PSXseecon

Manual

Version 3.20

Aug 15, 2019

Nico Kaan

1 Introduction

PSXseecon connects the Aerowinx Precision Simulator version 10 (PSX) with the SIOC software of Opencockpits. It acts as a bridge between these two worlds, "talking" the Aerowinx Network Protocol over TCP/IP to PSX and "talking" the IOCP protocol over TCP/IP to the SIOC Server.

Although both worlds offer a TCP/IP based interface, they speak different protocols and they are both servers. PSXseecon acts as a client program to PSX and as a client program to SIOC and performs protocol translation.

2220 predefined SIOC Variables representing 99% of the functionality of the PSX interface are available. These variables give access to almost all states and controls of the PSX. You can receive information about state changes in the 744 and you can set Buttons, Switches, levers, Throttles, Dials, what have you, in the 744. Everything you need to build a cockpit.

For ease of SIOC coding PSXseecon provides extra functionality, such as led variables with built in support for bulb-failure, power failure and MD&T test; rotary switch terminal variables; OnOffOn switch terminal variables; direct support for Rotary Encoders; direct support for the Opencockpits USB Key-card (in the CDU); etc.

To support debugging, PSXseecon provides facilities for monitoring the messages that flow between PSX and PSXseecon and between SIOC and PSXseecon.

Examples for SIOC scripts for PSX can be found in my PSXcockpit folder.

Note that PSXseecon detects the predefined SIOC variables for PSX in SIOC scripts automatically!

The first version of PSXseecon was released September 3, 2014.

1.1 Technical information

PSXseecon is a 64-bit Windows application, written in C++, for Windows 7 or later.

1.2 System Requirements

You need a 64-bit Windows version.
In order to run it, you need Microsoft Visual C++ Redistributable for Visual Studio **2015**.
The standalone installers for x86 and x64 can be downloaded here at Microsoft.

1.3 Forum

For Questions & Answers and Discussions use the Aerowinx forum.

2 Installing and configuring of PSXseecon

2.1 Installing PSXseecon

2.1.1 Downloading and Installation

Download the PSXseecon_vx.y.zip from my website. Unzip. You may install PSXseecon at any Windows PC (Vista or higher) in your local network.

Installation is easy, there is no install procedure, no registry entries, no program groups. Just unzip the archive wherever you want. In order to prevent UAC problems installation outside C:\Program Files is recommended.

PSXseecon.exe is the application.

Documentation is provided by means of a Manual, Release Notes and SIOC variable definitions (explained in this Manual in Chapter 2.2.5). Aerowinx MCDU Large and Small fonts are provided in case you want to use PSXseecon as a CDU screen (section 3.19).

The PSXcockpit folder contains SIOC scripts for building a PSX cockpit.

	An important file in the archive is the EULA.pdf. This End-User License Agreement is a legally enforceable contract between you and the Author. **By downloading and installing the PSXseecon package you agree to be bound by the terms of this EULA**.

Note: You may install and run the software more than once at the same PC. Each PSX-seecon with its own PSXseecon.cfg containing parameters pointing to different SIOC instances (at the same or at other PC's), as long as each SIOC runs a disjoint set of SIOC PSX **control** variables (information variables do not matter).

2.1.2 Upgrading PSXseecon to a new version

Just install the new version by unzipping and overwriting files. The new version will continue to run with your settings in PSXseecon.cfg (if in use).

2.1.3 Uninstalling PSXseecon

Removing PSXseecon from your PC can be done just by deleting the files. There are no registry entries.

2.2 Configuring PSXseecon

Configuration information has to be put in a plain text file named **PSXseecon.cfg** in the **same** folder as PSXseecon. Note that this file is not in the software package, it will be created the first time PSXseecon runs (if not already there).

Make sure you are using the correct parameter names. This can be checked in the log file Log.txt (see section 4.4).

2.2.1 Position of PSXseecon on your screen

PSXseecon will remember the last position of PSXseecon on your screen by writing (and reading) the following parameters to your PSXseecon.cfg.

MAIN_WINDOW_X=number
MAIN_WINDOW_Y=number

If PSXseecon.cfg does not exist, it will create one. The values of these parameters can be changed manually, but it is easier just to place the main window where you want it.

2.2.2 Connection parameters

Always run PSX as **main server** first *(in the network tab*)!

What you have to do depends on the computers where PSX, SIOC and PSXseecon run.

Scenario A: all on one computer
In this scenario no configuration work is needed. PSXseecon will find PSX and SIOC and it will find the SIOC script(-s) that you are running automatically.

Scenario B: SIOC at a different computer then PSXseecon
In this scenario you have to give PSXseecon information about where to find SIOC and where to find the script(-s) that you are running. The following parameters have to be defined in PSXseecon.cfg (with your data of course):

IOCP_HOSTADDRESS=192.168.1.70
IOCP_port=8092
CONFIG_FILE=\\ADAM\SIOC\cockpit744.lst

The IOCP_HOSTADDRESS parameter is the IP-address of the PC running SIOC.

The IOCP_port is the Port used by the SIOC Server. You can find that information in the <IOCP> SERVER section of the main window of the SIOC.exe program.

The CONFIG_FILE parameter specifies the very same file as is specified in sioc.ini. Depending on how you use SIOC it can be either a .ssi file or a .lst file. Note that you must specify a full path name (contrary to sioc.ini). In this example ADAM is the logical name of the PC running SIOC and cockpit744.lst defines all the SIOC scripts for a 744 cockpit.

Scenario C: PSXseecon at another comuter then PSX
In this scenario you have to give PSXseecon the IP-address of the PSX Server.

PSX_HOSTADDRESS=192.168.1.71

2.2.3 Key definitions for a CDU [Optional]

See section 3.2

2.2.4 CDU screen parameters [Optional]

See section 3.19

2.2.5 Position of PSXseecon as CDU Window on your screen [Optional]

PSXseecon will remember the last position of PSXseecon as CDU window (see section 3.19) on your screen by writing (and reading) the following parameters to your PSXseecon.cfg.

MAIN_CDU_WINDOW_X=number
(may also be negative)
MAIN_CDU_WINDOW_Y=number

If PSXseecon.cfg does not exist, it will create one. The values of these parameters can be changed manually, but it is easier just to place the main window where you want it.

2.2.6 Infinite Rotary selector direction [Optional]

See section 3.3

2.2.7 Black- or Whitelisting of PSXseecon variables [Optional] Experts Only!

PSXseecon by default detects all PSXseecon variables in the .ssi file pointed to by the CONFIG_FILE parameter (section 2.2.2) and connects these parameters to PSX.

However, you may exclude one or more PSXseecon variables from being connected. The concepts of Black- or Whitelisting are introduced for that purpose.

With the parameter BLACKLIST you can **exclude** one or more variables and with the parameter WHITELIST you can specify which variables you want to be connected **only**.

BLACKLIST=var1
WHITELIST=var1, var2, var3

[specify one or more variable numbers, separated by comma's.]

A situation for which these concepts are particularly useful is this:

Suppose you have two Opencockpits hardware CDU's (V3) connected to the same PC, one for CDU Left and one for CDU Right. You combine the two SIOC scripts from my PSXcockpit archive (cduv31.txt and cduv32.txt) by putting these filenames in a .lst file and let one SIOC instance run the file.

You want two PSXseecon instances, one to expand to the Left CDU screen (with parameter CDU_L=1) and the other to the right CDU screen (with parameter CDU_R=1).

However, the Left instance should not respond to changes in the right CDU SIOC script and vice versa. This can be specified in PSXseecon.cfg of the Left CDU instance as follows:

WHITELIST=1401, 2086

and in the PSXseecon.cfg of the Right CDU instance you should specify:

WHITELIST=1403, 2088

2.2.8 Version Check [Optional]

VERSION_CHECK=1

At each start-up PSXseecon will check if there is a new version available. If so the – normally- green version message in the main window will turn yellow, and the new version number will be shown in parenthesis.

This is just a reminder, you have to download and install the new version yourself (see section 2.1.2). If you do not want this check, give this parameter the value 0.

2.2.9 Monitoring messages [Optional]

Sometimes it can be helpful to see the messages that pass between PSX and PSXseecon and or the ones that pass between PSXseecon and SIOC.

With the following parameters you can activate or de-activate the logging of messages in the PSXseecon.log file.

To log the Qh, Qi and Qs messages send by PSX to PSXseecon, write *(_IN_ from PSXseecons perspective)*:

MONITOR_PSX_IN_Qh=1
MONITOR_PSX_IN_Qi=1
MONITOR_PSX_IN_Qs =1

Note: The following Qs messages are never logged, while they can be up to 50.000 characters long and they are not really needed for cockpit builders: Qs375, QS376, Qs377, Qs407, Qs408, Qs409, Qs410, Qs411 and Qs412.

To log the non-Q messages send by PSX to PSXseecon, write:

MONITOR_PSX_IN_NONQ=1

To log all the messages send by PSXseecon to PSX, write *(_OUT_ from PSXseecons perspective)*:

MONITOR_PSX_OUT=1

To log the messages send by SIOC to PSXSeecon, write:

MONITOR_SIOC_IN=1

To log the messages send by PSXseecon to SIOC, write:

MONITOR_SIOC_OUT=1

To de-activate monitoring specify a 0 as value, or remove the parameter.
The default value of all these parameters is 0.

If you want to focus on a specific set of variables you may specify variable lists. To specify the Qh, Qi and Qs variables you want to monitor between PSX and PSXseecon (so both ways), write:

Qh_VarList=num1, num2 *Example: Qh_Varlist=426, 313*
[specify one or more variable numbers, separated by comma's.]

Qi_VarList=num1, num2
Qs_VarList=num1, num2

If a list is empty, all variables will be monitored.

In the log file, messages received by PSXseecon are preceded by a "<" (less than) character. Messages send by PSXseecon are preceded by a ">" (greater than) character.

The SIOC messages start with "Arn". For a description of the SIOC messages see the IOCP Protocol specification version 1.1

3 Predefined SIOC variables for PSX

The pre-defined SIOC variables reflecting the PSX interface are defined in a separate machine produced and machine readable document called SIOC variables by Qcode.txt.

The PSX interface is documented by Hardy in network documentation.txt and variables.txt, that are available in the 744 Forum, sub forum networkers, and in the developers folder at the 744 DVD.

SIOC variables are always **32** bit. After an introduction in section 3.1, specific variables will be detailed.

3.1 Introduction

3.1.1 Documentation method used

Each line in the documents 'SIOC variables_by_Qcode.txt' or 'SIOC_variables_by_Name.txt' specifies a SIOC variable.
A line contains five pieces of information, separated by commas. They are:

Name
The name of the variable. For most variables this name is equal to the name that is in use in the variables.txt document of PSX.

Qcode
The related Qcode.

SIOC
The number of the corresponding predefined SIOC variable (between 1000 and 5499).

Control Type *(control variables only)*
Possible control values:

Type	Meaning
[x..y]	Range of values, x=lower bound, y = upper bound. The brackets mean that values sent to this variable less than the lower bound will be set to lower bound and values greater than the higher bound will be set to higher bound.
P270	Potentiometer 270 degrees, range 0..4713 As [x..y] above, but in order to cope with jitter, with suppression of delta values less than 47.
[x.%.y]	As [x..y], but in order to cope with jitter, with suppression of delta values less 1% of the range.
x..y	example: -1..68 Values sent to this variable that are out of this range will be neglected.
A	Alternate Switch, values 0 or 1
M	Momentary Switch, values 0 or 1, at each 0->1 or 1>0 change (in SIOC) the function is activated
H	Momentary Switch (see above), that can be held. *Used for instance for some ACP and RCP buttons.*
R	Rotary Switch terminal, values 0 or 1. Put a 1 in this variable if the terminal is selected and a 0 if not. *Note: a Rotary Switch can have two or more terminals*
O	On/Off/On Switch terminal, values 0 or 1 At most one terminal can be selected, indicated by value 1. *Note: a On/Off/On Switch always has two terminals.*
E	Rotary Encoder
K	Opencockpits USB-Keys card

Information Value (*information variables only)*
The format of the information value, which can be:

- *Qcode value*
  for the definition of the value consult PSX's variables.txt document.
  PSXseecon transfers these bits integrally.

- Qh value (Virtual Switch position)
  See section 3.8

- a *value range*
  such as 0..1001

- *b n*
  meaning bit n

The way a bit (n) is indicated here is different from PSX in their network documentation document. It's n in **2 to the power of n**. This bit number can be used directly in SIOC's TESTBIT statements.

- *b 14 15 16*
  14 to 16 are bit numbers for leds in MCP Momentary Action Switches and in Indicator Lights that are not integrated. 14=left bulb, 15=right bulb and 16 is left or right bulb.

  If the bit is 1 the led will be lit; MD&T test, bulb failure and whether there is power in the power source for the bulb is incorporated.

  bits 0 to 13 are as defined in network documentation.txt.

- *b 14 15 16 17 18 19 20 21*
  14 to 21 are bit numbers for leds in Big Alternate or Big Momentary Action Switches. 14=upper/left bulb, 15=upper/right bulb, 16=upper/left or upper/right bulb, 17=lower/left bulb, 18=lower/right bulb and 19=lower/left or lower/right bulb.

  If the bit is 1 the led will be lit; MD&T test, bulb failure and whether there is power in the power source for the bulb is incorporated.

  For Big Alternate Action Switches bit 20 indicates that switch is pushed and there is (panel) power source for this switch; use this bit if the upper/left part of the switch is <u>not</u> an indicator light.

  For Big Alternate Action Switches bit 21 indicates that switch is pushed and there is (panel) power source for this switch; use this bit if the lower/right part of the switch is <u>not</u> an indicator light.

  bits 0 to 13 are as defined in network documentation.txt.

- *b x() y(z) …*
  whereby x, y and z are bit numbers for leds, with between parenthesis some clue about what they represent. Consult PSX's network documentation document for more information.

- BCDn
  n hexadecimal digits, range 0x0 .. 0xB, whereby A='blank, B='minus sign'
  These vars are meant for Display windows (see section 3.1.4).

3.1.2 Mapping of the Q Variable to a SIOC variable number

The pre-defined SIOC variables for PSX are defined in the range **1000 .. 5499**.
There are three Q categories: Qi (mostly information variables, some can be set too), Qh (control variables), Qs (mix of information and control)):

- The **Qh** category start at **1000**, e.g. Qh169 = 1169;
- The **Qi** category start at **2000**, e.g. Qi21 = 2021;
  The **Qi** category variables that can be set start at **5000**, e.g. Qi12 = 5012;
- The **Qs** category is not mapped one to one to a SIOC range because in SIOC strings are not supported. PSXseecon splits these strings up into a number of SIOC variables starting from **3000**.

3.1.3 Qh vars with information elements

PSX's Qh variables for big alternate action switches, momentary action switches, MCP momentary actions switches, Fuel Cutoff switches, Engine Start Switches, Green squib pushbuttons and Fire Switches have as well a control part as an information part.

In order to hide you from that level of detail, PSXseecon splits these Qh variables up into two SIOC variables, one for the control and one with the information. PSXseecon takes care of "splitting up/combining and re-sending" when communicating with PSX.

Take for instance Qh201:

```
BusTie1          , Qh201, 1201, A        ,
BusTie1          , Qh201, 3201,          , b 14 15 16 17 18 19 20 21
```

1201 is the control variable, it can be used in SIOC by linking an On/Off switch to it.
3201 is the information variable, it not only contains the bits from Qh201, but also eight extra bits.

Note: as a rule of thumb: the information variable always has the same number as the control variable +2000

If you are using two leds for each function in the switch (so 4 in total) you should take bits 14,15 and 17,18 for indicators and bit 20 and 21 for non-indicator functions. If you are using one led, you should take bits 16 and 19 for indicators and 20 and 21 for non-indicators.

NOTE: You cannot implement the information variable only, that will not work. Implement both, or at least the control variable.

3.1.4 Display windows

Values for 7-segment digits are packed in **BCD** (Binary Coded Decimal). Each digit takes four bits (a 'nibble') and the values are 0 to 9. On top of that there are the following special values defined by PSXseecon:

- Decimal 10 (= hexadecimal 'A') representing **blank**
- Decimal 11 (= hexadecimal 'B') representing the **minus sign**

With these definitions PSXseecon can provide support for blanks and minus signs.

The following combinations may occur:

BCD Format	Meaning
BCDx	x digits with leading zeroes
BCDxb	x digits with leading blanks (but at least one non blank)
BCD3b1	3 digits with at most one leading blank
BCDxbn	x digits, or x-1 digits and a with a minus sign, and possibly also leading blanks (and at least one non blank)

In the definition of a Variable you may find these codes.

The SIOC script for your cockpit can now be very simple, just map each digit straightforward to a hardware digit. In the PSXcockpit archive (at my web site) you will find the file 1_Digit.txt with all kinds of SIOC subroutines to assist you in doing that.

3.2 CDU keyboard

For the CDU keyboard there are two implementation options. Take for instance Qh401:

```
KeybCduL        , Qh401, 1401, K        ,
KeybCduL        , Qh401, 3401, -1..68   ,
```

SIOC variable 1401 can be linked directly to an Opencockpits USB-Keys card. PSXseecon will convert the integer key card codes to the codes expected by PSX in Qh401.

This variable works **out of the box** for USB-Keys cards coming with the Opencockpits 767/747 CDU (assembled or as kit)!

For USB-Keys cards with other key arrangements, built by yourself, you have to define the integer code for each key in **PSXseecon.cfg**, like this (in PSXseecon.cfg one parameter per line of course):

LSK_L1=56
LSK_L2=64
LSK_L3=72
LSK_L4=80
LSK_L5=88
LSK_L6=87
LSK_R1=86
LSK_R2=85
LSK_R3=84
LSK_R4=83
LSK_R5=82
LSK_R6=74
KEY_INITREF=79
KEY_RTE=78
KEY_DEPARR=77
KEY_ATC=76
KEY_VNAV=75
KEY_FIX=71
KEY_LEGS=70
KEY_HOLD=69
KEY_FMCCOMM=68
KEY_PROG=67
KEY_EXEC=66

KEY_MENU=63
KEY_NAVRAD=62
KEY_PREV=55
KEY_NEXT=54
LET_A=61
LET_B=60
LET_C=59
LET_D=58
LET_E=57
LET_F=53
LET_G=52
LET_H=51
LET_I=50
LET_J=49
LET_K=29
LET_L=28
LET_M=27
LET_N=26
LET_O=25
LET_P=21
LET_Q=20
LET_R=19
LET_S=18

LET_T=17
LET_U=13
LET_V=12
LET_W=11
LET_X=10
LET_Y=9
LET_Z=5
KEY_SP=4
KEY_DEL=3
KEY_SLASH=2
KEY_CLR=1
DIG_1=32
DIG_2=31
DIG_3=30
DIG_4=24
DIG_5=23
DIG_6=22
DIG_7=16
DIG_8=15
DIG_9=14
KEY_DOT=8
DIG_0=7
KEY_PLUSMINUS=6

The integer codes for your key card can be found easily with the siocmonitor program. In the cockpit744 folder you will find a file with all these parameters to be filled in.

And do not forget (!) to add a line

OCCDU=0

in order to switch off the OCCdu default.

Note that keeping key CLR or key ATC pushed does not work. Therefore CLR has been implemented just as any other key: a push will clear one character (if any). To simulate ATC Hold you have to push the ATC key once and to push it a second time in order to clear the hold.

SIOC variable 3401 behaves as defined in the network documentation document.

3.3 Infinite Rotary Selectors

The infinite rotary variables can be linked directly to a Rotary Encoder (with just one line of SIOC code). Take for instance Qh78:

```
McpTurnHdg        , Qh78 , 1078, E       ,
```

In SIOC it would read:

```
Var 1078 Static name TurnHdg Link IOCARD_ENCODER Aceleration 4
Device 2 Input 6 Type 2
```

If turning right does not increment but decrements the value, you can change that by changing the wires, or –easier- by adding a parameter to PSXseecon.cfg.

For each encoder there is a parameter named Invert+"variable name". Setting it to 1 will make the direction turn. E.g. to change the direction of the small freq knob of RCP L, write:

InvertSmallFreqKnobL=1

Default value is 0.

The possible parameters are (see also the PSXseecon.log file):

InvertDeltaStbyIasBug1	**InvertDeltaStbyIasBug2**	**InvertDeltaStbyIasBug3**
InvertAbsStbyIasBug1	**InvertAbsStbyIasBug2**	**InvertAbsStbyIasBug3**
InvertEcpMinsCp	**InvertMcpTurnSpd**	**InvertTurnStbyBaro**
InvertEcpBaroCp	**InvertMcpTurnHdg**	
InvertEcpMinsFo	**InvertMcpTurnVs**	
InvertEcpBaroFo	**InvertMcpTurnAlt**	
InvertHfSenseL	**InvertHfSenseC**	**InvertHfSenseR**
InvertBigFreqKnobL	**InvertBigFreqKnobC**	**InvertBigFreqKnobR**
InvertSmallFreqKnobL	**InvertSmallFreqKnobC**	**InvertSmallFreqKnobR**

3.4 Rotary switches

For all Rotary Switches PSXseecon defines a variable for each rotary switch terminal, take for instance Qh421:

```
SeatbeltsOff      , Qh421, 3783, R       ,
SeatbeltsAuto     , Qh421, 3784, R       ,
SeatbeltsOn       , Qh421, 3785, R       ,
```

In SIOC you can link an IOCARD_SW Type I to each variable. PSXseecon will convert this to the right value (0, 1 or 2) for Qh421.

3.5 OnOfOn Toggle switches

For all OnOffOn toggle switches PSXseecon defines a variable for each toggle switch terminal, take for instance Qh183:

```
LtIndTestOnL      , Qh183, 3751, O       ,
LtIndTestOnR      , Qh183, 3752, O       ,
```

In SIOC you can link an IOCARD_SW Type I to each variable. PSXseecon will convert this to the right value (-1, 0 or 1) for Qh183. The variable with suffix "L" is coded -1, the variable with suffix "R" is coded +1. The middle position is coded 0.

3.6 Power sources for indicators lights and panel lights

Qs323 MdtVoltages is converted by PSXseecon to 16 different SIOC variables starting from 4920.

Qs324 LtVoltages is converted by PSXseecon to 22 different SIOC variables starting from 4936.

3.7 Rudder and Aileron Trim commands

The network commands for Rudder (Qh416) and Aileron Trim (Qh418) have been added integrally. This is how to use the values for these Qh variables:

Qh416:
-2: fast left.
-1 slow left.
0: neutral
1: slow right
2: fast right;

Qh418:
0: both switches center.
1: upper switch left.
2: lower switch left. (so you need 3 to trim to the left.
4: upper switch right.
8: lower switch right (so you need 12 to trim to the right).

Note they have to be sent in DELTA mode (see Hardy's network documentation.txt)

3.8 Hardware (solenoid) synchronisation

In general your hardware switches will be dominant over the virtual switches in PSX; PSXseecon will take care of that.

However, there are a few switches for which that is not always the case. These are the GearLockOverride Switch, the Engine Start Switches, the Fire Engine Override Switches, Fire Apu Override Switch and the Autobrakes Selector.

The virtual Autobrakes, for instance, will automatically return (to either OFF or DISARM) and the Engine Start Switches will jump back in PSX if N2 reaches a certain value.

In order to be able to let your hardware switches (via solenoids) synchronise with the virtual switches PSXseecon provides information variables (4700 to 4009) for that purpose, indicating the position of these virtual switches.

In the SIOC variable definitions this is indicated by:
Qh value (Virtual Switch position)

Look in the Qcode specifications of Hardy what bit you need. For instance for the FireEngx Switch bit 0032 (bit 5) is the override lock pushed value.

3.9 AutoThrottle

PSXseecon derives from Qs434 the SIOC variable **3434 ATActive** with values:
1 = AT is moving the virtual Throttle Levers, and
0 = AT is not moving the virtual Levers (BLANK or HOLD).

If ATActive = 1 psxeecon, will upon receival of a virtual Lever update from PSX, not sent the hardware Throttle Lever variables to PSX (so the hardware is no longer dominant!). But, if ATactive is going from 1 to 0, PSXseecon will immediately write the hardware Throttle Lever variables to PSX (once).

If ATActive = 1 SIOC variable **4739 TlaMotor** is +1 if at least one of the four Levers is moving upwards, -1 if at least one of the four Levers is moving downwards and 0 otherwise. If ATActive = 0, TlaMotor will be 0.

The four variables, **Tla1Virtual**, **Tla2Virtual**, **Tla3Virtual** and **Tla4Virtual**, give the value of the virtual Throttle Levers. They are only written if ATActive = 1 and can be used to set the hardware Throttle levers according to the virtual ones.

3.10 Temperature potentiometers

Qh302="TempPassenger"; Mode=ECON; Min=-4000; Max=4000;
Qh303="TempFlightDeck"; Mode=ECON; Min=-4000; Max=4000;

The pot range is -2400 to +2400.
When the knob moves outside this range, the knob jumps to 6 o'clock.

The C toggle position is -2800.
The W toggle position is +2800.
The 6 o'clock toggle position is 3141.

The values are radians * 1000, i.e. 3141 = pi * 1000 = 180°

3.11 TCAS Rotary Switches

Qs117 is a complex "string" variable because it controls the 4 rotary switches of the TCAS. It is translated by PSXseecon to 11 Rotary terminals in SIOC:

```
Qs117, TcasPanSelAb       , 3015, R        ,
Qs117, TcasPanSelBe       , 3017, R        ,
Qs117, TcasPanSelL        , 3013, R        ,
Qs117, TcasPanSelNo       , 3016, R        ,
Qs117, TcasPanSelOff      , 3011, R        ,
Qs117, TcasPanSelOn       , 3012, R        ,
Qs117, TcasPanSelR        , 3014, R        ,
Qs117, TcasPanSelRATA     , 3021, R        ,
Qs117, TcasPanSelStby     , 3018, R        ,
Qs117, TcasPanSelTA       , 3020, R        ,
Qs117, TcasPanSelXpdr     , 3019, R        ,
```

Be aware that you have to implement them all, otherwise the rotary(ies) that are missing will be forced to the 0 position, and you can no longer control these with the mouse.

3.12 Towing control variable

In order to be able to operate the Towing controls via hardware instead of with mouse in the Instructor panel, a SIOC **control** variable 5191 is provided with the interface of Qi191, next to the already available SIOC **information** variable 2191 for Qi191.

Both variables always consist of 6 digits. E.g. 180359

Digit 1:
1 = "Push back" selected
2 = "Push forward" selected

Digits 2 and 3:
20 = "Stop" selected
80 = "Auto" selected
99 = "Start" selected

Internal process (if "Start" selected):
98 = Phase: "Steering pin inserted; release brakes please."
97 = Phase: "Pushing back." (Not called when pushing forward.)

Internal process (if "Auto" selected):

79 = Phase: "Steering pin inserted; release brakes please."
78 = Phase: "Pushing back." (not called when pushing forward.)

Internal process (general):
15 = Phase: "Set parking brake, please."
10 = Phase: "Towing system removed, and steering pin removed."

Digits 4, 5, 6:

Target heading (true north reference).

An example SIOC script, named **0_Towing.txt**, is available in my file PSXcockpit.zip

3.13 Variables for dedicated hardware

Note that these variables are not needed in a full 747-400 cockpit with LCDs and instrument windows.

Removed in version 2.0.0, except:

3.13.1 Altitude, Heading and TAS

From Qs121 PSXseecon derives three SIOC variables:

```
Qs121, Altitude      , 4886,    , BCD5b (if positive), 0xAA + BCD3bn (if negative)
Qs121, TrueHeading , 4887,    , BCD3
Qs121, TAS           , 4888,    , integer value (note: not * 1000)
```

3.14 PSXseecon added control variables

The control variables in this section are introduced by PSXseecon (they are not in Hardy's list).

3.14.1 CDU Screen Selector

```
Qh998, CduScreenSelector        , 1998, [0..2]
```

With this control variable you can dynamically switch the information at your CDU screen (see section 3.19) to CDU_L (value 0), CDU_C (value 1) or CDU_R (value 2).

An example SIOC script for fully sharing a MCDU, so also keyboard and leds, can be found in my **PSXcockpit** archive at my website; the file is called **5_Cdu1_2_and3Shared.txt**. There is also a file 5_Cdu1and2Shared.txt if you want to share Left and Center.

3.15 Configuration variables

The 32 Qi Cfgxyz variables (Qi0 .. Qi31) are available in the SIOC range 5000 .. 5031. These variables determine the configuration of the 747-400. They are useful for guaranteeing that each flight you load adapts to your hardware.

See for example the SIOC script I'm using:

```
Var 0 Value 0
{
    // always set CF6 engines because I'm using TopCat + PFPX
   &CfgEngModCtrl = 0

   // My MCP has just two mach digits
   &CfgMachSCtrl = 1
}

Var 5001 Static name CfgEngModCtrl
Var 5013 Static name CfgMachSCtrl
```

3.16 MalFunction variables

The Malfunction variables Qs137 up to Qs320 are available as SIOC control variables 5237 up to 5420 with a 3 bits interface.

Take for instance Qs213 MfFlameout1:

Qs213, MfFlameOut1 , 5313, bits 0 & 1: Severity, b 2: Set/Clear MF,

The first two bits (0 & 1) can be used to set severity:

- If both bits are 0 the severity value from the Severe checkbox in the Instructor panel is taken.
- If bit 0 is set and b 1 is not set a '**non severity'** is set.
- If bit 0 is 'not set and b 1 is set a '**severity'** is set.

Bit 2 determines whether the resulting Qs message is triggering or clearing the Malfunction.

So the possible assignments in SIOC are:

```
Var 5313 Static name MfFlameout1

&MfFlameout1 = 6  // trigger a severe malfunction
&MfFlameout1 = 5  // trigger a non severe malfunction
&MfFlameout1 = 4  // trigger a malfunction with the severity
                  // defined in the instructor panel
&MfFlameout1 = 0  // repair the malfunction, and let the
                  // severity as is
&MfFlameout1 = 1  // repair the malfunction, and set the
                  // severity to non-severe
&MfFlameout1 = 2  // repair the malfunction, and set the
                  // severity to severe
```

Note that the last two assignments are possible with this interface but not really needed. Severity is more relevant when triggering a malfunction.

For convenience PSXseecon will always automatically generate a **Qi130 MfEvent** message after each Malfunction message so PSX will act immediately, making your SIOC code as simple as possible.

For examples of the use of Malfunction variables see my file **0_MalFunction.txt** in cockpit744.zip (at the PSXseecon web page).

3.17 Doors Control variables

With the following Qi control variable we can control opening and closing of Doors.

Qi179, DoorComBitsCtrl , 5179, Qcode value,

bit	Command
0	COM_CLOSE
1	COM_OPEN
2	COM_PARK
3	COM_BOARD_L
4	COM_BOARD_R
5	COM_OFF

The COM commands set the Doors checkboxes on the Instructor, see also Aerowinx Manual page 40.

(!) The COM_OFF command sets all three radio buttons to zero. You have to do that before you can make use of the following two control variables, otherwise the PSX doors randomizer function will interfere with what we would like to do...

With the following two Qi control variables we can open/close several doors and/or set the escape slide mode from Auto to Manual.

```
Qi180, DoorOpenBitsCtrl    , 5180, Qcode value,
Qi181, DoorManBitsCtrl     , 5181, Qcode value,
```

bit	Command	C	P	F
0	DR_NOSE_CARGO			x
1	DR_FWD_OVHD			x
2	DR_MAIN_ELEC	x	x	x
3	DR_ENTRY_1L	x	x	x
4	DR_UPPER_L	x	x	
5	DR_ENTRY_2L	x	x	
6	DR_ENTRY_3L	x	x	
7	DR_ENTRY_4L	x	x	
8	DR_SIDE_CARGO	x		x
9	DR_ENTRY_5L		x	x
10	DR_ENTRY_1R	x	x	
11	DR_FWD_CARGO	x	x	x
12	DR_UPPER_R	x	x	x
13	DR_ENTRY_2R	x	x	
14	DR_CTR_ELEC	x	x	x
15	DR_ENTRY_3R	x	x	
16	DR_ENTRY_4R	x	x	
17	DR_AFT_CARGO	x	x	x
18	DR_BULK_CARGO	x	x	x
19	DR_ENTRY_5R		x	

C : Combi
P : Pax
F : Freighter

Note that these three control variables are leading (have priority) over settings in the instructor panel.

3.18 Call From Cabin variable

With the CallFrCabinCtrl variable we can simulate calls from the Cabin:

```
Qi102, CallFrCabinCtrl     , 5102, Qcode value,
```

Possible values are:

Val	Call
01	DIRECTORY
02	VIDEO IN USE
03	PA IN USE
04	CABIN READY (with Call light)
05	PILOT ALERT
06	PRIORITY
07	PARTY LINE
11	DOOR 1 LEFT
12	DOOR 2 LEFT
13	DOOR 3 LEFT
14	DOOR 4 LEFT
15	DOOR 5 LEFT
16	U/D LEFT DOOR
21	DOOR 1 RIGHT
22	DOOR 2 RIGHT
23	DOOR 3 RIGHT
24	DOOR 4 RIGHT
25	DOOR 5 RIGHT

26	U/D RGHT DOOR
31	PILOT
32	PRIORITY STA
33	PRIORTY PILOT
34	PURSER
36	U/D STAIR
41	PA FIRST CL
42	PA BUSINESS
43	PA ECONOMY CL
44	PA CREW REST
46	PA ALL AREAS
47	PA PRIORITY
51	DOOR 2 GALLEY
52	M/D GALLEY
53	DOOR 4 GALLEY
55	ALL CALL
56	U/D GALLEY
61	DR5 CREW REST (Pax only)
66	CR REST PILOT
67	CABIN READY (inhibits Call light)
71	GND CREW CALL
72	CARGO CALL (Combi only)
73	U/D CREW REST
74	M/D CREW REST
77	PILOT ALERT

One useful use could be sending value 67 indicating CABIN READY

3.19 CDU screen

Acknowledgement
I owe particular thanks to Martin Erdelen, Jeroen Hoppenbrouwers and Hardy Heinlin for the creation of the Aerowinx MCDU Large and Small true type fonts. Thank you!

The main window of PSXseecon can be changed dynamically into a CDU screen window.

To do this add **one** of these lines to PSXseecon.cfg:

CDU_L=1
CDU_C=1
CDU_R=1
(possible values 0 .. 1)

and install the **Aerowinx MCDU Large** and **Small fonts** which are available in the PSXseecon package. Help about how to install fonts can be found here.

The screen change happens as soon as SIOC and PSX are connected. The main PSXseecon window will then look like this (example):

The size of this **PSXseecon window** is defined by the following parameters in PSXseecon.cfg (t*he values shown are the **default** values*);

PSXSEECON_AS_CDU_WIDTH=900
(possible values 600 ..1200)
PSXSEECON_AS_CDU_HEIGHT=650
(*possible values* 400 *.. 900)*

In that space you can (re-)size and (re-)position the CDU window so it will fit perfectly over you CDU monitor with black space all around, by defining the following parameters:

CDU_WIDTH=620
(possible values 200 ..1200)
CDU_HEIGHT=510
(possible values 200 ..1000)

The CDU_WIDTH defines the width of the CDU window; CDU_HEIGHT defines the height of the CDU window.

With the CDU_X_OFFSET you shift the CDU window to the right in the PSXseecon window. With the CDU_Y_OFFSET you shift the CDU window down:

CDU_X_OFFSET=100
(possible values 0 .. 400)
CDU_Y_OFFSET=50
(possible values 0 .. 400)

There are 14 lines of text in a CDU window. Text and data can be written with large and small font characters/symbols. The size of these fonts (as well for Large as Small) is defined by the CDU_FONTSIZE parameter:

CDU_FONTSIZE=44
(*possible values 12 .. 76)*

With this rather special parameter you can shift the small font characters/symbols a little bit up, so they appear at the positions as in the CDU screen of PSX:

CDU_SMALL_FONT_Y_OFFSET=7
(possible values 0 .. 15)

If CDU_BOLD is 1 PSXseecon will emulate BOLDNESS by re-printing the char/symbol a little bit to the right and by reprinting the char/symbol a little bit up:

CDU_BOLD=1
(possible values 0 .. 1)

The color (green) can be changed as well, by defining the RGB values:

CDU_COLOR_R=50
CDU_COLOR_G=205
CDU_COLOR_B=50
(possible values 0 .. 255)

PSXseecon will remember its window position used as CDU screen; it writes the X and Y position to your PSXseecon.cfg:

MAIN_CDU_WINDOW_X
MAIN_CDU_WINDOW_Y

These values can be adjusted manually. When you start again it will automatically move on top of your CDU monitor to its last position.

With the following parameter you can get rid of the Title Bar and Borders of the PSXseecon main window:

CDU_BORDER=1
(possible values 0 .. 1)

If CDU_BORDER = 0 you will see this kind of window:

A drawback is that you now no longer can drag the window to its position on top of the CDU Monitor.

To go around this “problem” you can start with BORDER=1 and drag the window to the CDU monitor first, before switching to BORDER=0. You will notice that the window has

gone up and/or left a bit so you slightly change that by manually changing the MAIN_CDU_WINDOW_X and MAIN_CDU_WINDOW_Y parameters in PSXseecon.cfg.

If everything is set you are ready; next time you start PSXseecon it will jump to the right position on your CDU monitor from the start.

Note that with a special control variable ***CduScreenSelector*** it is possible to display as well CDU-L, CDU-C as CDU-R on one physical MCDU. Read more about it in section 3.14.1

NOTE:

You may run three PSXseecons as CDU screen, each for a different CDU. These PSXseecons can use one (shared) SIOC script running by one SIOC instance (each PSXseecon using a different part of that same script, see section 2.2.7) or each PSXseecon uses a separate script at a separate SIOC instance, or a combination of both approaches.

3.20 Flight Controls

Qs120, AileronsVirtual , 4743, , Qcode value
Qs120, ElevatorVirtual , 4744, , Qcode value
Qs120, RudderVirtual , 4745, , Qcode value

Virtual variables 4743, 4744 and 4745 (deduced from Qs120) can be used to read the values for Ailerons, Elevator and Rudder in PSX *(so called panel values or virtual values contrary to your hardware real values).*

Qs120, AileronControlW , 3709, [-999.%.999]
Qs120, ElevatorStick , 3708, [-999.%.999]
Qs120, RudderPedals , 3710, [-999.%.999]

With the variables 3709, 3708 and 3710 you can feed var Qs120 with information about the state of your hardware Aileron, Elevator and Rudder. They are used to write that info into PSX. It is not obligatory to use all three. If, for instance, the rudder is controlled via a USB device you may leave 3710 out.

3.21 ToeBrakes

Qs357, ToeBrakeLeftVirtual , 4746, , Qcode value
Qs357, ToeBrakeRightVirtual, 4747, , Qcode value

Virtual variables 4746 and 4747 (deduced from Qs357) can be used to read the values for the ToeBrakes in PSX.

Qs357, ToeBrakeLeft , 3711, [0.%.1000]
Qs357, ToeBrakeRight , 3712, [0.%.1000]

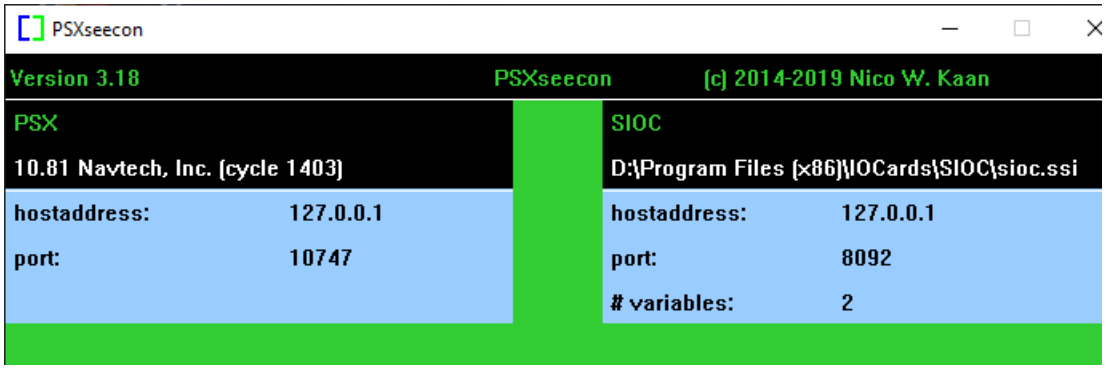
With the variables 3711 and 3712 you can feed var Qs357 with information about the state of your hardware ToeBrakes. They are used to write that info into PSX. It is not obligatory to use both.

4 Running PSXseecon

4.1 Starting sequence

Start PSX, SIOC and PSXseecon.exe. In principle the order does not matter. PSXseecon will continuously try to connect to the IOCP Server of SIOC and to PSX.

If PSX and SIOC are connected, the window will look like this:



The window shows status information only, there are no menu's with commands, other than the standard window controls from Microsoft (top right).

The header gives version, title , copyright and author information.

The left half gives information about PSX and the right half gives information about the IOCP Server in SIOC.

In the black area (in white) the version of PSX and the SIOC CONFIG_FILE that is in use are given.

The PSX part shows hostaddress and port of PSX.

The SIOC part shows hostaddress and the port of the IOCP Server found. The number of PSXseecon variables found in your CONFIG_FILE is also shown.

Note that IP address 127.0.0.1 means that the server is at the same PC as the one running PSXseecon.

If both servers are connected PSXseecon will start exchanging information (its actual job!). It will automatically stop without further notice and exit if the link to SIOC breaks down, for instance because SIOC has exit, or if the link to PSX breaks down. You can manually stop PSXseecon via the red Windows X button in the PSXseecon window.

PSXseecon will remember the position of its main window on your monitor (it writes the X and Y position to your PSXseecon.cfg). When you start again it will pop up at the same place.

If PSXseecon as CDU screen is wanted, the main window will expand and change to a CDU window (see section 3.19).

4.2 Starting PSXseecon and hardware Switches synchronisation

Hardware switches are dominant over virtual switches. When you start PSXseecon while PSX is in a flight, it might be that your hardware switches are not in the same position as the virtual ones. An illustrative example is the Landing Gear. You do not like it to be lowered when you are in a cruise flight … Or that the Autopilot switches off, or the Fuel Switches …

The same problem occurs while PSXseecon is already running, but you load a new situ in PSX (without stopping PSXseecon).

To synchronise your switches with the virtual ones in those situations you better start PSX and SIOC first (and not PSXseecon). Load the wanted flight in PSX, then start PSXseecon.

For fine tuning you could also Freeze PSX after loading (with the M key), then start PSXseecon, fine tune, and finally start PSX by unfreezing with the M key.

4.3 Update frequency

PSXseecon buffers messages coming from and going to SIOC and PSX. Twenty times a second it flushes these buffers and it checks if messages has to be sent to SIOC and or PSX. So in the terminology of Hardy it runs at **20Hz**. However, changes in values of Information variables are checked at 5Hz, which is sufficient for flight-decks.

4.4 Log file

During each run a log file named **log.txt** will be written in the **same** folder as PSXseecon.exe. The previous log will be saved under the name log0.txt.

Logged are version, configuration parameters, the PSXseecon Variables used, error messages and warning messages. It is certainly worthwhile to have a look at it every now and then.

4.5 Error messages

Error messages are always severe. They are shown at the bottom of the main window in yellow with a red background.

After reading the message, you have to stop PSXseecon.exe manually, using the standard Window's X button (top-right).

The error messages that may appear have been described in the following Tables.

SIOC related error messages:

#	**Error Message** *(and help)*
1	**No variables found** PSXseecon variables are numbered from of 1000 to 4999 and must have attribute static.
2	**Var <var> in your <CONFIG FILE> is not a PSXseecon variable** Update your SIOC source first.
11	**sioc.ini and PSXseecon.cfg not found** If SIOC was not installed at your FS computer, add a PSXseecon.cfg file. If SIOC was installed at your pc, reinstall it in a folder C:\SIOC or preferably D:\SIOC
12	**IOCP_HOSTADDRESS information is missing in <file>** Add that information to <file>
13	**IOCP_port information is missing is missing in <file>** Add that information to <file>
14	**CONFIG FILE information is missing in <file>** Add that information to <file>
15	**CONFIG FILE has no .ssi or .lst extension** Is the CONFIG FILE parameter in your sioc.ini or PSXseecon.cfg correct?

16	**CONFIG FILE not found** Is the CONFIG FILE parameter in your sioc.ini or PSXseecon.cfg correct?
17	**Connect to TCP/IP failure, <hostaddress>, <port>** A Fire Wall might block the connection to SIOC request. Allow PSXseecon access.
18	**Connection attempt to SIOC timed out** Is the IOCP_HOSTADDRESS the same as the HOSTADDRESS in the SIOC main window?
19	**SIOC not detected** Is SIOC running?

PSX related error messages:

#	**Error Message** *(and help)*
21	**Connect to TCP/IP failure, <hostaddress>, <port>** A Fire Wall might block the connection to PSX request. Allow PSXseecon access.

Generic error message:

#	**Error Message** *(and help)*
100	**System Error** Please report this with your PSXseecon.cfg and PSXseecon.log files to the author or in the Aerowinx forum.