Logitech Gaming LCD SDK
V1.01

Overview and Reference

Contents

Overview

The Logitech Gaming LCD SDK enables applications such as games to display game specific information on Logitech devices that have a monochrome or color LCD.

The main goal is to display useful information that is not already being displayed on the game screen. For example the game can display on the LCD information that usually is only accessing when hitting tab or when going to the game menu.

Another option is to remove information from the game screen (e.g. current score, timer, frame rate, etc), and move it to the LCD instead.

The SDK is a Windows based API for C/C++ programmers. Games based on the Microsoft Win32 API do not access hardware directly. Instead, the Logitech Gaming LCD SDK interacts with supported Logitech devices on behalf of the games.

SDK Package

The following files are included:

- LogitechLcd.h: C/C++ header file containing function prototypes
- LogitechLcd.lib: companion lib file to access DLL exported functions (32 and 64 bit)
- LogitechLcd.dll: library of SDK functions (32 and 64 bit)

Requirements

The Logitech Gaming LCD SDK can be used on the following platforms:

- Windows XP SP2 (32-bit and 64-bit)
- Windows Vista (32-bit and 64-bit)
- Windows 7 (32-bit and 64-bit)
- Windows 8 (32-bit and 64-bit)

The Logitech Gaming LCD SDK is a C based interface and is designed for use by C/C++ programmers. Familiarity with Windows programming is required.

Interfacing with the SDK

Using LogitechLcd.h and LogitechLcd.lib to access LogitechLcd.dll

The application can include LogitechLcd.h and link to LogitechLcd.lib (see "Sample usage of the SDK" further below or sample program in Samples folder). Installation folder for the DLL needs to be the same as the main executable, or needs to be part of the Path in the system environment.

Using LogitechLcd.dll directly

Alternatively the game can use the DLL directly by loading it via LoadLibrary, and accessing its functions using GetProcAddress (see "Sample usage of the SDK" further below or sample program in Samples folder).

LCD equipped Logitech Gaming devices

Color LCD, *resolution of 320 by 240 pixels, full RGBA.*

G19

Monochrome LCD, *resolution of 160 by 43 pixels*

G510

G13

G15 v1

G15 v2

Do's and Don'ts

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

- For color use a splash screen when the game starts up.
- For color have a nice background image to take full advantage of the RGBA LCD
- Don't just display information on the LCD that is already being displayed on the main game screen. Instead display information that is not readily available for the user to see. Instead display information he can only see when hitting tab or going to the menu.
- Use the LCD to unclutter the main screen. You can move information such as score, timer, framerate and even the minimap to the LCD.
- Write support for both the color and monochrome LCDs, as both have an important user base.
- Text displayed on the LCD is fixed-width, so you can easily create multiple columns that always align correctly.
- If you want to create custom screens, draw your own bitmaps and update the background LCD bitmap up to 60 times/second.
- Use the buttons to create multiple pages or add functionality to the LCD.
- If you are working with Unreal Engine and DLLBind, and want to set the background, use the UDK specific functions, don't use those in any other application.

Sample usage of the SDK

Using header and lib

```
#pragma comment(lib, "LogitechLcd.lib")

#include "LogitechLcd.h"

…
```

```
//The parameters indicate the name of the app and the type of the target display
LogiLcdInit(text, type);


…


//this is your main loop
while(!done){
	//Every frame of your main loop you have to call the update function
	LogiLcdUpdate();

	//You can now call any of the function defined in LogitechLcd.h header file

}

LogiLcdShutdown ();
```

Using DLL directly

```
// LCD types
#define LOGI_LCD_TYPE_MONO    (0x00000001)
#define LOGI_LCD_TYPE_COLOR   (0x00000002)

//LCD Monochrome buttons
#define LOGI_LCD_MONO_BUTTON_0 (0x00000001)
#define LOGI_LCD_MONO_BUTTON_1 (0x00000002)
#define LOGI_LCD_MONO_BUTTON_2 (0x00000004)
#define LOGI_LCD_MONO_BUTTON_3 (0x00000008)

//LCD Color buttons
#define LOGI_LCD_COLOR_BUTTON_LEFT   (0x00000100)
#define LOGI_LCD_COLOR_BUTTON_RIGHT  (0x00000200)
#define LOGI_LCD_COLOR_BUTTON_OK     (0x00000400)
#define LOGI_LCD_COLOR_BUTTON_CANCEL (0x00000800)
#define LOGI_LCD_COLOR_BUTTON_UP     (0x00001000)
#define LOGI_LCD_COLOR_BUTTON_DOWN   (0x00002000)
#define LOGI_LCD_COLOR_BUTTON_MENU   (0x00004000)

//LCD Monochrome size
const int LOGI_LCD_MONO_WIDTH = 160;
const int LOGI_LCD_MONO_HEIGHT = 43;

//LCD Color size
const int LOGI_LCD_COLOR_WIDTH = 320;
const int LOGI_LCD_COLOR_HEIGHT = 240;

typedef BOOL (* LPFNDLLINIT)(wchar_t*, int );
typedef BOOL (* LPFNDLLISCONNECTED)(int);
typedef BOOL (* LPFNDLLISBUTTONPRESSED)(int);
typedef void (* LPFNDLLUPDATE)();
typedef void (* LPFNDLLSHUTDOWN)();
typedef BOOL (* LPFNDLLMONOSETBACKGROUND)(BYTE*);
typedef BOOL (* LPFNDLLMONOSETTEXT)(int, wchar_t*);
typedef BOOL (* LPFNDLLCCOLORSETBACKGROUND)(BYTE*);
```

```
typedef BOOL (* LPFNDLLCOLORSETTITLE)(wchar_t*, int, int, int);
typedef BOOL (* LPFNDLLCOLORSETTEXT)(int, wchar_t*, int, int, int);

LPFNDLLINIT LogiLcdInit = NULL;
LPFNDLLISCONNECTED LogiLcdIsConnected = NULL;
LPFNDLLISBUTTONPRESSED LogiLcdIsButtonPressed = NULL;
LPFNDLLUPDATE LogiLcdUpdate = NULL;
LPFNDLLSHUTDOWN LogiLcdShutdown = NULL;
LPFNDLLMONOSETBACKGROUND LogiLcdMonoSetBackground = NULL;
LPFNDLLMONOSETTEXT LogiLcdMonoSetText = NULL;
LPFNDLLCCOLORSETBACKGROUND LogiLcdColorSetBackground = NULL;
LPFNDLLCOLORSETTITLE LogiLcdColorSetTitle = NULL;
LPFNDLLCOLORSETTEXT LogiLcdColorSetText = NULL;

…

HINSTANCE logiDllHandle = LoadLibrary(L"LogitechLcd.dll");

if (logiDllHandle != NULL)
    {
        LogiLcdInit = (LPFNDLLINIT)GetProcAddress(logiDllHandle, "LogiLcdInit");
        LogiLcdIsConnected = (LPFNDLLISCONNECTED)GetProcAddress(logiDllHandle,
"LogiLcdIsConnected");
        LogiLcdIsButtonPressed = (LPFNDLLISBUTTONPRESSED)GetProcAddress(logiDllHandle,
"LogiLcdIsButtonPressed");
        LogiLcdUpdate = (LPFNDLLUPDATE)GetProcAddress(logiDllHandle, "LogiLcdUpdate");
        LogiLcdShutdown = (LPFNDLLSHUTDOWN)GetProcAddress(logiDllHandle,
"LogiLcdShutdown");
        LogiLcdMonoSetBackground = (LPFNDLLMONOSETBACKGROUND)GetProcAddress(logiDllHandle,
"LogiLcdMonoSetBackground");
        LogiLcdMonoSetText = (LPFNDLLMONOSETTEXT)GetProcAddress(logiDllHandle,
"LogiLcdMonoSetText");
        LogiLcdColorSetBackground =
(LPFNDLLCCOLORSETBACKGROUND)GetProcAddress(logiDllHandle, "LogiLcdColorSetBackground");
        LogiLcdColorSetTitle = (LPFNDLLCOLORSETTITLE)GetProcAddress(logiDllHandle,
"LogiLcdColorSetTitle");
        LogiLcdColorSetText = (LPFNDLLCOLORSETTEXT)GetProcAddress(logiDllHandle,
"LogiLcdColorSetText");
       //The parameters indicate the name of the app and the type of the target display
        ret = LogiLcdInit(text, type);
    }

//this is your main loop
while(!done){
       //Every frame of your main loop you have to call the update function
       LogiLcdUpdate();

       //You can now call any of the function defined in LogitechLcd.h header file

}

LogiLcdShutdown ();
```

Reference

Generic Functions

LogiLcdInit

The **LogiLcdInit** () function makes necessary initializations. You must call this function prior to any other function in the library.

```
bool LogiLcdInit(wchar_t* friendlyName, int lcdType);
```

Parameters

- friendlyName : the name of your applet, you can't change it after initialization.
- lcdType : defines the type of your applet lcd target, it can be one of the following :
  - LOGI_LCD_TYPE_MONO
  - LOGI_LCD_TYPE_COLOR
  - If you want to initialize your applet for both LCD types just use
    LOGI_LCD_TYPE_MONO | LOGI_LCD_TYPE_COLOR

Return value

If the function succeeds, it returns true. Otherwise false.

LogiLcdIsConnected

The **LogiLcdIsConnected** () function checks if a device of the type specified by the parameter is connected.

```
bool LogiLcdIsConnected(int lcdType);
```

Parameters

- lcdType : defines the lcd type to look for, it can be one of the following :
  - LOGI_LCD_TYPE_MONO
  - LOGI_LCD_TYPE_COLOR
  - If you want to look for both LCD types just use
    LOGI_LCD_TYPE_MONO | LOGI_LCD_TYPE_COLOR

Return value

If a device supporting the lcd type specified is found, it returns true. If the device has not been found or the LogiLcdInit function has not been called before, returns false.

LogiLcdIsButtonPressed

The **LogiLcdIsButtonPressed** () function checks if the button specified by the parameter is being pressed.

```
bool LogiLcdIsButtonPressed(int button);
```

Parameters

- button : defines the button to check on, it can be one of the following :
  - LOGI_LCD_MONO_BUTTON_0
  - LOGI_LCD_MONO_BUTTON_1
  - LOGI_LCD_MONO_BUTTON_2

- LOGI_LCD_MONO_BUTTON_3
- LOGI_LCD_COLOR_BUTTON_LEFT
- LOGI_LCD_COLOR_BUTTON_RIGHT
- LOGI_LCD_COLOR_BUTTON_OK
- LOGI_LCD_COLOR_BUTTON_CANCEL
- LOGI_LCD_COLOR_BUTTON_UP
- LOGI_LCD_COLOR_BUTTON_DOWN
- LOGI_LCD_COLOR_BUTTON_MENU

Return value

If the button specified is being pressed it returns true. Otherwise false.

Notes

The button will be considered pressed only if your applet is the one currently in the foreground.

LogiLcdUpdate

The **LogiLcdUpdate** () function updates the lcd display.

```
void LogiLcdUpdate();
```

Notes

You have to call this function every frame of your main loop, to keep the lcd updated.

LogiLcdShutdown

The **LogiLcdShutdown** () function kills the applet and frees memory used by the SDK.

```
void LogiLcdShutdown();
```

Monochrome Lcd Function

LogiLcdMonoSetBackground

The **LogiLcdMonoSetBackground** () function sets the specified image as background for the monochrome lcd device connected.

```
bool LogiLcdMonoSetBackground(BYTE monoBitmap[]);
```

Parameters

- monoBitmap: the array of pixels that define the actual monochrome bitmap

  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)

Return value
True if it succeeds, false otherwise.

Notes
The image size must be 160x43 in order to use this function. The SDK will turn on the pixel on the screen if the value assigned to that byte is >= 128, it will remain off if the value is < 128.

LogiLcdMonoSetText
The **LogiLcdMonoSetText** () function sets the specified text in the requested line on the monochrome lcd device connected.

```
bool LogiLcdMonoSetText(int lineNumber, wchar_t* text);
```

Parameters
- lineNumber: the line on the screen you want the text to appear. The monochrome lcd display has 4 lines, so this parameter can be any number from 0 to 3.
- text: defines the text you want to display

Return value
True if it succeeds, false otherwise.

Color LCD functions

LogiLcdColorSetBackground
The **LogiLcdColorSetBackground** () function sets the specified image as background for the color lcd device connected.

```
bool LogiLcdColorSetBackground(BYTE colorBitmap[]);
```

Parameters

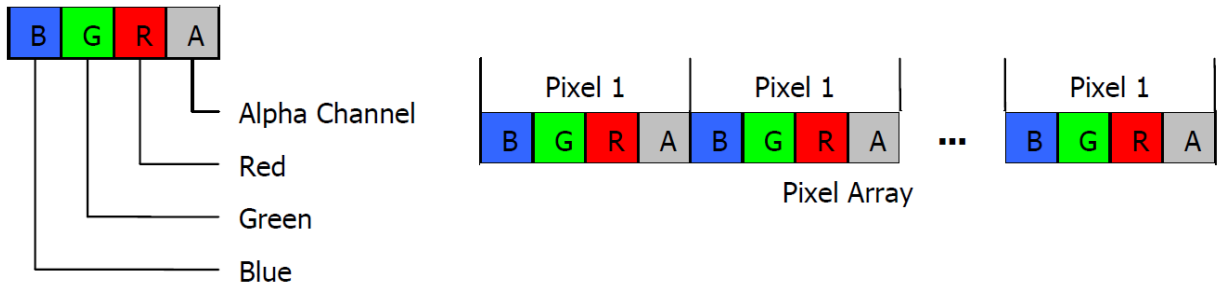
- colorBitmap: the array of pixels that define the actual color bitmap

  The array of pixels is organized as a rectangular area, 320 bytes wide and 240 bytes high. Since the color lcd can display the full RGB gamma, 32 bits per pixel (4 bytes) are used. The size of the colorBitmap array has to be 320x240x4 = 307200 therefore. 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 0-3 (0,0)	byte 4-7 (1,0)	byte 8-11 (2,0)	…	byte 1268-1271 (317,0)	byte 1272-1275 (318,0)	byte 1276-1279 (319,0)
byte 1280-1283 (0,1)	byte 1284-1287 (1,1)	byte 1288-1291 (2,1)	…	byte 2549-2552 (317,1)	byte 2553-2555 (318,1)	byte 2556-2559 (319,1)
…	…	…	…	…	…	…
byte 304640-304643 (0,238)	byte 304646-304649 (1,238)	byte 304650-304653 (2,238)	…	byte 305908-305911 (317,238)	Byte 305912-305915 (318,238)	byte 305916-305919 (319,238)
byte 305920-305923 (0,239)	byte 305924-305927 (1,239)	byte 305928-305931 (2,239)	…	byte 307188-307191 (317,239)	byte 307192-307195 (318,239)	byte 307196-307199 (319,239)

32 bit values are stored in 4 consecutive bytes that represent the RGB color values for that pixel. These values use the same top left to bottom right raster style transform to the flat character array with the exception that each pixel value is specified using 4 consecutive bytes. The illustration below shows the data arrangement for these RGB quads.



Each of the bytes in the RGB quad specify the intensity of the given color. The value ranges from 0 (the darkest color value) to 255 (brightest color value)

Return value
True if it succeeds, false otherwise.

Notes
The image size must be 320x240 in order to use this function.

LogiLcdColorSetTitle

The **LogiLcdColorSetTitle** () function sets the specified text in the first line on the color lcd device connected. The font size that will be displayed is bigger than the one used in the other lines, so you can use this function to set the title of your applet/page.

```
bool LogiLcdColorSetTitle(wchar_t* text, int red, int green, int blue);
```

Parameters

- text: defines the text you want to display as title
- red, green, blue: this lcd can display a full RGB color gamma, you can define the color of your title using this parameters. Values between 0 and 255 are accepted. The default value for this parameters is 255, so if you don't specify any color, your title will be white.

Return value
True if it succeeds, false otherwise.

LogiLcdColorSetText

The **LogiLcdColorSetText** () function sets the specified text in the requested line on the color lcd device connected.

```
bool LogiLcdColorSetText(int lineNumber, wchar_t* text, int red, int green, int blue);
```

Parameters

- lineNumber: the line on the screen you want the text to appear. The color lcd display has 8 lines for standard text, so this parameter can be any number from 0 to 7.
- text: defines the text you want to display
- red, green, blue: this lcd can display a full RGB color gamma, you can define the color of your text using this parameters. Values between 0 and 255 are accepted. The default value for this parameters is 255, so if you don't specify any color, your text will be white.

Return value
True if it succeeds, false otherwise.

Unreal Engine and DLLBind specific functions

The following functions are specific for Unreal Engine script integration through DLLBind :

- `LogiLcdColorSetBackgroundUDK`
- `LogiLcdColorResetBackgroundUDK`
- `LogiLcdMonoSetBackgroundUDK`
- `LogiLcdMonoResetBackgroundUDK`

The main reason for these different functions is the array size limitation in the UnrealScript language to 2048. As described before the bitmap in the LCDs is represented by an array of bytes. For both the monochrome and the color version the array size is way bigger than that limitation. Using these functions in Unreal Script you will still be able to send bitmaps to the LCDs. For more information on how to use these functions check the document UdkDLLBindInstructions in the Doc folder of this package.

REMARK: Don't use this functions if you are not using Unreal Engine and integrating Logitech support using DLLBind technology.