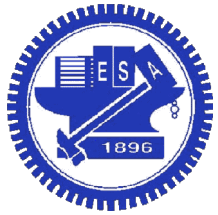
Lab 7: SD Card Reader Circuit



National Chiao Tung University
Chun-Jen Tsai
11/16/2018

Lab 7: SD Card Reader Circuit

- In this lab, you will design a circuit to read a text file from an SD card, and count the number of three-letter words in the file.
  - Example of three-letter words: “the”, “and”, “all”, …
  - The number of the three-letter words will be displayed on the 1602 LCD screen
- The deadline of the lab is on 11/27

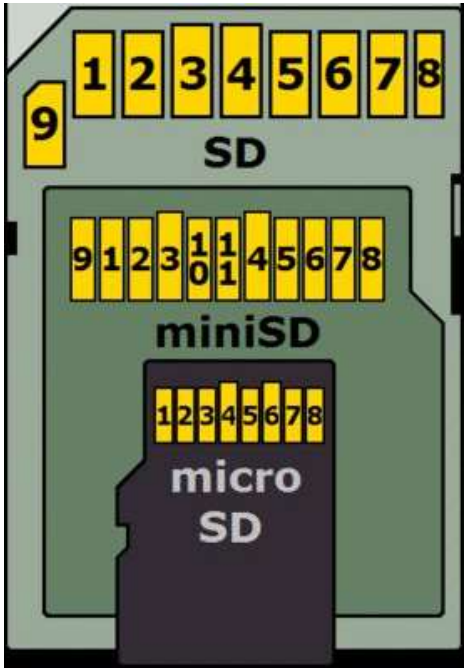
SD Card Specification

- The SD card that we use follows the SDHC standard, formatted with the FAT32 file system
- The logical structure is composed of 512-byte blocks, starting at block number 0
  - An 8GB SD card will be used in the lab
- SD cards support at least two different I/O interfaces. In this lab, we use the serial SPI interface to read data

SD Card I/O Interface

- An SDHC card has three different operation modes:
  - SPI mode
  - One-bit SD bus mode
  - Four-bit SD bus mode

SD Pin	Name	SPI Mode Function
1	nCS	SPI Card Select [CS] (Negative logic)
2	DI	SPI Serial Data In [MOSI]
3	VSS	Ground
4	VDD	Power
5	CLK	SPI Serial Clock [SCLK]
6	VSS	Ground
7	DO	SPI Serial Data Out [MISO]
8	NC	Unused
9	NC	Unused
10	NC	Reserved
11	NC	Reserved

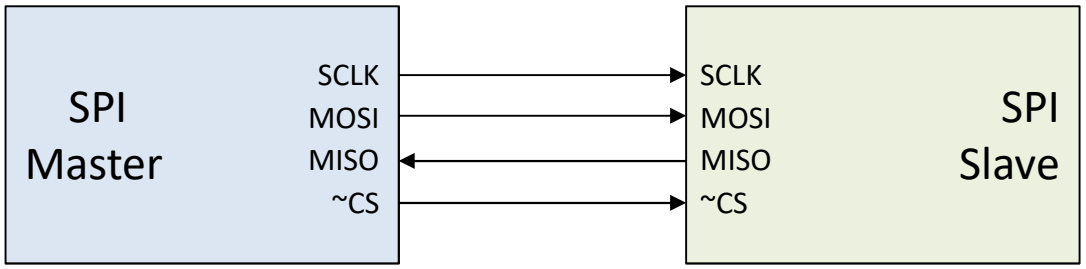


SD Card Initialization

- During the initialization phase, the SD card controller negotiates with the card to determine which type of card is used: MMC, SD, SDHC, SDXC, …, etc.
  - The controller uses a slower clock (500kHz) to talk to the SD card during the negotiation phase
- Once the card is initialized, the SD card controller can use a faster clock (e.g., the system clock) for read/write operations, as long as the card can handle the speed

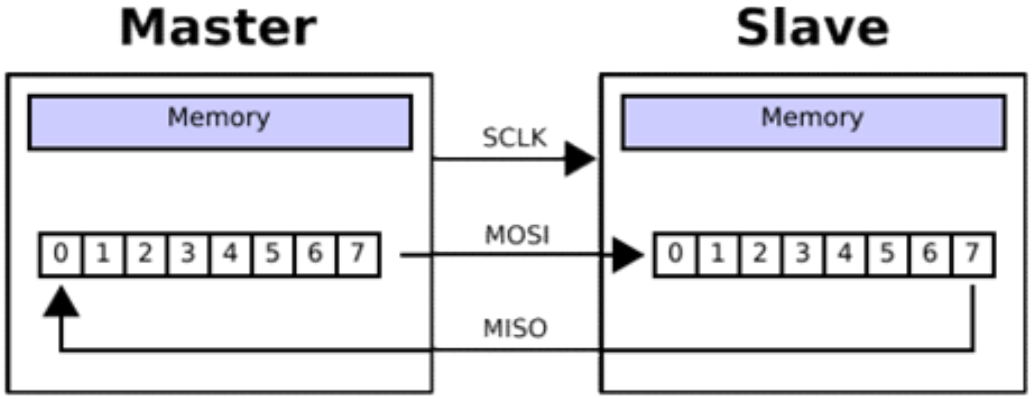
Serial Peripheral Interconnect (SPI)

- SPI is introduced by Motorola in 1980's for their MCU
  - Short-distance synchronous serial communications for SD cards, LCDs screens, audio codecs, boot flash, etc.
  - A four-wire, full-duplex, master-slave serial bus
  - One master, multiple slaves
  - Open-loop transmission, no slave acknowledgement protocol



SPI Data Communication

- The master selects the target slave via the CS pin first, then sends the clock signal to the slave
- The master and slave exchange data one bit per clock cycle using shift registers
  - Data sizes can be of 8-, 12-, or 16-bit, depending on the device
  - The data sampling clock edge (rising or falling) also depends on the device → read data sheet of the device!

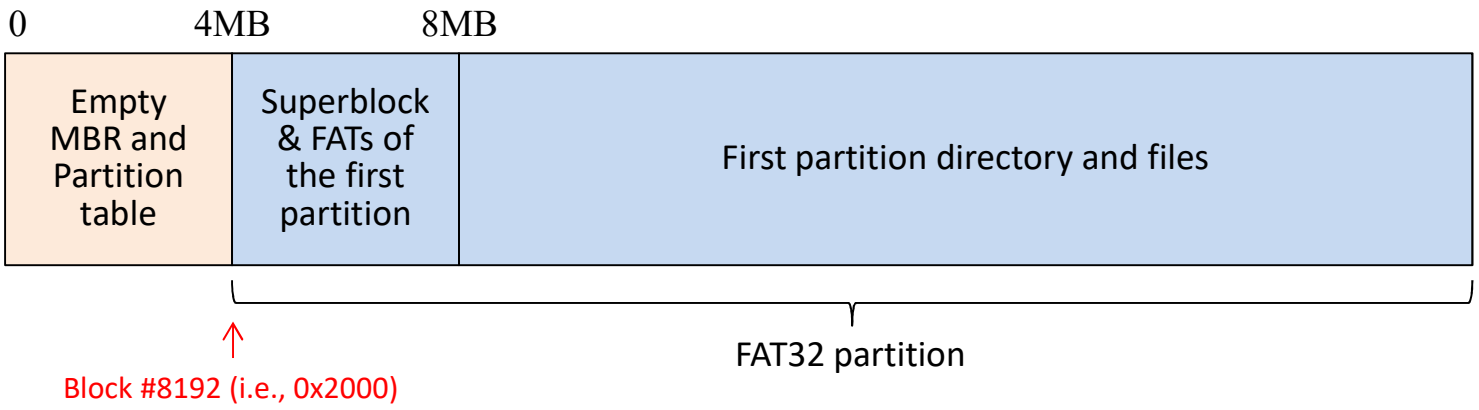


Physical Structure and File Systems

- The logical structure of an SD card is simply composed of a sequence of 512-byte blocks
  - Physically, SD cards are divided into 4KB ~ 32KB sectors
- To create directories for file storage on the card, we must first partition the SD card and then format a logical file system on that partition
- An SD card usually has one partition. However, it is possible to store multiple partitions and multiple file systems on a single SD card

Disk Partitions

- A physical disk can have several disk partitions, each partition can be formatted to a file system
- Typical partition structure of an SDHC card:



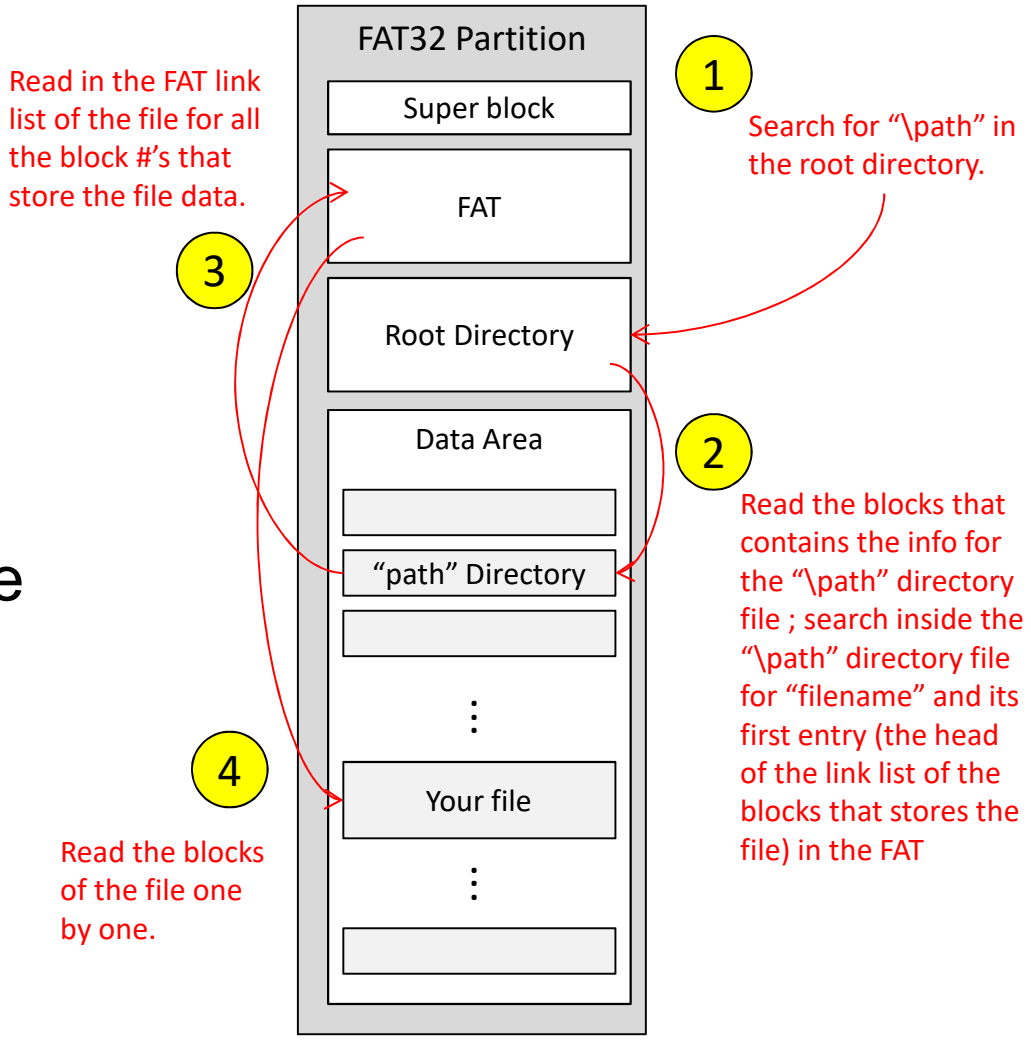
- If the card is bootable or has more than one partitions, then the Master Boot Record (MBR) and the partition table will not be empty

FAT32 Structure

- The FAT32 file system is a standard by Microsoft, and popularly used for mass storage devices
- An FAT32 file system has the following components:
  - Boot sector: 512 bytes, possibly block#0, a.k.a. the super block
    - The boot sector contains information such as the size of the FAT, root directories, boot code, etc.
  - File allocation table (FAT): a table that shows which file is stored in which allocation units (each allocation unit is composed of several consecutive blocks); FAT basically contains many link lists of block numbers (one list per file)
  - Root directory: contains file names, file attributes, and the first allocation unit of all files in the root directory
  - Data area: the data blocks that actually store files

File Structure of an FAT32 Partition

- To read a file of "\path\name" in an FAT32 file system, one shall follow the four steps shown in the figure.
- However, it is possible to read a small file without going through these steps!

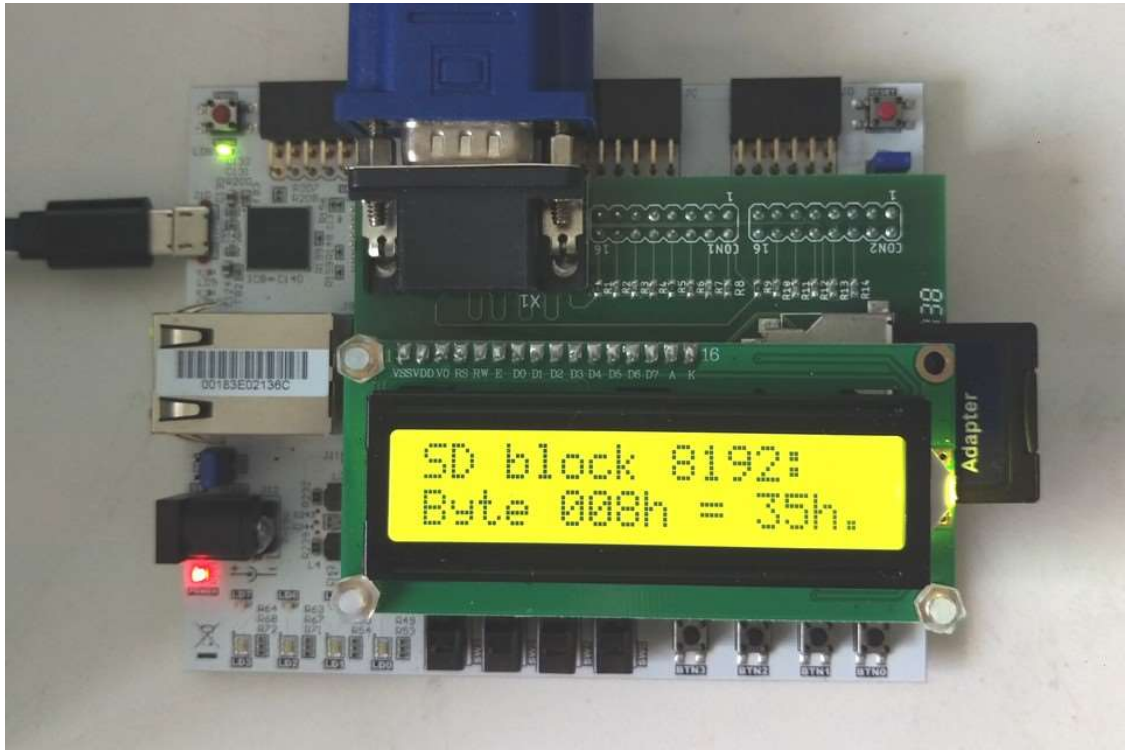


The Sample Project Files of Lab 7

- The sample project files of Lab 7 has a top-level circuit, `lab7.v`, an SD card controller, `sd_card.v`, and other supporting files such as: `debounce.v`, `LCD_module.v`, `clk_divider.v`, and `lab7.xdc`
- The circuit performs the following actions:
  - When the board is powered up, the SD card controller will initialize itself and enter a ready state
  - When the user presses BTN2, the circuit reads a block of data from the SD card, and then print the first byte in the block on the LCD module
  - Every time BTN2 is pressed, the next byte will be displayed

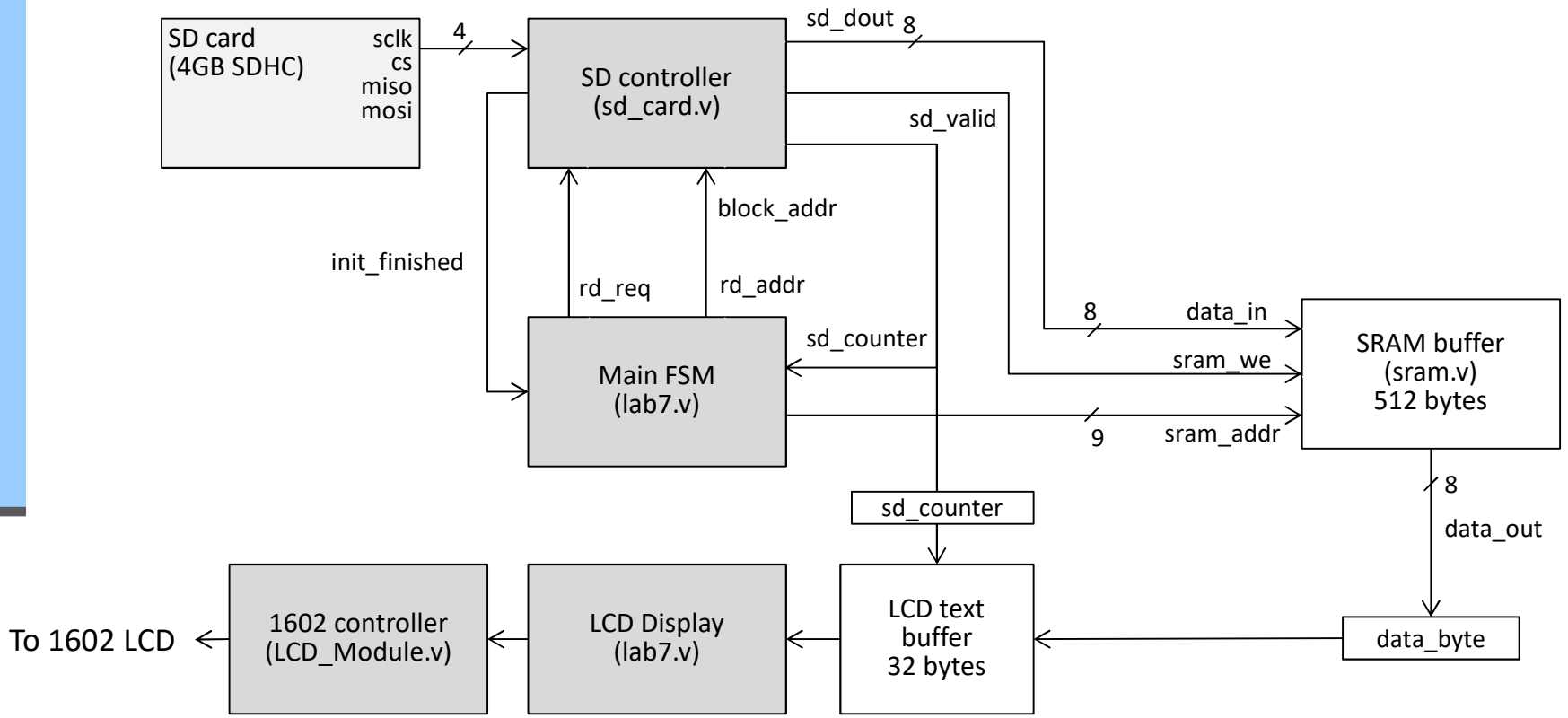
The Output of Lab 7 Sample Circuit

- For example, the following message is shown after 9 button hits (different card may show different result):



System Diagram of the Sample Code

- The block diagram of the sample code of Lab 7:



SD Controller Signals

- The key signals that control SD card operations:
  - rd_req: Triggers the reading of a block
  - block_addr[31:0]: The block # of the SD card to read
  - init_finished: SD card initialization is finished?
  - dout[7:0]: Output one byte of data in the block per clock cycle.
  - sd_valid: The output byte in dout[7:0] is ready

- If you set rd_req to 1 for one clock cycle, the SD card controller will read the data of the target block one byte at a time. Each time a data byte (of the 512 bytes) is ready, the flag sd_valid raise to 1 for one clock cycle.

What You Need to Do for Lab 7

- You must design a circuit that:
  - When BTN2 is pressed the circuit reads the SD card and search for an ASCII file, “test.txt”
  - You can download “test.txt” from E3 and save it on the SD cards in the lab if it’s not there already
  - The file begins with the word “DLAB_TAG” and ends with the word “DLAB_END”; each word is separated by punctuation marks or line feeds from the next word
  - The circuit reads through the text file and counts the number of three-letter words in the text file, and print the number to the 1602 LCD module using 4-digit hexadecimal format as follows:

```
Found 007B words
in the text file
```

The Format of the Input Text File

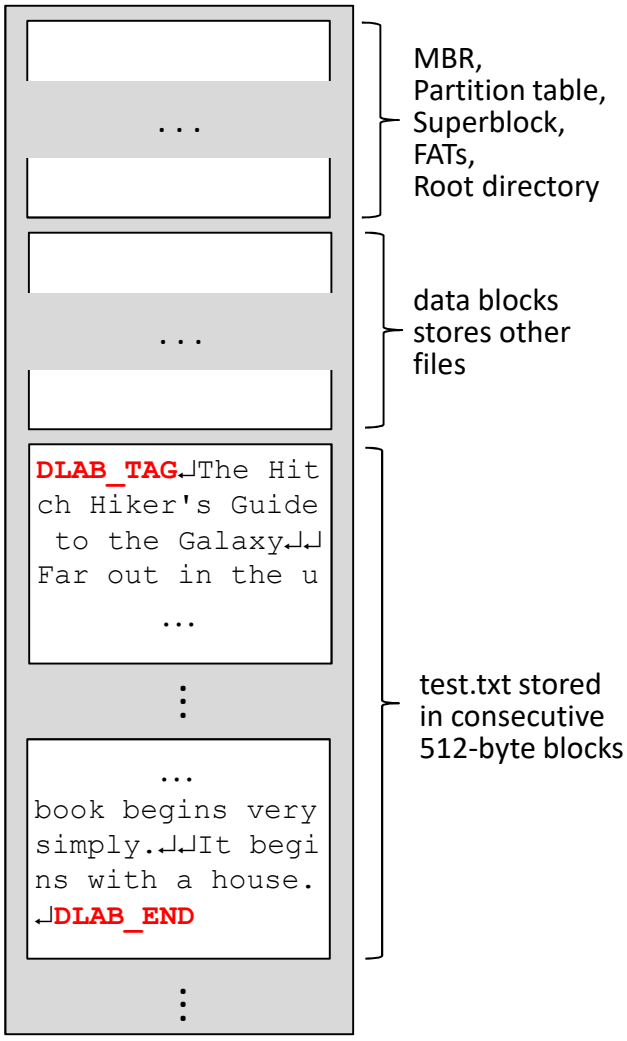
- The sample input text file, test.txt, is as follows.

```
DLAB_TAG↵
The Hitch Hiker's Guide to the Galaxy↵
↵
Far out in the uncharted backwaters of the unfashionable  end  of
the  western  spiral  arm  of  the Galaxy lies a small unregarded
yellow sun.↵

                                . . . . .


But the story of this terrible, stupid Thursday, the story of its
extraordinary  consequences,  and  the  story  of  how  these
consequences are inextricably intertwined  with  this  remarkable
book begins very simply.↵
↵
It begins with a house.↵
DLAB_END↵
```

Note: ↵ stands for 0x0A.

The layout of the File on the SD card

- A newly formatted SD card will have its initial files stored in consecutive 512-byte blocks
- After some files are deleted, new files added may be stored in non-contiguous blocks
- You can assume that test.txt is stored in consecutive blocks



Some Comments

- Although the sample code of Lab 7 use a SRAM to buffer the data from the SD card, you do not have to use an SRAM buffer for this lab
  - It is possible to design a circuit to count the number of three-letter words on-the-fly without using any SRAM buffer
- Note that the signal ‘sd_valid’ may not always be 1 during the reading of a block of 512 bytes
- A three-letter word may lay across two sectors