

General

UTS Drivers DLL Specification

DWS – 780436-0000

Revision A6

SW: Bird Hsieh

Logitech Far East Ltd.

LOGITECH CONFIDENTIAL

**Revision and history:**

# Table of Content

[**Revision and history:** 2](#_Toc42174779)

[1 Table of Content 3](#_Toc42174780)

[2 History 5](#_Toc42174781)

[3 Purpose 10](#_Toc42174782)

[4 DLL Directory 11](#_Toc42174783)

[5 Command List 12](#_Toc42174784)

[5.1 General 12](#_Toc42174785)

[4.1.1 Initialize 13](#_Toc42174786)

[4.1.2 Release 14](#_Toc42174787)

[4.1.3 SetDebugFlag 14](#_Toc42174788)

[4.1.4 DevExist 14](#_Toc42174789)

[4.1.5 GetDevInformation 15](#_Toc42174790)

[4.1.6 SetDevStatus 16](#_Toc42174791)

[4.1.7 SetDriverMode 17](#_Toc42174792)

[5.2 Buffer Control 18](#_Toc42174793)

[4.2.1 (Buffer) Open 19](#_Toc42174794)

[4.2.2 (Buffer) Close 19](#_Toc42174795)

[4.2.3 (Buffer) Clear 20](#_Toc42174796)

[4.2.4 (Buffer) Read 20](#_Toc42174797)

[4.2.5 (Buffer) Setpattern 21](#_Toc42174798)

[4.2.6 (Buffer) SearchData 24](#_Toc42174799)

[4.2.7 (Buffer) SetFilter 24](#_Toc42174800)

[4.2.8 (Buffer) ClearDevIDFamily 24](#_Toc42174801)

[4.2.9 (Buffer) ClearDevIDSpecific 25](#_Toc42174802)

[5.3 USB Section 26](#_Toc42174803)

[4.3.1 (USB) OpenDevice 27](#_Toc42174804)

[4.3.2 (USB) OpenDeviceByDeviceType 28](#_Toc42174805)

[4.3.3 (USB) CloseDevice 29](#_Toc42174806)

[4.3.4 (USB) VendorCommand 29](#_Toc42174807)

[4.3.5 (USB) USBRequest 30](#_Toc42174808)

[4.3.6 (USB) HID++Request 30](#_Toc42174809)

[4.3.7 (USB) EndPointRequest 31](#_Toc42174810)

[4.3.8 (USB) ExeParseDLL 32](#_Toc42174811)

[4.3.9 (USB) GetEndpointID 34](#_Toc42174812)

[4.3.10 (USB) GetIDByEpNo 35](#_Toc42174813)

[4.3.11 (USB) GetUSBPortTree 35](#_Toc42174814)

[4.3.12 (USB) GetUSBPortDevIDByCOMNumber 36](#_Toc42174815)

[5.4 PS2 Section 37](#_Toc42174816)

[4.4.1 (PS2) OpenDevice 37](#_Toc42174817)

[4.4.2 (PS2) CloseDevice 37](#_Toc42174818)

[5.5 Extra function 38](#_Toc42174819)

[6 DLL Installation Guide 39](#_Toc42174820)

[6.1 UTS\_Drivers DLL Installation 39](#_Toc42174821)

[7 Appendix A: Exported functions in UTS\_Drivers 42](#_Toc42174822)

# History

| **Rev** | **Description** | **Modified by** | **Date** |
| --- | --- | --- | --- |
| A6 | V2.0.0.0   1. Add Max Record Length parameter to UD\_InitializeDriver and Exe\_Initialize function. 2. Add UD\_BufferSetPatternByName function   Now it support open or close specific phase by string and the origin method still exist. UTS still call Exe\_TUBuffer\_SetPattern but give one more parameter to choose whether use it or not.   1. Add HIDPlusPlus\_GetFirstRecInBuffer function to distinguish the buffer read between USBRequest and HID++. 2. Add New Respond DevID Parameter to SendHIDPlusPlusRequestByString function to support multiple test Use Commad ~DevID to replace DevID = -2 store in BufferPointerArray 3. Add ClearDevIDFamily and ClearDevIDSpecific function to support buffer clear by Device family or Device specific. Both of them support UTS and PDViewer. 4. Add Exe\_OpenDeviceByDeviceType function to support open device by its’ type. Remove origin part in Exe\_TU\_OpenDevice. Now it is an independent function. 5. Add GetUSBPortTree and GetUSBPortDevIDByCOMNumber function to support get port tree by Device ID or by COM Number. Both of them support UTS and PDViewer. 6. Add UD\_OpenAllDevByVIDPID function to support open devices in multiple test 7. Add a parameter to ExeParseDLL function   This parameter is the whole route of txt file which is used to replace the data read from bushound. | Danny Yeh | 08/29/2014 |
| A5 | V1.0.3.3  1. Add SetDevStatus function.  V1.0.3.2  1. Add GetDevInformation function.  V1.0.2.31  1. In SendUSBRequestByString, increase the output command buffer to 1024 bytes.  2. Sove bug: In SendUSBRequestByString function, put command only for command byte not for all the data buffer  V1.0.2.30  1. Update a bug for GetStrParameterByIndex(int iIndex, char \* Para, int Para\_Len)  V1.0.2.29 [8-Apr-05 BiHs]  1. Bug ->GetFirstRecInBuffer --> if IfExportTimeStamp=2, solve the bug that high 4 bytes is '00'  V1.0.2.28 [2-Apr-05 BiHs]  1. Bug after 1.0.2.24 --> Exe\_TUBuffer\_Open --> If Para5 >1 2 , then set it to 0. ( 2 means export 8 bytes delta time)  V1.0.2.27 [18-Mar-05 BiHs]  1. SendHIDPlusPlusRequestByString -->Let buffer pointer (-1) to the end of buffer  2. UD\_BufferOpen --> remove reset buffer pointer activity  V1.0.2.26 [16-Mar-05 BiHs]  1. SendHIDPlusPlusRequestByString --> remove buffer open/clear/close control  2. GetFirstRecInBuffer --> If DevID=-2, then read data without DevID check  3. SendHIDPlusPlusRequestByString --> check HID++ error response XX ?? 8F ?? XX...  V1.0.2.25 [9-Mar-05 BiHs]  1. GetDevIDbyPIDVID --> solve a bug which the DevID is smaller than the previous opened one.  2. GetDevIDbyPIDVID --> New Command!!  3. Exe\_TUBuffer\_Open --> If Para5 >1 , then set it to 0. (Some old test plan put the wrong parameter here) | Bird Hsieh | 07/15/2005 |
| A4 | V1.0.2.23 [10-Jan-05 BiHs]  1. GetFirstRecInBuffer --> if Buffer pointer is out of range, it would set buffer pointer to the begin (end) pointer of buffer. (depends on the buffer pointer is too large (the end of pointer ) or too small (the begin of pointer))  V1.0.2.22[3-Jan-05 BiHs]  1. Open Buffer --> Support export Delta time with 8 bytes  P5-> Export DeltaTime  0: not export (default)  1: export 4 bytes  2: export 8 bytes  V1.0.2.21[21-Dec-04 BiHs]  1. Support 'UTS\_Parse\_SendCMD' function  'USB;-1;80 06 00 01 00 00 12 00'  'EP;-1;80 06 00 01 00 00 12 00;1'  USB-> USB request  HID++-> HID++ request  Vendor-> Vendor command  EP -> Send command via Endpoint request  V1.0.2.10 [3-Dec-04 BiHs]  1. Solve bug : support format change on Buffer\_Read and Buffer\_SearchData command  compatible with Capri FT2  V1.0.2.9 [20-Oct-04 BiHs]  Add UD\_GetFirstDataInBuffer function for get the whole buffer information (not only data)  Version 1.0.2.8 [14-Oct-04 BiHs]  1. In GetDevIDbyPIDVID and GetUnSelectDevIDbyDeviceType function, check if device exist, if yes, then it could be opened or it would find the next one.  Version 1.0.2.7   1. Based on Bus\_hound **5.02 I** 2. Add DevExist function for get the device present status   Version 1.0.2.2  1. Based on Bus\_hound **5.02G**  2. Remove ParentID parameter in function UD\_OpenDeviceByDeviceType  3. Add parameter CMDTimeout of sending USB, Vendor commands | Bird Hsieh | 01/26/2005 |
| A3 | Version 1.0.1.16  1. PS2 OpenDevice: make sure it would open the PS2 device  Version 1.0.1.15   1. Add SetFilter function in Buffer group 2. Modify the data Delta function, the time delta will be the time difference between the last read data and the current one. | Bird Hsieh | 10/1/2004 |
| A2 | Based on Version 1.0.1.11  Solve a bug for reading buffer (when buffer is empty, it would always get the data)  Version 1.0.1.10  1. For the different DevID, it would have the different buffer pointer for reading the data  2. Export some functions out, such as  UD\_BufferOpen  UD\_BufferRead  UD\_BufferSetPattern  UD\_InitializeDriver  UD\_ReleaseDriver  Version 1.0.1.9  1. Buffer\_Control->Open: Add a paratmeter to specify if you want to export TimeStamp (delta) value in the end of data report  Based on Version 1.0.1.8  2. Exe\_Parse\_DLL : add parameter 12 to indicate if you want to control Buffer Open/Clear/Close in this command | Bird Hsieh | 05/08/2004 |
| A1 | Based on Version 1.0.1.5  1. Change USB-> USB Request Command | Bird Hsieh | 3/5/2004 |
| A0 | First Revision | Bird Hsieh |  |

# Purpose

UTS\_Drivers if for providing the UTS DLL interface of USB/PS2/COM test driver which can access test data from those IO port.

For V1.0.1.X is based on the Bus\_hound filter driver.

From V2.0.0.0 start to support Win7/8

V2.0.0.0 also support Win XP if compile with visual studio 2012 update 4

# DLL Directory

UTS\_Drivers.DLL should be in the sub folder “\\UTS\_Drivers” of the UTS main software so the commands to load the DLL should as:

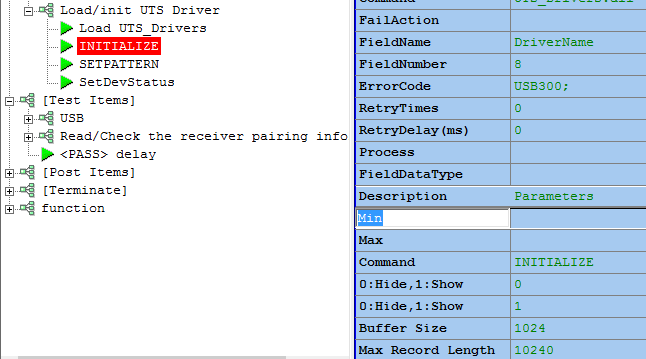
|  |  |  |  |
| --- | --- | --- | --- |
| **Purpose** | **Parameters** | **Values** |  |
| Load UTS\_Drivers.DLL | **Section1** | DLL | |
| **Section2** | Load | |
| **Command** | UTS\_Drivers.DLL | |
| **V1(Min)** |  | |
| **V2(Max)** |  | |

# Command List

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **P3** | **P4** | **P5** | **P6** | **P7** | **Result** |
| [Initialize](#_Initialize) | If not Hide Data  0 or “” = Hide  1 = not Hide | If not Hide Incoming Data  0 or “” = Hide  1 = not Hide | Buffer Size  (K byte)  Default = 512 | Max Record Length(Byte)  Default = 4096 | DriverName |
| [Release](#_Release) |  |  |  |  | Release  Driver |
| [SetDebugFlag](#_SetDebugFlag) | Debug data  1 = show  0 = hide |  |  |  |  |
| [DevExist](#_DevExist) | Device ID |  |  |  | true/false |
| [GetDevInformation](#_GetDevInformation) | Device ID | Information Index |  |  | Wanted  Information |
| [SetDevStatus](#_SetDevStatus) | Device ID | Mask  Device flag first &= ~Mask | Value  Device flag then |= (Mask & Value) |  | DevStatus (0x????) |
| [SetDriverMode](#_SetDriverMode) | Mode Flag | Enable  0 = &~Mode flag  1 = | Mode flag |  |  | Mode open or close |

## General

Taking Initialize for example show how UTS use the upper form. Others are the same.



**P3**

**P4**

**P5**

**P6**

**P7**

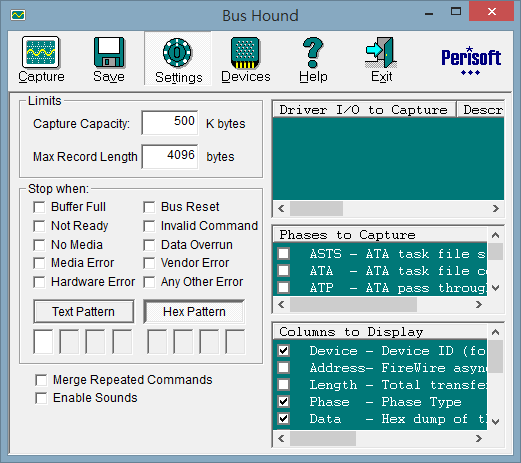
### 4.1.1 Initialize

**Purpose:** Get driver handle and initialize the driver if needed.

**Recommend:** You **MUST** have this command in [Initiation] group in your test plan if you want to use the commands set in this DLL.

**For example:**

| **P3**  **Group** | **P4**  **Hide data or not**  **0 or “” : hide**  **1: not hide** | **P5**  **Hide new USB device or not**  **0 or “” : hide**  **1: not hide** | **P6**  **Buffer Size (K byte)**  **If <512, then set it to 512** | **P7**  **Max Record**  **Length (Byte)**  **If <4096, then set it to 4096** | **Result** |
| --- | --- | --- | --- | --- | --- |
| Initialize | 0 | 1 | 512 | 4096 | PASS |



### 4.1.2 Release

**Purpose:** Release driver handle and resource if needed.

**Recommend:** You **MUST** have this command in [Terminate] group in your test plan if you have initialize command in [Initiation] group.

**For example:**

| **P3**  **Group** | **Result** |
| --- | --- |
| Release | PASS |

### 4.1.3 SetDebugFlag

**Purpose:** Enable or disable the running information to show in UTS main application. By default, it would be disable.

**Recommend:** In formal release test plan, it is better to remove this command.

**For example:**

| **P3**  **Group** | **P4**  **1: enable**  **0: disable** | **Result** |
| --- | --- | --- |
| SetDebugFlag | 1 | PASS |

### 4.1.4 DevExist

**Purpose:** Check if Device is plugged in to PC or not via Dev ID. The result 1 means the device exists in PC, or 0, means doesn’t exist.

**Recommend:** Even the bus\_hound can detect the device is existing in PC but that doesn’t mean that is ready to use (such as sending USB command). Thus it is better to wait for 3 sec for PC emulating.

**For example:**

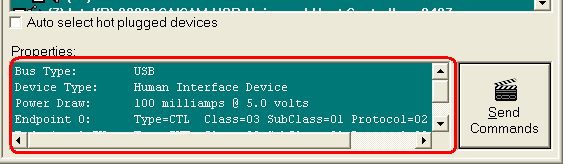
| **P3**  **Group** | **P4**  **DevID** | **Result** |
| --- | --- | --- |
| DevExist | 10 | 1/0(true/false) |

### 4.1.5 GetDevInformation

**Purpose:** For getting the device properties which **is provided by Bus\_hound**. You can base on **Information Index** as the following to get the property.

|  |  |
| --- | --- |
| Information Index | Wanted Information |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | ParentID | 4 | BusType | 7 | BusSpeed (megabit) | 10 | MicroAmps |
| 2 | Periph | 5 | CmdTypes | 8 | Name | 11 | MicroVolts |
| 3 | Flag | 6 | Resources | 9 | ROM | 12 | USB Port number |



**Recommend:** Don’t use this if there has the other method since this is bus\_hound relative command.

**For example:**

| **P3**  **Group** | **P4**  **DevID** | **P5**  **Information Index** | **Result** |
| --- | --- | --- | --- |
| GetDevInformation | 10 | 12 | 10’s USB Port Number |

### 4.1.6 SetDevStatus

**Purpose:** For setting/getting the device status which **is provided by Bus\_hound**. You can base on status parameter as the following to get/set the status:

|  |  |  |
| --- | --- | --- |
| **Device Status Definition** | **Parameter** | **Comment** |
| #define DEV\_PRESENT | 0x0001 | // device is present |
| #define DEV\_SELECTED | 0x0002 | // device is selected for capture |
| #define DEV\_MICROVOLTS | 0x0004 | // the microvolts field is valid |
| #define DEV\_MICROAMPS | 0x0008 | // the microamps field is valid |
| #define DEV\_LUN | 0x0010 | // lun field is valid |
| #define DEV\_TARGET\_ID | 0x0020 | // target id field is valid |
| #define DEV\_BUS | 0x0040 | // bus# field is valid |
| #define DEV\_PORT | 0x0080 | // port# field is valid |
| #define DEV\_NODE | 0x0100 | // 1394 node field is valid |
| #define DEV\_CONFIGROM | 0x0200 | // 1394 config rom is valid |
| #define DEV\_LOCKED | 0x0400 | // device is locked for exclusive access |
| #define DEV\_HIDE\_INPUT | 0x0800 | // device input is hidden from the system |
| #define DEV\_LOW\_POWER | 0x1000 | // set device to its lowest power mode |

**Recommend:** Don’t use this if there has the other method since this is bus\_hound relative command.

**Parameters expand:**

**P4** choose the Dev you want, and we will get its’ **Device Flag**.

**P5** is used to set **Device Flag &= ~ Mask (P5)**, so status will close first.

**P6** is used to set **Device Flag |= (Mask & Value) (P5 & P6)**, so status will open secondly.

**For example:**

| **P3**  **Group** | **P4**  **DevID** | **P5**  **Mask** | **P6**  **Value** | **Result**  **Final Status** |
| --- | --- | --- | --- | --- |
| SetDevStatus | 10 | 0x0002 | 0x0002 | Device Flag = 0x0002  Dev10 will be selected |

### 4.1.7 SetDriverMode

**Purpose:** For setting/getting the driver mode which **is provided by Bus\_hound**. You can base on mode parameter as the following to get/set the mode:

|  |  |  |
| --- | --- | --- |
| **Driver Mode Definition** | **Parameter** | **Comment** |
| #define MODE\_RUN | 0x000001 | // run/stop state |
| #define MODE\_CLEAR | 0x000002 | // clear capture contents |
| #define MODE\_REPS | 0x000004 | // merge identical commands |
| #define MODE\_STOP\_ON\_FULL | 0x000008 | // disable buffer wrapping |
| #define MODE\_STOP\_ON\_PATTERN | 0x000010 | // stop on data pattern match |
| #define MODE\_STOP\_ON\_NOT\_READY | 0x000020 | // stop on not ready errors |
| #define MODE\_STOP\_ON\_NO\_MEDIA | 0x000040 | // stop on media changes |
| #define MODE\_STOP\_ON\_MEDIA\_ERROR | 0x000080 | // stop on bad blocks |
| #define MODE\_STOP\_ON\_INVALID\_CMD | 0x000100 | // stop on invalid command |
| #define MODE\_STOP\_ON\_RESET | 0x000200 | // stop on bus reset |
| #define MODE\_STOP\_ON\_OVERRUN | 0x000400 | // stop on data overrun/underrun |
| #define MODE\_STOP\_ON\_VENDOR\_ERROR | 0x000800 | // stop on vendor unique error |
| #define MODE\_STOP\_ON\_HARDWARE\_ERROR | 0x001000 | // stop on hardware failure |
| #define MODE\_STOP\_ON\_xxx | 0x002000 | // reserved |
| #define MODE\_STOP\_ON\_OTHER\_ERROR | 0x004000 | // stop on other error conditions |
| #define MODE\_HIDE\_INPUT | 0x008000 | // hide input of new devices |
| #define MODE\_AUTO\_SELECT | 0x010000 | // auto select hot plugged devs |
| #define MODE\_CLEAR\_DEVICE\_HISTORY | 0x020000 | // deselect all non-present devs |

**Recommend:** Don’t use this if there has the other method since this is bus\_hound relative command.

**Parameters expand:**

**P4** is used to choose the mode you want, for example: P4 = 0x000002 + 0x008000 which stand for MODE\_CLEAR and MODE\_HIDE\_INPUT

**P5** is used to enable or disable the mode you choose. See the following example.

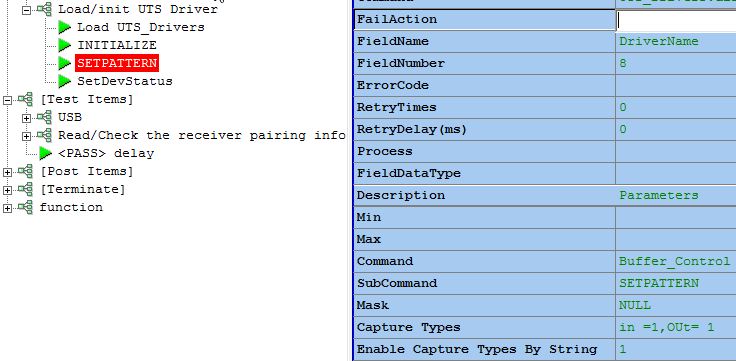
**For example:**

| **P3**  **Group** | **P4**  **Mode Flag** | **P5**  **Enable** | **Result**  **Final Driver Mode** |
| --- | --- | --- | --- |
| SetDriverMode | 0x0001 | 0 | ~MODE\_RUN |
| SetDriverMode | 0x0002 | 1 | MODE\_CLEAR |

## Buffer Control

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **P3** | **P4** | **P5** | **P6** | **P7** | **Result** |
| Buffer\_Control | [Open](#_4.2.1_Buffer_Open)  (same as driver’s MODE\_RUN) | ExportDelta Timp  0:not export (default)  1:export 4 bytes  2:export 8 bytes |  |  |  |
| [Close](#_4.2.2_Buffer_Close)  (same as driver’s ~MODE\_RUN) |  |  |  |  |
| [Clear](#_4.2.3_Buffer_Clear)  (same as driver’s MODE\_CLEAR) |  |  |  |  |
| [Read](#_4.2.4_Buffer_Read) | Device ID | Timeout  default=0 |  |  |
| [SetPattern](#_4.2.5_Buffer_Setpattern) | Mask Pattern | By default : Capture types Data\_In only | Enable\_CaptureType\_By\_String(1/0) |  |
| [SearchData](#_4.2.6_SearchData) | Device ID | Sub String | Timeout |  |
| [SetFilter](#_4.2.7_SetFilter) | FilterString |  |  |  |
| [ClearDevIDFamily](#_4.2.8_ClearDevIDFamily) | Any Device ID in a family tree |  |  |  |
| [ClearDevIDSpecific](#_4.2.9_ClearDevIDSpecific) | Device ID |  |  |  |

Take SetPattern as a sample to illustrate how to use the Buffer Control functions:



**P3**

**P4**

**P5**

**P6**

**P7**

### 4.2.1 (Buffer) Open

**Purpose:** Open the buffer for getting device communication data.

You can set the parameter to specify if you want to export Delta time.

**Recommend:** Although it is same as MODE\_RUN, please still call this to open buffer rather than SetDriverMode.

**For example:**

| **P3**  **Group** | **P4**  **Sub command** | **P5**  **Export Delta time** | **Result** |
| --- | --- | --- | --- |
| Buffer\_Control | Open  (same as driver’s MODE\_RUN) | 0: not export (default)  1: export 4 bytes  2: export 8 bytes | PASS |

### 4.2.2 (Buffer) Close

**Purpose:** Close the buffer.

**Recommend:** It is simpler to call this to close buffer than SetDriverMode. However, you can also call SetDriverMode(~MODE\_RUN) to close buffer.

**For example:**

| **P3**  **Group** | **P4**  **Sub command** | **Result** |
| --- | --- | --- |
| Buffer\_Control | Close | PASS |

Or

| **P3**  **Group** | **P4**  **Mode Flag** | **P5**  **Enable** | **Result**  **Final Driver Mode** |
| --- | --- | --- | --- |
| SetDriverMode | 0x0001 | 0 | ~MODE\_RUN |

### 4.2.3 (Buffer) Clear

**Purpose:** Clear the all data in buffer. It is the real way to clear buffer.

**Recommend:** It is simpler to call this to clean buffer than SetDriverMode. However, you can also call SetDriverMode(MODE\_CLEAR) to clear buffer.

**For example:**

| **P3**  **Group** | **P4**  **Sub command** | **Result** |
| --- | --- | --- |
| Buffer\_Control | Clear | PASS |

Or

| **P3**  **Group** | **P4**  **Mode Flag** | **P5**  **Enable** | **Result**  **Final Driver Mode** |
| --- | --- | --- | --- |
| SetDriverMode | 0x0002 | 1 | MODE\_CLEAR |

### 4.2.4 (Buffer) Read

**Purpose:** Read the 1st data in buffer which sent by Device you specify. So you must specify the Device ID for what you want.

**Usage 1:**

If like to summit command and get command echo could be done by below two steps:

1. Summit command : **USBRequest**
2. Read echo: **Buffer\_Control Read**

The more efficiency way is using HID++Request to summit command and get feedback in one command.

**Usage 2:**

To read mouse motion data in buffer, this Read command can read one record data from buffer

**For example:**

| **P3**  **Group** | **P4**  **Sub command** | **P5**  **Device ID** | **Result** |
| --- | --- | --- | --- |
| Buffer\_Control | Read | 10 | Data |

### 4.2.5 (Buffer) Setpattern

**Purpose:** Choose the phase type you want to capture.

**Recommend:** There are two methods to call SetPattern. One way is called by integer value, and the other is called by setting Name value. The second way is simpler.

**Parameters expand:**

**P7** is used to enable set capture type by string. In other word, setting **P7 = 1** is choose **Method 2**, **others** is **Method 1**.

**P6** (Capture type’s example):

**Method 1** (origin method)

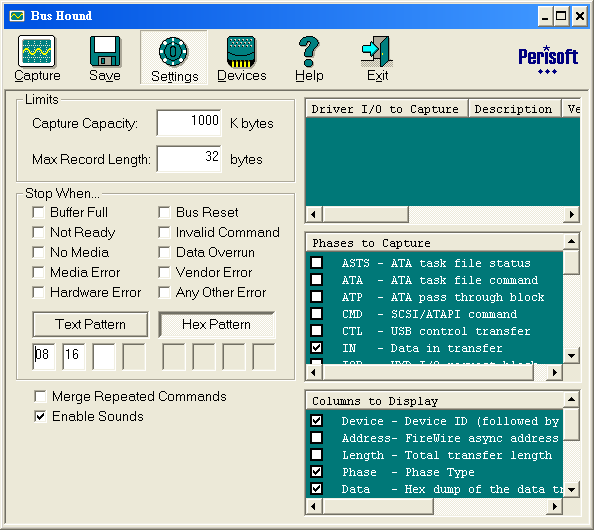
If you want to capture Data IN and Data Out record, you need to set **P6 = 8+16=24** whose values are from lower form.

|  |  |  |
| --- | --- | --- |
| **Phase Definition** | **Parameter** | **Comment** |
| #define NULL\_PHASE | 0x00000000 | // internal use |
| #define CDB\_PHASE | 0x00000001 | // cdb bytes |
| #define SENSE\_PHASE | 0x00000002 | // sense data |
| #define OK\_PHASE | 0x00000004 | // command complete |
| #define DATA\_IN\_PHASE | 0x00000008 | // data in bytes |
| #define DATA\_OUT\_PHASE | 0x00000010 | // data out bytes |
| #define ATA\_PHASE | 0x00000020 | // ATA task file command |
| #define ATA\_STATUS\_PHASE | 0x00000040 | // ATA task file response |
| #define ATP\_PHASE | 0x00000080 | // ATA pass through structure |
| #define ISOC\_PHASE | 0x00000100 | // isochronous phase |
| #define SPT\_PHASE | 0x00000200 | // SCSI Pass through struct |
| #define SRB\_PHASE | 0x00000400 | // SRB structure |
| #define SRB\_STATUS\_PHASE | 0x00000800 | // srb status byte |
| #define IRP\_PHASE | 0x00001000 | // IRP structure |
| #define IRP\_STACK\_PHASE | 0x00002000 | // IRP stack locations |
| #define NT\_STATUS\_PHASE | 0x00004000 | // nt status |
| #define IOR\_PHASE | 0x00008000 | // legacy; not used |
| #define IOR\_STATUS\_PHASE | 0x00010000 | // legacy; not used |
| #define URB\_PHASE | 0x00020000 | // URB structure (USB) |
| #define URB\_STATUS\_PHASE | 0x00040000 | // URB status |
| #define CTL\_PHASE | 0x00080000 | // USB control transfer |
| #define IRB\_PHASE | 0x00100000 | // 1394 IRB structure |
| #define LOCK\_PHASE | 0x00200000 | // 1394 lock |
| #define RESET\_PHASE | 0x00400000 | // 1394 bus reset |
| #define NDIS\_PHASE | 0x00800000 | // ndis packet; not used |
| #define BRB\_PHASE | 0x01000000 | // bluetooth request block |
| #define KEY\_PHASE | 0x02000000 | // keyboard input |
| #define PS2\_PHASE | 0x04000000 | // ps2 command |
| #define SCSI\_STATUS\_PHASE | 0x08000000 | // scsi status code More definition please |

**Method 2** (input parameter by string)

If you want to capture Data IN and Data Out record, you need to set **P6 =” In= 1, OUT =0 “**which is case-insensitive, whitespace is OK and use comma to separate one by one.

P5 is used to set the pattern you hope bus\_hound to search and once get this string, it would stop capture data.



**For example:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **P3**  **Group** | **P4**  **Sub command** | **P5**  **Mask** | **P6**  **Capture Types** | **P7**  **Enable\_CaptureType\_By\_String** | **Result** |
| Buffer\_Control | SetPattern | 10 20 00 | 47 | 0 or “” | PASS |
| Buffer\_Control | SetPattern | NULL | In= 1,OUT =1 | 1 | PASS |

### 4.2.6 (Buffer) SearchData

**Purpose:** Within <time out> parameter, it will search the sub string in the buffer and once it got, it would return the full string.

**For example:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **P3**  **Group** | **P4**  **Sub command** | **P5**  **Device ID** | **P6**  **Sub String** | **P7**  **Timeout** | **Result**  **Full string** |
| Buffer\_Control | SearchData | 10 | 10 20 00 | 1000 | 00 10 20 00 01 |

### 4.2.7 (Buffer) SetFilter

**Purpose:** You can set the filter string for Buffer. Based on this string, you can screen out the data you want. The filter format will be the same as UTS Compare string format.

**For example:**

|  |  |  |  |
| --- | --- | --- | --- |
| **P3**  **Group** | **P4**  **Sub command** | **P5**  **Filter string** | **Result** |
| Buffer\_Control | SetFilter | 00 XX | PASS |

### 4.2.8 (Buffer) ClearDevIDFamily

**Purpose:** You can clear a family buffer by any Device ID in the family tree.

**Recommend:** You **must not** use it **when not using open\_device\_by\_PIDVID** because the family tree build only when using it. It means that using open\_device\_by\_DeviceType can not use this function.

**For example:**

| **P3**  **Group** | **P4**  **Sub command** | **P5**  **Any Device ID in the family tree** | **Result** |
| --- | --- | --- | --- |
| Buffer\_Control | ClearDevIDFamily | 10 | Clear Device 10’s family buffer |

### 4.2.9 (Buffer) ClearDevIDSpecific

**Purpose:** You can clear a Device’s buffer by its’ Device ID.

**Recommend:** No having such limit like ClearDevIDFamily.

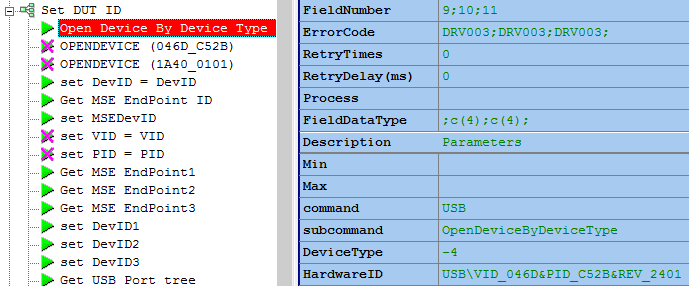
**For example:**

| **P3**  **Group** | **P4**  **Sub command** | **P5**  **Device ID** | **Result** |
| --- | --- | --- | --- |
| Buffer\_Control | ClearDevIDSpecific | 10 | Clear Device 10’s buffer |

| **P3** | **P4** | **P5** | **P6** | **P7** | **P8** | **P9** | **P10** | **P11** | **P12** | **Result** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| USB | [OpenDevice](#_(USB)_OpenDevice) | VID | PID |  |  |  |  |  |  | **Device ID; VID; PID** |
| [OpenDevice](#_(USB)_OpenDevice) | Descriptor |  |  |  |  |  |  |  | **Device ID; VID; PID** |
| [OpenDeviceByDeviceType](#_4.3.2_(USB)_OpenDeviceByDeviceType) | Device Type | Hardware ID |  |  |  |  |  |  | **Device ID; VID; PID** |
| [CloseDevice](#_(USB)_CloseDevice) | Device ID |  |  |  |  |  |  |  |  |
| [VendorCommand](#_(USB)_VendorCommand) | Device ID | Command list | Format | CMDTimeout  default=infinite |  |  |  |  | **Command result if it has** |
| [USBRequest](#_(USB)_USBRequest) | Device ID | Command list | Format | CMDTimeout  default=infinite |  |  |  |  | **Command result if it has** |
| [HID++Request](#_(USB)_HID++Request) | Device ID | Command list | Timeout | Format | CMDTimeout  default=infinite | Respond  DevID |  |  |  |
| [EndPointRequest](#_(USB)_EndPointRequest) | Device ID | EndpointID | Command |  |  |  |  |  |  |
| [ExeParseDll](#_(USB)_ExeParseDLL) | Device ID | DLLName | StartCmd | ConfirmStr | EndCmd | Timeout | The Parameters of Parse DLL | IfControlBuffer | **DLL result** |
| [GetEndpointID](#_Extra_function) | Device ID | Keyword |  |  |  |  |  |  | **ParentID;**  **EndpointID;** |
| [GetIDByEpNo](#_(USB)_GetIDByEpNo) | Device ID | EP number |  |  |  |  |  |  | **EpID** |
| [GetUSBPortTree](#_4.3.11_(USB)_GetUSBPortTree) | Device ID |  |  |  |  |  |  |  |  |
| [GetUSBPortDevIDByCOMNumber](#_4.3.12_(USB)_GetUSBPortDevIDByCOMNu) | COMNumber |  |  |  |  |  |  |  |  |

## USB Section

Taking OpenDeviceByDeviceType for example shows how UTS use USB Group. Others are the same.



**P3**

**P4**

**P5**

**P6**

### 4.3.1 (USB) OpenDevice

**Purpose:** Before you use the other commands in USB section, you need to get the [Device ID] by this command.

It would set device (including the endpoints which belong to this device) as selected in Bus\_hound driver and [if you enable “HIDE input data” (by general section-> initialize)](#_General), it would hide the USB data information to PC.

**Recommend:** You could put this command in [Initiation] group in your test plan.

There are two methods to get the device ID for your DUT:

1. **By PID/VID**

**For example:**

| **P3**  **USB** | **P4**  **Function Name** | **P5**  **VID** | **P6**  **PID** | **Result** |
| --- | --- | --- | --- | --- |
| USB | OpenDevice | 0x046D | 0xC702 | 10;046D;C702 |

1. **By Descriptor**

It would get descriptor by USB request: Get\_Descriptor\_String\_Product 80 06 02 03 00 00 1A 00 and if it match the string you specify in test plan, it would return its device ID.

**For example:**

| **P3**  **USB** | **P4**  **Function Name** | **P5**  **Descriptor** | **Result** |
| --- | --- | --- | --- |
| USB | OpenDevice | Receiver | 10;046D;C702 |

### 4.3.2 (USB) OpenDeviceByDeviceType

**Purpose:** Use the following Device Type form to search the Device. Then use Hardware ID to check whether the device is or not. If match, open it and its’ children.

|  |  |  |
| --- | --- | --- |
| **Device Type Definition** | **Parameter** | **Comment** |
| #define PERIPH\_DISK | 0 | // disk drive |
| #define PERIPH\_TAPE | 1 | // tape drive |
| #define PERIPH\_PRINTER | 2 | // printer |
| #define PERIPH\_CPU | 3 | // processor |
| #define PERIPH\_WORM | 4 | // write once device |
| #define PERIPH\_CDROM | 5 | // optical device |
| #define PERIPH\_IMAGING | 6 | // scanner, camcorder |
| #define PERIPH\_OPTICAL | 7 | // r/w optical |
| #define PERIPH\_CHANGER | 8 | // media changer |
| #define PERIPH\_COMM | 9 | // communications |
| #define PERIPH\_FLOPPY | 10 | // floppy disk (observed through vxd) |
| #define PERIPH\_CTLR | -1 | // controller |
| #define PERIPH\_UNDEF | -2 | // unknown peripheral type |
| #define PERIPH\_MOUSE | -3 | // mouse |
| #define PERIPH\_HID | -4 | // human interface device |
| #define PERIPH\_NET | -5 | // network |
| #define PERIPH\_KEYBOARD | -6 | // keyboard |
| #define PERIPH\_PORTS | -7 | // serial/parallel port |
| #define PERIPH\_BLUETOOTH | -8 | // bluetooth radio |
| #define PERIPH\_INFRARED | -9 | // infrared |
| #define PERIPH\_MODEM | -10 | // modem |

**Recommend:** If you don’t give it Hardware ID, it will still open the device which match the upper type and exists. But you have better give it Hardware ID to make sure the correctness.

**For example:**

| **P3**  **USB** | **P4**  **Function Name** | **P5**  **Device Type** | **P6**  **Hardware ID** | **Result** |
| --- | --- | --- | --- | --- |
| USB | OpenDeviceByDeviceType | -4 | HID\VID\_046D&PID\_C52B&REV\_2401 | 36;046D;C52B |

### 4.3.3 (USB) CloseDevice

**Purpose:** It would un-select the device (including the endpoints which belong to this device) in Bus\_hound driver and it would disable hide the USB data information to PC.

**Recommend:** You could put this command in [Terminate] group in your test plan.

**For example:**

| **P3**  **USB** | **P4**  **Function Name** | **P5**  **Device ID** | **Result** |
| --- | --- | --- | --- |
| USB | CloseDevice | 10 | PASS |
| USB | CloseDevice | [&Var:DevID] | PASS |

### 4.3.4 (USB) VendorCommand

**Purpose**: Send vendor command by Device ID.

**Recommend:** You can [use “Format” parameter](#_Extra_function) to extract the data you want.

**For example:**

| **P3**  **USB** | **P4**  **Function Name** | **P5**  **Device ID** | **P6**  **Command** | **P7**  **Format** | **Result** |
| --- | --- | --- | --- | --- | --- |
| USB | VendorCommand | 10 | 40 02 05 00 01 00 00 00 |  |  |
| USB | VendorCommand | [&Var:DevID] | C0 09 0B 00 01 00 02 00 | S([1][2]) | 00 |

### 4.3.5 (USB) USBRequest

**Purpose**: Send USB Request by Device ID and

Before V1.0.1.4, it is the same the Vendor command calling rule.

After V1.0.1.5, if the Command length is over 8, it would fill in the extra data into buffer for sending to USB device.

**Recommend:** You can [use “Format” parameter](#_Extra_function) to extract the data you want.

**For example:**

| **P3**  **USB** | **P4**  **Function Name** | **P5**  **Device ID** | **P6**  **Command** | **P7**  **Format** | **P8**  **CMDTimeout** | **Result** |
| --- | --- | --- | --- | --- | --- | --- |
| USB | USBRequest | 10 | 80 06 00 01 00 00 12 00 |  |  | PASS |
| USB | USBRequest | 10 | 21 09 00 01 00 00 01 00 01 |  |  | PASS |

### 4.3.6 (USB) HID++Request

**Purpose**: Send HID++ Request (21 09 10 02 01 00 07 00 XX XX XX XX XX XX XX) by Device ID and now, **it will wait for** DUT response for this HID++ request.

It would check if the response is based on this HID++ request. The checking rule is that the byte[1][2][3] of response must be the same as the byte[1][2][3] of command (Note: Skip for checking byte[0]) based on Logitech HID++ spec.

**Recommend:** You can [use “Format” parameter](#_Extra_function) to extract the data you want.

**Parameter expand:** **P10 is a new parameter start from version 2.0.0.0 (for win7/Win8) to support multiple test to get the right echo. Each device echo data will not affect each other on multiple testing.**

**Note:**

1. **This function not suitable for multiple testing since echo data may cleared or overwrite by others device id echo data.**
2. **For single DUT test, this function is works fine in both old and new version.**

| **P3**  **USB** | **P4**  **Function Name** | **P5**  **Device ID** | **P6**  **Command** | **P7**  **Timeout** |
| --- | --- | --- | --- | --- |
| USB | HID++Request | 13 | 21 09 10 02 01 00 07 00 10 FF 80 80 01 00 00 | 1000 |
| **P8**  **format** | **P9**  **CMD timeout** | **P10**  **WantDevID**  **(Respond ID)** | **Result** |  |
|  | 0 | 16 |  |  |

**For example:**

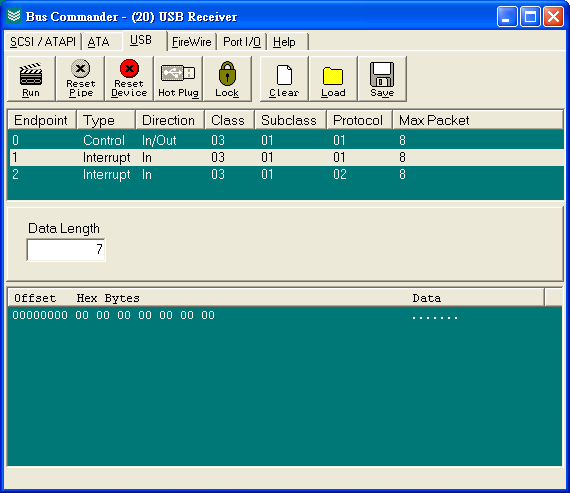
### 4.3.7 (USB) EndPointRequest

**Purpose**: Send Endpoint Request by Device ID and EndpointID.

**Recommend:** You can [use “Format” parameter](#_Extra_function) to extract the data you want.

**For example:**

| **P3**  **USB** | **P4**  **Function Name** | **P5 Device ID** | **P6**  **EndpointID** | **P7**  **Command** | **Result** |
| --- | --- | --- | --- | --- | --- |
| USB | EndPointRequest | 10 | 1 | 10 FF 80 80 01 00 00 |  |



### 4.3.8 (USB) ExeParseDLL

**Purpose**: Load the UTS Parse DLL and pass the data into that for analyze.

**Recommend:** Since this command will use bus\_hound buffer, it would clean buffer data before execute. And after finish, it would close buffer automatically.

**For example:**

| **P3**  **USB** | **P4**  **Function Name** | **P5**  **Device ID** | **P6**  **DLLName** | **P7**  **StartCmd** | **P8**  **ConfirmStr** |
| --- | --- | --- | --- | --- | --- |
| USB | ExeParseDLL | 10 | .\UTS\_Parse\_Btn.dll |  |  |
| **P9**  **EndCmd** | **P10**  **Timeout** | **P11**  **The Parameters of Parse DLL** | **P12**  **IfControl Buffer** | **P13**  **File Location** | **Result** |
|  | 60000 | \*Part 1 | \*Part 2 | D:\SRC\teset\_data.txt | **DLL result** |

**Parameter expand:**

\*Part 1: it could be

"[Data]"

["TopX="+'&Image\_Pos\_X']

["TopY="+'&Image\_Pos\_Y']

["BackgroundPicture="+'&TestPLANPath'+"Bach\_btn.bmp"]

BtnPos=1

DataLen=4

"[Parameter]"

ButtonSequence=01 02

XString=90 147

YString=200 145

WidthString=30 72

HeightString=65 72

\*Part 2: By default, it would 0

0: Control Buffer Open/Clear/Close inside this command

1 or not 0: doesn’t control buffer, just read data from buffer and send to Parse\_DLL.

\*If P13 is added, then the data will read from test\_data txt file not bushound.

If P13 is NULL, then it will do as before.

**Note: Debug data file contains data look like**

**00000100FFFF0000000007D0 4 bytes Timestamp must be added**

Ini setting on Parse Tracking:

DataLen=12

XPos=S([7][8][5][6])

YPos=S([11][12][9][10])

TimeTick\_us=1

TimeStampPos=9 10 11 12

### 4.3.9 (USB) GetEndpointID

**Purpose:** Get the Endpoint ID by the DeviceType (0: keyboard, 1: Mouse) under the device you specify.

**P6 parameter:**

0: Keyboard

1: Mouse

**For example:**

| **P3**  **USB** | **P4**  **Function Name** | **P5 Device ID** | **P6**  **Device Type** | **Result**  **EpID (BusType= USB);**  **SubEpID(DeviceType = 0/1)** |
| --- | --- | --- | --- | --- |
| USB | GetEndpointID | 20 | 0 | 21;23; |
| USB | GetEndpointID | 20 | 1 | 22;24; |



### 4.3.10 (USB) GetIDByEpNo

**Purpose:** Get the DevID by the Endpoint Number (1...N) under the device you specify.

**P6 parameter:**

0…N: Endpoint number

**For example:**

| **P3**  **USB** | **P4**  **Function Name** | **P5**  **Device ID** | **P6**  **EndPoint Number** | **Result**  **EpID (BusType= USB);** |
| --- | --- | --- | --- | --- |
| USB | GetIDByEpNo | 20 | 1 | 21; |
| USB | GetIDByEpNo | 20 | 2 | 22; |

### 4.3.11 (USB) GetUSBPortTree

**Purpose:** No matter device ID selected or not, get USB port tree by **Device ID**.

**For example:**

| **P3**  **USB** | **P4**  **Function Name** | **P5**  **Device ID** | **Result** |
| --- | --- | --- | --- |
| USB | GetUSBPortTree | 36 | Port Tree: 00001788  Device ID (Port) Tree:  0(0)<--0(0)<--0(0)<--0(0)<--9(0)<--12(1)<--32(7)<--34(8)<--36(8) |

### 

### 4.3.12 (USB) GetUSBPortDevIDByCOMNumber

**Purpose:** No matter device ID selected or not, get USB port tree by **COM Number**.

**For example:**

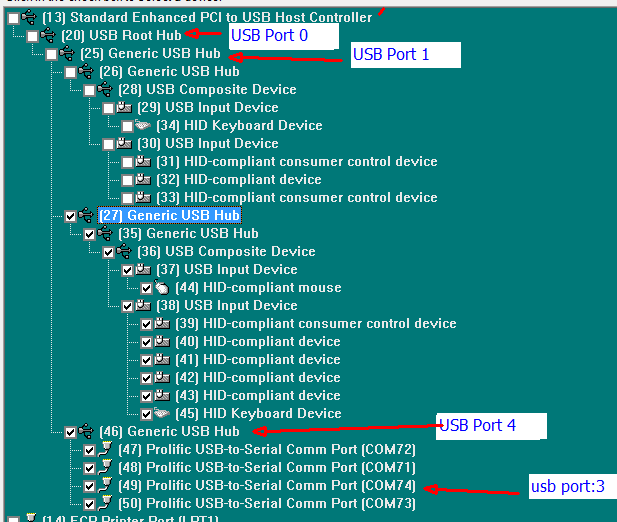
| **P3**  **USB** | **P4**  **Function Name** | **P5**  **COM Number** | **Result** |
| --- | --- | --- | --- |
| USB | GetUSBPortDevIDByCOMNumber | 20 | Port Tree: 00001822  Device ID (Port) Tree:  0(0)<--0(0)<--0(0)<--0(0)<--9(0)<--12(1)<--44(8)<--46(2)<--47(2) |

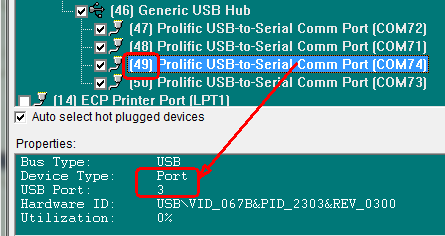
For example:

Input COM port number 74

Result:

Port Tree: 00000143

Tree: 0(0)<--0(0)<--0(0)<--0(0)<--13(0)<--20(0)<--25(1)<--46(4)<--49(3) 



## PS2 Section

| **P3**  **USB** | **P4**  **Function Name** | **P5** | **Result** |
| --- | --- | --- | --- |
| PS2 | OpenDevice | 0: KBD  1: MSE | **Device ID;** |
| PS2 | CloseDevice | Device ID |  |

### 4.4.1 (PS2) OpenDevice

**Purpose:** Before you use the other commands in PS2 section, you need to get the [Device ID] by this command.

**Recommend:** You could put this command in [Initiation] group in your test plan.

### 4.4.2 (PS2) CloseDevice

**Purpose:** It would un-select the device (including the endpoints which belong to this device) in Bus\_hound driver and it would disable hide the PS2 data information to PC.

**Recommend:** You could put this command in [Terminate] group in your test plan.

| **P3**  **USB** | **P4**  **Function Name** | **P5**  **Device ID** | **Result** |
| --- | --- | --- | --- |
| PS2 | CloseDevice | 10 | PASS |
| PS2 | CloseDevice | [&Var:DevID] | PASS |

**For example:**

## Extra function

For each command, you can add the parameter which name is “Format” to change to result string.

For example, if you read the BT address of the DUT and its result is “0000 1234 005B 0002”. You can add a parameter “Format” to ask the main program to change the result to “0002:5B:001234” by the translation format “S([16][17][18][19]:[ 13][14]:[3][4][6][7][ 8][9])”

In Test Plan:

|  |  |  |
| --- | --- | --- |
| **V6** | Format | “S([16][17][18][19]:[ 13][14]:[3][4][6][7][ 8][9])” |

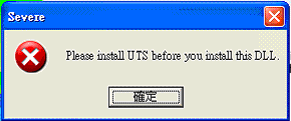
Result:

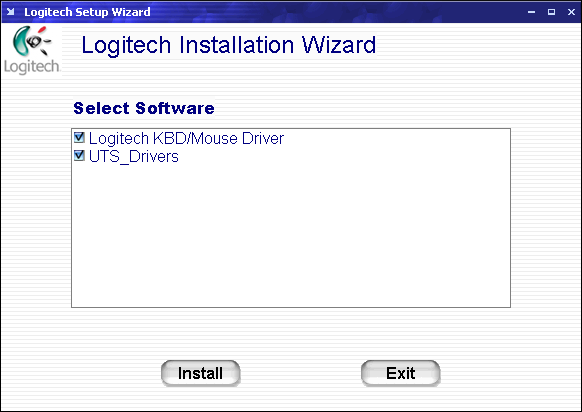
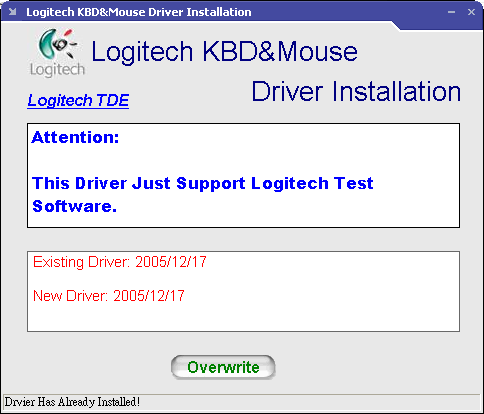
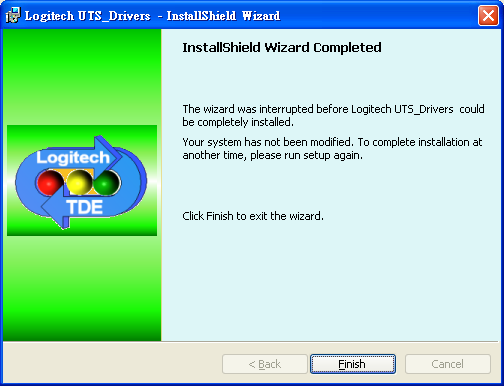
|  |  |  |
| --- | --- | --- |
| **Result** | PSKey | “0002:5B: 001234” |

# DLL Installation Guide

## UTS\_Drivers DLL Installation

1. Before you install the UTS\_Drivers.DLL, you need to make sure you had removed the old UTS\_Drivers if you once installed that. You can use ‘Add/remove program’ in the ‘control panel’ to remove that.
2. Make sure you already install the UTS main program. (Part ID 780192-0000 PART,TDE,SW, UTS). Or you will get the following error message.



1. In the setup zip file, there has three exe files: setup.exe, DrvSetup.exe, UTS\_DriversXXXXXXXX.exe.   
   **UTS\_DriversXXXXXXXX.exe**: install UTS\_Drivers.dll. But this DLL need that you install the bus\_hound filter driver.  
   **DrvSetup.exe**: install bus\_hound filter driver.  
   **setup.exe**: install both bus\_hound filter driver and UTS\_Drivers.dll.
2. If you run setup.exe. It would run bus\_hound (Logitech KBD/Mouse Driver) setup and UTS\_Drivers setup.  
   
3. The Logitech KBD/MOUSE driver install form could be as the following:  
   
4. In UTS\_Drivers setup file, if you got the following information, then the installation is finished.   
     
   UTS\_Driver.DLL should be in the “.\UTS\_Driver” folder relative to UTS main software so the commands to load the DLL will as:

|  |  |  |  |
| --- | --- | --- | --- |
| **Purpose** | **Parameters** | **Values** |  |
| Load UTS\_Imicrosira.DLL | **Section1** | DLL | |
| **Section2** | Load | |
| **Command** | .\UTS\_Drivers\UTS\_Drivers.DLL | |
| **V1(Min)** |  | |
| **V2(Max)** |  | |

1. Re-boot PC manually.

# Appendix A: Exported functions in UTS\_Drivers

For re-using the UTS\_Drivers.dll, we export some useful functions for the other AP.

1. UD\_InitializeDriver(int NotHideData, int NotHideIncomingData, int buffersize\_k, int Max\_Record\_Length);

**Purpose:** Get driver handle and set four parameters to initialize the driver.

1. UD\_Set\_Parameter\_IfNotHide(int iIfNotHide);

**Purpose:** Directly set the parameter NotHideData.

1. UD\_ReleaseDriver();

**Purpose:** Release driver handle.

1. UD\_BufferOpen(int iIfExport);

//0: not export (default value)

//1: export and in the end of data

//-1: not change

**Purpose:** Open the buffer.

1. UD\_BufferClose();

**Purpose:** Close the buffer.

1. UD\_BufferClearAll();

**Purpose:** Truly clear all buffer in driver.

1. UD\_BufferClearDevIDFamily(int DevID);

**Purpose:** Clear device ID family buffer by any device ID in the family tree.

1. UD\_BufferClearDevIDSpecific(int DevID);

**Purpose:** Clear the device ID buffer.

1. UD\_BufferSetPattern(int \* MaskList, int MaskLen, int PatternTypes);

**Purpose:** Select the **whole** phase type you want to capture.

**Parameter expand:** MaskList is used to set the pattern you hope bus\_hound to search and once get this string, it would stop capture data. And MaskLen is MaskList’s length. And PatternTypes values please reference [this form.](#_4.2.5_(Buffer)_Setpattern)

1. UD\_BufferSetPatternByName(int \* MaskList, int MaskLen, char\* PatternTypes);

**Purpose:** Select the **specific** phase type you want to capture or not.

**Parameter expand:** MaskList is same as UD\_BufferSetPattern. However, PatternTypes is a string, which is directly chose on or off. For example, “in= 0 , Out =1”, which is case-insensitive, and whitespace is OK and use comma to separate one by one.

1. UD\_BufferRead(int DevID, char \* Buffer, int Buffer\_len);

**Purpose:** Read the 1st data in buffer which sent by Device you specify. So you must specify the Device ID for what you want. And Buffer will be returned.

1. UD\_BufferSetFilter(char \* SetFilterString);

**Purpose:** You can set the filter string for Buffer. Based on this string, you can screen out the data you want. The filter format will be the same as UTS Compare string format.

1. UD\_OpenUSBDeviceByPIDVID(int PID, int VID, int \* DevID);

**Purpose:** Open device by VID and PID, then return the parent ID.

1. UD\_OpenDeviceByDeviceType(int DeviceType, int \* iDevID,char \* PHardwareID);

**Purpose:** Open device by DeviceType which is defined [here](#_(USB)_CloseDevice), and check the PHardwareID to make sure the correct device. Then it will return the parent ID.

1. UD\_OpenAllDevByVIDPID(int VID, int PID, int MaxNb, int \* NbofReady, char \* ReadyFixtureID, char \* DevMAP, int PortSubID);

**Purpose:** Open the number of MaxNb devices by VID and PID, and will check PortSubID if need (set 0 means needn’t), then return the number of ready and ReadyFixtureID (need 8 bytes) and DevMAP (need 32 bytes).

1. UD\_CloseDevice(int DevID);

**Purpose:** Close the Device.

1. UD\_UnselectAllDevice();

**Purpose:** Unselect all Device.

1. UD\_SelectDevID(int DevID, int IfSelect);

**Purpose:** Choose the Device ID and decide select or not select.

1. UD\_GetExistDevList(int \* DevList, int ListLen );

**Purpose:** Return the device list whose device is present and selected. ListLen is DevList’s length you give.

1. UD\_SendVendorCommand(int DevID, char \* CommandList, char \* ResultList, unsigned int BufferLength, int CMDTimeout);

**Purpose:** Send vendor command by Device ID.

1. UD\_SendUSBRequest(int DevID, char \* CommandList, char \* ResultList, unsigned int BufferLength, int CMDTimeout);

**Purpose:** Send USB Request by Device ID.

1. UD\_SendHIDPlusPlusRequest(int DevID, int timeout, char \* CommandList, char \* ResultList, unsigned int BufferLength, int CMDTimeout, int WantDevID);

**Purpose:** Send HID++ Request by Device ID and get the echo by WantDevID.

1. UD\_GetFirstDataInBuffer(int DevID ,RecPhaseInfo \* ReturnPhaseData);

**Purpose:** Get the buffer which is not read by Device ID.

1. UD\_GetParentIDByDevID(int iDevID, int \* ParentDevID);

**Purpose:** Get the parent ID by Device ID.

1. UD\_GetEndpointByKBD\_MSE(int ParentID, int MSE\_KBD, int \* iDevID);

//MSE\_KBD:

//0: Keyboard

//1: Mouse

**Purpose:** Get the endpoint device ID by parent ID and its’ device type (MSE or KBD).

1. UD\_GetEpIDbyEpNo(int ParentID, int EpNo, int \* EpID);

**Purpose:** Get the endpoint device ID by parent ID and the Nth number of endpoint.

1. UD\_GetUSBPortTree(int DevID, char \*port\_tree, char \*whole\_tree);

**Purpose:** Get the port tree (only port) and whole tree (device ID and port tree) by Device ID.

1. UD\_GetUSBPort\_DevID\_byCOMnumber(int COMnumber, char \*port\_tree, char \*whole\_tree);

**Purpose:** Get the port tree (only port) and whole tree (device ID and port tree) by COM number.

1. UD\_SWSuspendByDevID(int DevID, bool ifSuspend);

**Purpose:** Choose the device by Device ID. And decide going to low power mode or not by ifSuspend (true/false).

1. UTS\_Parse\_SendCMD(char \* CommandLine, char \* ResultString, int ResultStringLen);

**Purpose:** Send Request by CommandLine and Return by ResultString.

**Parameter expand:**

CommandLine is separated to three parts.

Part 1: CMD Type selection (USB, VENDOR, HID++, EP) and use”;” at the end.

Part 2: Give the Device ID and also use”;” at the end.

Part 3: The CMD List ex: USB Request CMD List.

ResultStringLen is ResultString’s length you give.

1. UD\_SetDevStatus(int DevID, int Mask,int Value, char \* ReturnBuffer, int bufferlength);

**Purpose:** Same as [SetDevStatus](#_4.1.6_SetDevStatus). bufferlength is ReturnBuffer’s length you give.

1. UD\_SetDriverMode(int ModeFlag, int enable);

**Purpose:** Same as [SetDriverMode](#_4.1.7_SetDriverMode).

1. UD\_GetDriverVersion(int \* pVer, int \* pMinVer);

**Purpose:** Get the Ver and pMinVer.

1. UD\_GetDevInformation(int DevID, int ParameterIndex, char \* DataReturn, int length);

**Purpose:** Same as [GetDevInformation](#_4.1.5_GetDevInformation). And the return will stored in DataReturn. Length is DataReturn’s length you give.

The header definition by Dephi is 