# Xsaitekpanels


Ver 3.00

William R Good

# Xsaitekpanels Users Manual

Purpose - One plugin to allow support for multiple Saitek Pro Flight panels in X-Plane 10.20+ using Linux, Windows and the Mac.

Thanks to Alan Ott's HIDAPI (https://github.com/signal11/hidapi) I am able to now support Linux, Windows and Mac with my plugin Xsaitekpanels.

Many thanks go to the following people who throughout this long process allowed me to get my plugin to this point. Sandy Barbour, Tom Sedge, Michal (uglyDwarf), Philipp, Stefan Oberender, Andy Goldstein, Bernhard (JaXXoN), Fabio, Marcos Lemos, Steven King and Chris Strosser.

With this plugin I am supporting the switch, radio, multi and BIP panels.

x737 support is now outside of the plugin to provide more configuration to the user and not the programmer.

There is a preference file called xsaitekpanels.ini located in the Xsaitekpanels plugins folder and on the next page will explain it in detail.

I will also list the data reference or command reference that I am using for each switch or knob.

*  =  X-Plane normal data reference and command reference.
**  =  Remapable command reference.
***  =  Remapable data reference.

**********

## How to install

This is a fat plugin so the folder named Xsaitekpanels which have a 64 folder, xsaitekpanels.ini, D2B_config.txt, D2B_config2.txt, D2B_config3.txt, lin.xpl, win.xpl and mac.xpl in it and needs to reside in /Resources/plugins/ folder. Inside the 64 folder is lin.xpl and win.xpl.

Linux users only you will need to install a new 51-Xsaitekpanels.rules in /lib/udev/rules.d for these Saitek panels to be seen by this plugin and is included in this archive. You will also have to reboot for the rule to become active and you should have a saitekpanels files in your dev directory if all is working correctly. After you have that working you might experience some interaction with the panels and your mouse. If that is the case please read the “Linux_Users_Please_Read.txt” for further imformation to help resolve that issue.

**********

## Cold and Dark

When Xsaitekpanels starts it is normal for the displays on the radio and multi panels to be dark. This is because they do not know where there switches are and for that reason do not know what to display. I have asked many people for a solution but have not found any so this is the way it is and this is how I resolve it for myself. When X-Plane is starting as a preflight task I move one switch on my switch, 2 radio and multi panels and when X-Plane starts they display correctly.

# Support

With the last couple of requests for help I think I need to set some rules.

First Read this manual as it is why I took the time to write it.

Second read the change logs as I try to give the details that changed in that version.

**Third. If you are having a issue with Xsaitekpanels not working I am glad to help but if you PM or contact me and do not include a Log.txt I will consider that you are not serious and will delete the PM or not respond to you. I am to busy to beg someone for information that should be given on the first contact.**

The Log.txt is located in the root of X-Plane the same place the executable resides and if you need any more info use Google.

This is the kind of info I am looking for.

http://forums.x-plane.org/index.php?/announcement/15-tech-support-information-we-need-information/

# Source Code

This project and my projects going forward will be open source and reside at https://github.com/sparker256/xsaitekpanels for anyone interested in how A HID plugin is created for all platforms.

For the people that wanted to donate use the donate button there. I am not requesting donations but when people ask me where they can donate I tried to make it as easy as possible.

# Supported Platforms

## Linux

Supported platforms at this release are any current Ubuntu version. I am using Ubuntu 20.04 LTS and 18.04 LTS with X-Plane 10.51r2 & 11.53r1.  You will need to install a new 51-Xsaitekpanels.rules in /lib/udev/rules.d for these Saitek panels to be seen by this plugin. It is included in this archive.

**********

## Windows

Supported and tested platforms at this release is Windows 10 and X-Plane_10.51r2 and X-Plane 11.53r1. When X-Plane initially loads the radio and the multi panel may be dark. This is because initially the switches are not read so do not know what to display. Move any switch on the radio or multi panel and if the battery and avionics switch are on they should display the appropriate items.

Windows 10 users need to be aware there is a change in how USB is handled and it breaks part of Xsaitekpanels.

There is also a program called codelegend_APME_fix_for_Saitek.zip at the bottom of this page that I used on my Windows 10 install with great success.

http://www.codelegend.com/idealflight/download/default.htm

There is also a problem if Saitek Smark Technology (SST) Profile Launcher is running and you have a BIP plugged in it will conflict with Xsaitekpanels. This will cause a crash of X-Plane but if you disable SST all will be fine. My thanks to skyhopper who helped us find this issue.

There is also a issue if you are running the BIP Profiler that will crash X-Plane because Xsaitekpanels need full access to all the panels.

**********

## Mac

I build on OSX 10.10.5 and have run it on 10.11.6. When X-Plane initially loads the radio and the multi panel may be dark. This is because initialy the switches are not read so do not know what to display. Move any switch on the radio or multi panel and if the battery and avionics switch are on they should display the appropriate items. Supported X-Plane versions are X-Plane 10.51r2 & 11.53r1.

# xsaitekpanels.ini

xsaitekpanels.ini is configuration file for Xsaitekpanels that allows your menu selections to stick and much more. For this to work you have to edit the xsaitekpanels.ini but you can do this while X-Plane is running and then reload a plane or use the reload xsaitekpanels.ini button and the menu selection will reflect the changes. **At this point in time there is no way to write changes to the xsaitekpanels.ini except to edit it yourself but this might change in future releases but low on my priority list at this time.**

xsaitekpanels.ini comes from the archive residing in the plugin folder but can also reside in the aircraft folder to allow different configurations for different aircraft.

**Starting with version 2.70 you can also have aircraft name _xsaitekpanels.ini so for the default Cessna it would be Cessna_172SP_xsaitekpanels.ini.**

I first look in the current aircraft folder and if I cannot find it I will look in the Xsaitekpanels plugin folder.

The second function of the configuration file is to disable – enable – remap all switches on the switch , multi and some of the radio panel. I have re organized the xsaitekpanels.ini to put the configuration information just above the command, datareference or value. This is a work in progress and will be expanding with other features in later releases and am looking forward to suggestions on where to go.

Look into the xsaitekpanels.ini in the archive for a better idea of how this all works.

This is hard to explain but if you come up with better words please let me know and they will be in the manual.

# Xsaitekpanelsdatarefs.text

In the archive there is a file called Xsaitekpanelsdatarefs.text which is a list of all the datarefs that Xsaitekpanels creates with the biggest part being the status of all the switches of the Saitek panels. You do not to put it anywhere it is just information as to what is available. You can see the status live using DataRefTool.

# How to create a custom xsaitekpanels.ini for your add on aircraft.

I have used the following techniques and thought it would be good to broadcast to a wider audience.

To create a custom xsaitekpanels.ini for use in your add on aircraft you will need some information and some tools.

First you need to know what datarefs are being used by the aircraft you are trying to control with Xsaitekpanels. This used to be very difficult but there is a new tool that I have been using and it solves all the previous issues.

The only tool I use now is called DataRefTool https://github.com/leecbaker/datareftool/releases which is a open source replacement for DataRefEditor that I used before.

Now copy the un modified xsaitekpanels.ini from the Xsaitekpanels folder to the aircraft folder you are working with.

Now we can start X-Plane and select your aircraft and when you start DataRefTool and you should see the new datarefs that it found for you current aircraft. They are live so if it is a avionics master switch dataref and you click on it in the virtual cockpit you should see the value for it change and the dataref will change color for 10 seconds to help you find it.

Next step is to start making the saitek panels do the same thing with out the mouse. Open the xsaitekpanel.ini copy that you put in your aircraft folder in your editor of choice and find the switch you want to work with. To help with this DataRefTool has a feature that you can copy the dataref you found and paste it into xsaitekpanels.ini which saves a lot of time.  Work through them one at a time and you will slowly remove much of the need for a mouse when flying your aircraft which is my goal.

**FlyWithLua**

First FlyWithLua is a Lua 5.1 script engine with the power of a C/C++ plugin that is as easy as BASIC and can be found here.

http://forums.x-plane.org/index.php?s=374bcbfcf9c2e8e06b71d0d79ddcfe73&app=downloads&showfile=17468

I started using FlyWithLua shortly after it came out as a replacement for Button2DataRef that I was using with my Saitek TPM to map the switches to my liking. At that time I did not see the need to use it with Xsaitekpanels but the more I used it I saw how I could use it as a extension of Xsaitekpanels and that is how I use it now.

It has allowed me to add new aircraft that may have a feature that only it has and before I would have to rewrite my plugin to support it I can now write a script for that feature. As a side benefit my users can do the same thing as I do with very little programing knowledge and have your aircraft better supported.

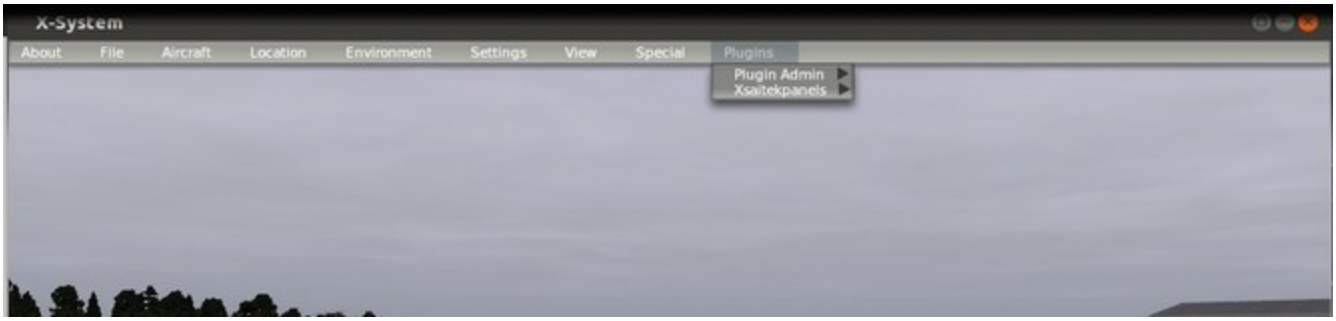
Now to use it with Xsaitekpanels you need to download FlyWithLua and install it. I would highly recommend looking at the manual that comes with it to get a better understanding of how I am using it. Next look in the folder in the archive called FlyWithLua with two subfolders called Modules and Scripts which match the folders for the FlyWithLua plugin.

In the Modules folder there are five files that need to be copied to /Resources/plugins/FlyWithLua/Modules folder. These define every switch on the switch, radio1, radio2, radio3 and the multipanel so FlyWithLua can talk to them.

In the Scripts folder there are scripts that I have configured for different aircraft but only copy the ones that you also have that aircraft. They need to be copied to /Resources/plugins/FlyWithLua/Scripts folder.

In most of the scripts I tried to add comments to help understand what I am doing and if it is not clear enough ask questions as it will only make Xsaitekpanels better.

## Plugin Menu Selections




## Xsaitekpanels Menu Selections


## Xsaitekpanels Multipanel Menu


**Top Section**

Allows control of the sensitivity of the silver knob.

This is the number of switch pulses from the knob per command.

The default is 3 switch pulses per command.

**Middle Section**

Allows control of the sensitivity of the trim.

The default is 1

**Lower Section**

Allows you to enable of disable the auto throttle switch.

# Xsaitekpanels Switchpanel Menu


Bat Alt Normal matches the labels on the switch panel.

Alt Bat Cessna matches the position of a Cessna 172.

Start Switch Old is the original style of starting the engines.

Start Switch New allows easier starting of turbines and jets.

As you can see the complete switch panel can be enabled – disabled - remapable.

This widget allows a quick method to test out what you want to do before changing the xsaitekpanels.ini file.

# Xsaitekpanels Radiopanel Menu


**Top Section**

Allows control of the sensitivity of the silver knob.

This is the number of switch pulses from the knob per command.

The default is 3 switch pulses per command.

**Middle Section**

Allows selection of one or two ADF Tuners

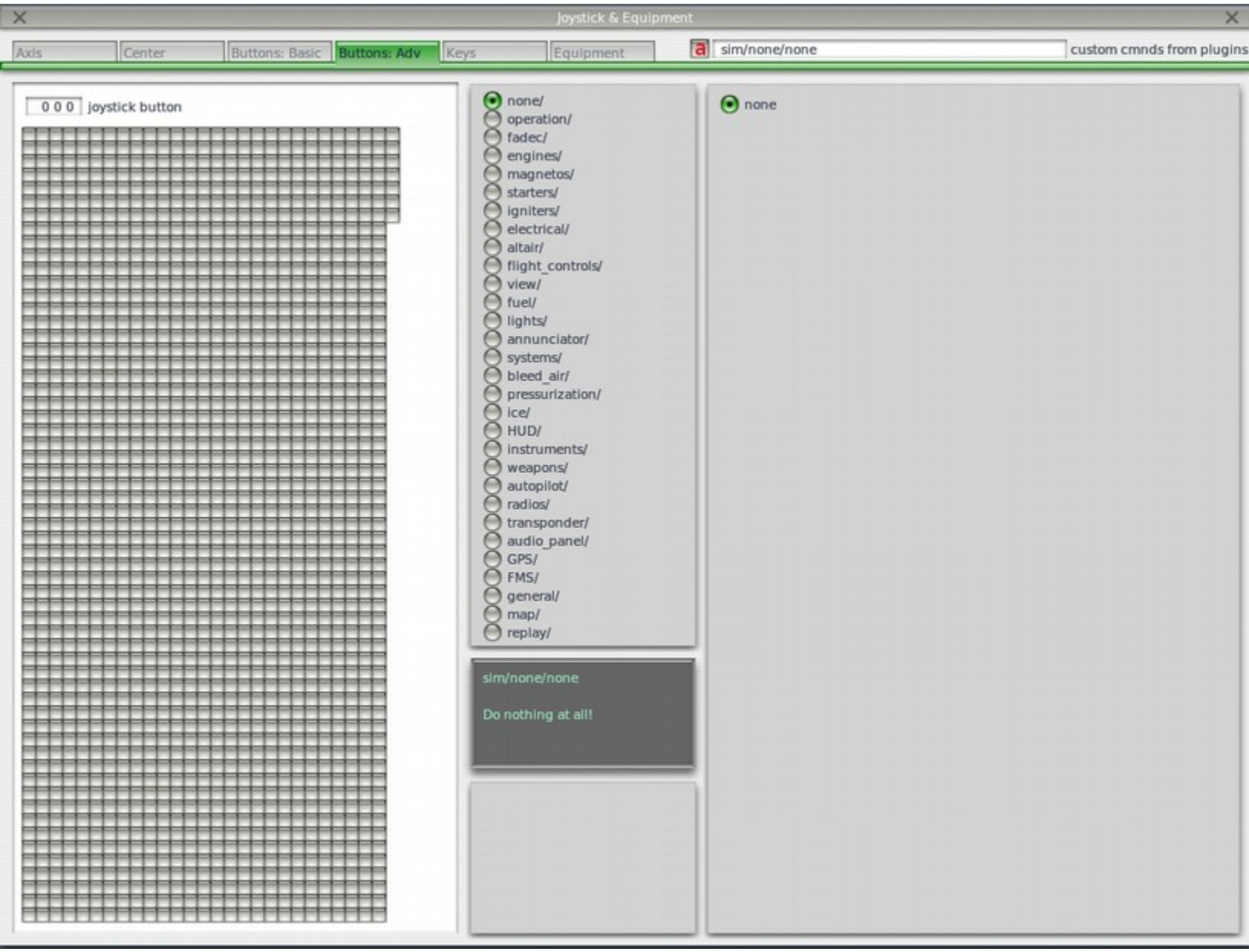
**Lower Section**

Allows selection of QNH in inHg or hPa.

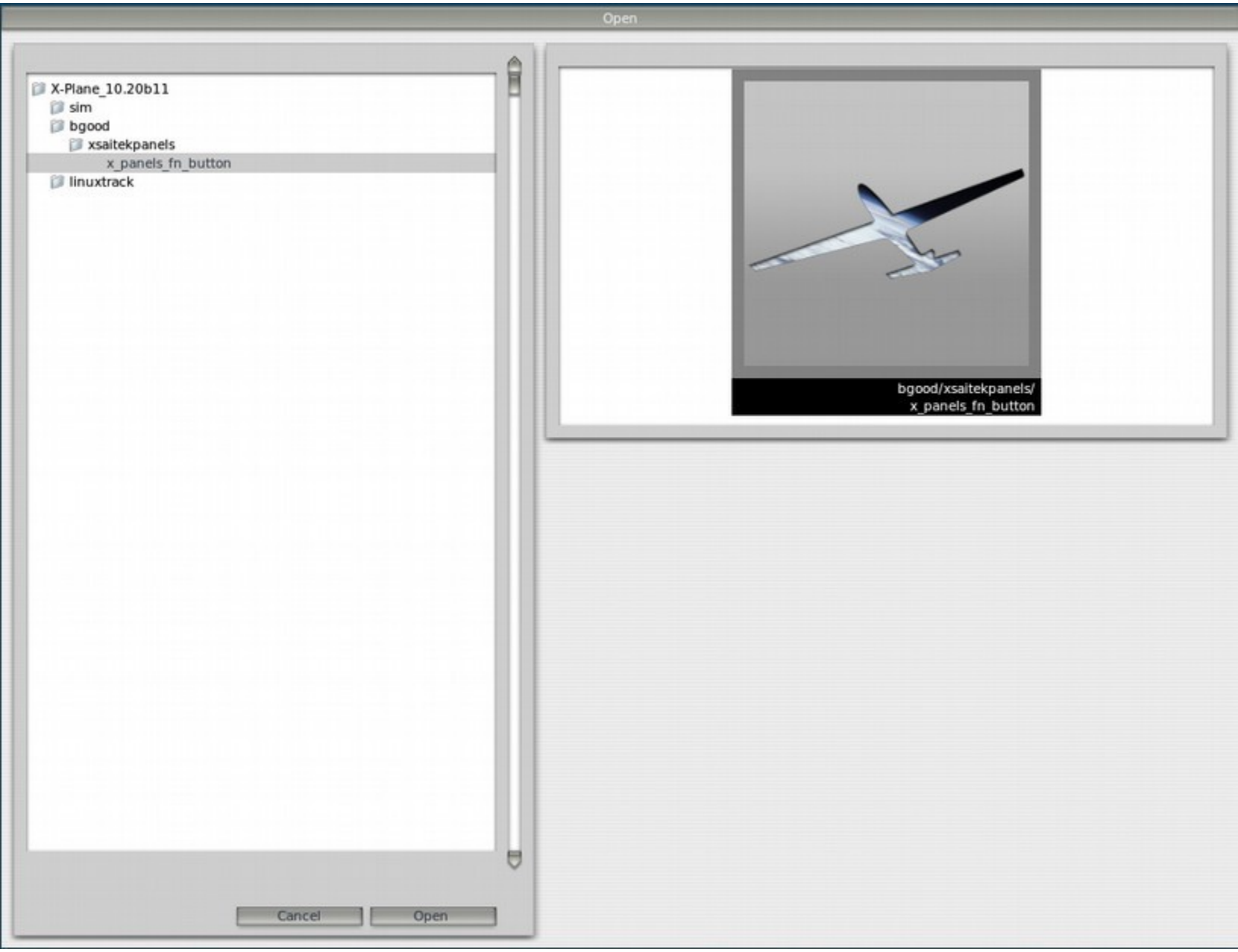
There is a Fn key that allows expanded use of the switches on the panels.
It can be mapped for a joystick button or a keyboard key combo.
The path is bgood/xsaitekpanels/x_panels_fn_button.

I have added a second Fn key to help in starting the left engine using the switch panel.
The path is bgood/xsaitekpanels/left_start_fn_button. Use the following pages to get it working.

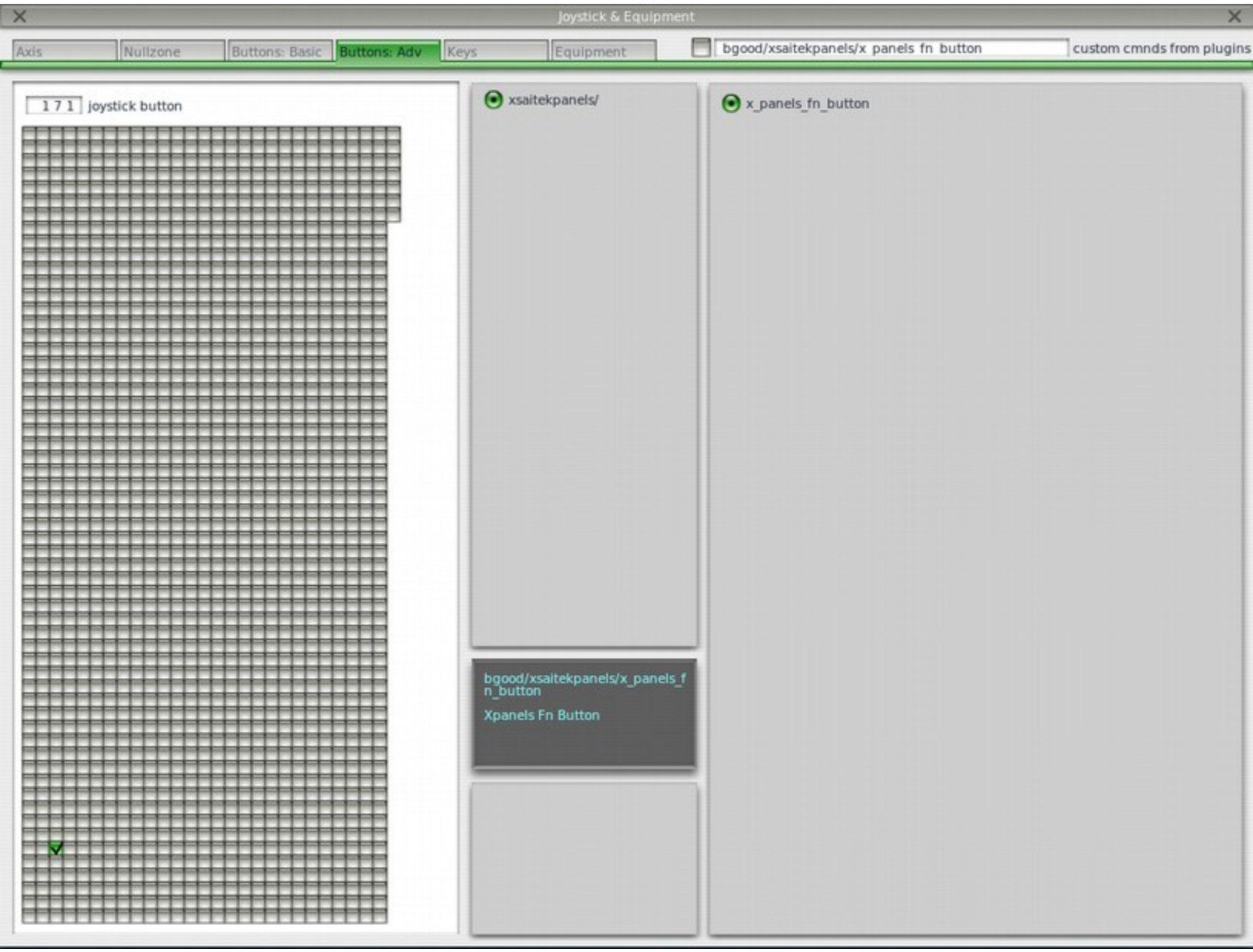
Joystick buttons are setup using Buttons:Adv pane in the Joystick & Equipment window, that is accessible through Settings / Joystick, Keys & Equipment menu. Press the desired joystick button, and now click the box next to the tabs I have marked with a red a. An Open dialog will appear.



Select the x_panels_fn_button under bgood > xsaitekpanels in the left hand pane of the dialog and then press open.



Finally select the desired command (x_panels_fn_button).



**********

Switch Panel (http://www.saitek.com/uk/prod/switch.html)

**********

The switches are mapped in the following manner for the default behavior. The xsaitekpanels.ini allows you to change from the default and what is possible is described in that file. For more info look in the xsaitekpanels.ini section of this manual.

For the engine I look at the number of engines and control that many.
In the present version I support four engines.

There is a check box for old and new style for the starter switch.

* = Old Original way

*n = New where you can use the starter switch to start a turbine or a jet. You may also need the apu running but the starter switch will do the rest.

**Engine mags off switch**

This grounds out the mags and stops the engine.

* X = 1 thru 4
* MagOffX = XPLMFindCommand("sim/magnetos/magnetos_off_X")

```
*n IgnSwitchArray = XPLMFindDataRef("sim/cockpit2/engine/actuators/ignition_key");
*n XPLMSetDatavi(IgnSwitchArray, ignition_key_array, 0, 8);
*n ignition_key_array[0] = 0 This is for The first engine.

*n EngnMixt = XPLMFindDataRef("sim/flightmodel/engine/ENGN_mixt");
*n XPLMSetDatavf(EngnMixt, engn_mixt, 0, 8);
*n engn_mixt[0] = 0 This is for The first engine.

*n IgniterOn = XPLMFindDataRef("sim/cockpit2/engine/actuators/igniter_on");
*n XPLMSetDatavi(IgniterOn, igniter_on, 0,  8);
*n igniter_on[0] = 0 This is for The first engine.

*n BleedAirMode = XPLMFindDataRef("sim/cockpit/pressure/bleed_air_mode");
*n XPLMSetDatai(BleedAirMode, bleed_air_mode);
*n bleed_air_mode = 0
```

**Engine mags right switch**

This selects only the right mags to test in run up.

* X = 1 thru 4
* MagRightX = XPLMFindCommand("sim/magnetos/magnetos_right_X")

```
*n IgnSwitchArray = XPLMFindDataRef("sim/cockpit2/engine/actuators/ignition_key");
*n XPLMSetDatavi(IgnSwitchArray, ignition_key_array, 0, 8);
*n ignition_key_array[0] = 1; // This is for The first engine

*n BleedAirMode = XPLMFindDataRef("sim/cockpit/pressure/bleed_air_mode");
*n XPLMSetDatai(BleedAirMode, bleed_air_mode);
*n bleed_air_mode = 0
```

**Engine mags left switch**

This select only the left mags to test in run-up.

* X = 1 thru 4
* MagLeftX = XPLMFindCommand("sim/magnetos/magnetos_left_X")

```
*n IgnSwitchArray = XPLMFindDataRef("sim/cockpit2/engine/actuators/ignition_key");
*n XPLMSetDatavi(IgnSwitchArray, ignition_key_array, 0, 8);
*n ignition_key_array[0] = 2; // This is for The first engine

*n BleedAirMode = XPLMFindDataRef("sim/cockpit/pressure/bleed_air_mode");
*n XPLMSetDatai(BleedAirMode, bleed_air_mode);
*n bleed_air_mode = 0
```

**Engine mags both switch**

This selects both mags and is the normal position when the engine is running.

* X = 1 thru 4
* MagBothX = XPLMFindCommand("sim/magnetos/magnetos_both_X")

```
*n IgnSwitchArray = XPLMFindDataRef("sim/cockpit2/engine/actuators/ignition_key");
*n XPLMSetDatavi(IgnSwitchArray, ignition_key_array, 0, 8);
*n ignition_key_array[0] = 3; // This is for The first engine

*n IgniterOn = XPLMFindDataRef("sim/cockpit2/engine/actuators/igniter_on");
*n XPLMSetDatavi(IgnSwitchArray, ignition_key_array, 0, 8);
*n igniter_on[0] = 0 This is for The first engine.

*n BleedAirMode = XPLMFindDataRef("sim/cockpit/pressure/bleed_air_mode");
*n XPLMSetDatai(BleedAirMode, bleed_air_mode);
*n bleed_air_mode = 0
```

**Engine start switch**

This engages the starter.

* X = 1 thru 4
* EngStartX = XPLMFindCommand("sim/starters/engage_starter_X");

```
*n IgnSwitchArray = XPLMFindDataRef("sim/cockpit2/engine/actuators/ignition_key");
*n XPLMSetDatavi(IgnSwitchArray, ignition_key_array, 0, 8);
*n ignition_key_array[0] = 4; // This is for The first engine

*n EngnMixt = XPLMFindDataRef("sim/flightmodel/engine/ENGN_mixt");
*n XPLMSetDatavf(EngnMixt, engn_mixt, 0, 8);
*n engn_mixt[0] = 1 This is for The first engine.

*n IgniterOn = XPLMFindDataRef("sim/cockpit2/engine/actuators/igniter_on");
*n XPLMSetDatavi(IgniterOn, igniter_on, 0,  8);
*n igniter_on[0] = 1 This is for The first engine.

*n BleedAirMode = XPLMFindDataRef("sim/cockpit/pressure/bleed_air_mode");
*n XPLMSetDatai(BleedAirMode, bleed_air_mode);
*n bleed_air_mode = 4

*n AcfEnType = XPLMFindDataRef("sim/aircraft/prop/acf_en_type");
*n XPLMGetDatavi(AcfEnType, acf_en_type, 0, 8);

*n If turbine of jet acf_en_type[0] > 1 so use mixture, igniters and bleed air
  *n (acf_en_type[0] > 1) {
     *n XPLMSetDatavf(EngnMixt, engn_mixt, 0, 8);
     *n XPLMSetDatavi(IgniterOn, igniter_on, 0,  8);
     *n PLMSetDatai(BleedAirMode, bleed_air_mode);
  *n {
```

There is a menu selection called "Bat Alt Normal or Alt Bat Cessna"
	In normal it is as labeled on the panel.
	In Cessna match the position of a Cessna 172SP

**Master Battery switch**
	In on position turns on the number of batteries in the plane.
	* BatOnX = XPLMFindCommand("sim/electrical/battery_X_on")
	** bat_master_switch_on = getOptionToString("bat_master_switch_on_cmd");
	** BatMasterSwitchOnCmd   = XPLMFindCommand(bat_master_switch_on.c_str());

	In off position turns off the number of batteries in the plane.
	X = 1 thru 2
	* BatOffX = XPLMFindCommand("sim/electrical/battery_X_off")
	** bat_master_switch_off = getOptionToString("bat_master_switch_cmd");
	** BatMasterSwitchOffCmd   = XPLMFindCommand(bat_master_switch_off.c_str());

**Master Alternator switch**
	In on position turns on the number of generators as engines.
	* GenOnX = XPLMFindCommand("sim/electrical/generator_X_on")
	** alt_master_switch_on = getOptionToString("alt_master_switch_on_cmd");
	** AltMasterSwitchOnCmd   = XPLMFindCommand(alt_master_switch_on.c_str());

	In off position turns off the number of generators as engines.
	X = 1 thru 4
	* GenOffX = XPLMFindCommand("sim/electrical/generator_X_off")
	** alt_master_switch_off = getOptionToString("alt_master_switch_off_cmd");
	** AltMasterSwitchOffCmd   = XPLMFindCommand(alt_master_switch_off.c_str());

**Avionics master switch**
	In the on position turns on all radio equipment.
	* AvLtOn = XPLMFindCommand("sim/systems/avionics_on")
	** av_master_switch_on = getOptionToString("av_master_switch_on_cmd");
	** AvMasterSwitchOnCmd   = XPLMFindCommand(av_master_switch_on.c_str());

	In the off position turns off all radio equipment.
	* AvLtOff  = XPLMFindCommand("sim/systems/avionics_off")
	** av_master_switch_off = getOptionToString("av_master_switch_off_cmd");
	** AvMasterSwitchOffCmd   = XPLMFindCommand(av_master_switch_off.c_str());

**Fuel pump switch**
	In on position turns on the same number of fuel pumps as engines.
	* FuelPumpOnX = XPLMFindCommand("sim/fuel/fuel_pump_X_on")
	** fuel_pump_switch_on = getOptionToString("fuel_pump_switch_on_cmd");
	** FuelPumpOnCmd   = XPLMFindCommand(fuel_pump_switch_on.c_str());
	** fuel_pump2_switch_on = getOptionToString("fuel_pump2_switch_on_cmd");
	** FuelPump2OnCmd   = XPLMFindCommand(fuel_pump2_switch_on.c_str());
	** fuel_pump3_switch_on = getOptionToString("fuel_pump3_switch_on_cmd");
	** FuelPump3OnCmd   = XPLMFindCommand(fuel_pump3_switch_on.c_str());
	** fuel_pump4_switch_on = getOptionToString("fuel_pump4_switch_on_cmd");
	** FuelPump4OnCmd   = XPLMFindCommand(fuel_pump4_switch_on.c_str());


	In off position turns off the same number of fuel pumps as engines.
	X = 1 thru 4
	* FuelPumpOffX = XPLMFindCommand("sim/fuel/fuel_pump_X_off")
	** fuel_pump_switch_off = getOptionToString("fuel_pump_switch_off_cmd");
	** FuelPumpOffCmd   = XPLMFindCommand(fuel_pump_switch_off.c_str());
	** fuel_pump2_switch_off = getOptionToString("fuel_pump2_switch_off_cmd");
	** FuelPump2OffCmd   = XPLMFindCommand(fuel_pump2_switch_off.c_str());
	** fuel_pump3_switch_off = getOptionToString("fuel_pump3_switch_off_cmd");
	** FuelPump3OffCmd   = XPLMFindCommand(fuel_pump3_switch_off.c_str());
	** fuel_pump4_switch_off = getOptionToString("fuel_pump4_switch_off_cmd");
	** FuelPump4OffCmd   = XPLMFindCommand(fuel_pump4_switch_off.c_str());

**De-Ice switch**

In the on position turns on anti-ice.

```
*  AntiIce = XPLMFindDataRef("sim/cockpit/switches/anti_ice_on")
** deice_switch_on = getOptionToString("deice_switch_on_cmd");
** DeiceOnCmd   = XPLMFindCommand(deice_switch_on.c_str());
** deice2_switch_on = getOptionToString("deice2_switch_on_cmd");
** DeiceOnCmd2   = XPLMFindCommand(deice2_switch_on.c_str());
** deice3_switch_on = getOptionToString("deice3_switch_on_cmd");
** DeiceOnCmd3   = XPLMFindCommand(deice3_switch_on.c_str());
** deice4_switch_on = getOptionToString("deice4_switch_on_cmd");
** DeiceOnCmd4   = XPLMFindCommand(deice4_switch_on.c_str());
** deice5_switch_on = getOptionToString("deice5_switch_on_cmd");
** DeiceOnCmd5   = XPLMFindCommand(deice5_switch_on.c_str());
** deice6_switch_on = getOptionToString("deice6_switch_on_cmd");
** DeiceOnCmd6   = XPLMFindCommand(deice6_switch_on.c_str());
** deice7_switch_on = getOptionToString("deice7_switch_on_cmd");
** DeiceOnCmd7   = XPLMFindCommand(deice7_switch_on.c_str());
** deice8_switch_on = getOptionToString("deice8_switch_on_cmd");
** DeiceOnCmd8   = XPLMFindCommand(deice8_switch_on.c_str());
```

In the off position turns off anti-ice

```
** deice_switch_off = getOptionToString("deice_switch_off_cmd");
** DeiceOffCmd   = XPLMFindCommand(deice_switch_off.c_str());
** deice2_switch_off = getOptionToString("deice2_switch_off_cmd");
** DeiceOffCmd2   = XPLMFindCommand(deice2_switch_off.c_str());
** deice3_switch_off = getOptionToString("deice3_switch_off_cmd");
** DeiceOffCmd3   = XPLMFindCommand(deice3_switch_off.c_str());
** deice4_switch_off = getOptionToString("deice4_switch_off_cmd");
** DeiceOffCmd4   = XPLMFindCommand(deice4_switch_off.c_str());
** deice5_switch_off = getOptionToString("deice5_switch_off_cmd");
** DeiceOffCmd5   = XPLMFindCommand(deice5_switch_off_str());
** deice6_switch_off = getOptionToString("deice6_switch_off_cmd");
** DeiceOffCmd6   = XPLMFindCommand(deice6_switch_off.c_str());
** deice7_switch_off = getOptionToString("deice7_switch_off_cmd");
** DeiceOffCmd7   = XPLMFindCommand(deice7_switch_off.c_str());
** deice8_switch_off = getOptionToString("deice8_switch_off_cmd");
** DeiceOffCmd8   = XPLMFindCommand(deice8_switch_off.c_str());
```

**Pitot heat switch**

In on position turns on pitot tube heat.

```
*  PtHtOn = XPLMFindCommand("sim/ice/pitot_heat_on")
** pitot_heat_switch_on = getOptionToString("pitot_heat_switch_on_cmd");
** PitotHeatOnCmd   = XPLMFindCommand(pitot_heat_switch_on.c_str());
** pitot2_heat_switch_on = getOptionToString("pitot2_heat_switch_on_cmd");
** Pitot2HeatOnCmd   = XPLMFindCommand(pitot2_heat_switch_on.c_str());
```

In off position turns off pitot tube heat.

```
*  PtHtOff  = XPLMFindCommand("sim/ice/pitot_heat_off")
** pitot_heat_switch_off = getOptionToString("pitot_heat_switch_off_cmd");
** PitotHeatOffCmd   = XPLMFindCommand(pitot_heat_switch_off.c_str());
** pitot2_heat_switch_off = getOptionToString("pitot2_heat_switch_off_cmd");
** Pitot2HeatOffCmd   = XPLMFindCommand(pitot2_heat_switch_off.c_str());
```

**Cowl Flaps switch**

For this to work turn off auto cowl flaps in plane maker.
In the open position opens the number of cowl flaps as engines.
*  ClFlOpn  = XPLMFindCommand("sim/flight_controls/cowl_flaps_open")
** cowl_flaps_open = getOptionToString("cowl_flaps_open_cmd");
** CowlFlapsOpenCmd   = XPLMFindCommand(cowl_flaps_open.c_str());
** cowl2_flaps_open = getOptionToString("cowl2_flaps_open_cmd");
** Cowl2FlapsOpenCmd   = XPLMFindCommand(cowl2_flaps_open.c_str());

In close position closes the number of cowl flaps as engines.
*  ClFlCls = XPLMFindCommand("sim/flight_controls/cowl_flaps_closed")
** cowl_flaps_close = getOptionToString("cowl_flaps_close_cmd");
** CowlFlapsCloseCmd   = XPLMFindCommand(cowl_flaps_close.c_str());
** cowl2_flaps_close = getOptionToString("cowl2_flaps_close_cmd");
** Cowl2FlapsCloseCmd   = XPLMFindCommand(cowl2_flaps_close.c_str());

**Panel lights switch**

In the on position turns on the panel lights.
* CockpitLights = XPLMFindDataRef("sim/cockpit/electrical/cockpit_lights")
** panel_lights_switch_on = getOptionToString("panel_lights_switch_on_cmd");
** PanelLightsOnCmd   = XPLMFindCommand(panel_lights_switch_on.c_str());

In the off position turns off the panel lights.
** panel_lights_switch_off = getOptionToString("panel_lights_switch_off_cmd");
** PanelLightsOffCmd   = XPLMFindCommand(panel_lights_switch_off.c_str());

**Beacon switch**

In the on position turns on the beacon lights.
* BcLtOn = XPLMFindCommand("sim/lights/beacon_lights_on")
** beacon_lights_switch_on = getOptionToString("beacon_lights_switch_on_cmd");
** BeaconLightsOnCmd   = XPLMFindCommand(beacon_lights_switch_on.c_str());

In the off position turns off the becon lights.
* BcLtOff  = XPLMFindCommand("sim/lights/beacon_lights_off")
** beacon_lights_switch_off = getOptionToString("beacon_lights_switch_off_cmd");
** BeaconLightsOffCmd   = XPLMFindCommand(beacon_lights_switch_off.c_str());

**Nav switch**

In the on position turns on the navagation lights.
* NvLtOn   = XPLMFindCommand("sim/lights/nav_lights_on")
** nav_lights_switch_on = getOptionToString("nav_lights_switch_on_cmd");
** NavLightsOnCmd   = XPLMFindCommand(nav_lights_switch_on.c_str());

In the off position turns off the navagation lights.
* NvLtOff  = XPLMFindCommand("sim/lights/nav_lights_off")
** nav_lights_switch_off = getOptionToString("nav_lights_switch_off_cmd");
** NavLightsOffCmd   = XPLMFindCommand(nav_lights_switch_off.c_str());

**Strobe switch**

In the on position turns on the strobe lights.
* StLtOn = XPLMFindCommand("sim/lights/strobe_lights_on")
** strobe_lights_switch_on = getOptionToString("strobe_lights_switch_on_cmd");
** StrobeLightsOnCmd   = XPLMFindCommand(strobe_lights_switch_on.c_str());

In off position turns off the strobe lights.
* StLtOff  = XPLMFindCommand("sim/lights/strobe_lights_off")
** strobe_lights_switch_off = getOptionToString("strobe_lights_switch_off_cmd");
** StrobeLightsOffCmd   = XPLMFindCommand(strobe_lights_switch_off.c_str());

**Taxi switch**

In the on position turns on the taxi lights.

* TxLtOn = XPLMFindCommand("sim/lights/taxi_lights_on")
** taxi_lights_switch_on = getOptionToString("taxi_lights_switch_on_cmd");
** TaxiLightsOnCmd   = XPLMFindCommand(taxi_lights_switch_on.c_str());

In the off position turns off the taxi lights.

* TxLtOff = XPLMFindCommand("sim/lights/taxi_lights_off")
** taxi_lights_switch_off = getOptionToString("taxi_lights_switch_off_cmd");
** TaxiLightsOffCmd   = XPLMFindCommand(taxi_lights_switch_off.c_str());

**Landing switch**

In the on position turns on the landing lights.

* LnLtOn = XPLMFindCommand("sim/lights/landing_lights_on")
** landing_lights_switch_on = getOptionToString("landing_lights_switch_on_cmd");
** LandingLightsOnCmd   = XPLMFindCommand(landing_lights_switch_on.c_str());

In the off position turns off the landing lights.

* LnLtOff = XPLMFindCommand("sim/lights/landing_lights_off")
** landing_lights_switch_off = getOptionToString("landing_lights_switch_off_cmd");
** LandingLightsOffCmd   = XPLMFindCommand(landing_lights_switch_off.c_str());

**Gear Indicator Lights**

Off: Undercarriage Retracted
Green: Undercarriage Deployed
Red: Undercarriage In-Transition or Faulted

* GearRetract = XPLMFindDataRef("sim/aircraft/gear/acf_gear_retract")
* LandingGearStatus = XPLMFindDataRef("sim/aircraft/parts/acf_gear_deploy")
* Gear1Fail = XPLMFindDataRef("sim/operation/failures/rel_lagear1")
* Gear2Fail = XPLMFindDataRef("sim/operation/failures/rel_lagear2")
* Gear3Fail = XPLMFindDataRef("sim/operation/failures/rel_lagear3")

**Landing Gear Switch**

Gear Up or Down

* GearRetract = XPLMFindDataRef("sim/aircraft/gear/acf_gear_retract")

**********

Radio Panel (http://www.saitek.com/uk/prod/radio.html)

**********

The switches are mapped in the following manner for there default behavior. The default behavior can be changed using the xsaitekpanels.ini file and for more information look in that section of the manual.

The radio panel uses the Fn key to expanded use of some of the switches.
I will explain in detail where that is true.
I support up to three radio panels.

There is a upper and lower section and are completely independent of each other.
I will show the mapping for the upper.

**COM1 switch position**

Left display is COM1 Active.
* Com1ActFreq = XPLMFindDataRef("sim/cockpit/radios/com1_freq_hz")

Right display is COM1 Stanby
* Com1StbyFreq = XPLMFindDataRef("sim/cockpit/radios/com1_stdby_freq_hz")

Large silver knob controls the whole numbers
* Com1StbyCorseUp = XPLMFindCommand("sim/radios/stby_com1_coarse_up")
* Com1StbyCorseDn = XPLMFindCommand("sim/radios/stby_com1_coarse_down")

Small silver knob controls the fractional numbers
* Com1StbyFineUp = XPLMFindCommand("sim/radios/stby_com1_fine_up")
* Com1StbyFineDn = XPLMFindCommand("sim/radios/stby_com1_fine_down")

ACT/STBY moves the left and right display
* Com1ActStby = XPLMFindCommand("sim/radios/com1_standy_flip")

**COM2 switch position**

Left display is COM2 Active
* Com2ActFreq = XPLMFindDataRef("sim/cockpit/radios/com2_freq_hz")

Right display is COM2 Stanby
* Com2StbyFreq = XPLMFindDataRef("sim/cockpit/radios/com2_stdby_freq_hz")

Large silver knob controls the whole numbers
* Com2StbyCorseUp = XPLMFindCommand("sim/radios/stby_com2_coarse_up")
* Com2StbyCorseDn = XPLMFindCommand("sim/radios/stby_com2_coarse_down")

Small silver know controls the fractional numbers
* Com2StbyFineUp = XPLMFindCommand("sim/radios/stby_com2_fine_up")
* Com2StbyFineDn = XPLMFindCommand("sim/radios/stby_com2_fine_down")

ACT/STBY moves the left and right display
* Com2ActStby = XPLMFindCommand("sim/radios/com2_standy_flip")

**NAV1 switch position**

Left display is NAV1 Active
* Nav1ActFreq = XPLMFindDataRef("sim/cockpit/radios/nav1_freq_hz")

Right display is NAV1 Stanby
* Nav1StbyFreq = XPLMFindDataRef("sim/cockpit/radios/nav1_stdby_freq_hz")
If Fn key pushed display Obs1

Large silver knob controls the whole numbers
* Nav1StbyCorseUp = XPLMFindCommand("sim/radios/stby_nav1_coarse_up")
* Nav1StbyCorseDn = XPLMFindCommand("sim/radios/stby_nav1_coarse_down")
If Fn key pushed.
* Obs1Up = XPLMFindCommand(“sim/radios/obs1_up);
* Obs1Down = XPLMFindCommand(“sim/radios/obs1_down);

Small silver know controls the fractional numbers
* Nav1StbyFineUp = XPLMFindCommand("sim/radios/stby_nav1_fine_up")
* Nav1StbyFineDn = XPLMFindCommand("sim/radios/stby_nav1_fine_down")
If Fn key pushed.
* Obs1Up = XPLMFindCommand(“sim/radios/obs1_up”); X 10
* Obs1Down = XPLMFindCommand(“sim/radios/obs1_down”); X 10

ACT/STBY moves the left and right display
* Nav1ActStby = XPLMFindCommand("sim/radios/nav1_standy_flip")

**NAV2 switch position**

Left display is NAV2 Active
* Nav2ActFreq = XPLMFindDataRef("sim/cockpit/radios/nav2_freq_hz")

Right display is NAV2 Stanby
* Nav2StbyFreq = XPLMFindDataRef("sim/cockpit/radios/nav2_stdby_freq_hz")
If Fn key pushed display Obs2

Large silver knob controls the whole numbers
* Nav2StbyCorseUp = XPLMFindCommand("sim/radios/stby_nav2_coarse_up")
* Nav2StbyCorseDn = XPLMFindCommand("sim/radios/stby_nav2_coarse_down")
If Fn key pushed.
* Obs2Up = XPLMFindCommand(“sim/radios/obs2_up”);
* Obs2Down = XPLMFindCommand(“sim/radios/obs2_down”);

Small silver knob controls the fractional numbers
* Nav2StbyFineUp = XPLMFindCommand("sim/radios/stby_nav2_fine_up")
* Nav2StbyFineDn = XPLMFindCommand("sim/radios/stby_nav2_fine_down")
If Fn key pushed.
* Obs2Up = XPLMFindCommand(“sin/radios/obs2_up); X 10
* Obs2Down = XPLMFindCommand(“sin/radios/obs2_down); X 10

ACT/STBY moves the left and right display
* Nav2ActStby = XPLMFindCommand("sim/radios/nav2_standy_flip")

**Upper ADF switch position**

If one ADF selected from menu upper and lower are as shown.

Left display is ADF Active
* Adf1ActFreq = XPLMFindDataRef("sim/cockpit/radios/adf1_freq_hz")

Right display is ADF Standby
* Adf1StbyFreq = XPLMFindDataRef("sim/cockpit/radios/adf1_stdby_freq_hz")
If Fn key pushed display Adf1 card direction.
* Adf1CardDirDegm = XPLMFindDataRef("sim/cockpit/radios/adf1_cardinal_dir");

If Fn key is not pushed
Large silver knob controls the selected number to increase or decrease.
The position is marked with a decimal point.
Small silver knob controls increase of decrease of the selected digit.

Ones selected
* Afd1StbyOnesUp = XPLMFindCommand("sim/radios/stby_adf1_ones_up")
* Afd1StbyOnesDn = XPLMFindCommand("sim/radios/stby_adf1_ones_down")

Tens selected
* Afd1StbyTensUp = XPLMFindCommand("sim/radios/stby_adf1_tens_up")
* Afd1StbyTensDn = XPLMFindCommand("sim/radios/stby_adf1_tens_down")

Hundereds selected
* Afd1StbyHunUp = XPLMFindCommand("sim/radios/stby_adf1_hundreds_up")
* Afd1StbyHunDn = XPLMFindCommand("sim/radios/stby_adf1_hundreds_down")

ACT/STBY moves the right and left display
* Adf1ActStby = XPLMFindCommand("sim/radios/adf1_standy_flip")

If Fn key is pushed

Large silver knob adds or subtracts 10 to
* Adf1CardDirDegm = XPLMFindDataRef("sim/cockpit/radios/adf1_cardinal_dir");

Small silver knob adds or subtracts 1 to
* Adf1CardDirDegm = XPLMFindDataRef("sim/cockpit/radios/adf1_cardinal_dir");

**Lower ADF switch position**

If two ADF's selected from menu upper is the same and lower are as shown below.

Left display is ADF Active
* Adf1ActFreq = XPLMFindDataRef("sim/cockpit/radios/adf2_freq_hz")

Right display is ADF Standby
* Adf1StbyFreq = XPLMFindDataRef("sim/cockpit/radios/adf2_stdby_freq_hz")

If Fn key pushed and 1 adf display Adf1 card direction.
* Adf1CardDirDegm = XPLMFindDataRef("sim/cockpit/radios/adf1_cardinal_dir");

If Fn key pushed and 2 adfs display Adf2 card direction.
* Adf2CardDirDegm = XPLMFindDataRef("sim/cockpit/radios/adf2_cardinal_dir");

Large silver knob controls the selected number to increase or decrease.
The position is marked with a decimal point.
Small silver knob controls increase of decrease of the selected digit.

Ones selected
* Afd1StbyOnesUp = XPLMFindCommand("sim/radios/stby_adf2_ones_up")
* Afd1StbyOnesDn = XPLMFindCommand("sim/radios/stby_adf2_ones_down")

Tens selected
* Afd1StbyTensUp = XPLMFindCommand("sim/radios/stby_adf2_tens_up")
* Afd1StbyTensDn = XPLMFindCommand("sim/radios/stby_adf2_tens_down")

Hundereds selected
* Afd1StbyHunUp = XPLMFindCommand("sim/radios/stby_adf2_hundreds_up")
* Afd1StbyHunDn = XPLMFindCommand("sim/radios/stby_adf2_hundreds_down")

ACT/STBY moves the right and left display
* Adf1ActStby = XPLMFindCommand("sim/radios/adf2_standy_flip")

If Fn key is pushed and 1 adf

Large silver knob adds or subtracts 10 to
* Adf1CardDirDegm = XPLMFindDataRef("sim/cockpit/radios/adf1_cardinal_dir");

Small silver knob adds or subtracts 1 to
* Adf1CardDirDegm = XPLMFindDataRef("sim/cockpit/radios/adf1_cardinal_dir");

If Fn key is pushed and 2 adfs

Large silver knob adds or subtracts 10 to
* Adf2CardDirDegm = XPLMFindDataRef("sim/cockpit/radios/adf2_cardinal_dir");

Small silver knob adds or subtracts 1 to
* Adf2CardDirDegm = XPLMFindDataRef("sim/cockpit/radios/adf2_cardinal_dir");

**DME switch position**

Left display depends on the mode selected

RMT selected displays speed in knots
* Nav1DmeSpeed = XPLMFindDataRef("sim/cockpit2/radios/indicators/nav1_dme_speed_kts")
* Nav2DmeSpeed = XPLMFindDataRef("sim/cockpit2/radios/indicators/nav2_dme_speed_kts")

FRQ selected displays frequency
* DmeFreq =  XPLMFindDataRef("sim/cockpit2/radios/actuators/dme_frequency_hz")

GS/T selected displays Ground speed in knots
* DmeSpeed = XPLMFindDataRef("sim/cockpit2/radios/indicators/dme_dme_speed_kts

Right display depends in the mode selected

RMT selected displays distance in nautical miles
*Nav1DmeNmDist=XPLMFindDataRef("sim/cockpit2/radios/indicators/nav1_dme_distance_nm")
*Nav2DmeNmDist=XPLMFindDataRef("sim/cockpit2/radios/indicators/nav2_dme_distance_nm")

FRQ selected displays time in minutes
* DmeTime = XPLMFindDataRef("sim/cockpit2/radios/indicators/dme_dme_time_min")

GS/T selected displays time in minutes
* DmeTime = XPLMFindDataRef("sim/cockpit2/radios/indicators/dme_dme_time_min")

Large silver knob depends on mode selected
RMT selected has no operation
FRQ selected adjusts the whole numbers on the display on the left GS/T selected has no operation

Small silver knob depends on mode selected
RMT selected has no operation
FRQ selected adjust the fractional numbers on the display on the left
GS/T selected has no operation

If Fn key is not pressed.
ACT/STBY selects the three modes of the DME
RMT Remote DME channels when you select your NAV frequency on the Nav receiver.
FRQ Frequency DMT displays distance and selected frequency adjusted by silver knobs
GS/T Groundspeed/Time-to-Station

If Fn key is pressed.
ACT/STBY selects N1 or N2

**XPDR switch position**

The Fn key is used here to add functionality to this position.

Left display is the barometer setting for the altimeter.
A menu selection allows you to select inHg or hPa to be used in setting the altimeter.
There is also a xsaitekpanels.ini setting to change the default.
If the Fn key is pushed a decimal point is displayed in the left hand side.
* BaroSetting = XPLMFindDataRef("sim/cockpit/misc/barometer_setting")

Right display is the transponder code.
There is a decimal point at the digit that can be adjusted
* XpdrCode = XPLMFindDataRef("sim/cockpit/radios/transponder_code")

Large silver knob depends on the function button.
If the Fn key is not pushed.
Adjusts the selected digit to be changed for the transponder.

If the Fn key is pushed.
Adjusts the barometer in X 10 adjustments.

Small silver knob depends on the function button
If the Fn key is not pushed.
Adjust the selected digit for the Transponder.

If the Fn key is pushed.
Adjusts the barometer in X 1 adjustments.

ACT/STBY depends on the function button
If the Fn key is not pushed.
Selects the modes of the transponder.

If the Fn key is pushed.
Sets the barometer setting to 29.92

**********

Multi Panel (http://www.saitek.com/uk/prod/multi.html)

**********

The switches are mapped in the following manner for there default behavior. The default behavior can be changed using the xsaitekpanels.ini file and for more information look in that section of the manual.

The multi panel uses the Fn key to expanded use of some of the switches.
I will explain in detail where that is true.

The NAV button has been tweaked to allow users to control VORLOC and LNAV. The behavior of the NAV button is dependent upon the HSI selector. If NAV1 or NAV2 is selected, the NAV button and LED are associated with VORLOC; otherwise, it is linked to LNAV. Lastly, the FN+IAS button controls the IAS-to-Mach changeover.

**ALT switch position**

Upper display is the auto pilot altitude
* ApAlt = XPLMFindDataRef("sim/cockpit/autopilot/altitude")

Lower display is the auto pilot vertical speed
* ApVs = XPLMFindDataRef("sim/cockpit/autopilot/vertical_velocity")

Silver knob adjusts the altitude up and down
If the Fn key is not pushed.
Adjust in 100 foot increments
*  ApAlt = XPLMFindDataRef("sim/cockpit/autopilot/altitude")
** alt_switch_up_remapable = getOptionToString("alt_switch_up_remapable_cmd");
** AltSwitchUpRemapableCmd   = XPLMFindCommand(alt_switch_up_remapable.c_str());
** alt_switch_dn_remapable = getOptionToString("alt_switch_dn_remapable_cmd");
** AltSwitchDnRemapableCmd   = XPLMFindCommand(alt_switch_dn_remapable.c_str());
*** alt_switch_data_remapable = getOptionToString("alt_switch_remapable_data");
*** AltSwitchRemapableData   = XPLMFindDataRef(alt_switch_data_remapable.c_str());

If the Fn key is pushed.
Adjusts in 1000 foot increments

**VS switch position**

Upper display in the auto pilot alititude
* ApAlt = XPLMFindDataRef("sim/cockpit/autopilot/altitude")

Lower display is the auto pilot vertical speed
* ApVs = XPLMFindDataRef("sim/cockpit/autopilot/vertical_velocity")

Silver knob adjusts vertical speed up and down
If the Fn key is not pushed.
Adjusts in 100 foot increments.
*  ApVsUp = XPLMFindCommand("sim/autopilot/vertical_speed_up")
*  ApVsDn = XPLMFindCommand("sim/autopilot/vertical_speed_down")
** vs_switch_up_remapable = getOptionToString("vs_switch_up_remapable_cmd");
** VsSwitchUpRemapableCmd   = XPLMFindCommand(vs_switch_up_remapable.c_str());
** vs_switch_dn_remapable = getOptionToString("vs_switch_dn_remapable_cmd");
** VsSwitchDnRemapableCmd   = XPLMFindCommand(vs_switch_dn_remapable.c_str());
*** vs_switch_data_remapable = getOptionToString("vs_switch_remapable_data");
*** VsSwitchRemapableData   = XPLMFindDataRef(vs_switch_data_remapable.c_str());

If the Fn key is pushed.
Adjusts in 200 foot increments.
*  ApVsUp = XPLMFindCommand("sim/autopilot/vertical_speed_up")
*  ApVsDn = XPLMFindCommand("sim/autopilot/vertical_speed_down")
** vs_switch_up_remapable = getOptionToString("vs_switch_up_remapable_cmd");
** VsSwitchUpRemapableCmd   = XPLMFindCommand(vs_switch_up_remapable.c_str());
** vs_switch_dn_remapable = getOptionToString("vs_switch_dn_remapable_cmd");
** VsSwitchDnRemapableCmd   = XPLMFindCommand(vs_switch_dn_remapable.c_str());
*** vs_switch_data_remapable = getOptionToString("vs_switch_remapable_data");
*** VsSwitchRemapableData   = XPLMFindDataRef(vs_switch_data_remapable.c_str());

**IAS switch position**

Upper display is the auto pilot indicated air speed
* ApAs = XPLMFindDataRef("sim/cockpit/autopilot/airspeed")

Lower display is blank

Silver knob adjusts indicated air speed up and down
If the Fn key is not pushed.
Adjusts by a factor of 1 if knots .01 if mach
*  AirspeedIsMach = XPLMFindDataRef("sim/cockpit/autopilot/airspeed_is_mach")
*  Airspeed = XPLMFindDataRef("sim/cockpit/autopilot/airspeed")
** ias_switch_up_remapable = getOptionToString("ias_switch_up_remapable_cmd");
** IasSwitchUpRemapableCmd   = XPLMFindCommand(ias_switch_up_remapable.c_str());
** ias_switch_dn_remapable = getOptionToString("ias_switch_dn_remapable_cmd");
** IasSwitchDnRemapableCmd = XPLMFindCommand(ias_switch_dn_remapable.c_str());
*** ias_switch_data_remapable = getOptionToString("ias_switch_remapable_data");
*** IasSwitchRemapableData = XPLMFindDataRef(ias_switch_data_remapable.c_str());

If the Fn key is pushed.
Adjusts by a factor of 10 if knots .1 if mach
*  AirspeedIsMach = XPLMFindDataRef("sim/cockpit/autopilot/airspeed_is_mach")
*  Airspeed = XPLMFindDataRef("sim/cockpit/autopilot/airspeed")
** IasSwitchUpRemapableCmd   = XPLMFindCommand(ias_switch_up_remapable.c_str());
** ias_switch_dn_remapable = getOptionToString("ias_switch_dn_remapable_cmd");
** IasSwitchDnRemapableCmd = XPLMFindCommand(ias_switch_dn_remapable.c_str());
*** ias_switch_data_remapable = getOptionToString("ias_switch_remapable_data");
*** IasSwitchRemapableData = XPLMFindDataRef(ias_switch_data_remapable.c_str());

**HDG switch position**

Upper display is the auto pilot heading.
* ApHdg = XPLMFindDataRef("sim/cockpit/autopilot/heading_mag")

Lower display is blank

Silver knob adjusts the heading up and down
If the Fn key is not pushed.
Adjusts by a factor of 1.
*  ApHdg = XPLMFindDataRef("sim/cockpit/autopilot/heading_mag")
** hdg_switch_up_remapable = getOptionToString("hdg_switch_up_remapable_cmd");
** HdgSwitchUpRemapableCmd   = XPLMFindCommand(hdg_switch_up_remapable.c_str());
** hdg_switch_dn_remapable = getOptionToString("hdg_switch_dn_remapable_cmd");
** HdgSwitchDnRemapableCmd = XPLMFindCommand(hdg_switch_dn_remapable.c_str());
*** hdg_switch_data_remapable = getOptionToString("hdg_switch_remapable_data");
*** HdgSwitchRemapableData = XPLMFindDataRef(hdg_switch_data_remapable.c_str());

If the Fn key is pushed.
Adjusts by a factor of 10.
*  ApHdg = XPLMFindDataRef("sim/cockpit/autopilot/heading_mag")
** HdgSwitchUpRemapableCmd   = XPLMFindCommand(hdg_switch_up_remapable.c_str());
** hdg_switch_dn_remapable = getOptionToString("hdg_switch_dn_remapable_cmd");
** HdgSwitchDnRemapableCmd = XPLMFindCommand(hdg_switch_dn_remapable.c_str());
*** hdg_switch_data_remapable = getOptionToString("hdg_switch_remapable_data");
*** HdgSwitchRemapableData = XPLMFindDataRef(hdg_switch_data_remapable.c_str());

**CRS switch position**

Upper display is the auto pilot course
* ApCrs =  XPLMFindDataRef("sim/cockpit2/radios/actuators/hsi_obs_deg_mag_pilot")

Lower display is blank

Silver knob adjusts the course
If the Fn key is not pushed.
Adjusts by a factor of 1.
** crs_switch_up_remapable = getOptionToString("crs_switch_up_remapable_cmd");
** CrsSwitchUpRemapableCmd   = XPLMFindCommand(crs_switch_up_remapable.c_str());
** crs_switch_dn_remapable = getOptionToString("crs_switch_dn_remapable_cmd");
** CrsSwitchDnRemapableCmd = XPLMFindCommand(crs_switch_dn_remapable.c_str());
*** crs_switch_data_remapable = getOptionToString("crs_switch_remapable_data");
*** CrsSwitchRemapableData = XPLMFindDataRef(crs_switch_data_remapable.c_str());

If the Fn key is pushed ajusts by a factor of 10

** ** crs_switch_up_remapable = getOptionToString("crs_switch_up_remapable_cmd");
** CrsSwitchUpRemapableCmd   = XPLMFindCommand(crs_switch_up_remapable.c_str());
** crs_switch_dn_remapable = getOptionToString("crs_switch_dn_remapable_cmd");
** CrsSwitchDnRemapableCmd = XPLMFindCommand(crs_switch_dn_remapable.c_str());
*** crs_switch_data_remapable = getOptionToString("crs_switch_remapable_data");
*** CrsSwitchRemapableData = XPLMFindDataRef(crs_switch_data_remapable.c_str());

**AP Button**

Selects flight director modes.
Off = 0
On = 1
Auto = 2
*  ApMstrStat = XPLMFindDataRef("sim/cockpit2/autopilot/flight_director_mode")
** ap_button_remapable = getOptionToString("ap_button_remapable_cmd");
** ApButtonRemapableCmd   = XPLMFindCommand(ap_button_remapable.c_str());

**AP Indicator Light**

Flight director mode
Off = light is off = 0
On  = light is flashing = 1
Auto = light is on = 2

```
*  ApMstrStat = XPLMFindDataRef("sim/cockpit2/autopilot/flight_director_mode")
** ap_light_remapable = getOptionToString("ap_light_remapable_data");
 *** ApLightRemapableData   = XPLMFindDataRef(ap_light_remapable.c_str());
*** ap_light_flash_remapable = getOptionToString("ap_light_flash_remapable_data");
*** ApLightFlashRemapableData   = XPLMFindDataRef(ap_light_flash_remapable.c_str());
```

**HDG Button**

Toggle On or Off Autopilot heading-hold

```
*  ApHdgBtn = XPLMFindCommand("sim/autopilot/heading");
** hdg_button_remapable = getOptionToString("hdg_button_remapable_cmd");
** HdgButtonRemapableCmd   = XPLMFindCommand(hdg_button_remapable.c_str());
```

If the Fn key is pushed Heading is synced to magnetic heading

**HDG Indicator Light**

Heading Status
Off: Auto Pilot off or Autopilot Heading Status > off = 0
Flashing: Autopilot Heading Status > armed = 1
On: Autopilot Heading Status > captured = 2

```
*  ApHdgStat = XPLMFindDataRef("sim/cockpit2/autopilot/heading_status")
*** hdg_light_remapable = getOptionToString("hdg_light_remapable_data");
*** HdgLightRemapableData   = XPLMFindDataRef(hdg_light_remapable.c_str());
*** hdg_light_flash_remapable = getOptionToString("hdg_light_flash_remapable_data");
*** HdgLightFlashRemapableData   = XPLMFindDataRef(hdg_light_flash_remapable.c_str());
```

**NAV Button**

Toggle On or Off Autopilot VOR/LOC arm

```
*  ApNavBtn = XPLMFindCommand("sim/autopilot/NAV")
** nav_button_vorloc_remapable = getOptionToString("nav_button_vorloc_remapable_cmd");
** NavButtonVorlocRemapableCmd   = XPLMFindCommand(nav_button_vorloc_remapable.c_str());
** nav_button_lnav_remapable = getOptionToString("nav_button_lnav_remapable_cmd");
** NavButtonLnavRemapableCmd   = XPLMFindCommand(nav_button_lnav_remapable.c_str());
```

If the Fn key is pushed NAV Button toggles OSB1 and OBS2

**NAV Indicator Light**

Nav mode
Off: Autopilot off or Autopilot Nav Status > off = 0
Flashing: Autopilot Nav Status > armed = 1
On: Autopilot Nav Status > captured = 2

```
*  ApNavBtn = XPLMFindCommand("sim/autopilot/NAV")
 *** nav_light_vorloc_remapable = getOptionToString("nav_light_vorloc__remapable_data");
*** NavLightVorlocRemapableData   = XPLMFindDataRef(nav_light_vorloc_remapable.c_str());
*** nav_light_vorloc_flash_remapable = getOptionToString("nav_light_vorloc_flash_remapable_data");
*** NavLightVorlocFlashRemapableData = XPLMFindDataRef(nav_light_vorloc_flash_remapable.c_str());
*** nav_light_lnav_remapable = getOptionToString("nav_light_lnav_remapable_data");
*** NavLightLnavRemapableData   = XPLMFindDataRef( nav_light_lnav_remapable.c_str());
*** nav_light_lnav_flash_remapable = getOptionToString("nav_light_lnav_flash_remapable_data");
*** NavLightLnavFlashRemapableData   = XPLMFindDataRef( nav_light_lnav_flash_remapable.c_str());
```

**IAS Button**

Toggle On or Off Auto pilot IAS guidance mode

```
*  ApIasBtn = XPLMFindCommand("sim/autopilot/level_change")
*** ias_button_remapable = getOptionToString("ias_button_remapable_cmd");
*** IasButtonRemapableCmd   = XPLMFindCommand(ias_button_remapable.c_str());
*** ias_changeover_button_remapable = getOptionToString("ias_changeover_button_remapable_cmd");
*** IasChangeoverButtonRemapableCmd = XPLMFindCommand(ias_changeover_button_remapable.c_str());
```

If the Fn key is pushed toggle between knots and mach

```
*  ApKnotsMachTgl = XPLMFindCommand("sim/autopilot/knots_mach_toggle");
```

**IAS Indicator Light**

Off: Autopilot Speed-hold (via pitch) status > off = 0
Flashing: Autopilot Speed-hold (via pitch) status > armed = 1
On: Autopilot Speed-hold (via pitch) status > captured = 2

```
*  ApIasStat = XPLMFindDataRef("sim/cockpit2/autopilot/speed_status")
*** ias_light_remapable = getOptionToString("ias_light_remapable_data");
*** IasLightRemapableData   = XPLMFindDataRef(ias_light_remapable.c_str());
*** ias_light_flash_remapable = getOptionToString("ias_light_flash_remapable_data");
*** IasLightFlashRemapableData   = XPLMFindDataRef(ias_light_flash_remapable.c_str());
```

**ALT Button**

Toggle On or Off Autopilot altitude select or hold

```
*  ApAltArmBtn = XPLMFindCommand("sim/autopilot/altitude_arm");
** alt_button_remapable = getOptionToString("alt_button_remapable_cmd");
** AltButtonRemapableCmd   = XPLMFindCommand(alt_button_remapable.c_str());
```

**ALT Indicator Light**

Off: Autopilot Altitude hold status > off = 0
Flashing: Autopilot Altitude hold status > armed = 1
On: Autopilot Altitude hold status > captured = 2

```
*  ApAltStat = XPLMFindDataRef("sim/cockpit2/autopilot/altitude_hold_status")
*** alt_light_remapable = getOptionToString("alt_light_remapable_data");
*** AltLightRemapableData   = XPLMFindDataRef(alt_light_remapable.c_str());
*** alt_light_flash_remapable = getOptionToString("alt_light_flash_remapable_data");
*** AltLightFlashRemapableData   = XPLMFindDataRef(alt_light_flash_remapable.c_str());
```

**VS Button**

Toggle On or Off Autopilot vertical speed

```
*  ApVsBtn = XPLMFindCommand("sim/autopilot/vertical_speed")
** vs_button_remapable = getOptionToString("vs_button_remapable_cmd");
** VsButtonRemapableCmd   = XPLMFindCommand(vs_button_remapable.c_str());
```

**VS Indicator Light**

Off: Autopilot VVI Status > off = 0
Flashing: Autopilot VVI Status > armed = 1
On: Autopilot VVI Status > captured = 2

```
*  ApVsStat = XPLMFindDataRef("sim/cockpit2/autopilot/vvi_status")
*** vs_light_remapable = getOptionToString("vs_light_remapable_data");
*** VsLightRemapableData   = XPLMFindDataRef(vs_light_remapable.c_str());
*** vs_light_flash_remapable = getOptionToString("vs_light_flash_remapable_data");
*** VsLightFlashRemapableData   = XPLMFindDataRef(vs_light_flash_remapable.c_str());
```

**APR Button**

Toggle On or Off Autopilot approach

```
*  ApAprBtn = XPLMFindCommand("sim/autopilot/approach")
** apr_button_remapable = getOptionToString("apr_button_remapable_cmd");
** AprButtonRemapableCmd   = XPLMFindCommand(apr_button_remapable.c_str());
```

If the Fn key is pushed CRS is synced to magnetic heading

**APR Indicator Light**

Off: Autopilot approach status > off = 0
Flashing: Autopilot approach status > armed = 1
On: Autopilot approach status > captured = 2

```
*  ApAprStat = XPLMFindDataRef("sim/cockpit2/autopilot/approach_status")
*** apr_light_remapable = getOptionToString("apr_light_remapable_data");
*** AprLightRemapableData   = XPLMFindDataRef(apr_light_remapable.c_str());
*** apr_light_flash_remapable = getOptionToString("apr_light_flash_remapable_data");
*** AprLightFlashRemapableData   = XPLMFindDataRef(apr_light_flash_remapable.c_str());
```

**REV Button**

Toggle On or Off Autopilot back-course

```
* ApRevBtn = XPLMFindCommand("sim/autopilot/back_course")
** rev_button_remapable = getOptionToString("rev_button_remapable_cmd");
** RevButtonRemapableCmd   = XPLMFindCommand(rev_button_remapable.c_str());
```

**REV Indicator Light**

Off: Autopilot Back-course Status > off = 0
Flashing: Autopilot Back-course Status > armed = 1
On: Autopilot Back-course Status > captured = 2

```
* ApRevStat = XPLMFindDataRef("sim/cockpit2/autopilot/backcourse_status")
*** rev_light_remapable = getOptionToString("rev_light_remapable_data");
*** RevLightRemapableData   = XPLMFindDataRef(rev_light_remapable.c_str());
*** rev_light_flash_remapable = getOptionToString("rev_light_flash_remapable_data");
*** RevLightFlashRemapableData   = XPLMFindDataRef(rev_light_flash_remapable.c_str());
```

**Auto Throttle Switch**

Off: Auto-throttle off = 0
Arm: Auto-throttle on = 1

```
*  ApAutThr = XPLMFindDataRef("sim/cockpit2/autopilot/autothrottle_enabled")
boolean     Auto-throttle on, 0 or 1.  This is the switch.
** attr_switch_remapable = getOptionToString("auto_throttle_switch_remapable_data");
** AttrSwitchRemapableData   = XPLMFindDataRef(attr_switch_remapable.c_str());
```

**Flaps**

Up: Retracts flaps by one notch from current position

```
* FlapsUp = XPLMFindCommand("sim/flight_controls/flaps_up")
** flaps_up_remapable = getOptionToString("flaps_up_remapable_cmd");
** FlapsUpRemapableCmd = XPLMFindCommand(flaps_up_remapable.c_str());
```

Down: Extends flaps by one notch from current position

```
* FlapsDn = XPLMFindCommand("sim/flight_controls/flaps_down")
** flaps_dn_remapable = getOptionToString("flaps_dn_remapable_cmd");
** FlapsDnRemapableCmd = XPLMFindCommand(flaps_dn_remapable.c_str());
```

Fn Key: No Special Effect

**Pitch Trim Wheel**

DN: Moves the trim wheel in the down direction

```
* PitchTrimDn = XPLMFindCommand("sim/flight_controls/pitch_trim_down")
** trim_dn_remapable = getOptionToString("trim_dn_remapable_cmd");
** TrimDnRemapableCmd   = XPLMFindCommand(trim_dn_remapable.c_str());
```

UP: Moves the trim wheel in the up direction

```
* PitchTrimUp = XPLMFindCommand("sim/flight_controls/pitch_trim_up")
** trim_up_remapable = getOptionToString("trim_up_remapable_cmd");
** TrimUpRemapableCmd   = XPLMFindCommand(trim_up_remapable.c_str());
```

**********

Backlight Information Panel (http://www.saitek.com/uk/prod/bip.html)

**********

It uses a Config file that resides in /Resources/plugins/Xsaitekpanels/D2B_config.txt is for the first BIP. It may also reside in the aircraft folder and I will look in the current aircrafts folder first and if not found revert back to the Xsaitekpanels plugin folder.

If you have a second BIP it uses the Config file that resides in /Resources/plugins/Xsaitekpanels/D2B_config2.txt. It may also reside in the aircraft folder and I will look in the current aircrafts folder first and if not found revert back to the Xsaitekpanels plugin folder. I have started using the serial numbers of the BIP's so that when 2 are in use it matches a serial number to  D2B_config.txt and the other serial number to  D2B_config2.txt. This has proven to be a much better way to keep track of two BIP's.

If you have a third BIP it uses the Config file that resides in /Resources/plugins/Xsaitekpanels/D2B_config3.txt. It may also reside in the aircraft folder and I will look in the current aircrafts folder first and if not found revert back to the Xsaitekpanels plugin folder. I have started using the serial numbers of the BIP's so that when 3 are in use it matches a serial number to  D2B_config.txt and the other serial number to  D2B_config2.txt and the other serial number to  D2B_config3.txt. This has proven to be a much better way to keep track of three BIP's.

**********

**To make it very easy to edit the files please look at D2B-Tool for Xsaitekpanels BIP 1.0.rc2 located at http://forums.x-plane.org/index.php?app=downloads&showfile=16139 by CouchPilot. It creates the text files in a GUI way but also allows you to print your own template so you can have as many different tiles as you would like. Thanks for this great new tool.**

---

Look at as a example of what is possible. The D2B_config.txt has two parts a default section and a test section.

The default section mimics the annunciator panel in a Cessna 172.

The test section test all the indicators when you select test and click on the annunciator test button.

Look at your /Resources/plugins/DataRefs.txt to see what data reference's are available to you.

The config file structure is as follows

Every line not starting with # is ignored. So please fill it with comments.

You can use the following commands in the config file:

```
*****************************************************************************
#SET BIP A 0 G FROM  DATAREF sim/cockpit/warnings/annunciator_test_pressed RANGE 1 TO 1
#SET BIP A 1 R FROM  DATAREF sim/cockpit/warnings/annunciators/low_voltage RANGE 1 TO 1
#SET BIP A 2 R FROM  DATAREF sim/cockpit2/annunciators/fuel_quantity RANGE 1 TO 1
#SET BIP A 3 R FROM  DATAREF sim/cockpit2/controls/parking_brake_ratio RANGE 1 TO 1
#SET BIP B 2 R FROM  DATAREF sim/cockpit2/annunciators/low_vacuum RANGE 1 TO 1
#SET BIP B 3 R FROM  DATAREF sim/cockpit2/annunciators/autopilot_disconnect RANGE 1 TO 1

#SET BIP <a> <b> <c> FROM  DATAREF <d> RANGE <e> TO <f>

a) Which row with A being the top row A, B, C.
b) Position on the Row starting with 0. (0  -  7).
c) Which color do you want to display. G = Green, R = Red, A = Amber.
d) Data Reference to tell the indicator to turn on.
e) First number to check.
f) Second number to check.

*****************************************************************************
#SET BIP B 0 R FROM  ARRAY sim/cockpit2/annunciators/oil_pressure_low 0 RANGE 1 TO 1
#SET BIP A 0 R FROM  ARRAY sim/cockpit2/annunciators/generator_off 0 RANGE 1 TO 1
#SET BIP B 1 R FROM  ARRAY sim/cockpit2/annunciators/oil_temperature_high 0 RANGE 1 TO 1

#SET BIP <a> <b> <c> FROM ARRAY <d> <e> RANGE <f> <g>
a) Which row with A being the top row A, B, C.
b) Position on the Row starting with 0. (0  -  7).
c) Which color do you want to display. G = Green, R = Red, A = Amber.
d) Data Reference to tell the indicator to turn on.
e) Array position. For multi engine 0 = first engine.
f) First number to check.
g) Second number to check.
```