FSUIPC for Programmers

By Pete Dowson, Pete Dowson's Support Forum

[**Please also see the FSUIPC4 Offsets Status document for use with FSX]**

This document has been updated for Version 3.99 and later of FSUIPC3.

First, a word of clarification for programmers wanting to write applications to interface to FS2000, FS2002 or FS2004: FSUIPC isn't a DLL you can use directly: it is an FS DLL—i.e. it is part of the FS process, not part of yours. It provides data through a complicated process called "Inter-Process Communication", hence the name (IPC).

The interface it uses is not mine. It was invented by Adam Szofran in his FS6IPC module, several years ago. However, you should find all the assistance and information you need to use it in the FSUIPC Developer Kit, of which this document is now a part. The data you need has been split into separate ZIPs: for C/C++ programmers, for Delphi programmers (contributed by Pelle Liljendal with some extra work by Chris Brett), for Visual Basic programmers (contributed by Chris Brett with some extra work by Enrico Schiratti)—even one for ASM programmers, using MASM32 (contributed by Andrea Brunori). Many thanks to Andrea, Pelle, Chris, Enrico, and Alan Dyer (for the Borland C++ example) for making this Kit so much more useful for so many more programmers.

The offsets for positional data within FS98's GLOBALs are published in some of files you can find on the Internet, but I have tried to collate all the information I know about these here. Please refer to the table later in this document. If you are not sure of any of this, or really want to "see" what is going on, you could try using Pelle Liljendal's excellent "FSInterrogate2" package to investigate variables yourself (this package is included with this Kit). I have supplied a project file for FSInterrogate2 called "FSUIPC.fsi" which basically contains the same information as in both the tables that appear later as well as updates specifically for FSX.

Access to FS2000, FS2002 and FS2004 Variables

Since version 1.94 of FSUIPC, a large number of the additional variables provided by FS2000 and documented in Microsoft's Panels SDK (plus some that aren't) can be accessed by using a special range of new "offsets" supported by extended mapping in FSUIPC, in the range 2000 – 3FFF hex. Some, but by no means all, of these are also mapped for access in FS2002 and (fewer still) in FS2004.

There is a Table later in this document listing these variables, by the same names as used in the Microsoft toolkit. No other information is available for these at present, but note that there are some absurdities. This is not a fault in the mapping within FSUIPC, but in the allocations of these named variables all to a value always obtained from one place in FS. I believe that these values, like many of the others, are never actually set to anything useful, so it probably doesn't matter.

Note that all of these are supported for both reading and writing, but I haven't the foggiest idea what would happen if they are written to—such writes could have an effect or not, and that effect could be quite drastic, like crashing FS.

The odd offsets chosen are very specifically designed to make the mapping to the actual locations in FS2000 efficient. For example, the original values in the 2xxx range are actually in the SIM1.SIM module (SIM1.DLL in FS2002), and most (but not all) of those in the 3xxx range are in the PANELS.DLL module, but the addresses are encoded in a way allowing me to calculate the pointers quickly. They are not really "offsets" as such. Unfortunately the same efficiency has not always carried forward to FS2002 and even less to FS2004. Such is life. It is a good job processors are getting faster! <G>

There are some important omissions. These are the token variables which do not appear to exist in specific locations, but which are calculated or derived in some way by procedure calls. I have not mapped these as regular access to such routines would be very inefficient, and it would be impossible to do an equivalent "write" for each "read" possibility. If specific items are needed from those which are omitted I would have to consider them on an individual basis, as there is no generic solution.

FS2002/4 A.I. Traffic Data (for TCAS applications and similar)

Applications interfacing to FSUIPC can obtain data on the AI traffic generated in FS2002 and FS2004. This is suitable for a mapping or TCAS implementation.

The offset area from D000 to FFFF inclusive is reserved for this. It is nearly all read-only.

There are four tables, two main ones at E000 to EFFF for ground aircraft and F000 to FFFF for airborne aircraft, plus two smaller additional ones for FS2004 only at D000 and D800 respectively. There are 'N' slots available for up to 'N' A.I. aircraft in each table. At present N is equal to 96, but this may change if more or less data is found to be required. The N slots start at offsets E080, D040 (ground) and F080, D840 (airborne) respectively. The 128 bytes before the main tables are used for housekeeping, the 64 bytes before the FS2004 add-on tables are currently reserved.

Each main table slot is 40 bytes in length, and contains data as follows (C terms used):

```
typedef struct _TCAS_DATA
{   DWORD id; // 0 = empty, otherwise this is an FS-generated ID.
            (Do not use this for anything other than checking if the slot is empty or used—it may be re-
            used for other things at a later date).
    float lat; // 32-bit float, in degrees, –ve = South
    float lon; // 32-bit float, in degrees, –ve = West
    float alt; // 32-bit float, in feet
    WORD hdg;       // 16-bits. Heading. Usual 360 degrees == 65536 format.
                    // Note that this is degrees TRUE, not MAG
    WORD gs;        // 16-bits. Knots Ground Speed
    short vs; // 16-bits, signed feet per minute V/S
    char idATC[15]; // Zero terminated string identifying the aircraft. By default this is:
            // Airline & Flight Number, or Tail number
            // For Tail number, if more than 14 chars you get the *LAST* 14
            // Airline name is truncated to allow whole flight number to be included
    BYTE bState; // Zero in FS2002, a status indication in FS2004—see list below.
    WORD com1;      // the COM1 frequency set in the AI aircraft's radio. (0Xaabb as in 1aa.bb)
            // NOTE that since FSUIPC 3.60, in FS2004 this is set to 0x9999 whilst the aircraft
            // is in "SLEW" mode rather than normal flight mode.
} TCAS_DATA;
```

Note that the format above, with the COM1 frequency, came into operation with FSUIPC version 2.861.

The "bState" value is new for FS2004. When non-zero it gives the current status of the AI aircraft, as follows:

0x80	128	Initialising	0x8A	138	Taking off
0x81	129	Sleeping	0x8B	139	Departing
0x82	130	Filing flight plan	0x8C	140	Enroute
0x83	131	Obtaining clearance	0x8D	141	In the pattern
0x84	132	Pushback (back?)	0x8E	142	Landing
0x85	133	Pushback (turn?)	0x8F	143	Rolling out
0x86	134	Starting up	0x90	144	Going around
0x87	135	Preparing to taxi	0x91	145	Taxiing in
0x88	136	Taxiing out	0x92	146	Shutting down
0x89	137	Take off (prep/wait?)			

Each FS2004 additional table slot is 20 bytes in length, and contains data as follows (C terms used):

```
typedef struct _TCAS_DATA2
{   DWORD key; // This is the flight identifier Key (usually negative, as in 0xFFFFxxxx) which can be
        used to identify the flight in the Traffic file. It is not guaranteed to be unique except in one traffic
        File. [Note: this is not available in FSX and is used differently!]
    WORD file; // This is a file Id, to distinguish between multiple traffic files. The default FS2004 Traffic
        File has an Id of 14. [Note: this is not available in FSX!]
    short sPitch; // Aircraft pitch in degrees * 65536 / 360 (Also available in FS2002)
    char chICAO[4]; // Departure airport ICAO Identifier
    char chICAO[4]; // Arrival airport ICAO identifier
    BYTE runway; // 0 if not assigned for take-off or landing. Else 1-36, or one of 37=N, 38=NE, 39=E,
        40=SE, 41=S, 42=SW, 43=W, 44=NW
    BYTE runway_des; // 0 or runway designator: 1=L, 2=R, 3=C, 4=W (water)
    short sBank; // Aircraft bank in degrees * 65536 / 360 (Also available in FS2002)
} TCAS_DATA2;
```

Note that this data came into operation with FSUIPC version 3.113 (pich and bank 3.471). On FS versions before FS2004 this area will be all zero exceptin the pitch and bank. Each 20 byte slot in the table starting at D040 is related to the 40 byte slot in the corresponding position in the table starting at E080. Similar considerations apply to D840 and F080. Consequently, if you have the offset of a desired main TCAS slot, the additional slot offset is obtained as follows:

```
DWORD Add_Offset = 0xD000 + ((Main_Offset – 0xE000) / 2);
```

But do not try to do it the other way, as the data in the additional tables is not guaranteed to be valid for any invalid or unused main slot.

There is an option in the FSUIPC.INI file [General] section to select the contents for the ATC id string. For example, you can restrict it to tail numbers only, so they match those displayed in FS2002's windows. This option is:

```
TCASid=<option>
```

Where <option> is one of these:

Flight	Gives airline+flight number, or tail number if no airline (this is the default)
Tail	Gives only tail numbers
Type	Gives the ATC type (up to first 15 characters)
Type+	Gives up to 11 characters of the ATC type, followed by the last 3 digits of the Tail number
Title	Gives up to 15 characters from the aircraft Title (in the AIRCRAFT.CFG file)
Model	Gives the ATC model (up to first 15 characters)

The important parts of the housekeeping area are as follows:

E000 or F000	WORD	this gives the size of each slot (currently 40)
E002 or F002	WORD	maximum number of slots which will be used (N=96)
E004 or F004	WORD	number of slots used so far (keeps increasing, never decreases)
E006 or F006	WORD	changes count: incremented every time *any* slot is changed
E008 or F008	BYTE	slotChanges[]: an array of N bytes, each one being incremented when relevant slot is changed
E068 or F068	BYTE[8]	option settings (separately for Ground E068, Airborne F068). See below.
E07E or F07E	WORD	the FSUIPC offset for the slot with the nearest aircraft to the user aircraft (on the ground E07E, airborne F07E).

You can use the two fields indicating changes to reduce demands across the FSUIPC interface, re-reading only slots which have changed. However, if your application normally runs under WideFS this isn't really much of a saving.

Note that the data is not updated in strict real time, but on a scanning basis, to avoid any adverse affects on FS performance. The default scan is 10% of all AI aircraft (whether ground or airborne) on each FS frame, but this can be adjusted using the TrafficScanPerFrame parameter in the FSUIPC.INI file. If you are providing a real-time airport movement display like that provided by TrafficBoard you may want your users to increase this to 100% but tell them watch out for FS performance degradation. 100% seems okay on reasonably fast PCs.

Only aircraft within a certain range of the user aircraft are included. For airborne aircraft this range can be set by the user (or by program), and can be unlimited but defaults to 40nm. For ground aircraft the range is 3 nm if the user aircraft is on the ground, 6 nm if the user aircraft is airborne, but both can be temporarily changed by program.

Once the slots are all used, additional aircraft will be range-checked. If they are nearer than others in the table then the furthest ones will be removed to make room. There is an option for inactive (sleeping or initialising) ground aircraft to be treated as more distant than active aircraft so that they get removed from the table first.

Options settings

The 8 bytes at offsets E068 and F068 independently contain the current option settings for Ground (E068) and Airborne (F068) aircraft. They are used as follows:

Byte 0 — Range in nm (0 = unlimited). For ground, this is the range when the user aircraft is airborne. Defaults are 40nm (airborne), 6nm (Ground)

Byte 1 — Range in nm (0 = unlimited) for Ground aircraft only, when the User aircraft is also on the ground. Default is 3 nm.

Byte 2 — The TCASid option setting, thus:

0 = Tail number
1 = Airline + Flight number
2 = Type
3 = Title
4 = Type + last 3 digits or tail number
5 = Model

Byte 3 — = 0 normally, giving preference to nearer aircraft when the table is full

≠ 0 to give preference to active aircraft [FS2004 and later]. This is only applicable to ground traffic, and an aircraft is considered inactive if it is in states 80 or 81 (initialising or sleeping).

Bytes 4–7 — Reserved.

Normally most of these options will be as set by the user via the FSUIPC options dialogue or INI file. Applications can change them by writing to these bytes, independently for ground and airborne traffic. However, FSUIPC will automatically re-instate the user's settings in approximately 20 seconds after the last write to any one of these bytes (airborne or ground). If an application wants to continue with changed settings it must re-write that changed setting at regular intervals. I would suggest using an interval of no more than 5 seconds in order to allow for delays when Networking is being used or FS is under other loads.

Detecting runways in use

An extra facility is provided on FS2004 which allows applications a better chance of detecting the runways in use at any selected airport in range (i.e. within 85nm or so of the user aircraft). The Weatherset2 program provided with FSUIPC makes use of this to show any runways currently assigned when AI traffic is active at a weather station selected by ICAO code.

This is the interface for this:

D000	32-bit signature (see below)
D004	4 character ICAO of airport
D008	32-bit timestamp
D00C	4 bytes giving up to 2 departure runways, format: Number (1 byte), Designator (1 byte)
D010	4 bytes giving up to 2 arrival runways, format: Number (1byte), Designator (1 byte)

Runway numbers: 1–36 plus 37=N, 38=Ne, 39=E, 40=Se, 41=S, 42=Sw, 43=W, 44=Nw
Designators: 0=none, 1=L, 2=R, 3=C, 4=W

Procedure:

1. Write your signature value (generated by your program, to prevent simultaneous access by others), and the ICAO at the same time. If you use separate writes, write the ICAO first, but use one FSUIPC_Process call.
2. Read the timestamp. This is best done in the same FSUIPC_Process call as the writes.

3. Read the ICAO, timestamp and 8 bytes of runway details until the timestamp changes (or until you time-out). Then check that the ICAO you read is the one you want. If so, then the runway bytes are either zero (if there aren't any known) or they are filled in for you.

4. Write zero to the signature to free the interface for others. If you don't do this, FSUIPC will clear it in any case within about 12-15 seconds of action 1 above.

Notes:

The runways are gleaned from the data detailed earlier, in the new tables at D040 and D840, but FSUIPC is here looking through ALL the traffic, i.e. all traffic within FS's own 80–90nm radius. It is not restricted it by the user-set radius, nor the smaller ground limit.

However, and this is an important point, it only scans them for one iteration. When it sees your request it starts looking next time is restarts a complete scan, and finishes next time it needs to start a scan. This has two implications:

(a) If there doesn't happen to be any flights with allocated runways at this time, you won't get any listed. FSUIPC does not build up a 'memory' of runways allocated.

(b) The time taken between asking for the data and getting it will vary. It may be up to two scan times (because it only starts looking at the start of a scan), *plus* the normal timing overheads you might get, say, with process switching and WideFS use. Note that one complete scan can take many FS frames—this is dependent upon the percentage traffic scanned on each frame, which defaults to 10%

Reading full AI Traffic identity strings

The offset area at D000 can also be used to read full AI aircraft data strings. To do this, proceed as follows:

1. Write the selected command, from list below, to D004 (32-bit DWORD)

2. Read the timestamp at D008 (32-bit DWORD)

3. Write the AI id (from the TCAS table, see earlier) to D00C (32-bit DWORD)

4. Write a signature to D000 (32-bit DWORD)

It is probably best to do all that in one FSUIPC Process call—in recent versions of WideFS the read should be separated out for you in any case. The order isn't important except that you must write the signature last.

If you want to do another within 14 seconds, use the same signature. Use a signature of zero to allow anyone to do the same thing at the same time, but then be aware that your data may not be what you asked for.

5. Wait till the timestamp in D008 changes.

6. Read string result (up to 48 bytes including terminating zero) from D010.

The command values available are:

1 = Tail Number
2 = Airline name + Flight number
3 = ATC aircraft type, plus ATC aircraft model *
4 = Aircraft title
5 = ATC aircraft type + last 3 digits of tail number

* The aircraft type is one zero-terminated string, and the model is another, following immediately. If either are missing you'll still get the null string (i.e. just the zero terminator).

Except for the last case where 3 digits are extracted deliberately (in accordance with ATC practice), none of these strings are likely to be abbreviated, except perhaps any long Aircraft Titles. In other words don't expect the string read in command 2 to be the same as the 14 character version in the TCAS tables—though the beginning and end will be, of course.

TCAS Data from External Programs

FSUIPC can accept data to be placed into the TCAS table (airborne *only*), as described above, from programs interfacing via IPC. This can be used to supply MultiPlayer aircraft data, for instance. [*NOTE the word "data". This is simply a data passing facility. It is NOT a replacement for the FS multiplayer interface and does NOT generate aircraft!*]

To do this simply write (in one IPC block, i.e. one call to FSUIPC_Write) a TCAS_DATA structure to offset 0x1F80. Note that this address actually goes somewhere different internally. You cannot read back what you have written here.

You can add more writes to the same (or other) offsets before actually sending them (e.g. via FSUIPC_Process). The only important thing is that the whole TCAS_DATA structure is written in one block, with the length obviously set to sizeof(TCAS_DATA)—or, better, the size read from 0xF000.

The data this structure should contain is as follows:

id — Any id number UNIQUE to all aircraft you supply. It does not have to be unique to the AI aircraft. FSUIPC keeps an internal flag to distinguish the two types. [*Note that if in the future this field is re-used for other indications, FSUIPC may have to adjust the value supplied*].

lat, lon, alt, hdg, gs, vs, com1 — As possible: all would be good, but obviously a minimum of lat/lon/alt.

idATC — Any string of up to 14 chars (or 15 if no "state" information is needed), plus a zero terminator, to identify the aircraft. This doesn't need to be unique but it could be rather confusing to the user if it isn't.

To erase an aircraft provide the specific id for that entry, and set the idATC field to null (i.e. zero length string, just a zero).

In any case, FSUIPC will automatically erase any externally supplied aircraft after about 8–12 seconds if it receives no further updates in that time. Even if the aircraft is static you'll need to supply updates for it regularly.

Apart from the user-adjustable range, which is applied, FSUIPC is not performing any filtering for these aircraft—i.e. you can include aircraft on the ground if required. However, once the airborne TCAS table is full (current capacity 96) whether with AI aircraft, MP aircraft, or a mixture, no others will be accepted until slots become free. So in this sense slot management is up to you.

Path reading mechanism

FSUIPC provides a generic method of reading string data, primarily long pathnames, based around writing requests to specific offsets and reading results in others. The offsets used are 0FF0 to 10FF inclusive. (These were part of FS's Globals.DLL but don't appear to have been useful for anything in years so they are being grabbed for this).

The recommended procedure for using this facility is as follows (refer to the extensive notes below for a detailed explanation of each step and the data values needed):

1. Generate an initial (random?) signature value (DWORD). (*This should be done once oncly, somewhere in your program initialisation probably*).

Each time you want to use the facility:

2. FSUIPC Read the DWORD signature flag at 0FF0—see (1) below.
3. FSUIPC Read the DWORD timestamp at 0FFC
4. FSUIPC Write the command details (offsets 0FF4, 0FF6, 0FF8, as applicable)—see (2) below.
5. FSUIPC Write your signature value to 0FF0—see (2) also
6. FSUIPC Process call.

   Explanation so far: If another program is already using the facility then the signature read from 0FF0 will *not* be zero, and, unless by coincidence your signature is the same as the other's, the write will fail to cause your command to execute. That's safe and we deal with that in the next few steps. But by combining this check with the attempt to execute the command, all in one FSUIPC Process call, we make it 'atomic', in the sense that another program cannot get in and 'steal' the facility between you checking 0FF0 and writing your own data. This, in fact, is the only 'foolproof' method!

7. Check the value read from 0FF0. If this is zero, your command will have been accepted! Skip to step 9.

8. On the next time interval, or after processing Windows messages (usual clever Windows programming), return to step 2 above for another try. (Best make a time limit on how long you do this for, then skip to step in any case, hoping that the other program wanted the same as you).
9. Increment your signature, so that next time you don't 'accidentally' get accepted even with 0FF0 signalling an active user—if the active user is your program, your command will be executed even though you think it isn't.
10. FSUIPC Read the DWORD timestamp at 0FFC—see (3) below
11. FSUIPC Process. Check if the new timestamp value is different from the one read above. If so, skip to step 12. Otherwise, on the next time interval, or after processing Windows messages (usual clever Windows programming), return to step 10. Best place a sensible time limit on this—500 mSecs or so should cover all eventualities, but it depends on your application.
12. Read the data you asked for at offset 1000—still (3) below

Okay, that's it!

(1) See if FSUIPC can accept a path request:

Read DWORD at 0FF0. If this is zero, go ahead. If it is non-zero, there is an active request. If you know this to have come from your program, then go ahead using the same signature as before. If the previous request was made using a different signature, then the system is locked for up to 8 seconds to allow the previous result to be read.

(2) **To execute the request**

In one FSUIPC_Process call do the following:

Read the DWORD at 0FFC. This contains a "timestamp" indicating changes to the path read area. You need to read this first so you can see when it changes.

Write the Command WORD (16 bit) to 0FF4.

The only commands available at present are:

1	Read default Flight path [all FS versions]
2	Read AI traffic pathname for specified AI aircraft (see parameter) [FS2004 only]
3	Read filename (no path) of the last saved Flight (FLT) file. (Use the counter of flights saved, see offset 3BD2, to determine when this has changed). [**NOTE**: *since version 3.47 of FSUIPC, the filename of the last saved flight has also been readable directly at offset 0400, so this option is really only retained for backward compatibility. New programs should refer to offset 0400*].

Write the Command Option Flags WORD (16 bit) to 0FF6.

Options are by setting individual flags in this word. Set zero for defaults. The only option available at present is:

Bit 0 (2^0) Provide UNC path (i.e. network-usable) if the path is *not* within the FS installed path. This won't work if the drive or at least some of the path is not shared for Network access.

For FS sub-folders this option will make no difference. You can differentiate different path types as follows:

2nd character = : means it is a local full path
1st & 2nd characters = \\ means it is a UNC full path
else it is a sub-path within the FS folder (UNC path available elsewhere)

This option bit is ignored for command 3.

Bit 1 (2^1) For command 3 (reading FLT filename) only:

A value of 0 for this flag allows FSUIPC to overwrite the previous Flight name immediately, provided there is no other command pending for this path reading mechanism. When it does so it will set flag bit 2 (2^2) and update the timestamp. Using this method for reading flight filenames is fast but you do run a risk of missing a FLT name when they are so close you don't read one before the other.

If this bit is set to 1, it prevents FSUIPC providing a new Flight name ***if*** bit 2^2 = 1 (in other words it has already written one). With this method you need to read the FLT name then clear bit 2^2 (i.e. write just 2 to 0FF6) to allow the next FLT name to be provided. If there is a new one already, it will be provided immediately and the 2^2 bit will again be set.

Bit 2 (2^2) For command 3 (reading FLT filename) only:

This flag is *set* by FSUIPC when a new FLT filename is made available in offset 1000 ff. If this bit is still set when the next FLT file is saved, it will only overwrite the previous one in 1000 ff if flag bit 1 (2^1 above) is cleared.

If applicable, w**rite the DWORD parameter to 0FF8**. At present this is only needed for AI traffic pathnames. The value needed here is the DWORD "id" field from the TCAS table.

Write a non-zero signature to 0FF0. This prevents others interfering with your request until you've had time to read the results. FSUIPC clears down the request about 8 seconds after this Write is received. Successive requests within the 8 seconds are accepted but only if the same signature is used. You cannot read the signature last used.

Note that the Write to 0FF0 is the activator. Only at that point is the command and parameter accessed and used. For safety you should use FSUIPC_Write(s) for the command and parameter *then* the FSUIPC_Write for the signature. You can, if you wish, write all 12 bytes in one FSUIPC_Write, as the total write is done before the signature is checked.

(3) To read the results

Wait for the DWORD timestamp at 0FFC to change. This will happen as soon as FSUIPC sees your Process call (the request is executed immediately, *not* scheduled for the next FS frame—I am hoping this isn't needed in this case), but if you are running on WideFS you have to allow turn-round time.

For full safety, you may actually want to read the 12 bytes from **0FF4**, and check that the values for the Command and Parameters are yours—if you found the DWORD at **0FF0** was non-zero and assumed this was from your request, this would be a way of verifying that. You can include the 12 byte read in the same process as the Command write, provided it is later, as the command will be accepted or rejected as soon as FSUIPC sees the Signature write.

When 0FFC has been updated, the 256 bytes at offset 1000 will contain the path/filename as an ASCII zero-terminated string if the request was successful, or a null string (i.e. first byte = 0) if it failed for any reason.

Note that the default user Flight path will only be a full path if it is not within the FS folder. If the string does not begin with a drive letter (x:)—i.e. the second character is not a colon—then the path is a sub-folder in the main FS path. This applies to FS2002 and before.

Similar considerations apply to the AI Traffic file path. This gives all the details, right to the filename and file type, but the path is within the FS folder if no drive details are given. In other words, the entry is similar to those in the SCENERY.CFG file, from which they presumably derive. (See the way FSUIPC logs Traffic file data).

Weather (and other) Data for Adventures *[Not applicable to FS2004]*

Since version 1.55 of FSUIPC, the "**PatchWeatherToADV=Yes**" facility (defaulted on) has provided weather data to adventures in FS2000 (and now FS2002). This data is directly through the same variables as used in FS98, so, apart from some minor extensions (more of which below), they only cater for that subset of FS2000/2002 weather which is, in fact, common to both simulators.

For FS2002 the same facility also provides full autopilot access and control, and pushback, to adventures. This is included in FSUIPC from version 2.76.

Here is a complete list of FS98 compatible adventure variables now provided by FSUIPC. For details of the values they can take, please see the appropriate APL documentation for FS98. Naturally, only the lowest two cloud layers are accessible (three if the lowest is a Thunderstorm), as are the lowest four temperature layers and lowest four wind layers. This is because only these numbers of layers were actually available in FS98.

BAROMETRIC_PRESSURE
BAROMETRIC_DRIFT *

CLOUD_THUNDER_ACTIVE
CLOUD_THUNDER_COVERAGE
CLOUD_THUNDER_TYPE
CLOUD_THUNDER_BASE
CLOUD_THUNDER_TOP
CLOUD_THUNDER_ICING
CLOUD_THUNDER_TURB

CLOUD_LOW_ACTIVE
CLOUD_LOW_COVERAGE
CLOUD_LOW_TYPE
CLOUD_LOW_BASE
CLOUD_LOW_TOP
CLOUD_LOW_ICING
CLOUD_LOW_TURB

CLOUD_HIGH_ACTIVE
CLOUD_HIGH_COVERAGE
CLOUD_HIGH_TYPE
CLOUD_HIGH_BASE
CLOUD_HIGH_TOP
CLOUD_HIGH_ICING
CLOUD_HIGH_TURB

TEMPERATURE_SURFACE_TEMP
TEMPERATURE_SURFACE_ALT
**
TEMPERATURE_LOW_TEMP
TEMPERATURE_LOW_ALT
TEMPERATURE_MID_TEMP
TEMPERATURE_MID_ALT
TEMPERATURE_HIGH_TEMP
TEMPERATURE_HIGH_ALT

WIND_SURF_VEL
WIND_SURF_DIR
WIND_SURF_DEPTH
WIND_SURF_TYPE
WIND_SURF_TURB***

WIND_LOW_VEL	WIND_MID_VEL	WIND_UP_VEL
WIND_LOW_DIR	WIND_MID_DIR	WIND_UP_DIR
WIND_LOW_BASE	WIND_MID_BASE	WIND_UP_BASE
WIND_LOW_TOP	WIND_MID_TOP	WIND_UP_TOP
WIND_LOW_TYPE	WIND_MID_TYPE	WIND_UP_TYPE
WIND_LOW_TURB	WIND_MID_TURB	WIND_UP_TURB

* Note that "BAROMETRIC_DRIFT" doesn't appear to have been implemented in FS2000/2002, although there is space for it, or something like it, in the weather structures. The value provided by FSUIPC is, since version 1.962, the difference between the current aircraft position QNH (which may be in transition), and the METAR reported QNH as set by the weather control program. Adding the Drift value to the Pressure will give the correct value for ATIS reports.

** The value placed into "TEMPERATURE_SURFACE_ALT" will either by the value provided for this by the external weather control program, or a value determined by FSUIPC to be the best for computing Cloud Base altitudes 'above ground level' (AGL). In the latter case it will be the currently determined 'Real Weather' METAR reporting station's ground elevation when local weather is in force, or the actual ground altitude below the aircraft when global weather is in force. As noted elsewhere, it is strongly recommended that all weather control programs place the METAR station's altitude into the FS98 equivalent of this variable (at global offset 0xF40) so that ATIS reports from Adventures can reliably give AGL cloud base information.

*** Since version 1.962 of FSUIPC, the WIND_SURF_TURB value is used to provide wind gust information. If it is zero there are no gusts. Otherwise it gives the upper gust velocity. This is usually derived from METAR data and set by the weather control program.

Some additional values are available, specifically for FS2000/2002, but access to these requires a recent version of Martin Smith's APLC32 adventure compiler (version 1.32 or later). Please also check Martin's documentation.

FSUIPC_VISIBILITY	in statute miles
FSUIPC_DEWPOINT	in degrees C (from lowest temperature layer only)
FSUIPC_PRECIPITATION_TYPE	0=none, 1=rain, 2=snow, from lowest cloud layer
FSUIPC_PRECIPITATION_RATE	0-5, from lowest cloud layer
FSUIPC_STATUS	fixed recognition value, –0.9977, to be checked before using above

Note that accessing any of these variables without 1.56 or later of FSUIPC installed will result in the Adventure terminating immediately. This apparently occurs because, without FSUIPC filling in the details, these variables are actually illegal.

Since version 1.80 of FSUIPC, the FSUIPC_PRECIPITATION_TYPE and FSUIPC_PRECIPITATION_RATE variables can also be written to in an adventure, to control precipitation. The other values are read only. (To write to these two variables you will need version 1.33 or later of APLC32).

Since version 1.97 of FSUIPC, its own (optional) semi-random rain and snow generation is suspended for 30 seconds each time the adventure sets either FSUIPC_PRECIPITATION_TYPE or FSUIPC_PRECIPITATION_RATE. This means that any adventure needing to have assured control over precipitation should refresh or change the variables at less than 30 second intervals, otherwise FSUIPC will assume it has finished and will re-enable its own rain/snow system if it is selected.

Since version 2.76 of FSUIPC, you can use variable SYSVAR_C4 to read the current FS2002 ***pushback*** status, and, indeed, to control the pushback. The values read/written are:

3 = off
0 = pushback straight
1 = pushback with tail swinging to left (to face aircraft to right)
2 = pushback with tail swinging to right (to face aircraft to left)

Please note that FS2002 pushes back straight for at least the length of the aircraft before swinging the tail in any case.

Additional Information for Application Programmers

GENERAL NOTE: Before using any of the added controls listed in this section, application programs should check that FSUIPC has initialised and is ready, otherwise any settings written may be overwritten shortly. This applies particularly when your applications are run on a Network via WideFS. Whilst WideClient is 'emulating' FS on a networked PC it will supply zeroes for all reads through the FS6IPC interface. When FS is loaded and FSUIPC is initialised on the Server PC, the values become valid. Only then make any changes. You can check for FSUIPC initialisation on FS by reading 0xFFFF at offset 0x4D2 and/or 0x8000 at offset 0x4D4. Also, with effect from version 1.40 you can read 0xFADE as offset 0x4D6.

From version 1.998e of FSUIPC you can read Version information too: please refer to offsets 3304-330B in the tables.

WEATHER in FS2004 is substantially changed from previous versions. Most if not all of the following notes will not apply, or at least not precisely as worded here. FSUIPC will offer powerful new features to both read and write FS2004 weather, both globally and locally (by METAR station ICAO identity). Further details will be published in due course.

WIND GUST control: Weather control programs designed only for FS98 provided only gust (1) or no gust (0) information. For these, FSUIPC generates a random gust speed. However, if the program supplies a value greater than 1, this is treated as a required gust speed in knots. If the value is greater than the base wind speed then it is used as the upper wind speed, otherwise it is treated as the required difference.

CLOUD type control: The cloud type is copied by FSUIPC if this is provided as a value from 1 to 10 by the weather control program. The automatic setting of cloud types by FSUIPC applies if the type provided is out of this range.

The cloud types used in FS98 still apply to FS2000/2002, although they all seem to be mapped by FS to one of Cirrus (1), Stratus (8), Cumulus (9), or Thunderstorm (10). I also think that the Cumulus setting gives an appearance that is far short of the requested coverage. (For automatic type setting, FSUIPC only chooses Cumulus for sparser cloud covers, or occasionally boosts the coverage from that requested).

RAIN and SNOW control: Your application can be made to control the rain and snow as follows:

> Set 16-bit word at global offset 0x4D2 as follows:
> Low byte = precipitation rate: 0–5 (0 none, 5 most)
> High byte = precipitation type: 0–2 (0 none, 1 rain, 2 snow)

This will only have any effect if there are any clouds specified. If this facility is used, FSUIPC's own random rain/snow generation is turned off. To re-enable this action (if it isn't disabled by user option), your application should restore the value at 0x4D2 to 0xFFFF before terminating.

Do not read this offset for precipitation details. FSUIPC will initialise it to FFFF, and if it is left at that will, by default, perform its own random rain/snow. To stop that you must write to it: 0x0000 for no precipitation, and so on.

DEW POINT control: As another interim measure, your application can be made to control the surface temperature layer's dew point, as follows:

> Set 16-bit word at global offset 0x4D4 to the dew point temperature in Degrees Centigrade x 256

To re-enable the automatic setting of dew point by FSUIPC, your application should restore the value at 0x4D4 to 0x8000 before terminating. Do not read this offset for dew point details. FSUIPC will initialise it to 0x8000, and if it is left at that it will, by default, perform its own random dew point calculations.

Upper temperature layer dew points are calculated by FSUIPC on the basis that the air becomes drier with altitude. This isn't always true—obviously the cloud layers have saturated air. However, it seems that FS2000 doesn't actually do anything with the dew point at present: it should affect things like icing, but this is apparently not so.

ADVANCED WEATHER Interface (AWI): A special interface is provided by FSUIPC that provides full writing and reading of all FS2000 and FS2002 global weather data. This involves the use of otherwise illegal global offset values, so to prevent applications crashing FS they should check whether the advanced interface is provided. FSUIPC indicates this support by providing the value 0xFADE in the 16-bit word at offset 0x4D6. If this special recognition pattern is not read at this address then the Advanced Weather facilities should not be used. [Programmers needing more information on this interface are referred to the files enclosed in UIPCAWI.ZIP, included with this release].

For FS2004, although the AWI can be used for Global weather, a **New Weather Interface** (NWI) is provided. This deals with all aspects of FS2004 weather and allows both global and local (weather station) weather to be set and read. This is important in FS2004 because so-called "global" weather does not stay global for long. The atmosphere

is 'alive' and constant changes are occurring, including the localisation of weather. The **NWI** is described separately in the SDK, look for the file "**New Weather Interface for FS2004.zip**".

ADDITIONAL Weather information: When FSUIPC's weather processing for FS2000/2002 is in operation, some extra weather data is available through the following (read-only) offsets:

0x4C8	16-bit word	Dew point at the lowest temperature layer, in degrees C x 256.
0x4CA	8-bit byte	Flag: 0 = no weather accessible 1 = global weather 2 = local or downloaded weather in operation, but not accessible 3 = local or downloaded weather
0x4CB	8-bit byte	Precipitation Rate (0-5), in lowest cloud layer only
0x4CC	8-bit byte	Precipitation Type (0=none, 1=rain, 2=snow), in lowest cloud layer only
0x4CD	8-bit byte	Thunderstorm cloud layer cover, 0-8 octas
0x4CE	8-bit byte	Lowest (non-storm) cloud layer cover, 0-8 octas
0x4CF	8-bit byte	Next highest cloud layer cover, 0-8 octas

SURFACE wind information: Normally the information on the surface wind and the first three upper wind layers is accessible from the same "current weather" offsets as in FS98. However, when FSUIPC's "Wind Transition" facility is enabled, this won't be so. There will only be one *very* thick surface wind layer, which FSUIPC manipulates to provide the correct wind values at the correct time.

Full layer information (not limited to FS98's four layers) is available through FSUIPC's Advanced Weather interface for FS2000/2002. Otherwise, programs needing details of the surface wind speed and direction can read them at offsets 0x4D8 (speed in knots) and 0x4DA (Magnetic direction in the usual FS98 units, i.e. 65536 == 360 degrees).

CLOUD BASES and GROUND ALTITUDE: There's a small problem with cloud bases. Within both FS98 and FS2000/2002 this altitude information is received and stored in metres above sea level (AMSL). But in METARs and ATIS reports, clouds are reported at levels above the reporting station. External weather control programs need to add the station's ground elevation to the METAR-extracted cloud bases before supplying them to FS. That is easy enough. But then Adventure and other programs need to recalculate the original AGL values for ATIS reports. Traditionally this has been done by subtracting the GROUND ALTITUDE, but this can go pretty badly wrong, for this is the elevation of the ground below the aircraft at the time. If an ATIS report is provided to an aircraft flying into Denver over the Rockies, with a cloud base over Denver well below the mountains, the result will not only be wrong but will be negative!

To get around this problem I am suggesting that all external weather control programs supply the reporting station's ground elevation as the surface temperature altitude—i.e. the value, in metres, which is placed into global offset 0xF40. This makes sense in any case, as both FS98 and FS2000/2002 take this, the lowest temperature layer, to be the surface temperature, and by definition the METAR *is* reporting the temperatures pretty close to the surface at the reporting station. Note that the surface temperature altitude specified via the Advanced Weather Interface will NOT be provided to adventures. For this special ground level operation the program *must* set 0x0F40.

Since version 1.60, FSUIPC has also been setting the surface temperature altitude to the ground altitude (unless over-ridden by the external weather program), so Adventures using this won't be any worse off than before. In fact, if local 'real' weather is in force, FSUIPC will derive the METAR station's ground elevation and supply that instead, in the same position.

So, all that is needed then is for Adventure writers to calculate AGL values using the surface temperature altitude instead of the ground altitude, and things will be better. <G>

WEATHER INDICATOR for FS2000 and FS2002 (*not* FS2004): FS has two weather modes, which I will call "Local" and "Global". Local weather is set when you use FS's "real weather" downloading facility to set your weather. In this mode weather control is left to FS. Note that Local Weather mode is also set by FS when a Flight is loaded which was saved with 'real weather' operating.

Using the "Clear All Weather" button in FS's weather dialogue reverts the simulator to "Global Weather" mode. In this mode external weather programs can always have an effect. Global Weather mode may also be re-enabled by loading a Flight which was saved in this mode.

If external inputs are received by FSUIPC when Local Weather is in force, then the result depends on the setting of the "AutoClearWeather" option. If it is disabled then external weather inputs are recognised and stored, but cannot be seen to operate until the user presses the "Clear All Weather" button, at which time all the received weather is enacted. If this option is enabled then FSUIPC will automatically clear the local weather upon any external weather input, and activate it within a few seconds (allowing time for the graphics to catch up). This is the default action.

FSUIPC provides an indicator of the current weather mode, as follows:

Byte at offset 0x4DD:

2^7 = 1 indicates "Local Weather" is in operation
2^7 = 0 indicates "Global Weather" is in operation

2^6 = 1 indicates FS2000 weather can be read (whether Local or Global)
2^6 = 0 indicates that the weather cannot be read (or this is not FS2000!)

The WeatherSet.exe application provided with FSUIPC uses these bits to display the current mode in its title bar.

OPTION control: There are now two different methods for external programs to change FSUIPC weather filtering options. The older method, now discouraged, is as follows:

Byte at 0x4DE = Mask: set bits here of each option to change
Byte at 0x4DF = Value: the related bits set or clear, as needed.

The 8 bits are allocated as follows (the '0' settings are the defaults):

0	1	GenerateRain	0=Yes	1=No
1	2	GenerateCirrus	0=Yes	1=No
2	4	GraduatedVisibility	0=Yes	1=No
3	8	WindTransitions	0=No	1=Yes
4	16	surface winds TRUE*	0=No	1=Yes
5	32	UpperWindGusts	0=No	1=Yes
6	64	StormsAutomatic	0=No	1=Yes
7	128	ExtendMetarMaxVis	0=Yes	1=No

* Surface wind directions input to FSUIPC via the old FS98 locations are treated as being Magnetic, for compatibility with FS98. If you *know* you are actually sending True directions (as given in METARs), then you can change this here. (Note: this used to be an INI file option, now removed to prevent user confusions!).

Your application should restore both bytes to zero before terminating, so that the user's settings again prevail.

The newer and more thorough method uses a new set of offsets and is safer. It has a built in safeguard against the program terminating untidily—something easily done if it is used under WideFS and the WideClient.exe is closed before the application!

This uses 9 bytes in the FSUIPC interface, as follows:

3127 1 byte timer. Nothing happens until you write to this byte. If you write zero here, FSUIPC reverts to the user-selected options. When you write a non-zero value, the options selected in the next 8 bytes (mask in DWORD at 3128, settings at 312C) are applied.

FSUIPC counts the value here down by 1 every "tick" (55mSecs). When it reaches zero, the user options are restored. This is a safeguard against your program hanging, crashing or terminating untidily. To retain the option selections you wish you must write a non-zero value here regularly. Each time you do so the values in 3128 and 312C are re-applied.

Obviously the maximum time between these updates is 255 x 55 mSecs = 14 seconds, but to be safe write every 5 or so seconds.

3128 32 bit Mask, defining the options you want to control (see list below)

312C 32 bit Settings, defining the state you want for each of the options selected by the mask. Note that the user can still opt to prevent you changing his options. That's an option you cannot change.

These are the bit assignments:

2^0	0x00000001	Clouds: do NOT generate rain/snow
2^1	0x00000002	Clouds: do NOT add a light cirrus layer
2^2	0x00000004	Visibility: DISABLE graduated visibility
2^3	0x00000008	Winds: transitioning option
2^4	0x00000010	(reserved, not changeable)
2^5	0x00000020	Winds: allow upper gusts
2^6	0x00000040	INI file only, StormsAutomatic
2^7	0x00000080	Visibility: do NOT random extend METAR max
2^8	0x00000100	Visibility: enable smoothing
2^9	0x00000200	Winds: do NOT set shear sharp
2^10	0x00000400	(reserved, not changeable)

2^11	0x00000800	(reserved, not changeable)
2^12	0x00001000	Technical: do NOT attempt to stop white-outs
2^13	0x00002000	(reserved, not changeable)
2^14	0x00004000	(reserved, not changeable)
2^15	0x00008000	Winds: do NOT extend top wind upwards
2^16	0x00010000	Clouds: set only one cloud layer
2^17	0x00020000	Clouds: set only thin cloud layers
2^18	0x00040000	Clouds: add random turbulence
2^19	0x00080000	Clouds: add random icing
2^20	0x00100000	Winds: add random turbulence
2^21	0x00200000	Winds: allow gusts in upper winds
2^22	0x00400000	(reserved, not changeable)
2^23	0x00800000	Clouds: set thunder cloud thickness
2^24	0x01000000	Clouds: add jet trails (FS2000) or V-sky layer (FS2002)
2^25	0x02000000	Winds: add value to wind layer altitudes
2^26	0x04000000	(reserved, not changeable)
2^27	0x08000000	(reserved, not changeable)
2^28	0x10000000	Winds: set turbulence as variability
2^29	0x20000000	Technical: smooth pressure changes
2^30	0x40000000	(reserved, not changeable)
2^31	0x80000000	Winds: do NOT raise subterranean winds

HOT KEYs for Applications: Facilities are provided to allow programs to detect selected key presses made when FS has the focus. Up to 56 such hot keys can be specified, but this number is shared by all running applications. Additionally an extra key pressed before the main hotkey is released can be requested and supplied, multiplying the number of possibilities immensely without needing many slots.

The facility operates using some of the extended IPC offsets, thus:

DWORD at offset 0x320C = size of Hot Key table (read only). Currently = 56.

Read this first to see how many entries there are altogether. If it reads Zero you have the wrong version of FSUIPC!

Say this contains 'n', then n DWORDs from 0ffset 0x3210 onwards (i.e. 0x3210, 0x3214 ...) are 'slots' for Applications to specify Hot Keys. These will be zero initially, and zero if free. The application must search through to find an empty slot, then set this into it:

Byte 0 (bits 0-7): Virtual Keycode (see the list in my FS Controls documents or the FSUIPC Advanced Users Guide).

Byte 1 (bits 8-15): Shift state indicator

- Bit 0, the least significant, = shift
- Bit 1= ctrl
- Bit 2= alt (but use of alt strongly discouraged, see Note 1)
- Bit 3= "expect another keypress". If this bit is set then when the Hot Key is detected FSUIPC waits for the KEYUP *or* another key press first. The virtual keycode for that keypress is then returned in Byte 3, below.
- Bit 4= tab (provided as an extra "shift", for more key press flexibility)

Byte 2 (bits 16-23): Flags from application.

- Bit 0 (1)=*reserved.* This was originally used to control the next option, but it was implemented incorrectly in FSUIPC, so now, to avoid problems, the bit is deliberately ignored.
- Bit 1 (2)= set if Hot Key should be passed through to FS, else it will be trapped. See Notes 1 & 2.

Byte 3 (bits 24-31): Flags or results from FSUIPC.

This byte needs to be cleared by the application so that it can detect when the Hot Key occurs. There is no queuing. If the Hot Key alone is seen, this byte is set to 1. If bit 3 was set in Byte 1 above *and* another key was pressed before the hotkey was released, then the virtual keycode for the extra key (2–255) is provided here.

Note 1: ALT key combinations are not a good idea, and cannot be stopped from passing to FS. You can get them, but FS will open the menu in any case.

Note 2: If the same Hot key is listed more than once (for instance by several applications), every copy for the same Hot Key will get the flag set, irrespective of the pass-through option. The option only applies to finally passing it to

FS. If any one Hot Key user says that the key is *not* to be passed to FS (i.e. by leaving Flag Bit 1 unset), then it isn't passed through.

Note 3: FSUIPC hotkeys, allocated in its "Technical" page, take precedence and are not passed through to applications or FS.

Use: Having found an empty slot, write the above value into it, then monitor the highest byte of that same slot for Non-Zero. That's the keystroke. Clear that byte to detect it again. If you register several Hot Keys it will be more efficient to only scan the slots themselves when a hot key actually occurs. To detect this, simply monitor the one byte at offset 32FE (this can be paired with 32FF to scan for keys and buttons together). When it changes, read and check the flags in your slots. (The count at 32FE may change without any of your keys occurring, of course, if other applications are trapping other hot keys).

When finished, and certainly before exit, be sure to clear the whole DWORD to zero so other applications can use it. If you only want to use keystrokes for a certain part of the operation of your program, only set the entries there and clear them when done.

Note that if several applications want the same keystroke, they will all get it. Of course, your application can check through the whole list to make sure there are no clashes/duplicates and warn the user if so. You might have to do that at intervals in case a clashing application is loaded after yours.

This system will work through WideFS with no problems too. You should be using WideFS 4.61 or later with this version of FSUIPC.

MODULES MENU access for Applications: From version 2.89 of FSUIPC, the Hot Key facilities are extended to allow an application to add an entry to the Modules menu. The Application finds a free Hot Key slot, then sets it up to receive notification on menu access, and writes the text needed for the menu item to another location. When the menu item is selected, the flag in the hot key slot is set just as when a hot key is used.

This way of accessing the menu has the advantage that it will also work when the application is running on another PC, via WideFS. Of course, any response to that menu selection will occur on whichever PC the application is running.

To avoid having menu items relating to applications that have crashed or terminated without tidying up correctly, each menu item added is subjected to a time-out. Applications have to refresh a count in the Hot Key slot at regular intervals (10 seconds or less) otherwise the menu item is deleted and the Hot Key slot freed. The time-out is suspended when FS is paused, and there is an option to have FS pause automatically when the menu entry is selected.

This is the way this facility is used:

1. Find a free Hot Key slot (i.e. search the 56 DWORDs at offset 0x3210 for a zero value). Say slot ***I*** is the one found.
2. Write 0x0000FFFF to the slot (i.e to the DWORD at offset 0x3210 + 4****I***). If you want FS to pause when the menu item is selected, write 0x0002FFFF instead. The 02 part is the flag indicating that a pause is required.
3. Write the text for the menu entry required to offset 0x2FE0, with the first byte set to the slot number (***I***). For example, for an entry "UIPC Hello" (H being the shortcut) you would set the string to be written to 0x2FE0 as follows:

   ```
   static chMenuEntry[] = "?UIPC &Hello";
   chMenuEntry[0] = I;
   ```

4. The '&' in the string tells Windows which character to underscore, and this denotes the shortcut key, but this is optional.
5. The string is limited to 31 characters, including the slot number at the beginning, plus a zero terminator. In other words the offset range is 0x2FE0–0x2FFF inclusive. This area is "write only". Don't expect to be able to read back what you write here.
6. The write to 0x2FE0 triggers FSUIPC into adding the menu entry to the Modules main menu item, but this is dependent upon the slot it references being set with 0xFF in its first (least significant) byte. From the moment the *slot* is set with 0xFF there it is changed every 55 mSecs or so, unless FS is paused or in a dialogue. The change is a decrement of the next byte in the slot—the other one you also set to 0xFF. When this reaches zero, the menu entry is removed and the slot is cleared. This gives a maximum timeout of 255 x 55mSecs, or about 14 seconds. You can make it less, of course, by initialising that byte to a lower value than 0xFF (255), but I'd recommend sticking to the maximum.

7. This means that if you want the menu entry to stay available you must write 0xFF (or whatever) to that byte (i.e. the slot offset + 1) at regular intervals, say every 10 seconds. The 4 second leeway allows some safety, but you may want more—very little FS overhead is caused by writing that one byte every 1 second if you need to, but this is really over the top. More overhead is caused by writes when running on another PC using WideFS, so I would suggest 5 seconds as a minimum.

8. When the user selects your menu entry, FSUIPC will set the 2^0 (0x01) bit in the top byte (offset+3) in your slot. Just as with Hot Keys, you need to be looking for this at regular intervals, perhaps every 200 milliseconds or so. Frequent reads pose little overhead for WideFS use, but very frequent ones should really be avoided when you are running on the FS PC.

9. After processing the user request, whatever it is, don't forget to clear the indicator so you can detect the next one—writing zero to the byte at the offset+3 is all that is needed.

10. Finally, if you opted for FS to pause when the menu item is selected you need to unpause FS so that it can continue. Write zero to the 16-bit value at offset 0x262.

When you no longer need the menu entry, or just before terminating your program, you should write zero to the DWORD Hot Key slot. This will make FSUIPC remove the menu entry immediately. If your program does not tidy up the entry will be removed on the timeout.

HOT BUTTONs for Applications: Facilities are provided to allow programs to detect selected joystick button presses. This facility is very similar to the Hot Key system described above. Up to 56 such hot buttons can be specified, but this number is shared by all running applications. The facility operates using some of the extended IPC offsets, thus:

DWORD at offset 0x290C = size of Hot Key table (read only). Currently = 56.

Read this first to see how many entries there are altogether. If it reads Zero *or some very large number*, then you have the wrong version of FSUIPC!

Say this contains 'n', then n DWORDs from 0ffset 0x2910 onwards (i.e. 0x2910, 0x2914 ...) are 'slots' for Applications to specify Hot Keys. These will be zero initially, and zero if free. The application must search through to find an empty slot, then set this into it:

Byte 0 (bits 0-7): Joystick number (0-15) + 128. In other words 128 for Joystick 0, 129 for joystick 1, etc.
Joysticks are numbered as in FS2000.CFG, from 0. Game Controllers numbers from 1.

Byte 1 (bits 8-15): Button number (0-39)
Again buttons are numbered here as in FS2000.CFG, from 0. Game Controllers numbers from 1. Buttons 0–31 are the normal buttons, numbers 32–39 are the FS representation of the 8 "Points of View" at 45 degree angles supported by some joystick drivers for the POV Hats.

Byte 2 (bits 16-23): Flags from application.
This byte indicates which change is to be notified:
= 0 for Off to On
= 1 for On to Off
= 2 for both Off to On and On to Off
= 3 for Off to On but repeating about 6 times per second whilst it is on.

Byte 3 (bits 24-31): Flags from FSUIPC.
Bit 0 (value 1) is set when the specified Hot Button change occurs. Needs to be cleared by Application when seen so it can detect another. (No queuing).
Bit 1 (value 2) is set when bit 0 is set only if the button is still pressed. This can be used to differentiate the two events when Byte 2 is given as "2" for both off–on and on–off events.

Note: If the same Hot button is listed more than once (for instance by several applications), every copy for the same Hot button will get the flag set.

Use: Having found an empty slot, write the above value into it, then monitor the highest byte of that same slot for Non-Zero. That's the button event. Clear that byte to detect it again. If you register several HotKey Buttons it will be more efficient to only scan the slots themselves when a hot button actually occurs. To detect this simply monitor the one byte at offset 32FF. (This can be paired with 32FE to scan for buttons and keys). When it changes, read and check the flags in your slots. (The count at 32FF may change without any of your buttons occurring, of course, if other applications are trapping other hot buttons).

When finished, and certainly before exit, be sure to clear the whole DWORD to zero so other applications can use it. If you only want to use joystick buttons for a certain part of the operation of your program, only set the entries there and clear them when done.

Note that if several applications want the same button, they will all get it. Of course, your application can check through the whole list to make sure there are no clashes/duplicates and warn the user if so. You might have to do that at intervals in case a clashing application is loaded after yours.

This system will work through WideFS with no problems too. You should be using WideFS 4.61 or later with this version of FSUIPC.

FSUIPC SOUND INTERFACE

[Applicable to FSUIPC versions 3.987 and 4.609 or later.]

These facilities are based on those originally released in the module called "ESOUND" many years ago. They uses FSUIPC offsets in the region 0x4200-0x42FF, as follows:

4200 BYTE — Command/Engaged Marker

0 = FREE: FSUIPC is ready for a sound command (written by FSUIPC)
1 = PLAY once command (written by application)
2 = PLAY looped command (written by application)
3 = STOP playing command (written by application)
4 = QUERY sound status (written by application)

4201 BYTE — Option -- device and position selection

Bits 0-3 = sound device number (0-15). 0 is the default device.
FSUIPC lists the devices in the [Sounds] section of its INI file.

Bits 4-6 = position, 0-7, 0 = fwd centre, 1 = fwd right, 2 = centre right, 3 = rear right,
4 = rear centre, 5 = rear left, 6 = centre left, 7 = fwd left
These are collapsed to left-centre-right with a stereo setup.

4202 WORD — Sound STATUS as a 16-bit word. Related to the REFERENCE (below)

0 = Status not set
1 = Playing sound once
2 = Playing sound in loop
3 = sound has ended, or reference not listed

4204 DWORD — REFERENCE as 32-bit non-zero integer.

The application can use this how it likes, but it should be unique,
otherwise the same sound is referenced.

4208 Char string — Up to 247 characters plus a zero terminator, identifying the Wave file to be played when a PLAY or PLAY LOOP command is given. The ".wav" suffix is not needed, but can be given.

The default sound path is the <FS>\Sound folder, unless changed in the [Sounds] section of the FSUIPC INI file. You can use subfolders within this. Include <drive>: details for a complete or separate path.

PROTOCOL

General:

1. Wait for COMMAND (Byte at 4200) = 0 (don't forget to Sleep a little).
2. Then write data to 4204 (DWORD) and, for a new sound to play, the file or pathname to 4208 (zero-terminated character string). For options write to 4201 (though this is best done by writing a WORD command to 4200 with the options in the high byte).
3. Finally write the Command to 4200 possibly as a WORD with the 4201 options in the high byte.

Note that the latter two parts, writing to 4204- and to 4200 *can* be done in the same "FSUIPC_Process" call, provided the FS6IPC structure to write the data to 4204 onwards is listed *before* that setting the command to 4200. Otherwise the command may be obeyed with wrong data!

Status:

For a new sound the "reference" DWORD should be unique. if your program is the only one using FSUIPC sound, and you are sure of this, then this is easy. You can use addresses of wave filename strings, or sequence numbers, whatever. But if you may be sharing the resource via IPC this may not be good enough. maybe the user even runs TWO or more copies of your program?

To get a unique reference there are two simple methods:

a) Use the current time or "tick count" as the reference (e.g. the result of the windows API call "GetTickCount()"). This isn't guaranteed to be unique, but it's a good start. for several close or simultaneous sounds, get one tick count and add 1 for each successive sound.

b) Ask FSUIPC for the status of the sound with a given reference. if it says the sound doesn't exist or has ended, then just use the same reference again.

Note that if you use the same reference as a playing sound to play a new sound, then the earlier one will stop and the new one will start. For a program with sequential sounds this is fine. You only need different references when you want overlapping or simultaneous sounds.

Okay. Here's how to get the Status:

1. Wait for COMMAND byte at 4200 = 0 (don't forget to Sleep a little)
2. Write your reference to the DWORD at 4204
3. Write 4 (QUERY STATUS command) to 4200
4. Wait for COMMAND byte at 4200 = 0 (best to sleep for 100mSecs or so before this)
5. Read STATUS in WORD at 4202.

If the STATUS value is 3 then the reference is unused or the referenced sound has Ended.

Play/Loop sound:

Having determined a reference to use in any way you like, as described above,

1. Wait for COMMAND byte at 4200 = 0 (don't forget to Sleep a little)
2. Write Reference to DWORD at 4204 and Wave filename or pathname to 4208 onwards.
3. Write PLAY (1) or PLAY LOOP (2) command to 4200, along with the Options in the high byte.

Stop sound:

1. Wait for COMMAND byte at 4200 = 0 (don't forget to Sleep a little)
2. Write Reference for sound to be stopped to DWORD at 4204.
3. Write STOP (3) command to 4200

Table of offsets for FS98 (and applicable to FS2000, FS2002 and FS2004 via FSUIPC)

*(The applicability to FS2002 and FS2004 is still currently not 100% Please refer to the additional column in the tables. Where it is empty this indicates that it's status in that version has not yet been determined. Note that many offsets will also be applicable to **CFS2** and some also perhaps to **CFS1** but I'm afraid you'll have to decide which for yourself)*

This table is derived from much foraging of my own, but certainly not without a great deal of initial assistance from the many existing lists of FS98 offsets available due to the hard work of others before me. I must therefore acknowledge those I know of:

Thanks to: Adam Szofran, Ted Wright, Richard Lovett, David Drouin, Joaquin Manuel, JF Vole, Jason Grooms, Alessandro Antonini, Luciano Napolitano, Hao Chen, Pelle Liljendal, Arno Gerretsen, Wim Van Ertvelde, Hervé Sors, Ian Donohoe and Matthias Neusinger. Apologies to anyone I've omitted. Many of the lists circulating appear to be anonymous, so thanks to him/them too! <G>

Note that the values listed here have mostly been verified and mapped correctly by FSUIPC (version 2.96 or later) for use in FS2000 (fully patched with Microsoft updates) and possibly CFS2, but there is no guarantee that any particular value is supported. Those marked as okay with FS2002 are assuming an official Released version of FS2002 (build 10919.01) with no patches, working with FSUIPC 2.97 or later (3.80 for some locations). The mappings confirmed in FS2004 need the official release (build 30612.02) or later, and FSUIPC version 3.98 at least.

In this table Offsets are always in hexadecimal. For everything else numbers are decimal unless indicated in the C-style (i.e. 0xFFFF is hexadecimal for 65535, or –1 in a 16-bit signed word). The size column is in bytes throughout.

Offset	Size	Use	FS2002	FS2004*
0020	4	Ground altitude in Metres x 256. (see also offset 0B4C)	**Ok**	**Ok**
0024	Varies	Zero terminated string giving the Start-Up situation or flight name, including the path from the FS folder (usually PILOTS\ …)	**Ok**	**Ok**
012C	Varies	Zero terminated string giving the name of the current Log book, with the default being called just 'logbook' instead of the true filename. [*This applies to FS2002, but hasn't been verified on the others*]	**Ok**	**Ok**
0238	1	Hour of local time in FS (0–23)	**Ok**	**Ok**
0239	1	Minute of local time in FS (0–59)	**Ok**	**Ok**
023A	1	Second of time in FS (0–59)	**Ok**	**Ok**
023B	1	Hour of Zulu time in FS (also known at UTC or GMT)	**Ok**	**Ok**
023C	1	Minute of Zulu time in FS2	**Ok**	**Ok**
023E	2	Day number in Year in FS (counting from 1)	**Ok**	**Ok**
0240	2	Year in FS	**Ok**	**Ok**
0246	2	Local time offset from Zulu (minutes). +ve = behind Zulu, –ve = ahead	**Ok**	**Ok**
0248	2	Season: 0=Winter, 1=Spring, 2=Summer, 3=Fall	**Ok**	**Ok**
0262	2	Pause control (write 1 to pause, 0 to un-pause).	**Ok**	**Ok**
0264	2	Pause indicator (0=Not paused, 1=Paused)	**Ok**	**Ok**
0274	2	Frame rate is given by 32768/this value	**Ok**	**Ok**
0278	2	Auto-co-ordination ("auto-rudder"), 1=on, 0=off	**Ok, but doesn't change Menu setting**	**Ok, as FS2002**
0280	1	Lights: this operates the NAV lights, plus, on FS2000, the TAXI, PANEL and WING lights. For separate switches on FS2000 (and CFS2?) see offset 0D0C	**Ok**	**Ok**
0281	1	Beacon and Strobe lights. For separate switches on FS2000 (and CFS2?(see offset 0D0C	**Ok**	**Ok**
028C	1	Landing lights. (See also offset 0D0C on FS2000, and maybe CFS2).	**Ok**	**Ok**

029C	1	Pitot Heat switch (0=off, 1=on)	**Ok**	**Ok**
02A0	2	Magnetic variation (signed, –ve = West). For degrees *360/65536. Convert True headings to Magnetic by *subtracting* this value, Magnetic headings to True by *adding* this value.	**Ok**	**Ok**
02B2	2	Zoom factor: FS2002 only, and read-only. 64=x1, 128=x2 et cetera	**Ok**	**Ok**
02B4	4	GS: Ground Speed, as 65536*metres/sec. Not updated in Slew mode!	**Ok**	**Ok**
02B8	4	TAS: True Air Speed, as knots * 128	**Ok**	**Ok**
02BC	4	IAS: Indicated Air Speed, as knots * 128	**Ok**	**Ok**
02C4	4	Barber pole airspeed, as knots * 128	**Ok**	**Ok**
02C8	4	Vertical speed, signed, as 256 * metres/sec. For the more usual ft/min you need to apply the conversion *60**3.28084/256	**Ok**	**Ok**
02CC	8	Whiskey Compass, degrees in 'double' floating point format (FLOAT64)	**Ok**	**Ok**
02D4	2	[*FS2004 only*] ADF2 Frequency: main 3 digits, in Binary Coded Decimal. See also offset 02D6. A frequency of 1234.5 will have 0x0234 here and 0x0105 in offset 02D6.	**No**	**Ok**
02D6	2	[*FS2004 only*] Extended ADF2 frequency. The high byte contains the 1000's digit and the low byte the fraction, so, for a frequency of 1234.5 this offset will contain 0x0105.	**No**	**Ok**
02D8	2	[*FS2004 only*] ADF2: relative bearing to NDB (*360/65536 for degrees, –ve left, +ve right)	**No**	**Ok**
02DC	6	[*FS2004 only*] ADF2 IDENTITY (string supplied: 6 bytes including zero terminator)	**No**	**Ok**
02E2	25	[*FS2004 only*] ADF2 name (string supplied: 25 bytes including zero terminator)	**No**	**Ok**
02FB	1	[*FS2004 only*] ADF2 morse ID sound (1 = on, 0 = off), read for state, write to control	**No**	**Ok**
0300	2	VOR1 DME distance, 16-bit integer, nm * 10 [FS2002+]	**Ok**	**Ok**
0302	2	VOR1 DME speed, 16-bit integer, kts * 10 [FS2002+]	**Ok**	**Ok**
0304	2	VOR1 DME time to station, 16-bit integer, secs * 10 [FS2002+]	**Ok**	**Ok**
0306	2	VOR2 DME distance, 16-bit integer, nm * 10 [FS2002+]	**Ok**	**Ok**
0308	2	VOR2 DME speed, 16-bit integer, kts * 10 [FS2002+]	**Ok**	**Ok**
030A	2	VOR2 DME time to station, 16-bit integer, secs * 10 [FS2002+]	**Ok**	**Ok**
030C	4	Vertical speed, copy of offset 02C8 whilst airborne, not updated whilst the "on ground" flag (0366) is set. Can be used to check hardness of touchdown (but watch out for bounces which may change this). [FS2002+]	**Ok**	**Ok**
0310	8	FS2002 timer (double float, elapsed seconds including fractions, incremented each 'tick' – i.e. 1/18th sec). This runs all the time. It is used for all sorts of things, including the elapsed time between key/mouse-originated controls, to determine whether to accelerate inc/dec types. See also 0368,	**Only FS2002/4**	**Ok**
032C	2	"Plane is in fuel box" flag (same as Scenery BGL variable 0288)	**Ok**	**Ok**
0330	2	Altimeter pressure setting ("Kollsman" window). As millibars (hectoPascals) * 16	**Ok**	**Ok**
0338	2	Airframe can suffer damage if stressed (0=no, 1=yes)	**NO**	**NO**
033A	2	Manual fuel tank selection if set (appears to be standard anyway in FS2000)	**NO**	**NO**
033C	2	Engine stops when out of fuel if set	**NO**	**NO**
033E	2	Jet engine can flameout if set (appears not an option in FS2000?)	**NO**	**NO**
0340	2	Manual magneto controls if set (appears to be standard anyway in FS2000)	**NO**	**NO**
0342	2	Manual mixture control if set	**No**	**No**
034C	2	ADF1 Frequency: main 3 digits, in Binary Coded Decimal. See also offset 0356. A frequency of 1234.5 will have 0x0234 here and 0x0105 in offset 0356. (See also offset 0389)	**Ok**	**Ok**

034E	2	COM1 frequency, 4 digits in BCD format. A frequency of 123.45 is represented by 0x2345. The leading 1 is assumed.	**Ok**	**Ok**
0350	2	NAV1 frequency, 4 digits in BCD format. A frequency of 113.45 is represented by 0x1345. The leading 1 is assumed. (See also offset 0388)	**Ok**	**Ok**
0352	2	NAV2 frequency, 4 digits in BCD format. A frequency of 113.45 is represented by 0x1345. The leading 1 is assumed. (See also offset 0388)	**Ok**	**Ok**
0354	2	Transponder setting, 4 digits in BCD format: 0x1200 means 1200 on the dials.	**Ok**	**Ok**
0356	2	Extended ADF1 frequency. The high byte contains the 1000's digit and the low byte the fraction, so, for a frequency of 1234.5 this offset will contain 0x0105.	**Ok**	**Ok**
0358	2	COM frequency settable in 25KHz increments if true (else 50KHz)	**?**	
035C	2	ADF frequency settable in 100Hz increments if true (else 1KHz)	**?**	
0366	2	Aircraft on ground flag (0=airborne, 1=on ground). Not updated in Slew mode.	**Ok**	**Ok**
0368	4	Control timer 2 (see also 0310), a 32-bit 'float'.	**?**	
036C	1	Stall warning (0=no, 1=stall)	**Ok**	**Ok**
036D	1	Overspeed warning (0=no, 1=overspeed)	**Ok**	**Ok**
036E	1	Turn co-ordinator ball position (slip and skid). –128 is extreme left, +127 is extreme right, 0 is balanced.	**Ok**	**Ok**
0371	1	*Reserved (used in FSUIPC4)*		
0372	2	Reliability % (0–100). (Not sure if this is effective in FS2000)	**Ok**	**No**
0374	2	NAV1 or NAV2 select (256=NAV1, 512=NAV2)	**?**	
0378	2	DME1 or DME2 select (1=DME1, 2=DME2)	**?**	
037C	2	Turn Rate (for turn coordinator). 0=level, –512=2min Left, +512=2min Right	**Ok**	**Ok**
0388	1	NAV radio activation. If you change the NAV1 or NAV2 frequencies, writing 2 here makes FS re-scan for VORs to receive on those frequencies.	**?**	
0389	1	ADF radio activation. If you change the ADF frequency, writing 2 here makes FS re-scan for an NDB to receive on that frequency. (Although FS2000 seems to do this quite soon in any case)	**?**	
038A	1	COM radio activation. If you change the COM radio, writing a 1 here makes FS scan for ATIS broadcasts to receive on that frequency.	**?**	
03B0	8	Left aileron deflection, in radians, as a double floating point value	**?**	
03B8	8	Right aileron deflection, in radians, as a double floating point value	**?**	
03C0	64	The current state of the buttons on actively scanned joysticks (local ones, 0 to 15). Each of the 16 DWORDS contain the 32-bit state of the joystick 0-15, in order. Button 0 is the least significant bit (bit 0) in each DWORD.	**Ok**	**Ok**
0400	128	The filename of the last flight (or situation) saved, as an ASCII string with a zero terminator. The filetype (.flt or .stn) is not included. Use the counter at 3BD2 to determine when this has changed.	**Ok**	**Ok**
0480	8	Aileron trim axis input, 64-bit floating point (double), read-only	**Ok**	**Ok**
0488	8	Rudder trim axis input, 64-bit floating point (double), read-only	**Ok**	**Ok**
0490	8	Aileron trim axis required value, 64-bit floating point (double). If 2^0 is set in the byte at 04A0, then, when written, this value is copied to the FS trim (2EB0) instead of the value in 0480	**Ok**	**Ok**
0498	8	Rudder trim axis required value, 64-bit floating point (double). If 2^1 is set in the byte at 04A0, then, when written, this value is	**Ok**	**Ok**

		copied to the FS trim (2EC0) instead of the value in 0488		
04A0	1	Aileron and rudder trim connection control. See offsets 480–0498 above. 2^0 = 1 to disconnect aileron trim (2EB0) from FS 2^1 = 1 to disconnect rudder trim (2EC0) from FS This byte will be cleared and the connection restored (together with the most recent axis values) within about 10 seconds of it being written non-zero, so you need to write this every few seconds.	**Ok**	**Ok**
04A8	8	Elapsed seconds value, as a double. Accurate to fractions of a second but only updated frame by frame. This value counts simulated time, stopping in paused and menu modes, speeding up and slowing down according to the actual sim rate.	**Ok**	**Ok**
04B0	48	Area reserved by FSUIPC. (See details for user accessible parts earlier in this document). [FS2000 & CFS2 only]. The more useful ones follow:	**Ok**	**Ok**
04B4	2	FS2K ADVENTURE WEATHER: This provides the TEMPERATURE_SURFACE_ALT in metres. This is used to provide the METAR reporting station altitude so that the cloud bases can be converted to AGL.	**Ok**	**Ok**
04BA	2	FS2K ADVENTURE WEATHER: This provides the WIND_SURF_TURB which is used to provide the surface wind's upper gust speed in knots, with zero indicating no gusts.	**Ok**	**Ok**
04BC	2	FS2K ADVENTURE WEATHER: This provides the BAROMETRIC_DRIFT variable, which is used to provide the *difference* between the current aircraft position QNH (which may be in transition), and the METAR reported QNH as set by the weather control program. Adding this 'drift' value to the pressure will give the correct value for ATIS reports	**Ok**	**Ok**
04C0	2	FS2K ADVENTURE WEATHER: This provides the FSUIPC_VISIBILITY in statute miles * 100	**Ok**	**Ok**
04C2	2	FS2K ADVENTURE WEATHER: This provides the CLOUD_THUNDER_BASE in metres AMSL	**Ok**	**Ok**
04C4	2	FS2K ADVENTURE WEATHER: This provides the CLOUD_LOW_BASE in metres AMSL	**Ok**	**Ok**
04C6	2	FS2K ADVENTURE WEATHER: This provides the CLOUD_HIGH_BASE in metres AMSL	**Ok**	**Ok**
04C8	2	Dew point as degrees C *256, for the surface temperature layer, FS2k/CFS2 read only	**Ok**	**Ok**
04CB	1	Precipitation rate, 0–5, FS2k/CFS2 read only. *Note that in FS2004, rate 0 = light drizzle. Type=0 is no rain/snow*	**Ok**	**Ok**
04CC	1	Precipitation type, 0=none, 1=rain, 2=snow, FS2k/CFS2 read only.	**Ok**	**Ok**
04CD	1	FS2K ADVENTURE WEATHER: This provides the CLOUD_THUNDER_COVER 0–8	**Ok**	**Ok**
04CE	1	FS2K ADVENTURE WEATHER: This provides the CLOUD_LOW_COVER 0–8	**Ok**	**Ok**
04CF	1	FS2K ADVENTURE WEATHER: This provides the CLOUD_HIGH_COVER 0–8	**Ok**	**Ok**
04D2	2	Precipitation control: write hi-byte=type 0–2 (see above), low byte=rate 0–5. Write 0xFFFF to release control back to FS2k/CFS2.	**Ok**	**Ok**
04D4	2	Dew point control: degrees C * 256. Sets surface layer dewpoint only, FSUIPC does rest. Write 0x8000 to release control back to FS2k/CFS2.	**Ok**	**Ok**
04D8	2	Surface layer wind speed, in knots (FS2k/CFS2). This may be different to the **current** wind speed at the aircraft—see offset 0E90. This also provides WIND_SURF_VEL for FS2k Adventures.	**Ok**	**Ok**
04DA	2	Surface layer wind direction, *360/65536 to get degrees	**Ok**	**Ok**

		MAGNETIC (FS2k/CFS2). This may be different to the **current** wind direction at the aircraft—see offset 0E92. This also provides WIND_SURF_DIR for FS2k Adventures.		
04E0	88	Area reserved for Project Magenta	**Ok**	**Ok**
0538	8	Design speed VS0 (stall speed full flaps), ft/sec, as a double (64-bit floating point).	**No**	**Ok**
0540	8	Design speed VS1 (stall speed clean), ft/sec, as a double (64-bit floating point).	**No**	**Ok**
0548	8	Design speed VC (cruise speed), ft/sec, as a double (64-bit floating point).	**No**	**Ok**
0550	8	Minimum drag velocity, ft/sec, as a double (64-bit floating point).	**No**	**Ok**
0560	8	Latitude of aircraft in FS units. To convert to Degrees: *If your compiler supports **long long** (64-bit) integers* then use such a variable to simply copy this 64-bit value into a **double** floating point variable and multiply by 90.0/(10001750.0 * 65536.0 * 65536.0). *Otherwise* you will have to handle the high 32-bits and the low 32-bits separately, combining them into one **double** floating point value (say **dHi**). To do, copy the high part (the 32-bit **int** at 0564) to one **double** and the low part (the 32-bit **unsigned int** at 0560) to another (say **dLo**). Remember that the low part is only *part* of a bigger number, so doesn't have a sign of its own. Divide **dLo** by (65536.0 * 65536.0) to give it its proper magnitude compared to the high part, then either add it to or subtract it from **dHi** according to whether **dHi** is positive or negative. This preserves the integrity of the original positive or negative number. Finally multiply the result by 90.0/10001750.0 to get degrees. Either way, a negative result is South, positive North. [Can be written to move aircraft: in FS2002 only in slew or pause states]	**Ok but different**	**Ok (can set in all modes)**
0568	8	Longitude of aircraft in FS format. To convert to Degrees: *If your compiler supports **long long** (64-bit) integers* then use such a variable to simply copy this 64-bit value into a **double** floating point variable and multiply by 360.0/(65536.0 * 65536.0 * 65536.0 * 65536.0). *Otherwise* you will have to handle the high 32-bits and the low 32-bits separately, combining them into one **double** floating point value (say **dHi**). To do, copy the high part (the 32-bit **int** at 056C) to one **double** and the low part (the 32-bit **unsigned int** at 0568) to another (say **dLo**). Remember that the low part is only *part* of a bigger number, so doesn't have a sign of its own. Divide **dLo** by (65536.0 * 65536.0) to give it its proper magnitude compared to the high part, then either add it to or subtract it from **dHi** according to whether **dHi** is positive or negative. This preserves the integrity of the original positive or negative number. Finally multiply the result by 360.0/(65536.0 * 65536.0) to get degrees. Either way, a negative result is West, positive East. If you did it all unsigned then values over 180.0 represent West longitudes of (360.0 – the value). [Can be written to move aircraft: in FS2002 only in slew or pause states]	**Ok but different**	**Ok (can set in all modes)**

0570	8	Altitude, in metres and fractional metres. The units are in the high 32-bit integer (at 0574) and the fractional part is in the low 32-bit integer (at 0570). [Can be written to move aircraft: in FS2002 only in slew or pause states]	**Ok but different**	**Ok (can set in all modes)**
0578	4	Pitch, *360/(65536*65536) for degrees. 0=level, –ve=pitch up, +ve=pitch down [Can be set in slew or pause states]	**Ok**	**Ok (can set in all modes)**
057C	4	Bank, *360/(65536*65536) for degrees. 0=level, –ve=bank right, +ve=bank left [Can be set in slew or pause states]	**Ok**	**Ok (can set in all modes)**
0580	4	Heading, *360/(65536*65536) for degrees TRUE. [Can be set in slew or pause states]	**Ok**	**Ok (can set in all modes)**
05B0	24	The viewpoint Latitude (8 bytes), Longitude (8 bytes) and Altitude (8 bytes) in the same format as 0560–0577 above. This is read only and seems to relate to the position of the viewer whether in cockpit, tower or spot views.	**Ok**	**Ok**
05C8	4	The viewpoint Pitch, *360/(65536*65536) for degrees. 0=level, –ve=pitch up, +ve=pitch down. [Read only]	**Ok**	**Ok**
05CC	4	The viewpoint Bank, *360/(65536*65536) for degrees. 0=level, –ve=bank right, +ve=bank left. [Read only]	**Ok**	**Ok**
05D0	4	The viewpoint Heading, *360/(65536*65536) for degrees TRUE. [Read only]	**Ok**	**Ok**
05D4	2	Smoke system available if True	**NO**	
05D8	2	Smoke system enable: write 1 to switch on, 0 to switch off (see also 05D4)	**Ok**	**Ok**
05DC	2	Slew mode (indicator and control), 0=off, 1=on. (See 05DE also).	**Ok**	**Ok but not like FS2002**
05DE	2	Slew control: write non-zero value here *at same time* as changing 05DC above, and the Slew mode change includes the swapping of the assigned joystick axes. [ignored in FS2004 – the axes are swapped in any case. See offset 310B for control of axis connection in slew mode]	**Ok**	**No**
05E4	2	Slew roll rate: 0=static, –ve = right roll, +ve=left roll, rate is such that 192 gives a complete 360 roll in about one minute.	**Ok**	**Ok**
05E6	2	Slew yaw rate: 0=heading constant, –ve = right, +ve=left, rate is such that 24 gives a complete 360 turn in about one minute.	**Ok**	**Ok**
05E8	2	Slew vertical rate: 16384=no change, 16385–32767 increasing rate down, 16383–0 increasing rate up. One keypress on Q (up) or A (down) makes a change of 512 units.	**Ok**	**Ok**
05EB	1	Slew forward/backward movement: +ve=backward, –ve=forward. Values 1–127 give slow to fast slewing (–128 is the fastest forward slew).	**Ok**	**Ok**
05ED	1	Slew left/right movement: +ve=right, –ve=left. Values 1–127 give slow to fast sideways slewing (–128 is the fastest leftward slew).	**Ok**	**Ok**
05EE	2	Slew pitch rate: 16384=no change, <16384=pitch up, >16384 pitch down, range 0–32767.	**Ok**	**Ok**
05F4	2	Slew mode display: 0=off, 1=coords/hdg/spd, 2=fps, 3=all	**Ok**	**Ok**
05FC	2	Flight mode display: 0=off, 1=coords/hdg/spd, 2=fps, 3=all	**Ok**	**Ok**
0609	1	Engine type: 0=Piston (and some FS2004 Helos), 1=Jet, 2=Sailplane, 3=Helo, 4=Rocket, 5=Turboprop	**Ok**	**Ok**
060C	2	Gear type. 0=non-retractable standard, 1=retractable, 2=slides	**No**	**No**
060E	2	Retractable gear flag (0 if not, 1 if retractable)	**No**	**No**
0612	2	Display IAS if TRUE, TAS otherwise	**No**	**No**
0628	4	Instant replay flag & control, 1=on, 0=off. Can write to turn on and off whilst there is still time to play (see offset 062C)	**Ok**	**Ok**
062C	4	Instant replay: time left to run, in seconds. Whilst this is non-	**Ok**	**Ok**

		zero, the flag in offset 0628 controls the playback.		
06D0	144	Area used for operating, controlling and configuring the facilities in FSUIPC for feedback flight control (bank, pitch, speed, yaw). For full details of this please see the separate TXT documentation in the SDK.	**Ok**	**Ok**
0760	4?	Video recording flag, 1=on, 0=off	**Ok**	**Ok**
0764	4	Autopilot available	**Ok**	**Ok**
0768	4	Autopilot V/S hold available	**No**	**No**
076C	4	Autothrottle airspeed hold available	**No**	**No**
0770	4	Autothrottle mach hold available	**No**	**No**
0774	4	Autothrottle RPM hold available	**No**	**No**
0778	4	Flaps available	**Ok**	**Ok**
077C	4	Stall horn available	**Ok**	
0780	4	Engine mixture available	**Ok**	**Ok**
0784	4	Carb heat available	**Ok**	**Ok**
0788	4	Pitot heat available	**No**	**No**
078C	4	Spoiler available	**Ok**	**Ok**
0790	4	Aircraft is tail dragger	**Ok**	**Ok**
0794	4	Strobes available	**Ok**	**Ok**
0798	4	Prop type available	**No**	**No**
079C	4	Toe brakes available	**Ok**	**Ok**
07A0	4	NAV1 available	**Ok**	**Ok**
07A4	4	NAV2 available	**Ok**	**Ok**
07A8	4	Marker indicators available	**No**	**No**
07AC	4	NAV1 OBS available	**No**	**No**
07B0	4	NAV2 OBS available	**No**	**No**
07B4	4	VOR2 gauge available	**No**	**No**
07B8	4	Gyro drift available	**No**	**No**
07BC	4	Autopilot Master switch	**Ok**	**Ok**
07C0	4	Autopilot wing leveller	**Ok**	**Ok**
07C4	4	Autopilot NAV1 lock	**Ok**	**Ok**
07C8	4	Autopilot heading lock	**Ok**	**Ok**
07CC	2	Autopilot heading value, as degrees*65536/360	**Ok**	**Ok**
07D0	4	Autopilot altitude lock	**Ok**	**Ok**
07D4	4	Autopilot altitude value, as metres*65536	**Ok**	**Ok**
07D8	4	Autopilot attitude hold	**Ok**	**Ok**
07DC	4	Autopilot airspeed hold	**Ok**	**Ok**
07E2	2	Autopilot airspeed value, in knots	**Ok**	**Ok**
07E4	4	Autopilot mach hold	**Ok**	**Ok**
07E8	4	Autopilot mach value, as Mach*65536	**Ok**	**Ok**
07EC	4	Autopilot vertical speed hold [*Not connected in FS2002/4*]	**No**	**No**
07F2	2	Autopilot vertical speed value, as ft/min	**Ok**	**Ok**
07F4	4	Autopilot RPM hold	**?**	
07FA	2	Autopilot RPM value ??	**?**	
07FC	4	Autopilot GlideSlope hold N.B. In at least FS2002 and FS2004 (and maybe FS2000 as well) setting this also sets 0800, approach hold. To clear both you need to write 0 to them in the same FSUIPC process call, as if they are separated by an FS frame, an interlock stops them clearing.	**Ok**	**Ok**
0800	4	Autopilot Approach hold. See the note above, for offset 07FC.	**Ok**	**Ok**
0804	4	Autopilot Back course hold. The note for offset 07FC may also apply here.	**Ok**	**Ok**
0808	4	Yaw damper	**Ok**	**Ok**
080C	4	Autothrottle TOGA (take off power)	**Ok**	**Ok**
0810	4	Autothrottle Arm	**Ok**	**Ok**
0814	4	Flight analysis mode (0=0ff, 1=Landing, 2=Course tracking, 3=Manoevres)	**No**	

0830	4	Action on crash (0=ignore, 1=reset, 2=graph). [*Graph mode not applicable to FS2002*]	**Ok, but different**	**Ok As FS2002**
0834	4	[FS2002/4 only]: DME2 Latitude when available separately. Same units as in 085C above.	**Only**	**Yes, but it is position of actual antenna**
0838	4	[FS2002/4 only]: DME2 Longitude when available separately. Same units as in 0864 above.	**Only**	**Yes, but it is position of actual antenna**
083C	4	[FS2002/4 only]: DME2 elevation in metres when available separately.	**Only**	**Yes, but it is position of actual antenna**
0840	2	Crashed flag	**Ok**	**Ok**
0842	2	Vertical speed in metres per minute, but with –ve for UP, +ve for DOWN. Multiply by 3.28084 and reverse the sign for the normal fpm measure. This works even in slew mode (except in FS2002).	**Ok – but not in slew mode**	**Ok (in slew mode too!)**
0844	2	NAV2 ILS localiser inverse runway heading if VOR2 is ILS. Convert to degrees by *360/65536. This is 180 degrees different to the direction of flight to follow the localiser. [FS2002+]	**Ok**	**Ok**
0846	2	NAV2 ILS glideslope inclination if VOR2 is ILS. Convert to degrees by *360/65536. [FS2002+]	**Ok**	**Ok**
0848	2	Off-runway crash detection	**No**	**No**
084A	2	Can collide with dynamic scenery	**No**	
084C	4	VOR2 Latitude, as in 085C above, except when NAV2 is tuned to an ILS, in which case this gives the localiser Latitude. [FS2002 and later]	**Only**	**Yes, but it is position of actual antenna**
0850	4	[FS2002/4 only]: VOR2 Longitude, as in 0864 above, except when NAV2 is tuned to an ILS, in which case this gives the localiser Longitude.	**Only**	**Yes, but it is position of actual antenna**
0854	4	[FS2002/4 only]: VOR2 Elevation, as in 086C above, except when NAV2 is tuned to an ILS, in which case this gives the localiser Elevation.	**Only**	**Yes, but it is position of actual antenna**
0858	4	VOR2 Latitude in FS form. Convert to degrees by *90/10001750. If NAV2 is tuned to an ILS this gives the glideslope transmitter Latitude. [FS2002+]	**Ok**	**Yes, but it is position of actual antenna**
085C	4	VOR1 Latitude in FS form. Convert to degrees by *90/10001750.If NAV1 is tuned to an ILS this gives the glideslope transmitter Latitude.	**Ok**	**Yes, but it is position of actual antenna**
0860	4	VOR2 Longitude in FS form. Convert to degrees by *360/(65536*65536). If NAV2 is tuned to an ILS this gives the glideslope transmitter Longitude. [FS2002+]	**Ok**	**Yes, but it is position of actual antenna**
0864	4	VOR1 Longitude in FS form. Convert to degrees by *360/(65536*65536). If NAV1 is tuned to an ILS this gives the glideslope transmitter Longitude.	**Ok**	**Yes, but it is position of actual antenna**
0868	4	VOR2 Elevation in metres. If NAV2 is tuned to an ILS this gives the glideslope transmitter Elevation. [FS2002+]	**Ok**	**Yes, but it is position of actual antenna**

086C	4	VOR1 Elevation in metres. If NAV1 is tuned to an ILS this gives the glideslope transmitter Elevation.	**Ok**	**Yes, but it is position of actual antenna**
0870	2	NAV1 ILS localiser inverse runway heading if VOR1 is ILS. Convert to degrees by *360/65536. This is 180 degrees different to the direction of flight to follow the localiser.	**Ok**	**Ok**
0872	2	NAV1 ILS glideslope inclination if VOR1 is ILS. Convert to degrees by *360/65536	**Ok**	**Ok**
0874	4	VOR1 Latitude, as in 085C above, except when NAV1 is tuned to an ILS, in which case this gives the localiser Latitude. [FS2002 and later]	**Only**	**Yes, but it is position of actual antenna**
0878	4	[FS2002/4 only]: VOR1 Longitude, as in 0864 above, except when NAV1 is tuned to an ILS, in which case this gives the localiser Longitude.	**Only**	**Yes, but it is position of actual antenna**
087C	4	[FS2002/4 only]: VOR1 Elevation, as in 086C above, except when NAV1 is tuned to an ILS, in which case this gives the localiser Elevation.	**Only**	**Yes, but it is position of actual antenna**
0880	4	[FS2002/4 only]: DME1 Latitude when available separately. Same units as in 085C above.	**Only**	**Yes, but it is position of actual antenna**
0884	4	[FS2002/4 only]: DME1 Longitude when available separately. Same units as in 0864 above.	**Only**	**Yes, but it is position of actual antenna**
0888	1	Active engine (select) flags. Bit 0 = Engine 1 selected … Bit 3 = Engine 4 selected. See notes against offset 0892.	**Ok**	**Ok**
0889	1	Rotor clutch switch, when applicable. 1=On, 0=Off. Can be read and written. [FS2004 only]	**No**	**Ok**
088A	2	[FS2002/4 only]: DME1 Elevation in metres, when available separately.	**Only**	**Yes, but it is position of actual antenna**
088C	152	ENGINE 1 values, as detailed below		
088C	2	Engine 1 Throttle lever, –4096 to +16384 [Programs controlling throttle directly from user inputs should write to 089A instead if the input should be disconnectable via offset 310A (e.g. for auto-throttle management)]	**Ok**	**Ok**
088E	2	Engine 1 Prop lever, –4096 to +16384	**Ok**	**Ok**
0890	2	Engine 1 Mixture lever, 0 – 16384	**Ok**	**Ok**
0892	2	Engine 1 Starter switch position (Magnetos), Jet/turbojet: 0=Off, 1=Start, 2=Gen Prop: 0=Off, 1=right, 2=Left, 3=Both, 4=Start **Notes** (for FS2K/CFS2): • Don't forget to switch fuel on to start (mixture to max). • For FS2k type starting you need to set the 'Start' value here and monitor the combustion flag (below). When that is set, change the starter switch to another position (Both or Gen). FS98 models start immediately but you should still adopt the same procedure. • The Engine addressed by writes to this and the equivalent Engine 2–4 offsets will become *selected* (see 0888 above). It needs to stay selected during engine start, which means you can only start engines in sequence, not together. The original	**Ok**	**Ok**

		selection is restored automatically, however—but only when the starter is 'released' by writing a non-start value here. • FS98 prop planes transposed to FS2000 have misbehaving Magneto/Starter switch controls (whether FSUIPC is installed or not). You can start engines okay, but don't expect to be able to select the Magnetos reliably.		
0894	2	Engine 1 combustion flag (TRUE if engine firing)	**Ok**	**Ok**
0896	2	Engine 1 Jet N2 as 0 – 16384 (100%). This also appears to be the Turbine RPM % for proper helo models (and now also for the FS2004 Robinson model and derivatives)	**Ok**	**Ok**
0898	2	Engine 1 Jet N1 as 0 – 16384 (100%), or Prop RPM (derive RPM by multiplying this value by the RPM Scaler (see 08C8) and dividing by 65536). Note that Prop RPM is signed and negative for counter-rotating propellers. In FS2004 this also now gives the Robinson model's RPM, when scaled by the RPM scaler.	**Ok**	**Ok**
089A	2	Engine 1 Throttle lever, –4096 to +16384, same as 088C above except that values written here are treated like axis inputs and are disconnectable via offset 310A, and have the last written value obtainable from offset 3330	**Ok**	**Ok**
08A0	2	Engine 1 Fuel Flow PPH SSL (pounds per hour, standardised to sea level). Don't know units, but it seems to match some gauges if divided by 128. Not maintained in all cases.	**Ok**	**Ok**
08B2	2	Engine 1 Anti-Ice or Carb Heat switch (1=On)	**Ok**	**Ok**
08B8	2	Engine 1 Oil temperature, 16384 = 140 C.	**Ok**	**Ok**
08BA	2	Engine 1 Oil pressure, 16384 = 55 psi. Not that in some FS2000 aircraft (the B777) this can exceed the 16-bit capacity of this location. FSUIPC limits it to fit, i.e.65535 = 220 psi	**Ok**	**Ok**
08BC	2	Engine 1 Pressure Ratio (where calculated): 16384 = 1.60	**Ok**	**Ok**
08BE	2	Engine 1 EGT, 16384 = 860 C. [*Note that for Props this value is not actually correct. For FS2004 at least you will get the correct value from 3B70. In FS2004 the value here has been derived by FSUIPC to be compatible with FS2002 et cetera*]	**Ok**	**Ok, but see text**
08C0	2	Engine 1 Manifold Pressure: Inches Hg * 1024	**Ok**	**Ok**
08C8	2	Engine 1 RPM Scaler: For Props, use this to calculate RPM – see offset 0898 *(On turboprops this will give the shaft RPM, since there is currently no Gear Reduction Ratio available to fix values on such aircraft)*	**Ok**	**Ok**
08D0	4	Engine 1 Oil Quantity: 16384 = 100% On FS2000 FSUIPC usually has to derive this from a leakage value as it isn't provided directly.	**Ok**	**Ok**
08D4	4	Engine 1 Vibration: 16384 = 5.0. This is a relative measure of amplitude from the sensors on the engine which when too high is an indication of a problem. The value at which you should be concerned varies according to aircraft and engine.	**Ok**	**Ok**
08D8	4	Engine 1 Hydraulic pressure: appears to be 4*psi	**Ok**	**Ok**
08DC	4	Engine 1 Hydraulic quantity: 16384 = 100%	**Ok**	**Ok**
08E8	8	Engine 1 CHT, degrees F in double floating point (FLOAT64)	**Ok**	**Ok**
08F0	4	Engine 1 Turbine temperature: degree C *16384, valid for FS2004 helo models	**Ok**	**Ok**
08F4	4	Engine 1 Torque % (16384 = 100%), valid for FS2004 helo models	**Ok**	**Ok**
08F8	4	Engine 1 Fuel pressure, psf (i.e. psi*144): not all aircraft files provide this, valid for FS2004 helo models.	**Ok?**	**Ok**
08FC	2?	Engine 1 electrical load, possibly valid for FS2004 helo models.		**Ok?**
0900	4	Engine 1 Transmission oil pressure (psi * 16384): for helos	**Ok?**	**Ok**
0904	4	Engine 1 Transmission oil temperature (degrees C * 16384): for helos	**Ok?**	**Ok**

0908	4	Engine 1 Rotor RPM % (16384=100%): for helos	**Ok?**	**Ok**
0918	8	Engine 1 Fuel Flow Pounds per Hour, as floating point double (FLOAT64)	**Ok**	**Ok**
0920	4	Engine 1 Torque, in FLOAT32 format, probably in ft-lbs. (not jets)	**No**	**Ok**
0924	152	ENGINE 2 values, as detailed below	**As ENG1**	**As ENG1**
0924	2	Engine 2 Throttle lever, –4096 to +16384 [Programs controlling throttle directly from user inputs should write to 0932 instead if the input should be disconnectable via offset 310A (e.g. for auto-throttle management)]		
0926	2	Engine 2 Prop lever, –4096 to +16384		
0928	2	Engine 2 Mixture lever, 0 – 16384		
092A	2	Engine 2 Starter switch position (Magnetos), Jet/turbo: 0=Off, 1=Start, 2=Gen; Prop: 0=Off, 1=right, 2=Left, 3=Both, 4=Start (See **Notes** in Engine 1 entry)		
092C	2	Engine 2 combustion flag (TRUE if engine firing)		
092E	2	Engine 2 Jet N2 as 0 – 16384 (100%)		
0930	2	Engine 2 Jet N1 as 0 – 16384 (100%), or Prop RPM (derive RPM by multiplying this value by the RPM Scaler (see 08C8) and dividing by 65536). Note that Prop RPM is signed and negative for counter-rotating propellers.		
0932	2	Engine 2 Throttle lever, –4096 to +16384, same as 088C above except that values written here are treated like axis inputs and are disconnectable via offset 310A, and have the last written value obtainable from offset 3332	**Ok**	**Ok**
0938	2	Engine 2 Fuel Flow PPH SSL (pounds per hour, standardised to sea level). Don't know units, but it seems to match some gauges if divided by 128. Not maintained in all cases.		
094A	2	Engine 2 Anti-Ice or Carb Heat switch (1=On)		
0950	2	Engine 2 Oil temperature, 16384 = 140 C.		
0952	2	Engine 2 Oil pressure, 16384 = 55 psi. Not that in some FS2000 aircraft (the B777) this can exceed the 16-bit capacity of this location. FSUIPC limits it to fit, i.e.65535 = 220 psi		
0954	2	Engine 2 Pressure Ratio (where calculated): 16384 = 1.60		
0956	2	Engine 2 EGT, 16384 = 860 C. [*Note that for Props this value is not actually correct. For FS2004 at least you will get the correct value from 3AB0. In FS2004 the value here has been derived by FSUIPC to be compatible with FS2002 et cetera*]		
0958	2	Engine 2 Manifold Pressure: Inches Hg * 1024		
0960	2	Engine 2 RPM Scaler: For Props, use this to calculate RPM – see offset 0930		
0968	4	Engine 2 Oil Quantity: 16384 = 100% On FS2000 FSUIPC usually has to derive this from a leakage value as it isn't provided directly.		
096C	4	Engine 2 Vibration: 16384 = 5.0. This is a relative measure of amplitude from the sensors on the engine which when too high is an indication of a problem. The value at which you should be concerned varies according to aircraft and engine.		
0970	4	Engine 2 Hydraulic pressure: appears to be 4*psi		
0974	4	Engine 2 Hydraulic quantity: 16384 = 100%		
0980	8	Engine 2 CHT, degrees F in double floating point (FLOAT64)		
0988	4	Engine 2 Turbine temperature: degree C *16384		
098C	4	Engine 2 Torque % (16384 = 100%)		
0990	4	Engine 2 Fuel pressure, psf (i.e. psi*144): not all aircraft files provide this.		
0998	4	Engine 2 Transmission pressure (psi * 16384): for helos		
099C	4	Engine 2 Transmission temperature (degrees C * 16384): for helos		

09A0	4	Engine 2 Rotor RPM % (16384=100%): for helos		
09B0	8	Engine 2 Fuel Flow Pounds per Hour, as floating point double (FLOAT64)		
09B8	4	Engine 2 Torque, in FLOAT32 format, probably in ft-lbs. (not jets)		
09BC	152	ENGINE 3 values, as detailed below	**As ENG1**	**As ENG1**
09BC	2	Engine 3 Throttle lever, –4096 to +16384 [Programs controlling throttle directly from user inputs should write to 09CA instead if the input should be disconnectable via offset 310A/B (e.g. for auto-throttle management)]		
09BE	2	Engine 3 Prop lever, –4096 to +16384		
09C0	2	Engine 3 Mixture lever, 0 – 16384		
09C2	2	Engine 3 Starter switch position (Magnetos), Jet/turbo: 0=Off, 1=Start, 2=Gen; Prop: 0=Off, 1=right, 2=Left, 3=Both, 4=Start (see N**otes** in Engine 1 entry)		
09C4	2	Engine 3 combustion flag (TRUE if engine firing)		
09C6	2	Engine 3 Jet N2 as 0 – 16384 (100%)		
09C8	2	Engine 3 Jet N1 as 0 – 16384 (100%), or Prop RPM (derive RPM by multiplying this value by the RPM Scaler (see 08C8) and dividing by 65536). Note that Prop RPM is signed and negative for counter-rotating propellers.		
09CA	2	Engine 3 Throttle lever, –4096 to +16384, same as 088C above except that values written here are treated like axis inputs and are disconnectable via offset 310A/B, and have the last written value obtainable from offset 3334	**Ok**	**Ok**
09D0	2	Engine 3 Fuel Flow PPH SSL (pounds per hour, standardised to sea level). Don't know units, but it seems to match some gauges if divided by 128. Not maintained in all cases.		
09E2	2	Engine 3 Anti-Ice or Carb Heat switch (1=On)		
09E8	2	Engine 3 Oil temperature, 16384 = 140 C.		
09EA	2	Engine 3 Oil pressure, 16384 = 55 psi. Not that in some FS2000 aircraft (the B777) this can exceed the 16-bit capacity of this location. FSUIPC limits it to fit, i.e.65535 = 220 psi		
09EC	2	Engine 3 Pressure Ratio (where calculated): 16384 = 1.60		
09EE	2	Engine 3 EGT, 16384 = 860 C. [*Note that for Props this value is not actually correct. For FS2004 at least you will get the correct value from 39F0. In FS2004 the value here has been derived by FSUIPC to be compatible with FS2002 et cetera*]		
09F0	2	Engine 3 Manifold Pressure: Inches Hg * 1024		
09F8	2	Engine 3 RPM Scaler: For Props, use this to calculate RPM – see offset 09C8		
0A00	4	Engine 3 Oil Quantity: 16384 = 100% On FS2000 FSUIPC usually has to derive this from a leakage value as it isn't provided directly.		
0A04	4	Engine 3 Vibration: 16384 = 5.0. This is a relative measure of amplitude from the sensors on the engine which when too high is an indication of a problem. The value at which you should be concerned varies according to aircraft and engine.		
0A08	4	Engine 3 Hydraulic pressure: appears to be 4*psi		
0A0C	4	Engine 3 Hydraulic quantity: 16384 = 100%		
0A18	8	Engine 3 CHT, degrees F in double floating point (FLOAT64)		
0A20	4	Engine 3 Turbine temperature: degree C *16384		
0A24	4	Engine 3 Torque % (16384 = 100%)		
0A28	4	Engine 3 Fuel pressure, psf (i.e. psi*144): not all aircraft files provide this.		
0A30	4	Engine 3 Transmission pressure (psi * 16384): for helos		
0A34	4	Engine 3 Transmission temperature (degrees C * 16384): for helos		

0A38	4	Engine 3 Rotor RPM % (16384=100%): for helos		
0A48	8	Engine 3 Fuel Flow Pounds per Hour, as floating point double (FLOAT64)		
0A50	4	Engine 3 Torque, in FLOAT32 format, probably in ft-lbs. (not jets)		
0A54	152	ENGINE 4 values, as detailed below	**As ENG1**	**As ENG1**
0A54	2	Engine 4 Throttle lever, –4096 to +16384 [Programs controlling throttle directly from user inputs should write to 0A62 instead if the input should be disconnectable via offset 310A/B (e.g. for auto-throttle management)]		
0A56	2	Engine 4 Prop lever, –4096 to +16384		
0A58	2	Engine 4 Mixture lever, 0 – 16384		
0A5A	2	Engine 4 Starter switch position (Magnetos), Jet/turbo: 0=Off, 1=Start, 2=Gen; Prop: 0=Off, 1=right, 2=Left, 3=Both, 4=Start (see **Notes** in Engine 1 entry)		
0A5C	2	Engine 4 combustion flag (TRUE if engine firing)		
0A5E	2	Engine 4 Jet N2 as 0 – 16384 (100%)		
0A60	2	Engine 4 Jet N1 as 0 – 16384 (100%), or Prop RPM (derive RPM by multiplying this value by the RPM Scaler (see 08C8) and dividing by 65536). Note that Prop RPM is signed and negative for counter-rotating propellers.		
0A62	2	Engine 4 Throttle lever, –4096 to +16384, same as 088C above except that values written here are treated like axis inputs and are disconnectable via offset 310A/B, and have the last written value obtainable from offset 3336	**Ok**	**Ok**
0A68	2	Engine 4 Fuel Flow PPH SSL (pounds per hour, standardised to sea level). Don't know units, but it seems to match some gauges if divided by 128. Not maintained in all cases.		
0A7A	2	Engine 4 Anti-Ice or Carb Heat switch (1=On)		
0A80	2	Engine 4 Oil temperature, 16384 = 140 C.		
0A82	2	Engine 4 Oil pressure, 16384 = 55 psi. Not that in some FS2000 aircraft (the B777) this can exceed the 16-bit capacity of this location. FSUIPC limits it to fit, i.e.65535 = 220 psi		
0A84	2	Engine 4 Pressure Ratio (where calculated): 16384 = 1.60		
0A86	2	Engine 4 EGT, 16384 = 860 C. [*Note that for Props this value is not actually correct. For FS2004 at least you will get the correct value from 3930. In FS2004 the value here has been derived by FSUIPC to be compatible with FS2002 et cetera*]		
0A88	2	Engine 4 Manifold Pressure: Inches Hg * 1024		
0A90	2	Engine 4 RPM Scaler: For Props, use this to calculate RPM – see offset 0A60		
0A98	4	Engine 4 Oil Quantity: 16384 = 100% On FS2000 FSUIPC usually has to derive this from a leakage value as it isn't provided directly.		
0A9C	4	Engine 4 Vibration: 16384 = 5.0. This is a relative measure of amplitude from the sensors on the engine which when too high is an indication of a problem. The value at which you should be concerned varies according to aircraft and engine.		
0AA0	4	Engine 4 Hydraulic pressure: appears to be 4*psi		
0AA4	4	Engine 4 Hydraulic quantity: 16384 = 100%		
0AB0	8	Engine 4 CHT, degrees F in double floating point (FLOAT64)		
0AB8	4	Engine 4 Turbine temperature: degree C *16384		
0ABC	4	Engine 4 Torque % (16384 = 100%)		
0AC0	4	Engine 4 Fuel pressure, psf (i.e. psi*144): not all aircraft files provide this.		
0AC8	4	Engine 4 Transmission pressure (psi * 16384): for helos		
0ACC	4	Engine 4 Transmission temperature (degrees C * 16384): for helos		

0AD0	4	Engine 4 Rotor RPM % (16384=100%): for helos	I	I
0AE0	8	Engine 4 Fuel Flow Pounds per Hour, as floating point double (FLOAT64)	I	I
0AE8	4	Engine 4 Torque, in FLOAT32 format, probably in ft-lbs. (not jets)	I	I
0AEC	2	Number of Engines	**Ok**	**Ok**
0AF0	2	Propeller pitch control: 0=Fixed, 1=Auto, 2=Manual. On FS2004 this is 0=fixed pitch, 1=constant speed, no differentiation between auto and manual.	**Ok**	**Ok but different**
0AF4	2	Fuel weight as pounds per gallon * 256	**Ok**	**Ok**
0AF8	2	Fuel tank selector: 0=None, 1=All, 2=Left, 3=Right, 4=LeftAux, 5=RightAux, 6=Centre, 7=Centre2, 8=Centre3, 9=External1, 10=External2, 11=Right Tip, 12=Left Tip, 14=Crossfeed LtoR, 15=Crossfeed RtoL. According to information received, in FS2002 all of these except the wing tip tanks can be selected and drained.	**Ok**	**Ok**
0B00	2	Throttle lower limit, 16384=100%. (e.g. for aircraft with reverse thrust this is normally: –4096 indicating 25% in reverse)	**Ok**	**Ok**
0B0C	4	Mach Max Operating speed *20480	**Ok**	**No**
0B18	8	Gyro suction in inches of mercury (Hg), floating point double (FLOAT64)	**Ok**	**Ok**
0B20	2	Sound control: 0 to switch off, 1 to switch on	**Ok**	**Ok**
0B24	2	Sound flag: reads 0 is off, 1 if on	**Ok**	**Ok**
0B4C	2	Ground altitude (metres). See 0020 for more accuracy.	**Ok**	**Ok**
0B60	2	Scenery complexity level, 0 – 4 in FS98, 0 – 5 in FS2000 on	**Ok**	**Ok**
0B64	1	Fail mode: 0 ok, ADF inoperable = 1 (both ADFs on FS2004)	**Ok**	
0B65	1	Fail mode: 0 ok, ASI inoperable = 1	**Ok**	
0B66	1	Fail mode: 0 ok, Altimeter inoperable = 1	**Ok**	
0B67	1	Fail mode: 0 ok, Attitude Indicator inoperable = 1	**Ok**	
0B68	1	Fail mode: 0 ok, COM1 radio inoperable = 1 See also 3BD6 (FS2002/FS2004)	**Ok**	
0B69	1	Fail mode: 0 ok, Mag Compass inoperable = 1	**Ok**	
0B6A	1	Fail mode: 0 ok, Electrics inoperable = 1	**Ok**	
0B6B	1	Fail mode: 0 ok, Engine inoperable = 1, extended for FS2000/CFS2 for up to 4 individual engines: bit 0 =Engine 1 … bit 3= Engine 4. (*but note that this may not work for FS98 aircraft transposed into FS2k/CFS2*).	**Ok**	
0B6C	1	Fail mode: 0 ok, Fuel indicators inoperable = 1	**Ok**	
0B6D	1	Fail mode: 0 ok, Direction Indicator inoperable = 1	**Ok**	
0B6E	1	Fail mode: 0 ok, VSI inoperable = 1	**Ok**	
0B6F	1	Fail mode: 0 ok, Transponder inoperable = 1	**Ok**	
0B70	1	Fail mode: 0 ok, NAV radios inoperable = 1 (NAV1 only in FS2002 and FS2004: see also 3BD6)	**Ok**	
0B71	1	Fail mode: 0 ok, Pitot inoperable = 1	**Ok**	
0B72	1	Fail mode: 0 ok, Turn coordinator inoperable = 1	**Ok**	
0B73	1	Fail mode: 0 ok, Vacuum inoperable = 1	**Ok**	
0B74	4	Fuel: centre tank level, % * 128 * 65536	**Ok**	**Ok**
0B78	4	Fuel: centre tank capacity: US Gallons (see also offsets 1244– for extra FS2k/CFS2 fuel tanks)	**Ok**	**Ok**
0B7C	4	Fuel: left main tank level, % * 128 * 65536	**Ok**	**Ok**
0B80	4	Fuel: left main tank capacity: US Gallons	**Ok**	**Ok**
0B84	4	Fuel: left aux tank level, % * 128 * 65536	**Ok**	**Ok**
0B88	4	Fuel: left aux tank capacity: US Gallons	**Ok**	**Ok**
0B8C	4	Fuel: left tip tank level, % * 128 * 65536	**Ok**	**Ok**
0B90	4	Fuel: left tip tank capacity: US Gallons	**Ok**	**Ok**
0B94	4	Fuel: right main tank level, % * 128 * 65536	**Ok**	**Ok**
0B98	4	Fuel: right main tank capacity: US Gallons	**Ok**	**Ok**

0B9C	4	Fuel: right aux tank level, % * 128 * 65536	**Ok**	**Ok**
0BA0	4	Fuel: right aux tank capacity: US Gallons	**Ok**	**Ok**
0BA4	4	Fuel: right tip tank level, % * 128 * 65536	**Ok**	**Ok**
0BA8	4	Fuel: right tip tank capacity: US Gallons	**Ok**	**Ok**
0BAC	2	Inner Marker: activated when TRUE	**Ok**	**Ok**
0BAE	2	Middle Marker: activated when TRUE	**Ok**	**Ok**
0BB0	2	Outer Marker: activated when TRUE	**Ok**	**Ok**
0BB2	2	Elevator control input: –16383 to +16383	**Ok**	**Ok**
0BB4	2	Elevator position indicator (maybe adjusted from input!)	**Ok**	**Ok**
0BB6	2	Aileron control input: –16383 to +16383	**Ok**	**Ok**
0BB8	2	Aileron position indicator (maybe adjusted from input!)	**Ok**	**Ok**
0BBA	2	Rudder control input: –16383 to +16383	**Ok**	**Ok**
0BBC	2	Rudder position indicator (maybe adjusted from input!)	**Ok**	**Ok**
0BBE	2	Helo pitch (elevator) trim control: –16383 to +16383, but only when "ApplyHeloTrim" set.	**Ok**	**Ok**
0BC0	2	Elevator trim control input: –16383 to +16383	**Ok**	**Ok**
0BC2	2	Elevator trim indicator (follows input)	**Ok**	**Ok**
0BC4	2	Left brake application read-out (0 off, 16383 full: parking brake=16383). You can apply a fixed brake pressure here, or else use the byte at 0C01 to apply brakes emulating the keypress. [*Note: In FS2002* ***reading*** *this ranges up to 32767, i.e. twice the written value.*]	**Ok (but different)**	**Ok**
0BC6	2	Right brake application read-out (0 off, 16383 full: parking brake=16383). You can apply a fixed brake pressure here, or else use the byte at 0C00 to apply brakes emulating the keypress. [*Note: In FS2002* ***reading*** *this ranges up to 32767, i.e. twice the written value.*]	**Ok (but different)**	**Ok**
0BC8	2	Parking brake: 0=off, 32767=on	**Ok**	**Ok**
0BCA	2	Braking indicator: brake applied if non-zero (16383=on, 0=off). *Note that in FS2002 this is artificially created by FSUIPC from the previous three settings.*	**Ok (but different)**	**Ok**
0BCC	4	Spoilers arm (0=off, 1=arm for auto deployment)	**Ok**	**Ok**
0BD0	4	Spoilers control, 0 off, 4800 arm, then 5620 (7%) to 16383 (100% fully deployed). The 4800 value is set by arming. Values from 0 to somewhere close to, but below, 4800 do nothing. The percentage extension is the proportion of the distance in the range 4800 to 16383, even though values 4800 to 5619 cannot be used—7% seems to be the minimum. [These details have now been verified on FS2000, FS2002 and FS2004.]	**Ok**	**Ok**
0BD4	4	Spoiler Left position indicator (0-16383)	**Ok**	**Ok**
0BD8	4	Spoiler Right position indicator (0-16383)	**Ok**	**Ok**
0BDC	4	Flaps control, 0=up, 16383=full. The "notches" for different aircraft are spaced equally across this range: calculate the increment by 16383/(number of positions-1), ignoring fractions. See also offset 3BFA below. N.B. Do not expect to read this and see 100% accurate values. For example, 3x2047=6141 for the 3rd détente up. But FS2000, at least, stores the flaps lever position in the FLT file as a % of 16384, and the percentage is stored to two decimal places. 6141 gets saved as 37.48% which converts back to 6140.7232 and this gets truncated here as 6140. However, 6140/2047 = 2.9995 which is as close as you need. Just round if you are using integers.	**Ok**	**Ok**
0BE0	4	Flaps position indicator (left). Note that in FS2002 and FS2004 this gives the correct proportional amount, with 16383=full	**Ok but different**	**Ok**

		deflection. It doesn't correspond to the equally spaced notches used for the control lever. If you know the maximum deflection angle you can derive the current angle by ((max * position indicator) / 16383). Also, in FS2002 and FS2004 this only gives the inboard trailing edge flaps. Please see offsets 30E0–30FF for greater details where needed.		
0BE4	4	Flaps position indicator (right). Note that in FS2002 and FS2004 this gives the correct proportional amount, with 16384=full deflection. It doesn't correspond to the equally spaced notches used for the control lever. Also, in FS2002 and FS2004 this only gives the inboard trailing edge flaps. Please see offsets 30E0–30FF for greater details where needed.	**Ok but different**	**Ok**
0BE8	4	Gear control: 0=Up, 16383=Down	**Ok**	**Ok**
0BEC	4	Gear position (nose): 0=full up, 16383=full down	**Ok**	**Ok**
0BF0	4	Gear position (right): 0=full up, 16383=full down	**Ok**	**Ok**
0BF4	4	Gear position (left): 0=full up, 16383=full down	**Ok**	**Ok**
0BF8	4	Unlimited visibility value, as 1600* statute miles. This is the value set in the Display Quality Settings.	**Ok**	**Ok**
0C00	1	Right toe brake control: 0 – 200, proportional braking with timed decay	**Ok**	**Ok**
0C01	1	Left toe brake control: 0 –200, proportional braking with timed decay	**Ok**	**Ok**
0C06	2	Helo bank (aileron) trim control: –16383 to +16383, but only when "ApplyHeloTrim" set to 'Both'.	**Ok**	**Ok**
0C08	2	Steering tiller input value (FSUIPC optional axis), -16384 to +16383, if calibrated	**Ok**	**Ok**
0C0A	2	Rudder input value, -16384 to +16383, if calibrated	**Ok**	**Ok**
0C18	2	International units: 0=US, 1=Metric+feet, 2=Metric+metres	**Ok**	**Ok**
0C1A	2	Simulation rate *256 (i.e. 256=1x)	**Ok**	**Ok**
0C20	9	Local time in character format: "hh:mm:ss" (with zero terminator)	**Ok**	**Ok**
0C29	5	DME1 distance as character string, either "nn.n" or "nnn." (when > 99.9 nm). The 5th character may be a zero or a space. Don't rely on it.	**Ok**	**Ok**
0C2E	5	DME1 speed as character string, "nnn" followed by either space then zero or just zero.	**Ok**	**Ok**
0C33	5	DME2 distance as character string, either "nn.n" or "nnn." (when > 99.9 nm). The 5th character may be a zero or a space. Don't rely on it.	**Ok**	**Ok**
0C38	5	DME2 speed as character string, "nnn" followed by either space then zero or just zero.	**Ok**	**Ok**
0C3E	2	Gyro drift amount (*360/65536 for degrees).	**Yes**	
0C44	2	Realism setting, 0 – 100	**No**	**No**
0C46	2	Realism options, bits allocated (but not all used in FS2K, necessarily): 0 ?? 1 elevator trim ratchets (?) 2 gyro drifts [FS2k ok] 3 lights burn out 4 fast throttle kills engine (?) 5 manual light control for instruments [FS2k ok] 6 pressure drifts (unlikely to apply to FS2k]	**No?**	
0C48	1	NAV1 Localiser Needle: –127 left to +127 right	**Ok**	**Ok**
0C49	1	NAV1 Glideslope Needle: –127 up to +127 down	**Ok**	**Ok**
0C4A	1	NAV1 Back Course flags:	**Ok**	

		0 BC available 1 Localiser tuned in 2 On Back Course (?) 7 Station active (even if no BC)		
0C4B	1	NAV1 To/From flag: 0=not active, 1=To, 2=From	**Ok**	**Ok**
0C4C	1	NAV1 GS flag: TRUE if GS alive	**Ok**	**Ok**
0C4D	1	NAV1 code flags, bits used as follows: 0 DME available 1 TACAN 2 Voice available 3 No signal available 4 DME transmitter at GS transmitter 5 No back course 6 GS available 7 This is a localiser (else it's a VOR) *[FS2002+, Not yet tested]*		
0C4E	2	NAV1 OBS setting (degrees, 0–359)	**Ok**	**Ok**
0C50	2	NAV1 radial (*360/65536 for degrees). Note that this is in degrees Magnetic for a VOR, but TRUE for an ILS LOC.	**Ok**	**Ok**
0C52	4	NAV1 signal strength: For Localisers, seems to be either 0 or 256 For VORs varies from 0 to over 1,000,000 when really close!		
0C56	2	NAV1: relative bearing to VOR1, in degrees (0–359)	**Ok**	**Ok**
0C59	1	NAV2 Localiser Needle: –127 left to +127 right	**Ok**	**Ok**
0C5A	1	NAV2 Back Course flags: 0 BC available 1 Localiser tuned in 2 On Back Course (?) 7 Station active (even if no BC)	**Ok**	**Ok**
0C5B	1	NAV2 To/From flag: 0=not active, 1=To, 2=From	**Ok**	**Ok**
0C5C	2	NAV2: relative bearing to VOR2, in degrees (0–359)	**Ok**	**Ok**
0C5E	2	NAV2 OBS setting (degrees, 0–359)	**Ok**	**Ok**
0C60	2	NAV2 radial (*360/65536 for degrees). Note that this is in degrees Magnetic for a VOR, but TRUE for an ILS LOC.	**Ok**	**Ok**
0C62	4	NAV2 signal strength: For Localisers, seems to be either 0 or 256 For VORs varies from 0 to over 1,000,000 when really close!		
0C6A	2	ADF1: relative bearing to NDB (*360/65536 for degrees, –ve left, +ve right)	**Ok**	**Ok**
0C6C	2	ADF1: dial bearing, where adjustable (in degrees, 1–360)	**Ok**	**Ok**
0C6E	1	NAV2 Glideslope Needle: –127 up to +127 down [FS2002+]	**Ok**	**Ok**
0C6F	1	NAV2 GS flag: TRUE if GS alive [FS2002+]	**Ok**	**Ok**
0C70	1	NAV2 code flags, bits used as follows: 8 DME available 9 TACAN 10 Voice available 11 No signal available 12 DME transmitter at GS transmitter 13 No back course 14 GS available 15 This is a localiser (else it's a VOR) *[FS2002+, Not yet tested]*		
0C92	2	Texture quality, 0–3, as on FS2K's slider in Display Quality	**Ok**	**No**
0D0C	2	Lights (FS2k/CFS2), a switch for each one (bits from lo to hi): 0 Navigation 1 Beacon 2 Landing 3 Taxi 4 Strobes	**Ok**	**Ok**

		5 Instruments 6 Recognition 7 Wing 8 Logo 9 Cabin		
0D50	24	The Tower Latitude (8 bytes), Longitude (8 bytes) and Altitude (8 bytes) in the same format as 0560–0577 above.	**Ok**	**Ok**
0D6C	4	Parameter associated with any Macro or Lua call sent to the following offset (0D70)	**Ok**	**Ok**
0D70	40	Write here the complete identity string of a Macro control or Lua program control in order to have FSUIPC execute it. For a Macro, the string should begin with up to 16 characters giving the .MCRO file name (just the name part, not the type), and then, separated by a ':' character, the macro name within that file—again, up to 16 characters. Spaces either side of the ':' are optional. For a Lua program operation, the actual Lua control should be provided, followed (with one space or ':' separator) by the Lua program name (without the .Lua suffix). The valid Lua controls are: Lua, LuaDebug, LuaKill, LuaSet, LuaClear, LuaToggle Note that a parameter should always be written first for the Set, Clear and Toggle controls as this specifies the flag to be changed (0–31). A parameter is never used with "Lua Kill". If a parameter is to be supplied, it should first be written to offset 0D6C, above. Otherwise whatever was last written there will be supplied.	**Ok**	**Ok**
0D98	2	International N/S setting: 2=North, 3=South	**Ok**	
0D9C	2	International E/W setting: 0=East, 1=West	**Ok**	
0DD6	2	Scenery BGL variable "usrvar" (originally 0312h in BGL) *(*In FS2004 this is moved to globals by FSUIPC unless 'MoveBGLvariables=No')*	**Ok**	**Ok***
0DD8	2	Scenery BGL variable "usrvr2" (originally 0314h in BGL) *(*In FS2004 this is moved to globals by FSUIPC unless 'MoveBGLvariables=No')*	**Ok**	**Ok***
0DDA	2	Scenery BGL variable "usrvr3" (originally 0316h in BGL) *(*In FS2004 this is moved to globals by FSUIPC unless 'MoveBGLvariables=No')*	**Ok**	**Ok***
0DDC	2	Scenery BGL variable "usrvr4" (originally 0318h in BGL) *(*In FS2004 this is moved to globals by FSUIPC unless 'MoveBGLvariables=No')*	**Ok**	**Ok***
0DDE	2	Scenery BGL variable "usrvr5" (originally 031Ah in BGL) *(*In FS2004 this is moved to globals by FSUIPC unless 'MoveBGLvariables=No')*	**Ok**	**Ok***
0DE2	2	Scenery BGL variable "spar10" (originally 031Eh in BGL) *(doesn't appear to be supported at all in FS2004)*	**Ok**	**No**
0DE4	2	Scenery BGL variable "spar11" (originally 0320h in BGL) *(doesn't appear to be supported at all in FS2004)*	**Ok**	**No**
0DE6	2	Scenery BGL variable "spar12" (originally 0322h in BGL) *(doesn't appear to be supported at all in FS2004)*	**Ok**	**No**
0DE8	2	Scenery BGL variable "spar13" (originally 0324h in BGL) *(doesn't appear to be supported at all in FS2004)*	**Ok**	**No**
0DEA	2	Scenery BGL variable "spar14" (originally 0326h in BGL) *(doesn't appear to be supported at all in FS2004)*	**Ok**	**No**
0DEC	2	Scenery BGL variable "spar15" (originally 0328h in BGL) *(doesn't appear to be supported at all in FS2004)*	**Ok**	**No**

0DEE	2	Scenery BGL variable "spar16" (originally 032Ah in BGL) *(doesn't appear to be supported at all in FS2004)*	**Ok**	**No**
0DF0	2	Scenery BGL variable "spar17" (originally 032Ch in BGL) *(doesn't appear to be supported at all in FS2004)*	**Ok**	**No**
0DF2	2	Scenery BGL variable "spar18" (originally 032Eh in BGL) *(doesn't appear to be supported at all in FS2004)*	**Ok**	**No**
0E5A	2	EFIS active (1=enabled)	**?**	
0E5C	2	EFIS VOR/ILS elevation in metres	**?**	
0E5E	2	EFIS density: 0=thin, 1=medium, 2=thick	**?**	
0E60	2	EFIS range: 0=short, 1=medium, 2=long	**?**	
0E62	2	EFIS mode: 0=normal, 1=reset, 2=plot intercept	**?**	
0E64	2	EFIS via VOR (2) or ILS (4)	**?**	
0E66	2	EFIS NAV select (1 or 2)	**?**	
0E68	2	EFIS display type: 0=rectangles, 1=telegraph poles, 2=yellow brick road	**?**	
0E8A	2	Current visibility (Statue miles * 100)	**Ok**	**Ok**
0E8C	2	Outside Air Temperature (OAT), degrees C * 256	**Ok**	**Ok**
0E8E	2	Dew point, degrees C * 256 [FS2004 only]. This is interpolated for the aircraft altitude from the temperature layer specifications below and above.	**No**	**Ok**
0E90	2	Ambient wind speed (at aircraft) in knots	**Ok**	**Ok**
0E92	2	Ambient wind direction (at aircraft), *360/65536 to get degrees Magnetic *or* True. For compatibility with FS98, the direction is Magnetic for surface winds (aircraft below the altitude set into offset 0EEE), but True for all upper winds. See offset 02A0 for magnetic variation and how to convert.	**Ok**	**Ok**
0E9A	112	Current Weather as Set: details follow. [See 0F1C for Global weather setting area] *On FS2000/CFS2 FSUIPC maps writes to this area to the Global weather area starting at 0F1C, and reads from the Global weather area to this Current weather area. Therefore you may not always read back what you last wrote. The main differences occur when FS local weather is in operation.* N.B. See also 0E8A above, which is the "current" visibility equivalent of the global setting at 0F8C.	**Ok**	**Ok**
0E9A	2	Upper cloud layer ceiling in metres AMSL	**Ok**	**Ok**
0E9C	2	Upper cloud layer base in metres AMSL	**Ok**	**Ok**
0E9E	2	Upper cloud layer coverage, 65535 = 8 oktas, … 32768= 4 oktas … 0 = clear	**Ok**	**Ok**
0EA0	2	Upper cloud layer, cloud altitude variation (metres)	**Ok**	**Ok**
0EA2	2	Lower cloud layer ceiling in metres AMSL	**Ok**	**Ok**
0EA4	2	Lower cloud layer base in metres AMSL	**Ok**	**Ok**
0EA6	2	Lower cloud layer coverage, 65535 = 8 oktas, … 32768= 4 oktas … 0 = clear	**Ok**	**Ok**
0EA8	2	Lower cloud layer, cloud altitude variation (metres)	**Ok**	**Ok**
0EAA	2	Storm layer ceiling in metres AMSL	**Ok**	**Ok**
0EAC	2	Storm layer base in metres AMSL (if a Storm layer is present, it must be the lowest, below "Lower Cloud").	**Ok**	**Ok**
0EAE	2	Storm cloud layer coverage, 65535 = 8 oktas, … 32768= 4 oktas … 0 = clear	**Ok**	**Ok**
0EB0	2	Storm cloud layer, cloud altitude variation (metres)	**Ok**	**Ok**
0EB2	2	Upper Temperature level, metres AMSL	**Ok**	**Ok**
0EB4	2	Upper Temperature in degrees C * 256	**Ok**	**Ok**
0EB6	2	Middle Temperature level, metres AMSL	**Ok**	**Ok**
0EB8	2	Middle Temperature in degrees C * 256	**Ok**	**Ok**
0EBA	2	Lower Temperature level, metres AMSL	**Ok**	**Ok**
0EBC	2	Lower Temperature in degrees C * 256	**Ok**	**Ok**

0EBE	2	Surface Temperature level, metres AMSL (best to be the ground elevation)	Ok	Ok
0EC0	2	Surface Temperature in degrees C * 256	Ok	Ok
0EC2	2	Temperature drift, degrees C *256 (not used in FS2k/CFS2?)	Ok	Ok
0EC4	2	Temperature day/night variation, degrees C *256	Ok	Ok
0EC6	2	Pressure (QNH) as millibars (hectoPascals) *16.	Ok	Ok
0EC8	2	Pressure drift as millibars *16 (not used on FS2k/CFS2?)	Ok	Ok
0ECA	2	Upper wind ceiling, metres AMSL	Ok	Ok
0ECC	2	Upper wind base, metres AMSL	Ok	Ok
0ECE	2	Upper wind speed, knots	Ok	Ok
0ED0	2	Upper wind direction, *360/65536 gives degrees True	Ok	Ok
0ED2	2	Upper wind turbulence setting, 0 none, 64, 128, 192, 224, 255 worst	Ok	Ok
0ED4	2	Upper wind gusts, enabled if True.	Ok	Ok
0ED6	2	Middle wind ceiling, metres AMSL	Ok	Ok
0ED8	2	Middle wind base, metres AMSL	Ok	Ok
0EDA	2	Middle wind speed, knots	Ok	Ok
0EDC	2	Middle wind direction, *360/65536 gives degrees True	Ok	Ok
0EDE	2	Middle wind turbulence setting, 0 none, 64, 128, 192, 224, 255 worst	Ok	Ok
0EE0	2	Middle wind gusts, enabled if True.	Ok	Ok
0EE2	2	Lower wind ceiling, metres AMSL	Ok	Ok
0EE4	2	Lower wind base, metres AMSL	Ok	Ok
0EE6	2	Lower wind speed, knots	Ok	Ok
0EE8	2	Lower wind direction, *360/65536 gives degrees True	Ok	Ok
0EEA	2	Lower wind turbulence setting, 0 none, 64, 128, 192, 224, 255 worst	Ok	Ok
0EEC	2	Lower wind gusts, enabled if True.	Ok	Ok
0EEE	2	Surface wind ceiling, metres AGL	Ok	Ok
0EF0	2	Surface wind speed, knots. [See also 04D8]	Ok	Ok
0EF2	2	Surface wind direction, *360/65536 gives degrees Magnetic (!). [See also 04DA]	Ok	Ok
0EF4	2	Surface wind turbulence setting, 0 none, 64, 128, 192, 224, 255 worst	Ok	Ok
0EF6	2	Surface wind gusts, enabled if True.	Ok	Ok
0EF8	2	Upper cloud layer type: 0=user-defined, 1=cirrus, 8=stratus, 9=cumulus	Ok	Ok
0EFA	2	Upper cloud layer icing: enabled if True	Ok	Ok
0EFC	2	Upper cloud layer turbulence (0 to 255 I think). Divided into steps by FSUIPC for FS2k/CFS2.	Ok	Ok
0EFE	2	Lower cloud layer type: 0=user-defined, 1=cirrus, 8=stratus, 9=cumulus	Ok	Ok
0F00	2	Lower cloud layer icing: enabled if True	Ok	Ok
0F02	2	Lower cloud layer turbulence (0 to 255 I think). Divided into steps by FSUIPC for FS2k/CFS2.	Ok	Ok
0F04	2	Storm layer type: 10=storm. [FSUIPC allows this to be a third and lowest layer of any type, for FS2k/CFS2, so then: 0=user-defined, 1=cirrus, 8=stratus, 9=cumulus]	Ok	Ok
0F06	2	Storm layer icing: enabled if True	Ok	Ok
0F08	2	Storm layer turbulence (0 to 255 I think). Divided into steps by FSUIPC for FS2k/CFS2.	Ok	Ok
0F1C	114	Global Weather setting area: details follow. [See 0E9A for Current weather setting area] *On FS2000/CFS2 FSUIPC maps reads from this area to the Current weather area starting at 0E9A, and writes to the Current weather area to this Global weather area. Therefore you may not always read back what you last wrote. The main differences occur when FS local weather is in operation.*	Ok	Ok
0F1C	2	Upper cloud layer ceiling in metres AMSL	Ok	Ok

0F1E	2	Upper cloud layer base in metres AMSL	Ok	Ok
0F20	2	Upper cloud layer coverage, 65535 = 8 oktas, … 32768= 4 oktas … 0 = clear	Ok	Ok
0F22	2	Upper cloud layer, cloud altitude variation (metres)	Ok	Ok
0F24	2	Lower cloud layer ceiling in metres AMSL	Ok	Ok
0F26	2	Lower cloud layer base in metres AMSL	Ok	Ok
0F28	2	Lower cloud layer coverage, 65535 = 8 oktas, … 32768= 4 oktas … 0 = clear	Ok	Ok
0F2A	2	Lower cloud layer, cloud altitude variation (metres)	Ok	Ok
0F2C	2	Storm layer ceiling in metres AMSL	Ok	Ok
0F2E	2	Storm layer base in metres AMSL (if a Storm layer is present, it must be the lowest, below "Lower Cloud").	Ok	Ok
0F30	2	Storm cloud layer coverage, 65535 = 8 oktas, … 32768= 4 oktas … 0 = clear	Ok	Ok
0F32	2	Storm cloud layer, cloud altitude variation (metres)	Ok	Ok
0F34	2	Upper Temperature level, metres AMSL	Ok	Ok
0F36	2	Upper Temperature in degrees C * 256	Ok	Ok
0F38	2	Middle Temperature level, metres AMSL	Ok	Ok
0F3A	2	Middle Temperature in degrees C * 256	Ok	Ok
0F3C	2	Lower Temperature level, metres AMSL	Ok	Ok
0F3E	2	Lower Temperature in degrees C * 256	Ok	Ok
0F40	2	Surface Temperature level, metres AMSL (set this to the ground elevation of the weather reporting station)	Ok	Ok
0F42	2	Surface Temperature in degrees C * 256	Ok	Ok
0F44	2	Temperature drift, degrees C *256 (not used in FS2k/CFS2?)	Ok	Ok
0F46	2	Temperature day/night variation, degrees C *256	Ok	Ok
0F48	2	Pressure (QNH) as millibars (hectoPascals) *16.	Ok	Ok
0F4A	2	Pressure drift as millibars *16 (not used on FS2k/CFS2?)	Ok	Ok
0F4C	2	Upper wind ceiling, metres AMSL	Ok	Ok
0F4E	2	Upper wind base, metres AMSL	Ok	Ok
0F50	2	Upper wind speed, knots	Ok	Ok
0F52	2	Upper wind direction, *360/65536 gives degrees True	Ok	Ok
0F54	2	Upper wind turbulence setting, 0 none, 64, 128, 192, 224, 255 worst	Ok	Ok
0F56	2	Upper wind gusts, enabled if True.	Ok	Ok
0F58	2	Middle wind ceiling, metres AMSL	Ok	Ok
0F5A	2	Middle wind base, metres AMSL	Ok	Ok
0F5C	2	Middle wind speed, knots	Ok	Ok
0F5E	2	Middle wind direction, *360/65536 gives degrees True	Ok	Ok
0F60	2	Middle wind turbulence setting, 0 none, 64, 128, 192, 224, 255 worst	Ok	Ok
0F62	2	Middle wind gusts, enabled if True.	Ok	Ok
0F64	2	Lower wind ceiling, metres AMSL	Ok	Ok
0F66	2	Lower wind base, metres AMSL	Ok	Ok
0F68	2	Lower wind speed, knots	Ok	Ok
0F6A	2	Lower wind direction, *360/65536 gives degrees True	Ok	Ok
0F6C	2	Lower wind turbulence setting, 0 none, 64, 128, 192, 224, 255 worst	Ok	Ok
0F6E	2	Lower wind gusts, enabled if True.	Ok	Ok
0F70	2	Surface wind ceiling, metres AGL	Ok	Ok
0F72	2	Surface wind speed, knots. [See also 04D8]	Ok	Ok
0F74	2	Surface wind direction, *360/65536 gives degrees Magnetic (!). [See also 04DA]	Ok	Ok
0F76	2	Surface wind turbulence setting, 0 none, 64, 128, 192, 224, 255 worst	Ok	Ok
0F78	2	Surface wind gusts, enabled if True.	Ok	Ok
0F7A	2	Upper cloud layer type: 0=user-defined, 1=cirrus, 8=stratus, 9=cumulus	Ok	Ok

0F7C	2	Upper cloud layer icing: enabled if True	**Ok**	**Ok**
0F7E	2	Upper cloud layer turbulence (0 to 255 I think). Divided into steps by FSUIPC for FS2k/CFS2.	**Ok**	**Ok**
0F80	2	Lower cloud layer type: 0=user-defined, 1=cirrus, 8=stratus, 9=cumulus	**Ok**	**Ok**
0F82	2	Lower cloud layer icing: enabled if True	**Ok**	**Ok**
0F84	2	Lower cloud layer turbulence (0 to 255 I think). Divided into steps by FSUIPC for FS2k/CFS2.	**Ok**	**Ok**
0F86	2	Storm layer type: 10=storm. [FSUIPC allows this to be a third and lowest layer of any type, for FS2k/CFS2, so then: 0=user-defined, 1=cirrus, 8=stratus, 9=cumulus]	**Ok**	**Ok**
0F88	2	Storm layer icing: enabled if True	**Ok**	**Ok**
0F8A	2	Storm layer turbulence (0 to 255 I think). Divided into steps by FSUIPC for FS2k/CFS2.	**Ok**	**Ok**
0F8C	2	Visibility setting as 100 * statute miles	**Ok**	**Ok**
0FF0	272	Path and filename reading facility: see section in text preceding this table	**Ok**	**Ok**
115E	1	Time of day indicator, 1=Day, 2=Dusk or Dawn, 4=Night. Set according to the local time, read for lighting effects and so on in BGLs.	**Ok**	**Ok**
11A2	1	Ground scenery shadows on/off (1=On, 2=Off). [*not checked on FS98 or FS2000*]	**Ok**	**Ok**
11A4	2	Aircraft shadows on/off. Can write to this to control them (1= On, 0=Off). [*not checked on FS98 or FS2000*]	**Ok**	**Ok**
11B6	1	Aircraft reflections on/off. (2=On, 1=Off). [*not checked on FS98 or FS2000*]	**Ok**	**Ok**
11BA	2	G Force: units unknown, but /625 seems to give quite sensible values.	**Ok**	**Ok**
11BE	2	Angle of Attack. This is actually a relative value, giving in %*32767 the difference between the current AofA and the maximum angle of attack for the current aircraft. For a relative measure of AofA calculate 100-(100*#/32767), where # is this number. (*Thanks to Sergey Khantsis for this clarification*). Note: later analysis leads me to believe that this is, in fact, an Indicator angle rather than the actual AofA in any case, with the angle expressed in usual FS units (360 degrees = 65536) and with 180 degrees pointing to the 0.0 position (right and down about 35 degrees in a Boeing type AofA indicator).	**Ok**	
11C6	2	Mach speed *20480.	**Ok**	**Ok**
11D0	2	Total Air Temperature (TAT), degrees Celsius * 256	**Ok**	**Ok**
11D4	4	This is an internal pointer, not for specific use by applications, *except* that it can be used as a flag to indicate when it is possible to read or write most of the simulation variables. When this DWORD is zero FSUIPC cannot obtain correct values from SIM1.SIM (SIM1.DLL in FS2002) because either it isn't loaded or because it is busy re-calculating values by reading and processing Flight or aircraft files.	**Ok**	**Not applicable**
1244	4	Fuel: centre 2 tank level, % * 128 * 65536 [FS2k/CFS2 only]	**Ok**	**Ok**
1248	4	Fuel: centre 2 tank capacity: US Gallons [FS2k/CFS2 only]	**Ok**	**Ok**
124C	4	Fuel: centre 3 tank level, % * 128 * 65536 [FS2k/CFS2 only]	**Ok**	**Ok**
1250	4	Fuel: centre 3 tank capacity: US Gallons [FS2k/CFS2 only]	**Ok**	**Ok**
1254	4	Fuel: external 1 tank level, % * 128 * 65536 [FS2k/CFS2 only]	**Ok**	**Ok**
1258	4	Fuel: external 1 tank capacity: US Gallons [FS2k/CFS2 only]	**Ok**	**Ok**
125C	4	Fuel: external 2 tank level, % * 128 * 65536 [FS2k/CFS2 only]	**Ok**	**Ok**
1260	4	Fuel: external 2 tank capacity: US Gallons [FS2k/CFS2 only]	**Ok**	**Ok**
1274	2	Text display mode (eg for ATIS): =0 static, =1 scrolling [FS2k/CFS2 only]. (*Note that this is accessible in FS98 at 1254, but this was discovered after the FS2k extra fuel information was*	**OK**	

		mapped.)		
132C	4	NAV/GPS switch, in FS2000 & FS2002. 0=NAV, 1=GPS	**Ok**	**Ok**
1334	4	Max Gross weight, lbs * 256. This is the maximum aircraft weight including payload and fuel.	**No**	**Ok**
13FC	4	Count of Payload Stations (FS2004 only)	**No**	**Ok**
1400	48 x n	A set of Payload Station data, 48 bytes for each payload station (the count is in 13FC above). [FS2004 only]. Each 48 byte entry contains: double weight (lbs) double lateral distance from datum (feet) double vertical distance from datum (feet) double longitudinal distance from datum (feet) char Name[16]; // 16 char name, including 0 at end There's room for up to 61 such stations here. If there are more you can't access them this way. These loadings can be changed, and this does have some effect, but such changes are not being promulgated to the overall weights (offsets 30C0, 30C8, 3BFC) nor balance (2EF8), and it looks like they have to refreshed, as FS overrides them from time to time. It has also been reported that FS can crash if a lot of changes are made here, so care and full testing is needed.	**No**	**Ok**
1F80	*	Write-only area for a TCAS_DATA structure, used to add entries to the TCAS data tables—see offset, below, and the section on TCAS earlier in this document. * The length of data written here is determined by the size of the TCAS_DATA structure, currently 40 bytes (but read this from offset F000).	**Ok**	**Ok**
2000	8	Turbine Engine 1 N1 value (%) as a double (FLOAT64). This is for jets and turboprops—it has no meaning on reciprocating prop aircraft.	**Ok**	**Ok**
2008	8	Turbine Engine 1 N2 value (%) as a double (FLOAT64). This is for jets and turboprops—it has no meaning on reciprocating prop aircraft.	**Ok**	**Ok**
2010	8	Turbine Engine 1 corrected N1 value (%) as a double (FLOAT64). This is for jets and turboprops—it has no meaning on reciprocating prop aircraft.	**Ok**	**Ok**
2018	8	Turbine Engine 1 corrected N2 value (%) as a double (FLOAT64). This is for jets and turboprops—it has no meaning on reciprocating prop aircraft.	**Ok**	**Ok**
2020	8	Turbine Engine 1 corrected fuel flow (pounds per hour) as a double (FLOAT64). This is for jets and turboprops—it has no meaning on reciprocating prop aircraft.	**Ok**	**Ok**
2028	8	Turbine Engine 1 max torque fraction (range 0.0–1.0) as a double (FLOAT64). (*Only tested on turboprops*).	**Ok**	**Ok**
2030	8	Turbine Engine 1 EPR as a double (FLOAT64). This is for jets and turboprops.	**Ok**	**Ok**
2038	8	Turbine Engine 1 ITT (interstage turbine temperature) in degrees Rankine, as a double (FLOAT64). This is for jets and turboprops.	**Ok**	**Ok**
2048	4	Turbine Engine 1 Afterburner switch (1 = on, 0 = off)	**Ok**	**Ok**
204C	8	Turbine Engine 1 jet thrust, in pounds, as a double (FLOAT64). This is the jet thrust. See 2410 for propeller thrust (turboprops have both).	**Ok**	**Ok**
205C	4	Turbine Engine 1, number of fuel tanks available	**Ok**	**Ok**
2060	8	Turbine Engine 1 fuel flow (pounds per hour) as a double (FLOAT64). This is for jets and turboprops.	**Ok**	**Ok**
206C	8	Turbine Engine 1 bleed air pressure (pounds per square inch) as	**Ok**	**Ok**

		a double (FLOAT64). This is for jets and turboprops.		
207C	8	Turbine Engine 1 reverser fraction, a double (FLOAT64), in the range 0.0–1.0, providing the reverse as a proportion of the maximum reverse throttle position.	**Ok**	**Ok**
2100	8	Turbine Engine 2 N1 value (%) as a double (FLOAT64). This is for jets and turboprops—it has no meaning on reciprocating prop aircraft.	**Ok**	**Ok**
2108	8	Turbine Engine 2 N2 value (%) as a double (FLOAT64). This is for jets and turboprops—it has no meaning on reciprocating prop aircraft.	**Ok**	**Ok**
2110	8	Turbine Engine 2 corrected N1 value (%) as a double (FLOAT64). This is for jets and turboprops—it has no meaning on reciprocating prop aircraft.	**Ok**	**Ok**
2118	8	Turbine Engine 2 corrected N2 value (%) as a double (FLOAT64). This is for jets and turboprops—it has no meaning on reciprocating prop aircraft.	**Ok**	**Ok**
2120	8	Turbine Engine 2 corrected fuel flow (pounds per hour) as a double (FLOAT64). This is for jets and turboprops—it has no meaning on reciprocating prop aircraft.	**Ok**	**Ok**
2128	8	Turbine Engine 2 max torque fraction (range 0.0–1.0) as a double (FLOAT64). (*Only tested on turboprops*).	**Ok**	**Ok**
2130	8	Turbine Engine 2 EPR as a double (FLOAT64). This is for jets and turboprops.	**Ok**	**Ok**
2138	8	Turbine Engine 2 ITT (interstage turbine temperature) in degrees Rankine, as a double (FLOAT64). This is for jets and turboprops.	**Ok**	**Ok**
2148	4	Turbine Engine 2 Afterburner switch (1 = on, 0 = off)	**Ok**	**Ok**
214C	8	Turbine Engine 2 jet thrust, in pounds, as a double (FLOAT64). This is the jet thrust. See 2510 for propeller thrust (turboprops have both).	**Ok**	**Ok**
215C	4	Turbine Engine 2, number of fuel tanks available	**Ok**	**Ok**
2160	8	Turbine Engine 2 fuel flow (pounds per hour) as a double (FLOAT64). This is for jets and turboprops.	**Ok**	**Ok**
216C	8	Turbine Engine 2 bleed air pressure (pounds per square inch) as a double (FLOAT64). This is for jets and turboprops.	**Ok**	**Ok**
217C	8	Turbine Engine 2 reverser fraction, a double (FLOAT64), in the range 0.0–1.0, providing the reverse as a proportion of the maximum reverse throttle position.	**Ok**	**Ok**
2200	8	Turbine Engine 3 N1 value (%) as a double (FLOAT64). This is for jets and turboprops—it has no meaning on reciprocating prop aircraft.	**Ok**	**Ok**
2208	8	Turbine Engine 3 N2 value (%) as a double (FLOAT64). This is for jets and turboprops—it has no meaning on reciprocating prop aircraft.	**Ok**	**Ok**
2210	8	Turbine Engine 3 corrected N1 value (%) as a double (FLOAT64). This is for jets and turboprops—it has no meaning on reciprocating prop aircraft.	**Ok**	**Ok**
2218	8	Turbine Engine 3 corrected N2 value (%) as a double (FLOAT64). This is for jets and turboprops—it has no meaning on reciprocating prop aircraft.	**Ok**	**Ok**
2220	8	Turbine Engine 3 corrected fuel flow (pounds per hour) as a double (FLOAT64). This is for jets and turboprops—it has no meaning on reciprocating prop aircraft.	**Ok**	**Ok**
2228	8	Turbine Engine 3 max torque fraction (range 0.0–1.0) as a double (FLOAT64). (*Only tested on turboprops*).	**Ok**	**Ok**
2230	8	Turbine Engine 3 EPR as a double (FLOAT64). This is for jets and turboprops.	**Ok**	**Ok**
2238	8	Turbine Engine 3 ITT (interstage turbine temperature) in degrees Rankine, as a double (FLOAT64). This is for jets and	**Ok**	**Ok**

		turboprops.		
2248	4	Turbine Engine 3 Afterburner switch (1 = on, 0 = off)	**Ok**	**Ok**
224C	8	Turbine Engine 3 jet thrust, in pounds, as a double (FLOAT64). This is the jet thrust. See 2610 for propeller thrust (turboprops have both).	**Ok**	**Ok**
225C	4	Turbine Engine 3, number of fuel tanks available	**Ok**	**Ok**
2260	8	Turbine Engine 3 fuel flow (pounds per hour) as a double (FLOAT64). This is for jets and turboprops.	**Ok**	**Ok**
226C	8	Turbine Engine 3 bleed air pressure (pounds per square inch) as a double (FLOAT64). This is for jets and turboprops.	**Ok**	**Ok**
227C	8	Turbine Engine 3 reverser fraction, a double (FLOAT64), in the range 0.0–1.0, providing the reverse as a proportion of the maximum reverse throttle position.	**Ok**	**Ok**
2300	8	Turbine Engine 4 N1 value (%) as a double (FLOAT64). This is for jets and turboprops—it has no meaning on reciprocating prop aircraft.	**Ok**	**Ok**
2308	8	Turbine Engine 4 N2 value (%) as a double (FLOAT64). This is for jets and turboprops—it has no meaning on reciprocating prop aircraft.	**Ok**	**Ok**
2310	8	Turbine Engine 4 corrected N1 value (%) as a double (FLOAT64). This is for jets and turboprops—it has no meaning on reciprocating prop aircraft.	**Ok**	**Ok**
2318	8	Turbine Engine 4 corrected N2 value (%) as a double (FLOAT64). This is for jets and turboprops—it has no meaning on reciprocating prop aircraft.	**Ok**	**Ok**
2320	8	Turbine Engine 4 corrected fuel flow (pounds per hour) as a double (FLOAT64). This is for jets and turboprops—it has no meaning on reciprocating prop aircraft.	**Ok**	**Ok**
2328	8	Turbine Engine 4 max torque fraction (range 0.0–1.0) as a double (FLOAT64). (*Only tested on turboprops*).	**Ok**	**Ok**
2330	8	Turbine Engine 4 EPR as a double (FLOAT64). This is for jets and turboprops.	**Ok**	**Ok**
2338	8	Turbine Engine 4 ITT (interstage turbine temperature) in degrees Rankine, as a double (FLOAT64). This is for jets and turboprops.	**Ok**	**Ok**
2348	4	Turbine Engine 4 Afterburner switch (1 = on, 0 = off)	**Ok**	**Ok**
234C	8	Turbine Engine 4 jet thrust, in pounds, as a double (FLOAT64). This is the jet thrust. See 2710 for propeller thrust (turboprops have both).	**Ok**	**Ok**
235C	4	Turbine Engine 4, number of fuel tanks available	**Ok**	**Ok**
2360	8	Turbine Engine 4 fuel flow (pounds per hour) as a double (FLOAT64). This is for jets and turboprops.	**Ok**	**Ok**
236C	8	Turbine Engine 4 bleed air pressure (pounds per square inch) as a double (FLOAT64). This is for jets and turboprops.	**Ok**	**Ok**
237C	8	Turbine Engine 4 reverser fraction, a double (FLOAT64), in the range 0.0–1.0, providing the reverse as a proportion of the maximum reverse throttle position.	**Ok**	**Ok**
2400	8	Propeller 1 RPM as a double (FLOAT64). This value is for props and turboprops and is negative for counter-rotating propellers.	**Ok**	**Ok**
2408	8	Propeller 1 RPM as a fraction of the maximum RPM. (double)	**Ok**	**Ok**
2410	8	Propeller 1 thrust in pounds, as a double (FLOAT64). This is for props and turboprops.	**Ok**	**Ok**
2418	8	Propeller 1 Beta blade angle in radians, as a double (FLOAT64). This is for props and turboprops.	**Ok**	**Ok**
2500	8	Propeller 2 RPM as a double (FLOAT64). This value is for props and turboprops and is negative for counter-rotating propellers.	**Ok**	**Ok**
2508	8	Propeller 2 RPM as a fraction of the maximum RPM. (double)	**Ok**	**Ok**

2510	8	Propeller 2 thrust in pounds, as a double (FLOAT64). This is for props and turboprops.	**Ok**	**Ok**
2518	8	Propeller 2 Beta blade angle in radians, as a double (FLOAT64). This is for props and turboprops.	**Ok**	**Ok**
2600	8	Propeller 3 RPM as a double (FLOAT64). This value is for props and turboprops and is negative for counter-rotating propellers.	**Ok**	**Ok**
2608	8	Propeller 3 RPM as a fraction of the maximum RPM. (double)	**Ok**	**Ok**
2610	8	Propeller 3 thrust in pounds, as a double (FLOAT64). This is for props and turboprops.	**Ok**	**Ok**
2618	8	Propeller 3 Beta blade angle in radians, as a double (FLOAT64). This is for props and turboprops.	**Ok**	**Ok**
2700	8	Propeller 4 RPM as a double (FLOAT64). This value is for props and turboprops and is negative for counter-rotating propellers.	**Ok**	**Ok**
2708	8	Propeller 4 RPM as a fraction of the maximum RPM. (double)	**Ok**	**Ok**
2710	8	Propeller 4 thrust in pounds, as a double (FLOAT64). This is for props and turboprops.	**Ok**	**Ok**
2718	8	Propeller 4 Beta blade angle in radians, as a double (FLOAT64). This is for props and turboprops.	**Ok**	**Ok**
281C	4	Master battery switch (1=On, 0=Off)	**Ok**	**Ok**
28C0	8	Ambient air density, in slugs per cubic foot, double floating point. (FS2002+)		**Ok**
28C8	8	Ambient air pressure, in lbs per square foot, double floating point. (FS2002+)		**Ok**
28D0	8	Static air temperature, in degrees Fahrenheit, double floating point. (FS2002+)		**Ok**
28D8	8	Static air temperature, in degrees Rankine, double floating point. (FS2002+)		**Ok**
28E0	8	"Theta", or standard temperature ratio (i.e ambient air temperature divided by standard ISO air temperature), double floating point. (FS2002+)		**Ok**
28E8	8	"Delta", or standard pressure ratio (ambient pressure divided by the ISO standard pressure, double floating point. (FS2002+)		**Ok**
28F0	8	"Sigma", or standard density ratio (ambient density divided by the ISO standard density, double floating point. (FS2002+)		**Ok**
2900	12	A.I. traffic control. Write all 3 32-bit values (i.e. 12 bytes) together to send an FS control to a specific AI aircraft. The values needed are: Bytes 0–3: Aircraft Id (from the TCAS table) Bytes 4–7: The FS Control (see published lists) Bytes 8–11: A parameter for the control, if needed Note that most of the many hundreds of FS controls will have no noticeable affect on the AI aircraft. Experimentation is needed. If folks find out what does what, please let me know and I'll try to publish a collated guide as an appendix later. FSUIPC offers one special extra control. Just set the aircraft ID and an FS control of 0xFFFF (65535) and the aircraft will be deleted. Note that you can write these values in separate FSUIPC Writes, but if you do the ID must be last, as it is only when this is written that the control is activated.	**Ok**	**Ok**
290C	4	Number of Hot Joystick Button slots available for Application Programs to use. Currently this is fixed at 56, representing the 56 DWORDs available in the following offsets:	**Ok**	**Ok**
2910	224	56 DWORDs containing zero (when free for use), or a Hot Joystick Button specification as detailed earlier in this document. See also 32FF below.	**Ok**	**Ok**

29F0	4	This DWORD provides a facility to set, clear or toggle any of the virtual buttons at offset 3340 without needing to read anything first. To do this, write to offset 29F0 a 32-bit value (4 bytes) made up as follows: Byte 0: Button Number on Joystick (0 - 31) Byte 1: Virtual Joystick Number (64 - 72) Byte 2: Action: 0 = Toggle 1 = Set (Press/On) 2 = Clear (Release/Off). Byte 3: 0 (Reserved)		
2A48	8	Folding wing (for reading), left percent, as double float. [FS2002+]	**Ok**	**Ok**
2A50	8	Folding wing (for reading), right percent, as double float. [FS2002+]	**Ok**	**Ok**
2A70	8	Canopy open, as double float. [FS2002+]	**Ok**	**Ok**
2A78	8	Water left rudder extended (double float)		**Ok**
2A80	8	Water right rudder extended (double float)		**Ok**
2A88	4	Water rudder handle position (BOOLean, I think?)		**Ok**
2AAC	4	NAV1 course deviation needle (CDI), 32-bit float value, –127.0 left to +127.0 right	**Ok**	**Ok**
2AB0	4	NAV1 glideslope needle (CDI), 32-bit float value, –127.0 up to +127.0 down	**Ok**	**Ok**
2AB4	4	NAV2 course deviation needle (CDI), 32-bit float value, –127.0 left to +127.0 right	**Ok**	**Ok**
2AB8	4	NAV2 glideslope needle (CDI), 32-bit float value, –127.0 up to +127.0 down	**Ok**	**Ok**
2B00	8	Gyro compass heading (magnetic), including any drift. 64-bit floating point.	**Ok**	**Ok**
2DC8	8	For FS2004 only, this is the wind at the aircraft in the lateral (X) axis—relative to the aircraft orientation, in feet per second, as a 64-bit double.	**No**	**Ok**
2DD0	8	For FS2004 only, this is the wind at the aircraft in the vertical (Y) axis—relative to the aircraft orientation, in feet per second, as a 64-bit double.	**No**	**Ok**
2DD8	8	For FS2004 only, this is the wind at the aircraft in the longitudinal (Z) axis—relative to the aircraft orientation, in feet per second, as a 64-bit double.	**No**	**Ok**
2DE0	8	For FS2004 only, Wind direction at the aircraft, in degrees True, as a 64-bit double floating point – for writing, not reading. See 3490 for reading. This can be written to directly affect the wind direction at the aircraft. This value is set *before* FSUIPC performs any smoothing or limiting actions, and effectively become the new target value. FSUIPC sustains this as a target for a maximum of 14 seconds, with the next write to the same location restarting this timeout. After the timeout has been allowed to expire the intended FS value will take over, with smoothing and so on if enabled. Note that wind direction set in this fashion is *not* reflected in any weather data supplied by the weather system in FS nor FSUIPC. It is acting locally to the aircraft and can be monitored by Shift+Z or the ambient weather read-outs in FSUIPC.	**No**	**Ok, but see Note ****
2DE8	8	For FS2004 only, Wind speed at the aircraft, in knots, as a 64-bit double floating point – for writing, not reading. See 3488 for reading. This can be written to directly affect the wind speed at the	**No**	**Ok**

		aircraft. This value is set *before* FSUIPC performs any smoothing or limiting actions, and effectively become the new target value. FSUIPC sustains this as a target for a maximum of 14 seconds, with the next write to the same location restarting this timeout. After the timeout has been allowed to expire the intended FS value will take over, with smoothing and so on if enabled.		
2DF0	8	For FS2004 only, Visibility at the aircraft, in metres, as a 64-bit double floating point – for writing, not reading. This can be written to directly affect the visibility at the aircraft. This value is set *before* FSUIPC performs any smoothing or limiting actions, and effectively become the new target value. FSUIPC sustains this as a target for a maximum of 14 seconds, with the next write to the same location restarting this timeout. After the timeout has been allowed to expire the intended FS value will take over, with smoothing and so on if enabled. Note that visibility set in this fashion is *not* reflected in any weather data supplied by the weather system in FS nor FSUIPC. It is acting locally to the aircraft and can be monitored by Shift+Z or the ambient weather read-outs in FSUIPC.	**No**	**Ok**
2E80	4	Master avionics switch (0=Off, 1=On)	**Ok**	**Ok**
2E88	4	Panel auto-feather arm switch (0=Off, 1=On)	**Ok**	**Ok**
2E98	8	Elevator deflection, in radians, as a double (FLOAT64). Up positive, down negative.	**Ok**	**Ok**
2EA0	8	Elevator trim deflection, in radians, as a double (FLOAT64). Up positive, down negative.	**Ok**	**Ok**
2EA8	8	Aileron deflection, in radians, as a double (FLOAT64). Right turn positive, left turn negative.	**Ok**	**Ok**
2EB0	8	Aileron trim deflection, in radians, as a double (FLOAT64). Right turn positive, left turn negative.	**Ok**	
2EB8	8	Rudder deflection, in radians, as a double (FLOAT64).	**Ok**	**Ok**
2EC0	8	Rudder trim deflection, in radians, as a double (FLOAT64).	**Ok**	
2EC8	4	Prop sync active (1=Active, 0=Inactive)	**Ok**	**Ok**
2ED0	8	Incidence "alpha", in radians, as a double (FLOAT64). This is the aircraft *body* angle of attack (AoA) not the *wing* AoA.	**Ok**	**Ok**
2ED8	8	Incidence "beta", in radians, as a double (FLOAT64). This is the side slip angle.	**Ok**	**Ok**
2EE0	4	Flight Director Active, control and indicator. 1=active, 0=inactive. [FS2000–FS2004 only]	**Ok**	**Ok**
2EE8	8	Flight director pitch value, in degrees. Double floating point format, only when FD is active. [FS2000–FS2004 only]	**Ok**	**Ok**
2EF0	8	Flight director bank value, in degrees. Double floating point format, right is negative, left positive. [FS2000–FS2004 only]	**Ok**	**Ok**
2EF8	8	CG percent, as a double (FLOAT64). This is the position of the actual CoG as a fraction (%/100) of MAC	**Ok**	**Ok**
2F70	8	Attitude indicator pitch value, in degrees. Double floating point format. This is the ATTITUDE_INDICATOR _PITCH_DEGREES variable previously listed as specific to FS2000. [FS2000/FS2002 only]	**Ok**	**Ok**
2F78	8	Attitude indicator bank value, in degrees. Double floating point format. This is the ATTITUDE_INDICATOR_BANK_DEGREES variable previously listed as specific to FS2000. [FS2000/FS2002 only]	**Ok**	**Ok**
2F80	1	PANEL AUTOBRAKE SWITCH Read to check setting, write to change it. 0=RTO, 1=Off, 2=brake1, 3=brake2, 4=brake3, 5=max	**Ok**	**Ok**
2F88	8	HSI CDI needle position, –127.0 to +127.0 double floating point. Full range represents –10 to +10 degrees for a VOR, –2.5 to +2.5 degrees for a LOC	**Ok**	**Ok**
2F90	8	HSI GSI needle position, –119.0 to +119.0 double floating point. Full range represents –0.7 to +0.7 degrees	**Ok**	**Ok**
2F98	8	HSI speed, as a double floating point. I think it should be in metres/sec,	**Ok?**	**Ok?**

		but it doesn't look right—feedback please!		
2FA0	8	HSI distance, as a double floating point. In metres.	**Ok**	**Ok**
2FA8	2	HSI bearing. In degrees? Doesn't seem to work. Feedback?		
2FAA	1	HSI CDI valid flag. Doesn't appear to work?		
2FAB	1	HSI GSI valid flag.	**Ok**	**Ok**
2FAC	1	HSI bearing valid flag. (Not seen this set yet – see 2FA8)		
2FAD	1	HSI To/From flag: 0=off, 1=To, 2=From	**Ok**	**Ok**
2FAE	1	HSI 'has localiser' flag	**Ok**	**Ok**
2FB0	6	HSI ident string	**Ok**	**Ok**
2FE0	32	Modules Menu, application item write area (see earlier in this document)	**Ok**	**Ok**
3000	6	VOR1 IDENTITY (string supplied: 6 bytes including zero terminator)	**Ok**	**Ok**
3006	25	VOR1 name (string supplied: 25 bytes including zero terminator)	**Ok**	**Ok**
301F	6	VOR2 IDENTITY (string supplied: 6 bytes including zero terminator)	**Ok**	**Ok**
3025	25	VOR2 name (string supplied: 25 bytes needed including zero terminator)	**Ok**	**Ok**
303E	6	ADF1 IDENTITY (string supplied: 6 bytes including zero terminator)	**Ok**	**Ok**
3044	25	ADF1 name (string supplied: 25 bytes including zero terminator)	**Ok**	**Ok**
3060	8	X (lateral, or left/right) acceleration in ft/sec/sec relative to the body axes (*see Note at end of table*). This is in double floating point format (FLOAT64). [FS2000 and later]	**Ok**	
3068	8	Y (vertical, or up/down) acceleration in ft/sec/sec relative to the body axes (*see Note at end of table*). This is in double floating point format (FLOAT64). [FS2000 and later]	**Ok**	
3070	8	Z (longitudinal, or forward/backward) acceleration in ft/sec/sec relative to the body axes (*see Note at end of table*). This is in double floating point format (FLOAT64). [FS2000 and later]	**Ok**	**Ok**
3078	8	Pitch acceleration in radians/sec/sec relative to the body axes (*see Note at end of table*). This is in double floating point format (FLOAT64). [FS2000 and later]	**Ok**	
3080	8	Roll acceleration in radians/sec/sec relative to the body axes (*see Note at end of table*). This is in double floating point format (FLOAT64). [FS2000 and later]	**Ok**	
3088	8	Yaw acceleration in radians/sec/sec relative to the body axes (*see Note at end of table*). This is in double floating point format (FLOAT64). [FS2000 and later]	**Ok**	
3090	8	Z (longitudinal, or forward/backward) GS-velocity in ft/sec relative to the body axes (*see Note at end of table*). This is in double floating point format (FLOAT64). [FS2000 and later]	**Ok**	**Ok**
3098	8	X (lateral, or left/right) GS-velocity in ft/sec relative to the body axes (*see Note at end of table*). This is in double floating point format (FLOAT64). [FS2000 and later]	**Ok**	
30A0	8	Y (vertical, or up/down) GS-velocity in ft/sec relative to the body axes (*see Note at end of table*). This is in double floating point format (FLOAT64). [FS2000 and later]	**Ok**	
30A8	8	Pitch velocity in rads/sec relative to the body axes (*see Note at end of table*). This is in double floating point format (FLOAT64). [FS2000 and later]	**Ok**	
30B0	8	Roll velocity in rads/sec relative to the body axes (*see Note at end of table*). This is in double floating point format (FLOAT64). [FS2000 and later]	**Ok**	
30B8	8	Yaw velocity in rads/sec relative to the body axes (*see Note at end of table*). This is in double floating point format (FLOAT64). [FS2000 and later]	**Ok**	
30C0	8	Current loaded weight in lbs. This is in double floating point format (FLOAT64). [FS2000 and later]	**Ok**	
30C8	8	Plane's current mass, in slugs (1 slug = 1lb*G = 32.174049 lbs) mass. This is in double floating point format (FLOAT64).	**Ok**	**Ok**

		The current mass = current loaded weight (as in 30C0) * G, where G is 32.174049.		
30D0	8	Vertical acceleration in G's. This is in double floating point format (FLOAT64). [FS2k only]	**No**	**No**
30D8	8	Dynamic pressure (lbs/sqft). [FS2k/CFS2/FS2002 only]	**Ok**	**Ok**
30E0	2	[FS2002/4 only]: Trailing edge left inboard flap extension as a percentage of its maximum, with 16383 = 100%	**Ok**	**Ok**
30E2	2	[FS2002/4 only]: Trailing edge left outboard flap extension as a percentage of its maximum, with 16383 = 100%	**Ok**	**Ok**
30E4	2	[FS2002/4 only]: Trailing edge right inboard flap extension as a percentage of its maximum, with 16383 = 100%	**Ok**	**Ok**
30E6	2	[FS2002/4 only]: Trailing edge right outboard flap extension as a percentage of its maximum, with 16383 = 100%	**Ok**	**Ok**
30E8	2	[FS2002/4 only]: Leading edge left inboard flap extension as a percentage of its maximum, with 16383 = 100%	**Ok**	**Ok**
30EA	2	[FS2002/4 only]: Leading edge left outboard flap extension as a percentage of its maximum, with 16383 = 100%	**Ok**	**Ok**
30EC	2	[FS2002/4 only]: Leading edge right inboard flap extension as a percentage of its maximum, with 16383 = 100%	**Ok**	**Ok**
30EE	2	[FS2002/4 only]: Leading edge right outboard flap extension as a percentage of its maximum, with 16383 = 100%	**Ok**	**Ok**
30F0	2	[FS2002/4 only]: Trailing edge left inboard flap extension in degrees * 256.	**Ok**	**Ok**
30F2	2	[FS2002/4 only]: Trailing edge left outboard flap extension in degrees * 256.	**Ok**	**Ok**
30F4	2	[FS2002/4 only]: Trailing edge right inboard flap extension in degrees * 256.	**Ok**	**Ok**
30F6	2	[FS2002/4 only]: Trailing edge right outboard flap extension in degrees * 256.	**Ok**	**Ok**
30F8	2	[FS2002/4 only]: Leading edge left inboard flap extension in degrees * 256.	**Ok**	**Ok**
30FA	2	[FS2002/4 only]: Leading edge left outboard flap extension in degrees * 256.	**Ok**	**Ok**
30FC	2	[FS2002/4 only]: Leading edge right inboard flap extension in degrees * 256.	**Ok**	**Ok**
30FE	2	[FS2002/4 only]: Leading edge right outboard flap extension in degrees * 256.	**Ok**	**Ok**
3100	1	Engine primer (just write a non-zero byte to operate the primer. This is a one-shot and reading it is meaningless) [FS2000+]	**Ok**	
3101	1	Alternator (1 = on, 0 = off), read for state, write to control [FS2000+]	**Ok**	
3102	1	Battery (1 = on, 0 = off), read for state, write to control [FS2000+]	**Ok**	**Ok**
3103	1	Avionics (1 = on, 0 = off), read for state, write to control [FS2000+]	**Ok**	**Ok**
3104	1	Fuel pump (1 = on, 0 = off), read for state, write to control [FS2000+]. For separate switches for separate fuel pumps see offset 3125.	**Ok**	
3105	1	VOR1 morse ID sound (1 = on, 0 = off), read for state, write to control [FS2000+]	**Ok**	
3106	1	VOR2 morse ID sound (1 = on, 0 = off), read for state, write to control [FS2000+]	**Ok**	
3107	1	ADF1 morse ID sound (1 = on, 0 = off), read for state, write to control [FS2000+]	**Ok**	
3108	1	Write 1 here to disable FSUIPC's "AutoTune ADF1" facility, if this has been enabled by the user in FSUIPC.INI.	**Ok**	**Ok**
3109	1	This is a bit-oriented control flag byte. These bits are allocated so far:	**Ok**	**Ok**

		2^0 (1) = 1 to disable AxisCalibration even if enabled in FSUIPC.INI. Note that this "AxisCalibration" is the one specifically concerned with direct offset values—see the Advanced User's guide for the description of the INI parameter for more details. 2^1 (2) = 1 to allow the older (FS98-compatible) axis controls to remain connected even when the main axis controls are disconnected via bits in 310A and 310B below. These are AILERON_SET, ELEVATOR_SET, ELEVATOR_TRIM_SET, RUDDER_SET, THROTTLE_SET and the four THROTTLEn_SET controls. Allowing these through will let autopilot of FBW programs control the relevant values without writing direct to the appropriate offsets, but take care also that the THROTTLEn_SET controls aren't being calibrated in the user's 4-throttle option (page 3 in FSUIPC options). 2^7(128) is reserved for external applications to use as they wish. In order to protect the user from a broken or crashed application, the 2^1 flag is cleared 10 seconds after it has been set, so applications will need to repeat the setting every few seconds.		
310A	1	Controls the joystick connection to the main flight controls. Normally all zero, set the following bits to actually disconnect the specific joystick axes (from least significant bit = 0): 0 Elevator 1 Aileron 2 Rudder 3 Throttles (all). 4 *See below (throttle sync control)* 5 Elevator trim 6 Throttle #1 7 Throttle #2 (see next byte for others) This feature is intended for use in protecting autopilot flight from interference from axis flutter. In order to protect the user from a broken or crashed application, all the flags are cleared 10 seconds after they have been set, so applications will need to repeat the setting every few seconds. If the user option is set to automatically disconnect the trim axis in FS A/P vertical modes, the disconnection of Elevator inputs via bit 0 above also disconnects Trim even if bit 5 is not also set. This allows existing A/P or fly-by-wire applications to work with those user implementations using a trim axis. Additionally, bit 2^4 is available to switch "throttle sync" on. In this mode all throttles are driven from the main throttle or throttle 1 inputs, and other throttle inputs are discarded. (The same option can also be used from an optional Hot Key). See also offset 3109 above, and also offsets 3328–3339, which provide the live axis values, post calibration. These would have been applied to FS if not prevented by the flags above. Applications can use these facilities to provide a responsive "fly-by-wire" control. Note: additional axes are covered in 310B and 341B.	**Ok**	**Ok**
310B	1	Controls the joystick connection to the slewing controls, and the	**Ok**	**Ok**

		other two separate throttle controls. Normally all zero, set the following bits to actually disconnect the specific axes (from least significant bit = 0): 0 Slew Ahead 1 Slew Side 2 Slew Heading 3 Slew Altitude 4 Slew Bank 5 Slew Pitch 6 Throttle #3 (see previous byte for #1, #2) 7 Throttle #4 In order to protect the user from a broken or crashed application, all the flags are cleared 10 seconds after they have been set, so applications will need to repeat the setting every few seconds. See also offset 3109 above.		
310C	4	*Reserved*		
3110	8	Operates a facility to send any 'controls' to Flight simulator. This works with *all* versions of FS & CFS. Write all 8 bytes for controls which use a value (axes and all _SET controls), but just 4 will do for 'button' types. This is really two 32-bit integers. The first contains the Control number (normally 65536 upwards), as seen in my FS Controls lists. The second integer is used for the parameter, such as the scaled axis value, where this is appropriate. Always write all 8 bytes in one IPC block if a parameter is used, as FSUIPC will fire the control when you write to 3110. Since version 3.40, FSUIPC-added controls (other than the offset ones) can be used via these offsets too. See the Advanced User's Guide for a current list.	**Ok**	**Ok**
3118	2	COM2 frequency (FS2002+ only), 4 digits in BCD format. A frequency of 123.45 is represented by 0x2345. The leading 1 is assumed.	**Ok**	**Ok**
311A	2	COM1 standby frequency (FS2002+ only), 4 digits in BCD format. A frequency of 123.45 is represented by 0x2345. The leading 1 is assumed.	**Ok**	**Ok**
311C	2	COM2 standby frequency (FS2002+ only), 4 digits in BCD format. A frequency of 123.45 is represented by 0x2345. The leading 1 is assumed.	**Ok**	**Ok**
311E	2	NAV1 standby frequency (FS2002+ only), 4 digits in BCD format. A frequency of 113.45 is represented by 0x1345. The leading 1 is assumed.	**Ok**	**Ok**
3120	2	NAV2 standby frequency (FS2002+ only), 4 digits in BCD format. A frequency of 113.45 is represented by 0x1345. The leading 1 is assumed.	**Ok**	**Ok**
3122	1	Radio audio switches (FS2002+ only). Read/write bit settings as follows: 2^7 COM1 transmit 2^6 COM2 transmit 2^5 COM receive both 2^4 NAV1 sound 2^3 NAV2 sound 2^2 Marker sound 2^1 DME sound 2^0 ADF1 sound For ADF2 sound, on FS2004, see offset 02FB.	**Ok**	**Ok**
3123	1	Radio Use/Standby swap toggles (FS2002+ only), Write bits to	**Ok**	**Ok**

		operate toggles. Don't bother to read it, there's no meaning to anything read. 2^3 COM1 swap 2^2 COM2 swap 2^1 NAV1 swap 2^0 NAV2 swap		
3124	1	FS2002 only: "electric always available" flag: set if 1, clear if 0. Can be controlled by writing also.	**Ok**	**No**
3125	1	FS2000/FS2002 only: separate switches for up to 4 Fuel Pumps (one for each engine). Bit 2^0=Pump1, 2^1=Pump2, 2^2=Pump3, 2^4=Pump4. (*see also offset 3104*)	**Ok**	
3126	1	Set view direction (write only, current view not detected). 0 = FORWARD 1–7 = FORWARD RIGHT and 45 degree views, clockwise 8 = DOWN 9 = UP 10-17 = FORWARD UP then 45 degree UP views, clockwise all other values = RESET	**Ok**	
3127	9	FSUIPC weather option control area: see text section earlier in this document.	**Ok**	**Ok**
3130	12	ATC flight number string for currently loaded user aircraft, as declared in the AIRCRAFT.CFG file. This is limited to a maximum of 12 characters, including a zero terminator. [FS2002+ only]	**Ok**	**Ok**
313C	12	ATC identifier (tail number) string for currently loaded user aircraft, as declared in the AIRCRAFT.CFG file. This is limited to a maximum of 12 characters, including a zero terminator. [FS2002+ only]	**Ok**	**Ok**
3148	24	ATC airline name string for currently loaded user aircraft, as declared in the AIRCRAFT.CFG file. This is limited to a maximum of 24 characters, including a zero terminator. [FS2002+ only]	**Ok**	**Ok**
3160	24	ATC aircraft type string for currently loaded user aircraft, as declared in the AIRCRAFT.CFG file. This is limited to a maximum of 24 characters, including a zero terminator. [FS2002+ only]	**Ok**	**Ok**
3178	8	Z (longitudinal, or forward/backward) TAS-velocity in ft/sec relative to the body axes. This is in double floating point format (FLOAT64). [[FS2002+ only]]	**Ok**	
3180	8	X (lateral, or left/right) TAS-velocity in ft/sec relative to the body axes (*see Note at end of table*). This is in double floating point format (FLOAT64). [[FS2002+ only]]	**Ok**	
3188	8	Y (vertical, or up/down) TAS-velocity in ft/sec relative to the body axes (*see Note at end of table*). This is in double floating point format (FLOAT64). [[FS2002+ only]]	**Ok**	
3190	8	Z (longitudinal, or forward/backward) GS-velocity in ft/sec relative to world axes (*see Note at end of table*). This is in double floating point format (FLOAT64). [FS2000+]	**Ok**	
3198	8	X (lateral, or left/right) GS-velocity in ft/sec relative to world axes (*see Note at end of table*). This is in double floating point format (FLOAT64). [FS2000+] N.B. The sign may be reversed in FS2002?	**Ok?**	
31A0	8	Y (vertical, or up/down) GS-velocity in ft/sec relative to world axes (*see Note at end of table*). This is in double floating point format (FLOAT64). [FS2000+]	**Ok**	
31A8	8	Pitch velocity in rads/sec relative to world axes (*see Note at end of table*). This is in double floating point format (FLOAT64).	**Ok**	

		[FS2000+]		
31B0	8	Roll velocity in rads/sec relative to world axes (*see Note at end of table*). This is in double floating point format (FLOAT64). [FS2000+] N.B. In FS2002 the sign may be reversed, and the units may be 16x	**Ok?**	
31B8	8	Yaw velocity in rads/sec relative to world axes (*see Note at end of table*). This is in double floating point format (FLOAT64). [FS2000+] N.B. In FS2002 the sign may be reversed, and the units may be 16x	**Ok?**	
31C0	8	X (lateral, or left/right) acceleration in ft/sec/sec relative to the world axes (*see Note at end of table*). This is in double floating point format (FLOAT64). [FS2002+]	**Ok**	
31C8	8	Y (vertical, or up/down) acceleration in ft/sec/sec relative to the world axes (*see Note at end of table*). This is in double floating point format (FLOAT64). [FS2002+]	**Ok**	
31D0	8	Z (longitudinal, or forward/backward) acceleration in ft/sec/sec relative to the world axes (*see Note at end of table*). This is in double floating point format (FLOAT64). [FS2002+]	**Ok**	
31D8	2	Slew mode longitudinal axis (i.e. forward/backward) input value, post calibration, just before being applied to the simulation (if allowed to by the byte at offset 310B)	**Ok**	**Ok**
31DA	2	Slew mode lateral axis (i.e. left/right) input value, post calibration, just before being applied to the simulation (if allowed to by the byte at offset 310B)	**Ok**	**Ok**
31DC	2	Slew mode yaw axis (i.e. heading) input value, post calibration, just before being applied to the simulation (if allowed to by the byte at offset 310B)	**Ok**	**Ok**
31DE	2	Slew mode vertical axis (i.e. altitude) input value, post calibration, just before being applied to the simulation (if allowed to by the byte at offset 310B)	**Ok**	**Ok**
31E0	2	Slew mode roll axis (i.e. bank) input value, post calibration, just before being applied to the simulation (if allowed to by the byte at offset 310B)	**Ok**	**Ok**
31E2	2	Slew mode pitch axis input value, post calibration, just before being applied to the simulation (if allowed to by the byte at offset 310B)	**Ok**	**Ok**
31E4	4	Radio altitude in metres * 65536 (Calculated by FSUIPC from ground altitude and aircraft altitude)	**Ok**	**Ok**
31E8	4	Surface type as a 32-bit integer (FS2002+ only). I think this only applies when the aircraft is on the ground. The values probably correspond to the surface encoding in the scenery files, thus: CONCRETE 0 GRASS 1 SOFT, BUMPY GROUND (LANDABLE) WATER 2 GRASS BUMPY 3 VERY BUMPY GRASS & MUD (CRASHABLE) ASPHALT 4 SHORT GRASS 5 LONG GRASS 6 HARD TURF 7 SNOW 8 ICE 9 URBAN 10 FOREST 11 DIRT 12 CORAL 13 GRAVEL 14 OIL TREATED 15 TAR & CHIP STEEL MATS 16 STEEL MESH TEMPORARY RUNWAYS BITUMINUS 17 BRICK 18 MACADAM 19 PLANKS 20	**Ok**	**Ok**

		SAND 21 SHALE 22 TARMAC 23 UNKNOWN 254		
31EC	4	Surface condition as a 32-bit integer (FS2002+ only), probably as follows: NORMAL 0 WET 1 ICY 2 SNOW 3 SNOW ON A NON-SNOW SURFACE	**Ok**	**Ok**
31F0	4	Pushback status (FS2002+). 3=off, 0=pushing back, 1=pushing back, tail to swing to left (port), 2=pushing back, tail to swing to right (starboard)	**Ok**	**Ok**
31F4	4	Pushback control (FS2002+). Write 0–3 here to set pushback operation, as described for the status, above.	**Ok**	**Ok**
31F8	4	Tug Heading control, for gliders I assume (FS2002+ only). [*write only*]. The units appear to be the same as the aircraft heading usits (see offset 0580).	**Ok?**	**Ok?**
31FC	4	Tug Speed control, for gliders I assume (FS2002+ only). [*write-only*]. Units not confirmed, but possible ft/sec.	**Ok?**	**Ok?**
3200	12	These locations operate the FSUIPC facility to send keystrokes to FS. For this to operate correctly the PC must be using Windows 98, ME or 2000. The facilities used just do not exist in Windows 95 nor NT. 3200 message (WM_KEYDOWN or WM_KEYUP) 3204 wParam for the message 3208 lParam for the message All 12 bytes must be written in one IPC write. (This feature is used in WideClient version 3.998 and later, when the [User] parameter "SendKeyPresses=Yes" is included in its .ini file, to relay all non-system (i.e. no Alt key) key presses it receives to the WideServer host).	**Ok**	**Ok**
320C	4	Number of Hot Key slots available for Application Programs to use. Currently this is fixed at 56, representing the 56 DWORDs available in the following offsets:	**Ok**	**Ok**
3210	224	56 DWORDs containing zero (when free for use), or a Hot Key specification as detailed earlier in this document. See also 32FE below.	**Ok**	**Ok**
32F0	4	This DWORD controls some protected mode facilities in FSUIPC, designed to set known conditions in FSUIPC and prevent access to specific menus, whilst an application is running. The whole 32 bit DWORD should be written at once, but the use is divided into Bytes, as follows: Bits 0–7 (byte at 32F0): FSUIPC option settings 2^0 Sets FSUIPC "normal defaults" 2^1 Sets FSUIPC "minimum weather defaults" any non-zero value in this byte stops entry to FSUIPC options Bits 8–15 (byte at 32F1): Flight Sim menu restrictions 2^10 Disable World menu 2^11 Disable Aircraft menu 2^12 Disable Flights menu 2^13 Disable Options menu 2^14 Disable Flights, Aircraft and World menus 2^15 Disable ALL Menus Bits 16–23 (byte at 32F2): *reserved* Bits 24–31 (byte at 32F3): Timeout (in ticks or 55 mSecs units)	**Ok**	**Ok**

		The application must write this DWORD regularly for the restrictions to stay in place. The count in the high byte is decremented by 1 every 55 mSecs, so a maximum time of 14 seconds can be set. To be safe the application should be re-writing this with a count of FF (255) every 5 or so seconds, especially if it is likely to be running across WideFS. When the count expires, or the application whites a zero DWORD here, all the options and menus return to normal.		
32F4	2	The 16-bit ID of the last menu command item accessed in FS can be read here. By "access" is not meant "used"–that cannot be determined easily. Just having a menu command highlit will denote an access. To decode command Ids, use FSUIPC logging. First, before running FSUIPC set "Debug=Please" and "LogExtras=64" into the FSUIPC.INI file. Then run FS and select the menu items in which you are interested. Examine the FSUIPC Log afterwards to determine the ID.	**Ok**	
32F6	2	FSUIPC selected technical option inhibits. Set bits here to turn *off* specific options and prevent the user turning them back on, for a limited time (max 14 seconds). To keep options turned off you need to write this mask at regular intervals (e.g. every 5 seconds). Note that this is not obeyed if the user has selected to option to disallow all external control of his options. If he has done this, you can detect it by reading this location back within the time limit. If it is zero, not the value written, then the user is preventing your control over his settings. Bits allocated so far are as follows (bit 0 = 2^0 bit): 0 Reverse elevator trim sense 1 Fix control accelerations 2 Rudder spike elimination 3 Elevator spike elimination 4 Aileron spike elimination 5 Autopilot altitude fix (enable V/S sign corrn.) 6 Extend battery life 7 FS clock seconds sync	**Ok**	**Ok**
32F8	1	This provides options to inhibit certain aircraft operations, for use in breakdown or precise control implementations. Set individual bits for individual subsystems. Currently the following are available, all related to hydraulic power: 2^0 Set to inhibit flap operation 2^1 Set to inhibit spoiler operation 2^2 Set to inhibit gear operation 2^3 *reserved* 2^4 Set to inhibit Engine #1 reverser 2^5 Set to inhibit Engine #2 reverser 2^6 Set to inhibit Engine #3 reverser 2^7 Set to inhibit Engine #4 reverser Note that these stop operation from axis and button controls very well, and also from key presses and mouse clicks—but in these latter two cases it is done by detecting a change in the system and changing it back. This works, but the device will sometimes try to move, and this can be noticeable, especially for some reason with the flaps—the indicator gives a little jump and the noise briefly starts. [FS2002 and later only]	**Yes**	**Yes**
32F9	1	Brakes being used flag. This is non-zero if the user has pressed	**Yes, but**	**Yes**

		the brakes (left, right or both) recently. It stays non-zero for a second after the last brake control or significant axis increase seen. It does *not* stay set for continued constant brake pressure via the axis inputs. It operates also for increasing values written to offset 0C00 or 0C01. Note: this is only fully implemented for FS2004+. It will work for buttons and axes in earlier versions, but not keyboard braking.		
32FA	2	Text display control word. You can display messages from an external program just like an Adventure. Write the message as a zero-terminated string to offset 3380 (see below), subject to the maximum of 128 characters *including* the zero terminator, then write a number to this offset, 32FA, as follows: 0 display till replaced +n display for n seconds, or until replaced –1 display and scroll, or until replaced –n display and scroll, or for n seconds, or until replaced In the last two cases, whether the message scrolls or not depends upon the setting of the "Options—Settings—General—Text Display" option. The time limit only applies when scrolling is off, otherwise the message simply expires when fully scrolled off the screen. See also offset 1274 above, and the "white messages" option in 3302 below.	**Ok**	**Ok**
32FC	2	AIR file change counter (incremented by FSUIPC whenever the AIR file as defined at offset 3C00 changes). This is also incremented when the FS2004 control to "reload user aircraft" is detected—assign it to a joystick button or to a Key in FSUIPC for this. FSUIPC cannot detect controls arising from key presses assigned in FS dialogues.	**Ok**	**Ok**
32FE	1	Hot Key change counter, incremented by FSUIPC whenever any of the Hot Keys defined in the table at offset 3210 occurs and therefore has its flag set by FSUIPC.	**Ok**	**Ok**
32FF	1	Hot Button change counter, incremented by FSUIPC whenever any of the Hot Buttons defined in the table at offset 2910 changes state in the right way, and therefore has its flag set by FSUIPC.	**Ok**	**Ok**
3300	2	[FS2k, CFS2 *and* FS98, as applicable] Additional radio and autopilot status indicators (read only access). Allocation by bits which are set when true. Bit 0 = least significant (value 1): 0 = reserved 1 = good NAV1 2 = good NAV2 3 = good ADF1 4 = NAV1 has DME 5 = NAV2 has DME 6 = NAV1 is ILS 7 = AP NAV1 radial acquired 8 = AP ILS LOC acquired (incl BC—see 10) 9 = AP ILS GS acquired 10=AP ILS LOC is BC 11=good ADF2 (FS2004)	**Ok**	**Ok**

		12=NAV2 is ILS 13–15 reserved		
3302	2	Assorted FSUIPC options, set by user parameters: read-only via the IPC. Those allocated so far (bits from least significant): 0 = Static (i.e. non-scrolling) messages sent to FS are to be displayed in white rather than the default red. (If AdvDisplay is installed it must be version 2.11 or later for this option). 1 = This is FS2004 (or later) but MakeItVersionFS2002 has been used in the INI to "fiddle" the reported value in 3308 to show FS2002.	**Ok**	**Ok**
3304	4	FSUIPC version number: The HIWORD (i.e. bytes 3306-7) gives the main version as BCD x 1000: e.g. 0x1998 for 1.998 The LOWORD (bytes 3304-5) gives the Interim build letter: 0=none, 1-26=a-z: e.g. 0x0005 = 'e'	**Ok**	**Ok**
3308	2	FS version, as determined by FSUIPC: Currently only one of these: 1 = FS98 2 = FS2000 3 = CFS2 4 = CFS1 5 = reserved 6 = FS2002 7 = FS2004 "A Century of Flight" 8 = FSX 9 = ESP *(see also bit 1 in offset 3302 above)*	**Ok**	**Ok**
330A	2	Fixed *read-only* pattern, set to 0xFADE. Use this to check that the values in 3304-3308 are valid (Note: the supplied LIB writes its version number here, but this has no effect and is only for assistance when viewing LOG files).	**Ok**	**Ok**
330C	2	Assorted status flags, the only ones which are of use to applications being: 2^1 When set this indicates that programs have full access to the IPC not. This can be read without triggering the message box to users which tells them of an unaccredited access attempt. Note that on WideClient it will always be set, assuming WideServer is registered on the FS PC. 2^2 Set if the user has fully registered FSUIPC 2^3 Set when FS is closing (EXIT event seen). This applies to FS2004 only, and isn't actually likely to be readable in time by applications. 2^4 Set when the user Throttle Sync option (in the Hot Keys page of FSUIPC options) is enabled.	**Variable, see text**	**Ok**
330E	1	Count of external IPC applications seen connecting since the session began. Keeps increasing till it gets to 255 then stays at that value.	**Ok**	**Ok**
330F	17	Reserved area for WideFS KeySend facility (version 4.23 and later)	**Ok**	**Ok**
3320	2	This word is used to activate a facility supported by WideFS to automatically shut down the PCs running WideServer (i.e. this one) and WideClient. The .ini files of each WideFS component which is to activate the shutdown needs the "AllowShutdown=Yes" parameter included. The application performing the shut down action must write 0xABCD to this	**Ok**	**Ok**

		offset. WideServer automatically resets this word to zero 5 seconds afterwards, before it initiates its own PC's shutdown if specified. This delay is to ensure the Clients get the message before the host dies, and the clearing to zero is done so that the survivors can continue. Since version 5.30, WideFS also provides the lesser option "AllowShutdown=App" which only closes down the WideClient or, in the case of WideServer, the FS session. Later still the "AppOnly" variation was added, which keeps WideClient running, ready to reload the applications when FS restarts. A hot key facility to invoke this WideFS shutdown from the FS keyboard is added in version 5.301 of WideServer. Since version 6.40 of WideFS the pattern 0xDCBA written here invokes a "close application" action. On all WideFS PCs with any form of shutdown allowed, this pattern closes only those applications loaded by WideFS and leaves WideClient running ready to reload them. On the Server, if it is allowed, it closes FS itself. A hot key facility is provided for this variant, too.		
3322	2	WideServer version number, if running *and* if version 5.00 or later. Otherwise this is zero. This is a BCD value giving the version number x 1000, for example 0x5110 means version 5.110. See also offset 333C.	**Ok**	**Ok**
3324	4	This is the altimeter reading in feet (or metres, if the user is running with the preference for altitudes in metres), as a 32-bit signed integer. Please check offset 0C18 to determine when metres are used (0C18 contains '2'). The same value can be calculated from the actual altitude and the difference between the QNH and the altimeter "Kollsman" pressure setting, but this value ensures agreement.	**Ok**	**Ok**
3328	2	Elevator Axis input value, post calibration, just before being applied to the simulation (if allowed to by the byte at offset 310A).	**Ok**	**Ok**
332A	2	Aileron Axis input value, post calibration, just before being applied to the simulation (if allowed to by the byte at offset 310A).	**Ok**	**Ok**
332C	2	Rudder Axis input value, post calibration, just before being applied to the simulation (if allowed to by the byte at offset 310A).	**Ok**	**Ok**
332E	2	Throttle Axis input value, post calibration, just before being applied to the simulation (if allowed to by the byte at offset 310A). This is the single throttle, applied to whichever engines are denoted by the bits in offset 0888.	**Ok**	**Ok**
3330	2	Throttle 1 Axis input value, post calibration, just before being applied to the simulation (if allowed to by the byte at offset 310A).	**Ok**	**Ok**
3332	2	Throttle 2 Axis input value, post calibration, just before being applied to the simulation (if allowed to by the byte at offset 310A).	**Ok**	**Ok**
3334	2	Throttle 3 Axis input value, post calibration, just before being applied to the simulation (if allowed to by the byte at offset	**Ok**	**Ok**

		310A).		
3336	2	Throttle 4 Axis input value, post calibration, just before being applied to the simulation (if allowed to by the byte at offset 310A).	**Ok**	**Ok**
3338	2	Elevator Trim Axis input value, post calibration, just before being applied to the simulation (if allowed to by the byte at offset 310A).	**Ok**	**Ok**
333A	2	Throttle lower limit (FS2002 and later only). This is normally 0 if no reverse is available, otherwise gives the reverse limit such as –4096 (for 25%). For earlier versions than FS2002 this location will be zero.	**Ok**	**Ok**
333C	2	WideFS flags: only set from version 5.50 or later of WideFS. Flags used so far are: 2^0 1 =if TCP is being used, 0 if SPX 2^1 1 if connected at all, 0 is waiting for connections See offset 3322 for WideFS version number, which also confirms that WideServer is installed and running.	**Ok**	**Ok**
333E	2	Weather clear count: This is incremented every time FS's "clear weather" routine is called, for whatever reason.	**Ok**	**Ok**
3340	36	This area is used for externally signalled "joystick button" control a set of 288 "virtual buttons". Each DWORD or 32 bits represents one "joystick" with 32 buttons. If an external program sets or clears a bit in any of these 9 DWORDS the "Buttons" page in FSUIPC will register the change as a button operation on one of Joystick numbers 64 to 73 (corresponding to the 9 DWORDs). So, FSUIPC can be used to program whatever actions the user wants. See also offset 29F0.	**Ok**	**Ok**
3364	1	FS2004 "Ready to Fly" indicator. This is non-zero when FS is loading, or reloading a flight or aircraft or scenery, and becomes zero when flight mode is enabled (even if the simulator is paused or in Slew mode).	**No**	**Ok**
3365	1	"In Menu or Dialog" flag. This byte is non-zero when FS is effectively paused because the user accessed the Menu, or is in a dialogue resulting from menu or other selection activity. The non-zero values are: 1 = FS frozen because of menu activity 2 = FS frozen because of modal dialogue Both bits may be set in dialogues accessed through the menu. Note that the 2 bit may flicker a little on exit from the dialogue, due to the way it is detected. In FS2004 the byte at 3364 should also be considered when using this flag—there are some conditions, like reloading scenery or aircraft or flights, which effectively freeze the sim in ways not detectable except by the method used for the "ready to fly" indicator.	**Ok**	**Ok**
3366	1	This byte reflects the FS2004 "Engine on Fire" flags. I'm not sure if FS actually simulates such events, but it appears to have allocated Gauge-accessible variables to indicate them. This byte uses bits 2^0–2^3 as flags for fires in Engines 1 to 4, respectively.	**No**	**Ok**
3367	1	This byte shows doors that are open (FS2004 only). At present this only provides bit 2^0 for the main doors.	**No**	**Ok**
3368	4	*Reserved* for PFC.DLL events.		
336C	2	Frame rate calling counter. This is simply a number that is	**Ok**	**Ok**

		incremented each time FSUIPC is entered from FS using the entry related to frame rates.		
336E	2	Toe brake axes have been selected as "Set" in FSUIPC's joystick pages if this is non-zero. Byte 336E is non-zero for Left Brake, byte 336F for Right Brake. Note that this only means that the user has told FSUIPC to handle the toe braking, by pressing "Set". It will only actually do so if it sees brake messages.	**Ok**	**Ok**
3370	4	Four single byte PFC driver "alive" counters: 3370 = COM port read thread alive and running 3371 = Elevator trim motor action (0=off, 1=up, 2=dn) 3372 = COM port write thread alive 3373 = Main FS chain alive N.B. without the main FS chain running the other three aren't maintained in any case, so mean nothing.	**Ok**	**Ok**
3374	4	This is the "live" millisecond count as used in the FSUIPC Log. It is updated on each FS chained call to FSUIPC.	**Ok**	**Ok**
3378	4	This is the millisecond timestamp value of the most recent line in the current FSUIPC Log. It is updated when each line is logged.	**Ok**	**Ok**
337C	1	Propeller de-ice switch, (1 = on, 0 = off), read for state, write to control [FS2002+]. *This should operate with aircraft defined to have the facility, but in fact it merely reflects the older Anti-Ice switch. The* TOGGLE_PROP_DEICE *control does nothing.*		
337D	1	Structural de-ice switch, (1 = on, 0 = off), read for state, write to control [FS2002+]. *Although this is documented in both FS2002 and FS2004 panel SDKs, with a token value and an FS control, it appears to do nothing. Possibly it needs some action in the AIR file or Aircraft.CFG, but there's nothing in the official documentation.*		
337E	2	FSUIPC activity count. Simply a number that is incremented every time FSUIPC receives a call or message from Flight Simulator. This can be used through WideFS to check if FS is still active, for example. Note that when FS is loading aircraft or scenery/textures, this value may not change for many seconds as FSUIPC is then not getting any processor time at all.	**Ok**	**Ok**
3380	128	Message text area, where AdvDisplay.dll (if running) stores a copy of the ADVenture text display. This can be useful for programs wishing to display the adventure texts on a separate PC, via WideFS (the freeware ShowText.exe is an example). The text is truncated if longer than 127 characters, there always being a zero terminator provided. You can also *write* messages to this area, always zero terminated, for display on the FS windshield *or* via AdvDisplay if it is running. After placing the message text, you must write the 16-bit timer value to offset 32FA to make FSUIPC send the message through to FS (see 32FA above).	**Ok**	**Ok**
3410	2	Assorted indicator flags. These are the only ones currently set (bit numbers, bit 0 = 2^0): 4 Engine 1 Reverser is set but inhibited* 5 Engine 2 Reverser is set but inhibited* 6 Engine 3 Reverser is set but inhibited* 7 Engine 4 Reverser is set but inhibited* * Reverser inhibits are set in offset 32F8. Note that these flags will be cleared only when the inhibit is removed *or* the relevant throttle input goes positive (i.e. not just to idle).	**Ok**	**Ok**
3412	2	Spoiler Axis input value, post calibration, just before being applied to the simulation (if allowed to by the byte at offset	**Ok**	**Ok**

		341A). Copy this to 0BD0 for normal spoiler action.		
3414	2	Flaps Axis input value, post calibration, just before being applied to the simulation (if allowed to by the byte at offset 341A). Copy this to 0BDC for normal flaps action.	**Ok**	**Ok**
3416	2	Left Brake Axis input value, post calibration, just before being applied to the simulation (if allowed to by the byte at offset 341A). Copy this to 0BC4 for normal left brake action.	**Ok**	**Ok**
3418	2	Right Brake Axis input value, post calibration, just before being applied to the simulation (if allowed to by the byte at offset 341A). Copy this to 0BC6 for normal right brake action.	**Ok**	**Ok**
341A	1	Controls the joystick connection for ancillary axis controls, currently Left and Right brake, flaps and spoiler axes. Normally all zero, set the following bits to actually disconnect the specific joystick axes (from least significant bit = 0): 0 Left brake ("Axis Left Brake Set") 1 Right Brake ("Axis Right Brake Set") 2 Flaps 3 Spoilers This feature is intended for use in simulating relevant subsystem failures or partial failures. Programs can read the input axis values from offsets 3412–3418 above, and apply them, after appropriate modification, to the relevant FS axis offsets (at 0BC4 and 0BC6 for Brakes, 0BDC for Flaps or 0BD0 for Spoiler. In order to protect the user from a broken or crashed application, the flags are cleared 10 seconds after they have been set, so applications will need to repeat the setting every few seconds. Note that this byte is effectively "write only". Upon reading it will always appear to contain zero.	**Ok**	**Ok**
3470	8	Ambient wind X component, double float (FS2002+), m/sec	**Ok**	**Ok**
3478	8	Ambient wind Y component, double float (FS2002+)., m/sec In FS2004 (only), values written here are sustained by FSUIPC for up to 14 seconds, or until another value is written. This can be used by applications to provide things like lift for glider soaring, or fast varitions to emulate vertical turbulence. The values written here are *not* subject to FSUIPC's smoothing, and won't be reflected in any normal weather read-outs like ATIS.	**Ok**	**Ok (with extra facilities)**
3480	8	Ambient wind Z component, double float (FS2002+), m/sec	**Ok**	**Ok**
3488	8	Ambient wind velocity, double float (FS2002+), m/sec	**Ok**	**Ok**
3490	8	Ambient wind direction, double float (FS2002+), True	**Ok**	**Ok**
3498	8	Ambient pressure, double float (FS2002+). This is accurate in FS2004, but suspicious in FS2002.	**See text**	**Ok**
34A0	8	Sea level pressure (QNH), double float (FS2002+)	**Ok**	**Ok**
34A8	8	Ambient temperature, double float (FS2002+)		**Ok**
34B0	8	Pressure Altitude (metres), double float. This is the indicated altitude when the altimeter Kollsman setting is 1013.2 hPa (29.92").	**Ok**	**Ok**
34B8	8	Standard ATM Temperature, degrees Rankine, double float. This is the expected temperature at the actual AMSL in the International Standard Atmosphere model.	**Ok**	**Ok**
34C8	8	Total velocity, ft/sec, double float. This is the resultant velocity of the three X,Y,Z orthogonal velocities given in offsets 3178, 3180 and 3188.	**No**	**Ok**
3500	24	ATC aircraft model string for currently loaded user aircraft, as declared in the AIRCRAFT.CFG file. This is limited to a	**Ok**	**Ok**

		maximum of 24 characters, including a zero terminator. [FS2002+ only]		
3518	8	This double provides the FS-set "Ambient Wind Y" value within about one second of offset 3478 being written by an application, to control up and down drafts. This allows such a program to monitor FS/scenery arranged updrafts and adjust its actions accordingly. [FS2004 only]	**No**	**Ok**
3520	2	Earliest version number of connected WideClients (or clients which have been connected). Zero if no connections have been made, or if all connected clients have been version 6.441 or before.	**Ok**	**Ok**
3541	1	This operates the FSUIPC "freeze flight position" facility. This keeps the aircraft at the same latitude and longitude for as long as it is engaged. The altitude and attitude of the aircraft is free to change, and, in fact, the aircraft flies as normal except for not changing its position over the ground. This is apparently a very useful facility for training environments. For program control, write a non-zero values to this one byte offset. This acts as a timer. The freeze will last for as long as this byte is non-zero. It is used as a time, counting down 1 every timer tick of 55 mSecs or so. To retain the freeze for a good time, write 255 here and do so every 5–10 seconds. Allow for WideFS delays. Note that if FS is paused, then the freeze lasts until the pause is released and re-engaged.	**Ok, but not smooth, and FS jumps to "real" position when released**	**Ok, works well.**
3542	2	Standby altimeter pressure setting ("Kollsman" window). As millibars (hectoPascals) * 16. [*This is used by FSUIPC to maintain offset 3544. It is not used by FS at all*]	**Ok**	**Ok**
3544	4	This is the standby altimeter reading in feet (or metres, if the user is running with the preference for altitudes in metres), as a 32-bit signed integer. Please check offset 0C18 to determine when metres are used (0C18 contains '2'). This value is maintained by FSUIPC using the pressure setting supplied in offset 3542. It isn't used in FS itself, but is supplied for additional gauges and external altimeters so that the standby can be kept at the correct (or last notified) QNH whilst the main altimeter is used for Standard settings (for airliners flying Flight Levels).	**Ok**	**Ok**
3548	8	Horizon bars offset, as a percentage of maximum, in floating point double format. (–100.0 down to +100.0 up). On the default Cessnas the maximum offset is 10 degrees. [Read only on FS2004]	**Ok**	**Ok**
3550	56	*Reserved for FSUIPC diagnostics related to Gauge Mousing*		
3590	4	Engine 1 Fuel Valve, 1 = open, 0 = closed. Can write to operate. [FS2002+]		**Ok**
3594	4	Engine 2 Fuel Valve, 1 = open, 0 = closed. Can write to operate. [FS2002+]		**Ok**
3598	4	Engine 3 Fuel Valve, 1 = open, 0 = closed. Can write to operate. [FS2002+]		**Ok**
359C	4	Engine 4 Fuel Valve, 1 = open, 0 = closed. Can write to operate. [FS2002+]		**Ok**
35A0	8	Airspeed Mach value, double float (FS2002+)	**Ok**	**Ok**
35A8	8	Reciprocating engine 4 manifold pressure, in lbs/sqft, as a double (FLOAT64). Divide by 70.7262 for inches Hg.	**Ok**	**Ok**
35B0	8	Engine 4 cowl flap position, as a double float: 0.0=fully closed, 1.0=fully open. Can be used to handle position and set it. .	**Ok**	**Ok**

		[FS2000–FS2004 only]		
35D0	4	Reciprocating engine 4, left magneto select (1 = on, 0 = off)	Ok	Ok
35D4	4	Reciprocating engine 4, right magneto select (1 = on, 0 = off)	Ok	Ok
35D8	8	Reciprocating engine 4 fuel/air mass ratio, as a double (FLOAT64).	Ok	Ok
35E0	8	Reciprocating engine 4 brake power in ft-lbs, as a double (FLOAT64). Divide by 550 for HP.	Ok	Ok
35E8	8	Reciprocating engine 4 carburettor temperature, in degrees Rankine, as a double (FLOAT64).	Ok	Ok
35FC	4	Reciprocating engine 4 emergency boost active flag (32-bit BOOLEAN). On some aircraft this controls whether the supercharger is active or not.	Ok	Ok
3600	8	Reciprocating engine 4 emergency boost elapsed time in seconds, as a double (FLOAT64). This counts how long the boost has been engaged, when it is made active by an FS control. FS turns it off when reaching 312. You can keep it going by occasionally writing 0 here.	Ok	Ok
3608	8	Reciprocating engine 4 wastegate position (read-only, effectively)	Ok	Ok
3628	8	Reciprocating engine 4 fuel pressure (double or FLOAT64)	Ok	Ok
3640	4	Reciprocating engine 4 tank selector, using the same numbers as 0AF8.	Ok	Ok
3648	4	Reciprocating engine 4, number of fuel tanks supplying fuel.	Ok	Ok
3654	4	Reciprocating engine 4 fuel available flag (0 or 1).	Ok	Ok
3668	8	Reciprocating engine 3 manifold pressure, in lbs/sqft, as a double (FLOAT64). Divide by 70.7262 for inches Hg.	Ok	Ok
3670	8	Engine 3 cowl flap position, as a double float: 0.0=fully closed, 1.0=fully open. Can be used to handle position and set it. [FS2000–FS2004 only]	Ok	Ok
3690	4	Reciprocating engine 3, left magneto select (1 = on, 0 = off)	Ok	Ok
3694	4	Reciprocating engine 3, right magneto select (1 = on, 0 = off)	Ok	Ok
3698	8	Reciprocating engine 3 fuel/air mass ratio, as a double (FLOAT64).	Ok	Ok
36A0	8	Reciprocating engine 3 brake power in ft-lbs, as a double (FLOAT64). Divide by 550 for HP.	Ok	Ok
36A8	8	Reciprocating engine 3 carburettor temperature, in degrees Rankine, as a double (FLOAT64). [*FSUIPC version 3.401 or later*]	Ok	Ok
36BC	4	Reciprocating engine 3 emergency boost active flag (32-bit BOOLEAN). On some aircraft this controls whether the supercharger is active or not.	Ok	Ok
36C0	8	Reciprocating engine 3 emergency boost elapsed time in seconds, as a double (FLOAT64). This counts how long the boost has been engaged, when it is made active by an FS control. FS turns it off when reaching 312. You can keep it going by occasionally writing 0 here.	Ok	Ok
36C8	8	Reciprocating engine 3 wastegate position (read-only, effectively)	Ok	Ok
36E8	8	Reciprocating engine 3 fuel pressure (double or FLOAT64)	Ok	Ok
3700	4	Reciprocating engine 3 tank selector, using the same numbers as 0AF8.	Ok	Ok
3708	4	Reciprocating engine 3, number of fuel tanks supplying fuel.	Ok	Ok
3714	4	Reciprocating engine 3, fuel available flag (0 or 1).	Ok	Ok
3728	8	Reciprocating engine 2 manifold pressure, in lbs/sqft, as a double (FLOAT64). Divide by 70.7262 for inches Hg.	Ok	Ok
3730	8	Engine 2 cowl flap position, as a double float: 0.0=fully closed, 1.0=fully open. Can be used to handle position and set it. . [FS2000–FS2004 only]	Ok	Ok
3750	4	Reciprocating engine 2, left magneto select (1 = on, 0 = off)	Ok	Ok

3754	4	Reciprocating engine 2, right magneto select (1 = on, 0 = off)	Ok	Ok
3758	8	Reciprocating engine 2 fuel/air mass ratio, as a double (FLOAT64).	Ok	Ok
3760	8	Reciprocating engine 2 brake power in ft-lbs, as a double (FLOAT64). Divide by 550 for HP.	Ok	Ok
3768	8	Reciprocating engine 2 carburettor temperature, in degrees Rankine, as a double (FLOAT64). [*FSUIPC version 3.401 or later*]	Ok	Ok
377C	4	Reciprocating engine 2 emergency boost active flag (32-bit BOOLEAN). On some aircraft this controls whether the supercharger is active or not.	Ok	Ok
3780	8	Reciprocating engine 2 emergency boost elapsed time in seconds, as a double (FLOAT64). This counts how long the boost has been engaged, when it is made active by an FS control. FS turns it off when reaching 312. You can keep it going by occasionally writing 0 here.	Ok	Ok
3788	8	Reciprocating engine 2 wastegate position (read-only, effectively)	Ok	Ok
37A8	8	Reciprocating engine 2 fuel pressure (double or FLOAT64)	Ok	Ok
37C0	4	Reciprocating engine 2 tank selector, using the same numbers as 0AF8.	Ok	Ok
37C8	4	Reciprocating engine 2, number of fuel tanks supplying fuel.	Ok	Ok
37D4	4	Reciprocating engine 2, fuel available flag (0 or 1).	Ok	Ok
37E8	8	Reciprocating engine 1 manifold pressure, in lbs/sqft, as a double (FLOAT64). Divide by 70.7262 for inches Hg.	Ok	Ok
37F0	8	Engine 1 cowl flap position, as a double float: 0.0=fully closed, 1.0=fully open. Can be used to handle position and set it. [FS2000–FS2004 only]	Ok	Ok
3810	4	Reciprocating engine 1, left magneto select (1 = on, 0 = off)	Ok	Ok
3814	4	Reciprocating engine 1, right magneto select (1 = on, 0 = off)	Ok	Ok
3818	8	Reciprocating engine 1 fuel/air mass ratio, as a double (FLOAT64).	Ok	Ok
3820	8	Reciprocating engine 1 brake power in ft-lbs, as a double (FLOAT64). Divide by 550 for HP.	Ok	Ok
3828	8	Reciprocating engine 1 carburettor temperature, in degrees Rankine, as a double (FLOAT64). [*FSUIPC version 3.401 or later*]	Ok	Ok
383C	4	Reciprocating engine 1 emergency boost active flag (32-bit BOOLEAN). On some aircraft this controls whether the supercharger is active or not.	Ok	Ok
3840	8	Reciprocating engine 1 emergency boost elapsed time in seconds, as a double (FLOAT64). This counts how long the boost has been engaged, when it is made active by an FS control. FS turns it off when reaching 312. You can keep it going by occasionally writing 0 here.	Ok	Ok
3848	8	Reciprocating engine 1 wastegate position (read-only, effectively)	Ok	Ok
3868	8	Reciprocating engine 1 fuel pressure (double or FLOAT64)	Ok	Ok
3880	4	Reciprocating engine 1 tank selector, using the same numbers as 0AF8.	Ok	Ok
3888	4	Reciprocating engine 1, number of fuel tanks supplying fuel.	Ok	Ok
3894	4	Reciprocating engine 1, fuel available flag (0 or 1).	Ok	Ok
38A8	8	General engine 4 throttle lever position, as a double (FLOAT64). 0.0=idle, 1.0=max	Ok	Ok
38B0	8	General engine 4 mixture lever position, as a double (FLOAT64). 0.0=cutoff, 1.0=full rich	Ok	Ok
38B8	8	General engine 4 propeller lever position, as a double (FLOAT64). 0–1	Ok	Ok
3918	8	General engine 4 oil temperature in degrees Rankine, as a double	Ok	Ok

		(FLOAT64).		
3920	8	General engine 4 oil pressure in lbs/sqft, as a double (FLOAT64). Divide by 144 for PSI.	**Ok**	**Ok**
3930	8	General engine 4 EGT in degrees Rankine, as a double (FLOAT64). Convert to Fahrenheit by Rankine – 459.67. FS default gauges show Centigrade.	**Ok**	**Ok**
3938	4	Engine 4 generator switch, a 32-bit BOOL (0 = off, 1= on) [FS2000–FS2004 only]	**Ok**	**Ok**
393C	4	Engine 4 generator active, a 32-bit BOOL (0 = off, 1= on) [FS2000–FS2004 only]	**Ok**	**Ok**
3958	4	Engine 4 fuel pump switch, a 32-bit BOOL (0 = off, 1= on) [FS2000–FS2004 only]	**Ok**	**Ok**
3968	8	General engine 3 throttle lever position, as a double (FLOAT64). 0.0=idle, 1.0=max	**Ok**	**Ok**
3970	8	General engine 3 mixture lever position, as a double (FLOAT64). 0.0=cutoff, 1.0=full rich	**Ok**	**Ok**
3978	8	General engine 3 propeller lever position, as a double (FLOAT64). 0–1	**Ok**	**Ok**
39D8	8	General engine 3 oil temperature in degrees Rankine, as a double (FLOAT64).	**Ok**	**Ok**
39E0	8	General engine 3 oil pressure in lbs/sqft, as a double (FLOAT64). Divide by 144 for PSI.	**Ok**	**Ok**
39F0	8	General engine 3 EGT in degrees Rankine, as a double (FLOAT64). Convert to Fahrenheit by Rankine – 459.67. FS default gauges show Centigrade.	**Ok**	**Ok**
39F8	4	Engine 3 generator switch, a 32-bit BOOL (0 = off, 1= on) [FS2000–FS2004 only]	**Ok**	**Ok**
39FC	4	Engine 3 generator active, a 32-bit BOOL (0 = off, 1= on) [FS2000–FS2004 only]	**Ok**	**Ok**
3A18	4	Engine 3 fuel pump switch, a 32-bit BOOL (0 = off, 1= on) [FS2000–FS2004 only]	**Ok**	**Ok**
3A28	8	General engine 2 throttle lever position, as a double (FLOAT64). 0.0=idle, 1.0=max	**Ok**	**Ok**
3A30	8	General engine 2 mixture lever position, as a double (FLOAT64). 0.0=cutoff, 1.0=full rich	**Ok**	**Ok**
3A38	8	General engine 2 propeller lever position, as a double (FLOAT64). 0–1	**Ok**	**Ok**
3A98	8	General engine 2 oil temperature in degrees Rankine, as a double (FLOAT64).	**Ok**	**Ok**
3AA0	8	General engine 2 oil pressure in lbs/sqft, as a double (FLOAT64). Divide by 144 for PSI.	**Ok**	**Ok**
3AB0	8	General engine 2 EGT in degrees Rankine, as a double (FLOAT64). Convert to Fahrenheit by Rankine – 459.67. FS default gauges show Centigrade.	**Ok**	**Ok**
3AB8	4	Engine 2 generator switch, a 32-bit BOOL (0 = off, 1= on) [FS2000–FS2004 only]	**Ok**	**Ok**
3ABC	4	Engine 2 generator active, a 32-bit BOOL (0 = off, 1= on) [FS2000–FS2004 only]	**Ok**	**Ok**
3AD8	4	Engine 2 fuel pump switch, a 32-bit BOOL (0 = off, 1= on) [FS2000–FS2004 only]	**Ok**	**Ok**
3AE8	8	General engine 1 throttle lever position, as a double (FLOAT64). 0.0=idle, 1.0=max	**Ok**	**Ok**
3AF0	8	General engine 1 mixture lever position, as a double (FLOAT64). 0.0=cutoff, 1.0=full rich	**Ok**	**Ok**
3AF8	8	General engine 1 propeller lever position, as a double (FLOAT64). 0–1	**Ok**	**Ok**
3B58	8	General engine 1 oil temperature in degrees Rankine, as a double (FLOAT64).	**Ok**	**Ok**
3B60	8	General engine 1 oil pressure in lbs/sqft, as a double	**Ok**	**Ok**

		(FLOAT64). Divide by 144 for PSI.		
3B70	8	General engine 1 EGT in degrees Rankine, as a double (FLOAT64). Convert to Fahrenheit by Rankine – 459.67. FS default gauges show Centigrade.	**Ok**	**Ok**
3B78	4	Engine 1 generator switch, a 32-bit BOOL (0 = off, 1= on) [FS2000–FS2004 only]	**Ok**	**Ok**
3B7C	4	Engine 1 generator active, a 32-bit BOOL (0 = off, 1= on) [FS2000–FS2004 only]	**Ok**	**Ok**
3B98	4	Engine 1 fuel pump switch, a 32-bit BOOL (0 = off, 1= on) [FS2000–FS2004 only]	**Ok**	**Ok**
3BA0	8	The tailhook position, as a double floating point value (0.0=fully retracted, 1.0=fully lowered). [FS2002 and FS2004 only]	**Ok**	**Ok**
3BA8	40	Area used by PFC.DLL for axis input, for optional assignment and calibration in FSUIPC. When the PFC driver is not being used, other programs can make use of these offsets to input axis values directly to FSUIPC, which also can then be assigned in FSUIPC and thence calibrated. Note that by default FSUIPC assumes that the normal input here is in the range 0–127, and scales it accordingly. For applications supplying a greater range, possibly up to the maximum allowed for joysticks (–16383 to +16383) the "RAW" option will be needed to avoid this scaling. On the other hand, when the PFC driver is running, application programs or modules can access the raw PFC axis values at these offsets, which are assigned to the hardware as listed below. One 16-bit word is allowed for each (although the PFC axes have a maximum range of 0 to 127). The axes are: 3BA8 0 Aileron 3BAA 1 Elevator 3BAC 2 Rudder 3BAE 3 Quadrant axis 5 3BB0 4 Quadrant axis 3 3BB2 5 Quadrant axis 1 3BB4 6 Left toe brake 3BB6 7 Quadrant axis 6 3BB8 8 Quadrant axis 4 3BBA 9 Quadrant axis 2 3BBC 10 Right toe brake 3BBE 11 Elevator trim 3BC0 12 Aileron trim 3BC2 13 Rudder trim 3BC4 14 Steering tiller 3BC6 15 *not used* There are control flags (to disconnect these axes) at offset 3BC8. Each bit, 2^0 to 2^15 can be set to disconnect the equivalent numbered axis above.	**Ok**	**Ok**
3BD2	2	This is a 16-bit counter that is incremented each time a FLT file is saved in FS. This applies to flights saved through FS Flights menu, the shortcut key (;), AutoSave, and via the FSUIPC flight saving facilities. The filenames of the saved flights can be read at offset 0400, or (historically) by using the path reading facility at offset 0FF0 and following.	**Ok**	**Ok**
3BD6	18	Panel failure modes (FS2002 and FS2004 only): one byte flag/control for each of the following "partial panel" modes: 3BD6 ADF (both on FS2004)	**Ok**	**Ok**

		3BD7 ASI 3BD8 Altimeter 3BD9 Attitude Indicator 3BDA COMM1 3BDB COMM2 3BDC Compass 3BDD ? (unknown) 3BDE Engine (see 0B6B for separate engines) 3BDF Fuel Indicator 3BE0 Heading Indicator 3BE1 NAV1 3BE2 NAV2 3BE3 Pitot heat 3BE4 Transponder 3BE5 Turn Co-ordinator 3BE6 Vacuum 3BE7 VSI		
3BE8	8	Attitude Indicator failure timer, as a double floating point value.	**No**	**No**
3BF0	4	Attitude indicator lock indicator, 32-bit integer but probable only Boolean (0 or 1) [FS2k/CFS2 only]	**No**	**No**
3BF4	4	Low vacuum indicator, 32-bit integer but probable only Boolean (0 or 1) [FS2k/CFS2 only]	**No**	**No**
3BFA	2	Flaps détente increment. The full range of flap movement is 0–0x3FFF (16383). Each détente position or "notch" is spaced equally over this range, no matter what flap angle is represented—a table in the AIR file gives those. To obtain the number of détentes, divide this increment value into 16383 and add 1. For example 2047 (0x7FF) would be the increment for 9 positions as on the default FS2K 737.	**Ok**	**Ok**
3BFC	4	Zero Fuel Weight, lbs * 256. This is the aircraft weight plus the payload weight, minus fuel. In FS2004 this changes as the payload is adjusted.	**Ok**	**Ok**
3C00	256	Pathname of the current AIR file, excluding the FS main path (see 3E00), but including everything from "Aircraft\…" to the final "…air". This is zero padded to fill the 256 bytes available. When this changes the 16-bit counter at 32FC is incremented, so interested programs don't have to keep on reading the whole 256 bytes to check.	**Ok**	**Ok**
3D00	256	Name of the current aircraft (from the "title" parameter in the AIRCRAFT.CFG file). Valid for FS2K only.	**Ok**	**Ok**
3E00	256	Path of the Flight Simulator installation, down to and including the FS main folder and a following \ character. If the PC is on a Network and WideFS is in use then the full UNC (universal naming convention) path is given if possible. Examples are: D:\FS2000\ (non-Network) \\MyMainPC\drived\FS2000\ (Network, named PC and named shared drive)	**Ok**	**Ok**
3F00	2	To load or save a Flight (FS2000/2002/FS2004) or Situation (FS98) you first set up the pathname (and optional description) at offset 3F04 below, then write here. Write one of these values: 0 to simply load the specified flight/situation. 1 to save the flight/situation with an empty description 257 to save the flight/situation with a description as well This facility works on FS98 through to FS2004 but not CFS1 nor CFS2. Also note that excepting FS2004, for Loading you don't have to have the files in the "Pilots" or "Situatio" folder (or "flights" for FS2002)—any folder within the FS main folder can	**Ok**	**Ok**

		be used to load Flights/Situations. However, they can only be saved in "Pilots" (FS98/FS2000) or "Flights\MyFlts" (FS2002), and this folder is assumed by default. On FS2004 flights are saved in the "My Documents" FS folder. Flights are loaded by default from there too – you don't have to specify a path. If you are Loading a file, please allow time for the file to load before expecting any further meaningful response across the FSUIPC interface. FSUIPC will probably not be able to respond for several seconds even on the fastest machines.		
3F02	2	FLT/STN file loading counter (incremented by FSUIPC whenever the FLT or STN file, as defined at offset 3F04 changes or is reloaded). This word is read only—attempting to write here will do no harm.	**Ok**	**Ok**
3F04	252	**READ:** (FS2000/2002/2004 only) Pathname of the currently loaded FLT file, excluding the FS main path (see 3E00) where applicable, but including everything from "Pilots\…" or "Situatio\…" (or "Flights\" in FS2002), or whatever, to the final ".flt" or ".stn". This is zero padded to fill the 252 bytes available, or truncated if longer. If the path is outside FS's own path then it is given in UNC format if possible when WideFS is in use. When this changes (or simply reloaded) the 16-bit counter at 3F02 is incremented, so interested programs don't have to keep on reading the whole 252 bytes to check. **WRITE:** (FS2000, FS2002, FS2004 and FS98) Write the file name for the FLT+WX (FS2000/FS2002) or STN (FS98) file you wish to Load or Save. The name can include the final ".flt" or ".stn" but this will be discarded in any case. You can specify a folder (existing within FS's main folder) such as "Pilots\" or "Situatio\" ("Flights\MyFlts\" in FS2002) for Loading, but files can only be saved to "Pilots\" in FS98/2000, "Flights\MyFlts\" in FS2002 and in your documents folder in FS2004. If you give a path for saving, it is discarded. There must be a zero terminator. If you are writing the file, a description can also be specified, following the pathname and its zero terminator. Obviously this is limited by the space available. It must also be terminated by a zero byte, and indicated in the value written to 3F00 above. See 3F00 above for details of actually Loading or Saving the Flight or Situation so identified.	**Ok**	**Ok**
4000	512	*Reserved*		
4200	256	FSUIPC's sound playing interface: see earlier section		
4300	7424	*Reserved*		
6000	512	FS2004 GPS data area—only known offsets listed below:	**No**	**Yes**
6004	4	FS2004 GPS flags (bits numbered from least significant): 0 ? 1 Active Plan 2 Active Way point 3 Arrived 4 ? 5 Direct To	**No**	**Yes**

		6 ? 7 Active way point locked 8 Approach loaded 9 Approach Active		
6008	4	FS2004 GPS: *Unknown progress flags*	**No**	**Yes**
600C	4	FS2004 GPS: Zulu time in seconds since midnight	**No**	**Yes**
6010	8	FS2004 GPS: aircraft latitude, floating point double, in degrees (+ve = N, –ve = S).	**No**	**Yes**
6018	8	FS2004 GPS: aircraft longitude, floating point double, in degrees (+ve = E, –ve = W).	**No**	**Yes**
6020	8	FS2004 GPS: aircraft altitude, floating point double, in metres.	**No**	**Yes**
6028	8	FS2004 GPS: magnetic variation at aircraft, floating point double, in radians (add to magnetic for true, subtract from true for magnetic).	**No**	**Yes**
6030	8	FS2004 GPS: aircraft ground speed, floating point double, metres per second.	**No**	**Yes**
6038	8	FS2004 GPS: aircraft true heading, floating point double, in radians.	**No**	**Yes**
6040	8	FS2004 GPS: aircraft magnetic track, floating point double, in radians.	**No**	**Yes**
6048	8	FS2004 GPS: distance to next waypoint, floating point double, in metres.	**No**	**Yes**
6050	8	FS2004 GPS: magnetic bearing to next waypoint, floating point double, in radians.	**No**	**Yes**
6058	8	FS2004 GPS: cross track error, floating point double, in metres.	**No**	**Yes**
6060	8	FS2004 GPS: required true heading, floating point double, in radians.	**No**	**Yes**
6068	8	FS2004 GPS: track error, floating point double, in radians.		
6078	8	FS2004 GPS: aircraft vertical speed (*Needs checking*)	**No**	
6080	1	FS2004 GPS: previous waypoint valid flag (=0 if not valid)	**No**	**Yes**
6081	6?	FS2004 GPS: string ID of previous way point, zero terminated	**No**	**Yes**
608C	8	FS2004 GPS: previous waypoint latitude, floating point double, in degrees (+ve = N, –ve = S).	**No**	**Yes**
6094	8	FS2004 GPS: previous waypoint longitude, floating point double, in degrees (+ve = E, –ve = W).	**No**	**Yes**
609C	8	FS2004 GPS: previous waypoint aircraft altitude, floating point double, in metres.	**No**	**Yes**
60A4	6?	FS2004 GPS: string ID of next waypoint, zero terminated	**No**	**Yes**
60AC	8	FS2004 GPS: next way point latitude, floating point double, in degrees (+ve = N, –ve = S).	**No**	**Yes**
60B4	8	FS2004 GPS: next waypoint longitude, floating point double, in degrees (+ve = E, –ve = W).	**No**	**Yes**
60BC	8	FS2004 GPS: next waypoint aircraft altitude, floating point double, in metres.	**No**	**Yes**
60E4	4	FS2004 GPS: Next waypoint ETE as 32-bit integer, in seconds	**No**	**Yes**
60E8	4	FS2004 GPS: Next waypoint ETA as 32-bit integer in seconds, local time	**No**	**Yes**
60EC	8	FS2004 GPS: Distance to next waypoint, floating point double, in metres	**No**	**Yes**
60F4	8	FS2004 GPS: Distance between previous and next waypoints, floating point double, in metres	**No**	**Yes**
60FC	4	FS2004 GPS: Approach mode, as 32-bit integer (*needs checking*)	**No**	
6100	4	FS2004 GPS: Approach way point type, as 32-bit integer (*needs checking*)	**No**	
6104	4	FS2004 GPS: Approach segment type, as 32-bit integer (*needs checking*)	**No**	
6108	1	FS2004 GPS: Approach mode, flag indicating approach waypoint is the runway (*needs checking*)	**No**	
610C	8	FS2004 GPS: Course to set (CTS), floating point double, in	**No**	**Yes**

		radians		
6120	4	FS2004 GPS: Flight Plan, total number of waypoints, as 32-bit integer	**No**	**Yes**
6128	4	FS2004 GPS: Approach way point count, as 32-bit integer (*needs checking*)	**No**	
6137	5	FS2004 GPS: Flight plan destination airport ID *(This appears to have been optimistic. I can't find the destination ID)*	**No**	**No?**
613C	4	FS2004 GPS: Approach way point index, as 32-bit integer (*needs checking*)	**No**	
6140	8	FS2004 GPS: Approach name	**No**	**Yes**
6150	4	FS2004 GPS: Approach transition index, as 32-bit integer (*needs checking*). –1 means not valid.	**No**	
6154	8	FS2004 GPS: Approach transition name	**No**	**Yes**
615C	1	FS2004 GPS: Approach is missed flag (*needs checking*)	**No**	
6160	4	FS2004 GPS: Approach type (*needs checking*)	**No**	
6168	4	FS2004 GPS: Approach time zone deviation, as 32-bit integer (*needs checking*)	**No**	
616C	4	FS2004 GPS: Current way point index, starting at 1, as 32-bit integer	**No**	**Yes**
6170	4	FS2004 GPS: Approach current way point index, as 32-bit integer (*needs checking*)	**No**	
6178	8	FS2004 GPS: *Unknown double floating point value*	**No**	**Yes**
6180	8	FS2004 GPS: *Unknown double floating point value*	**No**	**Yes**
6188	8	FS2004 GPS: *Unknown double floating point value*	**No**	**Yes**
6190	4	FS2004 GPS: Time last waypoint was crossed, seconds since Zulu midnight	**No**	**Yes**
6198	4	FS2004 GPS: Destination ETE as 32-bit integer, in seconds	**No**	**Yes**
619C	4	FS2004 GPS: Destination ETA as 32-bit integer, in seconds, local time	**No**	**Yes**
61A0	8	FS2004 GPS: Route total distance, double floating point, in metres	**No**	**Yes**
61A8	8	FS2004 GPS: Estimated fuel burn, double floating point, in gallons	**No**	**Yes**
61B0	4	FS2004 GPS: Time of last update to 61B8 (seconds since Zulu midnight)	**No**	**Yes**
61B8	4	FS2004 GPS: Count updated every 5 seconds. (*purpose as yet unknown*)	**No**	**Yes**
6200	1216	*Reserved*		
66C0	64	**Free for general use**, for example in button or keys programming.		
6700	1632	*Reserved*		
6D60	32	FSUIPC message window title—up to 32 characters including a zero terminator. (FS2004 only) The message window title can be set by the program using it, but as only one such Window is supported only one title is available. The first program writing it *and then* a multiline message wins! This only needs doing once, immediately before any multiline messages are sent to 3380.	**No**	**Yes**
6D80	1408	*Reserved*		
7300	112	*Available for applications:* ***apply for allocations to Pete Dowson***		
7370	1504	*Reserved*		
7840	144	*Available for applications:* ***apply for allocations to Pete Dowson***		
78D0	1840	*Reserved*		
8000	768	*Reserved for FSUIPC and WideFS internals*		
8300	256	Area in FS2002 and FS2004 reporting and controlling assorted views. Details of those values known follow. This information has been supplied by Matthias Neusinger.		

8320	1	Byte value, the view mode in the currently selected window (read/write): FS2004: 1=cockpit, 2=virtual cockpit, 3=tower, 4=spot plane, 5=top down FS2002: 0=cockpit, 1=virtual cockpit, 2=tower, 4=spot plane, 7=top down	**Ok**	**Ok**
832C	2	Zoom setting for selected window in cockpit mode (64 = 1x), read/write	**Ok**	**Ok**
832E	2	Zoom setting for selected window in virtual cockpit mode (64 = 1x), read/write	**Ok**	**Ok**
8330	2	Zoom setting for selected window in tower mode (64 = 1x), read/write	**Ok**	**Ok**
8334	2	Zoom setting for selected window in spot plane mode (64 = 1x), read/write	**Ok**	**Ok**
8336	2	Zoom setting for selected window in top down mode (64 = 1x), read/write	**Ok**	**Ok**
833C	2	Relative direction of spot plane from user aircraft, read/write (in degrees in usual 360 = 65536 format).	**Ok**	**Ok**
8340	4	Distance of spot plane from user aircraft, read/write (in metres * 256).	**Ok**	**Ok**
8345	1	Spot plane transition: gradual is 0, instant if 1. (read/write)	**Ok**	**Ok**
8348	4	Relative altitude of spot plane from user aircraft, read/write (in metres * 256).	**Ok**	**Ok**
83BC	24	View point latitude/longitude/altitude, exactly as at offset 05B0. Read only, FS2004 only.	**No**	**Ok**
83D4	12	View point pitch, bank and heading, in same format as that for the user's aircraft at offset 0578. Read only, FS2004 only.	**No**	**Ok**
8600	232	*Available for applications:* ***apply for allocations to Pete Dowson***		
86E8	2328	*Reserved*		
9000	2048	*Reserved for future improvements*		
9800	1024	*Used by Wideclient's Lua display control*		
9C00	9216	*Reserved*		
C000	4096	FS2004 New Weather Interface areas, allowing both local and global weather data to be read and written. (details of the NWI are provided separately in the SDK)	**No**	**Ok**
D000	2048	FS2004 A.I. ground aircraft additional traffic data (see section on AI Traffic earlier)	**No**	**Ok**
D800	2048	FS2004 A.I. airborne aircraft additional traffic data (see section on AI Traffic earlier)	**No**	**Ok**
E000	4096	FS2002/4 A.I. ground aircraft traffic data (see section on AI Traffic earlier)	**Ok**	**Ok**
F000	4096	FS2002/4 A.I. airborne aircraft traffic data (see section on AI Traffic earlier)	**Ok**	**Ok**

* The FS2004 column in the above table is a work-in-progress. In particular, those entries left blank are currently "don't knows", awaiting checking.

NOTE on aircraft dynamic values: (thanks to Ian Donohoe)

The aircraft linear velocity and acceleration values are, of course, related to specific references. In the cases of the values given these are the "body axes", and the "world axes". This can become more confusing because of the different ways of naming the axes. In the FS2000 .FLT files (the [SimVars] section), and in the table above, the names are different to those generally used in engineering and mathematics, as follows:

Description	FS notation	Engineering
Lateral, left–right	X	Y
Vertical, up-down	Y	Z

Longitudinal, forward–backward	Z	X
Pitch	P	Q
Roll, or bank	B	P
Yaw, or heading	H	R

Here are more specific definitions of the sets of linear reference axes themselves:

World frame of reference:

X-axis (Z-axis in FS) = True North-South
Y-axis (X-axis in FS) = True East-West
Z-axis (Y-axis in FS) = True Vertical

Body Frame Of Reference:

X-axis (Z in FS) = longitudinal through CG
Y-axis (X in FS) = lateral through CG
Z-axis (Y in FS) = vertical (in body terms) through CG

There's a complication with the body frame in deciding the longitudinal centreline from which the other axes are offset by 90 degrees. This is generally taken to be the zero lift line (i.e. alpha at which there is zero lift).

Note that some of the values obtained from FS2002 may not abide by exactly the same rules—but this is noted against the specific values in the Table. Clarification will be added as more details are discovered.

Table of additional PANELS variables for FS2000

Please refer to the Microsoft FS Panels SDK for more details of both the token variable names and the meanings of the assorted "type" names. In particular the file "gauges.h" contains these details. (Similar names, but more of them, apply to FS2002 and FS2004).

These variables can be read or written via FSUIPC since version 1.94, or through WideFS from version 3.96, but none of them are guaranteed, and they **may** not be carried over into future versions of FS or FSUIPC.

A program called "FSLOOK2.EXE" is supplied with this document which can be run with FSUIPC or WideFS and which will display all the tokenised variables listed (but not the extras added by FSUIPC, without token names). Click "file" and select "AutoRefresh" to see the values updated as you use FS.

With a few exceptions, as noted, access to these variables under FS98 will simply obtain zeroes on reading, and writes will be discarded. FSUIPC will not crash. But take care if you are using FS6IPC on FS98. These offsets will refer to completely different things in FS98 with FS6IPC, or will likely crash FS98.

The table is organised in order of the offsets assigned. Most variables are 4 or 8 bytes in length. Addresses in the 2000–3FFF range which are not implied by this list may read/write useful data—check the table above for uses of some addresses in this range by FSUIPC's mapping—but if so it is not identified here. Because of the way the mapping works, some such accesses may merely obtain one of the other listed values.

Offset	Token Name	Token Id	Type	FS2002	FS2004
2054	TURB_ENGINE_1_TANK_SELECTOR	635	SINT32	Yes	Use 3880
2058	TURB_ENGINE_1_TANKS_USED	636	SINT32	Yes	Use 3888
2068	TURB_ENGINE_1_FUEL_AVAILABLE	639	BOOL	Yes	Use 3894
2074	TURB_ENGINE_1_PCT_AREA	640	FLOAT64	Yes	
2084	TURB_ENGINE_1_VIBRATION	642	FLOAT64	Yes	
2154	TURB_ENGINE_2_TANK_SELECTOR	654	SINT32	Yes	Use 37C0
2158	TURB_ENGINE_2_TANKS_USED	655	SINT32	Yes	Use 37C8
2168	TURB_ENGINE_2_FUEL_AVAILABLE	658	BOOL	Yes	Use 37D4
2174	TURB_ENGINE_2_PCT_AREA	659	FLOAT64	Yes	
2184	TURB_ENGINE_2_VIBRATION	661	FLOAT64	Yes	
2254	TURB_ENGINE_3_TANK_SELECTOR	673	SINT32	Yes	Use 3700
2258	TURB_ENGINE_3_TANKS_USED	674	SINT32	Yes	Use 3708
2268	TURB_ENGINE_3_FUEL_AVAILABLE	677	BOOL	Yes	Use 3714
2274	TURB_ENGINE_3_PCT_AREA	678	FLOAT64	Yes	
2284	TURB_ENGINE_3_VIBRATION	680	FLOAT64	Yes	

2354	TURB_ENGINE_4_TANK_SELECTOR	692	SINT32	Yes	Use 3640
2358	TURB_ENGINE_4_TANKS_USED	693	SINT32	Yes	Use 3648
2368	TURB_ENGINE_4_FUEL_AVAILABLE	696	BOOL	Yes	Use 3654
2374	TURB_ENGINE_4_PCT_AREA	697	FLOAT64	Yes	
2384	TURB_ENGINE_4_VIBRATION	699	FLOAT64	Yes	
2420	PROPELLER_1_FEATHERING_INHIBIT	704	BOOL	Yes	
2424	PROPELLER_1_FEATHERED	705	BOOL	Yes	
2428	PROPELLER_1_SYNC_DELTA_LEVER	706	FLOAT64	Yes	
2430	PROPELLER_1_AUTOFEATHER_ARMED	707	BOOL	Yes	
2520	PROPELLER_2_FEATHERING_INHIBIT	712	BOOL	Yes	
2524	PROPELLER_2_FEATHERED	713	BOOL	Yes	
2528	PROPELLER_2_SYNC_DELTA_LEVER	714	FLOAT64	Yes	
2530	PROPELLER_2_AUTOFEATHER_ARMED	715	BOOL	Yes	
2620	PROPELLER_3_FEATHERING_INHIBIT	720	BOOL	Yes	
2624	PROPELLER_3_FEATHERED	721	BOOL	Yes	
2628	PROPELLER_3_SYNC_DELTA_LEVER	722	FLOAT64	Yes	
2630	PROPELLER_3_AUTOFEATHER_ARMED	723	BOOL	Yes	
2720	PROPELLER_4_FEATHERING_INHIBIT	728	BOOL	Yes	
2724	PROPELLER_4_FEATHERED	729	BOOL	Yes	
2728	PROPELLER_4_SYNC_DELTA_LEVER	730	FLOAT64	Yes	
2730	PROPELLER_4_AUTOFEATHER_ARMED	731	BOOL	Yes	
2824	TOTAL_LOAD_AMPS	750	FLOAT64	Yes	
282C	BATTERY_LOAD	751	FLOAT64	Yes	
2834	BATTERY_VOLTAGE	752	FLOAT64	Yes	
2840	MAIN_BUS_VOLTAGE	753	FLOAT64	Yes	
2848	MAIN_BUS_AMPS	754	FLOAT64	Yes	
2850	AVIONICS_BUS_VOLTAGE	755	FLOAT64	Yes	
2858	AVIONICS_BUS_AMPS	756	FLOAT64	Yes	
2860	HOT_BATTERY_BUS_VOLTAGE	757	FLOAT64	Yes	
2868	HOT_BATTERY_BUS_AMPS	758	FLOAT64	Yes	
2870	BATTERY_BUS_VOLTAGE	759	FLOAT64	Yes	
2878	BATTERY_BUS_AMPS	760	FLOAT64	Yes	
2880	GENERATOR_ALTERNATOR_1_BUS_VOLTAGE	761	FLOAT64	Yes	
2888	GENERATOR_ALTERNATOR_1_BUS_AMPS	762	FLOAT64	Yes	
2890	GENERATOR_ALTERNATOR_2_BUS_VOLTAGE	763	FLOAT64	Yes	
2898	GENERATOR_ALTERNATOR_2_BUS_AMPS	764	FLOAT64	Yes	
28A0	GENERATOR_ALTERNATOR_3_BUS_VOLTAGE	765	FLOAT64	Yes	
28A8	GENERATOR_ALTERNATOR_3_BUS_AMPS	766	FLOAT64	Yes	
28B0	GENERATOR_ALTERNATOR_4_BUS_VOLTAGE	767	FLOAT64	Yes	
28B8	GENERATOR_ALTERNATOR_4_BUS_AMPS	768	FLOAT64	Yes	
2A00	ELEVON_1_DEFLECTION	809	FLOAT64	Yes	
2A08	ELEVON_2_DEFLECTION	810	FLOAT64	Yes	
2A10	ELEVON_3_DEFLECTION	811	FLOAT64	Yes	
2A18	ELEVON_4_DEFLECTION	812	FLOAT64	Yes	
2A20	ELEVON_5_DEFLECTION	813	FLOAT64	Yes	
2A28	ELEVON_6_DEFLECTION	814	FLOAT64	Yes	
2A30	ELEVON_7_DEFLECTION	815	FLOAT64	Yes	
2A38	ELEVON_8_DEFLECTION	816	FLOAT64	Yes	
2B08	HYDRAULICS1_PRESSURE_PSF	732	FLOAT64	Yes	
2B1C	HYDRAULICS1_RESERVOIR_PCT	733	FLOAT64	Yes	
2C08	HYDRAULICS2_PRESSURE_PSF	734	FLOAT64	Yes	
2C1C	HYDRAULICS2_RESERVOIR_PCT	735	FLOAT64	Yes	
2D08	HYDRAULICS3_PRESSURE_PSF	736	FLOAT64	Yes	
2D1C	HYDRAULICS3_RESERVOIR_PCT	737	FLOAT64	Yes	
2E08	HYDRAULICS4_PRESSURE_PSF	738	FLOAT64	Yes	
2E1C	HYDRAULICS4_RESERVOIR_PCT	739	FLOAT64	Yes	
2E90	STANDBY_VACUUM_CIRCUIT_ON	778	BOOL	No	

2F00	CG_AFT_LIMIT	796	FLOAT64	No	
2F08	CG_FWD_LIMIT	797	FLOAT64	No	
2F10	CG_MAX_MACH	798	FLOAT64	Yes	
2F18	CG_MIN_MACH	799	FLOAT64	Yes	
2F20	CONCORDE_VISOR_NOSE_HANDLE	805	SINT32	Yes	
2F28	CONCORDE_VISOR_POS_PCT	806	FLOAT64	Yes	
2F30	CONCORDE_NOSE_ANGLE	807	FLOAT64	Yes	
2F38	GEAR_POS_TAIL	808	FLOAT64	No	
2F40	AUTOPILOT_MAX_SPEED	820	FLOAT64	Yes	
2F48	AUTOPILOT_CRUISE_SPEED	821	FLOAT64	Yes	
2F50	BARBER_POLE_MACH	822	FLOAT64	Yes	
2F58	SELECTED_FUEL_TRANSFER_MODE	823	SINT32	Yes	
2F60	HYDRAULIC_SYSTEM_INTEGRITY	824	FLOAT64	Yes	
2F68	ATTITUDE_CAGE_BUTTON	825	BOOL	Yes	
3420	RAD_INS_SWITCH	613	BOOL32	Yes	
3424	LOW_HEIGHT_WARNING	616	BOOL32	No	
3428	DECISION_HEIGHT	615	FLOAT64	Yes	
3438	ENGINE_1_FUELFLOW_BUG_POSITION	801	FLOAT64	Yes	
3440	ENGINE_2_FUELFLOW_BUG_POSITION	802	FLOAT64	Yes	
3448	ENGINE_3_FUELFLOW_BUG_POSITION	803	FLOAT64	Yes	
3450	ENGINE_4_FUELFLOW_BUG_POSITION	804	FLOAT64	Yes	
3458	PANEL_AUTOPILOT_SPEED_SETTING	817	FLOAT64	Yes	
3460	AUTOPILOT_AIRSPEED_HOLD_CURRENT	819	BOOL	No	
34D0	G_FORCE_MAXIMUM	605	FLOAT64	No	
34D8	G_FORCE_MINIMUM	606	FLOAT64	No	
34E8	ENGINE1_MAX_RPM	608	UINT32	No	
34EC	ENGINE2_MAX_RPM	609	UINT32	No	
34F0	ENGINE3_MAX_RPM	610	UINT32	No	
34F4	ENGINE4_MAX_RPM	611	UINT32	No	
3550	ENGINE4_THROTTLE_LEVER_POS	233	SINT16	No	
3552	ENGINE4_PROPELLER_LEVER_POS	234	UINT16	No	
3554	ENGINE4_MIXTURE_LEVER_POS	235	UINT16	No	
3556	ENGINE4_STARTER_SWITCH_POS	237	ENUM16	No	
3558	ENGINE4_MAGNETO_LEFT	238	BOOL16	No	
355A	ENGINE4_MAGNETO_RIGHT	239	BOOL16	No	
3560	ENGINE3_THROTTLE_LEVER_POS	198	SINT16	No	
3562	ENGINE3_PROPELLER_LEVER_POS	199	UINT16	No	
3564	ENGINE3_MIXTURE_LEVER_POS	200	UINT16	No	
3566	ENGINE3_STARTER_SWITCH_POS	202	ENUM16	No	
3568	ENGINE3_MAGNETO_LEFT	203	BOOL16	No	
356A	ENGINE3_MAGNETO_RIGHT	204	BOOL16	No	
3570	ENGINE2_THROTTLE_LEVER_POS	163	SINT16	No	
3572	ENGINE2_PROPELLER_LEVER_POS	164	UINT16	No	
3574	ENGINE2_MIXTURE_LEVER_POS	165	UINT16	No	
3576	ENGINE2_STARTER_SWITCH_POS	167	ENUM16	No	
3578	ENGINE2_MAGNETO_LEFT	168	BOOL16	No	
357A	ENGINE2_MAGNETO_RIGHT	169	BOOL16	No	
3580	ENGINE1_THROTTLE_LEVER_POS	128	SINT16	No	
3582	ENGINE1_PROPELLER_LEVER_POS	129	UINT16	No	
3584	ENGINE1_MIXTURE_LEVER_POS	130	UINT16	No	
3586	ENGINE1_STARTER_SWITCH_POS	132	ENUM16	No	
3588	ENGINE1_MAGNETO_LEFT	133	BOOL16	No	
358A	ENGINE1_MAGNETO_RIGHT	134	BOOL16	No	
35B8	RECIP_ENGINE4_CARB_HEAT_POS	513	FLOAT64	No	
35C0	RECIP_ENGINE4_ALTERNATE_AIR_POS	514	FLOAT64	Yes	
35C8	RECIP_ENGINE4_COOLANT_RESERVOIR_PCT	515	FLOAT64	Yes	
35F0	RECIP_ENGINE4_STARTER_TORQUE	522	FLOAT64	Yes	

35F8	RECIP_ENGINE4_TURBOCHARGER_FAILED	524	BOOL	Yes	
3610	RECIP_ENGINE4_TIT_DEGR rankine	531	FLOAT64	Yes	
3618	RECIP_ENGINE4_CHT_DEGR	532	FLOAT64	Yes	
3644	RECIP_ENGINE4_TANKS_USED	540	FLAGS	Yes	
3678	RECIP_ENGINE3_CARB_HEAT_POS	474	FLOAT64	No	
3680	RECIP_ENGINE3_ALTERNATE_AIR_POS	475	FLOAT64	Yes	
3688	RECIP_ENGINE3_COOLANT_RESERVOIR_PCT	476	FLOAT64	Yes	
36B0	RECIP_ENGINE3_STARTER_TORQUE	483	FLOAT64	Yes	
36B8	RECIP_ENGINE3_TURBOCHARGER_FAILED	485	BOOL	Yes	
36D0	RECIP_ENGINE3_TIT_DEGR rankine	492	FLOAT64	Yes	
36D8	RECIP_ENGINE3_CHT_DEGR	493	FLOAT64	Yes	
3704	RECIP_ENGINE3_TANKS_USED	501	FLAGS	Yes	
3738	RECIP_ENGINE2_CARB_HEAT_POS	435	FLOAT64	No	
3740	RECIP_ENGINE2_ALTERNATE_AIR_POS	436	FLOAT64	Yes	
3748	RECIP_ENGINE2_COOLANT_RESERVOIR_PCT	437	FLOAT64	Yes	
3770	RECIP_ENGINE2_STARTER_TORQUE	444	FLOAT64	Yes	
3778	RECIP_ENGINE2_TURBOCHARGER_FAILED	446	BOOL	Yes	
3790	RECIP_ENGINE2_TIT_DEGR rankine	453	FLOAT64	Yes	
3798	RECIP_ENGINE2_CHT_DEGR	454	FLOAT64	Yes	
37C4	RECIP_ENGINE2_TANKS_USED	462	FLAGS	Yes	
37F8	RECIP_ENGINE1_CARB_HEAT_POS	396	FLOAT64	No	
3800	RECIP_ENGINE1_ALTERNATE_AIR_POS	397	FLOAT64	Yes	
3808	RECIP_ENGINE1_COOLANT_RESERVOIR_PCT	398	FLOAT64	Yes	
3830	RECIP_ENGINE1_STARTER_TORQUE	405	FLOAT64	Yes	
3838	RECIP_ENGINE1_TURBOCHARGER_FAILED	407	BOOL	Yes	
3850	RECIP_ENGINE1_TIT_DEGR rankine	414	FLOAT64	Yes	
3858	RECIP_ENGINE1_CHT_DEGR	415	FLOAT64	Yes	
3870	ENGINE_PRIMER	361	FLOAT64	Yes	
3884	RECIP_ENGINE1_TANKS_USED	423	FLAGS	Yes	
38A0	GENERAL_ENGINE4_FAILURE	594	BOOL	Yes	
38A4	RECIP_ENGINE4_COMBUSTION	523	BOOL	Yes	
38C0	RECIP_ENGINE4_STARTER	518	BOOL	Yes	
38C0	GENERAL_ENGINE4_STARTER	593	FLOAT64	Yes	
3928	RECIP_ENGINE4_OIL_LEAK_PCT	536	FLOAT64	Yes	
3940	RECIP_ENGINE4_DAMAGE_PERCENT	545	FLOAT64	Yes	
3948	RECIP_ENGINE4_COMBUSTION_SOUND_PCT	543	FLOAT64	Yes	
3960	GENERAL_ENGINE3_FAILURE	584	BOOL	Yes	
3964	RECIP_ENGINE3_COMBUSTION	484	BOOL	Yes	
3980	RECIP_ENGINE3_STARTER	479	BOOL	Yes	
3980	GENERAL_ENGINE3_STARTER	583	FLOAT64	Yes	
39E8	RECIP_ENGINE3_OIL_LEAK_PCT	497	FLOAT64	Yes	
3A00	RECIP_ENGINE3_DAMAGE_PERCENT	506	FLOAT64	Yes	
3A08	RECIP_ENGINE3_COMBUSTION_SOUND_PCT	504	FLOAT64	Yes	
3A20	GENERAL_ENGINE2_FAILURE	574	BOOL	Yes	
3A24	RECIP_ENGINE2_COMBUSTION	445	BOOL	Yes	
3A40	RECIP_ENGINE2_STARTER	440	BOOL	Yes	
3A40	GENERAL_ENGINE2_STARTER	573	FLOAT64	Yes	
3AA8	RECIP_ENGINE2_OIL_LEAK_PCT	458	FLOAT64	Yes	
3AC0	RECIP_ENGINE2_DAMAGE_PERCENT	467	FLOAT64	Yes	
3AC8	RECIP_ENGINE2_COMBUSTION_SOUND_PCT	465	FLOAT64	Yes	
3AE0	GENERAL_ENGINE1_FAILURE	564	BOOL	Yes	
3AE4	RECIP_ENGINE1_COMBUSTION	406	BOOL	Yes	
3B00	RECIP_ENGINE1_STARTER	401	BOOL	Yes	
3B00	GENERAL_ENGINE1_STARTER	563	FLOAT64	Yes	
3B68	RECIP_ENGINE1_OIL_LEAK_PCT	419	FLOAT64	Yes	
3B80	RECIP_ENGINE1_DAMAGE_PERCENT	428	FLOAT64	Yes	
3B88	RECIP_ENGINE1_COMBUSTION_SOUND_PCT	426	FLOAT64	Yes	

Final Note for Developers

It is intended that FSUIPC.DLL be maintained and improved to cover more aspects of FS interfacing, as they are found to be needed by applications. A completely generic interface to several of FS's internal facilities and tables has been attempted, but was withdrawn because of the possibilities this extended for really ruining what little performance most folks are getting with FS in the first place. The data available and control possible is not all neatly concentrated in the one globals area as was the case with FS98, and it is even more distributed and complex to access in FS2002 and certainly FS2004.

I have therefore decided to add support for specific needs as they arise, rather than to provide anything generic but potentially inefficient. Application Programmers needing specific items of information, or specific controls into FS2004, should contact me by email in the first instance, with full details. However, it should be noted that FS2004 is now fairly long in the tooth: if items have not already been found and identified by now then it is most unlikely that anything can be done until the next version of FS, where a renewed effort would be needed in any case.

Published by Peter L. Dowson, 11th April 2011