vJoyOffsets

Linking FSUIPC Offset values to Virtual Joystick inputs

Introduction:

There are many reasons why you may want to provide a program with joystick inputs (axes, sliders, buttons, or POV hats) which can be supplied by FSUIPC applications or plug-ins via Offsets.

Usually this is for flight simulator related hardware drivers which interface to the simulator via FSUIPC. But, equally, it might be a program such an one providing ATC which needs joystick button presses to control things like PTT.

Luckily, there's a way to do this by defining "virtual joysticks".

The pre-requisites

First you'll need the virtual joysticks installed into your system. This need not be the flight simulator PC – if you are using a Network linked by WideFS then you install the virtual joystick(s) you need on each PC which is running a program needing joystick input. To create these virtual joysticks you will need vJoy.

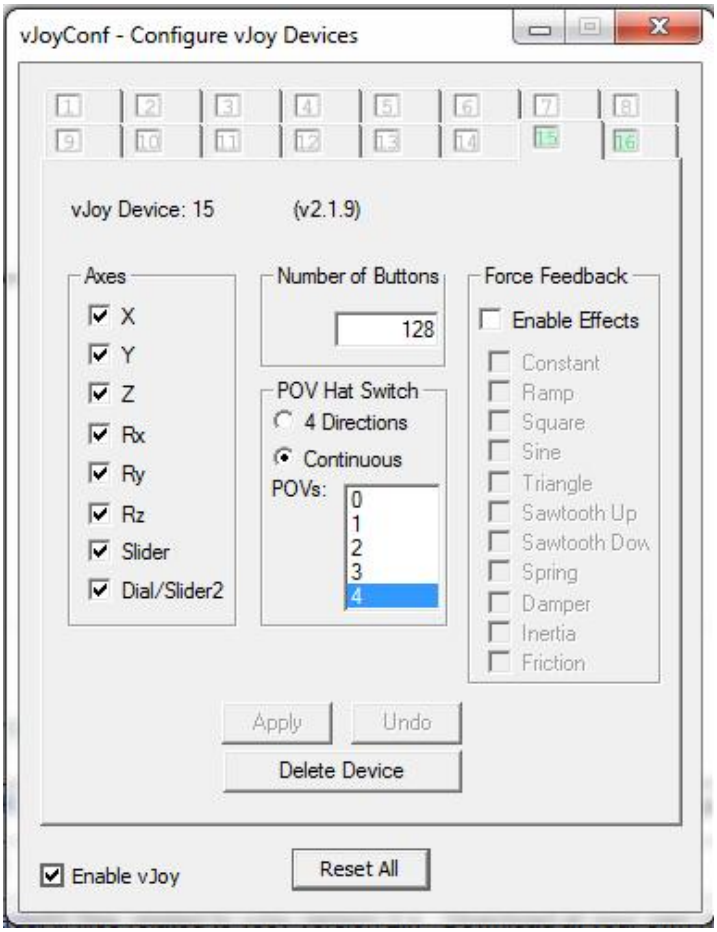
Thanks to Shaul Eizikovitch, vJoy is available as a freeware package. To download it Go to http://vjoystick.sourceforge.net/site/, and select the Download & Install section. There are separate download buttons for Windows 10 and "legacy" systems – Vista, Win7 and Win8.

As well as installing the appropriate driver, you will need the vJoyConfig program in order to define the parts of your virtual joysticks. The selection for that is further down that page.

vJoyConfig allows you to define up to 16 virtual joysticks, each with up to 8 axes, 4 POVs and 128 buttons, and the next section illustrates this and relates the vJoy identifiers to those known to FSUIPC.

Configuring your Virtual Joysticks

Here's what you might see when you run vJoyConfig. In this case I have two devices (15 and 16) already defined, as shown by the darkened selections at the top. The "Add Device" button is inactive for these as they've already been added:



Here I'm showing my Device 15, on which I've defined all 8 axes, all 128 buttons, and 4 continuous POVs. Ignore the force feedback section as vJoyOffsets doesn't use that.

Note that if you're intending to program these in FSUIPC assignments, currently whilst FSUIPC does detect the 8 axes, and the 4 POVs as axes too, it only sees the first 32 buttons, and the first POV as a set of 8 buttons.

A word about how FSUIPC would see them:

X, Y, Z are the same, X, Y and Z. But Rx, Ry, Rz, Slider and Dial/slider are seen as R, U, V, S and T respectively. The continuous POVs take any value from 0 to 35999 for the 360 degrees, but FSUIPC's axis assignments will see them in degrees (i.e. 0-359). The 4 direction setting will restrict it to either 4 buttons (POV1 only) or 4 axis values (0, 90, 180, 270). Also, note that the joystick numbers 1-16 used in vJoy are *not* the same IDs seen by FSUIPC, which are assigned by Windows.

Installing vJoyOffsets

The vJoyOffsets package consists only of vJoyOffsets.exe and this document. There are two ways of installing the program itself:

1. You can add it to the program folder already created by installing vJoy. This might be C:\Program Files\vJoy\x86 if you accepted its default install options. Note that vJoyOffsets is a 32-bit program, so it goes into the x86 subfolder, *not* the x64 one. Having done that, create a shortcut for the program so you can access it more easily.

2. Otherwise, create a folder for it and make a copy in that folder of the vJoyInterface.dll module which you'll find in the x86 subfolder mentioned above.

You then need to create a text file named vJoyOffsets.INI which will define the offsets you intend to use and what they should do. Before we get to that, though, there is a setting you'll want to add to start with:

```
[Logging]
LogConsole=Yes
```

By default the program gives no indication that it is doing anything – it just acts in the background. When you are defining its inputs (offsets) you will want to check the results. With "LogConsole=Yes" you will get these displayed sequentially in a console window, and there will also be a log file (vJoyOffsets.log) in the same folder. With LogConsole omitted or not set to "Yes" you will get neither.

You can also define:

```
ConsoleWindowTopMost=Yes
```

so the details will show on top of your other programs, like flight simulator.

You can size and position the console window and this will be remembered in the [Logging] section.

Finally, before getting to the nitty-gritty of offset definitions, note that you need FSUIPC or WideClient running and connected to the simulator *before* starting vJoyOffsets. Otherwise you'll just get an error message.

Defining the links between Offsets and Virtual Joysticks

Each virtual joystick you've defined in vJoyConfig should have a section in vJoyOffsets named accordingly, so

[Joystick1] ... [Joystick16]

Within each such section you define offsets for each aspect of that virtual joystick you want to control. The format of each of these is similar, with minor variations according to the type of input.

For Axes:

axis name = offset, size, default, minimum, maximum

Where:

axis name is one of X, Y, Z, R, U, V, S, T

offset is the offset base, in hexadecimal (but without the x or 0x), eg. 66C0

size is the number of bytes to be used. This should be 1, 2, or 4 for unsigned byte (UB), unsigned word (UW) or unsigned DWORd (UD), or -1, -2 or -4 for signed values.

default is the initial value to be set before any offset values are read.

minimum and maximum give the range. These are inclusive of the permitted values. If you want to reverse the action of the axis the minimum can be greater than the maximum.

Note that the value will be scaled to attempt to match the full normal axis range of -16384 to +16383. If you declare a range much less than those 32768 positions you might want to declare the maximum as something less than your true maximum. For example, scaling 0-127 to -16384 to +16383 will normally give a resulting maximum of 32512 (the nearest multiple of 256=32768/128 to the needed maximum). vJoyOffsets will ensure that the permitted range is not exceeded. In this example declaring the maximum as 126 will fix this even though you can get to 127.

Examples:

X=66C0,1,0,0,127

Y=66C1,1,127,0,127

Z=66C2,-2,0,-16384,16383

For Buttons:

The maximum of 128 buttons are dealt with as 4 batches of 32 buttons, with parameters Btns1, Btns2, Btn3 and Btns4. The format is:

BtnsN=offset, size, default

Here the size is 1, 2, 3, or 4 depending how many buttons you want to control in this set. 1 only addresses the first 8 buttons in this set – i.e. for Btns1 this would be buttons 1-8 (0-7 in FSUIPC terms) whilst with a size of 4 you control all 32 buttons.

Again, the default is the setting made before offsets are read. In this case it ia in hexadecimal preceded by an x.

Example: Btns1=66C4,4,x5555555

sets every other button in the range 1-32 (0-31)

Btns4=66C4,1,x40

sets button 102, the 6th one in the final range of 32 (96-128).

For POVs:

There are two types of POV definable in vJoy: the 4 switch variety (N, E, W, S) and continuous (all 360 degrees). You need to tell vJoyOffsets which type you are using. This is by a parameter:

POVswitch=Yes

Without this parameter the continuous type is assumed to be in use.

For the continuous type you define their setting here as follows:

POVN=offset, size

Where POVN is POV1 … POV4. The size needs to be at least 2 to accommodate the range of POV values 0-36000. This value is in 1/100ths of a degree.

When POVs are unpressed, the values in both types are -1. Since a two byte offset cannot accommodate both +36000 and -1, you would need 0xFFFF or 65535 to represent -1. It is less confusing if you can use a 4 byte (32-bit) signed value.

For the POVswitch type:

POVN=offset, size

But here are size of 1 is sufficient. The valid values you can write here are 0-3 (for N, E, S, W) and -1 (0xff) for released.

The default for all POVs is -1 for “released”.