卓岚设备管理函数库
用户手册

——ZLDevManageV2.dll 的使用

版权©2014 上海卓岚信息科技有限公司保留所有权力

ZL DUI 20140320.1.0

版权©2014 上海卓岚信息科技有限公司保留所有权力

版本信息

对该文档有如下的修改：

修改记录

日期	文档编号	修改内容
2014-3-18	ZL DUI 20140318.1.0	库 V2.0.0： 1) 增加设备单个参数获取及设置接口，不再使用结构体来保存参数。 2) 只保留内网设备的搜索及参数管理接口。

所有权信息

未经版权所有者同意，不得将本文档的全部或者部分以纸面或者电子文档的形式重新发布。

本文档只用于辅助读者使用产品，上海卓岚公司不对使用该文档中的信息而引起的损失或者错误负责。本文档描述的产品和文本正在不断地开发和完善中。上海卓岚信息科技有限公司有权利在未通知用户的情况下修改本文档。

目录

1. 概述

在使用卓岚联网设备时，在设备端，用户已经可通过串口方便地实现联网；在 PC 端，如何和设备进行通信是一个用户经常关心的问题。

基本上有两种方式实现该通信：

1. 虚拟串口方式：用户通过 ZLVirCom 程序虚拟一个虚拟串口，用户可以读写虚拟串口来实现对设备的读写。这样将网络编程转化为较为简单的串口通信编程。
2. socket 通信方式：用户使用标准的 TCP、UDP 编程接口实现网络编程，例如 Windows API 提供 socket 接口，MFC VC++提供 CSocket、CAsyncSocket 类等。

用以上的方法已经可以进行设备的通信。为了方便设备的管理、参数读取和修改，卓岚设备管理 DLL 库提供了设备的搜索、设备参数读取、设备参数修改的接口。

ZLDevManageDllV2 版本不再含有设备通信功能，虽然 ZLDevMangeDllV1 版本本身也提供了设备通信的接口，但是建议具有 socket 编程的用户还是用 socket 编程通信，因为 socket 通信更加直接和高效。

1.1. 函数库的功能

使用卓岚设备管理函数库可以实现如下功能：

1. 方便地获取所有物理子网卓岚联网设备。
2. 读取、修改某一设备的参数，可以将参数读取、修改功能集成到用户程序中。
3. 函数库具有跨平台的能力，不论用户采用的是 VC、VB、Delphi、C++Builder 都可以调用函数库。

1.2. 函数库的特点

卓岚设备管理函数库具有如下特点：

1. 卓岚设备管理函数库支持任意参数读取与更改，参数操作非常简便。
2. 卓岚设备管理函数库支持搜索局域网设备。
3. 不涉及到结构体定义及使用，简化函数库的调用。

2. 函数库描述

2.1 头文件

函数库用到的宏定义、函数定义全部在 ZLDevManageV2Dll.h 文件中（C 语言格式）

类型定义

```
typedef unsigned char    zl_u8;
typedef unsigned short   zl_u16;
typedef unsigned int     zl_u32;
typedef char             zl_s8;
typedef short            zl_s16;
typedef int              zl_s32;

typedef unsigned int     zl_bool;
typedef unsigned int     zl_boolean;
typedef char             zl_char;
typedef void             zl_void;
typedef int              zl_int;
```

函数指针类型定义

函数库中的函数指针类型定义如下，各个函数的含义将在后面讲解：

```
typedef zl_u8   *   (__stdcall * tZLDM_GetDevID)(zl_u8 nNum);
typedef zl_u8       (__stdcall * tZLDM_GetStatus)(zl_u8 nNum);
typedef zl_s8   *   (__stdcall * tZLDM_GetDevName)(zl_u8 nNum);
typedef zl_u8       (__stdcall * tZLDM_GetIPMode)(zl_u8 nNum);
typedef zl_s8   *   (__stdcall * tZLDM_GetIP)(zl_u8 nNum);
typedef zl_u16      (__stdcall * tZLDM_GetPort)(zl_u8 nNum);
typedef zl_s8   *   (__stdcall * tZLDM_GetGateWay)(zl_u8 nNum);
typedef zl_u8       (__stdcall * tZLDM_GetWorkMode)(zl_u8 nNum);
typedef zl_s8   *   (__stdcall * tZLDM_GetNetMask)(zl_u8 nNum);
typedef zl_s8   *   (__stdcall * tZLDM_GetDestName)(zl_u8 nNum);
typedef zl_u16      (__stdcall * tZLDM_GetDestPort)(zl_u8 nNum);
typedef zl_bool     (__stdcall * tZLDM_SetDevName)(zl_u8 nNum, zl_s8 * DevName);
typedef zl_bool     (__stdcall * tZLDM_SetIPMode)(zl_u8 nNum, zl_u8 IPMode);
typedef zl_bool     (__stdcall * tZLDM_SetIP)(zl_u8 nNum, zl_s8 * IP);
typedef zl_bool     (__stdcall * tZLDM_SetPort)(zl_u8 nNum, zl_u16 Port);
typedef zl_bool     (__stdcall * tZLDM_SetGateWay)(zl_u8 nNum, zl_s8 * GateWay);
typedef zl_bool     (__stdcall * tZLDM_SetWorkMode)(zl_u8 nNum, zl_u8 WorkMode);
typedef zl_bool     (__stdcall * tZLDM_SetNetMask)(zl_u8 nNum, zl_s8 * NetMask);
```

```
typedef zl_bool     (__stdcall * tZLDM_SetDestName)(zl_u8 nNum, zl_s8 * DestName);
typedef zl_bool     (__stdcall * tZLDM_SetDestPort)(zl_u8 nNum, zl_u16 DestPort);
typedef zl_bool     (__stdcall * tZLDM_SetBaudrateIndex)(zl_u8 nNum, zl_u8 Baudrate);
typedef zl_bool     (__stdcall * tZLDM_SetParity)(zl_u8 nNum, zl_u8 Parity);
typedef zl_bool     (__stdcall * tZLDM_SetDataBits)(zl_u8 nNum, zl_u8 DataBits);
typedef zl_bool     (__stdcall * tZLDM_SetFlowControl)(zl_u8 nNum, zl_u8 FlowControl);
```

宏定义

宏定义的含义标注如下：

```
#define MAX_DEV_NAME_LEN    10  //参数结构体中设备名称的最长长度
#define DNS_NAME_MAX_LEN    30  //目的地址的最长长度

#define WORK_MODE_SERVER    0   //参数结构体中 WorkMode 取值
#define WORK_MODE_CLIENT    1
#define WORK_MODE_UDP       2

#define IP_MODE_STATIC      0   //参数结构体中 IPMode 的取值
#define IP_MODE_DHCP        1

#define DEST_MODE_STATIC    0   //参数结构体中 DestMode 的取值
#define DEST_MODE_DYNAMIC   1

#define FLOW_C_NONE         0   //参数结构体中 FlowControl 的取值
#define FLOW_C_CTS_RTS      1

#define PARAM_PARITY_NONE   0   //参数结构体中 Parity 的取值
#define PARAM_PARITY_ODD    1
#define PARAM_PARITY_EVEN   2
#define PARAM_PARITY_MARK   3
#define PARAM_PARITY_SPACE  4

#define DATA_BITS_8         0   //参数结构体中 BaudrateIndex 的取值
#define DATA_BITS_7         1
#define DATA_BITS_6         2
#define DATA_BITS_5         3

#define BANDURATE_1200      0   //参数结构体中 DataBits 的取值
#define BANDURATE_2400      1
#define BANDURATE_4800      2
#define BANDURATE_7200      3
#define BANDURATE_9600      4
#define BANDURATE_14400     5
#define BANDURATE_19200     6
```

```
#define BANDURATE_28800          7
#define BANDURATE_38400          8
#define BANDURATE_57600          9
#define BANDURATE_76800          10
#define BANDURATE_115200         11



#define DEV_STATUS_CLOSED        0
#define DEV_STATUS_OPENED        1

#define ZLDM_HANDLER_ARRAY_MIN_SIZE     256  //设备句柄数据的建议最小值

#define ZLDM_FAST_SEND                  0    //快速发送模式
#define ZLDM_BULK_SEND                  1    //大数据量发送模式

#define ZLDM_VER                        9        //函数库版本号
```

2.2 函数库定义

ZLDM_StartSearchDev

搜索网络中所有联网的卓岚联网设备。

```
zl_u16 __stdcall ZLDM_StartSearchDev();
```

参数

无

返回值

返回类型为短整型，表示搜索到的设备个数。

描述

开始搜索网络中所有联网的 ZLSN 设备，包括物理子网中所有 ZLSN 设备。搜索结果会暂时保存起来，但是一旦有新的设备连接到网络中，或者有设备退出网络，那么需要重新搜索以获得最新的设备列表。

ZLDM_GetDevID

获取设备 MAC 地址

```
zl_u8 * __stdcall ZLDM_GetDevID(zl_u8 nNum)
```

参数

nNum:

设备序号（也就是搜索完成的设备列表中，该设备的位置）

返回值

返回类型为字符指针。

描述

返回设备的 MAC 地址，返回字符串类型，例如 MAC 地址 0x112233445566，那么返回则是”112233445566”字符串，无需转换。

ZLDM_GetDevName

获取设备名称

```
zl_s8 * __stdcall ZLDM_GetDevName(zl_u8 nNum)
```

参数

nNum:

设备序号

返回值

返回设备的名称字符指针。

描述

返回设备名字符串头指针，设备名长度为 10 个字节。

ZLDM_GetStatus

获取设备状态

```
zl_u8 __stdcall ZLDM_GetStatus(zl_u8 nNum)
```

参数

nNum:

设备序号

返回值

返回设备的链接状态（是否建立了链接），返回类型为字符指针。

描述

获取链接状态

ZLDM_GetIPMode

获取 IP 模式

```
zl_u8 __stdcall ZLDM_GetIPMode(zl_u8 nNum)
```

参数

nNum:

设备序号

返回值

0 表示由 DHCP 分配设备 IP，静态则是手动设置的 IP。

描述

获取设备 IP 获取方式。

ZLDM_GetIP

获取设备 IP

```
zl_s8 * __stdcall ZLDM_GetIP(zl_u8 nNum)
```

参数

nNum:

设备序号

返回值

返回 IP 地址字符串。

描述

返回 IP 地址字符串，例：192.168.0.1 返回的则是”192.168.0.1”

ZLDM_GetPort

获取设备端口

```
zl_u16 __stdcall ZLDM_GetPort(zl_u8 nNum)
```

参数

nNum:

设备序号

返回值

返回一个 2 字节短整形 IP 端口。

描述

返回 2 字节的 IP 端口。

ZLDM_GetWorkMode

获取工作模式

```
zl_u8 __stdcall ZLDM_GetWorkMode(zl_u8 nNum)
```

参数

nNum:

设备序号

返回值

0 工作在 TCP server 模式
1 工作在 TCP Client 模式
2 工作在 UDP 模式
3 工作在 UDP Group 模式

描述

返回设备的工作模式。

ZLDM_GetNetMask

获取子网掩码

```
zl_s8 * __stdcall ZLDM_GetNetMask(zl_u8 nNum)
```

参数

nNum:

设备序号

返回值

返回子网掩码 IP 地址字符串。

描述

返回子网掩码 IP 地址字符串，例：192.168.0.1 返回的则是”192.168.0.1”

ZLDM_GetGateWay

获取网关

```
zl_s8 * __stdcall ZLDM_GetGateWay(zl_u8 nNum)
```

参数

nNum:

设备序号

返回值

返回网关 IP 地址字符串。

描述

返回网关 IP 地址字符串，例：192.168.0.1 返回的则是”192.168.0.1”

ZLDM_GetDestName

获取目的 IP 或域名

```
zl_s8 * __stdcall ZLDM_GetDestName(zl_u8 nNum)
```

参数

nNum:

设备序号

返回值

返回一个字符串指针。

描述

返回目标 IP 或域名，返回一个字符串，最大长度为 30

ZLDM_GetDestPort

获取目的端口

```
zl_u16 __stdcall ZLDM_GetDestPort(zl_u8 nNum)
```

参数

nNum:

设备序号

返回值

返回一个 2 字节短整形 IP 端口。

描述

返回 2 字节的 IP 端口。

ZLDM_GetBaudrateIndex

获取串口波特率

```
zl_u8 __stdcall ZLDM_GetBaudrateIndex(zl_u8 nNum)
```

参数

nNum:

设备序号

返回值 返回 0-11 范围内的数值，数值具体含义如下

0:1200	1:2400
2:4800	3:7200
4:9600	5:14400
6 :19200	7:28800
8:38400	9:57600

10:76800　　11:115200

描述

返回波特率索引值。

ZLDM_GetDataBits

获取串口数据位

```
zl_u8 __stdcall ZLDM_GetDataBits(zl_u8 nNum)
```

参数

nNum:

设备序号

返回值

返回 0-3 范围内的数值，数值具体含义如下

0:8 位　　1:7 位

2:6 位　　3:5 位

描述

返回串口数据位。

ZLDM_GetParity

获取串口校验方式

```
zl_u8 __stdcall ZLDM_GetParity(zl_u8 nNum)
```

参数

nNum:

设备序号

返回值

返回 0-4 范围内的数值，数值具体含义如下

0:无校验　　1:偶校验

2:奇校验　　3:Mark 校验

4：space 校验

描述

返回串口校验方式。

ZLDM_GetFlowControl

获取串口流控方式

```
zl_u8 __stdcall ZLDM_GetFlowControl(zl_u8 nNum)
```

参数

nNum:

设备序号

返回值

返回 0-4 范围内的数值，数值具体含义如下

0:无流控　　1:CTS/RTS

2:DST/DTR　　3:XON/XOFF

描述

返回串口流控方式。

ZLDM_SetParam

设置参数生效

```
zl_s8 * __stdcall ZLDM_SetParam(zl_u8 nNum)
```

参数

nNum:

设备序号。

返回值

TRUE 修改设置成功 FALSE 没有找到设备。

描述

调用修改参数系列 API 如 ZLDM_SetDevName，ZLDM_SetIP 等之后，调用此函数使设置生效。

ZLDM_SetDevName

设置设备名称

```
zl_s8 * __stdcall ZLDM_SetDevName(zl_u8 nNum，　zl_s8 * DevName)
```

参数

nNum:

设备序号

DevName:

字符串指针，指向设备名字符数组。

返回值

TRUE 修改设备参数成功 FALSE 没有找到设备。

描述

设置设备名称。

ZLDM_SetIPMode

修改 IP 获取模式

```
zl_u8 __stdcall ZLDM_SetIPMode(zl_u8 nNum, zl_u8 IPMode)
```

参数

nNum:

设备序号

IPMode:

IP 获取模式。 0 代表 DHCP， 1 代表静态。

返回值

TRUE 修改设备参数成功 FALSE 没有找到设备。

描述

修改设备 IP 获取方式。

ZLDM_SetIP

修改设备 IP

```
zl_u32 __stdcall ZLDM_SetIP(zl_u8 nNum, zl_s8 * IP)
```

参数

nNum:

设备序号

IP:

IP 地址字符串 如：”192.168.0.1”。

返回值

TRUE 修改设备参数成功 FALSE 没有找到设备。

描述

修改 IP。

ZLDM_SetPort

修改设备端口

```
zl_u16 __stdcall ZLDM_SetPort(zl_u8 nNum，  zl_u16 Port)
```

参数

nNum:

设备序号

Port:

设备端口， 无符号短整型。

返回值

TRUE 修改设备参数成功 FALSE 没有找到设备。

描述

修改端口号。

ZLDM_SetWorkMode

修改设备工作模式

```
zl_u8 __stdcall ZLDM_SetWorkMode(zl_u8 nNum，  zl_u8 WorkMode)
```

参数

nNum:

设备序号

WorkMode:

设备工作模式

0 工作在 TCP server 模式

1 工作在 TCP Client 模式

2 工作在 UDP 模式

3 工作在 UDP Group 模式

返回值

TRUE 修改设备参数成功 FALSE 没有找到设备。

描述

修改设备的工作模式。

ZLDM_SetNetMask

修改设备子网掩码

```
zl_u32 __stdcall ZLDM_SetNetMask(zl_u8 nNum，  zl_s8 * NetMask)
```

参数

nNum:

设备序号

NetMask:

子网掩码 IP 地址字符串，如：”192.168.0.1”。

返回值

TRUE 修改设备参数成功 FALSE 没有找到设备。

描述

修改子网掩码。

ZLDM_SetGateWay

修改网关

```
zl_u32 __stdcall ZLDM_SetGateWay(zl_u8 nNum， zl_s8 * GateWay)
```

参数

nNum:

设备序号

GateWay:

网关 IP 地址字符串，如”192.168.0.1”。

返回值

TRUE 修改设备参数成功 FALSE 没有找到设备。

描述

修改网关。

ZLDM_SetDestName

修改目标 IP 或域名

```
zl_s8 * __stdcall ZLDM_SetDestName(zl_u8 nNum， zl_s8 * DestName)
```

参数

nNum:

设备序号

DestName:

字符串指针。

返回值

TRUE 修改设备参数成功 FALSE 没有找到设备。

描述

修改目的 IP 或域名。

ZLDM_SetDestPort

设置目的端口

```
zl_u16 __stdcall ZLDM_SetDestPort(zl_u8 nNum，  zl_u16 DestPort)
```

参数

nNum:

设备序号

DestPort:

目的端口。

返回值

TRUE 修改设备参数成功 FALSE 没有找到设备。

描述

修改目的端口。

ZLDM_SetBaudrateIndex

设置波特率

```
zl_u8 __stdcall ZLDM_SetBaudrateIndex(zl_u8 nNum，  zl_u8 Baudrate)
```

参数

nNum:

设备序号

DestPort:

串口波特率，参数看含义如下。

0:1200	1:2400
2:4800	3:7200
4:9600	5:14400
6 :19200	7:28800
8:38400	9:57600
10:76800	11:115200

返回值

TRUE 修改设备参数成功 FALSE 没有找到设备。

描述

修改串口波特率。

ZLDM_SetDataBits

设置数据位

```
zl_u8 __stdcall ZLDM_SetDataBits(zl_u8 nNum, zl_u8 DataBits)
```

参数

nNum:

设备序号

DataBits:

选取 0-3 范围内的数值，数值具体含义如下

0:8 位　　1:7 位

2:6 位　　3:5 位

。

返回值

TRUE 修改设备参数成功 FALSE 没有找到设备。

描述

修改串口数据位。

ZDM_SetParity

设置校验方式

```
zl_u8 __stdcall ZLDM_SetParity(zl_u8 nNum, zl_u8 Parity)
```

参数

nNum:

设备序号

DataBits:

选取 0-4 范围内的数值，数值具体含义如下

0:无校验　　1:偶校验

2:奇校验　　3:Mark 校验

4：space 校验

。

返回值

TRUE 修改设备参数成功 FALSE 没有找到设备。

描述

修改串口校验位。

ZLDM_SetFlowControl

设置流控

```
zl_u8 __stdcall ZLDM_SetFlowControl(zl_u8 nNum, zl_u8 FlowControl)
```

参数

nNum:

设备序号

DataBits:

选择 0-4 范围内的数值，数值具体含义如下

0:无流控　　1:CTS/RTS

2:DST/DTR　　3:XON/XOFF

返回值

TRUE 修改设备参数成功 FALSE 没有找到设备。

描述

修改串口流控方式。

3. 使用步骤

下面通过在 VC++6.0 上实现一个范例程序来讲解如何使用设备管理函数库。

3.1 ZLUseDevManage 范例功能

首先讲解范例的基本功能，以使读者对程序功能现有一个感性的认识。

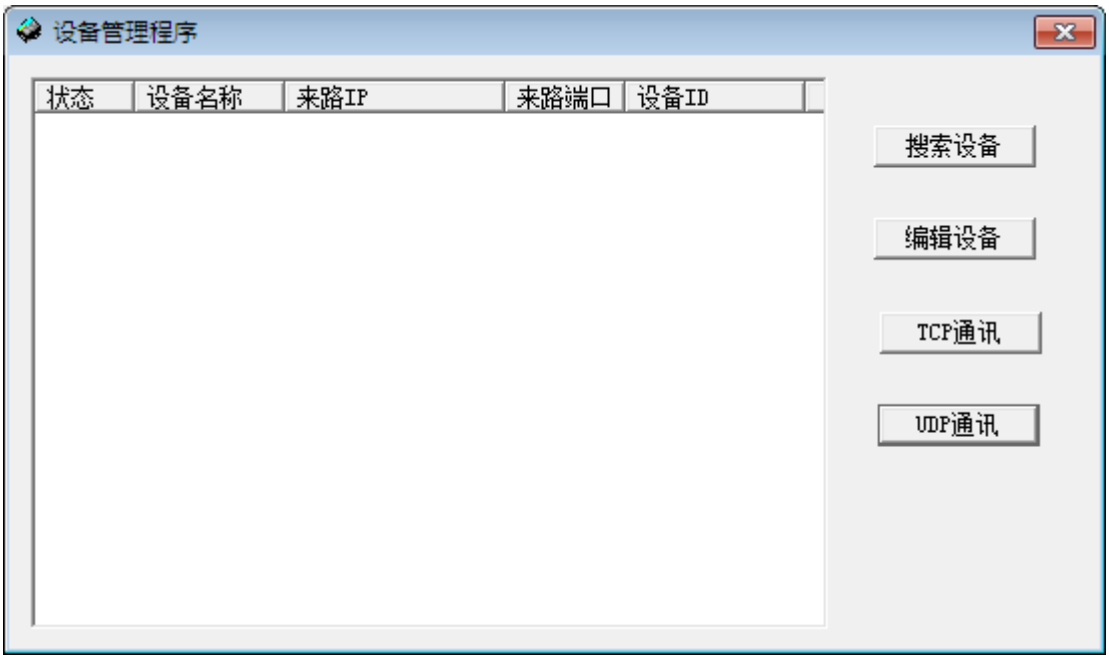

图 1 ZLUseDevManage 程序界面

范例程序界面如图 1 所示。该范例的功能有：

1. 搜索设备：点击该按钮可以搜索物理子网中的所有卓岚联网设备，搜索结果将保存在动态库链表中。
2. 选择一个设备：设备管理函数库将每一个设备映射为设备在列表中的序号，用户修改设备参数、与设备通信等功能都是通过设备序号进行的。
3. 参数查看和修改：使用设备管理函数库可以读取特定设备的参数、修改参数。
4. TCP 通讯：建立 TCP SOCKET 通讯的示例，可以工作在服务器及客户端模式与设备进行 TCP SOCKET 通讯。
5. UDP 通讯：UDP 通讯示例，用户通过这个示例了解 UDP 无连接通讯原理。

3.2 加载函数库

使用函数库之前需要将 ZLDevManageV2.dll 加载到内存中。

在 VC++6.0 中，可以在任何初始化的地方加载函数库，这里选择在 OnInitDialog()函数中加载。

1. 将 ZLDevManageV2.dll 拷贝到用户工程文件夹中。用户程序发布时，也需要将 ZLDevManageV2.dll 文件放在用户程序同一文件夹中。
2. 将 ZLDevManage V2Dll.h 拷贝到用户工程文件夹中，使用

```
#include "ZLDevManageDll.h"
```

包含函数库头文件。

3. 在 OnInitDialog()函数中加载 DLL。

```
HINSTANCE m_hZLDevManage;
m_hZLDevManage = LoadLibrary("ZLDevManageV2.dll");
```

4. 获取函数：为了方便后面程序的调用，初始化时可以先将函数库中的函数全部保存在全局函数指针中。

```
m_pZLDM_StartSearchDev=(tZLDM_StartSearchDev)GetProcAddress(m_hZLDevManage,"ZLDM_StartSearchDev");

m_pZLDM_GetDevID = (tZLDM_GetDevID)GetProcAddress(m_hZLDevManage, "ZLDM_GetDevID");
m_pZLDM_GetStatus = (tZLDM_GetStatus)GetProcAddress(m_hZLDevManage, "ZLDM_GetStatus");
m_pZLDM_GetDevName = (tZLDM_GetDevName)GetProcAddress(m_hZLDevManage, "ZLDM_GetDevName");
m_pZLDM_GetIPMode = (tZLDM_GetIPMode)GetProcAddress(m_hZLDevManage, "ZLDM_GetIPMode");
m_pZLDM_GetIP = (tZLDM_GetIP)GetProcAddress(m_hZLDevManage, "ZLDM_GetIP");
m_pZLDM_GetPort = (tZLDM_GetPort)GetProcAddress(m_hZLDevManage, "ZLDM_GetPort");
m_pZLDM_GetWorkMode = (tZLDM_GetWorkMode)GetProcAddress(m_hZLDevManage, "ZLDM_GetWorkMode");
m_pZLDM_GetNetMask = (tZLDM_GetNetMask)GetProcAddress(m_hZLDevManage, "ZLDM_GetNetMask");
m_pZLDM_GetGateWay = (tZLDM_GetGateWay)GetProcAddress(m_hZLDevManage, "ZLDM_GetGateWay");
m_pZLDM_GetDestName = (tZLDM_GetDestName)GetProcAddress(m_hZLDevManage, "ZLDM_GetDestName");
m_pZLDM_GetDestPort = (tZLDM_GetDestPort)GetProcAddress(m_hZLDevManage, "ZLDM_GetDestPort");
m_pZLDM_GetBaudrateIndex=(tZLDM_GetBaudrateIndex)GetProcAddress(m_hZLDevManage, "ZLDM_GetBaudrateIndex");
m_pZLDM_GetParity = (tZLDM_GetParity)GetProcAddress(m_hZLDevManage, "ZLDM_GetParity");
m_pZLDM_GetDataBits = (tZLDM_GetDataBits)GetProcAddress(m_hZLDevManage, "ZLDM_GetDataBits");
m_pZLDM_GetFlowControl = (tZLDM_GetFlowControl)GetProcAddress(m_hZLDevManage, "ZLDM_SetFlowControl");
m_pZLDM_SetDevName = (tZLDM_SetDevName)SetProcAddress(m_hZLDevManage, "ZLDM_SetDevName");
m_pZLDM_SetIPMode = (tZLDM_SetIPMode)SetProcAddress(m_hZLDevManage, "ZLDM_SetIPMode");
m_pZLDM_SetIP = (tZLDM_SetIP)SetProcAddress(m_hZLDevManage, "ZLDM_SetIP");

m_pZLDM_SetPort = (tZLDM_SetPort)SetProcAddress(m_hZLDevManage, "ZLDM_SetPort");

m_pZLDM_SetWorkMode = (tZLDM_SetWorkMode)SetProcAddress(m_hZLDevManage, "ZLDM_SetWorkMode");
```

```
m_pZLDM_SetNetMask        =        (tZLDM_SetNetMask)SetProcAddress(m_hZLDevManage,
                                                                    "ZLDM_SetNetMask");
m_pZLDM_SetGateWay        =        (tZLDM_SetGateWay)SetProcAddress(m_hZLDevManage,
                                                                    "ZLDM_SetGateWay");
m_pZLDM_SetDestName       =      (tZLDM_SetDestName)SetProcAddress(m_hZLDevManage,
                                                                    "ZLDM_SetDestName");
m_pZLDM_SetDestPort       =       (tZLDM_SetDestPort)SetProcAddress(m_hZLDevManage,
                                                                    "ZLDM_SetDestPort");
m_pZLDM_SetBaudrateIndex= (tZLDM_SetBaudrateIndex)SetProcAddress(m_hZLDevManage,
                                                                    "ZLDM_SetBaudrateIndex");
m_pZLDM_SetParity         =          (tZLDM_SetParity)SetProcAddress(m_hZLDevManage,
                                                                    "ZLDM_SetParity");
m_pZLDM_SetDataBits       =       (tZLDM_SetDataBits)SetProcAddress(m_hZLDevManage,
                                                                    "ZLDM_SetDataBits");
m_pZLDM_SetFlowControl    =    (tZLDM_SetFlowControl)SetProcAddress(m_hZLDevManage,
                                                                    "ZLDM_SetFlowControl");
```

其中各个函数指针的定义为：

```
tZLDM_GetDevID          m_pZLDM_GetDevID;
tZLDM_GetStatus         m_pZLDM_GetStatus;
tZLDM_GetDevName        m_pZLDM_GetDevName;
tZLDM_GetIPMode         m_pZLDM_GetIPMode;
tZLDM_GetIP             m_pZLDM_GetIP;
tZLDM_GetPort           m_pZLDM_GetPort;
tZLDM_GetWorkMode       m_pZLDM_GetWorkMode;
tZLDM_GetNetMask        m_pZLDM_GetNetMask;
tZLDM_GetGateWay        m_pZLDM_GetGateWay;
tZLDM_GetDestName       m_pZLDM_GetDestName;
tZLDM_GetDestPort       m_pZLDM_GetDestPort;
tZLDM_GetBaudrateIndex  m_pZLDM_GetBaudrateIndex;
tZLDM_GetParity         m_pZLDM_GetParity;
tZLDM_GetDataBits       m_pZLDM_GetDataBits;
tZLDM_GetFlowControl    m_pZLDM_GetFlowControl;
tZLDM_SetDevName        m_pZLDM_SetDevName;
tZLDM_SetIPMode         m_pZLDM_SetIPMode;
tZLDM_SetIP             m_pZLDM_SetIP;
tZLDM_SetPort           m_pZLDM_SetPort;
tZLDM_SetWorkMode       m_pZLDM_SetWorkMode;
tZLDM_SetNetMask        m_pZLDM_SetNetMask;
tZLDM_SetGateWay        m_pZLDM_SetGateWay;
tZLDM_SetDestName       m_pZLDM_SetDestName;
tZLDM_SetDestPort       m_pZLDM_SetDestPort;
tZLDM_SetBaudrateIndex  m_pZLDM_SetBaudrateIndex;
tZLDM_SetParity         m_pZLDM_SetParity;
tZLDM_SetDataBits       m_pZLDM_SetDataBits;
```

```
tZLDM_SetFlowControl       m_pZLDM_SetFlowControl;
```

之后库函数的调用可以通过函数指针调用，例如调用版本号函数为：

```
(*m_pZLDM_SetVer)();
```

3.3 搜索设备

使用 ZLDM_StartSearchDev 可以搜索物理子网内的设备。

```
m_DevCnt  =  (*m_pZLDM_StartSearchDev)();
```

搜索函数没有任何参数，仅返回搜索到的设备数量，而设备具体参数等都保存在 DLL 中的一个列表内。

在范例代码中，搜索到的设备将在一个 ComBox 控件中列举出来。用户在后续的操作前，必须首先在 ComBox 控件中选择一个设备，其实就是获取设备在列表中的序号。

3.4 参数操作

可以如下方法读取某个设备的参数：示例为读取设备 IP 地址

```
(*m_pZLDM_GetIP)(m_DevID)
```

其中 m_DevID 表示当前选择的设备在列表中的序号。

修改参数时，先选择设备，编辑要修改的参数即可。可以按如下的方式设置设备的 IP 参数 。

```
(*m_pZLDM_SetIP)(m_DevID，  ipaddr)
```

其中 m_DevID 表示当前选择的设备在列表中的序号。

Ipaddr 表示想要设置的 IP 地址

以上类似的修改参数的函数并没有真正修改设备的参数，它们仅仅将参数写到 DLL 库中对应设备的参数结构体中，最后通过调用下列函数使设置生效

```
(*m_pZLDM_SetParam)(m_DevID);
```

其中 m_DevID 表示当前选择的设备在列表中的序号。

修改成功后设备将自动重新启动。

注：如果需要修改多个参数的话，可以在修改完所有参数最后再调用(*m_pZLDM_SetParam)(m_DevID)接口，是设置生效。如：

```
(*m_pZLDM_SetIP)(m_DevID，  ipaddr);
(*m_pZLDM_SetPort(m_DevID, port);
(*m_pZLDM_SetDestIP(m_DevID, destIP);
(*m_pZLDM_SetDestPort(m_DevID, destPort);
(*m_pZLDM_SetParam)(m_DevID);//最后调用
```

而不用每设置一个参数就调用一次。

3.5 获取版本号

获取版本号的的方法如下：

```
(*m_pZLDM_GetVer)();
```

4. 售后服务和技术支持

上海卓岚信息技术有限公司

地址：上海市徐汇区漕宝路 80 号光大会展 D 幢 12 层

电话：021-64325189

传真：021-64325200

网址：http://www.zlmcu.com

邮箱：support@zlmcu.com