NESIZER: Period calculations

This document describes how period calculations are done effectively in the NESIZER code.

Note representation

A *note* is represented by a 16-bit value of the following struct (it is wrapped in a union to allow writing a raw value directly):

```
union tone {
  uint16_t raw_value;
  struct {
    uint8_t offset : 6;
    uint16_t semitone : 10;
  } __attribute__((packed));
};
```

The upper 10 bits are used to represent the semitone, while the lower 6 bits represent the *offset* from this semitone towards the next. The resulting frequency is given by

$$f = f_s \cdot 2^{\frac{1}{12} \cdot \frac{o}{64}}, \tag{1}$$

where f_s is the semitone's frequency and o is the offset.

Calculating timer values

The formula used for calculating timer values from this note representation is based on the following relationship between timer values and resulting frequencies in the pitched APU channels:

$$f = \frac{f_{cpu}}{16\alpha(T+1)},$$

where $\alpha = 1$ for square channels and $\alpha = 2$ for the triangle channel. This relationship comes directly from the fact that each timer countdown advances one step of the generated waveform, and square waves take 16 steps while triangle waves take 32 steps.

Solving this for T, we get

$$T = \frac{f_{cpu}}{16\alpha f} - 1.$$

Inserting the relationship in (1), we get

$$\begin{aligned} T &= \frac{f_{cpu}}{16\alpha f_s \cdot 2^{\frac{1}{12}\cdot\frac{o}{64}}} - 1 \\ &= \frac{f_{cpu}}{16\alpha f_s} \cdot 2^{-\frac{1}{12}\cdot\frac{o}{64}} - 1 \\ &= (T_s + 1) \cdot 2^{-\frac{1}{12}\cdot\frac{o}{64}} - 1. \end{aligned}$$

From here we see that T_s – the timer value required to give the semitone with frequency f_s – can easily be looked up in a table. To calculate the power of 2, a simple linear approximation has been found to be good enough:

$$2^{-\frac{1}{12}\cdot\frac{o}{64}} = [2^{-\frac{1}{12}}]^{\frac{o}{64}} \approx 1 - (1 - 2^{-\frac{1}{12}}) \cdot \frac{o}{64} \approx 1 - 0.00087696 \cdot o.$$

Using this approximation in the above expression for T, we get

$$T = (T_s + 1)(1 - 0.00087696 \cdot o) - 1.$$

This only involves one floating point multiplication and performes very well.