B O X Y 2

2018.07.22

Flight Analysis Script format

Table of Contents

A) Introduction

The „flight analysis script" format, short FAS format, is a scripting language which intends to perform some automatic analysis tasks on GPS tracks from balloon flights.

The FAS file format is a line based format. Empty lines are ignored. Each line contains an instruction which has to follow one of the here given syntaxes:

- standard syntax
- extended syntax

A script could use any mix of the two defined syntaxes.

1) Standard syntax

In the standard syntax, each line contains a list of comma delimited values, referred to here after as "fields". White-spaces before and after each field are to be ignored and trimmed out.

Syntax: `<recordType>,<name>, …`

<recordType> is always the type of the record (case insensitive)
<name> is always the name of the record (case ***sensitive***)

2) Extended syntax

The extended syntax enhances the overall legibility of the script by offering a function-oriented syntax.

Syntax: `<name> = <recordType>( … )`

Leading or trailing white-spaces must be ignored and trimmed out. All characters after the closing brakes will be ignored.

3) Line endings

There exists some special line ending symbols. Any line ending with

; is being evaluated but no output is being printed

! highlights the output of this line (default is **bold** and blue)

Currently the following types of records are defined:

POINT	defines a point	*the result is a point*
LABEL	outputs a label during the scripting result output	
RESULT	calculates a result	*the result is a value*
CHECK	performs a check on a point or a result	*the result is a boolean (except for PZ checks!)*
MAP	draws a visual element on a map	
SET	allows to set different options	

All following record definitions are given in the standard syntax.

Entries marked in red are not yet implemented.

B) POINT records

Fields: **point**,<name>,<type>, ...

<name> is any sequence of symbols. A name **must not** contain a comma!
<type> is one of the codes from the list below (case insensitive)

There are different type of points. The starting letter indicates which type of point it is:

- **A** (area intersection point)
- **D** (delayed point)
- **C** (copy)
- **N** (nearest point to)
- **S** (static point)
- **L** (point from a list)
- **M** (logger markers)
- **U** (user input)

1) Area intersection point

This adds a point relative to an area.

Fields: ***point***,<name>,<type>,<file>,<maxTime>,<minAlt>,<maxAlt> *(,<color>)*

<name> is any sequence of symbols. A name **must not** contain a comma!
<type> is one of the codes from the list below
<file> is an URL to the file the area (polygon) has to be loaded from (PLT, GPX or IGC)
<maxTime> is the maximum validity time (format = hh:mm:ss)
<minAlt> is the minimum altitude for validity (in feet)
<maxAlt> is the maximum altitude for validity (in feet)
<color> is the color the area is drawn on the map

AFITP Area First In Track Point
Returns the first entry point which is inside the area.

AFIIP Area First In Interpolated Point
Returns the first interpolated entry point which is inside the area.

AFOTP Area First Out Track Point
Returns the first exit point which is inside the area.

AFOIP Area First Out Interpolated Point
Returns the first interpolated exit point which is inside the area.

ALITP Area Last In Track Point
Returns the last entry point which is inside the area.

ALIIP Area Last In Interpolated Point
Returns the last interpolated entry point which is inside the area.

ALOTP Area Last Out Track Point
Returns the last exit point which is inside the area.

ALOIP Area Last Out Interpolated Point
Returns the last interpolated exit point which is inside the area.

2) Delayed point

This adds a point which occurs after a time or distance delay.

Fields: ***point***,<name>,<type>,<otherPoint>,<delay>,<maxTime>

<name>	is any sequence of symbols. A name **must not** contain a comma!
<type>	is one of the codes from the list below
<otherPoint>	is the name of another previously defined point
<delay>	is either a time delay (in minutes) or a distance delay (in meter)
<maxTime>	the maximum validity time (format = hh:mm:ss)

DITTP Delayed In Time Track Point

Starting at the <otherPoint>, this returns the last valid track point within the delayed time which is before <maxTime>.

DITIP Delayed In Time Interpolated Point

DIDTP Delayed In Distance Track Point

Starting at the <otherPoint>, this return the last valid track point within the delayed distance which is before <maxTime>.

DIDIP Delayed In Distance Interpolated Point

Starting at the <otherPoint>, this returns the last valid interpolated point which is before <maxTime>.

➢ If no point is found or can be calculated within the given delay and maximum time, a null point is returned!

3) Valid point

<name>	is any sequence of symbols. A name **must not** contain a comma!
<type>	is one of the codes from the list below
<refPoint>	the name of the point the time is being checked on
<usedPoint>	the name of the point that should be used it time of <refPoint> is valid
<from_x>	validity from (format = mm:ss)
<to_x>	validity to (format = mm:ss)

VTDTP Valid Time Dependent Track Point

*If the points time is in **not** in one of the [from_x, to_x] intervals, a null point is returned.*

4) Copy point

Copy a point.

Fields: ***point***,<name>,<otherPoint>,...

<name> is any sequence of symbols. A name **must not** contain a comma!
<otherPoint> is the name of another previously defined point

COPY Copy another named point but set a new height
Fields: ***point***,<name>,<otherPoint>,<newHeight>

<newHeight> is the new height in feet

CMOVE Copy another named point but set move it to a relative new position
Fields: ***point***,<name>,<otherPoint>,<eastingOffset>,<northingOffset>,<altitutudeOffset>

<eastingOffset> is the easting offset to apply to the copied point
<northingOffset> is the northing offset to apply to the copied point
<altitutudeOffset> is the altitude offset (in feet) to apply to the copied point

5) Nearest point to

This adds a point which is closest to any other point.

Fields: ***point***,<name>,<type>,<otherPoint>,<maxTime>

<name> is any sequence of symbols. A name **must not** contain a comma!
<type> is one of the codes from the list below
<otherPoint> is the name of another previously defined point
<maxTime> the maximum validity time (format = hh:mm:ss)

NTPTP Nearest To Point Track Point
Returns the closes valid track point to <otherPoint> which is before <maxTime>.
NTPIP Nearest To Point Interpolated Point

6) Static point

This adds a static point.

Fields: ***point***,<name>,<type>, ...

<name> is any sequence of symbols. A name **must not** contain a comma!
<type> is one of the codes from the list below

SPLL Static Point Latitude / Longitude
Fields: ***point***,<name>,**SPLL**,<lat>,<long> ,<alt>

<lat> is the latitude coordinate
<long> is the longitude coordinate
<alt> is the altitude (in feet)

SPCH Static Point in CH1903 coordinate system
Fields: ***point***,<name>,**SPCH**,<x>,<y> ,<alt>

<x> is the x coordinate
<y> is the y coordinate
<alt> is the altitude (in feet)

SPUTM Static Point UTM
Fields: ***point***,<name>,**SPUTM**,<latZone>,<longZone>,<easting>,<northing>,<alt> *(,<radius>)*

<latZone> is the latitude zone number
<longZone> is the longitude zone character
<easting> is the easting coordinate
<northing> is the northing coordinate
<alt> is the altitude (in feet)
<radius> is the radius to draw around the point on the map (in meter)

SPFT Static Point From Track
Fields: ***point***,<name>,**SPFT**,<PID>

<PID> is the PID of the given point from the track

SPFV Static Point First Valid
Fields: ***point***,<name>,**SPFV**

7) Point from a list

This adds a point from a list of other points.

Fields: ***point***,<name>,<type>,<point_1>,<point_2>, … , <point_N>

<name>	is any sequence of symbols. A name **must not** contain a comma!
<type>	is one of the codes from the list below
<point_x>	is the name of another previously defined point

LFNN List First Not Null

Returns the first point of the list of points that is not null.

LLNN List Last Not Null

Returns the last point of the list of points that is not null.

LCT List Closest To

Returns the point from the list of points which is closest to the track.

LCP List Closest Point

Returns the point from the list of points which is closest to <point_1>.

LFIT List First In Time

Returns the earliest points from the list of points.

LLIT List Last In Time

Returns the latest points from the list of points.

8) Logger Marks

This adds a point which comes straight from the logger.

Fields: ***point***,<name>,<type>,<number>, ...

<name> is any sequence of symbols. A name **must not** contain a comma!
<type> is one of the codes from the list below
<number> is the slot number where the mark has been made

MVMD Marker Virtual Marker Drop

- *Returns the first occurrence of the point where a virtual marker has been dropped.*
- *A warning is being output if a marke has been dropped more than one.*

Fields: ***point***,<name>,<type>,<number>,[<radius>]

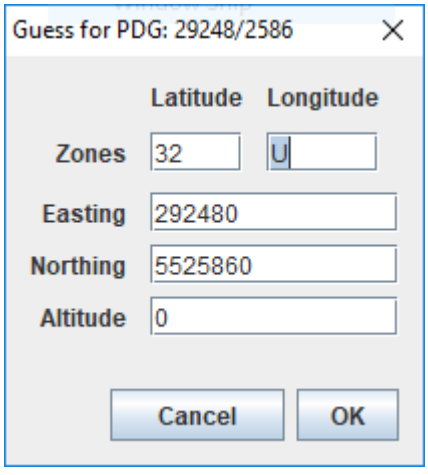
<radius> is the radius to draw around the point on the map (in meter)

MPDG Marker Pilot Declared Goal

- *Returns the last occurrence of the point the pilot has chosen.*
- *A point with the name "<Name>_declaration" will be added. This is the point where the pilot made his declaration. As position of the declaration the last track point just before the declaration is used.*
- *If the goal declared by the pilot is an already declared point, then a point with the name <Name> is declared as alias for the goal declared by the pilot.*
- *IIf no height is specified or if it is zero, the ground altitude of the indicated position is retrieved from the SRTM model available through a web service at: http://new.earthtools.org/aboutdata.htm*
- *A warning will be output if a declaration has been made more than once.*
- ***Boxy will try to guess the declared coordinates and will then pop up an editor where the user needs to confirm and adjust the declared goal coordinates.***

Fields: ***point***,<name>,<type>,<number>,[<webHeight>]

<webHeight> set to 1 to force retrieval of the ground altitude from the web *(only for MPDG)*



9) User input

This asks the user to enter a point manually.

Fields: ***point***,<name>,<type>,<init-values>

<name>	is any sequence of symbols. A name **must not** contain a comma!
<type>	is one of the codes from the list below
<init-values>	are the initial values the dialog is being filled with

UPUTM	User Point in UTM	
	The initial values taken in this order	
	<latZone>	is the latitude zone number
	<longZone>	is the longitude zone character
	<easting>	is the easting coordinate
	<northing>	is the northing coordinate
	<alt>	is the altitude (in feet)
UPCH	User Point in CH1903 format	
	The initial values taken in this order	
	<x>	is the x coordinate
	<y>	is the y coordinate
	<alt>	is the altitude (in feet)
UPLL	User Point in lat/long	
UPPID	User Point by PID	
	<message>	the message to be shown to the user

The user can enter a <NULL> point, which means that this point does not exist.

C) LABEL records

This prints only a label.

Fields: **label**,<name>,<text>

<name>	is any sequence of symbols. A name **must not** contain a comma!
<text>	is any sequence of symbols. This text **must not** contain a comma!

- *If you enter "-" as text, a horizontal line will be drawn.*
- *Instead of "somename = label(-)" you can use the simple alias "-"*

D) RESULT records

This calculates a result.

Fields: **result**,<name>,<type>,<firstPoint>,<secondPoint>

<name> is any sequence of symbols. A name **must not** contain a comma!
<type> is one of the codes from the list below
<firstPoint> is the name of another previously defined point
<secondPoint> is the name of another previously defined point

D2D Distance 2D (in meter)
Calculates the 2D distance between two points.

D3D Distance 3D (in meter)
Calculates the 3D distance between two points.

DP2P Distance Point to Point (in meter) – *always 2D*
Calculates the 2D distance "point to point" between the two points.

TSEC Time in Seconds
Determines the time in seconds between the two points.

TMIN Time in Minutes
Determines the time in minutes between the two points.

ATRI Area of Triangle (in km^2)

- *Calculates the area of the triangle given by the three points.*
- *Draws the triangle also on the map.*

Fields: ***result***,<name>,**ATRI**,<firstPoint>,<secondPoint>,<thirdPoint>

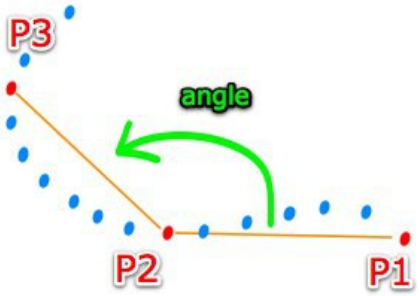
<thirdPoint> is the name of another previously defined point

ANG3P Angle between 3 Points (in °)

- *Calculates the angle between 3 points.*
- *Draws the triangle also on the map.*

Fields: ***result***,<name>,**ANG3P**,<firstPoint>,<secondPoint>,<thirdPoint>

<thirdPoint> is the name of another previously defined point

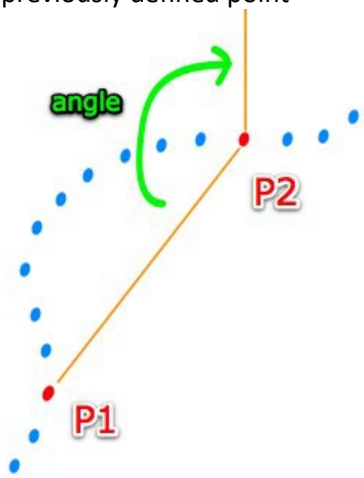


ANGTN Angle To the North

- *Calculates the angle to the north from two points.*
- *Draws the triangle also on the map.*

Fields: ***result***,<name>,**ANGTN**,<firstPoint>,<secondPoint>,<thirdPoint>

<thirdPoint> is the name of another previously defined point

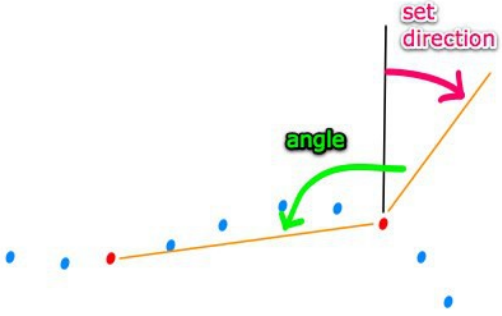


ANGSD Angle Set Direction

- *Calculates the angle between two points and a given set direction.*
- *Draws the triangle also on the map.*

Fields: ***result***,<name>,**ANGSD**,<firstPoint>,<secondPoint>,<setDirection>

<setDirection> is the given set direction



DAD Distance Altitude Dependent

- *Calculates the distance between two points depending on the altitude of the first given point. The altitude <absoluteAltitudeLimit> is an absolute altitude above sea level.*
- *If the altitude of <firstPoint> is below <absoluteAltitudeLimit>, the 2D distance will be calculated, else the 3D distance.*

Fields: ***result***,<name>,**DAD**,<firstPoint>,<secondPoint>,<absoluteAltitudeLimit>

<altitudeLimit> is an altitude (in feet)

RDAD Relative Distance Altitude Dependent

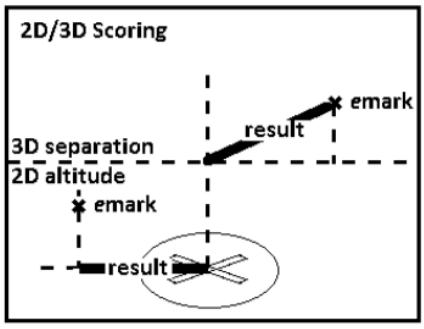
- *Calculates the distance between two points depending on the relative altitude between the first and the second given point.*
- Generally <firstPoint> is higher than <secondPoint>.
- *If the absolute value of the altitude of <firstPoint> minus the altitude of <secondPoint> is below <altitudeDifference>, the 2D distance will be calculated, else the 3D distance between <firstPoint>+<altitudeDifference> and <secondPoint>.*

Fields: ***result***,<name>,**RDAD**,<firstPoint>,<secondPoint>,<altitudeDifference>

<altitudeDifference> is the maximum altitude (in feet) allowed between the first and the second point

SDAD Separated Distance Altitude Dependent

- *Calculates the distance between two points depending on the altitude of the first given point. The altitude <absoluteAltitudeLimit> is an absolute altitude above sea level.*
- *If the altitude of <firstPoint> is below <absoluteSeparationAltitudeLimit>, the 2D distance will be calculated, else the 3D distance to the goals coordinates but using the separation altitude.*



Fields: ***result***,<name>,**SDAD**,<firstPoint>,<secondPoint>,<absoluteSeparationAltitudeLimit>

<absoluteSeparationAltitudeLimit> is an altitude (in feet)

E) CHECK records

Checks something.

Fields: **check**,<name>,<type>, ...

<name> is any sequence of symbols. A name **must not** contain a comma!
<type> is one of the codes from the list below

RESULT Result

Tests if a result is more or less a given static value.

Fields: ***check***,<name>,**RESULT**,<nameRes>,<operator>,<value>

<nameRes> is the name of another previously defined result
<operator> is one of the following (case insensitive):
- `more` more or equal
- `less` less or equal

<value> is the reference value (same unit as the unit of the compared result)

COMPARE Compare (same as AND and OR type, but I didn't remember having coded them ...)

Combine multiple results using a boolean operator

Fields: ***check***,<name>,**COMPARE**,<operator>,<check1>,<check2>, ...

<operator> is one of the following (case insensitive):
- `or`
- `and`

<check1> is the name of a previous made check
<check2> is the name of a previous made check
...

AREA Area

Tests if a point is inside or outside an area.

Fields: ***check***,<name>,**AREA**,<pointName>,<operator>,<filename>

<pointName> is the name of another previously defined point
<operator> is one of the following (case insensitive):
- `inside`
- `outside`

<filename> is the URI the area has to be loaded from (PLT, GPX or IGC)

POINT Result

Tests if the atribute of a point is more or less than the same value of another point or as a static value.

Fields: ***check***,<name>,**POINT**,<pointName>,<pointField>,<operator>,<value>[,<adjustment>]

<pointName> is the name of another previously defined point

<pointField> is one of the following (case insensitive):

- `latitude`
- `longitude`
- `easting`
- `northing`
- `altitude` (in feet)
- `time` (format = hh:mm:ss)
- `distance`

<operator> is one of the following (case insensitive):

- `more` more or equal
- `less` less or equal

<value> is the name of another point (if defined)

- or elsewise -

is the reference value (same unit as the unit of the compared point field)

in case of a distance it must be the reference to a point

<adjustment> is the adjustment to be made to the <value> before doing the check

- in case of time, the <adjustment> represents minutes, otherwise the indicated units
- in case of a distance, the adjustment is mandatory and used as reference value

~~***----- Temporarily removed! ----***~~

~~**PZ** PZ checks~~

~~*Checks for a PZ infringement (see PZL format definition file for more information)*~~

~~Fields: ***check***,<name>,**PZ**,<PZLfilename>~~

~~<PZLfilename> is the URI to the PZL file where the PZ's are defined in~~

VSPEED PZ checks

Fields: ***check***,<name>,**VSPEED**,<limit>,<sensitivity>

<limit> is the vertical speed limit (feet/minute)

< sensitivity> is the sensitivity time interval for the check (seconds)

AND logical “and” operator

Checks if all listed checks are fulfilled

Fields: ***check***,<name>,**AND**,<check_1>,<check_2>, … , <check_N>

<check_x> is the name of another previously defined check

OR logical “or” operator

Checks if at least on of the named checks is fulfilled

Fields: ***check***,<name>,**OR**,<check_1>,<check_2>, … , <check_N>

<check_x> is the name of another previously defined check

F) MAP records

Allows to add some extra information on the map.

Fields: **map**,<name>,<type>, ...

<name> is any sequence of symbols. A name **must not** contain a comma!
<type> is one of the codes from the list below

AREA Area
Fields: ***map***,<name>,**AREA**,<filename>,<color>

<filename> is the URI the area has to be loaded from (PLT, GPX or IGC)
<color> is the color the area has to be drawn in

POINTLL Point Latitude / Longitude
Fields: ***point***,<name>,**POINTLL**,<lat>,<long> ,<alt>

<lat> is the latitude coordinate
<long> is the longitude coordinate
<alt> is the altitude (in feet)

POINTCH Point in CH1903 format
Fields: ***point***,<name>,**POINTLL**,<x>,<y> ,<alt>

<x> is the x coordinate
<y> is the y coordinate
<alt> is the altitude (in feet)

POINTUTM Point UTM
Fields: ***point***,<name>,**POINTUTM**,<latZone>,<longZone>,<easting>,<northing>,<alt>,<radius>

<latZone> is the latitude zone number
<longZone> is the longitude zone character
<easting> is the easting coordinate
<northing> is the northing coordinate
<alt> is the altitude (in feet)
<radius> is the radius to draw around the point on the map (in meter)

G) SET records

*Allows to set different options. A set record has **no** name, thus it's extended syntax is somewhat simpler.*

Fields: **set**,<key>,<value>

Extended: **set**(key,value)

<key> is the name of the option to change
<value> is the value to assign to the option

***set*,grid,utm**	to make the script output points in UTM format (default)
***set*,grid,latlong**	to make the script output points in latitude / longitude format
***set*,grid,ch**	to make the script output points in CH format
***set*,output.height,feet**	to make the script output any height in feet (default)
***set*,output.height,meter**	to make the script output any height in meter
***set*,qnh,**<value>	sets the QNH and reloads the file with corrected values (IGC only!)

H) Examples / Script snippets

1) Maximum distance (last in / last out)

```
Title = label(Maximum distance)
A = point(ALIIP,c:/Tasks/areaOne.plt,09:00:00,0,4000)
B = point(ALOIP,c:/Tasks/areaOne.plt,09:00:00,0,4000)
distance = result(DP2P,A,B)
```

Point “A” is here to be defined as the last entry point in the area described by the given filename. Point “B” is the last exit point. Only track-points before the end of the scoring period (9:00 o'clock) an beneath a maximum altitude of 4000 feet are being considered. For this example, both points will no reflect existing track-points (as recorded by the GPS) but interpolated points which will be the real intersection between the track and the area. The resulting distance is the sum of all distances between the track points that are in-between the points “A” and “B”.

2) Hesitation waltz (nearest fly-by point)

```
Title = label(Hesitation waltz)
G020 = point(SPUTM,32,U,309590,5523270,0)
G032 = point(SPUTM,32,U,310720,5523220,0)
G056 = point(SPUTM,32,U,306080,5522780,0)
GOAL = point(LCT,G020,G032,G056)
NEAREST = point(NTPTP,GOAL,21:00:00)
distance = result(RDAD,NEAREST,GOAL,500)
```

To analyze this scenario, we start to define the three possible goals that pilots could have flown to (G020, G032 & G056). Next we need to get the one that the pilot was closest to (GOAL) and find the nearest point to this goal (NEAREST). The resulting distance is the distance 2D from the point “NEAREST” to the point “GOAL” if “NEAREST” is within an altitude of 500 feet above “GOAL”. If this is not the case, the resulting distance is the distance 3D from the point “NEAREST” to the point “GOAL”.

3) Hesitation waltz (with virtual marker drop)

```
Title = label(Hesitation waltz – virtual marker drop)
G020 = point(SPUTM,32,U,309590,5523270,0)
G032 = point(SPUTM,32,U,310720,5523220,0)
G056 = point(SPUTM,32,U,306080,5522780,0)
VMD1 = point(MVMD,1)
NEAREST_GOAL = point(LCP,VMD1,G020,G032,G056)
distance = result(RDAD,VMD1,NEAREST_GOAL,500)
```

To analyze this scenario, we start to define the three possible goals that pilots could have flown to (G020, G032 & G056). Next we need to get the point where the virtual marker number 1 has been dropped (VMD1) and find out which is the closest target (NEAREST_GOAL). The resulting distance is the distance 2D from the point “VMD1” to the point “NEAREST_GOAL” if “VMD1” is within an altitude of 500 feet above “NEAREST_GOAL”. If this is not the case, the resulting distance is the distance 3D from the point “VMD1” to the point “NEAREST_GOAL”.

4) Judge declared goal (with virtual marker drop)

```
QNH = set(1019)
Title = label(*** Task #1 - Judge Declared Goal)
G200 = point(SPUTM,32,U,313803,5519762,554)
VMD1 = point(MVMD,1)
distance = result(RDAD,G200,VMD1,500)
timecheck = check(POINT,VMD1,time,less,21:15:00)
distancecheck = check(result,distance,more,77)
MT = label(MT: )
-
```

So the first thing that is done here is that the QNH is being set to 1019. Next a title is being output. The declared goal is in Echternach on the island and the pilots have to drop a virtual marker. We are here using slot #1. The time of the button push has to be before 21:15 o'clock and the distance has to be over 77 m, because this was the maximum possible distance for a real marker drop (which would have taken precedence over the virtual marker drop). Finally a place for the measured result from the measuring team is let free before a horizontal line is being output.

5) Angle (with set direction and virtual marker drop)

```
Title = label(*** Task #15 - Angle)
VMD3 = point(MVMD,3)
VMD4 = point(MVMD,4)
angle = result(ANGSD,VMD3,VMD4,160)!
distance = result(D2D,VMD3,VMD4)
distancecheck = check(result,distance,more,500)
ordercheck = check(point,VMD3,time,less,VMD4)
timecheck = check(point,VMD4,time,less,08:30:00)
outsidecheck = check(AREA,N:\CLP-Field.igc,outside,VMD3)
allchecks = check(AND,ordercheck,timecheck,distancecheck,outsidecheck)!
split = label(Has ths angle *not* been split?     YES    /    NO)
```

First of all, this script snippet outputs a label before it tries to read the electronic marks from slot #3 and #4. On the next line, the angle between the two points and the set direction of 160° is calculated. Because of the "!" at the end of this line, it is highlighted in blue on the script report.

The next few lines do some important checks:

- Calculate the distance between the two logger marks and check that it is at least 500 meters.
- Check that the marker #3 has been registered before marker #4.
- Make sure that the virtual marker #4 has been dropped before 8:30 o'clock. Because of the "ordercheck", there is no need to repeat this test for marker #3.
- Check that the logger mark #3 has been made outside of the common launch point field.
- Combine those 4 checks with a logical AND operator and highlight this line.

The last line just does an output which the scorer / analyzer has to fill out manually, namely to check that the task has not been split, so that there is no other mark in between logger mark #3 and #4. This could easily be verified on the map. An automation of this check is possible but quite long and complex to script.

6) Fly on (with virtual goal declaration an virtual marker drop)

```
Title = label(*** Task #3 - FON)
079 = point(SPUTM,32,U,314569,5520242,786,1000)
117 = point(SPUTM,32,U,316162,5522690,842,1000)
119 = point(SPUTM,32,U,318996,5524034,868,1000)
VGD1 = point(MPDG,1)
VMD3 = point(MVMD,3)
distance = result(RDAD,VGD1,VMD3,500)
timecheck-drop-time = check(POINT,VMD1,time,less,21:00:00)
timecheck-declartion-time = check(POINT,VGD1_declaration,time,less,VMD2)
distancecheck-mma = check(result,distance,more,50)
```

In this task goal declaration #1 had to be done before the virtual marker drop #2.

First of all, the three points for the fly on are being declared. Notice that you **must** use the same three digits as names for the points as the pilots are going to use as declaration. Next the virtual goal #1 declaration is being extracted as well as the virtual marker drop #3.

The next three checks make sure that the virtual marker drop #3 has been dropped before the end of the scoring period, that the virtual goal declaration #1 had been done before the virtual marker drop #2 and that the obtained result is outside the MMA.

Please notice the difference between the "virtual goal declaration" (name = VGD1_declaration) and the "virtual declared goal" (name = VGD1).

7) Judge declared goal (with open / closing times for goals)

```
Title = label(---> Task 22 JDG)!
T101 = point(SPUTM,32,U,288116,5520127,935,1500)
T103 = point(SPUTM,32,U,287764,5520094,988,1500)
VMD2 = point(MVMD,2)
T101-validity = point(VTDTP,VMD2,T101,00:00,14:59,30:00,44:59)
T103-validity = point(VTDTP,VMD2,T103,15:00,29:59,45:00,59:59)
TARGET-valid = point(LFNN,T101-validity,T103-validity)
distance = result(SDAD,VMD2,TARGET-valid,1500)
outMMA = check(RESULT,distance,more,50)
scoring period check until = check(POINT,VMD2,time,less,08:00:00)
```

In this task, two goals were defined where each goal had a different opening time:

- T101 was valid from xx:00:00 to xx:14:59 and from xx:30:00 to xx:44:59
- T103 was valid from xx:15:00 to xx:29:59 and from xx:45:00 to xx:59:59

Virtual marker #2 is used to make the drop.

The **VTDTP** function is used to check the validity of the time of the point **VMD2**. If it is valid, the newly defined point will be a copy of the point **T101**. Otherwise is will be ***NULL***. This is used to define the two points "T101-validity" and "T103-validity".

The point "TARGET-valid" will be the first point which is not null from the given list of points. Seen that one of the previously defined points will be set to null (as it is will not be valid), the remaining valid point will be held by "TARGET-valid".

Finally the distance from the **VMD2** to this valid point is being calculated.