K8090D.DLL

Technical Guide

Table of contents

Introduction

General

The K8090D DLL provides a wrapper for the functionality offered by the K8090. This wrapper is introduced in the form of a **Dynamic Link Library (DLL)**. An application can use this DLL to communicate with the K8090 in a simplified manner. In this manual we will describe each of these functions provided by the DLL in detail.

Readers should have an understanding of the basic data types as well as basic knowledge of the Microsoft Windows operating system.

Calling convention

A calling convention is a scheme for how functions receive parameters from their caller and how they return a result. Different programming languages use different calling conventions, so it is important to know which calling convention is used by your programming language and which calling convention is used by the K8090 DLL.

The most common calling convention is the **stdcall** calling convention, and this is also the one we have used for our DLL.

If you are using .NET (VB.NET or C#) you do not need to worry about this since the calling convention in .NET is also stdcall. However if you are using C to import the functions provided by the DLL, you will need to pay special attention to this.

Function List

OpenDevice

Attempts to establish a connection to the K8090 using the specified serial communications port.

Syntax:

```
HANDLE WINAPI OpenDevice(
      __in LPCSTR szPort
      );
```

Parameters:

szPort [in]	Null-terminated string containing the port name

Return value:

If the function succeeds, the return value is a handle to the K8090 device. If the function fails the return value is INVALID_HANDLE_VALUE. To get extended information, call **GetLastError**.

Example:

```
int main()
{
      HANDLE hDevice;

      hDevice = OpenDevice("COM1");
      if (hDevice == NULL) {
            printf("Connection failed");
            return 1;
      }

      CloseDevice(hDevice);
}
```

CloseDevice

Closes a handle opened by a call to OpenDevice. This closes the connection and frees any used resources.

Syntax:

```
VOID WINAPI CloseDevice(
      __in HANDLE hDevice
      );
```

Parameters:

hDevice [in]	Device handle returned by a call to OpenDevice

SendCommand

Sends an asynchronous command packet to the K8090 device.

Syntax:

```
BOOL WINAPI SendCommand(
      __in HANDLE hDevice,
      __in BYTE cmd,
      __in BYTE mask,
      __in BYTE hparam,
      __in BYTE lparam
      );
```

Parameters:

hDevice [in]	Device handle returned by a call to OpenDevice
cmd [in]	Command byte.
mask [in]	Mask byte.
hparam [in]	First command parameter.
lparam [in]	Second command parameter.

View the K8090 Protocol Manual for more information about each parameter and their possible values.

Return value:

If the function succeeds, the return value is TRUE. If the function fails the return value is FALSE. To get extended information, call **GetLastError**.

Example:

```
#define COMMAND_SWITCH_RELAY_ON 0x11

int main()
{
      ... // connect

      SendCommand(hDevice, COMMAND_SWITCH_RELAY_ON,
            0x03, // Relay 1 and 2
            0x00, // Ignored
            0x00  // Ignored
            );
      ...
}
```

RegisterListener

Register an event window to receive event messages sent by the K8090 board. Only one window can be registered at a time.

Syntax:

```
VOID WINAPI RegisterListener(
      __in HANDLE hDevice,
      __in HWND hWnd
      );
```

Parameters:

hDevice [in]	Device handle returned by a call to OpenDevice
hWnd [in]	Window that is to receive K8090 event messages. These messages will have a message ID of WM_USER+1.

Example:

```
#define COMMAND_SWITCH_RELAY_ON 0x11

LRESULT CALLBACK WndProc(
      HWND   hWnd,
      UINT   message,
      WPARAM wParam,
      LPARAM lParam
) {
      BYTE cmd;

      switch(message)
      {
            case WM_USER+1:
                  cmd = (wParam >> 8);
                  switch(cmd)
                  {
                        case COMMAND_SWITCH_RELAY_ON:
                              // do something
                              break;
                  }
                  break;
            default:
                  return DefWindowProc(hWnd, message, wParam, lParam);
      }
}
```

UnregisterListener

Unregisters the event window registered by a call to RegisterListener.

Syntax:

```
VOID WINAPI UnregisterListener(
      __in HANDLE hDevice
      );
```

Parameters:

hDevice [in]	Device handle

Command List

This is an overview of all available commands.

```
#define CMD_SWITCH_RELAY_ON                0x11
#define CMD_SWITCH_RELAY_OFF               0x12
#define CMD_TOGGLE_RELAY                   0x14
#define CMD_QUERY_RELAY_STATUS             0x18
#define CMD_SET_MANUAL_OPERATION_MODE      0x21
#define CMD_QUERY_MANUAL_OPERATION_MODE    0x22
#define CMD_START_RELAY_TIMER              0x41
#define CMD_SET_RELAY_TIMER_DELAY          0x42
#define CMD_QUERY_TIMER_DELAY              0x44
#define CMD_BUTTON_STATUS                  0x50
#define CMD_RELAY_STATUS                   0x51
#define CMD_RESET_FACTORY_DEFAULTS         0x66
#define CMD_GET_JUMPER_STATUS              0x70
#define CMD_FIRMWARE_VERSION               0x71
```