

## FrSky Telemetry details:

To run the FrSky build of er9x you will need to make some modifications to the hardware in your radio and telemetry receiver. These modifications require soldering of small tracks on the radio circuit board. Be warned that it is possible to damage the board permanently if care is not taken.

An overview of the hardware modifications required to run the ER9x-FRSKY firmware build are:

1. Modify the TH9x to provide a serial port RS232 TTL interface circuit.
2. Modify the telemetry receiver D8R V2 to wire out the ADC1 voltage input. ( If using BEC/ESC )
3. Make up the special voltage sensor with divide ratio 0.13 ( If using BEC/ESC )
4. Load ER9X-FRSKY firmware in the TH9x

Gruvin has written an excellent description of the modifications needed here: <http://code.google.com/p/gruvin9x/wiki/FrskyInterfacing>.

**Note: I believe that the emitter and collector of the BC857 (Q3) are reversed on Gruvin's diagram. The emitter should connect to Vcc and the collector should connect to R75 and R76.**

*Nb. Gruvin is actually writing a side branch of er9x called gruvin9x specifically dedicated to FrSky telemetry. If you are serious about getting more in depth with the FrSky telemetry functions you might want to try it out. This er9x build provides basic telemetry support and has the beauty of incorporating all of Erazz's changes and bug fixes as he makes them. Gruvin has now deviated from er9x a bit so the er9x functions may take a while to make it to his code.*

An alternative to the two transistor RS232 TTL level converter circuit is to use a Maxim MAX232 IC. This is the approach I have taken. It simply requires the IC and four capacitors.

**Pisces** from RCGroups has also posted excellent build instructions for his modifications to a radio fitted with **s\_mack's** programming and EL board which I have included below with his permission:

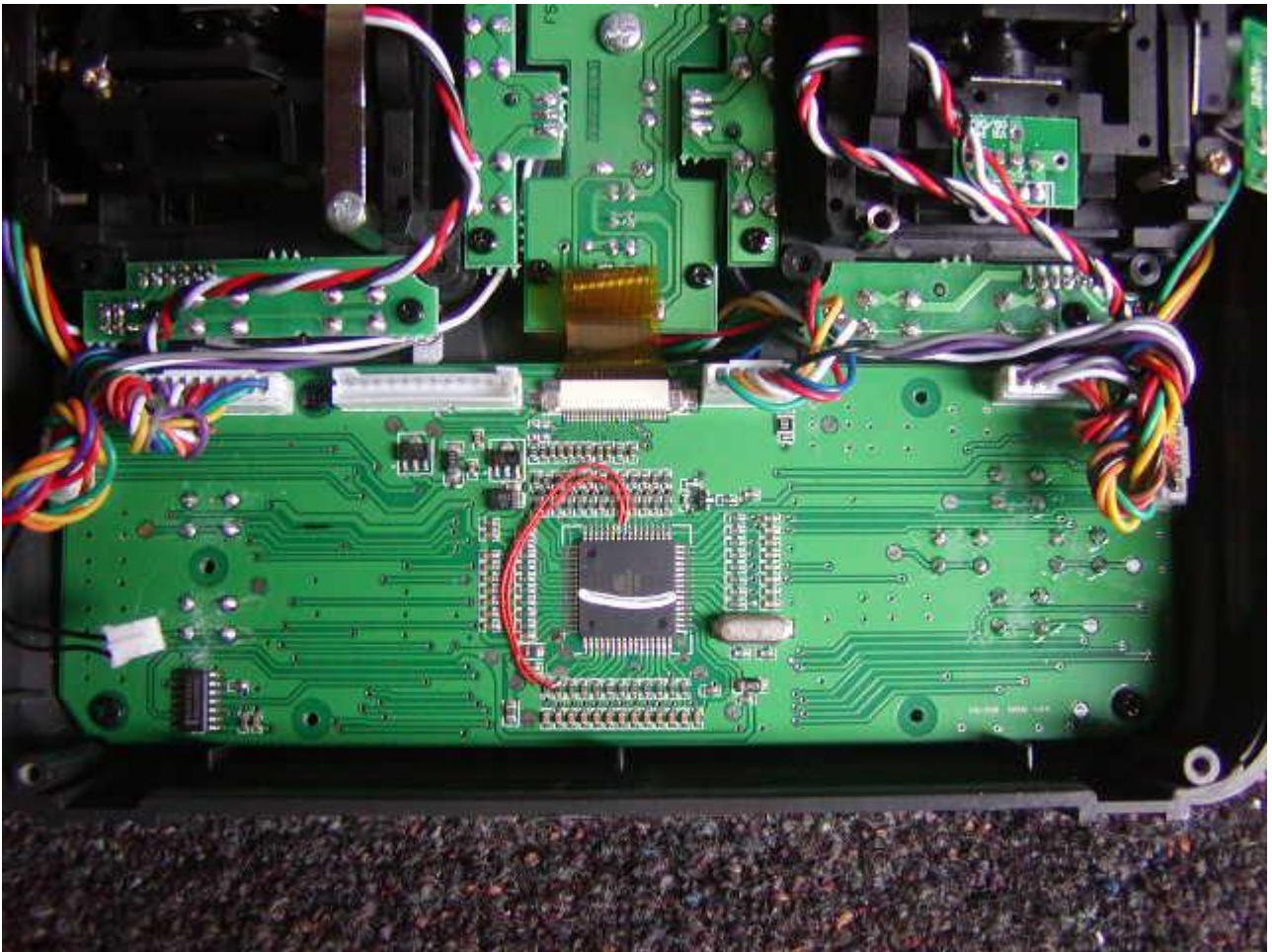
Gruvin9x indicated that the two transistor RS232 TTL level converter has the advantage that it can be left permanently connected on top of the wiring to the Atmega programmer. This has been verified by a number of RCGroups users who have installed the two transistor level converter.

*This modification is based on the very good instructions by Gruvin9x.*

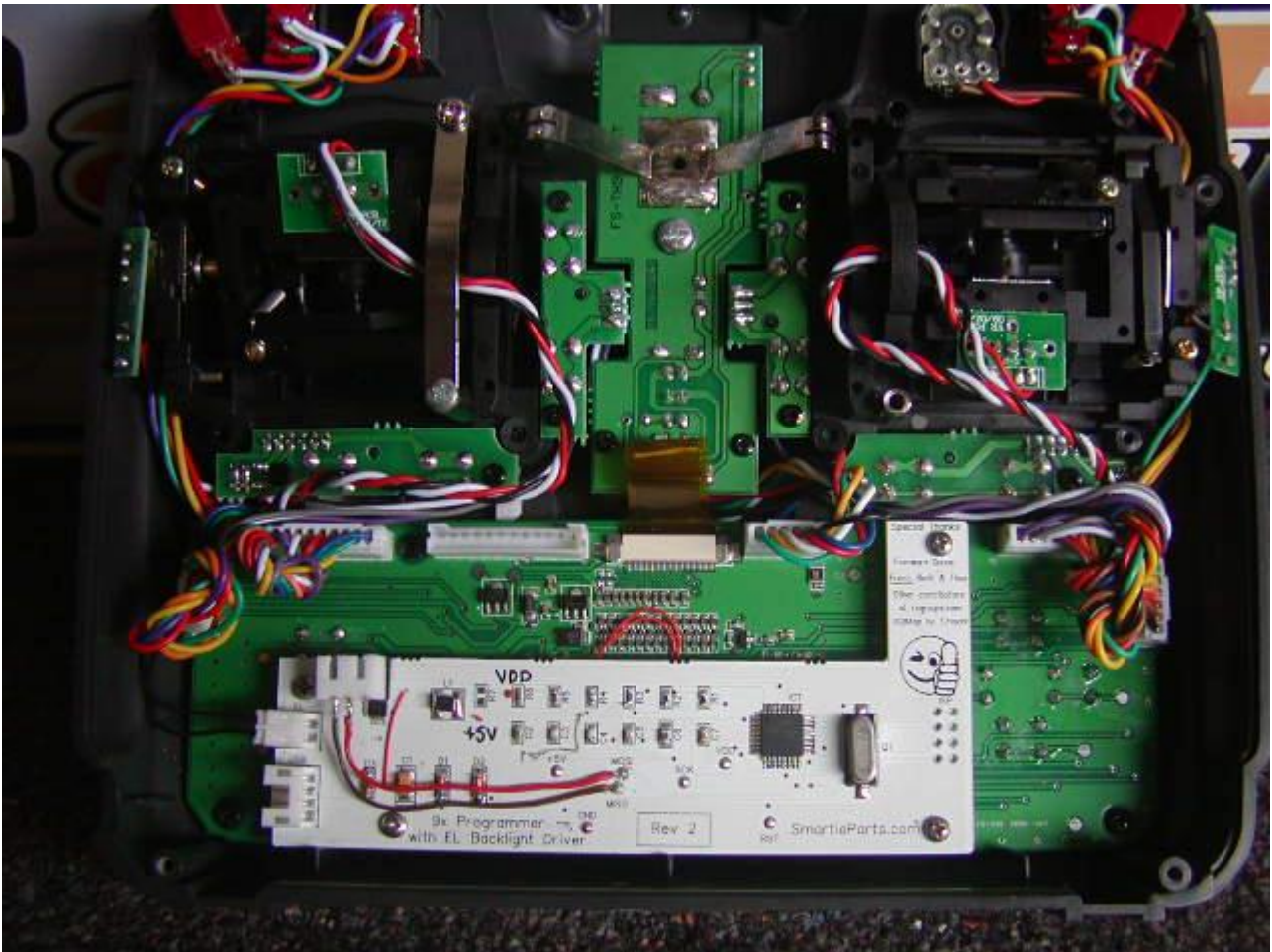
<http://code.google.com/p/gruvin9x/wiki/FrskyInterfacing>

*So please read these Gruvin9x instructions first. Note that i have changed the transistor Q3 pins e and c around so they are now reversed on my veroboard circuit. This is shown on my working notes page.*

*Step1. Move the THR CUT and AIL DR switches from the Atmega MOSI and MISO pins and wire to pins 41 and 42, (PC6 and PC7), on top row of Atmega pins. I used wire wrap wire. This is fine single core wire. I did not remove the resistors but instead cut the track very carefully with a sharp knife blade very close to the top of the 200ohm resistors. I then tack soldered the wires to the top of the 200ohm resistors. Route the wires clear of the ATmega/pogo pin pads.*



*Step2. Add the connector to the S Mack Pogo Pin PCB and wire to MOSI and MISO. I used a Lipo 2s balance connector. The connector is attached with double sided tape onto the PCB.*



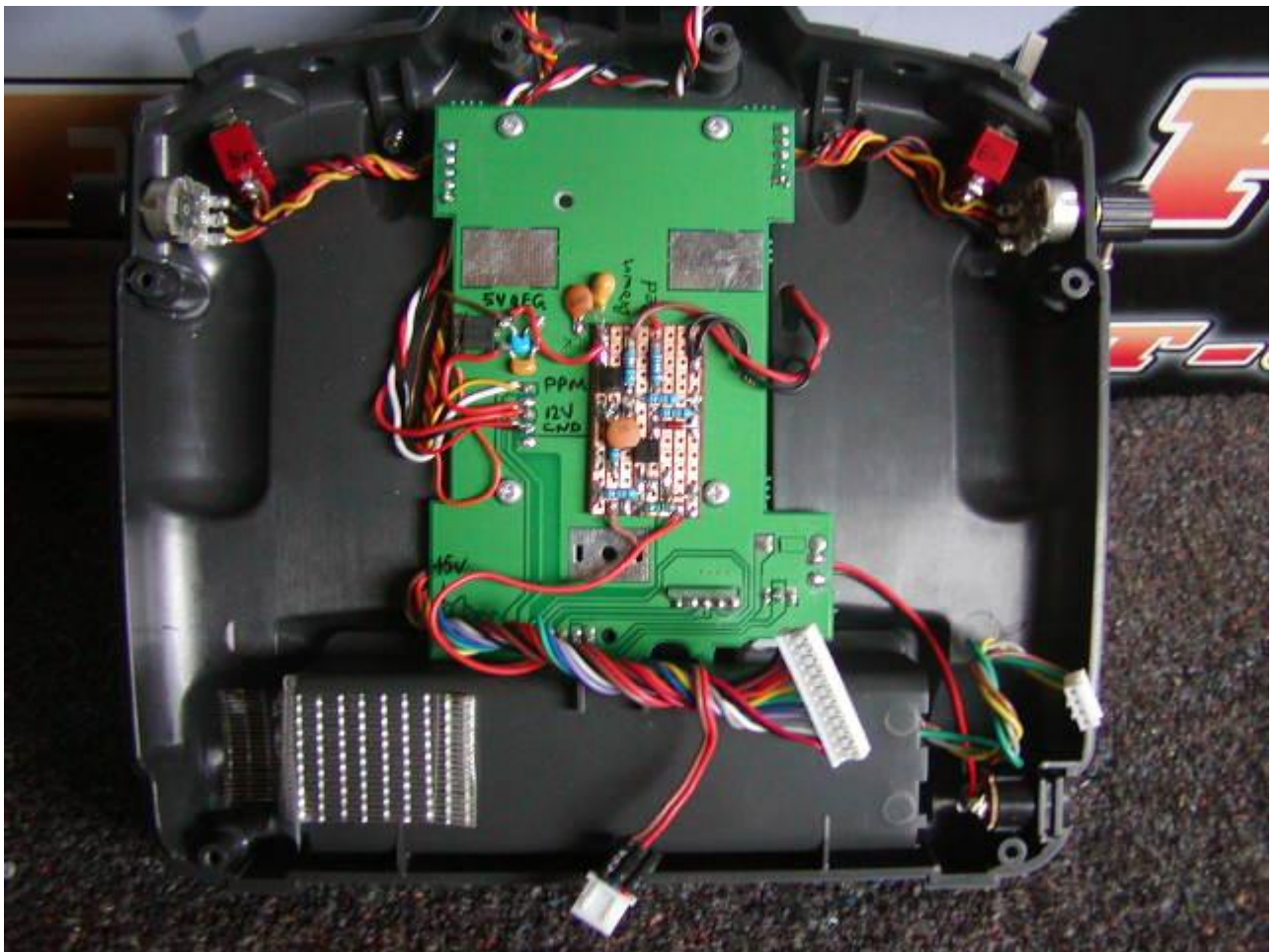
Step3. Make up the veroboard 2 transistor TTL-RS232 level converter/inverter. This can be other circuits for example a MAX232 IC TTL-RS232 converter. Or for example the method Phos used with the converter outside on the back of the TH9x case.

<http://www.rcgroups.com/forums/showpost.php?p=17344878&postcount=4799>

The veroboard construction is copper side up. This makes it easy to follow the wiring layout. The wiring /component pigtails were cut flush on the blank bottom side of the veroboard. This keeps the back of the board flat. Double sided tape is used to attach the veroboard to the inside rear PCB of the TH9x as shown in the photo.

Note that i soldered a +5V three terminal regulator in the TH9x PCB close to the Tx module bay pins. The veroboard is powered from this +5V regulator. The regulator input voltage is switched battery and is shown on the PCB Tx module pins as 12V. ( 3s LiPo TH9x battery ) This is a safe way of powering the veroboard RS232 TTL converter with its own +5V regulator. But of course the TH9x mainboard +5V regulator can be used to power the RS232 TTL level converter.







TH9X  $\leftarrow$  35pF  $\leftarrow$  0.01pF  $\leftarrow$  +5V

TH9X (brown)  $\rightarrow$  +5V

TH9X (red)  $\rightarrow$  +5V

RS232 TTL 0-5V (brown)  $\rightarrow$  +5V

RS232 TTL 0-5V (red)  $\rightarrow$  +5V

TH9X SERIAL/FRSKY INTERFACE (MODIFIED TO REVERSE E & C of Q3 BC557)

RS232 LEVEL CONVERTER/TTL-RS232 0-5V INVERTER

Frsky DJT Telemetry RS232 Connector

DJT MODULE

3 MISO

2 MISO

TELEM\_TX

TELEM\_RX

to PIN 1 RXD/MISO

to PIN 2 RXD/MISO

to PIN 3 TXD/MISO

to PIN 4 TXD/MISO

to PIN 5 TXD/MISO

to PIN 6 TXD/MISO

to PIN 7 TXD/MISO

to PIN 8 TXD/MISO

to PIN 9 TXD/MISO

to PIN 10 TXD/MISO

to PIN 11 TXD/MISO

to PIN 12 TXD/MISO

to PIN 13 TXD/MISO

to PIN 14 TXD/MISO

to PIN 15 TXD/MISO

to PIN 16 TXD/MISO

to PIN 17 TXD/MISO

to PIN 18 TXD/MISO

to PIN 19 TXD/MISO

to PIN 20 TXD/MISO

to PIN 21 TXD/MISO

to PIN 22 TXD/MISO

to PIN 23 TXD/MISO

to PIN 24 TXD/MISO

to PIN 25 TXD/MISO

to PIN 26 TXD/MISO

to PIN 27 TXD/MISO

to PIN 28 TXD/MISO

to PIN 29 TXD/MISO

to PIN 30 TXD/MISO

to PIN 31 TXD/MISO

to PIN 32 TXD/MISO

to PIN 33 TXD/MISO

to PIN 34 TXD/MISO

to PIN 35 TXD/MISO

to PIN 36 TXD/MISO

to PIN 37 TXD/MISO

to PIN 38 TXD/MISO

to PIN 39 TXD/MISO

to PIN 40 TXD/MISO

to PIN 41 TXD/MISO

to PIN 42 TXD/MISO

to PIN 43 TXD/MISO

to PIN 44 TXD/MISO

to PIN 45 TXD/MISO

to PIN 46 TXD/MISO

to PIN 47 TXD/MISO

to PIN 48 TXD/MISO

to PIN 49 TXD/MISO

to PIN 50 TXD/MISO

to PIN 51 TXD/MISO

to PIN 52 TXD/MISO

to PIN 53 TXD/MISO

to PIN 54 TXD/MISO

to PIN 55 TXD/MISO

to PIN 56 TXD/MISO

to PIN 57 TXD/MISO

to PIN 58 TXD/MISO

to PIN 59 TXD/MISO

to PIN 60 TXD/MISO

to PIN 61 TXD/MISO

to PIN 62 TXD/MISO

to PIN 63 TXD/MISO

to PIN 64 TXD/MISO

to PIN 65 TXD/MISO

to PIN 66 TXD/MISO

to PIN 67 TXD/MISO

to PIN 68 TXD/MISO

to PIN 69 TXD/MISO

to PIN 70 TXD/MISO

to PIN 71 TXD/MISO

to PIN 72 TXD/MISO

to PIN 73 TXD/MISO

to PIN 74 TXD/MISO

to PIN 75 TXD/MISO

to PIN 76 TXD/MISO

to PIN 77 TXD/MISO

to PIN 78 TXD/MISO

to PIN 79 TXD/MISO

to PIN 80 TXD/MISO

to PIN 81 TXD/MISO

to PIN 82 TXD/MISO

to PIN 83 TXD/MISO

to PIN 84 TXD/MISO

to PIN 85 TXD/MISO

to PIN 86 TXD/MISO

to PIN 87 TXD/MISO

to PIN 88 TXD/MISO

to PIN 89 TXD/MISO

to PIN 90 TXD/MISO

to PIN 91 TXD/MISO

to PIN 92 TXD/MISO

to PIN 93 TXD/MISO

to PIN 94 TXD/MISO

to PIN 95 TXD/MISO

to PIN 96 TXD/MISO

to PIN 97 TXD/MISO

to PIN 98 TXD/MISO

to PIN 99 TXD/MISO

to PIN 100 TXD/MISO

to PIN 101 TXD/MISO

to PIN 102 TXD/MISO

to PIN 103 TXD/MISO

to PIN 104 TXD/MISO

to PIN 105 TXD/MISO

to PIN 106 TXD/MISO

to PIN 107 TXD/MISO

to PIN 108 TXD/MISO

to PIN 109 TXD/MISO

to PIN 110 TXD/MISO

to PIN 111 TXD/MISO

to PIN 112 TXD/MISO

to PIN 113 TXD/MISO

to PIN 114 TXD/MISO

to PIN 115 TXD/MISO

to PIN 116 TXD/MISO

to PIN 117 TXD/MISO

to PIN 118 TXD/MISO

to PIN 119 TXD/MISO

to PIN 120 TXD/MISO

to PIN 121 TXD/MISO

to PIN 122 TXD/MISO

to PIN 123 TXD/MISO

to PIN 124 TXD/MISO

to PIN 125 TXD/MISO

to PIN 126 TXD/MISO

to PIN 127 TXD/MISO

to PIN 128 TXD/MISO

to PIN 129 TXD/MISO

to PIN 130 TXD/MISO

to PIN 131 TXD/MISO

to PIN 132 TXD/MISO

to PIN 133 TXD/MISO

to PIN 134 TXD/MISO

to PIN 135 TXD/MISO

to PIN 136 TXD/MISO

to PIN 137 TXD/MISO

to PIN 138 TXD/MISO

to PIN 139 TXD/MISO

to PIN 140 TXD/MISO

to PIN 141 TXD/MISO

to PIN 142 TXD/MISO

to PIN 143 TXD/MISO

to PIN 144 TXD/MISO

to PIN 145 TXD/MISO

to PIN 146 TXD/MISO

to PIN 147 TXD/MISO

to PIN 148 TXD/MISO

to PIN 149 TXD/MISO

to PIN 150 TXD/MISO

to PIN 151 TXD/MISO

to PIN 152 TXD/MISO

to PIN 153 TXD/MISO

to PIN 154 TXD/MISO

to PIN 155 TXD/MISO

to PIN 156 TXD/MISO

to PIN 157 TXD/MISO

to PIN 158 TXD/MISO

to PIN 159 TXD/MISO

to PIN 160 TXD/MISO

to PIN 161 TXD/MISO

to PIN 162 TXD/MISO

to PIN 163 TXD/MISO

to PIN 164 TXD/MISO

to PIN 165 TXD/MISO

to PIN 166 TXD/MISO

to PIN 167 TXD/MISO

to PIN 168 TXD/MISO

to PIN 169 TXD/MISO

to PIN 170 TXD/MISO

to PIN 171 TXD/MISO

to PIN 172 TXD/MISO

to PIN 173 TXD/MISO

to PIN 174 TXD/MISO

to PIN 175 TXD/MISO

to PIN 176 TXD/MISO

to PIN 177 TXD/MISO

to PIN 178 TXD/MISO

to PIN 179 TXD/MISO

to PIN 180 TXD/MISO

to PIN 181 TXD/MISO

to PIN 182 TXD/MISO

to PIN 183 TXD/MISO

to PIN 184 TXD/MISO

to PIN 185 TXD/MISO

to PIN 186 TXD/MISO

to PIN 187 TXD/MISO

to PIN 188 TXD/MISO

to PIN 189 TXD/MISO

to PIN 190 TXD/MISO

to PIN 191 TXD/MISO

to PIN 192 TXD/MISO

to PIN 193 TXD/MISO

to PIN 194 TXD/MISO

to PIN 195 TXD/MISO

to PIN 196 TXD/MISO

to PIN 197 TXD/MISO

to PIN 198 TXD/MISO

to PIN 199 TXD/MISO

to PIN 200 TXD/MISO

to PIN 201 TXD/MISO

to PIN 202 TXD/MISO

to PIN 203 TXD/MISO

to PIN 204 TXD/MISO

to PIN 205 TXD/MISO

to PIN 206 TXD/MISO

to PIN 207 TXD/MISO

to PIN 208 TXD/MISO

to PIN 209 TXD/MISO

to PIN 210 TXD/MISO

to PIN 211 TXD/MISO

to PIN 212 TXD/MISO

to PIN 213 TXD/MISO

to PIN 214 TXD/MISO

to PIN 215 TXD/MISO

to PIN 216 TXD/MISO

to PIN 217 TXD/MISO

to PIN 218 TXD/MISO

to PIN 219 TXD/MISO

to PIN 220 TXD/MISO

to PIN 221 TXD/MISO

to PIN 222 TXD/MISO

to PIN 223 TXD/MISO

to PIN 224 TXD/MISO

to PIN 225 TXD/MISO

to PIN 226 TXD/MISO

to PIN 227 TXD/MISO

to PIN 228 TXD/MISO

to PIN 229 TXD/MISO

to PIN 230 TXD/MISO

to PIN 231 TXD/MISO

to PIN 232 TXD/MISO

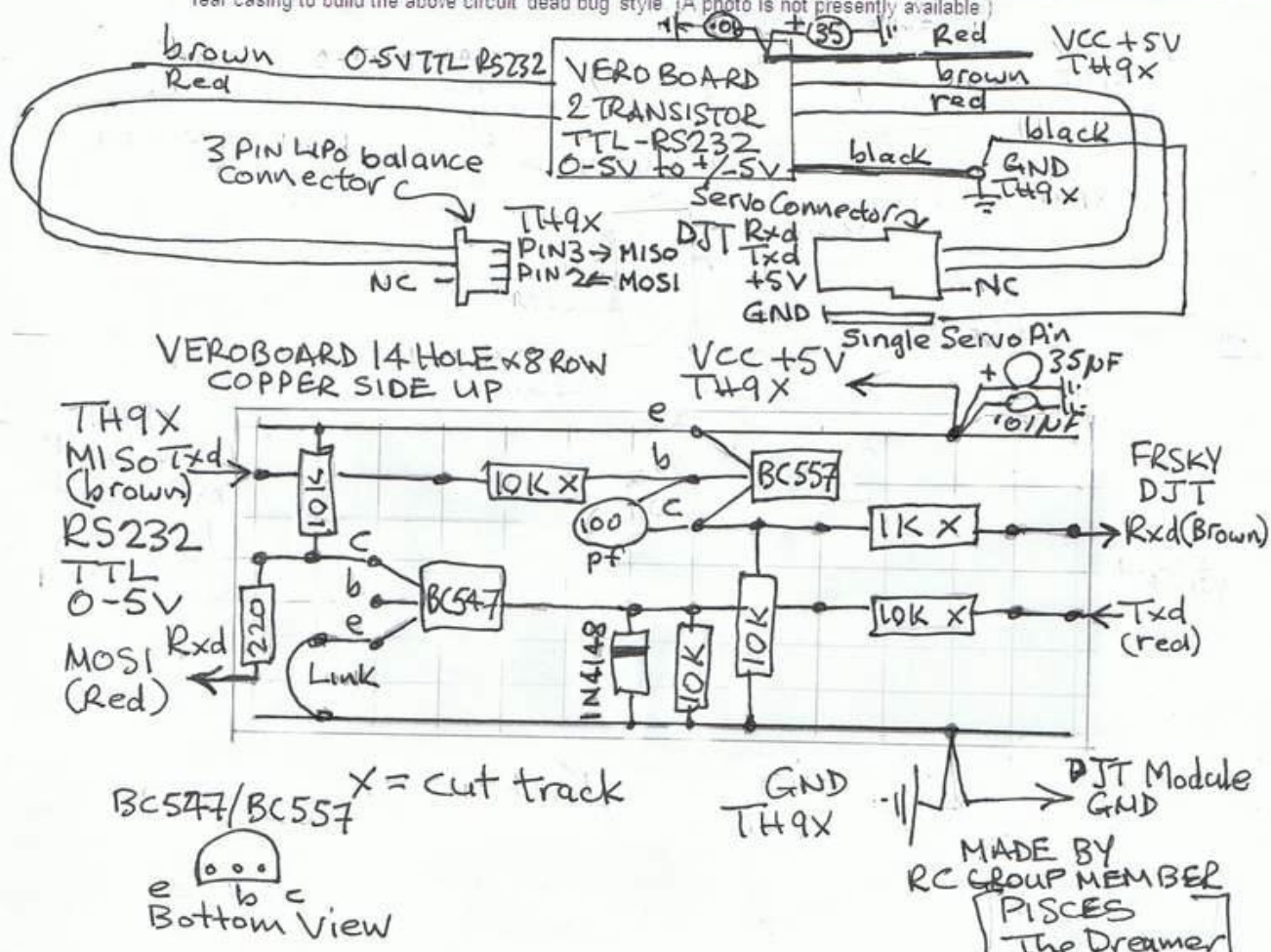
to PIN 233 TXD/MISO

to PIN 234 TXD/MISO

to PIN 235 TXD/MISO

to

For the prototype, Gruvin actually used a dremel too to carve up the mostly unused, legacy PCB mounted inside the '9X rear casing to build the above circuit 'dead bug' style. (A photo is not presently available.)



*Double check your wiring of this circuit. Note that I have added a 220ohm resistor in the collector of Q2. This protects the MOSI line the same as the 200ohm resistors in the THR CUT and AIL DR connections to the Atmega pins.*

*Also i found it necessary to add a 35uf and 0.01uf on the VCC +5V supply to the TTL converter board. A 100pf was soldered between Q2 b and c pins to reduce a spurious short pulse voltage spike.*

*Step 4..Put it all together..*

*Step 5. Load the new firmware ER9x-FRSKY on the Erazz project home page. Link is shown on post 1 of this thread.*

*Unplug the DJT module and the DJT telemetry connector when reading and writing firmware and EEPROM. This is a safety precaution. I have not tried to use eepe with the DJT telemetry plugged in. The transistor TTL-RS232 veroboard has been left connected inside the TH9x when using the eepe to read and write firmware and there was no problem.*

Pisces has further detailed a similar mod using a pre-made MAX232 module sourced from ebay in this post: <http://www.rcgroups.com/forums/showpost.php?p=17550465&postcount=5157> .

## **FrSky Telemetry receiver D8R V2**

FrSky have messed things up a little by hard wiring the voltage sensor in the D8R V2. Measuring the model rx pack is not possible on a model using a BEC without modification to the D8R V2 receiver. The ADC1 input is connected internally inside the receiver to the + supply voltage on the receiver servo connectors common centre pins. The receiver/servo voltage for electric plane using BEC to supply the receiver is usually regulated at +5V. To get access to the ADC1 telemetry input to measure rx pack voltage it is necessary to perform a modification on the receiver as described here: <http://www.rcgroups.com/forums/showpost.php?p=16900284&postcount=2441> .

Pisces has provided an alternative method of modifying the receiver without removing resistors from the D8R ADC1 telemetry input. <http://www.rcgroups.com/forums/showpost.php?p=17390531&postcount=4878>

## **Voltage Sensor**

As far as voltage readings go, the code was written around a V1 receiver without the built-in sensor on A1 and was configured to use a very convenient resistor ratio giving you a full scale reading of 25.4V. This meant that one type of voltage sensor cable could be used for anything up to 6s lipo and give an accurate reading. It also meant that it did not need any scaling in the firmware code. A setting to allow scaling on both the A1 and A2 inputs has now been added to the firmware so other ratios can be used.

## **Making a voltage sensor**

The recommended voltage sensor cable is described here: <http://www.rcgroups.com/forums/showpost.php?p=16889744&postcount=2403> to be connected to the A1 (or A2) input.

Pisces has described an adjustable voltage sensor here. This sensor allows for fine adjustment to get the exact calibration for the screen display rx pack volts.



### Using the firmware:

To access the FrSky screens, from the main er9x display press and hold the down arrow button. From there you can scroll through the screens by using the up and down buttons.

Currently there are 3 screens (see pictures below), the first displays the battery voltage connected to A1 and Rx RSSI. The second has a decimal representation of the analog reading on A2 (0-254) and the Tx RSSI, and the third is a configuration screen for setting the built in alarm points for the Tx module.







### Page 2 / 3:

If you want to use Analog 2 to display a voltage, (useful if you don't want to mod a V2 receiver but want to monitor a battery pack connected to a BEC or ESC) you can toggle a decimal place display on or off by repeatedly pressing the right key (marked '-' on my radio pictured above).



## The config screen:

The operation of the config screen is similar to the FrSky two-way instruction manual and the alarm programming software on the CD that comes with the DJT/D8R combo. But instead of using a PC with the FrSky serial cable and the FrSky software application fdd-lite-rev3-build100627 to program the alarm levels, it is done from the config screens in the TH9x.

In the config screen, pressing menu will allow editing of the fields. To save changes, navigate to the bottom of the config screen until 'save' is highlighted and press menu again. To discard changes, simply long-press exit to return to the er9x main screen or press menu with any other field highlighted to turn off edit mode. Doing this, your changes will still be temporarily shown in the config screen but will not be saved. When saving the values, the Cal values are saved to EEPROM for the current model so they will need to be set individually for each new model that is setup.

## What the display means:

Take the first line, a1 Cal: is referring to the resistive voltage divider that is used for measuring pack voltage. The reading is 1000:130. 130 is what you use with the recommended voltage divider described earlier. For the in-built sensor in the V2 receiver, use 1000:250. These values are derived as follows:

*There are two resistors connected in series that form the voltage divider. On the recommended cable, they are 10k (R1) and 1.5k (R2). You need to calculate  $(R2/(R1+R2)) * 1000$  to get the value, in this case 130. Note: as all resistors have a tolerance and very few exactly match their specified value, you may find that 129 or 131 for instance will actually give a more accurate reading.*

*If you are using some other sensor (eg altitude) and just want to display 1:1 of what the ADC is reading then also use a value of 130. Just ensure that you never feed more than 3.3V into the A1 or A2 pin or you will damage the receiver.*

On the second line, Alarm 1:1: means the first alarm for the A1 analog input (Alarm 1:2 is the second alarm for A1).

The 110 is the alarm point, as my A1 is a voltage sensor cable this means my alarm is set to 11.0V. The < sign means the alarm goes off if the reading is below 110.

The last field is the alarm level. As per FrSky's documentation there are three alarm levels: Red Orange and Yellow. This corresponds to how many beeps the alarm will sound. Red is 3 beeps, Orange is 2 beeps and Yellow is One beep. Select Off to disable the alarm.

## Final Note:

The RCGroups thread here: <http://www.rcgroups.com/forums/showthread.php?t=1266162> has a wealth of information in it and several talented and helpful contributors who will happily answer any questions that they feel they can to do with firmware mods and hardware mods. I encourage anyone attempting to modify their radio with this firmware to head over there and ask for help if you get stuck or need clarification.

I'd also like to acknowledge Pisces' efforts in contributing detailed descriptions and in editing this document.

Phil (Phos)