Sound Driver

This file documents the sound driver used in Hex Sweeper.

1 Overview

The purpose of this driver is to be small, while still making use of the YM2612 FM synthesis. Consequently, the driver is not very flexible. For example, the driver always operates on four channels of audio, with the panning of each channel set to center.

2 Formats

The data formats of the driver are meant to correlate closely with the YM2612 hardware. This allows more time to be spent playing the song instead of waiting to process it.

2.1 FM Voices

Voice data is stored as an array of 0x19 bytes, as follows:

Offset	**Meaning**	**“Grand Piano”**
0x00	Feedback/Algorithm	0x32
0x01	Detune / Multiplier	0x71 0x0d 0x33 0x01
0x05	Total Level	0x23 0x2d 0x26 0x00
0x09	Rate Scaling / Attack Rate	0x5f 0x99 0x5f 0x94
0x0d	AM level / Decay 1 Rate	0x05 0x05 0x05 0x07
0x11	Decay 2 Rate	0x02 0x02 0x02 0x02
0x15	Sustain Level / Release Rate	0x11 0x11 0x11 0xa6

2.2 Track Header

Following in the philosophy of simplicity, a song does not specify which instruments it should use. In fact, the only metadata the song stores is a set of pointers to track data. Each song contains four tracks, and the header consists of four 16-bit words in big-endian format describing the offset of each track from the beginning of the header.

2.3 Track Command Language

In this section we describe the track command language.

2.3.1 Playing Notes

```
  7    6 5 4   3 2 1 0
+-----------------------+
| 0   octave   position |  0x00-0x7F
+-----------------------+
```

Upon reaching a note command, the driver will perform a key-on for the specified note, delay (see Section 2.3.3 [Delay], page 2), then perform a key-off.

An octave may be divided into 12 notes spaced one semitone apart. Beginning with C, these would be listed as either

```
C C# D D# E F F# G G# A A# B, or
C Db D Eb E F Gb G Ab A Bb B.
```

The position is the zero-based index into this list. So C is 0, while G is 7. The octave is as in scientific-notation. Only those octaves between 0 and 7 inclusive are represented. Thus, middle C (C-4) is listed as 0x40. For any note, the octave number is the first digit in the hexadecimal notation.

Attempting to play a note with a position beyond B (that is, any from 0xc through 0xf inclusive) is undefined behavior.

2.3.2 Playing a Rest

```
  7 6 5 4 3 2 1 0
+-----------------+
| 1 0 0 0 0 0 0 0 |  0x80
+-----------------+
```

Upon reaching a rest command, the driver delays (see Section 2.3.3 [Delay], page 2), then continues.

2.3.3 Setting Delay Values

```
  7 6 5 4 3 2 1 0   7 6 5 4 3 2 1 0
+-----------------+-----------------+
| 1 1 0 1 0 0 0 0 | delay length    |  0xD0__
+-----------------+-----------------+
```

Upon reaching a delay command, the driver sets the delay length. For NTSC systems, an input length of n results in notes that are roughly

$$0.3125n \text{ seconds}$$

in length. However, an input length of zero does not make much sense, so as a special case this is treated as 256 instead.

A delay of 0x10 produces a quarter-note at 120 bpm.

2.3.4 Branch Setup

```
  7 6 5 4   3 2 1 0
+-------------------+
| 1 1 1 1   count   |  0xF_
+-------------------+
```

Upon reaching a branch setup command, the driver loads its current playback address into the loopback register and sets a counter to ‘count’. As a special case, a count of zero is treated as 16.

Branch setups may not be nested. A branch setup command will overwrite the data from any earlier ones.

2.3.5 Unconditional Branches

```
  7 6 5 4 3 2 1 0   7 6 5 4 3 2 1 0
+-----------------+-----------------+
| 1 1 1 0 0 0 0 1 |   displacement  |  0xE1__
+-----------------+-----------------+
```

An unconditional branch command causes the current playback location to be advanced then moved by the given eight-bit biased displacement. This is quite different from the conditional branch command (see Section 2.3.6 [Conditional Branches], page 3).

The bias is such that 0x00 corresponds to -253 bytes, and 0xff is only +2. This is the largest possible backward branch possible while still allowing a branch statement to be passed over, thereby allowing a song to use chained back-branches.

The halt command (see Section 2.3.8 [Halt], page 3) may seem to be equivalent to a command that is an unconditional branch to itself (0xe1 0xfb), encoded in a single byte. However, using this two-byte command will cause the driver to permanently stop playback until the CPU is reset, whereas the halt command allows other channels to continue playing.

A loop containing neither notes nor rests will prevent the driver from accepting commands, thereby requiring a complete reset of the CPU to continue. Please do not include such loops in songs.

2.3.6 Conditional Branches

```
  7 6 5 4 3 2 1 0
+-----------------+
| 1 1 1 0 0 0 0 0 |  0xE0
+-----------------+
```

When the driver meets a conditional branch command, it decreases the loop count register by one. If the result is equal to zero, playback continues as normal. Otherwise, the playback location is set to the contents of the loopback register.

2.3.7 Inhibiting Key-Off

```
  7 6 5 4 3 2 1 0
+-----------------+
| 1 1 1 0 1 1 1 1 |  0xEF
+-----------------+
```

Normally, a note is keyed off after its delay (see Section 2.3.1 [Notes], page 1). This can be toggled by the sustain command. Notes played after this command will not be keyed off until the setting is toggled again.

2.3.8 Terminating Playback

```
  7 6 5 4 3 2 1 0
+-----------------+
| 1 1 1 0 0 0 1 0 |  0xE2
+-----------------+
```

Upon reaching a halt command, a channel ceases processing. No new notes will be played until the driver is reset.