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

EMULÁTOR 3.5“ DISKETOVÉ MECHANIKY POMOCÍ RS232 A SD PAMĚŤOVÉ KARTY

EMULATOR OF 3.5" DISKETTE DRIVE USING RS232 AND SD MEMORY CARD

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — Bc. DAVID SEDLÁČEK

AUTHOR

VEDOUCÍ PRÁCE — Ing. ONDŘEJ PAVELKA

SUPERVISOR

BRNO 2012

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student: Bc. David Sedláček *ID*: 109716
Ročník: 2 *Akademický rok*: 2011/2012

NÁZEV TÉMATU:

Emulátor 3.5“ disketové mechaniky pomocí RS232 a SD paměťové karty

POKYNY PRO VYPRACOVÁNÍ:

Navrhněte a realizujte emulátor 3.5“ disketové mechaniky používané v PC. Emulátor bude vybaven stejným konektorem jako originální FDD mechanika. Vlastní data budou uloženy na SD paměťové kartě, jejíž obsah (a tedy i obsah „diskety“) bude modifikovatelný pomocí příkazů na RS232 sběrnici, popř. USB rozhraní. Emulátor realizujte pomocí mikrokontroléru ATmega, ovládací program pro nabíječ a ovládání vytvořte v jazyce C. Velikost realizovaného emulátoru bude shodná s velikostí standardní 3.5“ FDD mechaniky. Správnou funkci emulátoru ověřte na PC.

DOPORUČENÁ LITERATURA:

[1] Mann, B. : C pro mikrokontroléry, BEN, 2003. ISBN 80-7300-077-6
[2] Aradio PE č. 4/2007 - Multimedia karty a čo s nimi – str. 19

Termín zadání: 6.2.2012 ***Termín odevzdání***: 24.5.2012

Vedoucí práce: Ing. Ondřej Pavelka

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

Abstrakt

Diplomová práce se zabývá návrhem emulátoru 3,5“ disketové mechaniky pomocí procesoru ATMEGA. Tento emulátor byl navržen dle zásad konstruování elektronických zařízení, pro potřeby emulátoru byla také objektově naprogramována řídicí aplikace spolu s firmware k mikrokontroléru, který podporuje i MFM kódování. V této práci jsou rovněž uvedeny všechny formáty ukládaných a přenášených dat spolu s některými vývojovými diagramy.

Klíčová slova

atmega, emulátor, disketa, programování, disketová mechanika, rs232, usb, programování, c, c#, návrh plošných spojů, emulace, usart, atmel, avr

Abstract

This thesis deals with the design of the 3,5" floppy drive emulator with ATMEGA microprocessor unit. The emulator has been designed according to the principles of designing electronic devices, there is also object-control application and firmware for a microcontroller, which supports MFM coding. The thesis also lists all the formats of data stored or transmitted along with some flowcharts.

Keywords

atmega, emulator, floppy, programming, floppy drive, rs232, usb, programming, c, c#, printed circuit design, emulation, usart, atmel, avr

Bibliografická citace:

SEDLÁČEK, D. *Emulátor 3.5“ disketové mechaniky pomocí RS232 a SD paměťové karty*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2012. 52 s. Vedoucí diplomové práce Ing. Ondřej Pavelka.

Prohlášení

Prohlašuji, že svoji diplomovou práci na téma „Emulátor 3.5" disketové mechaniky pomocí RS232 a SD paměťové karty“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne ....

podpis autora

Poděkování

Na tomto místě chci poděkovat vedoucímu práce Ing. Ondřeji Pavelkovi za vedení, zapůjčení zařízení a pomoc při realizaci.

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

ÚVOD

Emulátor disketové mechaniky je elektronické zařízení, které umožňuje reprodukovat chování původní mechaniky, tedy simulovat všechny stavové a datové linky v závislosti na stavovém diagramu původní disketové mechaniky.

Tato diplomová práce popisuje emulátor z pohledu vývojáře software a hardware a věnuje se návrhu a realizaci emulátoru disketové mechaniky, jeho tištěnému spoji, schématu, řídicí aplikace pro PC a firmware mikroprocesoru v C.

Samotná práce je napsána především z pohledu programátora, a to z toho důvodu, že podstatná část funkce emulátoru je řešena mikrokontrolérem. Zároveň je nezbytné navrhnout a vystavět hardware, na kterém bude emulátor fungovat a od kterého se odvíjí předpoklad pro další správnou funkci zařízení, a návrh řídicí aplikace. Emulátor bude podporovat MFM kódování, načítání dat z paměťové karty a manipulaci s daty dle zadání.

V části práce s řídícím software je kladen důraz na aplikování objektově orientovaného stylu programování, jenž v důsledku tvoří velmi flexibilní aplikaci, která s jistými úpravami může sloužit nejen pro zařízení diskutované v této práci.

Práce je dělena do třech základních částí:

- teoretické poznatky pro sestavení emulátoru;
- výběr vhodného prostředí pro vývoj, popisy protokolů, práce disketové mechaniky a například i použitého kódování;
- popis samotného emulátoru – některé procesy, navržené formáty dat, styl komunikace;
- praktická ukázka ve formě fotografií.

Cílem této práce není popsat kompletně zdrojový kód nebo použité protokoly a struktury dat, ale přiblížit návrh zařízení z pohledu logické funkčnosti jednotlivých logických celků a jejich vzájemné návaznosti. V rámci této práce bude kladen důraz na použití vizuálních schémat vzhledem k jejich velké vyjadřovací schopnosti. Není-li uvedeno jinak, všechny ilustrace v dokumentu jsou dílem autora.

1 ROZBOR POŽADAVKŮ

1.1 Požadavky dle zadání

Hlavní požadavek dle zadání je jeden - navrhnout a realizovat emulátor 3,5“ disketové mechaniky. Emulátor bude mít stejné vlastnosti, jako má původní emulovaná disketová mechanika, tj.:

- komunikační rozhraní,
- silové napájecí rozhraní.

Jako úložné médium bude sloužit SD paměťová karta s kapacitou minimálně 150 MB. Ovládací rozhraní emulátoru bude řešeno pomocí dobře známé RS232 sériové linky. Hlavní mikroprocesor emulátoru bude Atmel AVR ATMEGA. Jako programovací jazyk je zvolen jazyk C.

1.1.1 Požadavky na funkci zařízení

Emulátor bude podporovat změnu jednotlivých sektorů, či na vyšší úrovni souborů pomocí sériového rozhraní. Formáty emulovaných disket budou IBM 2.0 a IBM 1.0, respektive 1,44 MB a 720 kB.

1.1.2 Požadavky na řídící aplikaci emulátoru

Aplikace bude podporovat správu všech „obrazů[1]“ disket – mazání, přidávání a úprava stávajících. Obrazů bude celkem 100.

1.2 Predispozice

Pro fungování emulátoru je nutné standardní IBM PC nebo jiné zařízení s řadičem disketové mechaniky kompatibilním s normou IBM 2.0 a IBM 1.0 a +5V SS napětí o výkonu minimálně 2 W.

1.3 Dodatečně zvolené požadavky

Tyto požadavky byly zvoleny dodatečně autorem této práce, vedou k upřesnění směřování práce a detailněji specifikují, anebo rozšiřují hlavní požadavky. Tyto požadavky byly zvoleny na základě teoretických znalostí či praktických zkuše-

[1] Obrazem je myšlen soubor 2880 sektorů, každý o délce 512 Bajtů.

ností – například použité vývojové prostředí a jeho ergonomie ovládání a orientace v něm.

Uživatelský ovládací panel bude mít dvě ovládací tlačítka – pro (in/de)cementaci čísla obrazu v rozsahu 0 - 99. Číslo aktuálně zvoleného obrazu bude indikováno 2 místným LED displejem. Paměťová karta, sloužící jako úložné zařízení, bude ve formátu micro-SD. Od standartní SD karty se liší se pouze svými menšími rozměry.

Řídící aplikace bude podporovat i formátování paměťové karty dále navrženého schématu. Jako souborový systém jednotlivých disketových obrazů bude ponechán FAT12.

2 AKTUÁLNÍ NABÍDKA EMULÁTORŮ NA TRHU

2.1 Výčet emulátorů

Disketové mechaniky jsou toho času relativně výběhová zařízení, která pro své špatné mechanické a funkční vlastnosti přirozeně téměř vymizela z trhu. Ale i přesto můžeme nalézt uplatnění v různých zařízeních, která jsou doposud v provozu – zejména strojnictví a různé obráběcí stroje. Z tohoto důvodu je trh s těmito zařízeními relativně malý, ale některé produkty nalezneme stále na Internetu. Kromě různých „domácích“ řešení, která jsou buď nedokončená, nebo nejsou z různých důvodů funkční, tu jsou i komerční produkty, a to například:

- SDCard HxC floppy emulator
- www.floppyemulator.com

Zmiňované emulátory jsou zde uvedeny pouze jako alternativa k navrhovanému zařízení v rámci této práce a za účelem srovnání.

2.2 SDCard HxC floppy emulator

Obr. 2.1: HxC2001 emulátor

Původně amatérský projekt pro počítače AMIGA, nyní komerční řešení. Jedná se o relativně výkonný emulátor, podporuje různé varianty souborových systémů a rychlostí kódování. Je postaven na CPLD[2] hradlovém poli. Je to univerzální 3“, 3,5“, 5,25“ a 8“ emulátor. Umí pracovat s DD[3], HD[4]-MFM[5] a FM[6] disketami.

[2] Complex programmable logic device
[3] Double density
[4] High density
[5] Modified frequency modulation
[6] Frequency modulation

Některé jeho parametry:

- Shugart rozhraní
- PC rozraní
- emulace dvou disketových mechanik
- podpora zápisu do obrazu diskety [1]

Podporuje různé délky datových sektorů – 128, 256, 512 a 1024 B. Má svoji uživatelskou základnu a v diskusním fóru lze s autorem řešit případné nedostatky či problémy. Jedná se o velice funkčně zajímavý produkt, ale vzhledem k jeho proprietárnímu charakteru ho nelze v této práci využít jako informační zdroj.

2.3 www.floppyemulator.com

Další z řady emulátorů. Jedná se o variaci na jeden typ emulátoru, který je dále „brandován“ a prodáván na Internetu pod mnoha různými označeními. O jeho technické specifikaci se dá ale najít méně informací.

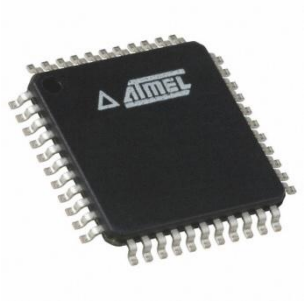
- kapacita: 1,44 MB; 1,2 MB; 720 kB
- počet tracků: 80
- sektorů / track: 18
- typ kódování souborů: MFM
- typ souborového systému: FAT12
- rozhraní: Shugart/PC
- přenosová rychlost 500 kbps surová data

Výrobce na svých stránkách uvádí, že má mezi svými zákazníky i nadnárodní firmy, z toho lze usoudit, že se jedná o relativně funkční produkt. [2]

3 ZAŘÍZENÍ

Pro realizaci emulátoru je nutné vybrat odpovídající zařízení. Výběr zařízení, nástrojů a vývojových prostředků bude podvolen hlavním požadavkům, které byly specifikovány v rozboru na začátku této práce.

3.1 Volba procesoru



Procesor je jádrem celého emulátoru. Zpracovává povely řídící aplikace, povely od disketového řadiče pomocí stavových linek a reaguje na ně v přesně stanovených časových relacích. Zpracovává vstupní data, která jsou uložena na paměťové kartě, provádí jejich transformaci a dále je prezentuje řadiči. Musí též reagovat na uživatelské události – v tomto případě na povely z ovládacího panelu.

Volba procesoru je kritická. Požadavky na procesor v rámci řady AVR ATMEGA byly následující:

- maximální pracovní kmitočet,
- maximální SRAM,
- FLASH paměť alespoň 16kB,
- USB rozhraní,
- vestavěná násobička,
- 32 vstupně výstupních pinů,
- USART[7] rozraní v módu SPI,
- rozhraní pro rozšíření operační paměti,
- dlouhodobý prodej modelu a jeho běžná dostupnost u nás.

Poslední zmíněný bod byl důležitý pro případnou snadnou reprodukci emulátoru i později.

[7] Universal synchronous and asynchronous receiver / transmitter

Po aplikaci filtru výše na řadu ATMEGA zbyly tyto modely:

Tab. 3.1: Výběr modelů MCU ATMEGA pro realizaci emulátoru

Model	Flash [kB]	Počet I/O pinů [-]	Max. takt [MHz]	SRAM [kB]
AT90USB647	64	48	16	4
ATMEGA644PA	64	32	20	4
ATMEGA644	64	32	20	4
ATMEGA644A	64	32	20	4
ATMEGA644P	64	32	20	4

Modely v Tab. 3.1 jsou 4. ATMEGA644 je výběhový model, ale stále prodávaný, což ho diskvalifikovalo. K jeho neprospěchu hovoří i fakt, že má pouze jedno rozhraní USART. Stejně tak modely ATMEGA644P a ATMEGA644A, které sice mají dvě USART rozhraní, ale jsou to výběhové modely. [3]

Jako hlavní mikroprocesor byl vybrán AT90USB647, který má na rozdíl od ATMEGA644 sníženou spotřebu energie až o polovinu a stále ho firma Atmel[8] vyrábí. Dále podporuje jako jediný USB rozhraní, USART v módu SPI a rozhraní pro rozšíření paměti.

Nicméně všechny výše uvedené procesory splňují požadavky na výkonnost a jsou v tomto ohledu rovnocenné. Stejně tak, pokud mají společné registry, jsou adresovány všechny stejně, pouze některé modely byly doplněné několika registry – zejména pro USART1 rozhraní.

3.2 Volba programovacího jazyka

3.2.1 Firmware mikrokontroléru emulátoru

Programovací jazyk slouží k interpretaci strojového kódu do lidsky srozumitelné podoby. Na počítačových platformách jsou dnes běžně používány vysokoúrovňové jazyky, jako je například JAVA, C#, které jsou několika logickými úrovněmi nadneseny nad strojový kód. Podobně je na tom jazyk C, který byl vybrán pro svoji populárnost v oblasti programování mikroprocesorů.

Jazyk C je velmi rozšířený, má mnoho kompilátorů do rozličných strojových platforem. V případě této práce je požadována existence kompilátoru pro platformu 8bit AVR.

[8] www.atmel.com

Toto splňují například:

- ICCAVR od firmy Atmel,
- CodeVisionAVR
- IAR Embedded Workbench for Atmel AVR,
- AVR GCC, avrlibc.

Všechny tyto kompilátory mají svoji uživatelskou základnu, což je důležité v případě nahodilého problému, jsou dostupné „zdarma“ (viz dále) a funkčně kryjí požadavky této práce.

Pouze AVR GCC[9] a binárními knihovnami s uživatelskými funkcemi AVRLIBC jsou tzv. „OpenSource[10]“, což znamená, že jsou volně šiřitelné a jejich zdrojové kódy jsou běžně dostupné zadarmo. To zaručuje jednak zpětnou kompatibilitu napříč novými verzemi a též širokou uživatelskou základnu včetně velkých korporátních společností, na které z velké části stojí jejich vývoj. Pro operační systém se dodává v balíku WinAVR[11].

3.2.2 Dodatek k volbě programovacího jazyka

Jak bylo zmíněno výše, jako programovací jazyk byl zvolen jazyk C. Z jeho podstaty je zřejmé, že můžou nastat specifické situace, které kompilátor neřeší maximálně efektivně. V tomto případě se jedná o obsluhu ISR[12]. Standardně se v AVR GCC ISR řeší konstrukcí `ISR(<vektor_přerušení>[, <atributy>])`.

[9] GNU compiler collection
[10] Doslovný překlad: volný zdroj – volně šiřitelný bezplatný kód
[11] http://winavr.sourceforge.net/
[12] Interrupt service routine – volně přeloženo jako obsluha přerušení běhu programu

Takto kompilovaná struktura bez atributů je ve strojovém kódu interpretována jako následující strojové instrukce:

```
push   R1
push   R0
in     R0, <adresa registru SREG>
push   R0
clr    R0
(…) kód programu (…)
pop    R0
out    <adresa registru SREG>, R0
pop    R0
pop    R1
```

Všechny tyto instrukce jsou v běžných případech žádoucí, programátorovi zajistí, že po návratu z obsluhy přerušení je procesor ve stejném stavu, jako předtím, než do ní vstoupil. V našem případě, jak se dozvíme dále, je to nežádoucí ztráta strojového času. Všechny instrukce výše spotřebují přibližně 15 strojových cyklů, což je při dané frekvenci procesoru doba o délce 750 ns. Pro porovnání, interval přenášených jednotek bitů kódu MFM při formátu IBM 2.0 je 1 μs, což odpovídá 20 strojovým cyklům při frekvenci při 20 MHz.

Z důvodu výše uvedeného byl jako doplňkový jazyk zvolen jazyk Assembler a to ve specifických – časově kritických sekcích programu mikrokontroléru - obsluze přerušení. To dává programátorovi naprostou kontrolu nad chodem rutiny a taktéž požaduje znalost interních pochodů procesoru. Tím je využit strojový čas procesoru velmi efektivně a realizace emulátoru je možná. Použití jazyka C by zde mělo za následek požadavek na vyšší pracovní frekvenci procesoru.

Je zde též vhodné zmínit tzv. „Inline C Assembly“, což je technika, při které se dá vložit do zdrojového C kódu kód Assembleru. Tento způsob zápisu je ale příliš komplikovaný, zápis pouze v Assembleru je funkčně identický, přímočařejší a více přehledný.

```
__asm__ ("movl %eax, %ebx\n\t"
         "movl $56, %esi\n\t"
         "movl %ecx, $label(%edx,%ebx,$4)\n\t"
         "movb %ah, (%ebx)");
```

[4]

Oba dva kódy jsou použitelné zároveň v jednom projektu. Assemblerovský kód se „přilinkuje“ ke kompilovanému C kódu linkerem a vzniká binární strojový kód.

3.2.3 Volba programovacího jazyka pro řídící software

Jazyk pro psaní řídící aplikace nebyl nijak specifikován. Mezi hlavní požadavky zde uvedeme:

- podpora vícevláknového zpracování, například pro oddělení grafického rozhraní a výkonové logiky;
- kvalitní podpora debugování, práce s programovými a datovými break-pointy, důležité z hlediska pozorování navržené aplikace a případné odlaďování výkonu či hledání problémových partií;
- snadná intergrace do systému – minimální počet potřebných knihoven – ty by měly být buď součástí SDK, nebo samotného operačního systému.

Možné varianty jsou například – JAVA, C#, C++. Pro tuto práci byl vybrán jazyk C# původem od firmy Microsoft. Je dobře integrovaný do stávajících populárních systému Microsoft Windows, ale je též podporován pod operačním systémem GNU/Linux pod názvem Mono[13]. I přes jeho relativně velkou vysokoúrovňovost je v něm možné pracovat přímo s jednotlivými komponentami počítače prostřednictvím jejich ovladačů. [5]

3.3 Volba vývojového prostředí

Vývojové prostředí do značné míry ovlivňuje efektivitu, a tedy i s ní spojenou cenu při vytváření výsledného produktu. Proto je nutné, aby prostředí vyhovovalo jak z hlediska ergonomie ovládání, tak například i rychlostí odezvy a minimálním množstvím chyb. Vývojové prostředí se zde rozumí software pro tvorbu jednotlivých komponent celého produktu – PCB, firmware, software, schéma zapojení součástek zařízení.

3.3.1 Software pro návrh plošných spojů

Pro návrh plošného spoje emulátoru a jeho ovládacího panelu byl zvolen CadSoft Eagle. Jedná se o prověřený komerční produkt, který

[13] http://www.mono-project.com/Main_Page

lze ale použít v omezené funkčnosti zdarma. Jeho přednosti jsou zejména obsáhlé knihovny součástek, pouzder, integrovaný autorouter, který lze využít při vytváření signálních cest dle logického zapojení součástek, export do mnoha druhů vrtaček a fréz, snadné vytváření nových pouzder součástek a značek a podobně.

3.3.2 Software pro programování firmware procesoru

Vývojové prostředí bylo zvoleno AVR Studio 4. Jedná se o ne zcela uživatelsky přívětivé prostředí, vzhledem k jeho novější verzi 5, ale zato velmi stabilní a podporuje projekt WinAVR. Má integrovaný simulátor, který ve verzi 2 částečně podporuje i některé periferie procesoru, jako je například USART. Jeho editor lze nahradit jakýmkoliv textovým editorem.

3.3.3 Software pro programování řídícího software emulátoru

Volba vývojového software je vzhledem k použité platformě MS Windows a programovacího jazyka C# jednoznačná – MS Visual C#, který je součástí vývojového balíku MS Visual Studio 2007.

4 DISKETOVÉ MECHANIKY

Disketová mechanika je zařízení, které je schopno načítat a ukládat data na disketové médium a dále data zpracovávat do formy přijatelné pro zařízení, které mechaniku ovládá. Pokud zjednodušíme vztah mezi řídícím zařízením a zařízením disketové mechaniky o její řadič[14], je to zpravidla počítač, ale stejně tak se může jednat o mikrokontrolér, strojní výrobní robot, hudební nástroje, jako jsou syntetizátory, samplery a podobně.

Disketové mechaniky jsou v současné době velmi minoritními hráči na poli úložných zařízení, ale doposud jsou používány. Od své první realizace prošly dlouhým vývojem a v Tab. 4.1 jsou uvedeny některé typy. Hlavními parametry jsou velikost, hustota záznamu dat a z toho vyplývající kapacita.

Tab. 4.1: Seznam parametrů některých disketových systémů

	Specifikace	360 KB 5.25"	1.2 MB 5.25"	720 KB 3.5"	1.44 MB 3.5"	2.88 MB 3.5"
Mechanika	Počet hlav / povrchů	2	2	2	2	2
	Rychlost rotace motorku	300 ot/min	360 ot/min	300 ot/min	300 ot/min	300 ot/min
Řadič	Přenosová rychlost	250 Kbits/s	500 Kbits/s	250 Kbits/s	500 Kbits/s	1 Mbits/s
Médium	Hustota stop	48	96	135	135	135
	Hustota bitů [bit na palec]	5,876	9,869	8,717	17,434	34,868
	Název hustoty	Double Density (DD)	High Density (HD)	Double Density (DD)	High Density (HD)	Extra-High Density (ED)
Geometrie	Stop	40	80	80	80	80
	Sektorů na stopu	9	15	9	18	36
	Celkem sektorů na disk	720	2,4	1,44	2,88	5,76
Souborový systém	Velikost clusteru	2	1	2	1	2
	Maximální počet adresářových záznamů v kořeni	112	224	112	224	448
Kapacita	Kapacita bez formátování	~480 KB	~ 1.6 MB	~1 MB	~2 MB	~4 MB
	Formátovaná kapacita [kbit]	360	1,2	720	1,44	2,88
	Formátovaná kapacita [kByte]	368,64	1,228,800	737,28	1,474,560	2,949,120
	Bajtů pro souborový systém	6,144	14,848	7,168	16,896	17,408
	Celk. kapacita pro uživ. Data [B]	362,496	1,213,952	730,112	1,457,664	2,931,712
	Celková kapacita uživ. data [kB]	0.346	1.158	0.696	1.390	2.796

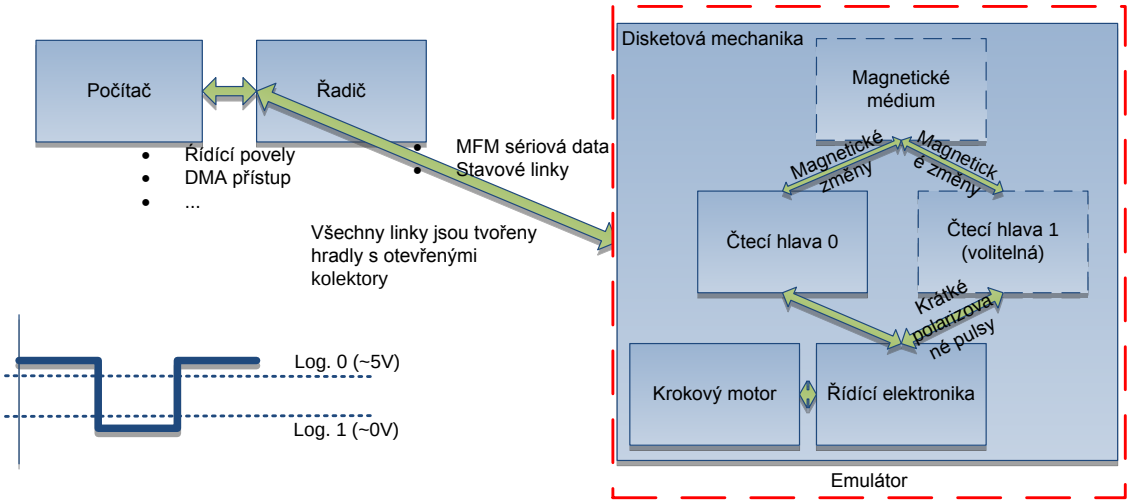
[6]

[14] Elektronický obvod, který ovládá a komunikuje s elektronickými, potažmo mechanickými jednotkami disketové mechaniky

V požadavku na realizaci emulátoru je uvedeno použití 1,44 MB a 720 kB formátu. Tyto dva formáty se od sebe liší především přenosovou rychlostí surových dat (MFM kódovaná původní data) mezi mechanikou a řadičem, a počtem sektorů. Při fyzickém porovnání obou médií jsou u 720 kB formátu data rozložena při poloviční počtu sektorů na stopě ve stejné délce, respektive intervaly vyčítání bitů jsou dvakrát delší, než u 1,44 MB formátu.

4.1 Princip fungování disketové mechaniky

Pro další pochopení tématu je nutné uvést některá fakta. Proto se v dalším textu budeme věnovat mechanikám odpovídajícím specifikaci IBM 2.0/1.0 a rozhraním PC34.



Obr. 4.1: Zjednodušené funkční blokové schéma disketového systému

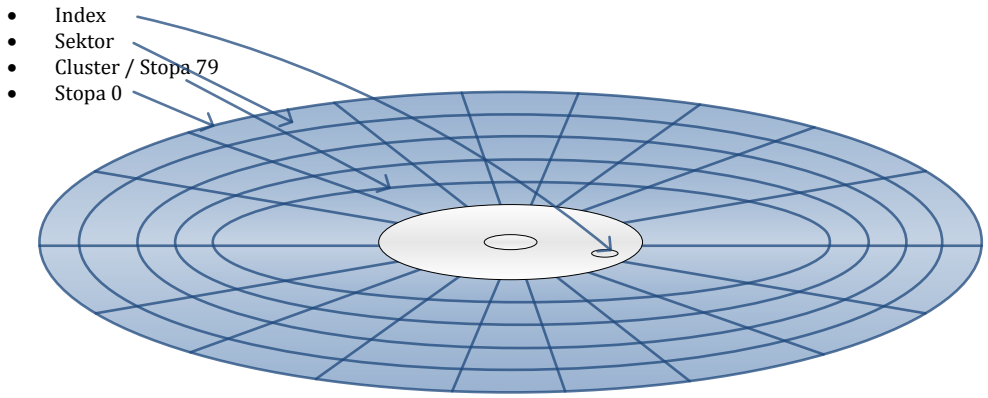
Na Obr. 4.1 je zjednodušené blokové schéma disketové mechaniky. Disketová mechanika se sestává:

- z čtecí/zapisovací hlavičky, motorku, který s nimi pohybuje mezi jednotlivými stopami;
- řídící elektroniky a vložného média – pružného disku.

Řídící elektronika tvoří stavový automat a na základě povelů od řadiče prostřednictvím stavových linek mechaniky se přesunuje do jednotlivých stavů – čtení, zápis, vyhledávání stopy, stand-by. [7]

Disketové vložné médium je tvořeno plastovým kotoučem, který je uzavřený do pevného obalu, vystlaného jemnou textilií, ve kterém se otáčí. Ten ho chrání před mechanickým poškozením. V obalu je vyřezán otvor kolmý na tečnu kruž-

nice stopy od středu po okraj, ve kterém se může pohybovat čtecí/zapisovací hlavička.



Obr. 4.2: Disketové magnetické médium

Na tomto médiu jsou uloženy soustředně jednotlivé datové stopy. Stopy jsou číslovány od 0 do n, kde stopa „00“ je umístěna na vnějším okraji média. V jednotlivých stopách jsou za sebou uloženy sektory, ty můžou být ve speciálních případech číslovány nesouvisle, například 1, 4, 7, 10, 13, 16, 19, (…), 2, 5, 8, 11. Toto rozložení sektorů dává zpracujícímu zařízení delší čas, což může čtení urychlit. Ve standardních MS-DOS formátech je ale pořadí od 1 do n s krokem 1. [8]

Média mohou být dvouvrstvá a jednovrstvá. V případě dvouvrstvých musí mít disketová mechanika dvě hlavy – pro každou vrstvu jedna. Jeden cluster obsahuje tolik stop, kolik je jich fyzicky téměř nad sebou – pokud má médium dvě strany, pak má jeden cluster dvě stopy. Stopy nejsou fyzicky přímo nad sebou, ale jsou posunuty o určitou vzdálenost, aby mezi nimi nedocházelo k interferencím.

Sektory na dřívějších médiích byly od sebe odděleny dírami ve stopách, před každým sektorem jedna – tzv. „hardsectored“. U současných médií jsou sektory „softsectored“, jsou rozlišeny datovými značkami, které jsou uloženy ve stopách spolu s daty. Tyto značky pak rozeznává řadič.

Další podstatný prvek zde je tzv. „Index“. Je to značka, která indikuje začátek stopy na médiu. Je tvořena otvorem na vnitřní straně média a rotuje nad fotozávorou, která ho snímá odraz od určité části obalu kruhového média.

4.1.1 Způsob zápisu dat

Data jsou na médium zapisována a čtena sériově. Z podstaty diskety je patrné, že nikdy nemůže dojít ke čtení/zápisu dokonale konstantní rychlostí, protože reálný motor, který rotuje s médiem, vykazuje určité kolísání rychlosti otáček. Proto je nutné nějakým způsobem zajistit, aby řadič rozeznal polohu právě čteného bitu. Tím je vložení hodinového signálu do dat. Jsou různé způsoby, jak toho docílit a v této práci zmíníme jeden z nich – kódování MFM[15]. [9]

4.1.2 Kódování dat - MFM

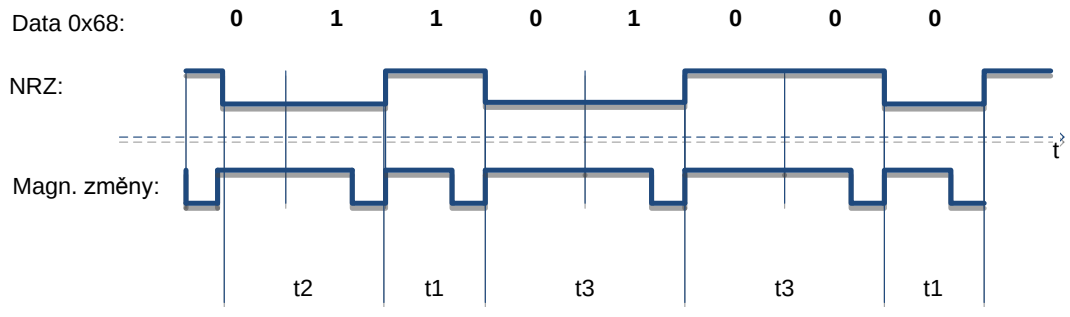
Vychází z FM kódování, ale liší se poloviční délkou zakódovaných dat (magnetických změn). Kódovaná data jsou dělena na buňky, které tvoří jednu datovou jednotku. Každá buňka má na počátku hodinový prvek, na konci datový prvek - ten vždy odpovídá datové hodnotě. Maximální počet po sobě jdoucích log.1 prvků je 0, maximální počet log.0 prvků je 3 – poté následuje vždy buď hodinový anebo datový log.1 prvek. Data `x, y, z` jsou kódována jako `(x, x NOR`[16] `y, y, y NOR z, z….)`. Více bude zřejmý zápis pomocí této tabulky:

Tab. 4.2: Kódovací tabulka MFM

Data	Kód
0	x0
1	01

Z Tab. 4.2 plyne, že každý datový bit má přiřazenou dvoubitovou kódovou značku, výsledný datový objem je tedy dvojnásobný.

Pokud je data bit 0, `x` znamená negovanou hodnotu předchozího dat. bitu. Na Obr. 4.3 je zakódovaná sekvence 68_{16}.



Obr. 4.3:Sekvence 0x68 zakódovaná pomocí MFM

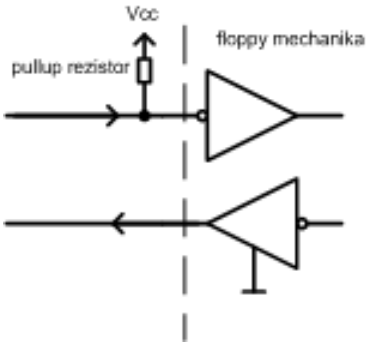
[15] Modified frequency modulation
[16] Negated OR – negovaný logický součet

Časy t_1, t_2, t_3 představují časové intervaly výskytu log. 1, t_1 a t_3 tvoří dvě různé frekvence výskytu datových jedniček v bitových sekvencích 01010101 a 11111111 resp. 00000000 v původním datovém streamu. Čas t_2 je pak přechod mezi nimi. [9]

Tab. 4.3: Časové intervaly log.1 při různých frekvencích

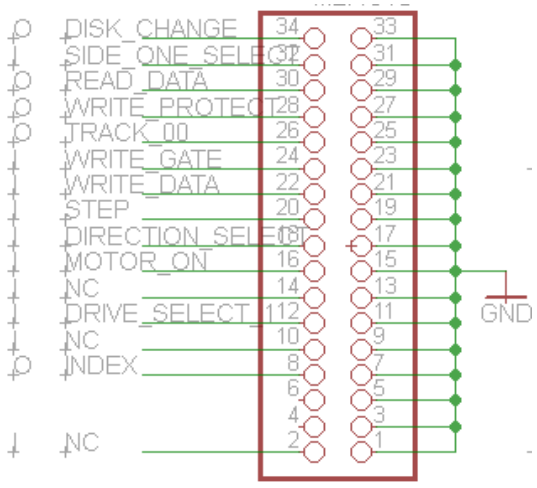
		Bitrate	
		250kbps	500kbps
t1	[µs]	2	4
t2		3	6
t3		4	8

4.2 Komunikační rozhraní mechaniky

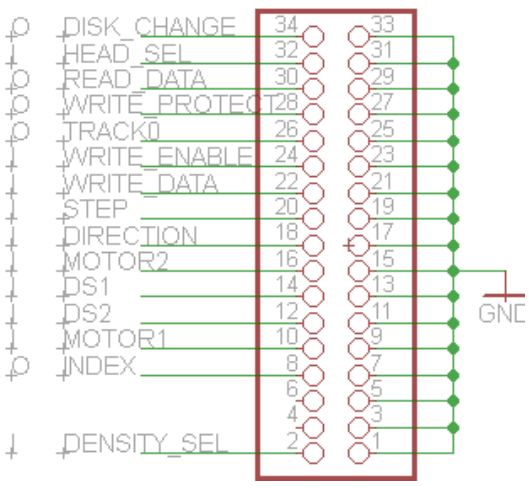


Disketová mechanika je řízena stavovými linkami. Tyto linky určují okamžitý stav, ve kterém se mechanika nachází – není zde žádná paměť, ani sekvenční obvody. Datové přenosy probíhají přes sériové linky – jak zápis, tak přenos. Všechny výstupní linky obsluhují hradla s otevřeným kolektorem, tzn. ve stavu log. 0 je linka ve stavu „vysoké impedance“, log. 1 otevřený kolektor – téměř nulová impedance.

U 3,5“ mechanik můžeme najít dva typy konektorů – jsou to Shugart 34 a IBM PC 34. Liší se rozmístěním linek a nejsou navzájem kompatibilní. V rámci této práce zmíníme pouze IBM PC 34 konektor a některé jeho linky. [10]



Obr. 4.4: Konektor SHUGART 34



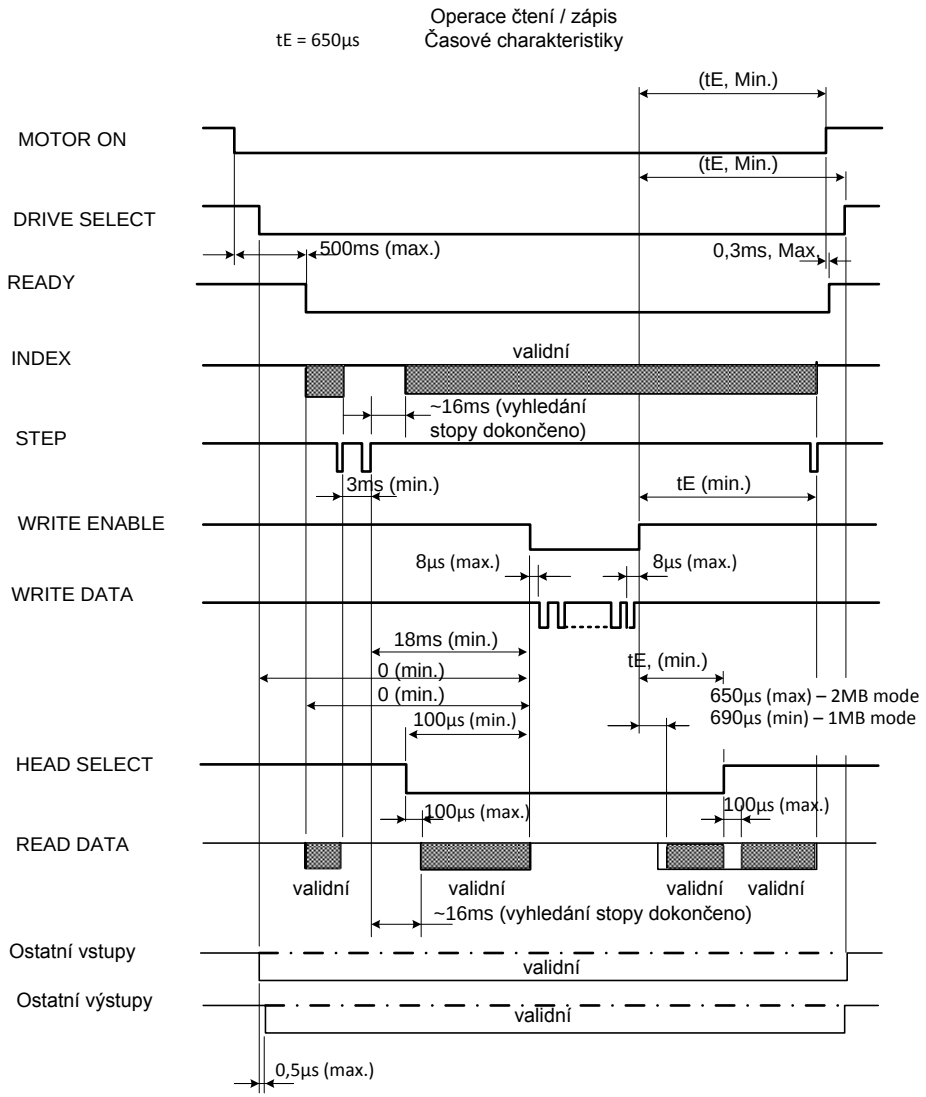
Obr. 4.5: Konektor IBM PC34

Úrovně linek jsou „zvedány“ tzv. „pullup“ rezistory na jedné straně komunikačního řetězce a nulovány sepnutím k zemi na druhé straně. To znamená, že datové symboly jsou přenášeny přes komunikační médium proudově.

Tab. 4.4: Některé důležité signály rozraní IBM PC 34

Pin	Význam
Head select	Vybírá hlavičku mechaniky; 0…vrchní, 1…spodní (volitelná)
Read data	Linka pro sériový přenos dat - krátké pulsy ze čtecí cívky jsou tvarovány a vystavovány zde
Track 00	Indikuje pozici hlavičky nad stopou 0
Write data	Linka pro zápis dat
Step	Posun hlavičky mezi jednotlivými stopami - v kombinaci s Direction
Direction	Směr posunu hlavičky
Index	Indikuje značku diskety Index nad fotozávorou mechaniky – logický začátek stopy

Na Obr. 4.6 jsou znázorněny časové charakteristiky linek, které by měly být dodrženy – v opačném případě hrozí například nezaregistrování pulsu řadičem, přeskočení vystavovacího mechanismu mechaniky na jinou stopu, chybné vstupní/výstupní data a podobně. V praxi tyto časovací mechanismy nebyly doposud standardizovány. [10]



Obr. 4.6: Časový operační diagram 3,5" disketové mechaniky

4.3 Souborový systém FAT12

FAT12 je souborový systém, který je považován v dnešní době za zastaralý. Používá se především na disketách a na některých svazcích MS Windows 2000. Má několik nevýhod, největší z nich je fakt, že používá 12 bitová datová slova, která jsou v počítačích se sběrnicemi o šířce násobků osmi komplikovaně zpracovávána.

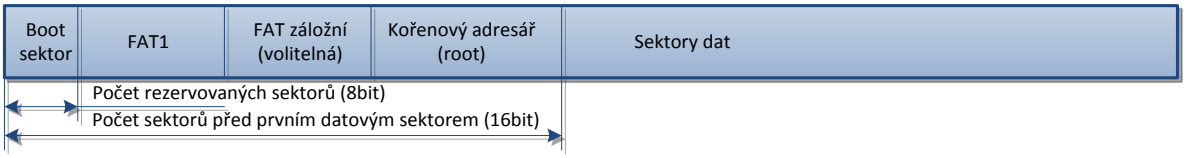
FAT12 používá 12 bitů dlouhé záznamy ve FAT[17] tabulce a každý tento záznam odpovídá jednomu clusteru na médiu. Z toho plyne, že maximální délka souboru, kterou je tento systém schopen zaadresovat, je při využití celé FAT tabulky 2^{12} * <délka clusteru> Bajtů. V praxi je často potřeba využít tabulku pro uložení

[17] File Access Table – tabulka přístupu k souborům na médiu

více souborů, a tak je délka souboru dělena několikanásobně dvěma. FAT12 dělí médium na několik sekcí:

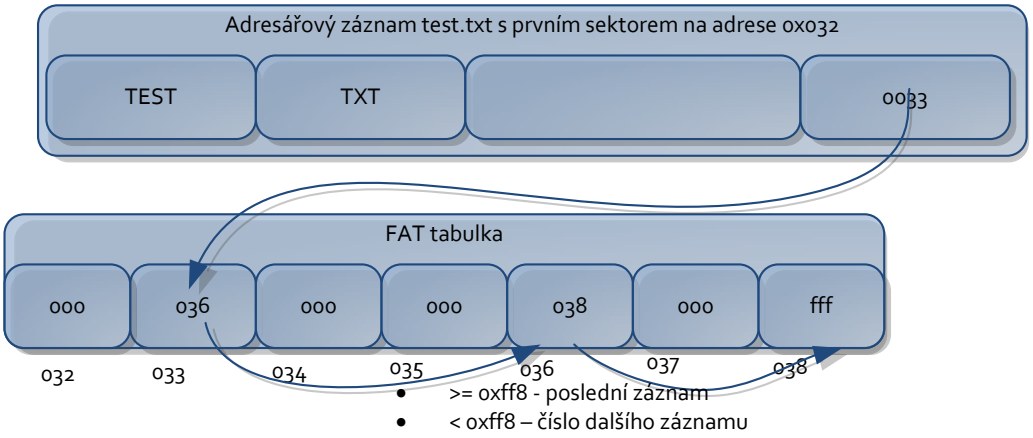
- Boot sektor [512 B] – jsou zde uloženy všechny klíčové informace o fyzickém médiu, umístění FAT tabulek, typ média, a podobně;
- 1. FAT tabulka;
- 2. FAT tabulka – volitelná, slouží jako záložní v případě poškození 1. FAT;
- kořenový adresář – obsahuje až 512 záznamů každý o délce 32 B;
- datová sekce – zbylé sektory, adresovatelné FAT tabulkou.

FAT12 má omezen počet adresářových (souborových) záznamů na maximálně 512.



Obr. 4.7: Rozložení sekcí na FAT12 médiu

FAT 12 původně využíval schéma označení souborů 8.3 – 8 znaků pro jméno, 3 znaky pro příponu souboru. Později bylo zavedeno rozšíření, které toto omezení ruší a za využití dalších adresářových záznamů lze rozšířit až na 24*256 znaků.

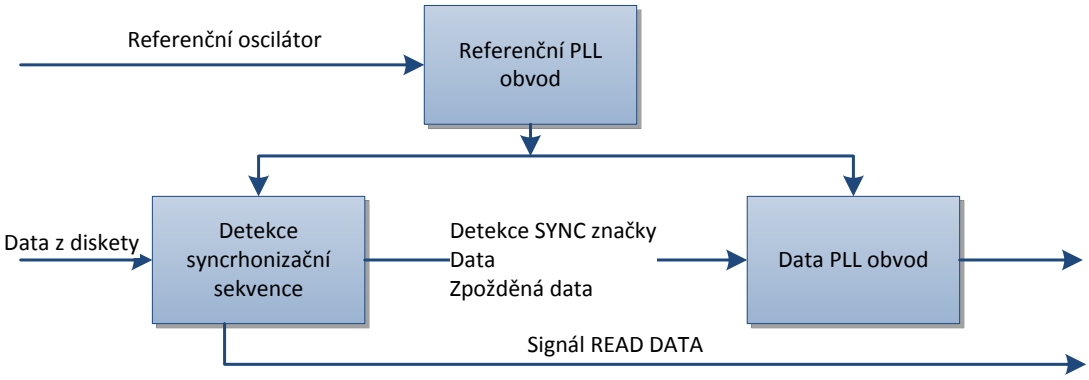


Obr. 4.8: Princip odkazů ve FAT12

Na Obr. 4.8 je znázorněna způsob nalezení a sestavení souboru ve FAT12 systému. Jedná se v podstatě o vázaný seznam. [11],[12]

4.4 Řadiče disketových mechanik

Řadič disketové mechaniky je elektronický obvod, který se stará o komunikaci s hostitelským zařízením. Zpracovává a vykonává příkazy, čte a zapisuje data, generuje žádosti o přerušení, kontroluje validitu dat pomocí CRC[18], případně přistupuje skrz DMA na hostitelskou sběrnici a přenáší data. V rámci této práce je jeho nejzajímavější funkce řízení disketové mechaniky, (de)kódování dat MFM. Obsahují také tzv. „PLL[19]“ regulační obvod, který má za úkol synchronizovat data z média a jeho hodinový signál s interním hodinovým signálem, podle kterého se určuje poloha v právě zpracovávaných datech a za pomocí obvodu pro separování dat tyto data extrahovat.



Obr. 4.9: Separátor dat

Na Obr. 4.9 je znázorněn separátor. Obsahuje dva obvody PLL – referenční a datový. Referenční slouží k nastavení operačního bodu datového PLL – eliminuje vlivy zkreslení, způsobené např. změnou teploty, napájecího napětí. [13]

Známé disketové řadiče:

- Intel 82077AA
- NEC µPD765
- Texas Instruments TMS990x
- Western Digital WD279x

[18] Cyclic redundancy check – cyklický redundantní součet
[19] Phase lock loop

4.5 Formáty fyzických dat disketových médií

V kap. 4.4 byla zmíněna tzv. „synchronizační značka“. Tyto značky souvisí s formátem dat na disketovém médiu a v této kapitole se s ním seznámíme o trochu blíže.

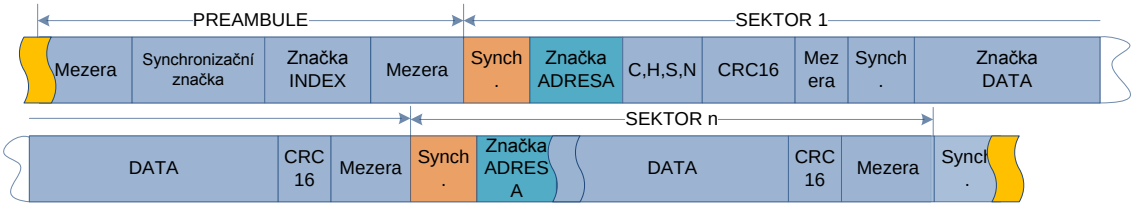
Médium je tvořeno nosnou hmotou, na kterou je nanesen feromagnetický materiál – musí být schopen po relativně dlouhou dobu uchovat magnetické pole – např. oxid železitý. Disketová mechanika ukládá data zapisovací hlavičkou s cívkou, ve které se po průchodu proudu indukuje magnetické pole, kolmé na směr protékání elektrického proudu. Směr proudu tedy určuje magnetickou polarizaci částeček. Zpětné čtení probíhá právě naopak – ve čtecí hlavičce se indukuje velmi malé polarizované napětí, které po zesílení zesilovačem reprezentuje orientaci částečky.

Pro vychýlení magnetického pole od hlavičky v jednom bodě – v místě nad médiem – je v hlavičce mezera vyplněná neferomagnetickým materiálem. [14]

Jak jsme se již dozvěděli v kap. 4.1.2 jsou data ukládána pomocí změn. Tyto změny jsou reprezentovány právě změnou polarizace mezi po sobě jdoucími částečkami.

Formáty takto ukládaných dat jsou v rámci zadání práce: IBM 1.0 MB a IBM 2.0 MB.

V [15] je kompletní struktura datové stopy, zde si ukážeme jeho zkrácenou variantu. Emulátor mechaniky musí tento formát bezezbytku podporovat.



Obr. 4.10: Struktura stopy IBM x.0MB

- Začátek stopy indukuje „Preambule“. Poté následuje 9/18 sektorů.
- Mezera – slouží k synchronizaci separovacího obvodu v řadiči s daty z média.
- Značka – návěstí identifikující následující sekvenci dat; značky jsou 3 - index, adresa a data. Každá z nich je z pohledu MFM kódování a pravidla

(1,3) nevalidní – obsahují chybějící hodinový signál, což slouží separátoru dat pro synchronizaci. Separátor pozice nalezne a „zamkne“ se k nim.

C, H, S, N – číslo cylindru, hlavičky, sektoru a délka uživatelských dat (DATA).

Přehled hodnot formátů použitých v rámci této práce (doplňuje Tab. 4.1):

Tab. 4.5: Kapacity používaných formátů

		IBM 2.0 MB	IBM 1.0 MB
Sektorů / stopa		18	9
Přenosová rychlost	**[kbps]**	500	250
Počet stran		2	2
Počet stop		80	80
Délka sektoru	**[B]**	512	512
Celková kapacita	**[B]**	1474560	737280

[13]

4.5.1 Adresování LBA[20] vs CHS[21]

Tato dvě adresování základních datových bloků na médiu se liší stupněm abstrakce. Zatímco CHS popisuje reálné prvky zařízení, které jsou požadovány při čtení ze zařízení a je při něm potřeba znát geometrické vlastnosti média, LBA adresování udává pouze číslo bloku v lineárním seznamu a často je za ním právě „skryto“ adresování CHS. Disketové mechaniky používají adresování CHS, a tak zde uvedeme i přepočty mezi těmito systémy:

$$C = int(LBA/SPT/HPC); \tag{4.1}$$

$$H = int\left(\left(\frac{LBA}{SPT}\right) modulo\ HPC\right); \tag{4.2}$$

$$S = (LBA\ modulo\ SPT) + 1; \tag{4.3}$$

$$LBA = \left(((C * HPC) + H) * SPT\right) + S - 1\ . \tag{4.4}$$

LBA...blok; *C*...cylindr; *H*...hlavička; *S*...sektor; *HPC*...počet hlaviček (stran); *SPT*...sektorů na stopu. [16]

[20] Linear block address
[21] Cyllinder –head– sector

5 PRAKTICKÁ ČÁST

Praktická část této práce poslední a nejdůležitější částí. Týká se konkrétní implementace výše uvedeného v jeden produkt, který bude emulovat funkce 3,5“ disketové mechaniky a komunikovat s řídícím počítačem. Všechna nalezená řešení jsou v souladu s požadavky uvedenými v kap. 1.1 . Je zde podrobně popsán princip fungování emulátoru, použité úložiště, komunikace mezi hostitelským počítačem a emulátorem, navržená aplikace a pro lepší názornost jsou uvedeny fotografické ukázky. V rámci práce byly navrženy, a v dalším textu jsou popsány, dvě verze s totožnou funkcí lišící se použitým procesorem a jeho periferiemi. Společné rysy jsou uvedeny na začátku kapitoly, dále následují specifika jednotlivých verzí.

5.1 Princip fungování

Emulátor funguje podobně jako disketová mechanika, jako stavový automat. Jeho momentální stav určují stavové linky, zejména: DRIVE SELECT, STEP, DIRECTION, MOTOR ENABLE.

Ve výchozím stavu, kdy jsou linky ve stavu log.1, je emulátor v klidovém stavu. V tomto stavu je též možné pracovat s ním pomocí řídící aplikace a reaguje na softwarové příkazy.

V dalším stavu přenosu dat se emulátor ocitne, pokud je splněna podmínka (¬DRIVE SELECT & ¬MOTOR ENABLE) = log. 1. Tehdy začne cyklicky generovat „disketovou“ stopu od indexu po konečnou synchronizační značku. Číslo stopy, a tedy i sektorů, určuje již uvedený vzorec $LBA = F(C,H,S)$. V tomto stavu emulátor reaguje na změny signálů STEP a DIRECTION, které určují pohyb mezi stopami – pokud (¬STEP & ¬DIRECTION) = log.1 zajistí jeden krok z vně disku ke středu, tzn. směr od stopy 00 ke stopě 79. Pokud (¬STEP & DIRECTION) = log.1, pak je směr opačný. [10],[7]

a) Vygenerování datové stopy

Z podstaty diskety je zřejmé, že po zafixování hlavičky nad určitou stopu a sepnutí linek ¬MOTOR ON a ¬DRIVE SELECT začne disketová mechanika generovat pulsy odpovídající zaznamenaným datům na magnetickém kotouči. Tento proces bylo potřeba dokonale promítnout do emulátoru.

5.2 Ovládání stavových linek

Z podstaty disketové jednotky je zřejmé, že některé stavové linky se mění na základě dat v magnetickém kotouči a na jejich vzájemné poloze s čtecí hlavičkou. Jsou to linky INDEX, RDATA. V emulátoru jsou tyto změny řešeny vložením odpovídajících „značek“ přímo do datové stopy. Při čtení dat pak zároveň čteme příkazy – pokud jsou rozpoznány.

Rozpoznání určují speciální změny v mezi daty, které odporují MFM kódování – více jedniček za sebou. To je při použití kódování MFM nepřípustné, proto je zaručeno, že značky se nebudou shodovat se samotnými daty. Značky jsou pro:

- nastavení signálu INDEX na log.1 `0x11000000`
- nastavení signálu INDEX na log.0 `0x01100000`
- „přetočení“ čtecího ukazatele na začátek `0x01110000`

Tyto značky eliminují jakékoliv výpočty v samotné obsluze, která musí být co nejkratší.

5.3 Generování RDATA impulsů na lince FLOPPY

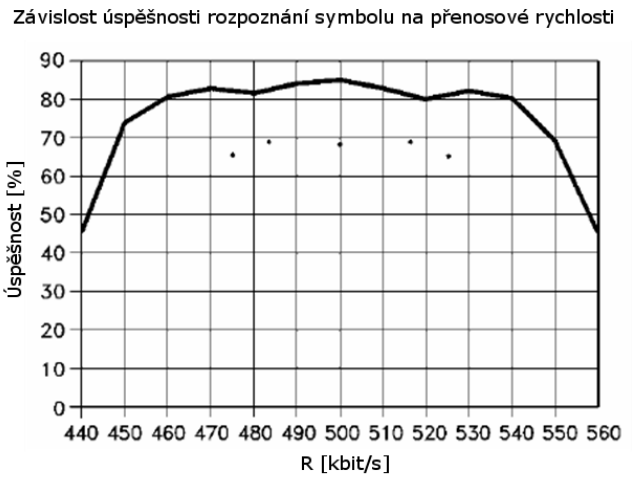
Výstupní data (log. úrovně) musí přicházet v určitém sledu a především při minimálním jitteru frekvence. Z principu floppy mechaniky je nutné generovat stopu signálem RDATA bez přerušení. Toleranci jitteru definují specifika použitého FDC a jeho data-separátoru[22] a závisí na rychlosti bitového toku. Je definována jako:

$$\frac{S}{\frac{1}{4}T} \cdot 100 \ [\%] \qquad (5.1)$$

přičemž

- S … maximální doba bit. posunu vůči původní pozici,
- T … perioda nominální přen.rychlosti. [17]

[22] Obvod použitý k separaci hodinového a datového signálu z MFM datového proudu.



Obr. 5.1: Tolerance jitteru separátorem v závislosti na rychlosti RDATA. Zdroj: [17]

Za předpokladu, že budeme chtít úspěšnost detekce symbolu min. 85 % při průměrné rychlosti 500 kbps, po dosazení do (5.1) nám vyjde maximální jitter 425 ns. Perioda hodinového cyklu MCU je při 16 MHz $1/16.10^6$=62,5 ns.

V implementaci emulátoru verze 2 byl zvolen jako generující prvek USART modul v MSPI módu. V tomto módu lze generovat pouze datové pulsy bez řídicích. Toto řešení je velmi výhodné, protože převod z paralelních dat na sériová se děje na pozadí a nezaměstnává tak procesor. Modul má paměť na 8 (až 16) bitů, to nám dává čas mezi plněními paměti z datového proudu přibližně $8*T_{SP} - T_{SP}$ je doba signálového prvku $1/500.10^3$ sekund – což je 16 µs či 16 µs/62,5 ns = 256 hodinových cyklů. V tomto čase se dějí obslužné operace, jako je dekódování a výkon instrukcí vložených do datového proudu. MCU tak není zcela vytížený a je zde prostor pro další instrukce. Plnění probíhá v *nekonečné*[23] smyčce.

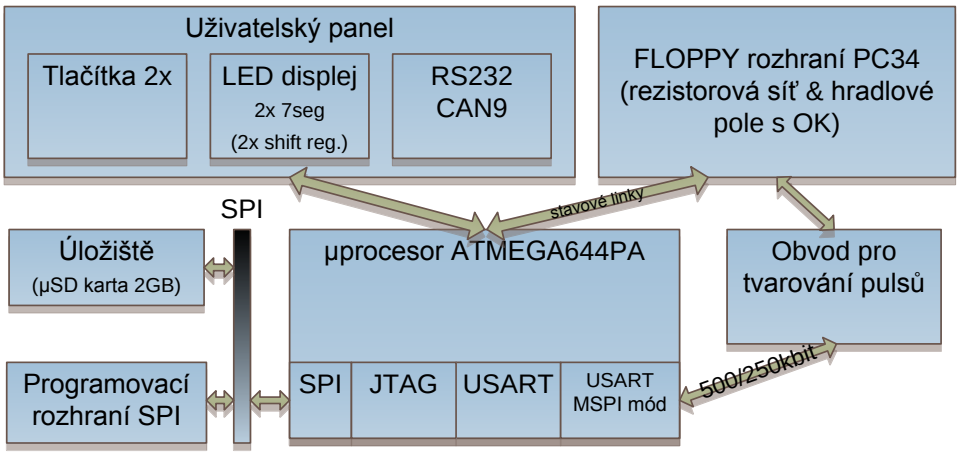
Naproti tomu verze 1 bez externí paměti a možnosti předkódování dat. proudu používá systém přerušení, který kontroluje stav paměti USART modulu a pokud se blíží její vyprázdnění, zavolá se „doplňující“ rutina. Tato rutina se vykoná v rozsahu 6 - 9 stroj. taktů (viz [18]) včetně první instrukce pro naplnění paměti z kdykoliv dostupného registru. Těchto 9 instrukcí je již „na pokraji“ námi stanovené tolerance jitteru (8,5 taktů). Návrhu bez předgenerování stopy nenahrává ani to, že je mezi plněními na popředí vykonáváno čtení z úložiště, které má též své prodlevy a jitter může ovlivnit o několik řádů více, než je odskok do obsluhy přerušení.

[23] Do té doby, dokud je platná současná adresa stopy.

5.4 Emulátor verze 1

5.4.1 Blokové schéma

Než se pustíme do samotného popisu, je vhodné seznámit se s blokovým schématem emulátoru. Jednotlivé bloky představují buď samotné funkční celky na PCB[24], anebo integrované periferie do pouzdra mikroprocesoru procesoru.



Obr. 5.2: Blokové schéma emulátoru

Na Obr. 5.2 je znázorněno blokové schéma emulátoru. Jednotlivé funkční bloky emulátoru jsou bez výjimky propojeny sériovými sběrnicemi, vyjma stavových linek, kteréžto mají pochopitelně svoje signální vodiče. Úložiště ve formě µSD karty komunikuje s procesorem skrz rozhraní SPI[25] na F_CPU[26]/2. Karta je napájena SS 3,3 V, z toho důvodu jsou na výstup mikroprocesoru zařazeny napěťové děliče 1 : 2. Komunikace přes sériové rozhraní RS232 s hostitelským PC nevyžaduje signalizační linky, je řízena výhradně na aplikační vrstvě.

V příloze na Obr. 0.3 je uvedeno podrobné schéma zapojení celého emulátoru a Obr. 0.6 čelního uživatelského panelu emulátoru.

5.4.2 Generování stopy emulátorem

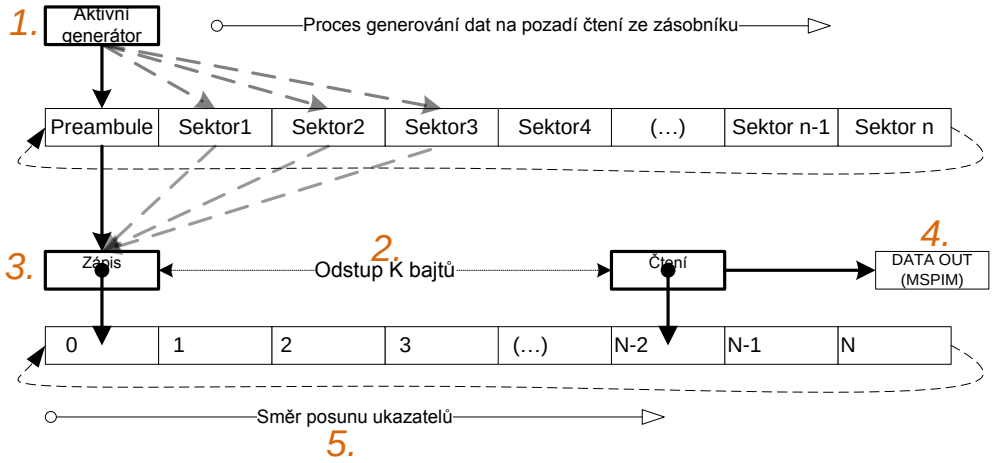
Mikroprocesor ATMEGA644PA má omezenou délku operační paměti na 4096 B. MFM kódovaná stopa diskety IBM 2.0 MB má přesně 100 kbit, z tohoto důvodu ji procesor není schopen předpočítat celou a dále ji jen vysílá. Je proto nutné vytvořit cyklický buffer se dvěmi ukazateli – jeden pro zápis, druhý pro

[24] Printed circuit board – tištěný spoj
[25] Seriál peripheral interface
[26] Frekvence procesoru

čtení. Zapisovací ukazatel nikdy nesmí předběhnout čtecí a zároveň nesmí být dost pomalý na to, aby mu čtecí ukazatel „neutekl“ o přibližně jednu délku bufferu.

Je také nutné zmínit, že proces vyčítání je řízen obsluhou přerušení rozhraní USART v MSPI módu `USART1_TX_vect` (shift i UDR registr jsou prázdné). V tomto módu je k dispozici tzv. „double buffered“ paměť rozhraní a vstupní data mohou být až 2 Bajty, což nám dává k dispozici 16 bitů pro MFM data, tzn. až 16*1 µs (2 µs pro 250 kbps) času, tj. při F_CPU 20 MHz 320 strojových taktů.

Samotný vstup do ISR trvá 8 strojových taktů, kód obsluhy přerušení 15 - 30, tzn. na samotné načtení z karty, zakódování, zapsání do zásobníku a ostatní potřebné operace pro zpracování dalších 16 bitů pro zapsání do USART registru zbývá ~280 taktů. [18]



Obr. 5.3: Kruhový zásobník a I/O operace

Na Obr. 5.3 je znázorněn způsob generování stopy do kruhového zásobníku. Proces je relativně přímočarý. Zjednodušeně si ho zde popíšeme:

1. nastavení aktivního generátoru úseku, tj. data, která se budou dále ukládat do zásobníku;
2. vyčkání do doby, než je čtecí pointer dostatečně daleko (zapisovací pointer je rychlejší);
3. zápis dat generátorem do zásobníku;
4. vyčtení dat ze zásobníku pomocí ISR v intervalech 1 nebo 2 µs (500/250 kbit);

5. posun čtecího pointeru a připravení dalších dat do registrů procesoru pro další skok do ISR;
6. bod 1.

Pro doplnění a správné pochopení funkce ještě zmíníme, že čtecí ukazatel při spuštění generování stopy nejprve předběhne zapisovací ukazatel o jednu délku zásobníku a tam „stojí“, viz výše.

a) Urychlení obsluhy přerušení

Jak jsme dověděli v dodatku k zadání, pro obsluhu přerušení byl použit Assembler. Toto řešení pomohlo zrychlit obsluhu přibližně na polovinu, oproti „C“ variantě. Zároveň byla využita technika namapování globální proměnné v jazyce C na registr procesoru pomocí konstrukce:

```
register uint8_t * dataBajt1 asm("R2").
```

Toto umožnilo mít permanentně k dispozici jednak proměnnou pro uložení 2 následujících Bajtů pro ISR a jednak čtecí ukazatel. Obě tyto datové položky jsou využity v ISR. Odpadá tím nutnost „uklizení“ registrů pomocí instrukcí `push,pop` před vstupem do ISR – každá po 2 strojových taktech.

Aby tyto registry nebyly použity nikde jinde v programu, je nutné dát kompilátoru flag „`-ffixed-<číslo registru>`“. Zde je nutné upozornit na použití předkompilovaných knihoven, které jsou na námi zakázaných registrech závislé, a nastane konflikt. V této práci žádné předkompilované knihovny použity nejsou.

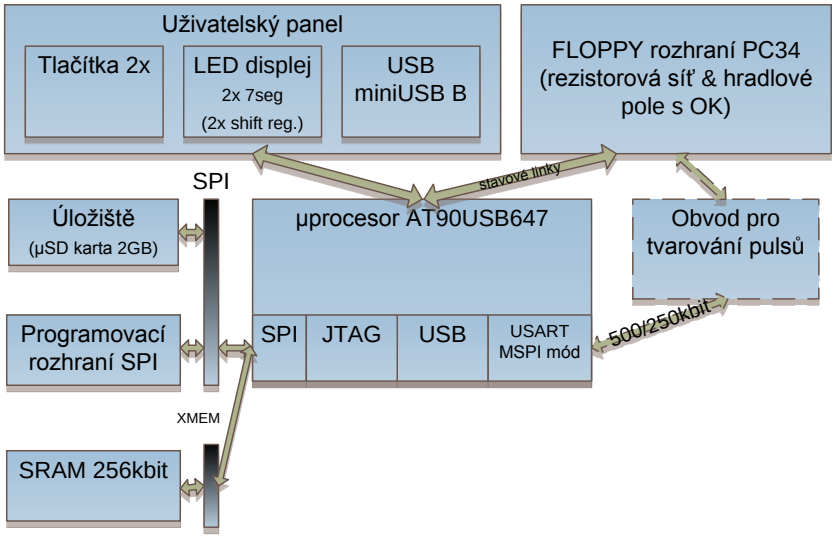
Též je vhodné v manuálu ke konkrétnímu překladači ověřit, které registry není vhodné takto vyřadit.

5.5 Emulátor verze 2

V předchozí kapitole byl naznačen postup realizace generování stopy emulátoru pomocí kruhového zásobníku. Tento postup generování je vykoupen cenou realizace, která nemusí obsahovat externí paměť (a latch[27] hradlo) pro uložení kompletní stopy image diskety a je tak zdánlivě levnější. Toto je hlavní změna ve

[27] Logické hradlo s pamětí

verzi 2. Pokud do systému vložíme externí paměť, bude struktura systému vypadat následovně:



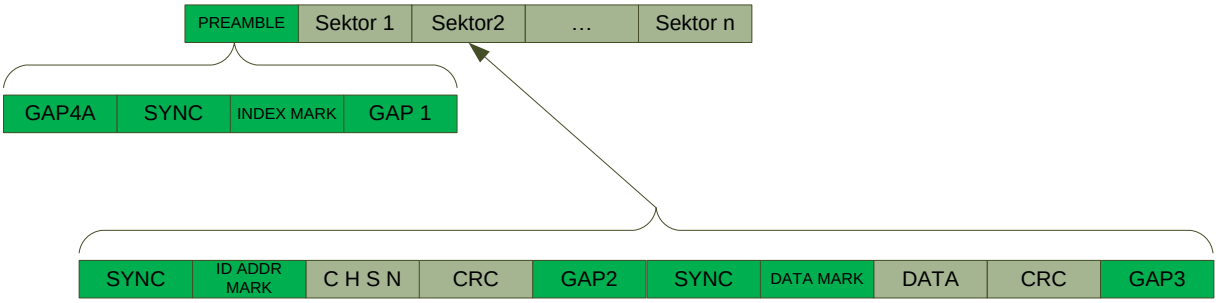
Obr. 5.4: Blokové schéma emulátoru s MCU AT90USB647

5.5.1 Generování datové stopy

Díky použití dostatečně velké externí paměti je možné datovou stopu předkódovat v okamžiku požadavku (změna CHS adresy) a poté ji číst jako cyklický lineární buffer. Toto řešení je velmi stabilní co do výstupní bitové frekvence – nevznikají zde žádné prodlevy způsobené datovým úložištěm, tak i následnou konzistencí stopy – nemůže nastat případ pře/podtečení bufferu.

Naměřené zkreslení výstupní frekvence změn hodnot na datové FDD lince RDATA je pak závislé pouze na přesnosti krystalového oscilátoru.

Prodleva mezi požadavkem na změnu CHS adresy a vygenerováním datové stopy je větší než v případě předchozí verze, protože je třeba načíst všechny sektory ve stopě v jeden okamžik, ale po vygenerování jsou data neměnná až do další změny CHS adresy.



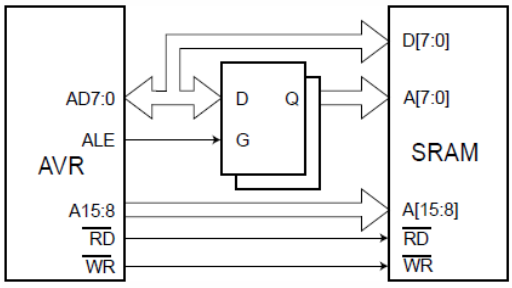
Obr. 5.5: Šablona stopy předpřipravená v programové paměti MCU

Na Obr. 5.5 jsou zelenou barvou vyznačena ta data, která jsou uložená v programové paměti MCU a zakódována MFM. Předkódování omezí operaci generování stopy v emulátoru při běhu a změně CHS pouze na zkopírování a zakódování dat z úložiště na příslušné offsety v SRAM paměti. Ostatní součásti stopy jsou již „ozrcadleny“ předem, ty se v průběhu práce emulátoru nemění.

Celý systém generování je synchronní s chodem hlavního programu (není zde použito přerušení pro plnění výstupní SPI sběrnice) kromě reakce na CHS signály, přicházející do systému v nepředvídatelných a relativně k frekvenci MCU krátkých okamžicích a je nutné je obsluhovat okamžitě pomocí systému přerušení.

5.5.2 Externí paměť

Na rozdíl od Obr. 5.2 je zde nejpatrnější změna oproti verzi 1 v použití externí SRAM paměti. Paměť je řešena klasicky po blocích 1 B a je připojena prostřednictvím XMEM[28] rozhraní. Schéma implementace je vyznačeno na Obr. 5.6. Velkou výhodou tohoto řešení je možnost použití jedné sběrnice pro adresaci paměti a zároveň pro čtení/zápis dat na datovou sběrnici. Toho je dosaženo vložením „latch“ hradla, jehož vstup je třístavový (HI,LO,vysoká impedance). Je proto možné v určitém stavu (vys. imp.) neblokovat sběrnici a umožnit její použití i jinými periferiemi. Toto se děje u XMEM rozhraní. Procesoru střídavě mění funkci sběrnice pro potřeby adresování/čtení a zápis dat pomocí pomocných řídicích linek ve smyslu:



Obr. 5.6: AVR XMEM rozhraní. Zdroj [19]

[28] Označení rozšiřující paměťové sběrnice firmy Atmel

nastavit sběrnici na výstupní s hodnotou adresy (spodních 8b) na latch, změnit vstup latch hradla na vysokou impedanci a poté číst (¬WE = 1, sběrnice na vstup), či zapisovat data do SRAM. [19]

5.6 Přístup uživatele k obsahu – souborům v úložišti

Dle zadání je nutné podporovat změnu a nahrávání souborů a manipulaci s datovými sektory. První popsaná verze tyto operace provádí přes RS232 linku pomocí řídicí aplikace. Práce probíhá offline. Při manipulaci se soubory se vytvoří „transakce" – tj. na začátku se data zkopírují jako sobory z emulátoru do PC, uživatel provede patřičné změny a data se vloží zpět do emulátoru. V případě manipulace se sektory se změny dějí online v reálném čase přes řídicí linku.

V druhé verzi probíhají operace s médiem online, protože je fyzický FAT12 svazek namapován prostřednictvím USB MASS STORAGE CDC[29] rozhraní k řídicímu počítači a lze tak přistupovat k obsahu jako ke kterémukoliv jinému fyzickému oddílu pomocí SCSI[30] příkazů – chcete-li přes průzkumník souborů. SCSI protokol je veřejně známý protokol, proto v této práci nebude dále rozebírán. Příkazy, které byly do emulátoru implementovány, jsou:

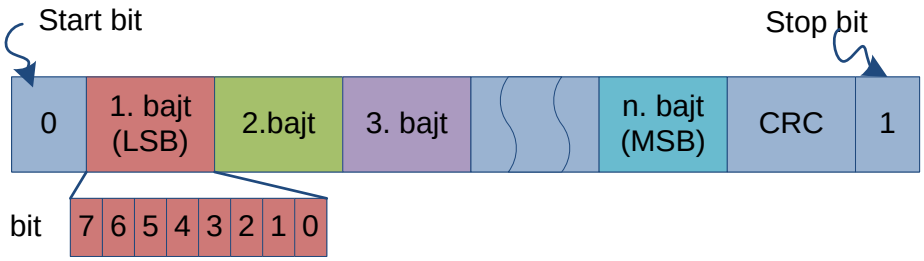
```
SCSI_CMD_INQUIRY
SCSI_CMD_REQUEST_SENSE
SCSI_CMD_READ_CAPACITY_10
SCSI_CMD_SEND_DIAGNOSTIC
SCSI_CMD_WRITE_10
SCSI_CMD_READ_10
SCSI_CMD_MODE_SENSE_6
SCSI_CMD_START_STOP
SCSI_CMD_TEST_UNIT_READY
SCSI_CMD_PREVENT_ALLOW_MEDIUM_REMOVAL
SCSI_CMD_VERIFY_10
```

[29] Communication device class – metoda přenosu dat přes USB datový kanál zakončený *endpointy*.
[30] Small Computer System Interface – protokol pro práci s paměťovými zařízeními.

5.7 Úložiště emulátoru

Jak již bylo zmíněno, emulátor používá jako úložiště paměťovou kartu. Tato karta může pracovat ve dvou módech komunikace – SPI a 2/4 bit mód. V rámci této práce je použit pouze SPI mód, proto zde ostatní popsány nebudou.

SPI mód je sériový přenos, kde jsou jednotlivé příkazy přenášeny přes jednu linku. Příkazy jsou tvořeny 6 bajty – číslo příkazu (1), parametr (4), CRC7 (1).



Obr. 5.7: Struktura SPI datové jednotky

Na Obr. 5.7 je znázorněna struktura jednotky přenášené přes SPI rozhraní. Zajímavostí je datový příkaz a jeho kontrolní součet CRC7. Je totiž nezvykle 7 bitový, a proto vyžaduje svoji implementaci podle popisu v [20]. V bajtu s CRC7 je přenášen i ukončovací bit, který je 1 pouze v případě, že se jedná o poslední bajt příkazu s CRC7. Zbylé bity tvoří právě CRC7 posunuté o jeden bit vlevo.

Jako další datová jednotka jsou zde samotná uživatelská data nebo obsahy registrů karty a můžou mít různé délky. Na konci jsou však vždy opatřeny CCIT CRC16 kontrolním součtem.

SPI rozhraní je krom adresných linek třílinkové – hodinový signál, data in a data out. V tomto režimu je vždy jedno zařízení v režimu „Master“ a druhé „Slave“ – doslovně otrok, kterému generuje hodinový signál právě Master zařízení. Slave pracuje pouze v této době. Po spuštění hodinového signálu začíná přenos dat ze zařízení Slave, ale zároveň také z Master a opačně, pokud je to žádoucí. Po příjmu lze ve specifických aplikacích ověřit CRC, ale zároveň je toto ověřování možno vypustit, pomocí nastavení – v kartě registr „SCR“ v rámci zvýšení propustnosti komunikace. [20]

5.7.1 Přístupová doba

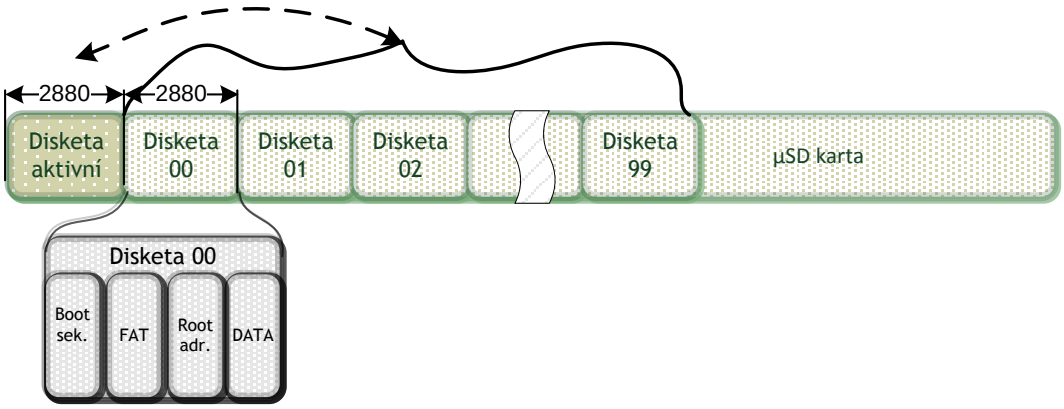
Přístupová doba je doba, o kterou naroste prodleva mezi požadavkem na přístup a jeho obsloužením. Tato doba je často ovlivněna adresou, ze které se data čtou,

a také prodlením vyčítacího interního mechanismu. Závisí také na tom, zda data čteme sekvenčně se změnou adresy o minimálním krok, nebo *náhodně* a adresa se mění o (pro médium) předem neznámou dobu. Ve specifikaci SD karty, která je použita jako úložiště, se uvádí maximální doba *až* 100 ms a tuto dobu pro konkrétní model najdeme i v CSD registru přímo na kartě [20], [21]. Je proto nutné s touto dobou počítat i v návrhu a přizpůsobit řešení.

5.7.2 Schéma rozložení obrazů disket

Pro emulátor bylo nutné navrhnout schéma rozložení jednotlivých disket v emulátoru. Disketou je se rozumí soubor 2880 sektorů o délce 512 Bajtů v případě IBM 2.0 MB a 1440 sektorů pro IBM 1.0 MB. V rámci zjednodušení implementace a zrychlení práce emulátoru byly tyto délky sjednoceny na 2880 sektorů, přitom při druhé zmiňované disketě se zbylé sektory nevyužijí. V dnešní době, kdy je poměr cena/kapacita výhodná a kapacita karty oproti kapacitě diskety řádově vyšší, je toto plýtvání přijatelné. Lze tedy násobkem 2880 jednoznačně určit, kde začíná a končí disketa n na kartě.

Diskety jsou uloženy těsně za sebou od sektoru 2880 a dále. Sektory 0 – 2779 slouží pro dočasné uložení současně aktivní diskety. Toto řešení s sebou nese zrychlení práce, neboť není nutné adresovat celý LBA prostor, ale pouze sektory 0 - 2779, na což stačí adresa o délce 2*8 bitů.



Obr. 5.8: Rozložení disket na paměťové kartě

Pokud je disketa označena jako aktivní ovládacím panelem nebo řídicí aplikací, je zkopírována na pozici „Disketa aktivní“. Existuje mnoho způsobů, jak toto rozložení vyřešit – tento je však nejjednodušší a výpočetně nejefektivnější. Uvedeme i dvě nevýhody tohoto řešení – je zde jistá prodleva (konkrétně ~20 sek.),

než dojde k překopírování diskety na aktivní pozici. Tímto zápisem je také karta degradována používáním, ale vzhledem ke garantovanému počtu zápisu u FLASH[31] pamětí na sektor v řádu 100 000 je toto zanedbatelné. [20]

5.8 Řídicí software

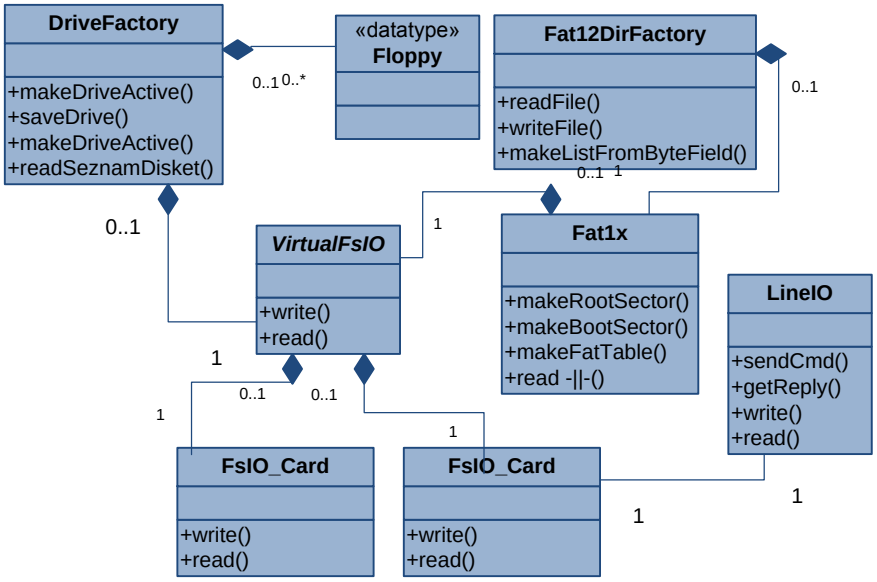
Pro potřeby ovládání a správu obsahu emulátoru byla naprogramována aplikace v jazyce C#. Tato aplikace komunikuje s emulátorem RS232 linkou, virtuální sériovou a MASS STORAGE kanálem nad USB CDC rozhraním, paměťovou kartou, pokud je ve čtečce, formátuje kartu, upravuje jednotlivé sektory, čte FAT tabulky a stahuje data ve formě binárních sektorů, souborů FAT12 či celých adresářů disket. V příloze E nalezneme několik obrazovek řídicí aplikace. Také zde nalezneme na Obr. 0.10 obrazovku „Computer“ systému Windows 7 při připojené kartě emulátoru v USB čtečce a vidíme, že se tváří jako obyčejná disketa s FAT12 formátem souborů. Hlavní sekce aplikace jsou:

- formátování,
- správa karty,
- správa obsahu diskety.

Lze tedy procházet kořenovým adresářem a stahovat jednotlivé soubory jak z karty, tak i RS232. Aplikace byla navržena objektově s rozvrstvenou hierarchií funkcí, lze tedy pouze „vyměnit“ čtecí vrstvu (RS232, USB disk apod), přičemž objekty nad touto čtecí vrstvou dále pracují a o této změně nemusí vědět.

Neuvádíme zde kompletní popis řídicí aplikace, ale popíšeme nejpodstatnější objekty.

[31] Nevolatilní paměť s omezeným počtem zápisů



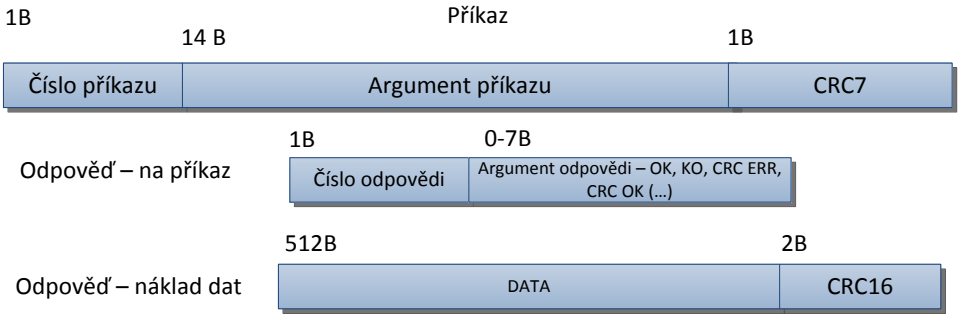
Obr. 5.9: Diagram hierarchie některých tříd aplikace

Na Obr. 5.9 jsou zjednodušeně znázorněny některé třídy a jejich hierarchie. Jedná se o ty nejdůležitější, které se starají například o převod binárních dat na třídy reprezentující souborový systém FAT(12/16), jednotlivé diskety, seznamy disket a podobně. Též je možné převést zpět tyto třídy do fyzické binární podoby, a uložit je na médium skrz souborové rozhraní – zde třída `VirtualFsIO`.

Dále stojí za zmínku třída `Fat12DirFactory`, která umí z binárních dat vyčíst celou adresářovou strukturu, tj. kde se dané soubory a adresáře nachází a jejich velikost. Poté následuje je jejich vyčtení pomocí FAT tabulky.

5.8.1 Komunikační pakety aplikace – emulátor

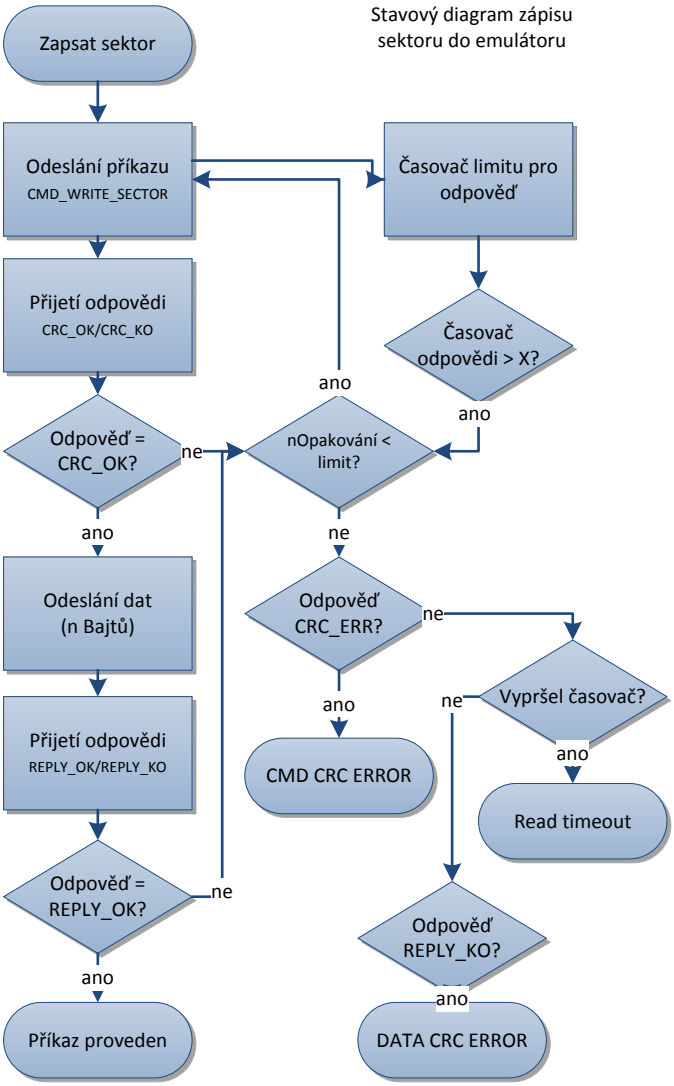
Aplikace s emulátorem komunikuje datovými pakety skrz kanál detekující chybu v datech. Detekce je dvojího typu – u přenášených dat z emulátoru pomocí CRC16 a u řídicích příkazů z počítače pak CRC7. Toto kódování bylo zvoleno, protože již bylo použito ve formě kontroly validity SPI příkazů karty, a též proto, že pro jeho přenos stačí pouze <= 8 bitů. Zároveň ale nedochází k přetečení jeho funkčního rozsahu, protože datové pakety příkazu jsou dlouhé <= 128 bitů. Existují 3 typy paketů: datový o délce 514 B, řídící, kde je odpověď 8 B a příkaz 16 B viz Obr. 5.10.



Obr. 5.10: Tvar komunikačního datového paketu

5.8.2 Operace zápisu sektoru do emulátoru

V předchozích kapitolách jsme se seznámili s řídícími a datovými pakety, jejich tvarem a parametry. Nyní je na místě uvést alespoň jednu operaci, která je všechny využívá. Tato operace je interně nazvaná „CMD_WRITE_SECTOR“ (Obr. 5.11), neboli zápis sektoru. Ostatní operace probíhají v podobném sledu.



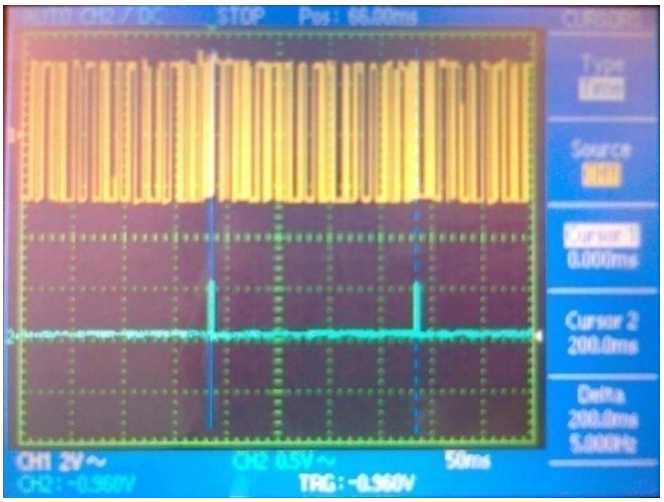
Obr. 5.11: Stavový diagram příkazu k zápisu sektoru do emulátoru

5.9 Doporučené další směřování vývoje

Žádný výrobek není dokonalý a platí to i o emulátoru probíraném v rámci této práce. Je zde mnoho dalších cest, kterými lze vylepšit funkčnost zařízení. Mezi nejvýznamnější patří:

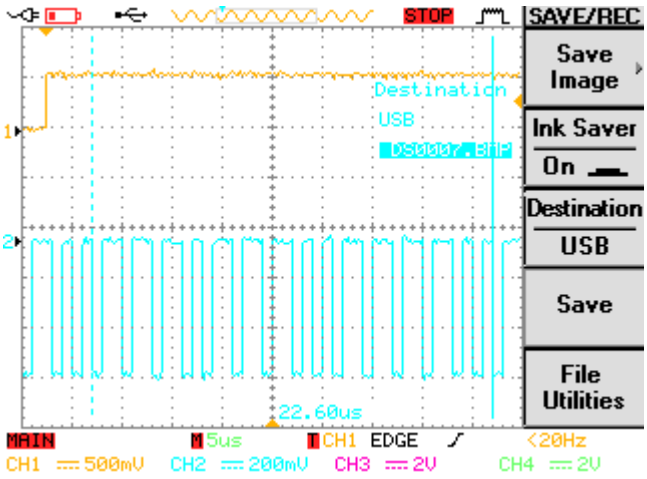
- návrh pevného plastového obalu přímo do pozice 3,5“;
- implementace podpory zápisu za předpokladu, že stávající MCU je dostatečně výkonný.

6 DATA Z REALIZACE EMULÁTORU



Obr. 6.1: Perioda "otáčky" stopy diskety (z emulátoru)

Na Obr. 6.1 je znázorněna perioda otáčky emulátoru. Nahoře jsou datové pulsy linky RDATA, dole pulsy linky INDEX – určuje počátek stopy. Perioda je přesně 200 ms, což odpovídá $f_{otáčení}$ = 5 Hz a rychlosti RAW dat 5*(125000 B)=500 kbps.



Obr. 6.2: MFM data z emulátoru na začátku stopy

Na Obr. 6.2 jsou čitelné pulsy RDATA. Je zde vidět proměnlivá šířka HIGH pulsů, které jsou poté invertovány hradlem s OK, respektive proměnlivá frekvence LOW pulsů, jež je podstatou MFM kódu.

Funkce Emulátoru byla otestována v módu 250/500 kbps na těchto chipsetech (FDC):

Intel 845, Intel 945GC, NVIDIA MCP73V, NVIDIA GeForce 8200.

7 ZÁVĚR

Cílem této práce bylo navrhnout a realizovat emulátor disketové 3,5“ mechaniky. V počátku této práce byly popsány všechny podstatné části, které byly nutné k sestavení a naprogramování emulátoru. Celá problematika je natolik rozsáhlá, že popis byl v rámci zachování informační hodnoty práce redukován na nejpodstatnější fakta při zachování popisu všech částí systému emulace FDD.

Byly analyzovány jednotlivé možnosti řešení – funkce emulátoru a způsobu jeho práce, komunikace s řídicí aplikací a dostupné vývojové nástroje. Také byly teoreticky rozebrány všechny části emulátoru.

V praktické části byl navržen emulátor v podobě schéma zapojení základní desky a čelního panelu a k nim byly navrženy dvouvrstvé prototypové plošné spoje – pro čelní uživatelský panel a pro základní desku emulátoru. Byla naprogramována řídicí aplikace v jazyce C#, podporující všechny požadované operace s určitým uživatelským komfortem, např. konverzi Bajtového pole až do jednotlivých souborů a FAT12 struktur nebo stahování kompletních stromů disket či jednotlivých souborů. Komunikace probíhá po virtuální sériové lince RS232 po USB kanálu, který je sám o sobě zabezpečen proti vzniku chyb. Aplikace také podporuje totožné operace nad paměťovou kartou připojenou lokálně k řídicímu počítači přes téže USB rozhraní a CDC kanál jako MASS STORAGE zařízení.

V průběhu práce byly navrženy dva mírně odlišné prototypy emulátoru. První verze byla popsána v praktické části 5.4 Druhá verze stále podporuje původní příkazy sériového rozraní zabezpečené pomocí CRC a potvrzovacího mechanismu, avšak díky použití USB rozhraní není nutné kanál dále zabezpečovat na aplikační vrstvě a komunikace tak byla zjednodušena a zrychlena. Dále byla použita externí SRAM, která dovoluje uložení celé stopy „diskety“, a tak není nutné data cyklicky číst a kódovat.

Čas pro přepnutí image disket je v druhé verzi 1 s oproti 50 s v první verzi.

V průběhu práce byl zvolen procesor AT90USB647, který na rozdíl od původního MCU ATMEGA644 v návrhu podporuje připojení externí SRAM, MSPI UART mód a USB rozraní. Toto vedlo k odstranění konvertoru log. úrovní

RS232 z návrhu a zvýšení datové propustnosti 4x. Byl odstraněn stabilizátor napětí 3,3 V a nahrazen pasivními prvky.

Bylo implementováno MFM kódování pomocí tabulky, díky kterému zakódování 1 bitu dat trvá průměrně 3,2 strojových cyklů v závislosti na vstupních datech (bylo zjištěno v simulátoru), tj. přibližně ¼ z původního exaktního matematického výpočtu (16 cyklů) procedurou přímo v procesoru.

Emulátor podporuje dva formáty floppy disků - IBM 1.0/2.0, lišící se strukturou stopy a také přenosovými rychlostmi – 250 a 500 kbit/s. Tyto formáty jsou rozeznávány z charakteru dat v boot sektoru FAT12 image floppy disku na úložišti.

Realizovaná konstrukce je vhodná pro umístění do původního 3,5“ slotu pro FDD mechaniky včetně případného krytu.

Velmi se také osvědčilo použití JTAG „in-system“ debuggeru Atmel Dragon při ověřování funkcí emulátoru.

Celkově lze realizaci práce zhodnotit jako proveditelnou, avšak v určitých případech (doba obsluhy přerušení) velmi náročnou na plánování strojového času díky typu použitého procesoru. Realizace je plně funkční a byla testována v reálném provozu bez omezení.

Kód řídicí aplikace a firmware mikrokontroléru je přiložen na CD.

8 LITERATURA

[1] **Nero, Jean-Francois Del.** HxC2001 floppy emulator. *About*. [Online] HxC2001, 9. 10 2011. [Citace: 7. 12 2011.] http://hxc2001.free.fr/floppy_drive_emulator/.

[2] **FLEXIDRIVES.** The Floppyemulator Team. *Floppy Drive Replacement - SD/USB Floppy emulator*. [Online] FLEXIDRIVES, 2011. [Citace: 7. 12 2011.] http://www.floppyemulator.com/.

[3] **Atmel.** AVR508: Migration from ATmega644 to ATMEGA644P. *Atmel*. [Online] 2006. [Citace: 7. 12 2011.] http://www.atmel.com/dyn/resources/prod_documents/doc8038.pdf.

[4] **IBIBILIO.** http://www.ibiblio.org/gferg/ldp/GCC-Inline-Assembly-HOWTO.html. *GCC-Inline-Assembly-HOWTO*. [Online] Sandeep S., 1. 3 2003. [Citace: 7. 12 2011.] http://www.ibiblio.org/gferg/ldp/GCC-Inline-Assembly-HOWTO.html.

[5] **Microsoft.** .NET Downloads, Developer Resources & Case Studies. *Microsoft .NET Framework*. [Online] 2011. [Citace: 14. 12 2011.] http://www.microsoft.com/net.

[6] **M.Kozierok, Charles.** Summary of Floppy disk types and specifications. *www.pcguide.com*. [Online] 17. 4 2001. [Citace: 7. 12 2011.] http://www.pcguide.com/ref/fdd/formatSummary-c.html.

[7] **Samsung.** Oem Manual. *3.5inch DUAL DENSITY MICRO FLOPPY DISK DRIVE*. [Online] 20. 8 2006. [Citace: 7. 12 2011.] http://www.techtravels.org/amiga/SAMSUNG-SFD321B-070103.pdf.

[8] *Diskety a disketové jednotky*. **Khol, Ivan.** 10, Praha : Amaro spol. s r.o., 1989. ISBN 0322 -9572.

[9] **Michael Haardt, Alain Knaff, David C. Niemi.** The Floppy User Guide. *www.moria.de*. [Online] 27. 3 2007. [Citace: 7. 12 2011.] www.moria.de/~michael/floppy/floppy.ps.

[10] **Teac.** TEAC FD-235HF. *MICRO FLOPPY DISK DRIVE.* [Online] 3. 7 2005. [Citace: 8. 12 2011.] www.msc-ge.com/.../teac/FD-235HF-C829.pdf.

[11] **MICROSOFT.** File systems. *Microsoft Technet.* [Online] 2011. [Citace: 10. 12 2011.] http://technet.microsoft.com/en-us/library/cc938950.aspx.

[12] **Schwarz, Thomas.** COEN 252 Computer Forensics. *FAT File Systems.* [Online] Santa Clara University, 2004. [Citace: 10. 12 2011.] http://www.cse.scu.edu/~tschwarz/coen252_04/Lectures/FAT.html.

[13] **Intel.** CHMOS SINGLE-CHIP FLOPPY DISK CONTROLLER. *Intel 82077AA.* [Online] 1. 5 1994. [Citace: 10. 12 2011.] http://www.osdever.net/documents/82077AA_FloppyControllerDatasheet.pdf.

[14] **Ramirez, Michael.** HOW A FLOPPY DISK READS & WRITES INFORMATION. *web.mit.edu.* [Online] 1999. [Citace: 11. 12 2011.] http://web.mit.edu/2.972/www/reports/floppy_drive_read_write/floppy_drive_read_write.html.

[15] **Citizen Systems Europe.** Model: X1DE-00R. *3.5" Micro Floppy Disk Drive Specification.* [Online] 2. 7 2001. [Citace: 9. 12 2011.] http://info-coach.fr/atari/hardware/fd-hard/citizen-x1de00a.pdf.

[16] **Mueller, Scott.** *Upgrading and repairing PCs.* Indianapolis : QUE publishing, 2004. ISBN 0-7897-1670-4.

[17] **Intel.** 82077AA. *CHMOS SINGLE-CHIP FLOPPY DISK CONTROLLER.* [Online] 1. 5 1994. [Citace: 20. 1 2012.] http://www.buchty.net/casio/files/82077.pdf2gfA.

[18] **Atmel Corporation.** ATMEGA644 preliminary. *8bit AVR Microcontroller with 64kB ISP flash.* [Online] 1. 7 2010. [Citace: 11. 12 2011.] http://www.atmel.com/dyn/resources/prod_documents/doc8011.pdf.

[19] —. AT90USB647. [Online] 1. 11 2009. [Citace: 18. 2 2012.] http://www.atmel.com/dyn/resources/prod_documents/7593S.pdf.

[20] **Technical Committee SD Card Association.** SD Specifications. *Physical Layer Simplified Specification.* [Online] 25. 9 2006. [Citace: 11. 12

2011.]
http://www.sandisk.com/Assets/File/OEM/Manuals/SD_SDIO_specsv1.pdfDHnZbR20B8IKNz5xrA.

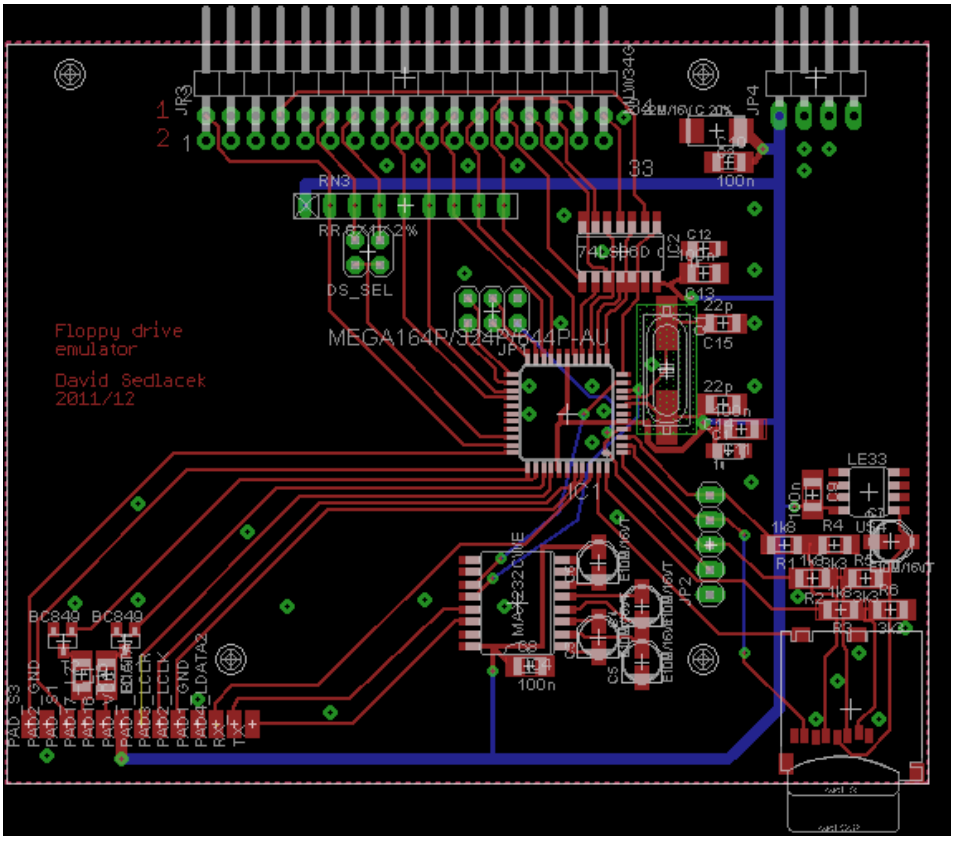
[21] **SANDISK.** SD Card Specification. *Secure Digital Input/Output (SDIO) Card Specification*. [Online] 2000,2001. [Citace: 20. 12 2011.] http://www.sandisk.com/Assets/File/OEM/Manuals/SD_SDIO_specsv1.pdf.

[22] **Mann, Burkhard.** *C pro mikrokontroléry*. Praha : BEN, 2003. ISBN 80-7300-077-6.

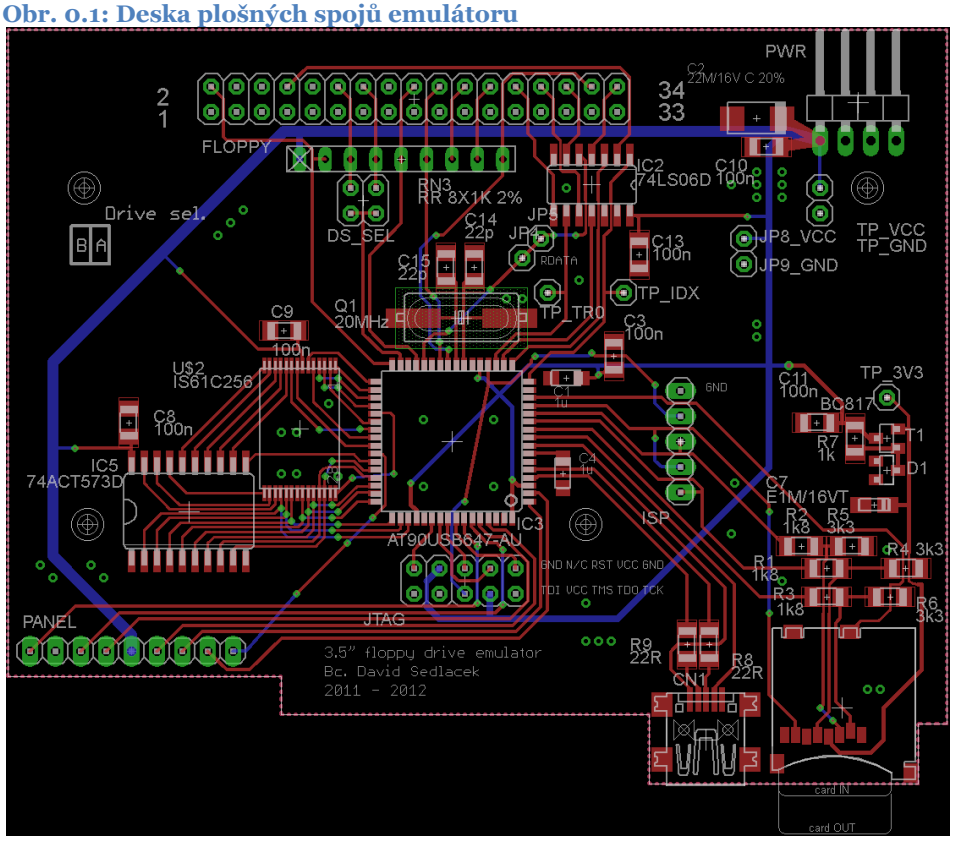
[23] *Multimedia karty a čo s nimi*. **Aradio PE.** 1, Praha : Amaro, 2007, Sv. 4.

PŘÍLOHY

A Návrh desky plošného spoje emulátoru

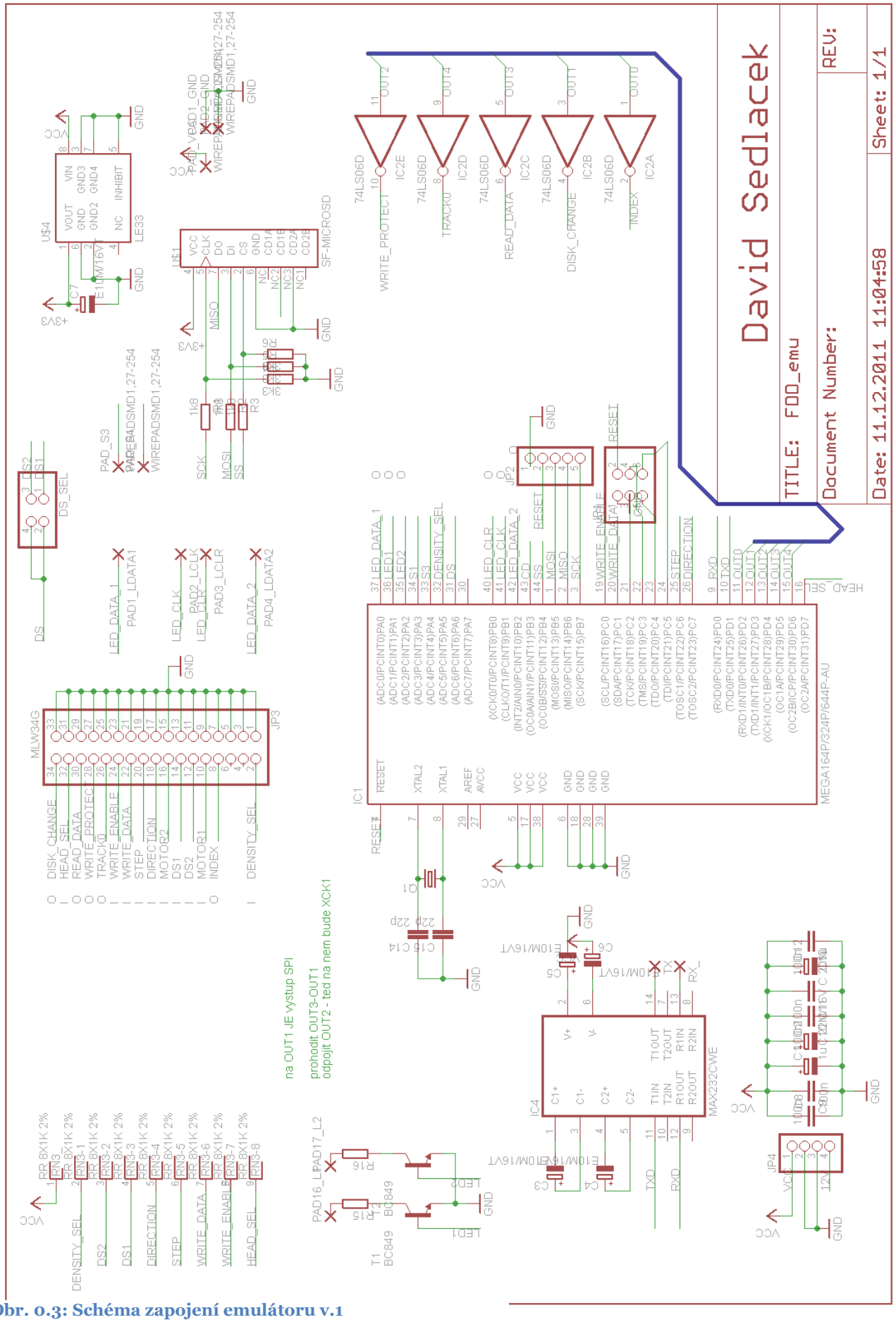


Obr. 0.1: Deska plošných spojů emulátoru

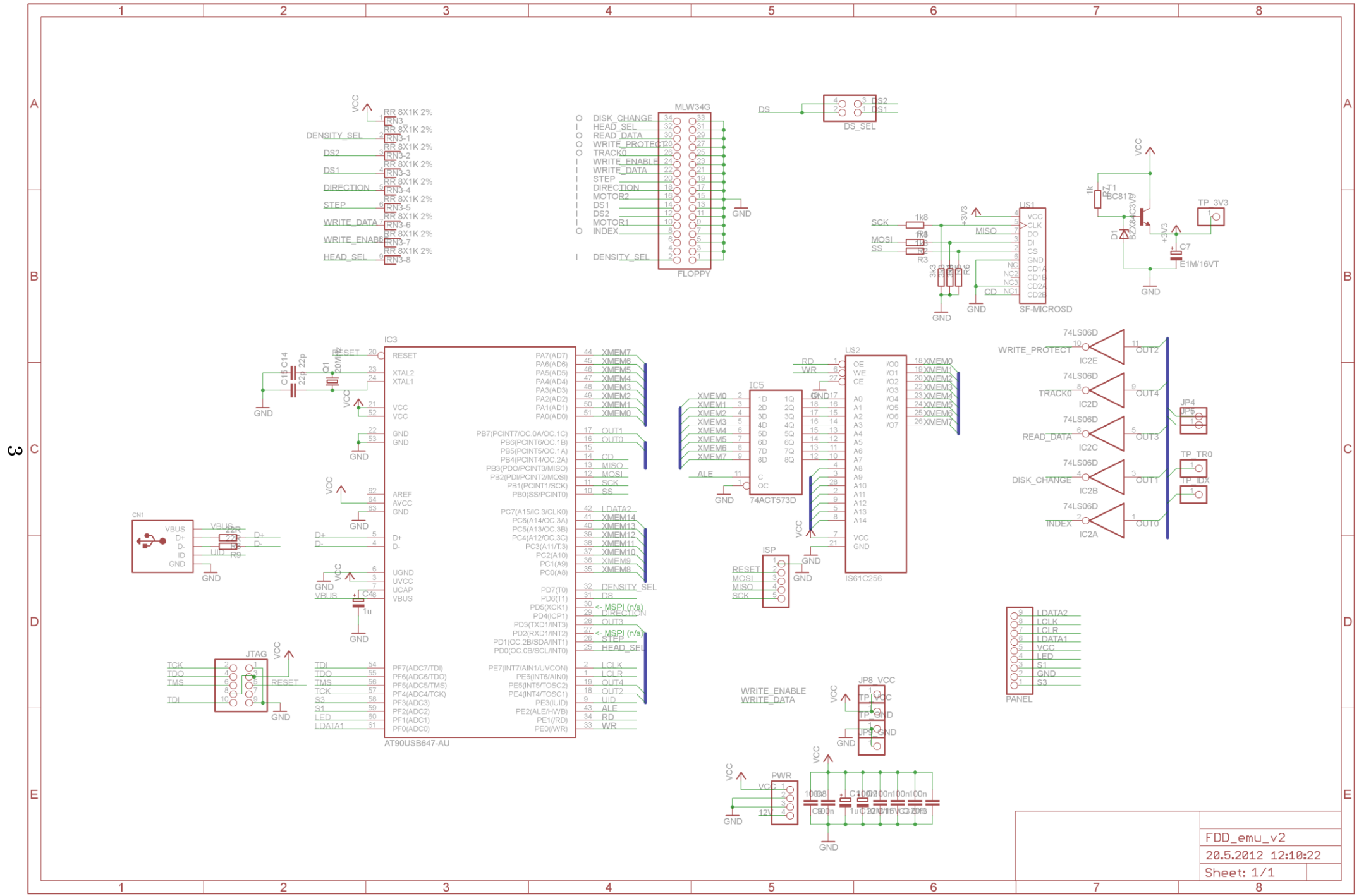


Obr. 0.2: Deska plošných spojů v.2

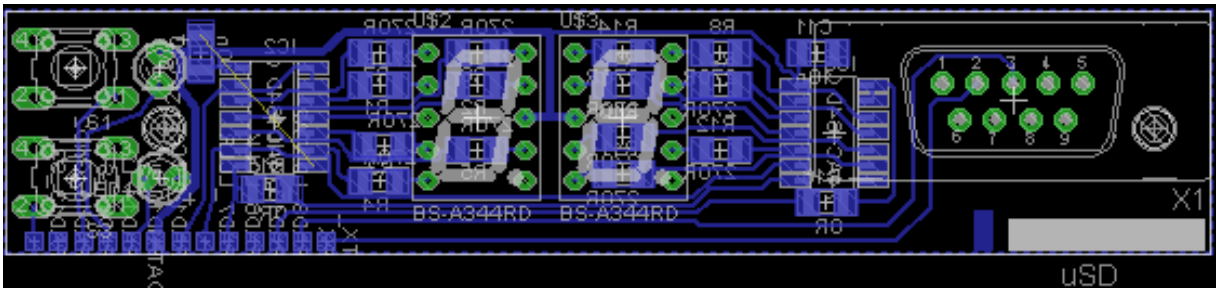
B Schéma zapojení emulátoru



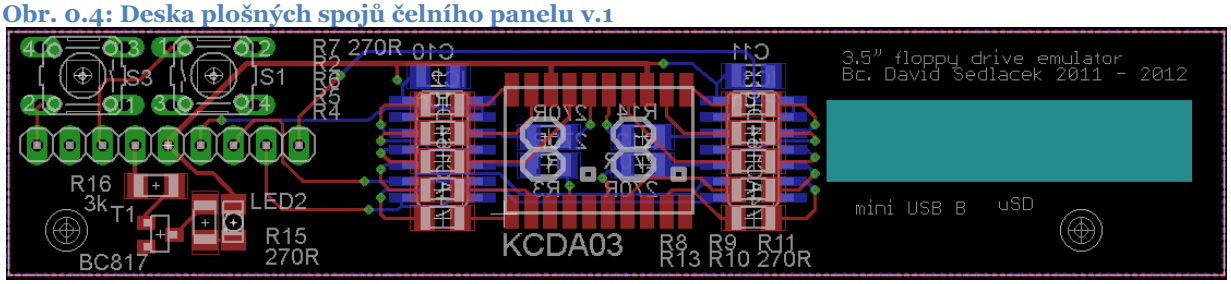
Obr. 0.3: Schéma zapojení emulátoru v.1


RR 8X1K 2%
RN3
RN3-1
RN3-2
RN3-3
RN3-4
RN3-5
RN3-6
RN3-7
RN3-8
DENSITY_SEL
DS2
DS1
DIRECTION
STEP
WRITE_DATA
WRITE_ENABLE
HEAD_SEL
MLW34G
DISK_CHANGE
HEAD_SEL
READ_DATA
WRITE_PROTECT
TRACK0
WRITE_ENABLE
WRITE_DATA
STEP
DIRECTION
MOTOR2
DS1
DS2
MOTOR1
INDEX
DENSITY_SEL
FLOPPY
GND
DS
DS2
DS1
DS_SEL
SCK
MOSI
SS
1k8
R3
3k3
R6
+3V3
MISO
US1
VCC
CLK
DO
DI
CS
GND
CD1A
CD1B
CD2A
CD2B
NC
NC2
NC3
NC1
CD
SF-MICROSD
1k
T1
BC817
D1
BZX84C3V9
C7
E1M/16VT
TP_3V3
IC3
RESET
20MHz
Q1
C15
C14
22p
XTAL2
XTAL1
VCC
GND
AREF
AVCC
PA7(AD7)
PA6(AD6)
PA5(AD5)
PA4(AD4)
PA3(AD3)
PA2(AD2)
PA1(AD1)
PA0(AD0)
XMEM7
XMEM6
XMEM5
XMEM4
XMEM3
XMEM2
XMEM1
XMEM0
PB7(PCINT7/OC.0A/OC.1C)
PB6(PCINT6/OC.1B)
PB5(PCINT5/OC.1A)
PB4(PCINT4/OC.2A)
PB3(PDO/PCINT3/MISO)
PB2(PDI/PCINT2/MOSI)
PB1(PCINT1/SCK)
PB0(SS/PCINT0)
OUT1
OUT0
CD
MISO
MOSI
SCK
SS
PC7(A15/IC.3/CLK0)
PC6(A14/OC.3A)
PC5(A13/OC.3B)
PC4(A12/OC.3C)
PC3(A11/T.3)
PC2(A10)
PC1(A9)
PC0(A8)
LDATA2
XMEM14
XMEM13
XMEM12
XMEM11
XMEM10
XMEM9
XMEM8
DENSITY_SEL
DS
PD7(T0)
PD6(T1)
PD5(XCK1)
PD4(ICP1)
PD3(TXD1/INT3)
PD2(RXD1/INT2)
PD1(OC.2B/SDA/INT1)
PD0(OC.0B/SCL/INT0)
<- MSPI (n/a)
DIRECTION
OUT3
<- MSPI (n/a)
STEP
HEAD_SEL
UGND
UVCC
UCAP
VBUS
C16
1u
D+
D-
PE7(INT7/AIN1/UVCON)
PE6(INT6/AIN0)
PE5(INT5/TOSC2)
PE4(INT4/TOSC1)
PE3(IUID)
PE2(ALE/HWB)
PE1(/RD)
PE0(/WR)
LCLK
LCLR
OUT4
OUT2
UID
ALE
RD
WR
AT90USB647-AU
CN1
VBUS
ID
R7
R8
R9
22R
JTAG
TCK
TDO
TMS
TDI
RESET
PF7(ADC7/TDI)
PF6(ADC6/TDO)
PF5(ADC5/TMS)
PF4(ADC4/TCK)
PF3(ADC3)
PF2(ADC2)
PF1(ADC1)
PF0(ADC0)
S3
S1
LED
LDATA1
IC5
74ACT573D
XMEM0
XMEM1
XMEM2
XMEM3
XMEM4
XMEM5
XMEM6
XMEM7
ALE
US2
IS61C256
OE
WE
CE
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
A13
A14
I/O0
I/O1
I/O2
I/O3
I/O4
I/O5
I/O6
I/O7
ISP
RESET
MOSI
MISO
SCK
74LS06D
IC2E
IC2D
IC2C
IC2B
IC2A
WRITE_PROTECT
TRACK0
READ_DATA
DISK_CHANGE
INDEX
OUT2
OUT4
OUT3
OUT1
OUT0
JP4
JP5
TP_TR0
TP_IDX
PANEL
LDATA2
LCLK
LCLR
LDATA1
VCC
LED
S1
GND
S3
WRITE_ENABLE
WRITE_DATA
JP8_VCC
TP_VCC
TP_GND
JP9_GND
PWR
12V
C100n
C1u
FDD_emu_v2
20.5.2012 12:10:22
Sheet: 1/1

C Návrh desky plošného spoje čelního panelu



Obr. 0.4: Deska plošných spojů čelního panelu v.1



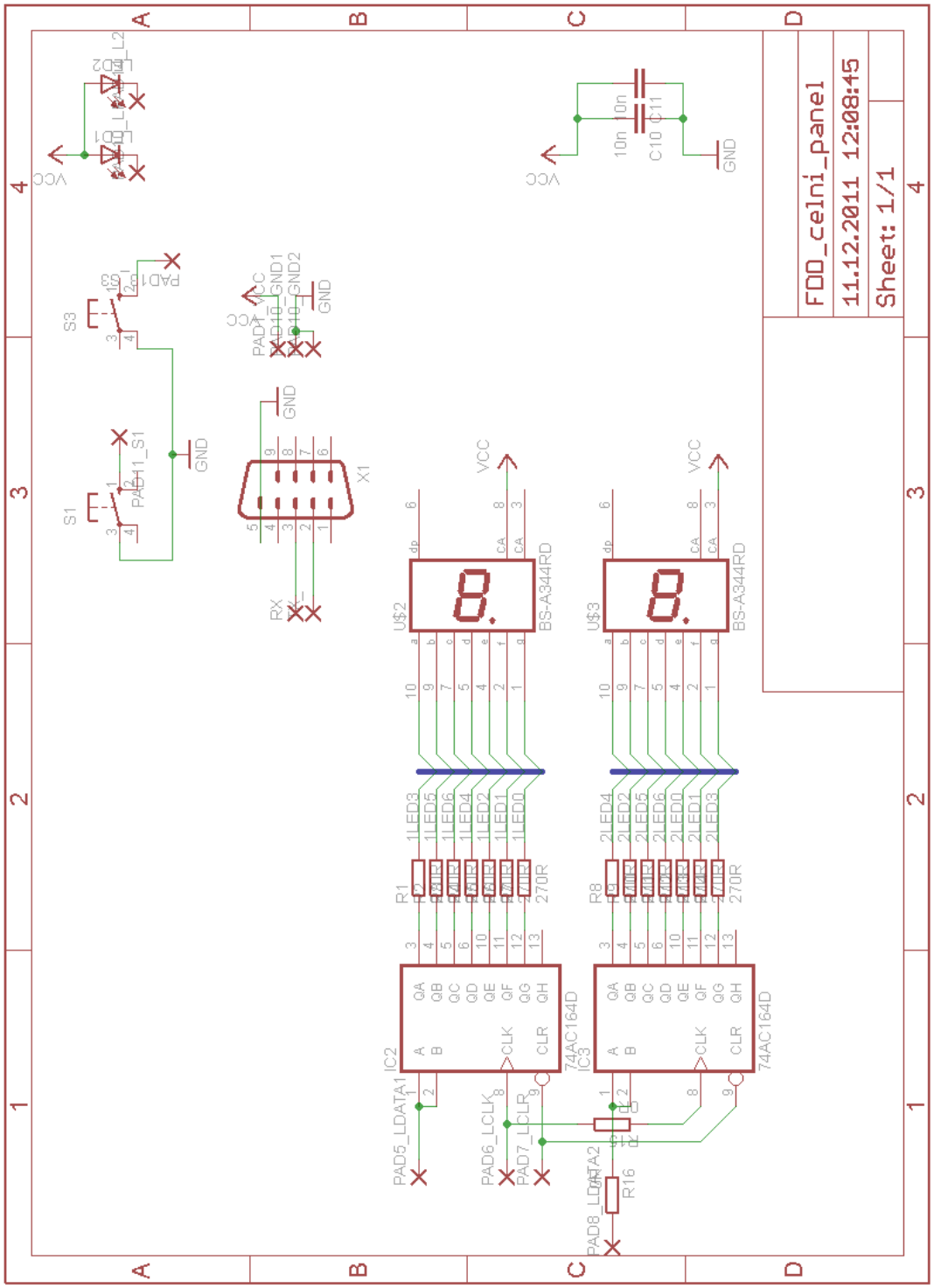
Obr. 0.5: Deska plošných spojů čelního panelu v.2

D DVD-ROM

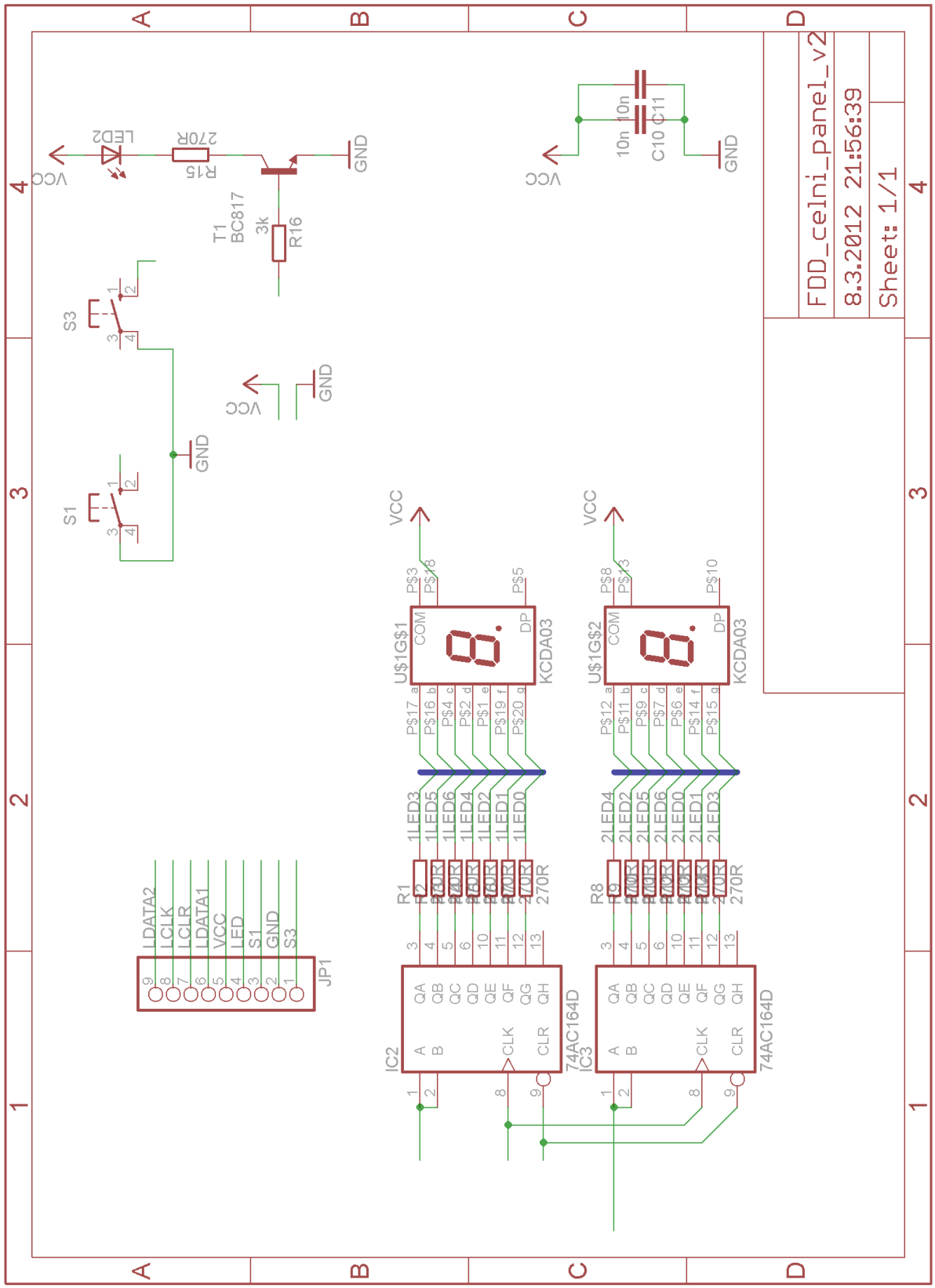
Obsahuje tuto kopii elektronického dokumentu práce, včetně zdrojových kódů pro emulátor a řídicí PC. Přehled verzí použitých vývojových rozhraní:

- Atmel® AVR Studio 5.1.208
- Atmel® AVR Dragon
- Microsoft® Visual Studio C# 2010
- Microsoft® .NET Framework 4.0.30319 RTMRel
- OS Microsoft® Windows 7 Professional

E Schéma zapojení čelního panelu

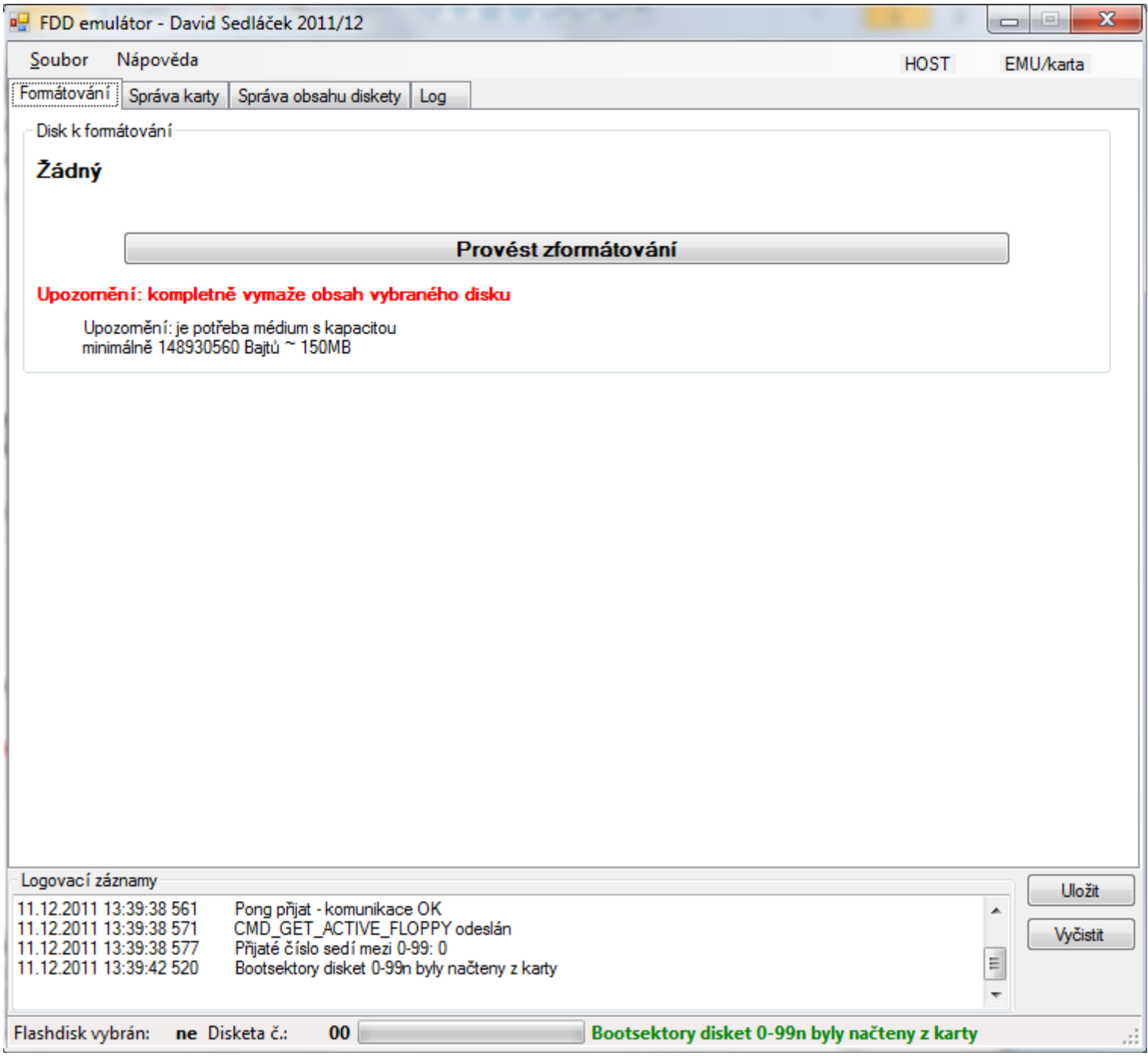


Obr. 0.6: Schéma zapojení čelního panelu emulátoru v. 1

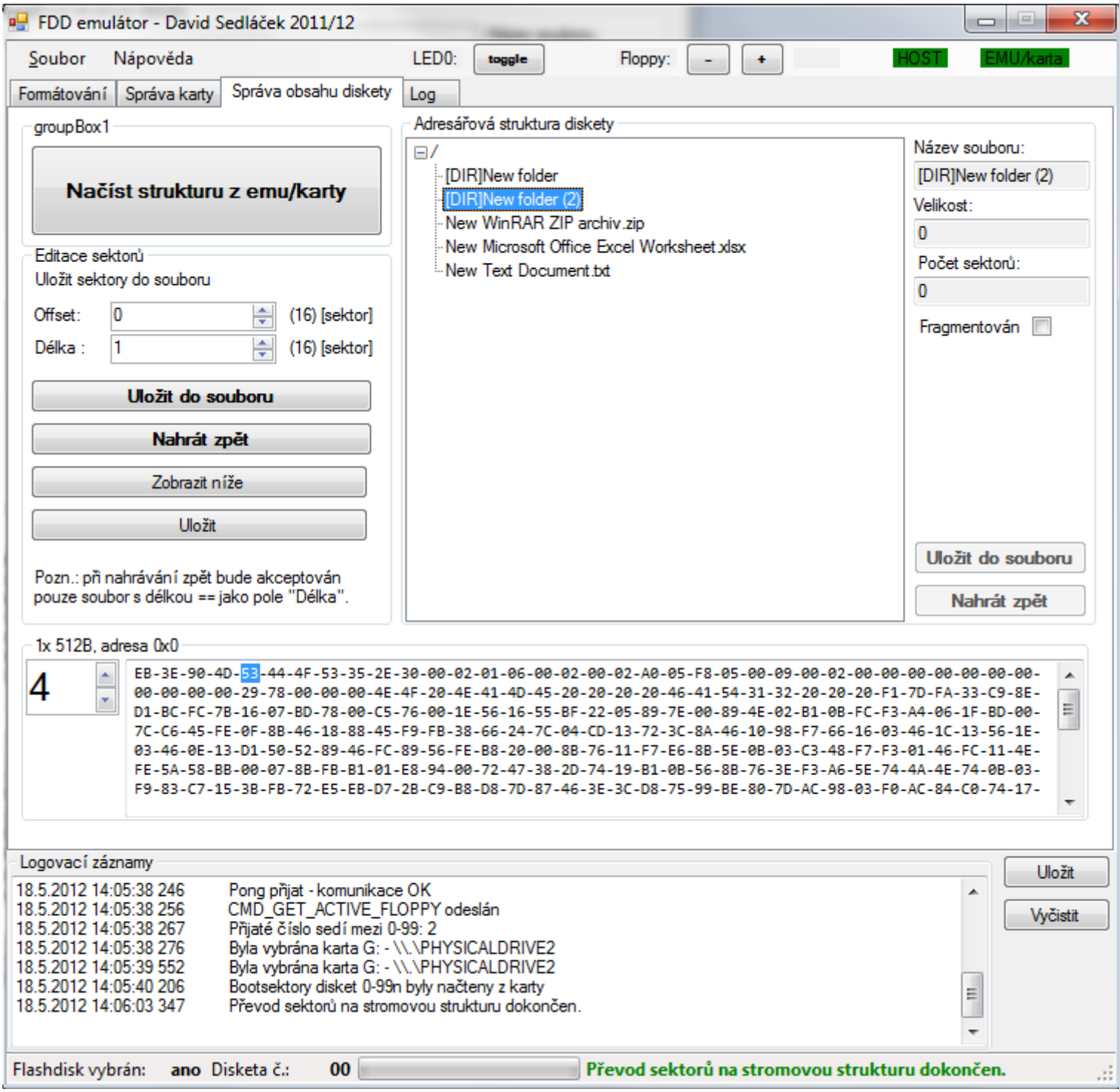


Obr. 0.7: Schéma zapojení čelního panelu v. 2

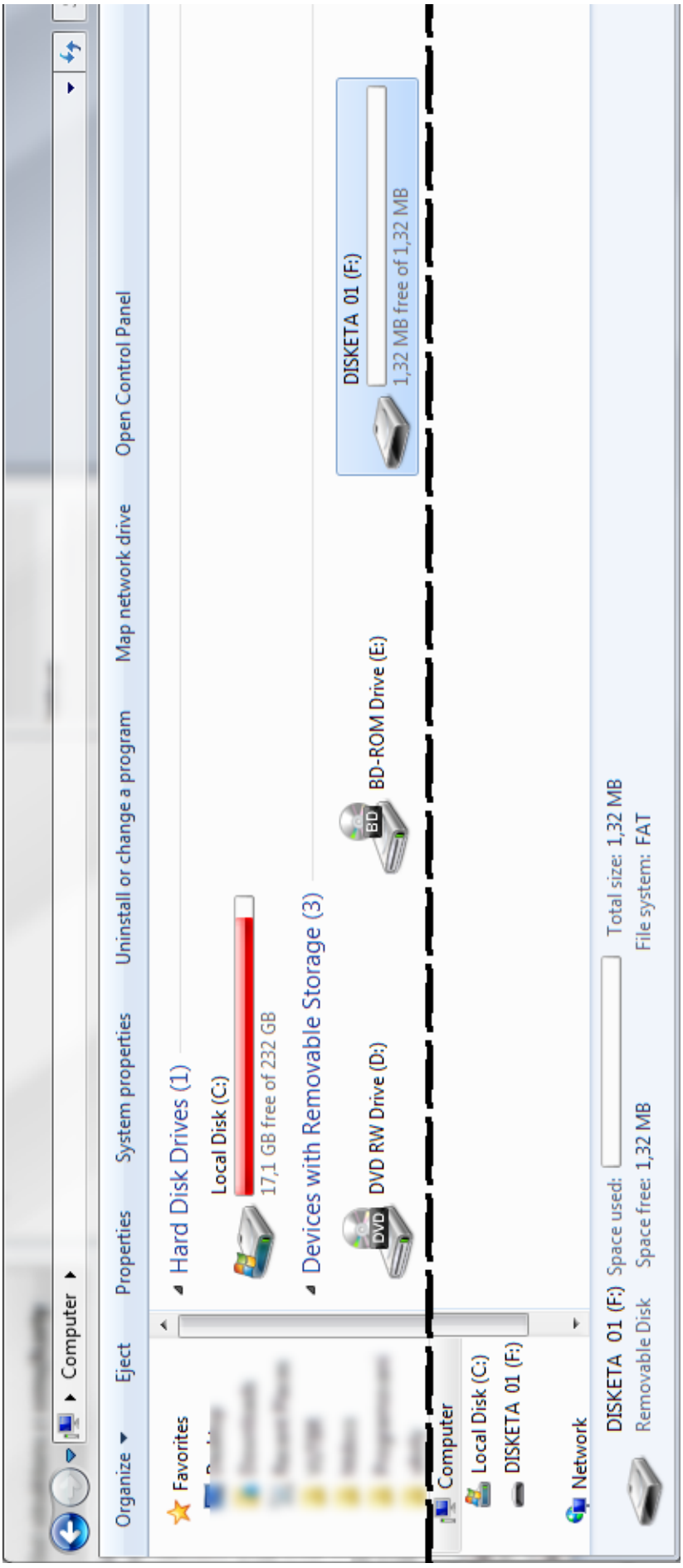
F Screenshoty řídicí aplikace



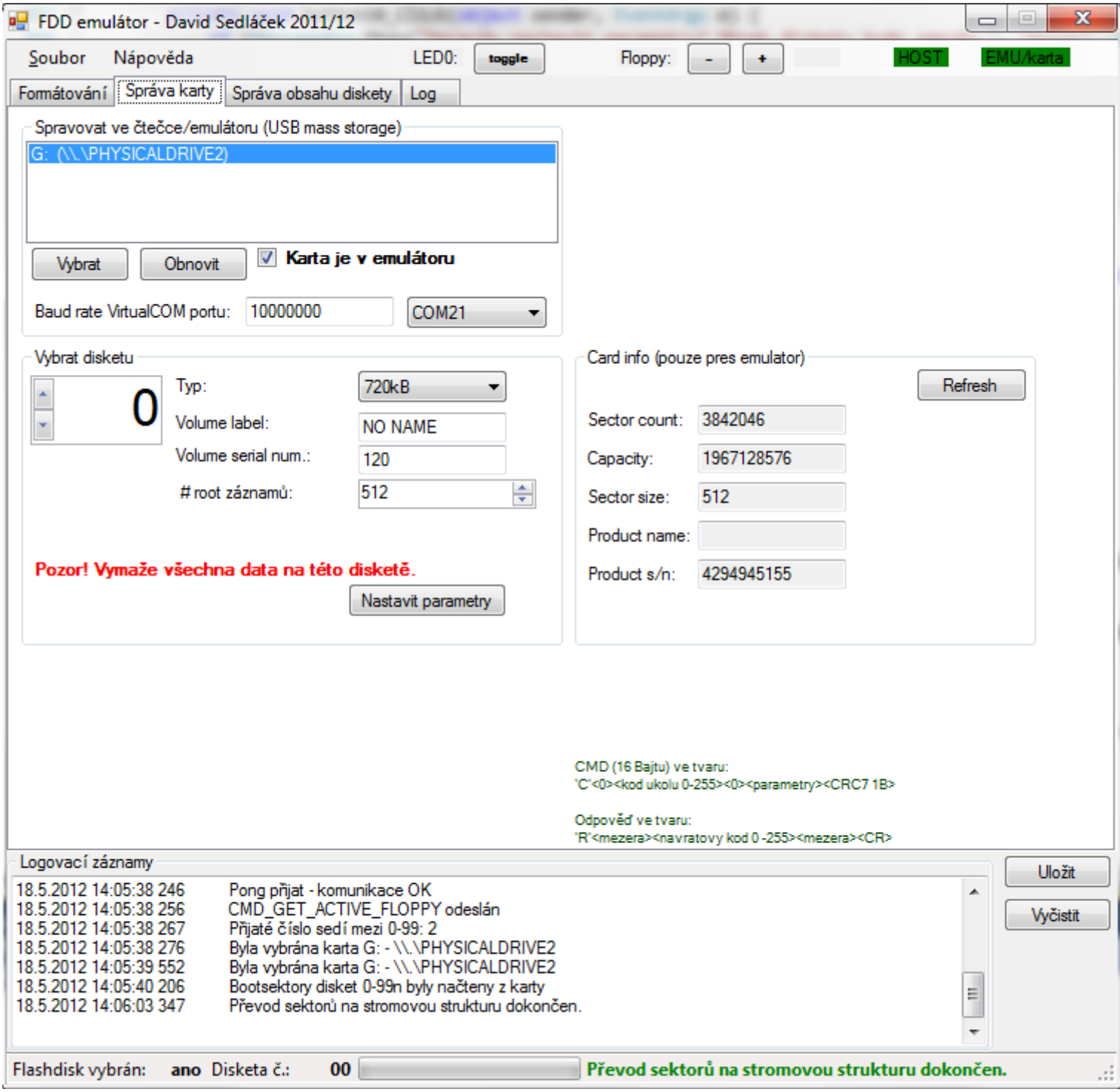
Obr. 0.8: Obrazovka formátování karty



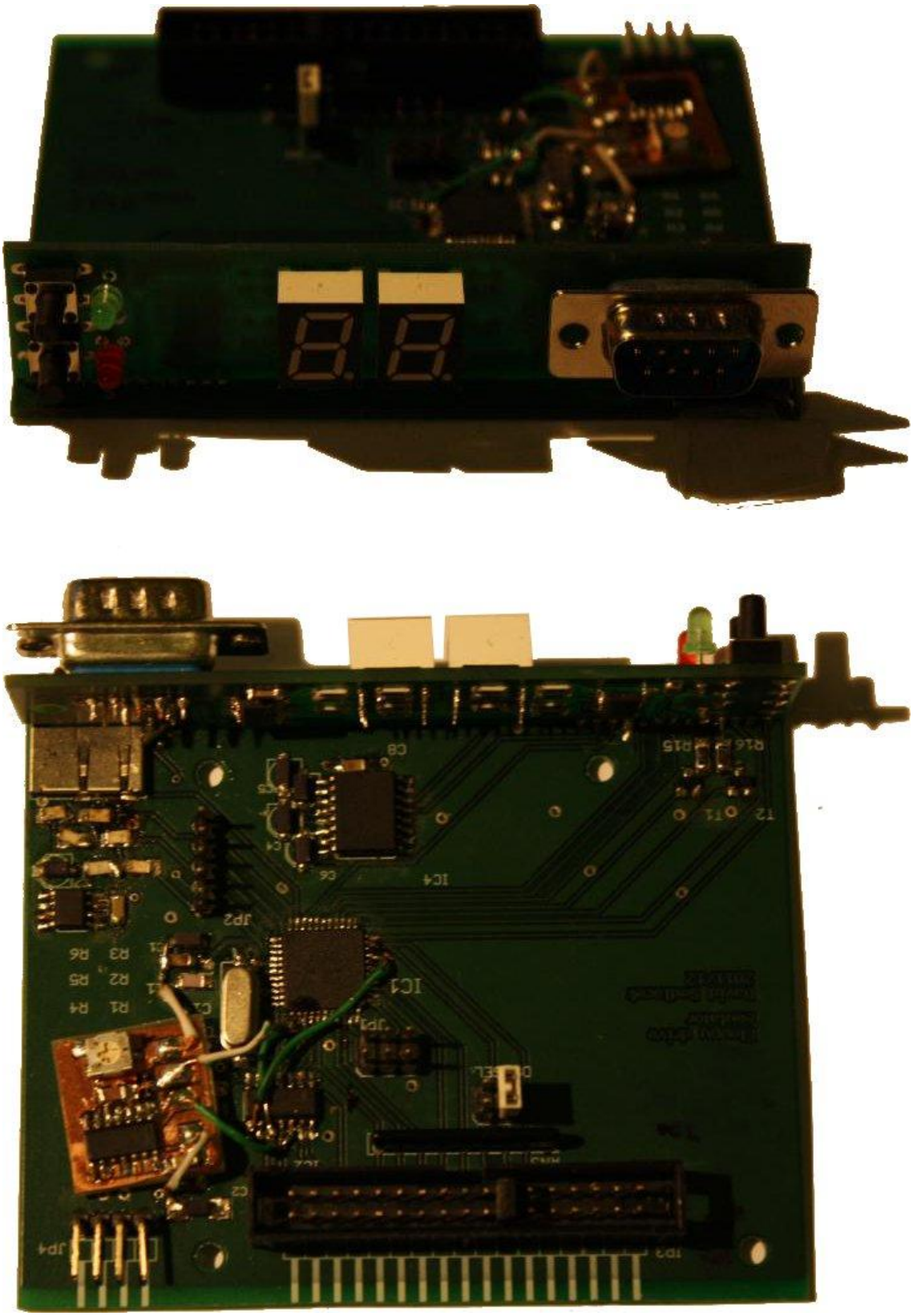
Obr. 0.9: Okno aplikace správa obsahu karty



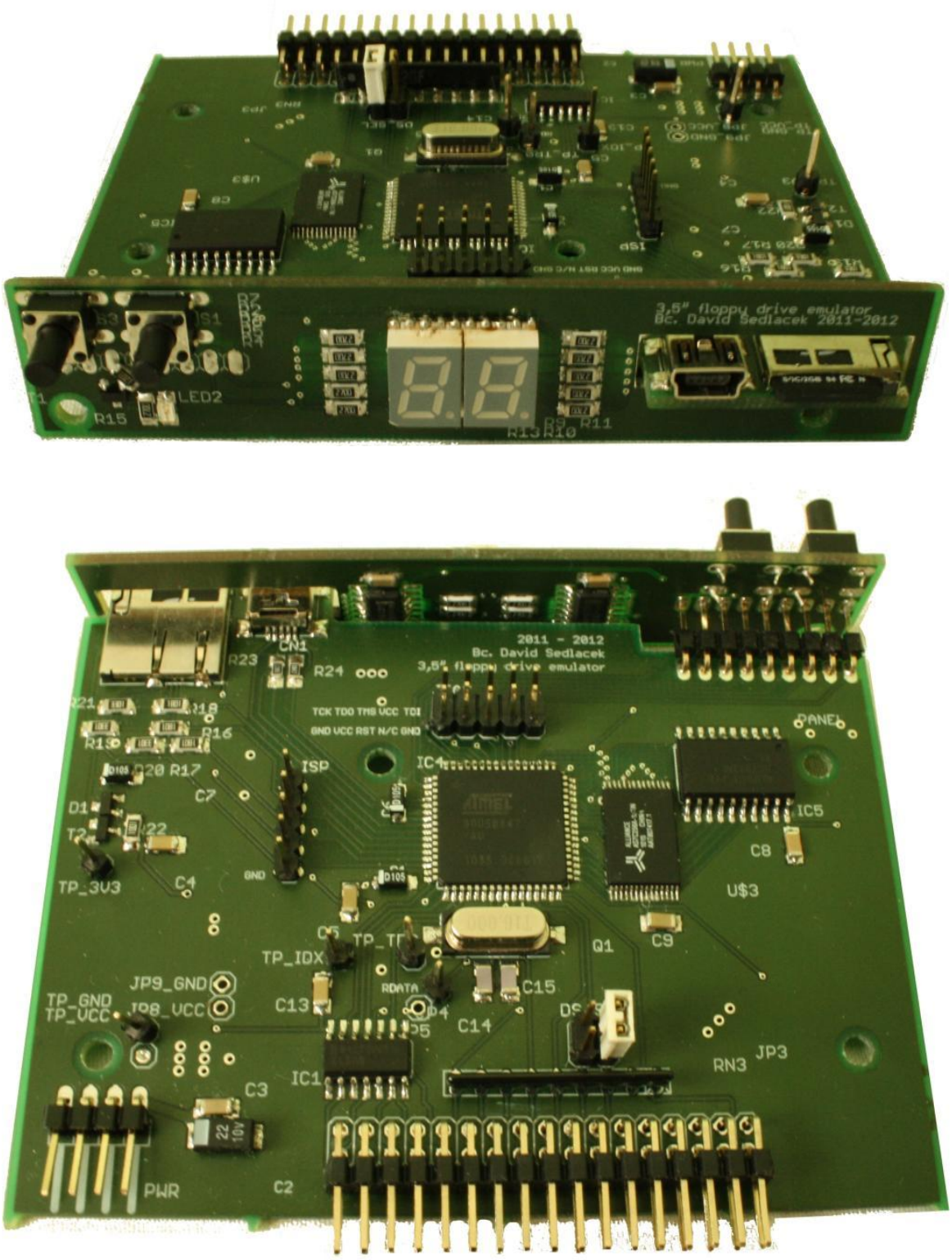
Obr. 0.10: Obrazovka "Computer" Windows 7 a připojená "disketa" – namapovaný fyz. svazek



Obr. 0.11: Okno aplikace správa karty



Obr. 0.12: FDD emulátor verze 1



Obr. 0.13: FDD emulátor verze 2