Jednoduchý záznamník zvuku s mikropočítačem

Simple sound recorder with microcontroler

Miroslav Smolinský

Bakalářská práce
2010

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2009/2010

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Miroslav SMOLINSKÝ**
Osobní číslo:	**A07095**
Studijní program:	**B 3902 Inženýrská informatika**
Studijní obor:	**Informační a řídicí technologie**
Téma práce:	**Jednoduchý záznamník zvuku s mikropočítačem**

Zásady pro vypracování:

1. **Seznamte se s existujícími konstrukcemi podobných zařízení a navrhněte vhodnou koncepci jednoduchého modulu pro záznam a přehrávání zvuku.**
2. **Navrhněte zapojení obvodu.**
3. **Realizujte funkční prototyp zařízení.**
4. **Pro zařízení vytvořte programové vybavení a ověřte jeho funkci.**

Rozsah bakalářské práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. **PINKER, Jiří. Mikroprocesory a Mikropočítače. Praha : BEN – technická literatura, 2004. 220 s. ISBN 80-7300-110-1.**

2. **MANN, Burkhard. C pro mikrokontroléry. Praha : BEN – technická literatura, 2004. 280 s. ISBN 80-7300-077-6.**

3. **AVR335: Digital Sound Recorder with AVR and DataFlash [online]. Atmel Corporation, 2005 [cit. 2010-01-25]. Dostupný z WWW: >http://www.atmel.com/dyn/resources/prod_documents/doc1456.pdf< .**

4. **ATMega8 Datasheet [online]. Atmel Corporation, 2009 [cit. 2010-01-26]. Dostupný z WWW: >http://www.atmel.com/atmel/acrobat/doc2486.pdf<**

5. **AT45DB321D DataFlash Datasheet [online]. Atmel Corporation, 2009 [cit. 2010-01-25]. Dostupný z WWW: >http://www.atmel.tw/dyn/resources/prod_documents/doc3597.pdf<**

6. **CATSOULIS, John. Designing Embedded Hardware. O'Reilly Media, 2005. 400 s. ISBN 978-0-596-00755-3.**

Vedoucí bakalářské práce: **Ing. Jan Dolinay**
Ústav automatizace a řídicí techniky

Datum zadání bakalářské práce: **5. března 2010**

Termín odevzdání bakalářské práce: **1. června 2010**

Ve Zlíně dne 5. března 2010

L.S.

prof. Ing. Vladimír Vašek, CSc.
děkan

doc. Ing. Ivan Zelinka, Ph.D.
ředitel ústavu

ABSTRAKT

Cílem této práce je navrhnout a vytvořit funkční model jednoduchého záznamníku zvuků určeného primárně na záznam lidského hlasu, a vysvětlit jeho princip fungování . Záznamník je konstruován pomocí mikropočítače a datové flash paměti tak, aby ho bylo možné napájet pomocí baterií. Disponuje základními funkcemi, jako jsou nahrávání zvuků pomocí analogově-digitální konverze, přehrávání nahraných zvukových záznamů využitím pulzně šířkové modulace, a mazání údajů uložených v dataflash paměti.

Klíčová slova: záznamník , mikropočítač, dataflash.

ABSTRACT

The goal of this thesis is to design and create a model of a simple sound recorder, designed specifically to record human voice, and explain the basis of its behavior. Recorder is designed using a microcomputer and a dataflash, to allow a battery power supply. It has basic functions, such as recording of sound through a analog-digital conversion, playback of sound recordings via pulse width modulation, and deletion of data recorded in the dataflash memory.

Keywords: recorder, microcontroler, dataflash.

Poďakovanie

Rád by som sa poďakoval pánovi Ing. Janovi Dolinayovi, vedúcemu mojej bakalárskej práce, za jeho cenné rady a pripomienky. Poďakovať by som sa chcel aj svojej rodine a priateľom za ich podporu a pochopenie, počas písania tejto práce.

Prohlašuji, že

- beru na vědomí, že odevzdáním 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 bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk 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 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 –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 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 bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem 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 bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně …….

podpis diplomanta

OBSAH

Úvod

S mikrokontrolérmi sa dnes človek stretá v každodennom živote takmer všade. Od hodiniek cez kalkulačky a domáce spotrebiče, až po dôležité obvody v automobiloch či v zariadeniach v zamestnaní. Tieto programovateľné obvody poskytujú veľmi veľa jednoduchých a praktických riešení v aplikáciách, kde by bolo podobné riešenie pomocou jednoduchej logiky náročné a menej výhodné. Navyše je možné bez zmeny v hardwérovom zapojení obvodu preprogramovaním zmeniť funkciu mikrokontroléra, čo robí tieto obvody ešte efektívnejšie využiteľnými. Problematike programovania mikrokontrolérov a ich uplatneniu v praktickom živote sa venujem už od strednej školy, preto som sa rozhodol vypracovať túto prácu práve s ich využitím. Cieľom tejto práce je navrhnúť a vyrobiť funkčný model jednoduchého záznamíka zvukov určeného predovšetkým na nahrávanie ľudského hlasu, ktorý disponuje základnými funkciami, ako je nahrávanie hovoreného slova, prehrávanie nahraných záznamov, ako aj ich mazanie. Získané dáta (zvukový záznam) sa digitálne zaznamenávajú na dataflash. V nasledujúcich kapitolách je popísaná teória digitálneho záznamu zvuku pomocou vzorkovania, kvantovania a analógovo-digitálneho prevodu, návrh hardwérového zapojenia zariadenia, ktoré obsahuje návrh schémy zapojenia a návrh dosky plošných spojov, ako aj popis programu, ktorý je v mikrokontrolére záznamníka uložený a podľa ktorého záznamník pracuje.

I. TEORETICKÁ ČASŤ

1 PREVOD ANALÓGOVEJ HODNOTY NA DIGITÁLNU A OPAČNE

Pre správnu funkciu nahrávania jednoduchého záznamníka hlasu je dôležité správne prevádzať vstupnú analógovú veličinu na digitálne hodnoty a ukladať ich do pamäte. Pre správnu funkciu prehrávania nahraných hlasov jednoduchého záznamníka hlasov je dôležité uložené hodnoty v správnom poradí a správnom čase načítavať z pamäte a pomocou pulzne šírkovej modulácie tieto dáta prevádzať späť na výstupnú analógovú veličinu. Tento výstup pulzne šírkovej modulácie je potrebné previesť cez RC filtry, ktoré odfiltrujú vysokofrekvenčnú zložku PWM. Takto upravený signál sa zosilňuje a privádza na slúchadlá.

1.1 Záznam analogovej veličiny na digitálne médium

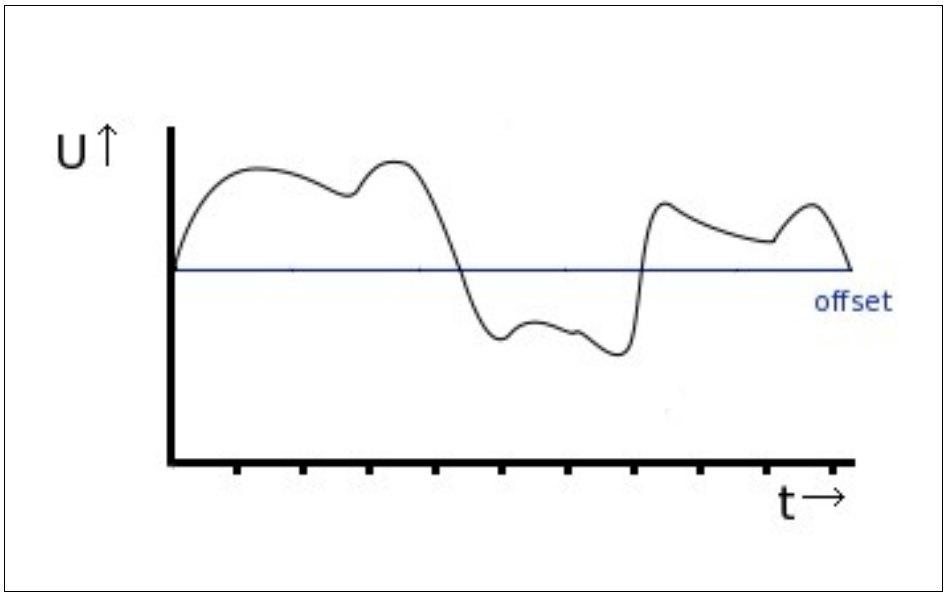
Ako bolo spomenuté v úvode tejto kapitoly, pre správnu funkciu nahrávania jednoduchého digitálneho záznamníka je potrebné správne prevádzať vstupný analógový signál na digitálny. Tejto časti spracovania analógového signálu sú venované nasledujúce kapitoly.

1.1.1 Vzorkovanie

Každý zvuk, ktorý je zaznamenávaný mikrofónom, má svoj priebeh, ktorý je charakterizovaný závislosťou napätia vychádzajúceho z mikrofónového obvodu na čase. Tento priebeh býva periodický a prenáša sa ním frekvencia a hlasitosť daného tónu.

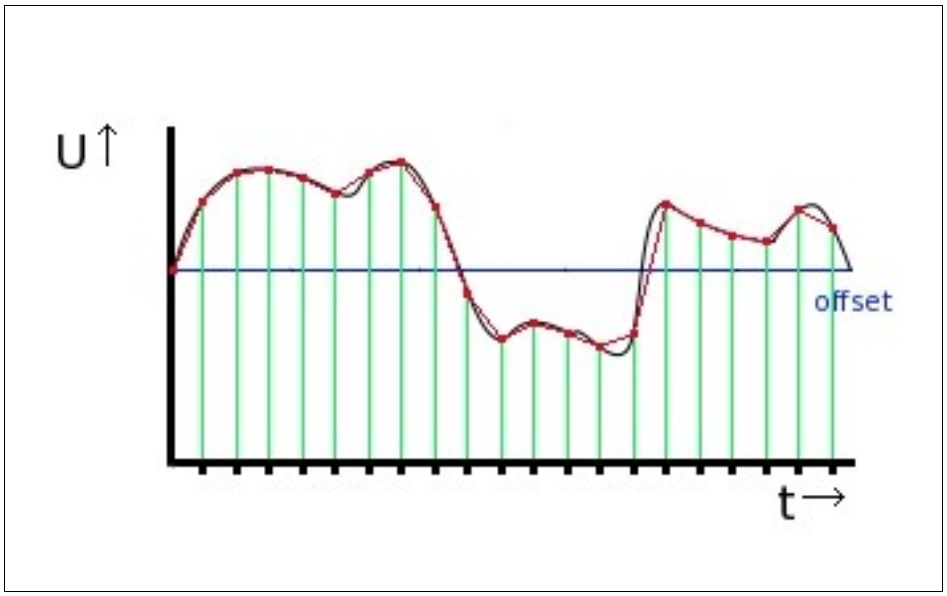
Podľa [1] sa zvuk vyznačuje svojou fyzikálnou intenzitou hladinou zvuku meranou v dB a fyziologickou hladinou svojej hlasitosti, čo je vlastne amplitúda. Mimo to sa hudobné zvuky vyznačujú frekvenciou, ktorá určuje ich výšku. Treťou základnou vlastnosťou zvuku je priebeh kmitania, ovplyvňujúci jeho zafarbenie. Trvanie zvuku v čase určuje jeho dĺžku.

Na nasledujúcom obrázku (Obr. 1) je zobrazený príklad priebehu signálu na vstupe AD prevodníka, ktorý vystupuje z mikrofónového predzosilňovača. Tento predzosilňovač si drží pri nulovom vstupnom napätí offset napätie na výstupe. Vstupný signál je potom zosilňovaný a jeho hodnoty sa pohybujú okolo offsetu. Realizácia s offsetom sa používa kvôli záporným polperiódam signálu, aby sa na vstupe AD prevodníka neobjavilo záporné napätie.



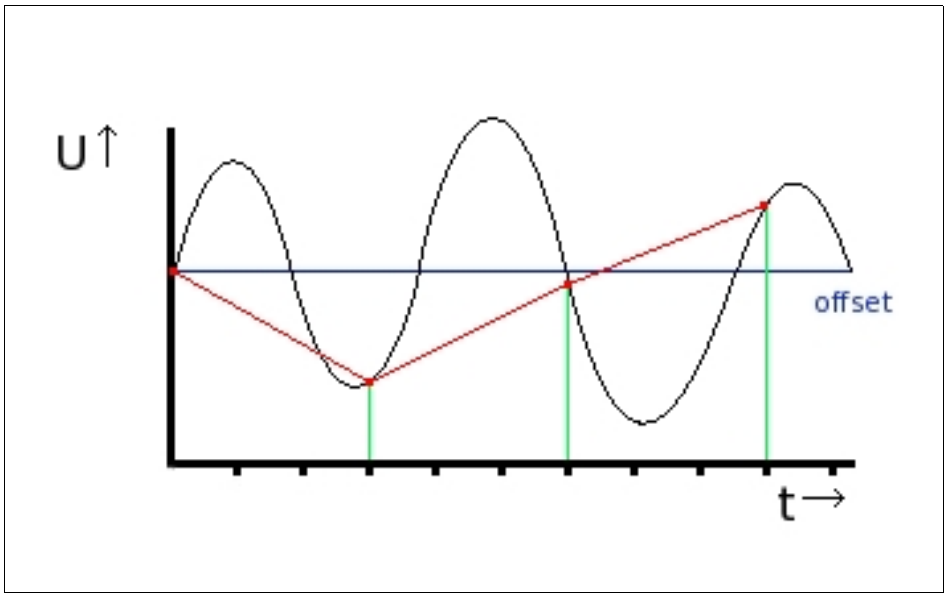
Obr. 1 Grafické znázornenie zvukového priebehu na vstupe AD prevodníka

Časový priebeh zvuku je potrebné nejakým spôsobom zaznamenávať a ukladať do pamäte. Keďže analógový priebeh zvuku je spojitý a spracovávanie dát pomocou mikropočítača je diskrétne, musíme vstupný signál zdiskretizovať. Diskretizáciu signálu prevádzame pomocou vzorkovania a kvantovania. Priebeh analógovej vstupnej veličiny sa pravidelne sníma a zaznamenáva v rovnakých časových intervaloch. Vzorkovaciu frekvenciu je možné vypočítať vďaka Shannon-Kotelnikovej teoréme. Táto teoréma určuje, že frekvencia vzorkovania musí byť minimálne dvakrát väčšia, ako je najvyššia frekvencia vstupného signálu. Pokiaľ sa nedodrží táto podmienka, môže pri vzorkovaní dochádzať k takzvanému aliasingu, ktorý má za následok, že zaznamenaný priebeh signálu je odlišný od snímaného. Na nasledujúcich obrázkoch je ukážka správne navzorkovaného signálu (Obr. 2) a priebeh vzorkovania v prípade aliasingu (Obr 3).



Obr.2 Vzorkovanie vstupného priebehu AD prevodníka

Na obrázku (Obr. 2) sú zelenou čiarou zaznačené vzorkovacie impulzy, ktoré zaznamenávajú hodnoty označené červenou bodkou. Tieto hodnoty sú pospájané červenou spojnicou, ktorá približne kopíruje vstupný priebeh AD prevodníka. Takto vzorkovaný signál je vzorkovaný správne.



Obr. 3 Ukážka nesprávne zvolenej vzorkovacej frekvencie

Na obrázku (Obr. 3) je načrtnutý preibeh vzorkovania pri menšej vzorkovacej frekvencii, než je udávaná Shannon-Kotelnikovou teorémou. Vzorkovacia frekvencia nie je postačujúca na zaznamenanie priebehu vstupného signálu, preto dochádza ku značnému skresleniu a nepoužiteľnosti získaného digtálneho záznamu zvuku.

V praktickej aplikácii popisovanej v praktickej časti tejto práce je počítané so vzorkovacou frekvenciou 15,625kHz. Frekvencia ľudského hlasu sa pohybuje okolo 3kHz a dolnopriepustné výstupné filtre sú nastavované na maximálnu prehrávanú frekvenciu 4kHz. Keďže vzorkovacia frekvencia, s ktorou sa pracuje, je štvornásobne väčšia ako maximálna snímaná frekvencia, je splnená podmienka pre správny priebeh vzorkovania podľa Shannon-Kotelnikovej teorémy. Nastavenie tejto frekvencie na časovači, ktorý ju udržuje, je podľa nasledujúceho vzorca:

$$frekvencia\,časovača = \frac{frekvencia\,oscilátora}{prescaler\,.\,(1+OCR)} = \frac{8000000}{128\,.\,(1+3)} = 15{,}625\,kHz \qquad (3)$$

Frekvencia oscilátora v tomto vzorci udáva taktovaciu frekvenciu mikrokontroléra, ktorá je v našom prípade 8MHz. Prescaler je delička tejto frekvencie a výstup z nej je pripojený k časovaču. Aby vyšla vzorkovacia frekvencia podľa plánovanej hodnoty, prescaler je nastavený na hodnotu 128 (čítač má taktovaciu frekvenciu 62,5 kHz). Premenná OCR určuje najvyššiu hodnotu čítača. Táto premenná je tiež zvolená kvôli plánovanej frekvencii vzorkovania na hodnotu 3.

1.1.2 Prevedenie analógovej hodnoty na digitálnu

Hodnoty napätia, ktoré sú zaznamenané pri vzorkovaní, treba nejakým spôsobom previesť na digitálne hodnoty. Táto časť procesu záznamu sa nazýva kvantovanie. Vstupný interval analógových hodnôt (0 až 2.56 V) sa rozdelí na intervaly, ktorým sa prideľuje rovnaká digitálna hodnota. Pre túto činnosť využijeme analógovo-digitálny prevodník.

Analógovo-digitálny prevodník je obvod, ktorý využíva porovnávanie vstupného napätia s referenčným napätím, ktoré má k sebe pripojené. Toto referenčné napätie je brané ako maximálna hodnota na vstupe, prislúcha jej maximálna číslicová hodnota, ktorou prevodník disponuje.

Citlivosť prevodníka je daná veľkosťou napätia, ktoré zodpovedá zmene výstupného čísla o 1, a je závislá na počte jeho výstupných bitov (počte napäťových intervalov, na ktoré sa daný interval rozdeluje), ako aj na veľkosti referenčného napätia. Citlivosť prevodu je možné vypočítať pomocou tohto vzorca:

$$citlivosť = \frac{U_{ref}}{rozlíšenie\ prevodníka}\left[\frac{V}{interval}\right] \qquad (4)$$

V praktickej aplikácii popisovanej v praktickej časti tejto práce je aktivovaný kontinuálny režim AD prevodníka, ktorý prevádza dáta neustále, a v časoch vzorkovania sa odčítava výsledok prevodu. Prevodník je 8-bitový, pracuje na frekvencii 125kHz a je použité interné referenčné napätie 2.56V. Jeho citlivosť môžeme vypočítať z uvádzaného vzorca:

$$citlivosť = \frac{2,56}{255} = 0,01\left[\frac{V}{interval}\right] \quad (5)$$

Binárne hodnoty získané pomocou AD prevodníka sa zapisujú do pamäte pomocou riadiaceho mikrokontroléra, ktorý ovláda všetky operácie s týmito hodnotami.

Mikrokontrolér je mikročip vybavený vnútornou pamäťou, vstupno-výstupnými portami a ostatnými integrovanými hardwérovými súčasťami, ako je napríklad aj spomínaný AD prevodník, ktoré robia použitie mikrokontroléra efektívnejším a jednoduchším. V prílohách (príloha IV) je načrtnutý blokový nákres použitého AVR mikrokontroléra. Keďže mikrokontroléry väčšinou nemajú veľkú vnútornú pamäť, do ktorej by sa dalo permanentne zapisovať kvantum navzorkovaných hodnôt, používajú sa externé pamäťové médiá, ako napríklad flash pamäte.

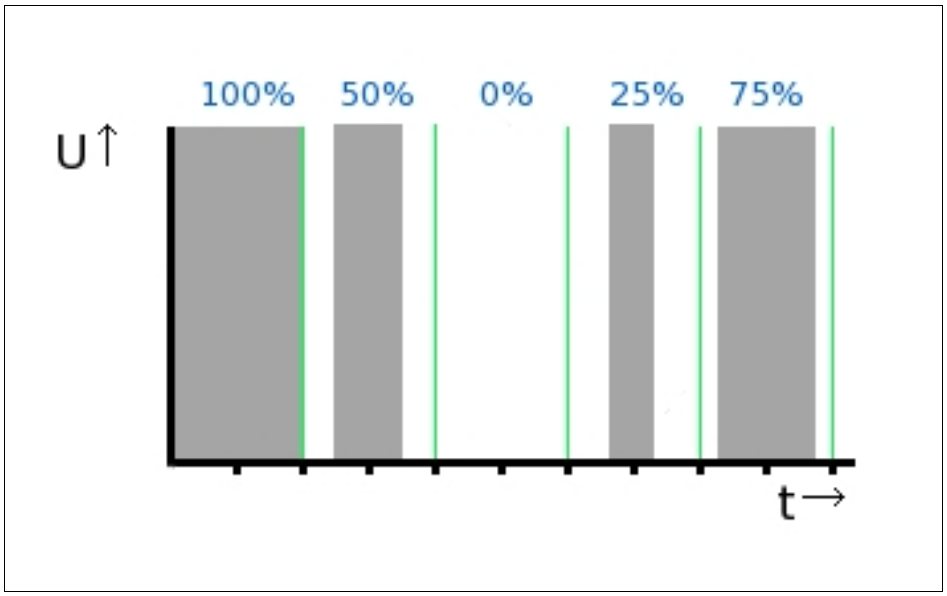
Flash pamäť je mikročip vybavený veľkou elektronicky prrepisovateľnou a mazateľnou flash pamäťou, menšími bufferami, komunikačným rozhraním pre zapisovanie do tejto pamäte a čítanie z nej. Pamäť býva pre jednoduchosť adresácie a prístupu rozdelená na sektory, bloky a stránky. Prehľad rozdelenia sektorov, blokov a stránok je uvedený v praktickej časti práce, v kapitole, ktorá popisuje jednotlivé použité obvody.

1.2 Prevod digitálnej veličiny na analógovú

Digitálne zaznamenané vzorky získané zo vstupného rozhrania záznamníka treba previesť na spojitú analógovú veličinu na výstupnom rozhraní záznamníka. Možnosti tejto konverzie sú dve: použitie D/A prevodníka alebo použitie PWM modulácie. Keďže obvod pre PWM moduláciu je súčasťou použitého mikrokontroléra, je jednoduchšie a praktickejšie zvoliť túto cestu generovania výstupného signálu.

1.2.1 PWM modulácia

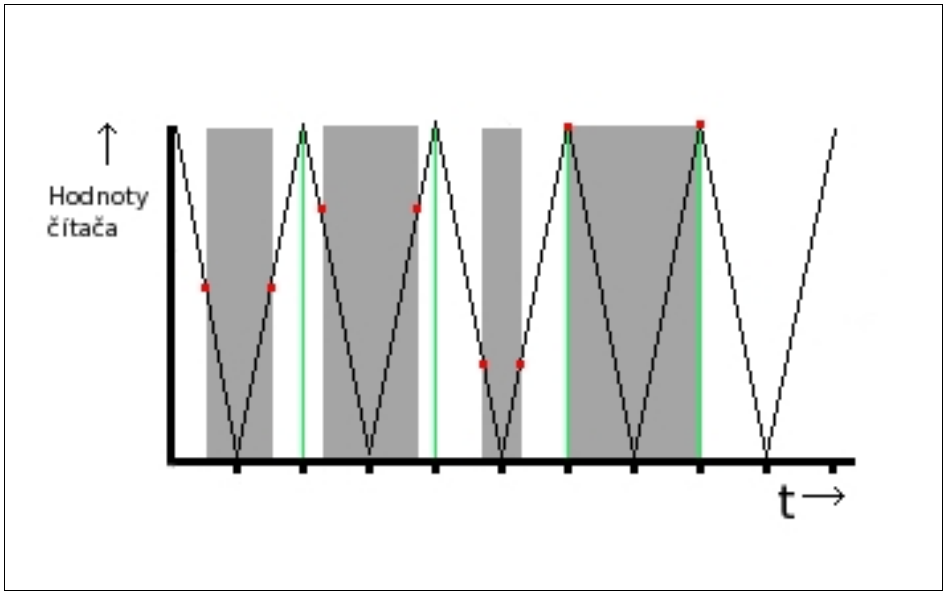
Princíp tohto generovania signálu spočíva v periodickom menení výstupnej hodnoty signálu v závislosti od nastavenej striedy. Nastavená strieda ovplyvňuje, akú časť periódy PWM bude nastavený výstupný signál na hodnotu logickej jednotky. V krajných prípadoch, keď je strieda nastavená na hodnotu 0% alebo 100%, je výstup buď trvale v hodnote logickej nuly, alebo v hodnote logickej jednotky.



Obr. 4 Grafické znázornenie princípu PWM.

Na obrázku (Obr. 4) vidíme príklad, aké priebehy na výstupe dostaneme, ak v jednotlivých periódach PWM budú nastavené hodnoty striedy 100%, 50% , 0%, 25% a 75% a bude zvolená fázovo správna neinvertujúca PWM. Začiatok a koniec periódy PWM je na obrázku vyznačený zelenou vertikálnou čiarou. Sivým pruhom je označená časť periódy, v ktorej je na výstupe PWM nastavená logická jednotka.

Principiálne tento druh PWM funguje tak, že sa spustí čítač, ktorý počíta od maximálnej hodnoty rozlíšenia PWM (v našom príklade je táto hodnota rovná 100) až po nulu a späť po maximálnu hodnotu. Ak sa hodnota striedy rovná aktuálnemu nastaveniu striedy pri počítaní smerom dole, na výstupe sa nastaví hodnota logickej jednotky. Ak sa hodnota počítadla PWM rovná hodnote striedy PWM pri počítaní smerom nahor, výstup PWM sa nastaví na hodnotu logickej nuly. V nasledujúcom obrázku (Obr. 5) je načrtnuté toto vyhodnocovanie počítadla pre prípady, že je hodnota striedy PWM postupne nastavovaná na hodnoty 50, 75, 25 a 100. Maximálna hodnota čítača PWM je v našom príklade 100. Sivými pruhmi je vyznačený časový úsek, v ktorom je na výstupe PWM logická jednotka.



Obr. 5 Grafické znázornenie priebehu PWM, nastavených hodnôt striedy, a výstupu PWM.

V aplikácii popisovanej v praktickej časti tejto práce je nastavená frekvencia PWM na rovnakú hodnotu ako vzorkovacia frekvencia, čiže 15,682 kHz. Toto nastavenie je vypočítané zo vzorca:

$$frekvencia\ PWM = \frac{frekcencia\ oscilátora}{2.\text{prescaler}.ICR} = \frac{8000000}{2.1.255} = 15{,}682\,[kHz] \qquad (6)$$

Prescaler je nastavený na hodnotu 1, čo znamená, že čítač počíta vo frekvencii oscilátora. ICR hodnota je najvyššia hodnota čítača, čím je aj určené, že PWM má rozlíšenie 8 bitov. Vysoká frekvencia PWM je z výstupného signálu odfiltrovaná výstupnými RC filtrami.

II. PRAKTICKÁ ČASŤ

2 HARDWÉROVÉ VYBAVENIE

V nasledujúcich kapitolách je popísaný návrh a funkcia hardwérového vybavenia záznamníka.

2.1 Čipy a integrované obvody

Pre vytvorenie hardwérového modelu sú pri vypracovaní práce použité nasledujúce integrované obvody a čipy:

- ATMEGA8L
- AT45DB321D-SU
- LM324N

2.1.1 Mikrokontrolér ATMEGA8L

Atmega8L je osembitový mikrokontrolér od spoločnosti Atmel, postavený na architektúre AVR RISC. Má 8-kilobytovú pamäť programu s podporou bootloadera a Read-While-Write self – programovania. Má zabudovaný nastaviteľný interný oscilátor, ktorý je možné použiť namiesto externého zdroja hodinových impulzov. Pre vstupné/výstupné operácie je vybavený troma osembitovými I/O portami. Obsahuje tiež 10-bitový AD prevodník, analógový komparátor, SPI rozhranie, dvojlinkové sériové rozhranie a 3 čítače/časovače (dva s možnosťou nastavenia PWM modulácie). Zvolil som ho práve vďaka jeho dosť širokým možnostiam, nízkej spotrebe, schopnosti pracovať aj s napájacím napätím 3.3 V, dostupnosti a možnosti programovania v praktickom vývojovom prostredí AVR Studio, ktoré je poskytované výrobcom. V nasledujúcich odsekoch popíšem funkciu niektorých častí tohto mikrokontroléra, ktoré využívam vo svojej aplikácii.

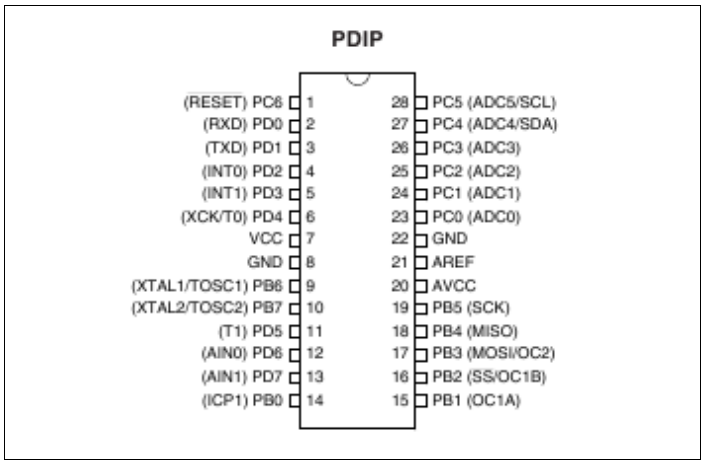
Vstupné/výstupné porty uložené v mikrokontroléri majú označenie: PORTB, PORTC a PORTD. Pomocou nich je možné zaznamenávať vonkajšie informácie, nastavovať výstupné hodnoty, ako aj komunikovať s ostatnými zariadeniami. Niektoré vstupno/výstupné piny týchto portov majú okrem I/O komunikácie na portoch možnosť využitia iných súčastí mikrokontroléra, ako napríklad AD prevodníka a SPI rozhrania. Preto je potrebné pri inicializácii zariadenia správne nastaviť ich funkciu.

AD prevodník, ktorý sa nachádza v mikrokontroléri, umožňuje prevod vstupnej analógovej veličiny na 10-bitové číslo. Doba prevodu analógovej veličiny na digitálne číslo sa pohybuje v rozmedzí 13 až 260 mikrosekúnd, v závislosti od zvolenej preddeličky frekvencie a požadovaného rozlíšenia. Umožňuje prevod až zo šiestich vstupov prepínaných pomocou multiplexora. Referenčné napätie môže byť v rozsahu 0V až Vcc, možné je však aj prepnutie na internú referenciu 2.56 V. Je potrebné ho napájať zvlášť, privedením Vcc na pin Avcc a pripojením GND k pinu AGND. Vo vypnutom stave (sleep mode) nemá takmer žiadnu spotrebu energie.

SPI rozhranie použité v mikrokontroléri je plne duplexné a podporuje trojcestné synchrónne prenosy dát. Je možné nastaviť 7 prenosových rýchlostí. Používa sa na komunikáciu medzi čipmi, avšak pripojením ISP programátora na SPI rozhranie sa dá mikrokontrolér programovať.

Mikrokontrolér obsahuje 3 čítače/časovače, označené Timer0, Timer1 a Timer2. Timer0 a Timer2 sú osembitové, Timer1 je šestnásťbitový. Timer1 a Timer2 disponujú Normal, CTC, Fast PWM a Phase Correct PWM módmi. Timer1 umožňuje použiť navyše režim Phase and Frequency Correct PWM. K všetkým trom čítačom/časovačom je možné pripojiť preddeličku, a tak kontrolovať ich rýchlosť.

Pre vytvorenie lepšej predstavy o práci s týmto mikrokontrolérom a s jeho konštrukciou uvádzam nákres vzhľadu mikrokontroléra s popisom jednotlivých pinov:



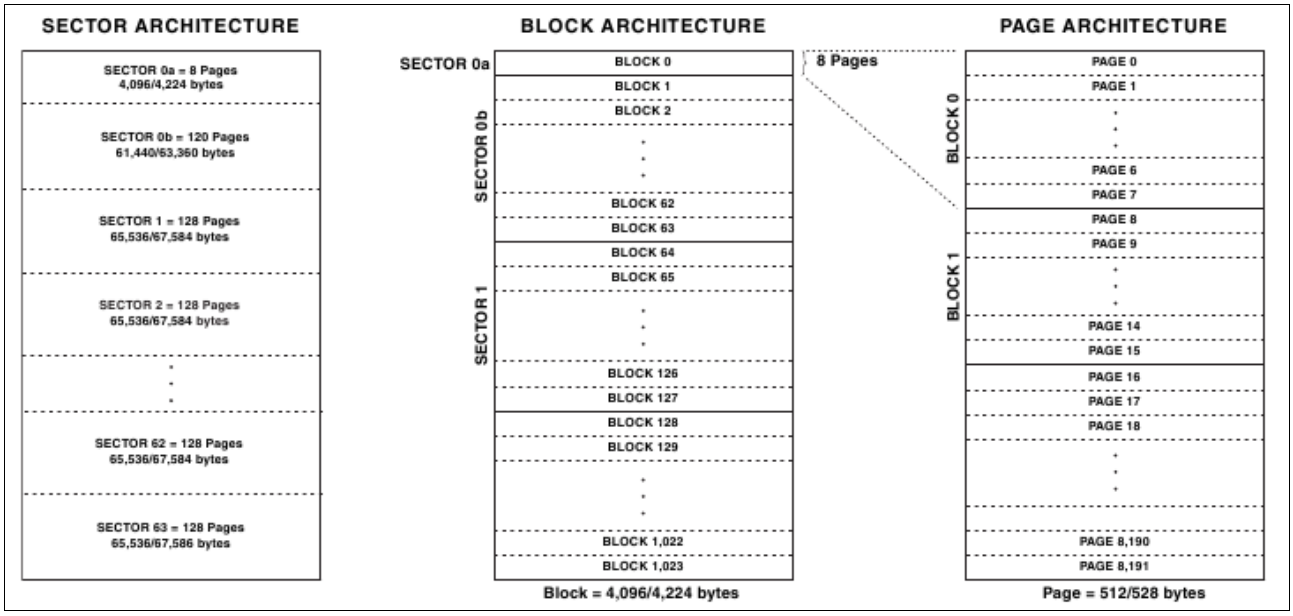
Obr. 6 Rozmiestnenie pinov mikrokontroléra a ich funkcie [2]

2.1.2 Flash pamäť AT45DB321D-SU

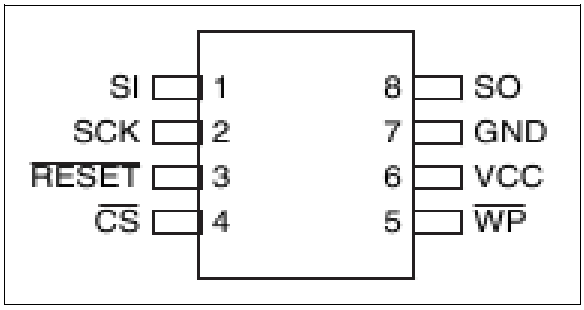
AT45DB321D-SU je flash pamäť vyrobená spoločnosťou Atmel, fungujúca pri napájacom napätí v rozmedzí 2.7 až 3,6V. Komunikácia prebieha cez SPI rozhranie. Maximálna taktovacia frekvencia SPI rozhrania je 66 MHz. Táto flash pamäť umožnuje nastavenie veľkosti stránok na 512 bytov, alebo 528 bytov. Do vlastnej pamäti je možnosť priamo vstupovať, alebo je možné operácie nad pamäťou vykonávať cez jeden z dvoch bufferov, ktoré sú v čipe implementované. Pomocou zápisu logickej nuly na pin WP je možné chrániť dáta pred zápisom či zmazaním. Energetická spotreba pamäte je závislá na režime, v ktorom sa pamäť nachádza, no pohybuje sa v rozmedzí 5 μA až 7mA.

Pamäť je rozdelená na stránky podľa nastavenej veľkosti. Stránky sú usporiadané v blokoch. Každý blok obsahuje 8 stránok. Bloky sú ďalej usporiadané do sektorov. Sektor 0a obsahuje len blok 0. Sektor 0b obsahuje blok 1 až blok 63. Ďalšie sektory obsahujú každý po 63 blokov. Počet sektorov v pamäti je 63.

Pre lepšiu predstavu o rozložení pamäte uvádzam obrázok s grafickým znázornením veľkosti a počtu stránok, sektorov a blokov a obrázok s grafickým zobrazením integrovaného obvodu (Obr. 7) a funkciami jednotlivých pinov(Obr. 8):



Obr. 7 Grafické zobrazenie rozloženia a architektúry pamäte AT45DB321D [3]

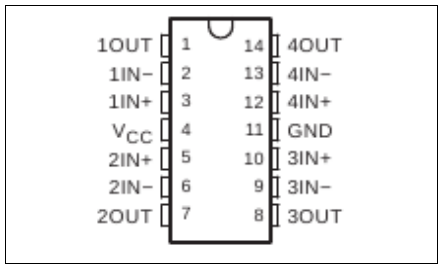


Obr.8 Rozmiestnenie pinov AT45DB321D, a ich funkcie [3]

2.1.3 Integrovaný obvod LM324N

LM324N je integrovaný obvod v obale DIL14, obsahujúci 4 frekvenčne kompenzované operačné zosilovače s vysokou vstupnou impedanciou. Tieto operačné zosilovače sú schopné pracovať v rozsahu napájacieho napätia 3 V až 32 V alebo pri symetrických zdrojoch od napätia ±1,5 V po ±16 V. Tieto operačné zosilňovače nie sú vyrábané priamo na produkciu a zosilňovanie zvuku a majú iba 3.3 V napájacie napätie, preto nie sú schopné dostatočne vybudiť reproduktor s nízkou impedanciou (8ohm). Táto skutočnosť však nie je prekážkou v konštrukcii, pretože náhlavné slúchadlá, na pripojenie ktorých je toto zapojenie konštruované, majú dosť vysokú impedanciu (približne 60 až 80 ohm), takže tento operačný zosilňovač je ich ešte schopný vybudiť.

Pre lepšiu predstavu o rozmiestnení jednotlivých pinov integrovaného obvodu a ich funkcií uvádzam obrázok (Obr. 9) tohto integrovaného obvodu aj s popisom funkcií jednotlivých pinov:



Obr. 9 Rozmiestnenie a funkcia pinov integrovaného obvodu LM324N [5]

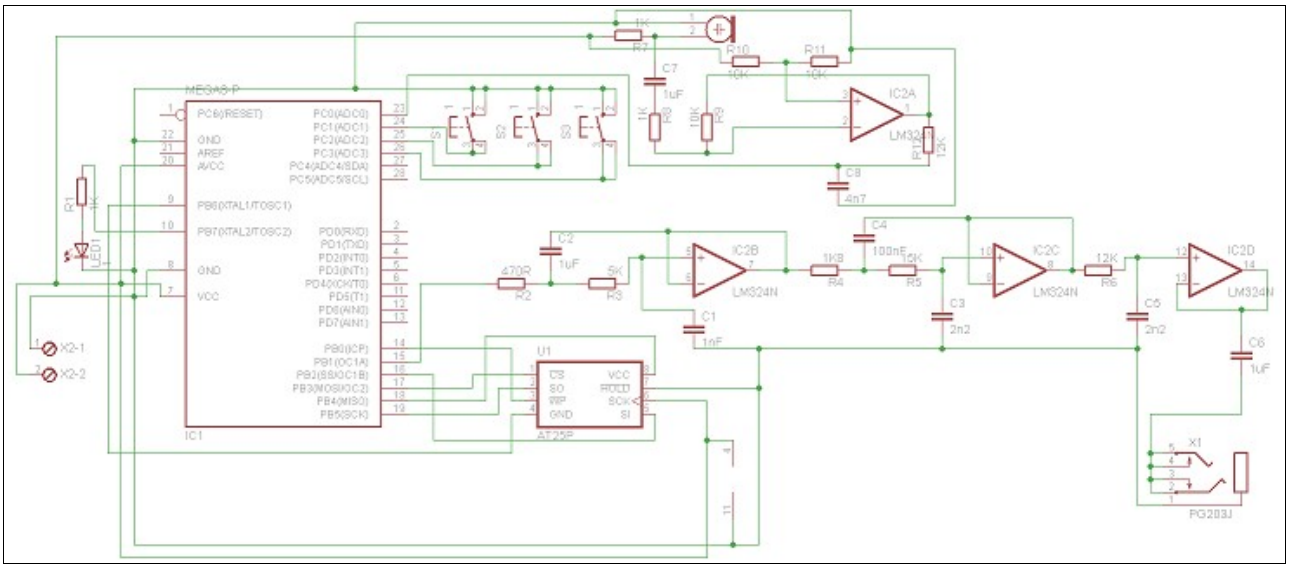
2.2 Schéma zapojenia a návrh plošného spoja

Táto podkapitola je venovaná návrhu schémy zapojenia, ako aj dosky plošných spojov.

2.2.1 Schéma zapojenia

Pri návrhu zapojenia záznamníka v freeware verzii programu Eagle 5.8.0 od spoločnosti Cadsoft je vychádzané zo schémy zapojenia a komentárov uvedených v aplication note od spoločnosti Atmel [4], v ktorej je použitý rovnaký integrovaný obvod s operačnými zosilňovačmi LM324N a ten slúži ako mikrofónový predzosilňovač a zároveň ako výstupný zosilňovač. Je vybavený dolnopriepustnými RC filtrami pre odstránenie jednosmernej zložky a vysokej frekvencie vystupujúcej z PWM, filtračnými kondenzátormi a elektretovou mikrofónovou vložkou. Mikročip Atmega8L pripojený na napájacie napätie 3.3 V je pomocou SPI zbernice pripojený k flash pamäti AT45DB321D-SU, tiež napájanej napájacím napätím 3.3 V. K mikročipu sú ešte pripojené 3 tlačidlá, určené na ovládanie záznamníka, a LED dióda určená na signalizovanie operácií záznamníka.

Na nasledujúcom obrázku (Obr. 10) je schéma zapojenia celého zariadenia navrhnutá v programe Eagle 5.8.0, určená pre vytvorenie lepšieho obrazu o konštrukcii zariadenia.



Obr. 10 Schéma zapojenia záznamníka

Zväčšená schéma celého zapojenia záznamníka je uložená v prílohách (príloha I). Detailnejši popis jednotlivých častí obvodu a ich funkcii je v uvedený v nasledujúcich kapitolách.

2.2.2 Mikrofónový predzosilňovač

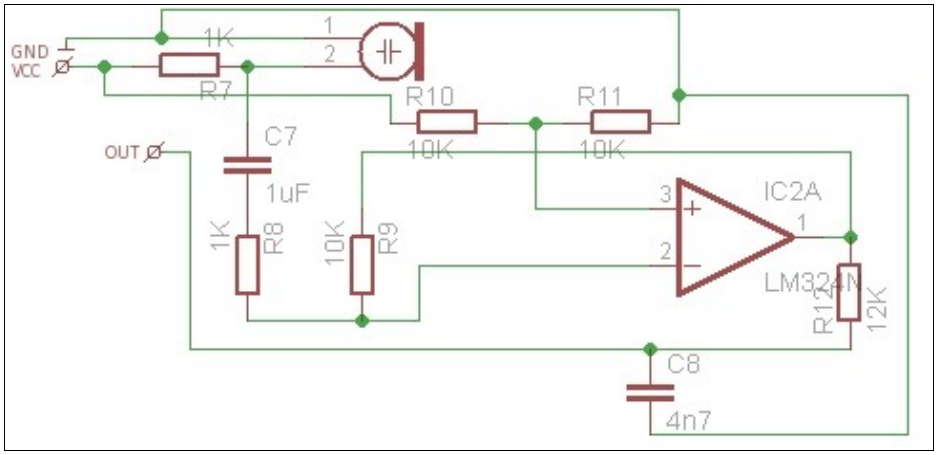
Obvod mikrofónového predzosilňovača je realizovaný pomocou operačného zosilňovača z integrovaného obvodu LM324N napájaného z VCC 3.3 V. Je to zapojenie jednoduchého invertujúceho zosilňovača. Jeho zosilnenie je určené pomerom rezistorov R8 a R9. Dá sa určiť pomocou jednoduchého vzorca:

$$zosilnenie = \frac{R9}{R8} = \frac{10000}{1000} = 10 \tag{1}$$

Elektretový mikrofón pre funkciu potrebuje napájacie napätie. Toto je k nemu pripojené cez rezistor R7. Aby do zosilňovača nevstupovala jednosmerná zložka, je použitý kondenzátor C7 s kapacitou 1μF, ktorý ju odfiltruje. Pomocou deličky napätia zloženého z rezistorov R10 a R11 je nastavený offset zosilovača.

$$offset = \frac{Vcc}{R10 + R11} . R10 = \frac{3.3}{12700} .2700 = 0.70157\text{V} \tag{2}$$

Rezistor R12 a kondenzátor C8 vytvárajú na výstupe operačného zosilňovača dolnopriepustný RC filter. Rezistor R12 veľkosťou svojho odporu navyše chráni zosilňovač pred zničením pri zapojení nakrátko.

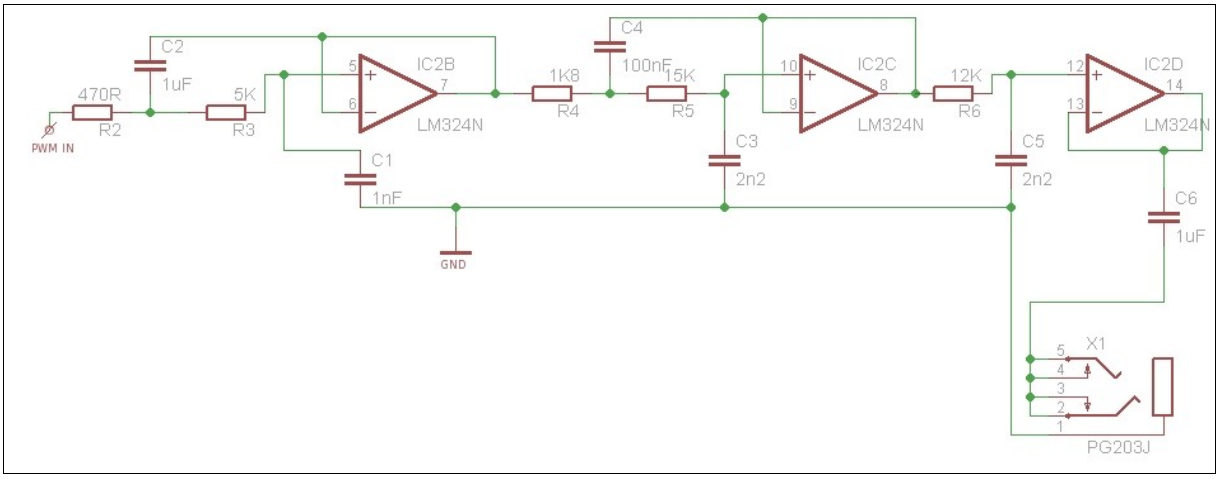


Obr. 11 Schéma zapojenia mikrofónového predzosilňovača

2.2.3 Koncový zosilňovač pre výstup z PWM

Koncový zosilňovač je realizovaný pomocou zvyšných troch operačných zosilňovačov z integrovaného obvodu LM324N. Jednotlivé operačné zosilňovače sú zapojené za sebou. Medzi jednotlivými zosilňovačmi sú pripojené dva Chebyschevove RC filtre druhej úrovne (R2, R3, R4, C1, C2 a R4, R5, R6, C3, C4) a jeden pasívny RC filter prvej úrovne (R6, C5). Filtre sú navrhnuté tak, že sú ich frekvenčné charakteristiky oproti sebe mierne posunuté, aby bol pokrytý celý limit filtrovaného a prepúštaného frekvenčného pásma. Výsledná maximálna prepúšťaná frekvencia takto spojených filtrov je 4 kHz, čo predstavuje asi jednu štvrtinu PWM frekvencie 15,682 kHz. Kondenzátor C6 má na výstupe zosilňovača funkciu filtra, ktorý blokuje a neprepúšťa do slúchadiel žiadnu jednosmernú zložku. K výstupu zo zosilňovača je pripojený konektor s 3,5mm Jack koncovkou, pre pohodlné a prenosné pripojenie akýchkoľvek slúchadiel s koncovkou tohto typu.

Schému zapojenia samotného koncového zosilňovača (Obr. 12) uvádzam pre lepšiu predstavu návrhu schémy a pre orientáciu v menovaných súčiastkach:



Obr. 12 Schéma koncového zosilňovača PWM pre slúchadlá.

2.2.4 Zapojenie mikrokontroléra a flash pamäte

Mikrokontrolér Atmega8L je pomocou pinov 7 a 8 pripojený na napájacie napätie. Vcc je pripojené k pinu 7 a GND k pinu 8. Okrem hlavného napájania mikrokontroléra ešte pripájam na napätie Vcc pin 20 (AVCC) a GND na pin 22. Slúžia na napájanie AD prevodníka. Napájacie napätie 3.3 V je tiež privedené na pin 6 dataflash pamäte AT45DB321D a GND je na tomto čipe privádzané na pin 7.

LED dióda použitá na signalizovanie stavu diktafónu, a tiež jeho funkcie je z pinu 10 pripojená cez rezistor R1 oproti zemi. Dióda svieti, pokiaľ je pin 10 nastavený ako výstupný a je na ňom nastavená logická jednotka. Na pine sa vtedy objaví napätie 3.3 V. Aby prúd pretekajúci výstupným pinom nebol príliš veľký (maximálny prúd pretekajúci I/O pinom je 40 mA), je použitý už spomínaný rezistor R1, na ktorom nastáva väčší úbytok napätia a tým cez diódu z mikrokontroléra tečie menší prúd.

Tlačidlá sú jedným kontaktom privedené na vstupné piny, ktorými je ovládaný mikrokontrolér, a druhým kontaktom sú všetky spojené, a privedené na GND. V mikrokontroléri sú na týchto vstupných pinoch nastavené pull-up rezistory, ktoré na pine bez jeho uzemnenia držia napájacie napätie. Hneď ako sa príslušný pin spojí cez tlačidlo s GND, na pine sa objaví hodnota logickej nuly. Pull-up rezistory využívam z dôvodu rôznych indukcií napätí na rozpojených vstupných kontaktoch, ktoré, keď nie sú pripojené (tlačidlo nie je stlačené), slúžia ako anténa, na ktorej sa indukujú napätia vyvolané okolitým elektrickým poľom. Ak by sa v takom prípade testovala úroveň logickej jednotky na vstupe, dochádzalo by k náhodnému spínaniu funkcií záznamníka. Zákmity na tlačidlách vo svojom zapojení neriešim, lebo program v mikrokontroléri je navrhnutý tak, že nie je schopný reagovať na tieto drobné kmity na vstupe.

Pamäť AT45DB321D je s mikrokontrolérom Atmega8L prepojená pomocou zbernice SPI. Je to zbernica určená na prepojenie takýchto zariadení, ako sú mikroradiče a flash pamäte. Táto zbernica je zložená z troch vodičov: MOSI, MISO, SCK. Pre komunikáciu po tejto zbernici sa však využívajú ešte pomocné signály medzi mikroradičom a pamäťou, ako: WP, CS, RST.

MOSI – je skratka pre anglické slovné spojenie Master Output Slave Input. Ako je už z jej názvu zrejmé, slúži na prepojenie výstupného pinu SPI zbernice na master zariadení s vstupným pinom SPI zbernice na slave zariadení. V našom prípade je to pin 17 (MOSI) na mikrokontroléri s pinom 1 (SI) na flash pamäti.

MISO – je skratka pre anglický názov Master Input Slave Output. Slúži na prepojenie vstupného pinu SPI zbernice na master zariadení s výstupným pinom SPI zbernice na slave zriadení. V našom prípade je to prepojenie pinu 18 (MISO) na mikrokontroléri s pinom 8 (SO) na flash pamäti.

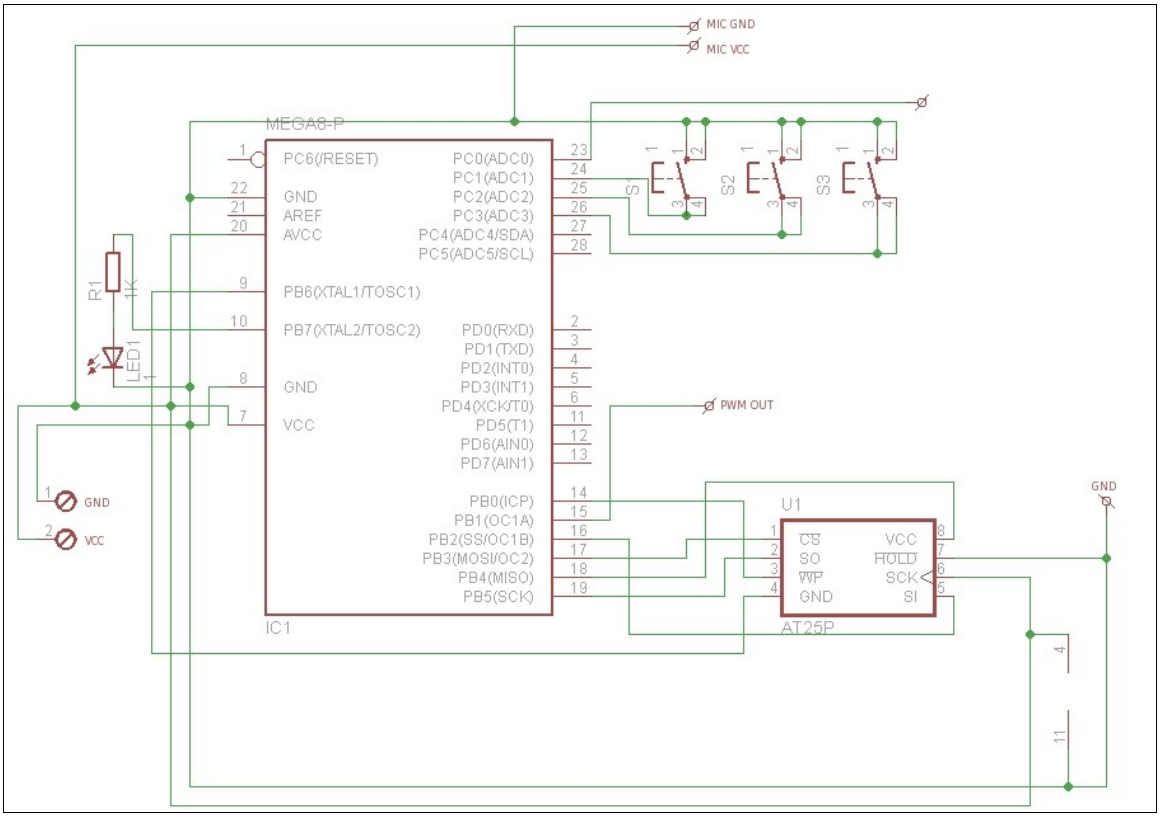
SCK – je kanál na zbernici SPI, cez ktorý master zariadenie vysiela taktovaciu frekvenciu zbernice, a slave zariadenie túto frekvenciu na svojom SCK vstupe prijíma. V našom prípade je to pin 19 (SCK) na mikrokontroléri Atmega8L a pin 2 (SCK) na flash pamäti AT45DB321D.

CS – skratka z anglického názvu Chip Select. Pomocou tohoto signálu si vyberá master zariadenie druhé zariadenie slave, s ktorým chce komunikovať. Na flash pamäti je vstupný pin 4 (CS) prepojený s pinom 9 (PB6), ktorý som nastavil ako výstupný pin na tento účel.

WP – skratka z anglického názvu Write Protect. Tento pin sa používa na ochranu dát pred zápisom. Vo svojom zapojení som ako výstupný pin pre túto funkciu na mikrokontroléri zvolil pin 16 (PB2), ktorý je nastavený ako výstupný, a je prepojený s pinom 5 (WP) na flash pamäti.

RST – je vstupný pin flash pamäte určený na resetovanie čipu. Pre ovládanie tejto funkcie som zvolil na mikrokontroléri pin 14 (PB0), ktorý je nastavený ako výstupný pin, a prepojený so zodpovedajúcim pinom (pin 3 - RST) na flash pamäti.

Pre lepšiu predstavu zapojenia tejto časti obvodu tu uvádzam zväčšenú časť schémy zapojenia jednoduchého záznamníku zvuku (Obr. 13), ktorá obsahuje spomínané zapojenie.

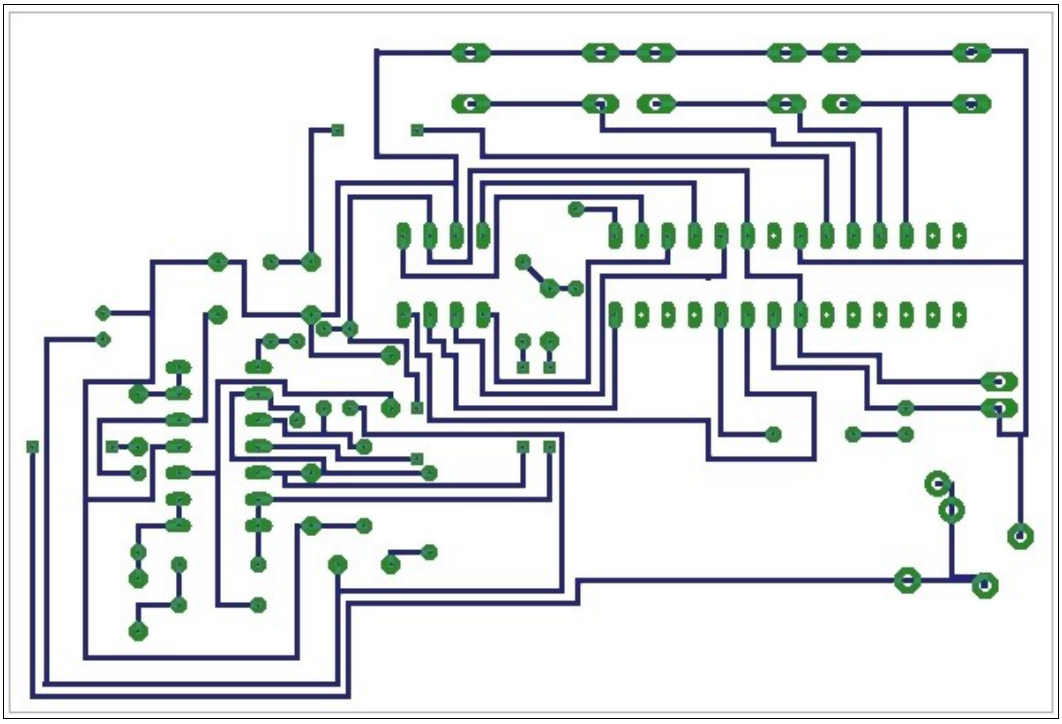


Obr. 13 Schéma pripojenia flash pamäti, tlačidiel a LED diódy ku mikropočítaču.

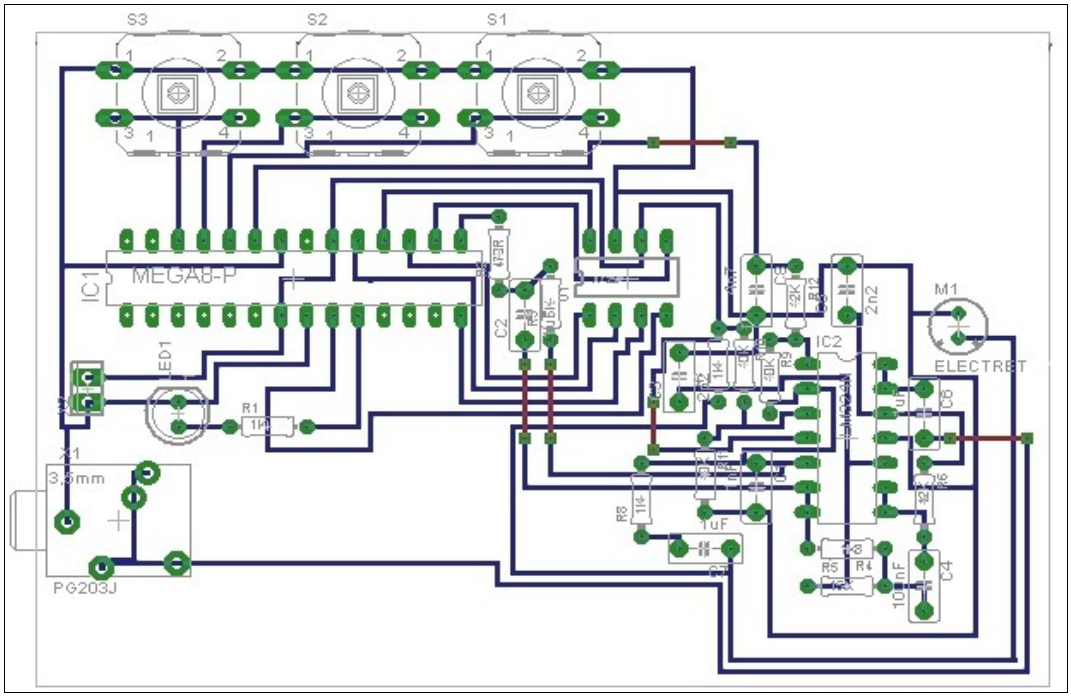
2.2.5 Návrh plošného spoja

Plošný spoj je navrhnutý priamo zo schémy zapojenia v programe Eagle 5.8.0 od spoločnosti Cadsoft. Keďže bola použitá bezplatná verziu programu, určená pre vzdelávanie, bola možnosť navrhovať plošný spoj len do maximálneho rozmeru 100 mm krát 80mm . Toto však veľmi veľká prekážka nebola, pretože doska plošných spojov pre záznamník nebola plánovaná vo väčších rozmeroch, než aké boli obmedzené vo free verzii programu. Výsledná DPS je široká 100 mm a vysoká 670 mm. Pre jednoduchosť konštrukcie je navrhnutá jednovrstvová doska plošných spojov. Síce na pár miestach dosky sú cesty hustejšie vedľa seba, čo robí trocha náročnejšie správne vyleptanie dosky plošného spoja, odpadá však problém s obojstranným spájkovaním na plošný spoj, prekovmi a podobne. V celom návrhu bolo potrebných len 5 drôtových premostení.

Na nasledujúcich obrázkoch je zobrazená doska plošného spoja zo spodnej strany (zrkadlovo otočená) (Obr. 14) a pohľad na osadenú dosku plošných spojov zhora (Obr. 15). Zobrazená je aj bottom vrstva pre lepšiu predstavu orientácie DPS .



Obr.14 Pohľad na dosku plošných spojov zospodu

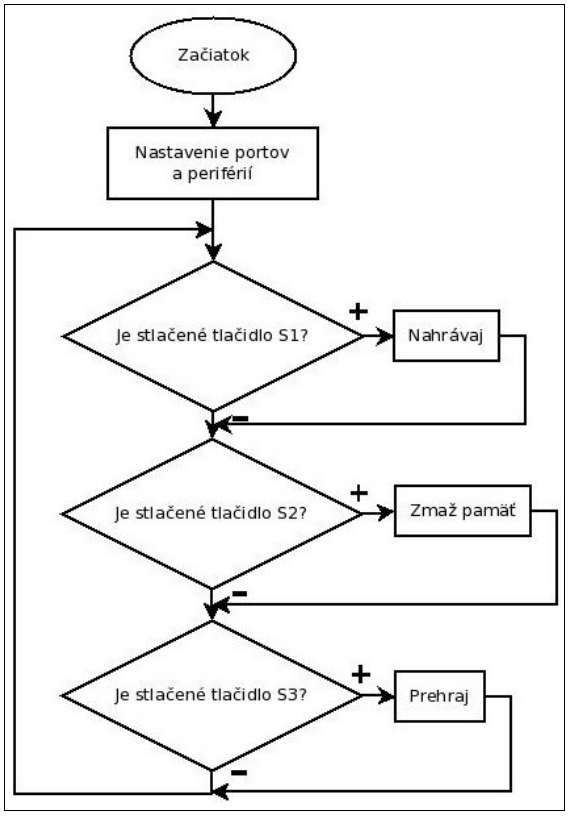


Obr.15 Zobrazenie dosky plošných spojov pri pohľade zhora.

3 PROGRAMOVÉ VYBAVENIE ZARIADENIA

V pamäti programu mikrokontroléra je nahratý program, ktorý riadi hardvér, ako je popísané v ďalších podkapitolách. Program po spustení mikrokontroléra nastaví potrebné porty a zariadenia a pokračuje tak, že v slučke stále testuje stlačenie tlačidiel. Pokým je stlačené tlačidlo S3, spustí sa prehrávanie už nahratých dát od začiatku a prehráva sa, pokým nie sú prehrané všetky údaje z dataflash (aj keď je prázdna, prehráva sa ticho), alebo pokým užívateľ nepustí tlačidlo S3. Pri stlačení tlačidla S2 sa premaže celá pamäť s nahranými dátami. Stlačením tlačidla S1 užívateľ spustí nahrávanie, čiže záznam zvukov zachytených mikrofónom. Nahrávanie sa skončí, keď je plná pamäť dát alebo ak užívateľ pustí tlačidlo S1. Pri opakovanom spustení nahrávania sa nahrané dáta pridávajú za už zaznamenané dáta nahrané v predchádzajúcom zázname. Svoju zaneprázdnenosť alebo činnosť signalizuje mikrokontrolér rozosvietením LED diódy.

Na nasledujúcom obrázku (Obr. 16) je vývojový diagram so zjednodušeným zobrazením funkcie programu:



Obr. 16 Jednoduchý vývojový diagram programu.

3.1 Rozdelenie častí kódu podľa funkcie

Aby program nahraný v mikrokontroléri nebol veľký, zložitý a neprehľadný, je rozdelený na jednotlivé funkcie. Tieto funkcie sú volané buď z hlavného programu alebo z nadradených funkcií. Tu je zoznam jednotlivých funkcií programu, ktoré budú v ďalších kapitolách podrobnejšie vysvetlené:

- setup
- zapis_cez_SPI
- cakajnaready
- zapis_na_pamet
- napln
- zmaz_flash
- druhastranka_druhybuffer
- aktivnybuffer_prehraj
- hraj
- prehraj
- main

3.1.1 Setup

Táto funkcia má za úlohu nastaviť porty , registre, čítače, časovače a AD prevodník na hodnoty a funkcie tak, aby ich bolo možné v ostatných častiach programu využívať podľa potreby. Funkcia začína nastavením pinov na porte B podľa funkcie, na ktorú sa budú používať, buď na vstupné, alebo na výstupné. Pokračujeme nastavením registrov SPCR a SPSR, pomocou ktorých sa nastavuje SPI rozhranie. SPI rozhranie týmto nastavením spúšťame, nastavujeme mikrokontrolér ako master zariadenie a taktovaciu frekvenciu zbernice volíme na hodnotu fosc/2, čo znamená 4MHz.

Ďalej pokračujeme nastavením portu B a portu C. Na porte B aktivujeme pre všetky vstupné piny Pull-up rezistory a všetky výstupné piny nastavíme na hodnotu logickej jednotky. Pretože máme na výstupe portu B pripojenú aj LED diódu signalizujúcu činnosť záznamníku, násobíme port B maskou, ktorá zmení logickú hodnotu na pine LED diódy na hodnotu logickej nuly. Na porte C dalšími dvoma príkazmi nastavíme všetky piny ako vstupné a všetkým vstupným pinom na tomto porte aktivujeme Pull-up rezistory.

Ako ďalšiu súčasť mikrokontroléra nastavujeme pulzne šírkovú moduláciu (PWM). Pomocou nastavení príslušných hodnôt na porty TCCR1A, TCCR1B a ICR1, nastavíme PWM nasledovné vlastnosti: výstup PWM je posielaný neinvertovaným spôsobom na pin OC1A (PB1), Phase correct PWM režim, vrchol čítača PWM je nastavený na hodnotu 255 a taktovacia frekvencia PWM je nastavená na nulu – PWM je momentálne nastavená, ale zastavená.

Pokračujeme nastavením funkcie AD prevodníka. Pomocou registrov ADMUX a ADCSRA nastavíme AD prevodníku nasledujúce vlastnosti: interné referenčné napätie 2,56 V, zarovnanie výsledku konverzie doľava, ako vstup AD prevodníka slúži pin ADC0 (PC0), prevodník je spustený v režime free-running, prerušenie pri ukončení konverzie je vypnuté a prescaler taktovacej frekvencie AD prevodníka je zvolený na fosc/64, čo je rovné taktovacej frekvencii 125 kHz.

Ako posledná súčasť mikrokontroléra je vo funkcii setup nastavovaný časovač 2. Vhodným nastavením registrov TCCR2 a OCR2 nastavíme časovač do non PWM režimu v CTC móde. Frekvencia čítača je zvolená na hodnotu fosc/128 a vrchol časovača je nastavený na hodnotu 3. Keďže je frekvencia čítača nastavená, čítač už beží, ale negeneruje žiadne prerušenia ani nenastavuje príznaky. Na to musí byť v programe nastavený ešte register TIMSK.

Ako posledná inštrukcia funkcie setup je použitý príkaz, ktorý povolí globálne prerušenia.

Pre lepšiu predstavu a lepšie pochopenie uvádzam zapoznámkovaný zdrojový kód tejto funkcie:

```
void setup(void)
{
  DDRB = 0xEF;                    // SPI Port inicializácia
                                  // LED , CS,  SCK  MISO, MOSI, WP ,  PWM,   RST
                                  // PB7, PB6,  PB5,  PB4, PB3, PB2 ,  PB1,   PB0
                                  // O    O     O    I    O    O     O     O
                                  // 1    1     1    0    1    1     1     1
  SPCR = (1<<SPE)|(1<<MSTR)|(0<<SPR0)|(0<<SPR1);
  SPSR = (1<<SPI2X);                                          //master, fosc/2

 //nastavenie portov
```

```
    PORTB = 0xFF;                          // všetky výstupy 1, vstupy majú
                                           // zapnuté pull-up (LED kontrolka nesvieti)
    PORTB &= 0x7F;                         //vypni LED
    DDRC = 0x00;                           // port C má všetky piny vstupné
    PORTC = 0xFF;                          // vstupy pull-up

    //PWM nastavenia (clock vypnuté)
    TCCR1A = (1<<COM1A1)|(0<<COM1A0)|(0<<COM1B1)|(0<<COM1B0)|(0<<FOC1A)|
             (0<<FOC1B)|(1<<WGM11)|(0<<WGM10);
    TCCR1B = (0<<ICNC1)|(0<<ICES1)|(1<<WGM13)|(0<<WGM12)|(0<<CS12)|(0<<CS11)|
             (0<<CS10);
    ICR1 = 255;

    //AD prevodník nastavenia
    ADMUX = (1<<REFS1)|(1<<REFS0)|(0<<ADLAR)|(0<<MUX3)|(0<<MUX2)|
            (0<<MUX1)|(0<<MUX0);
    ADCSRA = (1<<ADEN)|(1<<ADFR)|(0<<ADIE)|(1<<ADPS2)|(1<<ADPS1)|
(1<<ADPS0);

    //časovač nastavenia
    TCCR2 = (0<<FOC2)|(0<<WGM20)|(0<<COM21)|(0<<COM20)|(1<<WGM21)|
            (1<<CS22)|(0<<CS21)|(1<<CS20);
    OCR2 = 3;
    //nastavené counter2 nonPWM, CTC mode, clk/128,top 3

    sei();                                 // zapínam globálne prerušenia
}
```

3.1.2 Zapis_cez_SPI

Táto funkcia má jednoduchú úlohu odosielať bytové dáta cez SPI rozhranie. Je volaná s parametrom, ktorý má velkosť osem bitov (veľkosť premennej deklarovanej ako unsigned char). Tento parameter sa ukladá do premennej data, ktorá má rovnakú veľkosť. Ďalej program pokračuje vo vykonávaní funkcie už len dvoma inštrukciami. SPI datový register SPDR sa naplní obsahom premennej data, čím sa prijatý byt začne posielať cez SPI zbernicu do dataflash. V ďalšom príkaze program už len čaká na dokončenie odosielania v slučke while. Ukončenie odosielania zisťujeme pomocou príznaku SPIF v registri SPSR. Po ukončení vysielania program vyskočí zo slučky while a pokračuje ukončením funkcie zapis_cez_SPI.

Pre presnejšiu predstavu o činnosti tejto funkcie tu uvádzam jej zdrojový kód:

```
void zapis_cez_SPI(unsigned char data)
{
    SPDR = data;                          //načítam data do SPDR
    while(!(SPSR & (1<<SPIF)));           //čakám, než sa odošlú
}
```

3.1.3 Cakajnaready

Túto funkciu sme navrhli z dôvodu potreby kontrolovať pripravenosť flash pamäte AT45DB321D prijímať ďalšie príkazy. Pri flash pamätiach, zapuzdrených vo väčších obaloch, býva vyvedený aj pin s označením READY/BUSY. Keďže však naša pamäť má len 8 pinov, výrobca nemohol tento pin umiestniť na vonkajšiu časť obalu. Stav pripravenosti pamäti je však možné kontrolovať pomocou status registra. Funkcia sa volá bez parametrov. Má za úlohu iba zdržať program vo vykonávaní ďalších príkazov, kým nebude dataflash pripravená vykonávať ďalšie príkazy. Prvá inštrukcia nastaví hodnotu logickej nuly na pin CS, čím aktivuje flash pamäť. Následne na to sa cez rozhranie SPI pomocou funkcie zapis_cez_SPI odošle príkaz na vyvolanie status registra flash pamäte. Ďalej program behá v slučke Do-While. V tejto slučke sa odošle do flash pamäte pomocou funkcie zapis_cez_SPI hodnota 0xFF. Následne na to nám flash pamäť vždy pošle hodnotu svojho status registra. Tú hneď testujeme v samotnej podmienke cyklu Do-While. Ak sa v hodnote status registra objaví príznak ready, program vyskočí zo slučky a nastaví flash pamäti vstup CS do hodnoty logickej jednotky, čím ju deaktivuje.

Pripájam zdrojový kód tejto funkcie:

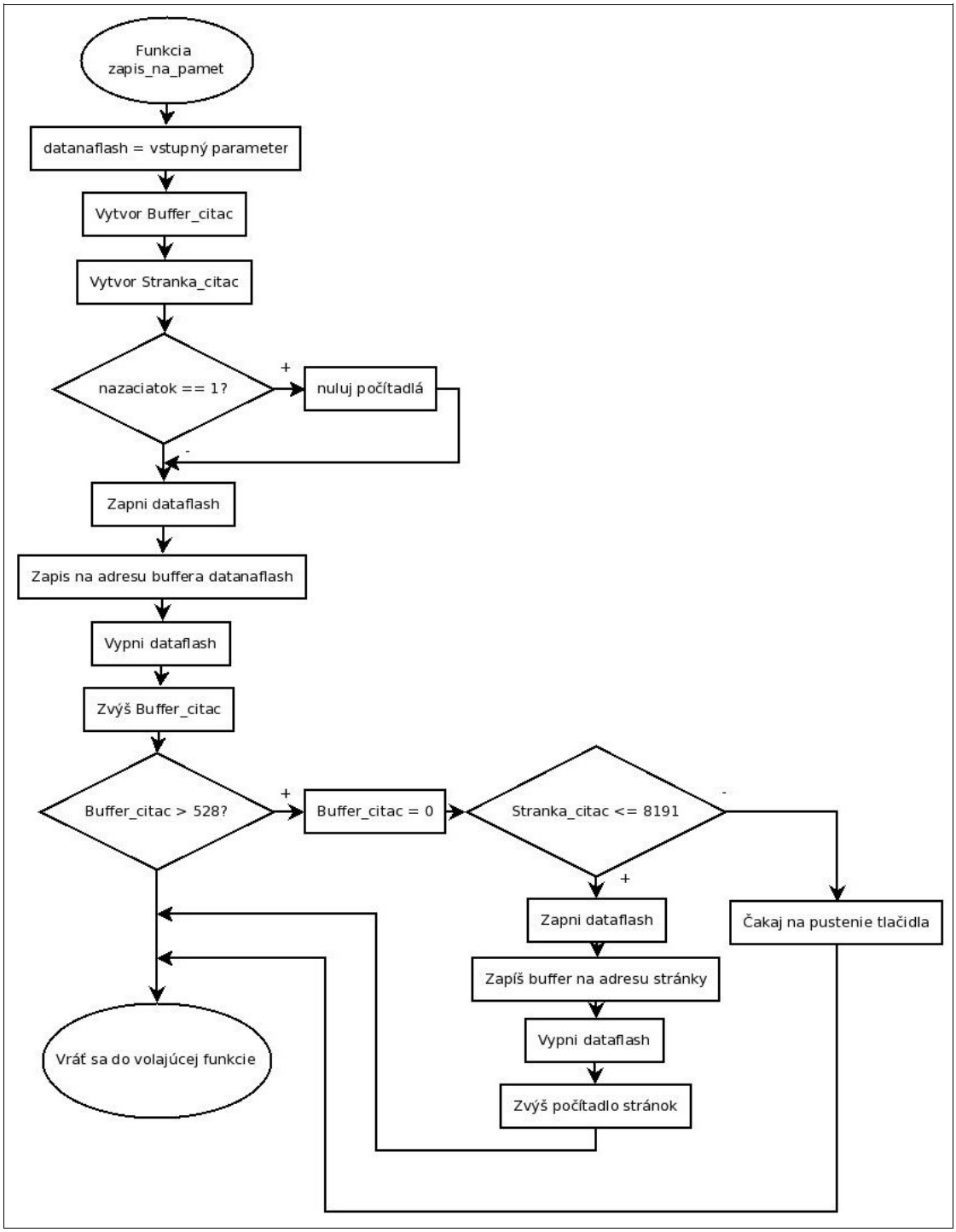
```
static void cakajnaready(void)
{
PORTB &= 0xBF;                          //zapneme flash pamäť nastavením 0 na CS
zapis_cez_SPI(STATUS_REGISTER);         //odošleme opkód cez SPI rozhranie
do {
     zapis_cez_SPI(0xFF);               // odošleme dummy hodnotu do SPDR
    } while ((SPDR & 0x80) == 0);       //beží v cykle, pokiaľ flash pamäť nieje ready
PORTB |= 0x40;                                  // vypni DataFlash
}
```

3.1.4 Zapis_na_pamet

Táto funkcia slúži na zápis dát za sebou na pamäť. Volá sa s parametrom o veľkosti jedného bytu. Funkcia si tento parameter ukladá do jednobytovej premennej datanaflash deklarovanej ako unisgned char. Na začiatku funkcie sú deklarované dve statické premenné buffer_citac a stranka_citac. Sú to statické premenné slúžiace ako počítadlo pre zaplnenie buffera a počítadlo pre počet zapísaných stránok. V nasledujúcom kroku funkcia testuje, či nebola nastavená statická premenná nazaciatok na hodnotu 1. Ak áno, vynuluje tieto počítadlá, čo znamená, že sa bude zapisovať od začiatku pamäte. Ak je premenná nazaciatok nastavená na hodnotu 0, celý cyklus If sa preskočí. Ďalšia inštrukcia, ktorá nasleduje, je zapnutie dataflash pomocou zápisu logickej nuly na bit CS. Hneď to tomto aktivovaní flash pamäte sa cez SPI rozhranie odosiela 5 bytov. Tieto byty obsahujú príkaz na zápis do bufferu 1 flash pamäte, 1don't care byt, 2 byty obsahujúce adresu v bufferi, na ktorú sa majú dáta zapísať, a posledný byt obsahuje samotné dáta určené na zápis. Po vykonaní tehto procedúry znova vypíname dataflash pomocou nastavenia logickej jednotky na pin CS a pripočítavame počítadlu buffer_citac hodnotu 1. Po inkrementovaní čítača adries buffera sa vykonáva kontrola, ktorá má za úlohu zistiť, či buffer, do ktorého sme zapisovali dáta, nie je už plný. Ak sa hodnota čítača rovná hodnote 528, čo znamená, že je buffer plný, čítač naplnenia buffera sa nuluje a testuje sa číslo stránky, ktorá má byť zapísaná. Ak je číslo v počítadle stránok menšie ako hodnota 8191, znamená to, že pamäť ešte nie je plná, a môže sa zapísať ďalšia stránka. Ak však rovnosť v podmienke platí, program skočí do slučky, v ktorej

zhasne LED diódu signalizujúcu nahrávanie, a čaká, pokým užívateľ nepustí tlačidlo pripojené na PINC 1 (tlačidlo na nahrávanie). Zápis ďalšej stránky v prípade, že nie je plná flash pamäť, prebieha nasledovne: aktivuje sa dataflash zapísaním logickej nuly na pin CS. Na pamäť sa cez rozhranie SPI pošle inštrukcia zápisu buffera 1 na pamäť, nasledovaná dvoma bytovými informáciami, obsahujúcimi adresu stránky, ktorá sa má zapísať, a jedným don't care bytom. Po tomto kroku sa vypne dataflash zapísaním logickej jednotky na pin CS, zvýši sa počítadlo stránok o hodnotu jedna a funkcia zapis_na_pamet skončí.

Pre lepšiu predstavu priebehu funkcie tu uvádzam jej vývojový diagram(Obr. 17). Zdrojový kód tejto funkcie z dôvodu jeho dĺžky uvádzam v prílohách (príloha II).



Obr.17 Vývojový diagram funkcie zapis_na_pamet

3.1.5 Napln

Funkcia s názvom napln má jednoduchý princíp. Volá ju hlavný program bez akéhokoľvek vstupného parametru. Po skočení programu do tejto funkcie sa ako prvý príkaz vykoná rozsvietenie LED diódy, pripočítaním špecifickej masky ku portu B. Ďalej nasleduje cyklus While, ktorý opakuje sekvenciu inštrukcií, ktorá je v ňom zapísaná, až pokým užívateľ nepustí tlačidlo na nahrávanie (na kontakte pinC 1 sa objaví logická jednotka). Tento cyklus obsahuje postupnosť troch inštrukcií. Prvá inštrukcia je ďalšia slučka While, ktorá čaká, pokým časovač nahrávania nenastaví premennú príznak na hodnotu 1. Po vyskočení z tohto cyklu (čakania) sa zavolá funkcie zapis_na_pamet a ako parameter tohto volania sa použije obsah registra ADCH (horný byt výsledku AD konverzie). Keď sa program vráti naspäť do funkcie napln, nastaví premennú priznak na hodnotu 0. V prípade, že užívateľ pustí tlačidlo na nahrávanie (pripojené ku vstupnému pinu pinC 1) program vyskočí z cyklu While, zhasne LED diódu vynásobením registra B príslušnou maskou a vyskočí z funkcie napln.

Uvádzam komentovaný zdrojový kód programu.

```
void napln(void)
{
    PORTB |= 0x80;                          // zažnem LED
    while(!(PINC & _BV(1)))                 // pokiaľ nie je pustené tlačidlo record
    {
        while(priznak != 1)                 //čakám na príznak
        {}
    zapis_na_pamet(ADCH);
    priznak=0;                              //nulujem príznak
    }
    PORTB &= 0x7F;                          //vypínam LED
}
```

3.1.6 Zmaz_flash

Táto funkcia slúži, ako je už z jej názvu jasné, na vymazanie celej flash pamäte. Toto mazanie sa prevádza po blokoch. Funkcia sa volá bez vstupného parametra. Hneď na jej začiatku spustíme cyklus For, ktorý si deklaruje internú premennú s názvom pocitadlo. Táto funkcia zopakuje vloženú sekvenciu príkazov toľkokrát, pokým sa premenná pocitadlo nerovná hodnote 0x3FF. Počítadlo je inicializované s hodnotou 0 a po každom zrealizovaní setu inštrukcií sa inkrementuje o hodnotu 1. V cykle For sú vložené tieto príkazy: program pomocou nastavenia logickej nuly na pin CS aktivuje flash pamäť. Následne na to odošle štvoricu bytov, z ktorých prvý byt je opkód pre inštrukciu mazanie bloku, druhý a tretí byt obsahujú číslo bloku, ktorý sa má vymazať, a posledný byt nesie hodnotu 0x00 (don't care value). Ďalším príkazom program volá funkciu cakajnaready, z ktorej sa vráti až po tom, čo je flash pamäť znovu pripravená prijímať príkazy. Následne sa nastavením pinu CS na logickú jednotku dataflash deaktivuje. Táto sekvencia príkazov sa opakuje a počítadlo sa inkrementuje, pokým niesu zmazané všetky bloky. Po vyskočení z cyklu For, sa nastaví hodnota premennej nazaciatok na hodnotu 1 (budúci zápis na pamäť odzačiatku), a vynásobením portu B vhodnou maskou sa zhasne dióda LED. Program potom čaká v cykle while na pustenie tlačidla pre mazanie pamäte (pinC 2) a keď z tejto slučky vyskočí, vráti sa do funkcie, z ktorej bolo mazanie volané.

Uvádzam komentovaný zdrojový kód tejto funkcie pre lepšiu predstavu jej princípu:

```
void zmaz_flash(void)
{
for(unsigned int poc=0 ; poc <= 0x3FF ; poc++)
        {
        PORTB &= 0xBF;                                  // zapni DataFlash
        zapis_cez_SPI(0x50);
        zapis_cez_SPI((char)(poc >> 3));
        zapis_cez_SPI((char)(poc << 5));
       zapis_cez_SPI(0x00);
       PORTB |= 0x40;                                   // vypni DataFlash
       cakajnaready();
       }
nazaciatok = 1;                                 //funkcia nahrávania od začiatku
```

```
PORTB &= 0x7F;                          //vypni LED
while (!( PINC & _BV(2) ));             //čakaj na pustenie tlačidla
}
```

3.1.7 Druhastranka_druhybuffer

Úlohou tejto funkcie je odoslať dataflash pamäti príkaz na načítanie ďalšej stránky v poradí prehrávania do nepoužívaného buffera. Funkcia je volaná pomocou dvoch parametrov s veľkosťou jeden byte. Prvý parameter udáva, ktorý buffer dataflash je aktuálne aktívny a fukncia si ho uloží do premennej s názvom aktívnypreh_buffer. Druhý parameter udáva číslo stránky, ktorá má byť načítaná do neaktívneho buffera. Tento parameter si funkcia uloží do premennej strankapreh_citac. Prvý krok funkcie spočíva v aktivovaní dataflash pomocou vynásobenia portu B príslušnou maskou. Tým je docielené nastavenie pinu s funkciou CS na nulu, čím sa dataflash aktivuje. Nasleduje podmienka if, ktorá rozhoduje, aký príkaz sa odošle dataflash. Ak sa hodnota uložená v premennej rovná 1, program odošle príkaz na plnenie buffera 2. Ak je hodnota uložená v tejto premennej, akákoľvek iná hodnota od hodnoty 1, pošle sa na dataflash príkaz na plnenie buffera 1. Po vykonaní rozhodnutia tejto podmienky a odoslaní príslušného príkazu cez SPI zbernicu pomocou funkcie zapis_cez_SPI nasleduje odoslanie ďalších troch bytov, ktoré obsahujú správne zarovnané číslo stránky v dvoch registroch a posledný byt s hodnotou 0x00 – don't care value. Tieto byty sú tiež odosielané cez SPI pomocou nám už známej funkcie zapis_cez_SPI. Po odoslaní týchto dokopy štyroch bytov je skompletovaná inštrukcia pre kopírovanie stránky na buffer. Samotné kopírovanie sa začne po deaktivovaní dataflash, ktoré program prevádza hneď v ďalšom kroku pomocou pripočítania správnej masky k portu B. Na pin CS sa nastaví jednotka a dataflash kopíruje určenú stránku do určeného buffera. Posledný príkaz funkcie druhastranka_druhybuffer volá funkciu cakajnaready bez vstupného parametra, čím docielime čakanie programu na dokončenie procesu interného kopírovania v dataflash. Po zmene stavu dataflash na READY funkcia druhastranka_druhybuffer končí a program sa vracia do funkcie, z ktorej bola táto funkcia volaná.

Pre lepšiu predstavu a pochopenie tejto funkcie uvádzam jej zapoznámkovaný zdrojový kód:

```
void druhastranka_druhybuffer(unsigned char aktivnypreh_buffer, unsigned int strankapreh_citac)
{
PORTB &= 0xBF;                          // zapni DataFlash
if (aktivnypreh_buffer == 1)            // pokiaľ je buffer 1 aktívny
   {
```

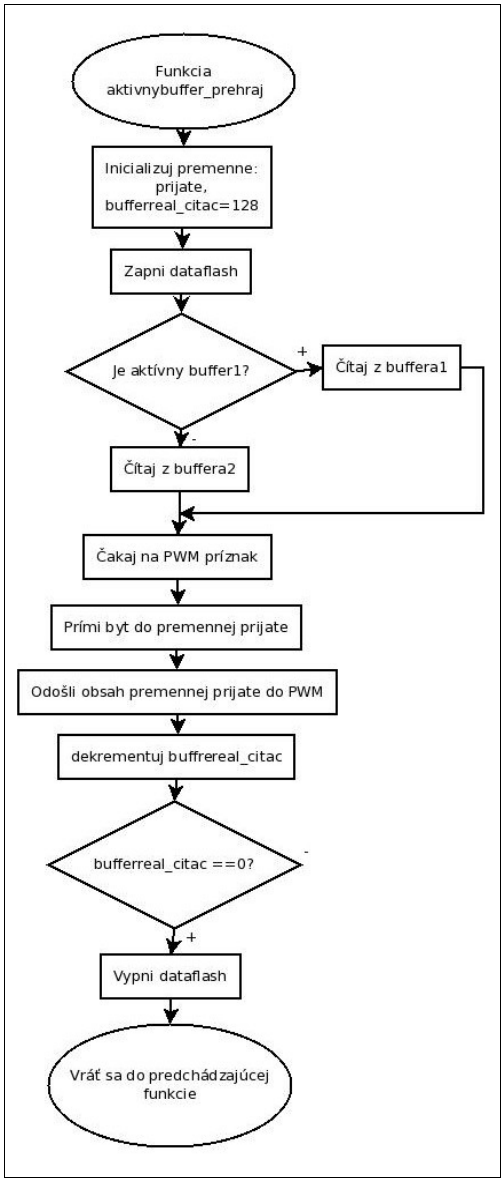
```
        zapis_cez_SPI(MM_PAGE_TO_B2_XFER);          // kopíruj stránku do buffera 2
    }
    else                                // else
    {
         zapis_cez_SPI(MM_PAGE_TO_B1_XFER);          // kopíruj stránku do buffera 1
    }
    zapis_cez_SPI((char)(strankapreh_citac >> 6));          //tri bajty obsahujúce zarovnanú adresu
    zapis_cez_SPI((char)(strankapreh_citac << 2));
    zapis_cez_SPI(0x00);                                    //a don't care value
PORTB |= 0x40;        // vypni DataFlash a spusti kopirovanie
cakajnaready();
}
```

3.1.8 Aktivnybuffer_prehraj

Funkcia aktivnybuffer_prehraj je funkciou, ktorá sa stará o samotné prijímanie dát z aktívneho buffera a zároveň ich odosielanie vzorkovacou frekvenciou do vstupu PWM. Volá sa len s jedným bytovým vstupným parametrom. Tento parameter určuje, ktorý buffer je aktívny, čiže určuje, z ktorého buffera sa má audio prehrávať, a ukladá sa do premennej aktivnypreh_buffer. Na začiatku funkcie sa deklarujú dve pomocné premenné typu unsigned int, s názvami prijate a bufferreal_citac. Program pokračuje tým, že vynásobením portu B vhodnou maskou zapíše na pin CS hodnotu logickej nuly, čím sa aktivuje dataflash. Nasleduje podmienka If, ktorá vyhodnocuje ktorý z bufferov je aktívny, a následne je odosielaný prvý byt cez SPI pomocou funkcie zapis_cez_SPI, ktorý obsahuje príkaz na čítanie z príslušného aktívneho buffera. Ak je hodnota premennej aktivnypreh_buffer rovná 1, aktívny buffer je buffer číslo jedna. V ostatných prípadoch je aktívny buffer bufer číslo 2. Po odoslaní tohto príkazu nasleduje ešte odoslanie dalších štyroch bytov cez zbernicu SPI. Prvé dva byty sú s hodnotou 0x00 – don't care. Ďalší byt špecifikuje adresu v bufferi, od ktorej sa začne buffer čítať, čo je tiež adresa 0x00. Posledný byt je ďalšia don't care hodnota. Po tejto inštrukcii by mala byť dataflash pamäť pripravená posielať dáta zo zvoleného buffera. Toto čítanie je zrealizované pomocou slučky Do-While, ktorá zopakuje danú postupnosť príkazov celkom 528-krát. Na začiatku tejto slučky je ďalšia slučka, ktorá má iba funkciu čakania na nastavenie príznaku prerušenia. Po príznaku prerušenia z PWM program odošle cez SPI port hodnotu 0xFF, čím si vyžiada byt dát z flash pamäte. Tento byt je okamžite

možné čítať z registra SPDR, čo aj robíme, a ukladáme ho do premennej prijate. V ďalších dvoch krokoch programu sa nastaví príznak v premennej priznak PWM na nulu a hodnota z pomocnej premennej prijate sa odošle do PWM cez register OCR1A. Keď funkcia while skončí, znamená to, že už je prečítaný celý buffer. Preto funkcia aktivnybufer_prehraj už len nastaví hodnotu logickej jednotky na pin CS, čím deaktivuje flash pamäť. Po vykonaní tohto posledného príkazu sa program vráti naspäť do funkcie, z ktorej bola táto funkcia volaná.

Pre prehľadnosť a správne pochopenie uvádzam vývojový diagram funkcie aktivnybufer_prehraj (Obr. 18) . Zdrojový kód tejto funkcie je pre jeho rozsiahlosť umiestnený v prílohách (príloha III).



Obr.18 Vývojový diagram funkcie aktivnybuffer_prehraj

3.1.9 Hraj

Funkcia hraj je vlastne funkcia, ktorá obsahuje algorytmus volania funkcií pomocou počítadiel a podmienok. Je typu void, čiže sa volá bez parametrov. Na začiatku funkcie sa deklarujú interné premenné funkcie strankapreh_citac a aktivnypreh_buffer. Čítač stránok je typu unsigned int a nastaví sa na hodnotu 0. Príznak aktívneho buffera je typu unsigned char a nastaví sa na hodnotu 1, čo znamená, že aktivujeme prvý buffer. Hneď na to však premennú aktivnypreh_buffer inkrementujeme, čo ho zvýši na hodnotu 2. Ďalej voláme funkciu druhastranka_druhybuffer, s aktuálnymi hodnotami pomocných premenných funkcie. Toto nám spôsobí, že buffer 1 sa naplní hodnotami prvej stránky pamäte. V ďalšom príkaze naspäť dekrementujeme príznak aktívneho buffera, čím nastavíme buffer 1 ako aktívny. Nasleduje cyklus while, ktorý opakuje sekvenciu príkazov dokola dovtedy, kým nie sú prečítané všetky stránky, alebo kým nie je pustené tlačidlo pre prehrávanie. Príkazy opakujúce v tomto cykle najprv inkrementujú čítač stránok a následne volajú funkciu druhastranka_druhybuffer, s aktuálnymi hodnotami pomocných premenných. Po naplnení neaktívneho buffera sa volá funkcia aktivnybuffer_prehraj, ktorá prehrá aktívny buffer pomocou PWM. Na konci postupnosti príkazov v slučke While je podmienka If, ktorá zistí aktívny buffer, ten deaktivuje a aktivuje neaktívny buffer jednoducho, pomocou prepísania pomocnej premennej aktivnypreh_buffer. Po vyskočení programu z prehrávacej slučky program nevykoná žiadny ďalší príkaz a vyskočí aj z vlastnej funkcie hraj do funkcie, z ktorej bol volaný.

Pre lepšie pochopenie princípu fungovania tejto funkcie uvádzam jej zakomentovaný zdrojový kód:

```
void hraj(void)
{
   unsigned int strankapreh_citac = 0;
   unsigned char aktivnypreh_buffer = 1;        // aktívny buffer = buffer1
        aktivnypreh_buffer++;                          // aktívny buffer = buffer 2

        druhastranka_druhybuffer(aktivnypreh_buffer , strankapreh_citac); // stranka 0 do buffer 1
        aktivnypreh_buffer--;                          // aktívny buffer = buffer 1

        while ((strankapreh_citac < 8191)&(!(PINC & _BV(3)))) // pokiaľ nie je koniec pamäte
```

```
                                                    // alebo nepustí tlačidlo
    {
      strankapreh_citac++;                          // zvýš čítač stránok
      druhastranka_druhybuffer(aktivnypreh_buffer , strankapreh_citac);
      aktivnybuffer_prehraj(aktivnypreh_buffer);    // prehraj cez PWM

      if (aktivnypreh_buffer == 1)                  // pokiaľ je buffer 1 aktívny
              {
               aktivnypreh_buffer++;                // nastav buffer 2  aktívny
               }
      else                                          // inak
              {
              aktivnypreh_buffer--;                 // nastav buffer 1 aktívny
               }
   }
}
```

3.1.10 Prehraj

Ide o jednoduchú funkciu, ktorá je volaná bez vstupného parametra. Jej prvá inštrukcia je volanie funkcie hraj. Po návrate z tejto funkcie sa vynásobí port B príslušnou maskou, čím spôsobíme zhasnutie LED diódy signalizujúcej činnosť záznamníka. Po vykonaní týchto dvoch inštrukcií sa program vráti do funkcie, z ktorej bola funkcia prehraj volaná.

Táto funkcia je jednoduchá, avšak pre úplnosť uvádzam jej zdrojový kód:

```
void prehraj(void)
{
hraj();
PORTB &= 0x7F;      //vypni LED
}
```

3.1.11 Main

Funkcia main je hlavná funkcia programu, ktorá sa spúšťa ihneď po pripojení mikrokontroléra na napájacie napätie (prípadne ešte po vykonaní funkcií bootloadera). Je to telo programu, ktoré riadi a volá jednotlivé funkcie. Od funkcie pre nastavenie portov, čítačov/časovačov a SPI rozhrania, cez nahrávanie na flash pamäť, až po mazanie flash pamäte. Po spustení funkcie main sa ako prvá inštrukcia vykonáva volanie funkcie setup. Funkciu setup voláme bez parametrov. Slúži na nastavenie portov, čítačov/časovačov, AD prevodníka a ostatných súčastí mikrokontroléra, ktoré budeme v ďalších častiach programu využívať. Po návrate programu z funkcie setup skočí program do nekonečnej slučky While. V tejto slučke program stále dokola testuje jednotlivé nadefinované podmienky If a v prípade, že hodnota parametru uvedená v podmienke If má hodnotu logickej jednotky, vykonajú sa príkazy, ktoré sú pre túto podmienku nadefinované. Jednotlivé podmienky a ich vykonanie rozoberáme v nasledujúcich odsekoch.

if (!(PINC & _BV(1))) - testujeme, či sa nachádza na pine 1 portu B logická nula. Ak nie, podmienka nie je splnená a program overuje ďalšie z podmienok. Ak však táto podmienka splnená je, program vykoná príkazy zabalené v tejto podmienke. Prvý príkaz spočíva v nastavení bitu ADSC v registri ADCSRA na hodnotu logickej jednotky, čím spustíme AD prevodník. Pomocou druhého príkazu nastavíme na hodnotu logickej jednotky bit OCIE2 z registra TIMSK, čím povolíme prerušenie pretečenia časovača 2. Nasledujúcim príkazom voláme funkciu napln bez parametra. Popis tejto funkcie a jej ďalších volaní je uvedený v predchádzajúcich kapitolách, preto len poznamenám, že táto funkcia slúži na spustenie nahrávania zvukov do pamäte flash. Návrat programu z tejto funkcie znamená, že užívateľ už pustil tlačidlo pre nahrávanie umiestnené na pine 1 portu B. Preto je nasledujúca inštrukcia programu určená na nastavenie hodnoty logickej nuly na bit OCIE2 registra TIMSK, čím vypneme prerušenie pretečenia časovača 2. Nasledujúci príkaz slúži na nastavenie hodnoty logickej nuly na bit ADCS registra ADCSRA, ktorý vypne konverziu AD prevodníka. Po tejto akcii program vyskočí z obsluhy splnenej podmienky If a znova behá stále dokola v slučke While.

else if (!(PINC & BV(2))) – ak program nevykonal obsluhu pre splnenú predchádzajúcu podmienku, testuje, či sa hodnota logickej nuly nenachádza na pine 2 portu B. Ak nie, program testuje podmienky ďalej. Ak sa však táto hodnota na určenom pine nachádza, podmienka je splnená a program vchádza do obsluhy splnenia podmienky. Tá je jednoduchá. Program ako prvú inštrukciu obsluhy splnenej podmienky vykoná zasvietenie LED diódy signalizujúcej činnosť záznamníka pomocou pripočítania správnej masky k portu B. Nasledujúci krok programu je volanie funkcie zmaz_flash bez parametru. Po návrate programu z tejto funkcie je flash už

zmazaná, tlačidlo pripojené k pinu 2 na porte B je pustené a LED dióda signalizujúca činnosť záznamníka je už zhasnutá. Preto program už len vyskočí z obsluhy splnenia podmienky.

else if (!(PINC & _BV(3))) – ak program nevykonal žiadnu z predchádzajúcich podmienok, testuje, či sa na pine 3 portu B nenachádza hodnota logickej nuly. Ak je podmienka splnená, program vykoná inštrukcie, ktoré sú v podmienke zabalené. Ako prvá inštrukcia nasleduje rozsvietenie LED diódy pomocou pripočítania masky k portu B. Túto inštrukciu nasleduje naplnenie bitu CS10 nachádzajúceho sa v registre TCCR1B hodnotou logickej jednotky, čím timeru nastavíme správny prescaler a PWM začne bežať. V ďalšom kroku naplníme bit TOIE1 nachádzajuci sa v registri TIMSK hodnotou logickej jednotky, čím povolíme prerušenia generované PWM. Nasledujúci príkaz volá funkciu prehraj bez parametra. Táto funkcia prehráva pomocou PWM dáta získavané z dataflash rýchlosťou určenou vzorkovacou frekvenciou. Program sa z tejto funkcie vráti až po pustení tlačidla pripojeného k pinu 3 portu B, slúžiaceho na prehrávanie. V ďalšom kroku nastaví program bit TOIE v registri TIMSK naspäť na hodnotu nula, čím zakáže prerušenia generované pomocou PWM. Tak isto sa vynuluje aj bit CS10 nachádzajúci sa v registri TCCR1B, čím timer 1 zastavíme (vypneme PWM). Program sa po vykonaní týchto príkazov vráti do slučky While nachádzajúcej sa vo funkcii main.

Else – ak nebola splnená žiadna z predchádzajúcich podmienok, program nič nevykoná a testuje jednotlivé piny portu v slučke While stále dokola.

Pre prehľadnosť a lepšie pochopenie programu uvádzam zakomentovaný zdrojový kód funkcie main.

```
int main()
{
 setup();
   while(1)
  {
     if (!(PINC & _BV(1)))                                  // ked stlačím tlačidlo PB1
     {
                    ADCSRA |= (1<<ADSC);                    //spusti prevodník
                    TIMSK = (1<<OCIE2)|(0<<TOIE2);          // spustíme prerušenia časovača
                    napln();
```

```
                TIMSK &= ~(1<<OCIE2);                    //zastavíme prerušenia časovača
                ADCSRA &= ~(0<<ADSC);                    //zastaví prevodník
            }
            else if (!(PINC & _BV(2)))                   // ked stlačíme tlačidlo PB2
            {
             PORTB |= 0x80;                                      // zažneme LED
             zmaz_flash();
             }
            else if (!(PINC & _BV(3)))                   // keď stlačíme tlačidlo PB3
            {
             PORTB |= 0x80;                                      // zažneme LED
            TCCR1B |= (1<<CS10);                    //spustíme PWM
            TIMSK = (0<<TICIE1)|(0<<OCIE1A)|(0<<OCIE1B)|(1<<TOIE1);
             prehraj();
             TIMSK &= ~(1<<TOIE1);
            TCCR1B &= ~(1<<CS10);                    //vypneme PWM
            }
            else
            {}                                               //nečinný
    }
        return(0);
}
```

3.1.12 Obsluhy prerušení

Obsluha prerušení je časť programu, ktorá sa vykoná po zaznamenaní niektorého z prerušení a je definovaná obsluha pre jeho vektor. V mojom programe používame dve obsluhy prerušení: TIMER2_COMP_vect a TIMER1_OVF_vect.

TIMER2_COMP_vect nastaví pri obsluhe prerušenia hodnotu premennej priznak na hodnotu logickej jednotky.

TIMER1_OVF_vect nastaví pri obsluhe prerušenia hodnotu premennej priznakpwm na logickú jednotku.

Obidve tieto premenné sú testované a nulované v programe.

Pre lepšie pochopenie a predstavu uvádzam zakomentovaný zdrojový kód týchto obslúh prerušení:

```
ISR(TIMER2_COMP_vect)          // ak pretečie časovač
{
        priznak=1;
}

ISR(TIMER1_OVF_vect)           // ak pretečie čítač PWM
{
priznakPWM=1;
}
```

ZÁVER

Cieľom tejto práce bol návrh a vytvorenie jednoduchého záznamníka zvukov. Tento záznamník je primárne určený na nahrávanie hovoreného ľudského slova a prehrávanie uložených záznamov. Jeho funkcie je možné jednoducho ovládať pomocou troch tlačidiel. Pri návrhu tohto zariadenia a jeho konštrukcii bolo narazené na niekoľko komplikácií, a ťažkostí spojených so správnou frekvenciou vzorkovania a správnym nastavením súčastí mikrokontroléra, ktoré sa však nakoniec z väčšej miery podarilo vyriešiť. Vytvorený prototyp tohto zariadenia je funkčný, a disponuje všetkými funkciami, ktoré boli pri jeho návrhu požadované.

Táto aplikácia má veľkú východu v možnosti jednoduchej modifikácie schémy zapojenia a programu, pre iné účely ako je samotné nahrávanie ľudského hlasu. Jednoduchou úpravou schémy zapojenia a programu je možné zariadenie skonštruovať pre opakované prehrávanie alebo nahrávanie hlasového záznamu, pri vyvolaní vonkajšieho podnetu.

Hlavný prínos tejto práce je uvedenie teoretických zanlostí o digitálnom zázneme zvuku, jeho ukladaní a prehrávaní do praxe. Aj konštrukciou tohto jednoduchého zariadenia sú čitateľovi priblížené veľké, stále narastajúce možnosti práce s mikrokontrolérmi a ich praktickým využitím v každodennom živote.

ZÁVER V ANGLICKOM JAZYKU

The goal of this thesis was to design and create a model of a simple sound recorder. It is designed specifically to record human voice and playback the sound recordings. Three buttons are used to access its functions. When developing this model, there came up several complications, and difficulties connected with the correct frequency of sampling and the right setting of the parts of the microcontroller, which however were to a large extent resolved. The designed prototype of this device is functional and has all the required funtions as set during designing of the draft.

One of the assets of this application is its simple way of modifying the scheme of connection and program, for other uses besides simple recording of human voice. With a simple modification of scheme of connection and program, it is possible to design the device for repetition recording, and recording of voice based on an external impulse.

Main contribution of this work is introduction of theoretic knowledge about digital sound recording, its storage and playback into reality. Even through design of this simple device, the reader can access big and ever enlarging ways of working with microcontrollers, and its practical use in everyday life.

ZOZNAM POUŽITEJ LITERATÚRY

[1] Syntéza psychostimulačných signálov a ich aplikácia[online]. Eduard Palkovský, 2009 [cit. 2010-5-21]. Dostupný zWWW: https://www.stag.utb.cz/apps/stag/dipfile/index.php?download=12139

[2] ATMega8 Datasheet [online]. Atmel Corporation, 2009 [cit. 2010-01-26]. Dostupný z WWW: http://www.atmel.com/atmel/acrobat/doc2486.pdf

[3] AT45DB321D DataFlash Datasheet [online]. Atmel Corporation, 2009 [cit. 2010-01-25]. Dostupný z WWW: http://www.atmel.tw/dyn/resources/prod_documents/doc3597.pdf

[4] AVR335: Digital Sound Recorder with AVR and DataFlash [online]. Atmel Corporation, 2005 [cit. 2010-01-25]. Dostupný z WWW: http://www.atmel.com/dyn/resources/prod_documents/doc1456.pdf

[5] LM124, LM124A, LM224, LM224A, LM324, LM324A, LM2902, LM2902V, LM224K, LM224KA, LM324K, LM324KA, LM2902K, LM2902KV, LM2902KAV QUADRUPLE OPERATIONAL AMPLIFIERS[online]. Texas Instruments, 2005 [cit. 2010-1-25]. Dostupný z WWW: http://www.datasheetcatalog.org/datasheet2/6/0ep2cg5u3zir6d7op48zff35l8py.pdf

[6] PINKER, Jiří. Mikroprocesory a Mikropočítače. Praha : BEN – technická literatura, 2004. 220 s. ISBN 80-7300-110-1.

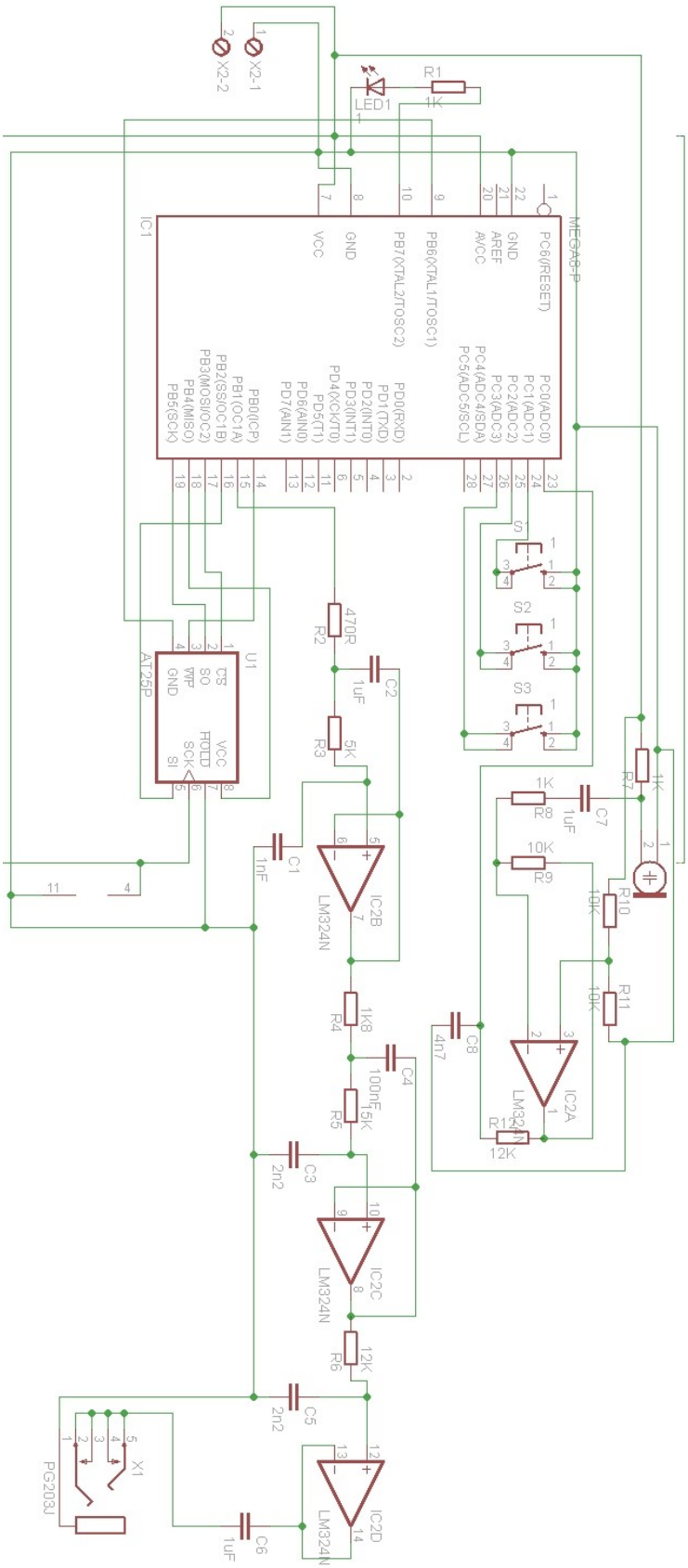
[7] MANN, Burkhard. C pro mikrokontroléry. Praha : BEN - technická literatura, 2004. 280 s. ISBN 80-7300-077-6.

[8] CATSOULIS, John. Designing Embedded Hardware. O'Reilly Media, 2005. 400 s. ISBN 978-0-596-00755-3.

ZOZNAM OBRÁZKOV

ZOZNAM PRÍLOH

PRÍLOHA I: ZVAČŠENÁ SCHÉMA ZAPOJENIA ZÁZNAMNÍKA



PRÍLOHA II: ZDROJOVÝ KÓD FUNKCIE ZAPIS_NA_PAMET

```
void zapis_na_pamet(unsigned char datanaflash)
{
	static unsigned int buffer_citac;		//buffer počítadlo
	static unsigned int stranka_citac;		//stránky počítadlo
	if(nazaciatok==1)	//ak zapisujeme od začiatku pamäti, nulujem počítadlá
	{
		buffer_citac=0;
		stranka_citac=0;
		nazaciatok=0;
	}
	else
	{}
	PORTB &= 0xBF;					//zapnem Dataflash
	zapis_cez_SPI(BUFFER_1_WRITE);
	zapis_cez_SPI(0x00);				// don't care
	 zapis_cez_SPI((char)(buffer_citac>>8));	// don't cares plus prveé dva bity adresy
	zapis_cez_SPI((char)buffer_citac);		// adresa buffera (max. 2^8 = 256 stranok)
	zapis_cez_SPI(datanaflash);		// zapíše data do SPIDR
	PORTB |= 0x40;					//vypnenem Dataflash
	buffer_citac++;
	if(buffer_citac > 528)
	{
		buffer_citac=0;
		if(stranka_citac <= 8191)
		{
			PORTB &= 0xBF;			// zapni DataFlash

	zapis_cez_SPI(B1_TO_MM_PAGE_PROG_WITHOUT_ERASE);// zapíš buffer 1
	zapis_cez_SPI((char)(stranka_citac>>6));
	zapis_cez_SPI((char)(stranka_citac<<2));
	zapis_cez_SPI(0x00);						// don't cares

	PORTB |= 0x40;					// vypni DataFlash
	stranka_citac++;
		}
```

```
                else
                {
                 PORTB &= 0x7F;      //vypni LED
                        while (!( PINC & _BV(1) )) //čakaj, kým nieje pustené tlacitko
                        {}
                }
        }
        else
        {}
}
```

PRÍLOHA III: ZDROJOVÝ KÓD FUNKCIE AKTIVNYBUFER_PREHRAJ

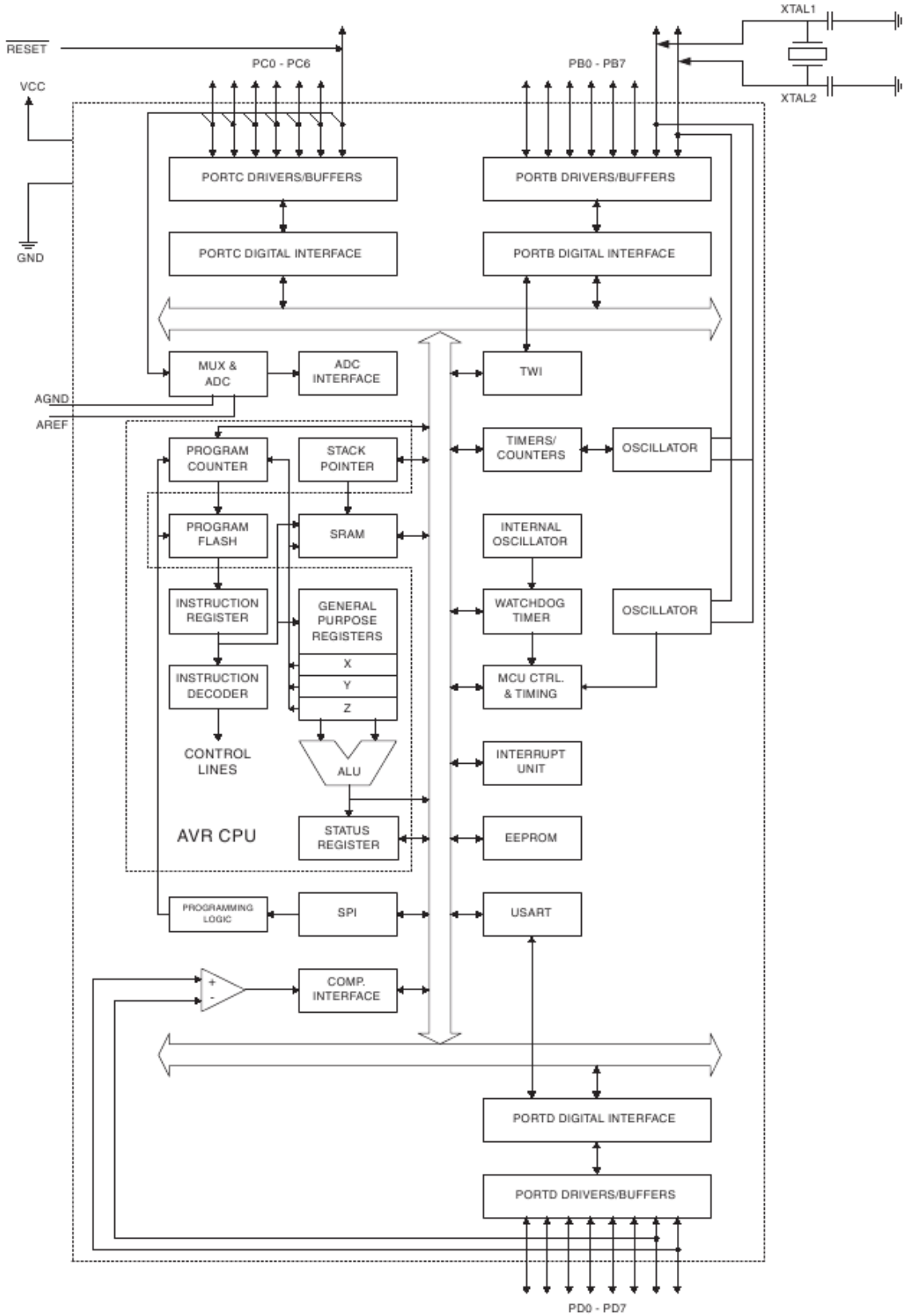
```
void aktivnybuffer_prehraj(unsigned char aktivnypreh_buffer)
{
   unsigned int prijate;
   unsigned int bufferreal_citac = 528;

   PORTB &= 0xBF;                                  // zapni DataFlash

   if (aktivnypreh_buffer == 1)                    // pokial je aktívny buffer 1
   {
      zapis_cez_SPI(BUFFER_1_READ);                // čítaj z buffera 1
   }
   else                                            // inak
   {
      zapis_cez_SPI(BUFFER_2_READ);                // čítaj z buffera 2
   }
   zapis_cez_SPI(0x00);                  // don't care
   zapis_cez_SPI(0x00);                  // don't care
   zapis_cez_SPI(0x00);                  // čítaj od pozície 0 v bufferi
   zapis_cez_SPI(0x00);                  // don't care

   do
   {
                        while(priznakPWM != 1) // pokiaľ nieje nastavený príznak, čakaj
                        {
                        }
      zapis_cez_SPI(0xFF);                  // odošli dummy hodnotu pre prijatie bytu
      prijate = SPDR;                              // zapíš data do premennej prijate
        priznakPWM=0;                              // nuluj príznak
        OCR1A = prijate;                           // odošli data do PWM
    } while (bufferreal_citac--);                  // dekrementuj citac, pokial > 0, pokračuj
    PORTB |= 0x40;                                 // vypni DataFlash
}
```

PRÍLOHA IV: BLOKOVÝ NÁKRES POUŽITÉHO MIKROKONTROLÉRA



PRÍLOHA V: CELÝ ZDROJOVÝ KÓD PROGRAMU

```
#include <avr/io.h>
#include<util/delay.h>
#include<avr/pgmspace.h>
#include <avr\interrupt.h>
#include "dataflash.h"
void setup (void);                                                    //deklarujem popredu
void napln (void);
void zapis_cez_SPI(unsigned char data);
void zapis_na_pamet(unsigned char datanaflash);
void zmaz_flash(void);
unsigned char min;
unsigned char max;
volatile char priznak;
volatile char priznakPWM;
volatile unsigned int temp;
volatile unsigned int temp2;
unsigned int nazaciatok = 0 ;
unsigned char predch = 0x00;
unsigned char predchodv = 0x00;
ISR(TIMER2_COMP_vect)                 // ak pretečie časovač
{priznak=1; }
ISR(TIMER1_OVF_vect)                  // ak pretečie čítač PWM
{priznakPWM=1;}
static void cakajnaready(void)
{
PORTB &= 0xBF;                                    //zapneme flash pamäť nastavením 0 na CS
zapis_cez_SPI(STATUS_REGISTER);              //odošleme opkód cez SPI rozhranie
do {
zapis_cez_SPI(0xFF);                          // odošleme dummy hodnotu do SPDR
 } while ((SPDR & 0x80) == 0);                //beží v cykle, pokiaľ flash pamäť nieje ready
PORTB |= 0x40;                                                    // vypni DataFlash
}
void setup(void)
{
   DDRB = 0xEF;                    // SPI Port inicializácia
                                 // LED , CS,  SCK  MISO, MOSI, WP ,  PWM,   RST
                                    // PB7, PB6,  PB5,  PB4, PB3, PB2 ,  PB1,   PB0
                                   // O   O    O    I   O   O    O    O
```

```
                                        // 1   1   1   0   1   1   1   1
   SPCR = (1<<SPE)|(1<<MSTR)|(0<<SPR0)|(0<<SPR1);
   SPSR = (1<<SPI2X);                                                    //master, fosc/2
  //nastavenie portov
   PORTB = 0xFF;                                          // všetky výstupy 1, vstupy majú
                                                  // zapnuté pull-up (LED kontrolka nesvieti)
   PORTB &= 0x7F;                                  //vypni LED
   DDRC = 0x00;                                   // port C má všetky piny vstupné
   PORTC = 0xFF;                                  // vstupy pull-up
     //PWM nastavenia (clock vypnuté)
   TCCR1A = (1<<COM1A1)|(0<<COM1A0)|(0<<COM1B1)|(0<<COM1B0)|(0<<FOC1A)|
        (0<<FOC1B)|(1<<WGM11)|(0<<WGM10);
   TCCR1B = (0<<ICNC1)|(0<<ICES1)|(1<<WGM13)|(0<<WGM12)|(0<<CS12)|(0<<CS11)|
        (0<<CS10);
    ICR1 = 255;
     //AD prevodník nastavenia
         ADMUX = (1<<REFS1)|(1<<REFS0)|(0<<ADLAR)|(0<<MUX3)|(0<<MUX2)|
               (0<<MUX1)|(0<<MUX0);
         ADCSRA = (1<<ADEN)|(1<<ADFR)|(0<<ADIE)|(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0);
             //časovač nastavenia
         TCCR2 = (0<<FOC2)|(0<<WGM20)|(0<<COM21)|(0<<COM20)|(1<<WGM21)|
         (1<<CS22)|(0<<CS21)|(1<<CS20);
         OCR2 = 3;
         //nastavené counter2 nonPWM, CTC mode, clk/128,top 3
 sei();                                          // zapínam globálne prerušenia
}
void zapis_cez_SPI(unsigned char data)
{
         SPDR = data;                                                  //načítam data do SPDR
         while(!(SPSR & (1<<SPIF)));                          //čakám, než sa odošlú
}
void zapis_na_pamet(unsigned char datanaflash)
{
         static unsigned int buffer_citac;          //buffer počítadlo
         static unsigned int stranka_citac;         //stránky počítadlo
         if(nazaciatok==1)        //ak zapisujeme od začiatku pamäti, nulujem počítadlá
         {        buffer_citac=0;
                  stranka_citac=0;
                  nazaciatok=0;   }
         else
         {}
         PORTB &= 0xBF;                                          //zapnem Dataflash
```

```
        zapis_cez_SPI(BUFFER_1_WRITE);
        zapis_cez_SPI(0x00);                            // don't care
        zapis_cez_SPI((char)(buffer_citac>>8));      // don't cares plus prveé dva bity adresy
        zapis_cez_SPI((char)buffer_citac);          // adresa buffera (max. 2^8 = 256 stranok)
        zapis_cez_SPI(datanaflash);                // zapíše data do SPIDR
        PORTB |= 0x40;                                          //vypnenem Dataflash
        buffer_citac++;
        if(buffer_citac > 528)
        {buffer_citac=0;
                if(stranka_citac <= 8191)
                {
                        PORTB &= 0xBF;                  // zapni DataFlash

      zapis_cez_SPI(B1_TO_MM_PAGE_PROG_WITHOUT_ERASE);// zapíš buffer 1
      zapis_cez_SPI((char)(stranka_citac>>6));
      zapis_cez_SPI((char)(stranka_citac<<2));
      zapis_cez_SPI(0x00);                                      // don't cares

      PORTB |= 0x40;                                            // vypni DataFlash
      stranka_citac++; }
                else
                { PORTB &= 0x7F;     //vypni LED
                        while (!( PINC & _BV(1) ))      //čakaj, kým nieje pustené tlacitko
                        {}
        }
        }
        else
        {}
}
void napln(void)
{
        PORTB |= 0x80;                                  // zažnem LED
        while(!(PINC & _BV(1)))                         // pokiaľ nie je pustené tlačidlo record
        {while(priznak != 1)                            //čakám na príznak
        {}
        zapis_na_pamet(ADCH);
        priznak=0;                                      //nulujem príznak
        }
        PORTB &= 0x7F;                                  //vypínam LED
}
```

```
void zmaz_flash(void)
{
for(unsigned int poc=0 ; poc <= 0x3FF ; poc++)
{
PORTB &= 0xBF;                                          // zapni DataFlash
zapis_cez_SPI(0x50);
zapis_cez_SPI((char)(poc >> 3));
zapis_cez_SPI((char)(poc << 5));
zapis_cez_SPI(0x00);
PORTB |= 0x40;                                          // vypni DataFlash
cakajnaready();
}
nazaciatok = 1;                                         //funkcia nahrávania od začiatku
PORTB &= 0x7F;                                          //vypni LED
while (!( PINC & _BV(2) ));                             //čakaj na pustenie tlačidla
}
void druhastranka_druhybuffer(unsigned char aktivnypreh_buffer, unsigned int strankapreh_citac)
{
PORTB &= 0xBF;                                          // zapni DataFlash
if (aktivnypreh_buffer == 1)                            // pokiaľ je buffer 1 aktívny
   {  zapis_cez_SPI(MM_PAGE_TO_B2_XFER);        // kopíruj stránku do buffera 2
   }
   else                          // else
   { zapis_cez_SPI(MM_PAGE_TO_B1_XFER);        // kopíruj stránku do buffera 1
   }
   zapis_cez_SPI((char)(strankapreh_citac >> 6));          //tri bajty obsahujúce zarovnanú adresu
   zapis_cez_SPI((char)(strankapreh_citac << 2));
   zapis_cez_SPI(0x00);                                 //a don't care value
PORTB |= 0x40;           // vypni DataFlash a spusti kopirovanie
cakajnaready();
}
void aktivnybuffer_prehraj(unsigned char aktivnypreh_buffer)
{  unsigned int prijate;
   unsigned int bufferreal_citac = 528;

   PORTB &= 0xBF;                                // zapni DataFlash

   if (aktivnypreh_buffer == 1)                  // pokial je aktívny buffer 1
   { zapis_cez_SPI(BUFFER_1_READ);           // čítaj z buffera 1
   }
```

```
   else                                          // inak
   { zapis_cez_SPI(BUFFER_2_READ);               // čítaj z buffera 2
   }
   zapis_cez_SPI(0x00);                  // don't care
   zapis_cez_SPI(0x00);                  // don't care
   zapis_cez_SPI(0x00);                  // čítaj od pozície 0 v bufferi
   zapis_cez_SPI(0x00);                  // don't care

   do
   {                 while(priznakPWM != 1) // pokiaľ nieje nastavený príznak, čakaj
                              {}
      zapis_cez_SPI(0xFF);                  // odošli dummy hodnotu pre prijatie bytu
      prijate = SPDR;                                    // zapíš data do premennej prijate
          priznakPWM=0;                                  // nuluj príznak
          OCR1A = prijate;                               // odošli data do PWM
    } while (bufferreal_citac--);                        // dekrementuj citac, pokial > 0, pokračuj
    PORTB |= 0x40;                                       // vypni DataFlash
}
void hraj(void)
{  unsigned int strankapreh_citac = 0;
   unsigned char aktivnypreh_buffer = 1;       // aktívny buffer = buffer1
          aktivnypreh_buffer++;                          // aktívny buffer = buffer 2

          druhastranka_druhybuffer(aktivnypreh_buffer , strankapreh_citac); // stranka 0 do buffer 1
          aktivnypreh_buffer--;                          // aktívny buffer = buffer 1

          while ((strankapreh_citac < 8191)&(!(PINC & _BV(3))))  // pokiaľ nie je koniec pamäte
                                                                 // alebo nepustí tlačidlo
    { strankapreh_citac++;                                       // zvýš čítač stránok
      druhastranka_druhybuffer(aktivnypreh_buffer , strankapreh_citac);
      aktivnybuffer_prehraj(aktivnypreh_buffer);                 // prehraj cez PWM
      if (aktivnypreh_buffer == 1)                               // pokiaľ je buffer 1 aktívny
      {
       aktivnypreh_buffer++;                                      // nastav buffer 2  aktívny
      }
      else                                                       // inak
      {
       aktivnypreh_buffer--;                                     // nastav buffer 1 aktívny
      }
   }
```

```
}
void prehraj(void)
{hraj();
PORTB &= 0x7F;      //vypni LED
}
int main()
{setup();
            while(1)
           {
      if (!(PINC & _BV(1)))                                    // ked stlačím tlačidlo PB1
      {
                              ADCSRA |= (1<<ADSC);                          //spusti prevodník
                              TIMSK = (1<<OCIE2)|(0<<TOIE2);       // spustíme prerušenia časovača
                              napln();
                              TIMSK &= ~(1<<OCIE2);                          //zastavíme prerušenia časovača
                              ADCSRA &= ~(0<<ADSC);                         //zastaví prevodník
                    }
                    else if (!(PINC & _BV(2)))                    // ked stlačíme tlačidlo PB2
      {
         PORTB |= 0x80;                                                     // zažneme LED
         zmaz_flash();
      }
                    else if (!(PINC & _BV(3)))                    // keď stlačíme tlačidlo PB3
      {
         PORTB |= 0x80;                                                     // zažneme LED
                              TCCR1B |= (1<<CS10);                  //spustíme PWM
                              TIMSK = (0<<TICIE1)|(0<<OCIE1A)|(0<<OCIE1B)|(1<<TOIE1);
         prehraj();
                              TIMSK &= ~(1<<TOIE1);
                              TCCR1B &= ~(1<<CS10);                                 //vypneme PWM
      }
                    else
                    {}                                                              //nečinný
           }
           return(0);
}
```