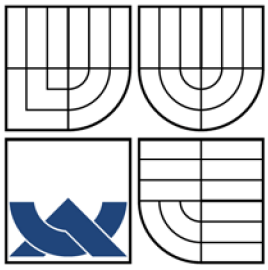
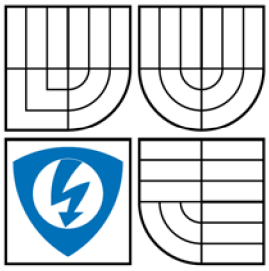


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV RADIOELEKTRONIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF RADIO ELECTRONICS

ZPRACOVÁNÍ SIGNÁLU PROSTŘEDNÍCTVÍM ZVUKOVÉ KARTY S VYUŽITÍM SYSTÉMU ASIO

SIGNAL PROCESSING USING SOUND CARD WITH ASIO SYSTEM

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR
ŠTĚPÁN POLANSKÝ

VEDOUCÍ PRÁCE
SUPERVISOR
JAROSLAV RUMÁNEK

BRNO 2009

ABSTRAKT

Práce je seznámením s profesionální systémem ASIO, který je určen pro zpracování zvukových signálů. Je zde vypracován přehled jeho funkcí a způsob použití. Dále je vyložena důležitá teorie týkající se zpracování signálu pomocí osobního počítače, jsou objasněny důležité pojmy a naznačeny metody použití zvukové karty pro zpracování signálů v reálném čase. Práce dále obsahuje postup při realizaci aplikačního prostředí pro zpracování zvuku v jazyce c++, základní metodiku programování s ovladačem ASIO a rozbor zvukových efektů.

KLÍČOVÁ SLOVA

ASIO, zpracování signálu, zvukové efekty, filtrace, spektrální analýza, c++.

ABSTRACT

This work is introduction of professional system ASIO which serves signal processing. It contains list of functions and applications. It clears up important theory of signal processing by personal comuputer and considerable conceptions. Work is engaged in possibilities of real time signal processing. Work contains procedure of realization the application interface for sound processing via language c++, basic methods for programing with ASIO driver and analyse of sound effects.

KEYWORDS

ASIO, signal processing, sound effects, filtering, spectral analyse, c++.

POLANSKÝ, Š. *Zpracování signálu prostřednictvím zvukové karty s využitím systému ASIO: bakalářská práce*. Brno: FEKT VUT v Brně 2009, 55 s., 3 příl.

Prohlášení

Prohlašuji, že svou bakalářskou práci na téma Zpracování signálu prostřednictvím zvukové karty s využitím systému ASIO jsem 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 bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne 5. června 2009 .....
podpis autora

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Jaroslavu Rumánkovi za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

V Brně dne 5. června 2009 .....
podpis autora

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Zvuková karta je základním kamenem multimediální výbavy počítače sloužící pro vstup a výstup zvukového signálu. Její hlavní funkcí je diskrétní zpracování signálu pomocí příslušného programu.

Na začátku byly všechny zvukové karty pod standardem Windows a i audio aplikace tento standard přebíraly. Vše tedy běželo pod ovladači MME nebo DirectX. Zpočátku to příliš nevadilo – zvukové karty byly maximálně stereofonní (2 audio kanály) a neměly integrovány digitální signálové procesory. S rostoucími požadavky však přibývaly kanály a podstatně se zvětšily možnosti celkové práce s mícháním a úpravou zvuku již v samotné kartě. Zde již standardní ovladače nestačily a nastal čas pro ASIO (Steinberg) a WDM (Microsoft).

Cílem této práce je seznámení s profesionálním ovladačem ASIO a jeho následné využití při realizaci aplikačního prostředí v jazyce c++. V dokumentu jsou srovnány jednotlivé přístupy k ovladači a na základě těchto informací lze založit výběr optimálního postupu při programování audio aplikací.

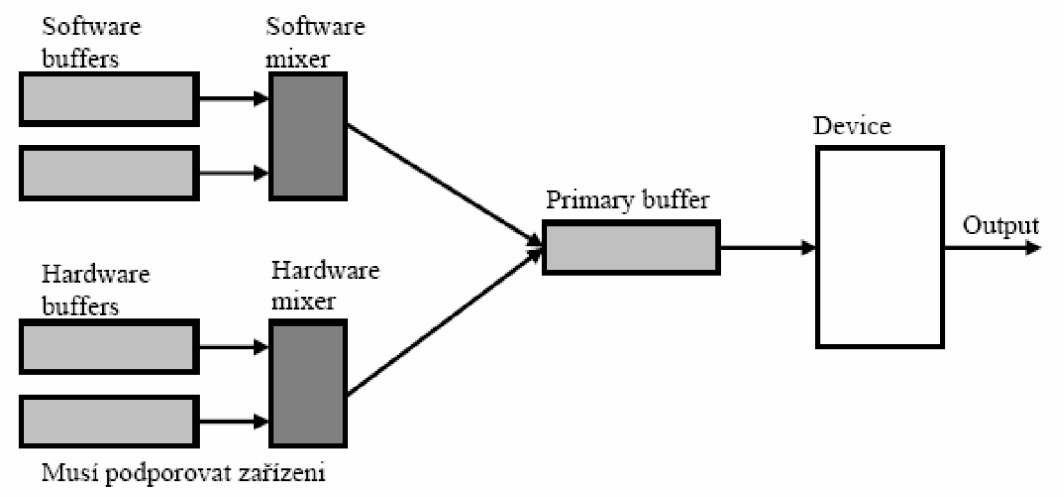
2 Nezbytný teoretický základ

2.1 Důležité pojmy

API je zkratka Application Programming Interface, což znamená rozhraní pro programování aplikací. Jde o sbírku procedur, funkcí či tříd nějaké knihovny, které může programátor využívat.

DirectX je programátorská knihovna obsahující nástroje pro tvorbu počítačových her a dalších multimediálních aplikací, vytvořená firmou Microsoft a vydaná v první verzi v roce 1995.

DirectSound (dříve též spolu s DirectMusic označováno souhrnným názvem DirectX Audio) slouží k podpoře přehrávání a záznamu zvuků. Má schopnost přistupovat k zařízení rychleji než MME. DirectSound zkouší jak moc dané zařízení podporuje hardwarovou akceleraci (přehrává několik bufferů najednou). Pokud akcelerace není podporována, je emulována softwarově. Funkce DirectSound je na obrázku xxx.



Obr. 2.1 Princip funkce DirectSound

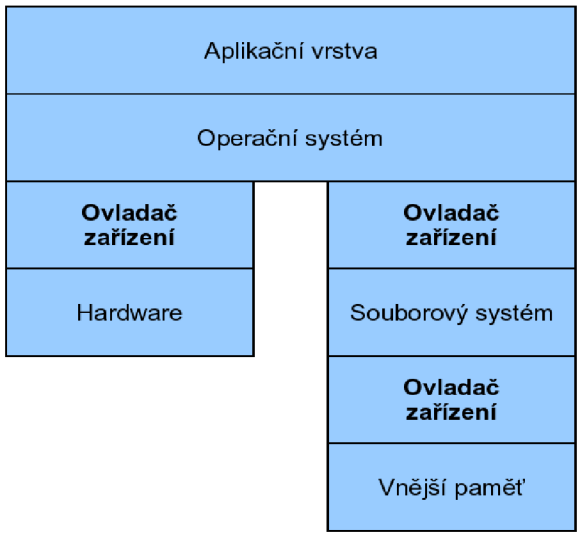
DirectMusic je podporou pro přehrávání a zpracování hudby

MME (MultiMedia Extensions) jsou ovladače představené v na podzim roku 1991 se systémem Windows 3.1. Byly vytvořeny pro podporu zvukových karet a CD-ROM zařízení. Jde o API jež je pod přímou kontrolou systému.

Ovladač (device driver) je software, jenž umožňuje operačnímu systému pracovat s hardwarem. Některé ovladače jsou součástí operačního systému, jiné jsou distribuovány s hardwarem (např. na CD-ROM).

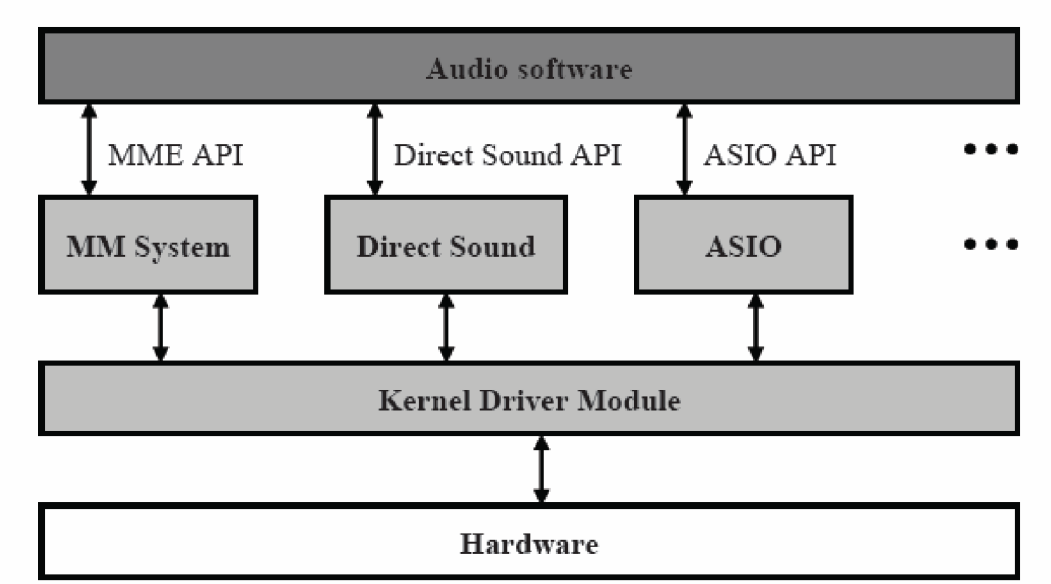
Ovladač zajišťuje řízení hardwaru a zároveň komunikuje se zbytkem operačního systému pomocí obecnějších rozhraní, která zajišťují abstrakci zařízení. Základní vlastností abstrakce je použití stejného nebo podobného rozhraní pro podobná zařízení: třeba abstrakce blokového zařízení umožňuje pracovat stejně s diskem, disketou a CD/DVD mechanikou. CD/DVD mechanika má kromě rozhraní blokového zařízení druhé rozhraní umožňující vypalování, ale program který z ní chce jenom číst soubory o tomto druhém rozhraní nepotřebuje vědět.

Zpravidla bývá rozhraní snazší k používání než přímý přístup na zařízení – například umožňuje spooling (technika sdílení vstupně výstupních zařízení multitaskingovém operačním systému) a bufferování i u zařízení, které ho nepodporují samy. Další informace a vysvětlení pojmů lze nalézt v článku [4]. Na obrázku 2-2 je umístění vrstvy ovladače v architektuře operačního systému.



Obr. 2.2 Ukázka umístění vrstvy ovladače zařízení v architektuře operačního systému

Základní struktura přístupu ke zvukové kartě je zobrazena na obrázku 2-3. Ovladač je zobrazen světle šedou barvou.



Obr. 2.3 Struktura přístupu aplikace k hardwaru

ReWire je rozhraní, sloužící k propojení profesionálních programů pro tvorbu hudby (např. Reason, Cubase, Ableton, Fruity Loops,…)

SDK (Software Development Kit). Balíček vývojových nástrojů, které umožňují vývoj aplikace pro specifickou platformu, počítačový systém, nebo operační systém.

VCL (Visual Component Library) je vizuální na komponentách založený framework určený pro vývoj aplikací pod Microsoft Windows. VCL byl vyvinut firmou Borland v jazyku Object Pascal a je určen pro vývojová prostředí Delphi a C++Builder. VCL využívá objektově orientovaný přístup. Vytváří objektovou hierarchii, kde všechny objekty dědí přímo nebo nepřímo od třídy TObject. Komponenty VCL zapouzdřují práci s ovládacími prvky Windows. Tento přístup umožňuje programátorům dále rozšiřovat VCL o řadu dalších vizuálních i nevizuálních komponent.

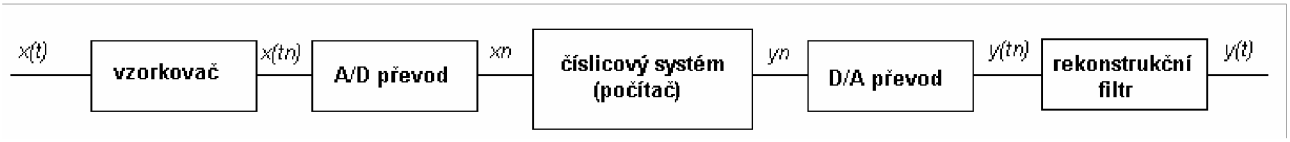
VST (Virtual Studio Technology) je softwarové rozhraní (definované API) vyvinuté firmou Steinberg a zajišťuje komunikaci dll knihoven (plug-inů) pracujících se zvukem a nějakého hostitelského prostředí. Rozhraní je definované tak, aby umožňovalo do tohoto kódu posílat v datovém proudu zvuk a na druhé straně ho zase číst nebo poslat do dalšího plug-inu. Plug-iny je možné takto za sebe řetězit stejně, jako analogové efekty ve studiu. Hlavní pointa tohoto rozhraní je v tvorbě efektů a nástrojů naprosto nezávisle na hostitelském programu. Během let se z něj díky rozšíření produktů firmy Steinberg stal celosvětový standard. Potěšujícím faktem je, že vývojové prostředky dává firma volně k dispozici a kdokoliv si tak může naprogramovat jakýkoliv nástroj nebo efekt, který bude fungovat v naprosté většině dnešních komerčních hudebních softwarů.

WDM (Windows Driver Model) je sestava ovladačů představených s operačním systémem Windows 98.

2.2 Diskrétní zpracování zvuku

Diskrétním zpracováním je myšlen přepočet vstupní posloupnosti hodnot na posloupnost hodnot výstupních. Soustavy, které takový výpočet realizují, označujeme jako diskrétní systémy, jež jsou matematicky popsány diferenciálními rovnicemi a jejich analýzou lze obdržet informace o vlastnostech soustavy vzhledem ke zpracování signálu. Pokud zpracováváme spojitý signál *x(t)* se záměrem získat opět spojitý výstupní signál *y(t)*, je prvým článkem zpracování *vzorkovač,* který v zadaných časových okamžicích t_n získává vzorky vstupu x_n, tvořící vstupní posloupnost diskrétního systému. Na základě disretní posloupnosti *y(n)*, kterou poskytuje diskrétní systém, je pak nutno vytvořit spojitou funkci *y(t)* pomocí *rekonstrukčního interpolátoru*.

Vlastní číslicový systém je v našem případě počítač, který, pokud jde o zpracování časových posloupností, musí pracovat v režimu reálného času a mít takovou výkonnost, aby v období mezi dvěma vzorky zvládl celý algoritmus výpočtu jednoho výstupního vzorku. Celý postup zpracování signálu pomocí digitálního systému je uveden na obrázku 2-4.



Obr. 2.4 Postup digitálního zpracování analogového signálu

2.2.1 Vzorkování a rekonstrukce

Zvukový signál je signálem spojitým a jednorozměrným – jde o funkci jedné spojité reálné proměnné *t (čas)*. **Vzorkování** slouží k vyjádření spojitého signálu *f(t)* jeho diskrétními vzorky $f_n = f(t_n)$ pro určité hodnoty t_n nezávisle proměnné *t*. Zpravidla jsou tyto vzorky ekvidistantní, tj. $t_n = n.T$, kde *T* je vhodná reálná konstanta, označována jako *perioda vzorkování*, takže

$$f_n = f(nT). \tag{2.1}$$

Spektrum ideálně a ekvidistantně vzorkovaného signálu je tvořeno součtem nekonečného počtu replik spektra původního analogového signálu, které jsou vzájemně posunuty o celistvé násobky úhlového vzorkovacího kmitočtu. Aby nedošlo ke ztrátě informace, musí každá replika nést úplnou informaci o původním signálu, což je ovšem možné jen nedojde-li k překrývání a tím k narušení dílčích spekter.
Odtud plynou podmínky rekonstruovatelnosti spojitého signálu ze vzorků [1]:

analogový signál musí mít omezené spektrum, tj.

$$F(\omega) = 0 \text{ vně intervalu } <- \omega_{max}, \omega_{max} > \tag{2.2}$$

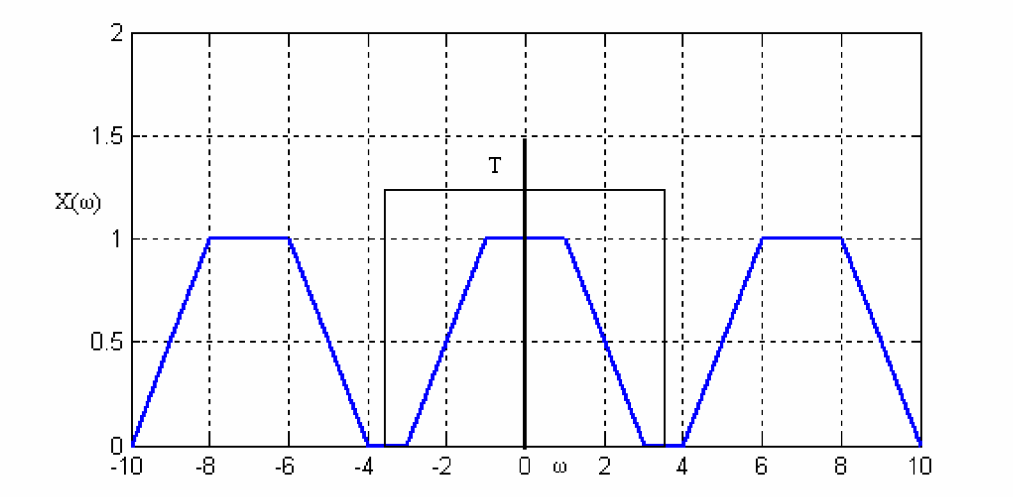
vzorkovací kmitočet musí splňovat vztah

$$\omega_{vz} > 2\omega_{max} \tag{2.3}$$

Prvá podmínka závisí na charakteru zpracovávaného signálu. Pokud signál vzniká v setrvačné fyzikální soustavě jako některá z fyzikálních veličin (např. elektrické napětí), nebo se takovou soustavou šíří, lze vždy očekávat, že složky vysokých kmitočtů jsou počínaje od určité meze natolik malé, že prvou podmínku znatelně nenarušují. Není-li možno takový předpoklad učinit, nebo je sice signál kmitočtově omezen, ale jeho kmitočet ω_{max} je natolik vysoký, že technické předpoklady neumožňují splnit podmínku, je nutno spektrum analogového signálu před vzorkováním omezit dolnofrekvenční propustí (tzv. antialiazingovým filtrem). To se ovšem týká i případu, kdy se volí nižší vzorkovací kmitočet vzhledem k tomu, že z hlediska zpracování signálu jsou jeho složky s vyššími frekvencemi nezajímavé.

Druhou podmínkou je tzv. *vzorkovací teorém* (Nyquistův, Kotělníkův nebo Shanonův), který určuje, jak vysokého vzorkovacího kmitočtu je pro daný typ signálu třeba použít. V praxi je, s ohledem na nedokonalé možnosti rekonstrukce signálu ze vzorků, zpravidla třeba splnit podmínku se značnou rezervou.

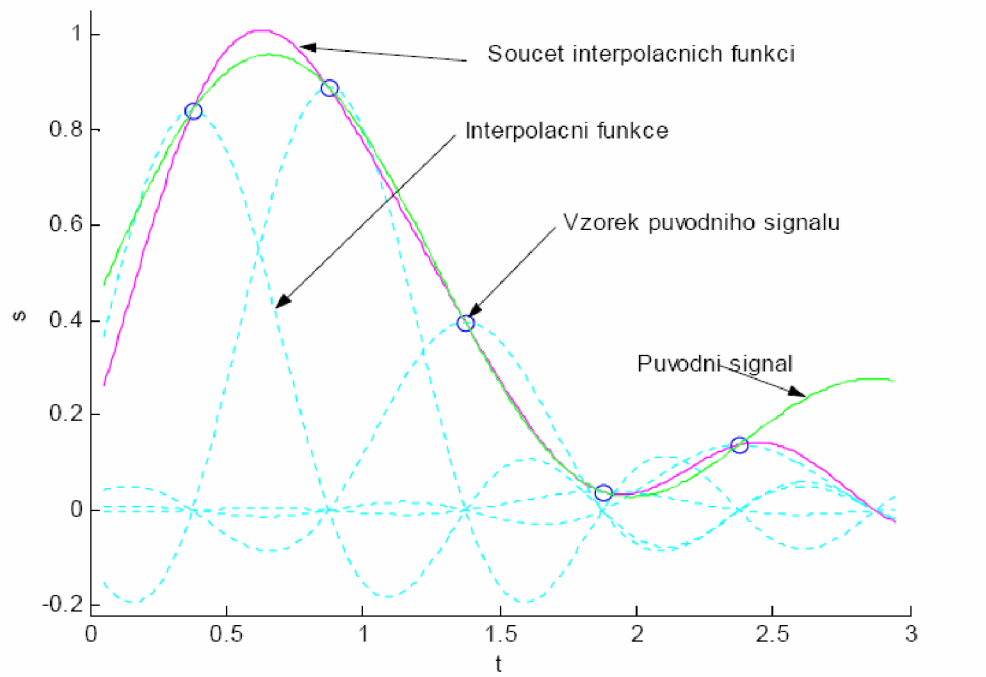
Rekonstrukce je proces inverzní ke vzorkování, jde tedy o vytvoření spojitého signálu *f(t)* na základě jeho vzorkované verze $f_v(t)$. Nejsnáze lze tento proces formulovat ve frekvenční oblasti: podaří-li se změnit spektrum vzorkovaného signálu $F_v(\omega)$ na spektrum původního signálu $F(\omega)$, bude vytvořen žádoucí signál v časové oblasti. Rekonstrukce interpretovaná ve frekvenční oblasti je na obrázku 2.5.



Obr. 2.5 Rekonstrukce interpretovaná ve frekvenční oblasti

Stačí potlačit zcela všechny repliky spektra mimo základní v kmitočtové pásmo $<-\omega_{vz}/2,\ \omega_{vz}/2>$, což lze, za předpokladu splnění podmínek vzorkovacího teorému realizovat průchodem signálu ideální dolní propustí s mezním kmitočtem $\omega_{vz}/2$. Vzhledem k tomu, že rekonstrukční filtr nemůže být ani teoreticky ideální, je třeba počítat s určitými „ochrannými“ pásmy zahrnujícími přechodné oblasti charakteristiky filtru mezi propustným a nepropustným pásmem.

V originální (časové) oblasti znamená rekonstrukční proces konvoluci s impulsní charakteristikou ideálního rekonstrukčního filtru.



Obr. 2.6 Rekonstrukce ze vzorků jako interpolace v časové oblasti (převzato z [2])

Rekonstrukce v časové oblasti je dána vztahem (literatura [1])

$$s(t) = \sum_{n=-\infty}^{\infty} s(nT) \sin c\left[\frac{\omega_1}{2}(t - nT)\right]. \tag{2.4}$$

Vztah 2.4 ukazuje, jak v časové oblasti můžeme na základě množiny vzorků stanovit hodnotu signálu *s(t)* pro libovolný reálný okamžik *t*. Funkce *sinc(.)* zde vystupují jako funkce interpolační. Několik členů řady z pravé strany rovnice je nakresleno na obrázku 2.6**.**

Velmi komplexní informace o číslicovém zpracování signálu lze nalézt v literatuře [1].

3 Audio Stream Input/Output (ASIO)

ASIO je ovladač pro zvukovou kartu vyvinutý německou firmou Steinberg. Tato firma se zabývá vývojem specializovaných zvukových zařízení určených pro osobní počítače. Zde jsou uvedeny základní produkty této firmy aby bylo patrné, že jde o velmi profesionální společnost pomocí jejíchž výrobků lze efektivně pracovat se zvukovými signály na všech úrovních.

Produkty:

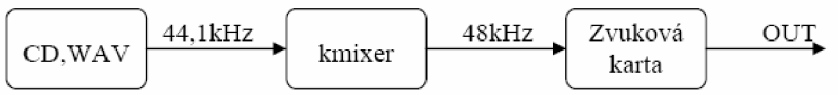
Tvorba hudby – Cubase, Sequel, V-Stack
Dodatečné zpracování zvuku – Nuendo
Úprava zvuku – Wavelab
VST nástroje – Groove Agent, HALion
Hardware – Kontrolery, DSP,…

3.1 ASIO

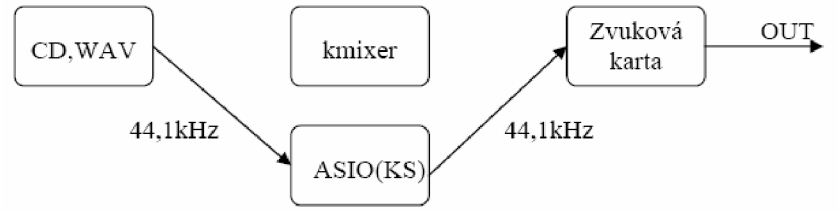
Technologie umožňující pohodlně zpracovávat a synchronizovat vstupy a výstupy na zvukové kartě, tak aby vznikala co nejmenší latence. **Latence** je termín pro časový úsek mezi vstupem signálu do zvukové karty a výstupem v podobě zvuku. Krátké zpoždění je zajištěno neposláním vstupního signálu do operačního systému (MS Windows XP). Hlavním důvod snižování zpoždění je zpracování zvukového signálu v reálném čase. Příkladem využití může být nahrávání zvukového signálu z hudebního nástroje do počítače kdy je třeba aby hudebník slyšel ve sluchátkách zvuk svého nástroje prakticky okamžitě – celý postup digitálního zpracování signálu musí proběhnout v co nejkratším čase.

Nejdůležitějšími úkoly systému ASIO je co největší snížení latence a **multitracking**. Systém tedy umožňuje nahrávat muziku na libovolné množství stop a navíc signál v reálném čase zpracovat (například efektovat) a posílat na libovolné množství výstupů.

Standardně je všechen zvuk přepočítáván v kmixeru (kernel audio mixer driver – součást WDM) na 48kHz, čímž dochází ke zpoždění a degradaci zvuku. Základními ovladači Windows XP pro zpracování a úpravu zvuku jsou MME a DirectX (DirectSound a DirectMusic). Tyto ovladače mají latenci pohybující se v rozmezí od několika desítek ms až do 1s. Takto vysoké zpoždění je pro využití zvukové karty v reálném čase naprosto nepřijatelné. Běžný průchod zvuku systémem je naznačen na obrázku 3.1 a na obrázku 3.2 je naznačeno zpracování zvuku s použitím ASIO ovladače.



Obr. 3.1 Standardní výstup zvuku z PC



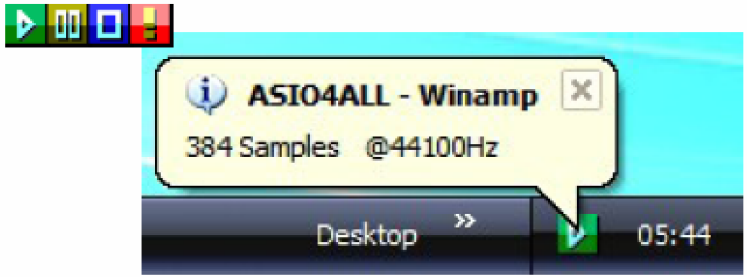
Obr. 3.2 Výstup při použití ASIO

Profesionální systém ASIO byl vyvinut především pro speciální zvukové karty a hudební hardware (např. výrobky firmy Allen&Heath, Mackie, M-Audio atd.). Postupem času se zlepšováním zvukových karet jež jsou standardním vybavením osobních počítačů však přišla nutnost vytvořit systém umožňující využití výhod ASIO i na běžných PC. 5.ledna 2004 tedy vyšla první verze systému ASIO 4 ALL, jenž je možné aplikovat na prakticky jakoukoli zvukovou kartu. Právě tímto systémem se budeme v dalším textu zabývat, nebude-li řečeno jinak. Systém nabízí možnost pracovat s rozdílnými vzorkovacími frekvencemi (32kHz – 96kHz) a různými formáty (16, 24, 32bit). Více informací o historii a vývoji lze nalézt v literatuře [3].

3.2 ASIO4ALL v2 – Universal ASIO Driver For WDM Audio

(Verze 2.9)

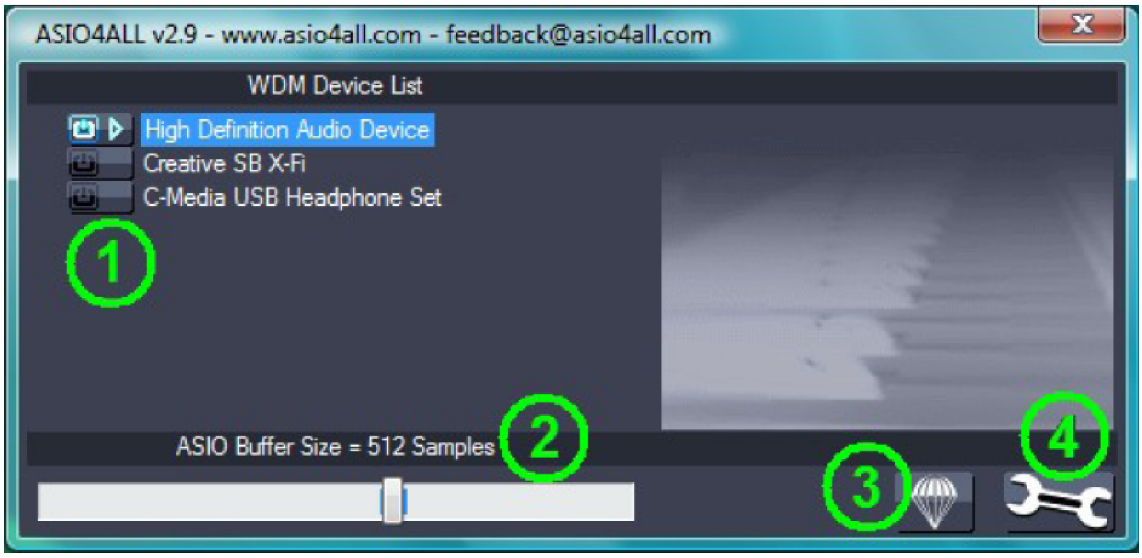
Instalace systému ASIO4ALL v2.9 je intuitivní. Pro využití všech funkcí kterými se text zabývá je třeba nainstalovat všechny části – ASIO4ALL, ReWuschel a Off-Line Settings. Verze 2.9 je k dispozici na přiloženém cd, systém je freeware (distribuován bezplatně). Úspěšné spuštění systému indikuje ikona v System Tray, viz obrázek 3.3.



Obr. 3.3 Indikace chodu systému ASIO4ALL

Ikony charakterizující momentální stav systému jsou vcelku jsou: systém běží, je pozastaven, vypnut nebo došlo k chybě.

3.2.1 Přehled funkcí systému ASIO4ALL



Obr. 3.4 Výchozí konfigurace ASIO4ALL v2 Off-Line Settings

Obrázek 3.4 znázorňuje ovládací panel, jež se zobrazí při prvním spuštění ASIO4ALL. Popis jednotlivých částí je uveden v následujícím textu.

1.) **Device list** – seznam zařízení

V seznamu jsou uvedena WDM audio zařízení detekovaná operačním systémem. Veškerá nastavení se týkají pouze zařízení uvedených v tomto seznamu. Aktuální stav jednotky je vyjádřen malou přiřazenou ikonou. Zařízení může být v těchto stavech:

Aktivní – Zařízení bylo úspěšně spuštěno

Neaktivní – Zařízení je schopno provozu, není ovšem dosud spuštěno

Nedostupné – Zařízení je používáno jinde, například jinou zvukovou aplikací nebo jej nelze použít

Mimo logiku – Zařízení se z neznámého důvodu odmítá spustit – nekomunikuje. Řešením může být znovuspuštění ovládacího panelu nebo odpojení a znovu připojení USB zařízení.

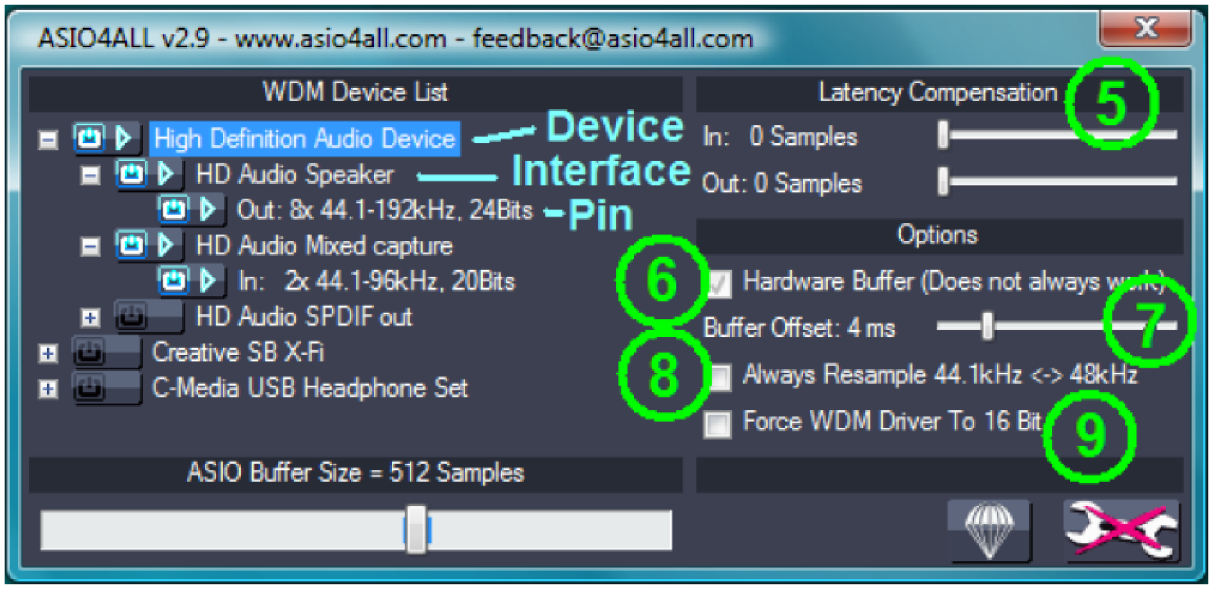
Pokud není v seznamu žádné zařízení, znamená to, že v systému není žádné samostatné WDM zařízení.

2.) **ASIO Buffer Size** – velikost vyrovnávací paměti ovladače

Využívá posuvník k nastavení velikosti vyrovnávací paměti právě zvýrazněného zařízení. Menší velikost paměti znamená nižší latenci. Pokud začne být slyšitelný praskot nebo začne být zvuk deformován, je třeba zvýšit velikost vyrovnávací paměti. Buffer size přímo souvisí s latencí a je tedy žádoucí, aby hodnota byla pokud možno co nejnižší.

3.) **Load Default Settings** – načtení výchozího nastavení

4.) **Switch To Advanced Mode** – přepnutí do rozšířeného módu nastavení



Obr. 3.5 Rozšířená konfigurace ASIO4ALL v2 Off-Line Settings

Po přepnutí do rozšířeného nastavení (obrázek 3.5) se stane seznam zařízení rozbalitelný. Nyní lze plně prozkoumat WDM architekturu systému.

Seznam zařízení obsahuje **Zařízení** (device)**, Rozhraní Zařízení** (device interface) a také takzvané **Piny** (pins). V obrázku 3.5 jsou tyto pojmy vyznačeny. Pomocí tohoto nastavení lze zajistit běh několika zařízení najednou. To vyžaduje aby všechny jednotky běžely se stejným hodinovým zdrojem. Toho lze dosáhnout třeba pomocí nízkoúrovňových hardwarových protokolů pro přenos digitálně kódovaného zvukového signálu – S/PDIF (Sony/Philips Digital InterFace). Naštěstí se většina USB jednotek synchronizuje automaticky se základovou deskou.

5.) **Latency Compensation** – kompenzace latence

Protože ASIO4ALL nemá dostatečnou znalost architektury hardwaru/ovladače, může pouze hádat aktuální latenci. Pomocí posuvníků lze kompenzovat latenci neznámou systému ASIO, tak aby všechny nahrávky byly řádně srovnány.

U vícestopého nahrávání budou použity největší příslušné hodnoty. Je-li tedy použito více zařízení s různými latencemi, zvukové umístění pro některé jednotky nebude správné.

6.) **Hardware Buffer on/of** – spuštění hardwarové vyrovnávací paměti

Tuto vyrovnávací paměť lze zapnout pouze pro miniporty „WavePCI“, ostatní typy WDM ovladačů obvykle neumožňují přímí přístup k hardwarové vyrovnávací paměti.
Pro nejlepší funkci hardwarové vyrovnávací paměti je dobré nastavit co nejnižší vyrovnávací pamět ASIO ovladače, zhruba mezi 128 a 256 vzorky.

Největší výhodou využití této paměti je menší zatížení CPU a možnost dalšího snížení latence. Při vícestopém využití lze hardwarovou vyrovnávací pamět kombinovat i se zařízeními, které tuto paměť nemají, není to ovšem doporučený postup. Pokud tato paměť není patřičnou jednotkou podporována, vede to k znatelnému nárůstu latence až v řádech stovek milisekund, což je již zřetelně slyšitelné.

Pro WaveRT ovladač (Vista), je tento box pojmenován „Allow Pull Mode (WaveRT)“ - použitím WaveRT(Vista) se zde nebudeme zabývat.

7.) **Kernel Buffers/Buffer Offset** – offset jádra vyrovnávací paměti

Pokud je hardwarová vyrovnávací paměť vypnuta, tento ovládací prvek dovolí přidat pro audio výstup další dvě vyrovnávací paměti do fronty. Každá přidaná vyrovnávací paměť zvýší výstupní latenci, proto by výchozí nastavení „2“ mělo být měněno pouze u málo výkonných strojů, kde malé ASIO vyrovnávací paměti nelze dosáhnout ve výchozím nastavení.

Pokud je hardwarová vyrovnávací paměť zapnuta, ovládací prvek umožňuje nastavit dobu (v milisekundách) mezi zápisem/čtením dat z hardwarové vyrovnávací paměti a aktuální pozicí.
Vyšší hodnoty mají za důsledek zvýšení latence a zlepšení stability, nižší mají opačný efekt. Nicméně je dobré nastavit hodnoty velmi blízké nule, což může být třeba 4ms, zatímco výchozí hodnota 10ms ukazuje, že je zde stále prostor pro zlepšení.

V případě zvukových karet typu Evy24-based PCI (např. Terratec) lze nastavit v ovládacím panelu zvukové karty položku „DMA Buffer Transfer Latency“. Opět je žádoucí nastavit co nejnižší hodnotu, v nejlepších případech lze dosáhnout i 1ms.

8.) **Always Resample 44.1 <->48 kHz** – zapnutí stálého převzorkování

Systém ASIO4ALL je schopen převzorkování signálu v reálném čase. Převzorkování automaticky probíhá kdykoli je systém spuštěn a WDM nevyžaduje jinou vzorkovací frekvenci.

Mohou nastat případy, kdy AC97 bude podporovat vnitřní převzorkování na 44.1 kHz. Kvalita převzorkování AC97 je ovšem extrémě slabá a může narušit stabilitu. Tomu lze předejít právě zapnutím stálého převzorkování pomocí ASIO. Při použití ovladače SoundMax WDM (smwdm.sys) je toto nastavení absolutně nutné pro práci na frekvenci 44.1 kHz.

9.) **Force WDM driver to 16 bit** – nastavení bitové hloubky WDM

Toto nastavení má efekt pouze je-li bitová hloubka WDM ovladače větší než 16 bitů a zároveň menší než 24. Některá zařízení AC97 hlásí 20 bitové rozlišení, ale nemohou být skutečně použity pro rozlišení vyšší než 16 bitů. S tímto se konkrétně potýkal ovladač SigmaTel AC97 WDM.

3.2.2 Způsob použití systému ASIO4ALL

Software pro živé hraní

V tomto případě nejsou potřeba audio vstupy. Nejlepší je tedy všechny vypnout, tím se zlepší stabilita při velmi malých velikostech vyrovnávací paměti. Dále je také vhodné odpojit všechny audio výstupy, jež nejsou skutečně potřeba. K vypnutí kanálů slouží rozšířený kontrolní panel – v rozbaleném seznamu zařízení lze vypnout vše potřebné.

Počítač jako efektový procesor

Vstupy opět nejsou potřeba a je vhodné vypnout všechny nevyužívané kanály. Vypnout resampling pokud to není skutečně nutné.

Sekvencer

K nejvíce výpadkům dochází při velké zátěži procesoru. Proto při využití VST je nejlepší regulovat latenci pomocí vyrovnávací paměti ASIO podle zrovna používaného pluginu. Také je třeba na to myslet při nahrávání, kdy jsou výpadky velmi nežádoucí.

Na latenci příliš nezáleží

V některých konfiguracích je ASIO4ALL vhodnější pro čistší zvuk než klasický ovladač pro Windows. Proto hudebníci preferují ASIO výstup před DirectSound neb MME, které zvuk mnohem více zkreslí. Pro tento případ je latence málo důležitá a audio vstupy nejsou potřebné. Je třeba se ujistit že všechny vstupy jsou vypnuté a nastavit velikost ASIO vyrovnávací paměti na maximum.

3.2.3 ReWuschel

Použitím ReWuschel se audio vstupy ASIO4ALL stanou přístupné jako ReWire vstupy. To je vhodné pro aplikace jež využívají ASIO pouze k výstupu, ale podporují ReWire. Příkladem může být **Reason** firmy Propelerhead kde za pomoci ReBirtch Input Machine a ASIO lze získat audio vstupy z jiných programů v reálném čase. Na obrázku 3.6 je panel ReBirth z programu Reason.

Obr. 3.6 ReBirth Input Machina – Reason 3

Informace potřebné pro práci s ASIO4ALL jsou obsaženy v literatuře [4].

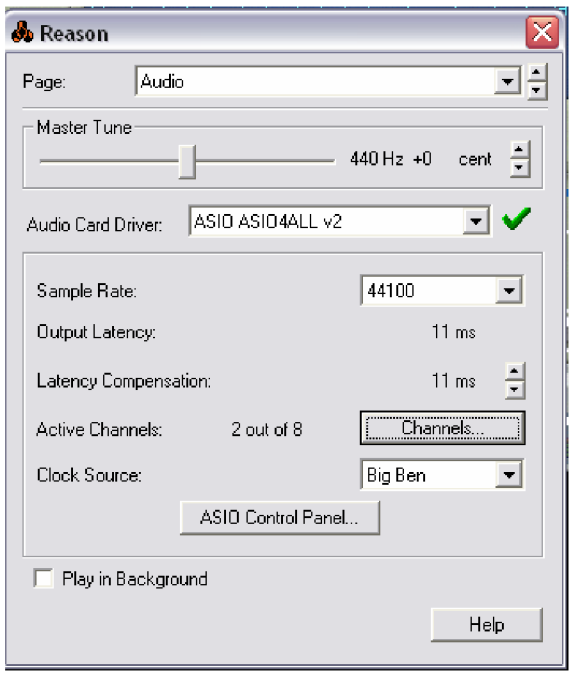
3.2.4 Srovnání latence různých ovladačů

Na obyčejné zvukové kartě *Realtek High Definiton audio od firmy Semiconductor* bylo provedeno srovnání základních ovladačů zvuku se systémem ASIO. Výsledky jsou získány pomocí zvukového nastavení programu Reason. Při použití systému ASIO4ALL bylo dosaženo jednoznačně nejlepších výsledků s latencí 11ms. Naproti tomu při využití ovladače MME byla latence 185 ms a byla jasně slyšitelná. Výsledky testu jsou zobrazeny v tabulce a na obrázku xx je vzhled panelu pro nastavení audio výstupu z programu Reason 3.

Použité nastavení ASIO4ALL: *ASIO Buffer size: 448 vzorků, Latency Compensation: 32 vzorků pro vstup i výstup, Kernel Buffers:2*

Tab. 3.1 Srovnání latence různých ovladaču – hodnoty získané v programu REASON

Ovladač	Vzorkovací frekvence [Hz]	Latence [ms]
ASIO4ALL v2	44 100	11
MME Realtek HD output	44 100	185
DX Realtek HD output	44 100	92



Obr. 3.7 Ilustrační snímek k nastavení ovladače v programu Reason

4 Návrh aplikačního prostředí

Při návrhu aplikace bylo jedním z cílů připravit prostředí, jež bude plně využívat funkcí ASIO4ALL. Postupně bylo pracováno se třemi různými možnostmi přístupu k tomuto ovladači – ASIO SDK, PortAudio a VCL AudioLab. Budou zde shrnuty jejich základní vlastnosti a metody použití.

Srovnání jednotlivých přístupů k ovladači ASIO je založeno na několika kritériích. Prvním je výstupní latence. Tento údaj je v běžné praxi poněkud zavádějící, neboť celková latence odpovídá této výstupní pouze v jednom ideálním případu, a to když je přehráván jeden soubor formátu wav a datový proud je posílán přímo na výstup zvukové karty. Při běžném použití je celková latence součtem vstupní latence, výstupní latence, latence způsobené uživatelskou aplikací, latence jednotlivých komponent a samozřejmě doby zpracování zvukového signálu (filtrace, modulace,…). Celková latence je tedy vždy určena konkrétním zpracováním signálu a je tedy větší než latence výstupní. Pro srovnání je ovšem dobré použít hodnotu jež má vždy stejný význam, tedy právě latenci výstupní. Všechny tři přístupy pro práci s ovladačem ASIO obsahují funkci vracející přímo hodnotu výstupní latence. Dalšími kritérii pro hodnocení je složitost ovládání systému, stabilita a množství potřebného zdrojového kódu.

4.1 ASIO SDK

(Software Development Kit) je balíček vývojových nástrojů od firmy Steinberg určený pro tvorbu aplikací s podporou ovladače ASIO4ALL. V podstatě jde o několik souborů obsahujících všechny funkce jež tyto ovladače nabízejí. Vývojové nástroje a dokumentaci poskytuje firma Steinberg po registraci zdarma. Potřebné informace jsem čerpal především z literatury [5].

4.1.1 Princip

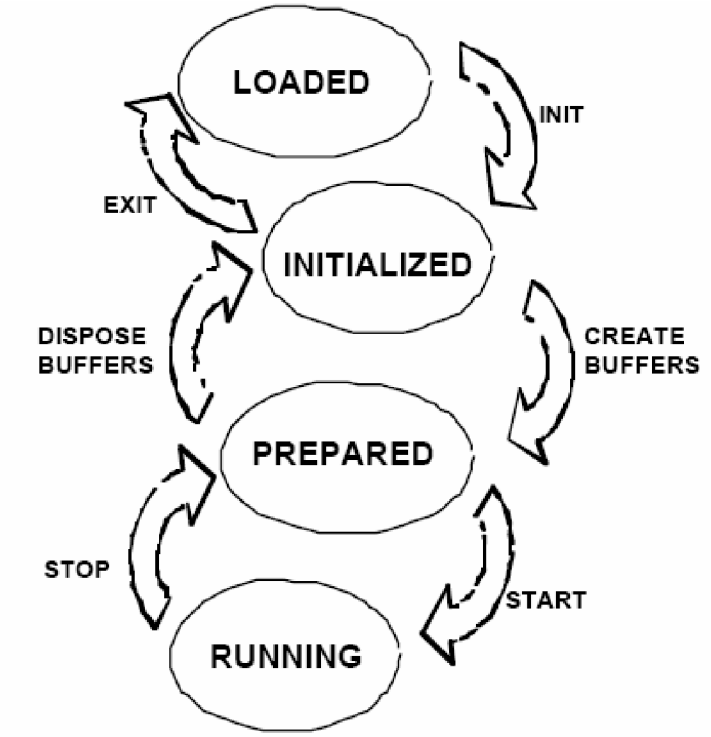
Hostující aplikace načte ovladač a odkáže jej na sebe. Přístup ovladače k aplikaci je patrný z obrázku 4.1, na němž je zobrazen životní cyklus ovladače.

Initialized: Ovladač je alokován aplikací a zjistí zda může být použit. Tato procedura není nezbytná pokud je hardware jednoznačně přidělen

Prepared: Jsou alokovány audio buffery a ovladač je připraven ke spuštění

Running: Hardware je v běhu a datový proud je spuštěn

Pro správné použití ovladače je třeba tyto kroky dodržet, především vždy vytvořit a po použití zrušit vyrovnávací paměť (buffers).



Obr. 4.0.1 Životní cyklus ovladače ASIO

4.1.2 Výsledky:

Práce s ASIO SDK umožňuje dosažení nejlepších výsledků (nejnižší latence), je ovšem velmi náročná. Pro správné použití ovladače je třeba znalost mnoha funkcí obsažených v SDK a je třeba napsat mnoho řádků zdrojového kódu. Při špatné manipulaci může dojít k nestabilitě a kolapsu operačního systému. Výstupní latence naměřená pomocí návratové funkce byla téměř rovna jedné milisekundě, testování se ovšem podařilo zprovoznit pouze pro konzolovou aplikaci.

4.2 PortAudio

PortAudio je knihovna určená pro práci se vstupně-výstupními zvukovými proudy. Jde o API (Application Programming Interface) pracující s mnoha platformami (Windows, Macintosh OS X, Linux, …). Lze tedy napsat jednoduchý „C“ program pro zpracování či generování audio signálu, jež poběží na rozdílných typech počítačů po překompilování zdrojového kódu. Hlavní výhodou je poměrně snadné ovládání umožňující práci s libovolným audio API. Životní cyklus ASIO ovladače musí být zachován.

4.2.1 Kroky pro psaní PortAudio aplikace

- Napsání návratové funkce jež bude volána PortAudiem když bude třeba provézt zpracování zvuku
- Inicializace PortAudio knihovny a otevření datového proudu pro zvukový vstup/výstup
- Start proudu. Návratová funkce bude nyní volána opakovaně pomocí PA na pozadí.
- Voláním lze číst audio data ze vstupní vyrovnávací paměti a zapisovat do paměti výstupní.
- Zastavení proudu pomocí „stop function“.
- Zavření proudu a ukončení práce s knihovnou.

PortAudio nabízí zjednodušení některých funkcí ASIO SDK, například jednou funkcí PortAudio „Pa_Initialize“ dojde provedení několika funkcí ASIO SDK (loadAsioDriver, ASIOInit, ASIOGetChannels, ASIOCanSampleRate, ASIOGetChannelInfo).

Při ovládání ASIO pomocí PortAudio je třeba stále dodržovat hierarchii životního cyklu ASIO. Ovládací funkce jsou tyto:

```
ASIOError ASIOInit(ASIODriverInfo *info); //inicializace
ASIOError ASIOCreateBuffers();            //vytvo•ení vyrovnávací pam•ti
ASIOError ASIOStart();                    //spušt•ní ovlada•e
ASIOError ASIOStop();                     //zastavení ovlada•e
ASIOError ASIODisposeBuffers();           //zrušení vyrovnávací pam•ti
ASIOError ASIOExit();                     //uvoln•ní ovlada•e
```

4.2.2 Spuštění ovladače ASIO pomocí PortAudio API

Pro kompilaci ASIO aplikace je třeba nejdříve stáhnout ASIO SDK. Dále je také třeba nainstalovat ovladač ASIO pro vaší zvukovou kartu (pro běžné zvukové karty ASIO4ALL). Použitá Verze PortAudio je V19.

Prvním krokem je vložení hlavičkového PA souboru do zdrojového kódu projektu

```
#include "portaudio.h"
```

Poté je třeba přidat k projektu následující .cpp soubory s přístupem k hlavičkovým souborům a projekt zkompilovat.

```
pa_asio.cpp                     (Portaudio\src\hostapi\asio)
pa_asio.h
asio.cpp                        (asio2sdk\common)
asio.h
asiodrivers.cpp                 (asio2sdk\host)
asiodrivers.h
asiolist.cpp                    (asio2sdk\host\pc)
asiolist.h
```

Dále je třeba přidat k projektu tyto .c soubory s přístupem k hlavičkovým souborům:

```
pa_allocation.c                 (Portaudio\src\common)
pa_allocation.h
pa_converters.c                 (Portaudio\src\common)
pa_converters.h
pa_cpuload.c                    (Portaudio\src\common)
pa_cpuload.h
pa_dither.c                     (Portaudio\src\common)
pa_dither.h
pa_front.c                      (Portaudio\src\common)
pa_process.c                    (Portaudio\src\common)
pa_process.h
pa_skeleton.c                   (Portaudio\src\common)
pa_stream.c                     (Portaudio\src\common)
pa_stream.h
pa_trace.c                      (Portaudio\src\common)
pa_trace.h
pa_win_hostapis.c               (Portaudio\src\os\win)
pa_win_util.c                   (Portaudio\src\os\win)
pa_win_waveformat.c             (Portaudio\src\os\win)
pa_win_waveformat.h
```

```
pa_x86_plain_converters.c       (Portaudio\src\os\win)

pa_x86_plain_converters.h

pa_win_wmme.c                   (Portaudio\src\hostapi\wmme)
```

Také je třeba aby byly k dispozici tyto soubory:
`asiodrv.h`
`asiodrv.cpp`
`asiosys.h`
`combase.cpp`
`combase.h`
`ginclude.h`
`iasiodrv.h`
`pa_debugprint.h`
`pa_endianness.h`
`pa_hostapi.h`
`pa_types.h`
`pa_util.h`
`pa_win_wmme.h`

Soubory potřebné pro běh pod MS Windows (jiné operační systémy je nevyžadují):
`assiothiscallresolver.h`
`assiothiscallresolver.cpp`

A samozřejmě:
`portaudio.h`

Celkově jde o 48 souborů, jež postačují pro vývoj většiny aplikací využívajících ovladače ASIO s PortAudio API.
Pro plnou funkčnost je třeba ještě editovat soubor `pa_win_hostapis.c` a přidat následující definice:

```
#define PA_NO_WMME
#define PA_NO_DS
```

4.2.3 Psaní návratové funkce

Pro psaní programu používajícího PortAudio, je vždy nutné vložit hlavičkový soubor "portaudiol.h". Tento soubor obsahuje popis všech potřebných funkcí a konstant PortAudio. Dalším úkolem je napsání funkce sloužící k zpětnému dotazu, tzv. „callback function". Tato funkce je volána pokaždé, když PortAudio zachytí audio data nebo jsou potřeba další audio data pro výstup.

4.2.4 Výsledky:

Použití PortAudio částečně odstraňuje hlavní vadu ASIO SDK, tedy nutnost psaní velkého množství zdrojového kódu, složitost ovšem zůstává poměrně vysoká. Jednou z výhod tohoto přístupu je přehledně vytvořená a obsáhlá dokumentace dostupná na http://portaudio.com/. Nevýhodou je však nutnost připojení mnoha souborů k projektu a menší efektivnost při využití ASIO ovladače. Latence je zvýšena díky využití další vyrovnávací paměti. Informace o práci s PortAudio byly získány především z dokumentů [6] a [7].

4.3 AudioLab VCL TALASIOAudioDevice

Jedná se o komponentu pro c++ Builder vytvořenou firmou MitovSoftware, jež je součástí balíčku komponent AudioLab. Tato komponenta byla doplněna až v nedávné době v poslední verzi AudioLab 4.0. Práce je velmi pohodlná, většinu vlastností lze nastavit přímo v object inspectoru c++ Builderu. Velkou nevýhodou je nárůst latence a problémy při používání více připojených komponent současně. Při vysoké složitosti aplikace se rapidně zvyšují nároky na výkon počítače. Podrobnější rozbor komponenty je uveden v další kapitole.

Tab. 4.1 Srovnání jednotlivých přístupů k ovladači ASIO4ALL v2.9. Uvedená výstupní latence je nejnižší možná.

Srovnání přístupů	Výstupní latence [ms]	Složitost manipulace	Stabilita
ASIO SDK	1,088	Velmi vysoká	Vysoká
PortAudio	7,896	Střední	Nízká
AudioLab	13,084	Nízká	Střední

5 Tvorba aplikačního prostředí

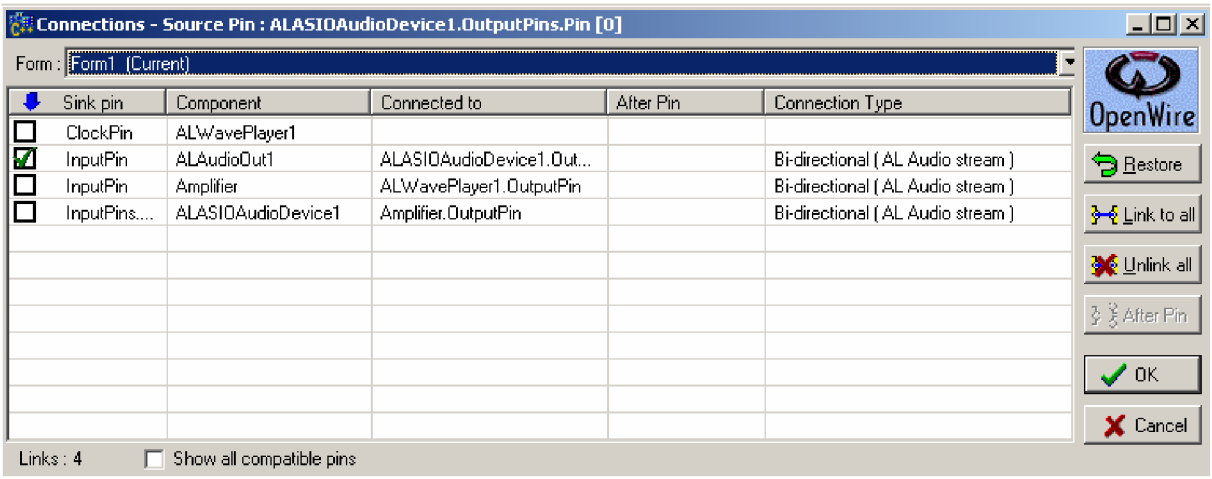
Po zvážení dosažitelných možností při tvorbě aplikačního prostředí bylo rozhodnuto postavit výslednou aplikaci s využitím VCL komponent od firmy MitovSoftvare. Tato firma nabízí několik balíčků komponent pro práci s datovými proudy a technologii OpenWire sloužící k propojení jednotlivých komponent. Všechny balíčky jsou pro nekomerční použití k dispozici zdarma, není ovšem možnost editovat zdrojové kódy komponent. Nabízené balíčky jsou: Audio Lab, Signal Lab, Plot Lab, Timing Lab, OpenWire, Video Lab, Instrument Lab, Vision Lab. Celkově se jedná o několik stovek komponent, z nichž některé byly použity při tvorbě aplikace. Informace o nejdůležitějších jsou zmíněny v dalším textu, o ostatních lze nalézt informace na webu: http://mitov.com/. Pro základní seznámení s VCL MitovSoftvare jsou vhodné QuickStart manualy od jednotlivých balíčků.

5.1 AudioLab

Sestava komponent určená pro zpracování zvuku. Knihovna umožňuje nahrávat, přehrávat, mixovat, analyzovat a zobrazovat zvukové signály při minimální potřebě psaní zdrojového kódu. Jedná se o VCL komponenty určené pro Delphi, C++Builder a .NET. AudioLab podporuje Win32 API (WaveAPI, Audio ACM), DirectX (DirectShow, DMO – Direct Media Object), Intel (Intel MMX, IPP - Intel Performance Primitives) a nejnovější verze obsahuje i komponentu pro práci s ovladači ASIO.

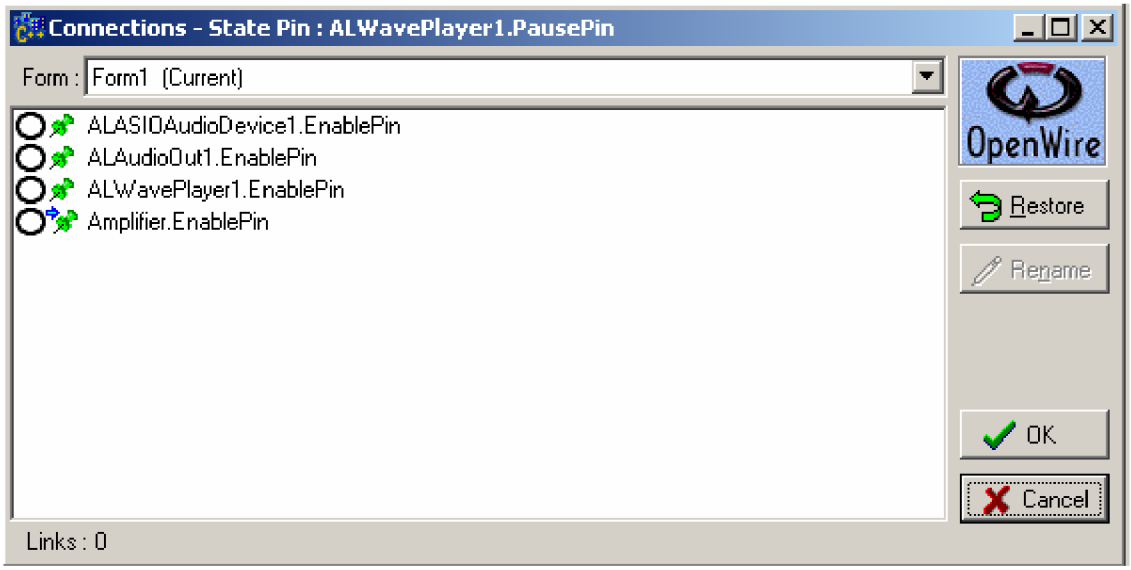
5.2 OpenWire

Technologie umožňující VCL a CLX komponentám vyměňovat data a oznámení o událostech mezi sebou pomocí jednoduchého systému. Jde o volně šiřitelnou knihovnu jež umožňuje komunikaci mezi komponentami bez potřeby znalosti jejich vzájemného vztahu. OpenWire definuje dva typy výměny dat – Streaming a State. Streaming je navržen pro posílání nepřetržitého datového proudu ze zdroje do přijímače. To je typické například pro získávání dat (nahrávání zvuku). State se používá pro synchronizaci vícenásobných komponent nebo pro synchronizaci s databázovými komponentami. Propojení jsou tvořena pomocí komponent nazývaných Piny (Pins). Tyto piny mohou být trojího druhu – Source, Sink nebo State.



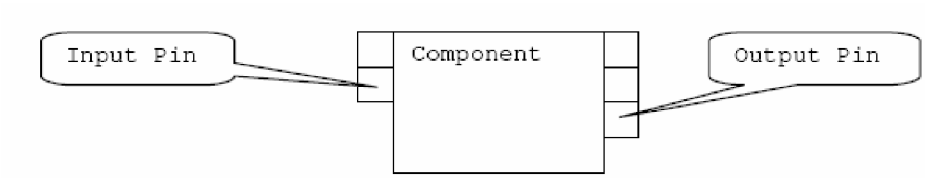
Obr. 5.1 Příklad zapojení source pins pomocí OpenWire

..



Obr. 5.2 Příklad připojení state pinů

Source a Sink piny jsou považovány za streamingové, nicméně je lze použít i pro State spojení. State piny jsou navrhnuty pro výměnu informací o stavu komponenty.



Obr. 5.3 Použití OpenWire (převzato z QuickStart OpenWire)

Na obrázku 5.3 je komponenta, která má 2 vstupní a 3 výstupní piny. Všechny vstupy a výstupy v OpenWire se nazývají souhrnně Piny. Vstupním pinům se říká Sinks a výstupním Sources. Každý Source pin může být připojen na více pinů sink. Jeden pin může vysílat datový proud různých typů (Float, Integer, Complex,…) pro další komponenty. Piny jsou optimalizovány pro velký výkon. Všechna spojení mohou být navržena bez potřeby psaní zdrojového kódu.

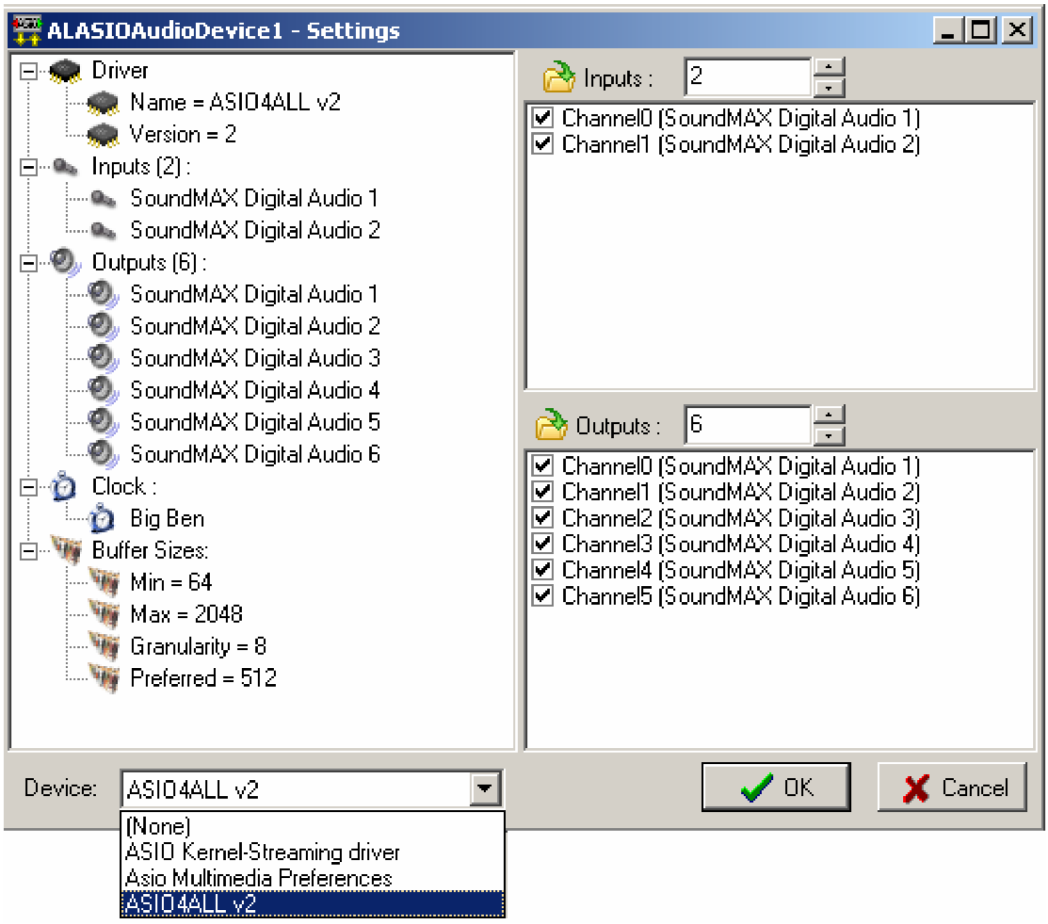
6 Zpracování signálu

6.1 Základní operace s audio signálem

6.1.1 ASIOAudioDevice

Jedná se o VCL Audio Lab, jež je k dispozici od verze Audio Lab 4.0. Na obrázku 6.1 je zobrazen panel nastavení této komponenty. Nejprve je třeba vybrat ze seznamu zařízení jež bude použito (vlevo dole). Při výběru ASIO4ALL je ovladač okamžitě v návrhovém zobrazení aktivován v režimu pause a objeví se příslušná ikona v systet tray (obrázek 3.3). Po uzavření nastavení je ovladač opět deaktivován. V levé části ovládacího panelu jsou informace o ovladači – název a verze ovladače, počet dostupných vstupních a výstupních kanálů, časování a informace o vyrovnávací paměti. Hodnota „Preffed Buffer Size“ určuje automatickou velikost vyrovnávací paměti. Pokud je třeba nastavit velikost vyrovnávací paměti manuálně, je nutno v object inspektoru nastavit vlastnost `BufferSize-UserPreffered` na `false` a ve vlastnosti `Size` zadat požadovanou velikost vyrovnávací paměti v rozmezí Min-Max zobrazeném v nastavení komponenty. V pravé části nastavení je zobrazen seznam aktivních vstupních a výstupních kanálů. Běžné zvukové karty umožňují využití dvou vstupních a šesti až osmi výstupních kanálů. Nastavení vstupních kanálů se provede v object inspectoru pomocí vlastnosti InputChannels-OutputPin a nastavení výstupních pomocí OutputChannels-InputPin. Jedinou přímo dostupnou vlastností jednotlivých kanálů je enabled (zapnuto – ano/ne).

ASIOAudioDevice funguju jako směšovač. Pokud připojíme další komponenty na jednotlivé kanály, dojde k jejich sloučení do jednoho proudu. Životní cyklus ovladače je kontrolován samotnou komponentou, při nastavení `enabled = true` dojde při spuštění aplikace i k plnému spuštění ovladače. Ovládat běh ovladače v run-time módu lze pomocí funkcí `start()` a `stop()` jež spustí/vypnou ovladač a funkcí `Open()` a `Close()` jež vytvoří/zruší vyrovnávací paměť a ošetří všechny ostatní potřebné kroky v životním cyklu ovladače (cyklus je zobrazen na obrázku 4.1).



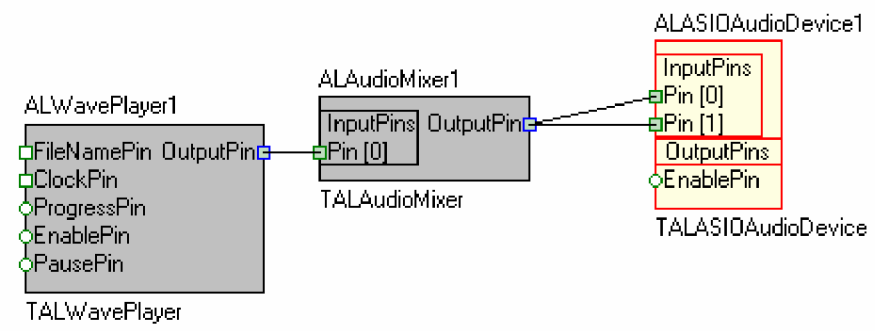
Obr. 6.1 Nastavení ASIOAudioDevice AudioLab

Ovladač ASIO4ALL se automaticky spustí při spuštění aplikace a jeho aktivita je indikována ikonou v syste tray. Poklepáním na tuto ikonu lze spustit dialogové okno ASIO4ALL off-line settings (obrázek 3.4). Jak již název napovídá, toto nastavení slouží ke změně parametrů ovladače ve vypnutém stavu. To je akceptováno mnoha profesionálními programy (např. NI Kontakt, Reaktor, …) jež je třeba restartovat aby se změny provedené v off-line settings projevily. Toto pravidlo však není neporušitelné. Příkladem může být opět program Propelerhead Reason, u nějž se provedené změny projevují v reálném čase. Při návrhu aplikace se bohužel nepodařilo navázat spolupráci s off-line settings a změny provedené v tomto nastavení nemají žádný vliv na parametry vlastní aplikace, pouze zobrazují stav ovladače při spuštění aplikace. Změnu parametrů ovladače lze tedy provézt pouze v návrhovém prostředí c++ Builderu.

6.1.2 Přehrávání

Některé principy a metody přehrávání a nahrávání zvukových souborů lze nalézt v literatuře [8], zde jsou zmíněny pouze o postupy jež byly využity při návrhu aplikace.

Jako základní komponenta je použit WavePlayer (AudioLab) jež slouží pro přehrávání souborů typu wav. AudioLab sice nabízí komponentu jež umožňuje přehrávání více typů zvukových souborů (DSAudioPlayer), je však určena pro přehrávání pomocí DirectX a není ji tedy možné využít pro připojení k ovladači ASIO. K dispozici jsou v balíčku AudioLab i přehrávače pro formáty ogg a raw. Pro možnost přehrávání MPEG-1 a dalších komprimovaných formátů je třeba použít dodatečné knihovny určené pro kompresi/dekompresi audio signálu. Ovládání komponenty pomocí zdrojového kódu je velmi snadné, význam funkcí je ihned zřejmý (`start(), stop(), pause(),...`). Za zmínku stojí vlastnost prgress pin, jež umožňuje sledování aktuální pozice přehrávače ve zvukovém souboru (lze připojit na měření času, progress bar,...). Další komponenta jež je obsažena ve všech zapojeních je AudioMixer. V podstatě jde o softwarový mixážní pult s nastavitelným počtem kanálu. Zapojení přehrávače je na obrázku 6.2.



Obr. 6.2 Schématické zapojení přehrávač pomocí OpenWire

6.1.3 Nahrávání

Záznam zvukových souborů bohužel s pomocí komponenty ASIOAudioDevice nelze uskutečnit neboť není kompatibilní s komponentou ovládající audio vstupy. V případě vynechání ovladače ASIO by nahrávání pomocí komponentů AudioLab bylo složeno z komponent AudioIn a WaveLogger.

6.2 Filtrace

Při zpracování hudebního signálu je často nutné některé jeho složky potlačit z důvodů vylepšení zvukového vjemu či odstranění šumu. Filtrace slouží k výběru jistých složek ze směsi více signálů a k potlačení složek jiných. Obecně lze chápat filtraci jako prostředek, umožňující měnit vlastnosti jednotlivých složek, např. jejich poměrné zastoupení nebo vzájemné časové vztahy ve výsledném signálu. Složky signálu jsou nejčastěji chápány ve frekvenční oblasti – pak jde o harmonické komponenty, jejichž amplitudy a časové vztahy (vyjádřené počátečními fázemi) se filtrací pozmění. Lineární časově invariantní filtry vždy realizují konvoluci mezi svou impulsní charakteristikou a vstupním signálem.

6.2.1 Číslicová filtrace

Číslicové filtr navazují na pasivní a aktivní analogové filtry a lze je navrhovat buď přímo (FIR), nebo převedením analogového prototypu (IIR). V reálném čase musí filtr mezi dvěma vzorky provést výpočet konvoluce, proto pro řešení pomocí algoritmu jsou vhodné filtry FIR.

Vlastnosti FIR filtrů

FIR = finite impulse response, jsou filtry s konečnou impulsní charakteristikou. Jsou plně definovány N hodnotami této charakteristiky, které tvoří současně vektor systémových konstant $\mathbf{h} = [h_n]$, $n \in \langle 0, N-1 \rangle$. Jejich diferenciálníí rovnice vyjadřuje konečnou diskrétní konvoluci

$$y_n = \sum_{k=0}^{N-1} x_{n-k} h_k \tag{6.1}$$

což je současně vyjádření tzv. přímého realizačního algoritmu, takže hodnoty impulsní charakteristiky jsou přímo systémovými realizačními konstantami. Obrazový přenos je

$$H(z) = \sum_{n=0}^{N-1} h_n z^{-n}, \tag{6.2}$$

takže reprezentace v rovině z je dána *jen nulovými body* (násobný pól v počátku vyjadřuje jen zpoždění resp. Fázový posun) a následně jsou FIR filtry *absolutně stabilní*. Frekvenční charakteristika je periodická funkce s periodou $2\pi/T$, vyjádřena Fourierovou řadou s koeficienty h_n

$$G(\omega) = H\left(e^{j\omega T}\right) = \sum_{n=0}^{N-1} h_n e^{-j\omega nT}. \tag{6.3}$$

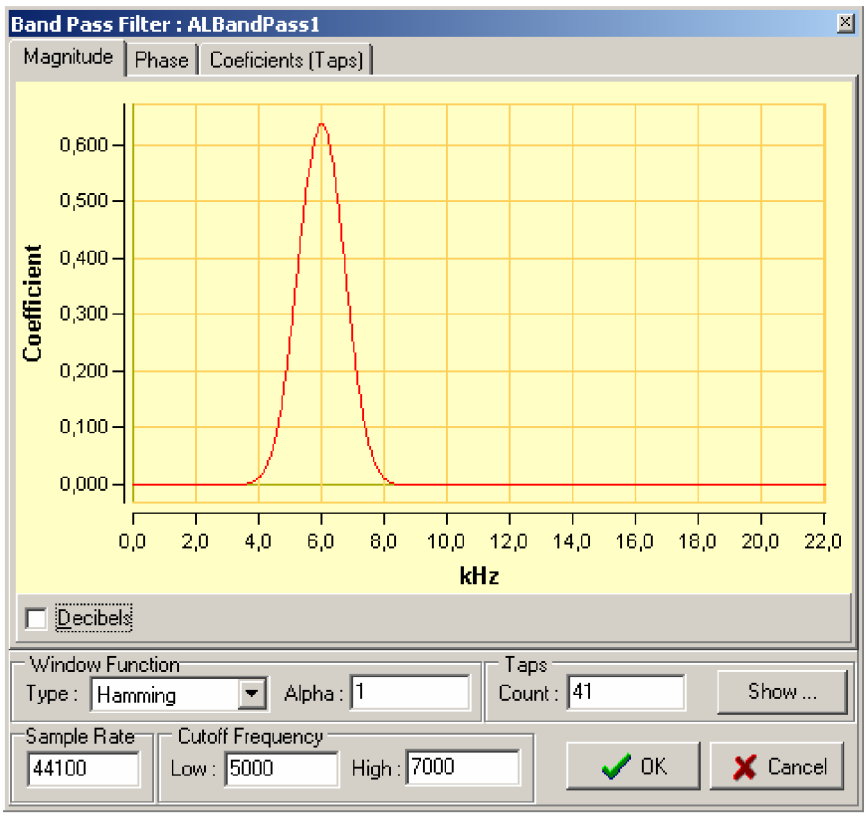
Bezprostřední souvislost frekvenční charakteristiky se systémovými koeficienty realizace umožňuje poměrně snadný návrh těchto filtrů.

V praxi lze použít několik metod pro návrh FIR filtrů. Jednou ze základních je **metoda váhování impulsní charakteristiky**. Tato metoda vychází ze znalosti obecně neomezeně dlouhé impulsní charakteristiky $\mathbf{h_d}$, popisující přesně požadovaný filtr. Vzhledem k tomu, že zpravidla jsou požadavky na filtr specifikovány ve frekvenční oblasti, tvoří pak prvý krok návrhu výpočet přesných hodnot $\mathbf{h_d}$. Při návrhu vycházíme z požadované frekvenční charakteristiky. Odtud dostaneme požadovanou impulsní odezvu, kterou vynásobíme vhodným oknem. Volba typu okna je při návrhu filtru důležitým rozhodnutím. Má vliv na zvlnění amplitudové charakteristiky a strmost přechodu z propustného do nepropustného pásma. Při využití komponent AudioLab je možnost vybrat jedno z těchto oken: obdélníkové, Bartlettovo, Blackmanovo, Hammingovo, Hanningovo a Kaiserovo. Při návrhu grafického ekvalizéru bylo použito váhování Hammingovým oknem. Podrobné vysvětlení realizace číslicových filtrů lze nalézt opět v literatuře [1].

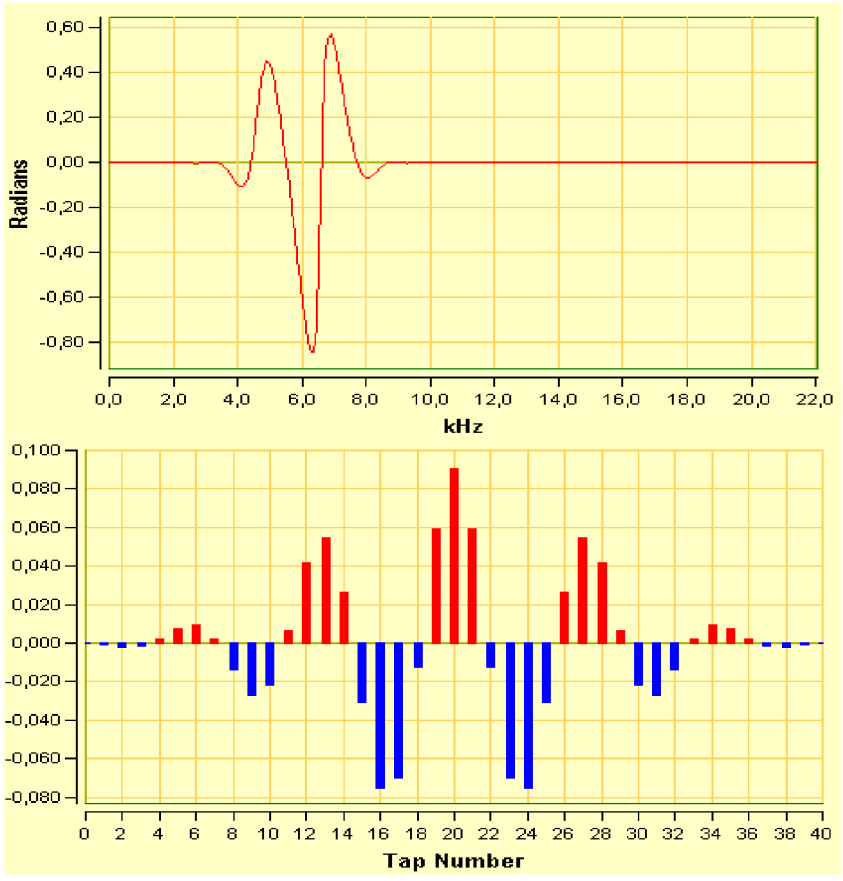
6.2.2 Ekvalizér

Ekvalizér je zařízení (nebo jeho část) sloužící k úpravám frekvenční charakteristiky zvukového signálu zesílením či potlačením některých částí akustického spektra, tedy tzv. pásem. Může být realizován jako softwarový algoritmus v digitálním systému, ve kterém je zvuk zpracováván. Základním typem je **grafický ekvalizér**, kde je signál přiveden na vstup sady pásmových propustí a za nimi následuje stejný počet potenciometrů. Signál z potenciometrů se sčítá a přivádí na výstup. Celé zapojení je provedeno tak, že při poloviční výchylce je frekvenční charakteristika nezměněna. V praxi se grafické ekvalizéry dělí podle počtu pásem. U profesionálních zařízení se rozlišují ekvalizéry podle počtu pásem na jednu oktávu (1,1/2,1/3), kde třetinooktávový ekvalizér má 32 pásem. Ekvalizéry s menším počtem pásem se pak objevují ve vstupních jednotkách mixážních pultů, u předzesilovačů, v kytarových a jiných kombech apod. Grafické ekvalizéry mají dvě zásadní omezení. Počet pásem, jejich šířka a střední frekvence jsou pevně dány, stejně jako rozsah regulace přenosu jednotlivých pásem. Z toho plyne jejich použití výhradně na úpravu frekvenční charakteristiky, např. pro korekci vlastností reprosoustav. Jejich použití pro odfiltrování nežádoucích zvuků je značně omezené a ani pro to nejsou určeny.

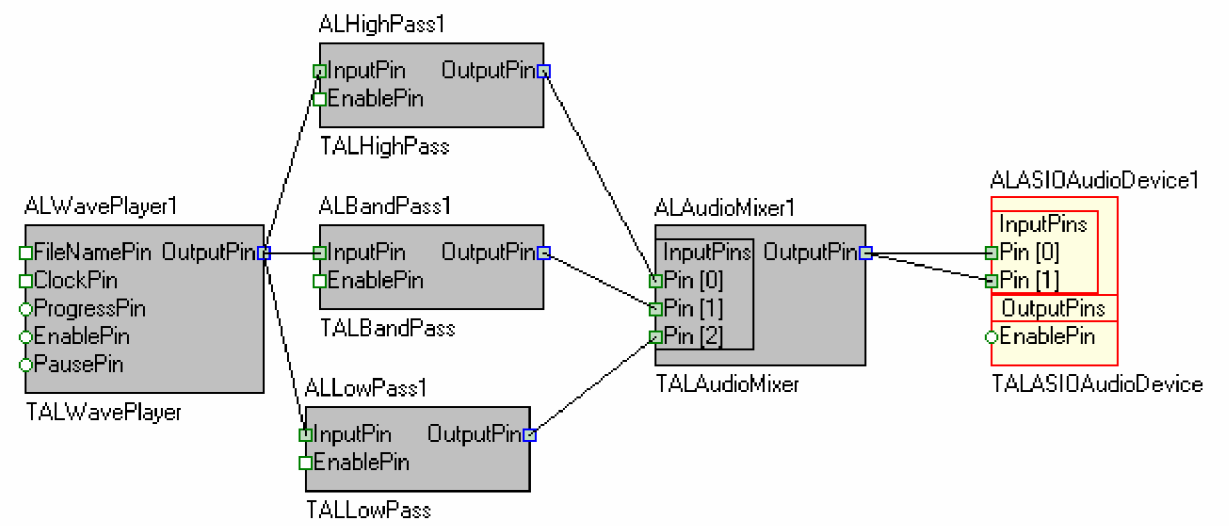
Pro aplikaci byl navržen a zrealizován tří pásmový grafický ekvalizér. Balíček AudioLab sice nabízí komponentu pro tvorbu grafického ekvalizéru (TALGraphicEqualizer), není ovšem kompatibilní s operačním systémem MS Windows XP 64bit na kterém probíhala velká část realizace aplikace. Návrh digitálního filtru pomocí komponent AudioLab je velmi snadný, lze jej uskutečnit bez nutnosti psaní zdrojového kódu. AudioLab nabízí komponenty pro návrh pásmové zádrže, pásmové propusti, dolní a horní propusti. Realizace pásmové propusti je ilustrována na obrázcích 6.3 a 6.4. Z obrázku je patrné, že lze navrhnout vlastnosti filtru (dolní a horní mezní frekvenci, počet koeficientů, vzorkovací frekvence, váhovací okno,…) pouhým zadáním parametrů. Princip zapojení je jasně patrný z obrázku 6.5, jež zobrazuje propojení komponent pomocí OpenWire. Ekvalizér se skládá z dolní propusti (f_{mez} = 400Hz), pásmové propusti (f_d = 700Hz, f_h = 5500Hz) a horní propusti (f_{mez} = 6000Hz). Horní a dolní propust mají 41 koeficientů. Pásmová propust má z důvodu zachování dostatečné strmosti frekvenční charakteristiky koeficientů 83. Všechny filtry váhovány Hammingovým oknem. Jednotlivá pásma jsou ovládána pomocí zesílení jednotlivých kanálů na Audio Mixeru. Všechny použité komponenty jsou součástí balíčku AudioLab.



Obr. 6.3 Frekvenční charakteristika pásmové propusti

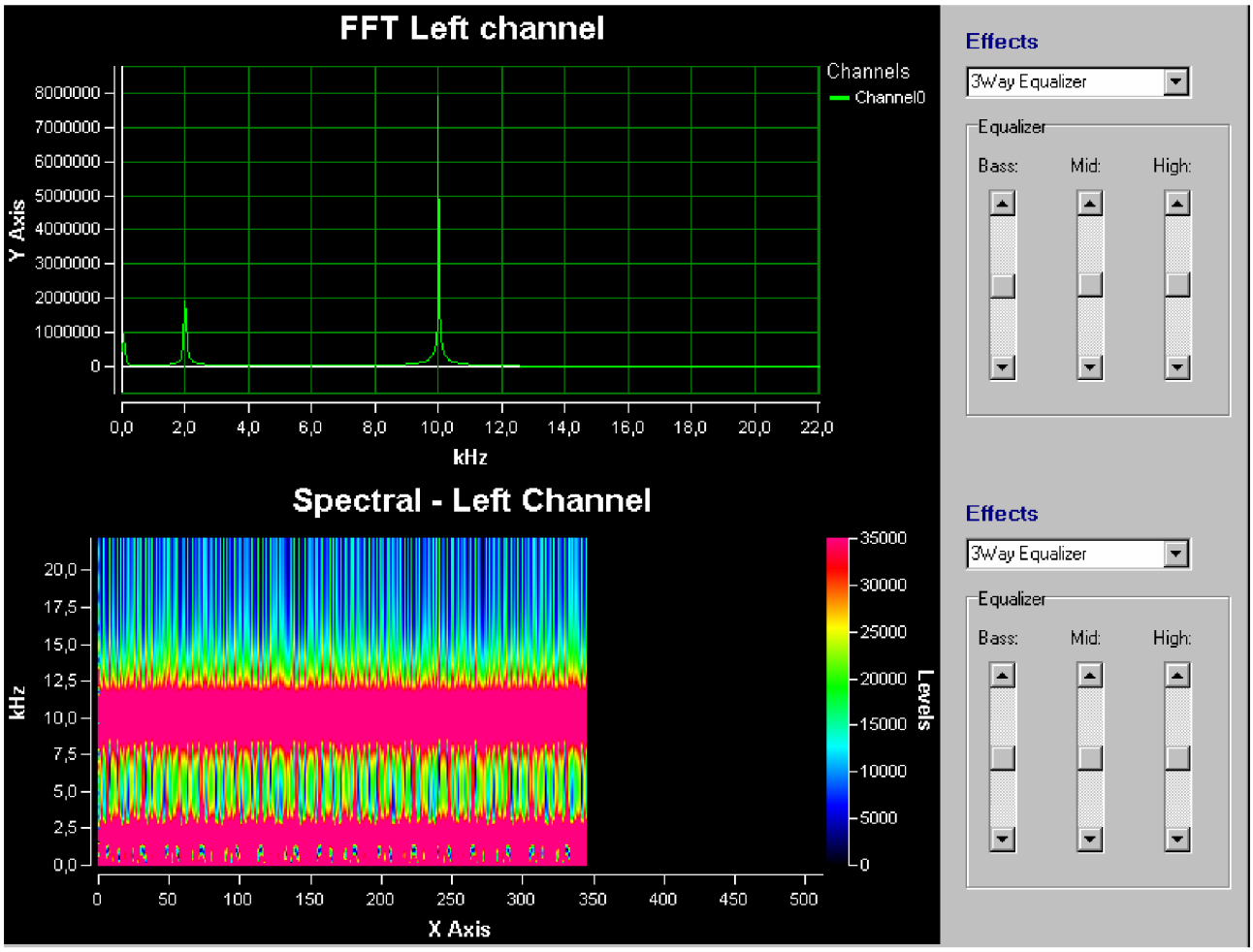


Obr. 6.4 Fázová charakteristika pásmové propusti (nahoře) a zobrazení koeficientů (dole)

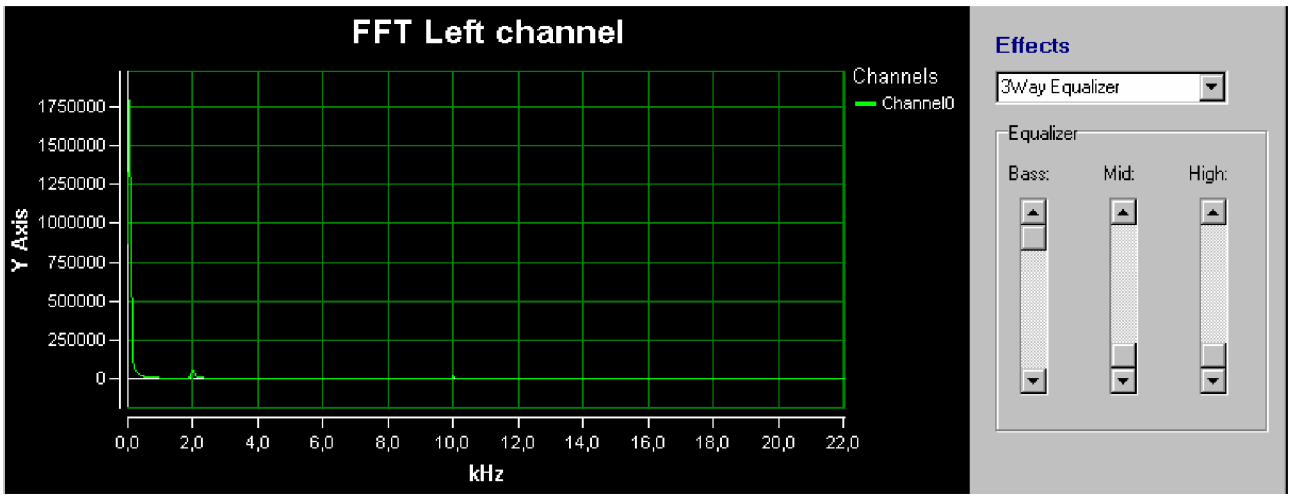


Obr. 6.5 Schématické zapojení grafického ekvalizéru pomocí OpenWire

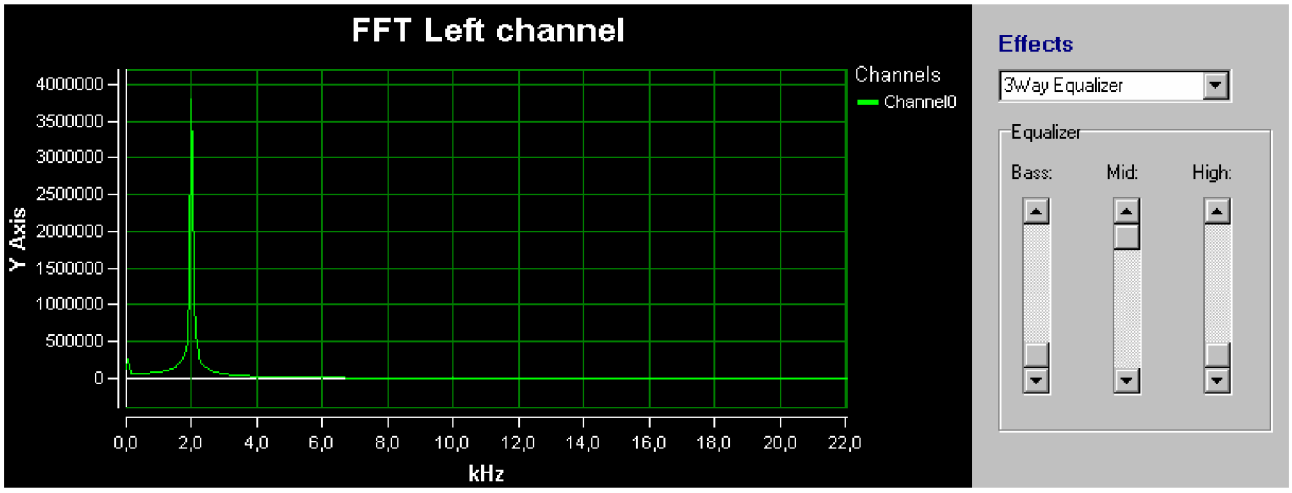
Správná funkce ekvalizéru byla ověřena pomocí testovacího signálu složeného ze tří frekvenčních složek. Směs tvoří signály s frekvencemi 60Hz, 2kHz a 10kHz. Na obrázku 6.6 je tato směs zobrazena ve frekvenční oblasti a pomocí časově frekvenční analýzy. Na obrázcích 6.7, 6.8 a 6.9 jsou zobrazena spektra testovaného signálu po průchodu ekvalizérem. V pravé části obrázků je vidět nastavení ekvalizéru.



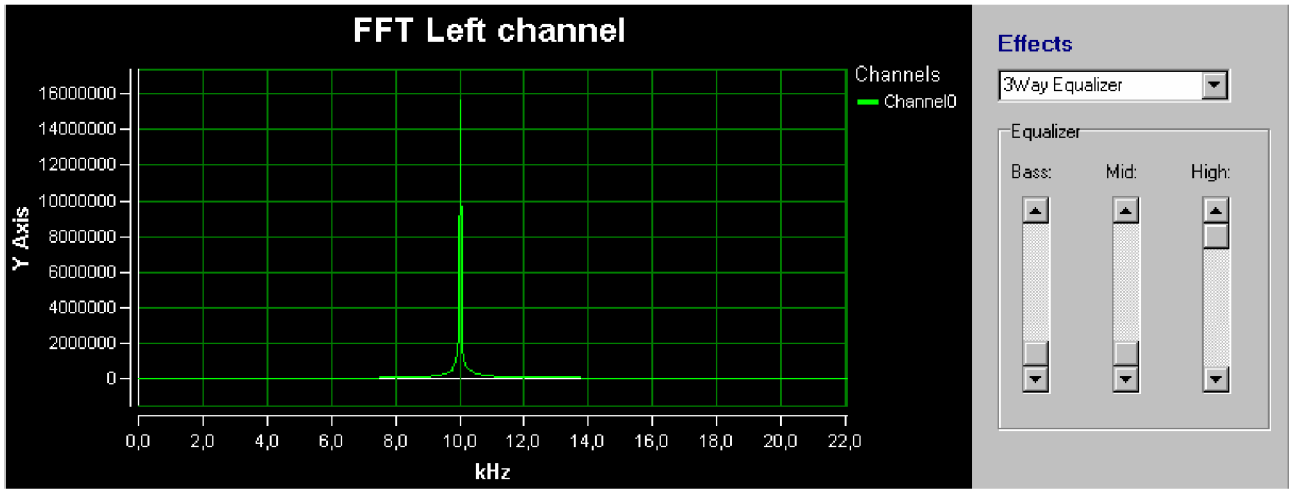
Obr. 6.6 původní spektrum a spektrogram testovacího signálu



Obr. 6.7 spektrum testovacího signálu po průchodu ekvalizérem – zdůraznění basů



Obr. 6.8 Spektrum testovacího signálu po průchodu ekvalizérem – zdůraznění středů



Obr. 6.9 Spektrum testovacího signálu po průchodu ekvalizérem – zdůraznění výšek

6.3 Zvukové efekty

Systémy pro úpravu zvuku lze rozdělit na dvě skupiny: signálové a efektové procesory. U prvých nelze měnit poměr mezi signálem upraveným (WET) a neupraveným (DRY). Patří sem například dynamické procesory, různá zařízení pro potlačení šumu apod. Do druhé skupiny pak patří většina zvukových efektů (echo, chorus, revers, delay, flanger, phaser,…), kde nastavujeme poměr mezi WET a DRY. Digitálních efektů existuje v dnešní době celá řada. Zvukový signál lze upravovat v časové, frekvenční a dynamické oblasti. Efekty jsou zaměřeny na různé úpravy v těchto souřadnicích jež se mohou zaměřovat pouze na jednu oblast nebo i na jejich kombinace. Tím vzniká v dnešní době několik stovek různých efektů využívaných jak pro korekci zvukového signálu, tak pro upravení uměleckého dojmu zvuku.

6.3.1 Delay

Tento „efekt“ vlastně způsobuje konstantní časové (fázové) zpoždění. S jeho pomocí se simuluje vzdálenost zdroje zvuku, popř. se dorovnávají časové rozdíly vzniklé v průběhu zpracování apod.. Fázové zpoždění se obvykle nastavuje v milisekundách, naproti tomu bývá u harmonických (sinusových) signálů fázové zpoždění charakterizováno úhlem (0° - 360°; 0 - - 2π rad) a pak je nutný přepočet na čas pomocí základní frekvence.

Další význam změny zpoždění je patrný např. při poslechu pomocí sluchátek, kdy bývá vhodnější simulovat stereofonní poslech především pomocí různého fázového zpoždění signálů L (left) a R (right), neboť lidské ucho je schopné na základě tohoto rozdílu (který činí maximálně vzdálenost uší v metrech dělenou 340 m/s) odhadnout polohu zdroje zvuku.

Tento efekt je jedním z velké skupiny efektů založené na využití zpožďovací linky. Vstupní signál je v přímé větvi násobený konstantou a_B (*blend*) a sečtený se signálem zpožděným o M vzorků násobeným konstantou a_{FF} (*feedforward*). Z výstupu zpožďovací linky je na její vstup opět přiváděn zpětnovazební signál násobený konstantou a_{FB} (*feedback*). Pro výstupní $y(n)$ signál platí

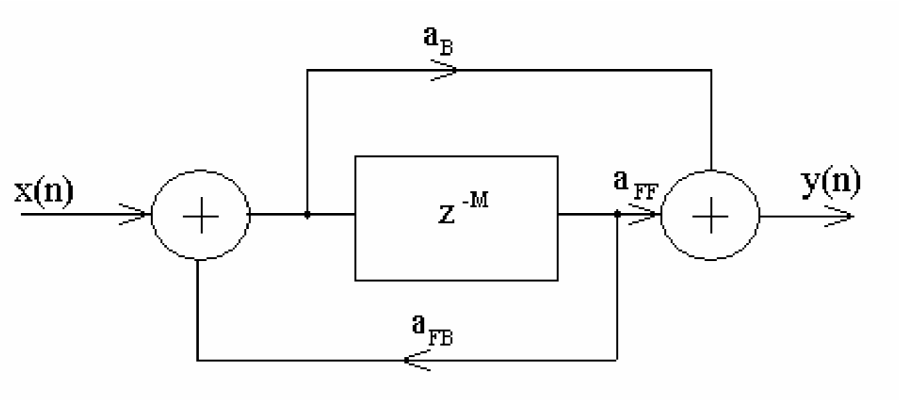
$$y(n) = a_B x(n) + (a_B a_{FB} + a_{FF}) v(n-M) \tag{6.4}$$

$$v(n) = x(n) + a_{FB} v(n-M) \tag{6.5}$$

kde $x(n)$ je vstupní signál a $v(n)$ je stavová proměnná. Přenosová funkce $H(z)$ systému je

$$H(z) = \frac{a_B\left(a_B a_{FB} + a_{FF} - a_{FB}\right) z^{-M}}{1 - a_{FB} z^{-M}}. \tag{6.6}$$

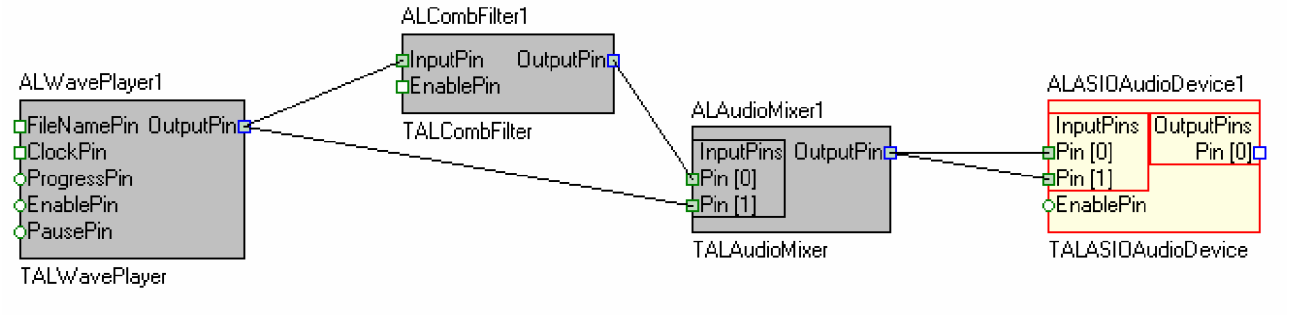
Jedná se o speciální kmitočtový filtr řádu m s koeficienty a_1 až a_{M-1} a b_1 až b_{M-1} rovnými nule. Tomuto filtru se říká *univerzální hřebenový filtr* a je zobrazen na obrázku 6.10.



Obr. 6.10 Univerzální hřebenový filtr

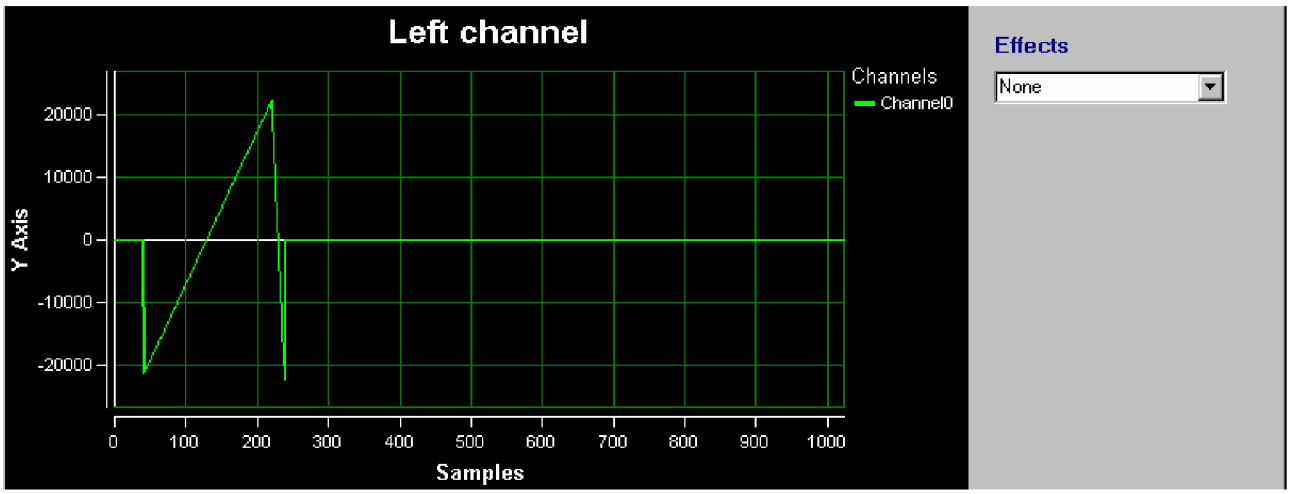
Hudební efekty tohoto druhu lze rozdělit podle zpoždění zpožďovací linky. Hlavním zdrojem informací byla literatura [8], podle níž byl vytvořen efekt zahrnující současně efekty *resonator* (zpoždění menší než 25ms), *slapback* (zpoždění mezi 25 až 50ms) a *delay* (zpoždění vetší než 50ms). V aplikaci je tento efekt nazván souhrnně delay a rozsah nastavení časového zpoždění je od 1 do 3500ms s nastavitelnou intenzitou (decay) do jeden a půl násobku vstupního signálu. Tyto parametry byly zvoleny záměrně pro možnost srovnání s efektem delay na mixážním pultu Pioneer DJM-600. Při návrhu je potřeba pamatovat na fakt, že hodnota delay CombFilter je udávána v počtu vzorků a je třeba ji přepočíst na čas.

Realizace efektu je založena na komponentě CombFilter, tedy na komponentě AudioLab představující univerzální hřebenový filtr. Zapojení v OpenWire je na obrázku 6.11.

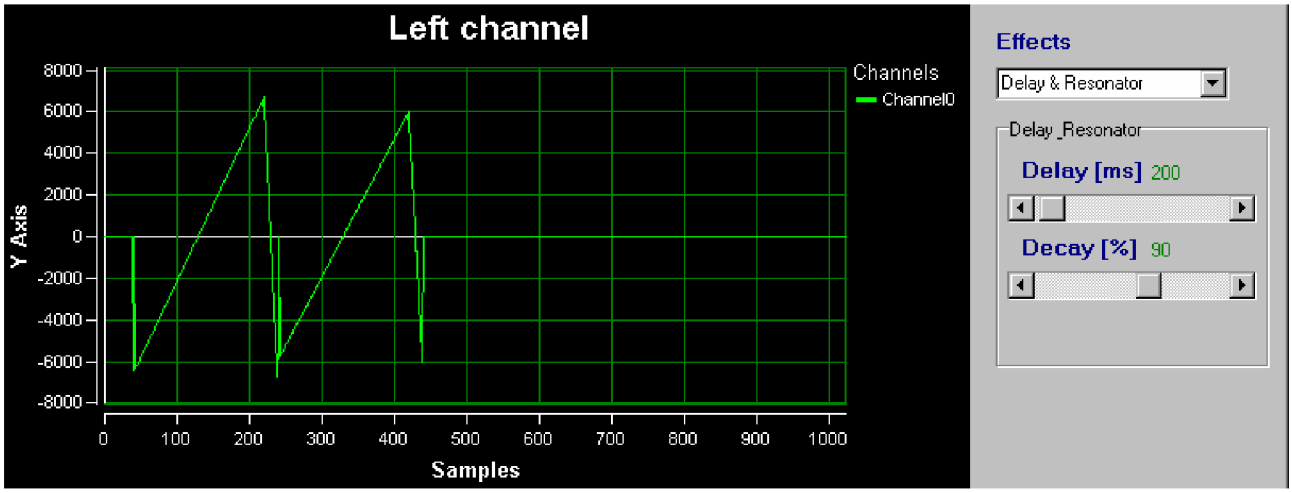


Obr. 6.11 Schématické zapojení efektu delay pomocí OpenWire

Pro otestování funkce efektu delay byl použit signál zobrazený na obrázku 6.12. Další obrázek (6.13) ukazuje změnu signálu při nastavení delay na 200ms a decay 90%. První, větší průběh zobrazuje signál originální a druhý, nižší, příspěvek efektu.



Obr. 6.12 Testovací signál pro efekt delay



Obr. 6.13 Změna testovacího signálu po použití efektu echo

6.3.2 Echo

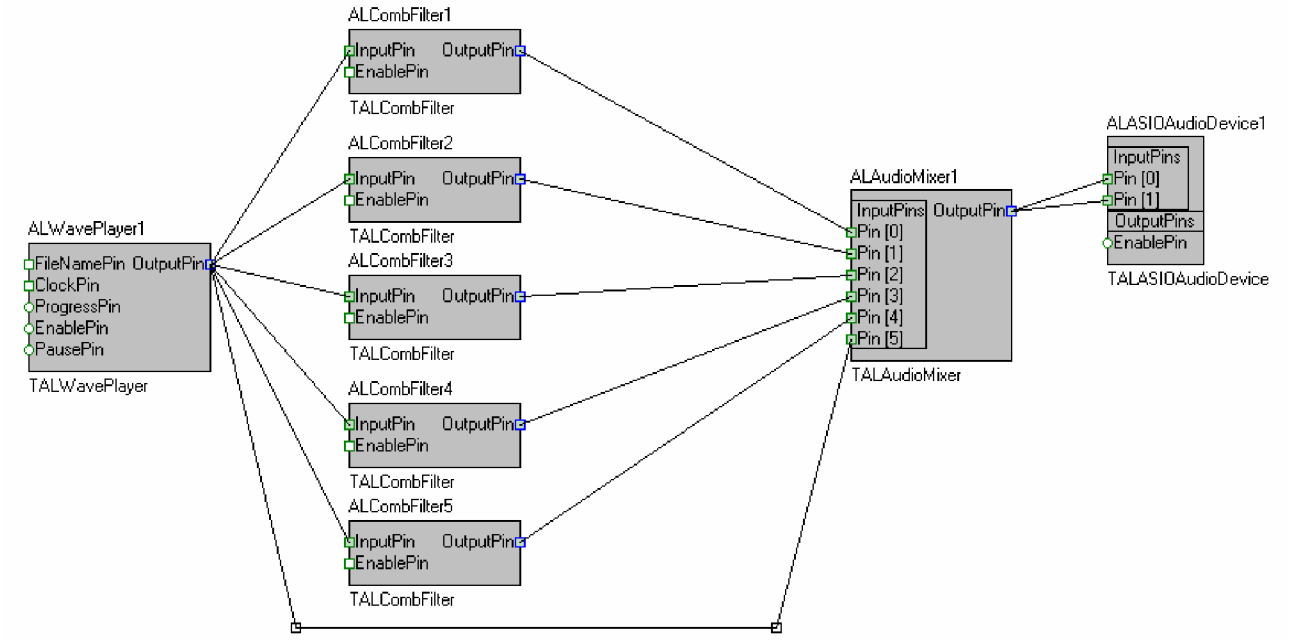
Echo pracuje tím způsobem, že původní signál zpozdí o nastavenou dobu (např. t = 0,1s) a přičte jej k původnímu signálu. Pak jej zpozdí o 2t (2x0,1s) a opět jej přičte a tak dále. Navíc mívá zpožděný signál nižší úroveň (záleží na nastavení). Princip je velmi dobře patrný z obrázků 6.15 a 6.16.

Echo vlastně simuluje jev zvaný „ozvěna“. Ten vznikne, stojíme-li např. na jinak volném prostranství před rozměrnou překážkou, která je vzdálena nejméně 17m. Zvuk, který vydáme, se od této překážky odrazí a vrátí se nám za 0,1s [(17m x 2)/(340 m/s)]. Tehdy již lidskému uchu přestávají oba zvuky splývat a vnímá je proto odděleně. Echo simuluje pouze velmi specifický případ, který se ve skutečném světě prakticky nevyskytuje a nemá se skutečným dozvukem příliš společného.

U tohoto efektu se nastavují tyto základní parametry:

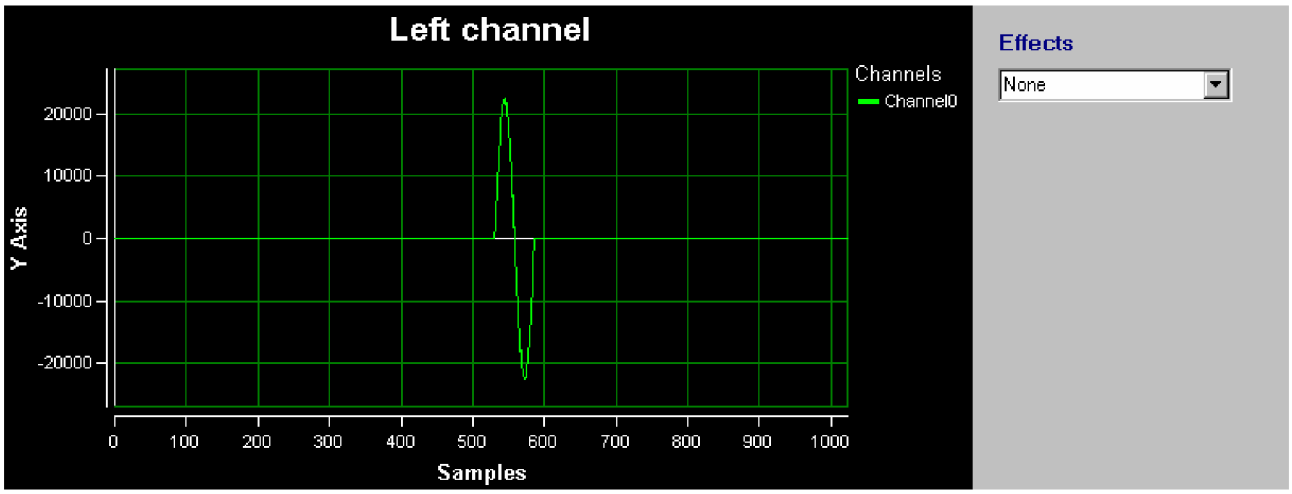
- doba zpoždění (delay)
- relativní „úroveň signálu z echa“ resp. úroveň 1. zpožděného signálu
- poměr amplitud signálu zpožděného o kt a (k+1)t - tzv. „Decay“. Pokud bychom nastavili Decay rovný 1, pak by všechny zpožděném signály měly stejnou úroveň a impulzy na by byly stejně vysoké jako 2. impuls.

Realizace echa je obdobná realizaci efektu delay. Je ovšem použito několik hřebenových filtrů sloužících jako zpožďovací linky. V programu je možno nastavit dobu 1. zpoždění a následující jsou vždy násobkem. Dále je možno nastavit relativní úroveň 1. zpožděného signálu a další se postupně snižují o 20%. Zpožďovacích linek je použito pět. Na obrázku 6.14 je zapojení efektu pomocí OpenWire.

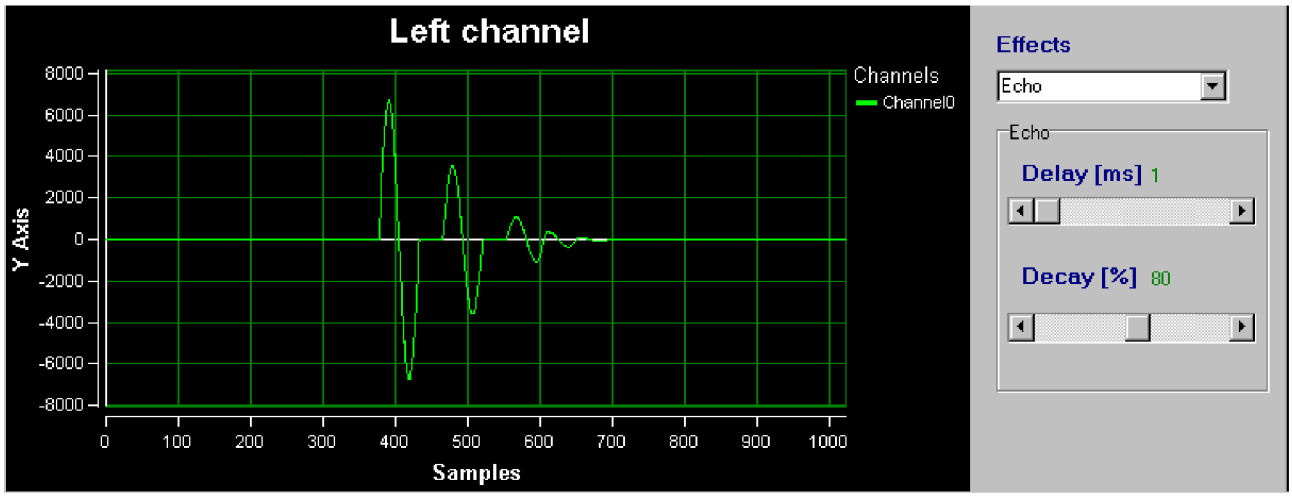


Obr. 6.14 Schématické zapojení efektu echo pomocí OpenWire

Pro otestování efektu byl zvolen signál obsahující pouze jednu periodu funkce sinus (obrázek 6.15). Celý princip dobře ilustruje obrázek 6.16)



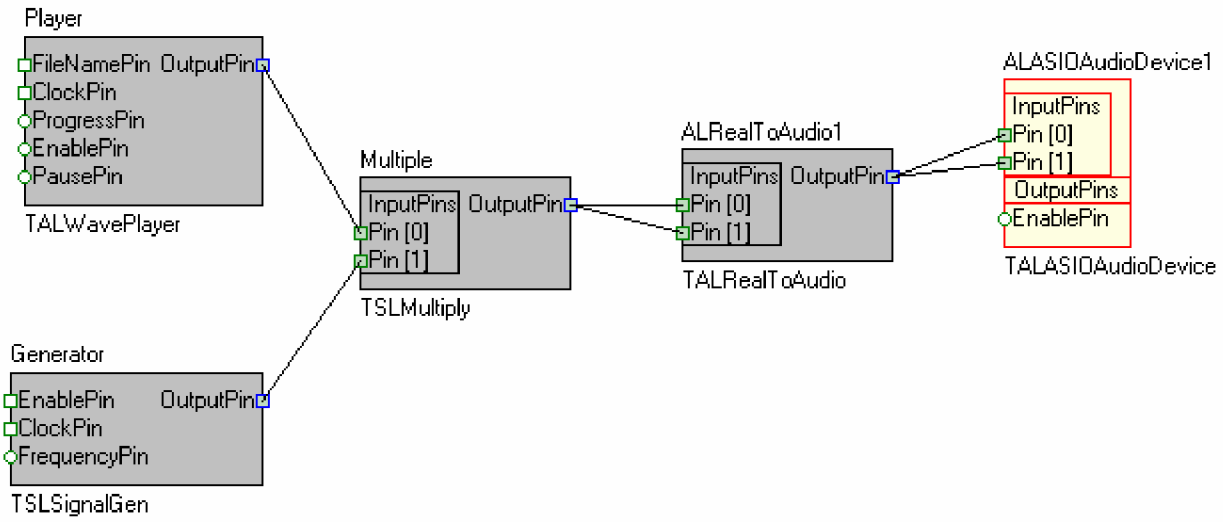
Obr. 6.15 Testovací signál pro efekt echo



Obr. 6.16 Výstupní signál při použití efektu echo

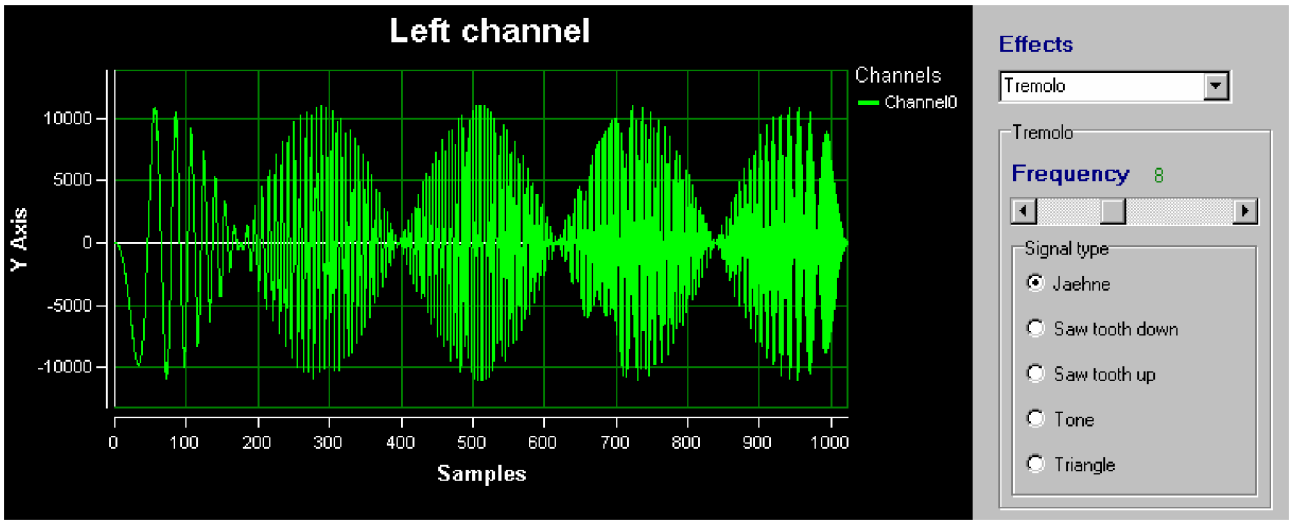
6.3.3 Tremolo

Jde o jeden z nejstarších efektů jež vznikl pro vylepšení zvuku elektrické kytary. Je v podstatě založen na změnách hlasitosti signálu. Tremolo provádí modulaci amplitudové obálky hudebního signálu sub-akustickým oscilátorem s kmitočtem řádově od 0,1 do 10 Hz. Kmitočet oscilátoru určuje rychlost a amplituda hloubku modulace. Zvuk prohnaný tremolem není nijak modulován. Houpe se a vlní, ale nemění se ani jeho charakter, ani výška. Lze jej použít na varhany, elektrickou kytaru nebo na výsledný zvuk nahrávky. Obecně zní dobře spíše na rovných než na perkusivních zvucích. Zapojení efektu pomocí OpenWire je na obrázku 6.17.

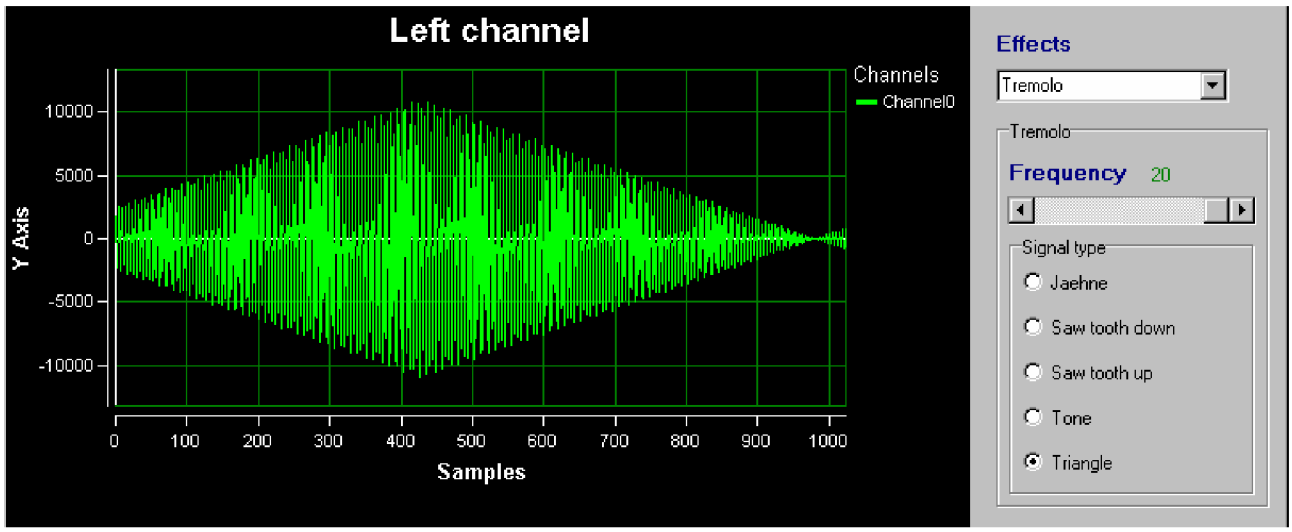


Obr. 6.17 Zapojení efektu tremolo pomocí OpenWire

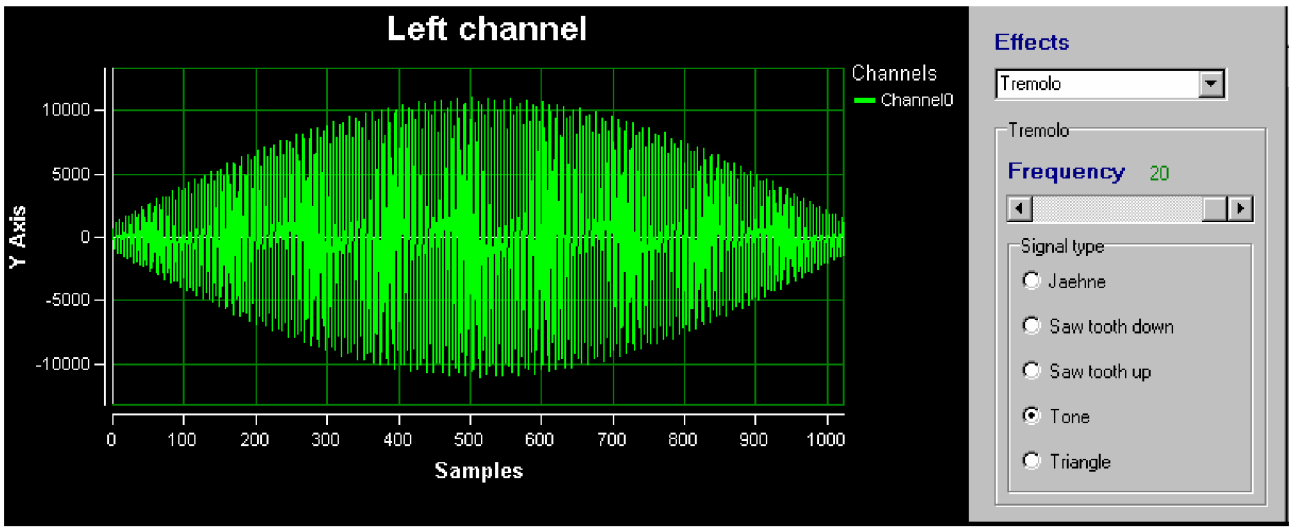
Pro realizaci efektu je třeba zvukový signál vynásobit signálem sub-akustického generátoru (SLSignalGen – Signal Lab), k tomu slouží komponenta TSLMultiply (Signal Lab). Tím vznikne datový signál jež je třeba převézt zpět na signál zvukový. K tomu slouží komponenta TALRealToAudio. Různé modifikace efektu lze získat změnou tvaru výstupního signálu generátoru. Program umožňuje nastavit jeden z těchto průběhů: jehlový, pilovitý směrem vzhůru, pilovitý směrem dolů, sinusový a trojúhelníkový. Dále je možné nastavit frekvenci generátoru v rozsahu od 1Hz do 20Hz. Působení efektu na sinusový signál je na obrazci 6.18, 6.19, 6.20 a 6.21.



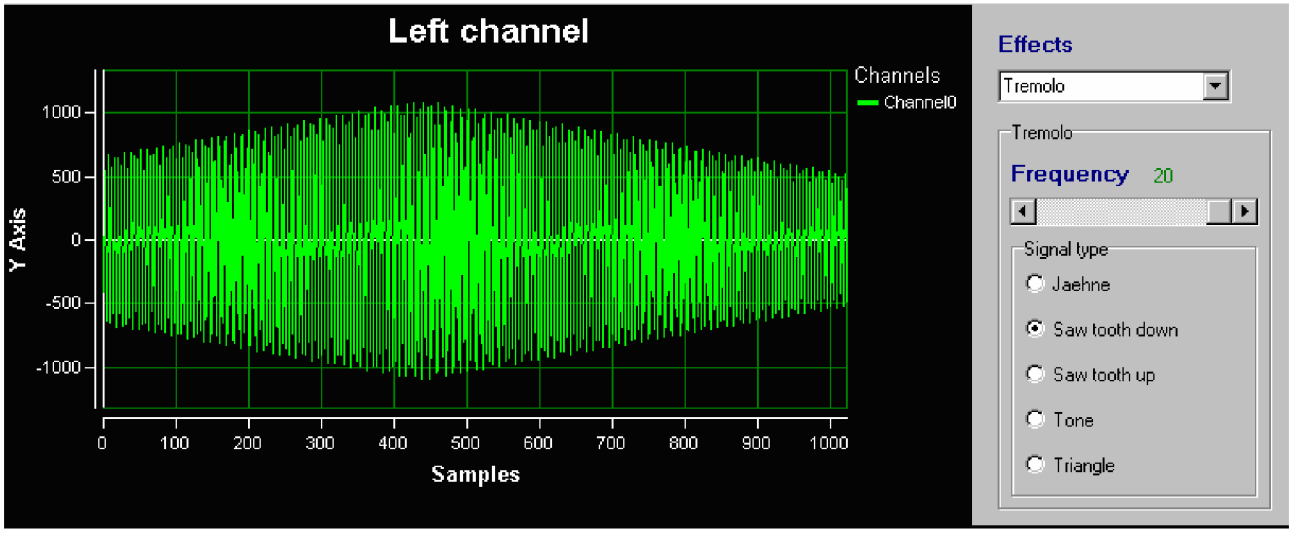
Obr. 6.18 Výstup efektu tremolo po modulaci 1kHz signálu jehlovitým průběhem



Obr. 6.19 Výstup efektu tremolo po modulaci 1kHz signálu trojúhelníkovým průběhem



Obr. 6.20 Výstup efektu tremolo po modulaci 1kHz signálu sinusovým průběhem

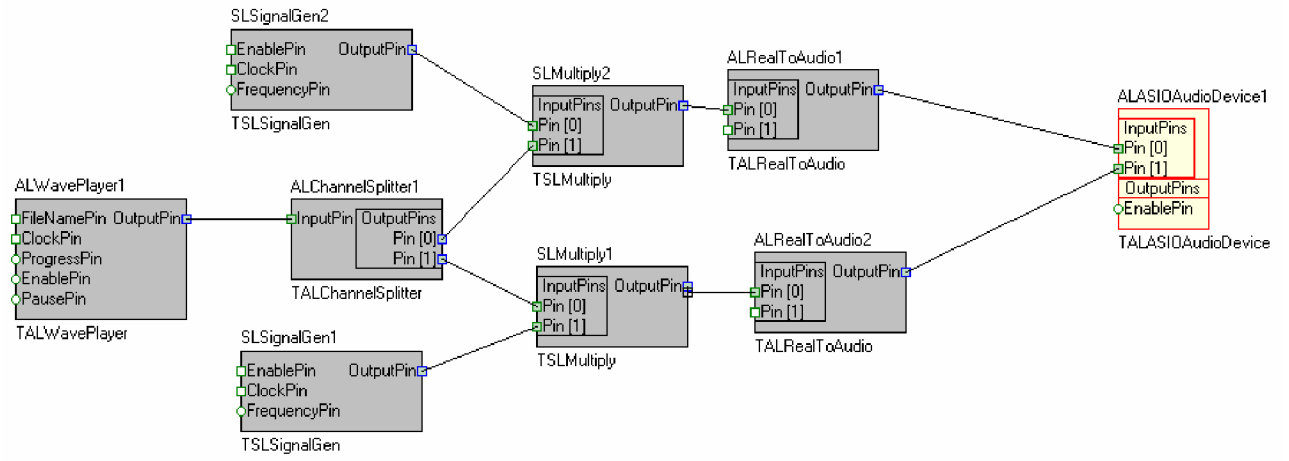


Obr. 6.21 Výstupní signál efektu tremolo po modulaci 1kHz signálu pilovitým průběhem

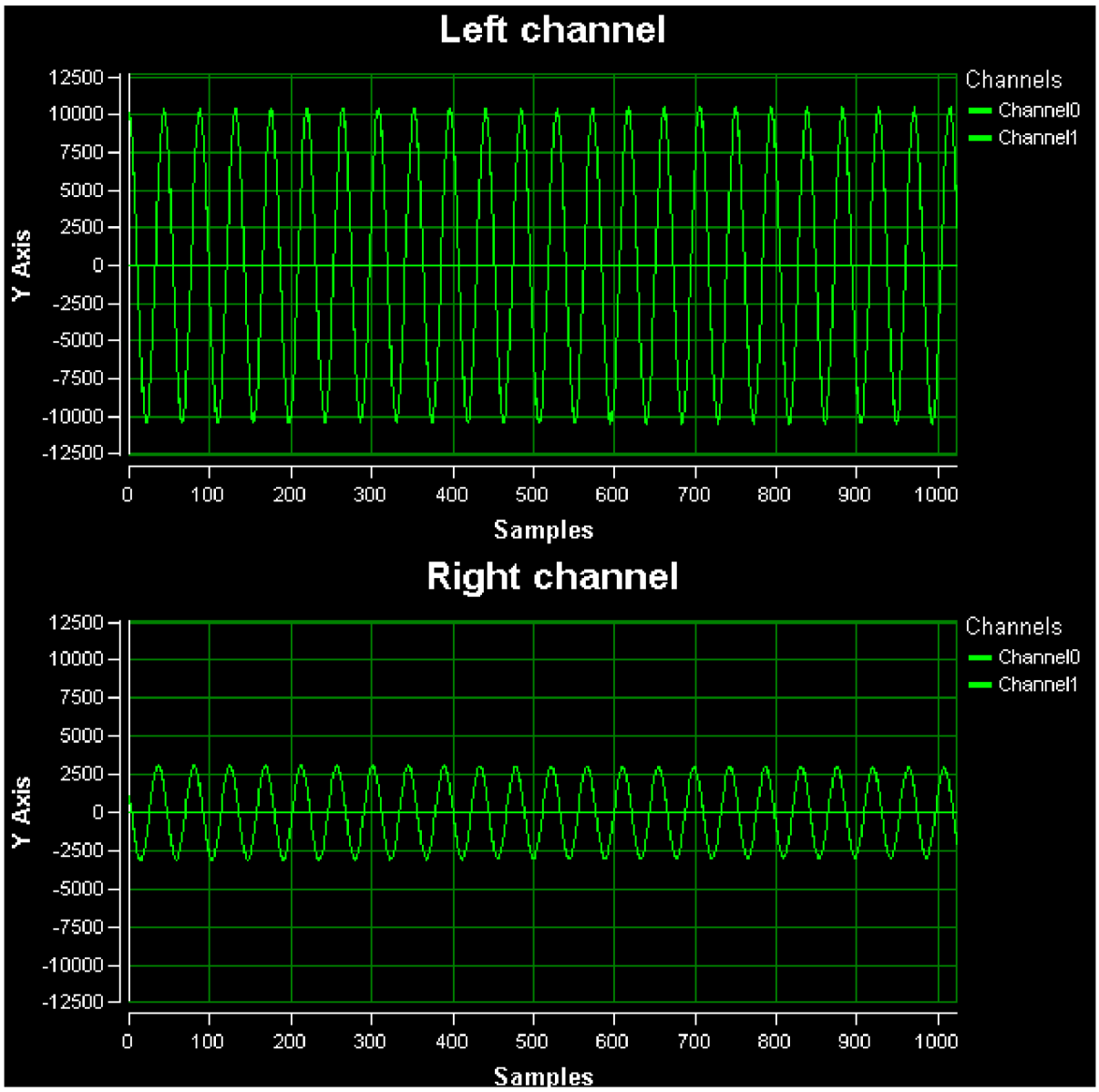
6.3.4 Panner

Pannery využívají binaurálního slyšení k vytvoření prostorového vjemu, původně tedy rozdělení signálu do dvou kanálů tak, aby výsledná subjektivní lokalizace zdánlivého zdroje signálu odpovídala představám autorů. Tyto efekty jsou založeny na intenzitní stereofonii. Jde o nejjednodušší způsob využívající toho, že z té strany, odkud zvuk přichází, je silnější zvukový vjem. Nebere v úvahu fakt, že frekvence pod cca 350Hz člověk vnímá stejně hlasitě z obou stran. Vyšší frekvence se vzhledem k velikosti lidské hlavy nemohou kolem ní „ohnout“ a hlava je tak akusticky odclání. Tento přechod je samozřejmě postupný. Další frekvenční clonou jsou též boltce uší, ovšem ty odclání zase podstatně vyšší kmitočty. Ucho je také schopno rozlišovat směr na základě vzájemného časového zpoždění vjemu z obou uší.

Efekt panner je v podstatě stereo tremolo, které přelévá zvuk z jedné strany na druhou. Samozřejmě správně funguje pouze při stereo zvuku. Jeden z generátorů musí mít posunutou fázi o 180° oproti druhému. Zapojení panneru je na obrázku 6.22 a obrázek 6.23 znázorňuje funkci při běhu efektu.



Obr. 6.22 Zapojení efektu panner pomocí OpenWire

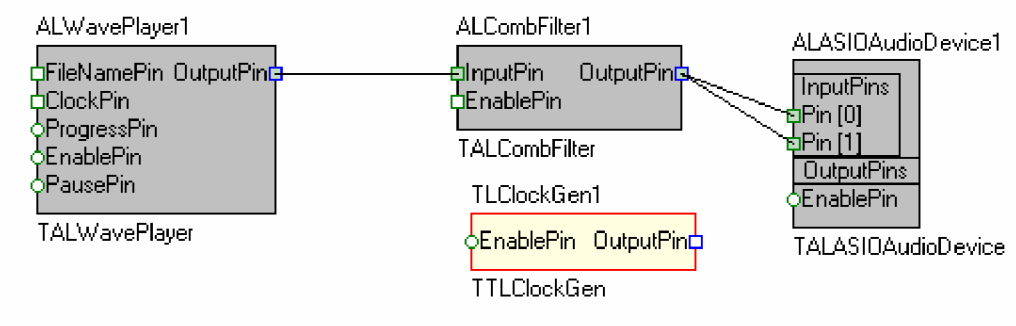


Obr. 6.23 Zobrazení průběhu efektu panner

6.3.5 Vibrato

Podstatou efektu je Dopplerův jev, jenž popisuje změnu frekvence a vlnové délky přijímaného signálu oproti vysílanému, způsobenou nenulovou vzájemnou rychlostí vysílače a přijímače. Poctivou realizací efektu by tedy byl například rotující reproduktor. Jde o hudební efekt se zpožďovací linkou s proměnným zpožděním, jež je řízeno sub-akustickým oscilátorem. Kmitočet určuje rychlost a amplituda hloubku modulace. Opět je využit univerzální hřebenový filtr (*obrázek 6.10)* tak jako například při užití efektu delay. Vibráto vznikne při zesílení přímé větve $a_B = 0$, zesílení zpětnovazební větve $a_{FB} = 0$, zesílení zpožděného signálu $a_{FF} = 1$ a hloubce modulace 0 až 3 ms.

Realizace v c++ Builderu pomocí OpenWire je na obrázku 6.24. K řízení je použita funkce sinus jejíž hodnoty jsou generovány v každém taktu časovače. Amplituda určuje hloubku modulace a frekvence časovače rychlost efektu. Je třeba dbát na to, aby hodnoty řídící hřebenový filtr nenabývaly záporných hodnot. Řízení efektu není příliš dokonalé, bohužel způsob jak řídit zpoždění hřebenového filtru naprosto plynule nebyl nalezen. Test funkčnosti tohoto efektu lze je vhodné provézt poslechem.



Obr. 6.24 Zapojení efektu vibrato pomocí OpenWire

Další efekty pracující na stejném principu, jež by vznikly drobnou úpravou, jsou například chorus a flanger.

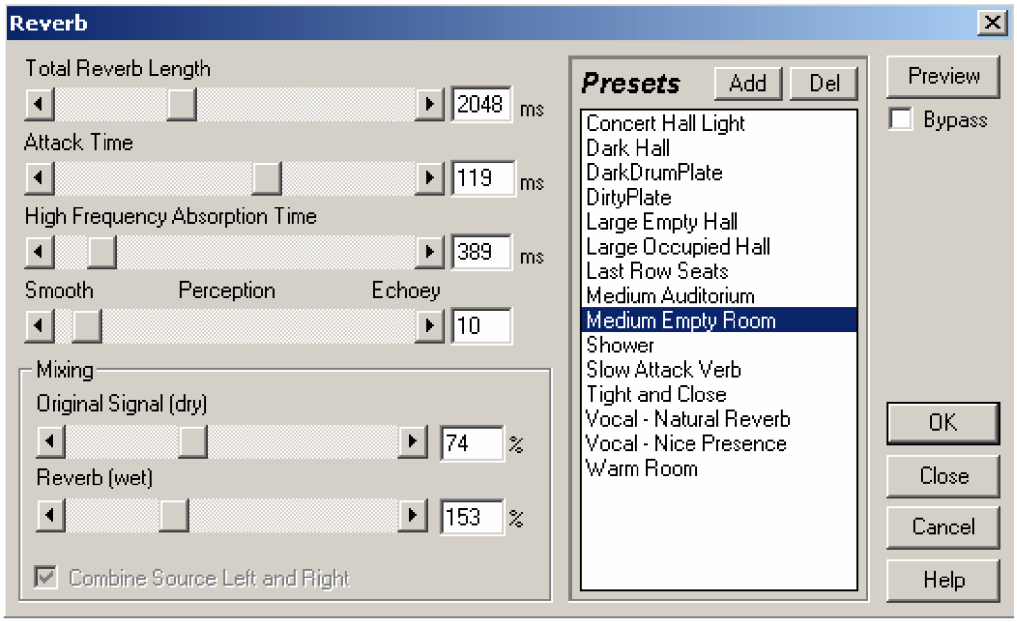
6.3.6 Reverb (dozvuk)

Samotný dozvuk jakožto akustický jev je velmi komplikovaný a matematicky obtížně popsatelný. Předpokladem vzniku dozvuku je uzavřený prostor, ve kterém může vzniknout stojaté vlnění. Pokud do tohoto prostoru umístíme zdroj zvuku (všesměrový), bude se zvuk v kterémkoliv místě v prostoru skládat ze zvuku přímého (zvuk přímo ze zdroje bez jediného odrazu) a ze zvuků, které prošly na cestě k danému místu jedním a více odrazy. Je jasné, že „odražený“ zvuk je jednak zpožděný za zvukem přímým (delší dráha), a navíc má menší intenzitu a jiné spektrální složení (vlivem odrazových ploch). I když by se mohlo zdát, že tento zvuk bude „maskován“ zvukem přímým (má přeci menší intenzitu), skutečnost je opačná. Jde o to, že energie soustředěná v „přímém“ zvuku je podstatně menší, než energie vyzářená do zbylého prostoru, jejíž nezanedbatelná část „doputuje“ až k nám (záleží také na směrovosti reproduktorů, která je markantní hlavně u vyšších kmitočtů). Ve výsledku se tedy dá říci, že pokud nesedíme přímo u zdroje zvuku, tak zvuk, který slyšíme, se skládá především ze zvuku odraženého.

Pokud vypneme zdroj zvuku, tak po chvíli ustane i přímý zvuk. Jenomže v prostoru se pohybuje ještě spousta zvukových vln, kterým ještě nějakou chvíli trvá, než se přemění v teplo na odrazových plochách, a část z nich také projde „poslechovým místem“. Zvuk „doznívá“, tj. v optimálním případě exponenciálně klesá intenzita zvuku v místnosti (i v referenčním bodě). Doba od vypnutí zdroje zvuku, za kterou klesne akustický tlak v místnosti na jednu miliontinu původní hodnoty (-60dB) se nazývá **doba dozvuku**.

Dozvuk je ve velké míře závislý i na tvaru a odrazivosti materiálu od kterého se zvuk odráží. Matematický popis jež zahrnuje všechny potřebné parametry lze nalézt v literatuře [9], kde jsou uvedeny rovnice P.E.Sabina, Eyringa a Millingtona, jež se podrobně tímto jevem zabívají.

Echo přidává do zvuku jen „odrazy“ s konstantním zpožděním, spektrálně shodné s původním zvukem, kdežto dozvuk přidává „nepřímý“ zvuk skládající se z velkého množství různě zpožděných zvukových vln. Těch je s přibývajícím časem stále více tak, jak k posluchači přicházejí postupně další a další vlny, které putovaly po delší dráze. Jejich intenzita pak postupně klesá. Navíc je nepřímý zvuk výrazně ovlivněn frekvenční závislostí odrazivosti materiálů - např. měkké, porézní a textilní povrchy výrazně tlumí vysoké kmitočty, naopak utlumit frekvence řádu stovek Hz již bývá problematické. A tak bývá obvykle nejvyšší uvažovaná frekvence obsažená v dozvuku do 10kHz.



Obr. 6.25 Nastavení efekt reverb v programu CoolEdit

Dozvukové jednotky (ať vědomě, či nevědomě) se používají především k simulaci poslechového prostoru (koncertní síň, náměstí, pokoj, tělocvična, kostel, ….). A tak je dobré mít na zřeteli, jakého efektu chceme dosáhnout.

Základním parametrem je již uvedená doba dozvuku. Typická doba dozvuku pro běžně zařízený pokoj je 0,5 ± 0,1s, vícekanálový kinosál - do 0,7s atd. Dále nastavení poměru původního signálu a „dozvukového signálu“ resp. nepřímého zvuku. Tím se simuluje vzdálenost od zdroje signálu (čím menší zastoupení původního/přímého signálu, tím jsme dále). Třetím důležitým parametrem je „doba náběhu“ (attack), která představuje prodlevu mezi přímým a odraženým zvukem (vzniklým dozvukem). Pokud budeme například v prázdné tělocvičně, tak tím více, čím my a reproduktor budeme vzdáleni od stěn, tím bude delší prodleva mezi příchodem přímého a odraženého zvuku a tedy i „attack time“. Dalším parametrem bývá úprava spektrálního složení odraženého zvuku. Čím více je v místnosti měkčích a poréznějších materiálů, tím menší bude obsah vyšších kmitočtů. V mramorové koupelně to samozřejmě bude jinak. Názorný příklad ovládacího prvku dozvuku je na 6.25.

Vzhledem ke složitosti algoritmu tohoto efektu a ke ztrátě smysluplnosti bez tvorby přednastavených akustických prostorů bylo od původního záměru upuštěno a efekt ve výsledné aplikaci není zařazen.

6.4 Spektrální analýza

Jde o rozklad signálu na jeho jednotlivé základní frekvenční složky za účelem nalezení popisu signálu. Setkáváme se v zásadě se dvěma případy: buď se jedná o analýzu jednotlivého konkrétního signálu, který si přejeme popsat ve spektrální oblasti zpravidla tak podrobně, aby jej bylo možno na základě získaného spektra úplně rekonstruovat, nebo jde o popis celé třídy signálů, které se ovšem vzájemně liší a úkolem analýzy pak je nalézt společné spektrální rysy, umožňující celou třídu jistým způsobem charakterizovat ve frekvenční oblasti. Koncepty spektrální analýzy v obou uvedených případech se významně liší.

Při analýze konkrétního signálu, popsaného deterministickou funkcí času, jde o to, nalézt co nejpřesnější koeficienty Fourierovy řady nebo vzorkyFourierovy transformace, obojí komplexní, tedy včetně fázové informace tak, aby popis umožňoval jeho zpětnou rekonstrukci. Integrální Fourierova transformace popisuje signál harmonickými složkami neměnných parametrů na oboustranně nekonečném časovém intervalu. U signálů jež se v čase mění, je vhodné použít k analýze tzv. krátkodobých spekter, umožňujících sledovat vývoj frekvenčního obsahu signálů v čase. Spektrální analýzu lze založit i na jiných transformacích, např. kosinové, Walshově, Haarově apod.

Konečný úsek N vzorků diskrétního signálu $\{f_n = f(nT)\}$, $n = 0,1,\ldots,N\text{-}1I,$ může být vyjádřen v kvazispojité reprezentaci jako součet vážených posunutých Diracových distribucí

$$f_s(t) = \sum_{n=0}^{N-1} f_n \delta(t - nT) \tag{6.7}$$

a jeho spektrum ve smyslu integrální Fourierovy transformace tedy je

$$F_s(\omega) = \mathbf{F}\left\{\sum_{n=0}^{N-1} f_n \delta(t-nT)\right\} = \sum_{n=0}^{N-1} f_n \mathbf{F}\{\delta(t-nT)\} = \sum_{n=0}^{N-1} f_n e^{-j\omega nT} = \mathbf{DTFT}\{f_n\} \tag{6.8}$$

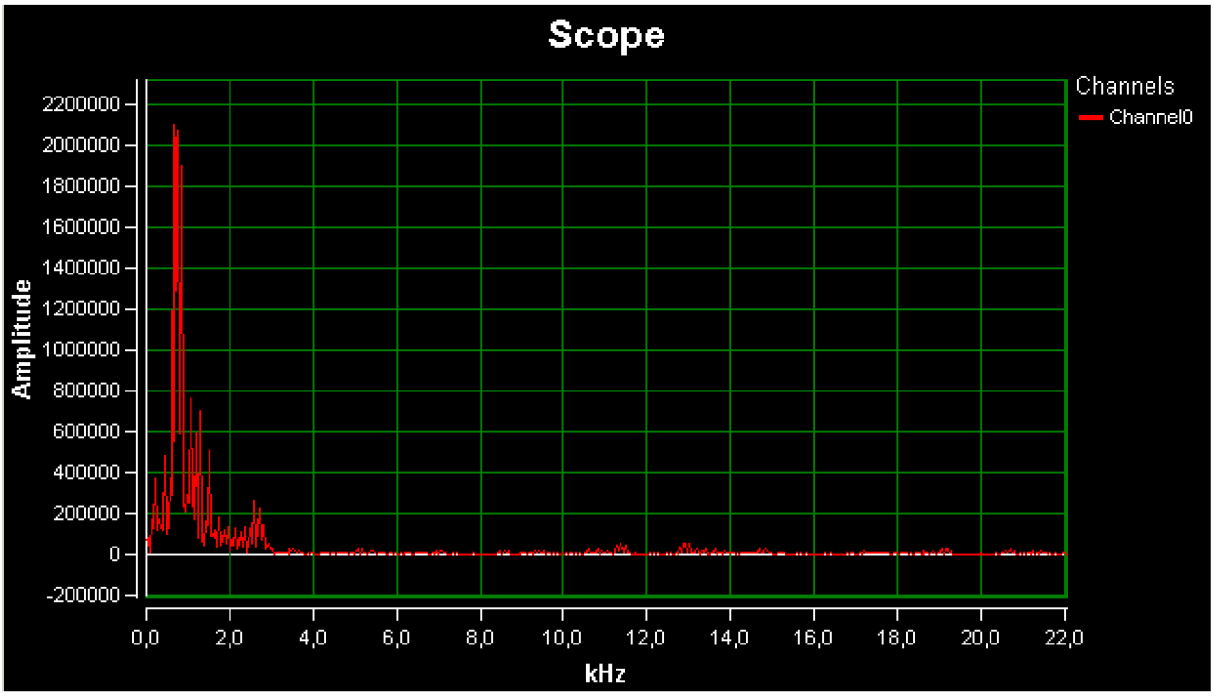
Nevzorkujeme-li toto spojité spektrum, které je periodické s periodou $2\pi/T$, N vzorky v jedné periodě, dostaneme

$$\{F_k\} = \{F_s(k\Omega)\} = \left\{\sum_{n=0}^{N-1} f_n e^{-jk\Omega nT}\right\} = \mathbf{DFT}\{f_n\}. \tag{6.9.}$$

Diskrétní Fourierova transformace poskytuje přesné vzorky spektra konečného a nevzorkovaného úseku analyzovaného signálu. Důsledkem diskretizace analyzovaného signálu je snížení rozlišovací schopnosti ve spektru. K tomu dochází díky omezení délky signálu způsobujícím tzv. prosakování ve spektru. Tento jev lze omezit prodloužením okna a tím zúžením hlavního laloku funkce $W(\omega)$, dále pak vhodnou volbou tvaru okna $w(n)$, ovlivňujícího následně tvar $W(\omega)$ a tím rozsah a charakter prosakování. Dále také vzniká periodizace spektra v důsledku vzorkování signálu. Přitom může dojít ke zkreslení spektra (aliasingem), pokud spektrum po předchozím kroku má významné složky nad Nyquistovým kmitočtem vzorkování. Náprava je možná zvýšením vzorkovacího kmitočtu nebo předzpracováním signálu anti-aliasingovým filtrem. Při číslicovém výpočtu spektra je nutno nevzorkovat také spektrum, což je vyjádřeno na straně spektra součinem s diskretizační posloupností impulsů. Na straně originálu následně dojde k periodizaci signálu. Při diskretizaci spektra podle definice DFT dochází k periodizace signálu s periodou rovnou délce vstupního okna, takže původně nulovým úsekům vně okna jsou formálně vnuceny opakované hodnoty signálu uvnitř

okna. Zpracovávaný úsek to ovšem nepostihuje. Vzorkování spektra však může vést k tomu, že zobrazený diskrétní výsledek nedává dobrou informaci o spektru původního signálu. Důkladný rozbor s jakými typy zkreslení spektra analyzovaných signálů je třeba počítat obsahuje literatura [1].

Fourierovu transformaci lze v prostředí c++ Builderu realizovat přímo pomocí komponenty TSLFourier z balíčku Signal Lab. Výstupem je amplitudové a fázové spektrum jež lze zobrazit pomocí komponenty SLScope (Plot Lab). V aplikačním prostředí je umožněno přepnutí zobrazení časového průběhu signálu na zobrazení amplitudového spektra. Příklad takového zobrazení je na obrázku 6.26



Obr. 6.26 Amplitudové spektrum získané pomocí komponent SLSFourier a SLScope (řeč – wav, 44100Hz, 1411kbps, 16bit)

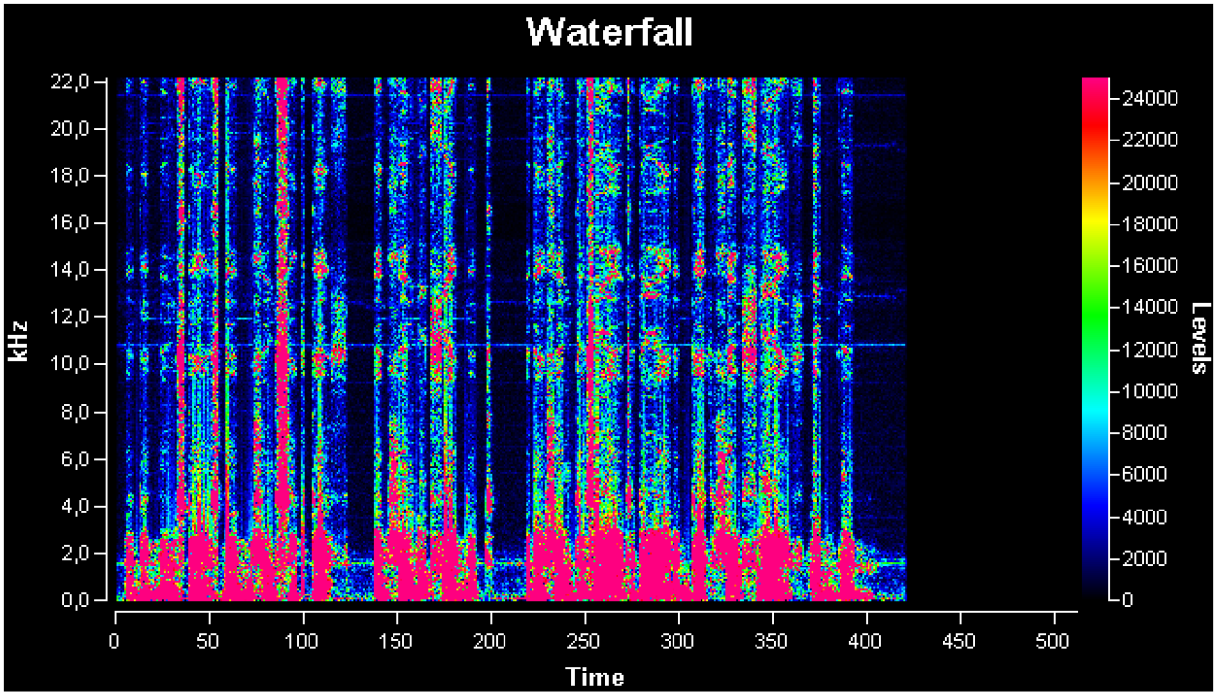
6.4.1 Časově-frekvenční analýza

Při analýze signálů přechodového charakteru (jejich charakter se v čase rychle mění) , je často účelné uvažovat o frekvenčním obsahu krátkých signálových úseků, což znamená rozvinout koncept tzv. krátkodobých spekter. Přestože integrální Fourierova transformace v teoretické podobě pracuje se signály nekonečné délky, praktická analýza vždy vychází jen z konečných úseků signálu, vymezených použitým oknem. Má-li okno má vhodnou délku a je formulováno jako klouzavé na časové ose, může tento přístup být použit pro časově-frekvenční analýzu. V tomto případě jde o co nejpřesnější lokalizaci výskytu složek signálu jak pokud jde o frekvenční, tak i časový údaj. Při této analýze je pozorovací interval určen kompromisem mezi požadavkem na dostatečnou rozlišovací schopnost ve frekvenční oblasti (rozlišitelná diference frekvencí je nepřímo úměrná délce okna) a současnou snahou o velké rozlišení také v čase (minimální rozlišitelný časový rozdíl je délce okna úměrný). Jeden z těchto požadavků bývá v konkrétní praktické aplikaci určující a délku okna (resp. příslušný počet vzorků při daném vzorkovacím kmitočtu) je pak třeba mu přizpůsobit. Použitím DFT na takto stanovené úseky signálu tak vznikají zmíněná krátkodobá spektra. Ty se zpravidla pořizují v celých sériích na základě signálových dat z delšího úseku signálu, v jehož rámci si

přejeme sledovat vývoj frekvenčního obsahu. Poněkud lze časovou rozlišovací schopnost zvýšit tím, že dílčí okna mají zvolený přesah, např. o polovinu své délky. Pak dostaneme podél časové osy přiměřeně více spekter a lze lépe sledovat případný rychlý vývoj zejména na straně vysokých kmitočtů. Takový soubor spekter, tzv. *spektrogram*, může být názorně zobrazen jako dvojrozměrný obraz, v němž jedna souřadnice odpovídá frekvenci, druhá času a barva nebo úroveň jasu odpovídá amplitudě (viz obrázek 6.27).

Délka okna musí být určena potřebnou frekvenční rozlišovací schopností na straně nízkých kmitočtů a omezuje tak časové rozlišení. Pro praktické aplikace je užitečnější, když konstantní je relativní frekvenční rozlišovací schopnost. Na straně vysokých kmitočtů stačí tedy kratší okno a časové rozlišení je detailnější. Dostupný úsek signálu je tedy analyzován několika spektrogramy s různými délkami oken. Každý ze spektrogramů popisuje optimálně pouze jistý rozsah frekvencí. Využijeme tedy pouze tu část spektrogramu, jež obsahuje přijatelná data. Složením modifikovaných spektrogramů vznikne výsledný složený (multiresoluční) spektrogram, jež bude mít stejný absolutní počet vzorků a lepší rozlišení. Další podrobnosti o analýze signálů lze najít v literatuře [1].

Časově frekvenční analýzu lze v c++ Builderu realizovat opět pomocí komponenty SLSFourier, jejíž výstupní amplitudové spektrum je připojeno na komponentu SLWaterfall (Plot Lab).



Obr. 6.27 spektrogram získaný pomocí komponent SLSFourier a SLWaterfall (řeč - wav, 44100Hz, 1411kbps, 16bit)

7 Závěr

Práce obsahuje teorii potřebnou pro diskrétní zpracování zvuku. Je zde vypracován komplexní přehled funkcí a použití profesionálního ovladače ASIO4ALL. Dokument popisuje tři metody přístupu k ovladači a uvádí jejich srovnání, jež může pomoci při výběru optimálního přístupu pro tvorbu aplikace s ovladačem ASIO. U návrhu aplikačního prostředí je vysvětlena teorie zvukové filtrace, hudebních efektů a analýzy. Tato teorie je vždy doplněna o postup řešení daného problému a testem dokazujícím funkčnost. Při realizaci aplikačního prostředí se nepodařilo zprovoznit nahrávání zvukového signálu. Navíc jsou zde předvedeny algoritmy potřebné pro realizaci několika hudebních efektů. Je zde také obsažen popis jednotlivých technologií a komponent podstatných pro návrh zvukového softwaru v jazyce c++.

Vytvořená aplikace je schopna zpracovávat signály s celkovou latencí pohybující se v blízkém okolí 30 ms. Tyto latence jsou lidských sluchem na hranici rozeznatelnosti. Ve srovnání s jinými aplikacemi je tento výsledek uspokojivý. Profesionální programy sice vykazují lepší výsledky, ovšem ve srovnání s běžnými aplikacemi běžícími pod MS Windows má program latence značně nižší. Je třeba však dodat, že využití ASIO4ALL je vždy částečně závislé na výpočetním výkonu počítače. Vzhled programu v návrhovém a run time módu společně s celkovým propojením si lze prohlédnout v přílohách. Součástí přiloženého cd je i ovladač verze v2.9.

8 Literatura

[1] JAN, J. *Číslicová filtrace, analýza a restaurace signálů*, VUTIUM Brno, 2002, ISBN 80-214-2911-9

[2] ŠEBESTA, V.,SMÉKAL, Z. *Signály a soustavy*, Brno: FEKT VUT v Brně, 2008

[3] TIPPACH, M. *Project Brief History,* 2008, dostupné na WWW:<http://www.asio4all.com/>

[4] TIPPACH, M. *ASIO4ALL v2 Instruction Manual*, Steinberg Media Technologies GmbH, 2008

[5] SCHEFFLER, S. *ASIO SDK 2.2. Audio Streaming Input Output Development Kit*, Steinberg Media Technologies GmbH, 2006

[6] BENCINA, R., BURK, P. *PortAudio: an Open Source Cross Platform Audio API,* Proceedings of the International Computer Music Coference 2001, International Computer Music Association, San Francisco

[7] LETZ, S. *Callback adapdation techniques*, GRAME – Computer music Research Lab., Technical Note – 01-11-07

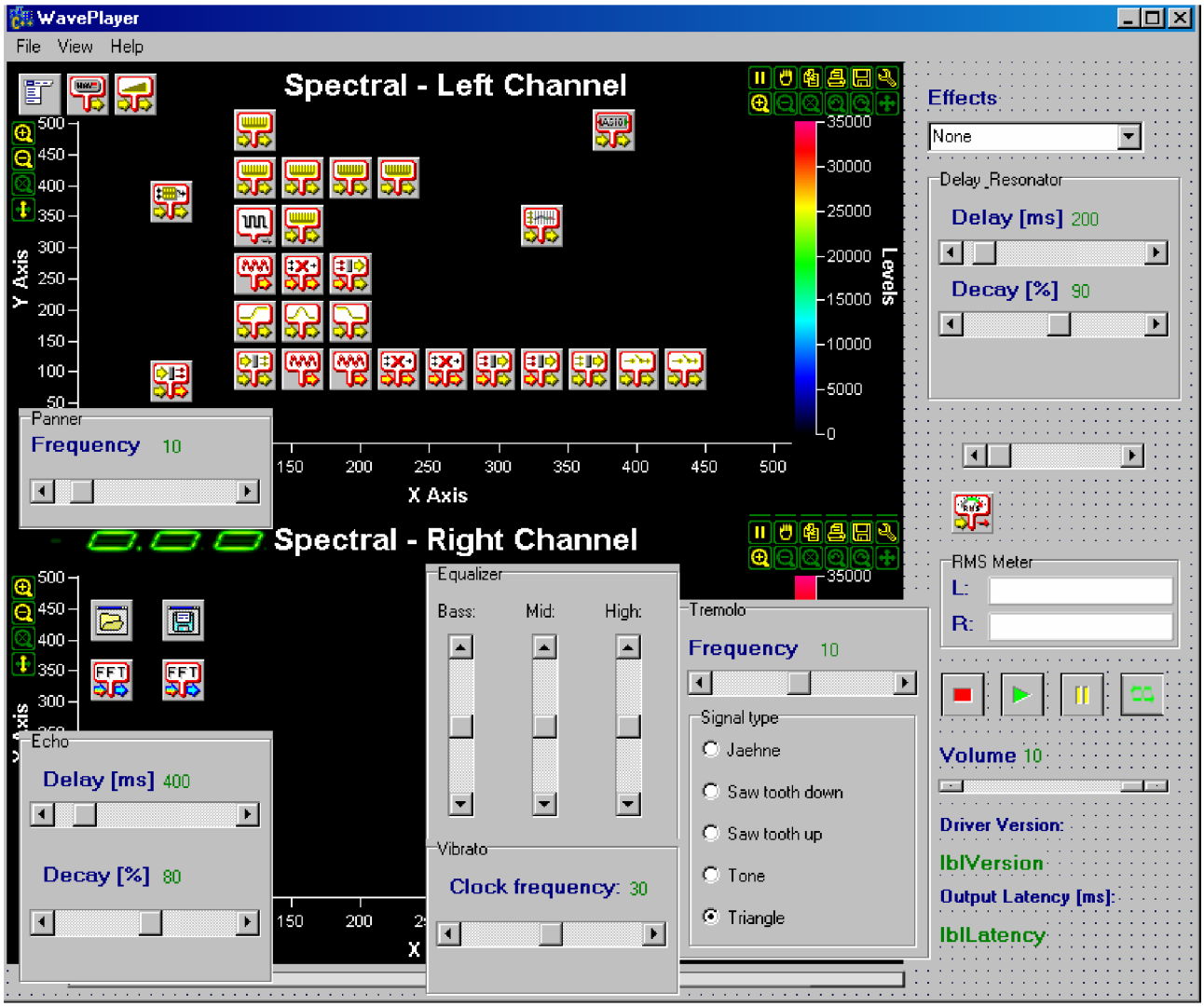
[8] KÁŇA, L., SCHIMMEL, J. *Studiová a hudební elektronika*, Brno: FEKT VUT v Brně, 2004

[9] KOLMER, F., KYNCL, J. *Prostorová akustika*, SNTL Praha, 1980, 04-514-80

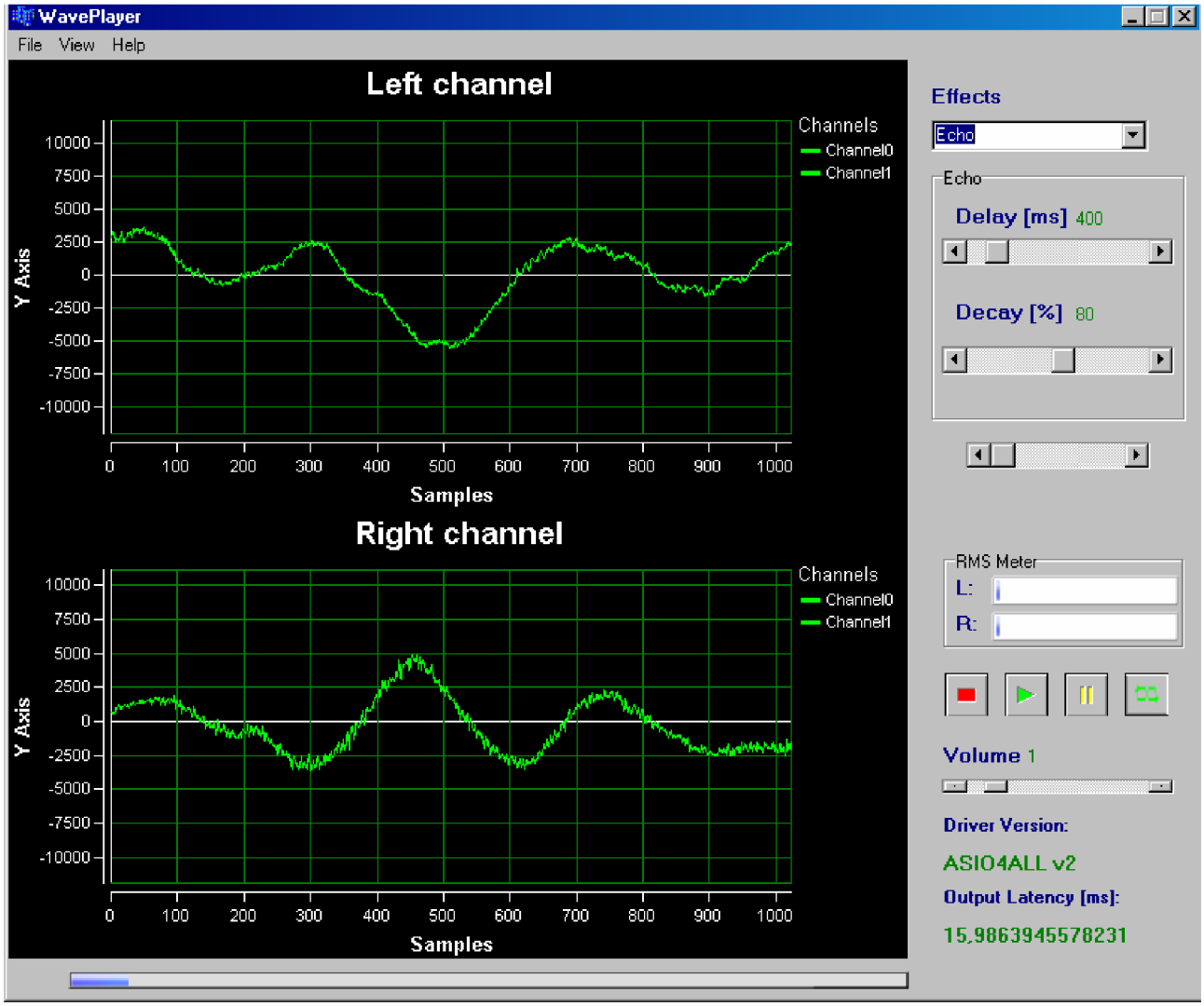
9 Seznam příloh

10 Přílohy

Příloha 1 Program v návrhovém zobrazení c++ Builder



Příloha 2 Program v real time zobrazení



Příloha 3 Kompletní zapojení aplikace pomocí OpenWire