KUB manual

Tomas Härdin

November 20, 2017

Contents

List of Figures

List of Tables

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 Assembly

The instrument consists of a stainless steel skeleton into which a number of plates are screwed. All plates have a printed circuit board (PCB) attached via standoffs or screwed directly to the plate. The combination of board and plate is called a module. In this section we assume all PCBs have already been soldered and cleaned.

The following tools and materials are needed:

- Wire cutters
- Wire strippers suitable for 0.4 mm and 0.9 mm diameter wire (26 and 20 AWG). Strippers with fixed holes are recommended
- A small round or semi-round file
- Torque wrenches capable of 0.3 Nm, 0.8 Nm and 2.0 Nm with the following bits:
  - PH1 Phillips
  - 9IP Torx-Plus or T10 Torx
  - 4.5 mm hex socket
  - 5.5 mm hex socket
  - 7.0 mm hex socket
- Loctite 638
- Scotch-Weld 2216

- Soldering station
- Helping hands
- 60/40 or 63/37 leaded solder
- Solder flux
- White gloves for handling silver
- Two fine paintbrushes
- Citadel Chaos Black primer
- The Army Painter WP1101 Matt Black
- Electrolube SCP03G Silver Conductive Paint
- Toothpicks
- Heat gun
- 0.62 mm^2 / 20 AWG Alpha Wire 5856 WH005 PTFE wire
- 2.4 mm $\rightarrow$ 1.2 mm Kynar shrink tube
- Masking tape
- Maxon EC motor can vice (3D printed)

The subsections that follow are ordered in the recommended order of assembly. The way the pin headers fit into corresponding sockets mean that the power4 module must be mounted first, then the cpu3 and credits modules can be mounted, and finally the fieldmill9 modules can be mounted. It's useful to mount the credits module last so that the mating of the fieldmill9 and cpu3 modules can be checked, and power-on tests performed.

In order to figure out what ID each DS18B20Z 1-wire temperature sensor has the instrument should be powered on after each module has been installed and a temperature measurement performed. The exception to this is when installing power4 and cpu3, since both are required to get an initial reading and both have a temperature sensor. Touching the DS18B20Z IC on the power4 board at this phase, thus warming it, should be enough to distinguish the two at that phase.

Use white gloves when handling any silver parts, to prevent them from being contaminated. Each aluminium plate requires 16 silver plates M3 screws (?? mm long, Torx T10) to mount it to the skeleton, 96 total.

All M3 screws are torqued to 0.8 Nm and all M2 screws are torqued to 0.3 Nm.

2.1 power4

Parts needed:

- One (1) power4 PCB, soldered and cleaned
- One (1) silver plated bottom aluminium plate (2mm thick)
- Two (2) power4 aluminium standoffs
- Four (4) PEEK washers (top kind)
- Four (4) PEEK washers (bottom kind)
- Twentyfour (24) silver plated M3 screws, ?? length inluding head (Torx T10)
- Four (4) pieces of Alpha Wire 5856 WH005 PTFE wire (TODO: length)
- 2.4 mm $\rightarrow$ 1.2 mm Kynar shrink tube
- One (1) DE-9 male connector
- A set of DE-9 panel mounting screws, washers and nuts
- Zip ties

Solder the four PTFE wires to the DE-9 connector, see figure TODO. Use Kynar shrink tubing to protect the solder points.

Fasten the DE-9 connector to the aluminium plate (inside or outside??) using the screws, washers and nuts. Torque to 0.8 Nm. Screw the standoffs to the plate, 0.8 Nm.

Solder the four PTFE wires to the appropriate places on the power4 board, see figure TODO. Zip tie the wires together at both ends or in the middle if there isn't enough room.

Place bottom PEEK washers on the standoffs, then the power4 board on the standoffs and finally the top wasters on top. Screw everything in place using M3 screws, 0.8 Nm.

Screw the finished module to the skeleton using M3 screws, 0.8 Nm.

2.2 cpu3

Parts needed:

- One (1) cpu3 PCB, soldered and cleaned
- One (1) silver plated cpu3 aluminium plate
- One (1) cpu3/credits aluminium standoff, short kind
- One (1) cpu3/credits aluminium standoff, long kind

- Four (4) PEEK washers (top kind)
- Four (4) PEEK washers (bottom kind)
- Twentyfour (24) silver plated M3 screws, ?? length inluding head (Torx T10)

Be careful to use the aluminium plate with holes countersunk in the correct direction. The credits plates are mirror images of the cpu3 plates.

Screw the standoffs to the plate using M3 screws, 0.8 Nm.

Places bottom PEEK washers, board and top PEEK washers on the standoffs. Screw in place using M3 screws, 0.8 Nm.

Screw the finished module to the skeleton using M3 screws, 0.8 Nm.

Perform a power-on test. Distinguish the two DS18B20Z temperature sensors now in the instrument by touching one of them, thus warming it. Write down which ID corresponds to which of the two sensors.

2.3 fieldmill9

Parts needed:

- One (1) fieldmill9 / fieldmill_top_plate4 PCB assembly
- One (1) fieldmill aluminium top plate
- One (1) fieldmill shutter plate
- One (1) Maxon 349694 EC motor
- Stencil with a 15x5 mm hole
- Two (2) 100 m stainless steel motor shims
- One (1) ITR20001/T IR reflex coupler
- Three (3) M2 screws, 6 mm length including head (Phillips PH1)
- Four (4) silver plated M3 screws, 10 mm length including head (Torx T10)
- Sixteen (16) silver plated M3 screws. Same as above??
- Four (4) M3 washers
- Four (4) M3 hex nuts
- One (1) M4 X 8/8 hex screw with 2 mm hole drilled through, preferably silver plated or silver painted if *someone* forgot to order silver plated M4 screws
- One (1) silver plated M4 flange nut

Before doing anything else, perform the following electrical tests using a multimeter

- Check connections to every +2.5V and -2.5V regulator. There's eight $(4 + 4 = 8)$ regulators per board
- Measure resistance from GND to every rail; +3.3V, +24V, +5V, -5V, every +2.5V and every -2.5V. Measure in both directions. Values should all be at least 1 kΩ
- Measure resistance from either side of R56 to GND. Measure in both directions. Value should be at least 8 kΩ
- Measure resistance across every NANOSMD PTC. Direction is not important. Values should be no more than 10 Ω. Each board has thirteen (13) PTCs

Applying power while these values are out of spec may cause PERMANENT DAMAGE to op-amps, DACs and/or linear regulators. In the worst case the +-5V regulator (TMR 6-2421WI) may also become damaged. Once checked and within spec the assembly process may continue:

Cut motor wires to 6-7 mm length. Strip so that 3 mm of insulation remains, with the wire stripper set to 0.4 mm. Twist and tin the ends of the wires. See figure 1 on the following page.

Put the stencil on the rotor can, tape it down using masking tape and paint the area matt black. One way to do this is to spray primer into a glass or glazed ceramic cup then dip a fine paintbrush into the primer and paint it onto the can. DO NOT USE A PLASTIC CUP TO HOLD THE PRIMER - IT WILL MELT. Wait at least 30 minutes for the primer to dry, then apply the matt black paint on top of it. The black painted area should cover roughly a 90 arc of the edge of the can. See figure 2 on the next page.

Use a small semi-round file to grind away the perforation remains in the middle hole of the PCB, so that a motor will fit loosely.

Mount the Maxon EC motor in the motor hole with two motor shims inbetween. The shims should raise the motor enough that a small lip / edge can be felt on the top side of the assembly between the motor housing and the fieldmill_top_plate4 PCB. Solder the five motor wires to the motor wire pads on the PCB.

Bend and cut the IR reflex coupler leads so that the reflex coupler looks directly at the EC motor's rotor can when inserted into its 2x3 socket (IR2 reference on PCB). See figure 2 on the following page. Bending then trimming all leads to the same length as the shortest lead will accomplish this, The resulting lead length should be 7 mm from the bottom of the reflex coupler. Insert the reflex coupler into its socket. The gap between the top of the 2x3 socket and the bottom of the reflex coupler should be 2 - 3 mm. Solder the reflex coupler into the socket. Use plenty of flux so the leads and socket are guaranteed to be soldered together. Some extra solder can be applied to the 2x3 socket pads at

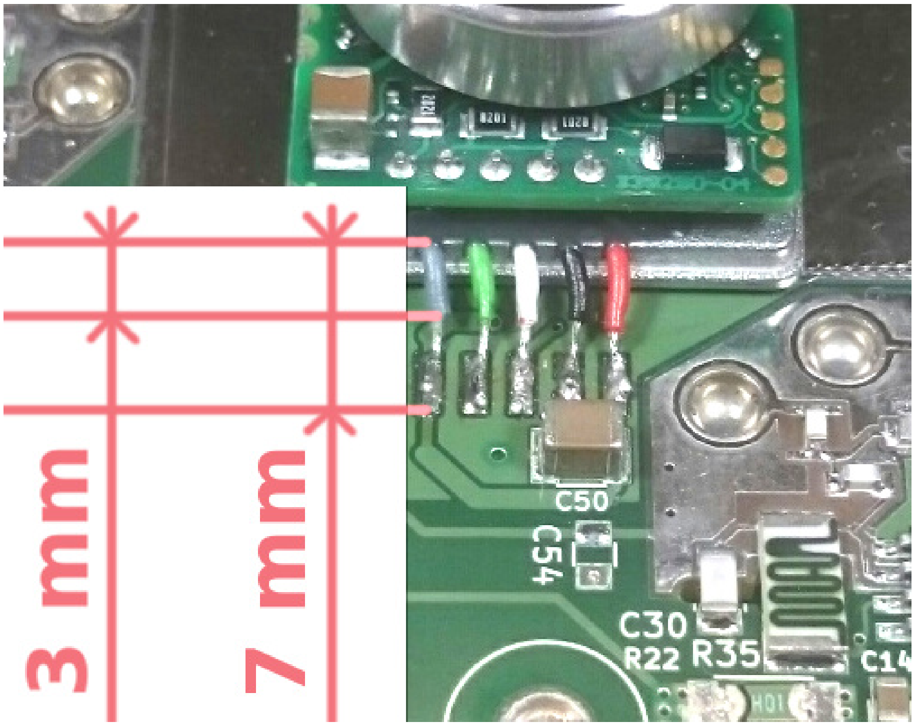

Figure 1: Motor wires soldered to their pads. Wire and insulation lengths marked.

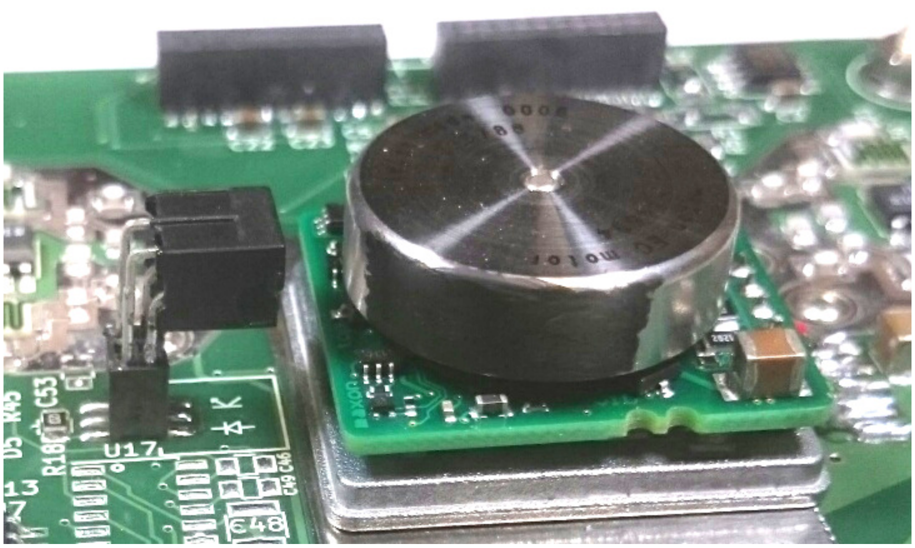

Figure 2: Reflex coupler and motor can. Black paint also visible on motor can.

this point. Carefully test that pulling on the reflex coupler doesn't cause it to come out. Check continuity with a multimeter, and that adjacent pins aren't shorted.

Screw the motor + PCB assembly to the aluminium plate using the M2 screws for the motor and silver plated M3 screws, washers and nuts for the PCB. First screw in the screws lightly, then tighten using the torque wrench. Use 0.3 Nm for the M2 screws and 0.8 Nm for the M3 nuts.

Use a toothpick to paint the motor bearing with SCP03G silver paint. Be careful not to get silver paint on the motor axis. There should be a slight resistance in the bearings which will go away after the first initial minutes of operation.

Use Loctite 638 to glue the M4 screw to the motor axis. Mount the screw so that the head points downward, toward the instrument, and so that it is flush with the inner ring of the motor bearing. When the glue has dried, put the motor can in the specially shaped vice. Put the shutter plate on the M4 screw and fasten it using the M4 flange nut. Torque to 2.0 Nm. Drop some SCP03G silver paint into the top of the screw so that it gets a good electrical connection to the motor shaft.

Screw each module to the skeleton using M3 screws, 0.8 Nm.

Perform a power-on test. Check that the +5V, -5V and +3.3V rails, and every +2.5V and every -2.5V rail are as expected. Measure the output of every channel while VGND = 0 V. The average voltage of each output should be within +- 200 mV of VGND.

Take a temperature reading after each fieldmill9 module has been inserted, and make a note of the new sensor ID each time and what module it corresponds to.

Start the motors and run them for at least five minutes. Measure the resistance between motor axis and instrument ground. The resistance should be less than 10 Ω.

2.4 credits

- One (1) credits PCB, soldered and cleaned
- One (1) silver plated credits aluminium plate
- One (1) cpu3/credits aluminium standoff, short kind
- One (1) cpu3/credits aluminium standoff, long kind
- Four (4) PEEK washers (top kind)
- Four (4) PEEK washers (bottom kind)
- Twentyfour (24) silver plated M3 screws, ?? length inluding head (Torx T10)

Be careful to use the aluminium plate with holes countersunk in the correct direction. The credits plates are mirror images of the cpu3 plates.

Screw the standoffs to the plate using M3 screws, 0.8 Nm.

Places bottom PEEK washers, board and top PEEK washers on the standoffs. Screw in place using M3 screws, 0.8 Nm.

Screw the finished module to the skeleton using M3 screws, 0.8 Nm.

Perform a power-on test and a temperature test. Take note of the new temperature sensor ID.

3 Command and response summary

All commands are human-readable, start with a single ASCII character and are termianted with a line ending. Comments may be added by inserting a hash sign ('#'). This causes the rest of the line to be ignored (characters are consumed and discarded until end-of-line). Mistakes can be corrected with backspace (BS, ASCII code 8) or delete (DEL, ASCII code 127), both of which are treated as a backspace. Line reading also understands escape (ESC, ASCII code 27), which aborts the current command.

Parameter parsing is handled by sscanf(), which allows for the same command character to take a varying number of parameters. An example of this is the 'M' command which exists in two-parameter and three-parameter forms:

```
M0 10       # Set speed of motor 0 to 10
M10 10 10   # Set speed of all motors to 10
```

Line endings can either be carriage return ('\r', ASCII code 13) or linefeed ('\n', ASCII code 10), but never both in the same line. In other words both Unix and Mac line endings are OK, but Windows line endings ("\r\n") are not. This ensures that *echo*, *minicom* and *picocom* work as expected. Output from the instrument is however terminated by Windows line endings ("\r\n"), in order to play nice with *minicom* and *picocom*. An ESC ('\x1B', ASCII code 27) anywhere in a line aborts that command. The instrument will respond to ESC immediately. There is no need to terminate ESC with newline.

Output from the instrument is delimited into frames. A frame start with the string "BEGIN\r\n" and ends with the string "END\r\n". Each frame is of a specific type, which is given on the line after BEGIN and preceded by a start (*) and terminated by \r\n. For example, the reply to the "M10 10 10" command example given earlier is of type MTR_PWM and would therefore be:

```
BEGIN
*MTR_PWM
10 10 10
END
```

Tables 1 and 2 summarize all commands and their parameters. More detailed descriptions of each command and response are given in the sections that follow.

Char	Description	Response
v	Measure system voltages	VOLTAGES
V	Enable 24V and +-5V	VOLTAGES
B	Disable 24V and +-5V	VOLTAGES
m	Read motor PWMs	MTR_PWM
K	Set motor PWMs to 50%	MTR_PWM
r	Measure motor speeds in RPM	MTR_SPD
1	List 1-wire device ROMs	ONEWIRE
!	Search for 1-wire devices	ONEWIRE
t	Measure temperatures	TEMPS
c	Read clock in cycles	CLOCK
?	Print help	INFO
\x1B (ESC)	Stop measurement, abort current recvline(). Can occur anywhere in a line, does not have to be terminated with newline.	ESC

Table 1: Command table, parameterless commands

Char	Parameter syntax	Description	Response
M	id pwm	Set motor PWM	MTR_PWM
M	pwm0 pwm1 pwm2	Set all motor PWMs	MTR_PWM
		Configure ADC	
		Read ADC configuration	
		Read registers ($0000 - $00FF)	
		Write registers ($0000 - $00FF)	
		Read RAM ($0100 - $FFFF)	
		Write RAM ($0100 - $FFFF)	
		Read EEPROM ($000 - $FFF)	
		Write EEPROM ($000 - $FFF)	
		Read ROM ($00000 - $1FFFF)	
		Read fuses	
C	cycles	Set clock in cycles	CLOCK
w	cycles	Wait given number of cycles	
		Configure measurement (block size + gap)	
		Read measurement configuration	
		Start measurement	

Table 2: Command table, commands with parameters

4 Responses

Since many commands share the same responses

4.1 INFO

Informative text. Display contents in text window or terminal. Should be displayed with normal colors. Example:

```
BEGIN
*INFO
Starting temperature conversion...
END
```

4.2 WARNING

Warning. Should be displayed with a yellowish or maybe orange color.

```
BEGIN
*WARNING
Instrument issues no warnings currently,
but may in the future.
END
```

4.3 ERROR

Some kind of error. Should be displayed with a reddish color.

```
BEGIN
*ERROR
One or more of PWMS 1111, 2222, and 3333
is greater than MOTOR_TOP = 1023
END
```

4.4 VOLTAGES

TODO

4.5 MTR_PWM

PWM value for each of the three motors, as integers between 0 - 1023. These correspond roughly to 0 - 6000 RPM, but not exactly. For more information see EC20 datasheet. Example response after issuing a "K" command:

```
BEGIN
*MTR_PWM
511 511 511
END
```

4.6 MTR_SPD

Motor speeds in RPM, and the number of tachometer impulses detected. For a sensible output at least two tachs must have been seen for each motor. Exact format isn't finalized, so display output as if it were INFO for now.

```
BEGIN
*MTR_SPD
Motor 0: 0 tachs (0 RPM)
Motor 1: 1 tachs (0 RPM)
Motor 2: 2 tachs (1234 RPM)
END
```

4.7 ONEWIRE

1-Wire devices. One line per device containing a 64-bit hexadecimal ROM string (16 characters).

```
BEGIN
*ONEWIRE
28d09948090000ec
286a1a690900005e
28ad7548090000c5
END
```

4.8 TEMPS

Temperature readings. One line per DS18B20Z device with 64-bit hexadecimal ROM and decimal temperature values in degrees Celsius. Range is -40..+125 with a resolution of 1/16 degrees Celsius.

```
BEGIN
*TEMPS
28d09948090000ec 24.12
286a1a690900005e 24.62
28ad7548090000c5 -18.56
END
```

4.9 CLOCK

Current time in CPU cycles as a 64-bit decimal value. May roll over in theory, but it would take 79,000 years unless the user specifically set the clock to something close to the maximum value (18446744073709551615).

```
BEGIN
*CLOCK
3702994144
END
```

4.10 SAMPLES

Sample data. This response is binary. It is possible, but very unlikely, that the binary data contains the string "END\r\n". To avoid trouble it is best to compute the exact number of bytes in the sample data by following listing 6.

```
BEGIN
*SAMPLES
[sample_packet_s]
END
```

5 Commands

All commands here are those that take parameters.

5.1 Set motor PWMs ('M')

Set one or more OCR1x. PWM values must be between 0 - 1023.

5.1.1 Two parameter form (id pwm)

5.1.2 Three parameter form (pwm0 pwm1 pwm2)

6 Listing

```
// Overview of the sample packet format:
//
// +--------------------------------------+
// | Header                (19 bytes)     |
// +--------------------------------------+
// | Tachometer timestamps (variable size) |
// +--------------------------------------+
// | Sample data           (variable size) |
// +--------------------------------------+
//
// The size of the packet can be summarized as:
//
//   packet_size = 19 + sum(num_tachs)*3 +
//     num_temps*4 +
//     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"
  uint8_t   version;      // format version
  uint24_t  first_frame;  // timestamp of first frame
  uint8_t   num_temps;    // DS18B20Z outputs (0..6)
  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;
```

```
// Temperature reading structure.
// Since temperature conversions take around 750 ms
// not every sample packet will have temperatures.
typedef struct temperature_s {
  // Bytes 1-2 of DS18B20Z ROM is enough to uniquely
  // identify the ones we have:
  //
  //  286a1a690900005e -> 6a 1a
  //  28f72a6909000021 -> f7 2a
  uint8_t   rom12[2];

  // Temperature in degrees Celsius * 16
  int16_t   temp;
} temperature_s;


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

  // Temperatures with structures defined above.
  // A reading like:
  //
  //  286a1a690900005e 23.12
  //  28f72a6909000021 -3.87
  //
  // will be encoded as:
  //
  //  6a 1a 71 01 f7 2a c2 ff
  //
  temperature_s *temps;

  // 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;
```