uCore-SMP 的移植

展示报告

陶天骅 2021.11.4

总体情况

- 实现了上学期的功能，在FU740上的运行
  - 用ramdisk，仍没有SD卡驱动
- 虽然SD卡驱动没成功跑起来，但是<u>对整体的流程有了了解</u>
- 然后发现细节超级复杂，跑不起来也正常
- 关于启动部分的流程，之前有两次报告，在这里略去
  - （概要：把kernel做成一个U–Boot支持的 Image，放在 SD 卡的 /boot 分区，让U–Boot加载启动）

SD卡驱动的困难

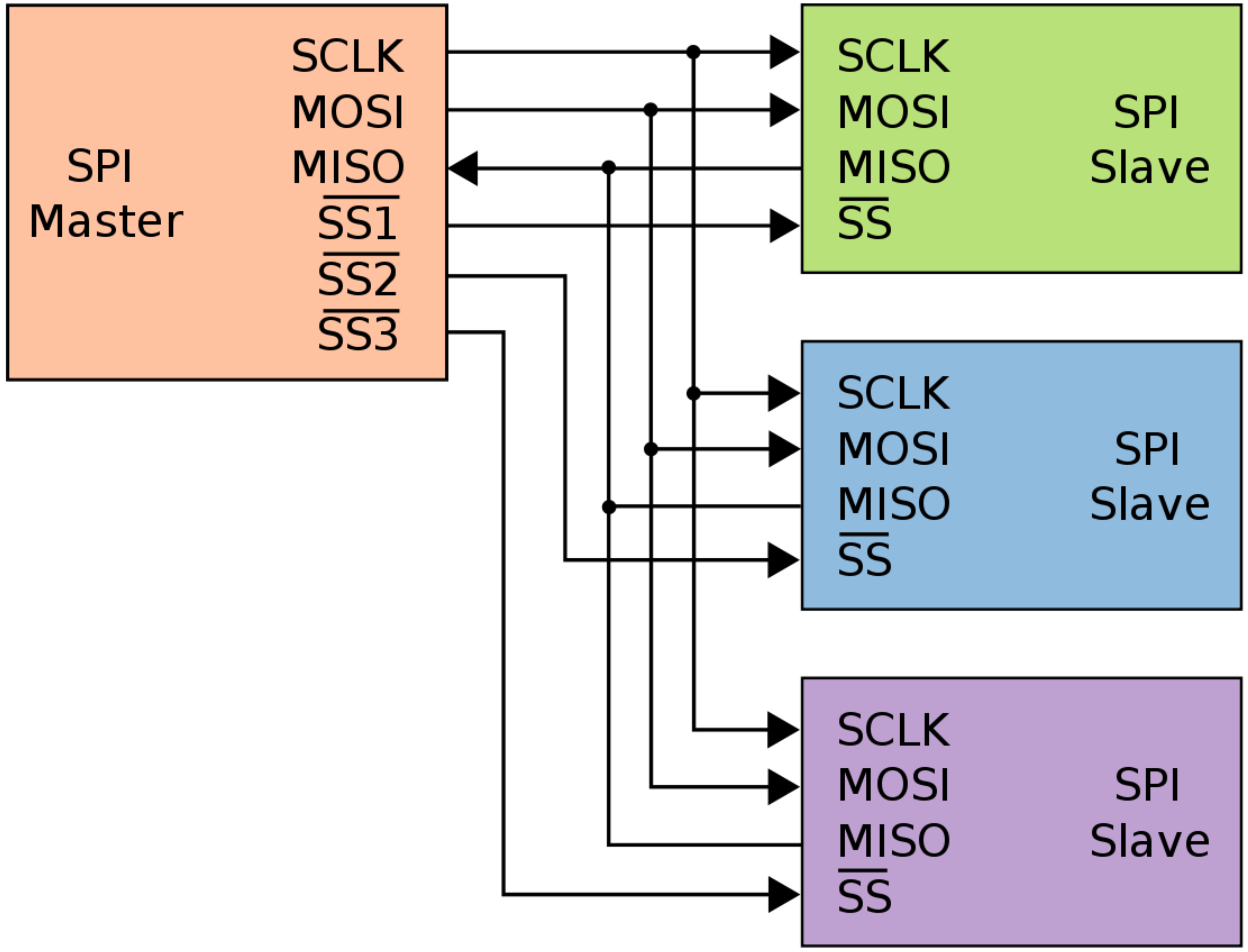
SPI + SD Card + 文件系统

- 简单来说，我要
    - 先符合SPI通信的协议；
    - 然后再其上，实现SD卡的规范；
    - 再然后，加入文件系统支持，比如FAT
- 3层抽象
- 现有的代码都不能很好的地集成进来

什么是SPI

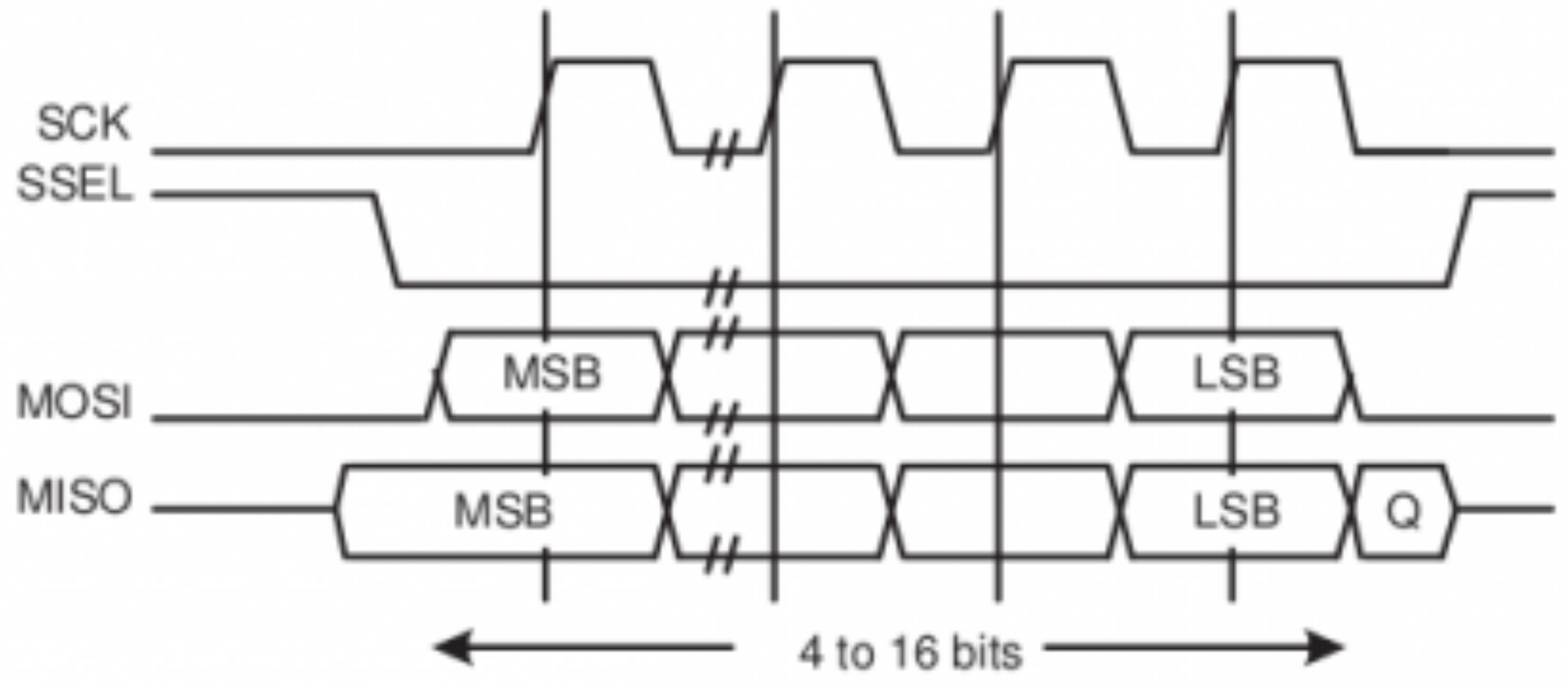
SPI的基本说明

- MOSI: Master Out Slave In
- MISO: Master In Slave Out
- SCLK: Clock
- SS: Chip Select



SPI Format

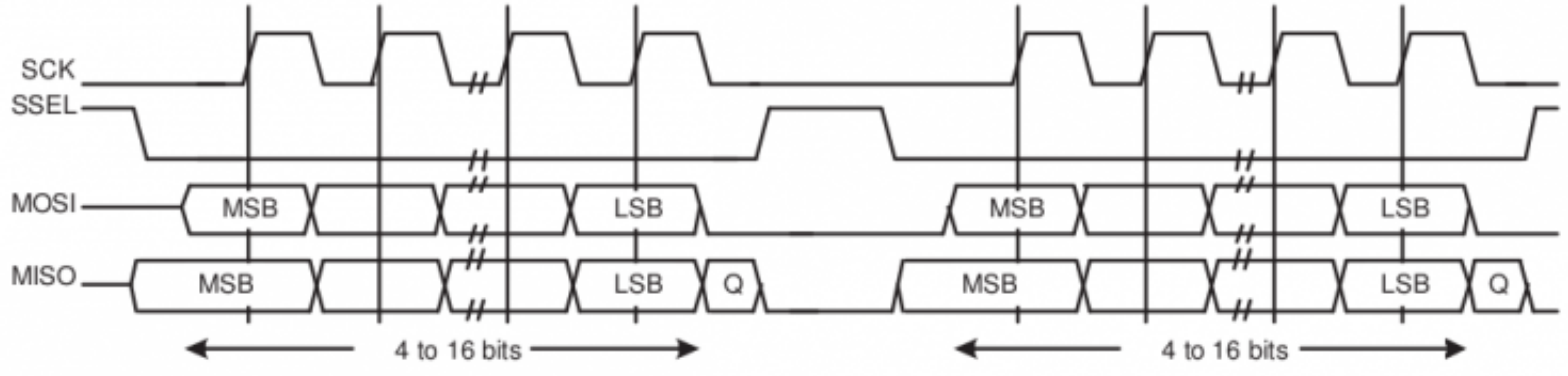
- SPI 使用 CPOL (Clock Polarity) 和 CPHA (Clock Phase) 定义 SCK 和 MOSI/MISO 的时序关系
  - CPOL 定义 Idle 时的 SCK 电平：0=Low，1=High
  - CPHA 定义取样的时钟边沿：0=第一个边，1=第二个边



a. Single transfer with CPOL=0 and CPHA=0

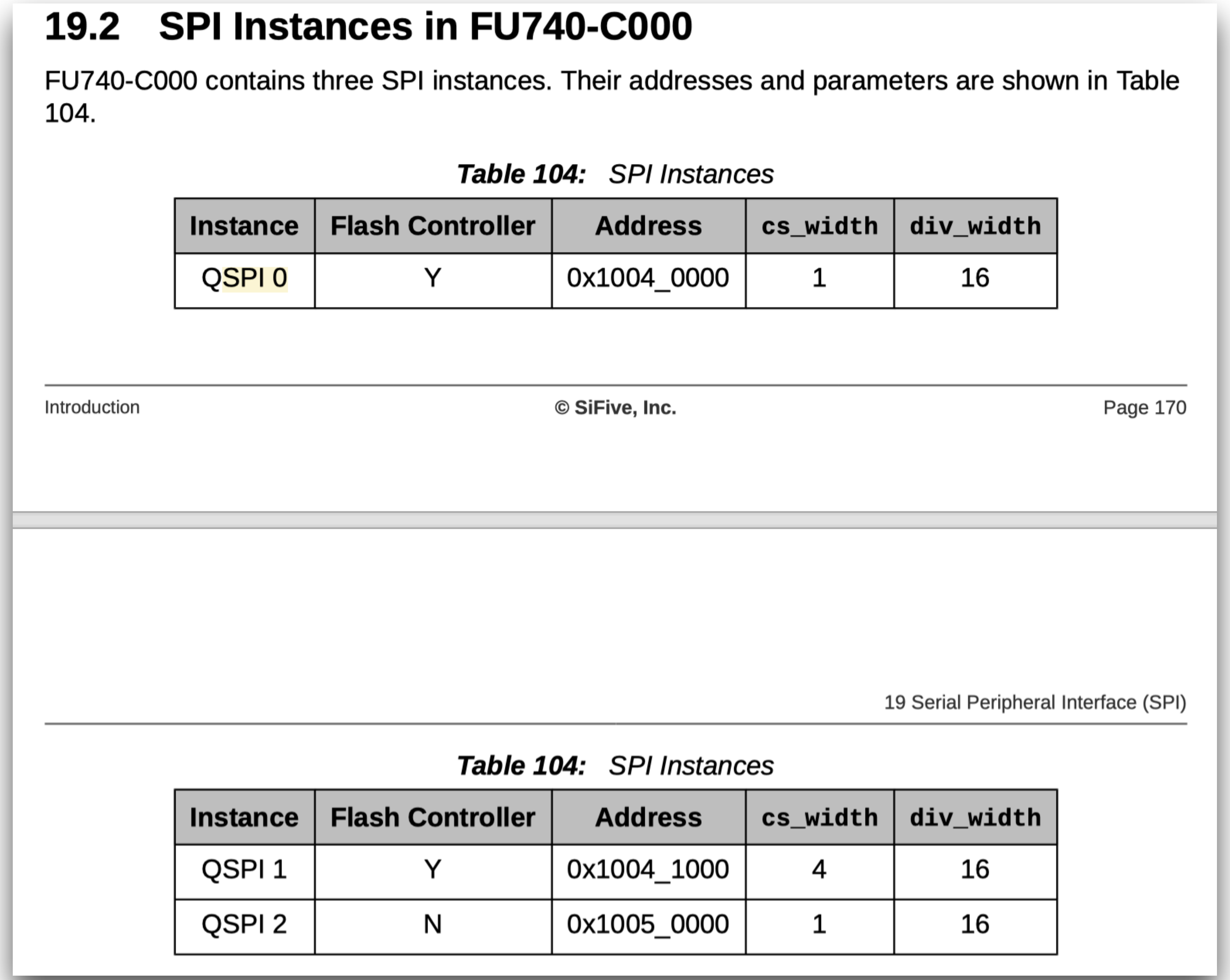
SPI Format

- SPI 使用 CPOL (Clock Polarity) 和 CPHA (Clock Phase) 定义 SCK 和 MOSI/MISO 的时序关系
  - CPOL 定义 Idle 时的 SCK 电平：0=Low，1=High
  - CPHA 定义取样的时钟边沿：0=第一个边，1=第二个边



b. Continuous transfer with CPOL=0 and CPHA=0

Unmatched 上的SPI



19.2 SPI Instances in FU740-C000

FU740-C000 contains three SPI instances. Their addresses and parameters are shown in Table 104.

Table 104: *SPI Instances*

Instance	Flash Controller	Address	cs_width	div_width
QSPI 0	Y	0x1004_0000	1	16
QSPI 1	Y	0x1004_1000	4	16
QSPI 2	N	0x1005_0000	1	16

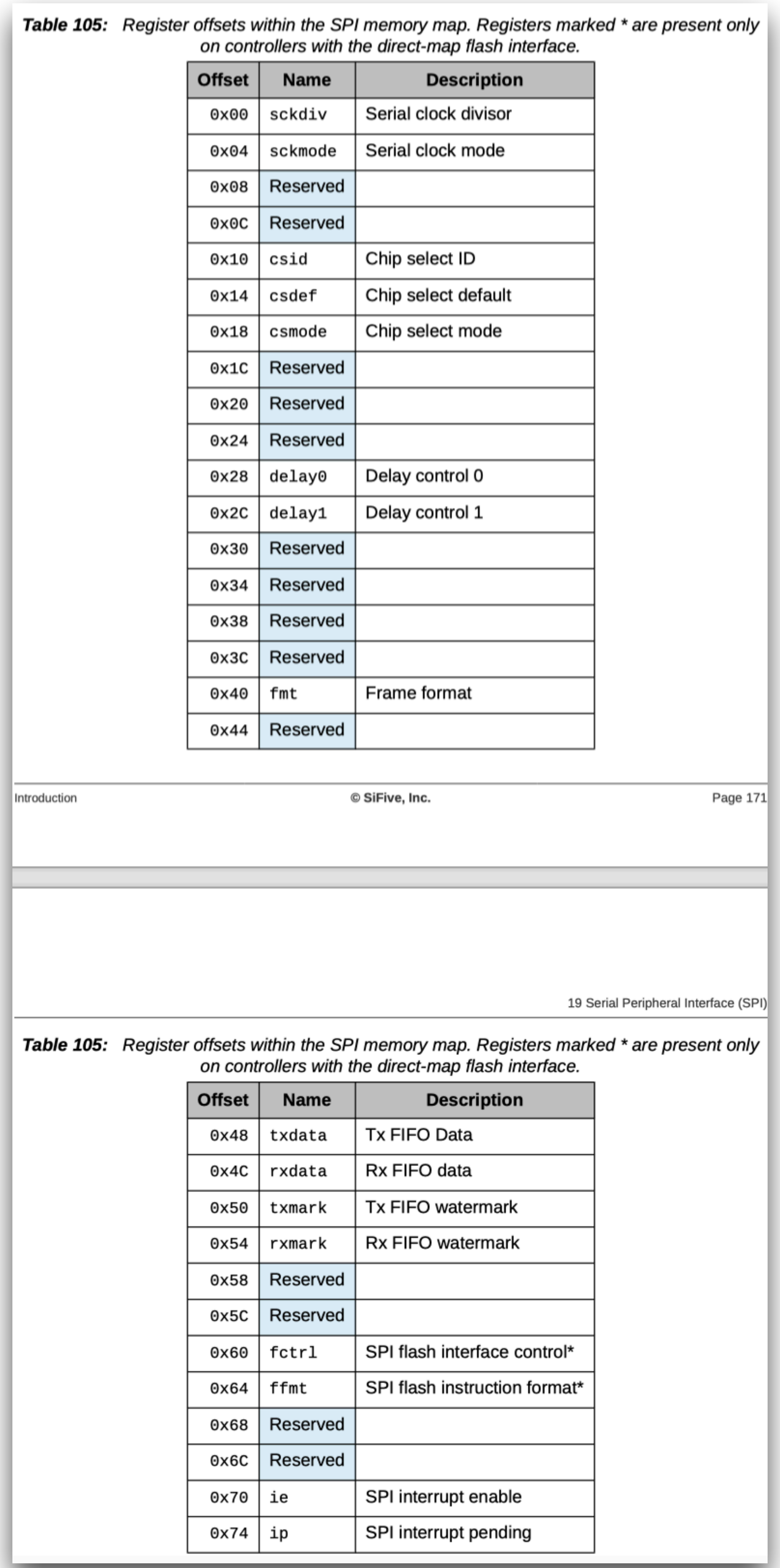


Table 105: *Register offsets within the SPI memory map. Registers marked * are present only on controllers with the direct-map flash interface.*

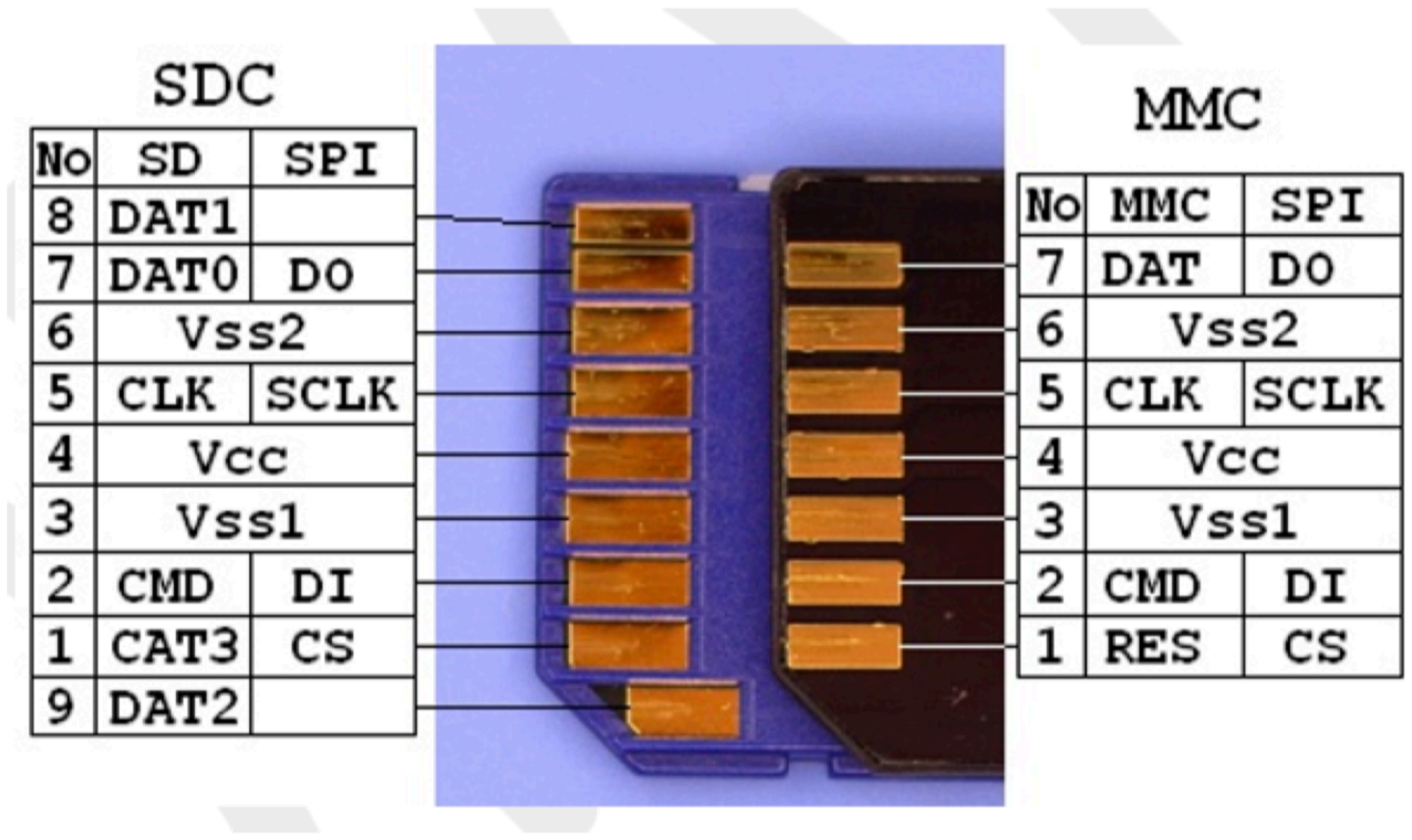
Offset	Name	Description
0x00	sckdiv	Serial clock divisor
0x04	sckmode	Serial clock mode
0x08	Reserved	
0x0C	Reserved	
0x10	csid	Chip select ID
0x14	csdef	Chip select default
0x18	csmode	Chip select mode
0x1C	Reserved	
0x20	Reserved	
0x24	Reserved	
0x28	delay0	Delay control 0
0x2C	delay1	Delay control 1
0x30	Reserved	
0x34	Reserved	
0x38	Reserved	
0x3C	Reserved	
0x40	fmt	Frame format
0x44	Reserved	
0x48	txdata	Tx FIFO Data
0x4C	rxdata	Rx FIFO data
0x50	txmark	Tx FIFO watermark
0x54	rxmark	Rx FIFO watermark
0x58	Reserved	
0x5C	Reserved	
0x60	fctrl	SPI flash interface control*
0x64	ffmt	SPI flash instruction format*
0x68	Reserved	
0x6C	Reserved	
0x70	ie	SPI interrupt enable
0x74	ip	SPI interrupt pending

有了SPI之后，怎么操作SD卡

SD卡的基本情况

- 什么是SDC
    - Secure Digital Card
- 什么是MMC
    - MultiMedia Card
- MMC更老，SD卡都兼容MMC模式
- 都可以在SPI模式下使用，这时可以用同一套代码

Figure 2. Card signals



SDC

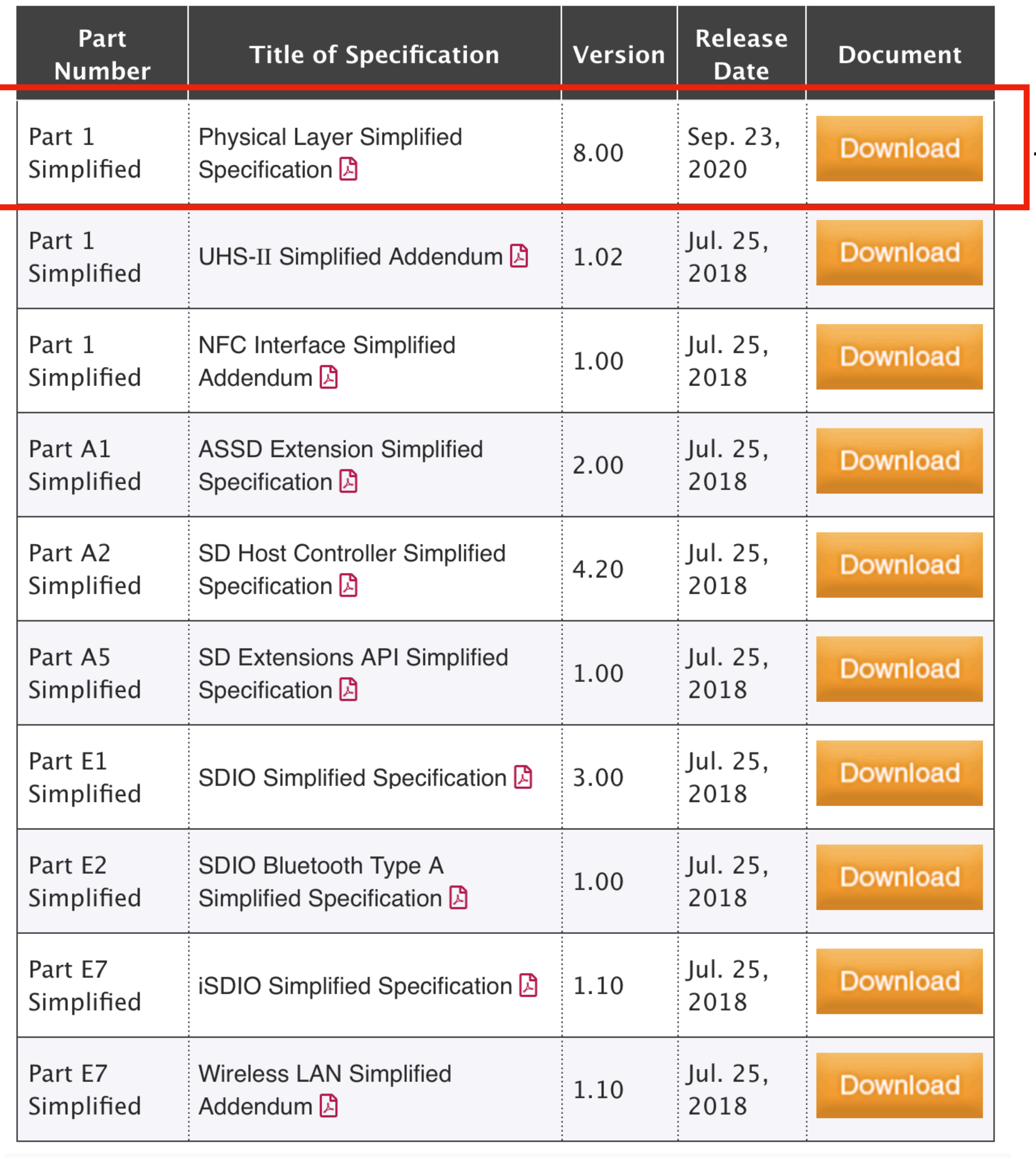
No	SD	SPI
8	DAT1	
7	DAT0	DO
6	Vss2	
5	CLK	SCLK
4	Vcc	
3	Vss1	
2	CMD	DI
1	CAT3	CS
9	DAT2	

MMC

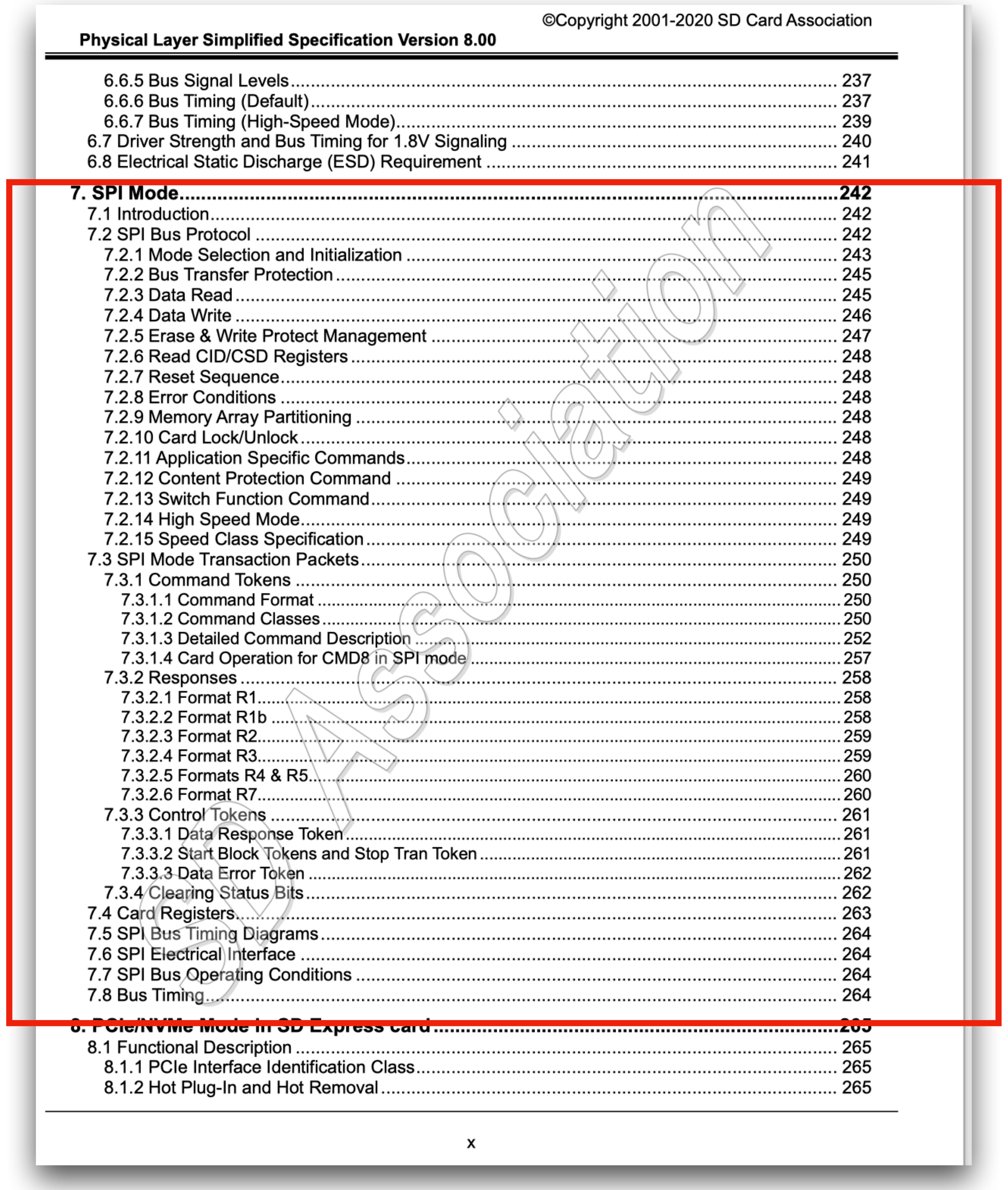
No	MMC	SPI
7	DAT	DO
6	Vss2	
5	CLK	SCLK
4	Vcc	
3	Vss1	
2	CMD	DI
1	RES	CS

有了SPI之后，怎么操作SD卡

SD卡的规范，https://www.sdcard.org/downloads/pls/



Part Number	Title of Specification	Version	Release Date	Document
Part 1 Simplified	Physical Layer Simplified Specification	8.00	Sep. 23, 2020	Download
Part 1 Simplified	UHS-II Simplified Addendum	1.02	Jul. 25, 2018	Download
Part 1 Simplified	NFC Interface Simplified Addendum	1.00	Jul. 25, 2018	Download
Part A1 Simplified	ASSD Extension Simplified Specification	2.00	Jul. 25, 2018	Download
Part A2 Simplified	SD Host Controller Simplified Specification	4.20	Jul. 25, 2018	Download
Part A5 Simplified	SD Extensions API Simplified Specification	1.00	Jul. 25, 2018	Download
Part E1 Simplified	SDIO Simplified Specification	3.00	Jul. 25, 2018	Download
Part E2 Simplified	SDIO Bluetooth Type A Simplified Specification	1.00	Jul. 25, 2018	Download
Part E7 Simplified	iSDIO Simplified Specification	1.10	Jul. 25, 2018	Download
Part E7 Simplified	Wireless LAN Simplified Addendum	1.10	Jul. 25, 2018	Download



©Copyright 2001-2020 SD Card Association

Physical Layer Simplified Specification Version 8.00

x

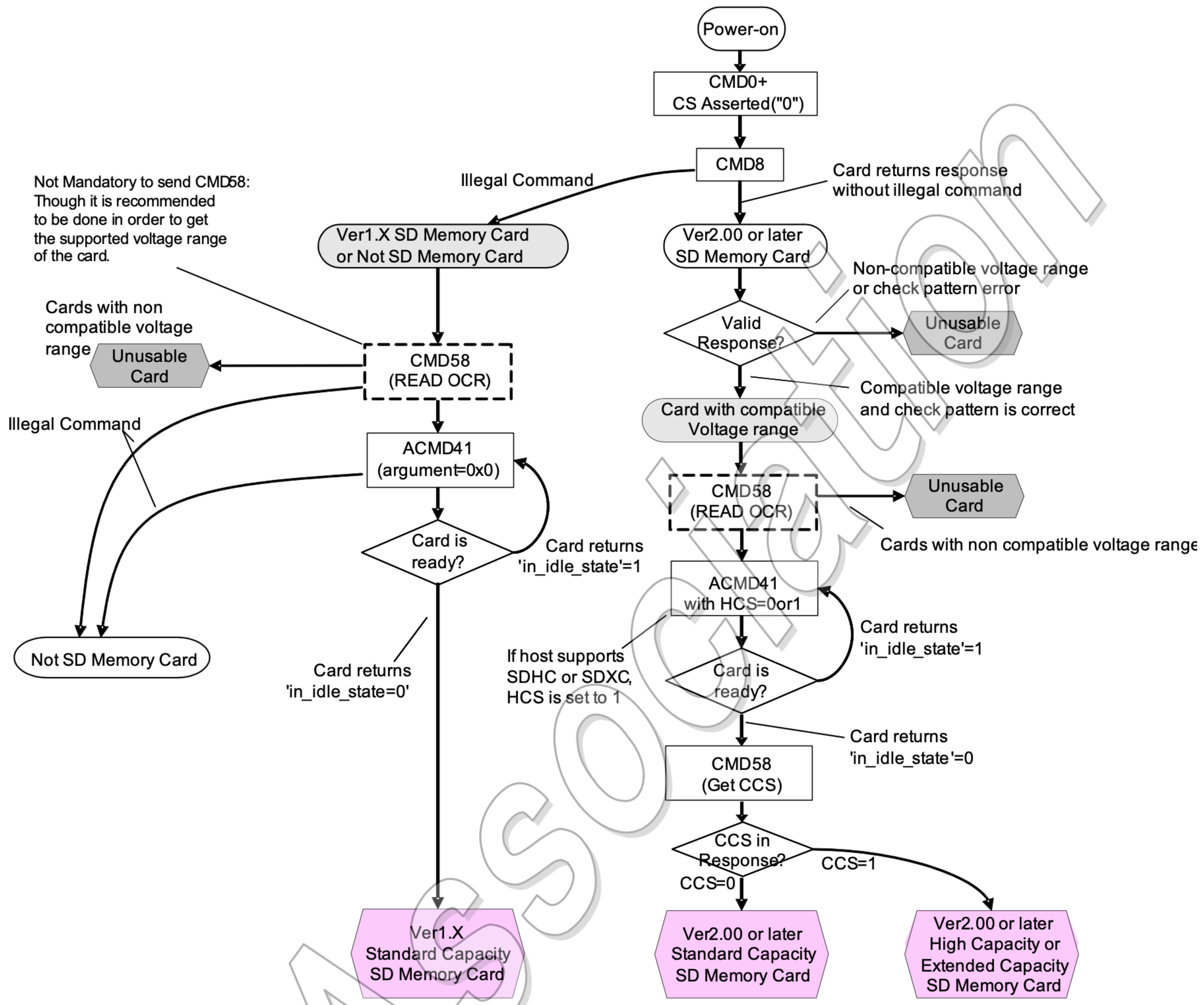
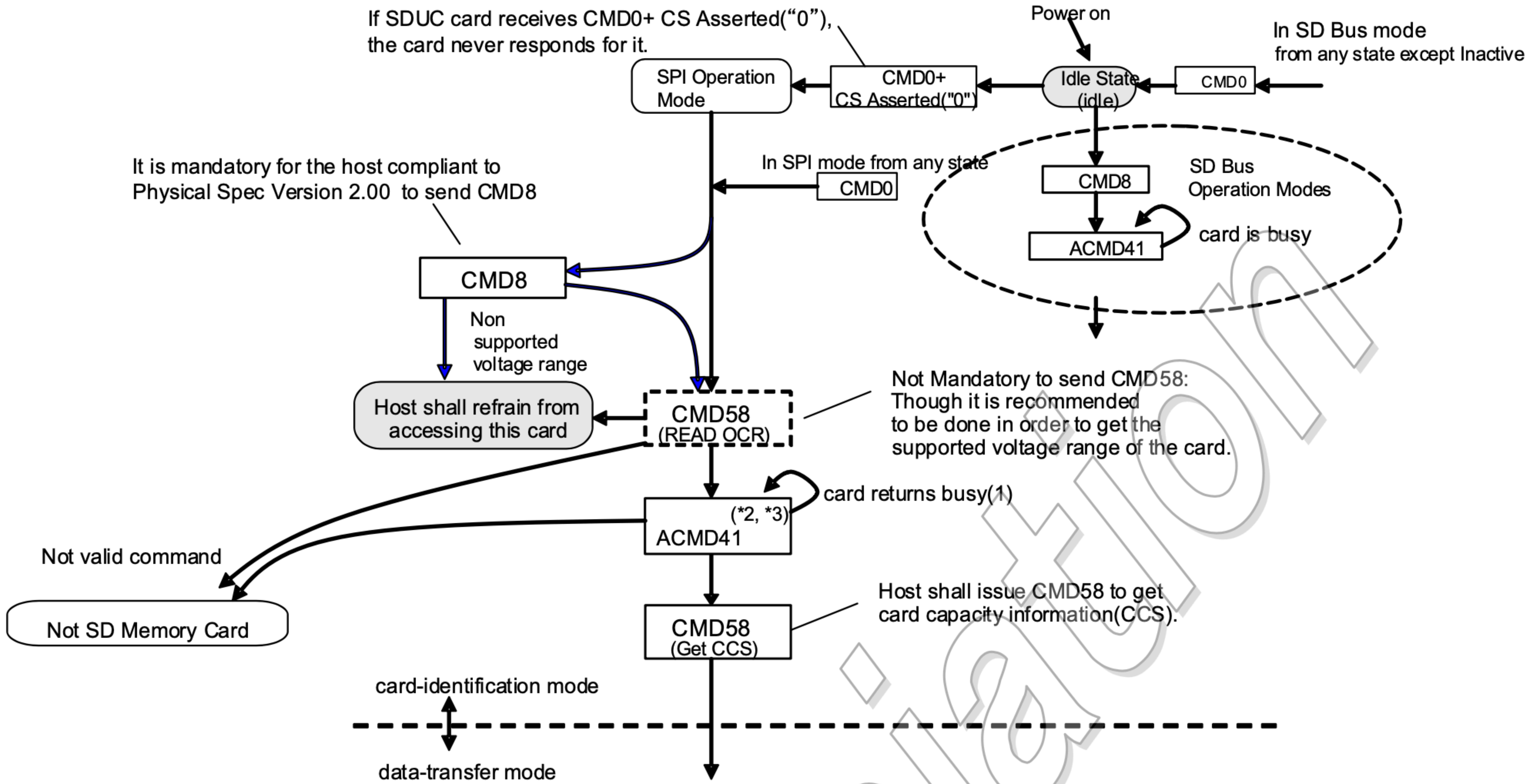

Power-on
CMD0+ CS Asserted("0")
CMD8
Illegal Command
Card returns response without illegal command
Not Mandatory to send CMD58: Though it is recommended to be done in order to get the supported voltage range of the card.
Ver1.X SD Memory Card or Not SD Memory Card
Ver2.00 or later SD Memory Card
Non-compatible voltage range or check pattern error
Valid Response?
Unusable Card
Cards with non compatible voltage range
Unusable Card
CMD58 (READ OCR)
Compatible voltage range and check pattern is correct
Card with compatible Voltage range
Illegal Command
ACMD41 (argument=0x0)
CMD58 (READ OCR)
Unusable Card
Cards with non compatible voltage range
Card is ready?
Card returns 'in_idle_state'=1
ACMD41 with HCS=0or1
Card returns 'in_idle_state'=1
Not SD Memory Card
If host supports SDHC or SDXC, HCS is set to 1
Card is ready?
Card returns 'in_idle_state=0'
Card returns 'in_idle_state'=0
CMD58 (Get CCS)
CCS in Response?
CCS=1
CCS=0
Ver1.X Standard Capacity SD Memory Card
Ver2.00 or later Standard Capacity SD Memory Card
Ver2.00 or later High Capacity or Extended Capacity SD Memory Card

Figure 7-2 : SPI Mode Initialization Flow



If SDUC card receives CMD0+ CS Asserted("0"), the card never responds for it.
Power on
In SD Bus mode from any state except Inactive
SPI Operation Mode
CMD0+ CS Asserted("0")
Idle State (idle)
CMD0
In SPI mode from any state
CMD0
CMD8
SD Bus Operation Modes
ACMD41
card is busy
It is mandatory for the host compliant to Physical Spec Version 2.00 to send CMD8
CMD8
Non supported voltage range
Host shall refrain from accessing this card
CMD58 (READ OCR)
Not Mandatory to send CMD58: Though it is recommended to be done in order to get the supported voltage range of the card.
card returns busy(1)
(*2, *3) ACMD41
Not valid command
Not SD Memory Card
CMD58 (Get CCS)
Host shall issue CMD58 to get card capacity information(CCS).
card-identification mode
data-transfer mode

(*1) Note: Card returns busy when
- Card executes internal initialization process.
- If the card is High capacity SD Memory Card, there are 2 cases
  1. CMD8 was not issued before ACMD41
  2. ACMD41 is issued with HCS=0

(*2) Note: 2.1mm SD Memory Card can be initialized using CMD1 and Thin (1.4mm) SD Memory Card can be initialized using CMD1 only after firstly initialized by using CMD0 and ACMD41. In any of the cases CMD1 is not recommended because it may be difficult for the host to distinguish between MultiMediaCard and SD Memory Card.
If the SD card is initialized by CMD1 and the host treat it as MMC card, not SD card, the Data of the card may be damaged because of wrong interpretation of CSD and CID registers.

(*3) Note: SDUC card, card can stay at busy status and does not reply ready to host during ACMD41 to let host know SDUC card cannot use SPI mode.

Figure 7-1 : SD Memory Card State Diagram (SPI mode)

7.2.3 Data Read

The SPI mode supports single block read and Multiple Block read operations (CMD17 or CMD18 in the SD Memory Card protocol). Upon reception of a valid read command the card will respond with a response token followed by a data token (Refer to Figure 7-3). In case of Standard Capacity Card, the size in the data token is determined by the block length set by SET_BLOCKLEN (CMD16). In case of SDHC and SDXC Cards, block length is fixed to 512 Bytes regardless of the block length set by CMD16.

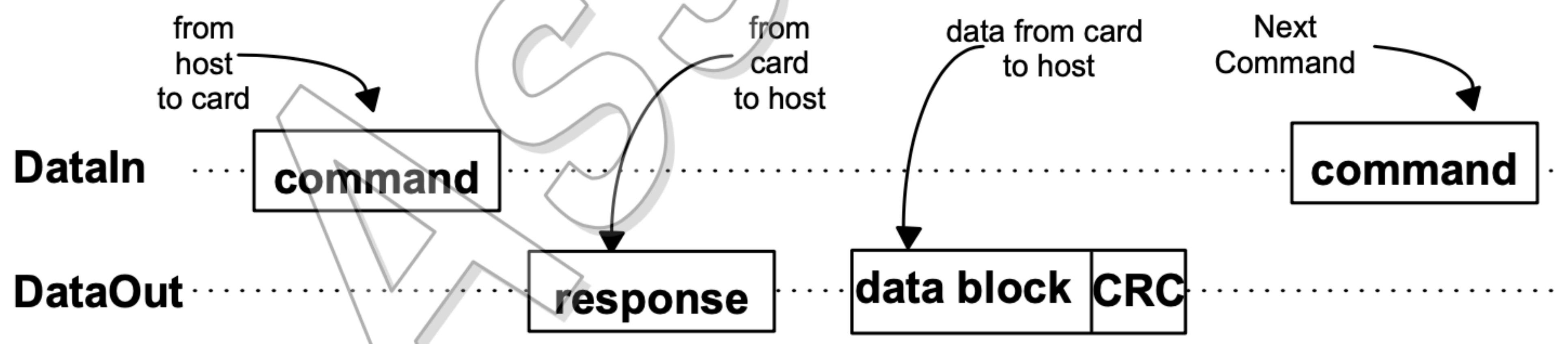


Figure 7-3 : Single Block Read Operation

7.2.4 Data Write

The SPI mode supports single block and multiple block write commands. Upon reception of a valid write command (CMD24 or CMD25 in the SD Memory Card protocol), the card will respond with a response token and will wait for a data block to be sent from the host. CRC suffix, block length and start address restrictions are (with the exception of the CSD parameter WRITE_BL_PARTIAL controlling the partial block write option and WRITE_BL_LEN) identical to the read operation (Refer to Figure 7-6).

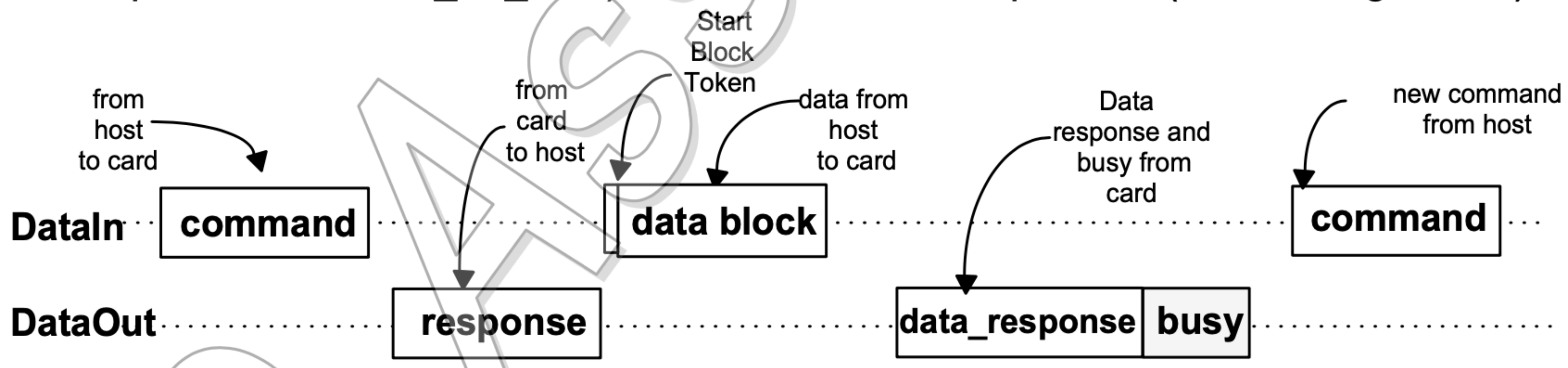


Figure 7-6 : Single Block Write Operation

7.3.1.3 Detailed Command Description

The following table provides a detailed description of the SPI bus commands. The responses are defined in Section 7.3.2. Table 7-3 lists all SD Memory Card commands. A "yes" in the SPI mode column indicates that the command is supported in SPI mode. With these restrictions, the command class description in the CSD is still valid. If a command does not require an argument, the value of this field should be set to zero. The reserved commands are reserved in SD mode as well.
The binary code of a command is defined by the mnemonic symbol. As an example, the content of the **command index** field is (binary) '000000' for CMD0 and '100111' for CMD39.
The card shall ignore stuff bits and reserved bits in an argument.

CMD INDEX	SPI Mode	Argument	Resp	Abbreviation	Command Description
CMD0	Yes	[31:0] stuff bits	R1	GO_IDLE_STATE	Resets the SD Memory Card
CMD1	Yes[1]	[31]Reserved bit [30]HCS [29:0]Reserved bits	R1	SEND_OP_ COND	Sends host capacity support information and activates the card's initialization process. HCS is effective when card receives SEND_IF_COND command. Reserved bits shall be set to '0'.
CMD2	No				
CMD3	No				
CMD4	No				
CMD5	Reserved for I/O Mode (refer to "SDIO Card Specification")				
CMD6[8]	Yes	[31] Mode 0:Check function 1:Switch function [30:24] reserved (All '0') [23:20] reserved for function group 6 (All '0' or 0xF) [19:16] reserved for function group 5 (All '0' or 0xF) [15:12] reserved for function group 4 (All '0' or 0xF) [11:8] reserved for function group 3 (All '0' or 0xF) [7:4] function group 2 for command system [3:0] function group 1 for access mode	R1	SWITCH_FUNC	Checks switchable function (mode 0) and switches card function (mode 1). See Section 4.3.10.
CMD7	No				

CMD INDEX	SPI Mode	Argument	Resp	Abbreviation	Command Description
CMD8[9]	Yes	[31:12]Reserved bits [11:8]supply voltage(VHS) [7:0]check pattern	R7	SEND_IF_COND	Sends SD Memory Card interface condition that includes host supply voltage information and asks the accessed card whether card can operate in supplied voltage range. Reserved bits shall be set to '0'.
CMD9	Yes	[31:0] stuff bits	R1	SEND_CSD	Asks the selected card to send its card-specific data (CSD)
CMD10	Yes	[31:0] stuff bits	R1	SEND_CID	Asks the selected card to send its card identification (CID)
CMD11	No				
CMD12	Yes	[31:0] stuff bits	R1b[5]	STOP_ TRANSMISSION	Forces the card to stop transmission in Multiple Block Read Operation
CMD13	Yes	[31:0] stuff bits	R2	SEND_STATUS	Asks the selected card to send its status register.
CMD14	reserved				
CMD15	No				
CMD16	Yes	[31:0] block length	R1	SET_BLOCKLEN	In case of SDSC Card, block length is set by this command. In case of SDHC and SDXC Cards, block length of the memory access commands are fixed to 512 bytes. The length of LOCK_UNLOCK command is set by this command regardless of card capacity.
CMD17	Yes	[31:0] data address[10]	R1	READ_SINGLE_ BLOCK	Reads a block of the size selected by the SET_BLOCKLEN command.[3]
CMD18	Yes	[31:0] data address[10]	R1	READ_MULTIPL E_BLOCK	Continuously transfers data blocks from card to host until interrupted by a STOP_TRANSMISSION command.
CMD19	reserved				
CMD20	No				
CMD21... CMD23	reserved				
CMD24	Yes	[31:0] data address[10]	R1	WRITE_BLOCK	Writes a block of the size selected by the SET_BLOCKLEN command.[4]
CMD25	Yes	[31:0] data address[10]	R1	WRITE_MULTIPL E_BLOCK	Continuously writes blocks of data until 'Stop Tran' token is sent (instead 'Start Block').
CMD26	No				
CMD27	Yes	[31:0] stuff bits	R1	PROGRAM_CSD	Programming of the programmable bits of the CSD.

CMD INDEX	SPI Mode	Argument	Resp	Abbreviation	Command Description
CMD28	Yes	[31:0] data address	R1b[5]	SET_WRITE_ PROT	If the card has write protection features, this command sets the write protection bit of the addressed group. The properties of write protection are coded in the card specific data (WP_GRP_SIZE). SDHC and SDXC Cards do not support this command.
CMD29	Yes	[31:0] data address	R1b[5]	CLR_WRITE_ PROT	If the card has write protection features, this command clears the write protection bit of the addressed group. SDHC and SDXC Cards do not support this command.
CMD30	Yes	[31:0] write protect data address	R1	SEND_WRITE_ PROT	If the card has write protection features, this command asks the card to send the status of the write protection bits.[6] SDHC and SDXC Cards do not support this command.
CMD31	reserved				
CMD32	Yes	[31:0] data address[10]	R1	ERASE_WR_BL K_START_ADDR	Sets the address of the first write block to be erased.
CMD33	Yes	[31:0] data address[10]	R1	ERASE_WR_BL K_END_ADDR	Sets the address of the last write block of the continuous range to be erased.
CMD34-37[8]	Reserved for each command system set by switch function command (CMD6). Refer to each command system specification for more detail.				
CMD38	Yes	[31:0] stuff bits	R1b[5]	ERASE	Erases all previously selected write blocks. FULE and DISCARD are not supported through SPI interface.
CMD39	No				
CMD40	No				
CMD41	Reserved				
CMD42	Yes	[31:0] Reserved bits (Set all 0)	R1	LOCK_UNLOCK	Used to Set/Reset the Password or lock/unlock the card. A transferred data block includes all the command details - refer to Section 4.3.7. The size of the Data Block is defined with SET_BLOCK_LEN command. Reserved bits in the argument and in Lock Card Data Structure shall be set to 0.
CMD43-49 CMD51	reserved				
CMD50[8]	Reserved for each command system set by switch function command (CMD6). Refer to each command system specification for more detail.				
CMD52-54	Reserved for I/O Mode (refer to "SDIO Card Specification")				

CMD INDEX	SPI Mode	Argument	Resp	Abbreviation	Command Description
CMD55	Yes	[31:0] stuff bits	R1	APP_CMD	Defines to the card that the next command is an application specific command rather than a standard command
CMD56	Yes	[31:1] stuff bits. [0]: RD/WR[7]	R1	GEN_CMD	Used either to transfer a Data Block to the card or to get a Data Block from the card for general purpose/application specific commands. In case of Standard Capacity SD Memory Card, the size of the Data Block shall be defined with SET_BLOCK_LEN command. In case of SDHC and SDXC Cards, block length of this command is fixed to 512-byte.
CMD57[8]	Reserved for each command system set by switch function command (CMD6). Refer to each command system specification for more detail.				
CMD58	Yes	[31:0] stuff bits	R3	READ_OCR	Reads the OCR register of a card. CCS bit is assigned to OCR[30].
CMD59	Yes	[31:1] stuff bits [0:0] CRC option	R1	CRC_ON_OFF	Turns the CRC option on or off. A '1' in the CRC option bit will turn the option on, a '0' will turn it off
CMD60-63	Reserved For Manufacturer				

1. CMD1 is valid command for the Thin (1.4mm) Standard Size SD Memory Card only if used after re-initializing a card (not after power on reset).
2. The default block length is as specified in the CSD.
3. The data transferred shall not cross a physical block boundary unless READ_BLK_MISALIGN is set in the CSD.
4. The data transferred shall not cross a physical block boundary unless WRITE_BLK_MISALIGN is set in the CSD.
5. R1b: R1 response with an optional trailing busy signal
6. 32 write protection bits (representing 32 write protect groups starting at the specified address) followed by 16 CRC bits are transferred in a payload format via the data line. The last (least significant) bit of the protection bits corresponds to the first addressed group. If the addresses of the last groups are outside the valid range, then the corresponding write protection bits shall be set to zero
7. RD/WR_: "1" the Host shall get a block of data from the card.
   "0" the host sends block of data to the card.
8. This command was added in spec version 1.10
9. This command is added in spec version 2.00
10. SDSC Card (CCS=0) uses byte unit address and SDHC and SDXC Cards (CCS=1) use block unit address (512 bytes unit).

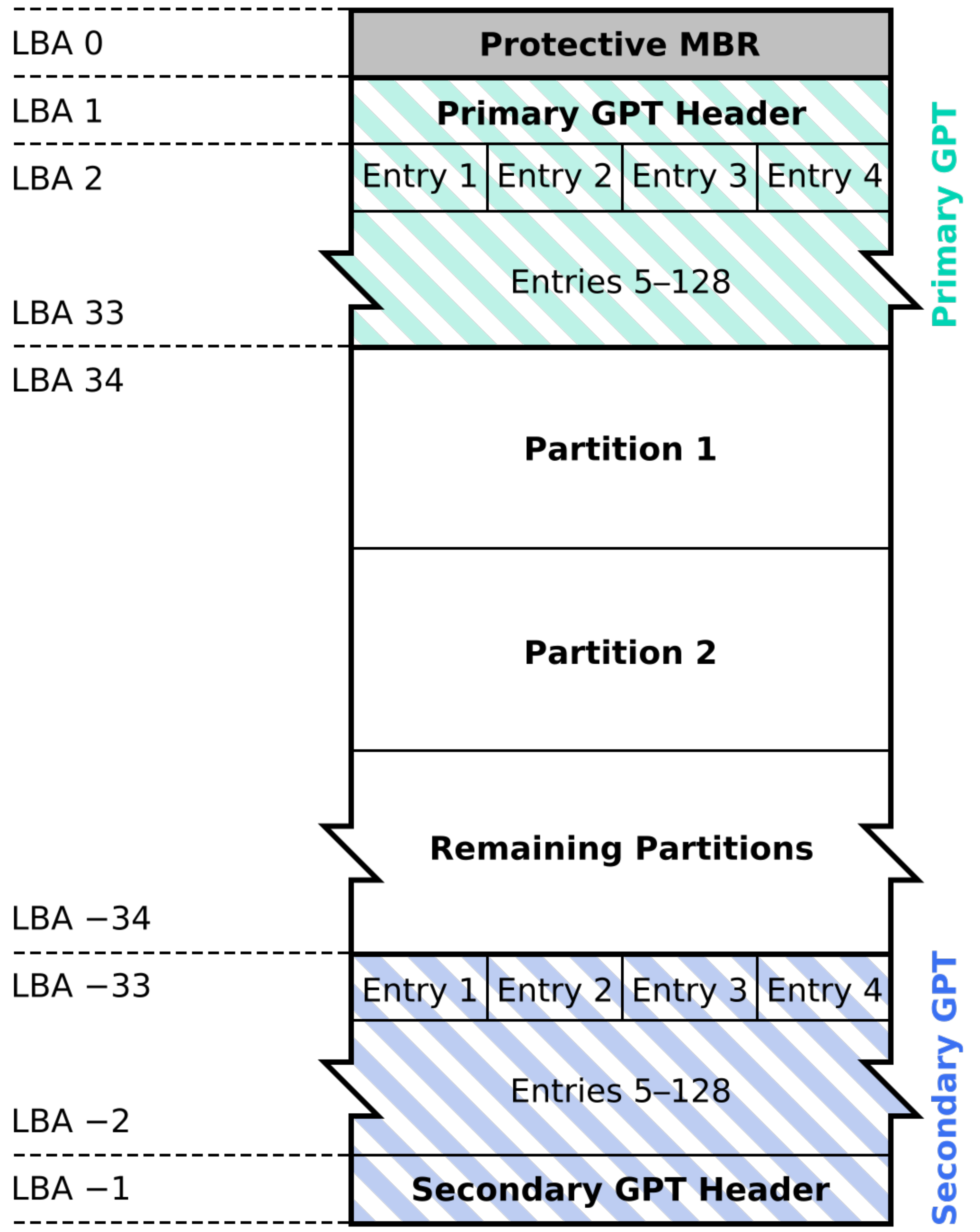
Table 7-3 : Commands and Arguments

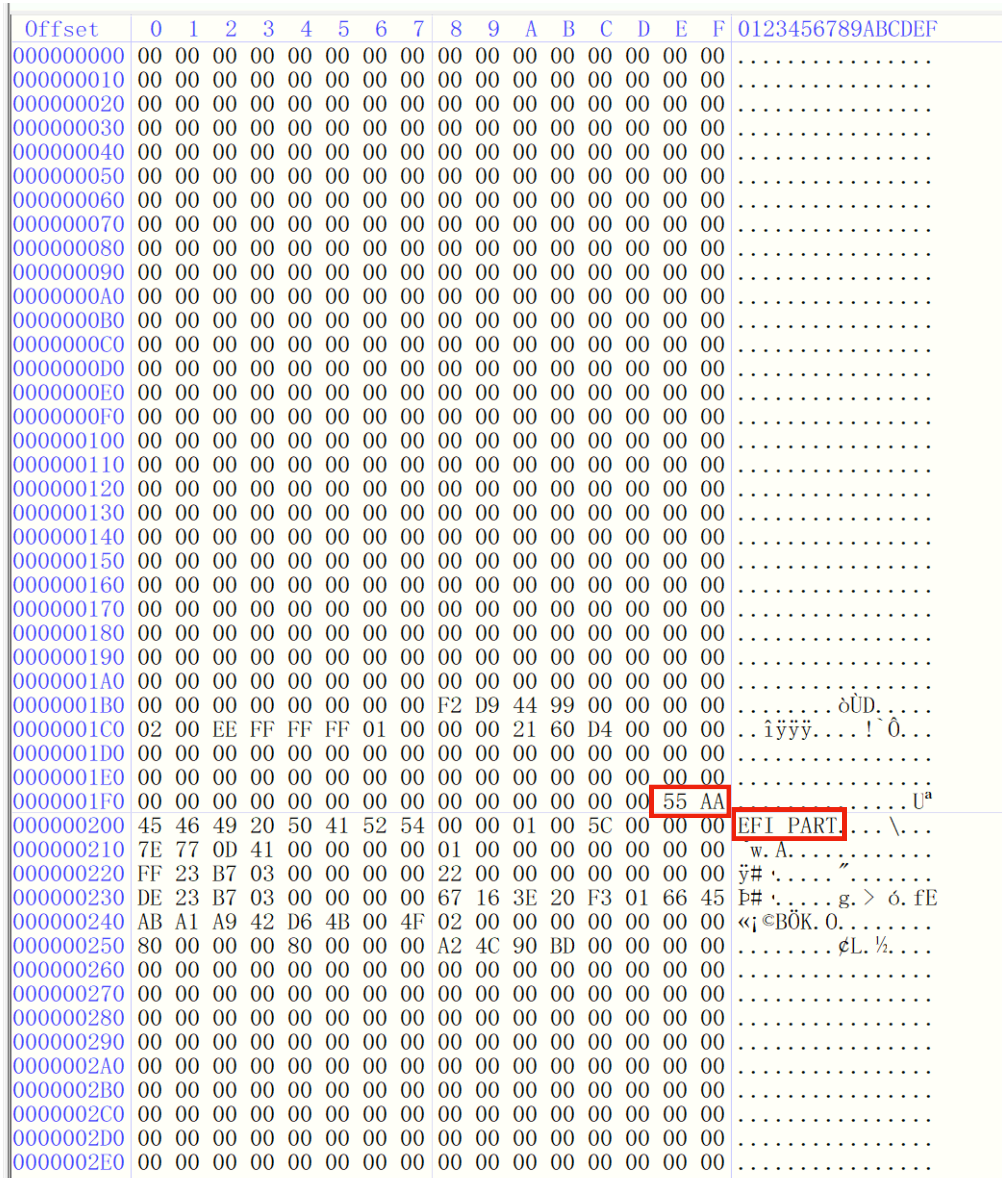
能操作SD卡之后，怎么识别里面的内容

需要读取分区表，识别分区格式

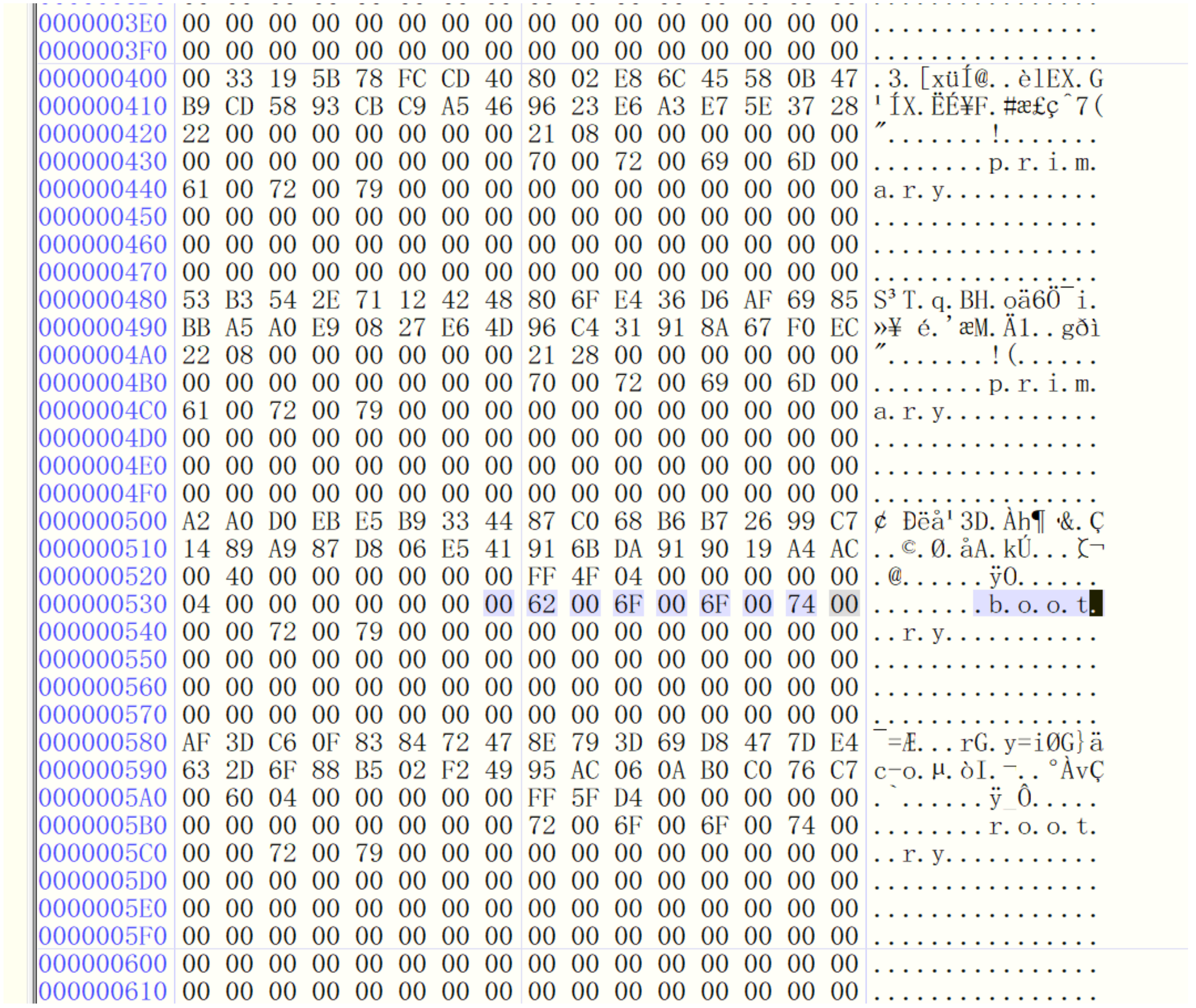
- 首先读取GPT头，然后再去找分区
- 找到具体分区之后，要有支持文件系统的代码（比如FAT32)

GUID Partition Table Scheme



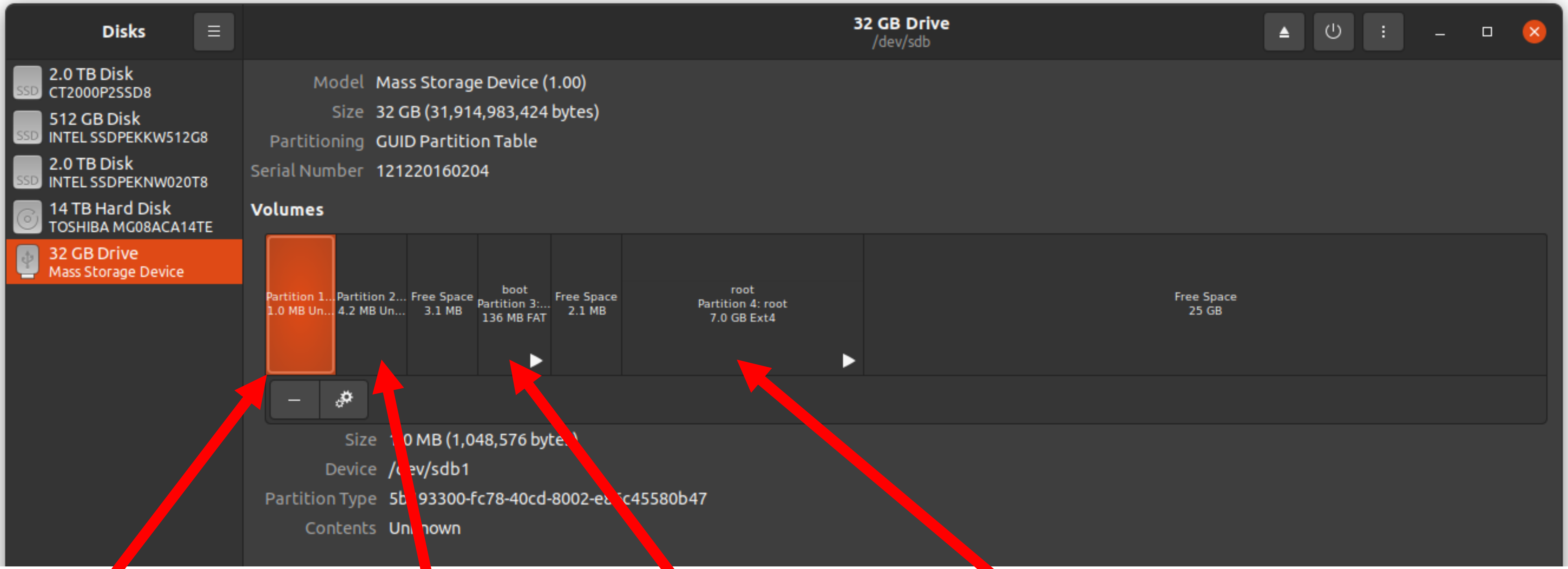


```
Offset     0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
000000000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000000010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000000020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000000030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000000040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000000050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000000060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000000070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000000080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000000090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0000000A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0000000B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0000000C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0000000D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0000000E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0000000F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000000100 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000000110 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000000120 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000000130 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000000140 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000000150 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000000160 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000000170 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000000180 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000000190 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0000001A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0000001B0 00 00 00 00 00 00 00 00 F2 D9 44 99 00 00 00 00 ........òÙD.....
0000001C0 02 00 EE FF FF FF 01 00 00 00 21 60 D4 00 00 00 ..îÿÿÿ....!`Ô...
0000001D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0000001E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0000001F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 55 AA ..............Uª
000000200 45 46 49 20 50 41 52 54 00 00 01 00 5C 00 00 00 EFI PART....\...
000000210 7E 77 0D 41 00 00 00 00 01 00 00 00 00 00 00 00 ~w.A............
000000220 FF 23 B7 03 00 00 00 00 22 00 00 00 00 00 00 00 ÿ#·.....".......
000000230 DE 23 B7 03 00 00 00 00 67 16 3E 20 F3 01 66 45 Þ#·.....g.> ó.fE
000000240 AB A1 A9 42 D6 4B 00 4F 02 00 00 00 00 00 00 00 «¡©BÖK.O........
000000250 80 00 00 00 80 00 00 00 A2 4C 90 BD 00 00 00 00 ........¢L.½....
000000260 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000000270 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000000280 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000000290 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0000002A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0000002B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0000002C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0000002D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0000002E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
```



```
0000003E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0000003F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000000400 00 33 19 5B 78 FC CD 40 80 02 E8 6C 45 58 0B 47 .3.[xüÍ@..èlEX.G
000000410 B9 CD 58 93 CB C9 A5 46 96 23 E6 A3 E7 5E 37 28 ¹ÍX.ËÉ¥F.#æ£çˆ7(
000000420 22 00 00 00 00 00 00 00 21 08 00 00 00 00 00 00 ".......!.......
000000430 00 00 00 00 00 00 00 00 70 00 72 00 69 00 6D 00 ........p.r.i.m.
000000440 61 00 72 00 79 00 00 00 00 00 00 00 00 00 00 00 a.r.y...........
000000450 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000000460 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000000470 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000000480 53 B3 54 2E 71 12 42 48 80 6F E4 36 D6 AF 69 85 S³T.q.BH.oä6Ö¯i.
000000490 BB A5 A0 E9 08 27 E6 4D 96 C4 31 91 8A 67 F0 EC »¥ é.'æM.Ä1..gðì
0000004A0 22 08 00 00 00 00 00 00 21 28 00 00 00 00 00 00 ".......!(......
0000004B0 00 00 00 00 00 00 00 00 70 00 72 00 69 00 6D 00 ........p.r.i.m.
0000004C0 61 00 72 00 79 00 00 00 00 00 00 00 00 00 00 00 a.r.y...........
0000004D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0000004E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0000004F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000000500 A2 A0 D0 EB E5 B9 33 44 87 C0 68 B6 B7 26 99 C7 ¢ Ðëå¹3D.Àh¶·&.Ç
000000510 14 89 A9 87 D8 06 E5 41 91 6B DA 91 90 19 A4 AC ..©.Ø.åA.kÚ...¤¬
000000520 00 40 04 00 00 00 00 00 FF 4F 04 00 00 00 00 00 .@......ÿO......
000000530 04 00 00 00 00 00 00 00 62 00 6F 00 6F 00 74 00 ........b.o.o.t.
000000540 00 00 72 00 79 00 00 00 00 00 00 00 00 00 00 00 ..r.y...........
000000550 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000000560 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000000570 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000000580 AF 3D C6 0F 83 84 72 47 8E 79 3D 69 D8 47 7D E4 ¯=Æ...rG.y=iØG}ä
000000590 63 2D 6F 88 B5 02 F2 49 95 AC 06 0A B0 C0 76 C7 c-o.µ.òI.¬..°ÀvÇ
0000005A0 00 60 04 00 00 00 00 00 FF 5F D4 00 00 00 00 00 .`......ÿ_Ô.....
0000005B0 00 00 00 00 00 00 00 00 72 00 6F 00 6F 00 74 00 ........r.o.o.t.
0000005C0 00 00 72 00 79 00 00 00 00 00 00 00 00 00 00 00 ..r.y...........
0000005D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0000005E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0000005F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000000600 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000000610 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
```

SD Card



Disks

2.0 TB Disk
CT2000P2SSD8

512 GB Disk
INTEL SSDPEKKW512G8

2.0 TB Disk
INTEL SSDPEKNW020T8

14 TB Hard Disk
TOSHIBA MG08ACA14TE

32 GB Drive
Mass Storage Device

32 GB Drive
/dev/sdb

Model	Mass Storage Device (1.00)
Size	32 GB (31,914,983,424 bytes)
Partitioning	GUID Partition Table
Serial Number	121220160204

Volumes

Partition 1... 1.0 MB Un...
Partition 2... 4.2 MB Un...
Free Space 3.1 MB
boot Partition 3:... 136 MB FAT
Free Space 2.1 MB
root Partition 4: root 7.0 GB Ext4
Free Space 25 GB

Size	1.0 MB (1,048,576 bytes)
Device	/dev/sdb1
Partition Type	5b193300-fc78-40cd-8002-e86c45580b47
Contents	Unknown

U-Boot SPL
1.0 MB
RAW

OpenSBI + U-Boot
(payload)
4.2 MB, RAW

/boot
136 MB
FAT16

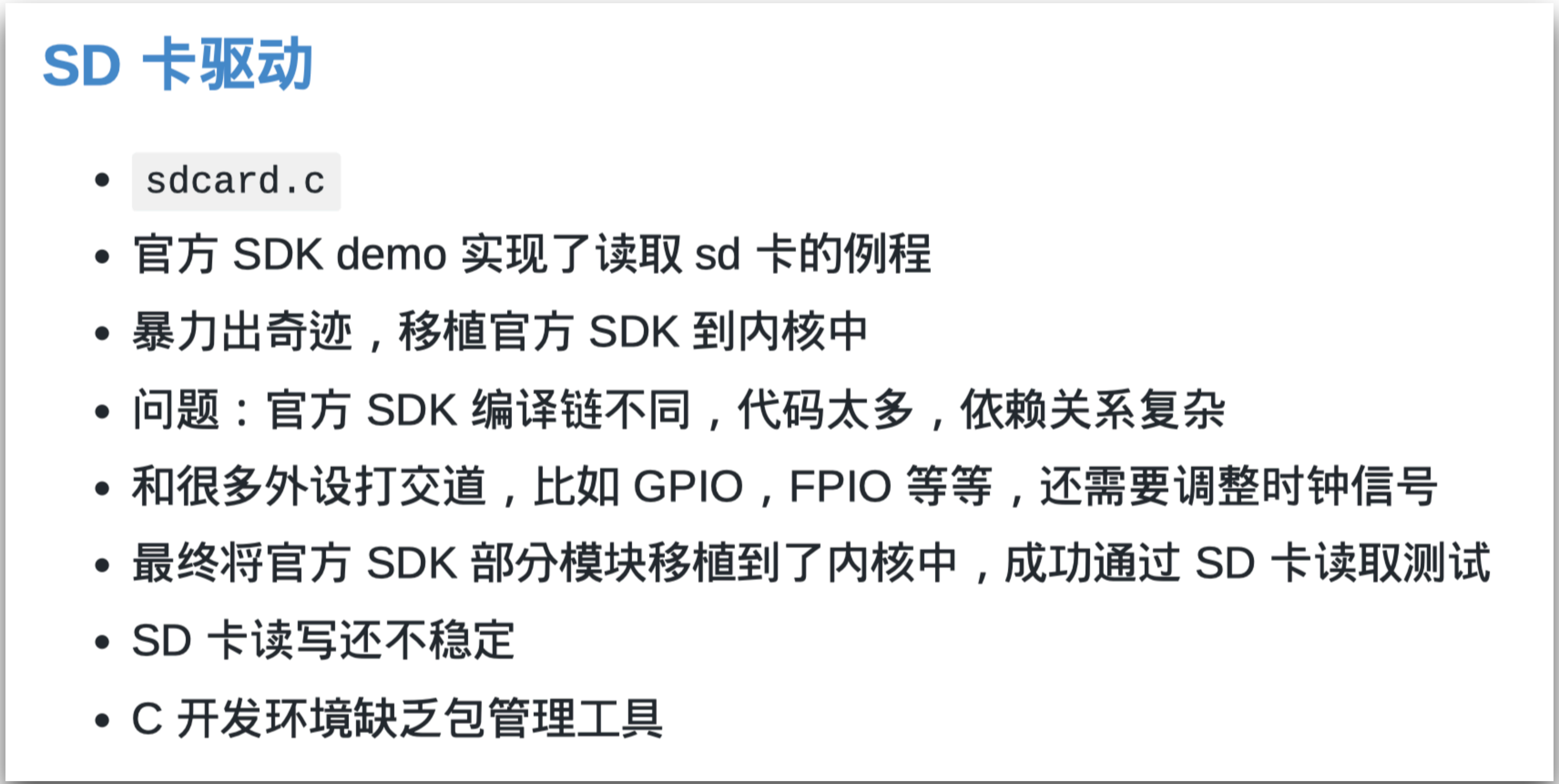
/root
7.0 GB
ext4

我做的一些尝试

- 有两个在k210上的参考，没有太多可以借鉴的地方
  - https://github.com/HUST-OS/xv6-k210
  - https://github.com/NKU-EmbeddedSystem/riscv64-ucore
- 硬件不一样，k210的流程：FPIOA (Field Programmable Input and Output Array) —> GPIO —> SPI
- FU740没有这个FPIOA
- 另外他们的驱动代码都是抄k210 SDK的，而且也有问题
- 我还尝试了U-Boot和Linux中的 mmc_spi 驱动，但依赖太多了

看一下这个人怎么做的

xv6-k210



SD 卡驱动

- `sdcard.c`
- 官方 SDK demo 实现了读取 sd 卡的例程
- 暴力出奇迹，移植官方 SDK 到内核中
- 问题：官方 SDK 编译链不同，代码太多，依赖关系复杂
- 和很多外设打交道，比如 GPIO，FPIO 等等，还需要调整时钟信号
- 最终将官方 SDK 部分模块移植到了内核中，成功通过 SD 卡读取测试
- SD 卡读写还不稳定
- C 开发环境缺乏包管理工具

一些要注意的地方

- FU740不支持修改页表项中的 A 和 D，所以要手动初始化的时候都设为1
- 修改时钟频率，和QEMU不同（会影响get_time( )的结果)
- 1+4 cores (mhartid=0 是小核S7，其他是大核U7)
  - S7只有M和U态，RV64IMAC
  - U7 支持 RV64GC
- 不支持 Misaligned load/store，所以要手动对齐到8 Bytes

Table 7: *U74 Feature Set*

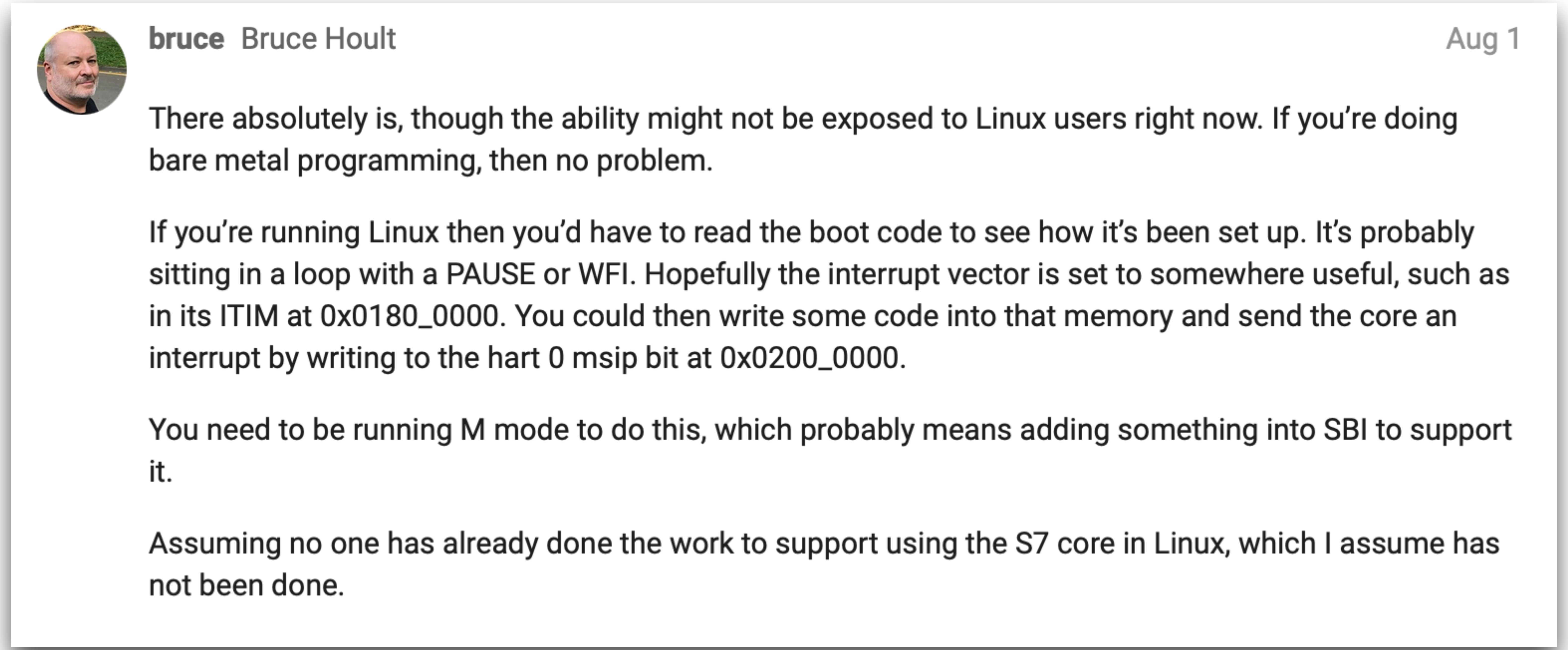
Feature	Description
ISA	RV64GC
SiFive Custom Instruction Extension (SCIE)	Not Present
Modes	Machine mode, user mode, supervisor mode
L1 Instruction Cache	32 KiB 4-way instruction cache
L1 Data Cache	32 KiB 8-way data cache
L2 Cache	2 MiB 16-way L2 cache with 4 banks
ECC Support	Single error correction, double error detection on the data cache and L2 cache.
Fast I/O	Present
Physical Memory Protection	8 regions with a granularity of 4096 bytes.
Memory Management Unit	Sv39 virtual memory support with fully-associative 40-entry L1 Data and Instruction TLBs, and a direct-mapped 512-entry L2 TLB.

Table 2: *S7 Feature Set*

Feature	Description
ISA	RV64IMAC
SiFive Custom Instruction Extension (SCIE)	Not Present
Modes	Machine mode, user mode
L1 Instruction Cache	16 KiB 2-way instruction cache
Data Tightly-Integrated Memory (DTIM)	8 KiB DTIM
L2 Cache	2 MiB 16-way L2 cache with 4 banks
ECC Support	Single error correction, double error detection on the DTIM and L2 cache.
Fast I/O	Present
Physical Memory Protection	8 regions with a granularity of 64 bytes.

关于利用S7 core

https://forums.sifive.com/t/how-about-the-s7-core/5028



bruce Bruce Hoult Aug 1

There absolutely is, though the ability might not be exposed to Linux users right now. If you're doing bare metal programming, then no problem.

If you're running Linux then you'd have to read the boot code to see how it's been set up. It's probably sitting in a loop with a PAUSE or WFI. Hopefully the interrupt vector is set to somewhere useful, such as in its ITIM at 0x0180_0000. You could then write some code into that memory and send the core an interrupt by writing to the hart 0 msip bit at 0x0200_0000.

You need to be running M mode to do this, which probably means adding something into SBI to support it.

Assuming no one has already done the work to support using the S7 core in Linux, which I assume has not been done.

总结

我的想法

- 比起上学期，这次的代码量不大，但是研究的资料远超上学期
- 虽然驱动没有跑通，但是学习到了很多，还看了很多U-Boot的代码
  - 驱动的工作量很大，又很繁琐，还和硬件关系很大
- 有前人引路很重要
  - 我研究了很久的启动流程，我去教之后的同学的时候，他很快就跑起来了

原来这么方便啊，感谢学长！

Thanks

2021. 11. 4