Logitech LCD SDK
for Microsoft® Windows®
(lglcd)
V1.03

Overview and Reference

© 2006 Logitech

Contents

Overview

What can I do with this library?

The lglcd library allows applications to interact with an LCD found on a Logitech product, such as the G15 G-series keyboard's LCD or the Z-10 speaker's LCD. Using this library, applications can display images and text on the LCD and read the LCD soft buttons. See the "Product Features" section for more details on the LCD and soft buttons.

What is included in this library?

The following files are included:

lglcd.h:
> C/C++ header file containing all the structure definitions and function prototypes for use with the lglcd.lib library.

lglcd.lib:
> Library to link to your executable.

The functionality contained in this library enables your application to connect to a Logitech software component called LCDMon that manages all attached LCDs, and regulates access to these LCDs by multiple applications. LCDMon ships as part of the G Series Keyboard Profiler software that customers will install when they set up their keyboard.
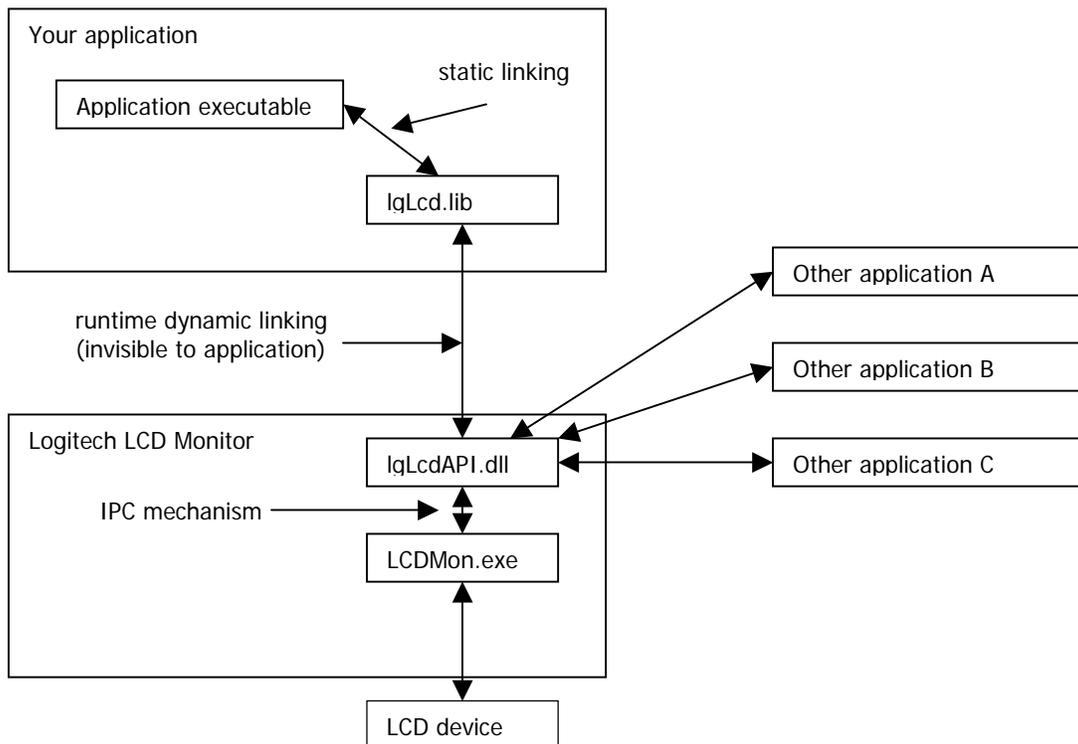
What do I need to use this library?

To use the library you need to have the following installed:

C/C++ development environment such as Microsoft® Visual Studio

How do I interface to this library?

You include the header file into your project and link with the appropriate library (pick from 32-bit i386 or 64-bit x86-64). You call the functions as detailed in this document and handle errors as they happen. There are essentially two errors that will happen all the time: either the user doesn't have the LCD software installed (because they have a different keyboard without an LCD), or the software is installed but the keyboard is not present. Both cases are very common and need to be dealt with gracefully. Additionally, there is the possibility of users disconnecting and re-connecting their keyboards, which should be handled as gracefully as possible.

Versioning

Versioning between the library and the LCD monitor portion is something that you don't need to worry about. More updated revisions of the monitor will support applications that have been linked with older libraries. Older revisions of the monitor will reject talking to newer libraries.

Synchronous vs. asynchronous operations

The lglcd library minimizes the time spent inside the library so making calls to the library will have minimal performance impact on your code. There is one function that can be determined to take place synchronously or asynchronously, namely, lgLcdUpdateBitmap(). This function takes a long time to execute (in the range of 30 milliseconds) and may be used as a "pacemaker" in applications that solely use the LCD. For all other applications, it is more desirable to have a quick as possible update, which is the asynchronous version. Should the application deliver more screens that can be written to the LCD, some will be dropped.

Sharing with other clients

Since the LCD is a shared resource, at any given time your application's bitmap may not be the one that is currently displayed. The user has the ability to switch among multiple clients writing to the LCD. None of the switching needs to be handled in your code. Your application gets its own "virtual LCD" and can continue to update screens and read soft buttons, completely unaware that it is not being displayed. Through a mechanism of priorities you can give a hint that your application has an "important" bitmap that might (depending on policy settings) allow your "important" bitmap to be displayed on the LCD.

Features of the Logitech G15 G-series keyboard and the Z-10 speaker LCD

Display properties

160 pixels horizontal, 43 pixels vertical, square pixel aspect ratio. Fully bitmapped display with 1 bit per pixel (monochrome). The maximum achievable frame rate on this display is around 30 frames per second, depending mainly on host processor load. Due to the nature of the display (passive-matrix display), rapidly moved images leave a blurry and unreadable image. In particular, text should not scroll too quickly.

The display is backlit and can be adjusted in brightness. The G15 supports adjustment of contrast by the user as well.

Buttons

The LCD has 4 "soft buttons" located below it which are unlabelled. The intent is to allow applications to use those buttons for whatever purpose they want to, much like a cell phone has labels above the respective controls. A suggested use would be "up/down" (or "previous/next") to select items or scroll, and an OK and Cancel functionality. Due to the real estate limitations on the display, these might be most efficiently represented with small icons above the buttons.

For multiple application situations, the application that currently owns the display will be able to read the buttons or get notifications on button changes. Once the application becomes paged into the background, it will not see any button presses anymore.

There is a fifth "dedicated" button which is not available to the applications. This button allows the user to navigate between multiple applications using the LCD, in a similar manner to the ALT-TAB mechanism on Windows®.

Picture

Image shown is a rendering of the display portion of the G15 G-series keyboard with the four soft buttons below it.

Dos and Don'ts of lglcd usage

These are a few guidelines that may help you implement better support in your application:

- Handle error conditions gracefully. There is a good chance that your application will not find a display, or not be able to connect to the LCD monitor at all. Not all of your customers will have a Logitech G Series Keyboard.

- Considering that this display is on a keyboard, handling unplug/replug scenarios might not seem as important. However, consider the case where such an auxiliary display is available stand-alone. Then, dynamic plugging/unplugging and possibly support of more than one display may become interesting.

- Think about what you want to present to the user and in which way. The constraints in terms of size and (in particular) the monochrome nature of the display make it a challenge to design nice-looking and informative screens.

- Remember that, while the refresh rate can reach 30 frames per second, the slow response time of the pixels causes some blur and unreadable text. Anything that is crucial to read should be moved slowly if at all. For scrolling text, it is advisable to make it "jumpier", i.e., scroll it by larger distances but less often. Also, don't rely on getting a solid 30 fps as this is dependent on environment influences (CPU usage and other things).

Reference

Structures

Functions

Structures

lgLcdDeviceDesc

The **lgLcdDeviceDesc** structure describes the properties of an attached device. This information is returned through a call to [lgLcdEnumerate]()().

```
typedef struct
{
    DWORD Width;
    DWORD Height;
    DWORD Bpp;
    DWORD NumSoftButtons;
} lgLcdDeviceDesc;
```

Members

Width

Specifies the width of the display in pixels.

Height

Specifies the height of the display in pixels.

Bpp

Specifies the depth of the bitmap in bits per pixel.

NumSoftButtons

Specifies the number of soft buttons that the device has.

Remarks

This structure is for compatibility with future devices. With the current hardware, only an LCD with 160x43x1 pixels and 4 soft buttons will be reported.

See Also

[lgLcdEnumerate]()()

lgLcdDeviceDescEx

The **lgLcdDeviceDescEx** structure describes the properties of an attached device. This information is returned through a call to lgLcdEnumerateEx().

```
typedef struct
{
    DWORD deviceFamilyId;
    TCHAR deviceDisplayName[MAX_PATH];
    DWORD Width;
    DWORD Height;
    DWORD Bpp;
    DWORD NumSoftButtons;
    DWORD Reserved1;
    DWORD Reserved2;
} lgLcdDeviceDescEx;
```

Members

deviceFamilyId

Specifies family Id of the device. Each device belongs to a family.

deviceDisplayName

Specifies the display name of the device.

Width

Specifies the width of the display in pixels.

Height

Specifies the height of the display in pixels.

Bpp

Specifies the depth of the bitmap in bits per pixel.

NumSoftButtons

Specifies the number of soft buttons that the device has.

Reserved1, Reserved2

Not used.

Remarks

This structure is used to return information about a device that is used by the LCD Manager.

See Also

[lgLcdEnumerateEx](
)

lgLcdBitmapHeader/lgLcdBitmap160x43x1

The **lgLcdBitmapHeader** exists at the beginning of any bitmap structure defined in lgLcd. Following the header is the actual bitmap as an array of bytes, as illustrated by **lgLcdBitmap160x43x1**.

```
typedef struct
{
    DWORD Format;
} lgLcdBitmapHeader;

typedef struct
{
    lgLcdBitmapHeader hdr;
    BYTE pixels[LGLCD_BMP_WIDTH*LGLCD_BMP_HEIGHT];
} lgLcdBitmap160x43x1;
```

Members

Format

Specifies the format of the structure following the header. Currently, only LGLCD_BMP_FORMAT_160x43x1 is supported.

pixels

Contains the display bitmap with 160x43 pixels. Every byte represents one pixel, with >=128 being "on" and <128 being "off".

Remarks

The array of pixels is organized as a rectangular area, 160 bytes wide and 43 bytes high. Despite the display being monochrome, 8 bits per pixel are used here for simple manipulation of individual pixels. To learn how to use GDI drawing functions efficiently with such an arrangement, see the sample code.

The pixels are arranged in the following order

byte 0 (0,0)	byte 1 (1,0)	byte 2 (2,0)	…	byte 157 (157,0)	byte 158 (158,0)	byte 159 (159,0)
byte 160 (0,1)	byte 161 (1,1)	byte 162 (2,1)	…	byte 317 (157,1)	byte 318 (158,1)	byte 319 (159,1)
…	…	…	…	…	…	…
byte 6560 (0,41)	byte 6561 (1,41)	byte 6562 (2,41)	…	byte 6717 (157,41)	byte 6718 (158,41)	byte 6719 (159,41)
byte 6720 (0,42)	byte 6721 (1,42)	byte 6722 (2,42)	…	byte 6877 (157,42)	byte 6878 (158,42)	byte 6879 (159,42)

See Also

[lgLcdUpdateBitmap]( )()

lgLcdConfigureContext/lgLcdOnConfigureCB

The **lgLcdConfigureContext** is part of the lgLcdConnectContext and is used to give the library enough information to allow the user to configure your application. The registered callback is called when the user clicks the "Configure…" button in the LCDMon list of applications.

```
// Callback used to allow client to pop up a "configuration panel"
typedef DWORD (WINAPI *lgLcdOnConfigureCB)(IN int connection,
                                            IN const PVOID pContext);


typedef struct
{
    // Set to NULL if not configurable
    lgLcdOnConfigureCB configCallback;
    PVOID configContext;
} lgLcdConfigureContext;
```

Members

configCallback

Specifies a pointer to a function that should be called when the user wants to configure your application. If no configuration panel is provided or needed, leave this parameter NULL.

configContext

Specifies an arbitrary context value of the application that is passed back to the client in the event that the registered configCallback function is invoked.

Remarks

For applications which don't have a stand-alone UI, such as a simple little utility that requires the LCD to be present, you can use the configuration callback mechanism to allow the user to access your UI through the common LCDMon front end. In LCDMon's list of applications, there is a button labeled "Configure…" which, when pressed, invokes the registered configuration callback. This saves you from having to implement any other UI (tray icon or regular window) if all the UI your application does is LCD related.

Note that this callback is executed in the context of a thread within the library; therefore take the necessary precautions for thread safety should the callback code share resources with other threads in your application.

See Also

lgLcdConnectContext, lgLcdConnect(), lgLcdConnectEx()

lgLcdNotificationContext/lgLcdOnNotificationCB

The **lgLcdNotificationContext** is part of the lgLcdConnectContextEx and is used to give the library enough information to allow the user to have notifications from the LCD Manager. The registered callback is called when there are any notifications for the application. Among notifications are arrivals and removals of new devices. This feature is not supported as of the 1.03 release of the LCD Manager software.

```
// Callback used to notify client of events from the LCD Manager

typedef DWORD (WINAPI *lgLcdOnNotificationCB)(IN int connection,
                                              IN const PVOID pContext,
                                              IN DWORD notificationCode,
                                              IN DWORD notifyParm1,
                                              IN DWORD notifyParm2,
                                              IN DWORD notifyParm3,
                                              IN DWORD notifyParm4);
typedef struct
{
    // Set to NULL if not configurable
    lgLcdOnNotificationCB notificationCallback;
    PVOID notificationContext;
} lgLcdNotificationContext;
```

Members

notificationCallback

Specifies a pointer to a function that should be called when LCD Manager has notifications for the application. If the application is not interested in receiving notifications, then it should set this parameter to NULL.

notificationContext

Specifies an arbitrary context value of the application that is passed back to the client in the event that the registered notificationCallback function is invoked.

Remarks

This feature is not implemented as of the 1.03 release of the LCD Manager.

See Also

lgLcdConnectContextEx, lgLcdConnectEx()

lgLcdConnectContext

The **lgLcdConnectContext** contains all the information that is needed to connect your application to LCDMon through lgLcdConnect(). Upon successful connection, it also contains the connection handle that has to be used in subsequent calls to lgLcdEnumerate() and lgLcdOpen().

```
typedef struct
{
    // "Friendly name" display in the listing
    LPCTSTR appFriendlyName;
    // isPersistent determines whether this connection
    // persists in the list
    BOOL isPersistent;
    // isAutostartable determines whether the client can be started by
    // LCDMon
    BOOL isAutostartable;
    lgLcdConfigureContext onConfigure;
    // --> Connection handle
    int connection;
} lgLcdConnectContext;
```

Members

appFriendlyName

Specifies a string that contains the "friendly name" of your application. This name is presented to the user whenever a list of applications is shown.

isPersistent

Specifies whether your connection is temporary (.isPersistent = FALSE) or whether it is a regular connection that should be added to the list (.isPersistent = TRUE).

isAutostartable

Specifies whether your application can be started by LCDMon or not.

onConfigure

Specifies context that is necessary to call back for configuration of your application. See lgLcdConfigureContext for more details.

connection

Upon successful connection, this member holds the "connection handle" which is used in subsequent calls to lgLcdEnumerate() and lgLcdOpen(). A value of LGLCD_INVALID_CONNECTION denotes an invalid connection.

Remarks

Typically, an application is either LCD-enabled (as one of the many features, such as for example a game) or an LCD-specific application (such as, for example, a clock that

shows on the LCD only). For an LCD-enabled type of application, it is common to have .isAutostartable set to FALSE and a configuration callback of NULL. For an LCD-specific example, .isAutostartable is usually TRUE and there can be a good reason for implementing the configuration callback (say, to switch between 12 and 24 hour format). It's up to the configuration callback to implement whatever UI is needed for that particular application.

Connections are kept open until lgLcdDisconnect() is called, or until the library is de-initialized with lgLcdDeInit().

The connection handle that is returned is used with lgLcdEnumerate() and lgLcdOpen(). To distinguish between valid and invalid connection handles, use the constant LGLCD_INVALID_CONNECTION.

A single application can open multiple connections to LCDMon. This can be useful if the client wants to appear with multiple separate entries in LCDMon's application list. For example, if your application hosts both a stock ticker display, as well as an e-mail polling application. In that case, your application would call lgLcdConnect() once with a friendly name of "Stock Ticker", and a second time with "E-mail poller". The two LCD clients, while sharing the same housing, are for all purposes separate (including separate configuration screens), and therefore require a separate connection.

See Also

lgLcdConnect(), lgLcdDisconnect(), lgLcdEnumerate(), lgLcdOpen()

lgLcdConnectContextEx

The **lgLcdConnectContextEx** contains all the information that is needed to connect your application to LCDMon through lgLcdConnectEx(). Upon successful connection, it also contains the connection handle that has to be used in subsequent calls to lgLcdEnumerate(), lgLcdEnumerateEx() and lgLcdOpen().

```
typedef struct
{
    // "Friendly name" display in the listing
    LPCTSTR appFriendlyName;
    // isPersistent determines whether this connection
    // persists in the list
    BOOL isPersistent;
    // isAutostartable determines whether the client can be started by
    // LCDMon
    BOOL isAutostartable;
    lgLcdConfigureContext onConfigure;
    // --> Connection handle
    int connection;
    DWORD dwAppletCapabilitiesSupported;
    DWORD dwReserved1;
    lgLcdNotificationContext onNotify;
} lgLcdConnectContextEx;
```

Members

appFriendlyName

Specifies a string that contains the "friendly name" of your application. This name is presented to the user whenever a list of applications is shown.

isPersistent

Specifies whether your connection is temporary (.isPersistent = FALSE) or whether it is a regular connection that should be added to the list (.isPersistent = TRUE).

isAutostartable

Specifies whether your application can be started by LCDMon or not.

onConfigure

Specifies context that is necessary to call back for configuration of your application. See lgLcdConfigureContext for more details.

connection

Upon successful connection, this member holds the "connection handle" which is used in subsequent calls to lgLcdEnumerate(), lgLcdEnumerateEx() and lgLcdOpen(). A value of LGLCD_INVALID_CONNECTION denotes an invalid connection.

dwAppletCapabilitiesSupported

This field contains an Or'd combination of the capabilities supported by the applet. This field is not yet implemented as of the 1.03 release of the LCD Manager software.

onNotify

Specifies context that is necessary to call back for notification to your application. See lgLcdNotificationContext for more details. This is not supported as of the 1.03 release fo the LCD Manager software.

Remarks

Typically, an application is either LCD-enabled (as one of the many features, such as for example a game) or an LCD-specific application (such as, for example, a clock that shows on the LCD only). For an LCD-enabled type of application, it is common to have .isAutostartable set to FALSE and a configuration callback of NULL. For an LCD-specific example, .isAutostartable is usually TRUE and there can be a good reason for implementing the configuration callback (say, to switch between 12 and 24 hour format). It's up to the configuration callback to implement whatever UI is needed for that particular application.

Connections are kept open until lgLcdDisconnect() is called, or until the library is de-initialized with lgLcdDeInit().

The connection handle that is returned is used with lgLcdEnumerate(), lgLcdEnumerateEx() and lgLcdOpen(). To distinguish between valid and invalid connection handles, use the constant LGLCD_INVALID_CONNECTION.

A single application can open multiple connections to LCDMon. This can be useful if the client wants to appear with multiple separate entries in LCDMon's application list. For example, if your application hosts both a stock ticker display, as well as an e-mail polling application. In that case, your application would call lgLcdConnectEx() once with a friendly name of "Stock Ticker", and a second time with "E-mail poller". The two LCD clients, while sharing the same housing, are for all purposes separate (including separate configuration screens), and therefore require a separate connection.

See Also

lgLcdConnectEx(), lgLcdDisconnect(),lgLcdEnumerate(),lgLcdEnumerateEx(), lgLcdOpen()

lgLcdSoftbuttonsChangedContext/lgLcdOnSoftButtonsCB

The **lgLcdSoftbuttonsChangedContext** is part of the lgLcdOpenContext and is used to give the library enough information to allow changes in the state of the soft buttons to be signaled into the calling application through a callback.

```
// Callback used to notify client of soft button change
typedef DWORD (WINAPI *lgLcdOnSoftButtonsCB)(IN int device,
                                             IN DWORD dwButtons,
                                             IN const PVOID pContext);

typedef struct
{
    // Set to NULL if no softbutton notifications are needed
    lgLcdOnSoftButtonsCB softbuttonsChangedCallback;
    PVOID softbuttonsChangedContext;
} lgLcdSoftbuttonsChangedContext;
```

Members

softButtonsChangedCallback

Specifies a pointer to a function that should be called when the state of the soft buttons changes. If no notification is needed, leave this parameter NULL.

softbuttonsChangedContext

Specifies an arbitrary context value of the application that is passed back to the client in the event that soft buttons are being pressed or released. The new value of the soft buttons is reported in the dwButtons parameter of the callback function.

Remarks

There are two methods of getting soft button states. One of them is using the polling method through lgLcdReadSoftButtons(); the second is by using this notification callback that alerts the application of any changes as they take place.

Note that this callback is executed in the context of a thread within the library; therefore take the necessary precautions for thread safety should the callback code share resources with other threads in your application.

See Also

lgLcdOpenContext, lgLcdOpen(), lgLcdReadSoftButtons()

lgLcdOpenContext

The **lgLcdOpenContext** contains all the information that is needed to open a specified LCD display through lgLcdOpen(). Upon successful completion of the open it contains the device handle that has to be used in subsequent calls to lgLcdReadSoftButtons(), lgLcdUpdateBitmap() and lgLcdClose().

```
typedef struct
{
    int connection;
    // Device index to open
    int index;
    lgLcdSoftbuttonsChangedContext onSoftbuttonsChanged;
    // --> Device handle
    int device;
} lgLcdOpenContext;
```

Members

connection

Specifies the connection (previously opened through lgLcdConnect()) which this lgLcdOpen() call is for.

index

Specifies the index of the device to open (see lgLcdEnumerate() or lgLcdEnumerateEx() for details).

onSoftbuttonsChanged

Specifies the details for the callback function that should be invoked when this device has changes in its soft button status, i.e. the user has pressed or released a soft button. For details, see lgLcdSoftbuttonsChangedContext.

device

Upon successful opening, this member holds the device handle which is used in subsequent calls to lgLcdReadSoftButtons(), lgLcdUpdateBitmap() and lgLcdClose(). A value of LGLCD_INVALID_DEVICE denotes an invalid device.

Remarks

Use this structure to open a device and retrieve its handle. The handle stays valid until the device is unplugged or lgLcdClose() is called. To distinguish between valid and invalid connection handles, use the constant LGLCD_INVALID_DEVICE.

For a given connection, multiple devices can be opened. While this might not make much sense for a keyboard (who will have two of those connected to the same system), it does provide the flexibility for having multiple standalone displays connected.

See Also

[lgLcdConnect](), [lgLcdEnumerate](),[lgLcdConnectEx](), [lgLcdEnumerateEx](), [lgLcdOpen]()

Functions

lgLcdInit

The **lgLcdInit**() function initializes the Logitech LCD library. You must call this function prior to any other function of the library.

```
DWORD WINAPI lgLcdInit(void);
```

Parameters

None

Return Values

If the function succeeds, the return value is ERROR_SUCCESS.

If the function fails, the return value can be one of the following:

Value	Meaning
RPC_S_SERVER_UNAVAILABLE	The Logitech LCD subsystem is not available (this is the case for systems that don't have the software installed)
ERROR_OLD_WIN_VERSION	Attempted to initialize for Windows 9x. The library only works on Windows 2000 and above.
ERROR_NO_SYSTEM_RESOURCES	Not enough system resources.
ERROR_ALREADY_INITIALIZED	lgLcdInit() has been called before.

Remarks

No other function in the library can be called before lgLcdInit() is executed. For result codes RPC_S_SERVER_UNAVAILABLE, ERROR_OLD_WIN_VERSION, and ERROR_NO_SYSTEM_RESOURCES, the calling application can safely assume that the machine it is running on is not set up to do LCD output and therefore disable its LCD-related functionality.

See Also

lgLcdDeInit()

lgLcdDeInit

Use **lgLcdDeInit()** after you are done using the library in order to release all resources that were allocated during lgLcdInit().

```
DWORD WINAPI lgLcdDeInit(void);
```

Parameters

None.

Return Values

If the function succeeds, the return value is ERROR_SUCCESS.

This function does not fail.

Remarks

All resources that were allocated during use of the library will be released when this function is called. After this function has been called, no further calls to the library are permitted with the exception of lgLcdInit().

See Also

lgLcdInit()

lgLcdConnect

Use **lgLcdConnect()** to establish a connection to the LCD monitor process. This connection is required for any other function to find, open and communicate with LCDs.

```
DWORD WINAPI lgLcdConnect(IN OUT lgLcdConnectContext *ctx);
```

Parameters

ctx

Pointer to a structure which holds all the relevant information about the connection which you wish to establish. Upon calling, all fields except the "connection" member need to be filled in; on return from the function, the "connection" member will be set. See lgLcdConnectContext for details.

Return Values

If the function succeeds, the return value is ERROR_SUCCESS.

If the function fails, the return value can be one of the following:

Value	Meaning
ERROR_SERVICE_NOT_ACTIVE	lgLcdInit() has not been called yet.
ERROR_INVALID_PARAMETER	Either ctx or ctx->appFriendlyName are NULL.
ERROR_FILE_NOT_FOUND	LCDMon is not running on the system.
ERROR_ALREADY_EXISTS	The same client is already connected.
RPC_X_WRONG_PIPE_VERSION	LCDMon does not understand the protocol.
Xxx	Other (system) error with appropriate error code.

Remarks

A connection needs to be established for an application to start using LCD devices. lgLcdConnect() attempts to establish that connection. If the LCD Monitor process is not running (either because it has not been started or not installed (the user is using a different keyboard)), the connection attempt will not succeed. In that case, your application should consider running without any LCD support.

Since a string is part of the connection context, this function exists in an ANSI and a UNICODE version. The header file picks the appropriate version depending on whether the symbol UNICODE is present or not.

See Also

lgLcdDisconnect(), lgLcdEnumerate(), lgLcdOpen()

lgLcdConnectEx

Use lgLcdConnectEx() to establish a connection to the LCD monitor process. This connection is required for any other function to find, open and communicate with LCDs.

```
DWORD WINAPI lgLcdConnectEx(IN OUT lgLcdConnectContextEx *ctx);
```

Parameters

ctx

Pointer to a structure which holds all the relevant information about the connection which you wish to establish. Upon calling, all fields except the "connection" member need to be filled in; on return from the function, the "connection" member will be set. See lgLcdConnectContext for details.

Return Values

If the function succeeds, the return value is ERROR_SUCCESS.

If the function fails, the return value can be one of the following:

Value	Meaning
ERROR_SERVICE_NOT_ACTIVE	lgLcdInit() has not been called yet.
ERROR_INVALID_PARAMETER	Either ctx or ctx->appFriendlyName are NULL.
ERROR_FILE_NOT_FOUND	LCDMon is not running on the system.
ERROR_ALREADY_EXISTS	The same client is already connected.
RPC_X_WRONG_PIPE_VERSION	LCDMon does not understand the protocol.
Xxx	Other (system) error with appropriate error code.

Remarks

A connection needs to be established for an application to start using LCD devices. lgLcdConnect() attempts to establish that connection. If the LCD Monitor process is not running (either because it has not been started or not installed (the user is using a different keyboard)), the connection attempt will not succeed. In that case, your application should consider running without any LCD support.

Since a string is part of the connection context, this function exists in an ANSI and a UNICODE version. The header file picks the appropriate version depending on whether the symbol UNICODE is present or not.

See Also

lgLcdDisconnect(), lgLcdEnumerate(), lgLcdEnumerateEx(), lgLcdOpen()

lgLcdDisconnect

Use **lgLcdDisconnect()** to close an existing connection to the LCD monitor process.

```
DWORD WINAPI lgLcdDisconnect(int connection);
```

Parameters

connection

> Specifies the connection handle that was returned from a previous successful call to lgLcdConnect() or lgLcdConnectEx().

Return Values

If the function succeeds, the return value is ERROR_SUCCESS.

If the function fails, the return value can be one of the following:

Value	Meaning
ERROR_SERVICE_NOT_ACTIVE	lgLcdInit() has not been called yet.
ERROR_INVALID_PARAMETER	Specified connection handle does not exist.
Xxx	Other (system) error with appropriate error code.

Remarks

Closing a connection invalidates all devices that have been opened using that connection. These invalid handles, if used after closing the connection, will cause errors to be returned by calls to lgLcdUpdateBitmap(), lgLcdReadSoftButtons(), and lgLcdClose().

Additionally, if a callback for configuration had been registered in the call to lgLcdConnect(), it will not be called anymore.

See Also

lgLcdConnect(), lgLcdConnectEx()

lgLcdSetDeviceFamiliesToUse

The **lgLcdSetDeviceFamiliestoUse**() function is used to inform the LCDManager of what device types (families are types) the applet is interested in using. After this call, all subsequent calls to lgLcdEnumerate() or lgLcdEnumerateEx() will only return devices that are in the families requested.

```
DWORD WINAPI lgLcdSetDeviceFamiliesToUse(IN int connection,
                                 IN DWORD dwDeviceFamiliesSupported,
                                 IN DWORD dwReserved1);
```

Parameters

connection

Specifies the connection that this command refers to.

dwDeviceFamiliesSupported

An or'ed list of device families the applet is interested in. Refer to lglcd.h for a complete listing of all supported device familes..

dwReserved1

Must be set to 0 for versions up to 1.03.

Return Values

If the function succeeds, the return value is ERROR_SUCCESS.

If the function fails, the return value can be one of the following:

Value	Meaning
ERROR_SERVICE_NOT_ACTIVE	lgLcdInit() has not been called yet.
ERROR_INVALID_PARAMETER	The dwDeviceFamiliesSupported is not valid.
Xxx	Other (system) error with appropriate error code.

Remarks

The connection parameter is returned by a call to lgLcdConnect() or lgLcdConnectEx().

Use this function prior to any enumeration of devices. This gives the applet the ability to only use devices that it is interested in and knows how to communicate with.

See Also

lgLcdConnect(), lgLcdConnectEx(), lgLcdConnectContext, lgLcdDeviceDesc, lgLcdDeviceDescEx(), lgLcdOpen(), lgLcdClose()

lgLcdEnumerate

The **lgLcdEnumerate**() function is used to retrieve information about all the currently attached and supported Logitech LCD devices.

```
DWORD WINAPI lgLcdEnumerate(IN int connection, IN int index,
                            OUT lgLcdDeviceDesc *description);
```

Parameters

connection

Specifies the connection that this enumeration refers to.

index

Specifies which device information is requested. See Remarks.

description

Points to an lgLcdDeviceDesc structure which will be filled with information about the device.

Return Values

If the function succeeds, the return value is ERROR_SUCCESS.

If the function fails, the return value can be one of the following:

Value	Meaning
ERROR_SERVICE_NOT_ACTIVE	lgLcdInit() has not been called yet.
ERROR_NO_MORE_ITEMS	There are no more devices to be enumerated. If this error is returned on the first call, then there are no devices attached.
ERROR_INVALID_PARAMETER	The description pointer is NULL.
Xxx	Other (system) error with appropriate error code.

Remarks

The connection parameter is returned by a call to lgLcdConnect() or lgLcdConnectEx().

To enumerate the attached devices, you should call lgLcdEnumerate() and pass in 0 as *index* parameter. On subsequent calls, increase the *index* parameter by 1 until the function returns ERROR_NO_MORE_ITEMS.

See Also

lgLcdConnect(), lgLcdConnectEx(), lgLcdConnectContext, lgLcdDeviceDesc, lgLcdOpen(), lgLcdClose()

lgLcdEnumerateEx

The **lgLcdEnumerateEx**() function is used to retrieve information about all the currently attached and supported Logitech LCD devices.

```
DWORD WINAPI lgLcdEnumerateEx(IN int connection, IN int index,
                              OUT lgLcdDeviceDescEx *description);
```

Parameters

connection

Specifies the connection that this enumeration refers to.

index

Specifies which device information is requested. See Remarks.

description

Points to an lgLcdDeviceDescEx structure which will be filled with information about the device.

Return Values

If the function succeeds, the return value is ERROR_SUCCESS.

If the function fails, the return value can be one of the following:

Value	Meaning
ERROR_SERVICE_NOT_ACTIVE	lgLcdInit() has not been called yet.
ERROR_NO_MORE_ITEMS	There are no more devices to be enumerated. If this error is returned on the first call, then there are no devices attached.
ERROR_INVALID_PARAMETER	The description pointer is NULL.
Xxx	Other (system) error with appropriate error code.

Remarks

The connection parameter is returned by a call to lgLcdConnect() or lgLcdConnectEx().

To enumerate the attached devices, you should call lgLcdEnumerateEx() and pass in 0 as *index* parameter. On subsequent calls, increase the *index* parameter by 1 until the function returns ERROR_NO_MORE_ITEMS.

See Also

lgLcdConnect(), lgLcdConnectEx(), lgLcdConnectContext, lgLcdDeviceDesc, lgLcdOpen(), lgLcdClose()

lgLcdOpen

The **lgLcdOpen**() function starts the communication with an attached device. You have to call this function before retrieving button information or updating LCD bitmaps.

```
DWORD WINAPI lgLcdOpen(IN OUT lgLcdOpenContext *ctx);
```

Parameters

ctx

Specifies a pointer to a structure with all the information that is needed to open the device. See lgLcdOpenContext for details. Before calling lgLcdOpen(), all fields must be set, except the "device" member. Upon successful return, the "device" member contains the device handle that can be used in subsequent calls to lgLcdUpdateBitmap(), lgLcdReadSoftButtons(), and lgLcdClose().

Return Values

If the function succeeds, the return value is ERROR_SUCCESS.

If the function fails, the return value can be one of the following:

Value	Meaning
ERROR_SERVICE_NOT_ACTIVE	lgLcdInit() has not been called yet.
ERROR_INVALID_PARAMETER	Either ctx is NULL, or ctx->connection is not valid, or ctx->index does not hold a valid device.
ERROR_ALREADY_EXISTS	The specified device has already been opened in the given connection.
Xxx	Other (system) error with appropriate error code.

Remarks

A handle retrieved through this function stays valid until either of the following conditions occurs:

- The device has been unplugged. Any operation with the given handle will result in an error code of ERROR_DEVICE_NOT_CONNECTED.

- The handle has been closed using lgLcdClose().

Part of the opening context is a callback that can be pointed to a function that is to be called when soft button changes take place on the LCD. This callback is issued when the LCD's soft buttons change while your application owns the LCD space. See the definition of lgLcdOpenContext and lgLcdSoftbuttonsChangedContext for details.

See Also

lgLcdOpenContext, lgLcdClose(), lgLcdEnumerate(), lgLcdEnumerateEx(), lgLcdUpdateBitmap(), lgLcdReadSoftButtons()

lgLcdClose

The **lgLcdClose**() function stops the communication with the previously opened device.

```
DWORD WINAPI lgLcdClose(IN int device);
```

Parameters

device

> Specifies the device handle retrieved in the lgLcdOpenContext by a previous call to lgLcdOpen().

Return Values

If the function succeeds, the return value is ERROR_SUCCESS.

If the function fails, the return value can be one of the following:

Value	Meaning
ERROR_SERVICE_NOT_ACTIVE	lgLcdInit() has not been called yet.
ERROR_INVALID_PARAMETER	The specified device handle is invalid.
Xxx	Other (system) error with appropriate error code.

Remarks

After calling lgLcdClose(), the soft button callback specified in the call to lgLcdOpen() will not be called anymore.

See Also

lgLcdOpen()

lgLcdReadSoftButtons

The **lgLcdReadSoftButtons**() function reads the current state of the soft buttons for the specified device.

```
DWORD WINAPI lgLcdReadSoftButtons(IN int device, OUT DWORD *buttons);
```

Parameters

device

Specifies the device handle for which to read the soft button state.

buttons

Specifies a pointer to a DWORD that receives the state of the soft buttons at the time of the call. See comments for details.

Return Values

If the function succeeds, the return value is ERROR_SUCCESS.

If the function fails, the return value can be one of the following:

Value	Meaning
ERROR_SERVICE_NOT_ACTIVE	lgLcdInit() has not been called yet.
ERROR_INVALID_PARAMETER	The specified device handle or the result pointer is invalid.
ERROR_DEVICE_NOT_CONNECTED	The specified device has been disconnected.
Xxx	Other (system) error with appropriate error code.

Remarks

The resulting DWORD contains the current state of the soft buttons, 1 bit per button. You can use the mask definitions LGLCDBUTTON_BUTTON0 through LGLCDBUTTON_BUTTON3 to check for any particular button in the DWORD.

If your application is not being currently displayed, you will receive a resulting "buttons" DWORD of 0, even if some soft buttons are being pressed. This is in order to avoid users inadvertently interacting with an application that does not presently show on the display.

See Also

lgLcdOpen()

lgLcdUpdateBitmap

The **lgLcdUpdateBitmap**() function updates the bitmap of the device.

```
DWORD WINAPI lgLcdUpdateBitmap(IN int device,
                               IN const lgLcdBitmapHeader *bitmap,
                               IN DWORD priority);
```

Parameters

device

Specifies the device handle for which to update the display.

bitmap

Specifies a pointer to a bitmap header structure. See comments for details.

priority

Specifies a priority hint for this screen update, as well as whether the update should take place synchronously or asynchronously. See comments for details.

The following priorities are defined:

Value	Meaning
LGLCD_PRIORITY_IDLE_NO_SHOW	Lowest priority, disable displaying. Use this priority when you don't have anything to show.
LGLCD_PRIORITY_BACKGROUND	Priority used for low priority items.
LGLCD_PRIORITY_NORMAL	Normal priority, to be used by most applications most of the time.
LGLCD_PRIORITY_ALERT	Highest priority. To be used only for critical screens, such as "your CPU temperature is too high"

In addition, there are three macros that can be used to indicate whether the screen should be updated synchronously (LGLCD_SYNC_UPDATE()) or asynchronously (LGLCD_ASYNC_UPDATE()). When using synchronous update the LCD library can notify the calling application of whether the bitmap was displayed or not on the LCD, using the macro (LGLCD_SYNC_COMPLETE_WITHIN_FRAME()). Use these macros to indicate the behavior of the library.

Return Values

If the function succeeds, the return value is ERROR_SUCCESS.

If the function fails, the return value can be one of the following:

Value	Meaning
ERROR_SERVICE_NOT_ACTIVE	lgLcdInit() has not been called yet.
ERROR_INVALID_PARAMETER	The specified device handle, the bitmap header

	pointer or the type of bitmap is invalid.
ERROR_DEVICE_NOT_CONNECTED	The specified device has been disconnected.
ERROR_ACCESS_DENIED	Synchronous operation was not displayed on the LCD within the frame interval (30 ms). This error code is only returned when the priority field of the lgLCDUpdateBitmap uses the macro LGLCD_SYNC_COMPLETE_WITHIN_FRAME().
Xxx	Other (system) error with appropriate error code.

Remarks

The bitmap header parameter should point to an actual bitmap. The current revision of the library defines a structure called lgLcdBitmap160x43x1 which holds as a first member a bitmap header. You would typically instantiate once of these structures, set the hdr.Format to LGLCD_BMP_FORMAT_160x43x1, then fill in the bitmap to be displayed in the pixels[] member. Finally, you call lgLcdUpdateBitmap(... &yourBitmap.hdr ...) to issue the bitmap update. Future versions of the SDK could have additional bitmap types declared, but all of them will have the same header at the beginning.

At any given time there may be multiple applications attempting to display a bitmap on the LCD. The priority parameter is a hint for LCDMon's display scheduling algorithm. In a scenario where there is contention for screen display time, LCDMon needs to determine which application's bitmap to display. In order to aid this scheduling, it can (but depending on user settings might not) take into account the hints that an application gives through the priority parameter. It is highly advisable that your application gives the appropriate priority for any given screen update to improve the user experience. A well-behaved LCD-enabled application should not use high priorities except for alerts.

The difference between asynchronous and synchronous updates is as follows: an asynchronous update will place the bitmap to be displayed into LCDMon and return immediately, before the bitmap is actually sent out to the device. For synchronous updates, the call to lgLcdUpdateBitmap() will only return after the bitmap has been sent out to the device, which takes 30 milliseconds or more. In case the application currently does not show on the LCD because another application is displayed, the synchronous update returns after a time that is similar to an update when the application is visible. If the macro LGLCD_SYNC_COMPLETE_WITHIN_FRAME() is used, an error is returned to the calling application when this condition arises.

The benefit of using the synchronous update is that your application will run "locked" with the LCD updates. It will be suspended for the entire duration of writing to the screen, and only get to run when the display is ready to accept a new screen. A "mini-game" on the LCD would profit from this behavior in order to get the highest possible frame rates while minimizing CPU usage.

The asynchronous updates are beneficial to applications that don't care about the exact sequence and timing of screen updates and have many other things to do. They just deposit a bitmap to be sent to the device every once in a while and don't worry about it actually going out and being in sync with this event.

See Also

lgLcdOpen(), lgLcdBitmapHeader, lgLcdBitmap160x43x1

lgLcdSetAsLCDForegroundApp

The **lgLcdSetAsLCDForegroundApp**() allows an application to become the one shown on the LCD and prevents the LCD library from switching to other applications, when the **foregroundYesNoFlag** parameter is set to **LGLCD_LCD_FOREGROUND_APP_YES**. When the calling application calls this function with **foregroundYesNoFlag** parameter set to **LGLCD_LCD_FOREGROUND_APP_NO**, the LCD library resumes its switching algorithm that the user had chosen.

```
DWORD WINAPI lgLcdSetAsLCDForegroundApp(IN int device,
                                        IN int foregroundYesNoFlag);
```

Parameters

device

Specifies the device handle for which the command is intented for.

foregroundYesNoFlag

Specifies whether the calling application is interested in becoming the frontmost application shown on the LCD or it is trying to remove itself from being the frontmost. See comments for details.

The following foregroundYesNoFlag values are defined:

Value	Meaning
LGLCD_LCD_FOREGROUND_APP_NO	Calling application does not want to be the only application shown on the LCD.
LGLCD_LCD_FOREGROUND_APP_YES	Calling application wants to be the only application shown on the LCD.

Return Values

If the function succeeds, the return value is ERROR_SUCCESS.

If the function fails, the return value can be one of the following:

Value	Meaning
ERROR_LOCK_FAILED	The operation could not be completed.
Xxx	Other (system) error with appropriate error code.

Remarks

An application such as a game that wants to be shown on the LCD and does not want to be swaped out by other applications, can use this call to become the frontmost application shown on the LCD. The LCD library will not swap out the application, except for other applications that call this function at a later date. The frontmost application's bitmaps supplied using the lgLcdUpdateBitmap() call will all be displayed on the LCD.

See Also

[lgLcdUpdateBitmap]()

License Agreement

End-User License Agreement for Logitech LCD SDK

This End-User License Agreement for Logitech LCD SDK ("Agreement") is a legal agreement between you, either an individual or legal entity ("You" or "you") and Logitech Inc. ("Logitech") for use of the Logitech LCD software development kit, which includes computer software and related media and documentation (hereinafter "LCD SDK"). By using this LCD SDK, you are agreeing to be bound by the terms and conditions of this Agreement. If you do not agree to the terms and conditions of this Agreement, promptly return the LCD SDK and other items that are part of this product in their original package with your sales receipt to your point of purchase for a full refund, or if you have downloaded this software from a Logitech web site, then you must stop using the software and destroy any copies of the software in your possession or control.

1   **Grant of License and Restrictions.** This Agreement grants You the following rights provided that You comply with all terms and conditions of this Agreement.

   (a)   Logitech grants You a limited, non-exclusive, nontransferable license to install and use an unlimited number of copies of the LCD SDK on computers . All other rights are reserved to Logitech.

   (b)   You shall not reverse engineer, decompile or disassemble any portion of the LCD SDK, except and only to the extent that this limitation is expressly prohibited by applicable law.

   (c)   At your option, you may provide reasonable feedback to Logitech, including but not limited to usability, bug reports and test results, with respect to the LCD SDK. All bug reports, test results and other feedback provided to Logitech by You shall be the property of Logitech and may be used by Logitech for any purpose.

   (d)   In the event Logitech, in its sole discretion, elects to provide copies of the LCD SDK to more than one individual employed by You (if You are not a single individual), each such individual shall be entitled to exercise the rights granted in this Agreement and shall be bound by the terms and conditions herein.

2   **Updates.** Logitech is not obligated to provide technical support or updates to You for the LCD SDK provided to You pursuant to this Agreement. However, Logitech may, in its sole discretion, provide further pre-release versions, technical support, updates and/or supplements ("Updates") to You, in which case such Updates shall be deemed to be included in the "LCD SDK" and shall be governed by this Agreement, unless other terms of use are provided in writing by Logitech with such Updates.

3   **Intellectual Property Rights.** The LCD SDK is licensed, not sold, to You for use only under the terms and conditions of this Agreement. Logitech and its suppliers retain title to the LCD SDK and all intellectual property rights therein. The LCD SDK is protected by intellectual property laws and international treaties, including U.S. copyright law and international copyright treaties. All rights not expressly granted by Logitech are reserved.

4   **Disclaimer of Warranty.** TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, LOGITECH AND ITS SUPPLIERS PROVIDE THE LCD SDK AND OTHER LOGITECH PRODUCTS AND SERVICES (IF ANY) AS IS AND WITHOUT WARRANTY OF ANY KIND. LOGITECH AND ITS SUPPLIERS EXPRESSLY DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD-PARTY RIGHTS WITH RESPECT TO THE LCD SDK AND ANY WARRANTIES OF NON-INTERFERENCE OR ACCURACY OF INFORMATIONAL CONTENT. NO LOGITECH DEALER, AGENT, OR EMPLOYEE IS AUTHORIZED TO MAKE ANY MODIFICATION, EXTENSION, OR ADDITION TO THIS WARRANTY. Some jurisdictions do not allow limitations on how long an implied warranty lasts, so the above limitation may not apply to you.

5   **Limitation of Liability.** IN NO EVENT WILL LOGITECH OR ITS SUPPLIERS BE LIABLE FOR ANY COSTS OF PROCUREMENT OF SUBSTITUTE PRODUCTS OR SERVICES, LOST PROFITS, LOSS OF INFORMATION OR DATA, OR ANY OTHER SPECIAL, INDIRECT, CONSEQUENTIAL, OR INCIDENTAL DAMAGES ARISING IN ANY WAY OUT OF THE SALE OF, USE OF, OR INABILITY TO USE THE LCD SDK OR ANY LOGITECH PRODUCT OR SERVICE, EVEN IF LOGITECH HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN NO

CASE SHALL LOGITECH'S AND ITS SUPPLIERS' TOTAL LIABILITY EXCEED THE ACTUAL

MONEY PAID FOR THE LOGITECH PRODUCT OR SERVICE GIVING RISE TO THE LIABILITY.

Some jurisdictions do not allow the exclusion or limitation of incidental or consequential damages,

so the above limitation or exclusion may not apply to you. The above limitations will not apply in

case of personal injury where and to the extent that applicable law requires such liability.

6   **U.S. Government Rights.** Use, duplication, or disclosure of the software contained in the LCD

SDK by the U.S. Government is subject to restrictions set forth in this Agreement and as provided

in DFARS 227.7202-1(a) and 227.7202-3(a) (1995), DFARS 252.227-7013(c)(1)(ii) (OCT 1988)

FAR 12.212(a) (1995), FAR 52.227-19, or FAR 52.227-14 (ALT III), as applicable. Logitech Inc.

6505 Kaiser Drive, Fremont, CA 94555.

7   **Export Law Assurances.** You agree and certify that neither the LCD SDK nor any other technical

data received from Logitech will be exported outside the United States except as authorized and as

permitted by the laws and regulations of the United States. If you have rightfully obtained the LCD

SDK outside of the United States, you agree that you will not re-export the LCD SDK nor any other

technical data received from Logitech, except as permitted by the laws and regulations of the

United States and the laws and regulations of the jurisdiction in which you obtained the LCD SDK.

8   **Termination:** This Agreement is effective until terminated. Upon any violation of any of the

provisions of this Agreement, rights to use the LCD SDK shall automatically terminate and the LCD

SDK must be returned to Logitech or all copies of the LCD SDK destroyed. You may also terminate

this Agreement at any time by destroying all copies of the LCD SDK in your possession or control.

If Logitech makes a request via public announcement or press release to stop using the copies of

the LCD SDK, you will comply immediately with this request. The provisions of paragraphs 3, 7, 8

and 12 will survive any termination of this Agreement.

9   **General Terms and Conditions.** If You are an individual signing this Agreement on behalf of a

company, then You represent that You have authority to execute this Agreement on behalf of such

company. This Agreement will be governed by and construed in accordance with the laws of the

United States and the State of California, without regard to or application of its choice of law rules

or principles. If for any reason a court of competent jurisdiction finds any provision of this

Agreement, or portion thereof, to be unenforceable, that provision of the Agreement shall be

enforced to the maximum extent permissible so as to affect the intent of the parties, and the

remainder of this Agreement shall continue in full force and effect. This Agreement constitutes the

entire agreement between You and Logitech respect to the use of the LCD SDK and supersedes all

prior or contemporaneous understandings, communications or agreements, written or oral,

regarding such subject matter.