Introduction

The purpose of this script set is to help in F3K tasks practice by providing dedicated timing.

I -ON4MJ- originally wrote it for the Taranis, then Taranis Plus. It requires OpenTX 2.0.13 or above.

It now works also as a big widget for the Horus with OpenTX 2.2.
[The initial port to the Horus was written by xStatiCa (Adam). Thank you, Adam :-)]

And it should work on the Q7 too, tough the status of this port is 'experimental' at the moment (only tested in the Companion simulator).

Pre-requisites

First, you must have OpenTX 2.0.13 or higher. The script will NOT work with an older revision of the firmware.
OpenTX 2.1 is supported from the 2.05 release of the script.
OpenTX 2.2 is supported from the 3.00 release of the script, and required for the Horus and the Q7.

The firmware must include the *LUA* option (and probably *no-heli* if you want to avoid memory problems).

Note: The purpose of this document is not to explain how to install a new firmware on your radio, but if you want to do it, this can be done from Companion. Be sure to set your radio type correctly in the settings, so that the program downloads the right firmware for you radio (Taranis and Taranis Plus use different versions of the firmware!). It's also in the settings that the "lua" check-box can be ticked.

If you're not too sure what all this means, please do read some up-to-date documentation first, or ask for help in your club or on your favourite internet forum. I will NOT provide any support on these matters.

Installation

The content of the `scripts` folder must be copied on the SD card of your radio:

1. There's a `F3K_TRAINING` subfolder which you have to copy in `/scripts` on the SD card on a Taranis (so, to be clear, the result is a `/scripts/F3K_TRAINING` folder containing several scripts).
   On a Horus, it must be copied in the `/widgets` directory.

   > *Note*: if you made modifications to the `custom.lua` file from a previous version, you might want to write them down before overwriting it with the new file, to report your modifications more easily in the new version. Pay attention that the syntax has changed from previous versions.

2. Then, if you're using a Taranis ('regular', Plus or Q7) there's a `f3k.lua` file in the `TELEMETRY` folder.
   a) If you're running **OpenTx 2.0**, move this file into the `/scripts/modelname` folder on your SD card (replace "modelname" by the name of your model). If the directory doesn't already exist, create it.

      > *Note*: for LUA scripts to work, the directory name (and thus, the model name) can't contain any whitespace. Replace spaces by underscores, or shorten your model name to a single word.

   The script must be renamed to `telem1.lua`. If you already have telemN.lua files in that directory, choose the first free number available (for example, telem3.lua).

      > *Note*: There's a memory limit for LUA scripts. If you run too many of them concurrently, there's a good chance that you'll get a "syntax error" message, because the firmware can't load one or more scripts.

   b) If you're running **OpenTX 2.1** or **OpenTX 2.2**, move the `f3k.lua` file into the `/scripts/TELEMETRY` folder on your SD card. There's no need to rename the file. However, you need to go to the telemetry tab (Companion) or model page (transmitter) to tell the firmware that you use a *script* telemetry screen, and choose the `f3k.lua` file.

      > *Note*: Companion 2.1.0 doesn't provide the file name choice, you have to do it on the transmitter. This was corrected in 2.1.1.

      > *Note*: In previous versions, it was necessary to copy a bunch of sound files. This is not needed anymore, they're included in a subfolder of F3K_TRAINING from release 3.00.

   If you are using a Horus, there's no need to copy anything else, the radio will load the `main.lua` script directly from your `F3K_TRAINING` directory.

Finally, **your model must not have any timer functionality**. If it does, this will compete with the script, and nothing will work correctly.
A possible solution is to copy your model in another model slot, remove your timer programming and use that model for training with the script.
The *Free Flight* task was added in order to provide the usual usage for timers, so hopefully, this problem should be history for most.

Usage

To go in the telemetry screens from the Taranis main screen, long-press the *Page* button. Then, *Page* again if there are other telemetry scripts installed for that model.

On the Horus, you need to set up the script as a widget first. You will need a full size widget, otherwise the script will only display a message telling you that your widget size is not supported.

Switch C is supposed to be up at that point.

> *NB*: this script is a "telemetry script" from the point of view of OpenTX, although it doesn't use any telemetry at all.

You should see the following menu:

You can select an entry in the menu with the **throttle stick**.

Then you can enter into the task by setting **Switch C** to the middle position. This selects the task, but doesn't start it right away. It might be a good idea to get your throttle stick back up at that point.

The task is started with Switch C in the down position. This will trigger a short vocal description of the task and start a preparation time of 16 seconds so that you can get ready. There's a countdown, then the task window timer starts. If you launch before the work time starts, an 'invalid flight' message will occur, and the launch won't be taken into account.

You can stop the current work time with Switch C in the middle position again. Putting it back in the up position will go back to the menu. Putting it back in the down position will reset and restart the task (from the start of the preparation time).

The start and end of flight are detected by the use of the *temporary switch*, which is used by most DLGers for their launch preset. This is **Switch F** by default.

Yes, the script assumes that you are right-handed and have swapped the top switches of your transmitter. If that's not the case, see the *Customization* section to change that.

The flight timer is started when the temporary switch is released after it has been held for more than 0.6 seconds.

The end of flight is triggered by the pull of the same switch. So there are two different cases when you land (or catch):

- a quick pull/release of the switch will stop the timer;
- a long pull/hold/release of the switch will stop the timer and restart it for a new flight.

This won't give perfectly accurate times (especially if you don't release your switch immediately after the plane leaves your fingers), but it's good enough for training, IMO.

Note that if you have a complex launch program (for example, holding the switch sets the launch mode and the release of the switch then sets the climb/zoom mode, you might need to adjust your program so that it matches the behaviour of the script, i.e. do nothing on a short pull of the switch).

Description of the task screens

> The script doesn't include Poker, because I'm still wondering how to implement that effectively.
> See/contribute to the discussion here: http://www.rcgroups.com/forums/showthread.php?t=2286831

The display will look a little different on a Horus or Q7 than in the Taranis screenshots shown in this section, but everything is mostly the same, with a few compromises to account for the different screen areas.

There are recurrent elements in all the screens:

- There's a big timer which is the work time counting down
- The lower left (upper left on the Q7) corner reminds you of the current task
- After the task name, there's the flight timer; it is backed up by vocal announcements. For most of them (all but 1234, in fact), the voice is a countdown, while the display is a count-up.
- The right side shows the list of times achieved for the task. There's a total displayed in reverse video for the regular F3K tasks.
- Some tasks have more or less interesting information to display depending on the context (there are also vocal announcements for different things)

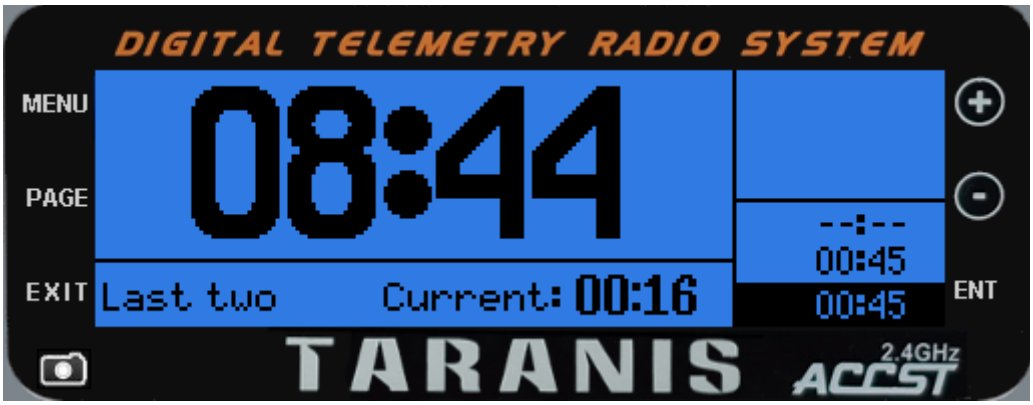
Task A - Last flight

There's only one interesting flight in the task, so, the flight list and total are in fact the same, unless your flight was longer than the limit.

The information message here tells you (when you've landed) by how much you could improve your time (“improvement margin”, truncated for space reasons).

Task A appears twice in the menu. The second one uses a 7 minute window. The max flight time is still 5 minutes.

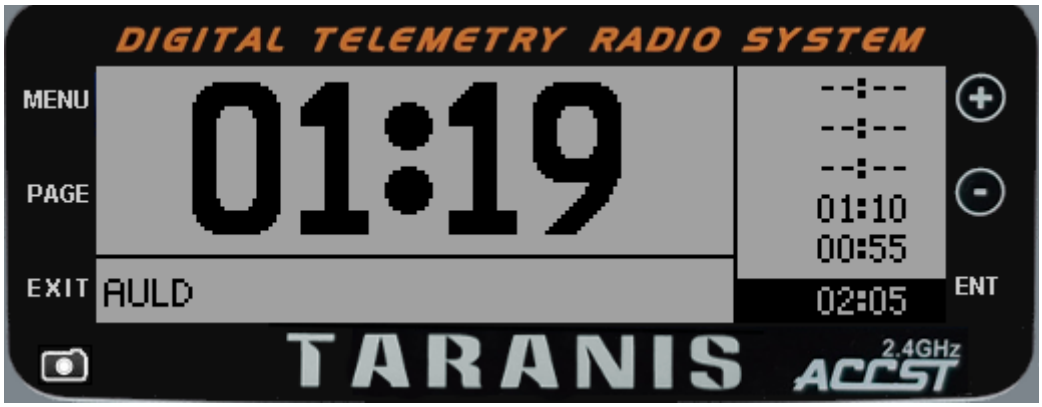
Task B - Next to last and last flight



The same as task A, except that the list displays the two last flights.

There are two variations: a 10 minute window with a max flight time of 4 minutes and a 7 minute window with a max of 3 minutes.

Task C – All Up Last Down



This task doesn't have a working time or a preparation time. You get five flights of maximum three minutes that you start whenever you want. The big timer display is the current flight time in this task. In practice, this could be run with a few friends, one calling a countdown for the launch. Everybody should launch at the same time (more or less). The goal is to stay aloft more than the others, up to three minutes. The winner is the one with the higher total after five flights (you could decide to fly only three or four if you want).

Task D – Ladder

The « current » flight has a label reminding you what the current target is.
The flight list shows what steps were valid by displaying them in reverse video.
The total for the task (`01:15` in the above picture) goes on the left of that frame for space constraint reasons.

There are dedicated vocal announcements to tell you what the next target is, etc... (note that this sounds a bit weird, since two different voices are used to make the message).

Task F - Three out of Six

The current flight has a label telling you which flight this is.
The times list on the right is sorted.
The task stops automatically after six flights.
The max flight time is 3 minutes.

Task G - Five longest flights

Nothing special, the flight list displays the five best, sorted.
The max flight time is 2 minutes.

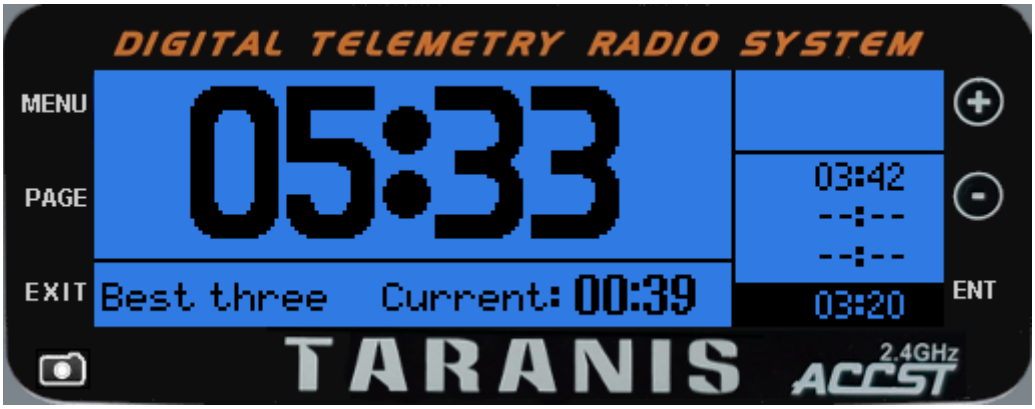
Task H - 1234

This one is a little different from the others, as it's not possible to know what your target is.
So, the announcements are set going up, and their granularity is a little different from the others.

The scripts tries to infer which target you completed, and displays those in reverse video.
The label in front of the current flight displays what the script thinks your higher unachieved target is.

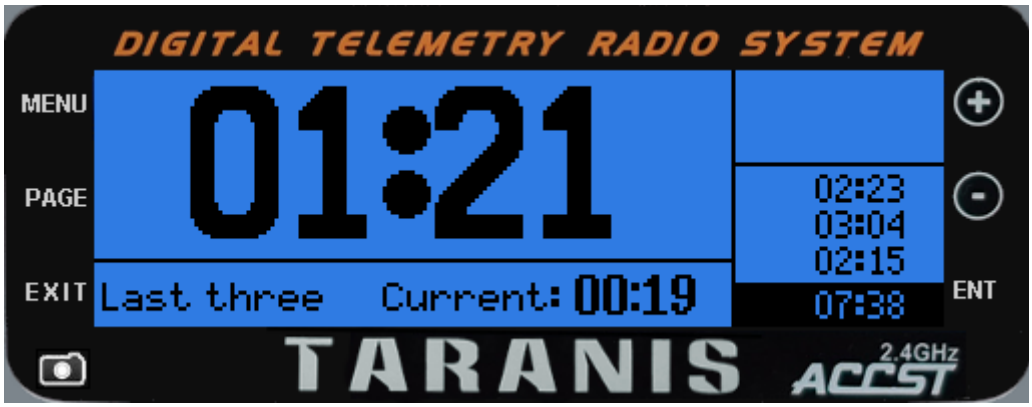
Finally, the four best flights are displayed in sorted order (note that they're not displayed on the same line as the achieved targets, unless you do them in the 4321 order).

Task I - Three longest flights



Like task G, but with only three flights.
The max flight time is 3m20.

Task J - Three last flights



Same as tasks A & B, but with the list of the three last flights.
The max flight time is 3 minutes.

Task M – Big Ladder

The « current » flight has a label reminding you what the current target is.

There are dedicated vocal announcements to tell you what the next target is, etc... (note that this sounds a bit weird, since two different voices are used to make the message).

The task stops automatically when the fifth flight is completed.

Note that unlike Ladder, each flight has a new target, whether the previous step was completed or not.

Task QT – Quick turnaround practice

This made-up “task” is there for those who want to practice their quick turnarounds.

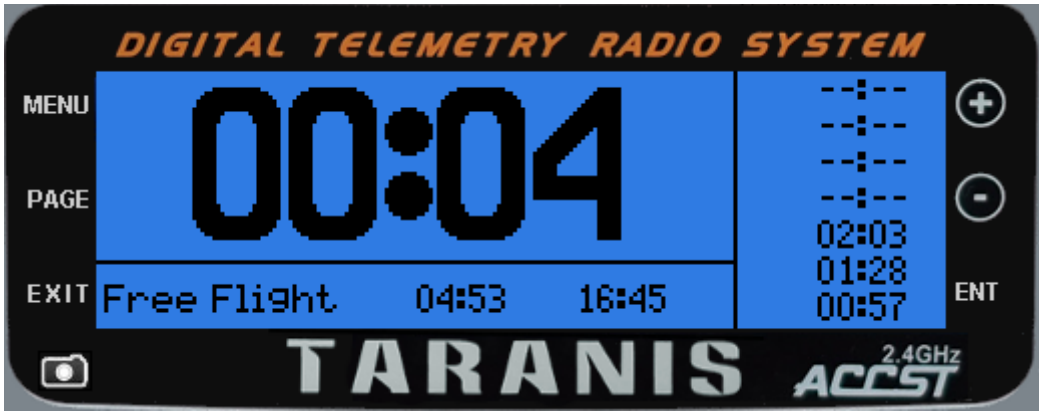
The task lasts ten minutes with a maximum of 15 flights. The flight duration is set to 40 seconds.

The right column shows your best time difference with the 40s goal (*Delta Min*), your worst performance (*Delta Max*) and the average delta. There's no penalty if you fly less than 15 times. Don't stress about that.

There's no difference if you're two seconds early or two seconds late to catch. The delta will be two seconds in both cases. I could have doubled the delta min/max for times over and under the goal, but I don't think it's really important. The only thing interesting here is to have a bunch of short flights with a countdown.

Revision 2.04 added your total flight time. It was a request, but I wonder if it wouldn't be best to display the “fumble” time (the total of time when the plane was not flying).

Task FF – Free Flight



Before that screen was added, if you wanted to simply fly your DLG without practicing the tasks, but still wanted to use the timers, you could run into troubles, because the script would still start and stop them. So, basically, you had to make a copy of your model which did not use the script to be able to run your own timers. Any modification to your model had then to be ported to the copy, etc... It was a pain.

This tries to solve that problem...

This screen starts an up-going timer when you start the "task" (with the 3-pos switch) which will run until you put that switch back up. Note that contrary to the other tasks, this total flight time (an "open window") is displayed on the bottom (that's the 04:53 in the above screenshot).

Each flight is timed with the same actions on the temporary switch as in the other tasks. Its display is the large one. The last flights are kept on the right side in a scrolling column.

And finally, the clock is displayed (16:45 in the screenshot). So, if you fly during your lunch break and have to return for a certain time, you should have the info you need there as well ;-)

Known issues:

- The clock access from LUA was broken in OpenTX 2.1.7 (and before), so, you'll have a 00:00 display there, but it was fixed in 2.1.8.

- Also, be aware that if you fly more than 60 minutes, the timer will go on counting the minutes up to 99, and a minute later, it will restart from 0.

Customization

There are a few variables which can be used to customize the script behaviour.
You will find them in the `custom.lua` file in the `F3K_TRAINING` folder.

```
-- >>> Customize your own switches/preferences here <<< ---

local PRELAUNCH_SWITCH = 'sf'        -- temporary switch on the left
--local PRELAUNCH_SWITCH = 'sh'      -- temporary switch on the right (or emulator)
local MENU_SWITCH = 'sc'             -- a 3-pos switch is mandatory here: up=menu; mid=stop; down=run
local MENU_SCROLL_ENCODER = 'thr'

local SOUND_PATH = F3K_SCRIPT_PATH .. 'sounds/
```

If you want to use Switch H as your temporary switch, remove the "--" in front of the line with the *'sh'* and put them in front of the line with the *'sf'*.

If you want to replace Switch C by another switch, change *'sc'* by something else (for example *'sd'* for Switch D).

MENU_SCROLL_ENCODER is the input used for navigating the menu. If you don't like using the throttle stick for that, you can use *'ls'* or *'rs'* for the left or right slider.

> Note: On Horus using widgets, you can customize the options in the GUI Widget settings menu by holding the *select* button while the widget is selected.

If you feel the need and are able to do it, the launch/landing detection system is this part:

```
-- >>> Launch / Land detection <<< ---

local function launched()
	local ret = false
	if getValue( Context.options.PrelaunchSwitch ) < 0 then
		-- if the tmp switch is held for more than 0.6s, it's a launch;
		-- otherwise it was just a trigger pull to indicate that the plane has landed
		if lastTimeLanded > 0 then
			if (getTime() - lastTimeLanded) > 60 then
				ret = true
			end
			lastTimeLanded = 0
		end
	else
		if lastTimeLanded == 0 then
			lastTimeLanded = getTime()
		end
	end
	return ret
end

local function landed()
	if getValue( Context.options.PrelaunchSwitch ) > 0 then
		lastTimeLanded = getTime()
		return true
	end
	return false
end

-- >>> End of launch/land detection section <<< ---
```

You can replace it by anything you want, but it must provide *launched()* and *landed()* functions returning boolean values.
Or you could simply change the duration of the switch "trigger". Replace the 6*0* by anything you want (it's expressed in 1/100th of seconds).

For example, you could write something like this:

```
-- >>> Launch / Land detection <<< ---
local function launched()
	return getValue( 'ls31' ) > 0
end

local function landed()
	return getValue( 'ls32' ) > 0
end
-- End of launch/land detection section -
```

And provide the launch / land detection in conventional programming with

- **LS31** : launch detected
- **LS32** : landing detected

I hope you'll enjoy this script.

You can post comments or bug reports on my blog on RC Groups here:

http://www.rcgroups.com/forums/showthread.php?t=2298914

Mike, ON4MJ