vJoy Feeder SDK

Version 2.0.3 – March 2014

Table of Contents

This SDK includes all that is needed to write a feeder for vJoy version 2.0.3
Check for the latest SDK.

Files listing:

inc	Include folder
inc\public.h	vJoy general public definitions
inc\vjoyinterface.h	Interface function declaration for vJoyInterface.dll
lib	Library folder (x86)
lib\vJoyInterface.dll	vJoy Interface module – must be included with the feeder (x86)
lib\vJoyInterface.lib	Import library – you must link your feeder to it (x86)
lib\amd64	Library folder (x64)
lib\amd64\vJoyInterface.dll	vJoy Interface module – must be included with the feeder (x64)
lib\amd64\vJoyInterface.lib	Import library – you must link your feeder to it (x64)
src	Sources of an example feeder
src\vJoyClient.cpp	Sources
src\vJoyClient.sln	VS2008 Express solution
src\vJoyClient.vcproj	VS2008 Express project
src\stdafx.h	Additional header files

c#	C# SDK
x86	Library folder (x86)
x64	Library folder (x64)
WrapperTest	Demo Wrapper Project (Visual Studio 2008 Express)

Fundamentals:

This interface and example will enable you to write a C/C++ vJoy feeder.
To write a C# refer to ReadMe file in C# folder.

It is advisable to start your feeder from the supplied example and make the needed changes. Here are the five basic steps you might want to follow:

Test Driver:	Check that the driver is installed and enabled. Obtain information about the driver. An installed driver implies at least one vJoy device.
Test Virtual Device(s):	Get information regarding one or more devices. Read information about a specific device capabilities: Axes, buttons and POV hat switches.
Device acquisition:	Obtain status of a vJoy device. Acquire the device if the device status is *owned* or is *free.*
Updating:	Inject position data to a device (as long as the device is owened by the feeder). Position data includes the position of the axes, state of the buttons and state of the POV hat switches.
Relinquishing the device:	The device is *owned* by the feeder and cannot be fed by another application until relinquished.

Reccomended Practices:

Test vJoy Driver:

Before you start, check if the vJoy driver is installed and check that it is what you expected:

```
	// Get the driver attributes (Vendor ID, Product ID, Version Number)
	if (!vJoyEnabled())
	{
		_tprintf("Failed Getting vJoy attributes.\n");
		return -2;
	}
	else
	{
		_tprintf("Vendor: %S\nProduct :%S\nVersion Number:%S\n",\
		TEXT(GetvJoyManufacturerString()),\
		TEXT(GetvJoyProductString()),\
		TEXT(GetvJoySerialNumberString()));
	};
```

Test vJoy Virtual Devices:

Check which devices are installed and what their state is it:

```
	// Get the state of the requested device
	VjdStat status = GetVJDStatus(iInterface);
	switch (status)
	{
	case VJD_STAT_OWN:
		_tprintf("vJoy Device %d is already owned by this feeder\n", iInterface);
		break;
	case VJD_STAT_FREE:
		_tprintf("vJoy Device %d is free\n", iInterface);
		break;
	case VJD_STAT_BUSY:
		_tprintf("vJoy Device %d is already owned by another feeder\nCannot continue\n",
iInterface);
		return -3;
	case VJD_STAT_MISS:
		_tprintf("vJoy Device %d is not installed or disabled\nCannot continue\n",
iInterface);
		return -4;
	default:
		_tprintf("vJoy Device %d general error\nCannot continue\n", iInterface);
		return -1;
	};
```

Now make sure that the axes, buttons (and POV hat switches) are as expected:

```
	// Check which axes are supported
	BOOL AxisX  = GetVJDAxisExist(iInterface, HID_USAGE_X);
	BOOL AxisY  = GetVJDAxisExist(iInterface, HID_USAGE_Y);
	BOOL AxisZ  = GetVJDAxisExist(iInterface, HID_USAGE_Z);
	BOOL AxisRX = GetVJDAxisExist(iInterface, HID_USAGE_RX);
	// Get the number of buttons supported by this vJoy device
	int nButtons  = GetVJDButtonNumber(iInterface);
	// Print results
	_tprintf("\nvJoy Device %d capabilities\n", iInterface);
	_tprintf("Numner of buttons\t\t%d\n", nButtons);
	_tprintf("Axis X\t\t%s\n", AxisX?"Yes":"No");
	_tprintf("Axis Y\t\t%s\n", AxisX?"Yes":"No");
	_tprintf("Axis Z\t\t%s\n", AxisX?"Yes":"No");
	_tprintf("Axis Rx\t\t%s\n", AxisRX?"Yes":"No");
```

Acquire the vJoy Device:

Until now you just made inquiries about the system and about the vJoy device status. In order to change the position of the vJoy device you need to Acquire it (if it is not already owned):

```
	// Acquire the target
	if ((status == VJD_STAT_OWN) || ((status == VJD_STAT_FREE) && (!
AcquireVJD(iInterface))))
	{
		_tprintf("Failed to acquire vJoy device number %d.\n", iInterface);
		return -1;
	}
	else
	{
		_tprintf("Acquired: vJoy device number %d.\n", iInterface);
	}
```

Feed vJoy Device:

The time has come to do some real work: feed the vJoy device with position data.

There are two approaches:

1. **Efficient**: Collect position data, place the data in a position structure then finally send the data to the device.
2. **Robust**: Reset the device once then send the position data for every control (axis, button,POV) at a time.

The first approach is more efficient but requires more code to deal with the position structure. The second approach hides the details of the data fed to the device at the expence of exessive calls to the device driver.

Efficient:

```
/*** Create the data packet that holds the entire position info ***/

// Set the device ID
id = (BYTE)iInterface;
iReport.bDevice = id;

// Set values in four axes (Leave the rest in default state)
iReport.wAxisX=X;
iReport.wAxisY=Y;
iReport.wAxisZ=Z;
iReport.wAxisZRot=ZR;

// Set buttons one by one
iReport.lButtons = 1<<count/20;

if (ContinuousPOV)
{
      // Make Continuous POV Hat spin
      iReport.bHats           = (DWORD)(count*70);
      iReport.bHatsEx1  = (DWORD)(count*70)+3000;
      iReport.bHatsEx2  = (DWORD)(count*70)+5000;
      iReport.bHatsEx3  = 15000 - (DWORD)(count*70);
      if ((count*70) > 36000)
      {
            iReport.bHats = -1; // Neutral state
            iReport.bHatsEx1 = -1; // Neutral state
            iReport.bHatsEx2 = -1; // Neutral state
            iReport.bHatsEx3 = -1; // Neutral state
      };
}
else
{
      // Make 5-position POV Hat spin
      unsigned char pov[4];
      pov[0] = ((count/20) + 0)%4;
      pov[1] = ((count/20) + 1)%4;
      pov[2] = ((count/20) + 2)%4;
      pov[3] = ((count/20) + 3)%4;

      iReport.bHats           = (pov[3]<<12) | (pov[2]<<8) | (pov[1]<<4) | pov[0];
      if ((count) > 550)
            iReport.bHats = -1; // Neutral state
};
```

If the structure changes in the future then the code will have to change too.

Robust:

```
	// Reset this device to default values
	ResetVJD(iInterface);

	// Feed the device in endless loop
	while(1)
	{
		for(int i=0;i<10;i++)
		{
			// Set position of 4 axes
			res = SetAxis(value+00, iInterface, HID_USAGE_X);
			res = SetAxis(value+10, iInterface, HID_USAGE_Y);
			res = SetAxis(value+20, iInterface, HID_USAGE_Z);
			res = SetAxis(value+30, iInterface, HID_USAGE_RX);
			res = SetAxis(value+40, iInterface, HID_USAGE_RZ);

			// Press Button 1, Keep button 3 not pressed
			res = SetBtn(TRUE, iInterface, 1);
			res = SetBtn(FALSE, iInterface, 3);
		}

	Sleep(20);
	value+=10;
	}
```

This code is readable and does not relay on any specific structure. However, the driver is updated with every *SetAxis()* and every *SetBtn()*.

Relinquish the vJoy Device:

You must relinquish the device when the driver exits:

```
RelinquishVJD(iInterface);
```

Interface Function Reference:

General driver data

The following functions return general data regarding the installed vJoy device driver. It is recommended to call them when starting your feeder.

```
VJOYINTERFACE_API BOOL   __cdecl   vJoyEnabled(void);
```
Returns TRUE if vJoy version 2.x is installed and enabled.

```
VJOYINTERFACE_API SHORT __cdecl    GetvJoyVersion(void);
```
Return the version number of the installed vJoy. To be used only after `vJoyEnabled()`

```
VJOYINTERFACE_API PVOID  __cdecl   GetvJoyProductString(void);
VJOYINTERFACE_API PVOID  __cdecl   GetvJoyManufacturerString(void);
VJOYINTERFACE_API PVOID  __cdecl   GetvJoySerialNumberString(void);
```
These functions return an LPTSTR that points to the correct data (Product, Manufacturer or Serial number). To be used only after `vJoyEnabled()`

Write access to vJoy Device

The following functions access the virtual device by its ID (rID). The value of rID may vary between 1 and 16.
There may be more than one virtual device installed on a given system.
VJD stands for Virtual Joystick Device.

```
VJOYINTERFACE_API enum VjdStat     __cdecl   GetVJDStatus(UINT rID);
```
Returns the status of the specified device
The status can be one of the following values:

- VJD_STAT_OWN // The vJoy Device is owned by this application.
- VJD_STAT_FREE // The vJoy Device is NOT owned by any application (including this one).
- VJD_STAT_BUSY // The vJoy Device is owned by another application.
  // It cannot be acquired by this application.
- VJD_STAT_MISS // The vJoy Device is missing. It either does not exist or the driver is disabled.
- VJD_STAT_UNKN // Unknown

```
VJOYINTERFACE_API BOOL        __cdecl   AcquireVJD(UINT rID);
```
Acquire the specified device.
Only a device in state VJD_STAT_FREE can be acquired.
If acquisition is successful the function returns TRUE and the device status becomes VJD_STAT_OWN.

```
VJOYINTERFACE_API VOID        __cdecl   RelinquishVJD(UINT rID);
```
Relinquish the previously acquired specified device.
Use only when device is state VJD_STAT_OWN.
State becomes VJD_STAT_FREE immediately after this function returns.

```
VJOYINTERFACE_API BOOL        __cdecl   UpdateVJD(UINT rID, PVOID pData);
```
Update the position data of the specified device.
Use only after device has been succesfully acquired.
Input parameter is a pointer to structure of type JOYSTICK_POSITION that holds the position data.
Returns TRUE if device updated.

vJoy Device properties

The following functions receive the virtual device ID (rID) and return the relevant data.
The value of rID may vary between 1 and 16. There may be more than one virtual device installed on a given system.
The return values are meaningful only if the specified device exists
VJD stands for Virtual Joystick Device.

```
VJOYINTERFACE_API int    __cdecl  GetVJDButtonNumber(UINT rID);
```
Returns the number of buttons in the specified device.
Valid values are 0 to 32

```
VJOYINTERFACE_API int    __cdecl  GetVJDDiscPovNumber(UINT rID);
```
Returns the number of discrete-type POV hats in the specified device
Discrete-type POV Hat values may be North, East, South, West or neutral
Valid values are 0 to 4 (from version 2.0.1)

```
VJOYINTERFACE_API int    __cdecl  GetVJDContPovNumber(UINT rID);
```
Returns the number of continuous-type POV hats in the specified device
continuous-type POV Hat values may be 0 to 35900
Valid values are 0 to 4 (from version 2.0.1)

```
VJOYINTERFACE_API BOOL   __cdecl  GetVJDAxisExist(UINT rID, UINT Axis);
```
Returns TRUE is the specified axis exists in the specified device
Axis values can be:

```
HID_USAGE_X     // X Axis
HID_USAGE_Y     // Y Axis
HID_USAGE_Z     // Z Axis
HID_USAGE_RX    // Rx Axis
HID_USAGE_RY    // Ry Axis
HID_USAGE_RZ    // Rz Axis
HID_USAGE_SL0   // Slider 0
HID_USAGE_SL1   // Slider 1
HID_USAGE_WHL   // Wheel
```

Robust write access to vJoy Devices

The following functions receive the virtual device ID (rID) and return the relevant data.
These functions hide the details of the position data structure by allowing you to alter the value of a specific control.
The downside of these functions is that you inject the data to the device serially as opposed to function *UpdateVJD()*.
The value of rID may vary between 1 and 16. There may be more than one virtual device installed on a given system.

```
VJOYINTERFACE_API BOOL    __cdecl   ResetVJD(UINT rID);
```
Resets all the controls of the specified device to a set of values.
These values are hard coded in the interface DLL and are currently set as follows:

- Axes X, Y & Z: Middle point.
- All other axes: 0.
- POV Switches: Neutral (-1).
- Buttons: Not Pressed (0).

```
VJOYINTERFACE_API BOOL    __cdecl   ResetAll(void);
```
Resets all the controls of the all devices to a set of values.
See function Reset VJD for details.

```
VJOYINTERFACE_API BOOL    __cdecl   ResetButtons(UINT rID);
```
Resets all buttons (To 0) in the specified device.

```
VJOYINTERFACE_API BOOL    __cdecl   ResetPovs(UINT rID);
```
Resets all POV Switches (To -1) in the specified device.

```
VJOYINTERFACE_API BOOL    __cdecl   SetAxis(LONG Value, UINT rID, UINT Axis);
```
Write Value to a given axis defined in the specified VDJ.
Value in the range 0x1-0x8000
Axis can be one of the following:

```
HID_USAGE_X     // X Axis
HID_USAGE_Y     // Y Axis
HID_USAGE_Z     // Z Axis
HID_USAGE_RX    // Rx Axis
HID_USAGE_RY    // Ry Axis
HID_USAGE_RZ    // Rz Axis
HID_USAGE_SL0   // Slider 0
HID_USAGE_SL1   // Slider 1
HID_USAGE_WHL   // Wheel
```

```
VJOYINTERFACE_API BOOL    __cdecl   SetBtn(BOOL Value, UINT rID, UCHAR nBtn);
```
Write **Value** (TRUE or FALSE) to a given button defined in the specified VDJ.
nBtn can in the range 1-32

```
VJOYINTERFACE_API BOOL    __cdecl   SetDiscPov(int Value, UINT rID, UCHAR nPov);
```
Write Value to a given discrete POV defined in the specified VDJ
Value can be one of the following:

0: North (or Forwards)
1: East (or Right)
2: South (or backwards)
3: West (or left)
-1: Neutral (Nothing pressed)

nPov selects the destination POV Switch. It can be 1 to 4

```
VJOYINTERFACE_API BOOL   __cdecl  SetContPov(DWORD Value,UINT rID,UCHAR nPov);
```

Write Value to a given continuous POV defined in the specified VDJ
Value can be in the range: -1 to 35999. It is measured in units of one-hundredth a degree. -1 means Neutral (Nothing pressed).
nPov selects the destination POV Switch. It can be 1 to 4

Build & Deploy:

The quickest way to build your project is to start from the supplied demo project written in C under Visual Studio 2008 Express. It will compile as-is for x86 target machines.
There's no real benefit in compiling for x64. However, the x64 library files (lib\amd64\vJoyInterface.dll and lib\amd64\vJoyInterface.lib) are provided.
When you deploy your feeder, don't forget to supply the user with file vJoyInterface.dll. It should be located on the target machine's DLL search path. Usually meaning the same directory as your feeder.