

PenMount Application Development Guide for Microsoft Windows XP / Windows 7

Version 3.1

25/Feb/'19



Preface

Disclaimer

The information in this document is subject to change without notice. The manufacturer makes no representations or warranties regarding the contents of this manual and specifically disclaims any implied warranties of merchantability or fitness for any particular purpose. Furthermore, the manufacturer reserves the right to revise this publication or make changes in the specifications of the product described within it at any time without notice and without obligation to notify any person of such revision.

Trademarks

PenMount is a registered trademark of **SALT International Corp.** Microsoft and Windows are registered trademarks of Microsoft Corp. Other product names used in this manual are the properties of their respective owners and are acknowledged.

Copyright

This publication, including all photographs, illustrations and software, is protected under international copyright laws, with all rights reserved. Neither this manual, nor any of the material contained herein, may be reproduced without the express written consent of the manufacturer.

©Copyright 2019 **SALT International Corp.**

Revision Table

| Date | Revision | Changes |
|--------------|----------|---|
| 08/May/2009 | 1.0 | Initial Release |
| 20/July/2009 | 2.0 | <p>This revision deals with the refinement of previous version. Chapter is reorganized and the following sections are added:</p> <p><u>2.1. General Function API</u></p> <p><u>2.1.1. Start or Stop Touch Notification</u></p> <p><u>2.1.2. Check If the Last Mouse Event Is Generated from PenMount Device</u></p> <p><u>2.3. Multi-Monitor API</u></p> <p><u>2.3.1. Set Total Monitor Number</u></p> <p><u>2.3.2. Set Last-Touched Mapped Monitor Rectangle</u></p> <p><u>2.4. Right-Mouse-Button API</u></p> <p><u>2.4.1. Emulate Right Mouse Button Click on Next Touch</u></p> |
| 09/Nov/2009 | 2.1 | <ul style="list-style-type: none"> ■ Deletion: Previous section 2.1.1. Start or Stop Touch Notification is deleted. |
| 09/Feb/2010 | 2.2 | The new <u>Appendix B: PenMount Calibration Setting Guide</u> is inserted. The original appendix B is shifted to <u>Appendix C: Multi-Touch Application Guide</u> . |
| 24/Mar/2010 | 2.3 | Controller and device driver requirements are added to all PenMount API. |
| 21/Apr/2010 | 2.4 | <u>Appendix C: PenMount Control Panel Command Line Options</u> is inserted. |
| 18/Jun/2010 | 2.5 | <p>Sections <u>8 Multiple Monitor API</u></p> <p><u>This chapter will</u> guide you through changing the PenMount device driver multiple monitor settings.</p> <p>Enable/Disable Monitor Mapping and <u>8.2 Set Mapped Monitor Rectangle</u> are added. Table of the available options of DMCCtrl.exe are revised under <u>Appendix C: PenMount Control Panel Command Line Options</u>.</p> |
| 08/Jul/2010 | 2.6 | <p>In section 5.23 Enable/ Disable Monitor Mapping, function name is revised to "pmloctl_SetMultiMonitorCount"</p> <p>To Appendix C: PenMount Control Panel Command Line Options, options "calibmmon" is added.</p> |

| | | |
|--------------------|-----|---|
| 10/Jan/2013 | 3.0 | Revised contents for PenMount Windows Universal Driver V2.2.0.309. The APIs are categorized the same as in the pmloctlAPI.h file. |
| 25/Feb/2019 | 3.1 | 9.4, 9.5: Lists of return values and panel type definitions. 9.6, 9.7: Lists of return values. |

Table of Content

| | |
|--|-----------|
| PREFACE | I |
| Disclaimer | i |
| Trademarks | i |
| Copyright | i |
| REVISION TABLE | II |
| 1. INTRODUCTION..... | 7 |
| 2. DEVICE API | 8 |
| 2.1. GET DEVICE HANDLE | 8 |
| 2.2. GET DEVICE ID | 9 |
| 2.3. GET DEVICE MODEL | 10 |
| 2.4. GET DEVICE FIRMWARE VERSION | 11 |
| 2.5. GET DEVICE DRIVER VERSION | 12 |
| 2.6. ENABLE/DISABLE CONTROLLER'S FLASH STORAGE..... | 13 |
| 2.7. SAVE DEVICE DRIVER SETTINGS TO CONTROLLER'S FLASH STORAGE 14 | |
| 3. TOUCH API..... | 16 |
| 3.1. GET LAST TOUCH POSITION | 16 |
| 3.2. CHANGE REPORT MODE..... | 17 |
| 3.3. GET CURRENT REPORT MODE | 18 |
| 3.4. REGISTER TOUCH INPUT CALLBACK..... | 19 |
| 3.5. UN-REGISTER TOUCH INPUT CALLBACK | 20 |
| 3.6. SET TOUCH OPERATION MODE | 21 |
| 3.7. GET CURRENT TOUCH OPERATION MODE | 22 |
| 3.8. GET TOUCH RESOLUTION | 23 |
| 4. BUTTON API..... | 25 |

| | | |
|------|--|----|
| 4.1. | SET MOUSE BUTTON TYPE..... | 25 |
| 4.2. | ENABLE/DISABLE RIGHT MOUSE BUTTON EMULATION | 26 |
| 4.3. | CHECK IF RIGHT MOUSE BUTTON EMULATION IS ENABLED | 27 |
| 4.4. | SET INTERVAL OF RIGHT MOUSE BUTTON EMULATION | 28 |
| 4.5. | GET CURRENT INTERVAL OF RIGHT MOUSE BUTTON EMULATION | 29 |
| 4.6. | SET OFFSET ALLOWANCE OF RIGHT MOUSE BUTTON EMULATION | 30 |
| 4.7. | GET CURRENT OFFSET OF RIGHT MOUSE BUTTON EMULATION..... | 31 |
| 5. | FILTER API | 33 |
| 5.1. | SET FILTER VALUE TO AVOID JITTER | 33 |
| 5.2. | GET CURRENT FILTER VALUE TO AVOID JITTER | 34 |
| 6. | CALIBRATION API..... | 35 |
| 6.1. | CHANGE CALIBRATION PARAMETERS..... | 35 |
| 6.2. | GET CURRENT CALIBRATION PARAMETERS | 36 |
| 6.3. | CHANGE ROTATION DEGREE | 37 |
| 6.4. | GET CURRENT ROTATION SETTING | 38 |
| 6.5. | SET SCREEN EDGE COMPENSATION PARAMETER..... | 39 |
| 6.6. | GET SCREEN EDGE COMPENSATION PARAMETERS..... | 40 |
| 7. | BEEP API..... | 42 |
| 7.1. | SET TOUCH BEEP MODE..... | 42 |
| 7.2. | GET CURRENT TOUCH BEEP MODE..... | 43 |
| 7.3. | SET TOUCH BEEP FREQUENCY | 44 |
| 7.4. | GET CURRENT TOUCH BEEP FREQUENCY | 45 |
| 7.5. | SET TOUCH BEEP DURATION..... | 46 |
| 7.6. | GET CURRENT TOUCH BEEP DURATION..... | 47 |

| | | |
|------|--|----|
| 8. | MULTIPLE MONITOR API..... | 48 |
| 8.1. | ENABLE/DISABLE MONITOR MAPPING | 48 |
| 8.2. | SET MAPPED MONITOR RECTANGLE | 49 |
| 9. | NCR API..... | 52 |
| 9.1. | SET MONITOR ID TO CONTROLLER | 52 |
| 9.2. | GET MONITOR ID FROM CONTROLLER | 53 |
| 9.3. | UPDATE MONITOR MAPPING..... | 54 |
| 9.4. | SET PANEL TYPE TO CONTROLLER..... | 54 |
| 9.5. | GET PANEL TYPE FROM CONTROLLER | 56 |
| 9.6. | SET DEVICE SERIAL NUMBER TO CONTROLLER | 57 |
| 9.7. | GET DEVICE SERIAL NUMBER FROM CONTROLLER | 58 |
| | APPENDIX A: CALIBRATION APPLICATION GUIDE | 60 |
| | APPENDIX B: PENMOUNT CALIBRATION SETTING GUIDE..... | 62 |
| | APPENDIX C: PENMOUNT CONTROL PANEL COMMAND LINE OPTIONS..... | 66 |

1. Introduction

This document describes the API provided by the PenMount SDK, and also include those that are specially developed for NCR COP Project.

The files included in the PenMount SDK are listed below.

| Item | Name | Directory |
|-----------------------|----------------|----------------|
| header | pmIoctlAPI.h | \Include |
| 32 bit library | pmIoctlAPI.lib | \Library\i386 |
| 64 bit library | pmIoctlAPI.lib | \Library\amd64 |
| 32 bit runtime | pmIoctlAPI.dll | \Library\i386 |
| 64 bit runtime | pmIoctlAPI.dll | \Library\amd64 |
| Sample code | TouchCallback | \Sample |

2. Device API

This chapter will guide you to the Ioctl API that is related to the PenMount device.

2.1. Get Device Handle

Syntax

```
LONG    pmIoctl_GetDeviceHandle (    LONG    type ,  
                                     LONG    ID ,  
                                     HANDLE * hDevice )
```

Description

This function gets the handle that points to the device according to the device type and ID.

Return Values

If function succeeds, the return value is nonzero.

If function fails, the return value is zero.

Parameters

1. Type

[in] The type of PenMount device. For PenMount 6000 USB, please use the following value:

```
#define PMDEVICE_USB6K 0x04
```

2. ID

[in] The ID of the device, starts from 0.

3. hDevice

[out] A pointer to the HANDLE of device.

This chapter will guide you through the functions that retrieves the information of PenMount device.

Requirements

| Item | Description |
|----------------------|--|
| Device Driver | PenMount Windows Universal Driver V2.2.0.285 |
| | PenMount Windows Universal Driver V2.2.0.309 |
| Include | pmloctlAPI.h |
| Library | pmloctlAPI.dll |

2.2. Get Device ID

Syntax

```
LONG    pmloctl_GetDeviceID (          HANDLE *    hDevice ,  
                                LONG *      ID )
```

Description

This function gets the device ID of the device handle.

Return Values

If function succeeds, the return value is nonzero.

If function fails, the return value is zero.

Parameters

1. hDevice
[in] The HANDLE that points to the device returned by pmloctl_GetDeviceHandle().
2. ID
[in] The ID of the device, starts from 0. This ID should be the same as the value passed to pmloctl_GetDeviceHandle().

Requirements

| Item | Description |
|------|-------------|
|------|-------------|

| | |
|----------------------|--|
| Device Driver | PenMount Windows Universal Driver V2.2.0.285 |
| | PenMount Windows Universal Driver V2.2.0.309 |
| Include | pmloctlAPI.h |
| Library | pmloctlAPI.dll |

2.3. Get Device Model

Syntax

```
LONG    pmloctl_GetDeviceModel (    HANDLE    hDevice ,
                                     LONG *      model )
```

Description

This function acquires the product model of PenMount touch controller.

The product number can be combined with the firmware version described in the next section to identify a PenMount device.

Return Values

If function succeeds, the return value is nonzero.

If function fails, the return value is zero.

Parameters

3. hDevice
[in] The HANDLE that points to the device returned by pmloctl_GetDeviceHandle().
4. model
[out] The PenMount touch controller firmware version.

| Value | Product |
|-------------|--------------------------|
| 6000 | PenMount 6000 Controller |

Requirements

| Item | Description |
|----------------------|--|
| Device Driver | PenMount Windows Universal Driver V2.2.0.285 |
| | PenMount Windows Universal Driver V2.2.0.309 |
| Include | pmloctlAPI.h |
| Library | pmloctlAPI.dll |

2.4. Get Device Firmware Version

Syntax

```
LONG    pmloctl_GetDeviceVersion (    HANDLE    hDevice,
                                      LONG *    version )
```

Description

This function acquires the firmware version of PenMount touch controller.

Please refer to [Device Driver Development Manuals](#) of PenMount touch controllers for the complete firmware format.

Return Values

If function succeeds, the return value is nonzero.

If function fails, the return value is zero.

Parameters

1. hDevice
[in] The HANDLE that points to the device returned by pmloctl_GetDeviceHandle().
2. version
[out] The PenMount touch controller firmware version.
This 32 bit value can be split into four bytes. For PenMount 6000.6.0.0, the version values are listed as below:

| Item | Description |
|------|-------------|
|------|-------------|

| | |
|----------------|-----------------------|
| Value | 393216 (0x00060000) |
| Byte[3] | 0 |
| Byte[2] | 6 |
| Byte[1] | 0 |
| Byte[0] | 0 |

Requirements

| Item | Description |
|----------------------|--|
| Device Driver | PenMount Windows Universal Driver V2.2.0.285 |
| | PenMount Windows Universal Driver V2.2.0.309 |
| Include | pmloctlAPI.h |
| Library | pmloctlAPI.dll |

2.5. Get Device Driver Version

Syntax

| | | | |
|------|----------------------------|--------|---------------|
| LONG | pmloctl_GetDriverVersion (| HANDLE | hDevice, |
| | | LONG * | MajorVersion, |
| | | LONG * | MinorVersion, |
| | | LONG * | BuildNumber) |

Description

This function gets the version of PenMount Windows driver.

The PenMount device driver version consists of three parts: major, minor and build.

Please refer to PenMount device driver release notes for the updates of each version.

Return Values

If function succeeds, the return value is nonzero.

If function fails, the return value is zero.

Parameters

1. **hDevice**
[in] The HANDLE that points to the device returned by `pmloctl_GetDeviceHandle()`.
2. **MajorVersion**
[out] The major version number of PenMount device driver. For PenMount Windows Universal Driver V2.2.0.285, this value is 2.
3. **MinorVersion**
[out] The minor version number of PenMount device driver. For PenMount Windows Universal Driver V2.2.0.285, this value is 3.
4. **BuildNumber**
[out] The build number of PenMount device driver. For PenMount Windows Universal Driver V2.2.0.285, this value is 6.

Requirements

| Item | Description |
|----------------------|--|
| Device Driver | PenMount Windows Universal Driver V2.2.0.285 |
| | PenMount Windows Universal Driver V2.2.0.309 |
| Include | pmloctlAPI.h |
| Library | pmloctlAPI.dll |

2.6.Enable/Disable Controller's Flash Storage

Syntax

```
LONG    pmloctl_EnableDeviceStorage (    HANDLE    hDevice,  
                                         LONG        state )
```

Description

This function enables or disables driver's using flash storage on controller.

When the state is set to TRUE, the driver will store the calibration data in flash storage after calibration.

Return Values

If function succeeds, the return value is nonzero.

If function fails, the return value is zero.

Parameters

1. `hDevice`
[in] The HANDLE that points to the device returned by `pmloctl_GetDeviceHandle()`.
2. `state`
[out] Sets the state to TRUE to enable flash storage, FALSE otherwise.

Requirements

| Item | Description |
|----------------------|--|
| Device Driver | PenMount Windows Universal Driver V2.2.0.285 |
| | PenMount Windows Universal Driver V2.2.0.309 |
| Include | pmloctlAPI.h |
| Library | pmloctlAPI.dll |

2.7. Save Device Driver Settings to Controller's Flash Storage

Syntax

```
LONG    pmloctl_SaveDeviceSetting (    HANDLE    hDevice )
```

Description

This function enables PenMount device driver to save calibration settings back to flash storage on control board.

Return Values

If function succeeds, the return value is nonzero.

If function fails, the return value is zero.

Parameters

1. hDevice

[in] The HANDLE that points to the device returned by pmloctl_GetDeviceHandle().

Requirements

| Item | Description |
|----------------------|--|
| Device Driver | PenMount Windows Universal Driver V2.2.0.285 |
| | PenMount Windows Universal Driver V2.2.0.309 |
| Include | pmloctlAPI.h |
| Library | pmloctlAPI.dll |

3. Touch API

This chapter will give descriptions of the touch relative APIs.

3.1. Get Last Touch Position

Syntax

| | | | |
|------|----------------------------|---------|-------------|
| LONG | pmIoctl_GetTouchPosition (| HANDLE | hDevice, |
| | | LONG | bCaibrated, |
| | | POINT * | point) |

Description

This function gets the last calibrated or raw touch position.

Return Values

If function succeeds, the return value is nonzero.

If function fails, the return value is zero.

Parameters

1. hDevice
[in] The HANDLE that points to the device returned by pmIoctl_GetDeviceHandle().
2. bCaibrated
[in] If this value is set to TRUE, the function will return the last calibrated touch position.
If this value is set to FALSE, the function will return the last touch position without calibration.
3. point
[out] The Return Values of the calibrated position data.
The value is scaled to fit the screen resolution in use.

Requirements

| Item | Description |
|------|-------------|
|------|-------------|

| | |
|----------------------|--|
| Device Driver | PenMount Windows Universal Driver V2.2.0.285 |
| | PenMount Windows Universal Driver V2.2.0.309 |
| Include | pmloctlAPI.h |
| Library | pmloctlAPI.dll |

3.2.Change Report Mode

Syntax

```
LONG    pmloctl_EnableTouchReport (    HANDLE    hDevice,
                                       LONG    mode )
```

Description

This function changes the report mode of PenMount device driver.

Return Values

If function succeeds, the return value is nonzero.

If function fails, the return value is zero.

Parameters

1. hDevice
[in] The HANDLE that points to the device returned by pmloctl_GetDeviceHandle().
2. mode
[in] The new state that driver reports touch events.
Passing FALSE will set the PenMount device driver ignore any touch event.
Passing TRUE will enable the device to send input report again.

Requirements

| Item | Description |
|----------------------|--|
| Device Driver | PenMount Windows Universal Driver V2.2.0.285 |
| | PenMount Windows Universal Driver V2.2.0.309 |

| | |
|----------------|----------------|
| Include | pmloctlAPI.h |
| Library | pmloctlAPI.dll |

3.3. Get Current Report Mode

Syntax

```
LONG    pmloctl_CheckTouchReport (    HANDLE    hDevice,
                                      LONG *    mode )
```

Description

This function gets the last state of the driver reporting input event.

Return Values

If function succeeds, the return value is nonzero.

If function fails, the return value is zero.

Parameters

1. hDevice
[in] The HANDLE that points to the device returned by pmloctl_GetDeviceHandle().
2. mode
[out] A pointer to a LONG type data.
If returned value is TRUE, the device driver will report touch inputs;
if returned value is FALSE, the device driver won't send any touch input.

Requirements

| Item | Description |
|----------------------|--|
| Device Driver | PenMount Windows Universal Driver V2.2.0.285 |
| | PenMount Windows Universal Driver V2.2.0.309 |
| Include | pmloctlAPI.h |
| Library | pmloctlAPI.dll |

3.4.Register Touch Input Callback

Syntax

```
LONG    pmIoctl_RegTouchCallback (    HANDLE        hDevice,
                                       FTOUCHCALLBACK * fCallback )
```

Description

This function registers the callback function that will be called when new PenMount touch input is reported. The caller must also call pmIoctl_UnRegTouchCallback() when terminating the program.

Return Values

If function succeeds, the return value is nonzero.

If function fails, the return value is zero.

Parameters

1. hDevice

[in] The HANDLE that points to the device returned by pmIoctl_GetDeviceHandle().

2. fCallback

[in] Specify the callback function that will be called when touch input. The prototype of the callback function is described below:

```
void    ( __cdecl *FTOUCHCALLBACK ) (    int            number,
                                         unsigned int    X,
                                         unsigned int    Y,
                                         unsigned int    buttons,
                                         int              bEnable )
```

| Parameter | Description |
|---------------|---|
| number | The ID of PenMount touch controller. |
| X | The calibrated touch coordinate of X axis. This value is ranged from 0 to 4095. |

| | |
|----------------|---|
| Y | The calibrated touch coordinate of Y axis. This value is ranged from 0 to 4095. |
| buttons | The touch buttons. 1 indicates left button, 0 indicates no button. This value is set according to the touch operation mode used. |
| bEnable | TRUE indicates the device driver is sending touch input report. FALSE indicates the device driver stops sending touch input report. |

Requirements

| Item | Description |
|----------------------|--|
| Device Driver | PenMount Windows Universal Driver V2.2.0.285 |
| | PenMount Windows Universal Driver V2.2.0.309 |
| Include | pmloctlAPI.h |
| Library | pmloctlAPI.dll |

3.5. Un-Register Touch Input Callback

Syntax

```
LONG pmloctl_UnRegTouchCallback ( HANDLE hDevice )
```

Description

This function unregisters the callback previously registered by pmloctl_RegTouchCallback().

Return Values

If function succeeds, the return value is nonzero.

If function fails, the return value is zero.

Parameters

1. hDevice

[in] The HANDLE that points to the device returned by pmloctl_GetDeviceHandle().

Requirements

| Item | Description |
|----------------------|--|
| Device Driver | PenMount Windows Universal Driver V2.2.0.285 |
| | PenMount Windows Universal Driver V2.2.0.309 |
| Include | pmloctlAPI.h |
| Library | pmloctlAPI.dll |

3.6.Set Touch Operation Mode

Syntax

| | | | |
|------|------------------------|--------|----------|
| LONG | pmloctl_SetTouchMode (| HANDLE | hDevice, |
| | | LONG | mode) |

Description

This function sets touch operation mode to mouse emulation or button emulation.

Return Values

If function succeeds, the return value is nonzero.

If function fails, the return value is zero.

Parameters

1. hDevice

[in] The HANDLE that points to the device returned by pmloctl_GetDeviceHandle().

2. mode

[in] The new operating mode for PenMount device driver. Available modes are listed in the table below.

| Mode | Description |
|------|-------------|
|------|-------------|

| | |
|---------------------------|--|
| PMTOUCHMODE_TOUCH | <p>Device driver won't report mouse down when touch in an area ;</p> <p>Device driver will send mouse down when move out the area ;</p> <p>Device driver will send mouse down and mouse up when release without moving out the area.</p> |
| PMTOUCHMODE_BUTTON | Device driver will send mouse down and mouse up when first touch. |
| PMTOUCHMODE_MOUSE | Device driver will send mouse down when touch and mouse up when release. |
| PMTOUCHMODE_HOVER | <p>Device driver won't send mouse down on touch.</p> <p>Device driver will send mouse down and mouse up when release.</p> |

Requirements

| Item | Description |
|----------------------|---|
| Device Driver | <p>PenMount Windows Universal Driver V2.2.0.285</p> <p>PenMount Windows Universal Driver V2.2.0.309</p> |
| Include | pmloctlAPI.h |
| Library | pmloctlAPI.dll |

3.7. Get Current Touch Operation Mode

Syntax

| | | | |
|------|------------------------|--------|----------|
| LONG | pmloctl_GetTouchMode (| HANDLE | hDevice, |
| | | LONG * | mode) |

Description

This function gets the current touch operation mode.

Return Values

If function succeeds, the return value is nonzero.

If function fails, the return value is zero.

Parameters

1. hDevice
[in] The HANDLE that points to the device returned by pmloctl_GetDeviceHandle().
2. mode
[out] A pointer to a LONG type data, which stores the current operating mode for PenMount device driver. Please refer to previous section for possible values.

Requirements

| Item | Description |
|----------------------|--|
| Device Driver | PenMount Windows Universal Driver V2.2.0.285 |
| | PenMount Windows Universal Driver V2.2.0.309 |
| Include | pmloctlAPI.h |
| Library | pmloctlAPI.dll |

3.8. Get Touch Resolution

Syntax

```
LONG    pmloctl_GetResolution (    HANDLE    hDevice,  
                                   LONG *      resolutionX ,  
                                   LONG *      resolutionY )
```

Description

This function gets the touch input coordinate range reported by PenMount device driver. Developers can use this value to map the touch input coordinate to screen coordinates.

Return Values

If function succeeds, the return value is nonzero.

If function fails, the return value is zero.

Parameters

1. `hDevice`
[in] The HANDLE that points to the device returned by `pmloctl_GetDeviceHandle()`.
2. `resolutionX`
[out] The resolution of X axis supported by PenMount device driver.
For PenMount Windows Universal Driver V2.2.0.285 and 309, this value is 4096.
3. `resolutionY`
[out] The resolution of Y axis supported by PenMount device driver.
For PenMount Windows Universal Driver V2.2.0.285 and 309, this value is 4096.

Requirements

| Item | Description |
|----------------------|--|
| Device Driver | PenMount Windows Universal Driver V2.2.0.285 |
| | PenMount Windows Universal Driver V2.2.0.309 |
| Include | pmloctlAPI.h |
| Library | pmloctlAPI.dll |

4. Button API

This chapter will guide you through the functions that PenMount device driver features.

4.1.Set Mouse Button Type

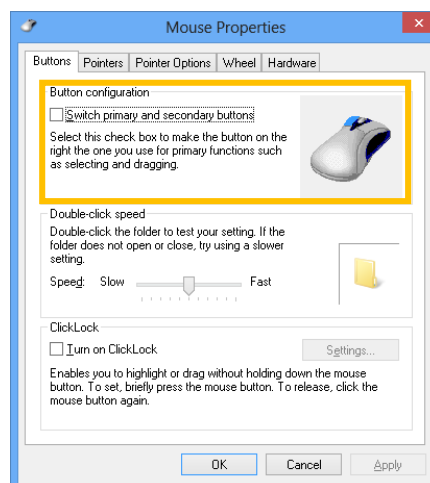
Syntax

| | | | |
|------|-------------------------|--------|----------|
| LONG | pmIoctl_SetMainButton (| HANDLE | hDevice, |
| | | LONG | button) |

Description

This function sets up the button report of PenMount device driver.

Since PenMount device driver reports mouse button events, upon user's changing the settings in **[Control Panel] -> [Mouse Properties] -> [Button Configuration] -> [switch primary and secondary buttons]**, the touches will start to generate right button event.



In this case **developers** should manually call this function to change the button settings in PenMount device driver.

Return Values

If function succeeds, the return value is nonzero.

If function fails, the return value is zero.

Parameters

1. `hDevice`
[in] The HANDLE that points to the device returned by `pmloctl_GetDeviceHandle()`.
2. `button`
[in] If the value is set to 1, the driver will left right button event on touch. If the value is set to 2, the driver will send right button event on touch.

Requirements

| Item | Description |
|----------------------|--|
| Device Driver | PenMount Windows Universal Driver V2.2.0.285 |
| | PenMount Windows Universal Driver V2.2.0.309 |
| Include | pmloctlAPI.h |
| Library | pmloctlAPI.dll |

4.2.Enable/Disable Right Mouse Button Emulation

Syntax

```
LONG    pmloctl_EnableRbuttonEmulation (    HANDLE    hDevice,  
                                           LONG    enable )
```

Description

This function enables/disables the right button emulation of PenMount driver.

Return Values

If function succeeds, the return value is nonzero.

If function fails, the return value is zero.

Parameters

1. `hDevice`
[in] The HANDLE that points to the device returned by `pmloctl_GetDeviceHandle()`.
2. `enable`

[in] The new state of right button emulation. Set it to TRUE to enable the emulation, FALSE otherwise.

Requirements

| Item | Description |
|----------------------|--|
| Device Driver | PenMount Windows Universal Driver V2.2.0.285 |
| | PenMount Windows Universal Driver V2.2.0.309 |
| Include | pmloctlAPI.h |
| Library | pmloctlAPI.dll |

4.3. Check If Right Mouse Button Emulation Is Enabled

Syntax

```
LONG    pmloctl_CheckRbuttonEmulation (    HANDLE    hDevice,  
                                           LONG *      state )
```

Description

This function gets the last state of right button emulation of PenMount driver.

Return Values

If function succeeds, the return value is nonzero.

If function fails, the return value is zero.

Parameters

1. hDevice
[in] The HANDLE that points to the device returned by pmloctl_GetDeviceHandle().
2. state
[out] The Return Values of right button emulation state. It should be either TRUE or FALSE only.

Requirements

| Item | Description |
|----------------------|--|
| Device Driver | PenMount Windows Universal Driver V2.2.0.285 |
| | PenMount Windows Universal Driver V2.2.0.309 |
| Include | pmloctlAPI.h |
| Library | pmloctlAPI.dll |

4.4.Set Interval of Right Mouse Button Emulation

Syntax

```
LONG    pmloctl_SetRbuttonInterval (    HANDLE    hDevice,  
                                         LONG        interval )
```

Description

This function sets the press and hold interval for right button emulation provided by the PenMount device driver.

Return Values

If function succeeds, the return value is nonzero.

If function fails, the return value is zero.

Parameters

1. hDevice
[in] The HANDLE that points to the device returned by pmloctl_GetDeviceHandle().
2. interval
[in] The new interval in milliseconds for press and hold to trigger the right button emulation.
For example, setting 1000 will have the driver trigger right mouse button event after user presses and holds for 1 second.

Requirements

| Item | Description |
|----------------------|--|
| Device Driver | PenMount Windows Universal Driver V2.2.0.285 |
| | PenMount Windows Universal Driver V2.2.0.309 |
| Include | pmloctlAPI.h |
| Library | pmloctlAPI.dll |

4.5. Get Current Interval of Right Mouse Button Emulation

Syntax

```
LONG    pmloctl_GetRbuttonInterval (    HANDLE    hDevice,  
                                         LONG *    interval )
```

Description

This function gets the current press and hold interval for right button emulation provided by the PenMount device driver.

Return Values

If function succeeds, the return value is nonzero.

If function fails, the return value is zero.

Parameters

1. hDevice
[in] The HANDLE that points to the device returned by pmloctl_GetDeviceHandle().
2. interval
[out] The interval in milliseconds for press and hold to trigger the right button emulation.
For example, value 1000 means that the driver trigger right mouse button event after user presses and holds for 1 second.

Requirements

| Item | Description |
|----------------------|--|
| Device Driver | PenMount Windows Universal Driver V2.2.0.285 |
| | PenMount Windows Universal Driver V2.2.0.309 |
| Include | pmloctlAPI.h |
| Library | pmloctlAPI.dll |

4.6.Set Offset Allowance of Right Mouse Button Emulation

Syntax

```

LONG    pmloctl_SetRbuttonOffset (    HANDLE    hDevice,
                                     LONG    offset )

```

Description

This function sets the right button emulation offset for PenMount device driver.

When user touches the panel, driver will start the right button timer. If user releases or moves out of the allowable offset before time up, the driver will send left mouse button event instead.

Return Values

If function succeeds, the return value is nonzero.

If function fails, the return value is zero.

Parameters

1. hDevice
[in] The HANDLE that points to the device returned by pmloctl_GetDeviceHandle().
2. offset
[in] The new offset of right button emulation in unit used by the PenMount driver.

The driver uses value 10 by default, which means user can only trigger the right mouse button emulation in a quite small area, therefore we recommend setting larger offset when using finger.

Requirements

| Item | Description |
|----------------------|--|
| Device Driver | PenMount Windows Universal Driver V2.2.0.285 |
| | PenMount Windows Universal Driver V2.2.0.309 |
| Include | pmloctlAPI.h |
| Library | pmloctlAPI.dll |

4.7. Get Current Offset of Right Mouse Button Emulation

Syntax

```
LONG    pmloctl_GetRbuttonOffset (    HANDLE    hDevice,  
                                       LONG *    offset )
```

Description

This function gets the last right button emulation offset of PenMount device driver.

Return Values

If function succeeds, the return value is nonzero.

If function fails, the return value is zero.

Parameters

1. hDevice
[in] The HANDLE that points to the device returned by pmloctl_GetDeviceHandle().
2. offset
[out] The last offset of right button emulation in unit used by the PenMount driver.

Requirements

| Item | Description |
|----------------------|--|
| Device Driver | PenMount Windows Universal Driver V2.2.0.285 |
| | PenMount Windows Universal Driver V2.2.0.309 |
| Include | pmloctlAPI.h |
| Library | pmloctlAPI.dll |

5. Filter API

This chapter will guide you through settings up the PenMount device driver filters.

5.1.Set Filter Value to Avoid Jitter

Syntax

```
LONG    pmIoctl_SetJitterFilter (          HANDLE    hDevice,  
                                         LONG        value )
```

Description

This function sets the new filter value to avoid jitter.

Return Values

If function succeeds, the return value is nonzero.

If function fails, the return value is zero.

Parameters

1. hDevice
[in] The HANDLE that points to the device returned by pmIoctl_GetDeviceHandle().
2. value
[in] The new jitter filter value. This value should be positive.
The value that driver uses by default is 3, which means a movements under 3 unit is considered jitter and will be filtered. If the value is set too large, the curve smoothness will be impacted when writing or drawing.

Requirements

| Item | Description |
|----------------------|--|
| Device Driver | PenMount Windows Universal Driver V2.2.0.285 |
| | PenMount Windows Universal Driver V2.2.0.309 |
| Include | pmIoctlAPI.h |

| | |
|----------------|----------------|
| Library | pmloctlAPI.dll |
|----------------|----------------|

5.2. Get Current Filter Value to Avoid Jitter

Syntax

```
LONG    pmloctl_GetJitterFilter (          HANDLE    hDevice,
                                         LONG *    value )
```

Description

This function gets the last jitter filter value from PenMount device driver.

Return Values

If function succeeds, the return value is nonzero.

If function fails, the return value is zero.

Parameters

1. hDevice
[in] The HANDLE that points to the device returned by pmloctl_GetDeviceHandle().
2. value
[out] The current jitter filter value used by device driver.

Requirements

| Item | Description |
|----------------------|--|
| Device Driver | PenMount Windows Universal Driver V2.2.0.285 |
| | PenMount Windows Universal Driver V2.2.0.309 |
| Include | pmloctlAPI.h |
| Library | pmloctlAPI.dll |

6. Calibration API

This chapter will guide you through changing the PenMount device driver calibration settings.

6.1.Change Calibration Parameters

Syntax

```
LONG    pmIoctl_SetCalibParameter (    HANDLE        hDevice,
                                       PMCALIB_PARAM *  param,
                                       LONG              degree )
```

Description

This function sets the calibration mode to advanced calibration.

Return Values

If function succeeds, the return value is nonzero.

If function fails, the return value is zero.

Parameters

1. hDevice
[in] The HANDLE that points to the device returned by pmIoctl_GetDeviceHandle().
2. param
[in] The calibration parameters.

The PMCALIB_PARAM structure is defined as below :

```
typedef struct _PMCALIB_PARAM
{
    LONG    mode ;

    POINT   point[25];
}          PMCALIB_PARAM ;
```

- a. mode
The reference point number used, should be 4, 9, 16, 25.
 - b. point
An array of the reference points.
3. degree
[in] The screen rotation degree when calibration utility is run.
The value should be 0, 90, 180 or 270 only. Driver will use this parameter as a reference.
Developers can pass NULL if they don't use screen rotation.

Requirements

| Item | Description |
|----------------------|--|
| Device Driver | PenMount Windows Universal Driver V2.2.0.285 |
| | PenMount Windows Universal Driver V2.2.0.309 |
| Include | pmloctlAPI.h |
| Library | pmloctlAPI.dll |

6.2. Get Current Calibration Parameters

Syntax

```

LONG    pmloctl_GetCalibParameter (
                                HANDLE    hDevice,
                                PMCALIB_PARAM * param,
                                LONG *    degree )

```

Description

This function gets the last advanced calibration parameters in use by PenMount device driver.

Return Values

If function succeeds, the return value is nonzero.

If function fails, the return value is zero.

Parameters

1. hDevice
[in] The HANDLE that points to the device returned by pmIoctl_GetDeviceHandle().
2. param
[out] The advanced calibration parameters.
Please refer to previous section for the PMCALIB_PARAM structure is definition.
3. degree
[out] the degree of screen rotation when running the calibration utility.
The value should be 0, 90, 180 or 270 only. Driver will use this parameter as a reference. Developers can pass NULL if they don't use screen rotation.

Requirements

| Item | Description |
|----------------------|--|
| Device Driver | PenMount Windows Universal Driver V2.2.0.285 |
| | PenMount Windows Universal Driver V2.2.0.309 |
| Include | pmIoctlAPI.h |
| Library | pmIoctlAPI.dll |

6.3.Change Rotation Degree

Syntax

| | | | |
|------|---------------------------|--------|----------|
| LONG | pmIoctl_SetRotateDegree (| HANDLE | hDevice, |
| | | LONG | degree) |

Description

When user rotates the desktop to degrees other than 0 degree, the PenMount device driver cannot detect this automatically, and the touch position won't be correct anymore.

PenMount has provided a user mode application "PenMount Monitor" to update the rotation setting in device driver. Developers may also use this function to set the rotation parameter.

Return Values

If function succeeds, the return value is nonzero.

If function fails, the return value is zero.

Parameters

1. hDevice
[in] The HANDLE that points to the device returned by pmIoctl_GetDeviceHandle().
2. degree
[in] The current degree used by the device driver, in counterclockwise. The values should be 0, 90, 180 or 270 only.

Requirements

| Item | Description |
|----------------------|--|
| Device Driver | PenMount Windows Universal Driver V2.2.0.285 |
| | PenMount Windows Universal Driver V2.2.0.309 |
| Include | pmIoctlAPI.h |
| Library | pmIoctlAPI.dll |

6.4. Get Current Rotation Setting

Syntax

| | | | |
|------|---------------------------|--------|----------|
| LONG | pmIoctl_GetRotateDegree (| HANDLE | hDevice, |
| | | LONG * | degree) |

Description

This function gets the last rotation parameter from PenMount device driver.

Return Values

If function succeeds, the return value is nonzero.

If function fails, the return value is zero.

Parameters

1. `hDevice`
[in] The HANDLE that points to the device returned by `pmIoctl_GetDeviceHandle()`.
2. `degree`
[out] The degree parameter that driver is using at the moment. The values should be 0, 90, 180 or 270.

Requirements

| Item | Description |
|----------------------|--|
| Device Driver | PenMount Windows Universal Driver V2.2.0.285 |
| | PenMount Windows Universal Driver V2.2.0.309 |
| Include | pmIoctlAPI.h |
| Library | pmIoctlAPI.dll |

6.5.Set Screen Edge Compensation Parameter

Syntax

```
LONG    pmIoctl_SetEdgeCompensate (    HANDLE    hDevice,  
                                         RECT *    edge )
```

Description

This function sets the screen edge compensation parameters for device driver.

Developers can use this when cursor cannot hit the edges.

Return Values

If function succeeds, the return value is nonzero.

If function fails, the return value is zero.

Parameters:

1. hDevice
[in] The HANDLE that points to the device returned by pmloctl_GetDeviceHandle().
2. edge
[in] A pointer to a RECT structure that stores the values that will be added to each edge of the screen. The RECT structure is defined in the following webpage:
<http://msdn.microsoft.com/en-us/library/windows/desktop/dd162897>
The PenMount device driver uses value 10 in default. Noted that the cursor might act strangely when setting to large values.

Requirements

| Item | Description |
|----------------------|--|
| Device Driver | PenMount Windows Universal Driver V2.2.0.285 |
| | PenMount Windows Universal Driver V2.2.0.309 |
| Include | pmloctlAPI.h |
| Library | pmloctlAPI.dll |

6.6. Get Screen Edge Compensation Parameters

Syntax

```
LONG    pmloctl_GetEdgeCompensate (    HANDLE    hDevice,  
                                       RECT *    edge )
```

Description

This function gets the last margin compensation parameters from PenMount device driver.

Return Values

If function succeeds, the return value is nonzero.

If function fails, the return value is zero.

Parameters

1. `hDevice`
[in] The HANDLE that points to the device returned by `pmloctl_GetDeviceHandle()`.
2. `edge`
[out] A pointer to a RECT structure that stores the values on each edge of the touch screen.

Requirements

| Item | Description |
|----------------------|--|
| Device Driver | PenMount Windows Universal Driver V2.2.0.285 |
| | PenMount Windows Universal Driver V2.2.0.309 |
| Include | pmloctlAPI.h |
| Library | pmloctlAPI.dll |

7. Beep API

This chapter will guide you through changing the PenMount device driver beep settings.

7.1.Set Touch Beep Mode

Syntax

```
LONG    pmIoctl_SetBeepMode (          HANDLE    hDevice,  
                                     LONG    mode )
```

Description

This function sets beep mode for touch.

Please notice that in PenMount Windows Universal Driver V2.2.0.285 and V309, the PenMount Monitor is in charge of making beep sound, so it must be running or no beep sound will be made.

Return Values

If function succeeds, the return value is nonzero.

If function fails, the return value is zero.

Parameters

1. hDevice
[in] The HANDLE that points to the device returned by pmIoctl_GetDeviceHandle().
2. mode
[in] The new beep mode. Available values are:

| Mode | Description |
|-------------------------|---|
| PMBEEP_DISABLE | PenMount Monitor will not make sound. |
| PMBEEP_BUZZER | PenMount Monitor will use buzzer to make sound. |
| PMBEEP_WAVESOUND | PenMount Monitor will use wave sound to make sound, and a PC speaker is required. |

Please notice that in Microsoft Windows 7, choosing buzzer sound works the same as choosing wave sound.

Requirements

| Item | Description |
|----------------------|--|
| Device Driver | PenMount Windows Universal Driver V2.2.0.285 |
| | PenMount Windows Universal Driver V2.2.0.309 |
| Include | pmloctlAPI.h |
| Library | pmloctlAPI.dll |

7.2. Get Current Touch Beep Mode

Syntax

```
LONG    pmloctl_GetBeepMode (          HANDLE    hDevice,  
                                     LONG *      mode )
```

Description

This function gets the current beep mode by the PenMount Monitor.

Return Values

If function succeeds, the return value is nonzero.

If function fails, the return value is zero.

Parameters

1. hDevice
[in] The HANDLE that points to the device returned by pmloctl_GetDeviceHandle().
2. mode
[out] A pointer to a LONG type data that stores the current beep mode. Please refer to previous sections for possible mode values.

Requirements

| Item | Description |
|----------------------|--|
| Device Driver | PenMount Windows Universal Driver V2.2.0.285 |
| | PenMount Windows Universal Driver V2.2.0.309 |
| Include | pmloctlAPI.h |
| Library | pmloctlAPI.dll |

7.3.Set Touch Beep Frequency

Syntax

| | | | |
|------|----------------------------|--------|-------------|
| LONG | pmloctl_SetBeepFrequency (| HANDLE | hDevice, |
| | | LONG | frequency) |

Description

This function sets beep frequency for touching.

Please notice that the PenMount Monitor is in charge of playing beep sound, and this function is only valid when using the buzzer beep mode.

Return Values

If function succeeds, the return value is nonzero.

If function fails, the return value is zero.

Parameters

1. hDevice
[in] The HANDLE that points to the device returned by pmloctl_GetDeviceHandle().
2. frequency
[in] The new beep frequency in Hz.
The frequency that the PenMount Monitor using is range from 38 Hz to 32766 Hz.

Requirements

| Item | Description |
|----------------------|--|
| Device Driver | PenMount Windows Universal Driver V2.2.0.285 |
| | PenMount Windows Universal Driver V2.2.0.309 |
| Include | pmloctlAPI.h |
| Library | pmloctlAPI.dll |

7.4. Get Current Touch Beep Frequency

Syntax

| | | | |
|------|----------------------------|--------|-------------|
| LONG | pmloctl_GetBeepFrequency (| HANDLE | hDevice, |
| | | LONG * | frequency) |

Description

This function gets beep frequency that the PenMount Monitor is using.

Return Values

If function succeeds, the return value is nonzero.

If function fails, the return value is zero.

Parameters

1. hDevice
[in] The HANDLE that points to the device returned by pmloctl_GetDeviceHandle().
2. frequency
[out] A pointer to a LONG type data which stores the current beep frequency that PenMount Monitor is using.

Requirements

| Item | Description |
|----------------------|--|
| Device Driver | PenMount Windows Universal Driver V2.2.0.285 |
| | PenMount Windows Universal Driver V2.2.0.309 |

| | |
|----------------|----------------|
| Include | pmloctlAPI.h |
| Library | pmloctlAPI.dll |

7.5.Set Touch Beep Duration

Syntax

| | | | |
|------|---------------------------|--------|------------|
| LONG | pmloctl_SetBeepDuration (| HANDLE | hDevice, |
| | | LONG * | duration) |

Description

This function sets beep duration for touch.

Please notice that making beep sound is the charge of PenMount Monitor, and this function is only valid when using buzzer beep mode.

Return Values

If function succeeds, return value is nonzero.

If function fails, return value is zero.

Parameters

1. hDevice
[in] The HANDLE that points to the device returned by pmloctl_GetDeviceHandle().
2. duration
[in] The new beep duration in milliseconds. The minimum duration is 100 ms.

Requirements

| Item | Description |
|----------------------|--|
| Device Driver | PenMount Windows Universal Driver V2.2.0.285 |
| | PenMount Windows Universal Driver V2.2.0.309 |
| Include | pmloctlAPI.h |

| | |
|----------------|----------------|
| Library | pmloctlAPI.dll |
|----------------|----------------|

7.6. Get Current Touch Beep Duration

Syntax

| | | | |
|------|---------------------------|--------|------------|
| LONG | pmloctl_GetBeepDuration (| HANDLE | hDevice, |
| | | LONG * | duration) |

Description

This function gets the current beep duration that PenMount Monitor is using.

Return Values

If function succeeds, return value is nonzero.

If function fails, return value is zero.

Parameters

1. hDevice
[in] The HANDLE that points to the device returned by pmloctl_GetDeviceHandle().
2. duration
[out] A pointer to a LONG type data which stores the current beep duration that PenMount Monitor is using.

Requirements

| Item | Description |
|----------------------|--|
| Device Driver | PenMount Windows Universal Driver V2.2.0.285 |
| | PenMount Windows Universal Driver V2.2.0.309 |
| Include | pmloctlAPI.h |
| Library | pmloctlAPI.dll |

8. Multiple Monitor API

This chapter will guide you through changing the PenMount device driver multiple monitor settings.

8.1.Enable/Disable Monitor Mapping

Syntax

```
LONG    pmIoctl_SetMultiMonitorCount (    HANDLE    hDevice,  
                                           LONG    MMCount )
```

Description

On systems with more than one monitors attached, the touch screen is mapped to the whole virtual desktop by default.

To make the touch screen work properly again, developers can call this function to enable or disable monitor mapping by updating the monitor counts in the device driver.

Return Values

If function succeeds, the return value is nonzero.

If the MMCount is less than 1, the function fails and returns zero.

Parameters

1. hDevice
[in] The HANDLE that points to the device returned by pmIoctl_GetDeviceHandle().
2. MMCount
[in] The number of monitors.
When MMCount is larger than 1, this function will enable the monitor mapping in device driver, so that every touch points will be mapped to the region defined by the pmIoctl_SetMultiMonitorConfig function.
When MMCount is equal to 1, it will disable the monitor mapping, and the device driver will ignore the region settings.

Requirements

| Item | Description |
|------|-------------|
|------|-------------|

| | |
|----------------------|--|
| Device Driver | PenMount Windows Universal Driver V2.2.0.285 |
| | PenMount Windows Universal Driver V2.2.0.309 |
| Include | pmloctlAPI.h |
| Library | pmloctlAPI.dll |

8.2.Set Mapped Monitor Rectangle

Syntax

```
LONG    pmloctl_SetMultiMonitorConfig (    HANDLE        hDevice,
                                           RECT *        pRect )
```

Description

This function defines the monitor rectangle that the touch screen is mapped to.

Please notice that this function must be called whenever a monitor changes its resolution whether this monitor is the one that the touch screen is mapped to or not.

Return Values

If function succeeds, the return value is nonzero.

If functions fails, the return value is zero.

Parameters

1. hDevice
[in] The HANDLE that points to the device returned by pmloctl_GetDeviceHandle().
2. pRect
[in] A pointer to a RECT buffer that defines the rectangle of the mapped monitor.
For example, if there are two monitors which both use 800 * 600 resolution:



If the touch screen is mapped to the primary monitor, the values of pRect should be:

```
pRect->left = 0
pRect->top = 0
pRect->right = 800
pRect->bottom = 600
```

If the touch screen is mapped to the secondary monitor, the values of pRect should be:

```
pRect->left = 800
pRect->top = 0
pRect->right = 1600
pRect->bottom = 600
```

Requirements

| Item | Description |
|----------------------|--|
| Device Driver | PenMount Windows Universal Driver V2.2.0.285 |
| | PenMount Windows Universal Driver V2.2.0.309 |
| Include | pmloctlAPI.h |

| | |
|----------------|----------------|
| Library | pmloctlAPI.dll |
|----------------|----------------|

9. NCR API

This chapter will guide you through changing the settings developed for NCR COP Project.

9.1.Set Monitor ID to Controller

Syntax

```
LONG    pmIoctl_SetMultiMonitorID (    HANDLE    hDevice,  
                                       LONG        ID )
```

Description

This function sets the corresponding monitor ID in the PenMount controller, and will be used to map touch to the correct monitor when installing the device driver.

Return Values

If function succeeds, the return value is nonzero.

If functions fails, the return value is zero.

Parameters

1. hDevice
[in] The HANDLE that points to the device returned by pmIoctl_GetDeviceHandle().
2. ID
[in] The new monitor ID. This ID should be the monitor sequence retrieved from the EnumDisplayMonitors() function.

<http://msdn.microsoft.com/en-us/library/windows/desktop/dd162610>

Requirements

| Item | Description |
|----------------------|--|
| Device Driver | PenMount Windows Universal Driver V2.2.0.285 |
| | PenMount Windows Universal Driver V2.2.0.309 |
| Include | pmIoctlAPI.h |

| | |
|----------------|----------------|
| Library | pmloctlAPI.dll |
|----------------|----------------|

9.2. Get Monitor ID from Controller

Syntax

```
LONG    pmloctl_GetMultiMonitorID (    HANDLE    hDevice,
                                       LONG *    ID )
```

Description

This function gets the current monitor ID value from the PenMount controller.

Return Values

If function succeeds, the return value is nonzero.

If functions fails, the return value is zero.

Parameters

1. hDevice
[in] The HANDLE that points to the device returned by pmloctl_GetDeviceHandle().
2. ID
[in] A pointer to a LONG buffer that stores the current monitor ID.

Requirements

| Item | Description |
|----------------------|--|
| Device Driver | PenMount Windows Universal Driver V2.2.0.285 |
| | PenMount Windows Universal Driver V2.2.0.309 |
| Include | pmloctlAPI.h |
| Library | pmloctlAPI.dll |

9.3.Update Monitor Mapping

Syntax

```
LONG    pmIoctl_UpdateMultipleMonitor (    HANDLE                hDevice )
```

Description

This function is a wrapper what calls pmIoctl_GetMultiMonitorID() for the current monitor ID, and then calls pmIoctl_SetMultiMonitorConfig() to update the multiple monitor settings in device driver.

Return Values

If function succeeds, the return value is nonzero.

If functions fails, the return value is zero.

Parameters

1. hDevice

[in] The HANDLE that points to the device returned by pmIoctl_GetDeviceHandle().

Requirements

| Item | Description |
|----------------------|--|
| Device Driver | PenMount Windows Universal Driver V2.2.0.285 |
| | PenMount Windows Universal Driver V2.2.0.309 |
| Include | pmIoctlAPI.h |
| Library | pmIoctlAPI.dll |

9.4.Set Panel Type to Controller

Syntax

| | | | |
|------|------------------------|---------------|-------------|
| LONG | pmIoctl_SetPanelType (| HANDLE | hDevice, |
| | | unsigned char | PanelType) |

Description

This function sets the corresponding panel type in the PenMount controller.

Return Values

If function succeeds, the return value is positive value.

If functions fails, the return value is zero or one of the negative values.

| Value | Definition | Description |
|-------------|-------------------------|--|
| -100 | PM_IOCTL_INVALID_HANDLE | The provided hDevice is not valid. |
| -300 | PM_IOCTL_SEND_CMD_FAIL | The set panel type command failed. Please use GetLastError() to get failing reason. |

Parameters

- hDevice
[in] The HANDLE that points to the device returned by pmIoctl_GetDeviceHandle().
- PanelType
[in] The new panel type. This value is defined by customer. Several predefined values can be used:

| Value | Description |
|-------------|----------------|
| 0xFF | 7" Panel Type |
| 0x01 | 10" Panel Type |

Requirements

| Item | Description |
|----------------------|--|
| Device Driver | PenMount Windows Universal Driver V2.2.0.330.R16 |
| Include | pmIoctlAPI.h |
| Library | pmIoctlAPI.dll |

9.5. Get Panel Type from Controller

Syntax

```
LONG    pmIoctl_GetPanelType (          HANDLE    hDevice,
                                         unsigned char * PanelType )
```

Description

This function gets the current panel type value from the PenMount controller.

Return Values

If function succeeds, the return value is positive value.

If functions fails, the return value is zero or one of the negative values.

| Value | Definition | Description |
|-------------|---------------------------|--|
| -100 | -PM_IOCTL_INVALID_HANDLE | The provided hDevice is not valid. |
| -200 | -PM_IOCTL_INVALID_POINTER | The provided PanelType pointer is not valid. |
| -300 | -PM_IOCTL_SEND_CMD_FAIL | The get panel type command failed. Please use GetLastError() to get failing reason. |

Parameters

1. hDevice
[in] The HANDLE that points to the device returned by pmIoctl_GetDeviceHandle().
2. PanelType
[in] A pointer to an unsigned char buffer that stores the current panel type. May be one of the following predefined values.

| Value | Description |
|-------|-------------|
|-------|-------------|

| | |
|-------------|----------------|
| 0xFF | 7" Panel Type |
| 0x01 | 10" Panel Type |

Requirements

| Item | Description |
|----------------------|--|
| Device Driver | PenMount Windows Universal Driver V2.2.0.330.R16 |
| Include | pmIoctlAPI.h |
| Library | pmIoctlAPI.dll |

9.6.Set Device Serial Number to Controller

Syntax

| | | | |
|------|------------------------------|----------------|------------|
| LONG | pmIoctl_SetDeviceSerialNum (| HANDLE | hDevice, |
| | | unsigned short | SerialNo) |

Description

This function sets the corresponding serial number in the PenMount controller.

Return Values

If function succeeds, the return value is positive value.

If functions fails, the return value is zero or one of the negative values.

| Value | Definition | Description |
|-------------|--------------------------|---|
| -100 | -PM_IOCTL_INVALID_HANDLE | The provided hDevice is not valid. |
| -300 | -PM_IOCTL_SEND_CMD_FAIL | The get serial number command failed. Please use GetLastError() to get failing reason. |

Parameters

1. hDevice
[in] The HANDLE that points to the device returned by pmIoctl_GetDeviceHandle().
2. SerialNo
[in] The new serial number. This value is defined by customer.

Requirements

| Item | Description |
|----------------------|--|
| Device Driver | PenMount Windows Universal Driver V2.2.0.330.R15 |
| Include | pmIoctlAPI.h |
| Library | pmIoctlAPI.dll |

9.7. Get Device Serial Number from Controller

Syntax

```
LONG    pmIoctl_GetDeviceSerialNum (    HANDLE        hDevice,  
                                         unsigned short *  SerialNo )
```

Description

This function gets the current serial number value from the PenMount controller.

Return Values

If function succeeds, the return value is positive value.

If functions fails, the return value is zero or one of the negative values.

| Value | Definition | Description |
|-------------|---------------------------|---|
| -100 | -PM_IOCTL_INVALID_HANDLE | The provided hDevice is not valid. |
| -200 | -PM_IOCTL_INVALID_POINTER | The provided SerialNo pointer is not valid. |

| | | |
|-------------|--------------------------------|---|
| -300 | -PM_IOCTL_SEND_CMD_FAIL | The get serial number command failed. Please use GetLastError() to get failing reason. |
|-------------|--------------------------------|---|

Parameters

3. hDevice
[in] The HANDLE that points to the device returned by pmioctl_GetDeviceHandle().
4. SerialNo
[in] A pointer to an unsigned short buffer that stores the current serial number.

| Value | Description |
|---------------|-----------------------------|
| 0xFFFF | Serial Number Uninitialized |

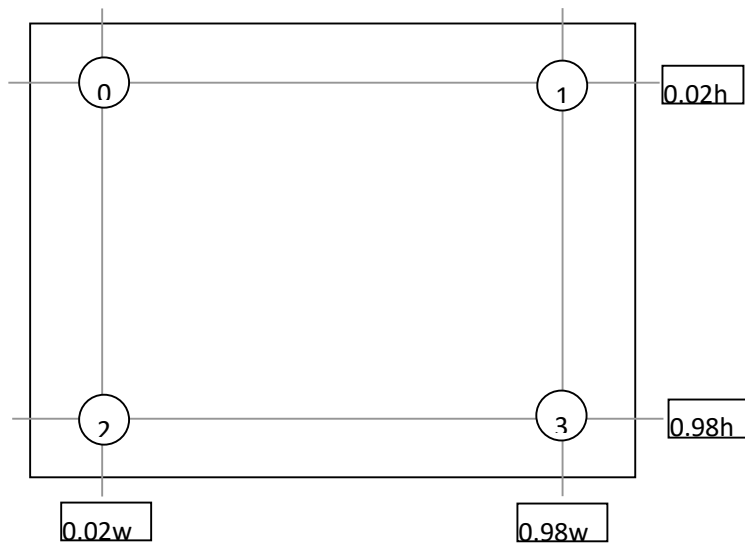
Requirements

| Item | Description |
|----------------------|--|
| Device Driver | PenMount Windows Universal Driver V2.2.0.330.R15 |
| Include | pmioctlAPI.h |
| Library | pmioctlAPI.dll |

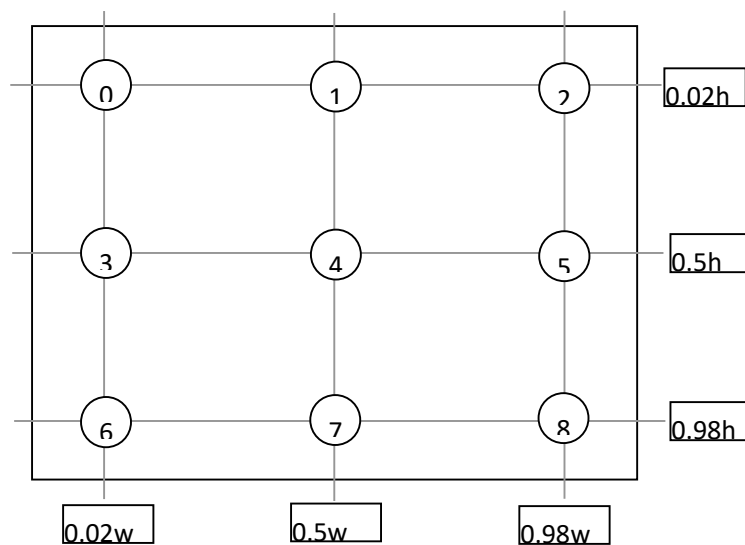
Appendix A: Calibration Application Guide

When developing custom calibration utilities with PenMount calibration API, draw calibration points on the correct screen position by following the guides below:

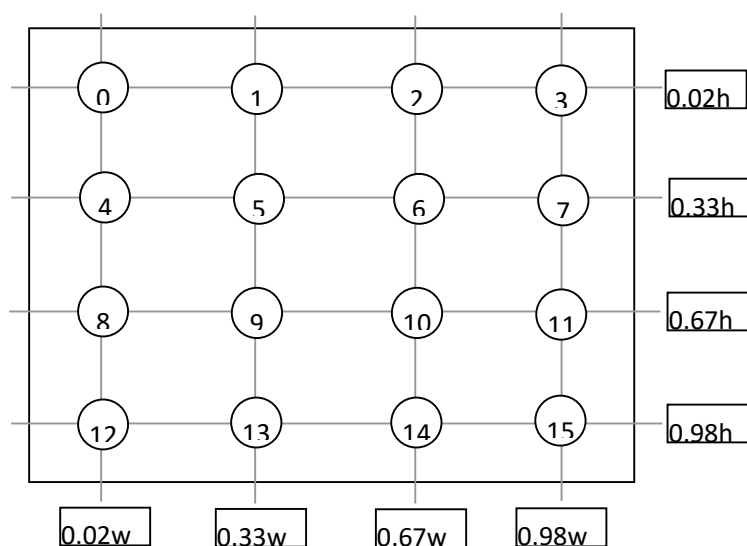
1. 4 point calibration mode



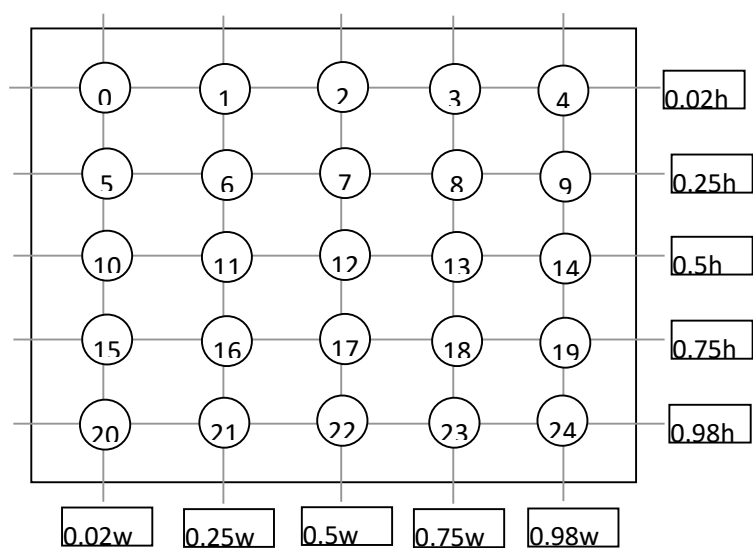
2. 9 point calibration mode



3. 16 point calibration mode



4. 25 point calibration mode



Appendix B: PenMount Calibration Setting Guide

This appendix describes the calibration settings in use by PenMount device drivers. By default, the PenMount device driver access settings from the system registry. This means that when system starts, the device driver loads settings from the registry, and when user change settings in the PenMount Control Panel, the registry settings are also updated.

The device driver can also use the calibration settings from PenMount touch controller, if the controller itself supports this feature, and if proper flags are set in the registry.

1. Registry Location

The registry locations listed in this section are based on the PenMount Universal Driver V2.1.0.263. These locations may be different in older versions of PenMount device drivers.

(a) USB Interface

The device settings can be found under:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\pmhidusb\Parameters\n
```

Where n is the index of the PenMount USB touch controller, started from 0.

The device driver maintains the index and device mapping, which can be found under:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\pmhidusb\Enum
```

(b) RS-232 Interface

The device settings can be found under:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\pmmouser\Parameters\n
```

Where n is the index of the PenMount serial device, started from 1.

The device index and device mapping can be found under:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\pmmouser\Enum
```

In general, index 1 indicates the settings for the PenMount touch controller attached to COM1.

2. Calibration Settings

This section describes the calibration settings in use by PenMount device driver V2.2.0.283 and earlier versions. Please notice that all changes of the settings will not take effect until the device or system restarts. If you wish to change settings that will take effect immediately, please use the PenMount API instead.

| Value | Description |
|---------------------|--|
| CalibPoints | The calibration mode. |
| 0 | Standard calibration |
| 4 | Advanced 4 point mode |
| 9 | Advanced 9 point mode |
| 16 | Advanced 16 point mode |
| 25 | Advanced 25 point mode |
| Rotate | The touch panel orientation set by the calibration utility. |
| 1 | Default orientation |
| 2 | Flip y axis |
| 3 | Flip x axis |
| 4 | Flip x and y axis |
| 5 | Swap x and y axis |
| 6 | Swap x and y axis, then flip y axis |
| 7 | Swap x and y axis, then flip x axis |
| 8 | Swap x and y axis, then flip x and y axis |
| EnableEEPROM | This enables the device driver to load calibration data from controller; it also enables the device driver to save calibration data to controller. |
| 0 | Only use the calibration data in the registry |
| 1 | Use the calibration data in controller |

| Standard Calibration Settings | |
|-------------------------------|---------------------------|
| Cal_Left_X | The minimum x axis value. |
| Cal_Left_Y | The minimum y axis value. |
| Cal_Right_X | The maximum x axis value. |
| Cal_Right_Y | The maximum y axis value. |

| Advanced Calibration Settings | |
|-------------------------------|--|
| X_0 | The x axis value of calibration reference point 0. |
| Y_0 | The y axis value of calibration reference point 0. |
| X_1 | The x axis value of calibration reference point 1. |
| Y_1 | The y axis value of calibration reference point 1. |

| | |
|-------------|---|
| X_2 | The x axis value of calibration reference point 2. |
| Y_2 | The y axis value of calibration reference point 2. |
| X_3 | The x axis value of calibration reference point 3. |
| Y_3 | The y axis value of calibration reference point 3. |
| X_4 | The x axis value of calibration reference point 4. |
| Y_4 | The y axis value of calibration reference point 4. |
| X_5 | The x axis value of calibration reference point 5. |
| Y_5 | The y axis value of calibration reference point 5. |
| X_6 | The x axis value of calibration reference point 6. |
| Y_6 | The y axis value of calibration reference point 6. |
| X_7 | The x axis value of calibration reference point 7. |
| Y_7 | The y axis value of calibration reference point 7. |
| X_8 | The x axis value of calibration reference point 8. |
| Y_8 | The y axis value of calibration reference point 8. |
| X_9 | The x axis value of calibration reference point 9. |
| Y_9 | The y axis value of calibration reference point 9. |
| X_10 | The x axis value of calibration reference point 10. |
| Y_10 | The y axis value of calibration reference point 10. |
| X_11 | The x axis value of calibration reference point 11. |
| Y_11 | The y axis value of calibration reference point 11. |
| X_12 | The x axis value of calibration reference point 12. |
| Y_12 | The y axis value of calibration reference point 12. |
| X_13 | The x axis value of calibration reference point 13. |
| Y_13 | The y axis value of calibration reference point 13. |
| X_14 | The x axis value of calibration reference point 14. |
| Y_14 | The y axis value of calibration reference point 14. |
| X_15 | The x axis value of calibration reference point 15. |
| Y_15 | The y axis value of calibration reference point 15. |
| X_16 | The x axis value of calibration reference point 16. |
| Y_16 | The y axis value of calibration reference point 16. |
| X_17 | The x axis value of calibration reference point 17. |
| Y_17 | The y axis value of calibration reference point 17. |
| X_18 | The x axis value of calibration reference point 18. |
| Y_18 | The y axis value of calibration reference point 18. |
| X_19 | The x axis value of calibration reference point 19. |
| Y_19 | The y axis value of calibration reference point 19. |
| X_20 | The x axis value of calibration reference point 20. |
| Y_20 | The y axis value of calibration reference point 20. |
| X_21 | The x axis value of calibration reference point 21. |
| Y_21 | The y axis value of calibration reference point 21. |
| X_22 | The x axis value of calibration reference point 22. |
| Y_22 | The y axis value of calibration reference point 22. |
| X_23 | The x axis value of calibration reference point 23. |
| Y_23 | The y axis value of calibration reference point 23. |
| X_24 | The x axis value of calibration reference point 24. |

| | | |
|-----------------------------|--|---------------------------|
| Y_24 | The y axis value of calibration reference point 24. | |
| XMinOffset | The edge compensation value of the left side of touch panel. | |
| XMaxOffset | The edge compensation value of the right side of touch panel. | |
| YMinOffset | The edge compensation value of the top of touch panel. | |
| YMaxOffset | The edge compensation value of the bottom of touch panel. | |
| RefScreenOrientation | The display rotation degree (counter clockwise) when doing calibration. | |
| | 0 | No display rotation |
| | 90 | Display rotate 90 degree |
| | 180 | Display rotate 180 degree |
| | 270 | Display rotate 270 degree |
| RelRotationDegCCW | The current display rotation degree (counter clockwise). This value will be updated by the PenMount monitor. | |
| | 0 | No display rotation |
| | 90 | Display rotate 90 degree |
| | 180 | Display rotate 180 degree |
| | 270 | Display rotate 270 degree |

Multiple Monitor Settings

| | | |
|-------------------------|---|-----------------------------|
| MultiMonitorMode | The monitor mapping mode. | |
| | 1 | Monitor mapping is disabled |
| | > 1 | Monitor mapping is enabled |
| XOFFSET | The x axis value of upper left corner of this monitor, assume that the full virtual desktop width is 1024. | |
| XSCALE | The width of this monitor, assume that the full virtual desktop width is 1024. | |
| YOFFSET | The y axis value of upper left corner of this monitor, assume that the full virtual desktop height is 1024. | |
| YSCALE | The height of this monitor, assume that the full virtual desktop width is 1024. | |

Appendix C: PenMount Control Panel Command Line Options

PenMount Control Panel is a program that allows users to perform touch screen calibration and the configuration of device driver behavior. Users can make use of this Control Panel by clicking the PenMount Control Panel icon in system's start menu, or run DMCCtrl.exe directly in Command Prompt. The DMCCtrl.exe is placed in the folder where PenMount Windows Universal Driver is installed.

When running DMCCtrl.exe in Command Prompt, users can pass additional arguments to perform specific tasks. These arguments are also useful for developers when they want to integrate functions such as touchscreen calibration into their own program. In the following table the available options of DMCCtrl.exe are listed:

| Option | Parameters | Meaning |
|---------------------|---------------|---|
| -calibration | [0 4 9 16 25] | Perform touch screen calibration. |
| | | 0 Standard calibration |
| | | 4 Advanced 4 point calibration |
| | | 9 Advanced 9 point calibration |
| | | 16 Advanced 16 point calibration |
| | | 25 Advanced 25 point calibration |
| -timeout | iTimeout | <p>Specify the calibration timeout in seconds. If no touch input during this interval, the calibration process will be terminated. To specify timeout value, users can also set the "CalibrationTimeOut" registry value in HKEY_CURRENT_USER\SOFTWARE\PenMount.</p> <p>If this value is missing, the -timeout parameter will be used.</p> <p>If no timeout is specified, the calibration program will use 10 second as the default timeout value.</p> |

| | | | | | | | | | | |
|----------------------|---------------------|---|---|---------------------|---|----------------|---|-----------------|---|------------------|
| -showcursor | | Specify whether to show the mouse cursor during touch screen calibration. | | | | | | | | |
| -mmon | | Perform multiple monitor mapping process. This option should not combined with other options. | | | | | | | | |
| -mmondev | iMon iDev | <p>Directly maps a specific PenMount touch controller to a monitor.</p> <p>iMon is the index of the monitor, started from 0.</p> <p>iDev is the index of the PenMount device, started from 0. All PenMount devices are listed in the About page of PenMount Control Panel.</p> <p>For example, DMCCtrl.exe –mmondev 0 0 will map the first PenMount touch controller to monitor 0.</p> | | | | | | | | |
| -opsel | iOP iDev | <p>Change the operation mode of a specific PenMount touch controller.</p> <p>iDev is the index of the PenMount device.</p> <p>iOP is the operation mode index. Possible values are:</p> <table><tr><td>0</td><td>Pen Input Emulation</td></tr><tr><td>1</td><td>Click on Touch</td></tr><tr><td>2</td><td>Mouse Emulation</td></tr><tr><td>3</td><td>Click on Release</td></tr></table> | 0 | Pen Input Emulation | 1 | Click on Touch | 2 | Mouse Emulation | 3 | Click on Release |
| 0 | Pen Input Emulation | | | | | | | | | |
| 1 | Click on Touch | | | | | | | | | |
| 2 | Mouse Emulation | | | | | | | | | |
| 3 | Click on Release | | | | | | | | | |
| -disabletouch | | Disable touch report. | | | | | | | | |
| -enabletouch | | Enable touch report. | | | | | | | | |