УПРАЖНЕНИЕ №7

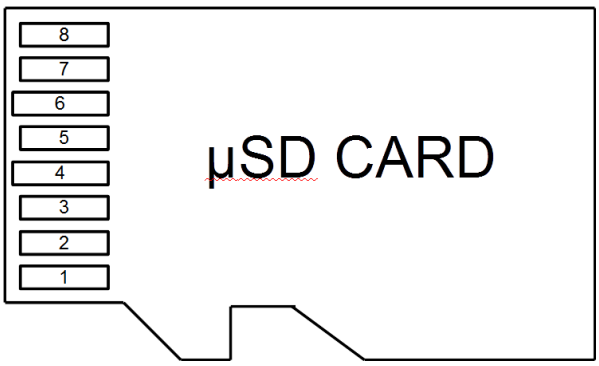
Работа с ARM Cortex-M4 базиран микроконтролер LM4F232. Запис и четене от SD карта.

I. Теоретична постановка.

SD (Secure Digital) картите са памети, проектирани за първи път през 1999 от фирмите Toshiba, SanDisk, Matsushita. Използват се за съхранение потребителски данни в множество портативни устройства. Произвеждат се в три различни корпуса: оригинален (9 извода / 25 MHz), мини (11 извода / 50 MHz), микро (8 / 50 MHz). В зависимост от капацитета на използваната флаш памет биват 4 вида: стандартен (SD Standard Capacity) до 2 GB, висок (SD High Capacity) до 32 GB, разширен (SD eXtended Capacity) до 2 TB, входно/изходен (SD Input/Output) – вместо флаш памет, в слота се включват GPS, модеми, FM радио, RFID четец, Wi-Fi модули и други.

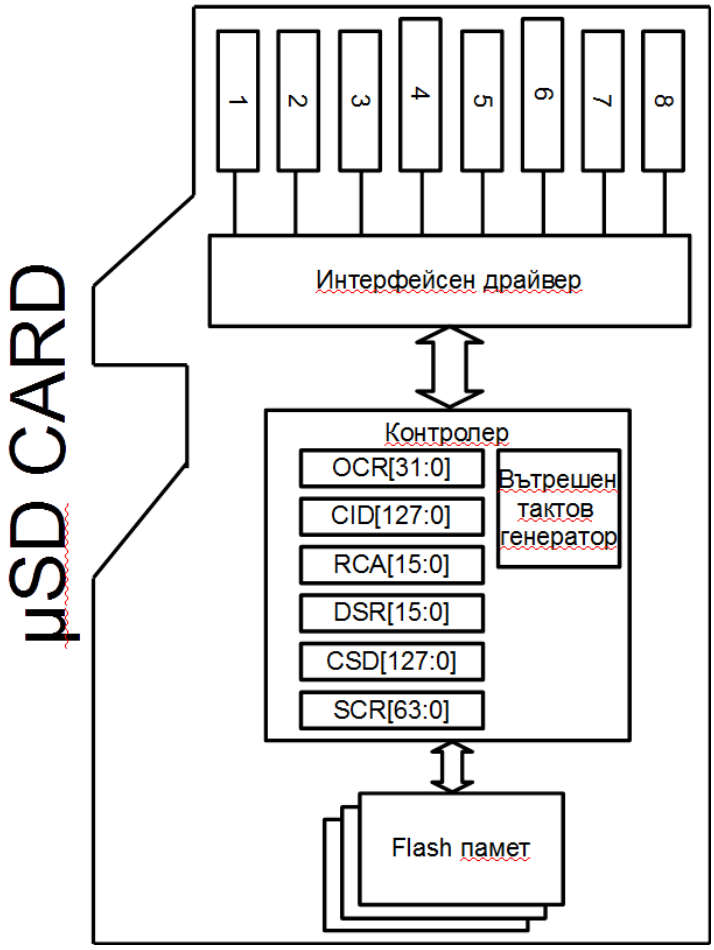
На **фиг. 1** са показани изводите и техните имена на микро SD карта. µSD картата може да използва 2 интерфейса – 4-битов паралелен или SPI (режим 0). Издърпващи резистори на DAT0/DAT1/DAT2/DAT3/CMD са желателни. Тяхната стойност е от 10 ÷ 100 kΩ. Те ще елиминират възможността извод, конфигуриран като вход, да остане плаващ и да увеличи консумацията на картата (изброените изводи са входове при първоначалното включване). Допълнително има механични ключове за защита от запис и детектиране за включена карта, които е желателно да бъдат свързани към управляващия микроконтролер.

Извод	Паралелен интерфейс	SPI интерфейс
8	DAT1	
7	DAT0	DO (MISO)
6	VSS	
5	CLK	SCLK (SCK)
4	VDD	
3	CMD	DI (MOSI)
2	DAT3	CS (SS)
1	DAT2	



Фиг. 1 – Разположение на изводите на микро SD карта.

На **фиг. 2** е показана блоковата схема на µSD. От нея се вижда, че има контролер, който приема команди по интерфейса и осъществява записа и четенето на данни от флаш паметта. Логиката се захранва с 2.7 ÷ 3.6 V, а консумацията при запис може да достигне десетки милиампери. Достъпът до паметта става с групи от байтове, наречени блок. Размерът на блока по подразбиране е 512 байта. Контролерът поддържа команда, с която може да се зададе друг размер на блока (чрез запис в CSD регистъра). Някой карти поддържат блокове от 1 ÷ 2048 байта, а други само 512/1024/2048.

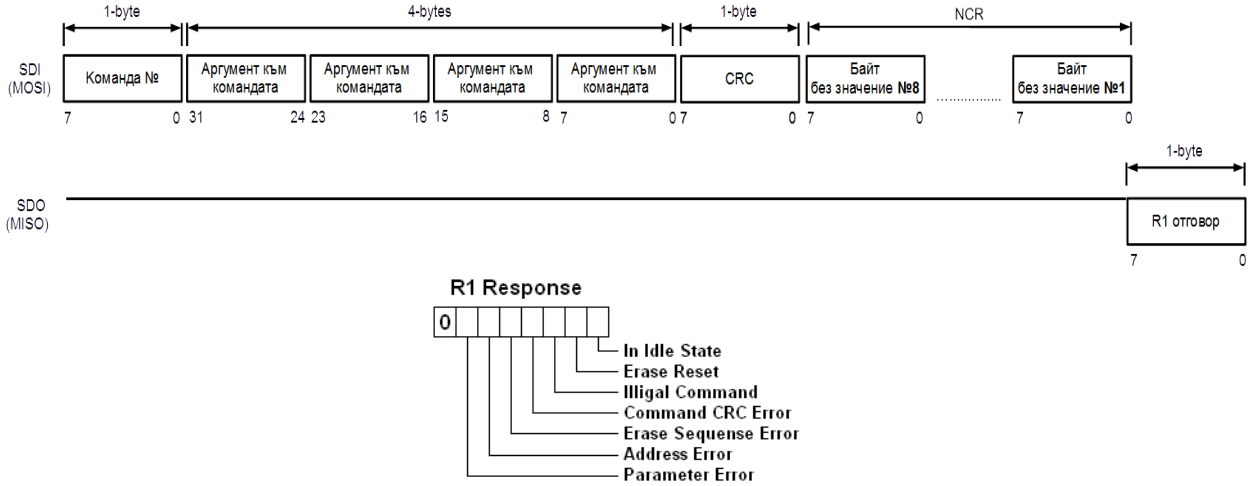


Фиг. 2 – Блокова схема на микро SD карта.

При подаване на захранване към картата (чрез вкарването ѝ в SD цокъла) по подразбиране тя е конфигурирана за работа с паралелен интерфейс. За да се избере SPI трябва DI (MOSI) и CS (SS) да се задържат в логическа единица, а SCLK трябва да генерира 74 или повече импулса с честота 100 kHz или 400 kHz. След това контролерът вече ще приема команди по SPI, като първата от тях трябва да е CMD0 (рестарт).

Подаването на команди към картата има форматът, показан на **фиг. 3**. Първо се изпраща номер на командата, след това аргумент, CRC байт за проверка и накрая от 0 ÷ 8 байта време (наречено NCR), в което контролерът на картата трябва да отговори. В зависимост от изпратеният номер на командата, контролерът връща като отговор различен брой байтове, чието съдържание обикновено са статус битове. Примерни видове отговори са R1(**фиг. 3**), R2, R3 и R7.

Някои команди отнемат повече време от NCR. Тогава се изпраща отговор R1b (R1 + busy flag), който представлява R1 отговор, последван от флаг за заето устройство, който в случая се реализира с SDO (MISO) = логическа 0 (т.е. предават се постоянно данни 0x00). Микроконтролерът трябва да изчака, докато не получи 0xFF от контролера на SD картата, което означава, че командата е била обработена. Списък с командите е даден на **фиг. 4**.



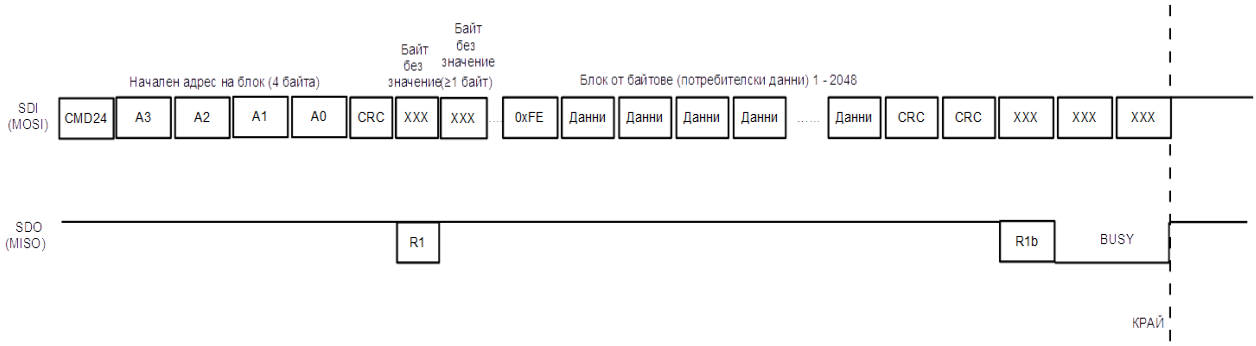
Фиг. 3 – Комуникация по SPI с SD карта.

Command Index	Argument	Response	Data	Abbreviation	Description
CMD0	None(0)	R1	No	GO_IDLE_STATE	Software reset.
CMD1	None(0)	R1	No	SEND_OP_COND	Initiate initialization process.
ACMD41(*1)	*2	R1	No	APP_SEND_OP_COND	For only SDC. Initiate initialization process.
CMD8	*3	R7	No	SEND_IF_COND	For only SDC V2. Check voltage range.
CMD9	None(0)	R1	Yes	SEND_CSD	Read CSD register.
CMD10	None(0)	R1	Yes	SEND_CID	Read CID register.
CMD12	None(0)	R1b	No	STOP_TRANSMISSION	Stop to read data.
CMD16	Block length[31:0]	R1	No	SET_BLOCKLEN	Change R/W block size.
CMD17	Address[31:0]	R1	Yes	READ_SINGLE_BLOCK	Read a block.
CMD18	Address[31:0]	R1	Yes	READ_MULTIPLE_BLOCK	Read multiple blocks.
CMD23	Number of blocks[15:0]	R1	No	SET_BLOCK_COUNT	For only MMC. Define number of blocks to transfer with next multi-block read/write command.
ACMD23(*1)	Number of blocks[22:0]	R1	No	SET_WR_BLOCK_ERASE_COUNT	For only SDC. Define number of blocks to pre-erase with next multi-block write command.
CMD24	Address[31:0]	R1	Yes	WRITE_BLOCK	Write a block.
CMD25	Address[31:0]	R1	Yes	WRITE_MULTIPLE_BLOCK	Write multiple blocks.
CMD55(*1)	None(0)	R1	No	APP_CMD	Leading command of ACMD<n> command.
CMD58	None(0)	R3	No	READ_OCR	Read OCR.
*1:ACMD<n> means a command sequense of CMD55-CMD<n>.					
*2: Rsv(0)[31], HCS[30], Rsv(0)[29:0]					
*3: Rsv(0)[31:12], Supply Voltage(1)[11:8], Check Pattern(0xAA)[7:0]					

Фиг. 4 – Видове команди, приемани от SD карта.

Инициализацията продължава с команда CMD8 и аргумент 0x1AA за прочитане диапазона на захранващото напрежение (SD карти версия 2 → 2.7 ÷ 3.6 V). Тази стъпка е задължителна за карти с голям капацитет. Следва подаването на команди CMD55 (специална) и CMD41 с аргумент 0x4000.0000, което инициализира и подготвя картата за работа.

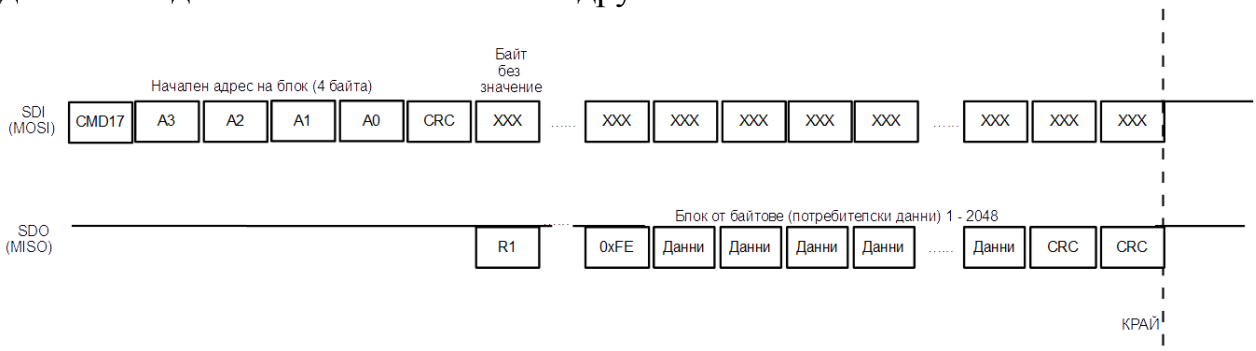
Запис в картата е показан на **фиг. 5**. Започва се с изпращане на команда CMD24 с аргумент число, указващо адреса на първия байт от блока, в който ще се записва. Изпраща се един байт с CRC. Получава се R1 отговор, след което се изчаква пауза от ≥ 1 байт. След това се изпраща т.нар. даннов символ (data token), представляващ байт 0xFE. След data token-а се изпращат байтовете на блока, чийто брой варира в зависимост от типа на картата (1 ÷ 2048 байта). Накрая се изпращат и 2 байта CRC към блока (в SPI режим CRC байтовете не се проверяват от SD по подразбиране, но тази функция може да се включи с CMD59; изключение правят команди CMD0 и CMD8, които винаги трябва да са последвани от валиден CRC байт). След двата CRC байта SD картата отговаря с R1b байт, като busy флагът е държан в нула, докато всички байтове от блока са записани. Последното може да отнеме десетки милисекунди, затова е предвидена команда за запис на няколко блока наведнъж (CMD25).



Фиг. 5 – Запис на блок в SD карта.

Четене от картата е показано на **фиг. 6**. Започва се с изпращане на командата CMD17 с аргумент число, указващо адреса на първия байт от блока, от който ще четем. Изпраща се един байт с CRC. Приема се R1 отговор. Изчакват се няколко байта (0xFF), докато пристигне данновия символ (0xFE). Когато той пристигне, следват прочетените данни от блока. В края на прочетените данни има два байта CRC, които могат да бъдат игнорирани в SPI режим. С тях приключва трансферът. Четенето може да стане на няколко блока с командата CMD18.

Файлова система се използва с различните видове памети. Файловата система контролира съхранението и извличането на данните от паметта. С помощта на файловата система данните се разделят и идентифицират лесно, което позволява по-добро съхранение на данни от различен тип. Без файлова система не бихме могли да кажем докъде се съхраняват байтовете на едни данни и къде започват байтовете на други.



Фиг. 6 – Четене на блок в SD карта.

При използване на файлова система данните (байтовете) се групират във логически единици, наречени файлове. Файловете се разполагат на различни адреси в паметта. Възможно е един файл да е разположен в няколко блока на различни адреси. Когато изтрием този файл, освободените блокове могат да се използват за други файлове. След дълга употреба на паметта е възможно голямо „накъсване“ на файловете по различни адреси. Тогава се казва, че

паметта е фрагментирана. Достъпът до файл във фрагментирана памет е по-бавен от достъп до файл, разположен на последователни адреси. Затова паметта трябва да се дефрагментира периодично.
Използването на файлова система води до намаляване на полезния размер на паметта. Във всеки един файл и в началните адреси от паметта има мета-данни, използвани само от файловата система. Тези данни не са потребителски. Въпреки това използването на файлова система е желателно, тъй като логическото разделяне на файлове води до улеснен достъп до информацията. В резултат програмата, използваща тази информация, не трябва да имплементира алгоритъм за достъп до паметта (това вече е направено от файловата система с функции като fopen, fclose, fseek и т.н.). Допълнително файловата система имплементира проверка за грешки (error correction), back-up на данните и контрол на достъпа до файловете.
Съществуват различни видове файлови системи. MMC/SD спецификациите препоръчват следните файлови системи:

- FAT12 за карти с обем ≤64MB;
- FAT16 при 128 MB ÷ 2 GB;
- FAT32 при 4 GB ÷ 32 GB;
- exFAT при 64 GB ÷ 2TB.

II. Литература.

[1] http://elm-chan.org/docs/mmc/mmc_e.html
[2] http://www.dejazzer.com/ee379/lecture_notes/lec12_sd_card.pdf
[3] SD Specifications, Part 1, Physical Layer, Simplified Specification, Version 2.00
[4] SD Specifications, Part E1, SDIO Simplified Specification, Version 2.00, February 8, 2007
[5] Samsung SD & MicroSD Card product family SDA 3.0 specification compliant-Up to High Speed mode
[6] TOSHIBA SD Card Specification
[7] https://www.sdcard.org
[8] AN10911 SD(HC)-memory card and MMC interface conditioning, NXP, 2013

III. Задачи за изпълнение.

1. Да се разучи блоковата схема на микроконтролера Stellaris LM4F232H5QD

2. Да се разучи принципната схема на макета Stellaris EKK-LM4F232

3. Да се отвори проекта sd_card_lowlevel със средата Keil µVision.
Project → Open Project → директория sd_card_lowlevel → sd_card.uvproj

4. Да се попълнят празните места в програмата, която осъществява запис и четене на 512 байта без използване на файлова система.

Задължително направете записа от стартов блок №2048! В противен случай може да изтриете file allocation таблицата, която се използва от файловата система във следващата задача!
Ако все пак това стане, помолете ръководителя на упражнението да ви форматира картата.

5. Да се отвори проекта sd_card_fat (работа със SD карта, форматирана с FAT12/FAT16/FAT32 файлова система). Да се реализира създаване и запис на текстови файл с произволен стринг.

Project → Open Project → директория sd_card_fat → sd_card.uvproj

Отворете sd_card_fat/fatfs/doc/00index_e.html и вижте описанието на достъпните за вас функции. Те са изброени в секцията Application Interface. Трябва да се извикат функциите f_open( ), f_write( ), f_close( ), както сте правили в курса ПИК.

6. Да се провери с SD-четец дали записът на файла е бил успешен.