

# TSMasterAPI. Interop 编程指导 V0.8



## 文档修订历史:

文件版本	日期	更新内容	备注
<b>V1.00</b>	2023.5.8	创建文档	

## 1. 目录

1. 什么情况下需要此文档? .....	3
2. 添加库文件 .....	3
3.数据类型定义 .....	3
1. TLIBCAN: CAN 总线数据类型 .....	3
成员: .....	4
调用示例: .....	4
2. TLIBCANFD: CANFD 总线数据类型 .....	4
成员: .....	5
调用示例: .....	6
3. TLIBLIN: LIN 总线数据类型 .....	6
成员: .....	6
4.报文发送 .....	7
5.报文接收 .....	7
1. 回调函数方式: .....	7
简介: .....	7
注册回调函数: .....	7
回调函数使用 .....	8
2. 读取设备消息缓存的方式: .....	8
简介: .....	8
tsfifo_receive_can_msgs .....	8
tsfifo_canfd_receive_buffers .....	9
6.TSMasterAPI 调用流程 .....	10
7.UDS 诊断接口说明 .....	10
8.接口函数介绍 .....	11

# 1. 什么情况下需要此文档?

用户基于 python 的编程语言，对上海同星智能科技有限公司的 TSCAN 系列工具（TSCANMini, TSLiteMini, TSCANFDMini, TSCANLINLite, TSCANLINPro）进行二次开发的时候，需要参考本文档，调用 API 函数来实现对设备的程序控制。

# 2. 添加库文件

- 1.Pip install 需要额外安装的库为(python-pyqt5)
- 2.只需将 TSMaster.dll 与工程脚本放在同一目录下即可。

# 3.数据类型定义

## 1. TLIBCAN: CAN 总线数据类型

```
class TLIBCAN(Structure):
    _pack_ = 1
    _fields_ = [("FIdxChn", c_uint8),
                ("FProperties", c_uint8),
                ("FDLC", c_uint8),
                ("FReserved", c_uint8),
                ("FIdentifier", c_int32),
                ("FTimeUs", c_int64),
                ("FData", c_uint8 * 8),
                ]
    def __init__(self, FIdxChn = 0, FDLC = 8, FIdentifier = 0x1, FProperties = 1, FData=[]):
        self.FIdxChn = FIdxChn
        self.FDLC = FDLC
        if self.FDLC > 8:
            self.FDLC = 8
        self.FIdentifier = FIdentifier
        self.FProperties = FProperties
        for i in range(len(FData)):
            self.FData[i] = FData[i]
```

## 成员：

**FData:** 帧数据。最大长度为 8Bytes

**FDLC:** 帧长度。

**FIdentifier:** 帧 ID，如果为 0xFFFFFFFF，表示当前帧为错误帧

**FIdxChn:** 帧通道，注意 CHANNEL\_INDEX。CHN1 = 0, 实际上是从 0 开始计算的。

**FTimeUS:** 帧时间戳，64 位 us 级时间戳。

**FProperties:** 存储 CAN 相关的属性，比如是否远程帧，是否扩展帧。

其中，属性字节定义如下：

**【1】 FProperties:** CAN 属性定义：该参数默认为 0，共八个 bits，每一个位的定义如下：

Bit	意义
0	0: Rx 接收报文；1: Tx 发送报文
1	0: data frame 数据帧；1: remote frame 远程帧
2	0: std frame 标准帧；1: extended frame 扩展帧
3-5	Reserved
6	0: 不记录；1: 已经被记录
7	Reserved

## 调用示例：

### 2. TLIBCANFD: CANFD 总线数据类型

```

class TLIBCANFD(Structure):
    _pack_ = 1
    _fields_ = [("FIdxChn", c_uint8),
                ("FProperties", c_uint8),
                ("FDLC", c_uint8),
                ("FFDProperties", c_uint8),
                ("FIdentifier", c_int32),
                ("FTimeUs", c_ulonglong),
                ("FData", c_ubyte * 64),
                ]
    def __init__(self, FIdxChn = 0, FDLC = 8, FIdentifier = 0x1, FProperties =
1, FFDProperties = 1, FData=[]):

        self.FIdxChn = FIdxChn
        self.FDLC = FDLC
        if self.FDLC > 15:
            self.FDLC = 15
        self.FIdentifier = FIdentifier

```

```

self.FProperties = FProperties
self.FFDProperties = FFDProperties
for i in range(len(FData)):
    self.FData[i] = FData[i]

```

## 成员：

**FData:** 帧数据。最大长度为 64Bytes

**FDLC:** 帧长度。

**FIdentifier:** 帧 ID，如果为 0xFFFFFFFF，表示当前帧为错误帧

**FIdxChn:** 帧通道，注意 CHANNEL\_INDEX。CHN1 = 0, 实际上是从 0 开始计算的。

**FTimeUS:** 帧时间戳，64 位 us 级时间戳。

**FFDProperties:** 存储 FD 相关的属性，如是否 FD 报文，发送过程中是否波特率可变。不同的字节位代表不同的属性值。

**FProperties:** 存储 CAN 相关的属性，比如是否远程帧，是否扩展帧。

其中，两个属性字节定义如下：

**【1】 FProperties:** CAN 属性定义：该参数默认为 0，共八个 bits，每一个位的定义如下：

Bit	意义
0	0: Rx 接收报文；1: Tx 发送报文
1	0: data frame 数据帧；1: remote frame 远程帧
2	0: std frame 标准帧；1: extended frame 扩展帧
3-5	Reserved
6	0: 不记录；1: 已经被记录
7	Reserved

**【2】 FDProperty:** FD 属性定义：

Bit	意义
0	0: 普通 CAN 报文；1: FDCAN 报文
1	0: 关闭 BRS；1: 开启 BRS
2	是否发生错误 (ESI Flag)
3-7	Reserved

// [7-3] tbd

// [2] ESI, The ERROR STATE INDICATOR (ESI) flag is transmitted dominant by error active nodes, recessive by error passive nodes. ESI does not exist in CAN format frames

// [1] BRS, If the bit is transmitted recessive, the bit rate is switched from the standard bit rate of the ARBITRATION PHASE to the preconfigured alternate bit rate of the DATA PHASE. If it is transmitted dominant, the bit rate is not switched. BRS does not exist in CAN format frames.

// [0] EDL: 0-normal CAN frame, 1-FD frame, added 2020-02-12, The EXTENDED DATA LENGTH (EDL) bit is recessive. It only exists in CAN FD format frames

## 调用示例:

### 3. TLIBLIN: LIN 总线数据类型

```
class TLIBLIN(Structure):
    _pack_ = 1
    _fields_ = [("FIdxChn", c_uint8),
                ("FErrStatus", c_uint8),
                ("FProperties", c_uint8),
                ("FDLC", c_uint8),
                ("FIdentifier", c_int8),
                ("FChecksum", c_uint8),
                ("FStatus", c_uint8),
                ("FTimeUs", c_int64),
                ("FData", c_uint8 * 8),
                ]
    def __init__(self, FIdxChn = 0, FDLC = 8, FIdentifier = 0x1, FProperties =
1, FData=[]):
        self.FIdxChn = FIdxChn
        self.FDLC = FDLC
        if self.FDLC > 8:
            self.FDLC = 8
        self.FIdentifier = FIdentifier
        self.FProperties = FProperties
        for i in range(len(FData)):
            self.FData[i] = FData[i]
```

## 成员:

**FData:** 帧数据。最大程度为 8Bytes

**FDLC:** 帧长度。

**FIdentifier:** 帧 ID, 如果为 0xFFFFFFFF, 表示当前帧为错误帧

**FIdxChn:** 帧通道, 注意 [CHANNEL\\_INDEX](#). CHN1 = 0, 实际上是从 0 开始计算的。

**FTimeUS:** 帧时间戳, 64 位 us 级时间戳。

**FProperties:** 存储 LIN 相关的属性, 比如报文方向, 是接收报文还是发送报文。

**FStatus:** 报文状态。

**FErrStatus:** 如果是错误帧, 对应的错误类型。

其中, 属性字节定义如下:

**【1】 Properties:** LIN 属性定义: 该参数默认为 0, 共八个 bits, 每一个位的定义如下:

Bit	意义
0	0: Rx 接收报文; 1: Tx 发送报文
1-3	Reserved
4-5	设备类型: 主节点, 从节点, 监听节点
6	0: 不记录; 1: 已经被记录
7	Reserved

## 4. 报文发送

## 5. 报文接收

### 1. 回调函数方式:

#### 简介:

设备接收到报文过后, 把报文整理成标准的 TCAN/TCANFD 数据结构。然后调用用户注册的接收回调函数, 通过参数把接收到的报文传递给调用者。libTSCAN 内部维护一个独立的线程, 每当消息达到后, 就会通过回调函数主动把数据传递给调用者, 用户不需要主动去调用读取函数。

#### 注册回调函数:

采用代理机制 (python 里面类 C 函数指针), 注册回调函数。需要注意的是, 一定要先申请一个代理对象, 然后把用户回调函数注册到该代理对象上, 如下所示:

```
PCANFD = POINTER(TLIBCANFD)
```

```
OnTx_RxFUNC_CANFD = WINFUNCTYPE(None, POINTER(c_int32), PCANFD)
```

```
PCAN = POINTER(TLIBCAN)
```

```
OnTx_RxFUNC_CAN = WINFUNCTYPE(None, POINTER(c_int32), PCAN)
```

```
PLIN = POINTER(TLIBLIN)
```

```
OnTx_RxFUNC_LIN = WINFUNCTYPE(None, POINTER(c_int32), PLIN)
```

## 回调函数使用

```
# 注册 can 发送—接收事件
def tsapp_register_event_can(obj: c_int32, OnFUNC):
    """
    obj = c_int32(0)
    #对通道 1 的 id 为 0x11 的报文 数据 1 进行打印输出
    def On_CAN_EVENT(OBJ, ACAN):
        if (ACAN.contents.FIdentifier == 0x11 and ACAN.contents.FIdxChn
    == 0):
            print(ACAN.FData[0])
    OnCANevent = OnTx_RxFUNC_CAN(On_CAN_EVENT)
    tsapp_register_event_can(obj, OnCANevent)
    """

    if isinstance(obj, int) or isinstance(obj, float):
        obj = c_int32(obj)
    r = dll.tsapp_register_event_can(byref(obj), OnFUNC)
    return r
```

### 2. 读取设备消息缓存的方式:

## 简介:

设备接收到报文过后，缓存在设备内部的 FIFO 中，外部程序调用函数接口从设备 FIFO 中把报文读取出来，FIFO 指针往后面移动；如果调用者一直不主动读取，会造成驱动内部 FIFO 溢出，最新的报文覆盖最旧的报文。

## tsfifo\_receive\_can\_msgs

```
# can fifo 接收
#ACANBuffers: TLIBCAN 数组
def tsfifo_receive_can_msgs(ACANBuffers: TLIBCAN, ACANBufferSize:
c_uint, AChn: CHANNEL_INDEX,
                             ARxTx: READ_TX_RX_DEF):
    """
    listcanmsg = (TLIBCAN * 100) ()

    listcanfdmsg = (TLIBCANFD * 100) ()

    cansize = c_int32(100)
```



```
    canfdsize = c_int32(100)

    tsfifo_receive_can_msgs(listcanmsg, cansize, 0,
    READ_TX_RX_DEF.TX_RX_MESSAGES)

    tsfifo_receive_canfd_msgs(listcanfdmsg, canfdsize, 0,
    READ_TX_RX_DEF.TX_RX_MESSAGES)
    """
    return dll.tsfifo_receive_can_msgs(ACANBuffers,
    byref(ACANBufferSize), AChn, ARxTx)
```

## tsfifo\_canfd\_receive\_buffers

```
# canfd 报文接收
#ACANFDBuffers: TLIBCANFD 数组
def tsfifo_receive_canfd_msgs(ACANFDBuffers, ACANFDBufferSize:
c_uint32, AChn: CHANNEL_INDEX,
                                ARxTx: READ_TX_RX_DEF):
    """
    listcanmsg = (TLIBCAN * 100) ()

    listcanfdmsg = (TLIBCANFD * 100) ()

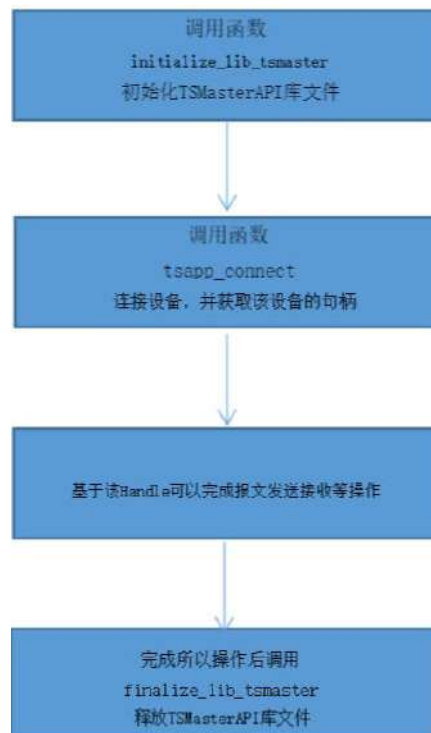
    cansize = c_int32(100)

    canfdsize = c_int32(100)

    tsfifo_receive_can_msgs(listcanmsg, cansize, 0,
    READ_TX_RX_DEF.TX_RX_MESSAGES)

    tsfifo_receive_canfd_msgs(listcanfdmsg, canfdsize, 0,
    READ_TX_RX_DEF.TX_RX_MESSAGES)
    """
    return dll.tsfifo_receive_canfd_msgs(ACANFDBuffers,
    byref(ACANFDBufferSize), AChn, ARxTx)
```

## 6.TSMasterAPI 调用流程



## 7.UDS 诊断接口说明

在完成 CAN 工具其他基本配置的基础上，诊断函数使用流程如下：



## 8.接口函数介绍

### 1. finalize\_lib\_tsmaster

函数名称	<code>finalize_lib_tsmaster()</code>
功能介绍	释放 TSMasterAPI 模块
调用位置	在退出程序之前，释放掉 TSMasterAPI 所使用的资源。
输入参数	无
返回值	无
示例	<code>finalize_lib_tsmaster()</code>

### 2. initialize\_lib\_tsmaster

函数名称	<code>initialize_lib_tsmaster(AppName: bytes)</code>
功能介绍	初始化 TSMasterAPI 模块
调用位置	在使用 TSMasterAPI 之前，必须先执行模块初始化函数，初始化内部参数，为驱动的运行准备好环境。
输入参数	参数:AppName 应用程序名称
返回值	无
示例	<code>initialize_lib_tsmaster("TSMaster".encode("utf8"))</code> 或 <code>initialize_lib_tsmaster(b"TSMaster")</code>

### 3. `tsapp_set_can_channel_count`

函数名称	<code>tsapp_set_can_channel_count(count: c_int32)</code>
功能介绍	设置应用程序的 CAN 通道数，可以通过 TSMaster 界面完成。
调用位置	映射 CAN 通道之前，用户可以根据应用程序的需求，分配需要用到的 CAN 通道数量。
输入参数	参数:count 需要用到的 CAN 通道数量
返回值	==0:设置成功 其他:失败，根据错误码查看错误类型。
示例	<code>count=c_int32(2)</code> <code>tsapp_set_can_channel_count(count)</code>

### 4. `tsapp_get_can_channel_count`

函数名称	<code>tsapp_get_can_channel_count(count: c_int32)</code>
功能介绍	获取 can 通道数
调用位置	
输入参数	参数:count 需要用到的 CAN 通道数量
返回值	==0:设置成功 其他:失败，根据错误码查看错误类型
示例	<code>count=c_int32(0)</code> <code>tsapp_get_can_channel_count(count)</code> <code>Print(count)</code>

5. `tsapp_set_lin_channel_count`

函数名称	<code>tsapp_set_lin_channel_count(count: c_int32)</code>
功能介绍	设置应用程序的 lin 通道数，可以通过 TSMaster 界面完成。
调用位置	映射 lin 通道之前，用户可以根据应用程序的需求，分配需要用到的 lin 通道数量。
输入参数	参数:count 需要用到的 lin 通道数量
返回值	<code>==0</code> :设置成功 其他:失败，根据错误码查看错误类型。
示例	<code>count=c_int32(2)</code> <code>tsapp_set_lin_channel_count(count)</code>

6. `tsapp_get_lin_channel_count`

函数名称	<code>tsapp_get_lin_channel_count(count: c_int32)</code>
功能介绍	获取应用程序的 lin 通道数，可以通过 TSMaster 界面完成。
调用位置	
输入参数	参数:count 需要用到的 lin 通道数量
返回值	<code>==0</code> :设置成功 其他:失败，根据错误码查看错误类型
示例	<code>count=c_int32(0)</code> <code>tsapp_get_lin_channel_count(count)</code> <code>Print(count)</code>

7. `tsapp_set_mapping_verbose`

函数名称	<code>tsapp_set_mapping_verbose</code> (AppName:bytes, TLIBApplicationChannelType: c_uint8, CHANNEL_INDEX: CHANNEL_INDEX, HW_name: str, BusToolDeviceType:c_int32, HW_Type:c_int32, AHardwareChannel:CHANNEL_INDEX, AEnableMapping: c_bool)
功能介绍	使用硬件设备之前, 为应用程序的通道(CAN/CANFD/LIN)映射(也就是绑定)指定 CAN 设备指定通道, 可在 TSMaster 管理界面完成, 代码中可以不需要同步操作。
调用位置	在应用程序连接 CAN 工具之前, 调用此函数完成硬件的绑定。
输入参数	<p>参数 1:AppName 应用程序名称, 绑定的名称要和 <code>initialize_lib_tsmaster</code> 中名称一致。</p> <p>参数 2:TLIBApplicationChannelType 通道类型, 包括 APP_CAN,APP_LIN。</p> <p>参数 3:CHANNEL_INDEX 应用程序的通道编号</p> <p>参数 4:HW_name 硬件名称, 比如 TOSUN, Vector。</p> <p>参数 5:BusToolDeviceType 硬件类型, 根据 TLIBBusToolDeviceType 囊括了市面上所有主流 CAN 卡硬件类型。工具枚举类型如下:</p> <pre>Publicenum TLIBBusToolDeviceType:int { BUS_UNKNOWN_TYPE=0, TS_TCP_DEVICE=1, XL_USB_DEVICE=2, TS_USB_DEVICE=3, PEAK_USB_DEVICE=4, KVASER_USB_DEVICE=5, ZLG_USB_DEVICE=6, ICS_USB_DEVICE=7, TS_TC1005_DEVICE=8};</pre> <p>参数 6:HW_Type 该设备下子设备, 比如 TS_USB_DEVICE 下面包含 TC1001, TC1011 等设备。</p> <p>参数 7:AHardwareChannel 硬件设备的通道, 比如 TC1005 有 5 个硬件通道。</p> <p>参数 8:AEnableMapping 是否使能该映射, 如果设置 False, 则该通道不能使用。</p>
返回值	<p>==0:设置成功</p> <p>其他:失败, 根据错误码查看错误类型。</p>
示例	<code>tsapp_set_mapping_verbose(b " appname ", TLIBApplicationChannelType.APP_CAN, CHANNEL_INDEX.CHN1, "TC1016".encode("UTF8"), TLIBBusToolDeviceType.TS_USB_DEVICE, TLIB_TS_Device_Sub_Type.TC1016, 0, True)</code>

8. `tsapp_del_mapping_verbose`

函数名称	<code>tsapp_del_mapping_verbose</code> (AppName:bytes , TLIBApplicationChannelType: c_uint8, APP_Channel: CHANNEL_INDEX)
功能介绍	删除映射消息，解除应用程序通道和硬件的绑定。
调用位置	想解除应用程序通道和硬件通道的绑定，调用此函数删除映射信息，解除配置信息即可。
输入参数	参数 1:AppName 应用程序名称,解除的名称和 initialize_lib_tsmaster 中名称一致。 参数 2:TLIBApplicationChannelType 应用程序通道类型，通道类型，包括 APP_CAN, APP_LIN。
返回值	==0:设置成功 其他:失败，根据错误码查看错误类型。
示例	<code>tsapp_del_mapping_verbose</code> (b " appname " , TLIBApplicationChannelType.APP_CAN, CHANNEL_INDEX.CHN1)

9. `tsapp_configure_baudrate_can`

函数名称	<code>tsapp_configure_baudrate_can</code> (APP_Channel:CHANNEL_INDEX, ABaudrateKbps:c_float, AListenOnly:c_bool, AInstallTermResistor1200hm: c_bool)
功能介绍	设置 CAN 通道的波特率等参数
调用位置	连接硬件设备之前，先设置通道参数。
输入参数	参数 1:APP_Channel 应用程序通道编号 参数 2:ABaudrateKbps 波特率 参数 3:AListenOnly 是否只听模式，如果开启，则只能接收数据。 参数 4:AInstallTermResistor1200hm 是否使能设备内部终端电阻
返回值	==0:设置成功 其他:失败，根据错误码查看错误类型。
示例	<code>ABaudrateKbps=c_float(500.0)</code> <code>tsapp_configure_baudrate_can</code> (CHANNEL_INDEX.CHN1, ABaudrateKbps, False, True)

10. `tsapp_configure_baudrate_canfd`

函数名称	<code>tsapp_configure_baudrate_canfd</code> (AIdxChn:CHANNEL_INDEX, ABaudrateArbKbps:c_float, ABaudrateDataKbps:c_float, AControllerType:c_int16, AControllerMode:c_int16, AInstallTermResistor1200hm:c_bool)
功能介绍	设置 CANFD 通道的波特率等参数
调用位置	连接硬件设备之前，先设置通道参数。
输入参数	参数 1:APP_Channel 应用程序通道编号 参数 2:ABaudrateKbps 仲裁场波特率 参数 3:ABaudrateDataKbps 数据场波特率 参数 4:AControllerType 控制器类型 参数 5:AControllerMode 控制器工作模式 参数 6:AInstallTermResistor1200hm 是否使能设备内部终端电阻
返回值	==0:设置成功 其他:失败，根据错误码查看错误类型。
示例	ABaudrateArbKbps=c_float(500.0) ABaudrateDataKbps=c_float(2000.0) <code>tsapp_configure_baudrate_canfd</code> (CHANNEL_INDEX.CHN1.value, ABaudrateKbps, ABaudrateDataKbps, fdtISOCANFD, fdmNormal, True)

11. `tsapp_configure_can_regs`

函数名称	<code>tsapp_configure_can_regs</code> (AIdxChn:CHANNEL_INDEX, ABaudrateKbps:float, ASEG1:int, ASEG2:int, APrescaler:int, ASJ2:int, AOnlyListen:int, A120:int)
功能介绍	设置 CAN 通道的采样率等参数
调用位置	连接硬件设备之前，先设置通道参数。
输入参数	参数 1:AIdxChn 应用程序通道编号 参数 2:ABaudrateKbps 仲裁场波特率 参数 3:ASEG1 相位缓冲段 1 参数 4:ASEG2 相位缓冲段 2 参数 5:APrescaler Prescaler 参数 6:ASJ2 SJ2 参数 7:AOnlyListen 是否只听模式，如果开启，则只能接收数据。 参数 8:A120 大于 0 表示激活终端电阻，=0 表示不激活
返回值	==0:设置成功 其他:失败，根据错误码查看错误类型
示例	ABaudrateKbps=c_float(500.0) <code>tsapp_configure_can_regs</code> (CHANNEL_INDEX.CHN1.value, ABaudrateKbps, 63, 16, 1, 80, 0, 1)



12. `tsapp_configure_baudrate_lin`

函数名称	<code>tsapp_configure_baudrate_lin</code> (AIdxChn:CHANNEL_INDEX, ABaudrateKbps: int, LIN_PROTOCOL: LIN_PROTOCOL)
功能介绍	设置 lin 通道的波特率等参数
调用位置	连接硬件设备之前，先设置通道参数。
输入参数	参数 1:AIdxChn 应用程序通道编号 参数 2:ABaudrateKbps 仲裁场波特率 参数 3:LIN_PROTOCOL LIN_PROTOCOL
返回值	==0:设置成功 其他:失败，根据错误码查看错误类型
示例	<code>tsapp_configure_baudrate_lin</code> (CHANNEL_INDEX.CHN1, 19.2, LIN_PROTOCOL.LIN_PROTOCOL_13)

13. `tslin_set_node_funtiontype`

函数名称	<code>tslin_set_node_funtiontype</code> (AIdxChn:CHANNEL_INDEX, TLINNodeType: T_LIN_NODE_FUNCTION)
功能介绍	设置 lin 节点工作模式：主节点，从节点，监听节点。
调用位置	初始化 lin 通道函数中
输入参数	参数 1:AIdxChn 应用程序通道编号 参数 2:TLINNodeType 0:主节点 1:从节点
返回值	==0:设置成功 其他:失败，根据错误码查看错误类型。
示例	<code>tslin_set_node_funtiontype</code> (CHANNEL_INDEX.CHN1, T_LIN_NODE_FUNCTION.T_MASTER_NODE) 主节点 <code>tslin_set_node_funtiontype</code> (CHANNEL_INDEX.CHN1, T_LIN_NODE_FUNCTION.T_SLAVE_NODE) 从节点

14. `tsapp_connect`

函数名称	<code>tsapp_connect</code> ()
功能介绍	连接应用程序。本函数执行的时候，会检查硬件设备通道是否全部就绪，硬件参数，完成设备和应用程序的连接。
调用位置	完成各个硬件参数设置后，调用此函数完成设备的连接，后面可以调用设备完成报文收发等功能。
输入参数	无
返回值	==0:设置成功 其他:失败，根据错误码查看错误类型。
示例	<code>tsapp_connect</code> ()

15. `tsapp_disconnect`

函数名称	<code>tsapp_disconnect()</code>
功能介绍	断开设备连接
调用位置	不需要使用硬件设备，调用此函数断开设备连接。
输入参数	无
返回值	==0:设置成功 其他:失败，根据错误码查看错误类型。
示例	<code>tsapp_disconnect()</code>

16. `tsapp_add_application`

函数名称	<code>tsapp_add_application(AppName: bytes)</code>
功能介绍	创建应用程序
调用位置	在初始化 TSMasterAPI 函数前调用此模块
输入参数	参数:AppName 后面初始化函数要和此名称保持一致
返回值	==0:设置成功 其他:失败，根据错误码查看错误类型
示例	<code>tsapp_add_application(b " appname " )</code>

17. `tsapp_del_application`

函数名称	<code>tsapp_del_application(AppName: bytes)</code>
功能介绍	删除应用程序
调用位置	在释放 TSMasterAPI 函数后调用此模块
输入参数	参数:AppName 要和初始化函数名称保持一致
返回值	==0:设置成功 其他:失败，根据错误码查看错误类型
示例	<code>tsapp_del_application(b " appname " )</code>

18. `tsapp_add_cyclic_msg_can`

函数名称	<code>tsapp_add_cyclic_msg_can(Msg: TLIBCAN, APeriodMS: c_float)</code>
功能介绍	增加周期发送的 CAN 报文，增加完成后，TSMasterAPI 自动完成报文的周期发送。
调用位置	在需要周期发送 CAN 报文的场合
输入参数	参数 1:Msg CAN 报文数据 参数 2:APeriodMS 周期值
返回值	==0:设置成功 其他:失败，根据错误码查看错误类型。
示例	Msg=TLIBCAN (FIdxChn=0, FDLC=8, FIdentifier=0X1, FProperties=1, FData=[1, 2, 3, 4, 5, 6, 7, 8]) APeriodMS= c_float(100.0) <code>tsapp_add_cyclic_msg_can(Msg, APeriodMS)</code>

19. `tsapp_del_cyclic_msg_can`

函数名称	<code>tsapp_del_cyclic_msg_can(Msg: TLIBCAN)</code>
功能介绍	删除周期发送 CAN 报文
调用位置	在需要周期发送 CAN 报文的场合
输入参数	参数:Msg 需要停止的 CAN 报文数据
返回值	==0:设置成功 其他:失败，根据错误码查看错误类型。
示例	Msg=TLIBCAN (FIdxChn=0, FDLC=8, FIdentifier=0X1, FProperties=1, FData=[1, 2, 3, 4, 5, 6, 7, 8]) <code>tsapp_del_cyclic_msg_can(Msg)</code>

20. `tsapp_add_cyclic_msg_canfd`

函数名称	<code>tsapp_add_cyclic_msg_canfd(Msg: TLIBCANFD, APeriodMS: c_float)</code>
功能介绍	增加周期发送的 CANFD 报文，增加完成后，TSMasterAPI 自动完成报文的周期发送。
调用位置	在需要周期发送 CANFD 报文的场合
输入参数	参数 1:Msg CANFD 报文数据 参数 2:APeriodMS 周期值
返回值	==0:设置成功 其他:失败，根据错误码查看错误类型。
示例	Msg=TLIBCANFD (FIdxChn=0, FDLC=8, FIdentifier=0X1, FProperties=1, FData=[1, 2, 3, 4, 5, 6, 7, 8]) APeriodMS= c_float(100.0) <code>tsapp_add_cyclic_msg_canfd(Msg, APeriodMS)</code>

21. `tsapp_del_cyclic_msg_canfd`

函数名称	<code>tsapp_del_cyclic_msg_canfd(Msg: TLIBCANFD)</code>
功能介绍	删除循环发送 <code>canfd</code> 报文
调用位置	在需要周期发送 <code>CANFD</code> 报文的场合
输入参数	参数 1:Msg CANFD 报文数据
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型。
示例	Msg=TLIBCANFD (FIdxChn=0, FDLC=8, FIdentifier=0X1, FProperties=1, FData=[1, 2, 3, 4, 5, 6, 7, 8]) <code>tsapp_del_cyclic_msg_canfd(Msg)</code>

22. `tsapp_delete_cyclic_msgs`

函数名称	<code>tsapp_delete_cyclic_msgs()</code>
功能介绍	删除所有循环发送报文
调用位置	在需要周期发送报文的场合
输入参数	无
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型。
示例	<code>tsapp_delete_cyclic_msgs()</code>

23. `tsapp_enable_bus_statistics`

函数名称	<code>tsapp_enable_bus_statistics(AEnable: c_bool)</code>
功能介绍	是否启用总线统计定时器来计算总线统计信息
调用位置	在需要计算总线统计信息地方调用
输入参数	参数:AEnable 是否使能总线统计定时器 True:开启 False:停止
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型。
示例	<code>tsapp_enable_bus_statistics(True)</code>

24. `tsapp_enumerate_hw_devices`

函数名称	<code>tsapp_enumerate_hw_devices</code> (ACount: <code>c_int32</code> )
功能介绍	枚举当前插在电脑上的能够使用的 CAN 设备数量
调用位置	在应用程序通道和硬件通道映射之前进行调用
输入参数	参数: AEnable 是否使能总线统计定时器 True:开启 False:停止
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型。
示例	<code>ACount=c_int32(0)</code> <code>tsapp_enumerate_hw_devices(ACount)</code>

25. `tsapp_get_hw_info_by_index`

函数名称	<code>tsapp_get_hw_info_by_index</code> (AIndex: <code>int</code> , PLIBHWInfo: TLIBHWInfo)
功能介绍	根据索引值获取硬件设备的信息
调用位置	在需要查询硬件信息的场合
输入参数	参数 1:AIndex 设备索引值 参数 2:PLIBHWInfo 结构体指针 硬件信息
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型。
示例	<pre>ACount = c_int32(0) tsapp_enumerate_hw_devices(ACount) print("在线硬件数量有%d个" % (ACount.value - 1)) PTLIBHWInfo = TLIBHWInfo() for i in range(ACount.value):     tsapp_get_hw_info_by_index(i, PTLIBHWInfo)     print(PTLIBHWInfo.FDeviceType,           PTLIBHWInfo.FDeviceIndex,           PTLIBHWInfo.FVendorName.decode("utf8"),           PTLIBHWInfo.FDeviceName.decode("utf8"),           PTLIBHWInfo.FSerialString.decode("utf8"))</pre>

26. `tsapp_get_error_description`

函数名称	<code>tsapp_get_error_description</code> (ACode: <code>c_int32</code> )
功能介绍	获取错误信息
调用位置	TSMasterAPI 执行后会返回错误编码值，调用此函数可以查看具体的错误编码含义。
输入参数	参数:ACode 错误编码值
返回值	错误编码代表具体含义字符串
示例	<pre>ret = tsapp_connect() if ret != 0:     tsapp_get_error_description(ret)</pre>

27. `tsapp_get_fps_can`

函数名称	<code>tsapp_get_fps_can</code> (AIdxChn:CHANNEL_INDEX, AIdentifier: <code>c_int32</code> , AFPS: <code>c_int32</code> )
功能介绍	获取 can 每秒帧数，需要先使能总线统计。
调用位置	在使能总线统计之后可以调用
输入参数	参数 1:AIdxChn 应用程序通道 参数 2:AIdentifier can 消息的标识符 参数 3:AFPS 帧每秒的特定标识符
返回值	==0:设置成功 其他:失败，根据错误码查看错误类型
示例	<pre>AFPS=c_int32(0) tsapp_enable_bus_statistics(True) tsapp_get_fps_can (CHANNEL_INDEX.CHN1, 0x123, AFPS) print (AFPS)</pre>

28. `tsapp_get_fps_canfd`

函数名称	<code>tsapp_get_fps_canfd</code> (AIdxChn: CHANNEL_INDEX, AIdentifier: <code>c_int32</code> , AFPS: <code>c_int32</code> )
功能介绍	获取 canfd 每秒帧数，需要先使能总线统计。
调用位置	在使能总线统计之后可以调用
输入参数	参数 1:AIdxChn 应用程序通道 参数 2:AIdentifier canfd 消息的标识符 参数 3:AFPS 帧每秒的特定标识符
返回值	==0:设置成功 其他:失败，根据错误码查看错误类型
示例	<pre>AFPS=c_int32(0) tsapp_enable_bus_statistics(True) tsapp_get_fps_canfd (CHANNEL_INDEX.CHN1, 0x123, AFPS) print (AFPS)</pre>

29. `tsapp_get_fps_lin`

函数名称	<code>tsapp_get_fps_lin</code> (AIdxChn:CHANNEL_INDEX, AIdentifier:c_int32, AFPS: c_int32)
功能介绍	获取 lin 每秒帧数，需要先使能总线统计。
调用位置	在使能总线统计之后可以调用
输入参数	参数 1:AIdxChn 应用程序通道 参数 2:AIdentifier lin 消息的标识符 参数 3:AFPS 帧每秒的特定标识符
返回值	==0:设置成功 其他:失败，根据错误码查看错误类型
示例	<pre>AFPS=c_int32(0) tsapp_enable_bus_statistics(True) tsapp_get_fps_lin (CHANNEL_INDEX.CHN1, 0x123, FPS) print (AFPS)</pre>

30. `tsapp_get_mapping_verbose`

函数名称	<code>tsapp_get_mapping_verbose</code> (APPName:str, ApplicationChannelType: c_int32, AMapping: TLIBTSMapping)
功能介绍	获取指定通道映射信息值
调用位置	在使能总线统计之后可以调用
输入参数	参数 1:APPName 应用程序名称要和初始化中名称一致 参数 2:ApplicationChannelType 应用程序通道类型 参数 3:AMapping 结构体类型 映射通道
返回值	==0:设置成功 其他:失败，根据错误码查看错误类型。
示例	<pre>AMapping=TLIBTSMapping() ChannelType= c_int32(0) tsapp_get_mapping_verbose(b " appname " , TLIBApplicationChannelType.APP_CAN, AMapping)</pre>

31. `tsapp_get_timestamp`

函数名称	<code>tsapp_get_timestamp</code> (ATimestamp: c_int32)
功能介绍	获取时间戳
调用位置	
输入参数	参数:ATimestamp 硬件时间戳
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>ATimestamp = c_int32(0)</code> <code>tsapp_get_timestamp(ATimestamp)</code>

32. `tsapp_get_turbo_mode`

函数名称	<code>tsapp_get_turbo_mode</code> (AEnable: c_bool)
功能介绍	获取极速模式是否开启
调用位置	在使用极速模式地方调用
输入参数	参数:AEnable 是否使能 True:开启 False:停止
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>tsapp_get_turbo_mode(True)</code>

33. `tsapp_set_turbo_mode`

函数名称	<code>tsapp_set_turbo_mode</code> (AEnable: c_bool)
功能介绍	设置开启极速模式
调用位置	
输入参数	参数:AEnable 是否使能 True:开启 False:停止
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>tsapp_set_turbo_mode(True)</code>



34. `tsfifo_enable_receive_fifo`

函数名称	<code>tsfifo_enable_receive_fifo()</code>
功能介绍	开启接受 FIFO 模式
调用位置	
输入参数	无
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>tsfifo_enable_receive_fifo()</code>

35. `tsfifo_disable_receive_fifo`

函数名称	<code>tsfifo_disable_receive_fifo()</code>
功能介绍	关闭接受 FIFO 模式
调用位置	
输入参数	无
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>tsfifo_disable_receive_fifo()</code>

36. `tsfifo_disable_receive_error_frames`

函数名称	<code>tsfifo_disable_receive_error_frames()</code>
功能介绍	关闭错误帧接收模式
调用位置	
输入参数	无
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>tsfifo_disable_receive_error_frames()</code>

37. `tsfifo_read_can_buffer_frame_count`

函数名称	<code>tsfifo_read_can_buffer_frame_count</code> (AIdxChn: CHANNEL_INDEX, ACount: c_int32)
功能介绍	读取通道 can 缓冲帧数量
调用位置	
输入参数	参数 1:AIdxChn 应用程序通道编号 参数 2:ACount can 报文数量
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	ACount=c_int32(0) <code>tsfifo_read_can_buffer_frame_count</code> (CHANNEL_INDEX.CHN1, ACount) <code>print</code> (ACount)

38. `tsfifo_read_can_tx_buffer_frame_count`

函数名称	<code>tsfifo_read_can_tx_buffer_frame_count</code> (AIdxChn:CHANNEL_INDEX, ACount: c_int32)
功能介绍	读取通道 can Tx 数量
调用位置	
输入参数	参数 1:AIdxChn 应用程序通道编号 参数 2:ACount can Tx 报文数量
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	ACount=c_int32(0) <code>tsfifo_read_can_tx_buffer_frame_count</code> (CHANNEL_INDEX.CHN1, Count) <code>print</code> (ACount)

39. `tsfifo_read_can_rx_buffer_frame_count`

函数名称	<code>tsfifo_read_can_rx_buffer_frame_count</code> (AIdxChn:CHANNEL_INDEX, ACount: c_int32)
功能介绍	读取通道 can rx 数量
调用位置	
输入参数	参数 1:AIdxChn 应用程序通道编号 参数 2:ACount can rx 报文数量
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	ACount=c_int32(0) <code>tsfifo_read_can_rx_buffer_frame_count</code> (CHANNEL_INDEX.CHN1, ACount) <code>print</code> (ACount)

40. `tsfifo_read_canfd_tx_buffer_frame_count`

函数名称	<code>tsfifo_read_canfd_tx_buffer_frame_count</code> (AIdxChn:CHANNEL_INDEX, ACount: c_int32)
功能介绍	读取通道 canfd tx 数量
调用位置	
输入参数	参数 1:AIdxChn 应用程序通道编号 参数 2:ACount canfd tx 报文数
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<pre>ACount=c_int32(0) tsfifo_read_canfd_tx_buffer_frame_count(CHANNEL_INDEX.CHN1, ACount) print(ACount)</pre>

41. `tsfifo_read_canfd_rx_buffer_frame_count`

函数名称	<code>tsfifo_read_canfd_rx_buffer_frame_count</code> (AIdxChn:CHANNEL_INDEX, ACount: c_int32)
功能介绍	读取通道 canfd rx 数量
调用位置	
输入参数	参数 1:AIdxChn 应用程序通道编号 参数 2:ACount canfd rx 报文数
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<pre>ACount=c_int32(0) tsfifo_read_canfd_rx_buffer_frame_count(CHANNEL_INDEX.CHN1. value, ACount) print(ACount)</pre>

42. `tsfifo_read_fastlin_buffer_frame_count`

函数名称	<code>tsfifo_read_fastlin_buffer_frame_count</code> (AIdxChn:CHANNEL_INDEX, ACount: c_int32)
功能介绍	读取通道 fastlin 缓冲帧数量
调用位置	
输入参数	参数 1:AIdxChn 应用程序通道编号 参数 2:ACount fastlin 报文数
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<pre>ACount=c_int32(0) tsfifo_read_fastlin_buffer_frame_count(CHANNEL_INDEX.CHN1, ACount) print(ACount)</pre>

43. `tsfifo_read_fastlin_tx_buffer_frame_count`

函数名称	<code>tsfifo_read_fastlin_tx_buffer_frame_count</code> (AIdxChn:CHANNEL_INDEX, ACount: c_int32)
功能介绍	读取通道 fastlin tx 数量
调用位置	
输入参数	参数 1:AIdxChn 应用程序通道编号 参数 2:ACount fastlin tx 报文数
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<pre>Count=c_int32(0) tsfifo_read_fastlin_tx_buffer_frame_count(CHANNEL_INDEX.CHN1.value, Count) print(ACount)</pre>

44. `tsfifo_read_fastlin_rx_buffer_frame_count`

函数名称	<code>tsfifo_read_fastlin_rx_buffer_frame_count</code> (AIdxChn: CHANNEL_INDEX, ACount: c_int32)
功能介绍	读取通道 fastlin rx 数量
调用位置	
输入参数	参数 1:AIdxChn 应用程序通道编号 参数 2:ACount fastlin rx 报文数
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<pre>Count=c_int32(0) tsfifo_read_fastlin_rx_buffer_frame_count(CHANNEL_INDEX.CHN1.value, Count) print(ACount)</pre>

45. `tsfifo_read_lin_buffer_frame_count`

函数名称	<code>tsfifo_read_lin_buffer_frame_count</code> (AIdxChn: CHANNEL_INDEX, ACount: c_int32)
功能介绍	读取通道 lin 缓冲帧数量
调用位置	
输入参数	参数 1:AIdxChn 应用程序通道编号 参数 2:ACount lin 报文数
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<pre>ACount=c_int32(0) tsfifo_read_lin_buffer_frame_count(CHANNEL_INDEX.CHN1, ACount) print(ACount)</pre>

46. `tsfifo_read_lin_tx_buffer_frame_count`

函数名称	<code>tsfifo_read_lin_tx_buffer_frame_count</code> (AIdxChn:CHANNEL_INDEX, ACount: c_int32)
功能介绍	读取通道 lin_tx 数量
调用位置	
输入参数	参数 1:AIdxChn 应用程序通道编号 参数 2:ACount tx 报文数
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	ACount=c_int32(0) <code>tsfifo_read_lin_tx_buffer_frame_count</code> (CHANNEL_INDEX.CHN1, ACount) <code>print</code> (ACount)

47. `tsfifo_read_lin_rx_buffer_frame_count`

函数名称	<code>tsfifo_read_lin_rx_buffer_frame_count</code> (AIdxChn:CHANNEL_INDEX, ACount: c_int32)
功能介绍	读取通道 lin_rx 数量
调用位置	
输入参数	参数 1:AIdxChn 应用程序通道编号 参数 2:ACount tx 报文数
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	ACount=c_int32(0) <code>tsfifo_read_lin_rx_buffer_frame_count</code> (CHANNEL_INDEX.CHN1, ACount) <code>print</code> (ACount)

48. `tsfifo_clear_can_receive_buffers`

函数名称	<code>tsfifo_clear_can_receive_buffers</code> (AIdxChn: CHANNEL_INDEX)
功能介绍	清除通道 can_receive_buffers
调用位置	
输入参数	参数:AIdxChn 应用程序通道编号
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>tsfifo_clear_can_receive_buffers</code> (CHANNEL_INDEX.CHN1)

49. `tsfifo_clear_canfd_receive_buffers`

函数名称	<code>tsfifo_clear_canfd_receive_buffers</code> (AIdxChn: CHANNEL_INDEX)
功能介绍	清除通道 <code>canfd_receive_buffers</code>
调用位置	
输入参数	参数:AIdxChn 应用程序通道编号
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>tsfifo_clear_canfd_receive_buffers</code> (CHANNEL_INDEX.CHN1)

50. `tsfifo_clear_fastlin_receive_buffers`

函数名称	<code>tsfifo_clear_fastlin_receive_buffers</code> (AIdxChn:CHANNEL_INDEX)
功能介绍	清除通道 <code>fastlin_receive_buffers</code>
调用位置	
输入参数	参数:AIdxChn 应用程序通道编号
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>tsfifo_clear_fastlin_receive_buffers</code> (CHANNEL_INDEX.CHN1)

51. `tsfifo_clear_lin_receive_buffers`

函数名称	<code>tsfifo_clear_lin_receive_buffers</code> (AIdxChn: CHANNEL_INDEX)
功能介绍	清除通道 <code>lin</code> 模块的 <code>buffers</code> 缓存
调用位置	在准备开始一次接收和发送之前, 调用此函数清除驱动中 <code>lin</code> 缓存的数据, 确保收到的数据是最新的。
输入参数	参数:AIdxChn 需要清除缓存的 <code>lin</code> 通道编号
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型。
示例	<code>tsfifo_clear_lin_receive_buffers</code> (CHANNEL_INDEX.CHN1)

52. `tsapp_register_pretx_event_canfd`

函数名称	<code>tsapp_register_pretx_event_canfd(obj: c_int32, OnFUNC)</code>
功能介绍	注册 <code>canfd</code> 预发送事件
调用位置	
输入参数	参数 1:obj 句柄 参数 2:OnFUNC 回调函数
返回值	<code>==0</code> :设置成功 其他:失败, 根据错误码查看错误类型
示例	<pre>obj = c_int32(0) #对通道 1 的 id 为 0x111 的报文数据 1 进行自增 def On_CANFD_EVENT(OBJ, ACAN):     if (ACAN.contents.FIdentifier == 0x111 and         ACAN.contents.FIdxChn == 0):         ACAN.contents.FData[0] +=1 OnCANFDevent = OnTx_RxFUNC_CANFD(On_CANFD_EVENT) tsapp_register_pretx_event_canfd(obj, OnCANFDevent)</pre>

53. `tsapp_register_pretx_event_can`

函数名称	<code>tsapp_register_pretx_event_can(obj: c_int32, OnFUNC)</code>
功能介绍	注册 <code>can</code> 预发送事件
调用位置	
输入参数	参数 1:obj 句柄 参数 2:OnFUNC 回调函数
返回值	<code>==0</code> :设置成功 其他:失败, 根据错误码查看错误类型
示例	<pre>obj = c_int32(0) #对通道 1 的 id 为 0x111 的报文数据 1 进行自增 def On_CAN_EVENT(OBJ, ACAN):     if (ACAN.contents.FIdentifier == 0x111 and         ACAN.contents.FIdxChn == 0):         ACAN.contents.FData[0] +=1 OnCANevent = OnTx_RxFUNC_CAN(On_CAN_EVENT) tsapp_register_pretx_event_can(obj, OnCANevent)</pre>



54. `tsapp_register_pretx_event_lin`

函数名称	<code>tsapp_register_pretx_event_lin(obj: c_int32, OnFUNC)</code>
功能介绍	注册 lin 预发送事件
调用位置	
输入参数	参数 1:obj 句柄 参数 2:OnFUNC 回调函数
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<pre>obj = c_int32(0) #对通道 1 的 id 为 0x111 的报文数据 1 进行自增 def On_LIN_EVENT(OBJ, ACAN):     if (ACAN.contents.FIdentifier == 0x11 and         ACAN.contents.FIdxChn == 0):         ACAN.contents.FData[0] +=1 OnLINEvent = OnTx_RxFUNC_LIN(On_LIN_EVENT) tsapp_register_pretx_event_lin(obj, OnLINEvent)</pre>

55. `tsapp_register_event_canfd`

函数名称	<code>tsapp_register_event_canfd(obj: c_int32, OnFUNC)</code>
功能介绍	注册 canfd 报文接收回调函数
调用位置	如果用户相基于接收回调机制处理接收报文, 在连接工具成功后可以调用此函数注册回调函数。
输入参数	参数 1:obj 句柄 参数 2:OnFUNC 处理接收报文的回调函数
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型。
示例	<pre>obj = c_int32(0) #对通道 1 的 id 为 0x11 的报文数据 1 进行打印输出 def On_CANFD_EVENT(OBJ, ACAN):     if (ACAN.contents.FIdentifier == 0x11 and         ACAN.contents.FIdxChn == 0):         print(ACAN.FData[0]) OnCANFDevent = OnTx_RxFUNC_CANFD(On_CANFD_EVENT) tsapp_register_event_canfd(obj, OnCANFDevent)</pre>

56. `tsapp_register_event_can`

函数名称	<code>tsapp_register_event_can(obj: c_int32, OnFUNC)</code>
功能介绍	注册 <code>can</code> 报文接收回调函数
调用位置	如果用户相基于接收回调机制处理接收报文，在连接工具成功后可以调用此函数注册回调函数。
输入参数	参数 1:obj 唯一句柄，可以起标识符作用 参数 2:OnFUNC 回调函数
返回值	<code>==0</code> :设置成功 其他:失败，根据错误码查看错误类型。
示例	<pre>obj = c_int32(0) #对通道 1 的 id 为 0x11 的报文数据 1 进行打印输出 def On_CAN_EVENT(OBJ, ACAN):     if (ACAN.contents.FIdentifier == 0x11 and         ACAN.contents.FIdxChn == 0):         print(ACAN.FData[0]) OnCANevent = OnTx_RxFUNC_CAN(On_CAN_EVENT) tsapp_register_event_can(obj, OnCANevent)</pre>

57. `tsapp_register_event_lin`

函数名称	<code>tsapp_register_event_lin(obj: c_int32, OnFUNC)</code>
功能介绍	注册 <code>lin</code> 报文接收回调函数
调用位置	如果用户相基于接收回调机制处理接收报文，在连接工具成功后可以调用此函数注册回调函数。
输入参数	参数 1:obj 唯一句柄，可以起标识符作用 参数 2:OnFUNC 回调函数
返回值	<code>==0</code> :设置成功 其他:失败，根据错误码查看错误类型。
示例	<pre>obj = c_int32(0) #对通道 1 的 id 为 0x11 的报文数据 1 进行打印输出 def On_LIN_EVENT(OBJ, ACAN):     if (ACAN.contents.FIdentifier == 0x11 and         ACAN.contents.FIdxChn == 0):         print(ACAN.FData[0]) OnLINEvent = OnTx_RxFUNC_CAN(On_LIN_EVENT) tsapp_register_event_lin(obj, OnLINEvent)</pre>

58. `tsapp_unregister_pretx_event_canfd`

函数名称	<code>tsapp_unregister_pretx_event_canfd(obj: c_int32, OnFUNC)</code>
功能介绍	注销 <code>canfd</code> 数据预发送函数
调用位置	
输入参数	参数 1:obj 唯一句柄, 可以起标识符作用 参数 2:OnFUNC 回调函数
返回值	<code>==0</code> :设置成功 其他:失败, 根据错误码查看错误类型。
示例	<pre>obj = c_int32(0) #对通道 1 的 id 为 0x11 的报文数据 1 进行打印输出 def On_CANFD_EVENT(OBJ, ACAN):     if (ACAN.contents.FIdentifier == 0x11 and         ACAN.contents.FIdxChn == 0):         print(ACAN.FData[0]) OnCANFDevent = OnTx_RxFUNC_CANFD(On_CANFD_EVENT) tsapp_unregister_pretx_event_canfd(obj, OnCANFDevent)</pre>

59. `tsapp_unregister_pretx_event_can`

函数名称	<code>tsapp_unregister_pretx_event_can(obj: c_int32, OnFUNC)</code>
功能介绍	注销 <code>can</code> 数据预发送函数
调用位置	
输入参数	参数 1:obj 唯一句柄, 可以起标识符作用 参数 2:OnFUNC 回调函数
返回值	<code>==0</code> :设置成功 其他:失败, 根据错误码查看错误类型。
示例	<pre>obj = c_int32(0) #对通道 1 的 id 为 0x11 的报文数据 1 进行打印输出 def On_CAN_EVENT(OBJ, ACAN):     if (ACAN.contents.FIdentifier == 0x11 and         ACAN.contents.FIdxChn == 0):         print(ACAN.FData[0]) OnCANevent = OnTx_RxFUNC_CAN(On_CAN_EVENT) tsapp_unregister_pretx_event_can(obj, OnCANevent)</pre>

60. `tsapp_unregister_pretx_event_lin`

函数名称	<code>tsapp_unregister_pretx_event_lin(obj: c_int32, OnFUNC)</code>
功能介绍	注销 lin 数据预发送函数
调用位置	
输入参数	参数 1:obj 唯一句柄, 可以起标识符作用 参数 2:OnFUNC 回调函数
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型。
示例	<pre>obj = c_int32(0) #对通道 1 的 id 为 0x11 的报文数据 1 进行打印输出 def On_LIN_EVENT(OBJ, ACAN):     if (ACAN.contents.FIdentifier == 0x11 and         ACAN.contents.FIdxChn == 0):         print(ACAN.FData[0]) OnLINEvent = OnTx_RxFUNC_CAN(On_LIN_EVENT) tsapp_unregister_pretx_event_lin(obj, OnLINEvent)</pre>

61. `tsapp_unregister_event_canfd`

函数名称	<code>tsapp_unregister_event_canfd(obj: c_int32, OnFUNC)</code>
功能介绍	注销 canfd 数据接收函数
调用位置	
输入参数	参数 1:obj 唯一句柄, 可以起标识符作用 参数 2:OnFUNC 回调函数
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型。
示例	<pre>obj = c_int32(0) #对通道 1 的 id 为 0x11 的报文数据 1 进行打印输出 def On_CANFD_EVENT(OBJ, ACAN):     if (ACAN.contents.FIdentifier == 0x11 and         ACAN.contents.FIdxChn == 0):         print(ACAN.FData[0]) OnCANFDevent = OnTx_RxFUNC_CANFD(On_CANFD_EVENT) tsapp_unregister_event_canfd(obj, OnCANFDevent)</pre>

62. `tsapp_unregister_event_can`

函数名称	<code>tsapp_unregister_event_can(obj: c_int32, OnFUNC)</code>
功能介绍	注销 can 数据接收函数
调用位置	
输入参数	参数 1:obj 唯一句柄，可以起标识符作用 参数 2:OnFUNC 回调函数
返回值	==0:设置成功 其他:失败，根据错误码查看错误类型。
示例	<pre>obj = c_int32(0) #对通道 1 的 id 为 0x11 的报文数据 1 进行打印输出 def On_CAN_EVENT(OBJ, ACAN):     if (ACAN.contents.FIdentifier == 0x11 and         ACAN.contents.FIdxChn == 0):         print(ACAN.FData[0]) OnCANevent = OnTx_RxFUNC_CAN(On_CAN_EVENT) tsapp_unregister_event_can(obj, OnCANevent)</pre>

63. `tsapp_unregister_event_lin`

函数名称	<code>tsapp_unregister_event_lin(obj: c_int32, OnFUNC)</code>
功能介绍	注销 lin 数据接收函数
调用位置	
输入参数	参数 1:obj 唯一句柄，可以起标识符作用。 参数 2:OnFUNC 处理接收报文的回调函数
返回值	==0:设置成功 其他:失败，根据错误码查看错误类型。
示例	<pre>obj = c_int32(0) #对通道 1 的 id 为 0x11 的报文数据 1 进行打印输出 def On_LIN_EVENT(OBJ, ACAN):     if (ACAN.contents.FIdentifier == 0x11 and         ACAN.contents.FIdxChn == 0):         print(ACAN.FData[0]) OnLINEvent = OnTx_RxFUNC_CAN(On_LIN_EVENT) tsapp_unregister_event_lin(obj, OnLINEvent)</pre>

64. `tsapp_unregister_pretx_events_canfd`

函数名称	<code>tsapp_unregister_pretx_events_canfd(obj: c_int32)</code>
功能介绍	注销 <code>canfd</code> 数据预发送函数
调用位置	
输入参数	参数: <code>obj</code> 唯一句柄, 可以起标识符作用。
返回值	<code>==0</code> : 设置成功 其他: 失败, 根据错误码查看错误类型。
示例	<code>obj=c_int32(0)</code> <code>tsapp_unregister_pretx_events_canfd(obj)</code>

65. `tsapp_unregister_pretx_events_can`

函数名称	<code>tsapp_unregister_pretx_events_can(obj: c_int32)</code>
功能介绍	注销 <code>can</code> 数据预发送函数
调用位置	
输入参数	参数: <code>obj</code> 唯一句柄, 可以起标识符作用。
返回值	<code>==0</code> : 设置成功 其他: 失败, 根据错误码查看错误类型。
示例	<code>obj=c_int32(0)</code> <code>tsapp_unregister_pretx_events_can(obj)</code>

66. `tsapp_unregister_pretx_events_lin`

函数名称	<code>tsapp_unregister_pretx_events_lin(obj: int)</code>
功能介绍	注销 <code>lin</code> 数据预发送函数
调用位置	
输入参数	参数: <code>obj</code> 唯一句柄, 可以起标识符作用。
返回值	<code>==0</code> : 设置成功 其他: 失败, 根据错误码查看错误类型。
示例	<code>obj=c_int32(0)</code> <code>tsapp_unregister_pretx_events_lin(obj)</code>

67. `tsapp_unregister_events_canfd`

函数名称	<code>tsapp_unregister_events_canfd(obj: c_int32)</code>
功能介绍	注销 <code>canfd</code> 数据预发送函数
调用位置	
输入参数	参数: <code>obj</code> 唯一句柄, 可以起标识符作用。
返回值	<code>==0</code> : 设置成功 其他: 失败, 根据错误码查看错误类型。
示例	<code>obj=c_int32(0)</code> <code>tsapp_unregister_events_canfd(obj)</code>

68. `tsapp_unregister_events_can`

函数名称	<code>tsapp_unregister_events_can(obj: int)</code>
功能介绍	注销 <code>can</code> 数据预发送函数
调用位置	
输入参数	参数: <code>obj</code> 唯一句柄, 可以起标识符作用。
返回值	<code>==0</code> : 设置成功 其他: 失败, 根据错误码查看错误类型。
示例	<code>obj=c_int32(0)</code> <code>tsapp_unregister_events_can(obj)</code>

69. `tsapp_unregister_events_lin`

函数名称	<code>tsapp_unregister_events_lin(obj: int)</code>
功能介绍	注销 <code>lin</code> 数据预发送函数
调用位置	
输入参数	参数: <code>obj</code> 唯一句柄, 可以起标识符作用。
返回值	<code>==0</code> : 设置成功 其他: 失败, 根据错误码查看错误类型。
示例	<code>obj=c_int32(0)</code> <code>tsapp_unregister_events_lin(obj)</code>

70. `tsapp_unregister_pretx_events_all`

函数名称	<code>tsapp_unregister_pretx_events_all()</code>
功能介绍	注销数据预发送函数
调用位置	
输入参数	无
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型。
示例	<code>tsapp_unregister_pretx_events_all()</code>

71. `tsapp_unregister_events_all`

函数名称	<code>tsapp_unregister_events_all()</code>
功能介绍	注销数据接收函数
调用位置	
输入参数	无
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型。
示例	<code>tsapp_unregister_events_all()</code>

72. `tsapp_show_tsmaster_window`

函数名称	<code>tsapp_show_tsmaster_window(AWindowName: str)</code>
功能介绍	打开 <code>tsmaster</code> 窗口
调用位置	
输入参数	参数: <code>AWindowName</code> 窗口名称, 要和内部定义名称保持一致。
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>tsapp_show_tsmaster_window(b "analyse ")</code>



73. `tsapp_start_logging`

函数名称	<code>tsapp_start_logging(filename: str)</code>
功能介绍	开始录制报文
调用位置	
输入参数	参数:filename filename
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>tsapp_start_logging(b"D:/1.blf")</code>

74. `tsapp_stop_logging`

函数名称	<code>tsapp_stop_logging()</code>
功能介绍	停止录制报文
调用位置	
输入参数	无
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>tsapp_stop_logging()</code>

75. `tsapp_transmit_can_async`

函数名称	<code>tsapp_transmit_can_async(Msg: TLIBCAN)</code>
功能介绍	异步发送单帧 can 报文
调用位置	异步发送 CAN 报文的场合
输入参数	参数:Msg 报文数据
返回值	==0: 发送成功 其他值: 发送失败, 查询错误码
示例	Msg=TLIBCAN (FIdxChn=0, FDLC=8, FIdentifier=0X1, FProperties=1, FFDProperties=1, FData=[1, 2, 3, 4, 5, 6, 7, 8]) <code>tsapp_transmit_can_async(Msg)</code>

76. `tsfifo_receive_can_msgs`

函数名称	<code>tsfifo_receive_can_msgs</code> (ACANBuffers:TLIBCAN, ACANBufferSize:c_uint, AChn:CHANNEL_INDEX, ARxTx: READ_TX_RX_DEF)
功能介绍	can fifo 接收
调用位置	
输入参数	参数 1:ACANBuffers ACAN 数组 参数 2:ACANBufferSize ACAN 数组大小 参数 3:AChn 接收数据的通道 参数 4:ARxTx 是否包含收发报文
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	ACANBuffers= (TLIBCAN * 100) () ACANBufferSize=c_int32(100) <code>tsfifo_receive_can_msgs</code> (ACANBuffers, ACANBufferSize, CHANNEL_INDEX.CHN1, READ_TX_RX_DEF.TX_RX_MESSAGES)

77. `tsapp_transmit_header_and_receive_msg`

函数名称	<code>tsapp_transmit_header_and_receive_msg</code> (AChn:CHANNEL_INDEX, ID:int, FDlc:c_uint8, receivedMsg: TLIBLIN, Timeout:c_int)
功能介绍	LIN 主节点: 发送 LIN 报文的 Header, 并接收从节点回复的 LIN 报文数据。
调用位置	此函数用于主节点, 发送 LIN 报文头, 并且接收从节点回复的数据。
输入参数	参数 1:AChn 通道编号 参数 2:ID 报文 id 参数 3:FDlc 要接收的数据直接长度 参数 4:receivedMsg 存储接收数据的报文 参数 5:Timeout 等待超时时间, 超过此时间即失败。
返回值	==0: 接收成功 其他值: 接收失败
示例	receivedMsg=TLIBLIN(FIdxChn=0, FDLC=8, FIdentifier=0x1, FProperties = 1, FData=[]) ID=0x123 FDlc=c_uint8(8) Timeout=c_int(10) <code>tsapp_transmit_header_and_receive_msg</code> (CHANNEL_INDEX.CHN1.value, ID, FDlc, receivedMsg, Timeout)

78. `tsapp_transmit_canfd_async`

函数名称	<code>tsapp_transmit_canfd_async</code> (Msg: TLIBCANFD)
功能介绍	异步发送单帧 canfd 报文
调用位置	在发送 canfd 报文的地方
输入参数	参数:Msg canfd 报文数据
返回值	==0: 接收成功 其他值: 接收失败
示例	Msg=TLIBCANFD (FIdxChn=0, FDLC=8, FIdentifier=0X1, FProperties=1, FFDProperties=1, FData=[1, 2, 3, 4, 5, 6, 7, 8]) <code>tsapp_transmit_canfd_async</code> (Msg)

79. `tsfifo_receive_canfd_msgs`

函数名称	<code>tsfifo_receive_canfd_msgs</code> (ACANFDBuffers, ACANFDBufferSize:c_uint32, AChn: CHANNEL_INDEX, ARxTx: READ_TX_RX_DEF)
功能介绍	canfd 报文接收
调用位置	
输入参数	参数 1:ACANFDBuffers ACAN 数组 参数 2:ACANFDBufferSize ACAN 数组大小 参数 3:AChn 通道编号 参数 4:ARxTx 是否收发报文
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	ACANFDBuffers = (TLIBCAN * 100) () ACANFDBufferSize=c_int32(100) <code>tsfifo_receive_canfd_msgs</code> (ACANFDBuffers , ACANFDBufferSize, CHANNEL_INDEX.CHN1, READ_TX_RX_DEF.TX_RX_MESSAGES)

80. `tsapp_transmit_lin_async`

函数名称	<code>tsapp_transmit_lin_async</code> (Msg: TLIBLIN)
功能介绍	异步发送单帧 lin 报文
调用位置	在需要发送 LIN 报文的场合
输入参数	参数:Msg lin 报文数据
返回值	==0: 接收成功 其他值: 接收失败
示例	Msg=TLIBCANFD (FIdxChn=0, FDLC=8, FIdentifier=0X1, FProperties=1, FData=[1, 2, 3, 4, 5, 6, 7, 8]) <code>tsapp_transmit_lin_async</code> (Msg)

81. `tsapp_receive_lin_msgs`

函数名称	<code>tsapp_receive_lin_msgs</code> (ALINBuffers, ALINBufferSize: c_int, AChn: CHANNEL_INDEX, ARxTx: READ_TX_RX_DEF)
功能介绍	lin 报文接收
调用位置	
输入参数	参数 1:ACANFDBuffers ALIN 数组 参数 2:ACANFDBufferSize ALIN 数组大小 参数 3:AChn 通道编号 参数 4:ARxTx 是否收发报文
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	ALINBuffers= (TLIBCAN * 100) () ALINBufferSize=c_int32(100) <code>tsapp_receive_lin_msgs</code> (ALINBuffers, ALINBufferSize, CHANNEL_INDEX.CHN1, READ_TX_RX_DEF.TX_RX_MESSAGES)

82. `tsapp_transmit_can_sync`

函数名称	<code>tsapp_transmit_can_sync</code> (Msg: TLIBCAN, ATimeoutMS: c_int32)
功能介绍	同步发送 CAN 报文, 并检测到发送成功后, 才退出此函数。此函数返回成功, 代表 CAN 报文一定已经成功发送到了 CAN 总线上面。
调用位置	同步发送 CAN 报文的场合
输入参数	参数 1:Msg can 报文数据 参数 2:ATimeoutMS 同步等待的超时时间
返回值	==0: 发送成功 其他值: 发送失败, 查询错误码
示例	Msg=TLIBCAN (FIdxChn=0, FDLC=8, FIdentifier=0X1, FProperties=1, FData=[1, 2, 3, 4, 5, 6, 7, 8]) TimeoutMS= c_int32(100) <code>tsapp_transmit_can_sync</code> (Msg, TimeoutMS)

83. `tsapp_transmit_canfd_sync`

函数名称	<code>tsapp_transmit_canfd_sync</code> (Msg:TLIBCANFD, ATimeoutMS:c_int32)
功能介绍	同步发送单帧 canfd 报文
调用位置	同步发送 CANfd 报文的场合
输入参数	参数 1:Msg can 报文数据 参数 2:ATimeoutMS 同步等待的超时时间
返回值	==0: 发送成功 其他值: 发送失败, 查询错误码
示例	Msg=TLIBCAN (FIdxChn=0, FDLC=8, FIdentifier=0X1, FProperties=1, FData=[1, 2, 3, 4, 5, 6, 7, 8]) ATimeoutMS=c_int32(10) <code>tsapp_transmit_canfd_sync</code> (Msg, ATimeoutMS)

84. `tsapp_transmit_lin_sync`

函数名称	<code>tsapp_transmit_lin_sync</code> (Msg:TLIBLIN, ATimeoutMS:c_int32)
功能介绍	同步发送单帧 lin 报文
调用位置	同步发送 lin 报文的场合
输入参数	参数 1:Msg can 报文数据 参数 2:ATimeoutMS 同步等待的超时时间
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	Msg=TLIBLIN (FIdxChn=0, FDLC=8, FIdentifier=0X1, FProperties=1, FData=[1, 2, 3, 4, 5, 6, 7, 8]) ATimeoutMS:c_int32(100) <code>tsapp_transmit_lin_sync</code> (Msg, ATimeoutMS)

85. `tscom_can_rbs_start`

函数名称	<code>tscom_can_rbs_start</code> ()
功能介绍	开启 rbs
调用位置	
输入参数	无
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>tscom_can_rbs_start</code> ()

86. `tscom_can_rbs_stop`

函数名称	<code>tscom_can_rbs_stop()</code>
功能介绍	停止 rbs
调用位置	
输入参数	无
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>tscom_can_rbs_stop()</code>

87. `tscom_can_rbs_is_running`

函数名称	<code>tscom_can_rbs_is_running(AIsRunning: c_bool)</code>
功能介绍	rbs 是否开启
调用位置	
输入参数	参数:AIsRunning rbs 是否开启
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>tscom_can_rbs_is_running(True)</code> 或 <code>tscom_can_rbs_is_running(false)</code>

88. `tscom_can_rbs_configure`

函数名称	<code>tscom_can_rbs_configure(AAutoStart:c_bool,AAutoSendOnModification:c_bool,AActivateNodeSimulation:c_bool,TLIBRBSInitValueOptions:c_int)</code>
功能介绍	配置 rbs
调用位置	
输入参数	参数 1:AAutoStart 是否自动启动 参数 2:AAutoSendOnModification 是否自动修正发送报文 参数 3:AActivateNodeSimulation 是否自动启动节点仿真 参数 4:TLIBRBSInitValueOptions 枚举值, 信号值初始值设置。
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>TLIBRBSInitValueOptions=c_int(0)</code> <code>tscom_can_rbs_configure(True, True, True, TLIBRBSInitValueOptions)</code>

89. `tscom_can_rbs_activate_network_by_name`

函数名称	<code>tscom_can_rbs_activate_network_by_name</code> (AIdxChn:c_int32, AEnable:bool, ANetworkName: str, AIncludingChildren: bool)
功能介绍	激活 network
调用位置	
输入参数	参数 1:AIdxChn 通道编号 参数 2:AEnable 是否激活 参数 3:ANetworkName network name 参数 4:AIncludingChildren AIncludingChildren
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>tscom_can_rbs_activate_network_by_name</code> (CHANNEL_INDEX.CHN1, True, b " CAN_FD_Powertrain " , True)

90. `tscom_can_rbs_activate_node_by_name`

函数名称	<code>tscom_can_rbs_activate_node_by_name</code> (AIdxChn:c_int32, AEnable: bool, ANetworkName: str, NodeName: str, AIncludingChildren: bool)
功能介绍	激活节点
调用位置	
输入参数	参数 1:AIdxChn 通道编号 参数 2:AEnable 是否激活 参数 3:ANetworkName network name 参数 4:NodeName 节点名称 参数 5:AIncludingChildren AIncludingChildren
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>tscom_can_rbs_activate_node_by_name</code> (CHANNEL_INDEX.CHN1, True, b " CAN_FD_Powertrain " , b " Engine " , True)

91. `tscom_can_rbs_activate_message_by_name`

函数名称	<code>tscom_can_rbs_activate_message_by_name</code> (AIdxChn:c_int32, AEnable:bool, ANetworkName: str, NodeName: str, MessageName: str)
功能介绍	激活报文
调用位置	
输入参数	参数 1:AIdxChn 应用程序通道 参数 2:AEnable 是否激活 参数 3:ANetworkName network name 参数 4:NodeName NodeName 参数 5:MessageName MessageName
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>tscom_can_rbs_activate_message_by_name</code> (CHANNEL_INDEX.CHN1, True, b " CAN_FD_Powertrain ", b " Engine ", b " EngineData ")

92. `tscom_can_rbs_activate_all_networks`

函数名称	<code>tscom_can_rbs_activate_all_networks</code> (AEnable:bool, AIncludingChildren: bool)
功能介绍	激活所有 networks
调用位置	
输入参数	参数 1:AEnable 是否激活 参数 2:AIncludingChildren AIncludingChildren
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>tscom_can_rbs_activate_all_networks</code> (True, True)

93. `tscom_can_rbs_enable`

函数名称	<code>tscom_can_rbs_enable</code> (AEnable:bool)
功能介绍	是否使能 rbs
调用位置	
输入参数	参数:AEnable 是否激活
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>tscom_can_rbs_enable</code> (True)



94. `tscom_can_rbs_get_signal_value_by_address`

函数名称	<code>tscom_can_rbs_get_signal_value_by_address</code> (ASymboladdress: str, Avalue:c_double)
功能介绍	获取指定信号值
调用位置	在需要获取信号值的场合调用
输入参数	参数 1:ASymboladdress 指定信号信息 参数 2:Avalue 设置的信号值
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	Avalue=c_double(0) <code>tscom_can_rbs_get_signal_value_by_address</code> (CHANNEL_INDEX.CHN1/b " CAN_FD_Powertrain " /b " Engine " /b " EngineData " /b " Gear " , Avalue)

95. `tscom_can_rbs_get_signal_value_by_element`

函数名称	<code>tscom_can_rbs_get_signal_value_by_element</code> (Aidchn: c_int32, ANetwork: str, ANodeName: str, AMessageName:str, ASignalName: str, Avalue: c_double)
功能介绍	获取指定信号值
调用位置	在需要获取信号值的场合调用
输入参数	参数 1:Aidchn 通道的编号 参数 2:ANetwork network name 参数 3:ANodeName 节点名字 参数 4:AMessageName 报文名称 参数 5:ASignalName 信号名字 参数 6:Avalue 信号值
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	Avalue=c_double(0) <code>tscom_can_rbs_get_signal_value_by_element</code> (CHANNEL_INDEX.CHN1,b " CAN_FD_Powertrain " ,b " Engine " ,b " EngineData " ,b " Gear " , Avalue)

96. `tscom_can_rbs_set_message_cycle_by_name`

函数名称	<code>tscom_can_rbs_set_message_cycle_by_name</code> (AIntervalMs:c_float, ANetwork: str, ANodeName: str, AMessageName: str)
功能介绍	设置信号值
调用位置	在设置报文信号的场合调用
输入参数	参数 1:AIntervalMs 毫秒为周期消息 参数 2:ANetwork network name 参数 3:ANodeName 节点名字 参数 4:AMessageName 报文名称
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	AIntervalMs=c_float(10.0) <code>tscom_can_rbs_set_message_cycle_by_name</code> (AIntervalMs,b "CAN_FD_Powertrain",b "Engine",b "EngineData")

97. `tscom_can_rbs_set_signal_value_by_address`

函数名称	<code>tscom_can_rbs_set_signal_value_by_address</code> (ASymboladdress:str, Avalue: c_double)
功能介绍	设置信号值
调用位置	在需要获取信号值的场合调用
输入参数	参数 1:ASymboladdress 信号信息 参数 2:Avalue 信号值
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	Avalue=c_double(0) <code>tscom_can_rbs_set_signal_value_by_address</code> (CHANNEL_INDEX.CHN1/b "CAN_FD_Powertrain" /b "Engine" /b "EngineData" /b "Gear", Avalue)

98. `tscom_can_rbs_set_signal_value_by_element`

函数名称	<code>tscom_can_rbs_set_signal_value_by_element</code> (AIdchn:c_int32, ANetwork: str, ANodeName: str, AMessageName: str, ASignalName: str, Avalue: c_double)
功能介绍	设置信号值
调用位置	在设置报文信号的场合调用
输入参数	参数 1:ASymboladdress Symboladdress 参数 2:ANetwork Network 参数 3:ANodeName 节点名称 参数 4:AMessageName 报文名称 参数 5:ASignalName 信号名称 参数 6:Avalue 信号值
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	Avalue=c_double(0) <code>tscom_can_rbs_set_signal_value_by_element</code> (CHANNEL_INDEX.CHN1, b " CAN_FD_Powertrain ", b " Engine ", b " EngineData ", b " Gear ", Avalue)

99. `tsdb_get_signal_value_can`

函数名称	<code>tsdb_get_signal_value_can</code> (ACAN:TLIBCAN, AMsgName:str, ASgnName: str, Avalue: c_double)
功能介绍	获取 can 信号值
调用位置	在获取 can 报文的场合调用
输入参数	参数 1:ASymboladdress Symboladdress 参数 2:Avalue 信号值
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	msg=TLIBCAN(FIdxChn=0, FDLC=8, FIdentifier=0X1, FProperties=1, FData=[1, 2, 3, 4, 5, 6, 7, 8]) Avalue=c_double(0) <code>tsdb_get_signal_value_can</code> (msg, b " EngineData ", b " Gear ", Avalue)

100. `tsdb_get_signal_value_canfd`

函数名称	<code>tsdb_get_signal_value_canfd</code> (ACANFD:TLIBCANFD, AMsgName:str, ASgnName:str, AValue:c_double)
功能介绍	获取 <code>canfd</code> 信号值
调用位置	在获取 <code>can</code> 报文的场合调用
输入参数	参数 1:ACANFD ACANFD 报文数据 参数 2:AMsgName 报文名称 参数 3:ASgnName 信号名称 参数 4:AValue 信号值
返回值	<code>==0</code> :设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>msg=TLIBCANFD(FIdxChn=0,FDLC=8,FIdentifier=0X1,FProperties=1,FFDProperties=1,FData=[1,2,3,4,5,6,7,8])</code> <code>AValue=c_double(0)</code> <code>tsdb_get_signal_value_canfd(msg,b " EngineData ", b " Gear " , AValue)</code>

101. `tsdb_set_signal_value_can`

函数名称	<code>tsdb_set_signal_value_can</code> (ACAN:TLIBCAN, AMsgName:str, ASgnName:str, AValue: c_double)
功能介绍	设置 <code>can</code> 信号值
调用位置	在设置 <code>can</code> 报文的场合调用
输入参数	参数 1:ACANFD ACANFD 报文数据 参数 2:AMsgName 报文名称 参数 3:ASgnName SgnName 参数 4:AValue 信号值
返回值	<code>==0</code> :设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>msg=TLIBCAN(FIdxChn=0,FDLC=8,FIdentifier=0X1,FProperties=1,FData=[1,2,3,4,5,6,7,8])</code> <code>AValue=c_double(20.0)</code> <code>tsdb_set_signal_value_can(msg,b " EngineData ", b " Gear " , AValue)</code>

102. `tsdb_set_signal_value_canfd`

函数名称	<code>tsdb_set_signal_value_canfd</code> (ACANFD:TLIBCANFD, AMsgName:str, ASgnName:str, AValue:c_double)
功能介绍	设置 <code>canfd</code> 信号值
调用位置	在设置 <code>canfd</code> 报文的场合调用
输入参数	参数 1:ACANFD ACANFD 报文数据 参数 2:AMsgName 报文名称 参数 3:ASgnName SgnName 参数 4:AValue 信号值
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>msg=TLIBCANFD(FIdxChn=0, FDLC=8, FIdentifier=0x1, FProperties=1, FFDProperties=1, FData=[1, 2, 3, 4, 5, 6, 7, 8])</code> <code>Value=c_double(20.0)</code> <code>tsdb_set_signal_value_canfd(msg, b " EngineData ", b " Gear ", AValue)</code>

103. `tsdb_load_can_db`

函数名称	<code>tsdb_load_can_db</code> (DBC_ADDRESS, ASupportedChannelsBased, idDBC:c_uint32)
功能介绍	加载 <code>dbc</code> 并绑定通道
调用位置	在加载 <code>dbc</code> 的场合调用
输入参数	参数 1:DBC_ADDRESS DBC 文件绝对路径 参数 2:ASupportedChannelsBased 绑定的通道 参数 3:idDBC DBC 句柄
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>idDBC=c_uint32(0)</code> <code>tsdb_load_can_db(b"C:/1.dbc", b"0,1", idDBC)</code>

104. `tsdb_unload_can_dbs`

函数名称	<code>tsdb_unload_can_dbs()</code>
功能介绍	解绑所有 <code>dbc</code>
调用位置	在需要解除绑定 <code>dbc</code> 的地方调用
输入参数	无
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>tsdb_unload_can_dbs()</code>

105. `tsdb_get_can_db_count`

函数名称	<code>tsdb_get_can_db_count</code> (ACount: <code>c_uint32</code> )
功能介绍	获取 dbc 数量
调用位置	在存放 dbc 的地方调用
输入参数	参数:ACount dbc 数量统计
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	ACount= <code>c_uint32</code> (0) <code>tsdb_get_can_db_count</code> (ACount)

106. `tsdb_get_can_db_id`

函数名称	<code>tsdb_get_can_db_id</code> (AIndex: <code>c_int32</code> , AId: <code>c_uint32</code> )
功能介绍	获取 dbc AId
调用位置	在存放 dbc 的地方调用
输入参数	参数 1:AIndex dbc 索引 参数 2:AId dbc AId
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	AIndex= <code>c_int32</code> (0) AId= <code>c_uint32</code> (0) <code>tsdb_get_can_db_id</code> (AIndex, AId)

107. `tsdb_get_can_db_info`

函数名称	<code>tsdb_get_can_db_info</code> (ADatabaseId: <code>c_int32</code> , AType: <code>c_int32</code> , AIndex: <code>c_int32</code> , ASubIndex: <code>c_int32</code> )
功能介绍	获取获取 dbc 信息
调用位置	在存放 dbc 的地方调用
输入参数	参数 1:ADatabaseId 数据库 Id 参数 2:AType 数据库类型 参数 3:AIndex 数据库索引 参数 4:ASubIndex SubIndex
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	ADatabaseId= <code>c_int32</code> (0) AType= <code>c_int32</code> (0) AIndex= <code>c_int32</code> (0) ASubIndex= <code>c_int32</code> (0) <code>tsdb_get_can_db_info</code> (ADatabaseId, AType, AIndex, ASubIndex)

108. `tslog_add_online_replay_config`

函数名称	<code>tslog_add_online_replay_config(AFileName:str, AIndex :c_int32)</code>
功能介绍	添加在线回放配置
调用位置	在总线记录结束后可调用
输入参数	参数 1:AFileName blf 文件名 参数 2:AIndex 文件索引
返回值	<code>==0</code> :设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>AIndex=c_int32(0)</code> <code>tslog_add_online_replay_config(b " replay1 " , AIndex)</code>

109. `tslog_set_online_replay_config`

函数名称	<code>tslog_set_online_replay_config(AIndex: c_int32, AName: str, AFileName: str, AAutoStart: c_bool, AIsRepetitiveMode: c_bool, AStartTimingMode:c_int32, AStartDelayTimeMs:c_int32, ASendTx: c_bool, ASendRx: c_bool, AMappings: c_char * 32)</code>
功能介绍	设置在线回放配置
调用位置	在添加在线回放配置后可调用
输入参数	参数 1:AIndex 回放文件索引 参数 2:AName Name 参数 3:AFileName 要重放的 blf 文件名称 参数 4:AAutoStart 连接应用程序是否自动回放日志文件 参数 5:AIsRepetitiveMode 文件是否重复回放 参数 6:AStartTimingMode 发送文件第一帧时间 参数 7:AStartDelayTimeMs 第一帧在指定延迟时间后以毫秒为单位发送 参数 8:ASendTx 文件中标记为 Tx 的帧将被回放 参数 9:ASendRx 文件中标记为 Rx 的帧将被回放 参数 10:AMappings Mappings
返回值	<code>==0</code> :设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>AIndex=c_int32(0)</code> <code>AStartTimingMode=c_int32(0)</code> <code>tslog_set_online_replay_config(AIndex, b " replay1 " , b " C:\Temp\log1.blf " , True, False, AStartTimingMode, 0, True, True, " 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32 " )</code>

110. `tslog_get_online_replay_count`

函数名称	<code>tslog_get_online_replay_count</code> (ACount: c_int32)
功能介绍	获取在线回放文件数量
调用位置	在总线记录结束后可调用
输入参数	参数: ACount 在线回放文件数量
返回值	==0: 设置成功 其他: 失败, 根据错误码查看错误类型
示例	<code>ACount=c_int32(0)</code> <code>tslog_get_online_replay_count</code> (ACount)

111. `tslog_get_online_replay_config`

函数名称	<code>tslog_get_online_replay_config</code> (AIndex: c_int32, AName: str, AFileName: str, AAutoStart: c_bool, AIsRepetitiveMode: c_bool, AStartTimingMode: c_int32, AStartDelayTimeMs: c_int32, ASendTx: c_bool, ASendRx: c_bool, AMappings: c_char * 32)
功能介绍	获取在线回放配置
调用位置	在总线记录结束后可调用
输入参数	参数 1: AIndex 回放文件索引 参数 2: AName Name 参数 3: AFileName 要重放的 blf 文件名称 参数 4: AAutoStart AutoStart 参数 5: AIsRepetitiveMode 文件是否重复回放 参数 6: AStartTimingMode 发送文件第一帧时间 参数 7: AStartDelayTimeMs 第一帧在指定延迟时间后以毫秒为单位发送 参数 8: ASendTx 文件中标记为 Tx 的帧将被回放 参数 9: ASendRx 文件中标记为 Rx 的帧将被回放 参数 10: AMappings Mappings
返回值	==0: 设置成功 其他: 失败, 根据错误码查看错误类型
示例	<code>ACount=c_int32(0)</code> <code>AStartTimingMode=c_int32(0)</code> <code>tslog_get_online_replay_config</code> (ACount, b "replay1", b "C:\Temp\log1.blf", True, false, AStartTimingMode, 0, True, True, " 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32 " )



112. `tslog_del_online_replay_config`

函数名称	<code>tslog_del_online_replay_config(AIndex:c_int32)</code>
功能介绍	删除在线回放配置
调用位置	
输入参数	参数:AIndex 回放文件索引
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>AIndex=c_int32(0)</code> <code>tslog_del_online_replay_config(AIndex)</code>

113. `tslog_del_online_replay_configs`

函数名称	<code>tslog_del_online_replay_configs()</code>
功能介绍	删除所有在线回放配置
调用位置	
输入参数	无
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>tslog_del_online_replay_configs()</code>

114. `tslog_start_online_replay`

函数名称	<code>tslog_start_online_replay(AInde:c_int32)</code>
功能介绍	开始在线回放
调用位置	
输入参数	参数:AInde 回放文件索引
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>AInde=c_int32(0)</code> <code>tslog_start_online_replay(AInde)</code>

115. `tslog_start_online_replays`

函数名称	<code>tslog_start_online_replays()</code>
功能介绍	所有文件开始回放
调用位置	
输入参数	无
返回值	<code>==0</code> :设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>tslog_start_online_replays()</code>

116. `tslog_pause_online_replay`

函数名称	<code>tslog_pause_online_replay(AInde:c_int32)</code>
功能介绍	暂停在线回放
调用位置	
输入参数	参数:AInde 回放文件索引
返回值	<code>==0</code> :设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>AInde=c_int32(0)</code> <code>tslog_pause_online_replay(AInde)</code>

117. `tslog_pause_online_replays`

函数名称	<code>tslog_pause_online_replays()</code>
功能介绍	所有文件暂停回放
调用位置	
输入参数	无
返回值	<code>==0</code> :设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>tslog_pause_online_replays()</code>

118. `tslog_stop_online_replay`

函数名称	<code>tslog_stop_online_replay(AInde:c_int32)</code>
功能介绍	停止在线回放
调用位置	
输入参数	参数:AInde 回放文件索引
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>AInde=c_int32(0)</code> <code>tslog_stop_online_replay(AInde)</code>

119. `tslog_stop_online_replays`

函数名称	<code>tslog_stop_online_replays()</code>
功能介绍	所有文件停止回放
调用位置	
输入参数	无
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>tslog_stop_online_replays()</code>

120. `tslog_get_online_replay_status`

函数名称	<code>tslog_get_online_replay_status(AIndex:c_int32, AStatus:c_int32, AProgressPercent100:c_float)</code>
功能介绍	获取在线回放状态
调用位置	
输入参数	参数 1:AIndex 回放文件索引 参数 2:AStatus 回放状态 1:启动回放 0:未启动回放 参数 3:AProgressPercent100 回放过程的百分比 从 0%到 100%
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>AIndex=c_int32(0)</code> <code>AStatus=c_int32(0)</code> <code>AProgressPercent100=c_float(0)</code> <code>tslog_get_online_replay_status(AIndex, AStatus, AProgressPercent100)</code>

121. `tslog_blf_read_start`

函数名称	<code>tslog_blf_read_start</code> (Pathfile:str, AHeadle:c_int32, ACount:c_int32)
功能介绍	开始读取 blf
调用位置	
输入参数	参数 1:Pathfile Pathfile 参数 2:AHeadle blf 句柄 参数 3:ACount blf 文件包含报文数
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	AHeadle=c_int32(0) ACount=c_int32(0) <code>tslog_blf_read_start</code> (b " D:/1.blf ", AHeadle, ACount)

122. `tslog_blf_read_object`

函数名称	<code>tslog_blf_read_object</code> (AHandle:c_int32, AProgressedCnt:c_int32, AType:TSupportedObjType, ACAN:TLIBCAN, ALIN:TLIBLIN, ACANFD:TLIBCANFD)
功能介绍	读当前报文的类型, 并赋值给报文指针。
调用位置	
输入参数	参数 1:AHandle blf 句柄 参数 2:AProgressedCnt 读取的报文指针 参数 3:AType 当前报文类型 参数 4:ACAN CAN 报文数据 参数 5:ALIN LIN 报文数据 参数 6:ACANFD CANFD 报文数据
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<pre>blfID = c_int32(0) count = c_ulong(0) tslog_blf_read_start(b'D:/1.blf', blfID, count) realCount = c_ulong(0) messageType = TSupportedObjType.sotUnknown CANtemp = TLIBCAN() CANFDtemp = TLIBCANFD() LINtemp = TLIBLIN() for i in range(count.value):     tslog_blf_read_object(blfID, realCount, messageType,         CANtemp, LINtemp, CANFDtemp)</pre>

123. `tslog_blf_read_end`

函数名称	<code>tslog_blf_read_end(AHandle: c_int64)</code>
功能介绍	结束写入
调用位置	
输入参数	参数:AHandle blf 句柄
返回值	<code>==0</code> :设置成功 其他:失败, 根据错误码查看错误类型
示例	<pre> AHandle=c_int64(0) blfID = c_int32(0) count = c_ulong(0) tslog_blf_read_start(b'D:/1.blf', blfID, count) realCount = c_ulong(0) messageType = TSupportedObjType.sotUnknown CANtemp = TLIBCAN() CANFDtemp = TLIBCANFD() LINTemp = TLIBLIN() for i in range(count.value):     tslog_blf_read_object(blfID, realCount, messageType, CANtemp, LINTemp, CANFDtemp)     if messageType.value == TSupportedObjType.sotCAN.value:         print(CANtemp.FTimeUs / 1000000, CANtemp.FIdxChn, CANtemp.FIdentifier, CANtemp.FProperties, CANtemp.FDLC, CANtemp.FData[0], CANtemp.FData[1], CANtemp.FData[2], CANtemp.FData[3], CANtemp.FData[4], CANtemp.FData[5], CANtemp.FData[6], CANtemp.FData[7]) tslog_blf_read_end(blfID) </pre>

124. `tslog_blf_write_start`

函数名称	<code>tslog_blf_write_start</code> (Pathfile: <code>str</code> , AHeadle: <code>c_int32</code> )
功能介绍	开始写 blf
调用位置	
输入参数	参数 1:Pathfile Pathfile 参数 2:AHeadle blf 句柄
返回值	<code>==0</code> :设置成功 其他:失败, 根据错误码查看错误类型
示例	Headle= <code>c_int32</code> (0) <code>tslog_blf_write_start</code> (b"D:/2.blf",Headle)

125. `tslog_blf_write_can`

函数名称	<code>tslog_blf_write_can</code> (AHeadle: <code>c_int32</code> , ACAN: <code>TLIBCAN</code> )
功能介绍	写入 can 报文
调用位置	
输入参数	参数 1:AHeadle blf 句柄 参数 2:ACAN can 报文数据
返回值	<code>==0</code> :设置成功 其他:失败, 根据错误码查看错误类型
示例	<pre>blfID = c_int32(0) count = c_ulong(0) tslog_blf_read_start(b'D:/1.blf', blfID, count) realCount = c_ulong(0) messageType = TSupportedObjType.sotUnknown CANtemp = TLIBCAN() CANFDtemp = TLIBCANFD() LINTemp = TLIBLIN() for i in range(count.value):     tslog_blf_read_object(blfID, realCount, messageType,         CANtemp, LINTemp, CANFDtemp)     if messageType.value == TSupportedObjType.sotCAN.value:         CANtemp.FIdxChn = 2         tslog_blf_write_can(writeHandle, CANtemp)</pre>

126. `tslog_blf_write_canfd`

函数名称	<code>tslog_blf_write_canfd</code> (AHeadle: <code>c_int32</code> , ACANFD: <code>TLIBCANFD</code> )
功能介绍	写入 <code>canfd</code> 报文
调用位置	
输入参数	参数 1:AHeadle blf 句柄 参数 2:ACANFD <code>canfd</code> 报文数据
返回值	<code>==0</code> :设置成功 其他:失败, 根据错误码查看错误类型
示例	<pre>blfID = c_int32(0) count = c_ulong(0) tslog_blf_read_start(b'D:/1.blf', blfID, count) realCount = c_ulong(0) messageType = TSupportedObjType.sotUnknown CANtemp = TLIBCAN() CANFDtemp = TLIBCANFD() LINTemp = TLIBLIN() for i in range(count.value):     tslog_blf_read_object(blfID, realCount, messageType,         CANtemp, LINTemp, CANFDtemp)     if messageType.value ==         TSupportedObjType.sotCANFD.value:         CANtemp.FIdxChn = 2         tslog_blf_write_canfd(writeHandle, CANFDtemp)</pre>

127. `tslog_blf_write_lin`

函数名称	<code>tslog_blf_write_lin</code> (AHeadle: <code>c_int32</code> , ALIN: <code>TLIBLIN</code> )
功能介绍	写入 lin 报文
调用位置	
输入参数	参数 1:AHeadle blf 句柄 参数 2:ALIN lin 报文数据
返回值	<code>==0</code> :设置成功 其他:失败, 根据错误码查看错误类型
示例	<pre> blfID = c_int32(0) count = c_ulong(0) tslog_blf_read_start(b'D:/1.blf', blfID, count) realCount = c_ulong(0) messageType = TSupportedObjType.sotUnknown CANtemp = TLIBCAN() CANFDtemp = TLIBCANFD() LINtemp = TLIBLIN() for i in range(count.value):     tslog_blf_read_object(blfID, realCount, messageType,         CANtemp, LINtemp, CANFDtemp)     if messageType.value == TSupportedObjType.sotLIN.value:         CANtemp.FIdxChn = 2         tslog_blf_write_lin(writeHandle, LINtemp) </pre>

128. `tslog_blf_write_end`

函数名称	<code>tslog_blf_write_end</code> (AHeadle: <code>c_int64</code> )
功能介绍	结束写入
调用位置	
输入参数	参数:AHeadle blf 句柄
返回值	<code>==0</code> :设置成功 其他:失败, 根据错误码查看错误类型
示例	<pre> AHeadle=c_int64(0) tslog_blf_write_end(AHeadle) </pre>



129. `tsdiag_can_create`

函数名称	<code>tsdiag_can_create</code> (udsHandle:c_int8, ChnIndex:CHANNEL_INDEX, ASupportFD:c_byte, AMaxdlc:c_byte, reqID:c_int32, ARequestIDIsStd:c_bool, resID:c_int32, resIsStd:c_bool, AFctID:c_int32, fctIsStd:c_bool)
功能介绍	创建诊断服务
调用位置	
输入参数	参数 1:udsHandle 诊断模块句柄 参数 2:ChnIndex 索引 参数 3:ASupportFD SupportFD 参数 4:AMaxdlc Maxdlc 参数 5:reqID 请求 id 参数 6:ARequestIDIsStd 请求 id 是否为标准数据帧 参数 7:resID 响应 id 参数 8:resIsStd 响应 id 是否为标准数据帧 参数 9:AFctID FctID 参数 10:fctIsStd 功能 id 是否为标准数据帧
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>pDiagModuleIndex=c_int8(0)</code> <code>tsdiag_can_create</code> (pDiagModuleIndex, 0, 0, 8, 0x123, True, 0X456, True, 0X789, True)

130. `tsdiag_set_p2_extended`

函数名称	<code>tsdiag_set_p2_extended</code> (pDiagModuleIndex:c_int8, TimeOut)
功能介绍	设置 <code>p2_extended</code> 时间
调用位置	
输入参数	参数:pDiagModuleIndex 诊断模块句柄 参数:TimeOut <code>extended</code> 时间
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>TimeOut=c_int32(200)</code> <code>pDiagModuleIndex = c_int8(0)</code> <code>tsdiag_set_p2_extended</code> (pDiagModuleIndex, timeOut)

131. `tsdiag_set_p2_timeout`

函数名称	<code>tsdiag_set_p2_timeout</code> (pDiagModuleIndex: c_int8, TimeOut)
功能介绍	设置 <code>p2_timeout</code> 时间
调用位置	
输入参数	参数:pDiagModuleIndex 诊断模块句柄 参数:TimeOut p2_timeout 时间
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	TimeOut=c_int32(200) pDiagModuleIndex = c_int8(0) <code>tsdiag_set_p2_timeout</code> (pDiagModuleIndex, TimeOut)

132. `tsdiag_set_s3_clienttime`

函数名称	<code>tsdiag_set_s3_clienttime</code> (pDiagModuleIndex: c_int8, TimeOut)
功能介绍	设置 <code>s3_clienttime</code> 时间
调用位置	
输入参数	参数:pDiagModuleIndex 诊断模块句柄 参数:TimeOut s3_clienttime 时间
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	TimeOut=c_int32(200) pDiagModuleIndex=c_int8(0) <code>tsdiag_set_s3_clienttime</code> (pDiagModuleIndex, TimeOut)

133. `tsdiag_set_s3_servertime`

函数名称	<code>tsdiag_set_s3_servertime</code> (pDiagModuleIndex:c_int8, TimeOut)
功能介绍	设置 <code>s3_servertime</code> 时间
调用位置	
输入参数	参数 1:pDiagModuleIndex 诊断模块句柄 参数 2:TimeOut s3_clienttime 时间
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	TimeOut=c_int32(200) pDiagModuleIndex=c_int8(0) <code>tsdiag_set_s3_servertime</code> (pDiagModuleIndex, TimeOut)

134. `tsdiag_can_delete`

函数名称	<code>tsdiag_can_delete(pDiagModuleIndex: c_int8)</code>
功能介绍	删除 <code>udsHandle</code> 对应诊断模块
调用位置	
输入参数	参数: <code>pDiagModuleIndex</code> 诊断模块句柄
返回值	<code>==0</code> :设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>pDiagModuleIndex=c_int8(0)</code> <code>tsdiag_can_delete(pDiagModuleIndex)</code>

135. `tsdiag_can_delete_all`

函数名称	<code>tsdiag_can_delete_all()</code>
功能介绍	删除所有诊断模块
调用位置	
输入参数	无
返回值	<code>==0</code> :设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>tsdiag_can_delete_all()</code>

136. `tstp_can_send_functional`

函数名称	<code>tstp_can_send_functional(pDiagModuleIndex:c_int8, AReqDataArray: bytearray, AReqDataSize: c_int32)</code>
功能介绍	<code>functionID</code> 发送
调用位置	
输入参数	参数 1: <code>pDiagModuleIndex</code> 诊断模块句柄 参数 2: <code>AReqDataArray</code> 发送的数据数组 参数 3: <code>AReqDataSize</code> 发送数据长度
返回值	<code>==0</code> :设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>pDiagModuleIndex=c_int8(0)</code> <code>AReqDataArray= bytes([10, 02])</code> <code>AReqDataSize=len(AReqDataArray)</code> <code>tstp_can_send_functional(pDiagModuleIndex, AReqDataArray, len(AReqDataArray))</code>

137. `tstp_can_send_request`

函数名称	<code>tstp_can_send_request</code> (pDiagModuleIndex:c_int8, AReqdataArray:bytearray, AReqDataSize:c_int32)
功能介绍	requestID 发送
调用位置	
输入参数	参数 1:pDiagModuleIndex 诊断模块句柄 参数 2:AReqdataArray 发送的数据数组 参数 3:AReqDataSize 发送数据长度
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<pre>pDiagModuleIndex = c_int8(0) AReqdataArray= bytes([10, 02]) AReqDataSize=len(AReqdataArray) tstp_can_send_request(pDiagModuleIndex, AReqdataArray, len(AReqdataArray))</pre>

138. `tstp_can_request_and_get_response`

函数名称	<code>tstp_can_request_and_get_response</code> (udsHandle:c_int8, dataIn:bytearray, ReqSize:c_int32, dataOut:bytearray, resSize:c_int32)
功能介绍	请求并获取响应数据 requestID 发送
调用位置	
输入参数	参数 1:udsHandle 诊断模块句柄 参数 2:dataIn 发送的数据数组 参数 3:ReqSize 发送的数据长度 参数 4:dataOut 接收 respondID 回复的数据数组 参数 5:resSize respondID 回复的数据长度
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<pre>pDiagModuleIndex=c_int8(0) AReqdataArray = bytes([0x10, 0x02]) max_len = 1000 AresponseDataArray = (c_uint8 * max_len)() AresponseDataSize = c_uint32(len(Aresdata)) tstp_can_request_and_get_response(pDiagModuleIndex, AReqdataArray, len(AReqdataArray), AresponseDataArray, AresponseDataSize)</pre>

139. `tsdiag_can_session_control`

函数名称	<code>tsdiag_can_session_control</code> (pDiagModuleIndex:c_int8, ASubSession:c_byte)
功能介绍	10 服务
调用位置	
输入参数	参数 1:udsHandle 诊断模块句柄 参数 2:ASubSession SubSession
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	ASubSession=c_byte(1) pDiagModuleIndex=c_int8(0) <code>tsdiag_can_session_control</code> (pDiagModuleIndex, ASubSession)

140. `tsdiag_can_routine_control`

函数名称	<code>tsdiag_can_routine_control</code> (pDiagModuleIndex:c_int8, ARoutineControlType:c_byte, ARoutintID:c_uint16)
功能介绍	31 服务
调用位置	
输入参数	参数 1:pDiagModuleIndex 诊断模块句柄 参数 2:ARoutineControlType RoutineControlType 参数 3:ARoutintID RoutintID
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	pDiagModuleIndex=c_int8(0) ARoutineControlType=c_uint8(1) ARoutintID=c_uint16(0xf100) <code>tsdiag_can_routine_control</code> (pDiagModuleIndex, ARoutineControlType, ARoutintID)

141. `tsdiag_can_communication_control`

函数名称	<code>tsdiag_can_communication_control</code> (pDiagModuleIndex:c_int8, AControlType:c_byte)
功能介绍	28 服务
调用位置	
输入参数	参数 1:pDiagModuleIndex 诊断模块句柄 参数 2:AControlType 控制类型
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>pDiagModuleIndex = c_int8(0)</code> <code>AControlType=c_uint8(1)</code> <code>tsdiag_can_communication_control</code> (pDiagModuleIndex, AControlType)

142. `tsdiag_can_security_access_request_seed`

函数名称	<code>tsdiag_can_security_access_request_seed</code> (pDiagModuleIndex:c_int8, ALevel:c_int32, ARecSeed:bytearray, ARecSeedSize:c_int32)
功能介绍	27 服务
调用位置	
输入参数	参数 1:pDiagModuleIndex 诊断模块句柄 参数 2:ALevel 安全访问等级 参数 3:ARecSeed RecSeed 参数 4:ARecSeedSize RecSeedSize
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>pDiagModuleIndex = c_int8(0)</code> <code>ALevel=c_int32(1)</code> <code>ARecSeed=(Bytearray*100)()</code> <code>ARecSeedSize= c_int32(100)</code> <code>tsdiag_can_security_access_request_seed</code> (pDiagModuleIndex, ALevel, ARecSeed, ARecSeedSize)

143. `tsdiag_can_security_access_send_key`

函数名称	<code>tsdiag_can_security_access_send_key</code> (pDiagModuleIndex:c_int8, ALevel:c_int32, AKeyValue:bytearray, AKeySize: c_int32)
功能介绍	27 服务
调用位置	
输入参数	参数 1:pDiagModuleIndex 诊断模块句柄 参数 2:ALevel 安全访问等级 参数 3:AKeyValue KeyValue 参数 4:AKeySize KeySize
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	pDiagModuleIndex=c_int8(0) AReqdataArray = bytes([0x27, 1, 1, 2, 3, 4]) <code>tsdiag_can_security_access_send_key</code> (pDiagModuleIndex, 2, AReqDataArray, len(AReqDataArray))

144. `tsdiag_can_request_download`

函数名称	<code>tsdiag_can_request_download</code> (pDiagModuleIndex:c_int8, AMemAddr:c_uint32, AMemSize:c_uint32)
功能介绍	34 服务
调用位置	
输入参数	参数 1:pDiagModuleIndex 诊断模块句柄 参数 2:AMemAddr MemAddr 参数 3:AMemSize MemSize
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	pDiagModuleIndex=c_int8(0) AMemAddr = c_uint32(0x80000010) AMemSize = c_uint32(0x1000) <code>tsdiag_can_request_download</code> (pDiagModuleIndex, MemAddr, AMemSize)

145. `tsdiag_can_request_upload`

函数名称	<code>tsdiag_can_request_upload</code> ( <code>pDiagModuleIndex:c_int8</code> , <code>AMemAddr:c_uint32</code> , <code>AMemSize:c_uint32</code> )
功能介绍	35 服务
调用位置	
输入参数	参数 1: <code>pDiagModuleIndex</code> 诊断模块句柄 参数 2: <code>AMemAddr MemAddr</code> 参数 3: <code>AMemSize MemSize</code>
返回值	<code>==0</code> :设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>pDiagModuleIndex = c_int8(0)</code> <code>AMemAddr = c_uint32(0x80000010)</code> <code>AMemSize = c_uint32(0x1000)</code> <code>tsdiag_can_request_upload</code> ( <code>pDiagModuleIndex</code> , <code>AMemAddr</code> , <code>AMemSize</code> )

146. `tsdiag_can_transfer_data`

函数名称	<code>tsdiag_can_transfer_data</code> ( <code>pDiagModuleIndex:c_int8</code> , <code>ASourceDatas: bytearray</code> , <code>ADataSize: c_int32</code> , <code>AReqCase: c_int32</code> )
功能介绍	36 服务
调用位置	
输入参数	参数 1: <code>pDiagModuleIndex</code> 诊断模块句柄 参数 2: <code>ASourceDatas SourceDatas</code> 参数 3: <code>ADataSize DataSize</code> 参数 4: <code>AReqCase ReqCase</code>
返回值	<code>==0</code> :设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>pDiagModuleIndex = c_int8(0)</code> <code>ASourceDatas = bytes[1, 2, 3, 4, 5, 6, 7, 2, 8]</code> <code>ADataSize= len(ASourceDatas)</code> <code>AReqCase=c_int32(1)</code> <code>tsdiag_can_transfer_data</code> ( <code>pDiagModuleIndex</code> , <code>ASourceDatas</code> , <code>len(ASourceDatas)</code> , <code>AReqCase</code> )



147. `tsdiag_can_request_transfer_exit`

函数名称	<code>tsdiag_can_request_transfer_exit(pDiagModuleIndex: c_int8)</code>
功能介绍	37 服务
调用位置	
输入参数	参数:pDiagModuleIndex 诊断模块句柄
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>pDiagModuleIndex = c_int8(0)</code> <code>tsdiag_can_request_transfer_exit(pDiagModuleIndex)</code>

148. `tsdiag_can_write_data_by_identifier`

函数名称	<code>tsdiag_can_write_data_by_identifier(pDiagModuleIndex:c_int8, ADataIdentifier:c_uint16, AWriteData:bytearray, AWriteDataSize: c_int32)</code>
功能介绍	2E 服务
调用位置	
输入参数	参数 1:pDiagModuleIndex 诊断模块句柄 参数 2:ADataIdentifier DataIdentifier 参数 3:AWriteData WriteData 参数 4:AWriteDataSize WriteDataSize
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>pDiagModuleIndex = c_int8(0)</code> <code>ADataIdentifier = c_uint16(0xf190)</code> <code>ASourceDatas = bytes[1, 2, 3, 4, 5, 6, 7, 2, 8]</code> <code>ADatasize= len(ASourceDatas)</code> <code>tsdiag_can_write_data_by_identifier(pDiagModuleIndex, ADataIdentifier, ASourceDatas, ADatasize)</code>

149. `tsdiag_can_read_data_by_identifier`

函数名称	<code>tsdiag_can_read_data_by_identifier</code> (pDiagModuleIndex:c_int8, ADataIdentifier:c_uint16, AReturnArray:bytearray, AReturnArraySize: c_int32)
功能介绍	22 服务
调用位置	
输入参数	参数 1:pDiagModuleIndex 诊断模块句柄 参数 2:ADataIdentifier DataIdentifier 参数 3:AReturnArray ReturnArray 参数 4:AReturnArraySize ReturnArraySize
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	pDiagModuleIndex = c_int8(0) ADataIdentifier=c_uint16(1) AReturnArray = (c_uint8*100) () AReturnArraySize = c_int32(len(AReturnArray)) <code>tsdiag_can_read_data_by_identifier</code> (pDiagModuleIndex, ADataIdentifier, AReturnArray, AReturnArraySize)

150. `tstp_lin_master_request`

函数名称	<code>tstp_lin_master_request</code> (AChnIdx: CHANNEL_INDEX, ANAD:c_int8, AData: bytearray, ADataNum: c_int, ATimeoutMs: c_int32)
功能介绍	Lin 诊断
调用位置	
输入参数	参数 1:AChnIdx 通道编号 参数 2:ANAD NAD 参数 3:AData 主节点发送数据 参数 4:ADataNum ADataNum 参数 5:ATimeoutMs 以毫秒为单位的超时时间
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	ANAD=c_int8(0) AData=(c_uint8*100) () ADataNum=c_int(0) ATimeoutMs=c_int32(10) <code>tstp_lin_master_request</code> (0, ANAD, AData, ADataNum, ATimeoutMs)

151. `tstp_lin_master_request_intervalms`

函数名称	<code>tstp_lin_master_request_intervalms</code> (AChnIdx: CHANNEL_INDEX, AData: c_int8)
功能介绍	Lin 诊断请求时间
调用位置	
输入参数	参数 1:AChnIdx 通道编号 参数 2:AData 主节点发送的数据
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	AData=c_int8(0) <code>tstp_lin_master_request_intervalms(0, AData)</code>

152. `tstp_lin_reset`

函数名称	<code>tstp_lin_reset</code> (AChnIdx: CHANNEL_INDEX)
功能介绍	Lin 通道连接
调用位置	
输入参数	参数:AChnIdx 应用程序通道
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>tstp_lin_reset(0)</code>

153. `tstp_lin_slave_response_intervalms`

函数名称	<code>tstp_lin_slave_response_intervalms</code> (AChnIdx: CHANNEL_INDEX, AData: c_int8)
功能介绍	Lin 诊断响应时间
调用位置	
输入参数	参数 1:AChnIdx 应用程序通道 参数 2:AData 从节点返回的数据
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	AData=c_int8(0) <code>tstp_lin_slave_response_intervalms(0, AData)</code>

154. `tsdiag_lin_read_data_by_identifier`

函数名称	<code>tsdiag_lin_read_data_by_identifier</code> (AChnIdx: CHANNEL_INDEX, ANAD: c_int8, AId: c_ushort, AResNAD: c_byte, AResData: bytearray, AResDataNum: c_int32, ATimeoutMS: c_int32)
功能介绍	Lin 诊断读取
调用位置	
输入参数	参数 1:AChnIdx 应用程序通道 参数 2:ANAD ANAD 参数 3:AId AId 参数 4:AResNAD AResNAD 参数 5:AResData ResData 参数 6:AResDataNum AResDataNum 参数 7:ATimeoutMS 以毫秒为单位的超时时间
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	ANAD=c_int8(0) AId=c_ushort(1) AResNAD=c_byte(0) AResData=bytes[1, 2, 3, 4, 5, 6, 7, 2, 8] AResDataNum=c_int32(0) ATimeoutMS=c_int32(10) <code>tsdiag_lin_read_data_by_identifier</code> (0, ANAD, AId, AResNAD, AResData, AResDataNum, ATimeoutMS)

155. `tsdiag_lin_write_data_by_identifier`

函数名称	<code>tsdiag_lin_write_data_by_identifier</code> (AChnIdx:CHANNEL_INDEX, ANAD:c_int8, AId:c_ushort, AReqData:bytearray, AReqDataNum:c_int32, AResNAD:c_byte, AResData:bytearray, AResDataNum:c_int32, ATimeoutMS: c_int32)
功能介绍	Lin 诊断写入
调用位置	
输入参数	参数 1:AChnIdx 应用程序通道 参数 2:ANAD NAD 参数 3:AId Id 参数 4:AResNAD ResNAD 参数 5:AResData ResData 参数 6:AResDataNum ResDataNum 参数 7:ATimeoutMS 以毫秒为单位的超时时间
返回值	==0:设置成功 其他:失败，根据错误码查看错误类型
示例	<pre> ANAD=c_int8(0) AId=c_ushort(1) AReqData=bytes[1, 2, 3, 4, 5, 6, 7, 2, 8] AReqDataNum=c_int32(0) AResNAD=c_byte(0) AResData=bytes[1, 2, 3, 4, 5, 6, 7, 2, 8] AResDataNum=c_int32(0) ATimeoutMS=c_int32(10) tsdiag_lin_write_data_by_identifier(0, ANAD, AId, AReqData, AReqDataNum, AResNAD, AResData, AResDataNum, ATimeoutMS) </pre>

156. `tsdiag_lin_session_control`

函数名称	<code>tsdiag_lin_session_control</code> (AChnIdx: CHANNEL_INDEX, ANAD: c_int8, ANewSession: c_byte, ATimeoutMS: c_int32)
功能介绍	Lin 诊断会话
调用位置	
输入参数	参数 1:AChnIdx 应用程序通道 参数 2:ANAD NAD 参数 3:ANewSession NewSession 参数 4:ATimeoutMS 以毫秒为单位的超时时间
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	ASession=c_byte(0) ANAD=c_int8(0) Atimeoutms=c_int32(10) <code>tsdiag_lin_session_control</code> (0, ANAD, ASession, Atimeoutms)

157. `tsdiag_lin_fault_memory_read`

函数名称	<code>tsdiag_lin_fault_memory_read</code> (AChnIdx: CHANNEL_INDEX, ANAD: c_int8, ANewSession: c_byte, ATimeoutMS: c_int32)
功能介绍	读取故障
调用位置	
输入参数	参数 1:AChnIdx 应用程序通道 参数 2:ANAD NAD 参数 3:ANewSession NewSession 参数 4:ATimeoutMS 以毫秒为单位的超时时间
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	ANAD=c_int8(0) ANewSession=c_byte(0) ATimeoutMS=c_int32(10) <code>tsdiag_lin_fault_memory_read</code> (0, ANAD, ANewSession, ATimeoutMS)

158. `tsdiag_lin_fault_memory_clear`

函数名称	<code>tsdiag_lin_fault_memory_clear</code> (AChnIdx: CHANNEL_INDEX, ANAD: c_int8, ANewSession: c_byte, ATimeoutMS: c_int32)
功能介绍	清除故障
调用位置	
输入参数	参数 1:AChnIdx 应用程序通道 参数 2:ANAD NAD 参数 3:ANewSession NewSession 参数 4:ATimeoutMS 以毫秒为单位的超时时间
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	ANAD=c_int8(0) ANewSession=c_byte(0) ATimeoutMS=c_int32(10) <code>tsdiag_lin_fault_memory_clear</code> (0, ANAD, ANewSession, ATimeoutMS)

159. `tsapp_transmit_flexray_sync`

函数名称	<code>tsapp_transmit_flexray_sync</code> (AFlexRay:TLIBFlexray, ATimeout:c_int32)
功能介绍	同步发送 FlexRay 报文
调用位置	
输入参数	参数 1:AChnIdx 应用程序通道 参数 2:ATimeout 超时时间
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	AFlexray=TLIBFlexray(FIdxChn=0, FSlotId=1, FChannelMask=1, FActualPayloadLength=32, FCycleNumber=1, FData=[1, 2, 3, 4, 5, 6, 7]) ATimeout = c_int32(10) <code>tsapp_transmit_flexray_sync</code> (AFlexray, TimeoutMS)

160. `tsapp_transmit_flexray_async`

函数名称	<code>tsapp_transmit_flexray_async</code> (AFlexRay: TLIBFlexray)
功能介绍	异步发送 FlexRay 报文
调用位置	
输入参数	参数: AFlexRay FlexRay 报文数据
返回值	==0: 设置成功 其他: 失败, 根据错误码查看错误类型
示例	AFlexray=TLIBFlexray (FIdxChn=0, FSlotId=1, FChannelMask=1, FActualPayloadLength=32, FCycleNumber=1, FData=[1, 2, 3, 4, 5, 6, 7]) <code>tsapp_transmit_flexray_async</code> (AFlexray)

161. `tsfifo_clear_flexray_receive_buffers`

函数名称	<code>tsfifo_clear_flexray_receive_buffers</code> (chn: c_int)
功能介绍	清空 flexray fifo buffer
调用位置	
输入参数	参数: chn 应用程序通道
返回值	==0: 设置成功 其他: 失败, 根据错误码查看错误类型
示例	chn = CHANNEL_INDEX.CHN1 <code>tsfifo_clear_flexray_receive_buffers</code> (chn)

162. `tsfifo_read_flexray_buffer_frame_count`

函数名称	<code>tsfifo_read_flexray_buffer_frame_count</code> (AIdxChn: c_int, ACount: c_int)
功能介绍	读取 flexray fifo buffer 总数量
调用位置	
输入参数	参数 1: AIdxChn 通道编号 参数 2: ACount 通道数量
返回值	==0: 设置成功 其他: 失败, 根据错误码查看错误类型
示例	chn = CHANNEL_INDEX.CHN1 ACount = c_int32(0) <code>tsfifo_read_flexray_buffer_frame_count</code> (chn, ACount)



163. `tsfifo_read_flexray_tx_buffer_frame_count`

函数名称	<code>tsfifo_read_flexray_tx_buffer_frame_count(AIdxChn:c_int, ACount:c_int)</code>
功能介绍	读取 flexray fifo buffer tx 数量
调用位置	
输入参数	参数 1:AIdxChn 应用程序通道 参数 2:ACount 通道数量
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>chn = CHANNEL_INDEX.CHN1</code> <code>ACount = c_int32(0)</code> <code>tsfifo_read_flexray_tx_buffer_frame_count(chn, ACount)</code>

164. `tsfifo_read_flexray_rx_buffer_frame_count`

函数名称	<code>tsfifo_read_flexray_rx_buffer_frame_count(AIdxChn:c_int, ACount:c_int)</code>
功能介绍	读取 flexray fifo buffer rx 数量
调用位置	
输入参数	参数 1:AIdxChn 应用程序通道 参数 2:ACount 通道数量
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>chn = CHANNEL_INDEX.CHN1</code> <code>ACount = c_int32(0)</code> <code>tsfifo_read_flexray_rx_buffer_frame_count(chn, ACount)</code>

165. `tsfifo_receive_flexray_msgs`

函数名称	<code>tsfifo_receive_flexray_msgs</code> (ADataBuffers:TLIBFlexray, ADataBufferSize:c_int, chn:c_int, ARxTx:c_int8)
功能介绍	获取 fifo 中 flexray 报文 当 ARxTx 非 0 时包含 TX 报文 为 0 时仅包含 RX 报文
调用位置	
输入参数	参数 1:ADataBuffers 缓存数据 参数 2:ADataBufferSize 缓存数据大小 参数 3:chn 通道编号 参数 4:ARxTx 等于 0 仅包含 RX 报文 非 0 时包含 TX 报文
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	ADataBuffers = (TLIBFlexray*100) () ADataBufferSize = c_int32(100) chn = CHANNEL_INDEX.CHN1 ARxTx = 0 <code>tsfifo_receive_flexray_msgs</code> (ADataBuffers, ADataBufferSize, chn, ARxTx)

166. `tsflexray_start_net`

函数名称	<code>tsflexray_start_net</code> (AChnIdx:CHANNEL_INDEX, ATimeout:c_int32)
功能介绍	启动 flexray 网络
调用位置	
输入参数	参数 1:AChnIdx 应用程序通道 参数 2:ATimeout 超时时间
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	chn = CHANNEL_INDEX.CHN1 ATimeout=c_int32(1000) <code>tsflexray_start_net</code> (chn, ATimeout)

167. `tsflexray_stop_net`

函数名称	<code>tsflexray_stop_net</code> (AChnIdx: CHANNEL_INDEX, ATimeout:c_int32)
功能介绍	停止 flexray 网络
调用位置	
输入参数	参数 1:AChnIdx 应用程序通道 参数 2:ATimeout 超时时间
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>chn = CHANNEL_INDEX.CHN1</code> <code>ATimeout=c_int32(1000)</code> <code>tsflexray_stop_net(chn , ATimeout)</code>

168. `tsflexray_wakeup_pattern`

函数名称	<code>tsflexray_wakeup_pattern</code> (AChnIdx:CHANNEL_INDEX, ATimeout:c_int32)
功能介绍	唤醒 flexray pattern
调用位置	
输入参数	参数 1:AChnIdx 应用程序通道 参数 2:ATimeout 超时时间
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>chn = CHANNEL_INDEX.CHN1</code> <code>ATimeout=c_int32(1000)</code> <code>tsflexray_wakeup_pattern(chn , ATimeout)</code>

169. `tsdb_load_flexray_db`

函数名称	<code>tsdb_load_flexray_db</code> (AFliepath:str, ASupportedChannels:str, AId:c_int32)
功能介绍	载入 flexray 数据库
调用位置	
输入参数	参数 1:AFliepath Fliepath 参数 2:ASupportedChannels SupportedChannels 参数 3:AId Id
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>AId = c_int32(0)</code> <code>tsdb_load_flexray_db(b"C:/1.xml", b'0,1', AId)</code>

170. `tsdb_unload_flexray_db`

函数名称	<code>tsdb_unload_flexray_db(AId:c_int32)</code>
功能介绍	卸载 flexray 数据库
调用位置	
输入参数	参数:AId 数据库 ID 编号
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>AId = c_int32(0)</code> <code>tsdb_unload_flexray_db(AId)</code>

171. `tsdb_unload_flexray_dbs`

函数名称	<code>tsdb_unload_flexray_dbs()</code>
功能介绍	卸载所有 flexray 数据库
调用位置	
输入参数	参数:AId 数据库 ID 编号
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>tsdb_unload_flexray_dbs()</code>

172. `tsdb_get_flexray_db_count`

函数名称	<code>tsdb_get_flexray_db_count(Acount:c_int32)</code>
功能介绍	获取加载的 flexray 数据库数量
调用位置	
输入参数	参数:Acount flexray 数据库数量
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>Acount=c_int32(0)</code> <code>tsdb_get_flexray_db_count(Acount)</code>

173. `tsdb_get_flexray_db_properties_by_address_verbose`

函数名称	<code>tsdb_get_flexray_db_properties_by_address_verbose(AAddr:str)</code>
功能介绍	通过地址获取 flexray 数据库属性信息
调用位置	
输入参数	参数:AAddr Addr
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	AAddr=b"0/network1" <code>tsdb_get_flexray_db_properties_by_address_verbose(b"0/network1")</code>

174. `tsdb_get_flexray_db_properties_by_index_verbose`

函数名称	<code>tsdb_get_flexray_db_properties_by_index_verbose(ADBIndex:c_int32)</code>
功能介绍	通过索引获取 flexray 数据库属性信息
调用位置	
输入参数	参数:ADBIndex 数据库索引
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	ADBIndex=c_int32(0) <code>tsdb_get_flexray_db_properties_by_index_verbose(ADBIndex)</code>

175. `tsdb_get_flexray_ecu_properties_by_address_verbose`

函数名称	<code>tsdb_get_flexray_ecu_properties_by_address_verbose(AAddr:str)</code>
功能介绍	通过地址获取数据库 ECU 信息
调用位置	
输入参数	参数:AAddr Addr
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	AAddr=b"0/network1/ecu1" <code>tsdb_get_flexray_ecu_properties_by_address_verbose(b"0/network1/ecu1")</code>

176. `tsdb_get_flexray_ecu_properties_by_index_verbose`

函数名称	<code>tsdb_get_flexray_ecu_properties_by_index_verbose</code> (ADBIndex:c_int32, AECUIndex:c_int32)
功能介绍	通过索引获取数据库 ECU 信息
调用位置	
输入参数	参数 1:ADBIndex 数据库索引 参数 2:AECUIndex ECU 索引
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	ADBIndex=c_int32(0) AECUIndex=c_int32(0) <code>tsdb_get_flexray_ecu_properties_by_index_verbose</code> (ADBIndex, AECUIndex)

177. `tsdb_get_flexray_frame_properties_by_address_verbose`

函数名称	<code>tsdb_get_flexray_frame_properties_by_address_verbose</code> (AAddr:str)
功能介绍	通过地址获取数据库 frame 信息
调用位置	
输入参数	参数:AAddr Addr
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>tsdb_get_flexray_frame_properties_by_address_verbose</code> (b"0/network1/ecu1/frame1")

178. [tsdb\\_get\\_flexray\\_frame\\_properties\\_by\\_index\\_verbose](#)

函数名称	<a href="#">tsdb_get_flexray_frame_properties_by_index_verbose</a> (ADBIndex:c_int32, AECUIndex:c_int32, AFrameIndex:c_int32, AIsTx:c_bool)
功能介绍	通过索引获取数据库 frame 信息
调用位置	
输入参数	参数 1:ADBIndex DBIndex 参数 2:AECUIndex ECUIndex 参数 3:AFrameIndex FrameIndex 参数 4:AIsTx 是否发送
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	ADBIndex=c_int32(0) AECUIndex=c_int32(0) AFrameIndex=c_int32(0) <a href="#">tsdb_get_flexray_frame_properties_by_index_verbose</a> (ADBIndex, AECUIndex, AFrameIndex, False)

179. [tsdb\\_get\\_flexray\\_signal\\_properties\\_by\\_address\\_verbose](#)

函数名称	<a href="#">tsdb_get_flexray_signal_properties_by_address_verbose</a> (AAddr:str)
功能介绍	通过地址获取数据库 signal 信息
调用位置	
输入参数	参数:AAddr Addr
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<a href="#">tsdb_get_flexray_signal_properties_by_address_verbose</a> (b"0/network1/ecu1/frame1/signal1")

180. `tsdb_get_flexray_signal_properties_by_index_verbose`

函数名称	<code>tsdb_get_flexray_signal_properties_by_index_verbose</code> (ADBIndex:c_int32, AECUIndex:c_int32, AFrameIndex:c_int32, ASignalIndex:c_int32, AIsTx:c_bool)
功能介绍	通过索引获取数据库 signal 信息
调用位置	
输入参数	参数 1:ADBIndex DBIndex 参数 2:AECUIndex ECUIndex 参数 3:AFrameIndex FrameIndex 参数 4:ASignalIndex SignalIndex
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	ADBIndex=c_int32(0) AECUIndex=c_int32(0) AFrameIndex=c_int32(0) ASignalIndex=c_int32(0) <code>tsdb_get_flexray_signal_properties_by_index_verbose</code> (ADBIndex, AECUIndex, AFrameIndex, ASignalIndex, False)

181. `tsdb_get_flexray_db_id`

函数名称	<code>tsdb_get_flexray_db_id</code> (AIndex:c_int32)
功能介绍	通过索引获取数据库 id
调用位置	
输入参数	参数:AIndex 索引
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	AIndex=c_int32(0) <code>tsdb_get_flexray_db_id</code> (AIndex)

182. `tscom_flexray_rbs_start`

函数名称	<code>tscom_flexray_rbs_start</code> ()
功能介绍	启动 flexray rbs
调用位置	
输入参数	无
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>tscom_flexray_rbs_start</code> ()



183. `tscom_flexray_rbs_stop`

函数名称	<code>tscom_flexray_rbs_stop()</code>
功能介绍	停止 flexray rbs
调用位置	
输入参数	无
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>tscom_flexray_rbs_stop()</code>

184. `tscom_flexray_rbs_is_running`

函数名称	<code>tscom_flexray_rbs_is_running()</code>
功能介绍	flexray rbs 是否启动
调用位置	
输入参数	无
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>tscom_flexray_rbs_is_running()</code>

185. `tscom_flexray_rbs_configure`

函数名称	<code>tscom_flexray_rbs_configure(AAAutoStart:c_bool, AAutoSendOnModification:c_bool, AActivateECUSimulation:c_bool, AInitValueOptions:c_int)</code>
功能介绍	flexray rbs 设置
调用位置	
输入参数	参数 1:AAutoStart 自动开启 参数 2:AAutoSendOnModification AutoSendOnModification 参数 3:AActivateECUSimulation ActivateECUSimulation 参数 4:AINitValueOptions InitValueOptions
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>AINitValueOptions=c_int(0)</code> <code>tscom_flexray_rbs_configure(False, False, False, AInitValueOptions)</code>

186. `tscom_flexray_rbs_activate_all_clusters`

函数名称	<code>tscom_flexray_rbs_activate_all_clusters</code> (AEnable:c_bool, AIncludingChildren:c_bool)
功能介绍	是否激活所以 flexray rbs cluster 并包括所有子节点
调用位置	
输入参数	参数 1:AEnable 是否使能 参数 2:AIncludingChildren IncludingChildren
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>tscom_flexray_rbs_activate_all_clusters</code> (True, False)

187. `tscom_flexray_rbs_activate_cluster_by_name`

函数名称	<code>tscom_flexray_rbs_activate_cluster_by_name</code> (AIdxChn:c_int, AEnable:c_bool, AClusterName:bytes, AIncludingChildren:c_bool)
功能介绍	通过 name 激活 cluster 并是否包括子节点
调用位置	
输入参数	参数 1:AIdxChn 通道编号 参数 2:AEnable 是否使能 参数 3:AClusterName ClusterName 参数 4:AIncludingChildren IncludingChildren
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>tscom_flexray_rbs_activate_cluster_by_name</code> (0, True, b"Network 1", False)

188. `tscom_flexray_rbs_activate_ecu_by_name`

函数名称	<code>tscom_flexray_rbs_activate_ecu_by_name</code> (AIdxChn:c_int, AEnable:c_bool, AClusterName:bytes, AECUName:bytes, AIncludingChildren:c_bool)
功能介绍	通过 name 激活 ecu 并是否包括子节点
调用位置	
输入参数	参数 1:AIdxChn 通道编号 参数 2:AEnable 是否使能 参数 3:AClusterName ClusterName 参数 4:AECUName ECUName 参数 5:AIncludingChildren IncludingChildren
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>tscom_flexray_rbs_activate_ecu_by_name(0, True, b"Network1", b"ECU1", False)</code>

189. `tscom_flexray_rbs_activate_frame_by_name`

函数名称	<code>tscom_flexray_rbs_activate_frame_by_name</code> (AIdxChn:c_int, AEnable:c_bool, AClusterName:bytes, AECUName:bytes, AFrameName:bytes)
功能介绍	通过 name 激活 msg
调用位置	
输入参数	参数 1:AIdxChn 通道编号 参数 2:AEnable 是否使能 参数 3:AClusterName ClusterName 参数 4:AECUName ECUName 参数 5:AFrameName FrameName
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>tscom_flexray_rbs_activate_frame_by_name(0, True, b"Network1", b"ECU1", b'Frame1')</code>

190. `tscom_flexray_rbs_get_signal_value_by_element`

函数名称	<code>tscom_flexray_rbs_get_signal_value_by_element</code> (AIdxChn:c_int 32, AClusterName:bytes, AECUName:bytes, AFrameName:bytes, ASignalName:bytes)
功能介绍	通过 element 获取信号值
调用位置	
输入参数	参数 1:AIdxChn 通道编号 参数 2:AEnable 是否使能 参数 3:AClusterName ClusterName 参数 4:AECUName ECUName 参数 5:AFrameName FrameName
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>tscom_flexray_rbs_get_signal_value_by_element</code> (0, b'PowerTrain', b'BSC', b'BackLightInfo', b'BrakeLight')

191. `tscom_flexray_rbs_get_signal_value_by_address`

函数名称	<code>tscom_flexray_rbs_get_signal_value_by_address</code> (AAddr:bytes)
功能介绍	通过地址获取信号值
调用位置	
输入参数	参数:AAddr Addr
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>tscom_flexray_rbs_get_signal_value_by_address</code> (b'0/PowerTrain/BSC/BackLightInfo/BrakeLight')

192. `tscom_flexray_rbs_set_signal_value_by_element`

函数名称	<code>tscom_flexray_rbs_set_signal_value_by_element</code> (AIdxChn:c_int 32, AClusterName:bytes, AECUName:bytes, AFrameName:bytes, ASignalName:bytes, AValue:c_double)
功能介绍	通过 element 设置信号值
调用位置	
输入参数	参数 1:AIdxChn 应用程序通道 参数 2:AClusterName ClusterName 参数 3:AECUName ECUName 参数 4:AFrameName FrameName 参数 5:ASignalName SignalName 参数 6:AValue 信号值
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>value = c_double(1.0)</code> <code>tscom_flexray_rbs_set_signal_value_by_element(0, b'PowerTrain', b'BSC', b'BackLightInfo', b'BrakeLight', value)</code>

193. `tscom_flexray_rbs_set_signal_value_by_address`

函数名称	<code>tscom_flexray_rbs_set_signal_value_by_address</code> (AAddr:bytes, AValue:c_double)
功能介绍	通过地址设置信号值
调用位置	
输入参数	参数 1:AAddr Addr 参数 2:AValue 信号值
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>value = c_double(1.0)</code> <code>tscom_flexray_rbs_set_signal_value_by_address(0, b'PowerTrain', b'BSC', b'BackLightInfo', b'BrakeLight', value)</code>

194. `tscom_flexray_rbs_enable`

函数名称	<code>tscom_flexray_rbs_enable(AEnable:c_bool)</code>
功能介绍	是否使能 flexray rbs 功能
调用位置	
输入参数	参数:AEnable 是否使能
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>tscom_flexray_rbs_enable(True)</code> 或 <code>tscom_flexray_rbs_enable(False)</code>

195. `tscom_flexray_rbs_batch_set_start`

函数名称	<code>tscom_flexray_rbs_batch_set_start()</code>
功能介绍	开始信号改值批处理
调用位置	
输入参数	无
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>tscom_flexray_rbs_batch_set_start()</code>

196. `tscom_flexray_rbs_batch_set_end`

函数名称	<code>tscom_flexray_rbs_batch_set_end()</code>
功能介绍	结束信号改值批处理
调用位置	
输入参数	无
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>tscom_flexray_rbs_batch_set_end()</code>

197. `tscom_flexray_rbs_batch_set_signal`

函数名称	<code>tscom_flexray_rbs_batch_set_signal</code> (AAddr:bytes, AValue:c_double)
功能介绍	设置信号值
调用位置	
输入参数	参数 1:AAddr Addr 参数 2:AValue 信号值
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>tscom_flexray_rbs_batch_set_signal(b"0/cluster1/ecu1/frame1/sgn1", c_double(1.2))</code>

198. `tscom_flexray_rbs_set_frame_direction`

函数名称	<code>tscom_flexray_rbs_set_frame_direction</code> (AIdxChn:c_int32, AIsTx:c_bool, AClusterName:bytes, AECUName:bytes, AFrameName:bytes)
功能介绍	设置 frame 为 tx 或 rx
调用位置	
输入参数	参数 1:AIdxChn 应用程序通道 参数 2:AIsTx 是否发送 参数 3:AClusterName ClusterName 参数 4:AECUName ECUName 参数 5:AFrameName FrameName
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>tscom_flexray_rbs_set_frame_direction(CHANNEL_INDEX.CH1, True, b"Cluster1", b"ECU1", b"Frame1")</code>

199. `tscom_flexray_rbs_set_normal_signal`

函数名称	<code>tscom_flexray_rbs_set_normal_signal</code> (ASymbolAddress:bytes)
功能介绍	设置信号为 normal 信号
调用位置	
输入参数	参数:ASymbolAddress SymbolAddress
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>tscom_flexray_rbs_set_normal_signal(b"0/Cluster1/ecu1/frame1/signal1")</code>

200. `tscom_flexray_rbs_set_rc_signal`

函数名称	<code>tscom_flexray_rbs_set_rc_signal</code> (ASymbolAddress:bytes)
功能介绍	设置信号为 rc 信号
调用位置	
输入参数	参数:ASymbolAddress SymbolAddress
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>tscom_flexray_rbs_set_rc_signal(b"0/Cluster1/ecu1/frame1/signal1")</code>

201. `tscom_flexray_rbs_set_rc_signal_with_limit`

函数名称	<code>tscom_flexray_rbs_set_rc_signal_with_limit</code> (ASymbolAddress:bytes, ALowerLimit:c_int32, AUpperLimit:c_int32)
功能介绍	设置 rc 信号值的限定范围
调用位置	
输入参数	参数 1:ASymbolAddress SymbolAddress 参数 2:ALowerLimit LowerLimit 参数 3:AUpperLimit UpperLimit
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	<code>ALowerLimit=c_int32(0)</code> <code>AUpperLimit=c_int32(14)</code> <code>tscom_flexray_rbs_set_rc_signal_with_limit(b"0/Cluster1/ecu1/frame1/signal1", ALowerLimit, AUpperLimit)</code>



202. `tscom_flexray_rbs_set_crc_signal`

函数名称	<code>tscom_flexray_rbs_set_crc_signal</code> (ASymbolAddress:bytes, AAlgorithmName:bytes, AIdxByteStart:c_int32, AByteCount:c_int32)
功能介绍	设置信号为 crc 信号
调用位置	
输入参数	参数 1:ASymbolAddress SymbolAddress 参数 2:AAlgorithmName AlgorithmName 参数 3:AIdxByteStart IdxByteStart 参数 4:AByteCount ByteCount
返回值	==0:设置成功 其他:失败, 根据错误码查看错误类型
示例	AAlgorithmName=b"mp. crc8" AIdxByteStart=c_int32(0) AByteCount=c_int32(2) <code>tscom_flexray_rbs_set_crc_signal</code> (b"0/Cluster1/ecul/frame1/signall", b"mp. crc8", c_int32(0), c_int32(2))