北京爱泰联合科技有限公司

CANFD 函数库使用手册

[iTekCANFD.dll 开发接口说明]

V1.0.1

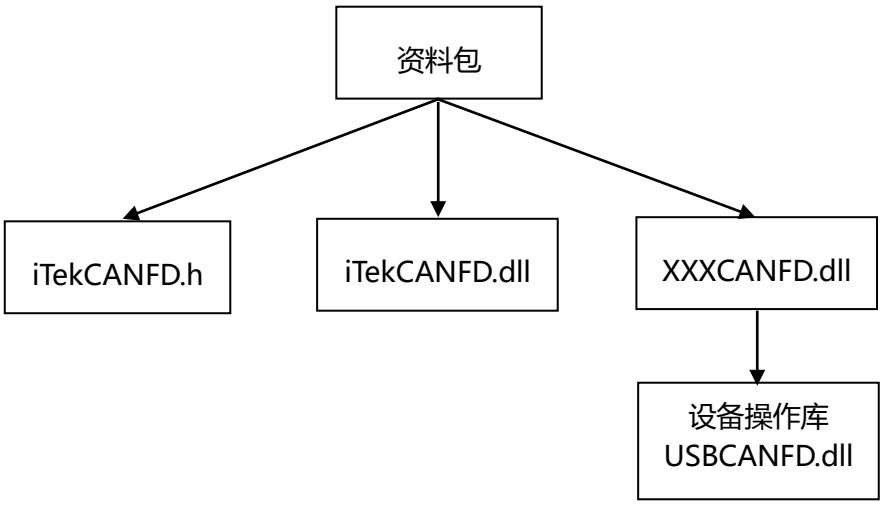
2021-1-25

目 录

1. 资料包介绍

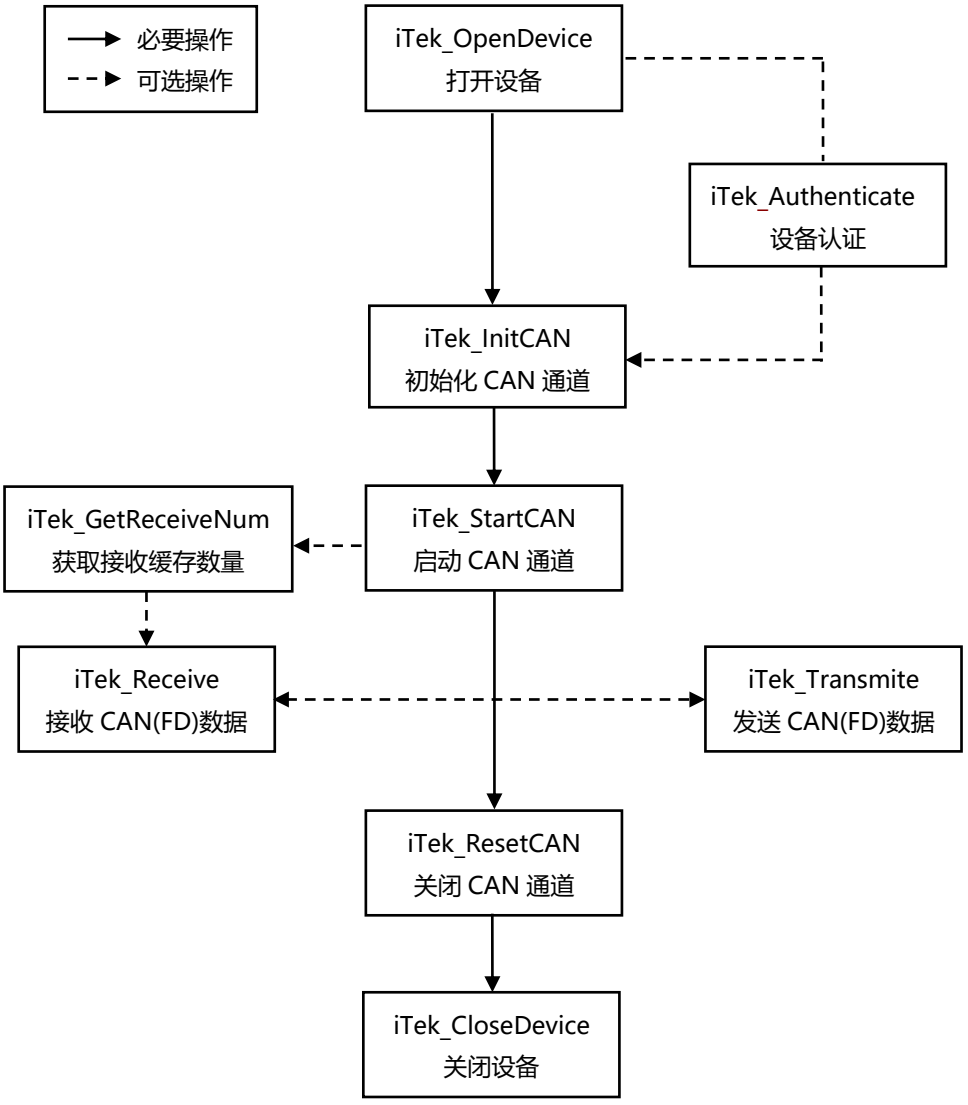
接口库以基于 window 系统的动态链接库（DLL）的方式提供，可以实现打开设备，配置 CAN(FD)参数，收发报文等功能。该函数库支持我司 CANFD 系列产品，如 USBCANFD、PCIeCANFD、NetCANFD、MiniPCIeCANFD 等产品。接口库采用 visual studio 2017 开发，依赖运行库 2017 版本。如计算机没有包含该运行库，可到微软官方网站下载安装。

应用程序根据窗口选择的设备类型加载不同产品的dll，iTekCANFD.h为接口描述头文件，iTekCANFD.dll和设备相关的XXXCANFD.dll需要放在可执行程序生成目录下。



Linux平台的函数接口和二次开发流程与Windows开发库相同，只是使用的开发库文件不同而已。

2. 开发流程图


必要操作
可选操作
iTek_OpenDevice
打开设备
iTek_Authenticate
设备认证
iTek_InitCAN
初始化 CAN 通道
iTek_StartCAN
启动 CAN 通道
iTek_GetReceiveNum
获取接收缓存数量
iTek_Receive
接收 CAN(FD)数据
iTek_Transmite
发送 CAN(FD)数据
iTek_ResetCAN
关闭 CAN 通道
iTek_CloseDevice
关闭设备

3. 数据结构定义

3.1. iTek_CANFD_DEVICE_INFO

设备信息结构体，包含硬件版本号、产品序列号、设备描述符等信息。

<iTek_CANFD_DEVICE_INFO 结构体>

```
typedef struct tagiTek_CAN_DEVICE_INFO {
    uint8_t   hw_Version[3];
    uint8_t   fw_Version[3];
    uint8_t   product_Version[3];
    uint8_t   can_Num;
    uint8_t   str_Serial_Num[20];
    uint8_t   str_hw_Type[40];
    uint16_t reserved[6];
}iTek_CANFD_DEVICE_INFO;
```

成员：

hw_Version	硬件版本，如 hw_Version={0x01, 0x00, 0x02}，代表 V1.0.2；
fw_Version	固件版本，如 fw_Version={0x01, 0x00, 0x03}，代表 V1.0.3；
product_Version	（暂不开放）；
can_Num	通道数量，如 can_Num=2，代表设备集成 2 路 CANFD 接口；
str_Serial_Num	出厂序列号，以‘\0’结束，如“6120101001”；
str_hw_Type	设备名称描述字符，以‘\0’结束，如“USBCANFD-X200”；
reserved	保留

3.2. iTek_CANFD_CHANNEL_INIT_CONFIG

定义 CAN 控制器初始化参数，在调用 iTek_initCan 之前要先初始化该结构体。

<iTek_CANFD_CHANNEL_INIT_CONFIG 结构体>

```
typedef struct tagiTek_CAN_CHANNEL_INIT_CONFIG {
    uint8_t     can_type;
    uint8_t     CANFDStandard;
    uint8_t     CANFDSpeedup;
    uint32_t    abit_timing;
    uint32_t    dbit_timing;
    uint8_t     workMode;
    uint32_t    res;
    tagFilterDataExtend Extend;
    tagFilterDataStandard Standard;
}iTek_CANFD_CHANNEL_INIT_CONFIG;
```

成员：

can_type	CAN 协议类型： 0=CAN 协议；1=CANFD 协议；
CANFDStandard	CANFD 标准：0=ISO 标准；1=非 ISO 标准；
CANFDSpeedup	CANFD 是否加速： 0=不加速； 1=加速；
abit_timing	仲裁波特率（见表 3-1）；
dbit_timing	数据波特率（见表 3-2）；
workMode	工作模式， 0=正常工作模式；1=只听工作模式；
res	保留位
Extend	扩展帧过滤器组（见 3.3 节）；
Standard	标准帧过滤器组（见 3.4 节）；

表 3-1 仲裁域波特率

波特率	abit_timing
5K	0x01F31302
10K	0x00F91302
20K	0x007C1302
40K	0x00630a02
50K	0x00311302
80K	0x001D1204
100K	0x00181302
125K	0x00131302
200K	0x00130A02
250K	0x00091302
400K	0x00090A02
500K	0x00070A02
800K	0x00021006
1000K	0x00040702

表 3-2 数据域波特率

波特率	dbit_timing
100K	0x001D0E30
125K	0x001F0A20
200K	0x00130A20
250K	0x000F0A20
400K	0x00090A20
500K	0x00070A20
800K	0x00040A20
1M	0x00040720
2M	0x00010A20
3M	0x00000D40
4M	0x00000A20
5M	0x00000720

3.3. FilterDataExtend

扩展帧过滤器组结构体，过滤器组数量根据实际使用情况自定义。

<FilterDataExtend 结构体>

```
typedef struct tagFilterDataExtend{
    int num;
    FilterDatafilterDataExtend[64];
}FilterDataExtend;
```

成员：

num	实际使用扩展帧过滤器组数量，取值范围 0-64，比如：num=2，代表前 2 组有效；
FilterData	扩展帧过滤器组寄存器结构体；

3.4. FilterDataStandard

标准帧过滤器组结构体，过滤器组数量根据实际使用情况自定义。

<FilterDataStandard 结构体>

```
typedef struct tagFilterDataStandard{
    int num;
    FilterDatafilterDataStandard[128];
}FilterDataStandard;
```

成员：

num　　实际使用标准帧过滤器组数量，取值范围 0-128，比如：num=5，代表前 5 组有效；

FilterData　　标准帧过滤器组寄存器结构体；

注意:CAN 控制器可分别对标准帧和扩展帧 ID 进行过滤,标准帧或扩展帧过滤器组应至少有一组有效,否则 CAN 控制器拒绝接收所有 id 的报文。

3.5. FilterData

过滤器组结构体，由帧类型、过滤方式和 ID 寄存器组成。

<FilterData 结构体>

```
typedef struct tagFilterData
{
    uint8_t     frameType;
    uint8_t     filterType;
    uint32_t    ID1;
    uint32_t    ID2;
}FilterData;
```

成员：

frameType　　帧类型，用于说明该组过滤器组是针对标准帧还是扩展帧 id 有效；0=标准帧有效；1=扩展帧有效；

filterType　　过滤器类型，用于说明本组过滤器组的过滤方式；0=范围 id；1=明确 id；2=掩码 id；不同方式的过滤结果见表 3-3；

ID1　　ID 寄存器 1；

ID2　　ID 寄存器 2；

表 3-3 过滤器组说明

	过滤器模式	过滤结果
1	范围ID	>= ID1 且 <= ID2的ID都可以通过。
2	明确ID	=ID1或 =ID2的ID可以通过。
3	掩码ID	当ID2的某一位为1时,使能该位过滤,只有这一位等于ID1的这一位才可以通过。 当ID2的某一位为0时，不启用该位过滤，该位为0或1都可以通过。 （标准帧，低11位有效；扩展帧，低29位有效）

3.6. iTek_CANFD_Receive_Data

接收 CAN(FD)报文信息结构体，包含 CAN 控制器硬件时间戳、CAN(FD)数据帧等。

<iTek_CANFD_Data 结构体>

```
typedef struct tagiTek_CANFD_Data{
    canfd_frame     frame;
    uint64_t        timestamp;
}iTek_CANFD_Receive_Data;
```

成员:

frame	CAN(FD)报文信息(见 canfd_frame 结构体定义)；
timestamp	CAN 时间戳，从 CAN 控制器上电开始计时，长度 64 位，单位 us（接收帧有效)；

3.7. iTek_CANFD_Transmit_Data

发送 CAN(FD)报文信息结构体，包含发送类型和 CAN(FD)帧结构。

<iTek_CANFD_Transmit_Data 结构体>

```
typedef struct tagiTek_CANFD_Transmit_Data{
    canfd_frame     frame;
    uint16_t        send_type;
}iTek_CANFD_Transmit_Data;
```

成员：

frame	CAN(FD)报文信息(见 canfd_frame 结构体定义)；
send_type	发送方式：0 = 正常发送；1 = 自发自收；

3.8. canfd_frame

CAN(FD)帧信息结构体，包含帧 id、帧格式、帧类型、帧数据等。

<canfd_frame 结构体>

```
typedef struct tag_canfd_frame {
    canid_t   can_id;
    uint8_t    len;
    uint8_t    flags;
    uint8_t    res;
    uint8_t    cantype;
    uint8_t    data[CANFD_MAX_DLEN];
}canfd_frame;
```

成员：

can_id	帧 ID，高 3 位属于标志位，低 29 位 ID 有效位，标志位含义见表 3-4；
len	数据长度，当前 CAN(FD)帧实际数据长度；
flags	错误帧标志位，0=正常数据帧；1=错误帧；当 flags=1 时，错误信息通过 CAN 帧数据位 data0-data7 表达，具体定义见表 3-5；

res　　保留位；
cantype　　CAN 类型，0 = CAN；2 = CANFD；3= CANFD 加速；
data　　数据，CAN 帧 data<=8，CANFD 帧<=64;

表 3-4 ID 位定义表

位	定义	说明
31	保留	
30	帧类型	0=标准帧；1=扩展帧；宏 IS_EFF 可获取该标志；
29	帧格式	0=数据帧；1=远程帧，宏 IS_RTR 可获取该标志；
28:0	帧 ID	帧 ID，29 位有效，宏 GET_ID 获取 ID；宏 MAKE_CAN_ID 构造 ID

表 3-5 错误帧信息定义

数据	位	定义	说明
Data0 - Data3	31:15	保留	
	14:8	REC	接收错误计数器，取值 0-127；
	7:0	TEC	发送错误计数器，取值 0-128；
Data4 - Data7	31:8	保留	
	7	BO	CAN(FD)离线标志，0=在线；1=离线；
	6	EW	错误预警标志，0=无预警；1=预警；
	5	EP	错误消极标志，0=错误活动；1=错误消极；
	4:3	保留	
	2:0	LEC	错误代码，0=没有错误；1=位填充错误；2=格式错误；3=应答错误；4=Bit1 错误；5=Bit0 错误；6=CRC 错误；7=无错误；

错误说明：

位填充错误：接收消息时，一部分序列中有超过 5 个连续相等的位；
格式错误：接收帧固定格式部分错误；
应答错误：未收到 CAN 节点接收确认信号；
Bit1 错误：消息传输期间，发送隐性位被总线显性位主导；
Bit0 错误：消息传输期间，发送显性位被总线隐性位主导；
CRC 错误：接收到的消息 CRC 校验错误。

4. 接口函数说明

4.1. iTek_OpenDevice

该函数用于打开指定设备类型的设备，获取该设备句柄。

```
DEVICE_HANDLE iTek_OpenDevice(uint32_t Device_Type, uint16_t device_index, uint32_t reserved);
```

参数：

Device_Type　　设备类型号，定义见表 4-1

device_index　　设备索引号，用于区分一台计算机上使用的多套同类型设备。如只插 1 台 USBCANFD 设备，device_index=0；

reserved　　仅作保留；

返回值：

为 INVALID_HANDLE_VALUE 表示操作失败，否则表示操作成功，返回设备句柄值。请保存该句柄值，后续初始化 CAN、认证设备、关闭设备等都是针对此句柄操作。

表 4-1 设备类型号定义

设备名称	设备类型号
USBCANFD-X100	1
USBCANFD-X200	2

4.2. iTek_InitCan

该函数用于初始化 CAN(FD)通道，将工作模式、波特率、过滤屏蔽码等参数写入 CAN(FD)控制器。

```
USBChannel_HandleiTek_InitCan(DEVICE_HANDLE usbhandle,uint8_t channel,iTek_CANFD_CHANNEL_INIT_CONFIG* config);
```

参数：

usbhandle　　设备句柄值，由 iTek_OpenDevice 函数返回；

Channel　　通道号 0=CAN0 通道；1=CAN1 通道；

Config　　CAN(FD) 通道初始化设置信息（详情参考 3.2 节）；

返回值：

CAN(FD)通道句柄，后续启动通道、发送数据、复位通道等都是针对此句柄操作。

4.3. iTek_StartCAN

启动 CAN(FD)通道，通道初始化后应调用该函数启动，之后才能进行数据收发。

```
bool iTek_StartCAN(CHANNEL_HANDLE channel_handle);
```

参数：

channel_handle　　CAN(FD)通道句柄值；

返回值：

TRUE=成功；FALSE=失败。

4.4. iTek_Receive

该函数用于读取缓存中 CAN(FD)数据。

```
uint32_t iTek_Receive(CHANNEL_HANDLE channel_handle, iTek_CANFD_Data* pReceive, uint32_t len, int wait_time)
```

参数：

Channel_handle　通道句柄值；

pReceive　用于接收数据结构体的首地址指针；

len　本次准备读取的报文数目；

wait_time　函数阻塞等待时间，单位毫秒。当读到的数据数目<len时，等待wait_time毫秒后返回。为-1则表示无超时，一直等待，直到读到数目=len时才返回。

返回值：

实际读取的报文数目。

4.5. iTek_Transmit

该函数用于发送数据

```
uint32_tiTek_Transmit (CHANNEL_HANDLEchannel_handle, iTek_CANFD_Transmit_Data* data, uint32_t len);
```

参数：

channel_handle　通道句柄值；

data　待发送数据结构体的首地址指针；

len　待发送的报文数目，如 len=5，表示一次传递 5 个 CAN(FD)报文。

返回值：

返回实际发送成功的报文数目。

4.6. iTek_ClearBuffer

该函数用于清除库缓存区

```
void __stdcalliTek_ClearBuffer(CHANNEL_HANDLE channel_handle)
```

参数：

channel_handle　通道句柄值；

返回值：

无。

4.7. iTek_GetReceiveNum

获取该通道缓存等待读取的报文数量。

```
uint32_tiTek_GetReceiveNum (CHANNEL_HANDLEchannel_handle);
```

参数：

channel_handle　通道句柄值；

返回值：

该通道等待读取的报文数量。

4.8. iTek_Authenticate

设备认证函数，为保护用户二次开发软件版权，建议使用该函数对设备进行认证。

```
void *iTek_Authenticate(DEVICE_HANDLE usbhandle, uint16_t key_type,uint8_t *pStr, int len, unsigned char *dStr);
```

参数：

usbhandle	设备句柄值；
key_type	密钥类型（注意：该值必须为 0，否则会认证失败）；
pStr	应用层生成的 16 位随机数首地址指针；
len	随机数长度，固定为 16；
dStr	对 16 位随机数加密运算后返回的密文首地址指针。

返回值：

无

认证过程说明：设备使用产品密钥作为 key 对应用层生成的 16 位随机数经过 SM4 算法加密生成 16 位密文，应用程序对 16 位密文使用产品密钥作为 key 进行 SM4 解密，解密结果和生成的 16 位随机数比对，通过比对判断设备保存的密钥和应用层保存的密钥是否相符。如果相符，认证通过，否者认证失败。产品出厂默认口令“8888888888888888”，默认产品密钥“8888888888888888”。

4.9. iTek_CloseDevice

关闭设备函数，在软件退出前应调用该函数以释放设备句柄，否则下次会打开设备失败。

```
void iTek_CloseDevice(DEVICE_HANDLE usbhandle)
```

参数：

usbhandle	需要关闭的设备句柄；

返回值：

无。

4.10. iTek_isConnected

设备连接状态检查函数，应用层可以定期调用该函数判断设备连接状态，该函数必须在 iTek_OpenDevice 函数之后执行。

```
bool iTek_isConnected(DEVICE_HANDLE usbhandle)
```

参数：

usbhandle	设备句柄值；

返回值：

TRUE=连接状态；FALSE=当前连接已断开。

4.11. iTek_GetDeviceInfo

获取该设备信息。

```
bool iTek_GetDeviceInfo(DEVICE_HANDEL usbhandle,iTek_CANFD_DEVICE_INFO* pinfo)
```

参数：

Usbhandle 设备句柄；

pInfo 返回的设备信息保存地址指针；

返回值：

TRUE=成功；FALSE =失败。

4.12. iTek_ResetCAN

复位 CAN(FD)通道。

```
bool iTek_RestCAN(CHANNEL_HANDLE channe_handle);
```

参数：

channel_handle 通道句柄值；

返回值：

TRUE=成功；FALSE=失败。