

Lorby “Axis And Ohs” Documentation

<http://www.axisandohs.com>

Version 4.30 - 02.07.2024

© 2021 LWR Inc.

1	Overview	6
2	Installation	7
3	Operations	
3.1	Starting the application	11
3.2	Main Window	12
3.3	Main Menu	13
3.4	Master configuration and clones	15
3.5	Control Layers	17
3.6	Locking the GUI controls	19
3.7	Handling joystick axis assignments	
3.7.1	Assign a new axis	20
3.7.2	Joystick Axis Sensitivity	23
3.7.3	Calibrate an axis	24
3.7.4	Axis advanced mode	25
3.7.5	Change an axis assignment	26
3.7.6	Assign an axis to a simulator variable	27
3.7.7	One shot and virtual events	27
3.7.8	Assign combo	29
3.7.9	Copy or Remove axis	30

3.8	Handling button, MIDI and keyboard assignments	
3.8.1	Assign a new button	31
3.8.2	Button actuation parameters	32
3.8.3	Sending values to events	33
3.8.4	Virtual Events	33
3.8.5	Assign combo	34
3.8.6	The button assignment control on the main list	35
3.8.7	Change a button assignment	36
3.8.8	Copy or Remove button assignments	37
3.9	Using the Event Selection boxes	38
3.10	Using Templates	39
3.11	Panel view modes	44
4	Voice Recognition	45
5	Scripting	57
6	Using other script languages than RPN	99
7	Mouse Yoke	104
8	Enhanced Power Management	106
9	Saitek Panels (Radio, Multi, Switch, BIP, FIP)	107

10	Desktop FIPs	113
11	Web FIPs	116
12	App and Web Windows	119
13	Disable simulator controllers	121
14	Hardware change	123
15	Device Blacklist	125
16	Config files, database and log files	127
17	PMDG Aircraft with AxisAndOhs	128
18	MIDI Out	135
19	Web API	137
20	Using the WebAPI as a web server	147
21	Importing Event and Variable lists	152
22	TextToSpeech: WinRT vs. SAPI	153
23	Advanced TextToSpeech: ChatGPT, Azure, Polly	155
24	RPN script files	159
25	Interactive Checklist	161
26	Wear & Tear simulation (experimental)	166
27	vJoy Interface	173
28	ViGEm Interface	174

29	Virtual mouse	175
30	CAN Interface	176
31	Sound effects for sound and speech output	179
32	ChatGPT interface	182
33	Command line parameters	185

1. Overview

The Lorby-SI “AxisAndOhs” app is designed to manage your joysticks and other controllers individually and automatically for each aircraft that you fly in your simulator. AxisAndOhs can remember each controller assignment down to each aircraft livery. If you already have an assignment for another aircraft of that type in the database, AxisAndOhs will assume and apply the same assignments.

All joystick movements are then routed through the app and control your aircraft in the simulator directly. No other control assignments are required, neither in-sim nor from an external module.

Lorby “Axis And Ohs” is compatible with

FSX boxed with Acceleration

FSX:SE

P3D 2.5

P3D 3.x

P3D 4.x

P3D 5.x

P3D 6.x

2. Installation

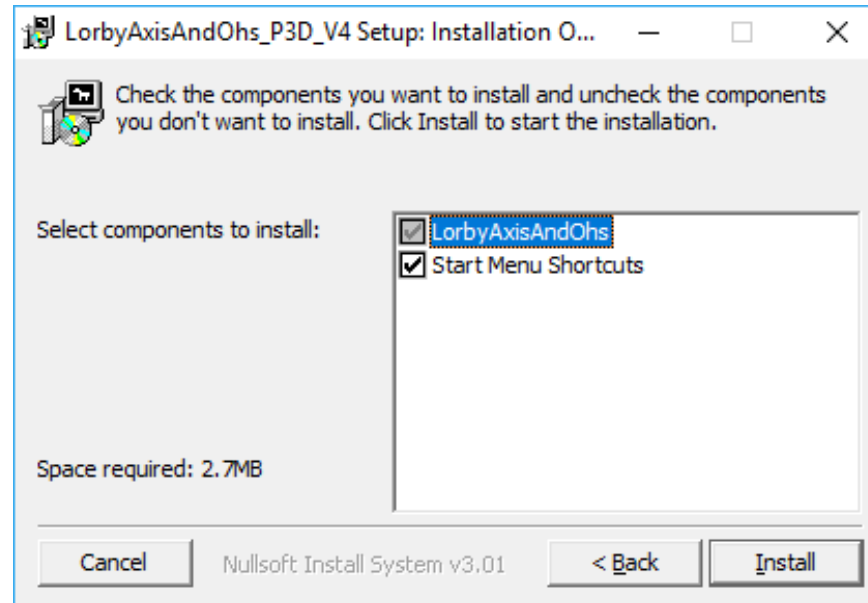
2.1 Distribution

Lorby AxisAndOhs is distributed as a self-extracting installer package.

2.2 Installation

- This application requires the .Net 4.8 Runtime and the VC++ Redist 2019 to be present on your computer
<https://dotnet.microsoft.com/download/dotnet-framework/net48>
<https://support.microsoft.com/en-us/help/2977003/the-latest-supported-visual-c-downloads>
- Please use the installer intended for your sim:
 - FSX Acceleration boxed or dual install with SE: *LorbyAxisAndOhs_Install.exe*
 - FSX SE stand alone: *LorbyAxisAndOhs_SE_Install.exe*
 - Prepar3D V2.5: *LorbyAxisAndOhs_P3D_Install.exe*
 - Prepar3D V3.x: *LorbyAxisAndOhs_P3D_V3_Install.exe*
 - Prepar3D V4.x: *LorbyAxisAndOhs_P3D_V4_Install.exe*
 - Prepar3D V5.x: *LorbyAxisAndOhs_P3D_V5_Install.exe*
 - Prepar3D V6.x: *LorbyAxisAndOhs_P3D_V6_Install.exe*

Running the installer:

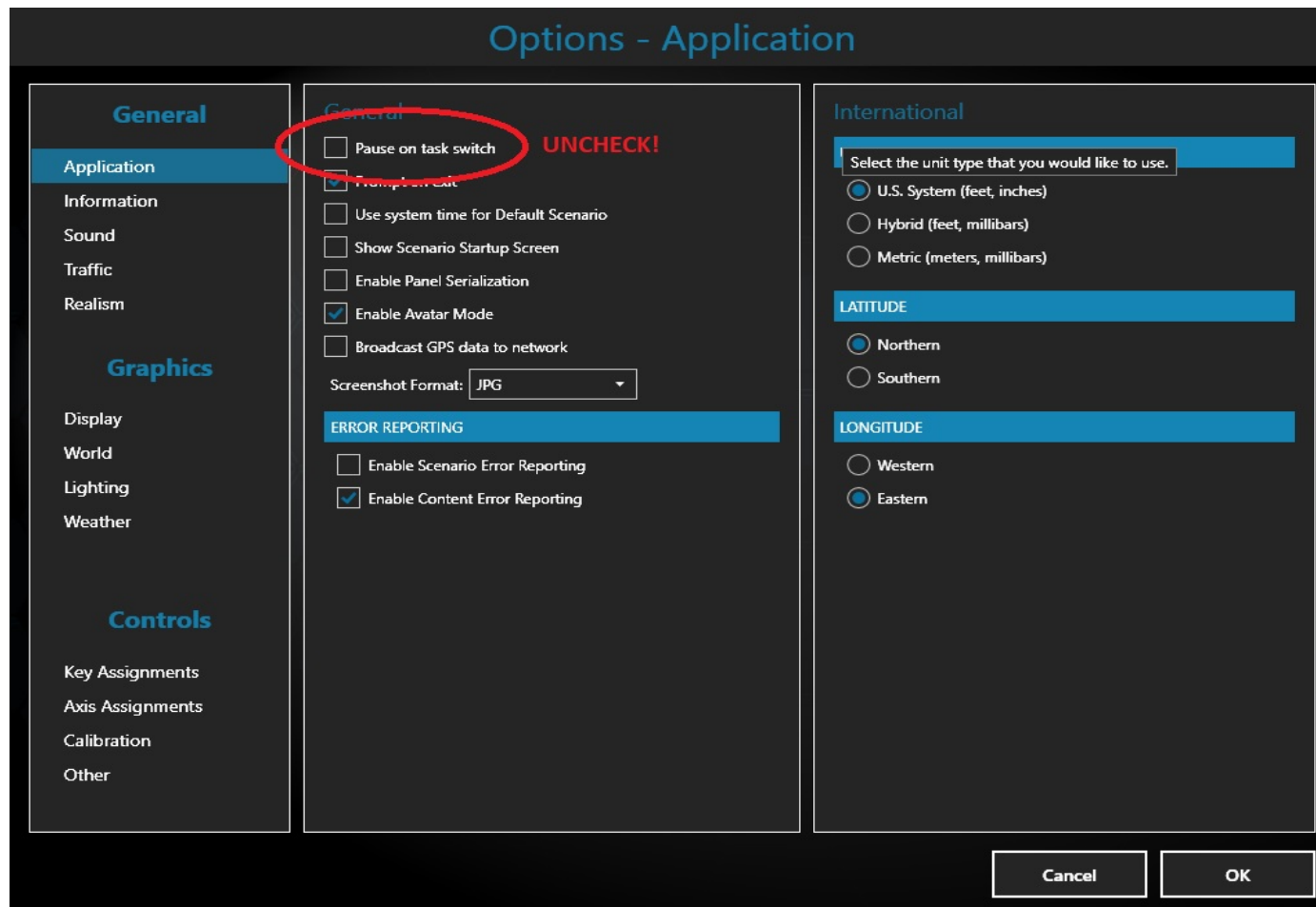


On the first page you may select optional installation targets:

- “Start Menu Shortcuts”: Lorby AxisAndOhs will be added to your Start Menu (advised)
- Selecting “Install” will begin the installation

2.3 Pause on task switch

Lorby AxisAndOhs is an external process. When you want to operate it from its own GUI, you must disable “Pause on task switch” in the simulator.



2.4 SimConnect

Lorby AxisAndOhs relies on SimConnect being installed correctly on your computer. SimConnect is a part of your simulator and it is set up automatically when you first install the simulator. No further installations are required.

FSX only: In case SimConnect is not installed, and Lorby AxisAndOhs does not start up, giving you an error message instead, you will have to install SimConnect manually:

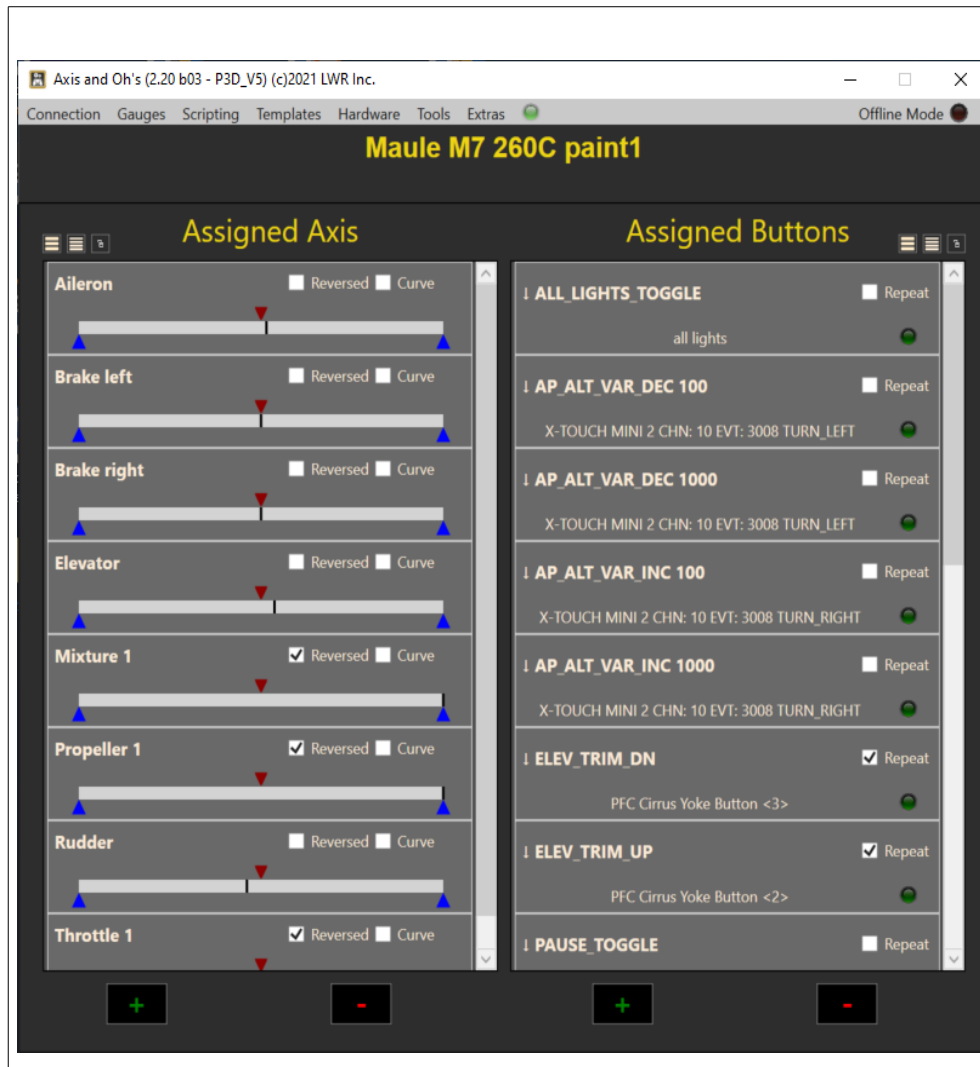
- FSX boxed users can find the “SimConnect.msi” installation file either online or in the FSX SDK folder “..\Microsoft Flight Simulator X SDK\SDK\Core Utilities Kit\SimConnect SDK\lib”
- FSX SE users find it here: “..\Steam\steamapps\common\FSX\SDK\Core Utilities Kit\SimConnect SDK\LegacyInterfaces\FSX-SP1\SimConnect.msi “

3. Operations

3.1 Starting the application

- Start your simulator
- Start the Lorby AxisAndOhs app
- Click on the green LED in the top menu bar
or open the “Connection” menu and select “Connect”
or set the app to “Connect automatically” in the same menu
- As soon as your aircraft has been detected and the simulation is running, you can start adding axis controls and buttons. The LAAO app will remember these settings for each aircraft
If the app doesn't detect the aircraft automatically when you are sitting in the cockpit, please use „Force Connection to running sim“ in the „Connection“ menu.
- To avoid conflicts with the controller settings in the simulator, the app has features to disable controllers in the sim. You can also disable all controllers in the Controls settings of the sim, but that will also disable mouse look and other mechanisms

3.2 Main Window



- **Assigned Axis:** list of all joystick axis that you have assigned to this aircraft

+/-: to **add** or **remove** an axis assignment.

- **Assigned Buttons:** All or your button assignments (Joystick, MIDI and keyboard). that you added to this aircraft

+/-: to **add** or **remove** a button assignment.

Assignments for controllers that are not attached will be highlighted in red



Spinning the mouse wheel over the gap between the two lists will change the size of the window. A right click on the gap resets the size back to the default.

3.3 Main menu

- **Application**

General settings

Configure the app to connect and/or minimize automatically upon launch

Configure the app to minimize to the system tray instead of the task bar

Start the sim automatically before connecting

Start the app in Offline Mode, with or without a preselected configuration

- **Gauges**

Activate Desktop and Web flight instruments

(for WebFIPs the app must be started „As Administrator“)

- **Scripting**

Manage/edit RPN scripts

- **Templates**

Assign an existing configuration to your current aircraft

Create and manage templates for aircraft configurations

- **Hardware**

Connect and manage Saitek devices, Connect MIDI devices, manage mouse sensitivity, device blacklist and hardware changes

- **Tools**

Save the assignment database now, manage EPM, alter Web ports, find a button

- **Extras**

Special functionality normally not found in the simulator

- **SimConnect Mode - Green LED**

Connect to / disconnect from the simulator by clicking on the LED

- **Offline Mode - Red LED**

In Offline Mode AAO will not connect to the simulator, but the input devices are all working. You can load a configuration or template manually or create a new one and operate it. Be mindful that the simulator variables and events are not available in this mode.

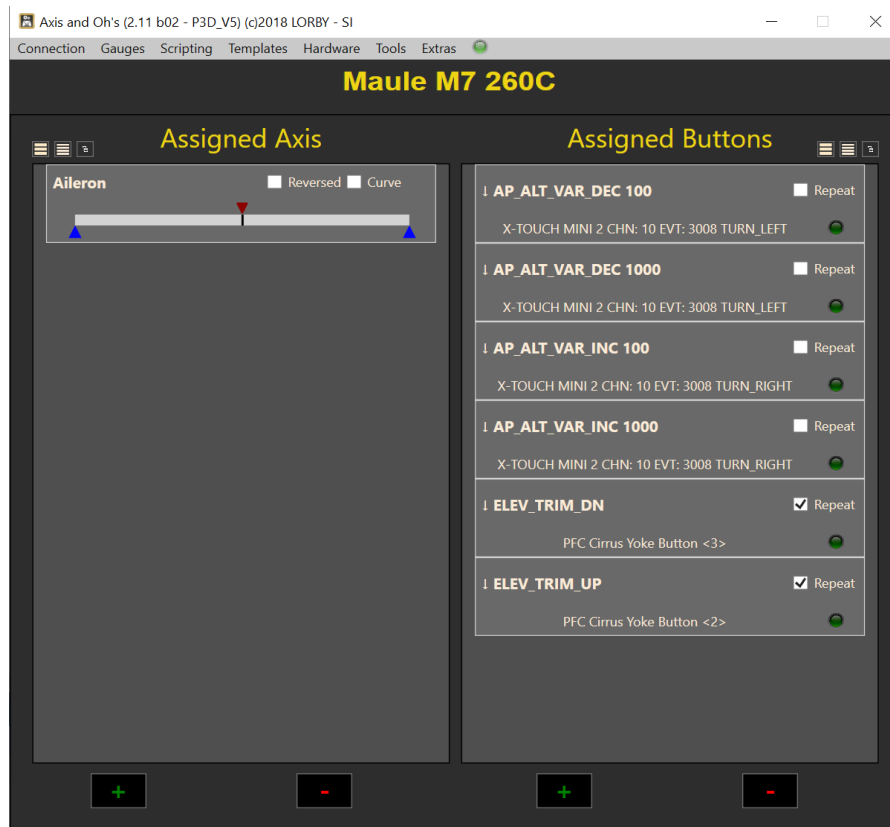
In offline mode, the app can be used as a "bridge" between your input devices and any other software that accepts joystick or game pade input, using the vJoy and ViGEm interfaces.

For example, you could use your flightsim yoke as the steering wheel in SnowRunner, although the game only accepts xBox controller input. Simply assign your yoke axis to the appropriate ViGEm axis.

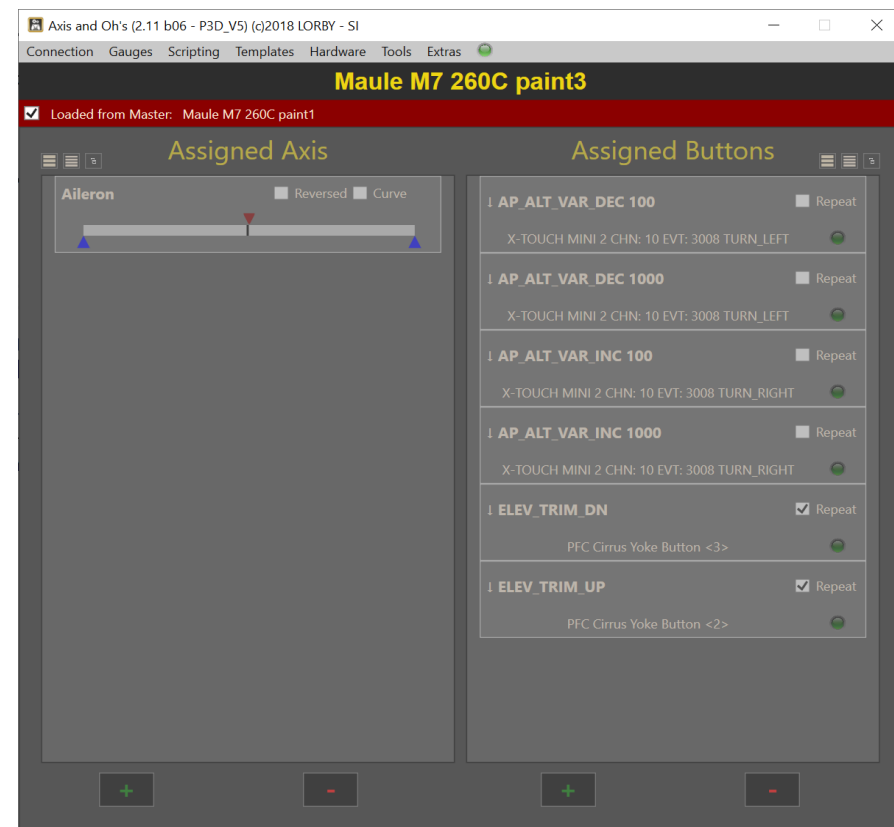
- to automatically start in "Offline Mode" enable "Application->Automatic offline mode"
- to select a specific configuration to load automatically, load it in AAO first, then enable the "Application->Load config:" option.

3.4 Master configuration and clones

The first configuration that you create for an aircraft is assumed to be the „master“. If you then load the same plane but with a different livery, AAO will assume that you want to use the same controls as with the master.



MASTER



CLONE

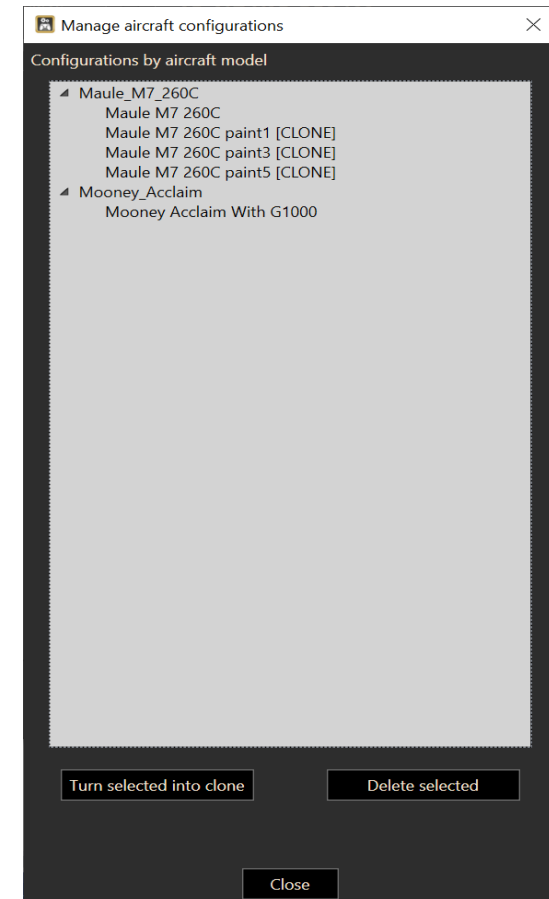
When a configuration is „cloned“ like above, all Axis and Buttons assets are locked – you can only change them on the „master“ configuration!

If you want to save a separate configuration for the plane, deactivate the checkbox in the red bar. AAO will then reload the config as a separate entity.

To reset a configuration, use „Templates → Clear current config“

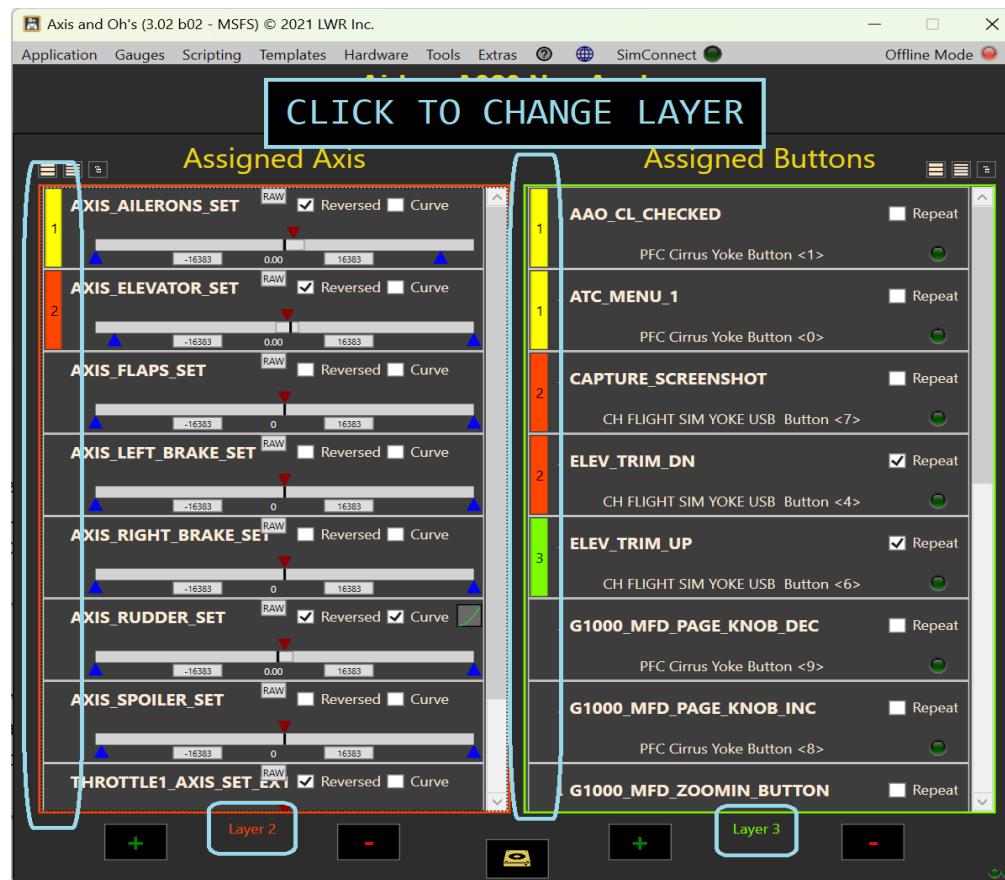
To manage the master/clone relationships use „Templates → Edit configs“

- Every aircraft must have at least one „master“ configuration
- The others can be turned into „clones“ with the button below the list.



3.5 Control Layers

Buttons and Axis definitions can be arranged in 4 layers. A button or an axis will only „fire“ when you select the corresponding layer in AAO. Axis and Buttons are split into separate layer sequences. Buttons and axis that are assigned only to the default layer „0“ will always fire, regardless of which Control Layer is currently set.



Layer 0 – default, no color: button or axis will always fire (even when another layer has been selected for AAO), unless they are also present on other layers.

Layer 1 – yellow, 2 – red, 3 – green, etc: button or axis will only fire when this Control Layer has been selected for the AAO app. The first three layers are hard coded. You can create additional layers with the dialog "Templates->Manage Control Layers"

To select a layer, you can

- either click on the labels below the axis and button panels
- or click on the left most section of the button or axis panel
- or use the following AAO events in scripts or assigned to buttons:

```
1 (>K:AAO_BUTTON_LAYER_UP), 1 (>K:AAO_BUTTON_LAYER_DOWN), 0..3 (>K:AAO_BUTTON_LAYER_SET)
1 (>K:AAO_AXIS_LAYER_UP), 1 (>K:AAO_AXIS_LAYER_DOWN), 0..3 (>K:AAO_AXIS_LAYER_SET)
```

The UP and DOWN events will only cycle through the active layers (=those that have buttons on them)

You can change the name of a layer by right-clicking on the label. A layer can only have one label and it will be the same for axis and buttons.

The current layer can be queried from internal LVars: (L:AaoButtonLayer) and (L:AaoAxisLayer).

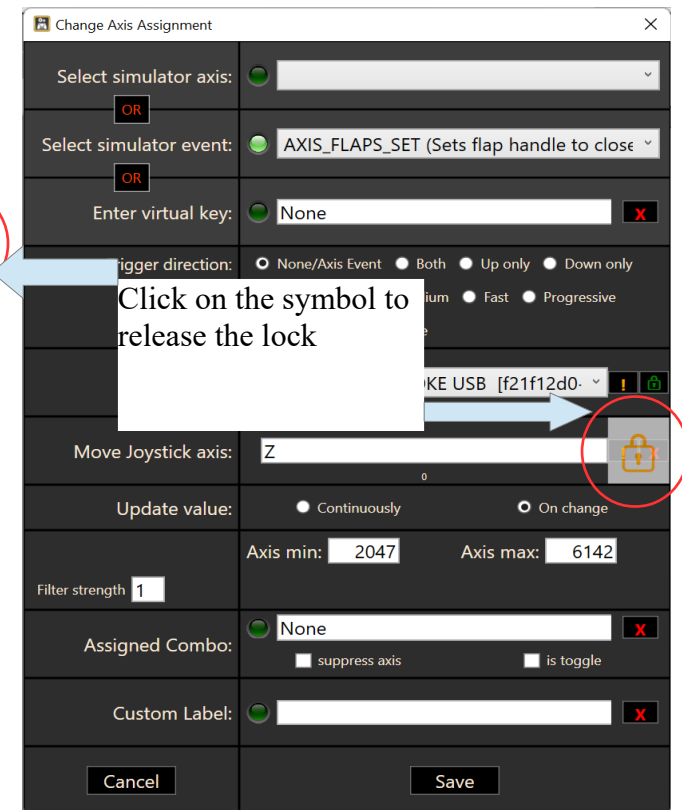
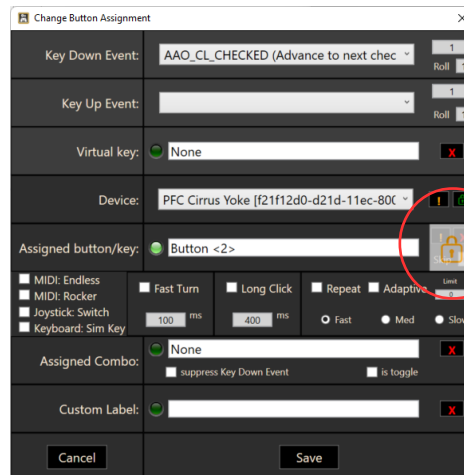
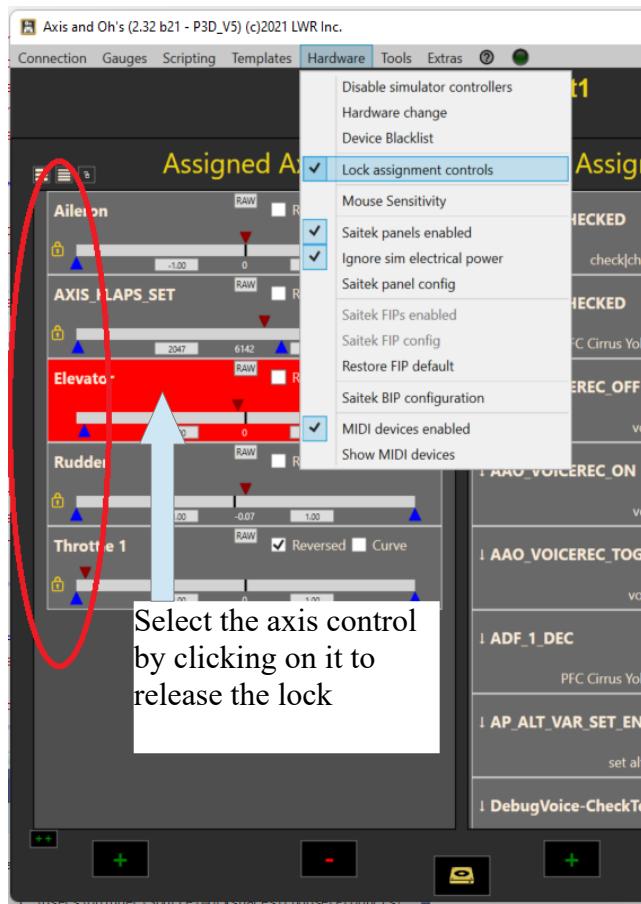
The current axis and button labels can be exported into LVars using the AAO events

1 (>K:AAO_EXPORT_BUTTON_LABELS) and 1 (>K:AAO_EXPORT_AXIS_LABELS)

The resulting LVars are named (L:AAO_<AXIS|BUTTON>_LABEL_<layer>_<index>, String)

3.6 Locking the GUI controls to prevent accidental changes

The direct axis controls and the Add/Change Axis/Button dialogs can be locked with „**Hardware → Lock assignment controls**“. This prevents accidental changes with the mouse or through „noisy“ controller hardware



3.7 Handling joystick axis assignments

3.7.1. Assign a new axis

Add Axis Assignment

Select simulator axis:

OR

Select simulator event:

OR

Enter Variable:

OR

Enter virtual event:

None

Value range: Axis min: -16384 Axis max: 16384

Update value:

Continuously

On change

 Rounding:

Float

Int

Trigger direction:

None/Axis Event

Both

Up only

Down only

Repeat:

Off

Slow

Medium

Fast

Progressive

Mode:

Momentary

Toggle

Device:

PFC Cirrus Yoke [f21f12d0-d21d-11ec

Move Joystick axis:

Z

Assigned Combo:

None

suppress axis

is toggle

Custom Label:

Cancel

Add

The top area controls the OUTPUT of the assignment.

Events coming in from your input device are translated into simulator data and are sent to the sim.

You can choose either an axis variable or a simulator event to control actions directly in the simulator, like Ailerons or Throttles. Not every axis variable or event work on every aircraft, you may have to try a few axis and events until you find the correct one.

When you choose an event, the "Trigger direction" box becomes visible

Alternatively you can enter a variable name directly or choose to send a virtual event.

The bottom area shows the INPUT of the assignment.

The "Device" and "Joystick axis" boxes are empty initially and will be populated with the input event that the app receives when you move your external controller. You can assign a "Combo" button to this axis, so the output action is only actuated when that button is/has been pressed - or not.

OUTPUT

The screenshot shows a configuration window for a simulator. It has several sections with radio buttons and dropdown menus. Colored circles and arrows highlight specific options: a purple circle around 'Select simulator axis', a green circle around 'Select simulator event', a red circle around 'Enter Variable', a blue circle around 'Enter virtual event', and a yellow circle around the 'Value range' and 'Update value' sections. A large green arrow points from the 'Select simulator axis' and 'Select simulator event' options to the text 'Select either one of the pre-configured axis variables'. A yellow arrow points from the 'Update value' section to the text 'This controls how the axis position is sent to the simulator:'. A red box at the bottom left contains text about variable names. A yellow box at the bottom right contains text about axis position and rounding.

Select simulator axis: ☐ [dropdown]

OR

Select simulator event: ☐ [dropdown]

OR

Enter Variable: ☐ [text box] +

OR

Enter virtual event: ☐ None X

Value range: Axis min: -16384 Axis max: 16384

Update value: ☐ Continuously ☐ On change Rounding: ☐ Float ☐ Int

Trigger direction: ☐ None/Axis Event ☐ Both ☐ Up only ☐ Down only

Repeat: ☐ Off ☐ Slow ☐ Medium ☐ Fast ☐ Progress

Mode: ☐ Momentary ☐ Toggle

Select either

one of the pre-configured axis variables

OR

A simulator axis type event

OR

enter a variable name

OR

select a virtual event to send

The **variable name** must be entered without brackets

Examples:

L:simulator\varname, Number

A:AILERON POSITION, Position

L:aa\varname

This controls how the axis position is sent to the simulator:

Continuously = all the time

On change = only when you move the lever

The controller input is mapped to the numerical range between Min and Max. The boxes can be changed with the mouse wheel. When you double click on them, you can enter the value with your keyboard

The axis output can be rounded to the nearest integer value if necessary

INPUT

Device:	PFC Cirrus Yoke [f21f12d0-d21d-11ec ▾] ! 🔒
Move Joystick axis:	Z 65535 ! X
Assigned Combo:	<input checked="" type="radio"/> None X <input type="checkbox"/> suppress axis <input type="checkbox"/> is toggle
Custom Label:	<input checked="" type="radio"/> X



Move the desired joystick axis until it shows up in the textboxes

You can **ignore** an axis or the entire controller with the exclamation marks (right click to reset)



You can lock to a specific controller with the „**lock**“ symbol

Please remember that the "Device" combobox is empty initially. Only when you move your external axis control it will be populated with the data from the incoming event.

The boxes together with the exclamations marks and the lock can be used to isolate or ignore devices and axis, should they interfere with normal pickup. **Using the lock to isolate a device will set the input to "high sensitivity" mode**, which enables you to pick up controls that send very small increments when being moved.

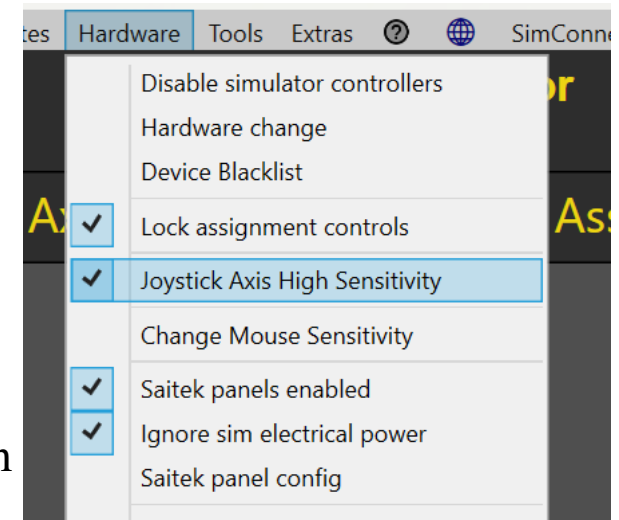
3.7.2. Joystick Axis Sensitivity

In normal operation, AxisAndOhs queries the joystick interface at a rate of 18Hz. It will also only process the last update of a joystick axis position that is transmitted from the game controller and disregard all others. This ensures fast operation without taxing the computer and the interface too much.

If you require more precision, you can activate the "Joystick Axis High Sensitivity" mode in the "Hardware" menu.

- In this mode the update rate is increased to 36Hz
- All updates coming from the game controller will be processed, (=not only the latest one, but all in between too)

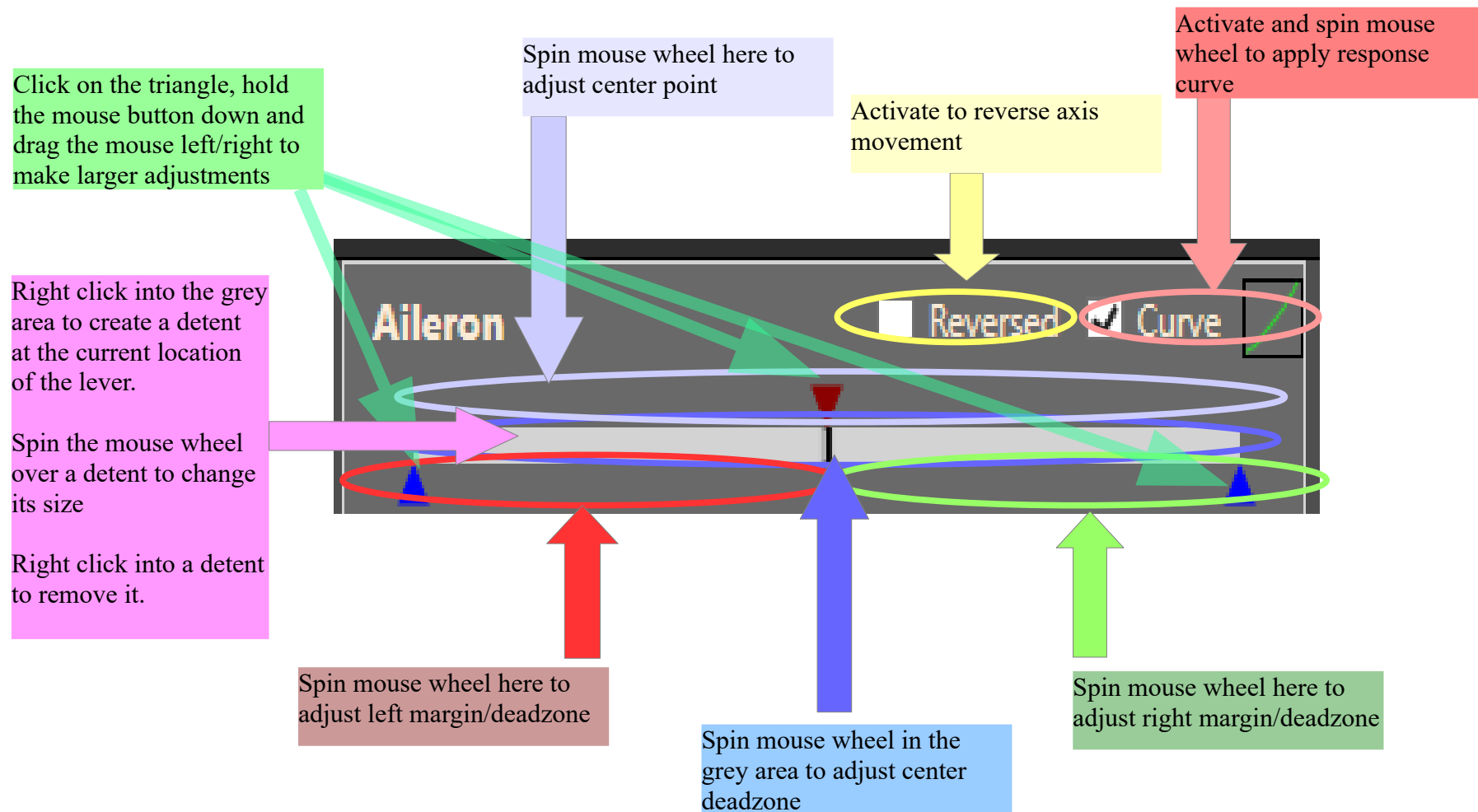
Please note that the high sensitivity mode requires a powerful computer. If you notice the actual controls in the sim lagging behind your controller input, then the computer cannot keep up with the high update rate.



High sensitivity can be enable or disabled at any time. You can also use the following AAO events:
(>K:AAO_HIGH_SENSITIVITY_ON), (>K:AAO_HIGH_SENSITIVITY_OFF),
(>K:AAO_HIGH_SENSITIVITY_TOGGLE) assigned to a button or script.

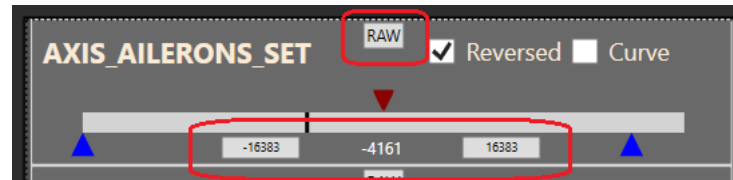
3.7.3. Calibrate an axis

Every assigned axis is calibrated directly on the main dialog, by hovering your mouse cursor over the various controls and turning your mouse wheel.



3.7.4. Axis advanced mode

To enable precise changes of the main axis parameters, activate „**Tools->Axis advanced mode**“.
With this option the axis control will show the current axis value that is sent to the sim, the current axis minimum and maximum, and an additional button „RAW“

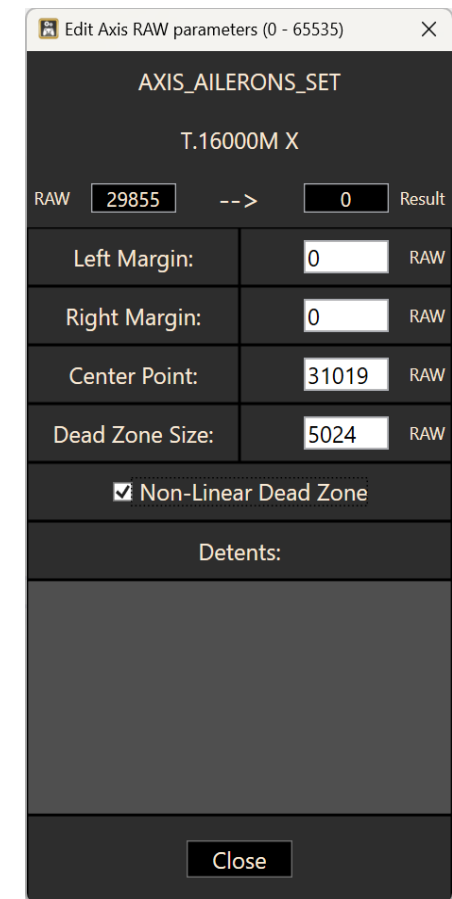


To set axis min and max, move the axis until it shows the desired value at the bottom (center), then click on the min or max numbers to the left/right.
To reset the value, right-click on the min or max number.

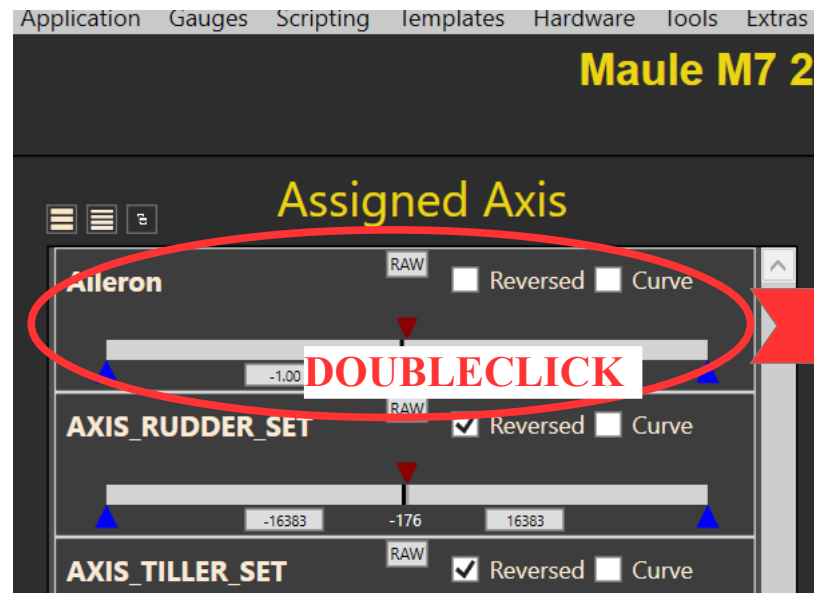
With the „RAW“ button you can call up an editor dialog for the main axis parameter values. The white numerical boxes can be changed with the mouse wheel or doubleclick.

The "Non-Linear Dead Zone" option determines if the axis values will be mapped onto the remaining range without the deadzone or if the deadzone is "cut" from the range.

The „Result“ is the value that is sent to the selected variable or event in the simulator



3.7.5. Change an axis assignment



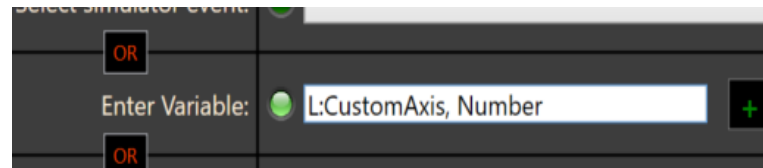
On this dialog you can change the assignment itself and all axis parameters.

Special notes:

The "Input ranges" correspond to the red and blue triangles of the axis visualization.

A "Filter strength" value above 1 forces an averaging function on the axis to iron out "fluttering" potentiometers.

3.7.6. Assign an axis to a simulator variable



If you want to control the value of a simulator variable with the assigned lever, enter the type, name and the unit of the variable into the textbox. You can select variables from a list with the „+“ key, but some of them may require manual corrections (for example if there is an „:index“ in the variable name, that must be replaced with „:1“, „:2“ etc.).

Make sure to adjust Axis Min and Axis Max to the value range that is required by the variable.

3.7.7. Trigger Mode and virtual events

It is also possible to assign one-shot events or even virtual key presses and vJoy events to an axis.

Depending on the "Trigger direction setting", the app will trigger the assigned event or key when you move the axis out of the deadzone in the assigned direction (Both, Up, Down) or send the value continuously (None)

Change Axis Assignment

Select Axis:

OR

Select Event:

OR

Enter Variable: +

OR

Enter virtual event:

Trigger direction: ☐ None/Axis Event ☐ Both ☐ Up only ☐ Down only

Repeat: ☐ Off ☐ Momentary ☐ Progressive

Mode: ☐ Momentary ☐ Progressive

Device: 1ec

Move Joystick axis:

Update value: ☐ Continuously ☐ On change Rounding: ☐ Float ☐ Int

Filter strength

Axis min: Axis max:

Assigned Combo: ☐ None ☐ suppress axis ☐ is toggle

Custom Label: ☐

Assign Virtual Event

Virtual keyboard key:

vJoy Button: Device # Button # OR POV # Direction

vJoy Axis: Device # Axis OR POV #

ViGEm X360 Button:

ViGEm X360 Axis: Axis OR Slider

Mouse action: Movement Scale

Mouse axis: Remember to assign a Combo!

Either enter a keyboard sequence by pressing the desired keys one after the other (=not at the same time) or select a vJoy/ViGEm/mouse action for trigger mode, or axis for continuous mode. Please refer to the chapters about the virtual interfaces at the end of the manual.

If you select a „Repeat“ value, the app will repeat the key press for as long as the axis stays in the configured location. By selecting „Progressive“, the speed with which the event/key is repeated increases, when you move the lever away from the center position.

You can select if you want a virtual key to be triggered and released immediately („Momentary“) or if each press will either activate or release it (Toggle)

3.7.8. Assign combo

A Combo key is a joystick/MIDI button or keyboard combo that must be pressed in addition to the assigned axis. This can be used as a toggle (first click: combo ON, second click: combo OFF) or in continuous operation (the assigned axis will only move as long as you keep the combo button pressed). Click on the green LED to activate the input field.

“Suppress Axis/Event”: when this is activated, the selected Axis is only triggered when the combo key is NOT pressed. That way you can create exclusive toggles.

Example:

1. A joystick X-axis is assigned to control the Aileron, with no further settings
2. The same X-Axis is assigned to control the Rudder, with the joystick button “5” as combo key, “Suppress Axis” is NOT activated

In this configuration, when nothing is pressed, the X axis will move the aileron. When “5” is pressed, aileron AND rudder will move.

3. The Aileron assignment is changed: the same joystick button “5” is assigned as combo, but with “Suppress Axis” *activated*

Now the movement is exclusive: when “5” is pressed, only the rudder moves, when it is released, the aileron moves.

3.7.9. Copy or Remove axis

The screenshot shows the 'Robinson R22' configuration window. It is divided into two main sections: 'Assigned Axis' on the left and 'Assigned Buttons' on the right. The 'Assigned Axis' section contains four entries: 'Aileron', 'Collective', 'Elevator', and 'Rudder'. Each entry has a horizontal slider and a 'Reversed' checkbox. The 'Aileron' and 'Elevator' entries are highlighted in red. The 'Assigned Buttons' section contains two entries: 'LANDINGLIGHTS_TOGGLE' and 'SMOKE_TOGGLE', each with a 'Repeat' checkbox. At the bottom of the 'Assigned Axis' section, there are three buttons: a green '++' button, a green '+' button, and a black '-' button. The '++' and '-' buttons are circled in red. A text box with an arrow pointing to the '++' button says 'Click “++” to duplicate the selected assignments'. A text box with an arrow pointing to the '-' button says 'Click “-” to remove the selected controls'. A text box with an arrow pointing to the 'Aileron' and 'Elevator' entries says 'Click on the controls to select them'. At the bottom of the window, there are two buttons: 'Disable simulator controllers' and 'Manage Joysticks'.

Robinson R22

Assigned Axis

Assigned Buttons

Aileron ☐ Reversed

Collective ☐ Reversed

Elevator ☐ Reversed

Rudder ☒ Reversed

LANDINGLIGHTS_TOGGLE ☐ Repeat

SMOKE_TOGGLE ☐ Repeat

Controller (XBOX 360 For Windows) Button <1>

SHIFT CTRL F (Sim)

Click “++” to duplicate the selected assignments

Click “-” to remove the selected controls

Click on the controls to select them

Disable simulator controllers

Manage Joysticks

3.8 Handling button, MIDI and keyboard assignments

3.8.1. Assign a new button

To assign a new button to your aircraft, click on the green “+” below the Button list on the right. Click on the green LEDs to activate the associated input fields.

Select either

the event that is to be sent when the button is pressed
AND/OR
the event that is to be sent when the button is released

OR

Assign a virtual keyboard sequence to be sent when the button is pressed

Press the desired control until it shows up in the textboxes

You can ignore a button or the entire controller with the exclamation marks (right click to reset)

You can lock to a specific controller with the „lock“ symbol.

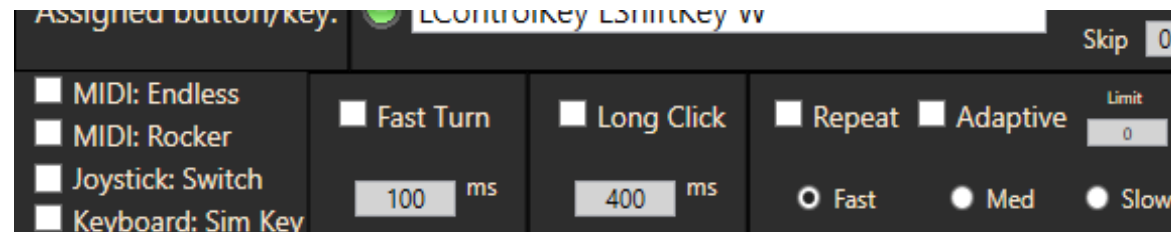


For Keyboard, Mouse or Voice assignments, the „Device“ must be locked to those inputs.

Keyboard combinations must be input in sequence, do not press all keys at the same time.

Example; For “Shift&Ctrl&F” you would press “Shift”, release it, then press “Ctrl”, release that and finally press “F”

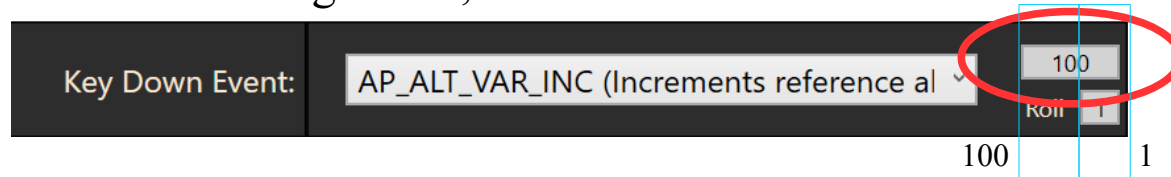
3.8.2. Button actuation parameters



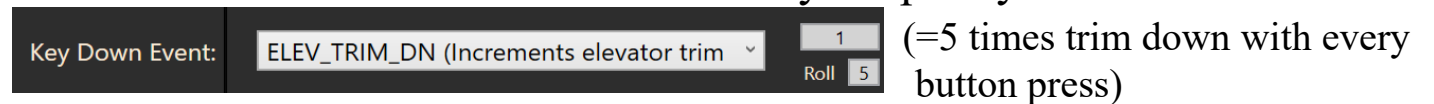
- MIDI: Endless: for MIDI rotary encoders with min/max stops
- MIDI: Rocker: for MIDI rotary encoders that only send three values (left – center – right)
- Joystick: Switch: for switches (=not momentary buttons) on joystick devices, this will synchronize the switch position with the sim when a new aircraft is loaded.
- Keyboard: Sim Key: use this if the simulator „catches“ your keyboard hotkeys
- Fast Turn: This setting is for rotary encoders. When activated, only a fast enough turn will trigger the Key Down Event
- Long click: The Key Down event is triggered when you hold the button down longer than x ms.
- Repeat: Controls if the event is to be repeated while the button is held down and sets the desired speed. „Adaptive“ accelerates the speed, the longer the button is pressed. „Limit“ will stop the repeat after the specified number of actuations. „Skip“ does the opposite, it ignores the number of input events that you specify before firing the down event again

3.8.3. Sending values to events

With some events you can send specific values when the button is pressed. Spin the mouse wheel over the numerical controls left or right half, or doubleclick it to enter a number directly.

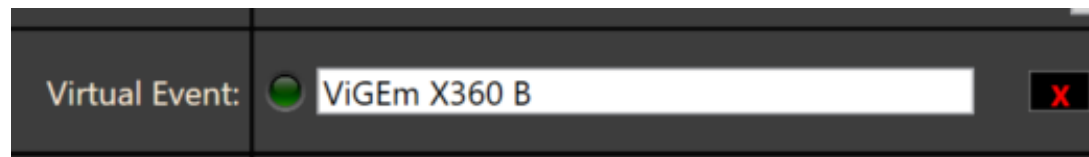


The „Roll“ control will repeat the event for the number of times that you specify in the box:



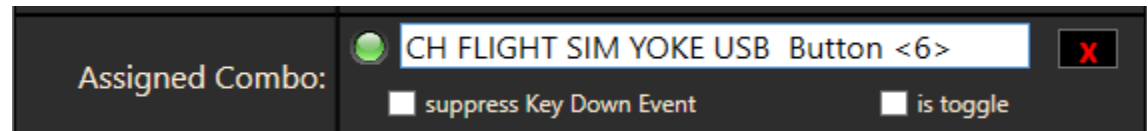
3.8.4. Virtual Events

If you assign a virtual event to the button (keyboard, vJoy, ViGEm, mouse), the app will send those events when the button is pressed, simulating an input of those devices.



To assign a virtual event, activate the green LED or click into the textbox.

3.8.5. Assign combo



Assigned Combo: CH FLIGHT SIM YOKE USB Button <6> X

☐ suppress Key Down Event ☐ is toggle

A Combo key is a joystick/MIDI button or keyboard combo that must be pressed in addition to the assigned key. This can be used as a toggle (first click: combo ON, second click: combo OFF) or in continuous operation (down event is only triggered as long as you keep the combo button pressed)

“Suppress Key Down Event”: when this is activated, the selected Event is only triggered when the combo key is NOT pressed. That way you can create exclusive toggles.

Example: a rotary MIDI encoder right turn shall be used for decreasing the AP selected altitude. You would create two assignments to that rotary encoder with the same combo key. One assignment sets a value of 100 WITH “suppress” checked, the other sets a value of 1000 with “suppress” NOT checked. In this configuration, when the combo key is not pressed, the rotary decreases the altitude by 100ft, and when the combo key is pressed it changes by 1000 feet.

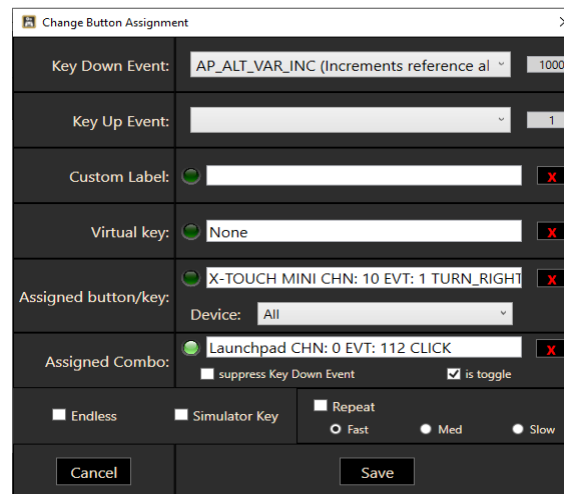


↓ AP_ALT_VAR_INC ☐ Repeat

X-TOUCH MINI CHN: 10 EVT: 1 TURNRIGHT ●

↓ AP_ALT_VAR_INC ☐ Repeat

X-TOUCH MINI CHN: 10 EVT: 1 TURNRIGHT ●



Change Button Assignment

Key Down Event: AP_ALT_VAR_INC (Increments reference al 1000

Key Up Event: 1

Custom Label: ● X

Virtual key: ● None X

Assigned button/key: ● X-TOUCH MINI CHN: 10 EVT: 1 TURN_RIGHT X

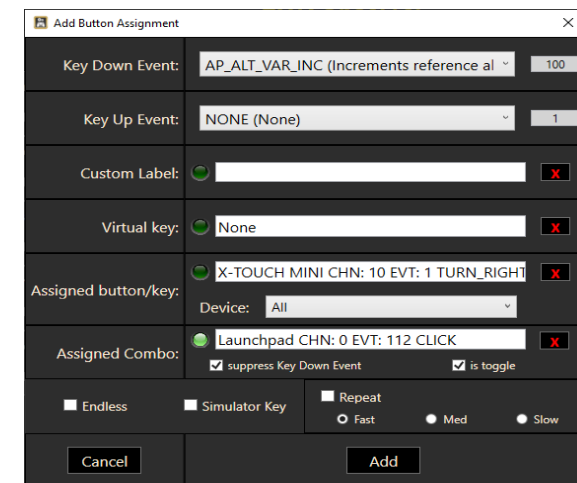
Device: All

Assigned Combo: ● Launchpad CHN: 0 EVT: 112 CLICK X

☐ suppress Key Down Event ☒ is toggle

☐ Endless ☐ Simulator Key ☐ Repeat ☐ Fast ☐ Med ☐ Slow

Cancel Save



Add Button Assignment

Key Down Event: AP_ALT_VAR_INC (Increments reference al 100

Key Up Event: NONE (None) 1

Custom Label: ● X

Virtual key: ● None X

Assigned button/key: ● X-TOUCH MINI CHN: 10 EVT: 1 TURN_RIGHT X

Device: All

Assigned Combo: ● Launchpad CHN: 0 EVT: 112 CLICK X

☒ suppress Key Down Event ☒ is toggle

☐ Endless ☐ Simulator Key ☐ Repeat ☐ Fast ☐ Med ☐ Slow

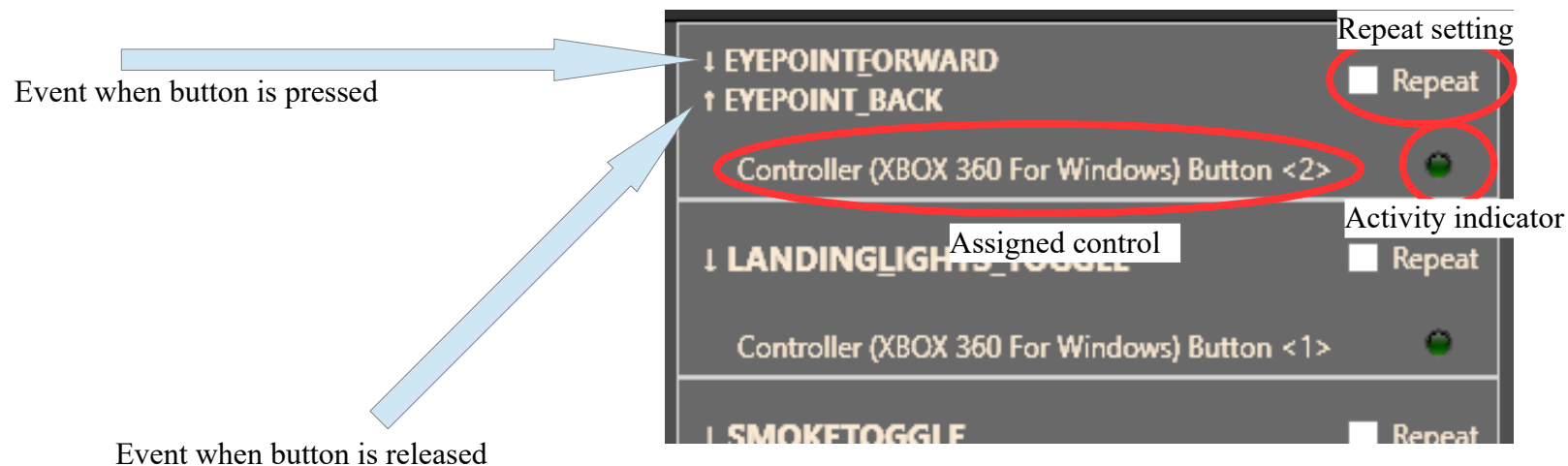
Cancel Add

To assign a combo, activate the green LED by clicking on it. Your joystick, MIDI and keyboard input will then be redirected to the textbox on the right. Deactivate the LED to return to normal operations. Use “X” to remove the assignment and clear the textbox. “**is toggle**” switches the toggle property, so you don't have to hold down the Combo key all the time.

3.8.6. Long click

When „Long click“ is activated, the button must be held down for at least the selected number of milliseconds before the Down Event is triggered. You can assign a „short press“ action to the Up Event too that triggers immediately.

3.8.7. The button assignment control on the main list



3.8.8. Change a button assignment



Change Button Assignment

Key Down Event: AP_ALT_VAR_INC (Increments reference al... 1 Roll 1

Key Up Event: 1 Roll 1

Virtual key: None X

Device: Thrustmaster dual analog 3.2 [ce44d9b0-... ! X

Assigned button/key: Button <1> ! X Skip 0

☐ MIDI: Endless ☐ MIDI: Rocker ☐ Joystick: Switch ☐ Keyboard: Sim Key

☐ Fast Turn 100 ms ☐ Long Click 400 ms ☐ Repeat ☐ Adaptive Limit 0

☐ Fast ☐ Med ☐ Slow

Assigned Combo: None X

☐ suppress Key Down Event ☐ is toggle

Custom Label: X

Cancel Save

The “Change Button Assignment” dialog works the same way as the “Add” dialog (see chapter 3.4.1). To find a specific button in the list, activate „Tools → Find Button“, then actuate it.

Press “Save” to save this assignment or “Cancel” to discard it.

3.8.9. Copy or Remove button assignments

The screenshot shows the 'Robinson R22' control assignment window. It is divided into two main sections: 'Assigned Axis' on the left and 'Assigned Buttons' on the right. The 'Assigned Axis' section contains four sliders for 'Aileron', 'Collective', 'Elevator', and 'Rudder'. Each slider has a 'Reversed' checkbox; 'Rudder' is checked, while the others are unchecked. The 'Assigned Buttons' section contains three red-highlighted entries: 'EYEPOINT_FORWARD' and 'EYEPOINT_BACK' (mapped to Xbox 360 Button <2>), 'LANDINGLIGHTS_TOGGLE' (mapped to Xbox 360 Button <1>), and 'SMOKE_TOGGLE' (mapped to SHIFT CTRL F (Sim)). Each button entry has a 'Repeat' checkbox, which is unchecked for all. At the bottom of the 'Assigned Buttons' section, there are two buttons: a green '++' button and a red '-' button. Red circles highlight these two buttons. A red arrow points from the text 'Click “++” to duplicate the selected assignments' to the '++' button. Another red arrow points from the text 'Click “-” to remove the selected controls' to the '-' button. Two more red arrows point from the text 'Click on the controls to select them' to the first two button entries in the 'Assigned Buttons' section. At the bottom of the window, there are two buttons: 'Disable simulator controllers' and 'Manage Joysticks'.

Robinson R22

Assigned Axis

Aileron ☐ Reversed

Collective ☐ Reversed

Elevator ☐ Reversed

Rudder ☒ Reversed

Assigned Buttons

↓ EYEPOINT_FORWARD
↑ EYEPOINT_BACK ☐ Repeat
Controller (XBOX 360 For Windows) Button <2>

↓ LANDINGLIGHTS_TOGGLE ☐ Repeat
Controller (XBOX 360 For Windows) Button <1>

↓ SMOKE_TOGGLE ☐ Repeat
SHIFT CTRL F (Sim)

Click “++” to duplicate the selected assignments

Click “-” to remove the selected controls

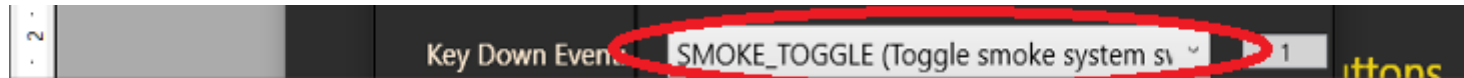
Click on the controls to select them

++ -

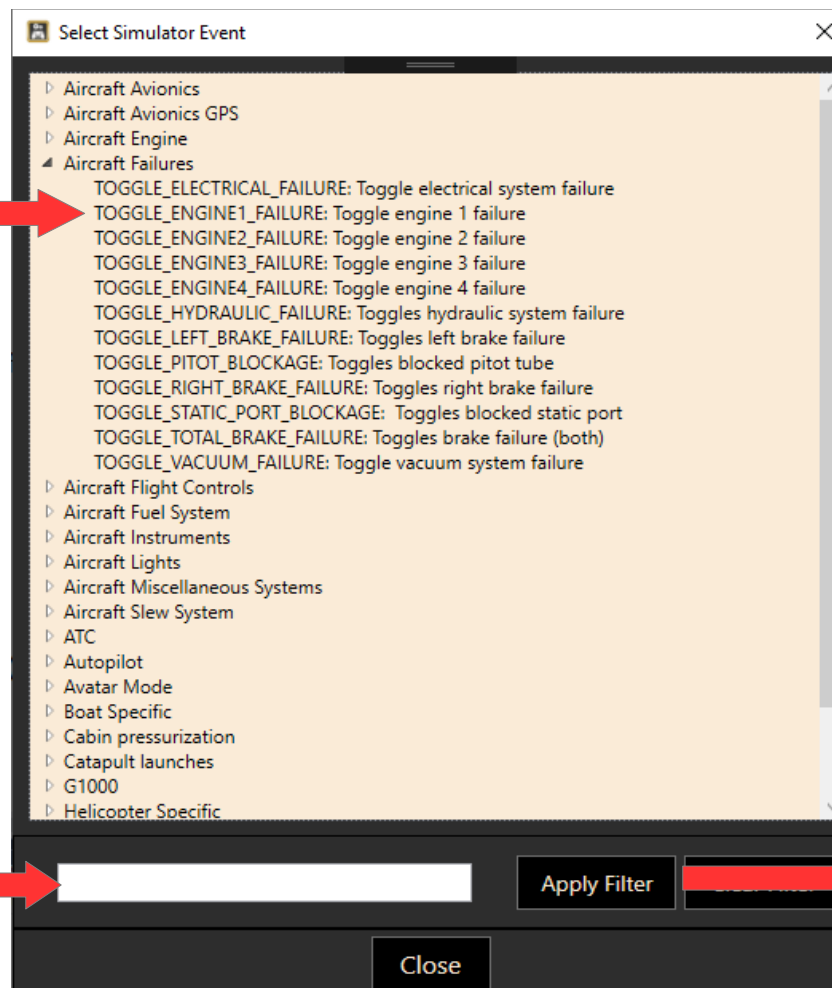
Disable simulator controllers Manage Joysticks

3.9 Using the event selection boxes

A left-click on the combobox opens the selection dialog, a right-click resets it to „None“



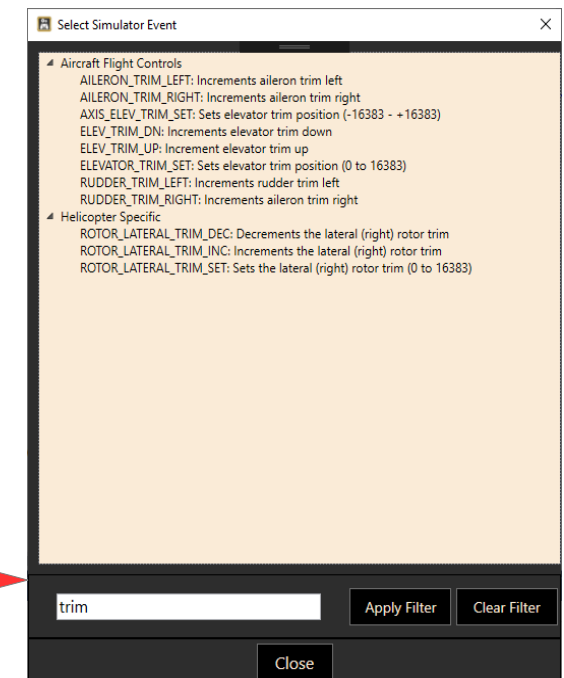
Doubleclick
on an
Event-ID
to select it



Input part of a text to
search in the event
list, then press
“Apply Filter”



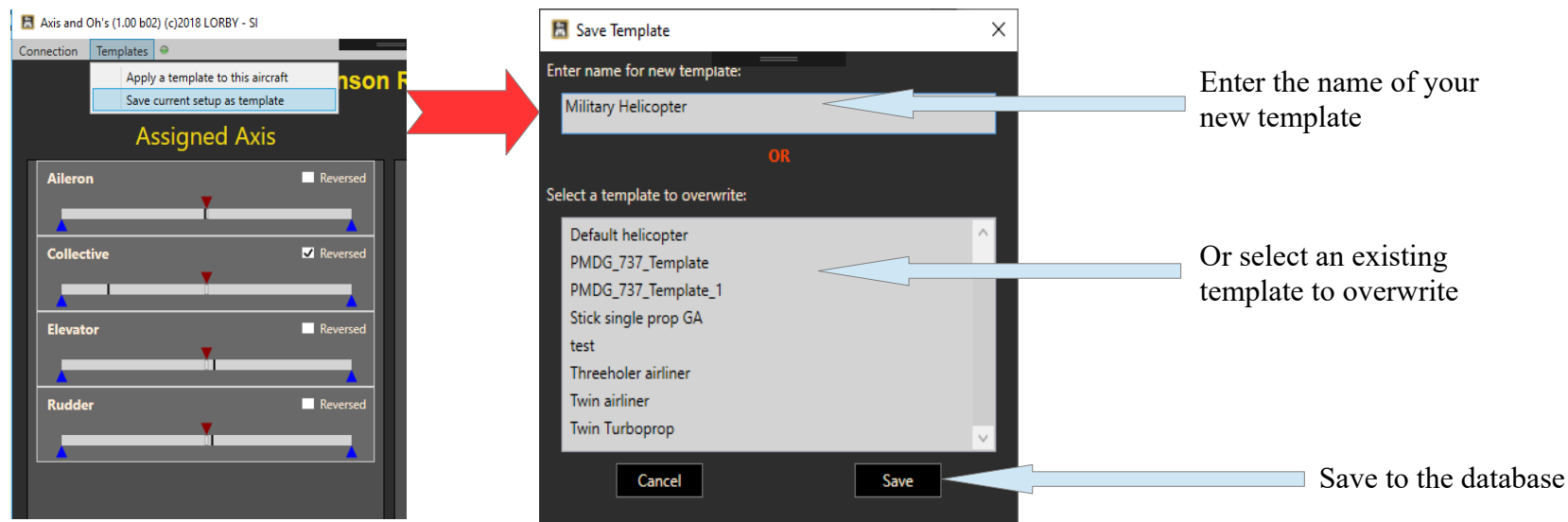
Filter example: searching for “trim”



3.10 Using Templates

You can save an aircraft configuration as a “Template”, so you can **apply** or **link** it to aircraft. Templates can be exported to files and re-imported again. When you import a template, the app will prompt you to reconfigure all controllers in the template. To ignore a controller, just „Cancel“ the assignment dialog for it when it comes up.

Saving a configuration as a Template

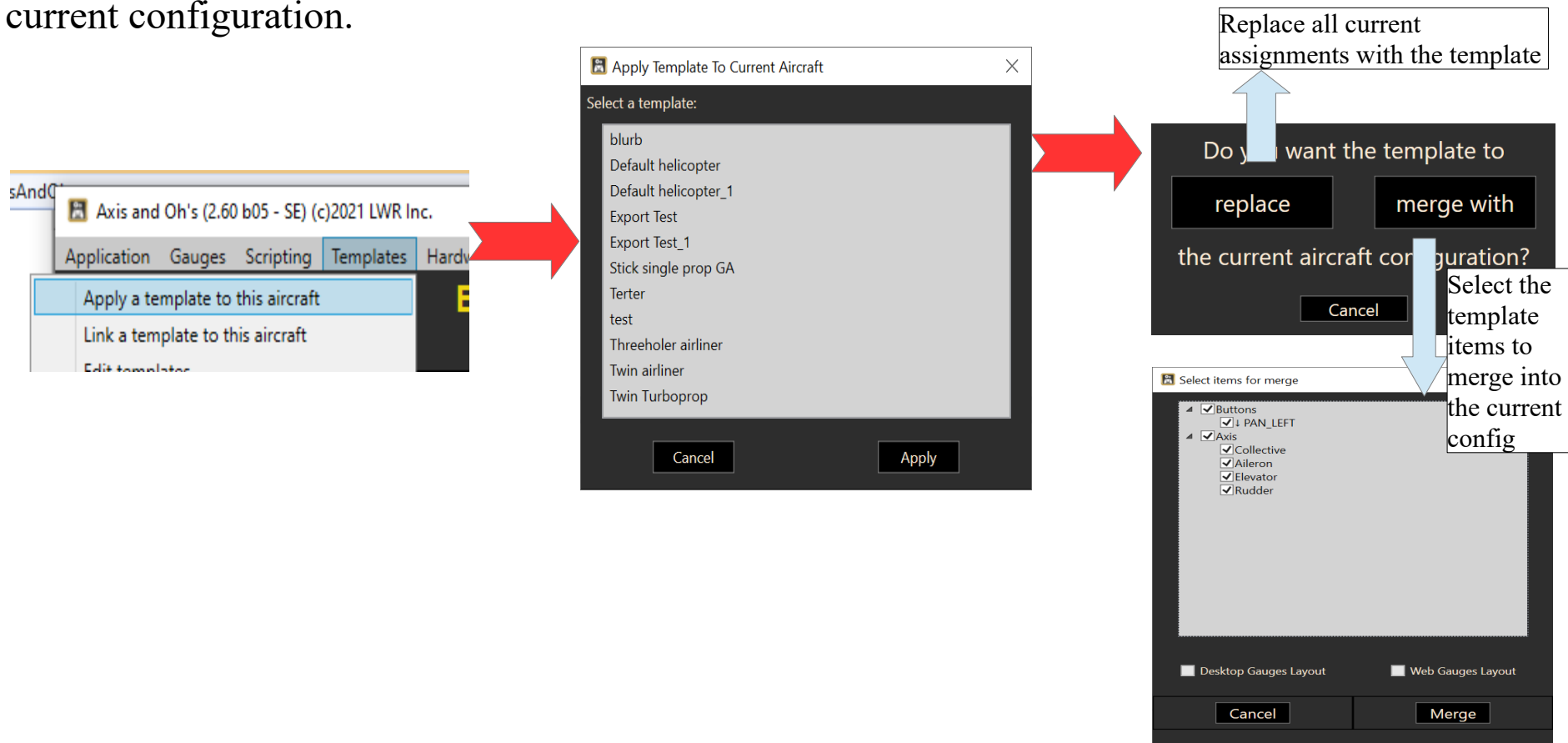


Editing a Template

Put AAO in Offline Mode (red LED) and load the template as you would load a configuration. Templates are at the end of the selection list.

Applying a Template to your current aircraft

When your aircraft has been recognized, use the top menu “Templates → Apply a template to this aircraft”. Applying a Template will create copies of the buttons and axis of the template in the current configuration.

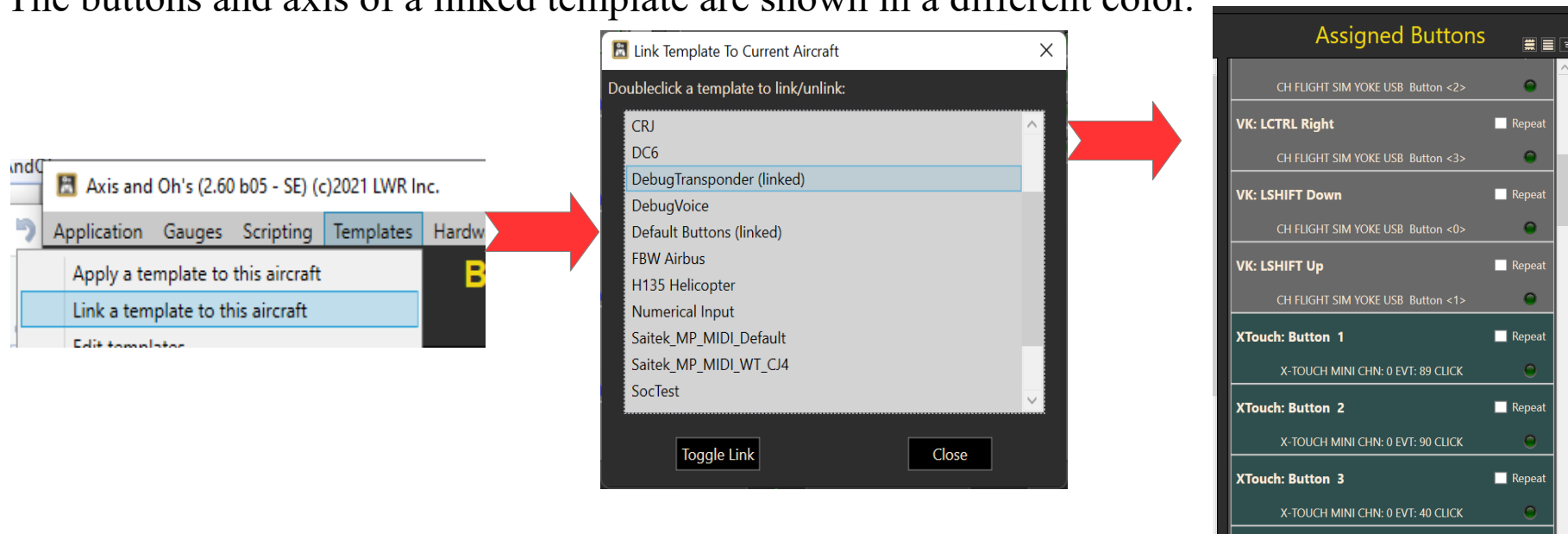


A change in the *applied* template will NOT update the aircraft configurations that were built with it.

Linking a Template to your current aircraft

Instead of *applying* a template, you can *link* them to your current aircraft using „Templates → Link a Template to this aircraft“.

In this case AAO does not create a copy of the buttons and axis in your aircraft configuration, it only references the Template. Changing the template will directly affect all linked aircraft configurations. The buttons and axis of a linked template are shown in a different color.



To link a template doubleclick it in the list. Templates that have already been linked to your aircraft are displayed as „ (linked)“ - doubleclick them when you want to remove the link.

A change in the *linked* template WILL affect all associated aircraft configurations.

Editing templates and configurations

You can edit existing templates and configs with „Tempates->Edit templates / Edit configs“

The image shows three screenshots of the software interface for editing templates and configurations, with various annotations and arrows explaining the functions.

Left Window: Edit Template

- Select a template:** A list of templates including "Advanced helicopter", "Default helicopter", "Export Test" (highlighted), "Stick single prop GA", "Threeholer airliner", "Twin airliner", and "Twin Turboprop".
- Buttons:** "Edit selected", "Delete selected", "Rename selected to", "Export Test" (text input), and "Close".
- Annotations:**
 - A green arrow points from the "Export Test" template to the right window.
 - A blue arrow points to the "Rename selected to" button with the text: "Rename the template to the contents of the textbox."
 - A red arrow points to the "Delete selected" button with the text: "Delete the selected template".

Middle Window: Edit template <XTouch>

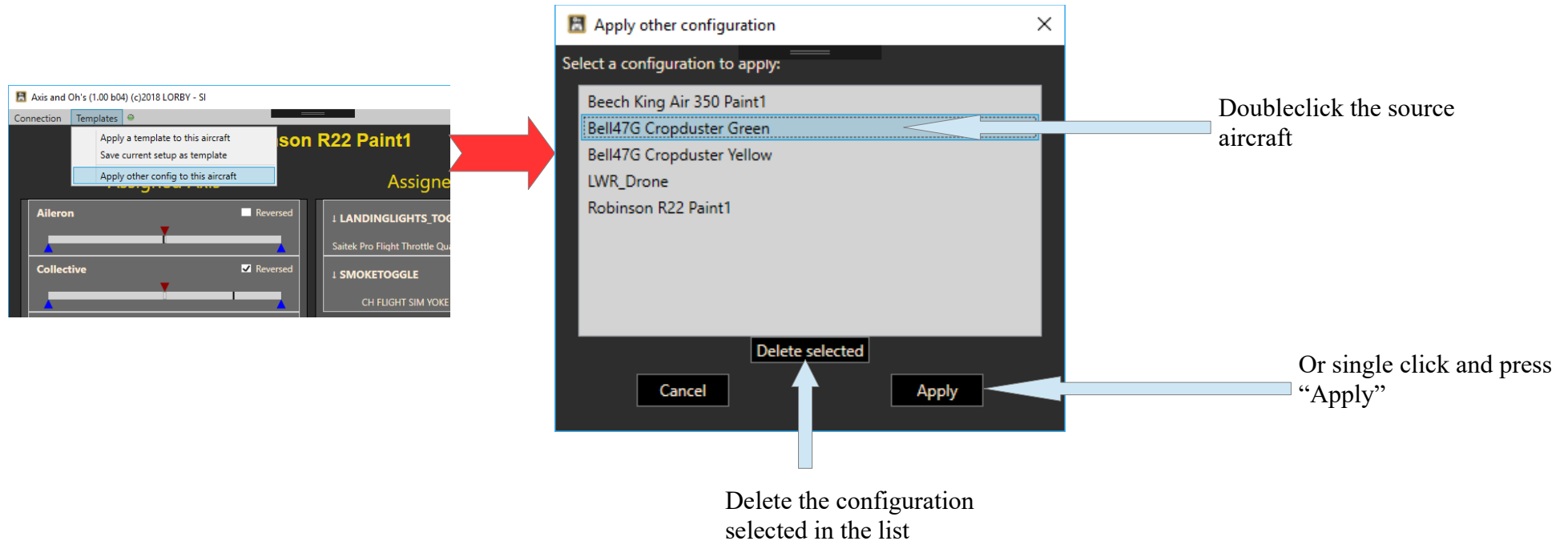
- Buttons:** "-", "+", "Apply template", "Change controller", "Cancel", and "Apply".
- Annotations:**
 - A red arrow points to the "-" button with the text: "Delete the selected items".
 - A blue arrow points to the "Apply template" button with the text: "Merge another template into this".
 - A blue arrow points to the "Change controller" button with the text: "Change the controller assignment for the selected items".

Right Window: Doubleclick item to add it to the template

- Buttons:** "Close".
- Annotations:**
 - A green arrow points from the middle window to this window with the text: "Add an item from the current aircraft profile to the template".
 - A green arrow points from this window back to the middle window with the text: "Doubleclick to transfer an item to the template".

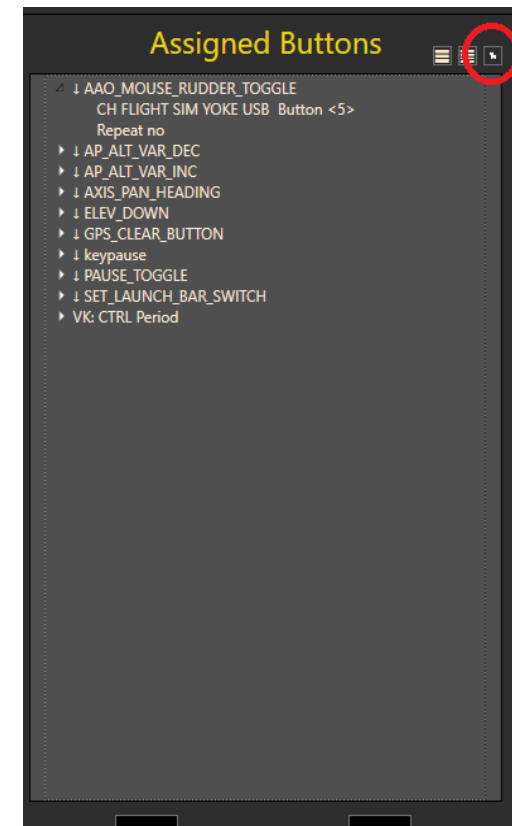
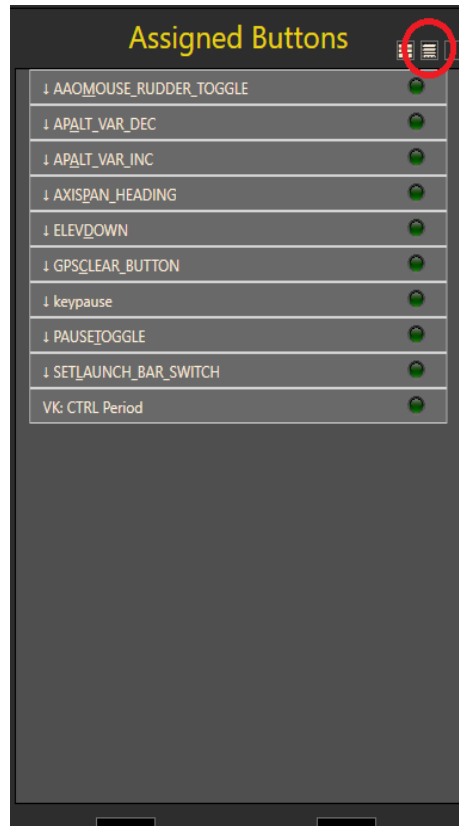
Applying an existing configuration to your current aircraft

When your aircraft has been recognized, use the top menu “Templates → Apply other config to this aircraft”



3.11 Panel view modes

The axis and button panels can be switched to three different view modes with the icons at the top



The treeview has a special use case: when you assign a Custom Label to a button, you can use that to group the buttons in the tree. For this, the label has to be written as „groupname:labelname“.

4. Voice Recognition

AAO can be instructed to listen to the standard voice recognition feature that is built into your Windows system. By adding phrases on the button assignment dialog, you can trigger events and scripts by speaking the phrase into the microphone of your default audio device. A „push-to-talk“ option is available, should you wish enable voice recognition manually only when needed.

Please note:

- The voice recognition is only as good as your local Windows system is. You need to train it properly, so it picks up your phrases reliably.
- Windows voice recognition is localized, if you want to use different languages, you will have to add appropriate Microsoft language packs to your Windows system.
- You can select a specific language for voice recognition in the menu „**Extras->Select language for speech recognizer**“. Doubleclick on a language in the list to select it. Use the red „X“ to reset the selection to your Windows default. You may need this if your default speech recognition is different from your default system language. (Note: this dialog will disconnect AAO from the simulator)

Assigning a voice phrase to an event

To assign a voice phrase, you use the „Add/Change“ Button dialogs:

- First, select „Voice“ from the „Device“ control
- Then type your desired phrase into the „Assigned Button/Key“ textbox
- You can add multiple phrases to the same assignment using the | „pipe“ symbol: „all lights|full lights|toggle lights“
- You can tell AAO to wait until you release the PTT button before it fires the event by adding „!“ at the end of the phrase. This is required for capturing numerical input.

The screenshot shows the 'Change Button Assignment' dialog box with the following fields and annotations:

- Key Down Event:** ALL_LIGHTS_TOGGLE (Toggle all lights) [Roll 1]
- Key Up Event:** [Roll 1]
- Custom Label:** [Text input field] [X]
- Device:** Voice [Dropdown menu] [!]
- Virtual key:** None [Text input field] [X]
- Assigned Button/key:** all lights [Text input field] [!]
- Assigned Combo:** None [Text input field] [X]
- Buttons:** ☐ Endless, ☐ Fast Turn, ☐ Long Click, ☐ Repeat, ☐ Adaptive, ☐ Sim Key, 100 ms, 400 ms, Fast, Med, Slow
- Buttons:** Cancel, Save

Annotations in the image include:

- A red box around the 'Device' dropdown menu with an arrow pointing to it and the text 'Select "Voice" here'.
- A red box around the 'Assigned Button/key' text input field with an arrow pointing to it and the text 'Type in a phrase using your keyboard'.

SRGS grammars

Instead of phrases separated by the | pipe symbol, you can also enter the name of an SRGS style XML file. Either enter an absolute path (C:\myfolder\...\agrammar.xml) or a file path relative to "\Documents\LorbyAxisAndOhs Files\Grammars\" (so just "agrammar.xml" if that file is in the folder \Grammars\)

SRGS specification:

[Speech Recognition Grammar Specification Version 1.0 \(w3.org\)](http://www.w3.org/2001/06/grammar)

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar version="1.0" xml:lang="en-US"
  xmlns="http://www.w3.org/2001/06/grammar" tag-format="semantics/1.0" root="Main">
<!-- Defines an SRGS grammar for requesting a flight. This grammar includes a Cities rule that lists
the cities that can be used for departures and destinations. -->
  <rule id="Main">
    <item> I would like to fly from <ruleref uri="#Cities"/> to <ruleref uri="#Cities"/>
    </item>
  </rule>
  <rule id="Cities" scope="public">
    <one-of>
      <item> Seattle </item>
      <item> Los Angeles </item>
      <item> New York </item>
      <item> Miami </item>
    </one-of>
  </rule>
</grammar>
```

Numerical input

You can set up phrases where you can enter numbers. All digits have to be spoken separately, followed by a multiplier of 100 or 1000. „back“ deletes the last digit.

The grammar for numerical input can be adjusted with the dialog

„**Extras->Numerical pattern recognition**“

You can enter the recognized phrases on this dialog. Use the pipe symbol „|“ to add multiple options of the same phrase.

Example: setting the AP altitude

1. Make sure to assign a PTT button (see next chapter)

2. Create a script:

Script group: Scripts

Script name: VoiceApAlt

Script code: (L:VoiceRecNum) (>K:AP_ALT_VAR_SET_ENGLISH)

3. Create a voice assignment

Key down event: „Scripts-VoiceApAlt“

Phrase: „set altitude!“

the ! is mandatory, it tells AAO to wait until the PTT has been released so you can speak the numbers

Action: press PTT, speak „set altitude two one thousand“, release PTT.

Speak slowly – leave a small pause between the text and the numbers.

0:	zero null
1:	one eins
2:	two zwei
3:	three drei
4:	four vier
5:	five fünf
6:	six sechs
7:	seven sieben
8:	eight acht
9:	nine neun
Decimal:	decimal komma
100:	hundred hundert
1000:	thousand tausend
back:	back zurück

Close

Utilizing the numerical input

There are two options to make use of the numerical input:

1. Simple integer values, like altitudes or degrees, you can use directly in the assignment with Events that process input values.

Example: Setting the AP altitude

When numerical input is received, AAO will ignore the KeyDownEvent value box and use the voice input instead.

In this example you would say „set altitude two two thousand“ to select an AP altitude of 22000 feet.

2. Numbers with decimal fractions, like frequencies, or more complex interactions, you can handle with RPN scripts. The recognized numerical value is available as a locald Lvar (L:AoVoiceRecNum)

Example: setting the COM 1 frequency

The image shows two screenshots from a software interface. The left screenshot is the 'Change Button Assignment' dialog box. It has several fields: 'Key Down Event' is set to 'Scripts-Setcom1 (RPN)', 'Key Up Event' is empty, 'Virtual key' is 'None', 'Device' is 'Voice', and 'Assigned button/key' is 'set com one to!'. There are also checkboxes for 'MIDI: Endless', 'MIDI: Rocker', 'Joystick: Switch', and 'Keyboard: Sim Key', as well as 'Fast Turn', 'Long Click', 'Repeat', and 'Adaptive' options. The 'Assigned Combo' field is 'None', and the 'Custom Label' field is empty. The 'Cancel' and 'Save' buttons are at the bottom. The right screenshot is the 'Setcom1' script editor. It shows the 'Script title' as 'Setcom1' and the 'Script group' as 'Scripts'. The script content is: `(L:AoVoiceRecNum) · 100 · * · dec2b16 · (>K:COM_RADIO_SET) · 1 · (>K:COM_STBY_RADIO_SWAP)`. The editor has a toolbar with icons for Cut, Copy, Paste, Undo, Redo, Bold, Italic, Underline, and a font dropdown set to 'Courier New'.

Voice input would be „set com one to one two four decimal two five“ to set a frequency of 124.25

NATO alphabet

Along the same lines as the numerical input you can input strings using the NATO alphabet.

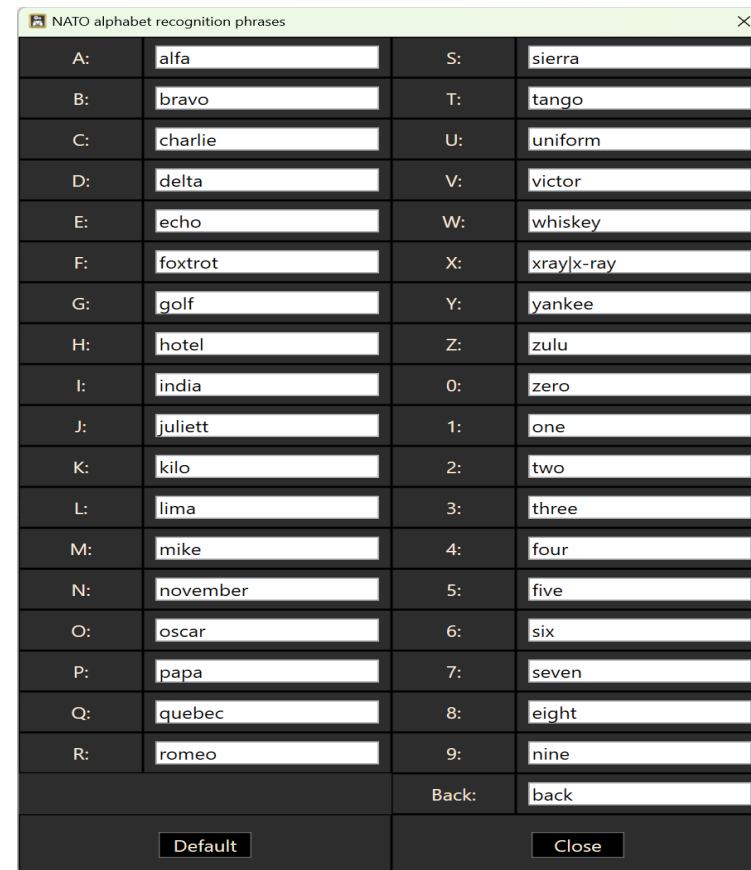
„Extras->NATO alphabet recognition“

You can enter the recognized phrases on this dialog. Use the pipe symbol „|“ to add multiple options of the same phrase.

To utilize the NATO alphabet input, add a "#" to your recognition string.

You cannot use the ! and # flag at the same time.

The recognized letters will be stored in "(L:AaoVoiceRecNato, String)".



NATO alphabet recognition phrases			
A:	alfa	S:	sierra
B:	bravo	T:	tango
C:	charlie	U:	uniform
D:	delta	V:	victor
E:	echo	W:	whiskey
F:	foxtrot	X:	xray/x-ray
G:	golf	Y:	yankee
H:	hotel	Z:	zulu
I:	india	0:	zero
J:	julieta	1:	one
K:	kilo	2:	two
L:	lima	3:	three
M:	mike	4:	four
N:	november	5:	five
O:	oscar	6:	six
P:	papa	7:	seven
Q:	quebec	8:	eight
R:	romeo	9:	nine
		Back:	back
Default		Close	

Adding the PTT button

If you don't want AAO to listen all the time, you can implement a PTT button like so:

- The required events are in the Event treelist in the group „Voice recognition“
- Select „AAO_VOICEREC_ON“ as Down Event and „AAO_VOICEREC_OFF“ as Up – Event

OR

- Select „AAO_VOICEREC_TOGGLE“ as there Down-Event

Change Button Assignment

Key Down Event: AAO_VOICEREC_ON (PTT - voice recognit) 1 Roll 1

Key Up Event: AAO_VOICEREC_OFF (PTT - voice recognit) 1 Roll 1

Custom Label: [Green circle] [Red X]

Device: PFC Cirrus Yoke [1e6c10b0-8f01-11eb-80...] [Warning icon]

Virtual key: [Green circle] None [Red X]

Assigned button/key: [Green circle] Button <6> [Button icon]

Assigned Combo: [Green circle] None [Red X]

☐ suppress Key Down Event ☐ is toggle

☐ Endless ☐ Fast Turn ☐ Long Click ☐ Repeat ☐ Adaptive

☐ Sim Key 100 ms 400 ms ☐ Fast ☐ Med ☐ Slow

Cancel Save

All „AAO_VOICEREC_xx“ Events can be assigned to voice commands too, they themselves are independent from the PTT logic.

LVars used by the voice recognition module

(L:AoVoiceRecOn)

value of 0/1, indicates if the voice recognition is currently active (=PTT pressed)

(L:AoVoiceRecPhrase, String)

string variable containing the currently recognized phrase. This can be useful for example when viewed on a FIP, so you can correct numerical input with the „back“ command.

(L:AoVoiceRecNum)

Floating point number, containing the final numerical input that has been recognized.

(L:AoVoiceRecNato, String)

String containing the letters that have been recognized.

All these variables are AAO internal, you cannot access them in the simulator.

Voice Recognition vs. Dictation

In AAO you can also use a "dictation mode". In this mode, the voice recognizer will just pass a string representation of your entire voice input in the variable "**(L:AaoVoiceRecPhrase, String)**". You can then parse that string in a script or you can direct AAO to try and parse it into button events same as the normal voice recognition described above.

To utilize dictation use these special AAO commands:

AAO_DICTATION_START: This command will start dictation mode using the currently selected voice recognition engine and locale.

AAO_DICTATION_STOP: This command stops the dictation mode, but leaves the dictation engine active.

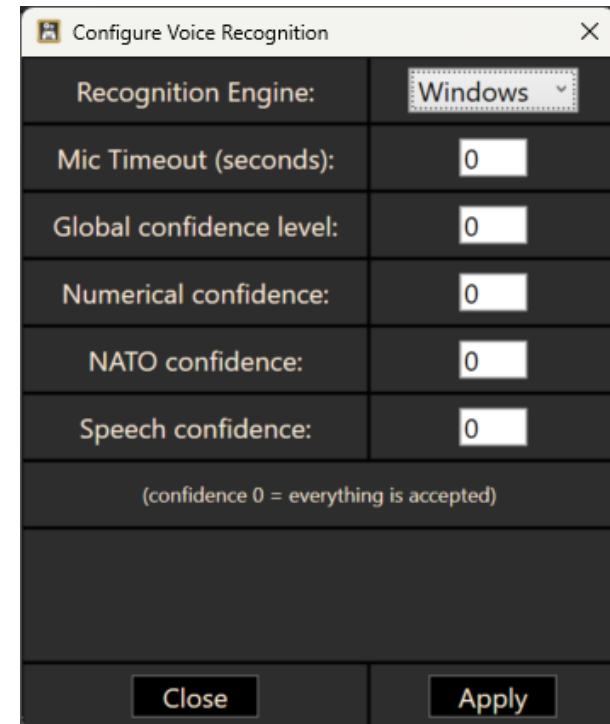
AAO_DICTATION_RESET: This will shut down the dictation engine and create a new one next time you use start. Use this when you are changing the language.

AAO_DICTATION_PARSE: If you've designed voice button events or are using listener scripts, this command will send the dictated input to the voice recognition module for processing. It will trigger all button events that it can find in your string, same as the normal voice recognition.

Voice Recognition Configuration

On the dialog "Extras->Configure speech recognizer" you can configure the voice recognition.

- You can choose between "Windows", "Azure" and "Cortana" recognition.
Be mindful that Azure requires a billable account, set up as described in the chapter about "Advanced text to speech"
The "Cortana" option requires the separate AaoCortanaBridge app.
- Mic Timeout: this closes the microphone automatically after the specified time (0 = no timeout)
- Confidence Levels (Windows engine only): this determines the accuracy of the speech recognition for the different grammars



The screenshot shows a dialog box titled "Configure Voice Recognition" with a close button (X) in the top right corner. The dialog contains several configuration options:

Recognition Engine:	Windows
Mic Timeout (seconds):	0
Global confidence level:	0
Numerical confidence:	0
NATO confidence:	0
Speech confidence:	0

Below these settings, a note states: "(confidence 0 = everything is accepted)". At the bottom of the dialog, there are two buttons: "Close" and "Apply".

Voice Recognition Desktop/Web FIP



There is a default Gauge „Lorby Voice Recognition Status“ that demonstrates how these LVars can be used. It shows if AAO is currently listening, and the recognized phrase. With this gauge you can correct the numerical input using the „back“ command.

5. Scripting

To emulate complex functionality on buttons you can create scripts. AxisAndOhs uses a simplified version of the legacy FSX RPN gauge notation, but you can use other languages too.

Test

Test: send script to processing

Green: AAO can process the script

Red: Hover mouse over the LED to see error details

Editor area

Ctrl&MouseWheel to zoom

Compiled script code

Doubleclick to Edit

Script title: ACT_HEADING_TO_AP

Script group: Scripts

Repeating

(A:PLANE·HEADING·DEGREES·GYRO,·Degrees)·(>K:HEADING_BUG_SET)

Insert sim var

Insert sim event

Insert AAO cmd

Insert RPN op

Insert script var

Insert BVar map

Insert VKey

Insert VMouse

Insert script call

New JScript

New VBScript

regular expression

Apply Filter

Clear Filter

Close

Delete script

Save as New

Update

- The “Script title” is the label that identifies this particular script. Titles must be unique for every script that you create.
- „Script group“: you can change the group that the script is listed in on the event assignment dialog. Use the „+“ key to add your own group names.
- A "Repeating" script will repeatedly execute the code until it is called again.
- You can access lists of variables and events to insert them into the script.
- You can insert line comments into the script by preceding them with „/*“ and ending them with a carriage return
- The little dots between the commands are only a visual cue in RPN Code – they show the position where there is a “space” character.

Naming convention

Avoid using the "-" and ":" characters in script, group and L-variable names!

The grey text box at the bottom shows the compiled script code. With RPN it is a single line of text, there are no carriage returns in it.

The only specification that is still available online about RPN scripting is that from P3D:

https://www.prepar3d.com/SDKv5/sdk/scripting/rpn_scripting.html

Be mindful that scripts should be as simple as possible, and that AAO doesn't support all operators, only the most common ones.

Assigning the script to a button

Add Button Assignment

Key Down Event:	ACT_HEADING_TO_AP (ACT_HEADING_TO_	
Key Up Event:	NONE (None)	1
Custom Label:		Clear
Virtual key:	None	<input type="checkbox"/> Assign Clear
Assigned button/key:	SHIFT CTRL W	
<input type="checkbox"/> Assign combo:	None	
	<input type="checkbox"/> suppress Key Down Event	<input type="checkbox"/> is toggle
<input type="checkbox"/> Simulator Key <input type="checkbox"/> Repeat <input type="radio"/> Fast <input checked="" type="radio"/> Med <input type="radio"/> Slow		
Cancel	Add	



Select Simulator Event

- ▷ Aircraft Avionics
- ▷ Aircraft Avionics GPS
- ▷ Aircraft Engine
- ▷ Aircraft Failures
- ▷ Aircraft Flight Controls
- ▷ Aircraft Fuel System
- ▷ Aircraft Instruments
- ▷ Aircraft Lights
- ▷ Aircraft Miscellaneous Systems
- ▷ Aircraft Slew System
- ▷ ATC
- ▷ Autopilot
- ▷ Avatar Mode
- ▷ Boat Specific
- ▷ Cabin pressurization
- ▷ Catapult launches
- ▷ G1000
- ▷ Helicopter Specific
- ▷ Miscellaneous Events
- ▷ Mouse Flight Controls
- ▷ Multiplayer
- ▷ Nose wheel steering
- Scripts**
- ACT_HEADING_TO_AP: ACT_HEADING_TO_AP
- ACT_HEADING_TO_OBS1: ACT_HEADING_TO_OBS1
- ACT_HEADING_TO_OBS2: ACT_HEADING_TO_OBS2
- ▷ View System

Doubleclick a script label

Apply Filter Clear Filter

Close

Pressing “Shift&Ctrl&W” will now set the current heading as the Autopilot heading.

Calling scripts as events

Scripts can be called like events, using „(>K:scriptgroup-scriptname)“.

Calling a script with parameters

You can pass up to 99 parameters to scripts when calling them as events like this:

„(>K:*scriptgroup-scriptname*;*param1*;*param2*)“

Inside the script that is being called, the strings „param1“, „param2“ etc. are replaced with what you specified in the call. Be mindful that this is a simple text replacement.

Example: handling an LVar that can have three values 0, 1 and 2. This simulates a three-state switch cycling 0-1-2-1-0-1-2... with every click of a button

- The script is in the group „Pattern“ and has been called „Lvar_0_1_2_1_0“
- The code of the script is using two parameters, param1 is the LVar being handled, param2 is a local LVar storing the direction in which the switch is moving (0 = up, 1 = down)

```
(param1) ·s0 ·  
(param2) ·s1 ·  
10 ·0 ·== ·if{ ·l1 ·0 ·== ·if{ ·1 ·s2 ·} ·els{ ·1 ·s2 ·} ·0 ·s1 ·} ·  
10 ·1 ·== ·if{ ·l1 ·0 ·== ·if{ ·2 ·s2 ·} ·els{ ·0 ·s2 ·} ·} ·  
10 ·2 ·== ·if{ ·l1 ·0 ·== ·if{ ·1 ·s2 ·} ·els{ ·1 ·s2 ·} ·1 ·s1 ·} ·  
l2 ·(>param1) ·  
l1 ·(>param2) ·
```

When this script is called using

```
1·(>K:Pattern-Lvar_0_1_2_1_0;L:SWS_FUEL_Switch_Pump_1, Enum;L:SwitchFPDir)
```

the resulting code that is being executed ist this:

```
(L:SWS_FUEL_Switch_Pump_1, Enum)·s0·  
(L:SwitchFPDir)·s1·  
10·0·==·if{·l1·0·==·if{·1·s2·}·els{·1·s2·}·0·s1·}·  
10·1·==·if{·l1·0·==·if{·2·s2·}·els{·0·s2·}·}·  
10·2·==·if{·l1·0·==·if{·1·s2·}·els{·1·s2·}·1·s1·}·  
12·(>L:SWS_FUEL_Switch_Pump_1, Enum)·  
11·(>L:SwitchFPDir)·
```

Processing the button or axis value

When assigned to a Button or Axis, the script can receive the value that is transmitted from the control using the script variable:

```
(L:Groupname-Scriptname)
```

That way you can use the same script for several different assignments. For example, instead of two scripts required to increment the AP altitude by 100 or 1000 you can just use the one, and set 100 and 1000 as Key Down Event value:

```
(L:Scripts-MyAltIncScript) (>K:AP_ALT_VAR_INC)
```

Repeating a >K: event

By adding the pipe symbol „|“ you can repeat an Event or Script multiple (n) times:

```
x (>K:EventOrScriptName|n)
```

Reading the result of a script

Scripts that return a value (numerical or string) can be accessed from other scripts like this:

```
(S:Groupname-Scriptname)
```

```
(S:Groupname-Scriptname, String)
```

Example:

The calculation script is in group „testgroup“, it is called „add“ and has the code „8 5 +“.

Then another script can call the result of this calculation:

```
(S:testgroup-add) 10 + (>L:mytestresult)
```

Running this script will write „23“ into the Lvar.

Local variables (LVARs):

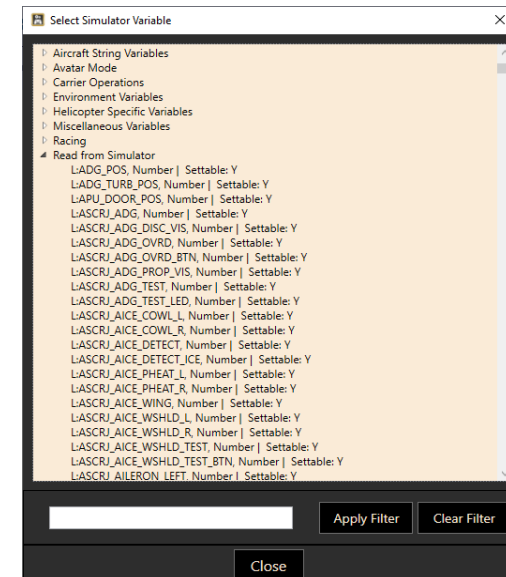
With AAO scripts you can read and write local variables too

- If the variable should only be local to AAO, don't supply a unit: (L:AaoLocalVar)
- String variables are also local to AAO: (L:AaoStringVar, String)
- If the variable is instead an LVAR from the simulator, the unit must be supplied (except String!): (L:SimLocalVar, Number)

Getting a list of all LVARs from the simulator

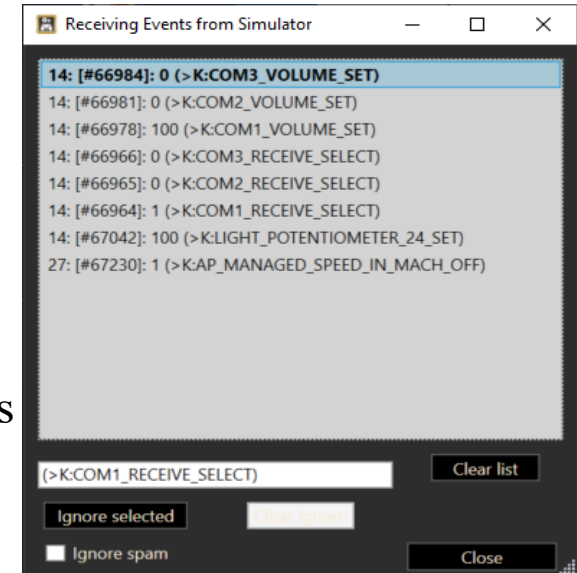
Use the menu option „Scripting → Read LVARs from sim“ to import a list of all LVARs that are currently active in the simulator. The result will be available on the „RPN Scripts Editor“ dialog using the button „Insert variable“: all acquired LVARs will be collected in the group „Local simulator variables“.

LVARs are only shown in this list when they have been used at least once in the simulator.



Tracking the default simulator events

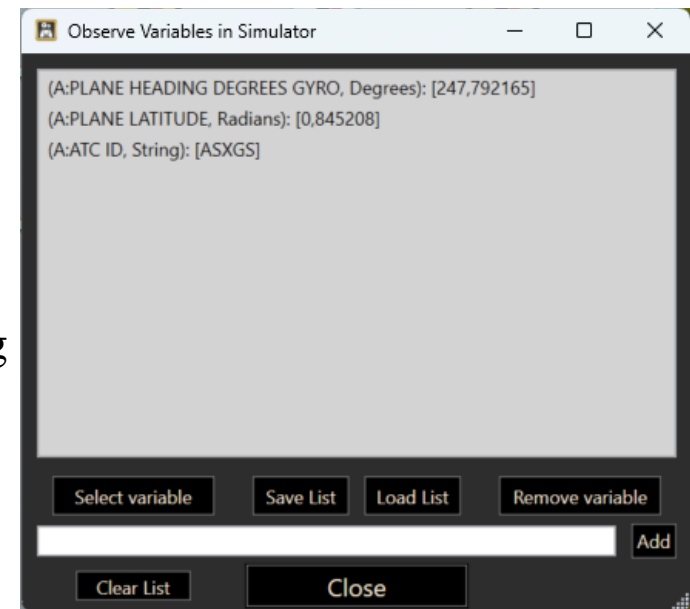
Use the menu option „Scripting → Watch simulator events“ to open the event watcher. This dialog will show simulator events when they are triggered – for example, when you click on a button in the VC. Be mindful though, that not all buttons are tied to simulator events. The more complex an aircraft is, the higher is the probability that the developer chose other means of button actuation. In many cases the aircraft logic itself will send events, sometimes a lot of them. Use the „Ignore spam“ checkbox to ignore all events automatically that have been received more than 40 times.



Tracking variables in the simulator

Use the menu option „Scripting → Watch simulator variables“ to open the variables watcher.

1. You can add simulator variables from the usual treelist using „Select variable“ or type their name into the textbox
2. Then press „Add“ to add the variable to the list
3. The list can be saved to/loaded from a text file



Tracking script execution

Use the menu option „Scripting → Watch AAO script processing“ to open the script handler console.

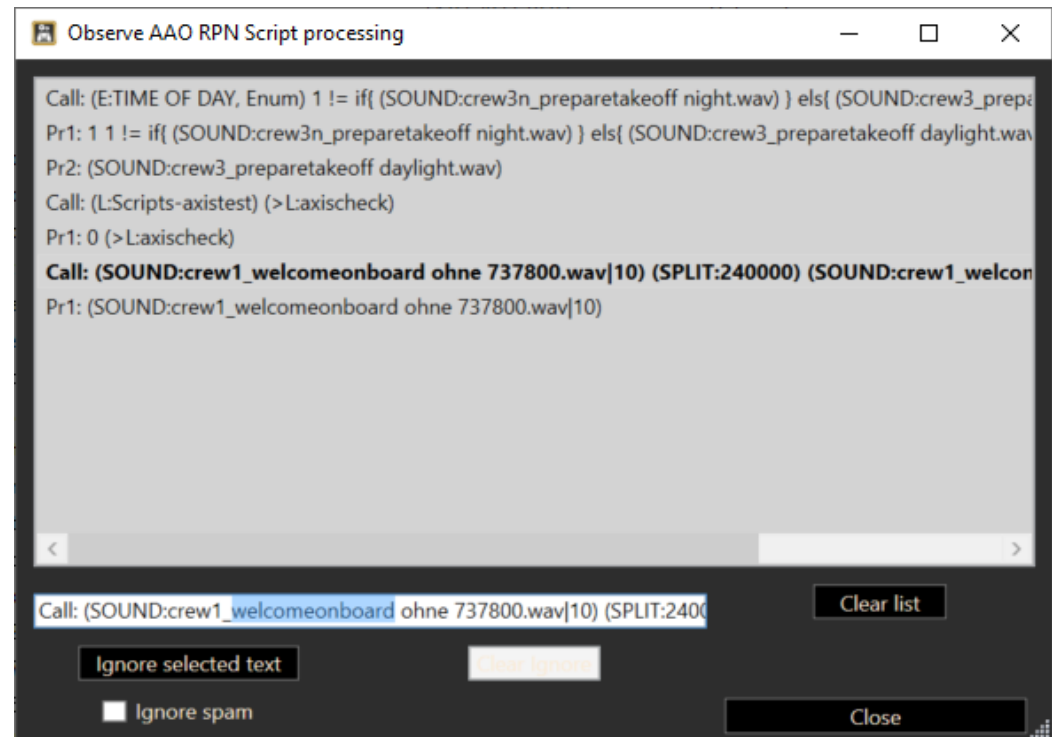
- Call: is the script code going into the script handler

- Pr1/Pr2 are processing steps

- „Ignore spam“ will automatically ignore a script when the same code is detected more than 20 times

- Click on a script to put it into the ignore box“

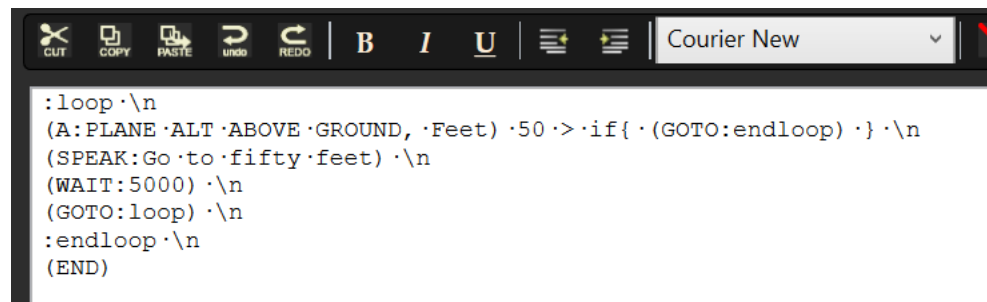
- With „Ignore selected text“ you can direct AAO to ignore all scripts that contain the text that is in the box or just the part that you highlight in the box with the mouse



Scripts with multiple lines:

Normally an RPN script is only a single line of code. In AAO you can create scripts with multiple lines, so you can use commands like GOTO. Multiline scripts are converted into CONVERSATION type objects internally (see chapter about „RPN Script Files“).

To break a script into several lines, add „\n“ at the end of each line in the RPN editor:

A screenshot of the RPN editor interface. The top toolbar includes icons for CUT, COPY, PASTE, UNDO, and REDO, followed by text formatting options (B, I, U), list creation icons, and a font dropdown menu set to 'Courier New'. The main text area contains a script with the following lines:

```
:loop·\n(A:PLANE·ALT·ABOVE·GROUND,·Feet)·50·>·if{·(GOTO:endloop)·}·\n(SPEAK:Go·to·fifty·feet)·\n(WAIT:5000)·\n(GOTO:loop)·\n:endloop·\n(END)
```

Exporting and importing scripts:

This feature is meant for exchanging scripts with other users

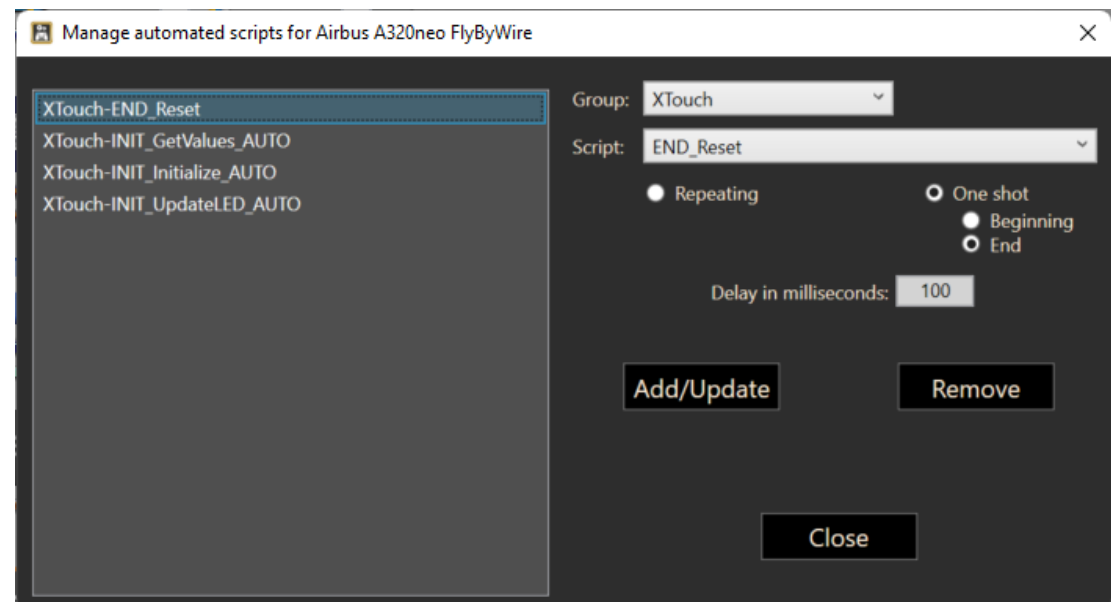
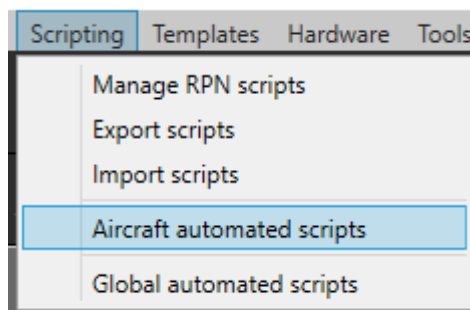
- With the dialog “Scripting->Export scripts” you can write your scripts to a simple XML file.
- With “Scripting->Import scripts” you can read the scripts from an XML file into your local AAO database. Script import is only possible when AAO is not connected to the simulator (= green LED is dark).

Automated scripts:

You can assign scripts to run automatically, without the necessity to press a button or controller. This assignment can be done on a global level, so the script runs all the time, regardless of the current aircraft. Or you can assign automated scripts to run only with a specific aircraft.

- „Repeating“ scripts will run every time the „Delay“ has expired.
 - A „One shot“ script is running only once after the „Delay“.
- You can choose to run it when the aircraft loads („Beginning“) or when the simulator session is finished („End“).

The „Delay“ can be changed with the mouse wheel.



RPN Macros

You can use a simplified version of the XML gauge macros in AAO. Macros are a means for simple text replacement:

→ Macro definition

```
<Macro Name="strobelightmacro">A:LIGHT STROBE, Bool</Macro>
```

→ Macro usage in your RPN scripts:

```
(@strobelightmacro) 1 == if{ 1 (>K:STROBES_TOGGLE) }
```

→ Result at runtime:

```
(A:LIGHT STROBE, Bool) 1 == if{ 1 (>K:STROBES_TOGGLE) }
```

Please note that macro definitions must be kept in their very own scripts. They are static and cannot be mixed with RPN code that would have to be parsed dynamically.

You can also use LVars as macros, AAO will replace the „@variablename“ with the actual value:

```
2 · (>L:engnum)
```

```
(A:ENG · COMBUSTION:@engnum, ·Bool)
```

```
=> (A:ENG · COMBUSTION:2, ·Bool)
```

RPN Operators in AAO

Operator	Operation	Arguments	Example	Result
Common Operators				
+	addition	2	3 5 +	8
-	subtraction. If the stack contains A B -, then the calculation is A - B.	2	(L:Value) 90 -	The local value minus 90.
/	division. If the stack contains A B /, then the calculation is A / B.	2	5 2 /	2.5
*	multiplication	2	pi 2 *	2 pi
%, mod	taking modulo	2	5 3 mod	2
++	increment	1	4 ++	5
--	decrement	1	4 --	3
/-/ neg	negates a number	1	4 -/	-4
Comparison Operators				
==	true if equal	2	(L:Value) 0 == if{ A }	Operation A is carried out if Value is 0.
!=	true if not equal	2	(L:Value) 0 != if{ A }	Operation A is carried out if Value is not 0.
>	true if greater than	2	(L:Value1) (L:Value2) > if{ A } els{ B }	If Value1 is greater than Value2, operation A is carried out, otherwise operation B is carried out.
<	true if less	2	(L:Value1) (L:Value2) < if{ A } els{ B }	If Value1 is less than Value2, operation A is carried out, otherwise operation B is carried out.
>=	true greater than or equal	2	(L:Value1) (L:Value2) >= if{ A } els{ B }	If Value1 is greater than or equal to Value2, operation A is carried out, otherwise operation B is carried out.
<=	true if less than or equal	2	(L:Value1) (L:Value2) <= if{ A } els{ B }	If Value1 is less than or equal to Value2, operation A is carried out, otherwise

Operator	Operation	Arguments	Example	Result
				operation B is carried out.
?	The third operand determines whether the first (True) or second (False) is selected.	3	A B True ?	This evaluates to A.
Bit Operators				
&	bitwise AND	2	5 3 &	1
	bitwise OR	2	5 3	7
^	bitwise XOR	2	5 3 ^	6
~	bitwise NOT	1	5 ~	-6
>>	shift right operand number of bits	2	5 3 >>	0
<<	shift left operand number of bits	2	5 3 <<	40
bytof	bit-wise conversion of double to float	2	1116892707587883008.000 · 4 · bytof	350
bytoi	bit-wise conversion of double to 32bit int	2	1116892707587883008.000 · 0 · bytoi	0
setbit	set a specific bit in a number to 1	3	0x0000 0 1 setbit	2
clrbit	set a specific bit in a number to 0	3	0x0011 0 0 clrbit	16
Bit Operators for byte array = hexadecimal strings ('22AACCFD55')				
&s	bitwise AND	2	'0A00000001' '0000000001' &s	'0000000001'
s	bitwise OR	2	'0A00000001' '0000000001' s	'0A00000001'
^s	bitwise XOR	2	'0A00000001' '0000000001' ^s	'0A00000000'
~s	bitwise NOT	1	'0A00000001' ~s	'F5FFFFFFFE'
>s	shift right operand number of bits	2	'0A00000001' 5 >s	'0050000000'
<s	shift left operand number of bits	2	'0A00000001' 5 <s	'4000000020'
setbit	set a specific bit in a number to 1	3	'010000000001' · 4 · 5 · setbits	'012000000001'
clrbit	set a specific bit in a number to 0	3	'010000000001' · 0 · 0 · clrbits	'010000000000'

Operator	Operation	Arguments	Example	Result
Logical Operators				
!, not	not	1	(L:Local) ! (>L:Local)	Toggles the variable Local
&&, and	and	2	(L:Local) 0xFF00 && (>L:Local)	The variable Local is ANDed with hex 0xFF00
, or	or	2	(L:Local) 07777 OR (>L:Local)	The variable Local is ORed with octal 7777.
Numerical Operators				
abs	Absolute value	1	-5 abs	5
int flr	Calculates nearest integer number which is less than the source number	1	5.98 flr	5
rng	Range; returns True if the third operand lies between values one and two.	3	4 7 6 rng	True
cos	Cosine (input in radians)	1	pi cos	-1
lg	Logarithm to base 10	1	10 lg	1
min	Minimum	2	5 2 min	2
sin	Sine (input in radians)	1	pi sin	0
acos	Arc cosine (returns radians)	1	pi acos	
ctg	cotangent (input in radians)	1	pi ctg	
ln	Natural logarithm	1	2.718282 ln	1
sqr	Square	1	5 sqr	25
asin	arc sine	1	pi asin	
eps	Floating-point relative accuracy	1	1 eps	2 ⁽⁻⁵²⁾
log	Logarithm of operand one, to the base of operand two.	2	8 2 log	3
pi	Pi = 3.14159; puts pi on the stack	0	pi	3.14159
sqrt	Square root	1	25 sqrt	5
atg2	arc tangent with two inputs (input in radians)	2		

Operator	Operation	Arguments	Example	Result
exp	Exponent; e to the power of the operand	1	1 exp	2.718282
max	Maximum	2	5 2 max	5
pow	Power of; the first value to the power of the second	2	2 5 pow	32
tg	Tangent (input in radians)	1	pi tg	0
atg	arc tangent with one input	1	pi atg	
Special Operators				
rnd	Creates a random integer value	1	10 rnd	A random value between 0 and 9
div	Divides integers; its result is always an integer	2	5 3 div	1
ceil	Calculates nearest integer number which is bigger than the source	1	4.3 ceil	5
near	Calculates the nearest integer number, rounding .5 up.	1	4.5 near	5
dnor d360 rdeg	Normalizes an angle expressed in degrees. The result is a value between 0 and 360.	1	-15 dnor	345
rddg	Converts radians to degrees	1	pi rddg	180
dgrd	Converts degrees to radians	1	180 dgrd	pi
rnor	Normalizes an angle expressed in radians, the result is between 0 and 2 pi	1	5 rnor	1.8584
ddiff	Calculates the angular difference between two headings (in degrees)	2	270 45 ddiff	-135
lldist	Distance between to points Lat/Lon	4	40.51·9.21·42.43·10.44·lldist	127.85 nautical miles
llbrg	Bearing from one Lat/Lon to another	4	40.51·9.21·42.43·10.44·llbrg	25.64 degrees
dec2b16	Converts decimal to BCD16 encoding, used in frequency setter events like (>K:COM_RADIO_SET)	1	13025 dec2b16	77861

Operator	Operation	Arguments	Example	Result
ar429	Convert ARINC429 4 byte value	1	14023863173 ar429	454.3399
unixts unixtsms	Convert an "epoch" (number of seconds since 01.01.1970) using a format string The format string can be any C# standard date format or 'd', 'M', 'y', 'h', 'm', 's' will return the specified timestamp element as a number. "unixtsms" can additionally use 'fff'	2	1685433887 'dd.MM.yyyy HH:mm:ss' unixts	'30.05.2023 08:04:47'
time	Current system timestamp in milliseconds	0	time 'HH:mm:ss.fff' unixtsms	'16:11:23.564'
lvar	Return the value of a named LVar	1	'(L:mylvar)' lvar '(L:mystrvar, String)' lvar	0 'test'
if{ }	If statement, note there is no space between the if and the {	1	(L:Value) 0 == if{ A }	Operation A is carried out if Value is 0.
els{ }	Else statement, note there is no space between the els and the {	1	(L:Value1) (L:Value2) <= if{ A } els{ B }	If Value1 is less than or equal to Value2, operation A is carried out, otherwise B
rndsel{...}	Random select, note there is no space between the rndsel and the { There can be only (>K events or AAO commands inside the braces {}. The app will select one of them at random	n	rndsel { (>K:PAUSE_TOGGLE) (SOUND:crew3_preparetakeoff daylight.wav) (SPEAK:Third alternative) (SPEAK:Fourth alternative) }	Depending on the outcome of the random selection, the sim is paused or the sound wav is played or the speech system speaks one of the two phrases
quit	The quit statement allows expression evaluation to stop completely, and avoid the use of nesting if{ statements.	0	pi quit (L:Value1) (L:Value2) <= if{ A } els{ B }	pi. The rest of the script is ignored.
g0...gn	Goto label. Execution will jump to the specified label. Labels are set by entering a colon followed by the label number.	0	g4	Execution jump to :4

Operator	Operation	Arguments	Example	Result
case	Case statement [list of return values] [number of values] [eval value] case	N + 2	50 40 30 20 10 5 (A:var, unit) case	The "5" indicates there are five case values, which are selected depending on the evaluation of (A:var). If the evaluation is equal to or greater than 0, but less than 1, the result is 10. If the evaluation is equal to or greater than 1, but less than 2, the result is 20, and so on.
nonlin	Nonlinear selection [list of values to compare] [number of values] [eval value] nonlin	N + 2	10 15 30 42 100 5 (A:var, unit) nonlin	The "5" indicates there are five case values, which are compared with (A:var). The operator returns the index of the value that is greater or equal to (A:var), in this case 0 – 4
seq	Sequence operator. Returns the value that is next in line after the comparison value [list of values to compare] [eval value] [number of values] seq	N + 2	0 1 2 (L:var) 3 seq (>L:var)	Every call to this script advances L:var through the list 0 - 1 - 2. When it reaches the value 2, it will start over at 0
iseq	Indexed Sequence operator. Returns the value that is next in line after the comparison value starting at a specific index [list of values to compare] [eval value] [index value] [number of values] iseq	N + 3	0 1 2 1 (L:var) (L:idx) 4 iseq (>L:var) (>L:idx)	Advances L:var through the sequence 0 - 1 - 2 - 1 - 0 etc. Current index is stored in L:idx, this must be a unique variable name
n iterate { rpncode }next	Repeat rpncode n times	1	3 iterate { 1 (>K:ELEV_TRIM_UP) }next	Elevator trim is performed three times
procexist	Check if a process is running	1	'prepar3d' procexist	1 when P3D is running, 0 otherwise
fileexist	Check if a file exists (absolute or relative to the AAO documents path)	1	'textfile.txt' fileexist	1 when file exists
direxist	Check if a folder exists (absolute or relative to the AAO documents path)	1	'mysubdir' direxist	1 when folder exists
mousex	Get horizontal mouse position	0	mousex	1288
mousey	Get vertical mouse position	0	mousey	632

Operator	Operation	Arguments	Example	Result
mousecmp	Check if mouse cursor is in a specific rectangular region	4	1000 100 500 50 mousecmp	1 when the mouse is in the box 1000,500 - 1100,550, otherwise 0
String Operators				
'...'	Denotes a string literal in the script (char 39, ALT&0-3-9) When an apostrophe is used inside the string, it must be replaced with "&apos;";		'This is a test'	Assigning a string to a variable: 'This is a test' (>L:MyTestVar, String) Using a string 'This is a test' (SPEAK:%s1)
lc	Converts a string to lowercase	1	'ABcd10' lc	abcd10
uc, cap	Converts a string to uppercase	1	'Abcd10' uc	ABCD10
chr	Converts a number to a symbol	1	65 chr	A
ord	Converts a symbol to an integer	1	'A' ord	65
scat	Concatenates strings	2	'abc' 'red' scat	abcred
scmp	Compares strings, case sensitive	2	'left' 'Left' scmp if{ 'yes' } els{ 'no' }	No
slen	Puts the length of a string on the stack	1	'right' slen	5
stod	Converts a numerical string to a number	1	'123' stod (>K:...)	123 (>K:...)
scmi	Compares strings, ignoring case	2	'left' 'Left' scmi if{ 'yes' }	yes
sstr	Finds a substring	2	'cd' 'abcde' sstr	2
scon	True, if second string contains the first	2	'320' 'Airbus A320' scon	1
ssub	Extracts a substring	2	'ab' 'abcde' ssub	'cde'
substr	Extracts a substring at pos a of length b	3	'abcdef' 2 3 substr	'cde'
symb	Extracts a single character	2	'abc' 1 symb	'b'
srep	Replace parts of a string	3	'abcde' 'bc' 'xy' srep	'axyde'
schr	Find position of character inside string	2	'12345' '3' schr	2
ssplit	Split a string by a substring and return a specific value at index.	3	'ABC DEF GHI' ' ' 1 ssplit	'DEF'

Operator	Operation	Arguments	Example	Result
	An index of -1 splits all items onto the string stack		'ABC DEF GHI' '' -1 ssplit	stack: 'ABC' 'DEF' 'GHI'
asplit	Split a string by a substring and write the results into a String Array	3	'ABC DEF GHI' '' 'arrayname' asplit	(STRARR:arrayname) contains the values ABC, DEF and GHI
shex	Encode a string into a ASCII hex array	1	'ABCD' shex	'41424344'
nhex	Encode a number into a hex array	1	12500 nhex	'0102050000'
%...%!...!	Formatting numbers, d = integer, f = floating point. Precede d with 0 or space for leading zeroes or spaces. %...% can contain RPN code and variables		%2%!02d! %12.34554%!4.2f!	02 12.35
Stack Operators				
c	Clears the numerical stack	0	stack: 1 2 3 c	stack:
d	Duplicates the value that is at the top	1	stack: 5 d	stack: 5 5
p	Pops and discards the top value	1	stack: 1 2 3 p	stack: 1 2
r	Reverses the top and second values	2	stack: 1 2 3 r	stack: 1 3 2
cs	Clears the string stack	0	stack: 'st' 'ab' 'xy' cs	stack:
ds	Duplicates the value that is at the top	1	stack: 'st' ds	stack: 'st' 'st'
ps	Pops and discards the top value	1	stack: 'st' 'ab' 'xy' ps	stack: 'st' 'ab'
rs	Reverses the top and second values	2	stack: 'st' 'ab' 'xy' rs	stack: 'st' 'xy' 'ab'
Numerical Registers				
s0 ... s99 gs0 ... gs99	Stores the value on the numerical stack into a local or a global register, the value remains on the stack.	1	stack: 1 2 3 s0	stack: 1 2 3 s0: 3
l0 ... l99 gl0 ... gl99	Loads a value from a register to the top of the numerical stack	1	stack: 1 2 3 s0 l0	stack: 1 2 3 3
sp0 ... sp99 gsp0 ... gsp99	Stores the top value and pops it from the numerical stack	1	stack: 1 2 3 sp0	stack: 1 2 s0: 3

Operator	Operation	Arguments	Example	Result
String Registers				
ss0 ... ss99 gss0 ... gss99	Stores the value on the string stack into a local or a global register, the value remains on the stack.	1	stack: 'A' 'B' 'Test' ss0	stack: 'A' 'B' 'Test' ss0: 'Test'
sl0 ... sl99 gsl0 ... gsl99	Loads a value from a register to the top of the string stack	1	stack: 'A' 'B' 'Test' ss0 sl0	stack: 'A' 'B' 'Test' 'Test'
ssp0 ... ssp99 gssp0 ... gssp99	Stores the top value and pops it from the string stack	1	stack: 'A' 'B' 'Test' ssp0	stack: 'A' 'B' ss0: 'Test'

AAO specific RPN commands

Command	Description	Example
(CALL:scriptgroup-scriptname)	Call another script. Parameters can be added separated by the symbol	(CALL:Scripts-ACT_HEADING_TO_AP) (CALL:MyScripts-AScript param1 param2)
(EXEC:xxxx,params)	Start a program on your computer, params are optional. You can dynamically insert data with %1 to %9 for numerical values %s1 - %s9 for strings	(EXEC:notepad.exe) will open Notepad (EXEC:notepad.exe,test.txt) will open the file (L:myvar1) (L:myvar2) (EXEC:app.exe,%1 %2) will pass the values of the LVars to the program as parameters
additional EXEC options	<pre>'explorer' 'steam:/' '/rungameid/1465360' scat (EXEC:%s1,%s2)</pre> start Steam App 1465360 (SnowRunner) <pre>'explorer' 'shell:appsFolder\Microsoft.FlightSimulator_8wekyb3d8bbwe!App' (EXEC:%s1,%s2)</pre> start MS Store App (MSFS). App-IDs for Steam and Store apps can be found with the PowerShell command "get-StartApps"	
(EXECBAT:...)	Same as EXEC, but for batch files/shell scripts Does not create an app window	
(WINVAR:xxxx)	Inserts a Windows Environment Variable as string	(WINVAR:windir) => 'C:\Windows'
(>WINVAR:xxxx)	Assign a string to an Environment Variable	'This is a test' (>WINVAR:mywinvar) %(A:PLANE ALTITUDE, feet)%!0.4f! (>WINVAR:mywinvar)
(FOCUS:xxxx)	Set the Windows focus on another process, for example to send virtual key input to it. xxxx is the process name from the task manager, without the ending (.exe,...)	Send a virtual key press to LNM, then switch back to P3D: (FOCUS:littlenavmap) (SPLIT:100) (VKD:29-157-162) (VKD:56-184-164) (VKD:20-148-84) (VKU:20-148-84) (VKU:29-157-162) (VKU:56-184-164) (SPLIT:100) (FOCUS:prepar3d)
(KILL:xxxx)	Terminate a running process	(KILL:littlenavmap)
(SCRIPTFILE:xxx) (SCRIPTFILE:xxx nnn) (SCRIPTFILE:... CACHE)	Load and execute a file with RPN scripts in it. The file must be in plain text format and saved in \Documents\LorbyAxisAndOhs Files\Scripts - nnn can be a line number or a label name - the „CACHE“ option keeps the script in memory	(SCRIPTFILE:mymycript.txt) (SCRIPTFILE:myfolder\mymycript.txt) (SCRIPTFILE:mymycript.txt 23) start the file beginning at line 23

(CHECKLIST:xxx) (CHECKLIST:xxx nnn) (CHECKLIST:.... CACHE)	Load and execute an RPN checklist type script file.	(CHECKLIST:mychecklist.txt) (CHECKLIST:myfolder\mychecklist.txt) (CHECKLIST:mymyscript.txt 23) start the file beginning at line 23
(CONVERSATION:xxx) (CONVERSATION:xxx nnn) (CONVERSATION:.... CACHE)	Load and execute an RPN conversation type script file.	(CONVERSATION:myconversation.txt) (CONVERSATION:myfolder\myconversation.txt) (CONVERSATION:mymyscript.txt 23) start the file beginning at line 23
(WSHSCRIPTFILE:filename language startmethod CACHE ASYNC)	Load and execute a file with Windows Script Host code in it (for example jscript), CACHE and ASYNC are optional, the file is cached and/or executed asynchronously.	(WSHSCRIPTFILE:jstest.txt jscript AaoEntry CACHE) Loads and executes the method "AaoEntry" in the jscript file and caches it for further use.
(GOTO:xxx)	Only available in scriptfiles: jump to the line that starts with „:“ and the desired label.	(L:checkvar) 2 == if{ (GOTO:mymark) } ...RPN code... :mymark ...RPN code
(GOLINE:xxx)	Only available in scriptfiles: jump to line xxx	(L:checkvar) 2 == if{ (GOLINE:123) }
(GOSUB:xxx) / (RETURN)	Only available in scriptfiles: jump to the line that starts with „:“ and the desired label. The code after that label must be concluded with (RETURN) in a single line, which will jump back to the calling code	(L:checkvar) 2 == if{ (GOSUB:mymark) } ...RPN code... :mymark ...RPN code (RETURN)
(END)	Only available in scriptfiles: end the script	
(LVARRPN:xxxx)	Process the contents of a String LVar as RPN code	variable containing code: (L:myvar, String) call: (LVARRPN:myvar)
(VKD:...) (VKU:...)	Virtual Key Down and Key Up events. The RPN Script editor has a special input dialog to map an actual keyboard action to the key codes (Button „Insert VKey“)	(VKD:44-172-89) · (SPLIT:100) · (VKU:44-172-89) ·
(VJBD:dev btn) (VJBU:dev btn)	vJoy button btn down/up on device dev	(VJBD:1 2) (VJBU:1 2)
(VJBD:dev pov dir) (VJBU:dev pov dir)	vJoy discrete POV down/up in direction dir (0=N, 1=E, 2=S, 3=W) on device dev	(VJBD:1 1 3) (VJBU:1 1 3)

(VJAX:dev axis value)	Send value (0 - 32767) to vJoy axis (X,Y,Z,RX,RY,RZ,SL0,SL1) on device dev	(VJAX:1 Y 16500)
(VJPV:dev pov value)	Send value (-1 - 36000) to vJoy continuous POV on device dev	(VJPV:1 3 4000)
(VIGBD:btn) (VIGBU:btn)	ViGEm button down/up X,Y,A,B,Up,Down,Right,Left,RightShoulder,LeftShoulder,RightThumb,LeftThumb,Back,Start,Guide	(VIGBD:LeftShoulder) (VIGBU:LeftShoulder)
(VIGAX:axis value)	Send value (-32676 - 32767) to ViGEm axis LeftThumbX,LeftThumbY,RightThumbX,RightThumbY	(VIGAX:LeftThumbY 6583)
(VIGSL:slider value)	Send value (0 - 256) to ViGEm slider LeftTrigger,RightTrigger	(VIGSL:LeftTrigger 94)
(VMOBD:btn) (VMOBU:btn)	Mouse button (Left,Middle,Right) down/up	(VMOBD:Left) (VMOBU:Left) (VMOBD:Left) (VMOBU:Left) is a double click at the current position
(VMOBD:btn x y) (VMOBU:btn x y)	Mouse button at screen position x,y	(VMOBD:Right 1000 500) (VMOBU:Right 1000 500)
(VMOXY:x y)	Move mouse cursor to screen position x,y	(VMOXY:1000 1000)
(VMOWH:val)	Spin mouse wheel by val (1 click is +/- 120)	(VMOWH:120)
(VMOWH:val x y)	Spin mouse wheel at screen position x,y	(VMOWH:-120 000 500)
(VMOMV:x y)	Mouse movement by x, y "mickeys" "A <i>mickey</i> is the amount that a mouse has to move for it to report that it has moved." (MSDN)	(VMOMV:48 0) Moves the mouse about 100 pixels to the right on a 4K monitor
(SOUND:xxxx) (SOUND:xxxx vvv) (SOUND:xxxx vvv bbb) (SOUND:xxxx vvv bbb dd) (SOUND:xxxx vvv bbb dd ms)	Plays a sound file, the files must be saved here: \\Documents\\LorbyAxisAndOhs Files\\Sounds vvv is the sound volume 0 – 100 bbb is the sound balance -100 – 100 dd is the device id (see "Extras->Show list of sound devices") ms is the duration of the sound in milliseconds	(SOUND:myfile.wav) (SOUND:myfile.wav 50)

(SOUNDLOOP:xxxx), (SOUNDLOOP:xxxx vvv) (SOUNDLOOP:xxxx vvv bbb) (SOUNDLOOP:xxxx vvv bbb dd) (SOUNDLOOP:xxxx vvv bbb dd ms)	Plays a sound file as a loop	(SOUNDLOOP:myfile.wav) (SOUNDLOOP:myfile.wav 50)
(SOUNDVOLUME:xxxx vvv) (RANDOMSOUNDVOLUME:xxxx vvv)	Change the volume of an active sound file (0 to 100)	(SOUNDVOLUME:myfile.wav 75)
(SOUNDBALANCE:xxxx bbb) (RANDOMSOUNDBALANCE:xxxx bbb)	Change the L/R balance of an active sound file (-100 to 100)	(SOUNDBALANCE:myfile.wav -50)
(STOPSOUND:xxxx)	Stops a sound file	(STOPSOUND:myfile.wav)
(RANDOMSOUND:xxxx) (RANDOMSOUND:xxxx vvv) (RANDOMSOUND:xxxx vvv bbb) (RANDOMSOUND:xxxx vvv bbb dd)	Plays a random sound file from a directory in \\Documents\\LorbyAxisAndOhs Files\\Sounds	(RANDOMSOUND:AtcChatter) (RANDOMSOUND:AtcChatter 50) using files from LorbyAxisAndOhs Files\\Sounds\\AtcChatter*.*
(STOPRANDOMSOUND:xxxx)	Stops the currently playing random sound	(STOPRANDOMSOUND:AtcChatter)
(ONESHOT)	Add this to the end of your script if you want it to execute only once during a flight	(LISTEN_FOR_RPN: (A:PLANE·ALTITUDE, ·Feet) ·3 000 ·>) ·if{ ·(SOUND:crew4_climbingdevice.wav) ·} ·(ONESHOT)
(SPEAK:xxxx)	Uses the Windows „TextToSpeech“ feature to make it speak a text. You can dynamically insert data with - '%1' to '%9' for numerical values - '%s1' to '%s9' for strings, append 'u'/'l' for upper/lower case, apped 's' to spell strings in NATO alphabet	This script will make Windows say your current altitude, heading and speed with the voice of „David“ (A:INDICATED·ALTITUDE, ·Feet) ·flr ·(A:PLANE·H EADING·DEGREES·GYRO, ·Degrees) ·flr ·(A:AIRSPE ED·INDICATED, ·Knots) ·flr ·(VOICE:Microsoft·D avid·Desktop) ·(SPEAK:We are at · %1·feet, ·heading·%2·degrees·at·%3·knots) Spelling an ICAO code: (A:GPS WP NEXT ID, String) (SPEAK:Next waypoint is %s1s)

(SPEAKTOFILE:ffff xxxx)	Save the speech output of xxxx to the file Documents\\LorbyAxisAndOhsFiles\\Sounds\\ffff	(SPEAKTOFILE:test.wav This is a test)
(VOICE:xxxx)	Changes the Voice of the Windows „TextToSpeech“ feature. Be mindful that voices are localized, make sure to select a voice that speaks the right language. All installed TTS voices in your Windows system, including their parameters, are shown on this dialog: „Extras->Show List of all TTS voices“ Right-clicking a voice name will copy it to the clipboard, so you can insert it into your script with Ctrl&V	This script says different sentences with different voices. You can only change the voice, rate or volume when the previous sentence has finished (VOICE:Microsoft·David·Desktop)·(SPEAK:Test ing·the·TTS·feature)·(WAIT:3000)·(VOICE:Microsoft·Zira·Desktop)·(SPEAK:I·want·to·test·too)·(WAIT:3000)·(VOICE:Microsoft·Hedda·Desktop)·(SPEAK:Ich·will·Auch·testen)
(VOICERATE:xxxx)	Change voice speed, -10 to 10	(VOICERATE:-10)
(VOICEVOLUME:xxxx)	Change voice volume 0 to 100	(VOICEVOLUME:50)
(VOICEBALANCE:xxxx)	Change left/right balance -100 to 100	(VOICEBALANCE:-75)
(VOICEDEVICE:xxxx)	Change the audio device ID for voice output (see "Extras->Show list of sound devices")	(VOICEDEVICE:1)
(VOICEEFFECTS:aaa-bbb-ccc)	Add an effects chain to the voice (see chapter about sound effects)	(VOICEEFFECTS:7,880,20-1,3-4,50,50)
(VOICEMIX:ffff)	Mix the voice output with the file Documents\\LorbyAxisAndOhsFiles\\Sounds\\ffff	(VOICEMIX:helicotper.wav)
(SAVEFLIGHT:xxxx)	Saves the current flight to the file „xxxx“. The saved flight can be found in the simulators file directory in \\Documents\\	(SAVEFLIGHT:myflight)
(SPLIT:xxx), (SPLIT:xxx yyy)	This command splits the script in two at this point and resumes processing the code after the time is up. All values that have been calculated on the RPN stack before the SPLIT command will be lost, all variables will be read one more time. SPLIT cannot be used inside if/els statements and it can only use fixed values, %n parameters don't work here.	(SPLIT:1000): wait 1 second then process the rest of the code as a new script. (SPLIT:2000 8000): wait a random number between 2 and 8 seconds then process the rest of the code as a new script.

(WAIT:xxx), (WAIT:xxx yyy)	will halt the processing of the script for the desired amount of time. All calculations have already been done at this point, this is only a pause in the execution. It can be used inside if/else structures.	<code>(WAIT:1000)</code> : pause processing for 1 second <code>(WAIT:2000 8000)</code> : wait a random number between 2 and 8 seconds then process with processing.
(CHANGEVIEW:xxxx)	P3D only! Switch the main view to the camera with the name xxx	(CHANGEVIEW:Top-Down)
(OPENVIEW:xxx yyy)	P3D only! Opens the camera yyy in the view xxx	(OPENVIEW:myview Top-Down)
(UNDOCKVIEW:xxx)	P3D only! Undock the view xxx	(UNDOCKVIEW:myview)
(DOCKVIEW:xxx)	P3D only! Dock the view xxx	(DOCKVIEW:myview)
(MOVEVIEW:xxx x y)	P3D only! Move the view xxx to position x,y	(MOVEVIEW:myview 2000 100)
(SIZEVIEW:xxx w h)	P3D only! Resize the view xxx to width, height	(SIZEVIEW:myview 800 600)
(CLOSEVIEW:xxx)	P3D only! Closes the view xxx	(CLOSEVIEW:myview)
(SET6DOF:x y z p b h)	Set the 6DOF position of the main camera x,y,z are in feet p,b,h in degrees (pitch, bank, heading)	(SET6DOF:0.1 0.2 0.3 0 0 90) Move the camera and turn it by 90 degrees.
(READFLIGHTPLAN), (READFLIGHTPLAN:xxxx)	This command reads the current flight plan or a flight plan file into a set of internal Lvars: (L:FPL_WP_CNT) (L:FPL_WP_ALT:n) (L:FPL_WP_LAT:n) (L:FPL_WP_LON:n) (L:FPL_WP_ICAO:n, String) n is the index of a waypoint in the flight plan, starting with 0, FPL_WP_CNT is the total number of waypoints.	<code>(READFLIGHTPLAN)</code> : load the current flight plan <code>(READFLIGHTPLAN:test.pln)</code> : load \\Documents\\<simulator> Files\\test.pln You can access waypoint data dynamically by using a macro LVar 5 (>L:fppos) ... (L:FPL_WP_ALT:@fppos, feet)
(HTTPPOST:www.xxx.yy contenttype zzzzzz)	Send a HTTP POST request to URL „http:// www.xxx.yy “ with content type and content zzzzzz	<code>(HTTPPOST:localhost:8083/graphql application/json {"variables":{"direction":1}})</code>

(HTTPSPOST:www.xxx.yy contenttype zzzzzz)	Same as HTTPPOST but for https://...	
(OPENURL:url)	Open a web page in your default browser. When the url does not start with http, it is assumed that it denotes a relative path to a AAO web page	(OPENURL:a320cdu/index.html) (OPENURL:http://www.google.com)
(QUERYJSON:url user password) (QUERYXML:url user password)	Query a web service returning either JSON or XML. Result is written into LVars of type String. User and password are optional parameters	(QUERYXML:https://aviationweather-cprk.ncep.noaa.gov/adds/dataserver_current/httpparam?dataSource=metars&requestType=retrieve&format=xml&stationString=EDDM&hoursBeforeNow=1) (L:response.data.METAR.raw_text, String): 'EDDM 301850Z AUTO 07004KT 030V100 CAVOK 17/09 Q1020 NOSIG'
(DOWNLOADJSON:url filename user password) (DOWNLOADXML:url filename user password)	Same as above, but results are written to a file instead of LVars (absolute path or relative to \Documents\LorbyAxisAndOhs Files\)	
(SENDMAIL:somebody@domain.com subject body)	Send an email with subject and body to the recipient. Remember to configure your mail server in the menu „Extras“!	(SENDMAIL:me@gmx.de Greetings from AAO This is a test message from AAO)
(LOAD_SIMBRIEF:xxx) (LOAD_SIMBRIEF_PLAN:xxx)	Load the OFP data of user xxx. This includes the tokens "general", "origin", "destination", "alternate", "aircraft", "fuel", "times", "weights", "atc". Every property is written to an LVar (numerical or String), the PLAN is written to the in-sim flightplan	(L:simbrief.origin.icao_code, String) (L:simbrief.destination.trans_alt) etc.

(LOAD_SIMBRIEF) (LOAD_SIMBRIEF_PLAN)	Same as above using the SimBrief user name that you can enter in the AAO menu „Extras“	
(AAO_ONLINE_MODE)	Toggle AAO connection to the simulator	
(AAO_OFFLINE_MODE:xxxx)	Toggle offline mode, load configuration xxx when connecting	(AAO_OFFLINE_MODE:Maule M7 206 paint1)
(LOADDESKLAYOUT:xxxxx)	Load Desktop FIPs/gauges layout named xxxxx	(LOADDESKLAYOUT:Cessna Analog Gauges)
(LOADWEBLAYOUT:xxxxx)	Load Web FIPs layout named xxxxx	(LOADWEBLAYOUT:Cessna Analog Gauges)
(LOADDESKWEBLAYOUT:xxx)	Load the App/Web layout named xxx	(LOADDESKWEBLAYOUT:Support Apps)
(AAO_SHOW_x:n) (AAO_HIDE_x:n)	Show/hide the gauge of type x and index n x is DESKFIP, WEBFIP, APPWEB n is 0 to (number of FIPs in the layout minus 1)	(AAO_HIDE_DESKFIP:2) hides the Desktop FIP located at the third position of the list on the handler dialog
(AAO_SAVE_DESKFIP:n lvar) (AAO_SAVE_WEBFIP:n lvar)	Render the gauge shown at index n as Base64 into (L:lvar, String)	
(RENDERGAUGE:name lvar)	Render the gauge definition with "name" as Base64 into (L:lvar, String)	
(UNLISTEN:xxx)	Stop a "LISTEN_FOR" command. XXX is either the ID provided with the script header or the K - Event	
(REGEXSPLIT:text regex case/nocase arrayname)	This will split a text using a regular expression (case sensitive or not) and saves the results into the array. Use "&pipe;" in your regex instead of the character " "	'testcase·atis·with·A·information'·(REGEXSP LIT:%s1 testcase·atis·with·([A- Z])·information case ATISARRAY)
(HIDWRITE:vid,pid,value)	Write value to the feature buffer of the HID device with PID and VID val can be in numeric or string hexadecimal format 0x11223344 is the same as '11223344'	(HIDWRITE:0x294B,0x1901,'00008000') set the left most bit in the second byte in the feature buffer or a Honeycomb Bravo (Low Hyd Pressure light) sets all others to OFF
(HIDREAD:vid,pid,lvar) (only with specific devices)	Read feature buffer of the HID device with PID and VID into the lvar. Only supply the name of the lvar.	(HIDREAD:0x294B,0x1901,hidtest) Reads the contents of the Honeycomb Bravo (=the state of the lights) into the variable (L:hidtest, String)

(HIDWRITEOR:vid,pid,val) <i>(only with specific devices)</i>	Add value to the feature buffer of the HID device using the bitwise operator; takes the existing buffer into account. Can be used for example to set individual bits in the buffer to 1	'0080000000' (HIDWRITEOR:0x294B,0x1901,'%s1') turns on the autopilot light on the Honeycomb Bravo, leaves all other lights as they were
(HIDWRITEAND:vid,pid,val) <i>(only with specific devices)</i>	Add value to the feature buffer of the HID device using the & bitwise operator; takes the existing buffer into account. Can be used for example to set individual bits in the buffer to 0	0x80000000 ~ (HIDWRITEAND:0x294B,0x1901,%1) turns off the autopilot light on the Honeycomb Bravo, leaves all other lights as they were
(HIDDEVICES)	Writes a list of all HID devices with vid and pid to \\Documents\\LorbyAxisAndOhs Files\\HID_Devices_List.txt	
(GETBUTTONS:guid lvarname)	Writes a CSV string of the current button states on the game controller with "guid" into (L:lvarname, String) The guid can be obtained by right clicking on the controller in the left list of the Hardware Change and Device Blacklist dialogs - this copies the guid in the Windows clipboard and you get insert it into the script	(GETBUTTONS:4efea320-6a76-11ee-8001-444553540000 btnlst) (L:btnlst, String) is "0,0,1,0,-1" = 4 buttons and a POV control. This list will contain as many buttons and POVs as there are on the device.
(GETAXIS:guid lvarname)	Writes a CSV string of the current axis values on the game controller with "guid" into (L:lvarname, String)	(GETAXIS:4efea320-6a76-11ee-8001-444553540000 axilst) · (L:axilst, String) is X,Y,Z,Rx,Ry,Rz,s11,s12
(DICTATION_START:engine language)	Starts the dictation mode. "engine" is one of "Windows", "Azure", "Cortana". Language is a locale string (en-US, de-DE etc.)	(DICTATION_START:Windows en-US)
(DICTATION_STOP) (DICTATION_END)	Will stop the dictation mode. "END" will also terminate the dictation engine, use this when changing language	
(DICTATION_STOP_AND_PARSE) (DICTATION_END_AND_PARSE)	Will stop the dictation mode and send the result to the voice recognition module for processing.	

File system operations

In all commands, if the file name is not fully qualified (C:\...), the file location is assumed to be relative to
\\Documents\LorbyAxisAndOhs Files

Command	Description	Example
(LOADTEXTFILE:filename arname cap)	Loads a text file "filename" into a STRARR "arname" with capacity "cap"	(LOADTEXTFILE:C:\mytextfile.txt MY_MENU 6) loads the first 6 lines of the text file into the array MY_MENU (STRARR:MY_MENU:6) 0 get will yield the first line, (STRARR:MY_MENU:6) 1 get the second one, etc.
(LOADTEXTFILE:filename var)	Loads the contents of the file into a single LVar of type String	(LOADTEXTFILE:C:\mytextfile.txt L:myvar) loads the entire text into (L:myvar, String)
(LOADNUMFILE:filename arname cap)	Loads a text file "filename" where every line is a number (like 32123.4323) into a NUMARR "arname" with capacity "cap"	(LOADNUMFILE:C:\mytextfile.txt MY_VALUES 6) loads the first 6 lines of the text file into the array MY_MENU as numbers (NUMARR:MY_VALUES:6) 0 get will yield the number from the first line (NUMARR:MY_VALUES:6) 1 get the second one, etc.
(WRITEFILE:filename arname APPEND)	Writes the array of name „arname“ into the text file „filename“. APPEND is optional, when it is omitted, the file is overwritten.	(WRITEFILE:C:\mytextfile.txt MY_MENU) overwrites the text file with the contents of the STRARR (WRITEFILE:C:\mynumfile.txt MY_VALUES APPEND) appends the contents of the NUMARR to the file
(DELETEFILE:filename)	Delete the file „filename“	
(COPYFILE:sourcefile targetfile)	Copies sourcefile to targetfile	
(MOVEFILE:sourcefile targetfile)	Moves sourcefile to targetfile	
(DIRLIST:dirname arname cap)	Read the filenames of dirname into the strarr arname with capacity cap (absolute or relative to AAO documents path)	(DIRLIST:MyGauge mystrarr 50)

Command	Description	Example
(LOADIMAGE:filename varname)	Loads a binary image file into a String Lvar in base64 format	<pre>(LOADIMAGE:C:\animage.png myimage) => (L:myimage, String) contains the base64 encoded image</pre>
(LOADIMAGE:filename varname col1 col2)	Loads the image and replaces the color col1 with col2. Colors are HTML hex color codes #RRGGBB or #RRGGBBAA	<pre>(LOADIMAGE:UserGauges\1024\McpEncoderKnob.png picture1 #767676 #0000FF) · => (L:picture1, String) contains the base64, light grey has been replaced with blue</pre>
(WRITEIMAGE:filename varname)	Write a base64 String into a binary image file	<pre>(WRITEIMAGE:C:\animage.png myimage) => (L:myimage, String) is written into a binary file</pre>
(REPLACECOLOR:varname col1 col2)	Replace a color in a base64 image string variable. Colors are HTML hex color codes #RRGGBB or #RRGGBBAA	<pre>(REPLACECOLOR:picture1 #8A8BB066 #FF000044)</pre>
(EXPORTVARS:filename)	Exports all variables that are currently in the AAO cache to the file „filename“. File format is text/TSV (tab separated values)	
(REPLACEVARS:sourcefile,targetfile)	Reads the sourcefile and replaces all variable strings in it with their actual values. The result is written to targetfile. The source file must be in text format.	
(EXPORTAI:filename)	Writes data about all AI objects currently in the sim to the file „filename“. File format is text/TSV (tab separated values)	

Dynamic parameters

You can use dynamic parameters in the AAO Commands

- Insert %1 to %9 to read values from the numerical stack
- Insert %s1 to %s9 to read values from the string stack

Examples:

300 (WAIT:%1) is the same as (WAIT:300)

'Left' 500 200 (VMOBD:%s1|%1|%2) is the same as (VMOBD:Left|500|200)

AAO specific Script headers

Script headers have to be put at the beginning of a script, as they decide about further processing

Header	Description
(WSH:language startmethod ASYNC)	Use the Windows Script Host with the desired language to process the code. With the optional ASYNC parameter the script is processed asynchronously (=in parallel). <code>(WSH:jscript AaoEntry) ...</code> <code>(WSH:vbscript AaoEntry) ...</code>
(SIMPROC)	Sends the script to the simulator for processing. This can help with exotic RPN logic that is not directly supported by SimConnect
(LISTEN_FOR_K:xxxx) <i>scriptcode</i>	This instructs AAO to listen for a specific Event ID (K-Events) Example: <code>(LISTEN_FOR_K:STROBES_TOGGLE) (SOUND:strobestoggled.wav)</code> will play a sound file every time the K:STROBES_TOGGLE event is received
(LISTEN_FOR_VOICE:xxxx) <i>scriptcode</i> (LISTEN_FOR_VOICE:xxxx id)	With this script header the scriptcode will be executed when AAO receives the voice command Multiple voice commands can be grouped with the symbol or you can use an SRGS file too. Add „(ONESHOT)“ to the end of the script code to make it a single event. The id is for the UNLISTEN command. <code>(LISTEN_FOR_VOICE:mayday help) (SPEAK:What is the problem?)</code>
(LISTEN_FOR_RPN:evalcode) <i>scriptcode</i> (LISTEN_FOR_RPN:evalcode id) (REPEAT_WHILE_RPN:evalcode) <i>scriptcode</i>	This will call the scriptcode when or while a specific condition is met - „evalcode“ is an RPN script that must return either 0 (false) or 1 (true). The result is passed on to the script. Add „(ONESHOT)“ to the end to make it a single event. The id is for the UNLISTEN command. <code>(LISTEN_FOR_RPN:(A:LIGHT STROBE, Bool) 1 ==) if{ (SPEAK:Strobes are on) (ONESHOT) }</code>
[rpn_condition] <i>rest of script</i>	If a script starts with an RPN conditional expression in [], then AAO will wait until that condition is met before executing the rest of the script. Contrary to LISTEN_FOR_RPN, this script is executed only once. <code>[(A:AUTOPILOT·HEADING·LOCK·DIR, ·Degrees) ·flr· (A:PLANE·HEADING·DEGREES·GYRO, ·Degrees) ·flr·==] · (SPEAK:Heading reached)</code>

RPN value arrays

In AAO scripts you can use arrays of numerical or string values. They are used similar to normal variables, with the following syntax:

(NUMARR:<arrayname>:<capacity>)

references an array of numerical values. Example: (NUMARR:mynumarray:5) is an array with a capacity of 5 numerical values.

(STRARR:<arrayname>:<capacity>)

references an array of strings. Example: (STRARR:mystrarray:8) is an array of 8 strings.

The capacity is a hard limit. If the number of values exceeds the capacity, the *first* value (index 0) will be removed from the array.

```
1 (NUMARR:myarr:4) push 3 (NUMARR:myarr:4) push 5 (NUMARR:myarr:4) push 7 (NUMARR:myarr:4) push [1,3,5,7]
```

```
9 (NUMARR:myarr:4) push [3,5,7,9]
```

```
(NUMARR:myarr:4) sum [24]
```

```
(NUMARR:myarr:4) 2 get [7]
```

```
12 (NUMARR:myarr:4) 2 set [3,5,12,9]
```

```
(NUMARR:myarr:4) sum [29]
```

```
(NUMARR:myarr:4) avg [7.25]
```

RPN dynamic lists

Dynamic lists are similar to arrays but they don't have a preset capacity. You can push as many values as you like into the list.

(NUMARR:<arrayname>)
references a dynamic list of numerical values. Example: (NUMARR:mynumlist)

(STRARR:<arrayname>:)
references a dynamic list of strings. Example: (STRARR:mystrarray)

You can use the RPN operator "next" to iterate through a list.

```
1 (NUMARR:myarr) push 3 (NUMARR:myarr) push 5 (NUMARR:myarr) push 7 (NUMARR:myarr) push [1,3,5,7]
9 (NUMARR:myarr) push [1,3,5,7,9]
(NUMARR:myarr) sum [25]
(NUMARR:myarr) 2 get [5]
12 (NUMARR:myarr) 2 set [1,3,12,7,9]
(NUMARR:myarr) sum [32]
(NUMARR:myarr) avg [6.4]
```

The following operators can be used with arrays and lists

Operator	Operation	Arguments	Example
push	Add a value to the end of the array	1	12.5 (NUMARR:myarr:4) push '12.5' (STRARR:myarr:4) push
pop	Remove the last value of the array and put it on the current processing stack	0	(NUMARR:myarr:4) pop (STRARR:myarr:4) pop
peek	Read the last value of the array without removing it and put it on the current processing stack	0	(NUMARR:myarr:4) peek (STRARR:myarr:4) peek
clear	Delete all variables in the array	0	(STRARR:myarr:4) clear
get	Get the value at a specific index of the array if it exists, index goes from 0 to the current number of values	1	(NUMARR:myarr:4) 2 get (STRARR:myarr:4) 1 get
set	Replace the value at specific position of the array, index goes from 0 to the capacity	2	12.5 (NUMARR:myarr:4) 2 set '12.5' (STRARR:myarr:4) 1 set
sum	Return the sum of all values (numerical) or a concatenated string of all values (string)	0	(NUMARR:myarr:4) sum (STRARR:myarr:4) sum
avg	Return the average of all values (numerical arrays only)	0	(NUMARR:myarr:4) avg
count	Return the number of values in the array	0	(NUMARR:myarr:4) count (STRARR:myarr:4) count
cnt	Check if an array contains a value and return the index. In case of string arrays, a matching substring will already return a hit.	1	'12345' (STRARR:myarr:4) 2 set '23' (STRARR:myarr:4) cnt returns 2
next	Return the next value in an array	0	(STRARR:myarr) next (>L:currentvalue, String)
rset	Reset the list counter (for next) to 0	0	(STRARR:myarr) rset
revs	Reverse the array	0	(STRARR:myarr) revs
sora, sord	Sort the array ascending or descending	0	(STRARR:myarr) sora
tocsv	Returns a string of all values separated by a char	1	' ' (STRARR:myarr) tocsv -> 'first second third'

RPN HashMaps

Hashmaps associate a string literal of your choice with another literal, or with an LVar, Array or Dynamic List.

Example 1: storing simple values in a HashMap:

```
'one' 'item1' (HASHMAP:mymap) mset  
'two' 'item2' (HASHMAP:mymap) mset
```

```
'item1' (HASHMAP:mymap) mget => 'one'  
'item2' (HASHMAP:mymap) mget => 'two'
```

```
'three' 'item1' (HASHMAP:mymap) mset  
'item1' (HASHMAP:mymap) mget => 'three'
```

Example 2: setting the values of two Lvars and storing the reference to the LVar in the HashMap:

```
(HASHMAP:mymap) mclr  
'Test' (>L:tester, String) 25 (>L:ntester)  
'L:tester, String' 'strvar' (HASHMAP:mymap) mset  
'L:ntester' 'numvar' (HASHMAP:mymap) mset
```

Retrieving the Lvar values

```
'strvar' (HASHMAP:mymap) mget => 'Test'  
'numvar' (HASHMAP:mymap) mget => 25
```

In this case, the HashMap keeps only a reference. When the value of the LVar is changed, retrieving the variable from the HashMap will return the new value too.

```
(L:ntester) 10 + (>L:ntester)  
'numvar' (HASHMAP:mymap) mget => 35
```

Instead of L: Variables you can also use STRARR and NUMARR items.

The following operators can be used with HashMaps:

Operator	Operation	Arguments	Example
mset	Add a value to the end of the array	2	'one' 'item1' (HASHMAP:mymap) mset
mget	Read the value referenced by the key from the map and put it on the current processing stack	1	'item1' (HASHMAP:mymap) mget
mrem	Remove the entry referenced by the key	1	'item1' (HASHMAP:mymap) mrem
mclr	Delete all entries in the map	0	(HASHMAP:mymap) mclr
mexist	Check if a key exists in the map, returns 0/1	1	'item1' (HASHMAP:mymap) mexist

Handling SimObjects with RPN scripts

AAO RPN has three commands that can be used to handle simulated objects.

Create a simulated object:

(CREATEOBJECT:title,REL|ABS,x,y,z,p,b,h,speed,onground,freeze,varname)

- *title*: is the „title=“ part from the sim.cfg of a simobject.
- *REL|ABS*: is the way of positioning an object, RELative to your position or at an ABSolute lat/lon location
- *x,y,z*: when REL is used, these are „bearing,distance,altitude offset“ relative to your own position.
When ABS is used they are „latitude,longitude,altitude“
- *p,b,h* are pitch,bank,heading of the created object. Heading is also relative or absolute
- *speed* is the initial speed in knots of the object
- *onground* is 0/1 and tells the sim if the object is supposed to be created on the ground or not
- *freeze* is 0/1 and when set, causes the object to freeze in its initial position
- *varname* is an lvar name of your choice to store the ObjectID of the simobject once it has been created

Example:

(CREATEOBJECT:VEH_air_firetruck_sm,REL,45,400,0,0,0,45,0,1,0,objid_1)

This creates a fire truck, 45 degrees to your right, 400 feet away, facing 45 degrees off your own heading, on the ground, storing the ObjectID in the variable (L:objid_1)

Remove a simulator object

(REMOVEOBJECT:objectid)

To remove an object you need the ObjectID that is generated by CREATEOBJECT

(REMOVEOBJECT:@objid_1) will remove the truck that we created earlier

Send a route to a simulator object

Some SimObjects can move around when they are being sent a waypoint list. These are most ground vehicles plus some animals and people. Not all objects will move when you send them a route though. Also, be mindful of the simulators collision detection, normally, objects will not move close to your own vehicle.

(SENDWAYPOINTS:objectid,filename,loop)

- *objectid* is the again the ID thas was generated by CREATEOBJECT
- *filename* is the name of a file with the waypoint list in the Scripts folder of AAO:
Documents\LorbyAxisAndOhs Files\Scripts\

- *loop* is 0/1, telling the sim if the object shall move around the route perpetually.

Example:

(SENDWAYPOINTS:(L:objid_1),relwps.csv,1)

with the file „relwps.csv“ looking like this:

REL, 90, 400, 0, 5, 1

REL, 90, 450, 0, 5, 1

REL, 90, 400, 0, 5, 1

REL, 90, 500, 0, 5, 1

The waypoints in the file have this syntax:

ABS|REL,x,y,z,speed,onground

- *REL|ABS*: is the way of positioning an object, RELative to the last waypoint or at an ABSolute lat/lon location
- *x,y,z*:
when REL is used, these are „bearing,distance,altitude offset“ relative to the last waypoint.
when ABS is used they are „latitude,longitude,altitude“
- *speed* is the speed in knots at the waypoint
- *onground* is 0/1, telling the sim that the waypoint is in the air or on the ground

6. Using other script languages than RPN

Instead of RPN code you can write scripts in all languages that the Windows Script Host (WSH) supports. In order to utilize the WSH capabilities, prepend your script with the "(WSH:language" script header. Your script language must be able to deal with objects using the "obj.property" syntax.

Example for JScript:

```
(WSH:jscript|AaoEntry)
function AaoEntry(){
  for(var i = 0; i < 1000; i++){
    (L:mylvar) = 1+7+(L:mylvar);
  }
  (L:mylvar, .String) .= .GetTitle();
  for(var i = 0; i < 10; i++){
    (NUMARR:testarr).push(i + (NUMARR:testarr).count());
  }
  (NUMARR:testarr).set(5, (NUMARR:testarr).get(5) + 1999);
  AaoCmd.exec("(SPEAK:Dies ist ein Test!)");
  return;
}

function GetTitle(){
  return .(A:TITLE, .String) .+ ."->.tested";
}
```

Note that JScript is not Javascript. Both share the syntax, but they are not the same. In JScript there is no "let" or "const", there are no objects, classes, imports or lambdas. In WSH it is also missing all the web context, so anything dependant on "window" or "document" will not work.

Example for VBScript

```
(WSH:vbscript|AaoEntry)
Function AaoEntry()
    Dim i
    Dim aVal
    For i = 0 To 1000
        (L:mylvar) = 1+7+(L:mylvar)
    Next
    (L:mylvar, String) = MyStringFunc((L:mylvar, String))
    For i = 0 To 10
        (NUMARR:testarr).push(i + (NUMARR:testarr).count())
    Next
    aVal = (NUMARR:testarr).get(5)
    (NUMARR:testarr).set 5, aVal + 1999
    AaoCmd.exec("(SPEAK:Dies ist ein Test!)")
End Function

Function MyStringFunc(lvar)
    MyStringFunc = lvar + ", seven"
End Function
```

Web resources

[Windows Script Host - Wikipedia](#)

[JScript Language Reference \(Windows Scripting - JScript\) \(archive.org\)](#)

Syntax rules for WSH:

- The method to call when the script is activated must be provided in the WSH: header:
(WSH:language|method)
- Parameters for that method can be added with a comma: (WSH:language|method,p1,p2,...)
- Be mindful of the AAO script parameters: do not use "param1" "param2" etc. in code unless required (= when calling the script as a K-Event with parameters)
- Simulator Variables are written directly into the code, with the same syntax as in RPN.
- The ">" setter-symbol is not used in WSH, variables are assigned directly with "="
- Simulator events (K:) are also not written with the ">" syntax. Instead you call the "exec" function on them and supply the value as parameter:
`(K:HEADING_BUG_SET).exec(240);`
- Dynamic lists (NUMARR, STRARR) can be used, but they aren't native script objects. You have to apply the same commands to them as in RPN (push, pop, etc., check the example above and the table in the chapter about "RPN dynamic lists")

- Hashmaps (HASMAP) can also be used, but they too aren't native script objects. You have to apply the same commands to them as in RPN (mget, mset, etc., same as with the dynamic lists, and check the table in the chapter about "RPN HashMaps" above)

Special AAO features for WSH

- **The "AaoCmd" object:**
 - The special AAO Commands or other RPN code can be called with "exec" (ansychronously) and "execwait" (synchronously, unless you are calling an asynchronous feature!):

```
AaoCmd.exec("(SPEAK:This is a test!)");
```

```
AaoCmd.execwait("(SPEAK:This is a test!)");
```
 - AaoCmd has "setTimeout/clearTimeout" and "setInterval/clearInterval". They work the same as in Javascript (but they are avalilable in all WSH languages)

- **Includes:**
 - You can include other scripts into your code using <include> tags:

```
(WSH:jscript|AaoEntry)
<include src="Tester-Alibrary" lang="jscript" alias="testlib" >
function AaoEntry() {
(....)
```

In this case we are including another jscript called "Alibrary", located in the group "Tester":

```
(WSH:jscript|speak)
function speak(vstr){ AaoCmd.exec("(SPEAK:Here " + vstr + ")");}
function add(n1,n2){ return n1 + n2;}
```

- We can then use the defined alias to call the methods of this library in our main script
 - `testlib.call("speak","tester");`
Calls the method "speak" from the Alibrary and passes "tester" as parameter
 - `var test = testlib.call("add",5,3);`
Calls the method "add" from the library and returns "8".
- The script referenced by the <include> can be written in any valid WSH language. For example, you can write a VBScript library that you then call from your JScript.

- **Inserts:**

- it is also possible to paste the code from a different script into the current one:

```
<insert src="Tester-Alibrary">
```

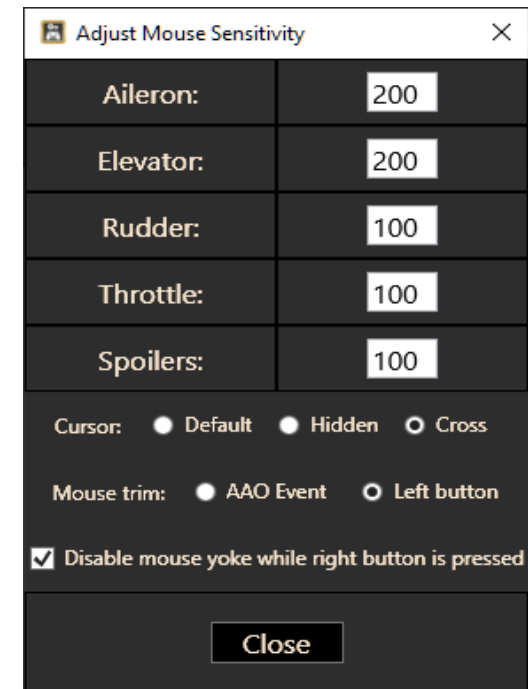
- This is a simple text replacement, AAO will past the code from the source script into your current one, at the exact position of the <insert>. The inserted script must use the same language.

Both <include> and <insert> can reference either scripts from the AAO database (src="group-name") or script files located in \Documents\Lorby AxisAndOhs Files\Scripts (src="filename").

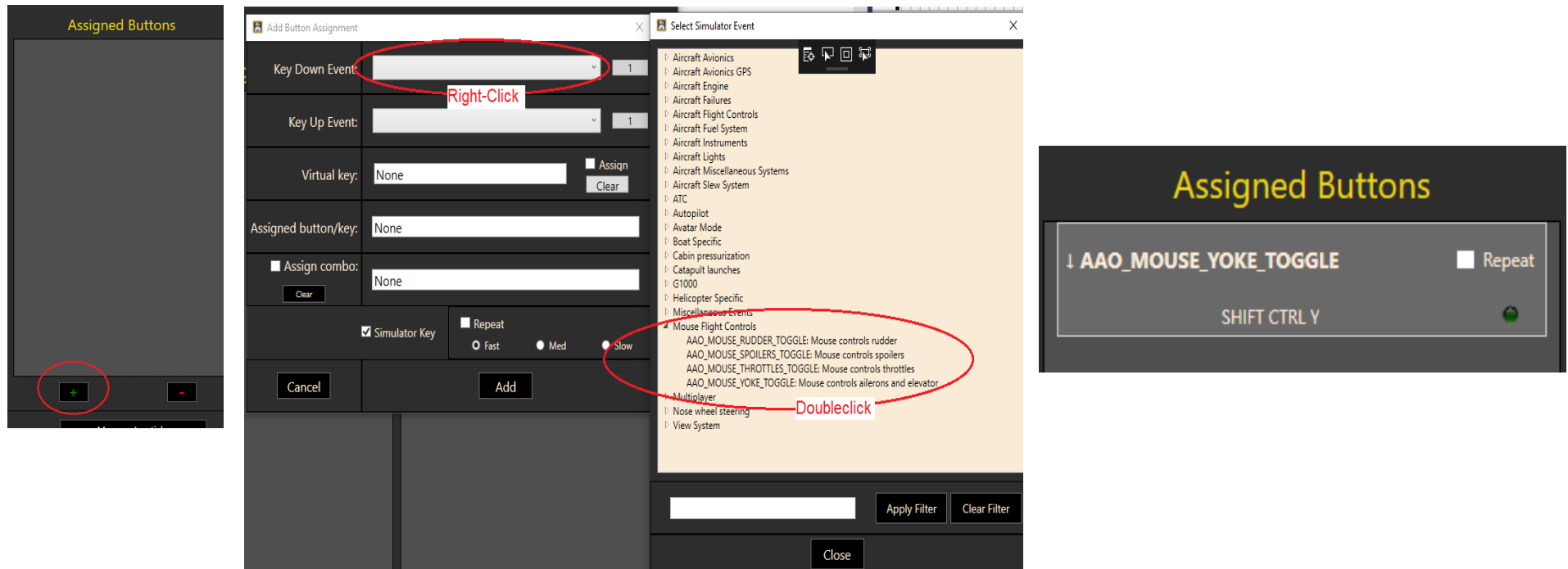
7. Mouse Yoke

AxisAndOhs has various options to use your mouse as a flight control device. You can use the mouse as yoke, rudder, throttle or spoiler handle. Mouse sensitivity and cursor type can be changed in the “**Hardware**” menu:

- To change the sensitivity values spin the mouse wheel over the numerical fields
- A value of 100 is a 1:1 translation of screen pixel distance into axis movement. Increasing the value increases the speed of the mouse movement, decreasing the value slows it down.
- You can choose a cursor type that shall be visible when the mouse yoke is active. Be mindful that this is a system global setting, should AAO crash, the cursor will remain that way until you restart the app and toggle mouse yoke again
- „Mouse trim“: the re-centers the mouse cursor, so you don't have to „jump“ with the mouse. You can select if you want to trigger the trim with the left mouse button or via an AAO event that you assign to a joystick button or keyboard key
- You can activate an option that disables the mouse yoke as long as you keep the right mouse button pressed.



To access the mouse yoke modes you have to assign the corresponding toggle events to buttons or keys:



In this example, “Shift Ctrl Y” has been assigned as the yoke toggle. Pressing these keys switches the mouse yoke on, pressing them again switches yoke mode off.

Note: On Windows systems, only the app that has the focus will receive keyboard events. If you want to use keyboard combos while the mouse yoke is active, you have to keep the mouse cursor setting to „Default“.

8. Enhanced Power Management (Win 8.1 and later)

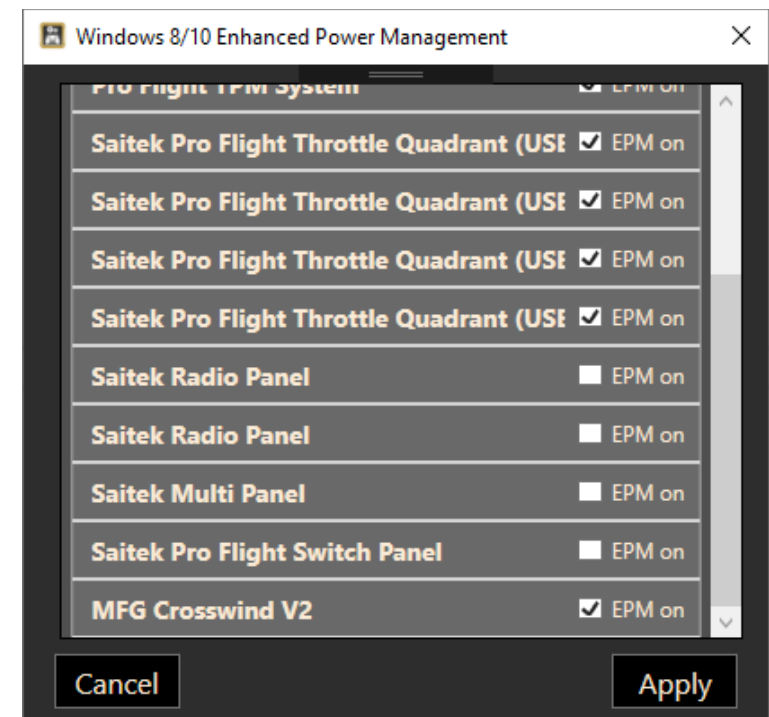
The older USB devices will have issues when they are used with a computer that is running Windows 8 or later, because of the “Enhanced Power Management” feature built into these operating systems. The most notable case is the Saitek Multi Panel, where the display would light up, but not show any text on it.

To help with this, you can disable the EPM using the “Win 8/10 Enhanced Power Management” feature in the “Tools” menu.

On this dialog you will see all devices that LAAO recognized and their current EPM status.

Turn EPM off if you experience issues with devices, like missing functionality or the device constantly connecting and disconnecting.

Note: to use this dialog it is necessary to run the app “As Administrator”.



9. Saitek Panels

Lorby AxisAndOhs can manage the Saitek Radio-, Multi- and Switchpanel devices. When you select “Enable Saitek” from the “Hardware” menu, the app will search for attached devices and connect them to the appropriate simulator events.

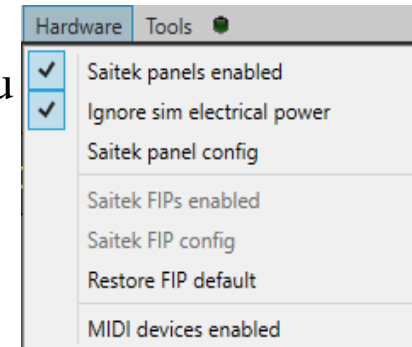
Radio panel special functionality:

- Holding down the button for one second switches between 25 and 8.333Hz spacing (P3D V5)
- When in “DME” mode, the button switches between DME1 and DME2
- the “ADF” setting shows ADF1 on the left and ADF2 on the right. Hold down the button on the device for 1 second to switch between the rotary controller altering ADF1 or ADF2.
- In the XPDR setting, the device shows the transponder code on the left and the current QNH on the right. Use the button to switch between editing the transponder code and adjusting the Kohlsmann setting
 - Editing the transponder code: the inner dial changes the numbers, the outer dial select the digit to be altered
 - Changing the Kohlsmann: the inner dial changes the value, the outer dial switches between mb and inHg.

Radio- / Switch- / Multipanel configuration

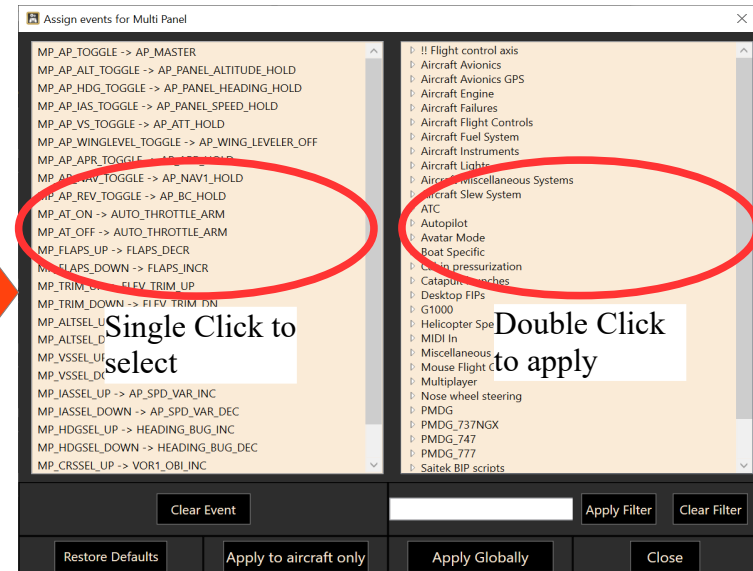
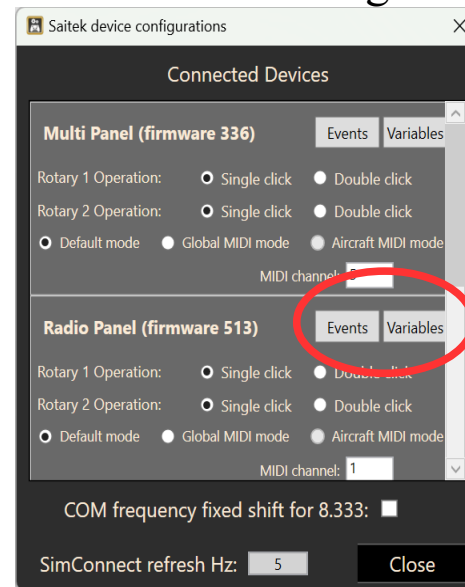
You can change the parameters for panel operations in the „Hardware“ menu

- “Saitek panels enabled” turns the panel connection on or off
- “Ignore sim electrical power” will activate the panel displays regardless of the availability of electrical power in the current aircraft
- “Saitek panel config” opens the event configuration dialog:



The rotary encoders can be set to single or doubleclick action. On older Radio Panels the double click is the default.

„**MIDI mode**“ disables all event assignments entirely. The device will send MIDI events instead. These you can assign as buttons in AAO.



On the “**Events**” and „**Variables**“ dialogs you can alter the configuration of a panel, what events are being sent and what variables are displayed. Click once on the item that you want to change in the left list, then doubleclick on the replacement event in the right list.

Lowering the SimConnect refresh rate (mouse wheel) can help with stutters in the sim (default: 18 Hz).

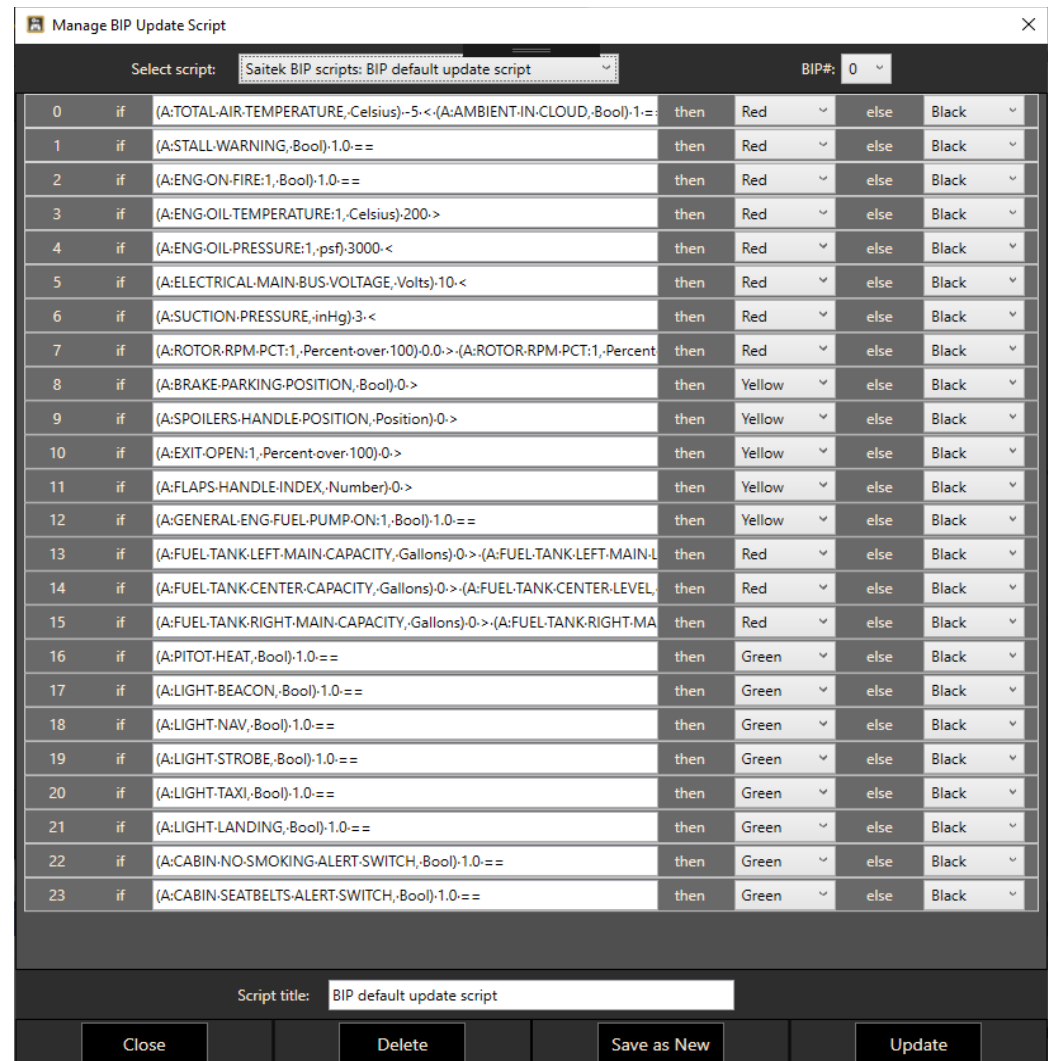
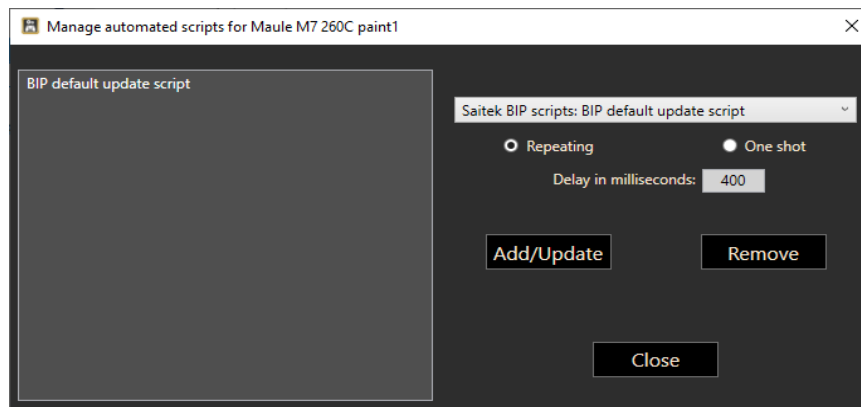
Backlit Information Panel “BIP”

BIPs are supported through RPN scripts. The general idea is that you create an automated script that sets the LEDs on the BIP according to simulator variables. AAO has a special dialog to create this update script in „**Hardware → Saitek BIP configuration**“.

Right click into the large text boxes to copy a simulator variable definition into the script.

You can enter any RPN code that you want, as long as it yields a boolean result of 0 or 1, that the RPN „if{„ clause can understand.

Saving or updating the list will create a single RPN script, that you can assign to your aircraft with „**Scripting → Automated scripts**“.



It is recommended to select a large delay, since most annunciator lights are not time critical.

Remember that each automated script is assigned to each individual aircraft configuration!

The default update script will service the following layout:

#0 Icing conditions	#1 Stall warning	#2 Engine Fire	#3 Oil Temp. high	#4 Oil pressure low	#5 Voltage low	#6 Vacuum low	#7 Rotor RPM low
#8 Parking brake	#9 Spoiler deployed	#10 Door open	#11 Flaps deployed	#12 Fuel Pump on	#13 Left tank low	#14 Center tank low	#15 Right tank low
#16 Pitot Heat on	#17 Beacon	#18 Nav lights	#19 Strobe	#20 Taxi Lights	#21 Landing Lights	#22 No smoking	#23 Seat belts

Flight Instrument Panels “FIP”s

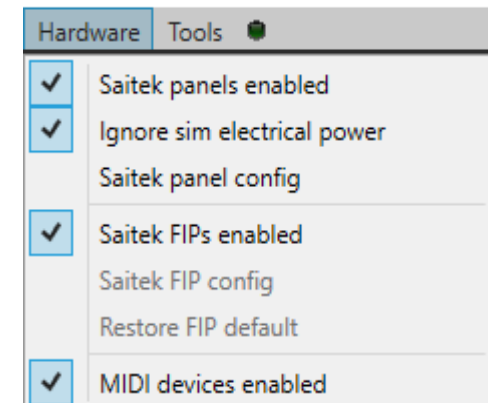
AxisAndOhs can interface with Saitek/Logitech “Flight Instrument Panel” devices.

- It is required to install the original Logitech 64 bit drivers version 8.0.150.0, released 2018-04-13 for the FIP. This is the older driver version, that still has the third party API that AAO needs.
- *It is NOT necessary or recommended to install a PlugIn from Logitech!*
- Every time before starting AxisAndOhs, make sure that the FIP devices have been activated by the driver.
 - The page up/page down buttons will blink red for a few seconds
 - The screen then turns to the Logitech default animation:



The devices can only be used by AxisAndOhs when they are in this state.

- Activate the FIPs connection in the “Hardware” menu:



If you want to substitute or add gauges, the definition files are located here:
C:\Users\...\AppData\Local\LORBY_SI\LorbyAxisAndOhs\FipGauges_*\

FIP configuration (you have to connect and run the FIPs at least once before this dialog works!)

Selecting “Saitek FIP config” in the “Hardware” menu opens the configuration dialog:

- The combobox “Select gauge” contains all gauge definitions that are available
- In the column “Label” you can alter the text that is displayed next to the buttons on the FIP
- In the column “Event-ID” you can change the simulator event that is associated with the button
Right-Click on the text to open the Event selection dialog
- The combobox “Select device” contains all FIPs that are present in the configuration file
- The text below the selection shows the gauge that this FIP is displaying.
To change it, select a gauge at the top, then press “Change gauge”
- “LEDs on” controls if the buttons on the FIP should be illuminated or not.

	Label	Event-ID
Button 1:	Map	FLIGHT_MAP
Button 2:	Main Panel	PANEL_1
Button 3:	Radios	PANEL_2
Button 4:	GPS	PANEL_3
Button 5:	Panel 4	PANEL_4
Button 6:	Panel 5	PANEL_5

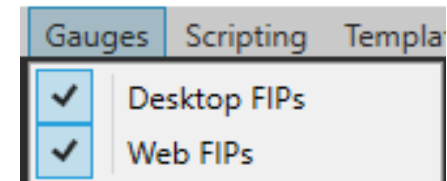
Select device: MZDC8E8352

Gauges\Altimeter.xml ☒ LEDs on Change gauge

Close

10. Desktop FIPs

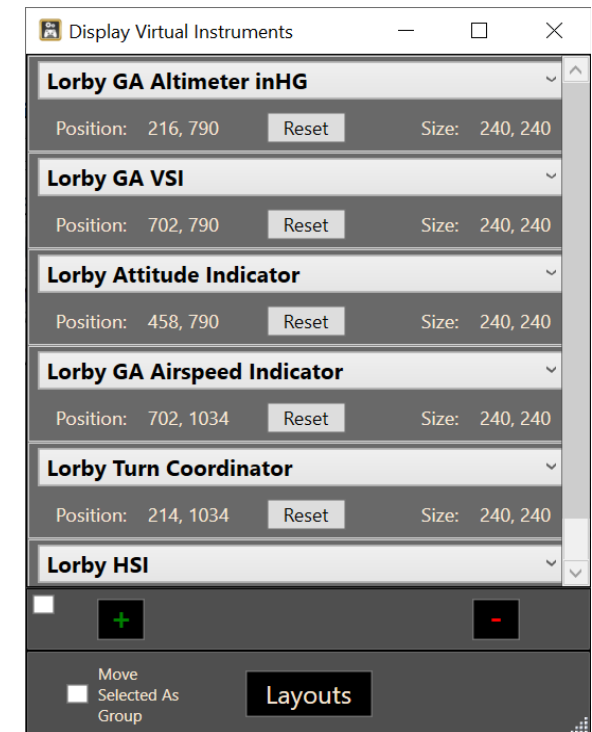
With the “Desktop FIPs” feature you can create virtual instruments on your desktop. To activate it, use the “Gauges” menu, and select “Desktop FIPs”.



This will open the management console:

- To add a virtual instrument, click on the “+” button
- To change the displayed instrument, select the desired gauge from the grey drop down list
- To remove an instrument, click into the corresponding panel so it turn red, then click on the “-” button
- Position and Size can be changed by spinning the mousewheel over the numbers.
- Selected gauges can be moved as a group with the checkbox

The virtual instruments can be dragged to any location and resized. If the gauge has mouse areas, the four soft keys at the bottom will trigger the associated events (altimeter setting, course, heading etc.)





The gauge assets are saved here: \Documents\LorbyAxisAndOhs Files\UserGauges

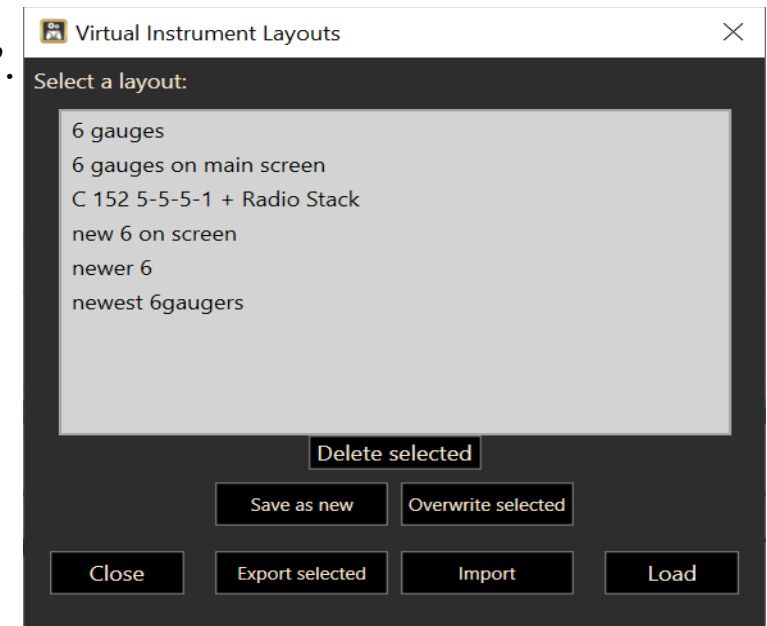
Saving and loading different instrument layouts

You can save different instrument configuration as “Layouts”.

The button “Layouts” will open the management dialog:

You can

- Save the current instrument layout as a new item or overwrite an existing one
- Load a previously saved layout
- Import and export layouts (you can select more than one layout for the export)
- Delete layouts



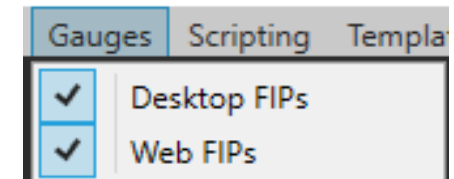
Special Commands for FIPs

AxisAndOhs has a couple of special commands built in for the FIPs. With these you can open, close, refresh FIPs or reload the layout. You will find the commands in the group "Desktop FIPs" on the event selection dialog (when assigning a button or adding an event to a script).

11. Web FIPs

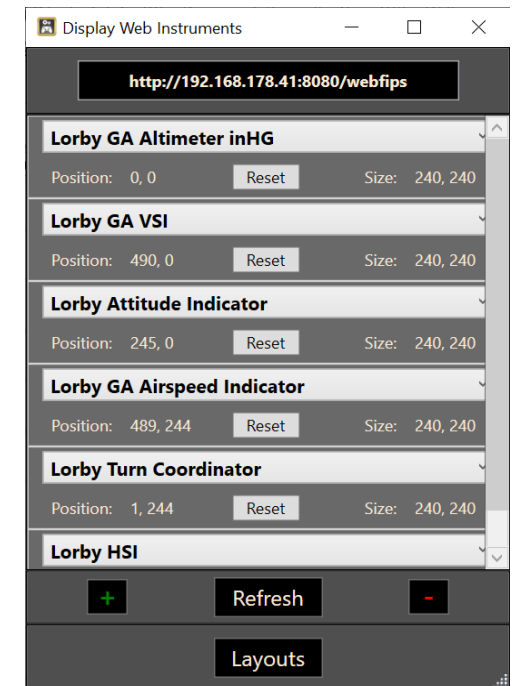
For the Web FIPs the AxisAndOhs app must be run “As Administrator”!

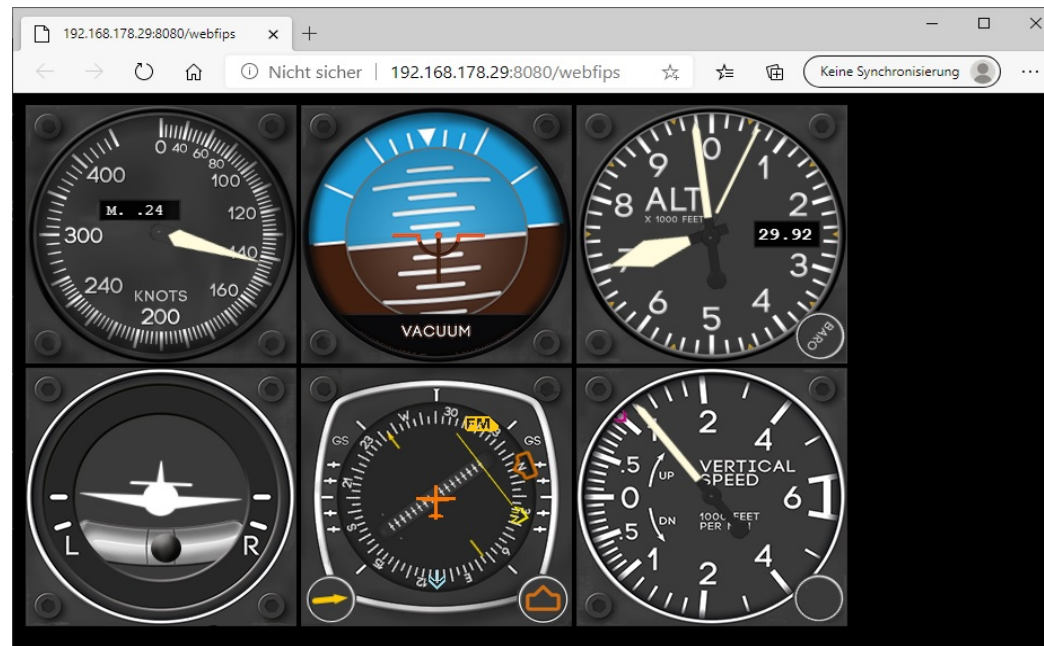
With the “Web FIPs” feature you can access virtual instruments using a standard HTML5 capable web browser on a device that has network access to your computer. To activate it, use the “Gauges” menu, and select “Web FIPs”. Aircraft layouts work the same way as the Desktop FIPS, see previous page.



This will open the management console:

- The textbox at the top displays the URL that you have to enter into the browsers address field. Doubleclicking it opens your local default browser to this URL
- To add a web instrument, click on the “+” button
- To change the displayed instrument, select the desired gauge from the grey drop down list
- To remove an instrument, click into the corresponding panel so it turns red, then click on the “-” button
- Position and Size can be changed by spinning the mousewheel over the numbers. Make sure to press „Refresh“ to apply changes





If you have other applications on your local network that use port 43280, the WebFIPs won't work. You can change the port using "**Tools->Configure WebFIP port**". Restart the app after the change.

The gauge assets are saved here: `\Documents\LorbyAxisAndOhs Files\UserGauges`

Accessing the WebFIPs

Enter the URL shown on the management dialog into your web browser:

```
http://192.168.178.29:43280/webfips
```

The WebFIPs can also be accessed separately. Append either the index of the FIP or the FIP name or the name of the XML file to the URL (indexes go from top to bottom, starting with 0)

```
http://192.168.178.29:43280/webfips/0
```

```
http://192.168.178.29:43280/webfips/Lorby GA VSI
```

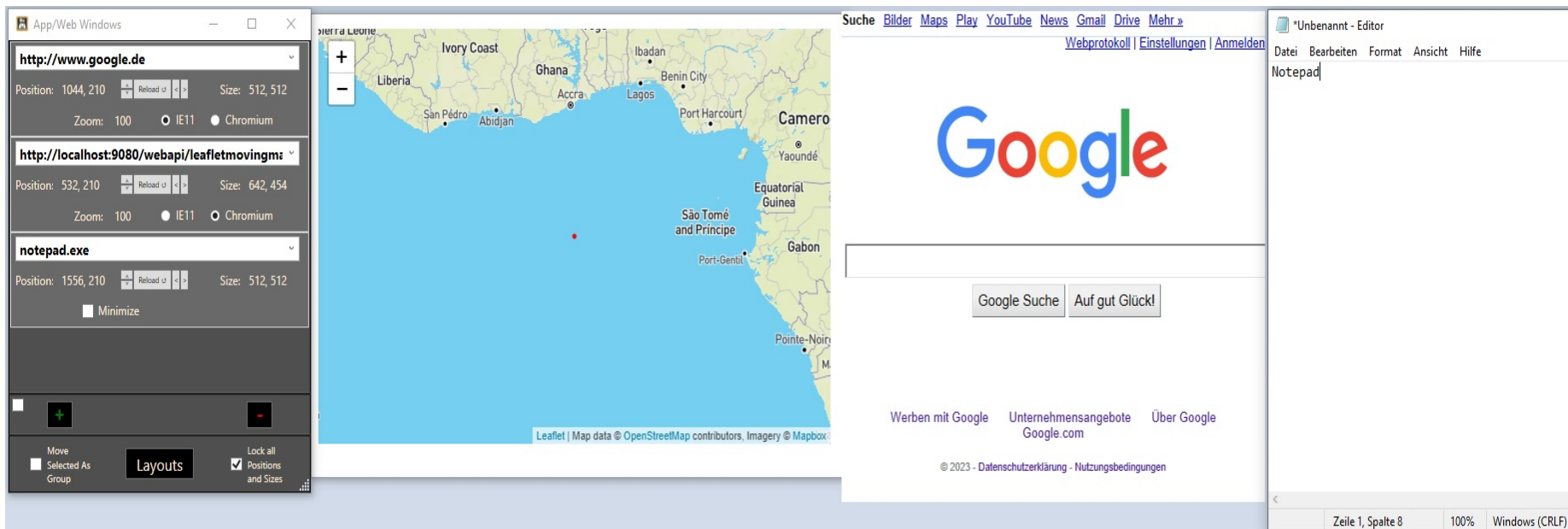
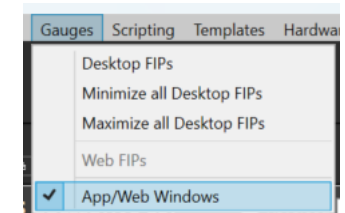
```
http://192.168.178.29:43280/webfips/LorbyGaAltimeterHG.xml
```

Special Commands for FIPs

AxisAndOhs has a couple of special commands built in for the FIPs. With these you can open, close, refresh FIPs or reload the layout. You will find the commands in the group "Desktop FIPs" on the event selection dialog (when assigning a button or adding an event to a script).

12. App and Web windows

In the "Gauges" menu you will also find the "App/Web Windows" feature. This allows you to start other apps and display web pages on fixed positions of your Desktop.



The management dialog works the same way as the Desktop and Web FIPs. Use "+" to add a new window, and "-" to remove it.

Type in a URL, the path to an executable (command line parameters can be provided after a comma), the name of a running process as it appears in TaskManager or select one of the AAO web pages from the dropdown control. Then click on "Reload".

If the targeted app is already running, AAO will grab that window, relocate and resize it.

Initially the windows will be displayed as empty apps that you can drag and resize. "Lock"ing them with the option on the bottom right of the dialog will remove the title bars of those windows and bring the target app to front.

Position, Size and Zoom can be changed by spinning the mouse wheel directly over the numbers on the dialog. This is dual action again, spinning the wheel over the left half of the number makes big changes, right half means small changes.

Use the "Layouts" feature to save your arrangement. Same as with the Desktop and Web FIPs, loading a layout will automatically associate it with the currently loaded aircraft.

WebPages can be displayed either in IE11 or Chrome compatibility mode. IE11 is easier on PC resources, Chrome has better compatibility.

13. Disable simulator controllers

The simulator has a “database” of many controllers. When you plug in a new joystick or similar device, the simulator will try to match control assignments to this device based on the database. The result is saved to the file “Standard.xml” in the simulators configuration directory (AppData\Roaming\...). This is the same for all simulators from FSX boxed to P3D V4.x.

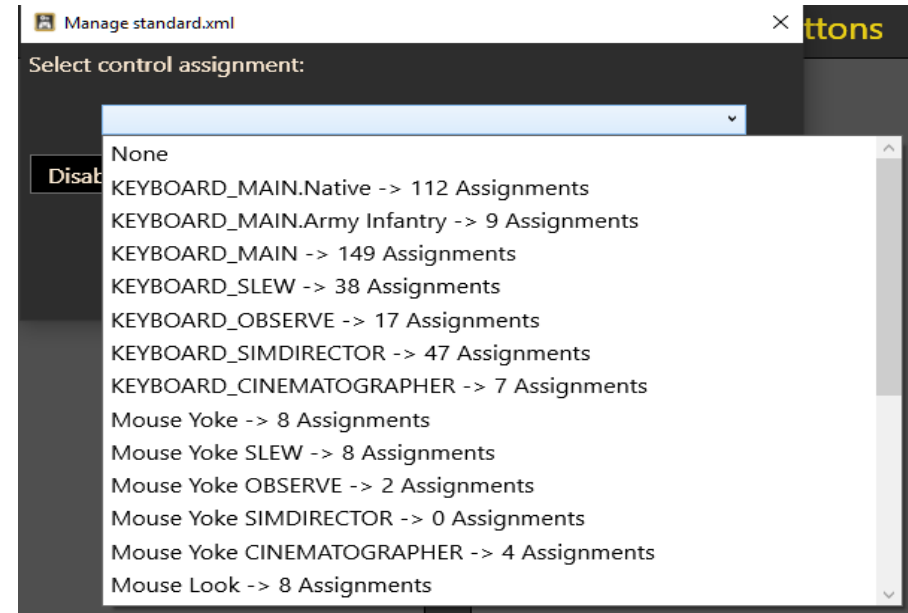
These assignments will get into conflict with what Lorby AxisAndOhs is trying to do, so they must be disabled when they are triggering the same controls. There are several ways to do that:

- You can disable the conflicting axis or buttons in the Control assignment dialog of the simulator.
- You can disable all external controllers in the simulator (there is a checkbox for that in the control assignments dialogs of the sim). This has the drawback of disabling all controllers, even the mouse (so mouse-look will no longer work)

Alternatively you can use the Lorby AxisAndOhs “Disable simulator controllers” feature:

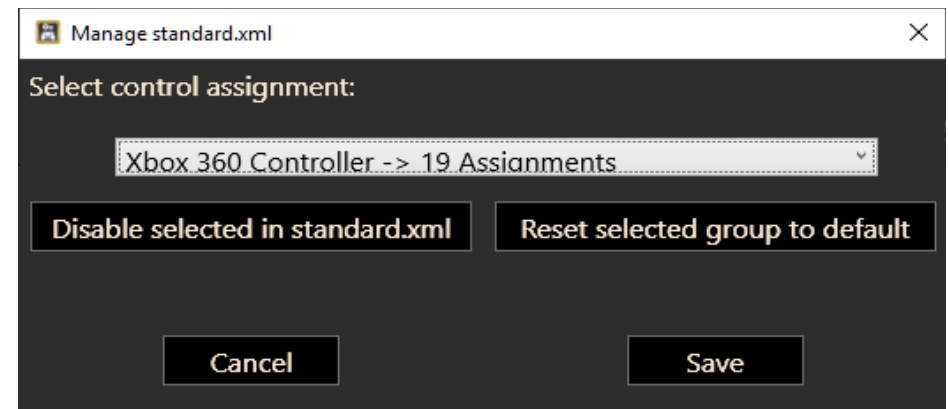
This action is only possible when your simulator is not running!

Opening the combobox will show you all controller assignments that the simulator is currently using.



After having selected one of them you can choose to

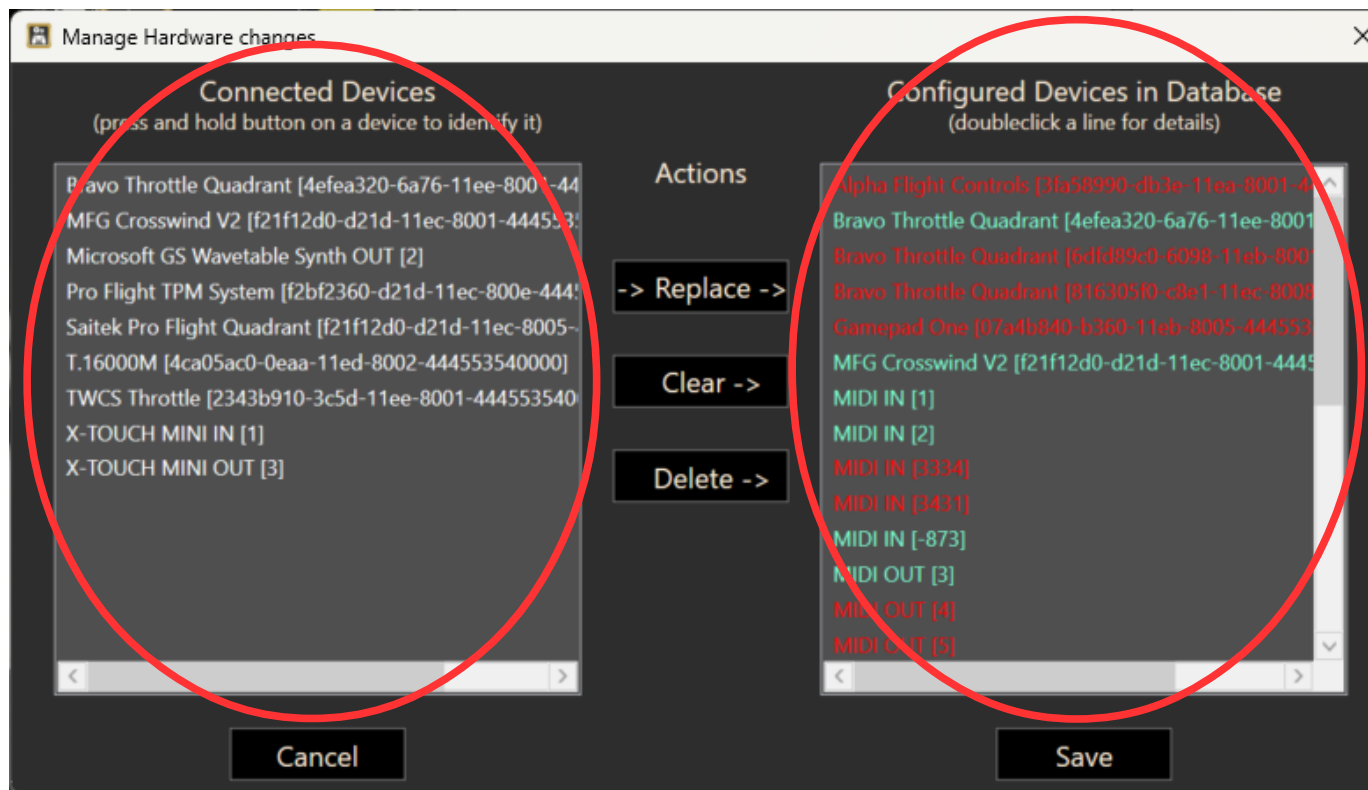
- Disable selected in standard.xml:
this will remove all assignments to this controller from the standard.xml file, but keep the definition itself intact – so the sim will not restore the assignments on its own
- Reset selected group to default:
this will remove all references to the selected controller from the standard.xml. When you next start and stop the simulator it will treat the controller as if it was a new one and restore the default settings.



14. Hardware change

If you make changes to your controller setup, for example replace a joystick with a new one, replace an USB hub etc., theoretically you would have to reassign all your controls. To help with that, Lorby AxisAndOhs has a dialog that allows you to manually reassign or delete controllers from the AxisAndOhs database. Go to „**Hardware → Hardware Change**“ to access it.

The list on the left shows all joystick devices that are currently connected to your computer



The list on the right shows all joystick devices that are present in the AxisAndOhs database.

Devices that match with a connected joystick are shown in **green**, devices that are no longer present are shown in **red**

Doubleclick on a line to get details about where the controller is assigned exactly

-> Replace ->

With this button you can replace a joystick device in the database with a different (connected) one. The typical use case would be that you have replaced a joystick or if Windows has decided to rearrange the USB devices representing the joysticks (and your assignments no longer match the controller they were made for).

- Select the new device in the left list
- Select the device that is to be replaced in the right list
- Press “- > Replace - >”
- After a safety dialog, the app will replace the old joystick with the new one.
- Press “Save” if you are happy with the changes.

Clear ->

With this button you can set a certain joystick device to represent an “empty” joystick. This is useful if you have to make a replacement but already have the new device in the database, or if you want to disable a certain controller.

- Select the device that is to be cleared in the right list
- Press “Clear - >”
- After a safety dialog, the app will replace the old joystick with an empty device.
- Press “Save” if you are happy with the changes.

Delete ->

This will completely remove the selected entry from the right list. Use this if you have assignments in the database that are obsolete or that you are not using any more.

- Select the device that is to be cleared in the right list
- Press “Delete - >”
- After a safety dialog, the app will delete the selected device from the database.
- Press “Save” if you are happy with the changes.

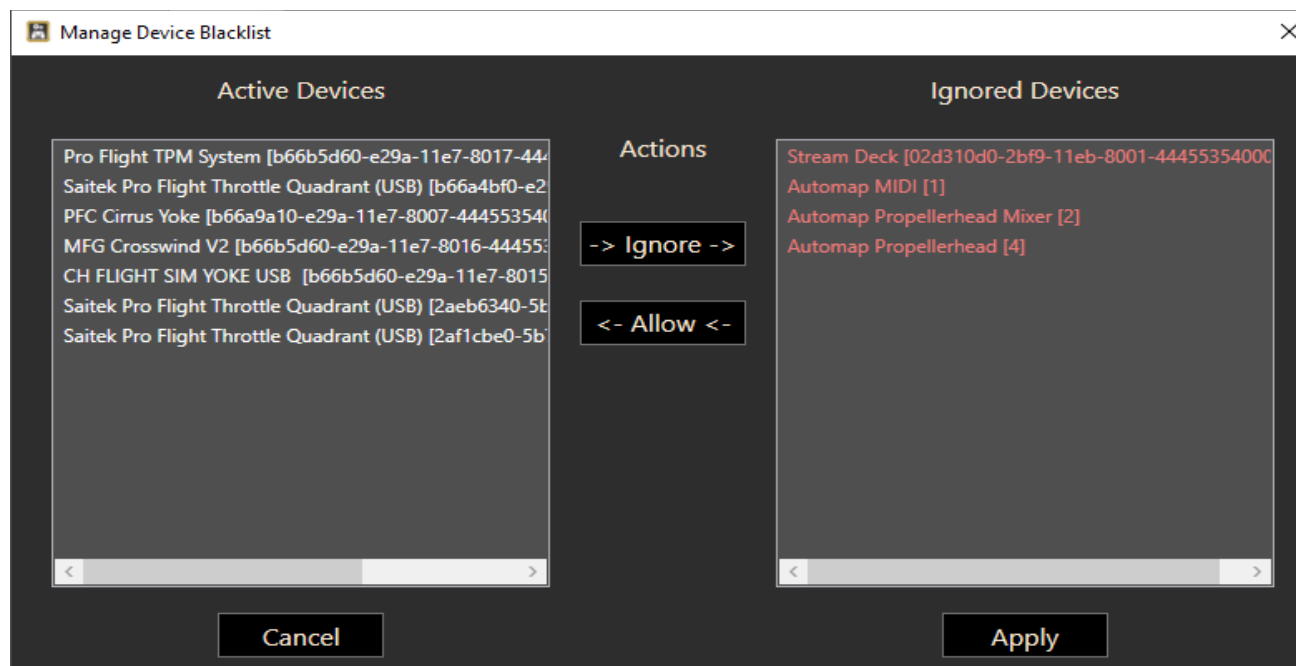
Disconnecting and reconnecting joysticks

A device that is disconnected while the app is running is being ignored. When you reconnect it, you must direct AxisAndOhs to scan for joystick devices or, better still, shut down and restart the app.

To scan for devices you can use the “Rescan Devices” option in the "Hardware" menu.

15. Device Blacklist

If you have other, non-controller USB devices that are flooding AxisAndOhs with unwanted input signals, you can direct the app to ignore them with „**Hardware → Device Blacklist**“.



16. Config files, database and log files

Database

Lorby AxisAndOhs is using a simple XML file to store all your joystick assignments. The file is named differently for each simulator that you installed the app for, and it is located here:

“C:\Users\...\AppData\Local\LORBY_SI\LorbyAxisAndOhs\ConfigDatabase_(sim-ID).xml”

The file is saved in multiple “generations”. If something goes wrong, you can return to an earlier state of the files by renaming the n-th backup “ConfigDatabase_(sim-ID)_n.xml” to “ConfigDatabase_(sim-ID).xml”

Startup log

This file lists all events that occurred during startup of the application. It is used for debugging purposes, should the app not start at all.

“C:\Users\...\AppData\Local\LORBY_SI\LorbyAxisAndOhs\LAAO_Startup_log.txt”

Other logfiles

Error in the app will be written to logfiles in this location. If you have duplicate aircraft definitions in the simulator, for example after an Alpha update, they will also be listed in a logfile here.

App configuration

Setting concerning the app itself are saved in this file

“C:\Users\...\AppData\Local\LORBY_SI\LorbyAxisAndOhs\LorbyAxisAndOhsConfiguration.xml”

User content

Normally, all user content is stored in \Documents\LorbyAxisAndOhs Files. When this folder is not available, for example on the OneDrive, then AAO switches to "C:\Users\...\AppData\Local\LORBY_SI\LorbyAxisAndOhs" instead. You must then place gauges etc. there.

Standard.xml

If you decide to disable components of the Standard.xml file (see chapter 4), AxisAndOhs will keep the original file and an additional backup. You will find both these backups at the same location where your “Standard.xml” is.

Examples:

“C:\Users\...\AppData\Roaming\Microsoft\FSX\Controls”

“C:\Users\...\AppData\Roaming\Lockheed Martin\Prepar3D v4\Controls”

17. PMDG Aircraft with AxisAndOhs

AxisAndOhs provides the Events of the PMDG SDK if you want to assign PMDG internal functionality to buttons or MIDI controllers. They are listed in separate groups on the Event assignment dialog.

Please make sure to check the PMDG SDK documentation. It should be present in the aircraft folder of the PMDG main installation (for example here : „<sim>\PMDG\PMDG 777X\SDK“)

Value increments

A good strategy for value increments is to use RPN scripts instead of single button presses.

```
(L:PMDG_MCP_ALT) · 100 · + · (>K:#84137) · (L:PMDG_MCP_ALT) · 100 · + · (>L:PMDG_MCP_ALT)
```

This script uses a local variable to store the current AP altitude, and upon a button press, it increases it by 100 feet – and sends the value to the PMDG variable. The default value of the PMDG variable is 10000ft, so this script would benefit from an Automated Script that is run once when the aircraft is loaded:

```
10000 (>L:PMDG_MCP_ALT)
```

Toggles

To make the PMDG events work like toggles instead of on/off switches, you can use scripts too. For example, this script toggles HDG_SEL in the PMDG 737 NGX

```
1 (L:PMDG_MCP_HDG_SEL) - (>K:#70024) 1 (L:PMDG_MCP_HDG_SEL) - (>L:PMDG_MCP_HDG_SEL)
```

Mouse events

It is possible to send specific mouse actions to the controls in a PMDG cockpit. The Mouse events can be found in the „PMDG Mouse“ group, they serve as input parameters for the >K events.

Event	Value to send
MOUSE_RIGHTSINGLE	2147483648
MOUSE_MIDDLESINGLE	1073741824
MOUSE_LEFTSINGLE	536870912
MOUSE_RIGHTDOUBLE	268435456
MOUSE_MIDDLEDDOUBLE	134217728
MOUSE_LEFTDOUBLE	67108864
MOUSE_RIGHTDRAG	33554432
MOUSE_MIDDLEDRA	16777216
MOUSE_LEFTDRAG	8388608
MOUSE_MOVE	4194304
MOUSE_DOWN_REPEAT	2097152
MOUSE_RIGHTRELEASE	524288
MOUSE_MIDDLEREL	262144
MOUSE_LEFTRELEASE	131072
MOUSE_WHEEL_FLIP	65536
MOUSE_WHEEL_SKIP	32768
MOUSE_WHEEL_UP	16384
MOUSE_WHEEL_DOWN	8192

Examples:

```
16384 (>K:#70016)
8192 (>K:#70016)
```

These scripts send “Mouse_Wheel_Up” (16384) and “Mouse_Wheel_Down” (8192) commands to the EVT_MCP_SPEED_SELECTOR (#70016) (PMDG 737 NGX)

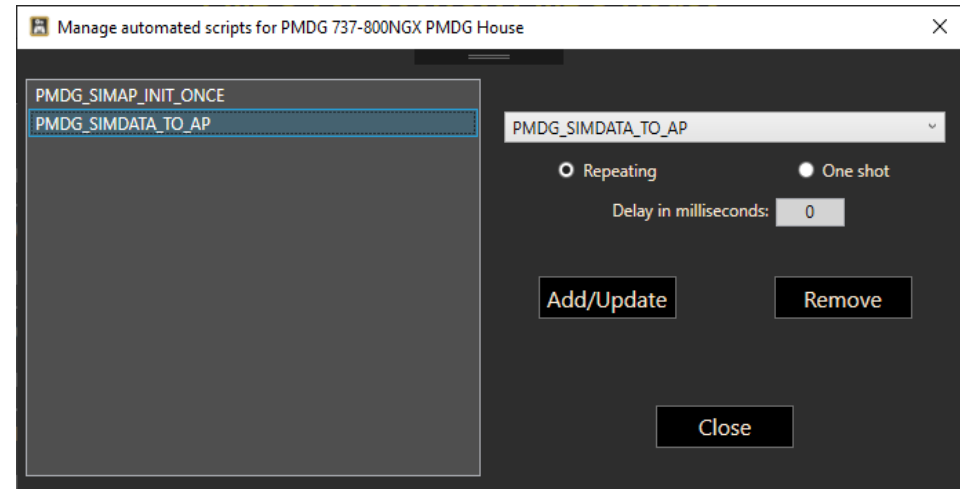
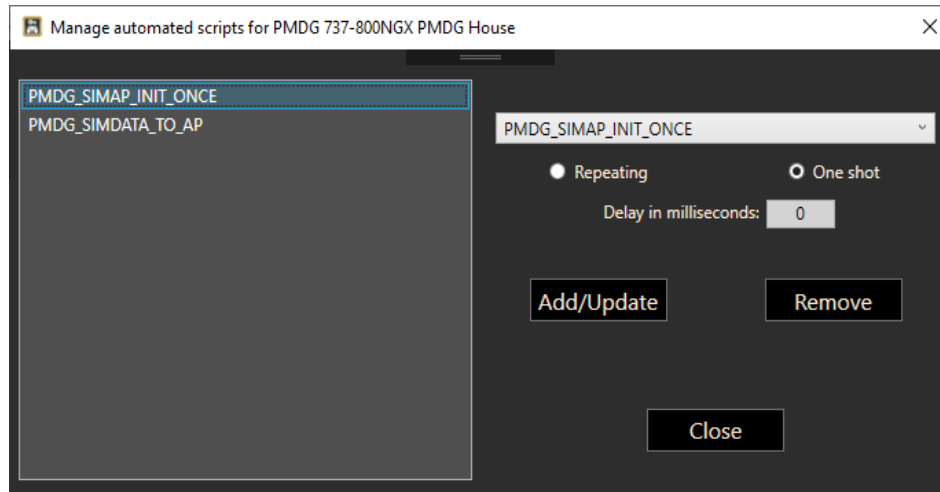
Simulator sync

Furthermore, there are two RPN scripts that you can use with any PMDG plane that will synchronize the PMDG autopilot variables with their default counterparts from within the simulator. Using these scripts you can utilize the default simulator events for example to change the autopilot heading or altitude.

Please note that using these scripts can cause the „input acceleration“ to trigger, meaning that the changes you make will suddenly be larger than you expect (as if you would turn a control really quickly). Try not to use the mouse in the cockpit on a control that is bound by this script (=the MCP rotary encoders)

How to set up the RPN scripts for AP synchronization:

1. Start the sim and AAO, then load your PMDG aircraft. Wait until AxisAndOhs has recognized the plane and has finished loading the config
2. In AAO go to „Scripting->Automated scripts“
3. Select „PMDG_SIMAP_INIT_ONCE“ from the dropdown to the right
4. Select „One shot“
5. Press „Add/Update“
6. Select „PDMG_SIMDATA_TO_AP“
7. Select „Repeating“
8. Press „Add/Update“
9. Close the dialog



From now on, when you load your aircraft, it will execute the „...INIT_ONCE“ script, and it will continuously synchronize the simulator AP variables with their PMDG counterparts.

So when you want to adjust the AP vertical speed, you can use the default simulator events „AP_VS_VAR_INC“ and „AP_VS_VAR_DEC“. Assigning buttons to the PMDG events is not required!

Using these scripts, the PMDG plane will also interact correctly with the Saitek MultiPanel – without any further assignments or changes.

Reading PMDG variables

AAO can access the data structures that are provided in the PMDG SDK, so you can read those values from the simulator too.

For that to work, you have to activate data transmission from the PMDG plane as described in the SDK PDF from PMDG. This requires a restart of the simulator. Quote from the SDK:

To enable the data communication output, you will need to open the file(...) _Options.ini that is located in the folder \PMDG\PMDG (...)

Once this folder is open, add the following lines to the bottom of the file:

[SDK]

EnableDataBroadcast=1

EnableCDUBroadcast.0=1

EnableCDUBroadcast.1=1

After enabling the broadcast you have to assign the appropriate automated script to your PMDG aircraft in AAO. You will find them in the group “PMDG”, and they must be run “Repeating” at a reasonable refresh rate (200ms)

“PMDG_NG3_Data”

“PMDG_NGX_Data”

“PMDG_777X_Data”

“PMDG_747QOTSII_Data”

When all this is in place, you can read the variables from the aircraft as follows:

(PMDG:LTS_LandingLtRetractableSw[0])

All available variables can be found on the “Insert Variable” dialog of the RPN script editor in their designated groups.

Be mindful of array-type variables like the one above. If designated “Array[]” there is more than one value to be accessed here (0 - x). Please refer to the PMDG SDK specification for details – you usually find it in the configuration folder of the aircraft in the main sim folder.

Example:

C:\Program Files\Lockheed Martin\Prepar3D v4\PMDG\PMDG 747 QOTS II\SDK

Note: you can't write to these variables. Use the PMDG events described above for that.

18. MIDI Out

Many MIDI controllers have LEDs to give you feedback upon key presses. These LEDs can be used to show the state of a variable in the simulator too. For that, MIDI controllers usually listen for MIDI Out events coming their way, on a specific channel, with a specific note at a specific velocity.

For example, the Berhinger X-Touch Mini listens on Channel 1, the note designates the button position (0 is top row left button, 15 is bottom row right button), velocity 1 turns the LED behind the button on, 0 turns it off.

A Novation Launchpad expects a note value that is calculated as „(16 * row) + column“. Velocity is binary encoded and quite complicated, but for example 12 is off and 62 is full intensity yellow. Look for the Launchpad-Programmers-Guide.pdf online for details.

To send MIDI Out events with AxisAndOhs, you have to embed them into an RPN script in the following syntax:

velocity (>MIDI:<DeviceId>:<Action>:<channel>:<note>)

<Action> can be one of the following: „NoteOn“, „NoteOff“, „CC“, „PrgChng“

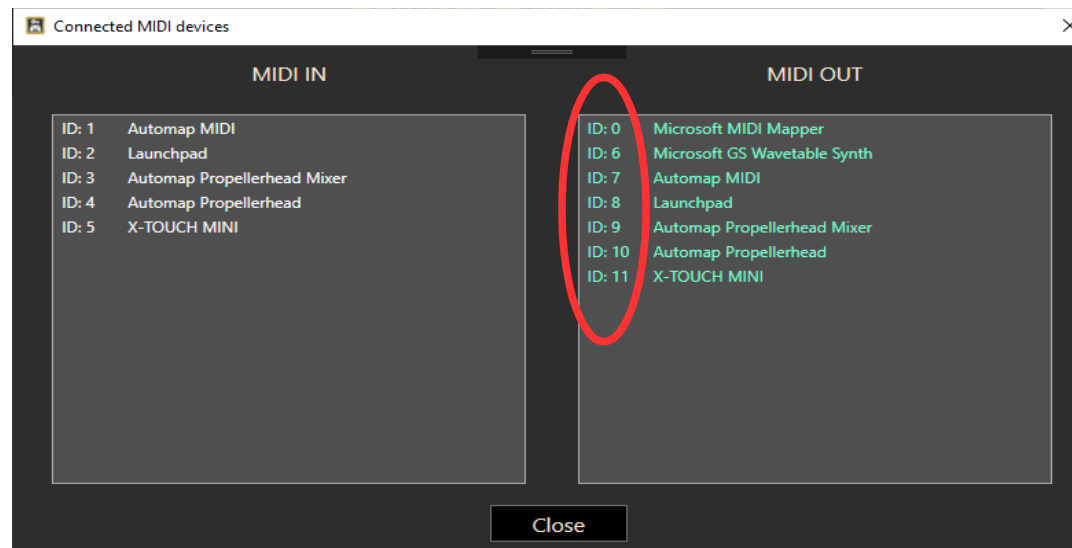
Example: „62 (>MIDI:8:NoteOn:1:17)“ turns the second button in the second row on a Launchpad yellow.

„Program change“ doesn't have a <note> : 1 (>MIDI:8:PrgChng:1)

You can also send SysEx messages: **(MIDIOUT:<DeviceId>:SysEx:[string of 2 digit hex values])**

(MIDIOUT:8:SysEx:010203AC) results in F0 01 02 03 AC F7. Do not use values above F0 in the message.

The **DeviceId** can be found on the dialog showing the Connected MIDI devices. (Top Menu → Hardware → Show MIDI devices)



Example: toggle the Autopilot with the first top row button on the Behringer X-Touch

Create a new RPN Script „MIDI_SYNC_AP_ON“ (or whatever you want to call it) with this content:

```
(A:AUTOPILOT MASTER, Bool) 0 == if{ 0 (>MIDI:11:NoteOn:1:0) } els{ 1 (>MIDI:11:NoteOn:1:0) }
```

Then assign it to your aircraft as an automated script, running every 200 – 400ms (no need to run it more often than that). From now on, whichever way you toggle the Autopilot, the LED of the first button on the Behringer will light up accordingly. If you now assign the AP_MASTER event to the same button in AAO, you have an AP toggle button with visual feedback.

19. Web API

AxisAndOhs offers an API based on web technology to access the simulator with any application that can send and receive HTTPRequests. The API is basically a miniature webserver. Using a simple JSON structure or special request parameters you can trigger events, read and write simulator variables and execute scripts.

For the Web API to work, AxisAndOhs must be run „As Administrator“!

19.1. URL

The API can be accessed on either of two ports. The base URLs for the API are:

http://<Local-IP-address>:43380/webapi

http://<Local-IP-address>:43381/webapi

<Local-IP-address> is either „localhost“ for connections on the same computer, or the local IP address.

Both ports are functionally identical. You can use either one, but it is advisable to distribute load between the two if you have multiple programs or addons accessing the API. The secondary port is always the primary + 1.

Be mindful of firewall restrictions. You may have to open the TCP ports in your local or router firewall.

19.2. Changing the ports

If you have other applications on your local network that use port 43380/43381, you will have to change it. The port change must always happen on both ends, in AAO and in the app/webpage that wants to talk to it.

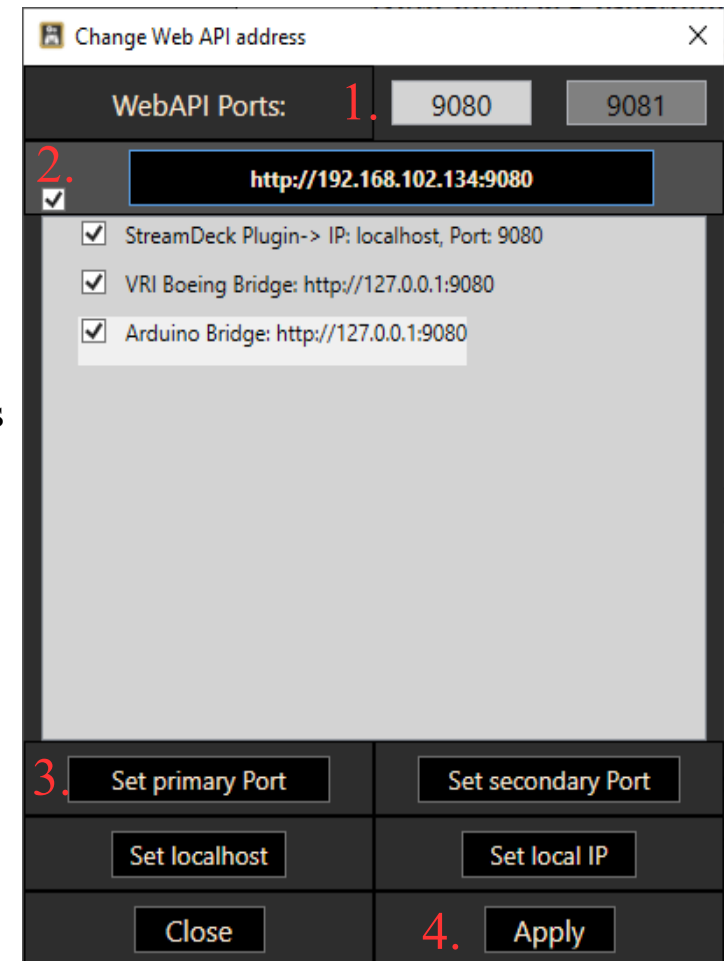
For AAO the port can be changed with „Tools → **Configure WebAPI Port**“. AxisAndOhs must be restarted when you change the port.

With this dialog you can set the port using your **mouse wheel**.

The list shows all AAO addons found on your computer that use this port. You can adjust the address in those addons with the buttons below the list:

- Set primary/secondary port: apply this port to the selected addons
- Set localhost/local IP: change the URL in the addons to localhost or your local IP address.

Please note that the addons can not be running at this point. If they are, they will be disabled in the list. You will have to terminate the addon program to make the change.



19.3. Sending a simulated button event

You can send controller button events using request parameters:

<http://localhost:43380/webapi?dev=1&chn=5&btn=4>

dev, chn and btn are arbitrary numerical values that you can set up according to your use case. These events will show up on the Add/Change button dialogs as WebAPI input.

- **dev:** is the ID of the „device“ that the event is sent from. This should be unique per web app/page that is communicating with AAO
- **chn:** can be used to make the same control (btn) do different things. With this you can establish several layers of actions on the same control
- **btn:** is the control itself.
- **bval: optional:** button value, 127 for button pressed, 0 for button released. You have to send both, otherwise the button will be stuck.

19.4. Sending a simulated axis event

You can send axis events using request parameters:

<http://localhost:43380/webapi?dev=1&chn=1&axis=4&aval=16234&dmin=0&dmax=65535>

dev, chn and axis are arbitrary numerical values that you can set up according to your use case. These events will show up on the Add/Change Axis dialogs as WebAPI input

- **dev:** is the ID of the „device“ that the event is sent from. This should be unique per web app/page that is communicating with AAO
- **chn:** can be used to make the same control (btn) do different things. With this you can establish several layers of actions on the same control
- **axis:** identifies the individual axis.
- **value:** current axis position between dmin and dmax.
- **dmin:** minimum value for the axis, for a joystick this would be 0.
- **dmax:** maximum value for the axis, for a joystick this would be 65535.

19.5. JSON interface

The API can also process a JSON data structure

```
{
  "buttons": [
    {
      "dev": <numerical device id>,
      "chn": <numerical channel id>,
      "btn": <numerical button id>,
      "bval": <optional: 127 for button pressed, 0 for button released>
    }, ...
  ],
  "axis": [
    {
      "dev": <numerical device id>,
      "chn": <numerical channel id>,
      "axis": <numerical axis id>,
      "value": <numerical value>,
      "devMin": <minimum numerical value for this axis>,
      "devMax": <maximum numerical value for this axis>,
    }, ...
  ],
  "triggers": [
    {
      "evt": "<simulator event-id>",
      "value": <numerical value to set>
    }, ...
  ],
  "getvars": [
    {
      "var": "<simulator variable get definition>",
      "value": 0.0
    }, ...
  ],
  "getstringvars": [
    {
      "var": "<simulator variable get definition>",
      "value": "<string>"
    }, ...
  ],
  "setvars": [
    {
```

```

    "var": "<simulator variable set definition>",
    "value": <numerical value to set>
  }, ...
],
"setstringvars": [
  {
    "var": "<variable name, this can only be used with AAO internal LVars of type , String>",
    "value": <string value to set>
  }, ...
],
"pullllvars": [
  {
    "var": "<simulator variable get definition>",
    "value": 0.0
  }, ...
],
"pullstringlvars": [
  {
    "var": "<simulator variable get definition>",
    "value": ""
  }, ...
],
"scripts": [
  {
    "code": "<RPN script code>"
  }, ...
],
"files": [
  {
    "fname": "<file path on the AAO webserver>",
    "content": "<file content as string>", [content and base64 are mutually exclusive, you can use only one of them]
    "base64": "<binary file content as Base64 encoded string>",
    "isvirtual": "true" , [optional, file is retained in memory on the server, not saved to disk]
  }, ...
]
}

```

"pull" requests return the value of the variable and reset it to 0/empty string at the same time. They are only applicable to AAO internal LVars, like (L:myvar) and (L:mystrvar, String)

With Javascript XMLHttpRequests you would send the data structure like this:

```
var requestObj = {};  
requestObj.buttons = [];  
var tosend = { "dev": 1, "chn": 1, „btn":1 };  
requestObj.buttons.push(tosend);  
tosend = { "dev": 1, "chn": 1, „btn":14 };  
requestObj.buttons.push(tosend);  
requestObj.triggers = [];  
tosend = {"evt":(">K:LANDING_LIGHTS_TOGGLE)","value":1.0};  
requestObj.triggers.push(tosend);  
requestObj.getvars = [];  
tosend = {"var": "(A:AUTOPILOT ALTITUDE LOCK VAR, feet)","value":0.0};  
requestObj.getvars.push(tosend);  
tosend = {"var": "(A:AUTOPILOT VERTICAL HOLD VAR, feet/minute)","value":0.0};  
requestObj.getvars.push(tosend);  
requestObj.setvars = [];  
tosend = {"var":(">A:NAV1 OBS, degrees)","value":135.0};  
requestObj.setvars.push(tosend);  
(...)  
  
var xhttp = new XMLHttpRequest();  
xhttp.addEventListener("load", requestListener);  
xhttp.open("GET", "http://localhost:43380/webapi?json=" + JSON.stringify(requestObj));  
xhttp.send();
```

For larger amounts of data consider using a POST request:

```
var xhttp = new XMLHttpRequest();  
xhttp.addEventListener("load", requestListener);  
xhttp.open("POST", "http://localhost:43380/webapi/", true);  
xhttp.send(JSON.stringify(requestObj));
```

The **response** is the same structure, with the „getvars“ fields containing the requested values

```
function requestListener() {  
    var commObj = JSON.parse(this.responseText);  
    var apAlt = commObj.getvars[0].value;  
    var apVs = commObj.getvars[1].value;  
}
```

You can also request values that have been calculated by an RPN script, using

- „(L:scriptgroup-scriptname)“ for numerical values, calculated by a script like this one:
„(A:INDICATED·ALTITUDE,·Feet)·10000·+“ (**getvars**)
- „(L:scriptgroup-scriptname, String)“ for string values that are produced by a script like this one.
„%(A:INDICATED·ALTITUDE,·Feet)·10000·+%!5d!“ (**getstringvars**)

and for an external script that you don't want to reside in AAO using the „S:“ header

- „(S:(A:INDICATED·ALTITUDE,·Feet)·10000·+)“ (**getvars**)
- „(S:%(A:INDICATED·ALTITUDE,·Feet)·10000·+%!5d!)“ (**getStringvars**)

The example in this chapter does the following:

- it sends two button clicks, 1 and 14
- it toggles the landing lights, then it sets the simulator into Pause mode
- it requests Autopilot selected altitude, vertical speed and heading variables
- finally it sets the NAV 1 OBS marker to 135 degrees

```
{
  "buttons": [
    {
      "dev": 1,
      "chn": 1,
      "btn": 1
    },
    {
      "dev": 1,
      "chn": 1,
      "btn": 14
    }
  ],
  "triggers": [
    {
      "evt": "(>K:PAUSE_ON) ",
      "value": 1.0
    },
    {
      "evt": "(>K:LANDING_LIGHTS_TOGGLE) ",
      "value": 1.0
    }
  ],
  "getvars": [
    {
      "var": "(A:AUTOPILOT ALTITUDE LOCK VAR, feet)",
```

```
    "value": 0.0
  },
  {
    "var": "(A:AUTOPILOT VERTICAL HOLD VAR, feet/minute)",
    "value": 0.0
  },
  {
    "var": "(A:AUTOPILOT HEADING LOCK DIR, degrees)",
    "value": 0.0
  }
],
"setvars": [
  {
    "var": "(>A:NAV1 OBS, degrees)",
    "value": 135.0
  }
],
"setstringvars": [
  {
    "var": "(>L:AAO_Internal_State, String)",
    "value": "ready"
  }
]
}
```

20. Using the WebAPI as a web server

The WebAPI can also be used as a simple webserver to host HTML pages, for example a button panel or flight instruments.

1. create a subfolder of your choice in \Documents\LorbyAxisAndOhs Files\WebPages
Example: \Documents\LorbyAxisAndOhs Files\WebPages\mybuttonwebpage
2. then copy your HTML, CSS, JS files and pictures into that subfolder
Example: \Documents\LorbyAxisAndOhs Files\WebPages\mybuttonwebpage\index.html
 \Documents\LorbyAxisAndOhs Files\WebPages\mybuttonwebpage\buttonlayout.css
 \Documents\LorbyAxisAndOhs Files\WebPages\mybuttonwebpage\background.png
3. After you have started AAO, you can access your page from any browser like this:
<http://localhost:43380/webapi/mybuttonwebpage/index.html>
(for remote access, localhost needs to be replaced with the actual IP address of the AAO computer)

There is a special code that you can use to make AAO inject the actual API address into your html or javascript: **<AAO_URL>**. This would typically be used when your web page has active elements that call back to AAO.

```
...
<body>
<script>
    var AAO_URL = "<AAO_URL>";
    var deviceId = 235467;
...
    xhttp.open("GET", AAO_URL + "?json=" + JSON.stringify(mainLoopRequestObj));
```

That way you don't have to concern yourself with the actual address in your code.

To make accessing the API easier in code, you can link to a special AAO JavaScript library:

```
<!DOCTYPE html>
<html>
<head>
  <script src="AaoWebApi.js"></script>
```

then, in your code, you create a new API object:

```
<body style="background: black">
...
<script>
  var webApi = new AaoWebApi();
...
  webApi.StartAPI(75);
```

The last command starts the data collection service of the API, in this case new data is fetched every 75 ms.

The webApi object has the following methods that you can use:

- StartAPI(frequency): start the data collection every frequency milliseconds
- StopAPI(): stop the data collection
- GetSimVar(varname); get the value of a specific simulator variable
`var val = webApi.GetSimVar("(L:FMC_EXEC_ACTIVE, Number)");`
- PullSimVar(varname); get the value and reset it to 0/empty
`var val = webApi.PullSimVar("(L:b747fmc/fmc.html_complete)");`
- ClearSimVar(varname): remove a variable from the data collection
- Reset(): clear all variables

- SetSimVar(value, varname): set a variable to the specified value
`webApi.SetVariable(1, '(>L:AAO_FMC_HTML_INIT, Number)');`
- SendEvent(value, event): trigger an event in AAO
`webApi.SendEvent(0, "(>H:B747_8_FMC_1_BTN_PLUSMINUS, Number)");`
- SendButton(deviceId, channelId, buttonId) send a button event
`webApi.SendButton(365, 10, 2);`
- SendScript(code): send RPN code to AAO for processing
`webApi.SendScript("1 (L:Statevar) - (>L:Statevar)");`

The WebAPI can also be used to create files on the webserver.

The structure to transport a file in the JSON request is

```
{  
  "fname": "/testfolder/testfile.txt",  
  "content": "this it the text content of the file",  
  "base64": "this it the binary content of the file"  
  "isvirtual": "true"  
}
```

You can only use either <content> or <base64> but not both. Every structure that has a <content> will be written as a text file, every file that has a <base64> will be written to a binary file. „isvirtual“ is optional, when present and set to „true“, the file will not be saved to disk on the server, but retained only in memory.

```
var requestObj = {};  
requestObj.files = [];  
var tosend = { "fname": "/testfolder/testfile.txt", "content": "this it the content of the file" };  
requestObj.files.push(tosend);  
var tosend = { "fname": "/testfolder/testimg.jpg", "base64": "iVBORw0KGgoAAAANSUhE...." };  
requestObj.files.push(tosend);  
(...)  
  
var xhttp = new XMLHttpRequest();  
xhttp.open("POST", "http://localhost:43380/webapi?json=" + JSON.stringify(requestObj));  
xhttp.send();
```

This would typically be used to send external html files or parts thereof to a website on the AAO server to be injected at runtime. This also works when you send data from within the simulator, using the same XMLHttpRequest logic as shown above.

Calling web pages

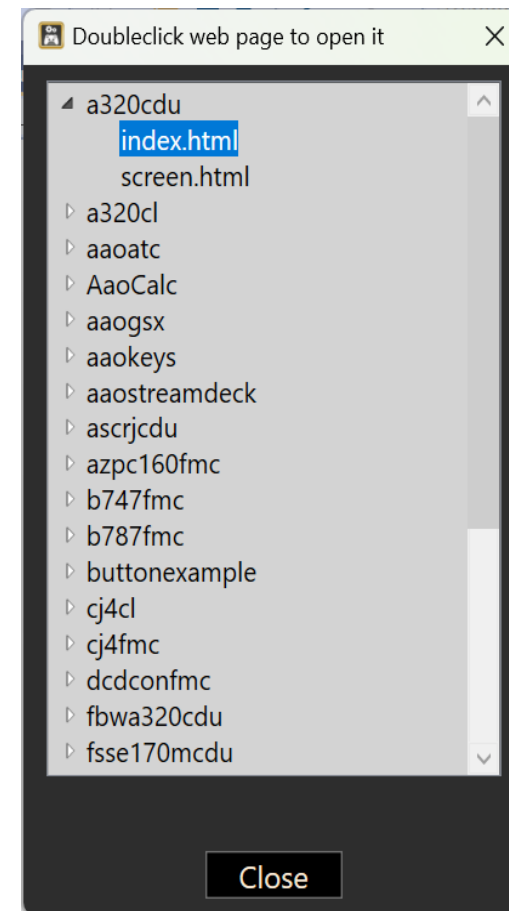
You can call the configured web pages and get their URLs with the dialog in "Gauges->Display WebPages" as soon as AxisAndOhs is connected to the simulator or running in Offline Mode

- *doubleclick* on a html page to open it in your default browser
- *left-click then right click* a html page to copy the URL to that page to the Windows clipboard. You can then paste it into your browser address bar with Ctrl & V.

Ready-made examples of AAO web pages can be downloaded from

<https://www.axisandohs.com/downloads.html>

These include several FMC/CDU instruments for various aircraft, a simple button page, a simple moving map and more.



21. Importing Event and Variable lists

With „Scripting → Import Events and Variables“ you can import csv-type text files with lists of events and variables into a custom group. „Scripting → Edit custom Events and Variables“ can be used to delete them again.

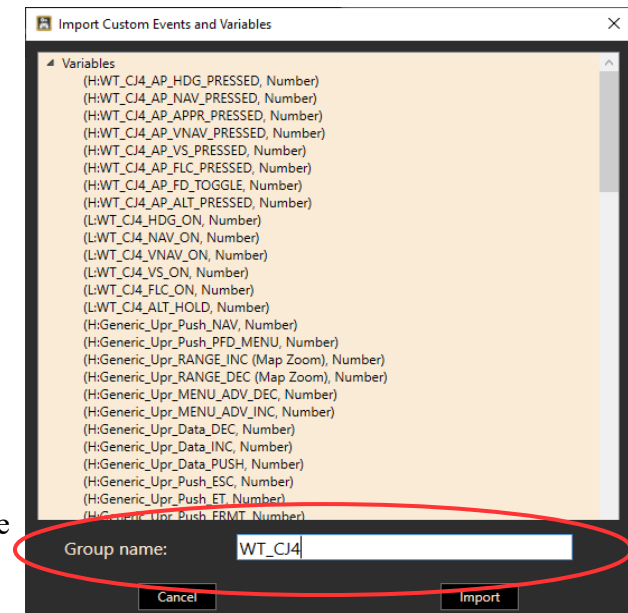
The file format is plain „comma separated values“ that can be created with any text file editor.

- for an event it is
`K,MyCustomEventID`
(only K – events are allowed and there can be no spaces in the event-ID)
- for a variable it is
`A,My Custom Variable Name,VariableUnit`
only A, L, H, E, P and C – variables are allowed, [, VariableUnit] is optional

Example of an import file (myfile.csv):

```
K,My_Own_Kevent_1
K,My_Own_Kevent_2
K,My_Own_Kevent_3
A,My Own AVar
A,Undocumented Headingvar,Degrees
L,MyOwnLVar,Number
H,MyOwnHVar,Number
H,AnotherHVar
```

Enter a group name
before importing



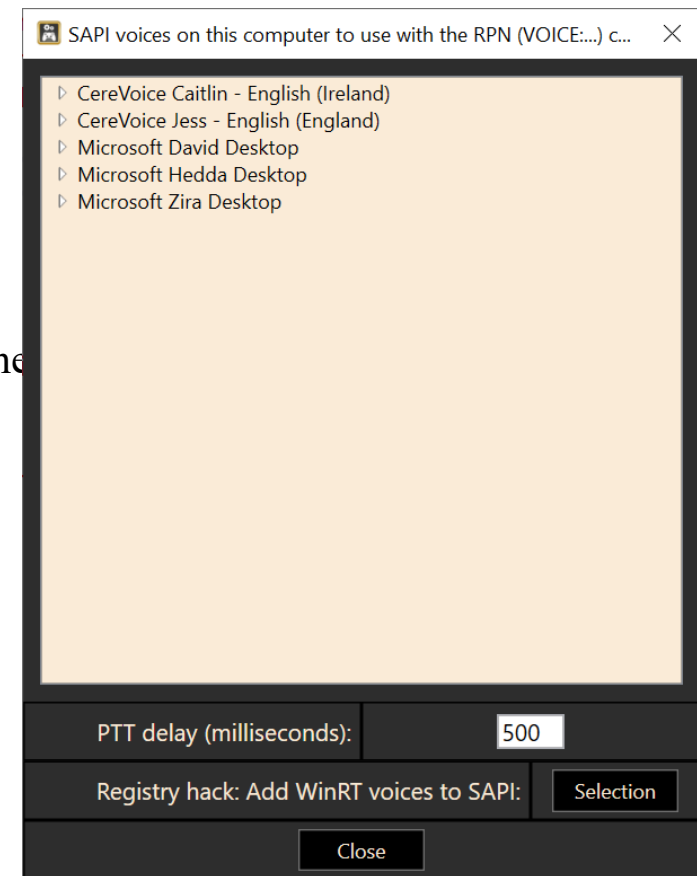
22. TextToSpeech: WinRT vs. SAPI

When using the (SPEAK: and (VOICE: commands, you will notice that on Windows 10 not all voices that you have available for Text to Speech are also visible in the AAO dialog „Extras ->Show list of all SAPI voices“.

The reason is, that Windows actually has two voice systems, „WinRT“ and „SAPI“. SAPI is the older, more well-known method. Voices that you can buy online are mostly in SAPI format.

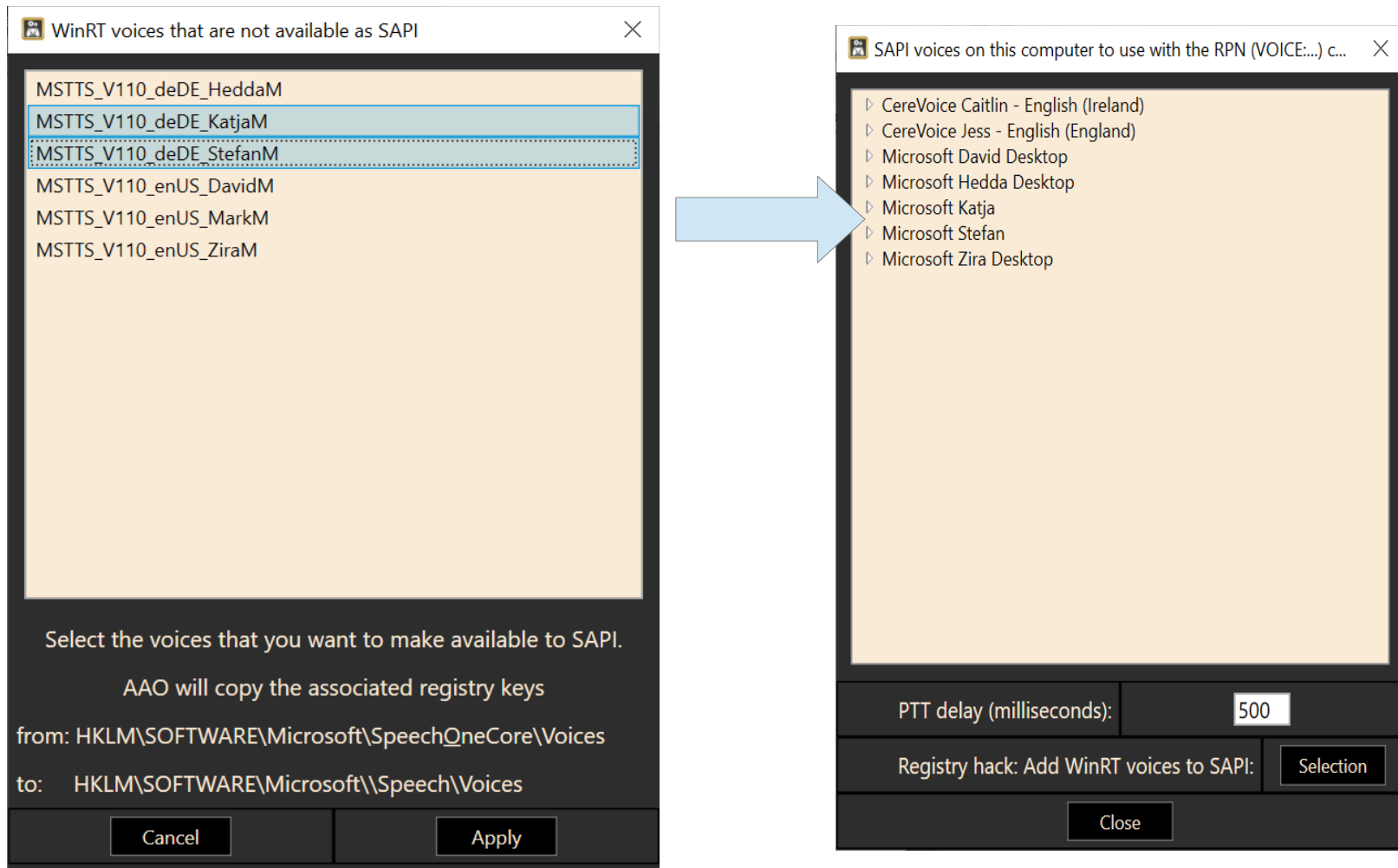
It is possible to use the WinRT voices as SAPI too, but for that, a bunch of registry keys has to be duplicated to another location.

AAO can do that for you, with the button „Selection“ at the bottom of the dialog. This button will only be active when AAO has been started „As Administrator“, and if there actually are additional WinRT voices that you can use.



When the button is pressed, you get a list of the available WinRT voices. Select those that you want to use as SAPI, then press „Execute“.

After the operation is complete, you have to restart your computer (!), otherwise the changes to the registry are not active.



23. Advanced TextToSpeech: Azure, Polly, ChatGPT

If you want to go beyond the capabilities of Windows SAPI, you can use the services of Microsoft Cognitive Speech („Azure“) and Amazon AWS Polly. Simply copy&paste a voice name from either of the services into the (VOICE: command in your script.

Furthermore, AxisAndOhs has an interface to the ChatGPT API that you can use to generate text.

Be mindful that these services all require that you get an API key from the provider, and that some are not free of charge.

Your API keys must be entered using the dialogs in the „Extras“ menu.

Microsoft Cognitive Speech („Azure“)

Azure can be used for speech output. It offers a great number of different voices for many localizations.

[Create Your Azure Free Account Today | Microsoft Azure](#)

Once you have created your account, add the „azure-speech-api“ as a Resource, then create the API key for it. The Azure service is currently free as long as your usages stays under the pre-defined limit of 500.000 characters per month.

After you have entered your API keys in AxisAndOhs, you can access a list of the available voices with „**Extras->Show list of Microsoft Azure Voices**“. To copy the name of a voice, left click then right click it. This copies the name into the Windows Clipboard.

Insert the voice name into your script code like this:

(VOICE:Azure|en-US-AndrewNeural)

and the next SPEAK/SPEAKBLK command will be spoken by Azure.

Amazon AWS Polly

AWS Polly can be used for speech output. It doesn't have as many voices as Azure, but the processing is faster.

[AWS Console - Signup \(amazon.com\)](https://aws.amazon.com/polly/)

After you have created your account, use the IAM to create a new group and add „AmazonPollyFullAccess“ as permission policy. Then create a new user in that group. Now you can create an API key for that user. The AWS service is currently free for 12 months. Please consult the AWS pricing tables for continued use.

You can access a list of the available voices with „**Extras->Show list of Amazon Polly Voices**“. To copy the name of a voice, left click then right click it. This copies the name into the Windows Clipboard.

Insert the voice name into your script code like this:

(VOICE:Polly|en-US Danielle|neural)

and the next SPEAK/SPEAKBLK command will be spoken by AWS Polly.

Chat GPT

You can also use your ChatGPT API account (see separate chapter below) to make the AI speak your text. ChatGPT has a limited choice of voices and it is slow compared to the others. The ChatGPT API service is not free of charge.

You can access a list of the available voices with „**Extras->Show list of ChatGPT Voices**“. To copy the name of a voice, left click then right click it. This copies the name into the Windows Clipboard.

Insert the voice name into your script code like this:

(VOICE:GPT|alloy)

and the next SPEAK/SPEAKBLK command will be spoken by ChatGPT.

24. RPN script files

If your RPN scripts are getting too complex or if you wish to run several scripts with one single command, consider using script files.

A script file is a simple text file where every line is a RPN script. The scripts can be as simple or complex as you like, but you have to make sure that each script always only occupies a single line:

```
(A:PLANE HEADING DEGREES GYRO, Degrees) (>K:HEADING_BUG_SET)
(FOCUS:outlook) · (VKD:28-156-13) · (SPLIT:100) · (VKU:28-156-13) · (SPLIT:5000) · (FOCUS:prepar3d)
53623 · (>L:MJC_VAR_READ_CODE, ·Number) · (L:MJC_VAR_READ_VALUE, ·Number) · (>L:MJC_APU_PWR)
0 (>K:VIEW_REAR)
0 (>K:VIEW_REAR)
0 (>K:VIEW_REAR)
0 (>K:VIEW_REAR)
0 (>K:VIEW_REAR)
...
```

All script files are to be saved in \Documents\LorbyAxisAndOhs Files\Scripts (you can create subfolders)

Running script files

There are two ways to run a script file:

- either manually from the dialog in „Scripting → Run script files“
- or by calling them from within a script using the „(SCRIPTFILE:filename)“ command (see the list of AAO commands above for specs). The filename must be prefixed with the subfolder name if there is one: (SCRIPTFILE:myfolder\myfile.txt) (=the whole path relative to \Scripts\)

Recording script files

AAO can record all input actions that you perform with the app, all scripts that are called and, if you open the event observer „Scripting->Watch simulator events“ at the same time, also the simulator events that are triggered from within the simulator itself.

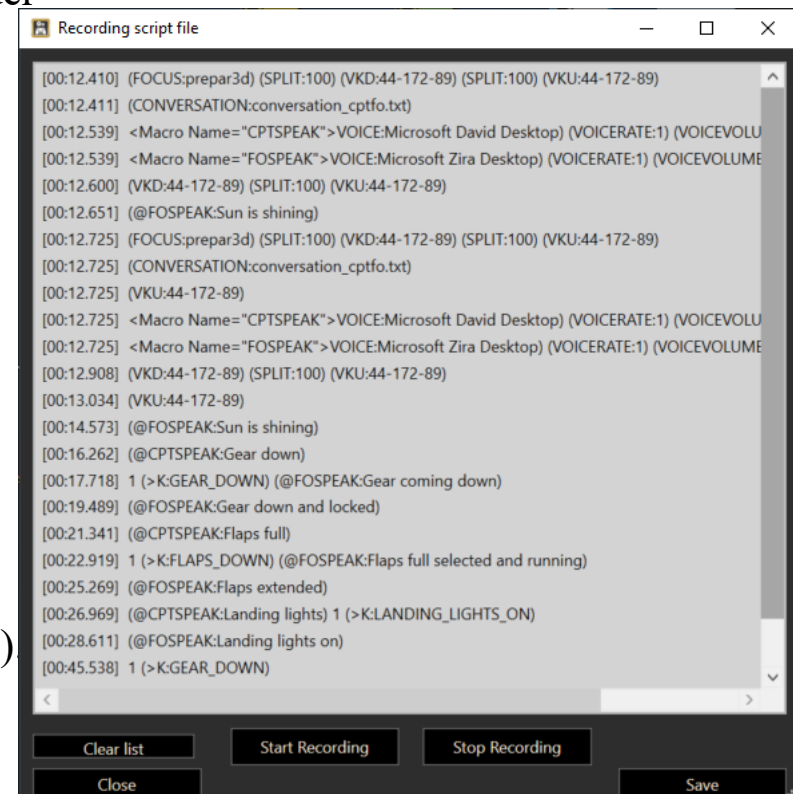
There are two ways to record actions:

1. using the AAO_RECORDING_ON / AAO_RECORDING_OFF commands

This will automatically create a script file in the script folder

2. using the dialog in „Scripting->Record script files“

- you can start and stop the recording on this dialog and watch the events coming in
- you can save to any file you wish, but keep them in the scripts folder
- upon saving, the app will ask if you want to include the time codes, so the events will be triggered at the same point in time as in your recording or not.
- If you want to record events that are coming from the simulator, you have to open the events observer dialog at the same time (all rules for ignoring events etc. apply)



25. Interactive Checklists

AAO has a hard coded feature to run interactive checklists. Checklists are RPN script files with a special format. Any line in the file that starts with [] (opening + closing bracket) will be regarded as part of a checklist, and AAO will switch to checklist mode at this point. Checklists are saved in \Documents\LorbyAxisAndOhs Files\Scripts.

Example:

Create a file: \Documents\LorbyAxisAndOhs Files\Scripts**a320_before_start_cl.txt**

```
<Macro Name="FOSPEAK">VOICE:Microsoft Zira Desktop) (VOICERATE:0) (VOICEVOLUME:75) (SPEAK</Macro>
(@FOSPEAK:Before start checklist) (WAIT:4000)
[] (@FOSPEAK:fueling)
(@FOSPEAK:checked)
[] (@FOSPEAK:chocks)
(@FOSPEAK:removed)
[] (@FOSPEAK:traffic cones)
(@FOSPEAK:clear)
[] (@FOSPEAK:A P U)
(@FOSPEAK:checked on)
[] (@FOSPEAK:A P U bleed)
(@FOSPEAK:checked on)
[] (@FOSPEAK:external power)
(@FOSPEAK:disconnected)
[] (@FOSPEAK:doors)
(@FOSPEAK:closed)
[] (@FOSPEAK:beacon)
(@FOSPEAK:checked on)
(WAIT:1500) (@FOSPEAK:before start checklist complete, pushback is next)
```

Operating the checklist:

The checklist is started by calling (**CHECKLIST:a320_before_start_cl.txt**) in an RPN script. On every line with the [] at the start, AAO will execute the rest of the line and then wait for your confirmation.

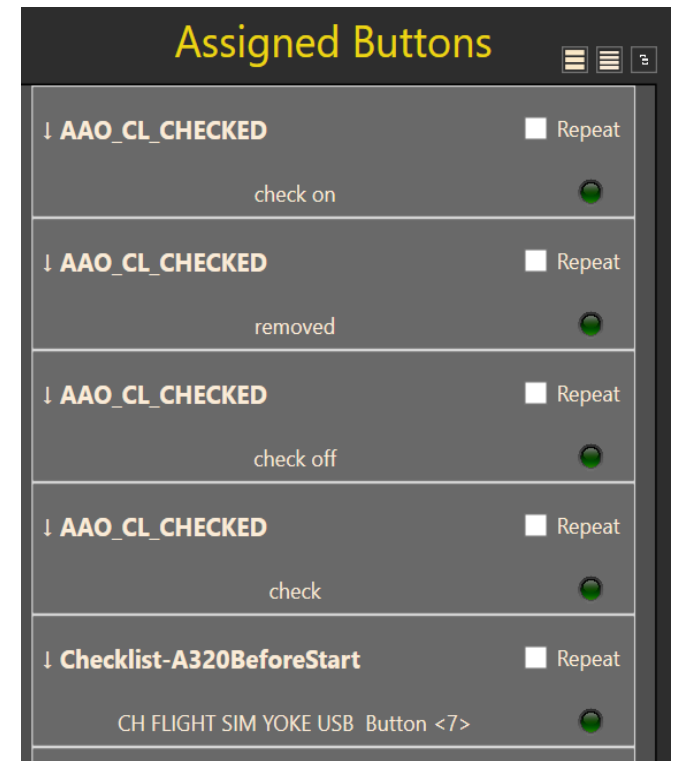
Confirmation is given by actuating the event „Audio Checklist → AAO_CL_CHECKED“ with a standard AAO button assignment. This can be done with any input device, including voice recognition.

If you don't confirm in 10 seconds, the command will be repeated automatically. If you fail to confirm more than 5 times, the checklist will be stopped.

The brackets [] can contain additional processing information:

- [0]: AAO will not wait for confirmation. See below example of a conversation between captain and first officer.
- [*number of ms*]: changes the confirmation timeout value. „[15000]“ means, that you will be asked for confirmation after 15 seconds. You can't set a value lower than 5000 ms (5 seconds)
- [**phrase**]: custom voice confirmation – see below
- [*rpn code*] will make the checklist wait at this point until the condition in the RPN code is met (see example on the next page)
- [*rpn code*|SKIP] will skip this line when the condition is not met

To stop a checklist you can use the AAO event „Audio Checklist → AAO_CL_STOP“ or the „(AAO_CL_STOP)“ RPN command



Checklist flow control

Flow control events are located in the group „Audio Checklist“ on the event selection dialog

AAO_CL_CHECKED: advance checklist

AAO_CL_STOP: stop checklist

AAO_CL_PAUSE_TOGGLE: suspend the checklist until the event is received again.

There are three LVars available that can be used for checklist flow control. They can be read and written to at any time. So if you want a different timeout, just set the LVar to a new value first thing in your checklist file.

(L:AaoClRepeatMs): contains the time in milliseconds until a waiting checklist starts to perform the repeat action

(L:AaoClTimeout): contains the number of repeat action that need to fail before the checklist will be stopped

For both the CHECKLIST and CONVERSATION you can query the current active line with (L:*filename.txt*).

A value of „0“ means that the scriptfile is not active.

Example: (L:conversation_cptfo.txt)

Custom repeat

If you want a different action to happen after the waiting checklist times out, add a line before the checklist item beginning and ending with a „*“, that contains the desired action:

```
*(VOICE:Microsoft Zira Desktop) (VOICERATE:0) (VOICEVOLUME:75) (SPEAK:Please check again)*  
[](@FOSPEAK:A P U)
```

Custom confirm (voice recognition only)

Parallel to the default confirmation event (AAO_CL_CHECKED) you can use lists of phrases inside the checklist item confirmation box that you want to speak yourself to advance the checklist. This only works with Windows voice recognition properly set up, see also the chapter about Voice recognition.

```
[yes|no|maybe|whatever] rpncode
```

After one of the phrases has been received, the checklist will be advanced and (L:AaoClConfirm) will contain the index of the phrase that has been recognized, starting with „1“. The default confirmation event has index „0“.

Example:

```
<Macro Name="COVOICE">VOICE:Microsoft Mark) (VOICERATE:0) (VOICEVOLUME:100) (SPEAK</Macro>
*(VOICE:Microsoft David Desktop) (VOICERATE:0) (VOICEVOLUME:100) (SPEAK:Please recheck)*
(@COVOICE: approach checklist) (WAIT:2000)
:redo
[*completed|no|maybe*] (@COVOICE: approach briefing)
(L:AaoClConfirm) 0 == if{ (@COVOICE:checked) (GOTO:cont) }
(L:AaoClConfirm) 1 == if{ (@COVOICE:confirmed) (GOTO:cont) }
(L:AaoClConfirm) 2 == if{ (@COVOICE:why not?) }
(L:AaoClConfirm) 3 == if{ (@COVOICE:what is maybe supposed to mean?) }
(GOTO:redo)
:cont
[] (@COVOICE: ecam status)
(@COVOICE:checked)
...
```

Conversation mode

You can also execute „conversations“ by calling a file in checklist format with (CONVERSATION:*filename.txt*)
For a conversation, no user interaction is required or expected, the list will just proceed through the lines.

You can run multiple conversation streams at the same time, but only one checklist.

In this example „conversation_cptfo.txt“, the captain gives commands to operate aircraft systems, which the first officer then executes and confirms. No interaction is required.

```
<Macro Name="CPTSPEAK">VOICE:Microsoft David Desktop) (VOICERATE:1) (VOICEVOLUME:95) (SPEAKBLK</Macro>
<Macro Name="FOSPEAK">VOICE:Microsoft Zira Desktop) (VOICERATE:1) (VOICEVOLUME:75) (SPEAKBLK</Macro>
;conversation example
[](@CPTSPEAK:Gear down)
[] 1 (>K:GEAR_DOWN) (@FOSPEAK:Gear coming down)
[(A:GEAR TOTAL PCT EXTENDED, Percent) 0.99 >] (@FOSPEAK:Gear down and locked)
[](@CPTSPEAK:Flaps full)
[] 1 (>K:FLAPS_DOWN) (@FOSPEAK:Flaps full selected and running)
[(A:TRAILING EDGE FLAPS LEFT PERCENT, Percent over 100) 0.99 >] (@FOSPEAK:Flaps extended)
[](@CPTSPEAK:Landing lights) 1 (>K:LANDING_LIGHTS_ON)
[(A:LIGHT LANDING ON, Bool)] (@FOSPEAK:Landing lights on)
```

To **stop** a conversation you can use the AAO command „(AAO_CV_STOP:*filename.txt*)“

To **pause** a conversation you can use the AAO command „(AAO_CV_PAUSE_TOGGLE:*filename.txt*)“

26. Wear & Tear simulation

The Wear & Tear module is an experimental feature in AAO. It is basically a wrapper around the automated RPN scripts, that can be used to simulate age and mechanical deterioration of parts of the aircraft, and to simulate random events or failures.

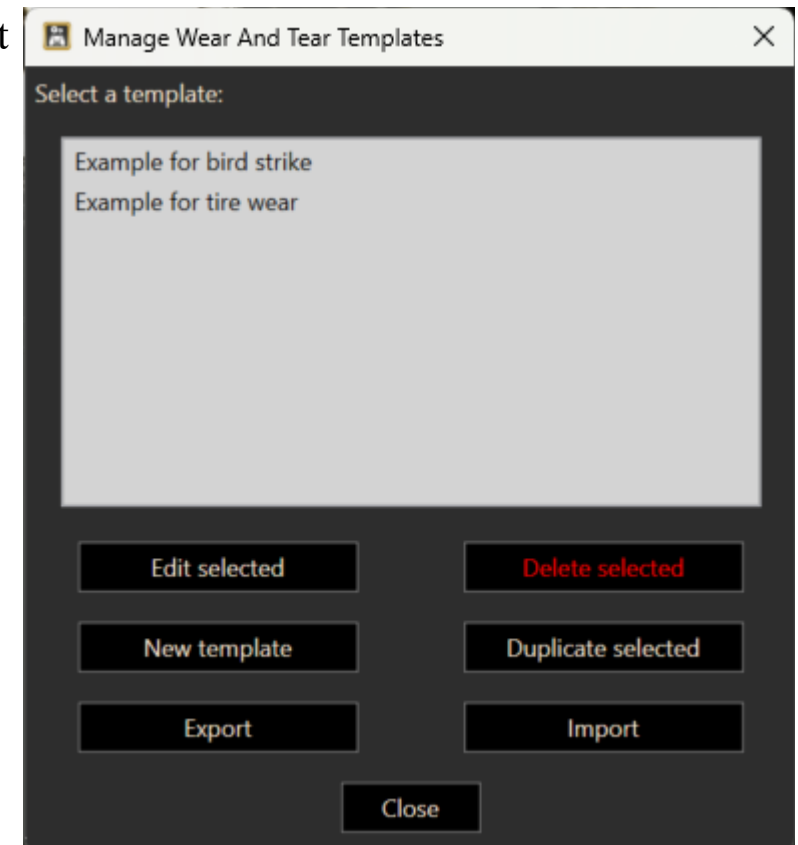
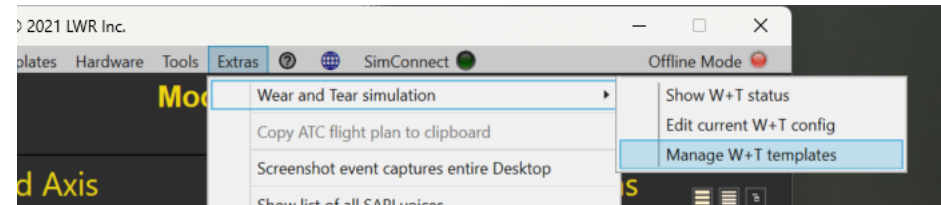
- The W&T simulation is organized in "**Templates**".
- Each template can have multiple "**Systems**"
- Each "**System**" has a status variable that shows the wear as a value between 0% and 100% (= new, 100% = run down)
- To each "System" you can add "**Rules**" that will affect the status variable. A "Rule" could be a simple counter adding a few points to the wear percentage every second or a more complex script adding points for example for environmental conditions (speed, altitude, temperature etc.)
- Finally, every "System" can have multiple "**Consequences**", which kick in once the wear status exceeds a certain threshold percentage. They are scripts that will either be executed once (and for example fail the combustion in an engine) or continuously (for example locking up the brakes on a the left gear to simulate a blown-out tire)

When you apply a Template to your aircraft, the app will create a copy of the template for this specific aircraft livery as a "**Configuration**" (same mechanism as with the buttons and axis). As a consequence, changing the template will not automatically change all W&T configurations for all aircraft!

The Wear & Tear feature is located in the "Extras" menu

- "Show W+T status" will display a dialog with the current values of the configured systems
- "Edit current W+T config" lets you change the configuration that is assigned to your current aircraft
- "Manage W+T Templates" can be used to edit, export, import Templates

Doubleclick on a Template to select it or use the buttons at the bottom of the dialog



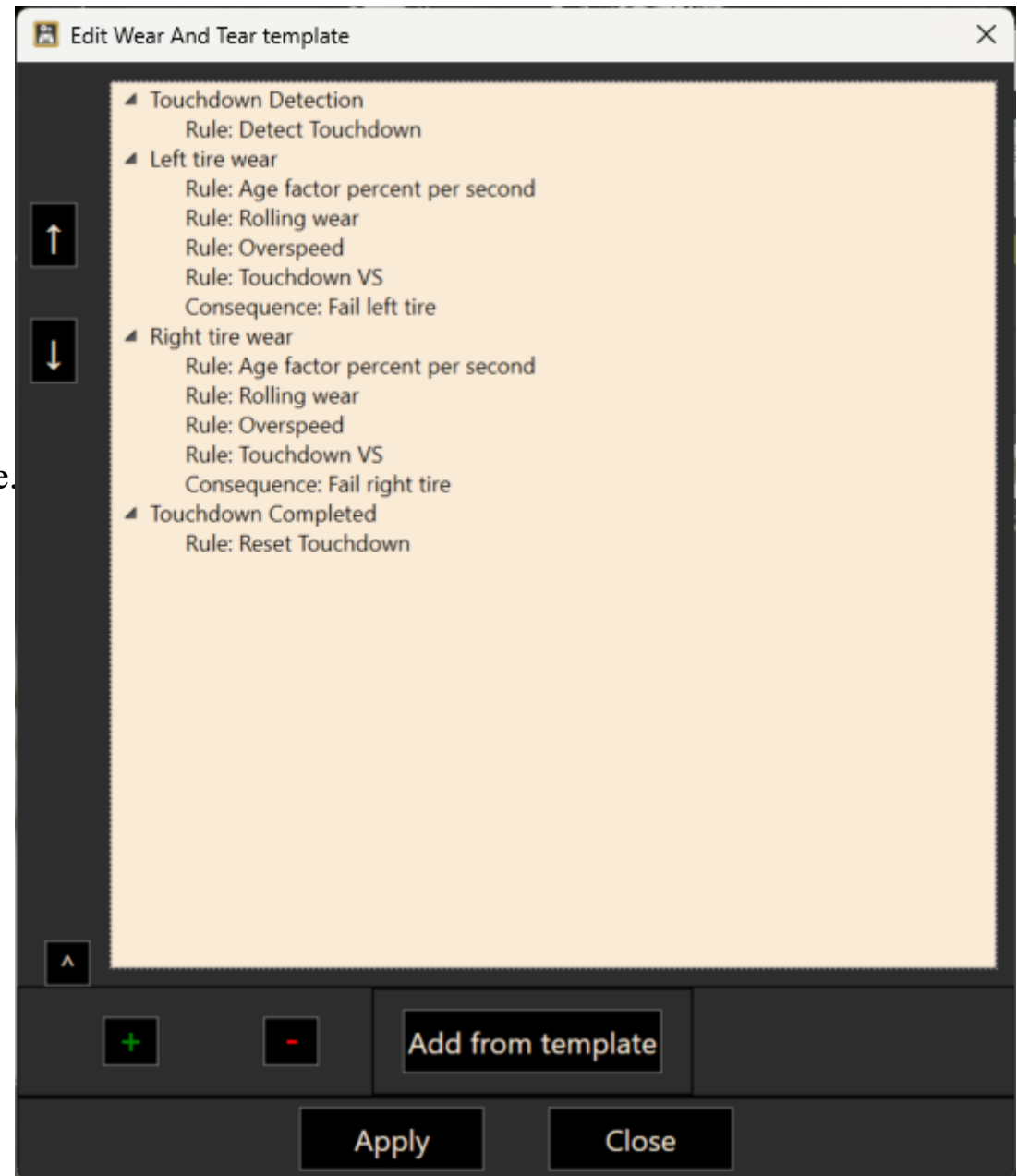
Editing a Template or a Configuration

This dialog looks the same for Templates and Configurations. It lists the active Systems with a brief description of their Rules and Consequences.

You can add a new System with "+" and remove the selected System with "-"

You can also add Systems from another Template.

Doubleclicking on a System will open it in the editor dialog.



Editing a System

On this dialog you can

- change the name of the System
- add, remove and re-arrange Rules
- add, remove and re-arrange Consequences
- Add an LVar of your choice to hold the current wear percentage of this system
(in case you want to display it on your own gauge or website)

Doubleclick a Rule or Consequence to edit it.



Editing a Rule

Rules consist of a Value Script and an Update Script. Both items are optional.

The purpose of the Value Script is to provide a value that will increase the wear percentage of the System every time the Rule is evaluated.

The Update Script is for extra code that you want to be executed with every update of the Rule.

The frequency of the updates can be set with the mouse wheel.

You can access the age of the System using "**(L:sysage)**" in your code. This is not a real LVar, the literal will be replaced at runtime with the actual age of this system in hours since the last reset.

The screenshot shows the 'Edit Rule' dialog box with the following fields and controls:

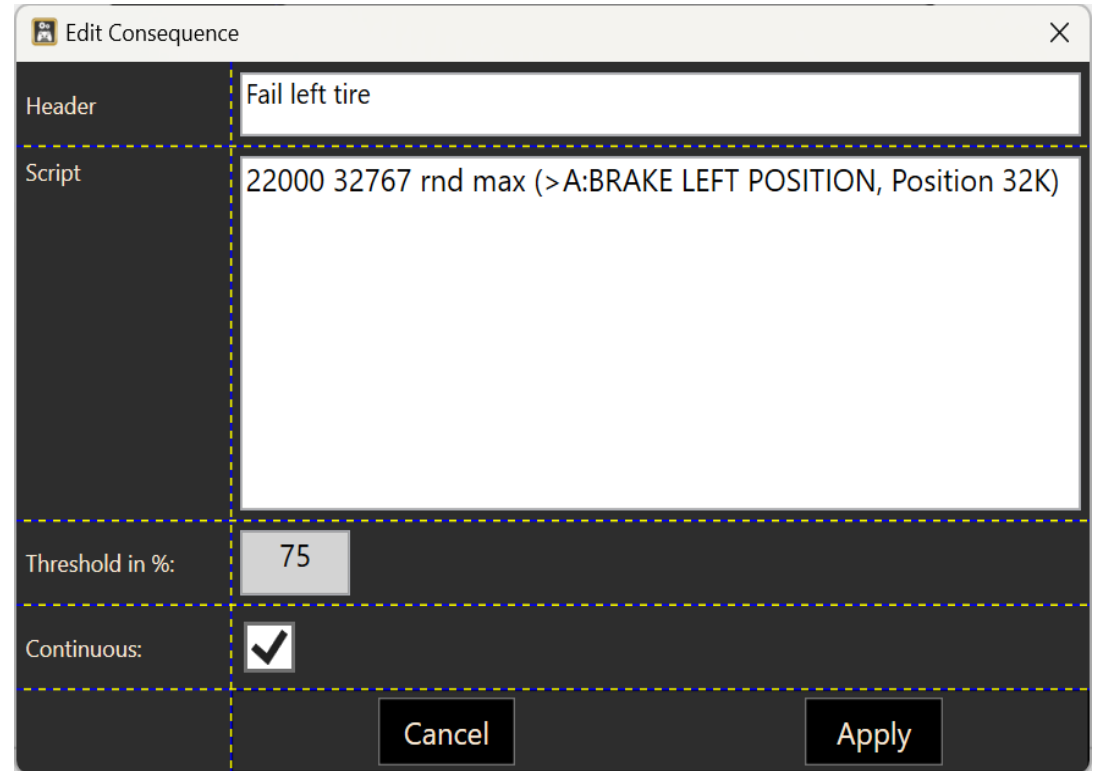
- Header:** A text field containing 'Rolling wear'.
- Value Script:** A checked checkbox followed by a text area containing the script: `(A:SIM ON GROUND, Bool) (A:GROUND VELOCITY, Knots) * 6000 /`.
- Update Script:** An unchecked checkbox followed by a greyed-out text area.
- Frequency in 100ms:** A numeric input field with the value '5'. To its right is a note: '1 = check/update 10 times per second, 10 = once per second, etc.'
- Buttons:** 'Cancel' and 'Apply' buttons at the bottom right.

Editing a Consequence

A Consequence contains a script that is executed once the wear percentage of the System exceeds the selected Threshold.

Normally the script is just executed once, but you can select "Continuous" to make it trigger all the time.

In this example, the left brake is applied with a randomized force all the time, thus simulating a blown tire.



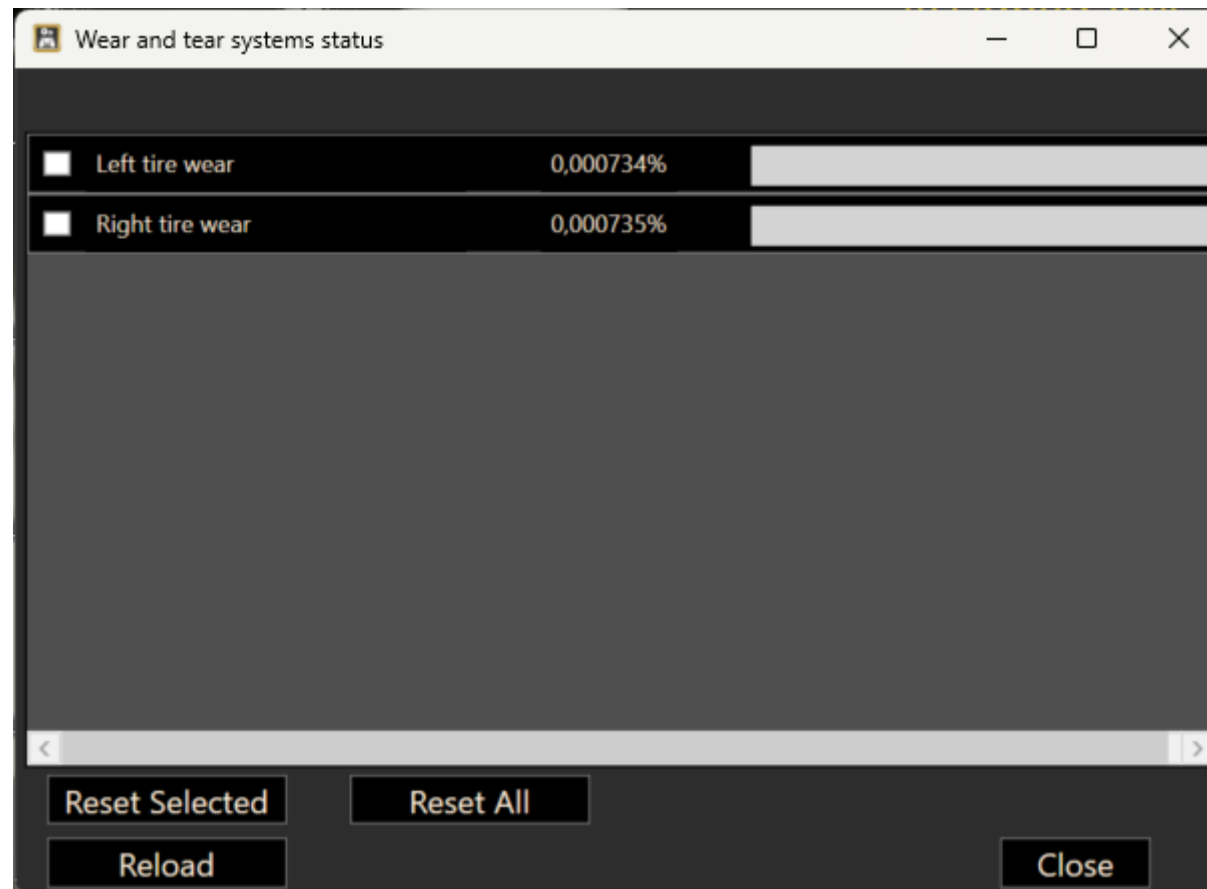
The screenshot shows a dialog box titled "Edit Consequence" with a close button (X) in the top right corner. The dialog is divided into four sections by dashed yellow lines:

- Header:** Contains the text "Fail left tire".
- Script:** Contains the text "22000 32767 rnd max (>A:BRAKE LEFT POSITION, Position 32K)".
- Threshold in %:** Contains a text input field with the value "75".
- Continuous:** Contains a checked checkbox.

At the bottom right of the dialog are two buttons: "Cancel" and "Apply".

Wear and Tear Status

When a Wear & Tear Configuration has been created for your aircraft, you can call up the current wear percentage status. You can also reset the Status back to 0% = "repair" the System.



If you added LVars to your System definitions, you can access these percentages from the outside too, by querying the LVar values.

27. vJoy Interface

If you have vJoy installed on your computer, you can use AAO to send virtual joystick button presses or axis movements to an application.

With this it is for example possible to trigger functionality that is not available in the SimConnect API, by assigning a vJoy button in the sim itself, and then trigger it from AAO.

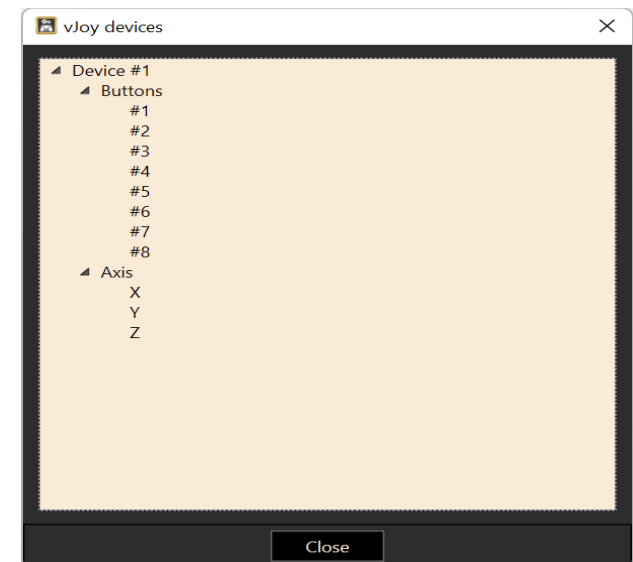
Furthermore, in Offline Mode, AAO can act as a "bridge" between more exotic hardware and any app that can be controlled with joystick buttons or movements. So the use of AAO is no longer limited to flight simulation. For example, if you want to use a Behringer X-Touch with your favourite racing game - you can.

vJoy can be downloaded from the project website <https://sourceforge.net/projects/vjoystick/>

The interface can be activated in the AAO menu with
"Hardware->vJoy interface enabled"

All vJoy devices that are available to AAO are shown with
"Hardware->Show vJoy devices"

vJoy allows only one app at a time to control a virtual device.
When a device is already in use, it will not show up in AAO.



28. ViGEm Interface

ViGEm is a small software that emulates an XBox 360 controller (like vJoy does for joysticks).

ViGEm can be downloaded from the project website <https://github.com/ViGEm/ViGEmBus/releases>

The interface can be activated in the AAO menu with
"Hardware->ViGEm interface enabled"

This is useful to bridge any hardware that AAO can handle to applications that only accept XBox controllers.

29. Virtual mouse

With AxisAndOhs you can send events to your mouse. That means, that you can control the mouse with an external device, position the cursor on the screen, send mouse button and wheel events. The location of the mouse cursor can be determined with „Hardware → Track mouse position“

Virtual mouse events can be assigned to axis and buttons alike.

Virtual mouse buttons and actions:

When you assign a button or an action in AAO, you can choose mouse actions as "virtual events".

"Move" and "Wheel" actions can be scaled with the "Movement scale" selection. Bigger values mean greater movement of the mouse cursor on screen.

Virtual mouse axis:

When assigning an axis, you can choose to bind the X or Y mouse movement to that axis. For this to work you must also configure a "Combo" button for the Axis, otherwise you could lock yourself out of the mouse controls entirely. For a mouse axis you must adjust Axis Min and Max so they fit your screen resolution, the mouse position is an absolute X,Y value on your screen. For relative movements, use the "Move..." mouse actions in axis Trigger Mode.

30. CAN Interface

AxisAndOhs can read CAN (Controller Area Network) messages from two types of physical interfaces:

- „CANable“ USB interface with „Candle“ compatible firmware
- CAN USB HID interface

CAN devices

The interface has been tested successfully using the following:

1. Arduino UNO with „keyestudio.com“ CAN-BUS shield
2. Arduino Mega with „seeedstudio.com“ CAN Bus shield V2
3. CAN HID hardware by CIS

[CAN-Bus Integration for Flight Simulators MSFS FSX X-Plane P3D \(caninsimulation.de\)](http://caninsimulation.de)

Sending CAN messages to hardware device with "device-id":

AAO Command	Parameters	Content
(CANMSG:device_id can_id data)	'can_id' (2 byte hex, 'hhhh')	data is a series of bytes in two digit hex format
(CANMSG:device_id can_id header n f)	'can_id' (2 byte hex, 'hhhh')	n is a floating point number
(CANMSG:device_id can_id header n i)	'can_id' (2 byte hex, 'hhhh')	n is a integer number

CAN message protocol

AAO will process the following messages:

1. CIS (CAN in simulation) standard

CIS is a subset of the CANas (CAN aerospace) standard. It covers the typical input events that are to be expected from simulation controllers, like joysticks, yokes, rotary encoders, keyboards and switchboxes.

Message format (8 bytes)

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Node ID	Data type	Encoder ID	Message counter	m1	m2	m3	m4

Node ID: unique numerical ID of the CAN node sending the message

Data type: identifier according to CANas standard

Encoder ID: ID of the encoder sending the data. Each node can have 256 encoders of every type
(rotary,switch, axis, keyboard)

Message Counter: 0-255 (increased with every new message)

m1-m4: data bytes

Message types

Type	CAN ID range	Data type	Data bytes
Rotary Encoder	708h - 70Fh	0Bh (BCHAR)	m1: Bit 7: fast rotation Bit 3: event PUSHBUTTON OFF Bit 2: event PUSHBUTTON ON Bit 1: event DOWN Bit 0: event UP
Switch	710h - 717h	0Bh (BCHAR)	m1: Bit 1: event OFF Bit 0: event ON
Analog (Axis)	718h - 71Fh	07h (USHORT)	m1, m2 16 bit integer value (0-65535)
Keyboard	720h - 727h	13h (UCHAR2)	m1: modifier code m2: key code
Value (results in an LVar)	730h - 73Fh	02h, 03h, 04h, 05h, 19h	m1, m2 16 bit integer value

- 2. CANaerospace standard, limited to the axis input messages in the CAN ID range of 4xx according to the chapter „Flight controls data“ in the „canas_17.pdf“ specification from „Stock Flight Systems“.**

31. Sound effects for audio and speech output

You can use a variety of sound effects with the SOUND and SPEAK commands. Sound effects are added to the script commands as follows:

- **For playing sound files**, the effects are added at the end of the command

(SOUND:xxxx|vvv|bbb|dd|eee-fff-ggg)

This plays sound file xxxx at the volume vvv and balance bbb on device ID dd using the effects chain eee-fff-ggg

- **With the text-to-speech output**, the sound effects have been moved to a separate command

(VOICEEFFECTS:eee-fff-ggg)

Be mindful that VOICE... commands are persistent. You have to switch the effects off if you don't want them on your next SPEAK command, by using (VOICEEFFECTS:0).

Effects chain format

An effects chain is built by supplying the effects ID, parameters and volume separated by commas, and adding other effects on top, separated by a minus sign. The effects build on each other, so the sequence in which you add them to the chain is important.

(VOICEEFFECTS:7,880,20-1,3-4,50,50)

This effects chain contains three effects:

- a sine wave tone at 880Hz
- a white noise hiss at low volume (3)
- a medium distortion effect (50) at medium volume (50)

When the next SPEAK command is called, you will hear a low hum and some white noise and the distorted voice.

(SOUND:mywave.mp3|100|0|-1|7,880,20-1,3-4,50,50): play the sound file with volume 100, center balance on the default sound device with the effects chain

Effect	ID	Parameters	Example
white noise	1	volume	1,3 - low hissing sound
pink noise	2	volume	2,50 - strong noise
flanger	3	volume	3,50 - moderate Flanger effect
distortion	4	amount,volume	4,10,50 - adds crackles and distortion
delay	5	amount,volume	5,100,50 - strong echo effect
tremolo	6	amount,volume	6,5,100 - makes the sound wobble"
sine wave	7	frequency,volume	7,440,25 - low standard pitch hum
square wave	8	frequency,volume	8,880,25 - high vibrating hum
sawtooth wave	9	frequency,volume	9,1320,25 - higher pitched vibrating hum

All parameter values are between 0 and 100, except for the frequency, which is in Hz.

Playing sound effects stand alone

Sound effects can also be played separately, on their own:

(PLAYEFFECT:id|vvv|bbb|dd|eee-fff-ggg|ms)

will play an effects chain **eee-fff-ggg** on device **dd** with volume **vvv** and balance **bbb** for **ms** milliseconds. Setting **ms** to 0 will play the effects chain indefinitely.

The "**id**" is a string of your choice to identify the signal when you want to stop it:

(STOPEFFECT:id)

32. Chat GPT interface

If you have a billable ChatGPT API account with OpenAI, you can use AAO to get text responses from the AI.

You can sign up for the OpenAI API here:

[Product \(openai.com\)](https://openai.com) Click on „Get started“

After you've created your account, click on the „flower“ symbol in the top left corner and select „API keys“. You can create a new key there. Make sure to read the pricing information, the ChatGPT API is not a free service.

Next, enter your API key in "Extras -> Enter ChatGPT API key". This is not the same as your credentials for the ChatGPT app, you have to create your own API account. Be mindful that this is not free of charge, using the API will cost you. At the moment, you can estimate about 1 Cent per request, depending on how big it is.

When you have supplied your key, you can query ChatGPT using RPN commands. AAO simulates a "chat completion" scenario, where you tell the AI who it is and what the situation is like, then ask it for a response.

(CHATGPTADDMESSAGE:role:text) adds an item to the scenario. Start with the role "system" to describe the situation, then use "user" and "assistant" to establish the chat.

(CHATGPTGETRESULT:engine|*lvarname*) sends the request to the AI. "engine" can be either "gpt-3.5-turbo" or "gpt-4". The resulting text will be written to "(L:*lvarname*, String)".

Example script:

```
''·(>L:purserspeak, ·String) ·\n
(ChatGPTADDMESSAGE:system:you ·are ·the ·purser ·on ·the ·lufthansa ·flight ·DLH2345 ·from ·munich ·to ·rome. ·It ·
is ·raining ·and ·the ·temperature ·is ·5 ·degrees ·celsius. ·The ·flight ·is ·departing ·at ·20:00) ·\n
(ChatGPTADDMESSAGE:user:please ·read ·the ·cabin ·announcement) ·\n
(ChatGPTGETRESULT:gpt-3.5-turbo|purserspeak) ·\n
```

Result:

(L:purserspeak, String)
containing

"Ladies and gentlemen, this is your purser speaking. On behalf of Lufthansa, I would like to welcome you aboard flight LH2345 from Munich to Rome.

I would like to inform you that our departure is scheduled for 20:00. However, due to the inclement weather conditions, it is currently raining with a temperature of 5 degrees Celsius in Munich. We kindly ask for your patience as we prepare the aircraft for departure.

As always, your safety and comfort are our top priority. Please ensure that your seatbelts are fastened securely, your electronic devices are switched off or in airplane mode, and your tray tables and seatbacks are in their upright position.

During the flight, our cabin crew will be available to assist you with any needs you may have. We offer a range of complimentary beverages and light snacks for your enjoyment during the flight.

Once again, we apologize for any inconvenience caused by the weather conditions. We will make every effort to minimize any delays and provide you with a pleasant flight experience.

Thank you for choosing Lufthansa, and we hope you have a wonderful journey to Rome."

The ChatGPT API is quite slow. If you are planning to, for example, have the result read to you, you must supply some SPLIT or GOTO logic and wait for the result:

```
'-'·(>L:purserspeak,·String)·\n
(ChatGPTADDMESSAGE:system:you·are·the·purser·on·the·lufthansa·flight·DLH2345·from·munich·to·rome·.It·
is·raining·and·the·temperature·is·5·degrees·celsius·.The·flight·is·departing·at·20:00)·\n
(ChatGPTADDMESSAGE:user:please·read·the·safety·instructions)·\n
(ChatGPTGETRESULT:gpt-3.5-turbo|purserspeak)·\n
:waitcomplete·\n
(L:purserspeak,·String)·slen·1·==·if{·(GOTO:waitcomplete)·}·\n
(VOICE:Azure|de-DE-ElkeNeural)·(VOICEEFFECTS:4,100)·(L:purserspeak,·String)·(SPEAK:%s1)·\n
```


33. Command line parameters

The AxisAndOhs exe file can be called with the following command line parameters

`-blacklistall`

This will put all devices that AAO can see on the blacklist. Use this if AAO doesn't want to start due to USB hardware issues

`-offline "aircraft livery"`

Will start AAO in offline mode and load the configuration for the aircraft.

34. Disclaimer

Lorby AxisAndOhs is licensed, not sold, for private use only. All property rights remain with the author. You may not distribute this package or parts of it. Disassembling, refactoring or changes of any kind are prohibited.

Disclaimer of Warranties. The author disclaims to the fullest extent authorized by law any and all other warranties, whether express or implied, including, without limitation, any implied warranties of title, non-infringement, merchantability or fitness for a particular purpose. Without limitation of the foregoing, the author expressly does not warrant that:

- the software will meet your requirements or expectations;
- the software or the software content will be free of bugs, errors, viruses or other defects;
- any results, output, or data provided through or generated by the software will be accurate, up-to-date, complete or reliable;
- the software will be compatible with third party software;
- any errors in the software will be corrected or that any further development will take place;
- the software will not cause errors or damage to the computer system it is installed on.

© 2021 Lorby Wildfire Response Inc.
Kamloops BC, Canada

support@wildfiretrainingsolutions.ca

<https://www.avsim.com/forums/forum/788-lorby-si-support-forum/>

<https://www.wildfiretrainingsolutions.ca/>