Elo TouchSystems Software Driver User Guide

Device Specific Registry Keys

Introduction

Registry Keys for Serial

Touch Mode

Differentiate Touch Data from Mouse Messages in Windows Application

Right Click On Hold

Serial Port Configuration

Enable / Disable Touch

Beep on Touch

Drag Delay

Full Screen Bounding Rectangle and Bounding Mode

Calibration

Edge of Touchscreen Cursor Acceleration

Registry Keys for USB Device

Enable / Disable Touch

Full Screen Bounding Rectangle and Bounding Mode

Calibration

Edge of Touchscreen Cursor Acceleration

Registry Keys for HID

P/N008579
Revision G

Disclaimer
The information in this document is subject to change without notice. Elo TouchSystems makes no representations or warranties with respect to the contents hereof, and specifically disclaims any implied warranties of merchantability or fitness for a particular purpose. Elo TouchSystems reserves the right to revise this publication and to make changes from time to time in the content hereof without obligation of Elo TouchSystems to notify any person of such revisions or changes.

Top of page

New Features

Universal Driver 4.X has the following new features:

- Support for AccuTouch five-wire resistive, AT4 four-wire resistive, CarrollTouch infrared, IntelliTouch/iTouch surface acoustic wave and Surface capacitive.
- The Universal driver package is has been localized to support the English, French, German, Spanish, Italian, Japanese, Simplified Chinese and Traditional Chinese.
- To keep the size of the Elo Universal driver default download package to a minimum, the Universal Driver package is provided as two separate packages:
  - Driver package
  - Users Guide
- The Universal Users Guide is also localized to support the English, French, German, Spanish, Italian, Japanese, Simplified Chinese and Traditional Chinese. The English Users Guide is included in the default download package. Language specific users guides can be downloaded from http://elotouch.com/Support/Downloads/dnld.asp#notes
- Surface Capacitive calibration has been modified to automatically timeout after 30 seconds if there is no user interaction. Previous versions made five attempts before switching to the next monitor or exiting.
- The initial delay before the right click on hold time out icon appears is configured by a registry value entry. Please see Appendix A for details on how to use this feature.
- If the system does not have a keyboard and mouse connected to it, the disable touch feature cannot be used from the task bar or control panel.
- For Infrared touchmonitors, special support has been added to detect and report blocked beams. The Advanced page for the touchmonitor in the Control Panel is used to configure this feature.
- A special hardware beep configuration feature for Infrared touchmonitors that include this special hardware feature has been added. The Advanced page for the touchmonitor in the Control Panel is used to configure this feature.

XPU 4.X can be preinstalled on a system that has no support for a mouse or HID devices, however, this requires the user to insert the Windows system disk during the installation.

Acoustic Pulse Recognition (APR)

- Support for Acoustic Pulse Recognition
- New way of sensing touch by recognizing the sound created when the glass is touched at a given position.

Bug fixes in this version:

- Intermittent loss of touch and activation of and the "Disable Touch" feature in the Elo control panel
- Windows Server 2003 would hang during the boot process because the Elo Service

Application failed to load.

- The EloGetTouch API function would not return touch data within a "no touch" rectangle. It now returns touch data irrespective of the touch rectangles defined on the screen.
- The EloGetTouch API function reports incorrect coordinates in 'Drag' mode.
- The installation application sometimes causes a system message pop-up dialog box that prompts users to select the file location when installing a Serial only package.
- The tool tray menu sometimes hides the taskbar. It now appears above the taskbar and is larger to permit easier option selection.
- The touch mode sometimes changes to Mouse emulation mode after the video alignment application exited with a timeout.

Home Page

Introduction

The Elo TouchSystems Universal Driver package contains:

- Native 32-bit drivers, for both serial and USB touchscreen controllers. Designed for Microsoft Windows 2K/XP and Windows Vista.
- Setup application.
- Video Alignment application, sometimes known as a "Calibration" application.
- Control Panel application.
- Tool Tray application.
- Right Button application.
- Edge Acceleration application.
- Center Desktop application.

Additional features for this driver include companion application programs. These may be obtained by contacting Elo's Applications Engineering Department (see Contacting Elo).

While this driver and User Manual are written for Windows 2K/XP and Windows Vista, differences in nomenclature between the three operating systems do exist. This manual uses the nomenclature and descriptions for Windows XP in cases where Windows 2K/XP and Windows Vista differ.

All Elo touchscreen controllers that utilize Elo's SMARTSET command set and data protocol are supported by this driver, with the exception of Elo's older ISA Bus controllers. The newest controller supported by Elo is the 5020. Supported controllers and touchmonitors which contain these controllers currently include:

- All Elo Entuitive brand touchmonitors with an internal serial or USB controller.
- All Touch Panel Systems (TPS) touchmonitors with an internal or external serial or USB controller.
- Special products from Elo Rochester containing one of the above controllers.
- Elo USB Controllers: AccuTouch 3000U, CarrollTouch 4000U, 4500U, IntelliTouch 5000RSU, 5020, 2701, 2500U, 2500UZ, 2600, 2310, 2216.
- Elo Serial Controllers: AccuTouch 2210, 2215, CarrollTouch 4000S, IntelliTouch 5000RSU, 5020, 2701, 2500S, 2500SZ, 2500SG, 2310B, 2310, 2216.
- Monitors from third party vendors containing one of the above controllers.

Home Page

Installing the Driver

Hardware

Touchscreen controllers must be configured for the driver prior to installation. Elo typically ships controllers in a default setup that is compatible with this driver. Controller requirements are:

Serial

- Serial controllers must conform to these minimum requirements:
  - Baud rate-9600 baud.
  - Data format-SMARTSET
  - It is desirable for a full handshaking connection to be established between the touchscreen controller and the computer, but it is not required. The controller will function with this driver in a "two-wire", Receive Data and Ground (RxD and Gnd) configuration. The consequences of a two-wire connection are:
    - Automatic detection of the controllers on serial ports during setup is not possible
    - Controller information will not be registered in the Property Page of the Control Panel

USB

- USB controllers require no configuration prior to installation.
- Elo's USB controllers are Human Interface Device (HID) compliant. The Windows 2K/XP and Windows Vista operating systems have native HID drivers that provide low-level support for Elo touchscreen controllers. An Elo touchmonitor with a USB touchscreen controller installed will provide the following limited operation with these native HID drivers. Operation of the controller will be limited without the installation of the software discussed in the next section:
  - The mouse cursor will move on the display in response to touch
  - The direction of motion on the display will depend on the orientation of the touchscreen and the defaults programmed into the controller and the driver
  - No "beep" will be heard when the screen is touched
  - No Control Panel application is available to configure the operation of the driver.

Software

There are many options to consider when installing the driver files. This section will explain the most common ways that the driver is installed. Consult Elo Applications Engineering for situations not covered in this section.

Setup from Windows Desktop

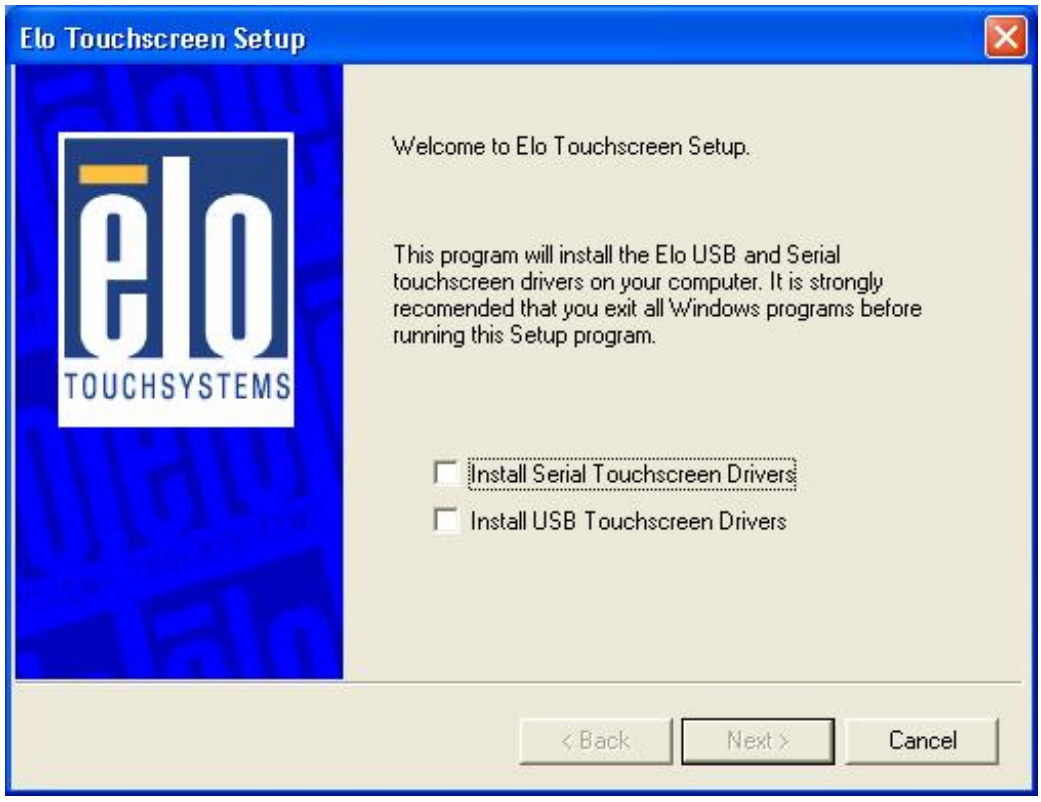
Single monitor, serial controller

- Run the self-extracting zip file to unzip the files, using the Run EloSetup option
- EloSetup will launch.
- Select the language for the Universal driver package.

  This language is used for all Elo components in future on the system. To change the user language you must install the package again on the system. Default selection from the dropdown uses the current users input language from the Windows subsystem.

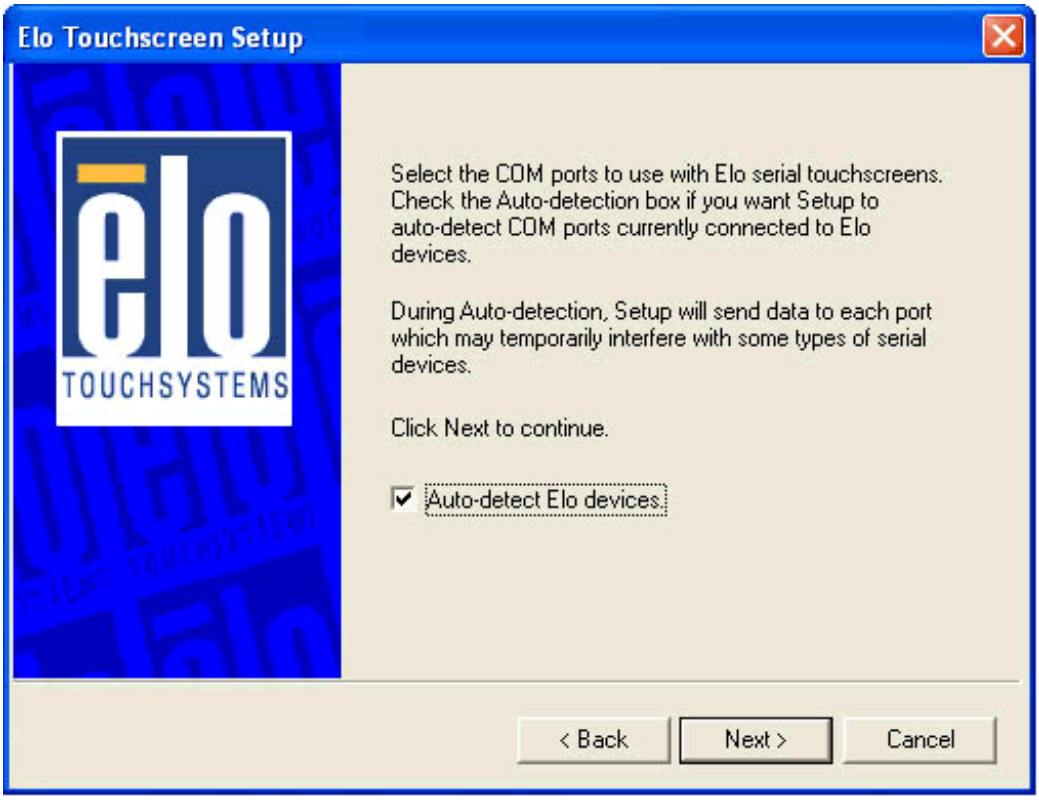
  - Currently the languages supported are English, French, German, Spanish, Italian, Japanese, Simplified Chinese and Traditional Chinese.

- Check the **Install Serial Touchscreen Drivers** box.



- Review and accept the License Agreement

- Check **Auto-detect Elo devices** if your touchmonitor serial cable is connected to the computer and the serial cable supports hardware handshaking. If you connect the serial cable later, do not check the **auto-detect** box. Click the **Next** button.

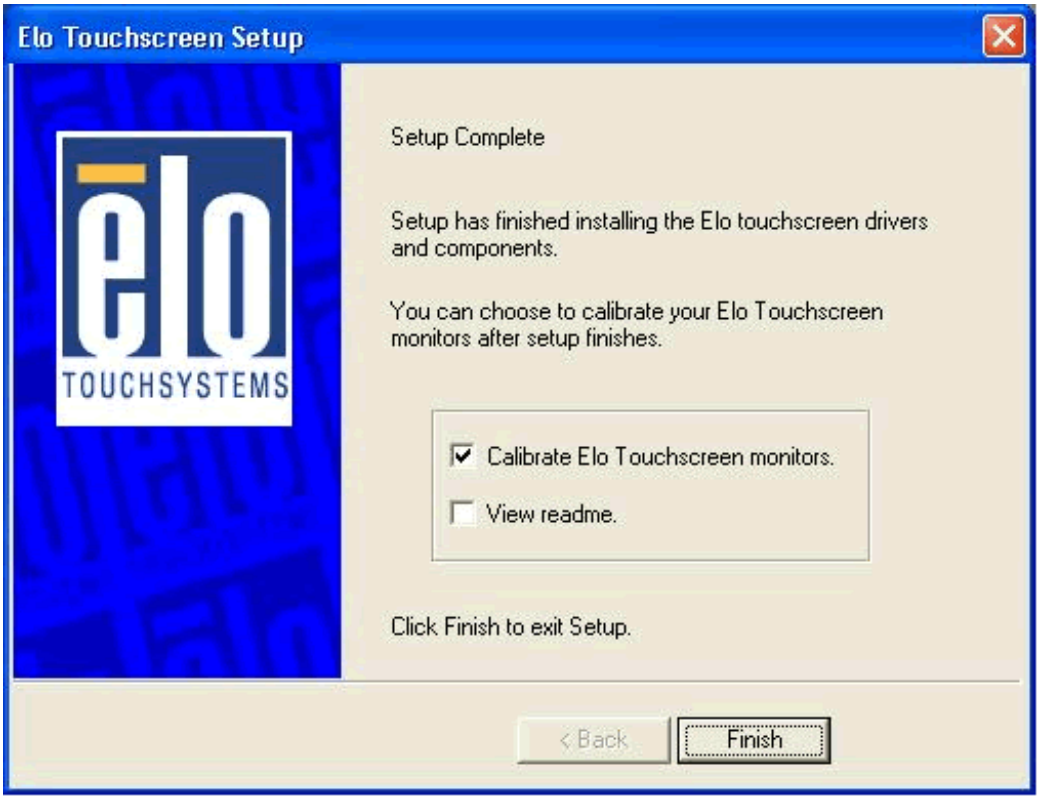


- A list of all the serial ports that the system detected is shown. If you checked the auto-detect box in the previous screen, and your touchmonitor serial cable is attached to the computer, the COM port that you selected should be checked in the list. If this is the case, click **Next** and proceed to the next screen.

- The COM port you selected will be shown in the window of this screen. This is your last opportunity to change your COM port selection before the driver files are installed. Use the Back button(s) to change any of the selections you previously made. Click **Next** to complete the installation of the driver files.

- Depending on the revision of the driver you are installing, Windows may advise you that the "software you are installing has not passed Windows logo testing." . Select **Continue Anyway** to proceed with the installation. This message can be suppressed in future installations by changing the selection in Driver Signing Options. From the **Control Panel**>**System**>**Hardware**>**Device Manager**>**Driver Signing**.

- When the **Setup Complete** screen appears, you may choose to run calibration the (Elo Video Alignment program, EloVA) immediately or wait until later. If you choose not to run this program now, you can run it from the Elo Control Panel application.



Single monitor, USB controller

1. Run the self-extracting zip file to unzip the files, using the **Run EloSetup** option.
2. EloSetup will launch. Select the language for the Universal driver package.
3. Check the **Install USB Touchscreen Drivers** box.
4. Review and accept the License Agreement.
5. The driver files will install. When the **Setup Complete** screen appears, you may choose to run calibration (the Elo Video Alignment program, EloVA) immediately or wait until later. If you choose not to run this program now, you can run it from the Elo Control Panel application.

Multiple monitors, Serial controllers

1. Follow the general procedure for Single monitor, serial controller installation above, auto detecting or selecting all the serial ports that you will use.
2. The driver files will install. When the **Setup Complete** screen appears, you may choose to run calibration (the Elo Video Alignment program, EloVA) immediately or wait until later. If you choose not to run this program now, you can run it from the Elo Control Panel application. When EloVA runs for the first time, it will attempt to calibrate all controllers and/or serial ports that were installed. Press the **Esc** key on the keyboard to terminate or skip calibration for any monitor, or allow the program to time out (as indicated by the progress bar reaching maximum).

Multiple monitors, USB controllers

1. Follow the procedure for Single monitor, USB controller installation above.
2. The driver files will install. When the **Setup Complete** screen appears, you may choose to run calibration (the Elo Video Alignment program, EloVA) immediately or wait until later. If you choose not to run this program now, you can run it from the Elo Control Panel application. When EloVA runs for the first time, it will attempt to calibrate all controllers and/or serial ports that were installed. Press the Esc key on the keyboard to terminate or skip calibration for any monitor, or allow the program to time out (as indicated by the progress bar reaching maximum). The program will continue until all controllers and/or ports have been calibrated.

Multiple monitors, Serial and USB controllers

1. Run the self-extracting zip file to unzip the files, using the Run EloSetup option.

2. EloSetup will launch. Check both the "Install Serial Touchscreen Drivers" and "Install USB Touchscreen Drivers" boxes.

3. Review and accept the License Agreement.

4. The program will appear to proceed as a single/multiple serial controller installation but USB files will also install. The driver files will install. When the **Setup Complete** screen appears, you may choose to run calibration (the Elo Video Alignment program, EloVA) immediately or wait until later. If you choose not to run this program now, you can run it from the Elo Control Panel application. When EloVA runs for the first time, it will attempt to calibrate all controllers and/or serial ports that were installed. Press the **Esc** key on the keyboard to terminate or skip calibration for any monitor, or allow the program to time out (as indicated by the progress bar reaching maximum). The program will continue until all controllers and/or ports have been calibrated.

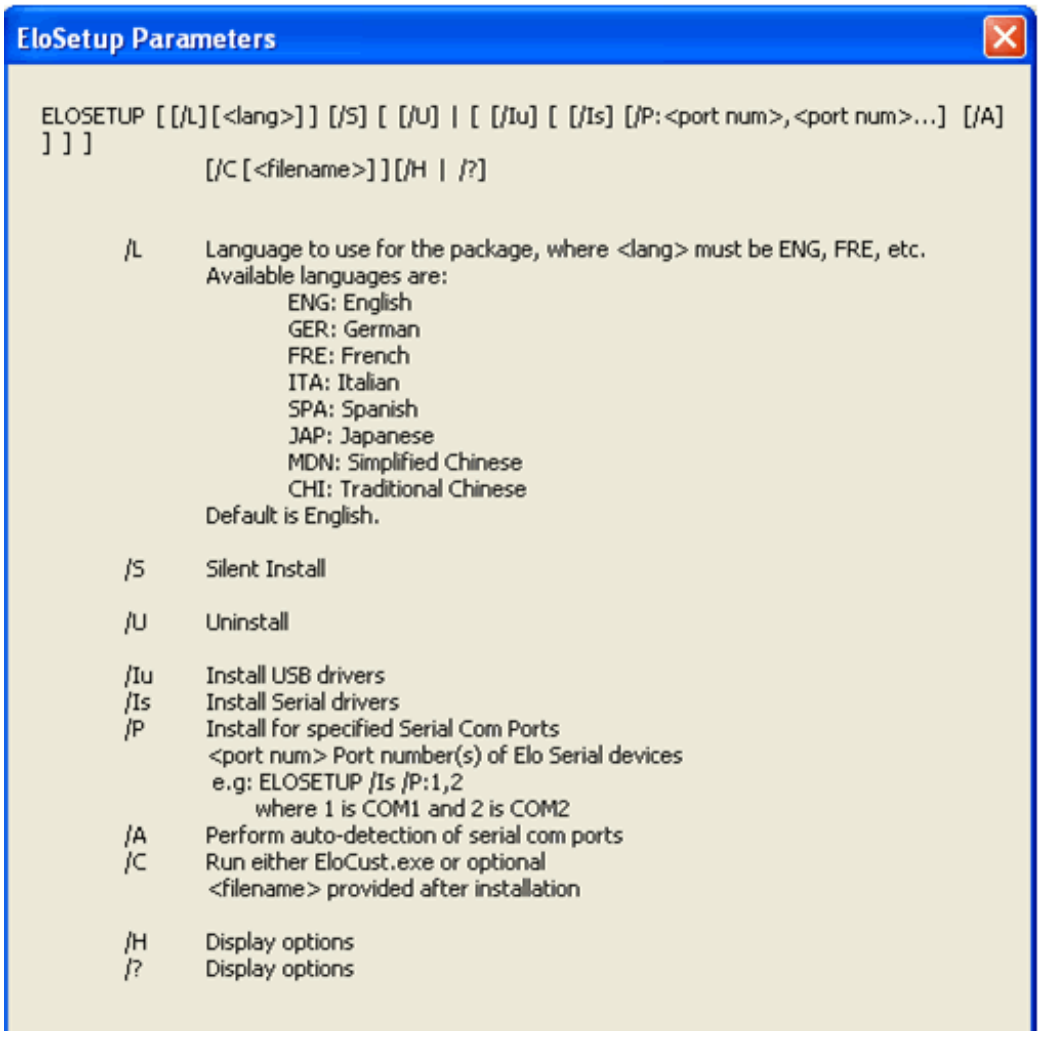
Silent Install

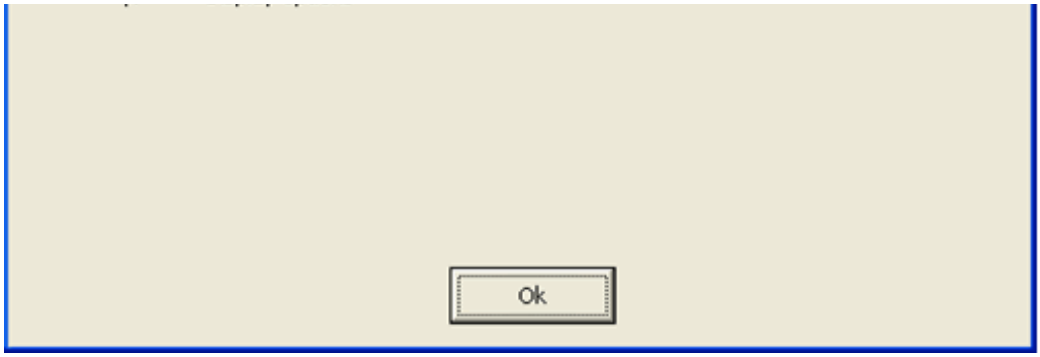
The EloSetup program for this driver may also be run as an attended or unattended program from a command line, batch file, etc. Example usage:

To Silent install for USB controllers use
EloSetup /iu /s

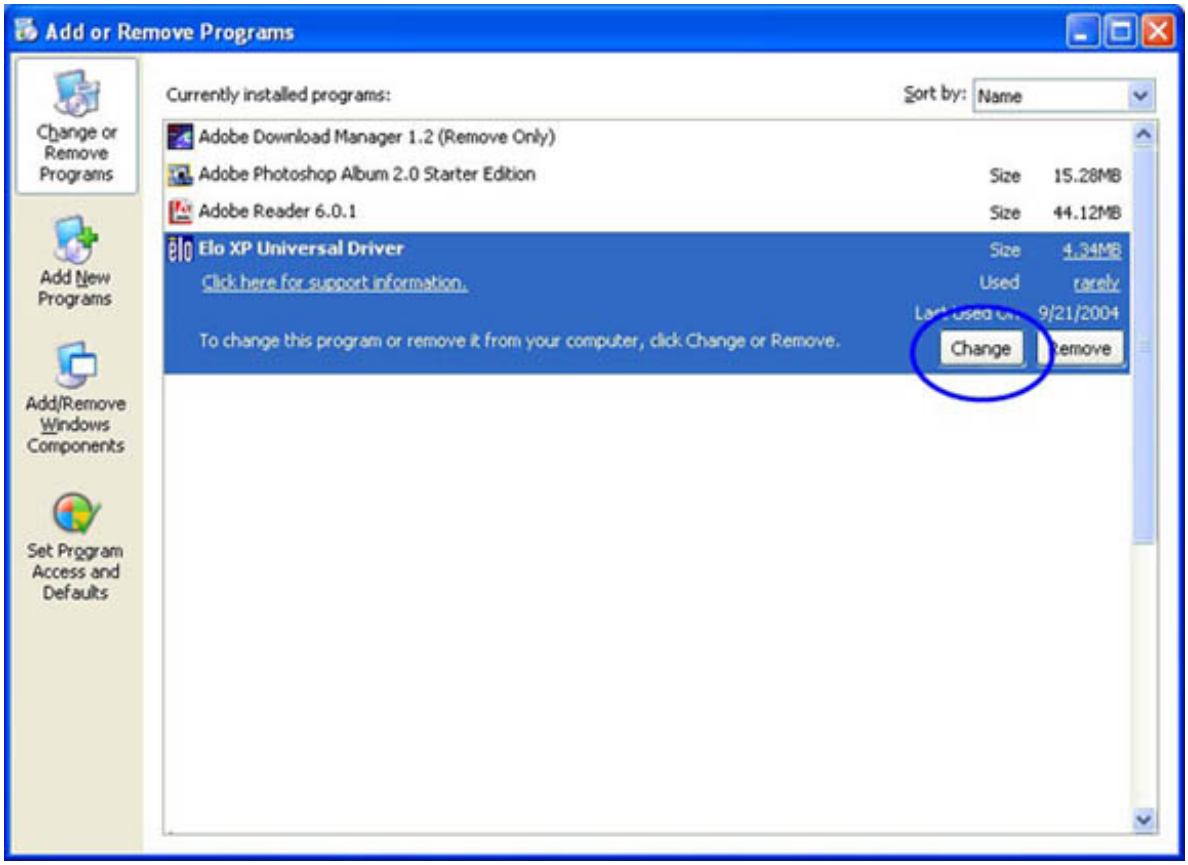
To Silent install for a Serial controller on COM1 use
EloSetup /is /P:1 /s

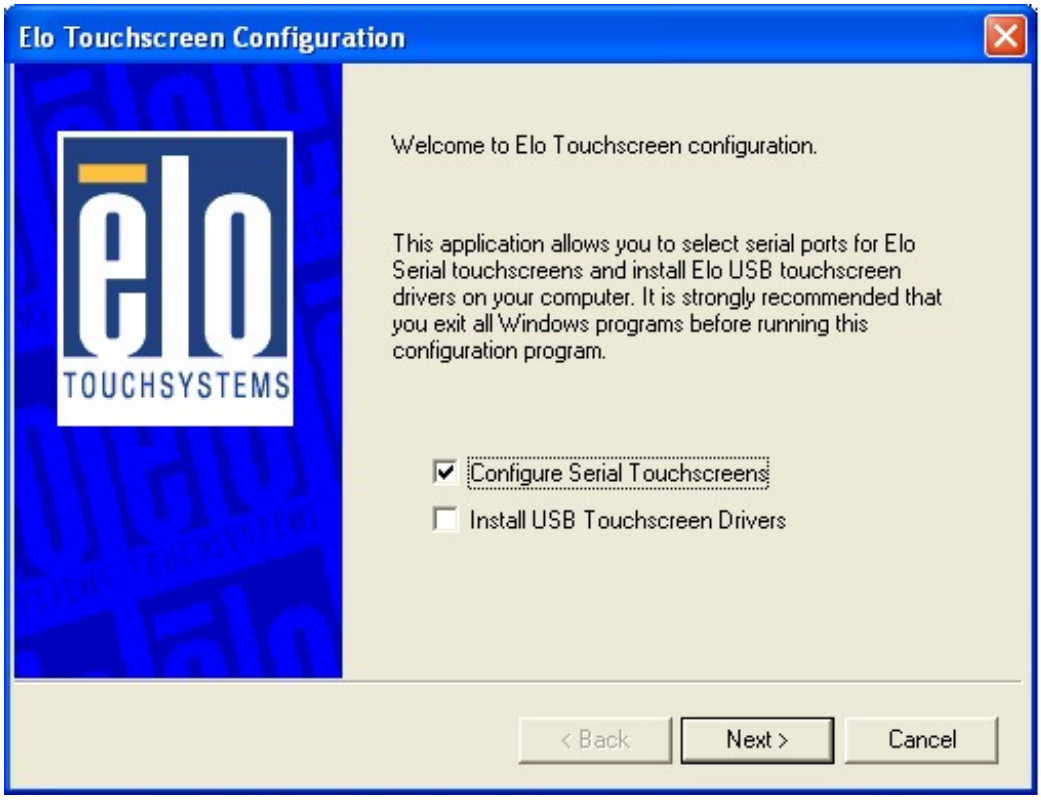
To view all the options available for this installation method, run EloSetup as **EloSetup /h**.



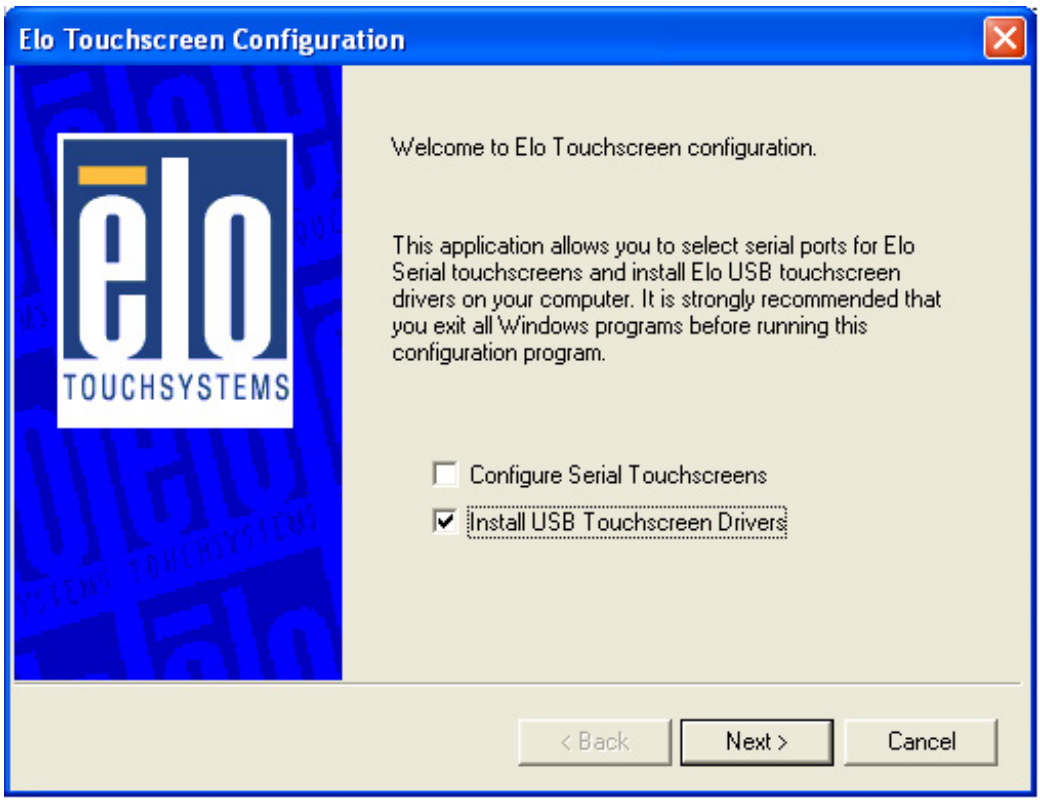


- **Adding Additional Serial Controllers**
  - Click on the **Change** option from the Add/Remove programs in the Control Panel, Elo entry.





- **Adding Additional USB Controllers**
  - If there are Elo USB devices already installed, plug the USB cable from the touchmonitor into the computer and run EloVA to calibrate the touchmonitors.
  - If there are no Elo USB devices attached to the system, click on the **Change** option from the Add/Remove programs in the Control Panel, Elo entry.



Installing the APR Drivers

The APR driver automatically detects the appropriate controller for installation. APR only works:

- on Windows XP
- with USB controllers

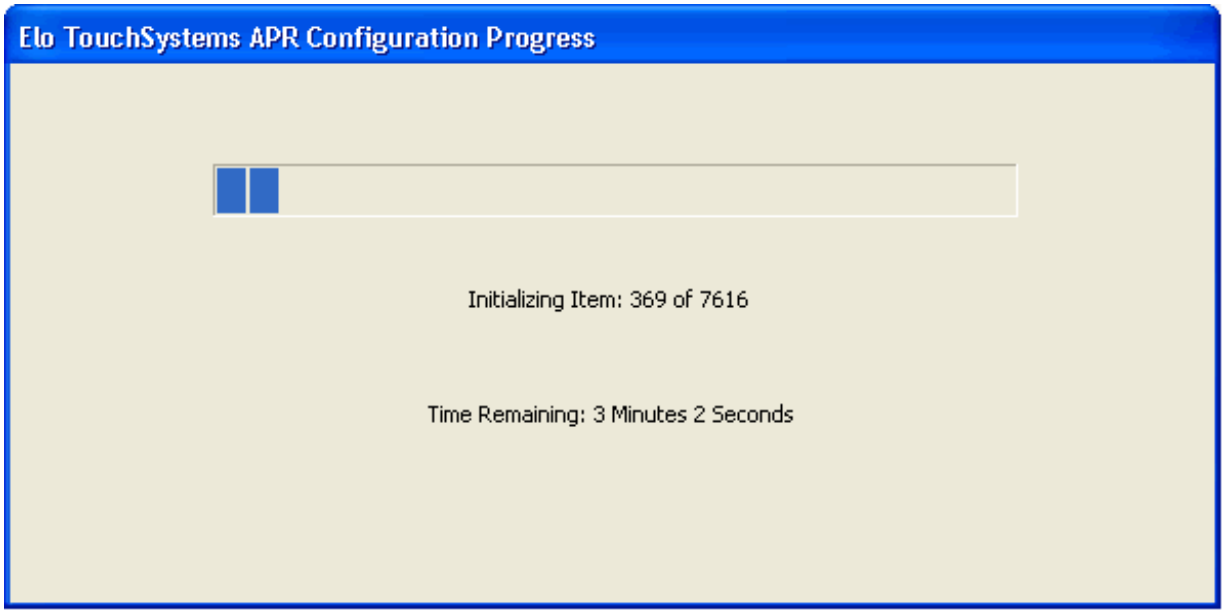
1. Run the self-extracting zip file to unzip the files.

2. EloSetup will launch. Select the language for the Universal driver package.

3. Check the **Install USB Touchscreen Drivers** box.

   Note: The **Install Serial Touchscreen Drivers** option is not available for APR monitors.

When a new APR monitor is first installed on a PC, you may see the following dialog box while the driver is reading the information from the sensor. This process may take up to 4 minutes before touch begins to work.



1. Review and accept the License Agreement.

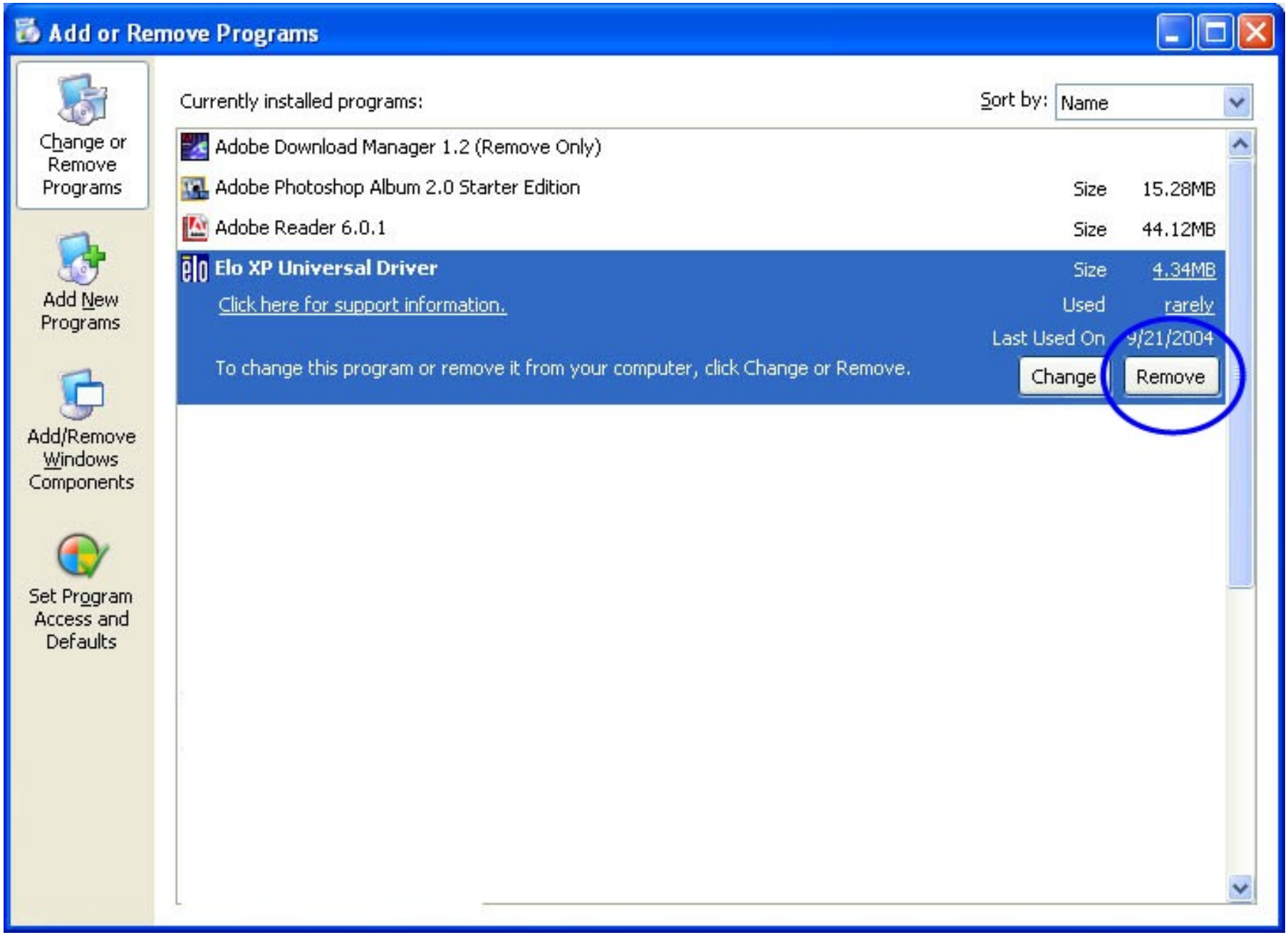
2. The Hardware Wizard dialog box may open. Follow the Wizard to complete the installation.

3. The driver files will install. When the **Setup Complete** screen appears, click on Finish. Video alignment is not required.

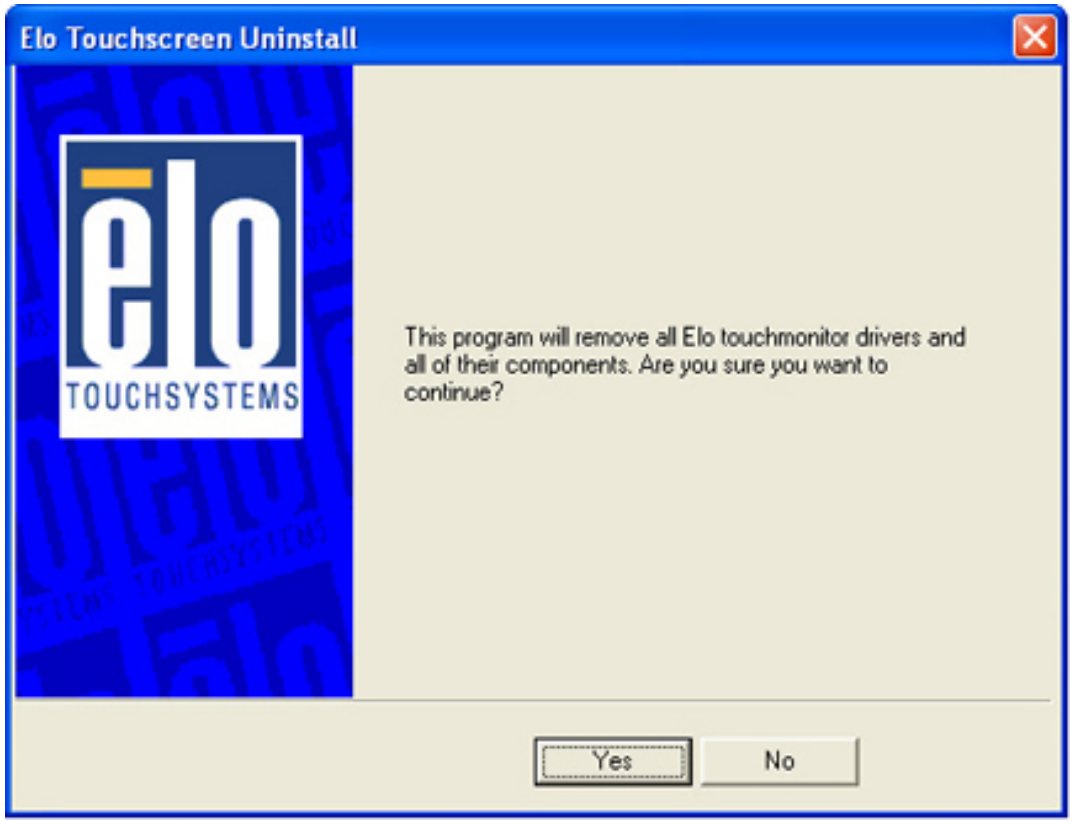
Home Page Top of page

Uninstalling the Driver

- Use the Add/Remove Programs feature in the Control Panel
  - Open the Control Panel
    - Click Start, hover over Settings, click Control Panel
  - Double-click the Add/Remove Programs or Programs and Features icon
    - On some machines there will be a moderate wait time for the program list to populate
  - Click the Elo Universal Driver item
  - Click the Change/Remove button or Uninstall button.



- Click the Yes button



- The Elo drivers will be removed. No computer restart is required.
- Click the Finish button when it appears.

- Run **EloSetup /U** from a command line or batch file. See Installing the Driver, Software, Silent Install for more details.

The following registry keys remain after the driver is uninstalled:

- The Secondary registry folders and keys (see Registry Entries section of this document)
- The USB portion of the Primary registry entries. See the Registry Entries section of this document for the location of the USB registry entries. Only the Vid_04e7&Pid_#### folder applies to Elo registry entries.

To facilitate reinstallation, the file folders remain intact after the driver is uninstalled:

\Program Files\Elo TouchSystems\Elo Universal Driver 4X

Disabling the Driver

Disabling Touch Functionality

Touch functionality may be quickly and easily disabled using the **Tool Tray** icon. If the Elo icon is not in the Tool Tray (lower right corner of the display), see the section on enabling the Tool Tray icon below.

- Click the **Elo icon** in the Tool Tray (lower right corner of the display)
  - The Elo Touchscreen Properties list is displayed
- Click **Disable Touch**
  - Touch functionality is disabled
- Touch functionality is re-enabled in the same manner
  - This is a toggle function (Enable/Disable)

To place the Elo icon in the Tool Tray:

- See Tool Tray section of this document.

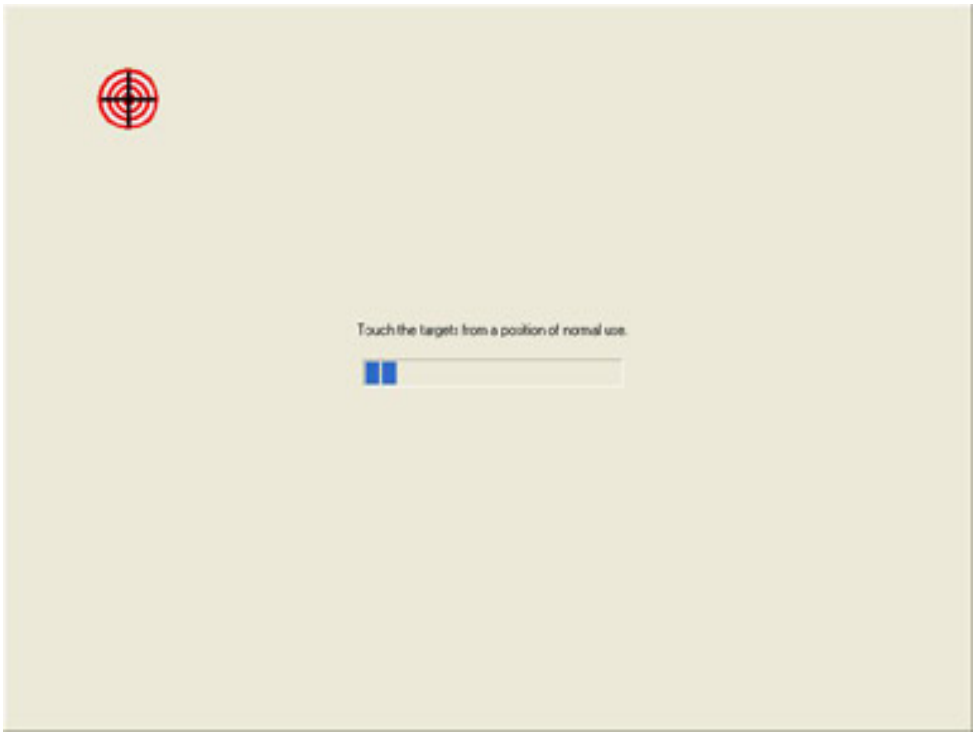
Home Page Top of page

Video Alignment (Calibration)

Video alignment or calibration as it is also called, ensures that the mouse cursor appears at the position of touch.

Elo uses a three-point calibration sequence that will accept touchscreens with any orientation of the X or Y axis, in landscape or portrait mode.

Once calibrated, the touchscreen will be ready to use automatically each time the system is restarted.



Options to Launch

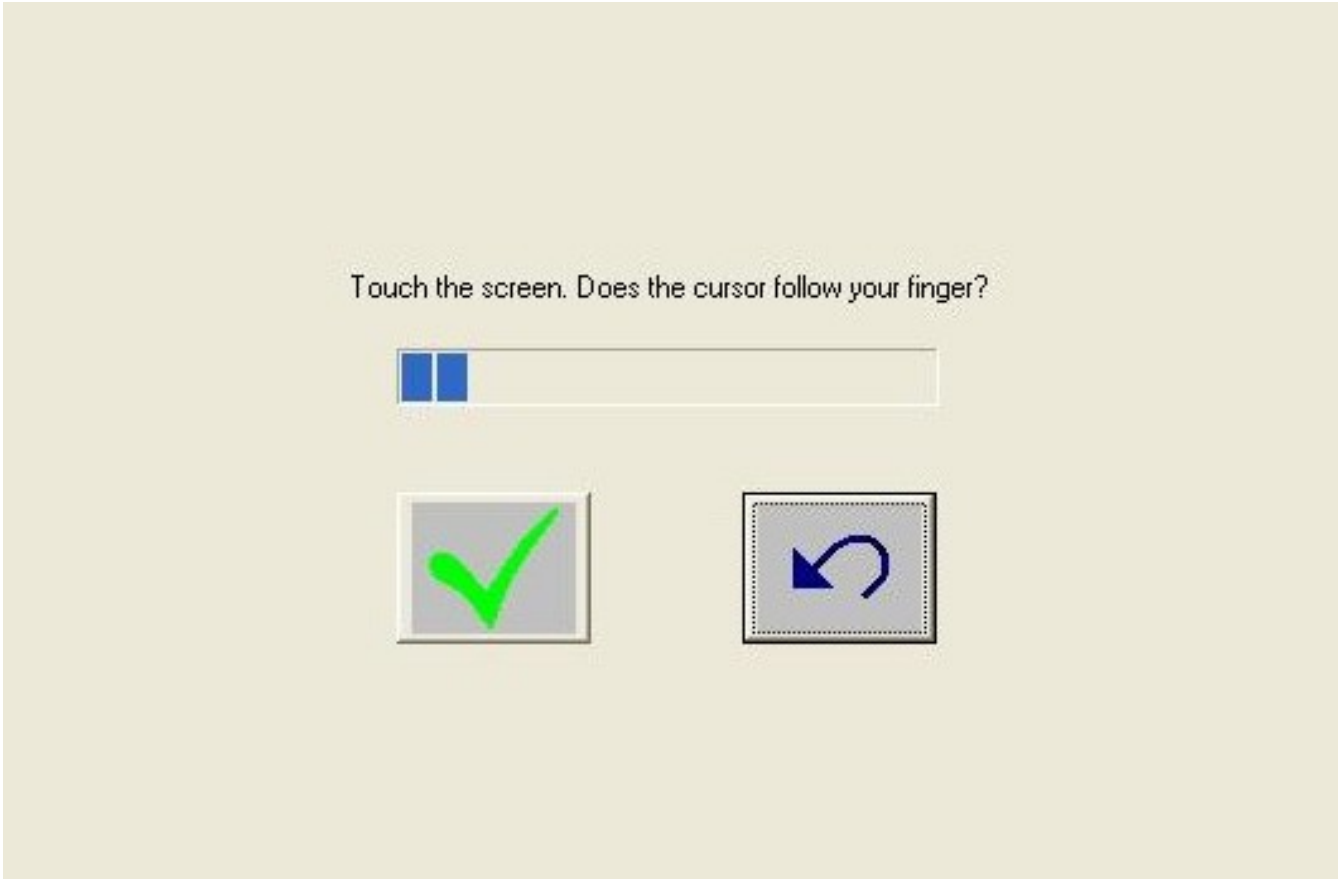
The video alignment program may be launched six different ways:

1. During setup a checkbox may be selected to run the alignment program automatically after setup is completed.
2. Via the Elo Control Panel application by selecting the General tab and clicking the alignment program icon.
3. Via the Elo Control Panel application by selecting the Properties tab for the individual monitor and clicking the alignment program icon.
4. By clicking the Elo icon in the window task bar at the bottom right of the display and selecting the video alignment option from the dialog list.
5. By double clicking the Elo icon in the Windows task bar at the bottom right of the display. This will bring up the Elo Control Panel application and the alignment program may be run via option two or three listed above.
6. By running EloVa.exe directly from the Windows command line or \Windows \System32 directory.
7. By calling EloVa.exe directly from an application.

Running the Alignment Program

When you run EloVA using one of the options above, touch the three targets as they appear.

You will then be asked to touch various points on the screen to verify that the cursor appears at the position touched. If the cursor appears at the position touched, click the green arrow. If the cursor does not appear at the position touched, click the blue curved arrow and the alignment program will run again.



If you are using multiple monitors the alignment program will run on each individual monitor. If one of the monitors is not a touchmonitor, Press the Esc key on the keyboard and the alignment program will advance to the next monitor, or wait until the program times out as indicated by completion of the progress bar.

Landscape/Portrait Mode

EloVA may be run in either landscape or portrait mode.

Unusual Combinations of Expanded Desktop with Multiple Monitors

Unusual multiple monitor combinations may create an expanded Windows desktop that is not recognized by the default video resolutions built into the EloVA program. For this reason, an auxiliary file, **EloVideo.txt**, is available to supplement the defaults. If you are running an expanded desktop multiple monitor resolution that is not multiples of standard monitor resolutions, you may experience difficulties in calibrating all of your monitors properly. Contact Elo Applications Engineering for availability and details of

the use of this file.

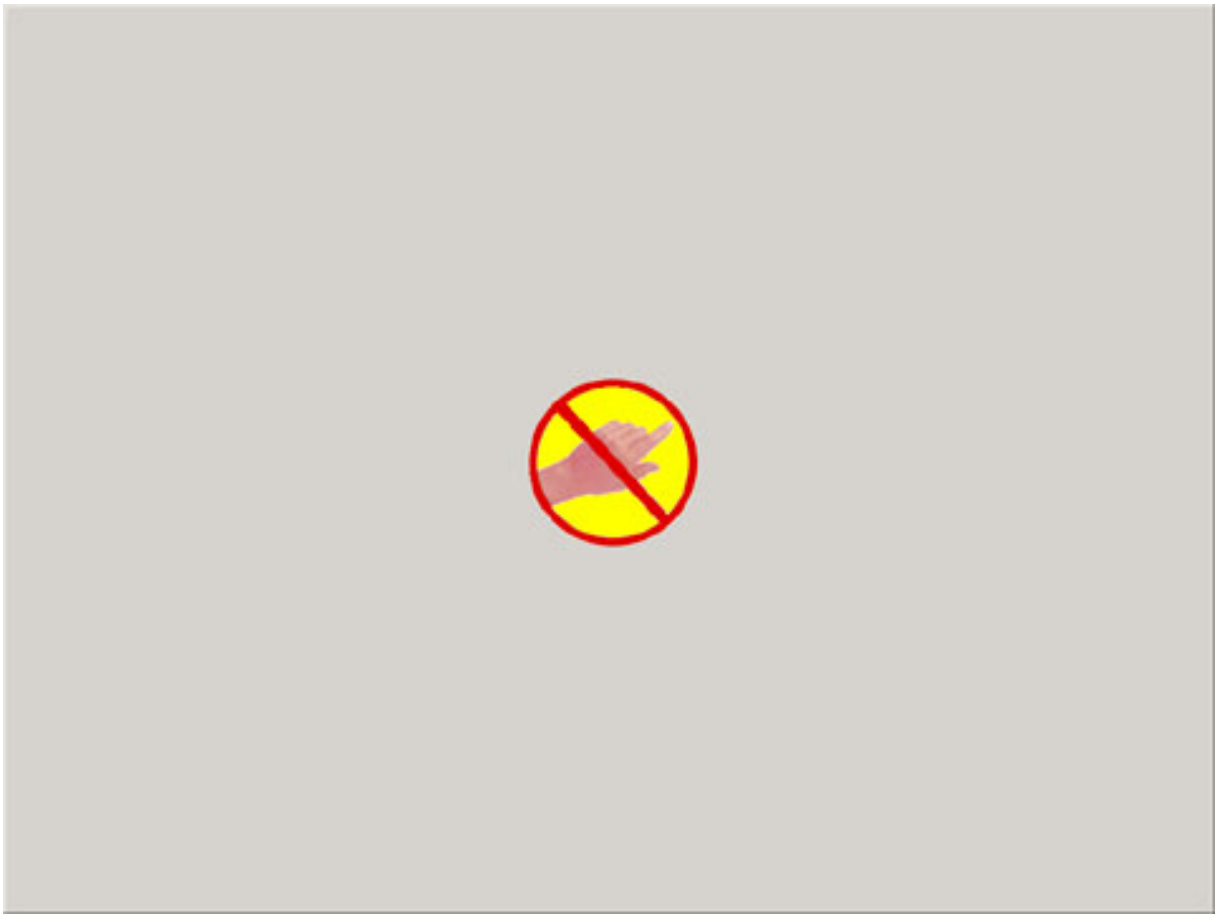
Target Locations on the Screen

When a touchscreen is mounted on to a monitor and the touch targets for calibration are not visible, target locations can be defined in an auxiliary file.

Target locations for every monitor can be defined in a configuration file, EloTarget.conf. Contact Elo Application engineering for availability and details of the use of this file.

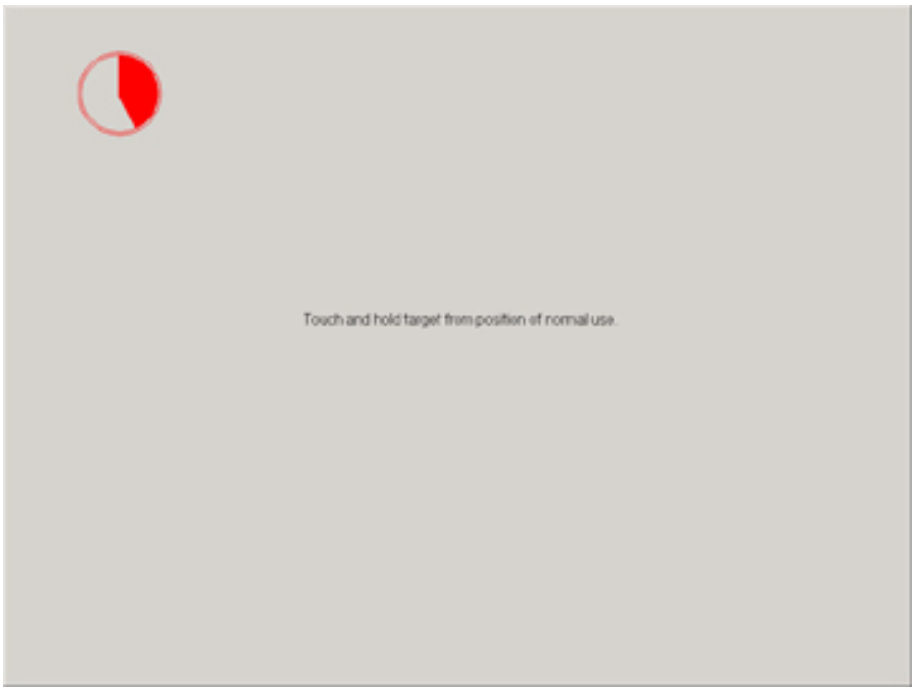
Video Alignment for 5000 or 5010 Capacitive Touchscreens

Video alignment for capacitive touchscreens follows a different procedure than that of the standard Elo video alignment. When running EloVa using one of the options above, the standard video alignment touch target window will appear on the screen. Once the first target is touched, the screen will change informing the user not to touch the screen.

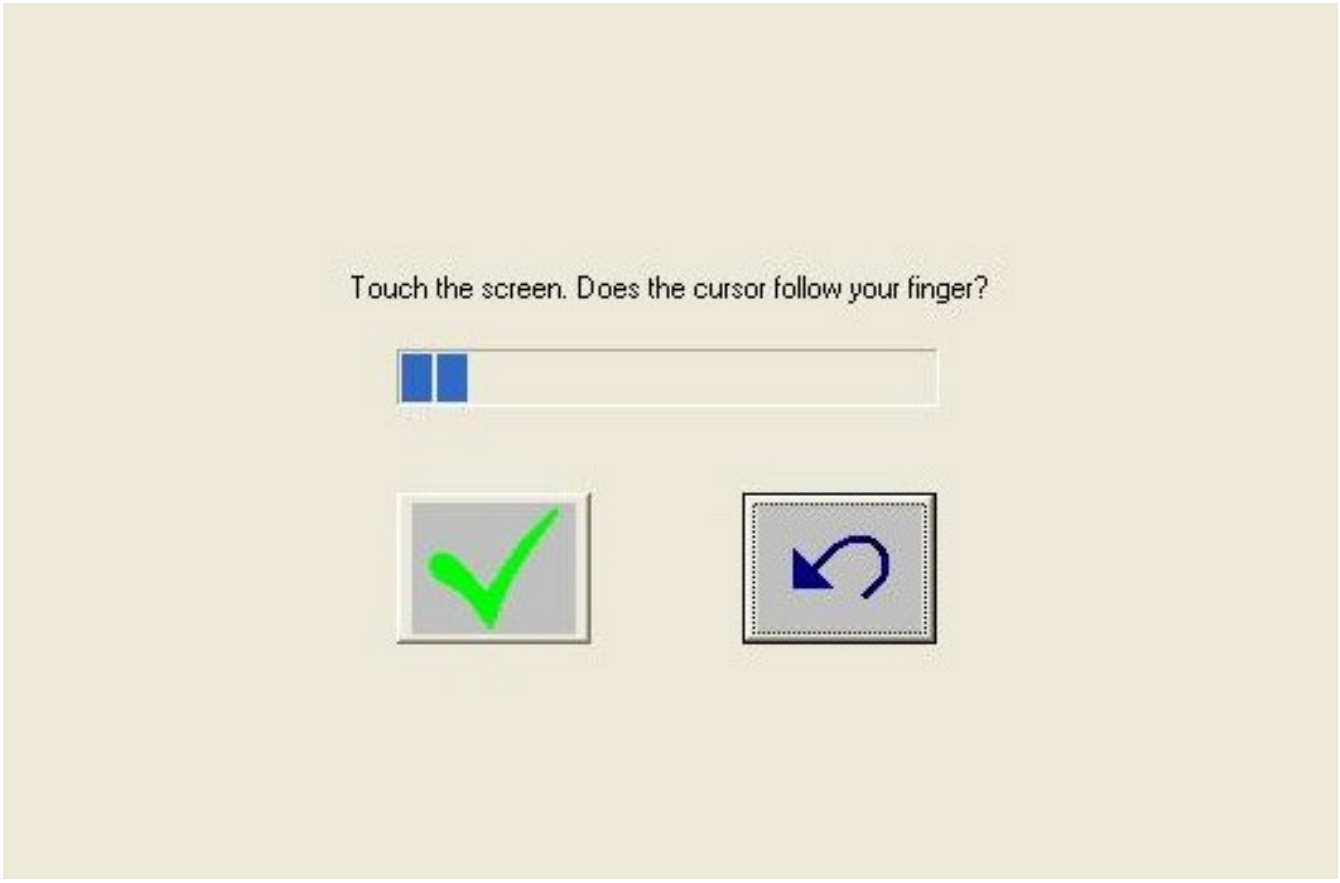


Once the initialization has been complete, the user should touch the screen. Data is now being collected. Continue to touch the screen until notified.

Follow the same procedure for all 3 targets as they appear.



Confirm the alignment by touching various locations on the screen and verifying that the cursor appears in the position touched. If the cursor appears at the position touched, click the green arrow. If the cursor does not appear at the position touched, click the blue arrow and the alignment program will run again.



If you are using multiple monitors the alignment program will run on each individual monitor. If one of the monitors is not a touchmonitor, Press the Esc key on the keyboard and the alignment program will advance to the next monitor, or wait until the program times out as indicated by completion of the progress bar.

Note: Acoustic Pulse Recognition (APR) is factory calibrated. Video alignment or calibration not required.

Home Page Top of page

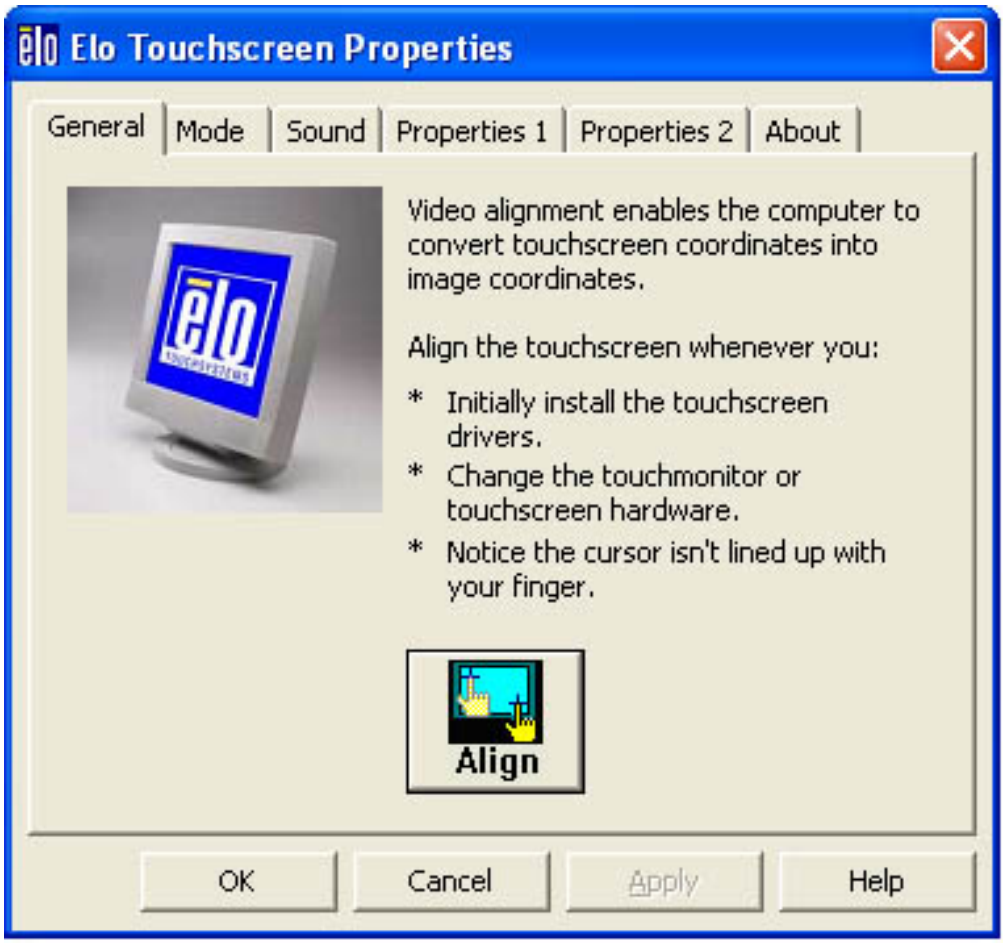
Control Panel (Elo Mouse Properties)

The Control Panel (CP) application allows configuration of the driver to suit application programs and presents system and diagnostic information to the user.

Each of the five or more tabs in the control panel are described below.

General

- The General tab is displayed when the Control Panel is opened. The other tabs described below may be selected and the video alignment program EloVA may be launched from this tab.

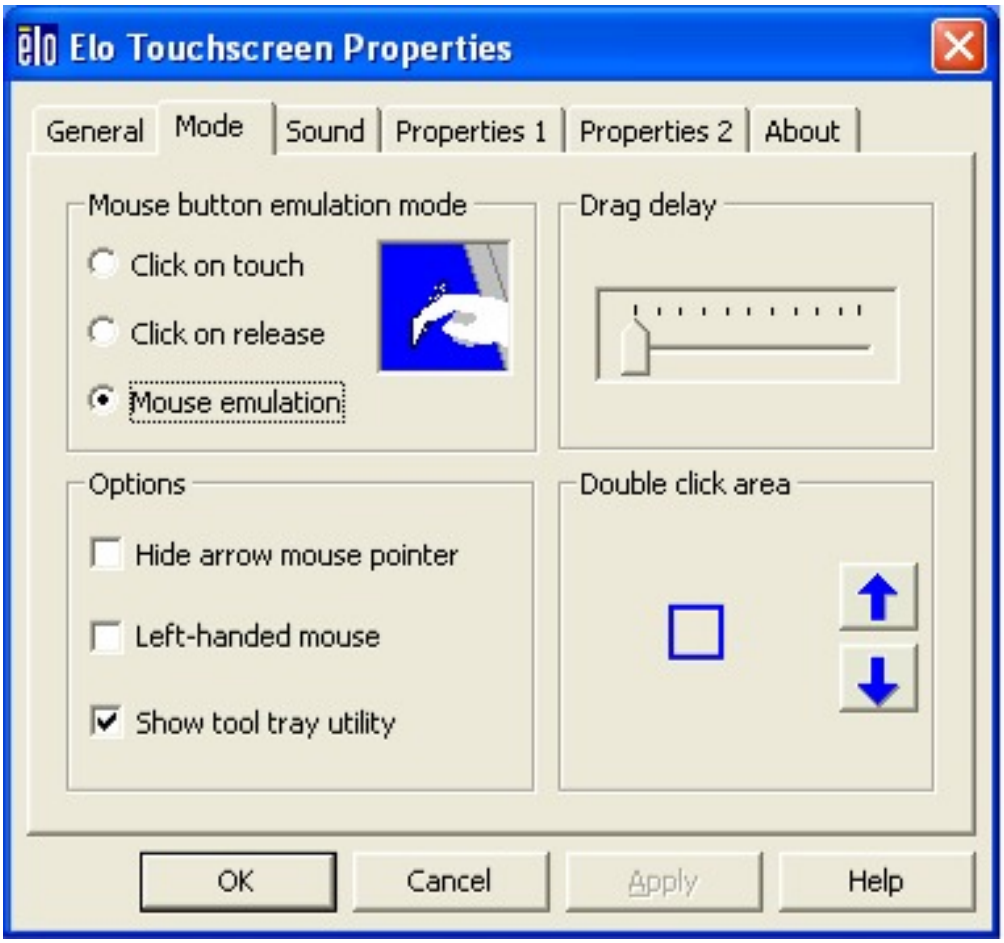


Mode

- The Mode tab selects touchscreen operating mode and configures the appearance and function ability of the desktop. All selections made in the Mode tab must be activated by clicking the "Apply" button at the bottom of the dialog.

Mouse Button Emulation Mode

This mode is selected by clicking the appropriate radio button.



Click on Touch

- Click on touch sends immediately upon touch a mouse down/up message at the point of touch on the touchscreen. The user's finger must be removed from the touchscreen before a new touch at any location will be recognized. The cursor or selected objects CANNOT be "dragged" on the screen in this mode.

Click on Release

- Click on release sends at the time of release (untouch) a mouse down/up message at the point that the screen was last touched. Dragging across objects on the screen will not highlight or select them unless untouch occurs when the touch is over the object.

Mouse Emulation (Drag and Double-click)

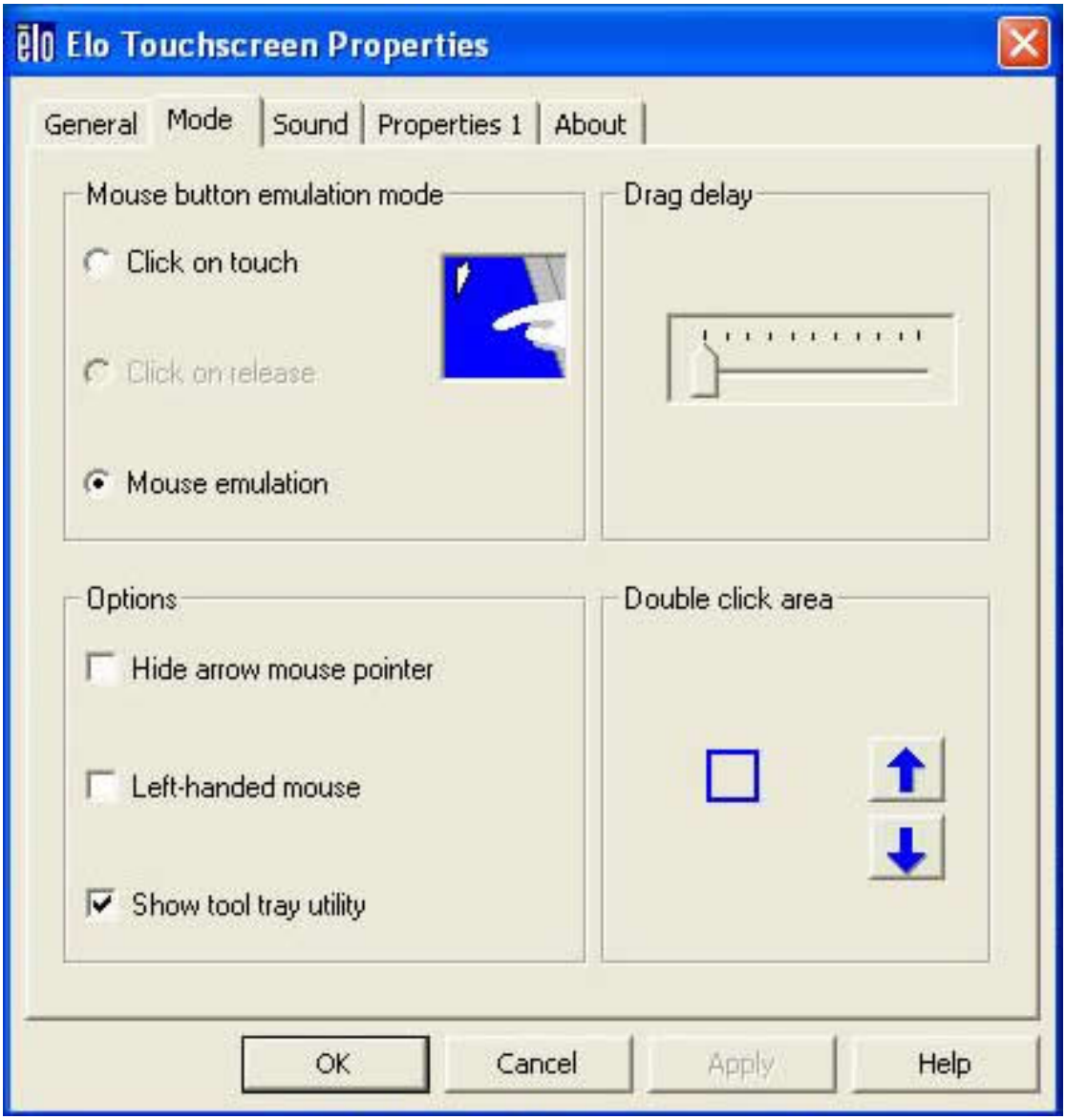
- Sends a mouse down message at the point of contact.
- Selects an object if it was at the initial point of contact.
- Drags a selected object on the screen.
- Response to dragging is set by the Drag delay slider bar.

- Sends a mouse up message at the point of untouch.
- Double-clicks on an object when the screen is touched twice in rapid succession at the same location.
- The speed to achieve double-click is identical to the speed of a successful double-click with the mouse.
- Double-click speed is set from the Mouse Properties control panel (Mouse Properties>Buttons>Double-click speed).
- Double click area graphically sets the dimensions of the location around each clickable icon or object on the screen which will be recognized by Windows as a double-click. The size of the wire-frame square displayed in the Double click area tab is the actual size of double-click area accepted by Windows. The square is increased or decreased in size by touching the appropriate arrows adjacent to the square. Note that the double-click box size is independent of screen resolution and must be defined for each user.

Options

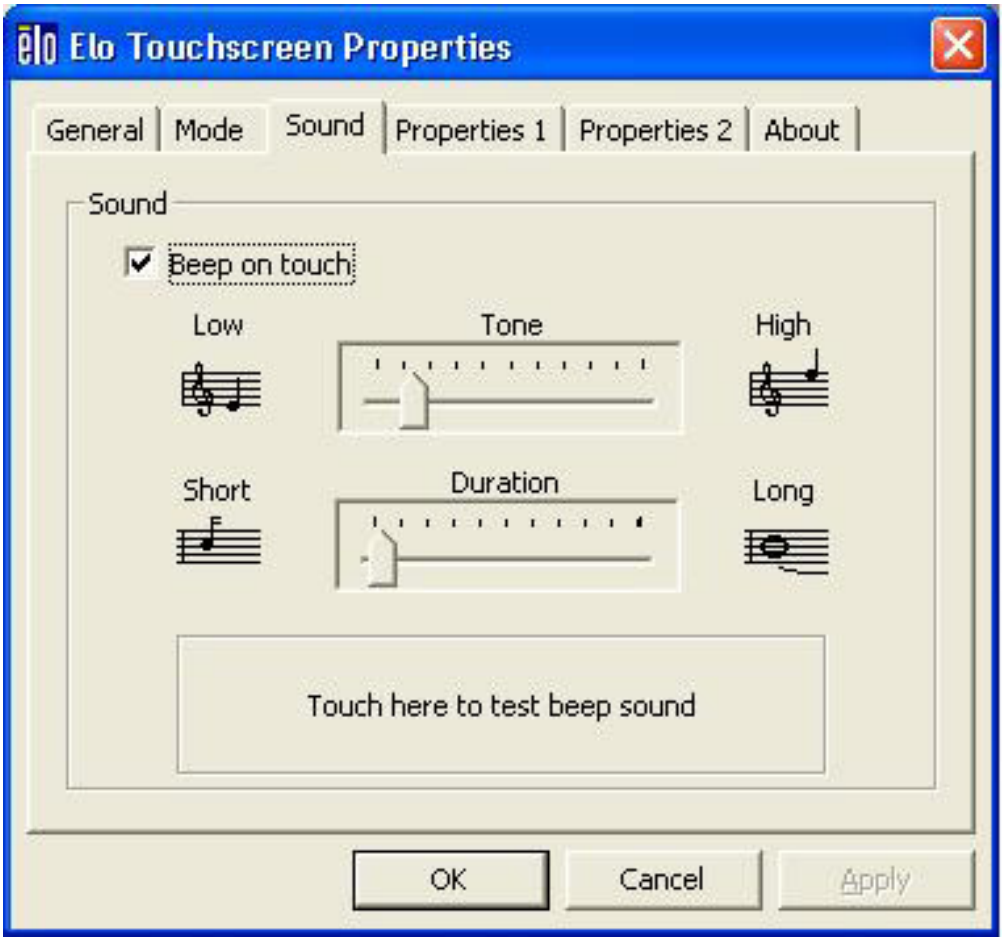
- Options allow various features of the desktop related to the touchscreen to be configured. Each of the features selected must be activated by clicking the "Apply" button at the bottom of the tab.
  - **Hide arrow mouse pointer** turns off the standard mouse cursor.
  - **Left-handed mouse** interchanges the standard two-button mouse button assignments.
  - **Show tool tray utility** activates the Tool Tray utility in the Windows Task Bar. See Tool Tray for a complete description of this feature.

Mouse Button Emulation Mode (APR)



The Mode options are the same for APR as noted above, with the exception of **Click on Release**. This option is not avaliable.

Sound



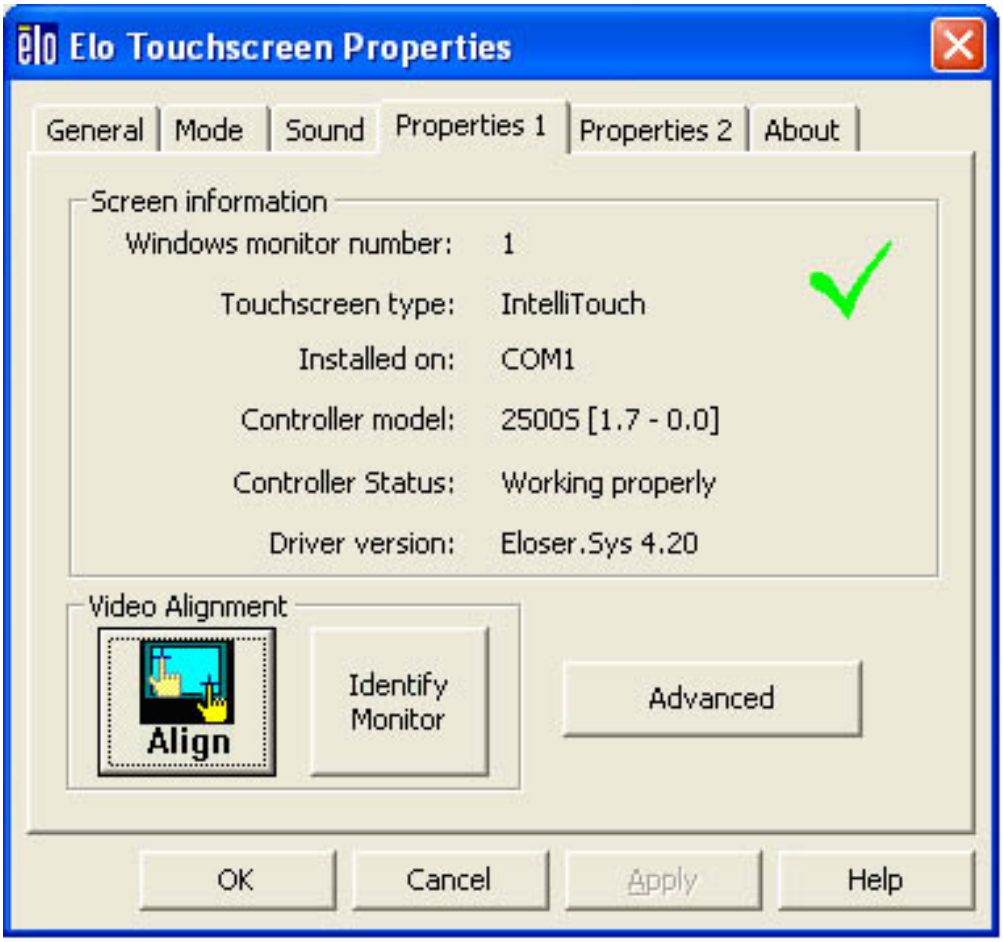
- The Sound tab sends a single-frequency tone or "Beep" to the system speaker each time that a valid touch occurs.
- The beep is enabled by default when the driver is installed. It may be turned off by unchecking the Beep on touch box in this tab.
- The frequency (Tone) and the Duration of the beep can be adjusted by moving the appropriate slider in this tab with the touchscreen or mouse, or by using the keyboard arrow keys.
- Selections or changes to the settings in this tab take effect when the "Apply" button is clicked.

Properties

A Properties page will be created for each touchscreen controller installed by EloSetup and for each serial port reserved for a controller, even if that controller is not present. A number will be assigned to each Properties page that is related to the order in which controllers or ports are detected or enumerated. Each Properties page contains information extracted from the touchscreen controller and the system about the monitor, touchscreen, controller and internal driver.

Screen Information

Properties 1



Windows Monitor Number

- Windows monitor number lists the monitor that corresponds to the "Windows Monitor" from the Windows Display Properties control panel.

Touchscreen Type

- Touchscreen Type lists the Elo touchscreen used by its marketing name, i.e., AccuTouch for Elo's five-wire resistive touchscreen

Installed On

- Installed On indicates USB or lists the Windows COM port that a serial touchscreen is installed on.

Controller Model

- Controller Model lists the marketing model number of the controller attached. It also lists the revision of the firmware loaded in the controller.

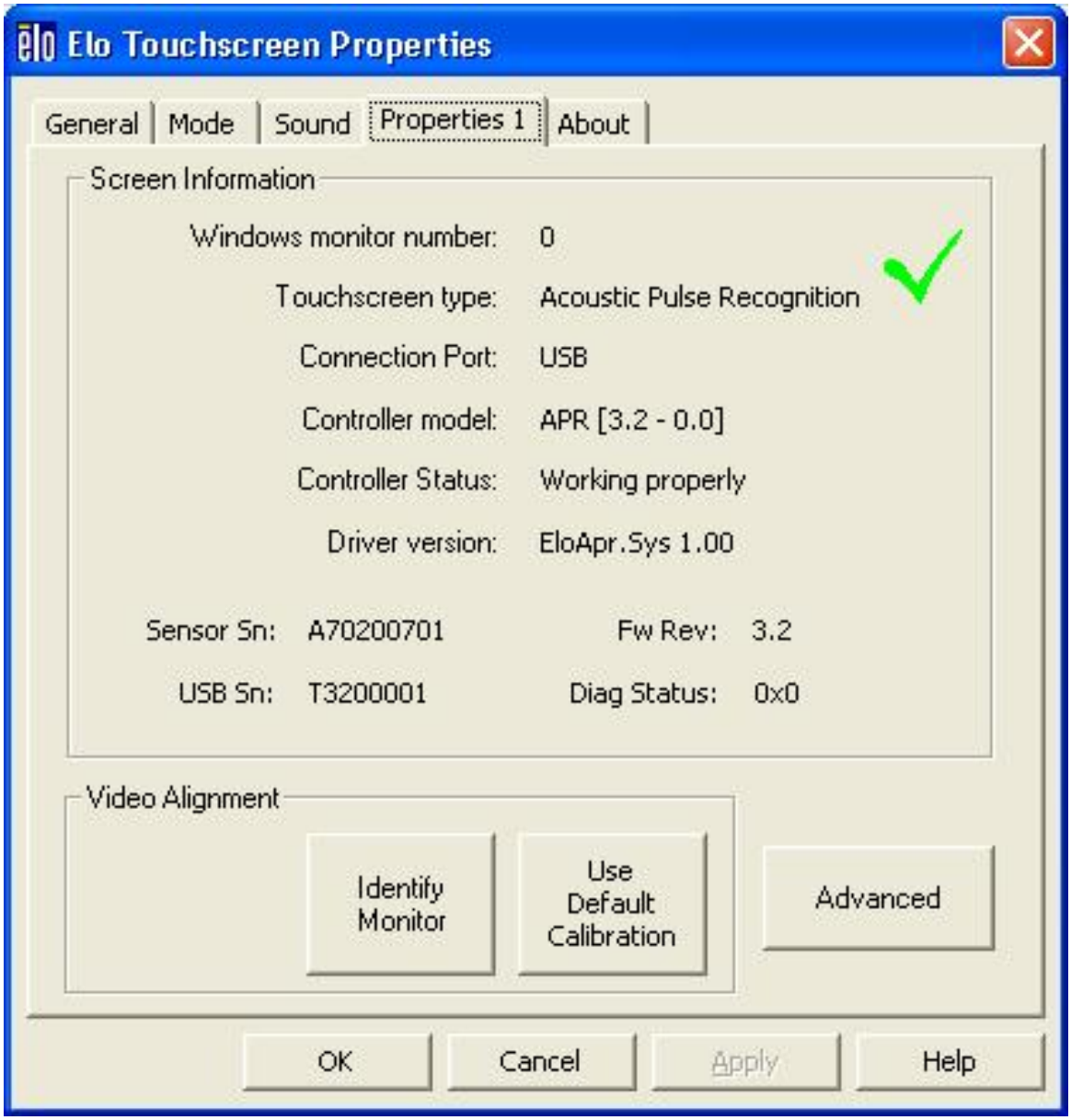
Controller Status

- Controller Status indicates either "Working properly" or reflects any errors reported from the controller.

Driver Version

- Driver Version lists the version number of the internal driver being used for serial and USB devices as appropriate.

The APR Property is the same as other touchscreens with the exception of the Video Alignment and Advanced and Use Default Calibration buttons. The **Use Default Calibration** button allows you to reset the calibration back to the original driver default mode.



The **Video Alignment** and **Advanced** sections appear for touchscreen properties only when the touchscreen is calibrated.

Each Properties page also contains a **Video Alignment** section that has two functions:

- The **Identify Monitor** button will display the Elo logo on the monitor associated with this Properties page for about one second each time this button is touched.

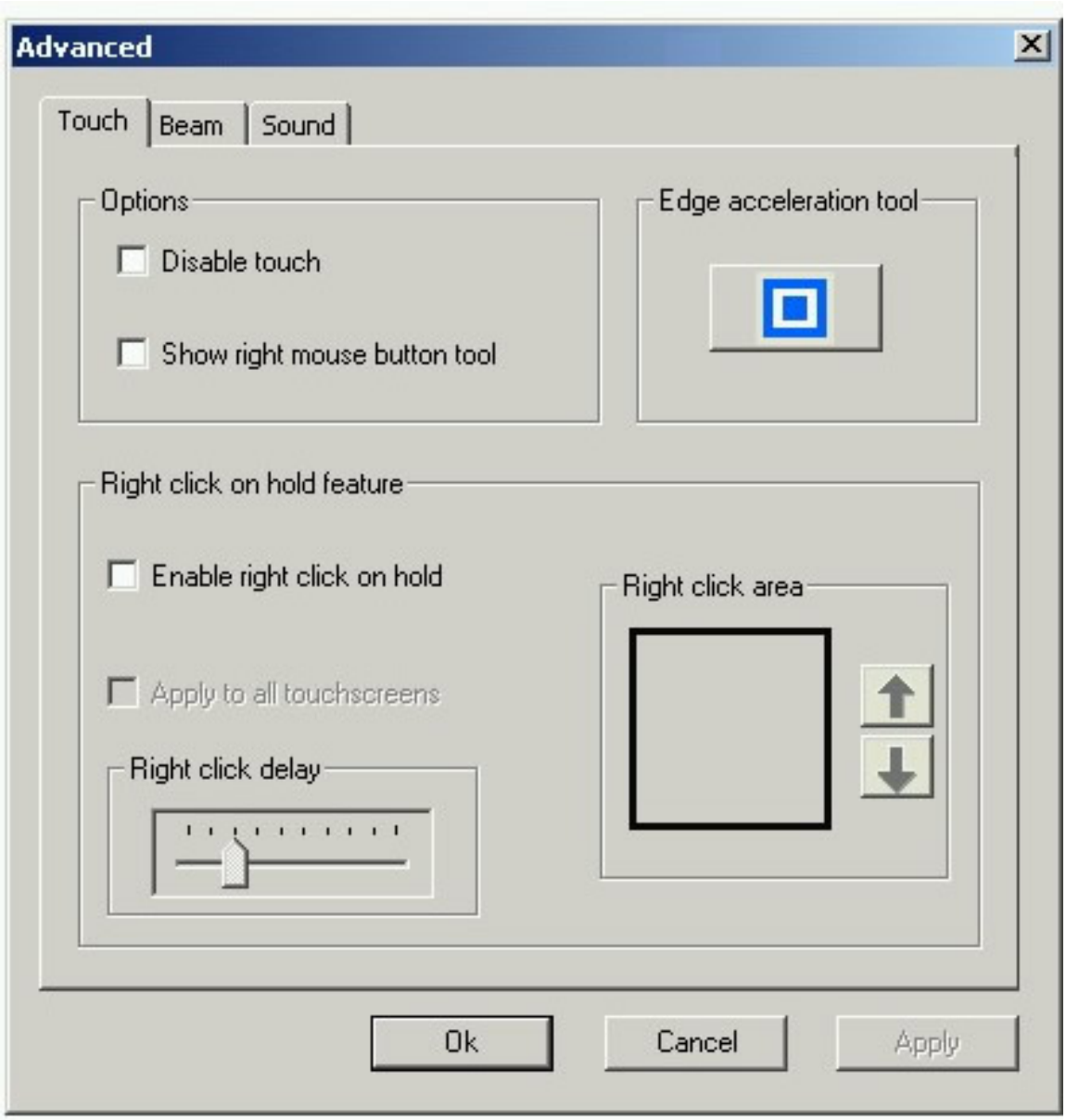
- The **EloVA icon**  will launch the video alignment program **only for the monitor associated with the current Properties page**. Running EloVA from the Properties page avoids the need to align all monitors in a multiple monitor application when only a single monitor alignment is desired.

If a serial port is reserved for a touchscreen controller that is not actually installed, the only information contained in the Screen information section of the associated

Properties page will be the number of the Windows COM port. No Screen configuration icons will be displayed.

Each Properties page also contains an **Advanced** section.

Advanced Touch Tab



Advanced page contains an Option and Right click on hold section.
Option section has:

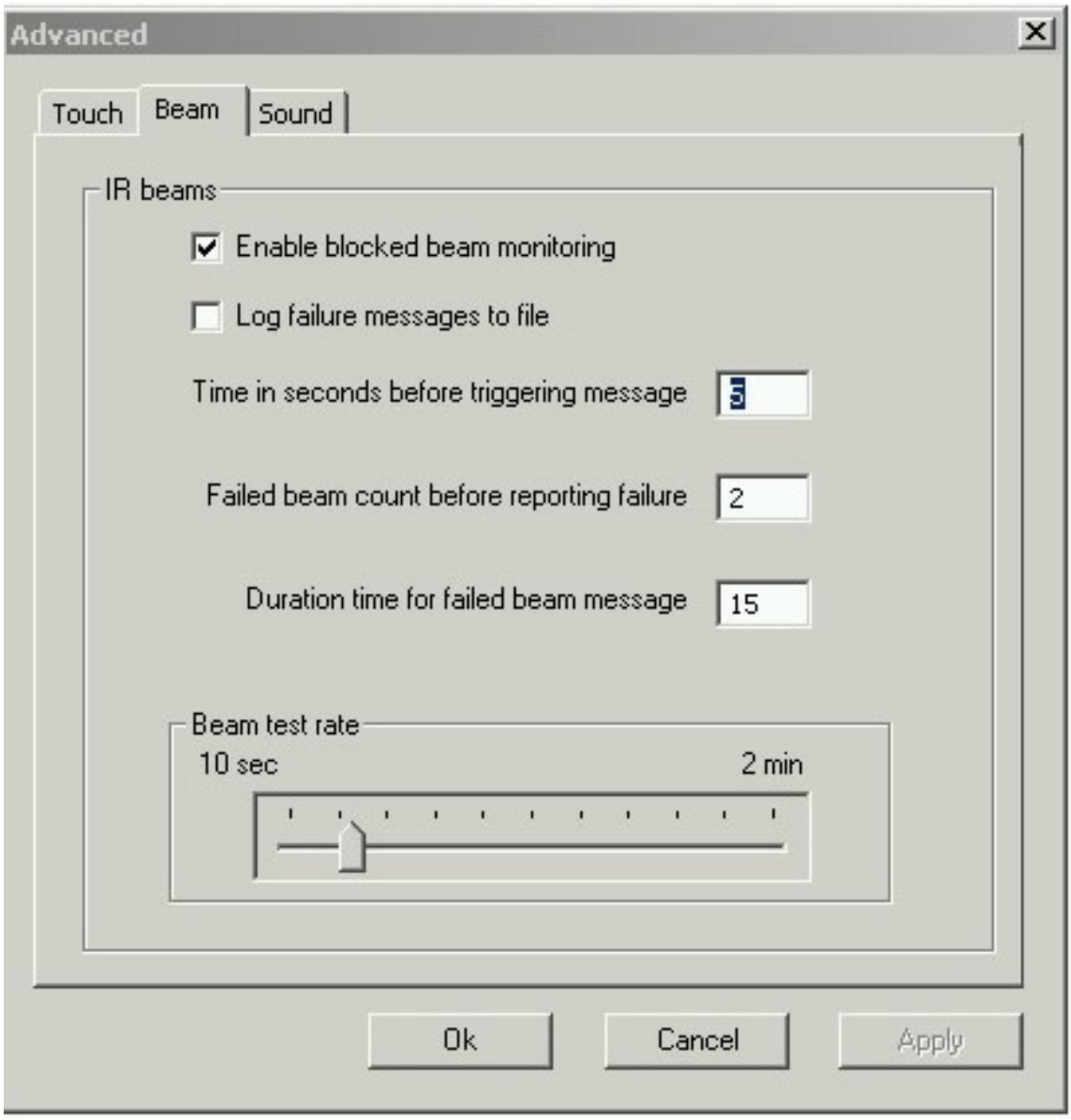
- **Show right mouse button tool** launches this feature for this monitor. See Right Mouse Button for a complete description of this feature.

- **Disable Touch** stops touch information from reaching Windows for this touchscreen. This feature may also be activated from the **Tool Tray** utility. Disabling touch has no effect on the operation of the standard mouse.

- **Edge Acceleration Tool** launches this utility for this monitor. See Edge Acceleration Tool for a complete description of this feature.

Advanced Beam Tab

This tab only appears for Carroll Touch screens (4000 and 4500)
It continuously monitors the health of the IR beams and reports any problems.



- **Enable blocked beam monitoring.** Enables or disables beam health monitoring.

- **Log failure message to file**. Check this box to write any future beam failure messages to a file named: "C:\EloDiags.log"

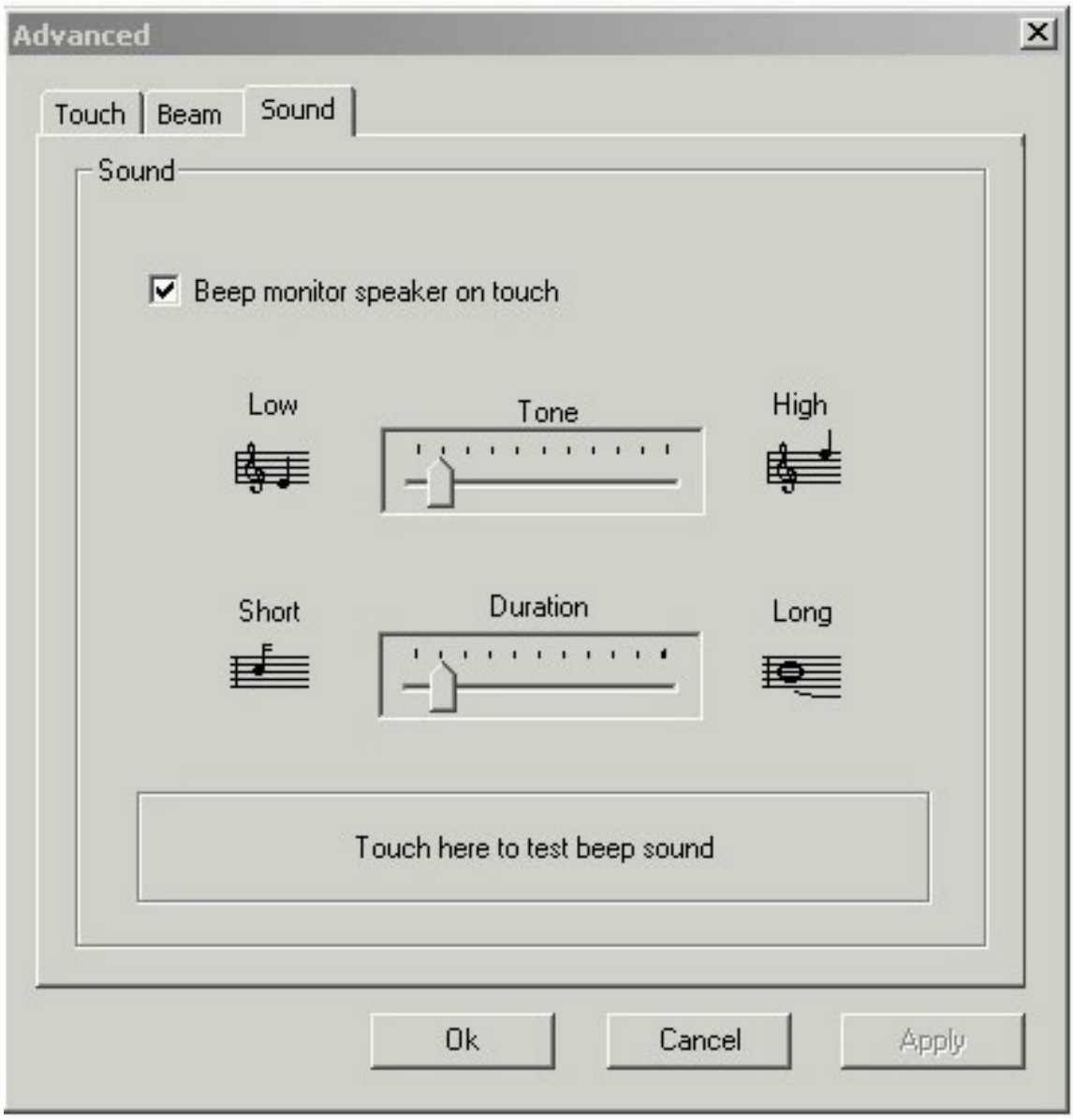
- **Time in seconds before triggering message.** Number of seconds before displaying or logging an error message after a beam failure is detected. This eliminates false alarms, for example, if a user places their hand on the screen

for an amount time.

- **Failed beam count before reporting failure.** Enter the minimum number of failed beams before a failure message is generated.

- **Duration time for failed beam message**. Enter the number of seconds that the failure message will be displayed on the screen. (Duration of the message). The message will repeat again if problem is not corrected.

Advanced Sound Tab



This tab only appears for those Carroll Touch screens that have a separate build in beeper. This feature can be used to provide audio feedback of touches in cases where the screen is placed at some distance from the host computer. Use this tab to configure the sound characteristics of the beeper similar to the "Sound" tab on the main page.

- **Beep monitor speaker on touch.** Enables/disables the internal beeper.

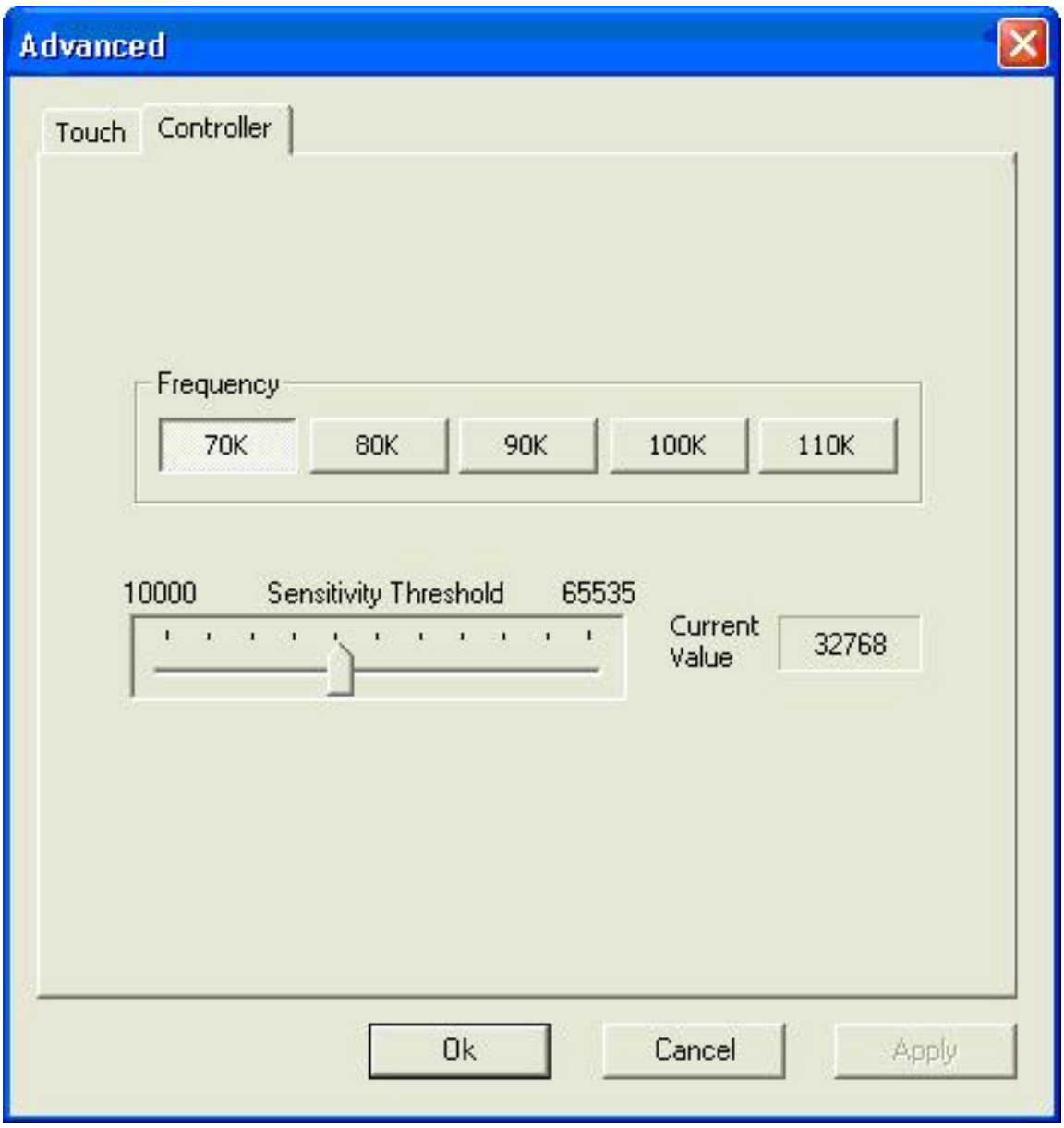
- **Tone.** Adjusts the tone of the touch beep.

- **Duration.** Sets the duration of the touch beep.

- **Touch here to test beep sound area.** Tests the current settings without changing focus away from the control panel. Click the **Apply** button in order for any changes to take effect.

Advanced Controller Tab

This tab only appears for Capacitive Touch screens (5020)



- **Frequency Threshold.** The frequency adjustment sets the touch screen driving frequency of the controller. Adjustment of this parameter may be needed when some of the external similar frequencies to the current drive frequency are being used nearby.

- **Sensitivity.** The sensitivity adjusts how difficult or easy it is to activate the touch screen; the lower the value is the more sensitive the sensor/controller will be. The allowable range is 10,000 to 65535 and the default is 32,768. The software displays the

current sensitivity setting.

Right Click on Hold

Right Click on Hold is a new feature added in version 4.20. This feature allows new and efficient ways of creating right clicks on the touch screen without needing to run the Right Mouse Button Tool. If the user touches the screen and continues to touch, a right click is generated on the screen if the user remains within the Right click area and after the Right click delay time-out has occured. The right click delay allows the user to perform other touchscreen operations normally. When the user touches the screen and holds his finger down, an animated clock is painted at the location and continues to tick and generate a right click at the location after it completion.

- Enable right click on hold: If checked, it enables this feature for this screen, otherwise, it is disabled.

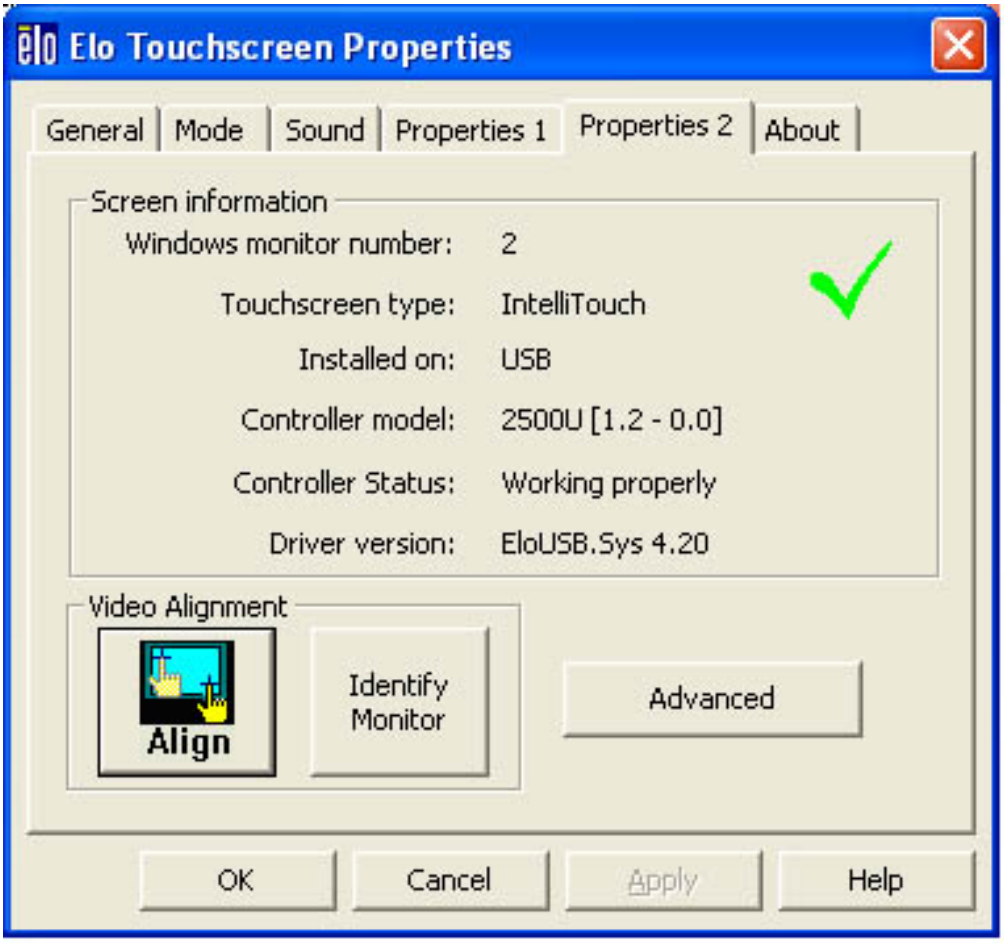
- Apply to all touchscreens: If checked, the current values are set to all touchscreens when "Apply" or "Ok" button is clicked.

- Right click delay: Defines the right click delay value in milliseconds. The right click delay allows the users to perform other touchscreen operations normally.
- Right Click Area: Touching within this area generates a right click. This area allows the user the flexibility to touch within a larger area

NOTE: Right click on hold feature is enabled only in "Mouse emulation" touch mode.
When Right click on hold feature is turned on, Drag Delay value is set to a minimum of 25 milliseconds.

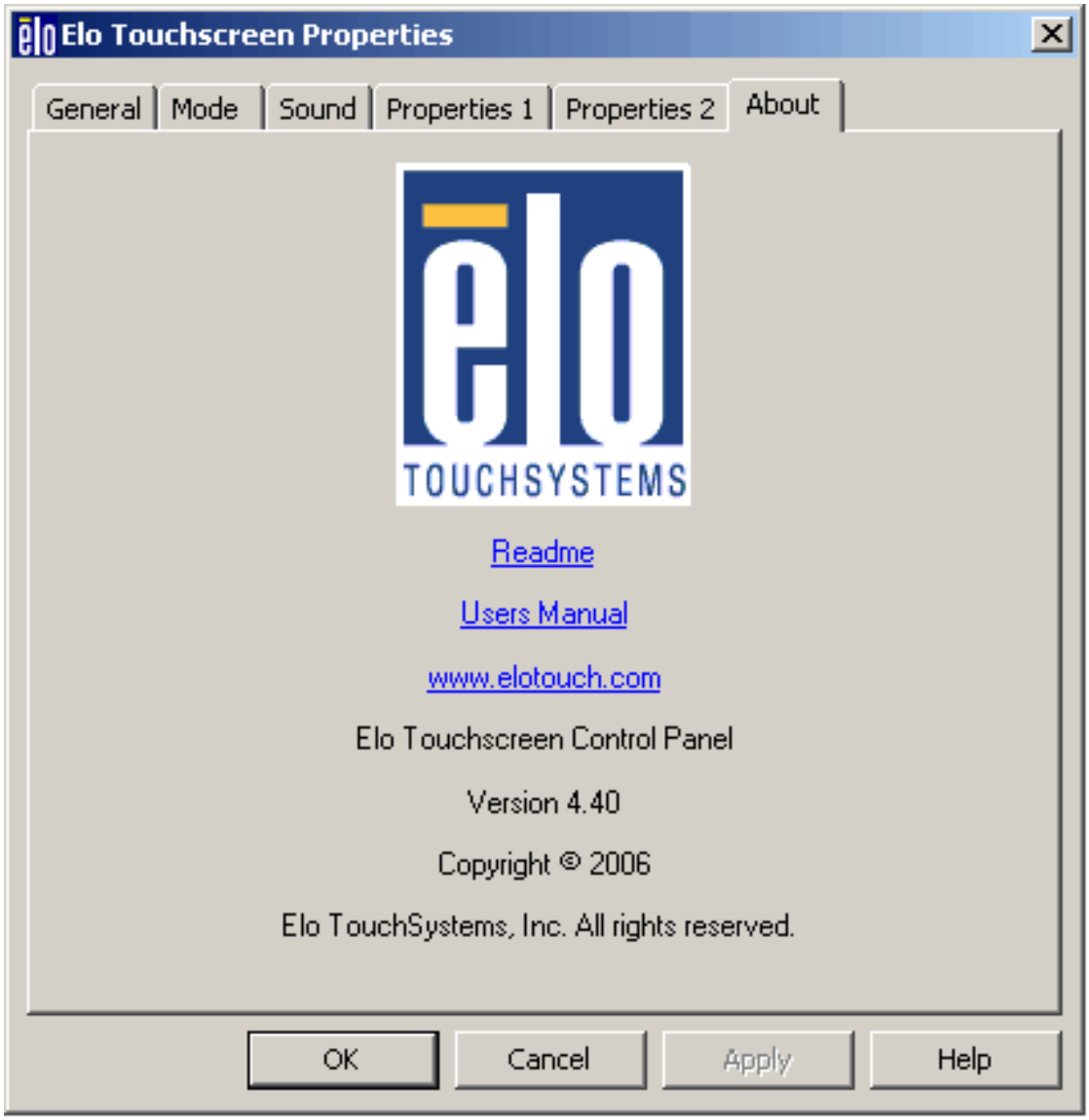
Right Click On Hold function is not supported for APR.

Properties 2 Tab



About

The About tab provides the version of the Control Panel and also provides links to:



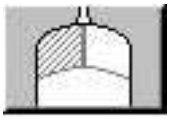
- Readme document included with the driver.
- Users manual
- Elo's web site

Home Page Top of page

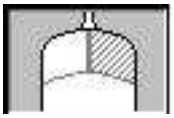
Elo Right Mouse Button Tool (RMBT)

Right Mouse Button tool allows a Windows right mouse button simulation on the touchscreen.

A representation of a typical two-button mouse is displayed in a small window on the desktop, when this application is run. The initial presentation of the RMBT shows the left button shaded, indicating that the left button is active. Any touch on the desktop or an application will produce a left button click consistent with the Button settings in the Control Panel.



A touch in the RMBT will change the shading to show the right mouse button active. Now any touch on this monitor on which RMBT is running will produce a right button click.



Typically, a right button action will activate a new dialog box for some function similar to mouse right button click. When that dialog box is touched, or any other touch on the touch monitor is made, this touch will be a left button event. Simultaneously, the mouse button shading in the RMBT will toggle back to the left button.

After the RMBT has been touched to toggle to the right button state, a second touch in the RMBT will activate a menu allowing the user to:

- Launch the Elo Control Panel (Elo Touchscreen Properties)
- Close the right button utility
- A right button click on the RMBT with the mouse will also activate this menu

The RMBT may be dragged to any location on the desktop by touching it and holding the touch momentarily until the crossed arrows appear. The RMBT may also be dragged with the standard mouse.

The RMBT cannot be resized.

The RMBT can be run for every touchmonitor from the **Elo Control Pane**l > **Properties** tab.

Only one RMBT can be active for each touchmonitor.

Home Page

Tool Tray

The Tool Tray application offers a convenient way to access commonly used functions of the driver from the operating system desktop.

To display the Tool Tray icon in the taskbar, it must be enabled from the Control Panel in the Mode tab. To remove the icon it must be disabled from the Control Panel.

The Tool Tray application is launched with a single left or right click and pops up a menu with several selectable functions.

The Tool Tray functions are:

Elo Touchscreen Properties

- A one click method to launch the Elo Control Panel.

Align

- Launches EloVA, the Elo Video Alignment program (Not applicavble for Acoustic Pulse Recognition (APR)).

Center Desktop Tool

- Launches Center Desktop Tool

Elo Right Mouse Button Tool (RMBT)

- Launches Right Mouse Button Tool for all touch monitors that have been calibrated.

Edge Acceleration Tool

- Launches Edge Acceleration tool.

Disable/Enable Touch

- When the driver is loaded and running, clicking this segment of the Tool Tray menu will disable the driver, stopping all touch reports from being sent from the driver to the mouse handler (MouClass)
- When disabled through the Tool Tray, touches made on the touchscreen are not buffered by the driver, and will not be sent from the driver to MouClass after the touchscreen is re-enabled.
- After the touchscreen has been disabled, the state of this segment of the Tool Tray menu changes from "Disable" to "Enable." Since the touchscreen is now disabled, it can only be enabled by using the mouse or keyboard. Clicking "Enable" in the Tool Tray will now restore touchscreen operation.
- It is also possible to re-enable the touchscreen from the Control Panel after touch has been disabled from the Tool Tray. Open the Control Panel with mouse or keyboard, uncheck the "Disable Touch" box in the Mode>Options tab, click "Apply" and touch is restored.

Readme

- Launches Readme instructions.

User Manual

- Quick link to this User Manual.

http://www.elotouch.com

- A quick link to Elo's website

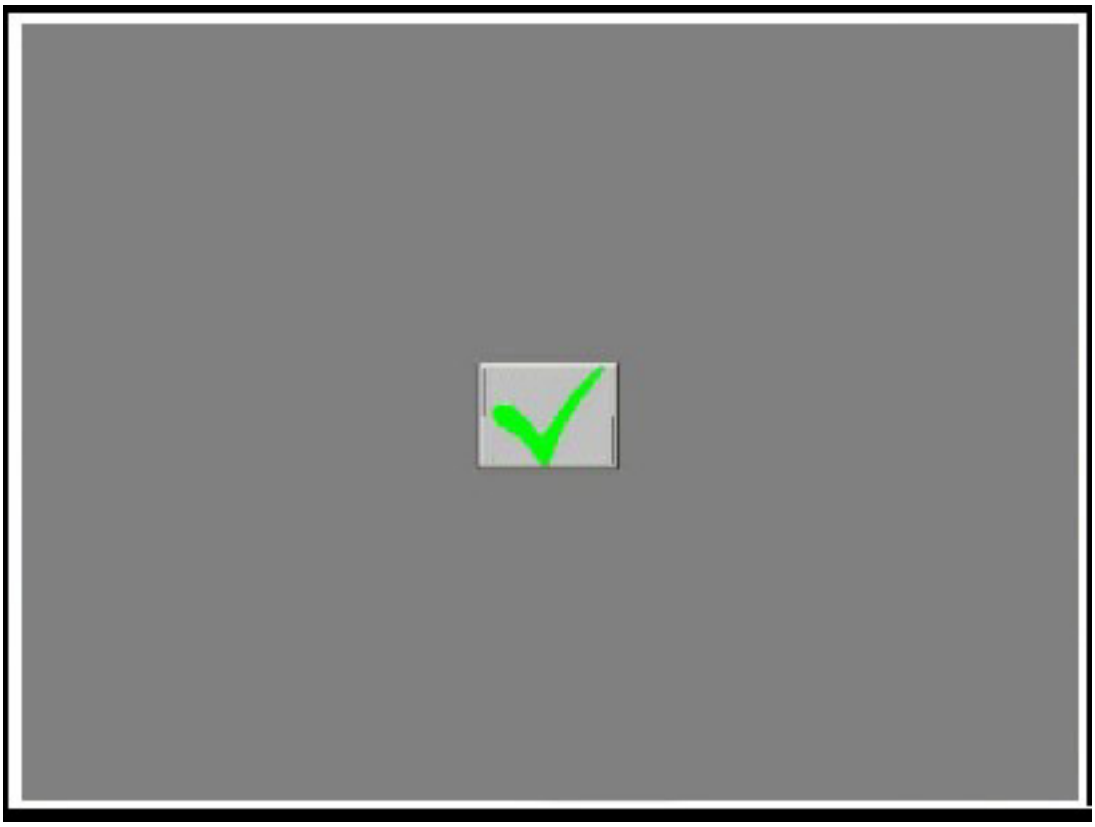
Exit

- Clicking this button exits the Tool Tray application and also removes it from the Windows Taskbar.
- The Tool Tray application may also be removed by unchecking it in the Control Panel and clicking "Apply."

Home Page Top of page

Center Desktop Tool

- Presents a full screen image for the selected monitor to allow proper adjustment of the video prior to running the EloVA video alignment program.
- Configurable parameters are
    - Border width, in pixels
    - Which monitor to adjust in a multiple monitor configuration
- Can be launched from the Tool Tray (for single monitor) or command line (for multiple monitors).
- Configuration syntax menu is available by running eloalmon /h. Only one monitor may be adjusted for each execution of the program.
- Program terminates when green "check" mark is touched or clicked with the mouse, or from the keyboard by pressing Esc, Enter or Space. See graphic below.



- The center desktop tool solves a common problem in video alignment programs-how to properly size the video image. The image presented is a black screen with a white border, which makes the edge of the image visible and allows it to be properly adjusted on CRT displays as well as flat panel displays of all technologies.

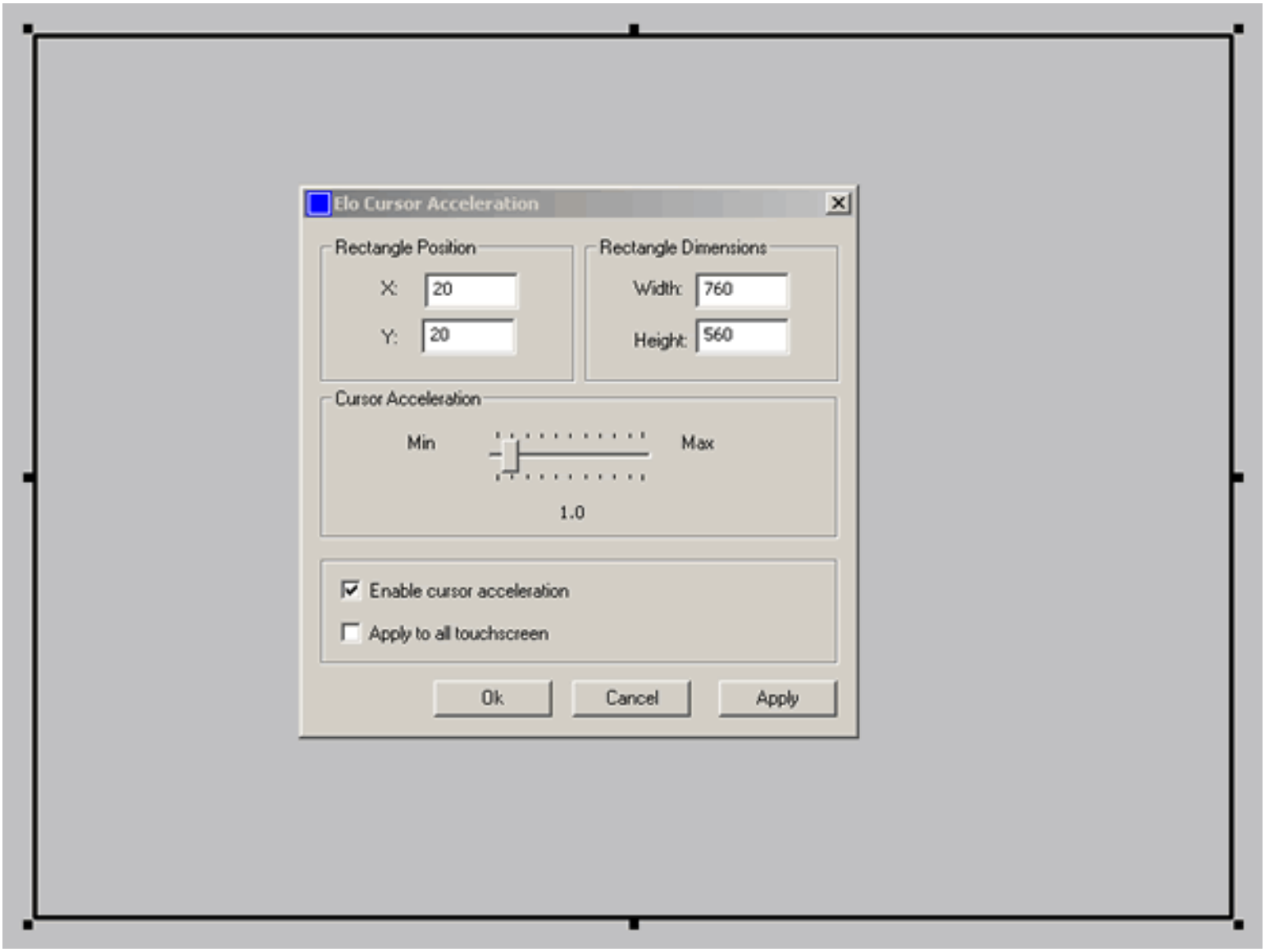
Home Page

Edge Acceleration Tool (EAT)

Edge Acceleration tool can be used to configure cursor acceleration towards the edge of the touchscreen. Edge Acceleration is a special feature provided to allow touches towards the edges.

EAT (i.e. EloAccel.exe) can be launched from **Program Files**> **EloTouchsystems folder**.

It presents a full screen image with the edge acceleration rectangle bounds and a **Cursor Acceleration Configuration** dialog, displaying the configuration parameters for Edge Acceleration.



Configurable parameters are:

- Cursor Acceleration scale. This scale can be set from a value of 0-10 where, 1 corresponds to no acceleration.
- Edge acceleration rectangle bounds dimensions.
- Edge acceleration rectangle bounds position
- Enable Edge acceleration

Cursor is accelerated beyond the rectangle bounds.

These parameters can be configured using the mouse, to resize or move the bounds rectangle or, by entering the values directly in the dialog box. Changes to the settings in the Cursor Acceleration dialog take effect when the **Apply** button is clicked.

This applies to **all touchmonitors** unless run with appropriate command line options. To view the options available, run EloAccel using **EloAccel /h**.

Home Page

File List

The following list identifies all the files in the basic driver package and the installed location of each file. Most of the files are common to both the serial and USB drivers. Those files that are associated with only one of the serial or USB drivers are identified as such.

XPU Files	Installed Location
Common Files	
EloAIMon.exe	\Windows\System32
EloDkMon.exe	\Windows\System32
EloIntf.dll	\Windows\System32
EloPubIf.dll	\Windows\System32
EloLCoin.dll	\Windows\System32
EloLnchr.exe	\Windows\System32
EloProp.dll	\Windows\System32
EloRtBtn.exe	\Windows\System32
EloSerCo.dll	\Windows\System32
EloSetup.exe	\Program Files\EloTouchSystems
EloStrings_ENG.dll	\Windows\System32
EloStrings_FRE.dll	\Windows\System32
EloStrings_SPA.dll	\Windows\System32
EloStrings_ITA.dll	\Windows\System32
EloStrings_CHN.dll	\Windows\System32
EloStrings_MDN.dll	\Windows\System32
EloStrings_JAP.dll	\Windows\System32
BeamMon.dll	\Windows\System32
EloSrvce.exe	\Windows\System32
EloSrvCt.exe	\Windows\System32
EloTouch.cpl	\Windows\System32
EloTTray.exe	\Windows\System32
EloUCoin.dll	\Windows\System32
EloVA.exe	\Windows\System32
EloInterface.h	\Program Files\EloTouchSystems
EloErrorCodes.h	\Program Files\EloTouchSystems
Null.cur	\Windows\cursors
Readme.txt	\Program Files\EloTouchSystems
Readme-S.txt	\Program Files\EloTouchSystems
Readme-U.txt	\Program Files\EloTouchSystems
XPU Driver Manual.htm	

XPU Driver Manual Folder

Serial

EloBus.sys	\Windows\System32\Drivers
EloSer.sys	\Windows\System32\Drivers
WinSerXP.inf	\Program Files\EloTouchSystems
WinSerXP.cat	\Program Files\EloTouchSystems

USB

EloFiltr.cat	\Program Files\EloTouchSystems
EloFiltr.inf	\Program Files\EloTouchSystems
EloFiltr.sys	\Windows\System32\Drivers
EloUSB.sys	\Windows\System32\Drivers
Elouxpsi.cat	\Program Files\EloTouchSystems
EloUXpSI.inf	\Program Files\EloTouchSystems

APR Files	**Installed Location**

Common Files

EloAIMon.exe	\Windows\System32
EloDkMon.exe	\Windows\System32
EloIntf.dll	\Windows\System32
EloPubIf.dll	\Windows\System32
ELoAprCo.dll	\Windows\System32
EloLnchr.exe	\Windows\System32
EloProp.dll	\Windows\System32
EloRtBtn.exe	\Windows\System32
EloSetup.exe	\Program Files\EloTouchSystems
EloStrings_ENG.dll	\Windows\System32
EloStrings_FRE.dll	\Windows\System32
EloStrings_SPA.dll	\Windows\System32
EloStrings_ITA.dll	\Windows\System32
EloStrings_CHN.dll	\Windows\System32
EloStrings_MDN.dll	\Windows\System32
EloStrings_JAP.dll	\Windows\System32
BeamMon.dll	\Windows\System32

EloSrvce.exe	\Windows\System32
EloSrvCt.exe	\Windows\System32
EloTouch.cpl	\Windows\System32
EloTTray.exe	\Windows\System32
EloVA.exe	\Windows\System32
EloInterface.h	\Program Files\EloTouchSystems
EloErrorCodes.h	\Program Files\EloTouchSystems
FlashMon.exe	\Windows\System32
Lut.bin	\Program Files\EloTouchSystems
Null.cur	\Windows\Cursors
Readme.txt	\Program Files\EloTouchSystems
Readme-S.txt	\Program Files\EloTouchSystems
Readme-U.txt	\Program Files\EloTouchSystems
XPU Driver Manual.htm	
XPU Driver Manual Folder	

USB

EloApr.cat	\Program Files\EloTouchSystems
EloApr.inf	\Program Files\EloTouchSystems
EloApr.sys	\Windows\System32\Drivers

Home Page Top of page

Registry Entries

Registry (Primary registry entries):

Serial

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\SERENUM\ELOSERIAL]

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\SERENUM\ELOSERIAL
\5&1fc2a90a&0&0000]
"Capabilities"=dword
"HardwareID"=hex(7)
"ClassGUID"="{4D36E96F-E325-11CE-BFC1-08002BE10318}"
"Service"="EloSer"
"ConfigFlags"=dword
"Driver"="{4D36E96F-E325-11CE-BFC1-08002BE10318}\\0003"
"Class"="Mouse"
"Mfg"="Elo TouchSystems, Inc."
"DeviceDesc"="Elo Serial Touchmonitor Interface"

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\SERENUM\ELOSERIAL
\5&1fc2a90a&0&0000\Device Parameters]
"PortFriendlyName"="Port on Digi ClassicBoard PCI (COM3)"
"Migrated"=dword
"HardwareHandshaking"=dword
"PacketEnable"=dword
"BeepFlag"=dword
"BeepTime"=dword
"BeepFreq"=dword
"DefaultDragDelay"=dword
"MinDragDelay"=dword
"MaxDragDelay"=dword
"MouseMode"=dword
"LeftHandedUser"=dword
"RightButtonActive"=dword
"EnableQuickTouch"=dword
"QuickTouchDx"=dword
"QuickTouchDy"=dword
"DebugLevel"=dword
"RightClickInitialTimeout"=dword
"RightClickHW"=dword
"DefaultRightClickDelay"=dword
"MaxRightClickDelay"=dword
"MinRightClickDelay"=dword
"RightButtonOnholdCount"=dword
"RightButtonOnholdActive"=dword

"UntouchTimeOut"=dword
"UntouchHeight"=dword
"UntouchWidth"=dword
"FullScreenVmode"=dword
"FullScreenBounds"=hex
"X_EloDx"=dword
"X_ScrDx"=dword
"X_Offset"=dword
"Y_EloDx"=dword
"Y_ScrDx"=dword
"Y_Offset"=dword
"Z_EloDx"=dword
"Z_ScrDx"=dword
"Z_Offset"=dword
"SwapXY"=dword
"WindowsMonitorNumber"=dword
"X_Resolution"=dword
"Y_Resolution"=dword
"xVirtScrSize"=dword
"yVirtScrSize"=dword
"xVirtScrCoord"=dword
"yVirtScrCoord"=dword
"xMonLocn"=dword
"yMonLocn"=dword
"VirtualDeskMode"=dword
"ExclusionFlag"=dword
"Num_Bounds"=dword
"ClippingBounds"=hex
"EdgeAccelerationEnable"=dword
"EdgeAccelerationScale"=dword
"EdgeAccelerationBounds"=hex

USB

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\USB\Vid_04e7&Pid_0007]

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\USB\Vid_04e7&Pid_0007\07U00635]
"DeviceDesc"="Elo TouchSystems 2500U IntelliTouch® USB Touchmonitor Interface"
"LocationInformation"="Elo TouchSystems IntelliTouch 2500U"
"Capabilities"=dword:00000094
"HardwareID"=hex(7)
"CompatibleIDs"=hex(7)
"ClassGUID"="{745A17A0-74D3-11D0-B6FE-00A0C90F57DA}"
"Class"="HIDClass"
"Driver"="{745A17A0-74D3-11D0-B6FE-00A0C90F57DA}\\0000"
"Mfg"="Elo TouchSystems, Inc."
"Service"="HidUsb"
"ConfigFlags"=dword
"ParentIdPrefix"="7&2c8bde9&3b"

"LowerFilters"=hex(7)

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\USB\Vid_04e7&Pid_0007\07U00635
\Device Parameters]
"SymbolicName"="\\??\\USB#Vid_04e7&Pid_0007#07U00635#{a5dcbf10-6530-11d2-901f-
00c04fb951ed}"
"PacketEnable"=dword
"EnableQuickTouch"=dword
"QuickTouchDx"=dword
"QuickTouchDy"=dword
"DebugLevel"=dword
"FullScreenVmode"=dword
"FullScreenBounds"=hex
"X_EloDx"=dword
"X_ScrDx"=dword
"X_Offset"=dword
"Y_EloDx"=dword
"Y_ScrDx"=dword
"Y_Offset"=dword
"Z_EloDx"=dword
"Z_ScrDx"=dword
"Z_Offset"=dword
"WindowsMonitorNumber"=dword
"SwapXY"=dword
"X_Resolution"=dword
"Y_Resolution"=dword
"xVirtScrSize"=dword
"yVirtScrSize"=dword
"xVirtScrCoord"=dword
"yVirtScrCoord"=dword
"xMonLocn"=dword
"yMonLocn"=dword
"Num_Bounds"=dword
"ExclusionFlag"=dword
"VDeskMode"=dword
"ClippingBounds"=hex
"EdgeAccelerationEnable"=dword
"EdgeAccelerationScale"=dword
"EdgeAccelerationBounds"=hex

HID

Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\HID\Vid_04e7&Pid_0007]

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\HID\Vid_04e7&Pid_0007
\7&2c8bde9&3b&0000]
"Capabilities"=dword:000000a0

"HardwareID"=hex(7)
"CompatibleIDs"=hex(7)
"ClassGUID"="{4D36E96F-E325-11CE-BFC1-08002BE10318}"
"Service"="mouhid"
"ConfigFlags"=dword
"Driver"="{4D36E96F-E325-11CE-BFC1-08002BE10318}\\0001"
"Class"="Mouse"
"Mfg"="ELO Touchsystems"
"DeviceDesc"="EloHID_PID_0007"
"UpperFilters"=hex(7)

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\HID\Vid_04e7&Pid_0007
\7&2c8bde9&3b&0000\Device Parameters]
"Migrated"=dword
"FlipFlopWheel"=dword
"ForceAbsolute"=dword
"BeepFlag"=dword
"BeepTime"=dword
"BeepFreq"=dword
"DefaultDragDelay"=dword
"MinDragDelay"=dword
"MaxDragDelay"=dword
"MouseMode"=dword
"LeftHandedUser"=dword
"RightButtonActive"=dword
"RightClickInitialTimeout"=dword
"RightClickHW"=dword
"DefaultRightClickDelay"=dword
"MaxRightClickDelay"=dword
"MinRightClickDelay"=dword
"RightButtonOnholdCount"=dword
"RightButtonOnholdActive"=dword
"EnableQuickTouch"=dword
"QuickTouchDx"=dword
"QuickTouchDy"=dword
"UntouchTimeOut"=dword
"UntouchHeight"=dword
"UntouchWidth"=dword
"DebugLevel"=dword

Secondary Registry Entries:

HKEY_LOCAL_MACHINE
System
CurrentControlSet
Services
elobus
Default
DisplayName
ErrorControl
ImagePath
Start
Type

Primary Registry Entries:

Enum
Default
0
Count
NextInstance
Security
Default
Security
EloSer
Same keys as elobus, above
Enum
Same keys as elobus Enum, above
Security
Same keys as elobus Security, above
EloSystemService
Same keys as elobus, above
Enum
Same keys as elobus Enum,above
Security
Same keys as elobus Security, above

Home Page Top of page

Troubleshooting

This driver only supports serial or USB touchscreen controllers.

If you have a serial controller and are unsure about whether or not the controller or touchscreen is functioning correctly, use Elo's serial port utility, COMDUMP.EXE. It is available from Elo's web site or any Elo TouchTools CD. The syntax for COMDUMP is:

COMDUMP [COM port]

This program must be run from a command line. Select a serial port that is NOT being used by the driver, or uninstall the driver first.

As an alternative, you may be able to see the status light on the serial controller, and determine whether or not the touchscreen and controller are operating properly from the status light function. If the light is visible, perform the following steps:

- Disconnect the serial cable from the computer.
- Observe the status light, a small green or yellow LED on the controller.
    - If the light blinks approximately once per second with no touch applied, touch the touchscreen and observe the light again while your finger is still touching the screen.
        - If the status light illuminates continuously while the screen is touched, stop touching.
        - If the light reverts to the once per second blink when no longer touched, the touchscreen and controller are probably functioning properly.
    - If the status light blinks approximately 2-3 times per second with no touch, cycle power on the display or controller first.
        - If cycling power causes the status light to revert to blinking once per second, follow the steps in the first case to evaluate performance.
        - If cycling the power produces no change in the blink rate, there is a hardware fault in the controller or touchscreen that may interfere with the operation of the system.
        - Touch the touchscreen and observe the light again while your finger is still touching the screen.
        - If the status light illuminates continuously while the screen is touched, stop touching.
        - If the light reverts to the 2-3 times per second blink when no longer touched, the touchscreen or controller may have a fault, but are probably functioning well enough to work with the driver.

If you are working with a USB controller, the best hardware troubleshooting method is to observe cursor motion on the touchscreen display with only the Windows native HID drivers installed. Even if the cursor does not move in the same direction that your finger does, when you drag your finger on the touchscreen, cursor motion itself is sufficient to tell you that your hardware is working correctly.

There are status lights for USB controllers as well:

- Connect a USB cable to the system
- A touch status light blinks once per second rate with no touch applied. Operation is similar to that of the serial controller status lights described above.
- A second status light (located very close to the USB connector) is continuously illuminated except when touch is applied. With touch, this light blinks at 2-3 times per second. This USB status light is only monitoring the status of the USB bus, and is not concerned with the interaction between the touchscreen and controller.

If your hardware is functioning correctly, and you believe that your problem is driver software related, start by:

- Uninstalling the driver
- If you are unsure as to the success of the uninstall process, check the file lists and the discussion in the Uninstalling the Driver section of this manual. You can successfully remove the files manually if necessary.
- Reinstall the driver.
- If problems persist, contact Elo Applications Engineering at one of the locations listed in Contacting Elo.

Home Page

Appendix A

Device Specific Registry Keys

Introduction

This document explains the use of all the registry keys used by the "Universal Driver Package".

In order for device specific changes to be effective, the device must be restarted (i.e. disable the device and then enable it) in Device Manager, or the system must be rebooted.

For user specific changes to become effective, it is necessary to log off and log back on again.

Registry Keys for Serial

Device Parameters for Elo serial touchscreen are located in subkeys of

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\SERENUM\

ELOSERIAL*******\Device Parameters key.

The "******" is the id used by the Windows system.

Touch Mode

Name:	MouseMode
Data Type:	DWORD (REG_DWORD)

Explanation:

Touchscreen mode

Possible Values:

Default is 6

0: Click on touch

1: Click on release

6: Mouse emulation

Name: EnableQuickTouch

Data Type: DWORD (REG_DWORD)

Explanation:

Enables / disables quick touch mode

Possible Values:

Default is 0

0: Disabled

1: Enabled

Name: QuickTouchDx

Data Type: DWORD (REG_DWORD)

Explanation:

X distance in Windows virtual coordinates outside which quick touch is enabled. Within this distance two quick touches are interpreted as regular touches.

Name: QuickTouchDy

Data Type: DWORD (REG_DWORD)

Explanation:

Y distance in Windows virtual coordinates outside which quick touch is enabled. Within this distance two quick touches are interpreted as regular touches.

Name: UntouchTimeOut

Data Type: DWORD (REG_DWORD)

Explanation:

"UntouchTimeOut" defines the time in milliseconds for the timeout. If there is constant touch at the same location a Button Up event will be generated automatically after the timeout period expires.

Possible Values:

Default value is 10 seconds.

Name:
UntouchHeight

Data Type:
DWORD (REG_DWORD)

Explanation:

"UntouchHeight" defines the height of the untouch timeout rectangle in windows virtual coordinates. If there is constant touch within the rectangle defined by UntouchHeight and UntouchWidth, a Button Up event is generated after the timeout period defined by UntouchTimeout expires.

Possible Values:

4000

Name:
UntouchWidth

Data Type:
DWORD (REG_DWORD)

Explanation:

"UntouchWidth" defines the width of the untouch timeout rectangle in windows virtual coordinates. If there is constant touch within the rectangle defined by UntouchHeight and UntouchWidth, a Button Up event is generated after the timeout period defined by UntouchTimeout expires.

Possible Values:

4000

Name:
SwapButton

Data Type:
DWORD (REG_DWORD)

Explanation:

Swaps the mouse left and right buttons. The buttons remain swapped for the count of touches this value is set to. It remains swapped until this count is decremented to 0. After that, the buttons are automatically swapped back to their original state.

Possible Values:

By default this key is not present in the registry. When the DLL API feature is used the mouse buttons are swapped and this value is added.

Name: LeftHandedUser

Data Type: DWORD (REG_DWORD)

Explanation:

This is applicable for left-handed mouse users.

Possible Values:

Default is 0

0: Disabled

1: Enabled

Name: RightButtonActive

Data Type: DWORD (REG_DWORD)

Explanation:

Indicates that the right button application should be run at user logon for this screen.

Possible Values:

1: Run right button at user logon.

2: Do not run right button for this screen at user logon.

Differentiate Touch Data from Mouse Messages in Windows Application

Name: MouseExtraInformation

Data Type: DWORD (REG_DWORD)

Explanation:

If you want to differentiate between touch data and mouse data, add this key to the registry. This value will be sent as mouse extra information while handling mouse messages.

Possible Values:

This key is not present by default.

Right Click On Hold

Name: RightClickHW

Data Type: DWORD (REG_DWORD)

Explanation:

Height and width of the Right click box within which touch is interpreted as same touch point. This is the width and height in windows virtual coordinates.

Possible Values: Default is 1867.

Name: DefaultRightClickDelay

Data Type: DWORD (REG_DWORD)

Explanation:

Right click on hold is generated after this timeout occurs. The delay is specified in milliseconds.

Possible Values:

Default is 750.

This value must be in the range from MinRightClickDelay and MaxRightClickDelay.

Name: MinRightClickDelay

Data Type: DWORD (REG_DWORD)

Explanation:

Minimum delay value allowed for Right click on hold. The delay is specified in milliseconds.

Possible Values:

Default is 250.

Name: MaxRightClickDelay

Data Type: DWORD (REG_DWORD)

Explanation:

Maximum delay value allowed for Right click on hold. The delay is specified in milliseconds.

Possible Values:

Default is 2500.

Name: RightButtonOnHoldCount

Data Type: DWORD (REG_DWORD)

Explanation:

The value for which right click on hold feature should be active. Once this value reaches 0 this feature is disabled. Value of 0xffffffff keeps it turned on forever.

Possible Values:

Default is 0xffffffff.

Name: RightButtonOnHoldActive

Data Type: DWORD (REG_DWORD)

Explanation:

If this value is 1 the right click on hold feature remains turned on after system reboot, else it gets turned off. This works in combination with the RightButtonOnHoldCount key.

Possible Values:

1: If right click on hold feature is turned on.

0: Otherwise.

Name: RightClickInitialTimeout

Data Type: DWORD (REG_DWORD)

Explanation:

The clock painted to indicate the right click on hold feature, begins to paint after this timeout occurs. This allows a delay before painting the clock on the screen. The delay is specified in milliseconds.

Possible Values:

Default is 200.

Serial Port Configuration

Name: PortFriendlyName

Data Type: String (REG_SZ)

Explanation:

PortFriendlyName identifies the serial port the touchscreen is connected to. The friendly name is the COM port identifier string used by Device Manager and can be found by expanding the Ports node of the Device Manager hardware tree.

To change the COM port associated with a touchscreen, replace the existing value in HKLM\System\CurrentControlSet\Services \EloTouchscreen\LegacyPorts**\FriendlyName with the friendly name of the desired COM port and restart the system.

Possible Values:

Communications Port (COM1)

Communications Port (COM2)

Name: BaudRate

Data Type: DWORD (REG_DWORD)

Explanation:

Baud rate used for serial port communication. Controller must be externally configured to support any baudrate other than 9600.

Possible Values:

Default is 9600.

Allowed: 1200, 2400, 4800, 9600

Name: HardwareHandshaking

Data Type: DWORD (REG_DWORD)

Explanation:

This key is used only for serial cables which are built using 2 wires and do not support hardware handshake.

Possible Values:

By default hardware handshaking is always turned on. To turn it off, explicitly set this key to 0.

Enable/Disable Touch

Name: PacketEnable

Data Type: DWORD (REG_DWORD)

Explanation:

Touch is enabled / disabled depending on this value.

Possible Values:

1: Enabled

2: Disabled

3: Reserved

Beep on Touch

Name: BeepFlag

Data Type: DWORD (REG_DWORD)

Explanation:

Beep is enabled / disabled depending on this value.

Possible Values:
0: Disabled
1: Enabled

Name: BeepTime

Data Type: DWORD (REG_DWORD)

Explanation:

Defines the duration of beep on touch in milliseconds.
Possible Values:
Default is 0x0014.

Name: BeepFreq

Data Type: DWORD (REG_DWORD)

Explanation:

Frequency of beep on touch in Hz.
Possible Values:
Default is 0x00300

Drag Delay

Name: DefaultDragDelay

Data Type: DWORD (REG_DWORD)

Explanation:

Drag delay value in milliseconds.
Possible Values:
Default is 50

Name: MinDragDelay

Data Type: DWORD (REG_DWORD)

Explanation:

Minimum drag delay value on the scale in Control Panel in milliseconds. Reserved for use by control panel and driver.

Possible Values:

Default is 50

Name: MaxDragDelay

Data Type: DWORD (REG_DWORD)

Explanation:

Maximun drag delay value on the scale in Control Panel in milliseconds. Reserved for use by control panel and driver.

Possible Values:

Default is 2500

Full Screen Bounding Rectangle and Bounding Mode

Name FullScreenVmode

Data Type: DWORD (REG_DWORD)

Explanation:

Defines the mode for the touchscreen boundary in the virtual desktop mode

Possible Values:

Default is 2

0: Disable virtual desktop. Touch on all screens will reflect on the primary monitor.

1: Enable virtual desktop. In this mode no individual screen bounds are used for the monitor. Touches toward the edge may generate clicks on adjacent monitors.

2: Enable virtual desktop. Screen bounds are enabled for each monitor. Touches on the screen always generate clicks on the associated monitor. A touch outside the bounding rectangle will cause the cursor to move to a point along the boundary that is nearest to the point of touch.

3: Enable virtual desktop. Screen bounds are enabled for each monitor. Touches on the screen always generate clicks on the associated monitor. Touches outside the bounding rectangle are not sent to the system.

Name: FullScreenBounds

Data Type: Array of DWORDs (REG_BINARY)

Explanation:

Defines the full screen bounding rectangle in Windows virtual coordinates for the screen.

Calibration

The following defines the calibration data used to convert touches from the Elo coordinate system to Windows virtual coordinate system.

The driver uses the following equation to convert a raw touch coordinate point from Elo coordinate system to the Windows screen coordinate system. Calibration equation is:

Xcal = a + m*Xuncal,

where,
Xuncal, is in Elo coordinate system
Xcal, is in Windows coordinate system
m = X_ScrDx / X_EloDx
"a", is the X offset value entry
X_ScrDx = distance between targets in virtual coordinates
X_EloDx = distance between targets in Elo coordinates.

Name: X_EloDx, X_ScrDx, X_Offset, Y_EloDx, Y_ScrDx, Y_Offset, Z_EloDx, Z_ScrDx, Z_Offset

Data Type: DWORD (REG_BINARY)

Explanation:

Data required by calibration. Use the equation above to calculate these values.

Name: SwapXY

Data Type: DWORD (REG_DWORD)

Explanation:

Specifies if the touchscreen is rotated 90 degrees or 270 degrees.

Possible Values:

Set to 1 if touchscreen is rotated 90 degrees or 270 degrees else set to 0

Name: WindowsMonitorNumber

Data Type: DWORD (REG_DWORD)

Explanation:

Windows monitor number.

Possible Values:

Windows monitor numbers are 1 based.

Name: X_Resolution

Data Type: DWORD (REG_DWORD)

Explanation:

Screen width in pixels.

Name: Y_Resolution

Data Type: DWORD (REG_DWORD)

Explanation:

Screen height in pixels.

Name: xVirtScrSize

Data Type: DWORD (REG_DWORD)

Explanation:

Width in pixels of the Windows virtual screen.

Name: yVirtScrSize

Data Type: DWORD (REG_DWORD)

Explanation:

Height in pixels of the Windows virtual screen.

Name: xVirtScrCoord

Data Type: DWORD (REG_DWORD)

Explanation:

Top left X coordinate of the Windows virtual screen.

Name: yVirtScrCoord

Data Type: DWORD (REG_DWORD)

Explanation:

Top left Y coordinate of the Windows virtual screen.

Name: xMonLocn

Data Type: DWORD (REG_DWORD)

Explanation:

Monitor location (X) for this monitor on the virtual desktop.

Name: yMonLocn

Data Type: DWORD (REG_DWORD)

Explanation:

Monitor location (Y) for this monitor on the virtual desktop.

Edge of Touchscreen Cursor Acceleration

Name: EdgeAccelerationEnable

Data Type: DWORD (REG_DWORD)

Explanation:

Enables / disables acceleration.

Possible Values:

Default is 0

0: Disable acceleration.

1: Enable acceleration

Name: EdgeAccelerationBounds

Data Type: Array of DWORDS (REG_BINARY)

Explanation:

Rectangle bounds (XMin, YMin, XMax, YMax) beyond which cursor movement is accelerated. Within the defined rectangle touch is interpreted normally. XMin, YMin, XMax, YMax are defined in Windows virtual screen coordinates.

Name: EdgeAccelerationScale

Data Type: DWORD (REG_DWORD)

Explanation:

Acceleration value.

Possible Values:

Default is unused.

It can be anywhere in the range of 0-100. 10 indicates no acceleration.

Registry Keys for USB Device

Device Parameters for USB touchscreen are located in the sub keys of the HID key and the USB key.

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\USB \Vid_04e7&Pid_???*******\Device Parameters key for USB.

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\HID\Vid_04e7&Pid_???*******\Device Parameters key for HID.

where, "???" is the product id.
And the "******" is the serial number for the touchscreen.

Enable / Disable Touch

Name: PacketEnable

Data Type: DWORD (REG_DWORD)

Explanation:

Touch is enabled / disabled depending on this value.

Possible Values:

1: Enabled

2: Disabled

Full Screen Bounding Rectangle and Bounding Mode

Name: FullScreenVmode

Data Type: DWORD (REG_DWORD)

Explanation:

Defines the mode for the touchscreen boundary in the virtual desktop mode.

Possible Values:

Default is 2.

0: Disable virtual desktop. Touch on all screens will reflect on the primary monitor.

1: Enable virtual desktop. In this mode no individual screen bounds are used for the monitor. Touches toward the edge may generate clicks on adjacent monitors.

2: Enable virtual desktop. Screen bounds are enabled for each monitor. Touches on the screen always generate clicks on the associated monitor. A touch outside the bounding rectangle will cause the cursor to move to a point along the boundary that is nearest to the point of touch.

3: Enable virtual desktop. Screen bounds are enabled for each monitor. Touches on the screen always generate clicks on the associated monitor. Touches outside the bounding rectangle are not sent to the system.

Name: FullScreenBounds

Data Type: Array of DWORDs (REG_BINARY)

Explanation:

Defines the full screen bounding rectangle in Windows virtual coordinates for the screen.

Calibration

The following defines the calibration data used to convert touches from the Elo coordinate system to Windows virtual coordinate system.

The driver uses the following equation to convert a raw touch coordinate point from Elo coordinate system to the Windows screen coordinate system.

Calibration equation is
Xcal = a + m*Xuncal,

where,
Xuncal, is in Elo coordinate system
Xcal, is in Windows coordinate system
m = X_ScrDx / X_EloDx
"a", is the X offset value entry
X_ScrDx = distance between targets in virtual coordinates
X_EloDx = distance between targets in Elo coordinates.

Name: X_EloDx, X_ScrDx, X_Offset, Y_EloDx, Y_ScrDx, Y_Offset, Z_EloDx, Z_ScrDx, Z_Offset

Data Type: DWORD (REG_DWORD)

Explanation:

Data required by calibration. Use the equation above to calculate these values.

Name: SwapXY

Data Type: DWORD (REG_DWORD)

Explanation:

Specifies if the touchscreen is rotated 90 degrees or 270 degrees.

Possible Values:

Set to 1 if touchscreen is rotated 90 degrees or 270 degrees else set to 0

Name: WindowsMonitorNumber

Data Type: DWORD (REG_DWORD)

Explanation:

Windows monitor number.

Possible Values:

Windows monitor numbers are 1 based.

Name: X_Resolution

Data Type: DWORD (REG_DWORD)

Explanation:

Screen width in pixels.

Name: Y_Resolution

Data Type: DWORD (REG_DWORD)

Explanation:

Screen height in pixels.

Name: xVirtScrSize

Data Type: DWORD (REG_DWORD)

Explanation:

Width in pixels of the Windows virtual screen.

Name: yVirtScrSize

Data Type: DWORD (REG_DWORD)

Explanation:

Height in pixels of the Windows virtual screen.

Name: xVirtScrCoord

Data Type: DWORD (REG_DWORD)

Explanation:

Top left X coordinate of the Windows virtual screen.

Name: yVirtScrCoord

Data Type: DWORD (REG_DWORD)

Explanation:

Top left Y coordinate of the Windows virtual screen.

Name: xMonLocn

Data Type: DWORD (REG_DWORD)

Explanation:

Monitor location (X) for this monitor on the virtual desktop.

Name: yMonLocn

Data Type: DWORD (REG_DWORD)

Explanation:

Monitor location (Y) for this monitor on the virtual desktop.

Edge of Touchscreen Cursor Acceleration

Name: EdgeAccelerationEnable

Data Type: DWORD (REG_DWORD)

Explanation:

Enables / disables acceleration.

Possible Values:

Default is 0.

0: Disable acceleration.

1: Enable acceleration.

Name: EdgeAccelerationBounds

Data Type: Array of DWORDS (REG_BINARY)

Explanation:

Rectangle bounds (XMin, YMin, XMax, YMax) beyond which cursor movement is accelerated. Within the defined rectangle touch is interpreted normally. XMin, YMin, XMax, YMax are defined in Windows virtual screen coordinates.

Name: EdgeAccelerationScale

Data Type: DWORD (REG_DWORD)

Explanation:

Acceleration value.

Possible Values:

Default is unused.

It can be anywhere in the range of 0-100. 10 indicates no acceleration.

Registry Keys for HID

Touch Mode

Name: MouseMode

Data Type: DWORD (REG_DWORD)

Explanation:

Touchscreen mode.

Possible Values:

Default is 6.

0: Click on touch.

1: Click on release.

6: Mouse emulation.

Name: EnableQuickTouch

Data Type: DWORD (REG_DWORD)

Explanation:

Enables / disables quick touch mode

Possible Values:

Default is 0.

0: Disabled

1: Enabled

Name: QuickTouchDx

Data Type: DWORD (REG_DWORD)

Explanation:

X distance in Windows virtual coordinates outside which quick touch is enabled. Within this distance two quick touches are interpreted as regular touches.

Name: QuickTouchDy

Data Type: DWORD (REG_DWORD)

Explanation:

Y distance in Windows virtual coordinates outside which quick touch is enabled. Within this distance two quick touches are interpreted as regular touches.

Name:
UntouchTimeOut

Data Type:
DWORD (REG_DWORD)

Explanation:

"UntouchTimeOut" defines the time in milliseconds for the timeout. If there is constant touch at the same location a Button Up event will be generated automatically after the timeout period expires.

Possible Values:

Default value is 10 seconds.

Name:
UntouchHeight

Data Type:
DWORD (REG_DWORD)

Explanation:

"UntouchHeight" defines the height of the untouch timeout rectangle in windows virtual coordinates. If there is constant touch within the rectangle defined by UntouchHeight and UntouchWidth, a Button Up event is generated after the timeout period defined by UntouchTimeout expires.

Possible Values:

Name:
UntouchWidth

Data Type:
DWORD (REG_DWORD)

Explanation:

"UntouchWidth" defines the width of the untouch timeout rectangle in windows virtual coordinates. If there is constant touch within the rectangle defined by UntouchHeight and UntouchWidth, a Button Up event is generated after the timeout period defined by UntouchTimeout expires.

Possible Values:

Name: SwapButton

Data Type: DWORD (REG_DWORD)

Explanation:

Swaps the mouse left and right buttons. The buttons remain swapped for the count of touches this value is set to. It remains swapped until this count is decremented to 0. After that, the buttons are automatically swapped back to their original state.

Possible Values:

By default this key is not present in the registry. When the DLL API feature is used the mouse buttons are swapped and this value is added.

Name: LeftHandedUser

Data Type: DWORD (REG_DWORD)

Explanation:

This is applicable for left-handed mouse users.

Possible Values:

Default is 0.

0: Disabled

1: Enabled

Name: RightButtonActive

Data Type: DWORD (REG_DWORD)

Explanation:

Indicates that the right button application, should be run at user logon for this screen.

Possible Values:

1: Run right button at user logon.

2: Do not run right button for this screen at user logon.

Differentiate Touch Data from Mouse Messages in Windows Application

Name: MouseExtraInformation

Data Type: DWORD (REG_DWORD)

Explanation:

If you want to differentiate between touch data and mouse data, add this key to the registry. This value will be sent as mouse extra information while handling mouse messages.

Possible Values:

This key is not present by default.

Right Click On Hold

Name: RightClickHW

Data Type: DWORD (REG_DWORD)

Explanation:

Height and width of the Right click box within which touch is interpreted as same touch point. This is the width and height in windows virtual coordinates.

Possible Values: Default is 1867.

Name: DefaultRightClickDelay

Data Type: DWORD (REG_DWORD)

Explanation:

Right click on hold is generated after this timeout occurs. The delay is specified in milliseconds.

Possible Values:

Default is 750.

This value must be in the range from MinRightClickDelay and MaxRightClickDelay.

Name:
MinRightClickDelay

Data Type:
DWORD (REG_DWORD)

Explanation:

Minimum delay value allowed for Right click on hold. The delay is specified in milliseconds.

Possible Values:

Default is 250.

Name:
MaxRightClickDelay

Data Type:
DWORD (REG_DWORD)

Explanation:

Maximum delay value allowed for Right click on hold. The delay is specified in milliseconds.

Possible Values:

Default is 2500.

Name:
RightButtonOnHoldCount

Data Type:
DWORD (REG_DWORD)

Explanation:

The value for which right click on hold feature should be active. Once this value reaches 0 this feature is disabled. Value of 0xffffffff keeps it turned on forever.

Possible Values:

Default is 0xffffffff.

Name:
RightButtonOnHoldActive

Data Type:
DWORD (REG_DWORD)

Explanation:

If this value is 1 the right click on hold feature remains turned on after system reboot, else it gets turned off. This works in combination with the RightButtonOnHoldCount key.

Possible Values:

1: If right click on hold feature is turned on.

0: Otherwise.

Name: RightClickInitialTimeout

Data Type: DWORD (REG_DWORD)

Explanation:

The clock painted to indicate the right click on hold feature, begins to paint after this timeout occurs. This allows a delay before painting the clock on the screen. The delay is specified in milliseconds.

Possible Values:

Default is 200.

Beep On Touch

Name: BeepFlag

Data Type: DWORD (REG_DWORD)

Explanation:

Beep is enabled / disabled depending on this value.

Possible Values:

0: Disabled

1: Enabled

Name: BeepTime

Data Type: DWORD (REG_DWORD)

Explanation:

Defines the duration of beep on touch in milliseconds.

Possible Values:

Default is 0x0014

Name: BeepFreq

Data Type: DWORD (REG_DWORD)

Explanation:

Frequency of beep on touch in Hz.

Possible Values:

Default is 0x00300

Drag Delay

Name: DefaultDragDelay

Data Type: DWORD (REG_DWORD)

Explanation:

Drag delay value in milliseconds.

Possible Values:

Default is 20.

Name: MinDragDelay

Data Type: DWORD (REG_DWORD)

Explanation:

Minimum drag delay value on the scale in Control Panel in milliseconds. Reserved for use by control panel and driver.

Name: MaxDragDelay

Data Type: DWORD (REG_DWORD)

Explanation:

Maximum drag delay value on the scale in Control Panel in milliseconds. Reserved for use by control panel and driver.

User Specific Registry Keys

User specific configuration data are saved in this key. User key is located in

HKEY_CURRENT_USER\Software\EloTouchscreen

Name: DoubleClickHW

Data Type: DWORD (REG_DWORD)

Explanation:

Defines the double click width and height.

Possible Values:

Default value is 25

Name: EloTTray

Data Type: DWORD (REG_DWORD)

Explanation:

This determines if Tool tray application should be run for this user when the user logs in.

Possible Values:

Default value is 1.

1: Run Elo tool tray app

2: Do not run Elo tool tray app.

Name: RButton

Data Type: Array of DWORDs(REG_BINARY)

Explanation:

Saves x,y Location of Right button application in Windows virtual coordinates.

Home Page Top of page

Appendix B

Elo Software Development Kit

Introduction

Elo SDK provides a programmers API for communication with the Elo Serial and USB drivers. The following functionality is available for programming:

Get Raw and Calibrated Touch Points from the Touchscreen

Gets touch data from the controller. This data can be retrieved in raw elo coordinates or translated windows coordinate format. To get data use EloGetTouch. This call may block depending on the flags used for type of data expected. To cancel the blocking call use EloCancel.

EloGetTouch

EloCancel

Change Touchscreen Operation Modes

Touchscreen operates in the following modes:

Click On Touch: Click on touch immediately sends a mouse down/up message at the point of touch on the touchscreen. The user's finger must be removed from the touchscreen before a new touch at any location will be recognized. The cursor or selected objects CANNOT be "dragged" on the screen in this mode.

Click On Release: At the time of release (untouch), a mouse down/up message is sent at the point that the screen was last touched. Dragging across objects on the screen will not highlight or select them unless untouch occurs when the touch is over the object. (Drag and Double-click)

Mouse Emulation: Sends a mouse down message at the point of contact. Selects an object if it was at the initial point of contact. Drags a selected object on the screen. Sends a mouse up message at the point of untouch. Double-clicks on an object when the screen is touched twice in succession at the same location.

These are the same as the touch modes in the Elo Control Panel. You may get or change the operation mode using the calls listed below:

EloGetMouseMode

EloSetMouseMode

Configure Drag Delay

Dragdelay allows the drag functionality for touchscreen. Touchscreen must be calibrated before using this functionality. This value can be retrieved or modified using:

EloGetDragDelay

EloSetDragDelay

Enable/Disable Touch Functionality

The Touchscreen can be enabled to report touch data to the windows system or disabled not to report any touch data.

EloGetTouchState

EloSetTouchState

Configure Sound During Touch

Every complete touch is indicated by a beep from the system. Touchscreen must be calibrated before using this functionality. The frequency and time and beep is configurable.

EloGetBeep

EloSetBeep

Save Alignment for Touchscreen Using Calibration API

Elo provides a standard calibration utility for video alignment. The CALIBRATION is used to convert touches from the Elo coordinate system to Windows virtual coordinate system.

The driver uses the following equation to convert a raw touch coordinate point from Elo coordinate system to the Windows screen coordinate system.

Calibration equation is

Xcal = a + m*Xuncal,

where,

Xuncal, is in Elo coordinate system

Xcal, is in Windows coordinate system

m = nScrDx / nEloDx

"a", is the X offset value entry

nScrDx = distance between targets in Windows virtual coordinates

nEloDx = distance between targets in Elo coordinates.

This is specially useful if you want to save the calibration over the network or some other place, and later configure the touchscreen using one common set of data.

To get the current calibration data from the driver, use EloGetCalibrationData. For changing the calibration data use EloSetCalibrationData.

EloGetCalibrationData

EloSetCalibrationData

Define Touchscreen Boundary

Touchscreen boundary defines the bounding rectangle for the touchscreen on Windows virtual desktop. This allows / prohibits touch on one monitor from being propagated to the corresponding monitor. Touchscreen must be calibrated before using this functionality.

EloGetTouchBoundary

EloSetTouchBoundary

Swap Touchscreen Buttons

Use this call to use the touchscreen with swapped button for next couple of touches on the screen.

Swaps the button for the touch monitor for the touch count specified. Once the count is exhausted, the SDK notifies the application by sending a WM_SWAPBUTTON message to the application window. Touchscreen must be calibrated before using this functionality.

EloSwapButton

Configure Touchscreen for Left-handed Users

For left-handed users of mouse, set the touchscreen mode using EloSetLeftHandedMouse call.

EloSetLeftHandedMouse

Retrieve Touchscreen Diagnostic Data

Diagnostic information for touchscreen can be retrieved using EloGetDiagnosticsData.

This information is also displayed in the Elo Control panel>Properties tab.

EloGetDiagnosticData

Configure Touchscreen for Gaming Specific Mode by Enabling Quick Touch

Quick touch is an enhanced mode of the touchscreen, usually used for gaming purposes.

In standard operation, touches on screen following each other very quickly in "Mouse Emulation" mode creates a system drag box.

This feature allows the touches following each other quickly to be interpreted as independent touches resulting in separate mouse clicks.

Touchscreen must be calibrated before using this functionality.

EloGetQuickTouch

EloSetQuickTouch

Configure Edge Acceleration Feature

Touching towards the edge of the touchscreen can be difficult at times. This feature allows the cursor to be accelerated towards the screen edges.

Touchscreen must be calibrated before using this functionality.

EloGetAcceleration

EloSetAcceleration

Define Touchscreen to Automatically Timeout After Constant Touch in Same Location Using Untouch Timeout Feature

If touches are created in the same constant location, an automatic untouch is generated by the driver based on this timeout.

For dirt on a screen this is a very handy feature.

Touchscreen must be calibrated before using this functionality.

EloGetUntouchTimeout

EloSetUntouchTimeout

Configure Right Click on Hold Functionality

This feature allows touchscreen to generate right clicks. A right click is generated when the user continues to touch the screen. Touchscreen must be calibrated before using this functionality. This can be configured using:

EloGetRightClickOnHold

EloSetRightClickOnHold

EloPubIf.lib provides the SDK for communicating with Elo Serial and USB drivers. In order to communicate with either of the drivers, add this library to the list of compile time dependencies of your application and use the appropriate interface function calls in your program. Include the header file, EloInterface.h and EloErrorCodes.h in your application.

These files can be found on Program Files\EloTouchsystems folder.

Please refer to the Appendix C, EnableDisable Sample Code for interface usage.

Elo Device Interface Functions

Function Name: EloGetScreenInfo

int EloGetScreenInfo(DWORD dwMonNum[32], int iScrCnt)

Parameters:
dwMonNum [in]: Array of DWORD to receive the Windows monitor number associated with the touchscreens.

iScrCnt [out]: It retrieves the total number of Elo touchscreens found.

Return Values: Returns EloSuccess if the call succeeds, it returns an error code otherwise. See Error Codes section for list of error values.

Remarks:

It returns the list of Windows monitor numbers associated with the touchscreens where, the index is the touchscreen number and the value is the Windows monitor number.

Touchscreens are 0 based and Windows monitor numbers are 1 based.

If a touchscreen is not calibrated, the windows monitor is returned as -1.

Maximum of 32 touchscreens are supported.

Requirements
Included in Version 4.20 and later.
Header: Declared in EloInterface.h; include EloInterface.h.
Library: Use EloPubIf.lib.

See Also Sample and Error Codes

Function Name: EloGetTouch

int EloGetTouch (PTOUCH_POINT xy, BOOL xlated, GETPOINTS_CODE getCode, UINT nScrNo)

Parameters:
xy [out]: Pointer to TOUCH_POINT structure to receive the touch coordinates from the touchscreen. The application is responsible for allocating / freeing memory. Please see structure definition section for TOUCH_POINT.

xlated [in]: If TRUE, the coordinates are returned translated for Windows coordinate system. If FALSE, raw coordinate data are returned.

getCode [in]: This can be one of the following values: ReturnImmediately, ReturnOnTouch, ReturnOnUntouch, ReturnOnNextValidTouch. Please see definition for GETPOINTS_CODE.

nScrNo [in/out]: This must be initialized to the zero based touchscreen number. Returns the zero based touchscreen number for the touchscreen touched.

Return Values: Returns EloSuccess if the call succeeds, it returns an error code otherwise. See Error Codes section for list of error values.

Remarks:
Unless the ReturnImmediately GetCode value is used, this call will not return until the user touches the screen specified.

To cancel the blocking wait use the EloCancel function.

Requirements
Included in Version 4.20 and later.
Header: Declared in EloInterface.h; include EloInterface.h.
Library: Use EloPubIf.lib.

See Also EloCancel, EloGetScreenInfo, Sample and Error Codes

Function Name: EloCancel

int EloCancel()

Parameters:
None.

Return Values: Returns EloSuccess if the call succeeds, it returns an error code otherwise. See Error Codes section for list of error values.

Remarks:
Cancels any pending request with the driver.

Requirements
Included in Version 4.20 and later.
Header: Declared in EloInterface.h; include EloInterface.h.
Library: Use EloPubIf.lib.

See Also EloGetTouch,EloGetScreenInfo, Sample and Error Codes

Function Name: EloGetMouseMode

int EloGetMouseMode (WORD *wMode,UINT nScrNo)

Parameters:
wMode [out]: Returns the touchscreen mode. Please see Constant definition section for TOUCHSCREEN_MODE.

nScrNo [in]: 0 based touchscreen number.

Return Values: Returns EloSuccess if the call succeeds, it returns an error code otherwise. See Error Codes section for list of error values.

Remarks:
Retrieves the touch mode for the specified touchscreen.

Requirements
Included in Version 4.20 and later.
Header: Declared in EloInterface.h; include EloInterface.h.
Library: Use EloPubIf.lib.

See Also EloSetMouseMode, EloGetScreenInfo, Sample and Error Codes

Function Name: EloSetMouseMode

int EloSetMouseMode (WORD wMode,UINT nScrNo)

Parameters:
wMode [in]: Touchscreen mode. Please see Constant definition section for TOUCHSCREEN_MODE.

nScrNo [in]: 0 based touchscreen number.

Return Values: Returns EloSuccess if the call succeeds, it returns an error code otherwise. See Error Codes section for list of error values.

Remarks:
Sets the touch mode for the specified touchscreen to wMode.

Requirements
Included in Version 4.20 and later.
Header: Declared in EloInterface.h; include EloInterface.h.
Library: Use EloPubIf.lib.

See Also EloGetMouseMode, EloGetScreenInfo, Sample and Error Codes

Function Name: EloGetTouchReportingState

int EloGetTouchReportingState (BOOL *bFlag , UINT nScrNo)

Parameters:
bFlag [out]: Touch state enabled / disabled. If TRUE, touch data is reported. If FALSE, touch data is not reported.

nScrNo [in]: 0 based touchscreen number.

Return Values: Returns EloSuccess if the call succeeds, it returns an error code otherwise. See Error Codes section for list of error values.

Remarks:
Gets the touch state for touchscreen specified. It saves it in bFlag.

Requirements
Included in Version 4.20 and later.
Header: Declared in EloInterface.h; include EloInterface.h.
Library: Use EloPubIf.lib.

See Also EloSetTouchReportingState, EloGetScreenInfo, Sample and Error Codes

Function Name: EloSetTouchReportingState

int EloSetTouchReportingState (BOOL bFlag, int nScrNo)

Parameters:
bFlag [in]: Touch state enabled / disabled. If TRUE, touch data is reported. If FALSE, touch data is not reported.

nScrNo [in]: 0 based touchscreen number.

Return Values: Returns EloSuccess if the call succeeds, it returns an error code otherwise. See Error Codes section for list of error values.

Remarks:
Enables or Disables touch depending on bFlag.

Requirements
Included in Version 4.20 and later.
Header: Declared in EloInterface.h; include EloInterface.h.
Library: Use EloPubIf.lib.

See Also EloGetTouchReportingState, EloGetScreenInfo, Sample and Error Codes

Function Name: EloGetDragDelay

int EloGetDragDelay (PDRAG_DELAY pDragDelay,UINT nScrNo)

Parameters:
pDragDelay [out]: Pointer to DRAG_DELAY to receive Drag Delay . Please see structure definition section for PDRAG_DELAY.

nScrNo [in]: 0 based touchscreen number.

Return Values: Returns EloSuccess if the call succeeds, it returns an error code otherwise. See Error Codes section for list of error values.

Remarks:
Gets the dragdelay parameters for the specified touchscreen.

Requirements
Included in Version 4.20 and later.
Header: Declared in EloInterface.h; include EloInterface.h.
Library: Use EloPubIf.lib.

See Also EloSetDragDelay ,EloGetScreenInfo, Sample and Error Codes

Function Name: EloSetDragDelay

int EloSetDragDelay (PDRAG_DELAY pDragDelay, UINT nScrNo)

Parameters:
pDragDelay [in]: Dragdelay data. Please see structure definition section for DRAG_DELAY.

nScrNo [in]: 0 based touchscreen number.

Return Values: Returns EloSuccess if the call succeeds, it returns an error code otherwise. See Error Codes section for list of error values.

Remarks:
Sets the drag delay for the specified touchscreen.

Requirements
Included in Version 4.20 and later.
Header: Declared in EloInterface.h; include EloInterface.h.
Library: Use EloPubIf.lib.

See Also EloGetDragDelay ,EloGetScreenInfo, Sample and Error Codes

Function Name: EloGetBeep

int EloGetBeep (PBEEP pBeepVal, UINT nScrNo)

Parameters:
pBeepVal [out]: Pointer to BEEP structure to receive data. Please see structure definition section for BEEP.

nScrNo [in]: 0 based touchscreen number.

Return Values: Returns EloSuccess if the call succeeds, it returns an error code otherwise. See Error Codes section for list of error values.

Remarks:
Gets the touchscreen beep data for the specified touchscreen.

Requirements
Included in Version 4.20 and later.
Header: Declared in EloInterface.h; include EloInterface.h.

Library: Use EloPubIf.lib.

See Also EloSetBeep, EloGetScreenInfo, Sample and Error Codes

Function Name: EloSetBeep

int EloSetBeep (PBEEP pBeepVal, UINT nScrNo)

Parameters:
pBeepVal [in]: Pointer to sound structure to receive data. Please see structure definition section for BEEP definition.

nScrNo [in]: 0 based touchscreen number.

Return Values: Returns EloSuccess if the call succeeds, it returns an error code otherwise. See Error Codes section for list of error values.

Remarks:
Sets the touchscreen beep data for the specified touchscreen.

Requirements
Included in Version 4.20 and later.
Header: Declared in EloInterface.h; include EloInterface.h.
Library: Use EloPubIf.lib.

See Also EloGetBeep, EloGetScreenInfo, Sample and Error Codes

Function Name: EloGetTouchBoundary

int EloGetTouchBoundary(PTOUCH_BOUNDARY ptBndry ,
UINT nScrNo)

Parameters:
ptBndry [out]: Pointer to TOUCH_BOUNDARY structure containing full screen boundary data in pixels. Please see structure definition section for TOUCH_BOUNDARY.

nScrNo [in]: 0 based touchscreen number.

Return Values: Returns EloSuccess if the call succeeds, it returns an error code otherwise. See Error Codes section for list of error values.

Remarks:
Gets the full screen bounding rectangle and bounding mode for the specified touchscreen.

Requirements
Included in Version 4.20 and later.
Header: Declared in EloInterface.h; include EloInterface.h.
Library: Use EloPubIf.lib.

See Also EloSetTouchBoundary, EloGetScreenInfo, Sample and Error Codes

Function Name: EloSetTouchBoundary

int EloSetTouchBoundary(PTOUCH_BOUNDARY ptBndry,
UINT nScrNo)

Parameters:
ptBndry [in]: Pointer to TOUCH_BOUNDARY structure containing full screen boundary data in pixels. Please see structure definition section for TOUCH_BOUNDARY.

nScrNo [in]: 0 based touchscreen number.

Return Values: Returns EloSuccess if the call succeeds, it returns an error code otherwise. See Error Codes section for list of error values.

Remarks:
Sets the full screen bounding rectangle and bounding mode for the specified touchscreen.

Requirements
Included in Version 4.20 and later.
Header: Declared in EloInterface.h; include EloInterface.h.
Library: Use EloPubIf.lib.

See Also EloGetTouchBoundary, EloGetScreenInfo, Sample and Error Codes

Function Name: EloGetCalibrationData

int EloGetCalibrationData (PCALIBRATION pCalData, UINT nScrNo)

Parameters:
pCalData [out]: Pointer to CALIBRATION structure to receive the calibration data. Please see structure definition section for CALIBRATION.

nScrNo [in]: 0 based touchscreen number.

Return Values: Returns EloSuccess if the call succeeds, it returns an error code otherwise. See Error Codes section for list of error values.

Remarks:
Retrieves the calibration data for the specified touchscreen.

Requirements
Included in Version 4.20 and later.
Header: Declared in EloInterface.h; include EloInterface.h.
Library: Use EloPubIf.lib.

See Also EloSetCalibrationData, EloGetScreenInfo, Sample and Error Codes

Function Name: EloSetCalibrationData

int EloSetCalibrationData (CALIBRATION *pCalData,UINT nScrNo)

Parameters:
pCalData [in]: Pointer to CALIBRATION structure which stores the calibration data for the touchscreen. Please see structure definition section for CALIBRATION.

nScrNo [in]: 0 based touchscreen number.

Return Values: Returns EloSuccess if the call succeeds, it returns an error code otherwise. See Error Codes section for list of error values.

Remarks:
Sets the calibration data for the specified touchscreen.

Requirements
Included in Version 4.20 and later.
Header: Declared in EloInterface.h; include EloInterface.h.
Library: Use EloPubIf.lib.

See Also EloGetCalibrationData, EloGetScreenInfo, Sample and Error Codes

Function Name: EloSwapButton

int EloSwapButton (HWND hWnd, DWORD dwCnt, UINT nScrNo)

Parameters:

hWnd [in]: Handle to window to receive the WM_ELOSWAPBUTTON message. if NULL, there is no message.

dwCnt [in]: Touch count for which the button should be swapped.

nScrNo [in]: 0 based touchscreen number.

Return Values: Returns EloSuccess if the call succeeds, it returns an error code otherwise. See Error Codes section for list of error values.

Remarks:
Swaps the button for the touch monitor for the touch count specified in dwCnt. Once the count is exhausted the DLL notifies the application, by sending a WM_ELOSWAPBUTTON message to the application window specified in hWnd.

Requirements
Included in Version 4.20 and later.
Header: Declared in EloInterface.h; include EloInterface.h.
Library: Use EloPubIf.lib.

See Also: EloGetScreenInfo, Sample and Error Codes

Function Name: EloSetLeftHandedMouse

int EloSetLeftHandedMouse (BOOL bFlag)

Parameters:
bFlag [in]: If TRUE it sets touchscreen mode to left-handed mouse.

Return Values: Returns EloSuccess if the call succeeds, it returns an error code otherwise. See Error Codes section for list of error values.

Remarks:
Sets the touchscreen mode to left-handed mouse. This does not affect the Windows left-handed mouse settings.

Requirements
Included in Version 4.20 and later.
Header: Declared in EloInterface.h; include EloInterface.h.
Library: Use EloPubIf.lib.

See Also: EloGetScreenInfo, Sample and Error Codes

Function Name: EloGetDiagnosticsData

int EloGetDiagnosticsData(PSCREEN_PROPERTIES pData, UINT nScrNo)

Parameters:
pData [in]: Pointer to SCREEN_PROPERTIES structure to receive the diagnostics for the touchscreen. Please see structure definition section for SCREEN_PROPERTIES.

nScrNo [in]: 0 based touchscreen number.

Return Values: Returns EloSucces if the call succeeds, it returns an error code otherwise. See Error Codes section for list of error values.

Remarks:
Gets the diagnostics for the specified touchscreen. This information is also displayed in the Elo Control panel>Properties tab.

Requirements
Included in Version 4.20 and later.
Header: Declared in EloInterface.h; include EloInterface.h.
Library: Use EloPubIf.lib.

See Also: EloGetScreenInfo, Sample and Error Codes

Function Name: EloGetQuickTouch

int EloGetQuickTouch (PQUICK_TOUCH pQTouch, UINT nScrNo)

Parameters:
pQTouch [out]: Pointer to QUICK_TOUCH structure to receive the quick touch configuration data from the selected touchscreen. Please see structure definition section for QUICK_TOUCH.

nScrNo [in]: 0 based touchscreen number.

Return Values: Returns EloSucces if the call succeeds, it returns an error code otherwise. See Error Codes section for list of error values.

Remarks:
Gets the Quick Touch configuration parameters for the specified touchscreen.

Requirements
Included in Version 4.20 and later.
Header: Declared in EloInterface.h; include EloInterface.h.
Library: Use EloPubIf.lib.

See Also EloSetQuickTouch,EloGetScreenInfo, Sample and Error Codes

Function Name: EloSetQuickTouch

int EloSetQuickTouch (PQUICK_TOUCH pQTouch, UINT nScrNo)

Parameters:
pQTouch [in]: Pointer to QUICK_TOUCH structure containing the quick touch configuration data for the touchscreen. Please see structure definition section for QUICK_TOUCH.

nScrNo [in]: 0 based touchscreen number.

Return Values: Returns EloSuccess if the call succeeds, it returns an error code otherwise. See Error Codes section for list of error values.

Remarks:
Sets the Quick Touch configuration parameters for all touchscreens.

Requirements
Included in Version 4.20 and later.
Header: Declared in EloInterface.h; include EloInterface.h.
Library: Use EloPubIf.lib.

See Also EloGetQuickTouch EloGetScreenInfo, Sample and Error Codes

Function Name: EloGetEdgeAccel

int EloGetEdgeAccel(PEDGE_ACCEL pAccel, UINT nScrNo)

Parameters:
pAccel [out]: Pointer to EDGE_ACCEL structure to receive the edge accleration data for the touchscreen. Please see structure definition section for EDGE_ACCEL.

nScrNo [in]: 0 based touchscreen number.

Return Values: Returns EloSuccess if the call succeeds, it returns an error code otherwise. See Error Codes section for list of error values.

Remarks:
Gets the edge acceleration boundary and acceleration value for the given touchscreen.

Requirements

Included in Version 4.20 and later.
Header: Declared in EloInterface.h; include EloInterface.h.
Library: Use EloPubIf.lib.

See Also EloGetScreenInfo, EloSetEdgeAccel, Sample and Error Codes

Function Name: EloSetEdgeAccel

int EloSetEdgeAccel(PEDGE_ACCEL pAccel, UINT nScrNo)

Parameters:
pAccel [in]: Pointer to EDGE_ACCEL structure to containing the edge accleration data for the touchscreen. Please see structure definition section for EDGE_ACCEL.

nScrNo [in]: 0 based touchscreen number.

Return Values: Returns EloSuccess if the call succeeds, it returns an error code otherwise. See Error Codes section for list of error values.

Remarks:
Sets the edge acceleration boundary and acceleration scale for the given touchscreen.

Requirements:
Included in Version 4.20 and later.
Header: Declared in EloInterface.h; include EloInterface.h.
Library: Use EloPubIf.lib.

See Also: EloGetScreenInfo, EloGetEdgeAccel, Sample and Error Codes

Function Name: EloGetUntouchTimeout

int EloGetUntouchTimeout(PUNTOUCH_TIMEOUT pUTimeout, UINT nScrNo)

Parameters:
pUTimeout [out]: Pointer to UNTOUCH_TIMEOUT structure to receive the untouch timeout data for the touchscreen. Please see structure definition section for UNTOUCH_TIMEOUT.

nScrNo [in]: 0 based touchscreen number.

Return Values: Returns EloSuccess if the call succeeds, it returns an error code otherwise. See Error Codes section for list of error values.

Remarks:
Gets the untouch timeout data for the given touchscreen.

Requirements:
Included in Version 4.20 and later.
Header: Declared in EloInterface.h; include EloInterface.h.
Library: Use EloPubIf.lib.

See Also: EloGetScreenInfo, EloSetUntouchTimeout, Sample and Error Codes

Function Name: EloSetUntouchTimeout

int EloSetUntouchTimeout(PUNTOUCH_TIMEOUT pUTimeout, UINT nScrNo)

Parameters:
pUTimeout [in]: Pointer to UNTOUCH_TIMEOUT structure to containing the untouch timeout data for the touchscreen. Please see structure definition section for UNTOUCH_TIMEOUT.

nScrNo [in]: 0 based touchscreen number.

Return Values: Returns EloSuccess if the call succeeds, it returns an error code otherwise. See Error Codes section for list of error values.

Remarks:
Sets the untouch timeout for the given touchscreen.

Requirements:
Included in Version 4.20 and later.
Header: Declared in EloInterface.h; include EloInterface.h.
Library: Use EloPubIf.lib.

See Also: EloGetScreenInfo, EloGetUntouchTimeout, Sample and Error Codes

Function Name: EloGetRightClickOnHold

int EloGetRightClickOnHold(PRIGHT_BUTTON pRBHold, UINT nScrNo)

Parameters:
pRBHold [out]: Pointer to RIGHT_BUTTON structure to receive the right button on hold data for the touchscreen. The application is responsible for allocating / freeing memory. Please see structure definition section for RIGHT_BUTTON.

nScrNo [in]: 0 based touchscreen number.

Return Values: Returns EloSuccess if the call succeeds, it returns an error code otherwise. See Error Codes section for list of error values.

Remarks:
Gets the right button on hold data for the given touchscreen.

Requirements:
Included in Version 4.20 and later.
Header: Declared in EloInterface.h; include EloInterface.h.
Library: Use EloPubIf.lib.

See Also: EloGetScreenInfo, EloSetRightClickOnHold, Sample and Error Codes

Function Name: EloSetRightClickOnHold

int EloSetRightClickOnHold(PRIGHT_BUTTON pRBHold, UINT nScrNo)

Parameters:
pRBHold [in]: Pointer to RIGHT_BUTTON structure to containing the right button on hold data for the touchscreen. Please see structure definition section for RIGHT_BUTTON.

nScrNo [in]: 0 based touchscreen number.

Return Values: Returns EloSuccess if the call succeeds, it returns an error code otherwise. See Error Codes section for list of error values.

Remarks:
Sets the right button on hold for the given touchscreen.

Requirements:
Included in Version 4.20 and later.
Header: Declared in EloInterface.h; include EloInterface.h.
Library: Use EloPubIf.lib.

See Also: EloGetScreenInfo, EloGetRightClickOnHold, Sample and Error Codes

Interface Data Structures

Structure Name: SCREEN_PROPERTIES

The SCREEN_PROPERTIES structure contains the diagnostics data for the touchscreen.

```
typedef struct _SCREEN_PROPERTIES
{

    int iWindowsMonNo ;
    ULONG Type;
    char Port[256];
    char SerialNumber[18];
    DWORD HardwareHandshaking ;
    CONTRL_STAT ctrl_status;
    LONG BaudRate;
    char crevminor;
    char crevmajor;
    char trevminor;
    char trevmajor;
    char diagcodes[8];
    char id[8];
    char cnt_id[8];
    char driver_id[32];

}SCREEN_PROPERTIES, *LPSCREEN_PROPERTIES ;
```

Members

iWindowsMonNo: Windows monitor number associated with the touchscreen.

Type: Defines the type of touchscreen. Expected values are

USB: 0x01
PNP_SERIAL: 0x02
NT_SERIAL: 0x04
LEGACY_SERIAL: 0x08
RESERVED: 0xFF

Port:The serial port on which this serial touchscreen device is connected. This is blank for USB.

SerialNumber: Serial number for connected touchscreen. Serial numbers uniquely identify a touchscreen.

HardwareHandshaking: This is used only for serial touchscreens. If it is set to 1 hardware handshaking is turned on else if it is set to 0 it is turned off.

ctrl_status: Controller status returned at the time of diagnostics. Please see constants definition for CONTRL_STAT.

BaudRate: This specifies the baud rate for the serial port, valid only for serial

touchscreens.

crevminor: Minor revision of controller.

crevmajor: Major revision of controller.

trevminor: Unused.

trevmajor: Unused.

diagcodes: The response received for the diagnostics smartest command sent to the controller.

id: Elo OEM identification returned from the controller.

cnt_id: Contains the response to controller id smartest command to the controller.

driver_id: Driver identification / version string.

Structure Name: TOUCH_POINT

The TOUCH_POINT structure defines a touch coordinate.

```
typedef struct TOUCH_POINT
{
    LONG x;
    LONG y;
    LONG z;
    GETPOINTS_STATUS Status ;
    }TOUCH_POINT, *PTOUCH_POINT ;
```

Members

x: x co-ordinate for touch.

y: y co-ordinate for touch.

z: z co-ordinate for touch.

Status: Receives the touch status. Please see constants definition for GETPOINTS_STATUS.

Structure Name: CALIBRATION

The CALIBRATION structure defines the calibration data used to convert touches from the Elo coordinate system to Windows virtual coordinate system.

The driver uses the following equation to convert a raw touch coordinate point from Elo coordinate system to the Windows screen coordinate system. Calibration equation is

Xcal = a + m*Xuncal,

where,
Xuncal, is in Elo coordinate system
Xcal, is in Windows coordinate system
m = nScrDx / nEloDx
"a", is the X offset value entry
nScrDx = distance between targets in Windows virtual coordinates
nEloDx = distance between targets in Elo coordinates.

```
typedef struct _Calibration
{

    LONG VDeskMode ;
    LONG nScrDx ;
    LONG nEloDx ;
    LONG nOffsetX ;
    LONG nScrDy ;
    LONG nEloDy ;
    LONG nOffsetY ;
    LONG xyswap;
    LONG MonitorNumber ;
    RegistryOperation CalMode ;
    LONG xRes ;
    LONG yRes ;
    LONG xVirtScrSize ;
    LONG yVirtScrSize ;
    LONG xVirtScrCoord ;
    LONG yVirtScrCoord ;
    LONG xMonLocn ;
    LONG yMonLocn ;

} CALIBRATION, *PCALIBRATION
```

Members
VDeskMode: Reserved.

nScrDx:

nEloDx:
nOffsetX:
nScrDy:
nEloDy:
nOffsetY:
Data required by calibration. Use the equation above to calculate these values.

xyswap: Specifies if the touchscreen is rotated 90 degrees or 270 degrees. Set to 1 if touchscreen is rotated 90 degrees or 270 degrees else set to 0.

MonitorNumber: Windows monitor numbers are 1 based.

CalMode: Set to TempWrite if the value does not have to be saved over system reboot. For values to be saved over system reboot set this value to Write. Please see constants definition for RegistryOperation.

xRes: Screen width in pixels.

yRes: Screen height in pixels.

xMonLocn: Monitor location (X) for this monitor on the virtual desktop in pixels.

yMonLocn: Monitor location (Y) for this monitor on the virtual desktop in pixels.

xVirtScrSize: Width in pixels of the windows virtual screen.

yVirtScrSize: Height in pixels of the windows virtual screen.

xVirtScrCoord: Top left X coordinate of the Windows virtual screen.

yVirtScrCoord: Top left Y coordinate of the Windows virtual screen.

Structure Name: DRAG_DELAY

The DRAG_DELAY structure is used to get or set drag delay.

```
typedef struct _Drag_Delay{

    DWORD MinDragDelay ;

    DWORD MaxDragDelay ;

    DWORD DragDelay ;
```

} DRAG_DELAY, *PDRAG_DELAY ;

Members
MinDragDelay:
MaxDragDelay:
Reserved.

DragDelay: Drag delay value in milliseconds.

Structure Name: BEEP

The BEEP structure is used to get or set beep parameter data.

```
typedef struct _Beep{

    BOOL BeepFlag;

    DWORD BeepFreq ;

    DWORD BeepTime ;

} SOUND, *PBEEP ;
```

Members
BeepFlag: If TRUE touchscreen beeps on touch.

BeepFreq: Frequency of beep in Hz.

BeepTime: Duration of beep in milliseconds.

Structure Name: TOUCH_BOUNDARY

The TOUCH_BOUNDARY structure defines the full screen bounding rectangle and bounding mode for the touchscreen.

```
typedef struct _Touch_Boundry

{

    ClippingBounds Bounds;

    LONG ClippingMode;

} TOUCH_BOUNDARY, *PTOUCH_BOUNDARY ;
```

Members
Bounds: Defines the touch bounding rectangle for the screen. See structure ClippingBounds. For interpretation of these bounds see ClippingMode below.

ClippingMode: Defines the mode for the touchscreen boundary on the virtual desktop. Possible values are as follows:
0: Disable virtual desktop. Touch on all screens will reflect on the primary monitor.
1: Enable virtual desktop. In this mode no individual screen bounds are used for the monitor. Touches toward the edge may generate clicks on adjacent monitors.
2: Enable virtual desktop. Screen bounds are enabled for each monitor. Touches on the screen always generate clicks on the associated monitor. A touch outside the bounding rectangle will cause the cursor to move to a point along the boundary that is nearest to the point of touch.
3: Enable virtual desktop. Screen bounds are enabled for each monitor. Touches on the screen always generate clicks on the associated monitor. Touches outside the bounding rectangle are not sent to the system.

Structure Name: QUICK_TOUCH

The QUICK_TOUCH data structure defines the Quick touch configuration parameters.

```
typedef struct _Quick_Touch

{

    DWORD bEnable ;

    ULONG Dx ;

    ULONG Dy ;

} QUICK_TOUCH, *PQUICK_TOUCH;
```

Members
bEnable: 0 disables quick touch , 1 enables it.

Dx: X distance in pixels. Touches outside this distance will generate quick touches.

Dy: Y distance in pixels. Touches outside this distance will generate quick touches.

Structure Name: ClippingBounds

The ClippingBounds structure is used to define the bounding rectangle.

```
typedef struct _ClippingBounds{

    ULONG X_Max;

    ULONG X_Min;

    ULONG Y_Max;

    ULONG Y_Min;;

    ULONG Z_Max ;

    ULONG Z_Min ;

} ClippingBounds, *PClippingBounds ;
```

Members

X_Min:
Specifies the X coordinate of the upper-left corner of the rectangle in pixels.

Y_Min:
Specifies the Y coordinate of the upper-left corner of the rectangle in pixels.

X_Max:
Specifies the X coordinate of the lower-right corner of the rectangle in pixels.

Y_Max:
Specifies the Y coordinate of the lower-right corner of the rectangle in pixels.

Z_Max:
Z_Min:
Reserved.

Structure Name: EDGE_ACCEL

The EDGE_ACCEL structure is used to define the bounding rectangle.

```
typedef struct _ Accel{

    ULONG Enable ;
```

ULONG Scale ;

ClippingBounds Bounds[1] ;

} EDGE_ACCEL, *PEDGE_ACCEL ;

Members
Enable: 0 disables acceleration, 1 enables it.

Scale: Acceleration scale be anywhere in the range of 0-100. 10 indicates no acceleration.

Bounds: Defines the edge of the touchscreen acceleration bounds. See structure ClippingBounds.

Structure Name: RIGHT_BUTTON

The RIGH_BUTTON structure is used to define the right click on hold feature.

typedef struct _ Right_Button{

DWORD RightClickHW ;

style='color:black;text-decoration: none;text-underline:none'>DWORD InitialTimeout ;

style='color:windowtext; text-decoration:none;text-underline:none'>DWORD DefaultRightClickDelay ;

DWORD MaxRightClickDelay ;

<DWORD MinRightClickDelay ;

ULONG ClickCount;

ULONG Active ;

} RIGHT_BUTTON, *PRIGHT_BUTTON;

Members
RightClickHW: Height and width in pixels defining the area in which right click is valid.

InitialTimeout: The initial timeout after which the cursor changes to the right click cursor.

DefaultRightClickDelay: Right click delay value in milliseconds.

MinRightClickDelay:
MaxRightClickDelay:
Reserved.

ClickCount: Touch count for which the right click on hold feature should be active. A value of 0xffffffff keeps it active for ever. A value of 0 turns it off.

Active: Value of 1 keeps right click on hold active on reboot. Value of 0 turns it off on reboot. This works in combination with ClickCount. To keep it turned on after reboot value for Active should be set to 1 and value for ClickCount should be set to 0xffffffff.

Structure Name: UNTOUCH_TIMEOUT

The UNTOUCH_TIMEOUT structure is used to define the untouch timeout.

```
typedef struct _UNTOUCH_TIMEOUT{

    ULONG Timeout ;

    ULONG Width ;

    ULONG Height ;

} UNTOUCH_TIMEOUT, *PUNTOUCH_TIMEOUT ;
```

Members

Timeout: Untouch timeout value in milliseconds.

Width: Specifies the width in pixels for the screen area within which a constant touch will automatically result in untouch after the Timeout occurs.

Height: Specifies the height in pixels for the screen area within which a constant touch will automatically result in untouch after the Timeout occurs.

Constants: GETPOINTS_CODE

The blocked call returns data depending on this value.

```
typedef enum _GETPOINTS_CODE

{

    ReturnImmediately=1,

    ReturnOnTouch,

    ReturnOnUntouch,

    ReturnOnNextValidTouch

}GETPOINTS_CODE ;
```

Values
ReturnImmediately: Returns immediately with the last touch data values. Does not wait for a user to touch the screen.

ReturnOnTouch: Waits for user to touch and then returns data.

ReturnOnUntouch: Waits for user to release touch and then returns data.

ReturnOnNextValidTouch: Waits for the user to touch the screen and returns on the next initial touch, stream touch or untouch event.

Constants: CONTRL_STAT

Controller status is returned as

```
typedef enum _CONTRL_STAT

{

    CS_OK = 0,

    CS_ConstantTouch,

    CS_CanNotFindController,

    CS_NoResponse,

    CS_InvalidResponse,
```

CS_CanNotSetBaudRate,

CS_CommandNotSent,

CS_SystemError,

CS_InvalidCommPort,

CS_CommPortFailedOpen,

CS_CommPortCommandError,

CS_CommPortNoController,

CS_UndefinedController

} CONTRL_STAT;

Values
CS_OK: Everything works fine.

CS_ConstantTouch: There is constant touch detected on the touchscreen.

CS_CanNotFindController: No controller found.

CS_NoResponse: No response from controller.

CS_InvalidResponse: Incorrect response, maybe due to out of sync.

CS_CanNotSetBaudRate: Baud rate cannot be set.

CS_CommandNotSent: Smartset command could not be sent to the controller.

CS_SystemError: System Error.

CS_InvalidCommPort: No touchscreen connected on the serial port specified.

CS_CommPortFailedOpen: Error opening serial port.

CS_CommPortCommandError: Communications error.

CS_CommPortNoController: No controller.

CS_UndefinedController: Unidentified controller.

Constants: RegistryOperation

Driver writes the values to the registry if Write flag is set else it initializes the local buffer and does not save it to the registry.

```
typedef enum

{

    TempWrite=1,

    Write

}RegistryOperation;
```

Values

TempWrite: Writes data to local buffer. Does not write it to the registry.

Write: Writes data to local buffer and registry.

Constants: TOUCHSCREEN_MODE

Valid TOUCHSCREEN_MODE for the touchscreen.

```
#define CLICK_ON_TOUCH          0

#define CLICK_ON_RELEASE        1

#define MOUSE_EMULATION         6
```

Values

CLICK_ON_TOUCH: Click on touch sends immediately upon touch a mouse down/up message at the point of touch on the touchscreen. The user's finger must be removed from the touchscreen before a new touch at any location will be recognized. The cursor or selected objects CANNOT be "dragged" on the screen in this mode.

CLICK_ON_RELEASE: Click on release sends at the time of release (untouch) a mouse down/up message at the point that the screen was last touched. Dragging across objects on the screen will not highlight or select them unless untouch occurs when the touch is over the object. (Drag and Double-click)

MOUSE_EMULATION: Sends a mouse down message at the point of contact. Selects an object if it was at the initial point of contact. Drags a selected object on the screen. Sends a mouse up message at the point of untouch. Double-clicks on an object when the screen is touched twice in succession at the same location.

Constants: GETPOINTS_STATUS

Valid mode for the touchscreen.

```
typedef enum _GETPOINTS_STATUS
    InitialTouch =          1,
    StreamTouch=            2,
    UnTouch =               4
GETPOINTS_STATUS;
```

Values

InitialTouch: The touch data was returned on initial touch.

StreamTouch: The touch data was returned on stream touch.

UnTouch: The touch data was returned on untouch.

Error Codes: Returned from the Interface DLL

#define EloSuccess	0
#define EloFailure	1
#define EloErrorInIoctl	2
#define EloErrorCallWasCancelled	3
#define EloErrorInvalidTouchScreen	4
#define EloErrorInvalidMonitorNo	5
#define EloErrorInvalidBuffer	6
#define EloErrorInvalidHandle	7
#define EloErrorNoDriver	8
#define EloErrorInvalidCommand	10

#define EloErrorVersionLock	11
#define EloErrorBufferSize	12
#define EloErrorTouchDeviceNotWorking	13
#define EloErrorTouchScreenNumberInvalid	14
#define EloErrorNoMoreTouchScreens	15
#define EloErrorInvalidScreenNumber	16
#define EloErrorUnknown	17
#define EloErrorNotImplemented	18
#define EloErrorDeviceBusy	19
#define EloErrorIncompatibleVersion	20

Home Page

Top of page

Appendix C
Enable/Disable Sample Code

```
// EnableDisable.cpp : Defines the entry point for the sample EnableDisable
// touch application.
//
// Copyright(c) 2004  Elo TouchSystems, Inc., all rights reserved.
//
// Sample application to Enable/Disable touch using Elo SDK.
// Include EloPubIf.lib in the linker dependencies.
// Files EloPubIf.lib, EloInterface.h, EloErrorCodes.h can be found in
// \Program Files\EloTouchSystems folder.
// This program can be compiled using Microsoft Visual Studio 7.0 or 6.0
//
// Usage: EnableDisableSample -[e/d] -n:<mon num>[,<mon num>...]
//                         -e                                      : Enable touch
//                         -d                                      : Disable Touch
//                         -n:<mon num>[,<mon num>...]          : Monitors to enable or disable
// ----------------------------------------------------------------------

#include <windows.h>
#include <stdio.h>
#include <tchar.h>
#include <stdlib.h>
#include "EloInterface.h"
#include "EloErrorCodes.h"

// ----------------------------------------------------------------------
// Globals
// ----------------------------------------------------------------------

BOOL bEnable = TRUE;
int dwMonParamCnt=0;
DWORD dwMonParam[MAX_SUPPORTED_SCR] ;

void ProcessCmdLine( int argc, char** argv) ;

// ----------------------------------------------------------------------
// main entry point
int _tmain(int argc, char** argv)
{
```

```
DWORD dwEnumMon[MAX_SUPPORTED_SCR] ;
int iScrCnt, iRet ;

if( argc < 3 )
{
	printf( "Usage: EloEnableDisableSample -[e/d] \
			-[n:<mon num>,[,<mon num>...]]\n\n \
			-e \t\t\t\t: Enable touch\n \
			-d \t\t\t\t: Disable Touch\n \
			-n:<mon num>[,<mon num>...] \
			\t: Monitors to enable or disable \n" ) ;
	return EloFailure ;
}

ZeroMemory( dwEnumMon, MAX_SUPPORTED_SCR ) ;
ZeroMemory( dwMonParam, MAX_SUPPORTED_SCR ) ;

// Get the list of all Elo Serial & USB screen and monitor association
iRet = EloGetScreenInfo(dwEnumMon,iScrCnt) ;
if(iRet != EloSuccess )
{
	printf( "Error Code = %d \n", iRet ) ;
	return EloFailure;
}
else
if(iScrCnt<0)
{
	printf( "No Elo touchscreens found\n" ) ;
	return EloFailure;
}

// Process Commandline
ProcessCmdLine( argc, argv ) ;

// For all screens of matching monitor number enable / disable touch
for( int i=0; i<dwMonParamCnt; i++ )
{
	// where j is the screen number associated with the monitor number
	for( int j=0; j<iScrCnt; j++ )
	{
		if( dwMonParam[i] == dwEnumMon[j] )
		{
			// Enables / Disables touch depending on the bFlag
```

```
                                // where j is the screen number
                                if( (iRet = EloSetTouchReportingState( bEnable , j )) ==
EloSuccess )
                                        printf( "EloSetTouchReportingState Returned success
\n" );
                                else
                                        printf( "EloSetTouchReportingState Returned failed.
Error \ Code=%d\n", iRet );
                        }
                }
        }

   return EloSuccess;
}

// ----------------------------------------------------------------------
// Processes command line options
void ProcessCmdLine( int argc, char** argv)
{
        char *lpArg, *lpArg2 ;
        char argstr[256]="";
        LPTSTR lpCmdLine = argstr ;
        strcpy( argstr,"" ) ;
        for( int i=0; i<argc ; i++ )
        {
                strcat( argstr," " ) ;
                strcat( argstr, argv[i] ) ;
        }

        lpArg = strstr(lpCmdLine,"-") ;

        while (lpArg != NULL)
        {
                switch (lpArg[1])
                {
                        case 'e':
                                // Enables / Disables touch depending on the Flag
                                // TRUE to enable , FALSE to disable
                                bEnable = TRUE;
                                break;
                        case 'd':
                                // Enables / Disables touch depending on the Flag
                                // TRUE to enable , FALSE to disable
```

```
                        bEnable = FALSE;
                        break;
                case 'n':
                        // monitor number
                        if (lpArg[2] == ':')
                                lpArg++ ;
                        lpArg2 = strstr(lpArg+1,"-") ;
                        if (lpArg2 != NULL)
                                lpArg2-- ;

                        dwMonParam[dwMonParamCnt] = atoi(lpArg+2) ;
                        if (dwMonParam[dwMonParamCnt] > 0)
                                dwMonParamCnt++ ;
                        lpArg = strstr(lpArg+1,",") ;

                        while(lpArg != 0)
                        {
                                if(lpArg2 && (lpArg >= lpArg2)) break ;
                                dwMonParam[dwMonParamCnt] = atoi(lpArg+1) ;
                                if (dwMonParam[dwMonParamCnt] > 0)
                                        dwMonParamCnt++ ;
                                lpArg = strstr(lpArg+1,",") ;
                        }
                        lpArg = lpArg2 ;
                        break ;
        default:
                printf( "Usage: EloEnableDisableSample -[e/d] \
                        -[n:<mon num>,[,<mon num>...]]\n\n \
                        -e \t\t\t\t: Enable touch\n \
                        -d \t\t\t\t: Disable Touch\n \
                        -n:<mon num>[,<mon num>...] \
                        \t: Monitors to enable or disable \n" ) ;
                exit(0);
                break ;

        }
        if (lpArg != 0)
                lpArg = strstr(lpArg+1,"-") ;
    }
}

// ----------------------------------------EOF-----------------------------
```

Home Page

Top of page

Contacting Elo

Elo has technical support offices around the world. By telephone, e-mail or fax, you can find an office that is open and staffed with personnel to assist you with questions or problems with Elo products. Consult the list below for the office which can best serve you.

Americas

North America

U.S.A	Tel: 1-800-557-1458 (toll free) Fax: 865-694-1731 Online form: request technical support

Latin America

English-speaking customers	Tel: 1-800-557-1458 (**toll free**) Fax: 865-694-1731 Online form: request technical support
Portuguese-speaking customers	Tel: (55 11) 3611-1311, ext 249 or 144 Fax: (55 11) 3611-4365 E-mail: elotechbr@elotouch.com
Spanish-speaking customers	Tel: 54 11 - 47332238 Fax: 54 11 - 47332245 Online form: request technical support

Asia-Pacific

Japan	Tel: (81) (0) 45-478-2166 Fax: (81) (0) 45-478-2181 E-mail: info@tps.co.jp
Taiwan	Tel: 886 2 26629788, ext 416 Fax: 886 2 26642725 E-mail: eric.chang@tycoelectronics.com

Europe (including Africa/Middle East)

Africa
Belgium
Greece
Italy
Luxemburg
Netherlands
Portugal
Spain

Tel: +32 (0)16 35 19 01
Fax: +32 (0)16 35 21 01
E-mail: elotecheu@elotouch.com

Austria
Czech Republic
Germany
Hungary
Kroatia
Poland
Romania
Russia
Slovakia
Slovenia
Switzerland

Tel: +49 (0)89 60 822 193
Fax: +49 (0)89 60 822 180
E-mail: elotecheu@elotouch.com

Denmark
Finland
Iceland
Israel
Ireland
Norway
Sweden
United Kingdom

Tel: +44 (0)1793 57 33 46
Fax: +44 (0)1793 57 33 45
E-mail: elotecheu@elotouch.com

France

Tel: 0825 825 497 (toll free)
E-mail: elotecheu@elotouch.com

Home Page Top of page