Lab 8: SD Card Reader Circuit

Chun-Jen Tsai and Lan-Da Van

Department of Computer Science

National Yang Ming Chiao Tung University

Taiwan, R.O.C.

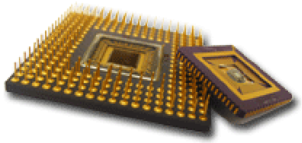
Fall, 2024

Lab 8: SD Card Reader Circuit

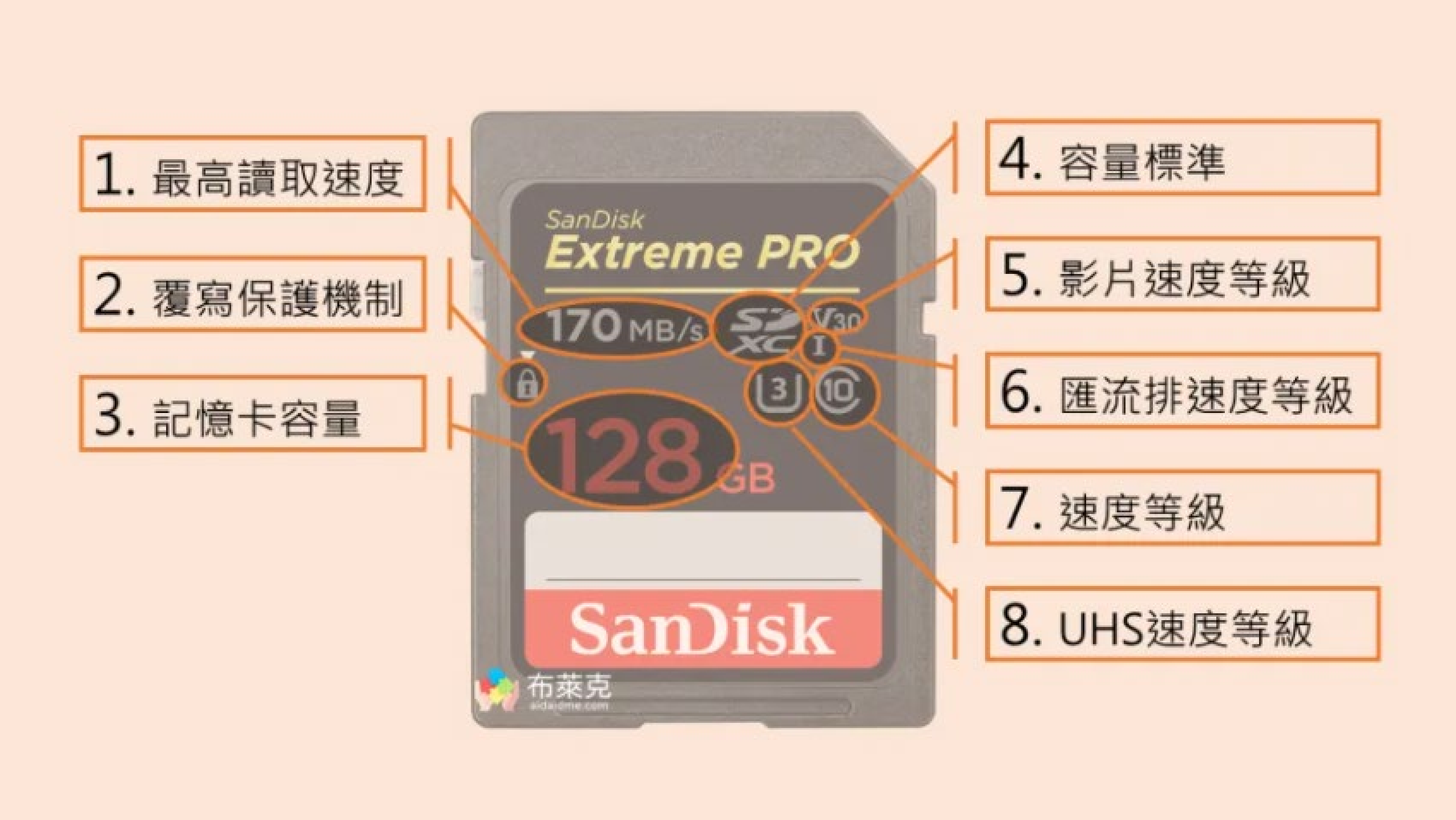
- In this lab, you will design a circuit to read a text file from an SD card, and using RGB LED to display it.
  - Also show some information in LCD display.

- The lab file submission deadline is on 11/11 by 6:00pm.

- Warning: Please notify TA if you have photosensitive epilepsy. We will assist you in making arrangements to change your lab session.

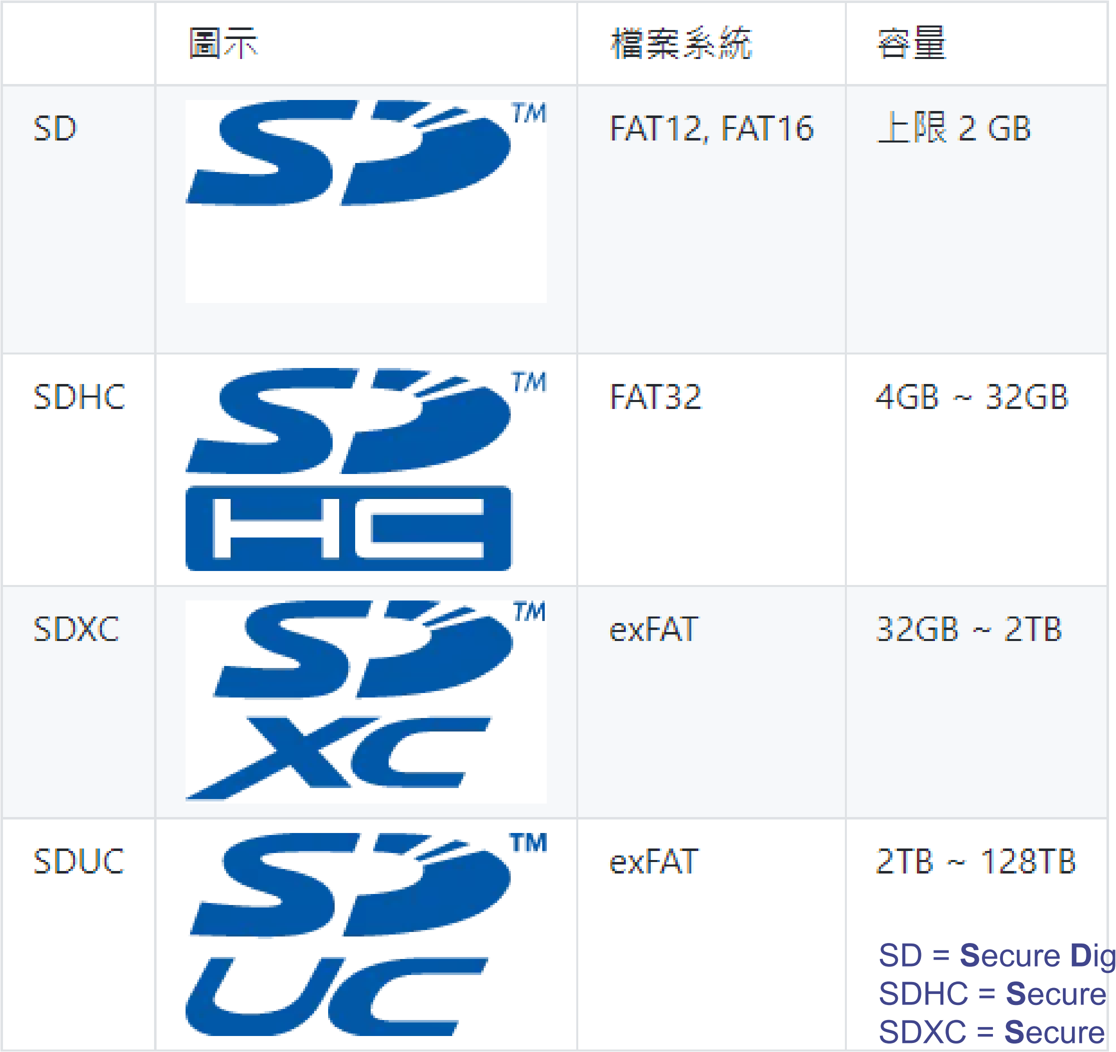


SD Card Specification (1/4)



https://aidaidme.com/how-to-choose-sd-card/

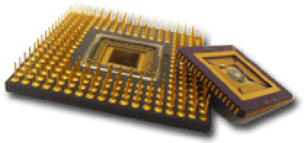
SD Card Specification (2/4)



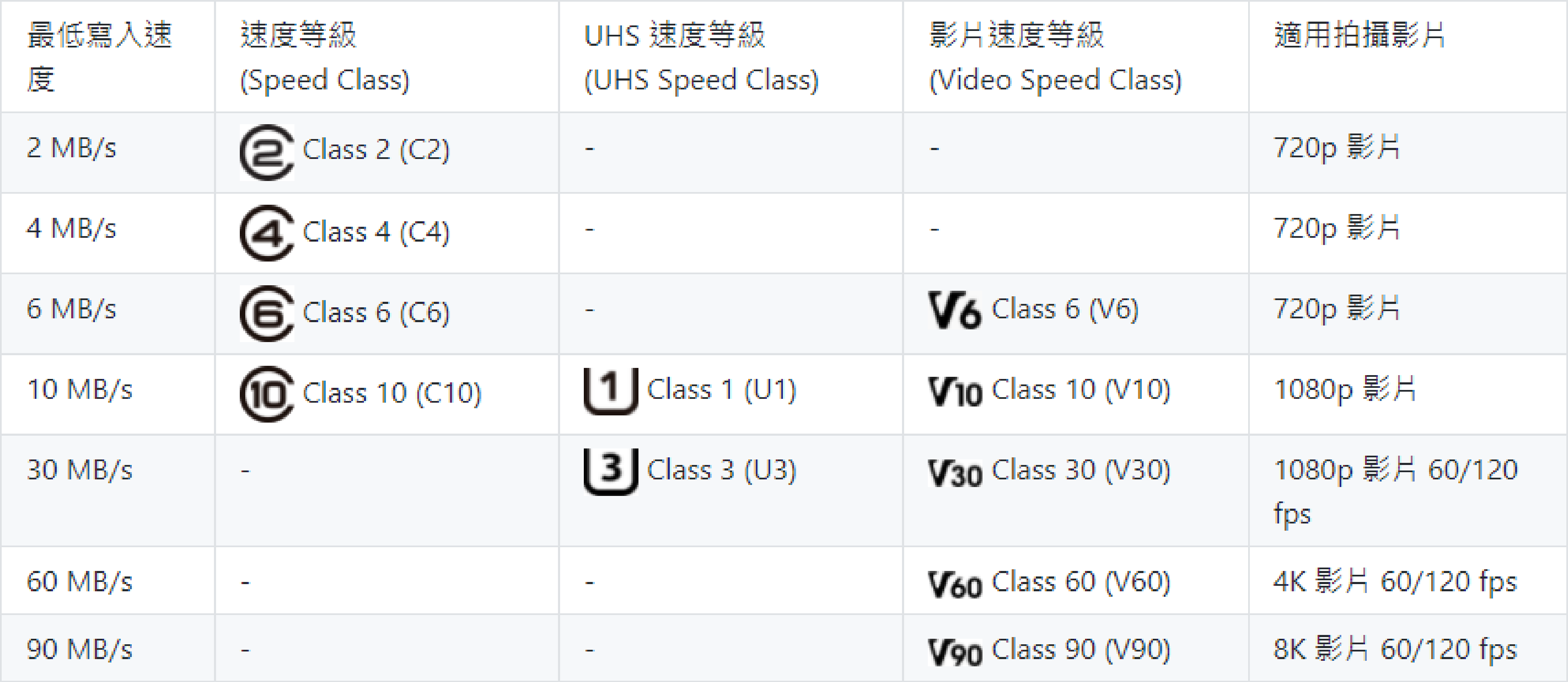
	圖示	檔案系統	容量
SD	SD™	FAT12, FAT16	上限 2 GB
SDHC	SD HC™	FAT32	4GB ~ 32GB
SDXC	SD XC™	exFAT	32GB ~ 2TB
SDUC	SD UC™	exFAT	2TB ~ 128TB

SD = **S**ecure **D**igital
SDHC = **S**ecure **D**igital **H**igh **C**apacity
SDXC = **S**ecure **D**igital e**X**tended **C**apacity
SDUC = **S**ecure **D**igital **U**ltra **C**apacity

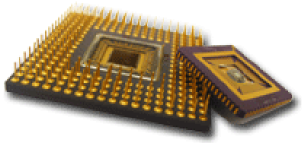
https://aidaidme.com/how-to-choose-sd-card/



SD Card Specification (3/4)



最低寫入速度	速度等級 (Speed Class)	UHS 速度等級 (UHS Speed Class)	影片速度等級 (Video Speed Class)	適用拍攝影片
2 MB/s	Class 2 (C2)	-	-	720p 影片
4 MB/s	Class 4 (C4)	-	-	720p 影片
6 MB/s	Class 6 (C6)	-	Class 6 (V6)	720p 影片
10 MB/s	Class 10 (C10)	Class 1 (U1)	Class 10 (V10)	1080p 影片
30 MB/s	-	Class 3 (U3)	Class 30 (V30)	1080p 影片 60/120 fps
60 MB/s	-	-	Class 60 (V60)	4K 影片 60/120 fps
90 MB/s	-	-	Class 90 (V90)	8K 影片 60/120 fps



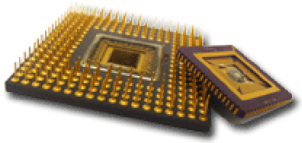
https://aidaidme.com/how-to-choose-sd-card/

SD Card Specification (4/4)

- The SD card that we use follows the secure digital high capacity (SDHC) standard, formatted with the FAT32 file system.

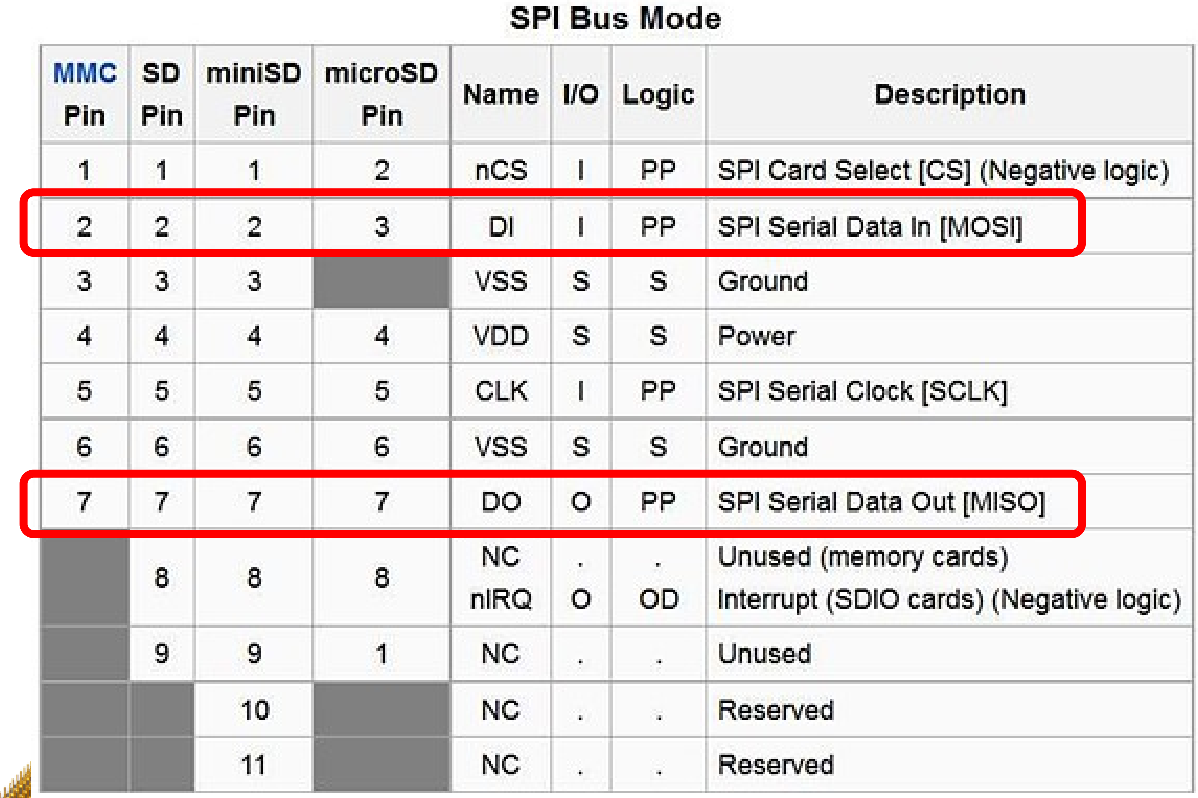
- The logical structure is composed of 512-byte blocks, starting at block number 0.
  - An 8GB SD card will be used in the lab.

- SD cards support at least two different I/O interfaces. In this lab, we use the serial Serial Peripheral Interconnect (SPI) interface to read data.



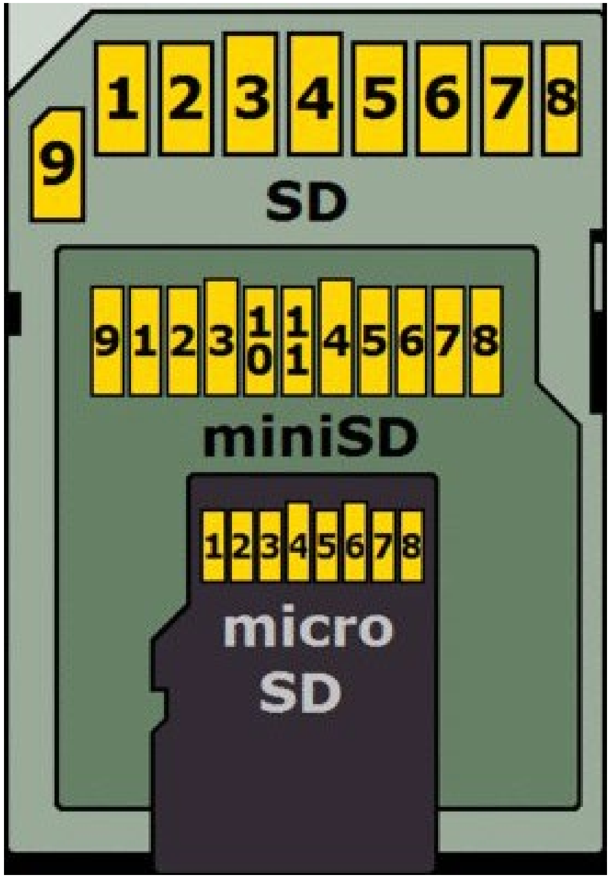
SD Card I/O Interface (1/2)

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



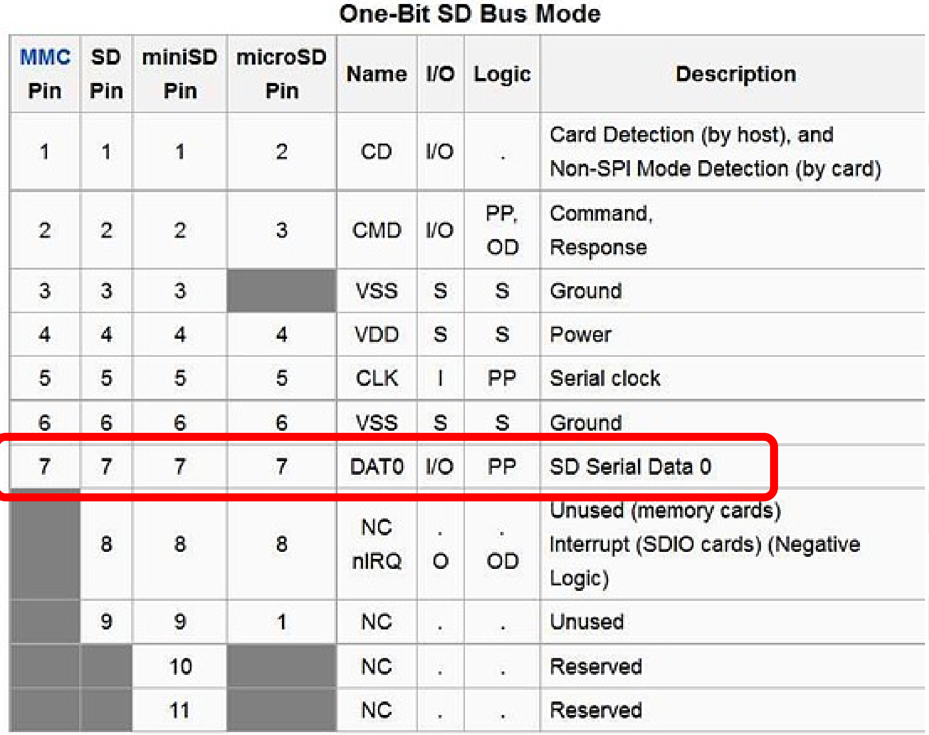
SPI Bus Mode

MMC Pin	SD Pin	miniSD Pin	microSD Pin	Name	I/O	Logic	Description
1	1	1	2	nCS	I	PP	SPI Card Select [CS] (Negative logic)
2	2	2	3	DI	I	PP	SPI Serial Data In [MOSI]
3	3	3		VSS	S	S	Ground
4	4	4	4	VDD	S	S	Power
5	5	5	5	CLK	I	PP	SPI Serial Clock [SCLK]
6	6	6	6	VSS	S	S	Ground
7	7	7	7	DO	O	PP	SPI Serial Data Out [MISO]
	8	8	8	NC nIRQ	. O	. OD	Unused (memory cards) Interrupt (SDIO cards) (Negative logic)
	9	9	1	NC	.	.	Unused
		10		NC	.	.	Reserved
		11		NC	.	.	Reserved



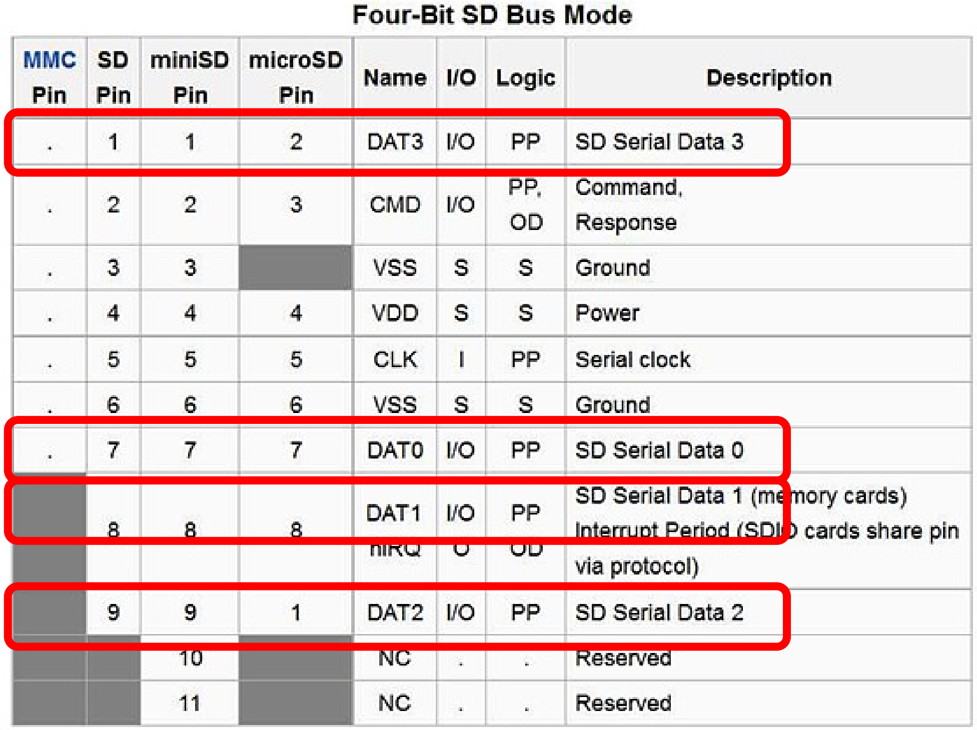
https://www.dpreview.com/forums/post/55490046

SD Card I/O Interface (2/2)



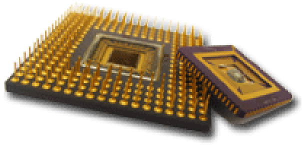
One-Bit SD Bus Mode

MMC Pin	SD Pin	miniSD Pin	microSD Pin	Name	I/O	Logic	Description
1	1	1	2	CD	I/O	.	Card Detection (by host), and Non-SPI Mode Detection (by card)
2	2	2	3	CMD	I/O	PP, OD	Command, Response
3	3	3		VSS	S	S	Ground
4	4	4	4	VDD	S	S	Power
5	5	5	5	CLK	I	PP	Serial clock
6	6	6	6	VSS	S	S	Ground
7	7	7	7	DAT0	I/O	PP	SD Serial Data 0
	8	8	8	NC nIRQ	. O	. OD	Unused (memory cards) Interrupt (SDIO cards) (Negative Logic)
	9	9	1	NC	.	.	Unused
		10		NC	.	.	Reserved
		11		NC	.	.	Reserved



Four-Bit SD Bus Mode

MMC Pin	SD Pin	miniSD Pin	microSD Pin	Name	I/O	Logic	Description
.	1	1	2	DAT3	I/O	PP	SD Serial Data 3
.	2	2	3	CMD	I/O	PP, OD	Command, Response
.	3	3		VSS	S	S	Ground
.	4	4	4	VDD	S	S	Power
.	5	5	5	CLK	I	PP	Serial clock
.	6	6	6	VSS	S	S	Ground
.	7	7	7	DAT0	I/O	PP	SD Serial Data 0
	8	8	8	DAT1 nIRQ	I/O O	PP OD	SD Serial Data 1 (memory cards) Interrupt Period (SDIO cards share pin via protocol)
	9	9	1	DAT2	I/O	PP	SD Serial Data 2
		10		NC	.	.	Reserved
		11		NC	.	.	Reserved

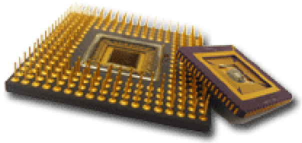


https://www.dpreview.com/forums/post/55490046

SD Card Initialization

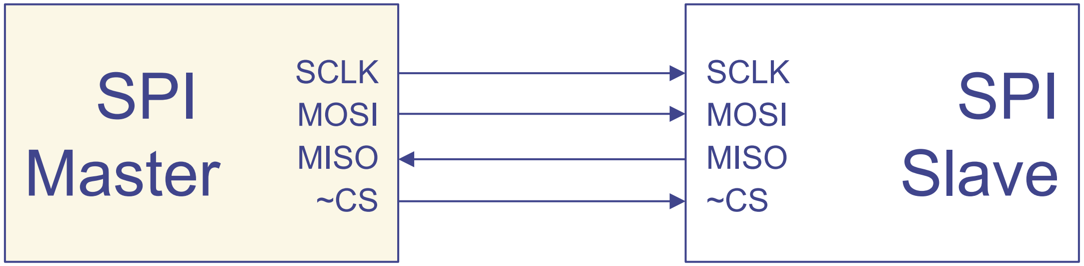
- During the initialization phase, the SD card controller negotiates with the card to determine which type of card is used: SD, SDHC, SDXC, and SDUC.
  - The controller uses a slower clock (500kHz) to talk to the SD card during the negotiation phase.

- 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.



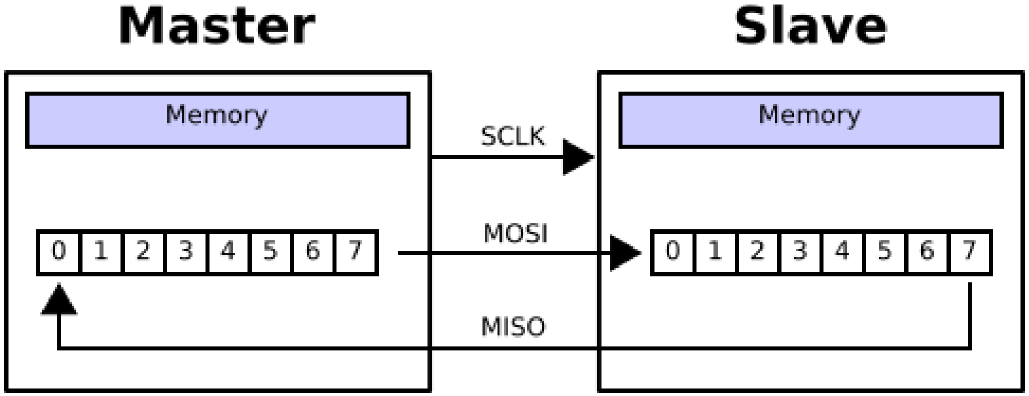
Serial Peripheral Interconnect (SPI)

- 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 (microSD pin 2) 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!

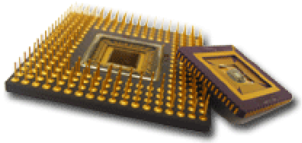


FPGA

SD Card

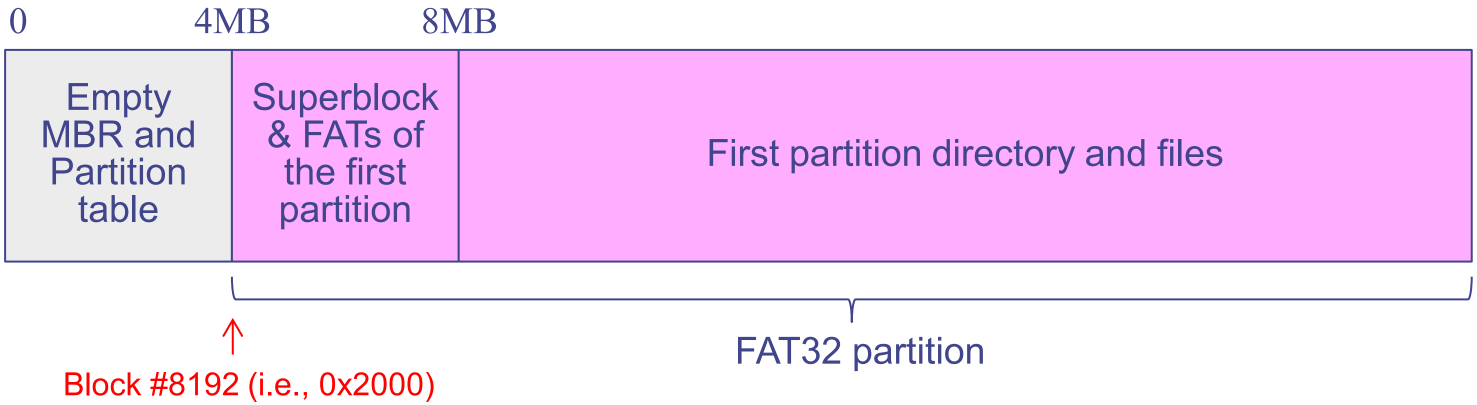
Physical Structure and File Systems

- 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.
  - 1 block = 512 Bytes
  - 1 sector = one or multiple blocks
- 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

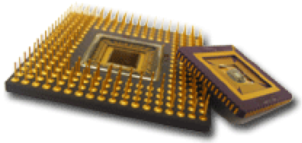
- A physical disk can have several disk partitions, and each partition can be formatted to a file system.
- Typical partition structure of an SDHC card:


0
4MB
8MB
Empty MBR and Partition table
Superblock & FATs of the first partition
First partition directory and files
FAT32 partition
Block #8192 (i.e., 0x2000)

- 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

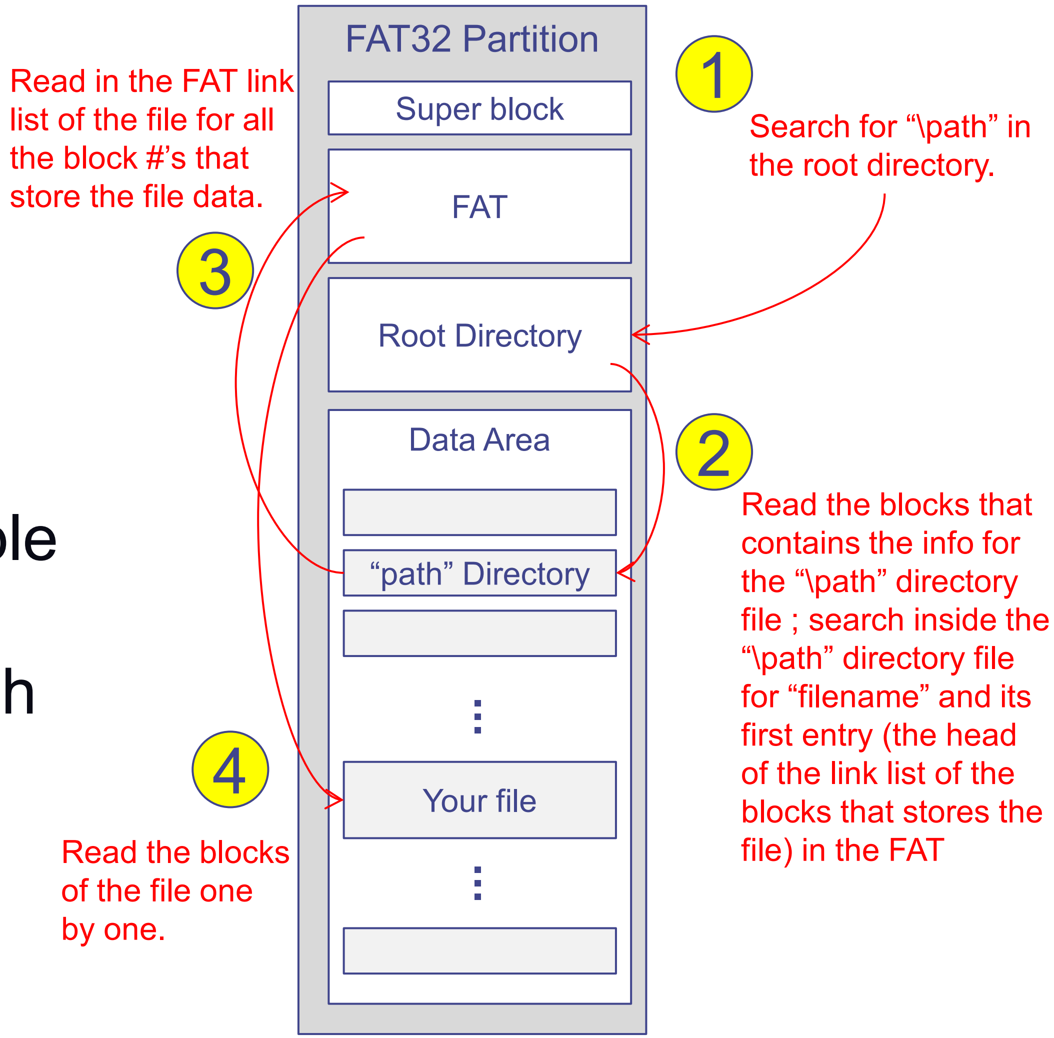
- The FAT32 file system is a standard by Microsoft, and popularly used for mass storage devices.
- An FAT32 file system has the following components:
  - Boot sector: 512 bytes, possibly block#0, a.k.a. the super block
    - The boot sector contains information such as the size of the FAT, root directories, 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!



FAT32 Partition

Super block

FAT

Root Directory

Data Area

"path" Directory

⋮

Your file

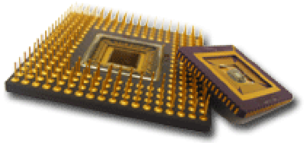
⋮

1 Search for "\path" in the root directory.

2 Read the blocks that contains the info for the "\path" directory file ; search inside the "\path" directory file for "filename" and its first entry (the head of the link list of the blocks that stores the file) in the FAT

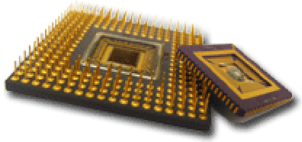
3 Read in the FAT link list of the file for all the block #'s that store the file data.

4 Read the blocks of the file one by one.



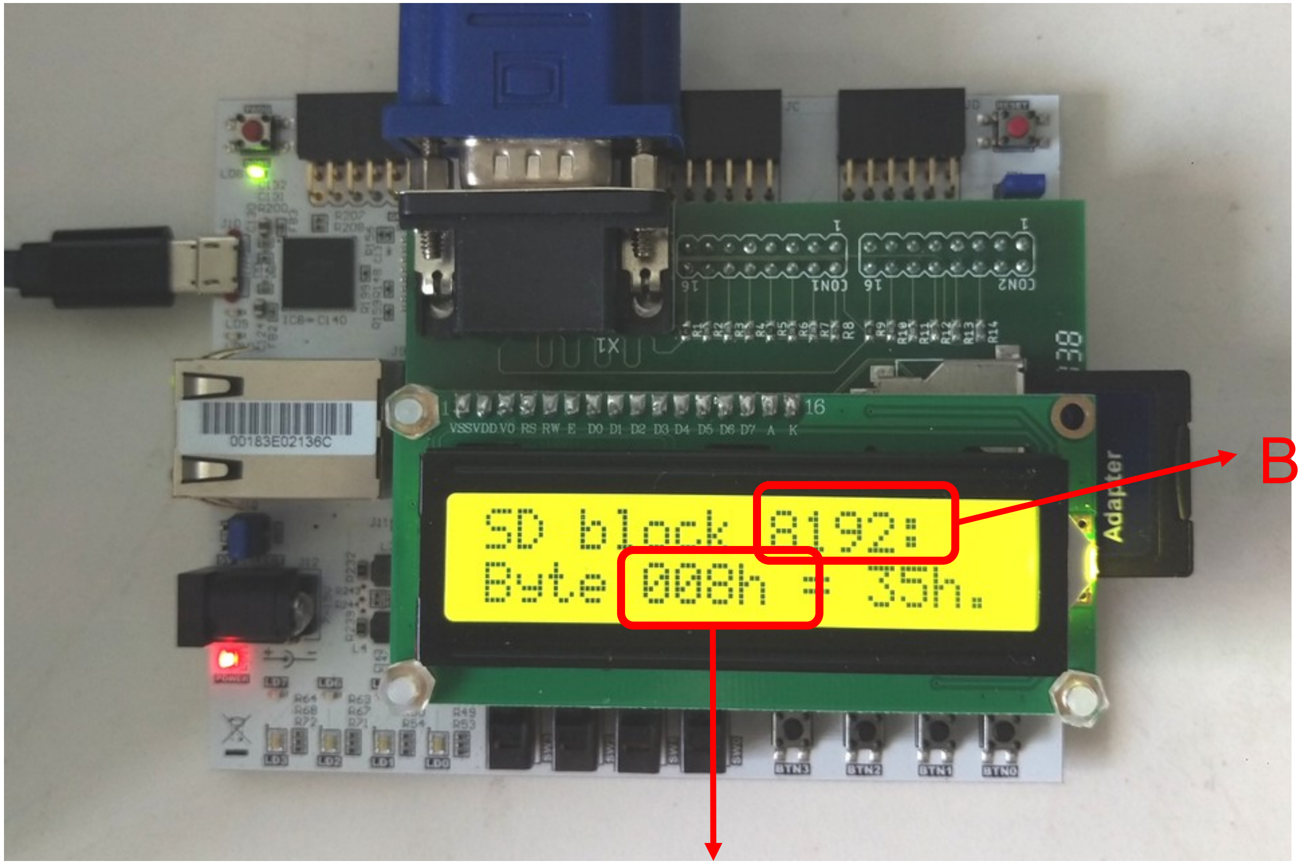
Sample Project Files of Lab 8

- The sample project files of Lab 8 has a top-level circuit, `lab8.v`, an SD card controller, `sd_card.v`, and other supporting files such as: `debounce.v`, `LCD_module.v`, `clk_divider.v`, and `lab8.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.



Output of Lab 8 Sample Circuit

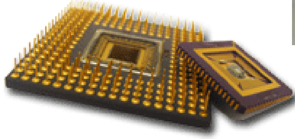
- For example, the following message is shown after 9 button hits (different card may show different result):



```
SD block 8192:
Byte 008h = 35h.
```

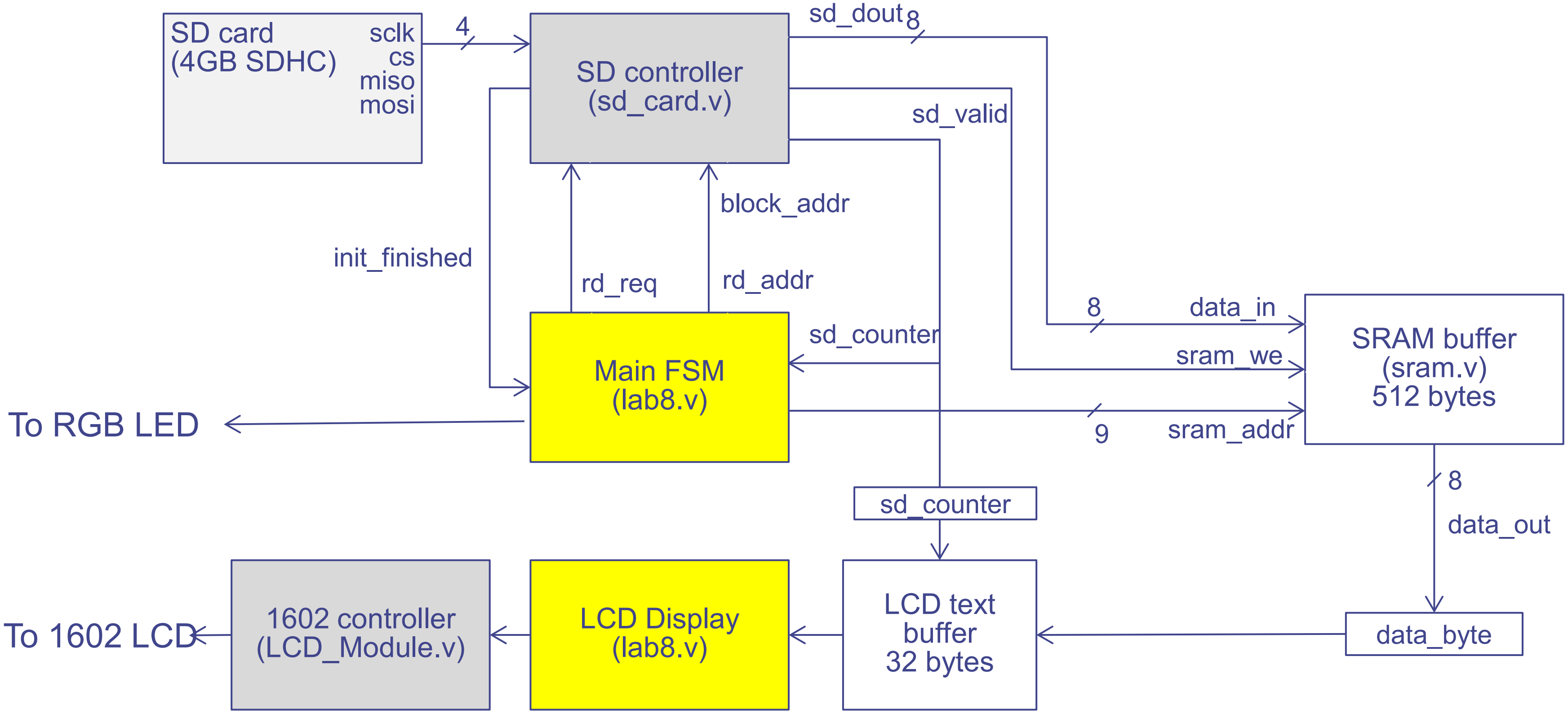
Block index

Address



System Diagram of the Sample Code

- The block diagram of the sample code of Lab 8:

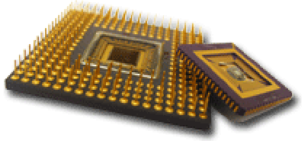


SD Controller Signals

- 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

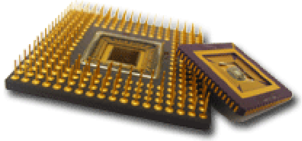
- If you set rd_req to 1 for one clock cycle, the SD card controller will read the data of the target block one byte at a time. Each time a data byte (of the 512 bytes) is ready, the flag sd_valid raises to 1 for one clock cycle.



Tri-color LED Signals

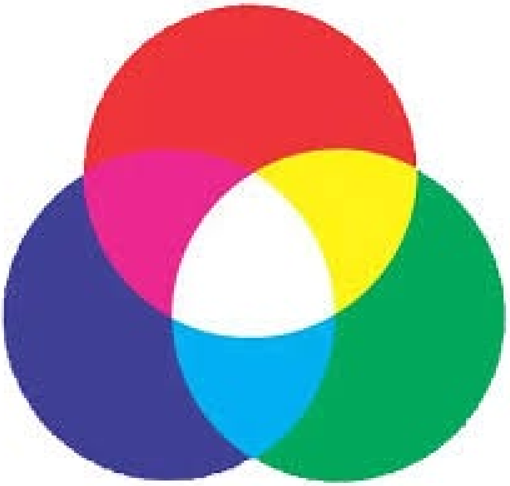
- The key signals that control RGB LED operations:
  - [3:0] rgb_led_r : control RED LEDs
  - [3:0] rgb_led_g : control GREEN LEDs
  - [3:0] rgb_led_b : control BLUE LEDs

- If you set rgb_led_r[0] to 1, the red LED 0 will be turn on.



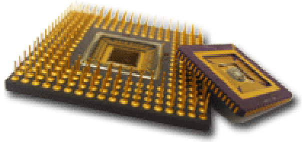
Tri-color LED Signals

- Due to the excessive brightness of the tri-color LED, it could potentially harm your eyes if not addressed. In this Lab, please reduce the PWM to a duty cycle of 5%.
- You can mix light colors to achieve colors beyond red, blue, and green.



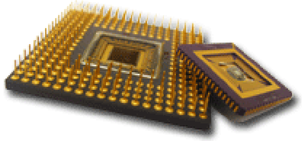
What You Need to Do for Lab 8

- You must design a circuit that:
    - You can download “test_EASY.txt” , “test_NORMAL.txt” or “test_HARD.txt” from E3 and save it on the SD cards.
    - When BTN2 is pressed, the circuit reads the SD card and searches for an ASCII file. This step must not exceed 10 seconds.
    - You must process the information which begins with the word “DCL_START” and ends with the word “DCL_END”. You should ignore other noise.
    - The circuit reads the text file and displays the information through a tri-color LED. Upon reading 'DCL_END,' the circuit stops and shows additional information on the LCD display.



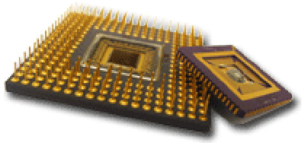
What You Need to Do for Lab 8

- Wait for button 2.
- Find the target ASCII file by searching “DCL_START”
- Show the information provided between “DCL_START” and “DCL_END” through tri-color LED.
- Upon reading “DCL_END”, display statistical results and stop your circuit.



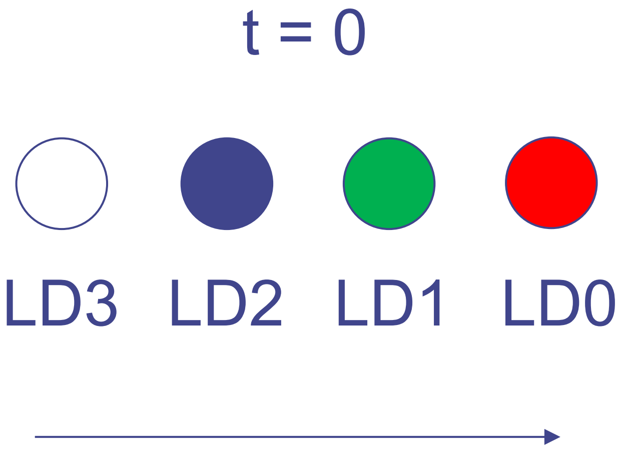
What You Need to Do for Lab 8

- In this lab, you will need to identify specific characters in an ASCII file and display them sequentially on four tri-color LEDs.
  - ‘R’ or ‘r’ : RED
  - ‘G’ or ‘g’ : GREEN
  - ‘B’ or ‘b’ : BLUE
  - ‘Y’ or ‘y’ : YELLOW
  - ‘P’ or ‘p’ : PURPLE
  - Other : Close LED
- Reduce the PWM to a duty cycle of 5%.



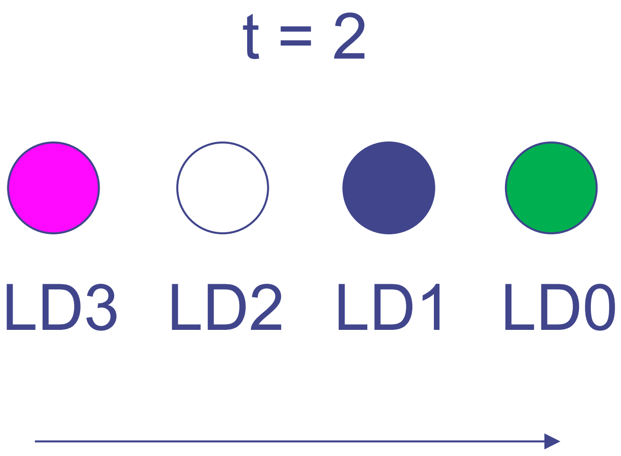
What You Need to Do for Lab 8

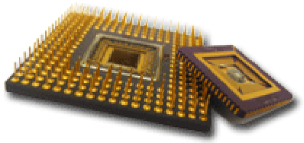
- For example:
  - “RDCL_STARTRGBAPYDCL_ENDR”
  - When t = 0, you should read 4 characters and display them via RGB LED together.
  - RDCL_STARTRGBAPYDCL_ENDR



What You Need to Do for Lab 8

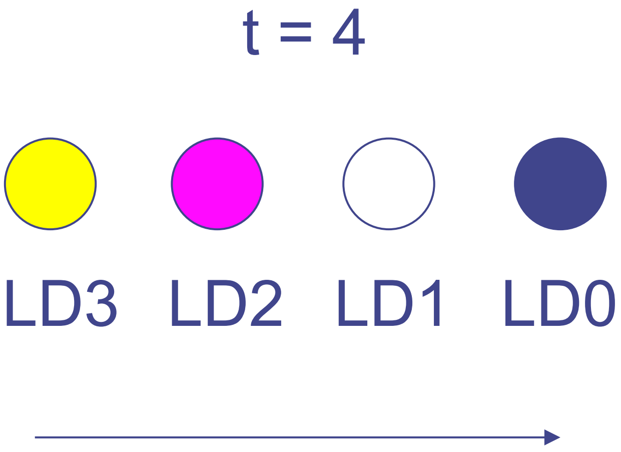
- For example:
  - “RDCL_STARTRGBAPYDCL_ENDR”
  - When t = 2, you should read next character and shift all the information.
  - RDCL_STARTRGBAPYDCL_ENDR

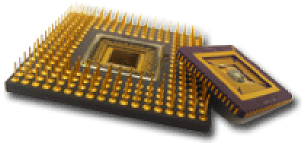




What You Need to Do for Lab 8

- For example:
  - “RDCL_STARTRGBAPYDCL_ENDR”
  - When t = 4, you should read next character and shift all the information.
  - RDCL_STARTRGBAPYDCL_ENDR



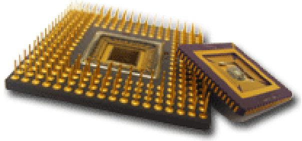


What You Need to Do for Lab 8

- When you read “DCL_END”, close all the tri-color LEDs and print the following information in LCD display.
  - *r* : The number of character “R” or “r”
  - *g* : The number of character “G” or “g”
  - …
  - *x* : The number of character that is not valid.
- For example, when finish processing “test_EASY.txt”:

```
RGBPYX
rgbpyx
```

```
RGBPYX
333334
```

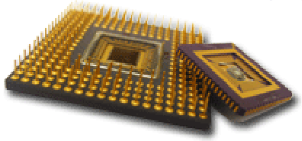


Format of the Input Text File

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

```
ABCDEFGHIJKLMNOPQUSDCLWXYZDCL_STARTRGBYP rgbyp A
RGBYPDCL_ENDABC
```

Note: ↵ stands for 0x0A.

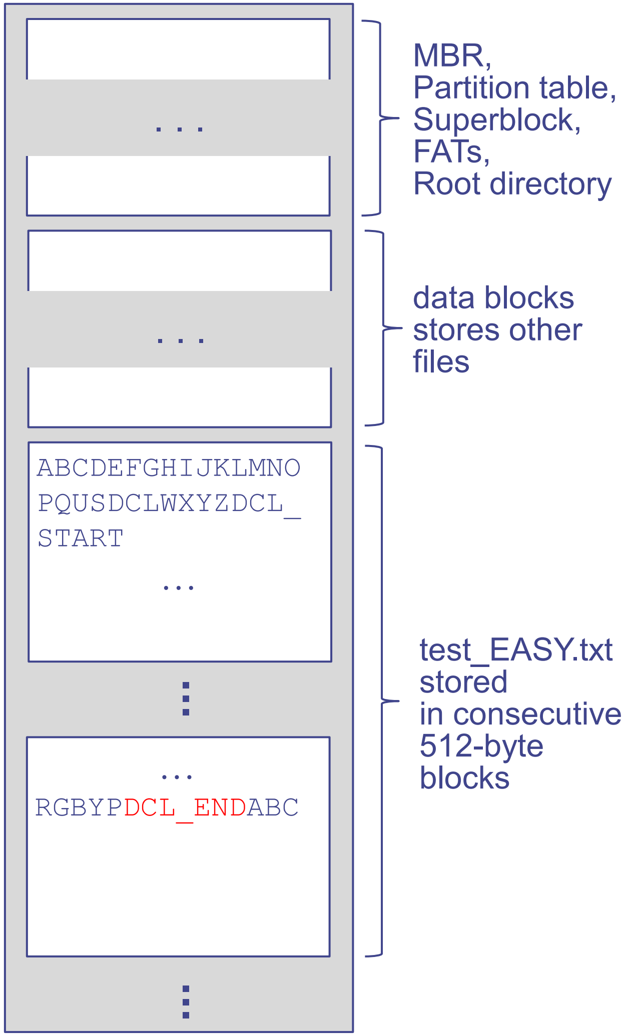


Layout of the File on the SD card

- 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.



. . .

MBR,
Partition table,
Superblock,
FATs,
Root directory

. . .

data blocks
stores other
files

ABCDEFGHIJKLMNO
PQUSDCLWXYZDCL_
START
…

⋮

…
RGBYPDCL_ENDABC

⋮

test_EASY.txt
stored
in consecutive
512-byte
blocks