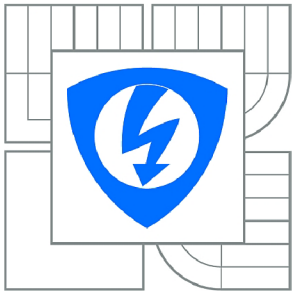
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

IMPLEMENTACE ČTENÍ A ZÁZNAMU NA SD KARTU

IMPLEMENTATION OF FUNCTIONS FOR WRITING/READING ON SD CARDS

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — JAROSLAV BAŠUS

AUTHOR

VEDOUCÍ PRÁCE — Ing. ONDŘEJ KRAJSA

SUPERVISOR

BRNO 2011

Anotace

Cílem bakalářské práce je vytvořit co nejvíce nezávislou hardwarovou knihovnu pro čtení a zápis dat na SD kartu v režimu SPI, který daný hardware podporuje. V první část je uveden popis komunikace s SD kartou v režimu SPI na fyzické vrstvě. Dále je uvedena komunikace s několika typy hardwarových skupin zařízení megaAVR a AVRXMEGA.

Klíčová slova:
SD karta, režim SPI, čtení dat, zápis dat, megaAVR, AVRXMEGA

Abstract

The aim of this thesis is to create as many hardware independent library for reading and writing data on an SD card in SPI mode and for hardware which supports SPI mode. The first part is a description of communication with the SD in SPI mode. It is set to communicate with several types of hardware devices joining groups microcontrollers megaAVR and AVRXMEGA.

Keywords:
SD card, SPI mode, Data read, Data write, megaAVR, AVRXMEGA

BAŠUS, J. *Implementace čtení a záznamu na SD kartu.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2011. 37 s. Vedoucí bakalářské práce Ing. Ondřej Krajsa.

Poděkování

Děkuji vedoucímu práce Ing. Ondřeji Krajsovi za velmi užitečnou metodickou pomoc a cenné rady při zpracování bakalářské práce.

V Brně dne …….. .....

podpis autora

Obsah

Úvod

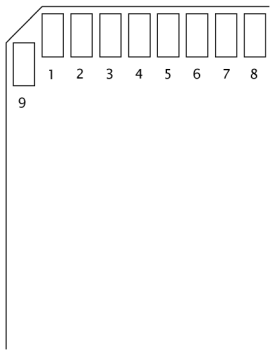
Dokument se zabývá implementací základních funkcí pro inicializaci, čtení a zápis dat na SD kartu pro skupiny mikrokontrolérů megaAVR a AVRXMEGA. V první části dokument popisuje teoretický rozbor problematiky obsluhy SD karty a mikrokontroléru v režimu SPI. V další části je uvedeno nastavení SD karty i mikrokontroléru pro vzájemnou komunikaci.

1 Standart SD

Secure digital – SD karta je standart pro přenosné paměťové úložiště navržené a licencované organizací SD card association. Pro její malé nároky na spotřebu energie, malé rozměry, jednoduchost a v dnešní době i velké kapacitě se karta velmi rozšířila do spotřební elektroniky jako jsou digitální fotoaparáty, kamery, PDA a přenosná hudební zařízení.

1.1 Režim SPI

Režim SPI je sekundární kompatibilní synchronní komunikační protokol paměťové karty SD navržený pro komunikaci přes SPI kanál pro propojení periferních zařízení s mikrokontrolérem. Většina moderních mikrokontrolérů SPI podporuje. Toto rozhraní je aktivováno během prvního resetovacího příkazu po přivedení napětí na kartu a není možné ho změnit dokud je karta zapnutá. Význam jednotlivých pinů na kartě (obrázek Obr. 1.1: Rozložení pinů na SD kartě) je znázorněn v Tab. 1.2: popis jednotlivých pinů.



Obr 1.1: Rozložení pinů na SD kartě

Tab. 1.2: Popis jednotlivých pinů

Pin	Název	Funkce (SPI)
1	CS	Chip select
2	DataIn	Vstup pro data
3	VSS1	Zem
4	VDD	Napětí
5	CLK	Hodinové pulzy
6	VSS2	Zem
7	DataOut	Výstup pro data
8	IRQ	Nevyužito
9	NC	Nevyužito

Standart SPI definuje pouze fyzickou vrstvu, ne kompletní datový přenos. Implementace SD karty v SPI režimu využívá stejné příkazy jako v režimu SD.

Pro režim SPI se využívají 4 signály uvedeny v následující tabulce:

Tab. 1.3: Signály SPI

CS	Chip select
CLK	Hodinový signál
DataIn	Data na kartu
DataOut	Data z karty

Identifikace karty a adresování jsou prováděny signálem Chip select (CS). Pro každý příkaz je karta vybrána nastavením signálu CS na nulu. Signál CS musí být nastaven po celou dobu komunikace (příkaz, odpověď, data). Jedinou výjimkou, kdy může být signál CS odpojen, je při programování dat na kartu.

1.2 Sběrnice SPI

Oproti kanálu SD, který je založen na příkazu a toku dat zahájeného start bitem a ukončeného stop bitem, kanál SPI komunikuje po bytech.
Podobně jako u režimu SD, SPI zprávy se skládají z příkazu, odezvy a datového bloku. Všechna komunikace mezi hostem a kartou je řízena uživatelem (master). Uživatel začíná každý přenos po sběrnici nastavením signálu CS(Chip select) na nulu (low)
Hlavní rozdíl v odezvě v režimu SPI oproti SD režimu:

- vybraná karta (CS) vždy odpovídá na příkaz (command)
- využití 8b a 16b struktury k odezvě
- pokud se karta setká s problémem při přenosu dat, odpoví chybovou odezvou (která nahradí očekávaná data)

1.3 Volba režimu SPI a inicializace

Po přivedení napájení je karta standardně inicializována v režimu SD. Do režimu SPI se karta přepne po přivedení signálu CS na nulu a poslání resetovacího příkazu (CMD0). Popis příkazového rámce (CMD) je popsán v kapitole 1.5 Komunikace pomocí SD protokolu. Příkaz CMD0 je statický příkaz a jeho 7 bitový kontrolní součet CRC je vždy 0x4Ah. Přidáním „1“ tzv. stop bitu na konec získáme poslední byte příkazového rámce jako 0x95h. Kompletní příkazový rámec bude vypadat následovně: 40h 00h 00h 00h 00 95h.

Dále pomocí příkazu CMD8 se ověřuje, zda je SD karta schopná provozu pod připojeným napětím. Pokud je karta provozu schopná zjišťuje uživatel z odpovědi na příkaz CMD8. Karta předpokládá, že napětí definované polem VHS v argumentu příkazu CMD8 je právě připojené napětí. Pouze jeden bit v poli VHS může být nastaven. Toho se využívá ke kontrole komunikace mezi uživatelem a kartou. Pokud karta odpoví, že daný příkaz (CMD8) je neplatný, znamená to, že karta je straší typ, který tento příkaz nepodporuje. Pokud karta příkaz CMD8 podporuje, nastaví příslušný příznak v odpověděním rámci. Když je v rámci bit VCA nastaven na nulu, karta nemůže pracovat se zvoleným napětím.

1.4 Ochrana dat při přenosu po sběrnici

Každý příkaz posílaný hostem na sběrnici je chráněn kontrolním součtem CRC. Při přepnutí do režimu SPI je tato kontrola primárně vypnuta a na kontrolní součet není brán zřetel. Ovšem při přepíná do režimu SPI pomocí příkazu CMD0 je stále kontrolní součet vyžadován, protože karta je v režimu SD. V režimu SPI se kontrolní součet CRC stále posílá u vyžadovaných rámců, ale není na něj brán zřetel a je ignorován při přijetí. Uživatel může kontrolní součet zapnut požitím příkazu CRC_ON_OFF (CMD59).

1.5 Komunikace pomocí SD protokolu

Komunikace probíhá pomocí command-response protokolu. Příkazy jsou posílány od uživatele, na které karta dopovídá pomocí odpovědních tokenů. Podle typu příkazu karta odpovídá pomocí tokenů R1, R2, R3 nebo datovými tokeny „token star block“, „stop tran“, případně chybovým tokenem „data error“. SD příkazy jsou posílány ve formě „CMDXX“ nebo „ACMDXX“, kde „CMD“ a „ACMD“ značí obecně příkaz a „XX“ dvoumístné číslo jednotlivého příkazu.

Příkazy jsou posílány pomocí příkazového rámce délky 6 bytů přes rozhraní SPI. Rámec vždy začíná sekvencí „01“ následovanou šesti bity označující číslo příkazu. Další čtyři byty je argument příkazu (první je poslán MSB). V posledním bytu je vyhrazeno 7 bitů pro kontrolní součet CRC a konečný stop bit „1“ je poslán jako poslední. Všechny byty jsou poslány uživatelem na port MOSI (DataIn). MSB se posílá jako první.V tabulce tab. 1.4 je znázorněna struktura rámce. V režimu SPI je kontrola CRC primárně vypnuta.

Tab. 1.4: Struktura rámce

1. byte			2. - 5. byte	6. byte	
0	1	Příkaz	Argument	CRC	1

1.6 Rozdělení paměťových částí

Základní jednotkou pro datový přenos je jeden byte. Každá datová operace definuje velikost bloku jako celočíselné násobky bytů. Pro blokově orientované příkazy se používají následující označení:

Blok – jednotka, která se vztahuje k blokově orientovaným příkazům pro čtení a zápis. Její velikost je počet bytů, které se přenáší, když je uživatelem poslán příkaz pro práci s blokem. Velikost bloku je buď programovatelná, nebo pevně daná. Informace o povolené velikosti je uložena v registru CSD.

U příkazů pro mazání není velikost jednotky stejná jako pro blokově orientované příkazy:

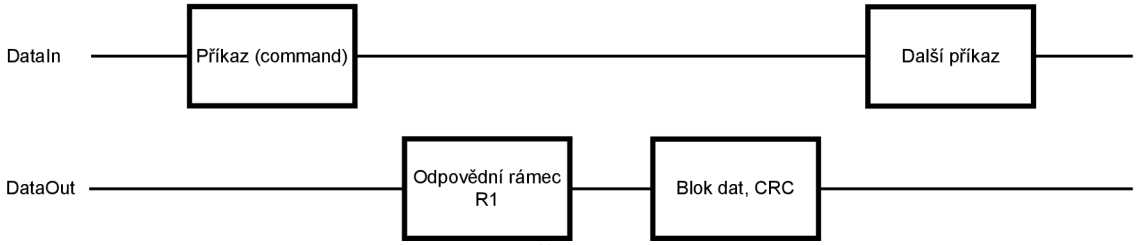
Sektor – jednotka, která se vztahuje k příkazům pro mazání. Její velikost je počet bloků, které jsou vymazány v jedné dávce. Velikost sektoru je pevně dána pro každé zařízení. Informace o velikosti sektoru (počet bloků) je uložena v registru CSD.

U zařízení, která využívají ochranu proti zápisu se používá následující označení:

WP skupina – je jednotka, která může mít individuální ochranu proti zápisu. Její velikost je definována jako počet skupin, které budou chráněny proti zápisu nebo vymazání. Velikost WP skupiny je pevně dána pro každé zařízení. Informace o velikosti je uložena v registru CSD.

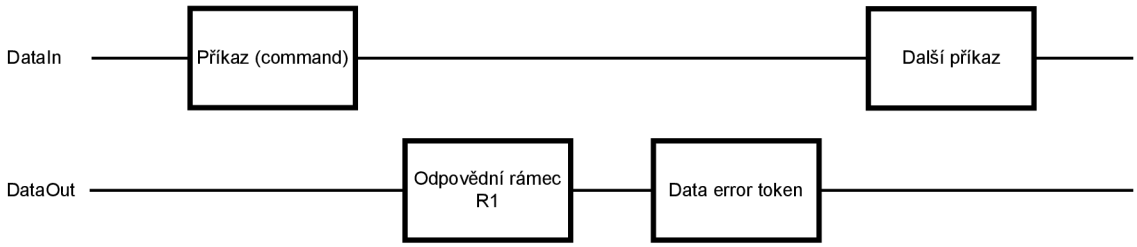
1.7 Čtení dat

Režim SPI podporuje jak jednotlivé čtení, tak čtení po blocích. Pro čtení dat je určen příkaz CMD17 nebo CMD18. Po přijetí příkazu pro čtení karta odpoví tokenem, který je následován blokem dat (viz obr. 1.5). Velikost jednoho bloku dat se definuje pomocí příkazu SET_BLOCKLEN (CMD16). Velikost bloku bývá standardně 512B. Maximální velikost bloku je uložena v registru CSD. Pro standardní kapacitu karet může být blok čten od jednotlivých bytů až po celý blok 512 B. U vysokokapacitních karet je možno pouze číst celé bloky 512 B.



Obr. 1.5: Čtení bloku dat

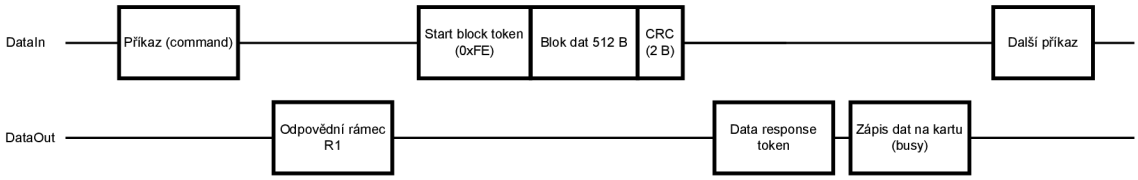
Adresa, odkud se mají číst data se posílá v příkazovém rámci. Je vždy nastavena na čtení začátku bloku. Není možné číst blok 512 B odkudkoliv. V případě, že se čtení dat nepovede, žádná data se nepřenáší a karta vyšle speciální chybové hlášení (token) na stranu hosta (viz obr. 1.6).



Obr. 1.6: Chyba při čtení dat

1.8 Zápis dat

Režim SPI umožňuje zápis jak po jednotlivých blocích, tak i po více blocích. Po přijetí příkazu pro zápis (příkaz CMD24, CMD25), karta odpoví a vyčká na blok dat poslaný od uživatele (viz obr 1.7).



Obr. 1.7: Zápis bloku dat

Každý blok dat má vlastní předponu (prefix 0xFEh), který určuje začátek bloku dat. Po přijetí dat karta odpoví, zda data byla přijata v pořádku. Pokud byla data přijata bez chyb, začnou se zapisovat na kartu. Po celou dobu zápisu dat na kartu posílá karta informaci o tom, že je zaneprázdněná. Jakmile je zápis dokončen, je na hostu, aby tuto událost zkontroloval pomocí příkazu CMD13. Pokud pří zápisu nastanou chyby(např.: zápis mimo paměť, ochrana proti zápisu) jsou odhaleny pouze při zápisu.

Zatímco je karta zaneprázdněna zápisem dat, můžeme odpojit signál CS, což nebude mít efekt na právě zapisovaná data. Karta uvolní linku DataOut a pokračuje se zápisem. Pokud karta bude znovu vybrána před dokončením zápisu dat, nastaví se linka DataOut na nízkou úroveň a odmítne všechny přicházející příkazy. Pokud resetujeme kartu(příkazem CMD0), ukončí se probíhající zapisování dat. To může vést ke ztrátě dat na kartě. Je na uživateli, aby se tomuto vyvaroval.

1.9 Čtení CID/CSD registrů

Čtení probíhá v režimu SPI jako příkaz pro čtení bloku, nikoliv jako v SD režimu, který posílá registr CID nebo CSD jako odpověď (token) na příkaz čtení. Karta odpoví start block tokenem (0xFEh), za kterým následuje blok dat dlouhý 16 bytů ukončený 16-ti bitovým CRC kontrolním součtem.

Pro čtení CSD registru slouží příkaz SEND_CSD (CMD9). Jako odpověď je poslán token R1 následovaný daty.

Pro čtení CID registru slouží příkaz SEND_CID (CMD10). Jako odpověď je poslán token R1 následovaný daty.

Registr CID

Registr „Card Identification“ má délku 128 bitů. Obsahuje identifikační údaje karty použity během identifikace karty. Každá karta má své jedinečně identifikační číslo.
Struktura CID registru je definována v následující tabulce:

Tab. 1.8: Obsah CID registru

Jméno	Pole	Velikost	CID pozice
Výrobní ID	MID	8	[127:120]
OEM	OID	16	[119:104]

jméno	PNM	40	[103:64]
verze	PRV	8	[63:56]
sériové číslo	PSN	32	[55:24]
rezervováno		4	[23:20]
datum výroby	MDT	12	[19:8]
kontrolní součet CRC7	CRC	7	[7:1]
nevyužito	-	1	[0:0]

Popis jednotlivých polí:

- **MID**

Osmi bitové číslo výrobce. Unikátní číslo výrobce je určováno licencí SD-3C, LLC.

- **OID**

2 znaky z ASCII kódu, které definují OEM nebo obsah karty (je-li používána jako ROM nebo FLASH). OID číslo je definováno licencí SD-3C, LLC.

- **PNM**

Jméno karty, obsahující 5 ASCII znaků

- **PRV**

Verze karty je obsažena ve dvou číslech po čtyřech bitech, reprezentovány jako „n.m“ přičemž „n“ je více významný nibble a „m“ je méně významný nibble.
Jako příklad PRV číslo „3.6“ odpovídá binárně: 0011 0110b

- **PSN**

Je sériové číslo dlouhé 32 bitů.

- **MDT**

Datum výroby je tvořeno dvěma hexadecimálními číslicemi. Jedno je osmi bitová reprezentace roku (y), další je čtyři bity dlouhá reprezentace měsíce (m).
Pole „m“ [11:8] je kód pro měsíc. 1 = leden
Pole „y” [19:12] je kód pro rok. 0 = 2000

- **CRC**

Představuje sedmi bitový kontrolní součet CRC7 pro obsah registru CID.

Registr CSD

Card-specific data registr obsahuje informace týkající se přístupu na kartu. CSD definuje datový formát. Programovatelná část registru je přístupná pomocí příkazu CMD27. Struktura CSD registru je definována v **tabulce CSD registr**. Význam značek v tabulce je: R = pouze pro čtení, W = přepisovatelný, W(1) = možný jeden zápis.

Struktura dat v CSD registru se liší v závislosti na verzi fyzické specifikace a kapacitě karty. Pole CSD_STRUCTURE v CSD registru naznačuje jeho strukturu. Tabulka 1.9 ukazuje číslo verze příslušné struktury CSD.

Tab. 1.9: Struktura CSD registru

CSD_STRUCTURE	CSD structure version	Verze fyzické specifikace/kapacita karty
0	CSD verze 1.0	Verze 1.01-1.10 Verze 2.00/Standardní kapacita
1	CSD verze 2.0	Verze 2.00/Vysoká kapacita
2-3	Rezervováno	

Základní přehled informací v CSD registru je v následující tabulce:

Tab 1.10: Obsah CSD registru

Jméno	Pole	Délka [b]	Typ	CSD-pozice
CSD structure	CSD_STRUCTURE	2	R	[127:126]
max. délka bloku dat při čtení	READ_BL_LEN	4	R	[83:80]
partial blocks for read allowed	READ_BL_PARTIAL	1	R	[79:79]
write block misalignment	WRITE_BLK_MISALIGN	1	R	[78:78]
read block misalignment	READ_BLK_MISALIGN	1	R	[77:77]
device size	C_SIZE	12	R	[73:62]
device size multiplier	C_SIZE_MULT	3	R	[49:47]
max. write data block length	WRITE_BL_LEN	4	R	[25:22]
partial blocks for write allowed	WRITE_BL_PARTIAL	1	R	[21:21]
File format group	FILE_FORMAT_GRP	1	R/W(1)	[15:15]
File format	FILE_FORMAT	2	R/W(1)	[11:10]

Popis jednotlivých polí:

- **READ_BL_LEN**

Maximální délka bloku dat při čtení je $2^{READ\_BL\_LEN}$. Maximální velikost bloku může být v rozmezí 512 – 2048 B. Velikost READ_BL_LEN je standardně rovna WRITE_BL_LEN. Přesný význam bitů je uveden v následující tabulce.

Tab. 1.11: Velikost bloku dat

Bit	Velikost bloku
0-8	Rezervováno
9	512 B
10	1012 B
11	2048 B
12-15	Rezervováno

- **READ_BL_PARTIAL**

Částečné čtení bloku je u SD karty vždy povoleno. Znamená to, že mohou být čteny data délky menší než je velikost bloku. Nejmenší velikost je jeden byte.

- **WRITE_BLK_MISALIGN**

Definuje, zda může být zapisovaný blok dat, definovaný jedním příkazem, rozdělen na více fyzických bloků v paměti SD karty.
Velikost bloku je definována v WRITE_BL_LEN. Pokud je WRITE_BLK_MISALIGN = 0, není povoleno rozdělit zapisovaná data. Při WRITE_BLK_MISALIGN = 1 je povoleno zapisovat na více fyzických bloků.

- **READ_BLK_MISALIGN**

Stejný význam jako WRITE_BLK_MISALIGN s rozdílem pro čtení dat.

- **C_SIZE**

Tento parametr se používá k výpočtu kapacity SD karty. Kapacita je určena pomocí C_SIZE, C_SIZE_MULT a READ_BL_LEN jako:

$$\text{Kapacita paměti} = \text{BLOCKNR} \cdot \text{BLOCK\_LEN}$$

kde

$\text{BLOCKNR} = (\text{C\_SIZE} + 1) \cdot \text{MULT}$

$\text{MULT} = 2^{\text{C\_SIZE\_MULT} + 2}$

$\text{BLOCK\_LEN} = 2^{\text{READ\_BL\_LEN}}$,

k indikaci karty o velikosti 2 GB musí být BLOCK_LEN 1024 bytů. Potom maximální kapacita se spočítá jako $4096 \cdot 512 \cdot 1024 = 2$ GB.

- **C_SIZE_MULT**

Tento parametr určuje činitel MULT pro počítání celkové velikosti SD karty. Je definován jako $\text{MULT} = 2^{\text{C\_SIZE\_MULT} + 2}$. Význam bitů je znázorněn v následující tabulce.

Tab. 1.12: Význam bitů v C_SIZE_MULT

C_SIZE_MULT	MULT
0	4
1	8
2	16
3	32
4	64
5	128
6	256
7	512

- **WRITE_BL_LEN**

Určuje maximální délku bloku dat při zápisu jako $2^{\text{WRITE\_BL\_LEN}}$. Maximální velikost bloku je v rozmezí 512-2048 B. Velikost 512 B je vždy podporována. Parametr WRITE_BL_LEN je vždy roven READ_BL_LEN. Význam hodnot je znázorněn v následující tabulce:

Tab. 1.13: Význam hodnot pole WRITE_BL_LEN

WRITE_BL_LEN	Délka bloku
0-8	rezervováno
9	512 B
10	1024 B
11	2048 B
12-15	rezervováno

- **FILE_FORMAT_GRP**

Indikuje zvolenou skupinu formátu souborů. Použití tohoto pole je zobrazeno v tabulce tab. 1.13.

- **FILE_FORMAT**

Slouží k identifikaci systému souborů na SD kartě. Význam hodnot je znázorněn v následující tabulce:

Tab. 1.14: Typ systémových souborů

FILE_FORMAT_GRP	FILE_FORMAT	typ
0	0	Systém souborů, jako pevný

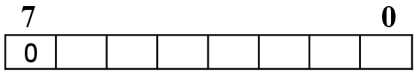
		disk s tabulkou oddílů
0	1	DOS FAT, pouze s bootovacím sektorem(bez tabulky oddílů)
0	2	Univerzální formát souborů
0	3	Ostatní/neznámý
1	0,1,2,3	Rezervováno

1.10 Odpovědní rámce

SD karta odpovídá na každý příkaz (CMD) specifickým odpovědním tokenem, který záleží na příkazu. Pro režim SPI jsou definovány tokeny R1, R2, R3.

Formát R1

Tento token je kartou posílán vždy po každém příkazu od uživatele s výjimkou příkazu SEND_STATUS. Token R1 má délku 1 byte a MSB je vždy nastaven na nulu. Ostatní bity indikují chybu, která je signalizována nastavením bitu na „1“. Struktura R1 je znázorněna na obrázku obr 1.15

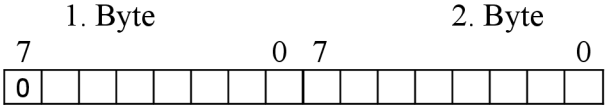


Obr. 1.15: Rámec R1

Význam bitů:
7 – vždy nastaven na „0“
6 – parametr error: argument příkazu(např.: adresa, délka bloku) byl mimo povolený rozsah katy
5 – address error: v příkazu byla použita adresa, která neodpovídá délce bloku
4 – erase sequence error: došlo k chybě v sekvenci příkazů pro mazání
3 – communication CRC error: kontrolní součet CRC posledního příkazu byl
2 – illegal command: byl detekován neplatný příkaz
1 – erase reset: mazací sekvence byla zrušena před dokončením, protože byl přijat příkaz out of erase
0 – in idle state: nečinný

Formát R2

Tento odpovědní token má délku 2 B a je odesílán jako odpověď na příkaz SEND_STATUS (CMD13). Formát je na obrázku obr 1.16.



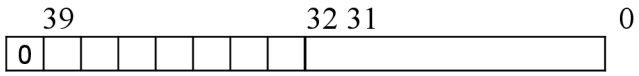
Obr. 1.16: Rámec R2

Význam bitů:
0 – Card is locked: je nastaven, pokud je karta zamčena uživatelem
1 – Wp erase skip | lock/unlock emd failed: tento bit má dva významy. Je nastaven, když se uživatel pokouší vymazat sektor chráněný proti zápisu, nebo zadá chybné heslo během procesu odemčení/zamčení
2 – Error: nastala obecná nebo neznámá chyba v průběhu operace
3 – CC error: chyba ve vnitřním řadiči karty
4 – Card ECC failed: vnitřní ECC bylo použito, ale nepodařilo se opravit data
5 – wp violation: příkaz zkoušel zapsat do bloku chráněného proti zápisu
6 – erase param: neplatný sektor nebo skupina pro vymazání
7 – out of range | csd overwrite
8 – in idle state
9 – erase reset
10 – illegal command
11 – com crc error
12 – erase sequence error
13 – address error
14 – parametr error

První byte je identický s R1

Formát R3

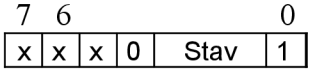
Tento odpovědní token je poslán kartou, pokud je přijat příkaz READ_OCR. Má délku 5 bytů, jak je ukázáno na obrázku obr. 1.17. Struktura prvního (MSB) bytu je shodná s R1. Zbylé čtyři byty obsahují registr OCR.



Obr 1.17: Rámec R2

Data Response Token

Každý zapsaný blok dat na kartu je potvrzen tzv. data response tokenem, který je dlouhý jeden byte a má následující formát:



Obr. 1.18: Data response token

Význam stavových biů:
010 – Data přijata
101 – Data odmítnuta z důvodu chybného kontrolního součtu CRC
110 – Data odmítnuta z důvodu chyby při zápisu

Při výskytu chyby může být použit příkaz ACMD22 pro zjištění správně zapsaných bloků dat.

Datové tokeny „token star block“ a „stop tran“

Příkazy pro čtení a zápis dat jsou spojeny s těmito tokeny. Data jsou posílána prostřednictvím datových tokenů. U všech dat je přenášen MSB jako první. Datové tokeny jsou dlouhé 4 až 515 bytů a mají následující formát:
Pro čtení jednoho bloku, zápis jednoho bloku a zápis více bloků:

- První byte: Start block

7							0
1	1	1	1	1	1	1	0

- Byty 2 – 513 (závisí na délce datového bloku): Uživatelská data
- Poslední dva byty: 16 bitů CRC

Pro zápis více bloků je token následující:

- První byte každého bloku:

Pokud jsou data k odeslání – Start block token

7							0
1	1	1	1	1	1	0	0

Pokud je přenos ukončen – Stop tran token

7							0
1	1	1	1	1	1	0	1

Tento formát je použit pouze při zápisu po více blocích. Pro ukončení čtení po více blocích je určen příkaz STOP_TRAN (CMD12).

Data error token

Pokud nastane chyba při čtení a karta nemůže poskytnout požadovaná data, vyšle se token „data eorror“. Tento token má velikost jeden byte v následujícím formátu:

7							0
0	0	0	0				

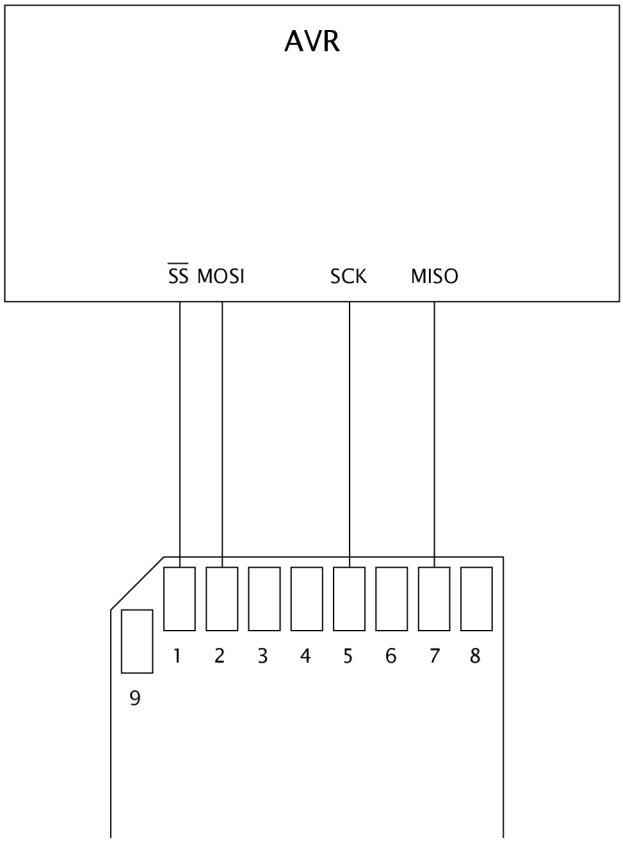
Obr. 1.19: Data error token

Význam bitů:
Bit 0 – Error
Bit 1 – CC Error
Bit 2 – Card ECC failed
Bit 3 – out of range

2 Komunikace s mikrokontroléry megaAVR a AVR XMEGA

2.1 Propojení mikrokontroléru a SD karty pomocí SPI

Systém propojení Master-Slave (uživatel a SD karta) se skládá ze dvou posuvných registrů a generátoru hodinových pulzů. Master začíná komunikovat po nastavení pinu SS (slave selecet) na nulu (u SD karty pin CS). Slave připraví data k odeslání pomocí svého posuvného registru a Master generuje hodinové pulzy na lince SCK (CLK na SD kartě). Data jsou vždy odeslána od zařízení pracující jako „master“ k „slave“ pomocí pinu MOSI (master out, slave in) a od zařízení „slave“ k „master“ pomocí pinu MISO(master in, slave out). Propojení je znázorněno na obrázku obr. 2.1. Po každém datovém paketu zařízení „master“ synchronizuje „slave“ nastavením pinu SS (slave select) na úroveň high.



Obr. 2.1: Propojení SD karty s mikrokontrolérem

2.2 Konfigurace jako master

Rozhraní SPI nemá automatickou kontrolu pinu SS. Musí být nastaven programově uživatelem před začátkem komunikace. Po tomto nastavení přesuneme jeden byte do „SPI data register“ a tím se spustí SPI generátor hodin. Hardware pošle osm bitů na zařízení pracující jako „slave“. Po přesunu jednoho bytu přes SPI se generátor hodin zastaví a nastaví příznak konec přenosu (end of transmissoin) SPIF. Pokud je povoleno

přerušení (SPIE) v registru SPCR, přijde žádost o přerušení. Master může pokračovat zapsáním dalšího bytu do registru SPDR, nebo poslat signál konec paketu nastavením signálu SS na úroveň high. Poslední přijatý byte je uložen ve vyrovnávací paměti (buffer register) pro pozdější použití.

2.3 Konfigurace jako slave

Rozhraní SPI setrvává v režimu spánku po celou dobu co je pin SS nastaven na úroveň high. V tomto případě můžou být data poslána do datového registru SPDR, ale nebudou poslána na sběrnici dokud nebude nastaven pin SS na úroveň „low“. Jakmile bude jeden byte kompletně přesunut, nastaví se příznak konec přenosu (end of transmission). Pokud je povoleno přerušení (SPIE) v registru SPCR(viz tab. 2.3), přijde žádost o přerušení. „Slave“ může pokračovat s umístěním nových dat, která mají být poslána do registru SPDR před čtením příchozích dat. Poslední přijatý byte je uložen ve vyrovnávací paměti (buffer register) pro pozdější použití.

Systém využívá jednu vyrovnávací paměť ve vysílacím směru a dvě vyrovnávací paměti ve směru přijímacím. To znamená, že byty, které mají být přeneseny, nemůžou být zapsány do SPI datového registru SPDR před tím, než je celý posouvací cyklus dokončen. Nicméně, když se data přijímají, musí být přijatý znak čten z SPI datového registru před tím, než je další znak kompletně odeslán. Jinak bude první byte ztracen.

2.4 Nastavení komunikace pomocí rozhraní SPI pro megaAVR

Mikrokontroléry megaAVR podporují rozhraní SPI. K tomuto rozhraní jsou určeny alternativní funkce portu B. SPI využívá z tohoto portu čtyři bity. Mikrokontroléry jsou rozděleny do čtyř skupin podle rozmístění komunikačních pinů na portu B. Rozdělení mikrokontroléru do skupin je popsáno v kapitole 2.5. Skupina 1 využívá piny uvedeny v tabulce tab. 2.2:

Tab. 2.2: Alternativní funkce portu B pro první skupinu

Pin	Alternativní funkce
PB4	MISO (SPI bus master input)
PB3	MOSI (SPI bus master output)
PB5	SCK (SPI bus seriál clock)
PB2	SS (SPI slave select input)

Skupina 2 využívá piny uvedeny v tabulce tab. 2.3

Tab. 2.3 Alternativní funkce portu B pro druhou skupinu

Pin	Alternativní funkce
PB6	MISO (SPI bus master input)
PB5	MOSI (SPI bus master output)
PB7	SCK (SPI bus seriál clock)
PB4	SS (SPI slave select input)

Skupina 3 využívá piny uvedeny v tabulce tab. 2.4

Tab. 2.4: Alternativní funkce portu B pro třetí skupinu

Pin	Alternativní funkce
PB3	MISO (SPI bus master input)
PB2	MOSI (SPI bus master output)
PB1	SCK (SPI bus seriál clock)
PB0	SS (SPI slave select input)

Skupina 4 využívá piny uvedeny v tabulce tab. 2.5

Tab. 2.5 Alternativní funkce portu B pro čtvrtou skupinu

Pin	Alternativní funkce
PB0	MISO (SPI bus master input)
PB1	MOSI (SPI bus master output)
PB7	SCK (SPI bus seriál clock)
PB3	SS (SPI slave select input)

Popis pinů:
MISO: SPI Master data input, slave data output pro kanál SPI. Když je zapnuto SPI jako master, tento pin je nastaven jako vstupní bez ohledu na nastavení DDB3. Pokud je SPI nastaveno jako slave, směr toku dat tohoto pinu se nastavuje pomocí DDB3.
MOSI: SPI Master data output, slave data input pro kanál SPI. Když je zapnuto SPI jako slave, tento pin je nastaven jako vstupní bez ohledu na nastavení DDB2. Pokud je SPI nastaveno jako master, směr toku dat je nastaven pomocí DDB2.
SCK: Master clock output, slave clock input pro kanál SPI. Když je SPI zapnuto jako slave, pin je nastaven jako vstupní, bez ohledu na nastavení DDB1. Když je SPI zapnuto jako master, směr toku je nastaven pomocí DDB1.
SS: Slave port select input. Pokud je SPI zapnuto jako slave, pin je nastaven jako vstupní bez ohledu na nastavení DDB0. Jako slave je SPI aktivováno při nastavení tohoto pinu no úroveň low. Při nastavení SPI jako master, směr toku dat je nastaven pomocí DDB0.

Samotné povolení režimu SPI na mikrokontroléru se provádí nastavením registru SPCR – SPI control register. Význam registru je zobrazen v tabulce Tab. 2.6: SPCR – SPI control register.

Tab. 2.6: SPCR – SPI control register

Bit	7	6	5	4	3	2	1	0
Zkratka	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0
Počáteční hodnota	0	0	0	0	0	0	0	0

Význam bitů:
7 - SPIE: SPI interrupt enable
Tento bit vyvolá přerušení při nastavení příznaku SPIF v registru SPSR a pokud je povoleno globální přerušení.
6 – SPE: SPI enable

Pokud je příznak SPE nastaven „1“, SPI je zapnuto. Tento bit musí být nastaven, aby bylo možné využít jakékoliv operace s SPI.

5 – DORD: Data order

Nastavením na „1“ znamená, že LSB je posílán jako první.
Nastavením na „0“ znamená, že MSB je posílán jako první.

4 – MSTR: Master/Slave Select

Tímto bitem se nastavuje master SPI mód, při nastavení na „0“ a slave SPI při nastavení na „1“.

3 – CPOL: Clock Polarity

Nastavením na „1“ je SCK nastaven na úroveň high při nečinném stavu. Nastavením na „0“ je SCK nastaven na úroveň low při nečinném stavu.

2 – CPHA: Clock Phase

Nastavením tohoto bitu se rozhoduje, zda se budou data vzorkovat na nástupnou nebo sestupnou hranu signálu SCK.

1, 0 – SPR1, SPR0: SPI Clock Rate Select 1 and 0

Tyto dva bity nastavují rychlost SCK nastaveného jako master. SPR1 a SPR0 nemají žádné efekt na straně slave. Vztah mezi SCK a taktovacím kmitočtem oscilátoru f_{osc} je znázorněn v následující tabulce:

Tab. 2.7: vztah mezi SCK a frekvencí oscilátoru

SPI2X	**SPR1**	**SPR0**	**frekvence SCK**
0	0	0	f_{osc} /4
0	0	1	f_{osc} /16
0	1	0	f_{osc} /64
0	1	1	f_{osc} /128
1	0	0	f_{osc} /2
1	0	1	f_{osc} /8
1	1	0	f_{osc} /32
1	1	1	f_{osc} /64

Abychom mohli zjistit, zda data, která jsme poslali do registru SPDR byla odeslána, musí se povolit příznak přerušení v registru SPSR – SPI status register, který je popsaný v následující tabulce:

Tab. 2.8: SPSR – SPI status registr

bit	7	6	5	4	3	2	1	0
zkratka	SPIF	WCOL	-	-	-	-	-	SPI2X
Počáteční hodnota	0	0	0	0	0	0	0	0

Popis bitů:

7 – SPIF: SPI Interrupt Flag

Pokud je sériový přenos dokončen (tj. přenos jednoho bytu pomocí registru SPDR), příznak SPIF je nastaven na „1“. Přerušení nastane pouze v případě, že je povolen příznak SPIE v registru SPCR a je povoleno globální přerušení. Příznak SPIF je nulován hardwarově při prvním čtení SPI status registru (SPSR), poté možný přístup k SPI data registru (SPDR).

6 – WCOL: Write COLlision Flag
Příznak je nastaven, pokud je registr SPDR přepsán během přenosu dat. Příznak WCOL (a příznak SPIF) jsou vynulovány při prvním čtení SPSR registru.
5:1 – Res: Reserved Bits
Tyto bity jsou rezervovány a vždy nastaveny na „0“.
0 – SPI2X: Double SPI Speed Bit
Při nastavení na logickou jedničku v „master“ módu bude rychlost přenosu po SPI (frekvence SCK) dvojnásobná. To znamená, že minimální perioda bude dvě hodinové periody CPU. Pokud je SPI nastaveno jako „slave“, zaručuje provoz pro frekvenci $f_{osc}/4$ nebo nižší.

Samotný přenos dat probíhá pomocí osmibitového posuvného registru **SPDR – SPI data registr** pro čtení i zápis. Je využit i při posílání řídících příkazů (CMD) a datových tokenů. Zapsání dat do registru vyvolá přenos dat. Čtení registru probíhá z vyrovnávací paměti posuvného registru.

Pro komunikaci pomocí rozhraní SPI jsme zvolili port B, protože jeho alternativní funkce toto rozhraní podporují. Zároveň umožňuje vstupní/výstupní komunikaci. Dále je potřeba rozlišit, který z daných pinů (MISO, MOSI, CLK, SS) má být vstupní, nebo výstupní. K tomuto účelu slouží směrový registr DDRB – port B data direction register. Význam bitů je znázorněn v následující tabulce:

Tab. 2.9: DDRB – Port B Data direction register

bit	7	6	5	4	3	2	1	0
zkratka	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0
Počáteční hodnota	0	0	0	0	0	0	0	0

Pin portu B bude sloužit jako výstupní, při nastavení daného bitu v registru DDRB na úroveň logické jedničky. Pokud má být daný pin portu B vstupní, nastaví se daný bit v registru DDRB na úroveň logické nuly.

2.5 Rozdělení mikrokontrolérů megaAVR do skupin

Pro univerzálnost jsou mikrokontroléry megaAVR rozděleny do skupin využívající stejné komunikační piny.

Do první skupiny jsou zahrnuty typy:
ATmega 8, 8L, 8A, 48, 88, 168, 48A, 48PA, 88A, 88PA, 168A, 168PA, 328, 328P, 48P, 88P, 168P.

Do druhé skupiny jsou zahrnuty typy:
ATmega 32,162,162V,8515,8515L, 8535, 8535L, 16, 16L, 164A, 164PA, 324A, 324PA, 644A, 644PA, 1284, 1284P, 164P/V, 324P/V, 644P/V.

Do třetí skupiny jsou zahrnuty typy:
ATmega 165P, 165PV, 165A, 165PA, 325PA, 3250PA, 3250A, 640, 645A, 645P, 6450A, 6450P, 325, 3250, 645, 6450, 16U4, 32U4, 8U2, 16U2, 1280, 1281, 2560, 25061, 169PA, 329A, 329PA, 649A, 649P, 3290A, 3290PA, 6490A, 6490P
AT90USB CAN32, AT90USB CAN64, AT90USB CAN128
AT90USB 649, AT90USB 647, AT90USB 1286, AT90USB 1287, AT90USB 82, AT90USB 162

Do čtvrté skupiny jsou zahrnuty typy:
ATmega 16M1, 32M1, 64M1
AT90 PWM 1, 216, 316, 2, 2B, 3, 3B

2.6 Nastavení komunikace pomocí rozhraní SPI pro AVR XMEGA

Mikrokontroléry AVR XMEGA podporují rozhraní SPI na portech PORT C, PORT D, PORT E a PORT F. Do této kategorie patří typy ATXmega 64A1, 128A1, 192A1, 256A1 a 384A1. Z těchto portů jsou vždy využity 4 piny pro SPI. V tabulce tab. 2.10 je uvedeno využití pinů pro PORT C. Pro ostatní porty je využití pinů stejné

Tab. 2.10: Piny pro komunikaci po SPI

Pin	Funkce
6	MISO (SPI bus master input)
5	MOSI (SPI bus master output)
7	SCK (SPI bus seriál clock)
4	SS (SPI slave select input)

Popis pinů:
MISO: SPI Master data input, slave data output pro kanál SPI. Když je zapnuto SPI jako master, tento pin je nastaven jako vstupní bez ohledu na nastavení PORTC.DIR.
MOSI: SPI Master data output, slave data input pro kanál SPI. Když je zapnuto SPI jako slave, tento pin je nastaven jako vstupní bez ohledu na nastavení PORTC.DIR. Pokud je SPI nastaveno jako master, směr toku dat je nastaven pomocí PORTC.DIR.
SCK: Master clock output, slave clock input pro kanál SPI. Když je SPI zapnuto jako slave, pin je nastaven jako vstupní, bez ohledu na nastavení PORTC.DIR. Když je SPI zapnuto jako master, směr toku je nastaven pomocí PORTC.DIR.
SS: Slave port select input. Pokud je SPI zapnuto jako slave, pin je nastaven jako vstupní bez ohledu na nastavení PORTC.DIR. Jako slave je SPI aktivováno při nastavení tohoto pinu no úroveň low. Při nastavení SPI jako master, směr toku dat je nastaven pomocí PORTC.DIR.

Samotné povolení režimu SPI na mikrokontroléru se provádí nastavením registru CTRL – SPI control register. Význam registru je zobrazen v tabulce Tab. 2.11: CTRL – SPI control register.

Tab. 2.11: CTRL – SPI control register

Bit	7	6	5	4	3	2	1	0
Zkratka	CLK2X	ENABLE	DORD	MASTER	MODE		PRESCALER	
Počáteční hodnota	0	0	0	0	0	0	0	0

Význam bitů:

7 – SPIECLK2X: SPI Clouck double

Pokud je tento bit nastaven, rychlost SPI(frekvence SCK) je dojnásobná(režim MASTER)

6 – ENABLE: SPI Enable

Nastavením tohot bitu se povoluje režim SPI. Bit musí být nasteven pro jakoukoliv SPI operaci

5 – DORD: Data order

Nastavením na „1“ znamená, že LSB je posílán jako první.

Nastavením na „0“ znamená, že MSB je posílán jako první.

4 – MSTR: Master/Slave Select

Tímto bitem se nastavuje master SPI mód, při nastavení na „1“ a slave SPI při nastavení na „0“.

3 :2– MODE[1:0]: SPI Mode

Nastavením těchto bitů se rozhoduje, zda se budou data vzorkovat na nástupnou nebo sestupnou hranu signálu SCK.

1:0 – PRESCALER[1:0]: SPI Clock Prescaler

Tyto dva bity nastavují vztah mezi frekvencí SCK a periferními hodinami jak je uvedeno v tabulce tab. 2.12

Tab. 2.12: Vztah mezi frekvencí SCK a periferními hodinami

CLK2X	**PRESCLAER[1:0]**	**frekvence SCK**
0	00	$clk_{PER}/4$
0	01	$clk_{PER}/16$
0	10	$clk_{PER}/64$
0	11	$clk_{PER}/128$
1	00	$clk_{PER}/2$
1	01	$clk_{PER}/8$
1	10	$clk_{PER}/32$
1	11	$clk_{PER}/64$

Abychom mohli zjistit, zda data, která jsme poslali do registru DATA byla odeslána, musí se povolit příznak přerušení v registru STATUS – SPI status register, který je popsaný v následující tabulce:

Tab. 2.13: STATUS – SPI status registr

Bit	7	6	5	4	3	2	1	0
Zkratka	IF	WCOL	-	-	-	-	-	-
Počáteční hodnota	0	0	0	0	0	0	0	0

Popis bitů:
7 – SPIF: SPI Interrupt Flag
Pokud je sériový přenos dokončen (tj. přenos jednoho bytu pomocí registru DATA), příznak IF je nastaven na „1“. Příznak IF je nulován hardwarově při prvním čtení SPI status registru (DATA).
6 – WCOL: Write COLlision Flag
Příznak je nastaven, pokud je datový registr DATA přepsán během přenosu dat. Příznak WCOL (a příznak IF) jsou vynulovány při prvním čtení datového registru DATA.
5:0 – Res: Reserved Bits
Tyto bity jsou rezervovány a vždy nastaveny na „0“.

Každý z portů má svůj STATUS registr, ke kterému se přistupuje spojením názvu portu a registru. Pro port C `SPIC.STATUS`, pro port D `SPID.DIR` atd.

Samotný přenos dat probíhá pomocí osmibitového posuvného registru DATA – SPI data registr pro čtení i zápis. Je využit i při posílání řídících příkazů (CMD) a datových tokenů. Zapsání dat do registru vyvolá přenos dat. Čtení registru probíhá z vyrovnávací paměti posuvného registru. Každý z portů má svůj datový registr DATA, ke kterému se přistupuje spojením názvu portu a registru. Pro port C `SPIC.DATA`, pro port D `SPID.DATA` atd.

Dále je třeba na portech využívající SPI (PORT C, D, E, F) určit směr komunikace jednotlivých pinů. K tomuto účelu slouží směrový registr DIR – Data Direction Register. Význam bitů je znázorněn v následující tabulce:

Tab. 2.14: DIR – Data Direction Register

Bit	7	6	5	4	3	2	1	0
zkratka	B7	B6	B5	B4	B3	B2	B1	B0
Počáteční hodnota	0	0	0	0	0	0	0	0

Pin daného portu bude sloužit jako výstupní, při nastavení daného bitu v registru DIR na úroveň logické jedničky. Pokud má být daný pin portu vstupní, nastaví se daný bit na úroveň logické nuly. Každý z portů má svůj směrový registr DIR, ke kterému se přistupuje spojením názvu portu a registru. Pro port C `PORTC.DIR`, pro port D `PORTD.DIR` atd.

Mikrokontroléry z řady XMEGA rozlišují několik priorit přerušení. Pro nastavení priority u SPI slouží registr INTCTRL – SPI interrupt control register v tabulce tab. 2.15

Tab. 2.15: INTCTRL – SPI Interrupt control register

Bit	7	6	5	4	3	2	1	0
zkratka	-	-	-	-	-	-	INTLVL	
Počáteční hodnota	0	0	0	0	0	0	0	0

Popis bitů:
7:2 – Reserved
Tyto bity jsou rezervovány pro budoucí použití.

1:0 – INTLVL[1:0]: SPI Interrupt level
Tyto bity povolují přerušení od SPI a prioritu přerušení uvedenou v tabulce 2.16

Tab. 2.16: Priority přerušení

Kombinace bitů	Popis
00	Přerušení vypnuto
01	Nízká priorita přerušení
10	Střední priorita přerušení
11	Vysoká priorita přerušení

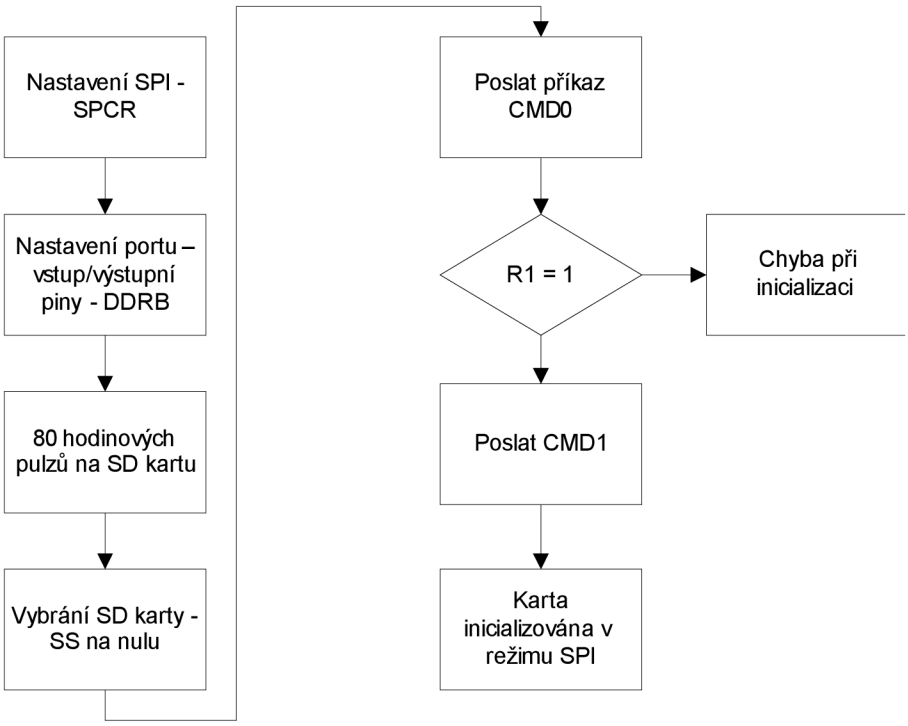
3 Popis imlementace s mikrokontroléry megaAVR

3.1 Inicializace

Nejprve na mikrokontroléru musíme nastavit rozhraní SPI, režim master a rychlost přenosu. To provede nastavením příslušných příznaků v kontrolním registru SPCR – SPI control registr. Pro tento účel to jsou příznaky SPE (povolí SPI), MSTR (režim master SPI) a SPR0 (nastaví frekvenci oscilátoru jako $f_{osc}/16$). V dalším kroku vybereme, které piny pro komunikace přes SPI budou vstupní a které výstupní. Výchozí nastavení portu je jako vstupní. Pro toto nastavení vyhovuje pouze pin MISO (master in, slave out, z pohledu SD karty DataOut). Proto pomocí registru určující směr toku dat DDRB nastavíme příznaky MOSI (master out, data in, z pohledu SD karty DataIn), SCK (hodinový signál), SS (slave select, chip select) jako výstupní. Po inicializaci SPI na mikrokontroléru musíme nastavit SD kartu. Pro přepnutí karty do režimu SPI pošleme 80 hodinových pulzů pro stabilizování karty, vybereme kartu nastavením SS na nulu. Pošleme příkaz CMD0, karta odpoví rámcem R1. Pokud je roven 1, pošleme příkaz CMD1 a krata je inicializovaná(funkce vrací 0 pokud proběhla inicializace v pořádku). Pokud rámec R1 není roven jedné, nastala chyba při inicializaci a funkce vrací návratový kód 1. Funkce pro inicializaci jsou rozděleny do skupin podle skupin v kapitole v kapitole 2.5. Celý proces je znázorněn ve vývojovém diagramu na obrázku 3.1. Pro první skupinu funkce `uint8_t sk1_init(void)`, pro druhou skupinu funkce `uint8_t sk2_init(void)`, pro třetí skupinu funkce `uint8_t sk3_init(void)`, pro čtvrtou skupinu `uint8_t sk3_init(void)`.

Po inicializaci je karta připravena k čtení a zápisu dat. Pokud ovšem uživatel nechce být s kartou trvale spojen, může ji z SPI odebrat pomocí funkce pro danou skupinu(kapitola 2.5). Pro první skupinu funkce `void uvolni_SD_sk1(void)`, pro druhou skupinu funkce `void uvolni_SD_sk2(void)`, pro třetí skupinu funkce `void uvolni_SD_sk3(void)`, pro čtvrtou skupinu `void uvolni_SD_sk4(void)`.

Pro opětovné vybrání karty slouží funkce `void vyber_SD_sk1(void)`, `void vyber_SD_sk2(void)`, `void vyber_SD_sk3(void)`, `void vyber_SD_sk4(void)`.



Obr. 3.1: Inicializace režimu SPI

3.2 Komunikace mezi SD a mikrokontrolérem

Komunikace mezi mikrokontrolérem a SD kartou probíhá po bytech. Data jsou posílána a přijímána datovým registrem SPDR. Dokud nejsou data přijata, nebo odeslána, probíha cyklus while, který testuje příznk SPIF ve stavovém registru SPSR. Jakmile je příznak nastaven, data jsou odeslána a program pokračuje dál. Funkci předáváme paramter, který byte má být poslán. Uživatel tuto funkci přímo nevyužívá.

```
uint8_t posli_byte(uint8_t pom)
{
  SPDR = pom;
      while(!(SPSR & (1<<SPIF))); // dokud neni vyvolano preruseni
cykly
      return SPDR;
}
```

3.3 Příkazový rámec

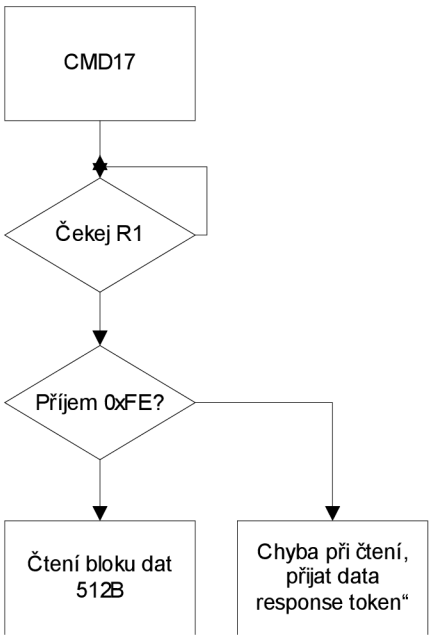
Pro posílání příkazových rámců(tzv. command frame) slouží funkce CMD. Struktura rámce je popsána výše v kapitole 1.5 Komunikace pomocí SD protokolu. Funkci předáváme tyto parametry: cmd – začíná sekvencí 01 a následuje číslo příkazu (předáváme již úplný byte zahrnující sekvenci 01 + číslo příkazu), dále dva 16-ti bitové argumenty příkazu. Kontrolní součet je fixně nastaven na 0x95 a v režimu SPI na něj není brán zřetel.

```
uint8_t CMD(uint8_t cmd, uint16_t high, uint16_t low)
{
  posli_byte(0xff);  // posle byte pro synchronizaci
      posli_byte(cmd);
      posli_byte((uint8_t)(high >> 8));
      posli_byte((uint8_t)high);
      posli_byte((uint8_t)(low >> 8));
      posli_byte((uint8_t)low);
      posli_byte(0x95); // crc v SPI nehraje roli, proto se neposila
jako parametr funkce a je fixne 0x95
      posli_byte(0xff);

  return posli_byte(0xFF); // vraci navratovy token
}
```

3.4 Čtení a zápis dat

Funkce pro čtení dat očekává při volání horní a dolní část 32 bitové adresy(rozdělené do dvou 16-ti bitových částí) odkud se mají data číst a ukazatel na buffer, kam se data budou ukládat. Po poslání příkazu CMD17 karta odpoví tokenem R1 a následuje datový token začínající sekvencí 0xFE. Po přijetí této sekvence probíhá načítání přijatých dat z datového registru SPDR do pole `*buffer` o velikosti 512 B. Následuje přijetí dvou bytů obsahujících kontrolní součet CRC, na který není brán zřetel. Pokud před tokem dat přijmeme chybový rámec data error (tj. Sekvence 0xFE nebyla doručena), data nebylo možno přečíst a funkce vrací chybové hlášení. Pokud čtení proběhlo v pořádku, funkce vrací 0. Vývojový diagram je zobrazen níže na obrázku obr. 3.2. Pro použití slouží funkce `int atmega_read(uint16_t AdrH, uint16_t AdrL, uint8_t *buffer)`

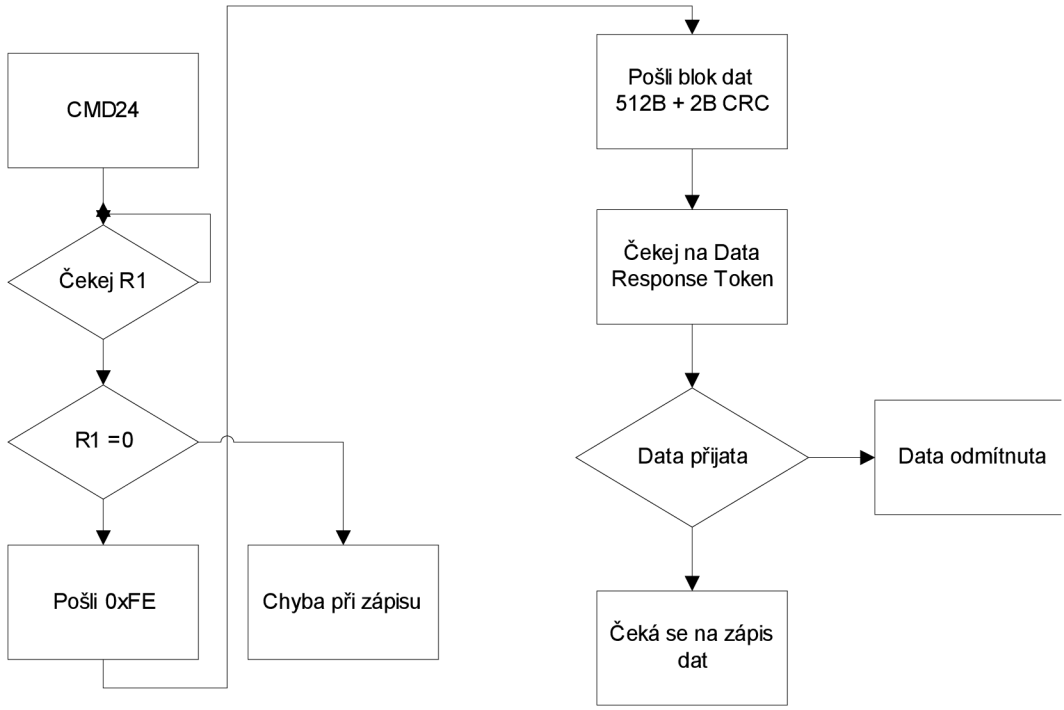


Obr. 3.2: Průběh čtení bloku dat

```
int atmega_read(uint16_t AdrH, uint16_t AdrL, uint8_t *buffer)
{
      uint16_t i;
      uint8_t er, R1;
```

```
      // CMD 17(+4B adresa), cteni z karty
      // do "R1" prijme odpoved R1
      // po prikazu pro cteni, karta odpovy tokenem R1, muze nastat
chyba definovana v R1
      if(CMD(0x51, uint16_t h, uint16_t l) != 0)
      {
    return 1; // navratova hodnota pro chybu
  }
      er = posli_byte(0xff);
  if(er == 0xFE) // pokud prijde 0xFE, znamena to, ze nasleduje blok
dat
      {
        for(i=0; i < 512; i++) //zacina cist blok 512 B
      {
      *buffer++ = posli_byte(0xff);
    }
      posli_byte(0xff);
        posli_byte(0xff); // vrati CRC check sum - 2B
      }
      else // nastala chyba cteni: Read Error : Bit 0 - Error, Bit 1 -
CC Error,Bit 2 - Card ECC failed, Bit 3 - out of range
      {
      er &= 0x0F; //nulovani nejvyssich ctyr neurcitych bitu
    if(er == 0x01)
    {return 2;} //chybovy kod
    if(er == 0x02)
    {return 3;} //chybovy kod
    if(er == 0x04)
    {return 4;} //chybovy kod
    if(er == 0x08)
    {return 5;}//chybovy kod
      }
  return 0;   // pri navratu 0, vse probehlo v poradku
}
```

Funkce pro zápis dat pracuje podobně jako funkce pro čtení dat. Očekává vstup od uživatele horních 16b a dolních 16b adresy a ukazatel na místo v paměti, kam se bude ukazovat. Funkce poté pošle pomocí funkce CMD příkaz pro zápis dat(CMD24) a adresu odkud se má číst z paměti(buffer). Dále se čeká na přijetí odpovědi rámce R1. Nyní se pošle sekvenci 0xFE, podle které karta pozná, že po této sekvenci následuje blok dat 512 bytů. Data se posílají přes datový registr SPDR a po každém naplnění registru SPDR se čeká na odeslání (tj. až nastane přerušení v stavovém registru SPSR). Po odeslání bloku dat následují dva byty reprezentující 16 bitů kontrolního součtu CRC. Po dokončení odesílání dat karta pošle data response token, ve kterém oznámí zda byla data přijata, nebo nastala nějaká chyba. Pokud žádná chyba nenastala, čeká se dokud karta nezapíše data do paměti. V tento okamžik je ve stavu busy. Pin MISO je po dobu zápisu na nule. Vývojový diagram pro zápis dat je zobrazen níže na obrázku obr 3.3. Pro použití slouží funkce `int atmega_write(uint16_t AdrH, uint16_t AdrL, uint8_t *buffer)`



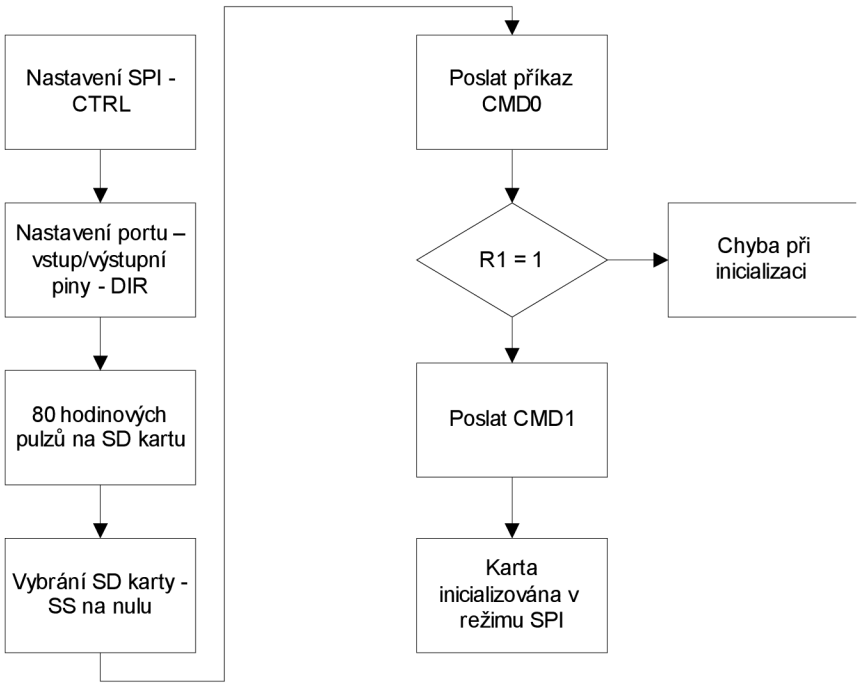
Obr. 3.3: Průběh zápisu bloku dat

```
int atmega_write(uint16_t AdrH, uint16_t AdrL, uint8_t *buffer)
{
      uint16_t i;
      uint8_t er;
      if(CMD(0x58, uint16_t h, uint16_t l) != 0)
      // CMD 24 - zapis jednoho bloku dat
      // po prikazu pro zapis, karta odpovy tokenem R1
      {
    return 1; // navratova hodnota pro chybu
  }
  posli_byte(0xfe); // host posle startovaci token „token star block"
      //zápis dat z bufferu 512B na kartu
      for (i = 0; i < 512; i++)
      {
    posli_byte(*buffer++);
      }
      posli_byte(0xff);
      posli_byte(0xff); // blok dat ukoncen 16 bit CRC
      // karta posila DATA RESPONSE TOKEN po prijeti dat
      er = posli_byte(0xff);
      er &= 0x1F;
      /*nulování nejvyšších tri neurcitych bitu a testovani na chybu,
      010 - Data přijata
  101 - Data odmítnuta z důvodu chybného kontrolního součtu CRC - tato
chyba nemuze nastat, v SPI je vypnuto
  110 - Data odmítnutnuta z důvodu chyby při zápisu*/
  if(er == 0x05 | er == 0x06)
  {return 2;}  //navratova hodnota 2 znamena chybu pri zapisu
  while(posli_byte(0xFF) != 0xFF); // data out line je DOWN po dobu
zapisu dat
      return 0; // vrati nulu, pokud se data zapsala v poradku
}
```

4 Popis imlementace s mikrokontroléry AVR XMEGA

3.1 Inicializace

Nejprve na mikrokontroléru musíme nastavit rozhraní SPI, režim master a rychlost přenosu. To provede nastavením příslušných příznaků v kontrolním registru CTRL – SPI control registr. Pro tento účel to jsou příznaky SPE (povolí SPI), MSTR (režim master SPI). V dalším kroku vybereme, které piny pro komunikace přes SPI budou vstupní a které výstupní. Výchozí nastavení portu je jako vstupní. Pro toto nastavení vyhovuje pouze pin MISO (master in, slave out, z pohledu SD karty DataOut). Proto pomocí registru určující směr toku dat DIR nastavíme příznaky MOSI (master out, data in, z pohledu SD karty DataIn), SCK (hodinový signál), SS (slave select, chip select) jako výstupní. Po inicializaci SPI na mikrokontroléru musíme nastavit SD kartu. Pro přepnutí karty do režimu SPI pošleme 80 hodinových pulzů pro stabilizování karty, vybereme kartu nastavením SS na nulu. Pošleme příkaz CMD0, karta odpoví rámcem R1. Pokud je roven 1, pošleme příkaz CMD1 a krata je inicializovaná(funkce vrací 0 pokud proběhla inicializace v pořádku). Celý proces je znázorněn ve vývojovém diagramu na obrázku 4.1. Pokud rámec R1 není roven jedné, nastala chyba při inicializaci a funkce vrací návratový kód 1. Jelikož SPI je podporováno na několika portech, slouží každá funkce pro inicializaci daného portu. Pro PORT C `uint8_t atx_init_pc(void)`, Pro PORT D `uint8_t atx_init_pd(void)`, Pro PORT E `uint8_t atx_init_pe(void)`, Pro PORT F `uint8_t atx_init_pf(void)`



Obr. 4.1: Inicializace režimu SPI

Po inicializaci je karta připravena k čtení a zápisu dat. Pokud ovšem uživatel nechce využívat být s kartou trvale spojen, může ji odebrat pomocí funkce. Opět pro každý port. Pro použití pro PORT C slouží funkce `void uvolni_SD_xc(void)`, pro PORT D `void uvolni_SD_xd(void)`, pro PORT E `void uvolni_SD_xe(void)`, pro PORT F `void uvolni_SD_xf(void)`.

Pro opětovné vybrání karty slouží pro PORT C funkce `void vyber_SD_xc(void)`, pro PORT D `void vyber_SD_xd(void)`, pro PORT E `void vyber_SD_xe(void)`, pro PORT F `void vyber_SD_xf(void)`.

3.2 Komunikace mezi SD a mikrokontrolérem

Komunikace mezi mikrokontrolérem a SD kartou probíhá po bytech. Data jsou posílána a přijímána datovým registrem DATA. Dokud nejsou data přijata, nebo odeslána, probíha cyklus while, který testuje příznk IF ve stavovém registru STATUS. Jakmile je příznak nastaven, data jsou odeslána a program pokračuje dál. Funkci předáváme paramter, který byte má být poslán. Uživatel tuto funkci přímo nevyužívá.

```
uint8_t posli_byte_xc(uint8_t pom)
{
  SPIC.DATA = pom;
      while(!(SPIC.STATUS & (1<<7))); // dokud neni vyvolano preruseni
cykly
      return SPIC.DATA;
}
```

3.3 Příkazový rámec

Pro posílání příkazových rámců(tzv. command frame) slouží funkce CMD. Struktura rámce je popsána výše v kapitole 1.5 Komunikace pomocí SD protokolu. Funkci předáváme tyto parametry: cmd – začíná sekvencí 01 a následuje číslo příkazu (předáváme již úplný byte zahrnující sekvenci 01 + číslo příkazu), dále dva 16-ti bitové argumenty příkazu. Kontrolní součet je fixně nastaven na 0x95 a v režimu SPI na něj není brán zřetel.

3.4 Čtení a zápis dat

Funkce pro čtení dat očekává při volání horní a dolní část 32 bitové adresy(rozdělené do dvou 16-ti bitových částí) odkud se mají data číst a ukazatel na buffer, kam se data budou ukládat. Po poslání příkazu CMD17 karta odpoví tokenem R1 a následuje datový token začínající sekvencí 0xFE. Po přijetí této sekvence probíhá načítání přijatých dat z datového registru SPDR do pole `*buffer` o velikosti 512 B. Následuje přijetí dvou bytů obsahujících kontrolní součet CRC, na který není brán zřetel. Pokud před tokem dat přijmeme chybový rámec data error (tj. Sekvence 0xFE nebyla doručena), data nebylo možno přečíst a funkce vrací chybové hlášení. Pokud čtení proběhlo v pořádku, funkce vrací 0. Vývojový diagram je zobrazen na obrázku obr. 3.2. Pro použití pro PORT C slouží funkce int `atx_read_xc(uint16_t AdrH, uint16_t AdrL, uint8_t *buffer)`, pro PORT D `int atx_read_xd(uint16_t AdrH, uint16_t AdrL, uint8_t *buffer)`, pro PORT E `int atx_read_xe(uint16_t AdrH, uint16_t`

`AdrL, uint8_t *buffer)`, pro PORT F `int atx_read_xf(uint16_t AdrH, uint16_t AdrL, uint8_t *buffer)`.

Funkce pro zápis dat pracuje podobně jako funkce pro čtení dat. Očekává vstup od uživatele horních 16b a dolních 16b adresy a ukazatel na místo v paměti, kam se bude ukazovat. Funkce poté pošle pomocí funkce CMD příkaz pro zápis dat(CMD24) a adresu odkud se má číst z paměti(buffer). Dále se čeká na přijetí odpovědi rámce R1. Nyní se pošle sekvenci 0xFE, podle které karta pozná, že po této sekvenci následuje blok dat 512 bytů. Data se posílají přes datový registr SPDR a po každém naplnění registru SPDR se čeká na odeslání (tj. až nastane přerušení v stavovém registru SPSR). Po odeslání bloku dat následují dva byty reprezentující 16 bitů kontrolního součtu CRC. Po dokončení odesílání dat karta pošle data response token, ve kterém oznámí zda byla data přijata, nebo nastala nějaká chyba. Pokud žádná chyba nenastala, čeká se dokud karta nezapíše data do paměti. V tento okamžik je ve stavu busy. Pin MISO je po dobu zápisu na nule. Vývojový diagram pro zápis dat je zobrazen níže na obrázku obr 3.3. Pro použití pro PORT C slouží funkce int `atx_ write_xc(uint16_t AdrH, uint16_t AdrL, uint8_t *buffer)`, pro PORT D `int atx_ write_xd(uint16_t AdrH, uint16_t AdrL, uint8_t *buffer)`, pro PORT E `int atx_ write_xe(uint16_t AdrH, uint16_t AdrL, uint8_t *buffer)`, pro PORT F `int atx_ write_xf(uint16_t AdrH, uint16_t AdrL, uint8_t *buffer)`

Závěr

Tato práce obsahuje náhled na problematiku základní obsluhy SD karty (čtení, zápis) na fyzické vrstvě pro co nejširší škálu obsluhujících mikrokontrolérů. Ačkoliv se může hardware pro obsluhu SD karty měnit, knihovna bude pracovat s daty vždy dle specifikace obsluhujícího mikrokontroléru. Předpokládané využití mikrokontrolérů je uvedeno v kapitole 2.5 a 2.6 a k nim příslušné obsluhující funkce v kapitole 3 a 4. Pro ještě větší hardwarovou nezávislost by bylo možné do knihovny implementovat obsluhu digitálních signálových procesorů(DSP).

Literatura

[1] *SD Specifications : Simplified Specification* [online]. 2400 Camino Ramon, Suite 375 : SD Card Association, September 25, 2006 [cit. 2010-12-16]. Dostupné z WWW: <http://www.sdcard.org/developers/tech/sdcard/pls/simplified_specs/Part_1_Physical_Layer_Simplified_Specification_Ver3.01_Final_100518.pdf>

[2] *SD Memory Card Specifications : PHYSICAL LAYER SPECIFICATION* [online]. [s.l.] : SD Group (MEI, SanDisk, Toshiba), 2001 [cit. 2010-12-16]. Dostupné z WWW: <http://www.uloz.to/6979728/spi-sd-bus-32-str-pdf>

[3] *FOUST, F. Secure Digital Card Interface for the MSP430* [online]. Michigan State University, : Michigan State University, Dept. of Electrical and Computer Engineering, 2004 [cit. 2010-12-16]. Dostupné z WWW: <http://alumni.cs.ucr.edu/~amitra/sdcard/Additional/sdcard_appnote_foust.pdf>.

Seznam příloh

Příloha č. 1: Knihovna pro práci s SD kartou
lib_SD.h

```
#include <avr/io.h>
#include "lib_SD.h"
/*                              ATmega AVR                                      */
/*******************************************************************************
********************** inicializace pro skupinu 1 ******************************
*******************************************************************************
ATmega 8, 8L, 8A, 48, 88, 168, 48A, 48PA, 88A, 88PA, 168A, 168PA, 328, 328P, 48P, 88P,
168P
*/
extern uint8_t sk1_init(void)
/*******************************************************************************
********************** inicializace pro skupinu 2 ******************************
*******************************************************************************
ATmega 32,162,162V,8515,8515L, 8535, 8535L, 16, 16L, 164A, 164PA, 324A, 324PA, 644A,
644PA, 1284, 1284P, 164P/V, 324P/V, 644P/V
*/
extern uint8_t sk2_init(void)
/*******************************************************************************
********************** inicializace pro skupinu 3 ******************************
*******************************************************************************
ATmega 165P, 165PV, 165A, 165PA, 325PA, 3250PA, 3250A, 640, 645A, 645P, 6450A, 6450P,
325, 3250, 645, 6450, 16U4, 32U4, 8U2, 16U2, 1280, 1281, 2560, 25061, 169PA, 329A,
329PA, 649A, 649P, 3290A, 3290PA, 6490A, 6490P
AT90USB CAN32, AT90USB CAN64, AT90USB CAN128
AT90USB 649, AT90USB 647, AT90USB 1286, AT90USB 1287, AT90USB 82, AT90USB 162
*/
extern uint8_t sk3_init(void)
/*******************************************************************************
********************** inicializace pro skupinu 4 ******************************
*******************************************************************************
ATmega 16M1, 32M1, 64M1
AT90 PWM 1, 216, 316, 2, 2B, 3, 3B
*/
extern uint8_t sk4_init(void)

// odebere kartu ze sbernice
extern void uvolni_SD_sk1(void)
extern void uvolni_SD_sk2(void)
extern void uvolni_SD_sk3(void)
extern void uvolni_SD_sk4(void)
// vybere kartu
extern void vyber_SD_sk1(void)
extern void vyber_SD_sk2(void)
extern void vyber_SD_sk3(void)
extern void vyber_SD_sk4(void)
/*
************* FUNKCE pro poslani prikazu COMMAND na SD kartu ******************
*/
// posila 5B(6. B je crc=vzdy 0x95) dle struktury command ramce, fce CMD posila prikaz
SD karte
extern uint8_t CMD(uint8_t cmd, uint16_t high, uint16_t low)
// posle 1B pres SPDR na sbernici
extern uint8_t posli_byte(uint8_t pom)
/*
********************** FUNKCE PRO CTENI ***************************************
*/
// funkce pro cteni, ocekava vstup: hornich 16b adresy, dolnich 16 b adresy(cela adresa
ma 32b), ukazatel na buffer, kam se data ulozi
extern int atmega_read(uint16_t AdrH, uint16_t AdrL, uint8_t *buffer)
/*
********************** FUNKCE PRO ZAPIS ***************************************
*/
// ocekava vstup: hornich 16b adresy, dolnich 16 b adresy(cela adresa ma 32b), ukazatel
na buffer, odkud se budou data posilat na SD kartu
extern int atmega_write(uint16_t AdrH, uint16_t AdrL, uint8_t *buffer)

/*                              ATXMEGA                                         */
```

```
// inicializace pro ATxmega 64A1, 128A1, 192A1, 256A1, 384A1
extern uint8_t atx_init_pc(void) // PORT C
extern uint8_t atx_init_pd(void) // PORT D
extern uint8_t atx_init_pe(void) // PORT E
extern uint8_t atx_init_pf(void) // PORT F
/*
************* FUNKCE pro poslani prikazu COMMAND na SD kartu ******************
*/
// posila 5B(6. B je crc=vzdy 0x95) dle struktury command ramce, fce CMD posila prikaz
SD karte
extern uint8_t CMD_xc(uint8_t cmd, uint16_t high, uint16_t low) // PORT C
extern uint8_t CMD_xd(uint8_t cmd, uint16_t high, uint16_t low) // PORT D
extern uint8_t CMD_xe(uint8_t cmd, uint16_t high, uint16_t low) // PORT E
extern uint8_t CMD_xf(uint8_t cmd, uint16_t high, uint16_t low) // PORT F
// posle 1B pres SPDR na sbernici
extern uint8_t posli_byte_xc(uint8_t pom) // PORT C
extern uint8_t posli_byte_xd(uint8_t pom) // PORT D
extern uint8_t posli_byte_xe(uint8_t pom) // PORT E
extern uint8_t posli_byte_xf(uint8_t pom) // PORT F
// odebere kartu ze sbernice
extern void uvolni_SD_xc(void) // PORT C
extern void uvolni_SD_xd(void) // PORT D
extern void uvolni_SD_xe(void) // PORT E
extern void uvolni_SD_xf(void) // PORT F
// prida kartu na sbernici
extern void vyber_SD_xc(void) // PORT C
extern void vyber_SD_xd(void) // PORT D
extern void vyber_SD_xe(void) // PORT E
extern void vyber_SD_xf(void) // PORT F
/*
********************** FUNKCE PRO CTENI ************************************
*/
// funkce pro cteni, ocekava vstup: hornich 16b adresy, dolnich 16 b adresy(cela adresa
ma 32b), ukazatel na buffer, kam se data ulozi

extern int atx_read_xc(uint16_t AdrH, uint16_t AdrL, uint8_t *buffer) // PORT C
extern int atx_read_xd(uint16_t AdrH, uint16_t AdrL, uint8_t *buffer) // PORT D
extern int atx_read_xe(uint16_t AdrH, uint16_t AdrL, uint8_t *buffer) // PORT E
extern int atx_read_xf(uint16_t AdrH, uint16_t AdrL, uint8_t *buffer) // PORT F
/*
********************** FUNKCE PRO ZAPIS ************************************
*/
// ocekava vstup: hornich 16b adresy, dolnich 16 b adresy(cela adresa ma 32b), ukazatel
na buffer, odkud se budou data posilat na SD kartu
extern int atx_write_xc(uint16_t AdrH, uint16_t AdrL, uint8_t *buffer) // PORT C
extern int atx_write_xd(uint16_t AdrH, uint16_t AdrL, uint8_t *buffer) // PORT D
extern int atx_write_xe(uint16_t AdrH, uint16_t AdrL, uint8_t *buffer) // PORT E
extern int atx_write_xf(uint16_t AdrH, uint16_t AdrL, uint8_t *buffer) // PORT F
```

Kompletní znění programu lib_SD.c je uvedeno na přiloženém DVD.