TOUCH documentation BASE

Controller Support

Revision 1.4 – 1st March 2013
www.touch-base.com\documentation\technical

Technical info | UPDD listing | Hardware specifics | Data packet format | Firmware | EEPROM | Info sources | Contact

Our UPDD driver supports 100's of touch controllers, mainly USB, serial and PS/2 devices and we try to keep support current with new controllers as they are released or made available.

Technical information

To support a controller we either need access to a controller or technical details needed to configure support for the controller.

A separate document is available to advise how to identify a device connected to a system.

In some cases we work directly with the manufacturer and are supplied all the information we need to be able to fully support the device. In other cases it might be distributors, integrators or end uses that need to use our driver on a device that is not supported. We document here the information required to support a controller and also advise how to obtain that information if technical documents are not readily available.

The two most common touch controller types are USB or Serial. Adding controller support is normally undertaken free of charge as we are keen to ensure UPDD supports as many pointer devices as possible. Once a controller is configured in our production system we can generate the full range of drivers for the new controller.

It is important to understand that any technical information supplied to us which is not in the public domain will be treated with the utmost confidentiality and never disclosed. If required we will enter into a joint NDA agreement to formalize this undertaking.

The minimum information needed to support a controller is a follows:

- **Controller name as listed in UPDD**
  We use three components in the naming of the controllers, company name, controller id and interface, e.g. if we created our own USB controller, called the TC001 then the controller would be named Touch-Base, TC001, USB.
- **Interface settings**

USB

Vendor identification (VID) — USB identification unique to each vendor.

Product identification (PID) — Allocated by the vendor and unique to each USB controller within the vendor's product range.

Drivers register vendor and product ids that they support. When a USB device is connected the vendor id and product id is used by the PnP sub-system to select the driver responsible for the device.

Interface and Endpoint usage — USB controllers use interfaces to exchange data between the host system and the device. Each interface carries a number of endpoints (unidirectional pipes) and each endpoint can be of a specific transfer type (Control, Interrupt, Bulk and Isochronous).

The common usage for transmitting touch data has been interface 0, endpoint 1 using interrupt transfers but this is not always the case.

We recommend the use of interrupt transfers.
Bulk transfers should work with UPDD asis but are likely to be unsuitable due to the inherent latency and this is in any case subject to confirmation as this transfer method has never been seen or tested.

Isochronous transfers are not currently supported, but could be if required, but they appear to offer no advantage over interrupt transfers.

The driver will connect to the device via the appropriate interface and endpoint.

An overview of USB basics is available here.

Initialisation — Some controllers need initializing before transmitting touch data or in some cases will output different structured packets depending on the host OS. For multi-touch devices we prefer to initialize the device to output multi-touch data packets irrespective of the OS in use and therefore we need to identify the USB set feature request used to set MT output.

Serial

Communication parameters — Baud rate, parity, data bits, stop bits e.g. 9600,N,8,1

For new controllers under development we have the following recommendation for USB:

1) If HID compatible the controller has the advantage of being used both with UPDD and alternatively with the HID driver supplied in many OS where some touch utilities are also available, such as calibration etc.
2) Interrupt transfer mode as this is the most suitable transfer mode for touch data. Further, by default early UPDD Windows drivers, assumed data was delivered on interface 0, endpoint 1. Starting with UPDD version 5 these are now configured in the controller definition.
3) We would also recommend that the controller be designed so that it is not necessary to send vendor specific

requests to activate it. This means that the controller can be verified easily using 3rd party tools to examine the data stream.

4) The controller must not make assumptions about the order of commands seen from Windows, but instead comply to the USB spec, otherwise it is likely not to work with future Windows versions and very likely not to work with non Windows systems.
5) We recommend that you execute the Winqual tests for pointer devices as these check various aspects of the controller. You will need to have HID, UPDD or another touch driver installed to make these checks. These tests can be undertaken by Touch-Base if requested. See WHQL documentation of more information.
6) It is recommended that each controller should carry a unique serial number in the string descriptor indexed by iSerialNumber. This allows for predicatble association of touch devices with monitors in a multi monitor environment. SString descriptors are explained at http://www.beyondlogic.org/usbnutshell/usb5.htm#StringDescriptors

- **Single touch data packet format**

  When a single pointer device is in use it delivers a data packet to the system that carries co-ordinate information (X and Y and sometimes Z - pressure data) along with other device information, such as lift off trigger indicator. We need to know the EXACT structure of this packet.

- **Multi touch data packet format**

  When a multi pointer device is in use it delivers data packets to the system that carries co-ordinate information (X and Y and sometimes Z - pressure data) along with other device information, such as lift off trigger and stylus information. We need to know the EXACT structure of this packet.

For new controllers under development we have the following comments regarding packet structure:

Generally speaking we are able to configure any given packet structure within our driver as we are able to define the packet structure at individual bit level so normally we expect to be informed of the packet structure implemented in the controller and we then define it as appropriate. However, for readers wanting advice as to a suitable packet structure, here is a example 12 bit packet structure that offers a status byte and 2 bytes per x and Y co-ordinate:

Bit	1	2	3	4	5	6	7	8
Byte 0	0	0	0	0	0	0	0	1
Byte 1	0	0	0	0	0	0	**T**	0
Byte 2	0	X6	X5	X4	X3	X2	X1	X0
Byte 3	0	0	0	X11	X10	X9	X8	X7
Byte 4	0	Y6	Y5	Y4	Y3	Y2	Y1	Y0
Byte 5	0	0	0	Y11	Y10	Y9	Y8	Y7

Where **T** =1 when touching and 0 when untouch (pen up) is detected.

This packet offers unique sync pattern (1st bit of each byte) such that lost bytes (say during transmission) will allow the driver to discard incomplete packets and resync on the 1st byte of the next packet (bit 8 = 1)

Notes:
HID may dictate packet structure format.
Ensure sync bits are defined, especially in serial devices.
Use unsigned integers for co-ordinate data (avoid FFFF = -1!).
Report (to Touch-Base) the exact numbers of co-ord bits used in the packet as UPDD definition relies on the correct definition to aid 'auto-calibration'. E.g. 8 bits = 256 co-ord range, 9 bits = 512, 10 bits = 1024, 11 bits = 2048, 12bits = 4096 etc.
For multi-touch some of the other status bits can be used to indicate stylus id, e.g. bits 7 and 8 :- 00 = stylus 1, 01 = stylus 2, 10 = stylus 3 and 11 = stylus 4.

A more comprehensive overview of touch data structures relating to HID and UPDD is discussed here.

Generally the above information is all we need to add support for a controller. However, there is some additional information that we made need to complete support or that allows us to implement support beyond mouse emulation as discussed below:

- **Firmware commands**

  Some controllers have a command set to adjust internal controller settings or to define settings that define a controller's functionality, which could be needed to work with our definition of the controller (e.g. some controllers can generate different data packet formats and UPDD would need to send the code to set the packet structure expected by UPDD)

  The UPDD Console program could also expose firmware settings to the end user so that they can be adjusted as required. The console could, for example, also show the version of the firmware if it is available along with any other information considered relevant.

  For USB devices we also need to know the end point used to send the firmware commands and the end point used to return the data.

- **EEPROM storage**

  UPDD normally stores calibration data locally in the computer but there can be times when it is advantageous to store calibration data in the controller and if a controller supports EEPROM or other forms of storage we need technical details to implement this functionality.

Information sources

Official Documentation

In most cases we would hope we could be supplied the official technical information relating to the information above and controller against which to test the newly configured controller. However, we acknowledge this is not always possible.

Failing that a technical contact within the company able to assist with our queries is very useful.

If neither of these is available then, if you have the controller to hand, here are ways to determine some of the required information:

Information gathering

Where official technical information is not readily available here are some approaches to identifying the required information if you have the controller to hand.

Serial

Given that the minimum information required is the serial interface parameters and the data packet format there are two approaches we can suggest, serial port data scope or UPDD capture tool.

Data scope

We have written a Windows serial port data scope that can be used to connect to a serial port and display the serial data. The data can then be saved to a file for analysis. We request that the data from a touch in each corner be saved in separate files and sent to us for further analysis.

The data scope program is available in the utilities section of our support page along with full documentation.

UPDD capture tool

We can configure a UPDD serial driver that can be installed and used to capture touch data packet information.

USB

The minimum information required is the Vendor and product identification, Interface and endpoint usage and the data packet format we document ways to identify this information:

Vendor and Product id

There are numerous ways to identify a USB device's vendor and product id as documented in our 'Identifying touchscreen controllers' document.

USB analysis tools

Various software USB data analysis tools are available as documented here. These tools show traffic between the host and USB device and can be used to identify the Interface and endpoint usage and see the touch data packet.

UPDD capture tool

We can configure a UPDD USB driver that will interface to a defined 'vendor and product id' that can be installed and used to capture touch data packet information.

Additional data processing

Some controllers output touch data packets that need additional processing in the host system (e.g. to extract the stylus touch data - x/y, stylus, pen status), specially for controllers that output raw sensor in bulk data format. We have added support for a number of these devices and our preferred implementation of such a controller is for the touch manufacturer to supply a library file to which UPDD can deliver the bulk data and receive back touch data packets for onward processing. If necessary we can write the pre-processing code given the correct level of technical details regarding the bulk data and the algorithms needed to extract the touch data. Preprocessing of the touch data is discussed in more detail here.

Contact

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