卓岚设备管理函数库
用户手册

——ZLDevManage.dll 的使用

ZL DUI 20090602.1.0

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
2009-6-02	ZL DUI 20090602.1.0	库 V1.0.7
2009-9-16	ZL DUI 20090602.1.0	库 V1.0.9： 增加断网恢复机制 可以搜索外网 Client 方式的设备。
2010-2-6	ZL DUI 20090602.1.0	库 V1.0.11： 1）优化外网设备的管理功能。
2010-2-23	ZL DUI 20090602.1.0	库 V1.0.12： 1）增加 UDP 设备的打开、通信功能。

所有权信息

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。

对于第二种方式，需要用户有一定的网络编程基础，虽然卓岚提供了网络编程参考例子（见光盘中的“Socket 编程例子程序”目录），但是对于没有接触网络编程的技术人员仍然有一定难度。

对此卓岚提供了设备管理函数库，将网络编程简化为类似文件的操作，方便了设备通信编程。

但是建议具有 socket 编程的用户还是用 socket 编程通信，这个 DLL 只是提供了搜索模块，读取模块参数和写模块参数的用途。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．方便地实现与设备的网络通信。并且由于函数库针对卓岚联网设备特点进行了优化，函数库内含了如下机制，这些机制一般 socket 不具有：

a) 创新的断网检测机制：无论卓岚联网设备处于内网还是外网，使用函数库中进行通信，可以在客户端或者服务端断网时自动恢复连接。

b) 提供快速模式，具有更为快速的发送速度，适合实时性要求高的应用。

2．方便地获取所有物理子网或外网卓岚联网设备。

3．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。

4．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. 注意事项

1. 函数库初始化以后会占用 UDP 的 4196 端口和 1 个 TCP 端口（该端口号由(*m_pZLDM_Init)(port);函数的 port 参数指定）。所以函数库不能够与 ZLVirCom 同时运行，否则会产生端口使用冲突。另外用户若只需要函数库的参数修改功能无需通过库与模块通信，则 port 参数可以为任意值，port 应该防止与用户程序的 socket 通信的端口相同而产生冲突。

1.3. 函数库的特点

卓岚设备管理函数库具有如下特点：

1．一般的设备管理函数库只支持设备搜索、参数读取、参数修改等功能，与此不同的是，卓岚设备管理函数库可以支持与设备的通信。
2．卓岚设备管理函数库支持一次性读取设备的全部参数，参数操作非常简便。
3．卓岚设备管理函数库支持搜索外网的联网设备。
4．卓岚设备管理函数库为数据接收提供事件消息机制，简化用户接收数据的实现。

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 ZLDevManage.h 文件中（C 语言格式）

1.1.1 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;

typedef zl_u32           IP_ADDR;
```

```
typedef zl_u32          ZLDevHandler;
```

其中：

1．IP_ADDR：这是函数库的 IP 地址的定义，它是一个 4 字节的变量类型。例如”192.168.0.254”表示为 IP_ADDR 类型的 IP 地址为十六进制数：0x0xFE00A8C0。

2．ZLDevHandler：这是一个 4 字节的变量类型，用于定义设备句柄。

1.1.2 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_bool     (__stdcall * tZLDM_Init)(zl_u16 ServerPort);
typedef void    (__stdcall * tZLDM_Exit)();
typedef zl_u16(__stdcall * tZLDM_StartSearchDev)(ZLDevHandler hDev[], zl_u16 hDevSize);
typedef zl_bool (__stdcall * tZLDM_GetDevParam)(ZLDevHandler h, struct SDevParam *pParam);
typedef zl_bool (__stdcall * tZLDM_SetDevParam)(ZLDevHandler h, struct SDevParam *pParam);
typedef zl_bool (__stdcall * tZLDM_DevOpen)(ZLDevHandler h, zl_u32 mode, HWND hWndCaller);
typedef zl_bool (__stdcall * tZLDM_DevClose)(ZLDevHandler h);
typedef int         (__stdcall * tZLDM_DevWrite)(ZLDevHandler h, const void *buf, int size);
typedef int         (__stdcall * tZLDM_DevRead)(ZLDevHandler h, void *buf, int size);
typedef int         (__stdcall * tZLDM_GetVer)();
typedef zl_bool (__stdcall * tZLDM_AddManualDev)(const char ip_dns[], zl_u16 port);
typedef void    (__stdcall * tZLDM_ClearManualDev)();
```

1.1.3 参数结构体

```
struct SDevParam
{
    zl_s8 DevName[MAX_DEV_NAME_LEN];

    zl_u8 IPMode;

    IP_ADDR LocalIP;
    IP_ADDR NetMask;
    IP_ADDR GateWay;
    IP_ADDR DnsServerIP;

    zl_u8 WorkMode;

    zl_u16 LocalPort;

    zl_u8 DestMode;
    char DestName[DNS_NAME_MAX_LEN];
    zl_u16 DestPort;
```

```
    zl_u8 BaudrateIndex;
    zl_u8 Parity;
    zl_u8 DataBits;
    zl_u8 StopBits;
    zl_u8 FlowControl;

    zl_u16 MaxPacketLen;
    zl_u8 MaxTimeGap;

    zl_u8 AppProtocol;

    zl_u8 ReconnectTime;

    zl_u8 KeepAliveTime;

    zl_u16 WebPort;

    zl_u8 status;
    IP_ADDR ResourceIP;
    zl_u16   ResourcePort;

    zl_u8 Ver;
    zl_u8 FuncSel;
    zl_u8 FuncSel2;

    IP_ADDR UDPGroupIP;

    zl_u8 DevID[ID_LEN];
    zl_u8 IO;
    zl_u8 FuncEn;
    zl_u8 ServerModeParamT;
};
```

读取、修改设备的参数时需要用到参数结构体，其中 status 为 DEV_STATUS_OPENED 时表示联网设备已经建立 TCP 连接，此时可以通过读取 ResourceIP 和 ResourcePort 来获得联网设备的来路 IP 和来路端口。其它各个参数的含义请参考《ZLSN2000 用户手册》的参数设置部分的参数说明小节。所有参数都是网络字节顺序，也就是 zl_u16 类型的数据需要通过 ntohs()函数转化为主机字节顺序。

1.1.4 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 APP_PRO_NONE            0   //转化协议类型定义
#define APP_PRO_MODBUS_RTU      1
#define MODBUS_TCP_SERVER_PORT  502
#define MODBUS_TCP_DATA_START   6

#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 WM_ZLDM_RECV                (WM_USER+1920)      //消息定义
```

```
#define ZLDM_VER 9 //函数库版本号
```

2.2. 库函数定义

ZLDM_Init

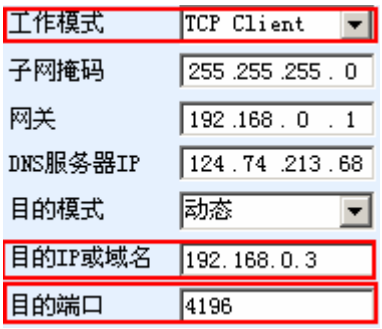
初始化函数库。

```
bool __stdcall ZLDM_Init (
zl_u16 ServerPort
);
```

参数

ServerPort

[in] 运行该设备管理函数库的计算机的监听端口。处于 TCP Client 的设备将向该端口发起连接。如果需要与处于 TCP Client 的设备通信则需要将 *ServerPort* 设置为与设备参数（如下图所示）的目的端口一样的值。由于 4196 是 ZLVirCom 的默认监听端口，使用 4196 时请先关闭 ZLVirCom 程序，或停止 ZLVirCom 的服务。



返回值

TRUE/FALSE，表示初始化是否成功。初始化失败的原因可能是 *ServerPort* 的端口已经被其他程序占用，例如 ZLVirCOM 正在运行时会占用 4196 端口。

描述

使用函数库前需要先进行初始化。

ZLDM_Exit

退出前调用该函数。

```
void __stdcall ZLDM_Exit (
```

```
);
```

参数

描述

该函数用于析构函数库内部变量，如果不调用该函数而退出，可能出现错误提示对话框。

ZLDM_StartSearchDev

搜索网络中所有联网的卓岚联网设备。

```
zl_u16 __stdcall ZLDM_StartSearchDev(
ZLDevHandler hDev[],
zl_u16 hDevSize
);
```

参数

hDev

[out] ZLDevHandler 类型的数组。该数组由调用者分配，数组最小长度建议为，ZLDM_HANDLER_ARRAY_MIN_SIZE，函数返回时数组内容为搜索到的设备的句柄，每一个设备有一个句柄。

hDevSize

[in] *hDev* 数组长度，该长度由调用者分配内存的时候决定。建议 *hDevSize* 大于 256，可以根据用户的 ZLSN 设备数量来调整。

返回值

写入 *hDev* 的句柄数量，当 *hDevSize* 大于等于联网设备数量时，返回值就是找到的设备的数量；当 *hDevSize* 小于联网设备数量时，返回值为 *hDevSize*。

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 设备和用户指定 IP 的设备。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_GetDevParam

获取某个设备的详细参数。

```
zl_bool __stdcall ZLDM_GetDevParam(
ZLDevHandler h,
struct SDevParam *pParam
);
```

参数

h

[in] 需要获取参数的设备对应的句柄，句柄通过 ZLDM_StartSearchDev 函数获得。

pParam

[out] 用于接收返回的参数，该结构对应的内存空间由调用者分配。

返回值

TRUE/FALSE，表示是否成功获取参数，如果 *h* 是一个无效的句柄则会使得获取参数失败。

描述

当使用 ZLDM_StartSearchDev 获得设备句柄列表之后，就可以通过设备对应的句柄获得设备的参数。通过该函数可以一次获得设备的所有参数。

ZLDM_SetDevParam

修改某个设备的详细参数。

```
zl_bool __stdcall ZLDM_SetDevParam(
ZLDevHandler h,
struct SDevParam *pParam
);
```

参数

h

[in] 需要修改参数的设备对应的句柄，句柄通过 ZLDM_StartSearchDev 函数获得。

pParam

[out] 新的参数。

返回值

TRUE/FALSE，表示是否成功修改参数，如果 *h* 是一个无效的句柄则会使得修改参数失败。

描述

当使用 ZLDM_StartSearchDev 获得设备句柄列表之后，可以先读取设备参数，然后针对其中某几个参数进行修改后，调用 ZLDM_SetDevParam 函数将参数写入设备。成功调用该函数后设备将自动重新启动。

ZLDM_DevOpen

打开某个设备，这样就可以和这个设备进行通信。注意打开 TCP Client 端的设备的时候，需要等待联网设备的 LINK 灯（绿灯）亮起后才可以打开，这时因为作为服务器端的 PC 无法主动和联网设备建立通信，而只能够等待联网设备的连接。联网设备的连接的间隔默认为 12 秒，可以通过设备参数“断线重连时间”设置为更短的时间，例如 1 秒。

```
zl_bool __stdcall ZLDM_DevOpen(
ZLDevHandler h,
zl_u32 mode,
zl_u32 hWndCaller
);
```

参数

h

[in] 设备对应的句柄。如果为 ZLDM_HANDLER_BROADCAST 则表示进行 UDP 广播通信。

mode

[in] 设备的通信模式。

如果 h 为 ZLDM_HANDLER_BROADCAST，则 *mode* 的高 16 位表示 UDP 广播的本地接收端口，*mode* 的低 16 位表示 UDP 广播的目的端口。

如果 h 为非 ZLDM_HANDLER_BROADCAST，*mode* 将是模式模式控制字节。*mode* 的值和含义是：

Bit 位	值	含义
0	ZLDM_FAST_SEND	每次发送的数据小（小于 100 字节），但是能够快速发送，实时性高。
	ZLDM_BULK_SEND	每次发送的数据较大（例如 1000 字节以上）。
1	ZLDM_CALL_MESSAGE	参数 *hWndCaller* 是窗口句柄

	ZLDM_CALL_BACK	参数 *hWndCaller* 是回调函数
2～31		保留

hWndCaller

[in] 根据 *mode* 的不同可以是

1）用户程序的窗口句柄，*hWndCaller* 可用于实现接收数据时的事件通知机制，如果用户不需要该机制可以设置该值为 0。为了读取接收的数据，用户可以在一个线程中不断调用 ZLDM_DevRead 读取数据。但是该方式占用较多的 CPU 资源，且实现起来不方便，为此卓岚提供了接收数据时的事件通知机制。*hWndCaller* 是实现该机制必要的一个参数，该值为用户程序窗口（例如对话框）的窗口句柄，在 MFC 中窗口句柄即是窗口类的 m_hWnd 成员变量。

2）回调函数。回调函数，提供了另外一种方便的接收数据通知机制。回调函数类型定义为：

```
typedef zl_s32 (__stdcall * tZLDM_RecvCallBack)(zl_u32 w, zl_s32 l);
```

首先用户需要定义一个回调函数，然后将该回调函数通过参数 *hWndCaller* 传递给 DLL 库。在该 h 对应的设备有数据接收的时候将会调用该回调函数。

返回值

TRUE/FALSE，表示是否成功打开设备，目前只支持打开工作于 TCP Server 模式的设备，其他工作模式将会导致打开失败。

描述

在使用 ZLDM_DevRead 和 ZLDM_DevWrite 和设备通信之前首先需要打开设备。

ZLDM_DevOpen 是阻塞模式的，即如果服务器无法和 PC 建立连接，ZLDM_DevOpen 最多等待 2 秒后返回。

ZLDM_DevClose

关闭设备。

```
zl_bool __stdcall ZLDM_DevClose(
ZLDevHandler h
);
```

参数

h

[in] 设备对应的句柄。如果为 ZLDM_HANDLER_BROADCAST 表示关闭 UDP 广播通信。

返回值

TRUE/FALSE，表示是否成功关闭设备。

描述

关闭设备相当于与设备断开连接。

ZLDM_DevWrite

向设备发送数据。

```
int __stdcall ZLDM_DevWrite(
ZLDevHandler h,
const void *buf,
int size
);
```

参数

h

[in] 设备对应的句柄。如果为 ZLDM_HANDLER_BROADCAST 表示发送 UDP 广播包。

buf

[in] 需要发送的数据的首地址。

size

[in] 需要发送的数据量。

返回值

如果有错误发生，返回-1，否则返回实际发送的数据量大小。

描述

在 ZLDM_DevOpen 成功返回以后，可以发送数据。

ZLDM_DevRead

接收设备发送过来的数据。

```
int __stdcall ZLDM_DevRead(
ZLDevHandler h,
void *buf,
int size
);
```

参数

h

[in] 设备对应的句柄。如果为 ZLDM_HANDLER_BROADCAST 表示接收 UDP 广播数据。

buf

[in] 由调用者分配的内存空间，用于接收数据。

size

[in] *buf* 的大小。

返回值

如果有错误发生，返回-1，否则返回实际接收的数据量大小。

描述

ZLDM_DevRead 是一个非阻塞函数，如果没有数据可以接收，函数也会立即返回。如果有有效数据接收，返回值应该大于等于 0。

ZLDM_GetVer

获取 ZLDevManage.dll 函数库的版本号。

```
int __stdcall ZLDM_GetVer(
);
```

参数

无

返回值

返回一个 int 型数据表示版本号。

描述

获取的版本号应该和 ZLDevManage.h 中的 ZLDM_VER 相同，否则说明头文件和 DLL 库不匹配。

ZLDM_AddManualDev

手动添加设备 IP。

```
zl_bool __stdcall ZLDM_AddManualDev(
const char ip_dns[ ],
zl_u16 port
);
```

参数

ip_dns

[in] 需要添加的设备的 IP 地址字符串或者域名字符串长度小于 DNS_NAME_MAX_LEN。

port

[in] 需要添加的设备的监听端口。

返回值

TRUE/FALSE，设备是否添加成功，如果该 IP 已经添加过，则会返回 FALSE。

描述

如果联网设备和服务器不处于一个物理子网中，需要手动添加此类设备的 IP，否则搜索设备功能将无法搜索到子网以外的设备。

ZLDM_ClearManualDev

删除由 ZLDM_AddManualDev()添加的设备。

```
void __stdcall ZLDM_ClearManualDev (
);
```

参数

描述

删除由 ZLDM_AddManualDev()添加的设备。

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。

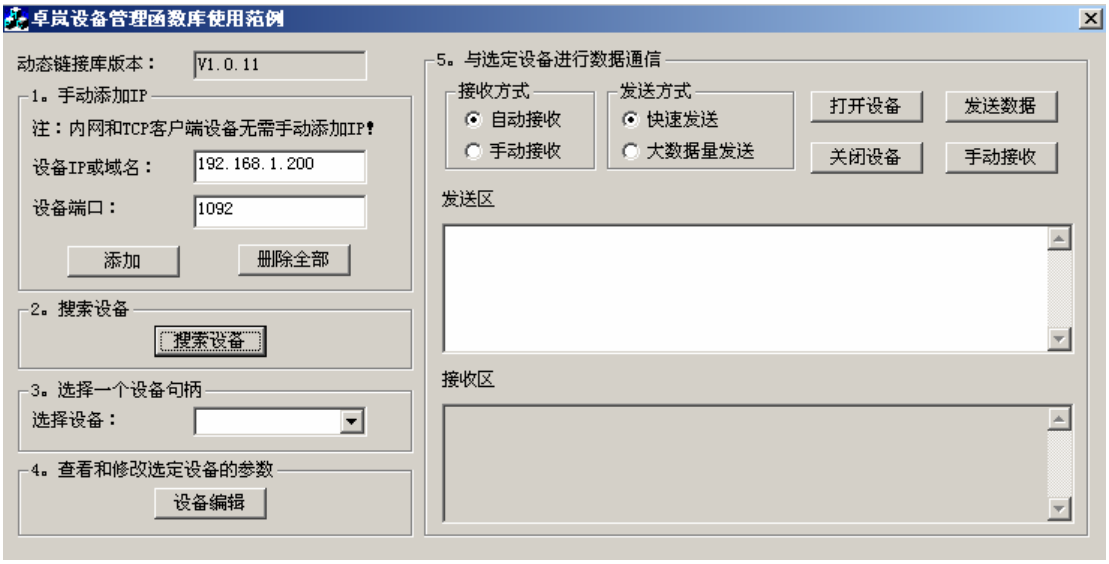


图 1 ZLUseDevManage 程序界面

范例程序界面如图 1 所示。该范例的功能有：

1. 手动添加 IP 与域名：只有外网的 TCP Server 模式的设备才需要手动添加，否则搜索设备功能将无法搜索到该类设备。
2. 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、外网 TCPClient 模式的设备和手动添加的外网 TCP Server 模式设备。搜索结果将返回一个设备列表（即：设备句柄列表）。
3. 选择一个设备：设备管理函数库将每一个设备映射为一个句柄，用户修改设备参数、与设备通信等功能都是通过设备句柄进行的。
4. 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。
5. 与选定设备进行数据通信：卓岚设备管理函数库封装了网络通信功能，用户和设备进行数据通信就和一般的文件操作一样，只需要打开设备、读设备、写设备、关闭设备即可。

3.2. 加载函数库

使用函数库之前需要将 ZLDevManage.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 ZLDevManage.dll 拷贝到用户工程文件夹中。用户程序发布时，也需要将

ZLDevManage.dll 文件放在用户程序同一文件夹中。

2．将 ZLDevManageDll.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("ZLDevManage.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_Init     = (tZLDM_Init)GetProcAddress(m_hZLDevManage,"ZLDM_Init");
m_pZLDM_Exit     = (tZLDM_Exit)GetProcAddress(m_hZLDevManage,"ZLDM_Exit");
m_pZLDM_StartSearchDev = (tZLDM_StartSearchDev)GetProcAddress(m_hZLDevManage,"ZLDM_StartSearchDev");
m_pZLDM_GetDevParam = (tZLDM_GetDevParam)GetProcAddress(m_hZLDevManage,"ZLDM_GetDevParam");
m_pZLDM_SetDevParam = (tZLDM_SetDevParam)GetProcAddress(m_hZLDevManage,"ZLDM_SetDevParam");
m_pZLDM_DevOpen = (tZLDM_DevOpen)GetProcAddress(m_hZLDevManage,"ZLDM_DevOpen");
m_pZLDM_DevClose = (tZLDM_DevClose)GetProcAddress(m_hZLDevManage,"ZLDM_DevClose");
m_pZLDM_DevWrite = (tZLDM_DevWrite)GetProcAddress(m_hZLDevManage,"ZLDM_DevWrite");
m_pZLDM_DevRead = (tZLDM_DevRead)GetProcAddress(m_hZLDevManage,"ZLDM_DevRead");
m_pZLDM_GetVer = (tZLDM_GetVer)GetProcAddress(m_hZLDevManage,"ZLDM_GetVer");
m_pZLDM_AddManualDev = (tZLDM_AddManualDev)GetProcAddress(m_hZLDevManage,"ZLDM_AddManualDev");
m_pZLDM_ClearManualDev = (tZLDM_ClearManualDev)GetProcAddress(m_hZLDevManage,"ZLDM_ClearManualDev");
```

其中各个函数指针的定义为：

```
tZLDM_Init          m_pZLDM_Init;
tZLDM_Exit          m_pZLDM_Exit;
tZLDM_StartSearchDev m_pZLDM_StartSearchDev;
tZLDM_GetDevParam   m_pZLDM_GetDevParam;
tZLDM_SetDevParam   m_pZLDM_SetDevParam;
tZLDM_DevOpen       m_pZLDM_DevOpen;
tZLDM_DevClose      m_pZLDM_DevClose;
tZLDM_DevWrite      m_pZLDM_DevWrite;
tZLDM_DevRead       m_pZLDM_DevRead;
tZLDM_GetVer        m_pZLDM_GetVer;
tZLDM_AddManualDev  m_pZLDM_AddManualDev;
tZLDM_ClearManualDev    m_pZLDM_ClearManualDev;
```

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_GetVer)();
```

3.3. 初始化函数库

使用函数库前必须初始化函数库，全局只需要进行一次初始化。

```
(*m_pZLDM_Init)(port);
```

初始化失败的原因可能是 *Port* 端口已经被其他程序占用，例如 ZLVirCOM 正在运行时会占用 4196 端口。

3.4. 退出函数库

在用户程序退出前，一般需要先退出函数库。

```
(*m_pZLDM_Exit)();
```

3.5. 搜索设备

使用 ZLDM_StartSearchDev 可以搜索物理子网、手动添加的外网设备、处于 Client 模式向服务器连接的外网设备。

```
m_DevCnt = (*m_pZLDM_StartSearchDev)(m_h, ZLDM_HANDLER_ARRAY_MIN_SIZE);
```

这里 m_h 是 ZLDevHandler 类型的数组，ZLDM_HANDLER_ARRAY_MIN_SIZE 是数组的大小。搜索到的设备句柄保存在 m_h 中，m_DevCnt 保存设备的个数。

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句柄。

3.6. 参数操作

可以如下方法读取某个设备的参数：

```
struct SDevParam param;
(*m_pZLDM_GetDevParam)(m_h[m_DevHandlerList.GetCurSel()], &param);
```

其中 m_h[m_DevHandlerList.GetCurSel()]表示当前选择的设备的句柄。

修改参数时，建议先读取参数到 param 结构体中，然后修改其中几个参数，再调用 ZLDM_SetDevParam 进行修改：

```
(*m_pZLDM_SetDevParam)(m_h[m_DevHandlerList.GetCurSel()], &param);
```

修改成功后设备将自动重新启动。

3.7. 数据通信

一般情况下用户可以通过 ZLVirCom 的虚拟串口和 socket 编程与卓岚嵌入式联网设备通信。对于 socket 方案，需要一定的网络编程基础，相对较为复杂；

另外用户需要维护一个联网设备的 IP 地址列表。卓岚的设备管理函数库简化了与设备数据通信的方法。用户只需要像文件读写一样地操作设备即可实现通信，并且可以实现数据接收的事件触发机制；用户不需要维护设备 IP 列表，只需要保存设备句柄即可实现通信，与器件搜索、参数修改可以很好地统一起来。

1.1.5 打开设备

与文件读写一样，联网设备的读写也需要先打开设备。打开设备的方法是：

```
if(m_ModeAutoRecv == 0)
    hWnd = m_hWnd;
else
    hWnd = NULL;
if(m_ModeFastSend == 0)
    mode = ZLDM_FAST_SEND;
else
    mode = ZLDM_BULK_SEND;
(*m_pZLDM_DevOpen)(m_h[m_DevHandlerList.GetCurSel()], mode, hWnd);
```

这里 mode 根据用户选择的是快速发送还是大数据量发送进行设置，hWnd 参数用于实现数据自动接收的事件通知，将在后面详述；不需要自动接收功能时，可以传入 NULL。

1.1.6 关闭设备

关闭设备的方法是：

```
(*m_pZLDM_DevClose)(m_h[m_DevHandlerList.GetCurSel()])
```

关闭设备时会与 TCP Server 方式的设备断开 TCP 连接，但是与 TCP Client 方式的设备不会断开连接。

1.1.7 发送数据

发送数据的操作相当于向文件中写入数据，发送数据代码如下：

```
(*m_pZLDM_DevWrite)(m_h[m_DevHandlerList.GetCurSel()], (LPCSTR)m_SendText, m_SendText.GetLength());
```

1.1.8 手动接收数据

在设备打开以后用户可以随时调用 ZLDM_DevRead()，从返回值可以判断是否有数据到来。一个最简单的方法就是：

```
while((recv_size = (*m_pZLDM_DevRead)(m_h[m_DevHandlerList.GetCurSel()], buf, 1024)) <= 0);
```

但是该方式会占用几乎 100%的 CPU 时间，效率很低。另一个选择是用户新建一个线程，在线程中运行以上语句，不过此方法也要占用近 50%的 CPU 时间。

另一个选择是采用如下代码：

```
        while((recv_size = (*m_pZLDM_DevRead)(m_h[m_DevHandlerList.GetCurSel()], buf, 1024)) <= 0)
        {
            Sleep(2);
            if(cnt ++ >= 10000)
                break;
        }
        if(cnt >= 10000)
        {
            AfxMessageBox("没有数据可以接收!");
        }
        else
        {
            AfxMessageBox("接收成功!");
        }
```

此处在 while 循环内部使用 Sleep()来休眠进程，这样可以将大部分的 CPU 时间空余出来。用户可以在一个线程中执行以上代码。

1.1.9 自动接收数据

手动接收数据需要程序不断调用 ZLDM_DevRead()，这增加了程序的复杂性，另外也占用了 CPU 资源。设备函数库提供了在有数据可接收时的事件通知机制，方便了数据的接收。当有数据可接收时，内部机制自动调用用户函数(范例中为 OnZLDMRecv()函数)，在用户函数中可以调用 ZLDM_DevRead()进行接收。

以下介绍在 VC++6.0 环境下增加自动接收用户函数的方法：

1．在打开设备的时候将窗口或者对话框的窗口句柄作为参数传入

```
hWnd = m_hWnd;        //窗口句柄 CWnd::m_hWnd
(*m_pZLDM_DevOpen)(m_h[m_DevHandlerList.GetCurSel()], mode, hWnd);
```

2．在窗口或对话框类 ZLUseDevManageDlg.h 内增加消息函数声明：

```
class CZLUseDevManageDlg : public CDialog
{
public:
    //{{AFX_MSG(CZLUseDevManageDlg)
    afx_msg LONG OnZLDMRecv(WPARAM w,LPARAM l); //增加的代码
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
```

3．在 ZLUseDevManageDlg.cpp 消息映射列表中增加 WM_ZLDM_RECV 消息

```
BEGIN_MESSAGE_MAP(CZLUseDevManageDlg, CDialog)
    //{{AFX_MSG_MAP(CZLUseDevManageDlg)
    ON_MESSAGE(WM_ZLDM_RECV, OnZLDMRecv)        //增加的代码
```

```
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
```

4．编写 OnZLDMRecv 函数，在函数内调用 ZLDM_DevRead()进行接收，其中参数 w 为发送该数据的设备的句柄。

```
LONG CZLUseDevManageDlg::OnZLDMRecv(WPARAM w, LPARAM l)
{
    CString str;
    str.Format("设备%x 接收到数据", w);
    (*m_pZLDM_DevRead)(m_h[m_DevHandlerList.GetCurSel()], buf, 1024));    //接收
    return 0;
}
```

3.8. 获取版本号

获取版本号的的方法如下：

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

3.9. 手动添加外网设备

如果联网设备和服务器不处于一个物理子网中且是 TCP Server 设备，需要手动添加此类设备的 IP，否则搜索设备功能将无法搜索到子网以外的设备。添加方法如下：

```
(*m_pZLDM_AddManualDev)(ip);
```

删除所有手动添加的设备的方法是：

```
(*m_pZLDM_ClearManualDev)()
```

4. 使用实例

1. 重启设备：任何时候调用 ZLDM_SetDevParam 函数都会引起设备重启。
2. 读取 MAC 地址：使用结构体中的 DevID 可以获得 MAC 地址。

3. UDP 模式设备通信

V1.0.12 版本的设备管理函数库可以和工作于 UDP 模式的设备通信。步骤如下：

1．加载函数库。

2．初始化函数库，调用 ZLDM_Init 函数;

3．如果有处于外网（比如 internet）的 UDP 模式的设备，则需要调用 ZLDM_AddManualDev 函数添加设备；否则跳过此步骤。

4．搜索设备，调用 ZLDM_StartSearchDev 函数获得设备句柄。

5．打开设备，调用 ZLDM_OpenDev 打开设备，函数参数是设备句柄。打开当前设备后，可以和当前设备通信。

a) 注意所有的 UDP 模式的设备都应该具有相同的目的 IP、目的端口，若某设备的该参数和当前设备不同，则无法和动态库通信。

b) 如果有多个设备只需要调用一次 ZLDM_OpenDev 即可，不用对所有的设备额调用 ZLDM_OpenDev。

c) 打开设备后，支持接收、发送 UDP 广播数据包。

d) 从返回值是否为 1 判断设备打开是否成功。注意 UDP 模式的设备的目的端口不要设置为 4196，否则打开设备会失败，因为该端口已经被系统使用。

6．发送数据，调用 ZLDM_DevWrite 发送数据。

7．接收数据，调用 ZLDM_DevRead 接收数据。或者采用自动接收机制，查看“自动接收数据”章节。

a) 当从设备 h 接收到数据时，自动接收方式下将调用 OnZLDMRecv(WPARAM w, LPARAM l)，其中参数 w 就是设备句柄。

b) 调用 ZLDM_DevRead 时，参数 h（设备句柄）必须为发送数据的设备，否则无法接收。

8．退出函数库，调用 ZLDM_Exit 函数。

4. UDP 广播通信

UDP 广播通信功能，让用户可以通过调用 DLL 库发送 UDP 广播给联网设备，同时也可以接收联网设备发来的广播数据。

具体使用方式是：

1．使用 h= ZLDM_HANDLER_BROADCAST，调用 ZLDM_Dev_Open，并且设置参数 mode 的高 16bit 为本地 UDP 用于接收广播的端口，mode 低 16bit 为远端的 UDP 广播接收端口（PC 发送的目的端口）。

2．使用 h= ZLDM_HANDLER_BROADCAST，调用 ZLDM_Dev_Write，即可发送广播数据。

3．使用 h= ZLDM_HANDLER_BROADCAST，调用 ZLDM_Dev_Read，即可接

收广播数据。同时类似非广播通信方式的自动接收方法，设置自动接收。

4．使用 h= ZLDM_HANDLER_BROADCAST，调用 ZLDM_Dev_Close，关闭广播接收，之后不再接收广播数据。

5. 版本升级方法

设备管理函数库随着函数库功能的增加版本也随着升级，用户从低版本的动态库升级为高版本的动态库的方法如下：

1．使用新的 ZLDevManageDll.h，其中：

a) struct SDevParam 结构体可能增加新的参数。

b) ZLDM_VER 会增加。

2．使用新的 ZLDevManage.lib 和 ZLDevManage.dll 库文件。

6. 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