

# USBCAN-8E-U

## USBCAN-8E-U Linux 驱动使用说明

UM01010101 V1.00 Date: 2017/11/01

产品用户手册

类别	内容
关键词	USBCAN-8E-U
摘要	Linux 驱动使用说明

修订历史

版本	日期	原因
V1.00	2017/11/01	创建文档

## 目 录

1. 安装驱动.....	1
2. 驱动开发库接口函数说明.....	2
3. 免责声明.....	9

## 1. 安装驱动

USBCAN-8E-U 设备提供了 i386、x86\_64、armhf 三个版本的驱动程序，请根据自己的系统下载合适的版本的安装包，下面以 x86\_64 为例说明驱动程序的安装方法：

下载到的 x86\_64 版本的驱动包名 `usbcan-8e_x86_64_yyyymmdd_installer` (yyyymmdd 代表发布日期)，使用一下命令进行安装：

```
./usbcan-8e_x86_64_yyyymmdd_installer
```

注：该命令将驱动文件安装至 `/usr/lib/` 目录，执行该程序需要 root 权限，请根据系统的实际情况进行提权。如果本程序的执行权限丢失请先执行 `chmod` 操作。

安装成功系统将提示 `Install OK`。

如果使用 `armhf` 版本驱动程序，并且使用交叉编译环境，可以在执行 `usbcan-8e_armhf_yyyymmdd_installer` 时加路径参数指定安装位置，并在交叉编译过程中使用该路径作为编译链接参数。

如指定了安装位置为 `/opt/usbcan-8e/`，安装驱动文件时指定：

```
./usbcan-8e_armhf_yyyymmdd_installer /opt/usbcan-8e/
```

安装程序将所有的库和开发头文件安装到 `/opt/usbcan-8e/` 目录下，交叉编译时通过指定 `-I` 和 `-L` 参数来标示该安装位置的 `include` 和 `lib` 目录。

本驱动程序为通用的 Linux 库，没有 Linux 发行版本及内核版本的限制，只需根据操作系统的位数选择即可。

## 2. 驱动开发库接口函数说明

USBCAN-8E-U 的开发库采用 zlgcan2.0 编程接口，使用该驱动库时需要包含 zlgcan.h 头文件：

```
#include <zlgcan/zlgcan.h>
```

链接程序时需要增加 -lusbcan-8e 及 -libusb-1.0 的链接标志。

### 1. 打开设备

```
DEVICE_HANDLE ZCAN_OpenDevice(UINT device_type, UINT device_index, UINT reserved);
```

函数说明：

该函数打开一个设备，并返回用于操作该设备的句柄，调用时请判断返回值是否为合法的句柄。

参数说明：

**device\_type**：要打开的设备类型，USBCAN-8E-U 对应值为 ZCAN\_USBCAN\_8E\_U；

**device\_index**：要打开的设备索引号，0 表示第一个设备，设备索引号与设备接入电脑的时间和接入的 USB 口相关，如果电脑只接入一个 USBCAN-8E-U 则使用 0；

**reserved**：无用途；

返回值：

成功：返回非 INVALID\_DEVICE\_HANDLE 的句柄，该句柄代表着打开的设备，后续对该设备的操作需要使用该句柄；

失败：返回 INVALID\_DEVICE\_HANDLE

注：本函数操作需要 root 权限。

### 2. 关闭设备

```
INT ZCAN_CloseDevice(DEVICE_HANDLE device_handle);
```

函数说明：

该函数用于关闭不再使用的设备。

参数说明：

**device\_handle**：需要关闭设备的句柄；

返回值：

成功：返回 0；

失败：返回-1；

### 3. 获取设备信息

```
INT ZCAN_GetDeviceInf(DEVICE_HANDLE device_handle, ZCAN_DEVICE_INFO* pInfo);
```

函数说明：

该函数用于获取设备的基本信息。

参数说明：

**device\_handle**：需要操作设备的句柄；

**pInfo**：存放设备信息的指针，需要传递类型为 ZCAN\_DEVICE\_INFO 的非 NULL 指针；

返回值：

成功：返回 0；

失败：返回-1；

ZCAN\_DEVICE\_INFO 结构体定义如下：

```
typedef struct tagZCAN_DEVICE_INFO {
    USHORT hw_Version;
    USHORT fw_Version;
    USHORT dr_Version;
    USHORT in_Version;
    USHORT irq_Num;
    BYTE   can_Num;
    UCHAR  str_Serial_Num[20];
    UCHAR  str_hw_Type[40];
    USHORT reserved[4];
}ZCAN_DEVICE_INFO
```

对应 USBCAN-8E-U 设备，只有 hw\_Version、fw\_Version、can\_Num、str\_Serial\_Num 着四个域有意义，分别代表着设备的硬件版本、固件版本、可操作的 CAN 数、产品序列号。其他区域读取值为 0。

#### 4. 初始化 CAN 通道

```
CHANNEL_HANDLE ZCAN_InitCAN(DEVICE_HANDLE device_handle, UINT can_index,
ZCAN_CHANNEL_INIT_CONFIG* pInitConfig);
```

函数说明：

该函数用于初始化指定的 CAN 通道。

参数说明：

device\_handle: 需要操作设备的句柄；

can\_index: 需要初始化的 CAN 通道号，0 表示第一个 CAN 通道，USBCAN-8E-U 取值区间为 0~7；

InitConfig: 表示初始化配置信息指针，该指针类型为 ZCAN\_CHANNEL\_INIT\_CONFIG。

返回值：

成功：返回非 INVALID\_CHANNEL\_HANDLE 的句柄，该句柄代表着初始化后的通道句柄，后续对该通道的操作需要使用该句柄；

失败：返回 INVALID\_CHANNEL\_HANDLE

ZCAN\_CHANNEL\_INIT\_CONFIG 结构体定义如下：

```
typedef struct tagZCAN_CHANNEL_INIT_CONFIG {
    UINT can_type; // 0:can 1:canfd, 对应 USBCAN-8E-U, 只值恒为 0
    union
    {
        struct
        {
            UINT acc_code;
            UINT acc_mask;
            UINT reserved;
        }
    }
}
```

```

        BYTE filter;
        BYTE timing0;
        BYTE timing1;
        BYTE mode;
    }can;
    struct
    {
        UINT acc_code;
        UINT acc_mask;
        UINT timing0;
        UINT timing1;
        UINT brp;
        BYTE filter;
        BYTE mode;
        USHORT pad;
        UINT reserved;
    }canfd;
};
}ZCAN_CHANNEL_INIT_CONFIG;

```

USBCAN-8E-U 需要关注该结构体中联合体的 can 中的成员，这些成员及含义如下：

- acc\_code: 验收滤波码
- acc\_mask: 验收屏蔽码
- filter: 0 表示双验收滤波，1 表示单验收滤波
- timing0: BTR0 的值
- timing1: BTR1 的值
- mode: 0 表示正常工作模式，1 表示只听模式。

关于验收滤波码、屏蔽码、单双滤波、BTR0、BTR1 的取值可以从 CANTest 软件中获取。

### 5. 获取属性接口

```

IProperty* GetIProperty(DEVICE_HANDLE device_handle);

```

函数说明：

该函数用于获取设备的属性接口，通过该接口可以为设备设置属性。

参数说明：

device\_handle: 需要操作设备的句柄；

返回值：

成功：返回设备的属性接口，通过该接口可进一步操作设备，接口类型为 IProperty；

失败：NULL；

IProperty 属性接口结构体定义为：

```

typedef struct tagIProperty
{
    SetValueFunc    SetValue;
    GetValueFunc    GetValue;
};

```

```
GetPropertysFunc GetPropertys;
}IProperty;
```

其中 USBCAN-8E-U 只有 SetValue 方法可用，其他方法为 NULL。SetValue 方法类型如下所示：

```
typedef int (*SetValueFunc)(const char* path, const char* value);
```

其中参数 path 指出设置的属性路径名，value 指出设置的属性值。方法成功返回 0，失败返回-1。

USBCAN-8E-U 支持的属性如表 1 所示：

表 1 USBCAN-8E-U 支持属性列表

可用属性名	可选属性值	说明
"info/channel/channel_x/redirect" x 代表通道号，取值 0~7	"y z" y: 表示转发的通道号，取值 0~7， 且 y 不等于 x z: 是否启用转发，0 表示取消转发，1 表示启用转发	设置转发

注：所有属性设置需要在启动 CAN 操作前完成才能生效。

## 6. 启动 CAN

```
INT ZCAN_StartCAN(CHANNEL_HANDLE channel_handle);
```

函数说明：

该函数用于启用 CAN 通道。

参数说明：

channel\_handle: 需要操作通道的句柄；

返回值：

成功：返回 0；

失败：返回-1；

## 7. 关闭 CAN

```
INT ZCAN_ResetCAN(CHANNEL_HANDLE channel_handle);
```

函数说明：

该函数用于关闭 CAN 通道。

参数说明：

channel\_handle: 需要操作通道的句柄；

返回值：

成功：返回 0；

失败：返回-1；

## 8. 清空发送缓存

```
INT ZCAN_ClearBuffer(CHANNEL_HANDLE channel_handle);
```

函数说明：

该函数用于清空指定 CAN 通道发送缓存。



参数说明:

channel\_handle: 需要操作通道的句柄;

返回值:

成功: 返回 0;

失败: 返回-1;

## 9. 发送 CAN 帧

```
INT ZCAN_Transmit(CHANNEL_HANDLE channel_handle, ZCAN_Transmit_Data* pTransmit, UINT len);
```

函数说明:

该函数用于发送 CAN 帧。

参数说明:

channel\_handle: 需要操作通道的句柄;

pTransmit: 需要发送帧列表, CAN 帧用 ZCAN\_Transmit\_Data 来表示。

len: 需要发送的帧数目;

返回值:

成功: 返回已发送帧数;

失败: 返回-1;

ZCAN\_Transmit\_Data 结构体定义为:

```
typedef struct tagZCAN_Transmit_Data
{
    can_frame frame;
    UINT transmit_type;
}ZCAN_Transmit_Data;
```

该结构体包含两个成员:

frame: CAN 帧结构, 类型为 can\_frame;

transmit\_type : 发送类型, 0 表示正常发送, 1 表示单次发送, 2 表示自发自收, 3 表示单次自发自收;

canframe 结构体定义如下:

```
#define CAN_MAX_DLEN 8

/* special address description flags for the CAN_ID */
#define CAN_EFF_FLAG 0x80000000U /* EFF/SFF is set in the MSB */
#define CAN_RTR_FLAG 0x40000000U /* remote transmission request */
#define CAN_ERR_FLAG 0x20000000U /* error message frame */
#define CAN_ID_FLAG 0x1FFFFFFFU /* id */

/* valid bits in CAN ID for frame formats */
#define CAN_SFF_MASK 0x000007FFU /* standard frame format (SFF) */
#define CAN_EFF_MASK 0x1FFFFFFFU /* extended frame format (EFF) */
#define CAN_ERR_MASK 0x1FFFFFFFU /* omit EFF, RTR, ERR flags */
```

```
#define CAN_ID(id, eff, rtr, err) (id | (!!eff) << 31 | (!!rtr) << 30 | (!!err) << 29)
#define IS_EFF(id) (!!id & CAN_EFF_FLAG) //1:extend frame 0:standard frame
#define IS_RTR(id) (!!id & CAN_RTR_FLAG) //1:remote frame 0:data frame
#define IS_ERR(id) (!!id & CAN_ERR_FLAG) //1:error frame 0:normal frame
struct can_frame {
    canid_t can_id; /* 29 bit CAN_ID + EFF/RTR/ERR flags */
    __u8 can_dlc; /* frame payload length in byte (0 .. CAN_MAX_DLEN) */
    __u8 __pad; /* padding */
    __u8 __res0; /* reserved / padding */
    __u8 __res1; /* reserved / padding */
    __u8 data[CAN_MAX_DLEN]/* __attribute__((aligned(8)))*/;
};
```

can\_frame 结构体包含多个变量，但目前 CAN 帧仅使用到 can\_id, can\_dlc 和 data 数组三个，这三个变量用途如下：

can\_id: 用于保存接收或发送帧的帧 ID 及远程帧、扩展帧、错误帧的标志位。其中 bit0~bit28 用于保存帧 ID，bit29 为错误帧标志，bit30 为远程帧标志，bit31 为扩展帧标志。当 CAN 帧为标准帧时，帧 ID 保存在 bit0~bit10，当 CAN 帧为扩展帧时，帧 ID 保存在 bit0~bit28。在 zlscan.h 头文件中已经定义了若干宏来帮助 can\_id 的处理。

can\_dlc: 用于保存帧数据的长度，值为 0~8。

data: 用于保存帧数据。

## 10. 接收 CAN 帧

```
INT ZCAN_Receive(CHANNEL_HANDLE channel_handle, ZCAN_Receive_Data* pReceive, UINT len, INT wait_time );
```

函数说明：

该函数用于接收 CAN 帧。

参数说明：

channel\_handle: 需要操作通道的句柄；

pReceive: 需要接收帧的指针，CAN 帧用 ZCAN\_Receive\_Data 来表示。

len: 本次接收最大帧数目；

wait\_time: >0 表示接收超时时间，单位为 ms。<0 表示一直等待。=0 立即返回。

返回值：

成功：返回接收到帧数；

失败：返回-1；

ZCAN\_Receive\_Data 结构体定义为：

```
typedef struct tagZCAN_Receive_Data
{
    can_frame frame;
    UINT64 timestamp;
}ZCAN_Receive_Data;
```

该结构体包含两个成员：

frame: CAN 帧结构, 类型为 can\_frame, 同发送函数。

timestamp : 接收时间, 单位为微秒。

### 11. 获取错误信息

```
INT ZCAN_ReadChannelErrInfo(CHANNEL_HANDLE channel_handle, ZCAN_CHANNEL_ERR_INFO* pErrInfo);
```

函数说明:

该函数用于读取一条错误信息。

参数说明:

channel\_handle: 需要操作通道的句柄;

pErrInfo: 需要错误信息的指针, 错误信息用 ZCAN\_CHANNEL\_ERR\_INFO 来表示。

返回值:

成功: 1 表示获取到 1 个错误帧, 0 表示无错误帧;

失败: 返回-1;

ZCAN\_CHANNEL\_ERR\_INFO 的结构体定义为:

```
typedef struct tagZCAN_CHANNEL_ERR_INFO {
    UINT error_code;
    BYTE passive_ErrData[3];
    BYTE arLost_ErrData;
} ZCAN_CHANNEL_ERR_INFO;
```

对应 USBCAN-8E-U 该结构只需关注 error\_code, error\_code 可以取值如表 2 所示:

表 2 帧状态对应的错误状态

状态字	帧状态的具体错误内容
0x1	CAN 控制器内部 FIFO 溢出
0x2	CAN 控制器错误报警
0x3	CAN 控制器消极错误
0x4	CAN 控制器仲裁丢失
0x5	CAN 控制器总线错误
0x6	
0x7	CAN 数据缓冲区溢出

### 3. 免责声明

## 销售与服务网络

### 广州致远电子有限公司

地址：广州市天河区车陂路黄洲工业区 7 栋 2 楼

邮编：510660

网址：[www.zlg.cn](http://www.zlg.cn)

全国销售与服务电话：400-888-4005



全国服务电话：400-888-4005

### 销售与服务网络：

#### 广州总公司

广州市天河区车陂路黄洲工业区 7 栋 2 楼

电话：020-28267893

#### 上海分公司

上海市北京东路 668 号科技京城东楼 12E 室

电话：021-53865720-801

#### 北京分公司

北京市丰台区马家堡路 180 号 蓝光云鼎 208 室

电话：010-62536178

#### 深圳分公司

深圳市福田区深南中路 2072 号电子大厦 12 楼 1203 室

电话：0755-82941683 0755-82907445

#### 武汉分公司

武汉市洪山区民族大道江南家园 1 栋 3 单元 602

电话：027-62436478 13006324181

#### 南京分公司

南京市秦淮区汉中路 27 号友谊广场 17 层 F、G 区

电话：025-68123919

#### 杭州分公司

杭州市西湖区紫荆花路 2 号杭州联合大厦 A 座 4 单元 508

电话：0571-86483297

#### 成都分公司

成都市一环路南 2 段 1 号数码科技大厦 319 室

电话：028-85439836-805

#### 郑州分公司

河南省郑州市中原区建设西路与百花路东南角锦绣华庭 A 座 1502

电话：400-888-4005 (0371)66868897

#### 重庆分公司

重庆市九龙坡区石桥铺科园一路二号大西洋国际大厦（百脑会）2705 室

电话：023-68797619

#### 西安办事处

西安市长安北路 54 号太平洋大厦 1201 室

电话：029-87881295

#### 天津办事处

天津市河东区津塘路与十一经路交口鼎泰大厦 1004

电话：022-24216606

#### 青岛办事处

山东省青岛市李沧区青山路 689 号宝龙公寓 3 号楼 701 室

电话：0532-58879795 17660216799

请您用以上方式联系我们，我们会为您安排样机现场演示，感谢您对我公司产品的关注！