

Beginners guide for Opencockpits devices and cards, some SIOC scripting - and interfacing with ProSim

By Espen Linnes
March 2024

Index

Disclaimer	4
Introduction	5
Plug&Play devices from OC	6
The “IOCMODULES-method”	7
The “SIOC Direct-method”	8
The “Combination-method”	9
ProSim and MCP	10
Configure “8.33 Voice Channel Spacing” in ProSim Instructor Station	11
IDC	12
Cards from OC	13
SIOC	14
SIOC.ini	16
Configuration File	16
Configure devices in SIOC.ini	18
IOCMODULES	21
IOCMODULES.ini	22
Configure IOCP Server in ProSim	23
IP-address	24
Port	25
Changing from “IOCMODULES” to “SIOC Direct”	27
Configure SIOC Direct in ProSim	28
SIOC Direct USB Panels	29
SIOC Direct Encoders	30
SIOC Direct EFIS	31
SIOC Direct Other	32
Adding OC cards in SIOC.ini	33
Connecting hardware to cards	36
IOCARD_SW	36
IOCARD_OUT	38
IOCARD_DISPLAY	39
IOCARD_ENCODER	40
USB_SERVOS	41
USB_SERVOS positions	42
SIOC scripting	43
Separator	46

Variable numbers	46
Variable names	46
Local variables	46
SIOC is event driven.....	48
Error handling.....	52
Comments	53
Auto-return for Engine Start switch	54
DELAY command.....	57
IF ELSE command.....	58
MOD command	59
DIV command	60
CALL and SUBROUTINE	62
LIMIT command.....	63
The reserved word “STATIC”	64
Variables – classification.....	65
Creating script file – alternative 1	66
Creating script file – alternative 2	68
Creating script file – alternative 3	70
SIOC documentation.....	71
ProSim and SIOC.....	72
SIOC Monitor	73
TCAS switch and IDENT button.....	74
“IOCMODULES-method” checklist.....	76
“SIOC Direct-method” checklist.....	77
“Combination-method” checklist.....	78
My Engine Start switch solution.....	79
Landing altitude without leading zeros	80
SIOC script for landing altitude without leading zeros	81
Fire Engines module USB Plug&play	82
How to control the backlight	83
How to avoid light pollution	86

Disclaimer

This guide:

- is not from Opencockpits or from ProSim-TS
- is from one amateur simulator builder to another, on how products from Opencockpits *can* interact with ProSim
- contains suggestions how to configure devices, and it is up to each individual to choose the method most suited. One method can work for some, but not for others.

The author of this guide:

- takes no responsibility for whether the solution works or not
- is not an expert in the subjects, nor try to pretend to be
- apologize for bad English and typos (blame on Google Translate)

Introduction

I have been a flight simulator (FS) enthusiast for a long time. My first FS experience was Microsoft Flight Simulator 2000, installed back in 1999. I was immediately hooked.

I had a simple setup: A computer, one monitor, keyboard, mouse and a joystick. As the years went by, I dreamed of something more, but first in 2017 I seriously started to explore the possibilities to build a full scala simulator (SIM). After a lot of time on the web, I learned enough to convince myself: "I can do this".

I chose Opencockpits (OC) because (this is my personal opinion/experience):

1. a wide range of components
2. okay web-shop
3. good price/quality
4. quick expedition
5. very good support

I chose ProSim because they are widely used by other SIM builders, and my experiences are:

1. stable and reliable software
2. easy to use
3. on-going development and bug-fixing
4. decent annual maintenance cost
5. the support is good
6. an active user forum

As a beginner, the need of support is huge, but you cannot ask Support about *everything*. It's a lot of documentation on the web, but during the building period I often thought about writing/sharing my experiences, so others can spend less time finding relevant information.

Note! What made me start writing this guide was a "frequency problem". I do not disregard that people who are wiser than me had solved the problem in an easier way. Anyway, there is a lot to learn in this guide, for beginners.

My simulator works very well (stable and flawless). It must be a sign that I did something right.

Plug&Play devices from OC

As a beginner, Plug&Play (P&P) sound like music to me. Why do it more complicated than necessary?

I have the following OC P&P devices in the SIM:

- MCP
- EFIS (two units)
- COM / VHF (two units)
- NAV (two units)
- ADF (only one unit)
- ATC
- CDU / FMC (only one unit)

The benefits with P&P:

- Easy to configure
- Requires minimal (or none) knowledge about SIOC scripts
- We can handle them as “not P&P”

The only downside (as far as I know) is:

- Completely “Cold & Dark” is not possible¹, the backlight is always on

¹ This is a half-truth, please see [How to control the backlight](#)

The “IOCMModules-method”

When I started to build the SIM, I used SIOC scripts and the IOCMModules-software to connect the devices to ProSim. We need these components:

- SIOC / IOCP server
- SIOC scripts for MCP, EFIS and CDU/FMC
- IOCMModules for COM/VHF, NAV, ADF and ATC

Let’s call this the “IOCMModules-method”.

I was very happy with this solution for a long time, but when I joined VATsim, the radio settings for communication with ATC was not sufficient. The “IOCMModules-method” only give you the possibility to use a COM/VHF frequency ending with NNN.N00, NNN.N25, NNN.N50 or NNN.N75.

The frequencies in the table below works very well with the “IOCMModules-method” because every frequency ends with 00, 50 or 75:

Archive Access: [ENGM App/Director/Final](#)

Facility	Frequency
Oslo Approach	118.475
Oslo Approach	120.450
Oslo Approach (Final)	128.900
Oslo Director	119.975
Oslo Director	136.400

Examples like these are not possible to tune: NNN.N70, NNN.N45 or NNN.N05.

To fix the “frequency problem”, I excluded both COM/VHS devices from IOCMModules and handled them as “not P&P devices”. This required SIOC scripting. There are a lot of SIOC scripts on the web showing how to program the COM/VHS devices. I managed to solve the decimal challenge (selecting frequencies in step by .NN5, not only by .N25), but when I turned the inner knob clockwise, the decimal value decreased. I never managed to solve this annoying issue, and I won't spend time explaining what I did because it didn't work properly anyway.

If you are not on VATsim (or similar), I recommend the “IOCMModules-method”. All the devices work very well with ProSim, except for ATC/Transponder TCAS switch and IDENT button. We have to do some SIOC scripting to fix it (later in this guide).

The “SIOC Direct-method”

To fix the “frequency problem” once and for all, I changed the configuration to this:

- SIOC / IOCP server
- SIOC Direct for all devices

No use of SIOC scripts!

Let’s call this the “SIOC Direct-method”.

I don’t know when ProSim released “SIOC Direct”, if it’s still in development, or fully implemented, but it appears to be both stable and reliable.

On ProSim-TS Forum, I read something like this (this is not a quote): “If you are happy with your current solution, there is no reason to choose SIOC Direct”.

In other words: If your SIM works fine, don’t convert to SIOC Direct.

Converting to SIOC Direct became an emotional roller coaster (mildly exaggerated). As mentioned before, I was very pleased with the SIM based on the “IOCMODULES-method”, with the exception of some frequencies I could not use.

I used SIOC Direct for all devices for a very short period of time. Some experiences:

- I was never completely satisfied with the rotary switches (most important factor)
- Setting the transponder code (Squawk) is easier (my opinion) with the “IOCMODULES-method”
- No scripting for the ATC/Transponder device is necessary. As mentioned above; No scripts at all!
- The “frequency problem” was gone

The “Combination-method”

Neither “IOCMODULES-method” nor “SIOC Direct-method” gave me the optimal solution.

I ended up with a combination of both; “I took the best from each”. But again, each builder must choose the configuration that suits him (or her).

I’m using this configuration in the SIM:

- SIOC / IOCP server
- SIOC scripts for MCP, EFIS and CDU/FMC
- SIOC Direct for COM/VHF, NAV, ADF and ATC

“IOCMODULES” is not in use.

The frequency problem is gone.

I’m very happy with this configuration.

ProSim and MCP

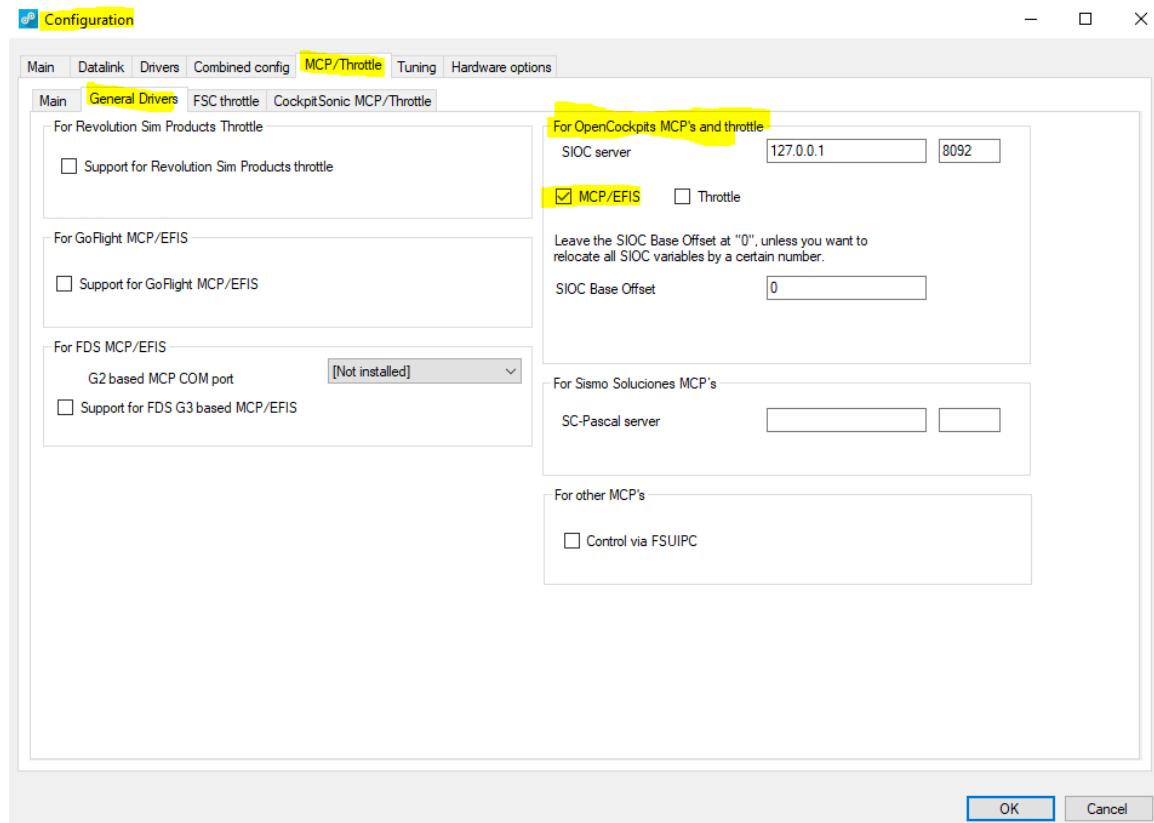
On the ProSim-TS Forum, you will find [ProSimB738 version 3.00 \(Update instructions\)](#).

Note! You need an account to open the site.

At the end of the information, you will see:

ProSimMCP has been removed, so all the MCP related configuration has to be done in ProSimB738-System -> Config -> Configuration. Please enable the appropriate drivers and options for your MCP and MCP based hardware modules.

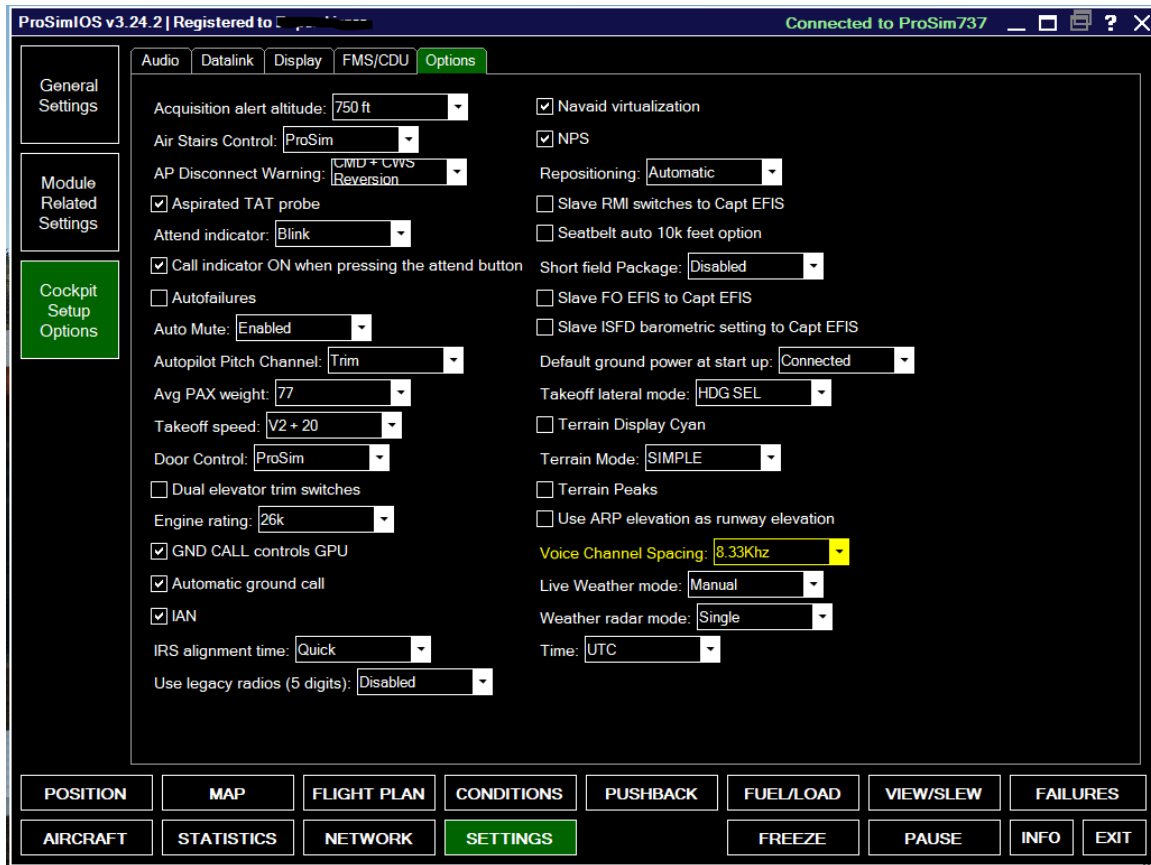
I did this:



The use of IP-address 127.0.0.1 and Port 8092 is explained later in this guide.

Configure “8.33 Voice Channel Spacing” in ProSim Instructor Station

If you want to use COM/VHF frequencies ending with for example: NNN.N70, NNN.N45 or NNN.N05, you need to enable “Voice Channel Spacing = 8.33Khz”. You do this in ProSim Instructor Station (ProSimIOS). Select the [SETTINGS] button, [Cockpit Setup Options] button, and the Options tab:



There is no [Save] button. ProSimIOS remembers your choices.

IDC

As mentioned earlier, I bought P&P because I wanted a “low entry level”, but if you want a completely dark cockpit, you cannot use P&P. OC offer a wide range of IDC devices (which is not P&P). Each IDC device require a SIOC script, and you need card(s) to connect IDC devices to your computer.

You can download scripts from:

https://www.opencockpits.com/catalog/info/information.php?info_id=44&language=en

You must be prepared to dive into scripting.

This guide does not include use of IDC devices, but we will look into SIOC scripting.

By-the-way: I believe IDC is short for “Insulation Displacement Contact or Connector”.

Cards from OC

In addition to devices, OC offer a range of **cards** to connect hardware (input and output) to the SIM:

- **Input** such as switches and encoders
- **Output** such as annunciators

This guide only deals with cards I have in my SIM. OC offers other cards in addition to the ones I use.

Before devices and cards from OC can work with ProSim, you need to install SIOC and maintain the SIOC.ini file. See next page.

SIOC

You need SIOC if you want to use devices and cards from OC. You can download the SIOC software from:

https://www.opencockpits.com/catalog/info/information.php?info_id=45&language=en

If the link above doesn't work, try:

1. <https://es.opencockpits.com/en/>
2. Click on the "Store" button
3. Click in the "SUPPORT" link
4. Click on the "Software" button

You shall see a page something like this:



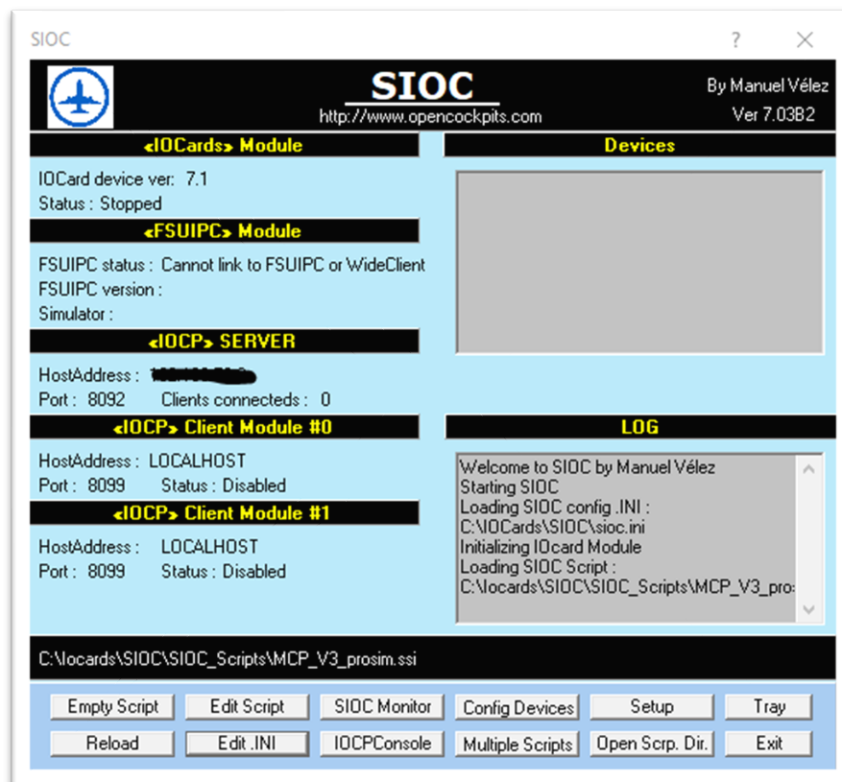
Download and install the latest version of SIOC. I have two computers in my SIM, and I have installed SIOC on both.

This guide does not cover the installation of SIOC, but here are some steps I in general follow when installing software:

- download (and unzip if necessary) the software package
- run the installation as Administrator
- choose an installation path outside "Program Files" (use something like this: "C:\Something", where "Something" relates to the software you are installing)
- you should consider to tell the anti-virus software (if in use) to leave this path and program alone

By-the-way: SIOC is freeware 😊

This is an example of SIOC running (version 7):



Comments for buttons I use:

[Empty Script]

“Erase” the script from the current SIOC session. There is no script running after you have used this button. It’s recommended to use this button before you load a new version of your script.

[Reload]

Starts SIOC again (without use of **[Exit]**)

[Edit .INI]

Edit SIOC.ini (see next page for details)

[SIOC Monitor] and **[IOCPConsole]**

Explained later in this guide

[Open Scrp. Dir.]

Opens “Windows File Explorer” with the folder containing your SIOC scripts

[Tray]

Hide/Minimize SIOC (SIOC is still running)

[Exit]

Stop and exit SIOC

SIOC.ini

Look at the “LOG”-view in the screenshot above (previous page). It tells you where the SIOC.ini file is located (in my case C:\IOCards\SIOC\).

You can edit SIOC.ini with Notepad (locate and open the file from Windows File Explorer), or you can click the **[Edit .INI]** button if SIOC is running.

SIOC.ini is self-documented; The documentation is in the file as “comments”, both in Spanish and English. A comment begins with “[” and ends with “]” like this:

```
[ This is a comment ]
```

You can add and remove comments as you like.

Configuration File

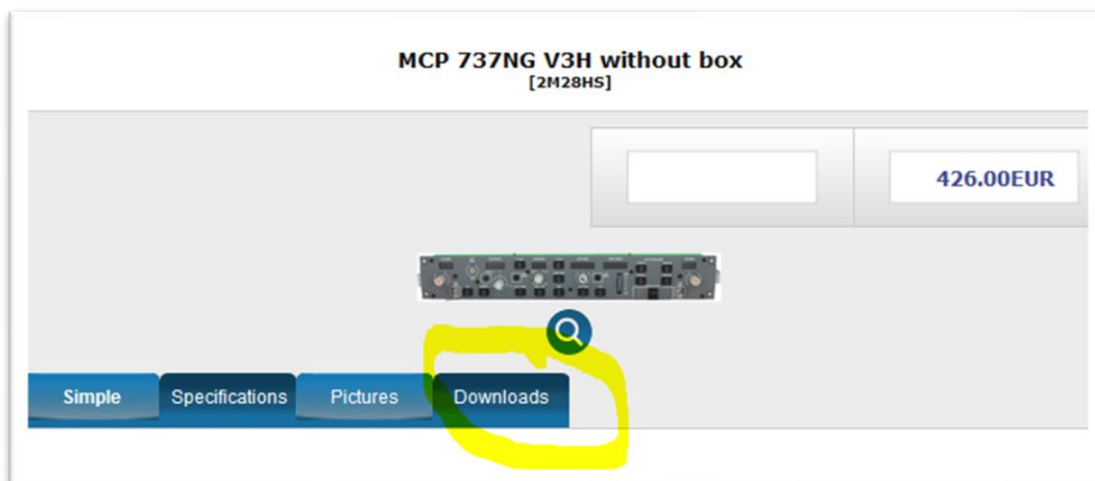
The “Configuration File” is a very important element in SIOC.ini! In fact, the “Configuration File” is the same as a “SIOC script”-file (explained later in this guide). **Note!** We don’t specify “Configuration File” if we only use the “SIOC Direct”-method.

When using “IOCMODULES-method”, “Combination-method”, IDC or need SIOC script to define one or more devices (or cards), we must specify the “Configuration File”. Below, I’m using “MCP_V3_prosim.txt” as Configuration File, which is a SIOC script for MCP, EFIS and CDU/FMC.

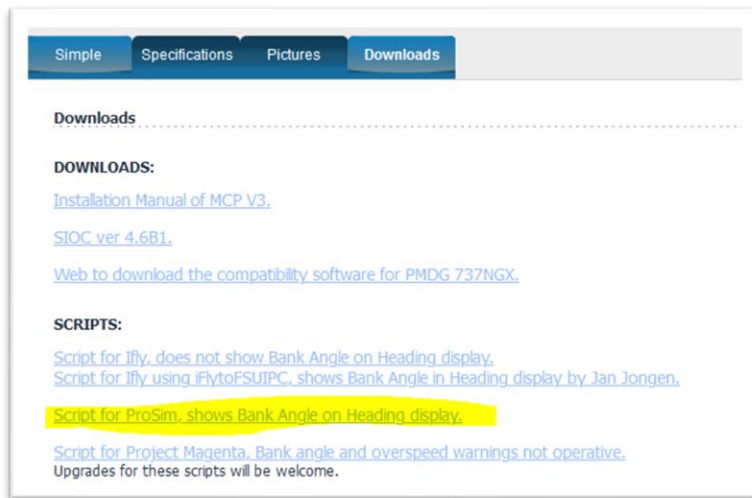
```
[ Fichero de configuracion ]  
[ Configuration File ]  
CONFIG_FILE=C:\Iocards\SIOC\SIOC_Scripts\MCP_V3_prosim.txt
```

You can download [MCP_V3_prosim.txt](#) from OC web-shop.

If the link does not work, you find the file and other useful material in the “Downloads” tab for the product. Locate the product on the web-shop, and click the *Downloads* tab (example):



This page contains the SIOC script we are looking for.



Save the file (in Notepad) as C:\Iocards\SIOC\SIOC_Scripts\MCP_V3_prosim.txt (or in a folder of your choice, and you can use another filename).

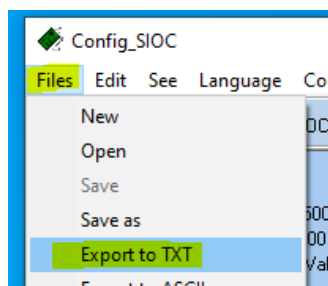
SIOC will automatically compile the text (txt) file and generate a “ssi” version of the txt file. Example: C:\Iocards\SIOC\SIOC_Scripts\MCP_V3_prosim.ssi

After C:\Iocards\SIOC\SIOC_Scripts\MCP_V3_prosim.ssi is generated, you can (but don't have to) change the Configuration File in SIOC.ini to this:

```
[ Fichero de configuracion ]  
[ Configuration File ]  
CONFIG_FILE=C:\Iocards\SIOC\SIOC_Scripts\MCP_V3_prosim.ssi
```

I find it easier to edit a script file in Notepad (as .txt) rather than in SIOC (as .ssi). In general, I recommend you to use the txt version, but if you shall not edit the configuration file, you can specify the ssi version.

Note! If you use .ssi and need to edit the file, you must use the **[Edit Script]** button in SIOC. The .txt version is not updated. If you have made changes to a .ssi file, and want to update the .txt version, you need to Export the .ssi file to TXT:



You don't have to use ssi due to performance reasons. The script is compiled once, during SIOC startup (and reload), and is quickly done.

Configure devices in SIOC.ini

When you connect your OC USB-devices (or USB-cards) to the computer, they will pop up in the “Devices”-view in SIOC (the view is empty if you have nothing from OC connected). **Note!** Not all USB-devices/cards from OC are displayed in the “Devices”-view, such as Yokes and USB Axes cards.

In SIOC.ini, you have to configure (or “define”) the devices you have connected to the computer.

This is a list of devices/cards you can connect (the list is from SIOC version 7), and will pop up in the “Devices”-view if connected:

```
[ type = 0 : Master Card Emulator // OBSOLETE ]
[ type = 1 : Master Card connected directly to parallel port // OBSOLETE ]
[ type = 2 : Master Card connected through compatibility cable to parallel port //OBSOLETE]
[ type = 3 : Expansion Card connected through parallel port //OBSOLETE ]
[ type = 4 : USBExpansion Card used ]
[ type = 5 : Opencockpits MCP module ]
[ type = 6 : USBOutputs Card used ]
[ type = 7 : EFIS module ]
[ type = 8 : Radio COM module ]
[ type = 9 : Radio NAV module ]
[ type = 10 : Radio ADF module ]
[ type = 11 : Radio ATC module ]
[ type = 12 : Radio RMP Airbus module ]
[ type = 13 : FMC-737 module ]
[ type = 14 : USBDCmotorPLUS Card used ]
[ type = 15 : MCP V3 module ]
[ type = 16 : CHRONO B737 module ]
[ type = 17 : USBDimcontrol card used ]
[ type = 18 : Audio B737 module ]
[ type = 19 : FIRE ENGINES B737 module ]
[ type = 20 : PEDESTAL B737 module ]
```

This is important syntax:

MASTER=(Device index) , (Type) , (Number of cards) , (Device number)

Explanation:

- Every line starts with “MASTER=”
- “Device index” (IDX) is a number starting with 0 and you add “one” (0,1,2,3,...) for each new device. This number has to be unique for each device. Try not to change those numbers after they have been assigned. The IDX is the number used in SIOC scripts. If you change the IDX, it will probably cause problems later (if you don’t know what you are doing ...)
- “Type” refers to the list above. E.g. type 15 is the “MCP V3 module”
- “Number of cards” is usually 1, but can be 1, 2, 3 or 4 for “**USBExpansion Card**” (see type 4 on the list above)
- “Device number” is a number given by Windows (USB device number). SIOC will display the “Device number” in the “Devices”-view. Until a correct “Device number” is added to the “MASTER”-line, a star (“*”) is displayed for the device like this: IDX = * -... See more about this on the next page.

The screenshot below illustrates the use of a "*" when a device has an incorrect "Device number" assigned in SIOC.ini.



Note!

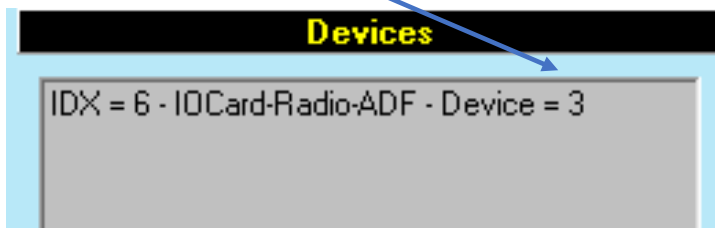
Although the "*" indicates something is wrong with "IDX = ", the problem is related to "Device = ", but the reason for this is because the view has to show us the correct device number to be used.

This is in SIOC.ini:

```
[ Radio ADF module ]  
MASTER=6,10,1,5
```

The "error" is the last element: 5. We have to replace 5 with 3, like this:

```
[ Radio ADF module ]  
MASTER=6,10,1,3
```



The screenshot above shows no "stars" after SIOC is restarted.

Note! If Windows assign a new/different "USB device number" to one or more devices (because of changes in the computer environment), you will see a "*" for each device affected. Do not panic! Just edit SIOC.ini and replace old device numbers with new numbers, as you see in the "Devices"-view in SIOC (in the same manner as described above). You don't have to change anything else in the system. This is the beauty of this solution. Device index/IDX is used in scripts, and they stay unchanged regardless of USB numbers/Device number. **There is one exception:** See [SIOC Direct Other](#).

Elements from one of mine SIOC.ini files are shown on the next page.

My devices are configured like this in SIOC.ini:

```
[ MCP V3 is type 15 ]
MASTER=0,15,1,23

[ EFIS is type 7, CAP ]
MASTER=1,7,1,21

[ EFIS is type 7, FO ]
MASTER=2,7,1,52

[ NAV 1 is type 9, CAP ]
MASTER=3,9,1,41

[ COM 1 is type 8, CAP ]
MASTER=4,8,1,56

[ NAV 2 is type 9, FO ]
MASTER=5,9,1,45

[ ADF is type 10 ]
MASTER=6,10,1,35

[ ATC is type 11 ]
MASTER=7,11,1,43

[ FMC / CDU is type 13 ]
MASTER=8,13,1,54

[ COM 2 is type 8, FO ]
MASTER=9,8,1,56
```

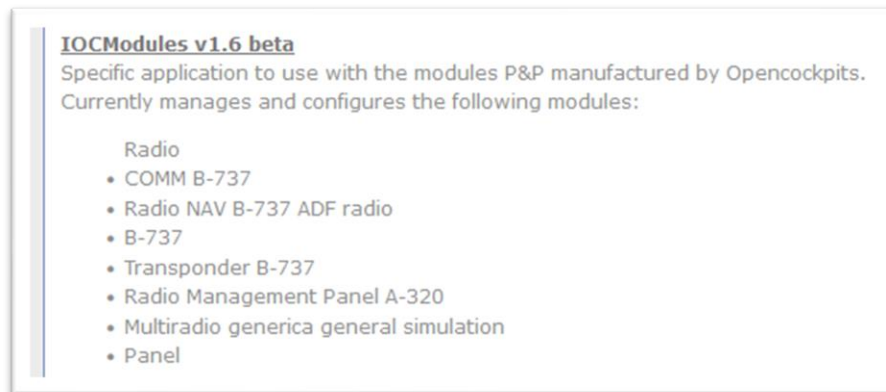
Note! The “Device index” (0, 1 and 2) for the three first devices are not random. Please read this in the file you downloaded and specified as “Configuration File” in SIOC.ini. In my case: **MCP_V3_prosim.txt** (the name specified inside the file is something else – it doesn’t matter):

```
// *****
// * Opencockpits MCP V3 - By Manolo Vélez - www.opencockpits.com
// * Adapted by Marty Bochane - Prosim
// *****
// * FileName : prosim_mcp_V3.txt
// * Date : 2013-02-14
//
//
// This SIOC script supports One MCP device, two EFIS devices and one CDU device
// For this script to work, the following device numbers are expected:
//
// Device 0: MCP
// Device 1: Efis Captain
// Device 2: Efis CO
//
// Devices do not need to be present, but if they are, they should be configured for the above device numbers.
// Configuration is done in the sioc.ini file with "master=" lines.
```

The order of other devices is not important. Just add new devices to the bottom of the list. Use next available number for “Device index”, assign “Type number” and “Number of cards”, and use a temporary/random number for the “Device number”, restart or reload SIOC, observe new line/lines in the “Devices”-view with a “*”, and edit SIOC.ini as explained above to correct the device number. Restart or reload SIOC, and check that everything is okay.

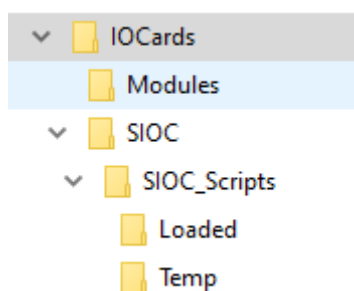
IOCMModules

Next to do (for the “IOCMModules-method” **only**), is to install and configure IOCMModules. You can download IOCMModules from the same web-page where you found the SIOC software. Look for:



Download and unzip IOCMModules16Beta1.zip. Run Setup.exe as Administrator. My installation path is C:\IOCards\Modules, which contains IOCMModules.ini.

This is the filesystem on my C-drive after installing SIOC and IOCMModules (you don't have to use the C-drive):



Note! You need to check/edit C:\IOCards\Modules\IOCMModules.ini. See next page.

IOCMModules.ini

Fragments from IOCMModules.ini:

```
[Brillo por defecto 1-121]
[Bright 1-121]
bright=65

[ Valores iniciales ]
[ Initial values ]

set_com1=118000
set_com2=118000
set_nav1=10800
set_nav2=10800
set_adf1=1000
set_adf2=1000
set_atc=1200

[ Activacion de Radios ]
[ Radio Activation ]

active_com1=Yes
active_com2=Yes
active_nav1=Yes
active_nav2=Yes
active_adf1=Yes
active_adf2=No
active_atc=Yes

[ Orden para COM , NAV y ADF]
[ COM , NAV % ADF modules order ]
[ Change to NO for device number minor assigned to COM2 , NAV2 or ADF2, if YES,
minor device number is COM1 , NAV1 or ADF1]
FIRST_DEVICE_COM1=Yes
FIRST_DEVICE_NAV1=Yes
FIRST_DEVICE_ADF1=Yes
FIRST_DEVICE_RMP1=Yes
```

You can set the brightens you prefer for the devices. I found 121 (default) too bright, and reduced the value from 121 to 65.

More important is the [Radio Activation] section. Set **No** for all the devices you don't have, or shall not be used by IOCMModules (if any).

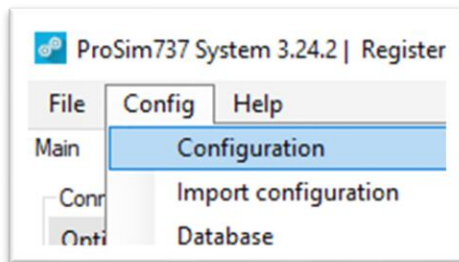
You can use the [COM , NAV % ADF modules order] section if you have to swap two identical devices. For example: If you change the frequency with COM1, and ProSim thinks you change COM2, then you set FIRST_DEVICE_COM1=No to get it right.

The last thing to do before all the P&P devices are working with ProSim, is to configure "IOCP Server" in ProSim, see next page.

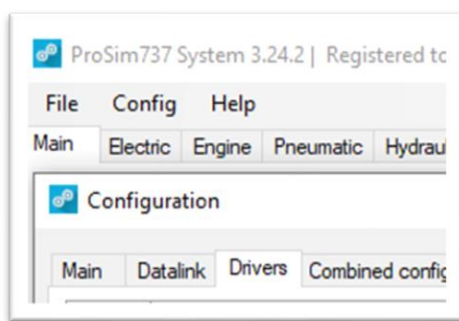
Configure IOCP Server in ProSim

When using SIOC, you need to configure “IOCP Server” in ProSim.

Start ProSim, and use menu-item: *Config/Configuration*:



Select *Drivers-tab*:



Use one “IOCP Server” for each “SIOC” running in your SIM.

Example with two computers running SIOC:

+	IOCP Server	127.0.0.1	8092
-	IOCP Server	FlightSim2	8093

Use the [+] button to add a “IOCP Server” if you need more than one.

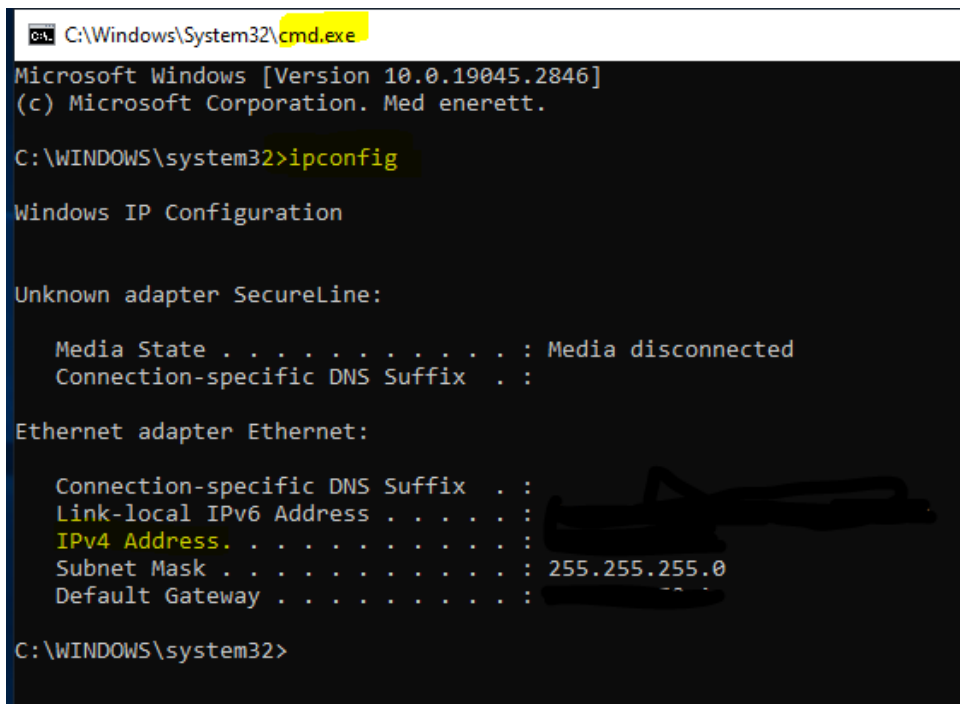
Each “IOCP Server” need this information:

- What is the IP-address for the computer running SIOC?
- Which Port is SIOC using on this computer?

You can use several methods to find the IP-address. I prefer the method described on the next page.

IP-address

You can find the IP-address (“IPv4 Address” in the screenshot below) for the computer by running the **ipconfig** command in a DOS-window (“RUN cmd.exe” first) like this:



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.2846]
(c) Microsoft Corporation. Med enerett.

C:\WINDOWS\system32>ipconfig

Windows IP Configuration

Unknown adapter SecureLine:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . :

Ethernet adapter Ethernet:

    Connection-specific DNS Suffix  . :
    Link-local IPv6 Address . . . . . :
    IPv4 Address. . . . . :
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . :

C:\WINDOWS\system32>
```

The “IPv4 Address” is masked in the screenshot above.

In ProSim, the IP-address can be specified by one of these options:

- **190.186.45.3** (The IP-address for “any computer”, 190.186.45.3 is only an example)
- **127.0.0.1** (“this computer”)
- **Localhost** (“this computer”)
- **FlightSim1** (The name for “any computer”, FlightSim1 is only an example)

You can use **127.0.0.1** or **Localhost** only for the computer running both SIOC and ProSim.

See next page for how to find and edit the Port.

Port

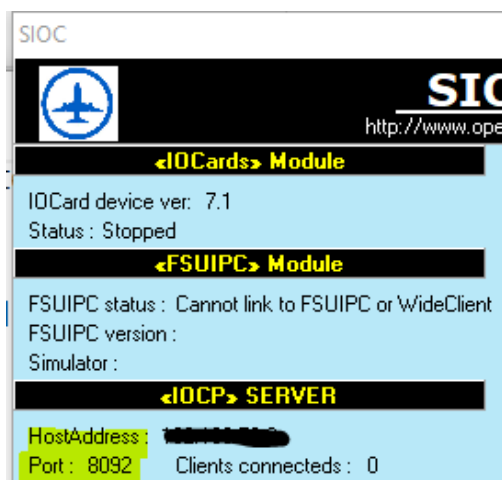
You can edit the Port in SIOC.ini (if you need it). 8092 is default Port:

```
[***** SIOC *****]

[ Nombre asignado al SIOC ]
[ SIOC name ]
Name=MAIN_SIOC

[ Puerto del servidor IOCP ]
[ IOCP port ]
IOCP_port=8092
```

The Host-address (IP-Address) and Port used, are displayed in SIOC:



If you have two or more computers running SIOC, you have to edit SIOC.ini on each additional computer regarding **Name** and **Port**. This is from my second computer (FlightSim2):

```
[ Nombre asignado al SIOC ]
[ SIOC name ]
Name=SECOND_SIOC

[ Puerto del servidor IOCP ]
[ IOCP port ]
IOCP_port=8093
```

On FlightSim2, I changed the name to “SECOND_SIOC” and the Port to 8093.

You *can* run SIOC on one computer only, but I find it convenient to use two computers. I use FlightSim1 for “devices”, and FlightSim2 for “cards”. It also gives me some load balancing, I guess ...

Note! After you have added the “IOCP Server” information in ProSim, and if you experience some strange behavior thereafter, restart the computer running ProSim. Do not perform a “shutdown and startup”, do a “restart”. Restart makes a clean start. Do the restart before you try other troubleshooting. The restart often helps.

The connection between the computers must be open. This guide does not cover firewall or network configurations, so if you experience connection problems you have to seek help somewhere else (sorry!).

To summarize, these are my two “IOCP Servers” configured in ProSim:

	IOCP Server	127.0.0.1	8092
	IOCP Server	FlightSim2	8093

Changing from “IOCMModules” to “SIOC Direct”

You can use this description if you prefer to start with the “SIOC Direct-method” in the first place.

Both methods are using “SIOC” and “IOCP server”, but the “SIOC Direct-method” doesn’t use any “SIOC scripts”, nor “IOCMModules”.

Install SIOC if you have not done it already (see section [SIOC](#)).

You shall not specify any “Configuration File” in SIOC.ini. Just convert the line to a comment, like this:

```
[ Fichero de configuracion ]  
[ Configuration File ]  
[ CONFIG_FILE=C:\Iocards\SIOC\SIOC_Scripts\MCP_V3_prosim.txt ]
```

Or this if you start with the “SIOC Direct-method” right away:

```
[ Fichero de configuracion ]  
[ Configuration File ]  
[ CONFIG_FILE= ]
```

The list of P&P devices is the same as for the “IOCMModules-method”. Leave the list as-is if you are a former “IOCMModules-method”-user. If you are a “new” user, follow the same procedure as described under section [SIOC.ini](#) to configure your devices.

If you already have IOCMModules installed, you don’t have to remove the software, **just not start it**, and the IOCMModules.ini file is not longer in use. Leave the file as-is if you are a former “IOCMModules-method”-user.

In ProSim, you have to configure one or more “IOCP Server”. Follow the procedure as described in [Configure IOCP Server in ProSim](#), if you are a “new” user.

Very important! If you have configured SIOC variables in ProSim related to P&P devices, you have to remove them! For example: I had configured the TCAS switch in ProSim. The switch did not work with the “SIOC Direct”-method before I removed my configurations for TCAS in ProSim.

To summarize:

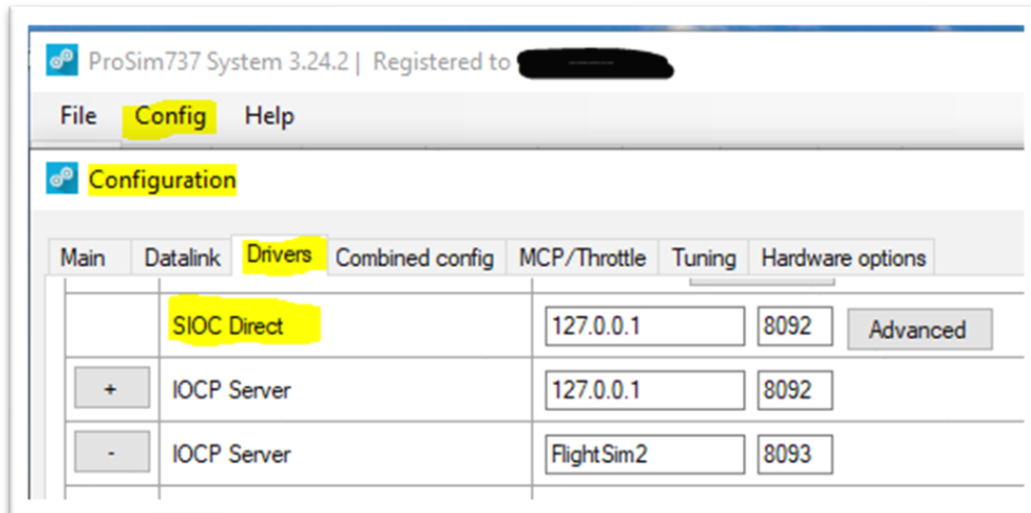
- You need SIOC
- You shall not specify any “Configuration File” in SIOC.ini
- You need to specify all your P&P devices in SIOC.ini (same as for the “IOCMModules-method”)
- You don’t need IOCMModules, do not install it, and not start it if installed
- You have to configure IOCP Server in ProSim (same as for the “IOCMModules-method”)
- Remember to remove “SIOC variable configurations” for P&P devices in ProSim if you have any

Note! In addition, you need to configure “SIOC Direct” in ProSim. See next page.

Configure SIOC Direct in ProSim

If you want to use the “SIOC Direct”-method, you need to configure “SIOC Direct” in ProSim after you have configured devices in SIOC.ini.

Start ProSim and use *Config/Configuration* and *Drivers* to configure “SIOC Direct”:



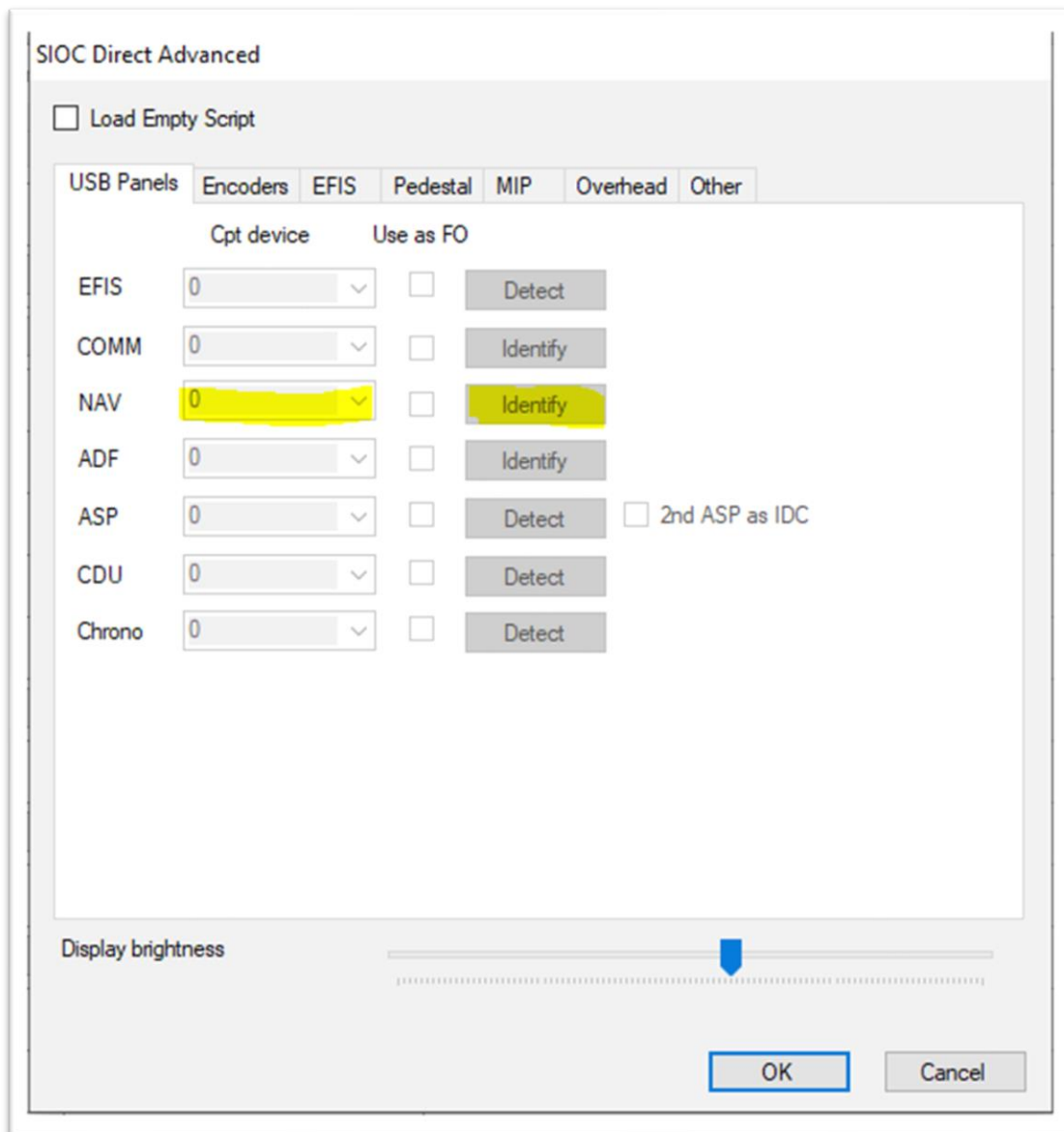
- In my case, the first “IOCP Server” (IP-address 127.0.0.1 and Port 8092) is configured for P&P devices. I use the same for “**SIOC Direct**”.
- Click the [OK] button at the bottom of the *Drivers* tab (the button is not visible in the screenshot above) before you do anything else.
- Re-open *Config/Configuration* and *Drivers*, and click the [Advanced] button for “SIOC Direct”.

SIOC Direct USB Panels

SIOC Direct will automatically detect all the devices you have added in SIOC.ini, but SIOC Direct doesn't know which device is for the Captain and which device is for the F/O, if you have two of the same kind. We need to help ProSim identify devices for the captain.

Example if you have two NAV devices:

When you click the **Identify** button (see the "NAV row" in the screenshot below), a number is displayed in each NAV device in your SIM. Let's assume number 37 appears in the NAV device for the Captain, and 38 in the NAV device for the F/O. The "select/drop-down" (marked **yellow** in the screenshot below) become clickable, and will contain 37 and 38. Choose 37, and ProSim will understand which NAV belongs to the Captain. The NAV for the F/O is given after this operation.



Click the **[OK]** button to confirm and save the settings.

SIOC Direct Encoders

The defaults values for encoders are “3” for “Clicks” and “10” for “Gain”. It didn't fit me, and I did some changes, but each must decide what optimal settings are.

Test the SIM with defaults values, and adjust if necessary.

These are just examples:

The screenshot shows the 'SIOC Direct Advanced' dialog box with the 'Encoders' tab selected. At the top left, there is a checkbox labeled 'Load Empty Script' which is currently unchecked. Below this, there are several tabs: 'USB Panels', 'Encoders' (selected), 'EFIS', 'Pedestal', 'MIP', 'Overhead', and 'Other'. The main area contains a table with four columns: 'Encoder', 'Clicks', 'Gain', and 'Alps Encoder'. The table lists various encoders with their respective settings. At the bottom of the dialog, there is a 'Display brightness' slider and two buttons: 'OK' and 'Cancel'.

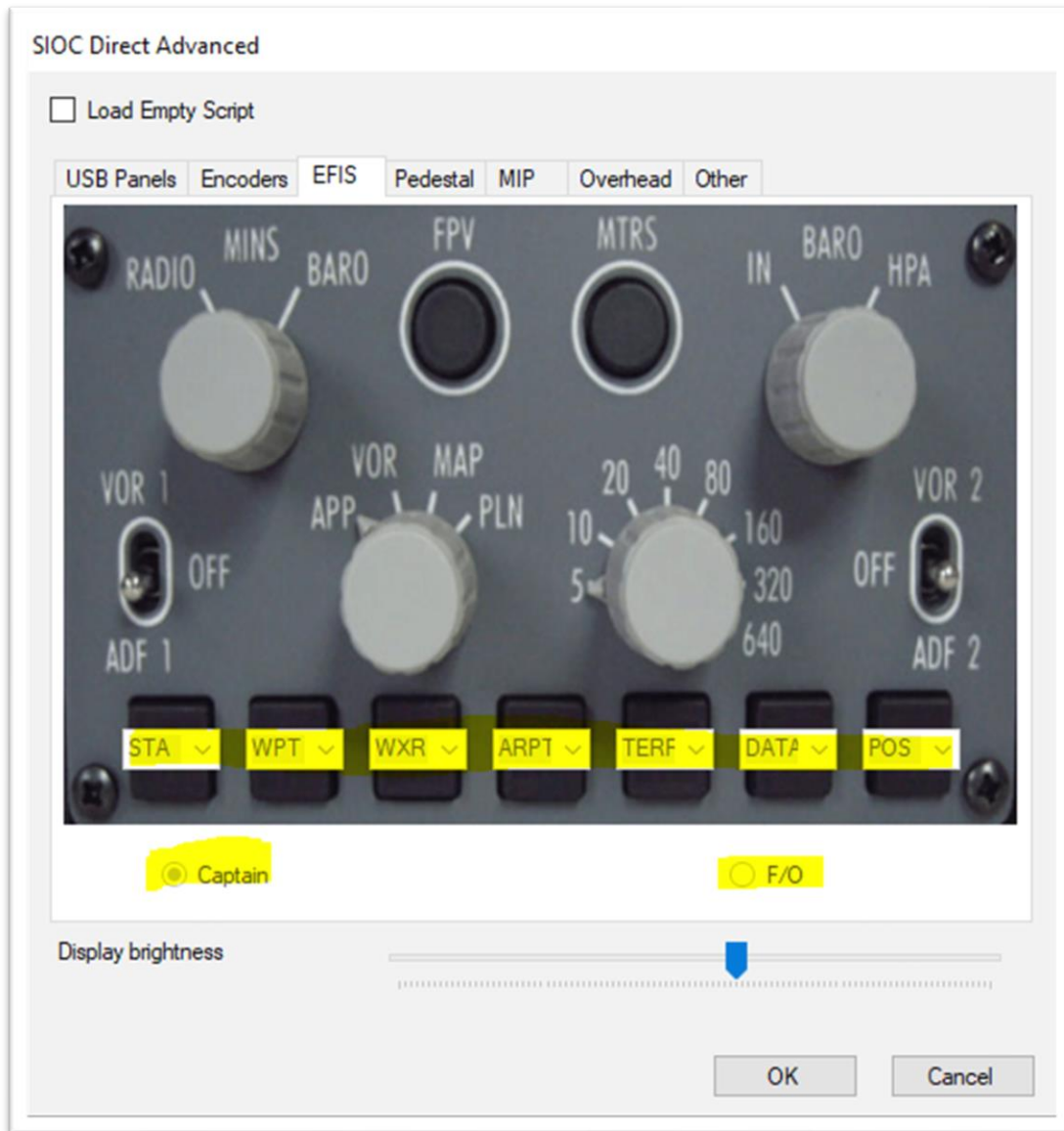
Encoder	Clicks	Gain	Alps Encoder
CRS1	5	10	<input type="checkbox"/>
SPD	6	10	<input type="checkbox"/>
HDG	6	10	<input type="checkbox"/>
ALT	5	10	<input type="checkbox"/>
VS	5	10	<input type="checkbox"/>
CRS2	5	10	<input type="checkbox"/>
BARO1	3	10	<input type="checkbox"/>
MINS1	5	10	<input type="checkbox"/>
BARO2	3	10	<input type="checkbox"/>
	3	10	<input type="checkbox"/>
MINS2			

Click the **[OK]** button to confirm and save the settings.

SIOC Direct EFIS

You must check the assignments for the buttons at the bottom of each EFIS. Be aware of which EFIS you are working with (*Captain* or *F/O*).

On each button, there is a selector (drop-down) where you choose which function you want. Select a function for each button, and repeat the operation for both EFIS.

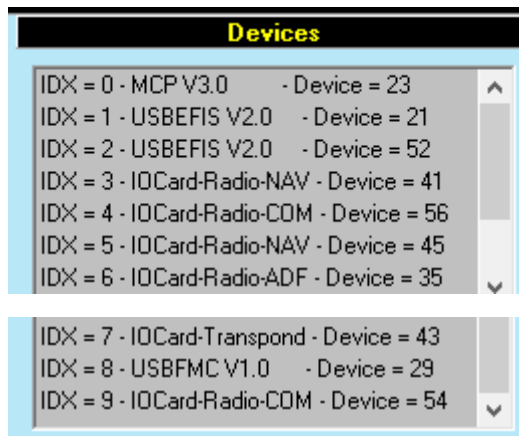
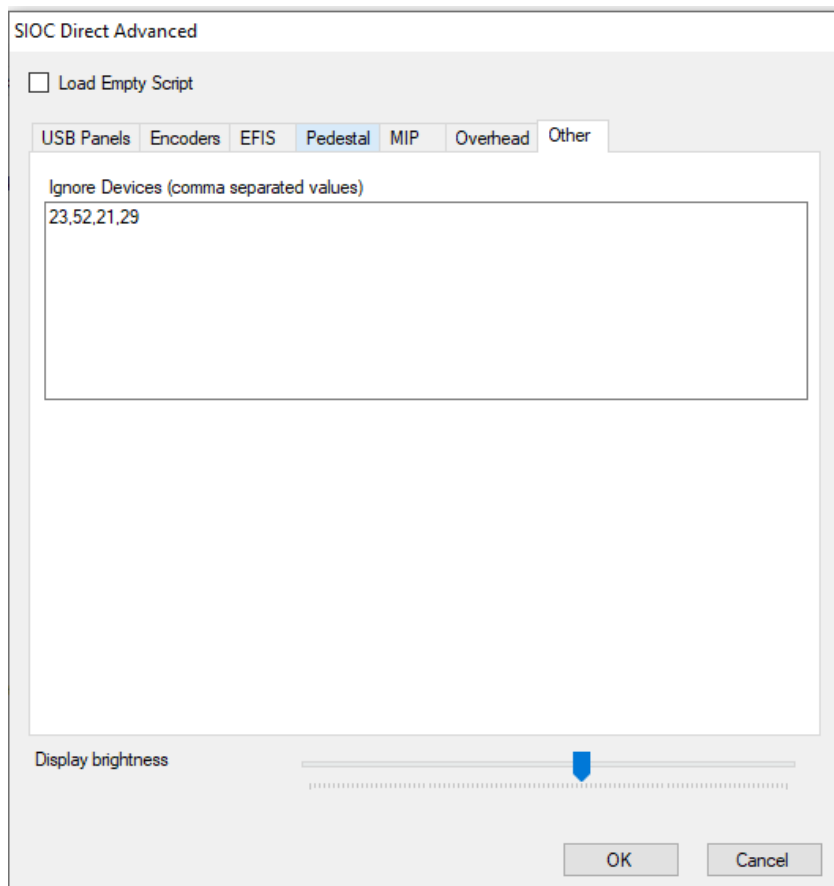


Click the **[OK]** button to confirm and save the settings.

Tip: Restart the computer after the first time you configured “SIOC Direct”. Later, you can change “SIOC Direct”-settings without restarting the computer.

SIOC Direct Other

If you want to exclude devices from SIOC Direct, you have to tell SIOC Direct which devices to ignore. In the example below, I have excluded four devices (23,52,21,29).



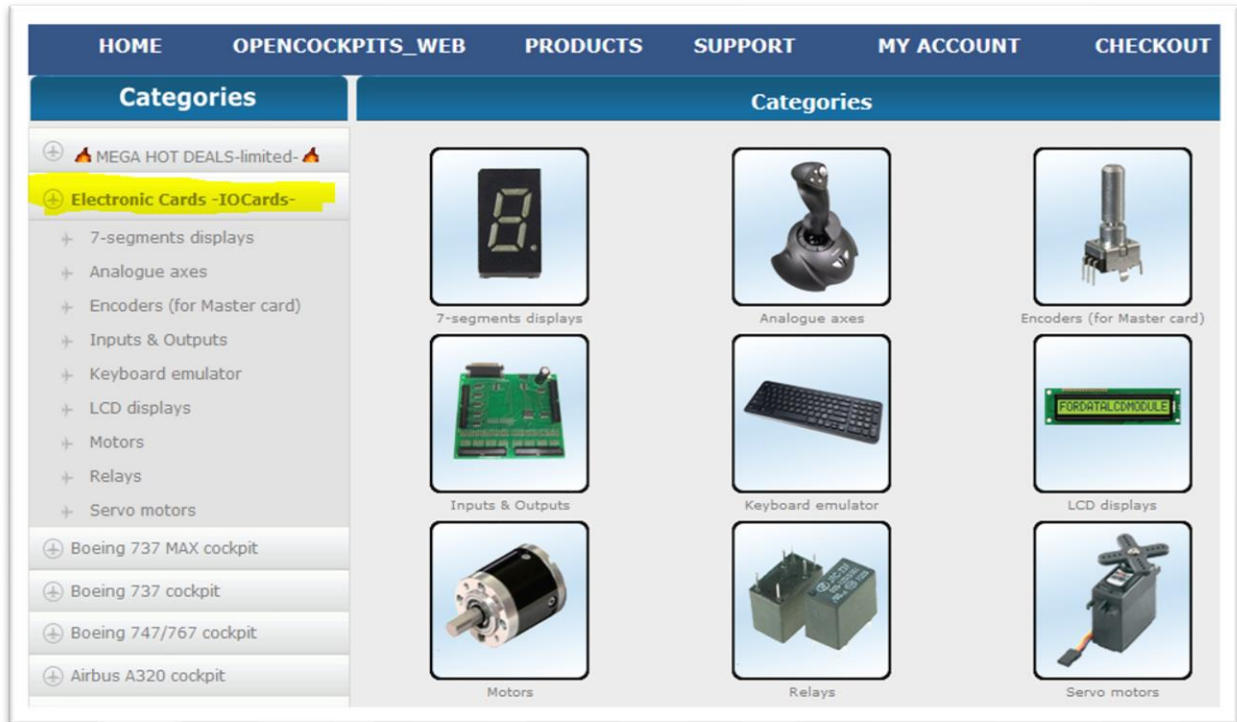
Note! Use "Device = "-number (not IDX). Remember to edit the list if Windows assigns new device-numbers.

I have instructed SIOC Direct to ignore the MCP (23), both EFIS (52 and 21), and the FMC (29). **Why?** Please see the Combination-method.

Adding OC cards in SIOC.ini

So far, this guide has covered OC P&P devices, but OC has a full range of cards you can use to connect hardware to the SIM.

Open the [web-shop](#) and click on menu-item “Electric Cards -IOCards-” to the left:



I will only cover cards I have in my SIM, which are:

1. USB Expansion card
2. Master card
3. Inputs connection card
4. Outputs connection card
5. Displays II card
6. USB servo motors card

Please read this review about the first four cards on the list above:

<https://www.opencockpits.com/uploads/tutoriales/Review%20Expansion%20Master%20entradas%20y%20salidas%20Ray%20aviation.pdf>

I will not cover the connection between the cards in details. It's very well documented in the link above, but I would like to mention:

- the first five cards on the list above are connected to each other
- the “USB servo motors card” is a “stand-alone card”

As mentioned several times, I have two computers in the SIM. FlightSim1 is the main computer running most of the software. FlightSim2 is used for displays/audio and the second SIOC installation.

FlightSim1 has all the OC P&P devices connected, and FlightSim2 all the OC cards connected. I'm very happy with this configuration, and have no problem recommending this to others. That doesn't mean there aren't other good alternatives.

Note! Everything below applies to FlightSim2, and the second installation of SIOC. Among all the cards I have in the SIM, only these cards are defined in sioc.ini, and connected directly to FlightSim2:

- USB Expansion cards (two units)
- USB servo motors card (one unit)

I'm using the same filename and path ("C:\IOCards\SIOC\sioc.ini") on both computers.

Remember to change the **Name** and the **Port** for the second installation of SIOC (described earlier in this guide). I also have a dedicated **Configuration File** (SIOC script) for the cards, "ProSim_B737-800.txt", as you can see below:

```
[***** SIOC *****]
[ Nombre asignado al SIOC ]
[ SIOC name ]
Name=SECOND_SIOC

[ Puerto del servidor IOCP ]
[ IOCP port ]
IOCP_port=8093

[ Tiempo de respuesta máximo de los paquetes IOCP ]
[ IOCP Timeout ]
IOCP_timeout=3500

[ Arranque minimizado en la barra ]
[ Start minimized in tray ]
Minimized=Yes

[ Retraso necesario para las variables toggles (Project Magenta) ]
[ Delay needed for var. toggles (Project Magenta) ]
Toggle_delay=20

[ Fichero de configuracion ]
[ Configuration File ]
CONFIG_FILE=C:\Iocards\SIOC\SIOC_Scripts\ProSim_B737-800.txt
```

Like configuration of "devices", we need to use the same syntax for "cards":

MASTER=(Device index), (Type), (Number of cards), (Device number)

For **two** USB Expansion cards:

- Device index: **Next free number** (continuing from FlightSim1)
- Type: **4**
- Number of cards: **2**
- Device number: **Given by Windows**

Next page shows the actual configuration.

```
[ My first USB Expansion card, used for the MIP ]  
MASTER=11,4,2,1
```

```
[ My second USB Expansion card, used for the OverHead panel ]  
MASTER=12,4,2,5
```

The USB Servo motor card has a simple syntax:

```
Name_of_card=(Device index), (Device number)
```

- Device index: **Next free number**
- Device number: **Given by Windows**

I'm using this:

```
[ Servo for engine start switch - auto-return ]  
USBServos=13,7
```

Remember to restart SIOC (FlightSim2 in my case) after you have changed the settings in SIOC.ini.

Connecting hardware to cards

We use SIOC scripts to define hardware we connect to cards. We “link” the variables to different cards, depending of the type of hardware.

My SIOC script is “C:\locards\SIOC\SIOC_Scripts\ProSim_B737-800.txt”. The script contains all the switches, annunciators, etc., I have in the SIM (MIP and OverHead). Later, I show you how I created the script from scratch.

IOCARD_SW

This part explain how you use “Inputs connection”-cards. Input-cards are connected to “Master”-cards. “Master”-cards are connected to a “USB Expansion”-card.

This fragment (from ProSim_B737-800.txt) shows inputs (switches):

```
Var 2040, Link IOCARD_SW, Device 11, Input 29 // Recall 1 Pushed  
Var 2041, Link IOCARD_SW, Device 11, Input 30 // Flap inhibit
```

Explanation:

Var 2040

The variable number

Link IOCARD_SW

The variable is linked to an input card

Device 11

This tells us which USB Expansion card (you have to use the IDX value) the input card is indirectly connected to (the Master card is between).

Input 29

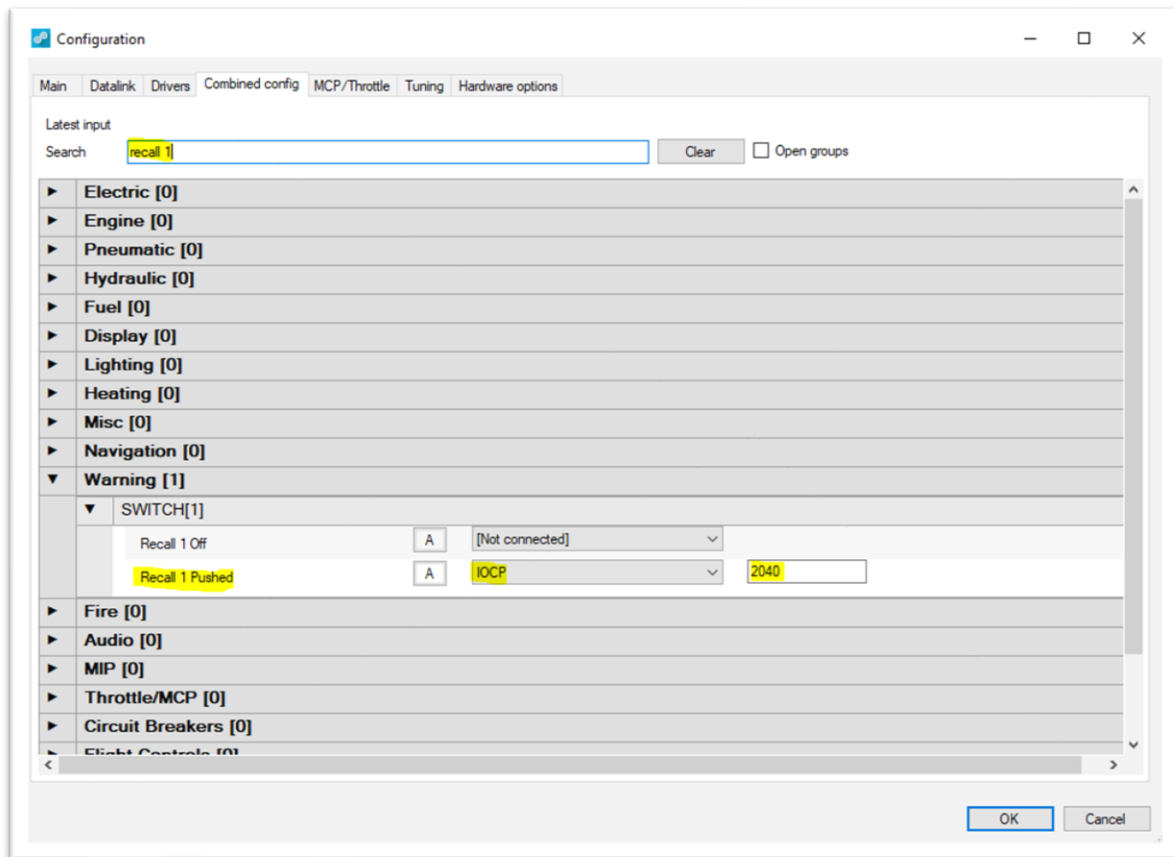
This tells us which connector the hardware is connected to

// Recall 1 Pushed

This is a comment telling use which switch is pushed in the SIM. The comment is related to the description used in ProSim.

The next page shows you how to interface this variable (or “switch”) with ProSim.

To configure variable 2040 (on previous page) in ProSim, you must find the function “Recall 1 Pushed” using “*Configuration/Combined config*”, and assign the function to “SIOC” variable “2040”:



Remember to click the **[OK]** button after changing the configuration in ProSim.

IOCARD_OUT

“Outputs connection”-cards and “Inputs connection”-cards are very similar. It is not easy to see the difference at first glance. As input-cards, output-cards are connected to “Master”-cards.

This fragment (from ProSim_B737-800.txt) shows outputs (annunciators):

```
Var 3043, Link IOCARD_OUT, Device 11, Output 22 // Autobrake disarm  
Var 3044, Link IOCARD_OUT, Device 11, Output 37 // Anti Skid INOP
```

Explanation:

Var 3043

The variable number

Link IOCARD_OUT

The variable is linked to an output card

Device 11

This tells us which USB Expansion card the output card is indirectly connected to

Output 22

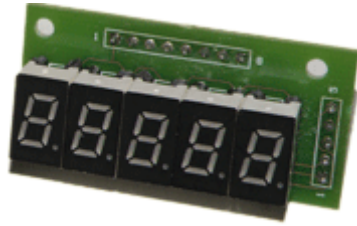
This tells us which connector the hardware is connected to

// Autobrake disarm

This is a comment telling use which annunciator shall light up in the SIM

IOCARD_DISPLAY

We use IOCARD_DISPLAY to (for example) display the LAND ALT value. The “7 segment display digits”-component below, is connected to a “Displays II”-card. “Displays II”-cards are connected to “Master”-cards. And again; Master-cards are connected to a “USB Expansion”-card.



This fragment (from ProSim_B737-800.txt) shows use of “7 segment display digits”:

```
Var 1320, name LA, Link IOCARD_DISPLAY, Device 5, Digit 69, Numbers 5 // Land Alt
Var 1325, name FA, Link IOCARD_DISPLAY, Device 5, Digit 64, Numbers 5 // Flt Alt
```

Explanation:

Var 1320

The variable number

Name LA

The variable name

Link IOCARD_DISPLAY

The variable is linked to a display-card

Device 5

This tells us which USB Expansion card the display card is indirectly connected to

Digit 69

See below

Numbers 5

See below

// Land Alt

A comment. In this case “Land Alt” (Landing altitude)

Rows for IOCARD_DISPLAY does not use **Output**. They are using **Digit** and **Numbers** instead.

The **Digit** represent one of the five units in the component above. The **Number** represent how large the number can be. For example: If **Number** is 3, you can display a number from 000 to 999. If **Number** is 1, you can display a number from 0 to 9.

Digit and **Numbers** are further explain in [Landing altitude without leading zeros](#).

IOCARD_ENCODER

We use “Inputs connection”-cards to connect an encoder. Definition for encoders is similar to switches.

Use IOCARD_ENCODER to (for example) set the LAND ALT value:

```
// Encoder input for LAND ALT  
Var 1332, Link IOCARD_ENCODER, Device 5, Input 128, Aceleration 3, Type 2
```

Explanation (attributes that differ from switches):

Input

When you connect an encoder to an **Input** card, you have to use two connectors (plus ground). An “ON/OFF” switch is using only one connector (plus ground). In the example above, we can assume the builder has connected the encoder to connector 128 and 129. You shall not specify the other connection (129 in this case) any other place.

Tip: If the encoder “turns the wrong direction”, you can swap the wires (between 128 and 129 in the case above).

Acceleration

How quickly the numbers should increase/decrease when turning the LAND ALT knob, is up to you to decide, and are defined with the Acceleration (Acceleration) value. You can start with 3 and change the value later if you are not satisfied.

Type

Type refers to what kind of encoder hardware you have installed in the SIM. I have never seen anything other than Type 2, but you can read more about encoders with this link:

https://www.opencockpits.com/uploads/datasheets/encoder_cts288.pdf

USB_SERVOS

We connect servo motors to “USB Servo Motors”-cards. You connect the card directly to the computer using a USB cable.

I’m using the servo to automatically turn the Engine Start switch from GRD to OFF when N2 is 56%.
(Sorry for the small font!)

```
Var 4060, name servo1, STATIC, Link USB_SERVOS, Device 12, Output 1, PosL 0, PosC 512, PosR 1023
```

Explanation:

Var 4060

The variable number

Name servo1

The variable name

STATIC

Keep the Var number (4060 in this case) unchanged if merging script files (explained later in this guide)

Link USB_SERVOS

The variable is linked to a servo

Device 12

This is defined in SIOC.ini

Output 1

The USB servo motors card I have can connect six servos (Output 1 to 6). I’m using the first connector.

PosL

This is the maximum left position the “servo-arm” can move. Usually 0 (zero), but you can use a higher number.

PosC

This is the center position

PosR

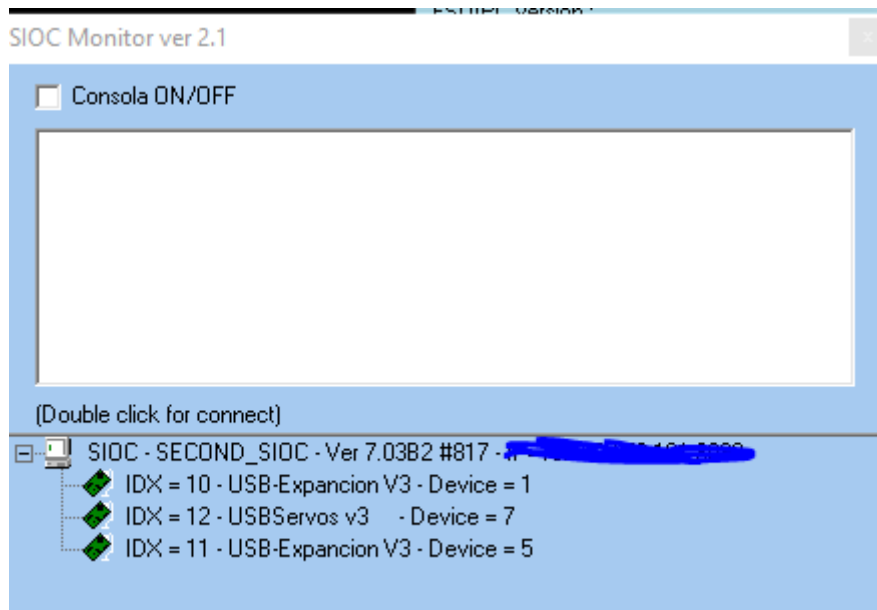
This is the maximum right position

Note! You find the values for PosL, PosC and PosR by using the [**SIOC Monitor**] button in SIOC. See next page.

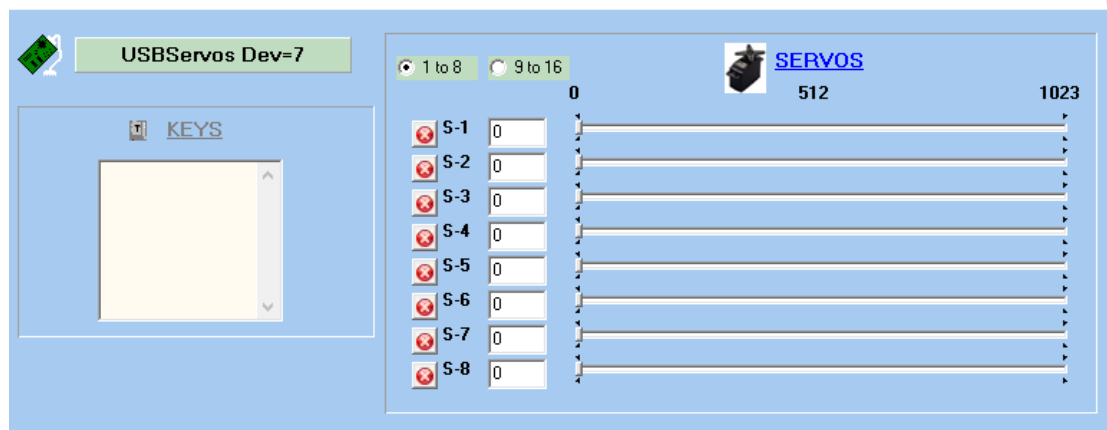
USB_SERVOS positions

Use [**SIOC Monitor**] and double-click on the servo device. See the screenshot below.

In my case: “**IDX = 12 - USBservos v3 - Device = 7**”.



You will see this page:



My servo is “S-1”. By moving the slider for “S-1”, I can see the position-value changing when I move the slider (the value is 0 in the screenshot above).

You need to see how the servo-arm moves when you move the slider. Make notes for max-left, center and max-right position, and use these values in the servo definition (see previous page).

SIOC scripting

Note! This guide is not meant to teach you all about SIOC scripting. The guide will only address a few basic points useful to know before studying/changing existing scripts, or making one from scratch.

First, we should split “scripting” into two levels:

1. Text editing
2. Programming

This is what I mean by “text editing”:

```
Var 201, name IAS_DUI, Link IOCARD_OUI, Output 20
Var 202, name O_N1, Link IOCARD_OUT, Output 21
Var 204, name O_AT, Link IOCARD_OUT, Output 22
Var 206, name O_SPEED, Link IOCARD_OUT, Output 23
Var 208, name O_LVLCHG, Link IOCARD_OUT, Output 24
Var 210, name O_VNAV, Link IOCARD_OUT, Output 25
Var 212, name O_HDGSEL, Link IOCARD_OUT, Output 26
Var 214, name O_APP, Link IOCARD_OUT, Output 27
Var 216, name O_VORLOC, Link IOCARD_OUT, Output 28
Var 218, name O_INAV, Link IOCARD_OUIT, Output 29
```

And this is “Programming”:

```
Var 302, name E_ALT, Link IOCARD_ENCODER, Input 2, Aceleration 4, Type 2
{
  If &Inhibit_Alt = 0
  {
    L0 = -100 * V302
    V106 = LIMIT 0,50000, L0
  }
}
```

Most of the scripting is text editing (no programming skills required). But if you need some functionality in the script, e.g.: The “Engine Start”-switch shall move from GRD to OFF automatically when N2 is 56%, then you have to do some programming.

I have created one script including both Master Instrument Panel and Overhead Panel. I have not configured all the functionality available in the panels yet (I have everything I need for daily use), and I assume the script consists of approximately 90% pure text and the rest is programming.

I mention this because I don't want to scare anyone from starting with SIOC scripting.

I will use the term “scripting” regardless of the “levels” mentioned above.

By-the-way: I guess you will see the use of “SIOC programming” and “SIOC code” from time to time. It is the same as “SIOC scripting” and “SIOC script”.

If you are familiar with computer programming, SIOC scripting is not hard to understand, but the *logic* can be a challenge in the beginning. For example, in this script, I have defined variable **Output** in line 7 after I have “used” the variable earlier in line 5:

```
1. var 0010, name Input
2. {
3.   L0 = 1
4.   L1 = 100
5.   &Output = RANDOM L0 L1
6. }
7. var 0020, name Output
```

For me, this is more logical: (**Define first, use later**)

```
1. var 0020, name Output
2. var 0010, name Input
3. {
4.   L0 = 1
5.   L1 = 100
6.   &Output = RANDOM L0 L1
7. }
```

It’s important to know that you can refer to a variable before you have defined it. When you know this, understanding existing scripts is much easier, and you can use this knowledge deliberately to achieve desired functionality.

If you test both versions above, you will learn the difference, when running the script first time, is:

- the upper version: **Output** has value 0 (zero)
- the lower version: **Output** gets a value between 1 and 100

Note! The line-numbers (1 to 7) are not part of the script. You need to use this if you want to test the script:

```
var 0010, name Input
{
  L0 = 1
  L1 = 100
  &Output = RANDOM L0 L1
}
var 0020, name Output
```

In SIOC scripts, **everything** is based on variables. A variable can represent itself, a switch, an annunciator, a servomotor, an encoder, and more.

I found this in “SIOC Reference Manual” explaining how to define a variable:

Variable definition

Var number **Attribute** parameter **Attribute** parameter (...)

{

Command parameters

Command parameters

Command parameters

...

}

number = Integer from 0 to 9999

parameter = information regarding attribute or command.

Examples:

```
Var 0001, Link FSUIPC_IN, Iffset $07F2, Length 2
```

```
Var 1002, Link IOCARD_ENCODER, Input 0, Aceleration 8, Type 1
```

```
Var 0086, Link IOCARD_DISPLAY, Digit 0, Numbers 5
```

```
Var 9023
```

And this, highlighting important programming rules (from “Introduction to SIOC”):

Scripts language main characteristics

- Different identifiers, variables, constants and other elements are always separated by **spaces**, **commas**, **brackets** or **tabulators**.
- { and } are used to indicate different **levels**.
- // can be used to insert general **comments** at the beginning of a line or at the end of a command line as a particular comment.
- Only **one definition** or **command** is allowed in each line.
- There's no difference between **uppercase** and **lowercase**.

23/10/2004

Introduction to SIOC

Separator

Defining variables with "space" as separator makes the script difficult to understand. I prefer to use "comma", and seems to be commonly used among script writers.

Variable numbers

Maximum number of variables are ten thousand (from 0 to 9999). You can use them with or without leading zeros: 1 and 0001 is the same variable.

You don't have to define variables in increasing order. SIOC will accept this:

```
var 0100, name First_Var, value 100
var 0300, name Second_Var, value 0
var 0200, name Third_Var, value 0
```

But each variable must have a unique number.

Variable names

If you have defined a variable with a name, you can use the variable later by referring to the name, or the number, like this:

```
var 1234, name Standby_Bat
&Standby_Bat = 1           // Using the variable name; Use & + the name
V1234 = 1                  // Using the variable number; Use V + the number
```

It is wise to give variables a name, especially if you shall use the variable somewhere else in the script. It's hard to remember what e.g. **V1234** represent. Names will make the script easier to understand/maintain.

Local variables

There are some "**local**" variables available too. You don't have to define them. They are:

- Real: L0, L1 and L2 (initial values are 0)
- Boolean: C0, C1 and C2 (initial values are false)

Local variables are in addition to the ten thousand variables SIOC allows, and you can use them as many times as you need in the same script. Next page shows an example where L0 is used three different places in the same script.

```
// This is fragment A in Script-ABC.txt
{
    L0 = &abc
    ...
}
.
.
// This is fragment B in Script-ABC.txt
{
    L0 = 100
    ...
}
.
.
// This is fragment C in Script-ABC.txt
{
    L0 = &xyz
    ...
}
```

You shall not use «&» when referring to *local* variables. Use (examples):

```
L0 = &abc      // Assign the value of &abc to L0
C0 = L0 > L1   // Assign True to C0 if the value of L0 is greater than the
               // value of L1, else the value of C0 is still False
```

Tip! Avoid lowercase for local variable L0, L1 and L2.

Lowercase in Notepad give us l0, l1 and l2

I don't know how many times I've read these as "ten", "eleven" and "twelve" 😞

SIOC is event driven

The most important to know about SIOC is that nothing will happen in the program (script) before a variable change its value. When you start SIOC, the script file you entered in SIOC.ini as the “Configuration File” will run once and then wait for something to happen in the SIM; SIOC is “event driven”.

Let me explain with this example:

```
// *****  
// * Config_SIOC ver 5.2 - By Manuel Velez - www.opencockpits.com  
// *****  
// * FileName : Test-script.txt  
// * Date : 21.04.2023  
  
var 0001, name Input // Change this variable  
{  
  L0 = 1  
  L1 = 100  
  &output = RANDOM L0 L1  
}  
  
Var 0002, name Output // Output will change randomly between 1 and 100 when alter Input
```

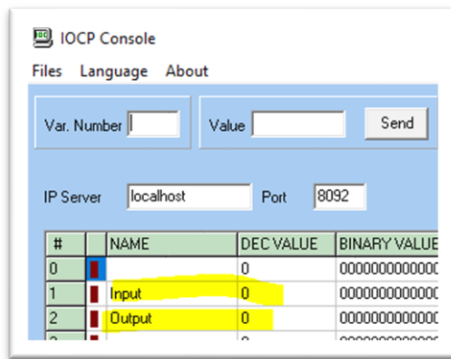
1. Copy the code below into Notepad, and save the file as Test-script.txt

```
var 0001, name Input  
{  
  L0 = 1  
  L1 = 100  
  &output = RANDOM L0 L1  
}  
var 0002, name Output
```

2. Use the file above as “Configuration File” in SIOC.ini:
CONFIG_FILE=C:\IOCards\SIOC\SIOC_Scripts\Test-script.txt.
3. Start or Reload SIOC.

The purpose of this tiny script is to assign a random value to **Output** every time we change the value of **Input**. The random value for **Output** will be between 1 and 100.

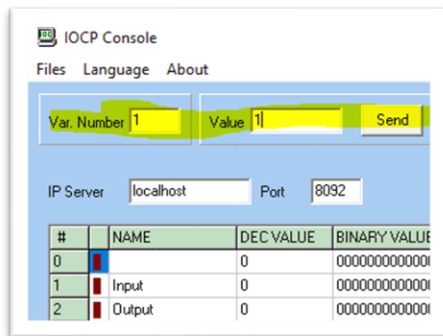
You have to use the [IOCPConsole] button in SIOC to test the script. Click [IOCPConsole] and you will see this (next page).



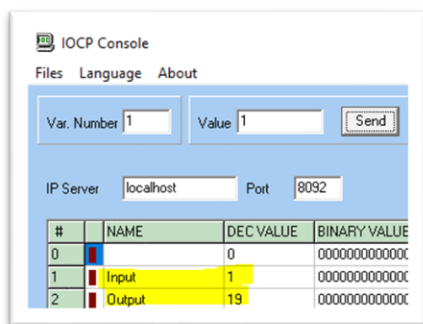
As we can see above, initially both **Input** and **Output** have a value of 0 (zero). This is because both variables were defined without the “value” attribute. “No value” is the same as 0.

If we “Send” value 1 to **Input** (variable 1), the value of **Output** (variable 2) will change automatically.

This is before “Send”:

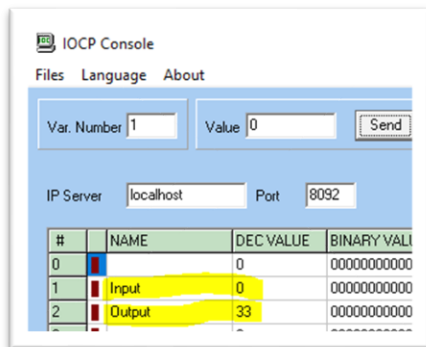


Here we can see **Output** is 19, after changing Input from 0 to 1 (using the [Send] button):



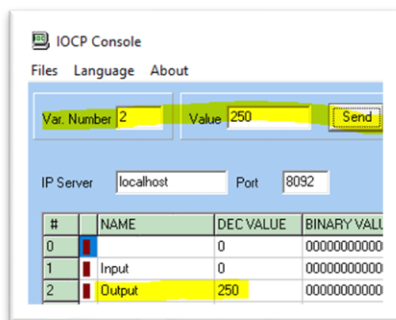
Continues next page.

If we want to give **Output** a new value, we need to change **Input** to something other than 1, for example back to 0. Here we can see **Output** gets 33, after changing **Input** from 1 back to 0:



If we “Send” the same value to **Input**, **Output** will not change because nothing has changed. As long as a variable-value is not changing - nothing happens.

We can send a new value directly to **Output**, and the script will execute because we are changing **Output**. Here we can see the result of sending 250 directly to **Output**:



Continues next page.

Let me explain the script one more time:

```
1. var 0001, name Input
2. {
3.   L0 = 1
4.   L1 = 100
5.   &output = RANDOM L0 L1
6. }
7. var 0002, name Output
```

Every time we change the value of **Input**, all the lines between the two brackets {...} (line 3 to 5) will be executed:

3. **L0** gets 1
4. **L1** gets 100
5. **Output** gets a random value between the value of **L0** and **L1**

When **Output** changes in line 5, SIOC must check: What will happen if **Output** is changed?

Answer: Nothing more than changing **Output** itself

Because there is no “action” {...} related to a value-change of **Output** (in line 7).

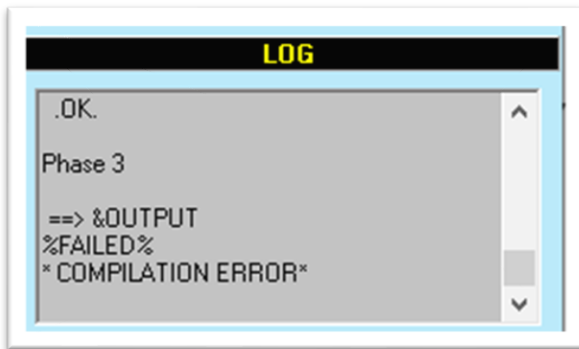
The script is waiting for the next event. The script stops when we stop SIOC, or load another script.

By-the-way: The script could be like this, but I wanted to show you the use of *local* variables:

```
var 0001, name Input
{
  &output = RANDOM 1 100
}
var 0002, name Output
```

Error handling

If there are one or more errors in the script, SIOC will display an error message in the LOG-view, like this:



Note! Every time you change and save the script in Notepad, you need to use the **[Reload]** button in SIOC so SIOC can reload and use the modified script.

Note! If SIOC cannot find the "Configuration File", you will see this error message (example):

Launch Compiler, file:

C:\IIOCards\SIOC\SIOC_Scripts\Test-script-x.txt

%FAILED%

*** COMPILATION ERROR***

The message is a little "diffuse". When I got this message, I thought there was something wrong with the script (not that the script was missing).

You are hereby warned 😊

Comments

Don't be afraid to make extensive use of comments. Script content often seems obvious and easy to understand during scripting, but can be difficult to understand later.

Comments in scripts starts with two “//”.

Example:

```
// this is a comment
```

You can use comments at the begging and at the end of a line, like this:

```
// *****
// * Config_SIOC ver 5.2 - By Manuel Velez - www.opencockpits.com
// *****
// * FileName : Test-script.txt
// * Date : 21.04.2023

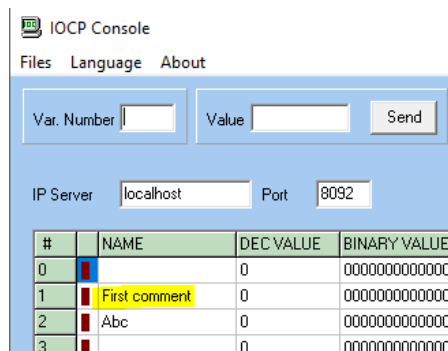
// Every time this variable changes its value, Output is assigned a new value too
var 0001, name Input
{
  L0 = 1 // Lowest random value
  L1 = 100 // Highest random value
  &output = RANDOM L0 L1 // Assign Output a random value between L0 and L1
}

// This variable does not do anything, except showing us the result of changing the value of Input
Var 0002, name Output
```

By-the-way: If we use a comment for a variable without name, we will see the comment as a “name” in IOCP Console:

```
//
var 0001, value 0 // First comment
var 0002, value 0, name Abc // Second comment
```

IOCP Console shows us:



The screenshot shows the IOCP Console window with a menu bar (Files, Language, About) and input fields for Var. Number, Value, and Send. Below these are fields for IP Server (localhost) and Port (8092). A table displays the current state of variables:

#	NAME	DEC VALUE	BINARY VALUE
0		0	000000000000
1	First comment	0	000000000000
2	Abc	0	000000000000
3		0	000000000000

The script I have used to demonstrate SIOC scripting so far, has nothing to do with the SIM, but next page shows an example how to achieve automatic return of the “Engine 1 Start switch” from GRD to OFF when N2 is approximately 56%.

Auto-return for Engine Start switch

The example below uses "Device 0". It's only a demo. You must use what's applicable in your SIM anyway.

```
// =====  
// Start: Engine 1 Start AUTO-OFF  
// Servo position 530 is "neutral" position  
// The chain of events:  
// 1. At 56% N2, the start switch is pushed (by the servo) from GRD to OFF (pushback solenoid)  
// 2. After one second, the servo returns to neutral position  
// =====  
  
Var 4059, name iniciaeng1, STATIC, Value 0 // Set engine 1 rotary switch to OFF  
{  
  &eng1 = 1  
}  
// This is the servo device  
Var 4060, name servo1, STATIC, Link USB_SERVOS, Device 0, Output 1, PosL 0, PosC 512, PosR 1023  
{  
  &servo1 = DELAY 530 ,100 // Set the servo position back to 530 after 1 second  
  &eng1 = 0  
}  
Var 4061, name eng1, STATIC // Define this in ProSim: Start 1 pushback solenoid  
{  
  IF &eng1 = 1  
  {  
    &servo1 = 755 // Move the servo to position 755 when N2 is 56%  
  }  
}  
// =====
```

To simulate the functionality of the script, I have renumbered the variables and deleted the link to the servo-card. I did it because I want to see all the variables I'm using on top of the SIOC Console, and I have no cards connected to the computer.

Note! You can use scripts with Link attributes during testing, because SIOC will not fail if cards are not present.

```
Var 0, name iniciaeng1, STATIC, Value 0 // Set engine 1 rotary switch to OFF  
{  
  &eng1 = 1  
}  
// This is the servo device  
Var 1, name servo1, STATIC  
{  
  &servo1 = DELAY 530 ,100 // Set the servo position back to 530 after 1 second  
  &eng1 = 0  
}  
Var 2, name eng1, STATIC // Define this in ProSim: Start 1 pushback solenoid  
{  
  IF &eng1 = 1  
  {  
    &servo1 = 755 // Move the servo to position 755 when N2 is 56%  
  }  
}
```

This is the script if you want to cut/paste:

```
//
Var 0, name iniciaeng1, STATIC, Value 0 // Set engine 1 rotary switch to OFF
{
  &eng1 = 1
}
// This is the servo device
Var 1, name servo1, STATIC
{
  &servo1 = DELAY 530 100 // Set the servo position back to 530 after 1 second
  &eng1 = 0
}
Var 2, name eng1, STATIC // Define this in ProSim: Start 1 pushback solenoid
{
  IF &eng1 = 1
  {
    &servo1 = 755 // Move the servo to position 755 when N2 is 56%
  }
}
```

Note! Var 0 is often reserved to specific use, but in this test it doesn't matter.

What happens when SIOC starts?

```
Var 0, name iniciaeng1, STATIC, Value 0 // Set engine 1 rotary switch to OFF
{
  &eng1 = 1
}
// This is the servo device
Var 1, name servo1, STATIC
{
  &servo1 = DELAY 530 ,100 // Set the servo position back to 530 after 1 second
  &eng1 = 0
}
Var 2, name eng1, STATIC // Define this in ProSim: Start 1 pushback solenoid
{
  IF &eng1 = 1
  {
    &servo1 = 755 // Move the servo to position 755 when N2 is 56%
  }
}
```

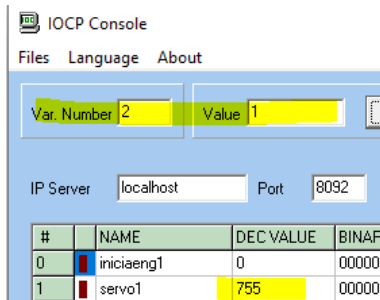
- **iniciaeng1** has value 0, and **eng1** gets value 1
- **servo1** gets value 530 (after one second), and **eng1** gets value 0
- **eng1** has value 0, and nothing more happens because the value of **eng1** is not equal 1

The IOCP Console confirms the observations above:

#	NAME	DEC VALUE
0	iniciaeng1	0
1	servo1	530
2	eng1	0
3		0

The only variable we shall change is **eng1** for “Start 1 pushback solenoid”, and is the only variable you need to configure in ProSim regarding this script.

If we send value 1 to **eng1**, we will see **servo1** equal 755 for one second. IOCP Console confirms:



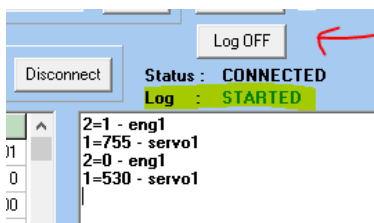
This is the script fragment in use when changing the value of **eng1**:

```
Var 2, name eng1, STATIC // Define this in ProSim
{
  IF &eng1 = 1
  {
    &servo1 = 755          // This will move the start switch back to OFF
  }
}
```

Before “Send”, **eng1** is equal 0, and we are now assigning **eng1** value 1. The **IF-test** is then TRUE, and **servo1** gets value 755. Because **servo1** gets a new value, our next focus is on this part:

```
Var 1, name servo1, STATIC
{
  &servo1 = DELAY 530 ,100 // Move the servo-arm back to neutral position
  &eng1 = 0
}
```

Initially **servo1** has value 530, but when **eng1** gets value 1, **servo1** gets value 755. Because **servo1** is changing from 530 to 755, the commands associating with a value-change of **servo1** are executed, and **servo1** is assigned back to 530. The value-change from 755 to 530 is delayed with one second. At the same time, **eng1** is changed back from 1 to 0. Again, SIOC checks what to do when **eng1** gets a new value, but as long as **eng1** is 0, nothing more will happen because of the IF-test.



You can click the [Log ON] button (now showing “Log OFF”) before you change the value of **eng1**. Then the Log-view will show you what is happening:

When N2 is 56%, ProSim will send “1” to variable **eng1**. SIOC will execute the chain of commands explain above. The servo will push the Engine Start switch back to OFF. The servo-arm will thereafter move back to neutral, and the system is ready to start the next engine. I know normal start sequence is engine 2 first, but is not important when it comes to this explanation.

DELAY command

This is the documentation for the **DELAY** command (from “SIOC Reference Manual”):

DELAY

Variable = DELAY **Parameter1** **Parameter2**

Runs the associated script previously waiting the time indicated by **Parameter2** (1/10 secs) and assigns the value indicated by **Parameter1** to the **variable**.

Parameter1 : Variable, real constant or integer constant. Value to set.

Parameter2 : Variable, real constant or integer constant. Delay time (1/10 secs).

Example:

```
V1354 = DELAY 57 10
```

Important to know! The DELAY command only delays the assignment of the specific variable. DELAY does not delay the execution of trailing commands!

Example:

```
&servo1 = DELAY 530 100 // servo1 gets value 530 after one second
&eng1 = 0                // eng1 gets value 0 immediately, and does
                        // not wait for the delay to finish
```

The LOG will show this order of events:

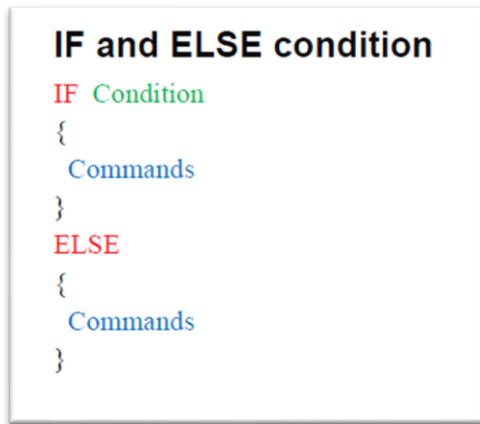
1. eng1 = 0
2. servo1 = 530

During testing, you can expand the delay. It's easier to see the delay if you use e.g. 5 seconds:

```
&servo1 = DELAY 530 500
```

IF ELSE command

IF tests are often used in scripts. This is the documentation for **IF ELSE** (from “SIOC Reference Manual”):



You can use an IF test alone (without the ELSE), and you can use several IF tests together. Example:

```
{
  IF L1 > 5
  {
    CALL V1000
  }
  ELSE
  {
    IF C2
    {
      L1 = L1 + 1
    }
  }
}
```

This script shows simple use of IF and ELSE:

```
//
var 0001, value 0
{
  if V0001 < 10
  {
    V0002 = V0001 + 10 // Do this as long as V0001 is less than 10
  }
  else
  {
    V0002 = V0001 - 10 // Do this as long as V0001 is 10 or more
  }
}
var 0002, value 0
```

Use [IOCPConsole], and change the value of variable 1.

MOD command

This is the documentation for the **MOD** command (from “SIOC Reference Manual”):

MOD

Variable = MOD **Parameter1** **Parameter2**

Calculates the remainder of the integer division **Parameter1** / **Parameter2**. The value is stored in **variable**.

Parameter1 : Variable, real constant or integer constant. Number to be divided.

Parameter2 : Variable, real constant or integer constant.

Example:

V0032 = MOD V0034 2

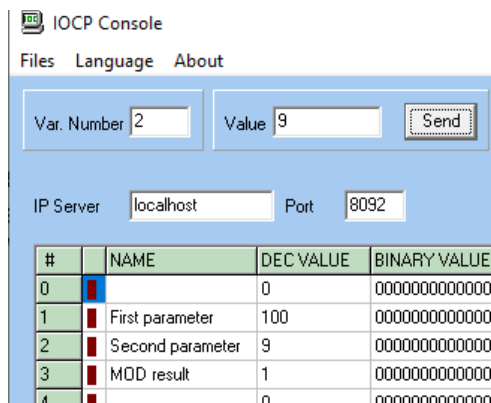
I found this general (not SIOC specific) explanation on the web:

*The modulo (or "modulus" or "mod") is **the remainder after dividing one number by another**. Example: 100 mod 9 equals 1. Because 100/9 = 11 with a remainder of 1. Another example: 14 mod 12 equals 2. Because 14/12 = 1 with a remainder of 2.*

We can test the explanation above with this script:

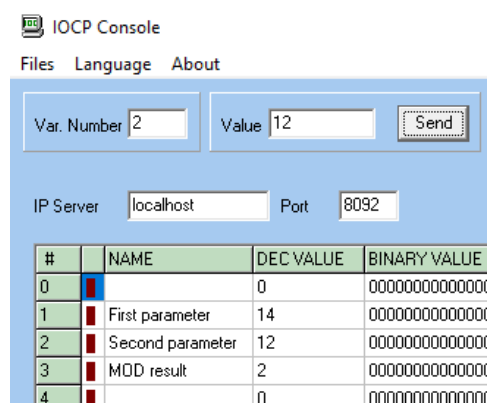
```
// MOD-test.txt
//
var 0001 // First parameter
var 0002 // Second parameter
{
  V0003 = MOD V0001 V0002
}
var 0003 // MOD result
```

The result for “100 MOD 9” and “14 MOD 12”:



The screenshot shows the IOCP Console interface. At the top, there are tabs for 'Files', 'Language', and 'About'. Below the tabs, there are input fields for 'Var. Number' (set to 2) and 'Value' (set to 9), with a 'Send' button. Below these, there are input fields for 'IP Server' (set to localhost) and 'Port' (set to 8092). At the bottom, there is a table with 4 columns: '#', 'NAME', 'DEC VALUE', and 'BINARY VALUE'. The table contains 4 rows of data.

#	NAME	DEC VALUE	BINARY VALUE
0		0	0000000000000000
1	First parameter	100	0000000000000000
2	Second parameter	9	0000000000000000
3	MOD result	1	0000000000000000
4		n	nnnnnnnnnnnnnnnn



The screenshot shows the IOCP Console interface. At the top, there are tabs for 'Files', 'Language', and 'About'. Below the tabs, there are input fields for 'Var. Number' (set to 2) and 'Value' (set to 12), with a 'Send' button. Below these, there are input fields for 'IP Server' (set to localhost) and 'Port' (set to 8092). At the bottom, there is a table with 4 columns: '#', 'NAME', 'DEC VALUE', and 'BINARY VALUE'. The table contains 4 rows of data.

#	NAME	DEC VALUE	BINARY VALUE
0		0	0000000000000000
1	First parameter	14	0000000000000000
2	Second parameter	12	0000000000000000
3	MOD result	2	0000000000000000
4		n	nnnnnnnnnnnnnnnn

DIV command

This is the documentation for the **DIV** (division) command (from “SIOC Reference Manual”):

DIV

Variable = **DIV** **Parameter1** **Parameter2**

Does the division **Parameter1** / **Parameter2**. The result is stored in the **variable**.

Parameter1 : Variable, real constant or integer constant. Number to be divided.

Parameter2 : Variable, real constant or integer constant.

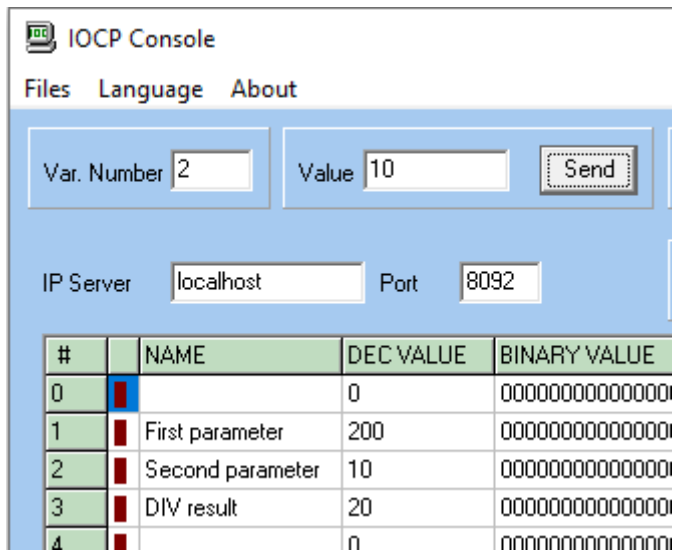
Example:

```
V0032 = DIV V0034 2
```

We can use this script to test DIV:

```
// DIV-test.txt
//
var 0001 // First parameter
var 0002 // Second parameter
{
  V0003 = DIV V0001 V0002
}
var 0003 // DIV result
```

Test e.g. with $200 / 10 = 20$:



#	NAME	DEC VALUE	BINARY VALUE
0		0	0000000000000000
1	First parameter	200	0000000000000000
2	Second parameter	10	0000000000000000
3	DIV result	20	0000000000000000
4		n	0000000000000000

Continues next page.

If we want to display landing altitude 750 without leading zeros, we have to split (or divide) 750 into 7, 5 and 0:

Position	LA1	LA2	LA3	LA4	LA5
LND ALT			7	5	0

To do this, we need both **DIV** and **MOD**:

```
// Script that simulates how to display landing altitude without leading zeros
// "9" in the comments below, illustrates the location of the digit in the display
Var 0001, name LA1 // 9NNNN
Var 0002, name LA2 // N9NNN
Var 0003, name LA3 // NN9NN
Var 0004, name LA4 // NNN9N
Var 0005, name LA5 // NNNN9

Var 0006, name LandAlt // Change this variable
{
  L0 = &LandAlt
  &LA1 = DIV L0 ,10000
  L0 = MOD L0 ,10000
  &LA2 = DIV L0 ,1000
  L0 = MOD L0 ,1000
  &LA3 = DIV L0 ,100
  L0 = MOD L0 ,100
  &LA4 = DIV L0 ,10
  L0 = MOD L0 ,10
  &LA5 = L0
}
```

The screenshot shows a software interface with a top section for variable input and a bottom section for a data table and log.

Top section:

- Var. Number: 6
- Value: 750
- Buttons: Send, Accept, Unsend
- IP Server: localhost
- Port: 8092
- Buttons: Connect, Disconnect
- Status: C
- Log: S

Bottom section (Table):

#	NAME	DEC VALUE	BINARY VALUE
0		0	00000000000000000000000000000000
1	LA1	0	00000000000000000000000000000000
2	LA2	0	00000000000000000000000000000000
3	LA3	7	00000000000000000000000000000111
4	LA4	5	00000000000000000000000000000101
5	LA5	0	00000000000000000000000000000000
6	LandAlt	750	0000000000000000000000000000101101110

Log:

```
6=750 - LandAlt
1=0 - LA1
2=0 - LA2
3=7 - LA3
4=5 - LA4
5=0 - LA5
```

This is happening:

```
L0 = 750 -> gives L0 = 750
&LA1 = DIV L0 ,10000 -> gives &LA1 = 0

L0 = MOD L0 ,10000 -> gives L0 = 750
&LA2 = DIV L0 ,1000 -> gives &LA2 = 0

L0 = MOD L0 ,1000 -> gives L0 = 750
&LA3 = DIV L0 ,100 -> gives &LA3 = 7

L0 = MOD L0 ,100 -> gives L0 = 50
&LA4 = DIV L0 ,10 -> gives &LA4 = 5

L0 = MOD L0 ,10 -> gives L0 = 0
&LA5 = L0 -> gives &LA5 = 0
```

You can use DIV-test.txt and MOD-test.txt to confirm the result above.

Note! LA1 and LA2 shows "0" above, but in the display they will be "blank". [See this page.](#)

CALL and SUBROUTINE

If the same set of commands should be performed more than once in a script, it is practical (and smart) to use a **SUBROUTINE** and **CALL** the subroutine every time you need it.

Use of SUBROUTINES makes our lives easier. We just need to check/edit the code one place in the script (for each SUBROUTINE).

This is just an example to illustrate use of **SUBROUTINE** and **CALL**, and has nothing to do with the SIM:

```
// SUB-test.txt
// This demonstrates the use of one subroutine
// Run the SUBROUTINE every time Var 0 or Var 1 is changing value
var 0000, name first_SW, value 0 // First switch to turn on
{
  CALL &set_var
}
var 0001, name second_SW // Second switch to turn on
{
  CALL &set_var
}
var 0002
var 0003
var 0004
//
// ***** START SUBROUTINE *****
var 0010, name set_var, link SUBROUTINE // Do this for first and second switch
{
  if &first_SW > 0
  {
    V0002 = 20
    V0003 = 30
    V0004 = 40
  }
  else
  {
    V0002 = -9
    V0003 = -9
    if &second_SW = 0
    {
      V0004 = -9
    }
  }
}
// ***** END SUBROUTINE *****
```

LIMIT command

This is the documentation for the **LIMIT** command (from “SIOC Reference Manual”):

LIMIT

Variable = **LIMIT** **Parameter1** **Parameter2** **Parameter3**

Increases or decreases the **variable** value with the value in **Parameter3**. If the result is higher than **Parameter2**, then **variable** = **Parameter2**. If the result is lower than **Parameter1**, then **variable** = **Parameter1**.

Parameter1 : Variable, real constant or integer constant. Lower value.

Parameter2 : Variable, real constant or integer constant. Upper value.

Parameter3 : Variable, real constant or integer constant. Increment/Decrement.

Example:

```
V9302 = LIMIT 0 60000 V0456
```

We use the LIMIT command to increase or decrease the value of a variable, and set the minimum and maximum value the variable may contain.

You can test the LIMIT command with this example:

```
// LIMIT-test.txt
//
var 0001      // Step value
{
  V0002 = LIMIT -5 10 V0001
}
var 0002      // LIMIT result
```

The “LIMIT result” (variable 2) in this example can never be lower than -5 and never higher than 10. Remember that the value of variable 2 is increasing or decreasing. Example: First I send value 2. “LIMIT result” gets value 2. Then I send value 3. “LIMIT result” gets value 5 (2+3), not value 3:

Var. Number	1	Value	2	Send
IP Server	localhost	Port	8092	
#	NAME	DEC VALUE	BINARY VALUE	
0		0	0000000000000000	
1	Step value	2	0000000000000000	
2	LIMIT result	2	0000000000000000	
3		0	0000000000000000	

Var. Number	1	Value	3	Send
IP Server	localhost	Port	8092	
#	NAME	DEC VALUE	BINARY VALUE	
0		0	0000000000000000	
1	Step value	3	0000000000000000	
2	LIMIT result	5	0000000000000000	
3		0	0000000000000000	

Try to send a value giving variable 2 a value outside the “limit range” and see what happens.

The reserved word “STATIC”

A lot of scripts contain the word “STATIC”.

Example:

```
Var 1, name servo1, STATIC
```

The use of STATIC means the variable number is kept unchanged. This has never been an issue for me, because I have never imported scripts or merged multiple scripts. But, let’s image a scenario where you have two scripts you want to merge to one (see the **[Multiple Scripts]** button in SIOC). SIOC will renumber the variables (starting with 0001, variable 0000 will never change number) and ensure that the result contains only unique variable numbers (and unique names). To avoid the renumbering, you need to use STATIC. If this is relevant for you, I recommend you to read more about merging scripts in SIOC reference documentations, or test it with two (or more) small test-scripts you merge to one.

Variables – classification

This table contains examples on how to classify variables, and you don't have to classify them if you don't want to.

This is from "IOCards Step by step, Part 2: SIOC language" by Claude Kieffer:

	From	To
Auto Pilot	0000	0299
Landing Gear	0300	0399
Flaps	0400	0499
Plane	0500	0599
Lights	0600	0699
Sound	0700	0799
COMMs, NAVs, ADFs, Transponder	0800	1099
Switches Audio	1100	1199
Visual and aural warnings	1200	1399
Engines	1400	1499
Fuel	1500	1599
Propellers	1600	1699
Overhead	1700	1999
Orders to USB Keys	2000	2099
Servos and Stepper variables	3000	3200

You can make your own classifications. If you are using existing scripts, look at them and make notes of which variables (ranges) already in use, and keep away from those numbers when you define your own variables.

Creating script file – alternative 1

To create a script file, you have several alternatives. One alternative is to use existing scripts from: https://www.opencockpits.com/catalog/info/information.php?info_id=44&language=en

Here is the top of the list:

GUIDE TO BUILD A FLIGHTDECK		
SIOC SCRIPTS		
Click to go to the scripts section		
B-737 SCRIPTS	A-320 SCRIPTS	GENERAL AVIATION SCRIPTS
B-737		
SECTION	DESCRIPTION	LINK
Overhead	Overhead B-737 Lights. All lights on the default 737 in FS (Landing, Taxi, Strobe, Beacon, Wing, Logo and Nav).	Download
Overhead	APU EGT gauge. Programmed for to use with Project Magenta. The script is very simple, because it simply take the value from Magenta offset and send it divided by 10 to the servo.	Download
Overhead	Overhead PMSYSTEMS. Complete logics for the B-737 Overhead using the PMSYSTEMS from Project Magenta. Write by one of our clients, we are sorry to can't tell who is, because we don't know who is. If the uthor identifies their script, please tell us on info@opencockpits.com , and we proceed to change the author's name.	Download
Overhead	Overhead Gauges PMSYSTEMS. Script with the configuration for Overhead gauges with PMSYSTEMS (Project Magenta software).	Download
Overhead	Overhead Boeing 737 full for ProSim, Ifly and Project Magenta (last revision July 2013). Full scripts for Project Magenta, Ifly and PROSIM 737 software. Overhead's I/O mapping included, ProSim's XML settings impor file are also included.	Download

See “Section/Description”, and download the ZIP file you want to use.

This is the content of Opencockpits_Overhead_files_for_ProSim_Project_Magenta_Ifly_07_2013.zip:

Archivos para ProSim		
Archivos personalizados para Overhead		
<input type="checkbox"/> Instrucciones ProSim.txt	Date modified: 17.01.2013 10:17 Size: 499 bytes → 306 bytes	Type: Text Document
<input type="checkbox"/> Opencockpits_Overhead_script_2012_final_ProSim.txt	Date modified: 02.08.2013 09:33 Size: 31,1 KB → 5,61 KB	Type: Text Document
<input type="checkbox"/> Opencockpits_Overhead_V2_ProSim_IOMapping_2013.xml	Date modified: 07.02.2013 17:16 Size: 28,1 KB → 3,08 KB	Type: XML File
<input type="checkbox"/> Script B737 ProSim OVH FWD only one servo card.txt	Date modified: 21.02.2019 10:20 Size: 31,3 KB → 5,79 KB	Type: Text Document
<input type="checkbox"/> sioc_overhead_prosim.ssi	Date modified: 02.08.2013 09:34 Size: 196 KB → 15,4 KB	Type: SSI File

This fragment is from Opencockpits_Overhead_script_2012_final_ProSim.txt:

```
Var 1010, name VHF_1, static, Link IOCARD_SW, Device 5, Input 40 // VHF NAV: Both on 1
Var 1011, name VHF_2, static, Link IOCARD_SW, Device 5, Input 39 // VHF NAV: Both on 2
Var 1012, name IRS_L, static, Link IOCARD_SW, Device 5, Input 42 // IRS: Both on L
Var 1013, name IRS_R, static, Link IOCARD_SW, Device 5, Input 41 // IRS: Both on R
```

If you start with a script from OC, you may have different approaches:

- You can buy the cards the script is made for, and wire the hardware according to the script
- You can copy rows you need and use/modify them in your script

Remember! IDX numbers (“Device”) in scripts must match corresponding cards in your SIOC.ini. You can renumber an IDX number in scripts if you need it.

I assume you want to use the variable numbers already in the script, and only changing the numbers if you have used the same numbers in another context.

The ZIP file contains a “XML mapping file” to be used with ProSim. The XML file maps SIOC variables with functionality in ProSim. Example:

```
<mapping connection="VHF NAV Both on 1">
  <iocp serial="0" port="1010"/>
</mapping>
<mapping connection="VHF NAV Both on 2">
  <iocp serial="0" port="1011"/>
</mapping>
```

The fragment above shows how the two first rows in the script (on previous page) are mapped in ProSim. If you need to change a variable number in the script, you have to remember to change the same variable number in the XML file too. If you forget this, the “Import configuration” functionality in ProSim will only make problems for you.

I have never used ProSim “Import configuration” functionality, and therefore have no experience related to this, but it looks quite useful. **Remember:** You can always configure functionality in ProSim manually.

To get most out of existing scripts, you must connect inputs and outputs to the corresponding connector on the cards. By other words: You cannot connect wires to cards randomly.

For example, for the variable below, you need to locate connector 40 (on the input card indirectly connected to USB Expansion card with IDX 5) for the function “VHF NAV: Both on 1”:

```
Var 1010, name VHF_1, static, Link IOCARD_SW, Device 5, Input 40 // VHF NAV: Both on 1
```

If you don’t do this, you have to renumber values for Input and Output, and maybe Device numbers too.

"Alternative 1" is very relevant (or “mandatory”) with IDC devices, but also great help in other cases as well; You can pick and choose code fragments to use in your script.

Creating script file – alternative 2

When I created my script (ProSim_B737-800.txt), I used a method I saw on YouTube. I did this:

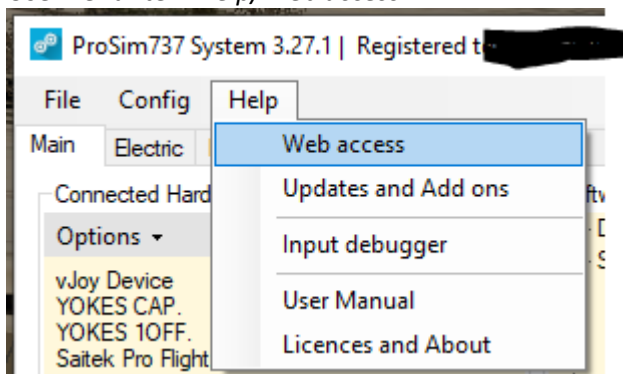
1. I made an Excel spreadsheet where I wrote all the variable numbers I needed:

Variable	Description	Device (IDX)	Connection	Comments
...				
2028	ASA A/P reset Capt Pushed	10	23	
...				

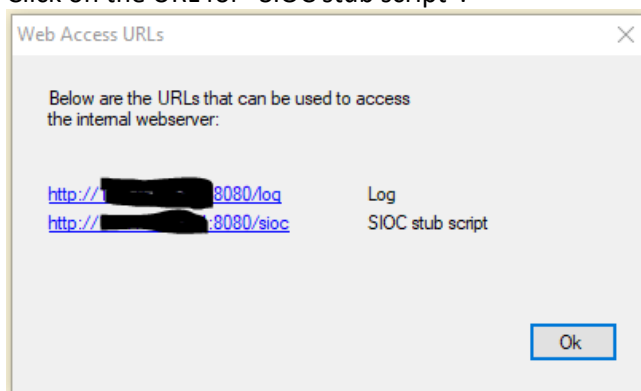
Later, when the information was available, I added IDX and connection to the spreadsheet. You don't need to do that. The script file is documentation good enough, but always make backups and keep them in a safe place 😊

2. I added all my variables to ProSim. See the [ProSim and SIOC](#) section how to configure SIOC variables in ProSim.
3. After I had added all the variables to ProSim, I used this functionality:

Use menu-item *Help/Web access*:



Click on the URL for “SIOC stub script”:



When you click the URL for “SIOC stub script”, a browser will show you all the configurations you have done in ProSim based on SIOC variables. Copy and paste the text into Notepad. See next page.

This is an example based on my choices of variable numbers. All the variables you see in the browser (or in Notepad after cut/paste) are configured in ProSim.

```
// Generated SIOC config stub for ProSim
// Make sure you configure the correct hardware inputs/outputs

Var 2010, Link IOCARD_SW, Input 0 // Fire warning 2 Pushed
Var 2011, Link IOCARD_SW, Input 0 // Master caution 2 Pushed
Var 2012, Link IOCARD_SW, Input 0 // Recall 2 Pushed
Var 2014, Link IOCARD_SW, Input 0 // Autobrake RTO
Var 2016, Link IOCARD_SW, Input 0 // Autobrake 1
Var 2017, Link IOCARD_SW, Input 0 // Autobrake 2
Var 2018, Link IOCARD_SW, Input 0 // Autobrake 3
Var 2019, Link IOCARD_SW, Input 0 // Autobrake Max
```

As you can see, the “stub” does not include **Device**, and all **Input** and **Output** numbers are 0 (zero). You can use Notepad to add **Device** and apply numbers to **Input** and **Output**. You have to use SIOC Monitor to find the correct information.

With “alternative 2” you can connect all switches and annunciators randomly to the input and output cards.

Use “SIOC stub script” only once, because the “stub” never gives us Device or Input/Output connector-numbers. Add as many variables in ProSim as possible before generating the “stub”. If you later need more variables, I recommend “alternative 3”. See next page.

Creating script file – alternative 3

Here you write the SIOC script from scratch. You know how to create a variable, but you can always cut and paste from existing scripts if you are unsure, or want to save some time.

Use this cycle for each variable you create:

1. Connect (wire) hardware (switch/annunciator/etc.) to a card
2. Create the variable defining the hardware
3. Save the script file
4. Reload SIOC
5. Test with [SIOC Monitor], and [IOCPConsole] if you have added some “programing”
6. Configure the SIOC variable in ProSim
7. Test from cockpit

This will give you excellent control. Any problems may be corrected before continuing.

Of course, you can connect several pieces of hardware before doing step 2 to 7.

You can always mix “script alternative” 1, 2 and 3. Anyway, it won't take long before you find a method that works for you.

SIOC documentation

I recommend downloading and studying scripts from OC or other sources, to see how they are written. To learn all about the SIOC script, please search the web.

You can try this link:

<https://www.opencockpits.com/index.php/en/download/category/english-3>

And this:

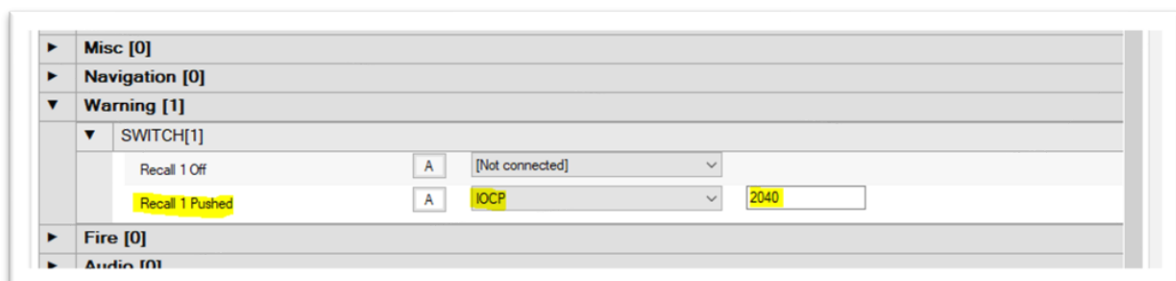
https://www.opencockpits.com/catalog/info/information.php?info_id=49&language=en

ProSim and SIOC

ProSim and SIOC works together and detect changes in the cockpit:

- If I turn a switch in the cockpit, SIOC sends a signal to ProSim using the variable I have defined for the switch, and ProSim will execute whatever shall happen in the aircraft based on my action
- If a situation occurs in the aircraft turning on a warning light, ProSim will send a signal to SIOC and the warning light associated with the variable I have defined for the annunciator will light up
- We don't have to know what kind of signal ProSim and SIOC are using internally, just assign SIOC variables to different functions in ProSim
- ProSim and SIOC exchange status several times per second

This is an example where **Recall** is pushed on the captain's side.



The Input card I have used is indirectly connected to a USB Expansion card with IDX 11 (Device 11 below), and I have connected the **Recall** button to connector 29 (Input 29 below). I have used variable number 2040:

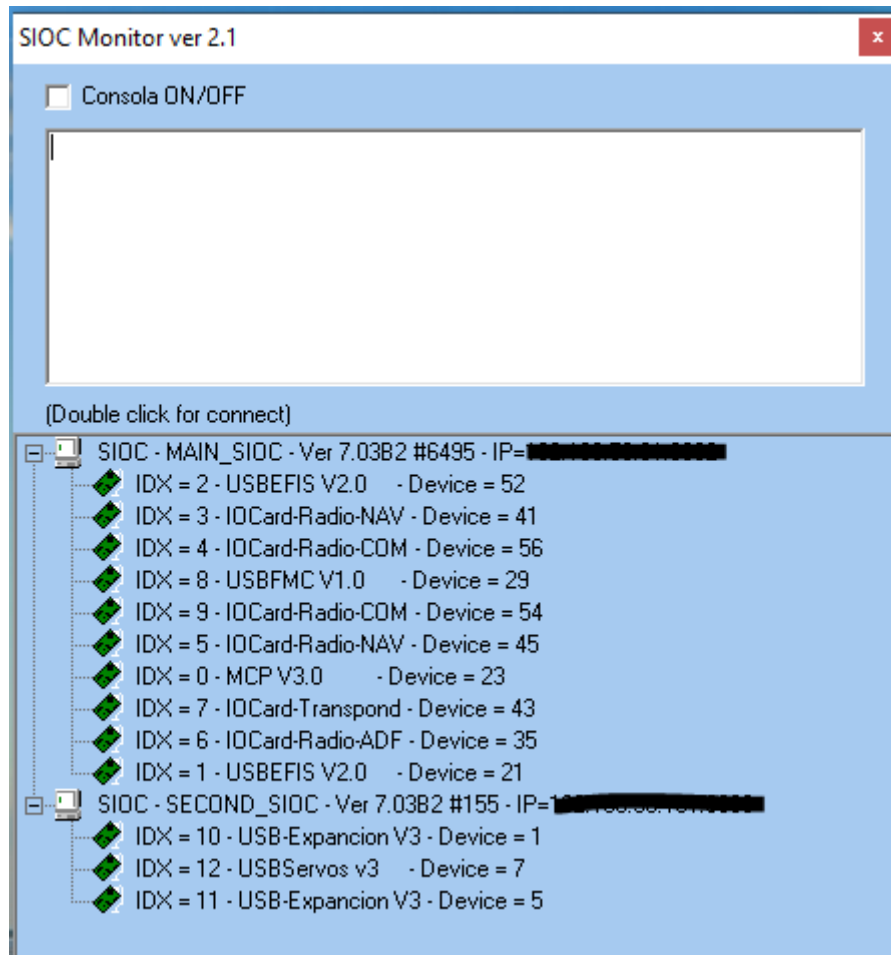
```
Var 2040, Link IOCARD_SW, Device 11, Input 29 // Recall 1 Pushed
```

You don't need to remember information about Expansion card or connection used on a card when you configure hardware in ProSim. All you need to know is this (example):

- Variable 2040 is defined for "Recall 1 Pushed"

SIOC Monitor

In SIOC, there is a button named [**SIOC Monitor**]. Click the button to see information for all USB devices and USB cards connected. If you have several computers running SIOC, you will see all devices and cards available in the network. This is an example from my SIM, with SIOC running on two computers:



We need SIOC Monitor when adding functionality to the SIM. For example, when adding new switches or annunciators.

I double-click the “**USB Expansion card**” the input- or output card I’m working with is indirectly connected to, for example “IDX = 10 – USB-Expansion V3 – Device = 1”.

If you have only one expansion card, it’s easy; Double-click the card.

If you have two or more expansion cards, you have to remember which expansion card is used for the input- or output card you are working with. I use one expansion card for the MIP, and one expansion card for the Overhead. You can specify this as comments in SIOC.ini, but comments in SIOC.ini are not displayed in the monitor, so you have to peek inside SIOC.ini if you have forgotten which expansion card to choose.

See next page for practical use of SIOC Monitor.

TCAS switch and IDENT button

As mention before, the **TCAS switch** and the **IDENT button** on the ATC/Transponder device, did not work with the “IOCMODULES-method” (at least not for me). I had to define in SIOC and configure in ProSim to make them work.

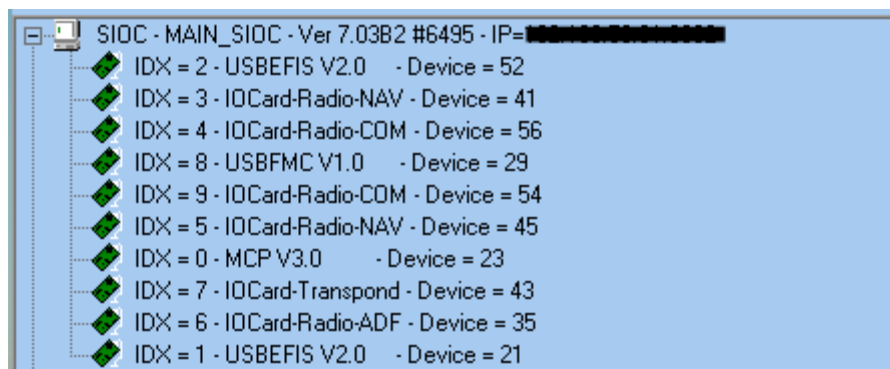
This is what I added to C:\locards\SIOC\SIOC_Scripts\MCP_V3_prosim.txt (FlightSim1):

```
Var 4406, name I_R_STB, Link IOCARD_SW, Device 7, Input 0 // STBY
Var 4408, name I_R_ALT, Link IOCARD_SW, Device 7, Input 1 // ALT RPTG OFF
Var 4410, name I_R_XPN, Link IOCARD_SW, Device 7, Input 2 // XPNDR
Var 4412, name I_R_TAO, Link IOCARD_SW, Device 7, Input 3 // TA ONLY
Var 4414, name I_R_TAR, Link IOCARD_SW, Device 7, Input 4 // TA/RA
Var 4416, name I_R_TEST, Link IOCARD_SW, Device 7, Input 12 // IDENT
```

“Device 7” above is related to “IDX = 7” in the screenshot below (this can be confusing because the screenshot below is also using the word “Device”).

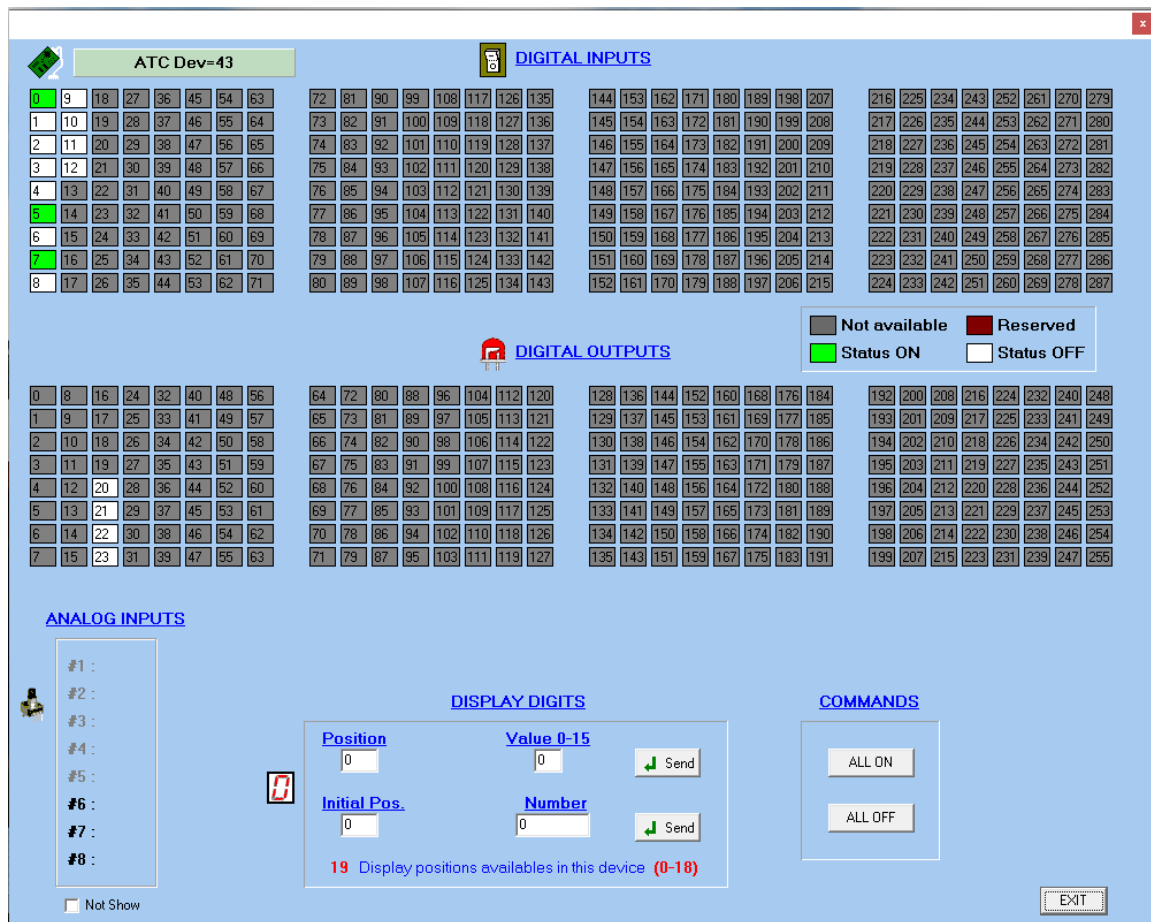
First you locate the correct device. Take note of the IDX number: “IDX = 7 - IOCard-Transpond”

Use “Device 7” in the script above. **Note!** The IDX number is likely something else in your SIM.



Then you double-click the device (in my case: “IDX = 7 – IOCard-Transponder – Device = 43”) to display details for the card inside the ATC/Transponder device. Continues next page.

The upper/left corner on the screenshot below, shows the connections available (0 - 12) on the input card inside the ATC/Transponder device.



As you can see above, the connection 0 (Input 0 in the script below) has green light. The switch is in position “STBY” on the ATC/Transponder device. If I turn the switch to “ALT RPTG OFF”, connector 0 turns white, and connector 1 (Input 1 in the script below) turns green. This way we can find all the connectors in use, and we can define all the variables, later used in ProSim, like this:

```
Var 4406, name I_R_STB, Link IOCARD_SW, Device 7, Input 0 // STBY
Var 4408, name I_R_ALT, Link IOCARD_SW, Device 7, Input 1 // ALT RPTG OFF
Var 4410, name I_R_XPN, Link IOCARD_SW, Device 7, Input 2 // XPNDR
Var 4412, name I_R_TAO, Link IOCARD_SW, Device 7, Input 3 // TA ONLY
Var 4414, name I_R_TAR, Link IOCARD_SW, Device 7, Input 4 // TA/RA
```

When I press the IDENT button, connector 12 turns green.

```
Var 4416, name I_R_TEST, Link IOCARD_SW, Device 7, Input 12 // IDENT
```

Don't forget to configure ProSim for use of variables 4406 to 4416 (you can use other variable numbers).

Note! If you want to use “SIOC Direct”, you have to remove all connections to variable 4406-4416 (in my case) from ProSim, if you previously have configured them.

“IOCMODULES-METHOD” CHECKLIST

- Configure [ProSim and MCP](#)
- Connect P&P devices to the computer
- Install SIOC
- Download MCP_V3_prosim.txt
- Edit SIOC.ini
 - Use MCP_V3_prosim.txt as Configuration File
 - Configure all your P&P devices (remember specific use of IDX 0, 1 and 2)
- Restart or Reload SIOC and check the result in the Devices-view, edit SIOC.ini if necessary, and repeat this until everything is okay (no “*”)
- Install IOCMODULES
- Edit IOCMODULES.ini
- Add IOCP Server(s) in ProSim Configuration
- Add SIOC variables/configurations in ProSim for the ATC/Transponder device (if necessary)

This “Every day startup sequence” works fine for me when using the “IOCMODULES-METHOD”:

On FlightSim1:

1. SIOC
2. IOCMODULES
3. ProSimDisplay
4. ProSimUtils, ActiveSky and CFY Console (CockpitForYou Throttle Quadrant)
5. FS
6. ProSim
7. SimSounds and vPilot

On FlightSim2:

1. SIOC
2. ProSimDisplay, ProSimCDU and ProSimAudio

I start FlightSim2 before FlightSim1.

“SIOC Direct-method” checklist

- Configure [ProSim and MCP](#)
- Connect P&P devices to the computer
- Install SIOC
- Edit SIOC.ini
 - No use of Configuration File
 - Configure all your P&P devices
- Restart or Reload SIOC and check the result in the Devices-view, edit SIOC.ini if necessary, and repeat this until everything is okay (no “*”)
- Add IOCP Server(s) in ProSim Configuration
- Configure SIOC Direct in ProSim Configuration
- Configure “8.33 Voice Channel Spacing” in ProSim Instructor Station
- Remove SIOC variables/configurations in ProSim for the ATC/Transponder device (if you have made them previously)

This “Every day startup sequence” works fine for me when using the “SIOC Direct-method”:

On FlightSim1:

1. SIOC
2. ProSimDisplay
3. ProSimUtils, ActiveSky and CFY Console (CockpitForYou Throttle Quadrant)
4. FS
5. ProSim
6. SimSounds and vPilot

On FlightSim2:

1. SIOC
2. ProSimDisplay, ProSimCDU and ProSimAudio

I start FlightSim2 before FlightSim1.

“Combination-method” checklist

- Configure [ProSim and MCP](#)
- Connect P&P devices to the computer
- Install SIOC
- Edit SIOC.ini
 - Use MCP_V3_prosim.txt as Configuration File (for MCP, EFIS and CDU/FMC)
 - Configure all your P&P devices (remember specific use of IDX 0, 1 and 2)
- Restart or Reload SIOC and check the result in the Devices-view, edit SIOC.ini if necessary, and repeat this until everything is okay (no “*”)
- Add IOCP Server(s) in ProSim Configuration
- Configure SIOC Direct in ProSim for specific devices (by your choice)
- Instruct SIOC Direct to ignore devices covered in MCP_V3_prosim.txt
- Configure “8.33 Voice Channel Spacing” in ProSim Instructor Station
- Remove SIOC variables/configurations in ProSim for the ATC/Transponder device (if you have made them previously)

This “Every day startup sequence” works fine for me when using the “Combination-method”:

On FlightSim1:

1. SIOC
2. ProSimDisplay
3. ProSimUtils, ActiveSky and CFY Console (CockpitForYou Throttle Quadrant)
4. FS
5. ProSim
6. SimSounds and vPilot

On FlightSim2:

1. SIOC
2. ProSimDisplay, ProSimCDU and ProSimAudio

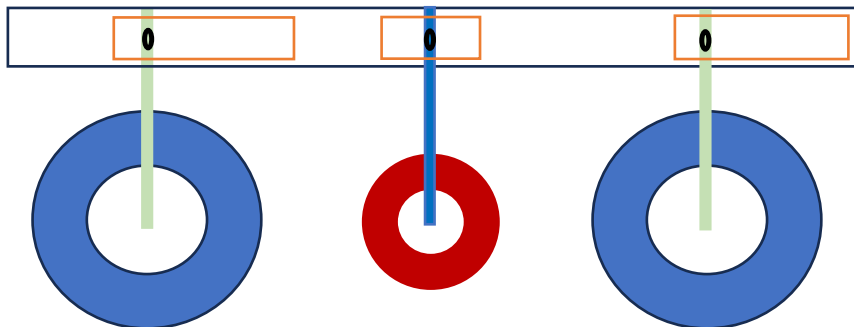
I start FlightSim2 before FlightSim1.

My Engine Start switch solution

The illustration below shows the use of one servo to get auto-return functionality for both Engine Start switches.

Explanation:

- The red circle represents the servo-motor
- The solid blue vertical rectangle, fixed to the servo-motor, represents the servo-arm
- The two blue circles represent the rotary switches in the Engine Start panel
- The two solid green vertical rectangles represent 2mm steel sticks (rods), fixed to each switch. Each stick follows the movement of the rotary switch it's connected to
- The white/black horizontal box represents the "bridge" (made of Acrylic Plexiglass) connecting the servo to the rotary switches
- The three white/orange horizontal boxes (inside the bridge) represent gaps in the bridge
- The black "dots" represent small screws fixed to the servo-arm and both sticks



In the illustration, both Engine start switches are in OFF position. The servo-arm and the bridge are in neutral position. The status is "OFF/neutral".

When turning one of the Engine start switches to GRD (turning to left), the green stick connected to the switch, pushes the bridge to the left (using the screw fixed to the stick). The servo does nothing.

When N2 is 56%, the servo-arm pushes the bridge to the right (back to neutral) and pull the Engine start switch back to OFF. After one second, the servo-arm goes back to neutral position. The system is back to status "OFF/neutral", and we can start the next engine.

The gaps in the bridge makes it possible to move both Engine start switches to all possible positions.

Adjusting the gaps and stick positions can be a challenge, but the solution is relatively easy to build, and it's cheap 😊

You have to use SIOC Monitor to find the values you need for neutral position, and how far right the servo-arm has to move, to bring the Engine start switch back to OFF. The servo-motor is strong, so let the bridge move without obstacles until you get a feeling of the distance the bridge has to move, and remember: The bridge cannot move outside the "Engine Start panel box". Please see the SIOC [script](#) programmed for this solution, if you consider the same solution in your SIM.

I'm very happy with the solution. It has never failed me. Of course, it's not close to the real thing, but for me the functionality is close enough.

Landing altitude without leading zeros

Use SIOC scripting if you want to avoid leading zeros in displays. I'm using the landing altitude as example. To understand what is going on, I show you the solution with leading zeros also.

Alternative 1, with leading zeros:

```
Var 1320, name LA, Link IOCARD_DISPLAY, Device 5, Digit 69, Numbers 5 // Land Alt
```

This illustrates the display with leading zero and landing altitude 750 feet:

Position	1	2	3	4	5
Digit	73	72	71	70	69
LND ALT	0	0	7	5	0

Alternative 1 begins the value of 750 in position 5 and the number is left-adjusted, as illustrated above. This alternative always displays five digits (because of "Numbers 5").

Alternative 2, without leading zeros:

```
Var 1320, name LA1, Link IOCARD_DISPLAY, Device 5, Digit 69, Numbers 1 // Land Alt 5th pos
Var 1321, name LA2, Link IOCARD_DISPLAY, Device 5, Digit 70, Numbers 1 // Land Alt 4th pos
Var 1322, name LA3, Link IOCARD_DISPLAY, Device 5, Digit 71, Numbers 1 // Land Alt 3rd pos
Var 1323, name LA4, Link IOCARD_DISPLAY, Device 5, Digit 72, Numbers 1 // Land Alt 2nd pos
Var 1324, name LA5, Link IOCARD_DISPLAY, Device 5, Digit 73, Numbers 1 // Land Alt 1st pos
```

Alternative 2 puts only one number (because of "Numbers 1") in each position, and this will be the result:

Position	1	2	3	4	5
Digit	73	72	71	70	69
LND ALT			7	5	0

Alternative 2 gives us the possibility to blank positions one by one, while alternative 1 only blanks all positions.

The SIOC script for alternative 2 is on the next page.

SIOC script for landing altitude without leading zeros

This script fragment is from [SIOC scripts](#).

```
Var 1303, name LandAlt, static, Value 100 // 100 is default value
{
    CALL &SetLandAlt
}

Var 1304, name SetLandAlt, static, Link SUBROUTINE
{
    IF &VoltDCStandby = 0    // Blank all
    {
        &LA1 = -999999
        &LA2 = -999999
        &LA3 = -999999
        &LA4 = -999999
        &LA5 = -999999
    }
    ELSE                      // Display land altitude
    {
        L0 = &LandAlt
        &LA1 = DIV L0 ,10000
        L0 = MOD L0 ,10000
        &LA2 = DIV L0 ,1000
        L0 = MOD L0 ,1000
        &LA3 = DIV L0 ,100
        L0 = MOD L0 ,100
        &LA4 = DIV L0 ,10
        L0 = MOD L0 ,10
        &LA5 = L0
    }
}

// Land Alt 5fh 7-segment display
Var 1320, name LA1, static, Link IOCARD_DISPLAY, Device 5, Digit 69, Numbers 1
// Land Alt 4th 7-segment display
Var 1321, name LA2, static, Link IOCARD_DISPLAY, Device 5, Digit 70, Numbers 1
// Land Alt 3rd 7-segment display
Var 1322, name LA3, static, Link IOCARD_DISPLAY, Device 5, Digit 71, Numbers 1
// Land Alt 2nd 7-segment display
Var 1323, name LA4, static, Link IOCARD_DISPLAY, Device 5, Digit 72, Numbers 1
// Land Alt 1st 7-segment display
Var 1324, name LA5, static, Link IOCARD_DISPLAY, Device 5, Digit 73, Numbers 1

Var 1330, name VoltDCStandby, static // AUTO: FLT ALT + LAND ALT displays activation
{
    CALL &SetLandAlt
    // CALL &SetFlightAlt    // If you want the same for Flight Alt. The code is
                            // very similar to Land Alt.
}

Var 1332, static, Link IOCARD_ENCODER, Device 5, Input 128, Aceleration 3, Type 2
// Encoder input for LAND ALT
{
    L0 = 50 * V1332
    &LandAlt = LIMIT 0 ,14000 ,L0
}
```

Fire Engines module USB Plug&play

Last Christmas, the SIM got a present from Santa:

“Fire Engines module USB Plug&play”

With my current configuration (the “Combination-method”), I only had to:

1. Connect the device to FlightSim1 (USB connection)
2. Connect the device to power (5V, 1Amp)
3. Add the device in SIOC.ini on FlightSim1
4. Restart SIOC on FlightSim1

That's it! The module works perfect!

How to control the backlight

As mentioned earlier, the backlight is always "ON" on P&P devices. They use USB connections, and as long as the USB cable is connected to the computer, and the computer is on, the backlight is on.

The first thing I do when I get into the cockpit is to turn the power on, as other pilots do, so this is no big deal for me. But, if it is time to replace broken light bulbs, we may consider doing something about the situation.

In my SIM, several light bulbs on the ADF device no longer worked, and one bulb flashed all the time. It was time for some maintenance:

1. Remove the device from the cockpit and take it to the workshop
2. Take the device apart
3. Use a soldering iron to remove all light bulbs in the device
4. Glue Pre-Soldered Micro LED everywhere a bulb was removed
5. Put the device together
6. Add functionality to your SIOC script
7. Install the device in the cockpit

Let me elaborate steps 4 to 7:

Step 4 - Glue Pre-Soldered Micro LED everywhere a bulb was removed

I bought LEDs from ebay. They come in different sizes and voltage. I chose 12V because I had a 12V power supply available, but we can use e.g. 5V. I chose warm-white, and type 0805 for the ADF, but other types (dimensions) are available:

	Approximately size in mm		
Type	Length	Width	Height
0402	1	0.5	0.5
0603	1.6	0.8	0.4
0805	2	1.25	1.1
1206	3.2	1.6	1.1

Note! Specified sizes may vary from manufacturer to manufacturer.

Check the bulb size before ordering LEDs. Different devices may use different bulb sizes.

We can glue the LEDs to the circuit board (no soldering necessary).

Step 5 - Put the device together

Reassemble the device.

I connected all the LED wires (in parallel) to the unit below. The unit needs minimum two connectors.



Collect all red wires (positive) to one connector, and all black wires (ground) to the other connector. Glue the unit to the bottom of the device.

In danger of destroying the circuit board, I did not make holes to bring the wires out of the device. I led the wires on the outside. The wires are thin, so it works okay. Or, if you are braver than me, make (if possible) holes in the circuit boards.

Step 6 - Add functionality to your SIOC script

To control the backlight, we use:

- Relay
- Outputs connection card
- SIOC script

Define one variable for each relay you have. Example:

```
// Relay for pedestal devices (Only ADF so far ...)  
Var 5068, name relay_Pedestal, Link IOCARD_OUT, Device 10, Output 87
```

We *can* insert a new line in the code used for VoltDCStandby (see [Landing altitude](#)):

```
Var 1330, name VoltDCStandby, static  
{  
  CALL &SetLandAlt  
  CALL &CtrlBackLight  
}
```

And we create the subroutine, which we can re-use if new relays are installed/defined later:

```
Var 9999, name CtrlBackLight, static, Link SUBROUTINE  
{  
  IF &VoltDCStandby > 0  
  {  
    &relay_Pedestal = 1  
  }  
  ELSE  
  {  
    &relay_Pedestal = 0  
  }  
}
```

A subroutine is not necessary in cases like this, but using one is not wrong. Either way, make the code the way you want, and don't forget to reload SIOC after editing the script.

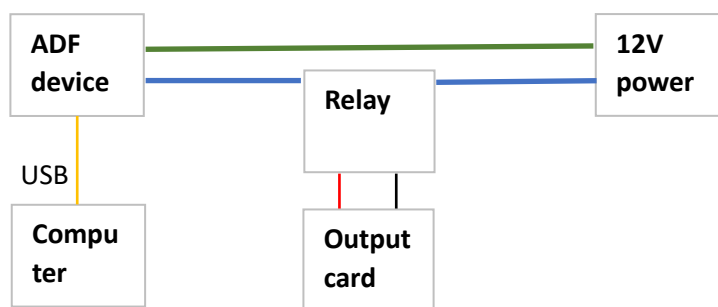
Step 7 - Install the device in the cockpit

Based on the example above, I have to connect the 5V positive wire from the relay to connector 87 on the Output-card, which is indirectly connected to USB Expansion card with IDX 10.

This is the relay I'm using:



An illustration of the configuration:



- The **green** wire represents 12V ground
- The two **blue** wires represent 12V positive
- The **red** (positive) and **black** (ground) wire are 5V, both connected to the Output-card

The Output-card is connected to a Master-card. The Master-card is connected to a USB Expansion-card.

The illustration does not include a dimmer, but the solution does not exclude the opportunity to use one.

Once you have edited the SIOC script and connected the device as illustrated above, it is time to test and see what happens when you turn the battery switch ON and OFF.

How to avoid light pollution

Most of my devices suffer from "light pollution"; They "leak" light. To reduce it, we can put them into boxes. I have considered different materials such as:

- 0.7 mm light metal
- 0.5 mm aluminum

The material must be thin and preferably easy to work with. The material does not have to be strong, as it does not get any load. I ended up with 300 grams of black cardboard which is:

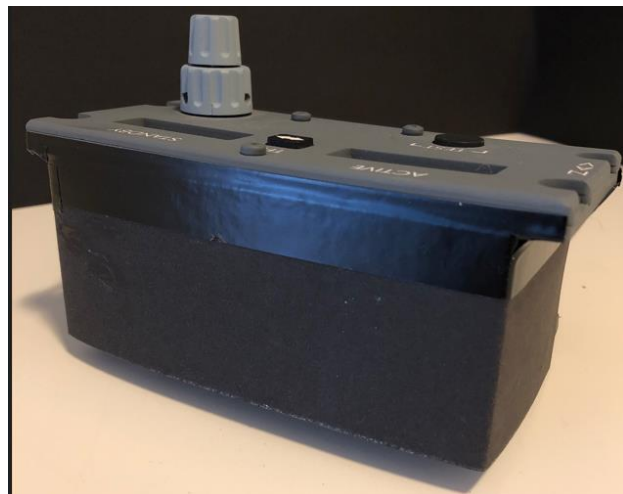
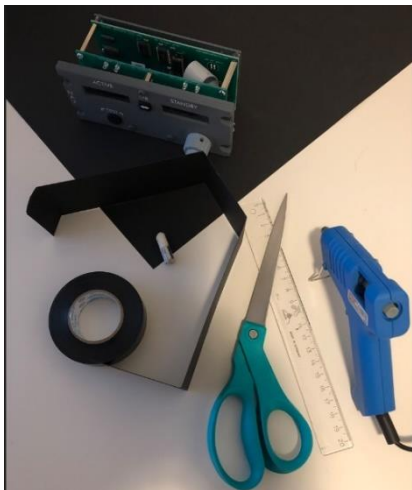
- easy to work with
- thin
- stiff
- cheap

I bought the cardboard in a book shop. I had to consider the risk of fire, but none of the devices emit a lot of heat. I kept the light on for a long time and checked the heat before installing the device in the pedestal.

I used 18mm black electrical insulation tape on top, and the reduction of light pollution was significant.

If you want to try, I used:

- black cardboard
- black PVC Electrical insulation tape (gray is a good alternative)
- **white marker**
- scissor
- ruler
- glue



Cut the box as one piece and remember an extra piece/overlap to glue the box together. Use the ruler to make sharp corners. Wrap the box around the device and glue the overlap.

The tape on top prevents light from coming out, and at the same time holds the box and device together.

After modification, the device still fits in "OC pedestal bay 737 V2".