



Newland

SCANNING MADE SIMPLE

UFCOM Driver User Guide

USB Flexible Virtual COM Driver User Guide

Table of Contents

- 1. UFCOM Introduction
- 2. Virtual COM port management with UFCOM
 - 2.1. UFCOM install and remove
 - 2.1.1. Silent installing and removing of UFCOM driver package
 - 2.2. UFCOM working behavior configuration
 - 2.2.1. Using PuTTY to test data receiving and sending
 - 2.2.2. VCOM Lifemode
 - 2.2.3. Modify port number for VCOM
 - 2.2.4. Know which application has opened the VCOM
 - 2.2.5. Cleanup stale VCOMs to release port number
 - 2.2.6. Port number Bindmode
 - 2.3. Troubleshooting
 - 2.3.1. Driver package installation troubleshooting
 - 2.3.2. VCOM device is not created (in Device Manager)
 - 2.3.3. Diagnose USB data communication error
- 3. Advanced Topics
 - 3.1. Export and Import UFCOM settings across different machines
 - 3.2. Viewing USB data flow statistics
 - 3.3. msi installer generated registry items
- 4. VCOM device API behavior
- 5. UFCOM-specific API guide
 - 5.1. Test UFCOM API with vcomtest program
 - 5.2. Detect USB device plugging and unplugging
 - 5.3. Use one-time Temporal feature to get instant USB unplugging notification
 - 5.4. How to determine whether a COM port device is from UFCOM

This document applies to UFCOM 1.7.9 .

1. UFCOM Introduction

UFCOM(abbreviation of USB Flexible COM Port) is a brand new Virtual COM Port driver developed by Newland Auto-ID. We provide this driver to our customer since year 2017. This driver works with our barcode scanner hardware; it enables bidirectional communication between your Windows applications and our scanners. UFCOM works on Windows XP ~ Windows 10, x86 & x64 platforms, and their Server counterparts.

UFCOM provides similar functionality as that of Windows stock usbser.sys and acts as a replacement for usbser.sys. UFCOM not only fixes many ugly behaviors of usbser.sys, but also is much more feature-rich. Highlights of UFCOM are listed below:

1. Multiple barcode scanners support.

UFCOM works with many USB devices(e.g barcode scanners) at the same time, at least eight. Each scanner will get associated with a separate Virtual COM Port device(VCOM) in your Windows system.

2. VCOM Lifemode.

By choosing a VCOM Lifemode, user can control whether the VCOM is still present after the scanner is unplugged from your system. In many cases, we intend the VCOM to remain present after scanner is unplugged(accidentally or purposely), so that when the scanner is plugged in again, the data flow resumes, and the application works smoothly – without experiencing ineffective device handles in the midway.

3. Port number Bindmode.

By choosing a Bindmode, user can decide whether VCOM port number association behavior is determined by scanner device-type or by device-serial-number, or by USB port location.

4. Dynamic device display name.

In Windows Device Manager, VCOM device display-name dynamically reflects device working state. For example, a normal working scanner will have [online] prefix, while an unplugged scanner will have [offline] prefix.

5. VCOM Settings UI.

Check into Device Manager's VCOM device property dialog, there is a VCOM Settings UI section, where users can conveniently adjust various properties of VCOM devices, e.g., viewing and changing Lifemode, Bindmode and port number. From there, you can even check USB link layer errors, view USB data flow statistics .

Some background information for our existing customers

Before UFCOM's debut, we used to provide two sets of Virtual COM Port drivers to our customers.

- A so-called USB Datapipe driver(legacy driver): You need this when scanner USB mode is set to "USB Datapipe" or "com0com virtual COM port".
- CDC Virtual COM driver: You need this when scanner USB mode is set to "USB-CDC" and you use it on Windows is 8.1 or older Windows versions. This driver actually refers to Windows stock usbser.sys . This means, you will not be able to upgrade this driver even if Microsoft has new fixes available for future Windows versions. For example, You can not use new usbser.sys from Windows 10 onto your Windows 7 – unless you upgrade your whole system to Windows 10.

Now, UFCOM is a thorough replacement of the two. That is, whether you set your scanner USB mode to any one above, UFCOM will create virtual COM ports for them.

UFCOM supports legacy Datapipe API, so legacy applications that use udp_op.dll can work with UFCOM. But note: some older applications may be buggy and newer version is required to work with UFCOM.

2. Virtual COM port management with UFCOM

2.1. UFCOM install and remove

【 Background information 】

For our customer to be bright on Windows system driver installation, I'd like to give some key introduction on this topic. "Driver installation" actually consists of three steps.

1. Copy a **driver package** to a system folder called **DriverStore**. On Windows 7 and above, this folder is C:\Windows\System32\DriverStore\FileRepository . A driver package typically contains file types of .inf, .sys and .cat , and possibly some dll and exe The inf file is the key, you can think of an inf file representing a driver package.
2. Windows searches for a driver package match for a detected device. It means, when the device(scanner) hardware is detected by Windows, the device reports its hardware-id to Windows, Windows then searches all inf files inside DriverStore for matches. It is possible to find multiple matches for a single hardware-id, but Windows at the same moment will pick only one to drive the device. Normally, Windows will pick the best one, typically, the one with digital signature and with newer date.
3. Load the driver code(.sys) into memory and execute the code. From this time one, hardware starts to exchange data with Windows.

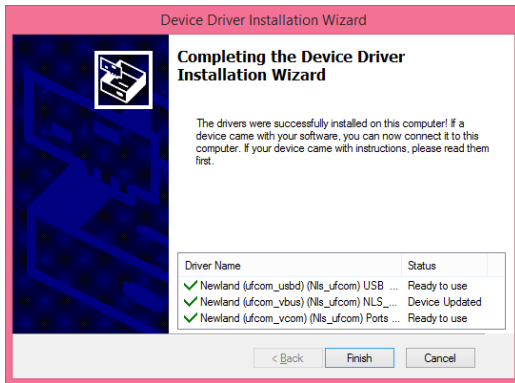
When Windows has determined a driver match, this matching is recorded into registry, and stay there even if the device is unplugged. So, when the device is plugged in a second time, Windows can skip the driver match step, and use the previously recorded one as driver to load into memory. But if you Uninstall a device from Device Manager , the matching information is deleted from registry, so next time device plugging-in will trigger driver package searching again.

【 UFCOM driver installation procedure 】

Since UFCOM 1.4.0, UFCOM installation package is presented as an msi file. You know that msi is the "standard" installation package format for Windows defined by Microsoft.

User double clicks the msi file, answer Next several times and the driver package installation is done. Sign of success is: we see a Completing the Device Driver Installation Wizard dialog with three green ticks.

If Windows prompts for reboot, do so to ensure installation completes.



UFCOM driver package installation does the following works:

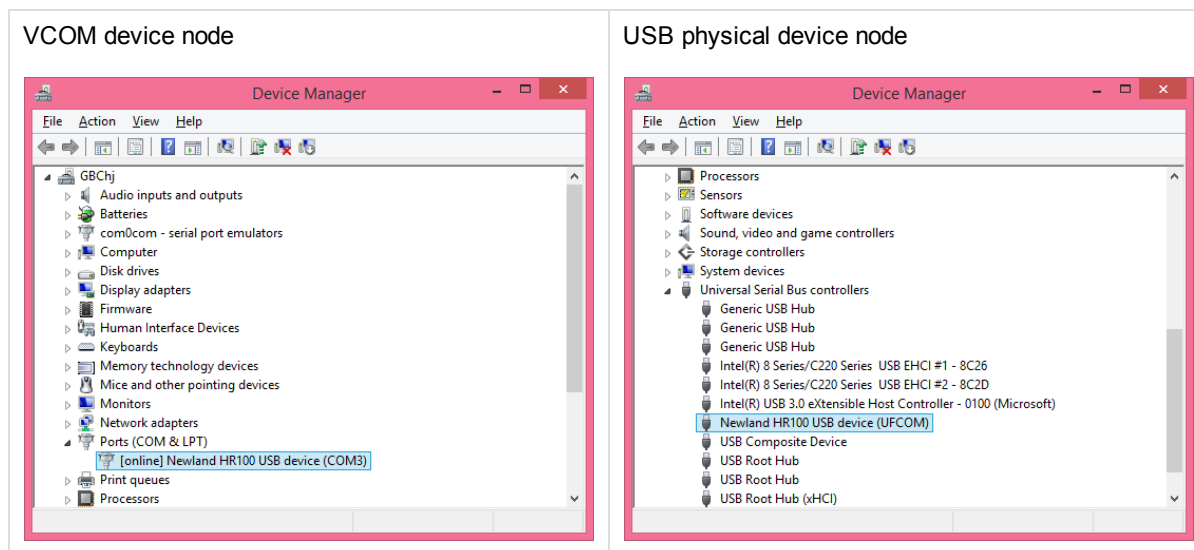
1. Extract inf-represented driver files to C:\Program Files\UFCOM (for 64-bit Windows, C:\Program Files (x86)\UFCOM).
2. Execute DPInst.exe, so that inf-represented driver files are further copied to DriverStore .
3. If Windows has existing devices that have been matched with UFCOM driver(e.g. already plugged in barcode scanners), these devices will be updated to match UFCOM driver package(if it's a better match).
4. If some old driver files cannot be unloaded from memory(COM port being opened for example), let Windows prompt user to reboot the whole system.

For first time installation, it does not matter you plug in scanner first or run installation package first. Both are OK.

We suggest you close all applications operating the scanner before running installation, this can reduce the chance you need to reboot Windows.

Now plug in your USB scanner(if you have not done that), Windows now searches and loads driver for it. For first scanner plugged in, it costs several to tens of seconds, depending on your Windows system performance.

To confirm that scanner is driven successfully by UFCOM, you can check it with Device Manager. In Device Manager, we actually see two new devices for a physical scanner. One is the VCOM device, another is your real scanner device(or call it physical device).



If the scanner is unplugged, the physical device node will definitely disappear(unplugged state), but the VCOM device node does not necessarily disappear. Whether VCOM should disappear is determined by VCOM Lifemode which you can control. VCOM Lifemode is explained later.

【 Doing an upgrade install 】

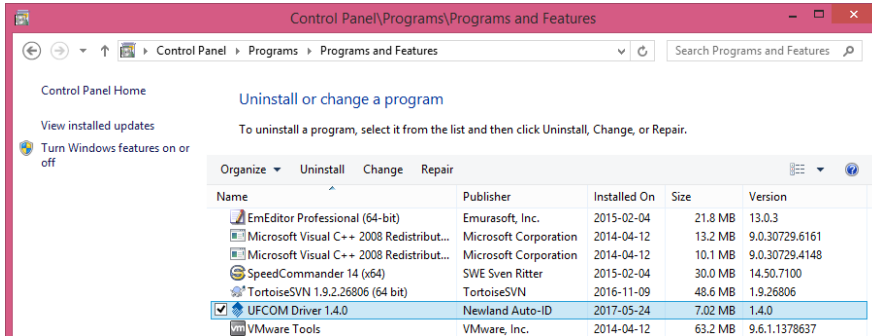
Running a newer version of UFCOM msi will automatically remove existing older UFCOM version.

With a newer UFCOM version already installed, an older version will refuse to install itself. If you need older version explicitly, you have to first remove older version manually.

【 Uninstalling UFCOM 】

From Control Panel(AppWiz.cpl), you can uninstall UFCOM driver package.

Find the entry with the name like **UFCOM Driver 1.4.0** , double click it to uninstall.



Un-installation here means:

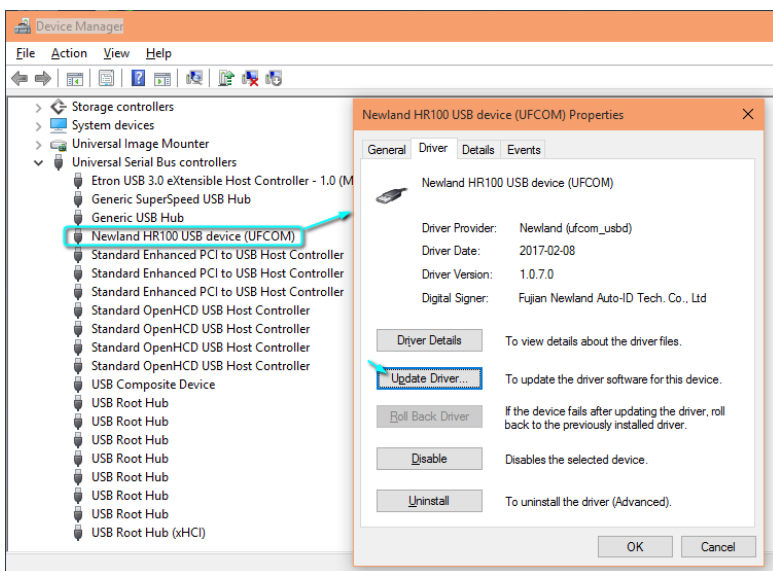
- UFCOM Driver package is removed from DriverStore .
- Device nodes previously driven by UFCOM will receive new driver matching, which may get a match of another driver or a stock Windows driver.

【 Manually choosing a driver package for USB device 】

If for any reason you need to go back to the old Datapipe driver, or Windows stock driver, you don't have to uninstall UFCOM package, because there is a simpler way provided by Windows itself.

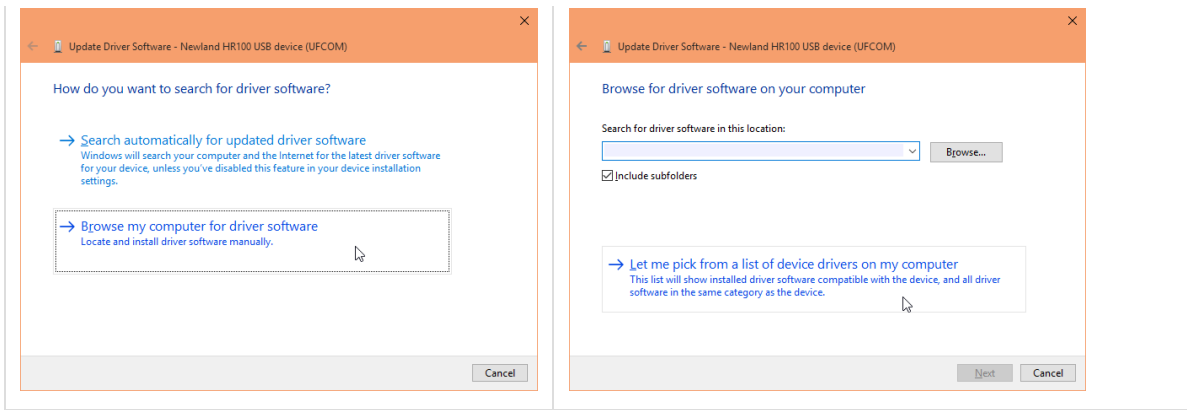
Use Windows 10 as example, do it as follows:

Find the physical device node in Device Manager(NOT the VCOM node), open its properties, click Update Driver.

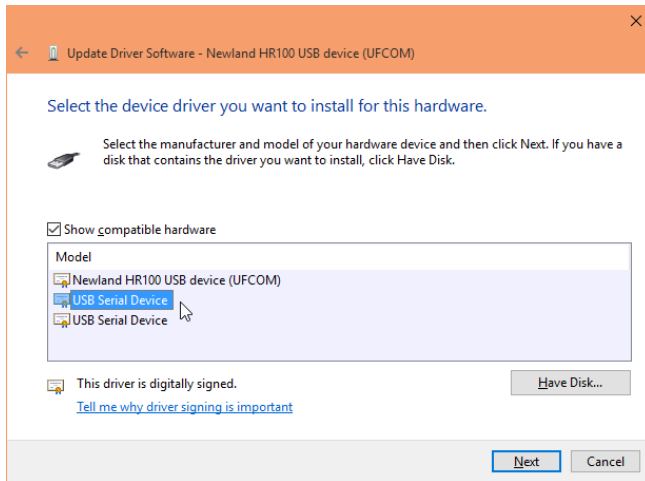


Browse my computer for driver software.

Let me pick from a list of device drivers on my computer.



In Show compatible hardware list box, pick a package to apply. Here, a driver-package is identified by "device model name" inside its corresponding inf file. In the example illustrated below, "Newland HR100 USB device(UFCOM)" represents UFCOM driver package, and "USB Serial Device" represents Windows stock usbser.sys .



Click Next a final time. To confirm your selection.

Of course, you can switch back to UFCOM driver by the same way.

NOTE: If you insist using legacy Datapipe driver side-by-side with UFCOM, and wish that legacy driver provides Datapipe functionality, you need one extra step. That is, **UFCOM Global Settings** → **Datapipe device lifemode** set to "Never".

For average users, We don't need to bother with this. Just leave **Datapipe device lifemode** option it default value "On device present".

2.1.1. Silent installing and removing of UFCOM driver package

UFCOM msi supports standard silent installing and removing operation. This enables integrating UFCOM installation procedure into your scripting environment. To install silently(i.e. quiet mode), just execute command:

```
msiexec /q /i ufcom-1.7.2.msi
```

NOTE: You must run that command with administrative rights, otherwise it will fail silently. On success, UFCOM driver entry will also appear in AppWiz.cpl .

If you launch silent install from a batch script, and would like to wait for msi install result before next batch command runs, you should instead write:

```
start /wait msiexec /q /i ufcom-1.7.2.msi
echo %ERRORLEVEL%
```


Friendly note: Never write `.\ufcom-1.7.2.msi` as `msiexec`'s parameter. The prefix `.\` will cause `msiexec` to do nothing. This is apparently a bug in `msiexec`, but Microsoft has not fixed it yet. By the way, using an absolute path is OK.

Silent installing a new UFCOM version will automatically uninstall older existing version.

`msiexec`'s exit code(as indicated by `%ERRORLEVEL%` in batch file) tells whether it succeeds. Exit code 0 means success, otherwise fail. Typical failure reasons are:

- You did not run `msiexec` as Administrator.
- A newer version of UFCOM already existed.
- (since 1.7.5) When upgrading to a newer version, some files from old version cannot be replaced(being used by the system). You need to reboot Windows so the replacement can take place.

To do silent removal, we need the original msi file matching currently installed version. For example, current installed version is 1.7.2, then you need `ufcom-1.7.2.msi` to do the removal(uninstall). Removal command line is like:

```
start /wait msiexec /q /x ufcom-1.7.2.msi
```

2.2. UFCOM working behavior configuration

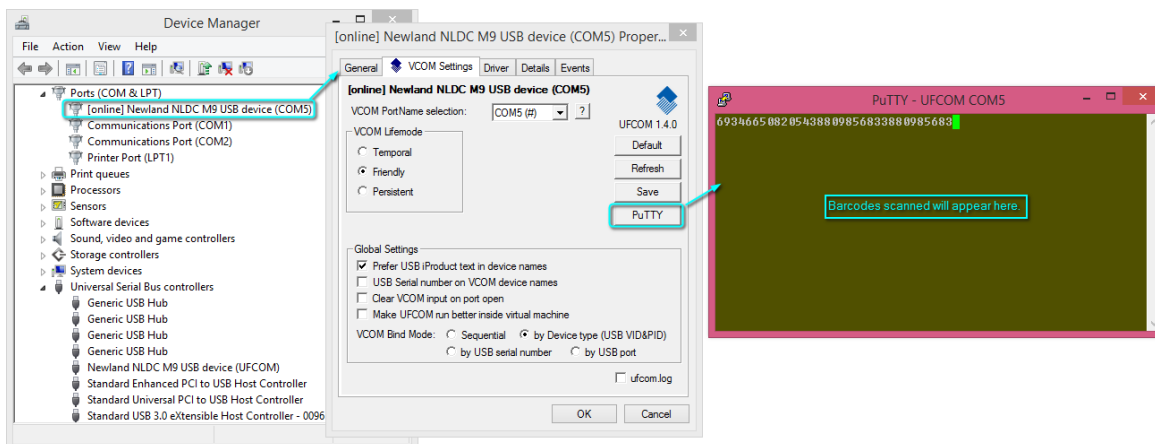
For UFCOM generated VCOM device, you can open it with any COM port communication software, operate it with standard Windows COM port API(or called Serial port API). Any data you read/write comes from/goes to USB device itself. On Windows XP, you can use HyperTerminal; on Windows 7 and above, no HyperTerminal any more, but it's good to know UFCOM provides free PuTTY to verify simple data receiving and sending. For some advanced configuration of your scanner, Newland Auto-ID provides EasySet configuration software.

Due to the fact the Windows COM port API is designed with physical RS-232 serial port, it does not make consideration about USB device's special traits, like unpluggability at any time, power management. So UFCOM needs to design special way to control these special traits.

2.2.1. Using PuTTY to test data receiving and sending

From Windows device manager, find your VCOM device node, double click it(or right-click menu, Properties), we see device property dialog, then go to **VCOM Settings** tab(we call it VCOM Settings UI later), click PuTTY button, now we get a terminal window to communicate with our scanner.

- Have your scanner scan some barcode; barcode content should appear in PuTTY window.
- Type some characters into PuTTY window(will see local echo), they will be sent to barcode scanner. If you need to send a long string to the scanner, you can type out that string in a text editor, copy the text, then go back to PuTTY window, mouse right-click to paste them.



2.2.2. VCOM Lifemode

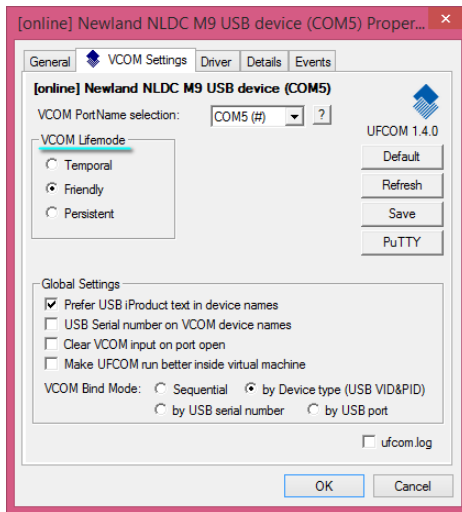
VCOM Lifemode determines when should VCOM appear(be present), when should it disappear. "Disappear" or called "non-present" means: You will not see it in Device Manager – unless you tell Device Manager to display hidden devices.

Lifemode	VCOM display prefix	Conditions that VCOM present
Temporal	<online>	When scanner plugs in, VCOM appears; when scanner unplugs, VCOM disappears immediately.
Friendly (default)	[online] [offline]	When scanner is plugged in, and there is application using this VCOM(has a opened handle of this VCOM), VCOM will not disappear, and the already opened handle remains valid. Later when the scanner re-plugs in, the communication channel is re-established. That means: an application holding a VCOM handle will not be interfered with temporary plug-in/unplug of the physical device. When scanner-unplugging and VCOM handle closing are both true, VCOM will disappear.
Persistent	[[online]] [[offline]]	VCOM exists forever. Persistent mode makes a VCOM look almost like a on-board physical COM port. It always exists no matter scanner is plugged-in or unplugged.

Meaning of VCOM display name prefix "online, offline":

- "online" means scanner is plugged in and the driver works normally.
- "offline" means scanner is physically unplugged.

Method to set Lifemode: In Device Manager, locate a VCOM device node(not physical device node), open its properties dialog. Switch to VCOM Settings UI, where you can set Lifemode for this VCOM.

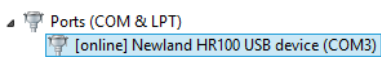


NOTE: If you are using two scanners, there will be two VCOM devices, and their Lifemode is configured separately.

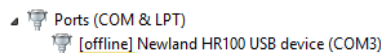
i You need administrative rights on Windows to make changes to VCOM Settings. For example, run devmgmt.msc as Administrator.

【 Observing VCOM device "online/offline" prefix change 】

Plug a scanner into Windows , the corresponding VCOM device display name will have a "online" prefix.



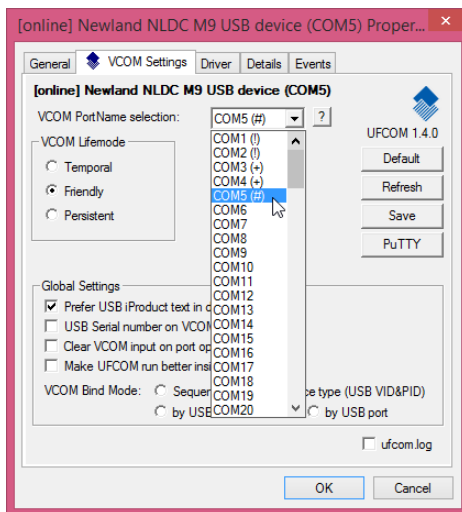
Unplug the scanner, and, if the VCOM should remain present according to its Lifemode rule, the prefix becomes "offline".



2.2.3. Modify port number for VCOM

This is a widely used feature.

When Windows creates a VCOM device for us, the Windows system picks a free(not yet reserved) port number for this VCOM, normally picking the one with lowest number available(from COM3 and above). If we are unsatisfied with the auto-picked one, we can change it. With VCOM Settings UI, you can do it without hassle.



Before picking a new port number(precisely called PortName), please be aware of the status mark alongside each PortName entry. These marks have special meanings:

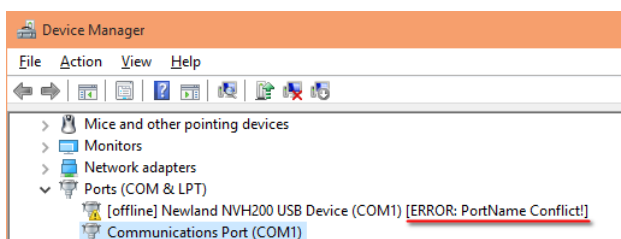
Mark	Meaning
#	This port number is being used by the VCOM itself.
!	This port number is being occupied by another COM port device, and this another COM port device now truly exists on the system(plugged-in state).
+	Some other UFCOM VCOM device has reserved this port number, and this other device is currently unplugged.
-	Windows ComDB database has reserved this port number for some other device, which is also currently unplugged.

No need to memorize those symbol marks. Clicking the small question mark button at right side, you get these explanations.

We suggest selecting a port number with no mark, which means not occupied or reserved, so you reduce the chance to interfere with other devices on the system.

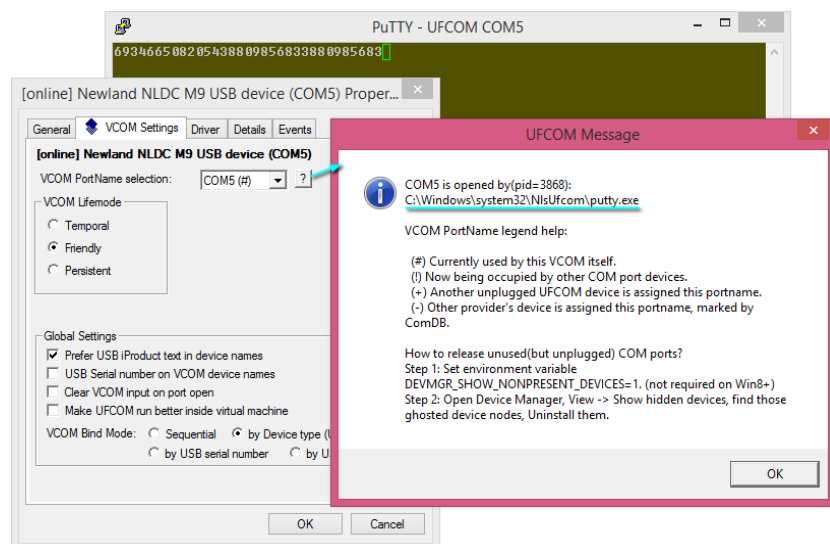
- If you select a port number with (!) mark, it will definitely cause error.
- Selecting (+) (-) mark may be OK, COM7 from the image for example. But be aware, when the original COM7 device(maybe a Virtual COM Port from another vendor) is re-plugged in, that one may or may not work normally. The final behavior is determined by that driver's code logic.

If you choose a port number with (!) mark, UFCOM will indicate the error reason on device display name. For example: When on-board COM1 exists and you force changing a VCOM to using COM1, Device Manager will show this VCOM as working abnormally and the device display name will have prompt text "[ERROR: PortName Conflict!]".



2.2.4. Know which application has opened the VCOM

Click the small question mark button beside PortName list box, we can know whether the VCOM is opened by some application. The example image below shows: COM5 is being opened by putty.exe .

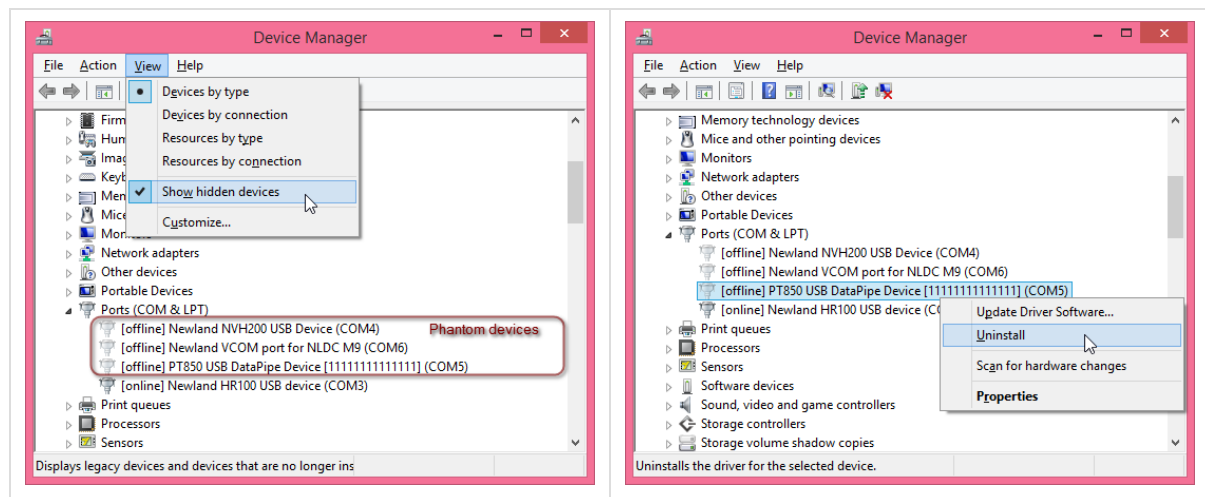


2.2.5. Cleanup stale VCOMs to release port number

After we unplug a scanner and close its VCOM handle, we see corresponding VCOM device node disappearing from Device Manager, but actually, it is only hidden. The VCOM related settings are still recorded in Windows registry. These settings include: VCOM port number, Lifemode etc. When the same VCOM device instance appears again, Windows will re-apply those settings. This is reasonable design. Tech note: a VCOM device instance is identified by the so-called "Device instance path", which can be queried from device properties, Details tab.

After many scanners have been plugged and unplugged, many port numbers would have been reserved. If we want to reclaim those port numbers so that they becomes free again for others, the best way is to "Uninstall" the old device nodes.

Way to do that: Open Device Manager, enable menu item View → Show hidden devices. Now we can see some more devices with dimmed icons, each of which represents an unplugged but still registered device. We call them phantom devices. Right click and Uninstall a phantom device, the corresponding port number is reclaimed.



NOTE: On Windows XP~Windows 7, in order to see phantom devices, we have to do an extra step, that is, add an environment variable `DEVMGR_SHOW_NONPRESENT_DEVICES=1` then restart Device Manager.

No need to do this since Windows 8.

2.2.6. Port number Bindmode

COM Port Bindmode is a global setting. It governs the rule how a physical scanner gets associated to a COM port number.

UFCOM provides four Bindmodes, You need some imagination to understand them.

Bindmode	Meaning
Sequential	<p>UFCOM considers all scanners equal. The result is, port number is allocated one by one, typically sequentially.</p> <p>Example: If we have two scanners, plugging in scanner A, we get COM3; plugging in scanner B at the same time, we get COM4. Later we will see, we unplug both scanners and plug in either one, the one will always get COM3 .</p>
by Device type (default)	<p>Simply speaking, UFCOM takes consideration of scanner device type. Different device types will get different COM port numbers. Scanner device type is identified by VID,PID tuple from USB device descriptor.</p> <p>Example: Assume we have an HR100 and an HR200 (different types). First time plugging in HR100, we get COM5; first time plugging in HR200, we get COM6. Later we will see: Unplug and replug the two scanners, HR100 will always get COM5, HR200 will always get COM6, regardless of whether the other scanner is present or not. So , COM5 is bound to HR100 and COM6 is bound to HR200. This binding relation persists until you Uninstall corresponding VCOM devices.</p>
by USB serial number	<p>Devices with different serial numbers will get different COM port numbers. For a USB device, the serial number is determined by iSerialNumber text reported in USB device descriptor.</p> <p>When we need to bind two same-type scanners to their own VCOM, we can use this Bindmode. But note: This can be achieved only if both scanners has serial numbers burned-in. For scanners with no serial number, Bindmode behavior will fall back to be "by device type".</p>
by USB port	<p>Devices at different USB ports will get different COM port numbers. So the device type and serial number is not considered.</p> <p>This Bindmode is available only in Windows 7 and above.</p>

Extra notes:

- Currently, switching Bindmode will cause all VCOM opened handles to become ineffective. Applications have to reopen VCOM to get new handles.
- When switching Bindmode. The port numbers used by previous bindmode is not reclaimed by system. So, switching bindmode usually causes the same scanner to receive a different port number. If you want the former port number, just manually change it.

2.3. Troubleshooting

2.3.1. Driver package installation troubleshooting

【 Collecting install logs 】

Driver package install problem can be diagnosed from several log files. We need log files from the following places:

Software component	How to collect log file
msiexec	<p>We need to enable msiexec logging. msiexec.exe is the program that process your msi files.</p> <p>Instead of double clicking msi file, we run it from a Administrator CMD window:</p> <pre>msiexec /i ufcom-1.7.2.msi /l*v my.log</pre> <p>Then collect my.log .</p>
DPInst	<p>After install finishes, collect C:\Windows\DPINST.LOG .</p> <p>Hint: we suggest removing or renaming old DPINST.LOG first, so that we can see new content easily.</p>

Software component	How to collect log file
SetupAPI	<p>After install finishes, collect C:\Windows\Inf\setupapi.dev.log .</p> <p>Hint: we suggest removing or renaming old setupapi.dev.log first, so that we can see new content easily.</p> <p>Hint: On Windows XP, the file is C:\Windows\setupapi.log .</p>

【 Fixing driver package 】

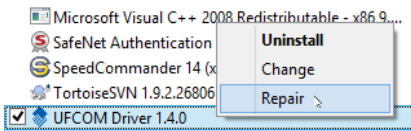
Sometimes, UFCOM driver files get accidentally deleted, rendering unworkable USB devices. We should try to fix it.

For example, when Uninstalling a VCOM device node from Device Manager, the driver files get removed in case we tell Windows to delete driver software.

When USB scanner is plugged in next time, VCOM device shows a yellow exclamation mark.

Device status is reported as "The drivers for this device are not installed. (Code 28)".

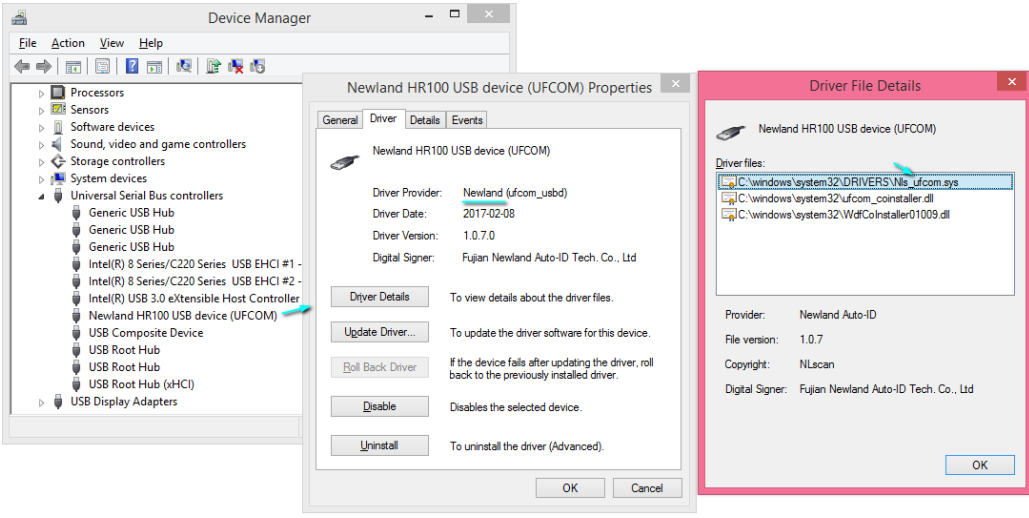
One way to solve it, is executing UFCOM msi again. There is a more concise way, that is, from Appwiz.cpl, right-click UFCOM Driver entry, execute Repair.

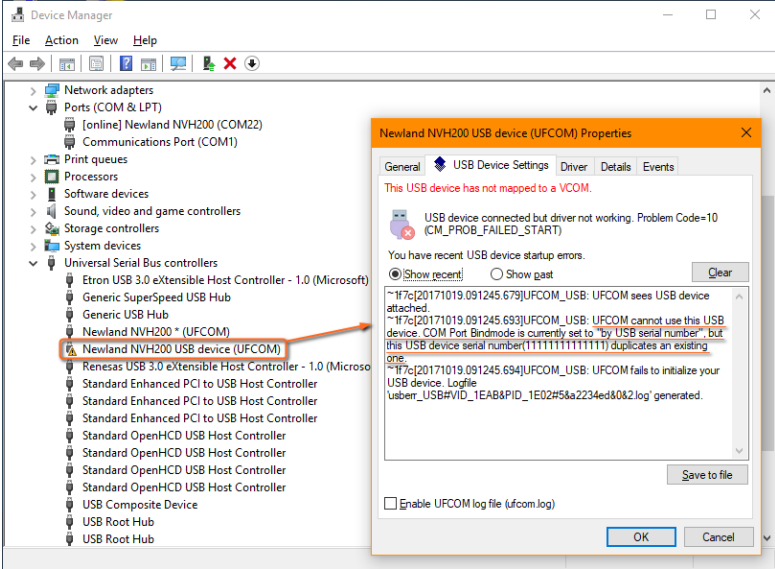


2.3.2. VCOM device is not created (in Device Manager)

There are many causes.

Category	Symptoms, cause and solutions
USB physical device is not detected by Windows	<p>Symptom: In Device Manager, in "Universal Serial Bus controllers" category, you do not see new device node. You do not see any new device node in other device categories either.</p> <p>Possible reasons:</p> <ul style="list-style-type: none"> • Scanner is not actually plugged into Windows machine. • Scanner has not enabled USB function. • Scanner has hardware problem. • Windows does not have correct USB system drivers(USB host controller drivers) installed. <p>Symptom: When scanner is plugged in, Windows task bar shows a popup saying USB Device Not Recognized</p> <p>Possible reasons:</p> <ul style="list-style-type: none"> • Scanner hardware may be malfunctioning. • USB cable problem. <p>The above symptoms share the same problem that Windows itself fails to recognize the USB physical device. At that moment, UFCOM has not been involved yet.</p>

Category	Symptoms, cause and solutions
<p>USB scanner is working as a keyboard</p>	<p>Symptom: VCOM device node does not show up when plugging the scanner, but a text editor(Windows Notepad etc) can show the barcode content.</p> <p>This means: The scanner is working in USB keyboard mode(typically our factory default). Your scanner needs to scan a special setting code to make it transform into USB virtual COM port mode(also known as USB-CDC mode).</p>
<p>Windows detects USB physical device, but uses another driver</p>	<p>Symptom: USB physical device display name does not end with "(UFCOM)".</p> <p>To confirm: Double click USB physical device node to check its driver properties. There should be "Newland" text in it, and Driver Details section refers to driver file Nls_ufcom.sys .</p>  <p>If you find that another driver is taking control of your scanner(Windows stock usbser.sys or the old Datapipe driver), please switch the driver to be UFCOM, as told in former section "Manually choosing a driver package for USB device".</p> <p>Please be aware, if your USB scanner is driven by other drivers, the device node for the physical device does not necessarily appear beneath "Universal Serial Bus controllers" category. It can appear beneath any category which is determined by the driver itself. Well, for USB scanners, most commonly used category is "Ports (COM & LPT)". If you are not sure, you have to plug and unplug your scanner multiple times to verify its appearing location.</p>

Category	Symptoms, cause and solutions
<p>USB physical device is detected, but UFCOM driver does not work correctly</p>	<p>Symptom: USB device icon has a yellow exclamation mark on it.</p> <p>To diagnose: Open USB physical device's property dialog, switch to <u>USB Device Settings</u> tab. Here, we can see USB startup error reasons detected by UFCOM driver.</p> <p>For example, with global option "When Bindmode is by USB serial number, require unique SN" is turned on, plugging in two scanners of the same type and same USB serial number will result in the second scanner failing to work. You will see the following error reporting.</p>  <p>In case similar problems happen, you can send us the error messages, screen shot or saved text, for diagnose.</p>
<p>USB physical device looks OK in Device Manager, but VCOM does not appear.</p>	<p>This is rare. If you encounters this, you can help us diagnose it by sending use UFCOM log file. Enable log file, reproduce the problem, send the log file to us.</p> <p>Way to enable UFCOM log file: Open the property dialog of any USB physical device, tick Enable UFCOM log file (ufcom.log) .</p> <p>Log file location is C:\Windows\Temp\ufcom.log . This file will be cleared and regenerated every time UFCOM driver is loading(e.g., after Windows reboots). So it won't bloat forever.</p> <p>Hint: Everytime you need to check ufcom.log, we suggest make a copy of it(from Explorer, Ctrl+C then Ctrl+V) then view that copy. If you just open ufcom.log itself from a text editor, the text editor will probably not aware of the background update of ufcom.log file content.</p>
<p>VCOM device looks OK, but application does not communicates correctly.</p>	<p>The reasons can be complicated. See next section.</p>

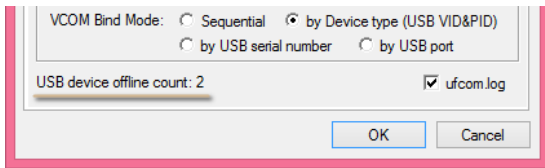
2.3.3. Diagnose USB data communication error

Sometimes, we may encounter data communication error with USB hardware, for example, not able to receive barcode data from VCOM, configuration app(EasySet etc) not able to detect attached scanners, Easyset failing to retrieve captured images etc. Before contacting support staff, we can try to identify the reason. In common sense, communication error may arise in the following segments:

- (A) Hardware/software inside USB Scanner itself.
- (B) Physical link between USB scanner and your PC.
- (C) PC side USB hardware and USB host controller driver(this driver is provided by Windows or host controller manufacturer).
- (D) PC side client driver, i.e. UFCOM.
- (E) The application that uses VCOM.

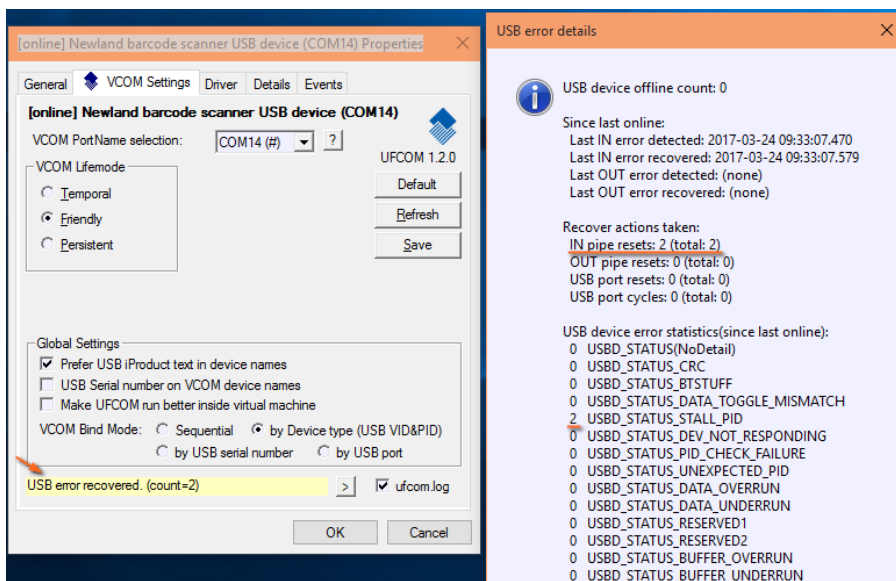
For average users, determine the faulting segment can be hard. Luckily, UFCOM can help you identify whether B is. When using poor quality cable, using buggy USB hub, receiving not enough voltage supply, or encountering electromagnetic interference, we can get faulting condition at B.

VCOM Settings UI's bottom status bar can tell you USB devices offline count. We know that manually unplugging the USB device will increase its count; but if you get increased count while in normal using conditions, it probably means there is problem with USB signals, which is a sign of faulting hardware.



In other circumstances, USB transfer error does not cause "offline", but exhibits a UsbdStatus error code reported to USB host controller driver, and Windows reports that error to UFCOM, UFCOM in turn presents such error to end user through VCOM Settings UI. In a well behaved operation environment, we do not expect to see even a single such error. If we see such an error, it implies some problem in the transfer system, probably in hardware. Sporadic appearing of such error can be recovered by UFCOM automatically, and the VCOM application has a good chance to work smoothly without being interfered; but if it arises too frequently, the application may not work as expected, and we have to go with further troubleshooting.

When UFCOM detects UsbdStatus error, VCOM Settings UI will report it to us. In the following figure, UFCOM detects two such errors, and they have been recovered. Click a small arrow at right-side, we get more error details. We can see that they were two USBD_STATUS_STALL_PID errors.



Be aware, in case the USB link is every unstable, the recovery action may not guarantee the link can become normal, even if UFCOM says "USB error recovered".

Note: The error statistics is per VCOM, that is, each VCOM has its own statistics. That statistics data persist until the VCOM device node is unplugged(disappeared from Device Manager).

If the problem exists, you may need to contact us for help.

3. Advanced Topics

3.1. Export and Import UFCOM settings across different machines

There are two levels of UFCOM settings:

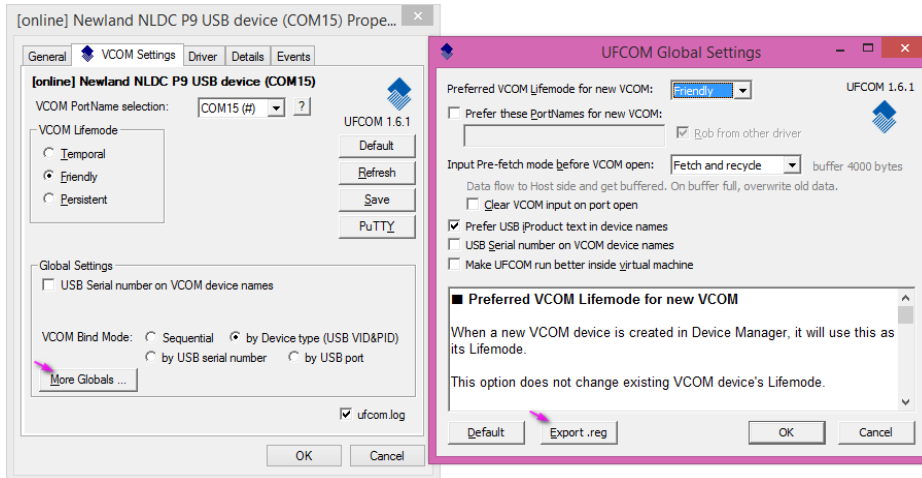
- Global settings. For example: PortName Bindmode, Prefetch mode, whether to display USB serial number in device manager.
- VCOM specific settings. For example: PortName for a VCOM, Lifemode for a VCOM.

Change of a global setting does not affect VCOM specific settings of any existing VCOM.

UFCOM provides facilities to easily export and import global settings. Using this feature, we can transfer global settings from a template machine to other machines quickly and accurately.

Step 1. We need to export settings from the template machine.

Locate any existing VCOM device from Device Manager, open its VCOM settings UI, click **[More Globals...]**, from the UFCOM Global Settings dialog, click **[Export .reg]** button, now we get a generated ucom-global.reg, which is a standard Windows registry file. This file contains all UFCOM global settings from the template machine.



Step 2. Import the global settings into target machine.

Copy ucom-global.reg to the target machine, and double click it. Windows will ask whether you want to import the registry content. Just answer yes.

Step 3. Make the new settings take effect.

After importing the registry file, we need to reload the driver in order for it to take effect. The most straightforward way is to reboot/restart Windows.

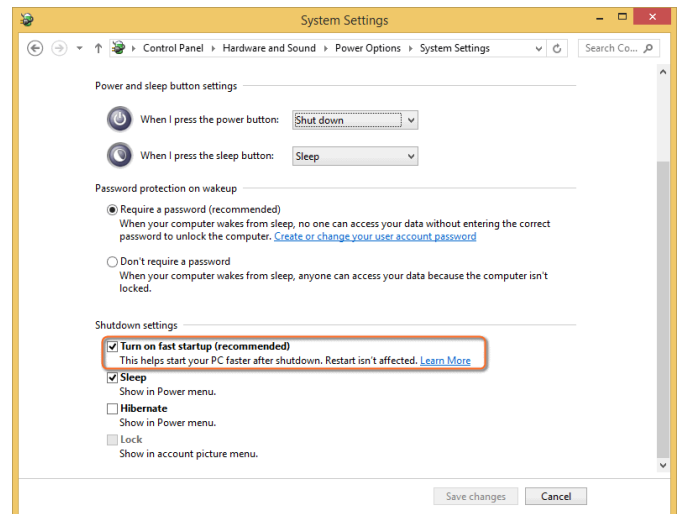


Hint: On Windows 8/10, "shutdown then power-on" is NOT identical to "restart Windows". The reason is: Since Windows 8, Microsoft introduced the so-called Fast-startup feature, which is defaultly enabled. When Fast-startup is enabled, Executing shutdown from Start menu does not do a real shutdown, but essentially do a hibernate. The essence of hibernation is to preserve RAM content to a file named hiberfil.sys. On next time machine power-on, the content in hiberfil.sys is loaded back into RAM and the system runs from where it was left off. So, doing hibernate then powering-on does NOT cause device drivers to reinitialize.

Trick: When Fast-startup is enable, and you wish to do a real shutdown, you can use command line:

```
shutdown /s /t 1
```

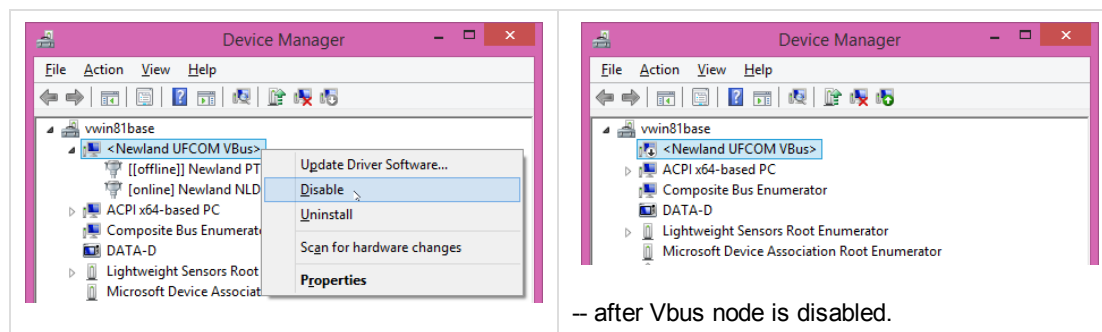
Before doing this, ensure application data have been saved.



If you think rebooting Windows is too time-consuming, there is a shortcut you can use. Do the following:

- Close all applications that could have VCOMs open.
- Unplug all USB devices that are managed by UFCOM.
- Disable UFCOM Vbus device then re-enable it.

Way to disable and enable the Vbus: Open Device Manager, **V**iew → **D**evice**s** by connection, then near the top you can find a device node named **<Newland UFCOM VBus>** , right click to Disable it, then Enable it.



Hint: If you don't switch to view devices by connection, you can still find Vbus node in "System devices" category.

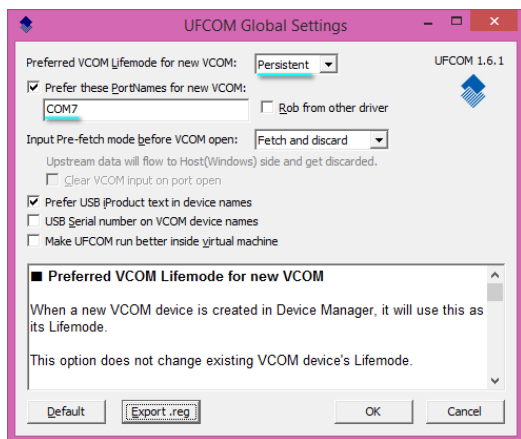
Import global settings at the same time you do a silent install

Now an example. Imagine you want to silent install UFCOM to many new Windows machines, and you expect to achieve two extra goals:

1. First USB scanner on the new machine automatically gets PortName COM7.
2. The newly generated VCOM is in Persistent Lifemode, instead of the default Friendly Lifemode.

Due to the fact that the two extra goals do not match UFCOM's default, you need the help of global settings registry file.

First set up your preferred global settings on your template machine. A special note: You need to set up **"Prefer these PortNames for new VCOM"**(we call it "GPV Portname list" for simplicity) instead of PortName for a specific VCOM.



Export ufcom-global.reg, copy it alongside UFCOM msi to your target machine. On the target machine, first import the registry file, then launch msi installation, and your goal is achieved.

To make it silent, you can use a small batch file like this.

```
@REM silent-deploy.bat
@set batdir=%~dp0
@set batdir=%batdir:~0,-1%

reg import "%batdir%\ufcom-global.reg"

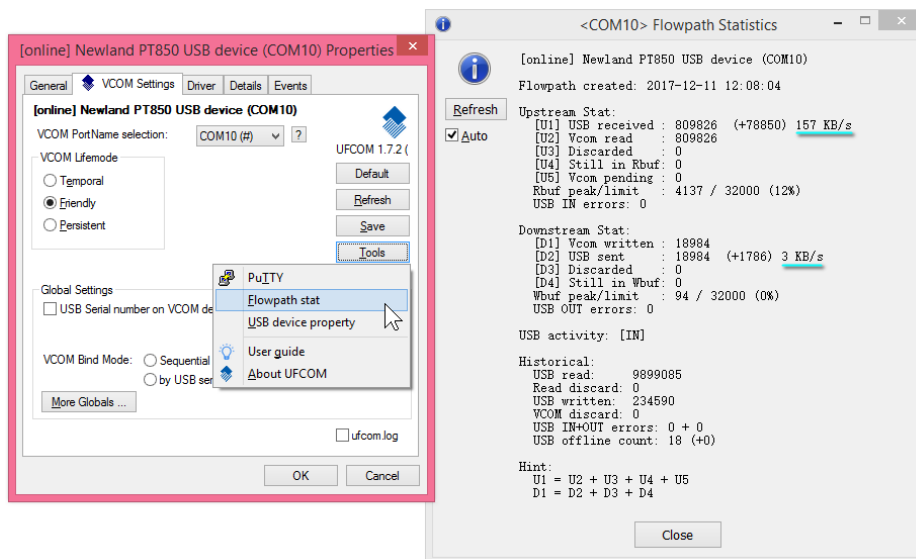
start /wait msixec /q /i "%batdir%\ufcom-1.6.1.msi"
```

Be aware, you need to run this batch file with Administrative right to make it success.

3.2. Viewing USB data flow statistics

Flowpath Stat feature present us with USB upstream and downstream data flow statistics, including byte count of current flowpath session, historical total byte count, USB error count, realtime dataflow speed etc. This help us determine whether data transfer problems lie in application or the driver/hardware.

View entrance: VCOM Settings UI → Tools → Flowpath stat .



Explanation of each item(from the perspective of Windows PC):

Direction	Item	Explanation
Upstream	U1. USB received	Byte count received from USB data line(not including USB protocol fields). For example, scanner scans a 13-byte barcode, and this value will increase by 13. The stat info is refreshed every 0.5 second (with Auto ticked). In the sample image above, (+78850) means this value is increased by 78850 compared to last refresh. The 157KB/s at tail means the calculated USB dataflow speed is 157 KBytes per second at this moment.
	U2. Vcom read	Byte count that has been read by ReadFile API.
	U3. Discarded	Byte count that has been received from USB data line, but get discarded. For example, when Prefetch mode is set to fetch-and-discard, all bytes received from USB line is discarded before the application opens VCOM for reading.
	U4. Still in Rbuf	Byte count that has been received from USB data line, but has not been read by application. These bytes stays temporarily in Rbuf. If application closes VCOM while there are still bytes in Rbuf, these bytes will be counted as discarded.
	U5. Vcom pending	Byte count that has been received from USB data line, copied to application buffer, but not yet returned to ReadFile API caller. For example, assume Rbuf is initially empty, ReadFile request reading 100 bytes(requiring a full read, would not return with partial data). Now, USB device sends 60 bytes, then the 60 bytes will count as U5. When later ReadFile completes, this 60-byte count will transfer into U2.
	Rbuf peak/limit	Rbuf usage peak and size limit. Peak value will not exceed limit value. Rbuf size can be adjusted by SetupComm API call, whose default value is 32000. For one VCOM open period, Rbuf size can be assigned only once.
	USB IN errors	USB upstream errors detected.

		Appearance of this value does not necessarily mean the application has received wrong USB data. USB error recovering mechanism may help correct such errors and the application may still get correct data.
Downstream	D1. Vcom written	Successful byte count reported by WriteFile API.
	D2. USB sent	Byte count successfully sent out of USB data line.
	D3. Discarded	<p>Byte count that WriteFile has reported success, but later discarded by UFCOM. Discard means not sending those bytes out of USB line.</p> <p>There is one case that this Discard value will increase. As mentioned above,</p> <p>When WriteTotalTimeoutMultiplier=0 , WriteTotalTimeoutConstant=0 , and <u>output flow is not enabled</u>. UFCOM guarantees WriteFile will not block forever. If scanner is unplugged or downstream flow blocked, WriteFile will complete with success after there is no downstream flow for continuous 1 second. This is to simulate that WriteFile to a on-board physical port will not block forever.</p> <p>In this situation, the bytes reported success will be counted as D3.</p> <p>If application closes VCOM while there are still bytes in Wbuf, these bytes will be counted as discarded.</p>
	D4. Still in Wbuf	Byte count that has been reported success by WriteFile, not discarded, and queuing for output to USB line. These bytes are queuing in a memory block called Wbuf.
	Wbuf peak/limit	<p>Wbuf peak usage and size limit. Peak value will not exceed limit value.</p> <p>Wbuf size can be adjusted by SetupComm API call, whose default value is 32000. For one VCOM open period, Wbuf size can be assigned only once.</p>
	USB OUT errors	<p>USB downstream errors detected.</p> <p>These errors can also be corrected by USB error recovering mechanism.</p>
-	Historical	<p>Historical total values of above items. Even if Windows reboots, historical totals are still preserved – because historical values are recorded in Windows registry.</p> <p>"USB offline count" has two values. The first one is the historical total offline count. The one inside bracket is the offline count within current flowpath session, and it will be merged into historical total when flowpath object is destroyed.</p>
-	USB activity	<p>This reflects USB physical line status. Four possible cases:</p> <ol style="list-style-type: none"> 1. (blank) 2. [IN] 3. [IN][OUT] 4. [OUT] <p>[IN] means IN data packet now appears on USB data line. [OUT] means OUT data packet now appears on USB data line.</p> <p>After VCOM has been opened by application, you normally see continuous IN, because USB Host is always querying USB Slave for newer data. If PC-side Rbuf is full(due to application cease reading data), IN packet will disappear.</p> <p>If application is not sending out data continuously while you see OUT persists, it means downstream link is suffering from congestion, that is, USB device does not retrieve downstream data in a timely manner.</p>

Other features:

- Un-tick Auto, and we get manual refresh mode. One click of Refresh button updates display once.
- Mouse right-click a blank area near left bottom or right-bottom, we can switch thousand separator on or off, copy text to clipboard etc.

What is a flowpath? and how it effects Flowpath Stat values

Simply speaking, one USB physical device and one corresponding VCOM constitute a flowpath.

A flowpath is destroyed when **both** the physical device and VCOM device are unplugged from system. During a flowpath object's life time, stat values are counted into session values; on flowpath's destruction, those values are merged into historical totals.

3.3. msi installer generated registry items

UFCOM msi installation generates following registry keys and items in your HLKM node.

32-bit Windows	HKLM\SOFTWARE\Newland Auto-ID\UFCOM
64-bit Windows	HKLM\SOFTWARE\Wow6432Node\Newland Auto-ID\UFCOM

Registry items inside:

Item name	Item value (sample)	Explain
Installed	1	This item means UFCOM msi has been installed on your system.
Version	1.7.0	Installed UFCOM version on your system.
InstallTargetDir_	C:\Program Files\UFCOM\	Tells which directory has UFCOM installed. NOTE: There is a trailing backslash. This is the same as Appwiz.cpl reported Location field.

System integrator can check these registry items to know whether UFCOM has been installed. This helps them decide whether to install UFCOM for end-users.


4. VCOM device API behavior

UFCOM implements Windows COM port API(serial port API) quite thoroughly.

If you program with Windows COM port API, you need to know information in this section.

API category	Implementation note
Dataflow integrity	<p>For upstream data, UFCOM guarantees data integrity. For example: USB device send lots of upstream data, PC application does not fetch these bytes timely, and the UFCOM upstream buffer finally gets full. In this situation, UFCOM will stop fetching data from USB device, instead of discarding received bytes. As soon as some bytes are fetched by PC application, the upstream data resume flowing.</p> <p>For downstream data, there are two cases:</p> <ul style="list-style-type: none">• If PC application does not enable downstream flow-control(the default behavior), data written by application may get silently discarded. The discarding behavior typically happens in the following situation: USB device has been unplugged, or, USB device does not promptly pull out the bytes sent from PC (each chunk of written data are preserved for at least 1000 milliseconds defaultly).• If PC application does enable downstream flow-control, UFCOM will guarantee data integrity, that is, UFCOM will not discard any downstream, which means WriteFile may block infinitely.
RS-232 parameter simulation	<p>No simulation for this. User can set any baudrate, but the value is simply ignored. Actually communication speed is determined by the USB system.</p> <p>Data bits, stop bits, parity settings are all ignored.</p>

API category	Implementation note
Modem signal simulation	<p>Output signals: RTS, DTR setting is ignored.</p> <p>Input signals: GetCommModemStatus always report RING and RLSD off . DSR and CTS is simulated.</p> <p>DSR: When scanner is plugged in, DSR=On ; when unplugged, DSR=Off .</p> <p>CTS: When scanner is plugged in and down stream is not blocked, CTS=On ; when scanner is unplugged or down stream is blocked, CTS=Off .</p>
hardware flow control	<p>UFCOM does not support upstream(input) flow control with simulated modem signal. Upstream flow control is entirely determined by USB system.</p> <p>UFCOM supports downstream(output) flow control.</p> <ul style="list-style-type: none"> • When VCOM is first opened by application, flow control is not enabled; • After application calls SetCommState() with DCB.fOutxCtsFlow=1 , the downstream flowing can be controlled, e.g., If CTS=Off, WriteFile may gets blocked. Be aware, If user sets COMMTIMEOUT.WriteTotalTimeoutMultiplier==0 and COMMTIMEOUT.WriteTotalTimeoutConstant==0 , it is possibly that WriteFile will get blocked forever(in case CTS remains off forever). To complete the blocking/pending WriteFile, user has to call Canceled.
software flow control	<p>XON/XOFF flow control is not supported.</p>
WriteFile behavior	<p>UFCOM VCOM's WriteFile accepts user data atomically. That is, WriteFile will complete with <u>all requested data</u> or <u>no data</u>; you will not see partial data accepted.</p>
COM port timeouts	<p>Implement COMMTIMEOUTS semantic completely.</p> <p>There are some special designs:</p> <ul style="list-style-type: none"> • When WriteTotalTimeoutMultiplier=0 , WriteTotalTimeoutConstant=0 , and <u>output flow is not enabled</u>. UFCOM guarantees WriteFile will not block forever. If scanner is unplugged or downstream flow blocked, WriteFile will complete with success after there is no downstream flow for continuous 1 second. This is to simulate that WriteFile to a on-board physical port will not block forever. • When WriteTotalTimeoutMultiplier=0 , WriteTotalTimeoutConstant=0 , and <u>output flow is enabled</u>, WriteFile will block before all data is written out. Of course, user can call Canceled to cancel WriteFile prematurely. • When WriteTotalTimeoutMultiplier or WriteTotalTimeoutConstant is non-zero, API-assigned timeouts will be effective, regardless whether flow control is enabled.

API category	Implementation note										
CommEvents	<p>UFCOM implements four notification events.</p> <table border="1" data-bbox="266 218 1531 632"> <thead> <tr> <th data-bbox="266 218 451 268">Event</th> <th data-bbox="451 218 1531 268">When to notify</th> </tr> </thead> <tbody> <tr> <td data-bbox="266 268 451 384">EV_RXCHAR</td> <td data-bbox="451 268 1531 384">When some bytes come to input buffer, EV_RXCHAR event arises. Once this event is reported, it will not be reported again unless new bytes comes to input buffer; on the other side, when some byte is received in input buffer, EV_RXCHAR will definitely arise one more time.</td> </tr> <tr> <td data-bbox="266 384 451 527">EV_TXEMPTY</td> <td data-bbox="451 384 1531 527">When output buffer becomes empty, this event arises. It's edge-triggered, that is, after EV_TXEMPTY is reported once, waiting this event again will not report a second EV_TXEMPTY; it is reported again only after some bytes are filled into output buffer by WriteFile and becomes empty again.</td> </tr> <tr> <td data-bbox="266 527 451 577">EV_DSR</td> <td data-bbox="451 527 1531 577">EV_DSR is edge-triggered, representing simulated DSR change.</td> </tr> <tr> <td data-bbox="266 577 451 632">EV_CTS</td> <td data-bbox="451 577 1531 632">EV_CTS is edge-triggered, representing simulated CTS change.</td> </tr> </tbody> </table> <p>UFCOM allows multithreaded operation. You can call WaitCommEvent in one thread, and ReadFile/WriteFile in another thread.</p> <div data-bbox="266 741 1531 879" style="border: 1px solid #add8e6; padding: 5px;"> <p> UFCOM author note: Making EV_RXCHAR and EV_TXEMPTY edge-triggered(as opposed to level-triggered) is not a good idea, though, Microsoft designed Serial.sys (for physical port COM1) this way, UFCOM follows it for better application code compatibility.</p> </div>	Event	When to notify	EV_RXCHAR	When some bytes come to input buffer, EV_RXCHAR event arises. Once this event is reported, it will not be reported again unless new bytes comes to input buffer; on the other side, when some byte is received in input buffer, EV_RXCHAR will definitely arise one more time.	EV_TXEMPTY	When output buffer becomes empty, this event arises. It's edge-triggered, that is, after EV_TXEMPTY is reported once, waiting this event again will not report a second EV_TXEMPTY; it is reported again only after some bytes are filled into output buffer by WriteFile and becomes empty again.	EV_DSR	EV_DSR is edge-triggered, representing simulated DSR change.	EV_CTS	EV_CTS is edge-triggered, representing simulated CTS change.
Event	When to notify										
EV_RXCHAR	When some bytes come to input buffer, EV_RXCHAR event arises. Once this event is reported, it will not be reported again unless new bytes comes to input buffer; on the other side, when some byte is received in input buffer, EV_RXCHAR will definitely arise one more time.										
EV_TXEMPTY	When output buffer becomes empty, this event arises. It's edge-triggered, that is, after EV_TXEMPTY is reported once, waiting this event again will not report a second EV_TXEMPTY; it is reported again only after some bytes are filled into output buffer by WriteFile and becomes empty again.										
EV_DSR	EV_DSR is edge-triggered, representing simulated DSR change.										
EV_CTS	EV_CTS is edge-triggered, representing simulated CTS change.										
Cancello	<p>Fully supported. User can Cancello ReadFile, WriteFile, WaitCommEvent.</p> <p>Cancello will succeed immediately, no blocking.</p> <p>Special note: If VCOM has received some bytes when ReadFile is cancelled, ReadFile will complete with success and report those already received bytes. This behavior is better than that of serial.sys and usbser.sys. The serial.sys and usbser.sys merely reports error result 995(ERROR_OPERATION_ABORTED) for ReadFile on cancelling.</p>										
VCOM TX & RX buffer size	<p>For each VCOM, UFCOM creates a pair of TX/RX internal buffer, defaultly 32000 bytes each. This means, a single WriteFile can accept at most 32000 bytes. If requested bytes exceeds this size, application gets ERROR_INVALID_PARAMETER(87) error code. To cope with this problem, you can choose between three ways:</p> <ol style="list-style-type: none"> 1. Modify application code. Break a large WriteFile call into multiple small WriteFile calls. 2. Modify application code. After CreateFile returns a VCOM handle, call SetupComm to enlarge TX queue. 3. If application source code is not available, you can change a UFCOM global registry value TxQueueDefaultSize to get a bigger TX buffer. After modifying this setting, UFCOM driver needs to restart to take effect. <p>For example, to change TxQueueDefaultSize to 1024000, you can import such a registry file:</p> <div data-bbox="266 1436 1531 1598" style="border: 1px dashed #ccc; padding: 10px;"> <pre>Windows Registry Editor Version 5.00 [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Nls_ufcom\Parameters] "TxQueueDefaultSize"=dword:000fa000</pre> </div> <p>Similarly, changing TxQueueDefaultSize assigns customized default RX queue size.</p>										

【 Special note for Temporal Lifemode 】

For Temporal Lifemode VCOM, corresponding USB physical device unplugging will render the VCOM handle ineffective, which means, calling ReadFile, WriteFile, WaitCommEvent on such ineffective handle will report error immediately – even if USB device has been re-plugged in. In order to recover communication with the USB device, you have to close the old handle and call CreateFile to get a new one.

5. UFCOM-specific API guide

This section is also for application developer. Applications can utilize these extra functionality to improve user experience.

Windows COM port API provides a set of abstract data input/output model for applications, and they work well in many scenarios. However, these APIs are designed before USB specification is published, so it does not take concern about USB's hot-plugging behavior, and there is no standard semantics for USB device plugging and unplugging. Therefore, UFCOM needs to make its own specification.

5.1. Test UFCOM API with vcomtest program

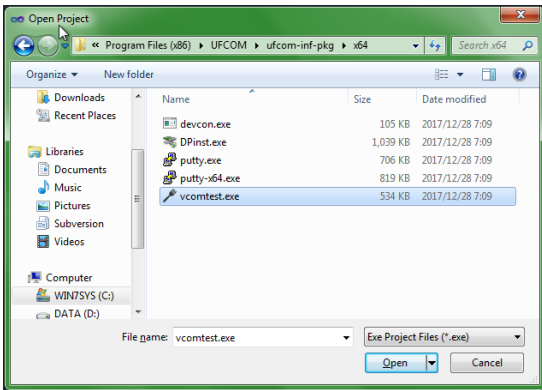
UFCOM implements most of Windows COM port API semantics, and we provide vcomtest to verify these semantics, as well as demonstrating UFCOM-specific features. We provide you pre-compiled vcomtest.exe, and its key-part source code with pdb file. So, our users can debug its source code in Visual Studio to know what work it does. Ugh, you just cannot compile that code yourself, because some headers and libraries are absent.

In order to debug vcomtest, you need Visual Studio 2005 or above.

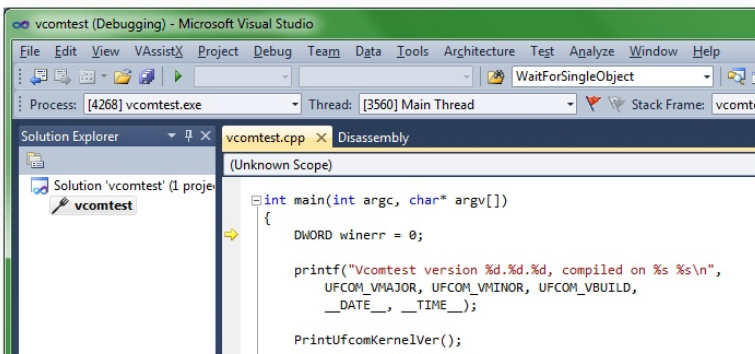
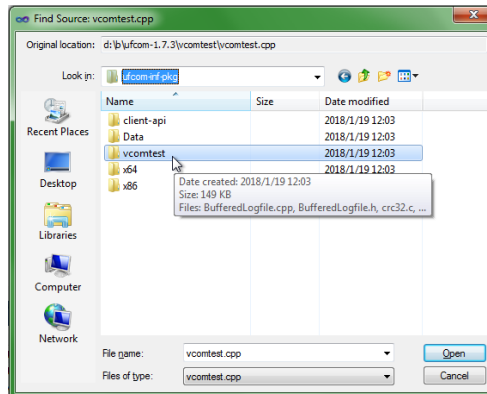
Now we show you the process with UFCOM 1.7.3 and Visual Studio 2010. We use \$UFCOM to represent UFCOM's install target directory.

Open Visual Studio IDE, File → Open → Project/Solution, tell it to open \$UFCOM\ufcom-inf-pkg\x64\vcomtest.exe (on 32-bit Windows, you should open the one in x86 directory. Note: we open vcomtest.exe as a project here, not as a normal file. VS IDE menu, Debug → Step Over(F10) starts debugging. You may see a dialog box asking whether you need privilege elevation, answer No is OK. Then, a **Find Source** dialog box pops out asking for vcomtest.cpp, browse to directory \$UFCOM\ufcom-inf-pkg\vcomtest and you're done. If everything goes smoothly, you should see debugger's current statement, the yellow arrow, stops at main(). Enjoy your debugging.

Open vcomtest.exe as a project.



Open vcomtest.exe as a project.



5.2. Detect USB device plugging and unplugging

Device plugged-in state corresponds to VCOM's online state shown in Device Manager, unplugged state corresponds to offline.

Windows API itself does not define how to represent online/offline state, so, many virtual COM port device drivers(besides UFCOM) indicates device's offline state by making the device handle ineffective. The handle here is the one returned by CreateFile. The word ineffective means, further ReadFile, WriteFile, WaitCommEvent on that handle will return failure immediately; a pending overlapped I/O operation will also be completed as soon as the handle becomes ineffective. This is an effective way of indicating device offline. For UFCOM, a VCOM of Temporal lifemode follows this behavior as well.

The special feature of UFCOM is, Friendly lifemode and Persistent lifemode provides durable device handle, that is, device unplugging will not cause device handle to become ineffective, which is convenient for applications in many circumstances. But in case applications want to detect device online/offline state change, for example, to a offline prompt on user interface, how do they achieve that?

UFCOM specifies it as follows: Use the DSR signal to indicate online/offline. DSR on means online, DSR off means offline. DSR is a physical line signal from RS-232 communication protocol which is used by physical serial port. This signal is controlled by communication peer. USB does not have that DSR signal, so UFCOM simulates that signal to indicate device online/offline. Applications can use **WaitCommEvent** to wait for EV_DSR event, so that DSR's change is known immediately.

Please be aware, EV_DSR event merely notifies DSR state change since last notification, it does not tell you whether DSR is on or off currently. You need to call **GetCommModemStatus** to query its real state. Now and example to clarify this. User code calls WaitCommEvent twice to wait for EV_DSR, and during the course, the following affairs happen in succession:

1. First WaitCommEvent returns due to DSR becomes off.
2. In very short time, DSR becomes on then off again.
3. User calls WaitCommEvent a second time.

Question: Will the second WaitCommEvent report EV_DSR immediately?

The answer is Yes. Reason: since "previous EV_DSR report", DSR has changed at least once, no matter how many times. "previous EV_DSR report" signifies the time point that WaitCommEvent returns changed-state to caller. In other word, as long as user code calls GetCommModemStatus(upon receiving EV_DSR report), user code can track DSR's final state correctly.

vcomtest guide: Use vcomtest to open a Friendly lifemode or Persistent lifemode VCOM , press keyboard **E** to launch a background thread. This thread calls WaitCommEvent in a cycle. Now, plugging and unplugging a USB device will cause it to generate EV_DSR events, and you will see those report on vcomtest console.

Press **e** to stop WaitCommEvent background thread.

In former example, I tell you that GetCommModemStatus may report same EV_DSR value for two consecutive WaitCommEvent return. This may upset you wondering whether one of them is a fake report. Don't worry, there is another way to assure you online/offline has really happened. Use UFCOM-specific **IOCTL_VCOM_GetDeviceState** to query VCOM device state, and the returned info will tell you the recent time point when the USB is online and offline. By comparing the timestamps, you get the truth.

```
IOCTL_VCOM_GetDeviceState_ost vcomstate = {};  
DWORD retbytes = 0, insize = sizeof(vcomstate);  
BOOL b = DeviceIoControl(hCom, IOCTL_VCOM_GetDeviceState, NULL, 0,  
    &vcomstate, insize, &retbytes, NULL);  
// check vcomstate members for details.
```

vcomtest guide: Use vcomtest to open a VCOM, press **i** to query current VCOM state.

Now unplug your USB scanner and re-plug in, query with **i**, you'll see uemsecUsbdLastOnline is behind uemsecUsbdLastOffline.

```
Modem lines: CTS on,DSR on, DCD off,RING off  
IOCTL_VCOM_GetDeviceState query result:  
  .lifemode : Friendly<1>  
  .isUsbdOnline: yes  
  .uemsecUsbdLastOnline : 2018-01-19 16:11:38.663  
  .uemsecUsbdLastOffline: 2018-01-19 16:11:35.657  
  .szUcomDevinstpath: NLS_USB\Nls_vcom\hid.0  
  .szUsbdDevinstpath: USB\VID_1EAB&PID_0501\135456  
  
<4>Again? 'R/r' read, 'U/u' write. More: 0,X,i,P,p,F,f,E,e,C,c,S,S.#,?
```

5.3. Use one-time Temporal feature to get instant USB unplugging notification

Background information: UFCOM's Friendly lifemode for VCOM brings so much convenience for applications and end-users. However, under some special circumstances, application code(not the end-user) may hope to get instant notification when USB device is unplugged(i.e. VCOM becomes [offline]). For example, during USB device firmware upgrading, USB device may go through a reboot process, and during the time frame it is rebooting, VCOM exhibits offline state. So the PC-side firmware upgrading program hopes to capture this short time frame and take special actions appropriately. There may be two ways to workaround this:

- Method one: Change VCOM's lifemode to Temporal, so that, when USB device is unplugged, the VCOM handle immediately becomes ineffective and the application code knows it immediately as well, and later read/write on this handle also fails. However, this requires the end-user to switch VCOM lifemode to Temporal before running firmware upgrade program, and then switch back to friendly lifemode, quite inconvenient.
- Method two: Using WaitCommEvent to wait for EV_DSR's change. The problem is, if you have organize your code in a WriteFile/ReadFile cycle, the code structure may need significant update to utilize WaitCommEvent, not convenient as well.

From UFCOM 1.7.3 on, there is a much better way. The application code can send `IOCTL_VCOM_OneTimeTemporalLifemode_Enable` control code to VCOM, so the VCOM enters one-time Temporal state. From this time one, as soon as USB device becomes offline, the corresponding VCOM handle will become ineffective immediately, just like that of a real Temporal lifemode.

The code to enable one-time Temporal feature is:

```
HANDLE hcom = CreateFile("\\\\.\\COM3", ...);

DWORD retbytes = 0;
BOOL bSucc = DeviceIoControl(hcom, IOCTL_VCOM_OneTimeTemporalLifemode_Enable,
    NULL, 0, // input
    NULL, 0, // output
    &retbytes, NULL);

if(bSucc)
    printf("Success.\n");
else
{
    DWORD winerr = GetLastError(); // ERROR_INVALID_FUNCTION if UFCOM version too old
    printf("Fail.\n");
}
```

More over, you can send `IOCTL_VCOM_OneTimeTemporalLifemode_Disable` to disable one-time Temporal, state, send `IOCTL_VCOM_OneTimeTemporalLifemode_Query` to query current one-time Temporal state. See coding detail in `vcomtest` source code.

Some more hints:

- The one-time Temporal state is enabled only for current VCOM only. In other word, if you close VCOM handle then re-open it, one-time Temporal state is automatically reset to disabled.
- If the USB device has been offline when you send `IOCTL_VCOM_OneTimeTemporalLifemode_Enable`, the VCOM handle will become ineffective right now.
- When USB device is re-plug in, its VCOM lifemode remains as usual, that is, a Friendly VCOM is still Friendly. That means, one-time Temporal never disturbs you when not needed.

One-time Temporal feature also applies on Persistent VCOM lifemode.

vcomtest guide: After vcomtest opens a VCOM,

- Press < to execute IOCTL_VCOM_OneTimeTemporalLifemode_Enable
- Press > to execute IOCTL_VCOM_OneTimeTemporalLifemode_Disable
- Press | to execute IOCTL_VCOM_OneTimeTemporalLifemode_Query

For a Friendly lifemode VCOM , after executing IOCTL_VCOM_OneTimeTemporalLifemode_Enable ,

- Unplug the device, then press r to do ReadFile once, we get ERROR_BAD_COMMAND .
- Call ReadFile first to wait for incoming data(pending read), then unplugging the device will cause pending ReadFile to complete immediately. The completion result depends on whether we have receive some data. If no a single byte has received, we got error ERROR_DEVICE_NOT_CONNECTED; but if some data has arrived(at UFCOM Rbuf), ReadFile will complete with success reporting those received bytes. ReadFile again will report failure – if all bytes have been retrieved.
- A pending WaitCommEvent will also report EV_CTS, EV_DSR and EX_RXCHAR change on device unplugging, calling again will return failure immediately.

Note: Windows error code on device unplugging can be either ERROR_DEVICE_NOT_CONNECTED or ERROR_BAD_COMMAND, or some other value. For any unrecognized error value, application code should always consider the handle ineffective.

5.4. How to determine whether a COM port device is from UFCOM

Two ways.

【 The easy way 】 Obtain the VCOM device handle first, then send it IOCTL_VCOM_OneTimeTemporalLifemode_Query(0x16127408). If DeviceIoControl reports success, we consider the VCOM driven by UFCOM.

【 The hard way 】 In this way, we can know the answer before a VCOM handle is obtained. To use this method, you need to use the SetupDi... functions to enumerate all COM port devices from the system, querying their SPDRP_SERVICE property. If the property value is **Nls_ufcom**, it must be a UFCOM VCOM device.

vcomtest guide: Execute vcomtest -l will enumerate all COM port devices from the system, and print there SPDRP_SERVICE property value.

```
c:\Windows\System32\NlsUfcom>vcomtest -l
vcomtest version 1.7.4, compiled on Jan 19 2018 15:22:00
UFCOM kernel version: 1.7.4
<#1>
  Friendly name: Communications Port <COM1>
  Service name : Serial
  DevOpenPath  : "\\?\acpi#pnp0501#1#<86e0d1e0-8089-11d0-9ce4-08003e301f73>"
  DevInstPath  : ACPI\PNP0501\1
<#2>
  Friendly name: Communications Port <COM2>
  Service name : Serial
  DevOpenPath  : "\\?\acpi#pnp0501#2#<86e0d1e0-8089-11d0-9ce4-08003e301f73>"
  DevInstPath  : ACPI\PNP0501\2
<#3>
  Friendly name: [online] Newland PT850 USB device <COM8>
  * Service name : Nls_ufcom
  DevOpenPath  : "\\?\nls_ubus#nls_ufcom#hid.0#<86e0d1e0-8089-11d0-9ce4-08003e301f73>"
  DevInstPath  : NLS_UBUS\NLS_UCOM\HID.0
  HwDevInstPath: USB\VID_1EAB&PID_0501\135456
Total 3 UCOM devices found.
```


Newland EMEA
+ 31 (0) 345 87 00 33
info@newland-id.com

Rolweg 25
4104 AV Culemborg
The Netherlands

Need more info?
Contact us or one of our partners at
newland-id.com/contact

