ZÁPADOČESKÁ UNIVERZITA V PLZNI

Fakulta elektrotechnická
Katedra elektroniky a informačních technologií

Bakalářská práce

Připojení zvukového generátoru AY-3-891x nebo YMF262 (OPL3)
k mikrokontroléru

Autor práce: Václav Sviták
Vedoucí práce: Ing. Petr Weissar, Ph.D.

Plzeň 2022

ZÁPADOČESKÁ UNIVERZITA V PLZNI

Fakulta elektrotechnická

Akademický rok: 2021/2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Václav SVITÁK**
Osobní číslo:	**E19B0116P**
Studijní program:	**B2612 Elektrotechnika a informatika**
Studijní obor:	**Elektronika a telekomunikace**
Téma práce:	**Připojení zvukového generátoru AY-3-891x nebo YMF262 (OPL3) k mikrokontroléru**
Zadávající katedra:	**Katedra elektroniky a informačních technologií**

Zásady pro vypracování

Navrhněte vhodné HW a SW řešení pro připojení obvodů zvukového generátoru AY-3-891x, YMF262 a dalších

1. Uveďte vlastnosti vybraných obvodů
2. Navrhněte vhodné výstupní obvody
3. Aplikaci řešte pro mikrokontroléry ARM CortexM
4. Přpravte alespoň jednoduchou možnost přehrávat hotové skladby ve formátech dostupných na internetu

Rozsah bakalářské práce: **30 – 40**
Rozsah grafických prací: **dle doporučení vedoucího**
Forma zpracování bakalářské práce: **elektronická**

Seznam doporučené literatury:

AY-3-8912 [online]. [cit. 2021-04-13]. Dostupné z: https://cs.wikipedia.org/wiki/AY-3-8912
Práce se zvukovým čipem AY-3-8910 (YM2149) [online]. [cit. 2021-04-13]. Dostupné z: https://www.root.cz/clanky/prace-se-zvukovym-cipem-ay-3-8910-ym2149/ *Hudební čipy Yamaha YM 3812 (OPL2) a YMF 262 (OPL3)* [online]. [cit. 2021-04-13]. Dostupné z: https://www.root.cz/clanky/hudebni-cipy-yamaha-ym-3812-opl2-a-ymf-262-opl3/

Vedoucí bakalářské práce: **Ing. Petr Weissar, Ph.D.**
Katedra elektroniky a informačních technologií

Datum zadání bakalářské práce: **8. října 2021**
Termín odevzdání bakalářské práce: **26. května 2022**

L.S.

Západočeská univerzita v Plzni
Fakulta elektrotechnická

Prof. Ing. Zdeněk Peroutka, Ph.D.
děkan

Doc. Ing. Jiří Hammerbauer, Ph.D.
vedoucí katedry

V Plzni dne 8. října 2021

Abstrakt

V práci je řešena problematika zvukových generátorů, které se používaly v 8bitových a 16bitových počítačích a jejich řízení moderním mikroprocesorem. Cílem bylo navrhnout vhodné HW a SW řešení pro připojení libovolného programovatelného zvukového generátoru z řady AY-3-891x nebo YAMAHA OPLx. Dalším úkolem bylo přehrávat zvukové soubory na některém z výše uvedených generátorů.

Jelikož většina materiálů a dokumentace je v angličtině, tak hlavní motivace k vytvoření této práce je připravit manuál v češtině, kde budou uvedené vlastnosti obvodů, princip jejich funkce a popis řízení. V práci jsou podrobně vysvětleny registry a bloky, které se podílí na generování zvuku. Dále jsou popsány zvukové formáty k oběma generátorům.

Praktická realizace proběhla na generátoru AY-3-8912, který byl ovládán mikrokontrolérem STM32F411RE. Pro přehrávání hudby byl zvolen formát YM a pro připojení k reproduktoru byl navržen předzesilovač.

Výsledkem práce je program, který umožňuje řídit jednotlivé části AY-3-8912 a přehrávat hudbu.

Klíčová slova

Yamaha OPL, AY-3-8910, AY-3-8912, Programovatelné zvukové generátory, FM syntéza, přehrávání souboru YM

Abstract

Sviták, Václav. *Connecting sound generator chip AY-3-891x or YMF262 (OPL3) to microcontroller* [*Připojení zvukového generátoru AY-3-891x nebo YMF262 (OPL3) k mikrokontroléru*]. Pilsen, 2022. Bachelor thesis (in Czech). University of West Bohemia. Faculty of Electrical Engineering. Department of Electronics and Information Technologies. Supervisor: Petr Weissar

---

The paper deals with the problem of sound generators used in 8-bit and 16-bit computers and their control by a modern microprocessor. The aim was to design a suitable HW and SW solution to connect any programmable sound generator from the AY-3-891x or YAMAHA OPLx series. The next task was to play the sound files on any of the above generators.

Since most of the material and documentation is in English, the main motivation for this work is to create a manual in Czech that lists the features of the circuits, the principle of their operation and a description of the control. The registers and blocks involved in sound generation are explained in detail in this thesis. Furthermore, the sound formats to both sound generators are described.

The practical implementation was carried out on the AY-3-8912 sound generator, which was controlled by the STM32F411RE microcontroller. The YM format was chosen for music playback and a preamplifier was designed for connection to the speaker.

The result of the work is a program that allows to control the different parts of the AY-3-8912 and play music.

Keywords

Yamaha OPL, AY-3-8910, AY-3-8912, Programmable sound generators, FM synthesis, YM file playback

Prohlášení

Předkládám tímto k posouzení a obhajobě bakalářskou práci, zpracovanou na závěr studia na Fakultě elektrotechnické Západočeské univerzity v Plzni.

Prohlašuji, že jsem svou závěrečnou práci vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 270 trestního zákona č. 40/2009 Sb.

Také prohlašuji, že veškerý software, použitý při řešení této bakalářské práce, je legální.

V Plzni dne 26. května 2022

Václav Sviták

.....

Podpis

Obsah

Seznam obrázků

Seznam tabulek

Seznam symbolů a zkratek

PSG	Programmable Sound Generator. Programovatelný zvukový generátor.
FPGA	Field-Programmable Gate Array. Programovatelné hradlové pole.
VHDL	VHSIC Hardware Description Language. Programovací jazyk pro popis hardwaru.
CLK	Clock. Synchronizační hodinové pulzy.
TTL	Transistor-Transistor Logic. Tranzistorově tranzistorová logika.
D/A	Digital/Analog. Digitál/Analog.
CPU	Central Processing Unit. Procesorová jednotka.
SPL	Sound Pressure Level. Hladina akustického tlaku.
I/O	Input/Output. Vstup/Výstup.
PCB	Printed Circuit Board. Deska plošných spojů.
PWM	Pulse Width Modulation. Pulzně šířková modulace.
PC	Personal Computer. Osobní počítač.
FM	Frequency Modulation. Frekvenční modulace.
ROM	Read-Only Memory. Pamět určená pouze pro čtení.
IRQ	Interrupt ReQuest. Výzva k přerušení probíhajícího procesu za účelem provedení důležitější akce.
ADSR	Attack Decay Sustain Release. Typ obálkového generátoru.
TDMA	Time Division Multiple Access. Metoda přístupu více uživatelů ke společnému médiu.
SRAM	Static Random Access Memory. Statická verze paměti s náhodným přistupem.
OPL	OPerator Type-L. Typ operátoru.
PCM	Pulse-Code Modulation. Pulzní kódová modulace.
LHA	Kompresní nástroj a formát souboru.
THT	Through-Hole Technology. Způsob montáže součástek na PCB.
AM	Amplitude Modulation. Amplitudová modulace.
MIDI	Musical Instrument Digital Interface. Prokol pro komunikaci mezi hudebním nástrojem a počítačem.

1

Úvod

Bakalářská práce se popisuje dva typy programovatelných zvukových generátorů. Budu se zabývat dvěma funkčně odlišnými generátory. Cílem práce je ovládat vybraný zvukový generátor pomocí mikrokontroléru, přehrávat pomocí něj hudební skladby a navrhnout vhodný výstupní obvod pro reproduktor. STM32F411RE bude mnou zvolený mikrokontrolér. Zvukový generátor, na kterém proběhne praktická realizace bude AY-3-8912.

V kapitole 2 bude popsán význam a důvod použití programovatelných zvukových generátorů v počítačích. Kapitola 3 se zabývá popisem jednotlivých verzí AY-3-891x. Detailní vysvětlení funkce bude popsána na AY-3-8910. U každého typu budou uvedeny odlišnosti od předchozích verzí a pár příkladů praktického použití v počítačích nebo zvukových kartách. Poté jsou popsány verze hudební formátu YM. V kapitole 4 je popsáno řízení AY-3-8912 pomocí mikrokontroléru. Zde jsou popsány jednotlivé funkce, které se podílejí na řízení. Kapitola 5 se zabývá přehráváním formátu YM. Zde je popsán převod na neprokládaný formát. Dále je uvedeno řešení výstupních obvodů pro připojení k reproduktoru.

V kapitole 6 proběhne popis jednotlivých verzí OPLx. Na začátku kapitoly je vysvětleno, proč se používaly jako náhrada za PC Speaker a jaké to mělo výhody. U každého typu budou uvedeny odlišnosti od předchozích verzí a pár příkladů praktického použití v počítačích nebo zvukových kartách. Detailní popis funkce a registrů proběhne na OPL. U následných verzí budou uvedeny nové registry a funkce.

2

Programovatelné zvukové generátory

V dobách, kdy na trhu byly osmibitové domácí počítače a herních konzole se ke generování zvuku nepoužívaly procesory. Tehdejší procesory neměly dostatečný výkon k tomu, aby současně dokázaly vykreslovat postavy a prostředí ve hře, snímat stisknutá tlačítka a starat se o generování zvuku. Navíc složitější zvuky zabíraly více místa na operační paměti. Například pro kanál, na kterém se bude přehrávat tón o frekvenci 8372 Hz, je třeba, aby se procesor věnoval generování tónu každých šedesát mikrosekund.[17]. Proto se používaly programovatelné zvukové generátory (PSG). Procesor počítače se pouze staral o to, aby se v jasně daných časových rámcích (např. 50x za sekundu) měnily data řídících registrů v PSG. Ten se pak postaral o generovaní zvuku. Posílání dat na sběrnici PSG je výkonově nenáročná a rychlá operace. Nejpoužívanější PSG byly: SID a PAULA od firmy Commodore, AY-3–891x od firmy General Instruments, POKEY od Atari a nakonec OPLx od firmy Yamaha. Každý z těchto generátorů má odlišný způsob, kterým vytváří hudbu. Také výsledný generovaný zvuk má jiné zabarvení na každém z uvedených generátorů.

3

Popis a vlastnosti obvodů AY-3-891x

Obvody AY-3-891x jsou tříkanálové programovatelné zvukové generátory navržené firmou General Instruments v roce 1978. Původně byly určené pro 16bitový mikropočítač CP1600. Své uplatnění našly v herních konzolích, jako je Amstrad CPC, Colour Genie nebo ZX Spectrum. Byly také použity na externí zvukové kartě Mockingboard k počítači Apple II. V roce 1987 převzala práva na výrobu od General Instruments firma Microchip Technology, která vyrobila vylepšenou verzi zvukového generátoru. Licenčně generátor také vyráběla společnost Yamaha jako YM2149F. Ten měl stejné rozložení vývodů jako AY-3-8910, ale uživatel měl možnost nastavit externě dělič hodinových pulzů. YM2149F se používal v počítači Atari ST. Dnes už se obvody AY-3–891x nevyrábí, ale je možné koupit jejich klony. Lze si vytvořit vlastní AY-3-891x na FPGA pomocí jazyku VHDL.

Zvukový generátor AY-3-891x byl vyráběn ve variantách:

1. AY-3-8910 (YM2149) [3][9][11][12]
2. AY-3-8912[4][7]
3. AY-3-8913[5][7]
4. AY-3-8914[6][13]
5. AY-3-8916[7]
6. AY-3-8917[8]
7. AY-3-8930[10]

Jednotlivé typy se od sebe lišily pouze typem pouzdra, počtem vývodů a počtem osmibitových vstupně/výstupních bran. Princip řízení a generování zvuku je u každé varianty stejný.

3.1 AY-3-8910

AY-3-8910 je ve 40 pinovém pouzdru. Má k dispozici dva univerzální 8bitové paralelní vstupně/výstupní porty A a B. Tato verze PSG je jedna z nejvíce rozšířených a byla velmi často používána. Verze 8910 byla použita v raných verzích zvukové karty Mockingboard a v domácích počítačích (MSX, Oric 1, Coulour Genie a Elektor TV Games Computer).[3] [11] [12]

AY-3-8910 obsahuje šestnáct 8bitových registrů. Do nich se data dostávají přes 8bitovou sběrnici. Předpokládejme, že sběrnice je ve stavu vysoké impedance, když chceme vybrat a zapsat hodnotu do registru, tak nejprve musíme sběrnici uvést do stavu, kdy jí chceme předat adresu vybraného registru. Poté zapíšeme adresu registru na sběrnici. Tím je vybrán registr, do kterého chceme zapsat data. Následně přepneme sběrnici do stavu pro zápis dat do PSG. Na sběrnici pošleme data, která se mají zapsat. Poté sběrnici uvedeme do stavu vysoké impedance, vynulujeme piny DA7 - DA0 a další zápisový cyklus může začít. Po zápisu do registrů PSG generuje tóny, takže se procesor může věnovat jiným činnostem. Každý z šestnácti registrů lze číst, takže mikroprocesor může zjistit hodnoty uložených dat.

3.1.1 Popis pinů

DA0 - DA7

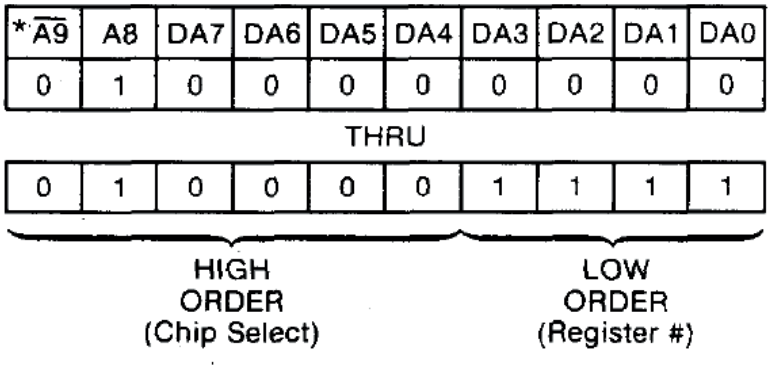
Piny lze provozovat v režimu: vstup, výstup a stav vysoké impedance.

Těchto 8 pinů tvoří 8bitovou obousměrnou sběrnici, kterou mikroprocesor používá k odesílání dat a adres registrů do PSG nebo k příjmu dat z PSG. V adresovém režimu (odesílání z mikroprocesoru do PSG) DA3 - DA0 volí adresu registru (R0 až R15) a DA7 - DA4 ve spojení s piny A9 a A8 tvoří chipselect. Nesprávné nastavení DA7 - DA4 uvede sběrnici do stavu vysoké impedance, protože nedojde k připojení třístavových hradel na vstupy registrů. Z výroby jsou tyto vstupy konfigurovány tak, aby při hodnotě 0000 na DA7 - DA4 došlo k připojení třístavových hradel. Vyráběly se ale i čipy, které měly nastavenou jinou hodnotu DA7 - DA4 k výběru registru (např. 0110), ale ty byly méně časté. Díky tomu bylo možné řídit dva PSG na jedné sběrnici.

A8, A9

Piny lze provozovat v režimu: vstup.

A9 musí být nastaven tak, aby na něm byla trvale log. 0 a na A8 musí být log. 1, jinak vstupy některých registrů nebudou připojeny na třístavová hradla. Lze je ponechat i nezapojené, protože A9 je vybaven pull-down rezistorem a A8 pull-up rezistorem. V prostředí, kde se vyskytuje rušení se doporučuje, aby byl A9 uzemněn a A8 připojen na +5 V.



Obr. 3.1: Příklad nastavení sběrnice pro výběr registru R15. |Převzato z [17]|

BDIR, BC2, BC1

Piny lze provozovat v režimu: vstup.

K ovládání sběrnice PSG se používají vstupy Bus DIRection (BDIR), Bus Control 2 (BC2) a Bus Control 1 (BC1).

Tab. 3.1: Funkce sběrnice pro jednotlivé kombinace na vstupech BDIR, BC2 a BC1.

BC2	BDIR	BC1	Funkce sběrnice
1	0	0	Stav vysoké impedance
1	0	1	Čtení dat z PSG
1	1	0	Zápis dat do PSG
1	1	1	Výběr registru pro zápis dat

Z tabulky vyplývá, že pin BC2 může být trvale připojen na +5 V.

RESET

Pin lze provozovat v režimu: vstup.

Přivedením log. 0 na pin RESET se všechny registry nastaví na hodnotu 0. Pin je vybaven pull-up rezistorem, který je integrován na čipu.

CLOCK

Pin lze provozovat v režimu: vstup.

Pin CLOCK dodává časovou referenci pro generátory tónu, šumu a obálky. Hodinové pulzy musí být kompatibilní s TTL. To znamená, že na vstupu se musí pohybovat úroveň log. 0 v rozmezí 0 až 0,8 V a log. 1 od 2 do 5 V.

Analog Channel A, B, C

Piny lze provozovat v režimu: výstup.

Každý kanál obsahuje D/A převodník, který podle obsahu registrů vytvoří odpovídající průběh signálu.

Input/Output A7 - A0, B7 - B0

Piny lze provozovat v režimu: vstup a výstup.

Každý z těchto dvou paralelních vstupních/výstupních portů umožnuje přenos dat do PSG z externích zařízení připojených k pinům IOA nebo IOB a naopak. Každý pin je vybaven pull-up rezistorem, takže v režimu vstup budou všechny piny ve stavu log. 1.

No Connect (TEST 1, TEST 2)

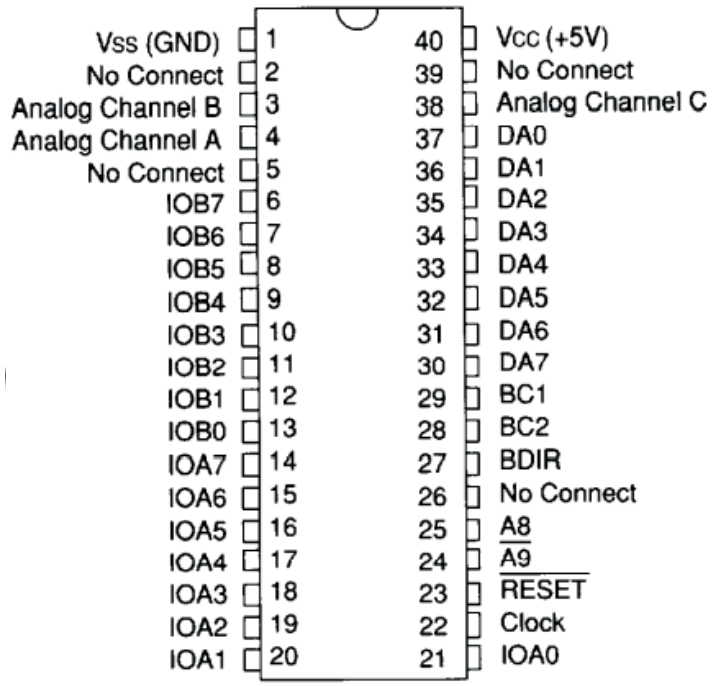
Tyto piny jsou určeny pouze testování a měly by zůstat nezapojené.

Vcc

Připojení napájení +5 V.

Vss

Připojení GND pro PSG.



Obr. 3.2: Označení vývodů na AY-3-8910. |Převzato z [1]|

3.1.2 Bloky pro generování zvuku

Generátory tónů

Vytvářejí obdélníkový signál o zadané frekvenci pro každý kanál (A, B, C).

Generátor šumu

Produkuje frekvenčně modulovaný pseudonáhodný obdélníkový signál.

Směšovače

Slučují výstupy tónových generátorů a generátoru šumu.

Generátor obálky

Produkuje obálku, kterou lze použít k amplitudové modulaci výstupu směšovače.

Řízení amplitudy

Ovlivňuje amplitudu signálu na výstupu D/A převodníků. Lze nastavit konstantní nebo proměnnou amplitudu signálu. Konstantní amplituda se nastavuje přímo mikroprocesorem, proměnnou amplitudu nastavuje generátor obálky.

D/A převodníky

Každý ze tří D/A převodníků vytváří až 16 úrovní výstupního signálu.

3.1.3 Popis registrů

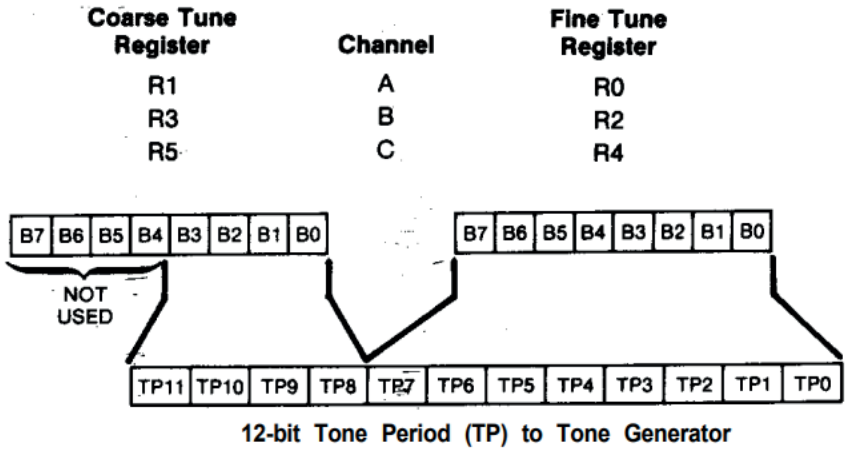
Osmibitové registry R0, R2 a R4 slouží k jemnému nastavení frekvence tónu pro daný kanál. Čtyřbitové registry R1, R3 a R5 slouží ke hrubému nastavení frekvence tónu pro daný kanál. Pětibitový registr R6 ovládá frekvenci šumu. O řízení orientace vstupně/výstupních portů, povolení výstupu tónového a šumového generátoru na daný kanál se stará registr R6. Pomocí registrů R8, R9 a R10 lze nastavit konstantní nebo proměnnou amplitudu pro příslušný kanál. Osmibitové registry R11 a R12 slouží k jemnému nastavení frekvence obálky (R11) a hrubému nastavení (R12). Tvar obálky lze nastavit pomocí registru R13. Pro zápis a čtení ze vstupně/výstupních portů slouží registry R14 a R15.

Tab. 3.2: Mapa registrů AY-3-8910.

REGISTER		BIT B7	BIT B6	BIT B5	BIT B4	BIT B3	BIT B2	BIT B1	BIT B0
R0	Channel A Tone Period	8-BIT Fine Tune A							
R1						4-BIT Coarse Tune A			
R2	Channel B Tone Period	8-BIT Fine Tune B							
R3						4-BIT Coarse Tune B			
R4	Channel C Tone Period	8-BIT Fine Tune C							
R5						4-BIT Coarse Tune C			
R6	Noise Period				5-BIT Period Control				
R7	Enable	IN/$\overline{\text{OUT}}$		$\overline{\text{Noise}}$			$\overline{\text{Tone}}$		
		IOB	IOA	C	B	A	C	B	A
R8	Channel A Amplitude				M	L3	L2	L1	L0
R9	Channel B Amplitude				M	L3	L2	L1	L0
R10	Channel C Amplitude				M	L3	L2	L1	L0
R11	Envelope Period	8-BIT Fine Tune Envelope							
R12		8-BIT Coarse Tune Envelope							
R13	Envelope Shape/Cycle					Continue	Attack	Alternate	Hold
R14	I/O Port A Data Store	8-BIT PARALLEL I/O on Port A							
R15	I/O Port B Data Store	8-BIT PARALLEL I/O on Port B							

3.1.4 Řízení tónového generátoru

Frekvence obdélníkového signálu, který je generován třemi tónovými generátory (jeden pro každý kanál) se získá tak, že se hodinové pulzy (CLK) nejprve vydělí 16 a výsledek se dále vydělí naprogramovanou 12bitovou hodnotou periody tónu. 12bitová hodnota se získá kombinací registrů pro hrubé a jemné nastavení.



Obr. 3.3: Vytvoření 12bitové hodnoty kombinací registrů. |Převzato z [17]|

Čím vyšší hodnotu zapíšeme do registrů hrubého a jemného ladění, tím nižší je výsledná frekvence tónu. Výše uvedené poznatky se dají zapsat matematicky jako:

$$f_P = \frac{f_{CLK}}{16 \cdot TP_{10}} \tag{3.1}$$

kde f_P je námi požadovaná frekvence tónu v Hz, f_{CLK} je frekvence hodinových pulzů přivedená na pin CLOCK v Hz, TP_{10} je 12bitová hodnota převedená do desítkové soustavy. Tento vztah se dá převést do praktičtější podoby:

$$TP_{10} = \frac{f_{CLK}}{16 \cdot f_P} \tag{3.2}$$

Pokud převedeme do dvojkové soustavy hodnotu TP_{10} a oddělíme horní čtyři bity od osmi dolních, tak můžeme do registru pro hrubé ladění zapsat horní čtyři bity a do registru pro jemné ladění zapsat dolních osm bitů. Tím získáme na výstupu námi zvoleného kanálu požadovanou frekvenci tónu.

Příklad: $f_{CLK} = 2$ MHz, $f_P = 440$ Hz, výstup: kanál B

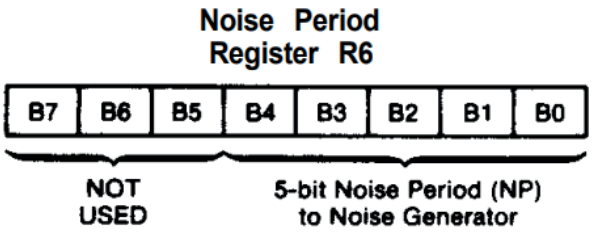
$$TP_{10} = \frac{f_{CLK}}{16 \cdot f_P} = \frac{2000000}{16 \cdot 440} = 284_{10} \tag{3.3}$$

$$284_{10} = \left[\underbrace{0001}_{\text{registr R3}} \; \underbrace{0001\,1100}_{\text{registr R2}} \right]_2 \tag{3.4}$$

Maximální frekvence tónu, kterou můžeme na výstupu získat je určena f_{CLK} a hodnotou TP_{10}, která může být od 1_{10} do 4095_{10}. Když krajní hodnoty TP_{10} dosadíme do vztahu 3.1 a $f_{CLK} = 2$ Mhz, tak frekvence na výstupu se může pohybovat od 30,5 Hz do 125 kHz. Pokud například bude $f_{CLK} = 1$ MHz, tak budeme mít frekvenční rozsah od 15,3 Hz do 62500 Hz.

3.1.5 Řízení generátoru šumu

Výstupní frekvence generátoru šumu se získá vydělením hodin (CLK) číslem 16 a následným vydělením výsledku naprogramovanou 5bitovou hodnotou periody šumu. Tato 5bitová hodnota se skládá z dolních 5 bitů (B4 - B0) registru R6.



Obr. 3.4: Vytvoření 5bitové hodnoty. |Převzato z [17]|

V registru R6 je uložena hodnota periody. Vyšší hodnota v registru znamená nižší výslednou frekvenci šumu. Pro frekvenci šumu platí vztah:

$$f_P = \frac{f_{CLK}}{16 \cdot NP_{10}} \tag{3.5}$$

kde f_P je námi požadovaná frekvence šumu v Hz, f_{CLK} je frekvence hodinových pulzů přivedená na pin CLOCK v Hz, NP_{10} je 5bitová hodnota periody v desítkové soustavě. Ta nabývá hodnot od 1_{10} do 31_{10}. Pokud dosadíme tyto krajní hodnoty do 3.5 a zvolíme $f_{CLK} = 2$ MHz bude se frekvence šumu na výstupu pohybovat od 4 kHz do 125 kHz. Pokud snížíme frekvenci f_{CLK} na 1 MHz, tak budeme mít frekvenční rozsah od 2016,5 Hz do 62500 Hz.

Pokud chceme znát hodnotu, kterou máme zadat do registru R6 při námi zvolené f_P, tak vztah upravíme na:

$$NP_{10} = \frac{f_{CLK}}{16 \cdot f_P} \tag{3.6}$$

Příklad: $f_{CLK} = 2$ MHz, $f_P = 5$ kHz

$$NP_{10} = \frac{f_{CLK}}{16 \cdot f_P} = \frac{2000000}{16 \cdot 5000} = 25_{10} \tag{3.7}$$

$$25_{10} = \left[000 \ \underbrace{1\,1100}_{\text{registr R6}} \right]_2 \tag{3.8}$$

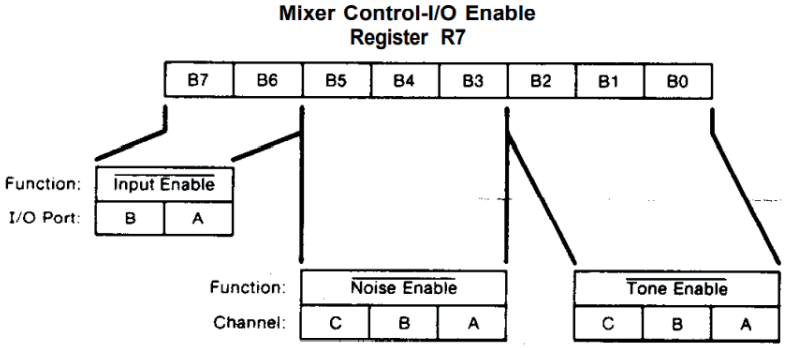
3.1.6 Řízení směšovače

R7 je registr, který připojuje výstupy generátoru šumu a tónu na jednotlivé kanály. Lze jím také ovládat orientaci portů A a B.

Bity B2 - B0 ovládají připojení tónových generátorů. Např. pokud chceme mít na kanálu B námi zvolený tón, tak na pozici B1 nastavíme 0. Když nechceme využívat kanály A a C, tak na pozice B0 a B3 nastavíme 1. Nastavením na 1 nezajistíme to, že kanál bude vypnutý. To umožňuje registr pro řízení amplitudy.

Bity B5 - B3 ovládají připojení generátoru šumu. Způsob ovládání je stejný jako u bitů B2 - B0. Natavením nuly připojíme generátor na určený kanál, nastavením 1 kanál od generátoru odpojíme.

Bity B7 a B6 ovládají orientaci portů. Nastavením 0 se port bude chovat jako vstup, když nastavíme 1, tak se bude chovat jako výstup.



Obr. 3.5: Popis jednotlivých bitů registru R7. |Převzato z [17]|

Noise Enable Truth Table:

R7 Bits B5	B4	B3	Noise Enabled on Channel		
0	0	0	C	B	A
0	0	1	C	B	—
0	1	0	C	—	A
0	1	1	C	—	—
1	0	0	—	B	A
1	0	1	—	B	—
1	1	0	—	—	A
1	1	1	—	—	—

Tone Enable Truth Table:

R7 Bits B2	B1	B0	Tone Enabled on Channel		
0	0	0	C	B	A
0	0	1	C	B	—
0	1	0	C	—	A
0	1	1	C	—	—
1	0	0	—	B	A
1	0	1	—	B	—
1	1	0	—	—	A
1	1	1	—	—	—

I/O Port Truth Table:

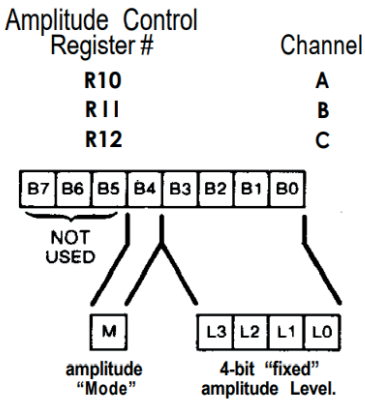
R7 Bits B7	B6	I/O Port Status IOB	IOA
0	0	Input	Input
0	1	Input	Output
1	0	Output	Input
1	1	Output	Output

Obr. 3.6: Shrnutí platných kombinací pro ovládání registru R7. |Převzato z [17]|

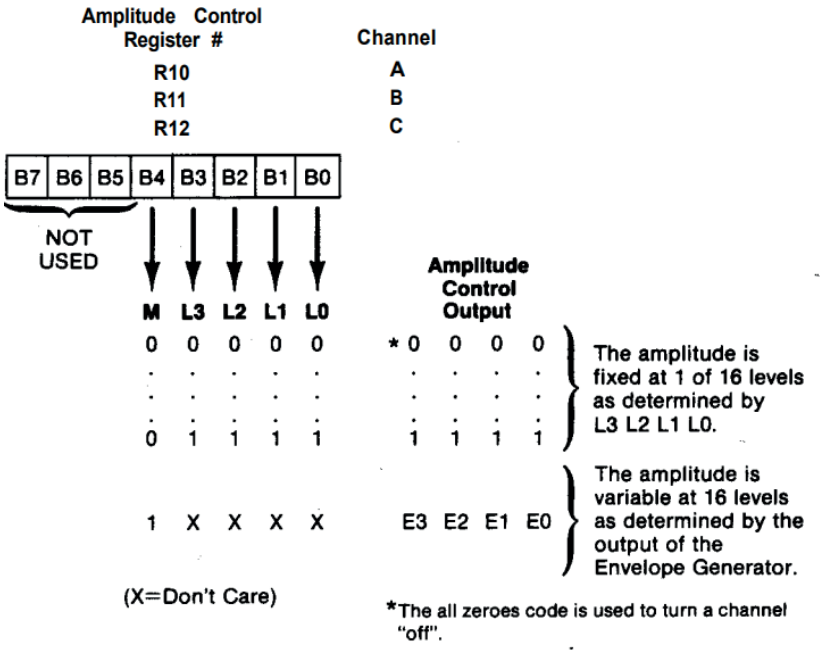
3.1.7 Řízení amplitudy

Amplitudy signálů, které jsou generovány D/A převodníky (pro každý kanál je jeden) jsou určeny pomocí spodních pěti bitů (B4 - B0) registrů R10, R11 a R12.

Pomocí bitu M se volí buď konstantní (M = 0) nebo proměnná amplituda (M = 1). Bity L3 - L0, kterými se určuje hodnota konstantní amplitudy, jsou aktivní pouze při M = 0. Při M = 1 jsou ignorovány. V konstantním režimu je amplituda přímo nastavována mikroprocesorem přes sběrnici. Lze si vybrat ze 16 různých úrovní amplitudy. Pokud bude B3 - B0 nastaveno na 0, tak daný kanál bude vypnutý. V proměnném režimu je amplituda určena frekvencí a typem obálky. To zajištují bity E3 - E0, které jdou z výstupu generátoru obálky.



Obr. 3.7: Popis jednotlivých bitů. |Převzato z [17]|



Obr. 3.8: Detailní popis kombinací pro ovládání amplitudy. |Převzato z [17]|

3.1.8 Řízení generátoru obálky

Pro generování obálky, jsou k dispozici dva způsoby řízení. Prvním je možné měnit frekvenci (periodu) obálky pomocí registrů R13 a R14. Druhým je změna tvaru a cyklu obálky pomocí registru R15.

Frekvence obálky se získá tak, že se hodiny nejprve vydělí 256 a výsledek se dále vydělí naprogramovanou 16bitovou hodnotou periody obálky (EP_{10}). Tato 16bitová hodnota se získá kombinací obsahu registrů pro jemné (R13) a hrubé (R14) nastavení periody obálky. Vyšší hodnota v registru znamená nižší výslednou frekvenci obálky. Vzorec pro výpočet požadované frekvence obálky je:

$$f_P = \frac{f_{CLK}}{256 \cdot EP_{10}} \tag{3.9}$$

kde f_P je námi požadovaná frekvence šumu v Hz, f_{CLK} je frekvence hodinových pulzů přivedená na pin CLOCK v Hz, EP_{10} je 16bitová hodnota periody v desítkové soustavě. Ta nabývá hodnot od 1_{10} do 65536_{10}. Pokud dosadíme tyto krajní hodnoty do 3.9 a zvolíme f_{CLK} = 2 MHz bude se frekvence obálky pohybovat od 0,12 Hz do 7812,5 Hz. Pokud snížíme frekvenci f_{CLK} na 1 MHz, tak budeme mít frekvenční rozsah od 0,06 Hz do 3906 Hz.

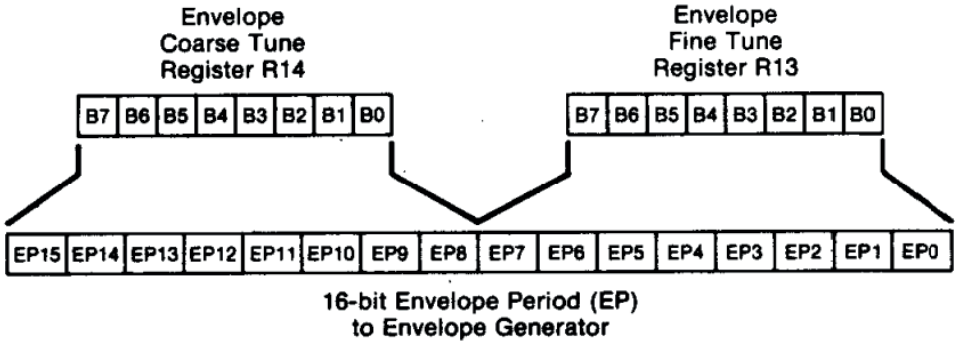
Pokud chceme znát hodnotu, kterou máme zadat do registrů R13 a R14 při námi zvolené f_P, tak vztah upravíme na:

$$EP_{10} = \frac{f_{CLK}}{256 \cdot f_P} \tag{3.10}$$

Příklad: f_{CLK} = 2 MHz, f_P = 15 Hz

$$EP_{10} = \frac{f_{CLK}}{256 \cdot f_P} = \frac{2000000}{256 \cdot 15} = 520_{10} \tag{3.11}$$

$$520_{10} = \left[\underbrace{0000\,0010}_{\text{registr R14}} \; \underbrace{0000\,1000}_{\text{registr R15}} \right]_2 \tag{3.12}$$



Obr. 3.9: Vznik 16bitové hodnoty. |Převzato z [17]|

Obálkový čítač dále dělí periodu obálky 16, čímž vytváří 16 stavový vzor obálky na cyklus. Stavy jsou definovány výstupem obálkového čítače, který generuje bity E3 - E0. Konkrétního tvaru a vzoru cyklu požadované obálky se dosáhne řízením počítání obálkového čítače. Řízení tvaru obálky/cyklu je obsaženo ve spodních 4 bitech (B3 - B0) registru R15. Každý z těchto 4 bitů řídí určitou funkci generátoru obálky.

Hold (B0)

Pokud je tato hodnota nastavena na log. 1, je obálka omezena na jeden cyklus. Napočítaná hodnota obálkového čítače se na konci cyklu uloží. Ta může být 0000 (3.13) nebo 1111 (3.14). To záleží na tom, jestli čítač počítal dolů nebo nahoru. V log. 0 není obálka omezena na jeden cyklus.

$$\underbrace{0}_{\text{E3}}\ \underbrace{0}_{\text{E2}}\ \underbrace{0}_{\text{E1}}\ \underbrace{0}_{\text{E0}} \tag{3.13}$$

$$\underbrace{1}_{\text{E3}}\ \underbrace{1}_{\text{E2}}\ \underbrace{1}_{\text{E1}}\ \underbrace{1}_{\text{E0}} \tag{3.14}$$

Alternate (B1)

Když je nastaven na log. 1, tak obálkový čítač po každém cyklu obrátí směr počítání (nahoru/dolů). V log. 0 je směr počítání určen bitem Attack. Pokud jsou bity Hold i Alternate v log. 1, obálkový čítač se po jednom cyklu vynuluje na počáteční hodnotu.

Attack (B2)

Když je nastaven na log. 1, obálkový čítač se začne počítat nahoru:

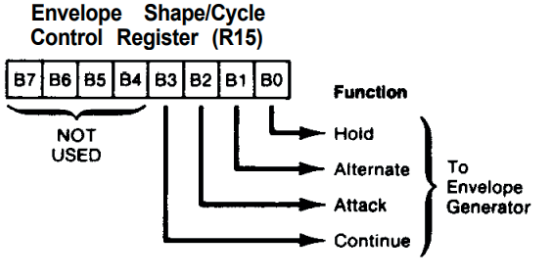
$$\underbrace{0}_{\text{E3}}\ \underbrace{0}_{\text{E2}}\ \underbrace{0}_{\text{E1}}\ \underbrace{0}_{\text{E0}} \longrightarrow \underbrace{1}_{\text{E3}}\ \underbrace{1}_{\text{E2}}\ \underbrace{1}_{\text{E1}}\ \underbrace{1}_{\text{E0}} \tag{3.15}$$

Při nastavení na log. 0 bude čítač počítat dolů:

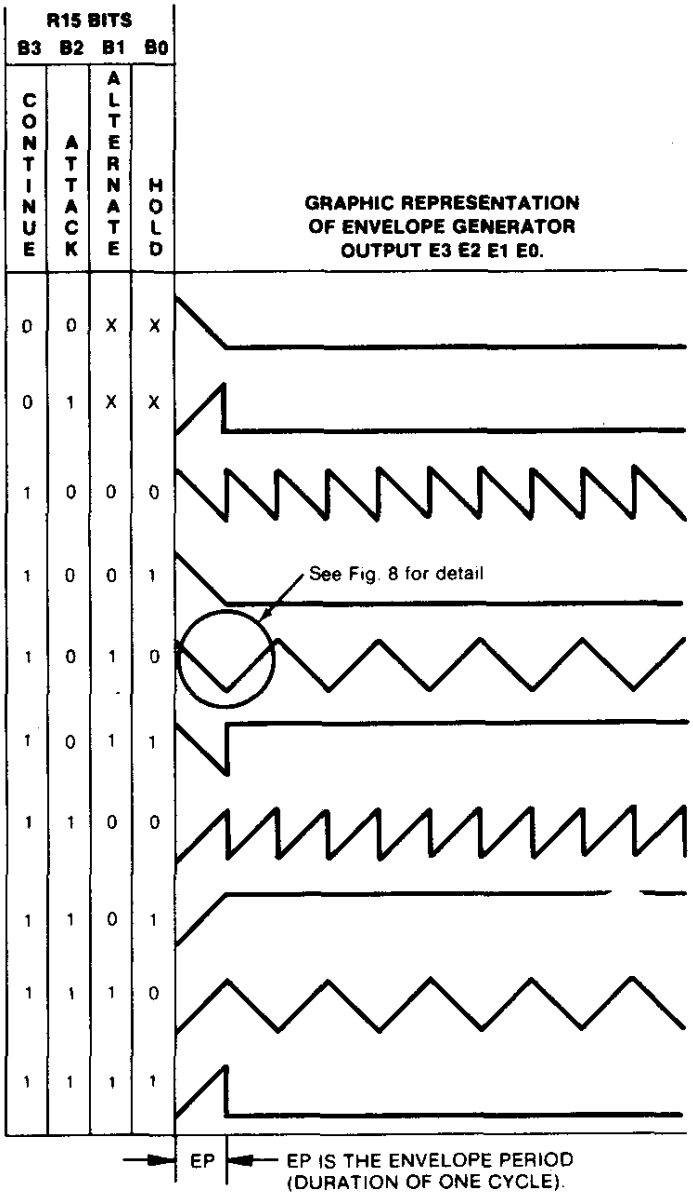
$$\underbrace{1}_{\text{E3}}\ \underbrace{1}_{\text{E2}}\ \underbrace{1}_{\text{E1}}\ \underbrace{1}_{\text{E0}} \longrightarrow \underbrace{0}_{\text{E3}}\ \underbrace{0}_{\text{E2}}\ \underbrace{0}_{\text{E1}}\ \underbrace{0}_{\text{E0}} \tag{3.16}$$

Continue (B3)

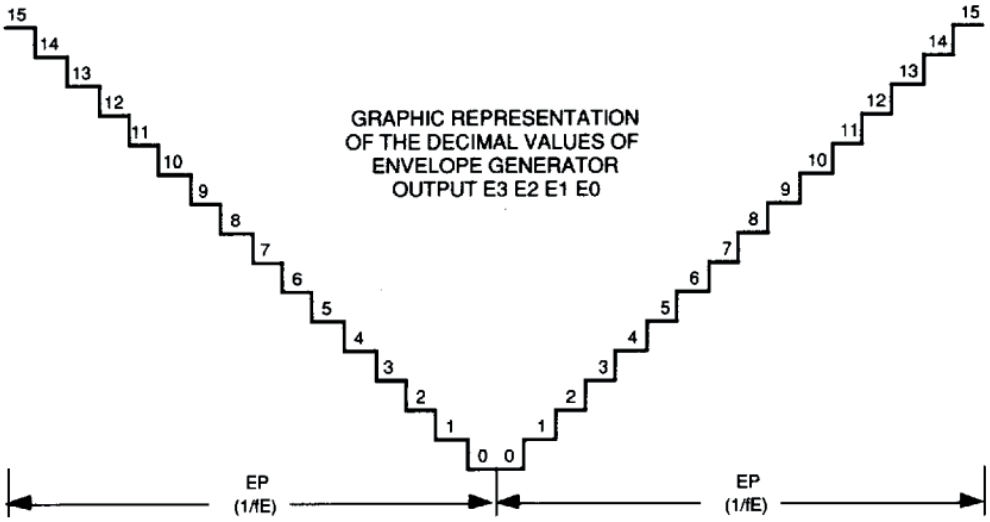
Pokud je nastaven na log. 1, cyklus bude probíhat tak, jak je definováno bitem Hold. Pokud je nastaven na log. 0, obálkový čítač se po jednom cyklu vynuluje na 0000 (E3 - E0) a udrží se na této hodnotě.



Obr. 3.10: Vysvětlení jednotlivých bitů registru R15. |Převzato z [1]|



Obr. 3.11: Přehled všech osmi druhů obálek a hodnoty jednotlivých bitů pro jejich nastavení. |Převzato z [1]|



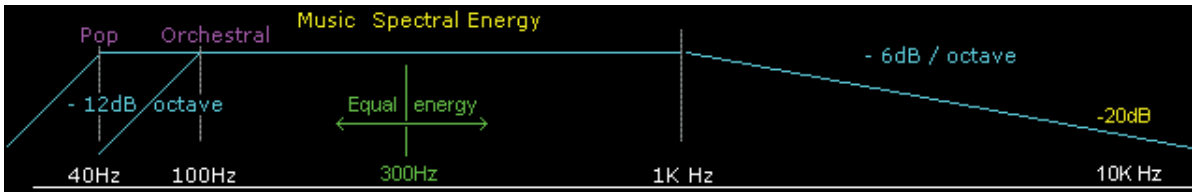
Obr. 3.12: Detailní pohled na obálku pro kombinaci 1010. Hodnoty 0 až 15 jdou z výstupu obálkového čítače na vstup D/A převodníku. |Převzato z [1]|

3.1.9 D/A převodník

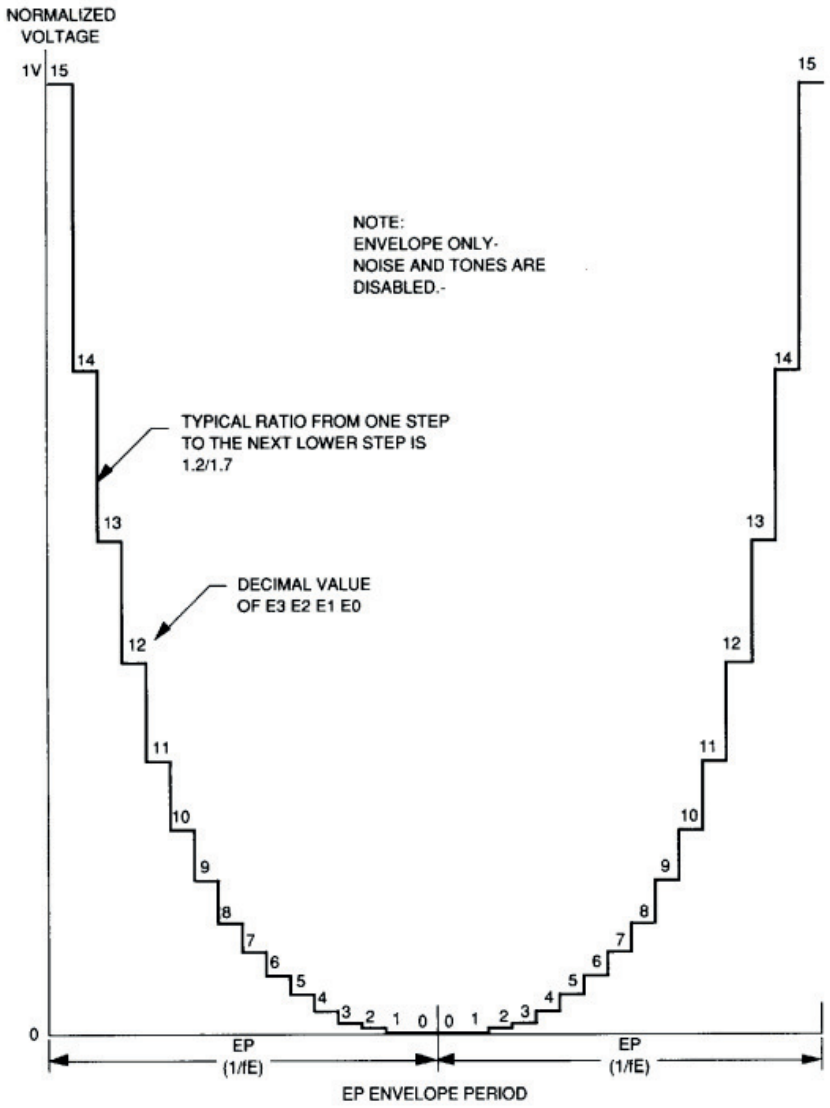
Poslední blok, který se podílí na generování zvuku je D/A převodník. Každý kanál má k dispozici jeden. Závislost hladiny akustického tlaku (SPL) na frekvenci není u lidského ucha lineární. Abychom dobře slyšeli nízké frekvence je potřeba velký akustický tlak. To znamená, že v tomto pásmu je lidské ucho málo citlivé. Nejcitlivější je ucho od cca 1 kHz do 5 kHz, kde stačí malé hladiny akustického tlaku. Se zvyšující se frekvencí se stává ucho méně citlivým a je třeba větší akustický tlak. Z toho důvodu výstupní napětí D/A převodníku odpovídá logaritmické stupnici. Nízkým frekvencím bude odpovídat vyšší napětí. Poté bude generované napětí cca konstantní.

Pro frekvence na 10 kHz bychom opět potřebovali zvýšit napětí na výstupu převodníku. To ale není třeba, protože v hudebním spektru je nejvíce energie (výkonu) soustředěno od 40 Hz do 2 kHz. Poté energie klesá se strmostí 6 dB/oktávu. Z toho plyne, že jen velmi malé množství energie je obsaženo ve frekvencích nad 10 kHz. Proto necitlivost ucha od této frekvence má malý vliv na celkový dojem z hudby generované tímto PSG.

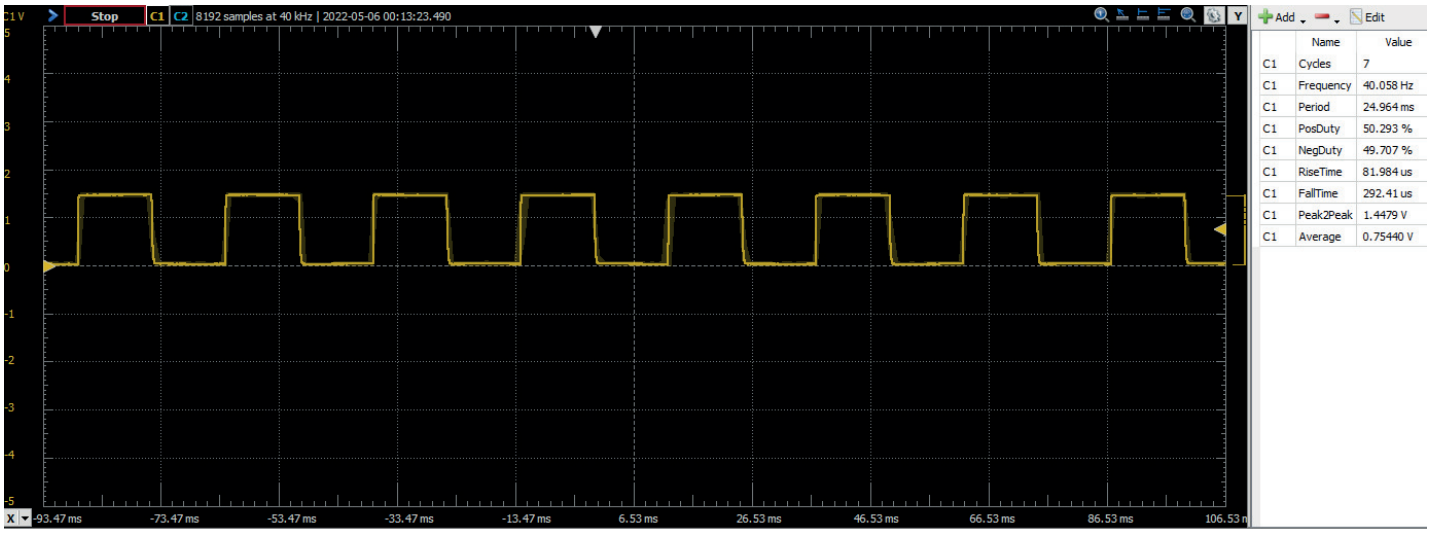
Řízení amplitudy každého ze tří D/A převodníků se provádí pomocí tří 4bitových registrů pro řízení amplitudy. Výstupy směšovače poskytují základní frekvenci signálu (šum nebo tón) pro převodník.



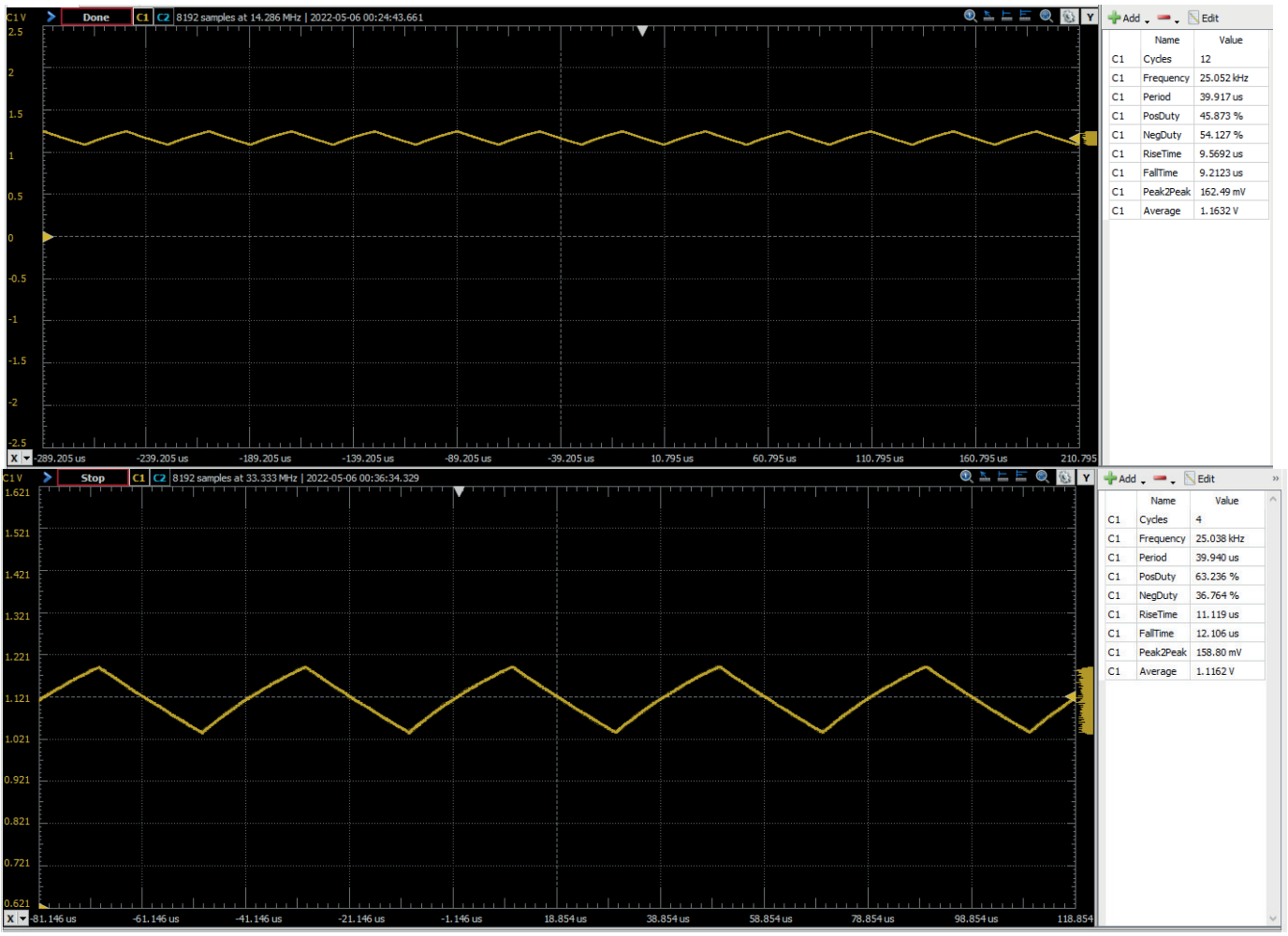
Obr. 3.13: Rozložení energie ve spektru. |Převzato z [18]|



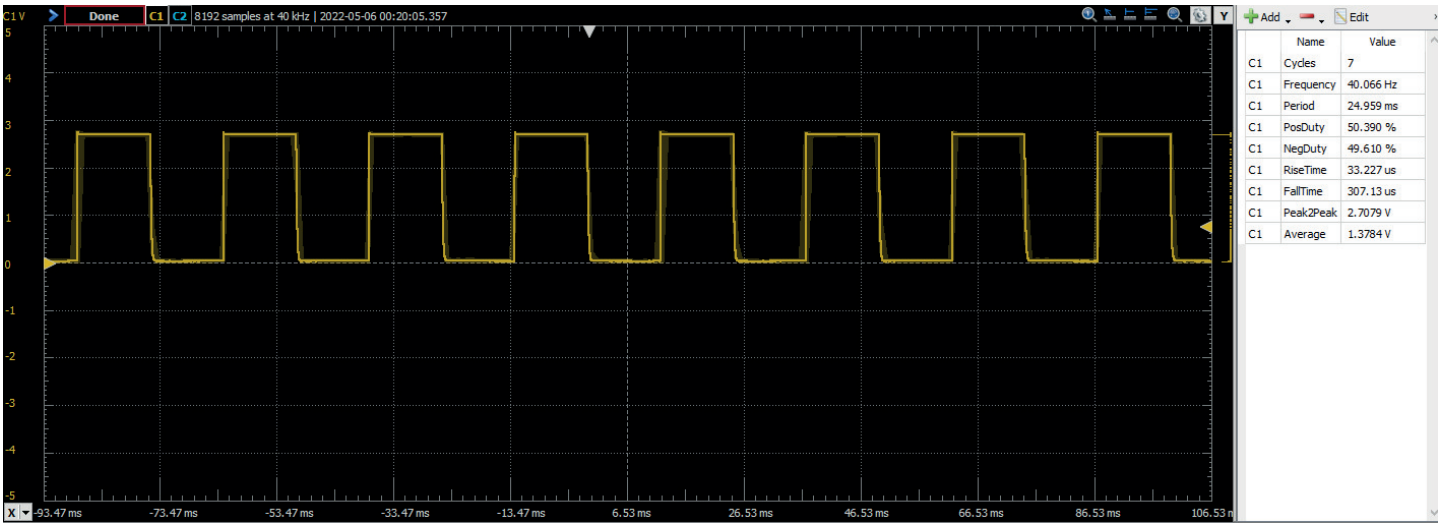
Obr. 3.14: Výstup D/A převodníku, když je použitý generátor obálky. Tvar odpovídá kombinaci 1010. |Převzato z [1]|



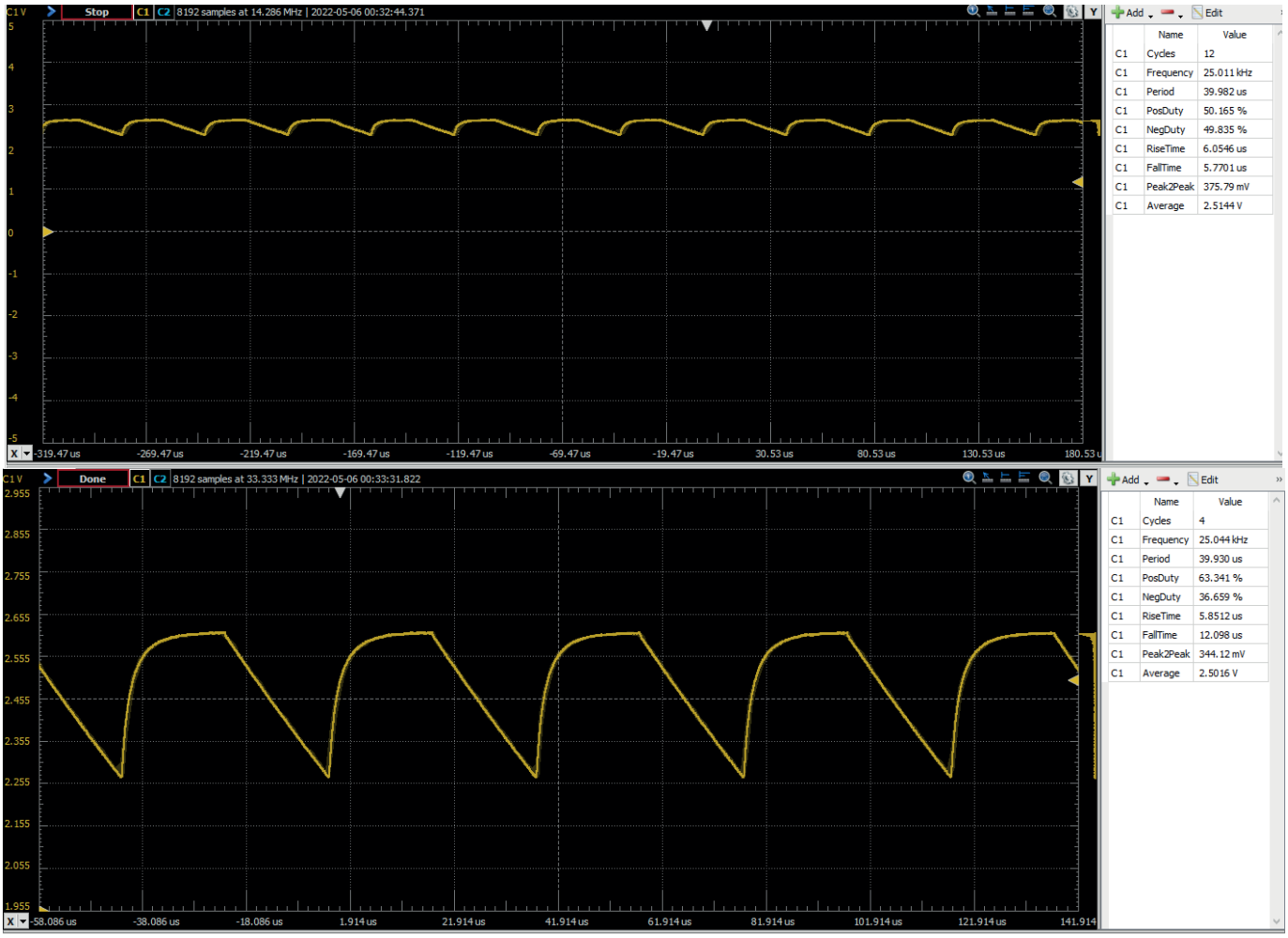
Obr. 3.15: Výstup převodníku pro frekvenci 40 Hz. Amplituda je nastavena na úroveň 1. Výstupní napětí je 0,75 V.



Obr. 3.16: Výstup převodníku pro frekvenci 25 kHz. Amplituda je nastavena na úroveň 1. Výstupní napětí je 1,1 V. Dole je přiblížený průběh.



Obr. 3.17: Výstup převodníku pro frekvenci 40 Hz. Amplituda je nastavena na úroveň 15. Výstupní napětí je 1,37 V.



Obr. 3.18: Výstup převodníku pro frekvenci 25 kHz. Amplituda je nastavena na úroveň 15. Výstupní napětí je 2,5 V. Dole je přiblížený průběh.

3.1.10 Ovládání I/O portů

R16 a R17 slouží jako mezipaměťové registry mezi datovou sběrnicí PSG/CPU (DA0 - DA7) a dvěma datovými sběrnicemi (IOA7 - IOA0 a IOB7 - IOB0). Oba porty jsou k dispozici pro AY-3-8910, AY-3-8912 má k dispozici pouze I/O port A. Použití registrů R16 a R17 pro přenos I/O dat nemá vůbec žádný vliv na generování zvuku.

Přenést data ze sběrnice CPU do připojeného periferního zařízení na I/O portu A vyžaduje následující kroky:

1. Vybrat registr R7.
2. Nastavení bitu B6 v registru R7 na 1.
3. Vybrat registr R14.
4. Zapsat data do registru R14. Data se přenesou na I/O port A.

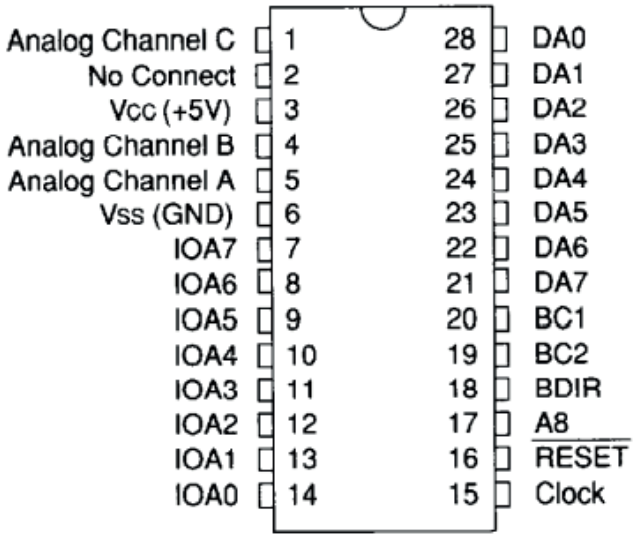
Přenést data z I/O portu A na sběrnici procesoru vyžaduje následující kroky:

1. Vybrat registr R7.
2. Nastavení bitu B6 v registru R7 na 0.
3. Vybrat registr R14.
4. Čtení dat z PSG na I/O portu A.

Pokud jsme načetli data ze sběrnice CPU, tak na portu zůstanou platná až do načtení jiných dat, do stisku tlačítka RESET nebo do přepnutí orientace portu v registru R7. Když jsme ve vstupním režimu (posíláme data z I/O A do CPU), tak se data v registrech R14 a R15 mění podle aktuálních hodnot na portu A.

3.2 AY-3-8912

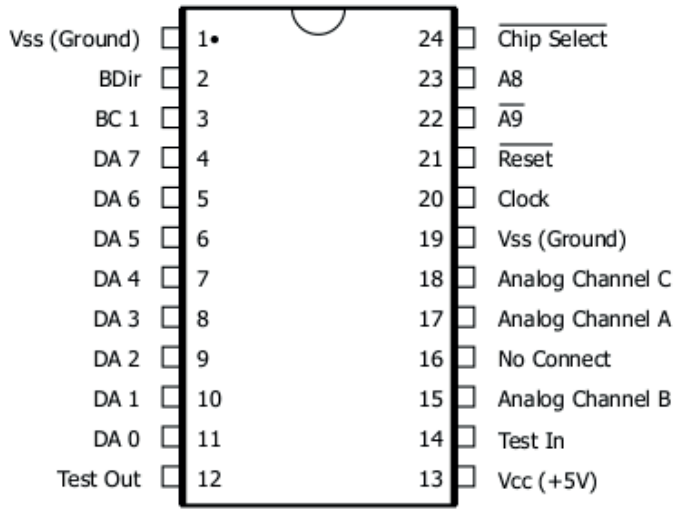
Verze 8912 je stejná jako 8910, pouze má 28 pinů, aby se ušetřilo na ceně a zmenšilo se potřebné místo na desce plošných spojů. Čip interně stále disponuje funkcí paralelního portu B, ale k vývodům je připojen pouze paralelní port A. Čip 8912 byl velmi oblíbený v evropských domácích počítačích (Vectrex, Amstrad CPC, ZX Spectrum a jeho pozdější modely). V 80. letech se čip nasazoval téměř do všech hracích automatů od firmy Bellfruit.[4]



Obr. 3.19: Označení vývodů na AY-3-8912. |Převzato z [1]|

3.3 AY-3-8913

Verze 8913 používá 24 pinové pouzdro, ale na vývody pouzdra není připojen port A a B, což umožňuje zmenšit plochu na desce plošných spojů. Byl vydán přibližně 6 měsíců po verzích 8910 a 8912. Ačkoli žádný domácí počítač nepoužíval čip 8913, lze jej nalézt v několika arkádových automatech. Nejvíce se verze 8913 uplatnila v pozdějších verzích zvukové karty Mockingboard.[5]



Obr. 3.20: Označení vývodů na AY-3-8913. |Převzato z [2]|

3.4 AY-3-8914

Verze 8914 je velmi podobná 8910, pouze má jiné uspořádání pinů a registrů. Čip se používal pouze v herních konzolích Intellivision a Intellivision II.[14] [17]

3.5 AY-3-8917

Verze 8917 má 40 pinové pouzdro. Čip se používal v zařízeních Intellivision, Entertainment Computer System.

3.6 AY-3-8930

AY-3-8930 je vylepšenou verzí modelu AY-3-8910. Mohl pracovat v režimu kompatibility s AY-3-8910 nebo v rozšířeném režimu. Tato varianta přidává několik vylepšení, jako jsou samostatné generátory obálky pro všechny tři kanály (na rozdíl od jednoho společného generátoru u AY-3-8910), větší bitová přesnost pro nastavení frekvence a hlasitosti tónu, lépe konfigurovatelný generátor šumu a přesnější D/A převodník. Funkce pinu BC2 je změněna (je ignorován a předpokládá se, že je na něm trvale log. 0), jinak je rozložení pinů stejné jako u AY-3-8910. Byl použit na zvukové kartě Covox Sound Master pro IBM-PC. Tento čip vyráběla pouze firma Microchip Technology.

3.7 Hudební formát YM

YM je zvukový formát, který odesílá uložené instrukce do čipu AY-3-8910 (8912) nebo YM2149. Odesílání dat do čipu probíhá v přesně daných časových rámcích. V čipu AY se obsahy registrů obvykle aktualizují jednou za časový rámec. Časový rámec se obvykle aktualizuje s frekvencí 50 Hz. Tento formát se používal se v Atari ST.

Všechny soubory YM jsou komprimované archivy LHA, které uvnitř obsahují jeden soubor. Komprimovaný soubor má příponu .ym. Ten lze extrahovat například pomocí programu 7-Zip nebo WinRar. Rozbalený soubor má vlastní hlavičku, která je poměrně jednoduchá a liší se podle použité verze YM („YM3!“ až „YM6!“). Extrahované soubory YM mají příponu .BIN nebo .YM.

AY-3-8910 generuje frekvence tónu na základě frekvence svých hodin. Frekvence hodin se mohla lišit pro každý počítač. Na ZX Spectrum měly hodiny frekvenci 1,7734 MHz, pro Atari ST byla frekvence 2MHz a na počítači Amstrad CPC je použita frekvence 1 MHz. Frekvence hodin se musí nastavit tak, aby odpovídala platformě, pro kterou byl určen soubor YM.

3.7.1 YM3!

Tab. 3.3: Hlavička formátu YM3.

Offset	Typ	Velikost	Popis
0	uint32_t	4	Indentifikace verze souboru „YM3!“
4	uint8_t	Podle souboru	Hodnoty registů (R0, R1, …, R13 pro každý časový rámec)
Podle souboru	uint32_t	4	Označení konce souboru „End!“

Po indentifikátoru následují datové bloky hodnot registrů pro čip AY. Registry se aktualizují jednou časový rámec. Pokud má hudba N časových rámců, pak se první datový blok skládá z N bajtů hodnot pro registr R0, N bajtů hodnot pro registr R1 a tak dále až do registru R13. Jeden časový rámec obsahuje N·14 bajtů.

Počet časových rámců v souboru se vypočítá jako:

$$\text{Počet časových rámců} = \frac{\text{velikost souboru YM} - 4}{14} \tag{3.17}$$

3.7.2 YM3b!

Tento formát je téměř totožný s formátem YM3. Přidává pouze možnost přehrávání ve smyčce. První čtyři bajty označují identifikátor „YM3b!“. Následující čtyři bajty obsahují číslo rámce, po kterém se smyčka opakuje. Poté následují bajty datových bloků (jako u YM3).

3.7.3 YM4!

Tab. 3.4: Hlavička formátu YM4.

Offset	Typ	Velikost	Popis
0	uint32_t	4	Indentifikace verze souboru „YM4!“
4	STRING[8]	8	Kontrolní řetězec „LeOnArD!“
12	uint32_t	4	Počet časových rámců v souboru
16	uint32_t	4	Vlastnosti skladby
20	uint16_t	2	Počet digidrum samplů (může být i 0)
24	uint32_t	4	Číslo rámce od kterého se skladba opakuje (většinou 0)
Pro každý digidrum sample (pokud jsou v souboru)			
28	uint32_t	4	Velikost samplu
32	uint8_t	1	Samplovaná data
Další informace			
Podle souboru	STRING	Podle souboru	Název skladby
Podle souboru	STRING	Podle souboru	Název autora
Podle souboru	STRING	Podle souboru	Komentář ke skladbě
Podle souboru	uint8_t	Podle souboru	Hodnoty registů (R0, R1, …, R15 pro každý časový rámec)
Podle souboru	uint32_t	4	Označení konce souboru „End!“

Jsou přidány další 2 registry pro uložení speciální efektů jako je digidrum. Digidrum je 4bitový sampl. Používal se pro kvalitnější reprodukci bicích. Soubor tedy obsahuje celkem 16 registrů pro každý časový rámec.

Datový blok souboru se skládá z:

Časový rámec 0 R0, Časový rámec 0 R1, ..., Časový rámec 0 R15

Časový rámec 1 R0, Časový rámec 1 R1, ..., Časový rámec 1 R15

⋮

Časový rámec n R0, Časový rámec n R1, ..., Časový rámec n R15

V takovém případě se jedná neprokládaný blok.

Datový blok může být uložen i jiném tvaru:

Časový rámec 0 R0, Časový rámec 1 R0, Časový rámec 2 R0,..., Časový rámec n R0

Časový rámec 0 R1, Časový rámec 1 R1, Časový rámec 2 R1,..., Časový rámec n R1

⋮

Časový rámec 0 R15, Časový rámec 1 R15, Časový rámec 2 R1,..., Časový rámec n R15

Pak se jedná o prokládaný blok.

Formát YM3 je prokládaný. Formát YM4 může být prokládaný i neprokládaný (prokládaný formát nabízí při kompresi pomocí LHA podstatné snížení velikosti souboru YM). Téměř všechny YM soubory dostupné na internetu jsou prokládané.

`Vlastnosti skladby` je 4 bajtová hodnota, která obsahuje informace o souboru. Význam bitů je následující:

Bit 0: Pokud je jeho hodnota 1, tak soubor obsahuje prokládané datové bloky.

Bit 1: Pokud je jeho hodnota 1, tak vzorky digidrumu jsou typu int8_t.

Bit 2: Pokud je jeho hodnota 1, tak je digidrum uložen ve 4bitovém formátu.

Bity 3 – 31 jsou nevyužity.

3.7.4 YM5!

Tab. 3.5: Hlavička formátu YM5.

Offset	Typ	Velikost	Popis
0	uint32_t	4	Indentifikace verze souboru „YM4!“
4	STRING[8]	8	Kontrolní řetězec „LeOnArD!“
12	uint32_t	4	Počet časových rámců v souboru
16	uint32_t	4	Vlastnosti skladby
20	uint16_t	2	Počet digidrum samplů (může být i 0)
22	uint32_t	4	Frekvence hodin v Hz (např. pro ATARI-ST je to 2000000)
26	uint16_t	2	Frekvence se kterou se aktualizují registry v Hz (většinou 50)
28	uint32_t	4	Číslo rámce od kterého se skladba opakuje (většinou 0)
32	uint16_t	2	Prostor pro přidání dalších dat (většinou 0)
Pro každý digidrum sample (pokud jsou v souboru)			
34	uint32_t	4	Velikost samplu
38	uint8_t	1	Samplovaná data
Další informace			
Podle souboru	STRING	Podle souboru	Název skladby
Podle souboru	STRING	Podle souboru	Název autora
Podle souboru	STRING	Podle souboru	Komentář ke skladbě
Podle souboru	uint8_t	Podle souboru	Hodnoty registů (R0, R1, . . ., R15 pro každý časový rámec)
Podle souboru	uint32_t	4	Označení konce souboru „End!“

Jedná se o nejčastěji používaný formát, který přidává další informace jako je: frekvence čipu AY pro kterou byl soubor vytvořen, název a jméno autora.

Obsah datového bloku je stejný jako u YM4 (14 registrů AY a 2 registry pro speciální efekty).

3.7.5 YM6!

Tento formát je ekvivalentní formátu YM5, ale přidává další speciální efekt.

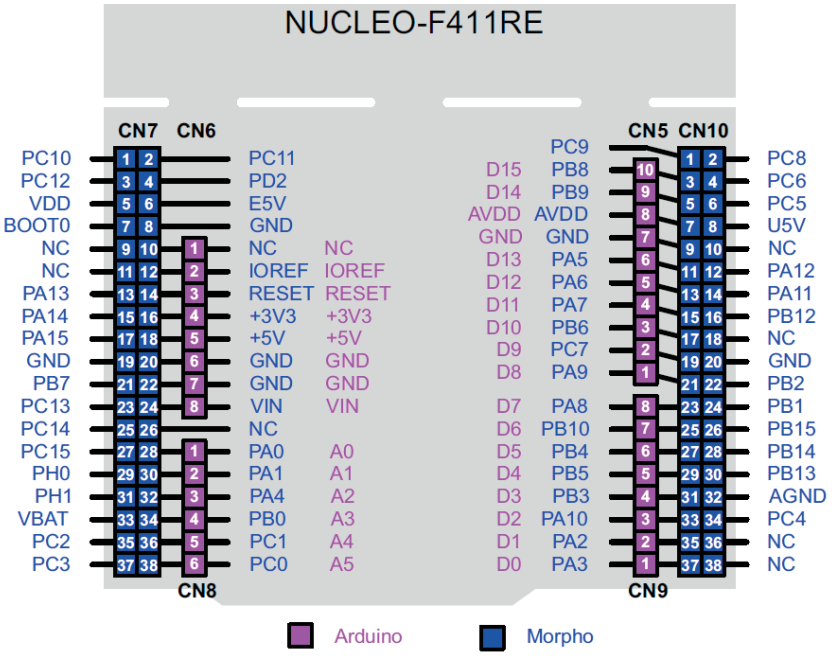
4

Řízení AY-3-8912 pomocí mikrokontroléru

V této kapitole bude popsáno řízení jednotlivých bloků AY-3-8912. Ten bude řízen pomocí mikrokontroléru STM32F411RE. Schéma zapojení je uvedeno v příloze.

Pro vkládání dat na sběrnici (DA0 – DA7) budu používat piny PC0 – PC7 na STM32F411RE. O řízení sběrnice (BC1 a BDIR) se postarají piny PC8 a PC9. Pro resetování AY-3-8912 bude sloužit tlačítko, které je připojeno na pin PC13. O generování hodin se postará časovač TIM_5 na pinu PA0. I/O porty nebudou použity.

Mikroprocesor byl programován v jazyce C v prostředí Atollic TrueSTUDIO. Níže uvedené funkce jsou součástí souboru funkce.c.



Obr. 4.1: Označení pinů na STM32F411RE. |Převzato z [63]|

4.1 Ovládání sběrnice

K ovládání sběrnice slouží funkce: `bus_inactive`, `bus_write`, `bus_latch_adress` a `bus_reset`. Funkce `bus_inactive` přepne sběrnici do režimu vysoké impedance, `bus_latch_adress` připraví sběrnici na příchod adresy registru, `bus_write` uvede sběrnici do režimu zápisu do registru a `bus_reset` nastaví piny DA0 – DA7 na nulu. To se využije k resetování sběrnice po zápisu do registru.

Funkce `bus_init` slouží k inicializaci sběrnice po zápisu programu do paměti. Proběhne nastavení orientace pinů, zvolí se rychlost, vypnou se pull-up/pull-down rezistory a nastaví se push-pull výstup. Na začátku funkce se zkontroluje, zda jsou hodiny portu C aktivní, když tomu tak není, tak se aktivují.

Funkce `reset_init` slouží k inicializaci pinu PC13, na kterém je připojeno tlačítko. Tlačítko bude sloužit k resetování AY-3-8912. Opět se zkontroluje, zda jsou hodiny portu C aktivní. Poté proběhne reset AY-3-8912 a pin PC13 se nastaví na 1 (jinak by AY-3-8912 byl pořád resetován).

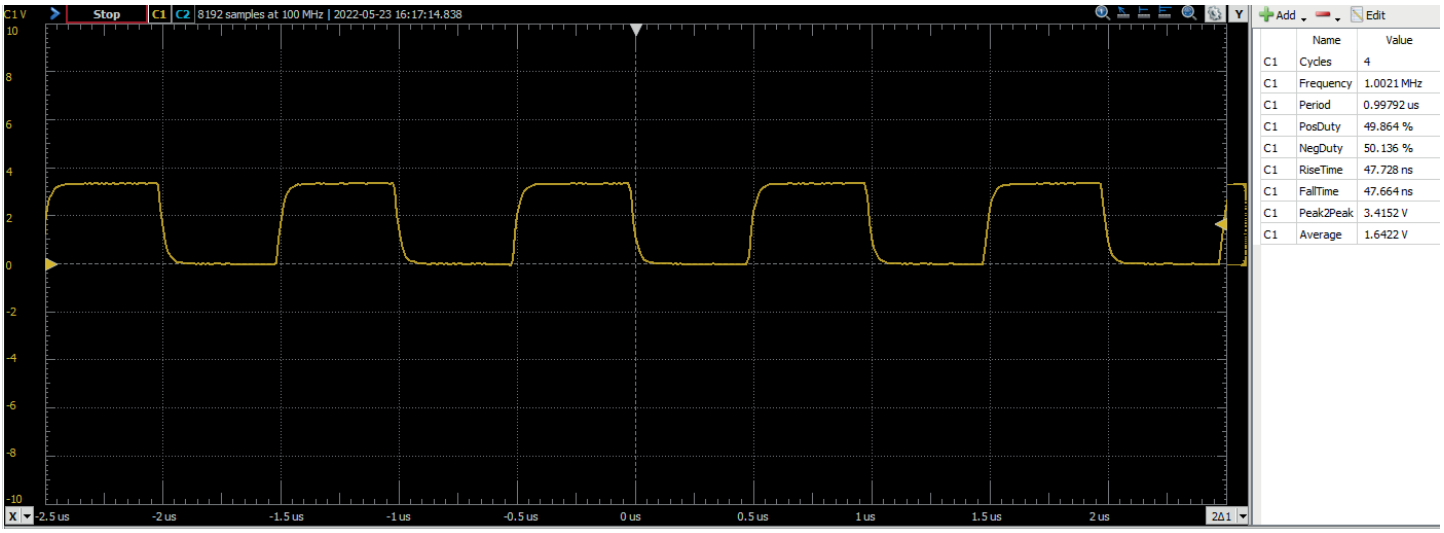
Funkce `read_reset_btn` čte stav tlačítka. Je využita pro detekci jeho stisku. Pokud tlačítko není stisknuté, tak funkce bude vracet hodnotu true, když bude zmáčknuté tak vrátí false (je aktivní na log. 0).

4.2 Generování hodin pro AY-3-8912

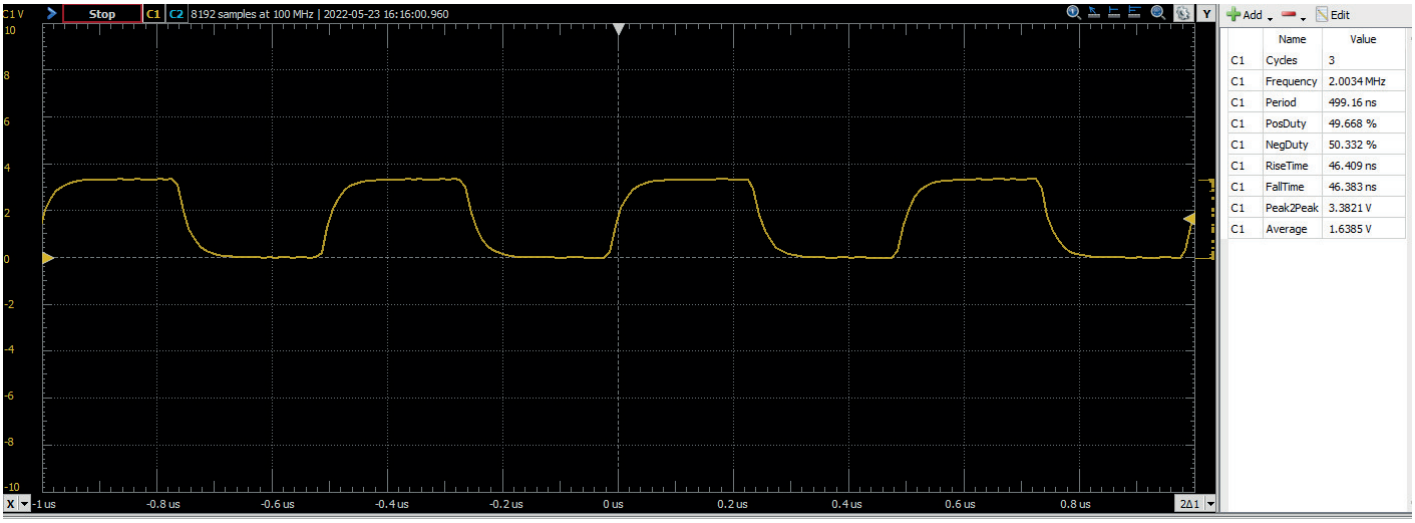
Funkce `clock_init` slouží k inicializaci časovače TIM_5. Na začátku funkce se provede kontrola aktivace hodin pro samotný časovač a pro port A. Jelikož chceme generovat hodinový signál se střídou 50:50, tak časovač přepneme do režimu PWM. Na pinu PA0 se deaktivují pull-up/pull-down rezistory, nastaví se push-pull výstup a aktivuje se alternativní funkce PA0 (výstup PWM signálu). Časovač bude počítat nahoru. Pomocí příkazu `#define CLOCK_1MHZ` a `#define CLOCK_2MHZ` lze přepínat mezi frekvencí hodin.

Vstupní frekvence čítače je 16 MHz. Pro generování nižších frekvencí použijeme předděličku. Ta vhodně vydělí kmitočet 16 MHz. Pro frekvenci 1 MHz nastavíme dělení osmi. Jeden krok (inkrementace vnitřní hodnoty čítače) bude trvat 0,5 μs. To odpovídá frekvenci 2 MHz. Ještě potřebujeme nastavit hodnotu, do které má čítač počítat. K tomu slouží registr ARR. Ten nastavíme na hodnotu 2. Dva kroky čítače budou trvat 1 μs, což odpovídá frekvenci 1 MHz. Obdobné nastavení proběhne i pro frekvenci 2 MHz. Časovač TIM_5 pracuje i s hodnotou 0, kdybychom nastavili do ARR přímo hodnotu 2, tak časovač bude počítat do 3. Proto nastavíme hodnotu 2–1. Tím jsme opět dostali 2 kroky čítače. Stejně to bude i u předděličky (8–1).

Poté následuje nastavení PWM režimu a volba střídy. Abychom měli střídu 50:50, tak do registru CCR1 dáme polovinu hodnoty registru ARR. Jako poslední akci provedeme zapnutí časovače.



Obr. 4.2: Výstup z časovače pro frekvenci 1 MHz.



Obr. 4.3: Výstup z časovače pro frekvenci 2 MHz.

4.3 Výběr a zápis do registrů AY-3-8912

Funkce `zapis_do_registru` slouží k výběru požadovaného registru a zápisu do něj. Funkce požaduje 2 parametry: `uint8_t registr` a `uint8_t hodnota`. Parametr `uint8_t registr` je adresa registru, do kterého chceme zapsat. Parametr `uint8_t hodnota` je hodnota, kterou chceme do registru vložit. Na začátku funkce je podmínka, která kontroluje, zda uživatel nezadal neplatnou adresu registru. Jelikož se nepoužívá R14 (I/O port A), tak maximální hodnota `uint8_t registr` bude 13 (to odpovídá registru R13). Poté se zavolá funkce `bus_latch_adress` a na DA0 – DA7 se vloží adresa registru. Tím je vybrán registr pro zápis. Poté se uvede sběrnice do stavu vysoké impedance a

piny DA0 – DA7 se vynulují. Na sběrnici se poté zapíše hodnota, která se má do registru zapsat. Sběrnice se uvede do stavu pro čtení (funkce `bus_write`). Hodnota ze sběrnice se zapíše do registru v AY-3-8912. Poté se zavolá funkce `bus_inactive` a `bus_reset`. Tím se dokončí zápisový cyklus a může proběhnout další.

4.4 Výběr kanálu a povolení generátoru šumu

Funkce `vyber_kanalu_tonu_noise` slouží k výběru kanálu a povolení generátoru šumu. Funkce požaduje dva parametry: `kanaly_ton channel` a `noise povolit_noise`. Parametr `kanaly_ton channel` je seznam všech kanálů, na kterých se může přehrávat tón. Parametr `noise povolit_noise` určuje, zda se povolí generátor šumu. Na začátku funkce je definována proměnná `uint8_t x`. Pomocí této proměnné se budou zapínat jednotlivé kanály. Proměnná je inicializováná na hodnotu 0xFF. Pro zapnutí kanálu je nutné, aby alespoň na jednom z prvních třech bitů proměnné `x` byla 0. Toho docílíme bitovým posunem. Na konci se zavolá funkce `zapis_do_registru`, která zapíše proměnnou `x` do registru R7.

4.5 Nastavení frekvence tónu

Funkce `nastaveni_frekvence_tonu` slouží k nastavení požadované frekvence na výstupu. Funkce požaduje dva parametry: `kanaly_ton channel` a `uint32_t frekvence`. Parametr `kanaly_ton channel` je seznam všech kanálů, na kterých se může přehrávat tón. Parametr `uint32_t frekvence` je námi požadovaná frekvence. Rozsah generovaných frekvencí na výstupu je určen frekvencí hodin AY-3-8912. Funkce si zjistí frekvenci hodin a zvolí frekvenční rozsah. Pro výpočet hodnoty, která se má vložit do registru se použije vzorec 3.2. Do vzorce se dosadí aktuální frekvence hodin AY-3-8912. Pokud hodnota po dělení vyjde větší něž 255, tak se proměnná `x` rozdělí do dvou registrů. Pokud vyjde menší, tak se nerozděluje. Poté se vybere kanál, na kterém bude zadaná frekvence.

4.6 Nastavení amplitudy kanálu

Funkce `nastaveni_amplitudy_kanalu` slouží k nastavení amplitudy vybraného kanálu. Funkce požaduje tři parametry: `kanaly_ton channel`, `uint8_t amplituda` a `envelope povolit_envelope`. Parametrem `uint8_t amplituda` se určuje amplituda. Parametr `envelope povolit_envelope` určuje, zda se aktivuje generátor obálky. Na začátku funkce se testuje, zda zadaná hodnota amplitudy je od 0 d 15. Když je požadován generátor obálky, tak se nastaví příslušný bit pro jeho aktivaci. Poté se zavolá funkce `zapis_do_registru`, která zapíše velikost amplitudy do registru.

4.7 Nastavení frekvence generátoru

Funkce `nastaveni_frekvence_noise_generatoru` slouží k nastavení frekvence generátoru šumu. Funkce požaduje parametr `uint32_t frekvence`. Tento parametr přestavuje požadovanou frekvenci šumu. Funkce si zjistí frekvenci hodin AY-3-8912 a podle vzorce 3.6 vypočítá hodnotu, která se vloží do registru R6.

4.8 Nastavení frekvence a tvaru obálky

Funkce `nastaveni_frekvence_tvaru_obalky` slouží k nastavení frekvence a tvaru obálky. Funkce požaduje dva parametry: `float frekvence` a `obalky tvar_obalky`. Parametr `float frekvence` je požadovaná frekvence obálky a `obalky tvar_obalky` je hodnota ze seznamu obálek, které AY-3-8912 umí generovat. Funkce si zjistí frekvenci hodin AY-3-8912 a zkontroluje, zda zadaná frekvence je v rozsahu frekvencí, které AY-3-8912 umí generovat. Když ano, tak se použije vzorec 3.10. Když je výsledek po dělení větší něž 255, tak se rozdělí do registrů R11 a R12. Když bude menší, tak se hodnota výsledku zapíše pouze do registru R11. Poté se vybere tvar obálky.

5

Přehrávání hudby na AY-3-8912

Pro přehrávání hudby jsem si zvolil formát YM. Většina souborů YM, které lze najít na internetu jsou prokládané. Soubor je sice menší, ale algoritmus na přehrávání je složitější. Proto jsem se rozhodl, že z prokládaného formátu udělám neprokládaný. Výsledný soubor bude větší, ale algoritmus na přehrávání bude mnohem jednodušší, něž u prokládaného formátu. Na internetu jsem našel python skript[59], který převod zařídí. Výstupem je soubor s příponou .SNG, který má následující strukturu.

Tab. 5.1: Začátek souboru SNG. Hodnoty registrů jsou převzaté ze skladby Footballer of the Year.

0x53	0x4E	0x47	0x01	0x00	0x00	0x98	0x01	0x01	0x02	0xE2
S	N	G	1	0	R0	Hodnota pro R0	R1	Hodnota pro R1	R2	Hodnota pro R2

První tři bajty značí formát souboru. Následující dva bajty značí jeho verzi. Poté už následují adresy registrů a data, která se do nich mají nahrát. V programu pro čtení souboru budou čtyři proměnné: `posuv_reg`, `posuv_hodnota`, `cislo_registru` a `hodnota_do_registru`. Proměnná `posuv_reg` bude začínat na pozici 5 a `posuv_hodnota` na pozici 6. Pro čtení následujících registrů stačí, aby se k oběma proměnným přičetla 2.

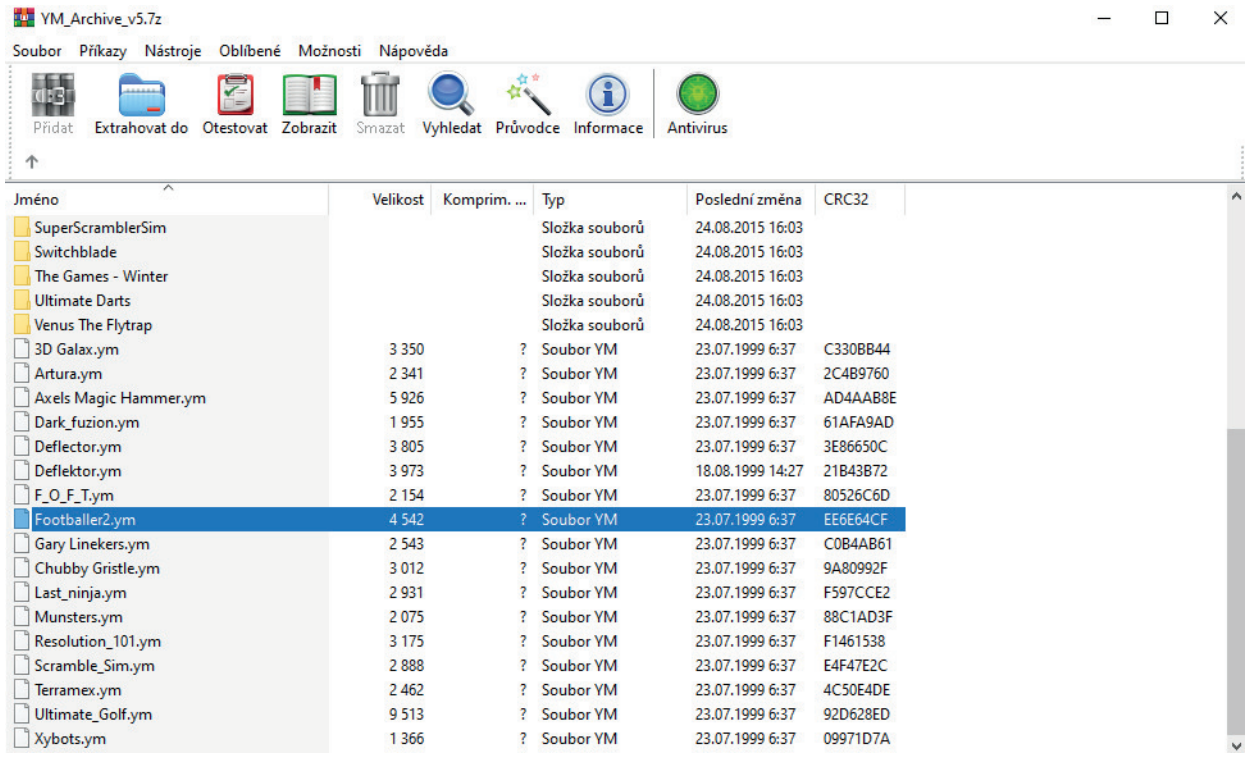
Tab. 5.2: Příklad dvou hodnot zpoždění.

0x10	0x00	0x10	0x02
Požadováno zpoždění	Délka zpoždění	Požadováno zpoždění	Délka zpoždění

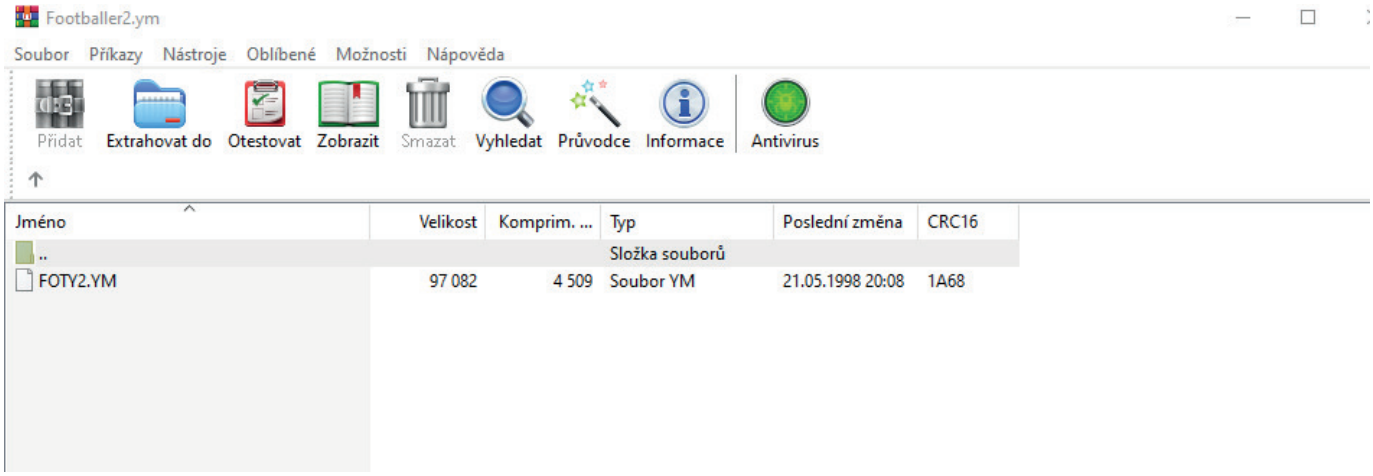
Pokud proměnná `cislo_registru` narazí na hodnotu 0x10, tak to znamená, že se požaduje zpoždění před čtením dalšího datového bloku. V proměnné `hodnota_do_registru` se uloží doba po kterou zpoždění bude trvat. Základní doba je 20 ms, to odpovídá hodnotě 0x00. Pokud `hodnota_do_registru` bude např. 0x02, tak doba zpoždění bude 60 ms. Konec souboru je detekován tak, že proměnná `cislo_registru` má hodnotu 0xFF.

5.1 Převod na neprokládaný formát

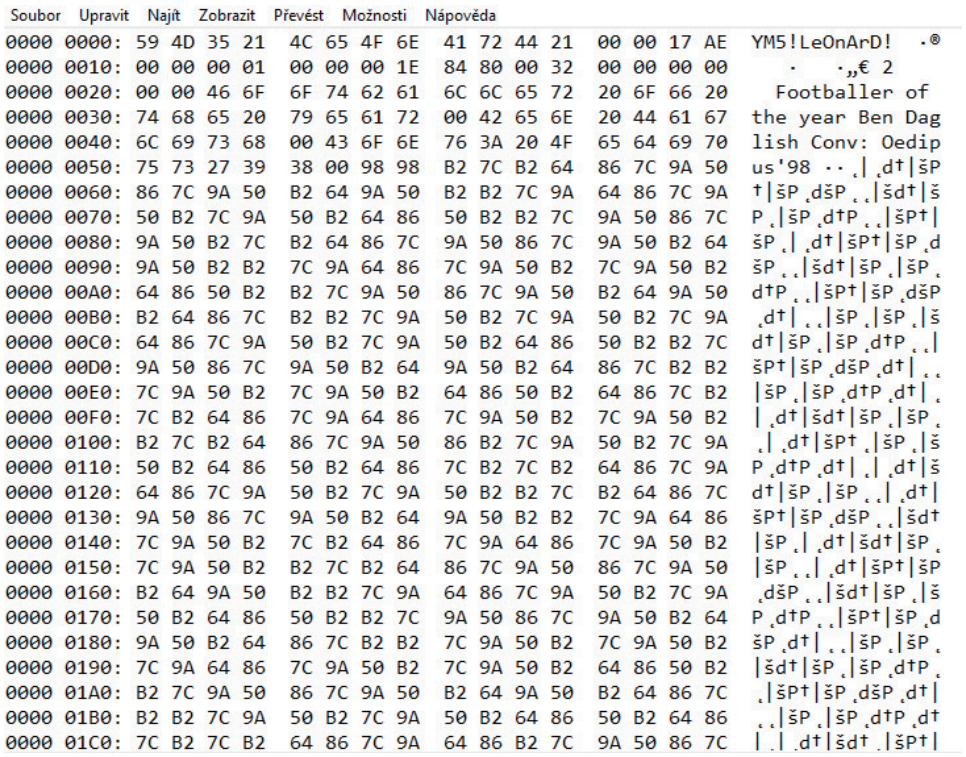
Na internetu lze najít několik archivů s hudebními soubory YM. Např. zde [61].



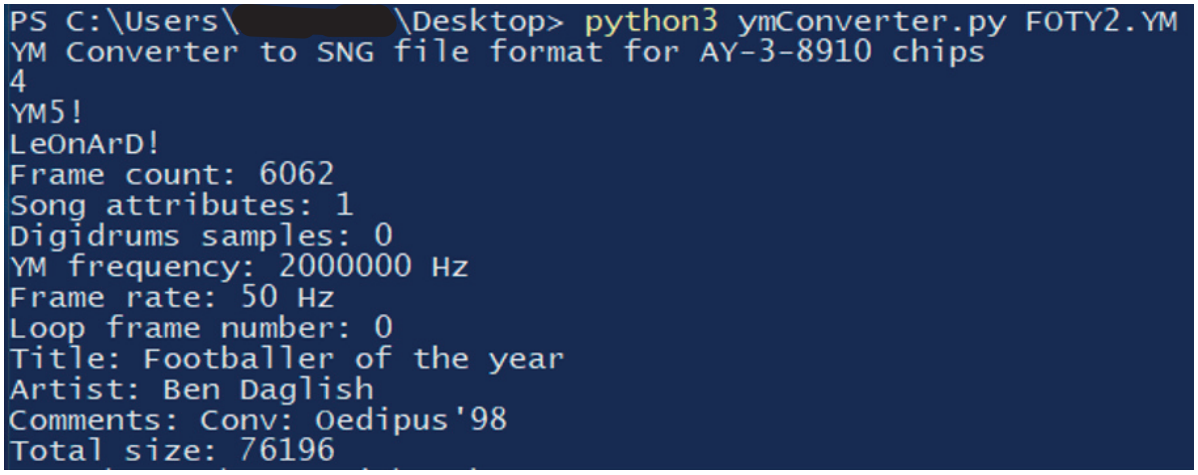
Obr. 5.1: Vybraná skladba, kterou budeme převádět. Tento soubor je komprimovaný.



Obr. 5.2: Po otevření souboru se zobrazí nekomprimovaný soubor (velikost souboru se zvětšila).

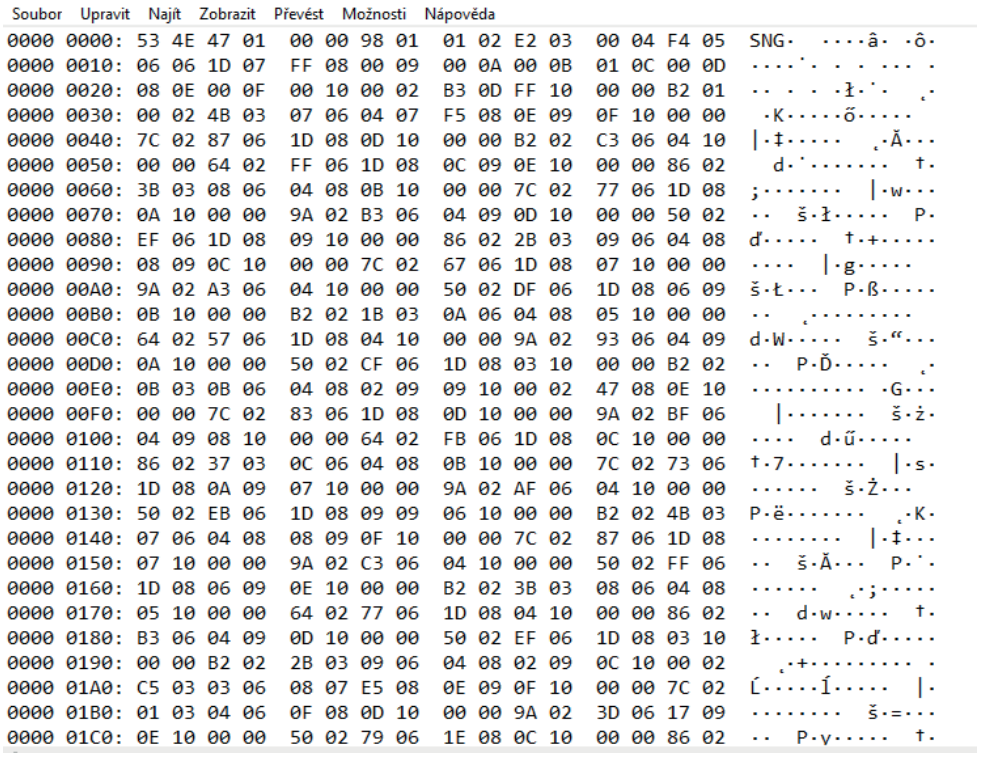


Obr. 5.3: Hexadecimální zobrazení části souboru YM v programu Salamander.



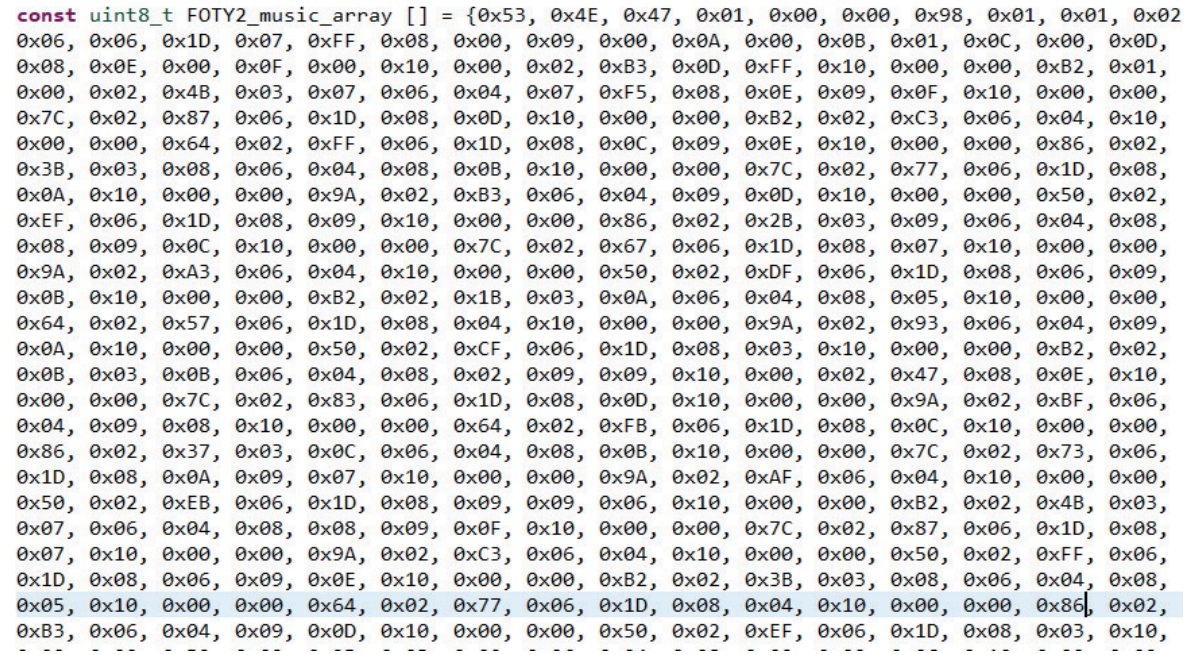
Obr. 5.4: Příkaz pro převod pomocí python skriptu a výpis hlavičky souboru YM5.

Pro přehrání tohoto souboru potřebujeme nastavit hodiny AY-3-8912 na 2 MHz. Základní doba zpoždění je odvozena od hodnoty Frame rate. V tomto případě to bude 20 ms.



Obr. 5.5: Hexadecimální zobrazení části souboru SNG v programu Salamander.

V souboru SNG jsou adresy registrů a hodnoty pro zápis uloženy v binární podobně. Lépe se pracuje s hexadecimálními hodnotami adres registrů. Pomocí [60] lze získat hexadecimální soubor SNG. Ten poté vložíme do prostředí Atollic TrueSTUDIO.



Obr. 5.6: Hexadecimální soubor v prostředí Atollic TrueSTUDIO.

5.2 Program na přehrávání

Pro přehrávání použijeme `mod_skladba`, který najdeme v main.c (celý kód je v příloze). Zakomentujeme `#define ovladani_vystupu` a `#define mod_reset` . Skladby se přepínají pomocí tlačítka, které je připojeno na pin PC13. Funkce `read_reset_btn` čte stav tlačítka. Pokud bylo zmáčknuté, tak `read_reset_btn` vrátí false. Po zmáčknutí se inkrementuje proměnná `cislo_skladby`. Podle její hodnoty se vybírá soubor, který se bude přehrávat. Pokud hodnota `cislo_skladby` bude 6, tak se hudba přestane přehrávat (v programu je k dispozici 5 hudebních souborů). Dalším stiskem se začne přehrávat první skladba. Skladba se může měnit i při přehrávání jiné. Do proměnné `uint8_t *pole` se přiřadí adresa pole skladby. Tato proměnná se pak používá pro čtení adres registrů a jejich hodnot. Ty se ukládají do proměnných `cislo_registru` a `hodnota_do_registru` . Pokud má `cislo_registru` hodnotu 0xFF, tak se soubor začne přehrávat od začátku.

5.3 Výstupní obvody

Zvukové výstupy AY-3-8912 jsou na pinech CH. A, CH. B a CH. C. Jejich připojení k reproduktoru se řešilo na každé platformě jinak. U zvukových karet Mockingboard se používal obvod LM 386. Na počítači Atari 1040ST se výstupy sečetly na bázi tranzistoru a na emitor se připojil audio jack konektor. Mnou zvolené zapojení výstupu pochází z počítače Vectrex. Obvod se skládal ze třech rezistorů, které byly spojeny se zemí a dalších třech rezistorů jejichž výstupy se spojily dohromady. Poté následoval elektrolytický kondenzátor a potenciometr pro regulaci hlasitosti. Jeho výstup se připojil na jack konektor. Elektrolytický kondenzátor se používal pro filtraci stejnosměrné složky napětí. Kdyby tam nebyl, tak toto napětí zhorší kvalitu reprodukce hudby. Nevýhoda zapojení je, že na vstupu do reproduktoru je jen zlomek napětí, které dokáže vytvořit výstup D/A převodníku. Tohle zapojení bylo úspěšně vyzkoušeno na reproduktoru Sony SRS-XB12 a Genius SP-S110. Oba jsou vybaveny vlastním zesilovačem.

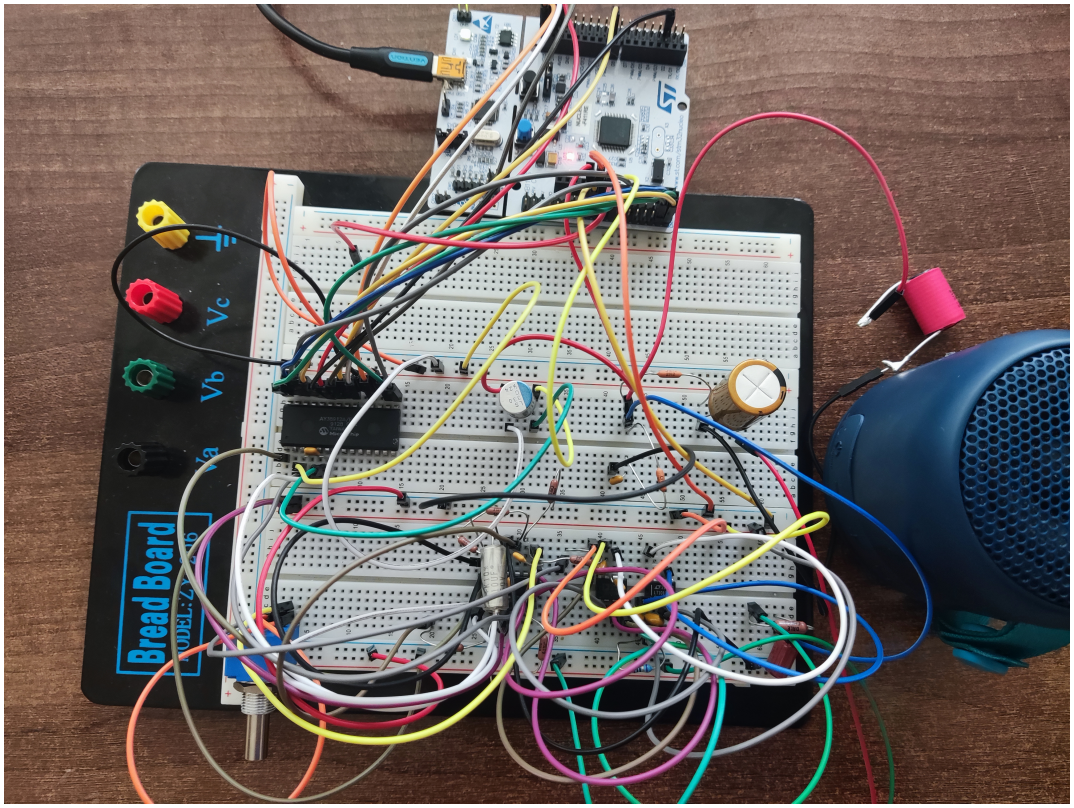
Kvůli výše uvedené nevýhodě jsem se rozhodl, že si zkusím navrhnout vlastní předzesilovač. V předzesilovači budou použity, rail-to-rail operační zesilovače, které budou napájeny napětím +5 V (nesymetrický zdroj napájení). Zdroj +5 V se nachází na desce STM32 Nucleo F411RE. Na každý zvukový výstup AY-3-8912 je připojen napěťový sledovač. Ten má za úkol kopírovat napětí kanálu na svém výstupu. Na každý výstup napěťového sledovače je připojen rezistor. Výstupy rezistorů jsou spojeny a připojeny na neinvertující vstup dalšího operačního zesilovače. Ten funguje jako neinvertující součtový zesilovač. Napětí na kanálech se sečte a zesílí. Ovládání zesílení (hlasitosti) je uskutečněno potenciometrem.

Výstup součtového zesilovače je připojen na další napěťový sledovač. K jeho výstupu je

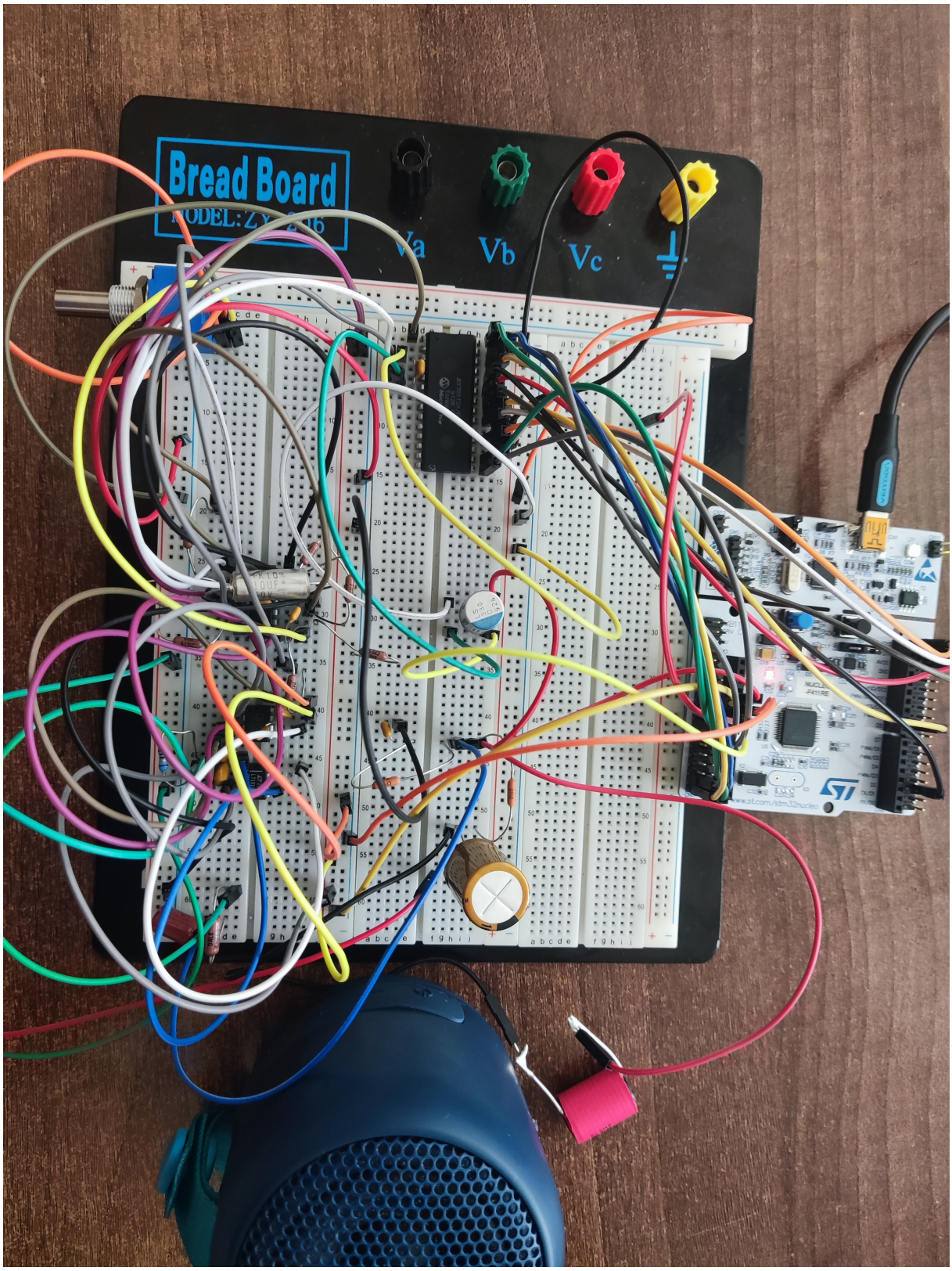
připojen obvod pro proudové posílení. Invertující vstup napěťového sledovače je připojen na výstup obvodu pro proudové posílení. Tím je zajištěno, že napětí bude stejné jako na výstupu součtového zesilovače. Většina rail-to-rail operačních zesilovačů dokáže dodat proud v řádu desítek mA. Čím nižší ohmickou hodnotu má zátěž, která je připojena k výstupu OZ, tím větší proud je třeba. U těchto typů zesilovačů dochází při nadměrném zatížení k nestabilitě. Z toho důvodu je potřeba obvod LT1010, který zajistí proudové posílení.

Další část tvoří filtr typu dolní propust, jehož propustné pásmo je do 20 kHz. Poslední částí před jack konektorem je Zobelův filtr. Ten zajistí to, aby reproduktor vykazoval ve slyšitelném pásmu pouze reálnou část impedance. Poslední část je elektrolytický kondenzátor, který plní stejnou roli jako u počítače Vectrex.

Zapojení bylo realizováno na nepájivém s použitím THT součástek. Předzesilovač fungoval, ale ve srovnání se zapojením v počítači Vectrex byla zvuková kvalita horší. To mohlo být způsobeno vodiči, kterými se jednotlivé součástky propojovaly. Pokaždé, když jsem se dotknul svazku vodičů, tak se zvuková kvalita zhoršila. Jakýkoliv pohyb nebo posun nepájivého pole po pracovní ploše znamenal zhoršení reprodukce zvuku a zvýšení hlasitosti. Možná by předzesilovač fungoval dobře na PCB, ale nejsem si tím na 100 % jistý.



Obr. 5.7: Realizace předzesilovače na nepájivém poli.



Obr. 5.8: Kliknutím na obrázek se zobrazí video se zvukovou ukázkou (nefunguje v internetových PDF prohlížečích). Přehráváná skladba je Footballer of the Year.

6

Popis a vlastnosti obvodů YAMAHA OPLx

Již zmíněné domácí počítače, jako je Atari, Commodore a ZX Spectrum měly vlastní čip pro přehrávání hudby. Na počítačích IBM PC se pro přehrávání hudby používal PC Speaker. To byl reproduktor, který byl připojen na zesilovač, na jehož vstup byl připojen časovač 8253 nebo 8254. Časovač vytvářel obdélníkový signál, který se zesílil a poslal na reproduktor. Tento způsob stačí na přehrávání jen velmi jednoduchých tónů. Pro přehrávání složitějších zvuků se musela použít PWM modulace. Aby byl výsledek PWM modulace kvalitní je nutné, aby frekvence referenčního signálu byla mnohonásobně vyšší než frekvence signálu vstupního. To však klade značné nároky na výpočetní výkon, protože před PWM modulací se musí provést mixáž všech zvukových kanálů.

Na IBM PC bylo přehrávání tak náročné, že ho nebylo možné použít přímo ve hrách, protože vykreslování vyžadovalo většinu výkonu procesoru.[19] Maximální dynamický rozsah při použití PWM modulace je cca 30 dB (CD má 96 dB). Z toho plyne, že se PC Speaker na přehrávání hudby nehodil.

Proto se začaly používat externí zvukové karty. Nejrozšířenější zvukové karty byly Adlib, Sound Blaster a Pro Audio Spectrum. Ty umožnily jednoduché přehrávání složitějších skladeb s malými nároky na výpočetní výkon procesoru. Srdcem těchto zvukových karet byly čipy OPLx od firmy Yamaha, které se staraly o generování zvuku.

Zvukový generátor YAMAHA OPLx byl vyráběn ve variantách:

1. OPL (YM3526)[21][23][26][31]
2. OPL2 (YM3812)[21][24][27][32][20]
3. OPLL (YM2413)[29]
4. OPL3 (YMF262)[21][25][30][33][20]
5. OPL3-L (YMF289)[21]

6. OPL4 (YMF278B)[34][21]

7. YMF278[22]

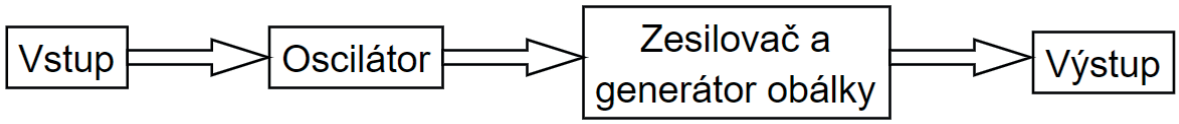
Jednotlivé typy se od sebe lišily typem pouzdra, počtem vývodů, registrů, operátorů, typem průběhů oscilátorů a typem syntézy. Až na OPL4 a YMF278, kde se používá samplová syntéza je princip generování zvuku u každé varianty stejný.

Pro generování zvuku se používaly dvě metody syntézy:

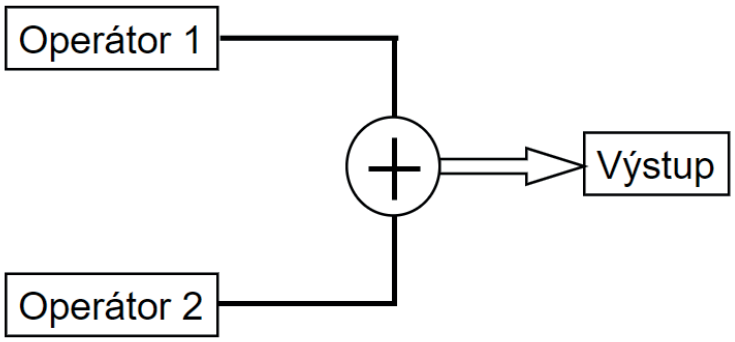
1. Aditivní – Máme dvě sinusové vlny, které obsahují pouze základní harmonickou. Vlny se mohou lišit frekvencí i amplitudou. Jejich sloučením získáme vlnu, která obsahuje vyšší harmonické. Ty mají vliv na výsledné zabarvení, které lidské ucho vnímá.

2. Pomocí frekvenční modulace (FM) – Pro změnu zabarvení se používá změna frekvence. Toho docílíme tím, že máme k dispozici dvě vlny: modulační a nosnou. Obě obvykle nemají stejnou frekvenci ani amplitudu. Mohou obsahovat i vyšší harmonické. Zvyšování nebo snižování frekvence modulační vlny má vliv na výsledné spektrum nosné vlny.

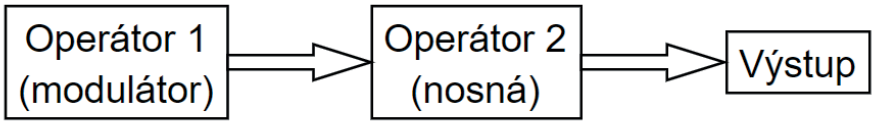
Pro realizaci obou typů modulací se používaly tzv. operátory. Podle způsobu jejich zapojení se používala aditivní nebo FM syntéza. Jeden operátor je tvořen: vstupními a výstupními obvody, oscilátorem, zesilovačem a generátorem obálky.



Obr. 6.1: Blokové schéma operátoru.



Obr. 6.2: Zapojení operátorů pro aditivní syntézu.



Obr. 6.3: Zapojení operátorů pro FM syntézu.

6.1 YAMAHA OPL

YM3526 známý jako OPL je první čip z řady OPLx. Čip se objevil na trhu v roce 1985. Vyráběl se ve 24 pinovém pouzdru. OPL produkuje monofonní zvuk a může pracovat v melodickém nebo perkusním režimu. V melodickém režimu lze vytvořit až 9 kanálů. Každý je tvořen dvěma operátory. V perkusním režimu je k dispozici až 6 melodických kanálů a 5 kanálů pro bicí nástroje. Čip obsahuje 200 osmibitových registrů, které řídí celý proces generování zvuku.

Byl použit v rozšiřující zvukové kartě pro Commodore 64/128 a v několika arkádových automatech jako Terra Cresta a Bubble Bobble. Některé arkádové automatech používaly dva čipy pro generování stereofonního výstupu. Nebyl tak oblíbený jako jeho nástupce OPL2, se kterým je zpětně kompatibilní.

6.1.1 Popis pinů

ϕM

Vstup synchronizačních pulzů. Frekvence se může pohybovat od 2 do 4 MHz. Nejčastěji je použita frekvence 3,58 MHz.

ϕSY, SH

ϕSY a SH jsou určeny pro řízení externího D/A převodníku, který převádí výstup generátoru na analogovou hodnotu. ϕSY jsou hodiny a SH je synchronizační signál.

D0 - D7

Jedná se o 8bitovou obousměrnou sběrnici, která odesílá a přijímá data mezi OPL a procesorem.

$\overline{\text{CS}}$, $\overline{\text{RD}}$, $\overline{\text{WR}}$, A0

Tyto bity řídí obousměrnou sběrnici D0 - D7.

Tab. 6.1: Stavy sběrnice pro OPL.

$\overline{CS}$	$\overline{RD}$	$\overline{WR}$	A0	Funkce sběrnice
0	1	0	0	Adresa registru se zapíše do OPL
0	1	0	1	Obsah registru je zapsán do OPL
0	0	1	0	Čte se obsah stavového registru
1	x	x	x	Stav vysoké impedance

$\overline{IRQ}$

Jedná se o signál přerušení, který vystupuje časovačů.

$\overline{IC}$

Při nastavení na log. 0 je PSG resetován. Obsah všech registrů se přepíše na 0.

MO

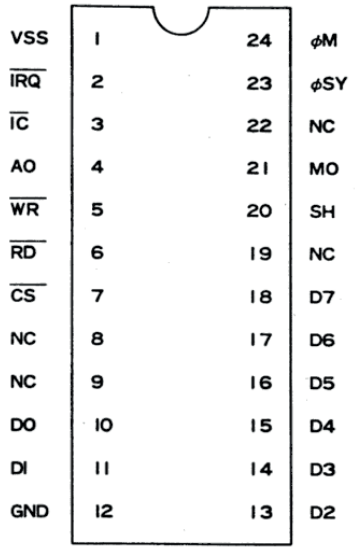
Jedná se o digitální výstup generátoru zvuku. Tento pin se zapojí na vstup externího D/A převodníku.

VCC

Napájecí svorka +5 V.

GND

Uzemňovací svorka.



Obr. 6.4: Zapojení vývodů pro OPL. |Převzato z [31]|

6.1.2 Bloky pro generování zvuku

Pole registrů (Register array)

OPL se je řízeno obsahem registrů. Pomocí nich se určuje se tvar obálky průběh FM syntézy.

Fázový generátor (PG)

Generuje frekvenci (fázi) pro každý operátor. Frekvence pro jednotlivé operátory jsou odvozeny z hodnot v příslušných registrech.

Generátor obálky (EG)

Vytváří obálku, která se mění v čase podle obsahu registrů.

Operátor (OP)

Operátor přijímá informace o aktuální hodnotě fáze φ z fázového generátoru a informace o obálce E z generátoru obálky a vypočítává $\mathrm{E} \cdot \sin(\varphi)$.

Akumulátor (ACC)

Ukládá výstupní data ze všech operátorů. Data se použijí pro vzorkování s frekvencí 50 KHz. Ty poté použije D/A převodník.

Oscilátor vibrato + oscilátor pro amplitudovou modulaci

Oscilační frekvence je 6,4 Hz pro vibrato a 7 Hz pro AM.

Časovače (Timer)

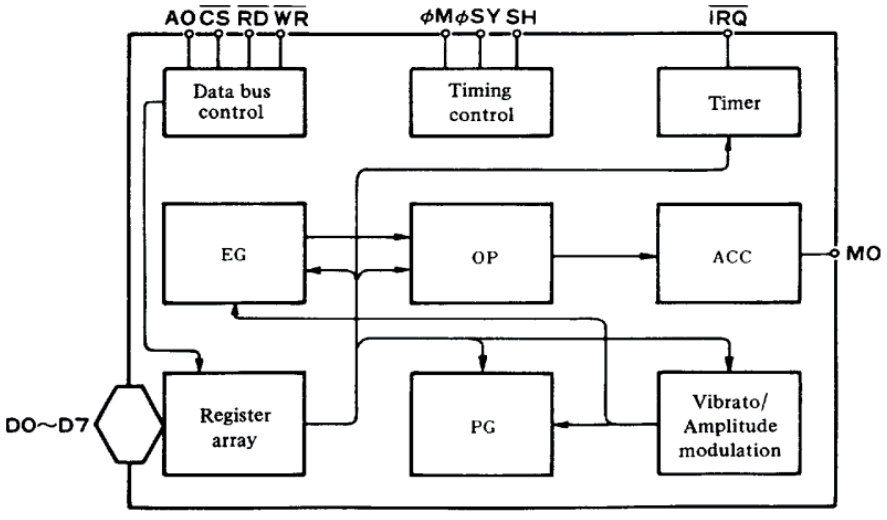
Dva časovače pro všeobecné použití. Timer 1 má časový rozsah od 80 μs do 20,4 ms, po přetečení se vygeneruje signál IRQ. Timer 2 má časový rozsah od 320 μs do 82 ms. Když časovače přetečou, nastaví se na naprogramovanou hodnotu v registrech, od které začnou opět počítat.

Řízení datové sběrnice (Data bus control)

Ovládá orientaci sběrnice a povoluje stav vysoké impedance.

Řízení časování (Timing control)

Nastavuje počáteční hodnotu a dobu, za kterou čítače přetečou.



Obr. 6.5: Zapojení jednotlivých bloků. |Převzato z [31]|

6.1.3 Popis registrů

OPL má k dispozici 200 registrů. Pro zápis do konkrétního registru je třeba poslat číslo registru (01 - C8) na sběrnici. Po zápisu čísla registru na sběrnici je třeba počkat 12hodinových cyklů, než se zpracují data. Poté se na sběrnici pošlou data, která se mají zapsat do registru. Po zápisu dat do registru musí uplynout 84hodinových cyklů než lze provést jakoukoli další operaci s OPL.

Status registr

Jediný registr, který lze číst. Ke čtení registru nastavíme sběrnici do režimu pro čtení. Na pinech D7, D6 a D5 můžeme číst aktuální stav.

Tab. 6.2: Bity status registru.

D7	D6	D5	D4	D3	D2	D1	D0
IRQFlag	T1Flag	T2Flag					

T1Flag

Tento příznak je nastaven na 1, když dojde k přetečení timeru 1. Tento příznak se po přetečení nenastaví zpět na 0, pokud je IRQReset nastaven na 1.

T2Flag

Tento příznak je nastaven na 1, když dojde k přetečení timeru 2. Tento příznak se po přetečení nenastaví zpět na 0, pokud je IRQReset nastaven na 1.

IRQFlag

Nastaví se na 1, pokud je přeteče timer 1 nebo 2. Tento příznak se po přetečení nenastaví zpět na 0, pokud je IRQReset nastaven na 1.

Tab. 6.3: Mapa registrů pro OPL.

Adresa registru	BIT B7	BIT B6	BIT B5	BIT B4	BIT B3	BIT B2	BIT B1	BIT B0
0x01	LSI Test Register							
0x02	Timer 1							
0x03	Timer 2							
0x04	IRQReset	T1 Mask	T2 Mask				T2 Start	T1 Start
0x08	CSW	Note-Sel						
0x20 - 0x35	Tremolo	Vibrato	Sustain	KSR	Frequency Multiplication Factor			
0x40 - 0x55	Key Scale Level		Output Level (Attenuation)					
0x60 - 0x75	Attack Rate				Decay Rate			
0x80 - 0x95	Sustain Level				Release Rate			
0xA0 - 0xA8	Frequency Number (Lower 8 bits)							
0xB0 - 0xB8			Key-On	Block Number			Frequency Number (High bits)	
0xBD	TremDep	VibrDep	PercMode	BD On	SD On	TT On	CY On	HH On
0xC0 - 0xC8					FeedBack Modulation Factor			SynthType

Ve skupině registrů 0x20, 0x40, 0x60 a 0x80 jsou pro každý kanál alokovány dva registry. Každý registr je přiřazen k jednomu operátoru. Následující tabulka ukazuje, který operátor se mapuje, na kterou sadu registrů a jeho offset (v hexadecimálním tvaru).

Tab. 6.4: Přiřazení adres operátorů skupině registrů pro melodický režim.

Kanál	1	2	3	4	5	6	7	8	9
Operátor 1	0x00	0x01	0x02	0x08	0x09	0x0A	0x10	0x11	0x12
Operátor 2	0x03	0x04	0x05	0x0B	0x0C	0x0D	0x13	0x14	0x15

Tab. 6.5: Přiřazení adres operátorů skupině registrů pro perkusní režim.

Kanál	1	2	3	4	5	6	BD	SD	TT	CY	HH
Operátor 1	0x00	0x01	0x02	0x08	0x09	0x0A	0x10	0x14	0x12	0x15	0x11
Operátor 2	0x03	0x04	0x05	0x0B	0x0C	0x0D	0x13				

Příklad přiřazení operátorů k registrům na kanále 5
0x29 - Operator 1 Tremolo/Vibrato/Sustain/KSR/Multiplication
0x2C - Operator 2 Tremolo/Vibrato/Sustain/KSR/Multiplication
0x49 - Operator 1 Key Scale Level/Output Level
0x4C - Operator 2 Key Scale Level/Output Level
0x69 - Operator 1 Attack Rate/Decay Rate
0x6C - Operator 2 Attack Rate/Decay Rate
0x89 - Operator 1 Sustain/Release
0x8C - Operator 2 Sustain/Release
0xA4 - Frequency Number (Lower 8 bits)
0xB4 - Key On/Block Number/Frequency (High 2 bits)
0xC4 - FeedBack/Synthesis Type

LSI Test Register
Všechny bity nastavit na 0.

Timer 1
Je to 8bitový čítač. Pokud je časovač 1 povolen, přednastavená hodnota v registru se bude zvyšovat, dokud nedojde k jeho přetečení. Pokud dojde k přetečení, nastaví se bit stavového registru na 1, pin $\overline{\text{IRQ}}$ bude v log. 0 a do časovače se znovu načte přednastavená hodnota. Hodnota tohoto časovače se inkrementuje každých 80 mikrosekund.

Timer 2
Je to 8bitový čítač. Pokud je časovač 1 povolen, přednastavená hodnota v registru se bude zvyšovat, dokud nedojde k jeho přetečení. Pokud dojde k přetečení, nastaví se bit

stavového registru na 1, pin $\overline{\text{IRQ}}$ bude v log.0 a do časovače se znovu načte přednastavená hodnota. Hodnota tohoto časovače se inkrementuje každých 320 mikrosekund.

IRQReset

Resetuje příznaky časovačů a IRQFlag ve stavovém registru, když je nastaven na 1. Všechny ostatní bity jsou při nastavení tohoto bitu ignorovány.

T1 Mask

Maska časovače 1. Je-li nastaven, tak stavový bit není při přetečením ovlivněn (zůstane na 0) a T1 Start je ignorován.

T2 Mask

Maska časovače 2. Je-li nastaven, tak stavový bit není při přetečením ovlivněn (zůstane na 0) a T2 Start je ignorován.

T2 Start

Nastavením na hodnoty na 1 si čítač 2 načte přednastavenou hodnotu ze svého registru a začne inkrementovat.

T1 Start

Nastavením na hodnoty na 1 si čítač 1 načte přednastavenou hodnotu ze svého registru a začne inkrementovat.

CSW

Je-li nastaven, vybírá složenou sinusovou syntézu řeči (všechny bity KEY-ON musí být prázdné). Je-li nastaven na 0, vybírá se režim FM hudby.

Note-Sel

Osm oktáv je rozděleno na polovinu. Tento bit ovládá jejich dělení. Když je NTS = 0, tak frekvence, kde se rozdělí oktávy se odvodí od dvou nejvyšších bitů registru Frequency Number. Když je NTS = 0 frekvence je odvozena od 8 nižších bitů registru Frequency Number.

When NTS=0

Číslo oktávy	0		1		2		3		4		5		6		7	
F-NUMBER MSB	*		*		*		*		*		*		*		*	
F-NUMBER 2nd	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
Key scale No.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

When NTS=1

Číslo oktávy	0		1		2		3		4		5		6		7	
F-NUMBER MSB	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
F-NUMBER 2nd	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
Key scale No.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

*: Don't care

Obr. 6.6: Rozdělení oktáv. |Převzato z [33]|

Tremolo

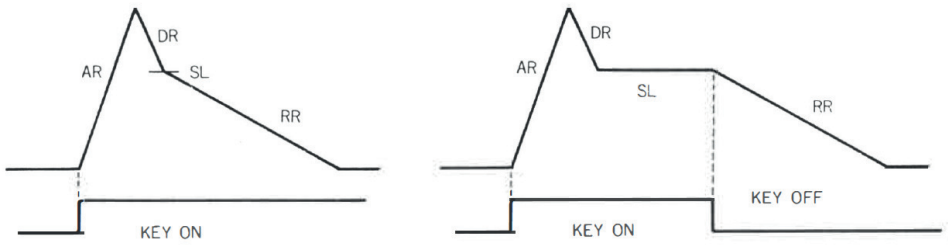
Nastavením na 1 se aktivuje efekt Tremolo (periodická změna amplitudy tónu). Pro jeho generování se použije oscilátor amplitudové modulace (frekvence 7 Hz). Hloubka AM je řízena bitem TremDep na adrese 0xBD.

Vibrato

Nastavením na 1 se aktivuje efekt Vibrato (periodická změna frekvence tónu). Pro jeho generování se použije Vibrato oscilátor (frekvence 6,4 Hz). Hloubka vibrata je řízena bitem VibrDep na adrese 0xBD.

Sustain

Je-li 1, bude výstupní úroveň operátoru udržována na úrovni Sustain (pokud je KEY-ON aktivní), dokud nebude provedeno KEY-OFF. Když je hodnota 0, zvuk začne doznívat okamžitě po dokončení fáze Decay (nezáleží na KEY-ON).



Obr. 6.7: Vliv KEY-ON na obálku ADSR. Vlevo je zobrazen typ obálky pro bicí nástroje, vpravo pro tóny. |Převzato z [33]|

KSR

U běžných hudebních nástrojů se strmost Attack/Decay zrychluje s rostoucí frekvencí (výškou) tónu, tento registr slouží k simulaci této funkce. Pokud je tento bit nastaven, je obálka zvuku zkracována podle toho, jak stoupá její frekvence.

Frequency Multiplication Factor

Tyto bity udávají, kterou frekvenci bude operátor produkovat po násobení. Výsledná frekvence operátoru je:

$$\text{Frequency Number} \cdot \text{Frequency Multiplication Factor} \tag{6.1}$$

Tab. 6.6: Význam úrovní v registru.

Hodnota registru HEX	Kterou frekvenci bude operátor produkovat po vynásobení
0	O velkou oktávu níže
1	Na zadané frekvenci jako je v registru Frequency Number
2	O oktávu výše
3	O oktávu a kvintu výše
4	O dvě oktávy výše
5	O dvě oktávy a velkou tercii výše
6	O dvě oktávy a kvintu výše
7	O dvě oktávy a malou septimu výše
8	O tři oktávy výše
9	O tři oktávy a velkou sekundu výše
A	O tři oktávy a velkou tercii výše
C	O tři oktávy a kvintu výše
E	O tři oktávy a velkou septimu výše

Key Scale Level

U hudebních nástrojů se hlasitost snižuje s rostoucí výškou tónu. Registr slouží k simulaci tohoto efektu.

Key scale	0	2	1	3
Útlum	0	1.5dB/oct	3dB/oct	6dB/oct

Obr. 6.8: Nastavení jednotlivých úrovní útlumu. |Převzato z [33]|

Output Level (Attenuation)

Tlumení výstupní úrovně operátoru. 0 je nejhlasitější, 3F je nejtišší. V aditivní syntéze změna výstupní úrovně libovolného operátoru mění hlasitost kanálu (nastavená hodnota platí pro oba operátory). V FM syntéze se změnou úrovně nosné mění hlasitost kanálu (nastavená hodnota platí jen pro druhý operátor, první zůstává nedotčen). Změna úrovně změní frekvenční spektrum vytvářené nosnou.

Útlum se vypočítá pomocí vzorce:

$$\text{Útlum} = 24 \cdot \text{D5} + 12 \cdot \text{D4} + 6 \cdot \text{D3} + 3 \cdot \text{D2} + 1,5 \cdot \text{D1} + 0,75 \cdot \text{D0} \,[\text{dB}] \tag{6.2}$$

Attack Rate

Určuje dobu náběhu zvuku. 0 je nejpomalejší, F je nejrychlejší.

$$\text{Attack Rate} = 2^3 \cdot \text{D7} + 2^2 \cdot \text{D6} + 2^1 \cdot \text{D5} + 2^0 \cdot \text{D4} \tag{6.3}$$

Decay Rate

Určuje dobu doznívání zvuku. 0 je nejpomalejší, F je nejrychlejší.

$$\text{Decay Rate} = 2^3 \cdot \text{D3} + 2^2 \cdot \text{D2} + 2^1 \cdot \text{D1} + 2^0 \cdot \text{D0} \tag{6.4}$$

Sustain Level

Určuje amplitudu, ve které zvuk přestane doznívat a bude mít konstantní úroveň. Sustain je vyjádřen jako podíl maximální úrovně. 0 je nejtišší a F je nejhlasitější úroveň (93 dB). Aby se tato funkce projevila, musí být nastaven bit Sustain v registru 0x20 – 0x35.

$$\text{Sustain Level} = 24 \cdot \text{D7} + 12 \cdot \text{D6} + 6 \cdot \text{D5} + 3 \cdot \text{D4} \,[\text{dB}] \tag{6.5}$$

Release Rate

Určuje rychlost, s jakou zvuk zmizí po vypnutí KEY-ON. 0 je nejpomalejší a F je nejrychlejší.

$$\text{Release Rate} = 2^3 \cdot \text{D3} + 2^2 \cdot \text{D2} + 2^1 \cdot \text{D1} + 2^0 \cdot \text{D0} \tag{6.6}$$

Frequency Number (Lower 8 bits)

Určuje výšku tónu v dané oktávě. Registr pro uložení osmi nejnižších bitů.

Key-On

Je-li 1, je výstup kanálu povolen (generuje se zvuk).

Block Number

Určuje oktávu tónu. 0 je nejnižší, 7 je nejvyšší.

Frequency Number (high bits)

Registr pro uložení dvou nejvyšších bitů.

$$\text{Frequency Number} = \frac{\text{f}_\text{P} \cdot 2^{19} \cdot 2^{\text{Block Number}-1}}{\text{f}_\text{s}} \tag{6.7}$$

$$\text{f}_\text{s} = \frac{\text{f}_\text{CLK}}{288} \tag{6.8}$$

kde f_P je požadovaná frekvence tónu v Hz, Block Number je číslo oktávy, f_s je vzorkovací frekvence v Hz a f_CLK je frekvence hodin v Hz.

TremDep
Nastavení hloubky tremola. 0 = 1,0 dB, 1 = 4,8 dB.

VibrDep
Nastavení hloubky vibrata. 0 = 7 centů, 1 = 14 centů. Cent je 1/100 půltónu.

PercMode
0 = Melodický režim.
1 = Režim bicích nástrojů.

BD On
Nastavením na 1 se aktivuje (KEY-ON) kanál basového bubnu (Base Drum).

SD On
Nastavením na 1 se aktivuje (KEY-ON) kanál Snare Drum.

TT On
Nastavením na 1 se aktivuje (KEY-ON) kanál Tom-Tom.

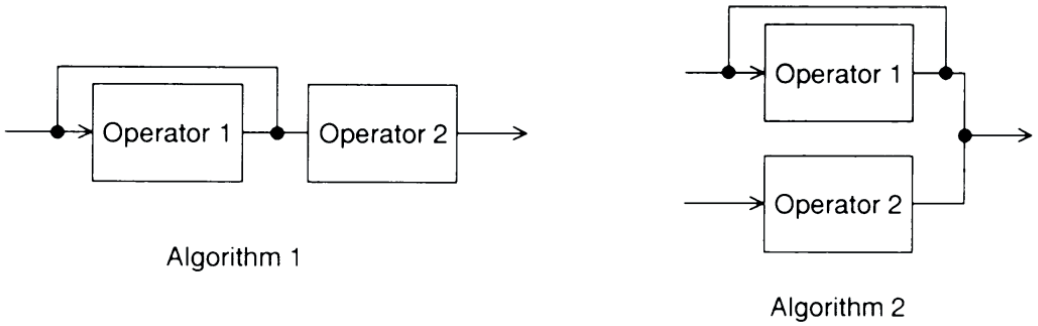
CY On
Nastavením na 1 se aktivuje (KEY-ON) kanál činelů (Cymbal).

HH On
Nastavením na 1 se aktivuje (KEY-ON) kanál Hi-Hat.

Pro použití perkusního režimu musí být registry KEY-ON kanálů 7, 8 a 9 nastaveny na 0. Ostatní parametry, jako je attack, decay, sustain, release a Frequency Number, musí být vhodně nastaveny.

FeedBack Modulation Factor
Lze zvolit zpoždění signálu, který se pomocí zpětné vazby vrací na vstup modulátoru. Používá se k napodobení zvuku strunových nástrojů. Pokud je hodnota 0, není přítomna žádná zpětná vazba. Je-li 1 - 7, modulátor pošle část svého výstupu zpět do svůj vstup.

SynthType
1 = Aditivní syntéza.
0 = Frekvenční modulace.



Obr. 6.9: Vlevo zapojení bloků pro aditivní syntézu, vpravo pro FM syntézu. |Převzato z [34]|

6.1.4 Princip funkce

Generování hudby je založeno na frekvenční modulaci. Frekvence prvního oscilátoru (modulátor) ovlivňuje výstup druhého oscilátoru (nosná). Změnou amplitudy a frekvence modulátoru se mění výška a zabarvení nosné. Změna výšky nosné je lidským uchem slyšitelná pouze při malé frekvenci modulátoru. Při vyšších frekvencích ucho rozezná pouze změnu barvy tónu. Vyšší frekvence modulátoru se používají k napodobení hudebních nástrojů, nižší frekvence se používají na vytváření zvukových efektů (vibrato). Frekvence nosné určuje základní frekvenci vytvořeného tónu.

Aby nově vytvoření tón byl harmonický (ve spektru jsou obsaženy pouze celočíselné násobky základní harmonické) je třeba, aby poměr mezi frekvencí modulátoru a frekvencí nosné byl celočíselný (2 : 1, 4 : 2, 6 : 4). Pokud je poměr např. 1,394 : 1, tak výsledný tón bude ve spektru obsahovat neceločíselné násobky základní harmonické. Tímhle spektrem se vyznačují bicí nástroje.

Všechny obvody řady OPLx obsahují paměť ROM, která má velikost 512 bajtů a je rozdělená přesně napůl. Oscilátory jsou plně digitální (všechny OPLx) a generují pouze sinusový signál (pouze OPL). Aby se nemusel vypočítávat jeho průběh je první čtvrtina sinusu uložena v prvních 256 bajtech paměti ROM. Zbytek průběhu se dá pomocí symetrie jednoduše odvodit. Hodnoty jsou uloženy v logaritmické škále. Druhých 256 bajtů obsahuje převodní hodnoty z logaritmických hodnot na exponenciální. Tato část tabulky je využita při převodu hodnot do lineárního měřítka. Výpočty potřebné pro FM obsahují operaci násobení, ale jelikož celý proces vytváření tónu probíhá v logaritmickém měřítku, tak se násobení změní na sčítání. To je mnohem rychlejší operace něž násobení. Poté se výstupy kanálů sloučí a jdou na vstup externího D/A převodníku.

6.1.5 Řízení sběrnice

Čip je vybaven obousměrnou osmibitovou sběrnicí. Ta je ovládána signály $\overline{\text{CS}}$, $\overline{\text{RD}}$, $\overline{\text{WR}}$ a AO. Nejprve sběrnici nastavíme na výběr indexu registru (1 - 200), do kterého chceme

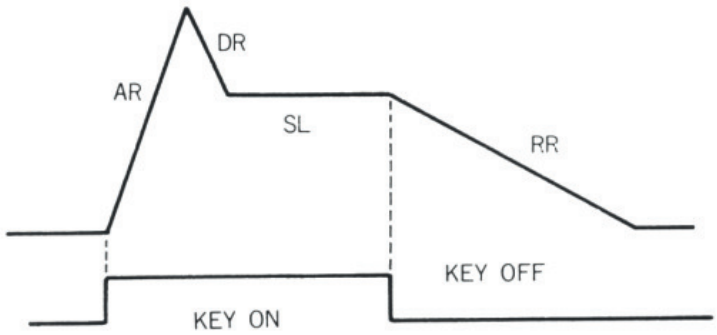
zapsat. Poté jí nastavíme do módu pro zápis hodnoty do registru. Registry řídí celý proces generování zvuku, proto je jejich počet vyšší než u AY-3-891x.

6.1.6 Ovládání obálky

Signál vytvářený v každém operátoru lze ovlivnit použitím obálky, pomocí které lze měnit amplitudu. Jedná se o obálku typu ADSR. Základní tvar obálky je určen parametry:

1. Attack rate - doba náběhu zvuku na 100% hodnotu amplitudy. Vyšší hodnota odpovídá strmějšímu náběhu.
2. Decay rate - doba poklesu od 100% hodnoty amplitudy. Čím vyšší je hodnota, tím kratší je doba poklesu.
3. Sustain level - určuje hodnotu konstantní amplitudy po Decay. V řídicích registrech je uložená jako zlomek maximální úrovně amplitudy.
4. Release rate - určuje rychlost s jakou zvuk dosáhne 0% amplitudy po odeznění signálu KEY-ON. Čím vyšší hodnota, tím strmější pokles.

Hodnoty Attack a Decay jsou společně uloženy v jednom osmibitovém registru a hodnoty Sustain a Release v registru druhém. Tyto registry jsou alokovány pro každý operátor zvlášť. Bitem KEY-ON se řídí doba úrovně Sustain. Tento bit je alokovaný pro každý kanál, tím je umožněno nezávislé ovládání kanálů.



Obr. 6.10: Vysvětlení úrovní ADSR. |Převzato z [32]|

6.2 YAMAHA OPL2

Verze OPL2 (YM3812) má více vnitřních registrů (244) a oscilátory umí generovat tři další průběhy signálu. Má stejný počet vývodů jako OPL a je s ním zpětně kompatibilní.

OPL2 se používal v mnoha arkádových automatech. Byl použit na zvukových kartách jako je Ad Lib Music Synthesizer Card a Sound Blaster. YM3812 produkuje monofonní

zvuk, ale některé zvukové karty jako například Sound Blaster Pro propojovaly dva čipy dohromady pro stereofonní výstup.

6.2.1 Registry

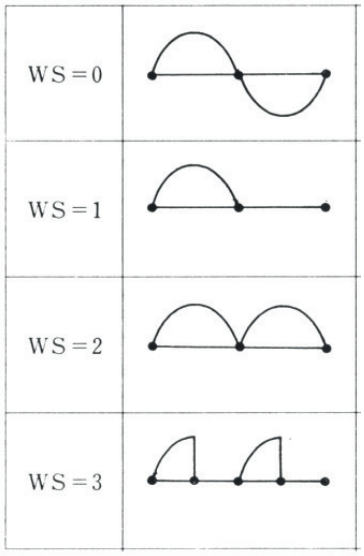
Tab. 6.7: Nové registry pro OPL2. Ostatní jsou stejné jako u OPL.

Adresa registru	BIT B7	BIT B6	BIT B5	BIT B4	BIT B3	BIT B2	BIT B1	BIT B0
0x01	LSI Test Register		WSEnable	LSI Test Register				
0xE0 - 0xF5						Waveform Select		

WSEnable

Pokud je v 0, budou všechny operátory používat sinusový průběh (jako u OPL). Pokud je nastaven na 1, bude použit průběh z registru Waveform Select.

Waveform Select



Obr. 6.11: Volba průběhů pro oscilátory. Sinusový, půl sinusový, absolutní sinusový a čtvrt sinusový. |Převzato z [32]|

6.2.2 YAMAHA OPLL

OPLL (YM2413) je zmenšená verze čipu OPL2. Aby se výroba čipu zlevnila, byly odstraněny některé registry. Má k dispozici 15 zvukových kanálu, ale v každém okamžiku lze přehrávat zvuk pouze na jednom kanálu, ostatní jsou zablokovány. Počet průběhů, které může generovat oscilátor se snížil na dva. Byl odstraněna aditivní syntéza. Výstupy kanálů nejsou míchány pomocí sčítačky, ale posílají se po určitých časových intervalech do D/A převodníku. Každý kanál má vyhrazený svůj časový interval (TDMA).

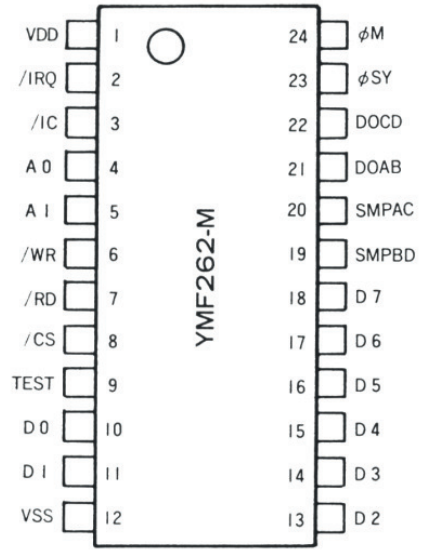
Byl použit v přídavné zvukové kartě pro Sega Mark III, v některých arkádových automatech (Sky Soldiers a Gang Wars) a v klávesách Yamaha PSS-170. Tato verze byla velmi rozšířená v Japonsku.

6.3 YAMAHA OPL3

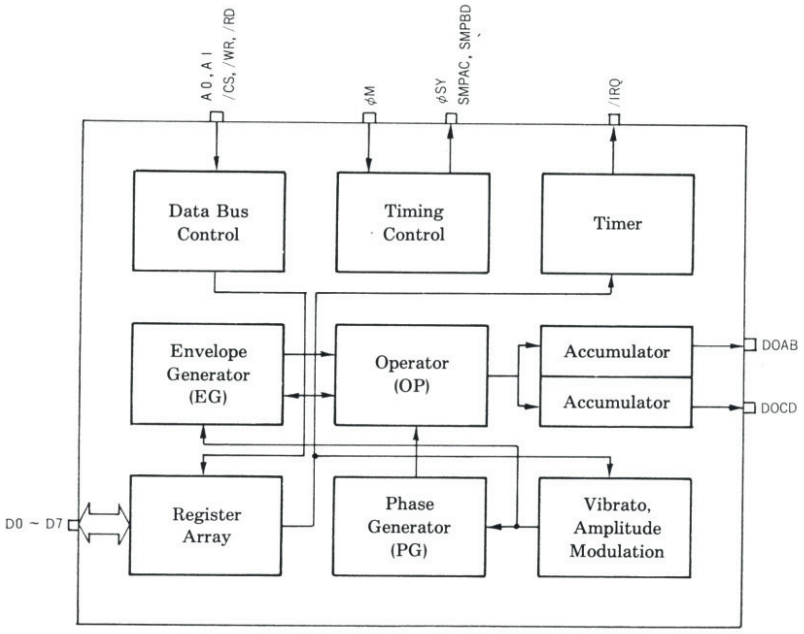
OPL3 (YMF262) je vylepšenou verzí OPL2. Má čtyřkanálový výstup (AB, CD) pro D/A převodník. Pro každou dvojici je potřeba jeden. To umožňuje vytvářet jednoduché stereo. Lze mít zapnuté všechny kanály nebo jenom jednu dvojici. Má dvakrát více kanálů (18 místo 9) než OPL2. V melodickém režimu jich máme k dispozici 18 (každý je tvořen dvěma operátory). V perkusním režimu je k dispozici až 15 melodických kanálů a 6 kanálů pro bicí nástroje (také po dvou operátorech). Nově můžeme v jednom kanálu provozovat 4 operátory. Díky tomu můžeme mít 6 melodických kanálů po 4 operátorech a 2 kanály po 2 operátorech nebo 6 melodických kanálů po 4 operátorech, 3 melodické kanály po 2 operátorech a 5 kanálů pro bicí nástroje po 4 operátorech.

Jsou přidány další průběhy pro oscilátory. Byla snížena latence pro výběr registru. Počet pinů zůstal stejný jako u OPL2, ale funkce některých se změnila. Nové piny jsou A1, SMPAC, SMPBD, DOAB, DOCD a ϕSY. SMPAC a SMPBD jsou synchronizační signály pro příslušný D/A převodník. ϕSY je synchronizační signál pro oba převodníky. DOAB a DOCD jsou výstupy dat pro D/A převodník. Data jsou posílána sériově po 16 bitech. Podle úrovně SMPAC se pozná, zda data určená pro převod patří kanálu A nebo C. Vzorkovací frekvence D/A převodníku byla snížena na 49,7 kHz.

Jelikož je k dispozici více operátorů, tak se musí zvětšit i počet registrů, které je ovládají. To je vyřešeno tak, že je paměť rozdělena na dvě banky registrů, mezi kterými lze přepínat pomocí bitů A1 a A0. Také došlo ke zvýšení frekvence hodin. Nový rozsah je od 10 MHz do 16 MHz. Typická frekvence je 14,32 MHz.



Obr. 6.12: Zapojení vývodů pro OPL3. |Převzato z [33]|



Obr. 6.13: Vnitřní zapojení bloků pro OPL3. Oproti OPL2 a OPL přibyl jeden akumulátor. |Převzato z [33]|

6.3.1 Registry

Jsou k dispozici dvě pamětové banky, mezi kterými lze přepínat pomocí bitu A1. Adresy registrů jsou kompatibilní s OPL2. Do prázdných pozic v mapě registrů jsou přidány nové funkce. Když je A1 = 0, tak ovládáme operátory 1 - 9. Pro ovládání operátorů 10 - 18 musí být A1 = 1.

Tab. 6.8: Mapa registrů pro A1 = 0.

A1 = 0								
Adresa registru	BIT B7	BIT B6	BIT B5	BIT B4	BIT B3	BIT B2	BIT B1	BIT B0
0x01	LSI Test Register		WSEnable	LSI Test Register				
0x02	Timer 1							
0x03	Timer 2							
0x04	IRQReset	T1 Mask	T2 Mask				T2 Start	T1 Start
0x08	CSW	Note-Sel						
0x20 - 0x35	Tremolo	Vibrato	Sustain	KSR	Frequency Multiplication Factor			
0x40 - 0x55	Key Scale Level		Output Level (Attenuation)					
0x60 - 0x75	Attack Rate				Decay Rate			
0x80 - 0x95	Sustain Level				Release Rate			
0xA0 - 0xA8	Frequency Number (Lower 8 bits)							
0xB0 - 0xB8			Key-On	Block Number			Frequency Number (high bits)	
0xBD	TremDep	VibrDep	PercMode	BD On	SD On	TT On	CY On	HH On
0xC0 - 0xC8	Channel A	Channel B	Channel C	Channel D	FeedBack Modulation Factor			SynthType
0xE0 - 0xF5						Waveform Select		

Tab. 6.9: Mapa registrů pro A1 = 1.

A1 = 1								
Adresa registru	BIT B7	BIT B6	BIT B5	BIT B4	BIT B3	BIT B2	BIT B1	BIT B0
0x01	LSI Test Register							
0x02								
0x03								
0x04			4-OP CH.12	4-OP CH.11	4-OP CH.10	4-OP CH.3	4-OP CH.2	4-OP CH.1
0x05								OPL3
0x08								
0x20 - 0x35	Tremolo	Vibrato	Sustain	KSR	Frequency Multiplication Factor			
0x40 - 0x55	Key Scale Level		Output Level (Attenuation)					
0x60 - 0x75	Attack Rate				Decay Rate			
0x80 - 0x95	Sustain Level				Release Rate			
0xA0 - 0xA8	Frequency Number (Lower 8 bits)							
0xB0 - 0xB8			Key-On	Block Number			Frequency Number (high bits)	
0xBD								
0xC0 - 0xC8	Channel A	Channel B	Channel C	Channel D	FeedBack Modulation Factor			SynthType
0xE0 - 0xF5						Waveform Select		

4-OP CH.12, 4-OP CH.11, 4-OP CH.10, 4-OP CH.3, 4-OP CH.2, 4-OP CH.1

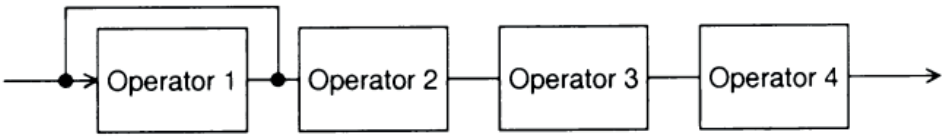
Pokud je některý z nich nastaven na 1, tak se na daném kanálu povolí čtyřoperátorový režim. Když je v 0, tak se používá dvouoperátorový režim pro dané kanály.

Channel A, Channel B, Channel C, Channel D

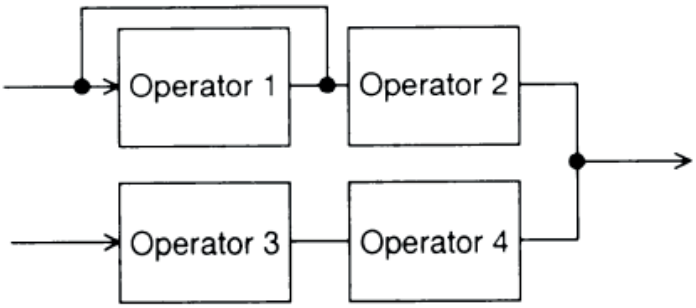
Pokud je některý z těchto bitů nastaven na 1, jsou data přivedena do příslušného kanálu. Bity CHA, CHB jsou vyvedeny na pin DOAB a CHC, CHD jsou vyvedeny na pin DOCD.

SynthType

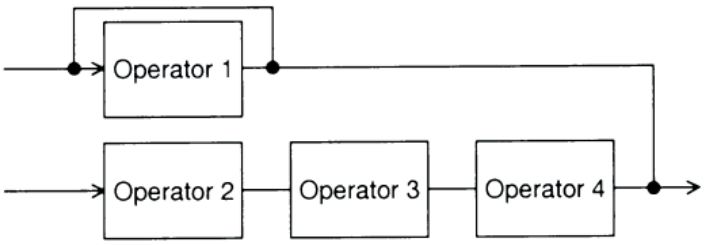
Přidáno ovládní čtyřoperátového režimu.



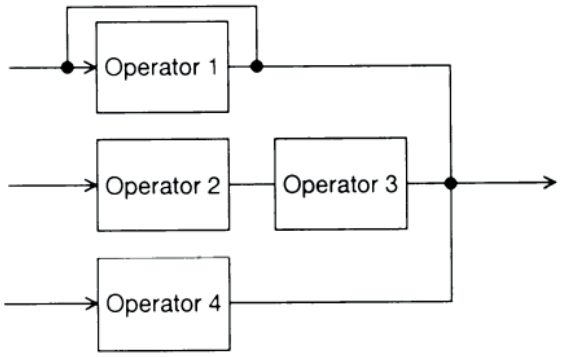
Obr. 6.14: Registr SynthType operátoru 1 nastavit na 0 a u operátoru 4 nastavit na 0. |Převzato z [33]|



Obr. 6.15: Registr SynthType operátoru 1 nastavit na 0 a u operátoru 4 nastavit na 1. |Převzato z [33]|



Obr. 6.16: Registr SynthType operátoru 1 nastavit na 1 a u operátoru 4 nastavit na 0. |Převzato z [33]|

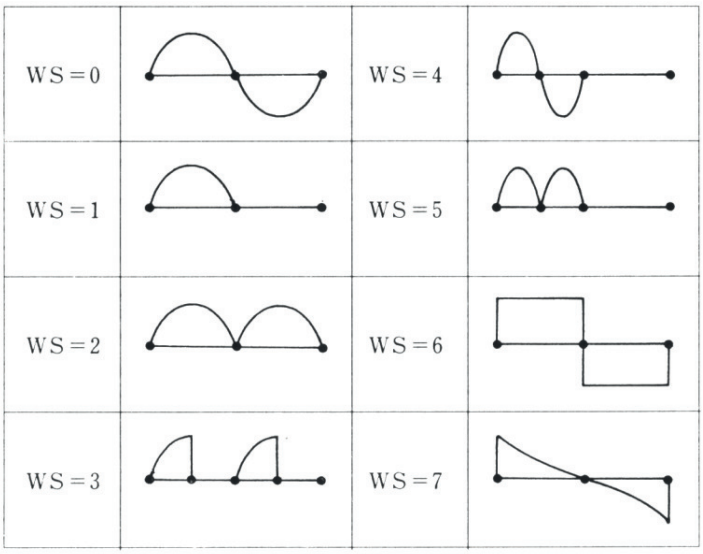


Obr. 6.17: Registr SynthType operátoru 1 nastavit na 1 a u operátoru 4 nastavit na 1. |Převzato z [33]|

OPL3

Nastavením na 1 se povolí režim OPL3 (36 operátorů, 8 průběhů pro oscilátory, stereo výstup). V nule se chová jako OPL2. Tento bit byl přidán kvůli kompatibilitě s OPL2.

Waveform Select



Obr. 6.18: Nové průběhy pro oscilátory. Sinusový - liché periody, absolutní sinusový - liché periody, obdélníkový a derivovaný obdélníkový průběh. |Převzato z [33]|

Tab. 6.10: Přiřazení adres operátorů skupině registrů pro melodický režim.

Kanál	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Operátor 1	0x00	0x01	0x02	0x08	0x09	0x0A	0x10	0x11	0x12	0x00	0x01	0x02	0x08	0x09	0x0A	0x10	0x11	0x12
Operátor 2	0x03	0x04	0x05	0x0B	0x0C	0x0D	0x13	0x14	0x15	0x03	0x04	0x05	0x0B	0x0C	0x0D	0x13	0x14	0x15

Tab. 6.11: Přiřazení adres operátorů skupině registrů pro perkusní režim.

Kanál	1	2	3	4	5	6	BD	SD	TT	CY	HH	12	13	14	15	16	17	18
Operátor 1	0x00	0x01	0x02	0x08	0x09	0x0A	0x10	0x14	0x12	0x15	0x11	0x02	0x08	0x09	0x0A	0x10	0x11	0x12
Operátor 2	0x03	0x04	0x05	0x0B	0x0C	0x0D	0x13					0x05	0x0B	0x0C	0x0D	0x13	0x14	0x15

Tab. 6.12: Přiřazení adres operátorů skupině registrů pro čtyřoperátorový melodický režim.

Kanál	1	2	3	7	8	9	10	11	12	16	17	18
Operátor 1	0x00	0x01	0x02	0x10	0x11	0x12	0x00	0x01	0x02	0x10	0x11	0x12
Operátor 2	0x03	0x04	0x05	0x13	0x14	0x15	0x03	0x04	0x05	0x13	0x14	0x15
Operátor 3	0x08	0x09	0x0A				0x08	0x09	0x0A			
Operátor 4	0x0B	0x0C	0x0D				0x0B	0x0C	0x0D			

Příklad přiřazení operátorů k registrům na kanále 2 ve čtyroperátorovém režimu.
4-OP CH.1 = 1 - Povolení čtyroperátorového režimu na kanále 1.
0x21 - Operator 1 - Tremolo/Vibrato/Sustain/KSR/Multiplication
0x24 - Operator 2 - Tremolo/Vibrato/Sustain/KSR/Multiplication
0x29 - Operator 3 - Tremolo/Vibrato/Sustain/KSR/Multiplication
0x2C - Operator 4 - Tremolo/Vibrato/Sustain/KSR/Multiplication
0x41 - Operator 1 - Key Scale Level/Output Level
0x44 - Operator 2 - Key Scale Level/Output Level
0x49 - Operator 3 - Key Scale Level/Output Level
0x4C - Operator 4 - Key Scale Level/Output Level
0x61 - Operator 1 - Attack Rate/Decay Rate
0x64 - Operator 2 - Attack Rate/Decay Rate
0x69 - Operator 3 - Attack Rate/Decay Rate
0x6C - Operator 4 - Attack Rate/Decay Rate
0x81 - Operator 1 - Sustain Level/Release Rate
0x84 - Operator 2 - Sustain Level/Release Rate
0x89 - Operator 3 - Sustain Level/Release Rate
0x8C - Operator 4 - Sustain Level/Release Rate
0xA1 - Frequency Number (low)
0xA4 - Nepoužito
0xB1 - Key On/Block Number/Frequency Number (high)
0xB4 - Nepoužito
0xC1 - FeedBack/Synthesis Type (část 1)
0xC4 - Synthesis Type (část 2)
0xE1 - Operator 1 - Waveform Select
0xE4 - Operator 2 - Waveform Select
0xE9 - Operator 3 - Waveform Select
0xEC - Operator 4 - Waveform Select

Tab. 6.13: Přiřazení adres operátorů skupině registrů pro čtyřoperátorový perkusní režim.

Kanál	1	2	3	BD	SD	TT	CY	HH	9	10	11	12	16	17	18
Operátor 1	0x00	0x01	0x02	0x10	0x14	0x12	0x15	0x11	0x12	0x00	0x01	0x02	0x10	0x11	0x12
Operátor 2	0x03	0x04	0x05	0x13					0x15	0x03	0x04	0x05	0x13	0x14	0x15
Operátor 3	0x08	0x09	0x0A							0x08	0x09	0x0A			
Operátor 4	0x0B	0x0C	0x0D							0x0B	0x0C	0x0D			

6.3.2 YAMAHA OPL3-L

OPL3-L (YMF289) je verze s nízkou spotřebou. Byla použita v některých zvukových kartách Sound Blaster 16. Je plně kompatibilní s OPL3, ale má několik odlišných vlastností. Napájecí napětí může být 5 V nebo 3,3 V. Ze všech registrů lze číst jejich aktuální hodnoty (v předchozích verzích se dalo číst pouze ze status registru). Vzorkovací frekvence je 44,1 kHz. Zabírá menší plochu na PCB ve srovnání s OPL3. Frekvence hodin byla zvýšena na 33,868 MHz.

6.4 YAMAHA OPL4

OPL4 (YMF278B) je zvukový čip, který umožnuje FM syntézu a nově také syntézu založenou na samplech (lze se setkat i s označením wavetable syntéza). FM syntéza se neliší od té na OPL3. Je zde stejný i počet operátorů a průběhů, které generují oscilátory. Syntéza pomocí samplů je založena na PCM modulaci. Samply (navzorkované zvuky hudebních nástrojů) musí být uloženy na externí paměti ROM nebo SRAM. Ty se uvnitř OPL4 dají různě upravovat. Pomocí této syntézy lze generovat až 24 simultánních zvuků. Je zvýšen počet výstupních kanálů pro D/A převodník na 6.

Adresy registrů pro řízení FM syntézy jsou stejné jako u OPL3. Vyráběl se v 80 pinovém pouzdru. Byl použit ve zvukové kartě Moonsound MSX a Yamaha SoundEdge pro IBM PC.

6.4.1 YAMAHA YMF278

Verze bez FM syntézy. Byl velmi často používán v elektronických klávesách (Yamaha PSS-51).

6.5 Formáty hudebních souborů

Stručný popis nejpoužívanějších formátů pro přehrávání hudby.

6.5.1 Ad Lib Music (MUS)

Ve formátu MUS [39] jsou uložen notový zápis a soubor SND [38](Ad Lib Sound Bank), který obsahuje definice hudebních nástrojů. V tomto ohledu je velmi podobný formátu ROL systému AdLib. Notový zápis je velmi podobný MIDI, ale má některé odlišnosti. Soubory MUS vypadají jako zkompilované soubory ROL a soubor SND je jako zkompilovaný soubor BNK[42].

6.5.2 ROL

Roll (ROL)[41][47][38] je zvukový formát od firmy Ad Lib, který obsahuje řídící instrukce pro FM syntézu. Je podobný notovému zápisu, protože obsahuje pouze příkazy pro přehrávání hudby, ale ne samotný zvuk. Soubory ROL musí být načteny společně se soubory hudebních nástrojů INS[38] (Ad Lib Instrument File). INS obsahuje definice pro jednotlivé nástroje, které se mají přehrávat. INS byl později nahrazen souborem banky nástrojů BNK (Ad Lib Instrument Bank). Ten se dal použít i pro formáty SX a SNG[38].

6.5.3 Ad Lib MIDI (MDI)

Formát MDI[50][48][37] je odvozen od formátu ROL. Používá standardní strukturu souboru MIDI (typ 0), která obsahuje události pro nastavení nástrojů OPL. Jedná se o způsob uložení nástrojů OPL uvnitř souboru MIDI. Definice hudebních nástrojů se nachází v souboru AdLib Timbre Bank Format[38].

Je podobný formátu CMF, který také ukládá hudbu OPL v notovém zápisu MIDI, ale na rozdíl od CMF se definice nástrojů nachází v samostatném souboru.

6.5.4 Creative Music Format (CMF)

Creative Music Format[35][45] (CMF) vytvořila společnost Creative Labs za účelem přehrávání hudby na svých zvukových kartách Sound Blaster vybavených čipem OPL. Struktura souboru je podobná formátům ROL společnosti Ad Lib, protože obsahuje všechny potřebné informace, které mají být odeslány do čipu OPL2. Na rozdíl od ROL soubory CMF obsahují definice nástrojů zabudované v souboru. To znamená, že není třeba soubor typu BNK, ale také to znamená, že soubory CMF jsou ze své podstaty větší než soubory ROL. Přehrávání hudby pomocí CMF nijak nezatěžovalo procesor, protože syntéza probíhá přímo na OPL.

Hudební soubory jsou uloženy ve formátu MIDI, což umožňuje snadný převod mezi soubory .mid a .cmf. Formát CMF také přišel s využitím MIDI kontrolérů, ty se vy-

užívají k řízení OPL (například k přepínání mezi devítikanálovým a šestikanálovým + pětiperkusním režimem).

6.5.5 id Software Music Format (IMF)

Formát id Music[46][36] je zvukový formát, který byl navržen, tak aby hudba šla přehrávat na kartách Ad Lib a Sound Blaster (obě používaly OPL). Tento formát ukládá číslo registru a hodnotu, která do něj má být zapsána. Z tohoto důvodu se jedná o jednoduše zpracovatelný formát.

Existují dva typy formátů IMF. První se nazývá Type-0. Type-0 IMF neobsahuje žádné hlavičkové informace a je odesílán přímo do čipu OPL2. Pozdější verze Type-1 má krátkou hlavičku s informacemi o skladbě. IMF neukládá informaci o frekvenci přehrávání. Nejběžnější jsou frekvence pro přehrávání skladeb jsou 560 Hz a 700 Hz. Existovaly i skladby, které používaly frekvenci 280 Hz.

Další formáty, které ukládaly adresy registrů a jejich hodnoty jsou: DRO Format[53] a RAW Format[52].

6.5.6 MUS Format

Formát MUS[51][40] je hudební formát, který je velice podobný MIDI. Původně jej vytvořil Paul Radek pro svou zvukovou knihovnu DMX. Tuto knihovnu použila společnost id Software pro hru Doom a několik dalších her.

Data jsou téměř totožná se standardním MIDI, ale bajty jsou uspořádány ve velmi odlišné a prostorově mnohem úspornější struktuře. Dokáže přehrávat pouze jednu stopu zvukových instrukcí, ale stopy dokáže přehrávat až na 16 kanálech. Tento formát byl používán ve všech hrách odvozených od Doomu až do poloviny 90. let.

7

Závěr

V práci byly popsány programovatelné zvukové generátory řady AY-3-891x a YAMAHA OPLx. U obou generátorů byly popsány významy registrů a jejich použití. U každé verze jsou uvedeny příklady použití a odlišnosti oproti starším verzím. Pro každý generátor jsou uvedeny příklady zvukových formát. U každého je uveden krátký popis. Formát YM je popsán více do hloubky. Bohužel se z časových důvodů nedostalo na popis jeho speciálních efektů.

Praktická realizace probíhala na generátoru AY-3-8912, který byl řízen mikroprocesorem STM32F411RE. Ten je součástí desky STM32 Nucleo F411RE. Program, který byl vytvořen řídí jednotlivé funkční bloky AY-3-8912 a umožňuje přehrávat zvukový formát YM. Byly vyzkoušeny dva výstupní obvody, pro připojení k reproduktoru. První byl převzatý z počítače Vectrex. Druhý je vlastní návrh předzesilovače pro tento obvod. Oba obvody byly úspěšně sestaveny na nepájivém poli a vyzkoušeny na reproduktorech Sony SRS-XB12 a Genius SP-S110. Kvalitnější a čistější zvuk produkovalo zapojení z počítače Vectrex. Mnou navržený předzesilovač fungoval, ale v některých pasážích skladby zvuk působil ořezaně. Regulace hlasitosti také nefungovala úplně dobře, protože při snaze otáčet potenciometrem se hlasitost měnila mezi její maximální a minimální hodnotou. To způsobovali vodiče, které měly špatný kontakt s potenciometrem. Jelikož jsem neměl náhradní, tak jsem je musel umístit do specifické polohy, ve které měly kontakt s potenciometrem. Regulace hlasitosti při této úpravě fungovala. Po úpravě dalších vodičů se kvalita reprodukce zvuku významně zlepšila, ale pořád nedosahovala kvality obvodu z počítače Vectrex. Bohužel se z časových důvodů nedostalo na detailní popis návrhu předzesilovače a na výpočet hodnot rezistorů, kondenzátorů a zesílení součtového zesilovače.

Pro generátor YAMAHA OPL3 bylo vypracováno schéma pro připojení k mikrokontroléru a D/A převodníkům.

Literatura

[1] Microchip Technology Inc. [online katalogový list]. *AY-3-891x Microchip Manual* Dostupné z: http://www.vgmpf.com/Wiki/images/6/6b/AY-3-891x_-_Microchip_Manual.pdf

[2] Video Game Music Preservation Foundation [online]. *AY-3-8913 - Pin Out* Dostupné z: http://www.vgmpf.com/Wiki/images/2/2c/AY-3-8913_-_Pin_Out.png

[3] Video Game Music Preservation Foundation [online]. *AY-3-8910* Dostupné z: http://www.vgmpf.com/Wiki/index.php?title=AY-3-8910#AY-3-8910

[4] Video Game Music Preservation Foundation [online]. *AY-3-8910* Dostupné z: http://www.vgmpf.com/Wiki/index.php?title=AY-3-8910#AY-3-8912

[5] Video Game Music Preservation Foundation [online]. *AY-3-8910* Dostupné z: http://www.vgmpf.com/Wiki/index.php?title=AY-3-8910#AY-3-8913

[6] Video Game Music Preservation Foundation [online]. *AY-3-8910* http://www.vgmpf.com/Wiki/index.php?title=AY-3-8910#AY-3-8914

[7] Video Game Music Preservation Foundation [online]. *AY-3-8910* Dostupné z: http://www.vgmpf.com/Wiki/index.php?title=AY-3-8910#AY-3-8916

[8] Video Game Music Preservation Foundation [online]. *AY-3-8910* Dostupné z: http://www.vgmpf.com/Wiki/index.php?title=AY-3-8910#AY-3-8917

[9] Video Game Music Preservation Foundation [online]. *AY-3-8910* Dostupné z: http://www.vgmpf.com/Wiki/index.php?title=AY-3-8910#YM2149

[10] Video Game Music Preservation Foundation [online]. *AY-3-8910* Dostupné z: http://www.vgmpf.com/Wiki/index.php?title=AY-3-8910#AY8930.2FP

[11] TIŠNOVSKÝ Pavel, ROOT.CZ_ [online]. *Zvukový čip AY-3-8910 (YM2149)* Dostupné z: https://www.root.cz/clanky/zvukovy-cip-ay-3-8910-ym2149/

[12] TIŠNOVSKÝ Pavel, ROOT.CZ_ [online]. *Práce se zvukovým čipem AY-3-8910 (YM2149)* Dostupné z: https://www.root.cz/clanky/prace-se-zvukovym-cipem-ay-3-8910-ym2149/

[13] Video Game Music Preservation Foundation [online]. *PSG* Dostupné z: https://wiki.intellivision.us/index.php/PSG#Registers

[14] AtariAge [online]. *AY-3-891x Sound Chips: Pin Assignment Differences?* Dostupné z: https://atariage.com/forums/topic/163248-ay-3-891x-sound-chips-pin-assignment-differences/

[15] Wikipedia [online]. *General Instrument AY-3-8910* Dostupné z: https://en.wikipedia.org/wikiGeneral_Instrument_AY-3-8910#Variants

[16] Wikipedia [online]. *General Instrument AY-3-8910* Dostupné z: https://en.wikipedia.org/wiki/General_Instrument_AY-3-8910

[17] General Instruments [online katalogový list]. *AY-3-8910/8912 PROGRAMMABLE SOUND GENERATOR DATA MANUAL* Dostupné z: https://f.rdw.se/AY-3-8910-datasheet.pdf

[18] BURNETT John, LENARD AUDIO INSTUTE_ [online]. *Amplifier parameters* Dostupné z: https://education.lenardaudio.com/en/12_amps_3.html

[19] TIŠNOVSKÝ Pavel, ROOT.CZ_ [online]. *Hudba a zvuk na PC: PC Speaker, Covox a Adlib* Dostupné z: https://www.root.cz/clanky/hudba-a-zvuk-na-pc-pc-speaker-covox-a-adlib/

[20] TIŠNOVSKÝ Pavel, ROOT.CZ_ [online]. *Hudební čipy Yamaha YM 3812 (OPL2) a YMF 262 (OPL3)* Dostupné z: https://www.root.cz/clanky/hudebni-cipy-yamaha-ym-3812-opl2-a-ymf-262-opl3/

[21] Wikipedia [online]. *Yamaha OPL* Dostupné z: https://en.wikipedia.org/wiki/Yamaha_OPL

[22] Wikipedia [online]. *Yamaha YMF278* Dostupné z: https://en.wikipedia.org/wiki/Yamaha_YMF278

[23] Yamaha Wiki [online]. *Yamaha YM3526* Dostupné z: https://www.yamaha-tech.com/wiki/Yamaha_YM3526

[24] Yamaha Wiki [online]. *Yamaha YM3812* Dostupné z: https://www.yamaha-tech.com/wiki/Yamaha_YM3812

[25] Yamaha Wiki [online]. *Yamaha YMF262* Dostupné z: https://www.yamaha-tech.com/wiki/Yamaha_YMF262

[26] Video Game Music Preservation Foundation [online]. *YM3526* Dostupné z: http://www.vgmpf.com/Wiki/index.php?title=YM3526

[27] Video Game Music Preservation Foundation [online]. *YM3812* Dostupné z: http://www.vgmpf.com/Wiki/index.php?title=YM3812

[28] Video Game Music Preservation Foundation [online]. *YMF262* Dostupné z: http://www.vgmpf.com/Wiki/index.php?title=YMF262

[29] Wikipedia [online]. *Yamaha YM2413* Dostupné z: https://en.wikipedia.org/wiki/Yamaha_YM2413

[30] ARNOŠT Vladimír [online]. *Programmer's Guide to Yamaha YMF 262/OPL3 FM Music Synthesizer* Dostupné z: https://i.iinfo.cz/urs-att/opl3_c-124223821727341.txt

[31] Yamaha Corporation [online katalogový list]. *YM3526 Manual* Dostupné z: http://www.vgmpf.com/Wiki/images/6/6e/YM3526_-_Manual.pdf

[32] Yamaha Corporation [online katalogový list]. *YM3812 Manual* Dostupné z: http://www.vgmpf.com/Wiki/images/f/fe/YM3812_-_Manual.pdf

[33] Yamaha Corporation [online katalogový list]. *YMF262 Manual* Dostupné z: http://www.vgmpf.com/Wiki/images/2/27/YMF262_-_Manual.pdf

[34] Yamaha Corporation [online katalogový list]. *OPL4 Manual* Dostupné z: http://www.msxarchive.nl/pub/msx/docs/datasheets/opl4.pdf

[35] Video Game Music Preservation Foundation [online]. *CMF* Dostupné z: http://www.vgmpf.com/Wiki/index.php?title=CMF

[36] Video Game Music Preservation Foundation [online]. *IMF* Dostupné z: http://www.vgmpf.com/Wiki/index.php?title=IMF

[37] Video Game Music Preservation Foundation [online]. *MDI* Dostupné z: http://www.vgmpf.com/Wiki/index.php?title=MDI

[38] Video Game Music Preservation Foundation [online]. *SND (AdLib)* Dostupné z: http://www.vgmpf.com/Wiki/index.php?title=SND_(AdLib)

[39] Video Game Music Preservation Foundation [online]. *MUS (AdLib)* Dostupné z: http://www.vgmpf.com/Wiki/index.php?title=MUS_(AdLib)

[40] Video Game Music Preservation Foundation [online]. *MUS* Dostupné z: http://www.vgmpf.com/Wiki/index.php?title=MUS

[41] Video Game Music Preservation Foundation [online]. *ROL* Dostupné z: http://www.vgmpf.com/Wiki/index.php?title=ROL

[42] Video Game Music Preservation Foundation [online]. *BNK* Dostupné z: http://www.vgmpf.com/Wiki/index.php?title=BNK

[43] Video Game Music Preservation Foundation [online]. *INS* Dostupné z: http://www.vgmpf.com/Wiki/index.php?title=INS

[44] Video Game Music Preservation Foundation [online]. *ROL* Dostupné z: http://www.vgmpf.com/Wiki/index.php?title=ROL

[45] DOS Game Modding Wiki [online]. *CMF Format* Dostupné z: https://moddingwiki.shikadi.net/wiki/CMF_Format#Default_instruments

[46] DOS Game Modding Wiki [online]. *IMF Format* Dostupné z: https://moddingwiki.shikadi.net/wiki/IMF_Format

[47] DOS Game Modding Wiki [online]. *ROL Format* Dostupné z: https://moddingwiki.shikadi.net/wiki/ROL_Format

[48] DOS Game Modding Wiki [online]. *AdLib MIDI Format* Dostupné z: https://moddingwiki.shikadi.net/wiki/AdLib_MIDI_Format

[49] DOS Game Modding Wiki [online]. *AdLib Timbre Bank Format* Dostupné z: https://moddingwiki.shikadi.net/wiki/AdLib_Timbre_Bank_Format

[50] DOS Game Modding Wiki [online]. *MDI Format* Dostupné z: https://moddingwiki.shikadi.net/wiki/MDI_Format

[51] DOS Game Modding Wiki [online]. *MUS Format* Dostupné z: https://moddingwiki.shikadi.net/wiki/MUS_Format

[52] DOS Game Modding Wiki [online]. *RAW Format* Dostupné z: https://moddingwiki.shikadi.net/wiki/RAW_Format_(Adlib)

[53] DOS Game Modding Wiki [online]. *DRO Format* Dostupné z: https://moddingwiki.shikadi.net/wiki/DRO_Format

[54] VIEIRA Sérgio [online]. *Using an AY-3-8910 programmable sound generator with an Arduino* Dostupné z: https://www.youtube.com/watch?v=srmNbi9yQNU

[55] VIEIRA Sérgio [online]. *Using an AY-3-8910 programmable sound generator with an Arduino* Dostupné z: https://github.com/internalregister/AY-3-8910

[56] GitHub.com [online]. *ym-file-format.txt* Dostupné z: https://github.com/skeezix/zikzak/blob/master/zik80/audio-gen/ym-file-format.txt

[57] CARRÉ Arnaud [online]. *YM File format* Dostupné z: http://leonard.oxg.free.fr/ymformat.html

[58] Video Game Music Preservation Foundation [online]. *YM* Dostupné z: `http://www.vgmpf.com/Wiki/index.php?title=YM`

[59] VIEIRA Sérgio [online]. *ymConverter.py* Dostupné z: `https://github.com/internalregister/AY-3-8910/blob/master/ymConverter.py`

[60] tomeko.net [online]. *File to hexadecimal converter* Dostupné z: `https://tomeko.net/online_tools/file_to_hex.php?lang=en`

[61] bulba.untergrund.net [online]. *YM Archive v5* Dostupné z: `https://bulba.untergrund.net/YM_Archive_v5.7z`

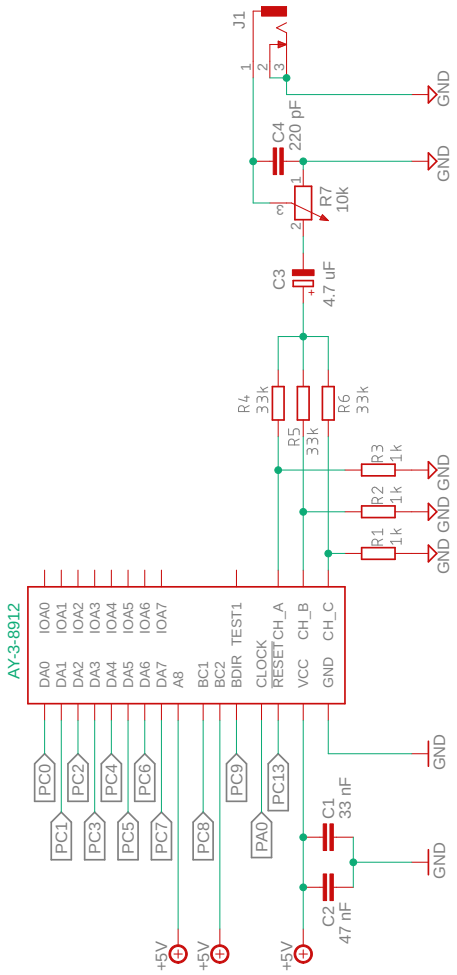
[62] Console5 Tech Wiki [online]. *Vectrex* Dostupné z: `https://console5.com/techwiki/images/d/d8/Vectrex---Logic-Board-Schematic.png`

[63] STMicroelectronics N.V. [online katalogový list]. *STM32 Nucleo-64 boards (MB1136)* Dostupné z: `https://www.st.com/content/ccc/resource/technical/document/user_manual/98/2e/fa/4b/e0/82/43/b7/DM00105823.pdf/files/DM00105823.pdf/jcr:content/translations/en.DM00105823.pdf`

Příloha A

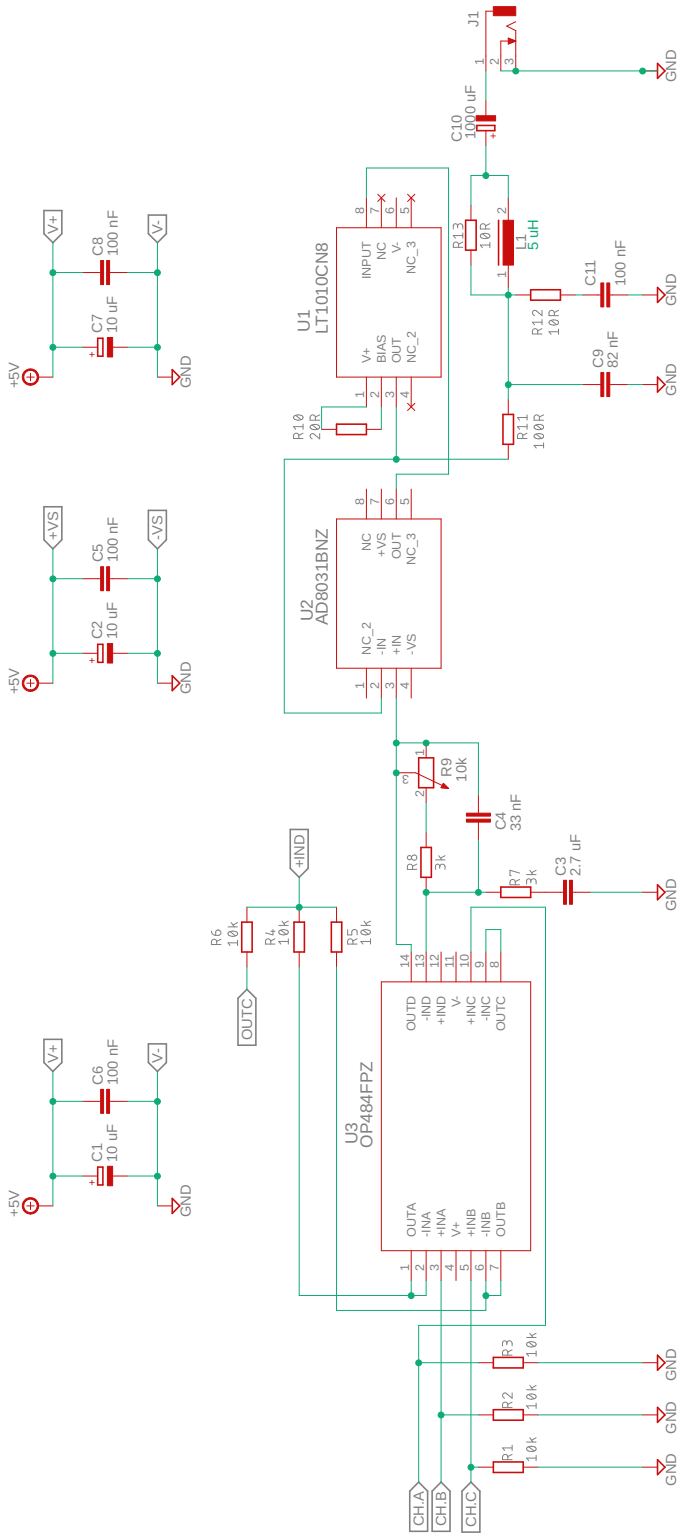
Schémata zapojení

A.1 AY-3-8912 s výstupním obvodem z počítače Vectrex



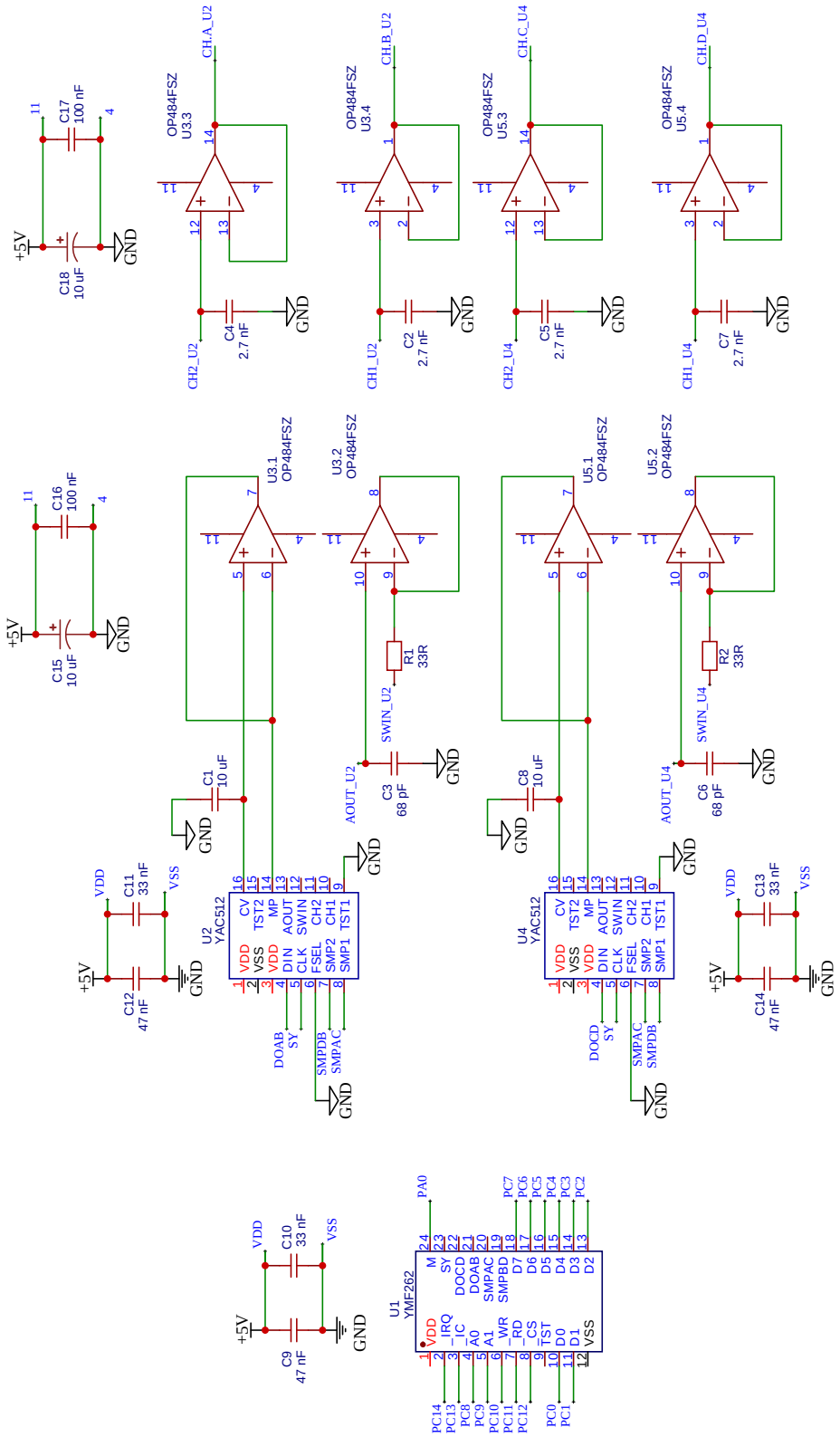
Obr. A.1: Propojení STM32F411RE s AY-3-8912 a zapojení výstupního obvodu.

A.2 Návrh předzesilovače pro AY-3-8912



Obr. A.2: Zapojení předzesilovače. Nad každým blokem s operačními zesilovači je tantalový a keramický kondenzátor pro napájení.

A.3 Připojení OPL3 k mikrokontroléru



Obr. A.3: Vstupy OPL3 jsou připojeny k STM32F411RE a výstupy jsou připojeny k D/A převodníkům YAC512. Na výstupy CH. A, CH. B, CH. C a CH. D se připojí zesilovač.

Příloha B

Použité skripty, zdrojové kódy

B.1 Soubor funkce.c

```
1  // vyber frekvence pro AY-3-8912
2  //#define CLOCK_1MHZ
3  #define CLOCK_2MHZ
4  volatile uint32_t _ticks = 0;   // promenna pro casovac SysTick
```

B.1.1 Funkce clock_init

```
1  void clock_init(void)                        // nastaveni citace pro generovani 1 MHz hodinoveho signalu
2  {
3    puts("Inicializace CLK....");
4
5    // povoleni hodin k casovaci TIM5_CH1 PA0
6    if (!(RCC->APB1ENR & RCC_APB1ENR_TIM5EN))   // kontrola aktivace hodin k casovaci
7    {
8      RCC->APB1ENR |= RCC_APB1ENR_TIM5EN;       // povoleni hodiny periferie
9      RCC->APB1RSTR |= RCC_APB1RSTR_TIM5RST;    // proved reset periferie
10     RCC->APB1RSTR &= ~RCC_APB1RSTR_TIM5RST;   // a konec resetu
11   }
12
13   // aktivace hodin portu A a reset
14   if (!(RCC->AHB1ENR & RCC_AHB1ENR_GPIOAEN))  // kontrola aktivace hodin hodin portu A
15   {
16     RCC->AHB1ENR |= RCC_AHB1ENR_GPIOAEN;      // povolit hodiny periferie
17     RCC->AHB1RSTR |= RCC_AHB1RSTR_GPIOARST;   // proved reset periferie
18     RCC->AHB1RSTR &= ~RCC_AHB1RSTR_GPIOARST;  // a konec resetu
19   }
20
21   // nastaveni konfiguracnich bitu do defaultniho stavu (nemususelo byt od drive)
22   // 0x03 =  0011
23   GPIOA->MODER &= ~(0x03);                    // vynuluj bity v registru
24   GPIOA->PUPDR &= ~(0x03);                    // vynuluj bity v registru
25   GPIOA->OSPEEDR &= ~(0x03);                  // vynuluj bity v registru
26
27   // 0x02 = 0010
28   // 0x01 = 0001
29   GPIOA->MODER |= 0x02;                       // 10 = alternativni fuknce pinu PA_0
30   GPIOA->OTYPER &= ~(0x01);                   // 0 = push-pull vystup
31   GPIOA->OSPEEDR |= 0x03;                     // 11 = high speed (rychlost zmen na pinu)
32   GPIOA->PUPDR &= ~(0x03);                    // 00 = vypnute pull-up/pull-down rezistory
33
34   // 0x000F = 0000 0000 0000 1111
35   // 0x0002 = 0000 0000 0000 0010
36   GPIOA->AFR[0] &= ~(0x000F);                 // vynulovani registru AFR_L
37   GPIOA->AFR[0] |= 0x0002;                    // 0010 = nastav alternativni fukci AF2 na pinu PA_0
38
39   TIM5->CR1 = TIM_CR1_ARPE;                   // DIR = 0 -> citac nahoru, bufferovany zapis do ARR
40   TIM5->CR2 = 0;                              // nic nenastavujeme
```

```
41
42
43 #ifdef CLOCK_1MHZ
44   TIM5->PSC = 8 - 1;                          // 16 MHz / 8 = 2 MHz (0.5 us) -> rychlost citani (jednoho kroku) citace
45   TIM5->ARR = 2 - 1;                          // maximalni hodnota citace
46 #endif
47
48 #ifdef CLOCK_2MHZ
49   TIM5->PSC = 4 - 1;                          // 16 MHz / 4 = 4 MHz (0.25 us) -> rychlost citani (jednoho kroku) citace
50   TIM5->ARR = 2 - 1;                          // maximalni hodnota citace
51 #endif
52
53   TIM5->CCMR1 &= ~TIM_CCMR1_OC1M;             // vynulovani OC3m
54   TIM5->CCMR1 |= TIM_CCMR1_OC1M_2 | TIM_CCMR1_OC1M_1;   // 110 = PWM mode 1
55   TIM5->CCER |= TIM_CCER_CC1E;                // povoleni vystupu OC1 na pin PA_0
56   TIM5->CCR1 = 1;                          //  1 ->  strida 50%
57
58   TIM5->CR1 = TIM_CR1_CEN;                    // aktivace citace (Counter Enable), poslední akce
59
60   puts("OK\n");
61 }
62
63 \subsection{Funkce bus\textunderscore init}
64 \begin{Verbatim}[fontsize=\scriptsize,numbers=left,numbersep=7pt,baselinestretch=1,xleftmargin=7pt]
65 void bus_init(void)                           // nastaveni sbernice - DA0-DA7, BDIR, BC2
66 {
67   puts("Inicializace BUS....");
68   // aktivace hodin portu C a reset
69   if (!(RCC->AHB1ENR & RCC_AHB1ENR_GPIOCEN))  // kontrola aktivace hodin hodin portu C
70   {
71     RCC->AHB1ENR |= RCC_AHB1ENR_GPIOCEN;      // povoleni hodiny periferie
72     RCC->AHB1RSTR |= RCC_AHB1RSTR_GPIOCRST;   // proved reset periferie
73     RCC->AHB1RSTR &= ~RCC_AHB1RSTR_GPIOCRST;  // a konec resetu
74   }
75
76   // nastaveni konfiguracnich bitu do defaultniho stavu (nemususelo byt od drive)
77   // 0x000F_FFFF = 0000 0000 1111 1111 1111 1111 1111
78   GPIOC->MODER &= ~(0x000FFFFF);              // vynuluje  bity v registru,
79   GPIOC->PUPDR &= ~(0x000FFFFF);              // vynuluje  bity v registru
80   GPIOC->OSPEEDR &= ~(0x000FFFFF);            // vynuluje  bity v registru
81
82   // nastaveni pinu na portu C
83   //0x055_5555 = 0000 0101 0101 0101 0101 0101 0101
84   //0x000F_FFFF = 0000 0000 1111 1111 1111 1111 1111
85   //0x03FF = 0000 0011 1111 1111
86   GPIOC->MODER |= 0x0055555;                  // 01 = vystupni orientace pinu
87   GPIOC->OSPEEDR |= 0x000FFFFF;               // 11 = high speed (rychlost zmen na pinech)
88   GPIOC->PUPDR &= ~(0x000FFFFF);              // 00 = vypnute pull-up/pull-down rezistory
89   GPIOC->OTYPER &= ~(0x03FF);                 // 0 = push-pull vystup
90
91   puts("OK\n");
92 }
```

B.1.2 Funkce reset_init

```
1 void reset_init(void)                         // nastaveni reset tlacitka na pinu PC_13
2 {
3   puts("\nInicializace RESETu....");
4   // aktivace hodin portu C a reset
5   if (!(RCC->AHB1ENR & RCC_AHB1ENR_GPIOCEN))   // kontrola aktivace hodin hodin portu C
6   {
7     RCC->AHB1ENR |= RCC_AHB1ENR_GPIOCEN;       // povolit hodiny periferie
8     RCC->AHB1RSTR |= RCC_AHB1RSTR_GPIOCRST;    // proved reset periferie
9     RCC->AHB1RSTR &= ~RCC_AHB1RSTR_GPIOCRST;   // a konec resetu
10  }
11
12  // nastaveni konfiguracnich bitu do defaultniho stavu (nemususelo byt od drive)
13  // 0x0C00_0000 = 1100 0000 0000 0000 0000 0000 0000
14  GPIOC->MODER &= ~(0x0C000000);              // vynuluj bity v registru,
15  GPIOC->PUPDR &= ~(0x0C000000);              // vynuluj bity v registru
16  GPIOC->OSPEEDR &= ~(0x0C000000);            // vynuluj bity v registru
```

```
17
18   // prvotni reset na nastaveni resetovaciho tlacitka na pinu PC_13
19   GPIOC->BSRR |= PC_13_RESET;      // nastaveni reset casti do 1 -> PC13 = 0 -> reset AY-891x (je aktivni na log. 0)
20   GPIOC->BSRR |= PC_13_SET;                  // nastaveni set casti do 1 -> PC13 = 1 (set ma vyssi prioritu ne reset)
21   puts("OK\n");
22 }
```

B.1.3 Funkce read_reset_btn

```
1  bool read_reset_btn(void)                    // cteni hodnoty reset tlacika
2  {
3    //puts("Cteni stavu tlacitka RESET");
4    return ((GPIOC->IDR & (1 << 13)) != 0);     // defaultni stav IDR_13 = 1 (nezmacknute tlacitko) -> 1 != 0 -> true
5    // zmacknute tlacitko IDR_13 = 0 -> 0 != 0 -> false
6    //puts("Cteni stavu dokonceno");
7  }
```

B.1.4 Funkce bus_inactive

```
1  void bus_inactive(void)                      // nastaveni sbernice do neaktivniho stavu
2  {
3    GPIOC->BSRR |= (BDIR_RESET | BC1_RESET);    // reset pinu  BDIR = 0, BC1 = 0
4    //puts("BUS je neaktivni");
5  }
```

B.1.5 Funkce bus_write

```
1  void bus_write(void)                         //priprav sbernici pro zapis hodnoty do registru
2  {
3    GPIOC->BSRR |= BDIR_SET;                   // nastavi BDIR = 1
4    //puts("BUS je v rezimu zapisu do registru");
5  }
```

B.1.6 Funkce bus_latch_adress

```
1  void bus_latch_adress(void)                  // priprav sbernici pro vyber registru
2  {
3    GPIOC->BSRR |= (BC1_SET | BDIR_SET);       // nastavi BDIR = 1, BC1 = 1
4    //puts("BUS je v rezimu vyberu registru");
5  }
```

B.1.7 Funkce bus_reset

```
1  void bus_reset(void)                         // nastavi DA0 -DA7 na nulu
2  {
3    GPIOC->BSRR |= (DA0_DA7_RESET);            // reset pinu DA0 - DA7 = 0
4    //puts("DA_0 az DA_7 jsou resetovany");
5  }
```

B.1.8 Funkce zapis_do_registru

```
1  void zapis_do_registru(uint8_t registr, uint8_t hodnota) // vyber a zapis hodnoty do registru
2  {
3   // omezeni cisla registru na max. 15 kvuli hornim 4 bitum sbernice, ktere musi byt pri vyberu registru na nule
4    if((registr >=0) && (registr <= 15))
5    {
6      //puts("Vyber registru");
7
8      bus_latch_adress();                        // priprav sbernici pro vyber registru
9
10     GPIOC->BSRR = registr;
11     //puts("Registr vybran");
```

```
12
13      bus_inactive();                         // nastaveni sbernice do neaktivniho stavu
14
15      bus_reset();                            // nastavi DA0 -DA7 na nulu
16
17      //puts("\nZapis dat do registru");
18      GPIOC->BSRR = hodnota;
19      //puts("Data pro zapis jsou na BUS");
20
21      bus_write();                            //priprav sbernici pro zapis hodnoty do registru
22      //puts("Data zapsana do registru");
23
24      bus_inactive();                         // nastaveni sbernice do neaktivniho stavu
25
26      bus_reset();                            // nastavi DA0 -DA7 na nulu
27
28      //puts("Vyber a ulozeni dokonceno\n");
29    }
30  }
```

B.1.9 Funkce vyber_kanalu_tonu_noise

```
1  void vyber_kanalu_tonu_noise(kanaly_ton channel, noise povolit_noise)
2  {
3    uint8_t x = 0xFF;              // promenna pomoci ktere se bude vybirat kanal, default stav je vsechny kanaly vypnute
4
5    {
6      switch(channel)              // vyber kanalu, nastaveni spodnich trech bitu
7      {
8        case CHANNEL_TONE_NONE:    // vsechny kanaly vypnute
9          zapis_do_registru(ENABLE_NOISE_TONE, 0xFF);
10         break;
11
12       case CHANNEL_TONE_A:       // zapnout kanal A
13         x &= ~(1 << 0);          // nastaveni nuly na danou pozici
14         puts("Vybran kanal A");  // vypis na seriovy kanal pomoci USARTu
15         break;
16
17       case CHANNEL_TONE_B:       // zapnout kanal B
18         x &= ~(1 << 1);          // nastaveni nuly na danou pozici
19         puts("Vybran kanal B");
20         break;
21
22       case CHANNEL_TONE_C:       // zapnout kanal C
23         x &= ~(1 << 2);          // nastaveni nuly na danou pozici
24         puts("Vybran kanal C");
25         break;
26
27       case CHANNEL_TONE_AB:       // zapnout kanal A, B
28         x &= ~((1 << 0) | (1 << 1));
29         puts("Vybran kanal A, B");
30         break;
31
32       case CHANNEL_TONE_AC:       // zapnout kanal A, C
33         x &= ~((1 << 0) | (1 << 2));
34         puts("Vybran kanal A, C");
35         break;
36
37       case CHANNEL_TONE_BC:       // zapnout kanal B, C
38         x &= ~((1 << 1) | (1 << 2));
39         puts("Vybran kanal B, C");
40         break;
41
42       case CHANNEL_TONE_ABC:       // zapnout kanal A, B, C
43         x &= ~((1 << 0) | (1 << 1) | (1 << 2));
44         puts("Vybran kanal A, B, C");
45         break;
46     }
47
48     if(povolit_noise)              // povoleni generatoru sumu na danem kanalu, nastaveni hornich trech bitu
49     {
```

```
      switch(channel)
      {
        case CHANNEL_TONE_NONE:     // neni povolen generator sum
          break;

        case CHANNEL_NOISE_A:       // generator sumu na kanalu A
          x &= ~(1 << 3);           // nastaveni nuly na danou pozici
          break;

        case CHANNEL_NOISE_B:       // generator sumu na kanalu B
          x &= ~(1 << 4);
          break;

        case CHANNEL_NOISE_C:       // generator sumu na kanalu C
          x &= ~(1 << 5);
          break;

        case CHANNEL_NOISE_AB:       // generator sumu na kanalu A, B
          x &= ~((1 << 3) | (1 << 4));
          break;

        case CHANNEL_NOISE_AC:       // generator sumu na kanalu A, C
          x &= ~((1 << 3) | (1 << 5));
          break;

        case CHANNEL_NOISE_BC:       // generator sumu na kanalu B, C
          x &= ~((1 << 4) | (1 << 5));
          break;

        case CHANNEL_NOISE_ABC:       // generator sumu na kanalu A, B, C
          x &= ~((1 << 3) | (1 << 4) | (1 << 5));
          break;
      }
    }
  }
  zapis_do_registru(ENABLE_NOISE_TONE, x);  // zapis vybranych kanalu do registru
}
```

B.1.10 Funkce nastaveni_frekvence_tonu

```
void nastaveni_frekvence_tonu(kanaly_ton channel, uint32_t frekvence)
{
#ifdef CLOCK_1MHZ
  if((frekvence >= 16) && (frekvence <= 62500)) // limitni frekvence ktere lze generovat pri 1 MHz CLK
#endif

#ifdef CLOCK_2MHZ
    if((frekvence >= 31) && (frekvence <= 125000))  // limitni frekvence ktere lze generovat pri 2 MHz CLK
#endif
    {
      uint32_t coarse_hodnota = 0;          // promenna pro ulozeni hodnoty registru R1, R3 a R5
      uint32_t fine_hodnota = 0;            // promenna pro ulozeni hodnoty registru R0, R2 a R4

#ifdef CLOCK_1MHZ
      uint16_t x = 1E6 / (frekvence << 4);   // vypocet pro CLK 1 MHz
#endif

#ifdef CLOCK_2MHZ
      uint32_t x = 2E6 / (frekvence << 4);  // vypocet pro CLK 2 MHz
#endif

      if(x >= 256)    // pokud je hodnota po deleni vetsi nez 255, tak se provede rozdeleni do dvou registru
      {
        coarse_hodnota = x >> 8;  // vznik hodnoty pro registr R1, R3 a R5
        fine_hodnota = x & 0x0FF; // vznik hodnoty pro registr R0, R2 a R4
      }
      else
      {
        fine_hodnota = x & 0x0FF; // když hodnota po deleni neni vetsi nez 255, tak se pouzije jen jeden registr
      }

```

```
32        switch(channel)             // vyber kanalu na kterem bude zadana frekvence
33        {
34          case CHANNEL_TONE_NONE:
35            break;
36
37          case CHANNEL_TONE_A:
38          {
39            zapis_do_registru(CHANNEL_A_TONE_COARSE, coarse_hodnota);
40            zapis_do_registru(CHANNEL_A_TONE_FINE, fine_hodnota);
41          }
42          break;
43
44          case CHANNEL_TONE_B:
45          {
46            zapis_do_registru(CHANNEL_B_TONE_COARSE, coarse_hodnota);
47            zapis_do_registru(CHANNEL_B_TONE_FINE, fine_hodnota);
48          }
49          break;
50
51          case CHANNEL_TONE_C:
52          {
53            zapis_do_registru(CHANNEL_C_TONE_COARSE, coarse_hodnota);
54            zapis_do_registru(CHANNEL_C_TONE_FINE, fine_hodnota);
55          }
56          break;
57
58          case CHANNEL_TONE_AB:
59          {
60            zapis_do_registru(CHANNEL_A_TONE_COARSE, coarse_hodnota);
61            zapis_do_registru(CHANNEL_A_TONE_FINE, fine_hodnota);
62            zapis_do_registru(CHANNEL_B_TONE_COARSE, coarse_hodnota);
63            zapis_do_registru(CHANNEL_B_TONE_FINE, fine_hodnota);
64          }
65          break;
66
67          case CHANNEL_TONE_AC:
68          {
69            zapis_do_registru(CHANNEL_A_TONE_COARSE, coarse_hodnota);
70            zapis_do_registru(CHANNEL_A_TONE_FINE, fine_hodnota);
71            zapis_do_registru(CHANNEL_C_TONE_COARSE, coarse_hodnota);
72            zapis_do_registru(CHANNEL_C_TONE_FINE, fine_hodnota);
73          }
74          break;
75
76          case CHANNEL_TONE_BC:
77          {
78            zapis_do_registru(CHANNEL_B_TONE_COARSE, coarse_hodnota);
79            zapis_do_registru(CHANNEL_B_TONE_FINE, fine_hodnota);
80            zapis_do_registru(CHANNEL_C_TONE_COARSE, coarse_hodnota);
81            zapis_do_registru(CHANNEL_C_TONE_FINE, fine_hodnota);
82          }
83          break;
84
85          case CHANNEL_TONE_ABC:
86          {
87            zapis_do_registru(CHANNEL_A_TONE_COARSE, coarse_hodnota);
88            zapis_do_registru(CHANNEL_A_TONE_FINE, fine_hodnota);
89            zapis_do_registru(CHANNEL_B_TONE_COARSE, coarse_hodnota);
90            zapis_do_registru(CHANNEL_B_TONE_FINE, fine_hodnota);
91            zapis_do_registru(CHANNEL_C_TONE_COARSE, coarse_hodnota);
92            zapis_do_registru(CHANNEL_C_TONE_FINE, fine_hodnota);
93          }
94          break;
95        }
96      }
97  }
```

B.1.11 Funkce nastaveni_amplitudy_kanalu

```
1  void nastaveni_amplitudy_kanalu(kanaly_ton channel, uint8_t amplituda, envelope povolit_envelope)
2  {
```

```
3   if(amplituda >= 16)     // ochrana proti zadani vetsi hodnoty amplitudy
4   {
5     return;
6   }
7
8   if(povolit_envelope)    // kdyz chceme povolit obalkovy generator
9   {
10    amplituda = 1 << 4;   // posun pro aktivaci generatoru
11  }
12
13  switch(channel)         // vyber kanalu
14  {
15    case CHANNEL_TONE_NONE:
16    {
17      zapis_do_registru(CHANNEL_A_AMPLITUDE, 0);
18      zapis_do_registru(CHANNEL_B_AMPLITUDE, 0);
19      zapis_do_registru(CHANNEL_C_AMPLITUDE, 0);
20    }
21    break;
22
23    case CHANNEL_TONE_A:
24    {
25      zapis_do_registru(CHANNEL_A_AMPLITUDE, amplituda);
26    }
27    break;
28
29    case CHANNEL_TONE_B:
30    {
31      zapis_do_registru(CHANNEL_B_AMPLITUDE, amplituda);
32    }
33    break;
34
35    case CHANNEL_TONE_C:
36    {
37      zapis_do_registru(CHANNEL_C_AMPLITUDE, amplituda);
38    }
39    break;
40
41    case CHANNEL_TONE_AB:
42    {
43      zapis_do_registru(CHANNEL_A_AMPLITUDE, amplituda);
44      zapis_do_registru(CHANNEL_B_AMPLITUDE, amplituda);
45    }
46    break;
47
48    case CHANNEL_TONE_AC:
49    {
50      zapis_do_registru(CHANNEL_A_AMPLITUDE, amplituda);
51      zapis_do_registru(CHANNEL_C_AMPLITUDE, amplituda);
52    }
53    break;
54
55    case CHANNEL_TONE_BC:
56    {
57      zapis_do_registru(CHANNEL_B_AMPLITUDE, amplituda);
58      zapis_do_registru(CHANNEL_C_AMPLITUDE, amplituda);
59    }
60    break;
61
62    case CHANNEL_TONE_ABC:
63    {
64      zapis_do_registru(CHANNEL_A_AMPLITUDE, amplituda);
65      zapis_do_registru(CHANNEL_B_AMPLITUDE, amplituda);
66      zapis_do_registru(CHANNEL_C_AMPLITUDE, amplituda);
67    }
68    break;
69  }
70 }
```

B.1.12 Funkce nastaveni_frekvence_noise_generatoru

```
void nastaveni_frekvence_noise_generatoru(uint32_t frekvence)
{
#ifdef CLOCK_1MHZ
  if((frekvence >= 2016) && (frekvence <= 62500))   // limitni frekvence ktere lze generovat pri 1 MHz CLK
  {
    uint8_t x = 1E6 / (frekvence << 4);             // vypocet pro CLK 1 MHz
    zapis_do_registru(NOISE_PERIOD, x);             // zapis frekvence do registru
  }

#endif

#ifdef CLOCK_2MHZ

  if((frekvence >= 4032) && (frekvence <= 125000))  // limitni frekvence ktere lze generovat pri 2 MHz CLK
  {
    uint8_t x = 2E6 / (frekvence << 4);             // vypocet pro CLK 2 MHz
    zapis_do_registru(NOISE_PERIOD, x);             // zapis frekvence do registru
  }
#endif
}
```

B.1.13 Funkce nastaveni_frekvence_tvaru_obalky

```
void nastaveni_frekvence_tvaru_obalky(float frekvence, obalky tvar_obalky)
{

#ifdef CLOCK_1MHZ
  if((frekvence >= 0.06) && (frekvence <= 3905))   // limitni frekvence ktere lze generovat pri 1 MHz CLK
#endif

#ifdef CLOCK_2MHZ
    if((frekvence >= 0.13) && (frekvence <= 7812))  // limitni frekvence ktere lze generovat pri 2 MHz CLK
#endif
    {
      uint16_t coarse_hodnota = 0;                // promenna pro ulozeni hodnoty registru R12
      uint16_t fine_hodnota = 0;                  // promenna pro ulozeni hodnoty registru R11
      uint16_t x = 0;

#ifdef CLOCK_1MHZ
      x = 1E6 / (frekvence * 256);              // vypocet pro CLK 1 MHz
#endif

#ifdef CLOCK_2MHZ
      x = 2E6 / (frekvence * 256);              // vypocet pro CLK 2 MHz
#endif

      if(x >= 256)  // pokud je hodnota po deleni vetsi nez 255, tak se provede rozdeleni do dvou registru
      {
        coarse_hodnota = x >> 8;            // vznik hodnoty pro registr R12
        fine_hodnota = x & 0x00FF;          // vznik hodnoty pro registr R11
      }
      else
      {
        fine_hodnota = x & 0x00FF;        // když hodnota po deleni neni vetsi nez 255, tak se pouzije jen jeden registr
      }

      zapis_do_registru(ENVELOPE_PERIOD_COARSE, coarse_hodnota);  // zapis do registru R12
      zapis_do_registru(ENVELOPE_PERIOD_FINE, fine_hodnota);      // zapis do registru R11

      switch(tvar_obalky)           // vyber tvaru obalky
      {
        case rampa_z_1_do_0:
          zapis_do_registru(ENVELOPE_SHAPE_CYCLE, 0);
          break;

        case trojuhelnikovy_pulz_z_0_do_0:
          zapis_do_registru(ENVELOPE_SHAPE_CYCLE, 0x4);
          break;

        case trojuhelnik_z_1_do_0:
```

```
48            zapis_do_registru(ENVELOPE_SHAPE_CYCLE, 0x8);
49            break;
50
51          case pila_z_1:
52            zapis_do_registru(ENVELOPE_SHAPE_CYCLE, 0xA);
53            break;
54
55          case trojuhelnikovy_pulz_z_1_do_1:
56            zapis_do_registru(ENVELOPE_SHAPE_CYCLE, 0xB);
57            break;
58
59          case trojuhelnik_z_0_do_1:
60            zapis_do_registru(ENVELOPE_SHAPE_CYCLE, 0xC);
61            break;
62
63          case rampa_z_0_do_1:
64            zapis_do_registru(ENVELOPE_SHAPE_CYCLE, 0xD);
65            break;
66
67          case pila_z_0:
68            zapis_do_registru(ENVELOPE_SHAPE_CYCLE, 0xE);
69            break;
70        }
71      }
72  }
```

B.1.14 Funkce delay_ms

```
1  void delay_ms(uint32_t cas)
2  {
3    _ticks = 0;               // nastaveni na nulu po kazdem pruchodu funkci
4    while(_ticks < cas)       // kdyz je zadany pocet _ticks mensi nez zadany cas, tak funkce ceka
5      ;
6  }
```

B.1.15 Funkce SysTick_Handler

```
1  void SysTick_Handler(void)
2  {
3    _ticks++;                   // inkrementace promenne po preteceni casovace
4  }
```

B.2 Soubor funkce.h

```
1  #ifndef FUNKCE_H_
2  #define FUNKCE_H_
3
4  #include <nucleo_usart.h>                                  // nastaveni USARTu pro odesilani dat na PC
5
6  // definice konstant pro rizeni sbernice a resetu
7  #define BC1_SET 1 << 8
8  #define BC1_RESET 1 << 24
9
10 #define BDIR_SET 1 << 9
11 #define BDIR_RESET 1 << 25
12
13 #define DA0_DA7_RESET 0x00FF0000
14 #define DA0_DA7_SET 0xFF
15
16 #define PC_13_SET 1 << 13
17 #define PC_13_RESET 1 << 29
18
19 // vyctove typy enum pro ovladani kanalu, registru a obalek
20 typedef enum
21 {
22   CHANNEL_A_TONE_FINE,
23   CHANNEL_A_TONE_COARSE,
```

```
24
25    CHANNEL_B_TONE_FINE,
26    CHANNEL_B_TONE_COARSE,
27
28    CHANNEL_C_TONE_FINE,
29    CHANNEL_C_TONE_COARSE,
30
31    NOISE_PERIOD,
32
33    ENABLE_NOISE_TONE,
34
35    CHANNEL_A_AMPLITUDE,
36    CHANNEL_B_AMPLITUDE,
37    CHANNEL_C_AMPLITUDE,
38
39    ENVELOPE_PERIOD_FINE,
40    ENVELOPE_PERIOD_COARSE,
41    ENVELOPE_SHAPE_CYCLE,
42  } registry;
43
44  typedef enum
45  {
46    CHANNEL_TONE_NONE,
47    CHANNEL_TONE_A,
48    CHANNEL_TONE_B,
49    CHANNEL_TONE_C,
50    CHANNEL_TONE_AB,
51    CHANNEL_TONE_AC,
52    CHANNEL_TONE_BC,
53    CHANNEL_TONE_ABC,
54  } kanaly_ton;
55
56  typedef enum
57  {
58    CHANNEL_NOISE_NONE,
59    CHANNEL_NOISE_A,
60    CHANNEL_NOISE_B,
61    CHANNEL_NOISE_C,
62    CHANNEL_NOISE_AB,
63    CHANNEL_NOISE_AC,
64    CHANNEL_NOISE_BC,
65    CHANNEL_NOISE_ABC,
66  } kanaly_noise;
67
68  typedef enum
69  {
70    NOISE_OFF,
71    NOISE_ON,
72  } noise;
73
74  typedef enum
75  {
76    ENVELOPE_OFF,
77    ENVELOPE_ON,
78  } envelope;
79
80  typedef enum
81  {
82    rampa_z_1_do_0 = 1,
83    trojuhelnikovy_pulz_z_0_do_0,
84    trojuhelnik_z_1_do_0,
85    pila_z_1,
86    trojuhelnikovy_pulz_z_1_do_1,
87    trojuhelnik_z_0_do_1,
88    rampa_z_0_do_1,
89    pila_z_0,
90  } obalky;
91
92  void bus_init(void);                          // nastaveni sbernice - DA0-DA7, BDIR, BC2
93  void reset_init(void);                        // nastaveni reset tlacitka na pinu PC_13
94  bool read_reset_btn(void);                    // cteni hodnoty reset tlacika
95  void clock_init(void);                        // nastaveni citace pro generovani 1 MHz hodinoveho signalu
96  void bus_inactive(void);                      // nastaveni sbernice do neaktivniho stavu
```

```
97  void bus_write(void);                          // priprav sbernici pro zapis hodnoty do registru
98  void bus_latch_adress(void);                   // priprav sbernici pro vyber registru
99  void bus_reset(void);                          // nastavi DA0 -DA7 na nulu
100 void zapis_do_registru(uint8_t registr, uint8_t hodnota); // vyber a zapis hodnoty do registru
101
102 // vyber kanalu na kterem se bude prehravat a povoleni generatoru sumu
103 void vyber_kanalu_tonu_noise(kanaly_ton channel, noise povolit_noise);
104
105 // nastaveni frekvence tonu na vybranem kanalu
106 void nastaveni_frekvence_tonu(kanaly_ton channel, uint32_t frekvence);
107
108 // nastaveni urovne D/A prevodniku na danem kanalu a povoleni obalkoveho generatoru
109 void nastaveni_amplitudy_kanalu(kanaly_ton channel, uint8_t amplituda, envelope povolit_envelope);
110 void nastaveni_frekvence_noise_generatoru(uint32_t frekvence);              // nastaveni frekvence generatoru sumu
111 void nastaveni_frekvence_tvaru_obalky(float frekvence, obalky tvar_obalky); // nastaveni frekvence a tvaru obalky
112 void delay_ms(uint32_t cas);                  // generovani casoveho zpozdeni
113 void SysTick_Handler(void);                   // funkce pro obsluhu preruseni od casovace SysTick
114
115 #endif /* FUNKCE_H_ */
116
```

B.3 Soubor main.c

```
1  #include <funkce.h>
2
3  // nastaveni rezimu ovladani AY-3-8912
4  #define mod_reset   // tlacitko na pinu PC13 plni funkci RESETu pro AY-3-8912, nepouzivat pro mod_skladba
5  #define mod_skladba // tlacitko na pinu PC13 plni funkci prepinani skladeb, nepouzivat pri mod_reset
6  #define ovladani_vystupu  // ovladani jednotlivych funkci AY-3-8912 jako je nastaveni obalky, frekvence, kanalu
7                            // nepouzivat s mod_skladba, pouzivat mod_reset
8
9  // pristup ke zvukovym souboru pri pouziti mod_skladba
10 #ifdef mod_skladba
11 #include <foty2.h>
12 #include <terminator.h>
13 #include <batman.h>
14 #include <garylineker.h>
15 #include <f15_strike_eagle.h>
16 #endif
17
18 int main(void)
19 {
20   bool min_stav_reset_btn =  true;
21
22 // promenne pro mod_reset
23 #ifdef mod_reset
24   bool zmacknuto = false;
25 #endif
26
27 // promenne pro mod_skladba
28 #ifdef mod_skladba
29   uint8_t cislo_registru = 0;
30   uint8_t hodnota_do_registru = 0;
31   uint8_t cislo_skladby = 0;
32   uint32_t posuv_reg = 5;
33   uint32_t posuv_hodnota = 6;
34   uint8_t *pole = NULL;
35 #endif
36
37   Usart2Init(9600);   // nastaveni rychlosti USARTu na 9600 Baud
38   puts("\n--------------------------\n");
39   puts("Start aplikace\n");
40
41   SystemCoreClockUpdate();  // aktualizuje globalni promennou SystemCoreClock
42   SysTick_Config(SystemCoreClock / 1000); // citac SysTick pretece za 1ms, frekvence citace je 1 kHz
43
44 // inicializace hodin, sbernice a pocatecni RESET AY-3-8912
45   clock_init();
46   bus_init();
47   bus_inactive();
```

```
48    bus_reset();
49    reset_init();
50
51  // funkce pro ovladani mod_skladba
52  #ifdef   ovladani_vystupu
53    vyber_kanalu_tonu_noise(CHANNEL_TONE_B, NOISE_OFF); // kanal B, sum vypnuty
54    nastaveni_frekvence_tonu(CHANNEL_TONE_B, 1000);     // kanal B, frekvence 1000 Hz
55    nastaveni_amplitudy_kanalu(CHANNEL_TONE_B, 15, ENVELOPE_OFF); // kanal B, amplituda 15, obalka je vypnuta
56    nastaveni_frekvence_tvaru_obalky(0.15, pila_z_1);   // frekvence obalky a jeji tvar
57    nastaveni_frekvence_noise_generatoru(3000);       // frekvence sumoveho generatoru je 3000 Hz
58  #endif
59
60  //  vypnuti kanalu pro mod_skladba, kdyz nebude zvolena skladba tak vystup nebude aktivni
61  #ifdef mod_skladba
62    vyber_kanalu_tonu_noise(CHANNEL_TONE_NONE, CHANNEL_TONE_NONE);  // pocatecni nastaveni
63  #endif
64
65    while (1)
66    {
67  #ifdef mod_reset
68      // cteni stavu tlacitka RESET, kdyz bylo stisknuto tak vysledek bude false
69      bool soucasny_stav_reset_btn = read_reset_btn();
70
71      if(min_stav_reset_btn != soucasny_stav_reset_btn) // porovnani soucasneho a predchoziho stavu
72      {
73        min_stav_reset_btn = soucasny_stav_reset_btn;   // ulozeni soucasneho stavu
74
75        if(!(soucasny_stav_reset_btn && zmacknuto))   // detekce stisku tlacitka, podminka funguje pro prvni pruchod funkci
76        {
77          zmacknuto = true;             // stisk byl detekovan
78          bus_inactive();                // nastaveni sbernice do neaktivniho stavu
79          bus_reset();                  // nastavi DA0 -DA7 na nulu
80          GPIOC->BSRR |= PC_13_RESET;   // nastaveni reset casti do 1 -> PC13 = 0 -> reset AY-8912 (je aktivni na log. 0)
81        }
82
83        if(soucasny_stav_reset_btn && zmacknuto) // podminka funguje pro druhy pruchod funkci (po stisku tlacitka)
84        {
85          zmacknuto = false;            // nastaveni na false pro dalsi detekci stisku
86          GPIOC->BSRR |= PC_13_SET;     // nastaveni set casti do 1 -> PC13 = 1 (set ma vyssi prioritu ne reset)
87        }
88      }
89  #endif
90
91  #ifdef mod_skladba
92      // cteni stavu tlacitka pro vyber skladby, kdyz bylo stisknuto tak vysledek bude false
93      bool soucasny_stav_reset_btn = read_reset_btn();
94
95      if(min_stav_reset_btn != soucasny_stav_reset_btn) // porovnani soucasneho a predchoziho stavu
96      {
97        min_stav_reset_btn = soucasny_stav_reset_btn; // ulozeni soucasneho stavu
98
99        if(soucasny_stav_reset_btn) // detekce stisku tlacitka, podminka funguje pri druhem pruchodu
100       {
101         cislo_skladby++;        // zvoleni skladby
102
103         // pro nastaveni pocatecniho stavu na nulu nebo kdyz chceme behem aktualniho prehravani skladby prehrat jinou
104         cislo_registru = 0;
105         hodnota_do_registru = 0;
106         posuv_reg = 5;
107         posuv_hodnota = 6;
108
109         if(cislo_skladby == 6)  //  sesty stisk tlacitka znamena konec prehravani
110         {
111           cislo_skladby = 0;
112           vyber_kanalu_tonu_noise(CHANNEL_TONE_NONE, CHANNEL_TONE_NONE);
113           pole = NULL;
114         }
115       }
116     }
117
118     if (cislo_skladby > 0)  // probehne kdyz je zvolena skladba
119     {
120       switch(cislo_skladby) // vyber skladby podle poctu stisku tlacitka
```

```
121       {
122         case 1:
123           pole = FOTY2_music_array;  // prirazeni adresy hudebniho souboru do promenne pole
124           break;
125         case 2:
126           pole = F15_STRIKE_EAGLE_music_array;
127           break;
128         case 3:
129           pole = GARYLINEKER_music_array;
130           break;
131         case 4:
132           pole = TERMINATOR_music_array;
133           break;
134         case 5:
135           pole = BATMAN_music_array;
136           break;
137       }
138     }
139
140     if(pole != NULL)  // probehne kdyz adresa pole nebude NULL (neni zvolena skladba)
141     {
142       cislo_registru = pole[posuv_reg]; // prochazeni hudebniho souboru, vyber registru
143       hodnota_do_registru = pole[posuv_hodnota];  // prochazeni hudebniho souboru, vyber hodnoty
144       posuv_reg += 2;     // pricteni pro skok na dalsi adresu registru
145       posuv_hodnota += 2; // pricteni pro skok na dalsi hodnotu registru
146
147       if(cislo_registru < 16) // detekce zapisovych dat
148       {
149         zapis_do_registru(cislo_registru, hodnota_do_registru); // zapis hodnot ze souboru na cip
150       }
151
152       if(cislo_registru == 16)  // detekce zpozdeni
153       {
154         delay_ms(20 * (hodnota_do_registru + 1)); // zpozdeni pred zpracovanim dalsiho datoveho bloku
155       }
156
157       // detekce konce souboru a nastaveni pocatecnich hodnot, aby mohla zvolena skladba hrat od zacatku
158       if(cislo_registru == 0xff)
159       {
160         cislo_registru = 0;
161         hodnota_do_registru = 0;
162         posuv_reg = 5;
163         posuv_hodnota = 6;
164       }
165     }
166 #endif
167   }
168 }
169
```