1 Greeting

Hello world!

Culpa aliquid aperiam consequatur iste aut corrupti. Eos non harum dolor corrupti magnam consequuntur quod. Eos et quam culpa asperiores inventore ad distinctio sunt. Non in alias facilis quam quis vero quod dignissimos. Culpa ipsam velit quia. Ipsa eum enim repudiandae ut sunt doloribus.

Aut assumenda aut asperiores. Rerum ut nihil quisquam et. Quia et tempora vel. Illo excepturi soluta repellendus id debitis. Totam fuga doloribus exercitationem eius.

Et dolores amet id velit. Et ab saepe suscipit temporibus iusto. Odio dicta error voluptas veritatis ab ut aut doloribus. Et facere illo autem ea saepe accusantium eos minus.

Quis quidem porro tempora consequatur. Voluptatibus adipisci quibusdam illo magnam. Ut consequatur quia ipsum. Enim ea tenetur mollitia sed quaerat et neque officiis. Nostrum harum explicabo ut.

Culpa quia voluptas cumque officia. Impedit est ad et assumenda dolorem ex et. Voluptatem atque alias dicta ratione laborum modi rem.

2 Commands

All commands are human-readable and start with a single ASCII character.

Poot table of commands here.

3 Listing

```
// Overview of the sample packet format:
//
// +-----------------------------------------+
// | Header                  (17 bytes)      |
// +-----------------------------------------+
// | Tachometer timestamps (variable size)   |
// +-----------------------------------------+
// | Sample data             (variable size) |
// +-----------------------------------------+
//
// The size of the packet can be summarized as:
//
//   packet_size = 17 + sum(num_tachs)*3 +
//     num_frames*popcount(channel_conf)*
//     bytes_per_sample(sample_fmt)
//
// A more detailed view follows, in the form of C
// structs which are shared between code and manual.
```

```
// Sample packet header. Fixed size.
typedef struct sample_packet_header_s {
  char      header[2];     // "SP"
  uint24_t  first_frame;   // timestamp of first frame
  uint16_t  num_tachs[3]; // tach impulses per channel
  uint16_t  num_frames;    // number of frames
  uint16_t  gap;           // gap between packets
  uint16_t  channel_conf; // channel bitmap. 3 nybbles

  // Sample format. There are currently several ideas
  // for sample formats:
  //
  // * 16-bit signed integer
  // * 24-bit signed integer
  // * 16-bit half-float with 3- or 4-bit exponent
  //
  // 16-bit integers will likely not have enough
  // dynamic range to be useful. Companding 24-bit to
  // less than 16-bit may also be possible, say 12-bit.
  // This complicates packet formatting somewhat, but
  // may be worth it for somewhat higher sample rates.
  // Finally, A-law and mu-law are 8-bit compandings
  // which may be useful if we need to sample around
  // 8 kHz or more continuously.
  uint8_t   sample_fmt;

  // For some sample formats it might be useful to
  // rescale the data. This value says what the full
  // scale of the data is. In other words, where 0 dB
  // is.
  //
  // To decode say an 8-bit sample to its original
  // 24-bit range you would do this:
  //
  //   out24 = in8 * scale / 128
  //
  // You would have to be careful to use appropriate
  // data types so the computation doesn't overflow.
  //
  // Whether or not scale is used should be indicated
  // in sample_fmt.
  //
  // The maximum value of scale is 2^23.
  uint24_t  scale;
} sample_packet_header_s;
```

```
// Sample packet itself is variable size.
typedef struct sample_packet_s {
  // Header defined above
  sample_packet_header_s header;

  // Tachometer timestamps.
  // Number of entries is sum(num_tachs).
  // Values are stored one channel after the other,
  // NOT interleaved. If num_tachs = {3, 5, 4} then
  // the order will be like this:
  //
  //   0 0 0 1 1 1 1 1 2 2 2 2
  //
  // Keep in mind num_tachs can be zero for one or more
  // channel. num_tachs = {3, 0, 4} would look like:
  //
  //   0 0 0 2 2 2 2
  //
  uint24_t  *tachs;

  // Sample data is stored as a series of frames.
  // Each frame is built up of samples, and the number
  // of samples is the same as the number of bits in
  // channel_conf. Or: popcount(channel_conf).
  // The order of the samples is the same as the order
  // of ones in channel_conf.
  //
  // If all three ADCs are used, but only the first
  // three channels in each ADC, then channel_conf will
  // be "0000 0111 0111 0111" (most significant bit
  // first). Each frame will consist of 9 samples.
  //
  // The size of each sample depends on sample_fmt.
  // If 24-bit samples are used then the total amount
  // of sample data is:
  //
  //   num_frames * popcount(channel_conf) * 3  (bytes)
  //
  // In the example above, if we have 1000 frames then
  // the size of the sample data is 1000*9*3 = 27000 B.
  uint8_t   *sample_data;

} sample_packet_s;
```