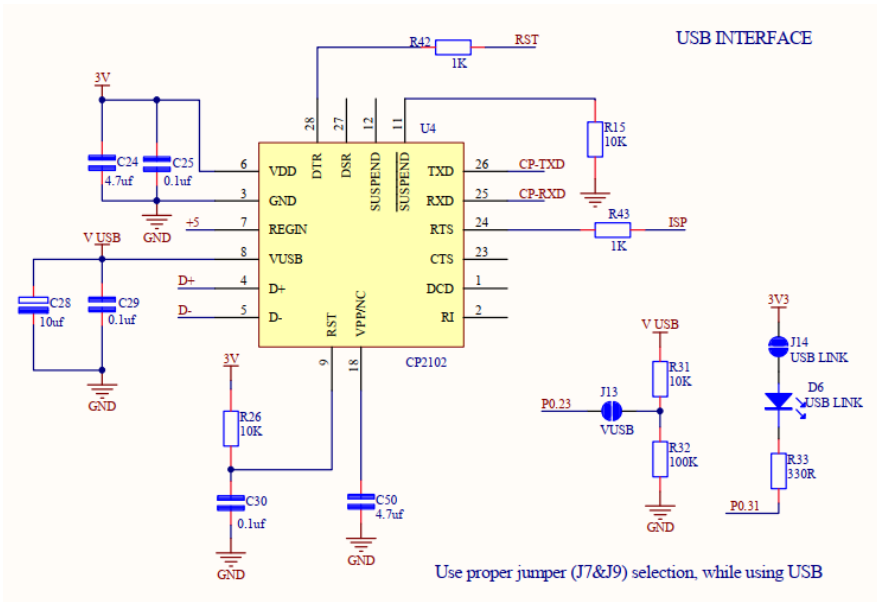


Figure 1: USB interface schematic of UM10139

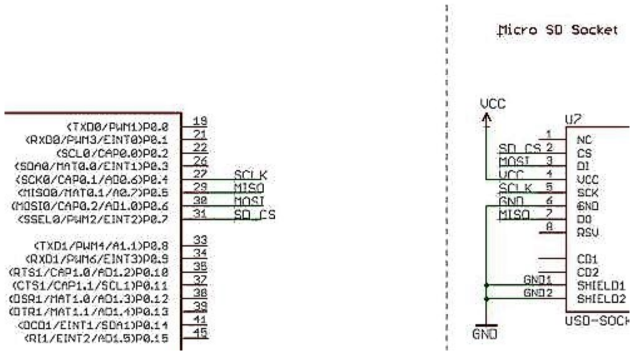
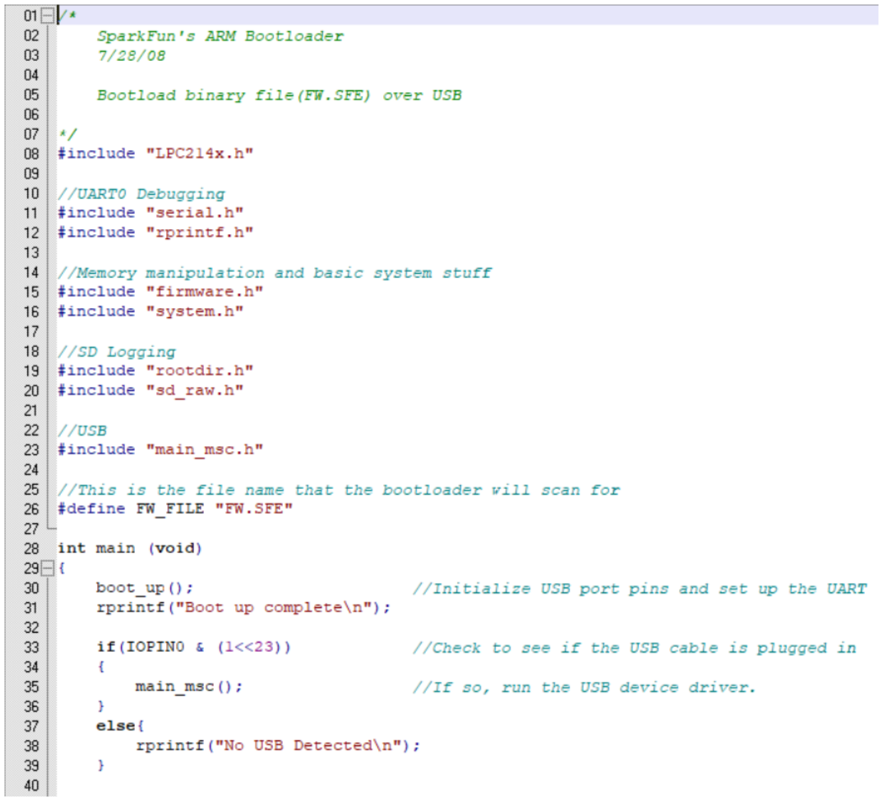


Figure 2: MicroSD Card Interface Connections with LPC2148

The libraries included in the bootloader are: “LPC214x.h,” the usual library needed for the register names and pin assignments and so on of the LPC2148.“Serial.h” and “rprintf.h” are for writing messages out through the serial port, mainly these are used for debugging and user notification. Next we have “firmware.h” and “system.h.” “System.h” contains functions used for initializing and resetting the ARM. “Firmware.h” is one of the key components of the bootloader. This library contains functions that will copy the code from the “FW.SFE” file from the SD card into the program memory of the ARM.

The next two libraries that are included are “rootdir.h” and “sd_raw.h,” both of these files are used for manipulating the FAT16 memory located on the SD card. Finally the library named “main_msc.h” is included; this file is the top level file for the USB device driver.

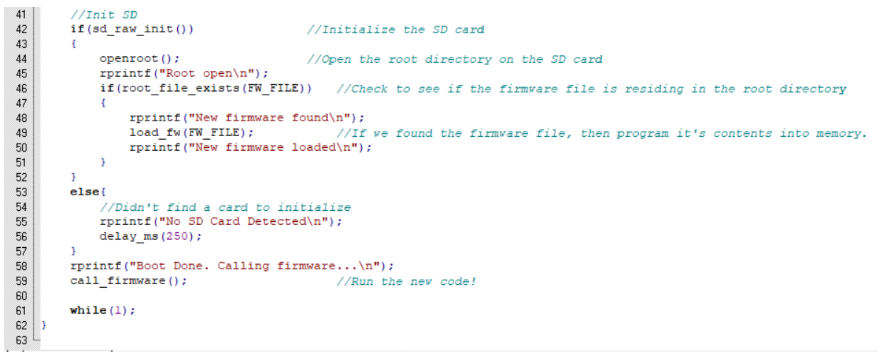


```
/*
    SparkFun's ARM Bootloader
    7/28/08

    Bootload binary file(FW.SFE) over USB

*/
#include "LPC214x.h"

//UART0 Debugging
#include "serial.h"
#include "rprintf.h"

//Memory manipulation and basic system stuff
#include "firmware.h"
#include "system.h"

//SD Logging
#include "rootdir.h"
#include "sd_raw.h"

//USB
#include "main_msc.h"

//This is the file name that the bootloader will scan for
#define FW_FILE "FW.SFE"

int main (void)
{
    boot_up();                      //Initialize USB port pins and set up the UART
    rprintf("Boot up complete\n");

    if(IOPIN0 & (1<<23))            //Check to see if the USB cable is plugged in
    {
        main_msc();                 //If so, run the USB device driver.
    }
    else{
        rprintf("No USB Detected\n");
    }

```

Figure 3: main.c

The first function in main.c, boot_up(), just initializes the LPC2148 pins and sets up the UART (for debugging). If there is a high voltage on pin 23, connected to the USB signal Vbus, then the USB cable is plugged in. If this is the case, then the function main_msc() is run. Main_msc() is the USB device driver. Once the USB cable is unplugged, the main_msc() function exits and the rest of the bootloader code resumes.

After checking for the USB cable, the firmware initializes the SD card (figure 4). Once the card is initialized, the root directory of the FAT16 structure is opened, a file named “FW.SFE” is searched for, and the binary firmware file contents are copied to Flash. Once the file has been copied into Flash, the bootloader is done. The last thing it does is call the function “call_fw.”

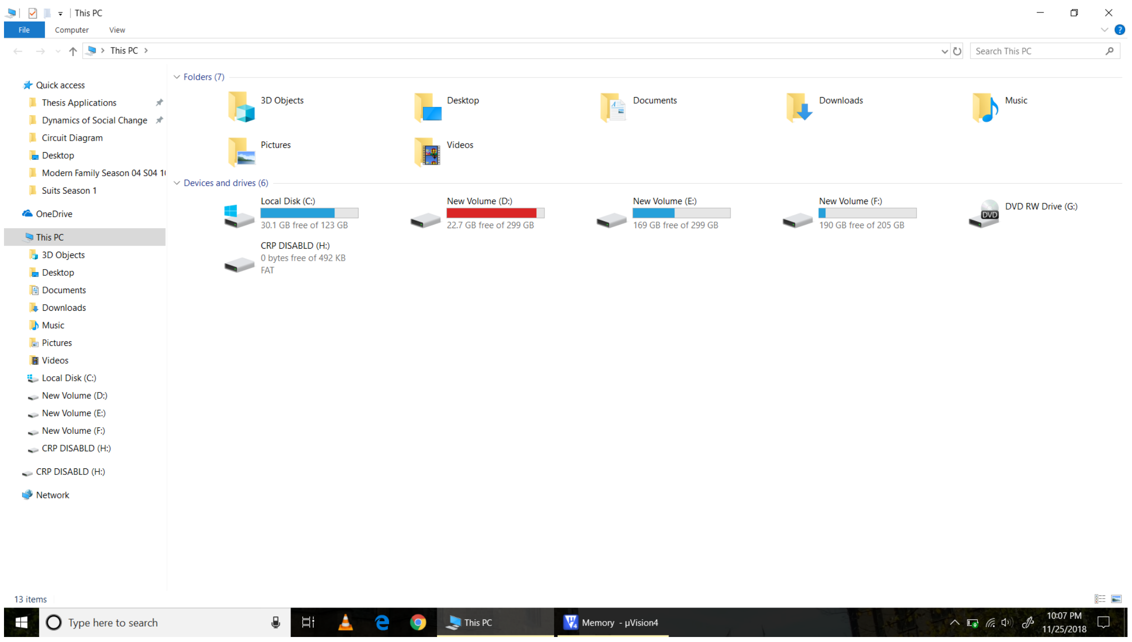


```c
    //Init SD
    if(sd_raw_init())               //Initialize the SD card
    {
        openroot();                 //Open the root directory on the SD card
        rprintf("Root open\n");
        if(root_file_exists(FW_FILE))   //Check to see if the firmware file is residing in the root directory
        {
            rprintf("New firmware found\n");
            load_fw(FW_FILE);           //If we found the firmware file, then program it's contents into memory.
            rprintf("New firmware loaded\n");
        }
    }
    else{
        //Didn't find a card to initialize
        rprintf("No SD Card Detected\n");
        delay_ms(250);
    }
    rprintf("Boot Done. Calling firmware...\n");
    call_firmware();                    //Run the new code!

    while(1);
}
```

Figure 4: main.c (Contd.)

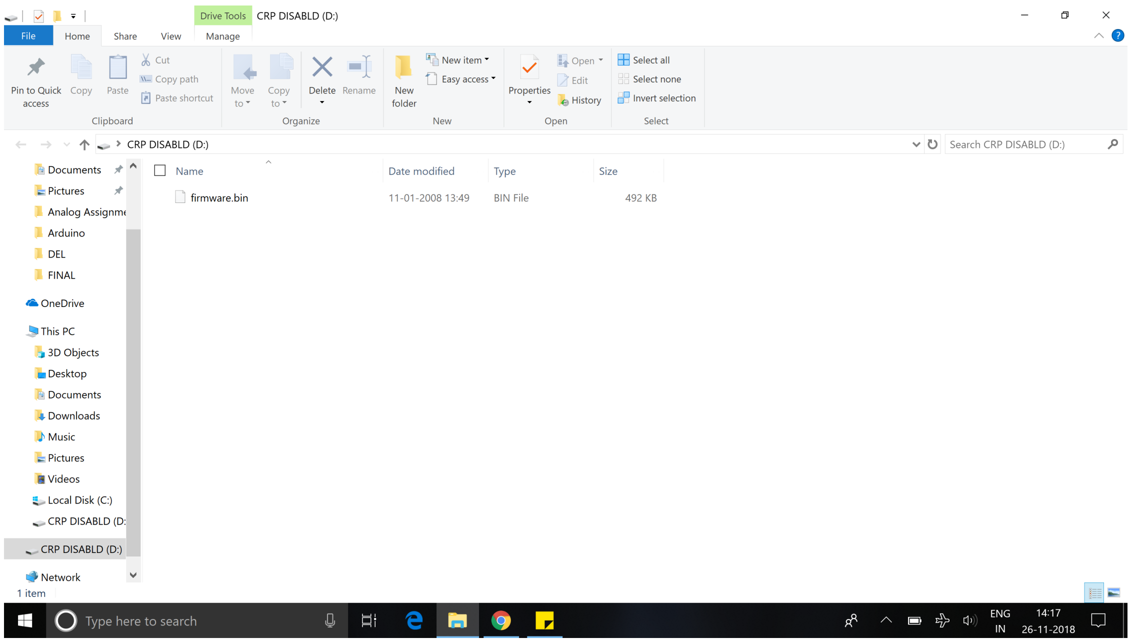
2.2 Log Update

- WinARM works with Windows 7 and earlier versions.
- SD Card formatting may be required for the bootloader to work.
- LPC2148 USB Bootloader Source Code Folder is a comprehensive directory containing various libraries and functions critical to bootloader.
- sd_raw.c contains functions for sending and receiving commands as well as bytes of data to/from the SD card.
- fat16.c elucidates the FAT table structure, headers etc. The application note however works with FAT12.
- main_msc.c has all USB mass storage descriptors and handles all mass storage class requests.
- A combination of programs/libraries from the USBMem demo of AN10711 and the SparkFun tutorial (drawn from Bertik Sikken), successfully works. Figures below show the LPC recognized as a Mass Storage device in an Explorer window.



Drive	Space
Local Disk (C:)	30.1 GB free of 123 GB
New Volume (D:)	22.7 GB free of 299 GB
New Volume (E:)	169 GB free of 299 GB
New Volume (F:)	190 GB free of 205 GB
DVD RW Drive (G:)	
CRP DISABLD (H:)	0 bytes free of 492 KB FAT

Figure 5: Windows Explorer window showing USB Mass Storage Device



Name	Date modified	Type	Size
firmware.bin	11-01-2008 13:49	BIN File	492 KB

Figure 6: firmware.bin

References

[1] ARM Keil Documentation, USBMem- Mass Storage Device,
Available online: http://www.keil.com/support/man/docs/mcb2140

[2] UM10139 LPC214x Manual, Rev. 4, NXP Semiconductors, 23 April 2012.

[3] SDCard Interfacing with LPC2148 using SPI Module - WikiNote

[4] SparkFun LPC2148 Bootloader Tutorial https://www.sparkfun.com/tutorials/94

[5] AN10711 USB secondary ISP bootloader: Application Note, Rev. 02, 15 July 2008.