Table of Contents

1 Definitions

1.1 Control Binding Files

These files define which physical input controls which command in DCS World, and how the input signals drive the commands. They all have the extension ".lua". For reading and editing, they are best opened with Notepad++.

Input Controls are :

- Keyboard Buttons
- On/Off Switches, 3-Position Switches, Hat Switches, Toggle Switches, Push-Buttons on Joystick, Throttle, button box etc.
- Axis Controls on Joystick and Throttle
- Slew Controls on Joystick and Throttle
- TrackIR etc.

Note: In this document we do not discuss Axis and Slew Controls.

1.2 Physical Switch Types

These are the switches you have on your HOTAS devices, button box etc.

1.2.1 Maintained Switches

These switches latch (snap) into the commanded position after actuation. When you set the switch to the ON Position, the signal changes to ON and stays ON. Likewise, when you switch to OFF the signal changes to OFF and stays OFF. Thus, the physical switch shows the state of the command (e.g., ON or OFF). Examples:

- 2-Position ON/OFF
  This type of switch is bound to one command in Controls/Options.
- 3-Position ON/OFF/ON
  This is actually a combination of 2 ON/OFF switches with a common lever. In the OFF position both of the switches are OFF. In the two other positions, either the first or the second switch is ON.
  This type of switch is bound to two commands in Controls/Options.
  Example: the FLAPS switch on the TM HOTAS Throttle.
- Multi-Position OFF/ON/ON/ON…
  The most common occurrence of these are the hat switches commonly found on HOTAS devices. But they can also be rotary switches and can have any number of contacts. Only one of the contacts can be ON at a time. They are not common on flight simulation controls, but if available they would map nicely to rotary switches in the simulation cockpit.

1.2.2 Momentary Switches

These switches are spring-loaded and return to their neutral position when you release the switch after actuation. Therefore they produce a pulse signal that ends when you release the switch. Examples:

- Pushbutton
  The pushbutton produces an ON signal only as long as you hold it down.
  This type of switch is bound to one command in Controls/Options.
- ON/OFF Toggle
  The switch is actuated by a spring-loaded lever and produces an ON signal only as long as you put

pressure on the lever. It has the same function as a pushbutton.
This type of switch is bound to one command in Controls/Options.

- ON/OFF/ON Toggle, UP/DOWN Toggle
  This is basically a combination of two ON/OFF toggles with a common lever. In the neutral center position both switches are OFF. When you press the lever in one direction, then one of the switches produces an ON signal until you release the lever, and vice versa.
  This type of switch is bound to two commands in Controls/Options.
- Rotary Encoder
  This is a rotary knob which can be turned in both directions incrementally. It has two output contacts. One of the contacts produces short ON-Pulses (1 pulse per increment) when you turn the knob clockwise (CW), and the other contact does the same when you turn it counter-clockwise (CCW).
  This type of switch is bound to two commands in Controls/Options.

Notes:

1. Pushbutton and ON/OFF toggle can be bound to two kinds of commands:
   - Single-state commands like “Eject” or “Release Bomb”
   - Toggled two-state commands. Example: a toggled ON/OFF command. When you actuate the switch the first time it initiates an ON command. When you actuate the switch again it initiates an OFF command, and the next time an ON command again, and so on.
2. ON/OFF/ON Toggles and Rotary Encoders can be used to bind two independent commands each; but they are most useful when we bind two opposed commands, e. g:
   - Dashboard Lamp Brightness Increase/Decrease
   - Bomb Fusing Mode Next/Previous
3. Some switches found on HOTAS devices are a combination of maintained and momentary switches. Examples:
   - Speed brake switches on the TM Warthog Throttle. The forward switch is maintained and the aft switch is momentary.
   - ENG OPER switches on the TM Warthog Throttle. The forward switch is momentary and the aft switch is maintained.

1.3 Simulated Cockpit Switches

In the cockpits of the simulated aircraft we find all the types of switches described above. In many cases the software gives us different options for binding physical switches to them. For example, the cockpit switch may be a maintained ON/OFF switch. But the “default.lua” may give us 3 options for binding it:

- ON/OFF Toggle: Requires only a pushbutton or ON/OFF toggle
- Separate ON and OFF commands: Requires 2 maintained ON/OFF switches or an ON/OFF/ON toggle
- 2-Position ON/OFF:: Requires only one maintained ON/OFF switch

Rotary switches in the simulated cockpit are usually implemented as two opposed commands like UP/DOWN INCR/DECR or CW/CCW, and are best bound to ON/OFF/ON toggles or rotary encoders.
Sometimes, the options for binding a command may appear unsatisfactory. For example, the KA-50 has some maintained ON/OFF switches in its simulated cockpit, but the only input option is a toggled ON/OFF soo you cannot bind a maintained ON/OFF switch to it.

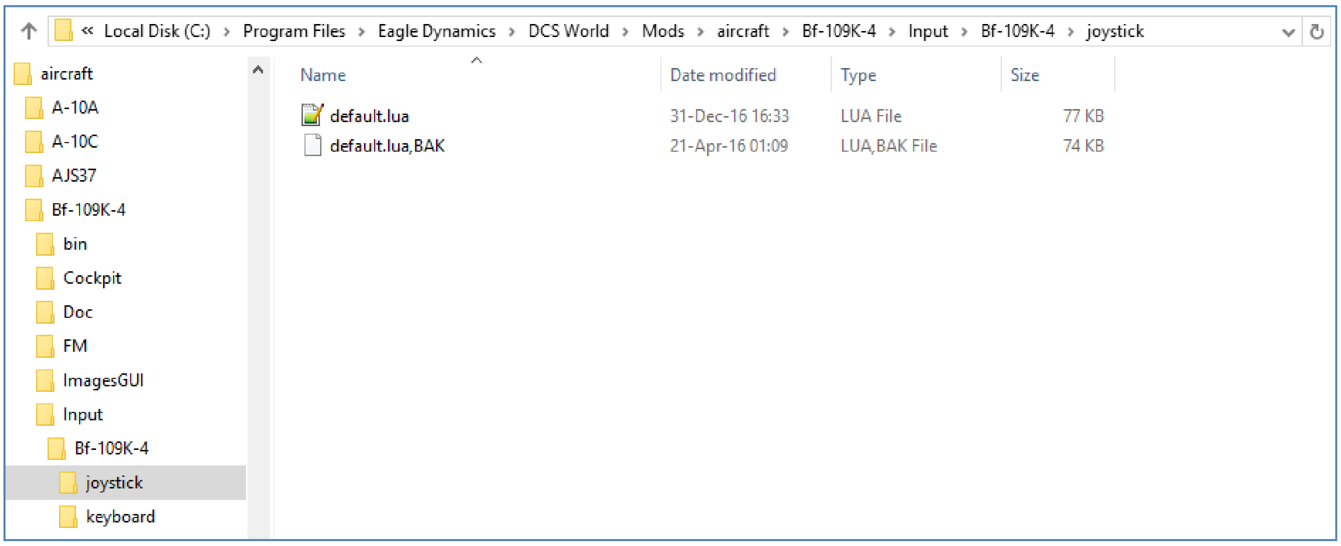
2 Why do we want to edit Control Binding Files?

In many cases, the original files prepared by the developers do not allow us to optimize the usage of the buttons on Joystick and Throttle. For example, they often require the use of 2 switches for switching a light On and Off, or 3 switches for 3-Position Control.

3 Which files are involved in the binding of controls?

3.1 Default Files

These files define the commands that can be bound to Input Controls. They can also define default bindings as we will see. The picture below shows where these files are located for the Bf-109K-4 Sim Mode.



« Local Disk (C:) › Program Files › Eagle Dynamics › DCS World › Mods › aircraft › Bf-109K-4 › Input › Bf-109K-4 › joystick

- aircraft
  - A-10A
  - A-10C
  - AJS37
  - Bf-109K-4
    - bin
    - Cockpit
    - Doc
    - FM
    - ImagesGUI
    - Input
      - Bf-109K-4
        - joystick
        - keyboard

Name	Date modified	Type	Size
default.lua	31-Dec-16 16:33	LUA File	77 KB
default.lua,BAK	21-Apr-16 01:09	LUA,BAK File	74 KB

In this example I have created a back-up file of “default.lua” before making changes. That’s good practice and provides a fall-back in case you screw up badly.

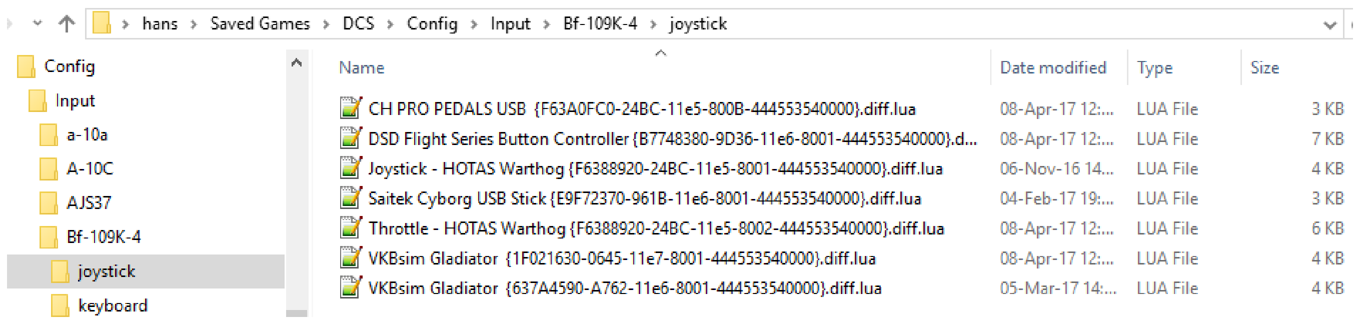
Note: There is also a folder-109K -“Bf4easy. This folder contains the equivalent files for Game Mode.

For some modules, this folder also contains one or more files for specific HOTAS devices, e.g. “Throttle - HOTAS Warthog.lua”. The content of these files is usually almost the same as that of the “default.lua”, but it can have additions; e.g. default bindings. Example: the one for the FW-190D contains default bindings for the 3-Position Flaps Switch on the TM Warthog Throttle.

While the folder “joystick” contains the Default Files for Joystick, Throttle and Pedals, the folder “keyboard” contains the keyboard commands with their keyboard bindings.

3.2 Diff Files

These files record the user entries and changes that are made via DCS World Options / Controls. They are located under the user's "Saved Games" folder as shown below:



hans › Saved Games › DCS › Config › Input › Bf-109K-4 › joystick

- Config
  - Input
    - a-10a
    - A-10C
    - AJS37
    - Bf-109K-4
      - joystick
      - keyboard

Name	Date modified	Type	Size
CH PRO PEDALS USB {F63A0FC0-24BC-11e5-800B-444553540000}.diff.lua	08-Apr-17 12:...	LUA File	3 KB
DSD Flight Series Button Controller {B7748380-9D36-11e6-8001-444553540000}.d...	08-Apr-17 12:...	LUA File	7 KB
Joystick - HOTAS Warthog {F6388920-24BC-11e5-8001-444553540000}.diff.lua	06-Nov-16 14...	LUA File	4 KB
Saitek Cyborg USB Stick {E9F72370-961B-11e6-8001-444553540000}.diff.lua	04-Feb-17 19:...	LUA File	3 KB
Throttle - HOTAS Warthog {F6388920-24BC-11e5-8002-444553540000}.diff.lua	08-Apr-17 12:...	LUA File	6 KB
VKBsim Gladiator {1F021630-0645-11e7-8001-444553540000}.diff.lua	08-Apr-17 12:...	LUA File	4 KB
VKBsim Gladiator {637A4590-A762-11e6-8001-444553540000}.diff.lua	05-Mar-17 14:...	LUA File	4 KB

If you have added, changed or deleted Control bindings at different points in time, you will find multiple files for some of the input devices. In that case it becomes difficult to see which changes one has made. I therefore does not modify these files directly. It is easier and safer to make these changes in the DCS World GUI.
Also if you add more input devices and create/change bindings for them, the system will add .diff.lua files for them as in the example above.

4 Reading the Default Files

4.1 Basic Example

Here is an example showing 3 lines out of the “default.lua” for the Bf-109K:

```
{down = device_commands.Button_7, cockpit_device_id = devices.WEAPONS_CONTROLS, value_down = 1.0, name = _('Gunsight Fold DOWN'), category = _('REVI 16 B Gunsight')},
{down = device_commands.Button_7, cockpit_device_id = devices.WEAPONS_CONTROLS, value_down = 0.0, name = _('Gunsight Fold UPRIGHT'), category = _('REVI 16 B Gunsight')},
 {down = device_commands.Button_8, cockpit_device_id = devices.WEAPONS_CONTROLS, value_down = 1.0, name = _('Gunsight Fold (toggle)'), category = _('REVI 16 B Gunsight')},
```

Note: some of the original file use lots of spaces or tabs which are supposed to improve readability but have no effect on functionality. These have been removed here to prevent wrapping.
The actions in this case are defined in a hierarchical structure; the first action being “Button_7” of the object “device.commands”, which is itself part of “devices.WEAPONS_CONTROLS. This example shows 5 keywords with their respective values:

- *down*: the action when the bound switch is in the ON position.
  The string “device_commands.Button_7” defines the command which will be executed. The format of these command ID’s varies and different developers have different “handwriting”.
  Other keywords in this category:
  - *up*: the action when the bound switch is in the OFF position. We will use this to get 2-Position control with a single switch
  - *pressed*: the action that will continue as long as the switch is held in the ON position. This is typically used for pushbutton control like “brightness increase”. Such commands are not candidates for customization as you cannot reduce the number of switches per command.
- *cockpit_device_id*: the group of commands to which this command belongs.
- *value_down*: the command’s value when the switch is ON.
  In this case, the value 1.0 is interpreted as “Gunsight Fold Down” and 0.0 as “Gunsight Fold Upright”. Again, the definition of the values is up to the developer.
  Other keywords in this category:
  - *value_up*: the command value when the switch is in the OFF position
  - *value_pressed*: the command value that continues to be active as long as the switch is in the ON position.
- *name*: the text string that will be used in DCS World Options / Controls. It has no effect on functionality but it needs to be chosen carefully so the user will know what to expect. And the name must be unique!
- *category*: this is used in DCS World Options / Controls to group the commands, and again it has no effect on functionality.

The first 2 lines result in 2 lines in DCS World Options / Controls, with the text “Gunsight Fold DOWN” and “Gunsight Fold UPRIGHT” respectively. This will require two switches to be bound for ON/OFF control. The 3rd line results in 1 line in DCS World Options / Controls, with the text “'Gunsight Fold (toggle)”. This requires only 1 switch or pushbutton to be bound, but has the disadvantage that you don’t know the actual position unless you look at the switch in the cockpit. (No problem in this case because you see the gun sight itself)

4.2 Variations

4.2.1 Simple 2-Position command

```
{down = iCommandPlaneWheelBrakeOn, up = iCommandPlaneWheelBrakeOff, name = _('Wheel brake Both'), category = _('Systems')},
```

This command is from the “Throttle - HOTAS Warthog.lua” of the FW-190D. It would be used to switch brakes ON and OFF with a single toggle switch or pushbutton, similar to using the “W” button on the keyboard. The actions are defined in a flat (not hierarchical) structure and therefore there is no need for a cockpit_device_id. Also in this case there is no need for “value_up” or “value_down” statements.

4.2.2 Very different “handwriting”

```
{down = 3647, up = 3647, cockpit_device_id = 4, value_down= 1, name = _('Left Fuel Boost Pump ON'), category = _('Engine & Fuel')},
 {down = 3647, up = 3647, cockpit_device_id = 4, value_down= 0, name = _('Left Fuel Boost Pump OFF'), category = _('Engine & Fuel')},
```

These lines are from the “default.lua” of the M-2000C. The actions and cockpit device_id’s are purely numeric identifiers. But in essence it’s the same thing as in the first example. They require two ON/OFF toggle switches or pushbuttons.

4.2.3 3-Position Control

```
{down=device_commands.Flaps0,cockpit_device_id=devices.FLAPS,value_down=1.0,name='Flaps 0',category='Flight Controls'},
{down=device_commands.Flaps25,cockpit_device_id=devices.FLAPS,value_down=1.0,name='Flaps 25',category='Flight Controls'},
{down=device_commands.Flaps45,cockpit_device_id=devices.FLAPS,value_down=1.0,name='Flaps 45',category='Flight Controls'},
```

These lines are from the “default.lua” of the Mig-21bis, and they define the 3-Position flaps control, requiring 3 ON/OFF toggle switches or pushbuttons for binding. We will use this as an example for modification.

5 Modifying the Default Files

5.1 Purpose of Modifications

We want to make these modifications in order to reduce the number of physical switches on HOTAS or button box devices that need to be bound. Thus, for 2-Position control we want to use one switch instead of 2, and for 3-Position control we want to use one 3-Position switch instead of 3. We do understand that every update will overwrite these files with their original version. Therefore we should always make backup copies of these modified files so we can restore the modifications after an update, or manage our modified files with a tool like OvGME.

5.2 Editor

It is highly recommended to use an editor that's designed for editing program files. I use Notepad++ which is free software and finds it very satisfactory.

5.3 Steps for making additions to a Default File

- Create a backup of the original file
- Start Notepad++ selecting "Open as administrator"
- Open the Default File, e.g., "default.lua" under "C:\Program Files\Eagle Dynamics\DCS World\Mods\aircraft\Bf-109K-4\Input\Bf-109K-4\joystick"
- Search for existing lines related to the command you want to add (e.g. "MW-50"
- Study and understand the existing lines
- Add the line(s) for 2-Position or 3-Position control, following the instructions below. Don't forget to create a new name!
- Save the file. Note: if you had not opened Notepad++ as administrator, it will pop up a message box saying that it cannot save because you don't have admin rights, and ask if you want to open Notepad++ as admin. Confirm this and Notepad++ will come back with the file still open. Now click "Save" and the file is saved.
- Start DCS world and bind the desired switch to the new command.
- Test the function in a mission
- Create a backup of the modified file for restoration after an update or repair (or use OvGME to manage the modified files)

5.4 Creating 2-Position control with a single ON/OFF switch (Bf-109K)

We want to use a single maintained ON/Off switch for the MW-50 ON/OFF control. In the “default.lua” we find these lines (numbered 357 – 359):

```
{down = device_commands.Button_36, cockpit_device_id = devices.FUSEBOX, value_down = 1.0, name = _('MW 50 Boost System ON'), category = _('Front Dash')},
{down = device_commands.Button_36, cockpit_device_id = devices.FUSEBOX, value_down = 0.0, name = _('MW 50 Boost System OFF'), category = _('Front Dash')},
{down = device_commands.Button_37, cockpit_device_id = devices.FUSEBOX, value_down = 1.0, name = _('MW 50 Boost System (toggle)'), category = _('Front Dash')},
```

The first 2 lines are defined for ON and Off control with 2 switches. The 3rd line is for ON/OFF toggle and not of interest. We will build on the 1st two lines, and we see that both ON and OFF use “device_commands.button_36. “value_down” is 1.0 for ON and 0.0 for OFF. With [Enter] we create two empty lines under line 359. We enter a comment line with initials and date in line 360. Then we copy the first line (357) and paste it into line 361. The additions so far look like this:

```
--added 1 line below HWF 31-JUL-2016
{down = device_commands.Button_36, cockpit_device_id = devices.FUSEBOX, value_down = 1.0, name = _('MW 50 Boost System ON'), category = _('Front Dash')},
```

Next we insert definitions for the switch’s OFF position and change the name:

```
{down = device_commands.Button_36, up = device_commands.Button_36, cockpit_device_id = devices.FUSEBOX, value_down = 1.0, value_up = 0.0, name = _('MW 50 Boost System 2-Pos ON/OFF'), category = _('Front Dash')},
```

The line “MW 50 Boost System 2-Pos ON/OFF” will now appear when we open Options / Controls in DCS World, and we can assign a switch to it. This needs to be a maintained ON / OFF switch.

5.5 Extending a control from “…keyboard\default.lua” to “…joystick\default.lua” (Mig-21bis)

The developers sometimes omit to include a command in the joystick Default File which is available in the keyboard Default File. We can fill this gap by copying the lines from the keyboard defaults and pasting them into the joystick defaults. In this example we want to use a 3-Position switch (rather than a slider) to change the ASP target size. In Options / Controls we see that “Target Size +” and “Target Size –“ are available on the keyboard only. So we will “steal” the command definition from the keyboard defaults.

Steps:

1. Create a back-up copy of the “default.lua” under …\joystick\”
2. Open both “default.lua” under …\joystick\” and “default.lua” under “…\keyboard\” Note: Notepad++ allows you to open several files with the same name (from different folders of course) in separate tabs.
3. In the keyboard file, find the 2 lines for Target Size (currently in line 619 and 621) and copy them to the clipboard
4. Go to the joystick file and find the category “Weapons / ASP” (currently in lines 535 – 550)
5. After the last line of the category, create 4 new lines
6. Create a comment line with initials and date in the first free line

7. Paste the lines copied from the keyboard file into the joystick file in the line below the comment line
8. Understand the functionality
9. Modify the 2 lines by removing the “combos” statement
10. Save the changed joystick file
11. Create a backup of the changed joystick file (or manage with OvGME)

Here are the added lines in our joystick file after step 7 and removing the blank line:

```
--added HWF 01-AUG-2016
{combos={{key='S'}},pressed=device_commands.ASPtargetSize_kb,up=device_commands.ASPtargetSize,cockpit_device_id=devices.ASP,value_pressed=0.02,name='Target Size +',category='Weapons / ASP'},
{combos={{key='S',reformers={'LAlt'}}},pressed=device_commands.ASPtargetSize_kb,up=device_commands.ASPtargetSize,cockpit_device_id=devices.ASP,value_pressed=-0.02,name='Target Size -',category='Weapons / ASP'},
```

After studying the two lines we conclude that both lines are good to go; except we do not want them bound to the keyboard buttons. We could actually replace the {{key=”S”}} and {{key='S',reformers={'LAlt'}} with {{key=”JOY_BTN4”}} and {{key=”JOY_BTN6”}} which would bind them to “Mic Forward” and “Mic Aft” on the TM Warthog Throttle. But we want to handle the binding via the Options / Controls GUI. So the only change we will make is to remove the “combos” statements from both lines. Here are the modified lines:

```
--added HWF 01-AUG-2016
{pressed=device_commands.ASPtargetSize_kb,up=device_commands.ASPtargetSize,cockpit_device_id=devices.ASP,value_pressed=0.02,name='Target Size +',category='Weapons / ASP'},
{pressed=device_commands.ASPtargetSize_kb,up=device_commands.ASPtargetSize,cockpit_device_id=devices.ASP,value_pressed=-0.02,name='Target Size -',category='Weapons / ASP'},
```

After saving the file we launch DCS World, go to Options / Controls for the Mig-21bis and see that the commands Target Size + And Target Size – are now open for throttle and joystick. This is because we have added the two lines to the joystick file. The commands are still bound to the keyboard buttons “S” and “Shift + “S”, because we have not removed the lines from the keyboard file. We can now run a quick mission to see that the ASP Target size wheel actually rotates up and down with our throttle commands.

5.6 Creating a 3-Position Flaps control (Mig-21bis)

The original joystick defaults file for the Mig.21bis provides 3 commands for Flaps Up/Flaps Take-Off Position/Flaps down, and without the modification we would need to sacrifice 3 switches. This could be the 2 switches of a 3-position switch plus a simple ON/OFF switch, or 3 switches from a hat switch, or 3 simple ON/OFF switches. None of these solutions is satisfactory. We want to use the 3-Position “Flaps” switch on our TM Warthog throttle for the 3 commands, and we can do it. It’s a bit trickier than for 2-Position ON/OFF commands, but no rocket science.

We open the joystick defaults file with Notepad++ in admin mode and find the 3 lines related to Flaps control:

```
{down=device_commands.Flaps0,cockpit_device_id=devices.FLAPS,value_down=1.0,name='Flaps 0',category='Flight Controls'},
{down=device_commands.Flaps25,cockpit_device_id=devices.FLAPS,value_down=1.0,name='Flaps 25',category='Flight Controls'},
{down=device_commands.Flaps45,cockpit_device_id=devices.FLAPS,value_down=1.0,name='Flaps 45',category='Flight Controls'},
```

The task at hand is to achieve the same result with only 2 lines. We see that this module uses 3 different command ID's for positions 0°, 25° and 45° respectively, rather than use different values for "value_down". We will copy and modify the 1st and the 3rd line, and insert the function of the 2nd line into both for the middle position (25°). We do not delete the original lines! Here are the completed new lines:

```
-- Added HWF 28-JUN2016
{down=device_commands.Flaps0,up=device_commands.Flaps25,cockpit_device_id=devices.FLAPS,value_down=1.0,value_up=1.0,name ='Flaps 3-Pos Up',category='Flight Controls'},
{down=device_commands.Flaps45,up=device_commands.Flaps25,cockpit_device_id=devices.FLAPS,value_down=1.0,value_up=1.0,name='Flaps 3-Pos Down',category ='Flight Controls'},
```

The 1st line states: If the switch is ON then flaps to 0° (UP), If the switch is OFF then flaps to 25° (take-off position)
The 2nd line states: If the switch is ON then flaps to 45°, If the switch is OFF then flaps to 25°. And we have changed the names to "Flaps 3-Pos Up" and "Flaps 3-Pos Down".

5.7 Using a Maintained ON/OFF switch for an ON/OFF toggle command (KA-50)

The simulated cockpit has a maintained ON/OFF switch for the command "Helmet Mounted System On/Off", but it provides a binding only for an ON/OFF toggle switch. This inconsistency between what we see in the cockpit and the physical switch is not to my liking. Fortunately there is a solution that works unexpectedly well. I want to use the "EAC" switch on my TM Warthog throttle which is a maintained ON/OFF switch. Therefore I need to make the addition in the file "Throttle - HOTAS Warthog.lua".
The original file has only this entry for the command:

```
{down = iCommandPlaneModeHelmet,  name = _("Helmet-mounted system On/Off"),  category = _("Ins Targeting Mode Controls Panel PVR"), },
```

I decide to try this addition, expecting it to be only half-satisfactory:

```
--Line added below HWF 26-APR-2017
{down = iCommandPlaneModeHelmet, up = iCommandPlaneModeHelmet, name = _('Helmet-mounted system 2-Pos. On/Off'), category = _('Ins Targeting Mode Controls Panel PVR')}, -- use with ON/OFF switch
```

This tells the software to toggle between OFF and ON each time I change my switch position from OFF to ON or from ON to OFF. Plain logic tells me that this will give me the desired result only if…

1. The Option “Synchronize Cockpit Controls with HOTAS controls at Mission Start” is active
2. The physical switch is in the OFF position when I launch the mission.

Otherwise the physical switch may be in the ON position while the Helmet Mounted System in the cockpit is actually OFF, and vice versa. That’s why I expected a half-satisfactory solution
When I tested this, I was positively surprised. If the physical switch is ON at mission start the Helmet Mounted System switch is also in the ON position, and (unless it’s a cold start of course) I actually see the helmet-mounted targeting sight. So the solution is fully satisfactory and I found that it also works with a few other aircraft. But unfortunately, only a few!
The option per 1. above should be active anyway so I don’t consider that as a restriction.

5.8 Creating new control options based on information in “clickabledata.lua”

The file “clickabledata.lua” is located (example for P-51D) in the folder “…\Mods\aircraft\P-51D\Cockpit\Scripts. It defines the interactions that are supported for clickable cockpits. In general, they define first the types of switches that are supported and their respective options, and then define the various cockpit commands and assign them to switch types.
In the definition of the cockpit commands there is always a text string containing the name of the command, and that allows us to correlate cockpit controls with the commands in the “default.lua”. For some aircraft, the “clickabledata.lua” defines commands which the developers have not added to the “default.lua”. In that case, we can create missing commands in “default.lua” based on what we find in “clickabledata.lua”. Examples:

5.8.1 Create Maintained ON/OFF commands for the P-51D

The P-51D cockpit has a number of 2-Position ON/OFF switches for which the “default.lua” only allows the use of pushbuttons or ON/OFF toggles. Examples: Battery, Generator, Pitot Heat, Fuel Shut-Off. We want to use Maintained ON/OFF switches for these commands. Luckily the “clickabedata.lua” of the P-51D gives us some options. I will show to create the command “Battery 2-Pos. ON/OFF”.
In “clickabledata.lua” we find several lines with battery commands:

```
elements["pnt_103"] = default_2_position_tumb(_("Battery Disconnect"),devices.RIGHT_SWITCH_PANEL, device_commands.Button_1,103)
elements["pnt_103_1"] = default_1_position_tumb(_("Battery Connect") ,devices.RIGHT_SWITCH_PANEL,device_commands.Button_12,103,1,{1,1})
elements["pnt_103_0"] = default_1_position_tumb(_("Battery Disconnect") ,devices.RIGHT_SWITCH_PANEL,device_commands.Button_12,103,0,{0,0})
```

The 1st line is for use with an ON/OFF toggle; but the 2nd and 3rd line define two separate commands for ON and OFF, and they tell us that the command ID is “device_commands.Button_12” in the device “RIGHT_SWITCH_PANEL”. We will build on this knowledge. In the “default.lua” we find this line:

```
{down = device_commands.Button_2, cockpit_device_id  = devices.RIGHT_SWITCH_PANEL, value_down = 1.0,         name = _('Battery'), category = _('Right Switch Panel')},
```

Under this line we add the new command:

```
-- added one line below HWF 04-06-2017
{down = device_commands.Button_12, up = device_commands.Button_12, cockpit_device_id  = devices.RIGHT_SWITCH_PANEL, value_down = 1.0,  value_up = 0.0,  name = _('Battery 2-Pos ON/OFF'), category = _('Right
Switch Panel')},
```

The easiest way to create the new command line is to copy/paste the original one and then make additions and changes. But you have to be careful to change everything that needs to be changed, and make sure you make the name of the command unique! After starting DCS World we go to Options/Controls and bind the newly created command.

5.8.2 Create maintained ON/OFF and 3-position commands for the KA-50

The KA-50 cockpit has many maintained switches, but the out-of-the-box software does not support the use of maintained switches for these commands. And the solution per 5.7 is not satisfactory.
Fortunately there is a solution using information from

- “clickabledate.lua” to find the command and device definition
- “devices.lua” to find the numerical device ID
- “command_defs.lua” to find numerical command ID

The KA-50 does not work with the mnemonic command ID (like “Button_1”) and device ID (like WEAP in the “default.lua”. That’s why we need to look at the additional files to find out what the numerical ID’s are.

We will implement the solution for the following commands:

- Master Arm ON/OFF
- Cannon Round Selector HE/API
- Cannon Rate of Fire LOW/HIGH
- Laser Standby ON/OFF
- Weapon Mode Burst Length SHORT/MEDIUM/HIGH

I describe the approach step by step for the Master Arm ON/OFF command:

5.8.2.1 *Find mnemonic command and device ID's in "clickabledata.lua"*

We use the "Search" in Notepad++ to locate "Master arm" and find this code line:

```
elements["MASTER-ARM-PTR"]= {class = {class_type.TUMB, class_type.TUMB}, hint = LOCALIZE("Master Arm"), device = devices.WEAP_INTERFACE, action = {device_commands.Button_1,device_commands.Button_1},
stop_action = {}, arg = {387,387}, arg_value = {-direction*1.0,direction*1.0}, arg_lim = {{0.0, 1.0},{0.0, 1.0}}, use_OBB = true, updatable = true}
```

The mnemonic device ID is "WEAP INTERFACE", the mnemonic command ID is "Button_1", and the command values seem to be 1.0 and 0.0.

5.8.2.2 *Find the numerical device ID*

We open "devices.lua" and search for "WEAP INTERFACE". We find that the numerical device ID is 12.

5.8.2.3 *Find the numerical command ID*

We open the "command_defs.lua" and look for the "Button_XX" commands. Way down below line 690 we find the definitions in these lines:

```
start_command   = 3000
device_commands =
        Button_1  = start_command + 1;
        Button_2  = start_command + 2;
        Button_3  = start_command + 3;
        Button_4  = start_command + 4;
Etc.
```

In layman's terms, we need to add the number XX in "Button_XX" to 3000. Thus, the numerical command ID for "Button_1" is 3001.

5.8.2.4 *Added code lines for "default.lua"*

Lines added to "default.lua" for use with my button box:

```
{down = 3001, up = 3001, value_down = 1.0, value_up = -1.0, cockpit_device_id = 12, name = _('Master arm 2-Pos ON/OFF'), category = _('Ins Weapons Status and Control Panel PUI-800')},
{down = 3006, up = 3006, value_down = 1.0, value_up = -1.0, cockpit_device_id = 12, name = _('Cannon round selector 2-Pos HE/API'), category = _('Ins Weapons Status and Control Panel PUI-800')},
{down = 3020, up = 3020, value_down = 1.0, value_up = 0.0, cockpit_device_id = 12, name = _('Cannon rate of fire 2-Pos LOW/HIGH'), category = _('Ins Weapons Status and Control Panel PUI-800')},
{down = 3001, up = 3001, value_down = 1, value_up = 0, cockpit_device_id = 11, name = _('Laser standby 2-Pos ON/OFF Switch'), category = _('Ins Targeting Mode Controls Panel PVR')},
{down = 3017, up = 3017, value_down = 1.0, value_up = -1.0, cockpit_device_id = 12, name = _('Automatic tracking/gun sight 2-Pos ON/OFF'), category = _('Ins Targeting Mode Controls Panel PVR')},
```

Lines added to "Throttle - HOTAS Warthog.lua" for use with the 3-position Autopilot switch (my button box doesn't have any 3-position switch):

```
{down = 3004, up = 3004, value_down = 0.2, value_up = 0.1, cockpit_device_id = 12, name = _('Weapon mode Burst Length 3-Pos HIGH/MEDIUM'), category = _('Ins Weapons Status and Control Panel PUI-800')},
{down = 3004, up = 3004, value_down = 0.0, value_up = 0.1, cockpit_device_id = 12, name = _('Weapon mode Burst Length 3-Pos LOW/MEDIUM'), category = _('Ins Weapons Status and Control Panel PUI-800')},
```

You might be surprised that two totally different commands have the same numerical command ID 3001. But this is no problem, because the commands are for different devices. For every device, the command numbering starts with 1.