

Výukový modul pro předmět Programování mikropočítačů: MP3 přehrávač

Jakub Husár

Bakalářská práce
2015

 Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2014/2015

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Jakub Husár**
Osobní číslo: **A11109**
Studijní program: **B3902 Inženýrská informatika**
Studijní obor: **Informační a řídicí technologie**
Forma studia: **prezenční**

Téma práce: **Výukový modul pro předmět Programování mikro počítačů:
MP3 přehrávač**

Téma anglicky: **A Tuition Module for the Microcontroller Programming Course:
MP3 Players**

Zásady pro vypracování:

1. **Prostudujte hardwarové vlastnosti vývojového kitu M68EVB908GB60 a navrhnete způsob připojení modulu MP3 přehrávače.**
2. **Provedte hardwarový návrh desky rozhraní pro převod napěťových úrovní logických signálů mezi vývojovým kitem a modulem MP3 přehrávače.**
3. **Realizujte desku rozhraní a ověřte její funkci na vývojovém kitu M68EVB908GB60.**
4. **Sestavte a odlaďte podpůrné programové vybavení pro obsluhu MP3 modulu ve formě knihovny podprogramů v C jazyce.**
5. **Vytvořte ukázkovou aplikaci s využitím realizované knihovny podprogramů, která bude demonstrovat funkci modulu.**

Rozsah bakalářské práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. AXIOM MANUFACTURING. M68EVB908GB60 Development Board for Freescale MC9S08GB60, Rev. C [online]. 2006 [cit. 2015-02-02]. Dostupné z: <http://www.axman.com>.
2. FREESCALE SEMICONDUCTOR. CPU08 Central Processor Unit Reference Manual [online]. 2001 [cit. 2015-02-02]. Dostupné z: <http://www.freescale.com>.
3. FREESCALE SEMICONDUCTOR. HCS08 Family Reference Manual, Rev.1. [online]. 2003 [cit. 2015-02-02]. Dostupné z: <http://www.freescale.com>.
4. FREESCALE SEMICONDUCTOR. MC9S08GB/GT Data Sheet, Rev.2.3. [online]. 2004 [cit. 2015-02-02]. Dostupné z: <http://www.freescale.com>
5. JURÁNEK, Antonín a Miroslav HRABOVSKÝ. EAGLE pro začátečníky I: uživatelská a referenční příručka .: 2. vydání. Praha: BEN - technická literatura, 2007, 191 s. ISBN 80-730-0213-2.
6. VLSI SOLUTION. VS1053b - Ogg Vorbis/MP3/AAC/WMA/FLAC/MIDI Audio Codec Circuit, version 1.2 [online]. 2012 [cit. 2015-02-02]. Dostupné z: <http://www.vlsi.fi>

Vedoucí bakalářské práce:

Ing. Petr Dostálek, Ph.D.

Ústav automatizace a řídicí techniky

Datum zadání bakalářské práce:

6. března 2015

Termín odevzdání bakalářské práce:

22. května 2015

Ve Zlíně dne 6. března 2015



doc. Mgr. Milan Adámek, Ph.D.
děkan



L.S.



prof. Ing. Vladimír Vašek, CSc.
ředitel ústavu

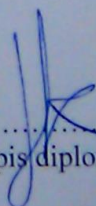
Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- Že odevzdaná verze diplomové/bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně 22.5.2015


.....
podpis diplomanta

ABSTRAKT

Cieľom tejto bakalárskej práce bolo vytvoriť prídavný vyukový modul „MP3 prehrávač“ k vývojovému kitu M68EVB908GB60 ako podporu cvičenia pre predmet Programovanie mikropočítačov.

Práca zahŕňa popis hardwarových vlastností vývojového kitu M68EVB908GB60 a MP3 Shield-u od spoločnosti SparkFun. Tieto informácie sú ďalej použité pri návrhu dosky pre prepojenie medzi kitom a modulom, čo zahŕňa aj prevod napäťových úrovní logických signálov. Pre vytvorenie serveru poskytujúceho MP3 súbory na sériovej linke vývojového kitu je použitý programovací jazyk JAVA s grafickým prostredím knižnice SWT. Knižnica s funkciami pre použitie modulu je vytvorená v programovacom jazyku C.

Kľúčové slová: Mikropočítač, Freescale, HCS08, MP3 Shield, MP3 prehrávač

ABSTRACT

The goal of this bachelor paper was to create additional educational module „MP3 player“ for developmental kit M68EVB908GB60 as a training support for the subject Programming of Microcomputers.

In this paper, the description of the hardware properties of the developmental kit M68EVB908GB60 and MP3 Shield by SparkFun was included. These pieces of information are used to design a board to interconnect kit and module that includes also transfer of voltage level of logical signals. To create a server that provides MP3 files on serial line of the developmental kit, the programming language JAVA with graphical surroundings of the library SWT was used. The library for the module is written in a programming language C.

Keywords: Microcomputer, Freescale, HCS08, MP3 Shield, MP3 Player

Rád by som poďakoval vedúcemu mojej práce, Ing. Petrovi Dostálkovi, Ph.D. za ochotu pri poskytovaní cenných rád a odbornej pomoci pri riešení problematiky tejto práce.

OBSAH

ÚVOD.....	9
I. TEORETICKÁ ČASŤ	10
1 JEDNOČIPOVÉ MIKROPOČÍTAČE	11
1.1 ZÁKLADNÁ CHARAKTERISTIKA	11
1.2 ROZDELENIE MIKROPOČÍTAČOV	11
1.2.1 PODEA ARCHITEKTÚRY	11
1.2.2 PODEA INŠTRUKČNEJ SADY	13
1.3 MIKROPOČÍTAČ MC9S08GB60	13
1.3.1 ZÁKLADNÉ VLASTNOSTI.....	13
1.3.2 REGISTRE	14
1.3.3 KOMUNIKAČNÉ ROZHRANIA	16
1.4 VÝVOJOVÝ KIT FREESCALE M68EVB908GB60	16
1.4.1 ŠPECIFIKÁCIE KITU	16
1.5 SÉRIOVÉ PERIFÉRNE ROZHRANIE SPI	18
1.5.1 PRINCÍP KOMUNIKÁCIE PRI ROZHRANÍ SPI	18
1.5.2 SPI NA MIKROPOČÍTAČI MC9S08GB60.....	19
2 SPARKFUN MP3 PLAYER SHIELD	20
2.1 VS1053B	21
2.1.1 KOMUNIKÁCIA POMOCOU ROZHRANIA SPI.....	22
3 ZVUKOVÝ FORMÁT MP3.....	23
3.1 HISTÓRIA	23
3.2 PRINCÍP	23
3.3 DEKÓDOVANIE	24
3.3.1 HARDWAROVÉ DEKÓDOVANIE	24
3.3.2 SOFTWAREOVÉ DEKÓDOVANIE.....	25
4 CODEWARRIOR	26
5 EAGLE	27
5.1 EDITOR SCHÉM.....	27
5.2 EDITOR SPOJOV	28
5.3 AUTOROUTER	28
II. PRAKTICKÁ ČASŤ	29
6 HARDWAROVÁ ČASŤ	30
6.1 BLOKOVÁ SCHÉMA PREPOJENIA	30
6.2 NÁVRH SCHÉMY A DOSKY PLOŠNÝCH SPOJOV.....	31
6.2.1 PREPOJENIE POTREBNÝCH PINOV	31
6.2.2 ZDROJ 5V	32
6.2.3 PREVOD NAPĚŤOVÝCH ÚROVNÍ.....	32
6.3 NASTAVENIE KITU M68EVB908GB60.....	34
7 SOFTWAREOVÁ ČASŤ	35

7.1	PODPORNÝ SERVEROVÝ PROGRAM	35
7.1.1	POPIS	35
7.1.2	PRÍKAZY	36
7.1.3	GRAFICKÉ PROSTREDIE	36
7.2	KNIŽNICE FUNKCIÍ	39
7.2.1	KNIŽNICA FUNKCIÍ PRE OVLÁDANIE SERVERA	39
7.2.2	KNIŽNICA FUNKCIÍ PRE OVLÁDANIE MP3 SHIELDU	41
7.3	UKÁŽKY POUŽITIA FUNKCIÍ V PROGRAME.....	48
	ZÁVER	50
	ZOZNAM POUŽITEJ LITERATURY	51
	ZOZNAM POUŽITÝCH SYMBOLOV A SKRATIEK.....	53
	ZOZNAM OBRÁZKOV	54
	ZOZNAM TABULIEK	56
	ZOZNAM PRÍLOH.....	57

ÚVOD

Existujú rôzne vývojové kity, ktoré sa používajú pri výuke programovania na školách zameraných na aplikovanú informatiku. Pri používaní vývojových kitov sa študenti učia základné praktické znalosti, ktoré môžu neskôr použiť v praxi. Na vývojových kitoch môžeme testovať a vyvíjať jednoduché, ale aj zložitejšie aplikácie pre amatérske využitie ba i pre návrh ovládania produktov väčších firiem.

Vývojové kity sú vždy osadené procesorom, ku ktorému sú pripojené periférne zariadenia, ako sú napr. displeje, tlačidlá, atď. Fakulta aplikovanej informatiky na Univerzite Tomáša Baťu v Zlíne pri výuke na predmete „Programovanie mikropočítačov“ používajú vývojové kity M68EVB908GB60 od spoločnosti Freescale, ktoré sú osadené procesorom MC9S08GB60 z rady HCS08. Tieto kity obsahujú periférne zariadenia ako kryštálový oscilátor, generátor hodinového signálu, sériové porty na prenos dát, LED diódy, pozičné prepínače, 2x16 znakový LCD displej, potenciometer a ďalšie. Študenti sa pomocou programovania procesora pokúšajú ovládať jednotlivé periférne zariadenia a tým získavajú skúsenosti. Tieto zariadenia sú pri výuke veľmi prospešné, ale stále nie sú úplne využiteľné v praxi, preto sa vyučujúci rozhodli priradiť ku kitom aj externé moduly, ktoré majú študenti za úlohu ovládať, resp. vyvinúť obslužný program daného modulu s použitím periférnych zariadení na kite a nakoniec odprezentovať svoj výtvar pred vyučujúcim. Študent je následne hodnotený na základe rozsahu práce a použitých periférnych zariadení.

Fakulta má momentálne k dispozícii moduly vytvorené študentami ako napr. Modul grafického displeja, Model digitálne riadeného zvukového procesoru, Modul krokového motora, atď. Jedným z modulov, ktoré sú k dispozícii je aj Modul tepelnej sústavy, ktorý som si mohol sám vyskúšať. Práca s modulom ma veľmi bavila a preto som sa pri výbere Bakalárskej práce dlho nerozhodoval. Ing. Petr Dostálek navrhol možnosť preštudovať MP3 Player Shield od spoločnosti SparkFun určený pre vývojové kity Arduino a prepojiť ho s vývojovým kitom M68EVB908GB60, čím sa vytvorí ďalší prídavný modul.

Prvá časť bakalárskej práce rozoberá teoretické informácie o použitom vývojovom kite, MP3 Shield-e a možnostiach komunikácie oboch zariadení.

Druhá, praktická časť práce sa zaoberá už samotným prepojením zariadení, riešením problémov s tým spojených a následným vývojom knižnice potrebnej k ovládaniu modulu, ktorý je praktickým výsledkom tejto práce.

I. TEORETICKÁ ČASŤ

1 JEDNOČIPOVÉ MIKROPOČÍTAČE

Jednočipový mikropočítač alebo microcontroller – MCU je prostriedok používaný pri rôznych riešeniach problémov spojených so spracovaním signálu a riadením.

1.1 Základná charakteristika

Microcontroller je integrovaný obvod, ktorý býva klasicky používaný v embedded (vstavaných) systémoch. Má vždy určenú špecifickú úlohu. Vývojom sa dosahuje jeho zmenšovanie a zároveň zrýchľovanie.

Pri navrhovaní týchto integrovaných obvodov sa kladie dôraz na spotrebu a nízku cenu. Mikropočítače dnes nájdeme v rôznych zariadeniach, napr. autá, kde sa používa mikropočítač na každú činnosť, od otvárania a zatvárania okien, až po riadenie vstrekovania paliva.

Najväčšou výhodou je to, že mikropočítač stačí len naprogramovať na určitú činnosť, pripojiť k napájaniu a funguje. [2] [3]

1.2 Rozdelenie mikropočítačov

Mikropočítače môžeme rozdeliť podľa dvoch kritérií :

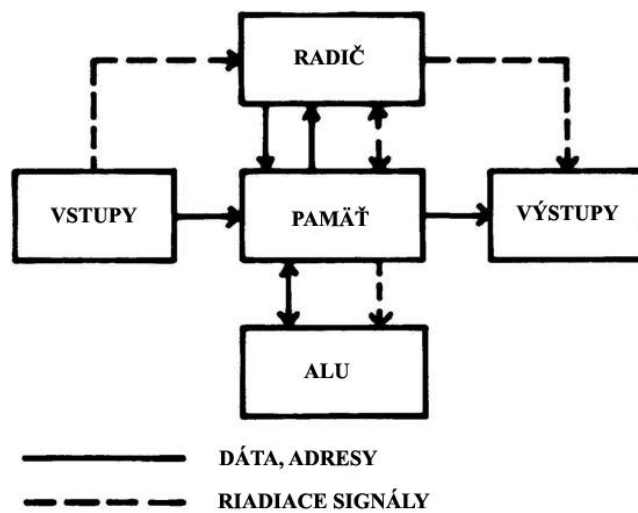
- podľa architektúry
- podľa inštrukčnej sady

1.2.1 Podľa architektúry

Architektúra mikropočítača je jeden z najdôležitejších aspektov, ktorý pôsobí na výkon mikropočítača. Najčastejšie sa používajú dve architektúry a to Von Neumannova architektúra a Harvardská architektúra, často ale narazíme aj na ich kombináciu.

Von Neumannova architektúra

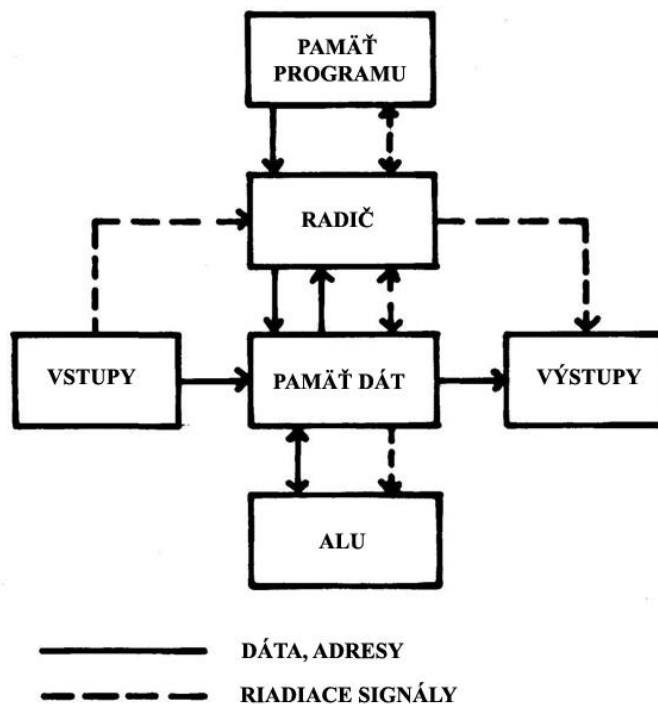
Pri tejto architektúre je charakteristickým znakom to, že dátová a programová pamäť sú spojené a teda používajú spoločnú dátovú a adresnú zbernicu, tzn. sekvenčné spracovávanie. (Obr. 1). Ušetrením vývodov spojením dátovej a programovej pamäti sa ale znížila rýchlosť spracovávania, keďže pamäte zdieľajú zbernicu, teda nemôžeme naraz čítať inštrukcie a dáta. [7]



Obr. 1. Von Neumannova architektúra

Harvardská architektúra

Táto architektúra je staršia ako Von Neumannova, na rozdiel od nej ma rozdelenú pamäť na programovú a dátovú (viď. Obr. 2), z čoho vyplýva, že inštrukcie a dáta sa spracúvajú naraz, ide teda o paralelné spracovávanie, ktoré je rýchlejšie ako sekvenčné spracovávanie Von Neumannovej architektúry. [7]



Obr. 2. Harvardská architektúra

1.2.2 Podľa inštrukčnej sady

CISC

Je skratka pre Complex instruction set computer, v preklade počítač s rozsiahlou inštrukčnou sadou. Ako teda z názvu vyplýva ide o rozsiahlu sadu inštrukcií, ktorá obsahuje viacero zložitých inštrukcií, ktoré síce šetria pamäťové miesto pre program, keďže obsahujú, napr. načítanie aj spracovanie dáta, ale sú náročné na dobu spracovania.

RISC

Je skratkou pre Reduced instruction set computer, v preklade počítač s obmedzenou inštrukčnou sadou. Z názvu vyplýva, že ide o sadu inštrukcií, ktorá je do istej miery obmedzená, resp. väčšina inštrukcií má rovnakú dĺžku a vykonáva sa v jednom inštrukčnom cykle. Táto sada síce znížila dobu spracovania, čím zvýšila rýchlosť, ale zároveň zväčšila veľkosť programu tým, že na zložitejšiu operáciu, kde pri CISC stačilo použiť jednu inštrukciu, v RISC treba použiť 2-3 inštrukcie, čo je pamäťovo náročnejšie.

V dnešnej dobe výrobcovia skôr kombinujú tieto dve sady, keďže sa snažia zachovať spätnú kompatibilitu s procesormi CISC, ale zložitejšie inštrukcie sú rozkladané na mikro-operácie a tie sa potom interne vykonávajú podobne ako u RISC procesorov.

1.3 Mikropočítač MC9S08GB60

MC9S08GB60 je výrobok firmy Freescale, ide o 8-bitový mikropočítač vychádzajúci z procesoru Motorola 6800, patrí do rady HCS08. Použitá je Von Neumannova architektúra, keďže dáta a program majú spoločnú pamäť. Vyrába sa v rôznych tvaroch a veľkostiach puzdra a s rôznymi typmi pamätí. [4]

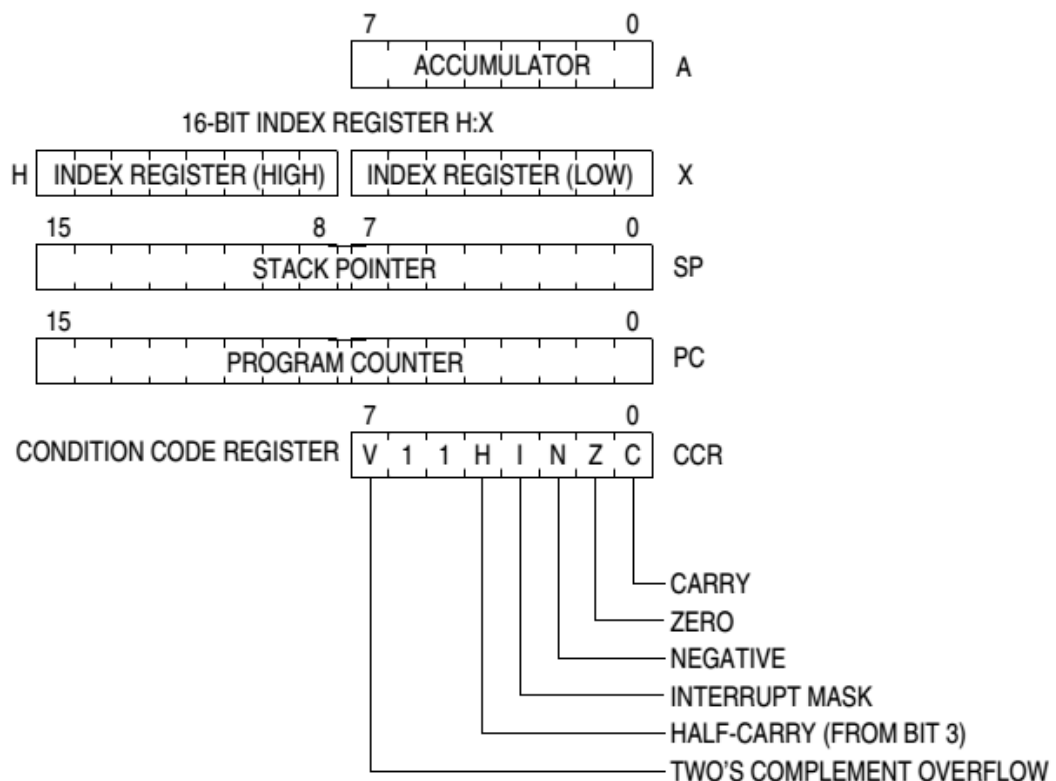
1.3.1 Základné vlastnosti

- 60 KB FLASH pamäte
- 4 KB RAM pamäte
- 8-bitová centrálna procesná jednotka HCS08

- 8-kanálový, 10-bitový A/D prevodník
- 56 obojsmerných (bidirectional) liniek – porty A až G (7 portov po 8 liniek)
- 8-bitový klávesnicový prerušovací systém (KBI)
- 2x asynchrónne sériové komunikačné rozhranie (SCI)
- 1x synchrónne sériové komunikačné rozhranie (SPI)
- 1x I2C rozhranie
- nastaviteľný interný generátor hodinového kmitočtu
- 3-kanálový TPM1 časovač a 5-kanálový TPM2 časovač
- COP watchdog systém [3]

1.3.2 Registre

Procesor obsahuje 5 registrov (Obr. 3):



Obr. 3. Registre HCS08 [3]

Akumulátor (A)

Univerzálny 8-bitový register. Jeden operand sa načíta do aritmeticko-logickej jednotky (ALU) prepojenej s akumulátorom a výsledok operácie sa uloží do akumulátora. Z a do akumulátora môžeme dáta presúvať pomocou rôznych spôsobov adresovania. [3]

Indexový register (H:X)

16-bitový register je zložený s dvoch 8-bitových registrov H a X, ktoré pracujú ako jeden 16-bitový ukazateľ, pri ktorom H obsahuje vyšší byte a X nižší byte adresy. Po reštarte mikropočítača sa register H nuluje, kdeže väčšina inštrukcií pracuje len s registrom X. [3]

Ukazateľ zásobníka (SP)

Anglicky Stack Pointer je 16-bitový register ukazujúci na ďalšie voľné miesto v zásobníku, ktorý sa nachádza v adresovom priestore pamäte RAM, je typu LIFO. Používa sa na uloženie návratovej adresy pri volaní podprogramu, uloženie registrov pri prerušení a na uloženie lokálnych premenných. [3]

Programový čítač (PC)

Anglicky Program Counter je 16-bitový register, ktorý obsahuje adresu operandu alebo inštrukcie, ktorá má byť spracovaná. Po spracovaní sa register inkrementuje. Pri skoku alebo prerušení sa do čítača nahrá adresa skoku alebo návratu. Pri reštarte sa do čítača nahrá vektor na adrese \$FFFF a \$FFFE, v tomto vektore je uložená prvá inštrukcia, ktorá má byť po reštarte vykonaná. [3]

Stavový register (CCR)

8-bitový register, ktorý obsahuje okrem masky prerušenia (I) aj 5 príznakov poukazujúcich na výsledok vykonanej operácie. Bity 5 a 6 sú trvalo nastavené na 1.

V - Two's Complement Overflow Flag

Príznak pretečenia sa nastaví, keď nastane pretečenie dvojkového doplnku.

H - Half-Carry Flag

Príznak sa nastaví pri prenose akumulátora z 3 na 4 bit.

I - Interrupt Mask Bit

Pri nastavenom príznaku prerušenia sú zakázané všetky maskované prerušenia. Pri prerušení sa príznak prerušenia nastaví a to hneď po uložení registrov CPU na zásobník.

N - Negative Flag

Príznak je nastavený, keď je výsledok aritmetickej, logickej alebo dátovej operácie záporný.

Z - Zero Flag

Príznak je nastavený, keď je výsledok aritmetickej, logickej alebo dátovej operácie nulový.

C - Carry/Borrow Flag

Príznak je nastavený v prípade, keď dochádza k prenosu alebo výpožičke na MSB, tj. najvýznamnejší bit. Používané pri sčítaní alebo odčítaní čísel, ktoré nedokáže ALU spracovať v jednom kroku. [3]

1.3.3 Komunikačné rozhrania

HCS08 je vybavený dvomi vzájomne nezávislými asynchrónnymi sériovými komunikačnými rozhraniami SCI, pomocou ktorých procesor komunikuje napr. s PC vďaka rozhraniu RS232.

Ďalej je vybavený aj synchronným sériovým komunikačným rozhraním (SPI – Serial Peripheral Interface), ktoré dokáže fungovať v režime Master alebo Slave, ide o duplexnú komunikáciu s kompatibilnými zariadeniami.

Medzi viacerými integrovanými obvodmi umožňuje sériovú komunikáciu kanál I2C (Inter-Integrated Circuit Module). [3]

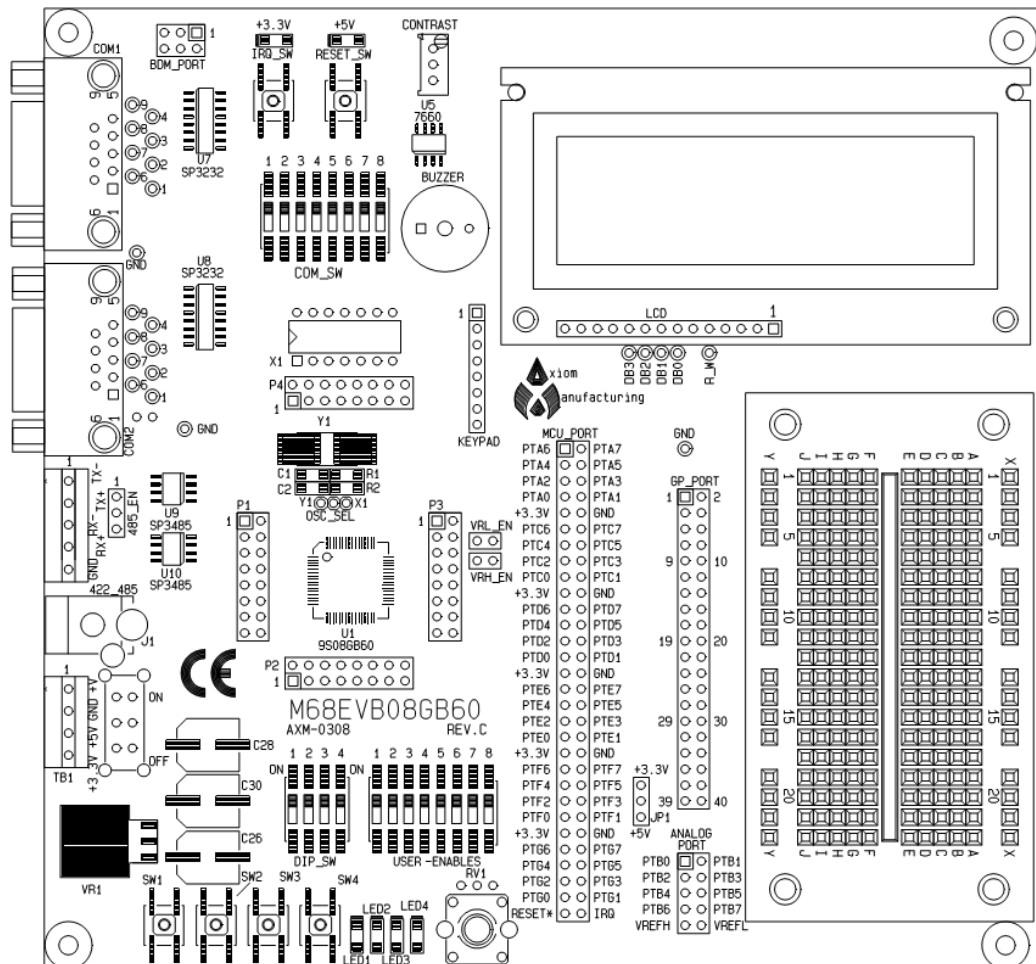
1.4 Vývojový kit Freescale M68EVB908GB60

M68EVB908GB60 je vývojový kit pre mikropočítač MC9S08GB60. Programovanie pomocou kitu je rýchlejšie a jednoduchšie na pochopenie.

1.4.1 Špecifikácie kitu

- mikropočítač M9S08GB60
- 32 kHz alebo 4 MHz kryštálový oscilátor
- nastaviteľný generátor hodinového signálu
- stabilizované napätie 3,3 V a 5 V
- COM1 sériový port, rozhranie RS232, konektor DB9-S (SCI0)
- COM2 sériový port, rozhranie RS232, konektor DB9-S (SCI1)

- signalizácia napájania, vypínač napájania
- užívateľské periférie :
 - o 4x LED dióda (PTF0-3)
 - o 4x DIP pozičný prepínač (PTB4-7)
 - o 4x spínač / tlačidlo (PTA4-7)
 - o 2x16 znakový LCD displej (PTG3-7, PTE6-7)
 - o bzučiak (PTD0)
 - o potenciometer (PTB0/AD0)
- MCU port poskytujúci všetky digitálne I/O
- analógový port
- kontaktné pole [1]



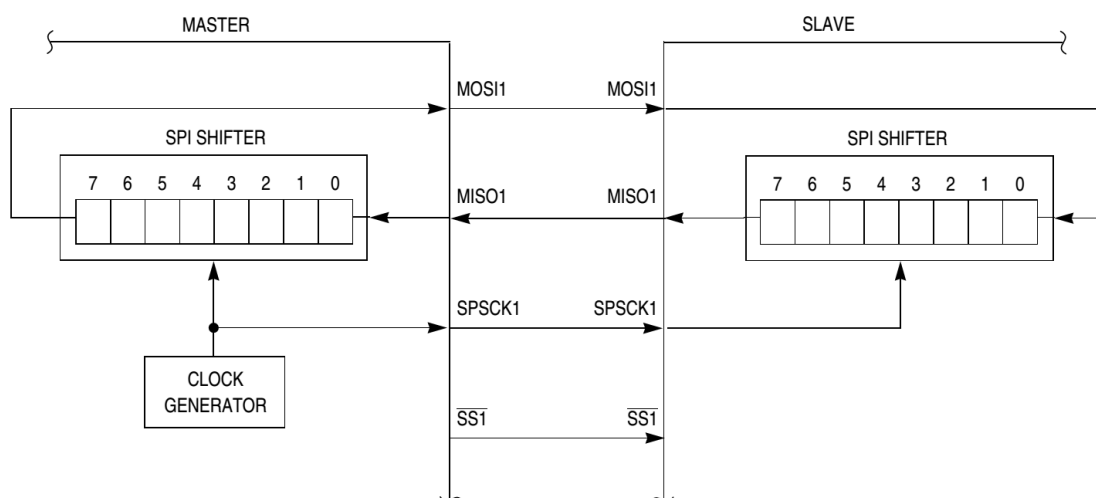
Obr. 4. Vývojový kit Freescale M68EVB908GB60 [1]

1.5 Sériové periférne rozhranie SPI

Sériové periférne rozhranie SPI (Serial Peripheral Interface) je používané na komunikáciu medzi riadiacimi mikroprocesormi a prídavnými perifériami, napr. A/D prevodník, displej. Toto rozhranie môže navzájom spájať dva a viac komunikačných bodov. Jeden z týchto bodov je nastavený v riadiacom režime Master a ostatné musia byť nastavené v načúvacom režime Slave. Master obsahuje generátor hodinového signálu, ktorý prenáša na ostatné Slave. Tento hodinový signál slúži na synchronizáciu, keďže ide o synchronný prenos dát. Okrem hodinového signálu sa po tomto rozhraní prenáša tiež dvojica dátových signálov MISO (Master In, Slave Out) a MOSI (Master Out, Slave In). Štvoricu SPI signálov uzatvára SSEL (Slave Select), ktorý slúži na výber niektorého zo Slave bodov. Tento signál sa používa len pri komunikácii viac ako 2 bodov, teda v prípade, keď je viac Slave bodov. [4]

1.5.1 Princíp komunikácie pri rozhraní SPI

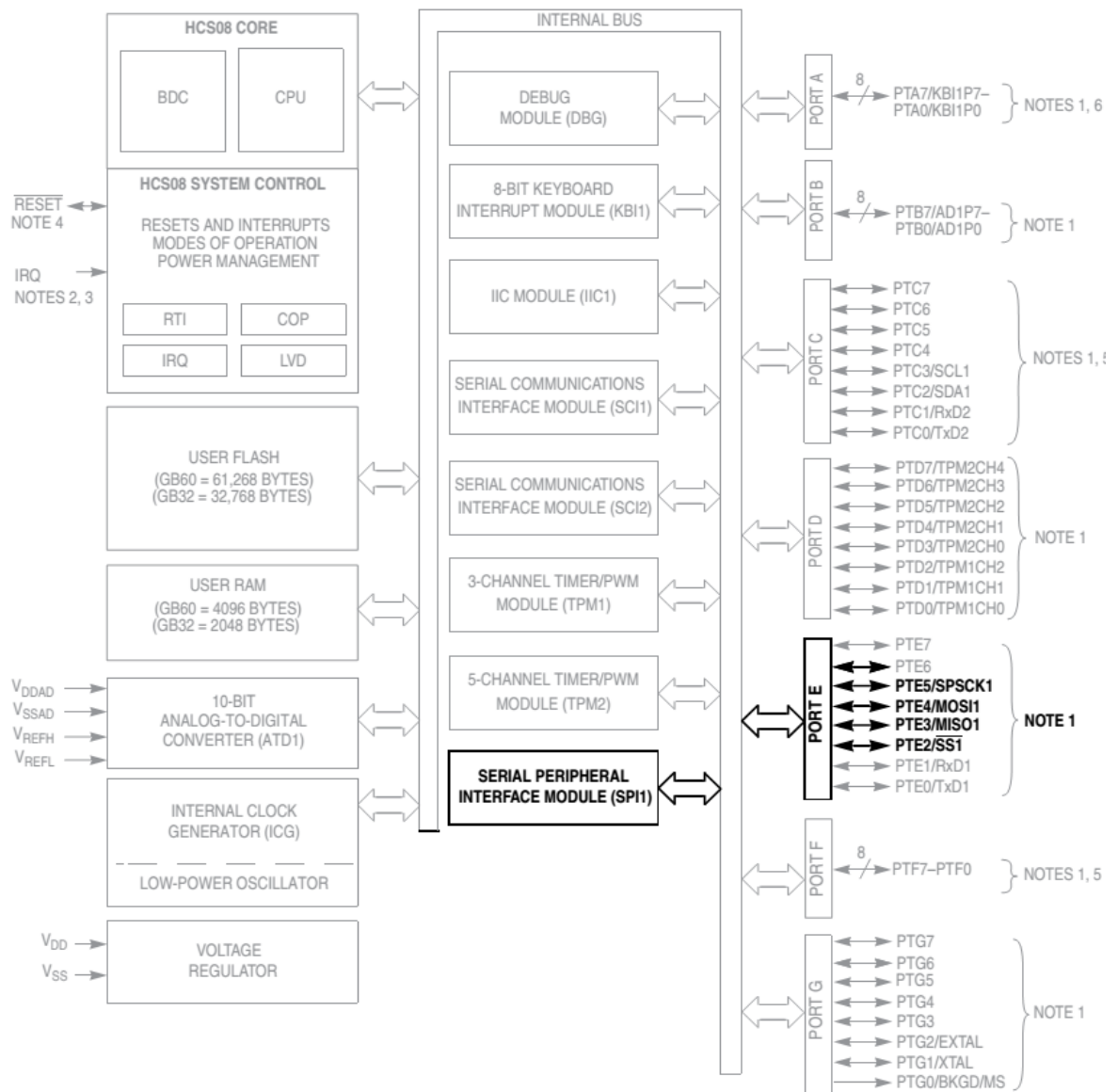
Obr. 5 zobrazuje prepojenie dvoch zariadení pri rozhraní SPI v zapojení Master – Slave. Master iniciuje všetky SPI dátové prenosy. Počas prenosu Master posúva jeden bajt dát pomocou pinu MOSI1 na Slave, ktorý súčasne posúva dáta pomocou pinu MISO1 na Master. Tento prenos je efektívny, pretože naraz vlastne prebehnú dva prenosy, z Mastera na Slave a zo Slave na Master. SPSC1 je pin pre hodinový signál, ktorý pri tejto komunikácii generuje Master. Pri použití viacerých Slave zariadení použijeme pre výber zariadenia pin SS1, ktorý reaguje na logickú 0. [4]



Obr. 5. Prepojenie dvoch zariadení pri rozhraní SPI [4]

1.5.2 SPI na mikroočítači MC9S08GB60

Mikroočítač MC9S08GB60 ponúka jeden SPI modul. Štyri piny spojené s SPI modulom sú zdieľané s portom E na pinoch 2-5 ako vidíme na obrázku 6. Tieto piny sú aktívne pre SPI komunikáciu len keď je povolené SPI, v opačnom prípade môžu byť tieto piny použité ako klasické I/O. [4]



Obr. 6. Bloková schéma mikroočítača MC9S08GB60 – SPI rozhranie [4]

2 SPARKFUN MP3 PLAYER SHIELD

Ide o rozširujúcu dosku určenú pre Open-Source platformu Arduino. Doska je osadená VS1053B MP3 audio dekodérom, ktorý je schopný dekódovať zvukové súbory Ogg Vorbis / MP3 / AAC / WMA / MIDI.

Táto posledná verzia Shield-u má pridaný slot pre SD kartu, kde môžu byť uložené zvukové súbory. Pomocou zbernice SPI môže byť súbor prenesený na VS1053B, kde je následne dekódovaný a odoslaný na 3,5 mm stereo jack umiestnený na doske. [10]



Obr. 7. MP3 Player Shield [10]

2.1 VS1053B

VS1053b je jednočipový audiodekodér. Obsahuje vysoko výkonný DSP procesor VS DSP 4, pracovní paměť, 16KiB instrukční RAM paměť a 0,5+ KiB datovou RAM paměť pro uživatelské aplikace běžící současně s akýmkoliv vstavaným dekodérem. Dále obsahuje sériové kontrolné a vstupné rozhraní, až 8 univerzálných I/O pinů, UART, vysoko kvalitní variabilně-vzorkovací stereo AD převodník (mikrofón, linka, linka + mikrofón nebo 2x linka) a stereo DA převodník s výstupem pro sluchadla.

VS1053b přijímá tok dat přes sériový vstup, který je v móde počívající, resp. SLAVE. Vstupný proud dat je dekodovaný a přes digitální regulaci hlasitosti převzorkovaný na 18-bit, multi-bit, sigma-delta DA převodník.

Dekódování je řízené pomocí sériové řídiacej zbernice. Okrem základného dekodovania je možné pridať špecifické črty, napr. DSP efekty na uživatelskej RAM pamäti. [6]



Obr. 8. Audiodekodér VS1053b

2.1.1 Komunikácia pomocou rozhrania SPI

V audiodekodéri VS1053b je komunikácia pomocou rozhrania SPI rozdelená do dvoch ďalších rozhraní :

- Serial Data Interface (SDI) – Dátové rozhranie
 - je určené na prenášanie komprimovaných dát rôzneho formátu na dekodér VS1053b. Ak je vstup dekodéra prázdny alebo plnený pomaly, analógový výstup je automaticky stlmený. [6]

- Serial Control Interface (SCI) – Riadiace rozhranie
 - je kompatibilné so špecifikáciou SPI rozhrania. Dátové prenosy sú tu vždy 16-bitové.
 - toto rozhranie slúži na manipuláciu (čítanie a písanie) s registrami audiodekodéra VS1053b.
 - hlavné prvky riadiaceho rozhrania :
 - riadenie prevádzkového režimu, hodín a efektov
 - prístup k informáciám a dátovým hlavičkám
 - príjem kódovaných dát v režime nahrávania
 - nahrávanie a riadenie používateľských programov [6]

3 ZVUKOVÝ FORMÁT MP3

Zvukový formát MP3 (MPEG-1 alebo MPEG-2, Audio Layer III) je patentovaný formát stratovej kompresie digitálneho zvuku. Kompresný algoritmus tohto zvukového formátu je definovaný skupinou MPEG (Motion Picture Experts Group). MP3 sú bežne používané pri ukladaní zvuku alebo pri streamovaní, momentálne je to vlastne zvukový štandard. [8] [9]

3.1 História

Začiatkom 80. rokov začal profesor Dieter Seitzer z Erlangen-Nuremberg University v Nemecku skúmať problém prenosu komprimovanej hudby po telefónnej linke. Zostavil skupinu vedcov zaoberajúcich sa výskumom kódovania zvuku. Behom vývoja študent profesora Seitzera, Karlheinz Brandenburg rozšíril základné princípy vnímania kódovaného zvuku ľudským uchom, popísal psychoakustiku.

V roku 1989 dokončil Karlheinz Brandenburg svoju dizertačnú prácu. Popísal v nej algoritmus OFC, ktorý je považovaný za predchodcu MP3. OFC algoritmus skenoval a odstraňoval zvuky pod a nad prahom počuteľnosti ľudského ucha.

Tím z Fraunhoferského inštitútu vylepšil tento algoritmus a zaviedol nový zvukový kodek ASPEC, ktorý neskôr spolu s MUSICAM vyhral po formálnych testoch súťaž pre chystaný zvukový štandard MPEG. Ustanovili sa dôležité vlastnosti ako rýchlosť vzorkovania, štruktúra rámcov, štruktúra hlavičiek, či počet vzoriek v rámci.

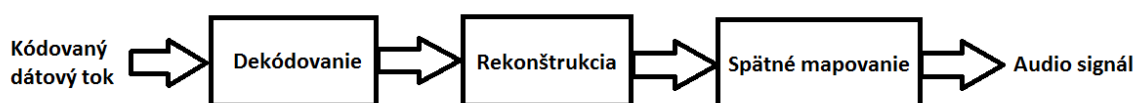
Samotný formát MP3 sa začal hromadne používať až v roku 1995, kedy bol uvoľnený prvý softwarový MP3 prehrávač Winplay3, ktorý pracoval v reálnom čase. Odvtedy bolo možné dekódovať a reprodukovat' formát MP3 na osobných počítačoch. [8]

3.2 Princíp

Formát MP3 využíva nedokonalosť ľudského ucha na základe psychoakustického modelu, čím sa snaží odstrániť redundanciu zvukového signálu, resp. frekvencie, ktoré človek nepočuje alebo nevníma sú zo stopy odstránené, a tým sa veľkosť výsledného signálu znižuje. Existujú viaceré modely ľudského ucha, takže existujú rôzne kodeky, ktoré dosahujú rôzne výsledky. Niektoré kodéry podporujú tzv. variabilný bitrate, kde tiché, resp. kľudné pasáže sú zaznamenané s nižším bitrate a dynamické pasáže naopak s vyšším bitrate. [9]

3.3 Dekódovanie

Tým, že dáta sú komprimované a kódované, nedá sa ich jednoducho poslať na D/A prevodník. Pri prehrávaní komprimovaného alebo kódovaného zvukového záznamu vo formáte MP3 je potrebné tento kódovaný tok dát pretaviť s malou odchýlkou, ktorú ľudské ucho nezachytí, na signál ako bol pred kódovaním. Dekodér je prostriedok, ktorý nám pri tejto dekomprimácii pomôže. [8]



Obr. 9. Bloková schéma dekodéru

Do dekodéru vstupujú kódované – komprimované dáta. V bloku Dekódovanie sa detegujú chyby a tok dát je rozdelený na jednotlivé rámce. V ďalšom bloku Rekonštrukcia sa obnoví amplitúdová verzia sady mapovaných vzoriek. Posledný blok Spätné mapovanie prevedie tieto rekonštruované vzorky na výsledný audio signál v digitálnej podobe, ktorý sa pošle na D/A prevodník a následne na zvukové zariadenie, napr. reproduktor. [8]

3.3.1 Hardwarové dekódovanie

Na dekódovanie mp3 dát sa používajú špeciálne embedded obvody, naprogramované digitálne signálne procesory určené priamo na dekódovanie zvukových formátov. Na ich ovládanie a posielanie dát sa používa I2C a SPI zbernica, ktoré sa dajú jednoducho pripojiť k jednočipovému počítaču a k pamäťovým médiám.

Príkladom takéhoto obvodu je VS1053B od fínskej firmy VLSI, ktorý má veľmi jednoduché zapojenie a tým je aj veľmi jednoduché jeho ovládanie a posielanie, napr. pomocou SPI rozhrania. Takéto zapojenia sa používajú v malých zvukových prehrávačoch.

3.3.2 Softwarové dekódovanie

Softwarové dekódovanie sa líši oproti hardwarovému, už ako podľa názvu vyplýva, v tom, že o dekódovanie sa stará software počítača, teda program, ktorý vďaka kodekom vie, akým spôsobom má daný súbor dekódovať a používa na to prostriedky procesora počítača, čo je síce časovo náročné, ale pre dnešné vysoko výkonné procesory to už nie je žiaden problém.

Príkladom softwarového dekodéru je WinPlay3, ktorý bol uvoľnený v roku 1995.



Obr. 10. Softwarový dekodér WinPlay3

Časom boli vyvinuté viaceré iné dekodéry, ktoré sa používajú dodnes, napr. WINAMP

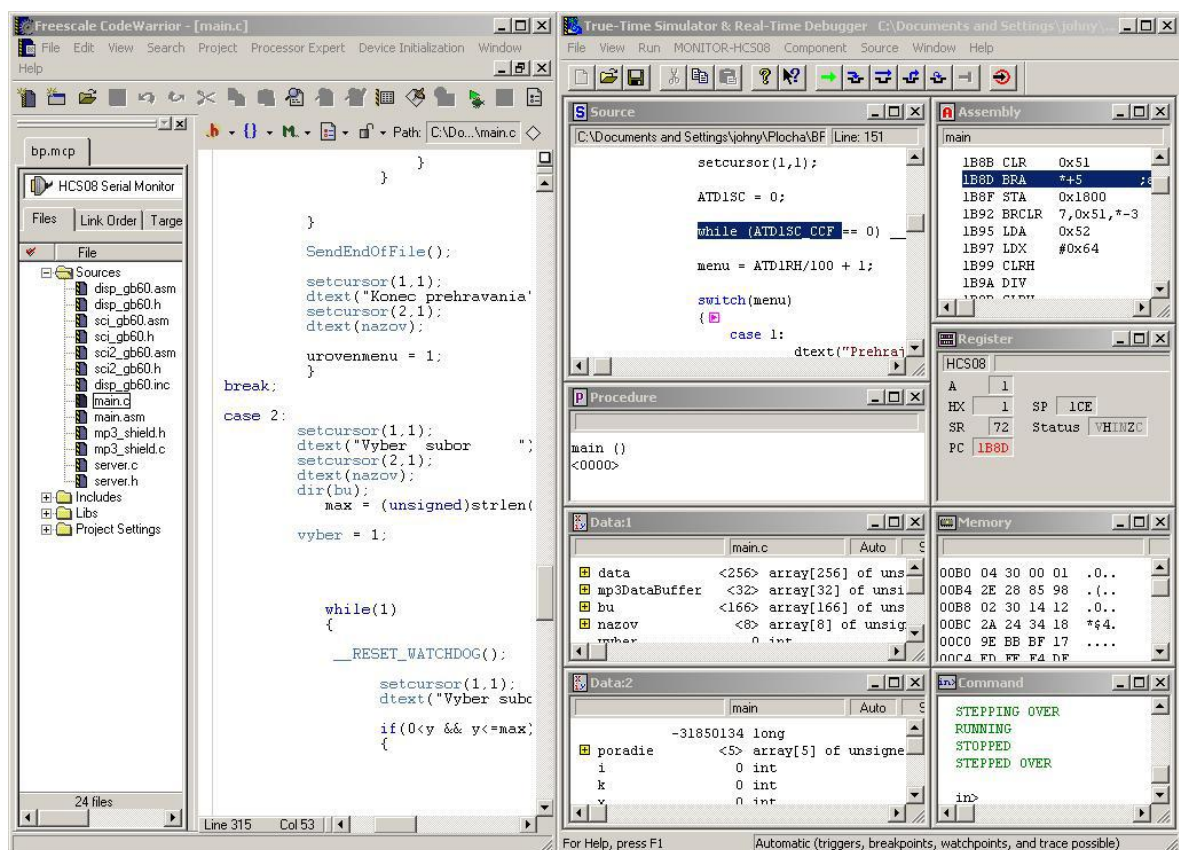


Obr. 11. Softwarový dekodér Winamp

4 CODEWARRIOR

CodeWarrior je komplexné vývojové prostredie od firmy Freescale, ktoré umožňuje rýchle vyvíjanie aj náročnejších aplikácií pre produkty firmy Freescale. Naša fakulta používa toto vývojové prostredie vo verzií 6.3. Pomocou tohto nástroja sa vo výuke programuje na procesoroch MC9S08GB60, ktoré sú zapojené v kite M68EVB908GB60, ktorý ma k dispozícií viacero periférnych zariadení.

V tomto vývojovom prostredí je možné programovať v jazyku symbolických adries, teda Assembler, rovnako ako aj v programovacom jazyku C a C++. Možné použiť aj kombináciu programovacích jazykov, napr. v prípade, keď je knižnica napísaná v jazyku symbolických adries, program sa dá jednoducho písať v jazyku C. Toto prostredie obsahuje okrem iného aj výkonný simulátor, resp. Debugger True-Time Simulator & Real-Time Debugger pre ladenie vyvíjaných aplikácií.



Obr. 12. Ukážka vývojového prostredia so spusteným Debuggerom

5 EAGLE

EAGLE je uživatelsky veľmi prívetivý a výkonný nástroj na návrh dosiek plošných spojov od firmy CadSoft. Názov nástroja je vlastne skratka spojenia Easily Applicable Graphical Layout Editor. [5]

Program má tri hlavné najčastejšie používané moduly :

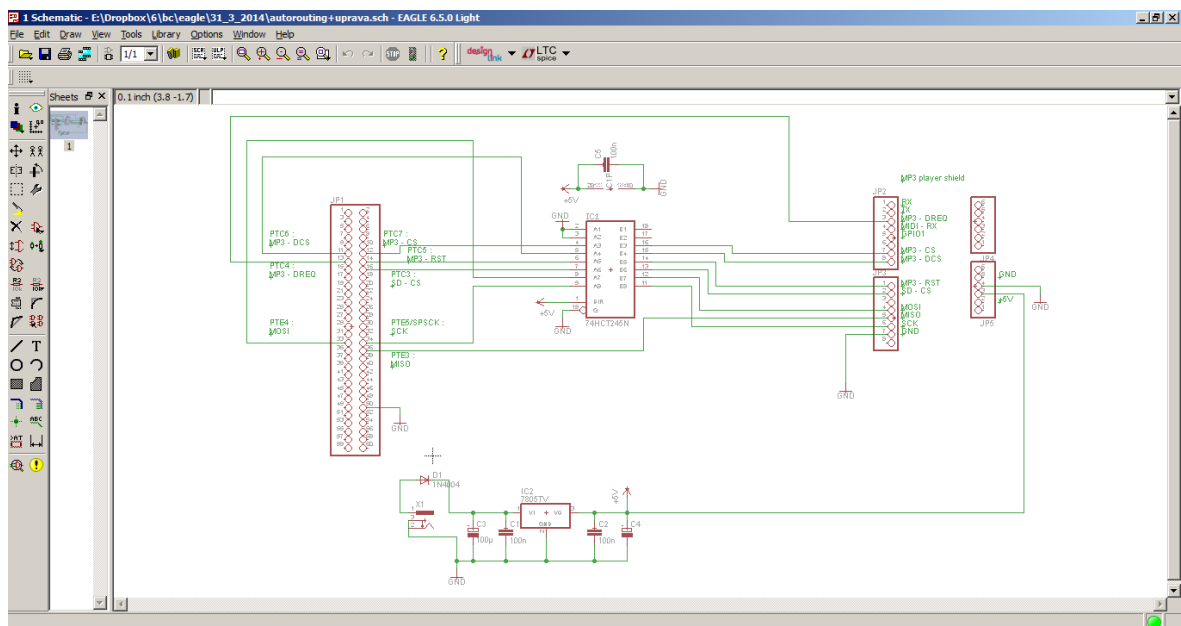
- Editor schém
- Editor spojov
- Autorouter

5.1 Editor schém

Editor schém zaručuje rýchle a efektívne navrhovanie schém. Je tu možné nastaviť viditeľnosť spojov, značiek, názov a hodnôt súčiastok.

Všetky dostupné súčiastky sú uložené v knižniciach, ktoré sú zoradené podľa typu alebo podľa výrobcu. Tieto knižnice dodávajú výrobcovia, prípadne súčiastku, ktorá v knižnici nie je, je možné navrhnuť a tým si vytvoriť vlastnú knižnicu.

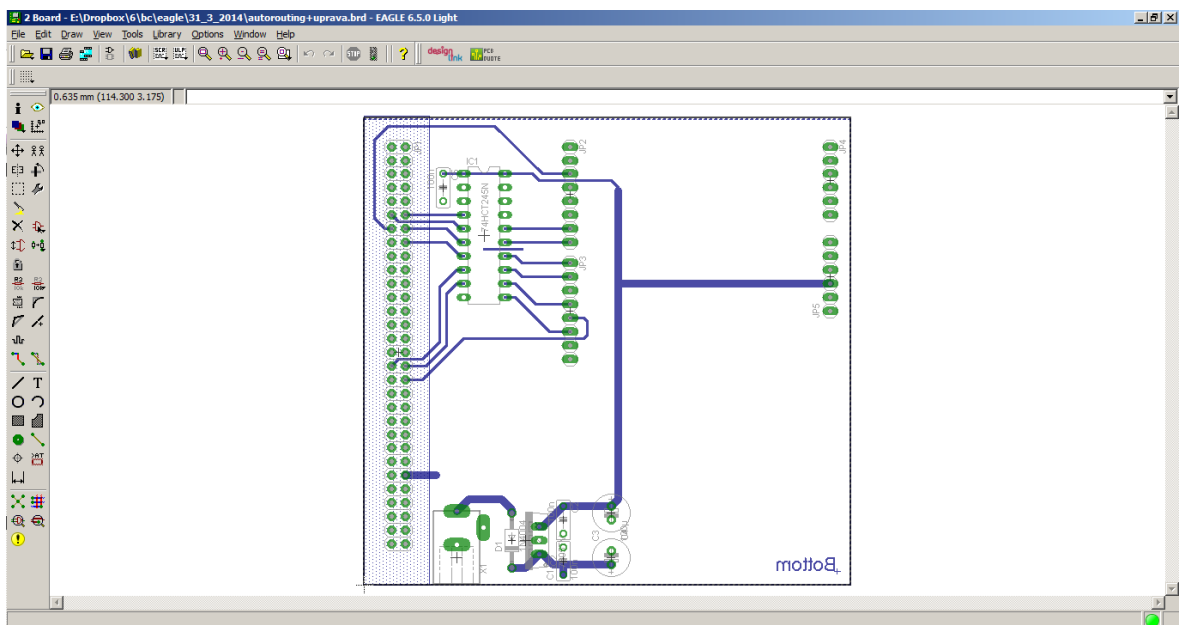
Po navrhnutí schémy je možné previesť elektrickú kontrolu ERC a odhaliť tak prípadné chyby v schéme. [5]



Obr. 13. EAGLE - Editor schém – ukážka

5.2 Editor spojov

Editor spojov sa len minimálne líši od editora schém. Po prepnutí z editora schém do editora spojov je potrebné poukladať súčiastky zo schémy na reálnu pozíciu na doske a následne určiť cestu plošným spojom. Cesty vidíme ako gumové spoje, ktoré sú prepojené podľa schémy a po zvolení hrúbky cesty môžeme určovať, kde má byť umiestnená daná cesta na doske plošných spojov. [5]



Obr. 14. EAGLE - Editor spojov - ukážka

5.3 Autorouter

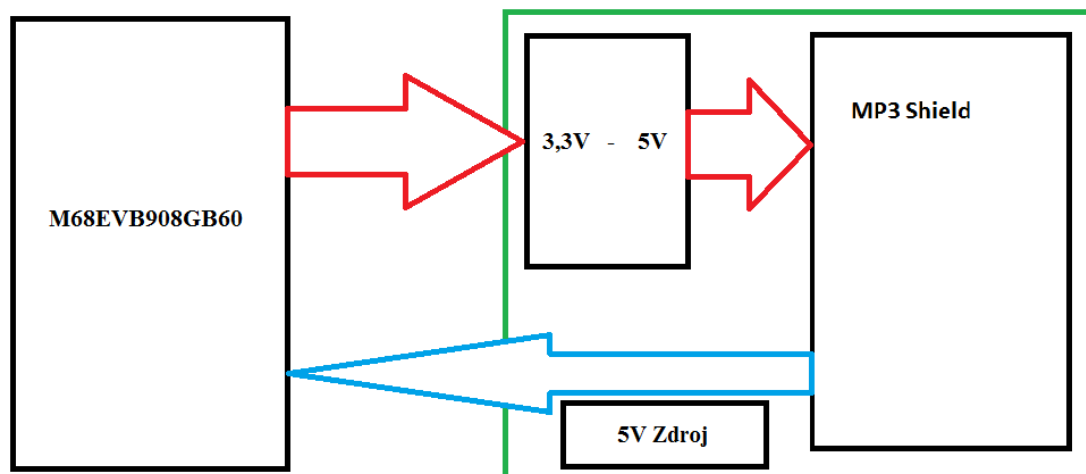
Autorouter je modul, ktorý bol vytvorený pre uľahčenie a predovšetkým urýchlenie práce užívateľa. Je to funkcia, ktorá po správnom nastavení slúži k automatickému generovaniu plošných spojov na doske. Autorouter pracuje metódou ripup&retry router, teda skúša rôzne spôsoby uloženia ciest. Najčastejšie sa používa kombinácia Autorouter s následnou ručnou úpravou, keďže niektoré cesty, napr. napájacie, potrebujú väčšiu hrúbku. [5]

II. PRAKTICKÁ ČASŤ

6 HARDWAROVÁ ČASŤ

Cieľom tejto práce je prepojiť vývojový kit, na ktorom sa pracuje pri výuke, s MP3 Shieldom tak, aby bolo možné tento Shield používať pri výuke ako rozširujúci model pomocou ktorého sa študenti budú učiť prenášať dáta cez sériovú linku mikropočítača HCS08.

6.1 Bloková schéma prepojenia



Obr. 15. Bloková schéma prepojenia

Ako je na obrázku vidno, prepojenie je medzi vývojovým kitom M68EVB908GB60 a MP3 Shieldom. Sice audiodekodér VS1053b pracuje na 3,3V logike, ale v zapojení MP3 Shieldu sa tieto signály prevádzajú na výstupe na 5V a naopak, keďže pôvodne je tento modul určený pre pripojenie k vývojovému kitu Arduino. Bolo teda potrebné signály z M68EVB908GB60, ktoré pracujú na 3,3V logike, previesť na 5V logiku. Kvôli tomu bolo treba pridať aj 5V zdroj napätia, ktorý potrebujeme na napájanie MP3 Shieldu.

6.2 Návrh schémy a dosky plošných spojov

6.2.1 Prepojenie potrebných pinov

Pred návrhom schémy bolo potrebné zvážiť, ktoré piny Shieldu budeme pri práci potrebovať a taktiež, na ktoré piny kitu ich privedieme.

PTA6/KBD6	1	2	PTA7/KBD7
PTA4/KBD4	3	4	PTA5/KBD5
PTA2/KBD2	5	6	PTA3/KBD3
PTA0/KBD0	7	8	PTA1/KBD1
3.3V	9	10	GND
PTC6	11	12	PTC7
PTC4/CLKOUT	13	14	PTC5
PTC2/SDA	15	16	PTC3/SCL
PTC0/TXD2	17	18	PTC1/RXD2
3.3V	19	20	GND
PTD6/TPM2CH3	21	22	PTD7/TPM2CH4
PTD4/TPM2CH1	23	24	PTD5/TPM2CH2
PTD2/TPM1CH2	25	26	PTD3/TPM2CH0
PTD0/TPM1CH0	27	28	PTD1/TPM1CH1
3.3V	29	30	GND
PTE6	31	32	PTE7
PTE4/MOSI	33	34	PTE5/SPSCK
PTE2/SS*	35	36	PTE3/MISO
PTE0/TXD1	37	38	PTE1/RXD1
3.3V	39	40	GND
PTF6	41	42	PTF7
PTF4	43	44	PTF5
PTF2	45	46	PTF3
PTF0	47	48	PTF1
3.3V	49	50	GND
PTG6	51	52	PTG7
PTG4	53	54	PTG5
PTG2/EXTAL	55	56	PTG3
PTG0/BGND/MS	57	58	PTG1/XTAL
RESET*	59	60	IRQ

Obr. 16. Použité piny kitu M68EVB908GB60 [1]

Vyššie vidno, ktoré piny kitu boli vyvedené z MCU portu. Keďže pri prenose dát bolo použité rozhranie SPI, potrebovali sme piny, ktoré sú určené pre túto komunikáciu a to PTE3/MISO (Master In, Slave Out), PTE4/MOSI (Master Out, Slave In) a PTE5/SPSCK (hodinový signál). Pin PTE2/SS (Slave Select) nebudeme potrebovať, keďže pôjde o prenos dát medzi 2 zariadeniami, teda jedno bude v režime Master (kit) a druhé v režime Slave (MP3 Shield). Na ovládanie samotného MP3 Shieldu boli použité piny PTC3-7.

Tab. 1. Priradenie pinov kitu k pinom MP3 Shield-u

Piny - Kit M68EVB908GB60		Piny - MP3 Shield
Vstup	PTE3/MISO	MISO
Výstup	PTE4/MOSI	MOSI
Výstup	PTE5/SPSCK	SCK
Výstup	PTC3	SD - CS
Vstup	PTC4	MP3 - DREQ
Výstup	PTC5	MP3 - RST
Výstup	PTC6	MP3 - DCS
Výstup	PTC7	MP3 - CS

6.2.2 Zdroj 5V

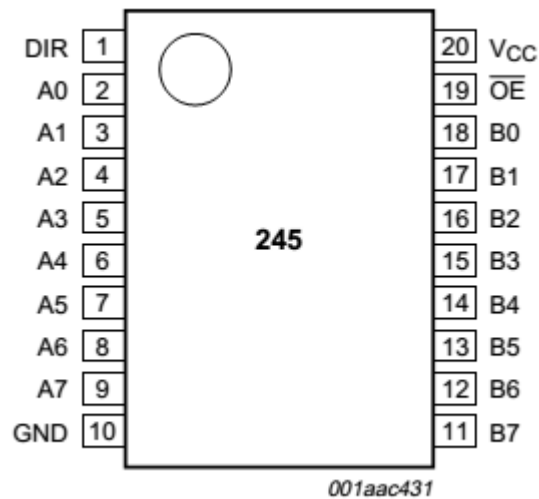
Problém s prevodom napät'ových úrovní si vyžadoval riešenie a začiatkom tohto riešenia je pridať zdroj 5V napätia, keďže aj MP3 Shield potrebuje napájanie 5V.

Z kitu je možné vyviesť 9V, keďže práve takýmto napätím je napájaný kit. Takže sme použili týchto 9V, aby sme si vytvorili stabilizovaný zdroj napätia 5V pomocou stabilizátora 7805TV. [11]

Pre ochranu pred prepólovaním bola použitá dióda. Pre vyhladenie napätia boli použité 4 kondenzátory.

6.2.3 Prevod napät'ových úrovní

O prevod napät'ových úrovní sa stará integrovaný logický obvod 74HCT245N, ktorý je schopný meniť napät'ové úrovne. V našom prípade potrebujeme prevádzať 3,3V na 5V.



Obr. 17. Rozloženie pinov integrovaného obvodu 74HCT245N [12]

Vcc je napájací pin, ku ktorému je privedené napájacie napätie zo zdroja 5V. Pin GND je spojený so zemou.

Tab. 2. Spôsoby nastavenia integrovaného obvodu [12]

Input		Input/output	
$\overline{\text{OE}}$	DIR	An	Bn
L	L	A = B	input
L	H	input	B = A
H	X	Z	Z

V tabuľke vyššie vidíme ako je možné nastaviť správanie integrovaného obvodu. Zvolil som druhú možnosť, keďže Piny A som použil ako vstupné, ktoré chcem dostať na Piny B. Preto som na pin OE spojil so zemou, keďže som tam potreboval pripojiť na trvalo logickú 0. Na pin DIR som privedol zo zdroja 5V, keďže tam bolo potrebné priviesť referenčné napätie, teda napäťovú úroveň, ktorá bude na výstupe prezentovať logickú 1. [12]

Cez integrovaný obvod som previedol len signály, ktoré boli výstupné pre vývojový kit a vstupovali do MP3 Shieldu. Výstupy z MP3 Shieldu, teda MISO a MP3 – DREQ nie je potrebné prevádzať.

6.3 Nastavenie kitu M68EVB908GB60

Pre správnu funkčnosť tejto práce bolo potrebné nastaviť prepínače na nastavenie sériových COM portov, tak aby bol aktívny sériový port COM2. K tomu potrebujeme nastaviť prepínače na COM_SW.

Tab. 3. Nastavenie prepínačov na COM_SW vývojového kitu [1]

COM SW	HCS08 Port	COM Signal	MCU PORT
1	PTE1/RXD1	COM1 RXD IN	38
2	PTF6	COM2 CD OUT	41
3	PTD6/TPM2CH3	COM2 DTR IN	21
4	PTC1/RXD2	COM2 RXD IN	18
5	PTF7	COM2 RTS OUT	42
6	PTD7/TPM2CH4	COM2 CTS IN	22
7	PTC1/RXD2	422 485 RXD IN	18
8	PTD0/TPM1CH0	BUZZER OUT	27

COM_SW poskytuje pre HCS08 vstupne-výstupný port pre komunikáciu pripájaných periférnych zariadení na vývojovú dosku. To umožňuje používateľovi použiť porty HCS08 na komunikáciu a prenos alebo použiť tieto porty ako štandardné I/O. [1]

Pre naše účely potrebujeme nastaviť COM_SW tak, aby sme povolili komunikáciu po sériovej linke COM1 a COM2. Takže nastavíme prepínače 1 a 4 do stavu ON, tzn. prepneme ich do hornej pozície.

7 SOFTWAREVÁ ČASŤ

Softwarová časť pozostáva z týchto súčastí:

- Podporný serverový program
- Knižnice funkcií
- Ukážky použitia funkcií v programe

7.1 Podporný serverový program

Serverový program som vytvoril v programovacom jazyku JAVA, keďže sme s ním pracovali aj počas štúdia. Tento program používa funkcie knižnice JSSC (java-simple-serial-connector). Táto knižnica slúži na komunikáciu pomocou sériového portu. [13]

GUI bolo vytvorené pomocou knižnice SWT vo vývojovom prostredí Eclipse. [14]

Program slúži ako náhrada dátového média, takže pomocou tohto programu sa budú prenášať dáta MP3 súborov na vývojový kit.

7.1.1 Popis

Základnými parametrami programu sú zložka, v ktorej sa nachádzajú MP3 súbory a port, na ktorý sa majú vybrané dáta posielat', resp. port, ku ktorému je v našom prípade pripojený vývojový kit.

Pozor ! Súbory v zložke musia mať koncovku mp3 a musia mať názov pozostávajúci zo 7 znakov, napr. „track01.mp3“. Súbory nemôžu byť väčšie ako 25MB !

Po testovaní sériového portu som zistil, že optimálne je posielat' dáta po balíku 256B, keďže pri posielaní packetu po sériovej linke je v packete obsiahnutých len 256B posielaných dát, takže som tomu prispôbil aj serverový program. Ten si zvolený súbor rozdelí na pole, kde prvok poľa má veľkosť 256B a dané prvky posiela na vyžiadanie na sériovú linku.

7.1.2 Příkazy

Ide o jednoduchý program, který čeká na příkazy zo sériovej linky. Příkazy musia mať vždy dĺžku 10 znakov.

***dir!00000**

- Príkaz DIR. Po prijatí tohto príkazu program odošle na sériovú linku obsah zvolenej zložky v tvare Stringu, teda odošle pole znakov, ktoré je potrebné na vývojevom kite zachytiť a spracovať. Na koniec poľa sa pridá znak *, pre rozoznanie konca poľa na vývojevom kite.

***o!nazov01**

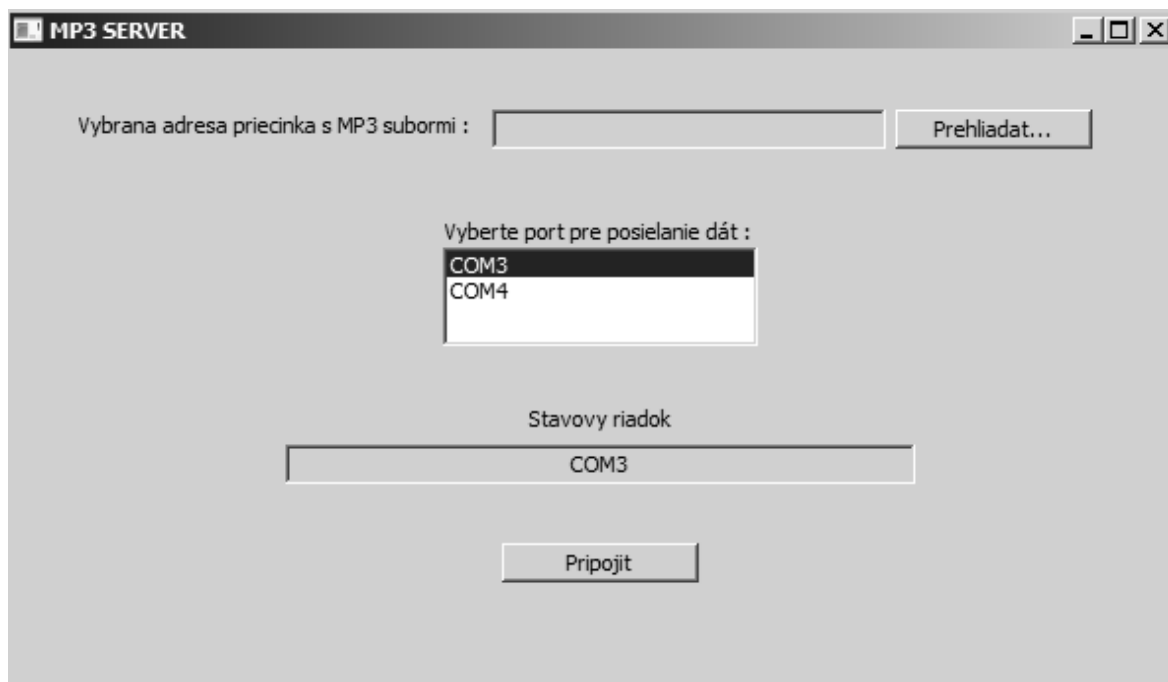
- Príkaz OPEN. Po prijatí tohto príkazu si program uloží súbor nazov01.mp3 do poľa, ktorého jeden prvok má veľkosť 256B. Následne zistí veľkosť poľa, resp. počet blokov po 256B a toto číslo odošle na sériový port.

***get!00000**

- Príkaz GET DATA. Po prijatí tohto príkazu odošle program na sériovú linku 256B prvok poľa s indexom 00000, ktorý bol zadaný za príkazom. Index teda môže byť maximálne 99999, z čoho vyplýva, že súbor, ktorý chceme posielat' nemôže mať veľkosť viac ako 25MB.

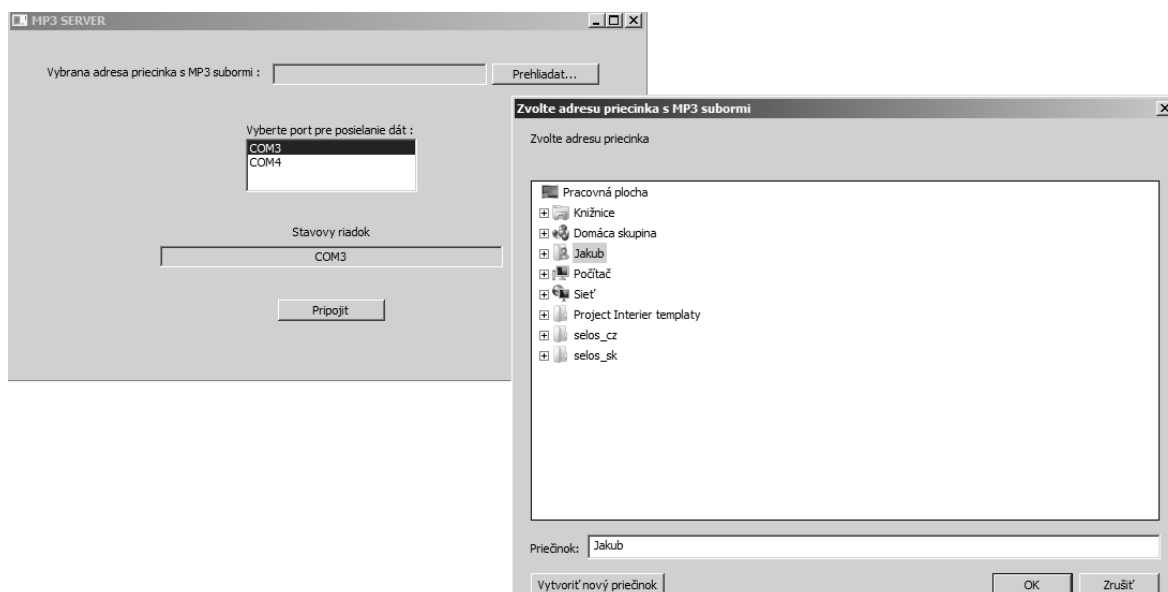
7.1.3 Grafické prostredie

Grafické prostredie bolo vytvorené pomocou knižnice SWT. Ide o jednoduché okno, v ktorom je vložené pole, kam sa vypisuje zvolený priečinok s MP3 súbormi a jednoduchý zoznam, kde sú vypísané momentálne fyzicky pripojené sériové porty.

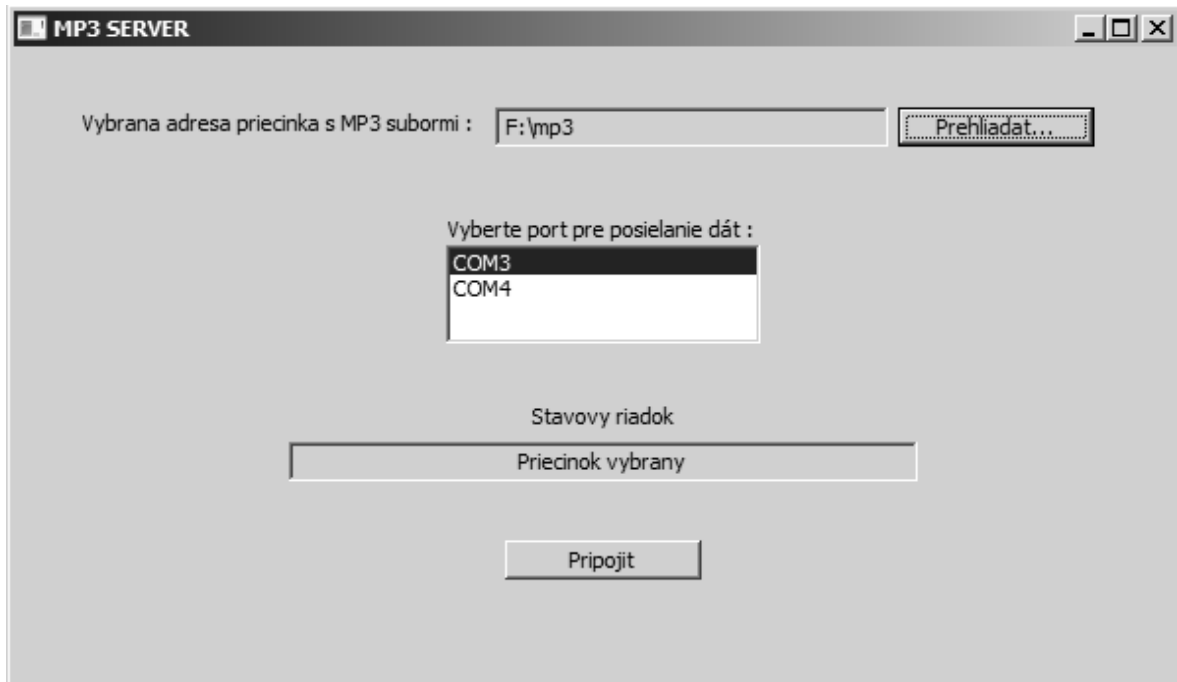


Obr. 18. Serverový program po spustení

Po stlačení tlačidla „Prehliadat...“ sa otvorí klasické okno, pre výber priečinka v počítači. Po vybratí stačí kliknúť na Ok.

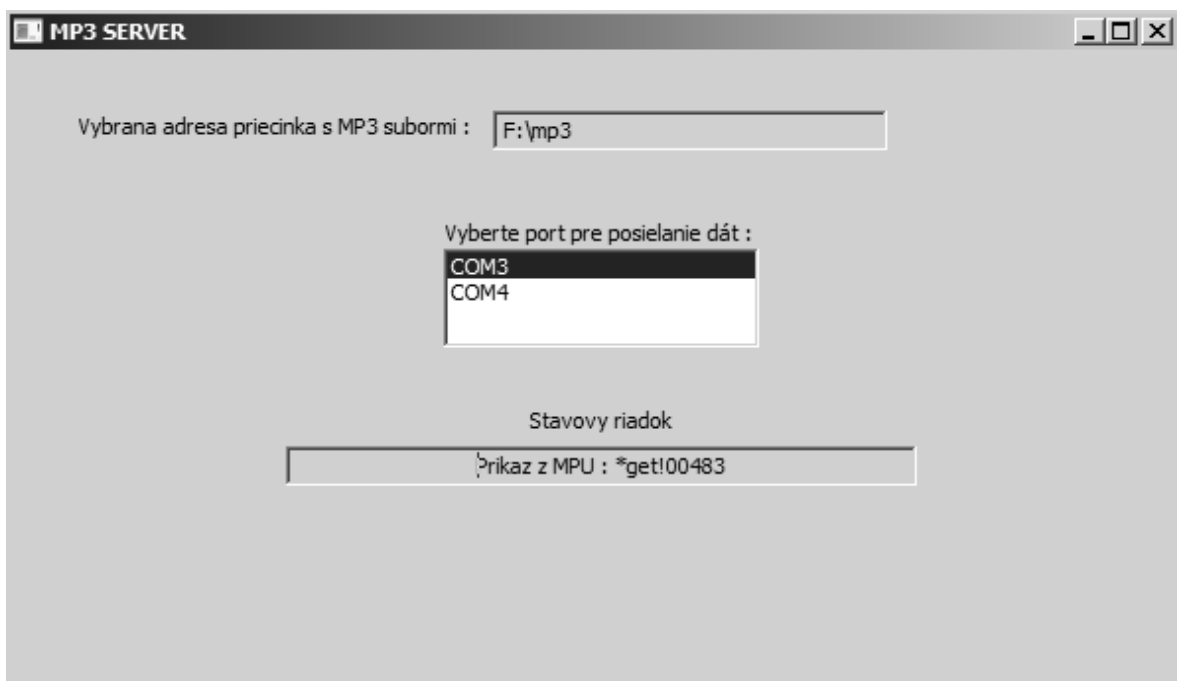


Obr. 19. Serverový program - výber priečinka



Obr. 20. Serverový program - vybraný priečinok

Po zvolení všetkých potrebných nastavení aplikácie klikneme na tlačidlo „Pripojit“ a aplikácia otvorí zadaný port a program čaká na príkazy zo servera.



Obr. 21. Serverový program – pripojený

Po pripojení je serverový program schopný prijímať príkazy cez sériovú linku a teda aj odosielať požadované dáta. Na ukážke (Obr. 21) je zachytené posielanie skladby.

7.2 Knižnice funkcií

Knižnice funkcií sú napísané v programovacom jazyku C.

Z hľadiska funkcionality môžeme knižnice rozdeliť na :

- Knižnica funkcií pre ovládanie servera
- Knižnica funkcií pre ovládanie MP3 Shieldu

Toto rozdelenie mi odporučil môj vedúci práce Ing. Petr Dostálek, keďže prídavný modul MP3 Shield disponuje aj slotom pre microSD kartu a kolega diplomant mal za úlohu vytvoriť vlastný súborový systém s následným využitím na SD karte a tým pádom by šlo použiť MP3 Shield bez servera.

7.2.1 Knižnica funkcií pre ovládanie servera

Táto knižnica obsahuje funkcie, ktoré slúžia na posielanie príkazov na server a taktiež na následné prijímanie dát zo serveru pomocou sériovej linky COM2 na vývojovom kite.

Knižnica obsahuje 3 funkcie :

- `dir`
- `openfile`
- `getdata`

7.2.1.1 Funkcia `dir`

Funkcia `dir` slúži na odoslanie príkazu `*dir!00000` na server pomocou sériovej linky. Po odoslaní tohto príkazu funkcia následne očakáva dáta a ukladá ich do poľa znakov. Ukazateľ na toto pole je parameter `buff`. Dáta očakáva až kým nepríde zo servera znak `*`, čo signalizuje koniec posielania dát a funkcia skončí. Je to funkcia bez návratovej hodnoty.

```
void dir(char *buff)
{
    int i=0;
    char znak;

    sci2_str_out("dir!00000");

    i = 0;
    while(1)
    {
        znak = sci2_in();
        if (znak == '*') break;
        buff[i]=znak;
        buff[i+1]='\0';
        i++;
    }
}
```

Obr. 22. Funkcia dir

7.2.1.2 Funkcia openfile

Funkcia openfile slúži na odoslanie príkazu pre otvorenie súboru. Odosielaný reťazec sa skladá z príkazu *o! a ďalších 7 znakov, ktoré sú vo funkcii prijaté pomocou parametra nazov[7]. Po odoslaní príkazu funkcia prijíma dáta do poľa znakov pokiaľ nepríde znak *, ktorý signalizuje koniec odosielania dát, ktoré sa nakoniec prevedú na číslo pomocou funkcie atoi. Toto číslo je použité ako návratová hodnota funkcie.

Príklad: Keď zavoláme funkciu s parametrom týmto spôsobom `int a = openfile("track01");` Výsledný príkaz, ktorý sa odošle po sériovej linke COM 2 na server bude `*o!track01` a v premennej `a` bude uložené prijaté číslo zo sériovej linky, teda počet 256B blokov.

```
int openfile(char nazov[7])
{
    int i=0;
    char prikaz[15];
    char cislo[5];

    sprintf(prikaz, "*o!%s", nazov);
    sci2_str_out(prikaz);

    while(1)
    {
        cislo[i] = sci2_in();
        if (cislo[i] == '*')
        {
            cislo[i] = '\0';
            break;
        }
        i++;
    }

    return atoi(cislo);
}
```

Obr. 23. Funkcia openfile

7.2.1.3 Funkcia *getdata*

Funkcia *getdata* slúži na odoslanie príkazu pre vyžiadanie dát zo servera. Odosielaný reťazec pozostáva z príkazu **get!* a ďalších 5 znakov, ktoré sú prijaté do funkcie ako parameter *poradie*. Ide o číslo, ktoré charakterizuje index pre 256B časť zvukového súboru v poli. Po odoslaní príkazu sa čaká na príjem 256B packetu zvukového súboru, ktorý sa uloží do poľa, ktorého ukazateľ je prijatý ako parameter *zvuk*.

Príklad: Keď zavoláme funkciu s parametrom týmto spôsobom *getdata(zvuk,55)*; Výsledný príkaz, ktorý sa odošle po sériovej linke COM 2 na server bude **get!00055* a do poľa *zvuk* sa príjmu dáta zo sériovej linky, teda 55.blok zvukového súboru .

```
void getdata(char * zvuk,int poradie) {
    int in=0;
    char prikaz[15];

    sprintf(prikaz,"*get!%05d", poradie);
    sci2_str_out(prikaz);
    while(1)
    {
        __RESET_WATCHDOG();

        zvuk[in] = sci2_in();
        in++;
        if(in==256)
        {
            break;
        }
    }
}
```

Obr. 24. Funkcia *getdata*

7.2.2 Knižnica funkcií pre ovládanie MP3 Shieldu

Táto knižnica obsahuje funkcie pre ovládanie MP3 Shieldu, teda na manipuláciu s registrami audiodekodéra VS1053b a zároveň pre posielanie dát na dekódovanie.

V tejto knižnici a teda v jej funkciách je použité rozhranie SPI na posielanie príkazov a dát do audiodekodéra VS1053b.

Táto knižnica sa skladá z nasledujúcich dôležitých funkcií :

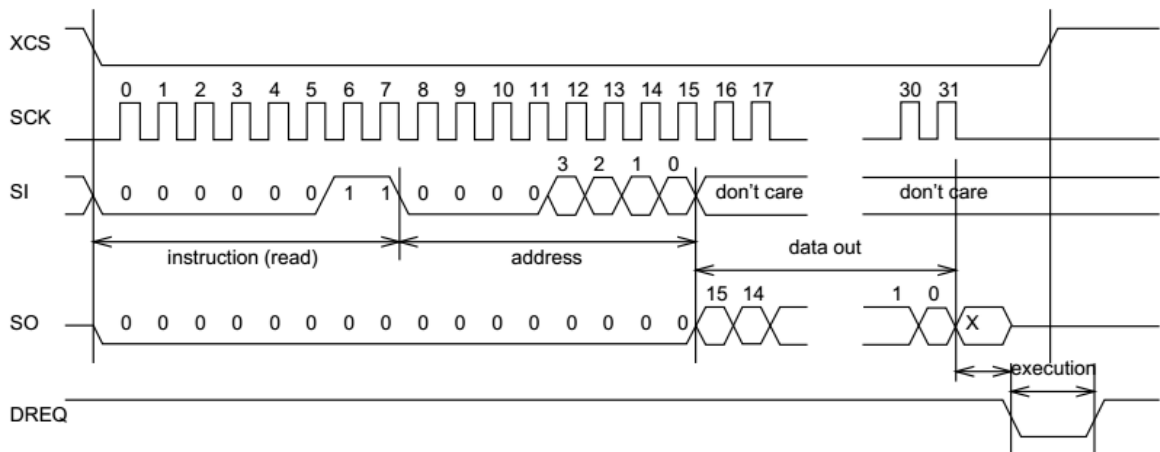
- *shield_sci_read*
- *shield_sci_readd*
- *shield_sci_write*
- *shield_sci_set*

- shield_sci_clr
- MP3SetVolume
- initMP3
- senddata
- SendEndOfFile

7.2.2.1 Funkcie shield_sci_read a shield_sci_readd

Ide o dve funkcie, ktoré zisťujú obsah registru, ale každá prijaté dáta ukladá iným spôsobom.

Zistíme, či je na pine DREQ logická 1, ak áno, môžeme pokračovať. Na pin XCS privedieme logickú 0 a pomocou rozhrania SPI sa postupne odosiela inštrukcia pre čítanie, čo je 0b00000011, ďalej sa posiela 8b adresa 16-bitového registru, z ktorého chceme čítať dáta a nakoniec posielame 16b hocijakých dát (v našom prípade dva krát 0b00000000) a medzitým sa prijíma 16b dát zo zvoleného registru. Pri SPI komunikácií musíme vždy pri posielaní 8b zároveň 8b aj prijať, takže tieto dáta musíme prečítať zo zásobníka SPI. Po skončení prenosu zistíme, či je na DREQ stále logická 1 a privedieme na XCS logickú 1. [6]



Obr. 25. SCI príkaz pre čítanie [6]

Funkcia shield_sci_read ukladá prijatých 16 bitov do poľa s dvoma prvkami, kde prvý prvok charakterizuje horný bajt a druhý spodný bajt 16bitového registru.

```

void shield_sci_read(int adresa, int * data)
{
    while(!DREQ);

    XCS = 0;

    while(SPI1S_SPTEF == 0) __RESET_WATCHDOG();
    SPI1D = 0b00000011;
    while(SPI1S_SPRF == 0) __RESET_WATCHDOG();
    tmp = SPI1D;

    while(SPI1S_SPTEF == 0) __RESET_WATCHDOG();
    SPI1D = adresa;
    while(SPI1S_SPRF == 0) __RESET_WATCHDOG();
    tmp = SPI1D;

    while(SPI1S_SPTEF == 0) __RESET_WATCHDOG();
    SPI1D = 0b00000000;
    while(SPI1S_SPRF == 0) __RESET_WATCHDOG();
    data[0] = SPI1D;

    while(SPI1S_SPTEF == 0) __RESET_WATCHDOG();
    SPI1D = 0b00000000;
    while(SPI1S_SPRF == 0) __RESET_WATCHDOG();
    data[1] = SPI1D;

    while(!DREQ);

    XCS = 1;
}

```

Obr. 26. Funkcia shield_sci_read

Funkcia shield_sci_readd ukladá prijatých 16 bitov ako číslo, integer bez znamienka.

```

unsigned int shield_sci_readd(int adresa)
{
    int hi;
    int lo;
    unsigned int vysledok;

    while(!DREQ);

    XCS = 0;

    while(SPI1S_SPTEF == 0) __RESET_WATCHDOG();
    SPI1D = 0b00000011;
    while(SPI1S_SPRF == 0) __RESET_WATCHDOG();
    tmp = SPI1D;

    while(SPI1S_SPTEF == 0) __RESET_WATCHDOG();
    SPI1D = adresa;
    while(SPI1S_SPRF == 0) __RESET_WATCHDOG();
    tmp = SPI1D;

    while(SPI1S_SPTEF == 0) __RESET_WATCHDOG();
    SPI1D = 0b00000000;
    while(SPI1S_SPRF == 0) __RESET_WATCHDOG();
    hi = SPI1D;

    while(SPI1S_SPTEF == 0) __RESET_WATCHDOG();
    SPI1D = 0b00000000;
    while(SPI1S_SPRF == 0) __RESET_WATCHDOG();
    lo = SPI1D;

    while(!DREQ);

    XCS = 1;

    vysledok = hi << 8;
    vysledok |= lo;

    return vysledok;
}

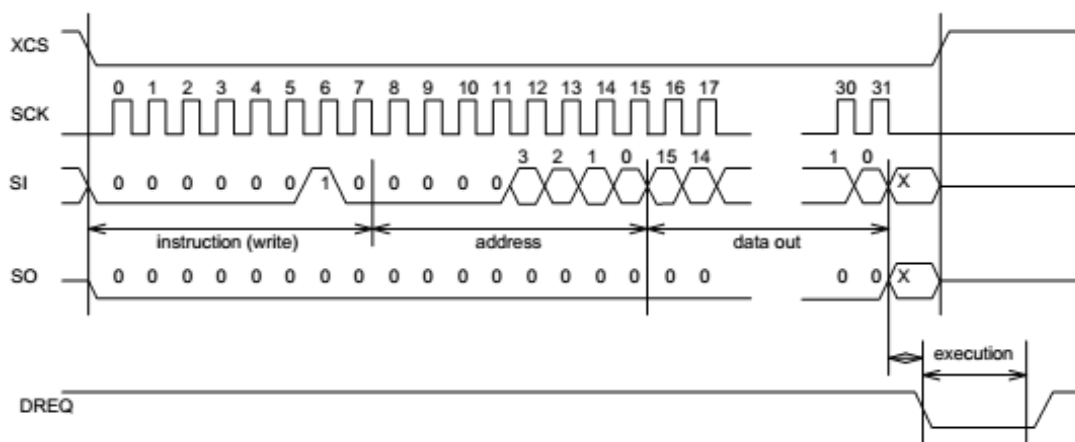
```

Obr. 27. Funkcia shield_sci_readd

7.2.2.2 Funkcia shield_sci_write

Funkcia, ktorá zapisuje pomocou rozhrania SPI do zvoleného registra vybrané dáta.

Zistíme, či je na pine DREQ logická 1, ak áno, môžeme pokračovať. Na pin XCS privedieme logickú 0 a pomocou rozhrania SPI sa postupne odosiela inštrukcia pre zápis, čo je 0b00000010, ďalej sa posiela 8b adresa 16-bitového registra, do ktorého chceme zapísať dáta a nakoniec posielame 16b dát. Pri SPI komunikácii musíme vždy pri posielaní 8b zároveň 8b aj prijať, takže tieto dáta musíme prečítať zo zásobníka SPI. Po skončení prenosu zistíme, či je na DREQ stále logická 1 a privedieme na XCS logickú 1. [6]



Obr. 28. SCI príkaz pre zápis [6]

```
void shield_sci_write(int adresa, int high, int low)
{
    while(!DREQ);

    XCS = 0;

    while(SPI1S_SPTEF == 0) __RESET_WATCHDOG();
    SPI1D = 0b00000010;
    while(SPI1S_SPRF == 0) __RESET_WATCHDOG();
    tmp = SPI1D;

    while(SPI1S_SPTEF == 0) __RESET_WATCHDOG();
    SPI1D = adresa;
    while(SPI1S_SPRF == 0) __RESET_WATCHDOG();
    tmp = SPI1D;

    while(SPI1S_SPTEF == 0) __RESET_WATCHDOG();
    SPI1D = high;
    while(SPI1S_SPRF == 0) __RESET_WATCHDOG();
    tmp = SPI1D;

    while(SPI1S_SPTEF == 0) __RESET_WATCHDOG();
    SPI1D = low;
    while(SPI1S_SPRF == 0) __RESET_WATCHDOG();
    tmp = SPI1D;

    while(!DREQ);

    XCS = 1;
}
```

Obr. 29. Funkcia shield_sci_write

7.2.2.3 Funkcia *shield_sci_set*

Táto funkcia využíva funkcie *shield_sci_read* a *shield_sci_write*. Je to vlastne také vylepšenie, v ktorom sa nemusí zadávať celých 16 bitov, ktoré chceme zapísať, ale len poradie bitu, ktorý chceme nastaviť na logickú 1. Vo funkcii si najskôr prečítame register na danej adrese a potom pomocou maskovania zapisujeme dáta podľa zadaného poradia bitu.

7.2.2.4 Funkcia *shield_sci_clr*

Táto funkcia pracuje podobne ako funkcia *shield_sci_set*, s tým rozdielom, že na zadané poradie bitu zapisuje logickú 0.

7.2.2.5 Funkcia *MP3SetVolume*

Táto funkcia len jednoducho nastavuje register, ktorý sa stará o nastavenie hlasitosti na výstupe. Dajú sa nastaviť jednotlivé kanály. Pre maximum treba zadať 0x00 a pre minimum 0xFF.

7.2.2.6 Funkcia *initMP3*

Funkcia *initMP3* slúži na inicializáciu a nastavenie všetkých potrebných portov, SPI prenosu a módu prenosu na MP3 Shielde.

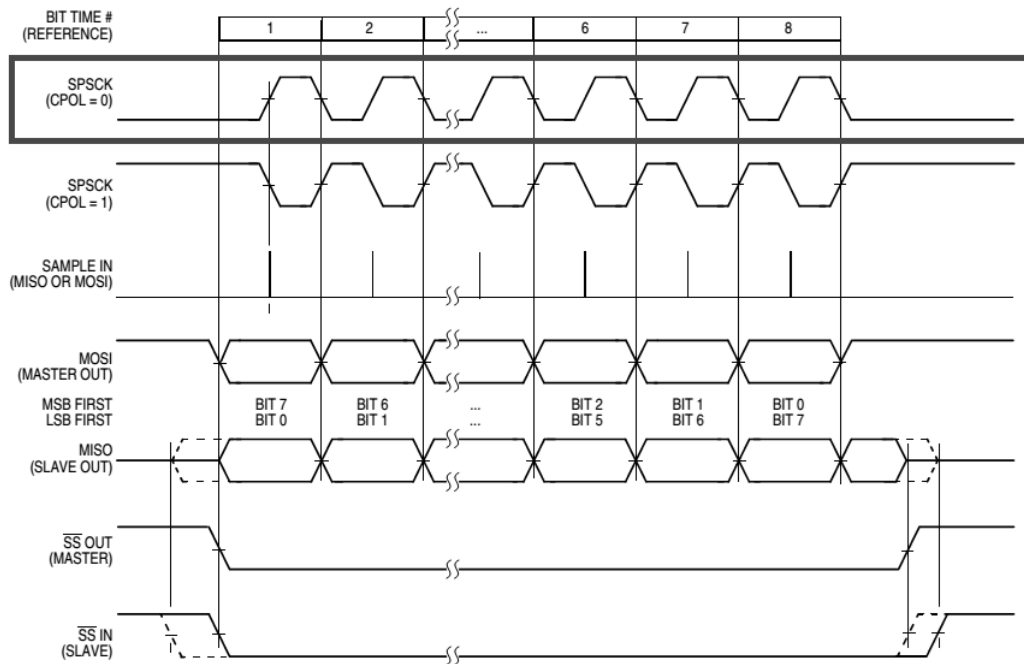
Nasledujúcim kódom sa nastavuje SPI prenos vo funkcii *initMP3*.

```
SPI1C1_SPIE = 0;  
SPI1C1_SPTIE = 0;  
SPI1C1_MSTR = 1;  
SPI1C1_CPOL = 0;  
SPI1C1_CPHA = 0;  
SPI1C1_SSOE = 0;  
SPI1C1_LSBFE = 0;  
SPI1C1_SPE = 1;  
  
SPI1C2 = 0;
```

Obr. 30. Nastavenie SPI

V tejto sekvencii sa zakazuje prerušenie, keďže je použité dotazovanie. Procesor HCS08 sa nastavuje ako Master zariadenie SPI komunikácie. Ďalej nastavujeme hodinový signál, jeho fáza a polarita je zvolená tak, že prvá hrana nastane v strede cyklu a aktívna v logickej 1,

teda nábežná hrana (Obr. 31). Keďže ide o prenos dát medzi 2 zariadeniami, teda máme len jedno Slave zariadenie (MP3 Shield), nepotrebujeme povoľovať výber Slave zariadení. Ďalej je vybrané poradie posielaných dát, ktoré bude začínať najvýznamnejším bitom a nakoniec je povolený systém SPI rozhrania. [4]



Obr. 31. Nastavenie fázy a polaroty hodinového signálu SPI rozhrania [4]

Dôležitá je inicializácia vstupných a výstupných kontaktov na ovládanie MP3 Shieldu. A to piny vývojového kitu PTC4-7.

```

//// nastavenie vystupnych kontaktov
// PTC7 = MP3-xCS
PTCDD_PTCDD7 = 1;
PTCPE_PTCPE7 = 0;
PTCD_PTCDD7 = 1;

// PTC6 = MP3-xDCS
PTCDD_PTCDD6 = 1;
PTCPE_PTCPE6 = 0;
PTCD_PTCDD6 = 1;

// PTC5 = MP3-RST
PTCDD_PTCDD5 = 1;
PTCPE_PTCPE5 = 0;
PTCD_PTCDD5 = 0;
PTCD_PTCDD5 = 1;

//// nastavenie vstupnych kontaktov
// PTC4 = MP3-DREQ
PTCDD_PTCDD4 = 0;
PTCPE_PTCPE4 = 1;

```

Obr. 32. Nastavenie vstupov a výstupov vývojového kitu

V tejto inicializačnej funkcii sa pomocou funkcie *shield_sci_write* v registri SCI_MODE nastavuje mód prehrávania a natívny mód dátovej komunikácie SDI, pri ktorom funguje prenos podobne ako pri SCI rozhraní. Ďalej sme na MP3 Shielde nastavili násobič rýchlosti SPI rozhrania a potom sme túto rýchlosť nastavili aj na vývojovom kite. [4] [6]

7.2.2.7 Funkcia *senddata*

Táto funkcia slúži na odosielanie dát pomocou dátovej komunikácie SDI. Keďže ide o natívny mód, sekvencia posielania je podobná ako pri SCI rozhraní.

Pozor ! Pred volaním tejto funkcie sa najskôr musíme uistiť, že je na pine DREQ logická 1, čo signalizuje, že dekodér MP3 Shieldu je schopný prijať 32B dát.

Na pin XDACS privedieme logickú 0 a pomocou rozhrania SPI sa postupne odosiela 32B časť zvukového súboru, ktorú do funkcie vkladáme pomocou ukazateľa na pole znakov ako parameter funkcie. Po ukončení prenosu dát privedieme na XDACS logickú 1, čím ukončíme SDI prenos. [6]

```
void senddata(char * zvuk)
{
    int in = 0;
    XDACS = 0;

    while(1)
    {
        while(SPI1S_SPTEF == 0) __RESET_WATCHDOG();
        SPI1D = zvuk[in];

        while(SPI1S_SPRF == 0) __RESET_WATCHDOG();
        tmp = SPI1D;

        in++;
        if(in==32)
        {
            break;
        }
        __RESET_WATCHDOG();
    }
    XDACS = 1;
}
```

Obr. 33. Funkcia *senddata*

7.2.2.8 Funkcia *SendEndOfFile*

Táto funkcia funguje podobne ako funkcia *senddata* a ako už z jej názvu vyplýva, posiela koniec súboru. Ide o sekvenciu, ktorá pomocou SDI rozhrania posiela na dekodér MP3 Shieldu 2048B núl, čo podľa DataSheet-u signalizuje pre dekodér koniec posielaného súboru.

```
void SendEndOfFile(void)
{
    int in=0;
    while(1)
    {
        while(!DREQ) __RESET_WATCHDOG();
        XDCS = 0;

        while(SPI1S_SPTEF == 0) __RESET_WATCHDOG();
        SPI1D = 0;

        while(SPI1S_SPRF == 0) __RESET_WATCHDOG();
        tmp = SPI1D;
        in++;
        if(in==2048)
        {
            break;
        }
    }
    while(!DREQ) __RESET_WATCHDOG();
    XDCS = 1;
}
```

Obr. 34. Funkcia *SendEndOfFile*

7.3 Ukážka použitia funkcií v programe

Ukážka použitia funkcií (Obr. 35) je sekvencia, ktorá slúži na prehratie súboru uloženého na disku. Tento súbor je vďaka podpornému serverovému programu posielaný na dotaz na vybranú sériovú linku.

```
char data[256];
char mp3DataBuffer[32];
int h,i,k,velkost;
initMP3();
velkost = openfile("track01");
h=0;
while(1)
{
  getdata(data,h);
  for(i = 0 ; i < 8 ; i++)
  {
    for(k = 0 ; k < 32 ; k++)
    {
      mp3DataBuffer[k] = data[i*32+k];
    }
  }
  while(!DREQ) __RESET_WATCHDOG();
  senddata(mp3DataBuffer);
}
h++;
if(h == velkost)
{
  break;
}
}
SendEndOfFile();
```

Obr. 35. Ukážka použitia funkcií

Vo vrchnej časti obrázku je zobrazené, aké premenné je potrebné si zaviesť. Následne vidno, že je použitá funkcia *initMP3*, ktorá slúži na inicializáciu potrebných portov a nastavenie SPI prenosu. Pomocou funkcie *openfile("track01")*, ktorá odošle príkaz **o!track01* na server, sa do premennej *velkost* uloží veľkosť poľa, do ktorého bol v serverovom programe uložený súbor *track01.mp3*, resp. počet 256B prvkov. V nekonečnom cykle *while* sa žiadajú 256B dáta zo servera pomocou funkcie *getdata*, funkcia odosiela príkaz **get!00000*, číslo za príkazom **get!* sa inkrementuje až kým sa nedostane po *velkost*. Pomocou *getdata* sa dáta ukladajú do poľa *data*. Ale keďže audio dekodér je schopný detegovať len, ak má 32B voľného priestoru v zásobníku, čo signalizuje pin DREQ, je potrebné si prijaté 256B pole rozdeliť 8x na 32B pole. Takže si postupne z poľa *data* ukladáme 32B do poľa *mp3DataBuffer*. Z tohto poľa je možné po overení pinu DREQ poslať dáta na sériovú linku, resp. na audiodekodér, pomocou funkcie *senddata*. Po odoslaní všetkých, resp. po ukončení nekonečného cyklu sa nakoniec odošle ukončovacia sekvencia pomocou funkcie *SendEndOfFile*.

ZÁVER

Cieľom tejto bakalárskej práce bolo vyhotoviť výukový modul MP3 prehrávač, ktorý sa bude používať pri výuke na predmete „Programovanie mikropočítačov“.

Samotný modul MP3 prehrávača sa skladá z MP3 Player Shield-u od spoločnosti Sparkfun a prepojovacej dosky medzi MP3 Shield-om a vývojovým kitom. Prepojovacia doska sa stará aj o prevod napät'ových úrovní vďaka intergrovanému logickému obvodu 74HCT245N a zdroju 5V napätia, ktorý je tvorený stabilizátorom a vyhladzovacími kondenzátormi. Prevod napät'ových úrovní je potrebný, pretože MP3 Shield pracuje na 5V logike a vývojový kit M68EVB908GB60 pracuje na 3,3V logike.

Po vyhotovení prepojovacej dosky bolo potrebné vyvinúť knižnicu funkcií, ktoré budú ovládať Audiodekodér VS1053b, ktorý je súčasťou MP3 Shieldu. Po preštudovaní dokumentácie dekodéra bola vyvinutá knižnica s funkciami, ktoré pomocou SPI rozhrania komunikujú s dekodérom, dokážu pracovať s registrami dekodéra a dokážu posielat' zvukové dáta, ktoré dekodér spracúva a odosiela na výstup.

Pre spracúvanie MP3 súborov bol vytvorený podporný serverový program v programovacom jazyku JAVA, pre ktorý bolo vytvorené aj jednoduché grafické prostredie. Po spustení aplikácie na počítači je potrebné vybrať priečinok, ktorý obsahuje mp3 súbory. Ďalej je potrebné vybrať komunikačný port, cez ktorý bude prebiehať komunikácia. Následne už len kliknúť na Pripojiť, čím sa otvorí vybraný port pre komunikáciu.

Zároveň s vývojom JAVA aplikácie bolo vytvorená knižnica s funkciami, ktoré ovládajú komunikáciu po sériovej linke, resp. funkcie, ktoré posielajú príkazy na server a žiadajú dáta. Boli vytvorené 3 jednoduché príkazy, ktoré stačia na toto jednoduché ovládanie : dir, openfile a getdata.

Po vytvorení knižníc na komunikáciu so serverovou aplikáciou a dekodérom MP3 Shieldu bol nakoniec vytvorený ukázkový program, ktorý dokáže prehrať MP3.

ZOZNAM POUŽITEJ LITERATURY

- [1] AXIOM MANUFACTURING. *M68EVB908GB60 Development Board for Freescale MC9S08GB60, Rev. C* [online]. 2006 [cit. 2015-02-02]. Dostupné z: <http://www.axman.com>
- [2] FREESCALE SEMICONDUCTOR. *CPU08 Central Processor Unit Reference Manual* [online]. 2001 [cit. 2015-02-02]. Dostupné z: <http://www.freescale.com>
- [3] FREESCALE SEMICONDUCTOR. *HCS08 Family Reference Manual, Rev.1.* [online]. 2003 [cit. 2015-02-02]. Dostupné z: <http://www.freescale.com>
- [4] FREESCALE SEMICONDUCTOR. *MC9S08GB/GT Data Sheet, Rev.2.3.* [online]. 2004 [cit. 2015-02-02]. Dostupné z: <http://www.freescale.com>
- [5] JURÁNEK, Antonín a Miroslav HRABOVSKÝ. *EAGLE pro začátečníky /: uživatelská a referenční příručka : 2. vydání.* Praha: BEN – technická literatura, 2007, 191 s. ISBN 80-730-0213-2.
- [6] VLSI SOLUTION. *VS1053b – Ogg Vorbis/MP3/AAC/WMA/FLAC/MIDI Audio Codec Circuit, version 1.* [online]. 2012. Dostupné z: <http://www.vlsi.fi>
- [7] OLIVKA, Petr. *Architektura počítačů* [online]. Ostrava, 2010 [cit. 2015-05-10]. Dostupné z: <http://poli.cs.vsb.cz/edu/arp/down/archpoc.pdf>. Studijní materiál pro předmět Architektury počítačů.
- [8] GREGAR, Petr. *Zvukový formát MP3* [online]. Pardubice, 2007 [cit. 2015-05-15]. Dostupné z: https://dspace.upce.cz/bitstream/10195/24757/1/GregarP_Zvukovy%20format_JH_2007.pdf. Bakalářská práce.
- [9] RAISSI, Rassol. *The Theory Behind Mp3* [online]. 2002 [cit. 2015-05-15]. Dostupné z: http://www.mp3-tech.org/programmer/docs/mp3_theory.pdf
- [10] SPARKFUN. *SparkFun MP3 Player Shield* [online]. [cit. 2015-05-10]. Dostupné z: <https://www.sparkfun.com/products/10628>
- [11] STMICROELECTRONICS. *L7800 SERIES: POSITIVE VOLTAGE REGULATORS* [online]. 2004 [cit. 2015-05-15]. Dostupné z: <http://www.ges.cz/sheets/l/178xx.pdf>
- [12] PHILIPS. *74HC245; 74HCT245* [online]. 2005 [cit. 2015-05-15]. Dostupné z: http://www.nxp.com/documents/data_sheet/74HC_HCT245.pdf

-
- [13] GOOGLE. *JSSC - java serial port communication library* [online]. 2013 [cit. 2015-05-16]. Dostupné z: <https://code.google.com/p/java-simple-serial-connector/>
- [14] ECLIPSE. *SWT: The Standard Widget Toolkit* [online]. 2014 [cit. 2015-05-16]. Dostupné z: <https://www.eclipse.org/swt/>

ZOZNAM POUŽITÝCH SYMBOLOV A SKRATIEK

MCU	Mikro počítač.
CISC	Počítač s rozsiahlou inštrukčnou sadou.
RISC	Počítač s obmedzenou inštrukčnou sadou.
KB	KiloByte.
A/D	Analógovo – digitálny.
D/A	Digitálno – analógový.
KBI	Klávesnicový prerušovací systém.
SCI	Sériové komunikačné rozhranie.
SCI	Sériové radiace rozhranie.
SPI	Sériové periférne rozhranie.
SP	Ukazateľ zásobníka.
PC	Programový čítač.
LIFO	Posledný dnu prvý von.
LED	Svetlo emitujúca dióda.
DIP	Pozičný prepínač.
LCD	Displej s kvapalnými kryštálmi.
MOSI	Riadiaci von, podriadený dnu.
MISO	Riadiaci dnu, podriadený von.
SPSCK	Hodinový signál sériového periférneho rozhrania.
RAM	Volatilná elektronicky programovateľná pamäť s ľubovoľným prístupom.
SDI	Sériové dátové rozhranie.
DREQ	Požiadavka na dáta.
MP3	Moving Picture Experts Group, Audio Layer III.
MPEG	Moving Picture Experts Group.

ZOZNAM OBRÁZKOV

<i>Obr. 1. Von Neumannova architektúra</i>	12
<i>Obr. 2. Harvardská architektúra</i>	12
<i>Obr. 3. Registre HCS08 [3]</i>	14
<i>Obr. 4. Vývojový kit Freescale M68EVB908GB60 [1]</i>	17
<i>Obr. 5. Prepojenie dvoch zariadení pri rozhraní SPI [4]</i>	18
<i>Obr. 6. Blokovaná schéma mikropočítača MC9S08GB60 – SPI rozhranie [4]</i>	19
<i>Obr. 7. MP3 Player Shield [10]</i>	20
<i>Obr. 8. Audiodekodér VS1053b</i>	21
<i>Obr. 9. Blokovaná schéma dekodéru</i>	24
<i>Obr. 10. Softwarový dekodér WinPlay3</i>	25
<i>Obr. 11. Softwarový dekodér Winamp</i>	25
<i>Obr. 12. Ukážka vývojového prostredia so spusteným Debuggerom</i>	26
<i>Obr. 13. EAGLE - Editor schém – ukážka</i>	27
<i>Obr. 14. EAGLE - Editor spojov - ukážka</i>	28
<i>Obr. 15. Blokovaná schéma prepojenia</i>	30
<i>Obr. 16. Použitie piny kitu M68EVB908GB60 [1]</i>	31
<i>Obr. 17. Rozloženie pinov integrovaného obvodu 74HCT245N [12]</i>	33
<i>Obr. 18. Serverový program po spustení</i>	37
<i>Obr. 19. Serverový program - výber priečinka</i>	37
<i>Obr. 20. Serverový program - vybraný priečink</i>	38
<i>Obr. 21. Serverový program – pripojený</i>	38
<i>Obr. 22. Funkcia dir</i>	40
<i>Obr. 23. Funkcia opendir</i>	40
<i>Obr. 24. Funkcia getdata</i>	41
<i>Obr. 25. SCI príkaz pre čítanie [6]</i>	42
<i>Obr. 26. Funkcia shield_sci_read</i>	43
<i>Obr. 27. Funkcia shield_sci_readd</i>	43
<i>Obr. 28. SCI príkaz pre zápis [6]</i>	44
<i>Obr. 29. Funkcia shield_sci_write</i>	44
<i>Obr. 30. Nastavenie SPI</i>	45
<i>Obr. 31. Nastavenie fázy a polarizácie hodinového signálu SPI rozhrania [4]</i>	46
<i>Obr. 32. Nastavenie vstupov a výstupov vývojového kitu</i>	46

<i>Obr. 33. Funkcia senddata</i>	<i>47</i>
<i>Obr. 34. Funkcia SendEndOfFile</i>	<i>48</i>
<i>Obr. 35. Ukážka použitia funkcií</i>	<i>49</i>

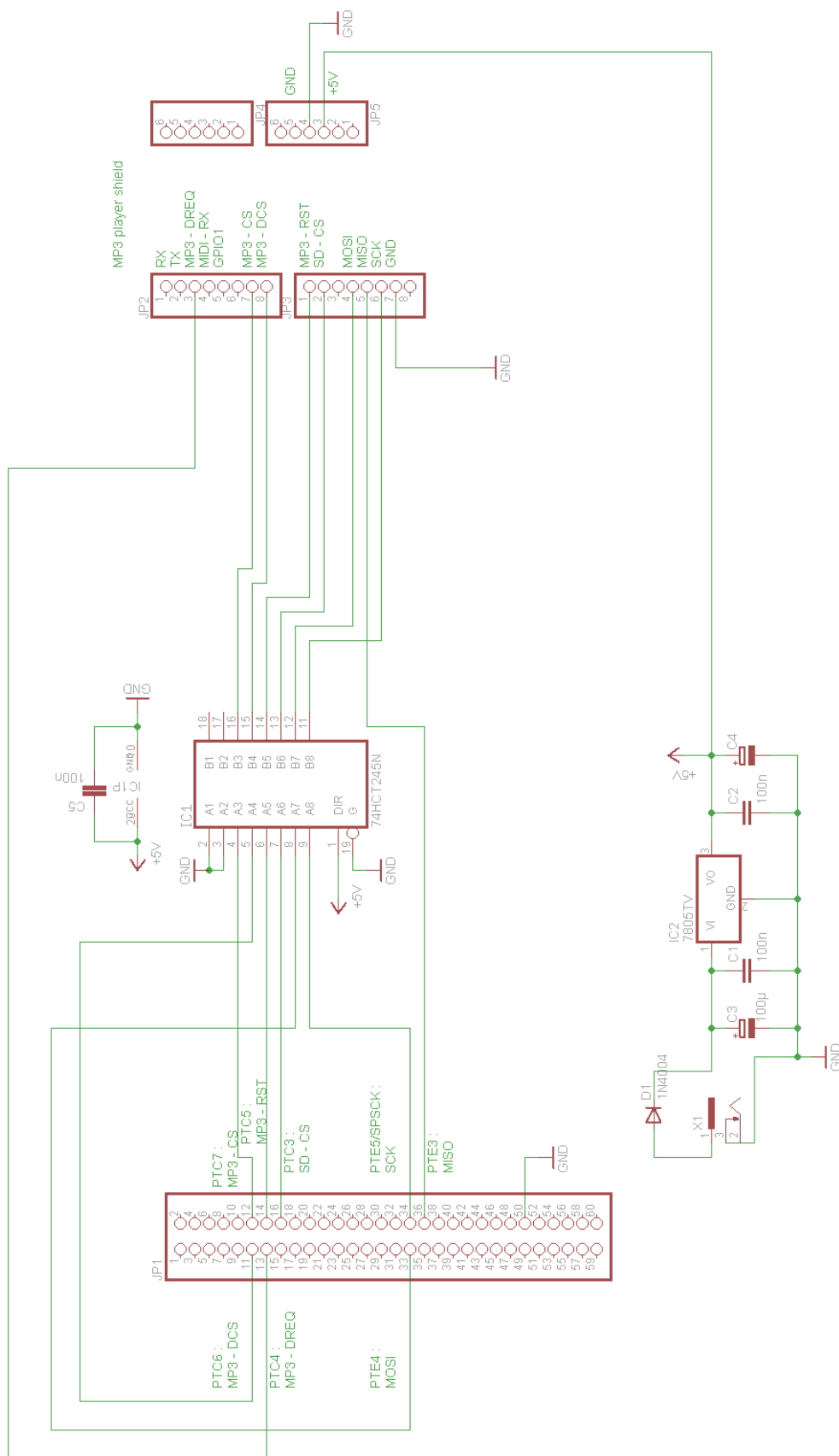
ZOZNAM TABULIEK

<i>Tab. 1. Priradenie pinov kitu k pinom MP3 Shiled-u</i>	<i>32</i>
<i>Tab. 2. Spôsoby nastavenia integrovaného obvodu [12]</i>	<i>33</i>
<i>Tab. 3. Nastavenie prepínačov na COM_SW vývojového kitu [1]</i>	<i>34</i>

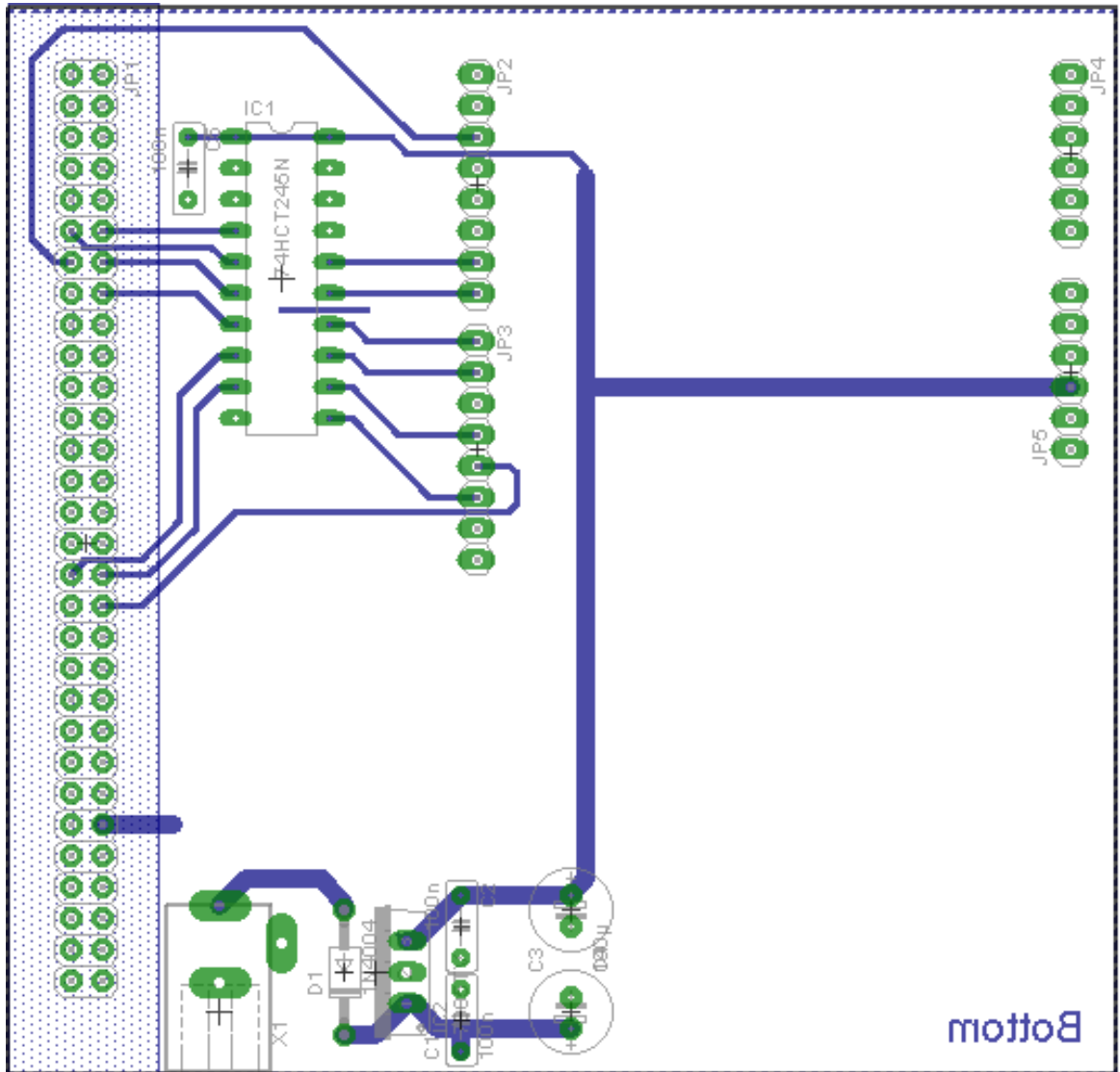
ZOZNAM PRÍLOH

- P I Schéma MP3 Shield-u
- P II Schéma prepojenia MP3 Shieldu a vývojovéhokitu M68EVB908GB60
- P III Doska plošných spojov
- P IV Fotografie dosky plošných spojov
- P V Fotografia kompletného zapojenia

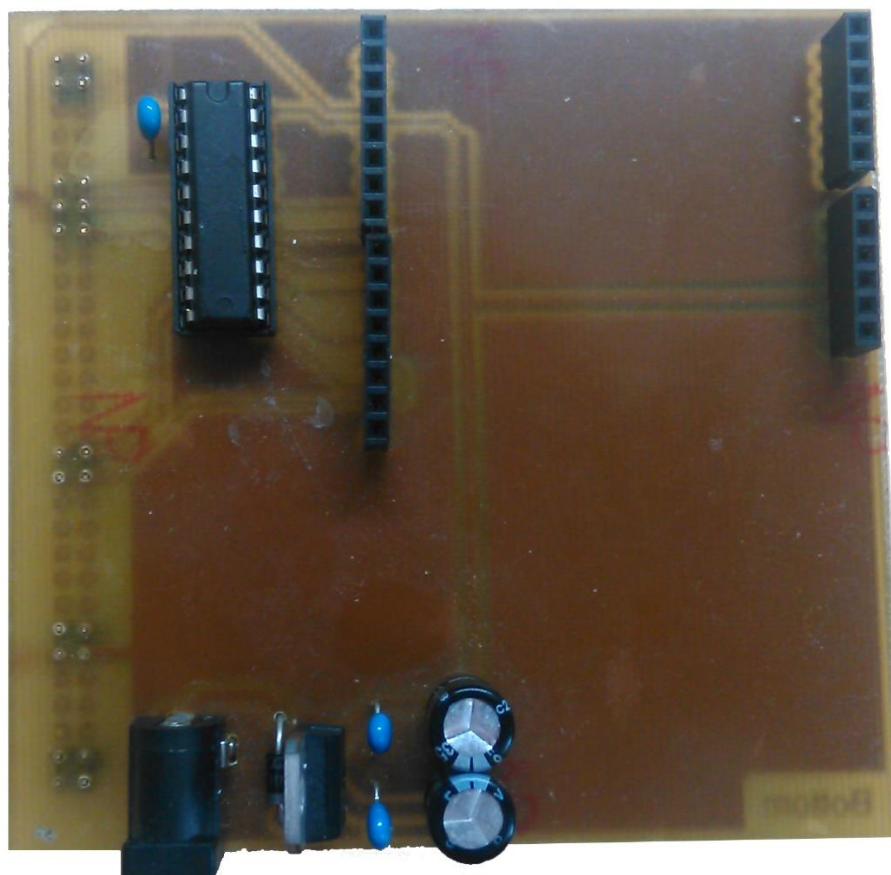
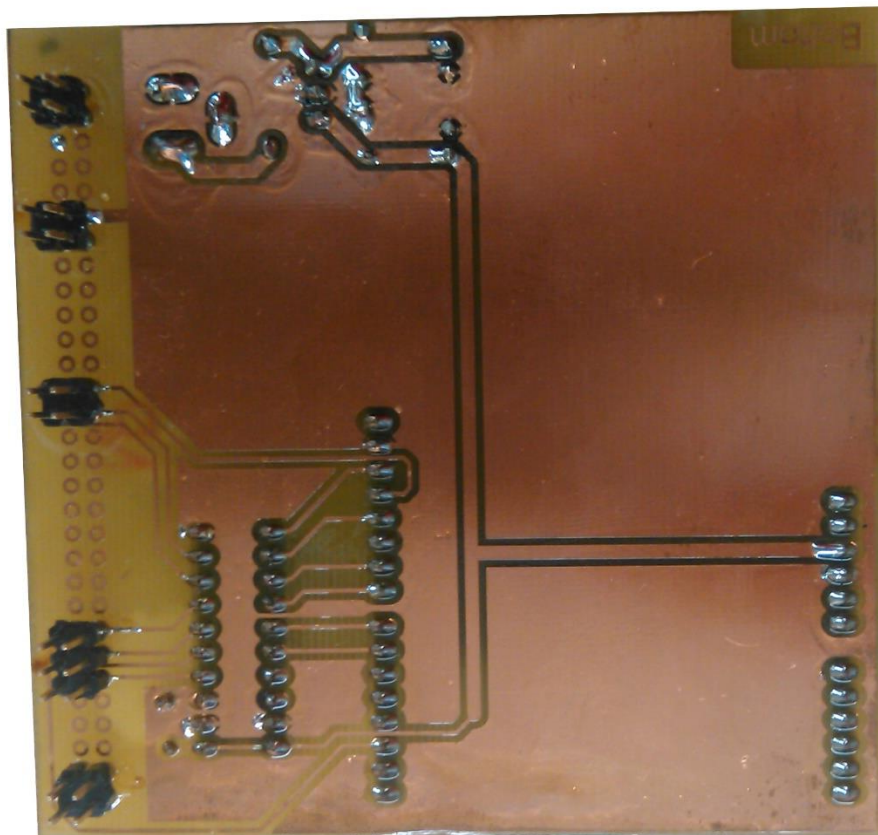
PRÍLOHA P II: SCHÉMA PREPOJENIA MP3 SHIELDU A VÝVOJOVÉHO KITU M68EVB908GB60



PRÍLOHA P III: NÁVRH DOSKY PLOŠNÝCH SPOJOV



PRÍLOHA P IV: FOTOGRAFIE DOSKY PLOŠNÝCH SPOJOV



PRÍLOHA P V: FOTOGRAFIA KOMPLETNÉHO ZAPOJENIA

