Helicopters and Open TX Inputs and Flight Modes

Your Presenter: Craig Oakley (HeliFreaks)

Introduction

So, you have an OpenTX radio and an RC helicopters. The basic setup is OK. but now you want to start to explore some of the extra features that inputs and flight modes can give. I will also assume from this that you have the hang of programming the radio either by menu (as the basic setup used) or Companion. I will also assume you are using v2.0 or later of OpenTX (as this is where custom inputs were first established).

This lesson extends on the basic setup detailed in: Creating a Basic Helicopter Model:

- http://open-txu.org/home/special-interests/helicopter/heli-part-1-setup/
- http://open-txu.org/home/special-interests/helicopter/heli-part-2-mixers/
- http://open-txu.org/home/special-interests/helicopter/heli-part-3-curves/

The basic setup did not use flight modes at all, it just used mixes to handle the pitch and throttle curves, nor did the basic setup switch DR & expo on the cyclics or the rudder.

Why use custom inputs?

Custom inputs are an automatic replacement on the first matching criteria.

Custom inputs can replace mixes (and you only have a limited number of mixes).

Custom inputs can be selected by flight modes.

Why use flight modes?

Flight modes allow a unique representation of the control set for different flying states such as:

- Flight modes display on the screen the mode you are in.
- Flight modes have the ability to smoothly transition from one mode to another.
- Inputs and Mixes can be selected based on flight mode (which makes it easier to see what you are doing.
- Flight modes have their own global variables. These variables can be reliant on other flight modes or independent. These variables can also be changed by functions (for example you can make a custom function to change a variable based on the trim adjusters. If your tail gyro is dependent on a variable, then you can adjust a gyro setting in flight with a trim).

There are a number of current debates as to whether to set Throttle Hold as a flight mode. There are good reasons both for and against. I use Throttle Hold as a flight mode on my helicopters, as it shows on the front screen (which I check before connecting battery), and it makes it easy to announce the mode I am changing into. (So I am teaching you how to do what I do).

I will assume for this lesson that you want Throttle Hold as a flight mode (FM0) as it has it's own throttle curve, pitch curve and gyro settings. You may even eventually want different cyclic and tail sensitivities during hold as well when doing auto-rotations.

Important note about flight modes

The first flight mode with a matching criteria is the flight mode selected. If no flight modes have a matching criteria, then the first flight mode is selected.

Let's say you want:

- Throttle Hold to be first flight mode 0 (FM0 - yes they start at 0 - programmer counting).
- Normal to be flight mode 1 (FM1 - selected by SEup)
- Idle Up 1 to be flight mode 2 (FM2 - selected by SEmid)
- Idle Up 2 to be flight mode 3 (FM3 - selected by SEdn)

If you just assign the flight modes 1 to 3 by SEup, SEmid and SEdn, you can never get to throttle hold.

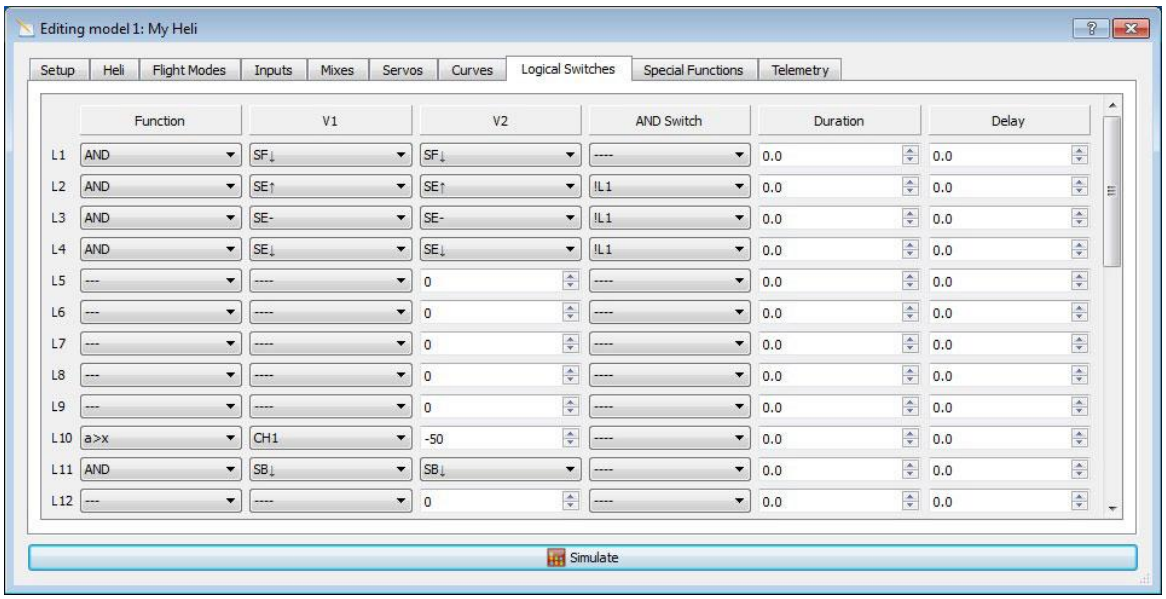
How do we get around this? We use Logical Switches.

Logical switches allow consistent control of model features and TX features (eg. a logical switch representing the states of Gyro heading hold and Gyro rate mode, by changing the assignment of the logical switch from one physical switch to another, the inputs and mixes based on the logical switch remain the same). Logical switches can specify input combinations, so multiple switches can be combined (like two 3 pos switches can be combined to produce nine flight modes). Logical switches can prevent conflict conditions (eg. Allow this condition only if another switch is not active). Logical switches also allow other logic model decisions based on observable behaviour (like throttle over 25%, or timer passed over a defined value).

Programming the Flight Modes

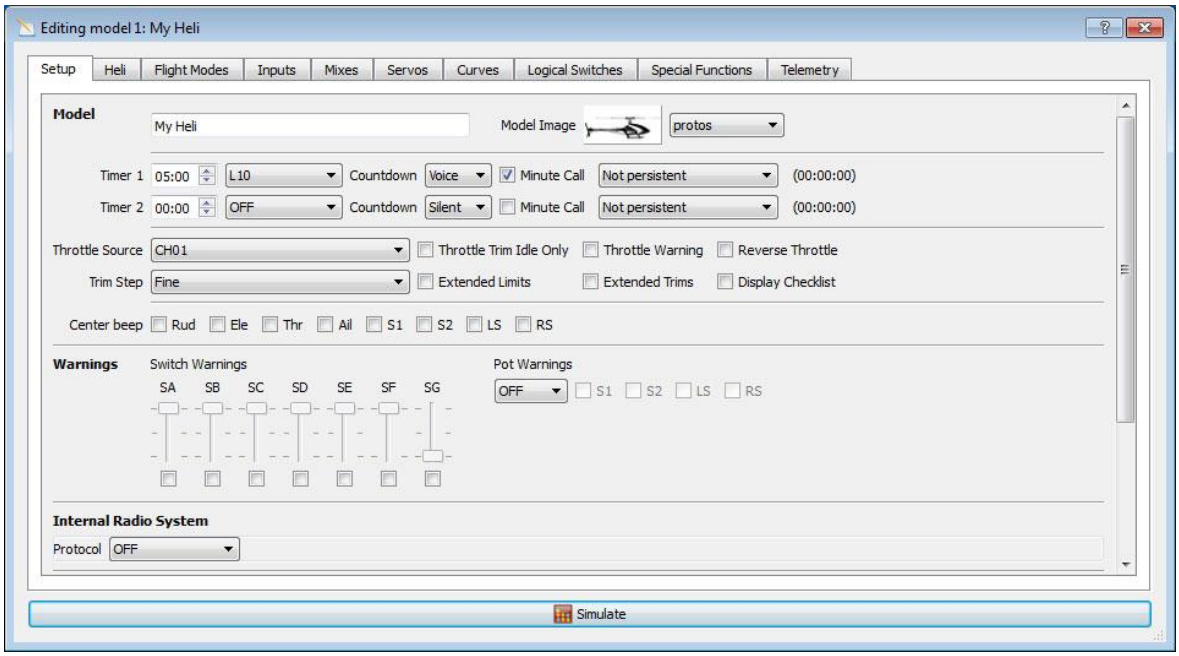
NOTE: I will **assume** you are starting with the basic setup from the previous lessons.

So set:
L1 - AND SF(dn) SF(dn)
L2 - AND SE(up) SE(up) and !L1
L3 - AND SE(mid) SE(mid) and !L1
L4 - AND SE(dn) SE(dn) and !L1
L10 - a>x CH1 -50 {this is what L1 used to be – In companion you can use copy and paste to move}
L11 - AND SB(dn) SB(dn) {this will be for Rate mode gyro}



Logical Switches

{Also alter the timer setup to L10 instead of L1. I am assuming from now on L1 to L9 are reserved for selection of the 9 flight modes}.



Setup

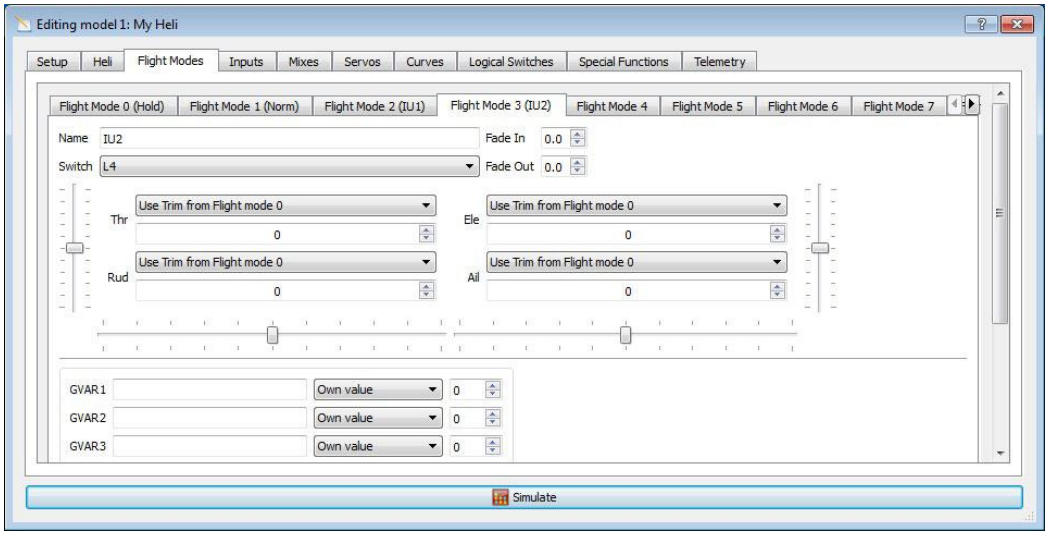
The good thing about using switches is that now you can set switch combinations for modes (I use 9 flight modes in my models). You can set up modes for precision hovering, particular manoeuvres, aggressive aerobatics, whatever you want.

By using the logical switches this way, it can be assured that the required flight mode is selected. The switches can also be used to activate voice alerts of the mode changes.

Next the flight modes themselves need to be set.

FM0 - Name: Hold
FM1 - Name: Norm - Switch: L2
FM2 - Name: IU1 - Switch: L3

FM3 - Name: IU2 - Switch: L4



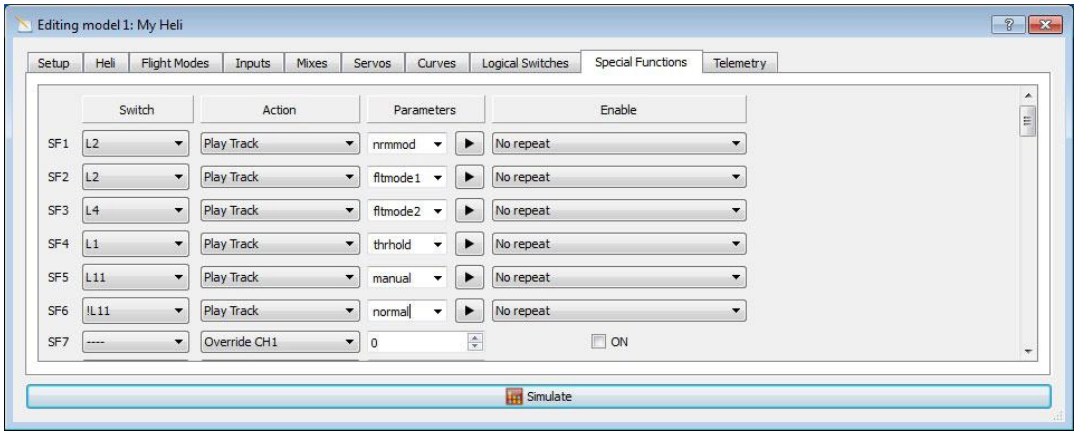
Flight Modes

Then the voice alerts need to be set.

SF1 - L2 - Play Track - nrmmod
SF2 - L3 - Play Track - fltmode1
SF3 - L4 - Play Track - fltmode2
SF4 - L1 - Play Track - thrhold

also two new voice alerts can be added.

SF5 - L11 - Play Track - manual
SF6 - !L11 - Play Track – normal



Special Functions

These are to tell you is heading hold is on or off (normal is heading hold on, manual is rate mode where you as a pilot must control tail drift to point nos into the wind). If you have the Amber sound pack (http://open-txu.org/amber-sound-pack/), then you could use "gyrhh" for "normal" and "gyrrate" for "manual". (I prefer the Amber pack to the default sounds, but you could also produce your own or find another sound pack).

Programming the mixes directly could be done, but in this lesson the custom inputs are programmed next.

Programming the Custom inputs

Custom inputs represent a fundamental shift in the thinking of how the OpenTX transmitter should work. Inputs used to just represent the physical inputs from the sticks (and allowed you to change them into logical stick inputs). In v2.0 inputs came to mean the logical inputs to the mixing functions, this could include switches, basic mixes, anything. Therefore custom inputs really should be used as translations from physical inputs (sticks, switches, buttons, sliders, pots, etc) to logical inputs for the model (aileron, elevator, throttle, rudder, collective, gyro) for a given control set.

{This is classic programmer system communication theory of going from a physical model, to a logical model, then back to a physical model again. In our case: Sticks (physical), inputs (logical), - now we have an idealistic way the model control should work -- Mixes (logical), Servos (physical).

The logical inputs for an RC collective pitch helicopter are:

- Aileron
- Elevator
- Rudder
- Throttle
- Collective
- Gyro (optional)
- Governor (optional)
- any other control function (like self-level, rescue, gear retracts, smoke, etc..)

So at a very minimum, Collective should now be an input.

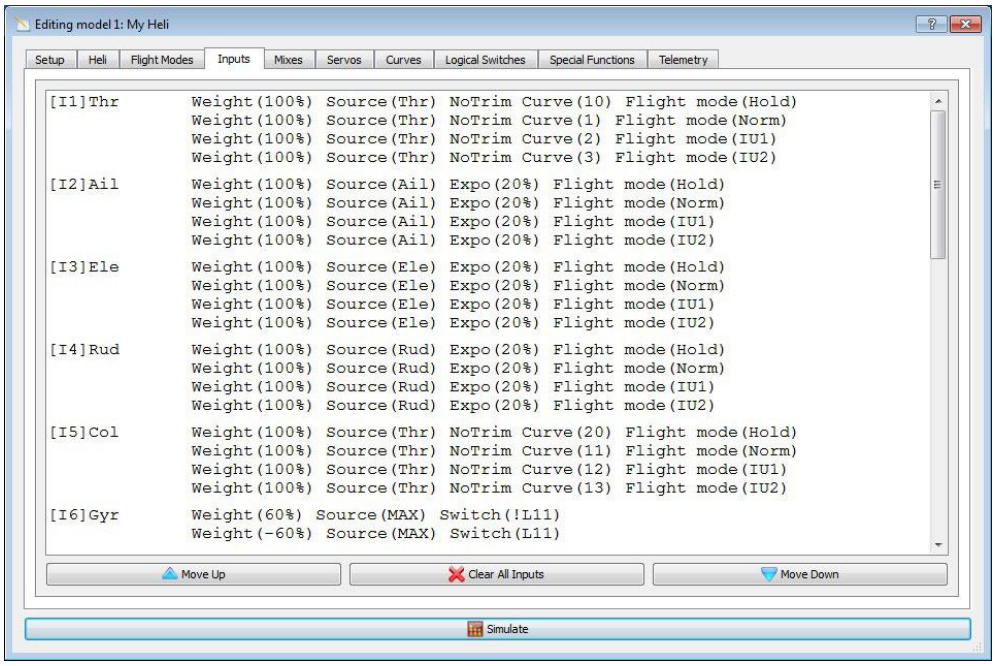
You will need to create an input for Collective. I have also specified in this guide separate settings for Aileron, Elevator and Rudder based on flight mode. It is your option if you want to use this, or just have a single mapping for all flight modes or another method entirely. I personally use a different switch (SG) to change the sensitivity of Aileron, Elevator and Rudder on my helicopters.

What you will end up with is an input set of:

- [I]Ail
    - FM0 (Hold) Ail 100% expo 20%
    - FM1 (Norm) Ail 100% expo 20%
    - FM2 (IU1) Ail 100% expo 20%
    - FM3 (IU2) Ail 100% expo 20%
- [I]Ele
    - FM0 (Hold) Ele 100% expo 20%
    - FM1 (Norm) Ele 100% expo 20%
    - FM2 (IU1) Ele 100% expo 20%
    - FM3 (IU2) Ele 100% expo 20%
- [I]Rud
    - FM0 (Hold) Rud 100% expo 20%
    - FM1 (Norm) Rud 100% expo 20%
    - FM2 (IU1) Rud 100% expo 20%
    - FM3 (IU2) Rud 100% expo 20%

- [I]Thr
    - FM0 (Hold) Thr NoTrim 100% curve 10
    - FM1 (Norm) Thr NoTrim 100% curve 1
    - FM2 (IU1) Thr NoTrim 100% curve 2
    - FM3 (IU2) Thr NoTrim 100% curve 3
- [I]Col
    - FM0 (Hold) Thr NoTrim 100% curve 20
    - FM1 (Norm) Thr NoTrim 100% curve 11
    - FM2 (IU1) Thr NoTrim 100% curve 12
    - FM3 (IU2) Thr NoTrim 100% curve 13
- [I]Gyr
    - FM(all) Max 60% Switch !L11
    - FM(all) Max -60% Switch L11

Note both the throttle and the collective work off the throttle stick input. There are no switches of buttons needed, just the flight modes. You may also notice that NoTrim is specified. This is so trim functions on the throttle stick do not affect input values (this will be especially important if you use a nitro or internal combustion helicopter. But more on that in another lesson).



Inputs

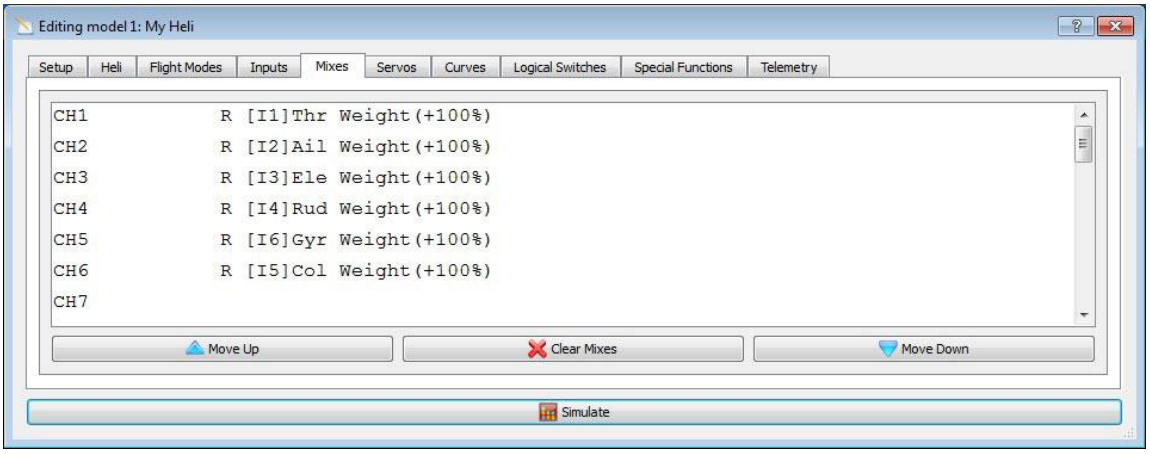
Now that all the inputs are specified time to re-do the mixes.

Programming mixes after the new Custom inputs

The custom inputs greatly simplify the mixes. Remove ALL mixes for channels 1 to 6 and replace with:

CH1 - [I]Thr 100%
CH2 - [I]Ail 100%
CH3 - [I]Ele 100%

CH4 - [I]Rud 100%
CH5 - [I]Gyr 100%
CH6 - [i]Col 100%



Mixes

All the work for the heli is now done in the Custom Inputs.

Why do all this? As you start to develop more complex programming, you will need as much space in the mixes as you can get. Things that can be done in mixes are where you start having interactive overrides of default behaviour (Like buddy box overrides, custom autorotation bailout, rescue facilities, switch failure failsafes, calibration facilities, etc..). Inputs are the default normal of in 99% of the time I want the model to do this from the sticks and switches. Mixes are for model specific mixes (like elevator/aileron mixing in CCPM) or "exception" cases (like throttle modulation during auto bailout in flight to prevent softstart on electrice helis without this built into the ESC - I use this on one of my helis).

Once all the changes are done, you can test in Companion and ensure it all still does as expected before sending to the transmitter.

In future lessons I expect to be building further on this.

So let's see if you have been paying attention:

Pop quiz:

1. What use are flight modes to an RC helicopter?
2. What is the chain commands pass through from sticks to servos?
3. What are custom inputs?
4. Why use logical switches instead of programming physical switches into inputs and mixes?

Answers (none of these should be surprise).

1. Allows unique representation of the control set for different flying states.
2. Sticks, Inputs, Mixes, Servos.
3. Translations from physical inputs (sticks, switches, buttons, sliders, pots, etc) to logical inputs for the model (aileron, elevator, throttle, rudder, collective, gyro) for a given control set.
4. Logical switches allow consistent control of model features and TX features. Logical switches can specify input combinations, so multiple switches can be combined. Logical

switches can prevent conflict conditions. Logical switches also allow other logic model decisions based on observable behaviour.

Resources:

Here are some helpful links for you:

"Newbies guide to the DX6i for RC helicopters" – http://www.helifreak.com/showthread.php?t=580508
"Beginners Taranis programming guide for RC helis" – http://www.helifreak.com/showthread.php?t=598718 (Although this is for another make of transmitter, it covers all the concepts as to how an why a transmitter works in relation to an RC helicopter).
"HeliFreak" – RC helicopter specific resource on the net – http://www.helifreak.com/
"HeliFreak OpenTX forum" – http://www.helifreak.com/forumdisplay.php?f=353
"RC Groups" – Lots of discussion on RC aircraft and helicopters – http://www.rcgroups.com
"FrSky Taranis with OpenTX 2.0 and beyond" – http://www.rcgroups.com/forums/showthread.php?t=2178865
"FRSKY Taranis 'How to' Thread" – http://www.rcgroups.com/forums/showthread.php?t=1914834