Telemetry

It's a bit like buying a dishwasher. Whenever someone says to me that he or she is thinking of doing so for the first time I always say, 'Only buy one if you are sure you will always be able to afford, or find room for, one. There is nothing worse than being forced to do without one again.' It's the same with telemetry. Once you have used it you will never want to be without it. If you have never used it you won't understand what the fuss is all about. After all, who minds drying up? Well, my hand shot up (dropping the wet cup in the usual tactical way). This article is about telemetry in general but FrSky is particular.

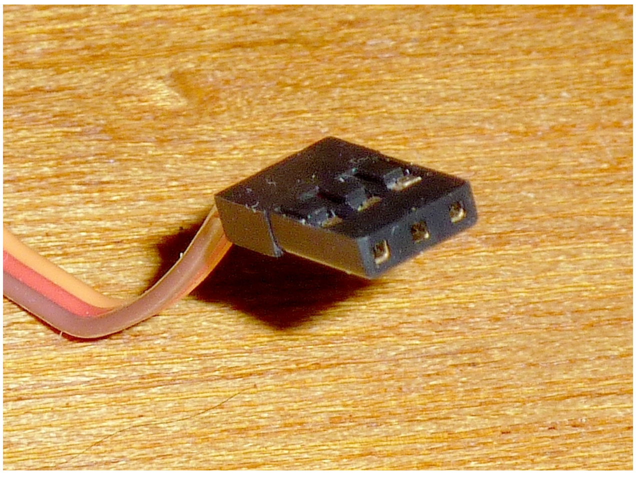
When I restarted flying about six years ago I bought a Spektrum DX6 because it was a make I had heard of. Then I read a series of articles in the UK magazine Radio Control Models and Electronics about OpenTx and FrSky ('Freesky'). I wanted flaps on a new glider, but the DX6 only had switched additional channels and I wanted a rotary or slider control. The FrSky Taranis has two of each.

So the DX6 was put on a shelf and it's still there. Any offers? I soon learned how open and flexible the Taranis was and got to like Amber's voice. But the biggest leap was telemetry. At modest cost and weight FrSky allows you read altitude, climb rate, GPS position, ground speed, air speed, motor current, motor rpm, receiver and battery voltages, ESC and other temperatures, liquid fuel consumed, mAh used and signal level received at the receiver. Yes, all of that. Sensors mostly weigh only a few grams and cost from £10 for the lipo voltage sensors to about £45 for the air speed indicators.

Other makes will have some, or even all, of the sensors. Having read here what FrSky can do, you can then compare and contrast the others for size and cost. FrSky has a big range of transmitters but I think the X9D offers the best value for money and is very flexible. If you are thinking of buying one you can read my free downloadable pdf manual on peterscott.website/flying.

The telemetry sensors run on a bus system similar to S.BUS rather than needing a hub like other makes. They connect in a daisy chain, connecting one to the next and so on, using female to female servo-type leads that finish in a single SmartPort port on the receiver (Rx). Note that I use 'female' here in the correct electronics meaning of metal sockets not metal pins in the connectors. An example is in picture 1. Some suppliers describe these as 'male to male'.

Picture 1



Female servo connector

Photo Peter Scott

OpenTx telemetry

OpenTx is the operating system that FrSky uses on the Taranis. It is open source so is written by volunteers and is free. The more I learn about it the more impressed I am. Setting up telemetry and designing the display screens is simple and as you learn more you can make wider use of the great strength, accuracy, reliability and flexibility of the system. To help with the hardware costs of the FrSky project I send a modest annual donation.

Real and virtual sensors

Confusingly the same name 'sensor' is used for both the real physical devices and the data they produce.

Real/physical sensors (electronics)

These are the actual electronic devices. Each has two IDs. The first is an ID for the device and has four digits, e.g. F210. This is unique and can't be changed. The second is the ID used by the Rx to identify the sensor, with a value between 1 and 32. This must be unique within a model setup. The default values can be changed. You might never need to do this, but if for example you want to monitor the rpm and temperature of two motors and ESCs in one model the second sensor must have a different ID.

Physical sensors produce one or more data. The system calls each datum a 'virtual sensor'.

Virtual sensors (data)

There are actually two types:

- Data produced by the real sensors
- Data calculated from one or more other data

For clarity I will use the words 'datum/data' rather than the term 'virtual sensor(s)'. Each datum can be reset individually using a special function and a real or logical switch, or you reset all at the same time using a Tx function.

1 Data produced by the real sensors – 'custom' data

Each datum has a name, unit and value. These can be changed using the **Edit** function (more below). One datum can be displayed more than once, for example to display height in both metres and feet, or cubits if you are really traditional.

2 'Calculated' data

Amongst other things, data can be added, averaged or multiplied, and the minimum or maximum of up to four data can be found. To do this you click **Add a new sensor**. This opens an edit screen. You give the data a name then set **Type** to **Calculated**. For example the battery's voltage multiplied by the motor current gives the power. Don't forget the calculated data are still called 'virtual sensors'. Editing is described later.

Using data

You can see data on one of the four telemetry screens, hear it read out to you at chosen intervals or use it as a logical switch to trigger an action - a function. You could even use it as an input, for example to change elevator trim settings depending on airspeed or to open an air intake to cool an overheating motor or ESC.

Editing a real or virtual sensor

Move to the screen that lists the telemetry sensors.
Select the sensor to edit or create a new one.
Long press on **ENT** and select **Edit**.
Type: Choose **Custom** or **Calculated** and click **ENT**

This is the list of things you can edit:

If **Custom** is selected:
Name: Change it if more than one of the same sensor.
Type: Reads **Custom**
ID: First is fixed. Don't change the second if already sensed unless it's a duplicate.
Unit: Use this to select, for example, m or ft for height.
Precision: Sets the number of decimal places. Useful for when value is read out by Amber.
Ratio: Multiplier to use to get the correct reading, e.g. for a voltage checked by voltmeter.
Offset: Sets initial value to non-zero, e.g. for GAlt when airfield not at sea level.
Positive: Ignores negative values, e.g. no vario sound when sinking.
Persist.: This tells the transmitter to retain the value from previous sessions, e.g. Aspd+ would give maximum ever airspeed for that model.
Logs: This tells the transmitter to store the data stream on your memory card.

If **Calculated** is selected some are different:
Type: Reads **Calculated**
Formula: Choose from **Add, Average, Min, Max, Multiply, Totalize, Cell, Consumpt, Distance.** (See more details below)
Source1 to 4: The lists will contain all of the data items currently in telemetry
Filter: Sveraging smooths values that change a lot, e.g. lipo voltage
Auto Offset: This sets the first received value as the zero when reset or at switch-on.

Formulae explained

Add, Average, Min, Max and Multiply are obvious.
Totalize: This adds the sensor value to running total
Cell: The value of an individual cell is found using:
- **Cell Index:** Number showing position in cell series
- **Lowest:** Obvious
- **Highest:** Obvious
- **Delta:** Highest minus lowest voltage

Consumpt: mAh used using current sensor data
Distance: Distance between the pilot and the model using GPS

Telemetry sensors with default IDs and data items

Sensor	ID	Data items
Air speed indicators (2 models)	10	Aspd
Current sensors (2 models)	3	VPAS (40A only), Curr
Liquid fuel monitoring	22	Temp 1 & 2, RPM, Volume remaining, Flow rate, Max flow, Average flow
GPS	4	GPS, Galt, Gspd, Hdg, Date, Time
Lipo voltage sensors (3 models)	2	Cels
RPM and temperatures	5	RPM, Tmp1, Tmp2 (see note below)
Receiver	25	RSSI, RxBt, SWR
Variometers (2))	1	Alt, Vsp
Neuron ESCs	17	EscV (volts), EscA (current), EscR (RPM), EscC (consumpt), EscT (ESC temperature)

Plus a sensor hub with sockets for even more compact sensors for:

- Fuel Gauge
- GPS
- Variometer
- Lipo Voltage
- Temperature
- Power Supply
- RPM Sensor
- Triaxial Accelerometer

For an RPM sensor you must set up the number of coil pairs on the sensor edit screen.

SP2UARTs

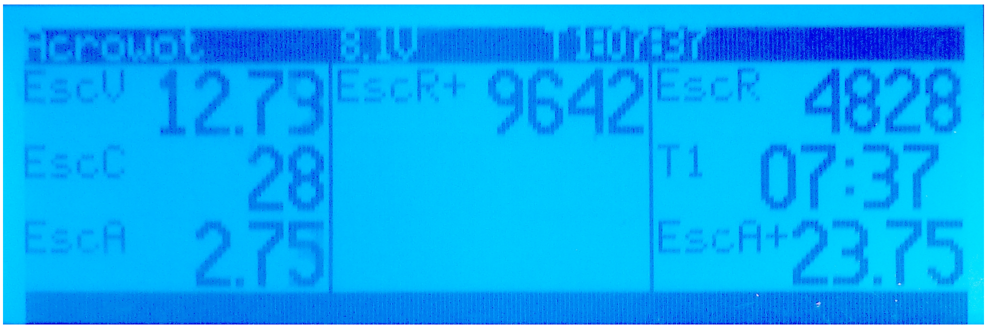
SP2UART (universal asynchronous receiver/transmitter) devices are like modems. You need a pair, one in the Rx and the other connected to a read/write device. They can transmit and receive analogue and RS232 serial data at up to 9600 baud (that takes you back a bit doesn't it?) but usually at no more than 300 baud. Yes 300 bit/s. I could imagine it being used in robotics but I can't yet see a use in model aircraft. The data they produce are called A3 to A6.

A3/A4 SP2UART host	ID = 6
A5/A6 SP2UART remote	ID = 7

Telemetry screen design

The FrSky Taranis allows four different telemetry screens each of which is blank until you fill it with your required data. Screen design is very easy. You move to the screen design page and first decide whether you want the screen to contain numbers or bar displays. With numbers (Pictures 2 and 3) you step to each of the nine positions and select the datum you want for that position from a list. On the bars screen you can only select three data (Picture 4), in this case just one.

Picture 2: Numbers

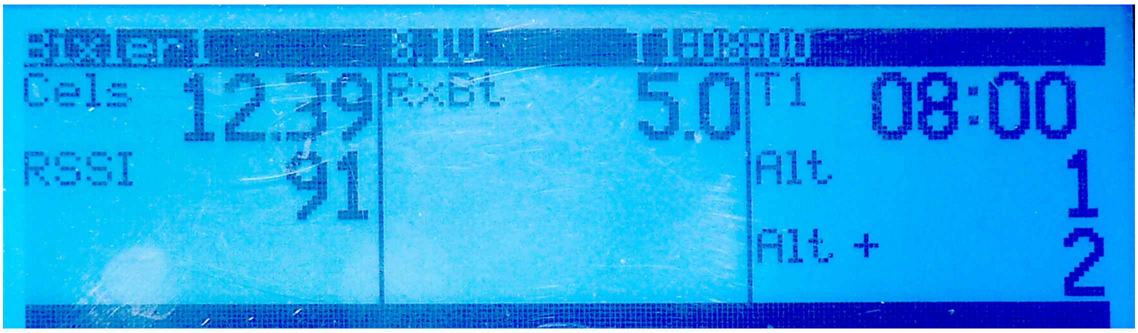


```
Acrowot 8.1V
EscV  12.73 | EscR+ 9642 | EscR    4828
EscC     28 |            | T1     07:37
EscA   2.75 |            | EscA+ 23.75
```

Photo Peter Scott

In Picture 2 you can see that I have throttled up to 9642 rpm where the current was 23.75 A. Then back to low with a current of 2.75 A and rpm 4828. In playing about to take this picture I have used 28 mAh of the 2200 so the lipo still has 12.73 V. Remember that the + means the maximum for that flight. The weird names for the data are because here I am using a FrSky Neuron ESC about which more later.

Picture 3: Numbers

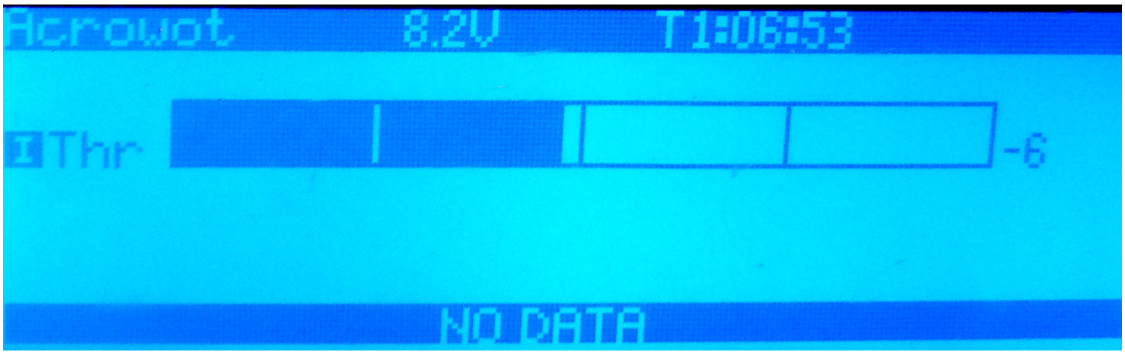


```
Cels  12.39 | RxBt   5.0 | T1  08:00
RSSI     91 |            | Alt     1
            |            | Alt +   2
```

Photo Peter Scott

Picture 3 is the screen for a simple glider not using a Neuron. The lipo voltage in this case comes from a separate sensor. The altitude data comes from a vario zeroed on my living room floor. RSSI means received signal strength indication, which is the signal seen by the receiver. This makes range checking very easy. 100 means full signal.

Picture 4: Bars
For completeness here is the alternative bars screen.



```
Acrowot          8.2V      T1:06:53

Thr                                   -6

               NO DATA
```

Photo Peter Scott

Standard telemetry screen design

As you see from above, up to now I have used different screens for different models. That can be confusing, so I decided to adopt a standard screen layout. This was my first draft:

For live flight data:	Live flight data	Flight record	Ground test
Plus voice message when a critical level reached and no **EscC** available	Battery voltage **EscV**	Maximum rpm **EscR+**	Rpm **EscR**
Plus voice message when at 30% mAh	mAh used **EscC**	Maximum current **EscA+**	Current **EscA**
Plus voice message every 10 seconds	Altitude **Alt**	Maximum power **Watt+**	Power **Watt**

The natures of the data are shown in standard weight letters. The names of the data on my systems are shown in bold. I fly mode 2 so it is safer to move my left hand to read the live flight data if I have to. They are in any case read out to me. Where a datum is not available such as altitude on an aerobatic power model the box will be left blank.

Telemetry voice messages

Glancing down at a screen when flying can be risky - less so for gliders than aerobatic models. Mostly I set up the Tx to speak the key data for me. Once the battery has reached about 30% charge I have the mAh used read out. For flying on a field with a height restriction I have altitude above ground level spoken every ten seconds, though I could get the message read out only when I reach 122 m. If I land and the voice says something like ‘minus two metres’ I know that atmospheric pressure is rising and the weather next day might again be good for flying.

FrSky Neuron ESC

FrSky sells a range of clever electronic speed controllers for 3S to 6S batteries. Called ‘Neuron’ they come in 40, 60 and 80 capacities with 50% more momentary current. They have a voltage adjustable 7A SBEC. The reason I mention them here is that they have built-in telemetry sensors for voltage, current, RPM, power consumption and temperature which plug into the receiver SmartPort. They are compact with good finned heat sinks. There is much more about these devices below.

The ESC comes packed in foam in a solid plastic box. It is very strongly made with a thick aluminium plate top and bottom. All sides are open. There are two servo-style ports to connect to the receiver – one for the throttle/BEC lead (PWM) and the other for the telemetry Smartport (S.Port). You need leads with both ends female, and must cut the red core on the PWM lead if using a separate receiver battery, but this is a separate lead so you are not cutting one that is permanently connected to the ESC. All versions are the same size and weight, though different prices, and can be connected to 3S to 6S batteries.

After soldering on XT90 and 4mm sleeved bullet connectors, the Neuron 60 device weighed 73g. This is exactly the same as a Turnigy Plush 60A, though of course the latter has no telemetry. The sizes are:
Neuron: 60 x 33 x 16 mm
Plush: 72 x 30 x 17 mm
So the Neuron is just a bit shorter. Other ESCs are available.

Clearly this is also designed with racing quads in mind, as it is marked BLheli. However I use it on fixed wing. The Neuron 60 is shown in Pictures 5 and 6.

Picture 5

Photo Peter Scott

Picture 6

Photo Peter Scott

I tested the telemetry using a Taranis X9D plus transmitter running OpenTx V2.2.2 and an X8R receiver. Initially no motor was connected, so the current, rpm and mAh consumption data were zero. I allowed the ESC to power up the receiver through the BEC and the voltage shown in RxBt was 4.9V. Using a voltmeter I checked whether this was the voltage sent by the BEC and it was, so the BEC appears to default to 5V. It can provide 7A. The voltage can be changed.

After using ‘Discover new sensors’ on the Taranis all of the data appeared as follows:

Names	Values during test.		
0E50	2560804	This stores encrypted BEC values whatever that means.	
EscV	16.71V	Lipo voltage	Volt
EscA	0.00A	Motor current	Amp
EscR	0rpm	Motor speed	RPM
EscC	0mAh	Consumpt – mAh used	mAh
EscT	44°C	ESC temperature	Temp

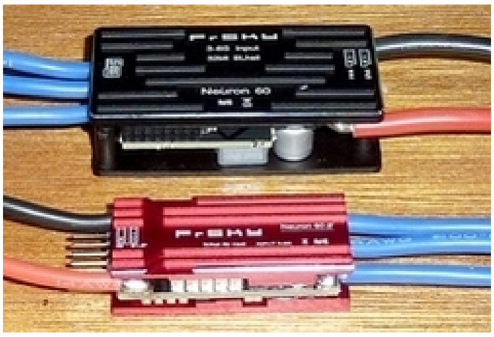
As you see the data (sensors) have different names from the ones created by separate telemetry devices. The default EscX ones seem sensible but if you want you can change them such as to those shown in the last column above. The only advantage is that the telemetry screen might be more understandable. All have the same device ID (17). If using two of these in a twin motor model you will have to change the device ID for one set of sensors.

I then added the new data to a numeric telemetry screen which worked perfectly. I added EscA+ and EscR+ so that I could have a reading of the maximum current and motor speed during the flight. These are essential to make sure that I have the correct propellor fitted. To get maximum safe power I want about 90% of the maximum current for the motor when the prop is unloaded in the air.

Then I connected a sizeable 4Max motor and ran it up, hand held, without a propellor. Sensors Amp, RPM, mAh now generated data. The RPM one showed over 30000 rpm which puzzled me until I remembered that RPM has to be calibrated for the number of coils in the motor, and defaults to one. I edited the sensor to the 12 coils for the motor. The defaults appear to work fine for fixed wing.

Neuron S versions

In late 2019 FrSky issued updated versions of the Neurons, designated ‘S’ (small?). I bought a 60S and a 40S. They are very much smaller as you can see from Pictures 7 and 8, which compare the 60 and the 60S. FrSky has done an amazing job squeezing the speed control circuitry and the telemetry into such a small device. The S has a jumper to select whether the BEC is used so you don’t need to cut the red core on the throttle lead. One last bonus was that the throttle did not need calibrating. It presumably defaults to 1000 to 2000 ms.

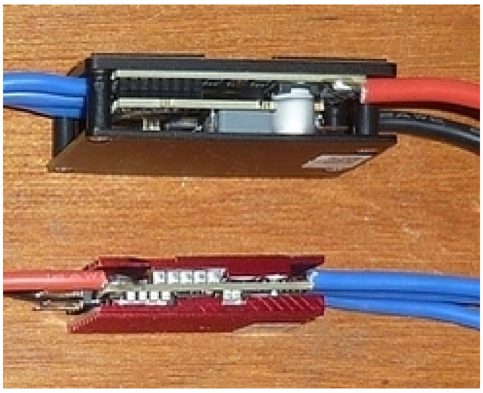


Picture 7

Photo Peter Scott

60	76g	60 x 33 x 16 mm
60S	47g	45 x 22 x 12 mm

In both cases weights are with one XT90 and three bullet connectors



Picture 8
Photo Peter Scott

I installed a 40S in my Acrowot foam-e. It fitted perfectly in the original position. I now don’t need to connect the battery balance lead to a voltage sensor for voltage telemetry so the wiring is much neater. And now I know when I’ve used 1500 of the 2200 mAh in the battery ‘cos Amber tells me. I used the default 5V BEC on the 40S.

I fitted the 60S in my Wot4. It was much smaller and lighter than the Turnigy Plush ESC I was using so I was able to put it in a more convenient place, which left room for a NiMH receiver battery and switch.

As always, in both cases the telemetry data was found by the Taranis without a problem.

Choosing a propellor

This is an area where the Neurons excel. Using RPM, power and current you can experiment with different props until you get the optimum size for maximum safe power and current for a given battery.

I now use Neurons as standard in all of my new models. All I then need in the gliders is a variometer. I could of course buy a FrSky receiver with a vario in it, but I prefer to stick to good old RX8R ones and FrSky varios weight next to nothing. Ain’t technology wonderful?

A word of warning

I have found that the S versions of Neurons are more fragile electronically. If you misconnect one it is likely to burst into flames so check very carefully before powering up.

I have no connection, financial or otherwise, with the makers or distributors of the items mentioned in the article.

Last edit 5 July 2022