MODEL SOUNDS INC.

R/C Models Like You've Never Heard Them Before!

SFXPC4 and ShockWave 2 Module - Advanced Topics

Revision 09

Date: January, 2018

TABLE OF CONTENTS

TABLE OF FIGURES

ABOUT THIS DOCUMENT

This document contains information of a more advanced nature than the SFXPC4 and **ShockWave 2** manuals and is intended to help technically savvy **ShockWave 2** module owners better understand how it works and how to get the best out of this amazing product. It is not necessary to read this document in order to operate the module or the SFXPC4 software.

If you do not know anything about digital audio and/or have difficulty with computers, it is recommended that you **NOT** read this document.

Throughout this document various terms are hyperlinked to detailed explanations in Wikipedia and other resources.

OVERVIEW OF DIGITAL AUDIO BASICS

All digital recordings of sound start off as raw PCM (Pulse Code Modulation) streams of data. They may be later processed with various compression algorithms into formats such as MP3, WMA etc. The **ShockWave 2** module plays back sound in its raw PCM format so the SFXPC4 software converts the sound files to this format when they are loaded into the application. Although the SFXPC4 application can read and load MP3 files as well as WAV files, they are converted into raw PCM during the loading process, so by the time they are displayed in the SFXPC4 sound clip lists, they are raw PCM.

To convert an audio waveform into a PCM stream, samples of the waveform are taken at regular intervals – the sample rate, and the amplitude (+/-) of the waveform at that instant is taken and stored as a digital number. Two engineers, Harry Nyquist and Claude Shannon, developed a theorem that states "a bandlimited analog signal that has been sampled can be perfectly reconstructed from an infinite sequence of samples if the sampling rate exceeds 2*F* samples per second, where *F* is the highest frequency in the original signal". The range of human hearing extends to about 20KHz (20000Hz) for young people with normally sensitive hearing, so this means that in order to reproduce audio frequencies of 20000Hz, the sampling rate has to exceed 40000 samples/second. ***Note:*** the correct term is 40Ksamples/second, not 40KHz since the unit Hertz refers to cyclic quantities and not discrete samples.

When the CD PCM format was invented in the early 1980's, its sample rate was set to 44100 samples/second, so that theoretically, frequencies up to 22KHz could be reproduced. The resolution of the PCM samples was set to 16 bits and the samples are signed numbers. 2^{16} = 65536, so a 16 bit sample can represent 65536 different values.

Rather than go from 0 to 65536, which is an unsigned number, the CD samples are signed which means values from -327868 to +32767 can be represented. ***Note:*** The upper value is +32767, not +32768, because the value zero takes up one of those available 65536 numbers.

Using a 16 bit sample, sound amplitude values as small as 1/32768 = 0.0000305 of the maximum value can be represented. This smallest value is called the quantization error and represents distortion of the sound. This value translates into 0.003% error – which is very, very small.

R/C sound modules from most of our competitors use 8 bit 22050 samples/second maximum. $2^8 = 256$, so values from 0 to 255 can be represented. 8 bit samples are always unsigned, so this means that the most negative amplitude value is represented as 0 and the most positive value is represented as 255. The waveform zero crossing point is represented as the value 128. So for 8 bit samples, the quantization error is 1/128 = 0.0078 which represents 0.78% error – a very high value.

The maximum sample rate supported by the ShockWave 2 module is 44100 samples/second, so the theoretical audio range of the ShockWave 2 module is 22050Hz with 0.003% error.

SHOCKWAVE 2 FILE FORMAT

As well as the raw audio data for many sound clips, the **ShockWave 2** module has to store a good deal of configuration data, much of which relates to those sounds. This additional data is called metadata. In order to maintain absolute data integrity between the sounds and their relevant metadata, all of the sounds and all of the configuration (meta) data is stored in a single file with our proprietary .sfx7 format. Thus wherever the sounds go, their configuration data goes with them. This guarantees correct operation of the sounds. This does mean, though, that if any of the sounds are added, changed or removed, the complete .sfx7 file must be downloaded to the module.

LOOPING THE SOUNDS

By default, all sounds in the **ShockWave 2** module will loop back to their start when they reach the end of the sound clip. However, the actual loop start point can be set to anywhere within the sound clip. The loop end point can also be set to anywhere in the sound clip as long as it is later than the loop start point.

Here's what happens when a sound clip plays on the ShockWave 2.

1. It starts at the sound's start and plays its entire length.
2. When it has reached the end, it starts playing at the loop start point.
3. It continues until the loop end point (which may be the end of the sound, or earlier).
4. When it reaches the loop end point it restarts at the loop start point.

This continues for as long as the sound is switched on. You can optionally set any "regular" (non-engine) sound to Play Once. In this mode, after (1) above, it stops playing. The engine shutdown sound is automatically also Play Once.

This looping behaviour supports the engine start-up sounds which start off with the engine starting up and then continues into its idling sound. Naturally with the throttle in the idle position you want to hear the idle sound, not the start-up every time. This also supports the playing of a music clip and repeating just the portion of the clip you want.

PREPARING SOUNDS FOR LOOPING

In order to not hear clicks or pops when the sound loops back from the loop end to the loop start, they must be prepared so that the amplitude **AND SLOPE** of the waveform at those points are very close. Any large step change in either might be heard as a click. So before sounds are loaded into the SFXPC4 application, they should be inspected using an audio editor such as the freeware Audacity program included on the **ShockWave 2** microSD card. Audacity is completely free with no restrictions and has excellent features and user interface.

In order to make the job of preparing sounds for looping easier, a few simple rules and tips should be followed.

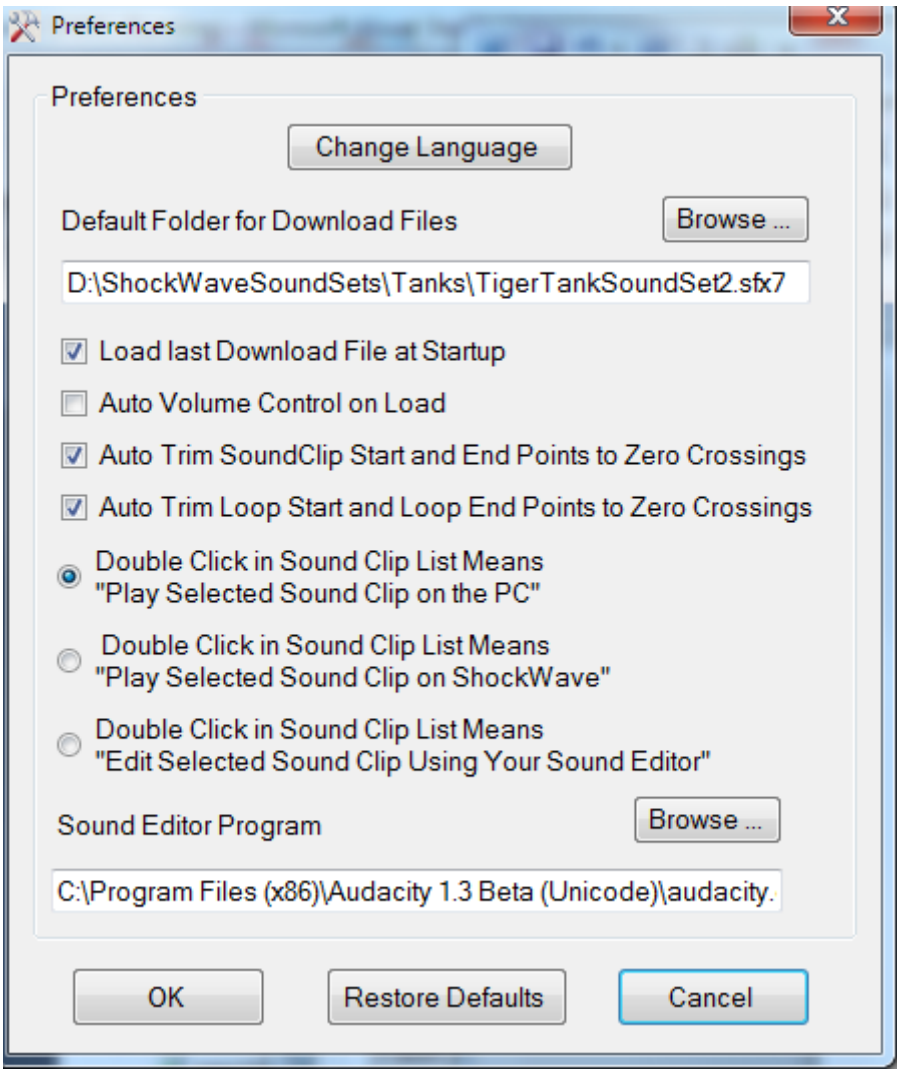
1. Always start the sound clip with a sample that is as close to zero as possible **AND** with a **GENTLE** positive going slope. If you find it is starting with a large amplitude swing, it is better to truncate the sound at its start until a much smaller amplitude swing is occurring.

2. Inspect the last few samples of the sound and make sure that it finishes with a similar valued positive going slope. **To be absolutely precise, the sound should finish, not at a zero sample, but just one sample before it reaches zero.** This is because, when it loops back to the start, there is already a zero sample there and we would get two zeroes in a row.

This cannot be heard when listening to complex waveforms, but if listening is done on pure sine wave tones, as we do in our labs sometimes, such a small error can actually be heard very faintly.

3. To assist in 1 and 2 above, in the SFXPC4 Application Preferences form there is an optional checkbox that allows an automatic "trim" adjustment to be made on both the start and end of the sound clips to achieve, as close as possible, the conditions described in 1,2. The algorithms used to achieve this are proprietary to Model Sounds Inc. and will not be disclosed. The Preferences form is shown next.

Figure 1 - Preferences Form Showing AutoTrim Features



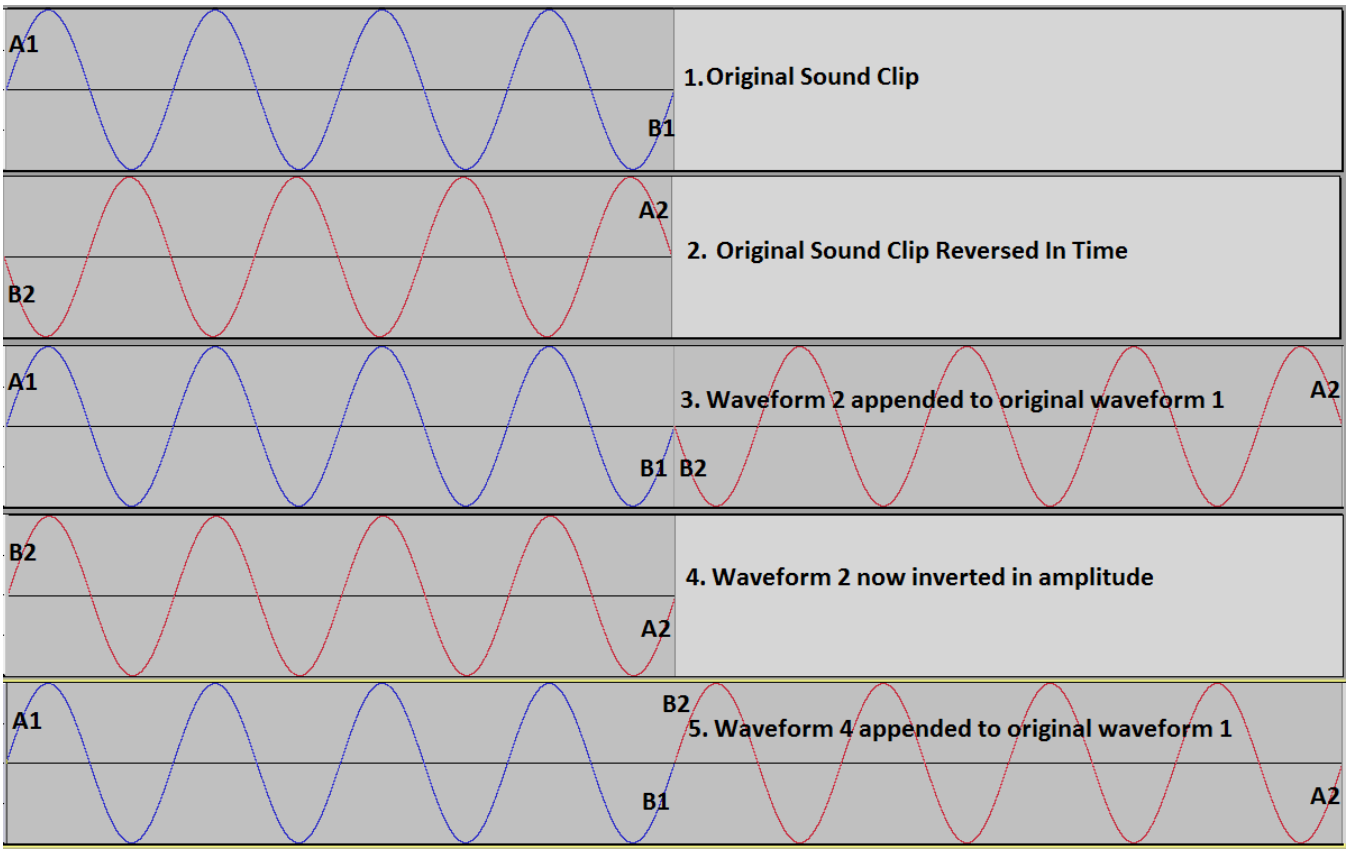
There are two optional Auto Trim features. The first one auto trims the start and end points of the whole sound clip when it is first loaded into SFXPC4 from your hard drive.

The second one auto trims when you set or adjust the Loop Start and Loop End Points.

These features are usually quite accurate and helpful in doing the trim process, but occasionally it does not help, and you may want to switch it off.

A "trick" that can be used with very regular waveforms where the loop start and end points are at the start and end of the sound clip, e.g. engine step sounds, is to append the same sound to its end, only reversed in time and inverted in amplitude. This is done using a waveform editor such as Audacity and is best described using a sine waveform as a sound sample as in the next figure.

Figure 2 - Appending a Reversed and Inverted Copy of Itself to an Original Sound Clip



The procedure to generate a longer sound clip with a seamless join and identical start and end slopes and amplitudes using the Audacity sound editor is as follows.

1. Open the original waveform in Audacity.

2. Check the start and end samples and, if they are not at the zero position having the same slope as described earlier, then trim the waveform so that they are.

3. Using the File->New menu, open a new Audacity window.

4. Copy and paste the whole of the original waveform (1) to the new Audacity window.

5. Select all of this copied waveform and reverse it in time. Use the Effect->Reverse menu item.

6. Keeping the entire copied and reversed sound clip selected, invert it in amplitude. Use the Effect->Invert menu item.

7. In the original waveform window, set the time cursor to the end of the original sound clip.

8. In the Copied, Reversed and Inverted waveform window re-select that entire waveform and copy and paste that into the original window where it should append to the end of the original waveform.

9. Export that new waveform in the original window to a WAV file with a name of your choosing, other than the original so as not to overwrite it.

10. If everything above has been done properly, the end of the new sound clip will be a perfect match in both amplitude and slope to the beginning of the sound clip.

> Note : If the copied and reversed waveform is appended without the additional step of inverting it, the result will be as waveform 3 above with a cusp at the join (B1, B2) and another cusp at the end/start Loop Point (A1, A2). This is because when the waveform is reversed in time, that also reverses the slopes.

The above process works for many sound clips that have a very regular and rhythmic sound, and which are very similar when played backwards (i.e. reversed in time).

PLAYING MULTIPLE SOUND CLIPS SIMULTANEOUSLY

The ShockWave 2 sound module can play two sounds simultaneously if the sample rate is 44100, or three sounds simultaneously if the sample rate is 22050 samples/second.

Whilst there are many small MP3 and other sound players that can play stereo music tracks at a full 44100 samples/second, playing two mono tracks that are completely unrelated at that speed is a much more difficult task.

Stereo sounds are represented in the digital domain by two consecutive 16 bit numbers, the first one representing the left channel and the second representing the right channel. Therefore if we were to playback a stereo sound all we would have to do is access the start of the first sample and stream the audio samples out of the flash memory chip, one after the other, 32 bits for each stereo sample. Thus theoretically, the microcontroller would have to send only the initial read command and three address bytes to the flash memory chip and then just read in the audio samples, one after the other.

However, two completely unrelated sounds (such as an engine sound and a gun sound) start at two completely different addresses on the microSD Card, Therefore when each audio buffer is empty for one sound, the microcontroller has to seek to a new position to find the sound samples for the other sound. Also the microcontroller then has to fill a 32 bit buffer with samples in order for the stereo DAC (Digital to Analogue Converter) to play back both sounds as though they were a stereo sound track.

Despite the fact that the 32 bit microcontroller is clocked at 80MHz, there are only just enough clock cycles in one sample period at 44100 samples/second to get everything done. Besides reading the samples for playback, there is much other processing that goes on "simultaneously" to interpret all the R/C inputs, process switched outputs, manage end of sound and looping conditions etc., etc. This is no mean feat.

At the lower sample rate of 22050 samples/second we have much more time available and there is sufficient time to read the microSD Card and fill three internal buffers and perform all the other processing.

INDEX