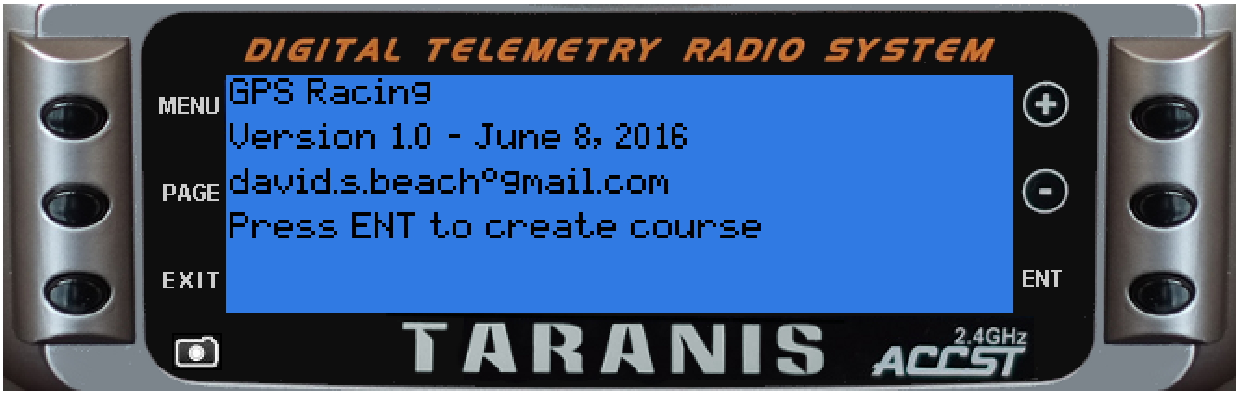
FrSKY Taranis GPS Racing Telemetery Script

User's Guide



GPS Racing
Version 1.0 - June 8, 2016
david.s.beach@gmail.com
Press ENT to create course

David Beach
david.s.beach@gmail.com
June 8, 2016

Table of Contents

Revision History

Jun 8, 2015 – Release 1.0

Script now compatible with OpenTx 2.0.13 and OpenTx 2.1.7 (and hopefully later releases)
Added Task Speed calculations (verified against RC Electronics T3000 triangle tasks)
Added XCRestrt mix script (support for finish/restart via user selectable switch)
Documentation updates
Minor bug fixes and interface changes

Jan 5, 2014 – Beta 0.2

Added support for Line / Cylinder and Custom courses. Additional documentation released.
Fixed bug related to nil telemetry values.

Dec 20, 2014 – Beta 0.1

Initial release with Triangle and 2 Line courses only. Distribution was limited and minimal documentation was provided.

Introduction

Welcome, and thanks for using the GPS Racing Telemetry Script for the FrSKY Taranis. The scripts are supported via the RCGroups thread at http://www.rcgroups.com/forums/showthread.php?t=2219128 , or via email to mailto:david.s.beach@gmail.com. If you find a problem and can reproduce it, there is a good chance I can fix it. If there are what seem to be random failures please report them and I'll try to figure out what can be done to prevent it.

The system has been developed for the FrSKY Taranis and Taranis Plus transmitters running OpenTX versions 2.0.13 or 2.1.7 (or later). Your transmitter firmware must include the LUA option, but there are no other specific firmware requirements. The telemetry scripts do not use any flight modes, global variables, logical switches, special functions or custom one-time scripts. Installation of the telemetry scripts should not introduce any side effects to existing model setups.

This release now includes an optional mix script (XCRestrt.lua) that enables finish/restart via a user selectable switch.

The scripts require an onboard FrSKY compatible GPS that makes latitude and longitude information available in the standard telemetry data stream. It has been tested with the FrSKY GPS V2 sensor and a generic GPS connected to the FrSKY sensor hub. Users have also reported success with the Altis GPS and home built GPSs based on OpenXSensor. Owners of FrSKY GPS V2 sensors should consider either upgrading the firmware (see http://www.rcgroups.com/forums/showthread.php?t=2600818) or purchasing a compatible sensor with a higher update rate.

Course navigation assistance in provided via audible feedback at all times, and via the transmitter display when the appropriate telemetry page is selected.

This system includes a GPS simulation option that allows you to become familiar with the scripts using only your transmitter or the OpenTX Companion software available at http://www.open-tx.org/downloads.html

Installation

Copy files to the transmitter

Everything is provided in a single zipped archive that must be unpacked. The individual files are then placed in the appropriate directory on the transmitter SD card. All the xc*.wav files must be placed in /SOUNDS/en/ directory.

OpenTx 2.0.x users should place telem1.lua and telem2.lua files in the /SCRIPTS/<*model name*>/ directory for all models used for GPS racing.

OpenTx 2.1.x users should copy the telem1 and telem2 scripts to the /SCRIPTS/TELEMETRY directory. Then reference the scripts in the telemetry configuration page for all models used for GPS racing.

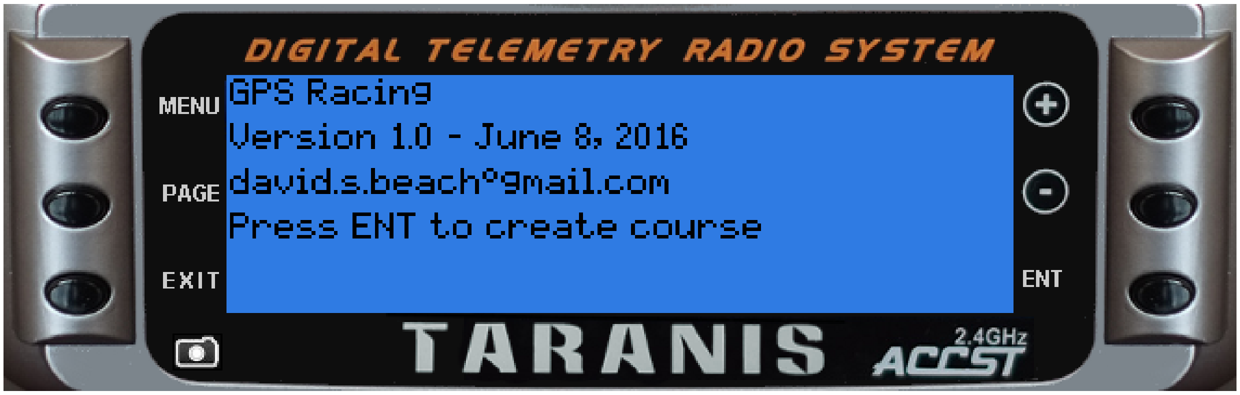
NOTE: The file telem1.lua is provided in LUA bytecode format and should not be modified using a typical text editor. While I do plan to make the source code available, it was too big to successfully compile and run in the transmitter.

The file telem2.lua is a normal text file and may be modified by users to set course defaults and override other parameters. It is also optional and can be omitted from a model if needed to make more memory available for other telemetry scripts.

On OpenTX 2.0 the telem1.lua file can be renamed, but must be loaded before telem2.lua if used. For example, if telem1.lua is renamed telem4.lua then telem2.lua must be renamed telem5.lua (or higher).

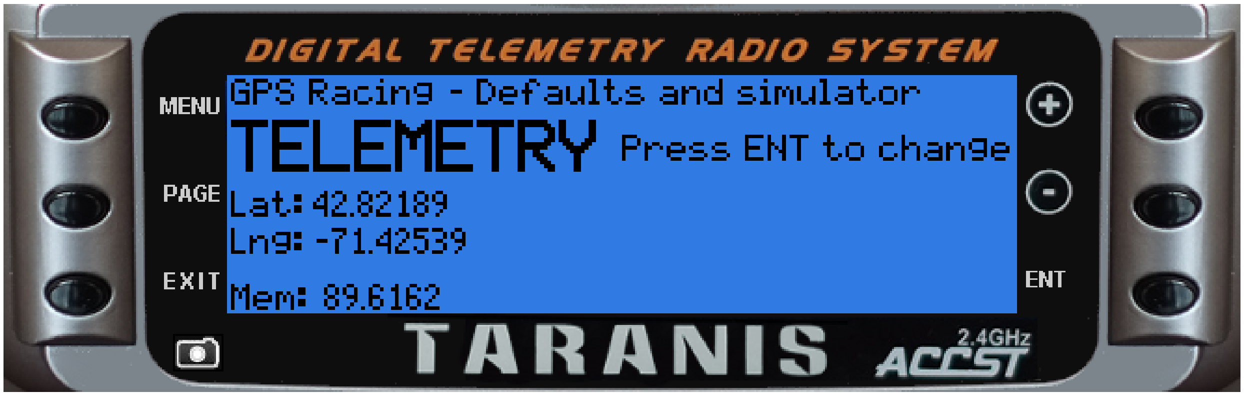
Installation verification

When properly installed you can select the telemetry display via a long press of the PAGE key. Continue to press the PAGE key and you should eventually see the telem1.lua display:



```
GPS Racing
Version 1.0 - June 8, 2016
david.s.beach@gmail.com
Press ENT to create course
```

followed by the telem2.lua display:



GPS Racing - Defaults and simulator

TELEMETRY Press ENT to change

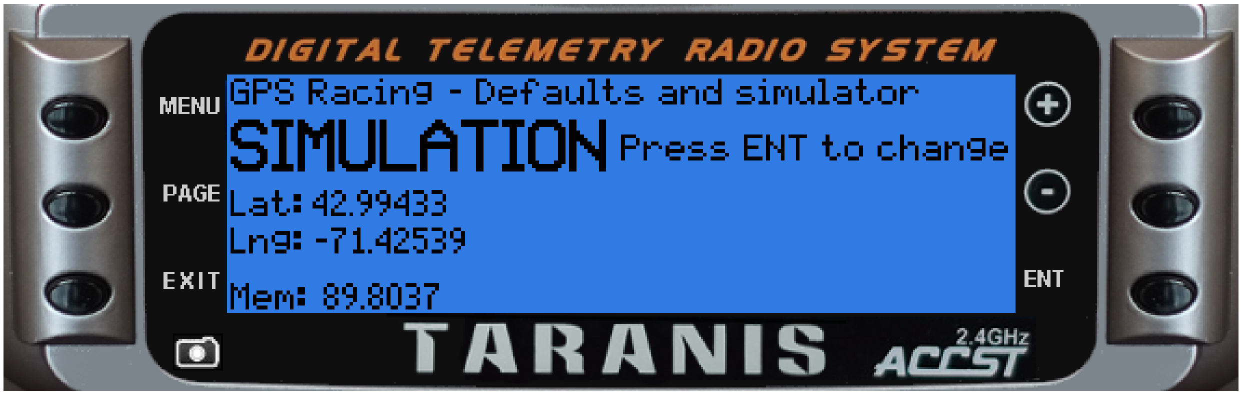
Lat: 42.82189

Lng: -71.42539

Mem: 89.6162

Using the simulated GPS

Activate the simulated GPS by going to the telem2 display and pressing the ENT button. It should now look something like this:



GPS Racing - Defaults and simulator

SIMULATION Press ENT to change

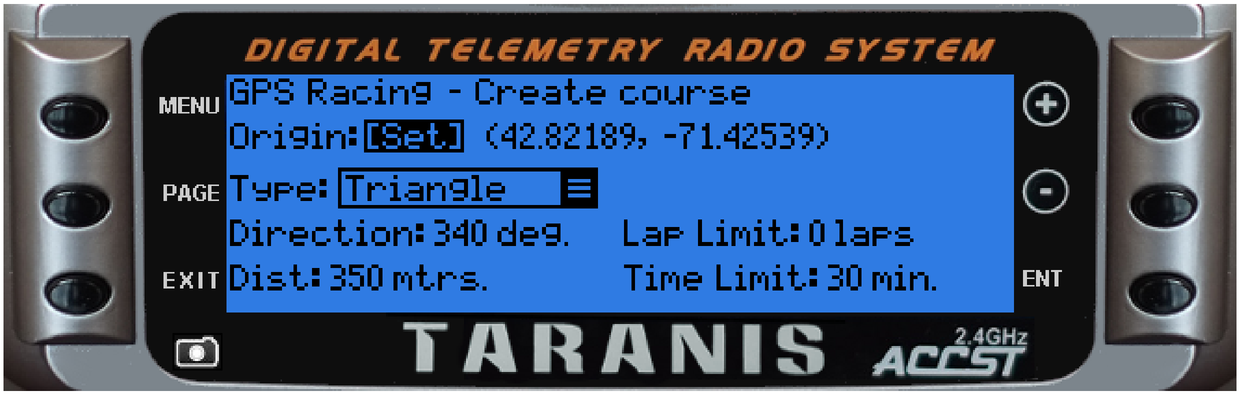
Lat: 42.99433

Lng: -71.42539

Mem: 89.8037

When in simulation mode, the throttle stick controls speed and the rudder and aileron sticks control turn rate (*I've only tested in mode two, if you fly mode one let me know if the sticks behave as you would expect*). When simulation is active you will notice changes in displayed Lat: and Lng: provided the throttle stick is not zero.

Now press PAGE repeatedly until you are back on the telem1 display. Press ENT from this screen to define a GPS based course.



GPS Racing - Create course

Origin: [Set] (42.82189, -71.42539)

Type: Triangle

Direction: 340 deg. Lap Limit: 0 laps

Dist: 350 mtrs. Time Limit: 30 min.

With [Set] selected (as shown above) pressing ENT will cause [Set] to blink and the displayed GPS coordinates will automatically update. Now you can raise the throttle and use the aileron stick to see that position updates are coming through. (Note that when the TELEMETRY option is active moving the plane (once it has a GPS lock) will also cause the coordinates to update.) Press ENT again to set the origin for the course. Using the + and – keys will navigate to the other options. Press ENT on any field to edit, press ENT again to exit edit mode.

Course Parameter Descriptions

Origin: This sets the starting latitude and longitude for each course type (except Custom courses).

Type: Course type –Triangle, 2 Lines, Line / Cyl, and Custom courses are available and explained in more detail below.

Direction: This is the initial heading (0 to 360 degrees true) from the origin to the first waypoint.

Dist: This is the “leg length” in meters for waypoints relative to the origin.

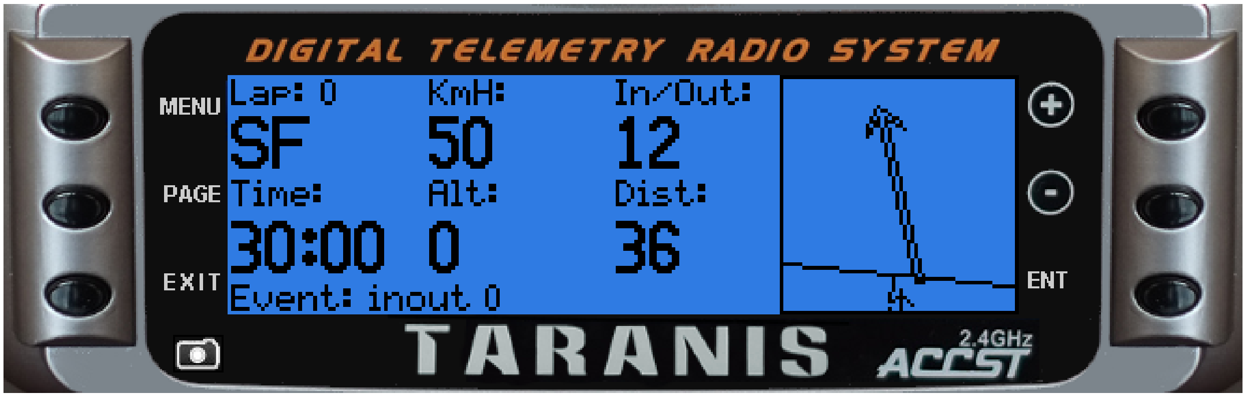
Lap Limit: Enter a value greater than zero if a lap limit is desired.

Time Limit: Enter a value greater than zero if a time limit is desired.

Note that if both “Lap Limit” and “Time Limit” are specified then the course will finish when either one of the defined limits is reached (whichever occurs first).

Let’s go racing!

When you are satisfied with the course parameters press EXIT to initiate course navigation. Note that any changes made here will be lost when the transmitter is turned off. However, all course parameters can have default values set by editing the telem2.lua script (see details below).



```
Lap: 0      KmH:      In/Out:
SF          50        12
Time:       Alt:      Dist:
30:00       0         36
Event: inout 0
```

Shown above is the main telem1 screen when a course is active. I’ll describe all the text fields first, then acquaint you with the graphical portion of the display.

Lap – the current lap number is displayed. Below that is the name of the next waypoint. Here “SF” stands for start/finish.

KmH – calculated ground speed in meters per second. (note: is speed is available via telemetry it is currently not used.

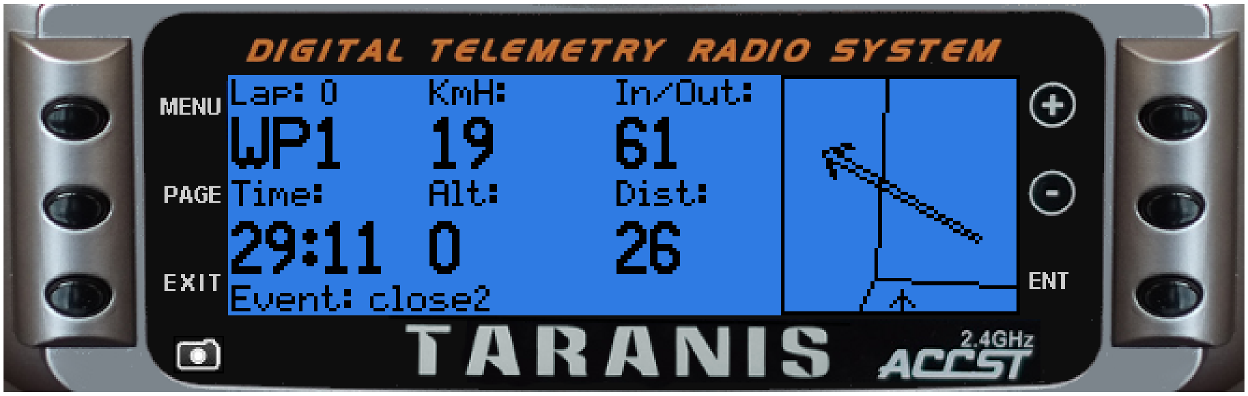
In/Out – this is the deviation from the ideal course between waypoints measured in meters. Positive numbers are to the outside (right) of the ideal line, negative numbers are inside (left).

Time – displays elapsed (count up) or remaining (count down) time depending on whether or not a time limit was specified.

Alt – altitude as reported by a variometer if present (unit of measure is determined by telemetry configuration)

Dist – distance in meters to the next waypoint. For triangle courses it will display the distance to the origin if you are 'inside' otherwise it will display the shortest distance to the first line describing the waypoint.

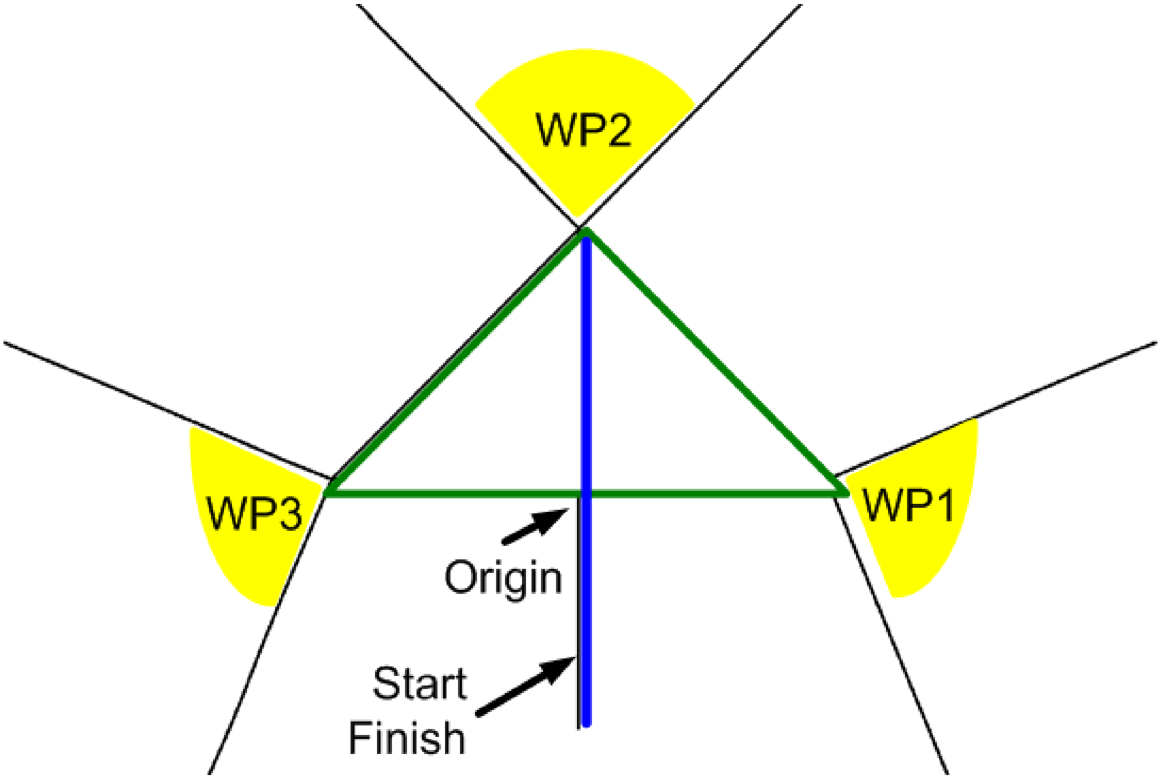
Graph – the tip of the small arrow at the bottom of the graph indicates the plane's relative position on the course. Straight lines are scaled representations of the next waypoint. The large outlined arrow shows the relative direction of the waypoint origin from the plane's perspective. In general, when the outline arrow points straight up you are headed in the correct direction. As you get nearer to a waypoint the arrow will rotate in the direction of the waypoint origin as shown below:



```
Lap: 0    KmH:    In/Out:
WP1       19      61
Time:     Alt:    Dist:
29:11     0       26
Event: close2
```

Triangle Course

The diagram below portrays the Triangle course:



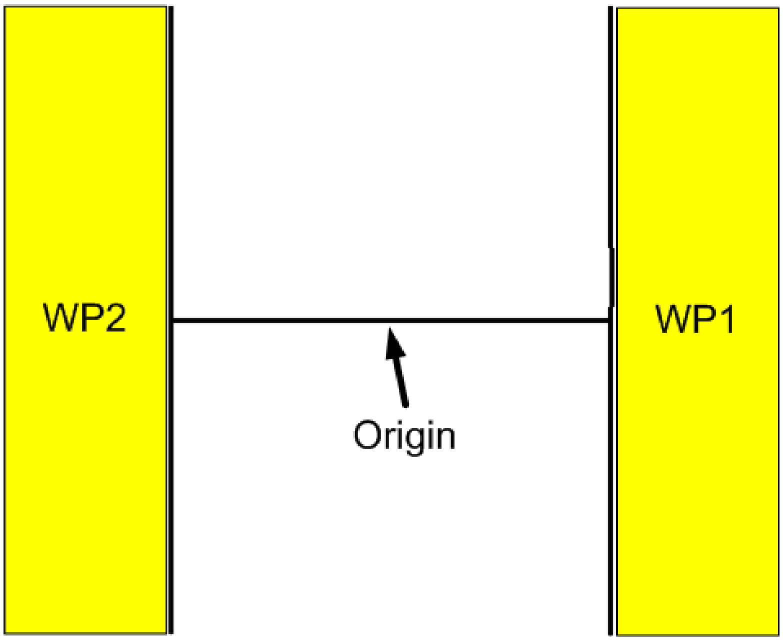
Given an origin coordinate, WP1 is located using the direction specified. Assuming this diagram is oriented North up, this example would be a 90 degree course. All three waypoints are equidistant from the origin.

Once a course has been activated, time starts when the start/finish line is crossed in the direction toward WP1. The waypoint is considered achieved any time the plane enters the yellow area (including the extended non-highlighted area).

A lap is complete when all waypoints have been achieved in order, and the start/finish line has been crossed again.

2 Line Course

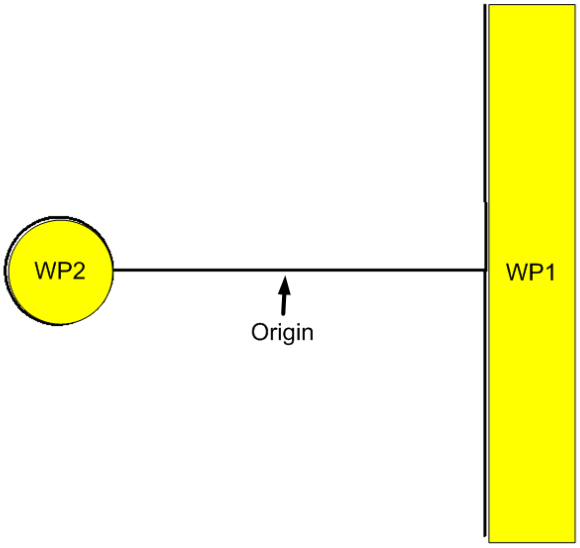
The diagram below portrays the 2 Line course.



Given an origin coordinate, WP1 is located using the direction and distance specified. The line defining WP2 located in the opposite direction. The course starts when the plane passes out of the yellow WP1 area back towards the origin. WP2 is achieved when the plane enters any part of the extended WP2 area. A lap is complete when the plane returns to WP1.

Line / Cylinder Course

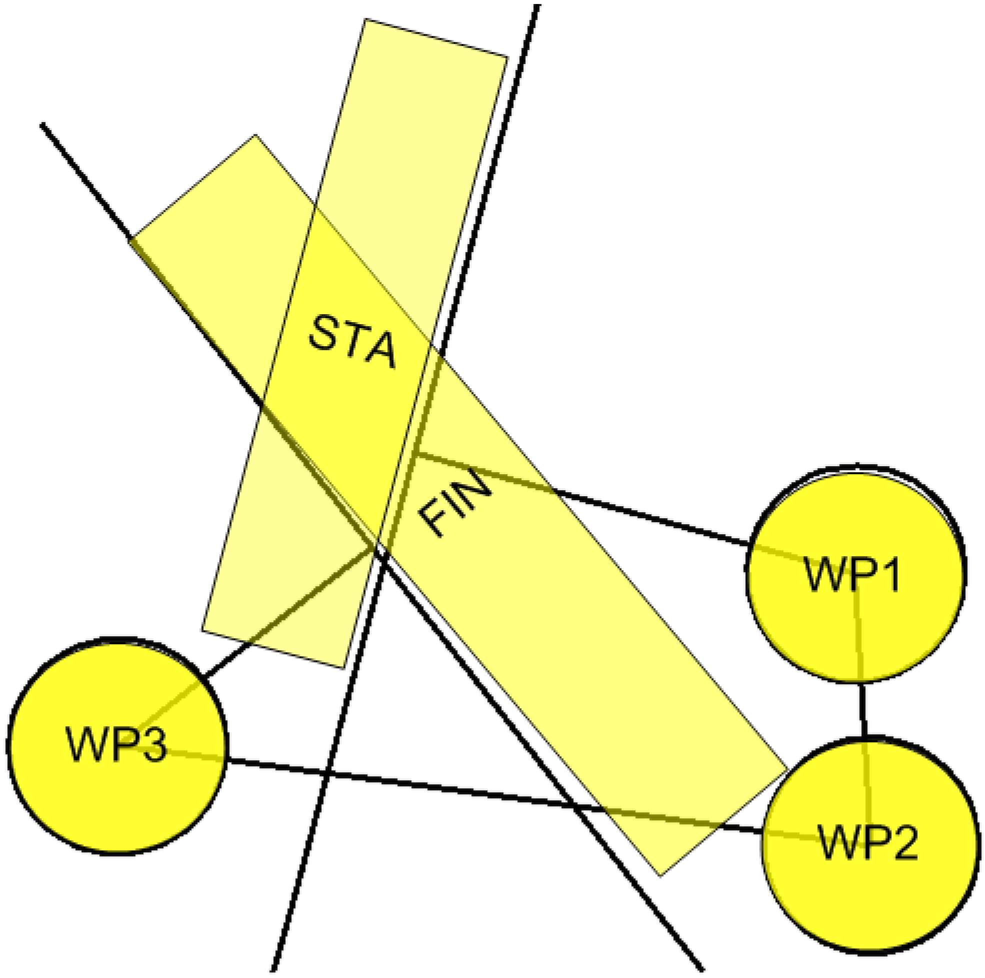
The diagram below portrays a Line / Cylinder course.



Given an origin coordinate, WP1 is located using the direction specified. Similar to the 2 Line course, WP2 is located in the opposite direction but it is defined as a vertically oriented cylinder whose closest point is determined by the distance specified. The course starts when the plane passes out of the yellow WP1 area back towards the origin. WP2 is only achieved when the plane enters the WP2 cylinder. A lap is completed when the plane returns to WP1.

Custom Course

The diagram below portrays a Custom course.



Custom courses must be defined in the Telem2.lua file. Any number of waypoints may be defined and given arbitrary names and coordinates. A minimum of three waypoints are required for a custom course. The first and last waypoints are rendered as lines, and all others are cylinders. The course starts when the plane passes out of the start area (here STA) toward WP1. A lap is complete when all cylinders have been achieved in order and the plane passes into the finish area (here FIN). Note that in the above diagram it is possible to finish a lap inside or outside of the start area. If you are outside the start area when a lap is complete, you must return to the start area to initiate a subsequent lap.

Setting custom course waypoints via telem2.lua

Here is an example custom course definition:

```
xcCustom = {
  {"STA", 42.824101, -71.423681},      -- parameters are waypoint name, latitude, longitude
  {"WP1", 42.824348, -71.428071},      -- a minimum of three waypoints are required
  {"WP2", 42.819736, -71.424117},      -- duplicates are OK as long as they are not adjacent
  {"WP3", 42.819988, -71.427973},      -- start and finish (first and last) will render as lines
  {"FIN", 42.824101, -71.423681},      -- all other waypoints are rendered as cylinders
  }
```

Setting defaults via telem2.lua

Telem2.lua is an optional lua source file that can be edited with any text editor. Modifications to this file must be valid lua syntax or the dreaded “telem2: Script syntax error” may result.

```
local function init()
  xcC={{
      42.82189, -- origin latitude
      -71.42539,-- origin longitude
      340,      -- initial course heading (0 - 360)
      350       -- leg length
    },
    {},
    0, -- default course type 0=Triangle, 1=2 Line, 2=Line / Cyl, 3=Custom
    30, -- time limit in minutes - zero is unlimited
    0  -- lap limit - zero is unlimited
  }
  sim=false -- simulator options are true or false
  -- set the default starting position for the simulator (will be overridden if GPS lock occurs)
  cPos[1]=42.82189
  cPos[2]=-71.42539
  xcIOWidth = 50 -- width of in/out announcements in meters
end

local function run(e)
  lcd.clear()
  lcd.drawText(1,1,"GPS Racing - Defaults and simulator",0)
  if sim then
    lcd.drawText(1,11,"SIMULATION",DBLSIZE)
  else
    lcd.drawText(1,11,"TELEMETRY",DBLSIZE)
  end
  lcd.drawText(106, 16,"Press ENT to change mode",0)
  lcd.drawText(1,31,"Lat:",0)
  lcd.drawText(lcd.getLastPos()+2,31,rnd(cPos[1],5),0)
  lcd.drawText(1,41,"Lng:",0)
  lcd.drawText(lcd.getLastPos()+2,41,rnd(cPos[2],5),0)
  lcd.drawText(1, 56, "Mem:", 0)
  lcd.drawText(lcd.getLastPos() + 4, 56, rnd(collectgarbage("count"), 4), 0)
  if e == EVT_ENTER_BREAK then sim = not sim end
end

return{init=init,run=run}
```

Future Enhancements

Telemetry enhancements

The stock GPS from FrSKY provides only latitude and longitude. Currently that information is the basis for calculating heading and speed. Ideally I'd like to get heading and speed from the GPS itself when it is available in the telemetry stream.