TOUCH-BASE documentation

Linux Installation

Revision 1.03, 21st Dev 2014

www.touch-base.com\documentation\installation

Deliverables	General Notes	Requirements	Distributions	Linux Notes	Installation notes	Installation	Driver Notes
Touch Interaction	Driver Utilities	Serial port	Uninstall	Mouse	Touch Utilities	Trouble Shooting	Contact

Welcome to UPDD Linux platform specific installation instructions and related notes for UPDD version 5.1.1124 and above. If using UPDD 4.1.x or 5.0.x you need to refer to the previous Linux installation instructions.

The previous documentation covered an evolutionary period of the Linux driver and thus covered a magnitude of issues, mainly related to installation and distribution quirks. This new documentation has been cut down to the basics to cater for the new driver and installation methods on recent and current distributions.

Development history can be viewed from the UPDD Console and can also be viewed here.

Under Linux we offer UPDD in two forms, UPDD Linux B(inary- pre compiled programs) and UPDD Linux S(ource).

The notes below refer to the UPDD Linux driver supplied in binary form for UPDD Linux B version 5.1.xxx, initial release April 2014. The source driver is available to touch manufacturers and integrators that require a source based Linux driver. Notes on the source driver are available here.

Very important general note: Linux is an open source environment utilised and supported by highly knowledgeable and capable developers. Many aspects of the Linux system are maintained by Linux groups and communities. Nearly all software created by these Linux groups is made available under open source license agreements, including any touch screen drivers that have been written, mainly by individuals. Linux distributions are many and varied and all have slight subtle differences. Many users, developers and integrators in the Linux community do not expect to pay for software. All of this makes it commercially very difficult to operate in the Linux environment with a niche product such as our touch screen driver products.

These notes refer to our Linux driver, supplied in binary form, and is mainly aimed at the non technical Linux users or larger commercial organisations that need a tried and tested touch driver solution that comes with comprehensive support. To this end this driver offers a generalised installation package with basic system requirements and in many cases should work 'out the box', especially with the main stream Linux distributions. If you are a Linux technician, with access to open source touch driver code that you can modify and make the necessary system configuration changes then UPDD is unlikely to match your specific requirements particularly as there are license costs involved.

Further, given the number of Linux distributions which undergo constant updates and modifications, if UPDD has any issues on distributions not listed as tested we may not be readily able to offer free support as the amount of support needed does not match the commercial viability of Linux sales.

License Notice

The software is licensed software and as such requires a license per system when the production version of the software is installed. Production software is either supplied by pointer device manufacturers or system integrators (who are entitled to distribute the driver) or is available directly from Touch-Base sales.

Evaluation version restrictions

Evaluation software, which is available from our Download Centre is mainly used to test the touch function is working, has certain restrictions:

- When working in 'mouse emulation' mode clicks do not work after 50 touches, only movement, at which point a nag screen is shown. Selecting OK on nag screen offers a further 50 clicks.
- The nag screen is processed by the driver's daemon task, aidaemon, so if this is not running the nag screen will not be shown and the counter will not be reset.

Copyright Notice

Deliverables

Unless otherwise requested, the software is distributed via email as an HTTP link that references a single compressed file linuxupdd_n.n.n_nn.tgz, where n.n.n = version number (e.g. 05.01.1124) and nn = 32 or 64 to indicate processor architecture.

General Notes

You should be aware that it is possible to load multiple X sessions on a machine and switch between them as required. At any one time an X session will be active on the desktop and the other X sessions will be in the background. The driver can be installed from any X session. Each X session running will receive pointer movements and click requests which might not be desirable as you will not know what is being selected in the hidden, background, X sessions. We believe that in a touch environment most users will only run with one X session and therefore we do not think this is a significant issue. Please contact us if this causes any problems, as it may be possible for us to limit mouse movement and clicks to the active X session if you must run in a multiple X session environment.

System Requirements

The driver runs in user space and is not a kernel mode driver and therefore does not have any dependency on a specific version of the kernel. However, the driver does have certain system requirements:

Processor	X86 – 32 bit	Fully supported and tested in-house
	X86 – 64bit	Fully supported and tested in-house
Interfaces	These interfaces are used by the driver to post touch data into system:	
uinput	Required if using the driver's Virtual Device interface Uinput is a linux component that allows client processes to create virtual input devices.	
X	Required if using the driver's X interface X is the standard windowing system used in Linux. When running linux in GUI mode this will always be present and the driver can use a direct interface to X to support touch in cases where uinput support is not available or incomplete.	
File utilities	mkdir, cd etc, Sound utility sox if using calibration beeps, see Hardware requirement below.	

Libraries	glibc	UPDD 5.1.x is dynamically linked with the C run time library version 2.11.1 so this version or higher must be available on the target. This should be the case for most current distributions, but older distributions might only support an older version in which case UPDD version 5.0.2 should be used.
Hardware	Sound card	If calibration beeps are enabled in the UPDD Console a sound card is required as we have not been able to access the internal PC speaker under Linux.
	Ports	UPDD uses components of the kernel to provide access to the various hardware ports, such as USB (via libusb1.0), PS/2 and serial ports. In order to access controller hardware using a distribution that does not implement or mount these sub-systems by default the integrator will need to use the kernel documentation for the distribution in question to enable the appropriate interface.
Access to TCP IP port 4141	The driver requires access to TCP IP port 4141 for internal computer processing only.	

If you need a driver for other environments, such as a different graphics manager or different processor we can update our driver to match your requirements but there will be a cost involved on a time and materials basis.

Distributions

With 5.1.xxx we have introduced a new distribution identification process to minimise the user interaction during install. When installing on a recognised distribution the associated installation script is invoked. For unrecognised distributions a generic install script is invoked that should work for most modern Linux distributions but this cannot be guaranteed. ***If a distribution is not recognised by the installer it lists the distribution identification details. Please let us know what is listed so we can set up an appropriate installation script***.

As of Dec 2014 we have tested on the following distributions:

Distribution	Name	Release	UPDD	Date	Tested	Interface	Notes
Ubuntu	Raring Ringtail	13.04	5.1.08xx	April 14	In-House	uinput	Should work for release 10:04 and above
	Saucy Salamander	13.10	5.1.08xx	April 14	In-House	uinput	
		14.10	5.1.1124	Dec 14	In-House	uinput	
openSuse		12.x	5.1.1124	Dec 14	In-House	X	Tested 12.2 and 12.3
Debian	Squeeze	6.x	5.1.08xx	April 14	In-House	uinput	
	Wheezy	7.x	5.1.08xx	April 14	In-House	uinput	
CentOS		6.x	5.1.08xx	April 14	In-House	X	Tested 6.2 and 6.5
Fedora Core	Beefy Miracle	15 - 21	5.1.1224	April 14	In-House	uinput	Should work for release 15 and above
Others							
Arch Linux			5.1.1069	Sept 14	Customer	uinput	5.1.1069 failed to work but the old 5.0.2 did. This implies that for 5.1.xxxx to work either switch to the old Xorg interface or install uinput modules.
Cisco-Edge	Bluebird	1.1	5.1.1124	Dec 2014	Customer	uinput	Derivative of FC16?
Mint	Petra	16	5.1.1124	Dec 2014	In-House	uinput	

Important Note – The table above shows the list of distributions that have been tested or reported as ok. However, given the number of Linux distributions it is impossible to guarantee that our install and/or driver will work in all environments or in all distributions, even if the same underlying kernel, X interface and desktop manager is in use or if a distribution is based on one of the supported distributions. To this end we have selected a number of main stream distributions to regularly test our latest driver and to try and keep the driver current, these being ubuntu, Fedora-Core, CentOS/Redhat and openSuse. Issues with our driver in any other distribution may incur a cost to investigate.

Linux Notes

We have observed some general and distribution specific issues that are documented below.

If you are using the graphical install on 'tested distributions' then, where possible, we have catered for the known issues in the individual Linux distribution setup scripts.

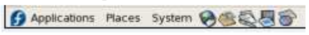
If you are manually installing, or installing on untested distributions or there are issues after install on tested distributions then please read the appropriate section below for your distribution:

General — **Description**

Usability

Window management issue

Generating a click via touch is sometimes ignored. In some areas of the Linux menu system we have found that unless the mouse is moved to the position of the click the click is ignored, such as the 2nd level of menus from the Application start menu option:



Menu level selection. Some items within the applications menu will expand to a 2nd level menu. Lifting the stylus off the first menu and clicking into the second menu item *will not* activate the selected item. You need to slide your finger from the first menu to the second menu and lift the stylus off when over the desired item. This problem can also be seen with a mouse by having the mouse at the position of the second level item and clicking when it is shown. With no mouse movement prior to click, the click is ignored.

Calibration

Full screen calibration issues

Correct operation of the calibration requires the calibration screen to be displayed in full screen mode.

We are aware that full screen mode via the Windows Manager can be a bit inconsistent under X and there appears to be no sure fire way to implement a foolproof solution. By default the calibration program invokes full screen outside the Windows Manager. If this does not work as expected there is a setting, tbcalibusewm, to enable window manager support in tbcalib and if this also fails we offset the calibration icons as discussed below.

With this setting missing or set to 0 we bypass the window manager full screen function and position the calibration window over the full area of the screen. Please note that with this mode enabled we do not recalculate the icon positions as described above.

This setting can be changed with the command 'tbutils nodevice setting dw tbcalibusewm n' where n = 0 or 1.

When requesting full screen via the WM with some Linux distributions, virtualized environments or certain visual effect settings cannot handle the method used by our graphical calibration program to force full screen with unpredictable results.

To date we have seen this with VMWare tilizedation, Linux distribution Maemo and main stream distributions with certain visual effects enabled.

On systems where the full screen issue occurs, one of the following actions will be necessary:

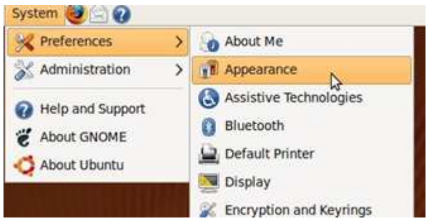
1) Configure the system to allow full screen

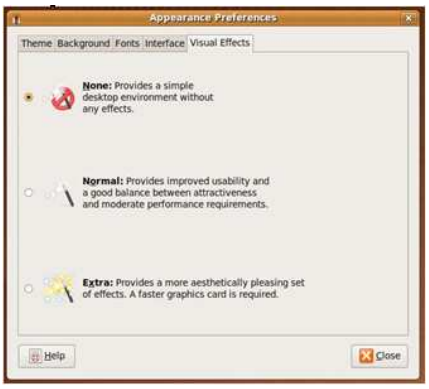
- For example if a window manager is preventing full screen mode try running without the window manager active. (This has been necessary on systems using the TWM window manager)

  or

- Where visual effects are preventing full screen temporarily disable the effect. We have had reports that on some Ubuntu systems the calibration screen is windowed (having a titlebar) instead of taking up the entire desktop. This is due to the desktop effects interfering with the window manager. If this effect is experienced then follow these instructions to temporarily disable desktop effects for the duration of the calibration.

Open the "Appearance" application as shown below Navigate to the "Visual Effects" tab





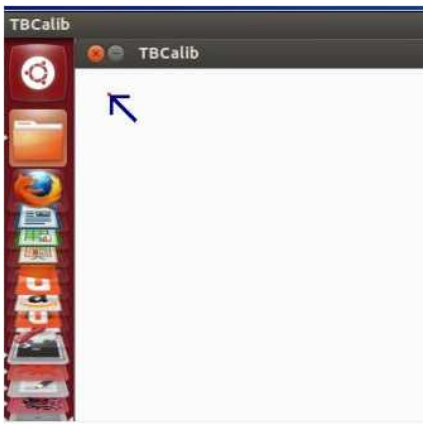
None: Provides a simple desktop environment without any effects.

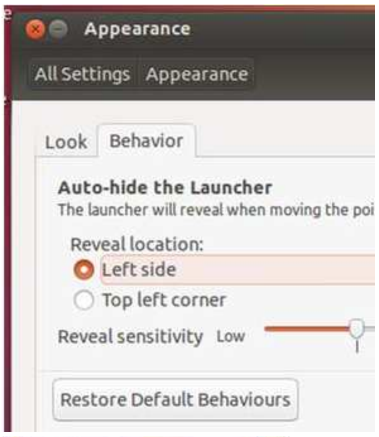
Normal: Provides improved usability and a good balance between attractiveness and moderate performance requirements.

Extra: Provides a more aesthetically pleasing set of effects. A faster graphics card is required.

If "Normal" or "Extra" is selected then select "None" and now try calibrating the touchscreen
When calibration is complete select the visual effect that was in use before and click "Close"

Here the auto-launcher is preventing the calibration screen covering the left-most side of the screen. This can be hidden to allow for calibration.



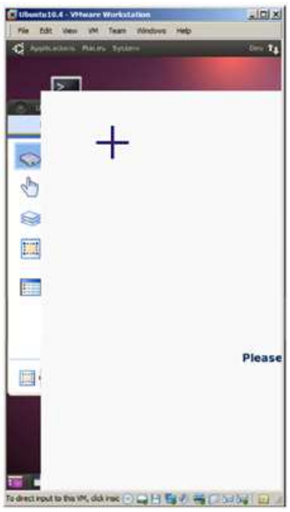


Auto-hide the Launcher
The launcher will reveal when moving the poi
Reveal location: [✓ Left side] Top left corner
Reveal sensitivity Low

2) Full screen not possible

In cases where full screen is not possible, such as the VMWare virtualized dialog to the right we have utilised UPDD calibration styles, named Custom2Point and Custom4Point, whereby calibration does not attempt full screen mode (and therefore cannot draw accurate calibration points) but instead requests that corners (2 or 4) of the calibration area be used as the calibration reference points. No other calibration patterns are catered for. Either request a build from Touch-Base with this calibration style pre-defined or use the UPDD Console, calibration dialog to add the style (deleting the existing style) and then calibrate to invoke corner calibration.

Where possible we try and calculate this offset and cater for this by adjusting the positioning of the crosses on the calibration screen.



In version 5.1.xx we have employed an updated graphics library and see some improvements in full screen support.

Changing desktop resolution

The driver does not automatically monitor desktop resolution and the resolution is captured at the point calibration is performed and used as part of the X interface. If resolution is changed a new calibration will be required.

`Driver priority`

We invoke a perl script 'tbrenice', which executes shortly after the tbupddwu process is launched and sets the priority of the driver components tbupddwu & tblinuxmouse to -5 (quite high).

The script is shown here for reference and the priority can be modified as required.

`Distribution`	**Description**
	No specific issues to report.

Touch devices
ELO

Description

ELO controller doesn't work with UPDD

Check the contents of /etc/X11/xorg.conf file for the following lines:

```
Section "InputDevice"
       Identifier  "elo"
       Driver        "elo"
       Option       "Device" "/dev/input/elo_ser"
       Option       "SendCoreEvents"  "true"
EndSection
```

If they exist then delete them and also delete the corresponding line:

"Inputdevice "elo""

in the “ServerLayout” section of your /etc/X11/xorg.conf file. The system should then be rebooted.”

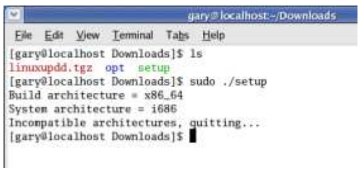
Linux devices	**Description**
Cisco DMP 4400G	X86 Linux system - UPDD must be manually installed on this device as per these instructions.
Cisco DMP 4310	MIPS Linux system - UPDD must be manually installed on this device as per these instructions.

CD Distribution

We believe the best approach is to expand the contents of the linuxupdd.tgz file to the CD ensuring the permissions on the extracted files are preserved. The end user should then be able to run the installer directly from the CD. Please test before creating many copies!

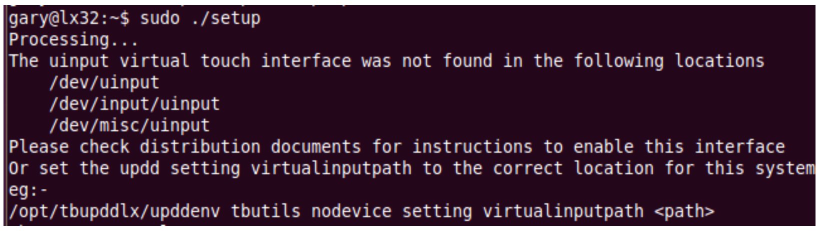
Installation notes

1. Please uninstall earlier versions of the UPDD software.
2. Different installs exist for 32 and 64 bit processors. Ensure you have the correct build to match your processor otherwise an error will occur. Installing 64 bit on 32 bit processor gives:



```
[gary@localhost Downloads]$ ls
linuxupdd.tgz  opt  setup
[gary@localhost Downloads]$ sudo ./setup
Build architecture = x86_64
System architecture = i686
Incompatible architectures, quitting...
[gary@localhost Downloads]$ 
```

3. By default the driver is configured to post touch data to the system via the uinput interface. The installer will check the common locations for this interface and will issue a warning if not found:



```
gary@lx32:~$ sudo ./setup
Processing...
The uinput virtual touch interface was not found in the following locations
    /dev/uinput
    /dev/input/uinput
    /dev/misc/uinput
Please check distribution documents for instructions to enable this interface
Or set the updd setting virtualinputpath to the correct location for this system
eg:-
/opt/tbupddlx/upddenv tbutils nodevice setting virtualinputpath <path>
```

4. The driver software has a number of dependencies, such as the NAS sub system for generating sounds on touch or during calibration. If these are missing the install may fail and display

   $sudo ./setup
   Processing
   Install failed

   In this case run ‘sudo apt-get install –qq nas’ to install nas and then try the install again.

5. **Customised settings**
   The setup package will create UPDD settings file tbupdd.ini. This contains the default driver, controller and calibration settings. To install with user defined custom settings we now allow end users to customise settings in a file - extra.ini. This file was previously embedded in the setup for customised builds. On Linux systems the extra.ini file can be embedded in the installer (tgz) file with the path ./opt/tbupddlx/extra.ini allowing most settings to be customised by an integrator.

   This will allow, for example, calibration data to be created (tbcalib dum4tba) from a calibrated system for use in subsequent installs.

6. **Utility shortcuts (symlinks):** If the distribution uses KDE or Gnome windows manager the installation procedure will ***attempt to***** create desktop shortcuts to UPDD utility programs. ****** The method used to create the shortcuts may not work in all distributions

   When the system has restarted and user log in is complete there are a number of ways to configure the driver.

   - There ***may be*** two new icons on the desktop, Console and Calibrate, which can be used to change driver settings and calibrate the touchscreen respectively.
   - On systems where the icons are not seen then a Terminal program should be executed and the user should type “/opt/tbupddlx/upddconsole” to run the Console, and /opt/tbupddlx/upddcalib” to run the Calibration program.
   - Alternatively manually create Linux shortcuts to the UPDD Console and Calibration program.

Installation

Please check out installation notes above before starting install.

Installation of UPDD basically requires the driver files to be copied to a UPDD application folder, installing library files, creating control files in the appropriate locations, creating shortcuts to the calibration and settings utilites and auto launching the various processes.

UPDD provides a number of installation scripts to perform the above and these are invoked by the GUI installation program. These scripts provide tailoring of the default package to the specific requirements of a given distribution

As Linux is very flexible and can be used in many different ways, these scripts are intended for mainstream distributions, where a standard Linux installation is performed, with default selections. If you use a non-standard installation you might need to implement a strategy to run the updd software as required. The post installation scripts are provided in the updd installation package and these can be used as a guide. You can of course repackage these to provide a customized UPDD package.

In most cases you will need copy the contents of the UPDD package to a suitable folder location and perform the following:

- Auto start Tbupddwu – the user mode driver
- Auto start aidaemon – the driver’s daemon process used to detect rotation automatically and adjust calibration.
- By default the driver interfaces to the system via the Virtual Device touch interface. Only if this interface does not work or is unavailable do you also need to launch the user mode X interface module tblinuxmouse.
- If required, create desktop shortcuts to UPDD Calibration and UPDD Console
- Register the QT4 library

This section describes the different installation options and the changes made to the system as part of the installation and covers the following topics:

GUI installation	Install the driver using the supplied GUI installation program and install script
Installation notes	Installation notes
Standard manual install	Manual installation steps for standard Linux distributions
Embedded manual install	Manual installation steps for embedded (cut down) distributions

Specialist installs	Cisco DMP 4400G	UPDD must be manually installed on this device as per these instructions.
	Cisco DMP 4310	UPDD must be manually installed on this device as per these instructions.

GUI installation Procedure

The main installation package is held within the compressed file called linuxupdd_nn.nn.nnn_xx.tgz, where nn.nn.nnn is the UPDD release number and xx is the processor support, 32 or 64 bit. Please ensure you have the correct setup file to match your processor's architecture.

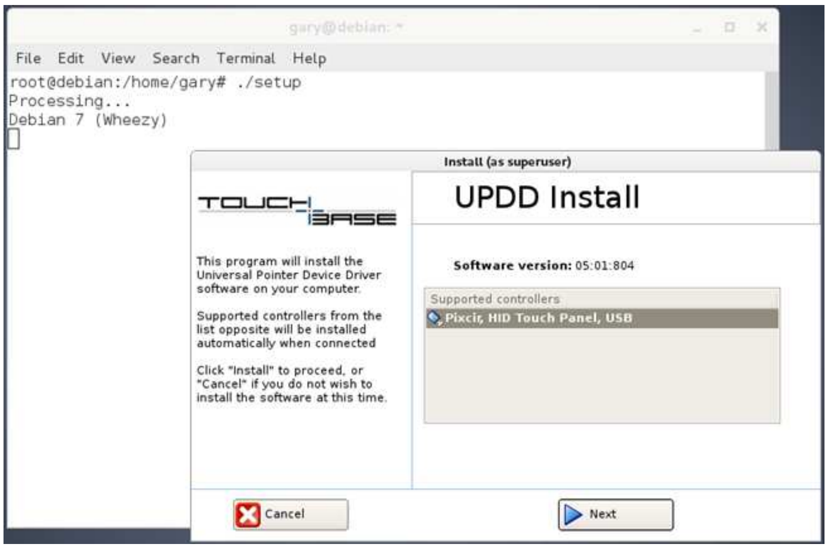
Copy the file into **a directory other than "/opt" or "/"** such as a users home directory on the Linux system, change to that directory, then decompress it by using the command "tar zxvf linuxupdd_nn.nn.nnn.tgz".

Installation of the driver must be performed as the "root user". It is usually possible to run with root privileges by typing command "su root" or preceding the setup command with "kdesu" or "sudo" as described here but we have found on some systems this is not sufficient in which case you must log on at system start as root. Note that on some systems "sudo" does not allow the installer to work and you must use either "su" or, "kdesu". In cases where the installer fails because the installer dialogs do not open try the command 'Sux' as this utilises a wrapper around su that transfers a user's X credentials.

Important - Do not decompress the software in the root directory or "/opt/" as the install will fail.

To install the software open a terminal window and "./setup" as a root user, see above. This will launch the setup program.
Important note: the 'setup' program is extracted directly into the extraction directory and is not located in the ./opt/tbupddlx subdirectory just created.

The Setup program will attempt to identify the distribution and if recognised will invoke the associated installation script as seen below. In this case the installer has recognised Debian 7:



```
root@debian:/home/gary# ./setup
Processing...
Debian 7 (Wheezy)
```

Install (as superuser)

UPDD Install

This program will install the Universal Pointer Device Driver software on your computer.

Supported controllers from the list opposite will be installed automatically when connected

Click "Install" to proceed, or "Cancel" if you do not wish to install the software at this time.

Software version: 05:01:804

Supported controllers
Pixcir, HID Touch Panel, USB

To complete the installation:

- From the device list dialog select the touch device or, if it is a PnP device (USB), optionally leave the PnP mechanism to pick up the device(s).

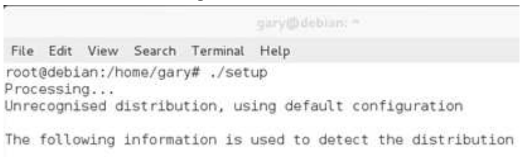
Install scripts can invoke question dialogs, such as 'OK to enable USB file system'. Answer these dialogs as appropriate.

For general installation issues please refer to the Installation notes below.

Once the install has completed and all being well the touch should now be active albeit not calibrated. If the cursor does not react and calibration fails then try rebooting the system. If touch is still not working after install and reboot please refer to the troubleshooting section below.

Unrecognised distributions

If the install does not recognise the distribution it will show the identification information it found.



```
root@debian:/home/gary# ./setup
Processing...
Unrecognised distribution, using default configuration

The following information is used to detect the distribution
```

For unrecognised distributions a generic install script is invoked that should work for most modern Linux distributions but this cannot be guaranteed.

Please email us the distribution in use and the identification information listed to technical@touch-base.com and indicate the success or otherwise of the generic script. Based on this information we will create a detection script for the distribution.

Script utilsation

The structure of the installation scripts are documented here. Should the installation fail you may be able to create you own scripts using these as a template. and re-embed in the compressed software package. If possible send us a copy so they can be utilised in future installations.

Manual Install

We highly recommend you install using the GUI installer. If for any reason this is not possible we have documented the steps needed to manually install the driver in both a standard Linux distribution and in a minimal embedded environment.

Driver Notes

General

The driver is installed such that should it crash it will automatically be restarted.

For serial devices the driver handles the serial device via standard COM port names (/dev/ttySnn) or USB to serial adaptors (/dev/ttyUSBn), so to use a serial device with a different name it is required to create a symbolic link to one of these port types. In the UPDD serial dropdown you have a choice of COMn for hardware serial ports (maps to /dev/ttysnn-1 (e.g COM1 = TTYS0)) or AdaptorN for virtual serial ports (maps to /dev/ttyUSBn-1 (e.g Adaptor1 = TTYUSB0)) created via a Serial to USB adaptor.

Following install, all configured UPDD devices will be set to control the pointer on the first monitor. In a multi monitor environment invoke the UPDD Console - Hardware dialog to configure multi monitor and/or touch devices. The UPDD Console can also be used to add additional non PnP devices (e.g serial) after installation, PnP devices (e.g. USB) will automatically be discovered. See the Multi monitor and multi device documentation for further information.

Spawned processes

Following installation the following processes should be seen from the command "ps aux | grep tb":

```
root     1209  0.0  0.0  11324  1360 ?       S    08:32   0:00 /bin/sh /opt/tbupddlx/startupdd
root     1211  0.0  0.0   3780   296 ?       Ss   08:32   0:00 startpar -f -- tbupdd
root     7477  0.4  0.3  97064  7156 ?       Sl   08:59   0:00 /opt/tbupddlx/tbupddwu
canto    7635  0.1  0.3  88720  7076 ?       Sl   08:59   0:00 /opt/tbupddlx/aidaemon
```

Folder structure

Following installation the following folder structure will have been created/updated on your Linux system:

/opt/tbupddlx/*

Contains the ini file, calib gif files, etc

/usr/local/lib or /usr/lib

Libraries used by the driver
libtbapi32.a
or
libtbapi64.a
libACE.so.5.6.2
libQt3Support.so.4.8.1',
libQtCore.so.4.8.1',
libQtGui.so.4.8.1',
libQtNetwork.so.4.8.1',
libQtSql.so.4.8.1',
libQtXml.so.4.8.1',
/etc/init.d/tbupdd
/etc/rc2.d/S90tbupdd
/etc/rc3.d/S90tbupdd
/etc/rc5.d/S90tbupdd
These automatically load the daemon on system boot.

Desktop and application touch interaction

The driver receives all incoming touches from the touch screen which are then dispatched to the system and applications and depending on the touch interfaces being used will be processed in various different ways.

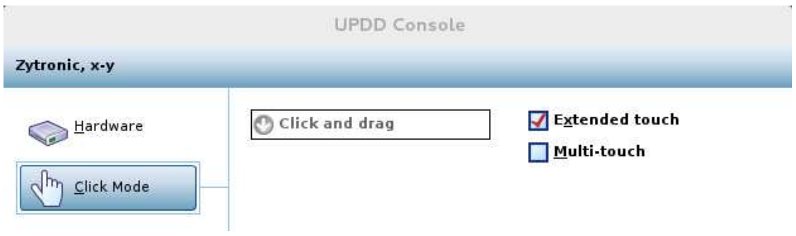
Depending on the distribution in use the driver will either post touch data to the single touch X interface or the single touch uinput interface and this should result in touch movement when the screen is touched due to the X graphical interface processing the co-ordinate and pen data. In many cases this will satisfy simple touch usage. However, there are a number of touch interfaces that can be utilized with our driver, as described below:

Interface	Brief description
Mouse emulation	Default interface that offers the most basic touch interface. The touch interaction is due to the touch screen emulation mouse functions such that a touch on the screen is emulating a mouse pen down, mouse movement and pen up. The standard methods are to either post data via X (the Linux graphical layer) or post data into the kernel input subsystem, either from a kernel driver or a user mode driver. The UPDD Linux software implements two such system interfaces as described in greater detail below.
Gestures	There is no specific UPDD supplied gesture software under Linux. Any users requiring gesture control will either have to utilise a gesture aware Linux desktop manager, such as Unity, or build their own application gesture aware application, such as a QT app using QT gesture functions.
Touch aware applications	There are many ways to drive an application via touch. It could be via the simple “mouse emulation” interface to more direct touch specific interface as implemented by the integrated development environment. There are other interfaces, such as direct file access to touch hardware (special device files), TSlib, Evdev, Evtouch – there may be other Linux application development kits. Specific development libraries such as QT, SDL Multimedia Library , Clutter Library , Pigment Toolkit, GTK etc, may also dictate the touch interface in use. For example, applications written with SDL or Clutter may utilise the TSlib interface if direct access to the touch device is required by the application. If using a library interface it is important that the underlying touch event is satisfied by one of the UPDD touch interfaces or the touch event will not be satisfied. It is also important to ensure that touch events can be shared such that a UPDD posted touch event is not blocked by a high level request e.g. a touch aware Windows Manager makes exclusive use of the touch and the event is not available to underlying applications.
TUIO client applications	Allows TUIO Client applications to receive touches via the UPDD TUIO server interface.
UPDD API applications	Allows UPDD API based touch applications to receive touches via the UPDD API.

System interfaces

There is no single standard for posting touch data into the Linux kernel as there are a number of different ‘touch interfaces’ available.

Currently, the driver utilises two system level interfaces to post touch data into the system; a **Virtual touch device interface** that post data to the kernel input subsystem or the **user mode X interface** to post touch data to X. The UPDD Console, Click Mode dialog indicates the interface to use:


UPDD Console
Zytronic, x-y
Click and drag
Extended touch [✓]
Multi-touch [ ]

The settings dictate the interface to use as follows:

Extended touch disabled	X interface
Extended touch enabled	Virtual device, Single touch
Extended touch and multi-touch enabled	Virtual device, Multi-touch

Both interfaces are further described below.

Virtual Device with both single and multi-touch interface

UPDD version 5.1.x implements a virtual touch device to post touch data. This interface method should satisfy both X and touch event driven services and applications and fits in to the Linux input stack schematic as follows:

The Linux input stack	Description
Applications	End user multi-touch applications
Multi-touch libraries	Development tools that utilise Linux touch event interface, such as GTK+, QT5, JavaFX, Kivy, QTWebKit, uTouch, etc
Xorg	Linux graphics layer
Linux kernel	Linux kernel with multi-touch capabilities (uinput, evdev etc)
Virtual touch device	UPDD created virtual touch device – with multi-touch capabilities
UPDD driver hardware interface	UPDD USB, PS/2 or serial interface
Touch Hardware	UPDD supported single and multi-touch hardware

The Virtual Touch Device replies on 'uinput' support being part of the Linux kernel. Uinput offers a user mode bridge to the Linux input subsystem and was introduced in Linux kernel 2.6.xx with multi-touch support introduced in approx 2011 with kernel release 2.6.3x and should be available for most standard distributions. The install script (setup) will print a message on the console if uinput is not available and may need to be installed. The driver creates either a single touch or multi-touch virtual device depending on the UPDD Console, Click Mode, Extended touch settings. The driver also creates an additional virtual device to pass right click requests when using UPDD Interactive touch click mode of operation.

Using the **lsinput** command you can view the UPDD virtual input device(s). *On most systems LSinput is **not** installed by default. Under ubuntu/debian **lsinput** is part of the **input-utils** package. To install it, run the command "sudo apt-get install input-utils". Under other Linux distributions the command for installing LSinput may be different.*

Now run 'sudo lsinput' to list the devices (*note the event number will vary between systems*):

Virtual device for posting touch data

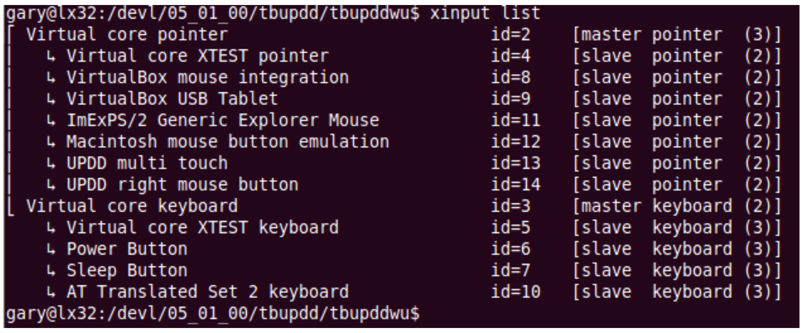
```
/dev/input/event13
   bustype : BUS_VIRTUAL
   vendor  : 0x1
   product : 0x1
   version : 1
   name    : "updd multi touch"
   bits ev : EV_SYN EV_KEY EV_ABS
```

Virtual device for posting right click events (since build 1069)

```
/dev/input/event14
   bustype : BUS_VIRTUAL
   vendor  : 0x1
   product : 0x1
   version : 1
   name    : "updd right mouse button"
   bits ev : EV_SYN EV_KEY EV_ABS
```

Note: When using UPDD click mode 'Interactive Touch' a second device is utilized to process right clicks. Using the primary virtual touch device to post both touch data and right clicks caused issues in some distributions which were not seen when using separate devices.

If all is ok then X will recognise the virtual devices and this can be confirmed by using the "xinput list" command
As shown in this example two UPDD devices should be listed. If the driver is configured to use a single touch virtual device then the "UPDD multi touch" device listed below is replaced by "UPDD single touch"



```
gary@lx32:/devl/05_01_00/tbupdd/tbupddwu$ xinput list
⎡ Virtual core pointer                          id=2    [master pointer  (3)]
⎜   ↳ Virtual core XTEST pointer                id=4    [slave  pointer  (2)]
⎜   ↳ VirtualBox mouse integration              id=8    [slave  pointer  (2)]
⎜   ↳ VirtualBox USB Tablet                     id=9    [slave  pointer  (2)]
⎜   ↳ ImExPS/2 Generic Explorer Mouse           id=11   [slave  pointer  (2)]
⎜   ↳ Macintosh mouse button emulation          id=12   [slave  pointer  (2)]
⎜   ↳ UPDD multi touch                          id=13   [slave  pointer  (2)]
⎜   ↳ UPDD right mouse button                   id=14   [slave  pointer  (2)]
⎣ Virtual core keyboard                         id=3    [master keyboard (2)]
    ↳ Virtual core XTEST keyboard               id=5    [slave  keyboard (3)]
    ↳ Power Button                              id=6    [slave  keyboard (3)]
    ↳ Sleep Button                              id=7    [slave  keyboard (3)]
    ↳ AT Translated Set 2 keyboard              id=10   [slave  keyboard (3)]
gary@lx32:/devl/05_01_00/tbupdd/tbupddwu$
```

Uinput notes

1) It is important to acknowledge that the virtual device implements a solution for native multi-touch support ***not*** MPX. MPX provides multiple independent pointers (mouse cursors) at the windowing system level.

2) Multiple interfaces via the uinput device
The uinput device, when configured for multi-touch use, is used to post both single and multi-touch data such that both types of system requests are satisfied. Single touch data is posted in a different manner to multi-touch data (even when a single touch is in use) so all touches result in two posting on the uinput device to satisfy both single and multi-touch system requests.

   UPDD can use uinput to create either a single touch or a multi touch virtual device. The single touch device is useful on systems that do not recognise, or have problems working with a multi touch device. In practice uinput support varies even on some recent distributions, so on some distributions the multi touch device does not work at all (e.g. Ubuntu 10:04 LTS) and some do not even support the single touch mode (e.g. Fedora). For this reason UPDD allows switching between single touch and multi touch uinput devices and a direct X (single touch only) interface.

   For distributions we have tested an appropriate default will be set by the installer.

Multi-monitor consideration

When using uinput in a multi-monitor configuration we announce an axis in the full range needed to cover all the configured monitors and then send touch events to the co-ordinate range appropriate to the monitor position. E.g. if a range of 0..1000 covers a single screen then the range of a 2nd screen is set to 1000..2000 and the posted events in this range map to the 2nd screen. It is our understanding that for this to work correctly you require X 1.13 and above.

Multi-Touch testing

Multi-touch support within a **Windows Manager** will be dependant on the level of support built into the Windows Manager. Our tests were conducted with ubuntu and the gesture aware Unity Windows Manager (https://wiki.ubuntu.com/Multitouch). In this distribution device test software showed the touch device and reported single and multi-touch input and both X and Unity gestures worked as expected.

Testing in openSuSe 12.2, kernel 3.4, the supplied test software did not show single or multi-touch input albeit the single touch X interface worked well which implies that this interface is likely to be working in this distribution.

Multi-touch support within an **application** is very much dependant on how the multi-touch support has been implemented:

Implementation	Comments
UPDD API	Touches will be received directly into the application from the driver via the TBAPIRegisterDataCallback function.
TUIO Client Application	Touches will be read by the UPDD TUIO Server (using the UPDD API) and broadcast to any TUIO Client applications
Specific MT Application	Specialist multi-touch library based development using touch specific function or gesture calls, such as QT touch events or gestures. To work with UPDD the underlying touch event implemented by the library must be satisfied by touch data being posted via the Linux input sub system (uinput) interface. If using QT it is our understanding that you need QT 5 and above for fully implemented multi-touch support under Linux.
Other	Please contact us to discuss further additional MT application touch support that has been implemented which is not satisfied by the existing UPDD touch interfaces.

In our tests to show the multi-touch mechanics are working as expected we tested the simple QT based multi-touch drawing and gesture application, referenced above, compiled under QT5 and run on ubuntu 13:10. With this combination it worked as expected and proved that the UPDD data input to the uinput interface 'works' as expected.

Currently under Linux there is no definitive definition of multi-touch support other then to say it can be achieved if the right components are in place.

There are utilities, such as evtest and mtdev-test, which can be used to show certain device information handled by the multi-touch device.

User mode X interface

X is the standard windowing system used in Linux. When running Linux with a graphical interface this will always be present and the driver can use a direct interface to X to support touch in cases where uinput support is not available or incomplete.

We have discovered that uinput is not always available or is not a standard component on some distributions and therefore this interface can be used as an alternative touch interface. In earlier versions of the driver we utilised a module tblinuxmouse to implement the X interface but it proved difficult to find a generic method that worked in all distributions to load at startup. With the release of version 5.1.1124 we now utilise a script /opt/tbupddlx/xhostenable which allows the X interface to work. The xhostenable script is invoked from our daemon process 'aidaemon'. It should be noted that when utilising X interface touch is not available until the daemon process is running and the xhostenable script has been executed, therefore touch does not work on the login screen in the current implementation. Because of this limitation we have set the uinput interface as the default for distribution where it is known to work only defaulting to X if the distribution does not, by default, contain the uinput modules.

A Linux system administrator can take the contents of the xhostenable script and run it during system startup if touch is required earlier in the system startup sequence.

Important security note. The X interface requires that the driver (tbupddwu) has access to the X session. This process runs under the local root account. In some scenarios this might be considered a security risk. This access is set by the xhostenable script

To prevent this mode of operation this script can be deleted, or edited to implement an alternate strategy. In the default implementation tbupddwu runs in the root account to allow access to the libusb interface. It might be possible in some situations to set up access to libusb without root access.

Application interfaces

In most cases you will be using X or the virtual device interface to interact with the desktop or specific applicationm. However, if you are using an alternative interface, e.g. TUIO or UPDD API, and neither standard system interface is required then you can disable the 'mouse port interface' or indicate that the incoming data stream is not to be processes as touch data as follows

Disable mouse port	upddenv tbutils global setting InitialMousePortEnabled 0
Redefine data processing	upddenv tbutils get packetdisposition This will return n.n.n.x where x = T or B (T = Touch / B = Both) upddenv tbutils setting dw packetdisposition n.n.n.D (Data only)

Summary

The driver receives all the touches from the touch device(s) which are then available on various interfaces.

Using the UPDD Virtual device component all touches will be posted to the Linux input subsystem via the uinput interface. This input method will satisfy most pending touch events, including X (mouse emulation), hence is the driver's default system interface.

Using the X interface a single touch will be fed to X graphical layer to move the mouse cursor.

Using UPDD TUIO server all touches will be broadcast to TUIO client applications

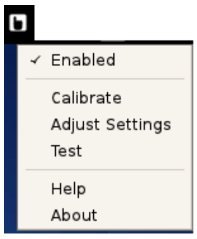
Applications using the UPDD read touch API will receive touches.

Hardware	Driver	Interface	Satisfies	Type	Example
Pointer Device Hardware (Touch Screen)	UPDD Driver hardware interface	UPDD Virtual Touch Device post data to uinput	All pending touch event requests	Mouse driver apps Touch driven apps	Desktop and standard application control Windows Manager Gestures (e.g Unity) Simple multi-touch drawing application that receives touch data via the QT Touch Event API, QTTouchPaintDemo. Also uses QT's gesture recognition functions to calculate gestures and display results. UPDD test program (UPDDdraw) – mouse mode
		X post data via X request	Mouse emulation	Mouse driven apps	Desktop and standard application control UPDD test program (UPDDdraw) – mouse mode
		UPDD API Post data to TBAPIRegisterDataCallback	All pending touch reads via UPDD API	UPDD Client apps	UPDD test program (UPDDdraw) – direct mode Simple multi-touch drawing application that receives touch data via the UPDD API, QTUPDDPaintDemo.
			TUIO Server	TUIO Client apps	Simple multi-touch drawing application that receives touch data via the TUIO interface, QTTUIOPaintDemo. Demos at the TUIO web site

Driver Utilities

System tray item

Introduced in release 5.1.x the UPDD Menu bar item offers links to the main driver utilities:



Double click on the icon to invoke the UPDD Console or single right click to show the menu:

Indicates if the driver is to process the touch data for the device – use with caution if touch is the only interface!

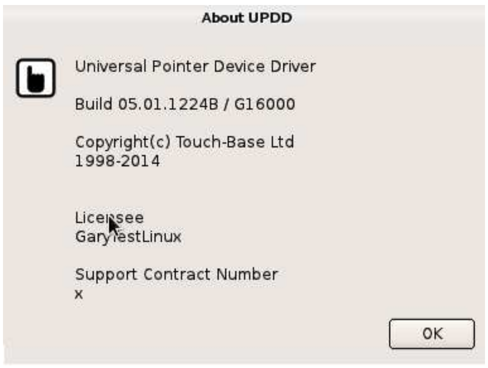
Invokes the touch screen calibration program, UPDD Calibration

Invokes the driver settings program, UPDD Console

Invokes the driver test program, UPDD Draw

Invokes the help system

Driver version information, Licensee name and Support Contract Number:


About UPDD

Universal Pointer Device Driver

Build 05.01.1224B / G16000

Copyright(c) Touch-Base Ltd
1998-2014

Licensee
GaryTestLinux

Support Contract Number
x

In some Linux distributions, specifically Ubuntu and it derivatives, you may need to follow these directions to get the UPDD system tray icon to appear:

http://techpublic.net/linux/show-selected-apps-in-ubuntu-unitys-system-tray-with-dconf-editor-how-to/

Calibration

Calibration is a procedure used to align the pointer device with the graphical display area or desktop segment. When using the pointer device the mouse cursor should normally position itself under the stylus when it is in contact with the pointer device. If this is not the case then calibration will be required and can be invoked in a number of ways:

1) UPDD system tray, Calibrate option (see above).
2) UPDD Console, Calibrate Option

3) Invoke from a terminal window and run the command /opt/tbupddlx/upddcalib
or
cd /opt/tbupddlx
./upddcalib
4) Create a link to it (/opt/tbupddlx/upddcalibrate) using the window manager:

and touch the calibration crosses, or arrows, as they appear.

Calibration notes:

- If the calibration screen does not cover the full desktop area see calibration notes above. Full calibration procedure information can be found in the Calibration document.
- Calibration can be performed in any screen resolution and the calibration data held is relative to the screen resolution. However, if the screen resolution is changed then a new calibration will be required as currently we do not automatically track screen resolution changes.
- If calibration cannot be activated when running as a root user then it is likely the user does not have permission to connect to the X server and as a consequence cannot run graphical programs.
The problem is that when starting an X session, the user is authenticated by X and given permission to connect to the server. When switching users by using "su", the new user no longer has the correct permission.

  To overcome this type “xhost SI:localhost:root” or “xhost +” before running su to switch users.

Driver/Device settings – the UPDD Console

The UPDD Console defines the functionality of the pointer device(s) and the UPDD driver environment and can be invoked in a number of ways:

1) Double click on the UPDD system tray option or single right click and select the Settings option.
2) Invoke from a terminal window and run the command /opt/tbupddlx/upddconsole
or
cd /opt/tbupddlx
./upddconsole
3) Create a link to it (/opt/tbupddlx/upddconsole) using the window manager:

and change the settings as required.

See the UPDD Console documentation and on-line help for further information.

All settings, including calibration data, are held in the UPDD settings file.

Test program

A simple test program is supplied to test the performance and accuracy of the touch device as handled by the UPDD driver. It can be used to view stylus input from single and multi-touch devices.

Command Line interface

A command line utility is supplied that can be used to perform certain configuration or device type functions.

Serial port notes

Change serial port connection

The UPDD Console - Hardware tab allows the COM port name to be reassigned after installation.

Standard serial ports

If using a standard serial port, Select Com1, Com2 etc in the COM port selector. Serial ports should be registered in the system as ttySn which is mapped to our driver to COMn+1 (e.g. COM1 = ttyS0)

Serial to USB adaptors

If using a serial to USB adaptor, select Adaptor 1, Adaptor 2, etc in the COM port selector. This has only been tested with the Keyspan adaptors so far but the Linux documentation states that the interface is the same for all serial adaptors; hence UPDD should work for all serial/USB adaptors. Serial to USB adaptors should be registered within the system as ttyUSBn which is mapped by our driver to Adaptor n+1. (e.g Adaptor1 = ttyUSB0)

Serial port reassignment

The driver handles serial devices via standard COM port names (/dev/ttySnn) or USB to serial adaptors (/dev/ttyUSBn), so to use a serial device with a different name it is required to create a symbolic link to one of these port types.

Example: Assuming you have a serial port referenced as ttyC1P3 to be reassigned. You need to open a terminal with root privileges and type the following:

ln -s /dev/ttyC1P3 /dev/ttyUSB0 (for Adaptor 1)

or

ln -s /dev/ttyC1P3 /dev/ttyS0 (for com port 1)

You will then need to open up the UPDD Console and change the COM port for your device to "Adaptor 1" or “Com 1”.

Serial port testing

Should the serial port connection not be working there are a number of procedures to follow to help identify the problem as described in the knowledge base article here.

Serial port issues

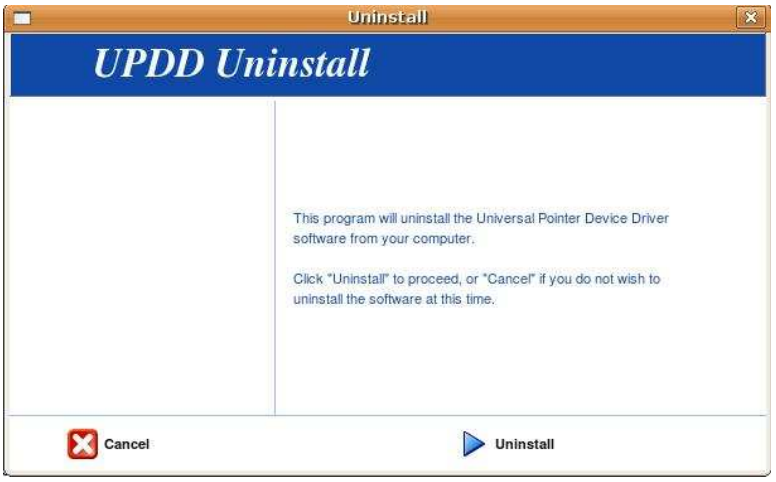
One customer reported that the serial touch screen stopped working after the initial boot. Further investigations showed that the root cause was the Bluetooth manager using the modem-manager to close all serial ports even though they had no Bluetooth devices on their system!!!

Uninstall

Automatic

Removal of the driver must be performed as the "root" user. It is usually possible to "become" the root user by typing command "su root" or preceding the setup command with “kdesu” or “sudo”, but we have found on some systems this is not sufficient in which case you must log on at system start as root. Note that on SuSE 10.3 and SLED 11 “sudo” does not allow the uninstaller to work and you must use either “su” or, optionally on SuSE 10.3, “kdesu”.

To uninstall the software open a terminal window and either use “su” or “kdesu” to become the root user then type “/opt/tbupddlx/uninstall”. Alternatively type “sudo /opt/tbupddlx/uninstall” and enter your password (not on SuSE 10.3 and SLED 11; see note above). This will launch the uninstall program:



This program will uninstall the Universal Pointer Device Driver software from your computer.

Click "Uninstall" to proceed, or "Cancel" if you do not wish to uninstall the software at this time.

To continue and remove the software click on “Uninstall”. To cancel, click “Cancel” and the system will remain unmodified.

Manual Uninstall

Type the following commands being careful to use the same case and spacing.

su
Enter the root password
rm -rf /tbupddlx *If this is mistyped, the whole system could be wiped.
rm /usr/X11R6/lib/modules/input/xf86_tbupddlx.o
For Systems using X11R6
rm /usr/X11R6/lib/modules/input/xf86_tbupddlx.o
For Systems using X11R7
rm /usr/lib/xorg/modules/input/xf86_tbupddlx_drv.so
rm /etc/rc.d/init.d/tbupddlx
rm /etc/rc.d/rc2.d/S90tbupddlx
rm /etc/rc.d/rc3.d/S90tbupddlx
rm /etc/rc.d/rc5.d/S90tbupddlx
Edit the file "/etc/X11/XF86Config-4" for Xfree86, or “/etc/X11/xorg.conf” for x.org and remove the following section:-

```
Section "InputDevice""
  Identifier "Updd0"
  Driver "xf86_tbupddlx"
  Option "Device" "/tbupddlx/comReadPipe"
EndSection
```

In the section that begins with:-
Section "ServerLayout"
Remove the line:-
InputDevice "Updd0" "SendCoreEvents"

Multi-monitor and multi-device support

Support for multiple monitors is covered in full in the multi-monitor and device document, Linux section.

Display rotation considerations

Linux supports rotated video modes for supported video cards under both Xfree86 and X.org. UPDD will work with rotated video and this is explained in detail in the separate rotate documentation.

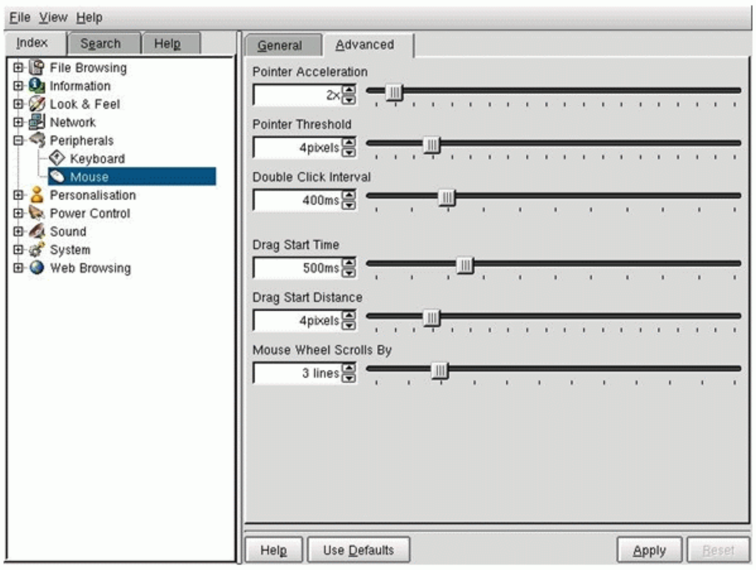
Display resolution / calibration considerations

The calibration mapping is based on the screen resolution setting at the time of calibration so if the resolution is changed the calibration will be inaccurate. To cater for this you will need to manually recalibrate after changing video resolution.

Future releases of the driver may well introduce a daemon process to automatically monitor video resolution and adjust automatically but until such times as this is available manual intervention is required.

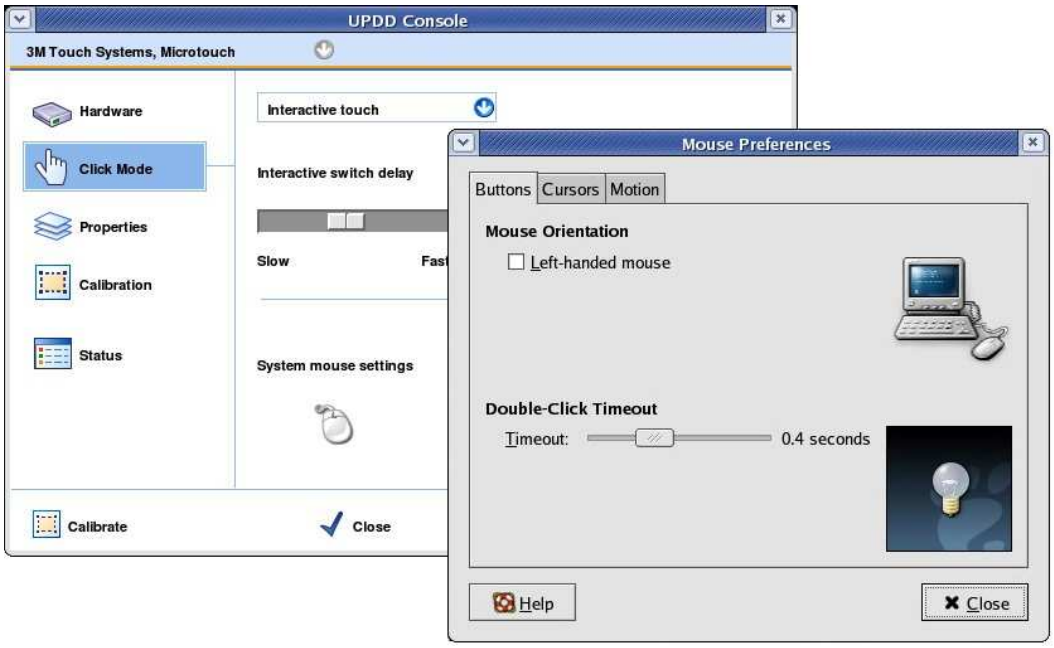
Mouse settings

Double click capabilities are affected by the system's Mouse settings. To achieve a double click using the pointer device these settings need to cater for the type of device in use. A touch screen may well require different settings to that required by a mouse. The main setting that affects the ability to double click is the double click speed. If this is set too fast it may be impossible to produce a double click. Ensure this is set to an appropriate value in the mouse settings to allow for double clicks via a stylus. In this example the mouse settings screen is from the KDE Control Centre. Other window managers will have different ways of configuring the settings.



Pointer Acceleration: 2x
Pointer Threshold: 4pixels
Double Click Interval: 400ms
Drag Start Time: 500ms
Drag Start Distance: 4pixels
Mouse Wheel Scrolls By: 3 lines

The UPDD Console, Click Mode dialog, System Mouse settings will invoke the Mouse settings for KDE and Gnome desktops, as shown in the following example:



Touch Utilities

Virtual Keyboards

A number of Virtual keyboards are available on the Web for Linux as detailed in the UPDD Virtual keyboard documentation.

Mouse Cursor

At the time of writing we are not aware of any specific end user utility to change the mouse cursor or turn it on/off. Please contact us if you find such a suitable utility that we can document for other users.

Troubleshooting

Following successful installation and reboot of the system (if requested/required) the device should respond to UPDD calibration (a moving cursor does not always indicate UPDD driver is in control of the touch device). The key elements to check are:

- The touch device is known to be working
- The device is plugged into the system and recognised by the OS.
- The driver is running
- The device is supported and is seen by the driver
- Touch data is being received from the device and made available on its API (driver utility program function as expected)
- The device is calibrated and associated with the correct desktop
- Touch data is being posted into the OS if using one of the standard input interfaces

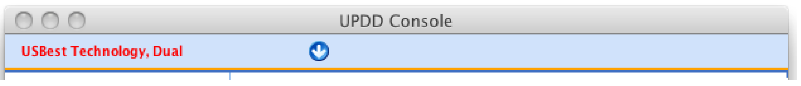
Follow these steps to help identify the problem:

Check the device is seen by the system

If using a USB device, ensure the device is plugged in and listed in the Linux devices and also identify if the device listed (by Manufacturer, USB Vendor id and Product id) is that expected by the driver.

Check the device is seen by the driver

The UPDD Console will list the controller name in black text if it has connected to the USB device or serial port.



The text 'No devices found' will be displayed if none of the devices configured or supported are found on the system. Red text listing can be because a serial port is incorrectly defined or not available or a USB device is no longer connected. If a USB device is plugged in it may not be supported by the driver or the device is being handled by another 3rd party driver, such as the standard HID driver.

If using a serial device ensure the serial port is defined in the system and correctly set in the UPDD Console, Hardware dialog.

Ensure no other 3rd party drivers are trying to handle the device.

If you have just installed, try rebooting, especially if no name is listed in the console.

Check the driver is running

Use the command "ps ax | grep tbupddwu" to determine that the driver is running.

If it is loaded, see if the UPDD Console displays the error 'No driver connection' this indicates the API interface between the applications and the driver is failing. This could indicate the driver is hung.

Check the driver is working

Invoke the UPDD test program and enter direct mode to see if touch data is being received by the driver and available on its API interface.

Check the device is calibrated and assigned to the correct desktop segment.

Defined the associated desktop (e.g. Monitor 1, Monitor 2 or Left half, Right Half etc) in the UPDD Console, Hardware dialog and then select calibrate.

Check interface components

If the system is configured to use one of the standard interface input methods to post touch data into the system ensure that the required components are present. If using the X interface ensure that the process tblinuxmouse (ps ax | grep tblinuxmouse) is running. If using the Virtual device interface ensure uinput is part of the kernel. Try switching between the interfaces to see if either work.

Check the hardware is functioning

Utilise another OS if possible to prove the touch hardware is OK, such as plugging a HID touch device into a Windows 7 or 8 system.

Known issues

No cursor movement, only clicks at current cursor location

If using Uinput Interface there is a possibility that the distribution does not like the method in which we post both single and multi-touch data into the system via the same virtual device. We believe this to be a bug in the kernel with older distributions.

Specifically, on Ubutntu 10.04 64 bit systems we have seen that only touch clicks are recognised, mouse movement (dragging) does not work and all clicks occur at the current mouse position.

Try switching between the interfaces to see if any work.

Current Limitations

UPDD was originally developed for Windows and has since been ported to other OS. Not all features have been ported to Linux, they include:

- Dynamic detection of system language.
- Serial port auto-detection
- Interactive touch – visual notification of right click count down
- Anchor mouse function
- Light pen calibration (the white lines on black mode)
- Toolbar actions

Contact

For further information or technical assistance please email the technical support team at technical@touch-base.com