Technical Specification

**Project: TeensyAudio Wavetable Synthesis**

Authors: Ryan Mellmer

Connor Delaplane

Aida Keifer

Nicholas Craig

Josh Bucklin

Jonathan Jensen

Xuan Tang

Date: 2016-11-28

Version: 1.0

**Document Control**

**Author**

|  |
| --- |
| Name |
| TeensyAudio Capstone Team |

**Revision history**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Issue date | Author/editor | Description/Summary of changes |
| v1.0 | 11/28/2016 | TeensyAudio Team | Creating the initial document version |
|  |  |  |  |

**Table of Contents**

[**1** **INTRODUCTION**](#_3znysh7)

[1.1 Objectives](#_2et92p0)

[1.2 Scope](#_tyjcwt)

[**2** **Functionality**](#_3dy6vkm)

[2.1 Site Map](#_4d34og8)

[**3** **Architecture**](#_17dp8vu)

[3.1 Architecture Overview](#_3rdcrjn)

[3.2 Application Interfaces](#_35nkun2)

[**4** **Design**](#_1ksv4uv)

[4.1 Object Diagram](#_44sinio)

[**5** **CODE**](#_4i7ojhp)

[5.1 File Names and Structure](#_2xcytpi)

[5.2 Executable File Names and Structure](#_1ci93xb)

# INTRODUCTION

## Objectives

The purpose of this document is to outline the technical specifications for the Wavetable Synthesis library to be used on the Teensy 3.2 microcontroller. It is primarily meant as a guide for the developers of the library and will cover the following areas:

* Definitions of all objects within the library along with their interfaces.
* Interactions with the existing Audiostream.h library API.

## Scope

The purpose of this project is to provide a C++ library and accompanying Python utility scripts allowing realistic instrumentation audio to be synthesized on the Teensy 3.2 Arduino Digital Analog Converter (DAC). This library will be exposed to developers, and will allow pitch shifting, looping, tremolo, and vibrato effects to be imposed on a raw byte buffer of recorded samples.

# Functionality

## Soundfont Decoding

The basic functionality of the SoundFont Decoding script is defined in the Wavetable Synthesis on Microcontroller Requirements document. A brief summary is provided here.

* User executes the script in a console
* User loads a .sf2 file into the utility by passing it’s location to the script
* If the .sf2 adheres to pre-defined limitations for what the Wavetable Library accepts, then allow the user to select which instrument to import
* Parse samples into logical units
* Output the resulting samples into a C++ struct/object within a .cpp file for use within the Wavetable Synthesis Library

## Wavetable Synthesis Library

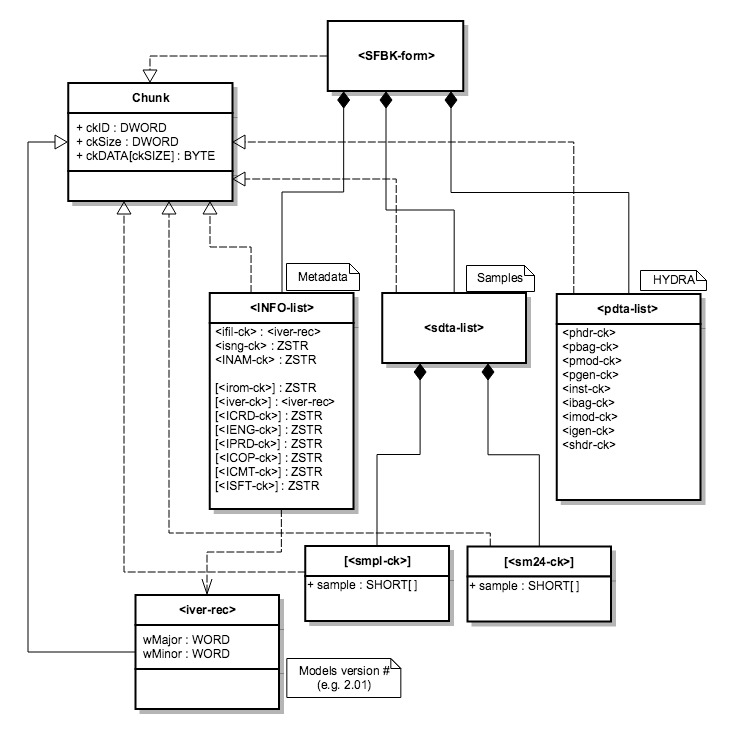
The basic functionality of the Wavetable Synthesis Library is defined in the Wavetable Synthesis on Microcontroller Requirements document. A brief summary is provided here.

* Load audio data from the file produced by SF2 extraction process
* Play silence in the event of read error
* Create interpolated audio data on the Teensy
* An audio sample’s sustain data can be looped
* Audio data can be modified with a tremolo effect
* Audio data can be modified with a vibrato effect

# Architecture

## Architecture Overview

**SF2 file object diagram:**

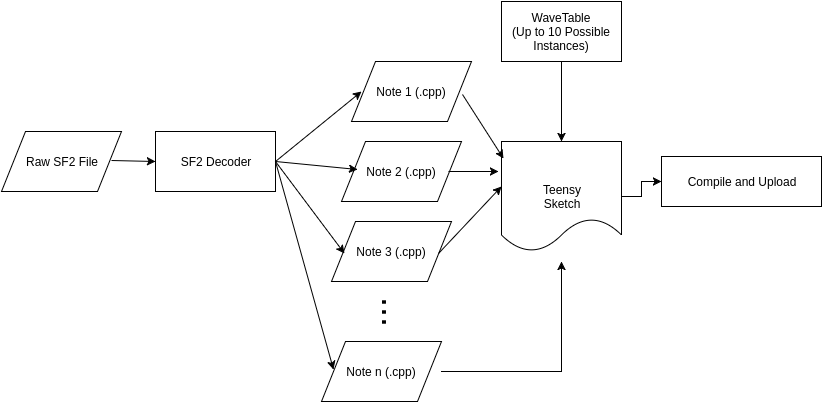


**SoundFont Decoding:**



**Teensy C++ Library:**

This diagram gives a general overview of how both the SF2 Decoder and the Teensy C++ Library will work together. It’s important to notice that a single Note # file generated by the SF2 Decoder will contain all three phases for a single sample (attack, sustain, release). In addition, while there is only a single arrow from the Wavetable to Teensy Sketch there could be up to 10 instances. There should be at least one Wavetable object for every note included in the Teensy Sketch file.



## Application Interfaces

This section contains a description of all SF2 Decoding and Teensy C++ Library application interfaces.

**Python SF2 Decoding:**

The SoundFont decoding script will be ran only through the command-line, so the only interface to be described is the command-line. The user will need to input the location of the SoundFont file to decode, and the script will walk the user through the decoding process.

**Teensy C++ Library:**

**Name:** Wavetable(SF\_Extraction\_Data)

**Description:** The constructor for the Wavetable object. It’s primary responsibility is to initialize all the attack, sustain, and release pointers to point to their respective sample arrays within the provided SF\_Extracation\_Object struct.

**Input:**

SF\_Extraction\_Object - A c++ struct/class with 3 const int arrays that contain attack, sustain and release arrays for a single note.

**Name:** update()

**Description:** This method is required when inheriting from the Audiostream library. It is responsible for managing all of the Wavetable object’s audio processing and is responsible for allocating, transmitting and releasing all audio blocks associated with the object.

**Input**: N/A

**Name:** soundOn()

**Description:** This method will set the playing field to 1 (true) and transmit the attack samples, followed by continuous transmission of the sustain samples until the soundOff() method is called. This uses the stored frequency and intensity values in the Wavetable object.

**Input:** N/A

**Name:** soundOn(float frequency, uint intensity)

**Description:** This method works the same as the no parameter version of soundOn() except that it first sets both the frequency and intensity fields in the Wavetable object.

**Input:**

frequency: a float value that represent the number of samples/sec that the Wavetable will play.

intensity: a uint value that represent the intensity multiplier for the transmitted sample data.

**Name:** soundOff()

**Description:** This method first sets the playing field to 0 (false). Then it completes transmission of the current iteration of sustain samples so that it doesn’t cut off abruptly and lastly transmits the audio samples for the release phase.

**Input:** N/A

**Name:** setFrequency(float frequency)

**Description:** This method sets the frequency field in the Wavetable object.

**Input:**

frequency: a float value that represent the number of samples/sec that the Wavetable will play.

**Name:** setIntensity(uint intensity)

**Description:** This method sets the intensity field in the Wavetable object.

**Input:**

intensity: a uint value that represent the intensity multiplier for the transmitted sample data.

# Design

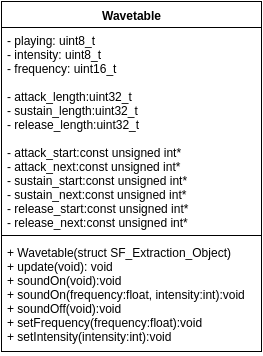
## Class Diagrams

**Soundfont Decoding:**

The SoundFont decoding script will be contained within a single Python file. Since the script will be a purely command-line driven tool, an open source library, sf2utils, will be utilized to assist in the soundfont decoding process.

**Teensy C++ Library:**

The Wavetable Synthesis library consists of one primary class called Wavetable. A class diagram that lists the fields and methods of this class is given below. This is followed by a brief description of each element within the diagram. Note: The public methods and their functionality are listed in section 3.2, Application Interfaces.



**Wavetable Field Descriptions:**

playing: Keeps track of whether or not the wavetable is currently playing samples. Will be either 0 (false) or 1 (true).

frequency: The number of samples/sec that the wavetable object will be able to transmit.

intensity: Keeps track of the intensity of the transmitted audio data.

attack\_length: The number of attack samples in the attack array produced by the SF2 Decoder.

sustain\_length: The number of sustain samples in the sustain array produced by the SF2 Decoder.

release\_length: The number of release samples in the sustain array produced by the SF2 Decoder.

attack\_start: The starting address of the attack sample array. This won’t move during wavetable operation.

attack\_next: The next location within the attack sample array that will be read from.

sustain\_start: The starting address of the sustain sample array. This won’t move during wavetable operation.

sustain\_next: The next location within the sustain sample array that will be read from.

release\_start: The starting address of the release sample array. This won’t move during wavetable operation.

release\_next: The next location within the release sample array that will be read from.

# CODE

## File Names and Structure

This section lists the name, purpose and structure of all expected files for this project.

**SoundFont Decoding Utility:**

**File #1:** SF2Decode.py

**Purpose:** Main Python script file which the user will actually interact with to decode samples from a soundfont file.

**Structure:** This file will contain the entirety of the necessary functionality for the SF2 decoding script.

**Teensy C++ Library:**

**File #1:** Wavetable.h

**Purpose:** This file contains the Wavetable class prototype and is intended to be #include(d) within any file using the Wavetable object.

**Structure:** This file will contain a single class prototype, Wavetable, that is derived from the Audiostream class. As such the prototype will contain the update() method along with all other fields and methods listed in the Wavetable class diagram found in section 4.1.

In addition to the class prototype the Audiostream.h library will be #include(d) so it is available for the new Wavetable class.

**File #2:** Wavetable.cpp

**Purpose:** This file contains the implementation of all of the Wavetable class methods.

**Structure:** This file will #include the Wavetable.h file and will contain implementations for all public and private methods of the class.