

iCAN 主站函数库使用手册

V2.91

目 录

第 1 章	函数库说明	1
1.1	主站卡类型	1
1.2	函数调用错误码	1
1.3	数据结构说明	1
1.3.1	ROUTECFG	1
1.4	iCAN 网络控制函数说明	2
1.4.1	Mgr_AddRoute	2
1.4.2	Mgr_DelRoute	2
1.4.3	Mgr_DelAllRoute	2
1.4.4	Mgr_StartSys	3
1.4.5	Mgr_StopSys	3
1.4.6	Mgr_IsStarted	3
1.5	iCAN 网络函数说明	3
1.5.1	Route_AddSlave	3
1.5.2	Route_DelSlave	4
1.5.3	Route_DelAllSlave	4
1.5.4	Route_SetConfig	4
1.5.5	Route_GetConfig	4
1.5.6	Route_GetSlavebyID	5
1.5.7	Route_ConnectAllSlaveAsync	5
1.5.8	Route_IOCTL	5
1.6	iCAN 网络控制从站函数说明	6
1.6.1	Slave_Connect	6
1.6.2	Slave_Disconnect	7
1.6.3	Slave_IsConnected	7
1.6.4	Slave_GetID	7
1.6.5	Slave_GetIODataLen	7
1.6.6	Slave_GetAIData	8
1.6.7	Slave_GetDIDData	8
1.6.8	Slave_GetData	9
1.6.9	Slave_SendData	9
1.6.10	Slave_SetConfig	9
1.6.11	Slave_GetConfig	10
1.6.12	Slave_SetCycle	11
1.6.13	Slave_GetCycle	11
1.6.14	Slave_GetTriggerData	11
第 2 章	函数使用	13
2.1	系统操作流程	13
第 3 章	使用举例	14
3.1	如何添加 iCAN 网络	14

3.2	如何添加从站.....	14
3.3	如何启动系统.....	14
3.4	如何连接从站并向其发送数据和接收数据.....	14
3.5	如何判断从站是否已经连接.....	15
3.6	详细使用例程.....	15
第 4 章	产品服务.....	16
4.1	软件升级.....	16
4.2	技术支持.....	16

第 1 章...函数库说明

1.1 主站卡类型

可作为 iCAN 网络主站卡的 ZLGCAN 系列接口卡类型如下：

设备名称	设备类型号
PCI5121	1
PCI9810	2
USBCAN1	3
USBCAN2	4
PCI9820	5
PCI5110	7
CANlite(CANmini)	8
ISA9620	9
ISA5420	10
PC104-CAN	11
CANET-UDP	12
DNP9810	13
PCI9840	14
PC104-CAN2	15
PCI-9820I	16
CANET-TCP	17
PEC-9920	18
PCIE-9220	18(与 PEC_9920 同)
PCI-5010-U	19
USBCAN-E-U	20
USBCAN-2E-U	21
PCI-5020-U	22
EG20T-CAN	23
ICANCFGCARD	30

ICANCFGCARD 特别说明：该型号设备是虚拟的 CAN 设备，用于当 iCAN 设备是 HID 设备时，*icandll.dll* 动态库可通过 USB 配置 iCAN 设备。对应的动态库是 *kerneldlls* 文件夹中的 *hidVicCan.dll*。

1.2 函数调用错误码

名称	值	描述
ICANOK	0x00000000	操作正确
ICANERR_FUNCNOTEXIST	0x00000001	功能码不存在
ICANERR_SRCNOTEXIST	0x00000002	资源不存在
ICANERR_CMDNOTSUPPORT	0x00000003	命令不支持
ICANERR_CMDILLEGAL	0x00000004	参数非法
ICANERR_CONNECTNOTEXIST	0x00000005	连接不存在

ICANERR	0x000000f1	不确定的错误
ICANERR_USING	0x000000f3	资源被占用
ICANERR_SETCAN	0x000000f4	打开设备失败或初始化 CAN 失败
ICANERR_SRVSTARTED	0x000000f5	服务已启动，无法进行此项操作
ICANERR_TIMEOUT	0x000000f6	操作超时
ICANERR_ITEMEXIST	0x000000f7	目标已经存在
ICANERR_SLAVENOTEXIST	0x000000f8	从站不存在
ICANERR_ROUTENOTEXIST	0x000000f9	主站网络不存在
ICANERR_SYSNOTSTARTED	0x000000fa	系统未启动
ICANERR_BUFNOTEXIST	0x000000fb	函数接口传递缓冲区指针错误

1.3 数据结构说明

1.3.1 ROUTECFG

此数据结构用来设置初始化 iCAN 网络所需要的一些必要参数，声明如下：

```
typedef struct _tagRouteCfg
{
    int iCardType;//CAN 接口卡类型
    int iCardInd;//CAN 接口卡序号
    int iCANInd;//CAN 路数
    WORD wCANBaud;//0x311c - 10K,0x041c - 100K,0x001c - 500K
    WORD wMasterID;//iCAN 网络主站 ID，iCAN 网络中主站和从站的地址范围为从 0 到 255
    int iMasterCycle;//iCAN 网络主站定时循环参数，范围 1 到 255，单位为 10ms
}ROUTECFG;
```

1.4 iCAN 网络控制函数说明

1.4.1 Mgr_AddRoute

描述

调用此函数添加一个新的 iCAN 网络到 iCAN 系统中。

```
DWORD __stdcall Mgr_AddRoute(ROUTECFG cfg, HANDLE* phRoute);
```

参数

cfg

所要添加的 iCAN 网络初始化参数。

phRoute

新 iCAN 网络句柄指针，用以存储返回的新 iCAN 网络句柄。

返回值

正确为 ICANOK，否则为错误码。

1.4.2 Mgr_DelRoute

描述

调用此函数从 iCAN 系统中删除一个 iCAN 网络。

```
DWORD __stdcall Mgr_DelRoute(HANDLE hRoute);
```

参数

hRoute

所要删除的 iCAN 网络句柄。

返回值

正确为 ICANOK，否则为错误码。

1.4.3 Mgr_DelAllRoute

描述

调用此函数删除 iCAN 系统中所有 iCAN 网络。

```
DWORD __stdcall Mgr_DelAllRoute();
```

参数

无。

返回值

正确为 ICANOK，否则为错误码。

1.4.4 Mgr_StartSys

描述

调用此函数启动 iCAN 系统。

```
DWORD __stdcall Mgr_StartSys();
```

参数

无。

返回值

正确为 ICANOK，否则为错误码。

1.4.5 Mgr_StopSys

描述

调用此函数停止 iCAN 系统。

```
DWORD __stdcall Mgr_StopSys();
```

参数

无。

返回值

正确为 ICANOK，否则为错误码。

1.4.6 Mgr_IsStarted

描述

调用此函数判断 iCAN 系统是否已经启动。

```
DWORD __stdcall Mgr_IsStarted();
```

参数

无。

返回值

已经启动为 1，否则为 0。

1.5 iCAN 网络函数说明

1.5.1 Route_AddSlave

描述

调用此函数往指定 iCAN 网络中添加一个从站。

```
DWORD __stdcall Route_AddSlave(HANDLE hRoute, DWORD SlaveID, HANDLE* phSlave);
```

参数

hRoute

指定要添加从站的 iCAN 网络句柄。

SlaveID

要添加的从站 ID。

phSlave

新从站句柄指针，用以存储返回的新从站句柄。

返回值

正确为 ICANOK，否则为错误码。

1.5.2 Route_DelSlave

描述

调用此函数从指定 iCAN 网络中删除一个从站。

```
DWORD __stdcall Route_DelSlave(HANDLE hRoute, HANDLE hSlave);
```

参数

hRoute

指定要删除从站的 iCAN 网络句柄。

hSlave

要删除的从站句柄。

返回值

正确为 ICANOK，否则为错误码。

1.5.3 Route_DelAllSlave

描述

调用此函数从指定 iCAN 网络中删除所有从站。

```
DWORD __stdcall Route_DelAllSlave(HANDLE hRoute);
```

参数

hRoute

指定要删除从站的 iCAN 网络句柄。

返回值

正确为 ICANOK，否则为错误码。

1.5.4 Route_SetConfig

描述

调用此函数设置指定 iCAN 网络配置参数。

```
DWORD __stdcall Route_SetConfig(HANDLE hRoute, ROUTECFG cfg);
```

参数

hRoute

指定 iCAN 网络句柄。

cfg

配置参数。

返回值

正确为 ICANOK，否则为错误码。

1.5.5 Route_GetConfig

描述

调用此函数获取指定 iCAN 网络配置参数。

```
DWORD __stdcall Route_GetConfig(HANDLE hRoute, ROUTECFG* pcfg);
```

参数

hRoute

指定 iCAN 网络句柄。

pcfg

配置参数指针，存储返回的配置参数。

返回值

正确为 ICANOK，否则为错误码。

1.5.6 Route_GetSlavebyID

描述

调用此函数获取指定 iCAN 网络从站句柄。

```
DWORD __stdcall Route_GetSlavebyID(HANDLE hRoute, DWORD SlaveID, HANDLE* phSlave);
```

参数

hRoute

指定 iCAN 网络句柄。

SlaveID

指定从站 ID。

phSlave

从站句柄指针，存储返回的从站句柄。

返回值

正确为 ICANOK，否则为错误码。

1.5.7 Route_ConnectAllSlaveAsync

描述

调用此函数连接所有已添加从站，此操作为异步操作，调用此函数后可调用 Slave_IsConnected 函数来查询从站是否已连接。

```
DWORD __stdcall Route_ConnectAllSlaveAsync(HANDLE hRoute);
```

参数

hRoute

指定 iCAN 网络句柄。

返回值

正确为 ICANOK，否则为错误码。

1.5.8 Route_IOCTL

描述

调用此函数发送控制命令。

```
DWORD __stdcall Route_IOCTL(HANDLE hRoute, DWORD code, BYTE* pinbuff=NULL,
                             int inlen=0, BYTE* poutbuff=NULL, int outlen=0);
```

参数

hRoute

指定 iCAN 网络句柄。

code

控制码。

pinbuff

输入参数缓冲区。

inlen

输入缓冲区长度。

poutbuf

输出参数缓冲区。

outlen

输出缓冲区长度。

返回值

正确为 ICANOK，否则为错误码。

注：控制码列表如下

设备类型	描述	code	pinbuff	inlen(bytes)	poutbuf	outlen(bytes)
CANET-UDP/ CANET-TCP	设置本地端口	1	32 位长整型缓冲区 指针，存储本地端 口号	4	NULL	0
	设置 CANET IP 地址	2	4 字节数组指针， 存储 CANET IP 地址	4	NULL	0
	设置 CANET 工 作端口	3	32 位长整型缓冲区 指针，存储 CANET 工作端口	4	NULL	0
	启动此网络， 当系统启动后 才调用 Mgr_AddRoute 函数添加 CANET 时，设置 完 CANET 参数 后需要发送此 命令来启动此	4	NULL	0	NULL	0

	网络，如果在系统没有启动时添加 CANET 则不需要发送此命令					
	此命令在 CANET-TCP 时有效，用于设置其工作方式	6	32 位长整型缓冲区 指针： 0 为客户端 1 为服务器	4	NULL	0
PCI-5010-U PCI-5020-U USBCAN-E-U USBCAN-2E-U	设置波特率，这几个设备的波特率设置比较特殊，在调用 Mgr_AddRoute 之后还需要调用 Route_I0Ctl 设置一下波特率	5	32 位长整型缓冲区 指针： 0x060003 为 1000Kbps 0x060004 为 800Kbps 0x060007 为 500Kbps 0x1C0008 为 250Kbps 0x1C0011 为 125Kbps 0x160023 为 100Kbps 0x1C002C 为 50Kbps 0x1600B3 为 20Kbps 0x1C00E0 为 10Kbps	4	NULL	0

1.6 iCAN 网络控制从站函数说明

1.6.1 Slave_Connect

描述

调用此函数连接指定从站。

```
DWORD __stdcall Slave_Connect(HANDLE hSlave);
```

参数

hSlave

指定从站句柄。

返回值

正确为 ICANOK，否则为错误码。

1.6.2 Slave_Disconnect

描述

调用此函数断开指定从站。

```
DWORD __stdcall Slave_Disconnect(HANDLE hSlave);
```

参数

hSlave

指定从站句柄。

返回值

正确为 ICANOK，否则为错误码。

1.6.3 Slave_IsConnected

描述

调用此函数判断指定从站是否已经连接。

```
DWORD __stdcall Slave_IsConnected(HANDLE hSlave);
```

参数

hSlave

指定从站句柄。

返回值

已连接为 1，否则为 0。

1.6.4 Slave_GetID

描述

调用此函数获取指定从站 ID。

```
DWORD __stdcall Slave_GetID(HANDLE hSlave);
```

参数

hSlave

指定从站句柄。

返回值

正确为 ICANOK，否则为错误码。

1.6.5 Slave_GetIODataLen

描述

调用此函数获取指定从站 IO 数据长度。

```
DWORD __stdcall Slave_GetIODataLen(HANDLE hSlave, DWORD *pDI, DWORD* pDO, DWORD* pAI, DWORD* pAO);
```

参数

hSlave

指定从站句柄。

pDI

存储 DI 数据长度。

pDO

存储 DO 数据长度。

pAI

存储 AI 数据长度。

pAO

存储 AO 数据长度。

返回值

正确为 ICANOK，否则为错误码。

1.6.6 Slave_GetAIData

描述

调用此函数获取指定从站 AI 数据。

```
DWORD __stdcall Slave_GetAIData(HANDLE hSlave, BYTE* pRecbuf, DWORD* pReclen);
```

参数

hSlave

指定从站句柄。

pRecbuf

接收数据缓冲区指针，接收缓冲区大小必须大于或等于 AI 数据字节长度。

pReclen

输入为接收缓冲区长度，输出为接收到的数据长度。

返回值

正确为 ICANOK，否则为错误码。

1.6.7 Slave_GetDIData

描述

调用此函数获取指定从站 DI 数据。

```
DWORD __stdcall Slave_GetDIData(HANDLE hSlave, BYTE* pRecbuf, DWORD* pReclen);
```

参数

hSlave

指定从站句柄。

pRecbuf

接收数据缓冲区指针，接收缓冲区大小必须大于或等于 DI 数据字节长度。

pReclen

输入为接收缓冲区长度，输出为接收到的数据长度。

返回值

正确为 ICANOK，否则为错误码。

1.6.8 Slave_GetData

描述

调用此函数从指定从站接收数据。

```
DWORD __stdcall Slave_GetData(HANDLE hSlave, DWORD SourceID, BYTE* pRecbuf, DWORD *pReclen);
```

参数

hSlave

指定从站句柄。

SourceID

资源 ID。DI 最大长度为 32 字节，资源 ID 范围 0x00 到 0x1f；AI 最大长度为 32 字节，资源 ID 范围 0x40 到 0x5f。

pRecbuf

接收数据缓冲区指针。

pReclen

输入为接收缓冲区长度，输出为接收到的数据长度。

返回值

正确为 ICANOK，否则为错误码。

1.6.9 Slave_SendData

描述

调用此函数向指定从站发送数据。

```
DWORD __stdcall Slave_SendData(HANDLE hSlave, DWORD SourceID, BYTE* pSendbuf, DWORD Sendlen);
```

参数

hSlave

指定从站句柄。

SourceID

资源 ID。DO 最大长度为 32 字节，资源 ID 范围 0x20 到 0x3f；AO 最大长度为 32 字节，资源 ID 范围 0x60 到 0x7f。

pSendbuf

发送数据缓冲区。

Sendlen

发送数据长度。

返回值

正确为 ICANOK，否则为错误码。

1.6.10 Slave_SetConfig

描述

调用此函数对从站进行配置操作。

```
DWORD __stdcall Slave_SetConfig(HANDLE hSlave, DWORD SourceID, DWORD* pSubsourceID, BYTE* pSendbuf, DWORD Sendlen);
```

参数

hSlave

指定从站句柄。

SourceID

资源 ID。

pSubsourceID

资源 ID 子索引号指针，为 NULL 时表示没有子索引号。

pSendbuf

发送数据缓冲区。

Sendlen

发送数据长度。

返回值

正确为 ICANOK，否则为错误码。

注：配置资源 ID 列表，RO 表示只读，R/W 表示可读写

SourceID	Bytes	Function	Attrib	Description	SubsourceID
0xE0~0xE1	2	Vendor ID	RO	厂商代码，固定值	-
0xE2~0xE3	2	Product Type	RO	产品类型，固定值	-
0xE4~0xE5	2	Product Code	RO	产品型号，固定值	-
0xE6~0xE7	2	Hardware Version	RO	产品硬件版本	-
0xE8~0xE9	2	Firmware Versin	RO	产品固件版本	-
0xEA~0xED	4	Serial Number	RO	4 字节产品 SN 号码	-
0xEE	1	MAC ID	R/W	本机节点的 ID 编号	
0xF5	1	CyclicMaster	R/W	主站通讯定时参数 时间单位为：10ms	-
0xF7	1	Master MAC ID	R/W	主站 MAC ID	-
0xFA~0xFF	6	Reserve	-	-	-

1.6.11 Slave_GetConfig

描述

调用此函数获取从站配置。

```
DWORD __stdcall Slave_GetConfig(HANDLE hSlave, DWORD SourceID, DWORD* pSubsourceID,
                                BYTE* pRecbuf, DWORD Reclen);
```

参数

hSlave

指定从站句柄。

SourceID

资源 ID。

pSubsourceID

资源 ID 子索引号。

pRecbuf

接收数据缓冲区。

Reclen

接收数据长度。

返回值

正确为 ICANOK，否则为错误码。

1.6.12 Slave_SetCycle

描述

设置从站的定时循环周期，默认为 ROUTECFG.iMasterCycle。

```
DWORD __stdcall Slave_SetCycle(HANDLE hSlave, DWORD dwCycle);
```

参数

hSlave

指定从站句柄。

dwCycle

定时循环周期，范围 1 到 255，单位 10ms。

返回值

正确为 ICANOK，否则为错误码。

1.6.13 Slave_GetCycle

描述

获取从站的定时循环周期。

```
DWORD __stdcall Slave_GetCycle(HANDLE hSlave);
```

参数

hSlave

指定从站句柄。

返回值

返回从站的定时循环周期。

1.6.14 Slave_GetTriggerData

描述

获取从站的触发数据。

```
DWORD __stdcall Slave_GetTriggerData(HANDLE hSlave, DWORD* pSourceIDSt,  
    BYTE* pRecbuf, DWORD* pRecLen);
```

参数

hSlave

指定从站句柄。

pSourceIDSt

存储主动上传数据的资源 ID。范围为 0x00 到 0x1f 为 DI 数据，0x40 到 0x5f 为 AI 数据。

pRecbuf

接收数据缓冲区指针。

pRecLen

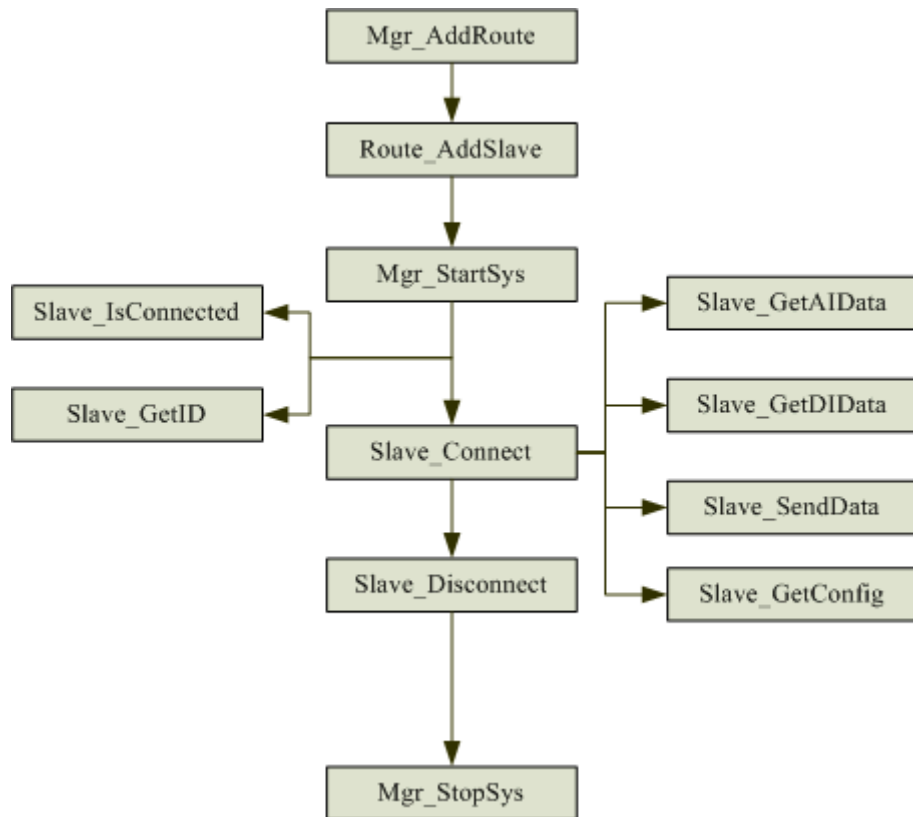
存储接收到的数据长度。

返回值

正确为 ICANOK，否则为错误码。

第2章...函数使用

2.1 系统操作流程



第 3 章...使用举例

3.1 如何添加 iCAN 网络

```
HANDLE hRoute;
ROUTECFG cfg;
cfg.iCardType=5;//PCI9820
cfg.iCardInd=0;//卡序号
cfg.iCANInd=0;//CAN 路数, 0 表示第 0 路 CAN, 1 表示第 1 路 CAN
cfg.wCANBaud=0x001c;//500K 波特率
cfg.iMasterCycle=500;//主站循环周期
cfg.wMasterID=0;//主站 ID

Mgr_AddRoute(cfg, &hRoute); //添加 iCAN 网络
```

3.2 如何添加从站

```
HANDLE hSlave;
Route_AddSlave(hRoute, 30, &hSlave); //添加从站, ID 为 30
```

3.3 如何启动系统

```
if(Mgr_StartSys() != ICANOK)
{
    MessageBox("启动失败");
}
```

3.4 如何连接从站并向其发送数据和接收数据

```
//假设从站的 D0 长度为 10 字节 (所占用资源 ID 范围为 0x20 到 0x29), A0 长度为 20 字节 (所占用
//资源 ID 范围为 0x60 到 0x73), DI 为 5 字节, AI 为 10 字节
BYTE buf[32]={0};
int len;
Slave_Connect(hSlave);
Slave_SendData(hSlave, 0x20, buf, 10); //往从站发送 10 字节 D0 数据
Slave_SendData(hSlave, 0x60, buf, 20); //往从站发送 20 字节 A0 数据
len=5; //DI 长度为 5 字节
Slave_GetDIData(hSlave, buf, &len); //len 中存储返回的实际接收到字节数
len=10; //AI 长度为 10 字节
Slave_GetAIData(hSlave, buf, &len); //len 中存储返回的实际接收到字节数
```

或

```
//假设从站的 D0 长度为 10 字节 (所占用资源 ID 范围为 0x20 到 0x29), A0 长度为 20 字节 (所占用
//资源 ID 范围为 0x60 到 0x73), DI 为 5 字节, AI 为 10 字节
BYTE buf[32]={0};
int len;
```

```
Slave_Connect(hSlave);
Slave_SendData(hSlave, 0x20, buf, 10); //往从站发送 10 字节 D0 数据
Slave_SendData(hSlave, 0x60, buf, 20); //往从站发送 20 字节 A0 数据
len=5; //DI 长度为 5 字节
Slave_GetData(hSlave, 0x00, buf, &len); //SourceID 设为 0x00, 表示接收 DI 数据, len 中存储返回
//的实际接收到字节数

len=10; //AI 长度为 10 字节
Slave_GetData(hSlave, 0x40, buf, &len); //SourceID 设为 0x40, 表示接收 AI 数据, len 中存储返回
//的实际接收到字节数
```

3.5 如何判断从站是否已经连接

```
if (Slave_IsConnected(hSlave) != 1)
{
    MessageBox("从站已经断开连接");
}
```

3.6 详细使用例程

在“例子”目录中有在 VC、VB、CB 和 Delphi 下如何使用此动态库的完整例程，请自行查看。

第 4 章...产品服务

4.1 软件升级

iCAN 的驱动软件终生免费升级。

4.2 技术支持

iCAN 的技术支持邮箱:

app@zlg.cn

技术支持专业主页:

<http://www.zlg.cn>

技术讨论园地:

<http://www.21icbbs.com/club/bbs/main.asp?boardid=43>