

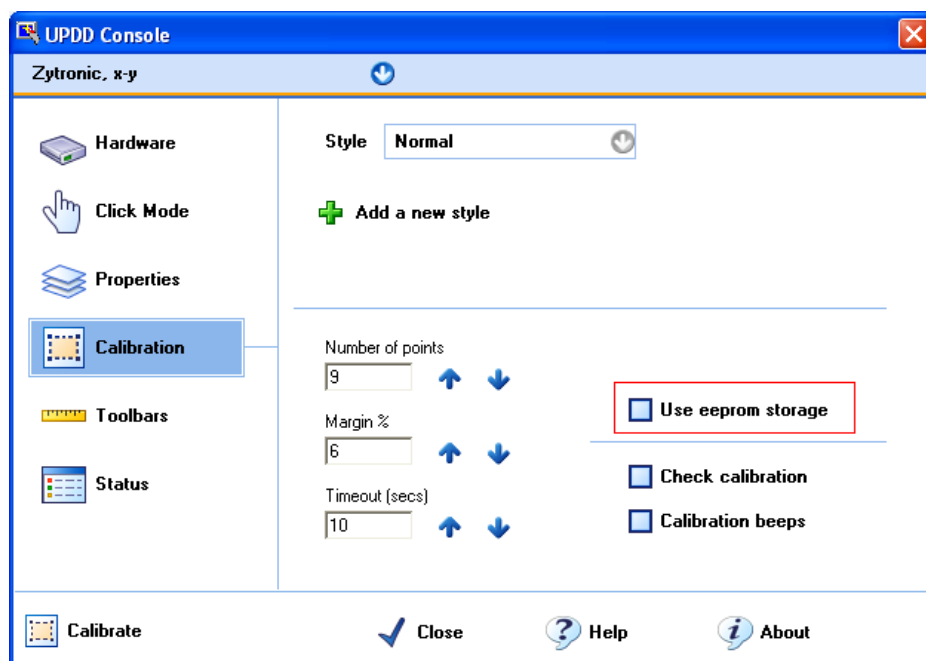
Some pointer device controllers have onboard EEPROM (Electrically Erasable Programmable Read Only Memory). EEPROM can be used to store information pertinent to the controller and also offers a memory storage area for applications and drivers to store data. One of the main uses for this memory with UPDD is to store the calibration data within the controller rather than locally on a system.

In all cases an absolute pointer device needs to be calibrated with the systems video display such that the point of contact is aligned with the video image. [UPDD calibration](#) is performed by drawing points on the video display at known locations that are then touched and the touch co-ordinates are noted for the given point of display. The driver can then use this data to scale and map the touch input to the video location. This information, known as the calibration data, needs to be stored for future reference, such that the device is calibrated at each system power up. This information can be stored locally on the system (e.g. the system's persistent memory, within a file, etc) or externally in the controller's EEPROM.

Given that UPDD supports 100's of different pointer devices, all with different characteristics, our driver generally stores its calibration data locally within the system.

However, there are cases where it is desirable to store calibration data externally. This is especially true of systems where the calibration data cannot be stored locally, such as some embedded system configurations where the entire system is locked down and any changes made are lost when the device is powered off. In other cases it is desirable to calibrate a touch screen prior to usage on a similar system so that it is calibrated when first used and will only require recalibrating if the EEPROM calibration is inaccurate.

As and when we have been requested to support EEPROM calibration for a given controller it has been added. When one of these controllers is defined in the UPDD build (and with [EEProm enabled](#)) and added in the UPDD Console as an active device an EEPROM check box is shown on the UPDD Console, Calibration screen.



When this EEPROM check box is enabled (or set internally in the UPDD settings), it indicates that;

- 1) EEPROM calibration is utilised at the NEXT calibration. ***It is important to note that this setting enables the EEPROM calibration procedure such that at the point of calibration the EEPROM calibration procedure is invoked.***
- 2) Since UPDD [4.1.8, build id 1738](#) the [calibration EEPROM retrieval function](#) is invoked to read calibration data in cases where UPDD native calibration data is stored on the controller.

## EEPROM implementations

EEPROM calibration falls into two distinct categories, one whereby UPDD stores its own native calibration data in EEPROM and one whereby the controller's own firmware calibration procedure is invoked, more commonly known as a firmware calibration.

EEProm is implemented in UPDD by the use of specific eeprom protocol code that is written for each controller. When a controller is defined in our controller production system we indicate the eeprom protocol to use to enable eeprom calibration.

The protocol also indicates if eeprom calibration utilises the controllers own firmware calibration capabilities or stores UPDD

calibration data for later retrieval, as described below

| Method          | Description   |
|-----------------|---|
| <b>Firmware</b> | UPDD calibration invokes the controller's firmware calibration commands to perform calibration such that scaled touch co-ordinates are delivered when the device is in use. |

In most cases, the calibration function implemented within the controller's firmware dictates the number and location of the calibration points and this will be reflected within the available settings on the calibration dialog.

Following calibration the controller's output is adjusted to map to the video display.

Where firmware calibration is used the controller internally rescales the co-ordinate output to be calibrated with the video system and thus generated co-ordinate data is "pre-calibrated" and therefore a retrieval process is not relevant.

|                    |   |
|--------------------|---|
| <b>UPDD Native</b> | UPDD calibration procedure stores its own calibration data in the controller's EEPROM. To utilize the calibration data stored in the controller it must be retrieved using the <a href="#">calibration EEPROM retrieval function</a> , typically in the following situations: |
|--------------------|---|

- **System startup**, In locked down environments where updd settings are not held in a persistent state, such as embedded systems, it is important to retrieve the calibration data from the controller as any data stored locally is likely to be lost over a reboot, reverting back to the data built into the embedded image.

Since [4.1.8, build id 1738](#) eeprom calibration is automatically retrieved in a Windows desktop environment at system startup for each controller that has the EEPROM setting enabled

- **Device hot plug** (since [4.1.8, build id 1738](#)) eeprom calibration is automatically retrieved in a Windows desktop environment when a USB device is plugged in and when a new serial device is configured and the EEPROM setting is enabled.
- **Manually requested** (since [4.1.8, build id 1738](#)) in the UPDD Console, Calibration dialog enabling the 'Use eeprom storage' option will invoke a read of the calibration data.

If eeprom data is not retrieved then locally stored calibration data is used. TBcalib returns an [error code](#) if the calibration data checksum is incorrect (e.g. retrieving calibration data from a controller that has no calibration data such as a new controller used for the first time).

#### Storage requirements

With UPDD 4.1.10, version 2240 and above, we have rewritten the UPDD EEPROM framework to make it easier to implement across all OS platforms. With this new implementation of eeprom calibration it requires the following amount of storage.

number of points x 4 + 22 bytes

So

4 points needs 38 bytes  
9 points needs 54 bytes  
25 points needs 122 bytes

16 bytes is to hold the [calibration style](#), which must be a maximum of 15 bytes (15 + endofstring)

Since moving to this new framework (which caters for old eeprom implementation) we have made many internal code changes that relate to eeprom support but are not in a position to test all eeprom capable controllers. It is for this reason that we request that you contact us as soon as you have a eeprom storage requirement and we will advise at that time.

### Enabling EEPROM protocols

Given that the EEPROM interface to each controller is unique there is separate protocol code written for each controller. When we define a controller we identify the protocol to use. If the EEPROM protocol setting is blank then the EEPROM option will not be shown

The protocol setting protocol identifiers are shown below. UPDD version numbers indicate when support was added. Those listed in **Red** indicate additional development work is required to support in the OS and those listed in **Gray** are untested but in theory should work.

| Controller                | Port type | Protocol id | Description                                | OS and UPDD version |     |       |       |
|---------------------------|-----------|-------------|--|---------------------|-----|-------|-------|
|                           |           |             |  | Win                 | CE  | Mac   | Linux |
| <b>Firmware interface</b> |           |             |  |                     |     |       |       |
| DMC Fit 10                | Serial    | dmc-serial  | Firmware calibration for scaled touch data | 4.1.0               | TBA | 4.1.8 | 4.1.8 |
|                           | USB       | dmc-usb     | Firmware calibration for scaled touch data | 4.1.0               | TBA | 4.1.8 | 4.1.8 |

*When enabled the calibration procedure is restricted to the range of calibration points supported by the controller's firmware calibration procedure.*

|                            |   |                 |   |               |        |        |
|----------------------------|---|-----------------|---|---------------|--------|--------|
| 3M SCnnn                   | Serial / USB  | n/a             | Hard coded - Firmware calibration for scaled touch data | TBA           | TBA    |        |
|                            | <i>UPDD invokes the controller's 3 point calibration procedure. In some older versions of the driver it is important to manually set the number of calibration points to 3 at the same time as enabling EEPROM.</i>   |                 |   |               |        |        |
|                            | <i>This procedure is also required to initialise / linearise the controller and should be performed at least once irrespective of the need to subsequently utilise EEPROM functionality.</i>  |                 |   |               |        |        |
|                            | <i>Customers have reported that in some systems with video resolution 800 x 600 this procedure has failed but has worked fine with a 1024 x 768 resolution! Yet to be fully understood!</i>   |                 |   |               |        |        |
| ELO Smartset               | USB   | smartset-usb    | Firmware calibration for scaled touch data              | 4.1.6         | TBA    | 4.1.1  |
|                            | <i>UPDD invokes the controller's 3 point calibration procedure. Internally it actually uses a 2 point algorithm with the 3<sup>rd</sup> point to determine orientation.</i>   |                 |   |               |        |        |
| Nihon Kaiheiki             | USB   | nikkai-usb      | Firmware calibration for scaled touch data              | 4.1.8         | TBA    | 4.1.8  |
|                            | <i>When enabled the calibration procedure is restricted to the range of calibration points supported by the controller's firmware calibration procedure, being 2, 4, 5 and 9.</i>   |                 |   |               |        |        |
| <b>Software interface</b>  |   |                 |   |               |        |        |
| Zytronic x-y               | Serial  | zytronic-serial | EEProm storage or UPDD data                             | 4.1.1         | 4.1.10 | 4.1.10 |
|                            | USB   | zytronic-usb    | EEProm storage or UPDD data                             | 4.1.1         | 4.1.10 | 4.1.10 |
| Zytronic ZXY 100           | Serial  | zxy100-serial   | EEProm storage or UPDD data                             | 4.1.8<br>1810 | 4.1.10 | 4.1.10 |
|                            | USB   | Zxy100-usb      | EEProm storage or UPDD data                             | 4.1.8         | 4.1.10 | 4.1.10 |
| Hampshire/Microchip Tsharc | USB   | tsharc-usb      | EEProm storage of UPDD data                             | 4.1.1         | 4.1.10 | 4.1.8  |
|                            | <i>Only works with EEPROM capable controllers. Further, due to current firmware limitations the data is written to the controller, 1 byte at a time, and depending on the number of calibration points can take a significant time to store and retrieve. 4 points takes approx 20 seconds.</i> |                 |   |               |        |        |
| Hampshire/Microchip AR1100 | USB   | mc-ar1100       | EEProm storage of UPDD data Under development           | 4.1.10        | 4.1.10 | 4.1.10 |
| Data Modul                 | USB   | EETI            | EEProm storage of UPDD data                             | 4.1.8<br>1800 | 4.1.10 | 4.1.8  |
| TRS Star                   | USB   | TRS             | EEProm storage of UPDD data                             | 4.1.10        | 4.1.10 | 4.1.10 |

For UPDD 4.1.x the EEPROM protocol id settings are held in the UPDD settings file, TBUPDD.INI, within the branch [updd\parameters\n] (for configured/active controllers) and [updd\parameters\controller\ts00n] the default settings for controllers supported by the driver:

```
eeeprom calibration=0x1
eeeprom protocol='protocol id'
```

Any other EEPROM settings within the file are redundant in 4.1.x and above.

For earlier UPDD versions the EEPROM interface is hard coded and does not utilise the protocol id setting.

For CE see [EEPROM notes](#) in the CE documentation. At the time of writing the 4.1.10 CE driver only supports EEPROM in controllers whereby UPDD calibration is being stored and not firmware based calibration controllers. This support will be added as required.

If you are using a driver which supports one of the above controllers but does not show the EEPROM option then it is likely the EEPROM setting was/is not defined in the definition of the driver at build time. If this is the case contact Touch-Base to request an updated build.

### Adding EEPROM support

We can add support in UPDD for any EEPROM enabled controller as required once supplied the technical documentation which describes the controller's EEPROM calibration interface.

If you are a controller manufacturer and would like to add EEPROM capabilities to you controller, we are aware of three methods that have been utilised as described below:

#### Raw Mode

Firmware commands are used to specify that raw blocks of data can be stored and retrieved from EEPROM. In this mode, UPDD simply stores and retrieves the calibration from the controllers EEPROM.

This is the simplest method to implement for both the firmware developers and our driver as the EEPROM read and write functions are already built into the UPDD calibration program.

#### Formatted mode

The controller's firmware dictates the format of the calibration data passed to the controller which is further processed by the controller. The co-ordinate data is then scaled and mapped to a grid co-ordinate range, based on the formatted data.

In this case the calibration program has to gather calibration data as dictated by the data format requirement.

**Co-operative mode**

The controller's firmware exposes calibration commands such that the calibration program issues a firmware command for each calibration point displayed and the firmware captures the calibration points in real-time. The number of calibration points and the pattern used is dictated by the firmware's requirements. The co-ordinate data is then scaled and mapped to a grid co-ordinate range, based on the capture calibration information.

**Contact**

For further information or technical assistance please email the technical support team at [technical@touch-base.com](mailto:technical@touch-base.com)