Integrating Ambient Light Sensors with Windows 7 Computers

New Opportunities and Considerations for Interfacing Ambient Light Sensors with Windows

February 19, 2009

Abstract

This paper provides information about new ambient light sensor (ALS) features in the Windows® 7 family of operating systems. It provides guidelines for hardware manufacturers and original equipment manufacturers (OEMs) to integrate ALS hardware with computers that use the adaptive brightness feature and with applications that use light sensor data.

Currently, integrating and supporting ALSs for automatic brightness control in Windows are difficult and costly because of the required implementation of multiple components and drivers. Additionally, these ALSs are limited in scope to brightness control functionality.

In Windows 7, ALSs and adaptive display brightness are supported end to end. The goals for adaptive brightness are to provide an optimal user experience for readability, legibility, and eye comfort, while at the same time to optimize power consumption and battery life.

In addition to the adaptive brightness feature, applications can also use light sensor data to optimize their content for readability in many different lighting conditions. This application integration is done by using the Windows Sensor and Location Platform, which is new for Windows 7.

The information in this paper applies to the Windows 7 operating system.

References and resources discussed here are listed at the end of this paper.

For the latest information, see:
 [http://www.microsoft.com/whdc/device/sensors/default.mspx](http://www.microsoft.com/whdc/device/sensors/default.mspx%20)

Disclaimer: This is a preliminary document and may be changed substantially prior to final commercial release of the software described herein.

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This White Paper is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS DOCUMENT.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, email address, logo, person, place or event is intended or should be inferred.

© 2009 Microsoft Corporation. All rights reserved.

Microsoft, Windows, and Windows NT are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

Revision History

| **Date** | **Change** |
| --- | --- |
| February 18, 2009 | Bring into WHDC template; copy edit; no information changed |
| October 27, 2008 | First publication |

Contents

[Light Sensor Features 4](#_Toc222813646)

[Automatic Screen Brightness 4](#_Toc222813647)

[Automatic Keyboard Illumination 4](#_Toc222813648)

[Light-Aware Applications 4](#_Toc222813649)

[Light-Specific Themes and Actions 5](#_Toc222813650)

[Using the Windows Sensor and Location Platform 5](#_Toc222813651)

[Ambient Light Sensor Integration Checklist 5](#_Toc222813652)

[Understanding Adaptive Brightness Support in Windows 7 6](#_Toc222813653)

[Drivers for ACPI Integrated Ambient Light Sensors 6](#_Toc222813654)

[Sensor Monitor Windows NT Service 6](#_Toc222813655)

[Power Setting 7](#_Toc222813656)

[Registry Configuration Parameters 7](#_Toc222813657)

[Types of Light Sensors 7](#_Toc222813658)

[Integrating Light Sensors with Computer Hardware 8](#_Toc222813659)

[Number of Light Sensors 9](#_Toc222813660)

[Light Sensor Placement 9](#_Toc222813661)

[Light Sensor Filters, Lenses, Enclosures, and Calibration 10](#_Toc222813662)

[ACPI Integration 10](#_Toc222813663)

[Configuring Sensor Permissions 10](#_Toc222813664)

[Optimizing and Integrating LCD Displays 11](#_Toc222813665)

[Windows Display Driver Model Support 11](#_Toc222813666)

[Steps and Transitions 11](#_Toc222813667)

[Tuning and Configuring Adaptive Brightness Behavior 11](#_Toc222813668)

[Constructing an ALR Curve from User Preference Data 13](#_Toc222813669)

[Exposing ALR Curve Data through the Device Driver 15](#_Toc222813670)

[Overriding ALR Data with the ALR Registry Setting 16](#_Toc222813671)

[Illuminance Change Sensitivity Registry Setting 16](#_Toc222813672)

[Display Response Interval Registry Setting 17](#_Toc222813673)

[Conclusion 17](#_Toc222813674)

[Resources 17](#_Toc222813675)

# Light Sensor Features

Ambient light sensors (ALSs) are used to measure current lighting conditions. When a computer is aware of the surrounding lighting conditions, it can do several things with this information. Light sensors support the following:

* Automatic screen brightness.
* Automatic keyboard illumination.
* Light-aware applications.
* Light-specific themes and actions.

## Automatic Screen Brightness

What a user perceives as optimal screen brightness on a laptop that has an LCD screen is a function of the current lighting conditions and other factors, such as how the user’s eyes adjust to lighting conditions over time. In low-light conditions, such as in a dark room, the optimal screen brightness is a lower value. As surrounding lighting levels increase, the optimal screen brightness also increases. In varying lighting conditions, users prefer the computer screen to automatically adjust its own screen brightness.

In addition to the viewing experience, an automatic display brightness control can increase battery life. In Windows® 7, this is known as “adaptive brightness.” By dimming the display when it is appropriate, a computer can reduce power consumption.

## Automatic Keyboard Illumination

Computer keyboards are typically not usable in dark lighting conditions. To resolve this problem, either top illumination or reverse illumination can help users see the keyboard when they type. To optimize power consumption, hardware manufacturers can use ALSs to control the intensity of the illumination based on the lighting conditions.

## Light-Aware Applications

Typically, laptops are not usable in direct sunlight because of the limited capability of the screen to maintain contrast in bright light. If applications respond to current lighting conditions, they can alter their user information (UI) content to maximize contrast and legibility. Combining applications with adaptive brightness can make computers more usable in adverse lighting conditions.

For more information about how to build light-aware applications, see “Implementing Light-Aware UI Using the Windows Sensor and Location Platform” on the WHDC [Windows Sensor and Location Platform page](http://www.microsoft.com/whdc/device/sensors/default.mspx).

## Light-Specific Themes and Actions

These light sensor features are based on practical issues and needs. You can also add some fun to lighting conditions by applying application or system themes that are based on the lighting. These features are implemented by using the Sensor API. Examples of these features include changing the desktop background image and changing sound themes based on the current lighting conditions.

# Using the Windows Sensor and Location Platform

Often sensor technology is not adopted because of a lack of standards and developer resources for interacting with sensor devices on Windows.

Windows 7 addresses this problem by introducing a comprehensive new platform for sensor and location devices that is called the *Windows Sensor and Location Platform*. Windows 7 provides a common driver model, API, permissions model, and configuration UI for interacting with sensor and location devices.

For a more detailed introduction to the Windows Sensor and Location Platform, see “Introducing the Windows Sensor and Location Platform” on the WHDC [Windows Sensor and Location Platform page](http://www.microsoft.com/whdc/device/sensors/default.mspx).

# Ambient Light Sensor Integration Checklist

The following is a high-level checklist for developers for integrating sensor hardware into computers. The rest of this document describes the process and background information in detail.

* Select a suitable light sensor.
* Select optimized placement for the light sensor in your computer enclosure.
* Perform integration and calibration with the embedded controller firmware.
* Use Advanced Configuration and Power Interface (ACPI) BIOS firmware that exposes the ALS hardware so that it complies with the ACPI 3.0b specification. The firmware must enumerate the light sensor with Plug and Play ID ACPI0008, support the \_ALI method with notifications, and support the \_ALR method.
* Construct an ambient light response (ALR) curve that is based on relevant user preference data. This step is critical because this data has the largest influence on the overall user experience for adaptive brightness.
* Integrate light sensor and adaptive brightness configuration into the computer imaging and preconfiguration process. Use the SensorsConfig.exe sensor configuration tool to configure sensor permissions. Set registry configuration parameters for ALR curve data and for display response interval and Illuminance change sensitivity.
* Make sure that the display implementation complies and is optimized. Specifically, do the following:
* Make sure that the brightness control is exposed by using Windows Display Driver Model (WDDM). Implement \_BCL and \_BCM ACPI brightness methods or WDDM brightness control in the graphics adapter.
* Make sure that transitions are smooth for display brightness level changes and implement these changes in hardware.
* Make sure that sufficient display brightness levels are exposed. We suggest the maximum supported ACPI count of 0 to 100 percent in 1‑percent increments, for a total of 101 total levels.
* Test light sensors and adaptive brightness by doing the following:
* Make sure that the SensrSvc service is started.
* Validate that display brightness changes when the lighting changes.
* Test behavior with users to validate ALR data.
* If you have implemented a sensor driver to support non-ACPI ALSs, make sure that your driver passes the applicable logo tests (INPUT-0048, INPUT‑0049, and INPUT-050). For more information about these requirements, see LogoPoint.

# Understanding Adaptive Brightness Support in Windows 7

In Windows 7, automatic brightness control with ALSs (adaptive brightness) is fully supported in the box. In many ways, the screen brightness experience is enhanced for Windows 7, and adaptive brightness is an important aspect of these changes.

The following components for adaptive brightness are included in Windows 7:

* Drivers for ALSs that are exposed through ACPI and comply with the ACPI 3.0b specification.
* A sensor monitoring Windows NT® service that consumes light sensor data and then controls the display brightness.
* A power setting exposed in the Control Panel Advanced Power Configuration application that consumers use to enable or disable adaptive brightness.
* Adaptive brightness registry configuration parameters.

To better understand these components, the following examines how they support adaptive brightness.

## Drivers for ACPI Integrated Ambient Light Sensors

Windows 7 now features class driver support for ALS implementations that comply with ACPI 3.0b. This is good news for OEMs and independent hardware vendors (IHVs) that would otherwise be required to write custom drivers for their hardware to support ALSs. In addition to directly supporting the adaptive brightness feature, these drivers also support client applications that access light sensors by using the Sensor API, because the drivers integrate with the Windows Sensor and Location Platform.

## Sensor Monitor Windows NT Service

In Windows 7, brightness control with ALSs is handled in a Windows NT service. This service starts on demand when ALS hardware is present and therefore does not impose overhead on systems that do not have ALS hardware. This start optimization is accomplished through the new service control manager (SCM) for Windows 7. To control the screen brightness level, this Windows NT service connects to ALSs that are built into the computer, processes data from the sensors, and then adjusts the brightness accordingly. If adaptive brightness is disabled through the power setting, this service stops automatically.

## Power Setting

To let the user turn on or turn off adaptive brightness, a power setting for adaptive brightness is exposed in the Control Panel Advanced Power Configuration application. For each power plan, two settings exist for adaptive brightness: one for AC power and one for DC power. These settings have both an on and an off state.

## Registry Configuration Parameters

Windows 7 provides the following registry parameters that you can use to fine-tune the behavior of the adaptive brightness feature:

* **An ALR curve**, which is a collection of data that determines how the brightness of the screen changes, based on lighting conditions and the user's screen brightness preference.
* **Display response interval**, which specifies how frequently adaptive brightness can change the screen brightness. By varying this parameter, you can alter the overall responsiveness for more subtle or more responsive behavior.
* **Illuminance change sensitivity**, which establishes a threshold where brightness must change by at least the value that was specified before screen brightness changes occur. You can also use this parameter to control how subtle or responsive the user perceives the adaptive brightness feature.

For more information about these parameters, see “Tuning and Configuring Adaptive Brightness Behavior” later in this document.

# Types of Light Sensors

ALSs come in two fundamental types:

* **Analog light sensors** are connected to an embedded controller with an A/D converter, and they require firmware that can accurately interpret the light sensor data and compensate for various conditions and phenomena that affect readings. Some examples of these phenomena include infrared light rejection and light frequency compensation. For example, fluorescent lights vary in intensity with the frequency of the AC power that is supplied to the fixture. Analog sensors are typically very inexpensive.
* **Digital light sensors** are more expensive than analog sensors but have advantages over analog light sensors. These sensors can automatically compensate for various conditions and phenomena. Digital light sensors are also extremely compact and can be more expensive than analog sensors.

Regardless of which type of light sensor you choose, take care to ensure that accurate readings are taken and exposed to the system.

# Integrating Light Sensors with Computer Hardware

Selecting a suitable ALS device is critical. Several things can greatly affect what can be done with the information that the light sensors provide. These considerations include the following:

* The type of sensor (analog versus digital).
* The cost per unit.
* The accuracy and resolution of the sensor.
* The dynamic range of the sensor.
* Supported bus technology (digital only).
* The sampling rate (digital only).
* Power consumption.
* Packaging and placement options.

Although these are all important considerations, two factors warrant special consideration:

* Accuracy and resolution

To deliver an optimal user experience for adaptive brightness and light-aware UI in applications, accurate sensor data is required as input. Generally, the more accurate the sensor is, the better the corresponding user experience will be. A good goal for actual calibrated values that an ALS exposes is a consistent accuracy of within 10 to 20 percent of actual lighting conditions.

* Dynamic range

The dynamic range of a light sensor limits the environments in which it can be used. Because applications can be light-aware in Windows, outdoor lighting conditions are an important consideration.

Table 1. Common Lighting Conditions and Approximate Illuminance Values

| **Lighting condition** | **Illuminance (lux)** |
| --- | --- |
| Pitch black | 1          |
| Very dark | 10          |
| Dark indoors | 50          |
| Dim indoors | 100          |
| Normal indoors | 300          |
| Bright indoors | 1,000          |
| Dim outdoors | 5,000          |
| Cloudy outdoors | 30,000          |
| Direct sunlight | 100,000          |

Based on these values, we recommend a dynamic range of at least 10,000 to 40,000 lux. Display brightness control is generally performed in the lower ranges (such as up to 1,000 to 2,000 lux) where light-aware applications optimize their content for outdoor viewability in the higher ranges (such as 5,000 to 50,000 lux). With digital light sensors, you can typically change the dynamic range by sending commands to the sensor, which alter the analog gain.

## Number of Light Sensors

Generally, the more ALSs that are available to measure a lighting condition, the better the estimate of the actual illuminance. As a practical matter, each light sensor adds cost and uses space on the surface of the computer.

A minimal implementation for ALS hardware is a single sensor. Although this functions sometimes, regardless of where the sensor is positioned it can be obscured by the user’s hand, a shadow, or other obstructions. Adding at least one additional sensor can give a much better approximation of the actual lighting conditions. This is especially true when applications apply heuristics to the data from multiple light sensors to arrive at a “best approximation” of the actual lighting conditions. This can range from simple averaging of values to taking the highest reading—assuming obstruction of the sensor instead of point illumination such as glare.

## Light Sensor Placement

Correct placement of light sensors is another important aspect of good system design. The light sensors should be located on the same plane as the display (facing the user). This is necessary because the sensor is measuring the lighting conditions of the screen itself to support adaptive brightness and light-aware applications that display their content on the screen.

Also, avoid placing the light sensor in areas of the computer that are likely to be obscured from the light source or sources by shadows.

Figure 1 is an example user scenario in which a direct light source is behind the user. A shadow is cast over the lower half of the screen and the base of the computer. This scenario shows the fact that you should place light sensors near the top of the screen and facing the user.



Figure 1: Optimal light sensor placement

## Light Sensor Filters, Lenses, Enclosures, and Calibration

Usually, light sensors are integrated into the computer and include a filter or lens, which changes the response of the ALS. You should carefully calibrate light sensors in the computer enclosure so that they provide accurate data. This calibration is typically performed in the embedded controller that is used to interface with the light sensor.

## ACPI Integration

Correctly integrating the ALSs with ACPI ensures that in-box drivers load and work with the adaptive brightness feature.

For a sensor to function, it must meet the following high-level requirements:

* The \_ALI (Illuminance) method is implemented, with support for notifications.
* The \_ALR (light response curve) method is implemented, but notifications are not required.
* The device uses a Plug and Play ID ACPI0008.

After the light sensor meets these criteria, it loads the ACPI ALS in-box drivers and the adaptive brightness feature functions. Table 2 shows the ACPI 3.0b methods and corresponding support that the in-box ACPI light sensor drivers in Windows implements.

Table 2. ACPI 3.0b Methods and Corresponding Support

| **Method** | **GetData() supported** | **OnDataUpdated supported (notifications)** | **Change sensitivity supported** |
| --- | --- | --- | --- |
| **\_ALI** | x | x | x |
| **\_ALR** | x |   |   |
| **\_ALC** | x | x |   |
| **\_ALT** | x | x |   |
| **\_ALP** |   |   |   |

Be aware that the change sensitivity value is measured in percentage change, not in lux change.

## Configuring Sensor Permissions

Sensors can expose sensitive data, such as the user’s current location and health-related data. Because of this, sensors are disabled by default. Applications can request permissions to sensors, and the user can navigate to the Control Panel Location and Other Sensors application to view and configure sensor permissions. In some cases, the user should not be required to grant permissions. ALSs, for example, do not usually expose sensitive data about the user, so they should always be enabled.

You can preconfigure sensor permissions by using the SensorsConfig.exe sensor configuration tool. For the adaptive brightness feature to work, ALSs that are used with adaptive brightness must have permissions configured.

For more information about the sensor configuration tool, see the WHDC [Windows Sensor and Location Platform page](http://www.microsoft.com/whdc/device/sensors/default.mspx).

# Optimizing and Integrating LCD Displays

You can optimize and integrate LCD displays in two ways:

* WDDM support.
* Steps and transitions.

## Windows Display Driver Model Support

For the adaptive brightness feature to work, brightness control must be supported by using WDDM. The following are two options for WDDM brightness control:

* Implement the \_BCL and \_BCM ACPI brightness methods in the ACPI BIOS firmware.
* Implement support for WDDM brightness control in the graphics adapter.

For more information about brightness control in WDDM, see “Brightness Control in WDDM” on the WHDC Web site.

## Steps and Transitions

To create a fluid user experience, the display should smoothly transition from one brightness setting to another. You should implement transitions in hardware by ramping display brightness over a set of levels from the original value to the desired value over time when display brightness levels are set.

The number of brightness levels that a display device exposes is important. ACPI supports 101 levels of brightness for displays (from 0 to 100 percent in 1-percent increments). We recommend exposing the full range. If light sensors control the screen brightness or if the user controls screen brightness with a slider, you can obtain optimal brightness by using all available steps. If the user manipulates brightness by using function keys (incrementally raising or lowering the brightness levels), the screen brightness level is stepped up or down automatically in reasonable increments so that the number of required key presses is minimal. This manual stepping behavior is implemented in the Windows brightness control methods.

# Tuning and Configuring Adaptive Brightness Behavior

**Before you read this section, it would be helpful to read the appropriate ALS section in the latest ACPI specification. The following format for ALR curves is derived from the current 3.0b specification, section 9.2.**

You can collect data to determine optimal screen brightness in several ways. This data can reflect the user’s brightness preference in many different lighting conditions, or it can be calculated and measured based on required power consumption and corresponding battery life. In either case, an optimal user experience should be the result, based on the needs of the intended user.

In the following examples, we base our data and measurements on an optimal user experience on both screen readability and eye comfort.

Screen brightness data varies, depending on the characteristics of the LCD panel. Depending on the output (maximum brightness) of the display, the user brightness preference (measured as a percentage of maximum brightness) can vary significantly. Consider a portable computer that has a 200-nit (cd/m2 or candela-per-square-meter) display. In a particular lighting condition, the user’s brightness preference is 80 percent. If the laptop has a 400-nit display, in the same lighting conditions the same user would have a brightness preference of about 40 percent. When other display characteristics such as reflectivity, resolution, and overall size are considered, it is obvious that optimal screen brightness data must be collected for each type of LCD panel.

The following data was recorded with two identical laptops and involves many different users. We collected the data by asking users to read a document under controlled lighting conditions where the overall ambient light level was incrementally increased, with time allowed for the users’ eyes to adjust before each illuminance change.

Table 3: Example User Brightness Preference Data

| **Ambient light level (lux)** | **Optimal screen brightness(percent)** | **Optimal screen brightness(nits, or candelas per square meter)** |
| --- | --- | --- |
|  1 |  35 |  52.7548 |
|  9 |  42 |  63.3444 |
|  38 |  53 |  79.9852 |
|  65 |  71 | 107.2156 |
|  146 |  76 | 114.7796 |
|  296 |  85 | 128.3948 |
|  545 |  93 | 140.4972 |
|  924 |  98 | 148.0612 |
|  1,254 | 100 | 151.0868 |
|  1,700 | 100 | 151.0868 |
|  1,860 | 100 | 151.0868 |

**Optimum Brightness:**

**Illuminance (lux) vs. Screen Brightness**

Figure 2. Nonlinear nature of the illuminance-to-screen brightness data

The trend line shows a mathematical representation of this data. In this case, a logarithmic equation best fits the data. This lets us calculate the optimal screen brightness for any arbitrary lighting condition.

## Constructing an ALR Curve from User Preference Data

The goal of this exercise is to arrive at an ALR curve that the adaptive brightness feature uses to maintain optimal screen brightness based on the current lighting conditions.

The ALR curve is a set of points that map illuminance—measured in lux—to screen brightness offsets that are based on the baseline, expressed as a positive integer. For more information about ALR curves, see the ACPI 3.0b specification, section 9.2.

Because ACPI does not support signed types, offsets are all represented as positive integers. The actual brightness is calculated based on the user’s baseline brightness preference—which corresponds to favorable lighting conditions such as typical indoor lighting—and an offset that is calculated based on lighting conditions. The ALR curve supplies this offset.

The following formula calculates actual (corrected) screen brightness based on baseline and ALR data:

Where:

Table 4. Example ALR Curve Data

| **Offset** | **Illuminance** | **Description** |
| --- | --- | --- |
| 40     | 5     |  -60% adjust at 5 lux |
| 60     | 50     |  -40% adjust at 50 lux |
| 80     | 100     |  -20% adjust at 100 lux |
| 100     | 300     |  0% adjust at 300 lux |
| 120     | 700     |  +20% adjust at 700 lux |
| 150     | 1500     |  + 50% adjust at 1500 lux |

The following is an example calculation of actual screen brightness based on a baseline screen brightness of 80 percent, an illuminance of 50 lux, and an offset of 60:

For more information about how ALR data is represented, see the ACPI specification.

Now that we have some data to work with—our actual user brightness preference data—and we know what the ALR data we must produce looks like, we can create an ALR curve that is based on our example data.

There is no specific requirement for the number of points that we include in the ALR dataset, as long as there at least two. Perhaps the best way to approach this problem is to take evenly spaced brightness intervals (such as by 10 increments) and to calculate ALR data for each interval. That way, regardless of the actual screen brightness, the adaptive brightness feature can use accurate ALR data to control screen brightness.

First, select a baseline brightness preference that corresponds to favorable indoor lighting conditions. From our data, we will assume that we have determined that ~300 lux and ~85-percent screen brightness are good baselines because they represent typical indoor working conditions. Based on this baseline and on the screen brightness levels that the display supports, we can calculate the corresponding ALR offset values with the following equation:

Where:

 (%)

Next, use the equation that we obtained from the trend line in our chart, which is useful only for the example dataset:

Where:

This equation provides preferred screen brightness with lighting conditions as input. Now, take the fixed screen brightness values as input and determine the lighting values that correspond to these brightness levels. To do this, take the example equation and solve it for illuminance as a function of preferred brightness, which yieldsthe following equation and is useful only for the example dataset:

Where:

Now that we have equations to calculate the ALR offset values and illuminance values for the display’s supported brightness levels based on the baseline data, we can calculate the required values for an ALR curve.

Table 5. Calculated ALR Curve Data

| **Actual brightness (%)** | **ALR offset** | **Illuminance (lux)** |
| --- | --- | --- |
|  10 |  11.8 | 0.18         |
|  20 |  23.5 | 0.49         |
|  30 |  35.3 | 1.34         |
|  40 |  47.1 | 3.68         |
|  50 |  58.8 | 10.07         |
|  60 |  70.6 | 27.57         |
|  70 |  82.4 | 75.44         |
|  80 |  94.1 | 206.48         |
|  90 | 105.9 | 565.10         |
| 100 | 117.6 | 1,546.61         |

After we examine the calculated values, it is apparent that the first two rows of data are not meaningful because they correspond to lux values that are less than 1.0, so these rows are omitted from the final dataset. All these values are also rounded to the nearest unsigned integer value because that is how they are represented in actual ALR curve data.

If we graph the calculated values in Table 5 and compare them to the original user brightness preference data, we can see that the values align well with the preference data.

Figure 3. Preference data compared to ALR data

Now that the ALR dataset is calculated and validated, we can discover how this data and other parameters are exposed on the systems on which the adaptive brightness feature is ultimately configured.

## Exposing ALR Curve Data through the Device Driver

You can supply the ALR curve through the sensor driver and expose it by using the following property:

SENSOR\_PROPERTY\_LIGHT\_RESPONSE\_CURVE

This property is exposed as a vector of unsigned integers. For an example of format, see “Overriding ALR Data with the ALR Registry Setting” next in this paper.

For more information about how sensor device drivers expose properties, see the Windows 7 Windows Driver Kit (WDK).

If you correctly implement the \_ALR method in your ACPI BIOS, the in-box ACPI ALS drivers automatically expose the ALR curve data through this property, but you can override it in the registry if you want to.

Note that drivers that you write by using the Windows Sensor and Location Platform are subject to sensor logo tests. These tests are covered in LogoPoint under sections INPUT-0048, INPUT-0049, and INPUT-0050.

## Overriding ALR Data with the ALR Registry Setting

If the sensor driver does not expose ALR curve data or if the data is incorrect, you can override the ALR curve in the registry by setting the following value:

#### Key:

HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion
\AdaptiveDisplayBrightness

#### Value:

|  |  |
| --- | --- |
| ALRPoints (REG\_BINARY) | A binary block where pairs of unsigned 32-bit integers are stored in sequence.For the previous example, the following is stored in the following sequence: x1 y1 x2 y2 x3 y3 …70,0,73,10,85,80,100,300,150,1000 |

#### Example ALR registry data:

[HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\
AdaptiveDisplayBrightness]

"ALRPoints"=hex:00,00,00,00,46,00,00,00,0a,00,00,00,49,00,00,00,50,00,00,00,55,\

 00,00,00,2c,01,00,00,64,00,00,00,e8,03,00,00,96,00,00,00

## Illuminance Change Sensitivity Registry Setting

You can specify the sensitivity of the adaptive display brightness feature by the following value:

#### Key:

HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\
AdaptiveDisplayBrightness

#### Value:

|  |  |
| --- | --- |
| IlluminanceChangeSensitivity (REG\_DWORD) | The required percentage change in illuminance (lux) to cause a change in display brightness. It is specified in percentage change since the last display brightness change.Valid range: [1-100] |

## Display Response Interval Registry Setting

You can specify the desired display response interval for display brightness by the following value:

#### Key:

HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\
AdaptiveDisplayBrightness

#### Value:

|  |  |
| --- | --- |
| DisplayResponseInterval (REG\_DWORD) | The minimum time, in milliseconds, between changes in display brightness because of lighting conditions. Valid range: [100 - 3,600,000]\*\*3600000ms = 1 hour |

Correctly understanding and setting these adaptive brightness configuration parameters are critical to an optimal user experience. Because these parameters are specified in the registry, fine tuning and testing the behavior of the adaptive brightness feature are straightforward.

# Conclusion

If you integrate ALSs into your computer hardware, you can greatly improve the user experience. Light sensors in Windows 7 provide many opportunities for OEMS and hardware manufacturers. Adaptive brightness and light-aware applications represent just two examples. Correctly selecting and integrating light sensor hardware will help you take advantage of drivers and features that are now included in Windows.

# Resources

WHDC Windows Sensor and Location Platform Web:

[http://www.microsoft.com/whdc/device/sensors/default.mspx](http://www.microsoft.com/whdc/device/sensors/default.mspx%20)

Brightness Control in WDDM

<http://www.microsoft.com/whdc/device/display/aero/brightness.mspx>

ACPI ****3.0b specification****

<http://acpi.info>/