Introducing

LINDA

Your controls!

by Artem Crum und Günter Steiner

written by brian747

Table of Contents

Introduction

This document is a supplement to the LINDA manual (which is v1.01, at the time of writing). It aims to give a little more background information about LINDA, as well as attempting to answer various questions which the writer has seen posted on numerous Internet fora, and to lead those new to LINDA through the process of getting to know the program and beginning to benefit from using it.

Nonetheless, the LINDA manual itself remains the authoritative source of information, and amongst its many illustrations and explanations you will find the various processes described in this document expressed in a more succinct form, which is more accessible for those already familiar with the program.

The writer would like to express his grateful thanks to Artem and Günter for LINDA, and also for patiently answering the questions which arose during the writing process, but most of all for making available to the flight simming community such a masterpiece of software engineering. Thanks to LINDA, we can now harness the power of Lua scripting to link our hardware to many controls in fsx add-on aircraft which were previously unavailable to us.

From the contents list on the previous page, you will see that the overall structure of this introductory document is straightforward: it begins with a review of what LINDA is and what it can do for you, and then moves via the installation process to look at the various sorts of button assignment which are facilitated by the program. Finally, it covers the topic of constructing simple Lua functions of your own, with a few illustrative examples.

However, please be aware from the outset that this manual does not attempt to explore in depth the art of tracing and exposing functions within the add-on aircraft of your choice, since that topic (which is as much an art as a science) is covered on pp 18 – 24 of the original manual: nor will you find here any kind of reference for the Lua language, since there are a number of those available online [1]. Instead, I will attempt to build on the official documentation by illustrating, with examples, how it is possible to use LINDA with the many functions that have been made available in the first year of LINDA's life for a variety of aircraft. I will proceed at a relatively gentle beginner-friendly pace to do so, and if some related issues are also clarified along the way, then that will be good, too

My most grateful thanks to all who have contributed to the LINDA module collection, but principally to Artem and Günter, whose brainchild has so greatly enriched the flight simulation scene.

brian747

Hampshire, England.

Version 1.0, Summer 2012.
Version 1.01 (this version) corrects some illustration elements on page 28.

[1] For example: http://www.anadrac.com/FSUIPC-Lua-Tutorial/index.html

What can LINDA do for me? (Beginners start here…)

Unsurprisingly, in view of its nature, one of the most popular questions asked about LINDA is simply "What is it?", since at first sight it may not be immediately obvious what LINDA is actually *for*. This section aims to answer that question at the outset, rather than expecting the reader to plough through many pages of text in order to discover what LINDA is all about. Let's begin by stating the most basic facts that you need to know:

- LINDA enables the connection of flight simulation hardware to the controls of fsx add-on and default aircraft, but in an extended way that makes available the use of controls which were not previously accessible for assignment. Which is great — but if you don't have fsx, or you haven't any hardware (what, not even a joystick with a few spare buttons?), then LINDA can't help you.

- LINDA provides a superset of the functionality offered by FSUIPC [2], so it's probably fair to say that if you're not sure what FSUIPC is, then your learning curve is probably going to be a little bit steeper than would otherwise be the case. More importantly, though, LINDA builds upon FSUIPC's fairly recent ability [3] to make an Lua scripting interface available to the simmer: hence — since Lua is only available with the licensed (paid) version of FSUIPC — you will also need Pete Dowson's licensed software (and a relatively recent version of it).

If you're still reading this, then hopefully that means you are still interested and so we can get on with telling you the good news about LINDA, starting with some relevant background information for those who may be unaware of it. 

"LINDA"?

LINDA collects together a number of useful tools to assist flight simmers, with the result that a suitably descriptive name was not easy to find. Finally, the authors came up with the now-famous acronym —

Lua **I**ntegrated **N**on-complex **D**evice **A**ssigning

……which summarises in the minimum number of words what LINDA is basically about, but also offers little hint of the rich treasure trove which is contained within it. This document therefore aims to discuss the available functionality in a little more detail than the LINDA manual itself, as a way of assisting those who are meeting LINDA for the first time to get up to speed.

Nonetheless, a year on from the initial release date it is already clear that no single document will ever be able to keep up with LINDA's rapid evolution, hence the reader is encouraged to visit the forum at http://forum.avsim.net/forum/424-linda/ for the latest news and updates.

---

[2] The FSUIPC support forum is at http://forum.simflight.com/forum/30-fsuipc-support-pete-dowson-modules/

[3] It is worth underlining the fact that you therefore need an ***up to date version*** of FSUIPC — v4.831 is current at the time of writing (and the updates are free).

Your own ‘pit

This introductory section of the document consists mostly of general information to place the need for LINDA in some sort of context. If you’re already aware of some parts of what follows, please feel free to skip through any aspects which are familiar to you.

Although fsx went to considerable lengths to make available as many controls as possible using your computer’s keyboard, it’s something of a pain to have to remember so many convoluted keystrokes. Even more importantly, though, trying to “fly” using the arrow keys to control the aircraft is something that makes life unnecessarily difficult, if not downright impossible, and so most people use either a joystick or else (space permitting, for non-military aircraft — with the possible exception of the Airbus) a yoke. Often, these devices have an additional control axis that can be used to operate the throttle, too, which is also more intuitive than using the keyboard.

So, quite early on, the flight simming community came to realise that using add-on hardware to control their aircraft instead of the keyboard brought a whole new dimension of enjoyment to the hobby. It’s now possible to buy everything up to and including fully motorised throttle units that are replicas of the ones in the real aircraft (or even a complete cockpit if you are very rich), and many people around the world have taken the ultimate step of building their own replica cockpit (or ‘pit, for short) in their home: if you need further evidence of this, simply consult your favourite Internet search engine!

For most of us, however, a replica full-sized cockpit will forever remain just a dream — unless a Lottery win should supervene — so we have to make do as best we can with the lower-priced hardware that is available from companies such as Saitek and VRinsight. These days, all but the most basic joysticks build in a number of buttons, along with the obvious controls for ailerons and elevators, rudder, and perhaps throttle. They normally provide a means to allow these buttons to generate various keystrokes too; so if you can somehow make a button generate a G keystroke, then in fsx you can use it to toggle the landing gear.

But supposing you want to assign a button to do something for which no keyboard shortcut exists? Or what happens if you need a single button to carry out several functions in sequence, such as bringing all throttles to idle and then engaging reverse, and (on releasing the same button) bringing the throttles back to idle again? In either case you are likely to find that neither fsx nor your chosen add-on aircraft is able to help you directly.

Which is where FSUIPC v4 comes in, of course. Over many years Pete Dowson has produced and evolved an enormously useful piece of software which can help you in many ways (such as smoothing fsx’s sometimes violently erratic wind changes, for example), and as time has gone by he has steadily added to the repertoire of functions that FSUIPC provides. As mentioned above, FSUIPC now provides us with the ability to utilise Lua scripting — which is what made LINDA possible (thank you, Pete!), and what gives LINDA the power to add so enormously to the possible and achievable functions we are able to utilise. Best of all, we can design and implement our own functions as required (I will give some examples later in this document), since all that is needed is to learn the straightforward syntax of a language called Lua.

So who needs Lua?

Lua is a fairly simple language which readily deploys libraries and is generous in its affinity with extensions, with the result that it's ideal for use in a simming context (not least since it's free). But DON'T PANIC — you don't have to become an expert Lua programmer (although you may later decide to do so, of course) since Pete Dowson, Artem, and Günter have already done that for you. In fact, if you use Günter and Artem's already-available modules for a variety of popular add-on aircraft, you don't need to know much about Lua other than the fact that it exists.

But if you do want to 'roll your own' functions for your latest add-on aircraft of choice, all you will need to do is learn how to put together a few very simple elements and you can create your own functions and assign them to your hardware, once you have discovered how to flip the correct switch or whatever in that particular aircraft. And to help you do just that, LINDA also provides its own 'Developer mode', enabling you to trace the working of the switch you wish to move in a way which has never been possible before, other than for skilled programmers.

Too good to be true? Not at all, read on — or if you really can't wait, have a look at the main LINDA documentation, starting on page 18 (in v1.01).

To utilise LINDA without any programming, though, all you need to do is to download and install the latest version of LINDA itself (which is v1.11 at the time of writing, available from http://fs-linda.com/downloads/LINDA-v1.11.zip), and then add the module or modules you need for your aircraft. Have a look at the Downloads section of the forum at http://forum.avsim.net/forum/427-linda-downloads and then install what you need: thanks to the patient work of Günter and others, and their generosity in sharing it with the simming community, many add-on aircraft are now supported.

An illustrated description of how to install LINDA will be found below: but now that you understand the need for Lua let's consider the devices that LINDA can — and can't — work with.

What devices can LINDA work with?

Perhaps the best way to summarise the answer to this question, following the sequence used above, is that LINDA cannot be used for assigning keystrokes from your PC's keyboard [4], nor can it be used for assigning axes [5]. There's a very simple reason for that: FSUIPC has for a great many years been perfecting methods to help you deal with axes and keystrokes, so there is no need whatsoever for LINDA to cover the same ground.

Those who know FSUIPC will be saying to themselves at this point, "But FSUIPC can deal with buttons as well, so what does LINDA bring to the party?". The answer lies in the example I gave above about a button for engaging idle then reverse thrust when pushed and also idle thrust on release — FSUIPC can be persuaded to do this, but only after using various techniques explained in the Advanced manual, including editing the fsuipc.ini file directly — something which many people hesitate to do (or even decline to do). It was because Pete Dowson recognised that he had reached the limit of what was possible in such ways that he added the Lua scripting ability, since much more complicated tasks can now be done using that capable programming language, thus removing the need for editing sensitive initialisation files whilst also vastly expanding the possibilities of what can be achieved.

LINDA currently recognises several different sorts of device (more details will be given on page 16):

- Ordinary joysticks
- HOTAS joysticks
- Flight yokes
- Throttle quadrants
- "Other"

LINDA does its best to automatically detect which sort of device you have connected, but it is good practice to check the category it has assigned so that you can adjust it if necessary, and also give it an appropriate short name. (More about that when we get to the Installation section).

But perhaps LINDA's best-known claim to fame lies in its ability to interface with the VRinsight MCP Combo, both version I (which at the time of writing is no longer available) and also version II. The original SerialFP2 software provided by VRinsight suffered from many limitations, but happily LINDA completely replaces it, giving much increased functionality (and supporting a much wider range of up-to-date aircraft, in addition to the ones that come with fsx).

A HID — what's that?

If you take a look within the Device Manager of your Windows installation, you will find a category called Human Interface Devices (or HIDs, for short). To explain it as concisely as possible, let's simply say that these devices came into being as a way to enable input devices used by humans as controls to be readily connected to a computer, latterly via a versatile plug-in interface which we now know as USB. The original devices used with this genre are of course keyboards and mice, which have obviously been around for a long time, and hence those two are nowadays often recognised by the Operating System without the need for additional drivers [6].

---

[4] Although see "Record keypress" on p. 28

[5] An axis, in simming terms, is usually a lever of some kind, or perhaps a sliding or rotary potentiometer control. Unlike buttons and switches which are either on or off, axes return a range of values.

[6] Those with long memories may recall that early keyboards and mice (and joysticks) were for a while connected via serial ports, but USB quickly took over as soon as it became available.

Reference has already been made to the fact that attempting to control an aircraft using a keyboard and a mouse is very far from being the best way to do it, and so it was that joysticks and yokes soon came along, clearly in the category of HID devices, and (like mice and keyboards) quickly came to be connected via USB. Unsurprisingly, therefore, although flight sim control devices became increasingly varied and complex, they also continued to be identified to the Operating System as HIDs, which is why you may observe that very often the calibration procedures native to Windows use a similar presentation for each of them.

So please be aware that some simming control devices may also be referred to as HIDs from time to time; and that occasionally you may need to check their operation using the Windows HID routines [7].

Very complex devices such as the VRinsight MCP Combo units are not technically members of this category — they are rendered accessible once again owing to the tireless efforts of Pete Dowson [8].

How LINDA began

It is sometimes said that to understand anything, you also need to understand the history of how it came to be. LINDA is the product of a happy marriage of expertise between the two experts, Günter Steiner and Artem Crum.

Günter's speciality was originally the production of Lua scripts for various add-ons (which he posted in the FSUIPC forum). However, these modules needed to be able to reach into the code of the add-on concerned in order to be able to actuate some control — which was often otherwise unavailable for use by simmers. Günter therefore became adept at searching for those parts of the code which could be accessed to produce the desired effect, although this clearly took considerable effort and expertise.

However, in early 2011, Artem provided him with a rather special set of Lua scripts which set up a standard for those to come, as well as providing unheard-of functionality for the VRinsight MCP Combo [9]. The combined system was further refined between the two authors as development continued, and so Artem decided to add a GUI "front end". This not only provided an attractive and easy-to use interface to the LINDA functionality, but also added a Tracer which greatly simplified the work involved in discovering addresses and offsets within the code of fsx's many add-on aircraft, thereby increasing Günter's productivity [10] (as well as that of others).

So, as Günter puts it: "LINDA is a tool to make gathering and assigning Lua Variables and controls a bit easier, and should be both a platform and standard for upcoming Lua scripts." (Although we should also notice that the provision of all that functionality was made possible by Pete Dowson's provision of Lua scripting, as an extension via FSUIPC).

We simmers owe all three men a very considerable vote of thanks for their superb contributions to our hobby / obsession.

---

[7] See, for example, the discussion on page 11 about connecting the VRinsight TT Panel to fsx.

[8] Thank you, Günter, for the clarification. 

[9] For a quick look at some of the functionality available, take a glance at Appendix 4.

[10] Using the LINDA Tracer, it was possible to create the first NGX module less than 24 hours after the release of the NGX!

LINDA — installation and setup

Hardware used in the preliminary examples

Just before I begin, a few words on the examples that I will use in this next section.

One of the problems that we are faced with as simmers is the huge variety of hardware and software which is available, resulting in millions (or perhaps billions) of combinations of hardware possibilities (processor, graphics card, amount of memory, joysticks, etc) multiplied by the even larger number of possible combinations of software in use. So the chances of finding anyone with exactly the same configuration as yours or mine are vanishingly small — which also means that giving examples which are useful to the maximum number of people becomes more difficult than it may appear.

So I will be using four hardware devices in the initial examples in this document. I will now quickly describe their functions rather than what they look like, since their functionality is the significant factor in interfacing them to fsx. Bear in mind that although for completeness I mention the number of axes available on two of them, LINDA is *not concerned with axes*: those need to be programmed in FSUIPC (or perhaps the Saitek or other manufacturer's software, if you absolutely insist).

1. The VRinsight TT Panel

This consists of a number of buttons and switches, mounted in a sturdy metal frame. Each button or switch can be either pushed (or clicked on, in the case of switches) or released (turned off). There are four of the buttons which are internally configured to behave like a hat, but otherwise the device is about as simple as you can possibly get. (Like devices 2 and 3 it is still a HID, of course).

2. The Saitek throttle quadrant

This consists of three throttle levers, each of which has a switch incorporated which can be operated by moving the lever past the lower part of its travel (clearly intended to operate the throttle reversers). In addition, there is a two-way (up/down) rocker below each lever, giving a further six buttons which can be programmed with the device. So as well as nine buttons, here we have added three axes to the picture.

3. The Saitek X52 Pro HOTAS

This "Hands on Throttle And Stick" device represents a further step-up in complexity, consisting as it does of a joystick with multiple hats, buttons and switches together with an integrated twist action axis for the rudder making three axes on the stick, allied with a separate throttle control which also possesses a number of additional buttons and axes — and even a simulated mouse.

4. The VRinsight MCP Combo I

This device is one of the most complex available, since it has numerous buttons and switches, together with some rotary controls and also displays which can be used (with the benefit of the authors' programming) to show information from fsx. Be aware that although the rotary controls here may appear similar to axes, in fact internally they are something else entirely — since they work on encoders using phase differences to pass information about which direction they are being turned, and at what speed. But happily you don't need to worry about the technicalities, since they are entirely taken care of for you by LINDA.

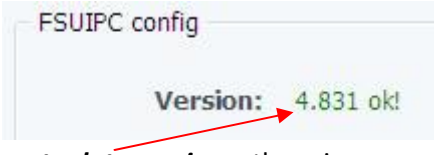
So between them those devices cover a large part of the spectrum of complexity in terms of things that you might want to use to control your aircraft, so that hopefully the examples given will therefore enable you to connect any similar but different devices that you may have, using LINDA.

But first of all, we need to install LINDA. This process is covered in the LINDA manual, of course, and those who are confident of their abilities may wish to use the shorter explanations to be found there. In this document, however, we take a more leisurely approach which hopefully will be helpful for those with less experience.

Installation prerequisites 1: *Back up your Modules folder now!*

You are about to make a number of changes to your Modules folder [11] and the folders beneath it. It is therefore vital to back up those folders as they currently are, by copying them away to somewhere safe, which also means outside the fsx directory tree. Then, should you by any chance need to start from scratch, you can copy them back to return to the point where you currently are, and try again.

Installation prerequisites 2: the latest, licensed, FSUIPC



As already mentioned, you will need a licensed (paid for) version of FSUIPC v4, since otherwise you do not have access to the Lua facilities upon which LINDA depends. More than that, though, I must stress that you need to ***ensure that you have an up to date version***, otherwise you may experience a number of ill effects. LINDA does try to check for a totally unsuitable version during the installation process, but I would recommend that in any event you simply go and download the latest FSUIPC from Pete Dowson's site and forum, and then bring yourself up to date with the latest available version:—

I am always uneasy about giving links [12], since these inevitably change, but at the time of writing you should first of all visit the following page to download the current major version:—

http://www.schiratti.com/dowson.html

Having done that, do not fail to visit these links, too, for the most recent updates:—

http://forum.simflight.com/topic/66139-updated-modules/

(Be careful, though: that page also contains links for v3 (fs9), as well as the v4 (fsx) modules).

And:

http://forum.simflight.com/topic/68257-latest-lua-package-for-fsuipc-and-wideclient/

Once you have confirmed that your version of FSUIPC is properly installed and fully up to date, you can proceed. The next task is not inherently complex, but there are various factors which may affect how much you may need to do, so I will make a couple of recommendations first for special cases.

---

[11] This is located below your fsx base directory (wherever that is), so that it might be, for example, C:\fsx\Modules.

[12] Although I have reluctantly done so on this page and at several points in this document: I can only apologise if they no longer work when you come to try them!

Installation prerequisites 3a: a recommendation for EZCA users

At this point I do need to put in a quick word for those readers who have installed EZCA (a.k.a. EZdok). You will already have been through a fair amount of pain when you had to remove the assignments for your numeric keypad and so on, so it may well be that you have already done a lot (or even all) of the work needed to ensure that no button assignments remain in fsx or FSUIPC. So I would simply suggest that you review LINDA's requirements and then check that your own installation is compatible with both EZCA and LINDA too [13].

Installation prerequisites 3b: a comment for PMDG NGX users

In pages 0.00.20 and 0.00.21 of PMDG's '*Introduction and Use*' manual for the NGX, Ryan makes a number of comments about the use of FSUIPC. Unfortunately, for copyright reasons I can't reproduce those comments here, but a significant aspect is that PMDG claim that there may be problems if you calibrate your "flight controllers" using FSUIPC, rather than the software provided by the manufacturer.

I have no wish to enter into a dispute about this, and clearly PMDG would not make such a recommendation lightly: all I can say is that I took the decision to continue to use FSUIPC to calibrate my rudder axis (with LINDA handling the buttons), and have experienced no problems or difficulties whatsoever during nine months of flying only the NGX. Furthermore, when a question was asked about PMDG's comment on the FSUIPC forum, Pete Dowson's reply was: "All that's talking about [is] deleting your settings and re-doing assignments and calibrations to suit the NGX, because it just might be that your own axis calibrations and assignments may not suit the way the NGX is set up. All you really need to do is create a new Profile or Aircraft-specific set."

Pete's comments seem entirely sensible (and consistent with my own experience): however, Your Mileage May Vary, as the saying goes, so do be aware, at least, of PMDG's recommendation. Certainly, many of the other recommendations made in the NGX *Introduction* document are extremely sound (the whole document is very well worth reading carefully), but *caveat emptor*.

Installation prerequisites 3c: removing button assignments from fsx

The reason for first mentioning EZCA and the NGX in this connection is that you do now need to ensure that there are no conflicts between fsx, FSUIPC, and LINDA — in other words, to ensure that no devices are assigned to more than one of them. In addition, if you use 'raw' Lua routines or FSUIPC macro files, you may wish to consider removing them from FSUIPC and using LINDA instead.

The LINDA manual is written for experts, and suggests deleting the FSUIPC4.ini file, so that it is recreated (without any customisations for buttons, keyboard, or axes) when FSUIPC next runs. This is certainly a very efficient way to remove all the button assignments, but if you have made any other customisations within the ini file, they will be obviously be lost also. So before following that advice do please review your ini file to make sure that nothing unexpected happens.

---

[13] Your key mappings are held in the Standard.xml file, of course, and operating on this directly rather than using the fsx interface can (as long as you know what you're doing) save you a lot of time. If you are nervous about doing this, or are unsure which Standard.xml is the "real" one, then allow me to suggest that you download and use TweakFS's free XML utility — http://tweakfs.com/download/fsx_xml_toolbox.zip

The LINDA manual also suggests editing the standard.xml file to remove all existing assignments (have a look at pages 4 and 5 (in v1.01 of the manual) to see the recommendations there). In 3a, above, and the associated footnote, I advocate the use of a utility for helping you to edit the file — which is fine if you are happy about doing so. But this approach might cause problems for those who are unused to the format of Standard.xml (or indeed, unused to editing configuration files in general), so although it will take longer (and is, I know, rather tedious) you may prefer to achieve the same result by simply going through your control assignments in fsx and removing all the assignments for your joystick's axes and buttons.

Having used whichever method suits you best, you will then be free to assign the axes and keystrokes in FSUIPC and the buttons (and VRinsight MCP Combo, if you have one) in LINDA, thus getting the best of both worlds.

Finally in your preparations, get yourself the current version of LINDA together with the aircraft modules you intend to use (make sure the aircraft modules are all compatible with your LINDA version), and you're ready to go. When the time comes (see p. 30), be sure to configure fsx's own built-in aircraft first, however, since they provide the basis for the other assignments.

But more of that anon — first let's look at the process of installing a simple HID device into Windows, ready for button assignment in LINDA.

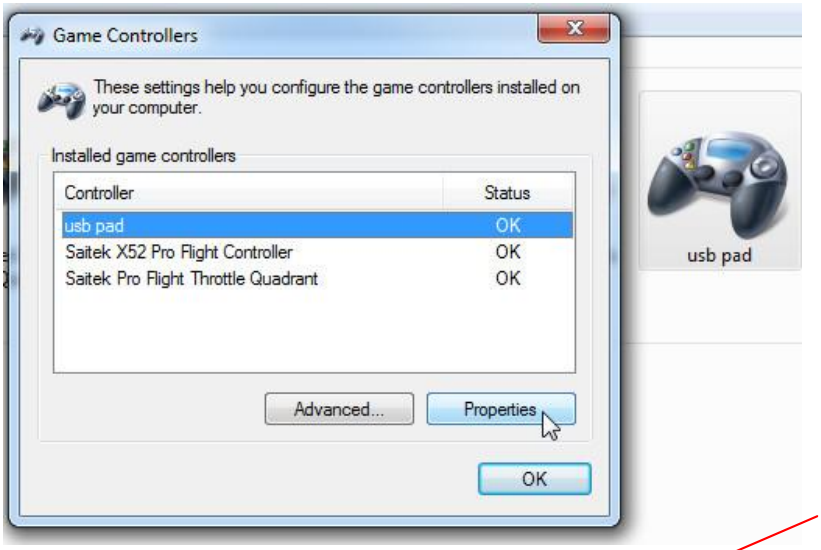
Installing / checking a simple HID device (in this case, the TT panel)

For simple devices such as the TT panel and basic joysticks, all that may be necessary is to check and configure them using the Windows facilities available to you (these examples use Windows 7, but XP is not wildly different). From the Start Orb, select Devices and Printers. A series of devices is displayed

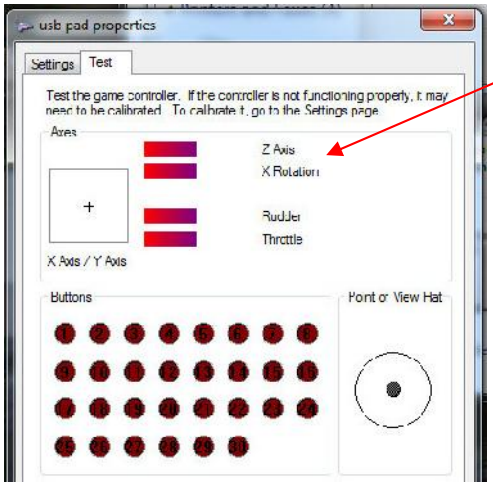
— notice that although Windows does its best to show an appropriate picture, it doesn't know anything about the TT panel, so it assigns it a name ("usb pad" d'uh….) and a picture (in this case, a generic hand-held device).

Right-click the device, and select "Game controller settings", as shown.

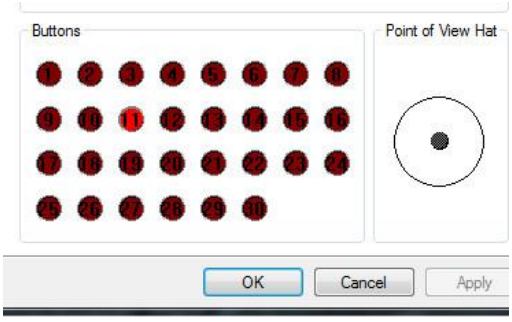
Then click the Properties button for your device.



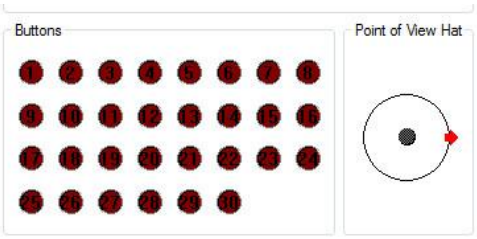
You now see a fairly generic properties page, as mentioned in the introduction. Even though the TT panel has no axes, for example, they are still present on the page, along with a number of buttons.

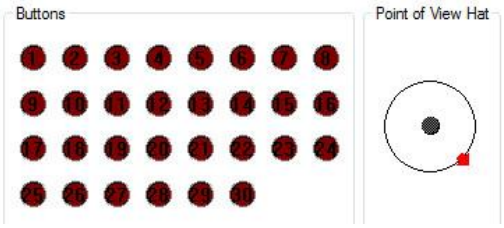


The actual assignment of the buttons is done in LINDA: what you are doing here is ascertaining that all the buttons are working, as well as what each one is called, as a cross-check. Here, for example, I am pressing button 11:



A peculiarity of the TT panel is that four of the buttons (labelled C2, C3, C4, and C5 on the unit itself) are configured internally as a hat switch, so that pressing C2, for example, generates a right movement of the virtual hat, as shown.





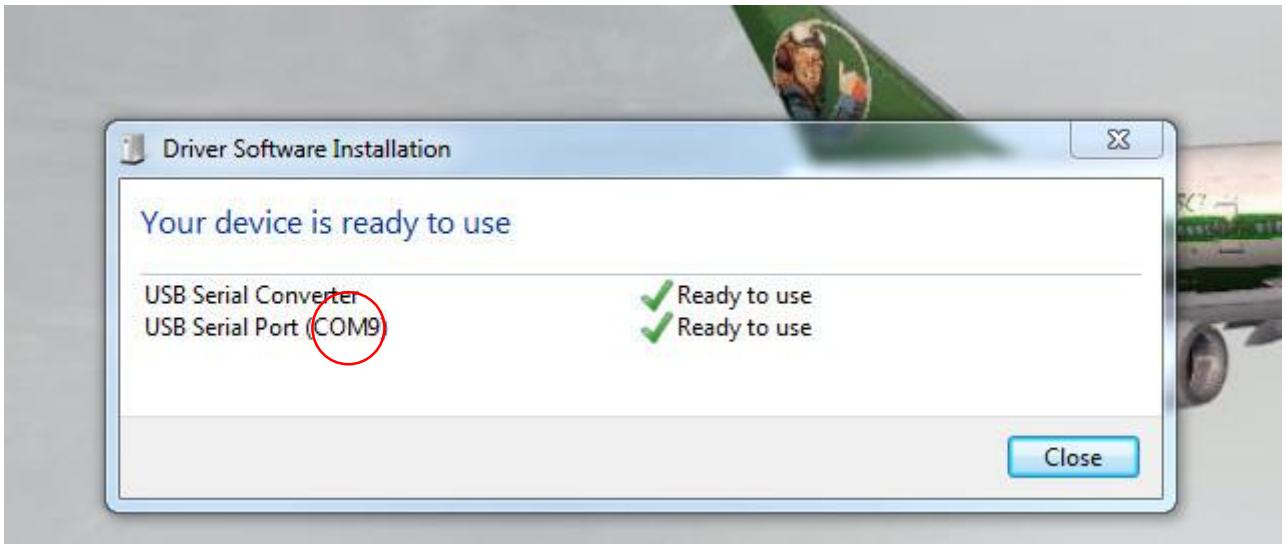
Not referred to in the documentation (not that I could find, at least) is the fact that pressing two adjacent buttons generates an intermediate position of the hat — so pressing C2 and C3 together results in the position shown to the left.

Having checked that your TT panel or other device is correctly connected, you are ready to assign some buttons using LINDA, as described on page 22. But first let's consider installing a much more advanced HID device, the VRinsight MCP Combo.

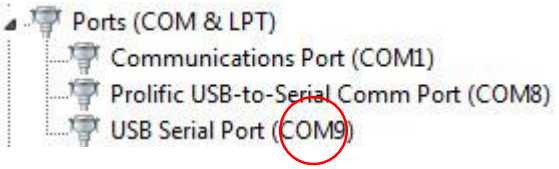
Installing your MCP Combo

If you are installing LINDA to use with a new MCP Combo I or II, then it could be helpful to install the Combo box first (if not, please skip to "Installing LINDA", below).

The important thing here is to ignore VRinsight's suggestion that you install their SerialFP software: LINDA makes it redundant, as well as outdated. You do, however, have to introduce your new hardware to the Operating System, so set up the power to the unit (I prefer always to use a separate power supply for significant devices such as this, rather than drawing what might be a relatively large amount of current from a USB port) and then plug the USB plug into the most convenient socket.



After a short pause, Windows should announce that it has seen and recognised the device (as above). Do **note the COM port number that it is using**, since you will need that information soon. (This is a *virtual* COM port which is set up for you automatically: nonetheless it is utilised by the O/S as though it is real — and in fact if you look at Device Manager you will see it listed under your COM and LPT ports section, which is another way



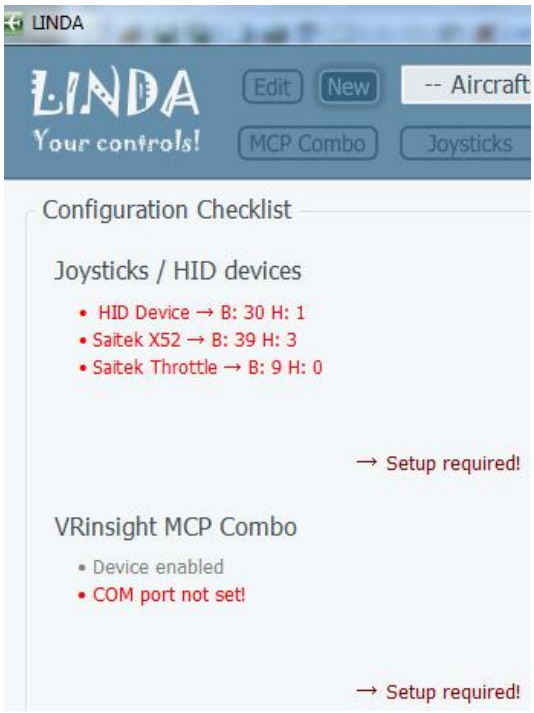
of finding the number of the COM port concerned. (Ignore the COM8 device shown in my ports list, which is used for another MCP unit which I also have installed. Which brings up an important point: in general, please be aware that Windows will assign the port number for your VRinsight MCP Combo according to its own logic, so that the port that is assigned will vary — many people finding that Windows assigns COM port 3 or 4, for example. It doesn't matter, as long as you know the number).

Installing LINDA

After the rather long discussion about your preliminary button and axis assignment setup, you will be relieved to hear that installing LINDA is simplicity itself. No installer is required: instead, all that is needed is to unzip the LINDA files to your fsx Modules folder, which means they will automatically be installed in the correct place in the directory tree [14].

If, however, you are uncertain about unzipping files to a specific location, or you simply want to be 100% sure of what is happening, you can unzip the LINDA files to a newly-created and empty directory of your choice, and check to see what you have. Once you are satisfied that you know exactly what files need to be copied where, you can then simply use Windows Explorer (not *Internet* Explorer, the other one) to copy or move the relevant files to the Modules folder. If, below the Modules folder, you now have one folder called linda and another called linda-cfg, all should be well.

Configuring LINDA: the 'joysticks'



Start LINDA by running LINDA.exe. Since this is a very friendly program, it leads you through the process of making the preliminary checks and setting up whatever is required.

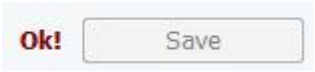
As you can see from the illustration on the left, LINDA has detected all four of the devices which I have connected, and displayed them in red to indicate that some setup is needed. If you are really observant, you will also notice that it gives the number of Buttons and Hats that Windows reports in each case — except for the Combo, since until the COM port has been specified it cannot communicate with it.

It may be worth stressing that this information is *what Windows reports,* which does not always correspond to reality: for example, the "HID device" in fact has 28 buttons / switches available, with four other buttons configured as the hat. (That device is actually the TT panel, but it is here initially given its Windows-assigned name of "usb pad", so I will need to configure that with a more meaningful name quite soon).

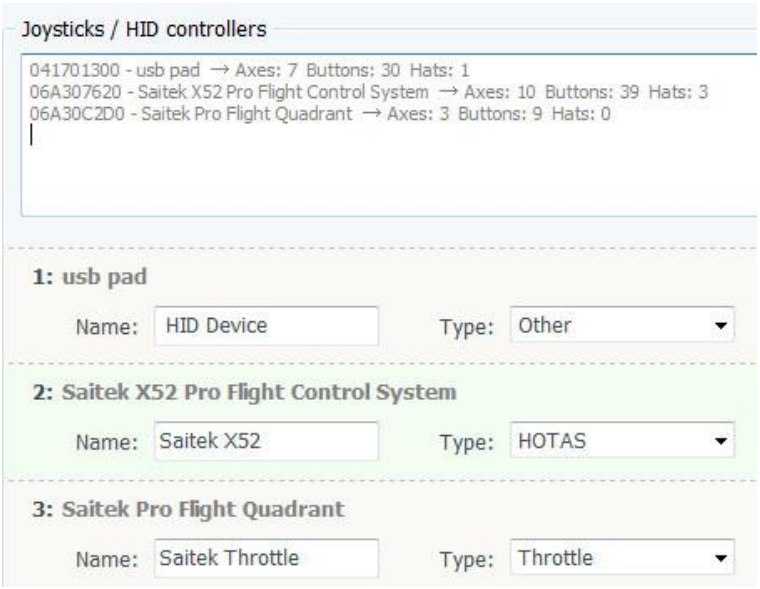
[14] Hopefully, you have complied with the well-established recommendation **NOT** to install fsx in the default location, since with Windows 7 this can result in UAC messages and other problems — with LINDA and also with many other add-ons. My fsx installation is in C:\fsx, which also shortens the associated path names.

Following LINDA's suggested order of business, I set up the 'joysticks' first.

As you can see, LINDA has accurately identified the Type of each device, so all that was necessary was to save the configuration.



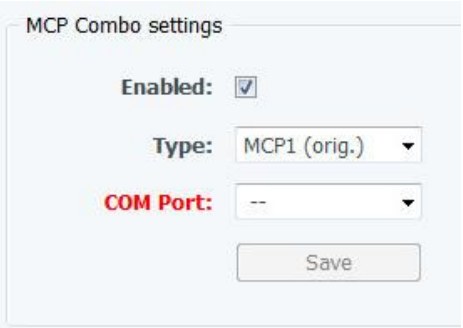
(Oops: you may notice that I forgot to change the name of the "usb pad" before saving, but I will correct that subsequently).

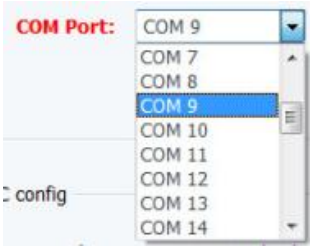


Configuring LINDA: the MCP Combo and FSUIPC

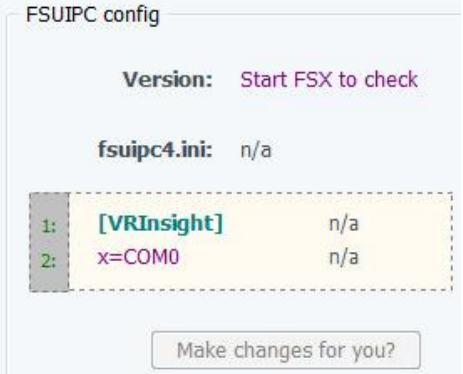
(If you don't have the MCP combo, please skip the next bit and go straight to the paragraph about FSUIPC, overleaf).

Again, LINDA makes it very easy for you. Simply use the drop-down box to set the Type to MCP1 (the original Combo I) or MCP2, for those lucky enough to have waited for the newer version.





Then comes the important task of specifying the COM port. Again, use the drop-down box to select whatever value Windows decided upon, as discussed above.



Your Combo box is now configured.

FSUIPC configuration is also straightforward. As referred to above, if fsx is running then LINDA will determine the FSUIPC version (and warn you if it is far too old to work properly with the program). But there is one further task needed.

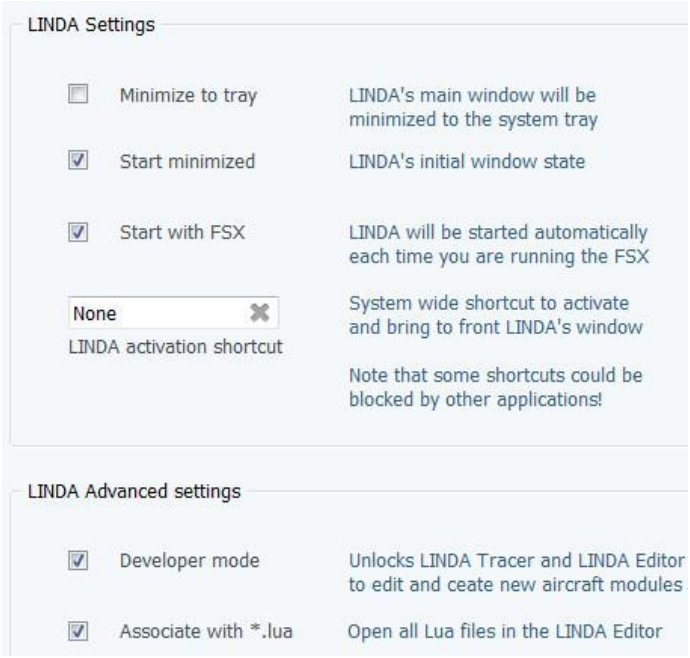
FSUIPC has its own configuration file, called fsuipc4.ini, which sits in the Modules folder. In order for the Combo to work, two lines need to be added to this file. You could edit the file and add them yourself if you wish, but LINDA is ready and willing to do it for you. Once connected to fsx, LINDA determines whether the two lines are present: if not, the program indicates the correct format for the two required lines, and if you click on the "Make

changes for you?” button, they will be added automatically: the angry red colour is replaced by a tranquil green, and the message that all is now well.

(If you already had the necessary lines in place, it would have looked like that from the beginning, of course).

Configuring LINDA: setup options

The third and final stage of the configuration process is to select which options you wish LINDA to use.



The settings which I used are as shown. Please note that unless you check the "Developer mode" option, you will be unable to make use of the LINDA Tracer facility to discover useful and usable characteristics of your add-on aircrafts' internals.

Similarly, if you would like to use LINDA's friendly and colour syntax-highlighted facilities to edit and develop your Lua code (see p. 25), you will need to check the "Associate with *.lua" box.

You can set up a shortcut to bring LINDA's window to the front, however I opted not to do so, since I usually run fsx in full screen mode.

And that's it! LINDA is now fully configured.

We will now move on to configuring your buttons (as found in your joystick and/or throttle (or HOTAS) and/or other device such as a TT panel and/or an MCP Combo). Before doing so, however, you may wish to assign any axes you have, using the facilities within FSUIPC. This topic being outside the remit of this document, may I refer the reader to Pete Dowson's own manuals, especially the User Manual and the Advanced manual — although you may also wish to become familiar with some of the other documentation, notably the Lua and Library documents.

So having configured the axes with FSUIPC (bearing in mind the caveats mentioned in 3a and 3b, above), you can then move on to configuring the buttons on your various devices, using LINDA. First, choose the device you are going to configure by clicking on the appropriate icon (see the examples below).

Using fsx routines and supplied aircraft profiles

Having configured your controls (HID devices) so that they are recognised by both Windows and LINDA, and assigned your axes within FSUIPC, you are now ready to assign your buttons and other controls to various functions within the aircraft in your hangar. Before moving on to the more complex add-on aircraft, however, it is important to note that you should set up any "vanilla" actions for fsx's default aircraft first.

The LINDA v1.01 manual explains this clearly —

> **Default actions**
>
> "The functions you assign for the FSX Default aircraft will be used as the default actions for all other aircraft — unless you reassign the same buttons/switches to other functions. The logic is similar to that of FSUIPC — you have default assignments, but can override them for specific aircraft if need be. You can see that there is an assigned default when it is shown with a little star and coloured light green".
>
> "We strongly recommend that you select and assign the FSX default planes first of all.
> The assignments you make for the FSX standard planes are the defaults for all other add-ons aircraft.
> e.g. If you assign 'LIGHTS Strobe' for the FSX standard planes, then 'LIGHTS Strobe' will also be available for your add-on aircraft — providing you do not set it to 'do nothing', or overwrite it with another function".

So if you have hardware that you want to operate the same functions throughout your whole collection of aircraft, then assign them to a default plane and you won't have to reassign the same command every time.

But there is something else you may wish to bear in mind……

When it comes to discussing assigning controls for an add-on aircraft I am going to take as my example the most complex one available at the time of writing, the PMDG "NGX" (the Boeing 737 – 600, -700, -800, and -900 models). That aircraft has a beautifully-rendered virtual cockpit that is a joy to use, and approaches reality very closely. It follows, therefore, that some of the standard fsx functions will be of little use in that context. In fact, looking down my list of controls that I have assigned through LINDA, there are very few which would be relevant to the default aircraft.

That isn't a problem for me since I fly a relatively restricted range of complex aircraft (and *never* any of the fsx default aircraft), and so I'm perfectly happy to design each control set individually. But if you do regularly fly the fsx default planes then you may need to do a bit of thinking about which controls you want as standard.

For example, your list might include a switch that you decide to use for your undercarriage, up for gear up, and down for gear down. That's fine for the simplified fsx aircraft, of course — but what happens when you fly the more complex ones where there's a 'gear off' position (to depressurise the

hydraulics) between gear up and gear down? As I said, some preliminary planning at this stage will certainly save you a lot of revision later. (I'll return to the planning process in a few moments).

Hardware usage in the examples for the rest of this document

I alluded earlier to the difficulties attendant upon writing a guide such as this bearing in mind the millions of different combinations of hardware and software which are in use by simmers all over the globe. So now that we are approaching the part of the manual where the practicalities of assigning controls on your own hardware to the controls in the cockpit are under discussion, in order to meet the needs of as many people as possible I have decided to use for the rest of the document not only a complex add-on aircraft (the NGX) but also a more complex set of controls (the Thrustmaster Warthog HOTAS combination in place of the Saitek yoke and throttles), along with the VRinsight MCP and a simple Logitech stick. This makes very little difference to the examples in fact, and indeed I have avoided using the complex devices in my examples wherever possible, with the exception of one question which I have been asked several times, and so will be specifically covered as I go along.

My reasons for changing are that (i) in general, those who need LINDA are those who will be using rather more than a basic joystick anyway, and (ii) that whilst it is straightforward to simplify a complex example, the reverse is not the case — so if I gave only relatively simple examples that would inevitably leave many questions unanswered for those with more complex hardware. So with apologies to those with Saitek kit, I will switch to the Thrustmaster stick and throttle unit for the rest of the description, although I will add to the mix an old Logitech stick that I occasionally use for additional buttons or axes as well.

Unfortunately, I am one of the many simmers whose home 'pit is too small to allow the use of a yoke (or pedals) and so I have to use a stick for flying airliners. [15] However, since the Thrustmaster Warthog (or 'hog, for short) offers a range of different and perhaps unusual controls which can be adapted with advantage — for example it has several three-position switches that are ideally suited to tasks such as being assigned to the undercarriage with a central off position — such features will enable me to make suggestions which can hopefully be adapted to a variety of other kit, too.

Designing your set of control assignments to suit your hardware

This document is about LINDA, hence I will not dwell overlong on this aspect of your planning (important though it undoubtedly is), but before you proceed I would strongly recommend at the very least making a list of all the switches which are present on your available control hardware; making a second list containing all the commands that you wish to assign; and then coming up with the best possible list of assignments for your available switches.

Believe me, it is time that is hugely well spent, and time that will save you a lot of the pain associated with reassigning things later on, at which point each reassignment causes a domino effect whereby lots of other things consequently need to be reassigned too.

It is usually possible to browse the Internet and download a picture of your control device — if not, take a photo and use that. But you will find it helpful to label each control visually, as part of your documentation. (An example page from my own documentation for using the 'hog with the NGX is given in Appendix 2, on page 46).

[15] Which isn't a problem for Airbus drivers, of course.

If you are using a LINDA module for a specific add-on aircraft, it is also helpful to also note the function name you intend to associate with that switch movement. But, for example, suppose you wanted to assign a switch to turning on the APU bleed air, what routine do you need?

Your identification problem is exacerbated by the fact that, owing to the enormous and dedicated efforts of Günter and others, no less than 700 functions for the NGX are available for your use! So you need to search for the one that will do the job. Happily, they are all listed within the NGX module 1.7, which is helpfully commented, so it is not difficult to find the function you want. I have also provided a sorted list of NGX module v1.7 functions in Appendix 1, and so you should be able to quickly locate the routine you would need to call, which in this case is `function NGX_APUBLEED_on ()`.[16]

Three positions for the gear stick

Exactly how you assign that or any other function to your hardware will be covered very soon, but first allow me to give you a specific example (which also answers a question that I have been asked numerous times) — how do you program a three-position switch to faithfully emulate the gear lever?

The Thrustmaster HOTAS has several three-position switches, but the one I opted to use for this particular function is the one conveniently situated on the front right of the throttle base. There are two aspects to making the decision: the first is the obvious one about the control's suitability (so for this purpose three positions were clearly essential), but the second concerns the control's accessibility *in the light of other things going on at the time* (this switch's position means that it is easy to reach out, locate, and flip it from the down position to the up position whilst still flying the aircraft in the early stages of takeoff).

Electrically speaking, switch 28 is on when the switch is 'down' (forward), switch 27 is on when it is 'up' (back) and when it is in the middle position both switches are off. How is this used with LINDA to emulate the gear lever?

Consider the sequence of events in practice. Until takeoff, the switch is in the down position (sw 28 on, 27 off). Once a positive rate of climb is established, the switch is moved to the up position (sw 28 off, sw 27 on), and when the flaps are retracted the switch is returned to the centre position (both off). On the approach, the switch is returned to the down position (sw 28 on). Using the On Press and On Release columns for these two switches in LINDA, the programming is therefore as follows:—

27	NGX GEAR up	empty	NGX GEAR off
28	NGX GEAR down	empty	empty

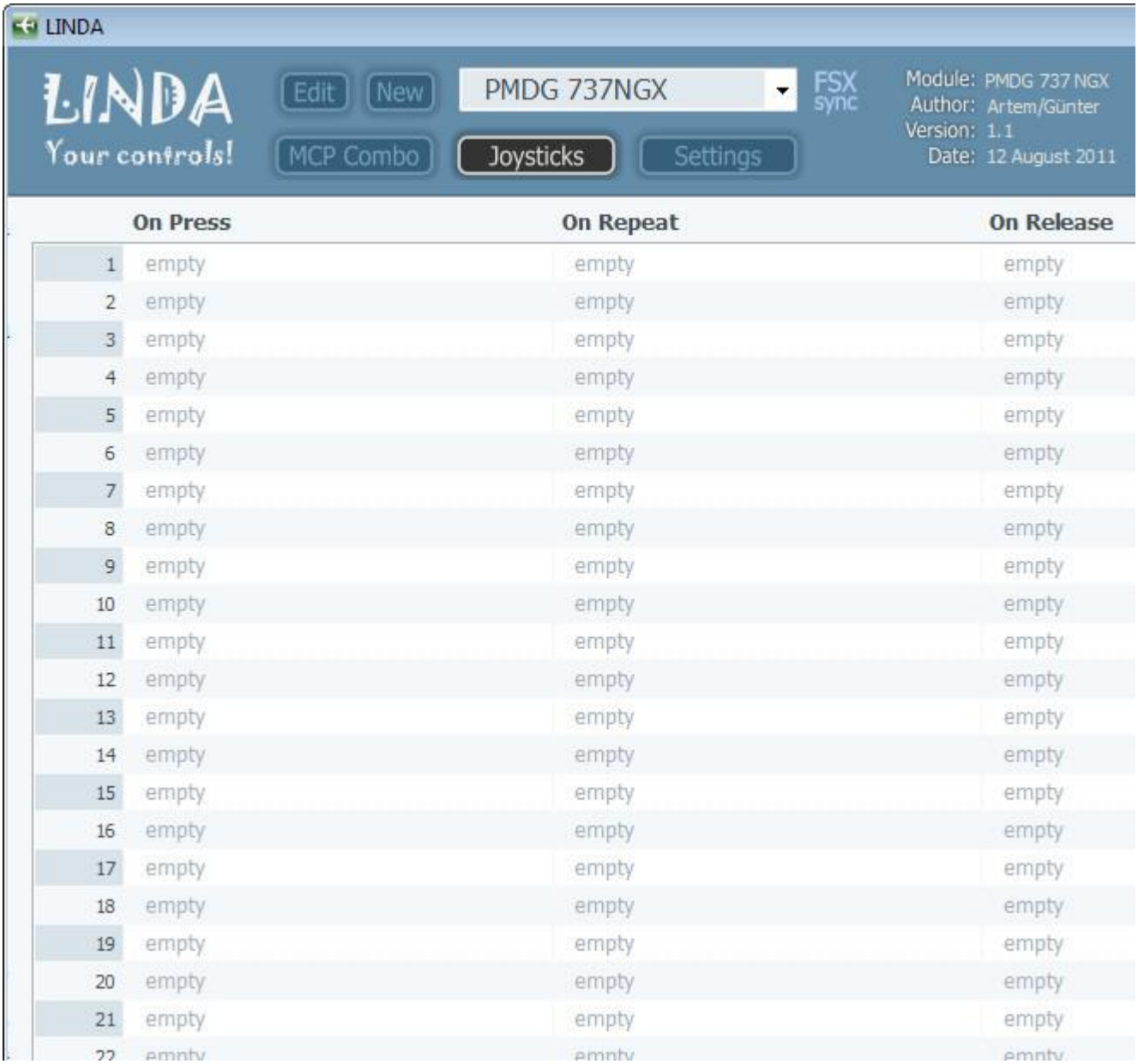
So when the switch is moved to 'up' on leaving the runway, LINDA looks at On Release for sw 28 (which is empty, so nothing happens) and then On Press for sw 27, whereupon it triggers the NGX gear up function. When the switch is moved to the centre position, LINDA actuates the On Release for sw 27, which calls the NGX's gear off function. And when the switch is moved to the down position again for landing, LINDA calls the NGX's gear down function (On Press for sw 28). Which happens to be exactly what we wanted.

The careful design of all your control assignments is vital to a pleasurable experience in the cockpit. I promise you!

But let's now look in general at the overall process of assigning commands to buttons, using LINDA.

[16] Caution: Lua is case-sensitive, so you would be well-advised to copy and paste the name from the original code, to be sure of not making a mistake. I'll mention this again later, since it's important.

LINDA's hardware support — assigning buttons and switches



If you have no current assignments for the device being configured, the Joysticks screen will look similar to the above. As you can see, you can define actions for three different events for each button — On Press and On Release are self-explanatory, whilst On Repeat is triggered whilst the button is held down (or in the case of a toggle switch, whilst the switch is set to the On position).

There is also (top right, although not shown in the above illustration) a check box entitled Shifted: this determines whether the functions on the page apply to the button in its normal (unshifted) state, or when one of the two LINDA shift functions are in force (which thus doubles the available number of buttons!).

The LOCAL SHIFT function applies the shifted property to this joystick only.

The GLOBAL SHIFT function affects all joysticks.

Button assignment basics

The way of assigning functions to buttons in LINDA has been made breathtakingly simple for us (thank you, Artem!). Let's suppose, for example, that you wish to assign a button to incrementing (lowering, increasing) your flap setting. Press the button you want to use whilst watching the list of button numbers on the left of the LINDA page — your button press will be detected, and the appropriate button number [17] highlighted. Also, the appropriate device will be highlighted, in the lower part of the display.

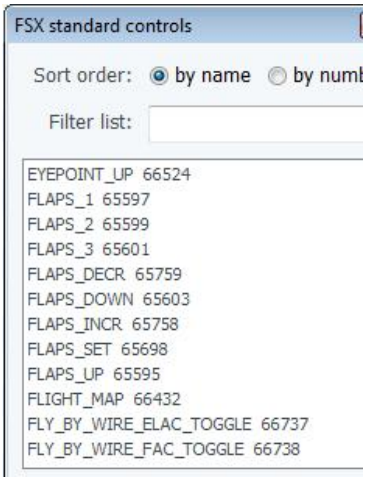
So now you know which button number (on which device) to program. Your next decision is whether you want the action to be performed when the button is pressed (in which case you use On Press, the first column), or to repeat for as long as the button is held down (On Repeat), or — and perhaps less usually — when the button is released (On Release). That's normally an easy decision, with the possible exception of cases where you are using as a button a device which is in fact a switch. In such cases, when the switch is open / off this is the equivalent of the button not being pressed, and when the switch is closed / on this is the button being pressed, and also held down (On Repeat).

So you know which button to use, and also how you want to use it. The third step is to assign a suitable routine (a.k.a. function) to the button. If you have pre-planned your button assignments as suggested, then you will have already researched this question, but let's assume that you are just trying this out for the first time.

Assuming that you want the button press (in this case) to increment the flap setting: all you need to do is to look up what you want in the list of functions for the aircraft you are using, right? Well actually, not quite. If you search through the 700 NGX functions that I have listed in Appendix 1 you will be disappointed to discover that flap functions are not included. So what's gone wrong?

So many functions

Nothing is wrong in such a situation, in fact — it's just that you're probably looking in the wrong place. Some very common functions such as those for flaps (and perhaps throttles, brakes, and so on), are included in the library of functions *for FSX itself*, rather than the specific aircraft you are using. So if you draw a blank in the list of functions for your aircraft don't despair, simply search the list of fsx's functions to see if you can find what you need in there. It's probably fair to say, however, that it's a good idea to search the list of functions which are specific to your add-on aircraft *first*, since if it exists there, there's a good chance that the standard fsx function isn't going to work, or not work as well.



I have included a list of some standard fsx functions in Appendix 3: the extract on the right includes the flaps functions you can assign, and for the current example you're clearly going to select FLAPS_INCR and assign it to the appropriate column for your selected button. (Further examples will be given below).

At this point we encounter an aspect which makes the writing of this guide more difficult: there are so many aspects of all this which are "obvious" to experienced folk, but not so obvious to the beginner. Let's consider a related example to show what I mean.

[17] LINDA's button numbers start from 1 (as in the Windows Game Controllers' configuration windows). However, be aware (since it often causes confusion) that in FSUIPC numbering starts from **zero**, so button 5 in LINDA or Windows is the same button as button 4 when seen in FSUIPC. Just thought I'd mention it….

Combining existing functions to make new ones

Using the fsx flaps increment and flaps decrement commands we have seen how you can assign buttons to move the flaps, one position at a time. But, of course, some aircraft have more increments between fully up and fully down than others. So, for example, if you wanted to assign a button to raise your flaps completely for your choice of add-on aircraft, how could you do it?

Let's once again consider the 737 NG. If you were at flaps 1, then a single "decrement" command would accomplish what you need; but if you were at flaps 40 then you would need 9 decrement commands to do the job. So how are you going to assign a single button press to fully raise the flaps regardless of the current flap position?

In cases such as this, you will need to write a small Lua function of your own to accomplish the task. Whilst this guide does not set out to teach Lua programming or syntax (a search of the Internet will offer you several alternative tutorials for that purpose), the basics of achieving what you need can fairly simply be achieved — by using the large amount of existing code that comes with LINDA to give you examples of how the Lua syntax works. (Perhaps it's cheating, but it works). 

The first question has to be — where are you going to put your new function so that LINDA can find it, and hence it becomes usable? LINDA very thoughtfully provides a location that is made specifically for routines of your own. You may recall installing LINDA into the <*wherever your fsx base directory is*>\Modules directory. Beneath the Modules directory there is now a directory called (appropriately enough) "linda", and below that again are other directories which comprise the LINDA system. The file you are looking to add your new function to is this one:—

<*fsx base directory*>\Modules\linda\libs\lib-user.lua

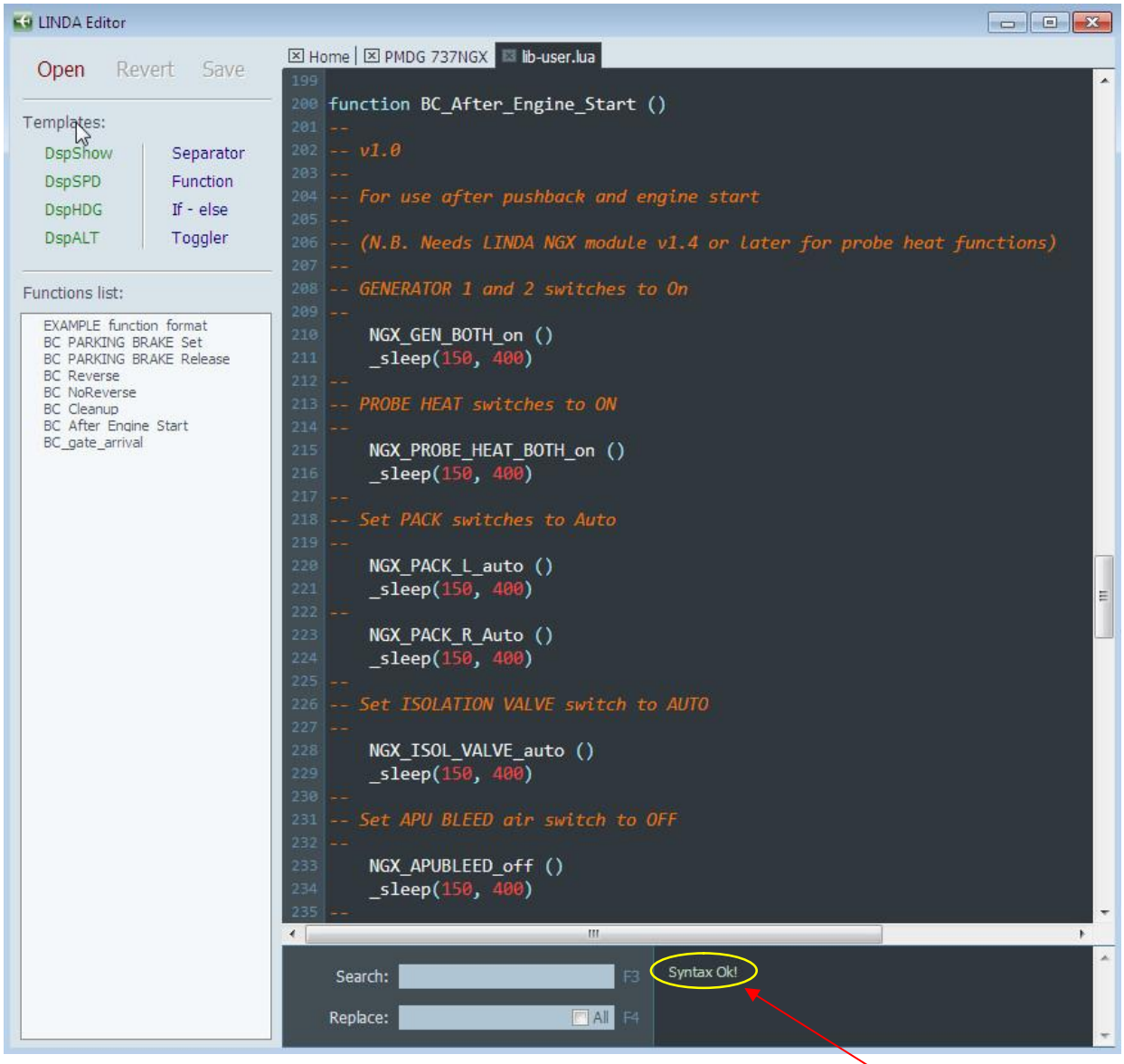
As supplied, the initial contents of that file are as follows:

```
-- User functions

-- v1.1 April 15th, 2012

-- Use this file to insert your own functions and macros.
-- Be careful with syntax as it could ruin the whole VRIDRV system.
-- Use unique function names to not interfere with existing system or
aircraft functions.

--[[
EXAMPLE
--]]

function EXAMPLE_function_format ()

	-- some code
	-- for ex. GEAR UP command
	_GEAR_UP ()                    -- the same as -> ipc.control(66079)

	-- log/debug your function
	-- view results in *\FSX\modules\fsuipc.log
	_log("Gears up, Captain!")

	-- to put something on MCP display use
	-- both strings should be 4 chars exactly - no more, no less
	DspShow ("Gear", "Up")

end
```

You could, if you wish, remove the example by deleting every line starting at “function EXAMPLE….” through to “end”, inclusive — or else simply leave it there, because *it won’t do anything unless it is specifically assigned to a button or called in some other way*. So let’s now add our new function below the example, using either a text editor or (preferably) LINDA’s Editor for the purpose.

The LINDA Editor

On the right of the LINDA Settings screen (click the Settings button at the top if you are in a different part of LINDA) there’s a column headed “Summary”, and in there you will see “LINDA Editor” — otherwise simply press F2 whilst in LINDA and you will be taken to the Editor directly.



As you can see from the illustration, this is a fully-featured editor complete with syntax checking and highlighting and other advanced features, which is the ideal place to enter and save your new function, especially if you are already familiar with programming software. Alternatively, if you would rather keep things as simple as possible, you can edit the file directly using Notepad or any other **text** editor (please don’t try using WordPad, Word, or any similar programs). That means you can’t use the LINDA Editor’s advanced features, but the choice is yours (and either method will work).

To make our new function to fully retract the flaps we take advantage of the fact that you can continue decrementing the flaps position as many times as you like — but once they are fully retracted the command will have no further effect. So we simply decrement the flaps the maximum number of times (for the current aircraft), and we know that irrespective of their previous position the flaps will be moved to the fully up state as a result. So here's our new function —

```
function YourName_retract_flaps ()
--
-- Retract the flaps to the UP position
--
	_FLAPS_DECR ()
	ipc.sleep(50)
	_FLAPS_DECR ()
	ipc.sleep(50)
	_FLAPS_DECR ()
	ipc.sleep(50)
	_FLAPS_DECR ()
	ipc.sleep(50)
	_FLAPS_DECR ()
	ipc.sleep(50)
	_FLAPS_DECR ()
	ipc.sleep(50)
	_FLAPS_DECR ()
	ipc.sleep(50)
	_FLAPS_DECR ()
	ipc.sleep(50)
	_FLAPS_DECR ()
--
	_sleep(150, 400)
--
end
```

This once again illustrates how a function needs to be constructed.

- First of all, you need a line containing the function name and a pair of round brackets (don't worry about those for now, just include them), then some lines of code, which may include calls to existing functions, and finally the "end" line that tells Lua that it has reached the end of the function.

- Anything on a line following two adjacent minus signs (- -) is ignored, so you can use this feature to add as many comments as you like (a practice which is highly recommended!).

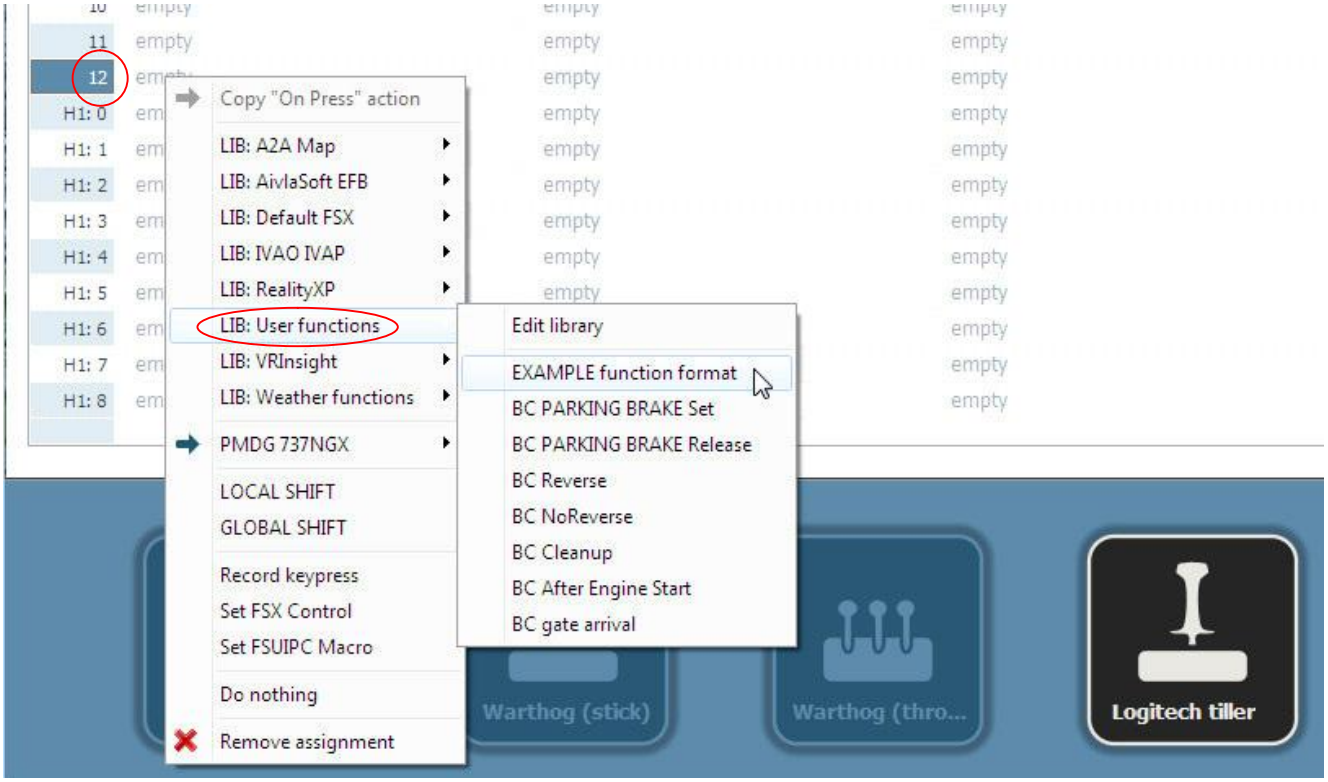
- The "sleep" functions give the processor time to go away and find work to do elsewhere, to avoid a situation where you issue a whole string of commands which tie it up completely, with the result that fsx starts to stutter and stagger. So `ipc.sleep(50)` pauses for 50 milliseconds, whilst `_sleep(150, 400)` pauses for a random time between 150 and 400 milliseconds (don't forget the underscore prefix before the word `sleep`, when using this format!).

- Finally, always put your own name or some other unique identifier in the function name, just to guard against the possibility that an identically-named routine already exists. (It also helps you when you need to debug things too……).

In the "Button assignment basics" section, above, we saw that it is extremely simple to assign any function to a button, and some further examples of that process were promised. So here they are, beginning with assigning a routine of our own.

Button assignment — practical examples

Let's assume that for some reason I wanted to assign the "EXAMPLE function format" function (or any of the others that exist in my lib-user.lua). On my Logitech joystick I pressed a button: as you can see from the picture below, LINDA recognised which control I was using and highlighted button number 12 for me. I then clicked on the word "empty" in the "On Press" column, which opened a list of possibilities for me to use. From the list I selected "LIB: User functions", and from the next column clicked on the function that I wanted to be called when the button was pressed.



It really is as easy as that. You will also notice that there are a number of LIBraries already provided for you, so you can assign functions from any of those categories — including "LIB: Default FSX" (of which more in a moment).

Also present in the list you will see "PMDG 737NGX" — assuming of course that you have downloaded and installed it from the LINDA forum [18]. Clicking on that will give you access to all of the 700 functions that are listed in Appendix 1, which are presented in groups to make it easier to quickly identify the one you need.

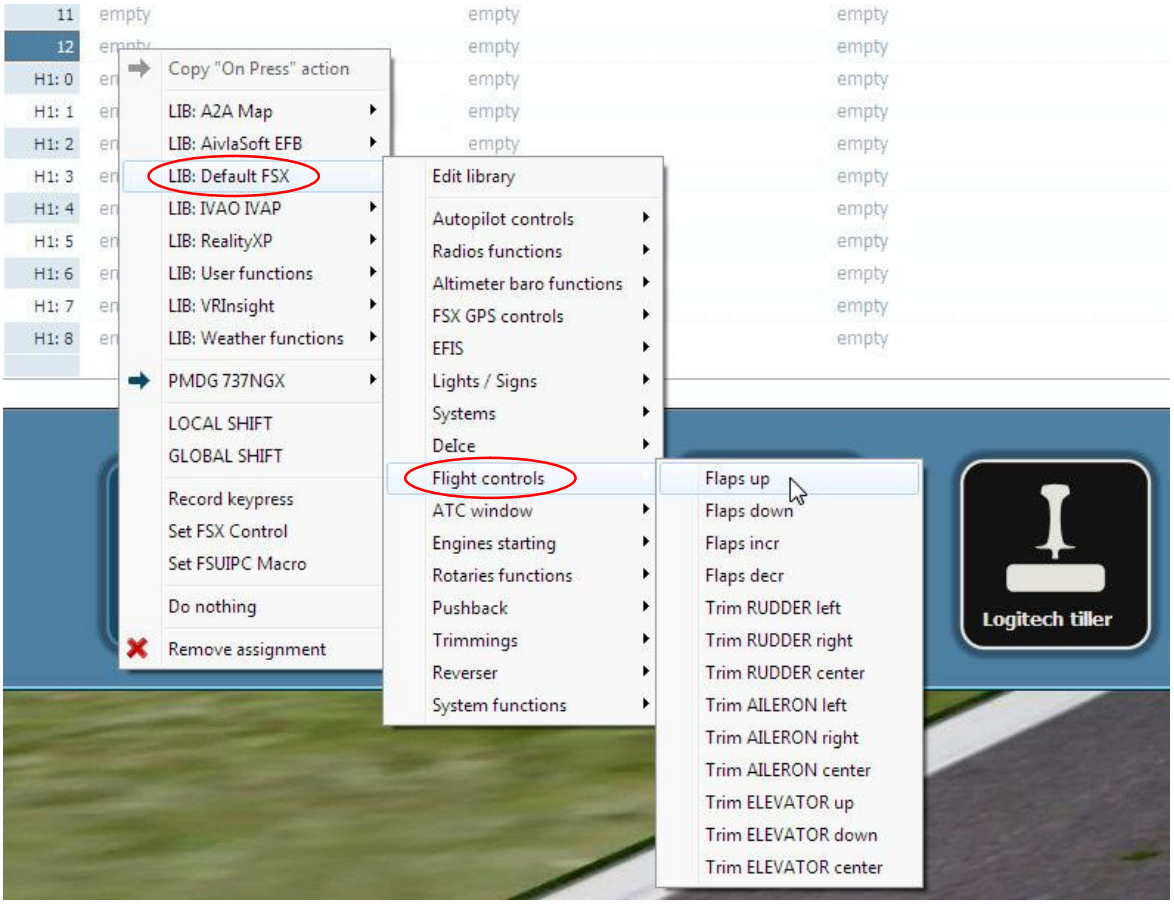
"Set FSX Control" enables you to use the FSX Controls which are listed in Pete Dowson's FSUIPC documentation.

"Record keypress" is fairly obvious (but is occasionally useful to assign a key that is already programmed in fsx).

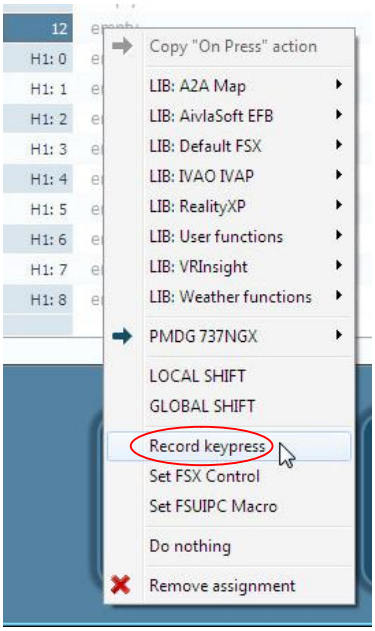
Finally "Do nothing" and "Remove assignment" are even more obvious.

[18] The current version of the PMDG NGX module at the time of writing is v1.7: please don't confuse this with *the version number of the LINDA software*, which is currently v1.11.

Now let's return to our trivial example of a routine to raise the flaps. I have to confess at this point that we did not, in fact, have to write a function of our own — we could have simply used one from the "Default FSX" category, as the following illustration shows:



In fsx, you may also recall that, if we wished to raise the flaps completely (providing that the default button assignments were still in place), we could simply press F5. So instead of either of the two methods shown above, we could simply have used the "Record keypress" option (shown right) and then pressed F5 [19].



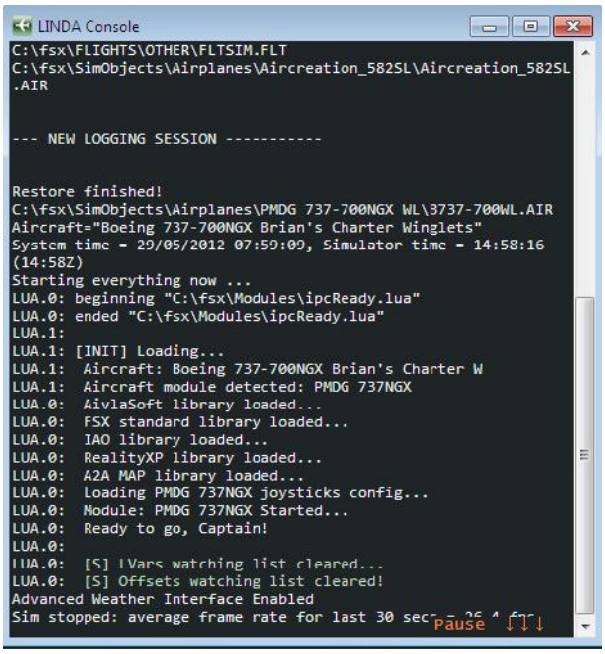
As you can see, when you combine the huge number of functions which LINDA provides (and which Günter and some dedicated users have made available) with the ability to use them within your own Lua routines, your ability to control your aircraft using your own hardware is now vastly enhanced.

Hopefully, therefore, the above has given you some idea of the range of options that LINDA makes available to you when it comes to assigning keystrokes to buttons.

(Personally, I couldn't imagine flying without it)!

[19] Unless, of course, you have reassigned it within fsx. Hence my suggestion for cross-checking *all* the existing assignments whilst planning your control layout.

What if something goes wrong? — the LINDA Console



It is possible, especially if you are using a text editor rather than LINDA's more advanced toolbox, that during your explorations into Lua you may occasionally make a 'syntax error' — in other words, type something that the Lua engine cannot understand. Your first indication of this (especially if you run fsx in full-screen mode) may be that LINDA doesn't seem to be working — how do you find and fix the problem?

The answer lies in LINDA's Console which, even if you are flying in windowed mode, may not be immediately visible onscreen. If it isn't, find the main LINDA screen, give it the focus by clicking on it, and press F1 (or choose "LINDA Console" from the Summary column on the right-hand side of the Settings page).

The graphic (above right) shows an example of what the Console might look like if all is well: but if something really is wrong then you will see an error message (in red, for easy identification). Since this usually gives you the module and the line number within that module of the error, identification and rectification are often rendered straightforward.

The problem might be a misspelling of a keyword, or perhaps that you have inadvertently changed the capitalisation of a function name (unlike Windows, **Lua is case-sensitive**, remember!). An example of this which caught me out was when I was writing a routine which included setting both Pack switches to Auto, as in the following snippet —

```
-- Set PACK switches to Auto
--
       NGX_PACK_L_auto ()
       _sleep(150, 400)
--
       NGX_PACK_R_Auto ()
       _sleep(150, 400)
--
```

Did you spot the difference? Look at the capitalisation of the word 'auto' in the function names: the above (amended) version works, but if you make the capitalisation consistent, it won't (or only for one of them). The moral is to always copy and paste the function name *directly from the function itself* to be sure — I certainly do, now……

Once you have fixed the problem, however, you also need to reload the Lua engine so that your new version is used by LINDA: go to the Settings page and click on the 'Reload Lua engine' option in the right-hand column.

LINDA aims to make your life easier!

LINDA's most complex hardware support — the VRinsight MCP Combo

I thought I might put in a quick word about this, although the topic is well covered on pp 14 – 17 of the LINDA manual.

Installing the VRinsight MCP has also been covered (see page 14 of this document, and also p.9 of the LINDA manual). However, it is worth stressing the importance of selecting (i.e. telling LINDA) ***which MCP Combo version you have.*** If you have bought it recently, then it is pretty much bound to be version 2, but if like me you have the original version then you need to select version 1. Probably the main difference between the two versions is the increased display space in version 2, which (I am told) is fully utilised by the LINDA authors, so make certain that you specify which version you have! It is, after all, a one-time adjustment.

In Appendix 4 you will find a diagram which gives you a rough idea of how the default assignment of LINDA's controls is laid out. Two caveats about this, however: first of all, I compiled it a couple of versions ago, so Günter may have since then made improvements which are not shown; and secondly this obviously refers to the Combo I, since I don't have the Combo II. Nonetheless, it may be useful for those who have just attached their MCP Combo to LINDA for the first time, and would like a quick idea of what all the buttons and knobs actually do.

But, of course, this is LINDA — so if you would like to change anything about the control assignments, you are entirely free to do so. Good luck, if so. Personally, I find the default assignments suit my needs very well, but Your Mileage May Vary as the saying goes……

In conclusion……

I hope this brief tour of some of LINDA's functions and abilities may be of help, especially to those who are encountering LINDA for the first time, or whose computer knowledge is relatively slight.

The writer is aware of the many limitations and imperfections of this document; notably in that it does not include the frequently-requested guide to using the LINDA Tracer [20] to discover and make available the functions within any of the add-on aircraft which are not already available on the LINDA forum (look at http://forum.avsim.net/forum/428-linda-development) for the modules already available for download). But frankly, such a task would result in an additional document at least as long as this one, and furthermore there are relatively few people who are qualified to write it. Certainly Günter (who has to be more qualified than anyone, since he has been investigating the internals of add-on aircraft for a long time) is far too busy investigating and publishing additional modules — as well as providing support on the forum, which is a not inconsiderable task by itself. So I fear that those who seek a shortcut to the delicate arts of Tracing will have to be content with the various replies which have been given on the LINDA forum during the past year. I must add also that, as with any such skilled activity, the relevant skill can only be honed and perfected *by practice*, rather than by reading about it. (A bit like flying, in fact….).

But in all fairness, this document is entitled "*Introducing* LINDA", and I hope that it has succeeded in that modest aim. Whilst writing it I have once again been forcibly reminded of just how much the simming community owes to Artem and Günter, who have supplied for our use a professional-level tool, and the associated add-ons to make it immediately useful to many simmers. The LINDA main screen has a Donate button: please use it to encourage the developers in their efforts (just imagine how much this utility would have cost if it had been provided by any of the major simming software houses!).

And final and most sincere thanks also to those members of the LINDA forum who have contributed to the code base of LINDA aircraft modules: the more people who do so, the more we will all benefit in the long run. (And the more examples you will have to look at when making and debugging your own code).

But the business of writing this document has for too long kept me from simming, to which I will now return with (I must confess) a certain relief.

Blue skies and happy landings, my friends!

Cheers,

Brian

Hampshire, England.

Summer 2012

[20] The best information that I am aware of on this topic (i.e. from Günter himself) is to be found here: http://forum.simflight.com/topic/63146-tutorial-how-to-get-lua-vars-or-commands-out-of-fs/

Appendix 1 — alphabetic list of NGX functions (module v1.7)

FUNCTION NAME	FUNCTION GROUP
function InitDsp ()	System functions
function InitVars ()	System functions
function NGX_AIL_TRIM_leftWingDown ()	Trimmings
function NGX_AIL_TRIM_rightWingDown ()	Trimmings
function NGX_AIL_TRIM_show ()	Trimmings
function NGX_AIL_TRIM_stop ()	Trimmings
function NGX_ANTI_ICE_ENG_1_off ()	Anti Ice
function NGX_ANTI_ICE_ENG_1_on ()	Anti Ice
function NGX_ANTI_ICE_ENG_1_toggle ()	Anti Ice
function NGX_ANTI_ICE_ENG_2_off ()	Anti Ice
function NGX_ANTI_ICE_ENG_2_on ()	Anti Ice
function NGX_ANTI_ICE_ENG_2_toggle ()	Anti Ice
function NGX_ANTI_ICE_ENG_both_off ()	Anti Ice
function NGX_ANTI_ICE_ENG_both_on ()	Anti Ice
function NGX_ANTI_ICE_ENG_both_toggle ()	Anti Ice
function NGX_ANTI_ICE_ENG_WING_off ()	Anti Ice
function NGX_ANTI_ICE_ENG_WING_on ()	Anti Ice
function NGX_ANTI_ICE_ENG_WING_toggle ()	Anti Ice
function NGX_ANTI_ICE_WING_off ()	Anti Ice
function NGX_ANTI_ICE_WING_on ()	Anti Ice
function NGX_ANTI_ICE_WING_toggle ()	Anti Ice
function NGX_AP_ALT_dec ()	Autopilot Dials
function NGX_AP_ALT_decfast ()	Autopilot Dials
function NGX_AP_ALT_inc ()	Autopilot Dials
function NGX_AP_ALT_incfast ()	Autopilot Dials
function NGX_AP_ALT_INTV ()	Autopilot buttons
function NGX_AP_ALT_show ()	Autopilot Dials
function NGX_AP_ALTHLD ()	Autopilot buttons
function NGX_AP_APP ()	Autopilot buttons
function NGX_AP_AT_FMC_reset ()	Warnings
function NGX_AP_ATHR_arm ()	Autopilot switches
function NGX_AP_ATHR_off ()	Autopilot switches
function NGX_AP_ATHR_toggle ()	Autopilot switches
function NGX_AP_BACK_SYNC ()	System functions
function NGX_AP_BANK_dec ()	Autopilot Dials
function NGX_AP_BANK_inc ()	Autopilot Dials
function NGX_AP_BANK_show ()	Autopilot Dials
function NGX_AP_CMDA_toggle ()	Autopilot buttons
function NGX_AP_CMDB_toggle ()	Autopilot buttons
function NGX_AP_CO ()	Autopilot buttons
function NGX_AP_CRS_LR_dec ()	Autopilot Dials
function NGX_AP_CRS_LR_decfast ()	Autopilot Dials
function NGX_AP_CRS_LR_inc ()	Autopilot Dials
function NGX_AP_CRS_LR_incfast ()	Autopilot Dials
function NGX_AP_CRS_LR_sync ()	Autopilot Dials
function NGX_AP_CRS_LR_toggle ()	Autopilot Dials
function NGX_AP_CRSL_dec ()	Autopilot Dials

function NGX_AP_CRSL_decfast ()	Autopilot Dials
function NGX_AP_CRSL_inc ()	Autopilot Dials
function NGX_AP_CRSL_incfast ()	Autopilot Dials
function NGX_AP_CRSL_show ()	Autopilot Dials
function NGX_AP_CRSR_dec ()	Autopilot Dials
function NGX_AP_CRSR_decfast ()	Autopilot Dials
function NGX_AP_CRSR_inc ()	Autopilot Dials
function NGX_AP_CRSR_incfast ()	Autopilot Dials
function NGX_AP_CRSR_show ()	Autopilot Dials
function NGX_AP_CWSA ()	Autopilot buttons
function NGX_AP_CWSB ()	Autopilot buttons
function NGX_AP_FD1_off ()	Autopilot switches
function NGX_AP_FD1_on ()	Autopilot switches
function NGX_AP_FD1_toggle ()	Autopilot switches
function NGX_AP_FD2_off ()	Autopilot switches
function NGX_AP_FD2_on ()	Autopilot switches
function NGX_AP_FD2_toggle ()	Autopilot switches
function NGX_AP_FD_both_off ()	Autopilot switches
function NGX_AP_FD_both_on ()	Autopilot switches
function NGX_AP_HDG_BANK_toggle ()	Autopilot Dials
function NGX_AP_HDG_dec ()	Autopilot Dials
function NGX_AP_HDG_decfast ()	Autopilot Dials
function NGX_AP_HDG_inc ()	Autopilot Dials
function NGX_AP_HDG_incfast ()	Autopilot Dials
function NGX_AP_HDG_show ()	Autopilot Dials
function NGX_AP_HDGSEL ()	Autopilot buttons
function NGX_AP_INFO ()	System functions
function NGX_AP_LNAV ()	Autopilot buttons
function NGX_AP_LVLCHG ()	Autopilot buttons
function NGX_AP_MASTER_off ()	Autopilot switches
function NGX_AP_MASTER_on ()	Autopilot switches
function NGX_AP_MASTER_toggle ()	Autopilot switches
function NGX_AP_MODES_UPDATE ()	System functions
function NGX_AP_N1 ()	Autopilot buttons
function NGX_AP_P_reset ()	Warnings
function NGX_AP_soft_disconnect ()	Autopilot switches
function NGX_AP_SPD_dec ()	Autopilot Dials
function NGX_AP_SPD_decfast ()	Autopilot Dials
function NGX_AP_SPD_inc ()	Autopilot Dials
function NGX_AP_SPD_incfast ()	Autopilot Dials
function NGX_AP_SPD_INTV ()	Autopilot buttons
function NGX_AP_SPD_show ()	Autopilot Dials
function NGX_AP_SPEED ()	Autopilot buttons
function NGX_AP_TOGA ()	Autopilot switches
function NGX_AP_VNAV ()	Autopilot buttons
function NGX_AP_VORLOC ()	Autopilot buttons
function NGX_AP_VS ()	Autopilot buttons
function NGX_AP_VS_dec ()	Autopilot Dials
function NGX_AP_VS_inc ()	Autopilot Dials
function NGX_AP_VS_show ()	Autopilot Dials
function NGX_APU_dec ()	Electrics (including APU)
function NGX_APU_GEN_BOTH_off ()	Electrics (including APU)

function NGX_APU_GEN_BOTH_on ()	Electrics (including APU)
function NGX_APU_GEN_L_off ()	Electrics (including APU)
function NGX_APU_GEN_L_on ()	Electrics (including APU)
function NGX_APU_GEN_R_off ()	Electrics (including APU)
function NGX_APU_GEN_R_on ()	Electrics (including APU)
function NGX_APU_inc ()	Electrics (including APU)
function NGX_APU_off ()	Electrics (including APU)
function NGX_APU_on ()	Electrics (including APU)
function NGX_APU_show ()	Electrics (including APU)
function NGX_APU_Start_or_ShowEGT ()	Electrics (including APU)
function NGX_APUBLEED_off ()	Bleed and Packs (including APU bleeds)
function NGX_APUBLEED_on ()	Bleed and Packs (including APU bleeds)
function NGX_APUBLEED_show ()	Bleed and Packs (including APU bleeds)
function NGX_APUBLEED_toggle ()	Bleed and Packs (including APU bleeds)
function NGX_AT_P_reset ()	Warnings
function NGX_ATHR_soft_disconnect ()	Autopilot switches
function NGX_AUTOBRAKE_1 ()	Autobrake
function NGX_AUTOBRAKE_2 ()	Autobrake
function NGX_AUTOBRAKE_3 ()	Autobrake
function NGX_AUTOBRAKE_calc ()	Autobrake
function NGX_AUTOBRAKE_dec ()	Autobrake
function NGX_AUTOBRAKE_inc ()	Autobrake
function NGX_AUTOBRAKE_MAX ()	Autobrake
function NGX_AUTOBRAKE_OFF ()	Autobrake
function NGX_AUTOBRAKE_RTO ()	Autobrake
function NGX_AUTOBRAKE_show ()	Autobrake
function NGX_BAT_off ()	Electrics (including APU)
function NGX_BAT_on ()	Electrics (including APU)
function NGX_BAT_toggle ()	Electrics (including APU)
function NGX_BEACON_off ()	Lights, external
function NGX_BEACON_on ()	Lights, external
function NGX_BEACON_toggle ()	Lights, external
function NGX_BelowGS_Inhibit ()	Warnings
function NGX_BLEED1_off ()	Bleed and Packs (including APU bleeds)
function NGX_BLEED1_on ()	Bleed and Packs (including APU bleeds)
function NGX_BLEED1_show ()	Bleed and Packs (including APU bleeds)
function NGX_BLEED1_toggle ()	Bleed and Packs (including APU bleeds)
function NGX_BLEED2_off ()	Bleed and Packs (including APU bleeds)
function NGX_BLEED2_on ()	Bleed and Packs (including APU bleeds)
function NGX_BLEED2_show ()	Bleed and Packs (including APU bleeds)
function NGX_BLEED2_toggle ()	Bleed and Packs (including APU bleeds)
function NGX_CDU2_0 ()	CDU FO
function NGX_CDU2_1 ()	CDU FO
function NGX_CDU2_2 ()	CDU FO
function NGX_CDU2_3 ()	CDU FO
function NGX_CDU2_4 ()	CDU FO
function NGX_CDU2_5 ()	CDU FO
function NGX_CDU2_6 ()	CDU FO
function NGX_CDU2_7 ()	CDU FO
function NGX_CDU2_8 ()	CDU FO
function NGX_CDU2_9 ()	CDU FO
function NGX_CDU2_A ()	CDU FO

function NGX_CDU2_B ()	CDU FO
function NGX_CDU2_C ()	CDU FO
function NGX_CDU2_CLB ()	CDU FO
function NGX_CDU2_CLR ()	CDU FO
function NGX_CDU2_CRZ ()	CDU FO
function NGX_CDU2_D ()	CDU FO
function NGX_CDU2_DEL ()	CDU FO
function NGX_CDU2_DEP_ARR ()	CDU FO
function NGX_CDU2_DES ()	CDU FO
function NGX_CDU2_Doors()	Enter CDU FO pages
function NGX_CDU2_Doors_Airstair()	CDU FO Doors
function NGX_CDU2_Doors_CargoAFT()	CDU FO Doors
function NGX_CDU2_Doors_CargoFWD()	CDU FO Doors
function NGX_CDU2_Doors_Equip()	CDU FO Doors
function NGX_CDU2_Doors_LeftAFT()	CDU FO Doors
function NGX_CDU2_Doors_LeftFWD()	CDU FO Doors
function NGX_CDU2_Doors_LeftWING()	CDU FO Doors
function NGX_CDU2_Doors_RightAFT()	CDU FO Doors
function NGX_CDU2_Doors_RightFWD()	CDU FO Doors
function NGX_CDU2_Doors_RightWING()	CDU FO Doors
function NGX_CDU2_E ()	CDU FO
function NGX_CDU2_EXEC ()	CDU FO
function NGX_CDU2_F ()	CDU FO
function NGX_CDU2_FIX ()	CDU FO
function NGX_CDU2_Fuel ()	Enter CDU FO pages
function NGX_CDU2_FwdSlash ()	CDU FO
function NGX_CDU2_G ()	CDU FO
function NGX_CDU2_GroundConn ()	Enter CDU FO pages
function NGX_CDU2_H ()	CDU FO
function NGX_CDU2_HOLD ()	CDU FO
function NGX_CDU2_I ()	CDU FO
function NGX_CDU2_INIT_REF ()	CDU FO
function NGX_CDU2_J ()	CDU FO
function NGX_CDU2_K ()	CDU FO
function NGX_CDU2_L ()	CDU FO
function NGX_CDU2_LEGS ()	CDU FO
function NGX_CDU2_LSK_1L ()	CDU FO
function NGX_CDU2_LSK_1R ()	CDU FO
function NGX_CDU2_LSK_2L ()	CDU FO
function NGX_CDU2_LSK_2R ()	CDU FO
function NGX_CDU2_LSK_3L ()	CDU FO
function NGX_CDU2_LSK_3R ()	CDU FO
function NGX_CDU2_LSK_4L ()	CDU FO
function NGX_CDU2_LSK_4R ()	CDU FO
function NGX_CDU2_LSK_5L ()	CDU FO
function NGX_CDU2_LSK_5R ()	CDU FO
function NGX_CDU2_LSK_6L ()	CDU FO
function NGX_CDU2_LSK_6R ()	CDU FO
function NGX_CDU2_M ()	CDU FO
function NGX_CDU2_MENU ()	CDU FO
function NGX_CDU2_N ()	CDU FO
function NGX_CDU2_N1_LIMIT ()	CDU FO

function NGX_CDU2_NEXT_PAGE ()	CDU FO
function NGX_CDU2_O ()	CDU FO
function NGX_CDU2_P ()	CDU FO
function NGX_CDU2_Payload ()	Enter CDU FO pages
function NGX_CDU2_Period ()	CDU FO
function NGX_CDU2_PREV_PAGE ()	CDU FO
function NGX_CDU2_PROG ()	CDU FO
function NGX_CDU2_Pushback ()	Enter CDU FO pages
function NGX_CDU2_Q ()	CDU FO
function NGX_CDU2_R ()	CDU FO
function NGX_CDU2_RTE ()	CDU FO
function NGX_CDU2_S ()	CDU FO
function NGX_CDU2_Sign ()	CDU FO
function NGX_CDU2_Space ()	CDU FO
function NGX_CDU2_T ()	CDU FO
function NGX_CDU2_U ()	CDU FO
function NGX_CDU2_V ()	CDU FO
function NGX_CDU2_W ()	CDU FO
function NGX_CDU2_X ()	CDU FO
function NGX_CDU2_Y ()	CDU FO
function NGX_CDU2_Z ()	CDU FO
function NGX_CDU_0 ()	CDU Capt
function NGX_CDU_1 ()	CDU Capt
function NGX_CDU_2 ()	CDU Capt
function NGX_CDU_3 ()	CDU Capt
function NGX_CDU_4 ()	CDU Capt
function NGX_CDU_5 ()	CDU Capt
function NGX_CDU_6 ()	CDU Capt
function NGX_CDU_7 ()	CDU Capt
function NGX_CDU_8 ()	CDU Capt
function NGX_CDU_9 ()	CDU Capt
function NGX_CDU_A ()	CDU Capt
function NGX_CDU_B ()	CDU Capt
function NGX_CDU_C ()	CDU Capt
function NGX_CDU_CLB ()	CDU Capt
function NGX_CDU_CLR ()	CDU Capt
function NGX_CDU_CRZ ()	CDU Capt
function NGX_CDU_D ()	CDU Capt
function NGX_CDU_DEL ()	CDU Capt
function NGX_CDU_DEP_ARR ()	CDU Capt
function NGX_CDU_DES ()	CDU Capt
function NGX_CDU_Doors()	Enter CDU Capt pages
function NGX_CDU_E ()	CDU Capt
function NGX_CDU_EXEC ()	CDU Capt
function NGX_CDU_F ()	CDU Capt
function NGX_CDU_FIX ()	CDU Capt
function NGX_CDU_Fuel ()	Enter CDU Capt pages
function NGX_CDU_FwdSlash ()	CDU Capt
function NGX_CDU_G ()	CDU Capt
function NGX_CDU_GroundConn ()	Enter CDU Capt pages
function NGX_CDU_H ()	CDU Capt
function NGX_CDU_HOLD ()	CDU Capt

function NGX_CDU_I ()	CDU Capt
function NGX_CDU_INIT_REF ()	CDU Capt
function NGX_CDU_J ()	CDU Capt
function NGX_CDU_K ()	CDU Capt
function NGX_CDU_L ()	CDU Capt
function NGX_CDU_LEGS ()	CDU Capt
function NGX_CDU_LSK_1L ()	CDU Capt
function NGX_CDU_LSK_1R ()	CDU Capt
function NGX_CDU_LSK_2L ()	CDU Capt
function NGX_CDU_LSK_2R ()	CDU Capt
function NGX_CDU_LSK_3L ()	CDU Capt
function NGX_CDU_LSK_3R ()	CDU Capt
function NGX_CDU_LSK_4L ()	CDU Capt
function NGX_CDU_LSK_4R ()	CDU Capt
function NGX_CDU_LSK_5L ()	CDU Capt
function NGX_CDU_LSK_5R ()	CDU Capt
function NGX_CDU_LSK_6L ()	CDU Capt
function NGX_CDU_LSK_6R ()	CDU Capt
function NGX_CDU_M ()	CDU Capt
function NGX_CDU_MENU ()	CDU Capt
function NGX_CDU_N ()	CDU Capt
function NGX_CDU_N1_LIMIT ()	CDU Capt
function NGX_CDU_NEXT_PAGE ()	CDU Capt
function NGX_CDU_O ()	CDU Capt
function NGX_CDU_P ()	CDU Capt
function NGX_CDU_Payload ()	Enter CDU Capt pages
function NGX_CDU_Period ()	CDU Capt
function NGX_CDU_PREV_PAGE ()	CDU Capt
function NGX_CDU_PROG ()	CDU Capt
function NGX_CDU_Pushback ()	Enter CDU Capt pages
function NGX_CDU_Q ()	CDU Capt
function NGX_CDU_R ()	CDU Capt
function NGX_CDU_RTE ()	CDU Capt
function NGX_CDU_S ()	CDU Capt
function NGX_CDU_Sign ()	CDU Capt
function NGX_CDU_Space ()	CDU Capt
function NGX_CDU_T ()	CDU Capt
function NGX_CDU_U ()	CDU Capt
function NGX_CDU_V ()	CDU Capt
function NGX_CDU_W ()	CDU Capt
function NGX_CDU_X ()	CDU Capt
function NGX_CDU_Y ()	CDU Capt
function NGX_CDU_Z ()	CDU Capt
function NGX_Cockpit_Prepare ()	Cockpit prepare
function NGX_CROSSFEED_feed ()	Fuel
function NGX_CROSSFEED_off ()	Fuel
function NGX_CROSSFEED_toggle ()	Fuel
function NGX_DOME_bright ()	Lights, cockpit
function NGX_DOME_cycle ()	Lights, cockpit
function NGX_DOME_dim ()	Lights, cockpit
function NGX_DOME_off ()	Lights, cockpit
function NGX_Doors_Airstair()	CDU Capt Doors

function NGX_Doors_CargoAFT()	CDU Capt Doors
function NGX_Doors_CargoFWD()	CDU Capt Doors
function NGX_Doors_Equip()	CDU Capt Doors
function NGX_Doors_LeftAFT()	CDU Capt Doors
function NGX_Doors_LeftFWD()	CDU Capt Doors
function NGX_Doors_LeftWING()	CDU Capt Doors
function NGX_Doors_RightAFT()	CDU Capt Doors
function NGX_Doors_RightFWD()	CDU Capt Doors
function NGX_Doors_RightWING()	CDU Capt Doors
function NGX_DU1_INBD_BRT_dec ()	Disp. brightness
function NGX_DU1_INBD_BRT_inc ()	Disp. brightness
function NGX_DU1_LOWER_dec ()	Disp. modes
function NGX_DU1_LOWER_inc ()	Disp. modes
function NGX_DU1_LOWER_show ()	Disp. modes
function NGX_DU1_MAIN_dec ()	Disp. modes
function NGX_DU1_MAIN_inc ()	Disp. modes
function NGX_DU1_MAIN_show ()	Disp. modes
function NGX_DU1_OUTBD_BRT_dec ()	Disp. brightness
function NGX_DU1_OUTBD_BRT_inc ()	Disp. brightness
function NGX_DU2_INBD_BRT_dec ()	Disp. brightness
function NGX_DU2_INBD_BRT_inc ()	Disp. brightness
function NGX_DU2_LOWER_dec ()	Disp. modes
function NGX_DU2_LOWER_inc ()	Disp. modes
function NGX_DU2_LOWER_show ()	Disp. modes
function NGX_DU2_MAIN_dec ()	Disp. modes
function NGX_DU2_MAIN_inc ()	Disp. modes
function NGX_DU2_MAIN_show ()	Disp. modes
function NGX_DU2_OUTBD_BRT_dec ()	Disp. brightness
function NGX_DU2_OUTBD_BRT_inc ()	Disp. brightness
function NGX_DU_ALL_BRT_dec ()	Disp. brightness
function NGX_DU_ALL_BRT_decfast ()	Disp. brightness
function NGX_DU_ALL_BRT_inc ()	Disp. brightness
function NGX_DU_ALL_BRT_incfast ()	Disp. brightness
function NGX_DU_ALL_BRT_show ()	Disp. brightness
function NGX_DU_CENTER_dec ()	Disp. brightness
function NGX_DU_CENTER_decfast ()	Disp. brightness
function NGX_DU_CENTER_inc ()	Disp. brightness
function NGX_DU_CENTER_incfast ()	Disp. brightness
function NGX_DU_LEFT_BRT_dec ()	Disp. brightness
function NGX_DU_LEFT_BRT_decfast ()	Disp. brightness
function NGX_DU_LEFT_BRT_inc ()	Disp. brightness
function NGX_DU_LEFT_BRT_incfast ()	Disp. brightness
function NGX_DU_LOWER_BRT_dec ()	Disp. brightness
function NGX_DU_LOWER_BRT_inc ()	Disp. brightness
function NGX_DU_LOWER_eng ()	Disp. modes
function NGX_DU_LOWER_sys ()	Disp. modes
function NGX_DU_LR_BRT_dec ()	Disp. brightness
function NGX_DU_LR_BRT_decfast ()	Disp. brightness
function NGX_DU_LR_BRT_inc ()	Disp. brightness
function NGX_DU_LR_BRT_incfast ()	Disp. brightness
function NGX_DU_RIGHT_BRT_dec ()	Disp. brightness
function NGX_DU_RIGHT_BRT_decfast ()	Disp. brightness

function NGX_DU_RIGHT_BRT_inc ()	Disp. brightness
function NGX_DU_RIGHT_BRT_incfast ()	Disp. brightness
function NGX_DU_UPPER_BRT_dec ()	Disp. brightness
function NGX_DU_UPPER_BRT_inc ()	Disp. brightness
function NGX_EFIS_ARPT ()	EFIS controls
function NGX_EFIS_BARO_dec ()	EFIS controls
function NGX_EFIS_BARO_inc ()	EFIS controls
function NGX_EFIS_BARO_MODE_hpa ()	EFIS controls
function NGX_EFIS_BARO_MODE_inHg ()	EFIS controls
function NGX_EFIS_BARO_MODE_toggle ()	EFIS controls
function NGX_EFIS_BARO_STD_toggle ()	EFIS controls
function NGX_EFIS_DATA ()	EFIS controls
function NGX_EFIS_FPV ()	EFIS controls
function NGX_EFIS_MINS_dec ()	EFIS controls
function NGX_EFIS_MINS_inc ()	EFIS controls
function NGX_EFIS_MINS_MODE_baro ()	EFIS controls
function NGX_EFIS_MINS_MODE_radio ()	EFIS controls
function NGX_EFIS_MINS_MODE_toggle ()	EFIS controls
function NGX_EFIS_MINS_RST ()	EFIS controls
function NGX_EFIS_MTRS ()	EFIS controls
function NGX_EFIS_NAV1_adf ()	EFIS controls
function NGX_EFIS_NAV1_dec ()	EFIS controls
function NGX_EFIS_NAV1_inc ()	EFIS controls
function NGX_EFIS_NAV1_off ()	EFIS controls
function NGX_EFIS_NAV1_vor ()	EFIS controls
function NGX_EFIS_NAV2_adf ()	EFIS controls
function NGX_EFIS_NAV2_dec ()	EFIS controls
function NGX_EFIS_NAV2_inc ()	EFIS controls
function NGX_EFIS_NAV2_off ()	EFIS controls
function NGX_EFIS_NAV2_vor ()	EFIS controls
function NGX_EFIS_ND_MODE_CTR ()	EFIS controls
function NGX_EFIS_ND_MODE_dec ()	EFIS controls
function NGX_EFIS_ND_MODE_inc ()	EFIS controls
function NGX_EFIS_ND_MODE_show ()	EFIS controls
function NGX_EFIS_ND_RNG_dec ()	EFIS controls
function NGX_EFIS_ND_RNG_inc ()	EFIS controls
function NGX_EFIS_ND_RNG_show ()	EFIS controls
function NGX_EFIS_ND_RNG_TFC ()	EFIS controls
function NGX_EFIS_POS ()	EFIS controls
function NGX_EFIS_STA ()	EFIS controls
function NGX_EFIS_TERR ()	EFIS controls
function NGX_EFIS_WPT ()	EFIS controls
function NGX_EFIS_WXR ()	EFIS controls
function NGX_ELEV_TRIM_down ()	Trimmings
function NGX_ELEV_TRIM_downfast ()	Trimmings
function NGX_ELEV_TRIM_show ()	Trimmings
function NGX_ELEV_TRIM_up ()	Trimmings
function NGX_ELEV_TRIM_upfast ()	Trimmings
function NGX_EMER_lights_armed ()	Lights, cockpit
function NGX_EMER_lights_off ()	Lights, cockpit
function NGX_EMER_lights_on ()	Lights, cockpit
function NGX_ENG1_cutoff ()	Engine start

function NGX_ENG1_idle ()	Engine start
function NGX_ENG1_START_CONT ()	Engine start
function NGX_ENG1_START_dec ()	Engine start
function NGX_ENG1_START_FLT ()	Engine start
function NGX_ENG1_START_GRD ()	Engine start
function NGX_ENG1_START_inc ()	Engine start
function NGX_ENG1_START_OFF ()	Engine start
function NGX_ENG1_START_show ()	Engine start
function NGX_ENG2_cutoff ()	Engine start
function NGX_ENG2_idle ()	Engine start
function NGX_ENG2_START_CONT ()	Engine start
function NGX_ENG2_START_dec ()	Engine start
function NGX_ENG2_START_FLT ()	Engine start
function NGX_ENG2_START_GRD ()	Engine start
function NGX_ENG2_START_inc ()	Engine start
function NGX_ENG2_START_OFF ()	Engine start
function NGX_ENG2_START_show ()	Engine start
function NGX_FLIGHT_INFO ()	System functions
function NGX_FLOOD_ALL_dec ()	Lights, cockpit
function NGX_FLOOD_ALL_decfast ()	Lights, cockpit
function NGX_FLOOD_ALL_inc ()	Lights, cockpit
function NGX_FLOOD_ALL_incfast ()	Lights, cockpit
function NGX_FLOOD_ALL_show ()	Lights, cockpit
function NGX_FLOOD_MCP_dec ()	Lights, cockpit
function NGX_FLOOD_MCP_inc ()	Lights, cockpit
function NGX_FLOOD_PANEL_dec ()	Lights, cockpit
function NGX_FLOOD_PANEL_inc ()	Lights, cockpit
function NGX_FLOOD_PDST_dec ()	Lights, cockpit
function NGX_FLOOD_PDST_inc ()	Lights, cockpit
function NGX_FLTALT_dec ()	Bleed and Packs (including APU bleeds)
function NGX_FLTALT_inc ()	Bleed and Packs (including APU bleeds)
function NGX_FMC_P_reset ()	Warnings
function NGX_GEAR_down ()	Other
function NGX_GEAR_off ()	Other
function NGX_GEAR_show ()	Other
function NGX_GEAR_up ()	Other
function NGX_GEN_BOTH_off ()	Electrics (including APU)
function NGX_GEN_BOTH_on ()	Electrics (including APU)
function NGX_GEN_L_off ()	Electrics (including APU)
function NGX_GEN_L_on ()	Electrics (including APU)
function NGX_GEN_R_off ()	Electrics (including APU)
function NGX_GEN_R_on ()	Electrics (including APU)
function NGX_GRD_PWR_off ()	Electrics (including APU)
function NGX_GRD_PWR_on ()	Electrics (including APU)
function NGX_HGS_down ()	Other
function NGX_HGS_mode ()	Other
function NGX_HGS_toggle ()	Other
function NGX_HGS_up ()	Other
function NGX_HYD_A_Elec2_off ()	Hydraulics
function NGX_HYD_A_Elec2_on ()	Hydraulics
function NGX_HYD_A_Elec2_toggle ()	Hydraulics
function NGX_HYD_A_Eng1_off ()	Hydraulics

function NGX_HYD_A_Eng1_on ()	Hydraulics
function NGX_HYD_A_Eng1_toggle ()	Hydraulics
function NGX_HYD_B_Elec1_off ()	Hydraulics
function NGX_HYD_B_Elec1_on ()	Hydraulics
function NGX_HYD_B_Elec1_toggle ()	Hydraulics
function NGX_HYD_B_Eng2_off ()	Hydraulics
function NGX_HYD_B_Eng2_on ()	Hydraulics
function NGX_HYD_B_Eng2_toggle ()	Hydraulics
function NGX_HYD_ELEC_Both_off ()	Hydraulics
function NGX_HYD_ELEC_Both_on ()	Hydraulics
function NGX_HYD_ELEC_Both_toggle ()	Hydraulics
function NGX_HYD_ENG_Both_off ()	Hydraulics
function NGX_HYD_ENG_Both_on ()	Hydraulics
function NGX_HYD_ENG_Both_toggle ()	Hydraulics
function NGX_IGN_both ()	Engine start
function NGX_IGN_left ()	Engine start
function NGX_IGN_right ()	Engine start
function NGX_IRS_L_align ()	IRS
function NGX_IRS_L_att ()	IRS
function NGX_IRS_L_calc ()	IRS
function NGX_IRS_L_dec ()	IRS
function NGX_IRS_L_inc ()	IRS
function NGX_IRS_L_nav ()	IRS
function NGX_IRS_L_off ()	IRS
function NGX_IRS_L_show ()	IRS
function NGX_IRS_R_align ()	IRS
function NGX_IRS_R_att ()	IRS
function NGX_IRS_R_calc ()	IRS
function NGX_IRS_R_dec ()	IRS
function NGX_IRS_R_inc ()	IRS
function NGX_IRS_R_nav ()	IRS
function NGX_IRS_R_off ()	IRS
function NGX_IRS_R_show ()	IRS
function NGX_ISOL_VALVE_auto ()	Bleed and Packs (including APU bleeds)
function NGX_ISOL_VALVE_dec ()	Bleed and Packs (including APU bleeds)
function NGX_ISOL_VALVE_inc ()	Bleed and Packs (including APU bleeds)
function NGX_ISOL_VALVE_off ()	Bleed and Packs (including APU bleeds)
function NGX_ISOL_VALVE_open ()	Bleed and Packs (including APU bleeds)
function NGX_ISOL_VALVE_show ()	Bleed and Packs (including APU bleeds)
function NGX_LAND_ALL_off ()	Lights, external
function NGX_LAND_ALL_on ()	Lights, external
function NGX_LAND_ALL_toggle ()	Lights, external
function NGX_LAND_FIXED_BOTH_off ()	Lights, external
function NGX_LAND_FIXED_BOTH_on ()	Lights, external
function NGX_LAND_FIXED_L_off ()	Lights, external
function NGX_LAND_FIXED_L_on ()	Lights, external
function NGX_LAND_FIXED_R_off ()	Lights, external
function NGX_LAND_FIXED_R_on ()	Lights, external
function NGX_LAND_RETR_BOTH_extend ()	Lights, external
function NGX_LAND_RETR_BOTH_off ()	Lights, external
function NGX_LAND_RETR_BOTH_on ()	Lights, external
function NGX_LAND_RETR_L_extend ()	Lights, external

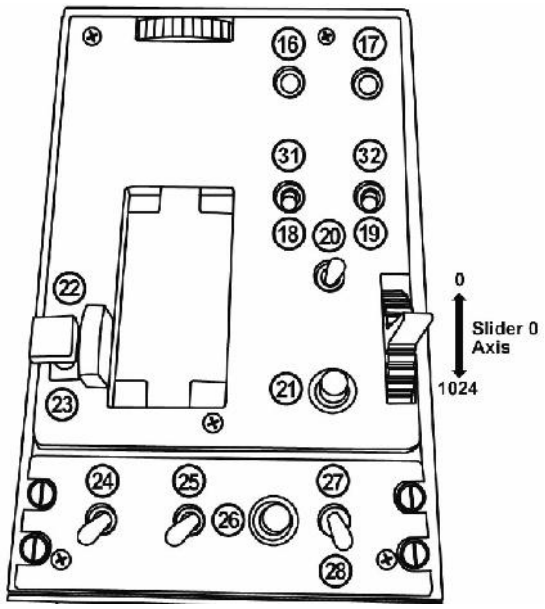
function NGX_LAND_RETR_L_off ()	Lights, external
function NGX_LAND_RETR_L_on ()	Lights, external
function NGX_LAND_RETR_R_extend ()	Lights, external
function NGX_LAND_RETR_R_off ()	Lights, external
function NGX_LAND_RETR_R_on ()	Lights, external
function NGX_LandALT_dec ()	Bleed and Packs (including APU bleeds)
function NGX_LandALT_inc ()	Bleed and Packs (including APU bleeds)
function NGX_LIGHT_ALL_dec ()	Lights, cockpit
function NGX_LIGHT_ALL_decfast ()	Lights, cockpit
function NGX_LIGHT_ALL_inc ()	Lights, cockpit
function NGX_LIGHT_ALL_incfast ()	Lights, cockpit
function NGX_LIGHT_ALL_show ()	Lights, cockpit
function NGX_LIGHT_CB_dec ()	Lights, cockpit
function NGX_LIGHT_CB_inc ()	Lights, cockpit
function NGX_LIGHT_OVH_dec ()	Lights, cockpit
function NGX_LIGHT_OVH_inc ()	Lights, cockpit
function NGX_LIGHT_PANEL_L_dec ()	Lights, cockpit
function NGX_LIGHT_PANEL_L_inc ()	Lights, cockpit
function NGX_LIGHT_PANEL_R_dec ()	Lights, cockpit
function NGX_LIGHT_PANEL_R_inc ()	Lights, cockpit
function NGX_LIGHT_PDST_dec ()	Lights, cockpit
function NGX_LIGHT_PDST_inc ()	Lights, cockpit
function NGX_LOGO_and_NAV_off ()	Lights, external
function NGX_LOGO_and_NAV_steady ()	Lights, external
function NGX_LOGO_off ()	Lights, external
function NGX_LOGO_on ()	Lights, external
function NGX_LOGO_toggle ()	Lights, external
function NGX_Marker_off ()	VHF Panel
function NGX_Marker_on ()	VHF Panel
function NGX_Marker_show ()	VHF Panel
function NGX_NAV_cycle ()	Lights, external
function NGX_NAV_off ()	Lights, external
function NGX_NAV_steady ()	Lights, external
function NGX_NAV_strobe ()	Lights, external
function NGX_PACK_L_auto ()	Bleed and Packs (including APU bleeds)
function NGX_PACK_L_dec ()	Bleed and Packs (including APU bleeds)
function NGX_PACK_L_high ()	Bleed and Packs (including APU bleeds)
function NGX_PACK_L_inc ()	Bleed and Packs (including APU bleeds)
function NGX_PACK_L_off ()	Bleed and Packs (including APU bleeds)
function NGX_PACK_L_show ()	Bleed and Packs (including APU bleeds)
function NGX_PACK_R_Auto ()	Bleed and Packs (including APU bleeds)
function NGX_PACK_R_dec ()	Bleed and Packs (including APU bleeds)
function NGX_PACK_R_High ()	Bleed and Packs (including APU bleeds)
function NGX_PACK_R_inc ()	Bleed and Packs (including APU bleeds)
function NGX_PACK_R_off ()	Bleed and Packs (including APU bleeds)
function NGX_PACK_R_show ()	Bleed and Packs (including APU bleeds)
function NGX_PROBE_HEAT_BOTH_off ()	Probe Heat
function NGX_PROBE_HEAT_BOTH_on ()	Probe Heat
function NGX_PROBE_HEAT_BOTH_toggle ()	Probe Heat
function NGX_PROBE_HEAT_L_off ()	Probe Heat
function NGX_PROBE_HEAT_L_on ()	Probe Heat
function NGX_PROBE_HEAT_L_toggle ()	Probe Heat

function NGX_PROBE_HEAT_R_off ()	Probe Heat
function NGX_PROBE_HEAT_R_on ()	Probe Heat
function NGX_PROBE_HEAT_R_toggle ()	Probe Heat
function NGX_PUMP1_AFT_off ()	Fuel
function NGX_PUMP1_AFT_on ()	Fuel
function NGX_PUMP1_AFT_toggle ()	Fuel
function NGX_PUMP1_FWD_off ()	Fuel
function NGX_PUMP1_FWD_on ()	Fuel
function NGX_PUMP1_FWD_toggle ()	Fuel
function NGX_PUMP2_AFT_off ()	Fuel
function NGX_PUMP2_AFT_on ()	Fuel
function NGX_PUMP2_AFT_toggle ()	Fuel
function NGX_PUMP2_FWD_off ()	Fuel
function NGX_PUMP2_FWD_on ()	Fuel
function NGX_PUMP2_FWD_toggle ()	Fuel
function NGX_PUMPCTR_L_off ()	Fuel
function NGX_PUMPCTR_L_on ()	Fuel
function NGX_PUMPCTR_L_toggle ()	Fuel
function NGX_PUMPCTR_R_off ()	Fuel
function NGX_PUMPCTR_R_on ()	Fuel
function NGX_PUMPCTR_R_toggle ()	Fuel
function NGX_PUMPS1_off ()	Fuel
function NGX_PUMPS1_on ()	Fuel
function NGX_PUMPS1_toggle ()	Fuel
function NGX_PUMPS1and2_off ()	Fuel
function NGX_PUMPS1and2_on ()	Fuel
function NGX_PUMPS2_off ()	Fuel
function NGX_PUMPS2_on ()	Fuel
function NGX_PUMPS2_toggle ()	Fuel
function NGX_PUMPSCTR_off ()	Fuel
function NGX_PUMPSCTR_on ()	Fuel
function NGX_PUMPSCTR_toggle ()	Fuel
function NGX_RECIRC_L_auto ()	Bleed and Packs (including APU bleeds)
function NGX_RECIRC_L_off ()	Bleed and Packs (including APU bleeds)
function NGX_RECIRC_L_toggle ()	Bleed and Packs (including APU bleeds)
function NGX_RECIRC_R_auto ()	Bleed and Packs (including APU bleeds)
function NGX_RECIRC_R_off ()	Bleed and Packs (including APU bleeds)
function NGX_RECIRC_R_toggle ()	Bleed and Packs (including APU bleeds)
function NGX_RecircANDIsol_auto ()	Bleed and Packs (including APU bleeds)
function NGX_RecircANDIsol_off ()	Bleed and Packs (including APU bleeds)
function NGX_RUD_TRIM_left ()	Trimmings
function NGX_RUD_TRIM_right ()	Trimmings
function NGX_RUD_TRIM_show ()	Trimmings
function NGX_RUD_TRIM_stop ()	Trimmings
function NGX_SIGNS_CHIME_off ()	Other
function NGX_SIGNS_CHIME_on ()	Other
function NGX_SIGNS_CHIME_toggle ()	Other
function NGX_SIGNS_SEAT_auto ()	Other
function NGX_SIGNS_SEAT_off ()	Other
function NGX_SIGNS_SEAT_on ()	Other
function NGX_SPOILER_100 ()	Other
function NGX_SPOILER_50 ()	Other

function NGX_SPOILER_arm ()	Other
function NGX_SPOILER_detent ()	Other
function NGX_SPOILER_off ()	Other
function NGX_STBY_POWER_Auto ()	Electrics (including APU)
function NGX_STBY_POWER_bat ()	Electrics (including APU)
function NGX_STBY_POWER_off ()	Electrics (including APU)
function NGX_SYSTEM_ANN_reset ()	Warnings
function NGX_TAXI_ALL_off ()	Lights, external
function NGX_TAXI_ALL_on ()	Lights, external
function NGX_TAXI_ALL_toggle ()	Lights, external
function NGX_TAXI_off ()	Lights, external
function NGX_TAXI_on ()	Lights, external
function NGX_TAXI_toggle ()	Lights, external
function NGX_TCAS_test ()	Transponder
function NGX_TURNOFF_BOTH_off ()	Lights, external
function NGX_TURNOFF_BOTH_on ()	Lights, external
function NGX_TURNOFF_BOTH_toggle ()	Lights, external
function NGX_TURNOFF_LEFT_off ()	Lights, external
function NGX_TURNOFF_LEFT_on ()	Lights, external
function NGX_TURNOFF_LEFT_toggle ()	Lights, external
function NGX_TURNOFF_RIGHT_off ()	Lights, external
function NGX_TURNOFF_RIGHT_on ()	Lights, external
function NGX_TURNOFF_RIGHT_toggle ()	Lights, external
function NGX_W_HEAT_ALL_LEFT_off ()	Window Heat
function NGX_W_HEAT_ALL_LEFT_on ()	Window Heat
function NGX_W_HEAT_ALL_LEFT_toggle ()	Window Heat
function NGX_W_HEAT_ALL_off ()	Window Heat
function NGX_W_HEAT_ALL_on ()	Window Heat
function NGX_W_HEAT_ALL_RIGHT_off ()	Window Heat
function NGX_W_HEAT_ALL_RIGHT_on ()	Window Heat
function NGX_W_HEAT_ALL_RIGHT_toggle ()	Window Heat
function NGX_W_HEAT_ALL_toggle ()	Window Heat
function NGX_W_HEAT_L_FWD_off ()	Window Heat
function NGX_W_HEAT_L_FWD_on ()	Window Heat
function NGX_W_HEAT_L_FWD_toggle ()	Window Heat
function NGX_W_HEAT_L_SIDE_off ()	Window Heat
function NGX_W_HEAT_L_SIDE_on ()	Window Heat
function NGX_W_HEAT_L_SIDE_toggle ()	Window Heat
function NGX_W_HEAT_R_FWD_off ()	Window Heat
function NGX_W_HEAT_R_FWD_on ()	Window Heat
function NGX_W_HEAT_R_FWD_toggle ()	Window Heat
function NGX_W_HEAT_R_SIDE_off ()	Window Heat
function NGX_W_HEAT_R_SIDE_on ()	Window Heat
function NGX_W_HEAT_R_SIDE_toggle ()	Window Heat
function NGX_W_HEAT_TEST_off ()	Window Heat
function NGX_W_HEAT_TEST_ovht ()	Window Heat
function NGX_W_HEAT_TEST_pwr ()	Window Heat
function NGX_WARN_FIRE_reset ()	Warnings
function NGX_WARN_MASTER_reset ()	Warnings
function NGX_WHEEL_WELL_off ()	Lights, external
function NGX_WHEEL_WELL_on ()	Lights, external
function NGX_WHEEL_WELL_toggle ()	Lights, external

function NGX_WING_off ()	Lights, external
function NGX_WING_on ()	Lights, external
function NGX_WING_toggle ()	Lights, external
function NGX_Wiper_both_dec ()	Wiper
function NGX_Wiper_both_inc ()	Wiper
function NGX_Wiper_L_calc ()	Wiper
function NGX_Wiper_L_dec ()	Wiper
function NGX_Wiper_L_high ()	Wiper
function NGX_Wiper_L_inc ()	Wiper
function NGX_Wiper_L_int ()	Wiper
function NGX_Wiper_L_low ()	Wiper
function NGX_Wiper_L_park ()	Wiper
function NGX_Wiper_L_show ()	Wiper
function NGX_Wiper_R_calc ()	Wiper
function NGX_Wiper_R_dec ()	Wiper
function NGX_Wiper_R_high ()	Wiper
function NGX_Wiper_R_inc ()	Wiper
function NGX_Wiper_R_int ()	Wiper
function NGX_Wiper_R_low ()	Wiper
function NGX_Wiper_R_park ()	Wiper
function NGX_Wiper_R_show ()	Wiper
function NGX_XPND_MODE_cycle ()	Transponder
function NGX_XPND_MODE_dec ()	Transponder
function NGX_XPND_MODE_inc ()	Transponder
function NGX_XPND_MODE_show ()	Transponder
function NGX_YAW_DAMPER_off ()	Other
function NGX_YAW_DAMPER_on ()	Other
function NGX_YAW_DAMPER_toggle ()	Other
function Timer ()	System functions

Appendix 2 — example page from documented control assignments

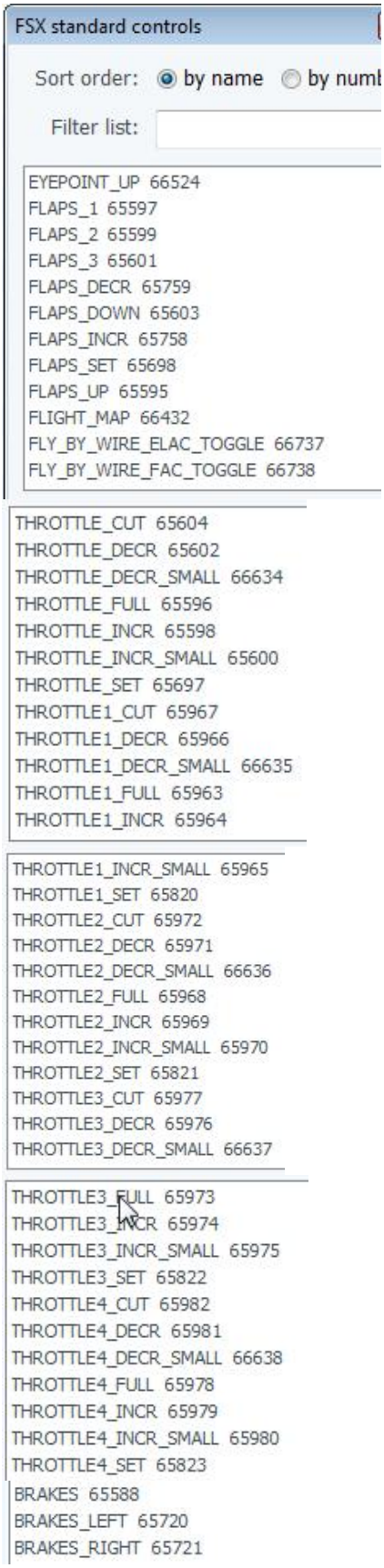


Below: ^ indicates a centre off position; S means sprung to return to centre when pressure released.

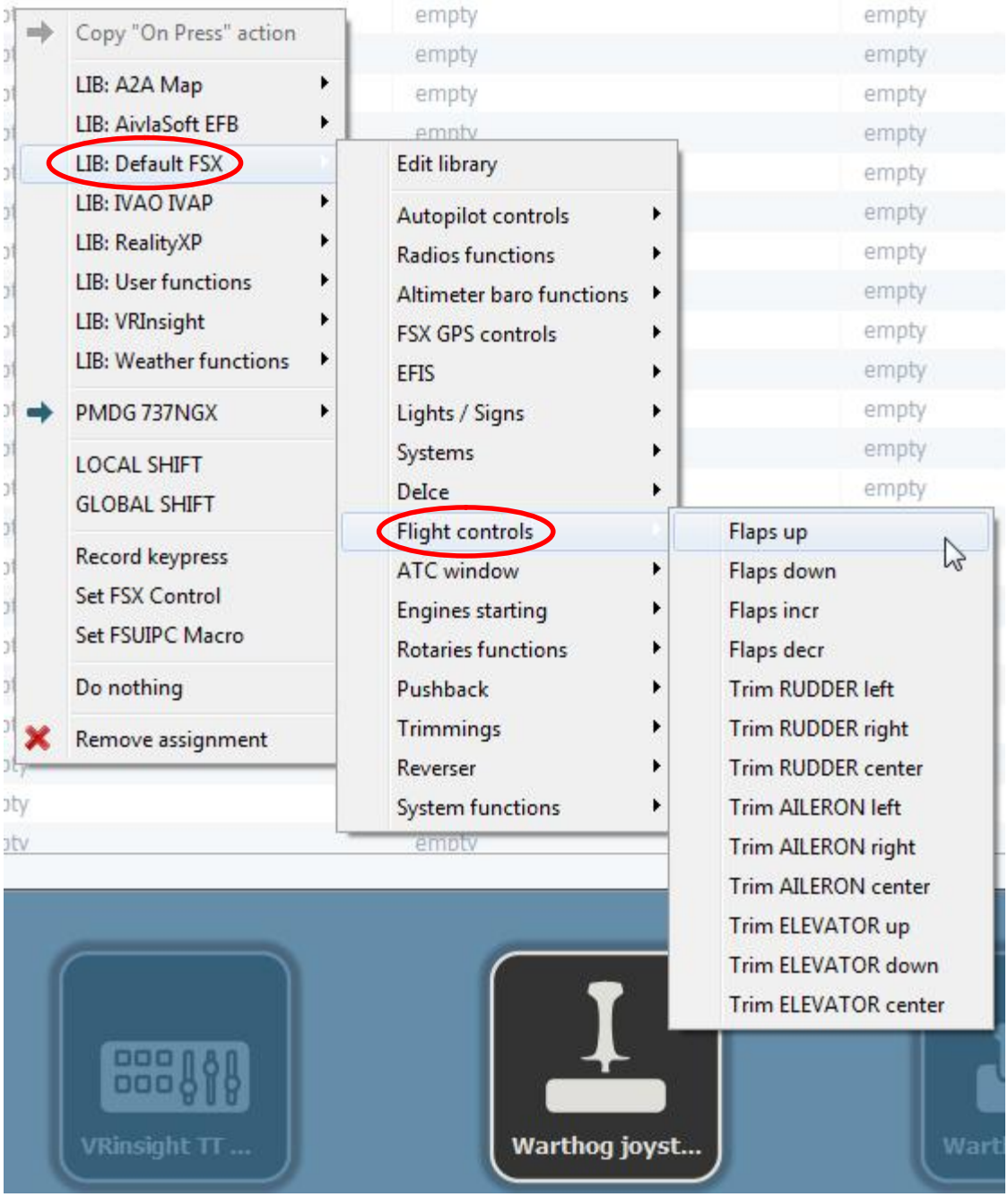
Throttle base — button/switch numbers &c

Control label	No.	Assignment
ENG L switch	↑ 16 on, off↓	**Pre-taxi routine** BC_After_Engine_Start (in lib-user.lua)
ENG R switch	↑ 17 on, off↓	**Runway exit cleanup routine** BC_Cleanup (in lib-user.lua)
IGN LEFT switch	↑31S, ^, 18↓	↑**All cockpit lights inc**, —, ↓**All cockpit lights dec** NGX_LIGHT_ALL_inc, NGX_LIGHT_ALL_dec (not inc/dec *fast*)
IGN left **SHIFTED**	↑31 **shifted**	↑**ENG1 cutoff** NGX_ENG1_cutoff
IGN RIGHT switch	↑32S, ^, 19↓	↑**ALT INTV**,^ , ↓**SPD INTV** NGX AP ALT INTV,^,NGX AP SPD INTV
IGN right **SHIFTED**	↑32 **shifted**	↑**ENG2 cutoff** NGX_ENG2_cutoff
APU START/OFF sw	↑20 on, off↓	↑**Dome light cycle** ↑ NGX_DOME_CYCLE
Silence Horn btn	21	**Set to local barometric pressure** (B) LINDA (Keyboard)
FLAPS switch	↑22 UP, ^, 23 DN ↓	**Flaps less** ↑, **Flaps more** ↓ FSX: FLAPS_DECR, FSX: FLAPS_INCR
EAC switch	↑24 on, off↓	↑ **parking brakes on**, ↓ **parking brakes off** (in lib-user.lua) (BC_PARKING_BRAKE_Set, BC_PARKING_BRAKR_Release)
RDR ALTM switch	↑25 on, off↓	↑ **HUD up**, ↓ **HUD down** NGX_HGS_up, NGX_HGS_down
A/P toggle button	26	**TO/GA** NGX_AP_TOGA
A/P mode switch	↑27, ^, 28↓	↑**Gear up**, **Gear off**, ↓Gear down NGX_GEAR_up, NGX_GEAR_off, NGX_GEAR_down
A/P mode switch **SHIFTED**	↑27 **shifted**, ^, 28↓ **shifted**	↑**APU on and start**, ↓**APU off** On= NGX_APU_GEN_BOTH_on, Rpt=NGX_APU_Start_or_ShowEGT, Off=NGX_APU_GEN_BOTH_off
Silence Horn button **SHIFTED**	btn 21 **shifted**	(push) **Gate arrival** BC_Gate_Arrival (in lib-user.lua)
AXES:		
"Incr Decr" lever	Slider	**Steering tiller**

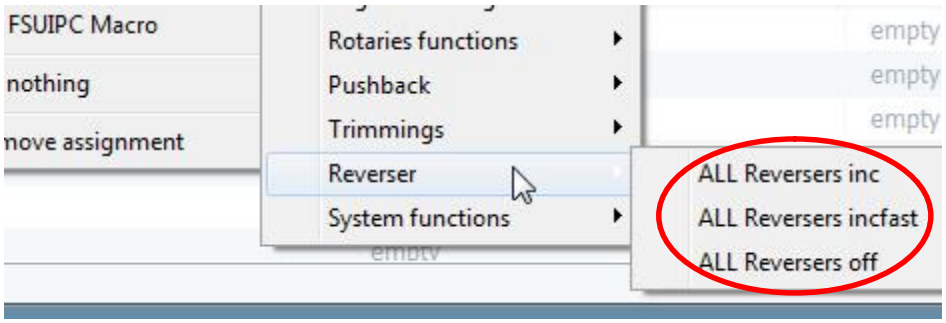
Appendix 3 — fsx standard controls



Examples of LIB: Default FSX



Also —



Appendix 4 — Combo I standard assignments

See `C:\fsx\Modules\linda-cfg\aircrafts\PMDG 737NGX` — `config.mcp.lua & config-mcp.default`