KUB manual

Tomas Härdin

October 15, 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 and 0.8 Nm with the following bits:
  - PH1 Phillips
  - 9IP Torx-Plus or T10 Torx
  - 5.5 mm hex socket
  - 7.0 mm hex socket
  - ?? mm square socket? For DE-9. Or suitable hex socket
- Loctite 603
- 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

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.

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 → 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.

Screw the DE-9 connector to the aluminium plate (inside or outside??), ?? 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.

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

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.

Put the stencil on the rotor can and paint the area matt black. One way to do this is to spray primer into a plastic cup then dip a fine paintbrush into the primer and paint it onto the can. Once the primer has dried the matt black paint can be applied on top of it. The black paint should cover roughly a 90 arc of the edge of the can.

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). Solder the reflex coupler into the socket. Use plenty of flux

so the leads and socket are guaranteed to be soldered together. Carefully test that pulling on the reflex coupler doesn't cause it to come out.

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. There should be a slight resistance in the bearings which will go away after the first initial minutes of operation.

Use Loctite 603 to glue the M4 screw to the motor axis. Mount the screw so that the head points downward, toward the instrument. When the glue has dried, put a shutter plate on the M4 screw and fasten it using the M4 flange nut. Torque to ?? 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 power-on test and motor test. Measure the resistance between motor axis and instrument ground after running the motor for a few minutes. 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 thermometer test.

3 Commands

All commands are human-readable, start with a single ASCII character and are termianted with a line ending. If the command takes no parameters then the line ending is optional. 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). 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 both *echo* and *minicom* works as expected. Output from the instrument is terminated by Windows line endings however, in order to work well with *minicom*.

Table 1 on the following page summarizes all commands and their parameters. More detailed descriptions of each command are given in the subsections that follow.

Command	Parameter count	Parameter syntax	Description
v	0		Measure system voltages
V	0		Enable 24V and +-5V
B	0		Disable 24V and +-5V
m	0		Read motor speeds
M	2	ID spd	Set motor speed
M	3	spd spd spd	Set motor speeds
K	0		Set motor speeds to 50%
	0		Stop all motors
	1	ID	Stop specific motor
	0		Measure motor speeds in RPM
	1		Measure specific motor speed in RPM
1	0		List 1-wire device ROMs
!	0		Search for 1-wire devices
			Measure temperatures
			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
			Read clock
			Set clock
			Configure measurement (block size + gap)
			Read measurement configuration
			Start measurement
\x1B (ESC)			Stop measurement

Table 1: Command table

3.1 Read motor speed ('m')

Prints three integers containing OCR1A, OCR1B and OCR1C respectively. 0 - 255 roughly corresponding to 0 - 100% or 0 - 6000 RPM.

3.2 Set motor speed(s) ('M')

Hardware will refuse if values are so low that they risk turning the motors off.

4 Listing

```
// Overview of the sample packet format:
//
// +-----------------------------------------+
// | Header                  (18 bytes)      |
// +-----------------------------------------+
// | Tachometer timestamps (variable size)   |
// +-----------------------------------------+
// | Sample data             (variable size) |
// +-----------------------------------------+
//
// The size of the packet can be summarized as:
//
//   packet_size = 18 + 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"
  uint8_t   version;      // format version
  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 companding
  // 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;
```