A Complete Guide to

manual v1.6

A Complete Guide to

Plogue chipsounds ®

Including the ARIA™ Engine

Produced by:	David Viens
ARIA Engine Design/Development:	Plogue Art et Technologie, Inc.
Sound Programming:	David Viens Eric Patenaude Bryan Lee of Xenos Soundworks
Hardware research and modeling :	David Viens
MIDI Processors:	Pascal Maheux
In house Testing and QA:	Eric Patenaude
Ambience Reverb:	Magnus Jonsson
Art :	James Mireau,Eric Patenaude, Flavours of Lime
Additional hardware research:	François Léveillé
Crucial Comments and Bug reports:	nitro2k01 (Didrik Madheden) Chupathingy (Chris Duddridge)
Original ARIA Manual Template:	Gary Garritan
Operations:	Max Deland

Plogue Art et Technologie, Inc.
CP 37313 CSP Marquette, Montcalm, QC, Canada, H2E 3B5
e-mail: info@plogue.com
Visit us on the World Wide Web: **www.plogue.com**

Welcome to Plogue chipsounds

Like some of you I'm sure, my childhood was filled with the images, sounds and excitement of early video games and computers. In the early 80's, each of my friends had a different computer or video game system. We visited each other in turns and discovered their differences. They all looked, played and sounded very different to us *even then*. We all had our favorite game or favorite sound effect and each of us tried to impress the other with a new high score, secret level, or cool program we had made.

The 8bit era was a unique point in time when there were no set rules in what we now call interactive media. *Everything had yet to be invented.* Musicians, graphic artists and programmers (often the same person) had to cope with a limited palette of sounds, voices, and memory, in order to create the inspiring soundtracks for the games we played. They came up with original techniques and tricks that – since they were stopgaps for them at the time - were abandoned later on as the technology evolved.

Those sounds were unique in their own way.

Recently, with the advent of emulators, cross assemblers and DIY cartridges, many people have discovered (or re-discovered) those sounds and got an interest in making original music with the vintage 8bit hardware in ways that combined classic 80s chip techniques, but also *brand new* tricks and discoveries of hackers and enthusiasts from all over the world. The chip music scene nowadays goes *WAY* beyond video game or 80s nostalgia in general, and has become a true art form in its **own right.** A good indication is that a new younger generation of musician that *didn't grow up in that period* are praising the palette of sounds that can be created out of these machines; raw, pure, gritty and bright. Whether you compose uniquely with them or use them as an addition, they are a real **arsenal** to your sonic palette.

For me *personally,* it would be pointless to try and separate my passion for the chips and how they sound from their associated nostalgia. Over the years I've been working in the music software field, I've always kept a huge interest in them, collecting (hoarding would be a more appropriate term maybe), tons of consoles and started modifying them to get cleaner sound and image out of them, and even transforming some of them into synthesizers to use with my bands. As I was improving my electronics skills on them as a hobby I sadly discovered that many units I had collected over the years had either become flaky, partially working or were simply dead. And it's at that precise moment that I realized, that like everything else, they would not live forever. Someone needed to record and analyze all the sounds could generate in the highest possible quality before it was too late. It's **then** that I started the research that, without any initial plan of the sort, eventually transformed itself into chipsounds, which I think is my best attempt at making a musician's tool out of bare research.

My father bought a Commodore VIC-20 for me and my brother when I was around nine. After a few days with it, I guess he knew I had found my calling. There is not a year that passes without me reminding him how bringing that computer home some cold autumn night changed my life.

Dad, this one's for you,

David Viens
October 2009

What is chipsounds?

Plogue chipsounds is a state-of-the-art software synthesizer that simulates the behavior and sounds of more than ten classic integrated circuits of the 80's, mostly from the second and third generation of video game consoles.

chipsounds integrates our uniquely powerful and high-performance specialized software synthesizer/sampler engine, ARIA.

chipsounds can work as a standalone application or as a plug-in for most major sequencing audio programs and supported tracker programs.

Uses for Plogue chipsounds

- Electronic and chip music composition.
- Remixes of classic video games soundtracks
- A portable chip synthesizer instrument for playing live.

Our goal is to bring the rich sounds of the classic game consoles to as many musicians as we can with the highest degree of realism possible on *current* home computer hardware.

Features of Plogue chipsounds

- **A full set of classic sound chips at your disposal.**
- **Nothing Else Required** - The entire collection is integrated and works as a virtual instrument. No need to purchase a separate synthesizer.
- **Easy-to-Use –** Start creating chip music now. Just load your instruments and play. Standardized controls allow you to become familiar with the collection quickly and master all the sounds easily. Play in **real-time** and get expressive sounding performances. Express your musical ideas fast and with minimal effort.
- **Universal Format** - Supports all popular formats, Mac and PC, as a standalone program or as a plug-in (VST, RTAS, and OS X AU), and works with supported sequencers and tracker programs.

What's Included

This 1.6 version of chipsounds includes the following:

- Download file containing the complete chipsounds software.
- The unique graphical license key for the product.
- This digital PDF manual.

Before you begin installation, make sure you have read the End User Licensing Agreement in the pages which follow. By installing the software you are indicating you agree to the terms of the license.

How to Use This Manual

The goal of this manual is to help you learn how to use chipsounds. Although many of us dislike reading manuals, if you wish to get the most out of this new synthesizer it is absolutely essential that you read this manual. Doing so will help you understand how to use this software instrument. The operation of many of the essential features is not obvious in casual use and we realize many users are not music technologists. We'll do our best to make this easy for you in this manual and to explain technical concepts. We have attempted to make this manual easy to read and have provided information about the various waveforms, playing techniques and modes of control.

You can refer to this manual whenever you wish. This manual is provided in digital form as an Adobe Acrobat document file (also known as a PDF) which can be viewed on a computer monitor or printed. If you do not have the Adobe Acrobat Reader, it is available for free from www.adobe.com.

Further Documentation and Resources

For the latest information including updated documentation, visit our support pages at:
www.plogue.com. There you can find: Updated information provided after the manual was written Corrections or additions to this manual FAQ pages answering common questions suggestions from the users of Plogue software and news about upcoming Plogue releases. You can also visit the Plogue Forums for up-to-date information.

The address is: www.plogue.com/phpBB3/ .

Please send any reports of errors in this manual or suggestions for improvement to :
chipsounds.support@plogue.com

End User License Agreement

Please read the terms of the following software licensing agreement before using this software. By installing and loading this product on your computer you acknowledge that you have read this license agreement, understand the agreement, and agree to its terms and conditions. If you do not agree to these terms and conditions, do not install or use the sounds contained herein. This is the complete agreement between you and Plogue Art et Technologie, Inc that supersedes any other representations or prior agreements, whether oral or in writing.

An important thing to understand is that YOU ARE OBTAINING A LICENSE FOR YOUR USE ONLY—THE SAMPLES AND PROGRAMMING DO NOT BELONG TO YOU. The implications are described below. The sounds, samples and programming in the Plogue chipsounds program remain the sole property of Plogue Art et Technologie, Inc. and are licensed (not sold) to you.

What You Can Do: You can use these sounds in music productions, public performances, and other reasonable musical purposes within musical compositions. You can use these sounds in your own musical compositions as much as you like without any need to pay Plogue Art et Technologie Inc or obtain further permission. If you do use these sounds, we kindly ask that in any written materials or credits accompanying your music that utilizes material from Plogue chipsounds (CD booklet, film credits, etc), that you include the following courtesy credits: "Some Virtual Instruments used in this recording are from Plogue chipsounds, or a similar credit where practicable. If you can't disclose your use of chipsounds for any reason to the general public, we would at least love to be aware of the usage of chipsounds in a commercial recording for our own PRIVATE records.

What You Cannot Do: The enclosed sounds cannot be re-used in any other commercial sample library or any competitive product. You are absolutely forbidden to duplicate, copy, distribute, transfer, upload or download, trade, loan, reissue or resell this product or any of the contents in any way to anyone. You cannot redistribute them through an archive, nor a collection, nor through the Internet, nor binaries group, nor newsgroup, nor any type of removable media, nor through a network. The sounds and samples contained herein cannot be edited, modified, digitally altered, re-synthesized or manipulated without direct written consent of Plogue Art et Technologie Inc.

Disclaimers and Conditions: A right to use Plogue chipsounds is granted to the original end-user only, and this license is not transferable unless there is written consent from Plogue Art et Technologie, Inc and payment of an additional fee. The sounds of Plogue chipsounds will only work with the bundled Plogue ARIA Engine and will not work with any other engine. Licensor will not be responsible if the content does not fit the particular purpose of the Licensee. Please make sure before installing this item that it meets your needs. Information contained herein is subject to change without notice and does not represent a commitment on the part of Plogue Art et Technologie, Inc. The sounds are licensed "as is" without warranties of any kind. Neither Plogue Art et Technologie, Inc, nor any agent or distributor can be held responsible for any direct or indirect or consequential loss arising from the use of this product in whatever form. The Aria Engine is covered by the installer's End User License Agreement and is incorporated by reference. Plogue chipsounds may not be returned for any reason. The terms of this license shall be construed in accordance with the substantive laws of Canada and the province of Quebec.. The user agrees to read the manual before seeking technical support and to make sure his or her system meets the recommended requirements.

Specifications & Computer System Requirements

The following table lists the computer and hardware requirements for using chipsounds. You can use chipsounds on most any modern personal computer that meets the specifications listed below. The specifications provide the minimum standards. For optimal functioning, it is recommended you have a powerful enough computer with a fast CPU processor, and a large amount of RAM. The ARIA audio engine is designed to make use of the processing power of your computer's CPU. The powerful and complex algorithms of the ARIA Engine work best on modern CPU's. We think that's a small price to pay for the results you will get. Please also observe the systems requirements of your host application, tracker program and/or sequencing program if applicable. Please see the Plogue forum or website if you are looking for recommendations or for more information.

Computer System Requirements		
Computer	***Operating System***	***Hardware***
Windows PC	Microsoft Windows XP (SP2 or SP3 recommended) Microsoft Windows Vista 32 or 64 bits Microsoft Windows 7 32 or 64 bits	♦2.4 GHz CPU Pentium 4 or better, 2.0 GHz Core Duo or better recommended ♦1 GB Minimum, 2 GB RAM recommended if used with other instruments at the same time. ♦150 MB of free hard drive space ♦Internet connection for download version, CD ROM drive required for boxed version installation ♦Monitor with 1,024x768 resolution or better for usage of the default skin. ♦A sound card compatible with ASIO, DirectSound, or MME ♦Keyboard: A MIDI interface may be required if you are using a MIDI keyboard. Some keyboards use USB. ♦High quality speakers and amplifier, or high quality headphones ♦Internet connection for download, updates.
Mac	Mac OS X 10.6 minimum	♦Mac Intel; Core Duo or better recommended ♦1 GB Minimum, 2 GB RAM recommended if used with other instruments at the same time. ♦150 MB of free hard drive space ♦Internet connection for download version, CD ROM drive required for boxed version installation ♦Monitor with 1,024x768 resolution or better for usage of the default skin. ♦A MIDI interface may be required if you are using a MIDI keyboard. Some keyboards use USB. ♦High quality speakers and amplifier, or high quality headphones ♦Internet connection for download, updates and online registration

The stated requirements represent minimum guidelines for the Standalone Plogue chipsounds Player. If you are using chipsounds within a host music program, then there may be other additional resource requirements. Please also observe the system requirements of your host application, tracker program and/or sequencing program if applicable. The demands of various other processing software (including the sequencer, audio and effects processors, other plug-ins, and so on) can affect functionality.

Regarding Sound Cards & Midi Interfaces

The quality of the audio interface will have a significant effect on the quality of the sound you will hear from Plogue chipsounds. It will also have a substantial effect on performance (latency). Therefore, a good sound card is one of the most important components in optimizing the sound and performance of Plogue chipsounds.

In theory, any audio or sound interface which your OS and computer supports should work. However, you are unlikely to get the best sonic results from a sound card designed for computer games or system sounds. Most computers come with a consumer-grade sound card, and we recommend that you get a good quality sound interface beyond that which is built into your computer. Older SoundBlaster sound cards (which do not support multiple sample rates) and gamer-oriented or home system sound cards may be problematic. It is not possible for us to test all built-in or third-party sound cards, and some interfaces do have problems on some platforms.

Technical Info

A low latency audio interface with ASIO 2.0 (Windows), or Core Audio (Mac), drivers is required for chipsounds to work as a stand-alone program. These drivers are normally installed with the audio interface, or the most recent versions can be acquired from the manufacturer's website. Contact the manufacturer of your interface for more information. The drivers should be set to 24 bit, buffer size 256 samples (optimal) or 512 (more latency, but less CPU load) and 44100Hz Sampling Rate.

Please note:

When Plogue chipsounds is running as a plug-in, it uses the audio driver selected by the host's setup. If the host (typically your sequencer or tracker program) is set up properly and works well, then the chipsounds plug-in should pass through the same audio and MIDI setup. For this information, please refer to your sequencer, tracker program or host's manual.

Similarly, any MIDI interface the manufacturer supports for your system should work with Plogue chipsounds.

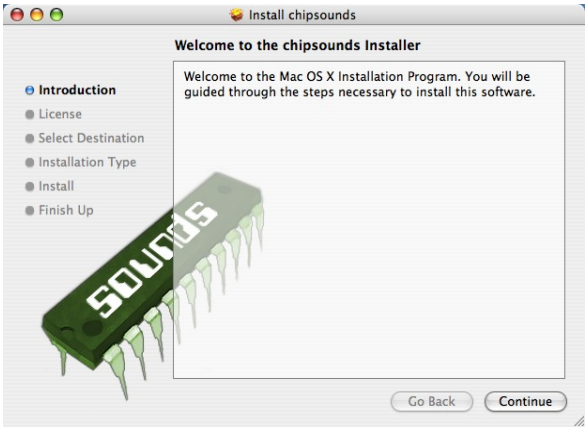
INSTALLATION

Installing Plogue chipsounds

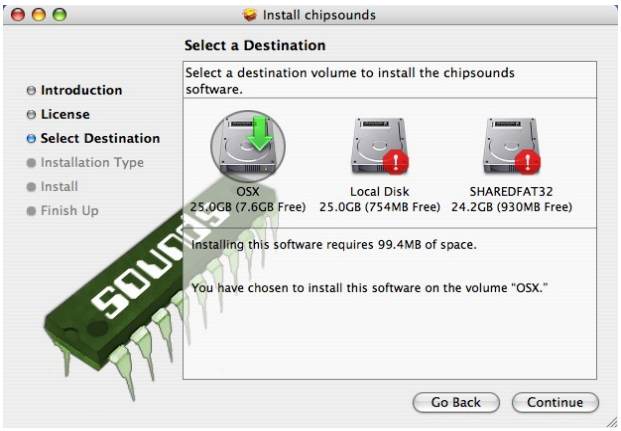
Installing Plogue chipsounds is easy. Before you begin, make sure you meet the system requirements The full installation requires approximately one hundred megabytes of free hard disk space. Installation involves installing the chipsounds instruments and also the advanced ARIA engine. A setup program will guide you through the process step-by-step.

Mac Setup

- First, make sure your audio and MIDI hardware is set up and working with your computer.
- Close any programs you are running.
- For the Download version of chipsounds, the Archive Utility will unpack this to your default downloads directory.
- Double click on MAC_chipsounds_v1.6.pkg (Accept the defaults on the installer. For CD versions of chipsounds, take the Installation disk out of its case, put it) into your CD drive in the computer and close the drive tray. You should see a Welcome Screen.

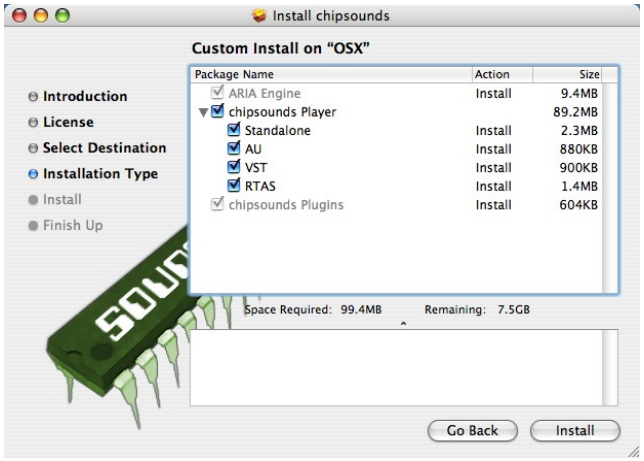


- You will then be asked to accept the End User License Agreement before proceeding with installation.
- And choose which partition to install it on:

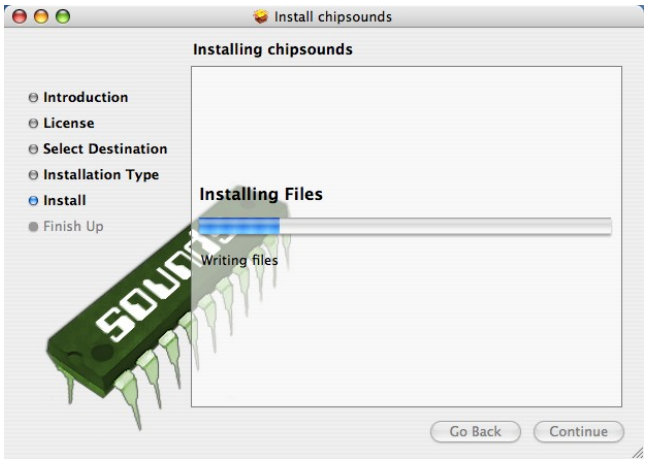


- The setup lets you select which plug-in formats you require. These plug-in formats allow Plogue chipsounds, in addition to standalone use, to run as a virtual instrument plug-in that seamlessly integrates into your favorite music software program or sequencer (assuming that it accepts such instrument plug-ins). Please refer to the chapter “Using Plogue chipsounds as an Instrument Plug-In” for further information.

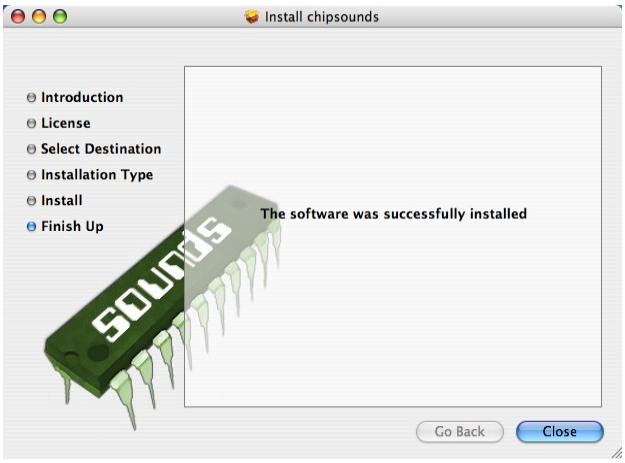
  For Mac, the Choices are Standalone, Audio Units, RTAS and VST at least ONE needs to be chosen



- Setup is now ready to begin installing Plogue chipsounds. Click on “Install” to begin installation and the program will begin installing. This process should not take more than a minute.



- After a few moments a dialog box will notify you that the setup wizard has finished installing Plogue chipsounds on your computer.



When installation is complete, you should see a folder containing the following files in your /Application/Plogue chip-sounds folder:

You can drag the “chipsounds” chip icon onto your Dock.
Don't click on anything else, or move **ANYTHING** away from this folder. **All the files in this folder are useful.**

When you are finished with installation, remove the disc(s) from your drive and store them in a safe place if you have the CD version. If you purchased the download version make sure to make a backup copy. If anything happens to your computer, you can reinstall Plogue chipsounds from the discs or the backup.

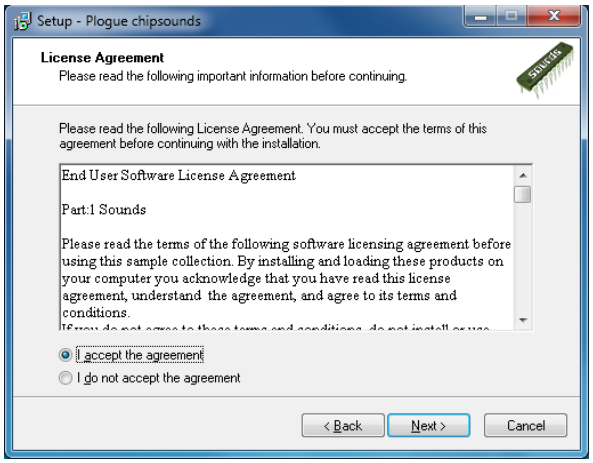
Important:

Please do not cancel setup after installation begins, otherwise a partial, broken installation may result.

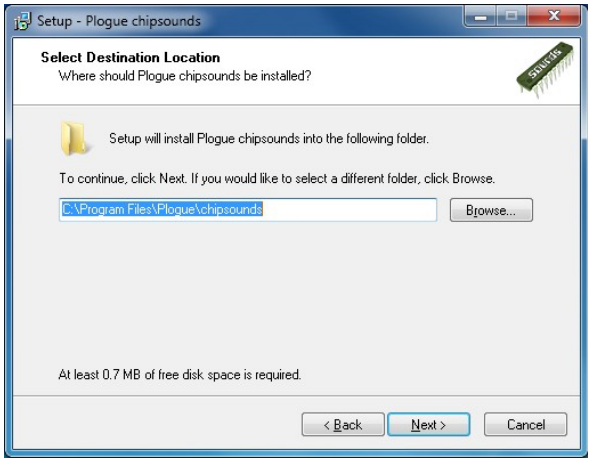
Windows Setup:

It is recommended that you just select “Next” for each screen and use the defaults.

- First make sure your audio and MIDI hardware is set up and working with your computer.
- Close any programs you are running.
- For the Download version of chipsounds, after download, unzip by double-clicking WIN_chipsounds_v1.6.exe. For CD versions of chipsounds, take the Installation disk out of its case, put it) into your CD drive in the computer and close the drive tray. You should see a Welcome Screen. Press Next (If the setup screen does not automatically appear: Use the Windows Explorer (PC) to open the installation CD, or for Download users, WIN_Plogue_chipsounds_v1.6.exe (PC).)
- You will then be asked to read and accept the End User License Agreement before proceeding with installation.

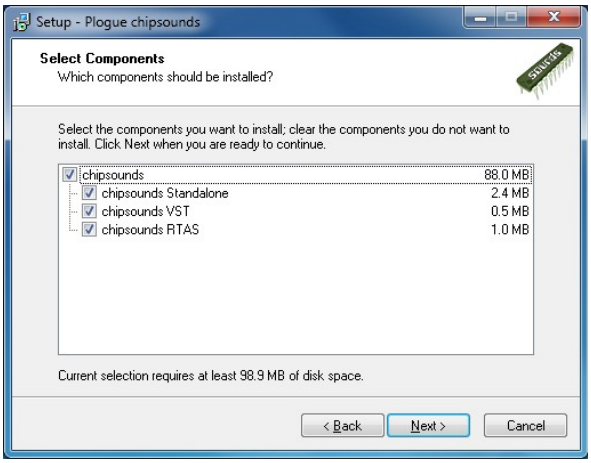


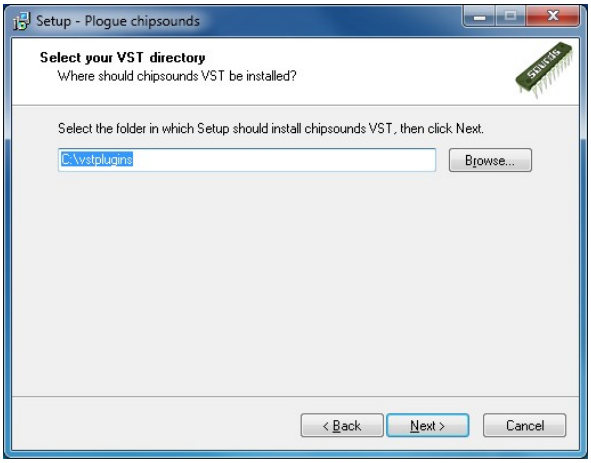
- And the destination folder. Defaults are fine since no samples are *streamed* while using this virtual instrument. (the full size is close to 100MB.)



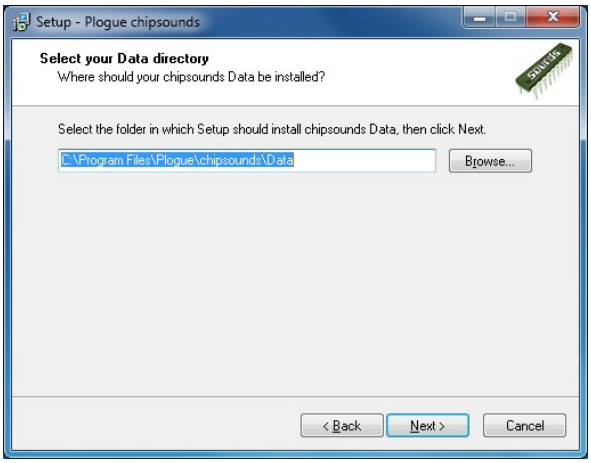
- The setup lets you select which plug-in formats you require. These plug-in formats allow Plogue chipsounds, in addition to standalone use, to run as a virtual instrument plug-in that seamlessly integrates into your favorite music software program or sequencer (assuming that it accepts such instrument plug-ins). Please refer to the chapter “Using Plogue chipsounds as an Instrument Plug-In” for further information .For Windows the choices are VST & RTAS

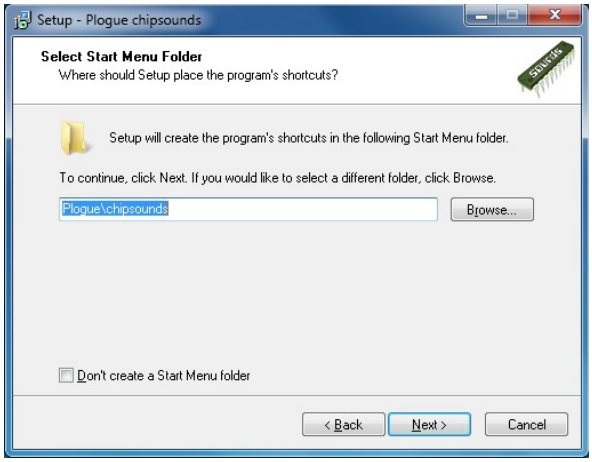
- If you select to install the VST32 and VST64 versions of chipsounds, you will also be prompted with your preferred VST plugins folder in each case:



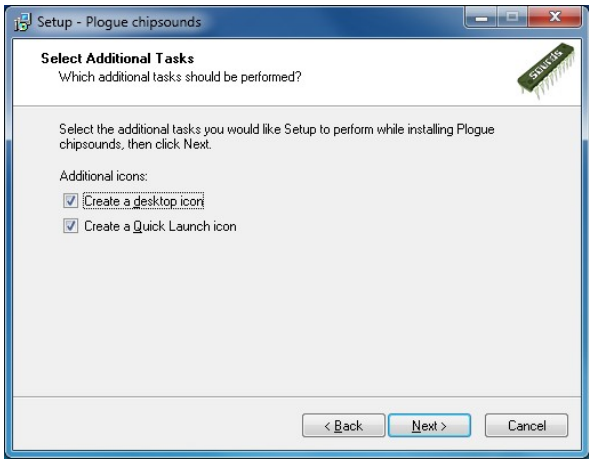


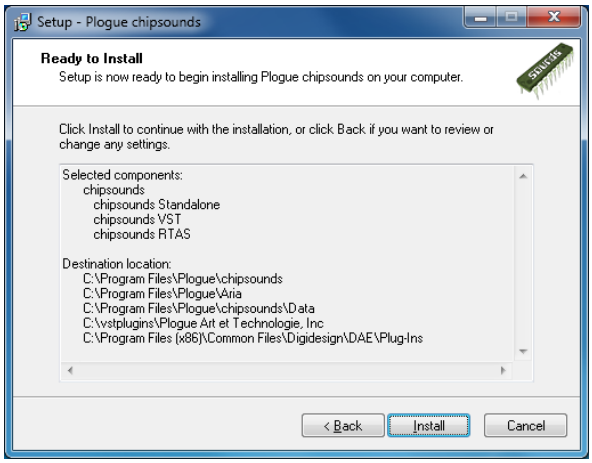
- Where you want to put chipsounds'' data Directory (samples and other data files) and where the shortcuts and icons should be placed. For most instances, use the default.



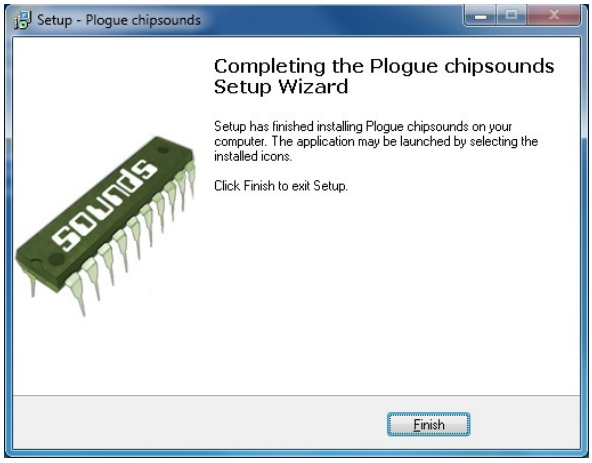


- Setup is now ready to begin installing Plogue chipsounds. Click on “Install” to begin installation and the program will begin installing.





You will see a final screen indicated that the full installation is complete.



When installation is complete, you will need to authorize your current copy of Plogue chipsounds. This procedure is described in the next chapter.

When you are finished with installation, remove the disc(s) from your drive and store them in a safe place if you have the CD version. If you purchased the download version make sure to make a backup copy. If anything happens to your computer, you can reinstall Plogue chipsounds from the discs or the backup.

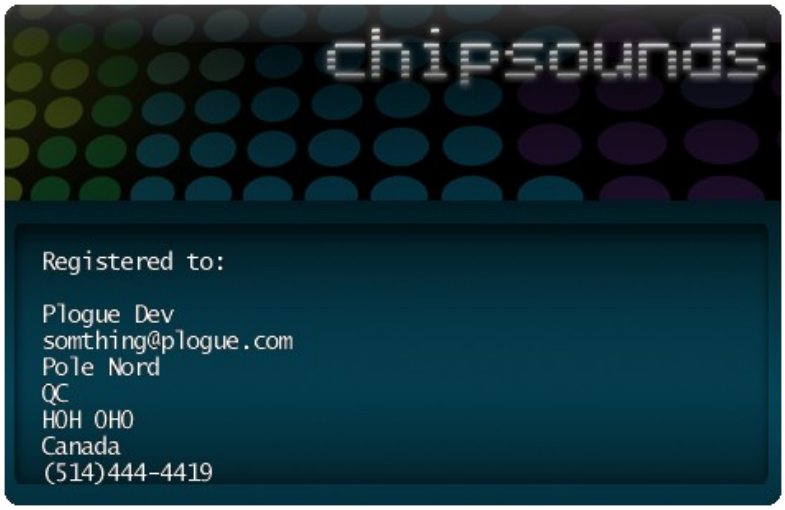
Important:

Please do not cancel setup after installation begins, otherwise a partial, broken installation may result.

Authorizing chipsounds

chipsounds needs to be authorized in order to be fully functional, otherwise it will run in **DEMO** mode.
When you order a license from Share*it, you receive a personal Activation Key card named ***aria_key_1009.png*** as an email.

The Activation Key card is an image resembling a typical credit card.
This image contains your registration and details encoded within the Key card image. It will look like this:



You should save the aria_key_1009.png image file to your hard drive (keep it in a safe place).
For convenience, we recommend that you initially save the .png file to your desktop.
You will also receive a copy of the license card in your email inbox.

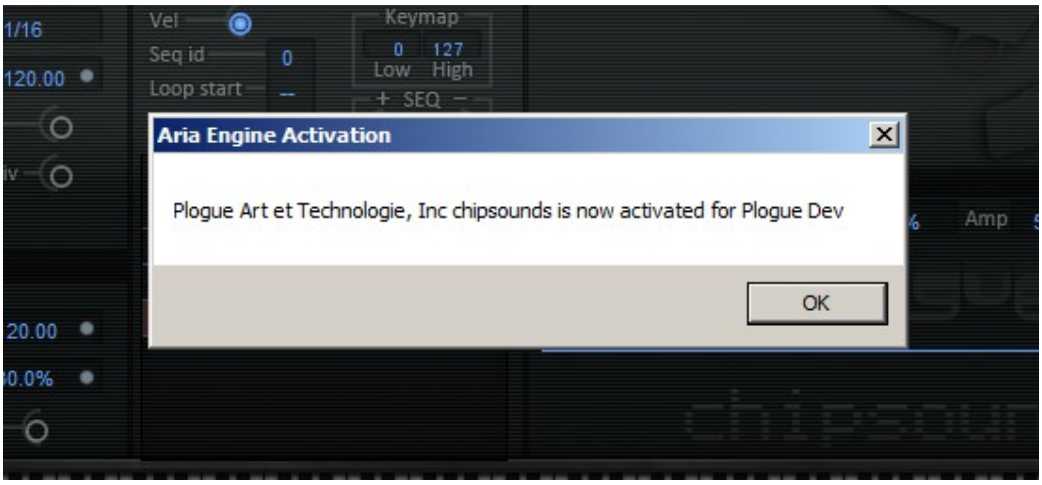
1) Locate the “license card” image where you saved it on your hard drive. (***aria_key_1009.png)***

2) Open the chipsounds software application, or launch your favorite host and make sure you see chipsounds interface.

3) Simply click and hold on the file, drag the “license image” or file icon directly onto the application's UI itself, and release it.

If you don't get any message (or are not able to drop the key in that host), try to ***import*** the png file from the the snap-shot load menu or the slot load import. (use *.* as file filter)

You should be presented with a message such as this one:



If none of those methods work, please contact chipsounds.support@plogue.com and attach your key.

Extremely Important!!

The ***aria_key_1009.png*** file contains your ***sensitive personal information***, encrypted inside, including your **full name and address** taken from the online shop. Carefully protect this file. DO NOT GIVE THIS FILE TO ANYONE OR DISTRIBUTE IT IN ANY WAY OR YOUR PERSONAL INFORMATION WILL BE COMPROMISED. IF THE FILE BECOMES PUBLIC THE CARD NUMBER WILL BE BLACKLISTED AND THE CARD REVOKED. WE ARE NOT RESPONSIBLE IF YOU GIVE YOUR PERSONAL DETAILS TO A THIRD PARTY. IF THE CARD IS STOLEN, CONTACT US IMMEDIATELY. Without a valid card you will also not be able to obtain critical updates to the program.

Important Note:

If you have special circumstances or require site licensing, please contact us.

Updating to the Latest Version

Be sure to check the Plogue Web sites for any possible updates that have occurred since the time your version of the software was released. Software is frequently updated and a more recent version may be available.

How to Use Plogue chipsounds

Once installed and authorized, it's time to get started with Plogue chipsounds.
There are three ways to use Plogue chipsounds: as a plug-in within a sequencer, with a supported tracker program, or you can play it 'live' as a standalone application.

Playing chipsounds as a Standalone Application ('Live' Play)

Plogue chipsounds can be launched by itself and played live via MIDI keyboard or other MIDI controller. The standalone version of Plogue chipsounds effectively makes your computer, audio hardware and MIDI keyboard into a virtual synthesizer that can be played independently of other programs. Unlike using it as a plug-in within a sequencer, your recording ability is limited and you can not edit your performance (though you can use various audio software programs for this).

Launching chipsounds in Standalone Mode

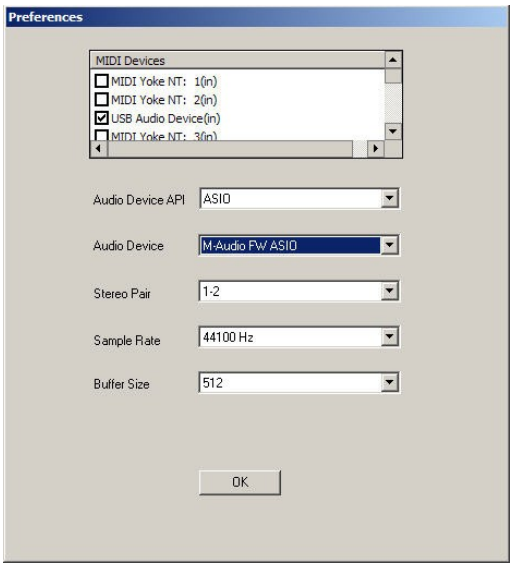
First, make sure that you have followed the instructions in the installation section of this manual. Be certain that your audio/sound interface and MIDI hardware interfaces are properly connected to the computer, your speakers or headphones are connected and everything is powered up.

To launch Plogue chipsounds as a standalone application, click on the Plogue chipsounds logo on your desktop or go to the Program Files or Applications folder and launch "chipsounds".

Basic Setup Information for Standalone Mode

To use the standalone version you have to configure the Audio and MIDI settings in the Preferences dialog box (found in the Tools menu) before you can play. When used as a plug-in, the host sequencer or tracker program has already set up its audio and MIDI connections, and the Plogue chipsounds "plugs in" to them. However, with standalone operation Plogue chipsounds communicates directly with your audio and MIDI interface. Setup for Mac and Windows computers is similar, except where indicated. Note that if you change your audio interface, you will almost certainly need to readjust these settings.

Call up the Preferences setup dialog from the Tools menu on the Plogue Aria standalone interface. You'll see drop-down menus for MIDI Device, Audio Devices, Stereo Pair, Sample Rate and Buffer Size.

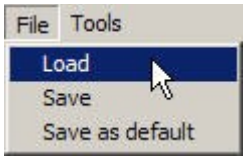


- *MIDI Device Menu:* All supported (and installed) MIDI interfaces are available in this drop-down list. Select the desired MIDI device from the list. The Plogue Aria Engine sends and receives MIDI on these selected devices.

- *Audio Device Menu:* All supported (and installed) audio interfaces are available in this drop-down list. Select the desired audio device from the list.
- *Stereo Pair:* Here you can define which of the stereo outputs should be used. Many pro audio devices have a variety of outputs, so you may choose which of these are available on your system that you would like chipsounds to output through.
- *Sample Rate:* Depending on the sound card and driver you are using, various sample rates are available. Set the desired sample rate here.
- *Buffer Size:* The buffer size setting will determine the delay between pressing a key on your MIDI keyboard and hearing the sound (a/k/a 'latency'). The default buffer size of 512 samples typically works well, but smaller buffer sizes will give a faster response (lower latency) and higher buffer sizes will give better audio performance (more polyphony and higher fidelity). Most modern computers and audio interfaces can handle a buffer size of 512 samples without a significant reduction in polyphony. If the sound is breaking up or crackling when a note sounds, then first check that the audio connections and wiring are good. Then, try a larger audio buffer size setting. Please note that there is typically a trade-off between higher buffer sizes (polyphony and sound fidelity) and lower buffer sizes (faster response or lower latency). Also note that the sound card buffer size settings determine latency, rather than Plogue chipsounds Player itself.

Once you have your Audio and MIDI set up, and have loaded one of the snapshots, you can begin playing chipsounds. Press (use your mouse to click on) a key on the on-screen piano keyboard on the chipsounds Player interface. If you can hear the chips play, try playing a key on your MIDI keyboard. If the MIDI and audio configurations are correct, you should hear the corresponding synth note. If not, check the MIDI connections and wiring, and the MIDI output channel of your MIDI keyboard. Also check that the channel is specified correctly. If you are hearing the notes play, then the basic configuration is complete, and you are ready to use Plogue chipsounds.

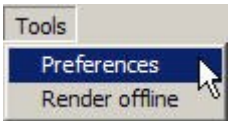
File Menu for Loading and Saving Snapshots in the Standalone



Configuration presets (.aria files - binary) for chipsounds can be saved and loaded. This gives the user the ability to customize instruments setups to suit personal preferences and save configurations for convenient future use. The File menu choices are:

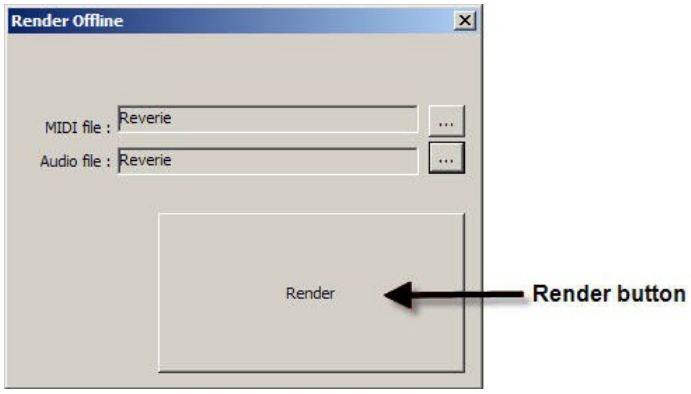
- *Load*—any saved configuration preset files in Aria format can be loaded by clicking on this choice and selecting the desired file.
- *Save*—any configuration can be saved by clicking on this choice, typing a name for the custom preset and saving to a desired location.
- *Save as default*—any settings can be saved as part of the default, to be loaded automatically at the time the chipsounds player is booted in standalone mode.

Tools Menu in Standalone ONLY



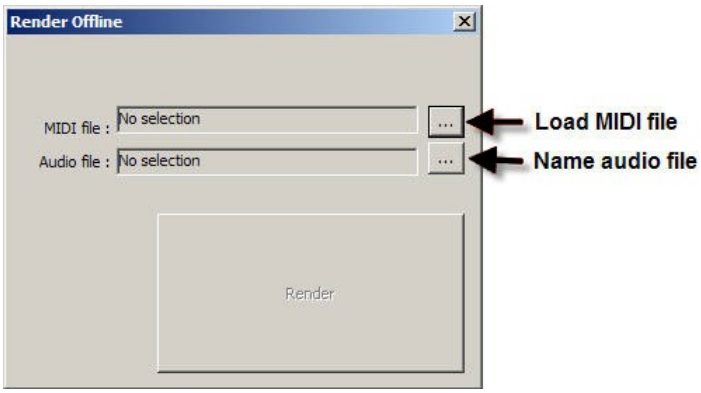
The Tools menu includes:

- *Preferences*—as described in the basic setup information above.
- *Render offline*—MIDI files can be rendered to audio offline using this feature.



To use the Render Offline feature:

1. Click on the Load MIDI file button.
2. Select the desired MIDI file

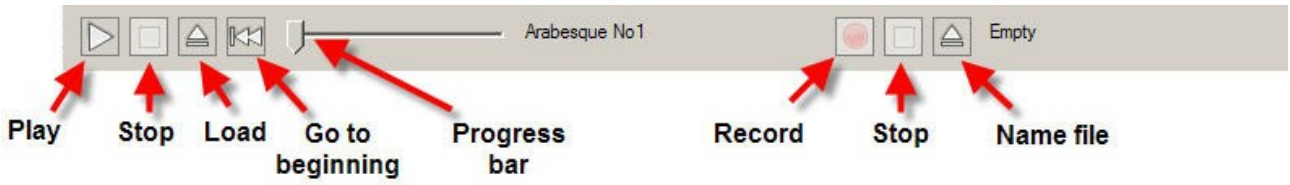


3. Click on the Name audio file button.
4. Name the audio file and specify its location.

Once the files are in place, there is just one more step:

Click on the Render button. The audio file will be rendered to the specified location.

MIDI File Player and Audio Recorder (Standalone Version Only)



The standalone version of the player includes a MIDI file player and Audio Recorder. It is located at the bottom of the standalone window. This feature enables the user to load and play a standard MIDI file and render an audio file from it. The MIDI controls are located on the left hand side and the audio controls on the right.

To load and play a MIDI file:

1. Click on the “Load” button.
2. Choose the file you wish to load.
3. Click on the “Play” button.

The MIDI file will play back using the presently loaded chip setup and any other features (e.g. Ambience reverb) activated in the player. The progress bar will move to the right as the file is played. To start the file from the beginning, click on the “Go to beginning” button. The file can be stopped at any time by clicking on the “Stop” button.

To record your playback to an audio file:

1. Click on the “Name file” button.
2. Name the file and specify its desired location.
3. Click on the “Record” button.
4. Start playback by clicking on the MIDI “Play” button.
5. When playback finishes, click on the MIDI “Stop” button.
6. Click on the Audio “Stop” button.

The audio file will be located on your drive in the location you specified.

Note:

There is also an offline rendering feature (explained previously) that is located in the “Tools” menu of the standalone version.

Using Plogue chipsounds as an Instrument Plug-In

When used as a plug-in, Plogue chipsounds is not a standalone program but rather a virtual instrument module that is seamlessly integrated into your favorite music software program or sequencer (assuming that it accepts such instrument plug-ins). They are called “plug-ins” because these are modular software applications that run inside a “host” music application, (*e.g.* a sequencer or tracker program, typically).

There are various uses as a plug-in:

- MIDI recording and sequencing of Plogue chipsounds.
- Audio mixing of Plogue chipsounds with other instrument tracks within a single program
- Easy automation of Plogue chipsounds parameters in the music software program through the use of MIDI CC's, which gives you the possibility of automating all 8 slots. Or using your host's “parameter automation”, which allows you to automate the first slot only.
- Effect processing of Plogue chipsounds using effect plug-ins in a music software program Saving and recalling of all plug-in settings when the music software program or sequencer file is reloaded
- Integration with other instruments into a “virtual studio”

A great thing about plug-ins is that they work with a large variety of compatible music programs. For example, Plogue chipsounds can be used as a VST plug-in in many VST music programs, sequencers, supported tracker programs and hosts. chipsounds can also be used as an Audio Units plug-in or as an RTAS plug-in.

Plug-in Standard	Description	Windows	Mac
VST	The VST plug-in stands for Virtual Studio Technology and was developed by Steinberg, the makers of the Cubase family of audio programs. It is also used by Cakewalk Sonar, Ableton Live, Renoise, Bidule, FL Studio and other sequencers.	X	X
Audio Units	The Audio Units (AU) plug-in standard was developed by Apple Inc. for Core Audio under Mac OS X. Audio Units is the preferred plug-in format on Mac OS X and is used by Apple GarageBand & Logic and MOTU Digital Performer.		X
RTAS	RTAS plug-ins (Real Time Audio Suite) are designed to work in the Digidesign Pro Tools environment. Pro Tools hardware and software are used extensively in the pro audio and post production communities.	X	X

Basic Setup Information for Using Plogue chipsounds as a Plug-In Instrument

To use Plogue chipsounds as a plug-in instrument, you simply launch your host music application/sequencer first and then launch Plogue chipsounds from within it. Make sure that your sequencing host program is properly installed and configured, and that it is producing sound properly. Used as a plug-in, chipsounds’s audio and MIDI data is managed by the host music software application.

Using Plogue chipsounds in a Specific Music Program or Sequencer

Plogue chipsounds works as a plug-in instrument within many popular music software programs. Each music software application has its own approach to handling plug-in instruments. They each have a different method of installation as well as differing means of loading and accessing plug-ins. It is important to make sure that you refer to the instructions in your music software application’s manual regarding the loading and operation of plug-in instruments.

Although it is not within the scope of this manual to delve into how plug-ins work for the various music applications, there will be tutorials on how to use Plogue chipsounds with the various music software programs. Please refer to the support pages on the Plogue website at www.plogue.com

Saving chipsounds Parameters in a Music Program or Sequencer

While using Plogue chipsounds with a host music application, when you save a sequence or project with the host program, you will save all of Plogue chipsounds' parameters as well (As long as chipsounds is properly registered with the current user). The parameter saving will occur automatically when you save the file in your music application and you don't need to do anything in the chipsounds Player interface for this to happen. When you re-load your host music project, Plogue chipsounds settings will revert to the state in which they were when you saved your project file.

Important Technical Notes:

Windows VST applications only: To use Plogue chipsounds with *more than one* VST application, you need to manually copy the "**chipsounds VST_x86.dll"** file, installed into the chosen folder during installation of the software, to the appropriate VST-compatible host application's VST folder. Please refer to your particular application's user's guide and the Plogue support site for more information.

Regarding 64-bit hosts: Some hosts have one common VST folder for both x64 and 32-bit plug-ins, please only use the version of the plug-in that is native to your host, *e.g.*: x64 bit version of Sonar, use the chipsounds VST_x64.dll. Mac OS X has standard folders for both VST and AU plug-ins and do not require this extra step.

An additional copy of the VST plug-in is available in the main Plogue chipsounds applications folder, in the VST subfolder. Please don't use this folder as your main "vstplugins" folder. Please note that a saved sequence in one music application may not be usable in other music applications, as each application generally has its own proprietary format. However if you can export and import standard MIDI files between two programs, then combined with .aria(x) files it can serve as a means to exchange tracks between them.

How chipsounds Works

The Basic Interface and the View Screens

The Aria Engine is a custom-made Virtual Instrument Engine developed specifically for Plogue and Garritan sound libraries and synthesizers, such as chipsounds. It constitutes the best sounding and most powerful sample/synthesis engine available, built from the ground up for high performance and exceptional quality. The Aria Player enables you to load chipsounds instrument sounds, control various parameters for playback, and do a number of other things. The software engine was custom designed and programmed by Plogue Art et Technologie Inc. The Aria Player has various view screens or "windows" that are accessed by the five view tabs across the bottom right. Those screens consist of the "Mixer" window, the "Controls" window, the "modulation", "Effects" and the "Settings" window.

Chip Music Dogma:

A good part of the dogma of composing chip music is about being creative with limited resources, something that is natural when using the original 8bit hardware. While we can't claim chipsounds *replaces* the original hardware, we can at least offer the possibility to emulate its inherent limitations.

Musicians willing to accept the challenge of composing music using the same limitations as they would have using 8bit console trackers need to follow certain guidelines and be mindful of certain key aspects of the graphical user interface parameters. We will note these in a special blue section like this one.

Similar cues exists in other types of sound libraries, for example with a drum sample library.

You would never hear a real drummer strike a cymbal, a snare and a tom at the exact same time unless he had a third arm.

The Mixer tab:

A. Slot Number (1 – 8): You can have up to 8 chips or instruments loaded in each instance of the player. An instrument can be the monophonic voice of a classic sound chip, a full sampled drum kit, etc. Click on a slot number to edit it's parameters in the controls and modulation tabs.

While the interface lets you load and configure 8 such voices, you should try and make sure you never have your sequencer play more than 4 or 5 voices play at once (chip dependent). Why do we provide 8 of them then? Simple! You might require a specific instruments only at certain spot in your piece. For example a bridge would use another waveform and arpeggio on voice "1" than the rest of the song. And by setting another slot with this configuration you are relieved of reconfiguring the current one while it plays, which is not always realistic in real-time

B. MIDI Channel Assignment:

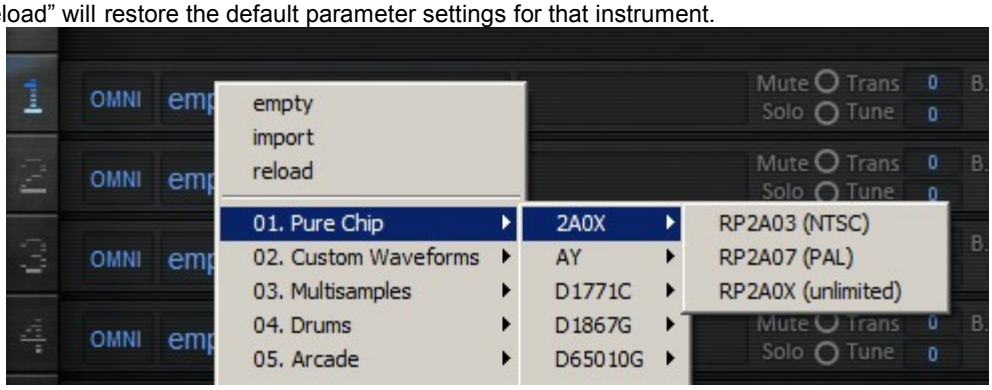
Click the number to select the MIDI channel that you require this slot to respond to. If you specify "OMNI" this instrument slot will receive all events, regardless of the source MIDI channel. In other words the program in the slot it will play all incoming notes.

C. Load Program: Click here to load a program in the corresponding slot.

Clicking the chip slots brings up a dialog box from which you can load the program of your choosing. The instruments will be grouped in a simple hierarchical menu for each ARIA-based Library you have installed.
The instruments are also placed in sub groups: "01 - Pure Chip" , "02 – Custom Waveforms" etc.

Choosing "Empty" will remove an assigned instrument from a channel and wipe all previous controller data received on this slot.
Choosing "Import" will allow you to load any standard SFZ 1.0 and SFZ 2.0 file, to use as an instrument sound.

Choosing “Reload” will restore the default parameter settings for that instrument.



NOTE: The chip/instrument slot is also mirrored in the controls tab for ease of access.

D. Key Switch

Current active switch on that slot (see Controls section)

E. Mute/Solo

A muted slot will be silent in the mix, a Soloed slot will mute all other non soloed slot.

F. Transpose / Tuning Settings

Click/drag the numbers to change the current slot from -64 to +64 semi tones, and -100 to +100 cents. This is very useful to thicken an instrument used as a layer in a more complex sound. **NOTE,** depending on the target chip/program and the specific note played, even a 100 cent change might not be enough of a change to generate a new frequency from that slot. (example: TIA (NTSC) has can only play 32 different notes spanning the full keyboard)

G. (top) Pitch bend range

Allows to change the default +-48 semi tone bend range to something more standard like +-2 semitone, or another setting you prefer.

G. (bottom) Polyphony Setting

Click the number to change the polyphony for this slot. If your goal is to use the arpeggiator, the wave sequencer or any other *monophonic trick*, leave it at the default “1”.
If you want to play full chords on one slot, then you can decide to raise this.

The original sound chips had very limited polyphony. If you decide to have one channel play sustained chords, then limit the number of extra slots playing at the same time elsewhere.

H. Volume Control Knobs (CC7)

This knob controls the relative volume level of the corresponding slot.
The volume of each possible waveform in a console has been tweaked to be right relative to the other waveforms that chip can produce. If you want to play multiple slots of the same chip, don't change the volume of the slot.

I. Pan Knobs (CC10)

This knob controls the effect sends of the corresponding channel.
Only the DMG-CPU instrument had panning, and even then it was either center, full left or full right, AND you couldn't automate panning without sonic artifacts

J. Send Control (CC91)

This knob sends a portion of this slot generate signal into the effect bus.

Leave it to zero all the time as there were no DSP effects on those early consoles, at least until the fourth console generation.

K. Output Selection

When using the multiple output VST or the AU version of chipsounds you redirect the output of each slot to any external bus or plugin of your host for extra processing.

L. Keyboard

The bottom of the window features a virtual on screen keyboard that indicates the range of notes that can be played on that instrument setting. When a channel with an instrument assigned to it is selected, a section of the keyboard will be highlighted. Keys that are being played will be shown in real time.

M. Key Switches (KS)

Keyswitches are certain reserved keys (usually the lowest octave on the keyboard) that are used to rapidly change from one type of waveform to the other without requiring loading a new sound. You can also change the current waveform (or KS) in the controls tab chip section, or using MIDI Program Changes.

KEYSWITCH TIPS

- Always put the keyswitching note for the particular instrument *before* the first note of the articulation you want to play, not on the same time!
- Remember that when you hit "play" on the sequencer, ARIA will remember the last keyswitch used.
- If you transpose your score, you must be sure NOT to transpose the KS notes!! Any transposition to these notes will change (or eliminate) their function.

N. Snapshots

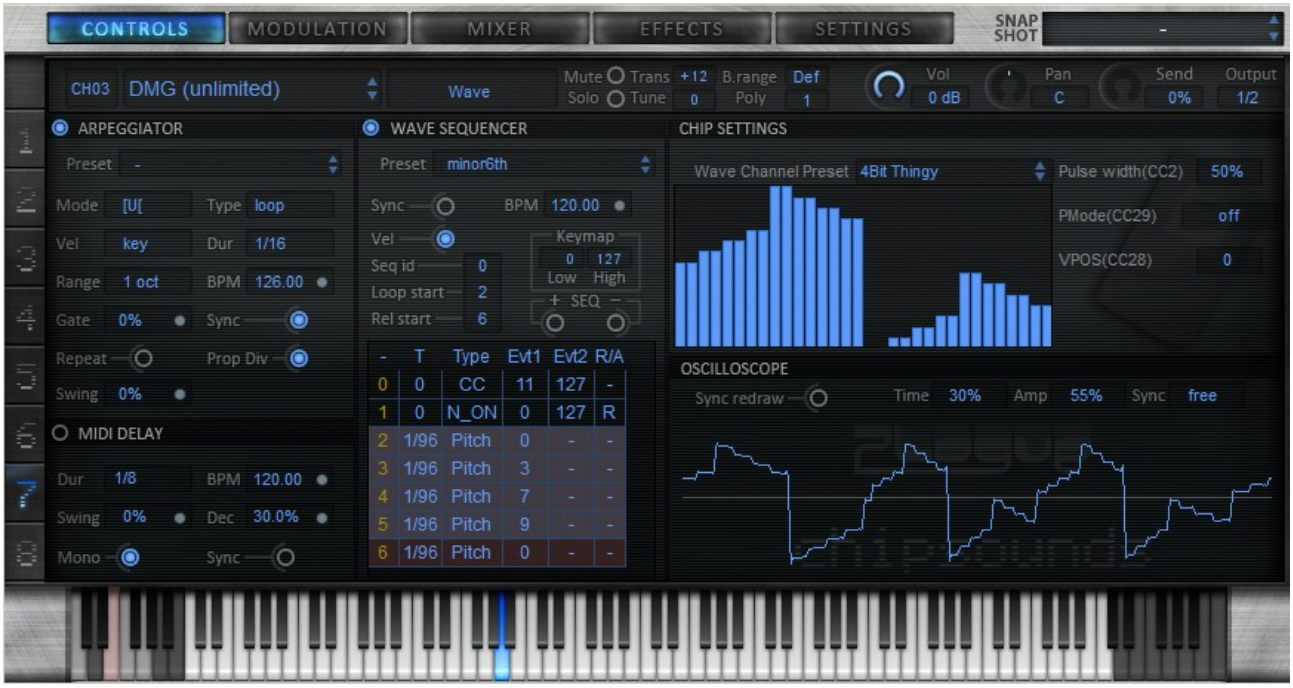
Snapshots are “Master Programs”, or complete multi-slot setups with a “snapshot” of the current sate of the system at the moment of saving, including parameter values (CC/pitchbend), arpeggiator and wave sequencer settings. You can save any number of snapshots into your user folders which mimic the structure of the the default programs.

The Snapshot selection box is always available regardless of your currently selected tab.

You can reload snapshots starting at ANY slot, that way you can use more than one “preset” at once.

Snapshots have the .aria or .ariax extension.

Controls Tab : The main sound Editing guts of a chip



Channel Strip:

All the parameters from the mixer view have been transposed here for the selected slot. Some were in the Chips Settings section of chipsounds version 1.0: The **Load Program** (see the Mixer section) and the **Key switch** drop downs.

Key switching (or "**KS**") is used to manually change waveforms by only using the keys. (It's a technique used by old organs, and more recently by many large sample libraries – including the whole Garritan offering, and it's not the typical thing you find on software synthesizers. **KS**'s act as a *kind-of* program change to rapidly change the waveshape of that chip without reloading the whole chip/slot.

Basically, if you have big MIDI controller (61 notes and higher), you can hit the lowest octave – which in everyday use is pretty useless-to act as a fast live toggle for sound variations.
Hit a low "E" (MIDI Note 004) and change the playback sound from pulse to the triangle waveform on the (2A03) setting.

Key switching provides some great advantage for MIDI automation in your sequencer, and for automation through the built in wave sequencer module. As it doesn't reload anything, it"s instantaneous and causes no CPU hit.

Arpeggiator:

When “on” this Component acts as a MIDI pre-processor transforming notes and chords struck on the your MIDI keyboard (or prerecorded sequence) into a stream of fast monophonic MIDI patterns.
This is a great live inspirational tool, but it has some important “gotcha's”.
Since each note generated from the arp is in fact a full “new” note, this will RESTART the envelopes and LFO's on each voice that plays in succession.. (So this is not what you would typically use for typical chip music fast arps)

On/Off: (CC20 OFF:0. ON: >=1)

Arp Modes: This, along with “Octave Range”, and the set of MIDI notes that are received will generate a new sequence of MIDI events at the outputs.

- **[U[:** Upward exclusive.
- **[D[:** Downward exclusive.
- **[U ; D[:** Up and down exclusive.
- **[U ; D] :** Up and down inclusive.
- **[D ; U[:** Down and up exclusive
- **[D ; U] :** Down and up inclusive.
- **Key :** Notes are arpeggiated in the order they are played.
- **[rdm[:** Random exclusive.
- **[rdm] :** Random inclusive.

Trigger Type:

- **loop**: will loop through the entire sequence until notes are released.
- **single**: will play through the sequence once.
- **sust. last**: will play through the sequence one and keep the last note playing until you release keys.

Vel (velocity): Affects the velocity of the output notes from the sequence.

- **key**: will play back the velocity of each input note in the output sequence.
- **1st:** will apply the velocity of the first received note to all the output notes.
- **norm**: normalizes all output notes to the same value.

Dur (Duration): length of each output note in musical time. (¼ being a quarter note).

(Octave) Range: 1, 2 or 3 octaves copies of each note are going to be added to the sequence.

BPM: Manual tempo selection. (independent of host tempo).

Gate (length): proportion of silence between successive notes.

(One Note) Repeat: when receiving only one input note output note once or repeat it.

(Host) Sync: Will follow the host's tempo.

Prop Div (Proportional Division): Sequences will take the same time to play, however manynotes they contain.

Swing: Adds groove to the timing of the events.

MIDI Delay:

When “on” this Component acts as a MIDI pre-processor that inserts copies of notes at lower velocities when no other notes are playing (keeping polyphony to 1). This allows to reproduce another a classic chipmusic trick to make more with less voices. Of course you can also bump up the polyphony and turn it into a polyphonic MIDI Delay as well.

On/Off: (CC22: OFF:0. ON: >=1)

(Dur) Duration: length of each repeated note in musical time (¼ being a quarter note).

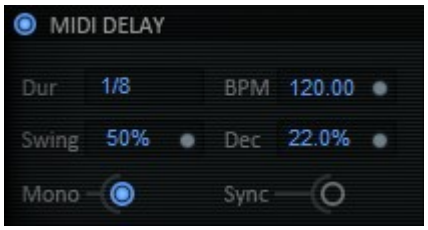
BPM: Manual tempo selection (independent of host tempo).

Swing: Adds groove to the timing of the repeats.

Dec. (Decay): Sets the number of repeats. A lower value will yield more repeats.

Mono: Monophonic delay. Each new note played will stop repeats from the previous note.

(Host) Sync: Will follow the host's tempo.



Wave sequencer: (WS)

This is the second MIDI pre-processor in the chain, and it's the real power horse of the system.
It's in many ways similar to the “instrument” tool on typical 8bit trackers.

It's a spreadsheet-like grid with musical and parameter events that are sequenced from top to bottom.
What it does is that it waits for an incoming note and then start sending elements form a pre-programmed list one at a time. This is what you would use in order to reproduce many typical chip music tricks like fast looping arps, or other custom frequency or amplitude modulating tricks. This module sends timed parameter changes to the current voice that is playing. It doesn't force reset the LFO's nor the pitch or amplitude envelopes like the basic arpeggio does. However you can use it to create a list of notes too.

A good selection of presets are already available in preset drop down menu.

Make sure a chip is loaded in slot 1, then click the wave sequencer's preset menu and load the one named “Classic Arpeggios/major7th”, then click on the upper left corner “off” button to turn it “on”, press a key on the keyboard, you should hear a major7th chord playing.

A wave sequencer can hold up 12 different sequences which can be mapped to any key (or continuous range of keys) on the keyboard. But you can only play and edit one at a time.

On/Off: (CC21 OFF:0. ON: >=1)

BPM: Manual tempo selection (independent of host tempo). [10; 500] default:120 (can be synced to host)

Seq Id: [0; 11] manually chose the EDIT sequence.

Key map: [0; 127] set note trigger range for this sequence. Note, the wave sequencer will play the first sequence whose range fits the incoming note. EG, if sequence #0 and #1 both span then entire keyboard, then only the sequence #0 will ever be playing.

SEQ+: increments the current EDIT sequence.
SEQ- : decrements the current EDIT sequence.

A sequence can contain 3 separate sections:

1) the **black section:(Start)** events in this section will only play at the start of the incoming note.
2) the **blue section: (Loop)** will cycle until the incoming note is released.
3) the **orange section:(Release)** will play after the note has been released.

A sequence can have any combination of such sections, but they will always be in that order.

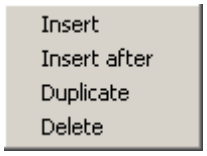
To Specify a **blue section (Loop) :** Set *Loop start* to something else than "--".
The Loop end is implicitly interpreted as either the last element in the list, or the Release Start marker, whichever comes first.

To Specify a **orange section:(Release) :** Set Release *start* to something else than "--". The Release section is a handy tool to reset/undo some of the changes that have been made to the system in the previous sections(amplitude, frequency, ADSR, etc), or to play a few notes at the end... warning, notes will hang until next note starts or wave sequence is turned off if the slot's ADSR's sustain is not 0.

Vel On/Off:
when OFF: the WS disregards any input MIDI velocity and overrides it with the velocity defined by the notes in the sequence.
When ON: the WS multiplies the input MIDI Note Velocity with the values specified in the table.

WS GRID EDITING

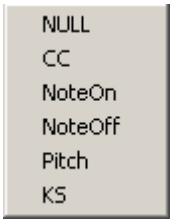
Column "-" : (Event ID)
To add an event, left click on an existing element from the leftmost column. This will prompt you a choice of action:



Insert: inserts a NULL element BEFORE at the selected index, with a duration that matches the one on the selected index.
Insert after: inserts a NULL element AFTER at the selected index, with a duration that matches the one on the selected index.
Duplicate: inserts a copy of the selected element.
Delete: deletes the selected element.

Column 'T' : (Musical Duration)
Changes the duration of the current event, in other words, or how long it will take for the next element to play, in standard musical durations: ¼ being a quarter note.

Column 'Type' : (MIDI Event Type – AKA MIDI Status)
Click on this column to bring forth the following menu:



- **CC** is to modify any CC seen in this interface including everything on the modulation tab and in that chip's custom parameters.
- **Note On** and **Off** are used to start and close notes:If you leave the slot's polyphony to "1", you don't have to close the notes that you left opened, since the previous note will automatically be "stolen" by the next one. If you do use extra polyphony mode, then you can create sequences of polyphonic patterns as you wish, but it is not a common usage scenario.
- **Pitch** send a private CC (12) that will tell the voice to pitch shift up to 2 octaves up or down in semitone steps.
- **KS** Changes the current Key Switch. Handy for SID-style wavetables like snare drums.

Column 'Evt1' : (MIDI Evt1)
Depending on the MIDI Event type, different values will be possible.

Column 'Evt2' : (MIDI Evt2)
Again like with Evt1, different values will be possible depending on the Event type. You can here SHIFT-CLICK to enter direct values using your keyboard. **-> Only used for Notes and CC's.**

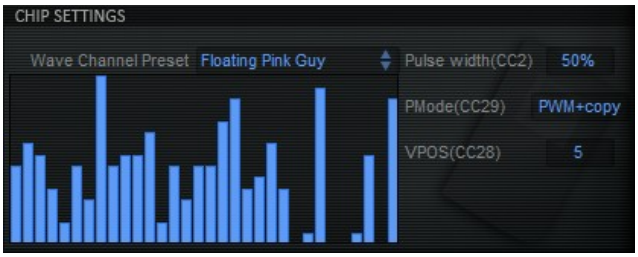
Column 'R/A' : (Relative/Absolute)
Absolute: Makes sequenced notes fixed whatever note you pressed.
Relative: Makes sequenced note relative to the input note (delta mode).

You can combine the two MIDI processors in unique ways for totally new sound patterns, especially considering the possibility to have each of them follow the host tempo or to follow their individual independent BPM values. (abusing the gate setting on the arpeggiator is cool, too.)

Chip-Specific Settings:

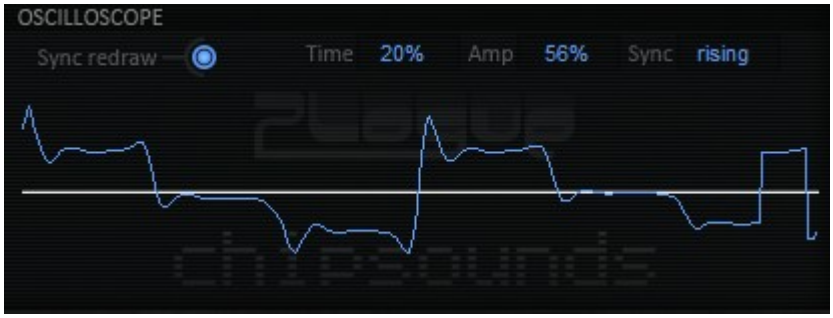
Some chips like the SID, have up to 6 specific controls. Some have none.

Each chip/program will be described in detail in the next chapter.

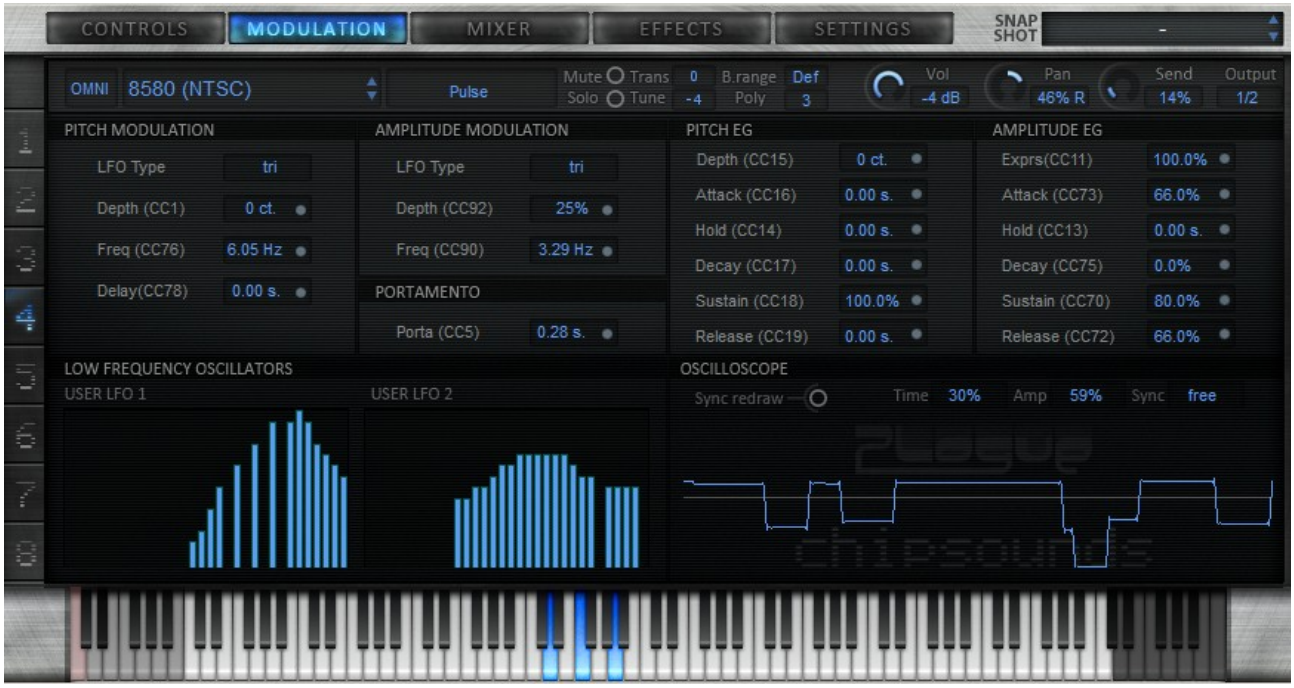


Oscilloscope:

Just a fun graphical representation of what you are hearing.
You can tweak the settings to change the time/amplitude scale and change the way it refreshes, just like a real one.



Modulation tab



User LFO's:

You can edit custom LFO waveshapes graphically with the mouse. Assignable to either pitch or amplitude.

Pitch Mod(ulation):

Pitch LFO with fixed waveshapes and two user drawable LFO's. You can control the LFO by automating CC's 1, 76 and 78.

Amplitude Mod(ulation):

Amplitude LFO with fixed waveshapes and two user drawable LFO's. You can control the LFO by automating CC's 92 and 90.

You have also access to a portamento time setting to glide from the past note to the next
(sorry no monophonic synth legato action yet)

Note on porta: the glide will be from the last TWO FINAL NOTES, so if you are using the arpeggiator/wave sequencer you may not get what you intuitively thought you would)

Pitch EG (Envelope Generator):

Standard pitch envelope. Each control can be automated with CC's 14,15,16,17,18 and 19.

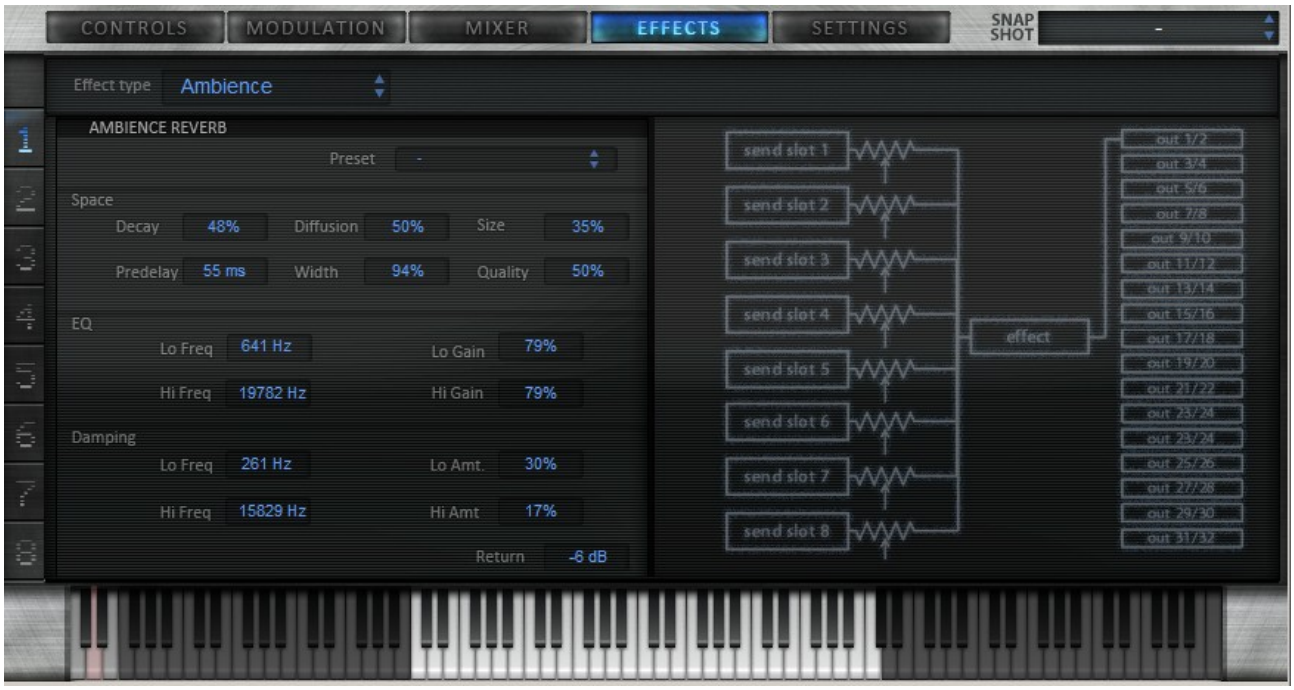
Amplitude EG (Envelope Generator):

The displayed times are valid for all chips except the SID's, which have different time calculations that currently don't show on the UI. Also Most chips can only be driven by 16 volume steps. while the SID supports more fluid decays. CC's are 13,73,75,70 and 72

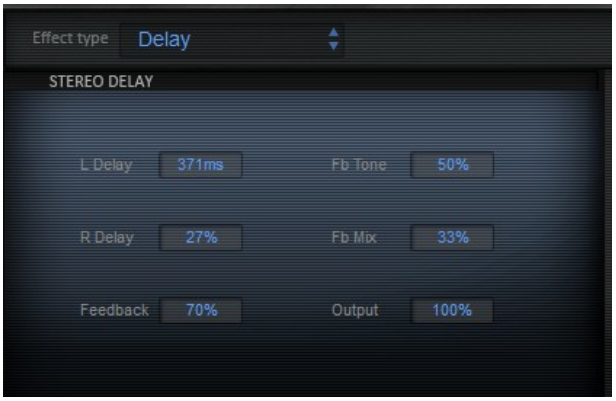
Expression (CC11) is reserved for Wave Sequence for quick volume edits and is only available here for troubleshooting purposes.

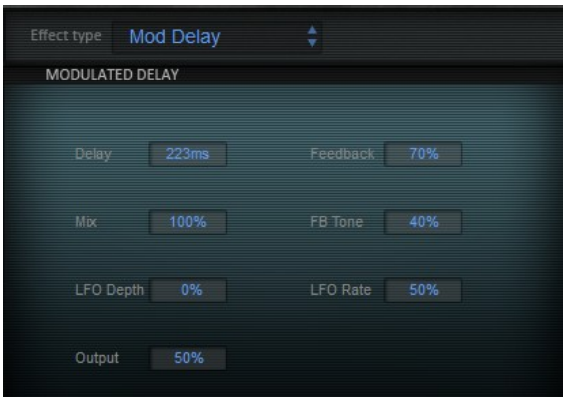
Effects tab:

The default master send effect is the Ambience reverb:



New in 1.6, two new effects Delay and Mod Delay... More to come!





Settings Tab

The Settings tab brings up additional information about the current ARIA Engine instance.
This is mostly there as an ARIA engine *troubleshooting* display. It displays the version of the software and its copyright information as well as information about how the ARIA engine is performing. The information in the left-hand column indicates the software's current RAM usage. The middle column contains the software's current resource pool status.

You *can* adjust some of the settings in this window by clicking on certain boxes to reveal a drop-down menu,
but for chipsounds, you shouldn't have to mess with this at all. Well you **could** import standard SCALA files and re-tune your "unlimited" TIA back into a gamelan though!

- **Dyn Max:** Determines the threshold for RAM usage by the software. The default is "256 MB."
- **Pre-Caching**: Determines how many samples of an instrument the software will process before playing them. A higher setting will result in longer initial load times, though you should increase this setting if you are using a slower computer. The default is "32 Kb." (this only applies to other third party instruments, not to chipsounds)
- **Quality**: Determines audio quality of the sample interpolation. The default is "Normal (Hermite)."
- **Tuning System:** Determines the tuning system of the instrument in the software. There are different tunings for regions throughout the world, but to start, we recommend the default, "International 440."
- **Inst. Poly:** Determines the maximum amount of polyphony (multiple notes sounding simultaneously) each instrument can play. A higher setting will allow for greater polyphony but also increase CPU usage. The default is "32." This setting applies for ALL slot... It's a Master cap.
- **Scala:** Selects a specific tuning file.
- **Scala center**: Select root note for the scale
- **MIDI Out: (VSTi only)**: Specify what to output back to the plugin *Mute*: don't output anything. *Processing*: This will output ALL MIDI processed by each slot and output them on MIDI channel tagged by each slot (not linked to Omni/channel settings). Including notes hit on the GUI and arpeggiator/wave sequencer etc. *Through*: Just forwards inputs to outputs.

Generic Notes on all ARIA-based UIs:

*All virtual knobs on the graphical user interface can be adjusted by holding down the left button of your mouse and moving it up or down (rather than circular, in the direction of the knob). Release the button to set the level. A box next to it will give a measurement reading.

Most of the knobs have been replaced with numeric boxes. There are two ways to modify the values:

Click on the pencil to manually enter the desired numerical value.

Click and drag up or down on the number, like you would with a regular knob.

*All virtual Knobs can be reset to their default value by using Ctrl+Click [Command+Click (Mac OS)].

Skinning:

No UI will ever please everyone. You can create your own skin by modifying the GUI directory.
If you can edit HTML layouts, and mess with Adobe PhotoShop, you will surely have no problems figuring out the ARIA xml GUI tags. In case of severe doubt, please post on the chipsounds forums for details.

chipsounds includes a separate skin “GUI_smaller” that can be substituted to “GUI” in the AriaSetup.xml file.
This is suggested for netbook users.

Important Notes regarding Pitch Resolution

With the exception of Bob Yannes' SID, most of these chips were not made by musicians, and some of them have a hard time reproducing western musical scales .(The 2600's TIA cannot do any *useful* scale for that matter).

There many parameters in chipsounds that can influence the target note frequency:

A)Actual Note that's been hit,
B)Pitch bend value (2 octaves in both direction),
C)Portamento/Glide time,
D)Pitch LFO,
E)Pitch EG,
F)Value of the detune setting for that slot in the mixer pane,
G)Value of the hidden CC12 parameter (wave sequencer's pitch modulator),

But ultimately the emulated chip will only play the ***CLOSEST FREQUENCY*** it can with regards to the final calculated pitch value generated by those parameters.

But ... This is where the fun resides, and a good part of the sonic authenticity factor.

Doing a pitch bend or some other heavy pitch modulations on a severely limited pitch precision chip like the TIA is part of what makes the 2600 sound effect so awesome... Try slowly pitch-bending a POKEY... you will be surprised of the amount of spectral content that change with only a few "*musician's cents".* In this context, pitch LFO's can be used as great beat generators.

Of course you don't *have* to suffer this if you are just looking for the classic waveforms and not the off key notes, beating and stepping.

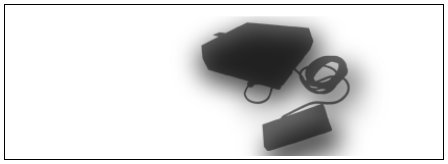
That's why we've added an "unlimited" version to most Chips in the list.

Chips, Program, and instrument details.

Most of these chips come as different slot "programs" and usually differ in subtle ways.
The most common difference is the master clock that drives the chip which is responsible for its exact tuning and pitch limits. NTSC stands for *National Television System(s) Committee,* and is the North American and Japanese analog television standard. PAL stands for *Phase Alternating Line* and is the European system.

An "**unlimited**" chip is a **fantasy invention**. It allows perfect tuning all the time. So it doesn't give an accurate representation of the normal pitch that these chips can produce.

RP2A03 (NTSC) and RP2A07 (PAL) and RP2A0X (unlimited)



This 6502 cpu clone was custom made by Ricoh to be included in the Big N's breakthrough 8bit console.
On top of being the main cpu for the console, it also served as the sound generating unit of the system.
While is it is not a dedicated audio IC, it is still quite remarkably versatile and unique sounding.
Two audio outputs pins were provided on the IC:

The first audio pin generated two independent pulse channels, with 4 width settings:
1/8, ¼, ½ and ¾ (¾ and ¼ being inverted in phase).

The second pin generated
1 - a triangle wave (4bit resolution) which was often used for baselines.
2 - noise (LFSR-based, short and long).
3 - DMC (Delta Modulation Channel), a crude sample playback mechanism.

chipsounds maps those sound sources into 5 distinct KS slots:

Pulse MIDI Note 0 (C0):
Extra parameter: CC2 : Dynamically change Pulse Width from 1/8, ¼, ½ and ¾.

Triangle MIDI Note 4(E0):
NOTE: amplitude EG has NO effect on this waveform since technically the waveform is the result of volume changes to the 4 bit DAC already.

Short Noise MIDI Note 5(F0):
Metallic sounding tone.

Long Noise MIDI Note 7(A0):
This generates a more evenly distributed frequency content than the short noise. A good sound source for snares and hi-hats.

DMC Channel MIDI Note 9(B0):
Due to obvious copyright reasons, no classic samples are provided, only a snare and bass drum sounds from Plogue's sample collection. You can add more samples to this switch by modifying the “Programs\09. User\Uuser1.sfz” file, and by adding your own 8bit unsigned RAW samples references inside.

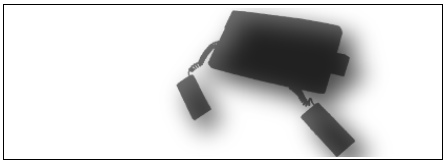
*NOTE1: DMC's, due to their nature of having preset pitches depending on the master clock, do NOT work correctly in “unlimited” mode. (they will just sound like chipmunks.)

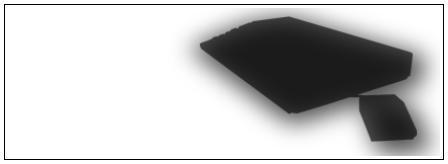
*NOTE 2: While all 2A03 tones are synthesized in this program, an additional program has been constructed as “extra” and is available in the “03 Multisamples” category:

2A0X Loops: Some fun Noise sweep samples recorded from our 1U NES rack and custom EPROM made with NESASM . Trying tho replicate this with the Wave Sequencer would be a interesting task, as each of the 16 noise frequencies is a table in the 2A03.

TECH: These are a 60Hz increment/decrement sweeps on $400E. (short and long.)
Sweeps sound different depending on the timing of 1/60's second changes which is why we've provided a few different ones here.

AY-3-8910 (various clocks sources), YM2149 (2Mhz)





The **"AY"** is a dedicated sound generator by General Instruments (now Microchip). It's an OEM part that has been included in hundreds of arcade games, consoles and computers. It has also been manufactured with a few changes by Yamaha as the YM2149 chip (envelope has 32 amplitude steps instead of 16), to be included in the ST line of computers.

It can generate 3 channels of 50% duty pulse at once, on three separate pins.
A separate LFSR-based noise generator can be mixed into each of these 3 separate outputs.

On top of that, a single incorporated AR (attack/release) envelope generator can modulate the levels of the 3 channels independently (NOTE: since there is only ONE AR envelope generator, you could use ONE envelope setting for all channels).

Recently on the ST platform, some new tricks have been implemented by Demo/Tracker programmers which uses the envelope generator as a new waveform type. This is known as the "Hardwave". A variant on the Hardwave is the "Sync Buzzer" technique which consists in carefully timed resetting of the Hardwave, which produces on top of a better frequency response than the bare Hardwave, a truly unique hard-sync effect using only the Envelope's waveshape.

chipsounds maps those sound sources into 6 distinct KS slots:

Square MIDI Note 0(C0)

Long Noise MIDI Note 2(D0)

Hard Wave 1 MIDI Note 4(E0): Rising envelope loop.

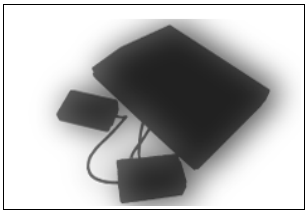
Hard Wave 2 MIDI Note 5(F0): Falling envelope loop.

Hard Wave 3 MIDI Note 7(G0): Rising/Falling envelope loop.

Sync Buzzer MIDI Note 9(A0): Rising/Falling envelope loop with resynch.
Extra parameter: CC71 : Dynamically alter the resetting timing of the envelope. (hard sync)
This uses CC71 since it's is often used as a filter cutoff/resonance frequency emulating trick.

NOTE Veteran AY programmers know that you can create a vast amount of cool sounds using combinations of the envelope and the pulse+noise mixed together. We are working on an update which will allow you to do some of those tricks See progress here: http://ploguechipsounds.blogspot.com/2009/10/full-ay-emu-prototype.html

D1771C Sound Generator of the Super Cassette Vision



Nearly nothing is known about this 'soundchip', apart from the suspicion that it is in fact a NEC 4bit MCU, or if you prefer, a microprocessor with embedded code on it. Since it runs a custom program that we don't have access to (until this chip gets de-capped), we can only approximate what it generates using empirical tests.
My research led me to create a small application that runs directly on the console:
(http://ploguechipsounds.blogspot.com/2009/08/upd1771c-tester-app.html)

This allowed me to capture and analyze the 8 basic 'pitched' waveforms it can generate with relatively good accuracy.
I then added those sounds into chipsounds and contributed them into MESS as well.

The main tone sources are:

D1771C_T0	MIDI Note 0(C0)
D1771C_T1	MIDI Note 1(C#0)
D1771C_T2	MIDI Note 2(D0)
D1771C_T3	MIDI Note 3(D#)
D1771C_T4	MIDI Note 4(E0)
D1771C_T5	MIDI Note 5(F0)
D1771C_T6	MIDI Note 6(F#0)
D1771C_T7	MIDI Note 7(F#0)

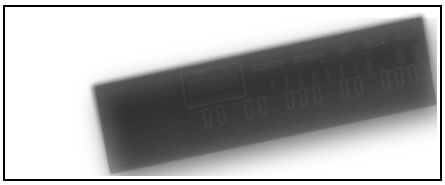
(you can use CC2 on these to change the offset into these 32 stepped waves)

The main noise sources are:

D1771C_N0	MIDI Note 8(G#0)
D1771C_N4	MIDI Note 9(A0)
D1771C_N5	MIDI Note 9(A#0)
D1771C_N7	MIDI Note 9(B0)

Other bits of research:
(http://ploguechipsounds.blogspot.com/2009/07/rom-trojaning-super-cassette-vision .html)
(http://ploguechipsounds.blogspot.com/2010/05/upd1771c-puzzle .html)

D1867G multipulse sound generator from the VL1 Mini keyboard



While not from a vintage video game console or computer *per se*, this chip produces very similar results and has gained good notoriety in the chip music scene with its totally unique waveforms.

The IC has multiple outputs, some of which are heavily post filtered.
Other outputs (used for beats) are unfiltered.

chipsounds maps those sound sources into 6 distinct KS slots:

Piano MIDI Note 0(C0)
Fantasy MIDI Note 1(C#0)
Violin MIDI Note 2(D0):
Flute MIDI Note 3(D#):
Guitar 1 MIDI Note 4(E0):
Guitar 2 MIDI Note 5(F0):
English Horn MIDI Note 6(F#0):
For these switches, you can remove the RC filter with CC79

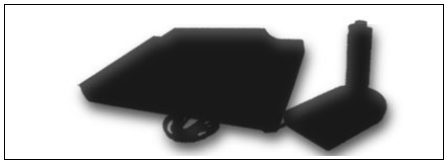
Drum Snare MIDI Note 7(G0):
Drum Noise MIDI Note 8()G#0

NOTE1: these are the names of the waveforms, not their equivalent “preset” on the mini keyboard, which ofte contain tremolo/vibrato as well. In order to reproduce those presets you have to program them using the modulation tab and the Wave Sequencer.

NOTE2: the frequency range of the chip is currently unknown, so we only provided an “unlimited” version.

NOTE3: The envelope on the drum channel is not stepped but a smooth exponential decay.

NEC D65010G031 3-Micron CMOS Gate Array used in the PV-1000



Nothing much to say, really only 3 square wav channels made by cutting the main clock by 1024,
then by the value of the 6bit period registers. But this creates a surprisingly accurate pitch range.

DMG-CPU, SGB and DMG(unlimited)

This CPU is another custom built CPU/Sound generator, this time manufactured by the “Big N” themselves for use in their very famous portable 8bit console. While this time it's a Z80 clone running much faster than the previous 2A03, its sound capacities really show a direct conceptual heritage.

The pulse channels have equivalent 1/8, ¼, ½ and ¾ widths
The noise channel has a different short noise (127 bits long instead of 93), but the same Long noise.
However it does not have a DMC channel, instead it has a Wave channel that can be configured to play any 4bit waveform of 32 samples. By rapidly changing the content of that register, longer samples could be played.

chipsounds maps those sound sources into 5 distinct KS slots:

Pulse MIDI Note 0 (C0):
Extra parameter: CC2 : Dynamically change Pulse Width from 1/8, ¼, ½ and ¾.

Wave MIDI Note 4(E0):
You can edit the waveform graphically with the mouse or choose from a drop down of presets
NOTE: amplitude EG has NO effect on this waveform since technically the waveform is the result of volume changes to the 4 bit DAC already.
New in chipsounds 1.5 Although the native hardware doesn't offer these features directly, three other MIDI CCs can now be used to further tweak the sound of the Wave Channel of the DMG, not unlike what can be done with contemporary DMG tracker cartridges.

CC2: Change the pulse width of the waveform, the effect of which depends on the Pmode
CC28:VPOS: changes the vertical offset of the waveform. (with wrap around)
CC29: Pmode. (phase mode)

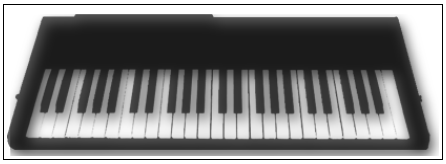
- off :no phase effect
- PWM :waveform is 'squeezed' into the left hand side.
- PWM+copy :same as PWM, but copies of the waveform are inserted after. (nasty)
- modulo :modulus of width is applied to the playback position
- seek :playback position speed is changed (**NOTE** pitch will not be preserved!)
- trunc :the waveform is truncated towards the width mark.

Other modes may be added in future versions, so the upper range of CC29 is not used.

Short Noise MIDI Note 5(F0):
Metallic sounding tone, but used in many games as a gritty springy effect.

Long Noise MIDI Note 7(A0):
This generates a more evenly distributed frequency content than the short noise.
Good sound source for snares and hi-hats.

M5232 (unlimited)



A very interesting chip specimen which appeared in a few obscure arcade games (Splendor Blast) but also in a few K_rg synthesizers. It generates tones by means of additive squarewave synthesis. Just like an organ, but using square pulses instead of tone wheels. (think 60's Combo Organs). The chip also permanently outputs a constant frequency LFSR-based noise waveform.

Just like the SID, this chip requires a fair amount of external components in order to be properly utilized. This of course has a great influence on its overall sound. For instance, external capacitors are required to allow the chip to generate amplitude envelopes. So depending on which arcade board or synth you have an M5232 on, the ADSR would have to be programmed differently. We based our emulator on its use in the K_rg Poly 800, including its external filter.
With a bit of work, it would be possible to recreate some of the Poly 800 presets using it, and we have added a few example presets to show it.

One thing that we removed is the limitation on the relative volume of each harmonic. The original chip allowed you to turn each of the 4 harmonics on and off, and to put them in either Square (equal volume), or Sawtooth (relative volume) mode. Here we allow the possibly to set the 4 harmonics to any level.

Original Harmonic Levels:

	Square	Sawtooth
CC024: 16' volume	0dB	0 dB
CC025: 8' volume	0dB	-6 dB
CC026: 4' volume	0dB	-12 dB
CC027: 2' volume	0dB	-18 dB

For CC minded-people, sawtooth harmonic levels correspond to: 127, 61, 31 and 15.

To access the Poly-800 filter settings:

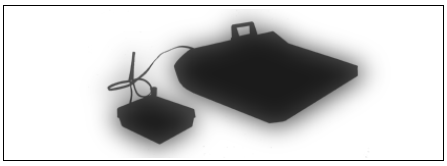
Base values:
CC071: Filter Resonance
CC074: Filter Cutoff Frequency

EG Modulation:
CC082: Filter EG Attack Time
CC083: Filter EG Decay Time
CC084: Filter EG Sustain Level
CC085: Filter EG Release Time

Differences with a real Poly800:

Octave should be emulated using chipsounds'' transpose.
Each of the two DCO's should be placed its own slot for layering.
Currently Filter LFO, and Chorus are unavailable.

P8244 (NTSC), P8245 (PAL)



This is a custom made Video Display Controller by Intel which on top of generating the the graphics for the O2, also generates its sound. The lone audio pin that can generate either a tone or a noise at once.
The pulse channel can only generate frequencies in the “key” of E5, meaning only E's and B's.

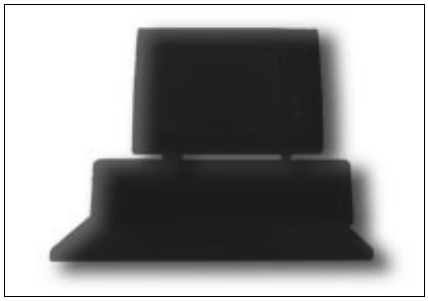
Pulse MIDI Note 0 (C0):
Extra parameter: CC2 : Dynamically change 24 bit pattern variations with some precomputed ones.
See the research blog for details on this.

Long Noise MIDI Note 7(A0):
This generates an evenly distributed frequency content.
Good sound source for snares and hi-hats.

NOTE: Due to the nature of the way this chip generates audio, an unlimited version would not make much sense.

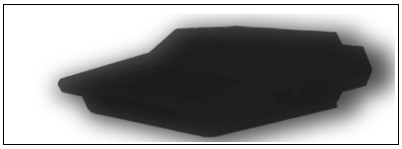
More info:
http://ploguechipsounds.blogspot.com/2009/10/last-minute-addition-odyssey-2-p824x.html

P8253 Intel 8253/8254 Programmable Interval Timers (PIT)



This is the chip that generated the infamous BEEPs of PC Speakers of old DOS PCs. While by itself it is nothing really interesting, this program in chipsounds embeds one of the PC Speaker impulses that will be found in the upcoming 'chipcrusher' product.

POKEY (various clock configuration)



The POKEY chip (aka C012294) is a multipurpose integrated circuit used in many arcades and in the 400 and 800 series 8bit computers. It can generate either 2 or 4 voices at once depending on software configuration.
In the 2 voices configuration, each channel has a 16bit frequency resolution, which gives it a very good frequency range.
In the 4 voices configuration, each channel has a 8bit frequency resolution, which gives it a poor frequency range.
(you can also use one 16bit channel combined with two 8bit channels)

However the most common usage in games was in 4 voice configuration, unless the music was a crucial part of the required experience.

The POKEY is a very unique chip which generates multipulse patterns in a very dynamic and unpredictable fashion.
This is due to the fact that the chip internally continuously generates various lengths high frequency polynomial counters which are “resampled” by a slower clock which is defined by the required music pitch. Depending on the alignment and configuration of the various clocks and required playback frequency at a given moment, you can get anything from a pure tone, to noise, raw grinding pattens to silence.

This one is particularly interesting to experiment with, as changes in pitch, and even re-triggering the same key can generate different results.

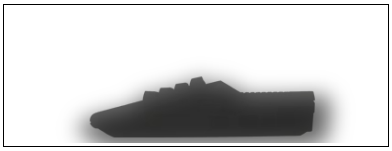
chipsounds configures the internal polynomials using these KS:

000 (5bit->17bit)	MIDI Note 0
000 (5bit->9bit)	MIDI Note 1
0X1 (5bit)	MIDI Note 2
010 (5bit->4bit)	MIDI Note 3
100 (17bit)	MIDI Note 4
100 (9bit)	MIDI Note 5
1X1 (pure)	MIDI Note 6
110 (4bit)	MIDI Note 7

The three digits represents bits 7,6 and 5 of the POKEY AUDx register.
The information in parenthesis explain the polynomial bit “resampling” signal path that takes place in the chip.Don't worry if you don't understand what it means, just use your ears and experiment.

More info:
http://ploguechipsounds.blogspot.com/2009/10/how-i-recorded-and-decoded-pokeys.html

SID (6581) and (8580)



This chip really requires no introduction. It's by ***far*** the most famous of all music chips in the world.
It is also by far the most complex analog/hybrid chip of the lot, and accordingly, the toughest to emulate. Originally designed as a OEM chip that could be used in commercial synthesizers, it became instead irrevocably associated with the C64 due to the incredible success of the later. Many versions of the chip exists, spanning the 10 year production of the C64, and each has their own subtle differences. However, most specialists agree to separate them into two main classes: The 6581 (R2,R3,R4, etc) and the 8580 (8580, 6582).

If you don't consider the on board filter which varies a lot even between weeks of manufacturing, the major difference between the 6580 and the 8580 classes is the sound of their *combined waveforms.*

Various approaches have been attempted over the years to capture this chip in its virtual form, some good, some bad, some truly worth a whole chapter in itself (the resid/fp projects). However our goal here, contrarily to the aforementioned project is not to play back existing .SID files so they sound exactly like the originals, but instead to allow a musician to tap in its powerful sound capabilities and to enjoy the *exact tone* of its combined waveforms, which have been carefully sampled at 96Khz for this project.

chipsounds maps the SID sound sources using the following switches:

Pulse MIDI Note 0(C0) (synthesized)
CC2 changes the PWM value, while CC3 and CC4 allow to apply an LFO to this value.

Noise MIDI Note 1(C#0) (synthesized)

Triangle MIDI Note 2(D0) (synthesized, samples are provided but are not currently mapped)

Saw MIDI Note 4(E0) (synthesized, samples are provided but are not currently mapped)

_ST MIDI Note 5(E0) (sampled combined saw/triangle)

P_T MIDI Note 6 (sampled combined pulse/triangle)

PS_ MIDI Note 7 (sampled combined pulse/saw)

PST MIDI Note 8 (sampled combined pulse/saw/triangle)

RM_T_T MIDI Note 9 (sampled ring mod between two triangle waveforms - unison)

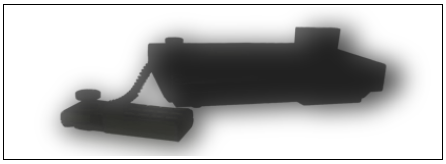
RM_TS_TS MIDI Note 10 (sampled ring mod between two triangle/saw waveforms - unison)

RM_TS_TS_MAJ MIDI Note 11 (sampled ring mod between two triangle/saw waveforms – major third)

NOTE1: Since the combined waveforms are sampled, some choices had to be made on the exact settings of the pulse width used, and frequency ratio used in the ring modulated cases. We chose the most musically useful configurations. Future versions might include a full emulation allowing a bigger variety of sounds. Also important is that the SID generates highly aliased spectra in the highest note registers. This is “normal” for the SID and is an artifact of the digital way in which it generates waveforms.

NOTE2: The chipsounds SID filter currently only emulates a severely distorting/non-linear specimen.
More filter specimens are planned in future updates.

SN76489(AN) (various clocks)



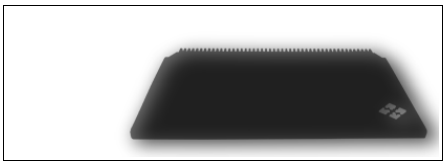
The “SN” is a relatively simple and efficient sound chip providing 3 channels of pure tones, and 1 channel of noise, short and small. No doubt due to its price and simplicity, it has been used in countless systems and arcades and represent the “standard” 8bit audio chip. Its frequency range is acceptable, although it's been often driven by high frequency clocks like 3.5Mhz in the CV console which limited its bass response. (the lower the clock, the better the bass response).

While the noise channel doesn't have great frequency precision, a special mode is accessible on the chip which consists in sacrificing one of the square channels and use its frequency register to drive the noise frequency.
This design has been copied, and licensed to be included in other system including the SMS and the 16bit system that followed it.

chipsounds maps the SN sound sources using the following switches:

Square	MIDI Note 0(C0)	
Short Noise	MIDI Note 2(D0)	(really just a 1/15 PWM tone)
Long Noise	MIDI Note 4(E0)	
Short Noise (ch3)	MIDI Note 5(F0)	(third pulse channel sacrificed to serve as noise freq source)
Long Noise (ch3)	MIDI Note 6(F#0)	(third pulse channel sacrificed to serve as noise freq source)

TED (8360 - various clocks)



Well, this one was a disappointment to everyone. Commodore was selling truckloads of C64s, and decided to release crippled machines to follow its steps: the plus/4 and the C16. The sound was generated by the MOS 7360/8360 “Text Editing Device” chip, a VDC that did both the audio and the video on that system. This IC can only generate two tones at once, the second can be used as a very basic short noise generator (255 bits long). SID this is not.

Working TED chips are very hard to come by, and we've luckily recently acquired a plus4.
It's included in chipsounds for completeness' sake, and to allow comparison with the VIC and SID.

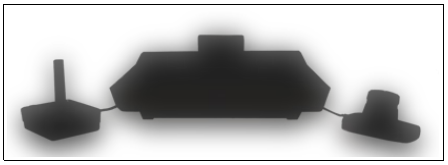
chipsounds maps the TED sound sources using the following switches:

Square MIDI Note 0(C0)

Noise MIDI Note 2(D0) (aliased on Note 1 as well)

More info:
http://ploguechipsounds.blogspot.com/2009/11/ted-100-noise-pattern.html

TIA (NTSC), TIA (PAL) and TIA (unlimited and full ranges)



The TIA (Television Interface Adapter) is a very curious integrated circuit that provides both the audio and the graphics to the 2600 console. This chip is responsible for many instantly recognizable tones like the engine sounds of many games of the era, and laser blasts of many space shooters.

Not unlike the POKEY, this chip generates its tones by combining various “polynomial counters” in various ways called “Distortions”. However, two major differences exists. Firstly, it can only play 32 different pitches per distortion (5bit pitch resolution), and secondly, all pitches have the same spectra.

Before rushing to use the unlimited pitch variant, please consider giving the NTSC/PAL variants a spin, especially if musical tones is not your intended goal. You will find that those 32 steps, once modulated, create all sorts of amazing sounds. Programmers of the time abused this a lot.

chipsounds configures the internal polynomials using these KS:

Dist 1 (SAW) MIDI Note 0
Dist 2 (Idle Tank) MIDI Note 1
Dist 3 (Engine) MIDI Note 2
Dist 4/5 (Square) MIDI Note 3
Dist 6/10 (Near Square) MIDI Note 4
Dist 7/9 (Pitfall) MIDI Note 5
Dist 8 (Noise) MIDI Note 6
Dist 12/13 (Square) MIDI Note 7
Dist 14 (Near Square) MIDI Note 8
Dist 15 (Buzz) MIDI Note 9

NOTE While the TIA program is *purely synthesized*, two sample-based programs are available as “extras” for the TIA in the “03. Multisamples” category

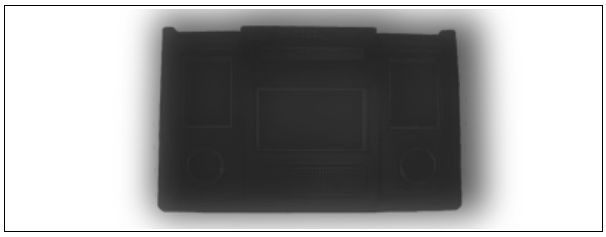
TIA Loops: Each distortion's 32 tones are played in succession to emulate a very common programming trick of the 2600 era. Two octaves are mapped, for the two lopping directions. This could be in theory programmed solely using the wave sequencer, but it since the TIA pitches are not equal temperament, it would make for one ugly sequence.

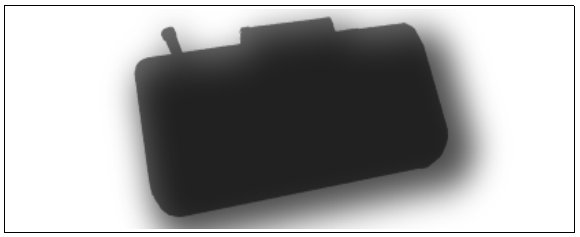
TIA Deaths: before blowing air into the cartridge connector became worldwide fad, there was a time when reinserting a 2600 cartridge many times in different angles was the only way to have a working game. Sometimes the screen went dark, sometimes it filled with all colors of the rainbow (or looked like a grunge plaid shirt), but sometimes, something magical happened, and you could definitely hear it. There is NO trick here, all the sounds in this multisample patch were recorded from the console in around 4 hours of inserting various carts from our collection, and recording everything.

If you let certain samples play a while they may even flat-line!

More info
http://ploguechipsounds.blogspot.com/2009/05/total-stereo-separation-atari-2600-jr.html

UVI 2637(NTSC) and 2637(PAL)





Another chip that was used for both audio and video was the UVI. It was part of a licensed architecture sold by Signetics and implemented in many different consoles which share many titles, but are physically incompatible with each other.

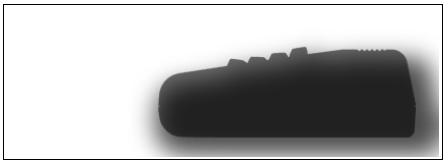
Interton, Arcadia, Leisure Vision, MPT-03, it seems each region of the world had its variants of that system. While not as pitch limited as the P824X, it is similar in terms of waveform generation.

Only one channel that can either play a tone, a noise, or a mixed (bit-wise AND) version of the two, which produces a very unique sound.

chipsounds maps the UVI sound sources using the following switches:

Square	MIDI Note 0
9Bit Noise	MIDI Note 2
Combined	MIDI Note 4

VIC-I : 6560 (NTSC) 6561(PAL) in various configuration.



The VIC-I was an attempt by MOS to create an all in one “video game/video display” chip for the OEM market. But it only ever made it into the VIC-20 and, possibly an obscure Japanese arcade game. (from MAME lore)

Each of the three voices (named Bass, Alto and Soprano) has a limited pitch range of 7bit which makes playing tuned scales quite a challenge. Bass Alto and Soprano each had their own pitch ranges and only partially overlapped each other in terms of discreet pitch values. Safe to say that some scales sound better than others.

The chip also provides a separate Noise channel which generates a waveform that is unlike any other chips in this product.

The original programmers of the time knew that once in a while some square waveforms they played didn't sound “right” under certain condition, but disregarded this as a Fluke. In 2003 a brilliant demo writer by the name of Viznut reversed engineered this side effect, mapping all possible “weird” waveforms that the chip was able to reproduce in a deterministic manner, and put the to good use in his now famous “Robotic Liberation” demo. (they are referred later with the reset patterns required to reproduce them.

chipsounds configures the possible waveforms in the KS:

Square	MIDI Note 0
10	MIDI Note 1 (Viznut waveform)
100	MIDI Note 2 “
110	MIDI Note 3
1000	MIDI Note 4
1010	MIDI Note 5
1011	MIDI Note 6
1110	MIDI Note 7
10010	MIDI Note 8
10100	MIDI Note 9
10110	MIDI Note 10
11000	MIDI Note 11
11010	MIDI Note 12
100100	MIDI Note 13
101010	MIDI Note 14
101100	MIDI Note 15
Noise	MIDI Note 16

NOTE: The noise KS, appears in all Bass, Alto and Soprano sub patches even if it's a distinct channel, this is to be able to do wave sequences containing both tones and noises.

02. Custom Waveforms

This category contains programs which will generate various waveforms that are not tied to any particular chip.
All of these are really just fun exercises in synthesis and are subject to change in revisions of chipsounds

Two Taps LFSR's is an exercise in trying many different configurations of Linear Feedback Shift Registers in order to generate other tones than “white noise”. Some configuration create very nice metallic and gritty sounds

Hard Sync: Using the “Sync Buzzer” logic to make hard sync, and other sounds.

Misc Multipulse: Multipulses without a purpose.

Music Formula's: Live calculations of some of the formulas from this video:
http://www.youtube.com/watch?v=GtQdIYUtAHg

03. Mutisamples:

2A0X Loops, **TIA Loops** and **TIA Deaths** are described in the appropriate parent chip sections

Melstrings(4-bit and DMC) Mellotron Samples by Bernie Kornowicz at Leisureland, used by permission.
Rendered and mapped as RAW 4 bit for extra crunch.

ST-01: Origninal Soundtracker's first bonus sample disk, whose copyrights are in a unknown state (public domain)
The source sounds do come from various commercial synthesizer of the day. Please see the ongoing project to document each sound http://eab.abime.net/showthread.php?p=435531.
When playing ST-01 sounds or importing amiga format files (see later) chisounds models the dynamic sample playback rates of the Amiga Paula chip and its volume limitations. The Amiga LED Filters will be included in chipcrusher.

04. Drums:

Various drum kits running either 4Bit “digi” players (DMG/C64) or 2a03 (DMC) players.
The selection of drums in this category is subject to change in revisions of chipsounds.

05. Arcade:

Various recordings made from our private collection of arcade PCBs. Most of these sounds are analog are not generated from dedicated 'chips' but by a series of electronic components in complex networks. As such, they are unique, and very complex to model (the latest versions of the MAME emulator includes most of these as actual discreet circuit emulation, but does require heavy real time computations. Since chipsounds is 'just a synth' and these sounds are 'just for fun' we have added these as samples (round robins of multiple takes):

- DK: Jump, walk and stomp sounds
- DKJR: Analog effects
- Galax: Analog effects and weird digital oscillator (the later is rendered using chipsounds oscs)
- Phoenix: Analog and 555 timer driven effects
- S.Invaders: Analog Effects and ... bass line!
- Zxxn: Analog Effects

IMPORTANT COPYRIGHT NOTES! While Plogue grants you rights to these actual 'sound recordings',
The sources themselves, **depending on your local laws** may **WELL STILL** be protected by their copyright holders!
See Sound Trademark. Use these sounds wisely!

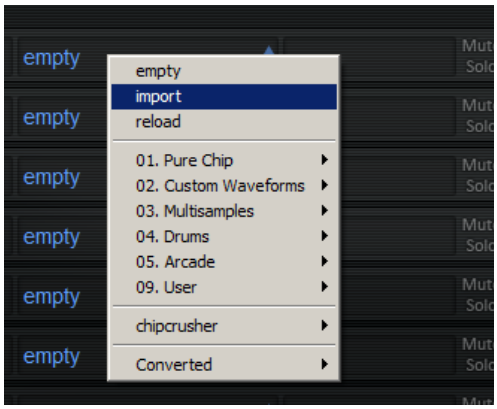
Note: DK and DKJR boards also generate other sounds through the means of sample playback.

These sounds are obviously copyrighted and can't be legally be redistributed.

Importing Sounds

New in chipsounds 1.6 is the ability to import the samples from your own MOD, MED and IFF(8SVX) files (or from files in the public domain, or that you otherwise have the right to use). The files must have a known extension for it to be recognized. Since Amiga file extensions were sometimes before the file name, both myTrack.mod and mod.myTrack are perfectly accepted (its not case sensitive)

Just click on a slot's file selection socket and use the 'import' menu, or (under some platforms/plugin APIs) just drag a file on the interface:



First time you will be asked for a destination repository for the conversion/extraction. We suggest using a folder/drive where you typically store your samples on. The Aria Engine will then create a “ARIAConverted” sub folder in the specified location and will store the results of any conversions in there.

Once a file is successfully converted, not only will it automatically load, but it will also be available in the ***Converted*** menu next time you start chipsounds (Or any other ARIA-based product, like the ARIA Player) as long as the ARIAConverted folder is kept intact. This folder is scanned upon start and the ***Converted*** menu is dynamically reconstructed.

MOD, MED and IFF(8SVX) files are the main supported files but you can also try out DBM, DW, and OKT files. **(*)**
In any case both a “file(Paula)” and “file(Interp)” will be created. The Paula version, as its name suggests, uses chipsounds generators to play the samples as they should sound on an Amiga OCS machine, while the (interp) is really a bare SFZ 1.0 mapping with wav files, in this case it will use the ARIA Engine's native interpolations.

Each converted file will contain a maximum of 36 samples (depending on the original format), to leave room on the keyboard for the key switches to activate each of them. (you can also use MIDI Program Change messages instead)

Tuning woes: Mod trackers usually didn't enforce a match between a sample and its root key. In the day, when you imported a sample, trackers assumed that its sample rate was ***8363Hz*** and that its root key was ***C4***, *regardless* of its real internal pitch. For instance a mod track could have been started with a sample containing an F, even though the tracker would assume it was a C. When importing any mod/med/iff file, be aware that you might have to tweak the tuning of each slot manually (using 'Trans' and 'Tune') to match the tuning of other slots.

The IFF(8SVX) file format was the first (to my knowledge) to optionally include a multisample mapping: one sub sample per octave, with each octave exactly half the length (in samples) of the previous one.
You can try this by downloading and decompressing http://aminet.net/package/mods/smpl/Instruments and adding a .iff extension to each file.

(*)While other formats can be converted (IT, STK, S3M), these extensions were not registered in the registry/preferences upon installation because they are not 'AMIGA native', and potentially contain 16PCM files. However, If you know your way there, you can add them, but we offer no support for this.

Future chipsounds releases will have even more formats to import.

chipsounds Sound Sources explained

Q) So is it Samples or Synthesis?

A) It's whatever made more sense to be true(er) to the original on a case by case basis.
As the application reaches 2.0 and up, I expect chipsounds to be nearly 100% synthesis. (however, the samples would still be available if needed for compatibility/curiosity)

Roughly 95% of it all - everything that is made of discreet steps - is generated using oversampled bandlimited impulse trains. All Squares, SID PWM, 2A03's triangle, YM2149 Buzzer, TIA/POKEY/VL1 multipulses, etc.

Those generators are configured to emulate the pitch limitations of each chips following EXACT maths
that can be found in the chip's datasheets (if available), development manuals, and were verified and sometimes even discredited/revised with comparison to the real chips in our possession.

The rest falls into two categories:

Purely sampled:

1. SID's combined and ring modulated waveforms are sampled across the whole range (2 notes by octave) @ 16bit/96kHz. These waveforms are unique, detailed and variable across the frequency range and they are different across 6581 and 8580 revisions of the SID chip.

2. TIA's Pitch sweeps Loops (16bit/96kHz). NOTE This could be done all in synthesis using the wave sequencer. They are just so fun to trigger and loop.

3. TIA's 'Bugs/Death' sounds are (16/44.1kHz) samples of me spending nearly a day recording and classifying the sounds of badly inserted cartridges into my “total stereo separated” 2600 Jr. ... some of them are pretty awesome I must say. Most looped sounds are due to random memory writes and cycling by fast rebooting of the chip (I assume). Safe to say, i don't think THAT can be emulated ever!

4. NES's NOISE registers loops (16bit/96kHz): This would be more difficult to emulate with synthesis, and would require some pretty otherwise useless code (due to internal frequency tables on the chip). So here they are in samples.

Notes on 2/3/4) Since these are variable pitched loops, you shouldn't use any pitch modulation, at ALL when playing these.

Generators that are half way between samples and Synths:

1) 4Bit Digis (old technique to generate samples by rapidly changing 4bit volume register of certain chips using an fast interrupt routine. - or by using the wave channel in the GB) are represented on disk as 8 bit unsigned RAW PCM, mostly originally recorded at around 32Khz.
When read into the 4bit oscillator, they get accurately resampled and reconstructed as bandlimited/oversampled square waves .. this is NOT a simple sample playback+aliasing bit reduction algorithm trick.!! This generates the same bandlimited pulses as all the other oscillators, but the source of the amplitudes is using a pre-calculated table, in that case a 8bit RAW file.

(NOTE currently included c64 DIGI *rips* – eg: funky drummer - are 4bit PCM padded into 8PCM with zeros, So they don't actually contain more information than the originally did)

2)6Bit DPCM (NES DMC Channel) are either read from disk as 8Bit RAW PCM and *reinterpreted*, or read from “ripped” DMC files. Due to copyright reasons, no DMC Rips have been included into chipsounds. Also even if you DID own the ROM and backup device you would STILL not be allowed to use those samples in a commercial recording without written consent from the original author. You have been warned.

Example DMC usage in an SFZ file:

(...)
sample_const_param03=54
<region> key=36 sample=*com.Plogue.DAC.6Bit.Linear.DPCM|drums/game_x_bd.dmc
<region> key=38 sample=*com.Plogue.DAC.6Bit.Linear.DPCM|drums/game_x_sn.dmc
(...)

Currently there is NO MEANS to inject your own samples using the UI (DMC/4Bit).
You can however hack the SFZ files/ bank to add your own, once they are converted to 8Bit Unsigned/32Khz (lower if you like more grunge)

This is an advanced topic however. Which i wont get into here. Same as any advanced SFZ file editing
If you want to get your hands dirty, either modify Joe or Jules kits (our samples) and add your own, or add entries in the bank yourself.
If you notice, the same SFZ file/Samples are used for both DMC/4Bit variants of those kits. -> flexibility.

You can find more about SFZ on Plogue's forums, including how to make your own sample libraries and mix them though chipsounds.

3) Not all noise patterns use a Linear Feedback Shift Register. Some of them use pre-calculated *tables* (bin files, or ASCII binary strings in SFZ files). Both representation would ultimately sound the same, so it's a bit of a non debate.

IMPORTANT NOTE ON PRESETS:

.aria files : full content of the UI is saved, including values of CC's at the time of saving.
(VST hosts fxb/fxp files are similar in that they contain the same data)

You can currently only save .aria files from the STANDALONE application.
If you use a VST host that support them, you can however save a FXB file.

You can load aria files in the standalone using the file/open menu, or by dragging them on the UI
(including fxb/fxp files) of any incarnation of chipsounds (standalone/VST/AU/RTAS)

.ariax files are just plain text XML versions of the contents of .aria files (easier to manually hack). You can load and save those through the “snapshots” menu.

.ariap files : presets to sub portions of the UI, - presets for the MIDI Arpeggiator, the Wave Sequencer, and the 4Bit waveform display of the DMG-CPU.

Notes on SFZ/Bank (For the VERY technically curious.)

Nearly Everything in chipsounds is plain text and can be edited by advanced users:
With the current SFZ “chip definitions” you could derive them to add chips that don't exist
or that are not currently covered by my version.

The SFZ files describe the inner workings of chips, while the chipsounds.bank.xml bank is a “master catalog” which lists and defines some variables that will influence each chip's sound and limitations, depending on the revision, or its regional setting (PAL, NTSC) etc.

EG: There is only one VIC-I.sfz file, but 8 entries in the bank in order to reproduce the various pitch settings.

Getting Help

The first place to look for a solution to any problem you may be experiencing is in this manual. Please read the manual before contacting support. Next, check the readme files (if any) which contain important information and all last-minute changes that haven't been available when creating this guide.

The Plogue chipsounds Player and sounds are dynamic — evolving and growing. Please check the support area of our website at www.Plogue.com for the latest up-to-date information products, troubleshooting, FAQs, helpful hints and tutorials. Another resource is the support forums.

Whenever you encounter problems, you should also check if you have installed the latest updates. The version number of your software is displayed in the **About** dialog. Updates are released regularly to fix known problems and to improve the software.

If you can't find a solution to your problem please email us at chipsounds.support@plogue.com. The best way to get the help you need is by giving us plenty of detailed information about the problem you are having. We do ask you to read this guide thoroughly and exhaust the other avenues of support before contacting us.

The Plogue forum can be accessed at: http://www.plogue.com/phpBB3/

You don't have to register to browse posts, but before you can post, you will have to sign up.

Acknowledgments

Producing Plogue chipsounds would not have been possible without the combined help, talent and support of many extraordinary people. I am grateful to those who have contributed and would like to thank them all.

François Léveillé (eslapion) for his immense help and in depth knowledge of electronics, analog filters and classic Commodore computers (Merci pour ta patience aussi!)

James Mireau: to believe in my project and making me a beautiful interface and logo for chipsounds.

Bryan Lee: for the awesome presets.

Magnus Jonsson: for your incredible Ambience reverb which makes all the chips shine.

Peter Kirn: for believing in this product on day one of our NAMM 2009 announcement.

Bram de Jong: for the long time friendship and DSP resource.

Gary Garritan for believing in us to design, develop and strengthen ARIA.

Patrice Roy, for being a programming mentor and a very dear friend.

Yann Bourdeau (first C64 on the block, and first BASIC teacher)

Previous research (no particular order):
Viznut, Lord Nightmare, balrog, kevtris, Paul Slocum, Eckhard Stolberg, the MIDIBox forum, Chris Crawford, Mr Enri, Blargg, MJCulross, Asger Alstrup , Marko Makela, Dag Lem, Antti Lankila, Brad Taylor, Sebastian Tomczak, Karen Collins, the MAME/MESS and VICE teams.

Your research was critical to confirm our own.

Testing and suggestions (no particular order):
nitro2k01, XC3N, 8Bit Weapon, ComputeHer, Chupathingy, 4Mat, sink, Peter Swimm, Bill Hicks, Bill Prokopow, Brady Leo, Chad Beckwith, Leif Bloomquist, Jeff Hurchalla, Markleford Friedman, Mathieu Gratton, Rhett Anderson, Jerome Lebel.

Classic Hardware Collectors and Resellers:
arcadecomponents.com
vintagearcade.net
www.GameRoomRepair.com
svdistributing (ebay) - could you believe I got 9 AY's and 25 SN's for $74 total??
slydc (ebay)

DW File import uses code from: Christian Corti (FLOD library - licensed)
MOD, MED File import uses code from libmodplug (public domain)

Musical Inspiration:
Rob Hubbard, Chris Huelsbeck, Kōji Kondō and many others.

Last but far from least:
The Plogue team for believing in my crazy project.. it took a while, but I knew I would convince you guys with the sounds sooner or later.

Martine for enduring the hoarding of all that “crap” in the basement, and my long technical gibberish rants.
And my son, for all that time I couldn't spend with you in the past few years, I'm terribly sorry.

David Viens, September 2012

Appendix A: Quick MIDI CC Reference Guide for chipsounds

chipsounds doesn't rely solely on “Plugin Parameters” for reasons that are not obvious to someone who has never used a multitimbral plugin before. Since this plugin has 8 different slots, the flat parameter list paradigm just does not make sense. Combine this with the fact that each slot has different parameters – so it requires a full list refresh in your DAW each time you change a slot, possibly breaking all your mappings.

But the most important reason why chipsounds only provide parameters for slot1 is that they are simply **not sample accurate**. Some hosts might cut audio timeslices to insert parameters , but this is not a method that we find efficient, nor is it the standard out there. Why do we care about sample accuracy? It is CRITICAL that perfect event timing gets interpreted in between NOTES received by chipsounds. Otherwise you can forget about automating envelope changes, pattern changes and all sorts of wavetable tricks on your end.

If you still need another reason, it would be that we actually want people to be able to exchange chipsounds tracks even if they are using different DAW's and platforms. By using a simple **MIDI file** and a **.ariax file,** You ***can*** do that.

Plogue chipsounds CC Mapping:
(black are GM2 mapping, blue are chipsounds custom)

These are fixed because:
-Aria / SFZ 2.0 has no concept of CC remapping
-The Wave sequencer (MIDI processor) assumes a fixed map.
-We actually want people to exchange projects across DAW's: using MIDI files and .ariax files.

CC001: Pitch LFO Modulation depth from 0 to 1200 Cents (one full octave)
CC002: PWM (Pulse Width Modulation)
CC003: PWM LFO Freq for (SID ONLY)
CC004: PWM LFO Modulation depth (SID ONLY)
CC005: Portamento Time (0 to 4 seconds)
CC007: Master volume (don't change if you want to be accurate to the CHIP!)
CC010: Pan (at your discretion, none of those early chips are stereo)
CC011: Expression (secondary volume – don't abuse either)
CC012: Private pitch shift for Wave Sequencer (does 2 octaves up and down)
CC013: Amp EG Hold Time
CC014: Pitch EG Hold Time
CC015: Pitch EG Depth (-1200 to +1200 cents)
CC016: Pitch EG Attack Time
CC017: Pitch EG Decay Time
CC018: Pitch EG Sustain Level
CC019: Pitch EG Release Time
CC020: Arpeggiator On/Off toggle
CC021: Wave Sequence On/Off toggle
CC022: MIDI Delay On/Off toggle
CC024: 16' volume (MSM5232 only)
CC025: 8' volume (MSM5232 only)
CC026: 4' volume (MSM5232 only)
CC027: 2' volume (MSM5232 only)
CC028: VPOS (DMG Wave channel only)
CC029: PMode (DMG Wave channel only)
CC070: Amp EG Sustain Level (NOTE ... NOT in GM2??)
CC071: Filter Resonance (SID, M5232, and YM2149 Buzz ratio)
CC072: Amp EG Release Time (different for some chips)
CC073: Amp EG Attack Time(different for some chips)
CC074: Filter Cutoff Frequency (SID , M5232, ONLY)
CC075: Amp EG Decay Time (different for some chips)
CC076: Pitch LFO Rate
CC078: Pitch LFO Delay
CC079: Lowpass Filter On/OFF switch (SID,VIC, VL-1)
CC080: Bandpass Filter On/OFF switch (SID only)
CC081: Highpass Filter On/OFF switch (SID only)
CC082: Filter EG Attack Time (M5232 only for now)
CC083: Filter EG Decay Time (M5232 only for now)
CC084: Filter EG Sustain Level (M5232 only for now)
CC085: Filter EG Release Time (M5232 only for now)
CC086: Pitch LFO Type
CC090: Amplitude LFO Freq
CC091: Reverb Send Level
CC092: Amplitude LFO Depth

Appendix B: Host Specific Issues

Renoise versions 2.1 and prior:
You HAVE to untick "Auto Suspend" in the bottom right of the chipsounds instrument box for the UI to be usable. This WON'T be changed, if you use Renoise get used to that. :)

Sonar versions prior to 8.5:
You may get audio dropouts when the chipsounds UI is shown, please visit our support pages for a registry based fix.