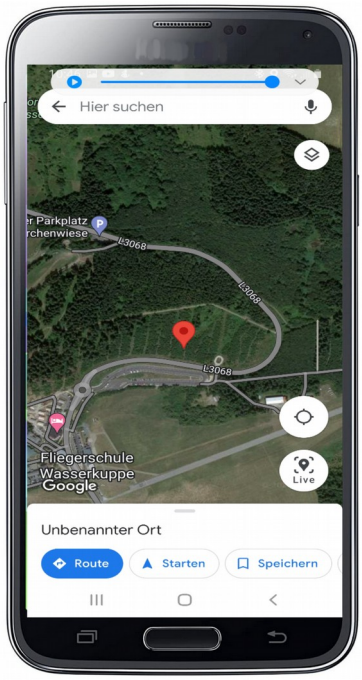
Manual for widget „MODELFINDER“ Rev 1.0

„dark“

„bright“



Inhaltsverzeichnis

Introduction

Modelfinder facilitates the search and location of a model, which is equipped with a GPS:

The last known position is displayed as a QR code in the widget if required.
This QR code is "recorded" using a standard cell phone in photo mode.
Smartphones nowadays automatically recognize the content of the QR code, so the cell phone suggests switching to Google maps.

If this suggestion is accepted, the cell phone starts Google Maps and shows the last known position there.

The widget runs (as of Nov. 2022) on Tandem and Horus transmitters.

Note:

The calculation of the QR codeconsumes much memory & produces heavy CPU Load !
The calculation takes about 6 seconds on a Tandem transmitter, about 9 seconds on a Horus transmitter.
Due to the ETHOS / LUA implementation, no operation on the Ethos interface is possible during this time, even a power off cannot be initiated !

To prevent unnecessary load the QR calculation must be started manually.
A new QR code is then calculated every 30 seconds.

The configuration is done via the Ethos typical widget config menu.

If the transmitter was localized "German" (system setting "display language") German texts are automatically "pulled", otherwise English ones.

You can choose between a "dark" and a "bright" theme.

For practice purposes, a test mode can also be activated there, which simulates a slightly varying GPS reception.

The display has its own "Configs" for X20 & Horus transmitters, X18 should work, as soon as a sim is available, this display can be optimized.

Horus transmitters must have logical switches defined to simulate the "buttons".
(see "Installation")
The assignment of which LSW takes over which button function can be done in the widget config.

Note: "Source for QR Code Calculation":

The source to create a QR Code is very complex and was provided via OpenSource:
https://github.com/speedata/luaqrcode

I would like to thank the speedata developer team, without their work this widget would not have been possible !

Application:

"Actually" the operation is simple:
After pressing "Start" and let draw the QR code, aim with cell phone,

- done -

Feel free to press "Save" to save the coordinates or "Load" to upload them.
But to minimize questions I have written a bit more about the operation. Some "Old school" guys like me still reads instructions,

Here you go (-;

The basic operation is described below.
It is assumed that the widget has been installed ready for use, for more details see chapter "Installation".

Rough functionality

The widget is divided into two scripts.

(1)

A "source script" takes care of the background activities.
It reads the current GPS telemetry coordinates, buffers them if there is no more reception, and saves or reads coordinates via file operations.
If "save" is triggered, the script creates a small file with two lines, the lat or lon coordinates.

The file gets the name of the model memory and is stored under
"/scripts/SRC_gps_UN/data".

Always 3 generations of a file are kept, so that "older" coordinates can be read out by PC in the emergency.

The "source script" provides the coordinates for the actual widget.

(2)

The widget is the "FrontEnd", here the user starts the QR calculation or initiates the saving or reading of coordinates.

As already mentioned, the QR calculation is complex, which is why it has to be started deliberately via a switch function (Horus) or button (touch display).

Even if the model has no power supply after a crash, the last received position is still visualized and can be saved.

Operation

After installing the widget, the interface appears without any QR calculation taking place.
You can recognize it by the fact that the start button is grey.
Let's imagine that the model has landed "outside" on a slope or has "stopped hard" in a cornfield for whatever reason.

We take a look at the widget:

In this example, the transmitter still displays a valid coordinate. If these coordinates no longer change, it is very likely that there is no GPS reception. If the model has no power supply after a crash, the acoustic message "Telemetry lost" could also be heard.

Save your coordinates:

In any case, it is advisable to trigger „SAVE“ in the first step (by touch or switch).

The Save button briefly turns blue, which means that the last saved position was no longer updated by the telemetry. So in case of a crash this button will stay blue !

Note: an "old" file is not simply overwritten, but renamed.
A maximum of three file versions (the current one and two predecessors) are stored.

Start / Cyclic calculation of the QR code

Then "Start" is triggered, and as soon as the time for a recalculation is reached, the QR code is processed and shown on the display.
The code is now recalculated cyclically. If a save was performed before, and the coordinates have changed, the Save button will change back to the color red, see example:

Should a calculation no longer be necessary:

UNCONDITIONALLY stop again to reduce the CPU load !

The above mentioned color change of the Save button can be used for the following scenario:
The model makes an outlanding, power supply remains intact.
Due to RF isolation, however, no GPS reception at the transmitter, so you save the last coordinates.
The button therefore turns blue, no change of the display.

Now you go via Maps to the area of the last known "location" of the model.
At some point the reception will occur again, by the color change of the Save button to red one recognizes that the model is further under Power and the now received coordinates will be more precise.

A certain amount of noise, i.e. small fluctuations in the coordinates, is natural and should be known.

If you still can't find the model, it is recommended to save three different data sets / files, if the coordinates change slightly (see above, three "generations" of files rotate).

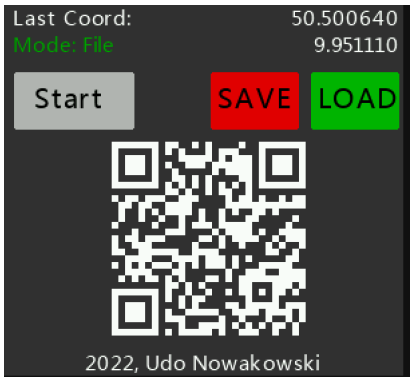
By triangulation afterwards (by PC) an even more precise indication should be possible.

hint:
If you restart the QR calculation and the last calculation is significantly older than the "refresh time", the QR code is usually calculated immediately without this being made known by a display text.
(... a case for Rev2.0)

Load cordinates from file:

If the transmitter has been switched off in the meantime and no GPS signal is received after switching on, the last file can be "uploaded".

Immediately after upload the QR code is calculated and displayed.



By means of the green "Load" button as well as the green mode text in the upper left corner you can see that currently the coordinates of a file are displayed.

Here, too, the color change of the "Save" button is an additional source of information:

>> If the button is blue, the coordinates have not changed since loading.

>> If the button is red, new coordinates have been received (see example above).

Installation:

(1) copy files & folders

After downloading the zip file you have to unzip it.
Now three folders are available, which must be copied into the script directory of the transmitter:

The "Modelfind_UN" folder contains the main widget

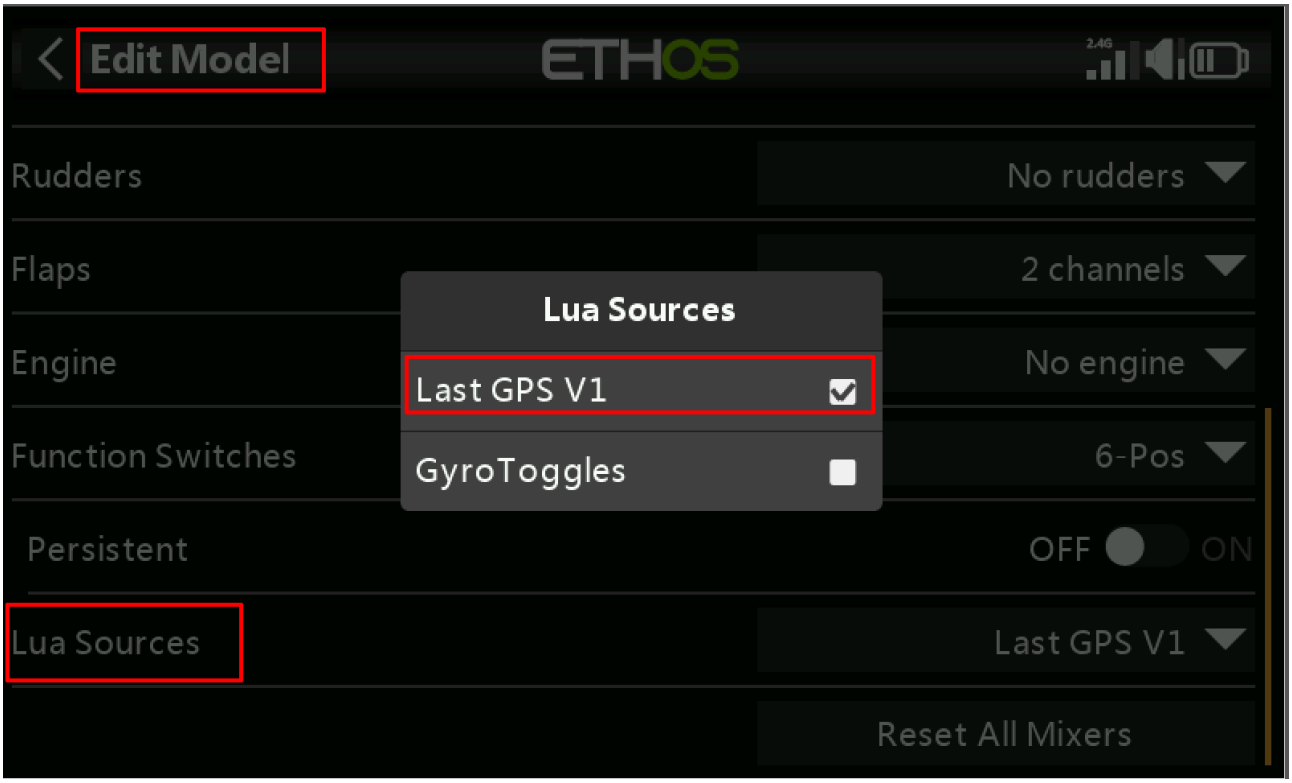
The "SRC_gps_UN" folder contains the source script

The "libUNow" directory contains common lua function definitions that can be used by multiple widgets as a „central“ source.

So the modelfinder widget needs this folder in order to call some functions.

(2) source script:

Source scripts are always activated under Ethos in the model setting (see also "Edit Model").



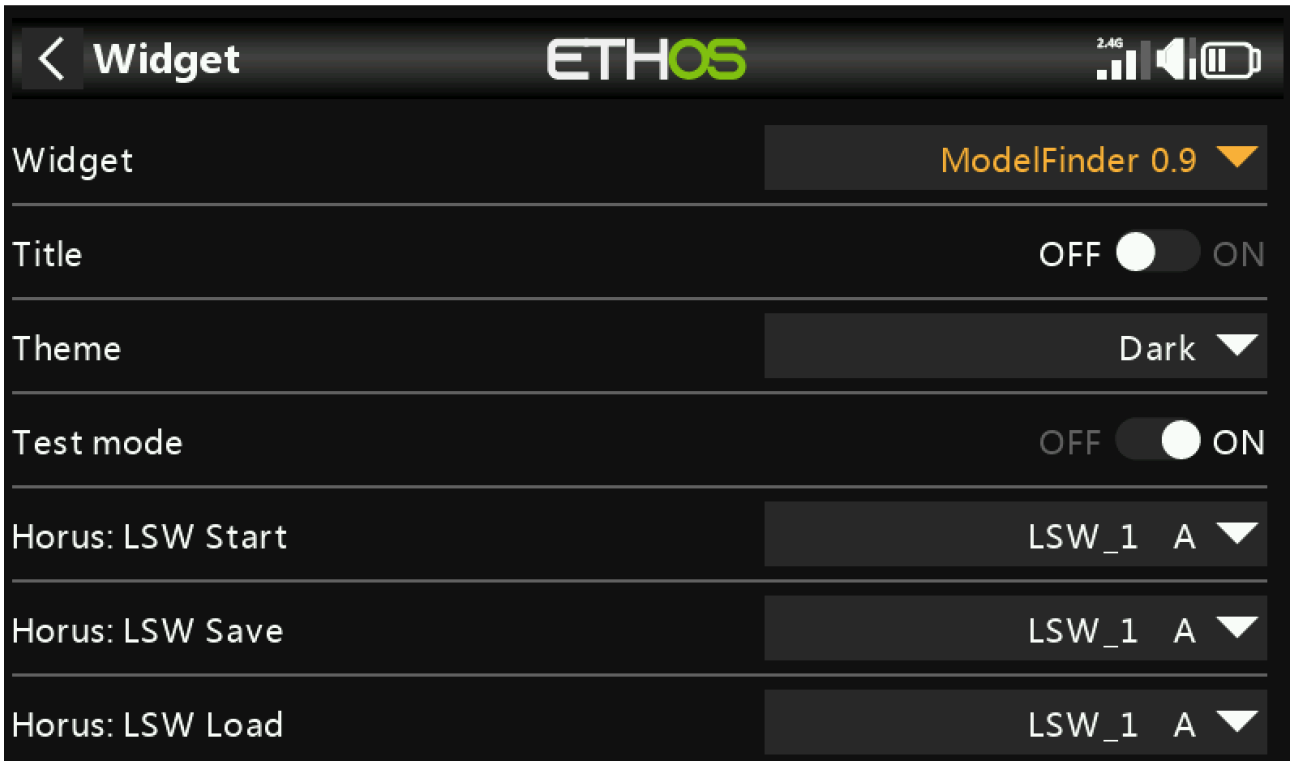
Under the corresponding form entry at the bottom ("Lua Sources") the "Last GPS" script has to be activated:

(3) widget einrichten

Das widget wird in einem Frame mit halber Seite Grösse eingerichtet.

Die Kenntnis darüber wie Ethos Seiten eingerichtet werden / Layouts zu wählen sind setze ich voraus.

- In der Widget Konfig wird zunächst das „Modelfinder“ widget ausgewählt.
- der Titel ist unbedingt abzuschalten
- die Darstellung (Thema / Farbschema) kann zwischen hell & dunkel gewählt werden
- wenn gewünscht kann der Testmodus scharf geschaltet werden (kein GPS / Empfang notwendig)
- Für Horus Sender **MÜSSEN** die logischen Schalter selektiert werden (näheres dazu weiter im Text..), Touch Sender können diese Parameter unangetastet lassen. Aufgrund der automatischen Sendertyperkennung im widget haben sie keinerlei Funktion.



(4) Horus only: definition of logical switches:

Since Horus transmitters do not offer a touch interface, the corresponding functions must be mapped via logical switches.

The widget always "reacts" to a status change of a LSW to simulate the touch operation.
So whenever a log switch changes from "off" to "on" or vice versa, the widget behaves as if a button was pressed on the touch screen.

Accordingly, three LSW's are to be created under Ethos.

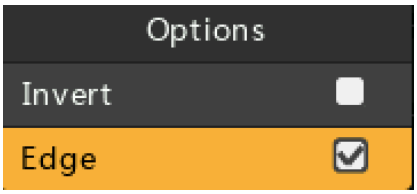
If the LSW's are defined, they are assigned in the widget config as described above.

The internal assignment in the widget is made by means of the NAME of the LSW, this ensures the persistent assignment, even if other LSWS are moved, deleted or newly created in the model config and thus the numbering changes !

Trim buttons can be used as a source for this:

- The LSW is defined by using the functionality "Sticky".

- Each time the button is operated, the switch status is to be changed.
  So you'll have to use LSW-functionality to react on an impulse ("Edge“)
  To achieve this, first select the trim button under "Trigger On",
  then the impulse variant is selected by "long press" on this entry.



The "Trigger Off" condition is defined in exactly the same way, except that "inverted" is also selected in the options.

This way you get a logical switch, which is switched on at the first "T5 left" click, and switched off at the next "T5 left" click.

For realization i'll show a screenshot, how e.g. T5 was defined as a switch for "Save":

Rev 1.0
unow, Nov 2022