

Programmer Manual

Line Display Mac OS API -- Mac Dynamic Lib

For Logic Controls USB Line Display

Version 1.3



Index

Introduction to USB Line Display Interface	3
Installing and Testing API	4
Using API Functions in Applications	5
API Functions	7
IOReturn LCPDGetDeviceDescriptors(LCPDDeviceDescriptorRef **foundInterfaces, int *foundCount)	7
void LCPDReleaseDescriptors(LCPDDeviceDescriptorRef *descr, UInt32 count)	7
UInt32 LCPDioctl(LCPDDeviceDescriptorRef descr, UInt32 command, char *buffer)	7
UInt32 LCPDWrite(LCPDDeviceDescriptorRef descr, char *buffer, UInt32 size)	8
bool LCPDRegisterWakeCallback(LCPDWakeCallback callbackfunction, CFRunLoopRef runLoopRef)	8

Introduction to USB Line Display Interface

This API (Application Program Interface) is a software interface which enables user's application program to communicate with the Logic Controls line display devices with USB connector on Mac PC.

This Logic Controls Mac OS API is provided in the form of a dynamic link library, *libLCPD-mac.dylib*, which contains some routines that can be called from the application software. This library implements the API similar to the Linux version of the library. The dll contains all functions, which will be described below, necessary to send data to Logic Controls line display devices with USB connector.

This library was tested under:

- Mac OS X 10.6.6 Intel
- Mac OS X 10.6.6 x64 Intel
- Mac OS X 10.5.8 Intel

The reference document "LogicControls.pdf" explains the details about Logic Controls line display command set.

Installing and Testing API

This section describes the steps to install the library on the Mac OS system and test the dylib with the testing program.

Step1:

Copy the file "Install_lcpdDll" into "Documents" folder.

Step2:

Unzip the file "QcPdUsb" into "Documents" folder.

Step3:

Install dylib LCPD driver by double-clicking on "Install_lcpdDll" and input the user password.

Step4:

Connection Logic Controls line display device to Mac PC USB socket. And wait until a logo message displayed on the screen of the device.

Step5:

Run the testing program by double clicking on "QcPdUsb" to test the API.

Note1:

If error messages box pop-out:

1. "QcPdUsb cannot be opened because of a problem" - the driver was not installed properly.
2. "No LCPD devices found" - check the connection of the USB pole display with Mac PC.
3. "... tty ..." - do the followings:
 - Open the terminal in the "Applications -> Utilities".
 - Run "cd usr".
 - Run "cd local".
 - Run "sudo rmdir lib" to remove the folder.
 - Run "cd .." to return to "usr" folder.
 - Run "sudo rmdir local" to remove the folder.
 - Run "cd .." to return to current user folder.
 - Run "sudo rmdir usr" to remove the folder.

Note2:

To run the testing program in a terminal:

4. Open a terminal window.
5. Get the testing program folder with "cd" command.
6. Run "chmod 755 example" in the terminal window.
7. Run "./example".

Using API Functions in Applications

This section shows step by step how to use the API functions, and build an example program.

In order to build an application with the API:

1. Copy files "usblcpdmac.h" into the project folder.
2. Include header file "usblcpdmac.h" in the application.
3. Add "-llcpd-mac" in the "Other Linker Flags" line of the building settings.
4. Add functions of API into application.
5. Build the application program.

Note:

In the menu bar, Project -> Edit Project Settings -> Build. Select "Debug" or "Release" in Configuration box, and find the line "Other Linker Flags" among "Linking" lines.

To create, build and run a 'C' project in Xcode environment (version 3.2.6):

1. Run Xcode by double clicking it in "Applications" folder.
2. In the menu bar, select File -> New Project -> Application -> Command Line Tool -> Type 'C' to create a new project.
3. Select the project folder, and enter project name.
4. Open the file "main.c" in edit panel.
5. Copy the header file "usblcpdmac.h" into the project folder.
6. Copy the following lines of codes into "main.c", and delete the original function "main()".

Example Codes:

```
#include <stdio.h>

#include <unistd.h>
#include "usblcpdmac.h"

#define LOGO_STRING " Logic Controls Inc      Bematech      "
#define SPACE_STRING "                               "
#define WAKEUP_STRING "  computer wakeup  "
#define STR_LENGTH 20
#define SLEEP_BETWEEN_TESTS 2 //seconds

LCPDDeviceDescriptorRef *deviceGot;
int deviceNumber;

int write_pd_text(LCPDDeviceDescriptorRef descr, char* text, int len)
{
    char buf[128];
    strncpy(buf, text, len);

    if(LCPDWrite(descr, buf, len))
    {
        fprintf(stderr, "--Error writing text\n");
        return -1;
    }

    return 0;
}

void DisplayText(char* text, int len)
{
    printf("start to display text\n");
```

```

    LCPDGetDeviceDescriptors(&deviceGot, &deviceNumber);
    if (!deviceNumber)
    {
        fprintf(stderr, "No LCPD devices found");
        LCPDReleaseDescriptors(deviceGot, deviceNumber);
        return;
    }
    printf("Found %d devices\n", deviceNumber);

    for (int i = 0; i < deviceNumber; i++)
    {
        write_pd_text(deviceGot[i], text, len);
    }
    sleep(SLEEP_BETWEEN_TESTS);

    LCPDReleaseDescriptors(deviceGot, deviceNumber);

    printf("display end !\n");
}

void WakeCallback()
{
    //LCPD devices are re-attached to the system after sleep and wake.
    //Perform tests with the new instances.
    DisplayText(WAKEUP_STRING, STR_LENGTH);
}

int main (int argc, const char * argv[])
{
    printf("\nExample Mac Program for LCI USB Pole Display\n");

    //register wake callback
    LCPDRegisterWakeCallback(WakeCallback, CFRunLoopGetCurrent());

    //output text to pole display
    DisplayText(LOGO_STRING, STR_LENGTH * 2 - 1);

    //receive wake notification
    CFRunLoopRun();

    //deregister callback
    LCPDRegisterWakeCallback(NULL, NULL);

    printf("\n quit the test program !\n");
    return(0);
}

```

7. Add the linking flag as described above.
8. Add "Corefoundation" and "IOKit" frameworks:
Right click project name, Add -> ExistingFrameworks, select "Frameworks", "Corefoundation.framework", and "IOKit.framework". Click "Add" button.
9. Build and run the project.

API Functions

This section describes all functions in the API dynamic library.

IOReturn LCPDGetDeviceDescriptors(LCPDDeviceDescriptorRef **foundInterfaces, int *foundCount)

This function try to get the descriptors of Logic Controls USB devices connected in the Mac PC USB sockets. It will return the number of the USB devices. There is no Logic Controls USB devices connected if the number is zero.

This function must be called first before all other functions in the API can be used.

Parameter:

foundInterfaces: the interfaces of Logic Controls USB devices.
foundCount: the number of Logic Controls USB devices.

Return:

IOReturn: a standard IO return code.

Example:

Example in 'C' - See the previous section

void LCPDReleaseDescriptors(LCPDDeviceDescriptorRef *descr, UInt32 count)

This function release the interfaces of Logic Controls USB devices connected in the Mac PC USB sockets.

Parameter:

descr: the interface of Logic Controls USB devices.
count: the number of of Logic Controls USB devices.

Return:

void

Example:

Example in 'C' See the former section

UInt32 LCPDioctl(LCPDDeviceDescriptorRef descr, UInt32 command, char *buffer)

This function try to get some information strings of Logic Controls USB devices connected in the Mac PC USB sockets.

Parameter:

descr: the interface of Logic Controls USB devices.
command: the command code of standard ioctl.
buffer: the information string returned.

Return:

UInt32: a standard IO return code.

Example:

Example in 'C'

```
#define GET_HARD_VERSION  IOCTL_GET_HARD_VERSION
#define GET_DRV_VERSION   IOCTL_GET_DRV_VERSION

char buf[128];
memset(buf,0,128);

// get driver version
if (LCPDioctl(descr, GET_DRV_VERSION, buf))
{
    fprintf(stderr,"IOCTL failed, could not get Driver Version!\n");
} else
{
    printf("Driver Version: %s\n", buf);
}

// get device version
if(LCPDioctl(descr, GET_HARD_VERSION, buf))
{
    fprintf(stderr,"IOCTL failed, could not get Hardware Version!\n");
} else
{
    printf("--Hardware Version: %s\n",buf);
}
```

UInt32 LCPDWrite(LCPDDeviceDescriptorRef descr, char *buffer, UInt32 size)

This function write data to bulk pipe of Logic Controls USB devices connected in the Mac PC USB sockets.

Parameter:

buffer: the data string in buffer.
count: the length of data string.

Return:

UInt32: a standard IO return code.

Example:

Example in 'C' - See the previous section

bool LCPDRegisterWakeCallback(LCPDWakeCallback callbackfunction, CFRunLoopRef runLoopRef)

This function registers Wake-notification callback function in the runloop.

This function must be called first before CFRunLoopRef() be called.

Parameter:

callbackfunction: pointer to callback function.
runLoopRef: framework function CFSRunLoopGetCurrent().

Return:

bool: true or fault.

Example:

Example in 'C' - See the previous section