[Title]PFChid.DLL and PFChid64.DLL: Precision Flight Controls USB HID device

Interfacing modules for Microsoft Flight Simulator and Lockheed-Martin Prepar3D

(Use PFChid.dll for FS and P3D1-3, PFChid64.dll for P3D4 (64-bit). Freeware by Pete Dowson, © July 2020 (January 2023 for PFCHid64.dll)

Support Forum:

http://forum.simflight.com/forum/30-fsuipc-support-pete-dowson-modules/

PFC is Precision Flight Controls, Inc,
makers of quality flight simulation controls and displays.

(see http://www.flypfc.com).

***Version 1.49, PFChid.DLL** (for FS9, FSX and P3D 1-3)*
Version 5.143, PFChid64.DLL (for P3D versions 4/5 and MSFS)

PFChid.dll also needs FSUIPC Version 3.90 or later for FS9 or 4.70 or later for FSX and P3D but it will work better with versions 3.999 or 4.80 or later.

PFChid64.dll also needs FSUIPC Version 6 fir P3D 4 or 5, or FSUIPC version 7 for MSFS (aka FS2020). Version 5.143 works well with recent Cirrus 2 consoles and radio stacks.

This package contains the following parts:

PFChid.dll	The PFChid module for 32-bit sims
PFChid64.dll	The PFChid module for 64-bit sims
PFChid.DLL User Guide.doc	This document (in Adobe Acrobat format)
PFCmacroIndex.csv	A basic list of macro names, for re-programming control via FSUIPC

Introduction: what is PFChid.DLL?

This program is a Microsoft Flight Simulator driver for the USB-connected equipment produced by PFC Inc. It is designed to handle all known PFC USB "HID" devices, but has only so far been fully tested on these:

- Cirrus II Professional Console
- Cirrus II Professional Avionics

PFChid is a module for FS and goes into the FS Modules folder. It needs FSUIPC installed beforehand. The FSUIPC package is *not* included here. If you wish to extend or change any of the switch and dial functions in your PFC HID device you need a registered FSUIPC installation—the re-programming is performed using FSUIPC's macro file system.

Installation

Install FSUIPC first according to its own instructions. You need FSUIPC3 (at least version 3.90) for FS9, and FSUIPC4 (at least 4.52) for FSX and ESP. If you are planning to customise your use of any of the facilities on your PFC HID device you will need to purchase a Key for FSUIPC and register it. So doing will actually also allow functions on other PFC HID devices to be utilised even though direct support is not yet provided in this driver.

For FSX or P3D1-3, after installing FSUIPC correctly, simply copy the PFChid.DLL file into your flight simulator Modules folder. For P3D4/5 and MSFS2020 you need FSUIPC6 (P3D) or FSUIPC7 (MSFS). For these put the DLL into the FSUIPC folder, the one you chose during FSUIPC installation

That's it!

Setting PFChid options

There are very few options in PFChid. The driver is intended to be used as supplied, with no changes or configuration needed by the user in the module itself.

If the device you are using PFChid for includes a PFC throttle quadrant, the one with up to 6 levers and usually with different optional arrangements, then you will need a Registered version of FSUIPC in order to assign these levers to their desired functions and calibrate them for FS use. The best way in FSUIPC is to use its "Profiles" feature, setting up one profile for each configuration you may want. Then different aircraft can be assigned to different profiles, and FSUIPC will automatically take care of the re-assignments of levers as different aircraft are loaded.

The few options which do exist in PFChid itself are controlled by parameter settings in a file called PFChid.INI. This is not supplied with PFChid, but it is generated the first time you run FS with PFChid correctly installed. It sits next to PFChid in the FS Modules folder.

None of the options are accessible whilst running FS—they are only controlled via the INI file. However, that file is re-read each time you load a new aircraft (because it can contain aircraft-specific options), so you can change things in the file then load a different aircraft to get them recognised.

There is a menu entry in FS for PFChid (in the Modules or Add-ons menu), but this is merely to confirm that the driver is correctly loaded, and to provide Version information.

The default INI file looks like this (section by section):

[Options]

```
FlashAPcapts=Yes
FlashMarkers=Yes
FlashTransponder=Yes
FlashADFindicator=Yes
DMEuseRMIselect=Yes
```

The four "Flash..." options are 'Yes' or 'No' to determine whether certain indications should flash or not. The Bendix-King avionics being emulated does have flashing indicators, but if these irritate (or worse) they can be inhibited here. "APcapts" refers to the AutoPilot mode indicators which flash during capture modes. The markers lights (OMI) flash as different frequencies when triggered. The transponder flashes when responding to pings, and the ADF mode indicator flashes when it is receiving okay.

The "DMEuseRMIselect" option determines whether the DME's little "Off/N2/N1" slide switch selects the source (NAV1 or NAV2) for the Distance/Speed/Time displays, or (as defaulted) the RMI N1/N2 toggle. I find the latter more convenient, so it is defaulted. There is no FS control for the RMI source is any case—where an RMI is fitted it provides both NAV1 and NAV2 needles in any case (and ADF alternative selections).

Even with the RMI Toggle used for selection, the Off/N2/N1 slider still operates as an Off/On switch. It is just that the N1/N2 positions are not differentiated.

Note that FS does not provide a different frequency setting for DME, so the frequency display and adjuster knobs on the DME unit operate on the "In Use" NAV1 or NAV2 frequencies directly, according to the selector.

[Debug]

```
Console=No
LogComms=No
LogData=No
LogDecode=No
LogDevices=Yes
LogDeviceChanges=Yes
LogToDebugger=No
```

```
LogIPCwrites=No
LogMacroNames=No
LogTxData=No
LogReadCounts=No
```

This section controls the amount of Logging which is provided when you are trying to work out what is happening or wish to debug some changes or Macros.

Console=Yes

Shows the log in real time in a console window. For this to be useful you need to be running FS in Windowed mode too.

LogComms=Yes

Shows all data received and sent. I can't recommend this option except when you are desperately trying to nail a low-level problem, because it produces huge amounts of data from the vast numbers of Hid Report 1's arriving—reports normally filtered off because they are handled by the regular Windows joystick drivers.

LogDecode=Yes

Shows all handled HID reports in decoded form.

LogDevices=Yes (defaulted)

Logs the connection, discovery and disconnection of all HID devices, known or not.

LogDeviceChanges=Yes (defaulted)

Logs in more detail the connection and disconnection of supported (or known) devices.

LogToDebugger=Yes

Copies log messages to the standard "debug string", which can be viewed in programs like DebugView or any regular Windows debuggers. This can be useful for real-time debugging from a separate, even remote, computer.

LogIPCwrites=No

Shows writes to FSUIPC, results from using controls and dials on the connected units. Only Analog inputs are excluded from this facility as they would be too numerous.

LogMacroNames=No

Shows the Macro name which would result from using the operated switch to dial with a macro override (see notes on Macros later).

LogTxData=No

Logs sent commands (i.e. HID report 3's) in decoded form.

LogReadCounts=No

Logs the actual number of bytes received, and the rate, at approximately one second intervals.

[Config]

Note that you can have one "generic" [Config] section, and any number of [Config.<aircraft name>] sections which are applicable to loaded aircraft with a matching name. The name part of the section title can be a match for any part of the aircraft 'title' (as recorded in the aircraft.CFG file),

```
TrimRange=256
ApBeepWave=sound\Caapdis
AlertWave=sound\AltAlert
MacroFilename=PFC
```

TrimRange=256

Sets the range of the trim wheel movement, where the hardware allows this to be changed. A smaller value here will give faster adjustment when turning the wheel (or using an electric trim switch which operates by turning the wheel). A larger value will make it less sensitive and require more turns for the same trim adjustment.

ApBeepWave=sound\Caapdis

The sound that is to be played when the Autopilot is disconnected and in the second period of the A/P Test. Note that the A/P cannot be engaged until the Test has been run.

The default sound is one supplied in FS.

AlertWave=sound\AltAlert

The sound that is to be played for the Altitude/VS Pre-select Alert phase. No default sound is provided, so please select or make your own.

MacroFilename=PFC

The filename part of the macro file to be used to replace/override built-in switch and dial operations. The default is "PFC.mcro", but aircraft-specific macro files would normally be used. The use of macros to change the way the driver operates is described later.

Macros: configuring a PFC HID device for FS add-ons

This driver module interfaces to and deals with FS9, FSX and ESP with its own, default, cockpit controls and facilities. It does *not* directly support the additional or different controls and features provided by add-ons such as the PMDG range of aircraft, or the LevelD ones. Nor does it directly support Project Magenta.

In general, such add-ons represent airliners which, in any case, do not suit the devices currently being handled by the driver.

However, every switch and encoder (but none of the analogue inputs) can be diverted, individually, to operate via FSUIPC "macros". This needs a registered install of FSUIPC, and a Macro file.

FSUIPC Macros will not be documented here. Please refer to the FSUIPC documentation for that. What I will endeavour to explain here is how you specify the link between a switch, button or encoder action and the macro which it to be executed as a result.

The Macro File

All macro operations needed by the driver must be in one file—one file for all aircraft not specifically detailed in the INI file, with optionally different files for each named aircraft. The macro file is named in the relevant [Config] section of the INI file. The filetype is always "mcro"—FSUIPC loads all such files it finds in the Modules folder.

Macro names

The PFChid driver requests FSUIPC to execute macros by *name*, with a parameter where needed. The names are composed from a "base name", which is the name of the switch or encoder, possibly with a selector where there are multiple such switches, and an operational appendage detailing things like press, release, increment, decrement, fast, slow.

For example, here are some base names which we'll use for examples:

LandingGear The gear switch, which is on or off (down or up)

Magneto The Magneto/Starter switches, two off, each with 5 positions.

Hdg The heading bug encoding knob

See the PFCmacroindex document, a 'comma-separated-values' file, for a complete list. One will be produced in your Modules folder automatically when you run FS with PFChid installed, and that file will then contain details of any specific macros you have programmed.

Taking the easiest one first, you could have a Macro named **LandingGear** which would be called each time the state of that switch was changed. The parameter passed to the macro would be 1 for "down" or 0 for "up". For switches and buttons the parameters is always a 1 or a 0: 1 for on or pressed, 0 for off or released.

More usefully you can have two macros:

LandingGear+ to handle the on/pressed/down change and

LandingGear– to handle the off/released/up change.

In many cases of buttons, you'd only be wanting to handle the + part in any case, so you'd not provide the other macro.

The **Magneto** case introduces two differences, however. First off there are TWO separate magneto switches on the Cirrus II Pro, left and right or 1 and 2. Second, there are more than the simple "on" and "off" values—there are 5 possibilities, in fact, 0–4. In this case you'd need two macros:

Magneto1 and **Magneto2**

And both of these would need to handle the parameter, which will be one of 0–4. Or you can have separate macros for each of the 5 switch positions, like this:

Magneto1/0 for Magneto1 position 0

Magneto2/4 for Magneto2 position 4

Note that if you use this method the parameter supplied to the assigned control will be unused (it will be zero). This format can cope with multiway switches with up to 10 positions, 0-9.

The selector between multiple switches of the same type can be used with the + and – appendages too. If there were separate left and right landing gear switches you could have macros **LandingGear1+**, **LandingGear2+**, **LandingGear1–** and **LandingGear2–**. But of course, there aren't.

Finally, looking at the encoder **Hdg**, if the macro name isn't augmented then this macro needs to process the parameter which is supplied. This is one of these values:

1	increment
255	decrement
2	fast increment
254	fast decrement

But, again, rather more sensibly you can have separate Macros for each of these results (or only each one you want to handle), thus:

Hdg+ increment heading

Hdg– decrement heading

Hdg> fast increment heading

Hdg< fast decrement heading

GOOD FLYING!

Published by Peter L. Dowson, 29th January 2023