# SoundFont Decoding Functional Specification

## Introduction

### Summary

The SoundFont decoding script will be a utility included with the TeensyAudio Wavetable codebase and will allow users to pull data for various instrumentation samples from SoundFont 2 files for use by the TeensyAudio Wavetable library.

### Requirements

See requirements documentation for detailed description of the functional requirements this document will implement.

### Existing System

Currently this functionality isn’t available to users of the Teensy library. The script to decode SF2 files will be a brand new addition to the TeensyAudio codebase.

### Terminology

**DAC** - Digital Analog Converter (the Teensy 3.2 is an example of a DAC).

**SF2** - SoundFont version 2 File Format (wavetable audio data wrapped w/ synthesis methods).

**Sample Array** - An array or list of individual byte samples to be produced by the SoundFont decoding script for use in the TeensyAudio Wavetable library.

**Attack** - Portion of the decoded sample array which plays immediately upon activation (e.g. piano key depression).

**Sustain** - Portion of the decoded sample array which can be looped indefinitely to produce a sustaining, consistent instrument tone.

**Release** - Portion of the decoded sample array which plays after the sustained section of a note, and will mark the natural end of a tone.

### References

Wavetable Synthesis on Microcontroller Requirements.docx - Rev 115

## Functional Description

**Use Cases**

1. **Load Soundfont/SF2**
   1. Description: User provides something for the utility to process.
   2. Actors: User; Soundfont;
   3. Entry Conditions: N/A
   4. Exit Conditions: Relevant contents of the Soundfont are displayed on screen in some selectable format.
   5. Flow:
      1. User invokes *utility*, specifying local file path to desired soundfont/SF2.
      2. Necessary checking for compatibility and proper file structure is done by the utility. Error and exit program if one of the following scenarios are met:
         1. SF2 file can’t be imported, or a file read issue occurs.
         2. Instrument/sample sections don’t exist within the file.
      3. Success message displayed to user if compatible.
         1. Failure message displayed to user if Soundfont does not meet the protocol or in the event of a File Read Error.
         2. Exit.
      4. Continue to Select Instrument.
2. **Select Instrument**
   1. Description: User needs to select an instrument(s) to use.
   2. Actors: User; Soundfont;
   3. Entry: Soundfont contents are displayed on screen.
   4. Exit: User has chosen *quit, or finish.*
   5. Flow:
      1. A prompt is displayed on screen to select an instrument, otherwise quit.
      2. User selects quit.
         1. Utility terminates.
      3. User inputs a selection.
         1. A prompt is displayed on screen to select an additional instrument, finish, or quit.
         2. If User selects an instrument
            1. go to 2.5.3.1
         3. If User selects finish
            1. Continue to Parse Samples.
         4. If User selects quit.
            1. Terminate utility.
3. **Parse Sample(s)**
   1. Description: User selects a sample(s) to export.
   2. Actors: User; Soundfont;
   3. Entry: Available samples are shown to the user for the selected instrument.
   4. Exit: User selects sample(s) to export, then the program continues to export samples.
   5. Flow:
      1. A prompt is displayed on screen to select an sample, otherwise quit.
      2. User selects quit.
         1. Utility terminates.
      3. User inputs a selection.
         1. Error and exit program if one of the following scenarios are met:
            1. Sample loop-end - loop-start > sample-end - sample-start (Check that the looping section is smaller than the sample itself.
            2. Sample loop-end < sample-end (Check that the sample loop end position isn’t past the end of the sample)
         2. If errors have occurred here, then we can potentially do some automated modification to the start and end index locations. Ask the user if they want the program to attempt to fix these issues, or quit.
         3. If no errors, a prompt is displayed on screen to select additional samples (if memory space on Teensy would allow so), finish, or quit.
         4. If User selects another sample
            1. Go to 3.5.3.1
         5. If User selects finish
            1. Continue to export samples.
         6. If User selects quit.
            1. Terminate utility.
4. **Export Sample(s)**
   1. Description: Selected samples get exported/encoded to a .cpp file with corresponding .h file, in either PCM or ulaw sound data encoding.
   2. Actors: User; Soundfont;
   3. Entry: Desired samples have been chosen for export.
   4. Exit: Program generates a corresponding .cpp and .h file for the selected sample contents.
   5. Flow:
      1. Desired sample data gets pulled from the SF2.
      2. User selects desired encoding method for sample data (options are PCM or ulaw)
      3. For each sample, write the following to a .cpp file:
         1. Output a struct declaration with an appropriate name for the sample.
         2. Within the struct, write out metadata about the sample, including sample rate, length, and frequency (if available).
         3. Within the struct, write out 3 separate arrays: AudioSample\_Attack, AudioSample\_Loop, and AudioSample\_Decay. These arrays will contain sample data for all 3 different sections of the sample.
      4. Confirmation is displayed to the user with information pertaining to where the file was saved, and how many samples were exported.
      5. Program exits.



**User Community**  
The demographic of this project is anyone wishing to extract audio data from a SoundFont 2 file. This may include musicians, programmers, audio engineers, students, and casual “tinkerers”. The primary target system for this project is the Teensy 3.2 microcontroller and its related software, particularly for the purpose of wavetable synthesis.

**Error Handling**

***File read error:*** In the event that the program either cannot locate the target file, cannot properly extract audio data from the file, or finds the file to be corrupt or otherwise unusable, it will throw an exception without generating any output audio data.

***Invalid instrument:*** If the implementation of instrument selection (described below) admits the possibility of making an invalid selection, then the input should be ignored and the user should be prompted again.

**Help**  
Help will be accessible through the design/implementation documents and comments in code.

**Interface**  
The user will be able to execute the script from a command line, giving an SF2 file pathname as input. Assuming the file read is successful, the user will be prompted to select from a range of instruments encoded in the file. If the selection is valid, the script will produce (along with a confirmation message) an output .cpp file with 3 unsigned int arrays containing the byte string audio data extracted from the SF2 file for the given instrument. These arrays represent the 3 sections of the SF2 audio data (attack, sustain, release).

**Platforms**

This software will be supported on all OSes which support SF2 and Python 2.7.

**Portability**  
This utility should be usable by any version of Python > 2.7.

**Documentation**  
Developer documentation to be provided to list out how to utilize this utility, as well as the different functions/options that are exposed.