LIKE CAN 计算机接口卡
通用 DLL 函数库应用手册 V2.2

南京来可电子科技有限公司

电话：025-83197120

传真：025-83197121

网址：http://www.njlike.com

地　址：南京市江宁区高湖路 9 号金聚龙大厦 6 楼西区

目录

1. 接口函数库说明和使用 3
1.1 数据结构定义 3
1.1.1 CAN_DeviceType 3
1.1.2 CAN_ErrorCode 3
1.1.3 CAN_DataFrame 4
1.1.4 CAN_InitConfig 4
1.1.5 CAN_DeviceInformation 5
1.1.6 CAN_ErrorInformation 5
1.1.7 CAN 寄存器定义 5
1.2 接口函数详细说明 8
1.2.1 CAN_DeviceOpen 8
1.2.2 CAN_DeviceClose 8
1.2.3 CAN_ChannelStart 8
1.2.4 CAN_ChannelStop 9
1.2.5 CAN_ChannelSend 9
1.2.6 CAN_ChannelReceive 9
1.2.7 CAN_GetReceiveCount 10
1.2.8 CAN_ClearReceiveBuffer 10
1.2.9 CAN_GetDeviceInfo 10
1.2.10 CAN_GetErrorInfo 11
1.2.11 CAN_ WriteRegister 11
1.2.12 CAN_ ReadRegister 11
1.3 接口函数库使用详解 13
1.3.1 接口库函数调用流程 13
1.3.2 VC 调用接口库方法 13
1.3.3 VB 调用接口库方法 13

1. 接口函数库说明和使用

1.1 数据结构定义

1.1.1 CAN_DeviceType

描述：接口卡类型定义。

```
//接口卡类型定义
enum CAN_DeviceType
{
    ACUSB_131B   =   1,        //   ACUSB_131B 接口卡设备
    ACUSB_132B   =   2,        //   ACUSB_132B 接口卡设备
    ACPCI_251    =   3         //   单通道 PCI CAN 接口卡
    ACPCI_252    =   4         //   双通道  PCI CAN  接口卡
    ACPCI_254    =   5         //   4 通道  PCI CAN  接口卡
    ACNET_600    =   6         //   1 通道  以太网  CAN  接口卡
    ACNET_622    =   7         //   2 通道  以太网 CAN  接口卡
    LCPCIE_251   =   8         //   1 通道  PCIe CAN  接口卡
    LCPCIE_252   =   9         //   2 通道  PCIe CAN  接口卡
};
```

1.1.2 CAN_ErrorCode

描述：错误码类别定义。

```
//CAN错误码
enum CAN_ErrorCode
{
    CAN_E_NOERROR              = 0x0000,    // 没有发现错误
    CAN_E_OVERFLOW             = 0x0001,    // CAN 控制器内部 FIFO 溢出
    CAN_E_ERRORALARM           = 0x0002,    // CAN 控制器错误报警
    CAN_E_PASSIVE              = 0x0004,    // CAN 控制器消极错误
    CAN_E_LOSE                 = 0x0008,    // CAN 控制器仲裁丢失
    CAN_E_BUSERROR             = 0x0010,    // CAN 控制器总线错误

    CAN_E_DEVICEOPENED         = 0x0100,    // 设备已经打开
    CAN_E_DEVICEOPEN           = 0x0200,    // 打开设备错误
    CAN_E_DEVICENOTOPEN        = 0x0400,    // 设备没有打开
    CAN_E_BUFFEROVERFLOW       = 0x0800,    // 缓冲区溢出
    CAN_E_DEVICENOTEXIST       = 0x1000,    // 此设备不存在
```

```
    CAN_E_LOADKERNELDLL         = 0x2000,    // 装载动态库失败
    CAN_E_CMDFAILED             = 0x4000,    // 执行命令失败错误码
    CAN_E_BUFFERCREATE          = 0x8000,    // 内存不足
};
```

1.1.3 CAN_DataFrame

描述：CAN 数据帧结构体，在发送和接收时存放数据帧的结构体定义。

```
//CAN 数据帧类型
typedef  struct  tagCAN_DataFrame{
    UINTuTimeFlag;                  // 时间标识,对接收帧有效，从 CAN 通道启动开始计时
    BYTE     nSendType;             // 发送帧类型,0-正常发送;1-单次发送;2-自发自收;3-单次自发自收
    BYTE     bRemoteFlag;           // 是否是远程帧，0 表示数据帧，1 表示远程帧
    BYTE     bExternFlag;           // 是否是扩展帧，0 表示标准帧，1 表示扩展帧
    BYTE     nDataLen;              // 数据长度(<=8)，也就是 arryData 的长度
    UINT     uID;                   // 报文 DI
    BYTE     arryData[8];           // 报文数据
}CAN_DataFrame,*PCAN_DataFrame;
```

1.1.4 CAN_InitConfig

描述：表示初始化 CAN 通道的配置，通过 CAN_ChannelStart 函数在启动 CAN 通道时配置。

```
//CAN 初始化配置
typedef struct tagCAN_InitConfig{
    UCHAR   bMode;                  //模式，0 表示正常模式，1 表示只听模式
    BYTE    nBtrType;               //位定时参数模式，0 表示 LPC21XX ，1 表示 SJA1000
    BYTE    dwBtr[4];               //位定时参数, dwBtr[0]表示 BTR0，dwBtr[1]表示 BTR1
    DWORD   dwAccCode;              // 验收码
    DWORD   dwAccMask;              // 屏蔽码
    BYTE    nFilter;                // 滤波方式(0 表示未设置滤波,1 表示双滤波,2 表示单滤波)
    BYTE    dwReserved;             // 预留字段
}CAN_InitConfig,*PCAN_InitConfig;
```

备注：nBtrType 默认为 1 表示 SJA1000，通常通过 dwBtr 两个低字节设置通道的波特率如下：

CAN 通道波特率	dwBtr[0]	dwBtr[1]
5Kbps	0xBF	0xFF
10Kbps	0x31	0x1C
20Kbps	0x18	0x1C
50Kbps	0x09	0x1C

100Kbps	0x04	0x1C
125Kbps	0x03	0x1C
250Kbps	0x01	0x1C
500Kbps	0x00	0x1C
800Kbps	0x00	0x16
1000Kbps	0x00	0x14

1.1.5 CAN_DeviceInformation

描述：表示接口卡设备的基本信息，通过 CAN_GetDeviceInfo 函数获取此设备信息。

```
//CAN 设备信息
typedef  struct  tagCAN_DeviceInformation{
    USHORT  uHardWareVersion;          // 硬件版本，用十六进制表示，比如 0x0100 表示 V1.00
    USHORT  uFirmWareVersion;          // 固件版本，用十六进制表示
    USHORT  uDriverVersion;            // 驱动版本，用十六进制表示
    USHORT  uInterfaceVersion;         // 接口库版本，用十六进制表示
    USHORT  uInterruptNumber;          // 中断号
    BYTE    bChannelNumber;            // 设备总共有几路 CAN
    CHAR    szSerialNumber[20];        // 设备序列号
    CHAR    szHardWareType[40];        // 硬件类型
    CHAR    szDescription[20];         // 设备描述
} CAN_DeviceInformation,*PCAN_DeviceInformation;
```

1.1.6 CAN_ErrorInformation

描述：表示运行过程中产生的错误信息，通过 CAN_GetErrorInfo 函数获取此错误信息。

```
//CAN 错误信息
typedef struct tagCAN_ErrorInformation{
    UINT        uErrorCode;          // 错误码，详情请参见 CAN_ErrorCode 定义
    BYTE        PassiveErrData[3];   // 当错误码为消极错误时，表示消极错误数据
    BYTE        ArLostErrData;       // 当错误码为仲裁错误时，表示仲裁错误数据
} CAN_ErrorInformation,*PCAN_ErrorInformation;
```

1.1.7 CAN 寄存器定义

描述：表示 CAN 控制器寄存器地址，在调用 CAN_ReadRegister 和 CAN_WriteRegister 时使用，寄存器数据含义请参考 LPC2366 数据手册。

表 1.1 CAN 寄存器定义

寄存器名称	寄存器地址	描述
MOD	0x000	控制 CAN 控制器的工作模式。
CMR	0x004	影响 CAN 控制器状态的命令位
GSR	0x008	全局控制器状态和错误计数器
ICR	0x00C	中断状态、仲裁丢失捕获、错误代码捕获 （保存最后一次产生错误时的内容，读后清零）
IER	0x010	中断使能
BTR	0x014	总线时序
EWL	0x018	错误报警界限
SR	0x01C	状态寄存器
RFS	0x020	接收帧状态
RID	0x024	接收到的标识符
RDA	0x028	接收到的数据字节 1-4
RDB	0x02C	接收到的数据字节 5-8
TFI1	0x030	发送帧信息（Tx 缓冲器 1）
TID1	0x034	发送标识符（Tx 缓冲器 1）
TDA1	0x038	发送数据字节 1-4（Tx 缓冲器 1）
TDB1	0x03C	发送数据字节 5-8（Tx 缓冲器 1）
TFI2	0x040	发送帧信息（Tx 缓冲器 2）
TID2	0x044	发送标识符（Tx 缓冲器 2）
TDA2	0x048	发送数据字节 1-4（Tx 缓冲器 2）
TDB2	0x04C	发送数据字节 5-8（Tx 缓冲器 2）
TFI3	0x050	发送帧信息（Tx 缓冲器 3）
TID3	0x054	发送标识符（Tx 缓冲器 3）
TDA3	0x058	发送数据字节 1-4（Tx 缓冲器 3）
TDB3	0x05C	发送数据字节 5-8（Tx 缓冲器 3）
WD I/O 控制寄存器		

IN	0x080	输入端口值，bit0～4 对应于输入端口 1～5 （输入端口悬空时为 0；与输入地短接时为 1）
OUT	0x090	输出端口值，bit0~4 对应于输出端口 1～5
DEF	0x0a0	输出端口默认输出值，bit0~4 对应于输出端口 1～5 （在设备上电时，输出端口输出该值）

1.2 接口函数详细说明

1.2.1 CAN_DeviceOpen

功能描述：此函数用于打开设备。

```
DWORD __stdcall  CAN_DeviceOpen(DWORD dwType,  DWORD dwIndex, char*pDescription);
```

参数：

dwType： 设备类型，请根据具体设备从 CAN_DeviceType 枚举定义中选择具体类型；

dwIndex： 设备端口号，从 0 开始，比如有两个 USB 设备，则对应的端口号分别为 0 和 1；

pDescription：设备描述，用户描述设备信息，可填为空；

返回值：

返回 0 表示打开失败，否则表示返回具体的设备句柄，以后全部通过此句柄操作设备。

1.2.2 CAN_DeviceClose

功能描述：此函数用于关闭设备。

```
DWORD __stdcall CAN_DeviceClose(DWORD dwDeviceHandle);
```

参数：

dwDeviceHandle：设备的句柄；

返回值：

参见宏定义，为 CAN_RESULT_OK 表示关闭成功，为 CAN_RESULT_ERROR 表示关闭失败。

1.2.3 CAN_ChannelStart

功能描述：此函数用于初始化并启动指定的 CAN 通道。

```
DWORD __stdcall CAN_ChannelStart(DWORD dwDeviceHandle, DWORD dwChannel, PCAN_InitConfig pInitConfig);
```

参数：

dwDeviceHandle：设备的句柄；

dwChannel： 设备通道号，索引从 0开始，比如有两个通道，则对应的通道号分别为 0和 1；

pInitConfig： 初始化参数结构，请参见下表详解；

变量名称	功能描述
bMode	模式，0 表示正常模式，1 表示只听模式
nBtrType	位定时参数模式，0 表示 LPC21XX ，1 表示 SJA1000，默认值为 1
dwBtr	位定时参数, dwBtr[0]表示 BTR0，dwBtr[1]表示 BTR1

dwAccCode	验收码，对应 SJA1000 中的四个寄存器 ACR0，ACR1，ACR2，ACR3，其中高字节对应 ACR0，低字节对应 ACR3
dwAccMask	屏蔽码，对应 SJA1000 中的四个寄存器 AMR0，AMR1，AMR2，AMR3，其中高字节对应 AMR0，低字节对应 AMR3
nFilter	滤波方式：0 表示未设置滤波功能，1 表示双滤波，2 表示单滤波
dwReserved	预留字段，当前没有使用

返回值：

请参见值宏定义，CAN_RESULT_OK 表示启动成功，CAN_RESULT_ERROR 表示启动失败。

1.2.4 CAN_ChannelStop

功能描述：此函数用于停止指定的 CAN 通道。

```
DWORD __stdcall CAN_ChannelStop(DWORD dwDeviceHandle, DWORD dwChannel);
```

参数：

dwDeviceHandle：设备的句柄；

dwChannel： 设备通道号，索引从 0开始，比如有两个通道，则对应的通道号分别为 0和 1；

返回值：

请参见值宏定义，CAN_RESULT_OK 表示停止成功，CAN_RESULT_ERROR 表示停止失败。

1.2.5 CAN_ChannelSend

功能描述：此函数用于从指定 CAN 通道发送数据。

```
ULONG __stdcall CAN_ChannelSend(DWORD dwDeviceHandle, DWORD dwChannel, PCAN_DataFrame pSend, ULONG nCount);
```

参数：

dwDeviceHandle：设备的句柄；

dwChannel： 设备通道号，索引从 0开始，比如有两个通道，则对应的通道号分别为 0和 1；

pSend： 预发送的数据帧数组首指针；

nCount： 预发送的数据帧数组的长度；

返回值：

实际发送成功的数据帧帧数。

1.2.6 CAN_ChannelReceive

功能描述：此函数用于从指定 CAN 通道读取帧数据。

```
ULONG __stdcall CAN_ChannelReceive(DWORD dwDeviceHandle, DWORD dwChannel, PCAN_DataFrame pReceive, ULONG nCount, INT nWaitTime=-1);
```

参数：

dwDeviceHandle：设备的句柄；

dwChannel： 设备通道号，索引从 0开始，比如有两个通道，则对应的通道号分别为 0和 1；

pReceive： 用来接收数据帧数组缓冲区的首指针；

nCount： 用来接收数据帧缓冲区的数组长度；

nWaitTime: 等待超时时间，以毫秒为单位；

返回值：

返回实际读到的帧数，如果返回值为 0x0，则表示读取数据失败，有错误发生，请调用 CAN_GetErrorInfo 函数来获取错误码。

1.2.7 CAN_GetReceiveCount

功能描述：此函数用以获取底层接口库缓冲区中接收到但尚未被读取的帧数。

```
ULONG __stdcall CAN_GetReceiveCount(DWORD dwDeviceHandle, DWORD dwChannel);
```

参数：

dwDeviceHandle：设备的句柄；

dwChannel： 设备通道号，索引从 0开始，比如有两个通道，则对应的通道号分别为 0和 1；

返回值：

返回尚未被读取的缓冲区帧数。

1.2.8 CAN_ClearReceiveBuffer

功能描述：此函数用以清空获取底层接口库缓冲区数据。

```
DWORD __stdcall CAN_ClearReceiveBuffer(DWORD dwDeviceHandle, DWORD dwChannel);
```

参数：

dwDeviceHandle：设备的句柄；

dwChannel： 设备通道号，索引从 0开始，比如有两个通道，则对应的通道号分别为 0和 1；

返回值：

请参见值宏定义，CAN_RESULT_OK 表示操作成功，CAN_RESULT_ERROR 表示操作失败。

1.2.9 CAN_GetDeviceInfo

功能描述：此函数用于获取当前设备信息。

```
DWORD __stdcall CAN_GetDeviceInfo(DWORD dwDeviceHandle, PCAN_DeviceInformation pInfo);
```

参数：

dwDeviceHandle：设备的句柄；

pInfo：用来存储设备信息的 CAN_DeviceInformation 结构体指针；

返回值：

请参见值宏定义，CAN_RESULT_OK 表示获取成功，CAN_RESULT_ERROR 表示获取失败。

1.2.10 CAN_GetErrorInfo

功能描述：此函数用于获取最后一次错误信息。

```
DWORD __stdcall CAN_GetErrorInfo(DWORD dwDeviceHandle,  DWORD dwChannel,
PCAN_ErrorInformation pErr);
```

参数：

dwDeviceHandle：设备的句柄；

dwChannel： 设备通道号，索引从 0开始，比如有两个通道，则对应的通道号分别为 0和 1；

pErr：用来存储错误信息的 PCAN_ErrorInformation 结构体指针；

返回值：

请参见值宏定义，CAN_RESULT_OK 表示获取成功，CAN_RESULT_ERROR 表示获取失败。

1.2.11 CAN_ WriteRegister

功能描述：此函数向 CAN 控制指定寄存器中写入数据。

```
DWORD __stdcall CAN_WriteRegister(DWORD dwDeviceHandle, DWORD dwChannel, DWORD dwAddr,
PBYTE pBuff, WORD nLen);
```

参数：

dwDeviceHandle：设备的句柄；

dwChannel：指定的 CAN 通道号；

dwAddr：指定 CAN 寄存器的地址，见 1.1.7 CAN 寄存器定义；

pBuff :写入 CAN 寄存器的数据值，小端结构，如 pBuff[0]～[3]的值依次为 0x78 0x56 0x34 0x12，则写入到寄存器的内容为 0x12345678；

nLen：指定写入的数据长度，默认为 4 个字节；

返回值：

请参见值宏定义，CAN_RESULT_OK 表示写成功，CAN_RESULT_ERROR 表示写失败。

1.2.12 CAN_ ReadRegister

功能描述：此函数用于读取 CAN 控制器指定寄存器的数值。

```
DWORD __stdcall CAN_ReadRegister(DWORD dwDeviceHandle, DWORD dwChannel, DWORD dwAddr,
PBYTE pBuff, WORD nLen);
```

参数：

dwDeviceHandle：设备的句柄；

dwChannel：指定的 CAN 通道号；

dwAddr：指定 CAN 寄存器的地址，见 1.1.7 CAN 寄存器定义；

pBuff：存放读取所得的寄存器值的缓冲区，小端结构，如读到的寄存器内容为 0x12345678，则 pBuff[0]～[3]的值依次为 0x78 0x56 0x34 0x12；

nLen：指定读取的数据长度，默认为 4 个字节；

返回值：

请参见值宏定义，CAN_RESULT_OK 表示获取成功，CAN_RESULT_ERROR 表示获取失败。

1.3 接口函数库使用详解

1.3.1 接口库函数调用流程

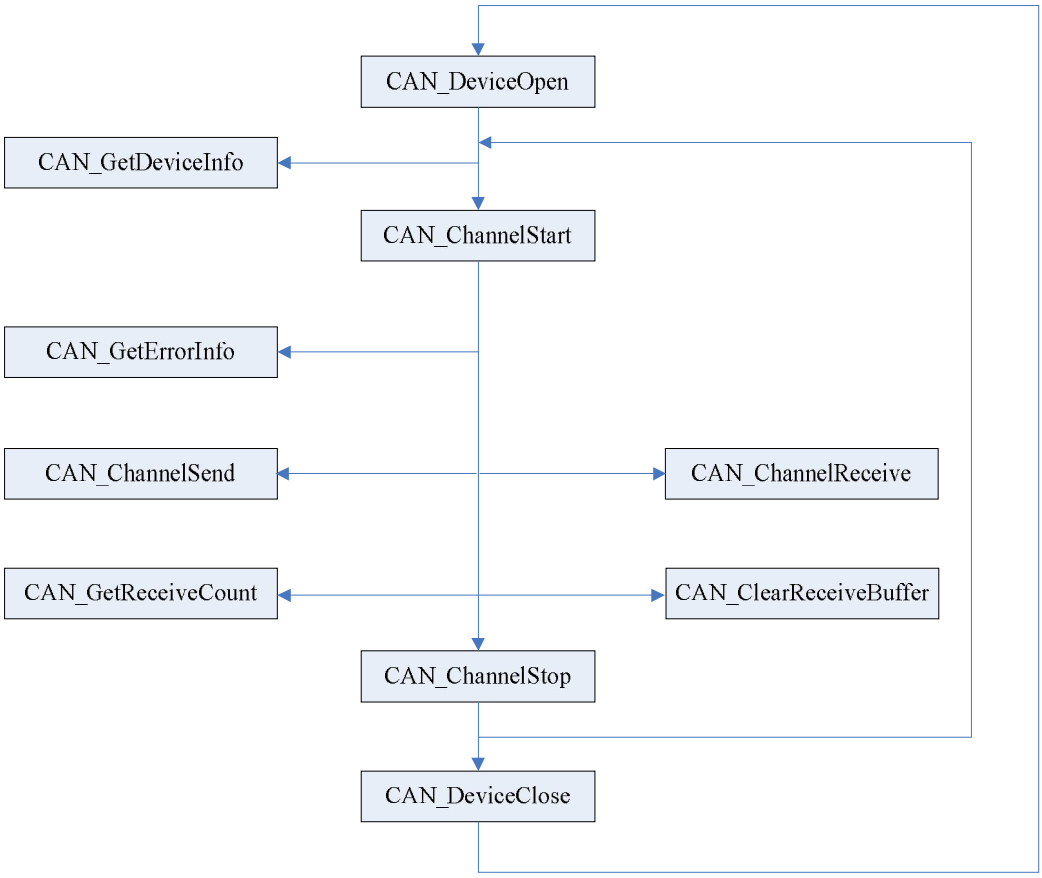

图 接口库函数调用流程

1.3.2 VC 调用接口库方法

1. 请将“ lib\ ”接口库文件夹下的接口文件：“ ICANCmd.h ”和“ CanCmd.lib ”，拷贝到您的工程目录下。
2. 请将接口库 lib 文件夹下两个接口库 DLL 文件：“ ACCAN.dll ”和“ CanCmd.dll ”，拷贝到可执行文件 exe 所在目录，比如“ Release ”或“ Debug ”。
3. 在工程中添加包含引用接口文件，比如在.CPP 中包含接口文件，参见下面代码：

```
#include "ICANCmd.h"
#pragma comment(lib, "CanCmd.lib")
```

4. 请参考以上接口函数调用说明和“ VC6 例子\Demo\ ”目录下的例子代码，根据您的实际应用进行开发。

1.3.3 VB 调用接口库方法

1. 请根据“ lib\ ”接口库文件夹下的接口文件 ICANCmd.h，实现对应 VB 的接口函数和相关数据结构的声明。
2. 请将接口库 lib 文件夹下两个接口库 DLL 文件：“ ACCAN.dll ”和“ CanCmd.dll ”，拷贝到可执行文件 exe 所在目录。
3. VB 的接口函数和相关数据结构的声明，可以参考“ VB6 例子\Demo\ ”目录下 Module1.bas 文件中的声明。
4. 请参考以上接口函数调用说明和“ VB6 例子\Demo\ ”目录下的例子代码，根据您的实际应用进行开发。