FSUIPC for Advanced Users (for FSUIPC Version 3.90, February 2009)

For changes from the previous release, please refer to the History document.

Contents

General description of FSUIPC main functions 2

Options in the FSUIPC.INI file *(for FS2000–FS2004 primarily)* 3
  Message Window Options (FS2004 only) 3
  General weather options 3
  Winds 5
  Visibility 7
  Clouds and precipitation 9
  Temperature 11
  Other options 11

Logging facilities 17

Monitor facilities 17

JoyNames 19

Profiles 19

Button Programming 19
  Format of button definitions 21
  Sequences, combinations and mixtures 23
  Adding offset conditions 25
  Errors in button parameters 28

Keyboard programming 29
  Format of key definitions 29
  Errors in key parameters 30

Additional “FS” controls added by FSUIPC 31

Macro controls 37

Axis assignments 41

Programs: facilities to load and run additional programs 42

Assignment of FLAPS_SET control (for FS2002 only) 43

Assignment of additional controls 43

Multiple joysticks 44

Helicopter pitch and bank trim 45

Message Filters 45

Facility for multiple INI installations 46

Appendix: “About the *Aircraft Specific* option and ‘*ShortAircraftNameOK*’” 47

General description of FSUIPC main functions

FSUIPC cannot and does not convert everything that may ever have been known in FS98 to work in the same way in FS2000, and it gets harder and harder in FS2002 and FS2004. It does its best, and will improve as it is developed and as more is found out about FS innards. Many actions in FS98 that were simply triggered by writing into the global data area simply will not work that way on the subsequent versions. For each specific action which may be needed and which doesn't simply map to a location, FSUIPC has to trap the changes and call routines in different parts of FS to cause the action to occur. So far this works with throttles and many of the other analogue inputs, and with some other things, but by no means everything (yet).

That understood, FSUIPC provides the following additional facilities when used with FS2000–FS2004:

- Converts FS98 offsets, for data in the FS globals area, into appropriate offsets for the same data in the later versions. This applies to those that are known to have moved but which are still available and located successfully. Note that sometimes the changes vary even in a single version: for example the N1% and N2% values for Jets are in the FS98 positions for transposed FS98 aircraft but swapped over for FS2000 native aircraft. FSUIPC deals with this particular difference (but not for FS2002 or FS2004, where it seems unnecessary).

- Obtains some information such as ambient wind details, jet EPR, Fuel Flow, and other engine related data which is not otherwise available in the later FS globals at all, or actually available for some aircraft types and not others.

- Provides operating analogue inputs from locations for throttles, propeller pitch, fuel mixture and many other aspects which otherwise don't occur because the data there is not acted upon when changed. Amongst the controls here are gear, brakes, spoilers, flaps and of course the primary flight controls.

- Constructs weather data structures for the different FS weather engines from FS98-style weather details placed into the old FS98 global weather positions. This allows programs such as Real Weather, FS_Meteo, Flight Director and SquawkBox to control the FS weather as they did in FS98, or better.

- Detects when FS2000 or FS2002 downloaded 'real weather' is in operation locally, and decodes this for application programs to use. If a weather control program wants to change the weather, FSUIPC automatically clears the local weather so that the external control can be implemented. In FS2004 external programs can read specific weather at named stations.

- Optionally patches the FS2000/2002 adventure interpreter so that the FS98 weather variables again contain relevant weather data (assuming the other weather features are also left enabled). See the section later, entitled "Weather Data for Adventures". Note that this is not applicable to FS2004—the old Adventure system is no longer supported by FS.

- Provides additional joystick calibration and centring facilities and enables fully proportional analogue toe brakes to be used in FS2000 (these weren't otherwise supported by FS then).

None of these functions are performed by WideServer (part of the WideFS package). Because of the complications surrounding the operation of throttles, flight controls and other inputs, and of course the weather system, WideServer depends upon FSUIPC to perform ALL accesses to FS's innards. It is therefore important, when both modules are used, to make sure they are compatible.

Options in the FSUIPC.INI file *(for FS2000–FS2004 primarily)*

In a user-registered FSUIPC installation, all of the interesting options can be controlled through the Options and Settings window obtained by selecting the FS Modules menu, then FSUIPC (ALT, M, then F). This is the recommended way, and allows changes 'on the fly'. Changes made in that dialogue are recorded in a file so that they are retained for the next re-load.

Almost all of these options are all recorded in the [General] section of FSUIPC.INI, which is an editable text file initially created for you in the Modules folder. There are many weather processing options, and almost all are only applicable to FS since FS2000 (and possibly CFS2). Those marked (*) can be controlled (i.e. overridden) by external programs interfacing to FSUIPC, unless this is prevented by the **ExternalOptionsControl** parameter.

Only those parameters shown **underlined** are *not* adjustable within the Settings window (for a registered user).

Message Window Options (FS2004 only)

Unlike almost all of the other parameters here, these are all available to unregistered users as well as those who have registered.

ShowMultilineWindow: This will be "Yes" if the relevant checkbox on FSUIPC's front page (About+Register) is checked. Multiline messages are directed to a translucent window like the one used by FS's ATC.

SuppressMultilineFS: Determines whether multiline messages are forwarded on to FS for its message window. If the above option is 'Yes' then this setting isn't relevant.

SuppressSingleline: Operates the option to prevent all single line messages in FS's (or AdvDisplay's) message window. Such messages are simply discarded if this option is selected.

See also **WhiteMessages** and **AdvDisplayHotKey**, both covered in the 'Other Options' section.

General weather options

PatchWeatherToADV: This parameter controls the facility, specific to FS2000/2002 only, to patch the variable table in ADVDRV.DLL (the adventure interpreter) so that the weather variables report the same sort of values for the same sort of weather as did FS98. In FS2002 this option also controls the patching of autopilot values and control facilities, which would otherwise be missing. Only set this to "No" if you do not want such FS98 compatibility for your adventures. Note that successful patching is Logged (if logging is enabled at all), as is an unsuccessful attempt. This facility is not applicable to FS2004 or later.

AdjustWeatherATIS: This option applies to FS2002 only and is primarily intended for FSMeteo users. When enabled, and the user is running with 'global' weather (not downloaded or manually set local weather), FSUIPC intercepts weather requests from ATIS and ATC and substitutes 'corrected' values. For cloud bases it provides AGL values, and for clouds, pressure (QNH) and visibility, it provides *destination* values. These can all be set separately by programs such as FSMeteo. The AGL values provided depend on the *surface temperature altitude* value being correctly set. FSMeteo sets this to the METAR station altitude. If it isn't being set, FSUIPC uses the current ground altitude, which may give odd results at times. This facility is not applied to FS2004 as it is not needed.

AutoClearWeather: FSUIPC will, by default, automatically operate the "Clear All Weather" function in FS2000–2004 if local weather is in force and:

(a) An FS98 weather control program changes the weather, or
(b) The "Force Weather" hot key is used (see next parameter), or
(c) The "Clear All" command is received on the Advanced or (in FS2004) New Weather Interfaces.

If this automatic action is not required, set this parameter to "No".

Note that in FS2004, FSUIPC assumes that local weather is *always* in force. The way FS2004's weather engine works allows no differentiation. Each weather station is populated with weather details which may be unique, or which may derive directly from a global set of values. In addition, FS2004 implements a weather changing algorithm, "weather dynamics", which can allow the weather to be changed individually at each station with time. FSUIPC will, by default, turn off the dynamic action when it clears all weather by any of the above methods *except* the New Weather Interface (which has its own Dynamics control facility). This option is set in FSUIPC's **Technical** page and by this INI parameter:

ClearWeatherDynamics: For FS2004 only, see above.

OwnWeatherChange: For FS2004, this is defaulted to 'No'. When set 'Yes' it allows the weather filtering options for clouds, winds and visibility to be applied to FS's own global weather (as opposed to global weather set through FSUIPC by an external program). The disadvantage of having this option enabled is that FS2004 weather then always reverts to "User Defined" whenever any weather filter options are enabled, preventing weather "Themes" remaining selected. This option is controlled by a check box on the Technical options page.

ForceWeatherKey: This allows you to assign a key press which, when used, will force-update the weather. When you use this keystroke it will re-form the FS2000/2002 weather from the last weather received from the external application. It will even remember the last weather received from the application after the latter is terminated. The second time it is used *with no intervening Weather change* it clears the weather completely, just as if FS2000's "Clear All Weather" button had been pressed.

Note that if the "AutoClearWeather" option is disabled you may need to clear the weather manually in the FS dialogues before the hotkey will restore the external weather.

The keystroke is defined as in Flight Simulator's own controls, and documented in my FS98 and FS2000 Controls documents (and listed below, in the Button Programming section). For example, I use (and recommend) "CTRL+SHIFT+W" which would be

```
ForceWeatherKey=87,11
```

The same control codes are used in FS2002 and FS2004.

SendWeatherInterval: In FS2002 (only), the weather provided by the built-in ATIS reports does not necessarily abide by the current weather prevailing. To get these reports updated, FSUIPC has to send weather change signals to other parts of FS2002. Unfortunately this results in an update of the ATIS identifier, which in real life only occurs at hourly intervals. So FSUIPC offers two options: this one, to set the *minimum* interval between weather updates for ATIS (in seconds), and the next one, to avoid sending any such updates unless the weather has changed noticeably.

The default interval is set for 60 seconds. You shouldn't set it too short, but you may want to set it to a larger value. But be aware that this interval is imposed even if you transit to another METAR station area, but FSUIPC does recognise when you've loaded a new flight, so the interval is not imposed at this time.

To stop FSUIPC ever sending weather change signals, set this parameter to 0.

SendWeatherAlways: [*FS2002 only*] By default the ATIS weather is only sent when there are noticeable changes in the weather (the criteria are listed below). By setting this parameter to 'Yes' you can make FSUIPC send the updates at the specified intervals whether there's been any change or not.

The criteria for deciding on weather changes are as follows. You should note that all these refer to *surface* weather, not necessarily the weather at the aircraft. Furthermore, if you are using FSMeteo and have set the destination weather, *and* FSMeteo has now supplied this for ATIS reports, it will be this weather which is used in the comparison even though FS's own ATIS will be reacting to current weather, not the station weather.

- Surface temperature altitude changed* (this is used for METAR station elevation)
- Temperature changed by more than 3C
- Wind speed changed by more than 5 knots
- Wind direction changed by more than 5 degrees
- Cloud base changed by more than 500 feet
- Cloud cover (lowest layer) changed by more than 2 oktas
- *Any* change in precipitation from the lowest cloud layer*
- Visibility changed by more than 50% of lower value
- Barometric Pressure changed by more than 5 mb

*Note that the broadcast is performed immediately and without further checks when there's any change in the METAR station altitude or in precipitation, no matter what these INI file parameters say. This is in an attempt to prevent the FS2002 problem whereby the cached weather retains everlasting rain even though "Clear All Weather" actions.

Winds

WindTransitions (*): [*Not FS2004*] If you enable this option FSUIPC will operate all FS2000/2002 winds other than those in ‘local weather’ mode (provided by the downloaded ‘real weather’ feature) in such a way that the transition across wind layers is reasonably smooth. It does this by setting only one wind layer into FS2000/2002—a very deep surface wind layer. The wind speed and direction is then programmed into this layer on a second-by-second basis according to the actual requested wind layer prescribed by the weather control program and the current plane altitude. At altitudes nearer to a layer boundary than 250 metres the actual speed and direction is computed proportionally.

Note that if 250 metres is more than 10% of the current layer’s thickness, then that 10% is used instead. This allows some amount of wind shear to be set if required by defining a very narrow upper wind layer.

WindSmoothing (*): [FS2004] This option controls the FS2004 wind smoothing options, both those dealing with externally set global weather (not so effective), and those operating only on the wind experienced at the aircraft.

WindSmoothness: Except on FS2004, where it operates differently, this facility only operates when WindTransitions are enabled, but unlike WindTransitions, it also works for downloaded ‘local weather’. It allows the wind changes to be restricted to a maximum of so many knots and so many degrees per second—with the default set to 5 (knots or degrees), which seems to work quite well. It is designed to prevent sudden wind changes when the weather control program selects a new METAR station, or the user loads a new METAR report. The feature does not operate when the aircraft is on the ground (or being slewed in the air from a ground start). To switch the smoothing off, set this parameter to 0.

On FS2004 wind smoothing operates on two ways. First, rather ineffectively (because FS2004 weather does not stay global), it works on global weather set by external programs. In this case it smooths changes to the wind layer in which the aircraft is currently situated, and the one immediately above and below. Second, with rather too much effect (as it smooths out gusts and turbulence too), it operates on the wind actually experienced at the aircraft. In this case the smoothed wind may not be reflected in ATIS and other weather reports.

WindSmoothingDelay:
WindSmoothAirborneOnly: These two, for FS2004 only, control the timing of the all-weather wind smoothing action, if it is enabled. The “delay” value is in seconds, and delays the start of smoothing after any “clear all weather” action (whether by loading a Flight, using the FS weather menus, using the FSUIPC clear weather facility, or by specific external program action). The option to smooth only when airborne allows the winds at the airport to be changed and set as desired, before take off.

ExtendTopwind: This option extends the highest current wind layer to operate all the way up to 100000 feet. This is really intended as a stop-gap for downloaded real weather, which only supplies a thin surface wind layer and no upper winds. On FS2004 this is only applied to global weather and all externally supplied weather.

MaxSurfaceWind: This allows the surface wind to be limited to a specified maximum wind speed, in knots. This facility is disabled if the value assigned here is 0. It applies to winds from any source.

WindDiscardLevel: This parameter sets a wind speed above which inputs from an external weather control program, using the FS98 interface (***not*** the Advanced or New Weather Interfaces) are ignored. The default for this value is 400 knots. If a weather control program tries to set a wind speed above this, it is ignored and the previously set speed for this wind layer is retained. (This parameter is provided specifically to prevent problems occurring with programs using corrupted data from an Internet download or other problems). Set this parameter to 0 to disable this check altogether.

WindLimitLevel: This parameter sets a limit on the wind speed which can be accepted from an external weather control program, using the FS98 interface (***not*** the Advanced or New Weather Interfaces). The default for this value is 200 knots. If a weather control program tries to set a wind speed above this (but below the “WindDiscardLevel” above), it is ignored and 200 knots is set instead. Set this parameter to 0 to disable this check altogether.

WindShearSharp: [not FS2004] Set to ‘No’ to make FSUIPC set the Wind Shear to the default (minimum) setting. FSUIPC normally sets this to "Sharp" to avoid horrible spurious winds occurring during the transition, apparently an FS2000 bug (which may or may not be fixed in FS2002). [Note: if “WindTransitions=Yes” there are no wind layer transitions seen by FS2000/2002, so this parameter then does nothing].

UpperWindGusts (*): Set to ‘Yes’ to make FSUIPC copy the upper wind gust information provided by the weather control program. These are normally suppressed by FSUIPC because upper winds aren't gusty, and FS's gusts can seem pretty wild anyway. *Note: this parameter is not operational if **SuppressAllgusts** has been enabled.* On FS2004 this is only applied to global weather and all externally supplied weather.

SuppressAllGusts: Set this to 'Yes' if you feel that FS's simulation of wind gusts is unrealistic, and cannot be corrected by the adjustments in the FS CFG suggested elsewhere. If this is set to "Yes" then the **UpperWindGusts** parameter is ineffective. On FS2004 this is only applied to global weather and all externally supplied weather.

GustsRelative: Set to 'No' to set gust velocities as the maximum gust speed, which is what FS2000 *should* be using according to the way the Winds dialogue works. The default setting (Yes) makes FSUIPC set gust speeds to the *difference* between the upper gust speed and the normal wind speed. This is to get around an apparent bug in FS2000 where it seems to add gust speeds to the wind speed rather than treat them as a maximum. *(See the hints in the main User Guide on getting FS2000 gusts working better).* The parameter is not used on FS2002 or later. The correct setting for FS2002 is actually **No** and this is assumed.

WindTurbulence: Set this to 'Yes' to make FSUIPC generate some random turbulence in all wind levels. This will range from none to extreme, but it will normally stay fairly mild. If this is set it overrides any other settings, from FS2000/2002 "real weather" or from an external weather program. It will vary over time as well.

On FS2004 this is only applied to global weather and all externally supplied weather.

SuppressWindTurbulence: Set this to 'Yes' to prevent any wind turbulence. This is mainly intended to help maintain good frame rates in FS2002 even with dense A.I. traffic. There's a similar option for cloud turbulence. On FS2004 this is only applied to global weather and all externally supplied weather**.**

LimitWindVariance: [FS2004 only] This option limits the amount of wind variance (direction changeability) which external programs can set. It is progressive—more variance is allowed for lower wind speeds.

ToggleTaxiWindKey: This allows you to assign a keypress which, when used, will swap the current surface layer wind speed and gust setting with a wind speed of 1 knot and no gusts (or, with FS2004's reduced crosswind mode enabled, just reduces the crosswind component). Using the same hot key again will restore the original speed and gust setting. Except on FS2004, if it is used when the current wind is *not* related to the requested surface wind layer then nothing is changed (but a 'beep' may be heard as a warning). On FS2004 it applies to the wind experienced at the aircraft, independently of wind layers.

Note that this is inoperative if ***AutoTaxiWind*** *is enabled*

This feature can be useful to avoid the excessive weather-vaning whilst taxiing. It works with any type of weather applied to FS2000/2002/2004, but only on the lowest wind layer in the first two. However, the **AutoTaxiWind** is probably more suitable for most uses.

The keystroke is defined as in Flight Simulator's own controls, and documented in my FS98 and FS2000 Controls documents (and listed below, in the Button Programming section). For example, I use (and recommend) "CTRL+SHIFT+T" which would be

```
ToggleTaxiWindKey=84,11
```

The same control codes are used in FS2002 and FS2004.

AutoTaxiWind: This feature operates the taxi wind automatically, normally setting it to 1 knot when the plane is on the ground, and allowing the correct wind to transition in (according to the WindSmoothing setting above) after take-off. If this option is enabled the manual taxi wind on/off switching is disabled.

On FS2004, if the reduced cross-wind option is enabled, then the taxi wind is implemented as a reduced cross wind instead of an outright reduction to 1 knot. The cross wind is reduced to near zero at ground speeds below 20 knots, then allowed to increase proportionally to the ground speed–more so on heavier aircraft. In autom,atic mode this applies both on the ground and when airborne but within 500 feet of the ground.

PropTaxiWind: This applies to FS2004 only, and is set to 'Yes' when the reduced cross-wind option is enabled. It is 'No' by default for compatibility with previous versions of FSUIPC.

WindAjustAltitude: (Apologies for mis-spelling). Set to 'Yes' if FSUIPC should add the value specified in **WindAjustAltitudeBy** to all the wind layer boundaries specified by an external weather control program using the FS98 interface (*not* the Advanced or New Weather Interfaces).

WindAjustAltitudeBy: See previous parameter. This is in feet and defaults to 2000.

WindSetVariance: [*Not FS2004*] When this option is enabled it makes FSUIPC convert any wind turbulence into wind "variance". On FS2000 this is done for all wind layers, but in FS2002 it is only applied to upper layers. This feature seems to at least make FS do something (actually it introduces random variations in the wind direction), whilst the turbulence options seem ineffective.

WindVarFactor: [*Not FS2004*] This sets a value from 1 to 20 (default 7) which effectively controls the percentsge effect of wind turbulence on wind variance, when The **WindSetVariance** option is enabled. The default of 7 is rather less that the value I felt was realistic, which would be 10 (equating to 100%). In FS2002 the factor is doubled internally before being applied, as the effect seems rather feeble otherwise.

MagWindsToFST: [*Not FS2004*] If this is set 'Yes' then the wind data supplied to FSTraffic gives the wind direction in degrees Magnetic. Otherwise this is in degrees True, as it would be without FSUIPC running.

UpperWindsToFST: [*Not FS2004*] Set this to tell FSUIPC to send a fixed surface wind direction to FSTraffic when the aircraft is above a specified altitude. This is needed by some tracks for airways. As an example:

```
UpperWindsToFST=270,18000
```

Will cause all surface winds reported to FSTraffic to be from 270 degrees (Mag), once the aircraft is above 18000 feet.

SubterraneanWindFix: This is a facility in FS2002 only to 'fix' the odd winds up to 1000 feet AMSL which occur in the FS2002 downloaded "real weather", even at METAR stations which are at altitudes over 1000 feet. It is defaulted on ('Yes'), because these inaccessible surface winds otherwise cause several other FSUIPC facilities to go wrong—most noticeably the Taxi Wind option.

Visibility

MinimumVisibility: This parameter, which defaults to 0 (meaning it is inactive), is used to prevent *any* weather source setting a visibility below a specified minimum. The value is set in hundredths of a statute mile (i.e. 100 = 1 mile). Note that there may be a short delay (possibly a second) after a new low visibility has been applied before it is detected and corrected by FSUIPC.

On FS2004 this is applied to global weather and all externally supplied weather, and since FSUIPC 3.51, to FS's own weather. However, if it is being imposed on FS's own weather the current visibility will not be reported correctly in weather reports such as those read by external programs and ATIS in FS. This is because the only way of imposing the visibility minimum is by changing the effect at the end stage, the rendering at the aircraft, and not in the weather system as such.

MaximumVisibility: This parameter, which defaults to 2000 (20 miles), is used to prevent *any* weather source setting a surface visibility above a specified maximum when there is any cloud layer with more than 2/8ths cover. The value is set in hundredths of a statute mile (i.e. 100 = 1 mile). Note that there may be a short delay after a new high visibility has been applied before it is detected and corrected by FSUIPC. The parameter is only effective if the value is greater than the MinimumVisibility parameter.

On FS2004 this is applied to all weather, and is independent of the visibility layer too.

MaximumVisibilityFewClouds: This is the same as the previous parameter, except that it gives the maximum to be used when there are no cloud layers of more than 2/8ths cover. It defaults to 6000 (60 miles) The idea is that the extended visibility gives bluer skies by day and more stars by night (but lower frame rates. Sorry, you can't win every way <G>).

On FS2004 this is applied to all weather, and is independent of the visibility layer too.

MaximumVisibilityOvercast: This is the same as the previous parameter, except that it gives the maximum to be used when there is at least one cloud layer of more than 6/8ths cover. It defaults to 2000 (20 miles).

On FS2004 this is applied to all weather, and is independent of the visibility layer too.

MaximumVisibilityRainy: This is the same as the previous parameter, except that it gives the maximum to be used when there is any rain or snow. It defaults to 1000 (10 miles). If it is raining and cloudy the lower of the applicable limits is used.

On FS2004 this is applied to all weather, and is independent of the visibility layer too.

LowerVisAltitude: [*Not FS2004*] When the visibility is set from an FS2000/2002 source, such as its downloaded 'real weather', or by the user setting it through the weather dialogues, there is already an upper altitude, above which global visibility values take over (unless influenced by FSUIPC's graduated visibility facility, described below). However, for visibility controlled by external programs, via the FS98-compatible interface, there is no such altitude so one has to be inserted by FSUIPC. This is specified by LowerVisAltitude in feet, which defaults to 6000.

GraduatedVisibility (*): With this enabled FSUIPC provides a smooth change in visibility from the upper altitude of the surface level visibility to a specified upper visibility at another, specified, upper altitude. The two parameters, UpperVisibility and UpperVisAltitude, control this. The surface visibility extends up to the LowerVisAltitude (above) for visibility controlled by external programs using the FS98 interface, but is controlled by FS2000/2002 for its "real weather" or for visibilities set through the FS2000/2002 dialogues.

On FS2004 this facility is applied to all weathers irrespective of the visibility layer.

UpperVisibility: On FS2000 and FS2002 this parameter, which defaults to 6000 (60 statute miles), is used to prevent *any* weather source setting a visibility above a specified maximum. The value is set in hundredths of a statute mile (i.e. 100 = 1 mile). If GraduatedVisibility is enabled, it is used in conjunction with the next parameter (on FS2004 too).

UpperVisAltitude: This is only used when GraduatedVisibility is enabled, and sets the altitude by which the UpperVisibility should be attained. Above this altitude the visibility stays fixed at this value. The default UpperVisAltitude is 25000 feet.

ExtendMetarMaxVis (*): This checks the visibility being set and adjusts it in three specific circumstances, as follows:

1. If the FS98 program sets it to a value between 99.95 and 100.04 miles, it is reset to 6.20 miles. This is in order to rectify the results from any programs that take the 9999 metre maximum METAR visibility and transmit it literally as a number of 1/100ths of statute miles.
2. If the value is then in the range 6.15 to 6.24 miles (i.e. close to the 9999 metres maximum of a metric METAR), it is adjusted to a random value between 6.20 miles and the current maximum value (which will either be the MaximumVisibility parameter value, or 150 miles).
3. If the value is between 9.95 and 10.05 miles (i.e. close to the 10 statute mile maximum of a U.S. METAR), then it is adjusted to a random value from 10 miles to the current maximum (which will either be the MaximumVisibility parameter value, or 150 miles).

Note that the random addition is computed only once every five minutes, to avoid constant changes in visibility should the weather control program re-write the value from time to time.

On FS2004 this is only applied to global weather and all externally supplied weather.

SmoothVisibility and **VisibilitySmoothness** control the option to smooth visibility changes from external programs. The former parameter switches the option on or off (default "No"), and the second sets the number of seconds of FS elapsed time for each 10% change in the visibility range (default 2). On FS2004 this applies to all weather, and is independent of the visibility layer.

VisSmoothingDelay:
VisSmoothAirborneOnly: These two, for FS2004 only, control the timing of the all-weather visibility smoothing action, if it is enabled. The "delay" value is in seconds, and delays the start of smoothing after any "clear all weather" action (whether by loading a Flight, using the FS weather menus, using the FSUIPC clear weather facility, or by specific external program action). The option to smooth only when airborne allows the visibility at the airport to be changed and set as desired, before take off.

SetVisUpperAlt and **VisUpperAltLimit** are for FS2004 only, and control the option to impose an upper limit on the FS2004 visibility layer, when the weather is being set from the global values or by an external program. The default is 6000 feet, and this is set when zero occurs here.

Clouds and Precipitation

These rain and storm facilities are for FS2000 and FS2002 only:

GenerateRain (*): Set this option to ‘Yes’ to allow FSUIPC to provide semi-random rain/snow generation, assuming the external weather program is not controlling this. For rain or snow FSUIPC requires 3 or more oktas of cloud (1 okta if it is a thunder cloud) and a cloudbase at no more than 3000’ AGL.

RainStarter: controls the probability of rain or snow starting. This check occurs every minute or so. The default is 75 (out of 100). A value of 100 guarantees rain starting, providing the cloud is suitable as described above.

RainStopper: controls the probability of rain or snow stopping. This check occurs every minute or so. The default is 75 (out of 100). A value of 100 guarantees rain stopping.

StormsAutomatic (*): Leave as ‘No’ to allow suitably programmed weather control programs to use all three FS98 cloud layers for any types of cloud. With this option set to ‘Yes’ the "thunderstorm" layer can only be used for storms, as it was in FS98.

StormProbability: A value from 0 to 100 representing a percentage probability of a storm. For a storm to be generated the winds and clouds must also be adequate—as defined in **StormParameters** below. This is checked every two minutes. The same probability is used to determine when a storm dissipates, after its minimum duration.

StormParameters: This should be used with care. The value provided is used to determine what conditions must prevail before a ‘random’ storm is even considered. It is used as follows:

StormParameters=WWCBBHD

All 7 characters are decimal,

WW	Minimum surface wind speed needed (in knots). Default is 10.
C	Minimum cloud cover (1–8, default 3).
BB	Maximum cloud base AGL, in thousands of feet (default 05, i.e. 5000 feet).
H	Minimum cloud thickness, in thousands of feet (default 3, i.e. 3000 feet).
D	Minimum duration of storm, in minutes, with 0 meaning 10 (default 0, i.e. 10 minutes)

Regardless of the value of D, the duration may be extended at random, with a probability then of it ending being the same as that of it starting. Of course if the cloud or wind conditions change the storm may end much earlier than the specified minimum.

The default parameter (applicable even if it not shown in the .ini file) is therefore:

StormParameters=1030530

StormMinTemp: This is an additional Storm Parameter, and sets the minimum *surface* air temperature at which the random storms will be allowed to occur. Default is 10 (Celsius), range –99 to 99.

Most of the other cloud facilities do apply to externally-supplied weather for FS2004:

GenerateCirrus (*): Set to ‘No’ to stop the occasional extra cirrus layer being added automatically by FSUIPC. Set to ‘Force’ to make FSUIPC add the occasional cirrus layer even if an external weather control program turns the option off.

CloudforJetTrails: For FS2000 and FS2002, set to ‘Yes’ to make FSUIPC often add a 1/8th cover cumulus layer, high (but below the added cirrus). This is in order to allow Jet Trails to be produced by FSClouds 2000.

CloudForVSky: For FS2002 only, set to ‘Yes’ to generate a top layer of overcast cirrus, for FS Sky World SE’s “virtual sky. The minimum altitude is then given by **MinVSkyAltitude** in feet.

OneCloudLayer: This defaults to ‘No’. Set it to ‘Yes’ to prevent there ever being more than one layer of clouds. This may help get better performance on slower machines. It won’t help much on faster machines.

ThinClouds, **ThinThunderClouds:** These default to ‘No’. Set to ‘Yes’ to prevent any single cloud layer being thicker than 1000 (or 10000) feet (or whatever is set in the **CloudThinness** (or **ThunderCloudThinness**) parameter, below), from the nominal cloud base to its top (not including any variations which may be set). This may help get better performance on slower machines. It won’t help much on faster machines, but it could be used to get more realistic cloud thicknesses if weather programs generate them too thick. Note that if the thunder cloud one is not enabled, the other applies to *all* cloud layers.

CloudThinness, **ThunderCloudThinness** allow the limits applied by the **ThinClouds** and **ThinThunderClouds** options to be changed. The defaults are 1000 and 10000 feet. The range accepted in both cases is 100–59999 (feet).

CloudTurbulence: Set this to 'Yes' to make FSUIPC generate some random turbulence in all cloud layers. This will range from none to extreme, but it will normally stay fairly mild. If this is set it overrides any other settings, from FS2000 "real weather" or from an external weather program. It will vary over time as well.

SuppressCloudTurbulence: Set this to 'Yes' to prevent any cloud turbulence from any source. This is mainly intended to help maintain good frame rates in FS2002 even with dense A.I. traffic. There's a similar option for wind turbulence.

CloudTurbulenceToWinds: This is for FS2000 global weather only, and is an alternative way of dealing with the frame rate hit with dense A.I. traffic. If the option is selected, cloud turbulence is removed and instead emulated by wind turbulence when flying in the cloud layer. This helps frame rates while outside the cloud layer, but not whilst within it.

CloudIcing: Set this to 'Yes' to make FSUIPC generate some random icing in clouds. This will range from none to extreme, but it will normally stay fairly mild. If this is set it overrides any other settings, from FS2000 "real weather" or from an external weather program. It will vary over time as well.

MaxIce: For FS2004 only (where the icing effects seem to be substantially increased over FS2002). The value here is 0–4 to limit the icing to that level (0 = No icing, 4 = Any icing), with the default set to 3 (just preventing "severe" icing, level 4). If the option is actually disabled the value is retained by saved as a negative number. In this case –1 represents 0 but disabled, to –5 representing 4 disabled. Note that in FS2004 only the global FS weather and inputs from weather programs can be so limited. Furthermore FS's own global weather is not changed unless the appropriate technical option is enabled.

MaxIce: Also for FS2004 only. The value here is 0–4 to ensure that icing is never below a given level (<=0 = No icing, 4 = Max icing). A value set greater than MaxIce will operate only to make all icing the same as the MaxIce level. Note that in FS2004 only the global FS weather and inputs from weather programs can be so set. Furthermore FS's own global weather is not changed unless the appropriate technical option is enabled.

ApplyVisFix: [*Not FS2004*] By default FSUIPC will attempt to stop the "stuck low visibility" or white-out problem, apparently due to a bug in FS2000's Weather.dll. It does this by checking the effective visibility once a second and trying to progressively correct it if it is lower than it should be when the aircraft is not inside a cloud layer. Note that the detection of cloud layers is not 100% reliably done at present (I'm working on it!), but at least the only bad symptom should be an occasional higher visibility than you'd expect. Note that it isn't proven than this 'fix' works in 100% of cases where a whiteout may occur, but it certainly reduces their frequency by a large amount!

When Microsoft fix the bug causing the problem, simply disable this work-around by setting ApplyVisFix=No. Whether the bug still exists in FS2002 has not been determined at the time of writing.

FixRainProblem: This applies to FS2002 only, and by default it is set to 'Yes'. It tells FSUIPC to take special steps to prevent the 'everlasting rain' problem occurring when using an external weather program, like FSMeteo. If MS does ever fix this FS2002-only bug then you can make FSUIPC a little more efficient by changing this option to 'No'.

KeepFS98CloudCover: When the clouds are set through the FS98 IPC interface (as from SquawkBox, but not FSMeteo), FSUIPC adjects the cover requested for Cumulus type clouds to make them "look right". It adds 2 to the Okta value for coverage below 5. Without this a cover requested as "scattered" (3/8) can look very sparse. If you want the okta coverage in FS to match the setting made by the external program, set this parameter to 'Yes'. (Note that you should not attempt to use FSClouds vapour trails if you do this, otherwise they can appear at the wrong altitude).

CloudTypesFixed: If you are using a weather setting program which tries to set cloud types not supported in FS2004 (resulting often in an eventual crash in Weather.DLL), you can add the parameter **CloudTypesFixed=Yes** to the [General] section of the FSUIPC.INI file. This tells FSUIPC to map all supplied cloud types to one of those known, i.e: 1 (Cirrus), 8 (Stratus), 9 (Cumulus), 10 (Cumulonimbus).

Temperature

CopyDewPtToDayNightVar: [*Not FS2004*] When there are multiple temperature layers, the ATIS reports of FS2002 (and perhaps FS2000) get the Dew Point wrong: they report the Day/Night variation as the Dew Point. To get around this, FSUIPC normally copies the Dew Point to the Variation. The latter is not actually used in FS2000 or FS2002 in any case. If you want to stop this occurring, just set this parameter to ‘No’.

Other options

RemoveATC: This makes FSUIPC apply patches to FS2004’s ATC.DLL module, to forcibly prevent the FS ATC windows from appearing at all, and to prevent any of three different crashes which can occur in ATC.DLL when running FS with third party ATC programs and FS’s ATC turned off. Note that this should *only* be used when you absolutely do not want FS’s ATC to apply to your flights, for instance, when you use only Radar Contact, or possibly VoxATC. It does not prevent the ATC voices, the AI traffic vocal interactions and ATIS read-outs. You may want those as additional chatter in any case, but if not just turn off the ATC sound, or turn it down, in FS’s sound options. This option is available to unregistered FSUIPC users too.

ExternalOptionControl: Set this to ‘No’ if you want to retain control over all the settings for FSUIPC. Normally some of the original options are available for an external weather control program to set according to its needs.

AdvDisplayHotKey: This allows you to assign a key press which, when used, will hide or (if there’s no other reason it is hidden) display the AdvDisplay and/or multliline message window. For this to work in AdvDisplay you need to be using AdvDisplay version 2 or later. The FSUIPC message window is only available in FS2004.

The keystroke is defined as in Flight Simulator’s own controls, and documented in my FS98 and FS2000 Controls documents (and listed below, in the Button Programming section). For example, I use (and recommend) "CTRL+SHIFT+A" which would be

```
AdvDisplayHotKey=65,11
```

The same control codes are used in FS2002/4.

PFCrestartHotKey: This allows you to assign a key press which, when used, will tell the PFC driver (PFC.DLL) to restart all of its serial port activity, including closing and re-opening the port. For this to work you need to be using PFC.DLL version 1.63 or later.

The keystroke is defined as in Flight Simulator’s own controls, and documented in my FS98 and FS2000 Controls documents (still applicable to FS2002/4, and listed below, in the Button Programming section). For example, I use (and recommend) "CTRL+SHIFT+P" which would be

```
PFCrestartHotKey=80,11
```

AllEngHotKey: This allows you to assign a key press which, when used, will re-select all engines on the currently loaded aircraft. It is effectively the same as using the keypress E plus 1, 2, 3, 4 on the main keyboard, depending on the number of engines, but the hot key will work when, apparently, the proper key sequence does not (on three engined aircraft it seems). See the preceding entry for details of how the key is defined.

StopAutoFuel: Set this to ‘Yes’ on FS2002/4 to stop automatic re-fuelling at scenery fuel boxes. With this selected you can only increase fuel via the FS menu or by using a program or gauge which does it via FSUIPC’s offsets.

CorrectVSsign (FS2002/4 only), or **PatchSimApAlt** (FS2000): These options provide one or two ‘improvements’, to the FS autopilot. First, for FS2000, the PatchSimApAlt option patches SIM1.SIM (the main aircraft simulating part of FS) to correct some inaccuracy in the autopilot’s altitude holding capability. The inaccuracy occurs when flying Flight Levels and increases with the difference between the altimeter setting (e.g. the standard pressure setting of 29.92” or 1013mb for Flight Levels) and the actual barometric pressure at sea level (QNH). This action is not applied to FS2002 or FS2004 as it seems to have been fixed in the later simulation engines.

Second, the same option (renamed to ‘CorrectVSsign’ for FS2002/4) corrects the vertical speed setting if it is set to climb when the aircraft would need to descend, or vice versa. It does this by inverting the sign of the vertical speed setting. It only does this when altitude acquire/hold is enabled, so that vertical speed control by itself is not affected. The correction is also not applied if the target altitude is set to a value over 65000 feet—a trick used by some panels to provide V/S controlled ascents and descents.

If neither of these functions are required, set the appropriate parameter to ‘No’. Note that the setting can be overridden for specific aircraft which have specific FSUIPC joystick calibrations by setting this parameter differently in that [JoystickCalibration …] section.

DisconnTrimForAP: When this option is enabled, FSUIPC disconnects the analogue elevator trim axis input to FS whenever either the FS autopilot is engaged in a vertical mode (altitude hold or glideslope acquired), or a program, gauge or module has disconnected the elevator axis via FSUIPC (offset 310A).

Note that the setting can be overridden for specific aircraft which have specific FSUIPC joystick calibrations by setting this parameter differently in that [JoystickCalibration …] section.

ZeroElevForAPAlt: controls the option for FSUIPC to automatically centre the elevator input each time the Autopilot altitude hold mode is changed (switched on or off, including AP engaged changes too). This option can be operated inside FS on FS2004, but if needed on earlier versions must be enabled in the INI file by setting this parameter to “Yes”.

Note that the setting can be overridden for specific aircraft which have specific FSUIPC joystick calibrations by setting this parameter differently in that [JoystickCalibration …] section.

ReversedElevatorTrim: This is probably not of any real use nowadays, as all the axes can be reversed in FSUIPC4’s joystick calibration facilities. Best left set to ‘No’.

Note that the setting can be overridden for specific aircraft which have specific FSUIPC joystick calibrations by setting this parameter differently in that [JoystickCalibration …] section.

PlanLoadNoPosition: This option changes the behaviour of the FS2000/2002 facility to load flight plans into the GPS. If you set “PlanLoadNoPosition” to Yes, the FS2000 plan loader will *not* position the aircraft for you. Many folks prefer this as they like to start the flight with the aircraft parked on the ramp, and taxi to the correct runway themselves. This option is defaulted off (=No) to avoid confusing users already used to and happy with FS2000 working as it does now. Set it to “Yes” if you would prefer the facility not to move the aircraft.

This option is not available in FS2004 as it is not needed—FS2004 provides such an option in any case.

MagicBattery: This reduces the discharge rate on the battery, keeping the voltage from dropping. If this is set ‘Yes’ or 0 then no drop is allowed. If set ‘No’ or 1 then the battery discharges normally. Any value from 2 to 999 acts as a divisor on the discharge rate, so 2 makes the battery last twice as long, and so on. This is designed to assist in getting over the apparent error in the airliners, which makes it discharge far too quickly before engine start.

ExtendedJoyCalib: This simply enables the extra three Joystick calibration pages in the Settings and Options display. The actions carried out are not affected, only whether the settings are shown or not.

N1N2asFS98: Set this option (add the line if it isn’t there) to “yes” if (and only if) you want FS2000 to start up with an FS98 jet aircraft with Engines off. It makes FSUIPC assume that the N1% and N2% values are provided as they are in FS98 (i.e. reversed), rather than as they have been ‘corrected’ in FS2000 for FS2000 aircraft. FSUIPC does do this automatically, but it cannot differentiate the two cases until the engines are running.

This is not really of any use in FS2002, as FS98 aircraft transferred to FS2002 seem either not to work correctly in any case, or to be ‘converted’ to FS2002 standard via parameters generated in the Aircraft.cfg file. It is not provided in FS2004.

AutoTuneADF: This controls an option to ‘auto-tune’ the ADF radio. If this is enabled, when FSUIPC detects no NDB signal being received it alternates the fractional part of the ADF frequency between .0 and .5 every seven seconds or so. This allows external cockpits built with only whole-number ADF radio facilities to be used in areas like the U.K. which have many NDB frequencies ending in .5.

AxisCalibration: This facility deals with inputs to the rudder, aileron and elevator axis offsets, via the FS98 offsets to the IPC interface. These values are subject to a range check, and always scaled down if this range is exceeded. The correct limits are –16383 to +16383.

Additionally, axis inputs can be scaled *upwards* to meet this extent, if required. To do this set:

```
AxisCalibration=Yes
```

By default with this option selected some flattening is applied to the values so that the response is not so vigorous near the centre (0). To calibrate the axes you must move all three controls to their maximum extends on each fresh load of FS2000.

Alternatively, you can set "AxisCalibration=Set". This operates as above, but adds a new section to the .ini file, thus:

```
[AxisCalibration]
Rudder=<max>,<slope>
Elevator=<max>,<slope>
Aileron=<max>,<slope>
```

The <max> values are those which are scaled to 16383, whilst the <slope> values control the amount of flattening in the centre: from 0 (no flattening) to 100 (maximum flattening. note that the flatter the centre, the steeper the sides, so it is always a compromise.

The default "slope" values are 50, 40, 40 respectively, for the three axes.

Once this calibration has been done and the section in the ini file produced (or added manually), there is no need to re-calibrate on each new FS reload. The "AxisCalibration" parameter resets automatically to "Yes".

Note that the "AxisCalibration=No" setting is equivalent to setting "Yes" and adding the section:

```
[AxisCalibration]
Rudder=16383,0
Elevator=16383,0
Aileron=16383,0
```

However, if these values are exceeded during an FS2000 session, the new maxima will replace any values in the ini file.

MainMenu=&Modules: This parameter controls which main (top-level) menu entry in Flight Simulator is used to access the FSUIPC Settings screen. The default, as shown here, is the Modules menu. Note the "&" character, which tells Windows which letter in the name is used for the keyboard accelerator (as in "Alt+M" here).

If you prefer to have FSUIPC accessed through, say, the Flights menu, then you can change this to &Flights. Note that the spelling and "&" characters *must* match whichever menu you are adding to, else a new one will be created instead. Foreign language versions of FS will have differences too, remember.

You can have FSUIPC Settings accessed *directly* from the top level menu if you like. To do this simply choose a unique menu name and add "…" to the end. FSUIPC will take this to mean that you want direct access. For example:

```
MainMenu=FS&UIPC …
```

Will create a top-level menu entry "FSUIPC …" which will lead directly to the Settings window. The accelerator here is U, because the F is already taken (for "Flights")—the Settings window can then be obtained very quickly by just Alt+U.

SubMenu=&FSUIPC …: This supplies the name and the keyboard accelerator character (the one following the "&") which will appear in the selected 'Main Menu' entry (see previous item) and which leads directly to the FSUIPC Settings window. If the Main Menu itself is made to lead directly to the Settings this entry is ignored.

FixWindows: set to 'Yes' prevents cockpit windows resizing and moving. This facility can also be used in FS98 but as there's no in-program options for FS98 you must make sure the panel and scenery windows are exactly as you want them to be before setting this parameter.

SmoothPressure: set to 'Yes' to smooth barometric pressure changes by limiting the changes from external programs to 1 millibar ever so many seconds. The number of seconds is given by **PressureSmoothness** which defaults to 5 and can have any value from 1 to 30, inclusive. On FS2004 this option is only applied to global weather and pressure set by external weather programs.

SetStdBaroKey: This allows you to assign a keypress which, when used, will set the 'Kollsman' window on the Altimeter to the standard pressure, 29.92" or 1013.2mb. This is used when flying 'flight Levels'.

The keystroke is defined as in Flight Simulator's own controls, and documented in my FS98 and FS2000 Controls documents (and listed below, in the Button Programming section). For example, I use (and recommend) "CTRL+SHIFT+B" which would be

```
SetStdBaroKey=66,11
```

The same control codes are used in FS2002.

TCASid: FSUIPC supplies data on the FS2002 additional "Artificially Intelligent" (A.I.) aircraft flying in the neighbourhood, for external TCAS or mapping programs to display. Normally the aircraft is identified by its Airline and Flight number, if there is one, otherwise by the Tail number.

However, other types of identification string can be chosen instead. In particular, the optional labels placed on the aircraft by FS in the scenery view only shows tail numbers, so if you want to match them up you'd want to set this parameter to "Tail". The full list of options here is:

Flight	for airline+flight, or tail number, as available (default)
Tail	for tail numbers only
Type	for the "ATC type", generally only the Make
Title	from the aircraft title (in the .CFG file), truncated to 17 characters
Type+	for the type as above, truncated if necessary, plus the last 3 characters of the tail number
Model	for the model description

The utility "TrafficLook" is supplied—you can see the difference in its display.

TCASrange: Sets the maximum range at which FS2002 A.I. aircraft will be added to the tables for external TCAS applications. This defaults to 40 nm. A value of 0 turns off the limit altogether. This parameter can be adjusted in the Technical page of the FSUIPC Options whilst FS2002 is running.

FixedTCASoptions=Yes can be added by the User if the above two settings are to remain locked, unchangeable except by editing here, in the INI file.

TrafficScanPerFrame: Sets the rate at which FSUIPC scans the AI traffic data for changes. This is a percentage (0–100) per flight simulator frame. The default is 10, which means it will take 10 frames to update all aircraft. You can try higher values if you want to see more fluidity in AI traffic movement, assuming the application itself can scan fast enough. The only penalty from higher values may be a performance hit on Flight Sim, or your application, but many modern PCs may allow even 100% updates per FS frame without measurable degradation. If you set this to 0 you will get no AI aircraft at all, though this will not stop externally injected data (e.g. from AIBridge).

Note that since FSUIPC version 3.51 the proportion of AI traffic processed in each frame will rise if the queue of AI traffic controls (those sent by programs such as Radar Contact, AI Smooth and AI Separation) builds up. This is automatic and is designed to keep those queues down.

TrafficControlDirect: This only applies to FS2004, where is will normally be left to its default value 'Yes'. It tells FSUIPC to send all AI traffic commands directly they are received, rather than queue them up and send them on the next frame-based traffic scan. It results in more effective and efficient control of AI traffic especially in critically busy periods. Only set this parameter to 'No' if it seems to resolve any stability problems on your FS2004 installation.

SetSimSpeedX1: optionally sets a Hot Key which when used resets the simulation rate to x1 (i.e. normal). The keystroke is defined as in Flight Simulator's own controls, and documented in my FS98 and FS2000 Controls documents (and listed below, in the Button Programming section). The same control codes are used in FS2002. As an example, for "CTRL+SHIFT+S" this would be

```
SetSimSpeedX1=83,11
```

ThrottleSyncToggle: sets a Hot Key which operates a facility to make all throttle inputs, for any engine, affect the throttle inputs to all engines. It's a toggle function—if it is on then using it again turns it off. If you are only using a single throttle then this won't make a lot of difference except that *every* time you use toggle it FSUIPC will make the throttle selection (i.e. the keypress E+1 … etc) apply to all engines. The keystroke is defined as in Flight Simulator's own controls, and documented in my FS98 and FS2000 Controls documents (and listed below, in the Button Programming section). The same control codes are used in FS2002 and FS2004. As an example, for "CTRL+SHIFT+E" this would be

```
ThrottleSyncToggle=69,11
```

ThrottleSyncAll: controls whether the Throttle Sync Hot Key operates on the Prop Pitch and Mixture values as well as throttles. This has no effect on jets and helicopters.

FixControlAccel: This, if enabled intercepts all controls, and changes the elapsed time location inside FS before forwarding every *different* (non-axis) control, so that the time elapsed looks large enough for the control not to be accelerated. If it sees successive identical controls then it leaves them, so they can be accelerated as normal. [**This should *not* be used by keyboard flyers!**]

For a fuller explanation, please see the User Guide.

TimeForSelect: [FS2004 only] This specifies the number of seconds for which the SELECT controls (normally assigned to main keyboard keys 1–4) should remain operative for controls that need them (like Engine select, or Aircraft Exit toggle), despite the intervention of other, different, controls. This only operates when the

FixControlAccel option is enabled. To disable this, set the time to 0. Also note that this does not influence the similar automatic facility for the FS pushback, which, on FS2004 only, ensures that the pushback direction remains selectable irrespective of the **FixControlAccel** option.

SpoilerIncrement: This controls the amount the FSUIPC "Spoiler inc" and "Spoiler dec" change the spoiler position on each use. The default is 512, giving 32 steps from spolers lowered (0) and fully deployed (16383).

TrapUserInterrupt: Another option for FS2002 only, defaulted on, this is provided to trap certain "User Interrupt" occurrences, which cause the "End Flight?" dialogue to appear on screen whilst flying. Apparently these can occur in certain configurations if the aircraft is over-stressed or has some minor damage inflicted by, for example, taxiing over rough ground.

NavFreq50KHz: It seems that, in FS2002 for the first time, the NAV radios are tunable to 25KHz frequencies, like the COM radios. Thus the increment/decrement is 25KHz instead of 50KHz. This can cause some difficulty with cockpit designs suited to the current actual 50KHz spacing, so FSUIPC provides this option to force NAV radio frequencies to abide by 50KHz spacing (.00 .05 .10 .15 … .95).

AileronSpikeRemoval
ElevatorSpikeRemoval
RudderSpikeRemoval: These control the options to ignore any aileron/elevator/rudder signals specifying maximum possible deflection. It is mainly useful in conjunction with Wilco's 767PIC on FS2002, which seems to cause these spurious 'spikes' on the elevator occasionally, and on the rudder when flown with the yaw damper switched off.

ClockSync: This facility, applicable only to FS2002 and FS2004, and kindly donated by José Oliveira, compensates for the odd phenomenon of FS losing time. It synchronises the seconds values with that of your PCs system clock. It is defaulted off (=No).

SmoothIAS: This option acts only on the Indicated Air Speed offered to external programs through the IPC interface—in other words, the DWORD at offset 0x0580. It smooths the value by automatically providing a moving average of 23 samples taken at roughly 55 mSec intervals. This appears to overcome the ratcheting effect which can be seen on steep climbs and descents. (*This is enabled by default since version 3.04*).

WhiteMessages: This controls an option to forward external application "Adventure messages" to FS for display in **white** on green instead of **red**. This only applies to non-scrolling messages. If AdvDisplay.dll is also installed and not trapping and diverting the messages anyway, the **white** message option needs AdvDisplay version 2.11 or later.

InitDelay: This controls the timing of FSUIPC's subclassing of the main FS window. This has always defaulted to 3000 (milliseconds) for 3 seconds, but now in FS2004 only it defaults to 0 in an attempt to reduce the probability of black screen problems in FS when switching video modes. [This parameter is not shown in the INI file when defaulted].

WeatherReadInterval: This controls the frequency at which FSUIPC reads the weather in FS2004, in order to populate the many weather variables accessible to applications. The interval actually controls the number of FS2004 frames which elapse between each read, and is given as an exponent of 2. The default value is 4 which means 2^4 or every 16 frames. A value of 0 would update the weather on every frame, and a value of 32 would effectively stop all updating.

Note that this also controls the rate at which any weather is updated using the old FS98 or AWI interfaces. However, it does not alter weather setting capabilities using the New Weather Interface (NWI).

MoveBGLvariables: By default, on FS2004, FSUIPC moves the five BGL user variables (addressed in BGLs by 312 to 31A) from their new location in G3D.DLL back into their old place in GLOBALS.DLL, so they can again be used to interaction between scenery and programs. It is not thought that there are any undesirable consequences of this, but in case there are, this parameter is provided. Set it to 'No' to stop the movement.

UseProfiles: By default this will be set to 'No', for backward compatibility, but set it to 'Yes' if you want to use the User Profile facilities rather than individual aircraft specific assignments and calibrations. The Profile facility has its own chapter in the User Guide.

ShortAircraftNameOk: This is normally set "no" to make sure all aircraft- or profile-specific Keys, Buttons and Joystick Calibration settings only apply to the specific aircraft which was loaded at the time they were assigned. However, if you have several "paints" and which the settings to apply to all, you need to set this parameter to "yes" then shorten the aircraft name either in the [Profiles] section, if you are using profiles, or else in the [Axes.<name>], [Buttons.<name>], [Keys.<name>] and [JoystickCalibration.<name>] section headings in the INI, as needed. The same facility could, for example, give all aircraft starting "Boeing" one set of assignments and all those starting "Airbus" another.

Further, you can set **ShortAircraftNameOk=Substring** to make FSUIPC match the shortened <name> in the INI section headings in *any* part of the full aircraft name, not just at the beginning.

TimeSetMode: In FS2004 FSUIPC intercepts writes to the time locations (for instance, by programs such as FSRealTime) and uses them to issue the appropriate commands to FS to change the time. This makes the new time correctly propagate through the AI system (and may trigger a traffic reload), and avoids subsequent otherwise inexplicable hangs.

Designing this in a way which avoids the problems with AI traffic reloading causing hangs, yet doesn't create too many unwanted pauses for reloading traffic whilst flying, has proved quite a thorny problem. So, a reasonable compromise has been provided as the default solution, but options are also available to allow you to select one of two other modes.

"**TimeSetMode=Partial**" is the default mode. With this set, any time change of more than one minute, or any change at all to the hour, day or year, results in propagation through FS and will therefore trigger at least an AI traffic reload, maybe more.

The other options, selectable *only* by editing the FSUIPC.INI file, are "**TimeSetmode=On"**, where all changes to minutes, hours, day and year are propagated throughout FS, and "**TimeSetMode=Off"**, where no changes to any of the date/time values are propagated at all. This last is how all versions of FSUIPC before 3.465 behaved.

Note that in order to try to prevent AI Traffic loading hangs when instigated by using the FSUIPC traffic density controls, FSUIPC propagates the complete Zulu time and date immediately before reloading the traffic if the new density value is higher than the old value.

ZapSound: This defines the sound to be used when the FSUIPC control for AI traffic deletion (the "Traffic Zapper") is successfully applied. This must be the name of a WAV file in the FS sound folder, the default being 'Firework'.

If you do not want a sound just set it to **ZapSound=None**. However, the reason for the sound is so that you know something has been Zapped. FSUIPC cannot tell what you can see, and the aircraft which is zapped may not be in your display so you may not see it disappear.

ZapAirRange=1.5
ZapGroundRange=0.25

These control the range of operation of the AI aircraft zapping facility. The units are nautical miles. Air and Ground refer to the user aircraft position, not the target. Note that you cannot change the acceptance angle explicitly. It is adjusted automatically, in linear inverse proportion to the change in the range—so with a larger range you would need to point the aircraft nose more accurately.

ZapCylinderAltDiff=n (where n is the maximum altitude difference), can be added to change the mode of the airborne Zapper. With this added, the target for zapping is the nearest aircraft to the airborne user which is within the upright cylinder of radius ZapAirRange and has a difference in altitude of n feet or less, including those on the ground below.

MouseWheelTrim: This records the setting of the 'Use mousewheel as trim' option on the Miscellaneous options tab. By default it is set to 'No'.

AxisInterceptsIfDirect=No: By default, FSUIPC does not intercept FS axis controls which have been assigned in its Axis Assignments to be sent 'direct' to calibration. This would not normally matter either way, as axes assigned directly should have been disabled in FS so no such controls should arrive. However, some add-on aircraft panels (most notably that for the LevelD 767) use some of the standard axis controls to operate the autopilot. If the FSUIPC calibration and slope changes are then applied to the values it would upset the A/P control.

Just in case there are installations which do need both direct and indirect calibration to work on the same axis controls, this bypass can be stopped by changing the **AxisInterceptIfDirect** parameter in the INI file from 'No' to 'Yes'.

Logging facilities

These options can be controlled 'on the fly' from the FSUIPC dialogue window (select the Modules menu them FSUIPC, ALT, M then F). FSUIPC always produces a text file called FSUIPC.LOG in the Modules folder. Entries in the log are timed, from the start of the FS session. The time is in milliseconds and appears on the extreme left of each line.

Please use the logging facilities to check things before reporting problems or omissions in FSUIPC, and supply an appropriate log file (or extract) properly zipped up with such reports.

Note that log files can get very large if all the options are turned on. Keep test flights short. You can read log files whilst flying provided you use a reader which shares access (like recent Notepad programs), or use the 'NewLogKey' described below to close logs and start new ones.

All Log control parameters go into the [General] section of FSUIPC.INI. None are included by default.

LogWeather=Yes: Logs weather data. This will log incoming data, set by a weather control program, on FS98 as well as FS2000–2004. On FS2000–2004 you will also get the actual weather data constructed by FSUIPC in FS terms. Then you get the weather read out by FSUIPC and lastly placed back into the globals for applications to read. Incoming weather control data on the Advanced Weather interface for FS2000–2004, and on the New Weather Interface for FS2004, is also logged in full.

LogWrites=Yes: Logs the global 'writes' received from applications, with global offset address and data size, plus all bytes of data. The offsets shown are the ones used by the application. [Take care: the Log file may get very large!]

LogReads=Yes: Logs the global 'reads' received from applications, with global offset address and data size, plus all bytes of data. The offsets shown are the ones used by the application. [Take care: the Log file may get very large!]

LogEvents=Yes: In FS2004 only, this option logs all FS "key events", other than those from axis controls. This can be very useful to those seeking to understand the actions of their buttons and keys, or to view the sorts of things some of the more complex panels do, repeatedly, every second.

LogAxes=Yes: Also in FS2004 only, this logs just the axis input events.

LogButtonsKeys=Yes: This logs most Keyboard events (KEYUPs only when programmed), and all button operations. The logging can get quite long, but it will be very useful when trying to analyse exactly what your complex FSUIPC button or key programming is doing.

LogExtras=Yes: This logs additional technical data about the inner workings of FSUIPC, the nature of which will vary from time to time according to needs. There is nothing here that would be of interest to the user, but when investigating problems users may be asked to enable it so that the logs returned can be more meaningful in solving them. Do not fly extensively with this option enabled or you will fill up your disk and probably compromise the simulator's performance!

Additional "Extras" logging facilities are available if the parameter **Debug=Please** is incorporated into the INI file. This changes the Extras logging flag into a numeric value that ranges from 0 (off) to 4095. In this range the '1' bit (i.e. any odd number) provides the normal Extras logging, and all others are used for specific debugging or performance measuring log entries which will vary from time to time. This facility is for use under instruction only.

NewLogKey, StopLogKey: These allow you to assign keypresses to close the current Log file (if logging was enabled), and start a new one. The 'NewLogKey' will carry on with the same logging options, whilst the s'StopLogKey' will revert to default logging (the minimum). Between them these two keys give complete control over the logging. (Note that both actions are also available in the FSUIPC dialogue window).

The current log file is always called FSUIPC.LOG. The others are named in numerical order FSUIPC.1.LOG, … 2.LOG, … etc.

The keystrokes are defined as in Flight Simulator's own controls, and documented in my FS98 and FS2000 Controls documents (and listed below, in the Button Programming section). For example, I use "Shft+Ctrl+L" and "Shft+Ctrl+O" (for "Log" and "Off" respectively) which would be

```
NewLogKey=76,11
StopLogKey=79,11
```

The same control codes are used in FS2002 and FS2004.

Monitor facilities

FSUIPC can monitor, on every FS frame, up to four values (or the same values in different formats, if needed), and display or log them when they change. For each value to be logged you enter or select four things:

Base: which will normally be fixed at ‘IPC’. The base is the name of the area of data from which the value shown will be taken. All the variables supported by FSUIPC through the IPC interface are at offsets relative to the IPC base. Only in FSUIPC debug mode, or in specific Beta versions, will other Base values be selectable.

Offset: which identifies the position of the value relative to the Base. This is a hexadecimal number, normally in the range 0000 to FFFF. Some of the non-IPC bases may allow larger offsets. For offsets to standard IPC variables see the Programmers Guide in the SDK.

Type: this defines the type of variable, so that the formatting in the display will show something meaningful. The types currently supported are tabulated below.

Type	Description	C type
S8	Signed 8-bit value, -128 to +127	**signed char**
U8	Unsigned 8-bit value, 0 to 255	**unsigned char**, or **BYTE**
S16	Signed 16-bit (2 byte) value	**short**
U16	Unsigned 16-bit (2-byte) value	**unsigned short**, or **WORD**
S32	Signed 32-bit (4-byte) value	**int**
U32	Unsigned 32-bit (4-byte) value	**unsigned int**, or **DWORD**
SIF16	2 byte Integer & Fraction: 8-bit fraction followed by 8-bit signed integer	Uses a **short**
UIF16	2 byte Integer & Fraction: 8-bit fraction followed by 8-bit unsigned integer	Uses an **unsigned short**
SIF32	4 byte Integer & Fraction: 16-bit fraction followed by 16-bit signed integer	Uses an **int**

Type	Description	C type
UIF32	4 byte Integer & Fraction: 16-bit fraction followed by 16-bit unsigned integer	Uses an **unsigned int**
SIF64	8 byte Integer & Fraction: 32-bit fraction followed by 32-bit signed integer	Uses an **unsigned** then signed **int**
UIF64	8 byte Integer & Fraction: 32-bit fraction followed by 32-bit unsigned integer	Uses two **unsigned int**s
FLT32	32-bit (4-byte) standard floating point value	**Float**
FLT64	64-bit (8-byte) standard floating point value	**Double**
ASCIIZ	A string of single-byte characters terminated by a zero byte. A length an limited number of these is shown	**Char[]**, or **ASCIIZ**
SA16	16-bit signed Angle in FS format (-180 degrees = max+1)	Uses a **short**
UA16	16-bit unsigned Angle in FS format (360 degrees = max+1)	Uses **unsigned short**
SA32	32-bit signed Angle is FS format	Uses **int**
UA32	32-bit unsigned angle in FS format	Uses **unsigned int**

Hex: For most numerical values the sensible display will be decimal. However, for the plain fixed point integer values (S8, U8, S16, U16, S32 and U32) you may want to view them in hexadecimal instead. This is likely for bit-oriented switch or flag collections, and of course the binary-coded decimal (BCD) values such as NAV and COM frequencies. The **Hex** checkbox can be checked for these numeric values only.

Then you have to tell FSUIPC how you want these monitored values to be displayed. There are four options, and any or all of these can be selected:

Normal Log File: Changes in the monitored values are listed in the FSUIPC.LOG for later viewing.

Debug String: The same messages are sent to a debugger or debugging monitor such as DebugView, for viewing in parallel to the FS actions. Note that you may have difficulty running a debugger with SafeDisk-protected versions of FS (FS2000 and FS2004).

AdvDisplay: The monitoring is done by using up to 4 lines in the erstwhile FS adventure display line. The appears near the top of the screen, and will show 4 lines well in FS2004, but not in earlier releases—they all end up on one line. The better alternative is to use my AdvDisplay.DLL module and have the four lines in a window of your sizing and positioning, or even on a separate PC via WideFS and ShowText.

FS Title Bar: The messages replace the FS title altogether. Only one is shown at a time, so this is only useful for monitoring one value.

If the value requested is not available at any time the result will show "<invalid>". When looking at some Engine or other aircraft things, this can happen transiently, for instance whilst an aircraft is being loaded.

Al the monitoring selections are saved in the FSUIPC.INI file, in a section called [Monitor].

JoyNames

The INI file section [JoyNames] is fully described in its own chapter in the User Guide.

Profiles

If you opt to use the Profile facilities, to have different button, key, axis and calibration settings for a number of types of aircraft (rather than specific named aircraft), then FSUIPC4 will create [Profile.<name>] sections in your INI file. These take the name of the profile you request, for example "Jets", "Props", "Helos", and simply contain a list, in the usual 1=<name>, 2=<name> ... format, of those aircraft names which belong to the particular profile, according to your assignments. Those aircraft names may be the full names, as when you assign in the FSUIPC4 options dialogue, or can be shortened or substring names, according to the "ShortAircraftNameOk" parameter already mentioned.

Button Programming

FSUIPC's options dialogue provides a page for programming button in all the main ways. Here we look at how this programming is encoded in the FSUIPC.INI file, and how the programming can be extended to provide multiple keystrokes and controls for a button, mixed if required, and to provide compound (conditional) actions—ones depending on other buttons, switch settings and even previous keyboard presses. There are even facilities to make Button actions depend upon values in offsets from the FSUIPC IPC interface, which really provides a wealth of possibilities (for that part you will need to get the FSUIPC SDK too, as the offset listings are provided in that package, in the Programmer's Guide).

FSUIPC reloads all Button parameters each time the aircraft is changed in Flight Simulator, so you can edit theses and test them out without having to reload Flight Sim every time.

Before embarking on the programming itself, four global parameters need to be described. These won't appear in the INI file unless you add them, and you only need to add them (in the main [Buttons] section) if you need something other than the defaults:

InitialButton: This controls a facility to make FSUIPC perform one-off actions when FS is first loaded and running (i.e. actually ready to fly). This is by programming a real or imaginary Button. Simply add the line "InitialButton=j,b" to the [Buttons] section. The values of j (0–255) and b (0–31) can specify a real joystick and button, or a non-existent one, it doesn't matter. Real ones can have an action assigned on-line, in the Buttons option page, but multiple actions for any button, real or not, can be accomplished by editing the INI file as described here.

IgnoreThese: This can be used to list a number of buttons which are to be ignored by FSUIPC in the Buttons & Switches tab. This is to deal with faulty button signals which are repeating without control and thus preventing the others from being registered on the screen ready to program. The parameter takes this form:

IgnoreThese= j.b, j.b, ...

listing the joystick number (j) and button number (b) of each button to be ignored. To make it easy, you can edit the INI file whilst in the Button assignments dialogue and simply press "reload all buttons" to activate the changes.

Note that the action of ignoring buttons only applies to those numbered 0–31 on each possible joystick (not any "POV" hats), and they are only ignored in the dialogue—if they are already assigned the assignment will still be effective.

EliminateTransients: This can be added, and set to 'Yes', to eliminate short (transient) button press indications. This is intended to help deal with some devices which create occasional spurious button press signals. It operates only with locally-connected joysticks (but not EPIC or GoFlight devices).

Note that enabling this option may mean you have to consciously press buttons for slightly longer. It depends on the **PollInterval** (below). A "transient" button indication is one which only exists for one poll, so a real press would have to last up to 50 mSecs (twice the default poll interval) to be sure of being seen (more, allowing for variations in the polling due to processor/FS activity). You may find you need to adjust the **PollInterval**.

PollEpicButtons=Yes: Set this to No if you experience any difficulty getting FSUIPC to operate correctly on a system with an EPIC installed but which you do not want to program via FSUIPC's "Buttons" page.

ButtonRepeat=20,10: The first number here controls the button repeat rate, when repeating is enabled for a specific button. The range is 1 to 100 and is the number of repeats per second. Note that the higher rates may not actually be achievable. If you want no limit placed, allowing the repeats to go as fast as they can under each circumstance, set this parameter to 0. This can be *very* fast, so beware!

Note that it is unlikely that this rate will be exactly maintained as it is subject to FS performance variations, depending on the action being repeated, but it acts as a good target control value.

The second number gives an initial delay, before repetitions begin. This is in terms of how many potential repetitions to miss, so with 20 repeats per second, 10 would give a delay of half a second. This allows the same button to operate to increment/decrement a value just once, or, by holding the button down, repeat until released.

A value of 0 for the initial delay value means there will be no delay before the repeats start -- this is how FSUIPC has been until the delay facility was added.

PollInterval=25: This parameter tells FSUIPC how often to read ("poll") the joystick buttons. The time is in milliseconds, and the default, as shown , is 25 (40 times a second) for Windows XP, but 50 (20 times per second) for other Windows versions—the difference is to get around problems arising from the earlier USB drivers in those systems..

A polling rate of 0 will stop FSUIPC4 looking at buttons altogether, and in fact this will remove the Buttons & Switches tab from the FSUIPC4 options. This may come in useful for checking whether a rogue joystick driver is causing problems.

A polling rate of 40 per second is more than adequate for all normal button programming. It is only when you come to the more advanced uses that you may want to change this. Rotary switches, for instance, may give pulses so fast that some are missed at such a rate.

Any value from 1 millisecond upwards can be specified, but those from 50 upwards result in a specific number of "ticks" (55 mSecs) being used. i.e. 40-82 actually result in 55 (1 tick), 83-138 in 2 ticks, and so on. Ticks are also approximate, in that they depend on the other activities and loading upon FS.

Values 1–59 milliseconds are actually handled by a separate thread in FSUIPC and give more accurate results, but note that polling the joysticks too frequently may damage FS's performance, and may even make its response to joystick controls more precarious. No truly adverse effects have been noticed during testing on Windows XP, but it is as well to be warned. If you think you need faster button polling, try values in the range 10–25, and make sure that FS is still performing well each time.

NOTE: Do *not* use a polling interval lower than 50 if you are not using Windows XP, as you them may experience odd hangs or other strange effects in Flight Simulator.

Note that PFC's "emulated" joysticks (those with numbers 16 upwards) are polled four times more frequently in any case—this is done because there is no overhead in doing so—there are no calls to Windows but merely some data inspections. GoFlight buttons (joystick numbers even higher) aren't polled at all—FSUIPC receives a call from the GoFlight driver interface (GFDev.DLL) whenever an event occurs.

FORMAT OF BUTTON DEFINITIONS

The button programming is saved in sections in the INI file. For globally operative buttons this is called [Buttons]. For aircraft-specific buttons it is [Buttons.<aircraft name>]. Up to 2048 separate entries defining button actions can be included in each section, normally numbered sequentially from 0, provided that the total of the definitions in the Global section and the largest aircraft-specific section is not greater than 2048.

If the [General] parameter **ShortAircraftNameOk** is set to **Yes** or **Substring**, the <aircraft name> part of the section heading can be abbreviated (manually, by editing the INI file) so that it applies to more than one aircraft. With the 'Yes' option, FSUIPC will automatically select the section with the *longest* match. The ordering of sections in the INI file is not relevant. However, with the 'Substring' option it will select the first section with a substring match – there's no concept of "longest match" in this case.

The basic format of each entry in the Buttons section is as follows:

For keypresses:

<Entry number> = <Action><Joy#>,<Btn#>,K<key>,<shifts>

and for controls:

<Entry number> = <Action><Joy#>,<Btn#>,C<control>,<parameter>

The format of the parameters becomes more complex for conditional actions, so they will be described later.

The <Entry number> is not material most of the time—except in sequences for single button presses/releases. It is just a sequence number from 0–2047 (but limited to a total of 2048 entries for the general section plus any one Aircraft-specific section).

Each entry must have a unique entry number, and the actual order is only important when multiple actions are defined for the same button. FSUIPC will retain the numbering, and hence the order which the number (not the line position) defines.

You can add comments following a semicolon (;) at the end of the line, and these will be retained. You can also insert lines containing only comments, but they need an <Entry number> too, otherwise they may not retain their relative position. Comments can contain up to 63 characters—longer ones will be truncated if and when the [Buttons] section is re-written by FSUIPC.

<Action> is a single letter denoting the action being defined:

P	Pulse the key press or control: i.e. do not hold the keys down whilst the button is held down. This is always the case for controls, and should always be the case for any key presses involving ALT key usage, because once the FS Menu is entered FSUIPC cannot supply further changes like key releases.
H	Hold the specified keys down until the button is released. (This doesn’t apply to Controls and will be treated like P in their case). Do *not* use this with key presses involving ALT, for the reason just given.
R	Repeat the key press or control whilst the button is kept held down. The repeat rate is approximately 6 per second and is not adjustable. Do *not* use this with key presses involving ALT, for the reason already given.
U	Pulse the key press or control when the button is released.

Any button can have a U entry as well as a P, H, or R entry. Provided the button only has one P, H or R, and/or one U entry, and that when it does have two they are either both key presses or both controls, then the button programming can be handled entirely in FSUIPC’s Buttons option page.

The <Joy#> identifies the joystick number (0–15 for normal joysticks, 16 upwards for PFC, GoFlight or other future ‘emulated’ joysticks) as displayed by FSUIPC, and the <Btn#> identifies the specific button (0–39), again as in FSUIPC’s display. Of these buttons 0–31 are regular buttons and 32–39 are the 8 possible POV view angles, starting forward and going clockwise every 45 degrees. (There are no emulated POVs so for joysticks 16 and upwards the buttons numbers are always in the 0–31 range).

Note that the Joystick numbers 0–15 may be replaced be an assigned letter (A–Z, omitting I and O) if the JoyNames facility is being used to assign joysticks indirectly, in case their real ID numbers change.

When buttons on WideFS clients are programmed, the Joystick number also includes a Client PC number—1000 for client 1, 2000 for client 2 and so on. The client numbering is actually handled by WideServer, which keeps a record of Client PC names and assigns them numbers in the WideServer.ini file. You only need to worry about that when changing PCs or renaming them.

For key presses, the <key> value following the letter ‘K’ is the virtual key code for the key to be pressed. Here’s a list for convenience (but note that not all of these will be usable):

0	Null (+ Alt, Shift etc alone)	35	End	49	1 on main keyboard
8	Backspace	36	Home	50	2 on main keyboard
12	NumPad 5 (*NumLock Off*)	37	Left arrow	51	3 on main keyboard
13	Enter	38	Up arrow	52	4 on main keyboard
19	Pause	39	Right arrow	53	5 on main keyboard
20	CapsLock	40	Down arrow	54	6 on main keyboard
27	Escape	44	PrintScreen	55	7 on main keyboard
32	Space bar	45	Insert	56	8 on main keyboard
33	Page Up	46	Delete	57	9 on main keyboard
34	Page Down	48	0 on main keyboard	65	A

66	B
67	C
68	D
69	E
70	F
71	G
72	H
73	I
74	J
75	K
76	L
77	M
78	N
79	O
80	P
81	Q
82	R
83	S
84	T
85	U
86	V
87	W
88	X
89	Y
90	Z
96	NumPad 0 (*NumLock ON*)
97	NumPad 1 (*NumLock ON*)
98	NumPad 2 (*NumLock ON*)
99	NumPad 3 (*NumLock ON*)
100	NumPad 4 (*NumLock ON*)
101	NumPad 5 (*NumLock ON*)
102	NumPad 6 (*NumLock ON*)
103	NumPad 7 (*NumLock ON*)
104	NumPad 8 (*NumLock ON*)
105	NumPad 9 (*NumLock ON*)
106	NumPad *
107	NumPad +
109	NumPad -
110	NumPad .
111	NumPad /
112	F1
113	F2
114	F3
115	F4
116	F5
117	F6
118	F7
119	F8
120	F9
121	F10
122	F11
123	F12
124	F13
125	F14
126	F15
127	F16
128	F17
129	F18
130	F19
131	F20
132	F21
133	F22
134	F23
135	NumPad Enter (or F24?)
144	NumLock
145	ScrollLock
186	; : Key*
187	= + Key*
188	, < Key*
189	- _ Key*
190	. > Key*
191	/ ? Key*
192	' @ Key*
219	[{ Key*
220	\ | Key*
221	] } Key*
222	# ~ Key*
223	` ¬ ¦ Key*

* These keys will vary from keyboard to keyboard. The graphics indicated are those shown on my UK keyboard. It is possible that keys *in the same relative position* on the keyboard will respond similarly, so here is a positional description for those of you without UK keyboards. This list is in left-to-right, top down order, scanning the keyboard:

223	` ¬ ¦	is top left, just left of the main keyboard 1 key
189	- _	is also in the top row, just to the right of the 0 key
187	= +	is to the right of 189
219	[{	is in the 2nd row down, to the right of the alpha keys.
221	]}	is to the right of 219
186	; :	is in the 3rd row down, to the right of the alpha keys.
192	' @	is to the right of 186
222	# ~	is to the right of 192 (tucked in with the Enter key)
220	\ |	is in the 4th row down, to the left of all the alpha keys
188	, <	is also in the 4th row down, to the right of the alpha keys
190	. >	is to the right of 188
191	/ ?	is to the right of 190

The <shifts> value is a combination (add them) of the following values, as needed:

1	Shift
2	Control
4	Tab
8	Normal (add this in anyway)
16	Alt (*take care with this one—it invokes the Menu*)
32	Windows key (left or right)
64	Menu key (the application key, to the right of the right Windows key)

[*Note that the Tab and Alt keys are denoted by opposite bits here than when used for key programming. Apologies for this, which was a design oversight now too late to change*]

If only "normal" is needed, the whole parameter and the preceding comma can be omitted. Usual values are:

9 for shift+ …
10 for control+ …
11 for shift+control+ …

For FS controls the <control> is a number from 65536 upwards, denoting the specific FS control number. Lists of these can be found in my various FS controls documents. In the FSUIPC Buttons page the controls are shown by name normally, but if you want to try a control which has no name but *might* do something useful for you, enter it here, in the INI file. In the Buttons page FSUIPC will show this by number instead of name.

The <parameter> for a control is optional – just omit this along with the preceding comma for most toggle/button type controls. A parameter value of 0 will be assumed anyway.

Either or both of the <control> and <parameter> values can be provided in hexadecimal, preceded by an 'x' character.

As well as the FS controls, a number of additional FSUIPC controls are available. These range from 1000 to 3000, and also values 'xcc00zzzz' (in hexadecimal) which encode the FSUIPC "Offset" controls. See the list below the discussion on 'Keys' for full details.

SEQUENCES, COMBINATIONS, and MIXTURES

The Buttons page in the FSUIPC options is deliberately kept rather simple, hiding some of the programming possibilities. By editing the INI file you can do more:

- Hold one key down whilst pressing another
- Press and release a sequence of keys
- Mix key presses and FS controls in one button operation
- Make button actions conditional on the state of other buttons (see 'Compound' buttons, below)
- Make button actions conditional on values in FSUIPC offsets (see 'Adding offset conditions', below)

The first three are simply done by defining the actions in separate entries, each referring to the same joystick/button number. I'd recommend you first use the Buttons page to get the initial action programmed (this making sure you have the right button number), then close FS and edit the entries already made in the INI file. The only important thing is to number the entries in sequence – preferably, but not necessarily, consecutively.

Examples:

```
16=H1,2,K69,8
17=H1,2,K49,8
```

Presses and holds the 'E' key then presses and holds the '1' key, so both are pressed together. They are both released (in the same order) when the button is released.

```
18=P1,3,K69,8
19=P1,3,K49,8
20=P1,3,K50,8
21=P1,3,K51,8
22=P1,3,K52,8
```

Presses and releases 'E', then '1', '2', '3', and '4' in rapid succession, selecting all Engines.

```
23=P2,3,K76,24
24=P2,3,K65,8
25=P2,3,K69,8
```

Presses and releases ALT+L then A then E, is *very* rapid succession! FSUIPC leaves no delays at all between actions when the ALT key has been used. Otherwise, as soon as it allows the processing of the keys to begin, the ALT key combination will bring up the menu item and (in this case) dialogue, and FSUIPC will not be running and will therefore not be able to provide the key releases. Horrible mix-ups may then ensue! <G>

This last example is a real one I am actually using. The ALT+L gets the Lago menu, the 'A' selects FSAssist, and the 'E' selects the Pushback with Engine Start. This puts you in the pushback dialogue, but then you are into using the mouse, I'm afraid. FSUIPC can help no more.

COMPOUND BUTTON CONDITIONS

Facilities are included to allow you to specify actions for one button which are dependent on the state of another button (or more likely, switch). This by using what I call "Compound" button programming—though it could equally be "Conditional" or "Co-operative". Anyhow, I use the letter C in the definitions, as follows:

```
n=CP(+j2,b2)j,b, ....
n=CU(+j2,b2)j,b, ...
n=CP(–j2,b2)j,b, ...
n=CU(–j2,b2)j,b, ...
```

Here the ‘C’ denotes compound button checking, whilst P = pulse on pressing, U = pulse on releasing, as before. You can also use CR in place of CP for a repeating action—the repeats continue whilst all the conditions are true. There is no facility for the Hold action with the compound facilities.

Inside the parentheses are details of the *secondary* button, which must be in a certain condition for the current button to operate:

> (+j2,b2) means that button b2 on joystick j2 must be pressed ("on") for the current button action (for j,b) to be obeyed.
>
> (–j2,b2) means that button b2 on joystick j2 must be released ("off") for the current button action (for j,b) to be obeyed.

The j,b, ... part is the usual button parameter, for the action of the “current” button which is button b on joystick j.

You can have one condition, as shown above, or two, or more (up to 16 in fact), like this:

```
n=CP(+j2,b2)(+j3,b3)j,b, ....
```

where, now, *both* the parenthesised conditions must be met for the ‘j,b’ button action to result in the defined event.

The conditions can be made to apply *not* to the *current* state of a button, but to the state of a ‘flag’ that is set and cleared by a button (or even a keypress). For every possible “normal” button (16 joysticks x 32 buttons = 512 buttons) FSUIPC maintains a “Flag” (F). Each time any button is pressed (goes from off to on) FSUIPC toggles its flag. This makes the buttons flag a sort of “latching” switch. You can test it in any parenthesised condition by preceding the condition by F, thus:

```
N=CP(F+j2,b2) …
```

This says the rest of this parameter is obeyed if the Flag associated with j2,b2 is set. A condition (F–j2,b2) tests for the Flag being clear. Note that the actual current state of the button j2,b2 is not relevant. All that matters is whether it last left its Flag set or clear.

None, either or both conditions in a multiple-conditioned setting may be on Flags.

These Button Flags can also be set, cleared and toggled by three special FS controls, **Button Flag Set** (C1003), **Button Flag Clear** (C1004), and **Button Flag Toggle** (C1005). In all three cases the Joystick (0–15 *only*) and Button (0–31) referenced is given in the Parameter, by a value calculated as:

256 * J + B (for example, Joystick 15, Button 31 would be 3871).

These three controls are listed in the FSUIPC options drop downs for assignment in both the Buttons and Keys pages, so you can program them there, or here in the INI file. With these themselves as controls resulting for conditional button actions, you can influence conditions for button actions in a whole multitude of ways.

One point to note: since you can use the keyboard or other compound button actions to set, clear or toggle the flags, the actual button for which the Flag is assigned *does not actually need to exist*!

Okay. Now what does this really mean? Some simpler examples will suffice here. I leave it to the more imaginative amongst you to come up with some really complex applications! <G>

First, it means that you can assign multiple uses to any number of buttons by making them conditional on a number of others. For example, a 12-position latching rotary switch could be wired to operate buttons 1 to 12 on joystick 1. Then for any other button I can program 12 different actions. For example, button 0,3 could have twelve different actions assigned, like this:

```
1=CP(+1,1)0,3, ...
2=CP(+1,2)0,3, ...
3=CP(+1,3)0,3, ...
...
12=CP(+1,12)0,3, ...
```

and so on. For example, you may have a set of assignments for ground operations, a set for take-off, a set for climb, a set for cruise, and so on.

Second, to economise sensibly on the use of buttons, where you really need a toggle you can make any button toggle between two actions by using a flag as a condition. For example, suppose your button is Joy 11, button 3, and a spare flag (a button on joysticks 0-15 not otherwise used) is 15, 2. Program your button with three lines in FSUIPC (the numbers on the left need to be sequential with whatever's there already, but I'll assume you have no others so will start with 1):

```
1=P11,3,C1005,3842
```

This says execute Control 1005 whenever your button is pressed. Control 1005 is "Button Flag Toggle". The parameter '3842' identifies the Flag: 256 x joystick 15 + button 2. So, this flag will now alternate between being set and clear each time you press the button.

```
2=CP(F+15,2)11,3, ...
```

This tells FSUIPC what to do if the button is pressed AND the flag is set. Replace the ... part by the Control number and parameter for one of the actions you need.

```
3=CP(F-15,2)11,3, ...
```

Similarly, this tells FSUIPC what to do when the button is pressed and the flag is not set.

Third, you can now program those two-phase type rotary switches, the ones where turning the spindle one way gives pulses on two lines phase shifted one way, and turning the spindle the other way gives the opposite phase relationship.

Say the inputs from the rotary are on Joystick 1, Buttons 1 and 2. When B1 is ON and B2 goes from off to on, then the spindle has turned one way. When B1 in ON and B2 goes from on to off, the spindle has turned the other. That is the simplest example:

```
1=CP(+1,1)1,2, ...        turn direction 1 action
2=CU(+1,1)1,2, ...        turn direction 2 action
```

You can also have double speed action, operating on every off to on and on to off change of B2. Just add two more conditions:

```
3=CP(–1,1)1,2, ...        turn direction 2 action (B2 goes off to on when B1 is off)
4=CU(–1,1)1,2, ...        turn direction 1 action (B2 goes on to off when B1 is off).
```

Since the whole thing is completely symmetric (there is no reason why B1 should control B2, it could also be the other way around), you can actually program it to act on ALL edges of both buttons, by adding another 4 conditions:

```
5=CP(+1,2)1,1, ...        turn direction 2 action (B1 goes off to on when B2 is on)
6=CU(+1,2)1,1, ...        turn direction 1 action (B1 goes on to off when B2 is on)
7=CP(–1,2)1,1, ...        turn direction 1 action (B1 goes off to on when B2 is off)
8=CU(–1,2)1,1, ...        turn direction 2 action (B1 goes on to off when B2 is off)
```

So, you can effectively choose how many pulses you will get for a given turning rate. As you can see, you can get rates of 1x, 2x or 4x—even 3x if you do one part for only half the changes! Note that for reliability at higher speeds you may need to reduce the **PollInterval**.

By the way, it is with some of these rotary switches where the double condition facility can come in very useful. If you have a single rotary of this type with also a push button action available, you can program it to adjust both the units and fractions of, say, a radio receiver. Just use the Flag associated with the button action to choose between one pair of actions or another, thus, supposing 1,3 to be the button:

```
1=CP(F+1,3)(+1,1)1,2, ... increment fraction
2=CU(F+1,3)(+1,1)1,2, ... decrement fraction
3=CP(F–1,3)(+1,1)1,2, ... increment integer
4=CU(F–1,3)(+1,1)1,2, ... decrement integer
```

One last thing. Using several rotaries of this type (that is, with the two signals in different phase relationships to indicate direction of turning), if they are of the type that have both signals ‘off’ in the detent you can save button connections by making one of them (on each one) common. If you do this you can only turn one of them at a time, but this is probably a worthwhile restriction if you are getting short of button connections.

ADDING OFFSET CONDITIONS

As well as all the above (and below, for Keys) any or all entries in all Buttons and Keys sections of FSUIPC.INI can each contain a single condition based on the value of bits, bytes, words or double words in the FSUIPC IPC interface. These values are addressed by an “offset” value in hexadecimal and include just about anything you can think of about what is happening in FS.

Just taking some examples, you can make conditions based on whether the aircraft is airborne or on the ground, whether the engines are running, whether one or more of specific lights are switched on or off, whether the gear is up or down, and even whether there are valid radio signals for NAV1, NAV2, GS, ILS LOC, and so on. The possibilities are endless!

To make good use of this you will need the Programmer's Guide, which lists all of the offsets. This document is in the FSUIPC SDK. You'll find a lot of data in there that you cannot make use of—the conditions here deal with bits or values in 8-bit bytes, 126-bit words and 32-bit "double words". You cannot make use of string values, tables ot floating point values.

You add an offset condition to any Key or Button parameter line in FSUIPC.INI as follows:

```
<sequence number>=<offset condition>  <usual parameter>
```

The space between the new condition and the normal parameter is essential.

A simple example will help. Take this button push parameter, designed to toggle the landing gear when the button is pushed:

```
1=P1,0,C65570,0
```

By adding an offset condition we can stop this doing anything when the aircraft is on the ground:

```
1=W0366=0 P1,0,C65570,0
```

The inserted part, "W0366=0" specify that the Word (16-bit or 2-byte value) at offset 0366 must be zero for this line to be obeyed. Offset 0366 contains 0 when the aircraft is airborne, 1 when it is on the ground.

The format of the condition is:

```
<size><offset><mask><condition>
```

where

<size>	is B for Byte, W for Word or D for Double Word,
<offset>	is the FSUIPC offset, an hexadecimal value between 0000 and FFFF,
<mask>	is optional, and if given selects one of more bits: specify as &x where 'x' is the 8, 16 or 32-bit mask in hexadecimal. The value in the offset is "ANDed" with this mask before being used,
<condition>	is one of: =value for equality !value for inequality <value for less than >value for greater than and the "value" here is *decimal* unless preceded by an x (or X) in which case it is *hexadecimal* like the offset and mask. FSUIPC will output hexadecimal where a mask is used, otherwise decimal. All values are treated as unsigned.

The optional mask facility is useful for testing specific bits, as in the case of the light switches in offset 0D0C or the radio reception details in offset 3300. For example, the offset condition:

```
W3300&0040!0
```

is TRUE when the currently tuned NAV1 is for an ILS.

The <condition> part is optional too, defaulting to !0 when omitted, so this last example could be abbreviated to:

```
W3300&0040
```

For Project Magenta users who sometimes use the default FS autopilot instead one very useful condition is simply:

```
W0500
```

Offset 0500 is non-zero when PM’s MCP is running, zero otherwise, so you can program buttons and keys to operate PM when it is running, but FS otherwise.

Finally, for clever switching you may want to consider using one button to adjust an FSUIPC offset value which then, via offset conditions, selects between a number of alternative button and/or key assignments. To assist in this, offsets 66C0 to 66FF are reserved purely for you to do with as you like. The offset cyclic increment/decrement controls allow, say, a byte value in offset 66C0 to cycle throgh a number of vlues, then each value selects particular actions for defined keys or buttons. The entries in Buttons or Keys might look like this:

```
31=P174,10,Cx510066C0,x00030001
32=B66C0=0 P117,6,C1030,0
33=B66C0=1 P117,6,C1034,0
34=B66C0=2 P117,6,C1038,0
35=B66C0=3 P117,6,C1042,0
```

Here the value in the Byte at offset 66C0 is cycled from 0–3, and back to 0, by button 174,10, and this value, in turn, selects what happens with button 117,6.

These are real examples related to programming of a Go-Flight GF45 unit for different frequency adjustments. Many fuller examples of all this will appear in the documentation for my GFdisplay program, due shortly. GFdisplay brings my support for GF devices to a completion with display handling to complement the button programming in FSUIPC.

ERRORS IN BUTTON PARAMETERS

When the [Buttons] sections are read (or re-read via the “Reload” button in the FSUIPC Buttons page), the lines are thoroughly checked. Any that are syntactically wrong are ignored. However, where a line is ignored, an error message is appended in the form:

… << ERROR n …

The error numbers possible here are listed below. You can then correct the line and press “Reload” again to re-check it. You don’t have to erase the << ERROR … additions. If the line is now okay, that message will be erased for you. If it is still in error a new error number may appear.

The errors are:

1 Offset condition: no hexadecimal offset following the size (B, W or D)
2 Offset condition: the offset is too big (more than 4 hex digits)
3 Offset condition: the ‘&mask’ part has no hexadecimal mask
4 Offset condition: the mask is too big (more than 8 hex digits)
5 Offset condition: condition not recognised (not =, !, <, > or space representing !0)
6 Offset condition: comparison value X for hex, not followed by hex value
7 Offset condition: comparison value X for hex, too big (more than 8 hex digits)
8 Offset condition: no decimal or Xhex value after =, !, < or >.
9 Button operation not specified as H, P, R, U or C
10 Conditional button operation, no P, R or U after the C
11 Too many (…) button conditions
12 Condition joystick number too big
13 Button number omitted in condition (the ,b in (j,b))
14 No matching) found for (condition
15 Button number cannot be > 31 in condition
16 Main button joystick number is too big
17 Main button number is greater than 39
18 Comma (,) missing after main button number
19 The C or K needed for Control or Key is missing
20 Unknown formatting, syntax unintelligible

Keyboard Programming

FSUIPC’s options dialogue provides a page for programming keypresses to assign specific single FS controls. Here we look at how this programming is encoded in the FSUIPC.INI file, and how the programming can be extended to provide multiple controls for a single keystroke combination.

FORMAT OF KEY DEFINITIONS

The key programming is saved in sections in the INI file. For globally operative keys this is called [Keys]. For aircraft-specific buttons it is [Keys.<aircraft name>]. Up to 1024 separate entries defining key actions can be included in each section, normally numbered sequentially from 0, provided that the total of the definitions in the Global section and the largest aircraft-specific section is not greater than 1024.

As with the Button parameters, Key press entries are reloaded each time you change aircraft in Flight Sim, so you can make changes in the INI file and test them without reloading FS.

If the [General] parameter **ShortAircraftNameOk** is set to **Yes** or **Substring**, the <aircraft name> part of the section heading can be abbreviated (manually, by editing the INI file) so that it applies to more than one aircraft. With the ‘Yes’ option, FSUIPC will automatically select the section with the *longest* match. The ordering of sections in the INI file is not relevant. However, with the ‘Substring’ option it will select the first section with a substring match – there’s no concept of “longest match” in this case.

The format of each entry in the Keys section is as follows:

n=key,shifts,control,parameter

for a key press action only, or

n=key,shifts,control1,parameter1,control2,parameter2

for a key with press (1) and release (2) actions.

Here n can run from 0 to 1023 (i.e. maximum 1024 different keystroke actions can be added),

key	virtual keycode, as in the FS CFG file (see list above, in the section about Buttons). Note: If the key press automatic repeats are to be ignored, this code is preceded by the letter ‘**N**’.	
shifts	8	normal
	+1	shift
	+2	control
	+4	alt (not really very useful)
	+16	tab (an added "shift" to give more combinations)
	+32	Windows key (left or right)
	+64	Menu key (the application key, to the right of the right Windows key)
		[*Note that the Tab and Alt keys are denoted by opposite bits here than when used for button programming. Apologies for this, which was a design oversight now too late to change*]
control		FS control number (as in my lists), or special FSUIPC number for additional controls. This can be in decimal, or, preceded by ‘x’ in hexadecimal. The additional FSUIPC controls range from 1000 to 3000, and also values xcc00zzzz in hexadecimal which encode the FSUIPC “Offset” controls. See list below for full details.
parameter		value to go with control, for "SET" types and some special FSUIPC controls. This also is normally in decimal, but can be in hexadecimal preceded by ‘x’.

You can do all of this programming directly in the FSUIPC “Keys” page whilst in FS. In fact it is better to do it there, so you can test it out directly. Note that some of the listed FS controls either do not work, or do not do as you might suppose! And some seem to be mixed up—for instance the “Zoom Out” and “Zoom In” controls appear to be switched, even though the Fine variants of these are okay.

There are two reasons you may want to edit the details in the INI file. The first is to make a single button press operate more than one control. You can specify such actions here, merely by adding the appropriate parameter lines. The controls will be sent in the order of the parameter entries (i.e. the ‘n’ in “n= …”). You can view all these, and delete them, in the Keys page on-line, but you cannot edit any other than the first such assignment for that key press.

The second reason is to add FSUIPC offset conditions. The facilities for making Button presses conditional upon assorted FS internals all apply to Key programming too, and the format and other details are the same as for Buttons. Please refer to the section above entitled “adding Offset Conditions”.

ERRORS IN KEY PARAMETERS

When the [Keys] sections are read (or re-read via the “Reload” button in the FSUIPC Keys page), the lines are thoroughly checked. Any that are syntactically wrong are ignored. However, where a line is ignored, an error message is appended in the form:

… << ERROR n …

The error numbers possible here are listed below. You can then correct the line and press “Reload” again to re-check it. You don’t have to erase the << ERROR … additions. If the line is now okay, that message will be erased for you. If it is still in error a new error number may appear.

The errors are:

1 Offset condition: no hexadecimal offset following the size (B, W or D)
2 Offset condition: the offset is too big (more than 4 hex digits)
3 Offset condition: the ‘&mask’ part has no hexadecimal mask
4 Offset condition: the mask is too big (more than 8 hex digits)
5 Offset condition: condition not recognised (not =, !, <, > or space representing !0)
6 Offset condition: comparison value X for hex, not followed by hex value
7 Offset condition: comparison value X for hex, too big (more than 8 hex digits)
8 Offset condition: no decimal or Xhex value after =, !, < or >.
20 Unknown formatting, syntax unintelligible
21 Virtual key number not in range 1–255
22 No comma (,) after key number
23 No comma (,) after shift code value
24 Bad control value

Additional "FS" Controls added by FSUIPC

All the true FS controls are represented by numbers above 65536. They are listed in my FS-version specific documents called "FSxxxx Controls …". FSUIPC has augmented these with its own set, programmable for both Button and Keys, and these utilise lower numbers, currently in the 1000–3000 range. These are:

1001	PTT on (for Squawkbox 3, Roger Wilco or AVC Advanced Voice Client)
1002	PTT off (for Squawkbox 3, Roger Wilco or AVC Advanced Voice Client)
1003	Set button flag (param = 256*joy + btn, or JjBb)
1004	Clear button flag (param = 256*joy + btn, or JjBb)
1005	Toggle button flag (param = 256*joy + btn, or JjBb)
1006	KeySend to WideClients (param = KeySend number, 1–255)
1007	Autobrake Set (param=0 for RTO, 1=off, 2-5 for 1,2,3,Max)
1008	Traffic Density Set (param = 0–100 %), FS2004 only
1009	Traffic Density Toggle (param = 0–100 %), FS2004 only
1010	Spoiler inc (by 512 or amount set in SpoilerIncrement= INI parameter
1011	Spoiler dec (by 512 or amount set in SpoilerIncrement= INI parameter
1012	Traffic labels set (param selects data in labels, see User Guide), FS2004 only
1013	Traffic labels toggle, FS2004 only
1014	Traffic labels on, FS2004 only
1015	Traffic labels off, FS2004 only
1016	Ap Alt Var Dec Fast (–1000)
1017	Ap Alt Var Inc Fast (+1000)
1018	Ap Mach Var Dec Fast (–.10)
1019	Ap Mach Var Inc Fast (+.10)
1020	Ap Spd Var Dec Fast (–10)
1021	Ap Spd Var Inc Fast (+10)
1022	Ap Vs Var Dec Fast (–1000)
1023	Ap Vs Var Inc Fast (+1000)
1024	Heading Bug Dec Fast (–10)
1025	Heading Bug Inc Fast (+10)
1026	Vor1 Obi Dec Fast (–10)
1027	Vor1 Obi Inc Fast (+10)
1028	Vor2 Obi Dec Fast (–10)
1029	Vor2 Obi Inc Fast (+10)
1030	Com1 use whole inc
1031	Com1 use whole dec
1032	Com1 use frac inc
1033	Com1 use frac dec
1034	Com2 use whole inc
1035	Com2 use whole dec
1036	Com2 use frac inc
1037	Com2 use frac dec
1038	Nav1 use whole inc
1039	Nav1 use whole dec
1040	Nav1 use frac inc
1041	Nav1 use frac dec
1042	Nav2 use whole inc
1043	Nav2 use whole dec
1044	Nav2 use frac inc
1045	Nav2 use frac dec
1046	Adf1 use whole inc
1047	Adf1 use whole dec
1048	Adf1 use frac inc
1049	Adf1 use frac dec
1050	Adf2 use whole inc
1051	Adf2 use whole dec
1052	Adf2 use frac inc
1053	Adf2 use frac dec
1054	Xpndr low NN dec
1055	Xpndr low NN inc
1056	Xpndr high NN dec
1057	Xpndr high NN inc
1058	Freeze pos on
1059	Freeze pos off

1060	Freeze pos toggle
1061	Engine 1 Autostart
1062	Engine 2 Autostart
1063	Engine 3 Autostart
1064	Engine 4 Autostart
1065	Throttles off
1066	Throttles on
1067	Throttles toggle
1068	PVT voice transmit on (for Squawkbox 3.0.4 or later)
1069	PVT voice transmit off (for Squawkbox 3.0.4 or later)
1070	Key Press and Release (param is Keycode + 256*Shift code, or JsBk)
1071	Key Press/Hold (param is Keycode + 256*Shift code, or JsBk)
1072	Key Release (param is Keycode + 256*Shift code, or JsBk)
1073	Advdisplay & FSUIPC display window toggle
1079	Traffic Zapper
1080	Wheel trim toggle (for mousewheel trim adjusting)
1081	Wheel trim faster
1082	Wheel trim slower
1083	Wheel trim speed toggle
1084	Lua Kill All
1085	Traffic Zapall
1930	FSUIPC bank hold off
1931	FSUIPC bank hold on
1932	FSUIPC bank hold set
1933	FSUIPC bank hold toggle
1934	FSUIPC mach hold off
1935	FSUIPC mach hold on
1936	FSUIPC mach hold set
1937	FSUIPC mach hold toggle
1938	FSUIPC pitch hold off
1939	FSUIPC pitch hold on
1940	FSUIPC pitch hold set
1941	FSUIPC pitch hold toggle
1942	FSUIPC speed hold off
1943	FSUIPC speed hold on
1944	FSUIPC speed hold set
1945	FSUIPC speed hold toggle
2010	PM MCP SPD push on B747
2011	PM MCP HDG sel on B747
2012	PM MCP ALT push on B747
2013	–
2014	–
2015	–
2016	–
2017	PM MCP FD2 off
2018	PM MCP FD2 on
2019	PM MCP A/T on
2020	PM MCP A/T off
2021	PM MCP THR mode button
2022	PM MCP SPD mode button
2023	PM MCP Mach/IAS sel
2024	PM MCP FLCH mode button
2025	PM MCP HDG mode button
2026	PM MCP VNAV mode button
2027	PM MCP LNAV mode button
2028	PM MCP LOC mode button
2029	PM MCP APP mode button
2030	PM MCP ALT mode button
2031	PM MCP VS mode button
2032	PM MCP AP1 (L) button
2033	PM MCP AP2 (C) button
2034	–
2035	–
2036	PM MCP AP3 (R) button
2037	PM MCP FD1 off
2038	PM MCP FD1 on
2039	–

2040	PM MCP AP Disc (not 747)
2041	PM MCP AP Eng (not 747)
2042	PM MCP AP Disc (747 only)
2043	–
2044	–
2045	–
2046	–
2047	–
2048	–
2049	PM AB LS button
2050	PM AB STD QNH rel (push)
2051	PM AB STD QNH set (pull)
2052	PM AB SPD button push
2053	PM AB SPD button pull
2054	PM AB HDG button push
2055	PM AB HDG button pull
2056	PM AB ALT button push
2057	PM AB ALT button pull
2058	PM AB VS button push
2059	PM AB VS button pull
2060	PM AB EXPED button
2061	PM AB TRKFPA button
2062	–
2063	–
2064	PM PFD Decision Ht Dec
2065	PM PFD Decision Ht Inc
2066	PM MCP Hdg Dec 1
2067	PM MCP Hdg Inc 1
2068	PM MCP Hdg Dec 10
2069	PM MCP Hdg Inc 10
2070	PM MCP Alt Dec 100
2071	PM MCP Alt Inc 100
2072	PM MCP Alt Dec 1000
2073	PM MCP Alt Inc 1000
2074	PM MCP Spd Dec 1/.01
2075	PM MCP Spd Inc 1/.01
2076	PM MCP Spd Dec 10/.10
2077	PM MCP Spd Inc 10/.10
2078	PM MCP V/S Dec 100
2079	PM MCP V/S Inc 100
2080	PM MCP Crs Dec 1
2081	PM MCP Crs Inc 1
2082	PM QNH Dec 0.01/1
2083	PM QNH Inc 0.01/1
2084	PM ND Range Dec
2085	PM ND Range Inc
2086	PM ND Mode Dec
2087	PM ND Mode Inc
2088	PM ND2 Range Dec
2089	PM ND2 Range Inc
2090	PM ND2 Mode Dec
2091	PM ND2 Mode Inc
2092	–
2093	–
2094	–
2095	–
2096	PM AB ND ILS Mode
2097	PM ND Map Arc Mode
2098	PM ND Map Ctr Mode
2099	PM ND Rose Mode
2100	PM ND Map Plan Mode
2101	PM ND Range 10
2102	PM ND Range 20
2103	PM ND Range 40
2104	PM ND Range 80
2105	PM ND Range 160
2106	PM ND Range 320

2107	PM ND Range 640
2108	PM ND VOR display
2109	PM ND NDB display
2110	PM ND WPT display
2111	PM ND ARPT display
2112	PM ND DATA display
2113	PM ND POS display
2114	PM AB ND VOR1 on
2115	PM AB ND ADF1 on
2116	PM AB ND VORADF1 off
2117	PM AB ND VOR2 on
2118	PM AB ND ADF2 on
2119	PM AB ND VORADF2 off
2120	PM AB ND Metric
2121	PM AB ND HDGVS/TRKFPA
2122	PM AB THR TOGA
2123	PM AB THR FLX/MCT
2124	PM AB THR CLB
2125	PM AB THR IDLE
2126	PM AB THR REV IDLE
2127	PM AB THR MAX REV
2128	PM AB ND2 ILS Mode
2129	PM ND2 Map Arc Mode
2130	PM ND2 Map Ctr Mode
2131	PM ND2 Rose Mode
2132	PM ND2 Map Plan Mode
2133	PM ND2 Range 10
2134	PM ND2 Range 20
2135	PM ND2 Range 40
2136	PM ND2 Range 80
2137	PM ND2 Range 160
2138	PM ND2 Range 320
2139	PM ND2 Range 640
2140	PM ND2 VOR display
2141	PM ND2 NDB display
2142	PM ND2 WPT display
2143	PM ND2 ARPT display
2144	PM ND2 DATA display
2145	PM ND2 POS display
2146	PM AB ND2 VOR1 on
2147	PM AB ND2 ADF1 on
2148	PM AB ND2 VORADF1 off
2149	PM AB ND2 VOR2 on
2150	PM AB ND2 ADF2 on
2151	PM AB ND2 VORADF2 off
2152	PM AB ND2 Metric
2153	PM AB ND2 HDGVS/TRKFPA
2154	–
2155	–
2156	–
2157	–
2158	–
2159	–
2160	PM EICAS Show Controls
2161	PM EICAS Standby Gauge
2162	PM EICAS Page Dec
2163	PM EICAS Page Inc
2164	PM EICAS Synoptic Dec
2165	PM EICAS Synoptic Inc
2166	PM AB ND ILS Mode
2167	PM ND Plan Wpt Dec
2168	PM ND Plan Wpt Inc
2950	PM Elec All Toggle
2951	PM Elec PFD Toggle
2952	PM Elec ND Toggle
2953	PM Elec EICAS Toggle
2955	PM Elec PFD2 Toggle

2956	PM Elec ND2 Toggle
2958	PM Elec Stdby Toggle
2966	PM Elec All ON
2967	PM Elec PFD ON
2968	PM Elec ND ON
2969	PM Elec EICAS ON
2971	PM Elec PFD2 ON
2972	PM Elec ND2 ON
2974	PM Elec Stdby ON
2982	PM Elec All OFF
2983	PM Elec PFD OFF
2984	PM Elec ND OFF
2985	PM Elec EICAS OFF
2987	PM Elec PFD2 OFF
2988	PM Elec ND2 OFF
2990	PM Elec Stdby OFF"

2994	PM Whazzup keys (by Param), see PM offsets list, 542E
2995	PM Quickmap keys (by Param), see PM offsets list, 542C
2996	PM GC keys (by Param), see PM offsets list, 542A
2997	PM CDU keys (by Param), see PM offsets list, 5428

Note: all the “Keys” inputs to PM modules provide efficient ways of directing specific keypresses to them, wherever they may be on the Network. The parameter in these is the keystroke code (see the list earlier in this document) , plus specific PM-defined values for shifts, thus:

256 for Shift, 512 for Ctrl, 1024 for Alt.

You don’t need to worry about changing other bits when two codes are the same—FSUIPC takes care of that automatically.

2998	PM MCP Kcodes (by Param), see Pm offsets list, 04F2

This way of controlling the PM MCP may offer some features not found elsewhere. The parameter is the number used in the Elan Informatique “Knnn” codes normally sent to the MCP via a serial connection. Here is a list of those known at present, but please refer to the PM offsets document for up-to-date information:

10 SPDP (SPD pushbutton 747 MCP,
Speed Intervention on B737 MCP)
11 HDGP (heading SEL pushbutton 747 MCP,
use 25 for HDG HOLD,
use 25 for HDG SEL on the 737)
12 ALTP (ALT pushbutton 747 MCP,
Altitude Intervention on 737 MCP)
17 FDON (switch on First Officer’s FD)
18 FDFF (switch off First Officer’s FD)
19 ATON (switch on)
20 ATFF (switch off)
21 THR
22 SPD
23 MACH
24 FLCH
25 HDG K025
26 VNAV K026
27 LNAV K027
28 LOC K028
29 APP K029
30 ALT K030
31 VS K031
32 AP1 K032
33 AP2 K033
34 CWSA K034
35 CWSB K035
36 AP3 K036
37 FDON K037 (switch on Captain’s FD)
38 FDFF K038 (switch off Captain’s FD)
40 APDI (AP Disengage - not used in 747-400 MCP)
41 APEN (AP Engage - not used in 747-400 MCP)
44 FPV
144 FPV Copilot
45 MTRS
145 MTRS Copilot
46 CTR ND
47 TFC (TCAS)
147 TFC (Copilot TCAS)
48 RST
148 RST Copilot RST
49 STD
149 STD Copilot STD
50 VOR1
51 ADF1
52 OFF1
53 VOR2
54 ADF2
55 OFF2
62 IN
63 HPA
64 setDH
65 setMDA
66 APP ND
67 VOR ND
68 MAP ND
69 PLN ND
70 VOR1 *(double for some mixed AB/Boeing setups)*
71 ADF1
72 OFF1
73 VOR2
74 ADF2
75 OFF2
170 VOR1 *F/O*
171 ADF1 *F/O*
172 OFF1 *F/O*
173 VOR2 *F/O*
174 ADF2 *F/O*
175 OFF2 *F/O*
80 STA
81 WXR
99 DISC (747 disconnect)

2999 Project Magenta GC Controls. Param specifies action, as shown below (list from Project Magenta “Offsets” publication, with permission). [*Add 100 for First Officer GC, else Captain side assumed.*]

Airbus
1 MAP (Captain Side, 101 F/O side)
2 NAV (Captain Side, 102 F/O side)
3 VOR (Captain Side, 103 F/O side)
4 PLAN (Captain Side, 104 F/O side)
5 ILS Mode

Boeing 'Classic Modes'
1 MAP ARC
2 MAP CTR
3 VOR
4 MAP PLAN

New ND Modes (!)
1 MAP
3 VOR
4 PLN
5 APP
6 CTR Pushbutton
7 Force display to 8 Modes (APP/VOR/MAP/PLN)

8 Show Controls in EICAS/ECAM
9 Hide Controls in EICAS/ECAM
10 PFD/ND -> PFD -> ND (like pressing F4,F1,F2 in GC)
11 PFD/EICAS
12 EICAS with Standby
13 EICAS without Standby
14 FPV (Boeing)
15 EICAS/ND
19 Toggle Controls in EICAS/ECAM
20 Incr Engine Page
21 Decr Engine Page
22 Toggle No Smoking
23 Toggle Seatbelts
24 Toggle Overview Page
25 Toggle RMI/HSI display in Boeing-Type ND MAP ARC
26 Metric Toggle

28/29 ND Mode INC/DEC for Airbus

30 Engine Page (Primary) 0
31 Engine Page 1
32 Engine Page 2
..
39 Engine Page 9 (if defined)

40 Range 5 NM
41 Range 10 NM
42 Range 20 NM
43 Range 40 NM
44 Range 80 NM
45 Range 160 NM
46 Range 320 NM
47 Range 640 NM
48 Range DEC
49 Range INC
50 TCAS Off
51 TCAS Alt
52 TCAS Callsign
53 TCAS All

54 Toggle TCAS Off/Alt

55 Show MCP Values in EICAS (Boeing) (Special PFC Display)
56 Hide MCP Values in EICAS (Boeing) (Special PFC Display)
57 PLAN mode next waypoint
58 PLAN mode previous waypoint
60 Show Overview Page in ND
61 Hide Overview Page in ND
62 Set/Reset Timer (AB Glass Cockpit)
72 Toggle WXR
73 VORADFL OFF
74 ADFL ON
75 VORL ON
76 VORADFR OFF
77 ADFR ON
78 VORR ON
80 Terrain Display On
81 Terrain Display Off
82 Toggle Terrain Display
83 Terrain Type Change
84 Terrain Colour/Mode Change
85 Terrain Size Change
86 Terrain 3D
90 STA
91 VOR
92 NDB
93 WPT
94 ARPT
95 DATA
96 POS

321 Decrease Synoptic/System Display Page
322 Increase Synoptic/System Display Page

(Boeing)
Secondary EICAS pages and functions 747 (*777*)
301 ENG
302 STAT
303 ELEC
304 FUEL (*777: HYD*)
305 ECS (*777: FUEL*)
306 HYD (*777: AIR*)
307 DRS (*777: DOORS*)
308 GEAR
309 --- (*777: FCTL*)
310 CANC
311 RCL

(Boeing)
401 Caution On (see 0x4FE)
402 Caution Reset

411 Show FuelUsed Toggle
412 ShowFuelUsed On
413 ShowFuelUsed Of
414 Reset FuelUsed = 0

(Both)
421 Toggle No Smoking
422 No Smoking On
423 No Smoking Off
424 Toggle Seatbelts
425 Seatbelts On
426 Seatbelts OfF

(Airbus)
Secondary EICAS pages and functions AB
301 ENG
302 BLEED
303 PRESS
304 ELEC
305 HYD
306 FUEL
307 APU
308 COND
310 DOOR
311 WHEEL
312 F/CTL
313 ALL
314 CLR
315 STS
316 RCL
317 CLR

70 Show WXR
71 Hide WXR
318 EL/DC (A330/340)
319 C/B (A330/340)

x0100zzzz	Offset Byte Set (offset = zzzz), hexadecimal
x0200zzzz	Offset Word Set (offset = zzzz), hexadecimal
x0300zzzz	Offset Dword Set (offset = zzzz), hexadecimal
x0500zzzz	Offset Byte Setbits (offset = zzzz), hexadecimal
x0600zzzz	Offset Word Setbits (offset = zzzz), hexadecimal
x0700zzzz	Offset Dword Setbits (offset = zzzz), hexadecimal
x0900zzzz	Offset Byte Clrbits (offset = zzzz), hexadecimal
x0A00zzzz	Offset Word Clrbits (offset = zzzz), hexadecimal
x0B00zzzz	Offset Dword Clrbits (offset = zzzz), hexadecimal
x0D00zzzz	Offset Byte Togglebits (offset = zzzz), hexadecimal
x0E00zzzz	Offset Word Togglebits (offset = zzzz), hexadecimal
x0F00zzzz	Offset Dword Togglebits (offset = zzzz), hexadecimal
x1100zzzz	Offset UByte Increment (offset = zzzz), hexadecimal *
x1200zzzz	Offset UWord Increment (offset = zzzz), hexadecimal *
x2100zzzz	Offset UByte Decrement (offset = zzzz), hexadecimal *
x2200zzzz	Offset UWord Decrement (offset = zzzz), hexadecimal *
x3100zzzz	Offset SByte Increment (offset = zzzz), hexadecimal *
x3200zzzz	Offset SWord Increment (offset = zzzz), hexadecimal *
x4100zzzz	Offset SByte Decrement (offset = zzzz), hexadecimal *
x4200zzzz	Offset SWord Decrement (offset = zzzz), hexadecimal *
x5100zzzz	Offset Byte Cyclic Increment (offset = zzzz), hexadecimal *
x5200zzzz	Offset Word Cyclic Increment (offset = zzzz), hexadecimal *
x6100zzzz	Offset Byte Cyclic Decrement (offset = zzzz), hexadecimal *
x6200zzzz	Offset Word Cyclic Decrement (offset = zzzz), hexadecimal *
x7000zzzz	Offset Float32 Set/1000 (offset = zzzz): the parameter is divided by 1000
x7400zzzz	Offset Float64 Set/1000 (offset = zzzz): the parameter is divided by 1000
x7800zzzz	Offset Float32 Inc/1000 (offset = zzzz): the parameter is divided by 1000
x7C00zzzz	Offset Float64 Inc/1000 (offset = zzzz): the parameter is divided by 1000 (For “decrements” use a negative parameter in the increment controls)

* The fixed point increment/decrement values operate on Unsigned (U) or Signed (S) values, and have a parameter with the unsigned or signed limit in the upper 16 bits and the increment/decrement amount (always unsigned) in the lower 16 bits.

Macro Controls

FSUIPC will read any file in the Modules folder which has file type “mcro”. Such files contain definitions of additional controls to be listed and assignable in FSUIPC's Keys, Buttons and Axis Assignments dialogues. All macro files are also re-read and re-installed whenever the Reload button in any of those three dialogues are used.

It is important that the file name (xxxx.mcro) be unique in the first 16 characters, as this will be used as part of the name of the added controls in the drop-downs. Best to keep the names short and to the point—probably the name of the program or program function for which the controls are being added.

Inside a macro file there should be just one section called [Macros]. This must contain definitions of numbered controls, with names also up to 16 characters. These names only have to be unique in that file.

Here is an example, here for a possible Project Magenta glass cockpit ND Mode switch:

```
[Macros]
1=MAP Capt=C2999,1
2=NAV Capt=C2999,2
3=VOR Capt=C2999,3
4=PLN Capt=C2999,4
5=APP Capt=C2999,5
6=CTR Capt=C2999,6
101=MAP F/O=C2999,101
102=NAV F/O=C2999,102
103=VOR F/O=C2999,103
104=PLN F/O=C2999,104
105=APP F/O=C2999,105
106=CTR F/O=C2999,106
```

Note that the numbers on the left do not have to be contiguous, but must be in the range 1–999 inclusive. These will be used internally, and in the FSUIPC INI file, to identify the control within the file.

Supposing the example above occurred in a file called ‘PM GC.mcro’. The names which would then appear, in proper alpha sequence in the FSUIPC drop-downs, would be:

PM GC: APP Capt
PM GC: APP F/O
PM GC: CTR Capt
PM GC: CTR F/O
PM GC: MAP Capt
PM GC: MAP F/O
PM GC: PLN Capt
PM GC: PLN F/O
PM GC: VOR Capt
PM GC: VOR F/O

The value assigned to each control is either another control (*any* FS or FSUIPC-added control, including offset controls and even macro controls—see later), or a Keypress. i.e:

Either:	Cn,p	(control number, parameter, optionally in hex with a preceding x)
Or:	Kk,s	(keycode and shifts).

Both of these are exactly as already defined for Button controls—see the earlier section on Button programming.

Macro Control References

Macro controls are represented internally in the same sort of way as FSUIPC offsets controls, by using high-value bits in the control number. However, the representation in Macro files and in the INI file is as follows:

Mm:n

where m if the Macro File number (see below) and n is the control number from the file, as described above.

Macro file numbers are assigned by FSUIPC when it loads the file. These are remembered in the INI file in a new section [MacroFiles]. For example, in the above case you might get:

```
[MacroFiles]
1=PM GC
```

making “PM GC.mcro” file number 1 for all reference purposes.

It is important to note that different users will have a different selection of macro files in different orders. If they wish to exchange Button assignments they will need to re-assign all macro controls after making their [MacroFiles] sections the same, or at least the same for those files they have in common.

Multiple actions in one macro control

A macro control is not limited to having only one resulting action. If more than one action is required several lines are used in the definition, as follows:

```
n=<name>
n.1=action1
n.2=action2
etc.
```

For an example consider a ‘Menu.mcro’ file containing these definitions:

```
[Macros]
1=Display
1.1=K79,16 ;O
1.2=K69,8 ;E
1.3=K68,8 ;D
```

```
2=FSUIPC
2.1=K77,16 ;Alt M
2.2=K70,8  ;F
```

This adds two controls, ‘Menu: Display’ and ‘Menu: FSUIPC’. The first uses ALT+O E D keystrokes to call up the FS display settings dialogue, the second uses ALT+M F to call up the FSUIPC options.

Note that there’s a limit of 2000 numbered parameters in total in the macro file—so, for instance, 999 macro numbers (1–999, the maximum) with an average of two actions each would be just two shy of the limit. Large files aren’t good in any case as the drop-down list will be full of the added controls all beginning with the same filename. Best to split into functional groups with meaningful filenames, to make the controls easier to locate.

Parameter passing

Normally, and certainly in all the above examples, any parameter set for a Macro Control, when assigned in the Buttons or Keys dialogues, would be discarded as not relevant. However, there is a facility to allow it to be used.

If the parameter part of *any* of the controls defined in the macro is omitted, the parameter value from the *calling* macro is substituted.

As a rather silly example, if you wanted a general PM GC control but not the one named already in FSUIPC, you could define it as

```
7=by param=C2999
```

This would appear in the drop-downs as ‘PM GC:by param’, and the parameter assigned by the user would be used in the C2999 operation. Note that in multiple-line definitions, the same parameter value substitutes for every omitted parameter value.

One interesting consequence of this is the possibility of defining axis controls. To make another silly example, if I define a macro like this:

```
8=Flaps=C66534       ;FS control 66534 is Axis Flaps Set
```

and then assign it to an axis in the Axis assignments dropdown, the axis I've assigned will operate exactly as the Axis Flaps Set axis.

This may not seem so futile when you realise that you can have multiple line mixtures of controls and keypresses also produced by the same Macro. I'm sure there would be wealth of ideas for using this ‘feature’ (which actually fell out of the implementation by accident rather than by design!).

Mouse macros

Another feature of Macro files is their ability to add controls to your armoury which operate switches, dials and other features of FS panels and gauges (mostly add-on ones) which can otherwise only be operated by using the mouse. Furthermore, this facility can actually be used even without recourse to manually preparing the macro files directly—that part is semi-automated via Mouse Macro buttons in both the Buttons and Keys option tabs in FSUIPC. Details of the automatic facilities are provided in the main User Guide. Here we just concentrate on the file itself, the format of the mouse macro lines.

Note that a single Macro file can contain any mixture of mouse and other macros. In fact Mouse, Control and Keypress actions can all be mixed and combined in a single Macro. Of course, this doesn’t happen for the automatically generated macros.

This mouse facility adds the rather obscure format:

```
R<module>:<rect#>,<mouseflag>
```

to those already described for Keys (K), Controls (C) and onward Macro references (M). The ‘R’ here is for mouse **R**ectangle, because it is via specific rectangular areas on screen that FS recognises mouse requests. An ‘M’ for Mouse would have been better, but that’s already used for Macro.

Now I’ll explain what the values in this specification actually mean, but in general no user will actually be concerned with them, as they either have to be supplied be the gauge maker (the add-on panel supplier), or, more usually, be generated automatically for you by FSUIPC, through use of the Mouse itself in mouse macro creation mode.

So, in the mouse action specification:

<module>: is optional. It is a reference to the Gauge or DLL filename, the part of the panel which will be asked to process the Mouse action. It is a numerical reference to another line which must also be present somewhere in the [Macros] section of the .MCRO file, one like this:

```
ModuleN=”name of gauge or DLL”
```

where ‘N’ can run from 1 to 99, or be omitted (so giving “Module=”...”).

If the <module>: part of the mouse action is omitted, the Module being referenced is the one with no number. Otherwise it is simply N:, referring directly to the module.

The **<rect#>** part is the only mandatory part. It is either a reference to the “MouseRect” number in the tables in the Module—as “Rn” referring to the nth rectangle, counting from 0—or a direct reference to the Mouse Function inside the module, as “RXxxxx*xxxx”, where the ‘xxxx’ parts refer to a hexadecimal offset and check-word, respectively. The offset is from the Module’s load address in memory, and the check word are the 16 bits around the mouse function’s entry point: 8 bits before and 8 bits after. The check-word is a safety measure, in case the macro is used on a different version of the same Gauge or DLL.

Finally, the **,<mouseflag>** part provides the actual mouse action required to operate the facility. This is encoded as a number and must be one of the following:

31	MOUSE_RIGHTSINGLE
30	MOUSE_MIDDLESINGLE
29	MOUSE_LEFTSINGLE
28	MOUSE_RIGHTDOUBLE
27	MOUSE_MIDDLEDOUBLE
26	MOUSE_LEFTDOUBLE
21	MOUSE_DOWN_REPEAT
19	MOUSE_RIGHTRELEASE *
18	MOUSE_MIDDLERELEASE *
17	MOUSE_LEFTRELEASE *
14	MOUSE_WHEEL_UP
13	MOUSE_WHEEL_DOWN
11	MOUSE_LEAVE *

Of these, 29 is bar far the most common and is assumed when the parameter is omitted. Note that the values actually equate to the mouse flags by those names in the FS Gauge C/C++ SDK.

Those marked * cannot be generated automatically by FSUIPC as they refer to the mouse buttons being released. However, they may be needed for some switch implementations, and you would need to add them yourself—experimentation is key here. There are examples in the main User Guide.

Of these, 29 is bar far the most common and is assumed when the parameter is omitted. Note that the values actually equate to the mouse flags by those names in the FS Gauge C/C++ SDK.

Just to put all this stuff into context, here are some actual examples. The first is from the FS9 PMDG737NG overhead:

```
Module="PMDG_737NG_Overhead.gau"
1=Batt=RX3170*X8b90
Module1="PMDG_737NG_OHD_APU.GAU"
40=APU=R1:1
```

If these lines are in a loaded MCRO file called “737 OHD” then the Buttons and Keys controls drop-downs would list “737 OHD:Batt”, which would operate the Battery switch, and “737 OHD: APU” which would operate the APU switch. These would only do anything if the overhead gauge is loaded—i.e. the aircraft is in use. Note that the Overhead gauge itself doesn’t have to be visible.

Here is an extract from the Macro file for the add-on gauge/DLL “APchart”:

```
Window="Airport Chart"
Module="APchart.gau"
1=Show/Hide=C66506,10000
...
7=Knob1 Down=R20,14
8=Knob1 Up=R20,13
```

This has a non-Mouse control included to show and hide the AP chart window. That uses the “PANEL ID SET” control with the panel ID number 10000 as parameter (gleaned from the Panel.cfg file). It also has a couple of entries shown which are operated by the mouse wheel.

But note that new parameter

```
Window="window title"
```

This needs to be present when it only makes sense to use the controls with the window both open and visible. This applies to APchart where zooming and moving the chart would be daft without seeing it. You will find Window names for panel parts in the Panel.CFG file. The automatic mouse macro generating facilities in FSUIPC never add a Window parameter, so this may be the one good reason you ever edit a MCRO file.

Axis Assignments

Axis assignments are saved in the [Axes] section, or [Axes.<aircraft name>] for aircraft specific assignments. Generic aircraft assignments can be made using the same parameter and name shortening as for the Buttons and Keyboard sections.

The polling interval can be changed by a parameter

```
PollInterval=10
```

inserted into the main [Axes] section. The units are milliseconds, 10 being the default.

The format of the axis parameters in these sections is as follows:

For the main axis entry (explanation of values below):

```
n=ja,(R)delta(/delay)
```

where the parentheses merely show optional parts, and

j = joystick # (0 to 18, 16 to 18 being PFC)
a = axis (XYZRUV)
R is only present when "Raw" mode is selected
delta is the delta value (eg 512, or 1 for Raw mode)
/delay is an optional delay*, in milliseconds

When axis controls are assigned (the left part of the options), this is extended by the definition of the controls:

```
n=ja,(R)delta(/delay),ForD,ctl1,ctl2,ctl3,ctl4
```

where

ForD is an F for "FS control" or D for "Direct to FSUIPC calibration"

ctl1 to ctl4 are the control numbers, or zero where unassigned. For Direct mode, these are the calibration indices, 1–4 on Page 1 of calibrations, 5–8 on page 2, etc. Numbers 45–48 are the “dual” controls, equating to others depending on whether FS is in flight mode or Slew mode.

> * FSUIPC can apply delays to any axis assigned through its Axis Assignment facilities. The delay is limited to a minimum of 2 x the axis polling interval (which defaults to 10 mSecs) and a maximum of 200 x this interval (i.e. 2 seconds with the default polling interval).
>
> Delays for axes have to be edited in the INI file. There is no facility to change them or even see them in the option screens. Delays of 200 mSecs or more should be reasonably accurately maintained most of the time, but short ones could vary quite a bit, the smaller you set them, because of the granularity of the polling interval and the sharing of the processor with other things going on in FS.

Here's an example of an axis assigned to the FSUIPC Spoiler, with a 1 second delay:

```
0=0Y,256/1000,D,22,0,0,0
```

If the axis is programmed to send controls based on the axis passing through zones (the right side of the options), there will also be entries for each such assignment, thus:

```
n=ja,UDorB(R),low,high,ctl,param
```

where UDorB is U for Up, D for Down or B for Both
R optionally specifies Repeat
low and high give the axis values for the zone

ctl and param are the Control numbers, and Parameter where used.

Here's an example for a Gear lever:

```
1=0Z,256/500
2=0Z,U,6400,16383,66079,0
3=0Z,D,-16384,-13783,66080,0
```

Note that the delay option (here half a second) still goes on the main axis entry, the one defining the delta (and "Raw" mode if applicable).

You can edit the INI file whilst FS is running, then simply going to the Axis Assignment options page and clicking the reload button at the bottom of the window.

Programs: facilities to load and run additional programs

FSUIPC can, as an extra, cause other programs to be run each time you load and run Flight simulator. Details of what programs to be run are provided in an additional section in the FSUIPC.INI file. This section cannot be edited in the on-line FSUIPC options dialogues. You need to either edit the details directly in the INI file, or use the excellent utility program “Run Options” provided separately by José Oliveira (you need the version of Run Options dated November 2002 to use the new ‘CLOSE’ option).

The additional section is

```
[Programs]
```

and can contain up to 16 requests to run other programs—up to 8 “Run” parameters Run1 to Run8, and up to 8 “RunIf” parameters, RunIf1 to RunIf8. Both sets are otherwise identical in format. The only difference is that the RunIf programs are not run if they appear to be already running. The ordinary “Run” programs will be loaded without such checking.

The format is simply:

```
        RunN=(Options,)<full pathname of program to be run>
or      RunIfN=(Options,)<full pathname of program to be run>
```

where N runs from 1 to 8. Details of options are given below, but if none are required the parameter simplifies into just the full pathname.

For example: `Run1=D:\RadarContact\RCV3.exe`

might be used to run Radar Contact version 3.

If the program needs command-line parameters these can be included by enclosing the whole value in quotes, so that the space(s) needed don't cause problems. You may also need to include the quotes if the pathname includes spaces.

For example:

```
Run2="c:\epic\loadepic fs98jet"
```

The programs are loaded in order of the run number, 1–8. If a mixture of Run and RunIf parameters are given, the order is Run1, RunIf1, Run2, RunIf2, and so on.

The Options you can use are as follows:

HIDE	tries to get the program to hide itself when it runs. This is only possible if the program defines its window to use default settings, so it isn’t very useful for many programs, unfortunately.
HIGH	runs the program at higher priority than FS. ***Use with care!*** Messing about with priorities doesn’t work well in all circumstances, and in particular FS2002 doesn’t seem to like it much.
CLOSE	closes the program tidily (if possible) when FS is terminated.
KILL	forcibly terminates the program, if possible, when FS is terminated.
LOW	runs the program at IDLE priority. Depending on what the program does, this may actually effectively stop it until you direct user focus to it, as FS tends to soak up all Idle time.

READY	delays loading and running the program until FS is up and ready to fly, and FSUIPC can supply valid data through its IPC interface. (This parameter may, of course, result in the programs being run in a different order to that specified by the Run number).

Of these really only CLOSE, KILL and READY are of general use. If you want to apply more than one option, list them separated by commas, but *no spaces*. For example:

```
RunIf1=READY,KILL,D:\FS2002\WeatherSet.exe
```

Assignment of FLAPS_SET control (for FS2002 *only*)

The Flaps calibration facility in FSUIPC cannot be used directly in FS2002, because the convenient FLAPS_SET cannot be assigned to an Axis in FS2002.CFG. This omission was corrected in FS2004, and in the latter you can even assign a Flaps Axis in FS's Options–Controls–Assignments dialogue.

To get around the problem in FS2002, you can select any one of the following Axis controls (obviously one you are not otherwise using!), assign it (by name) to your Axis in FS2002.CFG, tell FSUIPC to use this by declaring its numeric value, as about to be explained, *then* calibrate it in FSUIPC's Joystick section (as the FLAPS control, on page 6).

The AXIS controls at your disposal are listed below. Use the chosen name in FS2002.CFG and the relevant number in a new parameter in the [JoystickCalibration] section of FSUIPC.INI, thus:

```
FlapsSetControl=<control number>
```

This is set to 0 to disable the Flaps Set interception.

Valid Axis Controls (N.B. Not all tested. Please advise if you find any which don't work in FS2002):

AXIS_AILERONS_SET	65763
AXIS_ELEV_TRIM_SET	65766
AXIS_ELEVATOR_SET	65762
AXIS_LEFT_BRAKE_SET	66387
AXIS_MIXTURE_SET	66292
AXIS_MIXTURE1_SET	66422
AXIS_MIXTURE2_SET	66425
AXIS_MIXTURE3_SET	66428
AXIS_MIXTURE4_SET	66431
AXIS_PAN_HEADING	66504
AXIS_PAN_PITCH	66503
AXIS_PAN_TILT	66505
AXIS_PROPELLER_SET	66291
AXIS_PROPELLER1_SET	66421
AXIS_PROPELLER2_SET	66424
AXIS_PROPELLER3_SET	66427
AXIS_PROPELLER4_SET	66430
AXIS_RIGHT_BRAKE_SET	66388
AXIS_RUDDER_SET	65764
AXIS_SLEW_AHEAD_SET	65867
AXIS_SLEW_ALT_SET	65870
AXIS_SLEW_BANK_SET	65871
AXIS_SLEW_HEADING_SET	65869
AXIS_SLEW_PITCH_SET	65872
AXIS_SLEW_SIDEWAYS_SET	65868
AXIS_SPOILER_SET	66382
AXIS_THROTTLE_SET	65765
AXIS_THROTTLE1_SET	66420
AXIS_THROTTLE2_SET	66423
AXIS_THROTTLE3_SET	66426
AXIS_THROTTLE4_SET	66429

Assignment of additional controls

(Reverser, Aileron and Rudder Trims, and Cowl Flaps)

There are no axis controls provided in FS for jet thrust reversing nor for aileron or rudder trim or even for setting the cowl flaps. To get around this, you can select any FS Axis control (one you are not otherwise using!), and assign it to your Axis in FS's assignments dialogue. Then you need to tell FSUIPC which one to use this by declaring its numeric value, as about to be explained, and calibrating it in FSUIPC's Joystick section (on page 7 or 8).

Most of the AXIS controls at your disposal are listed above with their numeric equivalent. Others can be found in my FS Controls Lists which you can find on www.schiratti.com/dowson (separate ones for FS2000, FS2002 and FS2004). Use the relevant number in a new parameter in the [JoystickCalibration] section of FSUIPC.INI, thus:

```
ReverserControl=<control number>
AileronTrimControl=<control number>
RudderTrimControl=<control number>
CowlFlaps1Control=<control number>
CowlFlaps2Control=<control number>
CowlFlaps3Control=<control number>
CowlFlaps4Control=<control number>
```

These are set to 0 to disable the interception altogether, but FSUIPC assigns the AXIS_MIXTURE_SET control (number 66292) to the Reverser by default. There is one other parameter for the reverser:

```
MaxThrottleForReverser=0
```

This controls the interlock—the reverser will not engage until all throttles are reduced to this setting (normally 0, or idle). You can try a non-zero value here if you cannot calibrate your throttles to produce a stable idle zero.

Multiple Joysticks

Method 1:

On FS2000–2004, using the Joystick sections of the FSUIPC dialogue to calibrate the main flight controls, FSUIPC can also accept up to four different control inputs for each main flight control, treating them equally. You can have up to 4 aileron, elevator, rudder, throttle, left and right brake controls. FSUIPC takes the value from the input giving *maximum* deflection from 'neutral' or 'idle'. There's no averaging, or other types of conflict resolution, taking place.

You have to somehow *connect* up your multiple joystick axes, whether by using an EPIC card, multiple Game Ports, or multiple USB devices. FSUIPC cannot help there. Having done that, you need to find 'spare' FS controls which you will not otherwise be using from joystick inputs (see the lists in my FS2000 Controls documents)—it doesn't matter if you will be using those controls from the keyboard. FSUIPC only pinches the joystick inputs. You have to assign the additional joystick axes, wherever they may be, to these "spare" controls.

Now add to the FSUIPC.INI file's JoystickCalibration section (add the section if necessary) a list of declarations which define the additional controls you have assigned. You define these by *number*. The main flight controls are defined by parameters like this:

```
AileronB=<control number>
ElevatorB=<control number>
RudderB=<control number>
```

Other parameters here can define LeftBrakeB, RightBrakeB, ThrottleB, and also C and D versions of all 6 controls, so providing up to 4 copies of each one.

Note that you will need to calibrate all controls so that the ones controlling the same values are as close as possible in range and response. Do this first in Windows Control Panel, then, after making the above adjustments and assignments, in FSUIPC. Calibrate dead zones at the ends (and in the centre for aileron, elevator and rudder) to "cover up" any discrepancies—in other words, calibrate for the worst of each.

Method 2:

An easier method is now available, provided you use the FSUIPC Axis Assignments facility to assign your controls, deleting them from FS assignments.

FSUIPC's axis assignments allows any of your joystick axes to be assigned to any of FS's or FSUIPC's axis controls, and there's no restrictions on how many you can assign to any of them. So that's the first problem solved – you can assign two sets of yokes, rudders, whatever, to the same controls.

Both FSUIPC and FS take notice of the last movement in an axis. They don't "poll" them to get regular inputs, but only see changes coming from them. So both will see the last change from multiple axes. However, that might be from an unwanted jitter or small accidental movement. So, *provided you assign your axes for Direct FSUIPC Calibration* (as opposed to an FS control), FSUIPC now arbitrates, selecting the axis with the highest deflection (defined here as a difference from zero).

Note, however, that it still only sees axes when they change, so even if one axis is held at an large deflection, once another axis for the same control moves to a similar or higher position, that takes control then even if it moves lower than the held on—the latter is effectively "out of it" until it is moved.

The hints about calibration in Method 1 still apply.

HELICOPTER PITCH and BANK TRIM facilities

A facility to operate pitch and bank trims on helicopters is provided. This uses the normal FS elevator and aileron trim controls (and axes) to modify the end value on the “Y” (elevator) and “X” (aileron) axis of the cyclic. To use this you need to ensure that the axes are calibrated through FSUIPC (as the elevator and aileron axes respectively), and add

ApplyHeloTrim=Both

to the relevant [JoystickCalibration …] section(s) in FSUIPC.INI. Note that, as a precaution, the trim value will never be added to the relevant axis if the normal trim value is non-zero.

This new “helo trim” values are maintained in IPC offsets as follows:

0BBE	2 bytes	16-bit Helo Pitch Trim value, range –16383 to +16383
0C06	2 bytes	16-bit Helo Bank Trim value, range –16383 to +16383

Both of these can be written to for external program control.

Note that if you only require a pitch trim you can set

ApplyHeloTrim=Yes

Instead of ‘both’. The aileron/bank axis and trim values will then be left alone.

Message Filters

Messages sent to FSUIPC for display on the FS screen can be filtered and forcibly routed according to their first few characters. This is done by adding a new section to the FSUIPC.INI file, as follows:

```
[MessageFilters]
Suppress=...
SingleLine=...
MultiLine= ...
```

The “...” part is replaced by a list of up to 8 strings (in "quotes"), each of less than 16 characters. Messages sent to FSUIPC are compared with these. If they start with the same characters (case ignored) then the action taken is as follows:

Suppress: the message is discarded
SingleLine: the message is treated as a single line message even if it isn't
Multiline: the message is treated as a multiline message even if it isn't.

For example:

```
SingleLine="FDC","PM MCP"
```

will route messages beginning "FDC" or "PM MCP" to the single line window, unless such messages are suppressed by FSUIPC option.

Facility for multiple INI installations (FS2000–FS2004)

Different FSUIPC.ini files can be used for differing FS requirements, even loading from the same FS2000/2002 installation. This involves using multiple FS2000.CFG, FS2002.CFG or FS9.CFG files with different filenames, with the following section added in each one:

```
[FSUIPC]
ControlName=<name>
```

Then you load FS for each configuration with the command line parameter specifying the CFG file, thus, for FS2000 for example:

```
FS2000.exe /CFG:<filename>.CFG
```

And this will allow FSUIPC to identify its correct .INI file, <name>.ini.

IMPORTANT: For FS9 CFG files, please put your alternative CFG files into the main FS9 folder, *not* the "Documents and Settings" place where the default FS9.CFG goes. If you put it where it seems logical, FS9 will not see it and will create a new default CFG file with the name you specified, placing it in the default FS9 folder!

You FSUIPC.KEY file will need duplicating with your new name too, i.e. <name>.KEY. You can of course have different Keys in each, though the name/address details will have to be the same as they are cross-checked. The Log files will also use this <name>, not just FSUIPC.log etc.

The main use of this feature is so that a PC can be used in two or more modes with *one* FS installation, for example:

- As a WidevieW "slave" with the appropriate default Flight loaded by FS (to place it into slew mode with the correct view) and the correct FSUIPC options set for allowing WidevieW to copy the weather correctly, and:
- With different FS cfg and FSUIPC ini files to run FS in normal 'local control' mode with all normal options.

Appendix: About the *Aircraft Specific* option and "*ShortAircraftNameOK*"

Note: this is a contribution from a user, to whom thanks is expressed.

There are these three choices in FSUIPC settings:

ShortAircraftNameOK=No

ShortAircraftNameOK=Yes

ShortAircraftNameOK=Substring

Result: To get exactly the same settings for AXES, BUTTONS, KEYS and CALIBRATION for each plane repaint or variant.

The Short Aircraft Name in FSUIPC refers to the name in the Aircraft.cfg file under "title"

For example: Aerosoft DHC Beaver. There might be 7 variants or repaints

aircraft.cfg \(flightsim.X)\title= Aerosoft Beaver DHC-2A 55-0682
aircraft.cfg \(flightsim.X)\title=DHC-2A C-GSKY Beaver
aircraft.cfg \(flightsim.X)\title= Aerosoft DHC-2A C-GSKY modern
aircraft.cfg \(flightsim.X)\title=Beaver DHC-2A DQ-GEE
aircraft.cfg \(flightsim.X)\title=DHC-2A DQ-GEE modern
aircraft.cfg \(flightsim.X)\title= Aerosoft DHC-2A N299EE
aircraft.cfg \(flightsim.X)\title=Beaver Aerosoft DHC-2A N299EE modern

Edit the FSUIPC.ini file:

Scenario 1: If "ShortAircraftNameOK=No"

Presuming that you have already assigned the axes, keys and buttons and calibrated the joystick for one of the above variants or repaints: in order to get the same settings for the rest of the above variants/repaints of the Aerosoft Beaver you would need to edit the FSUIPC.ini file and add 4 separate entries for each title name (exactly as above) under [Axes], [Buttons], [Keys], [Joystick Calibration] to ensure that all of the settings were exactly the same, ie 28 entries in all. Pretty tedious in fact— I had over 40 variants/repaints of this plane so I would have need 160 entries in the FSUIPC.ini file.

[Axes. Aerosoft Beaver DHC-2A 55-068] [Buttons. Aerosoft Beaver DHC-2A 55-068] [Keys. Aerosoft Beaver DHC-2A 55-068] [JoystickCalibration.Aerosoft Beaver DHC-2A 55-068]
[Axes. DHC-2A C-GSKY Beaver] [Buttons. DHC-2A C-GSKY Beaver] [Keys. DHC-2A C-GSKY Beaver] [JoystickCalibration.DHC-2A C-GSKY Beaver]
[Axes. Aerosoft DHC-2A C-GSKY modern] [Buttons. Aerosoft DHC-2A C-GSKY modern] [Keys. Aerosoft DHC-2A C-GSKY modern] [JoystickCalibration.Aerosoft DHC-2A C-GSKY modern]
[Axes. Beaver DHC-2A DQ-GEE] [Buttons. Beaver DHC-2A DQ-GEE] [Keys. Beaver DHC-2A DQ-GEE] [JoystickCalibration.Beaver DHC-2A DQ-GEE]
[Axes. DHC-2A DQ-GEE modern]

[Buttons. DHC-2A DQ-GEE modern] [Keys. DHC-2A DQ-GEE modern] [JoystickCalibration.DHC-2A DQ-GEE modern]
[Axes. Aerosoft DHC-2A N299EE] [Buttons. Aerosoft DHC-2A N299EE] [Keys. Aerosoft DHC-2A N299EE] [JoystickCalibration. Aerosoft DHC-2A N299EE]]
[Axes. Beaver Aerosoft DHC-2A N299EE modern] [Buttons. Beaver AerosoftDHC-2A N299EE modern] [Keys. Beaver Aerosoft DHC-2A N299EE modern] [JoystickCalibration.Beaver Aerosoft DHC-2A N299EE modern]

Scenario 2: If "ShortAircraftNameOK=YES"

12 entries would be required to make sure all settings were the same

[Axes. Aerosoft	[Axes. DHC]	[Axes. Beaver]
[Buttons. Aerosoft	[Buttons. DHC]	[Buttons. Beaver]
[Keys. Aerosoft	[Keys.DHC]	[Keys.Beaver]
[JoystickCalibration.Aerosoft]	[JoystickCalibration.DHC]	[JoystickCalibration.Beaver]

Explanation:

1. "Aerosoft" would pick all those entries in the title STARTING with "AEROSOFT", but NOT Aerosoft in any other part of the title.
2. "DHC" would pick all those entries in the title STARTING with "DHC" but not those with "DHC" in any other part of the title
3. "Beaver" would pick all those entries in the title STARTING with "Beaver" but not those with "Beaver" in any other part of the title

Scenario 3: If "ShortAircraftNameOK=Substring"

4 entries only, i.e. "DHC" in the FSUIPC.ini file would result in all variants having exactly the same settings – "DHC" is common to all titles.

[Axes. DHC] [Buttons. DHC] [Keys. DHC] [JoystickCalibration.DHC]

To summarise:

ShortAircraftNameOK=No	**One entry for each different title in the aircraft.cfg file**
ShortAircraftNameOK=Yes	**Picks up the starting part of the title in the aircraft.cfg file**
ShortAircraftNameOK=Substring	**Picks up any part of the title in the aircraft.cfg file**

Title in aircraft.cfg file	ShortAircraftNameOK=		
	No	Yes	Substring

title=Airbus A321
title=Airbus A321 Paint2
title=Airbus A321 Paint4
title=Airbus A321 Paint5
title=Boeing 737-400
title=Boeing 737-400 Paint1
title=Boeing 737-400 Paint2
title=Boeing 737-400 Paint3
title=Boeing 737-400 Paint4
title=Boeing 747-400
title=Boeing 747-400 Paint1
title=Boeing 747-400 Paint2
title=Boeing 747-400 Paint3
title=Boeing 777-300
title=Boeing 777-300 Paint1
title=Boeing 777-300 Paint2
title=Boeing 777-300 Paint 3
	Separate entry for each title	**“Airbus”: Would apply to all entries starting with Airbus.**
		“Boeing” would apply to all entries starting with Boeing.
			“A321”: Any variant with A321 in the title.
			“Paint” Any variant with PAINT in the title.
			“737”: Any variant with 737 in the title.

Explanation: **ShortAircraftNameOK=Substring** Any text that is in any position in the “title” located in the aircraft.cfg file that is inserted in the ini file as above will result in the same settings for those aircraft. For instance choosing “737” ie **[Axes.737]** etc would result in all planes with 737 in the title having the same settings. Likewise choosing “Boeing” would cover all variants/repaints with Boeing in the title

To summarise if you had 20 variants/models/repaints with all different titles you would need 20 entries per section (80 in all) in the ini file. Using **ShortAircraftNameOK=Substring** you could cut this back to just 1 entry per section (4 in total).