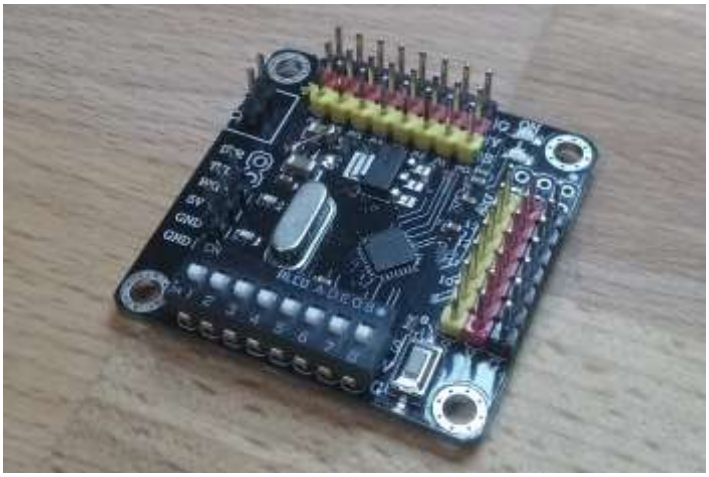
8 Channel Encoder for retro transmitter conversions "F3A V2.1C" Feb 2021



A simple encoder based on Phil Greens (Phil_G on many forums) 7 channel V2 encoder and modified by Mike Kitchen (Mike_K on many forums). This document is also based on and uses much of the words and images of Phil Green's manual with modifications by Mike_K.

The program modifications were aimed at making the encoder more suitable for transmitter conversions that will be used for the popular "F3A" type retro aerobatic model that is becoming very popular again. The rates/expo function has been modified so that you can set the rates/expo independently for ailerons and elevator – it is quite common to want say 30% expo on ailerons, but only 15% on the elevators. Most retro F3A type models were originally designed for a single aileron servo with torque rods, but most are now built with two aileron servo's and the program has been modified to provide a second aileron channel (on CH5), independent servo reverse for both aileron channels and aileron differential. The second aileron channel now makes it an eight channel encoder. The elevon mixing has been retained, but proper v-tail mixing between the rudder and elevator channel is added. Finally, throttle reverse has been added by dip-switch. Obviously there is a cost for these extra features and that is that the single channel features have been removed, sorry Phil!!

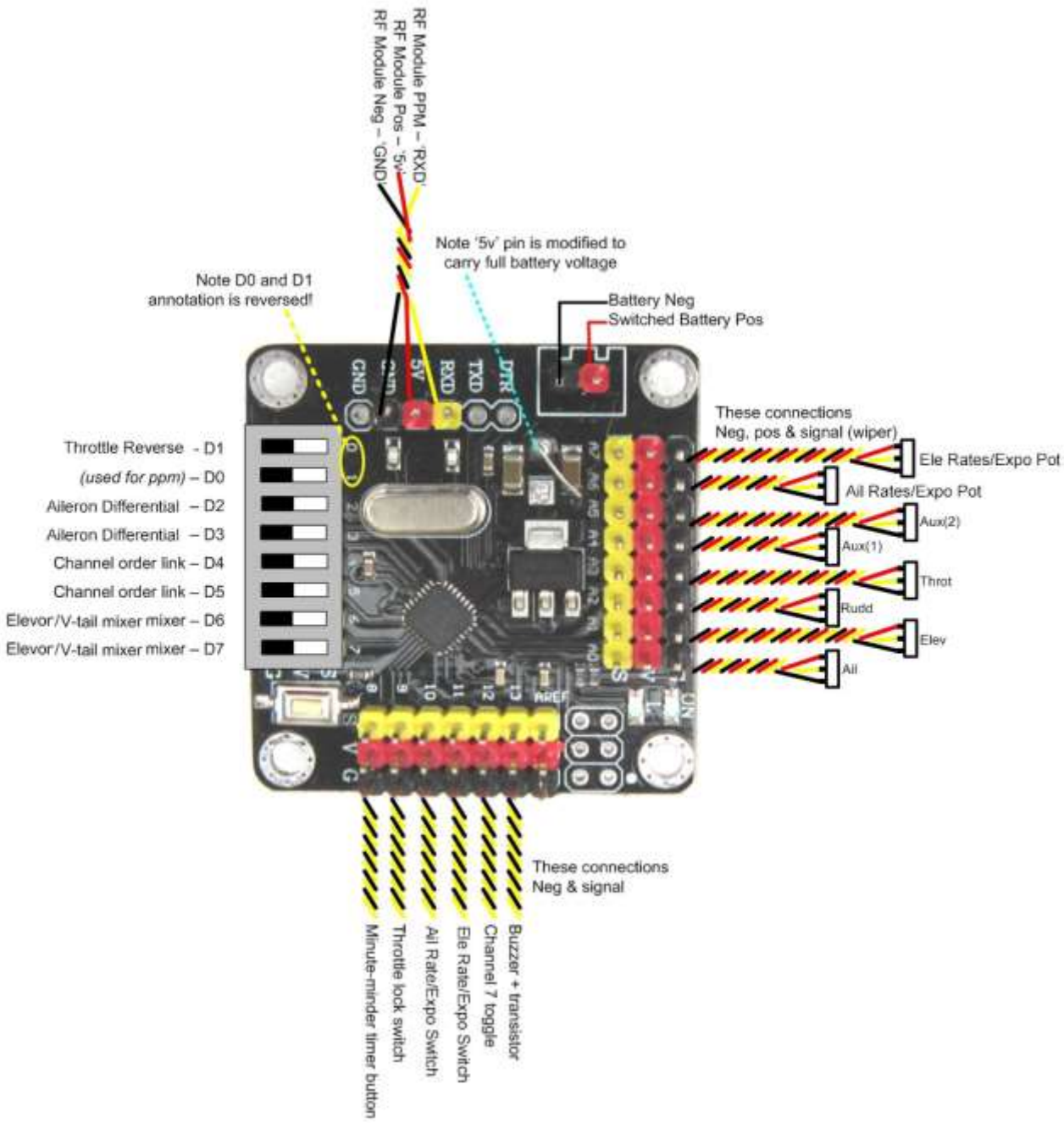
Features:

- 6+1 proportional channels, 2x Aileron, Elevator, Rudder, Throttle and two auxiliary pots plus one switched channel (Gear)
- Aileron differential with the Aileron (Right Hand) on the normal channel and Aileron (Left Hand) on channel 5. Set via dip-switches to give 0%, 20%, 30% or 40% differential (SW3 and SW4 on D2 and D3)
- 'Rates/Expo' on aileron and elevator, with one pot setting rates/expo on aileron and the other elevator rates/expo.
- Servo reversing by holding the stick over on power up for aileron/elevator/rudder
- Independent servo reverse for the two aileron channels, holding the aileron joystick to the right reverses the RH aileron servo, holding to the left, the LH servo.
- Throttle reverse by DIP switch SW1 (on input D1)
- Elevon set to 75:25 or 50:50 ratio or v-tail set to 50:50 ratio SW7 and SW8 (D6 and D7)
- Hardware throttle lock switch (D9)
- Software throttle lock – throttle must be fully closed before it can be opened
- Throttle lock override alarm
- Minute-minder for timing flights, can be extended or cancelled at any time (button on D8)
- Inactivity alarm to remind you that you've forgotten to switch off
- Single-handed range check mode, flick the ch7 toggle three times to invoke (throttle inhibited)
- Three-point self-calibrating sticks, tolerant of inaccurate stick centring, saved to EEPROM
- User selectable primary channel order dipswitch SW5 (D4):
  A(RH).E.T.R.A(LH) with Futaba "polarity" and 300uS pulse separator.
  T.A(RH).E.R.A(LH) with JR/Spektrum "polarity" and 400uS pulse separator.
- User selectable secondary channel order dipswitch SW6 (D5):
  aux1, aux2, switched channel
  switched channel, aux1, aux2

The construction of the Arduino DIY-More Pro-Mini-Strong" board is identical to Phil's original, but it is wired with a few important differences, so care need to be taken if you are familiar with the original. I won't repeat the modifications to the Pro-Mini-Strong as it is identical and Phil has done a very good job in describing it in his original instructions.

Ensure you order the 5V/16MHz Pro Mini version, not the Nano or 3.3V 8MHz versions. The Nano version has a micro-USB connector, but not the FTDI programming pins that are used for convenient connection to the 2.4GHz module and the timings could be altered for the 8MHz version, but this has not been tested.

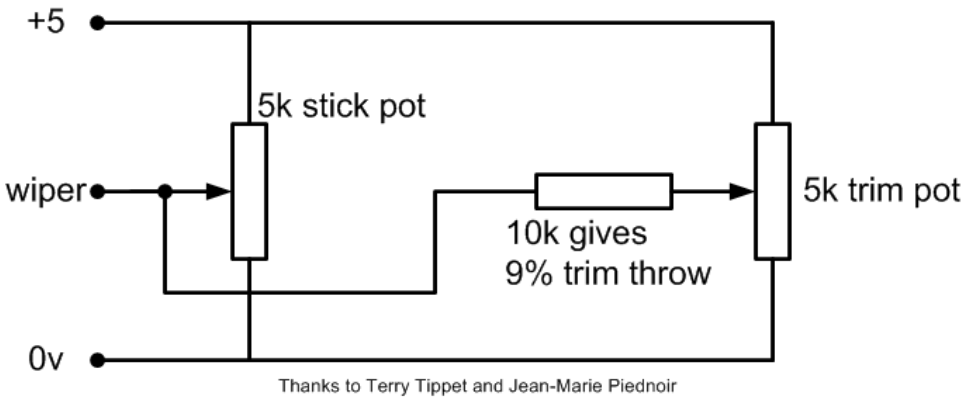
Wiring Up



The wiring is very similar to Phil's original except that the rate/expo switches and the rates/expo potentiometers are wired directly to the Arduino, not with the rates switches wired via the rates potentiometer as previous.

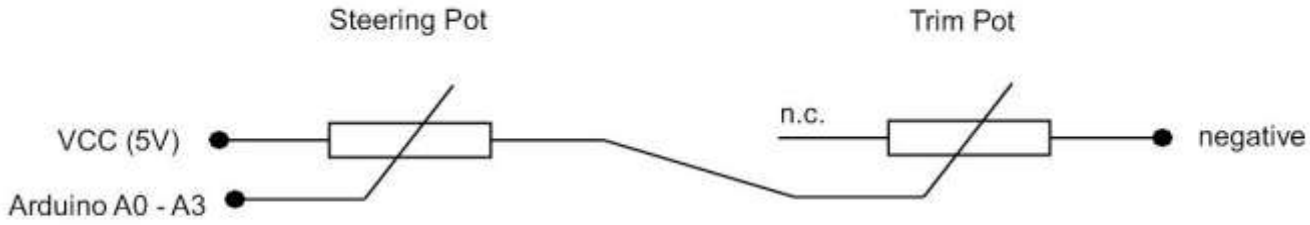
Trims

The encoder is intended for transmitters with mechanical-trim stick units, there are no separate trim inputs, and in fact no spare analogue inputs to the Arduino processor where trims could be connected. For independent electrical-trim stick units, you can resistively couple trimpots (as some manufacturers did) – the fixed resistor determines the trim travel, 10% needs 11.25k, so we use 10k for 9%



Thanks to Terry Tippet and Jean-Marie Piednoir

This alternative circuit also works for many sets, if the potentiometers are the same value and they both turn the potentiometers through the same angle



Alternative Trim Wiring (Thanks to Frank)

Joystick Calibration

The joystick calibration now uses the “minute-minder” push-button as there are no single channel push-buttons with the new encoder. But the sequence is the similar as previous:

- Centre all joysticks trims, hold the minute-minder button down and do not release it until calibration is complete.
- Switch on, and still holding the **minute-minder button** (there is no single channel switches anymore), move each of the sticks into all four corners a few times.
- Move the auxiliary levers or pots fully end to end and leave in the centre.
- Centre all the joysticks including the throttle and the auxiliary channels.
- When done, release the minute-minder button. All calibration values (maximum, minimum and neutral) from each joystick potentiometer are saved to EEPROM (which is memory retentive with the power off).
- Every time you power-up it reads the calibrated values from EEPROM and uses these for the stick maps.
- You can repeat the calibration as often as you like but it only needs to be done once unless you change the stick configurations.
- The 'calibrate' push-button isn’t dedicated - it’s also the minute-minder push button once powered up
- The 'three point' calibration automatically corrects for joysticks which don’t quite centre accurately, making the encoder far more tolerant of old stick units with their original pots.

Rates/Expo Potentiometers

The Rates/Expo feature has been fundamentally modified with the new encoder. The rate/expo switches and the rates/expo potentiometers are now wired directly to the Arduino.

The rates/expo for ailerons and elevator are now independently from each other, with one potentiometer/switch selecting the rates/expo for the ailerons and the other potentiometer/switch selecting the rates/expo for the elevator.

With the Rates/Expo Switch now switches between Rates and Expo. With the Rates/Expo switch OFF you get rates set by the potentiometer “red text”, so fully clockwise you get 100% rates, fully anti-clockwise you get 20% rates and the expo is off (0%).

With the Rates/Expo Switch ON you get the expo set by the potentiometer “blue text”, so fully clockwise you get 0% expo and fully anti-clockwise 80% expo.

Rates/Expo Potentiometer Settings

So expo% = 100% - rates%. Expo has a fairly linear response for roughly the first third of the joystick movement proportional to (100% - expo %), so with the rates/expo switch either on or off you will have the same servo movement for the first third stick movement, but with expo selected, it will continue to 100% movement, whereas the rates will be limit (to the rates%).

So for example (as in the diagram above), if you set the pot to 90% rates it will be 10% expo and the switch selects either 90% rates (and 0% expo) or 10% expo (and 100% movement). But 10% expo has the same effect as 90% rates for the first third of the joystick movement, so both will give the same feel around neutral (but with expo selected you have 100% movement).

This setup may not seem intuitive at first, but once tried, most will find it preferable to the original method where both ailerons and elevator had to have the same rates and expo set.

Most manufacturers do not specify the exact method they use for the exponential function, it seems a closely guarded secret. For the mathematically minded, most manufacturers use the following formula for their expo (exponential):

$$Output = Input^3 * expo + (1 - expo) * Input$$

Where: $-1 =< Input =< 1$: $-1 =< Output =< 1$; $-1 =< Expo =< 1$

So the Expo function is actually a cubic polynomial that equates roughly to an exponential function, but it is the formula that most of the major manufacturers use.

Servo Reversing

The elevator and rudder channels are reversed by holding the joystick fully over when powering up, once powered up the stick can be released and the movement will be reversed and saved to EEPROM. As there are now two ailerons channels, right and left, to reverse the right hand aileron servo, hold the aileron joystick to the right when powering on and similarly to reverse the left hand aileron servo, hold the aileron joystick to the left at power up.

The throttle servo is reversed by the DIP switch D1 (DIP switch 1), see below.

DIP switches

The DIP switches used for various options are much modified with this version. The switches are referred to by the Switch Number and the Arduino input they are connected to. They are only read at power up, so moving them when already powered up will have no effect until the power is cycled off and then back on.

SW1. Throttle Reverse (Arduino input D1)

SW1 (D1)		Throttle Reverse
OFF		Normal
ON		Reversed

DIP switch SW1 Off gives "normal" throttle direction (1mS closed and 2mS open) and with DIP Switch SW1 On it is reversed. Moving DIP-Switch SW1 when powered up will have no effect until the power is turned off and then back on again – the switch is only read at power on for safety.

Having Throttle Reverse does not really pose any additional risk as all modern ESC for electric powered models will not arm until the throttle is in the normal closed position ("1mS output"), so even with the throttle channel reversed, an electric motor will not start inadvertently at power up. Offering throttle reverse can be a big advantage with I/C powered models and it's provided by virtually all manufacturers, so this brings the encoder into line.

If you don't want throttle reverse, then cut the legs off DIP switch SW1 – then it can't be inadvertently operated.

SW2 (D0)

SW2 is not used as it would be connected to D0, but that is used for the PPM output. Cut the legs off SW2 so that if the switch is inadvertently operated it doesn't short out to ground the PPM signal.

SW3 and SW4 Aileron Differential (Arduino inputs D2 and D3)

There are four Aileron Differential options using the dipswitches on SW3 and SW4 (D2 & D3):

SW3 (D2)	SW4 (D3)	Aileron Differential
OFF	OFF	0%
ON	OFF	20%
OFF	ON	30%
ON	ON	40%

Aileron differential limits the movement of down-going aileron compared to the up-going aileron. The benefit of this is that a down-going aileron creates more drag than the up-going aileron, so by limiting the movement of the down-going aileron, you can get similar levels of drag from both and this creates less adverse yaw and allow more axial rolls. The amount of aileron differential required varies between models, but is typically between 20% and 40% and will have to be determined by trial and error.

If it is found that the aileron differential is negative ie the down-going aileron movers further than the up-going aileron, then swap the right hand and left hand aileron servo's. You'll just have to remember they are swapped if you want to reverse the servo direction for one of them.

SW5 and SW6 Channel mapping (Arduino inputs D4 & D5)

The channel mapping is changed from the original. DIP Switch SW5 now selects between Futaba and JR/Spektrum and DIP Switch SW6 selects the order of the auxiliary channels

SW5 (D4)		Channel Order (Primary)
OFF		Futaba A(RH).E.R.T.A(LH)
ON		JR/Spektrum T.A(RH).E.R.A(LH)

SW6 (D5)		Channel Order (Auxiliary)
OFF		Aux1.Gear.Aux2
ON		Gear.Aux1.Aux2

Where A(RH) = Aileron Right Hand, E = Elevator, R = Rudder, T = Throttle, A(LH) = Aileron Left Hand.

When Futaba channel order is selected, it sets the polarity to match the normal Futaba polarity to a module (not to the buddy box connector) and set the “separator” pulse to 300uS. When JR/Spektrum channel order is selected, it sets the polarity to match the normal JR polarity to a module and sets the “separator” pulse to 400uS. These settings have been tested OK with OrangeRx DIY hack modules, FrSky DHT and XHT modules and the Jumper 4-in-1 multi-protocol modules. There is no reason to assume that other modules will also work OK, but no other modules have been tested.

D6 & D7 Elevon and V-Tail Mixing

SW7 (D6)	SW8 (D7)	Elevon and V-Tail
OFF	OFF	Normal
ON	OFF	Elevon (75:25)
OFF	ON	Elevon (50:50)
ON	ON	V-Tail (50:50)

When Elevon mix is selected, it is output on Aileron (RH) and Elevator and is applied after the aileron differential, so ensure it is set normal (SW3 and SW4 OFF). There are two options, 75% Aileron and 25% elevator for flying wings with a sensitive elevator response (SW7 ON and SW8 OFF) and normal 50%:50% Aileron and Elevator (SW7 OFF and SW8 ON).

The original version only offered elevon/flying wing mix and not true v-tail, if you wanted v-tail on a 4 channel model (AERT) then there was no direct way to do it. The modified encoder offers a true v-tail mix where the elevator channel is mixed with the rudder channel and output on the elevator and rudder channel, leaving the aileron channels normal. It is selected with SW7 ON and SW8 ON.

Minute-minder (timer)
This can be ignored if not required, but the switch must be fitted for joystick calibration. The timer is set by holding the button (connected to D8) and counting pips, each pip representing one minute of flying. For a four-minute flight you would hold the button, count four pips, then release it. When the four minutes have elapsed, an alarm sounds. The timer can be extended by holding the D8 button again, or cancelled by briefly tapping the button.

Using fewer channels, or fewer facilities – say a trad 4ch set:
Any unwanted analogue inputs (say for a 4ch set) should be pulled to +5V, 0V or be tied to another analogue input that is used rather than left floating. The easiest is to use a jumper header to link +5V to the input.

For the stick calibration to work all the stick inputs need to be included, even if they're only mimicking another channel.
The toggle could be omitted but as well as losing the channel you would also lose the single-handed range-check.