User guide

Overview

FreeTX is an open-source radio control system you can use for just about anything. It combines features from different RC systems and brings in new ideas too. It’s perfect for creators, hobbyists, and engineers. The main goal is to give you more freedom and choice with what you want to do.

Default configuration

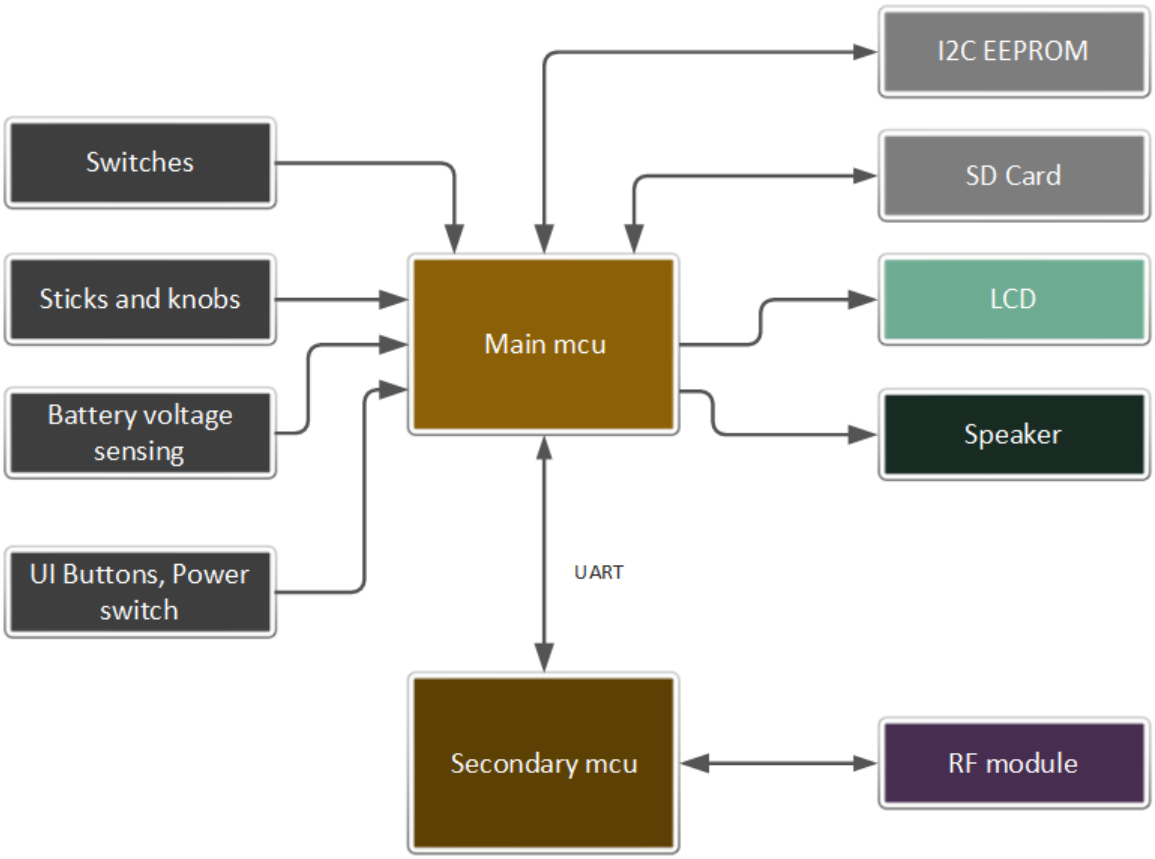
By default, the system is configured with the following features.

- 10 stick input axes, 8 switches, 2 knobs
- 40 mixer slots, 20 proportional RC channels
- 10 custom curves, 10 max points per curve
- 20 logical switches, 5 function generators, 3 timers, 5 counters
- 10 custom notifications
- 5 flight modes
- 8 telemetry sensor slots

Many of these numbers are easily customisable in the source code, the limit being the available memory and I/O pins.

System architecture

Hardware block diagram



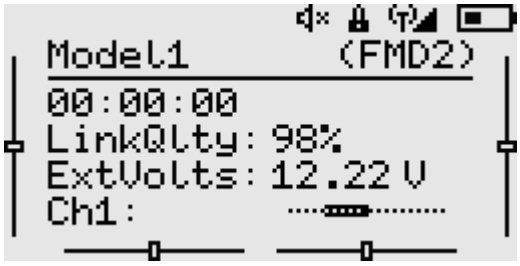
RF protocol

A custom RF protocol is used to communicate between the transmitter and receiver. The implementation builds on top of LoRa(R) transceivers. The documentation can be found in the Protocols folder.
The main features are:-

- 10 bit encoding for all RC channels
- 50 Hz update rate for all RC channels
- Frequency Hopping Spread Spectrum
- Telemetry support

Home screen

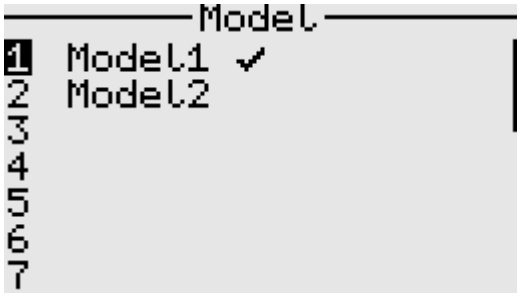
The home screen is the default screen shown when operating the transmitter. It displays basic information such as the active model name, trims, status icons, transmitter battery, active flight mode, and up to four custom widgets.

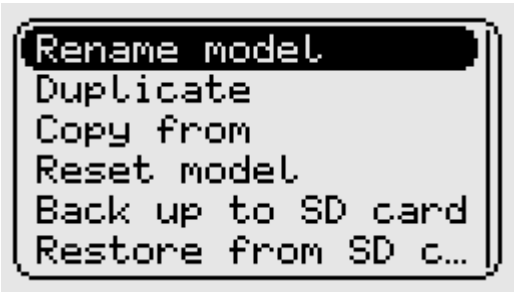


Key actions on the home screen are as follows:

- Down key
  - Click: Opens the context menu.
  - Hold: Quickly mutes/unmutes telemetry alarms.
- Up key
  - Click: Opens the channel monitor.
  - Hold: Enters the on-screen trim mode on handsets that lack physical trim buttons.
- Select Key
  - Click: Opens the main menu.
  - Hold: Navigate back. This is also the default behaviour in all screens.

Model manager





The model manager enables various actions, such as creating, loading, and deleting models. By default, model settings are stored in the EEPROM.

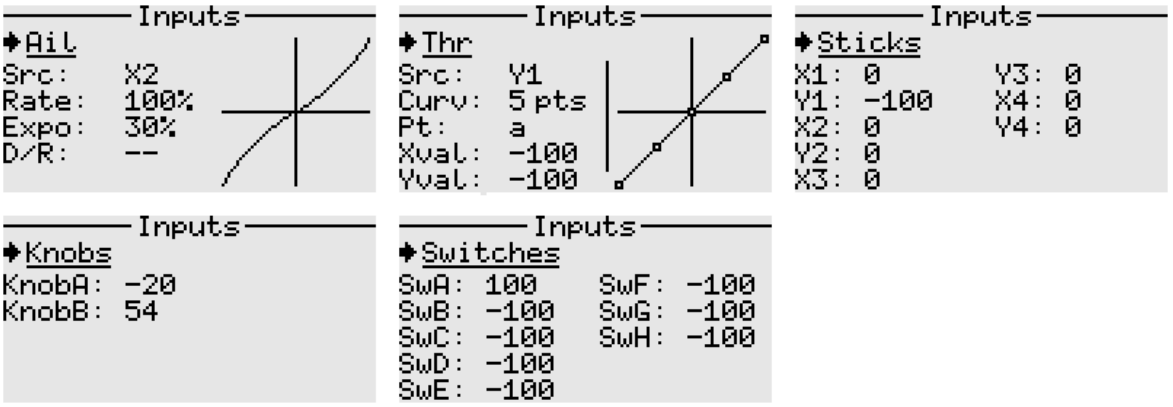
SD cards are also supported and are useful for backing up and restoring models. Model data on the SD card is stored in a human-readable text format similar to YAML.

Note:

1. An external EEPROM is recommended but not mandatory. However, without it, the system will only have one model slot available, which makes it cumbersome to switch between different models, as we would always need to rely on the SD card to load a different model.
2. Before updating the firmware in the transmitter, always back up your models to the SD card to avoid any potential data loss or corruption.
3. The "Copy from" feature only works between models of the same type.
4. Loading or selecting a new model disables RF output for safety reasons; therefore, RF output must be manually re-enabled afterward.

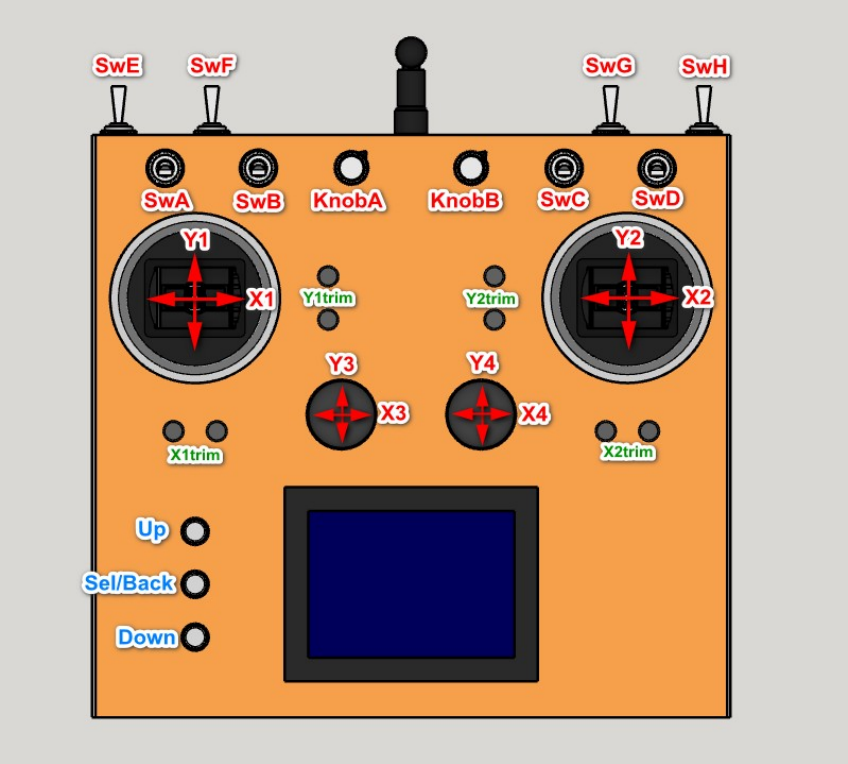
Inputs

The inputs screen allows setting dual rates and expo, throttle curve, as well as showing the values from the physical controls.



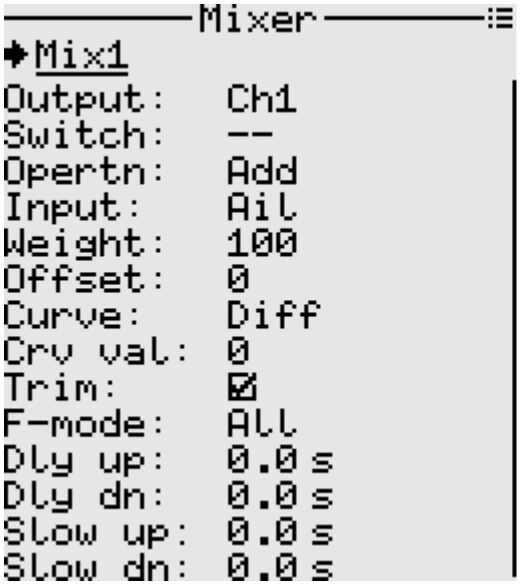
- For model type 'Airplane', the naming used for the main controls is Ail, Ele, Rud, Thr.
- For model type 'Multicopter', the nomenclature changes to Roll, Pitch, Yaw, Thr.
- The sources for the above inputs may default to a different order based on the chosen stick mode in System settings.
- The system accepts both 2-position and 3-position switches, as well as their momentary variants.

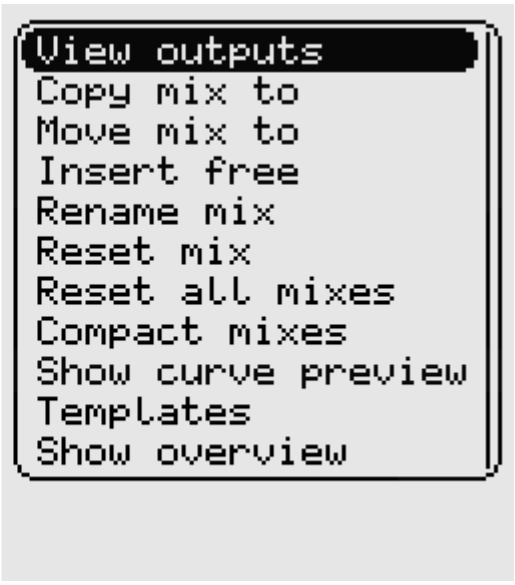
The image below shows a typical layout of the transmitter.



The Mixer

The mixer is where the inputs (mixer sources) get linked to the output channels (servos, etc.). This controller features a mixer system that draws inspiration from that found in OpenTX(R) based RC transmitters, though not directly compatible.





Mixer sources

Mixer sources can be any of the following:

- Raw stick axes (X1, Y1, X2, Y2, ...)
- Processed sticks (Rud/Yaw, Thr, Ail/Roll, Ele/Pitch)
- Knobs (KnobA, KnobB)
- Constants (Max)
- Function generators (Fgen1, Fgen2, ...)
- Trims (X1Trim, Y1Trim, X2Trim, Y2Trim)
- Physical switches (SwA, SwB, ...)
- Logical switches (L1, L2, ...)
- Channels (Ch1, Ch2, ...)
- Virtual channels (Virt1, Virt2, ...)

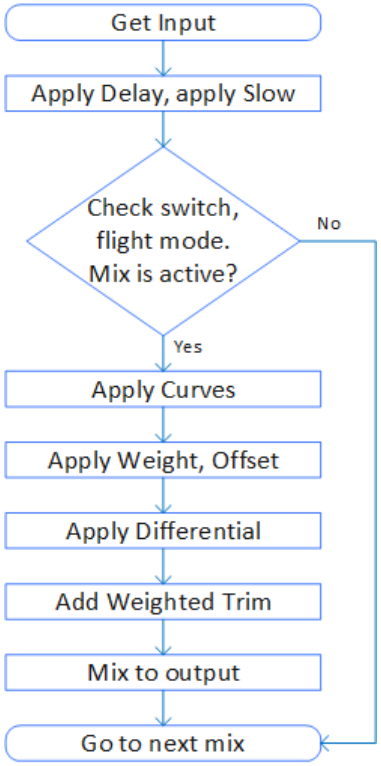
Counters can also be used as inputs in the mixer, although they are not classified as true mixer sources.

Mixer fields

- Output: The channel that is to be affected by the mix. If none, the mix is inactive and is not computed.
- Switch: The control switch to turn on or off the mix. The mix is always active if no is switch specified.
- Operation: Add, Multiply, Replace, Hold.
- Input: This is the mixer source.

- Weight: Determines how much of an effect the input has on the output. -100 to 100.
- Offset: The input is offset by this value after the weight has been applied. -100 to 100.
- Curve: Differential, Expo, Function, Custom curve.
- Curve value: Differential {-100 to 100}, Expo {-100 to 100}, Function {x>0, x<0, |x|}, Curve name if custom curve.
- Trim: Whether to add trim or not.
- F-Mode: The flight mode in which the mix applies.
- Delay Up and Down: This is used to delay the input. 0 to 60 seconds.
- Slow Up and Down: This when non-zero specifies how fast the input changes. 0 to 60 seconds.

Order of mixer operations



Important safety note

When editing mixes, it is recommended to disable RF output or remove propellers first.
This is because all changes/adjustments made are applied instantly and may bring surprises.

Example mixes

Note:

- The convention used in these examples is -ve for left/downward going control surfaces, +ve for right/upward going control surfaces. For example if the Ail stick is moved right, the left aileron moves down and the right aileron moves up. Hence we use a negative weight for the left aileron and a positive weight for the right aileron.

- If a servo is moving in wrong direction, reversing the direction in Outputs screen solves this.
- The Mixer is all about control logic. Any mechanical differences should be handled in the Outputs screen and not inside the Mixer.

Example 1: Basic 4 channel

Assuming Aileron servo in Ch1, Elevator servo in Ch2, Throttle in Ch3, Rudder servo in Ch4

```
1. Ch1  Add  Ail (Weight 100)
2. Ch2  Add  Ele (Weight -100)
3. Ch3  Add  Thr (Weight 100)
4. Ch4  Add  Rud (Weight 100)
```

Example 2: Elevon

Left servo in Ch1, right servo in Ch2

```
1. Ch1  Add  Ail (Weight -50)
2. Ch1  Add  Ele (Weight -50)
3. Ch2  Add  Ail (Weight 50)
4. Ch2  Add  Ele (Weight -50)
```

Example 3: Vtail

Left servo in Ch2, right servo in Ch4

```
1. Ch2  Add  Rud (Weight 50)
2. Ch2  Add  Ele (Weight -50)
3. Ch4  Add  Rud (Weight -50)
4. Ch4  Add  Ele (Weight -50)
```

Example 4: Aileron differential

Left aileron in Ch1, right aileron in Ch8. We want the downward going aileron to get less displaced than the upward going aileron. Due to differences in lift and air pressure on the top and bottom wing surfaces, the aileron surface produces more drag when it moves down than when it moves up. This tends to cause an adverse yaw effect hence the need for aileron differential.

```
1. Ch1  Add  Ail (Weight -100, Diff 30)
2. Ch8  Add  Ail (Weight 100, Diff 30)
```

Example 5: Tailerons

Left Elevator servo in Ch6, Right Elevator servo in Ch7. The mix is turned on/off with SwG.

```
1. Ch6  Add  Ele (Weight -60)
2. Ch6  Add  Ail (Weight -40, Switch SwG_Down)
3. Ch7  Add  Ele (Weight -60)
4. Ch7  Add  Ail (Weight 40, Switch SwG_Down)
```

Example 6: Flaperons

Left aileron in Ch1, right aileron in Ch8. Using SwC to operate the flaperons.
When SwC is in upper position, flaperons are off. In middle position half flaperons, and in lower position full flaperons.
For smooth flaperon deployment, we use the slow up/down feature.

```
1. Ch1  Add  Ail (Weight -100, Diff 30)
2. Ch1  Add  SwC (Weight -40, Offset -40, SlowUp 1 s, SlowDown 1 s)
3. Ch8  Add  Ail (Weight 100, Diff 30)
4. Ch8  Add  SwC (Weight -40, Offset -40, SlowUp 1 s, SlowDown 1 s)
```

Example 7: Crow braking

Left ail servo in Ch1, Right Ail servo in Ch8, left flap servo in Ch5, right flap servo in Ch6. Using the three position SwC for operation. Normal aileron action occurs when SwC is in upper or middle position. Half flaps are deployed when SwC is in middle position. When SwC is in lower position, both ailerons move upward and full flaps are deployed.

```
1. Ch1   Add  Ail (Weight -100, Diff 30)
2. Ch1   Add  SwC (Weight 60, Func x>0, SlowUp 1 s, SlowDown 1 s)
3. Ch8   Add  Ail (Weight 100, Diff 30)
4. Ch8   Add  SwC (Weight 60, Func x>0, SlowUp 1 s, SlowDown 1 s)
5. Ch5   Add  SwC (Weight -50, Offset -50, SlowUp 1 s, SlowDown 1 s)
6. Ch6   Add  Ch5 (Weight 100)
```

Example 8: Flaps working as ailerons

This mix allows the entire trailing edge of the wing (aileron and flap) to operate as ailerons.
Assuming the left flap servo in Ch5, right flap servo in Ch6.
Using a two position switch e.g. SwD to switch between flap mode and aileron mode.
Using the three position SwC to set the flap position in flap mode.

```
1. Ch5   Add    SwC (Weight -50,  Offset -50)
2. Ch5   RplcW  Ail (Weight -100, Diff 20, Switch SwD_Down)
3. Ch6   Add    SwC (Weight -50,  Offset -50)
4. Ch6   RplcW  Ail (Weight 100,  Diff 20, Switch SwD_Down)
```

Example 9: Differential thrust

Left motor in Ch3, right motor in Ch7. We can use a switch e.g. SwE to turn the differential thrust off while in the air. Also assume the Rud source is the X1 axis

```
1. Ch3  Add  Thr (Weight 100)
2. Ch3  Add  X1  (Weight 40, Switch SwE_Down)
3. Ch7  Add  Thr (Weight 100)
4. Ch7  Add  X1  (Weight -40, Switch SwE_Down)
```

For safety, we also need to set Failsafe for both Ch3 and Ch7 in the 'Outputs' screen.
We can also add throttle cut by specifying an override switch and a value of -100.

Example 10: Throttle elevator mixing

When the throttle is increased, some down elevator can be added to counteract the tendency of the model gaining altitude when the throttle increases.

```
1. Ch2  Add  Ele (Weight -100)
2. Ch2  Add  Thr (Weight -20, Offset -20)
```

Example 11: Variable steering depending on throttle position

We want the steering response to reduce with increasing throttle. Assuming we use the X1 axis to steer, Y1 axis as our throttle source, and the steering servo is on Ch6,

```
1. Ch6  Add    X1 (Weight 100)
2. Ch6  Mltply Y1 (Weight -40, Offset 60)
```

Example 12: Adjust idle throttle with knob

Suppose we have a gasoline-powered model and want to adjust the idle setting of engine with the knob without affecting full throttle. Assuming the throttle servo is in Ch3,

```
1. Ch3   Add     KnobA (Weight 20, Offset 20)
2. Ch3   Mltply  Thrt  (Weight -50, Offset 50)
3. Ch3   Add     Thrt  (Weight 100)
```

Example 13: Adjust maximum throttle with knob

We can adjust the maximum throttle without affecting the low throttle setting. Assuming the motor is in Ch3,

```
1. Ch3   Add     KnobA (Weight 20, Offset -20)
2. Ch3   Mltply  Thrt  (Weight 50, Offset 50)
3. Ch3   Add     Thrt  (Weight 100)
```

Example 14: Simple throttle cut

Assuming Ch3 is the throttle channel, and assigning SwA for cut. When SwA is in the up position, Ch3 is locked to -100, otherwise the Throttle input is sent to Ch3.

```
1. Ch3  Add    Thrt (Weight 100)
2. Ch3  RplcW  Max  (Weight -100, Switch SwA_Up)
```

Alternatively, a much safer throttle cut can be achieved in the Outputs screen by specifying the override as SwA and value -100.

Example 15: Sticky throttle cut

Assuming Ch3 is the throttle channel, and assigning SwA for cut. Also assume our Throttle source is Y1 axis. We want the throttle cut whenever SwA is in the up position, but to only remove the cut when the throttle is at minimum and SwA is in the down position.
This example makes use of logical switches as follows.

```
L1
Func:   Latch
Set:    SwA_Up
Reset:  L3

L2
Func:   a==x
Value1: Y1
Value2: -100

L3
Func:   AND
Value1: SwA_down
Value2: L2
```

Then in the mixer

```
1. Ch3  Add    Thrt (Weight 100)
2. Ch3  RplcW  Max  (Weight -100, Switch L1)
```

Alternatively, a much safer throttle cut can be achieved in the Outputs screen by specifying the override as L1 and value -100.

Example 16: Landing gear sequencer (gear doors stay open after gear is extended)

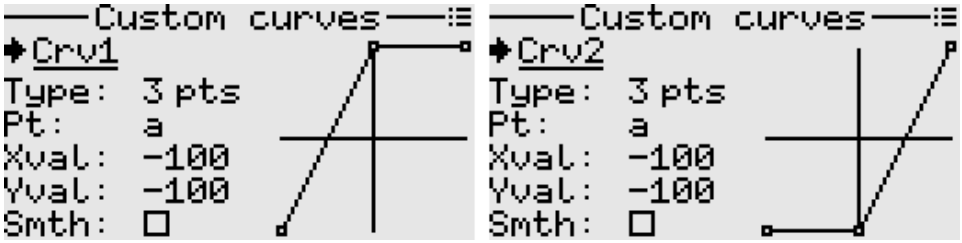
The sequence we want upon flicking the switch is: doors open, gear extends. Similarly in the opposite direction: gear retracts, doors close. Using SwD for operation and assuming gear doors on Ch6 and gear on Ch7.

```
1. Ch6  Add  SwD (Weight 100, DelayUp 0 s, DelayDown 3 s, SlowUp 2 s,
SlowDown 2 s)
2. Ch7  Add  SwD (Weight 100, DelayUp 3 s, DelayDown 0 s, SlowUp 2 s,
SlowDown 2 s)
```

Alternative method that uses custom curves

Set up two custom curves as follows.

```
Curve1
Type: 3 point
Point(x,y): PtA(-100,-100) PtB(0,100) PtC(100,100)

Curve2
Type: 3 point
Point(x,y): PtA(-100,-100) PtB(0,-100) PtC(100,100)
```



Then in the mixer, assuming gear door servos on Ch7, gear servos on Ch8, and using SwD to operate,

```
1. Virt1 Add  SwD   (Weight 100, SlowUp 5 s, SlowDown 5 s) //Create a
source that slowly ramps
2. Ch7   Add  Virt1 (Weight 100, Custom Curve1)           //Apply curve1
for the door sequence
3. Ch8   Add  Virt1 (Weight 100, Custom Curve2)           //Apply curve2
for the gear sequence
```

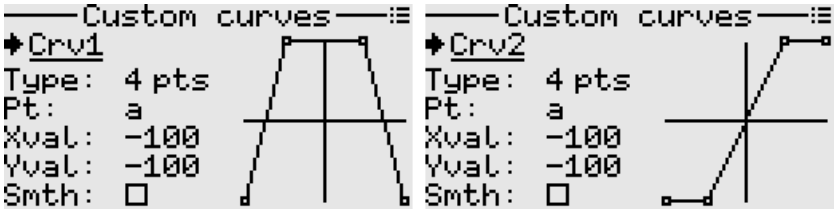
Example 17: Landing gear sequencer (gear doors close after gear is extended)

The extend sequence is doors open, gear is lowered, doors close.
The retract sequence is doors open, gear retracts, doors close.
We can achieve the above using custom curves in combination with a slowed switch.

Servo version

Set up two custom curves as follows.

```
Curve1
Type: 4 point
Point(x,y): PtA(-100,-100) PtB(-50,100) PtC(50,100) PtD(100,-100)

Curve2
Type: 4 point
Point(x,y): PtA(-100,-100) PtB(-50,-100) PtC(50,100) PtD(100,100)
```



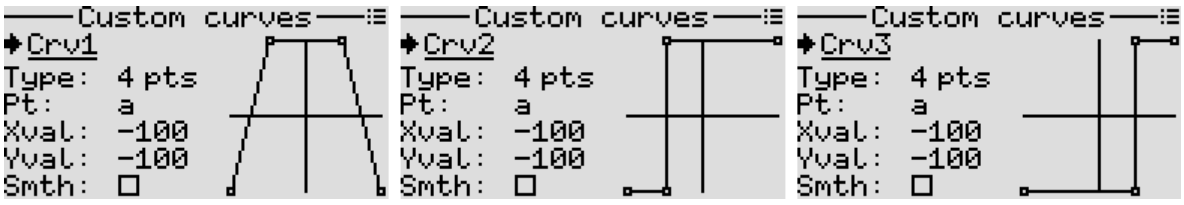
Then in the mixer, assuming gear door servos on Ch7, gear servos on Ch8, and using SwD to operate,

```
1. Virt1 Add  SwD   (Weight 100, SlowUp 7 s, SlowDown 7 s) //Create a
source that slowly ramps
2. Ch7   Add  Virt1 (Weight 100, Custom Curve1)           //Apply curve1
for the door sequence
3. Ch8   Add  Virt1 (Weight 100, Custom Curve2)           //Apply curve2
for the gear sequence
```

Retract actuator version

Most electric retract actuators require an on/off control signal, so this example is designed to work with these.
Set up three custom curves as follows.

```
Curve1
Type: 4 point
Point(x,y): PtA(-100,-100) PtB(-50,100) PtC(50,100) PtD(100,-100)

Curve2
Type: 4 point
Point(x,y): PtA(-100,-100) PtB(-50,-100) PtC(-50,100) PtD(100,100)

Curve3
Type: 4 point
Point(x,y): PtA(-100,-100) PtB(50,-100) PtC(50,100) PtD(100,100)
```



Then in the mixer, assuming gear door servos on Ch7, gear retracts on Ch8, and using SwD to operate,

```
1. Virt1 Add   SwD   (Weight 100, SlowUp 10 s, SlowDown 10 s)   //Create a
source that slowly ramps
```

```
2. Ch7   Add   Virt1 (Weight 100, Custom Curve1)                //Apply
curve1 for the door sequence
3. Ch8   Add   Virt1 (Weight 100, Custom Curve2)                //Apply
curve2 for the gear extend sequence
4. Ch8   RplcW Virt1 (Weight 100, Custom Curve3, Switch SwD_Up) //Apply
curve3 for the gear retract sequence
```

We can, if necessary, set Ch8 to 'Digital' mode in the receiver output configuration screen.

Example 18: Servo tester

We can program a simple servo tester that repeatedly moves a servo back and forth.
In the extras menu, navigate to Function generator and select the desired movement type.
Then in the mixer, assuming the servo is connected to Ch8,

```
1. Ch8  Add  Fgen1 (Weight 100)
```

We can also make it more interesting by adding the ability to switch between waveforms or patterns via a switch. This is left as an exercise.

Example 19: Random servo motion generator

Suppose we have a model equipped with a turret, or a figure. We want the servo to move automatically but in a random and intermittent manner.
We can achieve this with function generators and logical switches as follows.

```
Fgen1
Waveform: Random
Interval: 2.0 s

Fgen2
Waveform: Random
Interval: 1.3 s
L1
Func:   a>x
Value1: Fgen2
Value2: 0
```

Then in the mixer, assuming our servo is on Ch7,

```
1. Ch7   Add   Fgen1 (Weight 100)
2. Ch7   Hold        (Switch L1)
3. Ch7   RplcW Ch7   (Weight 100, SlowUp 2 s, SlowDown 2 s)
```

Explanation:
Fgen1 generates a random position periodically, Fgen2 via Logical switch L1 is used to hold the output randomly to create the impression of intermittent motion. The last mixer line helps smooth the movement.
If desired, we can also incorporate a custom curve to sort of bias the generated servo positions, for example if we want our turret be pointing around the middle most of the time.
We can also easily modify the setup to include an extra switch to manually stop the motion.

Example 20: Random servo motion generator (advanced version)

This example generates a random servo position at a random interval, with an added effect of random intermittency, and also random selection between two speeds.
Furthermore, a physical switch SwE is used to start/stop the random motion of the servo.

```
Fgen1
Waveform:      Random
Interval mode: Variable
Min Interval:  1.0 s
Max Interval:  2.0 s
Control:       Fgen2

Fgen2  (modulates the interval of Fgen1)
Waveform:      Random
Interval mode: Fixed
Interval:      1.0 s

Fgen3  (used for the intermittent motion)
Waveform:      Random
Interval mode: Fixed
Interval:      1.0 s

Fgen4  (used for the selection between two speeds)
Waveform:      Random
Interval mode: Fixed
Interval:      2.0 s
```

Then set up two logical switches as follows.

```
L1
Func:   a>x
Value1: Fgen3
Value2: 0

L2
Func:   a>x
Value1: Fgen4
Value2: 0
```

Then in the mixer, assuming our servo is on Ch7,

```
1. Ch7   Add    Fgen1 (Weight 100)
2. Ch7   Hold         (Switch L1)
3. Ch7   RplcW  Max   (Weight 0, SwE_Up)
4. Ch7   RplcW  Ch7   (Weight 100, SlowUp 5 s, SlowDown 5 s, Switch L2)
5. Ch7   RplcW  Ch7   (Weight 100, SlowUp 2 s, SlowDown 2 s)
```

Example 21: Oscillation without using the built-in function generator

Here is a way to create a crude oscillator from scratch.
Set up a custom curve that defines the shape of our arbitrary waveform. For example,

```
Curve1
Type: 5 point
Point(x,y): PtA(-100,0) PtB(-50,-100) PtC(0,0) PtD(50,100), PtE(100, 0)
```

```
Smooth: True
```

Then define a logical switch as follows.

```
L1
Func:   a<x
Value1: Virt1
Value2: 100
```

Then in the mixer, we make use of the 'slow' feature on the L1 mixer input. This essentially creates a sawtooth waveform on Virt1. Assuming our final output is on Ch1, we specify the input as Virt1 and apply the custom curve.

```
1. Virt1  Add  L1    (Weight 100, SlowUp 1 s, SlowDown 0 s)
2. Ch1    Add  Virt1 (Weight 100, Custom Curve1)
```

Example 22: Aileron rudder mixing with automatic switching

When the Rudder stick is centered and the Aileron stick is moved away from center, the mix is activated (Aileron stick now controls the rudder).
The mix is deactivated as soon as the Rudder stick is moved away from center (Rudder stick now controls the Rudder).
When both sticks are centered the mix gets activated again.
We can achieve this behaviour with the help of logical switches as follows.

```
L1
Func:   a==x
Value1: Rud
Value2: 0

L2
Func:   a==x
Value1: Ail
Value2: 0

L3
Func:   AND
Value1: L1
Value2: L2

L4
Func:   Latch
Set:    L3
Reset:  !L1
```

Then in the mixer, assuming the rudder servo is on Ch4,

```
1. Virt1 Add     Max   (Weight 100, Switch L4)
2. Ch4   Add     Ail   (Weight 100, No trim)
3. Ch4   Mltply  Virt1 (Weight 100, SlowDown 1 s)
4. Ch4   Add     Rud   (Weight 100)
```

Here, Virt1 is used to smooth the transition from the Aileron stick to the Rudder stick.

Example 23: Adjust aileron differential with the knob

This example illustrates a method for adjusting aileron differential in-flight, eliminating the need to land whenever we want to tweak the setting. It is quite handy for experimenting with different values and helping with fine tuning.
As we rotate the knob clockwise, the differential increases from 0% to 100%.

Setup a logical switch as follows.

```
L1
Func:   a>x
Value1: Ail
Value2: 0
```

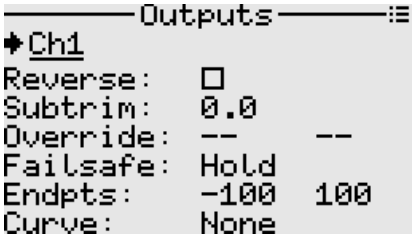
Then in the mixer, assuming the left aileron servo in Ch1, right aileron servo in Ch8

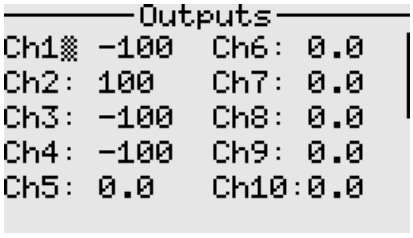
```
1. Ch1   Add     Ail    (Weight -100, No trim)
2. Ch1   Mltply  KnobB  (Weight -50, Offset 50, Switch L1)
3. Ch1   Add     X2Trim (Weight -100)
4. Ch8   Add     Ail    (Weight 100, No trim)
5. Ch8   Mltply  KnobB  (Weight -50, Offset 50, Switch !L1)
6. Ch8   Add     X2Trim (Weight 100)
```

Explanation:
Differential is applied to the downward going aileron. When the Ail stick is moved right, we apply differential to the left aileron. Similarly when the Ail stick is moved left, we apply differential to the right aileron.
We detect the direction of the Ail stick via the logical switch L1 and use it to selectively apply the differential effect via a multiply mix.
For this case, we also decouple the trim and add it afterward, to prevent it being affected by the differential.

Outputs

Outputs are the last stage of the processing chain (Inputs ---> Mixer ---> Outputs).
It is where final adjustments to the control data are made before transmission to the receiver.



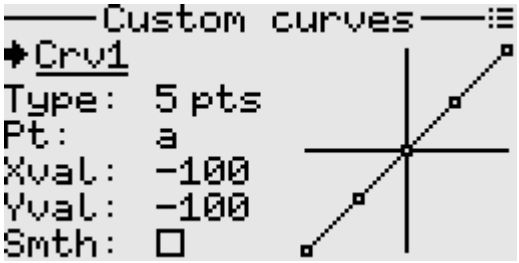


Fields

- Reverse: Useful for correcting the servo or actuator direction relative to the controls.
- Subtrim: Adjusts the center position of the servo or actuator. A side effect of using subtrim is that it causes clipping at the ends. E.g. if the mixer is outputting -100 to 100, a subtrim of say +5 shifts the usable range of the servo to -95 to 100.
- Override: This is mainly a safety feature used for throttle cut, etc. It overrides the channel's output with the specified value when the activation switch is on.
- Failsafe: The receiver will output the specified value after about 1 second of signal loss from the transmitter. Supported modes are Hold, No pulse, or custom value.
- Endpoints: Sets the overall travel or endpoints of the servo arm movement. Useful when we want to avoid any potential binding of a linkage or surface.
- Curve: Specifies the custom curve to use for this output. Useful when we want to correct for linkage geometry, mechanical differences, etc.

Custom curves

These can be used in the mixer and the outputs screen. They basically define a relation between an input (along the x) and output (along the y).
Any point/node on the curve can be freely moved around to get the desired shape of curve.
Up to 10 points/nodes can be defined.
The curve can also be optionally smoothed, although this feature should be used sparingly as it is relatively CPU intensive.



Combining two curves for extra points or nodes

If more curve points are needed, a work around is to combine two curves by switching out one for the other in the mixer, with help of a logical switch.

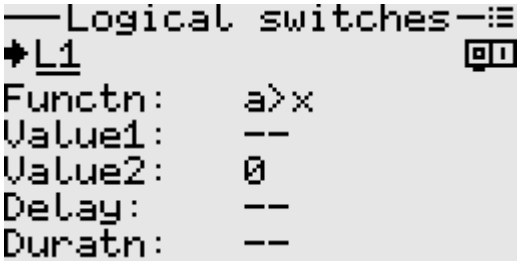
Usage examples

Below are some examples demonstrating the use of custom curves.

- Landing gear sequencer
- Landing gear sequencer - advanced
- Oscillator from scratch

Logical switches

These can be used to combine various conditions, compare values, etc. They are in essence user programmed virtual switches. They produce an on/off output.



The available functions can be categorized into the following types.

- Comparators (Greater than, Less than, Equal, Greater than or equal, Less than or equal)
- Combinational (AND, OR, XOR)
- Latches (SR latch)
- Flip-flops (Toggle)
- Generators (Pulse)

Usage examples

Example 1: A mix activated by combination of two switches

Suppose we want a mix to be turned on only when two switches (e.g. SwA, SwB) are both on. We need to set up a logical switch say L1 as follows.

```
L1
---
Functn: AND
Value1: SwA_down
Value2: SwB_down
```

Then in the mixer, we simply have to set the control switch to L1.

Example 2: LED lights that flash when flight battery is low

Suppose we have a model that is equipped with some LED lights that we want to switch on when the flight battery voltage (telemetry) falls below a certain threshold for more than say five seconds.
We can easily achieve this like so.

```
L1
---
Functn: a<x
Value1: FlghtBatt
Value2: 10.80 V
Delay:  5.0 s

L2
---
Functn: Pulse
Width:  1.0 s
Period: 2.0 s
```

Then in the mixer, we simply use L2 as the input source and L1 as the control switch for the mix, to operate the channel assigned to the LED lights.

Example 3: Stopwatch operated by a single momentary switch for start/stop and reset

When the switch is clicked, the timer runs/stops. When the switch is held long enough, the timer is reset.
For this, we set up two logical switches to operate the timer as below, assuming SwH is our momentary switch.

```
L1
---
Functn: Toggle
Clock:  SwH_down
Edge:   Falling
Clear:  L2

L2
---
Functn: a>x
Value1: SwH
Value2: 0
Delay:  0.5 s

Timer1
------
Switch: L1
Reset:  L2
```

Example 4: Throttle timer (simple)

When the throttle is moved beyond a certain threshold, the timer runs.
Also if we have a throttle cut function on say SwA, we want the timer to only run when throttle cut is inactive. Assume the throttle source is the Y1 axis.

```
L1
---
Functn: a>x
Value1: Y1
Value2: -90

L2
---
Functn: AND
Value1: L1
Value2: SwA_down

Timer1
------
Switch: L2
```

Example 5: Throttle timer (advanced)

Suppose we want the speed at which the timer runs to be proportional to the throttle value.
Also if we have a throttle cut function on say SwA, we want the timer to only run when throttle cut is inactive.
Assume our throttle source is the Y1 axis.

Method 1:

Using the built-in function generator to generate a pulse width modulated signal that controls the timer.

```
Fgen1
-----
Waveform:   Pulse
Width mode: Variable
Modulator:  Y1
Reverse:    False
Period:     1.0 s

L1
---
Functn: a>x
Value1: Fgen1
Value2: 0

L2
---
Functn: AND
Value1: L1
Value2: SwA_down

Timer1
------
```

```
Switch: L2
```

Method 2:

Using the mixer and a virtual channel to create a pulse width modulated signal that controls the timer.

```
Mixer
-----
1. Virt1  Add  L1 (Weight 99, SlowUp 0.5 s, SlowDown 0.5 s)

L1
---
Functn: Pulse
Width:  0.5 s
Period: 1.0 s

L2
---
Functn: a>b
Value1: Y1
Value2: Virt1

L3
---
Functn: AND
Value1: L2
Value2: SwA_down

Timer1
------
Switch: L3
```

If desired, we can also include a custom curve through a virtual channel, to compensate for the real world nonlinear relationship between current draw and throttle value.

Example 6: Zero crossing detection

Suppose we want to detect whenever a value from say a function generator crosses zero, so that we can trigger some other action.

```
L1
---
Functn: a>=x
Value1: Fgen1
Value2: 0

L2
---
Functn: |delta|>x
Value1: L1
Value2: 0
Dirctn: Both
```

L2 will now output a brief pulse whenever the output of the function generator crosses zero.

Example 7: Using a momentary switch to cycle among four values or states

Suppose we want an output channel to cycle among four levels whenever we click the momentary switch.

Method 1:

Set up a '2-bit' binary counter with its outputs on L2 and L1 as follows.

```
L1
---
Functn: Toggle
Clock:  SwH_down
Edge:   Rising

L2
---
Functn: Toggle
Clock:  L1
Edge:   Falling
```

Use a truth table to set up logical switches L3, L4, L5, and L6

L2	L1	
0	0	L3
0	1	L4
1	0	L5
1	1	L6

```
L3
---
Functn: AND
Value1: !L2
Value2: !L1

L4
---
Functn: AND
Value1: !L2
Value2: L1

L5
---
Functn: AND
Value1: L2
Value2: !L1

L6
---
Functn: AND
Value1: L2
Value2: L1
```

Then in the Mixer,

```
1. Ch1  RplcW  Max  (Weight -100,  Switch L3)
2. Ch1  RplcW  Max  (weight -33,   Switch L4)
3. Ch1  RplcW  Max  (weight  33,   Switch L5)
4. Ch1  RplcW  Max  (weight  100,  Switch L6)
```

Method 2:

This makes use of the built-in counter as follows.

```
Counter1
--------
Clock:   SwH_down
Edge:    Rising
Modulus: 4
Dirctn:  Up

L3
---
Functn: a==x
Value1: Counter1
Value2: 0

L4
---
Functn: a==x
Value1: Counter1
Value2: 1

L5
---
Functn: a==x
Value1: Counter1
Value2: 2

L6
---
Functn: a==x
Value1: Counter1
Value2: 3
```

Then in the Mixer,

```
1. Ch1  RplcW  Max  (Weight -100,  Switch L3)
2. Ch1  RplcW  Max  (weight -33,   Switch L4)
3. Ch1  RplcW  Max  (weight  33,   Switch L5)
4. Ch1  RplcW  Max  (weight  100,  Switch L6)
```

Method 3:

This makes use of the built-in counter directly in the Mixer. See the example under counters.

Example 8: Detecting receiver connection or disconnection

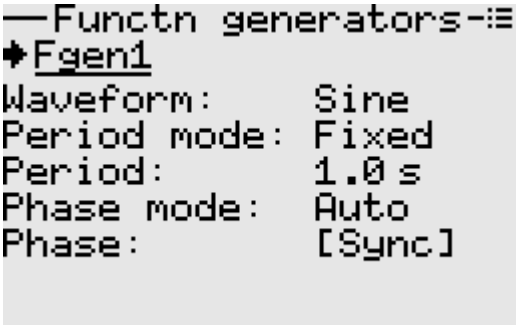
This relies on the Link Quality telemetry to detect connection/disconnection of a receiver.
First, configure the Link Quality telemetry (premade template already available).
Then set up a logical switch and custom notifications as follows.

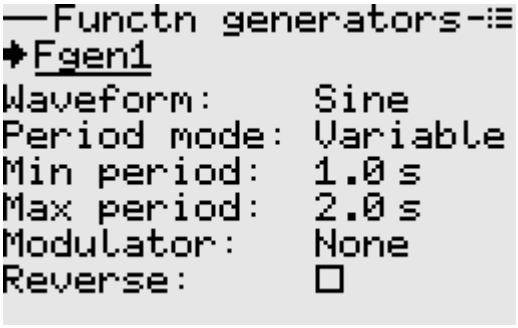
```
L1
---
Functn: a>x
Value1: LinkQlty
Value2: 0

Notification1
-------------
Switch: L1
Tone:   (Your preferred tone)
Text:   RX connect

Notification2
-------------
Switch: !L1
Tone:   (Your preferred tone)
Text:   RX disconnect
```

This example can also be adapted to detect when telemetry from a sensor is lost or recovered.

Function generators

These produce various waveforms over a range of frequencies or periods. They are implemented as low frequency oscillators, hence the reason why period instead of frequency is specifiable.
Supported waveforms are sine, square, triangle, sawtooth, pulse, and random.
The function generators are capable of frequency modulation, with smooth transition between frequencies.





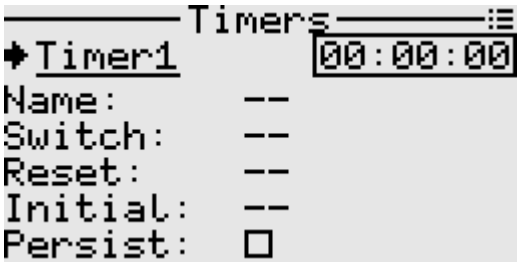
Usage examples

Below are some examples making use of function generators.

- Servo tester
- Random servo motion generator
- Random servo motion generator - advanced

Timers

Timers are used to time events, etc. The system provides up to 3 programmable timers. Timers can be configured to count up or count down.
In count down mode, an audible alarm is played when the timer reaches zero. The timer continues to count with a negative sign.



Fields

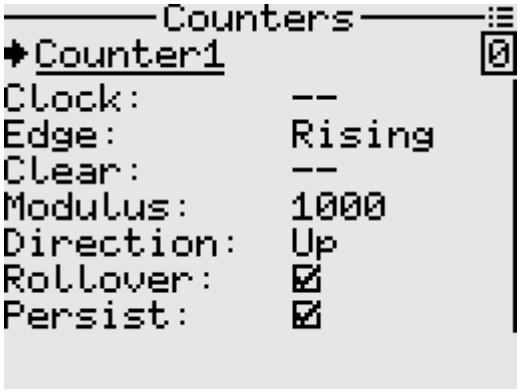
- Name: A name for the timer, otherwise a generic name is used.
- Switch: The control switch used to start and stop the timer. This is optional, as the timer can also be started or stopped from the context menu.
- Reset: The control switch used to reset the timer. This is optional, as the timer can also be reset from the context menu.
- Initial: The initial time if a countdown timer. If not specified, the timer will count up. A time picker dialog is used to easily set the initial time.
- Persist: Whether the timer is persistent i.e. if the value should be remembered between sessions.

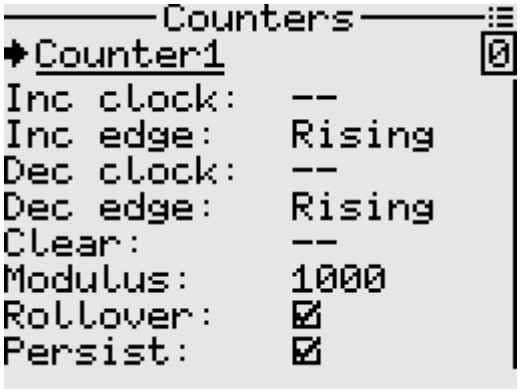
Examples

- Stopwatch operated by a single momentary switch
- Throttle timer - simple
- Throttle timer - advanced

Counters

A counters stores and shows the number of times a particular event has occurred, in relationship to a clock signal.





Fields

- Clock: The control switch that triggers the counter.
- Edge: Rising, Falling, or Dual edge triggering. Applies to the clock signal.
- Clear: The control switch that clears the counter register to zero.
- Modulus: The number of states in its count sequence. E.g. mod 4 produces the sequence 0, 1, 2, 3.
- Direction: Determines whether counts will increment or decrement.
- Rollover: Rolls back over to start counting again from zero.
- Persist: Whether to remember the value in the counter's register between sessions.

In advanced mode, the counter behaves as an up/down counter using separate "up" and "down" clocks for increment and decrement respectively.

Examples

Example 1: Using a 3 position momentary switch to sequentially select among values.

This example demonstrates use of the counter directly inside the mixer.
When the momentary switch is pushed in the downward direction, the counter value increments. Similarly, pushing the momentary switch in the opposite direction decrements the counter value.
In the counter screen context menu, change the counter type to advanced.
Assuming SwG is our 3 position momentary switch, setup the counter as follows.

```
Counter1
---------
Inc clock:  SwG_down
Inc edge:   Rising
Dec clock:  SwG_up
Dec edge:   Rising
Modulus:    4
Rollover:   False
```

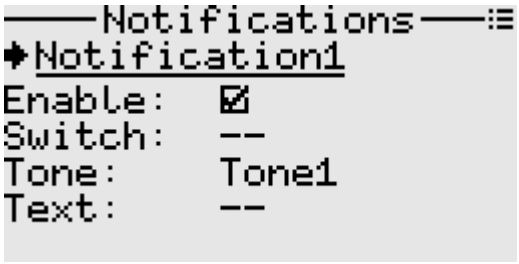
Then in the Mixer,

```
1. Ch1  Add  Counter1  (Weight 100)
```

Note: The mixer automatically maps the counter's value in proportion to the counter's modulus.
For example if the modulus is 4, then mapping is {(0,-100), (1,-33), (2,33), (3,100)}.

Notifications

Notifications are user-programmable and provide audible alarms to report about events.



Fields

- Enable: Whether the notification is enabled or disabled. Only enabled notifications will be sounded and displayed.
- Switch: The control switch used to trigger the notification. The notification gets only triggered on the positive going edge, i.e. when the state of the switch changes from low to high.
- Tone: The tune to play when the notification is triggered.
- Text: Optional text that will be displayed, as a toast message on the screen.

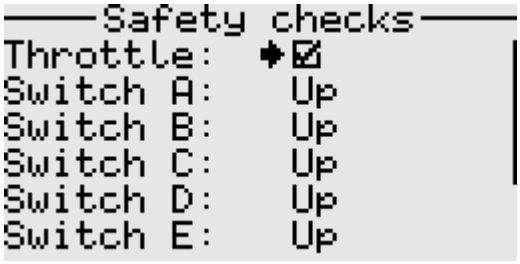
Note: System events like low transmitter battery, inactivity, etc. are natively handled by the system, thus there is no need to configure them inside custom notifications.

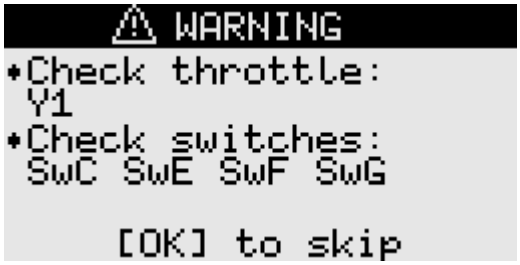
Usage examples

- Detecting receiver connection or disconnection

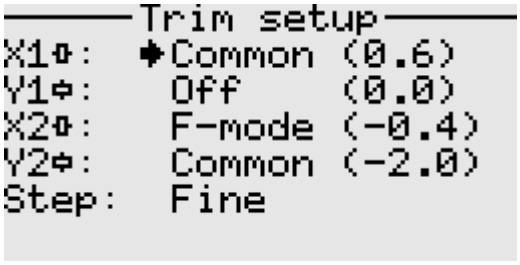
Safety checks

These checks are done before loading a model and on start up.
Positions of switches as well as the throttle source are checked, depending on the setting.
A warning will be shown if the set conditions are not met.





Trim setup



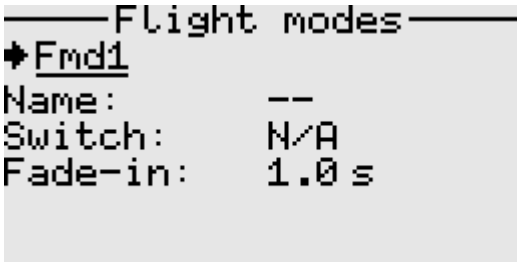
Available states are as follows.

- Common: Applies the same trim value across all flight modes.
- F-mode: Applies independent trim settings for each flight mode.
- Off: Trim is disabled globally for this axis, for example when we do not want trim on the throttle.

The Step setting determines how fast and how fine the trims move. Available options are Coarse, Medium, and Fine.

Flight modes

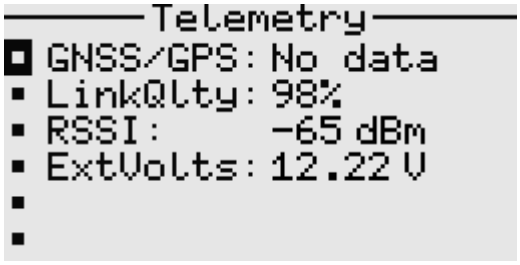
Flight modes allow you to easily reconfigure settings such as mixes and trim values with the simple flip of a switch. While it's possible to achieve similar adjustments using multiple switches, having the ability to make all these changes with just one switch is undoubtedly more efficient and convenient.



Fields

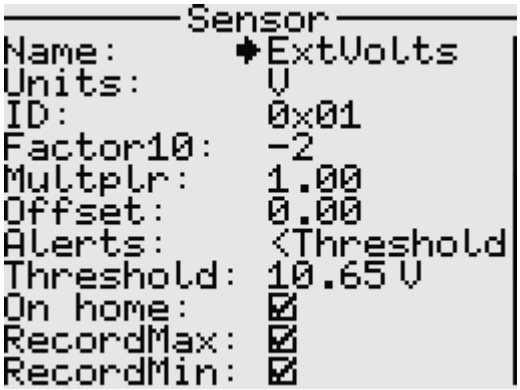
- Name: A name/label for this flight mode.
- Switch: The control switch that activates this flight mode.
- Fade-in: The transition time to effect the trim setting for this flight mode.

Telemetry



General telemetry

The receiver has built-in basic telemetry i.e. external voltage, RSSI, and packet rate. To measure other parameters, you will have to build custom sensors and extend the receiver firmware to support these.
Telemetry values are transmitted as 16-bit signed integers.
To quickly mute/unmute telemetry alarms, long press the Down key on the home screen.

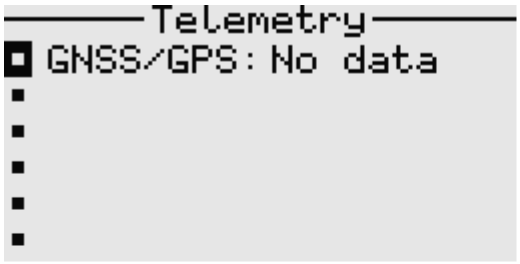


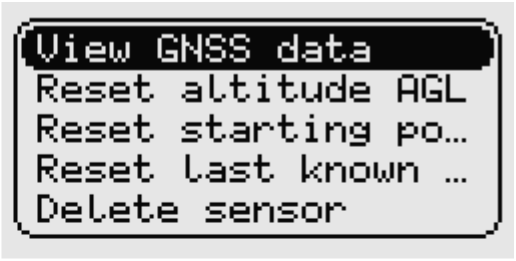
Fields

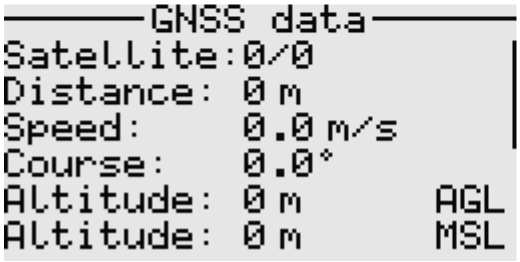
- Name: The sensor name.
- Units: The units of measurement.
- ID: This is the sensor ID, which must be unique. However, the same ID can be used for example to display a value in multiple ways.
- Factor10: The raw value is multiplied by 10 to power this factor. Useful when we want to scale the value.
- Multiplier: The raw value is multiplied by this number. Useful when we want to scale the value.
- Alerts: This parameter specifies how the alarm behaves.
- Threshold: This is the value below or above which the telemetry alarm will be sounded. The telemetry value will also flash when in alarm state.
- On home: Whether to automatically show the telemetry sensor on the home screen.
- Record max, and record min: Whether to track the max/min values received from the sensor. Useful for simple stats. Note that max and min are sensitive to outliers hence the filtering and smoothing should be done before sending the telemetry to the transmitter.

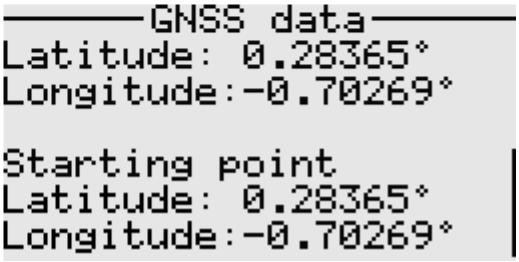
GNSS telemetry

GNSS telemetry is also supported. A premade template is used to add the GNSS sensor to the telemetry screen.
The receiver directly connects to the GNSS/GPS module via serial (UART), and handles parsing of the NMEA sentences as well as data conversion. GPS, GLONASS, BeiDou, Galileo are supported.
The system remembers the last known location in case of a lost model, even when the transmitter is powered off.









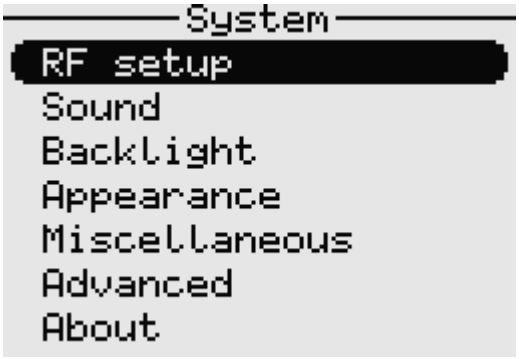
Fields

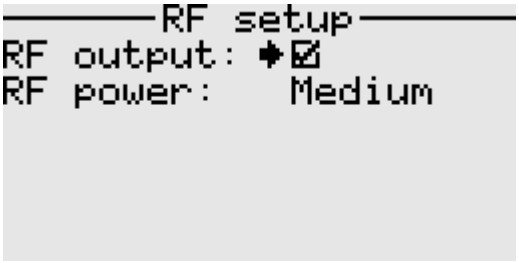
- Satellites: The number of satellites in use / in view. "Fix" is appended when we have a position fix.
- Distance: The calculated distance between the model's current location and the home location (starting point). If the displayed distance is inaccurate, simply reset the starting point from the context menu.
- Speed: The current speed over ground in metres per second. Defaults to 0 when there is no incoming telemetry.
- Course: The current course over ground in degrees. Defaults to 0 when there is no incoming telemetry.
- Altitude AGL: The altitude above ground level in metres. If the value is inaccurate, reset it from the context menu.
- Altitude MSL: The altitude relative to mean sea level in metres.
- Latitude and Longitude: The location coordinates of the model. Displays the last known location when there is no incoming telemetry.
- Starting point: Displays the coordinates of the home location.

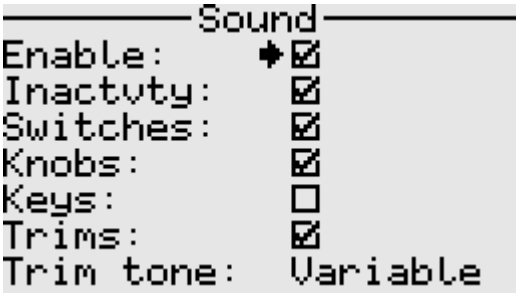
Note: If different units of measurement are desired, use the provided GNSS sensor sub-templates (speed, distance, altitude) and change the multiplier and units.

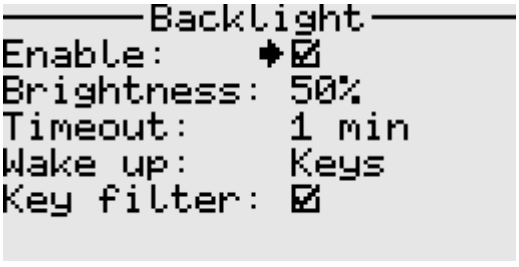
System settings

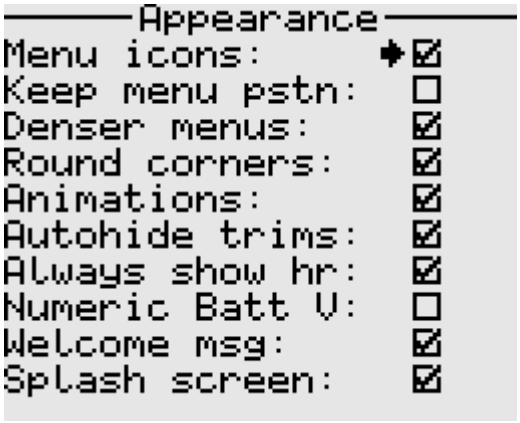
The system menu offers comprehensive configuration options, allowing users to tailor the system to their needs.
Entries are thoughtfully organized into categories for easy navigation and quick access to the desired settings.

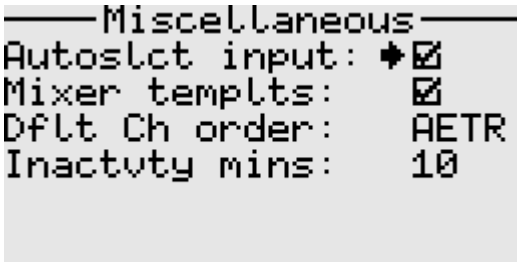


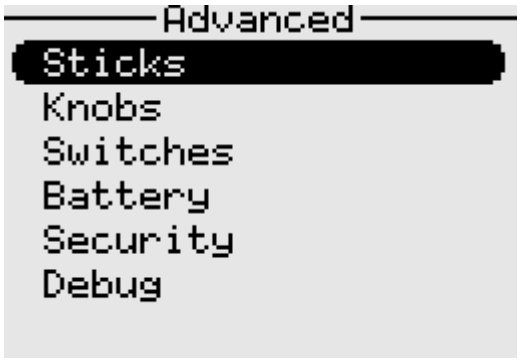


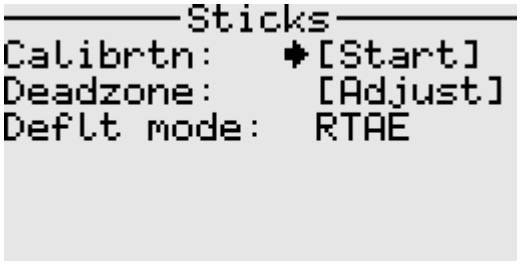


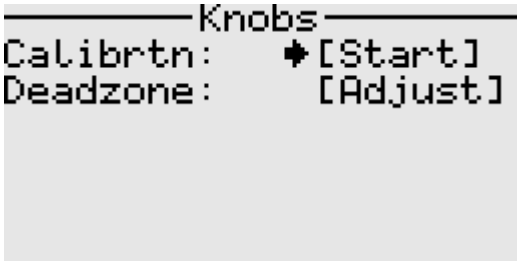


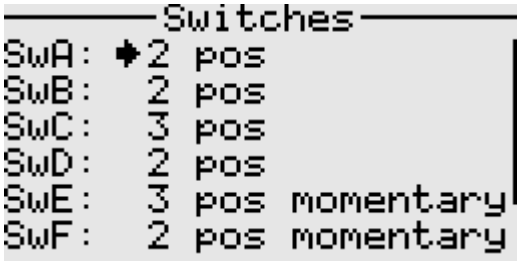

Appearance
Menu icons: ☑
Keep menu pstn: ☐
Denser menus: ☑
Round corners: ☑
Animations: ☑
Autohide trims: ☑
Always show hr: ☑
Numeric Batt V: ☐
Welcome msg: ☑
Splash screen: ☑

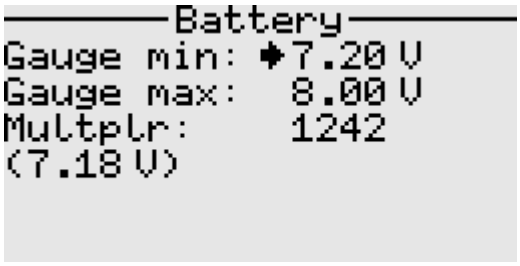

Miscellaneous
Autoslct input: ☑
Mixer templts: ☑
Dflt Ch order: AETR
Inactvty mins: 10

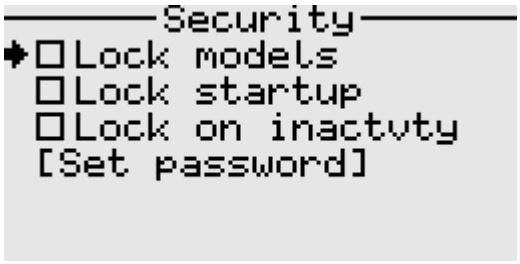

Advanced
Sticks
Knobs
Switches
Battery
Security
Debug

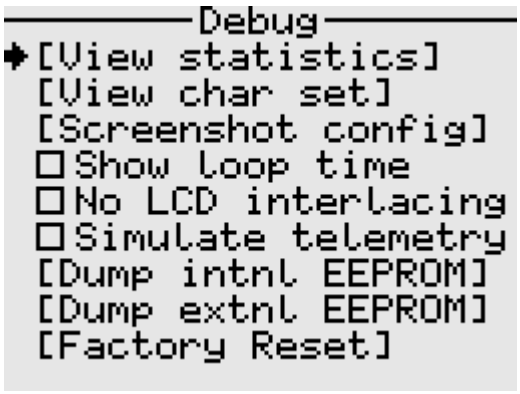

Sticks
Calibrtn: [Start]
Deadzone: [Adjust]
Deflt mode: RTAE


Knobs
Calibrtn: [Start]
Deadzone: [Adjust]


Switches
SwA: 2 pos
SwB: 2 pos
SwC: 3 pos
SwD: 2 pos
SwE: 3 pos momentary
SwF: 2 pos momentary


Battery
Gauge min: 7.20 V
Gauge max: 8.00 V
Multplr: 1242
(7.18 V)


Security
Lock models
Lock startup
Lock on inactvty
[Set password]


Debug
[View statistics]
[View char set]
[Screenshot config]
Show loop time
No LCD interlacing
Simulate telemetry
[Dump intnl EEPROM]
[Dump extnl EEPROM]
[Factory Reset]

Receiver

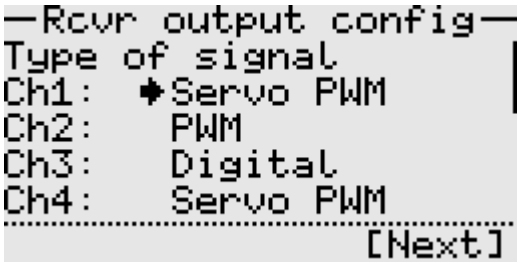
Each receiver supports up to 10 channels. To use all the 20 channels, two receivers are required.

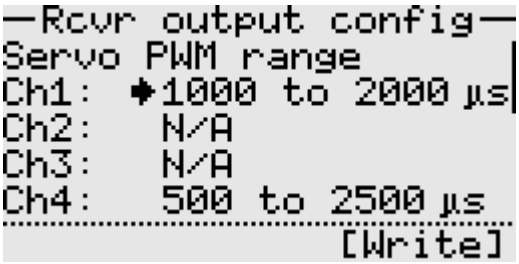
Binding to a receiver

A receiver can be bound as either a main receiver or a secondary receiver.
By default, RC channels 1 to 10 are handled by the main receiver, while RC channels 11 to 20 are handled by the secondary receiver.
To bind the receiver, on the transmitter side access the receiver menu and select the bind option. Then cycle the power to the receiver. If the binding is successful, a "Success" toast message will be shown and a tune played.

Configuring receiver outputs

The receiver outputs can be configured from the transmitter to any of the three signal types; servo PWM, digital on-off, or 'normal' PWM. For example 'normal' PWM output can be used to control brushed DC motors without the need for complex electronics. The digital on-off output makes it easy to directly control components such as electromechanical relays and lights.
The servo PWM range can as well be adjusted to get extra travel from a servo, without making physical modifications to the servo.





Note:

1. These configuration settings are stored in the receiver, not the transmitter.
2. Normal PWM is only available on certain pins, depending on the pin mapping.
3. Any changes made take effect immediately upon pressing the 'Write' button.
4. In digital on-off mode, the output value range of -100 to -50 corresponds to LOW, -49 to 49 is ignored, and 50 to 100 corresponds to HIGH.
5. When reusing a receiver between models, always check and set the appropriate signal format before connecting any actuators to the receiver.