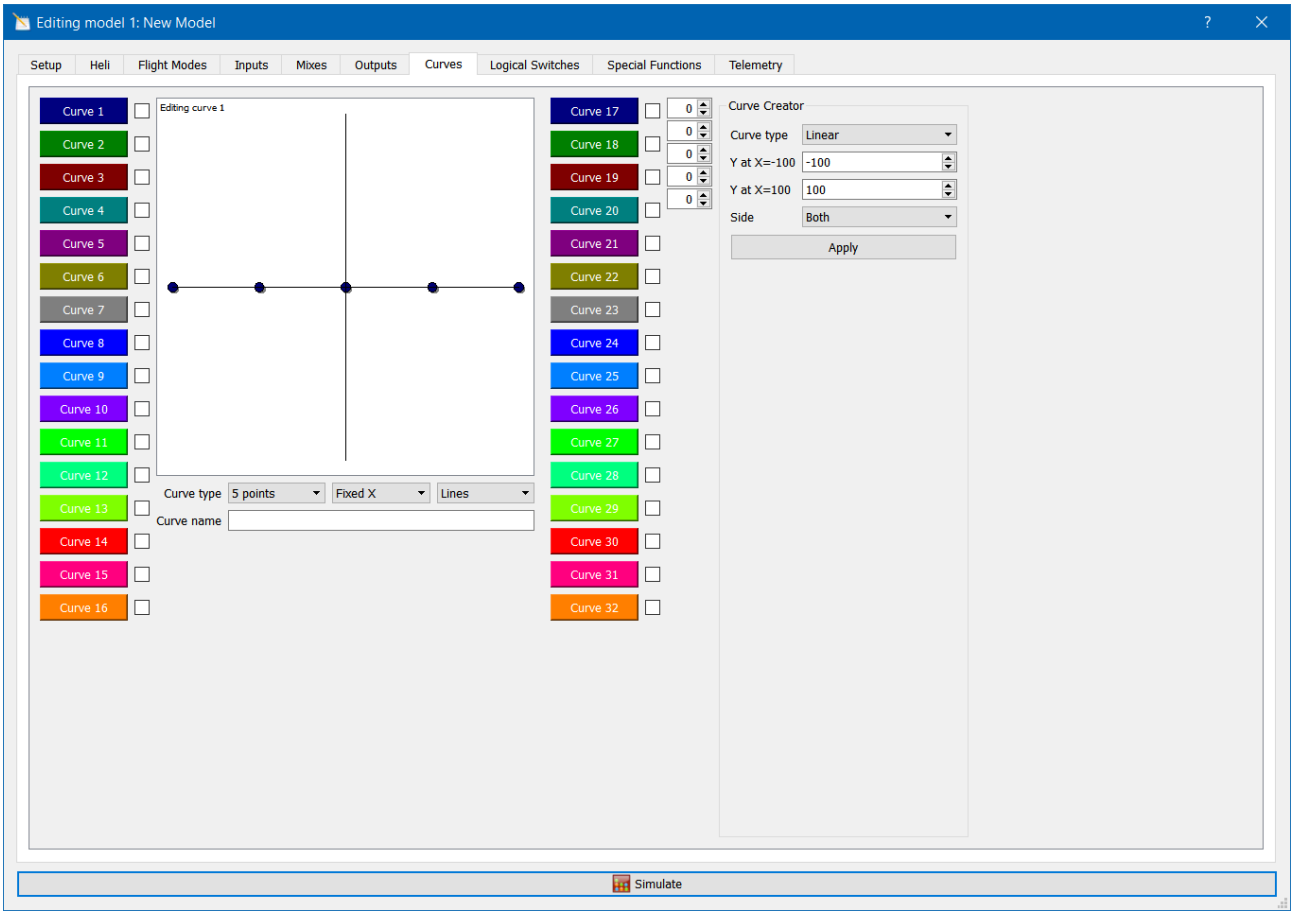
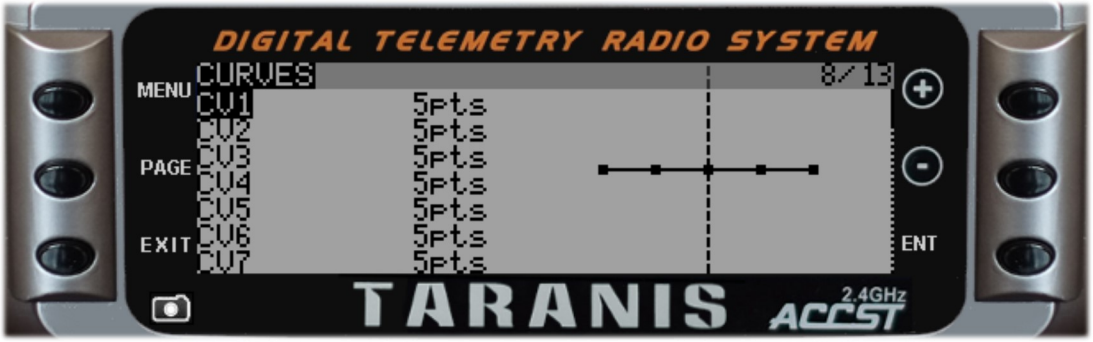
# The Curves Screen

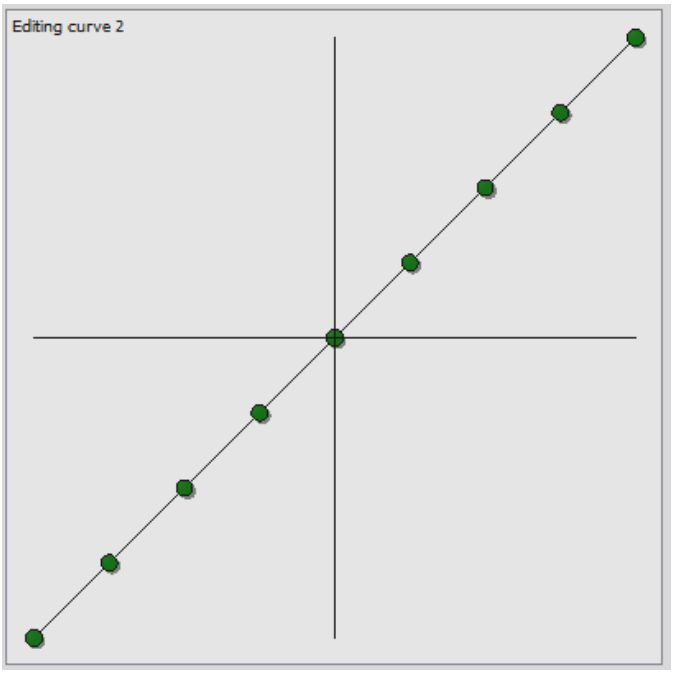


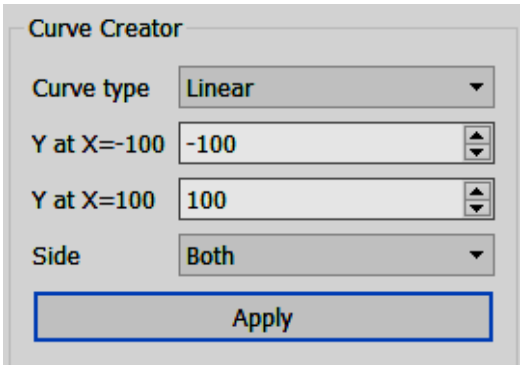

Custom curves can be used either in **Inputs**, **Mixes** or **Outputs**. There are 32 available, and they can have anything from 2 points to 17 points. They can have either fixed or user-definable X co-ordinates. Curves can be drawn on **OpenTX Companion** using either the mouse and moving points directly, by entering values for each point , or by using the **Curve Creator**.

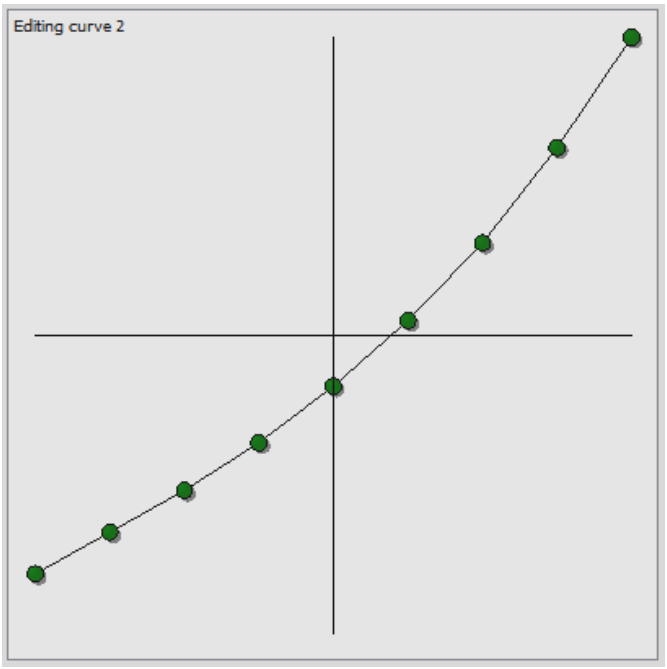
**The "X" (horizontal) axis is the input movement, i.e. a joystick, and the "Y" (vertical) axis is the output. The left hand side of the graph is with the input fully negative i.e. with the joystick fully left or fully down.**

The **Curve Creator** allows curves to be quickly created. To show how this works, it is easiest to give a few practical examples. Looking at a 9 point linear curve, we get this curve with the following settings in the curve creator.

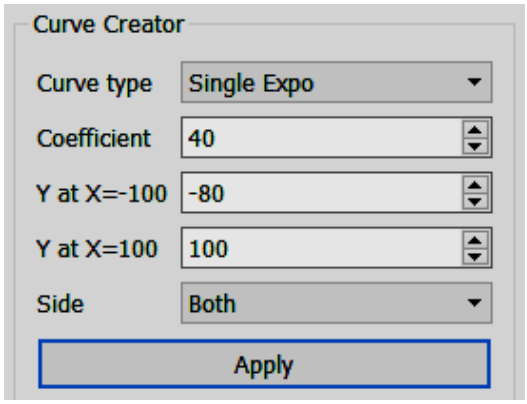


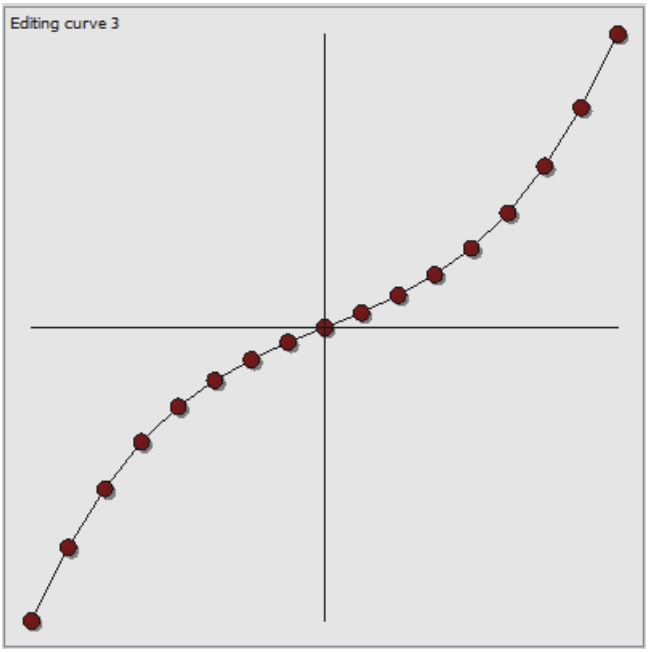

If **Y at X** is reduced to –80 then the curve will completely re-scale to give a curve that starts at
–80 and ends at +100.



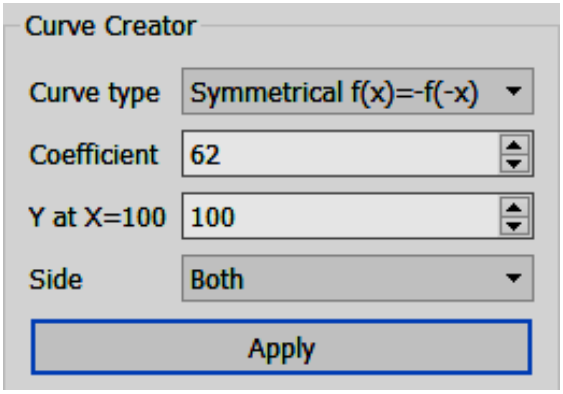



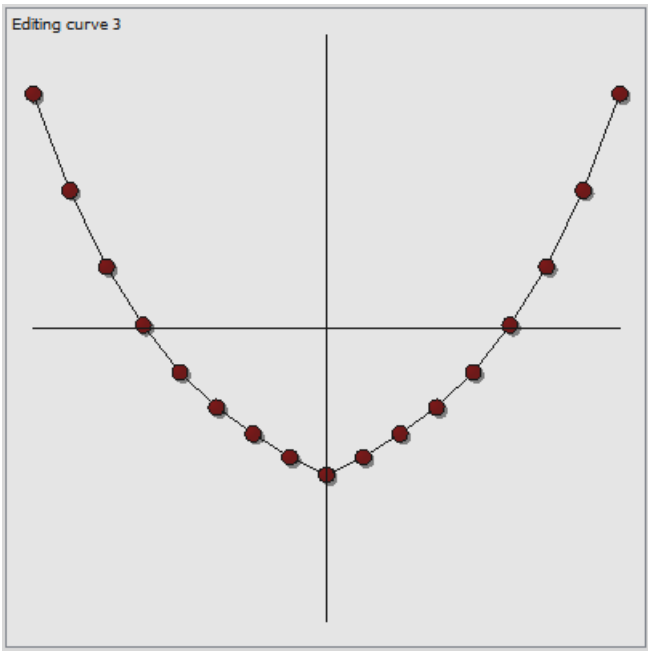

It is possible to also create exponential curves. Here we have an exponential curve which starts at –80 and ends at +100. Notice there is an extra box in the curve creator, **Coefficient**. This is the exponential value and has been set to 40 here.



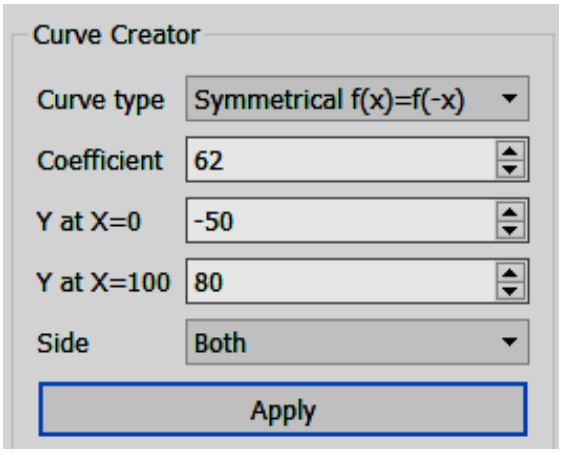


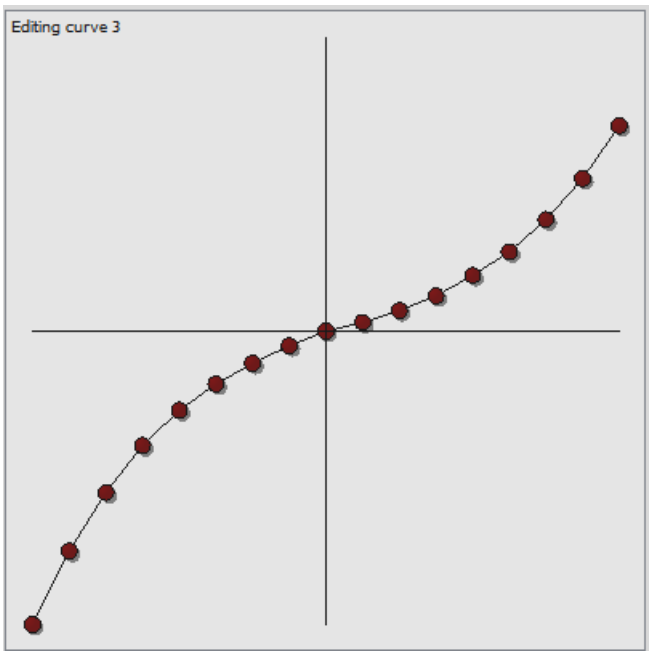

This is a 17 point curve using one of the functions, **Symmetrical f(x)=-f(-x)**.



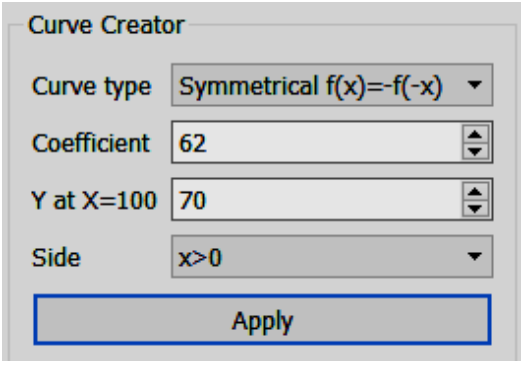



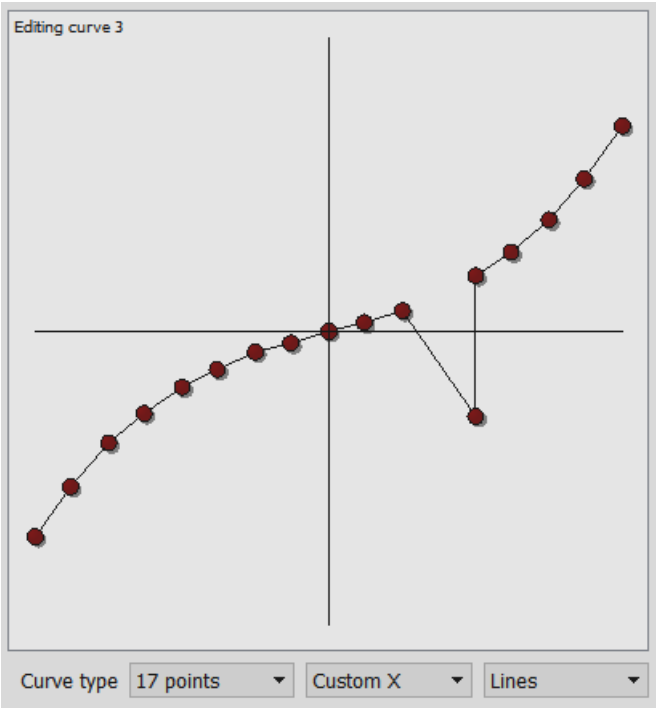

The function **Symmetrical f(x)=f(-x)** gives the following curve. Notice now there is an extra box to define **Y** when **X=0**.






Using the **Side** function, just one side of the curve can be changed. Here the upper limit is reduced to 70, but only for positive values of **X**. Thus there is an easy way of having asymmetrical curves for a variety of purposes.



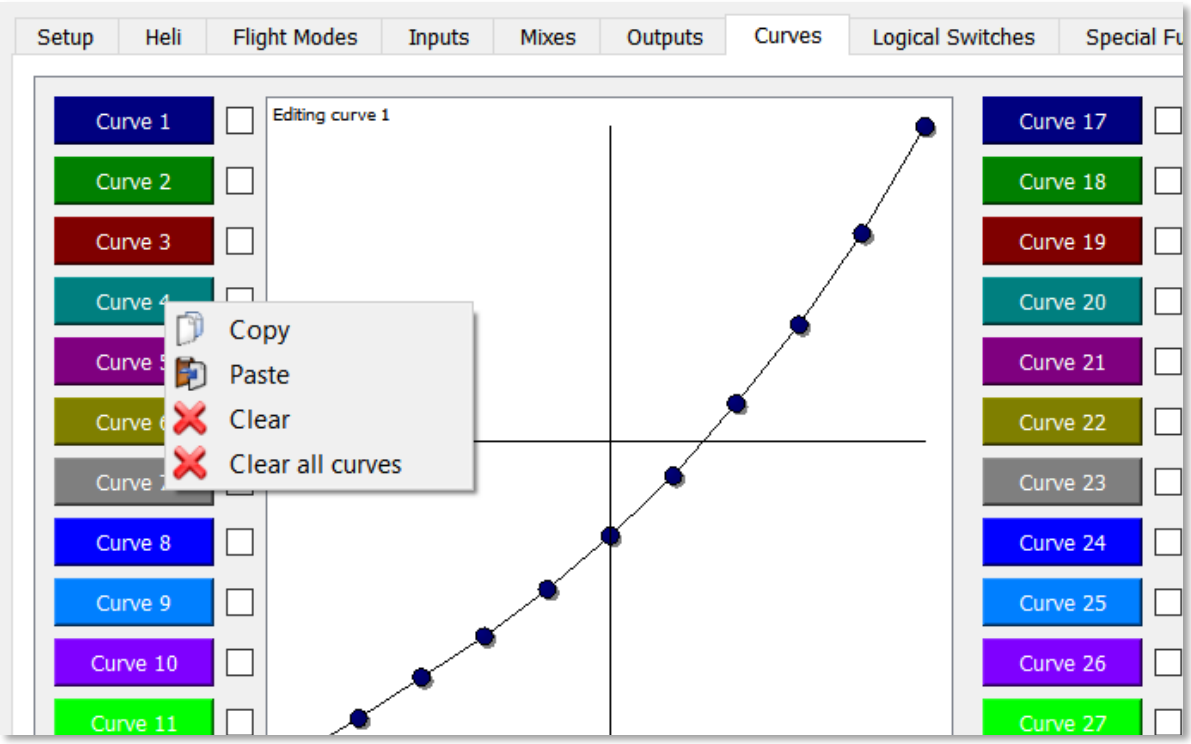




Finally, using the **Custom X** facility, individual points can be moved using the screen cursor to any point between the previous and the next **X** value and to any **Y** value.

Obviously, the scope and depth of the curve facility will entail much experimentation with an actual model in flight to perfect the operation. A simple technique to make this easier is to produce three similar curves, changing parameters slightly and assign them to a switch to be able to change curves in flight. This is easily done on the **Inputs** screen or the **Mixes** screen as shown below where switch A has been assigned to the three curves:

```
[I1]Thr        Thr Weight(+100%) Curve(1) Switch(SA↑)
               Thr Weight(+100%) Curve(2) Switch(SA-)
               Thr Weight(+100%) Curve(3) Switch(SA↓)
```

Another recently added feature that can simplify generating curves on the **OpenTX Companion** is a copy and paste facility. By right-clicking on the mouse, with the pointer on any of the 32 curve numbering boxes, a sub menu comes up to allow copying, pasting and deleting of curves.



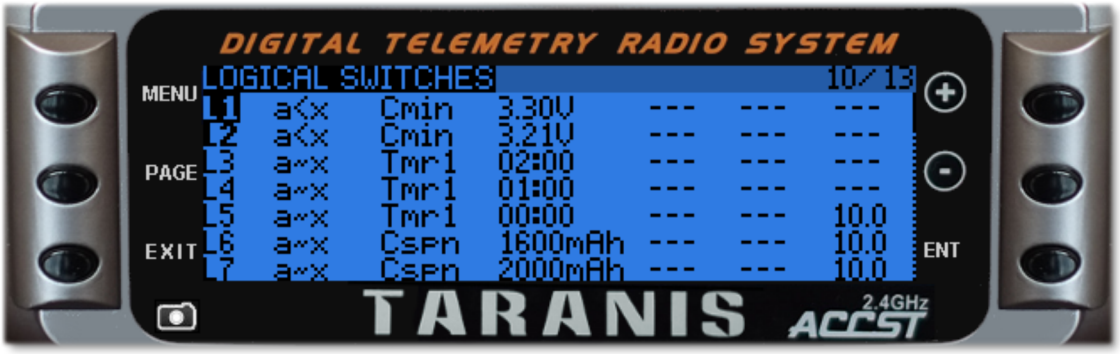
# The Logical Switches Screen

This screen, together with the **Special Functions** and the **Telemetry** form one of the most exciting parts of **OpenTX** which gives the system huge flexibility. Essentially it allows the user to create their own "switches" or conditions whereupon some other action will follow. Rather than being turned on or off by the action of physically adjusting a switch, they are turned on and off by evaluating the conditions set on the **Logical Switches** screen. These conditions may use a variety of inputs such as physical switches, other logical switches, sources such as telemetry values, channel values, timer values, or Global Variables. They can even use values returned by a LUA model script.

In the screenshot below, 8 logical switches have been set up, and between them they do three tasks. **L1** and **L2** look at a telemetry value, in this case the voltage of the lowest cell of a Lipo battery, and **L1** becomes true when the voltage drops below 3.3 volts. On the **Special Functions** screen, this is used to give a verbal "early warning" of low flight battery voltage. Similarly **L2** can give a critical battery warning. **L3** to **L5** look at the time set by **Timer 1**, used later to give minute countdowns, and **L6** to **L8** look at another telemetry reading, later used to give a verbal indication of the current consumption of a lipo battery.

Setup | Heli | Flight Modes | Inputs | Mixes | Outputs | Curves | Logical Switches | Special Functions | Telemetry

| # | Function | V1 | V2 | AND Switch |
|---|---|---|---|---|
| L1 | a<x | Cmin | 3.30 V | ---- |
| L2 | a<x | Cmin | 3.21 V | ---- |
| L3 | a~x | Timer1 | 00:02:00 [h:m:s] | ---- |
| L4 | a~x | Timer1 | 00:01:00 [h:m:s] | ---- |
| L5 | a~x | Timer1 | 00:00:00 [h:m:s] | ---- |
| L6 | a~x | Cspn | 1600 mAh | ---- |
| L7 | a~x | Cspn | 2000 mAh | ---- |
| L8 | a~x | Cspn | 2400 mAh | ---- |
| L9 | --- | ---- | 0 | ---- |



## Logical Switch Functions

Logical Switches are set by choosing the function, then refining the options (or parameters): **V1**, **V2**, **AND** switch, **Duration**, and a **Delay** for each switch. **Logical Switch** functions are given as expressions such as **a = x** where **a** represents the source to be used, given as **V1**, and **V2** (value 1 and value 2) gives the value for **x**. This can be a little confusing, but other conditions can be set also which do not use either **a** or **x**.

**a = x** — This is used to check if the value of a selectable source is equal to x, a chosen value.

**a ~ x** — This is used to check if the value is approximately equal. One needs great care when using the equals function because computers like to be exact! As an example, looking back at the previous screenshot, if one tests to see if the consumption is exactly 1600mAh, the actual telemetry reading may jump from 1598mAh to 1603mAh, so the condition is never met and therefore no message will be triggered. Most times it is better to use the approximately equals function rather than the "exactly" equals function. Do not use this function when the value will be precise.

**a < x** — The source, V1, is less than the chosen value V2.

**a > x** — The source, V1, is greater than the chosen value V2.

**| a | > x** — This is used to check if the absolute value (meaning irrespective of + or -) of a source V1 to see if it is greater than a chosen value of x, V2. Because it's an absolute function, whether the returned value is a positive or negative value doesn't matter.

**| a | < x** — This function operates similar to **| a | > x** except that it is true when **|a|** is less than x

**AND** — This switch checks that **both** the switches selected in V1 and V2 are true (i.e. ON). If both switches are true then the logical switch is true (ON).


In the above example, **L11** will be true if both switches **SA** and **SB** are set to

**OR** — This switch checks if **either** of the switches selected in V1 and V2 are true.


Now **L11** will be true if either (or both) of switches **SA** or **SB** are set to their mid positions.

**XOR** This switch checks if either, but not both of the switches selected in V1 and V2 are true.

**Edge** This is a momentary switch which can be activated by another switch (including logical and flight modes). The switch will stay on the length of time identified in duration. If testing with the simulator the switch will briefly flash green at the appropriate time if duration is 0, otherwise it will stay on for the duration time.

- V1 is an activating switch (including logical and flight modes).
- V2: is in two parts. The first, the delay before the switch activates, the second the "time window" for the switch to remain active. E.g.


Here **L1** will trigger if **SH** is held on for more than one second but less than three seconds.

- There are two general settings for **Edge**:
  - **0.0 (infinite)**, the default. The **Edge** switch becomes active when the triggering switch is **released**.


  - **(instant)**, The **Edge** switch becomes active once the triggering switch has been true for the minimum duration selected and then released. This setting is activated by clicking the down arrow next to the default value of **infinite**.


Using the simulator, one can compare **instant** to **infinite**.


**L2** will flash (i.e. signify true) when **SH** has been held down for one second, **L1** will then flash after **SH** is released. Neither will flash if **SH** is held down for a shorter period of time.

**a = b** This is used to check if the value of a selectable source is equal to b which is a different selectable source. It differs from **a = x** in that it compares two source values directly without specifying the numerical values for either.

**a > b** This is used to check if the value of a selectable source is greater than b, another selectable source.

**a < b** This is used to check if the value of a selectable source is less than b, another selectable source.

**d > = x** This function compares a change in value to a set value. This is used to check if the delta (the change in value) of a selectable source as chosen in V1, is greater than or equal to x as set in V2.

**| d | > = x** This is used to check if the absolute value of delta (i.e. delta always positive) of a selectable source is greater than or equal to x as set in V2. It operates the same as **d > x** without the need to specify a positive or negative signed value.

**Timer** This function is used to turn a logical switch ON or OFF at specified intervals. This is a repeating on/off timer with both variable on and off times.

- V1: the on time
- V2: the off time

Again this is very easily seen using the simulator.

**Sticky** This is a toggle. It can be thought of as another form of an on/off switch. It is turned on by the switch selected for V1 and turned off by the switch selected in V2.


The above function can be demonstrated using the simulator. Pulling **SH** on will first make **L18** true (go green). Pulling **SH** again will make **L18** false, in effect turning it off.

When using **Sticky**, take care as the flip toggle is turned on and off regardless of the **AND** switch on the same line. The logical switch will remain off if the **AND** Switch is false, but the toggle part will continue to turn on and off by V1 & V2, and is not reset by the **AND** switch.

## AND Switch, Duration and Delay

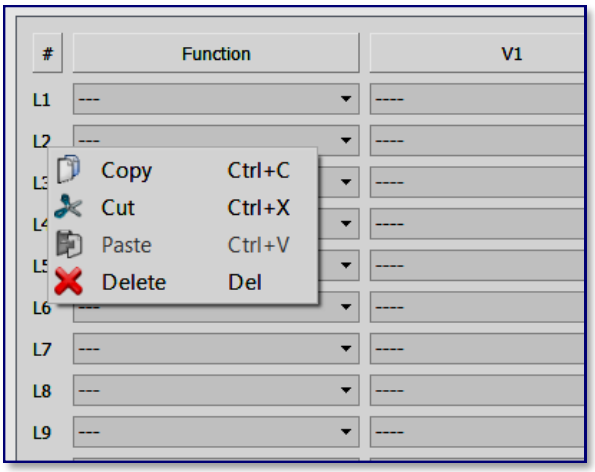
The **AND** switch, **Duration** and **Delay** for the switches work in the same for all switch functions.

**AND Switch** — Any physical switch, logical switch or flight mode can be selected from those available under the **AND** switch options. Only if this is condition is true and the rest of the switch conditions are true will the switch be on. The switch functions V1 & V2 are evaluated FIRST, then the **AND** switch applied afterwards. **This is important to remember, particularly with the Sticky.**

**Duration** — The length of time the switch will stay ON. If set to 0.0, the switch will remain on until the conditions make the switch off. Any other setting will cause the switch to go off after the number of seconds selected, even if the conditions remain true.

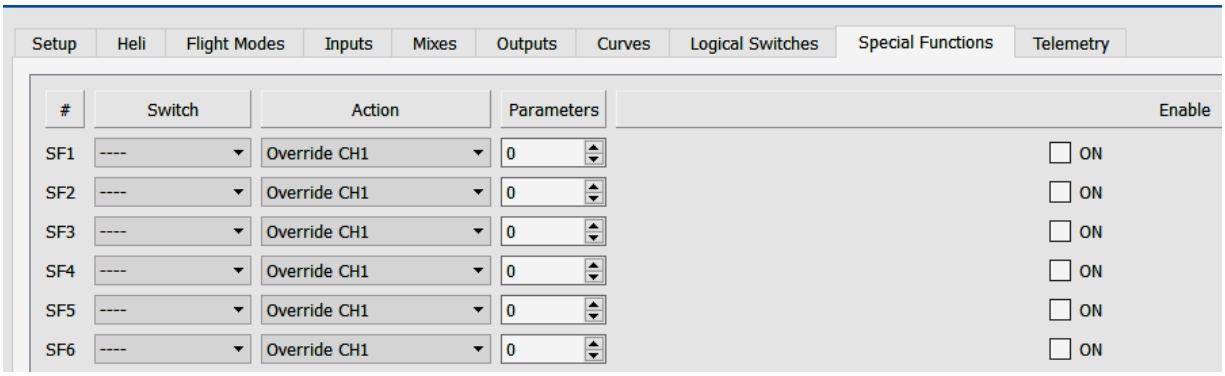
**Delay** — This is the delay before the switch comes on once the conditions are true.

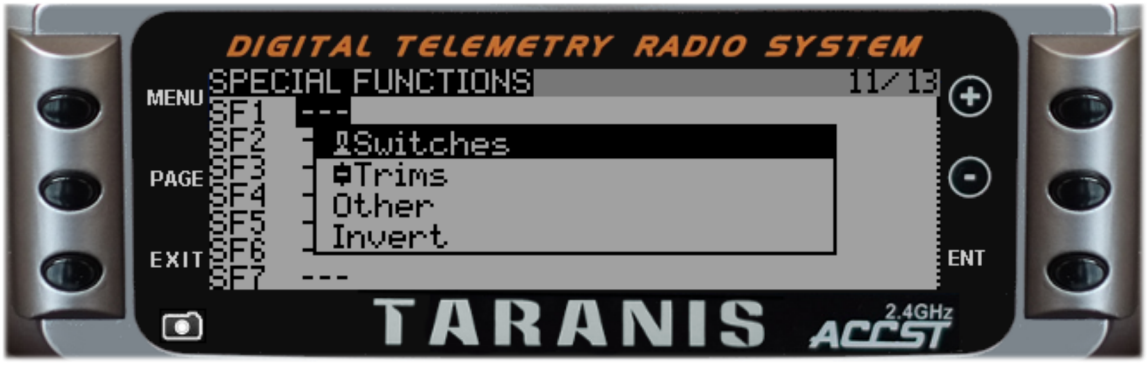
Finally, there is this neat little option. Right clicking on the **Logical Switch** number in the left hand column brings up a small menu to enable **Logical Switches** to be copied, cut or moved. Do note that moving the logical switch to another place on the screen will not alter the logical switch number in other screens of the program. This feature can also be used to copy a logical switch from one model to another.



# The Special Functions Screen

The **Special Functions** screen gives the option of adding all sorts of exciting features to a model program. For every model, there are up to 64 Special Functions available. In addition there are a further 64 **Global Functions** available. These are accessed from the **Setup** screen and are the same as the **Special Functions** expect that they operate on all model settings. Thus for a function one would like for every model, such as being able to adjust the volume level using one of the potentiometers, one can set it in **Global Functions** rather than having to set it each time for each model.




Basically there are four headings, **Switch**, **Action**, **Parameters**, and **Enable**.

## Switch

The switch input can be any one of the following

- The physical switches in any position.
- The trim switches when the trims are in the up, down, left and right positions.
- The 32 **Logical Switches**
- **On** This setting means the function is always enabled. e.g. for a volume control.
- **One** This setting triggers the function just once.
- Any of the 9 flight modes.

## Action

| | |
|---|---|
| **Override CH1 to CH32** | Weight -100 to +100 Forces a channel output to a specific value. |
| **Trainer: Rudder Elevator Throttle Aileron** | Enables trainer mode individually for each control. |
| **Trainer** | All channels enabled. |
| **Inst. Trim** | Adds the current stick position to the respective trims. When activating the selected switch the current stick positions will be added to their respective trims. This is typically assigned to a momentary switch, and used on a maiden flight if you expect trims to be way off. Instead of frantically clicking the trim tabs, you would hold the sticks so that the model flies straight, and depress the switch once. It is best to remove that entry after the maiden flight, to avoid hitting it by mistake and bringing the model badly out of trim again. |
| **Play Sound** | This will play any of the simple sounds listed in the drop down parameters box. |
| **Play Haptic** | Only works if haptic mode is enabled in the **General Settings**, and the transmitter supports haptic |
| **Reset** | Resets Timer 1, 2, 3, flight or telemetry.<br>✈ The selected timer value is reset to the value set by the timer parameters in the Model Setup screen.<br>✈ Telemetry resets telemetry values<br>✈ Flight resets both telemetry and timers |
| **Vario** | Vario will only sound when the designated switch is enabled. |
| **Play Track** | Plays any of the sound files stored on the SD card. A list of available sounds will appear in the parameters column in a drop down box. However, the **Companion** must have the correct pathway set (**Companion Settings Menu**) |
| **Play Value** | Speaks the value of any of a range of controls available in the parameters menu. These can include switch or joystick values, or telemetry values, or time. |
| **Play Script** | This will play a LUA script |
| **SD Logs** | This sets the frequency in seconds with which data logging is sampled and stored. (see data logging) |

**Volume** Used to adjust the volume. A simple application would be to use slider 2 to change volume. Note the use of the **ON** function.


**Backlight** Parameter shows a slider to set the level of the background light.


**Screenshot** This will take a screenshot of the transmitter screen. useful if one later wishes to check, say, a telemetry screen.

**Background Music** Switches on background music stored on the SD card.

**Background Music Pause** Pauses the background music.

**Adjust GV1 to GV9** Allows a global variable to be adjusted. There are four options:

- a **value** (between - 500 and + 500)
- **source** (the usual range of controls)
- **GVAR**, another global variable
- **increment** (+1 or - 1)

**SetFailsafe Int. Module** This feature allows the failsafe to be set dynamically, either on the ground or in the air. Assign a switch to this and with the control surfaces set as required for failsafe, simply pressing this switch will store their values. This particular feature if for the internal XJT module in a Taranis.

**SetFailsafe Ext. Module** As above but for an external module for the transmitter.

## Enable

There are several options under this heading:

**Played once, not during startup** This stops **OpenTX** playing this message when switched on

**No Repeat** Does not repeat the message. Otherwise setting the option to a time will make **OpenTX** repeat the message with a frequency set by the time stated.

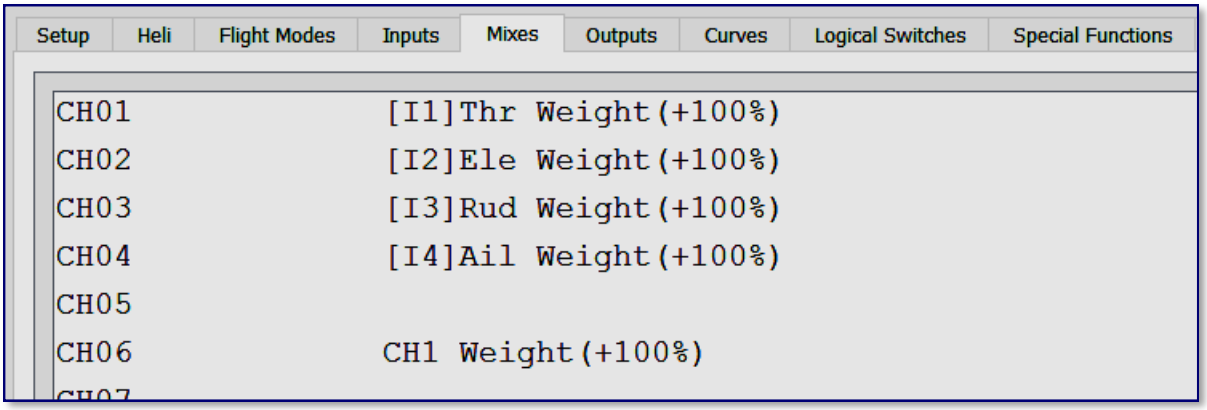
**ON** This box is ticked to enable the function at startup. Unticking this box is a simple way of disabling, say, the **Instant Trim** feature.

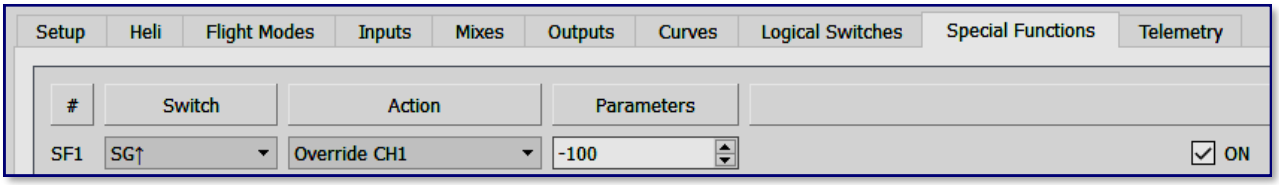
## Note on Override:

**Override** operates on a channel **Output** and is not part of the channel **Mixes** calculation. The consequence of this is that the **Mixes** are not aware of the override or its value. The channel mix value is not affected. The fact that an override is active due to a particular mix condition cannot be counted on as part of the mix calculation.

The purpose of **Override Special Function** is to force the channel **Output** to a particular value while the **Special Function** switch is activated, overriding all **Mixes** belonging to the channel. The **Output** value will be held even if the active mix is changed, unless the new active mix deactivates the **Special Function** switch.

This can be demonstrated thus:

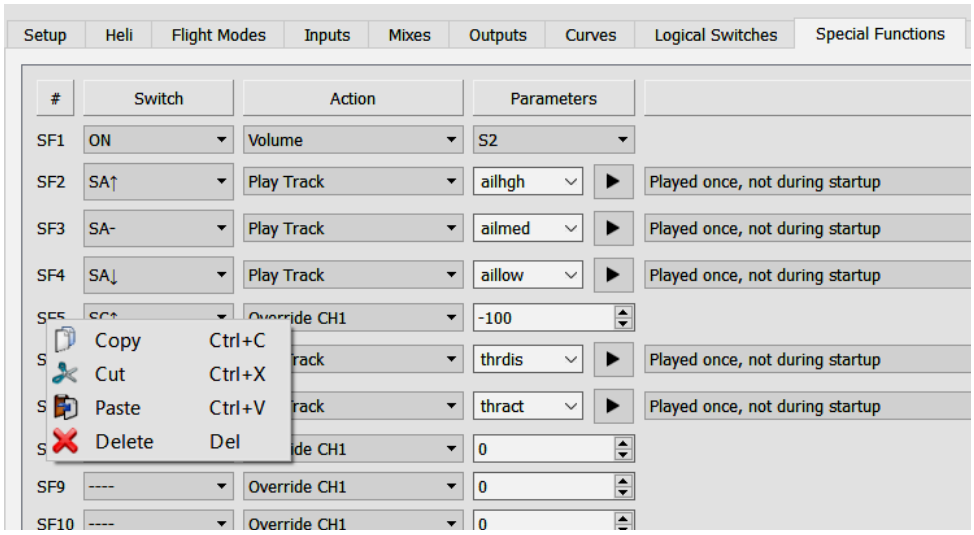


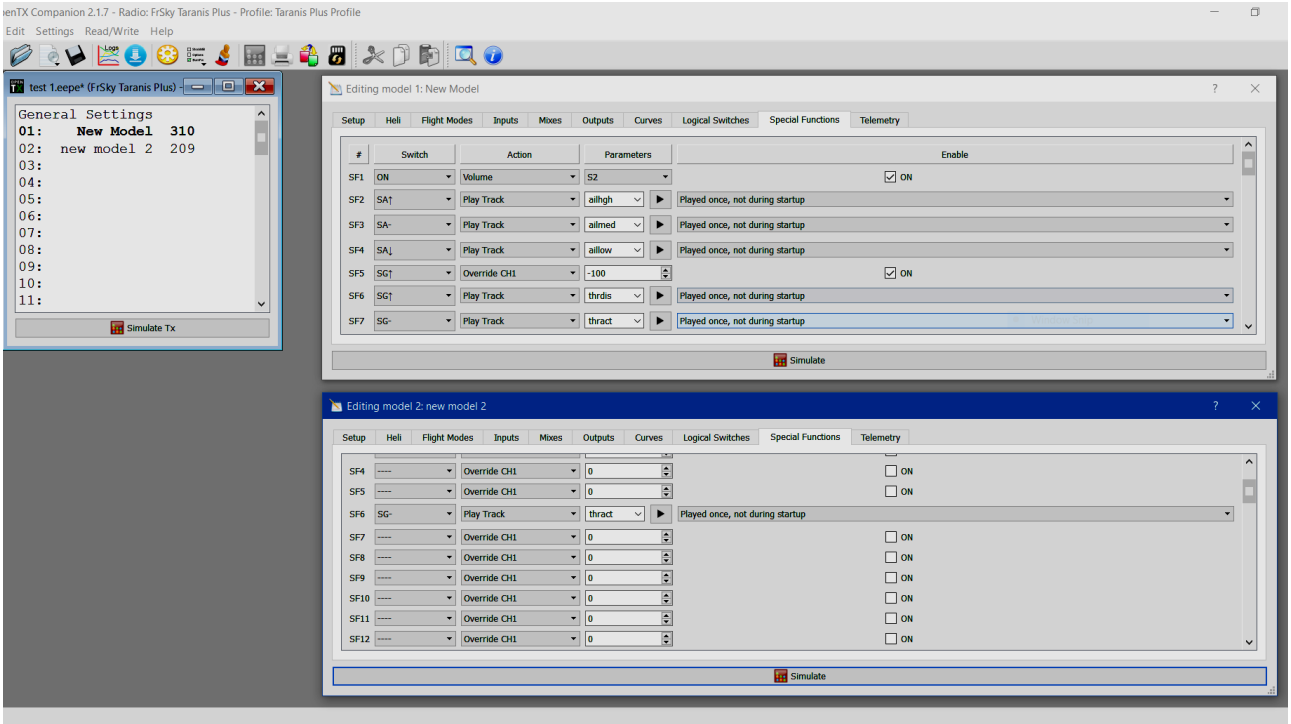

Using the simulator, one can see that **SG↑** channel 1 output is held at -100, whereas channel 6 will still show the channel 1 true value.

Checks also need to be made when using **Override** that servos are not driven past their normal operating range under any circumstances. **Override** takes place in the chain of commands after the **Outputs** screen, so any maximum and minimum limits set there are ignored.

A useful short cut when editing the **Special Functions** screen is right-clicking on the special function number in the left hand column to obtain the following menu box. This enables special function lines to be cut copied or moved not only within the same model functions screen, but between models too. This same feature is available with the **Logical Switches** as shown earlier.

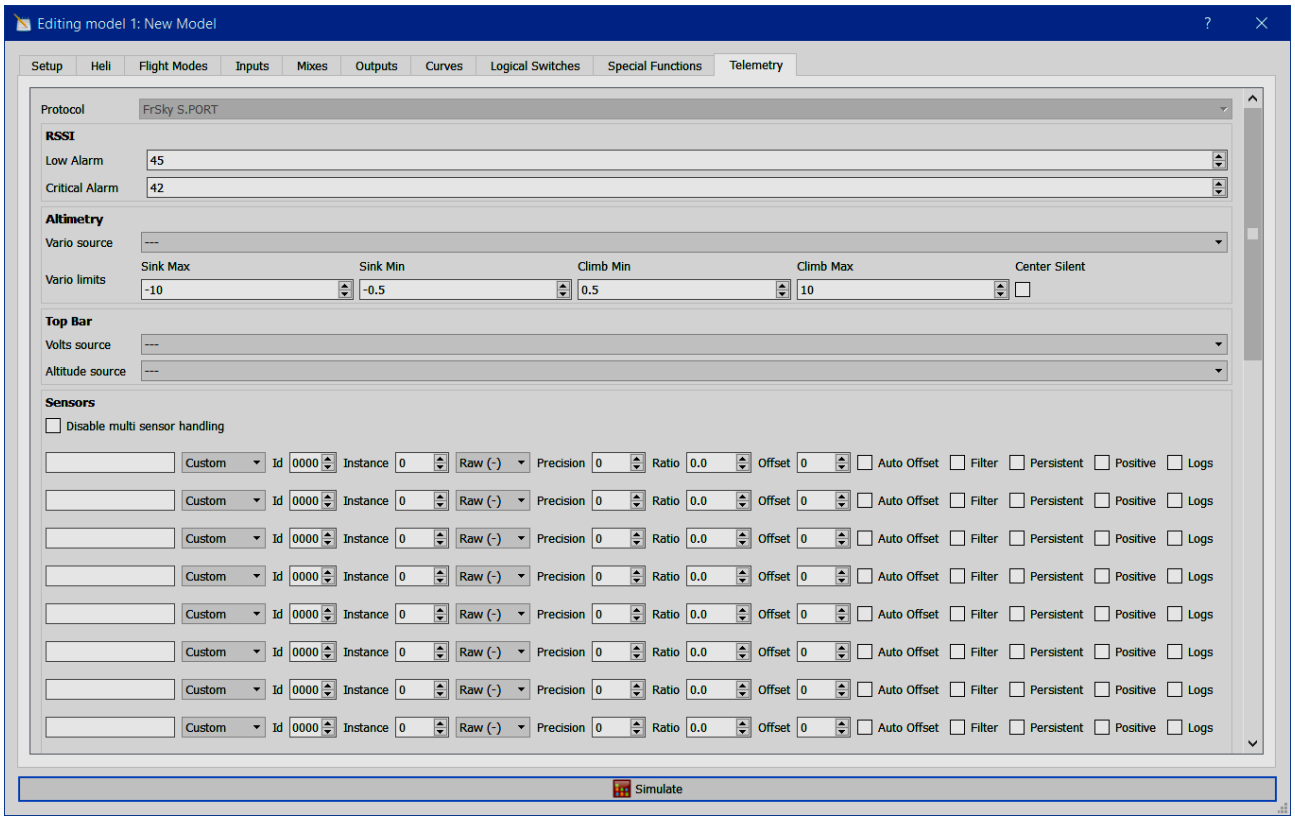


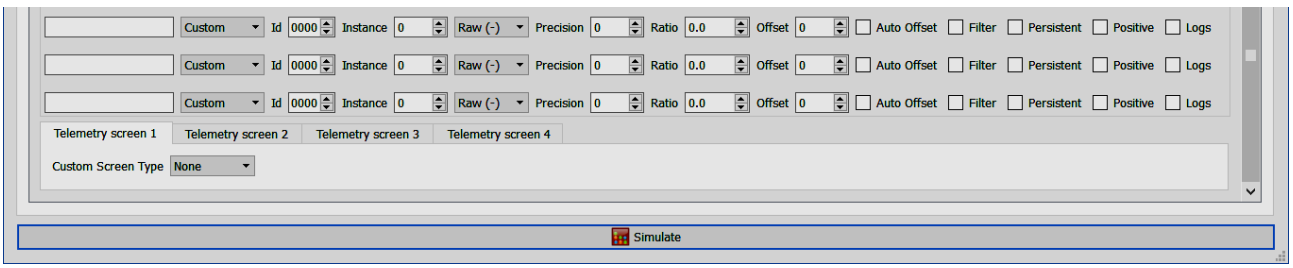

**OpenTX Companion** will allow several models to be edited at once, with several editing screens opened at once. By resizing the screens one can simply copy from one screen to the other.



# Telemetry and the Telemetry Screen

The **Telemetry Screen** opens a window onto a very comprehensive telemetry system. Together with the available FrSky Smart Port sensors, and those manufactured by third parties, the use of telemetry can considerably enhance the flying experience. For brevity, this manual will only consider the use of telemetry with the FrSky Smart Port sensors. Also it will only relate to the current range of telemetry enabled receivers available, i.e. the X4R, the X6R and the X8R. It should also be noted that this version of **OpenTx 2.1** telemetry is completely different to previous versions. Every sensor can be used everywhere for voice announcements, in logical switches, in inputs for proportional actions, and can be displayed on custom telemetry screens. One big advantage is that values can be seen directly on the telemetry setup page, so no need to add one to a custom screen just to have a look at it once. New data reception is visualised, and loss of a sensor is automatically detected.



Scrolling down, the bottom of the window looks like this:

**It is important to auto-identify the sensors early on in the programming process, otherwise they cannot be accessed by other screens.**

## Key Points

- FrSky Sport sensors are auto-discovered anytime the radio and receiver/sensors are powered up. This, obviously, cannot be done using the **Companion**.
- Each value sent from the receiver is treated as a separate "sensor" with its own properties (unit, decimal precision, ratio/offset) and options (auto offset, filtering, persistent storage at power off, logging enabled). Each sensor has its own user-defined name and keeps track of its own min/max value.
- Sensors can be duplicated within **OpenTX**, so for example a given value e.g. altitude from the same vario sensor can be displayed/announced/logged simultaneously in different units, or with different options (absolute altitude and altitude above start point with auto or manual offset,...)
- Multiple physical sensors returning the same value are supported as long as there is a way to differentiate them. Each FrSky sensor has its own default ID set. Therefore two identical sensors cannot be connected to one receiver without changing the ID of one of them. This can be done with the FrSky SBUS servo channel changer (SCC) as required so each ID is unique in the smart port chain. (The software for this can be downloaded from FrSky website.) Thus individual motor currents of an octocopter with 8 FCS-40A sensors can be handled, or individual cells of a 12S battery can be monitored by using two voltage sensors.
- "Calculated" virtual sensors can be manually created to combine values or extract extra data. Values can be added, averaged or multiplied, the minimum or maximum of a set of up to 4 values van be extracted. This also takes care of "special cases" like calculating GPS distance (2D or 3D), getting the value of a particular cell of a lipo cell sensor, calculating mAh consumption etc. For example **Power** can be calculated easily by multiplying the related voltage and current, total voltages of multiple lipo cell sensors can be added to get the total voltage of series-wired packs, the minimum cell of each of them can be extracted and the lowest of all can be found using the **Minimum** function.
- Each sensor can be reset individually with a special function, so no more losing all your min/max values when you just want to reset altitude offset to start point.
- The telemetry also logs all the switch and joystick positions from the transmitter if logging is enabled.

# The FrSky Smart Port Sensors

### FAS-40S, FrSky 40A Ampere Sensor-for S. Port

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Curr | Custom | Id | 0200 | Instance | 3 | A | Precision | 1 | Ratio | 0.0 | Offset | 0.0 |
| VFAS | Custom | Id | 0210 | Instance | 3 | V | Precision | 2 | Ratio | 0.0 | Offset | 0.00 |

This sensor is supplied with XT60 connectors to plug directly between the battery and the model's power lead. It will measure currents up to 40amps. This sensor uses an in-line resistor and measures the voltage drop across the resistor. If using near its current limit, check for temperature rises in the device.

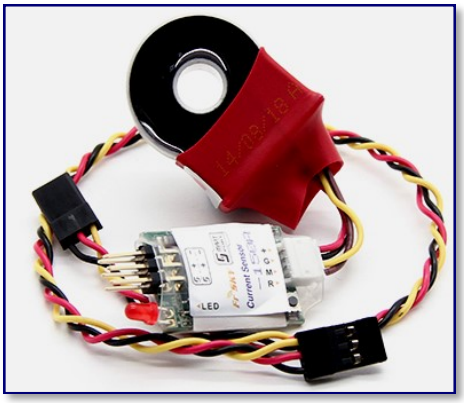
This sensor will also measure the lipo pack voltage, **VFAS** but not individual cells.


### FCS-150A, FrSky 150A Current Sensor for S. Port

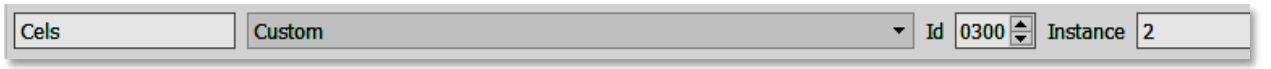
| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Curr | Custom | Id | 0200 | Instance | 8 | A | Precision | 1 | Ratio | 0.0 | Offset | 0.0 |

The FrSky FCS-150A is a current measuring sensor for applications drawing up to 150 amps. Simple setup of the FCS-150A means the ESC battery lead is passed through the sensor loop noting the current direction indicator on the side of the sensor. No reading will be obtained if the direction is incorrect.

Notice this sensor has the same **Id**, but a different **Instance** to the FAS-40S. It does not have a lipo voltage measuring capability.



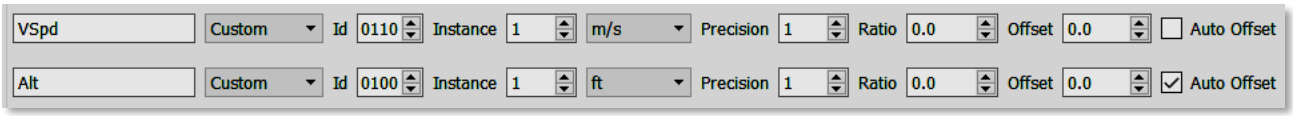
## FLVSS, FrSky Lipo voltage sensor for S. PORT






This sensor is used to monitor the voltages of LiPo packs. It incorporates a colourful easy-to-read (even in the dark!) OLED display showing both total voltage and individual cell voltages. Simply plug in to the lipo balance plug to view the voltage readings. The sensor can also be used as a handy, pocket-size, stand-alone lipo checker. As this sensor reads each individual lipo cell voltage, it can be configured to give a range of information about the state of the battery, and can be used with the **Logical Switches** and **Special Functions** to give battery warnings. A new version of this sensor, the "Mini" is available without the OLED display.

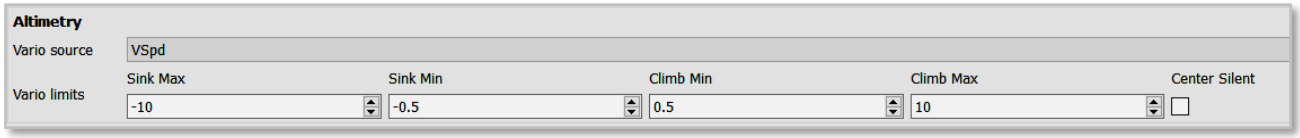

## FVAS-02H, FrSKY Variometer Sensor for S. Port



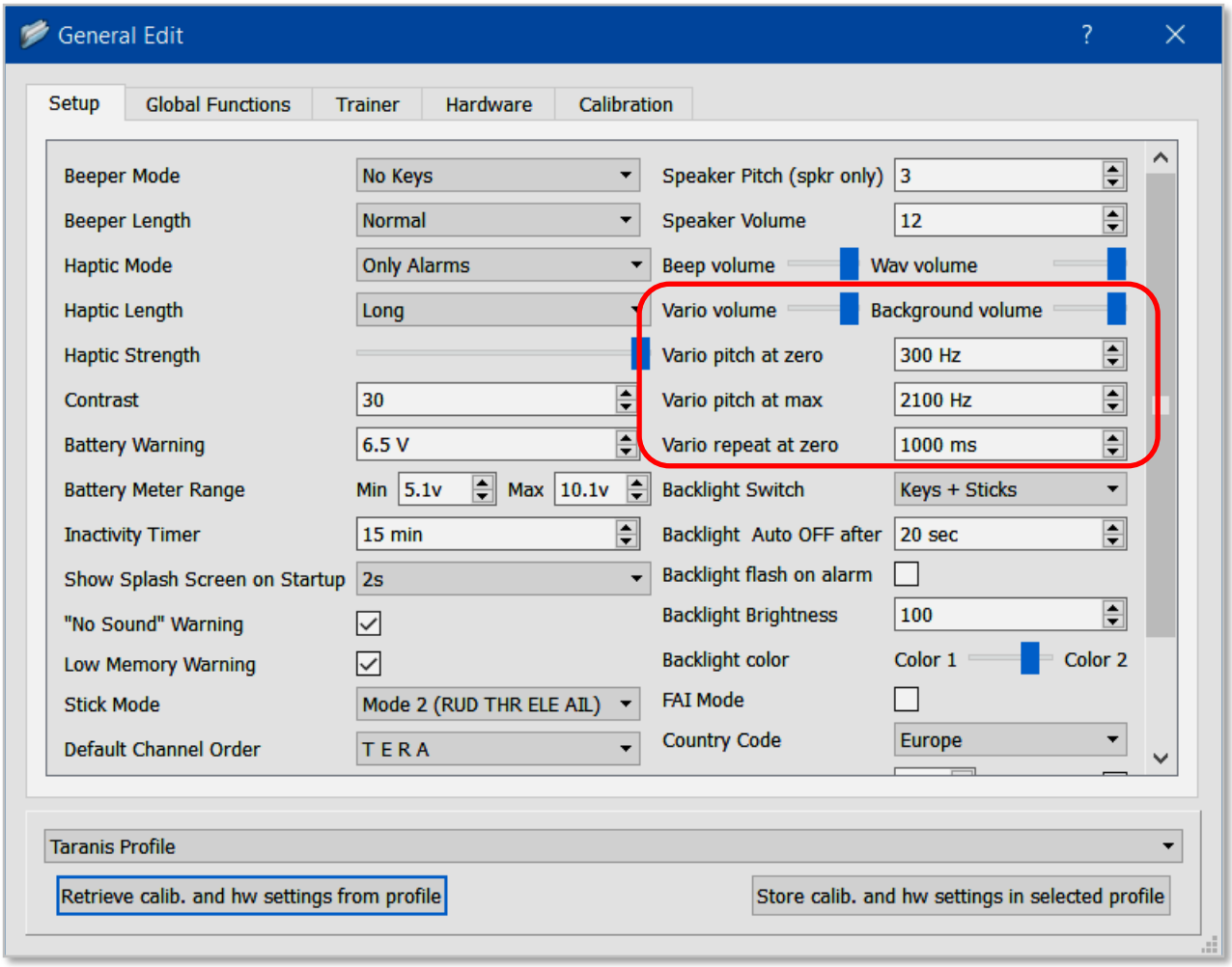

This sensor comes in two variants, normal precision and high precision. It creates two entries in the telemetry, **Vspd**, vertical speed, and **Alt**, altitude. It measures the rate of change of altitude by detecting the change in air pressure (static pressure) as altitude changes. It offers true glider vario features with tones that indicate lift and sink. As with most electronic barometric sensors, strong airflow and also pressure changes in weather will drastically decrease accuracy. The last letter of the part number designates the precision, "H", or high has a precision of 0.1m, "N" or normal has a precision of 1m.

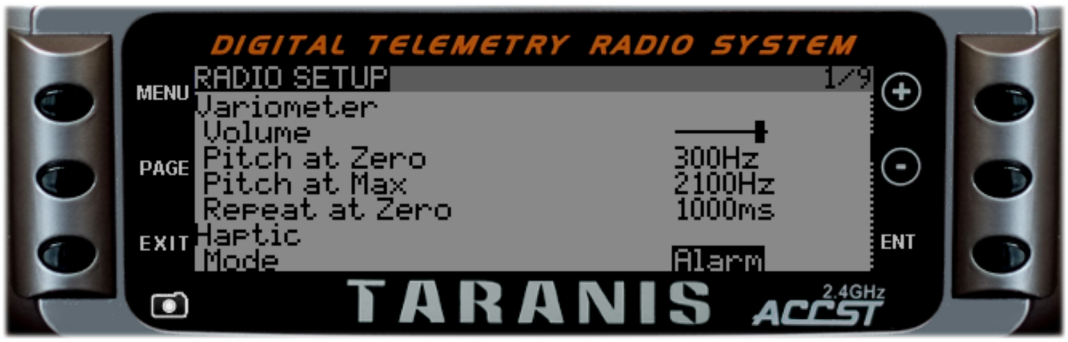

The vario offers two forms of telemetry. The first is a direct readout of height in one's chosen units. The second is a rising or falling tone produced by the transmitter to indicate whether the plane is rising or falling.

## Setting up the Vario



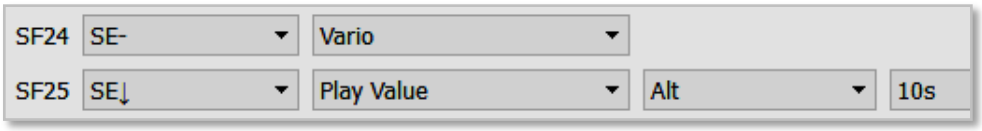

On the telemetry screen is the Altimetry box. First the **Vspd** must be entered as the **Vario source**. Then the limits can be set. They can be adjusted within the range ±17. Also the minimum points can be altered. These settings provide the rise and fall tone. Also there are some vario settings in the **General Settings Menu** on the **Companion**, or the **Radio Edit** on the transmitter:



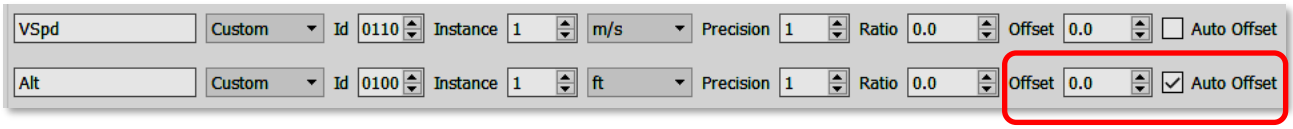



All these settings for the vario are down to personal preference and can be found by trial and error.

Obviously how one sets up the vario is up to the user, but one common method is to use a three position switch, and have the centre position for the vario and program the forward position to read out the height. Having switch SE off will disable both the vario and the height telemetry.



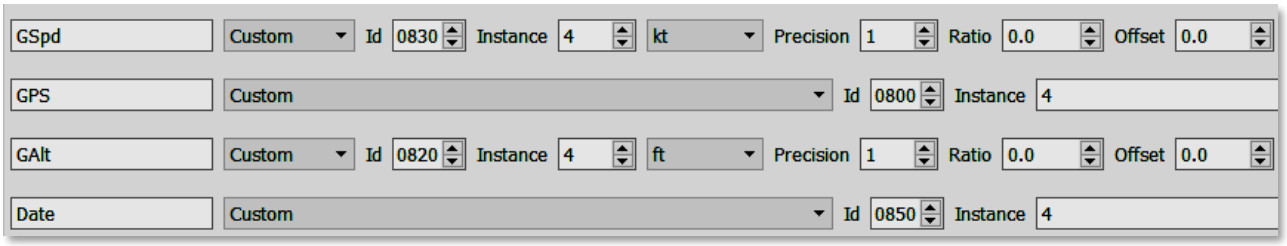

This is easily accomplished using a couple of special functions as shown above using switch **E**. All that is needed to program the vario is to set **SE-** to **Vario**. To give the spoken telemetry height simply select **Play Value** and select **Alt**. The setting in the right hand box is the delay between each reading. This can be set between 1 second and 60 seconds. One second gets very irritating, ten seconds seems to be a useful delay between readings, but as with so much in OpenTX this is down to the user's personal preference.




The **Alt** telemetry, by default, is set up to **Auto Offset** the height. This sets the height to zero at the place where the receiver is powered up. Thus one gets a height relative from ones flying position. If the absolute height is required, uncheck the **Auto Offset** box.

While the Vario might be seen as just the province of glider pilots, it can have other uses too. For the electric powered glider flying off a normal flying field, it opens up more precision in thermalling, or maybe this brings a whole new challenge instead of beating up the field seeing how fast it will go. It could be used with beginners in trainers to learn how to turn and maintain height, or even with more advanced flyers wanting to hone their skills.

## GPS-02, FrSKY GPS Sensor for S. Port



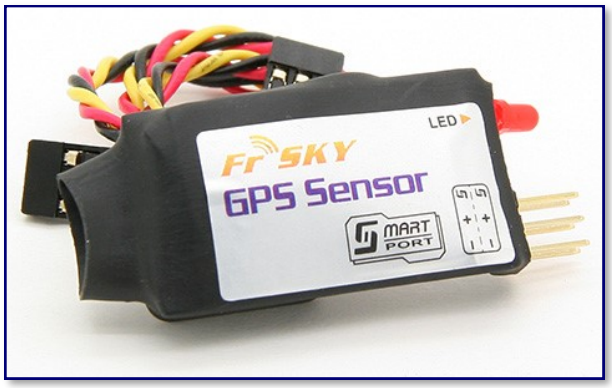

This particular sensor uses GPS to feed variable directional information.

**Gspd** Air speed

**GPS** GPS co-ordinates in format set in **General Settings**

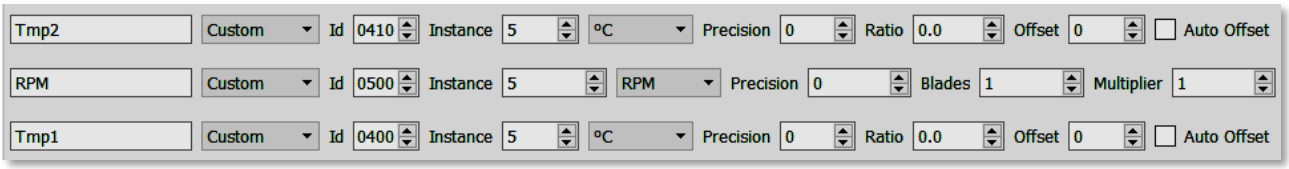
**Galt** Altitude

**Date** Taken from GPS data. Universal time co-ordinated (UTC).

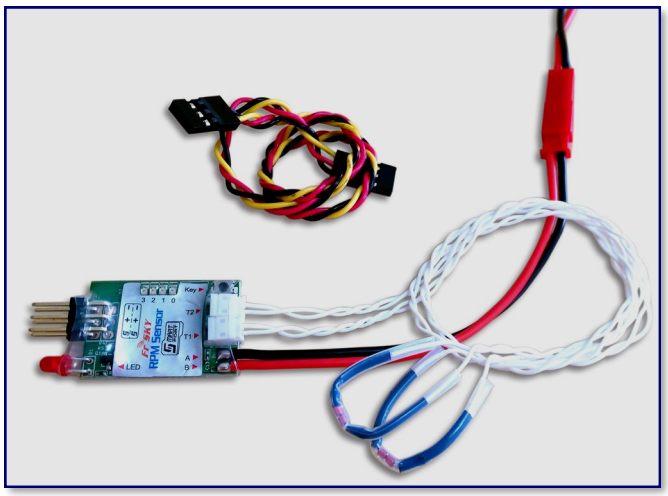



GPS co-ordinate format can be altered in the **General Settings** screen. The GPS sensor does require a few minutes to lock on to the GPS satellites.

## SP-RPM, FrSky RPM and Dual Temperature Sensor for S. Port

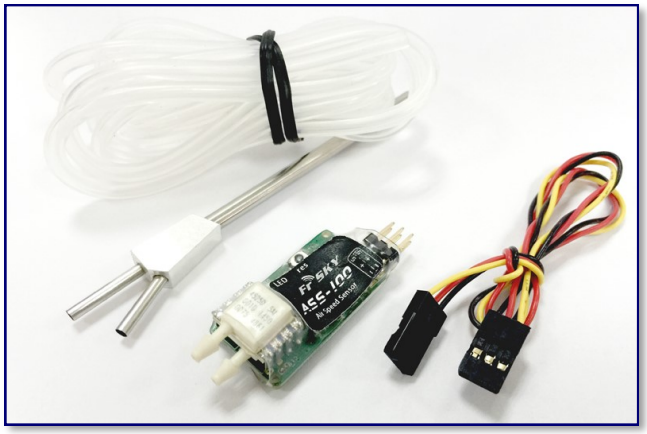



This sensor can work with different kinds of brushless motors and most speed control systems in market. Two temperature sensors are included with this product. These sensors can used to read temperatures of model accessories, and ambient air temperature.



### ASS-70/ASS-100 Frsky Air Speed Sensor for S. Port


This sensor comes in two versions, the ASS-70 has a measurement range up to 270km/h, whereas the ASS-100 will read up to 360km/h. It comes complete with a pitot tube and tubing.

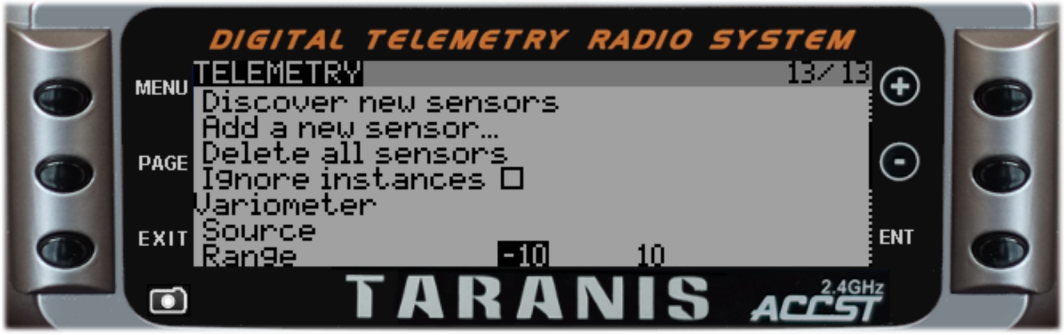


# Discovering the Smart Port Sensors

**Remember this discovery should be done at an early stage of the model programming to ensure that the sensors are visible in the other screens.**

Discovering new sensors must be added from the transmitter. They cannot be discovered using the **Companion**. However, once they have been discovered, the **Eepe** file can be uploaded to the computer when they will be accessible through the **Companion**.

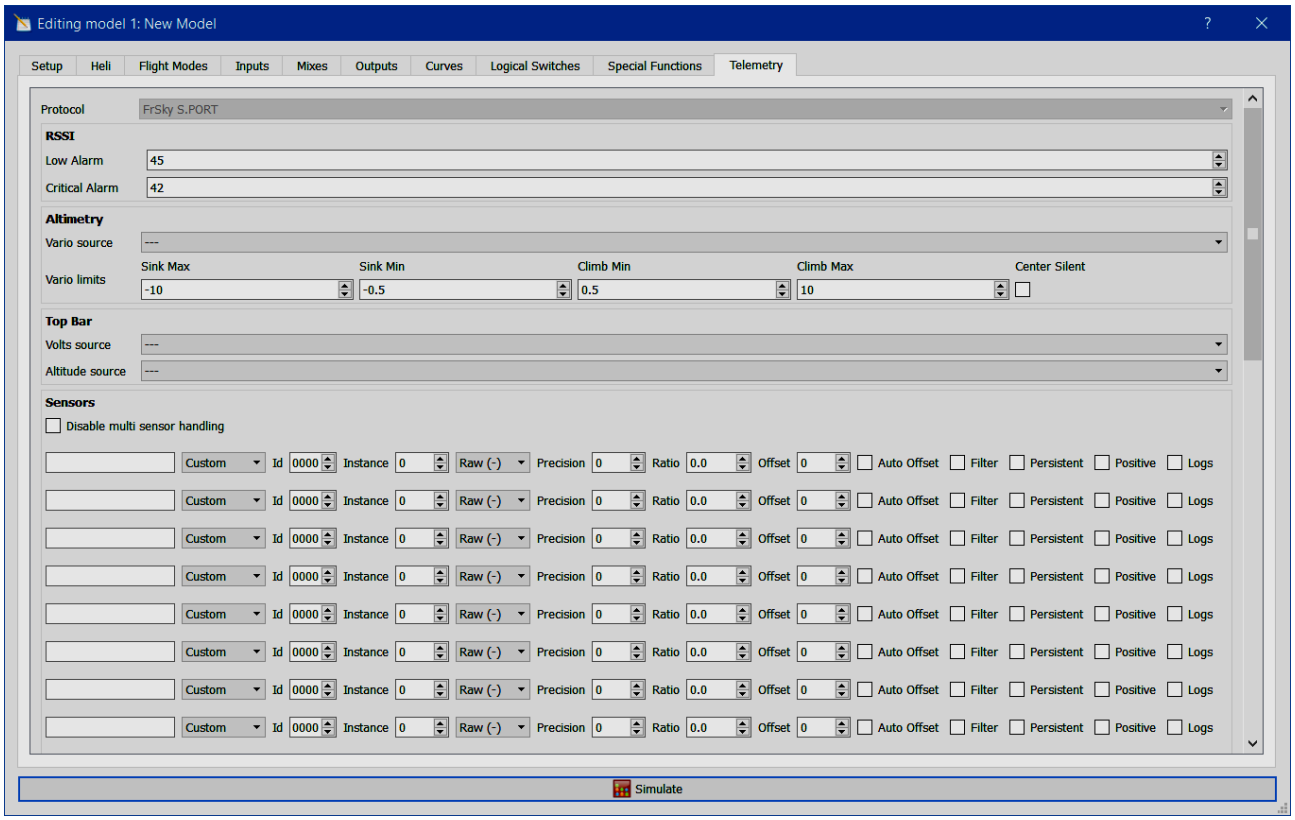
To discover new sensors, one needs the receiver, with sensors connected, powered up and bound to the transmitter set to the appropriate model. Go to the telemetry page and scroll down to **Discover new sensors** and press **ENT**. This will find any sensors. Then select **Stop discovery**. Remember to keep the transmitter at least 1m away from the receiver to ensure telemetry signals are received. You can also add and delete sensors from here. Although sensors will appear, the values might not appear straight away. GPS data takes several minutes to start coming through. However, this does not matter for the discovery process, and one does not need to wait for the data to appear. Once the sensors have been detected, the receiver can be powered down, and programming can continue.




**Add a new sensor** allows the used to add a calculated sensor. This is covered in more detail further on in this section.

**Ignore Instances** is the same as **Disable multi sensor handling** on the **Companion** screen. This should not normally be ticked. It is to stop third party sensors that flood the sensor screen filling all the slots.

# The Telemetry Screen




## Protocol

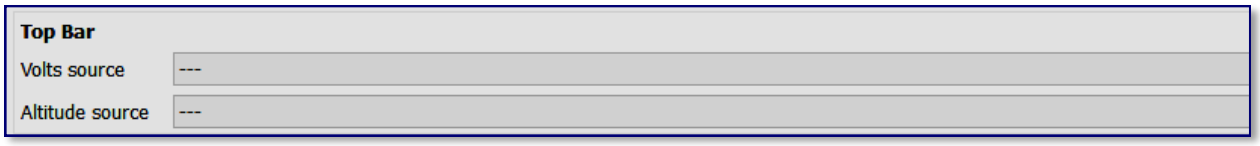
In this screenshot, the Protocol box is currently greyed out, and cannot be changed from FrSky S Port as the transmitter is set to Taranis. Other radios will allow different protocols to be used. These appear in a drop-down box.

## RSSI

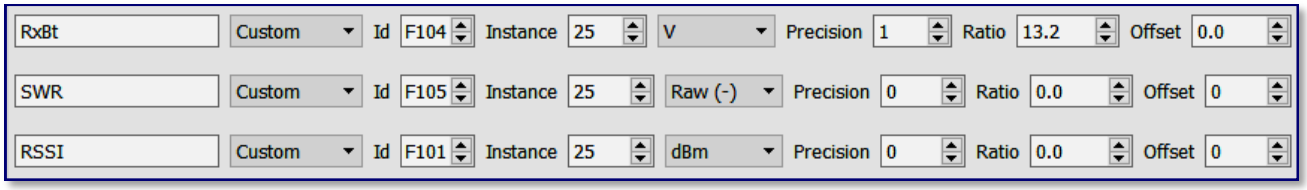
RSSI stands for Received Signal Strength Indicator. This is a value transmitted by the receiver in the model to a receiver in the transmitter that indicates how strong the signal is that's being received by the model. When it drops below a minimum value, this indicates that one is in danger of flying out of range. RSSI warnings are automatically built into the OpenTX software, and are triggered by the two settings automatically set in the telemetry window. These can be changed by the user, but the default values set, 45 and 42 should be reduced with extreme caution. However, increasing the values slightly would give a greater safety margin. It should be noted this is a built in telemetry function of the X4R, X6R and X8R receivers and requires no additional hardware.

## Altimetry

This option requires one of the vario sensors to be connected and discovered first.




These options refer to the top bar of the telemetry screen where you are able to show a voltage source and an altitude source, depending on what sensors have already been discovered.




Without any sensor connected, the above three “sensors” are always detected as they are built into each “X” series receiver.

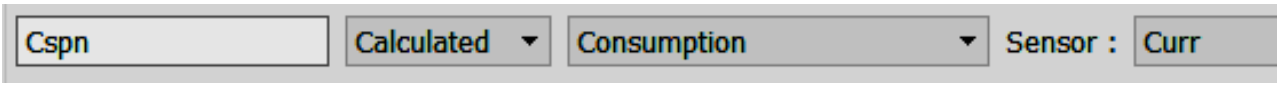
| | |
|---|---|
| **RxBt** | The voltage of the receiver battery. |
| **SWR** | SWR stands for "Standing Wave Ratio" This is a measure of transmitter antenna quality. Reading below 51 is normal. Note that SWR does not work on some earlier Taranis and Taranis Plus radios. |
| **RSSI** | See earlier notes about this. |
| **Disable multi sensor handling** | This should not normally be ticked. It is to stop third party sensors that flood the sensor screen filling all the slots. |
| **Units** | This drop down box offers a range of units. Conversion is carried out automatically if the unit is acceptable alternative for the default unit. i.e. **mph** can be changed to **km/h**, but not **volts**, although the units shown appear to have changed. |
| **Precision** | The number of decimal places required. |
| **Ratio** | This is to adjust sensor outputs to ensure to give a realistic range. |
| **Offset** | Offset adds a constant to the value. |
| **Auto Offset** | Stores the first received value after a reset as zero. |
| **Filter** | Used to make a value less “jumpy”. |
| **Persistent** | The value gets stored at power off and recalled next time. Can be useful for mAh or fuel totalizer, for example. |
| **Positive** | Gives only positive values. |
| **Logs** | Value will be set to the data log if ticked. |

## Calculated Telemetry

**OpenTX** not only gives the ability to see and use telemetry values transmitted directly from the receiver, but also to calculate new telemetry fields. Up to 32 physical and virtual sensors can be used for each model. When used in other **OpenTX** screens, most of the sensor values can have three options, the actual sensor value, the minimum sensor value (the sensor name denoted by a - sign in front) and the maximum sensor value (the sensor name denoted by a + sign in front). Thus if a current sensor is fitted, one can see the maximum current drawn during a flight. Each new virtual sensor can be given a name of up to 4 letters, and **OpenTX** differentiates between capital letters and lower case. Thus it is possible to have two virtual sensors names such as **Batt** and **batt**. If, during editing, a calculated sensor name is changed, other screens using that name will also change automatically. This does not normally happen with **OpenTX**.

## Examples of Calculated Telemetry

### Flight Battery Consumption




This requires the current sensor **Curr**, and the field is calculated. A very useful derived value for giving warnings of flight battery consumption.

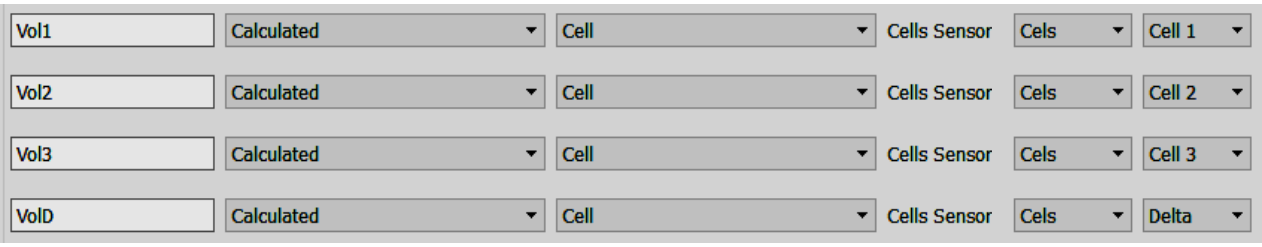
### Battery Power, Electric Flight


Here the current is multiplied by the voltage (using the voltage sensor built into the 40amp current sensor) to give the power in watts. Useful for seeing the maximum power drawn during a flight, and together with the logs can see how long this power was maintained, as it will reduce as the battery voltage reduces.

### Lowest Lipo Cell Voltage Reading (using the voltage FLVSS sensor)


This gives the voltage reading for the cell with the lowest voltage. Again this is very useful as an early warning of battery failure, depending on the voltage levels set in the **Logical Switches** section. Typically, if set appropriately, this can alert of a failing battery when full power is applied. Obviously the speed controller should be programmed to reduce voltage at a lower value than is set in the **Logical Switches.**

### Individual Lipo Cell data (using the voltage FLVSS sensor)




Here **Vol1**, **Vol2** and **Vol3** are the individual cell voltages. **VolD** gives the delta value (the difference in voltage between the highest cell and the lowest) for the battery. Note only three cells have been shown here, if it is a 4S battery, **VolD** will give the delta value between the four cells in the battery.

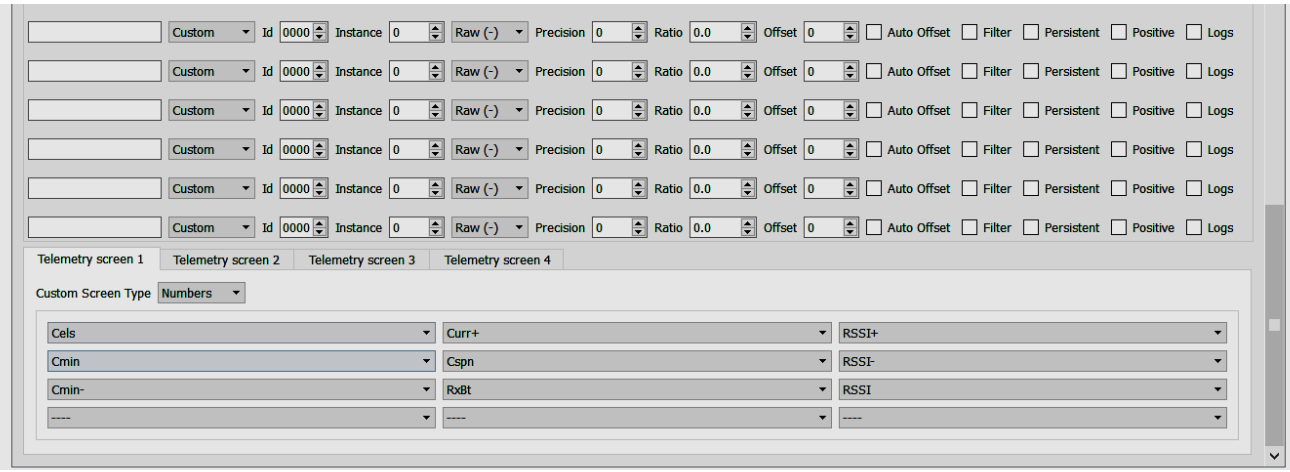
### Measuring Distance using the GPS

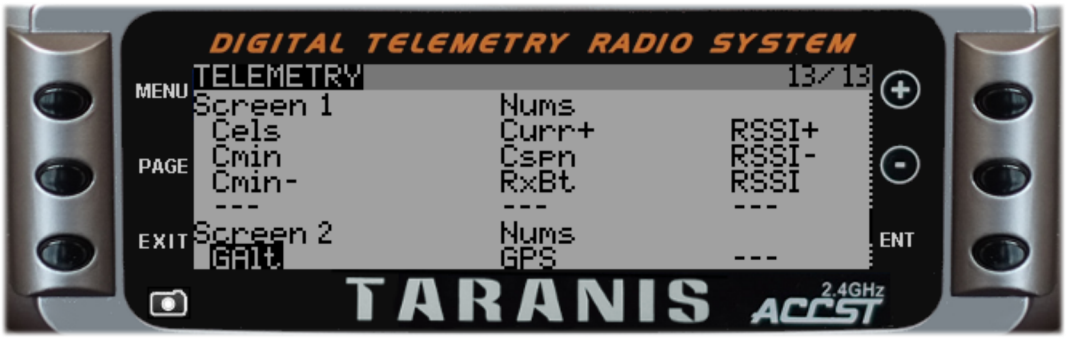

The calculated sensor is created using the formula **Dist** (Distance) with the **GPS** as the source. This will give a 2D distance.


If a 3D distance is required, first edit the **Altitude** field to turn on **Auto Offset**, then include the **Galt** source.

# Setting up Telemetry Displays

Up to 4 telemetry screens can be set up on the transmitter. Each screen can use numbers, bars or use Lua scripts to display the data. Setting up a telemetry screen can be found at the bottom of the main **OpenTX** telemetry page




First select the telemetry screen, then the custom screen type, then any of a whole range of parameters can be entered into the screens. This is probably the last operation to set up once a model has been programmed, as it can contain a range of parameters including logical switches, and telemetry values.

These screens are really very much down to individual preference, though for electric flight users, having a screen record of such things as maximum current drawn, pack voltages and maximum power can be very helpful.

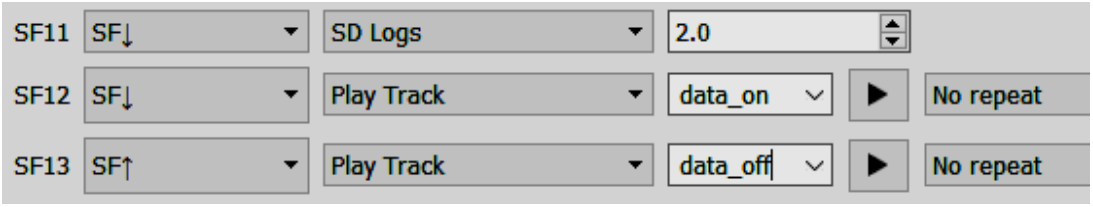
# Data Logging

Data is constantly being sent back to the transmitter from the receiver and any sensors connected to it. This data can be stored on the transmitters SD card, and downloaded to a computer using a USB cable and Bootlogger mode on the transmitter, or by removing the SD Card and plugging that into a computer.

To enable data logging:

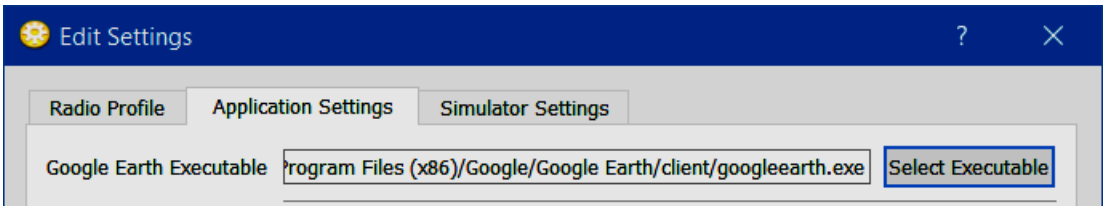
1. Ensure that the appropriate **Logs** box is ticked on the **Telemetry Screen**. It is easy to tick every box, but some data does not need to be logged. E.g. with the GPS sensor, it will report back on the date. Logging this field is mostly unnecessary, especially as the saved log file always included the date and time of the flight.

2. Several **Special Functions** need to be set up to enable logging. There are various ways of doing this:



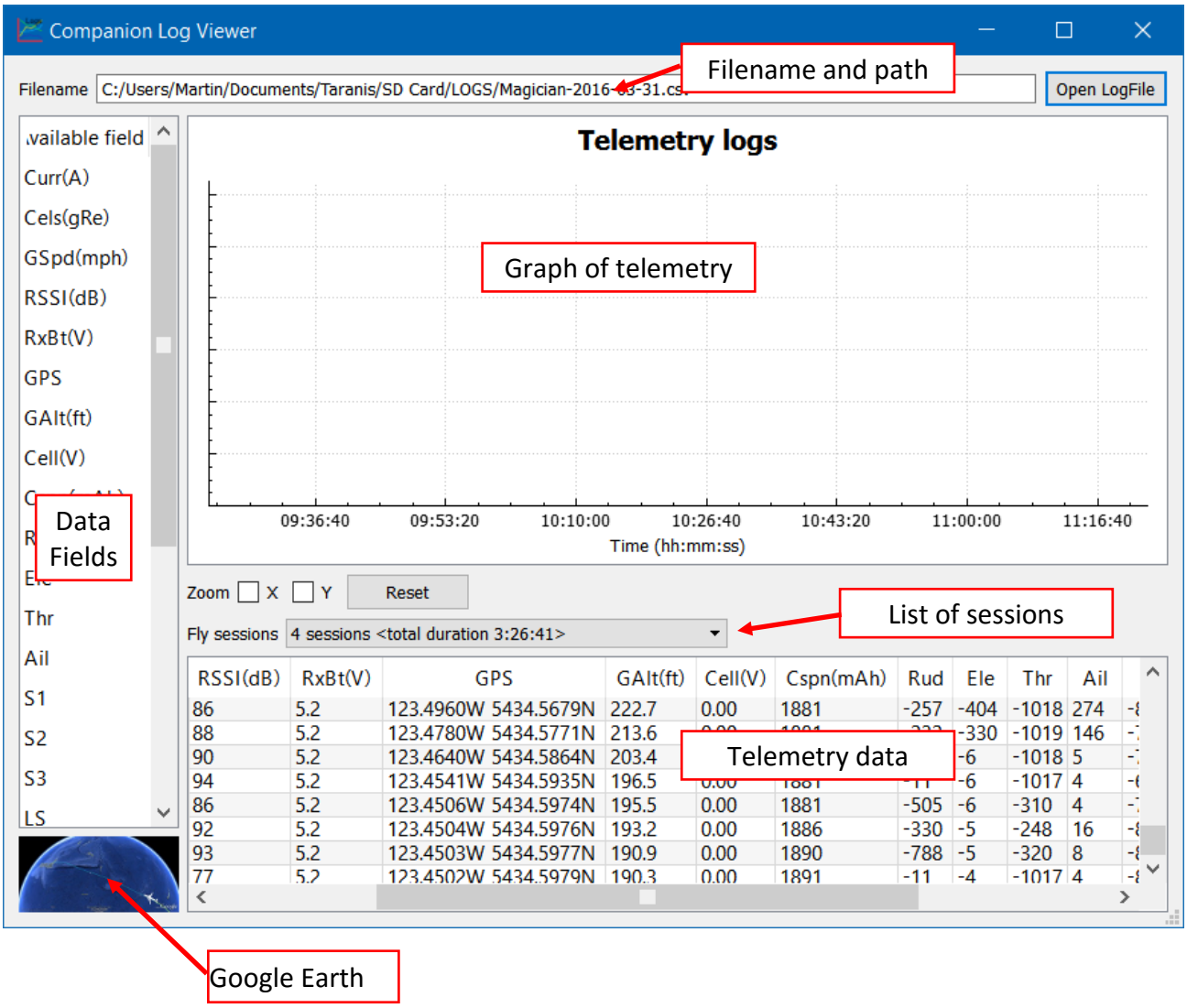

   This method simply uses switch F to start and stop logging. The parameter in **SF11** sets the frequency of logging, in this case every 2 seconds. **SF12** and **SF13** simply give verbal warnings of the data logging state. On a 10 minute flight, that still amounts to 600 lines of data!
   Other methods could include using a **Logical Switch** to determine when the countdown clock starts for instance, or using the speed parameter if a GPS or speed sensor is fitted, and enabling logging when the speed is greater than, say, 5mph.

3. If wanting to use Google Earth with the GPS sensor, then it is necessary first load Google Earth onto your computer, and then enter the correct path where Google Earth is stored on your computer from the **Edit Settings**, **Application** menu of the **Companion**.




The log files stored on the SD card will have the filename of the model, and the date recorded. Each log file may contain a number of logging instances undertaken during that day.

It should be noted that the GPS sensor requires a couple of minutes to activate, this could be important if one only switches power on just before a flight.



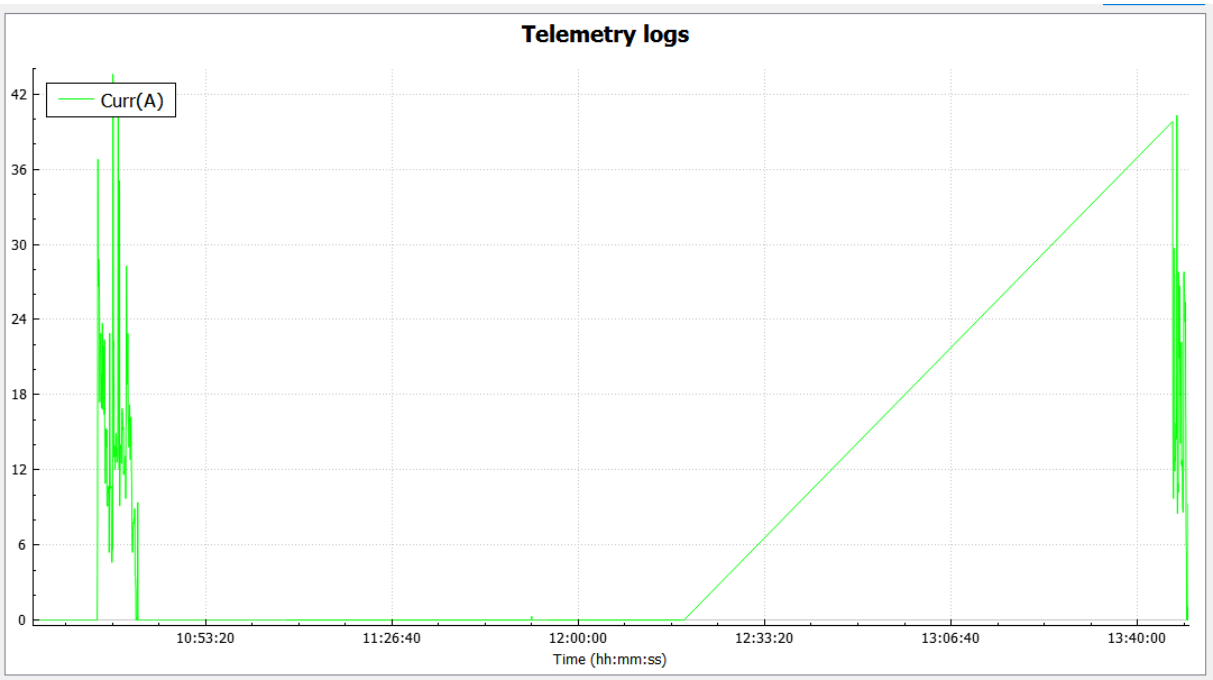

This **Telemetry Log Screen** is accessed from the **OpenTX Companion**, either from the **File** menu or from the screen menu icon. 


Notes

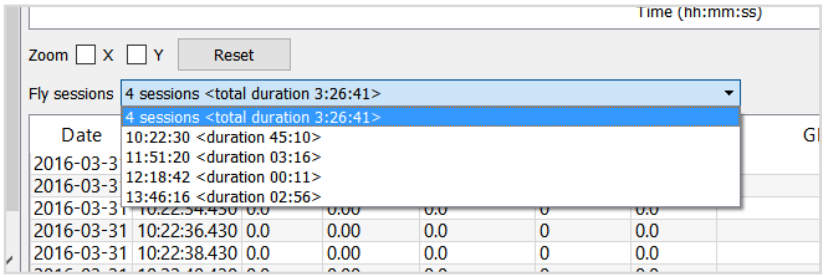
- The Google Earth icon in the corner of the screen will only appear if it has been correctly enabled in in the **Companion** settings.
- The data logging also logs the joystick, slider and switch positions on the transmitter. Even if no sensors are fitted it will also log the **RSSI** and receiver battery voltage. This latter ability is very useful with a new model to fully test that the receiver aerials are functioning well in all flight conditions.

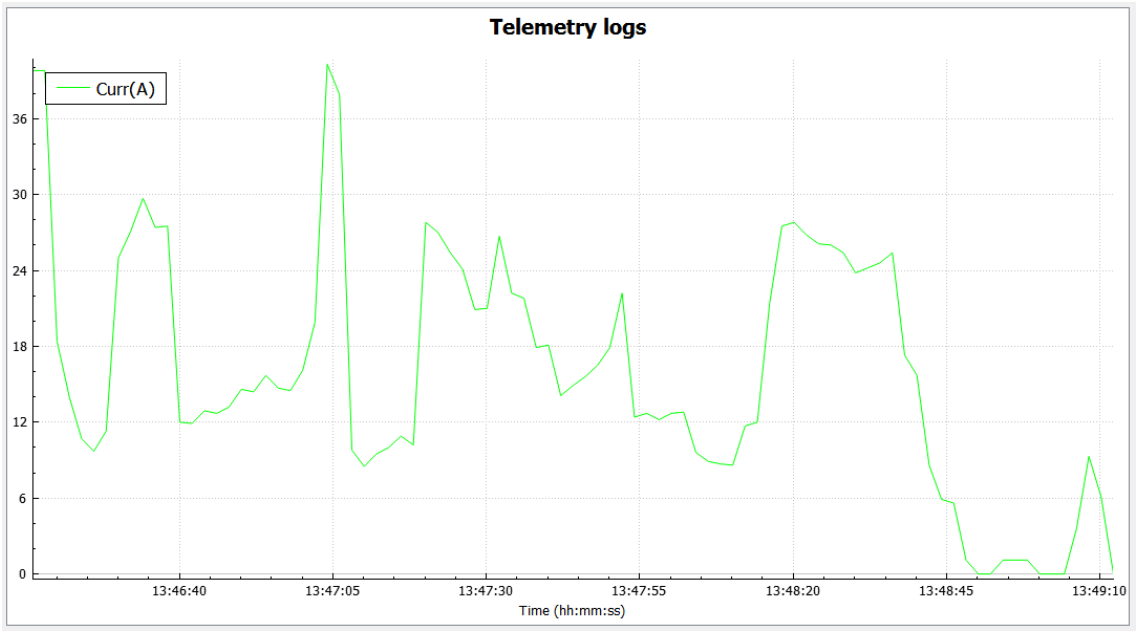
There are a number of points to note with the data logger.

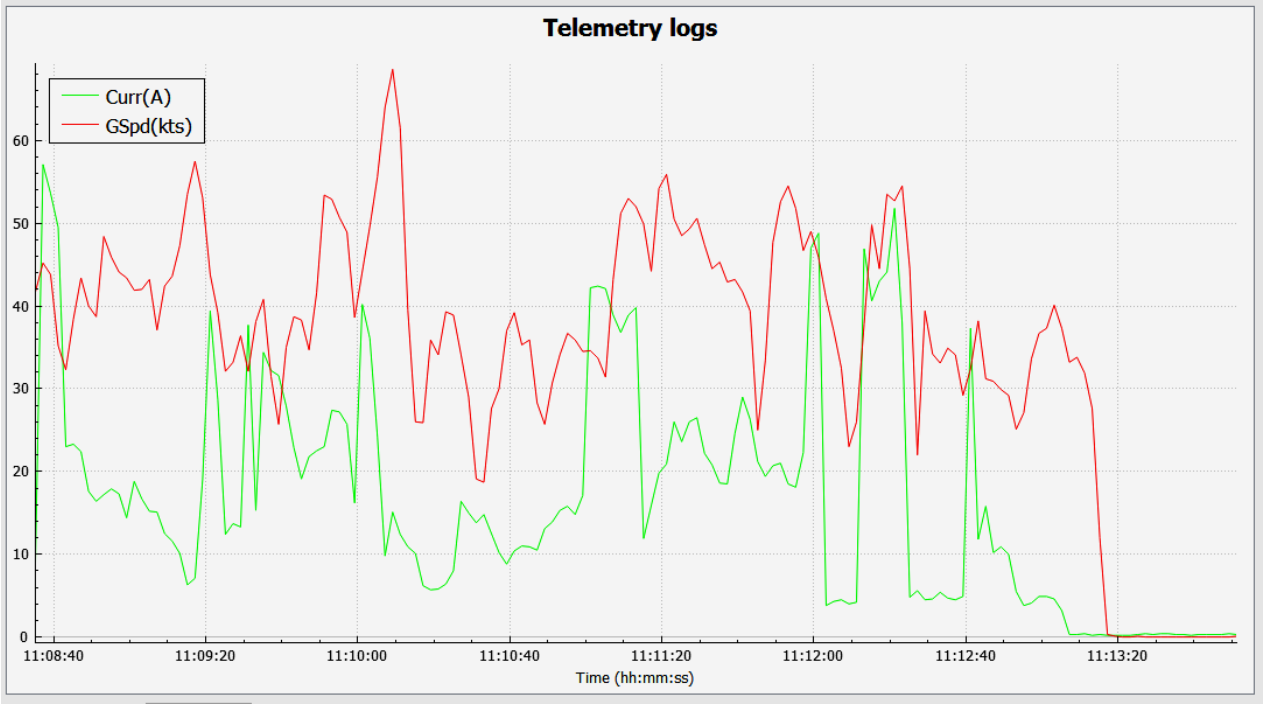
The logger will tend to log all the data. Then you will get a screen like this:




instead, load a single session instead of the whole file which lasted nearly 3 and a half hours.



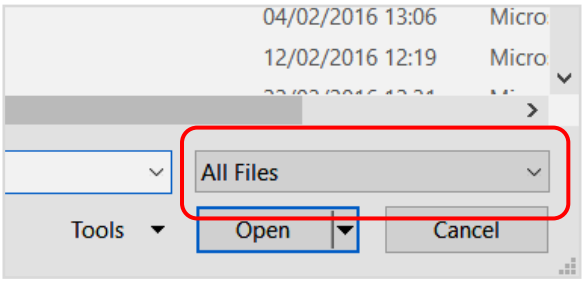



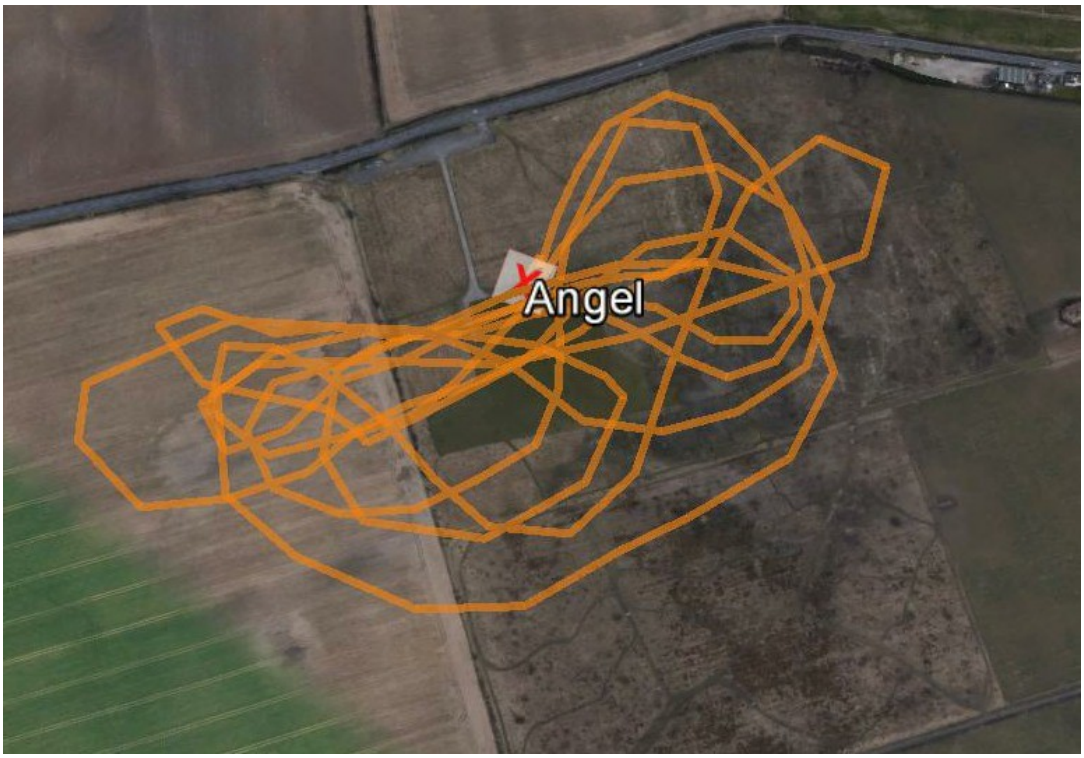



- Several data fields can be viewed together by holding down the CTRL key on the keyboard whilst selecting another data field.
- By highlighting just some of the lines of data it is possible to remove the parts of the log at the start and end of the flight where the plane is on the ground. The part in blue has been highlighted:

| Date | Time | Curr(A) | RSSI(dB) | RxBt(V) | GSpd(kts) | Cels(gRe) | GPS | GAlt(ft) | Date | Cmin(V) | Cspn(mA |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 2016-04-10 | 11:13:45.280 | 0.3 | 93 | 5.6 | 0.0 | 19.21 | 123.4127W 5434.5969N | 146.6 | 16-04-10 11:19:31 | 3.80 | 2226 |
| 2016-04-10 | 11:13:47.280 | 0.3 | 90 | 5.6 | 0.0 | 19.21 | 123.4127W 5434.5969N | 147.6 | 16-04-10 11:19:31 | 3.80 | 2226 |
| 2016-04-10 | 11:13:49.280 | 0.4 | 81 | 5.6 | 0.0 | 19.21 | 123.4127W 5434.5969N | 148.6 | 16-04-10 11:19:31 | 3.83 | 2226 |
| 2016-04-10 | 11:13:51.280 | 0.3 | 84 | 5.6 | 0.1 | 19.23 | 123.4128W 5434.5969N | 149.6 | 16-04-10 11:19:31 | 3.82 | 2226 |
| 2016-04-10 | 11:13:53.280 | 0.3 | 82 | 5.6 | 0.1 | 19.26 | 123.4128W 5434.5969N | 149.9 | 16-04-10 11:19:31 | 3.83 | 2226 |
| 2016-04-10 | 11:13:55.280 | 0.3 | 0 | 5.6 | 0.1 | 19.26 | 123.4128W 5434.5969N | 149.9 | | 3.83 | 2227 |
| 2016-04-10 | 11:13:57.290 | 0.3 | 0 | 5.6 | 0.1 | 19.26 | 123.4128W 5434.5969N | 149.9 | | 3.83 | 2227 |
| 2016-04-10 | 11:13:59.290 | 0.3 | 0 | 5.6 | 0.1 | 19.26 | 123.4128W 5434.5969N | 149.9 | | 3.83 | 2227 |

- If necessary, data logging files can be edited in Microsoft Excel. They are in CSV format and should be saved the same way. When loading, it is necessary to select **All Files** in the Excel file load window as shown opposite in order to see the logs.






Provided one has GPS logging data, clicking on the Google Earth button will automatically plot the flight. The name of the plane is shown, in this case Angel, and the flight path recorded. Ample proof for the club Safety Officer that I did not fly over the road with my plane! By altering the Google Earth perspective, different views can be obtained.