THE QUITE OK AUDIO FORMAT

Specification **Draft 0.1**, 2023.03.17 — qoaformat.org — Dominic Szablewski

This document defines the Quite OK Audio Format (QOA). QOA encodes pulse-code modulated (PCM) audio data with up to 255 channels, sample rates from 1 up to 16777215 hertz and a bit depth of 16 bits.

The compression method employed in QOA is lossy; it discards some information from the uncompressed PCM data. For many types of audio signals this compression is “transparent”, i.e. the difference from the original file is often not audible.

QOA encodes 20 samples of 16 bit PCM data into slices of 64 bits. A single sample therefore requires 3.2 bits of storage space, resulting in a 5× compression (16 / 3.2).

A QOA file consists of an 8 byte file header, followed by a number of frames. Each frame contains an 8 byte frame header, the current 16 byte en-/decoder state per channel and 256 slices per channel. Each slice is 8 bytes wide and encodes 20 samples of audio data.

All values, including the slices, are big endian. The file layout is as follows:

```
struct {
  struct {
    char     magic[4];       // magic bytes "qoaf"
    uint32_t samples;        // no. of samples in this file
  } file_header;

  struct {
    struct {
      uint8_t  num_channels; // no. of channels
      uint24_t samplerate;   // samplerate in hz
      uint16_t fsamples;     // no. of samples in this frame
      uint16_t fsize;        // frame size (includes this header)
    } frame_header;

    struct {
      int16_t history[4];    // most recent last
      int16_t weights[4];    // most recent last
    } lms_state[num_channels];

    qoa_slice_t slices[256][num_channels];

  } frames[ceil(samples / (256 * 20))];
} qoa_file_t;
```

Each **qoa_slice_t** contains a quantized scalefactor and 20 quantized residuals:

```
┌─ qoa_slice_t ── 64 bits, 20 samples ──────────/  /──────────┐
|        Byte[0]         |        Byte[1]         \  \  Byte[7]   |
| 7  6  5  4  3  2  1  0 | 7  6  5  4  3  2  1  0 /  /    2  1  0 |
├──────────┬──────────┬──┴──┬──────┬──────┬──────┬──────\  \──┬──────┤
|  sf_quant  |  qr00  |  qr01  |  qr02  |  qr03  /  /  |  qr19  |
└──────────┴──────────┴─────┴──────┴──────┴──────┴──────/  /──┴──────┘
```

The last frame of a file may contain less than 256 slices per channel. The last slice (for each channel) in the last frame may contain less than 20 samples; the slice will still be 8 bytes wide, with the unused samples zeroed out.

A valid QOA file must have at least one frame, containing at least one channel and one sample with a samplerate between **1..16777215**.

If the total number of samples is not known by the encoder, the **samples** in the file header may be set to **0x00000000** to indicate that the encoder is “streaming”. In a streaming context, the samplerate and number of channels may differ from frame to frame. For static files (those with **samples** set to a non-zero value), each frame must have the same number of channels and same samplerate.

A decoder should support at least 8 channels. The default channel layout is **FL, FR, C, LFE, BL, BR, SL, SR**, but an application is free to interpret the channel layout in whichever way needed.

QOA predicts each audio sample based on the previously decoded ones using a “Sign-Sign Least Mean Squares Filter“ (LMS). This prediction plus the dequantized residual forms the final output sample.

A QOA file or stream is decoded with the following steps:

- for each frame
  - read the frame header
  - for each channel
    - read the LMS history & weights for this channel
  - until frame end is reached
    - for each channel
      - read one slice
      - dequantize the scalefactor[1]
      - for each quantized residual
        - transform the residual using a lookup table[2]
        - multiply the transformed residual with the scalefactor[3]
        - predict the sample using this channel's LMS state[4]
        - add the dequantized and scaled residual to the prediction to form the output sample[5]
        - update the LMS weights for this channel[6]
        - update the LMS history for this channel[7]

[1] The scalefactor for each slice is dequantized into **sf** by:
```
sf = round(pow(sf_quant + 1, 2.75))
```

[2] Each quantized residual **qr** is an index into the **dequant_tab**:
```
dequant_tab = [0.75, -0.75, 2.5, -2.5, 4.5, -4.5, 7, -7]
```

[3] The multiplication with the scalefactor is followed by rounding “to nearest, ties away from zero”, i.e. positive and negative values are treated symmetrically. The dequantized and scaled residual **r** is thus formed by:
```
r = sf * dequant_tab[qr]
r = if (r < 0) ceil(r - 0.5) else floor(r + 0.5)
```

[4] The predicted sample **p** is the sum of the products of each history sample with the corresponding weight, right shifted by 13 bits:
```
p = 0
for (n = 0; n < 4; n++)
    p += history[n] * weights[n]
p >>= 13
```

[5] The final output sample **s** is computed by **p + r**, clamped to the signed 16 bit range **-32768 .. 32767**

[6] The LMS weights are updated using the dequantized and scaled residual **r**, right shifted by 4 bits:
```
delta = r >> 4;
for (n = 0; n < 4; n++)
    weights[n] += if (history[n] < 0) -delta else delta
```

[7] The LMS history samples are updated by moving each entry down by one (evicting the least recent **history[0]**) and moving the final output sample **s** into **history[3]**.