Open TX 2.1.8

Introduction

A frequent complaint about **OpenTX** is its lack of documentation, and therefore its difficulty of use. This guide sets out to try to rectify that. It is also the case that **OpenTX** is an evolving program, like most software. This brings the problem that much of what has been written is now partially or wholly out of date.

Several principles were used in the writing of this guide:

It would not try to cover any previous versions of **OpenTX**.

- It would use the **OpenTX Companion** as the primary starting point for programming a model and not the transmitter.
- The use of screen shots to simplify the text needed.
- It has been laid out to follow the menu system in the **OpenTX Companion**.
- It would focus on the Taranis X9D as perhaps the most popular transmitter using **OpenTX**.

The guide is broken down into three sections. This first, **Getting Started** takes the user gently through the actual programming of a model to give a flavour of how the program works. The second part, the **Reference Section** aims to explain briefly what everything does. Finally the last section of **How To's** goes explains in more detail some of the aspects of the program or other facilities that can enhance the program.

I would like to thank all those who have already contributed to developing the knowledge base of **OpenTX** on the internet, and especially those who have unknowingly contributed some part of this documentation.

Finally, please note that this is a guide only. It is down to each individual modeller to ensure for themselves that the way they have set a model up is safe both for the model and anybody nearby.

Martin Phillips

June 2016

Getting Started

Contents

This section of the guide takes a new user gently through the process of setting up a basic model.

There are two ways of programming **OpenTX**. Either the transmitter can be used working through the various screens, or **OpenTX Companion** can be downloaded to a Mac or PC and the transmitter programmed on the computer screen and then transferred to the transmitter using a USB cable. Neither method is mutually exclusive, both can be mixed and matched. Programmed model setups can be transferred back to the computer to be used in **OpenTX Companion**. Which method is the easiest? Using the transmitter there is only one system to learn, but that system is undoubtedly complex with many different pages of menus, some with 30-40 lines of features which can be programmed. The programming is basically done by using the 6 buttons on the transmitter and combinations of short and long presses. Using the computer more of the features can be seen at once, and by using a mouse, editing is far simpler, as is switching between screens.

A very useful feature of **OpenTX Companion** is the simulator which is an integral part of the program. The simulator allows a model setup to be tested and see just what effect there is on the servo outputs. An interesting feature of the simulator is the ability to also show the Taranis screen, and use the mouse to press the appropriate Taranis buttons. Using this, the Taranis menu pages can be quickly related to the **OpenTX Companion** in order to make changes on the transmitter.

To start using **OpenTX Companion**, go to this website **http://www.open-tx.org/**, click on the "Downloads" tab and then the latest version of **OpenTX** version 2.1. A new page opens where the appropriate version of the **Companion** can be selected for your computer. Download the latest 2.1 **Companion** version, and install it. **OpenTX** versions are delivered as sets of major and minor releases. Major releases represent big steps with lots of feature changes and new functionality, which require changes in the way model data is stored and thus breaks compatibility with the previous one. Minor releases within a major one will fix bugs, correct functionality that needs changes, add new functionality that doesn't require breaking compatibility, or remove features that are deemed no longer necessary or useful. There is a simple numbering system for the various software releases. For instance, with version 2.1.7, 2.1 is the latest major release, where important changes have been made to the software, and the "7" following is the latest minor update.

In your user area on your computer create a folder called Taranis. Then connect the Taranis to your computer with the bootloader enabled. – See "How to connect your transmitter to a computer" in part 3, "How to". Once connected 2 new folders can be seen. One will contain just 2 files, EEPROM and FIRMWARE. Do not touch these! However, copy the other folder, with all its subfolders to your newly created Taranis folder on your computer, and rename this folder SD Card. We are now ready to start thinking about the **OpenTX** approach and begin to see how it works.

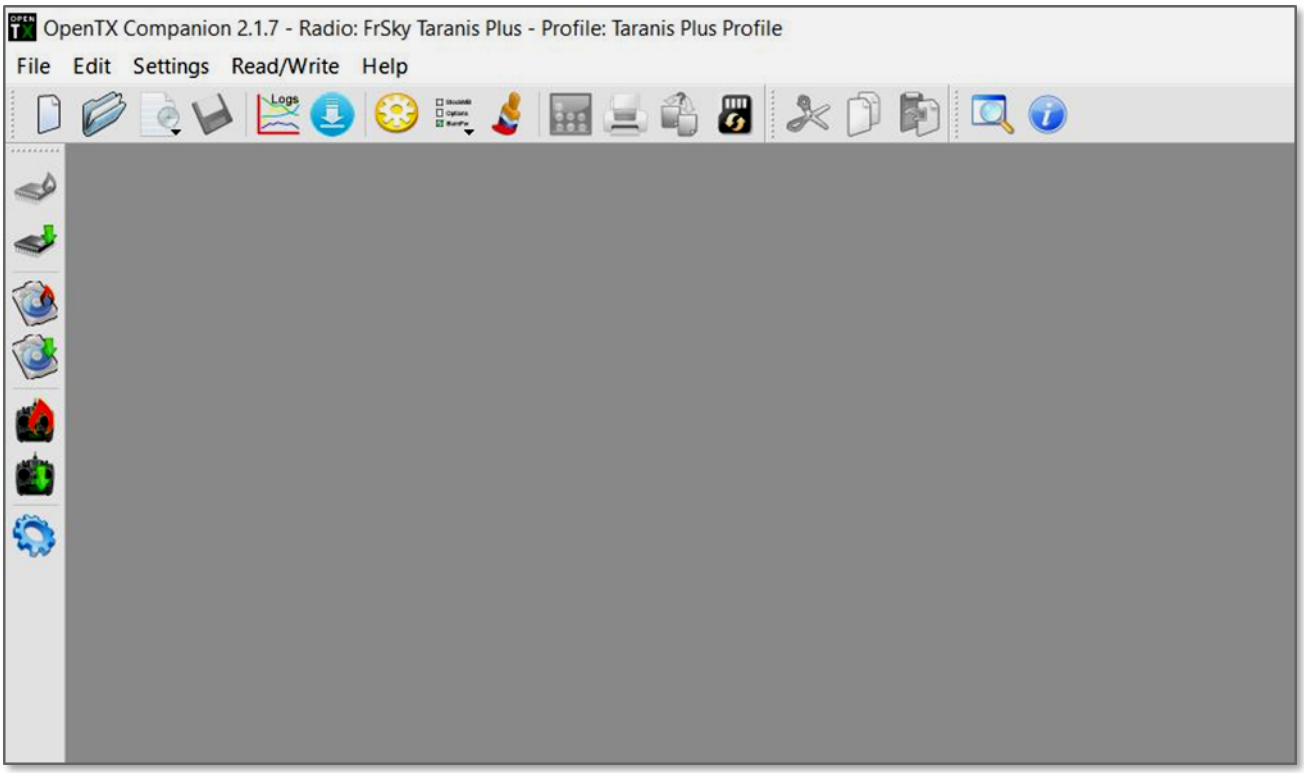
The first thing to learn with **OpenTX** is that it differs from the majority of mainstream radios by its programming philosophy. Owners of Multiplex radios will however feel at home very quickly, as the principles are very similar. Most common radios offer a choice between a limited set of predefined usage scenarios (airplane, glider, helicopter), a number of functions that are commonly used with such models (delta, flaperon, camber, butterfly, etc.), and have fixed assignations (i.e.

sticks always control their respective channels), **OpenTX** offers a blank canvas on which you will build your setup: the **Mixes** window. This approach ensures maximum flexibility because whatever you do you will never have to work around what the radio expects you to do. For some model types, the usual predefined functions can allow setting up a model in seconds (just enable a function), but for others you'll spend hours trying to get around its limitations. With **OpenTX** it might take a little longer at the beginning to set up a seemingly simple model, but a complicated one won't take much more time. As there is no existing function you can just turn on, it will require basic understanding of how your model is supposed to work, and what you want each control surface to do.

There are two golden rules for **OpenTX**:

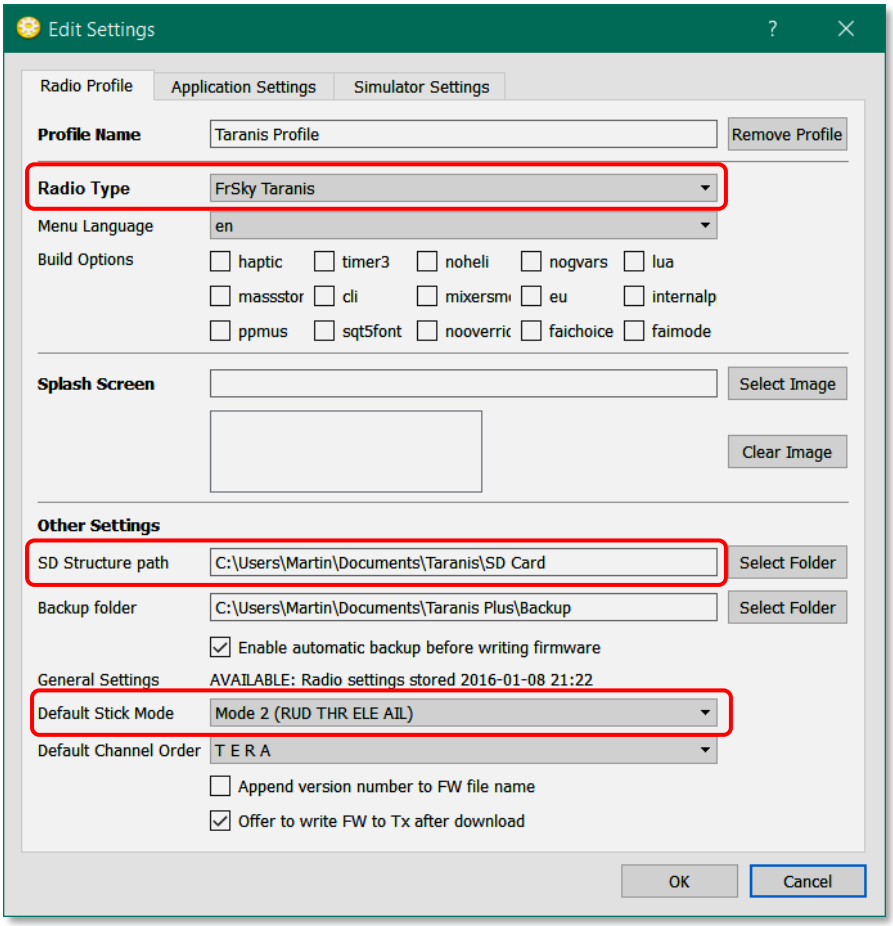
- We have all become used to the setting conventions provided by our preferred transmitter manufacturer, and because this is what we are used to we take this as the norm, and assume this is the way it must be done. **OpenTX** allows one to discard these preconceived views and adopt a much more flexible approach to settings and switch assignments to meet individual requirements.
- There is no single right way of doing anything on **OpenTX**, the same effect can often be achieved in a variety of ways. At times though, some ways are definitely better.

Once **OpenTX** is installed and run you will get a screen which looks like the screenshot on the previous page. Holding the mouse pointer over each icon will show what each icon does. Please note there are different themes available for this screen. Here we will use the “classical” theme as it provides the clearest icons for these screen shots.



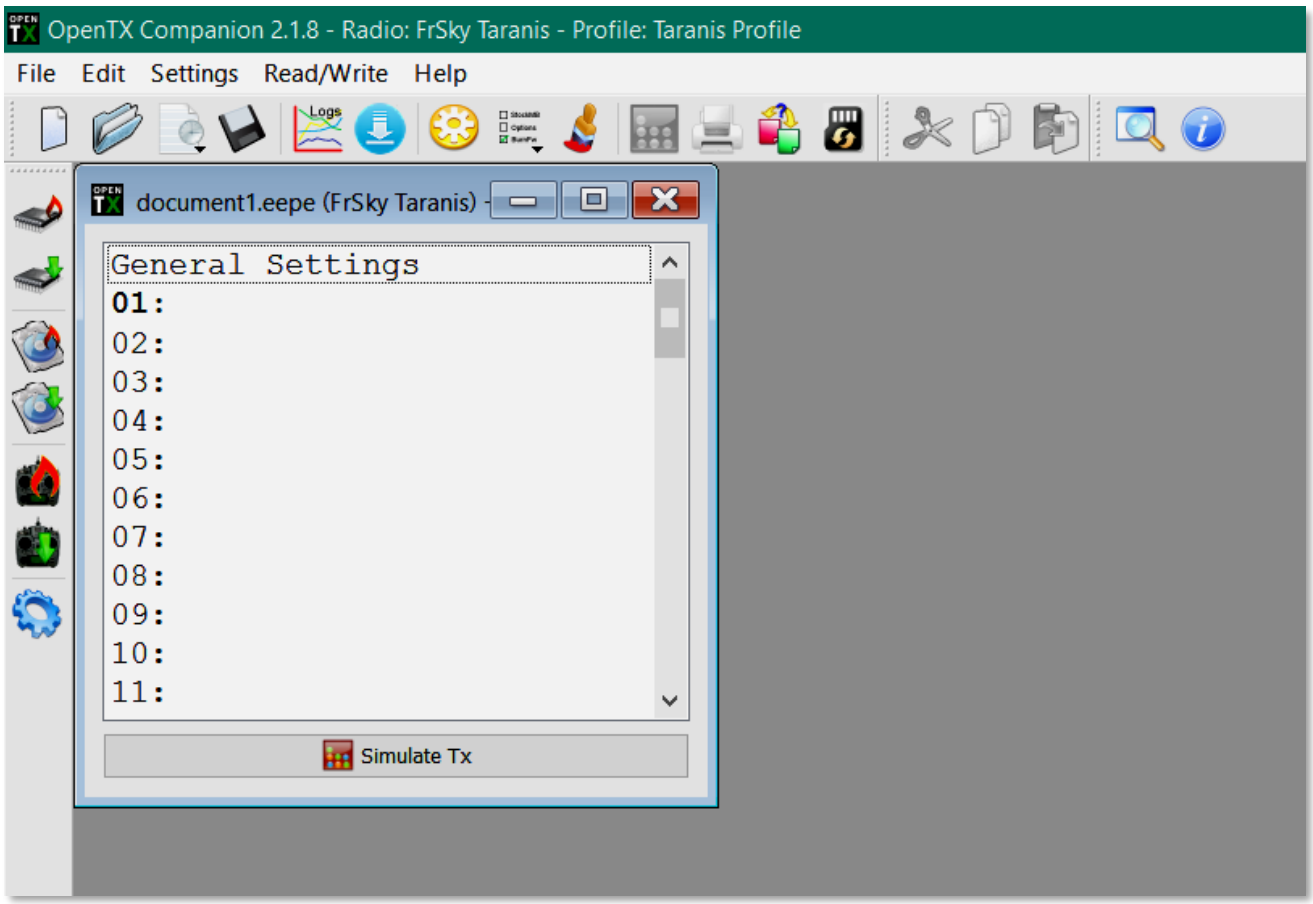
First we need to tell the program which transmitter we will be using. Select the **Settings** icon, the round yellow icon shown left

A new **Edit Settings** window will open up to enable the settings to be edited.

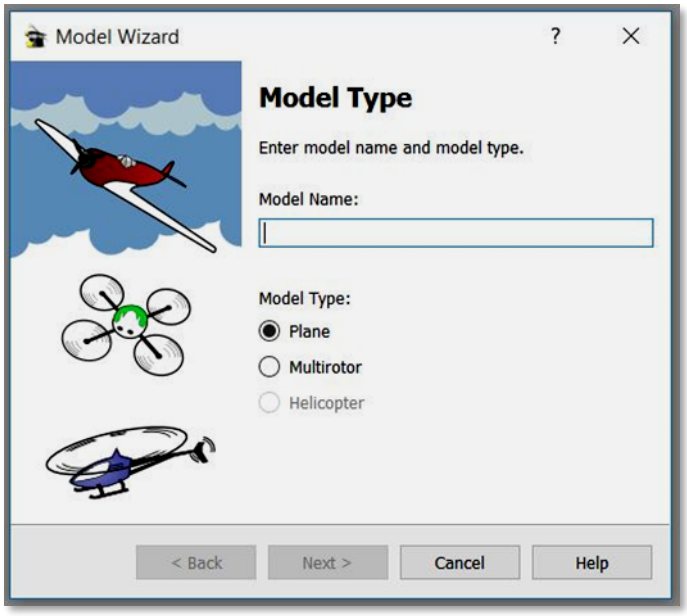


Now select the radio type. In this guide we will be using the Taranis Plus radio, which is currently perhaps the most common. Next you will need to select the folder where you stored the SD card files. If you click on Select Folder, the program will allow you to search for where the folder is located. Notice down at the bottom of the window the default channel order is set. Other radio manufacturers have their own default order of channels. This is usually printed on the receiver. Manufacturers all tend to do different things and have a different order. Here is the first occasion when we begin to see how **OpenTX** is different, and why one needs to leave one's pre-conceived ideas behind. You have the choice of which order. In reality it matters little. However, for those of us that like things neat and tidy, there may be good reason for a particular order. I normally fly planes, and tend to use channel 1 for throttle, and channel 4 for aileron because I almost invariably use individual aileron servos for each aileron and assign channel 5 for the second servo. Well, I think it is logical! Again with the flexibility of **OpenTX**, it makes sense to use individual aileron servos. The latest small servos are not only cheap but very powerful, and with the low cost of 8 channel receivers, we might as well use these channels. Individual aileron servos allow for all sorts of extra functions to be added such as flaperons, or differential, or some fascinating mixes. Everything else on the screen can be ignored for now, and once the OK button has been clicked the window can be closed.

Back on the main **OpenTX** screen, click on **File** and then select **New**, and the models window will open:



For the time being ignore the **General Settings** box at the top which provides some of the basic transmitter settings, and instead double click on model 01 which is currently blank. Incidentally the Taranis and **OpenTX** will store up to 60 models; enough for most of us, and by using the **Companion**, a copy of these can be stored on the computer, just in case! Now another new window will open up. This is the model wizard which provides a simple way of setting up a new model.

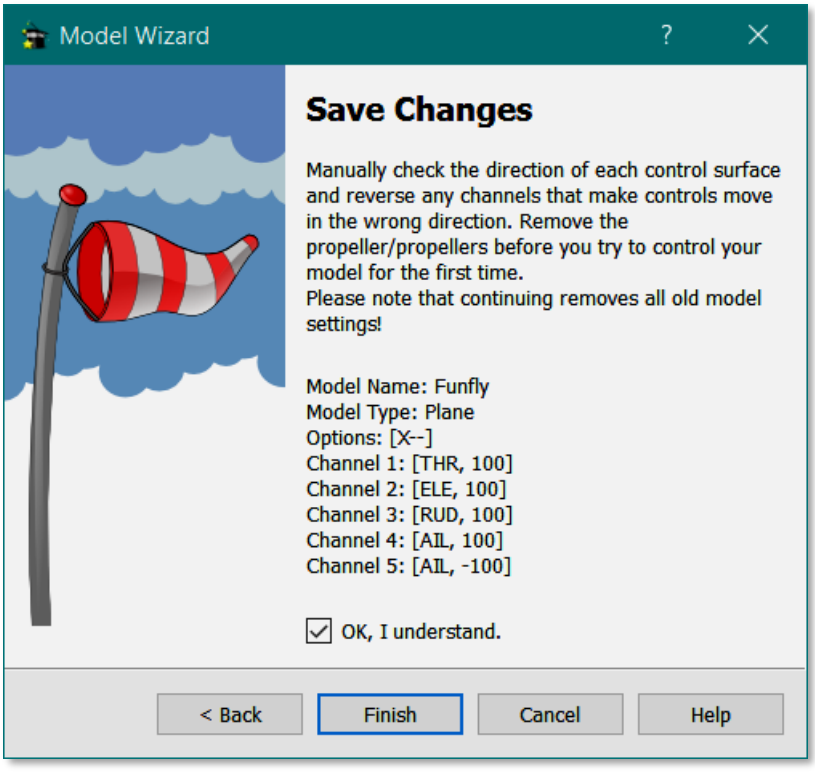


First enter a model name, and then you have the option of selecting either a plane or a multirotor. Sadly, the helicopter is greyed out because at the time of writing nobody has developed a model wizard for a helicopter, though **OpenTX** works well with helicopters. Try selecting **Plane**. Then follow through the

various steps to create your plane. At this stage select a basic wing, select ailerons controlled by two channels, ignore such things as flaps and airbrakes, and go for a standard tail configuration. These can be changed later by right clicking on the model name in the model window and selecting model wizard again. You might think it a bit odd that the program asks you which channel to use for the controls. As mentioned above, conventional radios assign channels automatically. **OpenTX** does not, and the receiver ports are not labelled throttle, rudder etc., but simply 1 to 8. At this stage it might appear an added complication, but in reality it adds more versatility.

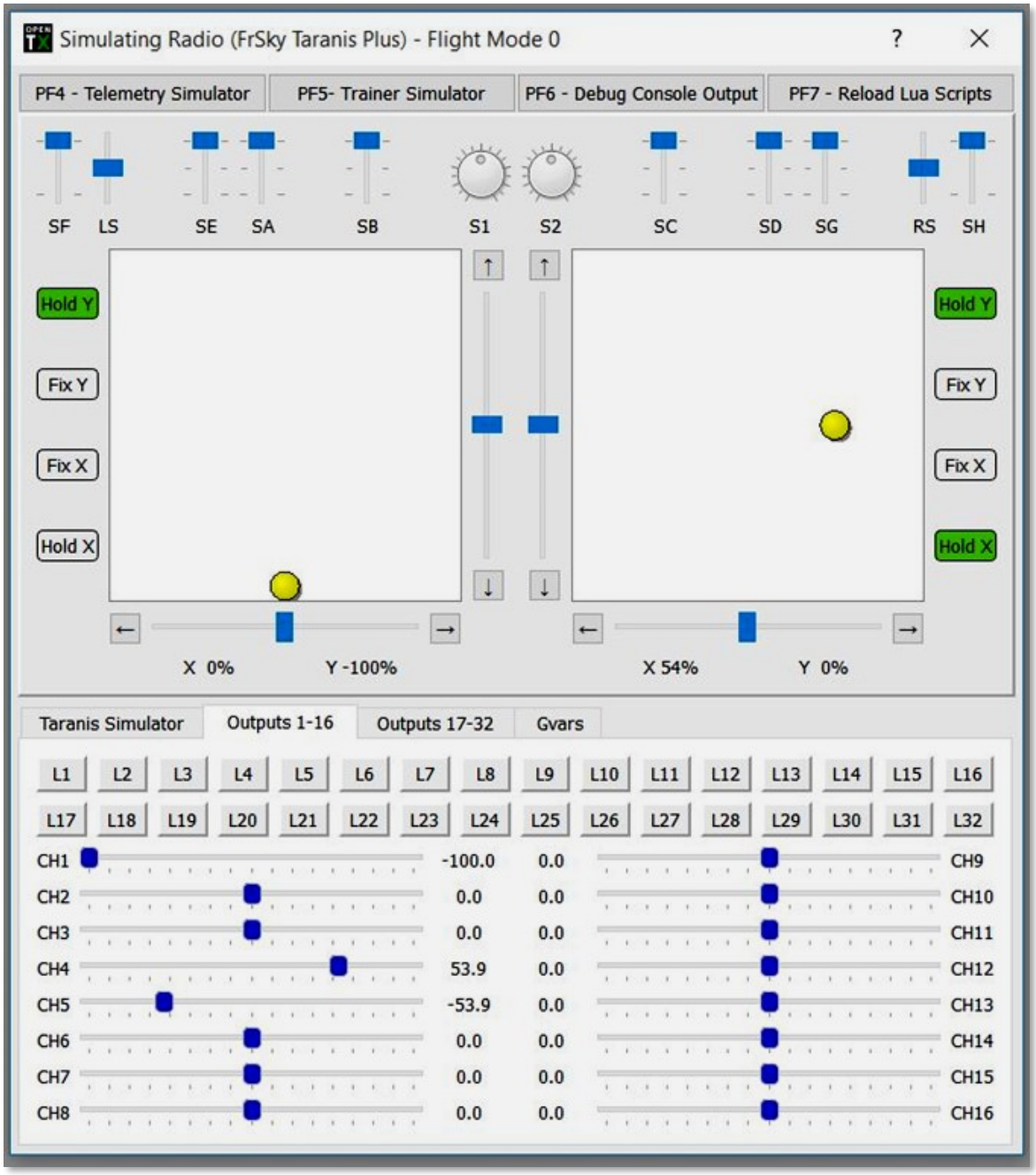
Once you have worked through the model wizard, your new model is programmed, and the program can be transferred to your transmitter via the bootloader. All that is needed then is to connect the servos to the appropriate channels on the receiver, link up a battery or if electric, the ESC, bind the receiver to the transmitter, and check servo directions and throws. However, it is useful to first check the operation of your model setup before transfer. Double click on your first model setup and a new, more complex window opens up. This is the main **OpenTX** editing window – well you must have been thinking there is more to the program that that already described. Clicking through the tabs at the top of this window reveal a plethora of options most of which probably mean very little at this stage. Later we will delve into each of these tabs in more detail showing how to add extra features which will make OpenTX really come alive and show the full potential of the system.

Here you can see the final screen of the model wizard set for a basic plane with two servos controlling the ailerons.

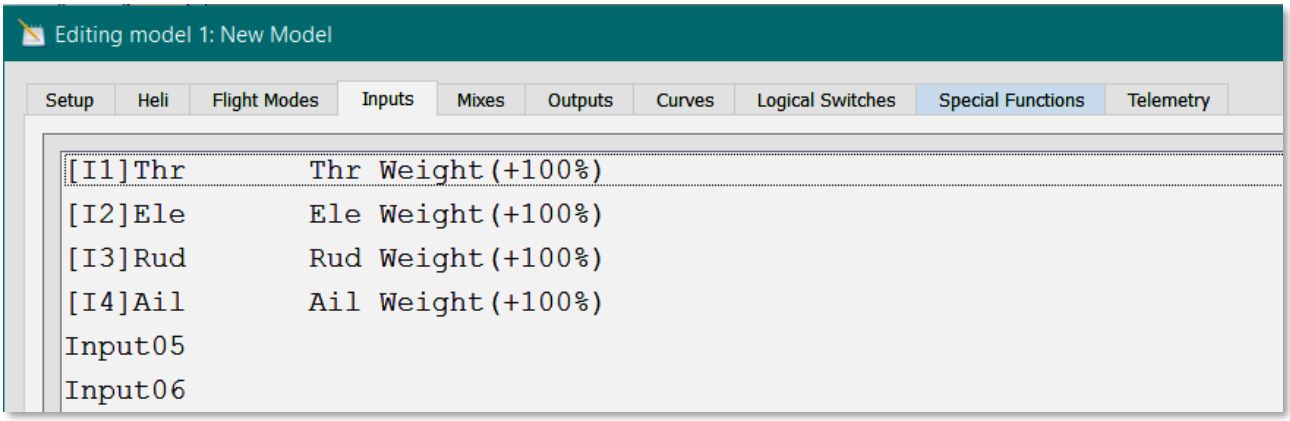


In the meantime, simply click on the **Simulate** box at the bottom of the editing window. The yellow dots are the joysticks which can be moved by clicking the mouse on the yellow dot and then with the mouse button still pressed moving the yellow dot. The small blue squares at the bottom simulate the servo movement. As we assigned two channels to the ailerons, on the simulator you will

see two blue blocks at the bottom move in opposite directions. Later we will be able to see how adding exponential, limits or reversing servos will alter the amount or direction the servos can travel for a given joystick movement.

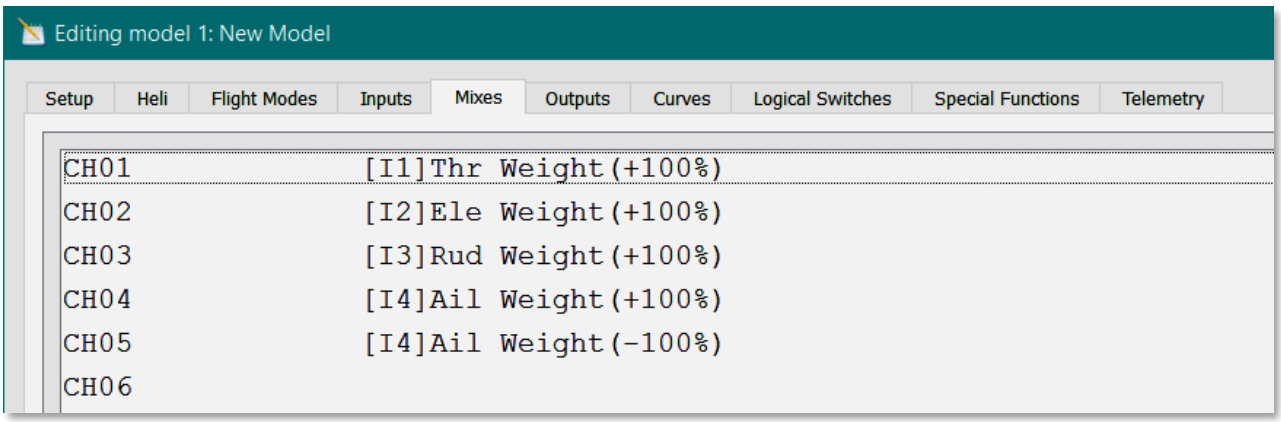


Now we can look in detail how the wizard set our plane up. With the model editing window open, click on the **Inputs** tab:



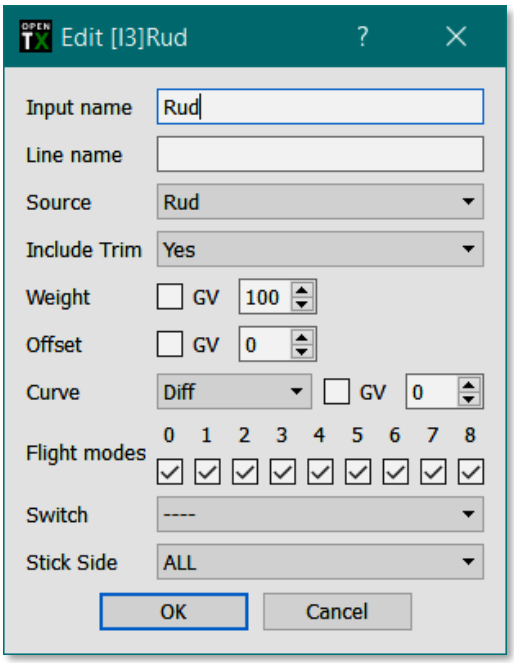
On the left we have 4 inputs. These relate to the transmitter sticks and appear in the order set on the **General Settings** window where we entered the transmitter type. The inputs are labelled **[I1]** to **[I4]** and this screen links them to the four main joystick controls, throttle, elevator, rudder and aileron. Then we have this peculiar feature of "weights". Essentially this is how far the servo will travel for a given maximum stick position, and is key to the whole operation of **OpenTX**. All your transmitter is ever doing is reading the joystick and switch positions, manipulating those values in a defined way and then telling each servo how far to move and in which direction. If the weight was set at 50% then the servo arm would rotate only half as far compared to a weight of 100%. A minus sign in front reverses the servo travel. At this point it should be pointed out that different makes of transmitter will move a servo a different amount for a full stick movement, and some have a slightly different centre point. This screen assigns individual sticks to actual inputs. The keen-eyed will have spotted that there is just one aileron programmed, yet we asked for two. It could have been included here, though it is handled in the next window, the **Mixes** window for reasons you will better understand later. Selecting now the **Mixes** tab, this screen links the inputs to outputs and handles all the mixing of the various functions. It is the programming heart of **OpenTX.**

Channels 1-4 are straightforward but channel 5 now takes **[I4]** again and links it to channel 5 also. This gives us the dual aileron servo function. However, this time it inverts the servo travel using a minus weight -100%. Thus when you use the simulator, you will see the servo travel for channels 4



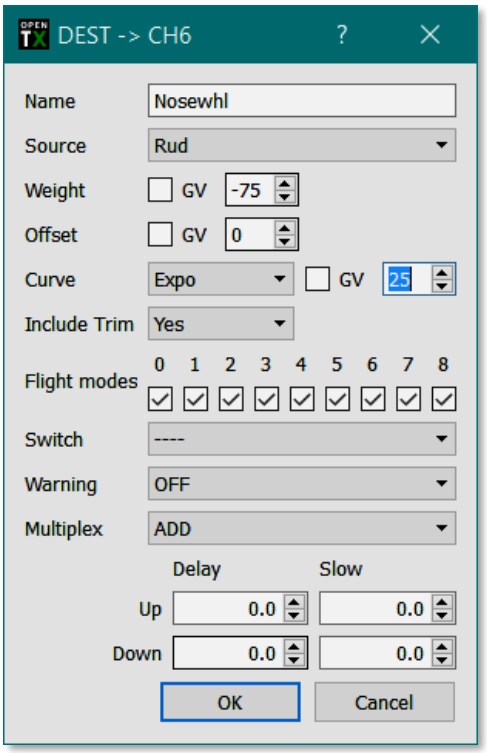
and 5 going in opposite directions. Many using 2 wing servos actually reverse the final movement to the aileron by linking to opposite sides of the servo arm. In this situation one would keep both channel 4 and channel 5 weights positive; in much the same manner as using a servo "Y" lead. OpenTX gives you the flexibility. How do the weights on the **Mixes** window compare to the weights on the **Inputs** window? **OpenTX** combines the two, so if both weights are set at 100%, then the servo travel will be 100%, if either weight is set at 80% then the overall servo travel will be 80%. Similarly, if both weights are set at 80% then the resulting servo travel will be just 64% (80% of 80%). The beauty of **OpenTX Companion** is that you can experiment on screen, and try the simulator to see the effect. To change the weighting value simply double click on the appropriate line and a new window will open up. Try it with the rudder on the **Inputs** window first.

This is a particularly powerful window which allows all sorts of useful functions to be added and changed. Most are not needed at this stage. All we need to do is change the weight line to 80 and close the box. Now check on the simulator what happens to the rudder.



Next move back to the **Mixes** window and similarly open up the rudder window. This window is slightly more complex, but again we simply change the weight to 80 and test on the simulator again. Now the servo should only travel between -64 and +64. Similarly, you can change the -100% weight in channel 5 to +100% and see the effect on the servo travel so both aileron servos move in the same direction.

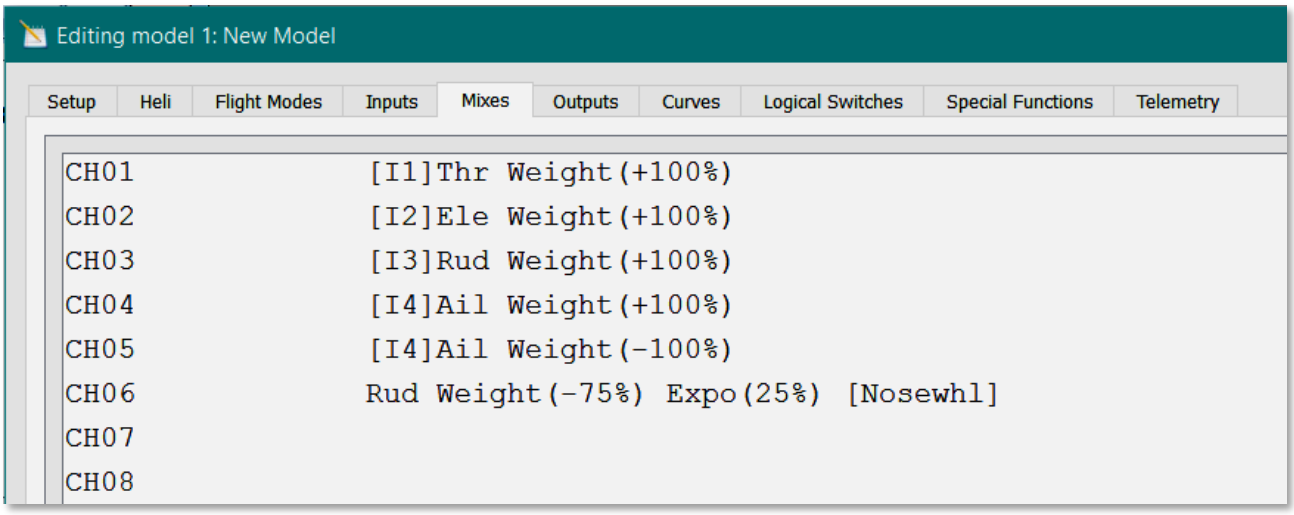
Having got this far, adding new functions is straightforward. With our 8 channel receiver, it is often also better to use a separate servo to drive a nose wheel. This keeps linkages simpler but then there is the complication of mixing back at the transmitter. Let's use channel 6 for a nose wheel servo. Having a separate servo gives us the flexibility to reverse direction if needed and alter the travel. In this example we will set our servo to reduce the travel to 75%, and in the opposite direction, -75%. We can easily do some clever things like add a little expo too. **OpenTX** uses positive expo. Right click on channel 6 to open up a new options window, and select **Edit** (an alternative to double clicking as we did above) and then add this line.



Notice left, **CH06** uses the **Rud** source, not the **[I3]Rud** source which is also available. This means that the source is the actual rudder joystick, not input 4 defined on the **Inputs** window. Why do this? It just means that if you change the weight of the rudder input on the **Inputs** window it will not affect the nosewheel as we saw above. You can test this out using the simulator. As has already been said, there are many ways of doing things on **OpenTX.** Much is down to personal preference.

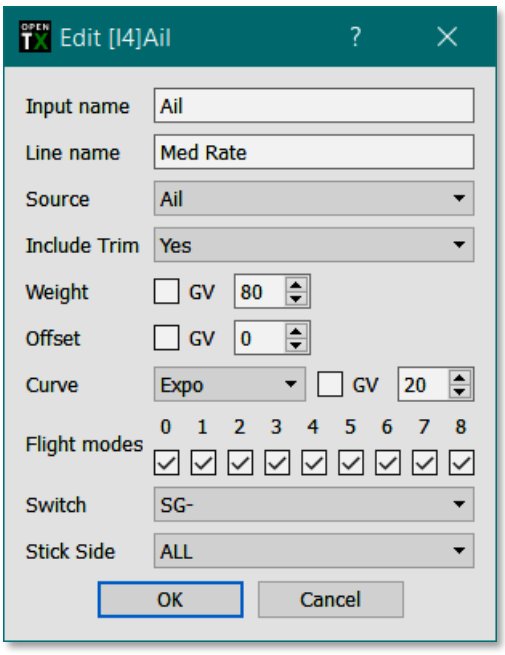
It is useful to also give your function a clear name. After programming several models, and returning to edit this one some months later, you will be glad you made the function of each line very clear.

The **Include Trim** feature is a way of linking or unlinking the rudder trim to this control. It is debatable whether we would need to do this for a nosewheel. The trim is more relevant to rudder operation in the air. However, you have the choice.

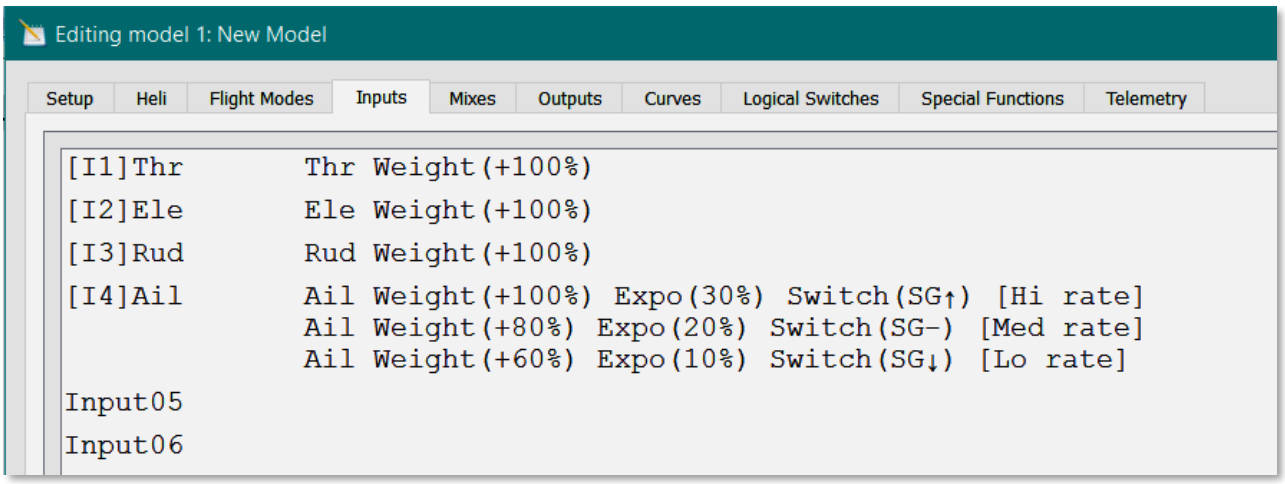


One common requirement is to add aileron and elevator rates, and perhaps add some expo too. On the Taranis there are 8 switches and 4 sliders. One switch has a momentary action, so that when you let go the switch returns back, one switch is simply on/off, but the rest are three position switches. The switches are labelled **SA** to **SH**, and basically any switch can be assigned to any task. To show how it works we will assign switch **G** to the aileron rates, and seeing as we have a three position switch we might as well add triple rates. We could add rates using either the **Mixes** window or the **Inputs** window, but it is easier to do this on the **Inputs** window as we only have to enter everything once. Otherwise on the **Mixes** window we would have to enter the same things for each aileron line. Down at the flying field where one wants to use the transmitter

screen to, say, edit the rates, this becomes far easier and there is less risk of accidentally setting each aileron differently. Right click on the line **[I4] Ail** and select **Add** to add a new line or **Edit** to change the existing one. The flexibility is there to copy an existing line and just change the relevant parts.



There are a couple of points to note here. First, we could use a separate switch to vary the expo if so required. Next, when one clicks on the switch button in the Edit Model window a number of options come up. Some of these will be covered later, otherwise see the reference section. The switches are designated with a symbol after them. **SG↑** means switch **G** is in the up position, **SG–** means switch **G** is in the mid position and **SG↓** means switch **G** is in the down position. An exclamation mark before the switch inverts the operation. This is one to watch, it is easy to accidentally select this and wonder why things don't work as they should! Also **OpenTX** works down the list of commands

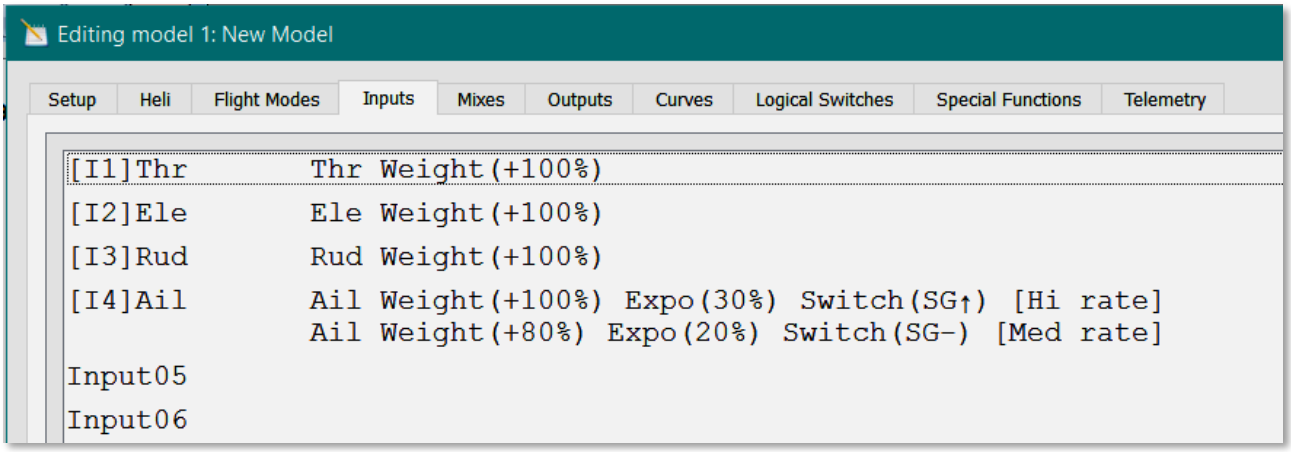


in order. As set out above the rates and expo work as expected on the simulator. You will find the three positions for **SG** at the top of the simulator window with a blue button. As with the joystick it is moved by clicking and holding on the blue button and dragging it down. However, try editing the list so that the first line reads:

Ail Weight (+100%) Expo (30%) [Hi Rate]

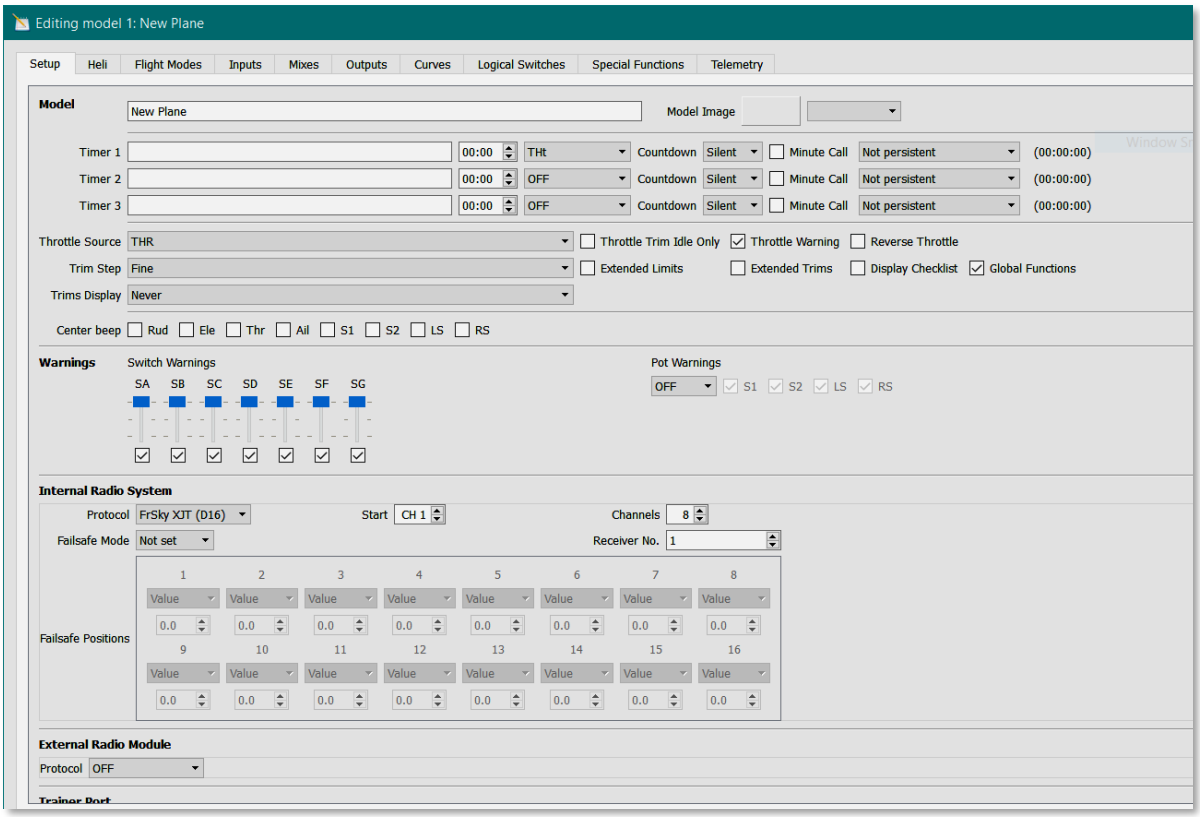
The rates no longer work, because this altered line now states that in all conditions make weight 100% and expo 30%. **OpenTX** thus ignores the other two lines.

Try altering the aileron input lines to now read the following so that you just have 2 rate settings and check the operation using the simulator.

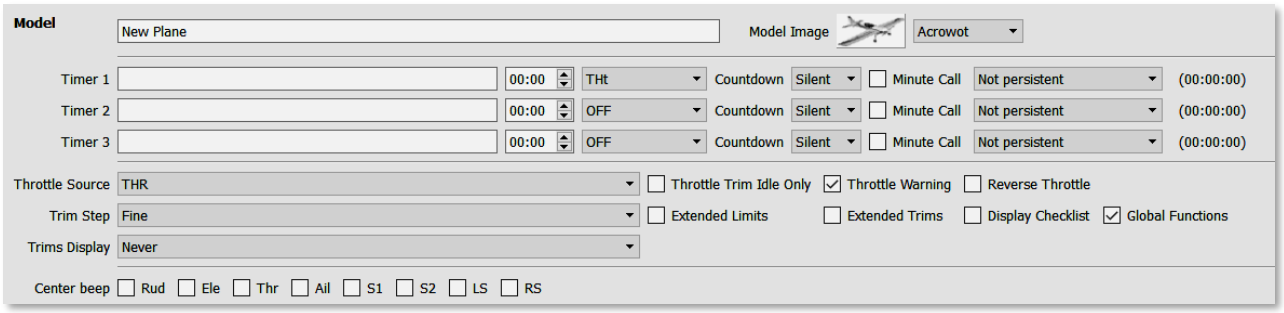


Can you see what has happened? If **SG** is in the down position, both the lines for the aileron are ignored, and effectively there is no aileron control. **OpenTX** does need to be used with some care, and this is why the **Companion** is so useful, it is very easy to check for mistakes using the simulator. However, it quickly becomes second nature to program in a way that avoids these errors.

Hopefully by now, the **OpenTX** approach is beginning to make some sense. Do remember that further detail on some functions can be found in the reference section. Next we will look at how to set up a new basic model using the **OpenTX Companion**, and prepare it for its maiden flight. The example will be a fairly basic model with a motor, a single aileron servo, and a traditional rudder and elevator. We will add a throttle timer. We can set this basic plane up using the **OpenTX Companion** Wizard:



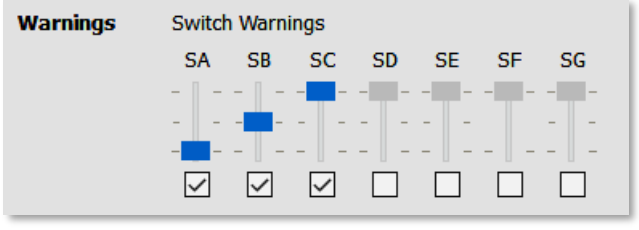
This is the basic **Setup** window created once the Wizard has finished. You need to either right click on the model in the model window and select edit, or double click on the model to open this window up. I've called the plane "New Plane", sorry if it's not very imaginative! One can add a small picture of one's plane the Taranis provides a whole file of them, about 350 in all, with more being added all the time. These pictures, and the sounds, will only show if you have downloaded the contents of the Taranis SD card to your computer.



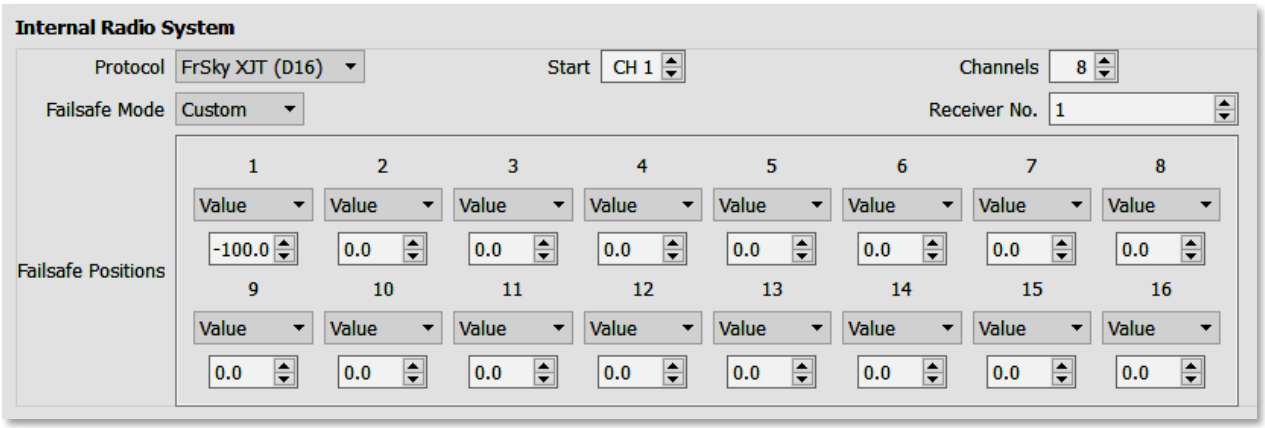
This next part of the screen shows various model settings, including the timer options. The timer options are very flexible, **Timer 1** will have been set to **THt** by the wizard, **Timer 2** and **Timer 3** are off. This means that **Timer 1** will start first time the throttle is opened. Other options include **THs** runs when the throttle is not at idle, **TH%** counts up as a percentage of the full stick range and **ABS** counts up all the time. If value is set to 00.00, the timer will count up from 0, if not, the timer will count down from a preset value entered in minutes and seconds. Countdown will give announcements several times during the last minute, whereas minute call will beep/say the time every full minute. **Persistent**, if ticked, means the value is stored when the radio is powered off and later switched on again. This is useful if one takes a number of short flights without refuelling or changing the battery.

The trim step is a useful function. Here one can set how the trims behave for a new model. **Exponential** is a useful option, together with a very novel feature which is covered later. For electric models, the throttle warning is very useful as it warns if the joystick is not fully down when switching on or changing models. Similarly, the reverse throttle may be of value to Futaba owners, as those sets reverse the throttle, awkward for electric flyers. It is not advisable to use extended limits to be able to move servos more than 100%. The normal servo travel with **OpenTX** is greater than, say, Spektrum gear and is equivalent to their setup with 25% extended trims. Some modern servos do very odd things when moved outside their normal operating range.

The next set of boxes are switch warnings. Basically a warning will be given when the transmitter is switched on if the ticked switches are not in the positions selected. This is very useful for some functions, say like a throttle disable switch. However, it can simply be a nuisance for non-critical or unused switches so they do need to be set correctly.

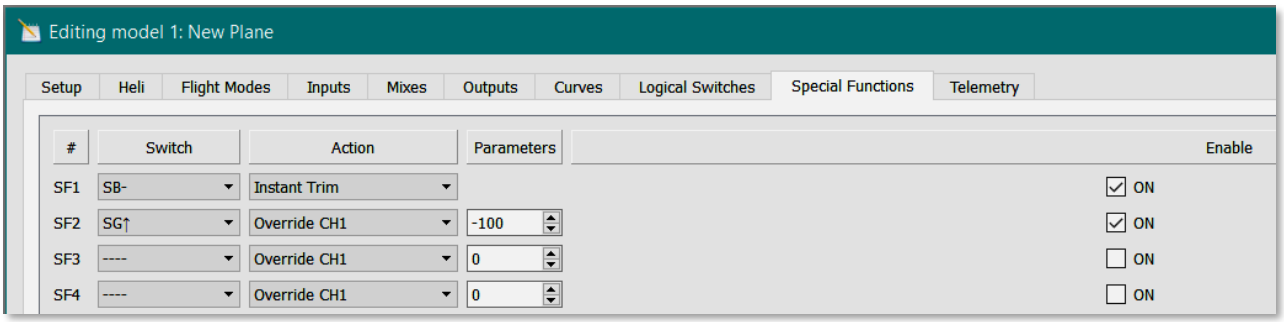


The box below looks complicated. Essentially one first sets it for the type of FrSky receiver being used. With the latest range, X8R, X6R and X4R, the XJT D(16) protocol is used. This relates to the FrSky bit of the firmware which takes the **OpenTX** outputs, and transmits the signal to the plane. Other protocols are available, however for most one needs a specific module to be inserted in the bay at the back for this to work.



It is best for each receiver to have its own number. A very important feature of any radio today is the failsafe mode. Here I have set the throttle to -100 which is off, and the other control surfaces to neutral. Obviously one can set these to one's preferred failsafe setting. If one does not set failsafe, a warning will come up on the transmitter every time that model is selected. This is a really simple and effective way to set the failsafe and can be done before or after one binds a receiver. It should be noted that values set here are not affected by any weighting given later. Failsafe can also be set in the more conventional way by pressing the F/S button on the receiver.

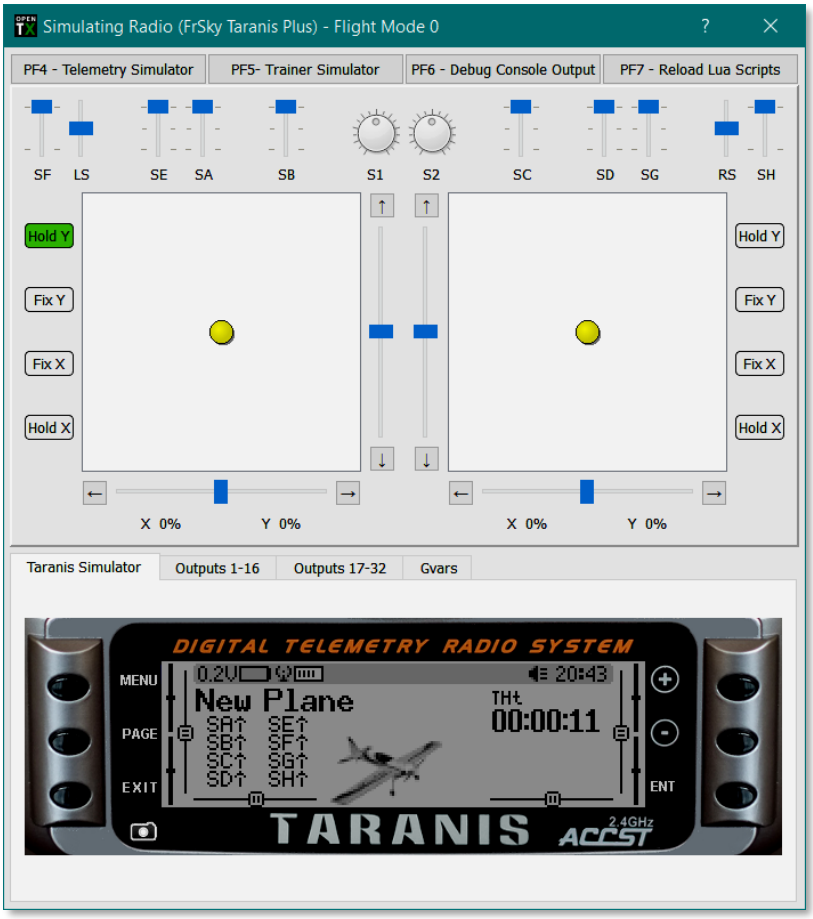
Next we shall move to an entirely new window, the **Special Functions** window. It is here we can create some of the magical effects which really make **OpenTX** so special. As this is a new plane, and we are somewhat worried about the maiden flight, we can either have a fellow flyer standing beside the transmitter ready to move the trims in response to one's frantic appeals for more left aileron trim, or we can program a special function to very simply do the job.



Here **SF1** sets switch **B** (a long switch above the left joystick) to instantly trim the model. Thus on the maiden flight you may well be having to hold quite a lot of aileron and elevator in to get level flight. Keep the model flying level, then flick switch **B** and this will set the trims to the current

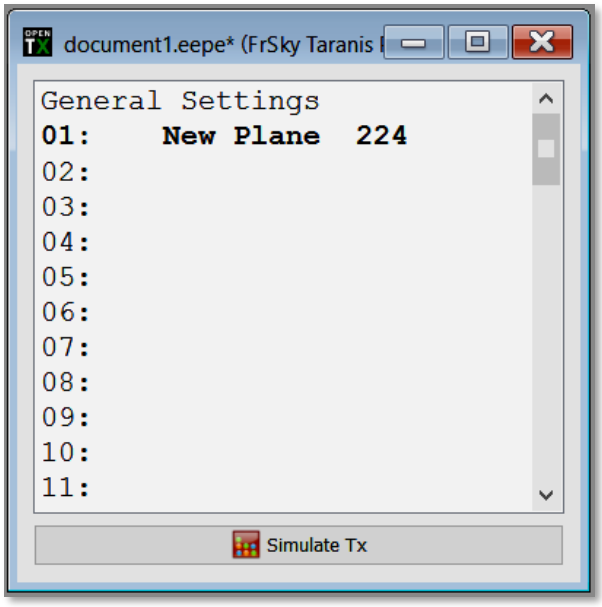
switch positions. You still have the option of using the normal trims. Switch **B** is perhaps the best switch for a mode 2 flyer, however, any unused switch can be used. **SF2** programs switch **G** in the up position to override whatever position channel 1, the throttle joystick, is in and give it a value of -100. In this case the throttle will be completely off. Here we have a simple switch setting to disable the throttle for electric flyers to prevent the motor starting by accidentally knocking the joystick. You can test both of these out using the simulator. To test the instant trim, you will need to click the **Hold X** box and the **Hold Y** box to stop the joystick self-centring while you move switch **G**. later we will look at how we can add a sound warning when the throttle is enabled.

Now go back to the first editing window on the **OpenTX Companion** editing window, **Setup**, and click on the simulator. Part way down, below the joystick representation is a tab labelled Taranis Simulator. Here we see a simulation of the Taranis transmitter screen. There are six buttons, three each side, and clicking on them operates the screen in exactly the same way as pressing the actual buttons on the transmitter. Basically the buttons all operate on a system of short or long presses. **EXIT** leaves that particular element, and when you get to a particular feature you want to edit, press **ENT**, (short for enter). The up and down buttons scroll up and down a menu or change a value. Menus will scroll round, so when you get to the bottom of a menu pressing down again will bring you back to the top. This is particularly useful when you know the item you want is near the bottom of a menu, pressing will take you to the bottom of the menu.

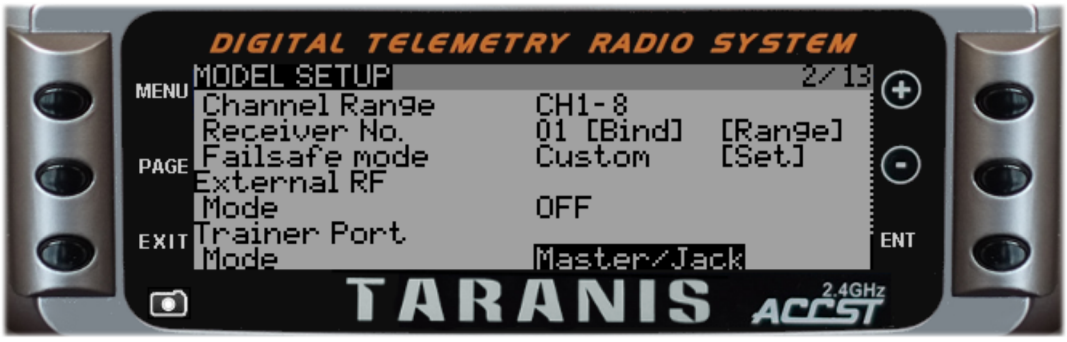


A long press on the **MENU** button takes one to the transmitter setup screens, nine in total. You get the next page of the menu by a short press on the page button. A long press takes you back. This sequence of menus equates to the menus seen in **General Settings**. Press the **EXIT** button to return to the main screen of the simulator.

A short press on the **MENU** button brings up the first of 13 screens which allows one to change models. This is quick and easy to do in the field. Pressing **PAGE** brings up screen 2. The screen number can be seen at the top right hand corner. Screen 2 corresponds to the Setup tab on the **Companion** editor. One only has to scroll through this screen a few times to realise how much easier it is to use the **Companion** than the transmitter for setting up. There are 2 features at the bottom of this screen not found on the **Companion**:

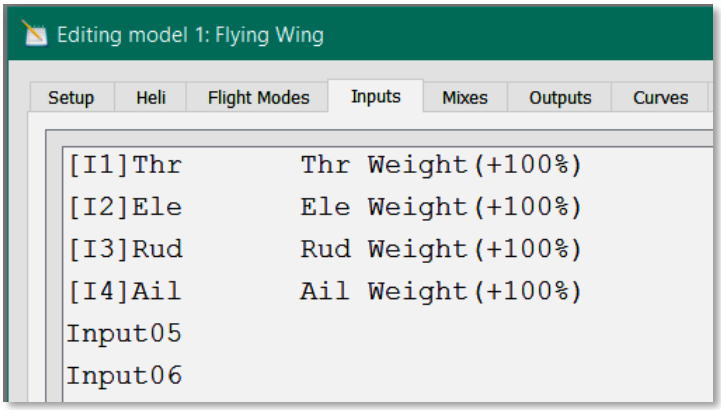


They are **Bind** and **Range**. **Bind** is used to bind a new receiver, and **Range** performs the field range test. The latter is great to use because you actually get to see (and hear) the signal strength seen by the receiver, so no faffing about waggling sticks needed. In reality expect to walk a long way! Incidentally, if you click on the failsafe mode **Set** tab, and then press **ENT**, another screen will pop up showing the current failsafe settings. Remember we set just the throttle to be off earlier.



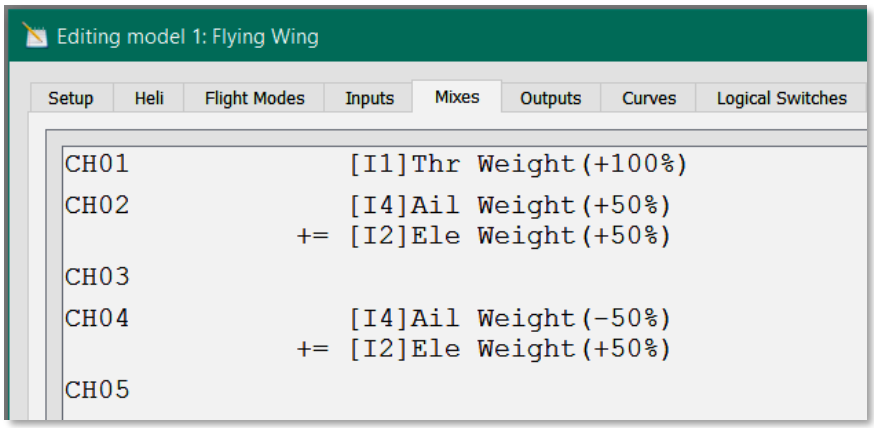
Let's develop what we have learned so far to show that once we have begun to understand how **OpenTX** works, it is very easily adaptable for different models and for advanced features. This is when **OpenTX** really begins to challenge that nagging feeling of, "What's the point, this is too complicated for me, bring back my basic 35Mhz radio".

To start, consider that new flying wing with an electric motor and just two servos to control elevons. Using the wizard, we can quickly program such a system with no rudder. Channel 1 output is set to throttle, and channels 2 and 3 outputs are set to elevons. The **Inputs** window looks like this:



The order of the 4 inputs is how they were identified up on the **Setup** page. Although we are not using the rudder it still shows here. It's not really needed and could be deleted. The **Mixes** window shows how the elevons are created from these inputs.

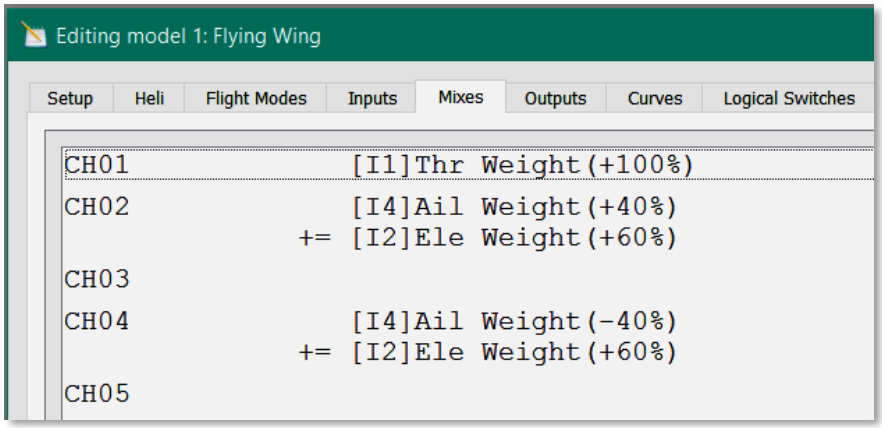
Channel 2 output adds the aileron input and the elevator input both at 50% of their weight, so the maximum combined weight is 100%. Channel 3 though is set with a negative aileron weight to give an opposite movement of the servo. This simple flying wing setup really demonstrates the power of **OpenTX**. Yes, there is a steep learning curve to understand the logic of the system, but then that logic can be applied very easily to much more complex models. In reality when one asked for elevons on any transmitter, this is exactly the mixing that goes on, though we are probably not aware of what is happening.

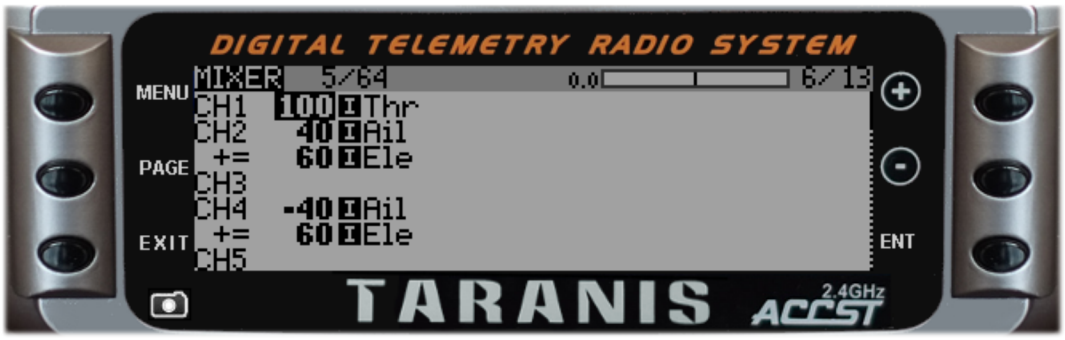


The simulator shows the various servo movements. If, on its test flight our flying wing proved to be a bit tame on the elevator, but very twitchy on ailerons, the weights can be adjusted to give more elevator movement and less aileron movement but keeping the total possible throw the same.

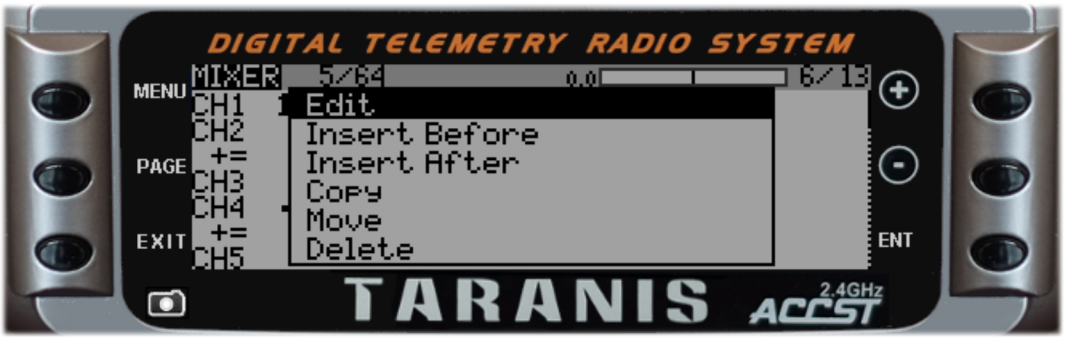
Using the simulator, we can practice altering these weights on the transmitter. Sadly, though it does not change the values in the script, though it will show on the **Outputs** window of the

simulator. It is useful to get this experience to be fully familiar with changing settings down at the field. Try altering the aileron weight to 30%. A short press of the **MENU** key brings up the **Model Selection** menu. Short presses of the **PAGE** button will then move forward through the various pages. Remember that a long press of the **PAGE** button will move back a page. Page 6 brings up the **Mixer** window:

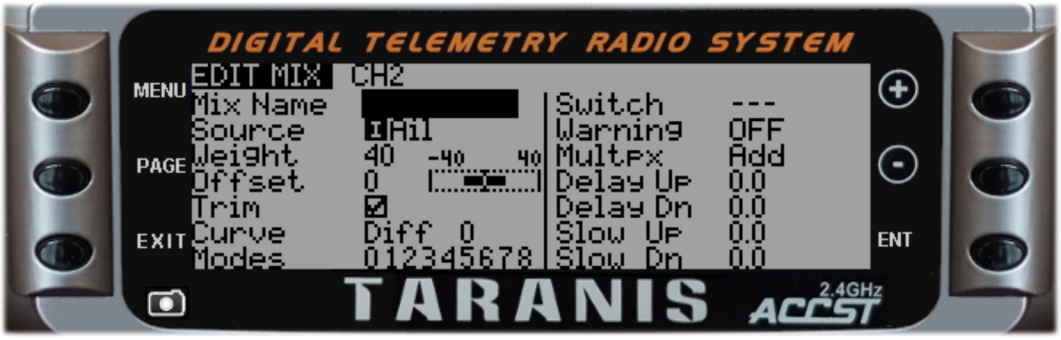




Then use the ⊕ and ⊖ buttons to move down to **CH2**, and after a long press of the **ENT** key, the following menu will appear:

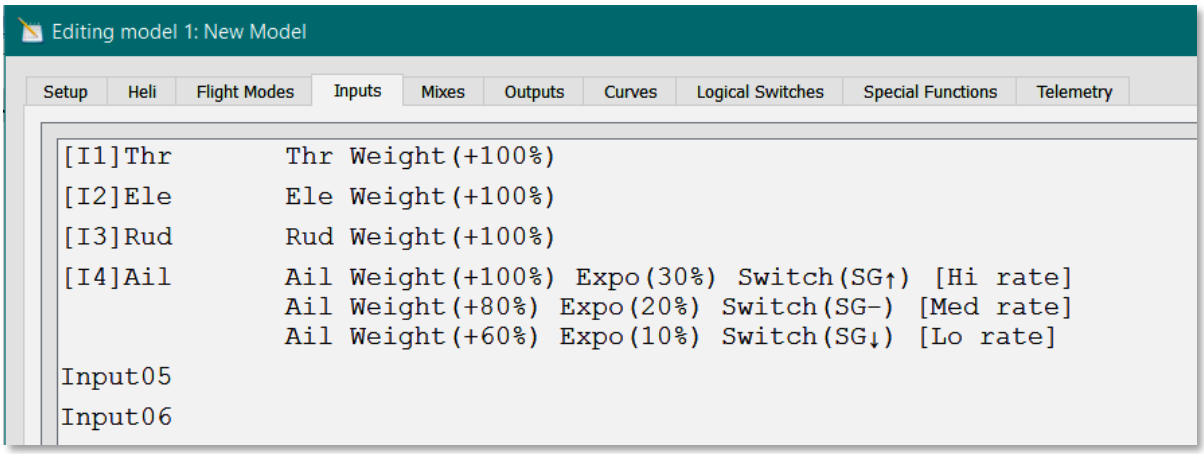


These are the same options as offered on the **Companion**. We want to edit the settings, so with **Edit** highlighted, press **ENT**.

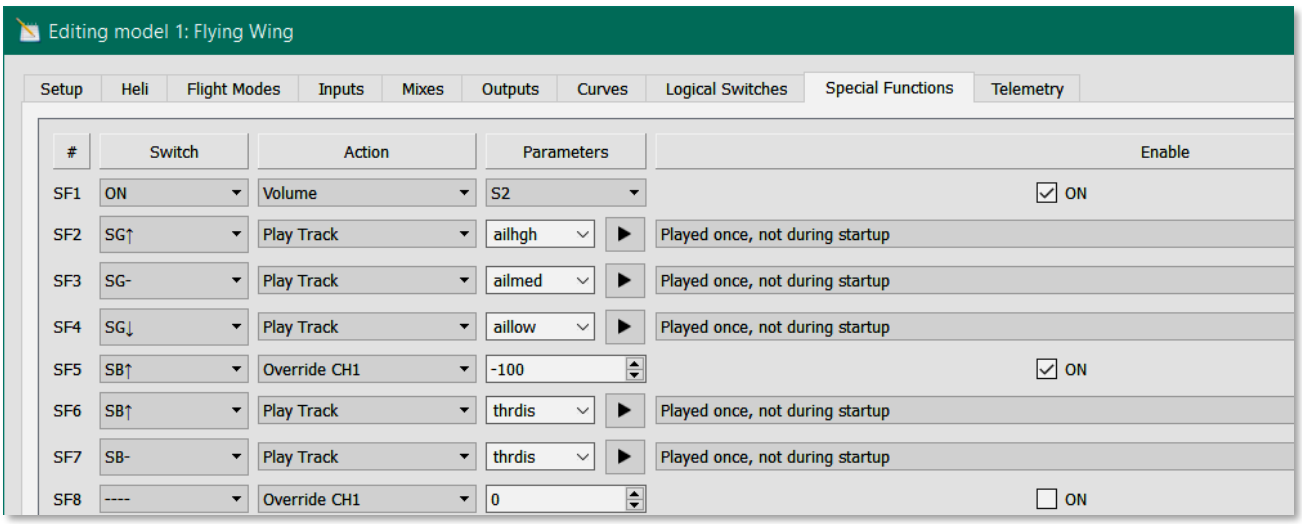


Again we get the same options that are available with the **Companion**. Scrolling down to the weight line with the ⊕ and ⊖ and pressing **Ent** will allow the weight to be changed to 30%. Repeated presses of the **EXIT** button will return to the main screen. By using the outputs screen, the aileron action can be tested. Remember, we only changed one aileron, so both ailerons will move differing amounts.

One aspect of **OpenTX** coupled with the Taranis is its ability to play sounds. While I know sound may annoy some flyers, the information and feedback provided by this system is enormously useful. If there are major grumbles about the sound one can always use a single earpiece plugged into the headphone socket. Earlier we saw how to add triple aileron rates:



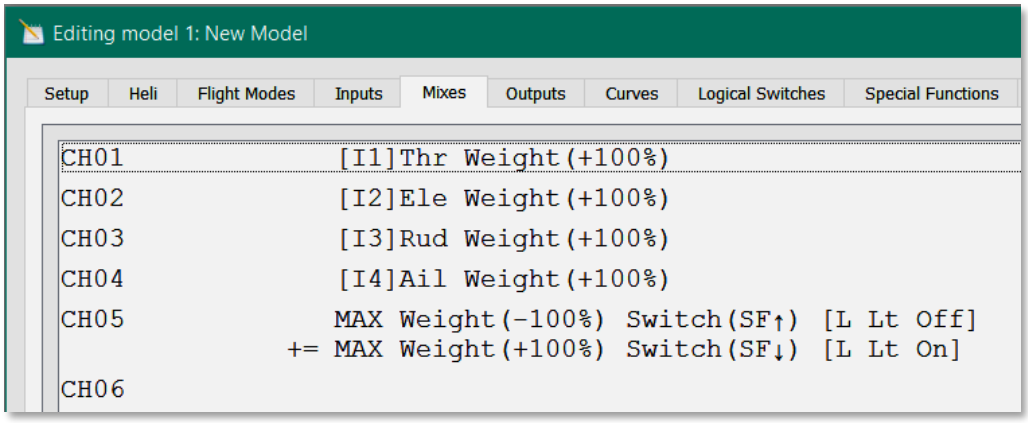
With so many switches on the Taranis, it makes sense to have audible feedback. It is very reassuring to know one has changed the right switch without having to look down at the transmitter. Sound is added on the **Special Functions** screen. We can also program one of the four sliders to act as a volume control.



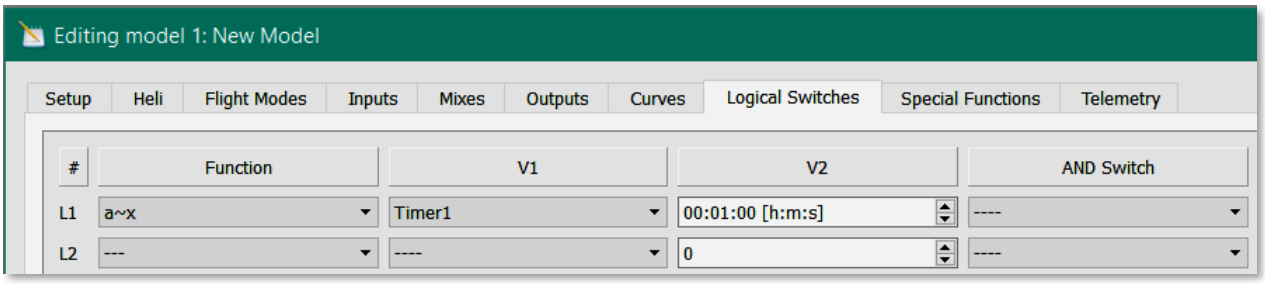
This screen shows a number of special functions assigned. One is allowed up to 64 special functions for each model. **SF1** is programmed to use the rotary knob **S2** as a volume control. **SF2**, **3** and **4** are programmed to play the appropriate voice message for each aileron rates switch position. If the file location of the SD card files on the computer is correctly identified on the **Setup** screen, then the list of available phrases will scroll down in the Parameters box. Each sound can be tested out. On the SD card, in the Sounds subdirectory, there is an Excel and a Jpeg file giving a list of available phrases. More phrases can be added by recording one's own using a microphone or using text-to-speech software (See the "How To" section). Earlier we saw the simple function added to switch **B** to disable the throttle. This is function **SF5**. We can add speech feedback to this switch too with "throttle disabled", or "throttle active". If one would prefer, there are phrases to say "engine disabled" or "engine off" or "engine on".

A further use of the sound feature is linked to the extensive, but very inexpensive telemetry options available. One can get **OpenTX** to automatically read out telemetry values when either certain conditions are met, or when a switch is enabled, however more on this later in the series.

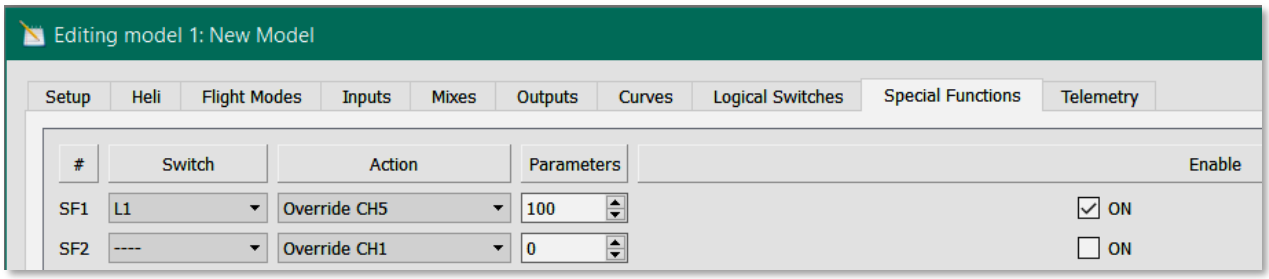
Next, a quick look at two of the more advanced options on **OpenTX**. The first is **Logical Switches**. Essentially these logical switches allow us to develop our own rules, which, if the condition is met, will have some outcome. Just to give a simple example, your favourite scale model has working landing lights which you would like to use every time you come in to land. Easy with **OpenTX**, simply assign a switch, say switch **F**, and link it to an output, say channel 5. On the mixes screen it would look like this:



This is a somewhat inelegant procedure. In the long list of sources is the value **MAX**, which sets the weight to 100%. However, our electronic switch in the plane expects a minimum value to switch off and a maximum value to switch on. Therefore, the two position switch is set to a weight of -100% in the off position, and +100% in the on position. But we're not satisfied with that. We also want the switch to come on automatically just before we land. Here we could tap into the timer we are using on the transmitter and when the countdown has reached one minute put the lights on for us. This is where the logical switches come in.



Add the line above to program logical switch **L1**, and then in the **Special Functions** window add a line to program Channel 5 output. Remember to tick the **ON** box.



Check in the **Setup** window, that a throttle timer has been set using **THt**, and the time set (1 minute 30 seconds is sensible for this demonstration), and open the simulator. As the throttle, by default, opens at mid-point on the simulator, the timer will start straight away. After half a minute, channel 5 will suddenly switch fully on and the **L1** box will show green. There are all sorts of applications for logical switches such as putting delays into bomb doors, or sequencing landing gear.

Going back to the **Logical Switches** screen, there are a list of functions in the scroll down box. The basic ones are **a=x** or **a~x**, meaning in the first case, **a**, or **V1** equals **x** or **V2**, so when **V1** and **V2** are equal, the logical switch will go active. The **~** sign means "roughly equal". It is usually better to use this roughly equal as occasionally **OpenTX** might never see the exact value of **x**, especially when looking at joystick values; computers are picky, precise machines. It should be noted however, that the **~** can be very imprecise when used with such things as time. If you want something to happen at an exact time, then use the **=**. We also have **a<x**, and **a>x**, a is less than **x** and a is greater than **x**. **|a|** means that **a** is converted to a positive value. **a=b** compares two sources directly without having to specify a value. Edge is quite interesting as it can be used to say get a momentary switch to do different things depending on how long the switch is pressed. It takes some experimenting though. The **AND** switch allows another condition to be added, **Duration** is the time the condition remains active, and **Delay** sets the time before the condition starts. Going back to our throttle disable system, we could enhance this and use the logical switches to ensure that the motor was enabled only if both the throttle was off and switch **SB** is on.

Finally, a quick mention of the **Outputs** screen. Generally, this is only useful for a quick way of reversing servos, or perhaps for setting servo travel limits and centres to suit other makes of radio gear if changing a receiver over to the FrSky Taranis system. Even then it is far better to spend some time to check and reset all the control surfaces to suit the new radio, and, if it is an electric plane, then naturally with the propeller removed! See the reference section for a more in-depth understanding of the **Outputs** screen.