UNIVERSITÀ DEGLI STUDI DI MILANO
FACOLTÀ DI SCIENZE MATEMATICHE,
FISICHE E NATURALI

Corso di Laurea in Scienze e Tecnologie della Comunicazione Musicale

UN SOFTWARE PER LA GESTIONE DEI PARAMETRI DEL SYNTH KORG MS2000

Relatore: Prof. Luca A. LUDOVICO
Correlatore: Dott. Adriano BARATÈ

Candidato:
Marco Zagni
Matr. 740717

Anno Accademico 2011/2012

UNIVERSITÀ DEGLI STUDI DI MILANO
FACOLTÀ DI SCIENZE MATEMATICHE,
FISICHE E NATURALI

Corso di Laurea in Scienze e Tecnologie della Comunicazione Musicale

UN SOFTWARE PER LA GESTIONE DEI PARAMETRI DEL SYNTH KORG MS2000

Relatore: Prof. Luca A. LUDOVICO
Correlatore: Dott. Adriano BARATÈ

Candidato:
Marco Zagni
Matr. 740717

Anno Accademico 2011/2012

Indice

Now, there's an XP "rule" called YAGNI: You Aren't Gonna Need It. This rule reminds us not to borrow trouble from the future and not to implement a feature today that we don't need today.

Often you will be building some class and you'll hear yourself saying "We're going to need...".

Resist that impulse, every time.

Always implement things when you **actually** need them, never when you just **foresee** that you need them.

Ron Jeffries

Capitolo 1

Introduzione

A partire dai primi anni ottanta, con l'introduzione dello standard MIDI, il mondo degli strumenti musicali elettronici è andato incontro a un'evoluzione continua. La proliferazione delle case di produzione, il costo non elevato di certi prodotti e la più recente possibilità di sincronizzazione con strumenti musicali elettronici virtuali, hanno contribuito alla conquista di una larga fetta di mercato rappresentata dai non-professionisti e a una diffusione di massa di questi prodotti. Già dopo pochi anni dalla nascita dello standard, se ne era intuito il potenziale [2]:

> The Musical Instrument Digital Interface (MIDI) is now a de facto standard for the digital representation of musical events. Actions of live musical performers are being "MIDIfied" commercial software is being based on MIDI-gated conceptions, and digital synthesizers are being slaved to MIDI masters. MIDI-based hardware and software is also proliferating at a tremendous rate, virtually ensuring that the characteristics of MIDI will play an important role in shaping a significant portion of future music.

Le caratteristiche hardware degli strumenti e le interfacce uomo-macchina si sono raffinate col tempo, rendendo più accessibile la gestione delle funzionalità offerte. Un aspetto che sembra meritare scarsa attenzione dalle case di produzione e che non sembra essere stato coinvolto nel processo evolutivo dell'hardware è quello che si riferisce ai software di gestione dei parametri "dall'esterno". Restringendo il campo degli strumenti ai sintetizzatori, è stata notata un'assenza diffusa di programmi dedicati disponibili al pubblico[1]. Questo lavoro si concentra sullo sviluppo di un

[1] È stata effettuata una ricerca tramite i siti ufficiali delle principali case di produzione (Korg, Yamaha, Moog, Kurzweill, Casio, Alesis, Roland). Alcune di queste (Moog, Kurzweill) forniscono

programma per uno specifico sintetizzatore: l'MS-2000 della Korg. La sua progettazione e il suo design ricalcano il modello classico di synth completamente analogico, quindi dotato di un numero considerevole di controlli manuali. Nonostante ciò, molti parametri modificabili sono raggiungibili solo tramite lo schermo LCD di dimensioni molto limitate, navigabile tramite dieci tasti.

Il prodotto di questo lavoro, MS-2000 Unfolder, darà la possibilità all'utente di avere una visione chiara e completa di tutte le funzioni di questo sintetizzatore, di modificarne i parametri e di salvarne le impostazioni in locale o direttamente nello strumento. Questa è una caratteristica del tutto nuova rispetto a quelle dei software attualmente disponibili, simili solo nelle modalità di invio/ricezione dei dati dal synth.

Questo elaborato si svilupperà in cinque capitoli. In seguito all'introduzione, sarà descritta la situazione di partenza, analizzando lo stato dell'arte in quanto a software specifici per lo strumento in questione. Si arriverà quindi a definire un insieme di funzioni disponibili da cui partire o da perfezionare. Il terzo capitolo si occuperà della descrizione degli strumenti che verranno utilizzati per giungere al risultato finale, sia software, sia hardware. Trattandosi di un lavoro suddiviso in più fasi, la strumentazione è destinata a variare molto, per esempio, tra la parte di analisi e quella di sviluppo del software. Dopo aver messo bene in chiaro la situazione, si passerà alla descrizione effettiva del procedimento che di genesi del software. In questo capitolo, considerato il "corpo" vero e proprio dell'elaborato, verranno affrontati i problemi relativi alla compatibilità dei file trasmessi dal synth, alla mappatura degli stessi e alla programmazione di un'interfaccia del tutto inedita e sperimentale. Infine si trarranno le conclusioni sull'operato, indicando se gli obiettivi prefissati saranno stati effettivamente raggiunti, se esistono sviluppi futuri e se sono disponibili perfezionamenti del risultato ottenuto.

software di ottima qualità per la gestione dei parametri sul proprio computer. Le altre, al contrario, sono parzialmente (o in certi casi totalmente) sprovviste di tale software.

Capitolo 2

Analisi e stato dell'arte

2.1 Il Sintetizzatore

L'MS2000 è un sintetizzatore basato sul sistema di modeling analogico DSP prodotto a partire dal 2000 [1] dalla casa di produzione giapponese KORG. Le prime due versioni dello strumento sono state rilasciate sotto il nome "MS2000" e "MS2000R", rispettivamente per la versione completa di tastiera di 3 ottave e 1/2 e la versione rack. Sebbene sia stato superato negli anni da nuovi modelli, innovativi sia nell'hardware, sia nel design, rimane tuttora uno degli strumenti più comuni e apprezzati nel suo genere. Il prezzo contenuto per uno strumento professionale e le caratteristiche timbriche che soddisfano più generi musicali, ne hanno facilitata la diffusione.

[1] Nel 2003 è stata lanciata sul mercato una versione aggiornata nel design e in qualche funzionalità, ma che ricalca fedelmente l'originale per quanto riguarda il software interno per la gestione dei parametri. Il nome della nuova versione è: "MS-2000B".

2.1.1 Caratteristiche principali

Figura 2.1: Pannello frontale del synth MS2000.

È stato costruito seguendo il classico modello di sintesi sottrattiva, basato su due oscillatori più un generatore di rumore bianco, processati da un filtro multi-banda. Da [1]:

> The MS2000/MS2000R provides eight types of oscillator algorithms, including waveforms of analog synthesizers, and places the most important sound parameters on the front panel so that you can modify sounds as you play or perform a variety of realtime editing, with the same ease of operation as on an analog synthesizer.

Sono presenti 128 programmi, suddivisi in 8 banchi (A-H) da 16 suoni ciascuno, tutti editabili e sovrascrivibili. L'MS2000 mette a disposizione un arpeggiatore e un mod sequencer a 16 passi, entrambi sincronizzabili con dispositivi esterni via MIDI. Il sequencer in particolare è molto vicino alla concezione tipica degli strumenti puramente analogici, così come il sistema di "Virtual Patch" [1]:

> Not only EG and LFO, but also velocity and keyboard tracking can be used as modulation sources, and assigned to parameters that make up the sound, giving you even more freedom to create sounds.

Il pannello frontale presenta 35 potenziometri per il controllo diretto di altrettanti parametri, 24 tasti per la selezione e due mod wheels (una per il pitch e l'altra

programmabile). Per l'interazione con le impostazioni generali della macchina o la modifica dei parametri non raggiungibili tramite i potenziometri, è presente uno schermo LCD relativamente piccolo[2], navigabile per mezzo di 10 tasti.

Per quanto riguarda le connessioni MIDI, l'MS2000 è dotato di tre porte standard, ossia In, Out e Thru. Il loro funzionamento è spiegato in maniera esaustiva in [11]:

> Le informazioni MIDI viaggiano in una direzione sola, come un cavo audio; non importa quale capo del cavo innestate nella chitarra e quale nell'amplificatore: il flusso della corrente elettrica andrà sempre dalla chitarra all'amplificatore. I cavi MIDI funzionano nello stesso modo, poiché il jack audio sulla chitarra è un'uscita e quello sull'amplificatore è un ingresso.
>
> **MIDI Out**
>
> Il MIDI Out ha la funzione dì uscita MIDI. Da questo connettore escono i messaggi MIDI generati dal dispositivo o dal software. È importante capire che ciò che esce è esclusivamente ciò che proviene dal trasmettitore o dal dispositivo master, che può essere una tastiera, un modulo sonoro, un sequencer o qualunque altro software o hardware che trasmette MIDI. Pertanto, se avete diversi dispositivi nella vostra configurazione e volete metterli in catena l'uno con l'altro, dovete utilizzare il MIDI Thru (il Thru fisico sul vostro dispositivo o un'impostazione Thru nel vostro software) per rimandare le informazioni che arrivano sul MIDI In. Il MIDI Out non invia informazioni audio, ma semplicemente messaggi MIDI interpretati dal ricevitore MIDI In di un altro dispositivo. Essi sono codici digitali che rappresentano che cosa e in che modo musica o eventi (come sono premuti i pedali, mossi i fader, cambiati i numeri di program) sono eseguiti sullo strumento master. Le uscite MIDI dovrebbero essere sempre agganciate agli ingressi MIDI, indipendentemente dal tipo di configurazione.
>
> **MIDI In**
>
> Il MIDI In ha la funzione di ingresso MIDI, il luogo in cui i dati MIDI arrivano e vengono inviati al processore hardware o software. Tutto quello che entra viene processato e, nel caso in cui vogliate utilizzare i dati MIDI in un altro dispositivo, può essere mandato indietro attraverso il MIDI Thru. Sostanzialmente, i dati MIDI indicano al modulo sonoro, per esempio, di suonare una certa nota a una determinata velocità e su un

[2]Lo schermo LCD è formato da due righe di 16 caratteri ciascuna.

determinato canale, utilizzando un determinato suono nella sua memoria in modo da far sì che il sintetizzatore produca un suono attraverso le sue uscite audio. Le uscite audio possono quindi essere monitorate o registrate. Ma non fate errori; ciò che state registrando o ascoltando sono le uscite audio, non le uscite MIDI. I MIDI In possono ricevere le informazioni sia dai MIDI Out, se il modulo precedente nella catena è il master, sia dai MIDI Thru, se il modulo precedente nella catena sta rimandando le informazioni inviate da un altro modulo master della stessa catena.

MIDI Thru

Il MIDI Thru ha la funzione di una ripetizione della porta MIDI In; ciò significa che tutto ciò che entra nell'ingresso MIDI esce attraverso il MIDI Thru, non il MIDI Out. In altre parole, esso ripete le informazioni che arrivano al MIDI In dello stesso dispositivo. Il MIDI Thru non ripeterà comunque le informazioni eseguite da questo modulo. Per esempio, tutto quello che suonate nel sintetizzatore A sarà inviato al sintetizzatore B attraverso il suo jack Out, non il suo jack Thru. Il sintetizzatore B dovrà utilizzare il MIDI Thru per inviare le informazioni dal sintetizzatore A al sintetizzatore C, ma se volete inviare al sintetizzatore C informazioni eseguite sul B, dovrete utilizzare il MIDI Out di quest'ultimo.

2.1.2 Com'è strutturato un programma

L'MS2000 è caratterizzato da una polifonia a 4 voci e dalla possibilità di gestire 2 timbri in un unico programma. Un programma può essere classificato in una delle due macro-categorie che dipendono dal parametro "Voice Mode": synth e vocoder. Della prima fanno parte le modalità "Single", per programmi a un timbro, "Dual" e "Split" per quelli in cui si possono utilizzare due timbri[3]. La categoria "Vocoder" invece include solo la modalità omonima.

[3]"Dual" permette di utilizzare due suoni diversi in contemporanea, ciascuno con il proprio set di parametri, modificabili effettuando uno switch tra i timbri tramite un tasto. "Split" divide la tastiera in un punto a scelta in base a nota e ottava, permettendo la separazione tra i due timbri.

2.1.2.1 Synth Program

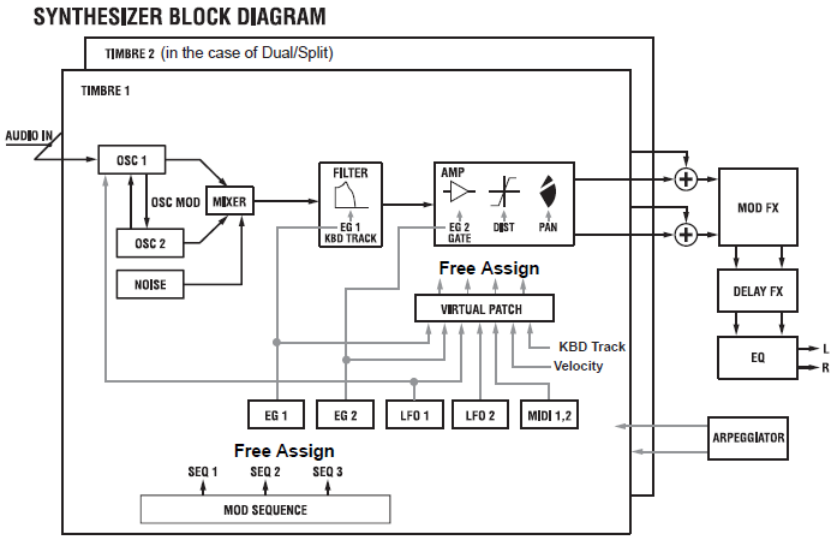

Figura 2.2: Schema di funzionamento di un programma di tipo "Synth".

Come s'intuisce dallo schema, un programma di tipo "synth" è formato dai timbri e dagli effetti[4].

Timbre 1/2 Un timbro è costituito da Osc1/Osc2/Noise, Mixer, Filter, Amp, Eg, Lfo, Virtual Patch, e Mod Sequence. Se la modalità voce è Single, solo un timbro sarà attivo. Se la modalità voce è Dual o Split, entrambi i timbri saranno disponibili.

Osc1 / Osc2 / Noise L'oscillatore 1 permette di selezionare otto algoritmi di oscillazione differenti, quali SAW (onda a dente di sega), PWM, Cross e DWGS (Digital Waveform Generator System)[5]. Dall'input jack AUDIO IN 1/2 può essere processata una forma d'onda esterna. L'oscillatore 2 permette di scegliere tra tre tipi di forma d'onda: SAW, SQUARE e TRIANGLE. Può essere anche usato come modulatore per le funzioni SYNC e RING tipiche dei synth analogici. Noise è il generatore di rumore bianco.

[4]Le caratteristiche descritte in 2.1.1, quindi arpeggiatore, sequencer e virtual patch, sono comuni ad ogni tipo di programma.

[5]Sistema ispirato dallo storico synth analogico Korg DW-8000 (vedi [3], p.13). Permette la generazione di 64 forme d'onda che imitano 64 strumenti musicali diversi.

Mixer Permette di stabilire il livello di OSC1, OSC2 e NOISE, e inviare il segnale combinato al FILTER.

Filter Taglia o enfatizza certe componenti frequenziali che arrivano dall'oscillatore, regolandone il tono. Secondo il meccanismo descritto in [5]:

> A filter is inserted between two sections of an electronic system in order to control the power transferred from one to the other in a desired way that depends on frequency. Such power control is achieved by shaping the spectrum of the input signal (Parseval's theorem) by leaving the frequencies of the passband almost unaffected (introducing zero or low and tolerable attenuation) and by eliminating or reducing the rest of the spectrum, through the introduction of high attenuation which may be occasionally infinite on a finite set of frequency points.

Sono disponibili 4 tipi di filtro: –12 o –24 dB/oct LPF (filtro passa-basso), –12 dB/oct BPF (filtro passa-banda), or –12 dB/oct HPF (filtro passa-alto).

AMP Costituito da AMP (amplificatore), DIST (distorsione), e PAN (panpot). AMP regola il volume, PAN setta il panning stereo in uscita.

EG 1/2 Un EG (Envelope Generator) applica una modifica tempo-variante ad un certo parametro del suono. Ce ne sono 2 per ogni timbro e ognuno ha 4 parametri: ATTACK (tempo d'attacco), DECAY (tempo di decadimento), SUSTAIN (livello di sostegno), e RELEASE (tempo di rilascio). L'EG1 è assegnato alla frequenza di cutoff del FILTER, EG2 al volume di AMP[6].

LFO 1/2 L'LFO (Low Frequency Oscillator) cambia ciclicamente un parametro specifico di un suono. Nel caso dell'MS2000, il primo LFO modifica il comportamento dell'OSC 1, LFO 2 funge da sorgente di modulazione per il pitch.

Effetti Ogni programma ha un effetto di modulazione, delay ed equalizzatore. Tra le modulazioni è possibile scegliere tra effetti come chorus, flanger e phaser. Sono invece tre i tipi di delay disponibili, incluso quello di tipo stereo.

[6]EG 1/2 possono essere assegnati ad altri parametri tramite il sistema di Virtual Patch.

2.1.2.2 Vocoder Program

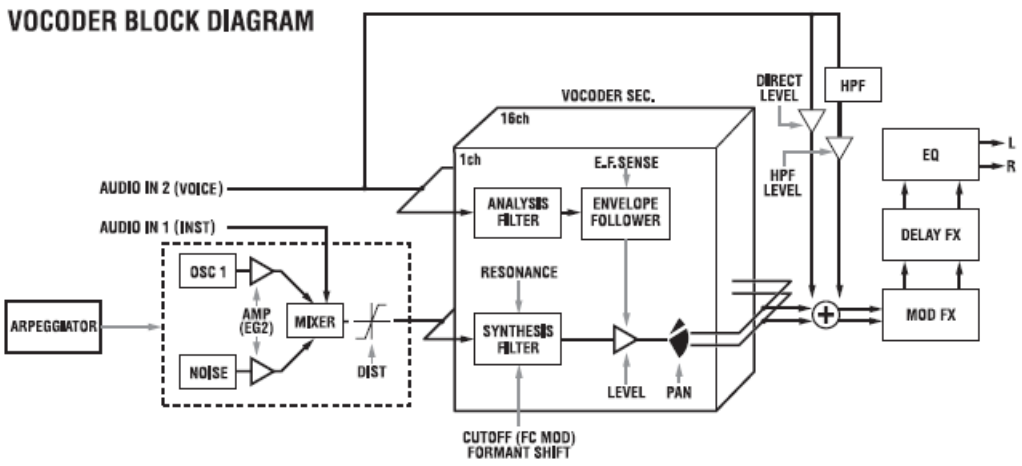

Figura 2.3: Schema di funzionamento di un programma di tipo "Vocoder".

Il Vocoder è un sistema di analisi/sintesi del suono, usato generalmente per riprodurre la voce umana. Quello implementato nell'MS2000 è basato sulla modifica del segnale generato internamente, in base alla fase di quello in entrata [6]:

> The phase vocoder is an analysis-synthesis system that has as intermediate data the time-variant discrete Fourier spectrum of the input signal. It can be formulated in such a way that the synthesized signal is identical to the original, both theoretically and practically. The intermediate data can be transformed, also with no loss of information, into the more conventional magnitude and frequency representation. These intermediate data can then be used to resynthesize the tone at different pitches or different rates than the original with the advantage that when no modification is made, the synthetic tone is absolutely identical to the original.

I programmi di tipo Vocoder sono costituiti da Osc1 / Noise, Mixer, Vocoder Sec., Effect, e Arpeggiatore. Non essendoci generazione timbrica pura come nei Synth Program, il vocoder riceve un segnale in entrata da AUDIO IN [7] cui è applicato un segnale generato internamente dall'oscillatore 1 o dal generatore di rumore bianco.

[7] L'MS2000 presenta un line in jack per ricevere segnali audio esterni. La versione MS2000B ha anche un'entrata XLR per microfono.

OSC1/NOISE/AUDIO IN 1 jack (carrier) Il segnale proveniente dall'oscillatore 1 e dal generatore di rumore è definito "portante" ed è quello cui si applica l'effetto di vocoding. È possibile affiancare ai generatori interni una forma d'onda proveniente dall'entrata jack AUDIO IN 1.

AUDIO IN 2 jack (modulator) Il segnale proveniente da questa entrata è definito "modulatore", poiché andrà a modificare il segnale portante. La scelta più comune è quella di utilizzare la voce tramite un microfono, ma è possibile introdurre qualsiasi tipo di forma d'onda.

VOCODER SEC. È costituito da due set da 16 filtri passa-banda ciascuno (Analysis Filter e Synthesis Filter) e da un Envelope Follower. Il segnale audio modulatore è trasmesso ai 16 filtri di analisi e l'Envelope Follower rileva le variazioni di volume nel tempo per ogni banda di frequenza. Quindi il segnale portante viene introdotto negli altri 16 filtri di sintesi e processato dai valori di volume rilevati in precedenza dall'Envelope Follower.

2.2 I Software

Korg non ha mai rilasciato software ufficiali per la gestione dei parametri dell'MS2000, nemmeno tool in gradi di salvare in locale le impostazioni interne o i preset. Esistono tuttavia delle semplici utility[8] che rendono possibili alcune operazioni di base. Tutti i programmi sono utilizzabili unicamente in ambiente Windows, ad eccezione di MS2000 Librarian.

[8]Korg ha reso disponibile sul proprio sito ufficiale un link alla pagina di Daz Richards, lo sviluppatore degli unici tool esistenti che permettono di gestire i SysEx dump dell'MS2000.

2.2.1 MS2K Patch Buddy

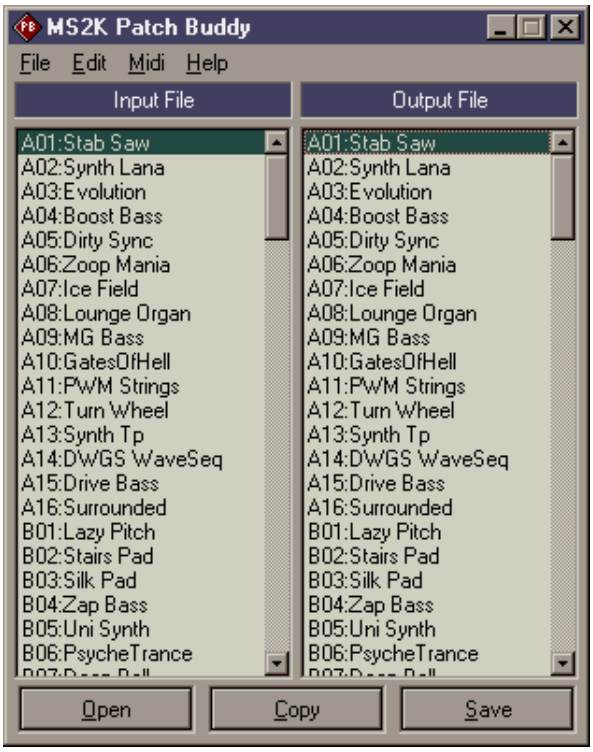

Figura 2.4: Schermata principale.

Questo è il programma da cui è partito il lungo processo di raffinamento e ampliamento delle funzionalità gestionali che ha portato al prodotto finale di questo lavoro. Le uniche funzioni disponibili sono quelle di importazione/esportazione dei file SisEx dal synth e di apertura/salvataggio in locale degli stessi. Funziona unicamente in ambiente Windows ed è stato sviluppato immediatamente dopo l'uscita del synth[9], quindi presenta qualche problema di compatibilità con i sistemi operativi più moderni[10]. Le possibilità di modifica del file SysEx sono limitate alla modifica del nome dei singoli suoni e al riposizionamento degli stessi all'interno del set.

[9]Presumibilmente sviluppato sotto Windows XP (2001).

[10]Durante la fase di studio del programma si sono presentati problemi di connessione e scambio di dati via MIDI e difficoltà d'impostazione delle interfacce MIDI utilizzate.

2.2.2 MS2K Syxplorer

Figura 2.5: Schermata principale.

MS2K Syxplorer è un software simile a quello appena descritto: permette di visualizzare il set completo di suoni e organizzarne la lista. È stata aggiunta la funzione "audition" che permette di inviare istantaneamente uno specifico suono al synth per poterne verificare la resa audio.

2.2.3 MS2000 Librarian

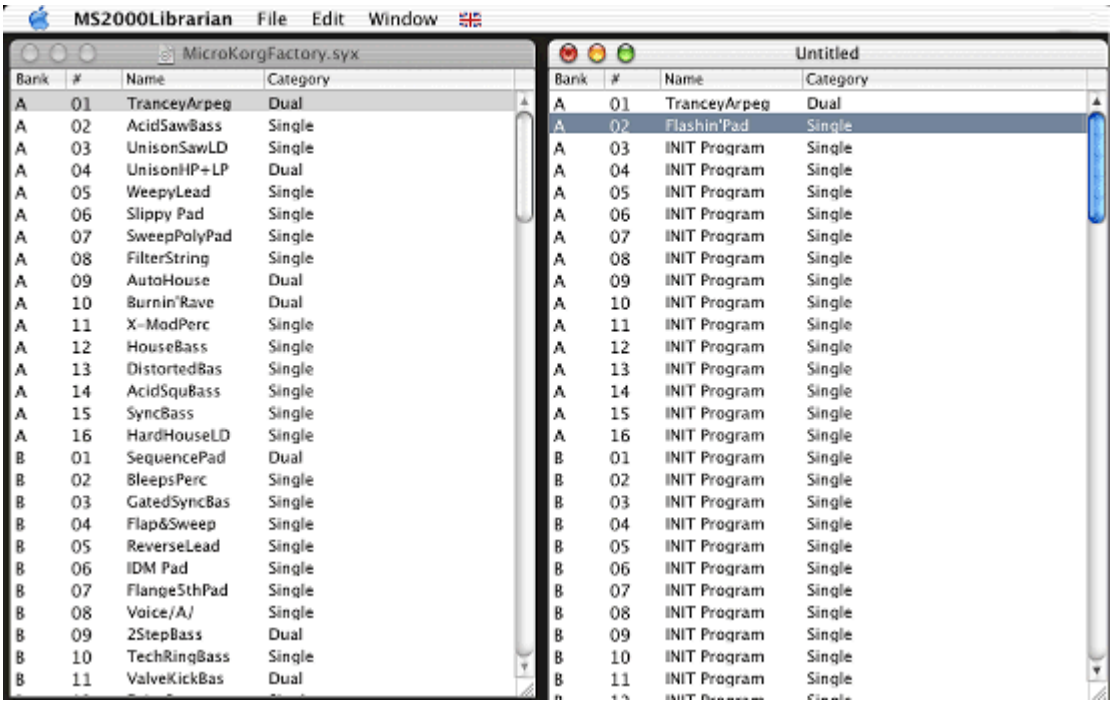

Figura 2.6: Schermata principale.

Software analogo al MS2K Syxplorer, ma per sistemi operativi Mac. Questo tool aiuta l'utente a compilarsi una libreria di suoni e a gestirla, senza poter modificare alcun suono.

2.2.4 MS2K Patch Randomizer

Figura 2.7: Schermata principale.

Questa utility genera un sound program caratterizzato da parametri aleatori e lo trasmette immediatamente all'MS2000. Non permette la trasmissione in entrata di file .syx e non gestisce più di un suono alla volta.

2.2.5 MS2K Patch Script Buddy

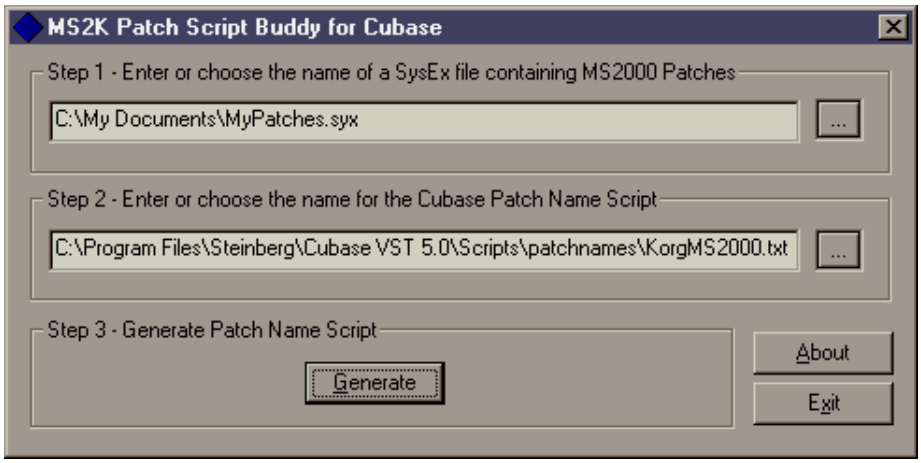

Figura 2.8: Schermata principale.

Permette la sincronizzazione nominale tra i Program Name contenuti nel file .syx dell'MS2000 e i nomi delle patch create sul proprio computer tramite Cubase o Sonar.

	P. Buddy	**Syxplorer**	**Librarian**	**P. Randomizer**	**P. Script Buddy**
Modifica parametri	solo nome	✗	✗	random singolo	✗
Audition	✗	✓	✓	✓	✗
Ricez./Invio Dump	✓	✗	✗	solo invio	✗
Modifica banchi	✓	visuale	visuale	✗	✗

Tabella 2.1: Tabella riassuntiva delle caratteristiche del software esistente.

Capitolo 3

Tecnologie utilizzate

Prima di descrivere gli strumenti con cui si lavorerà, è bene fare chiarezza sulle diverse fasi in cui è suddiviso l'approccio al caso di studio.

Gli step principali sono tre:

1. Reverse engineering partendo dal dump del synth e mappatura dello stesso
2. Scrittura del codice sorgente del software
3. Test delle funzionalità

Ogni passo richiede strumenti diversi, sebbene la prima e la terza fase abbiano molti punti in comune per quanto riguarda le operazioni da compiere.

3.1 Hardware

3.1.1 Korg MS2000B

Il sintetizzatore preso in considerazione per lo studio è il Korg MS2000 versione B, ossia quella rilasciata nel 2003. Le uniche differenze rispetto al suo predecessore sono la presenza di un'entrata XLR per il microfono (da utilizzare in modalità vocoder), i preset e il colore. L'hardware e il software sono identici a quelli dell'MS2000, quindi valgono le stesse specifiche descritte in 2.1. Il firmware installato al momento dell'analisi è la versione 1.07.

3.1.2 Edirol UM-2

Questa è l'interfaccia MIDI/USB utilizzata per rendere possibile la comunicazione tra PC e synth. Presenta due entrate e due uscite, un numero esiguo ma sufficiente per eseguire alcune operazioni descritte meglio in 4.2.

3.1.3 Dell Inspiron 1564

Laptop utilizzato per la comunicazione con l'interfaccia MIDI e, quindi, per ricevere i dump dal sintetizzatore. Monta un processore Intel Core I5-520M e 4GB di ram ddr3. Il sistema operativo installato è Windows 7 64bit Home Edition. Tutti i software descritti nella prossima sezione sono stati installati e saranno utilizzati su questa macchina.

3.2 Software

Durante la scelta dei software da utilizzare, è stata data la precedenza a materiale freeware che garantisse comunque le funzionalità necessarie allo sviluppo del lavoro. Microsoft Visual Studio 2012 rappresenta l'eccezione, essendo stato messo a disposizione dal Dipartimento di Informatica e Comunicazione tramite il programma MSDNAA di Microsoft.

3.2.1 Microsoft Visual Studio 2012

Ambiente di sviluppo integrato prodotto da Microsoft e basato sulla piattaforma .NET. Il linguaggio di programmazione scelto per sviluppare MS2000 Unfolder è stato C#. Per la parte grafica sarà utilizzata la libreria WPF[1] e il linguaggio XAML.

3.2.2 HxD

È un editor esadecimale gratuito sviluppato da Maël Hörz che permette il confronto, la modifica e la gestione di ogni tipo di file. Sarà utile in particolare nella fase di ingegneria inversa e in quella finale, al momento del confronto tra file SysEx originale e file SysEx processato dal programma.

[1] Windows Presentation Foundation è un sistema di presentazione basato su un motore di rendering vettoriale che si appoggia alle DirectX per sfruttare l'accelerazione hardware delle moderne schede grafiche.

3.2.3 MIDI-OX

MIDI-OX è un tool multiuso sviluppato da Jamie O'Connell e Jerry Jorgenrud: è al tempo stesso uno strumento diagnostico e un gestore di file System Exclusive. È in grado di eseguire il filtraggio e la mappatura dei flussi di dati MIDI, sia in entrata, sia in uscita.

3.2.4 Microsoft Excel

Programma prodotto da Microsoft, dedicato alla produzione e alla gestione dei fogli elettronici. È stato utilizzato per redigere la mappatura iniziale.

3.2.5 C# MIDI Toolkit

Insieme di strumenti per la piattaforma .NET che permettono di interagire con sistemi basati sull'architettura MIDI. È stato sviluppato da Leslie Sanford ed è specifico per un approccio C#/.NET.

3.2.6 MS2K Patch Buddy

Precedentemente descritto in 2.2.1, MS2K Patch Buddy è l'utility da cui si è sviluppata l'idea che ha portato a MS2000 Unfolder. Verrà riutilizzato a fine lavoro per testare la correttezza e la compatibilità dei file generati dal nuovo software.

Capitolo 4

Sviluppo del software MS2000 Unfolder

4.1 Preparazione degli strumenti

Il file attorno al quale ruota tutto il lavoro effettuato è un messaggio MIDI di tipo System-Exclusive (o SysEx, vedi 4.2.1), trasmesso dal synth attraverso un "MIDI-Dump" e solitamente salvato con estensione .syx. Per permetterne la trasmissione, si utilizza un "ponte" tra lo strumento e il computer. Questa funzione è svolta dall'interfaccia esterna.

Ogni dispositivo basato sull'architettura MIDI, gestisce un flusso di dati in entrata o uscita. I messaggi possono essere di tipo diverso, ma permane la necessità di regolare questi flussi attraverso un mezzo che sia sempre in movimento. Questa funzione è svolta dal MIDI-Clock [15]:

> As a hardware interface, MIDI is concerned with real-time processing. In the MIDI sense, music consists of a string of events. These events happen one at a time and from moment to moment. To process streams of those events, something must always be in motion. What is always in motion in MIDI hardware is a clock. [...] Traffic in both directions is regulated by the MIDI clock. Every hardware signal representing a musical event must be time-stamped by the clock, which is always moving.

Una volta collegata opportunamente all'MS2000[1] è necessario settare il MIDI-Clock interno al synth su "External", seguendo le indicazioni in[1]:

[1] Il collegamento tra interfaccia e synth avviene nel modo "classico": la porta "IN" del synth è connessa alla porta "OUT" dell'interfaccia e viceversa. Sono ovviamente necessari due cavi appositi.

Internal The MS2000/MS2000R will be the master (the controlling device). A connected external MIDI device (e.g., sequencer) will synchronize to the sequence or arpeggiator of the MS2000/MS2000R. If "Tempo Sync" is turned ON for LFO1 and LFO2, the LFO frequency will synchronize to the tempo specified by the [TEMPO] knob.

External The MS2000/MS2000R will be the slave (the controlled device). The MS2000/MS2000R's sequences and arpeggiator will synchronize to the MIDI clock from a connected external MIDI device. If "Tempo Sync" is turned ON for LFO1 and LFO2, the LFO frequency will synchronize to the MIDI clock from the external MIDI device.

Auto When MIDI clock data is input from a connected external MIDI device, the MS2000/MS2000R will automatically operate with the External setting. Normally, it will operate with the Internal setting.

Questo passaggio preparatorio si rivela di estrema importanza in quanto le impostazioni "Internal" e "Auto"[2] corrompono l'operazione di dump. In assenza di un MIDI-Clock in entrata, il flusso di byte che arriva al computer e dovrebbe rappresentare unicamente il SysEx, in realtà viene disturbato periodicamente dal messaggio di clock generato dall'arpeggiatore. Questo rende impossibile il riconoscimento del flusso come effettivo messaggio SysEx. Il parametro di MIDI-Clock non fa parte del set modificabile tramite MS2000 Unfolder (Program Mode), bensì rientra nelle impostazioni "Global Mode"[3].

[2] Nel caso in cui non ci sia un messaggio di MIDI-Clock inviato dal computer al synth.
[3] Alla pagina 3, tab C dello schermo LCD.

Key	Page
SELECT [1]	Page1A: GLOBAL “Mst.Tune”
SELECT [2]	Page1D: GLOBAL “Vel.Curve”
SELECT [3]	Page1F: GLOBAL “AudioInThru”
SELECT [4]	Page2A: MEMORY “Protect”
SELECT [5]	Page3A: MIDI “MIDI Ch”
SELECT [6]	Page3C: MIDI “Clock”
SELECT [7]	Page3D: MIDI “MIDI1”
SELECT [8]	Page3F: MIDI “MIDI Dump”
SELECT [9]	Page4A: MIDI FILTER “ProgChg”
SELECT [10]	Page4B: MIDI FILTER “CtrlChg”
SELECT [11]	Page4C: MIDI FILTER “P.Bend”
SELECT [12]	Page4D: MIDI FILTER “SystemEx”
SELECT [13]	Page6A: PEDAL&SW “A.Pedal”
SELECT [14]	Page6B: PEDAL&SW “A.SwFunc”
SELECT [15]	Page7A: USERSCALE “Key: Detune”
SELECT [16]	Page8A: CALIB (AS)

Tabella 4.1: Organizzazione della modalità “Global”.

L’altro accorgimento fondamentale è disabilitare la protezione della memoria interna. Questa operazione permette di salvare ogni modifica a un synth program, sovrascrivendo i valori precedenti. È possibile controllarne e cambiarne l’impostazione sempre nella modalità “Global”[4].

Infine, si dovrà settare il parametro *Sys* all’interno della pagina “MIDI FILTER - SystemEx” su *Enabled*, altrimenti non sarà possibile inviare un file SystemExclusive dall’esterno.

Settare il MIDI-Clock, la protezione di memoria e il Sys nel modo corretto sono azioni richieste anche prima dell’utilizzo di MS2000 Unfolder. Nel primo caso un MIDI-Clock non-External renderebbe illeggibile il flusso di dati trasmesso dal synth. In caso di “memory protect” attivo, al riavvio dell’MS2000, si ritroverebbero i banchi di suoni precedenti al trasferimento dal computer.

[4] Alla pagina 2, tab A dello schermo LCD.

4.2 Mappatura del dump

4.2.1 Il messaggio MIDI System-Exclusive

Nel formato MIDI esistono due tipi di messaggio: *Channel Message* e *System Message*. Quest'ultimo è il "macro-gruppo" che contiene il messaggio d'interesse. Come descritto in [8]:

> As the name implies, System messages are globally transmitted to every MIDI device in the MIDI chain. This is accomplished because MIDI channel numbers aren't addressed within the byte structure of a System message. Thus, any device will respond to these messages, regardless of its MIDI channel assignment. The three System message types are System-Common messages, System Real-Time messages, and System-Exclusive messages.
>
> [...] The System-Exclusive (SysEx) message lets MIDI manufacturers, programmers, and designers communicate customized MIDI messages between MIDI devices. These messages give manufacturers, programmers, and designers the freedom to communicate any device-specific data of an unrestricted length as they see fit. SysEx data is commonly used for the bulk transmission and reception of program/patch data, sample data, and real-time control over a device's parameters.

Come definito dallo standard MIDI, un messaggio SysEx è composto da 4 parti:

- **SysEx status header:** il primo Byte serve all'identificazione del tipo di messaggio che si sta inviando. Il valore esadecimale è 'F0'.
- **Numero ID:** ogni casa produttrice che adotta lo standard MIDI riceve un proprio numero identificativo. Questo valore univoco è trasmesso nei messaggi SysEx per avere la sicurezza di interagire solo con strumenti compatibili. Come descritto in [8]:

  > Manufacturers are granted unique identification (ID) numbers by the MMA or the JMSC, and the manufacturer ID number is included as part of the System Exclusive message. The manufacturers ID is followed by any number of data bytes, and the data transmission is terminated with the EOX message. Manufacturers are required to publish the details of their System Exclusive data formats, and other manufacturers may freely

utilize these formats, provided that they do not alter or utilize the format in a way which conflicts with the original manufacturers specifications. Certain System Exclusive ID numbers are reserved for special protocols. Among these are the MIDI Sample Dump Standard, which is a System Exclusive data format defined in the MIDI specification for the transmission of sample data between MIDI devices, as well as MIDI Show Control and MIDI Machine Control.

- **Dati:** il corpo dei dati effettivamente rilevanti, contenente le impostazioni globali o dei parametri. Può avere lunghezza arbitraria.
- **EOX (End Of Exclusive)**: byte specifico che indica la fine del SysEx. Il valore esadecimale è 'F7'.

In aggiunta all'ID, Korg aggiunge altri due byte, uno che identifica il modello dello strumento e uno per il tipo di dump che si sta trasmettendo.

Posizione	Valore Hex	Significato
0	F0	Exclusive Status
1	42	Korg ID
2	30	Global Midi Channel (3n)
3	58	Modello Strumento (MS2000)
4	4C	Tipo di dump (All Programs)
5 ~ 37152	-	Dati relativi ai parametri dei 128 suoni
37153	F7	End Of SysEx

Tabella 4.2: La composizione del file System-Exclusive esaminato.

4.2.2 Il primo approccio: reverse engineering

Nonostante Korg fornisca una documentazione MIDI (vedi [4] e Appendice A) con una propria mappatura del SysEx, si è rivelato necessario produrne una nuova, in quanto le indicazioni non combaciano effettivamente con i parametri impostati nel synth. Per raggiungere quest'obiettivo è stata percorsa la strada dell'ingegneria inversa. Attraverso l'uso del software MIDI-OX (descritto in 3.2.3), si è seguito il procedimento seguente:

1. Regolazione delle impostazioni generali[5] per lo scambio di dati tra Synth e MIDI OX.

2. Ricezione del dump tramite MIDI-OX.

3. Copia della sequenza di byte ricevuti nell'editor esadecimale HxD (vedi 3.2.2).

4. Modifica e sovrascrittura di un parametro del Sound Program "A01".

5. Nuovo invio del SysEx modificato.

6. Copia della sequenza di byte ricevuti da MIDI OX in un nuovo documento di HxD.

7. Uso della funzionalità "Confronta" di HxD[6].

8. Annotazione in un documento Excel della posizione dei byte che hanno variato valore in seguito alla modifica del parametro.

9. Ripetizione dei passaggi da 4 a 7 per identificare il comportamento dei byte per ogni possibile valore del parametro modificato[7].

La compilazione della mappatura ha richiesto un lungo lavoro, soprattutto per definire con correttezza il valore di certi byte. Alcuni, ad esempio, variavano il proprio valore in base alla modifica di tre o più parametri, cosa che ha reso il procedimento macchinoso e soggetto a continue variazioni.

Terminata l'analisi del suono "A01" e, quindi, ottenute tutte le informazioni sul comportamento del file SysEx alla modifica di ogni singolo parametro del sound program stesso, è stata provata l'applicazione dello stesso schema per i rimanenti. Il risultato è stato negativo, per qualche motivo il pattern posizione-Byte/parametro si è rivelato non replicabile e questo avrebbe rappresentato un impedimento consistente in termini di programmazione o un ridimensionamento delle potenzialità del software da sviluppare.

4.2.3 La soluzione: la conversione dei dati di dump

Chiaramente il dispaccio Korg sull'implementazione MIDI dell'MS2000 [4] si riferisce a un formato file diverso da quello che è processato internamente dallo strumento

[5] Si intende la scelta delle giuste porte MIDI in base al cablaggio dell'interfaccia Edirol UM-2 (descritta in 3.1.2) e la preparazione degli strumenti (descritta in 4.1).

[6] "Confronta" permette di rilevare ogni byte di differenza tra due file presi in esame.

[7] Solitamente si ripete due volte: una per il valore massimo e una per quello minimo.

stesso. È stata effettuata una consultazione di altre "MIDI implementation charts" Korg, riferite ad altri modelli di sintetizzatore, per trovare una conferma e una soluzione al problema. Questa è contenuta in un passaggio della documentazione del synth KRONOS[9]:

Internal data is converted to MIDI data using following method.

Internal data (bit image)	**MIDI data (bit image)**
Aaaaaaaa	0GFEDCBA
Bbbbbbbb	0aaaaaaa
Cccccccc	0bbbbbbb
Dddddddd	0ccccccc
Eeeeeeee	0ddddddd
Ffffffff	0eeeeeee
Gggggggg	0fffffff
Hhhhhhhh	0ggggggg
Iiiiiiii	0NMLKJIH
[...]	0hhhhhhh
[...]	[...]
Vvvvvvvv	000000WV
Wwwwwwww	0vvvvvvv
-	0wwwwwww
-	11110111 (EOX=F7H)

Tabella 4.3: Internal 7byte data <convert> MIDI 8 byte data.

da questa chiara rappresentazione si deduce che i dati interni dell'MS2000 e i dati MIDI che vengono trasferiti, hanno due formati differenti:

Dati Interni sfruttano tutti i bit a disposizione per ogni byte. Sono quelli rappresentati in [4].

Dati Midi il bit più significativo è sempre '0'. Questo lascia solo i restanti 7 bit per la rappresentazione dei dati effettivi.

Questa discrepanza richiede una riorganizzazione dei pacchetti di bit per ogni byte. Osservando la tabella si nota che la conversione da dato interno a MIDI si ottiene raggruppando tutti i MSB dei primi 7 byte dei dati interni. Essi compongono il primo Byte dei dati MIDI, a partire dal LSB. I bit restanti dei 7 byte dei dati interni

sono copiati esattamente nei 7 byte seguenti dei dati MIDI. Così facendo si ottengono 8 byte per ogni 7 byte, rendendo le dimensioni del file MIDI sensibilmente maggiori.

4.2.4 Struttura finale del SysEx in formato interno

La riconversione del SysEx in formato "dati interni" sarà eseguita automaticamente da MS2000 Unfolder (l'algoritmo è descritto in 4.3.2.3) e permetterà di seguire la mappatura ufficiale Korg (Appendice A).

Ogni sound program è composto da 254 byte, quindi la dimensione del file che sarà processato dal programma è di 32512 byte.

Byte Number	**Program Name**
000 ~ 253	A01
254 ~ 507	A02
508 ~ 761	A03
[...]	[...]
32004 ~ 32257	H15
32258 ~ 32511	H16

Tabella 4.4: Composizione del data dump "All Programs"

4.3 Architettura del software

L'ambiente di sviluppo .NET e, in particolare, l'utilizzo di Microsoft Visual Studio, permettono di costruire programmi passo a passo, senza separare nettamente la parte di scrittura del codice da quella di progettazione dell'interfaccia grafica. Nonostante questa efficiente integrazione, le due parti di codice rimarranno comunque separate. In questa sezione saranno descritte singolarmente, ponendo particolare attenzione alle interazioni tra loro.

4.3.1 Interfaccia grafica e funzionalità

La progettazione grafica del software si appoggerà alla libreria WPF. Quest'ultima mette a disposizione numerose funzionalità di sviluppo, incluso un modello dell'applicazione, le risorse, i controlli, la grafica, il layout, l'associazione dati, i documenti e la sicurezza.

Il design del software dovrà essere subordinato alle funzionalità, quindi sarà necessariamente chiaro ed essenziale.

4.3.1.1 La finestra principale

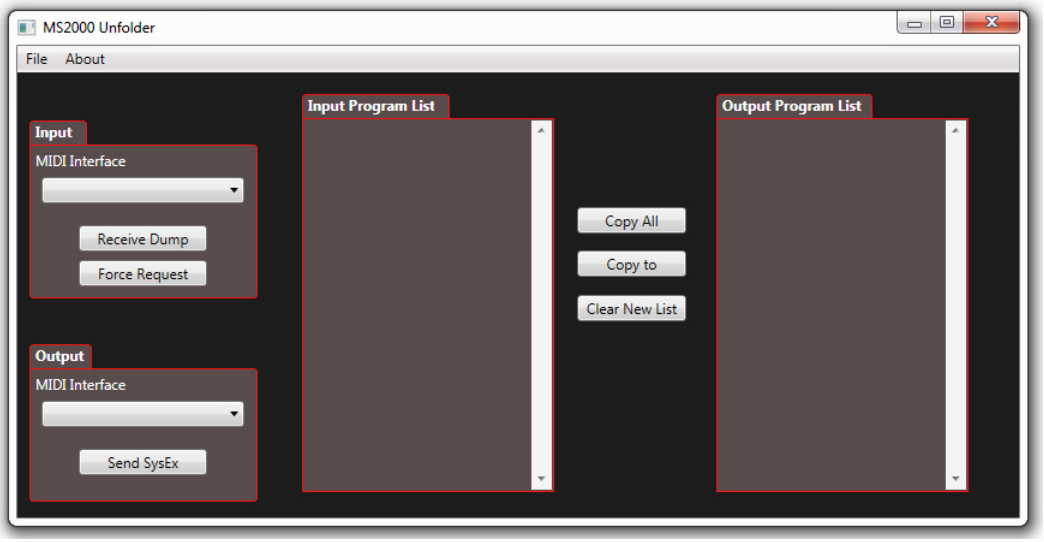

Figura 4.1: Schermata principale di MS2000 Unfolder.

La schermata principale permette la selezione dell'interfaccia MIDI che si intende utilizzare per la trasmissione. Viene concessa la possibilità di scegliere due interfacce

distinte per la fase d'invio e quella di ricezione. I differenti canali input/output saranno aggiunti alla lista automaticamente.

Una volta ottenuto il dump SysEx, vengono compilate le due liste con i Program Name di tutti i banchi di suoni disponibili. A ciascuno è assegnato un codice di posizione[8] che rende possibile ordinare i vari suoni a proprio piacimento all'interno del SysEx. Le due liste corrispondono al SysEx in entrata (sinistra) e al SysEx pronto per essere trasferito al synth o salvato in locale alla fine delle modifiche.

Attraverso il menù a tendina nella parte superiore della finestra principale è possibile caricare un file SysEx salvato sul proprio computer oppure esportare il risultato del proprio lavoro.

Elementi grafici WPF necessari

List-box Sono i due recipienti principali per la lista dei banchi di suoni, una per i dati in entrata, l'altra per quelli in uscita. Al momento dell'arrivo del SysEx, sia esso proveniente dall'interfaccia MIDI o da locale, sono riempite entrambe. Questo avviene solo in caso di list-box/input vuota. Se verrà caricato un ulteriore SysEx, il contenuto della prima list-box sarà sovrascritto, permettendo così la gestione di due SysEx diversi in simultanea.

Button Sono stati utilizzati per dare inizio alla ricezione dei dump, all'invio del SysEx finale e per la gestione dell'ordine dei suoni all'interno delle list-box. Sono dei componenti fondamentali di WPF, come ben spiegato in [10]:

> A basic button is a content control that can be clicked but not double-clicked. This behavior is actually captured by an abstract class called ButtonBase, from which a few different controls are derived. The ButtonBase class contains the Click event and contains the logic that defines what it means to be clicked. As with typical Windows buttons, a click can occur from a mouse's left button being pressed down and then let up or from the keyboard with Enter or spacebar, if the button has focus.

Combo-box Elementi in grado di permettere la scelta di un elemento all'interno di una lista. Sono stati impiegati per visualizzare le interfacce MIDI disponibili.

[8] I codici di posizione vanno da A01 ~ A16 a H01 ~ H16, esattamente come ordinati all'interno del sintetizzatore.

Rectangle Elemento puramente stilistico, non permette l'interazione e non ha alcuna funzionalità attiva.

Label Utilizzate per l'etichettatura e la scrittura di testo non modificabile dall'utente.

Grid Funzionalità di struttura, fa parte del set di pannelli messi a disposizione da WPF (vedi [13] e [10]):

> DataGrid: The DataGrid is WPF's most full-featured data display tool. It divides your data into a grid of columns and rows, like the ListView, but has additional formatting features (such as the ability to freeze columns and style individual rows), and it supports in-place data editing.
>
> [...]
>
> Grid is the most versatile panel and probably the one you'll use most often. (Visual Studio and Expression Blend use Grid by default for their projects.) It enables you to arrange its children in a multirow and multicolumn fashion, without relying on wrapping (like WrapPanel), and it provides a number of features to control the rows and columns in interesting ways. Working with Grid is a lot like working with a TABLE in HTML.

4.3.1.2 La finestra Sound Explorer

Cliccando su uno degli elementi contenuti nelle list-box, si avvia la lettura dei dati relativi al suono selezionato. Tramite l'apertura di una nuova finestra, denominata "Sound Explorer" queste informazioni sono rese disponibili alla lettura e alla modifica. L'organizzazione della finestra ricalca quella del Sound Program stesso, descritta ampiamente in 2.1.2. Tutte le tab hanno in comune due bottoni che permettono il ritorno alla schermata principale, attraverso il salvataggio o l'annullamento delle eventuali modifiche.

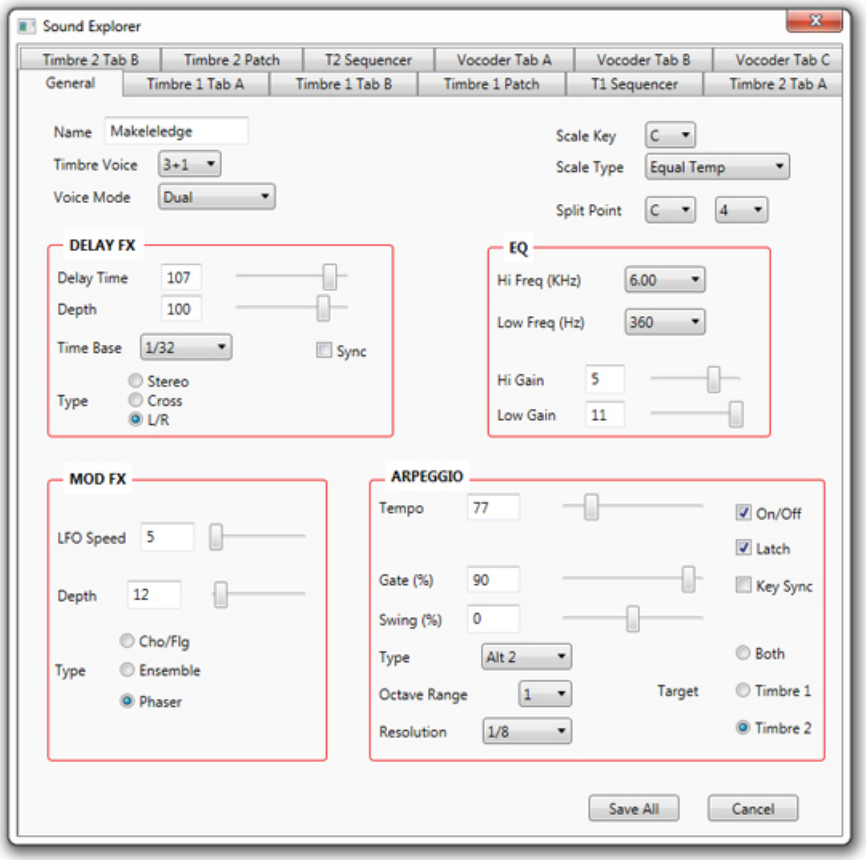

Figura 4.2: Schermata Sound Explorer, general.

La schermata "General" è l'unica che non sarà mai disattivata, in quanto sempre presente nel corpo dati del SysEx. All'interno di questa sezione si trovano i parametri comuni ai 128 suoni:

- Sound Name
- Timbre Voice
- Voice Mode
- Scale Key and Type
- Split Point

In aggiunta, si potranno modificare i valori di quattro macro-categorie:

- Delay
- Arpeggiatore

- Mod
- EQ

A seconda del voice mode selezionato, solo certi gruppi di tab saranno attivi. Lo schema sottostante illustra le regole di visualizzazione a proposito:

	Single	Split / Dual	Vocoder
General	✓	✓	✓
Timbre 1 Tab A	✓	✓	
Timbre 1 Tab B	✓	✓	
Timbre 1 Patch	✓	✓	
T1 Sequencer	✓	✓	
Timbre 2 Tab A		✓	
Timbre 2 Tab B		✓	
Timbre 2 Patch		✓	
T2 Sequencer		✓	
Vocoder Tab A			✓
Vocoder Tab B			✓
Vocoder Tab C			✓

Tabella 4.5: Tabella riassuntiva della modalità di attivazione tab.

Le tab relative al Timbro 1 e al Timbro 2 saranno identiche, in quanto i parametri di un timbro sono standard. Esse comprenderanno:

- MIDI Channel
- Key Priority
- Trigger Mode
- Assign Mode

e altri parametri specifici raggruppati in macro-categorie:

- Pitch
- Mixer
- Oscillator 1

- Oscillator 2
- Filter
- Amp
- Envelope Generator 1
- Envelope Generator 2
- Low Frequency Oscillator 1
- Low Frequency Oscillator 2
- Patch
- Sequencer

Le tre tab relative al Vocoder sono molto simili a quelle dei timbri. Non ci sono le parti che riguardano patch e sequencer, ma presentano dei controlli per modificare l'audio in entrata (Audio In 2) e il panning. Inoltre è presente solo una sezione "Oscillatore", al posto di due.

Elementi grafici WPF necessari Oltre agli elementi già citati in 4.3.1.1, sono stati utilizzati:

Radio Button Elemento utile per offrire una scelta tra due, massimo tre, possibilità. La selezione è resa univoca utilizzando il raggruppamento[14]:

> Since RadioButtons are meant to be used in groups, it's important to know how to group them. By default, all RadioButtons in a parent element belong to the same group. You can explicitly create RadioButton groups by setting the GroupName properties of the buttons. For each RadioButton, you can set its GroupName property to a string. All the buttons with the same GroupName string belong to the same group.

CheckBox Ogni volta che l'utente fa clic su un controllo CheckBox, questo si alterna tra gli stati checked e unchecked, abilitando o disabilitando la proprietà "IsChecked". Visualizza tale stato visivamente mostrando una piccola casella, con o senza un segno di spunta.

Slider Permetteranno di modificare i parametri con valori numerici[9], tramite il trascinamento di un "pomello", chiamato *thumb*, lungo il controllo. Di solito sono collegati dinamicamente a delle TextBox, tramite il "Data Binding" [16]:

> WPF has an extremely rich data binding model. It revolves around the notion that you can take almost any object as your binding source and bind it to almost any target UI element. The binding source can be another UI element, a property of the same element, an XML file, a custom business object, a database, or an in-memory collection. The binding target can be a WPF property, an individual UI element, or a WPF user control or window. But the essential idea is that once a binding is established, the data in the source is automatically and dynamically propagated to the binding target, and vice versa.

TextBox Visualizzano i valori dei parametri che prevedono un tipo di dato numerico. Inoltre è possibile digitare direttamente il valore desiderato. Non ci sarà il rischio d'immissione di valori fuori range[10] e caratteri non-numerici[11] (vedi 4.3.2.5).

4.3.2 La struttura del codice

Il linguaggio di programmazione scelto è C#. I fattori principali di questa scelta sono stati la chiarezza del linguaggio e la presenza di strumenti molto utili per il fine da raggiungere (ad esempio le classi e le strutture).

4.3.2.1 Il primo passo: la connessione tra i dispositivi

Il primo obiettivo da raggiungere era riuscire a mettere in comunicazione il software con il synth per consentire il dump dei dati. Per fare questo, è stato necessario rilevare la presenza (o meno) di una o più interfacce MIDI e presentarne una lista selezionabile. Come visto in figura 4.1, questa lista è contenuta da due ComboBox, una per le periferiche d'ingresso e una per quelle dedicate all'output.

[9] Solitamente da 0 a 127, oppure da -63 a +63.

[10] Il meccanismo di Data Binding visto in 4.3.1.2 lega ciascuna TextBox al proprio Slider. Questi accettano un determinato range di valori, quindi, se sarà immesso un valore fuori range nella TextBox, verrà automaticamente corretto nel valore massimo previsto dallo slider associato.

[11] In alcuni non è stato possibile effettuare questo tipo di controllo. Il carattere "-", necessario per indicare valori negativi, sarebbe rilevato come input "non numerico" e quindi bloccato.

L'implementazione delle librerie Sanford ha permesso di scrivere codice che possa interfacciarsi ottimamente con le risorse MIDI. In particolare, è stato necessario importare le seguenti librerie:

- Sanford.Collections
- Sanford.Multimedia.Midi
- Sanford.Multimedia.Midi.UI
- Sanford.Threading
- Sanford.Multimedia.Timers

```
private void Window_Loaded(object sender, RoutedEventArgs e)
{
    int nDevices = Sanford.Multimedia.Midi.InputDevice.DeviceCount;
    if (nDevices == 0)
    {
        MessageBox.Show("No MIDI devices available.", "Warning",
           MessageBoxButton.OK, MessageBoxImage.Warning);
    }

    for (int i = 0; i < nDevices; i++)
    {
        MidiInCaps mic = Sanford.Multimedia.Midi.InputDevice.
           GetDeviceCapabilities(i);
        comboBoxMIDIDevices.Items.Add(mic.name);
    }

    comboBoxMIDIDevices.SelectedIndex = 0;
    createInputDevice(0);
    int nOutDevices = Sanford.Multimedia.Midi.OutputDevice.
       DeviceCount;

    for (int i = 0; i < nOutDevices; i++)
    {
        MidiOutCaps mic2 = Sanford.Multimedia.Midi.OutputDevice.
           GetDeviceCapabilities(i);
        comboBoxMIDIDevices2.Items.Add(mic2.name);
    }

    comboBoxMIDIDevices2.SelectedIndex = 0;
    createOutputDevice(0);
```

```
    inDevice.SysExMessageReceived += HandleSysExMessageReceived;
}
```

Nel segmento di codice riportato viene descritto il procedimento di identificazione e aggiunta delle periferiche. All'apertura del programma viene effettuato un conteggio dei MIDI devices collegati al proprio PC. Nel caso non ce ne siano, la visualizzazione della schermata principale viene preceduta da una MessageBox che avverte l'utente di questa mancanza. Viceversa, per ogni interfaccia MIDI presente, è richiamata la funzione "GetDeviceCapabilities". Questa ne rileva le specifiche e le usa per compilare la struttura standard di ogni periferica MIDI input/output, ovvero "MidiInCaps" e "MidiOutCaps". I loro attributi "name" compilano le liste di oggetti delle ComboBox nella finestra principale.

Ogni volta che l'utente selezionerà una diversa interfaccia (o un diverso canale della stessa interfaccia), sarà creata una nuova "InputDevice" o "OutputDevice" a cui verrà fatto riferimento al momento della ricezione o dell'invio di dati.

4.3.2.2 Ricezione del MIDI Dump

Una volta stabilita la connessione, è possibile iniziare il trasferimento del SysEx. Esistono tre modi differenti per caricare quel tipo di file:

Ricezione Standard È l'opzione tradizionale, incontrata anche nei software già esistenti e attivabile tramite il tasto "Receive Dump". Dopo il click dell'utente, MS2000 Unfolder si metterà solo "in ascolto", attendendo l'arrivo di dati utili. Questo evento è scandito da una funzionalità molto utile messa a disposizione dal MIDI Toolkit di Sanford, ossia il riconoscimento automatico del tipo di messaggio in arrivo. Tramite l'evento "SysExMessageEventArgs", il programma distingue l'inizio e la fine di un messaggio System-Exclusive (come descritto nella tabella 4.2). Esso rimane in memoria per poter essere processato nella fase successiva. Alla fine del trasferimento, si visualizza una MessageBox che avverte l'utente del successo dell'operazione.

Force Request Differisce dalla ricezione standard solo nel primo passo. Se questa può essere considerata una modalità "passiva", in quanto il software si mette in ascolto ed è necessario avviare il dump dal terminale del synth, il force request è sicuramente "attiva". Cliccando sull'omonimo tasto, il programma invia un breve messaggio contenente una richiesta di dump. L'MS2000 riconoscerà il

formato particolare del messaggio e avvierà automaticamente il trasferimento. Questa funzione potrebbe rivelarsi molto utile in caso di malfunzionamento del terminale del synth.

Byte (Hex)	**Significato**
F0	Exclusive Status
42	Korg ID
30	Global Midi Channel (3n)
58	Modello Strumento (MS2000)
1C	Dump Request ID
F7	End of SysEx

Tabella 4.6: Formato del messaggio di richiesta Dump.

Open Sarà possibile importare un SysEx salvato sul proprio computer. In questo caso, per riconoscere la validità del file, non è utilizzato alcun meccanismo proveniente dalla libreria Sanford. Tutto il codice relativo all'apertura e analisi dei file è stato racchiuso in un Try-Catch, da [17]:

> *Try blocks* encapsulate the code that forms part of the normal operation of your program and that might encounter some serious error conditions.
>
> *Catch block*s encapsulate the code dealing with the various error conditions that your code might have encountered by working through any of the code in the accompanying try block. This block could also be used for logging errors.
>
> *Finally blocks* encapsulate the code that cleans up any resources or takes any other action that you normally want handled at the end of a try or catch block. It is important to understand that the finally block is executed whether or not an exception is thrown. Because the purpose of the finally block is to contain cleanup code that should always be executed, the compiler will flag an error if you place a return statement inside a finally block. An example of using the finally block is closing any connections that were opened in the try block. Understand that the finally block is completely optional. If your application does not require

any cleanup code (such as disposing of or closing any open objects), then there is no need for this block.

Si possono aprire solo file con due estensioni: .syx (SysEx) o .mid (MIDI)[12]. Se i file scelti risultano corrotti, in un formato errato o non validi, è generato un errore. Questo è intercettato dall'istruzione "catch", che visualizza il seguente messaggio:

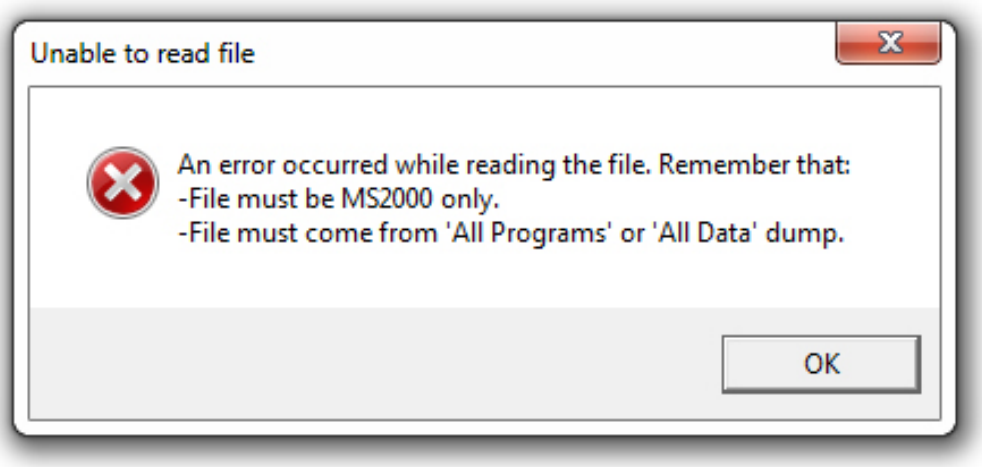

Figure 4.3: Il messaggio di errore in lettura.

4.3.2.3 Gestione del SysEx

L'algoritmo di conversione Il pacchetto di dati ricevuto (o caricato) è in formato "SysexMessage", quindi è un Array di 37163 Byte.

Questa matrice è sottoposta a un processo di conversione (la motivazione è descritta in 4.2.3) e, vista la sua natura, è necessaria la creazione di un nuovo Array, scrivendoci i valori convertiti passo per passo.

[12]Talvolta i file SystemExclusive per l'MS2000 sono resi disponibili nel formato MIDI. Questo non è lo standard preferito dallo strumento musicale, ma è stata comunque aggiunta una funzionalità che ne permette la lettura.

Il codice seguente descrive questo processo:

```
1  byte[] newarr = new byte[32512];
2
3  for (int j = 0; j<=4644; j++)
4      {
5       for (int i = 0; 0 <= i && i <= 6; i++)
6          {
7              if ((8 * j + i + 6) < 37162)
8               {
9             byte b = Convert.ToByte((e.Message[8 * j + 5] & Convert.
                  ToByte(Math.Pow(2, i))) << (Convert.ToByte(7 - i)) | (
                  e.Message[8 * j + i + 6]));
10
11                  newarr[7 * j + i] = b;
12             }
13             else
14               i = 10;
15         }
16     }
```

È stato necessario l'uso di una combinazione tra istruzioni decisionali e di iterazione, in questo caso l'istruzione *if* e il ciclo *for*.

Il ciclo for più esterno funge da contatore di iterazione: sappiamo che c'è un determinato numero di pacchetti da 8byteX7bit da convertire, usando questo ciclo è possibile evitare errori di compilazione. All'interno troveremo un altro ciclo for che si occupa di incrementare a ogni ciclo la posizione del byte da sottoporre alla conversione. Quest'operazione è effettuata, in definitiva, nella parte più interna di codice, quella relativa all'istruzione if.

Il cuore di tutto questo processo si trova alla linea 9 del listato precedente. Questo algoritmo sfrutta due operatori booleani (*And* e *Or*) e l'operatore di shift[13]. È selezionato il Byte contenente tutti i bit più significativi, che, a loro volta, sono presi a uno a uno tramite un *And* con una potenza di 2 che incrementa ad ogni iterazione. Questi singoli bit sostituiscono i MSB dei Byte seguenti, permettendo così la ricostruzione del SysEx "corretto".

Tutto il ciclo di conversione provvede alla rimozione dei Byte che non fanno parte del "corpo dati" utile per la modifica dei parametri, ossia i primi cinque e l'ultimo.

[13]L'operatore di spostamento a sinistra (<<) sposta il primo operando verso sinistra in base al numero di bit specificati nel secondo operando. Il tipo del secondo operando deve essere int.

La divisione dei 128 sound program Ottenuto un SysEx ordinato, si prosegue con la divisione in parti uguali dello stesso. L'Array principale da 32512 Byte genera 128 Array da 254 Byte ciascuno. Per mantenere l'ordine e per facilitare le operazioni successive (in particolare quelle descritte in 4.3.2.4), sono state utilizzate le SortedList [12]:

> The System.Collections namespace in the .NET contains data structures that can be used in C# to define different kinds of collection types. All collection classes implement the interface ICollection with additional functionality. [...]
>
> SortedList represents a collection of key and value pairs that are sorted by the keys and are accessible by key and by index.

La coppia chiave-valore è formata rispettivamente dal codice di posizione ("A01", "A02", ecc.) e dalla Struct "suono", descritta nella sottosezione seguente. Contemporaneamente, il ciclo popola le ListBox della finestra principale.

4.3.2.4 La struttura dati di un sound program

In questa fase ogni Array da 254 Byte corrisponde perfettamente allo schema descritto nella mappatura ufficiale (Appendice A), quindi si è provveduto all'estrazione mirata delle informazioni utili per la modifica dei parametri. Per fare ciò ci si è appoggiati a un particolare tipo valore, la *Struct*. Questa permette il raggruppamento di un largo numero di variabili correlate tra loro, rivelandosi uno strumento perfetto per rappresentare lo "scheletro" di un Sound Program.

La Struct, denominata "Sound", è formata da un numero di elementi pari a quello dei parametri modificabili (circa 450 in tutto). Ogni elemento contiene un'istruzione di *Get* e una di *Set*. La prima, quando invocata, estrae dall'Array il byte corrispondente al parametro d'interesse. La seconda, una volta modificato un parametro e salvate le impostazioni, ne scrive il valore nell'opportuna posizione all'interno dell'Array.

Talvolta in un Byte possono trovare informazioni di stato di più parametri (ad esempio nel Byte 16 riportato nella Table 1, Appendice A). In quel caso è necessario operare su singoli bit o su gruppi ristretti.

Di seguito, un esempio di variabile associata a un parametro (Sequencer Resolution):

```
public int seqRes
    {
        get
        {
            string ByteString = Convert.ToString(arr[90], 2).PadLeft
               (8, '0');
            int i = Convert.ToInt32(ByteString.Substring(3, 5), 2);
            return i;
        }

        set
        {
            string s = Convert.ToString(Convert.ToByte(value), 2).
               PadLeft(5, '0');
            string o = Convert.ToString(arr[90], 2).PadLeft(8, '0');
            o = o.Substring(0, 3);
            string t = o + s;
            arr[90] = Convert.ToByte(t,2);
        }
    }
```

Tutte le variabili sono di tipo *Intero* o *Booleano*. Il primo è il più diffuso, siccome molti dei parametri modificabili possono assumere un cero range di valori numerici. Il tipo *Bool* è invece adatto per i controlli come le CheckBox[14].

Le istruzioni di *Set* riceveranno sempre questo tipo di dati, quindi dovranno necessariamente convertire tale valore in Byte prima di copiarlo all'interno dell'Array.

4.3.2.5 La connessione tra schermata principale e finestra "Sound Explorer"

Con le ListBox della schermata principale ormai popolate, sarà possibile interagire con gli elementi in lista. Con un doppio-click sul nome del suono che si desidera modificare, si aprirà la finestra Sound Explorer. La composizione grafica e gli elementi contenuti sono già stati descritti in 4.3.1.2. Le figure delle schermate sono

[14] Questo tipo di controlli è basato sulla proprietà "Is.Checked". Se tale valore sarà *true*, allora la casella verrà spuntata. In caso contrario, rimarrà vuota.

consultabili nell'Appendice B. In questa sottosezione saranno descritte le caratteristiche interessanti (a livello di programmazione) degli elementi contenuti in Sound Explorer.

Sound Name Il primo elemento che balza agli occhi, contenuto nella tab *General*, è il nome del suono. Questo è riportato all'interno di una TextBox ed è limitato a un massimo di 12 caratteri ASCII. Per evitare problemi durante la fase di scrittura nell'Array in output, è doveroso limitare il numero di caratteri a 12. Nel caso di nomi più brevi, si aggiungono degli spazi fino a raggiungere la corretta quantità di caratteri:

```
sound.name = name.Text.PadRight(12, ' ');
```

Parametri fissi a scelta multipla Molti dei parametri non permettono l'immissione diretta di testo, bensì è possibile scegliere tra un set di opzioni ben definito. Questo modello si applica a tutte le ComboBox presenti. Gli *Items* sono caricati tramite la funzionalità di binding *ItemSource* [18]:

> Controls, such as ListBox, ComboBox, and TreeView, are considered item controls. These display a repeating sequence of values stored in collections, arrays, trees, and other repetitive data structures. The values can be simple things such as numbers or strings, or they can be more complex objects such as instances of the Order or Employee classes. Binding content controls is relatively straightforward. Set the binding's target, target property, source, and path; and you're done. On the surface, binding item controls is just as easy. Just set the control's ItemsSource property to some sort of collection that holds repeating values. If the values are simple numbers or strings, this works.

Nel caso studiato in questo elaborato, si è preferito l'utilizzo di Array di Stringhe o Interi.

Parametri numerici modificabili La maggior parte dei parametri interni dell'MS2000 rientra in questo gruppo. Come si può dedurre dal pannello frontale dello strumento, ci sono molti parametri che possono assumere tutti i valori all'interno di un certo range.

Range	Tipo di controllo
0 ~ 127	Non possono essere < 0 (delay time, depth ecc.)
-63 ~ +63	Possono assumere valori < 0 (Filtri, step del sequencer, patch intensity)
-24 ~ +24	Vocoder Transpose
-50 ~ +50	Tune [Cent]
-12 ~ +12	Band Range
0 ~ 100	Valori percentuali
20 ~ 300	Global Time (BPM)

Tabella 4.7: Tipi di parametri numerici modificabili.

Questo tipo di valori è visualizzato tramite l'accoppiata *TextBox + Slider*. L'utente sarà in grado di modificare il parametro trascinando il cursore lungo lo slider oppure immettendo direttamente il valore desiderato all'interno della TextBox. Anche in questo caso è stato necessario utilizzare il Data Binding tra i due elementi, per controllare che i valori immessi nella TextBox non fossero fuori scala. Il Binding si è rivelato utile anche nel risparmio di risorse computazionali, giacché l'istruzione di get di una variabile della struct Sound è richiamata una sola volta ed è visualizzata in due controlli diversi.

Inoltre, È stato necessario implementare del codice che non permettesse la digitazione di caratteri non numerici all'interno della TextBox. Un esempio è riportato in [7]:

> The best way to correct invalid input is to prevent it from being entered in the first place. This approach is easy to implement with the text box because it provides a KeyPress event that occurs after the keystroke has been received but before it's displayed. You can use the KeyPressEventArgs event parameter to effectively "cancel" an invalid keystroke by setting the Handled property to true.
>
> To allow only numeric input, you must allow a keystroke only if it corresponds to a number (0 through 9) or a special control key (such as delete or the arrow keys). The keystroke character is provided to the KeyPress event through the KeyPressEventArgs.KeyChar property. You can use two static methods of the System.Char class—IsDigit and IsControl—to quickly test the character.

Il codice implementato è il seguente:

```
private void TextBox_PreviewTextInput(object sender,
    TextCompositionEventArgs e)
    {
        e.Handled = !AreAllValidNumericChars(e.Text);
    }

    private bool AreAllValidNumericChars(string str)
    {
        foreach (char c in str)
        {
            if (!Char.IsNumber(c)) return false;
        }

        return true;
    }
```

Vista la difficoltà a regolare l'uso del carattere '-', necessario per i numeri negativi, le TextBox contenenti tali valori sono stati resi fruibili solo in lettura.

4.3.2.6 Modifica dei parametri e salvataggio

Ogni volta che un parametro è modificato, i nuovi valori sono mantenuti in visualizzazione in qualsiasi tab della finestra Sound Explorer. Nel caso venga chiusa, le modifiche sono annullate, tornando così al punto di partenza. Questo perché le istruzioni di set delle variabili della struct "Sound" sono invocate soltanto al click di un determinato tasto. Il button in questione è il *Save*, locato in fondo a ogni tab. È obbligatorio invocare tutte le istruzioni di set anche se si modifica un solo parametro. Questo per evitare problemi con la scrittura in posizioni dell'Array che presentano un singolo Byte per più parametri.

Un altro accorgimento necessario prima del salvataggio è controllare il valore del parametro "Voice Mode" (vedi Tabella 4.5). Siccome, all'interno del SysEx, i valori che si riferiscono alla modalità Vocoder occupano la stessa posizione di quelli della modalità Timbro, si deve decidere (attraverso un If) quale gruppo di istruzioni set invocare.

Una volta salvate le proprie modifiche, si torna alla schermata principale e la ListBox contenente il suono modificato viene aggiornata, così da visualizzare immediatamente il nuovo (in caso di modifica) Sound Name.

4.3.2.7 Ordinamento dei banchi di suoni

Dalla schermata principale sarà possibile organizzare a piacere l'ordine dei Sound Program all'interno del SysEx. La ListBox che è salvata o trasmessa al synth è quella destra (output program list). Ci sono tre controlli che permetteranno l'interazione tra le due liste:

1. Copy All: copia l'intero contenuto dell'input program list all'interno dell'output program list.

2. Copy To: permette la copia di un singolo sound program. Per funzionare devono essere precedentemente selezionati due ListBoxItems, uno da copiare (lista sinistra) e uno su cui sovrascrivere (lista destra)

3. Clear New: cancella il contenuto della list box in output.

Tutte queste funzioni non operano solo in superficie, modificando la lista degli items nelle ListBox, bensì a livello profondo, modificando gli Array dei sound program.

Il codice seguente descrive il metodo FillOutputList, invocato durante la copia di tutti i banchi:

```
public void FillOutputList()
   {
       splitted2 = new SortedList<string, Sound>();
       char lettera = 'a';
       int num = 1;
       string id = "";
       while (lettera < 'i')
       {
         if (num < 17)
           {
               Sound s = new Sound();
               id = lettera + num.ToString("D2");
               s.arr = (byte[])splitted[id].arr.Clone();
               splitted2[id] = s;
               pnBox2.Items.Add(id + ": " + splitted2[id].name);
               num++;
           }
           else
           {
               num = 1;
               lettera++;
           }
```

```
        }
    }
```

4.3.2.8 Ricostruzione del SysEx nel formato MIDI

Il passo che precede il salvataggio del proprio lavoro o l'avvio della trasmissione in uscita dei dati, è dedicato alla ricostruzione del file SysEx e alla sua riconversione.

Questo compito è stato affidato al metodo "BuildFinalSysEx" (Appendice C) che opera in diverse fasi:

1. Assembla l'Array da convertire (chiamato "arrayRicombinato"), estraendo la struttura Sound da ciascun elemento dalla SortedList e copiandone gli Array.
2. Crea un nuovo Array (sysExFinale) impostandone la dimensione (37163 Byte).
3. Ricostruisce i Byte contenenti tutti i MSB dei 7 seguenti (vedi 4.2.3) e li scrive in sysExFinale.
4. Recupera e inserisce in sysExFinale i restanti Byte.
5. Aggiunge i Byte esclusivi dello standard MIDI (vedi Tabella 4.2).

Un passaggio interessante di questo metodo è rappresentato nell'Appendice C, righe 38 / 55: dopo aver ricostruito il Byte desiderato, esso è copiato all'interno dell'Array "sysExFinale". Appena prima avviene un controllo che evita di sovrascrivere i valori immessi in precedenza (i Byte con tutti i MSB). Questi si trovano in posizioni determinate (ogni 7 Byte), quindi si utilizza un'istruzione If: il numero di posizione in cui si deve scrivere il Byte in questione è diviso per 8 e, se il resto è uguale a 0, tale posizione è scartata in favore della seguente. Questo check è possibile grazie all'operatore di resto "%" [19]:

> C# supports the regular arithmetic operations you learned in your childhood, but not all operators are applicable to all data types. The operators that you can use on a value depend on the value's type.
>
> C# also supports one less-familiar arithmetic operator: the remainder, or modulus, operator, which is represented by the percent sign (%). The result of x % y is the remainder after dividing the value x by the value y. So, for example, 9 % 2 is 1 because 9 divided by 2 is 4, remainder 1.

4.3.2.9 Salvataggio e trasmissione

L'ultimo step possibile è il salvataggio del file SystemExclusive. Può avvenire in locale sul proprio computer, oppure direttamente nel sintetizzatore. La prima soluzione si ottiene cliccando "Save", attraverso il menù a tendina "File" presente nella schermata principale. Il programma apre una DialogBox classica che permette la scelta del nome del file, la locazione del salvataggio, ma non il formato (questo sarà obbligatoriamente .syx).

In alternativa si può preferire la trasmissione via MIDI, caricando il SysEx all'interno dell'MS2000. Dalla schermata principale del programma, cliccando sul tasto "send sysex" nell'output box in basso a sinistra, si avvia il processo d'invio. Analogamente alla programmazione della ricezione del dump, sono state sfruttate le potenzialità della libreria Sanford per generare un "OutDevice" partendo dalle interfacce MIDI disponibili. A questo punto è invocata la funzione *Send* che dà inizio alla trasmissione. Al contrario, non si deve toccare nulla sul terminale del synth, ammesso si siano seguite le indicazioni in 4.1.

4.4 Testing e Debug

La fase di collaudo del software più importante ha riguardato le istruzioni di get e set delle variabili della struct Sound. È stata divisa in due momenti distinti, onde evitare complicazioni o falsi bug negli step successivi.

Una volta preparata l'interfaccia grafica della finestra Sound Explorer e terminata la scrittura della struct, è stata verificata la correttezza di tutte le istruzioni di get. Si è effettuata la trasmissione del SysEx dal synth e si sono confrontati i controlli visualizzati nel software con i parametri sull'MS2000. Date le diverse modalità d'estrazione dei bit utili a definire i valori dei parametri, è stato molto facile commettere qualche imprecisione durante la scrittura del codice. Anche le proprietà dei controlli WPF sono state passate al setaccio in caso d'incongruenze. Per esempio, la quantità di controlli Slider è elevata, ma non funzionano tutti allo stesso modo (come s'intuisce dalla tabella 4.7). Il tipo di valore che è processato da questi controlli è sempre Int, quindi trattare uno Slider da 0 ~ 127 come uno da -63 ~ +63 potrebbe non generare errori in fase di compilazione, ma porterebbe comunque alla visualizzazione di valori non aderenti alla realtà.

La seconda parte di testing della struct è avvenuta a programma completo. Dopo aver caricato il SysEx da locale, si apre la finestra Sound Explorer e si sceglie di salvare le modifiche. Non è necessario modificare i parametri perché il salvataggio

effettua comunque un repack del SysEx, richiamando tutte le istruzioni di set scritte in precedenza. Fatto ciò, è stato esportato il file in output ed è stato confrontato con l'originale tramite il programma HxD[15]. In questo modo è stato possibile effettuare un debug mirato, agendo soltanto sui byte effettivamente compromessi e non sull'intero array.

Un'altra parte importante del collaudo ha riguardato tutte le istruzioni di visualizzazione/attivazione delle tab e dei singoli controlli. I meccanismi che regolano le tab dovranno funzionare sia in fase di apertura della finestra Sound Explorer, sia al momento della modifica del VoiceMode. Allo stesso modo, i controlli in binding esclusivo[16] non dovranno creare problemi e dovranno sempre essere definiti.

Per quanto riguarda la schermata principale, ci si è assicurati che le modifiche agli elementi delle ListBox, compreso il processo di copia, agiscano sui valori effettivi (quindi a livello degli Array di Byte) e non per riferimento. Inoltre sono state controllate le regole di popolamento delle liste, impedendo che quella in output sia sovrascritta ogni volta che si apre o si riceve un nuovo SysEx.

[15]Durante questa fase il valore di alcuni byte potrebbe variare, ma si tratterebbe comunque di dati non rilevanti per la generazione del suono desiderato.

[16]Un esempio di questo tipo di binding si trova nella sezione Sequencer. il parametro "Run Mode" può essere definito da uno tra due RadioButton: OneShot e Loop. Tra questi, solo l'ultimo può essere sempre attivo. La modalità OneShot è disponibile soltanto se un altro parametro (KeySync) assumerà un valore diverso da "Off".

Capitolo 5

Conclusioni

5.1 Raggiungimento degli obiettivi

Lo spirito che ha pervaso ogni fase di questo lavoro sperimentale, dalla genesi dell'idea, sino alla fase di test, è stato quello dell'innovazione. L'obiettivo principale era quello di creare un software specifico che riuscisse a essere un passo avanti a quelli già esistenti.

La gestione dei dump del synth Korg MS2000 non è mai stata un punto fermo tra i programmi di sviluppo dell'azienda produttrice. Gli utenti "volenterosi" sono stati in grado di supplire a questa mancanza attraverso metodi scomodi e non alla portata di tutti. Chi possedeva la giusta attrezzatura e le conoscenze di base in ambito MIDI, poteva salvare sul proprio PC una serie di byte in arrivo dal synth, ma il processo era utile solo in ambito di archiviazione. L'altro uso possibile era la trasmissione dei banchi di suoni direttamente da un MS2000 all'altro, altro processo poco pratico. I primi passi avanti sono arrivati, com'era prevedibile, dagli utenti stessi. Poco tempo dopo l'uscita dello strumento musicale, sono nati i software che tuttora permettono la visualizzazione e l'ordinamento dei banchi di suoni contenuti nel file SysEx. Per anni la situazione è rimasta la stessa e la necessità di un'evoluzione in questo campo specifico si è fatta più urgente, soprattutto prendendo in considerazione lo stesso genere di software per synth sempre più moderni. Si è quindi fatto tesoro delle esperienze precedenti e delle necessità avvertite, per creare MS2000 Unfolder. Questo software si è rivelato più di un passo avanti rispetto ai predecessori, includendo funzionalità che finora non erano disponibili altrove.

L'obiettivo principale era quello di scardinare i vincoli legati alla complessità della gestione di un dump riferito a tutti i Sound Program dell'MS2000. Il lavoro è riuscito alla perfezione e non presenta problemi nella visualizzazione dei parametri

o nello store delle modifiche. La veduta dei parametri tramite le tab della finestra Sound Explorer aiuta molto la comprensione della natura del suono analizzato. In aggiunta, essa permette di rendersi conto molto più chiaramente delle potenzialità dello strumento, poiché dà una forma chiara a parametri che finora potevano essere modificati solo tramite lo schermo LCD limitato del synth. Il nome scelto per il software richiama proprio questo aspetto: *to unfold* rimanda al concetto di apertura, distensione, rivelazione completa di tutte le sfaccettature di un file System Exclusive.

A questa funzione ne sono state affiancate altre molto utili come il *Force Request*, con cui è possibile richiedere un dump direttamente dal software. Potrebbe rivelarsi vitale, ad esempio, in caso di malfunzionamenti al terminale LCD del sintetizzatore.

Sono state implementate anche caratteristiche non preventivate, cui si è pensato a sviluppo avviato. La possibilità di leggere impostazioni System Exclusive da file .mid non era stata presa in considerazione nella fase di studio del caso, ma più avanti ci si è resi conto che il MIDI è un formato alternativo di scambio SysEx, soprattutto online. I meccanismi di lettura/conversione dei file .mid sono leggermente diversi da quelli adottati per i .syx, quindi si è trovato un algoritmo efficace per gestirli a parte.

Il software sarà inizialmente distribuito come *freeware*[1] senza implementare il codice sorgente. Questa scelta è in linea con la politica di distribuzione adottata per software del genere. Se in futuro saranno implementate funzioni più elaborate e che andranno significativamente oltre l'ambito di questo lavoro, si prenderanno in considerazione altre forme di rilascio. Queste sarebbero comunque affiancate da una versione "base" del software sempre disponibile gratuitamente.

5.2 Perfezionamento

Sebbene tutti gli obiettivi principali siano stati raggiunti, rimangono alcune sottigliezze che potrebbero essere perfettibili o sviluppate meglio in futuro. Sicuramente il software vedrà nuove versioni, attraverso fix o anche patch maggiori che introducano nuove funzionalità. Il lavoro svolto è stato solo il primo passo e indica la direzione in cui ci si muoverà per migliorie e aggiunte future.

Per esempio, è stato deciso di concentrarsi unicamente su dump di tipo "All Program" o "All Data", ma ne esistono anche altri. Il dump "1Program" racchiude in sé solo i dati riguardanti il suono selezionato sul synth al momento della trasmissione dei dati. Riuscire a gestire questo tipo di dump permetterebbe una velocità maggiore nella customizzazione del proprio set di suoni. Il SysEx "Global" trasporta informa-

[1] Il termine freeware indica un software che viene distribuito in modo gratuito.

zioni riguardanti le impostazioni di sistema del synth, che non hanno a che fare con i Sound Program. Questo è uno dei motivi per cui non sono stati presi in considerazione, ma potrebbero esserlo per dare maggiore controllo all'utente. Gestire queste varianti avrebbe richiesto la scrittura di un blocco di codice a se stante, interfaccia grafica compresa, ma si sarebbe allontanato troppo dall'ambito dell'elaborato.

Un'altra funzione che avrebbe potuto rivelarsi utile è l'annullamento dell'ultima operazione compiuta (Undo). Questo è un aspetto da tenere presente prima di iniziare la fase di programmazione, ma per questo lavoro si è scelta un'altra via, quindi l'implementazione in corso sarebbe stata macchinosa, se non impossibile. Come spiegato piacevolmente in [21]:

> One of the biggest concerns with incremental development, such as we've been practicing in this book, is that there may be "cross-cutting" concerns that the design will not easily accommodate, resulting in a lot more work than if one had done "more" design. A favorite example, first raised to me by Gary Pollice of Rational, is "What if you're writing an editor, and you don't plan for Undo, and then they want it?" [...] Will Undo go in easily, or will it be very difficult, such that we will wish we had provided for it earlier? At this moment, I don't know.

Saranno sicuramente effettuati anche miglioramenti per quanto riguarda la grafica, anche se è generalmente considerato un aspetto di minore importanza in programmi di archiviazione e modifica dei dati. L'interfaccia è stata composta nel tempo, man mano che i parametri erano analizzati e descritti nella struct Sound. Si è mantenuta una linea che favorisse la chiarezza, anche se ciò ha portato a una concentrazione minore nei riguardi dell'estetica. Questo genere di software sviluppato dalla Moog per i propri strumenti, ad esempio, ricalca la grafica di un synth reale e l'effetto è molto appagante. L'avvento di un nuovo sistema operativo Microsoft[2] inoltre, potrebbe dare ulteriori nuove idee per il design.

5.3 Sviluppi futuri

Gli spunti per una ramificazione di questo software in ambiti e situazioni leggermente diverse non mancano. Prima di tutto si potrebbe espandere il bacino di utenza eseguendo un porting su altri sistemi operativi, come OS X o Linux. Le funzionalità

[2]Microsoft ha rilasciato ufficialmente Windows 8 il 26 ottobre 2012. Esso è il successore di Windows 7, sistema operativo utilizzato durante questo lavoro.

e i metodi per raggiungerle sono stati descritti nel dettaglio, quindi il lavoro sarebbe possibile. Più utenti saranno in grado di utilizzare un programma del genere, maggiore sarà l'interazione tra essi. Lo scambio e il confronto di System Exclusive per strumenti dello stesso tipo sono molto comuni, soprattutto online.

Un altro tipo di porting potrebbe essere mirato al cambio di hardware: il settore mobile è in piena espansione e un software così leggero potrebbe essere ben supportato. Questa idea si rivelerebbe particolarmente azzeccata per esibizioni in ambito live. Sebbene 128 Sound Program siano molti, si potrebbe aver bisogno di un suono particolare o di un intero set. Un'applicazione mobile faciliterebbe il cambio rapido delle impostazioni di sistema (midi channel, comportamento del pedale esterno, comportamento delle mod-wheel, ecc.).

La proiezione futura più stimolante, in tutti i casi, rimane quella circa un software che riesca a offrire le medesime funzionalità di MS2000 Unfolder, ma su larga scala. Adattare il paradigma su cui si basa quest'applicazione, rendendolo utilizzabile da altri modelli di sintetizzatore, persino di altre case produttrici. Questo richiederebbe uno sforzo importante a livello di programmazione, di grafica e di analisi. Esiste una miriade di sintetizzatori, anche solo in formato "classico", ossia con tastiera da pianoforte e vari controlli esterni per i parametri. Creare un unico editor/archivio sarebbe molto interessante sia per i vantaggi logistici, sia per lo studio e il confronto di strumenti diversi. Come "profeticamente" affermato nel 1995 in [22]:

> A good universal librarian should come preprogrammed with parameter settings for the synthesizers you use. The preprogrammed data should include the number of voices per bank and the system exclusive codes that identify the synthesizers, especially if those synthesizers are among the more popular models. You should also be able to program in the necessary data for synthesizers that the librarian does not implement. The librarian should be able to list all the synthesizer's voices on the computer screen and let you organize the voices into banks however you like. [...]
>
> Rather than buying separate patch editor programs, each with a different interface and unique commands, for each of the many instruments in a complex MIDI system, you may prefer a universal editor. Universal editors have a single interface and command structure but load templates for each MIDI instrument. Each template specifies an instrument's particular programming characteristics so that the universal editor can present editing controls appropriate for the instrument. (..)

> However, a universal editor/librarian may be excessively complex or lack features found in special-purpose programs.

Sono passati più di dieci anni ormai dal lancio dell'MS2000 e i musicisti che lo utilizzano sono ancora molti. Esso sembra si sia adattato all'ambiente circostante, resistendo alle mode musicali e riscoprendosi adatto a nuovi generi. Questo è accaduto per la quasi totalità dei sintetizzatori di successo prodotti dalla metà degli anni '80 in poi. In questo quadro, nel quale gli utenti, gli strumenti musicali e le tecnologie informatiche si adattano alle nuove situazioni, gli unici a non farlo sono stati i programmi gestionali. Soprattutto per i synth più vecchi, i software (se esistenti) sono rimasti sempre gli stessi. Nuovi sistemi operativi, nuove tecnologie e nuove necessità degli utenti, sembrano non averne influenzato lo sviluppo. L'unica tecnologia in questo ambito che sembra soffrire dello stesso immobilismo è lo standard MIDI stesso, ma questo è un discorso che sarebbe da affrontare in un'altra sede.

Bibliografia

[1] Korg MS2000 Owner's Manual. Korg, 2000.

[2] Moore, F.R.: The Dysfunctions of MIDI. Computer Music Journal, Vol. 12, No. l, Spring 1988.

[3] Korg DW-8000 Owner's Manual. Korg, 1985.

[4] Korg MS2000(R) MIDI Implementation Chart - Revision 1.1. Korg, 5/1/2000.

[5] Dimopoulos, H.: Analog Electronic Filters - Theory, Design and Synthesis. Springer, 2012.

[6] Moorer, J.A.: The Use of the Phase Vocoder in Computer Music Applications. Journal of the Audio Engineering Society, Volume 26, Number 1/2, January/February 1978.

[7] Jones, A.: C# Programmer's Cookbook. Microsoft Press, 2004.

[8] Huber, D.A.: The MIDI Manual - 3rd edition. Elsevier / Focal Press, 2007.

[9] Korg Kronos MIDI SysEx Chart - Version 1.07. Korg, 3/5/2011.

[10] Nathan, A.: WPF 4 Unleashed. Sams, 2010.

[11] Guérin, R.: Midi. L'interfaccia digitale per gli strumenti musicali. Apogeo Editore, 2003.

[12] Rahman, M.:Expert C# 5.0: with the .NET 4.5 Framework, Apress, 2012.

[13] MacDonald, M.: Pro WPF 4.5 in C# - Windows Presentation Foundation in .NET 4.5. Apress, 2012.

[14] Solis, D.M.: Illustrated WPF - Windows Presentation Foundation presented visually and concisely. Apress, 2009.

[15] Selfridge-Field, E.: Beyond Midi: The Handbook of Musical Codes. MIT Press, 1997.

[16] Noble, S., Bourton, S., Jones, A.: WPF Recipes in C# 2008 - A Problem-Solution Approach. Apress, 2008.

[17] Nagel, C., Evjen, B., Glynn, J., Watson, K., Skinner, M.: Professional C# 2012 and .NET 4.5. Wrox, 2012.

[18] Stephens, R.: WPF Programmer's Reference, Wrox, 2010.

[19] Sharp, J.: Microsoft Visual C# 2012 Step By Step, Microsoft Press, 2013.

[20] Heckroth, J.: Tutorial on MIDI and Music Synthesis. The MIDI Manufacturers Association, 2003.

[21] Jeffries, R.: Extreme Programming Adventures in C#. O'Reilly Media, 2009.

[22] Rothstein, J.: Midi - A Comprehensive Introduction, 2nd Edition. A-R Editions, 1995.

Appendice A

Mappatura di un Sound Program

TABLE 1 : PROGRAM PARAMETER (1 PROGRAM, CURRENT PROGRAM)

0~11	program name	ASCII code [0]~[15]=1st~12th
12~15	(not use)	
16 B6,7	Timbre Voice	0~2=1+3,2+2,3+1 (use Split/Dual Mode)
B4,5	Voice Mode	0~3=Single,Split,Layer,Vocoder
B0~3	not use	(0,0,0,0)
17 B4~7	Scale Key	0~11=C,C#,D,D#,E,F,F#,G,G#,A,A#,B
B0~3	Scale Type	0~9=Equal Temp~User Scale *T-1
18	Split Point (upper btm)	0~127=C-1~G9 (use Split Mode)
DELAY FX		
19 B7	Sync	0,1=Off,On
B4~6	not use	(0,0,0)
B0~3	Time Base	0~14=1/32~1/1 *T-2
20	Delay Time	0~127
21	Depth	0~127
22	Type	0~2=StereoDelay,CrossDelay, L/R Delay
MOD FX		
23	LFO Speed	0~127
24	Depth	0~127
25	Type	0~2=Cho/Flg,Ensemble,Phaser
EQ		
26	Hi Freq	0~29=1.00~18.0 [KHz] *T-12
27	Hi Gain	64+/-12=0+/-12
28	Low Freq	0~29=40~1000 [Hz] *T-13
29	Low Gain	64+/-12=0+/-12
ARPEGGIO		

30		tempo (MSB)	20~300
31		(LSB)	(SEQ tempo)
32	B7	Arpeggio On/Off	0,1=Off,On
	B6	Latch	0,1=Off,On
	B4,5	Target	0~2=Both,Timb1,Timb2
	B1	not use	(0)
	B0	Key Sync	0,1=Off,On
33	B0~3	Type	0~5=Up~Trigger *T-14
	B4~7	Range	0~3=1~4 Octave
34		gate time	0~100=0~100[%]
35		Resolution	0~5=1/24,1/16,1/12,1/8,1/6,1/4
36		Swing	0+/-100=0+/-100[%]
37		(dummy byte)	

Synth parameter (Mode = Single,Split,Dual)

38~145	TIMBRE1 DATA	Timbre parameter (TABLE 2)

Synth parameter (Mode = Split,Dual)

146~253	TIMBRE2 DATA	Timbre parameter (TABLE 2)

Vocoder parameter (Mode = Vocoder)

38~115	VOCODER DATA	Vocoder parameter (TABLE 4)
116~253	(dummy bytes)	

TABLE 2 : SYNTH PARAMETER (1 TIMBRE)

+0		MIDI ch.	-1,0~15=GLB,1~16ch
+1	B6,7	Assign Mode	0,1,2=Mono,Poly,Unison
	B5	EG2 reset	0,1=Off,On
	B4	EG1 reset	0,1=Off,On
	B3	Trigger Mode	0,1=Single,Multi (use Mono/Unison Mode)
	B0~1	Key Priority	0~2=Last,Low,High
+2		Unison Detune	0~99=0~99[cent] (use Unison Mode)
PITCH			
+3		Tune	64+/-50=0+/-50[cent]
+4		Bend Range	64+/-12=0+/-12[note]
+5		Transpose	64+/-24=0+/-24[note]
+6		Vibrato Int	64+/-63=0+/-63
OSC1			
+7		Wave	0~7=Saw~Audio In *T-3
+8		Waveform CTRL1	0~127
+9		Waveform CTRL2	0~127
+10		DWGS Wave	0~63=DWGS No. 1~64 (when OSC1 Wave is "DWGS")
+11		(dummy byte)	
OSC2			
+12	B6,7	not use	(0,0)
	B4,5	Mod Select	0~3=Off,Ring,Sync,RingSync
	B2,3	not use	(0,0)
	B0,1	Wave	0~2=Saw,Squ,Tri
+13		Semitone	64+/-24=0+/-24[note]

+14		Tune	64+/-63=0+/-63
PITCH (2)			
+15	B7	not use	(0)
	B0~6	Portamento Time	0~127
MIXER			
+16		OSC1 Level	0~127
+17		OSC2 Level	0~127
+18		Noise	0~127
FILTER			
+19		Type	0~3=24LPF,12LPF,12BPF,12HPF
+20		Cutoff	0~127
+21		Resonance	0~127
+22		EG1 Intensity	64+/-63=0+/-63
+23		Velocity Sense	64+/-63=0+/-63
+24		Keyboard Track	64+/-63=0+/-63
AMP			
+25		Level	0~127
+26		Panpot	0~64~127=L64~CNT~R63
+27	B7	not use	(0)
	B6	Amp SW	0,1=EG2,Gate
	B1~5	not use	(0,0,0,0,0)
	B0	Distortion	0,1=Off,On
+28		Velocity Sense	64+/-63=0+/-63
+29		Keyboard Track	64+/-63=0+/-63
EG1			
+30		Attck	0~127
+31		Decay	0~127
+32		Sustain	0~127
+33		Release	0~127
EG2			
+34		Attack	0~127
+35		Decay	0~127
+36		Sustain	0~127
+37		Release	0~127
LFO1			
+38	B6,7	not use	(0,0)
	B4,5	Key Sync	0~2=OFF,Timbre,Voice
	B2,3	not use	(0,0)
	B0,1	Wave	0~3=Saw,Squ,Tri,S/H
+39		Frequency	0~127
+40	B7	Tempo Sync	0,1=Off,On
	B5,6	not use	(0,0)
	B0~4	Sync Note	0~14=1/1~1/32 *T-6
LFO2			
+41	B6,7	not use	(0,0)

	B4,5	Key Sync	0~2=OFF,Timbre,Voice
	B2,3	not use	(0,0)
	B0,1	Wave	0~3=Saw,Squ(+),Sin,S/H
+42		Frequency	0~127
+43	B7	Tempo Sync	0,1=Off,On
	B5,6	not use	(0,0)
	B0~4	Sync Note	0~14=1/1~1/32 *T-6
PATCH			
+44	B4~7	Patch1 Destination	0~7=PITCH~LFO2FREQ *T-5
	B0~3	Patch1 Source	0~7=EG1~MIDI2 *T-4
+45		Patch1 Intensity	64+/-63=0+/-63
+46	B4~7	Patch2 Destination	0~7=PITCH~LFO2FREQ *T-5
	B0~3	Patch2 Source	0~7=EG1~MIDI2 *T-4
+47		Patch2 Intensity	64+/-63=0+/-63
+48	B4~7	Patch3 Destination	0~7=PITCH~LFO2FREQ *T-5
	B0~3	Patch3 Source	0~7=EG1~MIDI2 *T-4
+49		Patch3 Intensity	64+/-63=0+/-63
+50	B4~7	Patch4 Destination	0~7=PITCH~LFO2FREQ *T-5
	B0~3	Patch4 Source	0~7=EG1~MIDI2 *T-4
+51		Patch4 Intensity	64+/-63=0+/-63
SEQ			
+52	B7	SEQ On/Off	0,1=Off,On
	B6	Run Mode	0,1=1Shot,Loop (only Loop when KeySync is "OFF".)
	B5	not use	(0)
	B0~4	Resolution	0~15=1/48~1/1 *T-7
+53	B4~7	Last Step	0~15=1~16
	B2,3	Seq Type	0~3=Fowrd,Reverse,Alt1,Alt2
	B0,1	Key Sync	0~2=OFF,Timbre,Voice
+54~71		SEQ1 parameter	SEQ parameter [18] (TABLE 3)
+72~89		SEQ2 parameter	SEQ parameter [18] (TABLE 3)
+90~107		SEQ3 parameter	SEQ parameter [18] (TABLE 3)

TABLE 3 : SEQ PARAMETER

+0		Knob	0~30=None~Patch4Int *T-10
+1	B1~7	not use	(0,0,0,0,0,0,0)
	B0	Motion Type	0,1=Smooth,Step
+2~17		Step Value [0~15]	64+/-63=0+/-63

TABLE 4 : VOCODER PARAMETER

+0		MIDI ch.	-1,0~15=GLB,1~16ch
+1	B6,7	Assign Mode	0,1,2=Mono,Poly,Unison
	B5	EG2 reset	0,1=Off,On
	B4	EG1 reset	0,1=Off,On
	B3	Trigger Mode	0,1=Single,Multi (use Mono/Unison Mode)
	B0~1	Key Priority	0~2=Last,Low,High
+2		Unison Detune	0~99=0~99[cent] (use Unison Mode)

PITCH			
+3		Tune	64+/-50=0+/-50[cent]
+4		Bend Range	64+/-12=0+/-12[note]
+5		Transpose	64+/-24=0+/-24[note]
+6		Vibrato Int	64+/-63=0+/-63
OSC			
+7		Wave	0~7=Saw~Audio In *T-3
+8		Waveform CTRL1	0~127
+9		Waveform CTRL2	0~127
+10		DWGS Wave	0~63=DWGS No. 1~64 (when OSC Wave is "DWGS")
+11		(dummy byte)	
AUDIO IN2			
+12	B1~7	not use	(0,0,0,0,0,0,0)
	B0	HPF Gate	0,1=Dis,Ena
+13		(dummy byte)	
PITCH (2)			
+14	B7	not use	(0)
	B0~6	Portamento Time	0~127
MIXER			
+15		OSC1 Level	0~127
+16		Ext1 Level	0~127
+17		Noise Level	0~127
AUDIO IN2 (2)			
+18		HPF Level	0~127
+19		Gate Sense	0~127
+20		Threshold	0~127
FILTER			
+21		Shift	0~4=0,+1,+2,-1,-2
+22		Cutoff	64+/-63=0+/-63
+23		Resonance	0~127
+24		Mod Source	0~7=EG1~MIDI2 *T-4
+25		Intensity	64+/-63=0+/-63
+26		E.F.Sense	0~127
AMP			
+27		Level	0~127
+28		Direct Level	0~127
+29	B1~7	not use	(0,0,0,0,0,0,0)
	B0	Distortion On/Off	0,1=Off,On
+30		Vel.Sense	64+/-63=0+/-63
+31		KeyTrack	64+/-63=0+/-63
EG1			
+32		Attack	0~127
+33		Decay	0~127
+34		Sustain	0~127
+35		Release	0~127

EG2			
+36		Attack	0~127
+37		Decay	0~127
+38		Sustain	0~127
+39		Release	0~127
LFO1			
+40	B6,7	not use	(0,0)
	B4,5	Key Sync	0~2=OFF,Timbre,Voice
	B2,3	not use	(0,0)
	B0,1	Wave	0~3=Saw,Squ,Tri,S/H
+41		Frequency	0~127
+42	B7	Tempo Sync	0,1=Off,On
	B5,6	not use	(0,0)
	B0~4	Sync Note	0~14=1/1~1/32 *T-6
LFO2			
+43	B6,7	not use	(0,0)
	B4,5	Key Sync	0~2=OFF,Timbre,Voice
	B2,3	not use	(0,0)
	B0,1	Wave	0~3=Saw,Squ(+),Sin,S/H
+44		Frequency	0~127
+45	B7	Tempo Sync	0,1=Off,On
	B5,6	not use	(0,0)
	B0~4	Sync Note	0~14=1/1~1/32 *T-6
CH LEVEL		[0]~[15]=CH[1]~[16]	
+46~61		Level [0~15]	0~127
CH PAN		[0]~[15]=CH[1]~[16]	
+62~77		Pan [0~15]	1~64~127=L63~CNT~R63

Appendice B

Tab in Sound Explorer

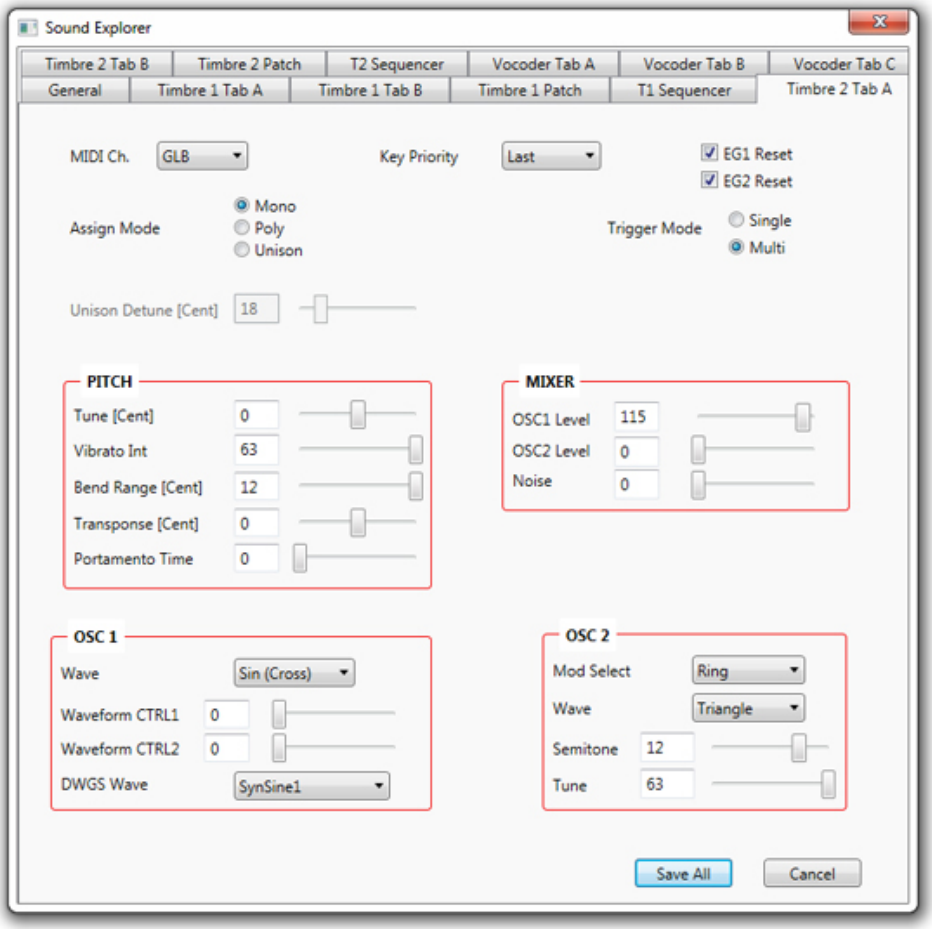

Figura B.1: Timbre Tab A.

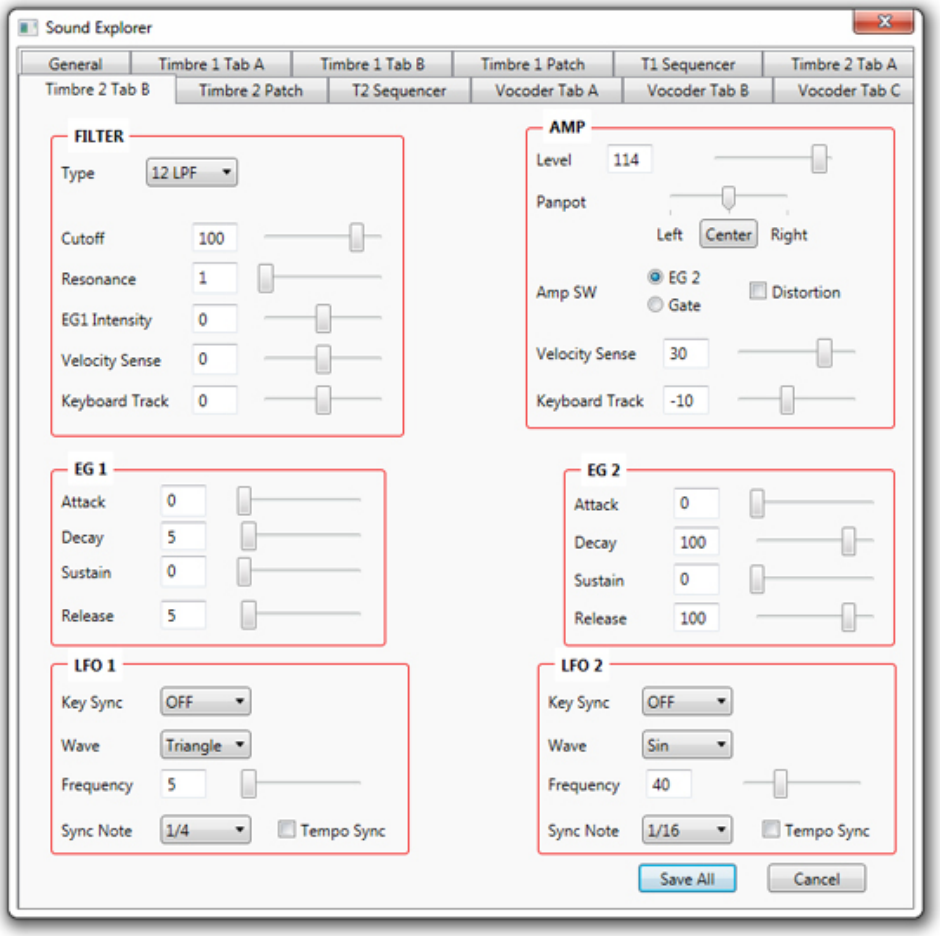

Figura B.2: Timbre Tab B.

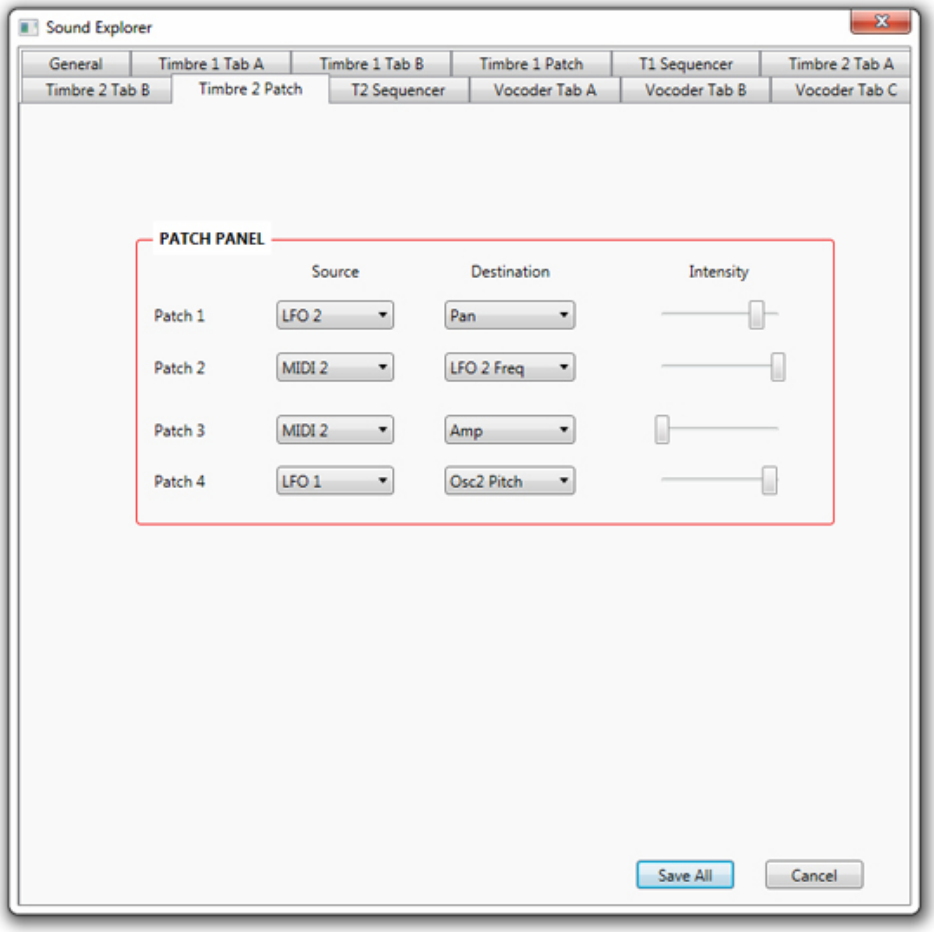

Figura B.3: Timbre Patch.

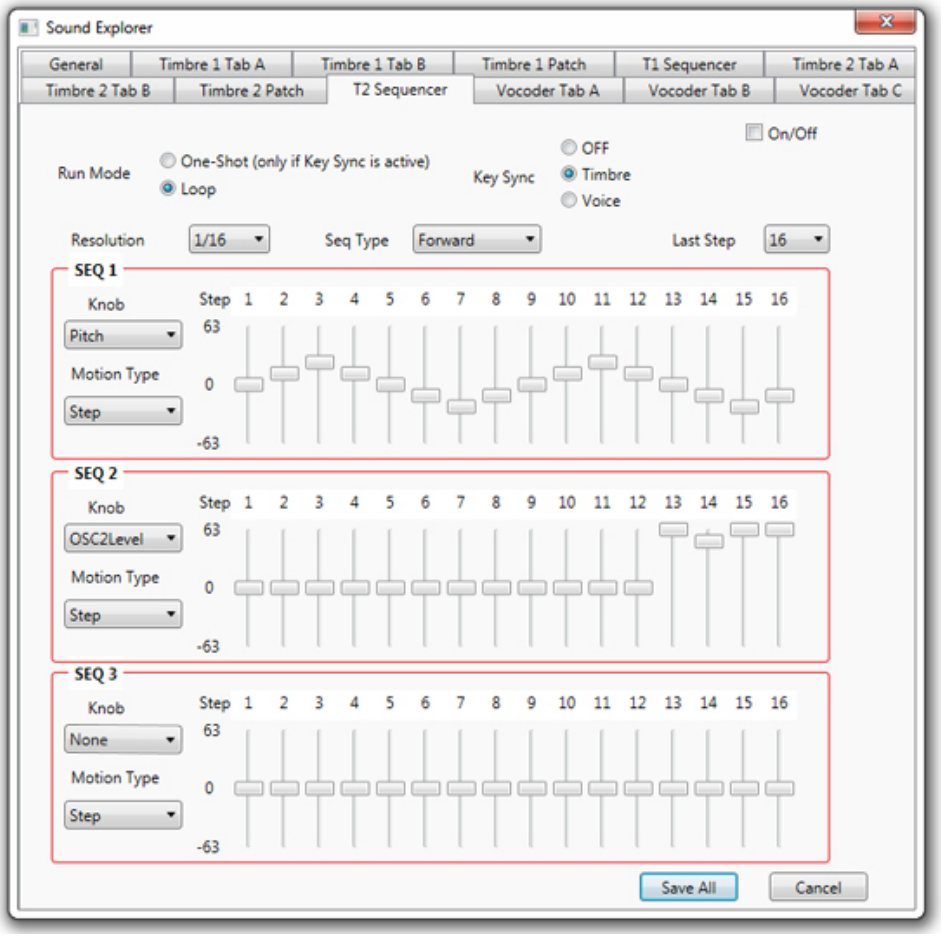

Figura B.4: Timbre Sequencer.

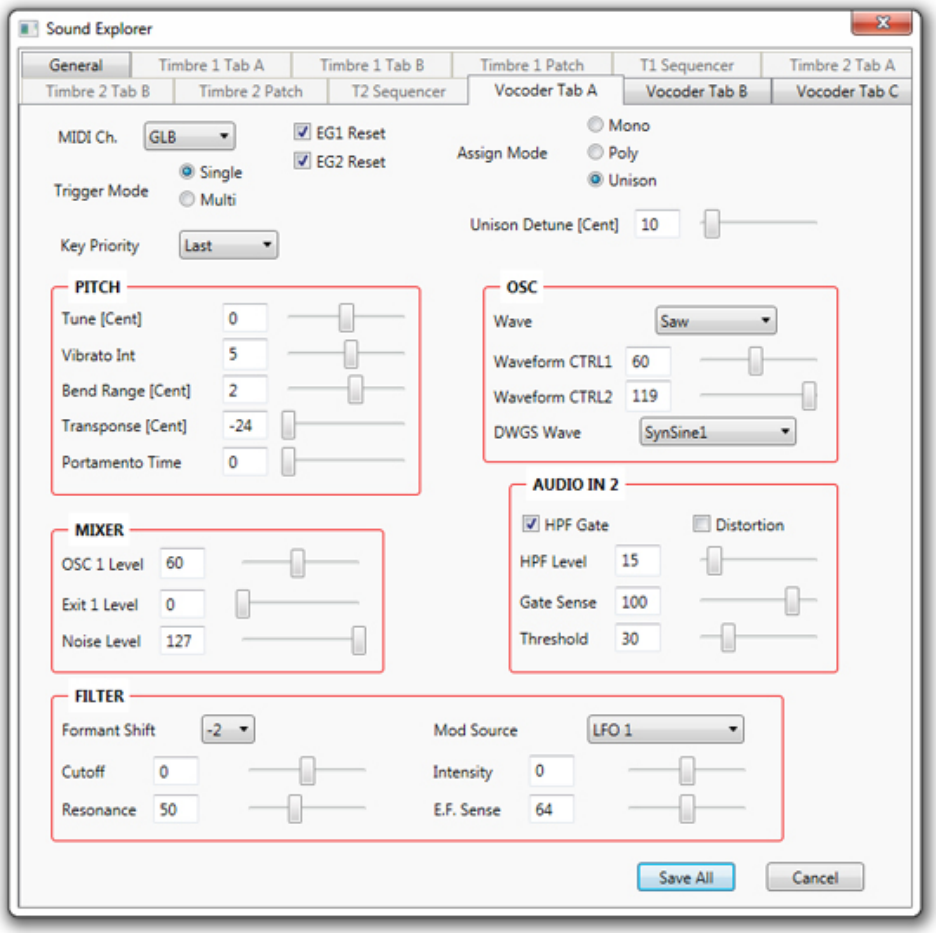

Figura B.5: Vocoder Tab A.

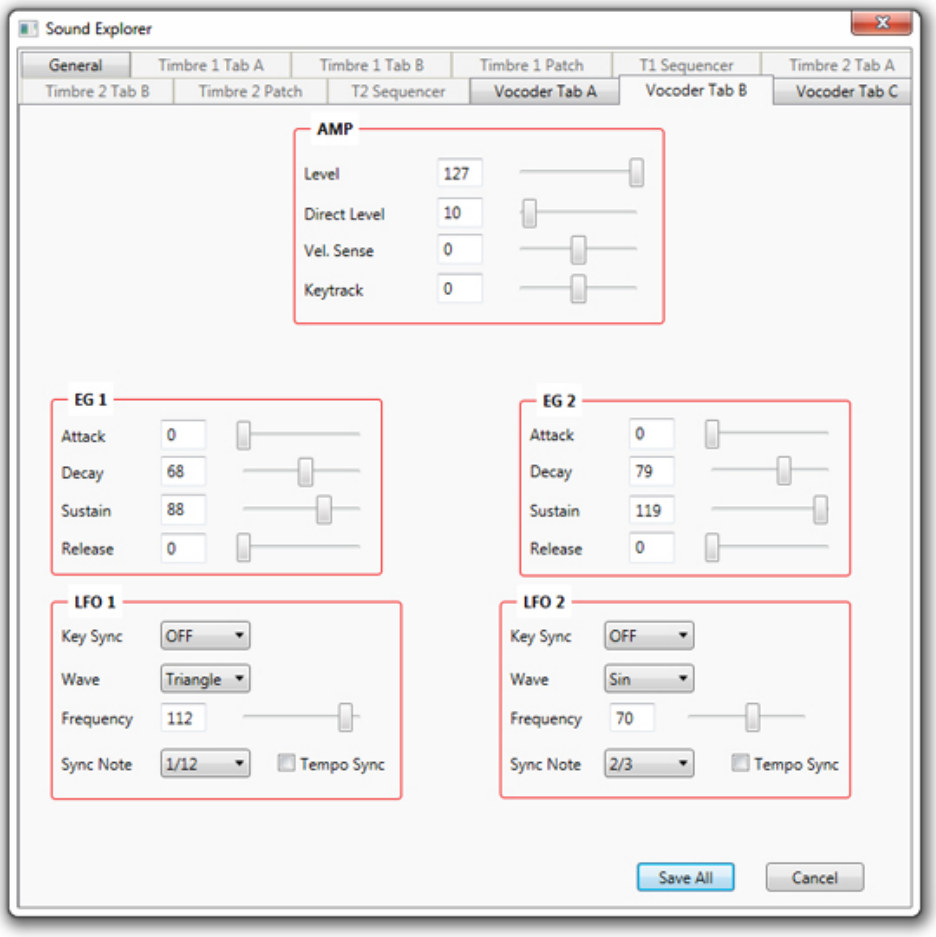

Figura B.6: Vocoder Tab B.

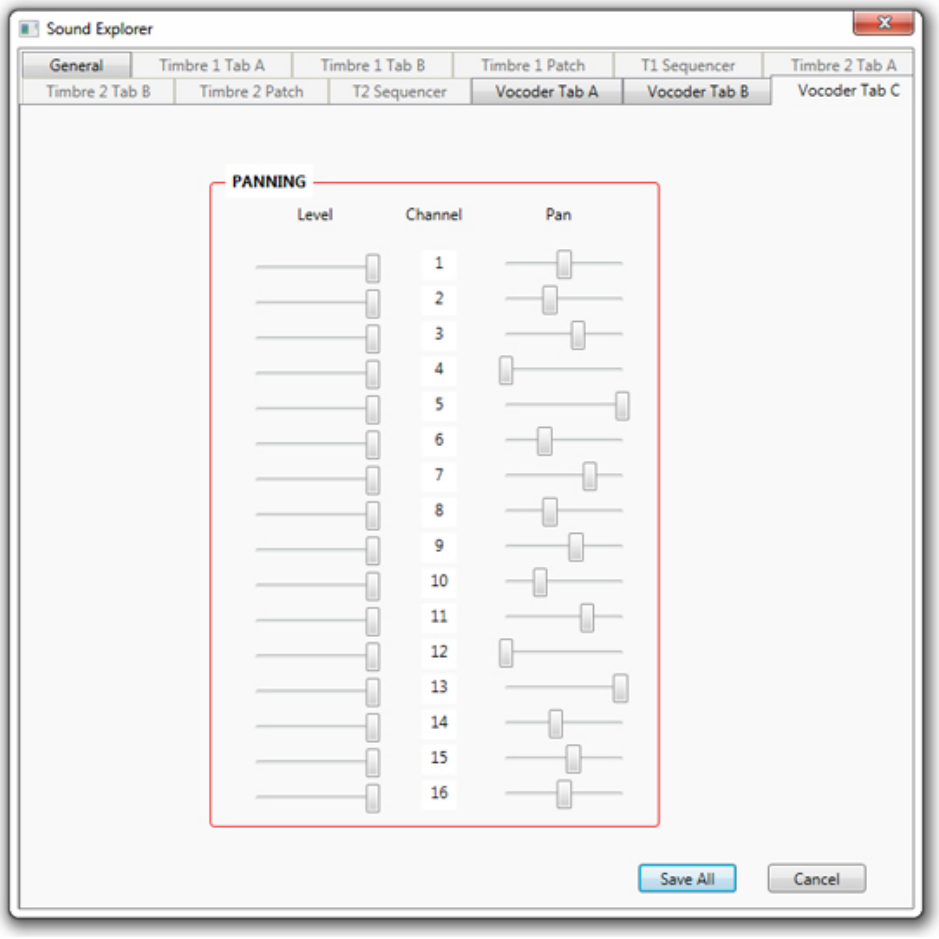

Figura B.7: Vocoder Tab C.

Appendice C

Metodo BuildFinalSysEx

```
1  private byte[] BuildFinalSysEx()
2          {
3              byte[] sysExFinale = new byte[37163];
4              byte[] arrayRicombinato = new byte[32512];
5
6              int cont;
7              char lettera5 = 'a';
8              int num5 = 1;
9
10             for (cont = 0; cont < 32511; cont = cont + 254)
11             {
12                 string idSuono5 = lettera5 + num5.ToString("D2");
13                 splitted2[idSuono5].arr.CopyTo(arrayRicombinato, cont);
14                 if (num5 < 16)
15                 {
16                     num5++;
17                 }
18                 else
19                 {
20                     num5 = 1;
21                     lettera5++;
22                 }
23             }
24
25             int w = 5;
26             int i;
27
28             for (i = 0; i < 32506; i = i + 7)
29             {
30                 string s0 = "0" + (Convert.ToString(arrayRicombinato[i +
                       6], 2).PadLeft(8, '0')).Substring(0, 1) + (Convert.
                       ToString(arrayRicombinato[i + 5], 2).PadLeft(8, '0')).
                       Substring(0, 1) + (Convert.ToString(arrayRicombinato[i +
                        4], 2).PadLeft(8, '0')).Substring(0, 1) + (Convert.
                       ToString(arrayRicombinato[i + 3], 2).PadLeft(8, '0')).
                       Substring(0, 1) + (Convert.ToString(arrayRicombinato[i +
```

```
                    2], 2).PadLeft(8, '0')).Substring(0, 1) + (Convert.
                    ToString(arrayRicombinato[i + 1], 2).PadLeft(8, '0')).
                    Substring(0, 1) + (Convert.ToString(arrayRicombinato[i +
                    0], 2).PadLeft(8, '0')).Substring(0, 1);
31                  byte b = Convert.ToByte(s0, 2);
32                  sysExFinale[w] = b;
33                  w = w + 8;
34              }
35
36              int p;
37              int w2 = 6;
38              for (p = 0; p < 32512; p++)
39              {
40                  string s = "0" + Convert.ToString(arrayRicombinato[p], 2).
                        PadLeft(8, '0').Substring(1, 7);
41                  byte n = Convert.ToByte(s, 2);
42
43                  if ((w2 - 5) % 8 == 0)
44                  {
45                      w2++;
46                      sysExFinale[w2] = n;
47                      w2++;
48                  }
49
50                  else
51                  {
52                      sysExFinale[w2] = n;
53                      w2++;
54                  }
55              }
56
57              sysExFinale[0] = 0xf0;
58              sysExFinale[1] = 0x42;
59              sysExFinale[2] = 0x30;
60              sysExFinale[3] = 0x58;
61              sysExFinale[4] = 0x4c;
62              sysExFinale[37162] = 0xf7;
63              return sysExFinale;
64          }
```