TOSUN同星

# TSMaster_COM API_Python 编程指导

版本：V1.0 | 中文

tosunai.com

文档修订历史：

| 文件版本 | 日期 | 更新内容 | 备注 |
| --- | --- | --- | --- |
| **V1.00** | 2023.11.30 | 创建文档 | |
| | | | |
| | | | |

# TSMaster_COM API_Python 编程指导

目录

**说明：**

- ✓ *第一页根据实际情况修改*
- ✓ *最后一页保持不动*
- ✓ *所有标题用的中文字体为黑体，段落无缩进，1.5 倍行距*
- ✓ *正文用的中文字体为宋体，大小为五号，段落首行缩进 2 字符，1.25 倍行距*
- ✓ *所有英文字体是 Times New Roman*
- ✓ *需要注意的是，每个标题只要在菜单选择了相应的格式，插入目录的时候就会自动关联并超链接*
- ✓ *页眉不用动，TOSUN 和 TSMaster 的 logo 是奇偶页轮换的，奇数页页眉是固定文字，偶数页页眉自动关联文档的总标题这个文字的*
- ✓ *页脚不用动，从目录页之后，也就是具体内容部分开始统一版式，并作为第一页开始显示页码*

***重要提示***，**为了保证整体模板不变，从其他文档复制信息过来粘贴的时候，建议点鼠标右键—>选择性粘贴—>无格式文本**

# 1. 什么情况下需要此文档

用户基于 Python 平台编程语言，对目前市面上主流的如 Vector、TOSUN、Peak、英特佩斯等工具进行二次开发的时候，需要参考本文档，调用 com 接口来实现对设备的程序控制。

# 2. 使用 SDK 工程

安装 TSMaster 文件后，在 SDK 的目录下面，提供了 C/C++,C#，LabView 等的示例工程。以本电脑为例，COM 接口的 Demo 工程目录如下所示：C:\Program Files (x86)\TOSUN\TSMaster\bin\Data\SDK\examples\COM Automation\OutOfProcess\

进入目录后，目录包含文件如下所示：


## 2.1 错误提示：

## 2.2 解决方法：

# 3. 驱动特点

本驱动提供了硬件抽象层，通过映射机制， 同样一套 API 函数，可以应用于多款市面上主流的 CAN 工具硬件而不用修改代码。其作用原理图如下所示：

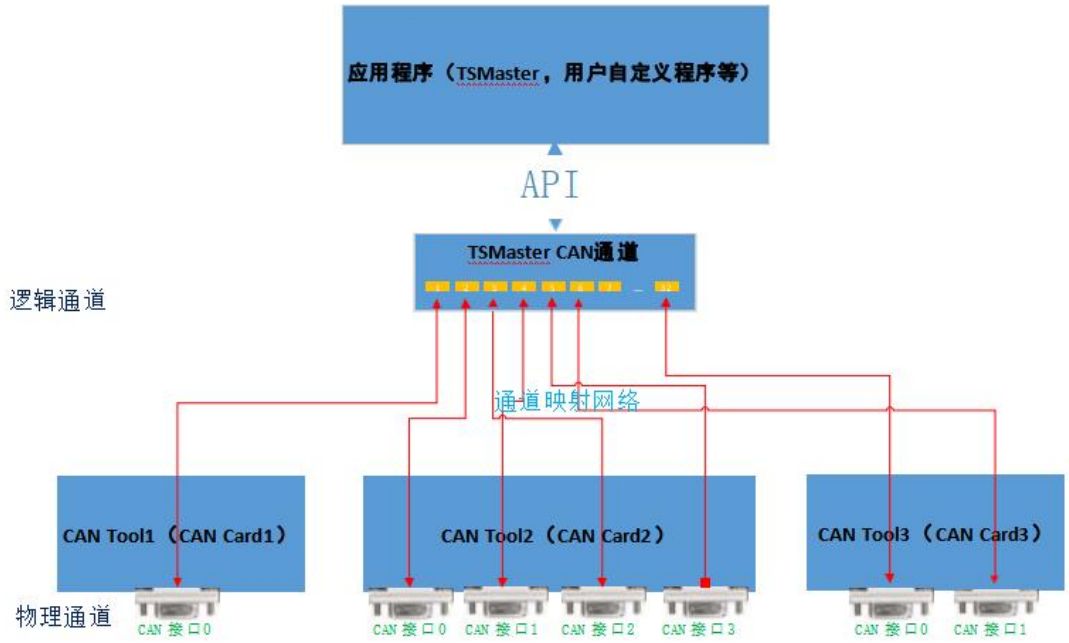



图 1. TSMasterAPI 通过逻辑通道给上层提供统一的 API 接口

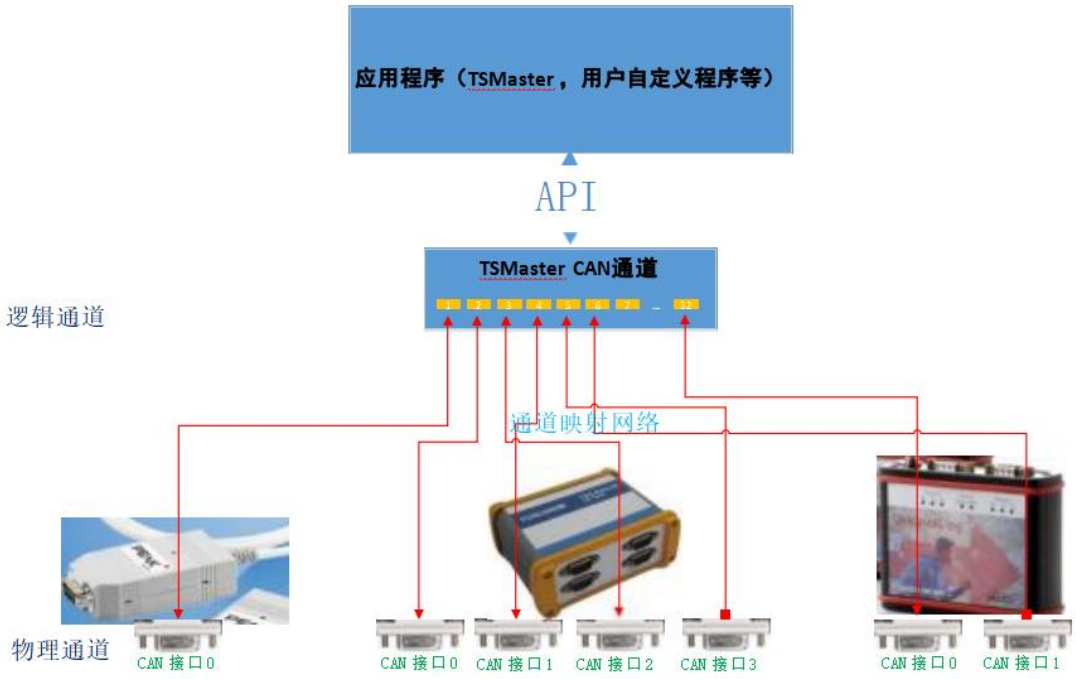



图 2. 基于 TSMasterAPI 混合连接不同硬件板卡示意图

如上图所示，TSMaster API 内部实现了对目前市面上主流硬件 CAN 卡的兼容，提供了统一的 com 接口给上层应用层。用户在开发上层的时候，只需要开发一套上层代码， 底层硬件可以任意切换，为用户硬件选择提供了最大的灵活性。

# 4. 添加 API 库文件

## 4.1 安装驱动运行环境 TSMaster

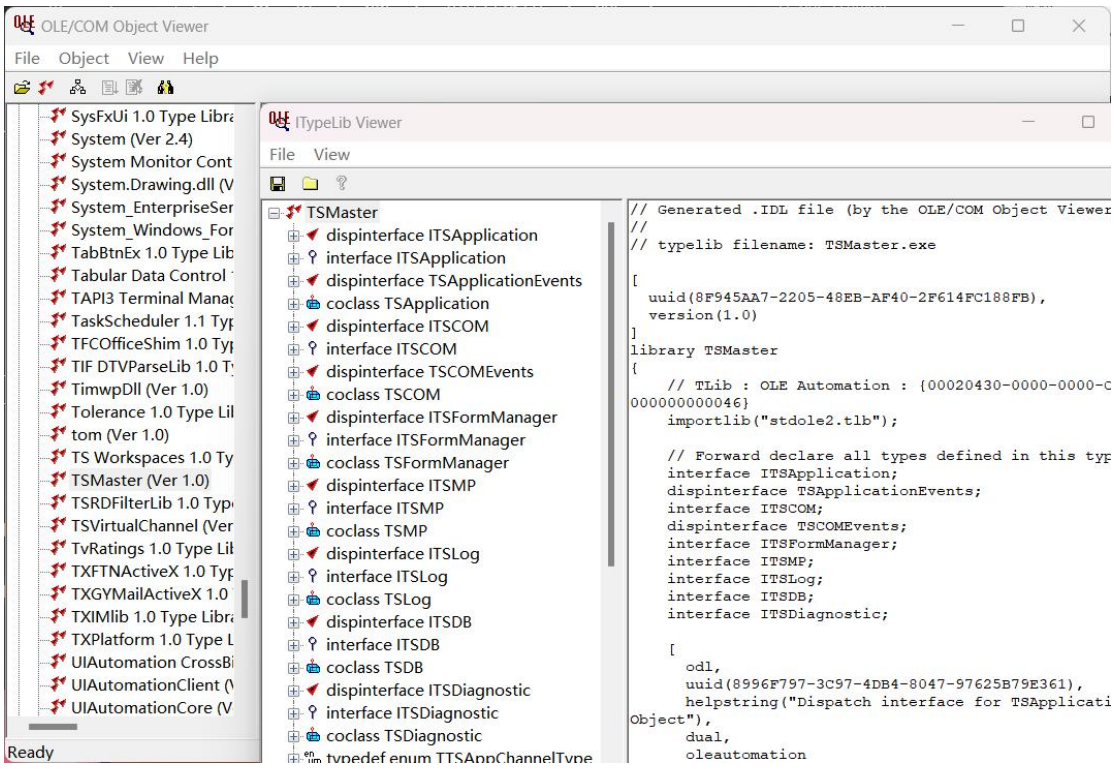
TSMaster API 提供了多种工具平台的支持，因此要使用 TSMaster 驱动，首先要安装 TSMaster 运行环境，该运行环境大小 200 兆左右，安装文件如下：


TSMaster 运行时环境为免费软件，可以直接到上海同星公司的官网下载，下载连接为：http://www.tosun.tech/TOSUNSoftware/TSMaster_Setup.exe

下载该文件，并安装，进入下一步。

## 4.2 添加库文件引用

安装完运行环境后，便有了运行时环境的支撑，如下图所示：




运行 oleview.exe 打开 Type Libraries，找到 TSMaster，其中便是 TSMaster 提供的 COM 接口。

# 5. 数据类型定义

## 5.1 TLIBCAN：Classic CAN 数据类型

**Struct 视图：**

```
typedef [uuid(19CEBEE3-37B4-4092-AC82-7D124EE2352C)]
struct tagTCAN {

long FIdxChn;

VARIANT_BOOL FIsTX;

VARIANT_BOOL FIsRemote;

VARIANT_BOOL FIsExtendedId;

VARIANT_BOOL FIsError;

long FDLC;

long FIdentifier;

int64 FTimeUs;

SAFEARRAY(char) FDatas;
} TCAN;
```

**构造函数 1：**

用户可灵活调用。在构造函数中没有传入的参数， 可以通过访问结构体对象直接修改。比如构造函数 1 没有传输数据组，但是可以在创建结构体对象过后，直接给 FData 成员赋值。

**成员：**

FIdxChn：应用程序通道，注意 CHANNEL_INDEX. CHN1 = 0,实际上是从 0 开始计算的。

FIsTX：是否为发送端

FIsRemote：是否远程帧

FIsExtendedId：是否为扩展帧

FIsError：是否为错误帧

FDLC：帧长度

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

FTimeUS：帧时间戳， 64 位 us 级时间戳。

FData：帧数据。最大程度为 8Bytes

**示例：**

```
# init a TCAN record for CAN message transmission
c = win32com.client.Record( name: "TCAN", app)

c.FIdxChn = 0
c.FIsTX=0
c.FIsRemote = False
c.FIsExtendedId = 0
c.FIsError = False
c.FDLC = 8
c.FIdentifier = 0x123
c.FTimeUs=50
c.FDatas = VARIANT(pythoncom.VT_ARRAY | pythoncom.VT_I1,  value: [1, 2, 3, 4, 5, 6,
7, 8])
```

## 5.2 TLIBCANFD：Classic CANFD 数据类型

**Struct 视图：**

```
typedef [uuid(F8CC7BEE-63F0-4FD2-AFB1-B1C313BECE43)]
struct tagTCANFD {

long FIdxChn;

VARIANT_BOOL FIsTX;

VARIANT_BOOL FIsExtendedId;

VARIANT_BOOL FIsError;

VARIANT_BOOL FIsEDL;

VARIANT_BOOL FIsBRS;

VARIANT_BOOL FIsESI;

long FDLC;

long FIdentifier;

int64 FTimeUs;

SAFEARRAY(char) FDatas;
} TCANFD;
```

**构造函数 2：**

用户可灵活调用。在构造函数中没有传入的参数， 可以通过访问结构体对象直接修改。比如构造函数 2 没有传输数据组，但是可以在创建结构体对象过后，直接给 FData 成员赋值。

**成员：**

FIdxChn：应用程序通道，注意 CHANNEL_INDEX. CHN1 = 0,实际上是从 0 开始计算的。

FIsTX：是否为发送端

FIsRemote：是否远程帧

FIsExtendedId：是否为扩展帧

FIsError：是否为错误帧

FIsEDL：0 表示正常 can 帧，1 表示 FD 帧

FIsBRS：位速率转换，0 表示转换可变速率

FIsESI：错误状态指示位

FDLC：帧长度

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

FTimeUS：帧时间戳， 64 位 us 级时间戳

FData：帧数据。最大程度为 8Bytes

**示例：**

```
# init a TCANFD record for CAN FD message transmission
cFD = win32com.client.Record( name: "TCANFD", app)

cFD.FIdxChn = 0
cFD.FIsTx=1
cFD.FIsRemote=0
cFD.FIsExtendedId = 0
cFD.FIsError=0
cFD.FIsEDL = True
cFD.FIsBRS = True
cFD.FIsESI = False
cFD.FDLC = 15
cFD.FIdentifier = 0x345
cFD.FTimeUS=50
cFD.FDatas = VARIANT(pythoncom.VT_ARRAY | pythoncom.VT_I1,
                     value: [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, 33, 34, 35, 36, 37, 38,
                             39, 40, 41, 42, 43, 44,45, 46, 47, 48, 49, 50,51, 52, 53, 54, 55, 56, 57,
                             58,59, 60, 61, 62, 63, 64])
```

## 5.3 TLIBLIN

**Struct 视图：**

```
typedef [uuid(F5C26CE2-3629-4BD0-A7F7-C2FAF8999736)]
struct tagTLIN {

long FIdxChn;

VARIANT_BOOL FIsTX;

VARIANT_BOOL FIsError;

long FDLC;

long FIdentifier;

long FChecksum;

int64 FTimeUs;

SAFEARRAY(char) FDatas;
} TLIN;
```

**构造函数 3：**

用户可灵活调用。在构造函数中没有传入的参数， 可以通过访问结构体对象直接修改。比如构造函数 3 没有传输数据组，但是可以在创建结构体对象过后，直接给 FData 成员赋值。

**成员：**

FIdxChn：帧通道，注意 CHANNEL_INDEX. CHN1 = 0，实际上是从 0 开始计算的。

FIsTX：是否为发送端

FIsError：是否为错误帧

FDLC：帧长度

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

FChecksum：checksum 校验和

FTimeUS：帧时间戳， 64 位 us 级时间戳。

FData：帧数据。最大程度为 8Bytes

**示例：**

```
cLIN = win32com.client.Record( name: "TLIN", app)
cLIN.FIdxChn = 0
cLIN.FIsTX = True
cLIN.FIsError = False
cLIN.FIdentifier = 0x3C
cLIN.Checksum = 0xE6
cLIN.FDLC = 8
cLIN.TimeUs= 50
cLIN.FDatas = VARIANT(pythoncom.VT_ARRAY | pythoncom.VT_I1,  value: [1, 2, 3, 4, 5, 6, 7, 8,])
```

## 5.4 TTSMapping

**Struct 视图：**

```
typedef [uuid(6EE71046-BB83-47A9-BC09-84A1B43C0699)]
struct tagTTSMapping {

BSTR FAppName;

long FAppChannelIndex;

TTSAppChannelType FAppChannelType;

TTSBusToolDeviceType FHWDeviceType;

long FHWIndex;

long FHWChannelIndex;

long FHWDeviceSubType;

BSTR FHWDeviceName;

VARIANT_BOOL FMappingDisabled;
} TTSMapping;
```

**成员：**

FAppName：应用程序名称

FAppChannelIndex：应用程序通道

FAppChannelType：通道类型

FHWDeviceType：硬件类型

FHWIndex：硬件序号

FHWChannelIndex：硬件通道序号

FHWDeviceSubType：硬件型号的具体类型

FHWDeviceName：硬件设备名称

FMappingDisabled：映射使能

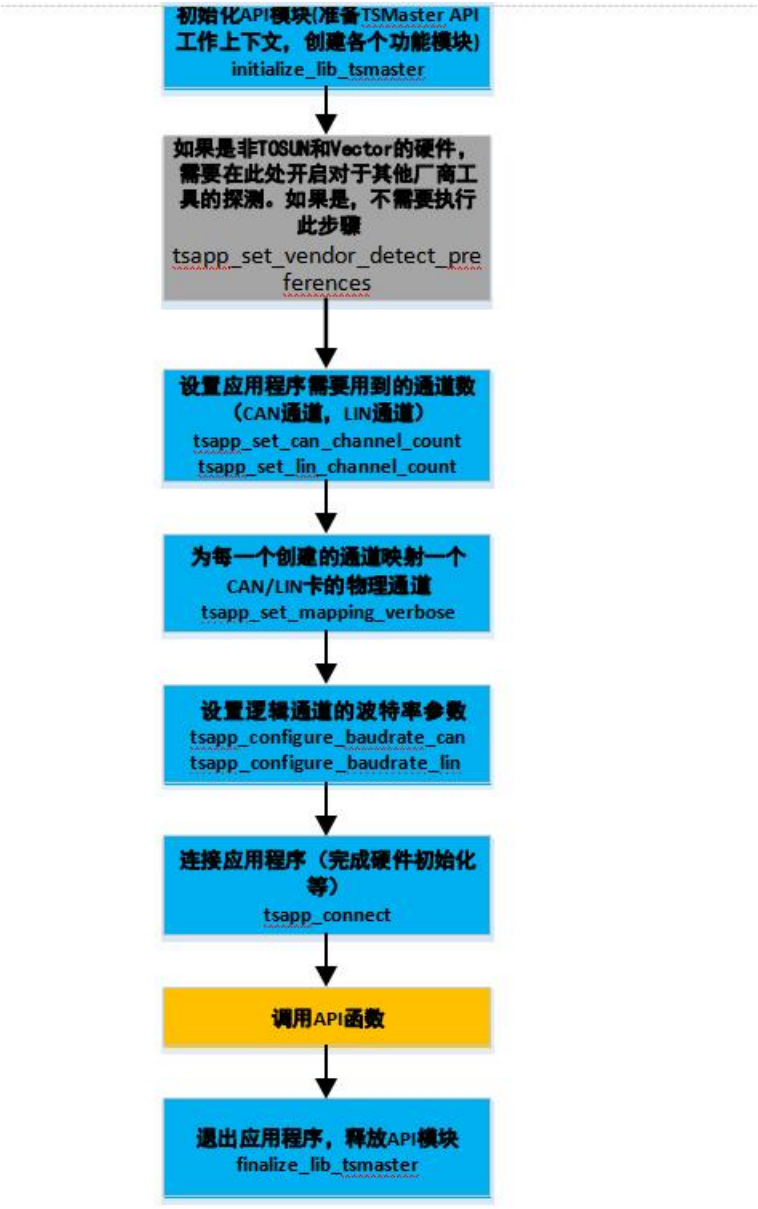
**示例：**

```
r = win32com.client.Record( name: "TTSMapping",app)
r.FAppName = comTSMaster
r.FAppChannelType=0
r.FAppChannelIndex = 0
r.FHWDeviceType = 3
r.FHWIndex = 0
r.FHWChannelIndex = 0
r.FHWDeviceSubType = 8  #为TC1014编号
r.FHWDeviceName = "TC1014"
r.FMappingDisabled = False
```

# 6. 设备初始化

## 6.1CAN 初始化

在实现 CAN 设备通讯之前，首先要进行设备初始化。 TSMaster 设备初始化包括以下步骤：




**注意：**

1、在使用 API 模块之前，需要调用 initialize_lib_tsmaster 函数创建该模块；在 使用过后， 需要调用 finalize_lib_tsmaster 释放该模块。这两个函数一定是成对出现的。

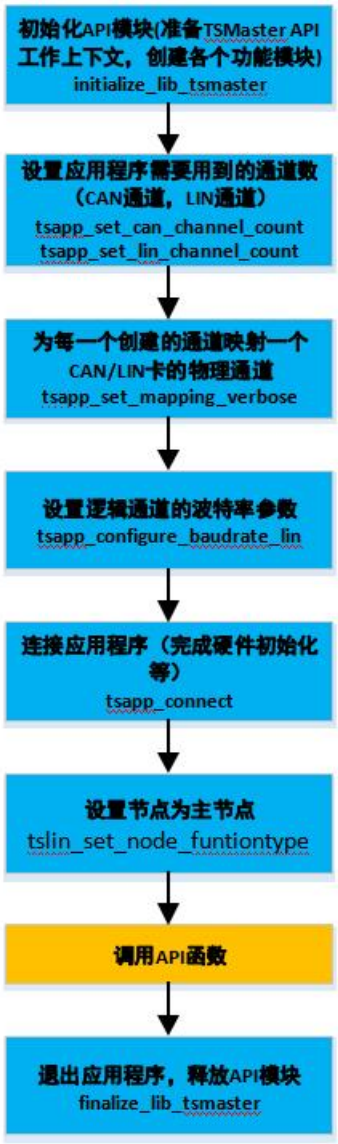
2、第二步中， 如果用户使用的是英特佩斯， Peak，Kavsar，ZLG 等硬件， 需要在这一步开启对这些工具的探测， 否则程序无法查询到这些硬件。

**CAN 初始化示例：**

```
pythoncom.CoInitialize() # enable multithread
#打开 TSMaster
app = win32com.client.Dispatch("TSMaster.TSApplication")
com = app.TSCOM() #
win32com.client.Dispatch("TSMaster.TSCOM")
formMan = app.TSFormManager()
#设置为主窗口
formMan.show_main_form()
app.set_can_channel_count(2) #设置 can 通道为 2
app.set_lin_channel_count(0) #设置 lin 通道为 0
r = win32com.client.Record("TTSMapping", app)
r.FAppName = APP_NAME
r.FAppChannelIndex = 0
r.FAppChannelType = 0
r.FHWIndex = 0
r.FHWDeviceType = 3
r.FHWDeviceSubType = 8   #为 TC1014 编号
r.FHWChannelIndex = 0
r.FHWDeviceName = "TC1014"
r.FMappingDisabled = False
app.set_mapping(r)
r.FAppChannelIndex = 1
r.FHWChannelIndex = 1
app.set_mapping(r)
app.configure_baudrate_canfd(0, 500, 2000,constants.lfdtISOCAN, constants.lfdmNormal,
True)
app.configure_baudrate_canfd(1, 500, 2000,constants.lfdtISOCAN, constants.lfdmNormal,
True)
app.connect()
```

**#关于硬件映射编号请打开 Tsmaster 帮助栏的软件开发 TSMasterAPI_Hardware_Map 文档查找。**

## 6.2LIN 初始化




**LIN 初始化示例：**

```
pythoncom.CoInitialize() # enable multithread
#打开 TSMaster
app = win32com.client.Dispatch("TSMaster.TSApplication")
com = app.TSCOM()    #win32com.client.Dispatch("TSMaster.TSCOM")
formMan = app.TSFormManager()
#设置为主窗口
formMan.show_main_form()
app.set_can_channel_count(1) #设置 can 通道为 2
app.set_lin_channel_count(2) #设置 lin 通道为 0
r = win32com.client.Record("TTSMapping", app)
```

```
r.FAppName = APP_NAME
r.FAppChannelIndex = 0
r.FAppChannelType = 1
r.FHWIndex = 0
r.FHWDeviceType = 3
r.FHWDeviceSubType = 10    #为 TC1026 编号
r.FHWChannelIndex = 0
r.FHWDeviceName = "TC1026"
r.FMappingDisabled = False
app.set_mapping(r)
r.FAppChannelIndex = 1
r.FHWChannelIndex = 1
app.set_mapping(r)
app.configure_baudrate_lin(0, 19.2, 3)
app.connect()
```

# 7. CAN 报文收发

## 7.1 报文发送

CAN 报文发送对应的代码如下：

```
app = win32com.client.Dispatch("TSMaster.TSApplication")
com = app.TSCOM()    #win32com.client.Dispatch("TSMaster.TSCOM")
c = win32com.client.Record("TCAN", app)
c.FIdxChn = 0
c.FIsTX=0
c.FIsRemote = False
c.FIsExtendedId = 0
c.FDLC = 8
c.FIdentifier = 0x123
c.FTimeUs= 50
c.FDatas = VARIANT(pythoncom.VT_ARRAY | pythoncom.VT_I1, [1, 2, 3, 4, 5, 6, 7, 8])
com.transmit_can_async(c)
```

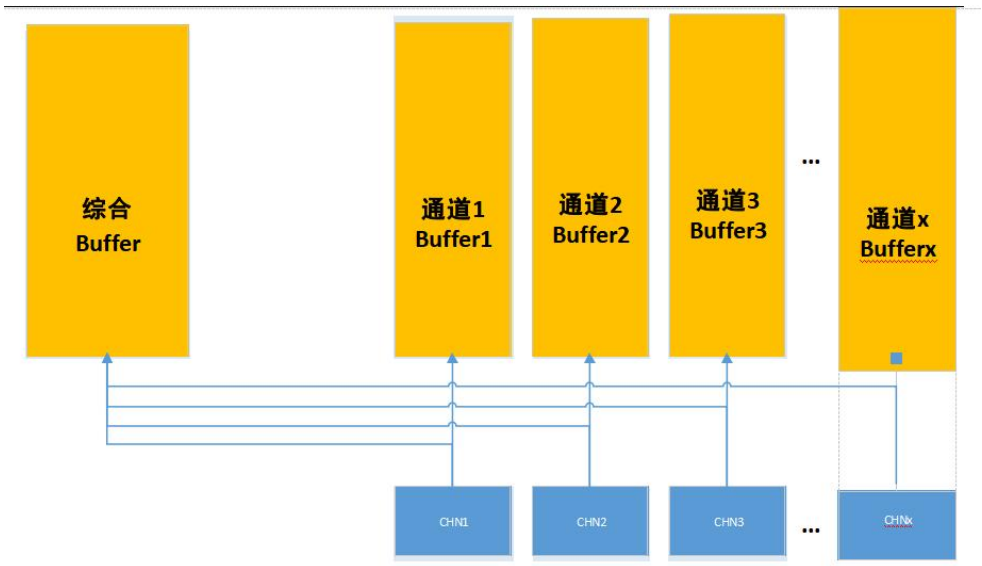
CANFD 报文发送对应的代码如下：

```
app = win32com.client.Dispatch("TSMaster.TSApplication")
com = app.TSCOM()    #win32com.client.Dispatch("TSMaster.TSCOM")
cFD = win32com.client.Record("TCANFD", app)
cFD.FIdxChn = 0
cFD.FIsTx=1
cFD.FIsRemote=0
cFD.FIsExtendedId = 0
cFD.FIsEDL = True
cFD.FIsBRS = True
cFD.FIsESI = False
cFD.FIdentifier = 0x345
cFD.FDLC = 15
cFD.FTimeUs=50
cFD.FDatas = VARIANT(pythoncom.VT_ARRAY | pythoncom.VT_I1, [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, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64])
# transmit this CAN FD frame asynchrnously
com.transmit_canfd_async(cFD)
```

## 7.2 报文接收

**读取设备消息缓存的方式：**

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。 TSMaster API 内部， 报文缓存机制如下图所示：




**综合 FIFO**：所有通道的报文根据接收顺序放在里面。

特点：可以看到不同通道报文的相对接收顺序。报文在里面按照接收顺序存放，最新的报文覆盖最旧接收的报文。

**通道 FIFO**：每一个通道有一个自己单独的报文 FIFO。

特点：专用于存储跟本通道相关的报文，通道之间互不干扰。报文在里面按照接收顺序存放，最新的报文覆盖最旧接收的报文。

TSMaster 提供了报文读取驱动。可以选择读取指定通道的报文集合， 也可以读取综合报文里面的报文集合。

**fifo_receive_can_msg**

功能：读取 Classic CAN 内部报文。

参数详解：

AIdxChnReq:请求数据通道

AIncludeTx：是否接收 TX

AIdxChnResp：回复数据通道

AIsRemote：是否是远程帧

AIsExtended：是否是扩展帧

ADLC：数据长度

AIdentifier：帧 ID

ATimestampUs：硬件时间

ADatas：帧数据

```
VARIANT_BOOL fifo_receive_can_msg(
                [in] long AIdxChnReq,
                [in] VARIANT_BOOL AIncludeTx,
                [out] long* AIdxChnResp,
                [out] VARIANT_BOOL* AIsRemote,
                [out] VARIANT_BOOL* AIsExtended,
                [out] long* ADLC,
                [out] long* AIdentifier,
                [out] int64* ATimestampUs,
                [out] BSTR* ADatas);
```

**fifo_receive_canfd_msg**

功能：读取 CANFD 内部报文。

参数详解：

AIdxChnReq:请求数据通道

AIncludeTx：是否接收 TX

AIdxChnResp：回复数据通道

AIsRemote：是否是远程帧

AIsExtended：是否是扩展帧

AIsEDL：是否为 can

AIsBRS：速率是否可以变

ADLC：数据长度

AIdentifier：帧 ID

ATimestampUs：硬件时间

ADatas：帧数据

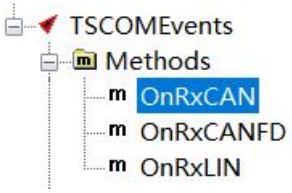
```
VARIANT_BOOL fifo_receive_canfd_msg(
                [in] long AIdxChnReq,
                [in] VARIANT_BOOL AIncludeTx,
                [out] long* AIdxChnResp,
                [out] VARIANT_BOOL* AIsRemote,
                [out] VARIANT_BOOL* AIsExtended,
                [out] VARIANT_BOOL* AIsEDL,
                [out] VARIANT_BOOL* AIsBRS,
                [out] long* ADLC,
                [out] long* AIdentifier,
                [out] int64* ATimestampUs,
                [out] BSTR* ADatas);
```

**回调函数方式：**

设备接收到报文过后，把报文整理成标准的 TLibCAN/TLibANFD 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（C#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：




**回调函数使用：**

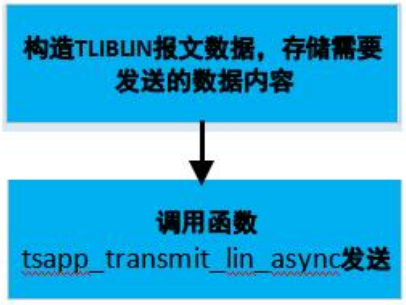
```
class TSMasterEvents:
    def OnRxCAN(self, AIdxChn, AIsTx, AIsRemote, AIsExtended, AIsError, ADLC, AIdentifier, ATimeUs, ADatas):
        print("CAN message received:", AIdxChn, AIdentifier, ADLC, ATimeUs / 1000000.0,
              [ADatas[0], ADatas[1], ADatas[2], ADatas[3], ADatas[4], ADatas[5], ADatas[6], ADatas[7]])

    def OnRxCANFD(self, AIdxChn, AIsTx, AIsExtended, AIsError, AIsEDL, AIsBRS, AIsESI, ADLC, AIdentifier, ATimeUs,ADatas
        if AIsEDL:
            print("CANFD message received:", AIdxChn, AIdentifier, ADLC, ATimeUs / 1000000.0,
                  [ADatas[0], ADatas[1], ADatas[2], ADatas[3], ADatas[4], ADatas[5], ADatas[6], ADatas[7],
                   ADatas[8], ADatas[9], ADatas[10], ADatas[11], ADatas[12], ADatas[13], ADatas[14], ADatas[15],
                   ADatas[16], ADatas[17], ADatas[18], ADatas[19], ADatas[20], ADatas[21], ADatas[22], ADatas[23],
                   ADatas[24], ADatas[25], ADatas[26], ADatas[27], ADatas[28], ADatas[29], ADatas[30], ADatas[31],
                   ADatas[32], ADatas[33], ADatas[34], ADatas[35], ADatas[36], ADatas[37], ADatas[38], ADatas[39],
                   ADatas[40], ADatas[41], ADatas[42], ADatas[43], ADatas[44], ADatas[45], ADatas[46], ADatas[47],
                   ADatas[48], ADatas[49], ADatas[50], ADatas[51], ADatas[52], ADatas[53], ADatas[54], ADatas[55],
                   ADatas[56], ADatas[57], ADatas[58], ADatas[59], ADatas[60], ADatas[61], ADatas[62], ADatas[63]])
```

每当设备发送/收到一帧报文， 就会调用此函数。参数：ref TLibCAN AData 就是对应的这一帧报文。用户可以根据报文的属性来决定如何处理这一帧报文。数据结构定义见章节：数据类型的定义

# 8. LIN 报文收发

## 8.1 报文发送

报文发送流程如下所示：

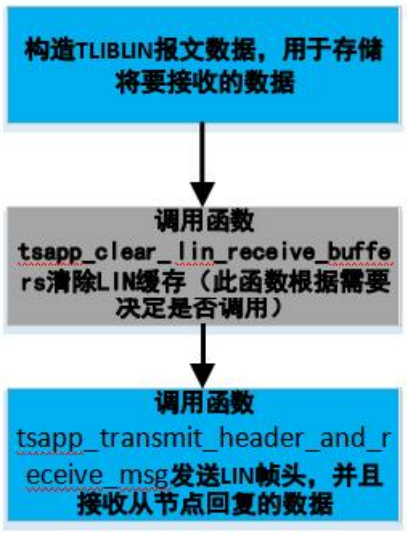


对应的代码如下：

```
app = win32com.client.Dispatch("TSMaster.TSApplication")
com = app.TSCOM()    #win32com.client.Dispatch("TSMaster.TSCOM")
cLIN = win32com.client.Record("TLIN", app)
cLIN.FIdxChn = 0
cLIN.FIsTX = True
cLIN.FIsError = False
cLIN.FIdentifier = 0x3C
cLIN.Checksum= 0xE6
cLIN.FDLC = 8
cLIN.TimeUs = 50
cLIN.FDatas = VARIANT(pythoncom.VT_ARRAY | pythoncom.VT_I1, [1, 2, 3, 4, 5, 6, 7, 8, ])
com.transmit_lin_async(cLIN)
```

## 8.2 报文接收

报文接收流程如下所示:




参数说明：

AIdxChnReq:请求数据通道

AIncludeTx：是否接收 TX

AIdxChnResp：回复数据通道

ADLC：数据长度

AIdentifier：帧 ID

ATimestampUs：硬件时间

ADatas：帧数据

```
VARIANT_BOOL fifo_receive_lin_msg(
                    [in] long AIdxChnReq,
                    [in] VARIANT_BOOL AIncludeTx,
                    [out] long* AIdxChnResp,
                    [out] long* ADLC,
                    [out] long* AIdentifier,
                    [out] int64* ATimestampUs,
                    [out] BSTR* ADatas);
```

# 9. 数据库（DBC）解析

# 10. 接口函数介绍

## 10.1 set_current_application

| | |
|---|---|
| 函数名称 | void set_current_application([in] BSTR AAppName); |
| 功能介绍 | 设置当前应用程序名 |
| 调用位置 | 软件连接之前 |
| 输入参数 | AAppName |
| 返回值 | 无 |
| 示例 | AAppName= "TSMaster"<br>set_current_application(AAppName) |

## 10.2 del_application

| | |
|---|---|
| 函数名称 | void del_application([in] BSTR AAppName) |
| 功能介绍 | 删除应用程序 |
| 调用位置 | 软件连接之前 |
| 输入参数 | AAppName |
| 返回值 | 无 |
| 示例 | AAppName= "TSMaster"<br>del_application(AAppName) |

## 10.3 add_application

| | |
|---|---|
| 函数名称 | void add_application([in] BSTR AAppName) |
| 功能介绍 | 添加应用程序 |
| 调用位置 | TSMaster 连接之前 |
| 输入参数 | AAppName |
| 返回值 | 无 |
| 示例 | AAppName= "TSMaster"<br>add_application(AAppName) |

## 10.4 get_application_list

| | |
|---|---|
| 函数名称 | BSTR get_application_list() |
| 功能介绍 | 获取所有的应用程序名 |
| 调用位置 | TSMaster 连接之前 |
| 输入参数 | 无 |
| 返回值 | 无 |
| 示例 | get_application_list() |

## 10.5 set_can_channel_count

| | |
|---|---|
| 函数名称 | void set_can_channel_count([in] long ACount) |
| 功能介绍 | 设置 CAN 通道数量 |
| 调用位置 | TSMaster 连接之前 |
| 输入参数 | 整形数，最大为 32 |
| 返回值 | 无 |
| 示例 | set_can_channel_count(2) |

## 10.6 set_lin_channel_count

| | |
|---|---|
| 函数名称 | void set_lin_channel_count([in] long ACount) |
| 功能介绍 | 设置 LIN 通道数量 |
| 调用位置 | TSMaster 连接之前 |
| 输入参数 | 整形数，最大为 32 |
| 返回值 | 无 |
| 示例 | set_lin_channel_count(0) |

## 10.7 get_can_channel_count

| | |
|---|---|
| 函数名称 | long get_can_channel_count() |
| 功能介绍 | 获取 CAN 通道数 |
| 调用位置 | TSMaster 连接之前 |
| 输入参数 | 无 |
| 返回值 | LIN 通道数 |
| 示例 | get_can_channel_count() |

## 10.8 get_lin_channel_count

| | |
|---|---|
| 函数名称 | long get_lin_channel_count() |
| 功能介绍 | 获取 LIN 通道数 |
| 调用位置 | TSMaster 连接之前 |
| 输入参数 | 无 |
| 返回值 | CAN 通道数 |
| 示例 | get_lin_channel_count() |

## 10.9 Log

| | |
|---|---|
| 函数名称 | void log([in] BSTR AMsg, [in] long ALevel); |
| 功能介绍 | 打印信息 |
| 调用位置 | |
| 输入参数 | AMsg 信息，ALevel |
| 返回值 | 无 |
| 示例 | log('connecting application...', constants.LVL_HINT) |

## 10.10 set_mapping

| | |
|---|---|
| 函数名称 | void set_mapping([in] TTSMapping* AMapping) |
| 功能介绍 | 通道映射 |
| 调用位置 | Tsmaster 连接之前 |
| 输入参数 | struct tagTTSMapping {<br>BSTR FAppName;<br>long FAppChannelIndex;<br>TTSAppChannelType FAppChannelType;<br>TTSBusToolDeviceType FHWDeviceType;<br>long FHWIndex;<br>long FHWChannelIndex;<br>long FHWDeviceSubType;<br>BSTR FHWDeviceName;<br>VARIANT_BOOL FMappingDisabled;<br>} TTSMapping; |
| 返回值 | 无 |
| 示例 | app = win32com.client.Dispatch("TSMaster.TSApplication")<br>r = win32com.client.Record("TTSMapping",app)<br>r.FAppName = APP_NAME<br>r.FAppChannelIndex = 0<br>r.FHWIndex = 0 |

| | |
|---|---|
| | r.FHWDeviceType = 3<br>r.FHWDeviceSubType = 8　#为 TC1014 编号<br>r.FHWChannelIndex = 0<br>r.FHWDeviceName = "TC1014"<br>r.FMappingDisabled = False<br>app.set_mapping(r) |

## 10.11 get_mapping

| | |
|---|---|
| 函数名称 | TTSMapping get_mapping(<br>[in] TTSAppChannelType AAppChannelType,<br>[in] long AAppChannelIndex); |
| 功能介绍 | 获取映射信息 |
| 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 | 参数 1：通道类型 ，参数 2：通道索引 |
| 返回值 | 映射信息， 结构体 TTSMapping |
| 示例 | get_mapping(TTSAppChannelType.APP_CAN,0) |

## 10.12 del_mapping

| | |
|---|---|
| 函数名称 | void del_mapping(　[in] TTSAppChannelType　AAppChannelType,<br>[in] long AAppChannelIndex); |
| 功能介绍 | 删除指定通道的映射信息值 |
| 调用位置 | 想查询应用程序指定通道当前绑定的硬件信息等内容的时候，调用此函数 |
| 输入参数 | AAppChannelType：通道类型<br>AAppChannelIndex：通道索引 |
| 返回值 | 无 |
| 示例 | del_mapping(TTSAppChannelType.APP_CAN,0) |

## 10.13 connect

| | |
|---|---|
| 函数名称 | void connect() |
| 功能介绍 | 完成各个硬件参数的设置过后，调用此函数完成设备的连接。后面就可以调用设备完成数据收发等功能了。 |
| 调用位置 | 连接应用程序。本函数执行的时候，会检查映射的硬件通道是否全部就绪，设置各个硬件参数，并完成设备和应用程序的连接 |
| 输入参数 | 无 |
| 返回值 | 无 |
| 示例 | connect() |

## 10.14 disconnect

| | |
|---|---|
| 函数名称 | void disconnect() |
| 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 | 无 |
| 示例 | disconnect() |

## 10.15 get_current_application

| | |
|---|---|
| 函数名称 | BSTR get_current_application() |
| 功能介绍 | 获取当前应用程序名称 |
| 调用位置 | TSMaster 连接之前 |
| 输入参数 | 无 |
| 返回值 | 无 |
| 示例 | AAppName = "TSMaster"<br>get_current_application( AAppName) |

## 10.16 get_error_description

| | |
|---|---|
| 函数名称 | BSTR get_error_description([in] long AErrorCode) |
| 功能介绍 | 获取错误描述信息 |
| 调用位置 | 调用此函数时，会返回一个 ErroCode 编码，通过调用此函数可以查询该编码代表的具体含义 |
| 输入参数 | 参数:AErrorCode 错误编码值 |
| 返回值 | 错误编码代表具体含义字符串 |
| 示例 | app = win32com.client.Dispatch("TSMasterTSApplication")<br>ret = app.connect()<br>if ret != 0:<br>app.get_error_description (ret) |

## 10.17 get_timestamp

| | |
|---|---|
| 函数名称 | int64 get_timestamp() |
| 功能介绍 | 获取时间戳信息 |
| 调用位置 | |
| 输入参数 | 整形数 |
| 返回值 | ==0:设置成功<br>其他:失败 |
| 示例 | ATimestampUs = int64(0)<br>get_timestamp(ATimestamp) |

## 10.18 configure_baudrate_can

| | |
|---|---|
| 函数名称 | void configure_baudrate_can(<br>[in] long AIdxChn,<br>[in] single ABaudrateKbps,<br>[in] VARIANT_BOOL AListenOnly,<br>[in] VARIANT_BOOL AInstallTermResistor120Ohm); |

| | |
|---|---|
| 功能介绍 | 配置 CAN 通道的波特率等硬件参数 |
| 调用位置 | 连接 CAN 工具之前， 先调用此函数配置硬件设备参数 |
| 输入参数 | "AIdxChn"：应用程序通道<br>"ABaudrateKbps"：波特率<br>"AListenOnly"：是否只听<br>"AInstallTermResistor120Ohm"：是否使能内部终端电阻 |
| 返回值 | ==0:设置成功<br>其他:失败 |
| 示例 | configure_baudrate_can(0,500,True,True) |

## 10.19 configure_baudrate_canfd

| | |
|---|---|
| 函数名称 | void configure_baudrate_canfd(<br>[in] long AIdxChn,<br>[in] single AArbRateKbps,<br>[in] single ADataRateKbps,<br>[in] TTSCANFDControllerType AControllerType,<br>[in] TTSCANFDControllerMode AControllerMode,<br>[in] VARIANT_BOOL AInstallTermResistor120Ohm); |
| 功能介绍 | 配置 CANFD 通道的波特率等硬件参数 |
| 调用位置 | 连接 CAN 工具之前， 先调用此函数配置硬件设备参数 |
| 输入参数 | "AIdxChn"：应用程序通道<br>"AArbRateKbps"：仲裁场波特率<br>"ADataRateKbps"：数据场波特率<br>"AControllerType"：控制器类型，包括：<br>CAN(lfdtCAN=0),<br>ISOCANFD(lfdtISOCAN=1),<br>NoISOCANFD(lfdtNonISOCAN = 2)<br>"AControllerMode"：控制器工作模式，包括：<br>正常工作模式(lfdmNormal = 0)<br>关闭 ACK 模式(lfdmACKOff = 1)<br>受限制模式(lfdmRestricted = 2<br>"AInstallTermResistor120Ohm"：是否使能内部终端电阻 |

| | |
|---|---|
| 返回值 | 无 |
| 示例 | configure_baudrate_canfd(0,500,2000,constants.lfdtISOCAN,<br>constants.lfdmNormal, True) |

## 10.20 Wait

| | |
|---|---|
| 函数名称 | Void wait([in] long ATimeMs); |
| 功能介绍 | 等待时间 |
| 调用位置 | |
| 输入参数 | ATimeMs |
| 返回值 | 无 |
| 示例 | wait(100) |

## 10.21 set_mapping_verbose

| | |
|---|---|
| 函数名称 | Void set_mapping_verbose(<br>[in] long AAppChannelIndex,<br>[in] TTSAppChannelType AAppChannelType,<br>[in] TTSBusToolDeviceType AHWDeviceType,<br>[in] long AHWIndex,<br>[in] long AHWChannelIndex,<br>[in] long AHWDeviceSubType,<br>[in] BSTR AHWDeviceName,<br>[in] VARIANT_BOOL AMappingDisabled); |
| 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 |
| 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 | 参数 1: 应用程序的通道编号<br>参数 2: 通道类型，包括 APP_CAN=0,APP_LIN=1，APP_Flexray=2<br>参数 3：硬件类型根据<br>TTSBusToolDeviceType:in t{<br>BUS_UNKNOWN_TYPE=0, |

| | |
|---|---|
| | TS_TCP_DEVICE=1,<br>XL_USB_DEVICE=2,<br>TS_USB_DEVICE=3,<br>PEAK_USB_DEVICE=4,<br>KVASER_USB_DEVICE=5,<br>ZLG_USB_DEVICE=6,<br>ICS_USB_DEVICE=7,<br>TS_TC1005_DEVICE=8}；<br>参数 4 :AHWIndex,硬件编号<br>参数 5：AHWChannelIndex 硬件的通道编号，当有多个同样类型的硬件存在的时候，这是第几个<br>参数 6： AHWDeviceSubType, 该设备下子设备，比如 TS_USB_DEVICE 下面包含 TC1001， TC1011 等设备。<br>参数 7:AHWDeviceName 硬件名称，比如 TOSUN,Vector。<br>参数 8:AMappingDisabled 是否使能该映射，如果设置 False，则该通道不能使用。 |
| 返回值 | 无 |
| 示例 | set_mapping_verbose(CHANNEL_INDEX.CHN1,0,1,0,1， TC1016, 0, True) |

## 10.22 get_mapping_verbose

| | |
|---|---|
| 函数名称 | void get_mapping_verbose(<br>[in] TTSAppChannelType AAppChannelType,<br>[in] long AAppChannelIndex,<br>[out] TTSBusToolDeviceType* AHWDeviceType,<br>[out] long* AHWIndex,<br>[out] long* AHWChannelIndex,<br>[out] long* AHWDeviceSubType,<br>[out] BSTR* AHWDeviceName,<br>[out] VARIANT_BOOL* AMappingDisabled); |
| 功能介绍 | 获取指定通道映射信息 |
| 调用位置 | 在使能总线统计之后可以调用 |

| | |
|---|---|
| 输入参数 | 参数 1:通道类型，包括 APP_CAN=0,APP_LIN=1，APP_Flexray=2<br>参数 2: 应用程序的通道编号 |
| 返回值 | 无 |
| 示例 | get_mapping_verbose(AAppChannelType.APP_CAN ，CHANNEL_INDEX.CHN1) |

## 10.23 set_vendor_detect_preferences

| | |
|---|---|
| 函数名称 | void set_vendor_detect_preferences(<br>[in] VARIANT_BOOL ADetectTOSUN,<br>[in] VARIANT_BOOL ADetectVector,<br>[in] VARIANT_BOOL ADetectPEAK,<br>[in] VARIANT_BOOL ADetectKvaser,<br>[in] VARIANT_BOOL ADetectZLG,<br>[in] VARIANT_BOOL ADetectIntrepidcs,<br>[in] VARIANT_BOOL ADetectCANable); |
| 功能介绍 | 设置供应商检测选项,连接其他厂家的硬件,需要先调用此函数 |
| 调用位置 | 通道映射之前 |
| 输入参数 | ADetectTOSUN：同星设备<br>ADetectVector:Vector 设备<br>ADetectPEAK:PEAK 设备<br>ADetectKvaser:Kvaser 设备<br>ADetectZLG:ZLG 设备<br>ADetectIntrepidcs:Intrepidcs 设备<br>ADetectCANable:CANable 设备 |
| 返回值 | 无 |
| 示例 | set_vendor_detect_preferences(true, true, false, false, false, false, false) |

## 10.24 get_vendor_detect_preferences

| | |
|---|---|
| 函数名称 | void get_vendor_detect_preferences(<br>out bool ADetectTOSUN, |

| | |
|---|---|
| | out bool ADetectVector,<br>out bool ADetectPEAK,<br>out bool ADetectKvaser,<br>out bool ADetectZLG,<br>out bool ADetectIntrepidcs,<br>out bool ADetectCANable); |
| 功能介绍 | 获取供应商检测选项 |
| 调用位置 | 通道映射之前 |
| 输入参数 | |
| 返回值 | 无 |
| 示例 | get_vendor_detect_preferences(true, true, false, false, false, false, false) |

## 10.25 make_toast

| | |
|---|---|
| 函数名称 | void make_toast(<br>[in] BSTR AString,<br>[in] long ALevel); |
| 功能介绍 | TSMaster 弹出提示窗,同时系统消息打印信息 |
| 调用位置 | 通道映射之前 |
| 输入参数 | AMsg:打印消息<br>ALevel:消息等级 可以在 TSMaster 系统常量里面查看 |
| 返回值 | 无 |
| 示例 | make_toast("测试消息", 3) |

## 10.26 get_system_var_generic

| | |
|---|---|
| 函数名称 | VARIANT_BOOL get_system_var_generic(<br>[in] BSTR ACompleteName,<br>[out] BSTR* AValue); |
| 功能介绍 | 获取系统变量值 |

| | |
|---|---|
| 调用位置 | |
| 输入参数 | ACompleteName:系统变量名 |
| 返回值 | AValue：系统变量值 |
| 示例 | get_system_var_generic("Var0") |

## 10.27 set_system_var_generic

| | |
|---|---|
| 函数名称 | VARIANT_BOOL set_system_var_generic(<br>[in] BSTR ACompleteName,<br>[in] BSTR AValue); |
| 功能介绍 | 设置系统变量值 |
| 调用位置 | |
| 输入参数 | ACompleteName:系统变量名<br>AValue：系统变量值 |
| 返回值 | |
| 示例 | set_system_var_generic(Var0, 0) |

## 10.28 wait_system_var

| | |
|---|---|
| 函数名称 | VARIANT_BOOL wait_system_var(<br>[in] BSTR ACompleteName,<br>[in] BSTR AValue,<br>[in] long ATimeoutMs); |
| 功能介绍 | 判断一定时间内,系统变量是否等于指定值 |
| 调用位置 | |
| 输入参数 | ACompleteName:系统变量名<br>AValue:系统变量值<br>ATimeoutMs:超时时间 |
| 返回值 | |
| 示例 | wait_system_var(Var0, 5, 50) |

## 10.29 transmit_can_async

| | |
|---|---|
| 函数名称 | void transmit_can_async([in] PCAN ACAN); |
| 功能介绍 | 以异步的方式发送 CAN 报文 |
| 调用位置 | 异步发送 CAN 报文 |
| 输入参数 | ACAN：CAN 数据包。 TLIBCAN 数据组成请查看章节：TLIBCAN：Classic CAN 数据类型 |
| 返回值 | 无 |
| 示例 | app = win32com.client.Dispatch("TSMaster.TSApplication")<br>com = app.TSCOM()<br>c = win32com.client.Record("TCAN", app)<br>c.FIdxChn = 0<br>c.FIsExtendedId = 0<br>c.FIsRemote = False<br>c.FIdentifier = 0x123<br>c.FDLC = 8<br>c.FDatas = VARIANT(pythoncom.VT_ARRAY \| pythoncom.VT_I1, [1, 2, 3, 4, 5, 6, 7, 8])<br>com.transmit_can_async(c) |

## 10.30 transmit_can_sync

| | |
|---|---|
| 函数名称 | void transmit_can_sync(<br>[in] PCAN ACAN,<br>[in] long ATimeoutMs,<br>[out] VARIANT_BOOL* ASuccess); |
| 功能介绍 | 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。 |
| 局限性 | 因为此函数需要设备 API 深度支持，因此，目前支持此函数接口的只有 TS 和 Vector 系列设备。 |
| 调用位置 | 同步发送 CAN 报文的场合 |
| 输入参数 | ACAN：报文数据<br>ATimeoutMs：超时时间 |

| | |
|---|---|
| 返回值 | ASuccess:表示发送是否成功 |
| 示例 | app = win32com.client.Dispatch("TSMaster.TSApplication")<br>com = app.TSCOM()<br>c = win32com.client.Record("TCAN", app)<br>c.FIdxChn = 0<br>c.FIsExtendedId = 0<br>c.FIsRemote = False<br>c.FIdentifier = 0x123<br>c.FDLC = 8<br>c.FDatas = VARIANT(pythoncom.VT_ARRAY \| pythoncom.VT_I1, [1, 2, 3, 4, 5, 6, 7, 8])<br>com.transmit_can_async(c，20) |

## 10.31 transmit_canfd_async

| | |
|---|---|
| 函数名称 | void transmit_canfd_async([in] PCANFD ACANFD); |
| 功能介绍 | 以异步的方式发送 CANFD 报文 |
| 调用位置 | 异步发送 CAN 报文 |
| 输入参数 | ACAN：CAN 数据包。 TLIBCAN 数据组成请查看章节：TLIBCANFD：CANFD 数据类型 |
| 返回值 | 无 |
| 示例 | com = app.TSCOM()<br>cFD = win32com.client.Record("TCANFD", app)<br>cFD.FIdxChn = 0<br>cFD.FIsExtendedId = 0<br>cFD.FIsEDL = True<br>cFD.FIsBRS = True<br>cFD.FIsESI = False<br>cFD.FIdentifier = 0x345<br>cFD.FDLC = 15<br>cFD.FDatas = VARIANT(pythoncom.VT_ARRAY \| pythoncom.VT_I1, [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, 33, 34, 35, 36, 37,38, 39, 40, 41, 42, |

| | |
|---|---|
| | 43, 44,45, 46, 47, 48, 49, 50, 51,52, 53, 54, 55, 56, 57, 58,59, 60, 61, 62, 63, 64])<br>com.transmit_canfd_async(cFD) |

## 10.32 transmit_canfd_sync

| | |
|---|---|
| 函数名称 | void transmit_canfd_sync(<br>[in] PCANFD ACANFD,<br>[in] long ATimeoutMs,<br>[out] VARIANT_BOOL* ASuccess); |
| 功能介绍 | 发送 CANFD 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 CANFD 报文一定已经成功发送到了 CAN 总线上面。 |
| 调用位置 | 发送 CANFD 报文 |
| 局限性 | 因为此函数需要设备 API 深度支持，因此，目前支持此函数接口的只有 TS 和 Vector 系列设备。 |
| 输入参数 | ACAN：CAN 数据包。 TLIBCAN 数据组成请查看章节：TLIBCANFD：CANFD 数据类型，ATimeoutMs 超时时间 |
| 返回值 | 无 |
| 示例 | com = app.TSCOM()<br>cFD = win32com.client.Record("TCANFD", app)<br>cFD.FIdxChn = 0<br>cFD.FIsExtendedId = 0<br>cFD.FIsEDL = True<br>cFD.FIsBRS = True<br>cFD.FIsESI = False<br>cFD.FIdentifier = 0x345<br>cFD.FDLC = 15<br>cFD.FDatas = VARIANT(pythoncom.VT_ARRAY \| pythoncom.VT_I1, [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, 33, 34, 35, 36, 37,38, 39, 40, 41, 42, 43, 44,45, 46, 47, 48, 49, 50, 51,52, 53, 54, 55, 56, 57, 58,59, 60, 61, 62, 63, 64])<br>com.transmit_canfd_sync(cFD， 50) |

## 10.33 transmit_lin_async

| | |
|---|---|
| 函数名称 | Void transmit_lin_async([in] PLIN ALIN); |
| 功能介绍 | 以异步的方式发送 LIN 报文 |
| 调用位置 | 异步发送 LIN 报文 |
| 输入参数 | ALIN：LIN 报文数据包 |
| 返回值 | ==0：发送成功<br>其他值：发送失败 |
| 示例 | com = app.TSCOM()<br>cLIN = win32com.client.Record("TLIN", app)<br>cLIN.FIdxChn = 0<br>cLIN.FIsTX = True<br>cLIN.FIsError = False<br>cLIN.FIdentifier = 0x3C<br>cLIN.FDLC = 8<br>cLIN.FDatas = VARIANT(pythoncom.VT_ARRAY \| pythoncom.VT_I1, [1, 2, 3, 4, 5, 6, 7, 8,])<br>com.transmit_lin_async(cLIN) |

## 10.34 transmit_lin_sync

| | |
|---|---|
| 函数名称 | void transmit_lin_sync(<br>[in] PLIN ALIN,<br>[in] long ATimeoutMs,<br>[out] VARIANT_BOOL* ASuccess); |
| 功能介绍 | 同步发送 LIN 报文 |
| 调用位置 | 在需要同步发送 LIN 报文的场合 |
| 输入参数 | ALIN：LIN 报文数据包<br>ATimeoutMs：超时时间 |
| 返回值 | ==0：发送成功<br>其他值：发送失败 |

| | |
|---|---|
| 示例 | com = app.TSCOM()<br>cLIN = win32com.client.Record("TLIN", app)<br>cLIN.FIdxChn = 0<br>cLIN.FIsTX = True<br>cLIN.FIsError = False<br>cLIN.FIdentifier = 0x3C<br>cLIN.FDLC = 8<br>cLIN.FDatas = VARIANT(pythoncom.VT_ARRAY \| pythoncom.VT_I1, [1, 2, 3, 4, 5, 6, 7, 8,])<br>com.transmit_lin_sync(cLIN，50) |

## 10.35 add_cyclic_msg_can

| | |
|---|---|
| 函数名称 | void add_cyclic_msg_can(<br>[in] PCAN ACAN,<br>[in] single APeriodMs); |
| 功能介绍 | 增加周期发送的报文，增加完成后，TSMasterAPI 自动完成报文的周期发送 |
| 调用位置 | 在需要周期性发送 CAN 报文的场合 |
| 输入参数 | ACAN：CAN 报文数据包<br>APeriodMS：周期值 |
| 返回值 | |
| 示例 | com = app.TSCOM()<br>c = win32com.client.Record("TCAN", app)<br>c.FIdxChn = 0<br>c.FIsExtendedId = 0<br>c.FIsRemote = False<br>c.FIdentifier = 0x123<br>c.FDLC = 8<br>c.FDatas = VARIANT(pythoncom.VT_ARRAY \| pythoncom.VT_I1, [1, 2, 3, 4, 5, 6, 7, 8])<br>com.add_cyclic_msg_can(c，20) |

## 10.36 add_cyclic_mag_canfd

| | |
|---|---|
| 函数名称 | void add_cyclic_msg_canfd(<br>[in] PCANFD ACANFD,<br>[in] single APeriodMs); |
| 功能介绍 | 增加周期发送的报文，增加完成后，TSMasterAPI 自动完成报文的周期发送 |
| 调用位置 | 在需要周期性发送 CANFD 报文的场合 |
| 输入参数 | ACAN：CAN 报文数据包<br>APeriodMS：周期值 |
| 返回值 | |
| 示例 | com = app.TSCOM()<br>c = win32com.client.Record("TCAN", app)<br>com.add_cyclic_msg_canfd(c，20) |

## 10.37 delete_cyclic_msg_can

| | |
|---|---|
| 函数名称 | void delete_cyclic_msg_can([in] PCAN ACAN); |
| 功能介绍 | 停止周期性发送 CAN 报文 |
| 调用位置 | 在需要停止周期性发送 CAN 报文的场合 |
| 输入参数 | ACAN：CAN 报文数据包 |
| 返回值 | ==0：删除成功<br>其他值：删除失败，查询错误码 |
| 示例 | com = app.TSCOM()<br>c = win32com.client.Record("TCAN", app)<br>com.delete_cyclic_msg_can(c) |

## 10.38 delete_cyclic_msg_canfd

| | |
|---|---|
| 函数名称 | void delete_cyclic_msg_canfd([in] PCANFD ACANFD); |
| 功能介绍 | 停止周期性发送 CANFD 报文 |

| | |
|---|---|
| 调用位置 | 在需要停止周期性发送 CANFD 报文的场合 |
| 输入参数 | ACANFD：CANFD 报文数据包 |
| 返回值 | ==0：删除成功<br>其他值：删除失败，查询错误码 |
| 示例 | com = app.TSCOM()<br>c = win32com.client.Record("TCAN", app)<br>com.delete_cyclic_msg_canfd(c) |

## 10.39 delete_cyclic_msg

| | |
|---|---|
| 函数名称 | void delete_cyclic_msgs(); |
| 功能介绍 | 停止周期性发送报文 |
| 调用位置 | 在需要停止周期性发送报文的场合 |
| 输入参数 | 无 |
| 返回值 | ==0：删除成功<br>其他值：删除失败，查询错误码 |
| 示例 | app = win32com.client.Dispatch("TSMaster.TSApplication")<br>com = app.TSCOM()<br>c = win32com.client.Record("TCAN", app)<br>com.delete_cyclic_msg(c) |

## 10.40 enable_bus_statistics

| | |
|---|---|
| 函数名称 | void enable_bus_statistics([in] VARIANT_BOOL AEnable) |
| 功能介绍 | 使能总线统计 |
| 调用位置 | 在需要计算总线统计信息的场合调用 |
| 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：删除成功<br>其他值：删除失败，查询错误码 |
| 示例 | enable_bus_statistics（true） |

## 10.41 clear_bus_statistics

| | |
|---|---|
| 函数名称 | void clear_bus_statistics() |
| 功能介绍 | 清除总线统计 |
| 调用位置 | 在需要计算总线统计信息的场合调用 |
| 输入参数 | 无 |
| 返回值 | 无 |
| 示例 | clear_bus_statistics() |

## 10.42 get_bus_statistics

| | |
|---|---|
| 函数名称 | double get_bus_statistics(<br>[in] TTSAppChannelType ABusType,<br>[in] long AIdxChn,<br>[in] TTSCANBusStatistics AIdxStat); |
| 功能介绍 | 获取总线统计信息 |
| 调用位置 | 在需要获取总线统计信息的场合调用 |
| 输入参数 | ABusType:通道类型<br>AIdxChn：应用程序通道<br>AIdxStat：总线统计类型 |
| 返回值 | 无 |
| 示例 | get_bus_statistics(TTSAppChannelType.APP_CAN, 0,<br>TTSCANBusStatistics.cbsBusLoad) |

## 10.43 get_fps_can

| | |
|---|---|
| 函数名称 | long get_fps_can( [in] long AIdxChn,<br>[in] long AIdentifier); |
| 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<br>参数 2:AIdentifier can 消息的标识符 |
| 返回值 | ==0:设置成功<br>其他:失败，根据错误码查看错误类型 |
| 示例 | get_fps_can（AIdxChn.CH1,0x123） |

## 10.44 get_fps_canfd

| | |
|---|---|
| 函数名称 | long get_fps_canfd( [in] long AIdxChn,<br>[in] long AIdentifier); |
| 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<br>参数 2:AIdentifier canfd 消息的标识符 |
| 返回值 | ==0:设置成功<br>其他:失败，根据错误码查看错误类型 |
| 示例 | get_fps_canfd（AIdxChn.CH1,0x123） |

## 10.45 get_fps_lin

| | |
|---|---|
| 函数名称 | long get_fps_lin( [in] long AIdxChn,<br>[in] long AIdentifier); |
| 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<br>参数 2:AIdentifier lin 消息的标识符 |
| 返回值 | ==0:设置成功<br>其他:失败，根据错误码查看错误类型 |
| 示例 | get_fps_lin（AIdxChn.CH1,0x123） |

## 10.46 start_logging

| | |
|---|---|
| 函数名称 | HRESULT start_logging([in] BSTR AFileName) |
| 功能介绍 | 启动报文记录，记录文件 blf 格式 |
| 调用位置 | 需要记录整个运行过程中总线上的报文通讯 |
| 输入参数 | "AFileName"：记录文件名（后缀为.blf），可以为空字符串<br>注意：如果 AFileName 传输为空字符串“”，则数据文件默认存储在 TSMaster 的安装路径下面。AFile 数据格式路径不能出错，如果路径是无效的， 创建 该数据文件无效， 会触发模块内部异常。 |
| 返回值 | ==0:设置成功<br>其他:失败，根据错误码查看错误类型 |
| 示例 | start_logging(“c:\\1.blf”) |

## 10.47 stop_logging

| | |
|---|---|
| 函数名称 | HRESULT stop_logging() |
| 功能介绍 | 停止报文记录 |
| 调用位置 | 该函数需要和 start_logging 配合使用， 停止报文记录， 报文会保存为相应的 blf 文件。 |
| 输入参数 | 无 |
| 返回值 | 无 |
| 示例 | Stop_logging() |

## 10.48 enable_event_can

| | |
|---|---|
| 函数名称 | Void enable_event_can([in] VARIANT_BOOL AEnable); |
| 功能介绍 | 注册 CAN 数据包接收回调函数 |
| 调用位置 | 如果用户想基于接收回调机制来处理接收的报文，在连接工具成功过后， 可以调用此函数注册回调函数。 |
| 输入参数 | Bool,是否启动 |

| | |
|---|---|
| 返回值 | 无 |
| 示例 | enable_event_can(True) |

## 10.49 enable_event_canfd

| | |
|---|---|
| 函数名称 | Void enable_event_canfd([in] VARIANT_BOOL AEnable); |
| 功能介绍 | 注册 CANFD 数据包接收回调函数 |
| 调用位置 | 如果用户想基于接收回调机制来处理接收的报文，在连接工具成功过后， 可以调用此函数注册回调函数。 |
| 输入参数 | Bool,是否启动 |
| 返回值 | 无 |
| 示例 | enable_event_canfd(True) |

## 10.50 enable_event_lin

| | |
|---|---|
| 函数名称 | Void enable_event_lin([in] VARIANT_BOOL AEnable); |
| 功能介绍 | 注册 lin 数据包接收回调函数 |
| 调用位置 | 如果用户想基于接收回调机制来处理接收的报文，在连接工具成功过后， 可以调用此函数注册回调函数。 |
| 输入参数 | Bool,AEnable 是否启动 |
| 返回值 | 无 |
| 示例 | enable_event_lin(True) |

## 10.51 transmit_can_async_verbose

| | |
|---|---|
| 函数名称 | void transmit_can_async_verbose(<br>[in] long AIdxChn,<br>[in] VARIANT_BOOL AIsRemote,<br>[in] VARIANT_BOOL AIsExtended, |

| | |
|---|---|
| | [in] long ADLC,<br>[in] long AIdentifier,<br>[in] BSTR ADatas); |
| 功能介绍 | 异步发送 can 报文 |
| 调用位置 | |
| 输入参数 | 参数 1：AIdxChn 通道设置<br>参数 2：AIsRemote 远程帧<br>参数 3：AIsExtended 扩展帧<br>参数 4：DLC 数据长度<br>参数 5：can 报文标识符<br>参数 6：报文数据 |
| 返回值 | 无 |
| 示例 | transmit_can_async_verbose(0, false，false，8,0x123，[0,1,2,3,4,5,6,7]); |

## 10.52 transmit_can_sync_verbose

| | |
|---|---|
| 函数名称 | void transmit_canfd_sync_verbose(<br>[in] long AIdxChn,<br>[in] VARIANT_BOOL AIsExtended,<br>[in] VARIANT_BOOL AIsEDL,<br>[in] VARIANT_BOOL AIsBRS,<br>[in] VARIANT_BOOL AIsESI,<br>[in] long ADLC,<br>[in] long AIdentifier,<br>[in] BSTR ADatas,<br>[in] long ATimeoutMs,<br>[out] VARIANT_BOOL* ASuccess); |
| 功能介绍 | 同步发送 can 报文 |
| 调用位置 | |
| 输入参数 | 参数 1：AIdxChn 通道<br>参数 2：AIsExtended 是否扩展帧<br>参数 3：AIsEDL 可扩展帧数据长度<br>参数 4：AIsBRS 位速率转换 0 表示转换可变速率<br>参数 5：AIsESI 错误状态指示位 |

| | |
|---|---|
| | 参数 6：DLC 数据长度<br>参数 7：数据标识符<br>参数 8：报文数据<br>参数 9：超时时间 |
| 返回值 | ASuccess:表示发送是否成功 |
| 示例 | transmit_canfd_sync_verbose(0,1,8,1,1，8,0x123，[0,1,2,3,4,5,6,7],20) |

## 10.53 transmit_canfd_async_berbose

| | |
|---|---|
| 函数名称 | void transmit_canfd_async_verbose(<br>[in] long AIdxChn,<br>[in] VARIANT_BOOL AIsExtended,<br>[in] VARIANT_BOOL AIsEDL,<br>[in] VARIANT_BOOL AIsBRS,<br>[in] VARIANT_BOOL AIsESI,<br>[in] long ADLC,<br>[in] long AIdentifier,<br>[in] BSTR ADatas); |
| 功能介绍 | 异步发送 canfd 报文 |
| 调用位置 | |
| 输入参数 | 参数 1：AIdxChn 通道<br>参数 2：AIsExtended 是否扩展帧<br>参数 3：AIsEDL 是否为 CANFD<br>参数 4：AIsBRS 位速率转换 0 表示转换可变速率<br>参数 5：AIsESI 错误状态指示位<br>参数 6：DLC 数据长度<br>参数 7：数据标识符<br>参数 8：报文数据 |
| 返回值 | |
| 示例 | transmit_canfd_async_verbose(0, false, true, false, false, 8, 0x123, [1,2,3,4,5,6,7,8]); |

## 10.54 transmit_canfd_sync_verbose

| | |
|---|---|
| 函数名称 | void transmit_canfd_sync_verbose(<br>[in] long AIdxChn,<br>[in] VARIANT_BOOL AIsExtended,<br>[in] VARIANT_BOOL AIsEDL,<br>[in] VARIANT_BOOL AIsBRS,<br>[in] VARIANT_BOOL AIsESI,<br>[in] long ADLC,<br>[in] long AIdentifier,<br>[in] BSTR ADatas,<br>[in] long ATimeoutMs,<br>[out] VARIANT_BOOL* ASuccess); |
| 功能介绍 | 同步发送 canfd 报文 |
| 调用位置 | |
| 输入参数 | 参数 1：AIdxChn 通道<br>参数 2：AIsExtended 是否扩展帧<br>参数 3：AIsEDL 是否为 CANFD<br>参数 4：AIsBRS 位速率转换 0 表示转换可变速率<br>参数 5：AIsESI 错误状态指示位<br>参数 6：DLC 数据长度<br>参数 7：数据标识符<br>参数 8：报文数据<br>参数 9：超时时间 |
| 返回值 | ASuccess 是否成功 |
| 示例 | transmit_canfd_sync_verbose(0, false, true, false, false, 15, 0x8,<br>"1,2,3,4,5,6,7,8", 1000); |

## 10.55 transmit_lin_async_verbose

| | |
|---|---|
| 函数名称 | void transmit_lin_async_verbose(<br>[in] long AIdxChn,<br>[in] long ADLC,<br>[in] long AIdentifier,<br>[in] BSTR ADatas); |

| | |
|---|---|
| 功能介绍 | 异步发送 lin 报文 |
| 调用位置 | |
| 输入参数 | 参数 1：发送通道<br>参数 2：DLC 数据长度<br>参数 3：数据标识符<br>参数 4：报文数据 |
| 返回值 | 无 |
| 示例 | transmit_lin_async_verbose(0,8,0x123，[0,1,2,3,4,5,6,7]) |

## 10.56 transmit_lin_sync_verbose

| | |
|---|---|
| 函数名称 | void transmit_lin_sync_verbose(<br>[in] long AIdxChn,<br>[in] long ADLC,<br>[in] long AIdentifier,<br>[in] BSTR ADatas,<br>[in] long ATimeoutMs,<br>[out] VARIANT_BOOL* ASuccess); |
| 功能介绍 | 同步发送 lin 报文 |
| 调用位置 | |
| 输入参数 | 参数 1：发送通道<br>参数 2：DLC 数据长度<br>参数 3：数据标识符<br>参数 4：报文数据<br>参数 5：超时时间 |
| 返回值 | ASuccess 是否成功 |
| 示例 | app = win32com.client.Dispatch("TSMaster.TSApplication")<br>com = app.TSCOM()<br>com.transmit_lin_sync_verbose(0, 8, 0x2, "1,2,3,4,5,6,7,8", 1000); |

## 10.57 add_cyclic_msg_can_verbose

| | |
|---|---|
| 函数名称 | void add_cyclic_msg_can_verbose(<br>[in] long AIdxChn,<br>[in] VARIANT_BOOL AIsExtended,<br>[in] long AIdentifier,<br>[in] long ADLC,<br>[in] BSTR ADatas,<br>[in] long APeriodMs); |
| 功能介绍 | 添加周期 can 报文发送 |
| 调用位置 | |
| 输入参数 | 参数 1：AIdxChn 通道配置<br>参数 2：AIsExtended 是否扩展帧<br>参数 3：AIdentifier 数据标识符<br>参数 4： ADLC 数据长度<br>参数 5：ADatas 数据报文<br>参数 6：APeriodMs 周期 |
| 返回值 | 无 |
| 示例 | add_cyclic_msg_can_verbose(0,false,0x345,8,[1,2,3,4,5,6,7,8],100); |

## 10.58 add_cyclic_msg_canfd_verbose

| | |
|---|---|
| 函数名称 | void add_cyclic_msg_canfd_verbose(<br>[in] long AIdxChn,<br>[in] VARIANT_BOOL AIsExtended,<br>[in] long AIdentifier,<br>[in] long ADLC,<br>[in] BSTR ADatas,<br>[in] long APeriodMs); |
| 功能介绍 | 添加周期 canfd 报文发送 |
| 调用位置 | |
| 输入参数 | 参数 1：AIdxChn 通道配置<br>参数 2：AIsExtended 是否扩展帧 |

| | |
|---|---|
| | 参数 3：AIdentifier 数据标识符<br>参数 4： ADLC 数据长度<br>参数 5：ADatas 数据报文<br>参数 6：APeriodMs 周期 |
| 返回值 | 无 |
| 示例 | add_cyclic_msg_canfd_verbose(0,false,0x345,8,[1,2,3,4,5,6,7,8],100); |

## 10.59 delete_cyclic_msg_can_verbose

| | |
|---|---|
| 函数名称 | void delete_cyclic_msg_can_verbose(<br>[in] long AIdxChn,<br>[in] VARIANT_BOOL AIsExtended,<br>[in] long AIdentifier); |
| 功能介绍 | 删除周期 can 报文发送 |
| 调用位置 | |
| 输入参数 | 参数 1：AIdxChn 应用程序通道配置<br>参数 2：AIsExtended 是否扩展帧<br>参数 3：AIdentifier 数据标识符 |
| 返回值 | 无 |
| 示例 | delete_cyclic_msg_can_verbose（0,0,0x123） |

## 10.60 delete_cyclic_msg_canfd_verbose

| | |
|---|---|
| 函数名称 | void delete_cyclic_msg_canfd_verbose(<br>[in] long AIdxChn,<br>[in] VARIANT_BOOL AIsExtended,<br>[in] long AIdentifier); |
| 功能介绍 | 删除周期 can 报文发送 |
| 调用位置 | |
| 输入参数 | 参数 1：AIdxChn 应用程序通道配置<br>参数 2：AIsExtended 是否扩展帧<br>参数 3：AIdentifier 数据标识符 |
| 返回值 | 无 |

| | |
|---|---|
| 示例 | delete_cyclic_msg_canfd_verbose（0,0,0x123） |

## 10.61 can_rbs_start

| | |
|---|---|
| 函数名称 | void can_rbs_start(); |
| 功能介绍 | 开启 can 的 rbs |
| 调用位置 | |
| 输入参数 | 无 |
| 返回值 | 无 |
| 示例 | can_rbs_start() |

## 10.62 can_rbs_stop

| | |
|---|---|
| 函数名称 | void can_rbs_stop(); |
| 功能介绍 | 停止 can 的 rbs |
| 调用位置 | |
| 输入参数 | 无 |
| 返回值 | 无 |
| 示例 | app = win32com.client.Dispatch("TSMaster.TSApplication")<br>com = app.TSCOM()<br>com.can_rbs_stop() |

## 10.63 can_rbs_is_running

| | |
|---|---|
| 函数名称 | void can_rbs_is_running([out] VARIANT_BOOL* AIsRunning); |
| 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<br>其他:失败，根据错误码查看错误类型 |
| 示例 | app = win32com.client.Dispatch("TSMaster.TSApplication")<br>com = app.TSCOM()<br>com.can_rbs_is_running(true) |

## 10.64 can_rbs_configure

| | |
|---|---|
| 函数名称 | void can_rbs_configure(<br>[in] VARIANT_BOOL AAutoStart,<br>[in] VARIANT_BOOL AAutoSendOnModification,<br>[in] VARIANT_BOOL AActivateNodeSimulation,<br>[in] TTSRBSInitValueOptions AInitValueOptions); |
| 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<br>参数 2:AAutoSendOnModification 是否自动修正发送报文<br>参数 3:AActivateNodeSimulation 是否自动启动节点仿真<br>参数 4:TLIBRBSInitValueOptions 枚举值，信号值初始值设置。 |
| 返回值 | ==0:设置成功<br>其他:失败，根据错误码查看错误类型 |
| 示例 | app = win32com.client.Dispatch("TSMaster.TSApplication")<br>com = app.TSCOM()<br>Com.can_rbs_configure(True,True,True,TLIBRBSInitValueOptions) |

## 10.65 can_rbs_activate_all_networks

| | |
|---|---|
| 函数名称 | void can_rbs_activate_all_networks(<br>[in] VARIANT_BOOL AEnable,<br>[in] VARIANT_BOOL AIncludingChildren); |
| 功能介绍 | 激活所有 networks |
| 调用位置 | 在需要 networks 的场合调用 |
| 输入参数 | 参数 1:AEnable 是否激活<br>参数 2:AIncludingChildren 是否包括子节点 |

| | |
|---|---|
| 返回值 | ==0:设置成功<br>其他:失败，根据错误码查看错误类型 |
| 示例 | app = win32com.client.Dispatch("TSMaster.TSApplication")<br>com = app.TSCOM()<br>Com.can_rbs_activate_all_networks(true,true) |

## 10.66 can_rbs_activate_network_by_name

| | |
|---|---|
| 函数名称 | void can_rbs_activate_network_by_name(<br>[in] long AIdxChn,<br>[in] VARIANT_BOOL AEnable,<br>[in] BSTR ANetworkName,<br>[in] VARIANT_BOOL AIncludingChildren); |
| 功能介绍 | 激活所有 networks |
| 调用位置 | |
| 输入参数 | 参数 1:AIdxChn 通道编号<br>参数 2:AEnable 是否激活<br>参数 3:ANetworkName 网络名称<br>参数 4:AIncludingChildren 是否包括子节点 |
| 返回值 | ==0:设置成功<br>其他:失败，根据错误码查看错误类型 |
| 示例 | app = win32com.client.Dispatch("TSMaster.TSApplication")<br>com = app.TSCOM()<br>Com.can_rbs_activate_network_by_name(CHANNEL_INDEX.CHN1,True,b "<br>CAN_FD_Powertrain " ,True) |

## 10.67 can_rbs_activate_node_by_name

| | |
|---|---|
| 函数名称 | void can_rbs_activate_node_by_name(<br>[in] long AIdxChn,<br>[in] VARIANT_BOOL AEnable,<br>[in] BSTR ANetworkName,<br>[in] BSTR ANodeName,<br>[in] VARIANT_BOOL AIncludingChildren); |

| | |
|---|---|
| 功能介绍 | 激活节点 |
| 调用位置 | |
| 输入参数 | 参数 1:AIdxChn 通道编号<br>参数 2:AEnable 是否激活<br>参数 3:ANetworkName 网络名称<br>参数 4:NodeName 节点名称<br>参数 5:AIncludingChildren AIncludingChildren |
| 返回值 | ==0:设置成功<br>其他:失败，根据错误码查看错误类型 |
| 示例 | app = win32com.client.Dispatch("TSMaster.TSApplication")<br>com = app.TSCOM()<br>com.can_rbs_activate_node_by_name(CHANNEL_INDEX.CHN1,True ,b " CAN_FD_Powertrain " ,b " Engine " ,True) |

## 10.68 can_rbs_activate_message_by_name

| | |
|---|---|
| 函数名称 | void can_rbs_activate_message_by_name(<br>[in] long AIdxChn,<br>[in] VARIANT_BOOL AEnable,<br>[in] BSTR ANetworkName,<br>[in] BSTR ANodeName,<br>[in] BSTR AMsgName); |
| 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<br>参数 2:AEnable 是否激活<br>参数 3:ANetworkName 网络名称<br>参数 4:NodeName 节点名称<br>参数 5:MessageName 信号报文名称 |
| 返回值 | ==0:设置成功<br>其他:失败，根据错误码查看错误类型 |
| 示例 | app = win32com.client.Dispatch("TSMaster.TSApplication")<br>com = app.TSCOM() |

| | |
|---|---|
| | com.can_rbs_activate_message_by_name(CHANNEL_INDEX.CHN1,True,b " CAN_FD_Powertrain " , b " Engine " ,b EngineData " ) |

## 10.69 can_rbs_get_signal_value_by_element

| | |
|---|---|
| 函数名称 | void can_rbs_get_signal_value_by_element(<br>[in] long AIdxChn,<br>[in] BSTR ANetworkName,<br>[in] BSTR ANodeName,<br>[in] BSTR AMsgName,<br>[in] BSTR ASignalName,<br>[out] double* AValue); |
| 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<br>参数 2:ANetwork network name<br>参数 3:ANodeName 节点名字<br>参数 4:AMessageName 报文名称<br>参数 5:ASignalName 信号名字 |
| 返回值 | ==0:设置成功<br>其他:失败，根据错误码查看错误类型 |
| 示例 | app = win32com.client.Dispatch("TSMaster.TSApplication")<br>com = app.TSCOM()<br>com.can_rbs_get_signal_value_by_element(CHANNEL_INDEX.CHN1,b " CAN_FD_Powertrain " ,b " Engine " ,b EngineData " ,b " Gear " , AValue) |

## 10.70 can_rbs_get_signal_value_by_address

| | |
|---|---|
| 函数名称 | void can_rbs_get_signal_value_by_address(<br>[in] BSTR ASymbolAddress,<br>[out] double* AValue); |
| 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 | ASymboladdress 指定信号信息 |
| 返回值 | Avalue 设置的信号值 |
| 示例 | app = win32com.client.Dispatch("TSMaster.TSApplication")<br>com = app.TSCOM()<br>com.can_rbs_get_signal_value_by_address(CHANNEL_INDEX.CHN1，b " CAN_FD_Powertrain " ， b " Engine " ， b " EngineData " ， b " Gear " , AValue) |

## 10.71 can_rbs_set_signal_value_by_element

| | |
|---|---|
| 函数名称 | void can_rbs_set_signal_value_by_element(<br>[in] long AIdxChn,<br>[in] BSTR ANetworkName,<br>[in] BSTR ANodeName,<br>[in] BSTR AMsgName,<br>[in] BSTR ASignalName,<br>[in] double AValue); |
| 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:AIdxChn 应用程序通道<br>参数 2:ANetwork 网络名称<br>参数 3:ANodeName 节点名称<br>参数 4:AMessageName 报文名称<br>参数 5:ASignalName 信号名称<br>参数 6:Avalue 信号值 |
| 返回值 | ==0:设置成功<br>其他:失败，根据错误码查看错误类型 |
| 示例 | app = win32com.client.Dispatch("TSMaster.TSApplication")<br>com = app.TSCOM()<br>Com.can_rbs_set_signal_value_by_element(CHANNEL_INDEX.CHN1,b " CAN_FD_Powertrain " ,b " Engine " ,b " EngineData " ,b " Gear " , AValue) |

## 10.72 can_rbs_set_signal_value_by_address

| | |
|---|---|
| 函数名称 | void can_rbs_set_signal_value_by_address(<br>[in] BSTR ASymbolAddress,<br>[in] double AValue); |
| 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 信号信息<br>参数 2:Avalue 信号值 |
| 返回值 | ==0:设置成功<br>其他:失败，根据错误码查看错误类型 |
| 示例 | app = win32com.client.Dispatch("TSMaster.TSApplication")<br>com = app.TSCOM()<br>com.can_rbs_set_signal_value_by_address(CHANNEL_INDEX.CHN 1/b " CAN_FD_Powertrain " /b " Engine " /b " EngineData " /b " Gear " , AValue) |

## 10.73 fifo_enable_receive_fifo

| | |
|---|---|
| 函数名称 | void fifo_enable_receive_fifo(); |
| 功能介绍 | 使能设备内部缓存接收模式 |
| 调用位置 | 应用程序如果想通过 读取缓存的方式读取 数据，在执行 tsfifo_receive_can/canfd/lin_message_list 之前，必须要执行此函数 |
| 输入参数 | 无 |
| 返回值 | 无 |
| 示例 | app = win32com.client.Dispatch("TSMaster.TSApplication")<br>com = app.TSCOM()<br>com.fifo_enable_receive_fifo() |

## 10.74 fifo_disable_receive_fifo

| | |
|---|---|
| 函数名称 | void fifo_disable_receive_fifo(); |

| | |
|---|---|
| 功能介绍 | 关闭设备内部缓存接收模式 |
| 调用位置 | 用户不需要采用读取缓存的方式接收报文的时候，调用此函数关闭内部缓存。 |
| 输入参数 | 无 |
| 返回值 | 无 |
| 示例 | app = win32com.client.Dispatch("TSMaster.TSApplication")<br>com = app.TSCOM()<br>com.fifo_disable_receive_fifo() |

## 10.75 fifo_enable_receive_error_frames

| | |
|---|---|
| 函数名称 | void fifo_enable_receive_error_frames(); |
| 功能介绍 | 使能驱动内部的接收错误帧机制 |
| 调用位置 | |
| 输入参数 | 无 |
| 返回值 | 无 |
| 示例 | app = win32com.client.Dispatch("TSMaster.TSApplication")<br>com = app.TSCOM()<br>com.fifo_enable_receive_error_frames() |

## 10.76 fifo_disable_receive_error_frames

| | |
|---|---|
| 函数名称 | void fifo_disable_receive_error_frames(); |
| 功能介绍 | 关闭驱动内部的接收错误帧机制 |
| 调用位置 | |
| 输入参数 | 无 |
| 返回值 | 无 |
| 示例 | app = win32com.client.Dispatch("TSMaster.TSApplication")<br>com = app.TSCOM() |

| | |
|---|---|
| | com.fifo_disable_receive_error_frames() |

## 10.77 fifo_receive_can_msg

| | |
|---|---|
| 函数名称 | VARIANT_BOOL fifo_receive_can_msg(<br>[in] long AIdxChnReq,<br>[in] VARIANT_BOOL AIncludeTx,<br>[out] long* AIdxChnResp,<br>[out] VARIANT_BOOL* AIsRemote,<br>[out] VARIANT_BOOL* AIsExtended,<br>[out] long* ADLC,<br>[out] long* AIdentifier,<br>[out] int64* ATimestampUs,<br>[out] BSTR* ADatas); |
| 功能介绍 | 读取缓存的 CAN 报文帧 |
| 调用位置 | 在需要读取报文数据包的场合 |
| 输入参数 | 参数 1：AIdxChnReq 通道请求索引<br>参数 2：AIncludeTx 是否包含发送报文 |
| 返回值 | 是否接收成功 |
| 示例 | app = win32com.client.Dispatch("TSMaster.TSApplication")<br>com = app.TSCOM()<br>com.fifo_receive_can_message_list(0,False,0,False,False,8,0x1,<br>100,ADatas) |

## 10.78 fifo_receive_canfd_msg

| | |
|---|---|
| 函数名称 | VARIANT_BOOL fifo_receive_canfd_msg(<br>[in] long AIdxChnReq,<br>[in] VARIANT_BOOL AIncludeTx,<br>[out] long* AIdxChnResp,<br>[out] VARIANT_BOOL* AIsRemote,<br>[out] VARIANT_BOOL* AIsExtended,<br>[out] VARIANT_BOOL* AIsEDL, |

| | |
|---|---|
| | [out] VARIANT_BOOL* AIsBRS,<br>[out] long* ADLC,<br>[out] long* AIdentifier,<br>[out] int64* ATimestampUs,<br>[out] BSTR* ADatas); |
| 功能介绍 | 读取缓存的 CAN FD 报文帧 |
| 调用位置 | 在需要读取 CANFD 报文数据包的场合 |
| 输入参数 | 参数 1：AIdxChnReq 通道请求索引<br>参数 2：AIncludeTx 是否包含发送报文 |
| 返回值 | 返回实际读取的报文数量， 如果没有任何报文，则返回值为 0。 |
| 示例 | app = win32com.client.Dispatch("TSMaster.TSApplication")<br>com = app.TSCOM()<br>com.fifo_receive_canfd_msg(0,False,0,False,False,False,15,<br>100,ADatas) |

## 10.79 fifo_receive_lin_msg

| | |
|---|---|
| 函数名称 | HRESULT fifo_receive_lin_msg(<br>[in] long AIdxChnReq,<br>[in] VARIANT_BOOL AIncludeTx,<br>[out] long* AIdxChnResp,<br>[out] long* ADLC,<br>[out] long* AIdentifier,<br>[out] int64* ATimestampUs,<br>[out] BSTR* ADatas,<br>[out, retval] VARIANT_BOOL* ASuccess); |
| 功能介绍 | 读取缓存的 LIN 报文帧 |
| 调用位置 | 在需要读取 LIN 报文数据包的场合 |
| 输入参数 | 参数 1：AIdxChnReq 通道请求索引<br>参数 2：AIncludeTx 是否包含发送报文 |
| 返回值 | 返回实际读取的报文数量和是否接收成功 |
| 示例 | app = win32com.client.Dispatch("TSMaster.TSApplication")<br>com = app.TSCOM()<br>com.fifo_receive_lin_msg(0,False,0,8,0x3d,100,ADatas,ASuccess) |

## 10.80 fifo_receive_fastlin_msg

| | |
|---|---|
| 函数名称 | VARIANT_BOOL fifo_receive_fastlin_msg(<br>[in] long AIdxChnReq,<br>[in] VARIANT_BOOL AIncludeTx,<br>[out] long* AIdxChnResp,<br>[out] long* ADLC,<br>[out] long* AIdentifier,<br>[out] int64* ATimestampUs,<br>[out] BSTR* ADatas); |
| 功能介绍 | 读取缓存的 LIN 报文帧 |
| 调用位置 | 在需要读取 LIN 报文数据包的场合 |
| 输入参数 | 参数 1：AIdxChnReq 通道请求索引<br>参数 2：AIncludeTx 是否包含发送报文 |
| 返回值 | 返回实际读取的报文数量和是否接收成功 |
| 示例 | app = win32com.client.Dispatch("TSMaster.TSApplication")<br>com = app.TSCOM()<br>com.fifo_receive_lin_msg(0,False,0,8,0x3d,100,ADatas) |

## 10.81 fifo_clear_can_receive_buffers

| | |
|---|---|
| 函数名称 | VARIANT_BOOL fifo_clear_can_receive_buffers([in] long AIdxChn); |
| 功能介绍 | 清除 CAN 通道里面缓存的报文 |
| 调用位置 | 在执行交互协议之前（比如 UDS 诊断），先调用此命令把缓存中的历史数据清除掉 |
| 输入参数 | "AIdxChn"：硬件通道 |
| 返回值 | 是否清除成功 |
| 示例 | app = win32com.client.Dispatch("TSMaster.TSApplication")<br>com = app.TSCOM()<br>com.fifo_clear_can_receive_buffers(0) |

## 10.82 fifo_clear_canfd_receive_buffers

| | |
|---|---|
| 函数名称 | VARIANT_BOOL fifo_clear_canfd_receive_buffers([in] long AIdxChn); |
| 功能介绍 | 清除 CANFD 通道里面缓存的报文 |
| 调用位置 | 在执行交互协议之前（比如 UDS 诊断），先调用此命令把缓存中的历史数据清除掉 |
| 输入参数 | "AIdxChn"：硬件通道 |
| 返回值 | 返回缓存中 CAN 报文的数量 |
| 示例 | app = win32com.client.Dispatch("TSMaster.TSApplication")<br>com = app.TSCOM()<br>com.fifo_clear_canfd_receive_buffers(0) |

## 10.83 fifo_clear_lin_receive_buffers

| | |
|---|---|
| 函数名称 | HRESULT fifo_clear_lin_receive_buffers(<br>[in] long AIdxChn,<br>[out, retval] VARIANT_BOOL* ASuccess); |
| 功能介绍 | 清除 LIN 模块的接收 Buffer 缓存 |
| 调用位置 | 在准备开始一次发送和接收之前，调用此函数清除驱动中的 LIN 缓存中的 数据，确保接收到的数据是最新的 |
| 输入参数 | "AIdxChn"：需要清除缓存的 LIN 通道 |
| 返回值 | ASuccess:表示是否清除成功 |
| 示例 | app = win32com.client.Dispatch("TSMaster.TSApplication")<br>com = app.TSCOM()<br>com.fifo_clear_lin_receive_buffers(APP_CHANNEL.CHN1,ASuccess) |

## 10.84 fifo_clear_fastlin_receive_buffers

| | |
|---|---|
| 函数名称 | VARIANT_BOOL fifo_clear_lin_receive_buffers([in] long AIdxChn); |
| 功能介绍 | 清除 LIN 通道里面缓存的报文 |

| | |
|---|---|
| 调用位置 | 在执行交互协议之前（比如 UDS 诊断），先调用此命令把缓存中的历史数据清除掉 |
| 输入参数 | "AIdxChn"：硬件通道 |
| 返回值 | 是否清除成功 |
| 示例 | app = win32com.client.Dispatch("TSMaster.TSApplication")<br>com = app.TSCOM()<br>com.fifo_clear_fastlin_receive_buffers(0) |

## 10.85 fifo_read_can_buffer_frame_count

| | |
|---|---|
| 函数名称 | VARIANT_BOOL fifo_read_can_buffer_frame_count(<br>[in] long AIdxChn,<br>[out] long* ACount); |
| 功能介绍 | 读取通道 can 缓冲帧数量 |
| 调用位置 | |
| 输入参数 | "AIdxChn"：硬件通道 |
| 返回值 | ACount can 报文数量 |
| 示例 | app = win32com.client.Dispatch("TSMaster.TSApplication")<br>com = app.TSCOM()<br>com.fifo_read_can_buffer_frame_count(CHANNEL_INDEX.CHN1, ACount) |

## 10.86 fifo_read_can_tx_buffer_frame_count

| | |
|---|---|
| 函数名称 | VARIANT_BOOL fifo_read_can_tx_buffer_frame_count(<br>[in] long AIdxChn,<br>[out] long* ACount); |
| 功能介绍 | 读取通道 can tx 数量 |
| 调用位置 | |
| 输入参数 | "AIdxChn"：硬件通道 |

| | |
|---|---|
| 返回值 | ACount can tx 报文数量 |
| 示例 | app = win32com.client.Dispatch("TSMaster.TSApplication")<br>com = app.TSCOM()<br>com.fifo_read_can_tx_buffer_frame_count(CHANNEL_INDEX.CHN1, Count) |

## 10.87 fifo_read_can_rx_buffer_frame_count

| | |
|---|---|
| 函数名称 | VARIANT_BOOL fifo_read_can_rx_buffer_frame_count(<br>[in] long AIdxChn,<br>[out] long* ACount); |
| 功能介绍 | 读取通道 can rx 数量 |
| 调用位置 | |
| 输入参数 | "AIdxChn"：硬件通道 |
| 返回值 | ACount can rx 报文数量 |
| 示例 | app = win32com.client.Dispatch("TSMaster.TSApplication")<br>com = app.TSCOM()<br>com.fifo_read_can_rx_buffer_frame_count(CHANNEL_INDEX.CHN1, Count) |

## 10.88 fifo_read_canfd_buffer_frame_count

| | |
|---|---|
| 函数名称 | VARIANT_BOOL fifo_read_canfd_buffer_frame_count(<br>[in] long AIdxChn,<br>[out] long* ACount); |
| 功能介绍 | 读取通道 canfd 报文缓存数量 |
| 调用位置 | |
| 输入参数 | "AIdxChn"：硬件通道 |
| 返回值 | ACount canfd 报文数量 |
| 示例 | app = win32com.client.Dispatch("TSMaster.TSApplication")<br>com = app.TSCOM() |

| | |
|---|---|
| | com.fifo_read_canfd_buffer_frame_count(CHANNEL_INDEX.CHN1, Count) |

## 10.89 fifo_read_canfd_tx_buffer_frame_count

| | |
|---|---|
| 函数名称 | VARIANT_BOOL fifo_read_canfd_tx_buffer_frame_count(<br>[in] long AIdxChn,<br>[out] long* ACount); |
| 功能介绍 | 读取通道 canfd tx 数量 |
| 调用位置 | |
| 输入参数 | "AIdxChn"：硬件通道 |
| 返回值 | ACount canfd tx 报文数量 |
| 示例 | app = win32com.client.Dispatch("TSMaster.TSApplication")<br>com = app.TSCOM()<br>com.fifo_read_canfd_tx_buffer_frame_count(CHANNEL_INDEX.CHN1, Count) |

## 10.90 fifo_read_canfd_rx_buffer_frame_count

| | |
|---|---|
| 函数名称 | VARIANT_BOOL fifo_read_canfd_rx_buffer_frame_count(<br>[in] long AIdxChn,<br>[out] long* ACount); |
| 功能介绍 | 读取通道 canfd rx 数量 |
| 调用位置 | |
| 输入参数 | "AIdxChn"：硬件通道 |
| 返回值 | ACount canfd tx 报文数量 |
| 示例 | app = win32com.client.Dispatch("TSMaster.TSApplication")<br>com = app.TSCOM()<br>com.fifo_read_canfd_rx_buffer_frame_count(CHANNEL_INDEX.CHN1, Count) |

## 10.91 fifo_read_lin_buffer_frame_count

| | |
|---|---|
| 函数名称 | VARIANT_BOOL fifo_read_lin_buffer_frame_count(<br>[in] long AIdxChn,<br>[out] long* ACount); |
| 功能介绍 | 读取 LIN 缓冲区报文帧数 |
| 调用位置 | |
| 输入参数 | "AIdxChn"：硬件通道 |
| 返回值 | ACount:缓存帧数 |
| 示例 | app = win32com.client.Dispatch("TSMaster.TSApplication")<br>com = app.TSCOM()<br>com.fifo_read_canfd_rx_buffer_frame_count(CHANNEL_INDEX.CHN1, ACount) |

## 10.92 fifo_read_lin_tx_buffer_frame_count

| | |
|---|---|
| 函数名称 | VARIANT_BOOL fifo_read_lin_tx_buffer_frame_count(<br>[in] long AIdxChn,<br>[out] long* ACount); |
| 功能介绍 | 读取 LIN tx 通道报文帧数 |
| 调用位置 | |
| 输入参数 | "AIdxChn"：硬件通道 |
| 返回值 | ACount:缓存帧数 |
| 示例 | app = win32com.client.Dispatch("TSMaster.TSApplication")<br>com = app.TSCOM()<br>com.fifo_read_lin_tx_buffer_frame_count(0, ACount); |

## 10.93 fifo_read_lin_rx_buffer_frame_count

| | |
|---|---|
| 函数名称 | VARIANT_BOOL fifo_read_lin_rx_buffer_frame_count(<br>[in] long AIdxChn, |

| | |
|---|---|
| | [out] long* ACount); |
| 功能介绍 | 读取 LIN rx 通道报文帧数 |
| 调用位置 | |
| 输入参数 | "AIdxChn"：硬件通道 |
| 返回值 | ACount:缓存帧数 |
| 示例 | app = win32com.client.Dispatch("TSMaster.TSApplication")<br>com = app.TSCOM()<br>com.fifo_read_lin_rx_buffer_frame_count(0, ACount); |

## 10.94 fifo_read_fastlin_buffer_frame_count

| | |
|---|---|
| 函数名称 | VARIANT_BOOL fifo_read_fastlin_buffer_frame_count(<br>[in] long AIdxChn,<br>[out] long* ACount); |
| 功能介绍 | 读取 fastlin 缓冲区报文帧数 |
| 调用位置 | |
| 输入参数 | "AIdxChn"：硬件通道 |
| 返回值 | ACount:缓存帧数 |
| 示例 | app = win32com.client.Dispatch("TSMaster.TSApplication")<br>com = app.TSCOM()<br>com.fifo_read_fastlin_buffer_frame_count(0, ACount); |

## 10.95 fifo_read_fastlin_tx_buffer_frame_count

| | |
|---|---|
| 函数名称 | VARIANT_BOOL fifo_read_fastlin_tx_buffer_frame_count(<br>[in] long AIdxChn,<br>[out] long* ACount); |
| 功能介绍 | 读取 fastlin tx 通道报文帧数 |
| 调用位置 | |

| | |
|---|---|
| 输入参数 | "AIdxChn"：硬件通道 |
| 返回值 | ACount:缓存帧数 |
| 示例 | app = win32com.client.Dispatch("TSMaster.TSApplication")<br>com = app.TSCOM()<br>com.fifo_read_fastlin_tx_buffer_frame_count(0, ACount); |

## 10.96 fifo_read_fastlin_rx_buffer_frame_count

| | |
|---|---|
| 函数名称 | VARIANT_BOOL fifo_read_fastlin_rx_buffer_frame_count(<br>[in] long AIdxChn,<br>[out] long* ACount); |
| 功能介绍 | 读取 fastlin rx 通道报文帧数 |
| 调用位置 | |
| 输入参数 | "AIdxChn"：硬件通道 |
| 返回值 | ACount:缓存帧数 |
| 示例 | app = win32com.client.Dispatch("TSMaster.TSApplication")<br>com = app.TSCOM()<br>com.fifo_read_fastlin_rx_buffer_frame_count(0, ACount); |

## 10.97 flexray_rbs_start

| | |
|---|---|
| 函数名称 | void flexray_rbs_start(); |
| 功能介绍 | 开启 FlexRay RBS 引擎 |
| 调用位置 | |
| 输入参数 | 无 |
| 返回值 | 无 |
| 示例 | flexray_rbs_start(); |

## 10.98 flexray_rbs_stop

| | |
|---|---|
| 函数名称 | void flexray_rbs_stop(); |
| 功能介绍 | 停止 FlexRay RBS 引擎 |
| 调用位置 | |
| 输入参数 | 无 |
| 返回值 | 无 |
| 示例 | flexray_rbs_stop(); |

## 10.99 flexray_rbs_is_running

| | |
|---|---|
| 函数名称 | void flexray_rbs_is_running([out] VARIANT_BOOL* AIsRunning); |
| 功能介绍 | 检查 FlexRay RBS 是否正在运行 |
| 调用位置 | |
| 输入参数 | |
| 返回值 | AIsRunning:FlexRay RBS 仿真运行状态 |
| 示例 | com.flexray_rbs_is_running(true); |

## 10.100 flexray_rbs_configure

| | |
|---|---|
| 函数名称 | void flexray_rbs_configure(<br>[in] VARIANT_BOOL AAutoStart,<br>[in] VARIANT_BOOL AAutoSendOnModification,<br>[in] VARIANT_BOOL AActivateECUSimulation,<br>[in] TTSRBSInitValueOptions AInitValueOptions); |
| 功能介绍 | FlexRay RBS 仿真配置 |
| 调用位置 | |
| 输入参数 | AAutoStart:自动启动 RBS 仿真<br>AAutoSendOnModification:信号修改时发送报文 |

| | |
|---|---|
| | AActivateECUSimulation:ECU 模拟 //有问题<br>AInitValueOptions:信号初始值 |
| 返回值 | 无 |
| 示例 | flexray_rbs_configure(true, true, false,<br>TTSRBSInitValueOptions.rivUseDB); |

## 10.101 flexray_rbs_activate_all_vlusters

| | |
|---|---|
| 函数名称 | void flexray_rbs_activate_all_clusters(<br>[in] VARIANT_BOOL AEnable,<br>[in] VARIANT_BOOL AIncludingChildren); |
| 功能介绍 | 设置激活或停用所有 FlexRay 网络 |
| 调用位置 | |
| 输入参数 | AEnable:使能<br>AIncludingChildren:激活包括子节点 |
| 返回值 | 无 |
| 示例 | flexray_rbs_activate_all_clusters(true, false); |

## 10.102 flexray_rbs_activate_cluster_by_name

| | |
|---|---|
| 函数名称 | void flexray_rbs_activate_cluster_by_name(<br>[in] long AIdxChn,<br>[in] VARIANT_BOOL AEnable,<br>[in] BSTR AClusterName,<br>[in] VARIANT_BOOL AIncludingChildren); |
| 功能介绍 | 是否激活或停用 FlexRay 网络 |
| 调用位置 | |
| 输入参数 | AIdxChn:通道索引<br>AEnable:使能<br>AClusterName:分组名<br>AIncludingChildren:激活包括子节点 |
| 返回值 | 无 |

| | |
|---|---|
| 示例 | flexray_rbs_activate_cluster_by_name(0, true, "PowerTrain",false); |

## 10.103 flexary_rbs_activate_ecu_by_name

| | |
|---|---|
| 函数名称 | void flexray_rbs_activate_ecu_by_name(<br>[in] long AIdxChn,<br>[in] VARIANT_BOOL AEnable,<br>[in] BSTR AClusterName,<br>[in] BSTR AECUName,<br>[in] VARIANT_BOOL AIncludingChildren); |
| 功能介绍 | 是否激活或停用 ECU 节点 |
| 调用位置 | |
| 输入参数 | AIdxChn:通道索引<br>AEnable:使能<br>AClusterName:分组名<br>AECUName:ECU 名<br>AIncludingChildren:激活包括子节点 |
| 返回值 | 无 |
| 示例 | flexray_rbs_activate_ecu_by_name(0, true, "PowerTrain", "BSC", false); |

## 10.104 flexray_rbs_activate_frame_by_name

| | |
|---|---|
| 函数名称 | void flexray_rbs_activate_frame_by_name(<br>[in] long AIdxChn,<br>[in] VARIANT_BOOL AEnable,<br>[in] BSTR AClusterName,<br>[in] BSTR AECUName,<br>[in] BSTR AFrameName); |
| 功能介绍 | 是否激活或停用 FlexRay 报文仿真 |
| 调用位置 | |
| 输入参数 | AIdxChn:通道索引 |

| | |
|---|---|
| | AEnable:使能<br>AClusterName:分组名<br>AECUName:ECU 名<br>AFrameName:报文名 |
| 返回值 | 无 |
| 示例 | flexray_rbs_activate_frame_by_name(0, true, "PowerTrain",<br>"BSC", "BrakeControl"); |

## 10.105 flexray_rbs_get_signal_value_by_element

| | |
|---|---|
| 函数名称 | void flexray_rbs_get_signal_value_by_element(<br>[in] long AIdxChn,<br>[in] BSTR AClusterName,<br>[in] BSTR AECUName,<br>[in] BSTR AFrameName,<br>[in] BSTR ASignalName,<br>[out] double* AValue); |
| 功能介绍 | 使用元素名称从 FlexRay RBS 获取信号实时值 |
| 调用位置 | |
| 输入参数 | AIdxChn:通道索引<br>AClusterName:分组名<br>AECUName:ECU 名<br>AFrameName:报文名<br>ASignalName:信号名 |
| 返回值 | AValue:信号值 |
| 示例 | flexray_rbs_get_signal_value_by_element(0, "PowerTrain", "BSC",<br>"BrakeControl", "BrakePressure", value); |

## 10.106 flexray_rbs_get_signal_value_by_address

| | |
|---|---|
| 函数名称 | void flexray_rbs_get_signal_value_by_address(<br>[in] BSTR ASymbolAddress,<br>[out] double* AValue); |
| 功能介绍 | 使用信号数据库地址从 FlexRay RBS 获取信号实时值 |

| | |
|---|---|
| 调用位置 | |
| 输入参数 | ASymbolAddress:信号地址 |
| 返回值 | AValue:信号值 |
| 示例 | flexray_rbs_get_signal_value_by_address(<br>"0/PowerTrain/BSC/BrakeControl/BrakePressure", value); |

## 10.107 flexray_rbs_set_signal_value_by_element

| | |
|---|---|
| 函数名称 | void flexray_rbs_set_signal_value_by_element(<br>[in] long AIdxChn,<br>[in] BSTR AClusterName,<br>[in] BSTR AECUName,<br>[in] BSTR AFrameName,<br>[in] BSTR ASignalName,<br>[out] double* AValue); |
| 功能介绍 | 使用元素名称从 FlexRay RBS 设置信号实时值 |
| 调用位置 | |
| 输入参数 | AIdxChn:通道索引<br>AClusterName:分组名<br>AECUName:ECU 名<br>AFrameName:报文名<br>ASignalName:信号名 |
| 返回值 | AValue:信号值 |
| 示例 | flexray_rbs_set_signal_value_by_element(0, "PowerTrain", "BSC",<br>"BrakeControl", "BrakePressure",0xff); |

## 10.108 flexray_rbs_set_signal_value_by_address

| | |
|---|---|
| 函数名称 | void flexray_rbs_set_signal_value_by_address(<br>[in] BSTR ASymbolAddress,<br>[out] double* AValue); |
| 功能介绍 | 使用信号数据库地址从 FlexRay RBS 设置信号实时值 |

| | |
|---|---|
| 调用位置 | |
| 输入参数 | ASymbolAddress:信号地址 |
| 返回值 | AValue:信号值 |
| 示例 | flexray_rbs_set_signal_value_by_address(<br>"0/PowerTrain/BSC/BrakeControl/BrakePressure",0xff); |

## 10.109 flexray_rbs_enable

| | |
|---|---|
| 函数名称 | void flexray_rbs_enable([in] VARIANT_BOOL AEnable); |
| 功能介绍 | 临时启用 FlexRay RBS 引擎，此功能用于在 RBS 引擎启动之前或之后对其进行后期配置 |
| 调用位置 | |
| 输入参数 | AEnable:使能 |
| 返回值 | 无 |
| 示例 | app = win32com.client.Dispatch("TSMaster.TSApplication")<br>com = app.TSCOM()<br>com.flexray_rbs_enable(flase); |

## 10.110 flexray_rbs_batch_set_start

| | |
|---|---|
| 函数名称 | void flexray_rbs_batch_set_start(); |
| 功能介绍 | 开始信号批量设置操作，在此调用后，所有信号设置被缓存，直到 can_rbs_batch_set_end 被调用，这确保当一个帧内的多个信号被设置时，只有一个帧被触发 |
| 调用位置 | |
| 输入参数 | 无 |
| 返回值 | 无 |
| 示例 | app = win32com.client.Dispatch("TSMaster.TSApplication")<br>com = app.TSCOM()<br>com.flexray_rbs_batch_set_start(); |

| | |
|---|---|
| | com.flexray_rbs_batch_set_signal("0/cluster1/ecu1/frame1/sgn1", 1);<br>com.flexray_rbs_batch_set_end(); |

## 10.111 flexray_rbs_batch_set_end

| | |
|---|---|
| 函数名称 | void flexray_rbs_batch_set_end(); |
| 功能介绍 | 停止信号批量设置操作，在此调用之后，所有缓存的信号都被更新，这确保了当一个帧中的多个信号被设置时，只有一个帧被触发 |
| 调用位置 | |
| 输入参数 | 无 |
| 返回值 | 无 |
| 示例 | app = win32com.client.Dispatch("TSMaster.TSApplication")<br>com = app.TSCOM()<br>com.flexray_rbs_batch_set_start();<br>com.flexray_rbs_batch_set_signal("0/cluster1/ecu1/frame1/sgn1", 1);<br>com.flexray_rbs_batch_set_end(); |

## 10.112 flexray_rbs_batch_set_signal

| | |
|---|---|
| 函数名称 | void flexray_rbs_batch_set_signal(<br>[in] BSTR AAddr,<br>[in] double AValue); |
| 功能介绍 | 以批量处理模式设置 FlexRay RBS 中的信号 |
| 调用位置 | |
| 输入参数 | AAddr:信号地址<br>AValue:信号值 |
| 返回值 | 无 |
| 示例 | app = win32com.client.Dispatch("TSMaster.TSApplication")<br>com = app.TSCOM()<br>com.flexray_rbs_batch_set_signal("0/cluster1/ecu1/frame1/sgn1", 1); |

## 10.113 flexray_rbs_set_frame_dircetion

| | |
|---|---|
| 函数名称 | void flexray_rbs_set_frame_direction(<br>[in] long AIdxChn,<br>[in] VARIANT_BOOL AIsTx,<br>[in] BSTR AClusterName,<br>[in] BSTR AECUName,<br>[in] BSTR AFrameName); |
| 功能介绍 | 将 ECU 的帧设置为 tx 或 rx 帧 |
| 调用位置 | |
| 输入参数 | AIdxChn:通道索引<br>AIsTx:发送帧还是接收帧<br>AClusterName:分组名<br>AECUName:ECU 名<br>AFrameName:报文名 |
| 返回值 | 无 |
| 示例 | app = win32com.client.Dispatch("TSMaster.TSApplication")<br>com = app.TSCOM()<br>com.flexray_rbs_set_frame_direction(0, true, "PowerTrain", "BSC", ""); |

## 10.114 flexray_rbs_set_normal_signal

| | |
|---|---|
| 函数名称 | void flexray_rbs_set_normal_signal([in] BSTR ASymbolAddress); |
| 功能介绍 | 将信号设置为正常信号 |
| 调用位置 | |
| 输入参数 | ASymbolAddress:信号地址 |
| 返回值 | 无 |
| 示例 | com = app.TSCOM()<br>com.flexray_rbs_set_normal_signal(<br>"0/PowerTrain/BSC/BrakeControl/BrakePressure"); |

## 10.115 flexray_rbs_set_rc_signal

| | |
|---|---|
| 函数名称 | void flexray_rbs_set_rc_signal([in] BSTR ASymbolAddress); |
| 功能介绍 | 将信号设置为滚动计数器信号 |
| 调用位置 | |
| 输入参数 | ASymbolAddress:信号地址 |
| 返回值 | 无 |
| 示例 | app = win32com.client.Dispatch("TSMaster.TSApplication")<br>com = app.TSCOM()<br>com.flexray_rbs_set_rc_signal_with_limit(<br>"0/PowerTrain/BSC/BrakeControl/BrakePressure"); |

## 10.116 flexray_rbs_set_rc_signal_with_limit

| | |
|---|---|
| 函数名称 | void flexray_rbs_set_rc_signal_with_limit(<br>[in] BSTR ASymbolAddress,<br>[in] long ALowerLimit,<br>[in] long AUpperLimit); |
| 功能介绍 | 将信号设置为滚动计数器信号 |
| 调用位置 | |
| 输入参数 | ASymbolAddress:信号地址<br>ALowerLimit:最小值<br>AUpperLimit:最大值 |
| 返回值 | 无 |
| 示例 | com = app.TSCOM()<br>com.flexray_rbs_set_rc_signal_with_limit(<br>"0/PowerTrain/BSC/BrakeControl/BrakePressure",0,14); |

## 10.117 flexray_rbs_set_rc_signal

| | |
|---|---|
| 函数名称 | void flexray_rbs_set_crc_signal( |

| | |
|---|---|
| | [in] BSTR ASymbolAddress,<br>[in] BSTR AAlgorithmName,<br>[in] long AIdxByteStart,<br>[in] long AByteCount); |
| 功能介绍 | 将信号设置为 CRC 信号 |
| 调用位置 | |
| 输入参数 | ASymbolAddress:信号地址<br>AAlgorithmName:小程序库算法函数名<br>AIdxByteStart:保护起始字节<br>AByteCount:保护的字节数 |
| 返回值 | 无 |
| 示例 | app = win32com.client.Dispatch("TSMaster.TSApplication")<br>com = app.TSCOM()<br>com.flexray_rbs_set_crc_signal(<br>"0/PowerTrain/Engine/EngineData/EngForce","crc.crc8",0, 7); |

## 10.118 flexray_start_net

| | |
|---|---|
| 函数名称 | void flexray_start_net(<br>[in] long AIdxChn,<br>[in] long ATimeoutMs); |
| 功能介绍 | 启用指定的 flexray 框架 |
| 调用位置 | |
| 输入参数 | AIdxChn:通道索引<br>ATimeoutMs:超时时间 |
| 返回值 | 无 |
| 示例 | app = win32com.client.Dispatch("TSMaster.TSApplication")<br>com = app.TSCOM()<br>com.flexray_start_net(0, 1000); |

## 10.119 flexray_stop_net

| | |
|---|---|
| 函数名称 | void flexray_stop_net(<br>[in] long AIdxChn,<br>[in] long ATimeoutMs); |
| 功能介绍 | 停止指定的 flexray 框架 |
| 调用位置 | |
| 输入参数 | AIdxChn:通道索引<br>ATimeoutMs:超时时间 |
| 返回值 | 无 |
| 示例 | app = win32com.client.Dispatch("TSMaster.TSApplication")<br>com = app.TSCOM()<br>com.flexray_stop_net(0, 1000); |

## 10.120 transmit_lin_wakeup_async

| | |
|---|---|
| 函数名称 | void transmit_lin_wakeup_async(<br>[in] long AIdxChn,<br>[in] long AWakeupLength,<br>[in] long AWakeupIntervalTime,<br>[in] long AWakeupTimes); |
| 功能介绍 | 发送 LIN 唤醒帧 |
| 调用位置 | |
| 输入参数 | AIdxChn:通道索引<br>AWakeupLength:唤醒长度<br>AWakeupIntervalTime:唤醒间隔时间<br>AWakeupTimes:唤醒时间 |
| 返回值 | 无 |
| 示例 | app = win32com.client.Dispatch("TSMaster.TSApplication")<br>com = app.TSCOM()<br>com.transmit_lin_wakeup_async(0, 5, 150, 10); |

## 10.121 transmit_lin_gotosleep_async

| | |
|---|---|
| 函数名称 | void transmit_lin_gotosleep_async([in] long AIdxChn); |
| 功能介绍 | 发送 LIN 休眠帧 |
| 调用位置 | |
| 输入参数 | AIdxChn:通道索引 |
| 返回值 | 无 |
| 示例 | transmit_lin_gotosleep_async(0); |

## 10.122 show_main_form

| | |
|---|---|
| 函数名称 | void show_main_form(); |
| 功能介绍 | 显示 TSMaster 窗体 |
| 调用位置 | |
| 输入参数 | 无 |
| 返回值 | 无 |
| 示例 | show_main_form(); |

## 10.123 hide_main_form

| | |
|---|---|
| 函数名称 | void hide_main_form(); |
| 功能介绍 | 隐藏 TSMaster 窗体 |
| 调用位置 | |
| 输入参数 | 无 |
| 返回值 | 无 |
| 示例 | hide_main_form(); |

## 10.124 load_project

| | |
|---|---|
| 函数名称 | void load_project( [in] BSTR AFileName,<br>[in] VARIANT_BOOL ADiscardCurrent); |
| 功能介绍 | 加载 TSMaster 工程 |
| 调用位置 | |
| 输入参数 | AFileName:工程路径<br>ADiscardCurrent:是否退出当前工程 |
| 返回值 | 无 |
| 示例 | load_project("C:\\Users\\LIN\\Desktop\\1", true) |

## 10.125 create_project

| | |
|---|---|
| 函数名称 | void create_project([in] VARIANT_BOOL ADiscardCurrent); |
| 功能介绍 | 创建新的 TSMaster 工程 |
| 调用位置 | |
| 输入参数 | ADiscardCurrent:是否退出当前工程 |
| 返回值 | 无 |
| 示例 | create_project(true); |

## 10.126 save_project

| | |
|---|---|
| 函数名称 | void save_project([in] BSTR AFileName); |
| 功能介绍 | 保存工程 |
| 调用位置 | |
| 输入参数 | AFileName:保存路径 |
| 返回值 | 无 |
| 示例 | save_project("C:/Users/LIN/Desktop/2"); |

## 10.127 show_tab_by_index

| | |
|---|---|
| 函数名称 | void show_tab_by_index([in] long AIndex); |
| 功能介绍 | 根据索引切换页面 |
| 调用位置 | |
| 输入参数 | AIndex:页面索引 |
| 返回值 | 无 |
| 示例 | show_tab_by_index(2); |

## 10.128 show_tab_by_name

| | |
|---|---|
| 函数名称 | void show_tab_by_name([in] BSTR AName); |
| 功能介绍 | 根据页面名切换页面 |
| 调用位置 | |
| 输入参数 | AName:页面名 |
| 返回值 | |
| 示例 | show_tab_by_name("页面 1"); |

## 10.129 get_mp_list

| | |
|---|---|
| 函数名称 | void get_mp_list([out] BSTR* AList); |
| 功能介绍 | 获取小程序库列表 |
| 调用位置 | |
| 输入参数 | 无 |
| 返回值 | AList:小程序库列表 |
| 示例 | get_mp_list( funList); |

## 10.130 get_mp_function_list

| | |
|---|---|
| 函数名称 | void get_mp_function_list(<br>[in] BSTR AMpName,<br>[out] BSTR* AFuncList); |
| 功能介绍 | 获取小程序库函数列表 |
| 调用位置 | |
| 输入参数 | AMpName:小程序库 |
| 返回值 | AList:函数列表 |
| 示例 | get_mp_function_list("crc.mp", funList); |

## 10.131 get_mp_function_prototype

| | |
|---|---|
| 函数名称 | void get_mp_function_prototype(<br>[in] BSTR AMpName,<br>[in] BSTR AFuncName,<br>[out] BSTR* APrototype); |
| 功能介绍 | 获取小程序库函数原型 |
| 调用位置 | |
| 输入参数 | AMpName:小程序库<br>AFuncName:函数名 |
| 返回值 | APrototype:函数原型 |
| 示例 | get_mp_function_prototype("crc.mp","CRC8",proType); |

## 10.132 dynamic_invoke

| | |
|---|---|
| 函数名称 | void dynamic_invoke(<br>[in] BSTR AMpName,<br>[in] BSTR AFuncName,<br>[in] BSTR AInParameters, |

| | |
|---|---|
| | [out] BSTR* AOutParameters); |
| 功能介绍 | 动态调用小程序库函数 |
| 调用位置 | |
| 输入参数 | AMpName:小程序库<br>AFuncName:函数名<br>AInParameters:输入参数 |
| 返回值 | AOutParameters:输出参数 |
| 示例 | dynamic_invoke("ccode8212.mp", "NewCustom_Function1", "1",outPar); |

## 10.133 load_mp

| | |
|---|---|
| 函数名称 | void load_mp([in] BSTR AMpFileName); |
| 功能介绍 | 加载小程序库 |
| 调用位置 | |
| 输入参数 | AMpFileName:小程序库名称 |
| 返回值 | 无 |
| 示例 | load_mp("c:\\tsmaster\\tsmaster\\bin\\rtluidiagnostics.bpl"); |

## 10.134 unload_mp

| | |
|---|---|
| 函数名称 | void unload_mp([in] BSTR AMpName); |
| 功能介绍 | 卸载小程序库 |
| 调用位置 | |
| 输入参数 | AMpName:小程序库名 |
| 返回值 | 无 |
| 示例 | unload_mp("rtluidiagnostics.bpl"); |

## 10.135 unload_all_mps

| | |
|---|---|
| 函数名称 | void unload_all_mps(); |
| 功能介绍 | 卸载全部小程序库 |
| 调用位置 | |
| 输入参数 | 无 |
| 返回值 | 无 |
| 示例 | unload_all_mps(); |

## 10.136 call_system_api

| | |
|---|---|
| 函数名称 | long call_system_api(<br>[in] BSTR AFuncName,<br>[in] VARIANT AInArgs,<br>[out] VARIANT* AOutArgs); |
| 功能介绍 | 调用系统 API |
| 调用位置 | |
| 输入参数 | AFuncName:函数名<br>AInArgs:输入参数 |
| 返回值 | AOutArgs:输出参数 |
| 示例 | |

## 10.137 call_library_api

| | |
|---|---|
| 函数名称 | long call_library_api(<br>[in] BSTR AFuncName,<br>[in] VARIANT AInArgs,<br>[out] VARIANT* AOutArgs); |
| 功能介绍 | 调用库 API |
| 调用位置 | |

| | |
|---|---|
| 输入参数 | AFuncName:函数名<br>AInArgs:输入参数 |
| 返回值 | AOutArgs:输出参数 |
| 示例 | |

## 10.138 add_online_replay_config

| | |
|---|---|
| 函数名称 | void add_online_replay_config(<br>[in] BSTR AFileName,<br>[out] long* AIndex); |
| 功能介绍 | 添加在线回放引擎 |
| 调用位置 | |
| 输入参数 | AFileName:回放文件名 |
| 返回值 | AIndex:在线回放引擎索引 |
| 示例 | log.add_online_replay_config(<br>"C:\\Users\\LIN\\Desktop\\1\\Logging\\Bus\\放电.blf",out index); |

## 10.139 set_online_replay_config

| | |
|---|---|
| 函数名称 | void set_online_replay_config(<br>[in] long AIndex,<br>[in] BSTR AName,<br>[in] BSTR AFileName,<br>[in] VARIANT_BOOL AAutoStart,<br>[in] VARIANT_BOOL AIsRepetitiveMode,<br>[in]TTSOnlineReplayTimingMode<br>AStartTimingMode,<br>[in] long AStartDelayTimeMs,<br>[in] VARIANT_BOOL ASendTx,<br>[in] VARIANT_BOOL ASendRx,<br>[in] BSTR AMappings); |
| 功能介绍 | 设置在线回放引擎 |

| | |
|---|---|
| 调用位置 | |
| 输入参数 | AIndex:在线回放引擎索引<br>AName:显示名<br>AFileName:文件名<br>AAutoStart:自启动激活<br>AIsRepetitiveMode:循环模式<br>AStartTimingMode:启动时序模式<br>AStartDelayTimeMs:启动延时时间<br>ASendTx:回放发送报文<br>ASendRx:回放接收报文<br>AMappings:回放通道 |
| 返回值 | 无 |
| 示例 | set_online_replay_config(index,"test",<br>"C:\\Users\\LIN\\Desktop\\1\\Logging\\Bus\\放电.blf",<br>false,false, TTSOnlineReplayTimingMode.ortImmediately,0,true,<br>false,"0,1"); |

## 10.140 get_online_replay_count

| | |
|---|---|
| 函数名称 | void get_online_replay_count([out] long* ACount); |
| 功能介绍 | 获取在线回放引擎数 |
| 调用位置 | |
| 输入参数 | |
| 返回值 | ACount:在线回放引擎数 |
| 示例 | get_online_replay_count(count); |

## 10.141 get_online_replay_config

| | |
|---|---|
| 函数名称 | void get_online_replay_config(<br>[in] long AIndex,<br>[out] BSTR* AName, |

| | |
|---|---|
| | [out] BSTR* AFileName,<br>[out] VARIANT_BOOL* AAutoStart,<br>[out] VARIANT_BOOL* AIsRepetitiveMode,<br>[out]TTSOnlineReplayTimingMode*<br>AStartTimingMode,<br>[out] long* AStartDelayTimeMs,<br>[out] VARIANT_BOOL* ASendTx,<br>[out] VARIANT_BOOL* ASendRx,<br>[out] BSTR* AMappings); |
| 功能介绍 | 获取在线回放引擎配置 |
| 调用位置 | |
| 输入参数 | AIndex:在线回放引擎索引 |
| 返回值 | AName:显示名<br>AFileName:文件名<br>AAutoStart:自启动激活<br>AIsRepetitiveMode:循环模式<br>AStartTimingMode:启动时序模式<br>AStartDelayTimeMs:启动延时时间<br>ASendTx:回放发送报文<br>ASendRx:回放接收报文<br>AMappings:回放通道 |
| 示例 | |

## 10.142 del_online_replay_config

| | |
|---|---|
| 函数名称 | void del_online_replay_config([in] long AIndex); |
| 功能介绍 | 按索引删除在线回放引擎 |
| 调用位置 | |
| 输入参数 | AIndex:在线回放引擎索引 |
| 返回值 | 无 |
| 示例 | del_online_replay_config(index); |

## 10.143 del_online_replay_configs

| | |
|---|---|
| 函数名称 | void del_online_replay_configs(); |
| 功能介绍 | 删除全部在线回放引擎 |
| 调用位置 | |
| 输入参数 | 无 |
| 返回值 | 无 |
| 示例 | del_online_replay_configs(); |

## 10.144 start_online_replay

| | |
|---|---|
| 函数名称 | void start_online_replay([in] long AIndex); |
| 功能介绍 | 按索引启动在线回放引擎 |
| 调用位置 | |
| 输入参数 | AIndex：在线回放引擎索引 |
| 返回值 | 无 |
| 示例 | start_online_replay(index); |

## 10.145 start_online_replays

| | |
|---|---|
| 函数名称 | void start_online_replays(); |
| 功能介绍 | 启动所有在线回放引擎 |
| 调用位置 | |
| 输入参数 | 无 |
| 返回值 | 无 |
| 示例 | start_online_replays(); |

## 10.146 pause_online_replay

| | |
|---|---|
| 函数名称 | void pause_online_replay([in] long AIndex); |
| 功能介绍 | 按索引暂停在线回放引擎 |
| 调用位置 | |
| 输入参数 | AIndex:在线回放引擎索引 |
| 返回值 | 无 |
| 示例 | pause_online_replay(index); |

## 10.147 pause_online_replays

| | |
|---|---|
| 函数名称 | void pause_online_replays(); |
| 功能介绍 | 暂停所有在线回放引擎 |
| 调用位置 | |
| 输入参数 | 无 |
| 返回值 | 无 |
| 示例 | pause_online_replays(); |

## 10.148 stop_online_replay

| | |
|---|---|
| 函数名称 | void stop_online_replay([in] long AIndex); |
| 功能介绍 | 按索引停止在线回放引擎 |
| 调用位置 | |
| 输入参数 | AIndex:在线回放引擎索引 |
| 返回值 | 无 |
| 示例 | stop_online_replay(index); |

## 10.149 stop_online_replays

| | |
|---|---|
| 函数名称 | void stop_online_replays(); |
| 功能介绍 | 停止所有在线回放引擎 |
| 调用位置 | |
| 输入参数 | 无 |
| 返回值 | 无 |
| 示例 | stop_online_replays(); |

## 10.150 get_online_replay_status

| | |
|---|---|
| 函数名称 | void get_online_replay_status(int AIndex,<br>out TTSOnlineReplayStatus AStatus,<br>out float AProgressPercent100); |
| 功能介绍 | 获取在线回放引擎状态 |
| 调用位置 | |
| 输入参数 | AIndex:在线回放引擎索引 |
| 返回值 | AStatus:回放状态<br>AProgressPercent100:回放百分比 |
| 示例 | |

## 10.151 load_can_db

| | |
|---|---|
| 函数名称 | void load_can_db(<br>[in] BSTR ADBCFile,<br>[in] BSTR ASupportedChannels,<br>[out] long* AId); |
| 功能介绍 | 把数据库加载到指定的通道上，并获取加载数据库的编号 |

| | |
|---|---|
| 调用位置 | 该函数必须在应用处于断开状态（执行 tsapp_connect 之前或者执行了 tsapp_disconnect 之后）调用。当应用处于连接状态的时候，不能删除， 加载数据库文件。 |
| 输入参数 | ADBCFile:DBC 路径<br>ASupportedChannels：加载通道 |
| 返回值 | AId：加载数据库成功过后，返回系统为该数据库分配 的唯一编号 |
| 示例 | load_can_db("C:\\TSMaster\\bin\\Data\\Demo\\Databases \\CAN_FD_Powertrain.dbc","0,1",dbcId); |

## 10.152 unload_can_db

| | |
|---|---|
| 函数名称 | void unload_can_db([in] long AId); |
| 功能介绍 | 卸载指定编号的数据库文件 |
| 调用位置 | 该函数必须在应用处于断开状态（执行 tsapp_connect 之前或者执行了 tsapp_disconnect 之后）调用。当应用处于连接状态的时候，不能删除， 加载数据库文件。 |
| 输入参数 | AId：加载数据库成功过后，返回系统为该数据库分配的唯一编号 |
| 返回值 | |
| 示例 | unload_can_db(dbcId); |

## 10.153 unload_can_dbs

| | |
|---|---|
| 函数名称 | void unload_can_dbs(); |
| 功能介绍 | 卸载所有已经加载的 DBC 文件 |
| 调用位置 | 该函数必须在应用处于断开状态（执行 tsapp_connect 之前或者执行了 tsapp_disconnect 之后）调用。当应用处于连接状态的时候，不能删除， 加载数据库文件。 |
| 输入参数 | 无 |
| 返回值 | 无 |
| 示例 | unload_can_dbs(); |

## 10.154 get_can_db_count

| | |
|---|---|
| 函数名称 | void get_can_db_count([out] long* ACount); |
| 功能介绍 | 获取加载的 CAN 数据库计数 |
| 调用位置 | |
| 输入参数 | 无 |
| 返回值 | ACount:加载的 CAN 数据计数 |
| 示例 | get_can_db_count(); |

## 10.155 get_can_db_id

| | |
|---|---|
| 函数名称 | void get_can_db_id(<br>[in] long AIndex,<br>[out] long* AId); |
| 功能介绍 | 按索引获取 CAN 数据库 ID |
| 调用位置 | |
| 输入参数 | AIndex:加载的 CAN 数据索引 |
| 返回值 | AId:数据库 ID |
| 示例 | get_can_db_id(0, dbcId); |

## 10.156 get_can_db_info

| | |
|---|---|
| 函数名称 | long get_can_db_info(<br>[in] long ADatabaseId,<br>[in] long AType,<br>[in] long AIndex,<br>[in] long ASubIndex,<br>[out] BSTR* AValue); |

| | |
|---|---|
| 功能介绍 | 获取数据库信息 |
| 调用位置 | |
| 输入参数 | ADatabaseId:数据库 ID<br>AType:信息类型<br>AIndex:索引<br>ASubIndex:子索引 |
| 返回值 | AValue:数据库信息 |
| 示例 | |

## 10.157 set_signal_value_can

| | |
|---|---|
| 函数名称 | void set_signal_value_can(<br>[in] PCAN AInCAN,<br>[in] BSTR AMsgName,<br>[in] BSTR ASgnName,<br>[in] double AValue,<br>[out] SAFEARRAY(char)* ADatas); |
| 功能介绍 | 根据 Message 名称，信号名称，把信号值更新到 CAN 报文中。 |
| 调用位置 | 加载数据库过后，任意位置（应用连接状态， 断开状态） 都可以调用此函数。 |
| 输入参数 | "ACAN">原始 CAN 报文(引用)<br>"AMsgName"：报文名称<br>"ASgnName"：信号名称<br>"AValue"：信号值 |
| 返回值 | ADatas：报文数据 |
| 示例 | msgName = "message1"<br>singalName = "signalName"<br>app = win32com.client.Dispatch("TSMaster.TSApplication")<br>com = app.TSCOM()<br>value = 0<br>com.set_signal_value_canfd(ACAN, msgName, singalName, value) |

## 10.158 get_signal_value_can

| | |
|---|---|
| 函数名称 | void get_signal_value_can(<br>[in] PCAN AInCAN,<br>[in] BSTR AMsgName,<br>[in] BSTR ASgnName,<br>[out] double* AValue); |
| 功能介绍 | 根据 Message 名称，信号名称，从 CAN 报文中读取信号值。 |
| 调用位置 | 加载数据库过后，任意位置（应用连接状态， 断开状态） 都可以调用此函数。 |
| 输入参数 | "ACAN"：原始 CAN 报文(引用)<br>"AMsgName"：报文名称<br>"ASgnName"：信号名称 |
| 返回值 | "AValue"：信号值 |
| 示例 | msgName = "message1"<br>singalName = "signalName"<br>app = win32com.client.Dispatch("TSMaster.TSApplication")<br>com = app.TSCOM()<br>com.get_signal_value_can(ACAN, msgName, singalName, value) |

## 10.159 set_signal_value_canfd

| | |
|---|---|
| 函数名称 | void set_signal_value_canfd(<br>[in] PCANFD AInCANFD,<br>[in] BSTR AMsgName,<br>[in] BSTR ASgnName,<br>[in] double AValue,<br>[out] SAFEARRAY(char)* ADatas); |
| 功能介绍 | 根据 Message 名称，信号名称，把信号值更新到 CANFD 报文中 |
| 调用位置 | 加载数据库过后，任意位置（应用连接状态， 断开状态） 都可以调用此函数。 |
| 输入参数 | "ACANFD"：原始 CANFD 报文<br>"AMsgName"：报文名称<br>"ASgnName"：信号名称 |

| | |
|---|---|
| | "AValue"：信号值 |
| 返回值 | ADatas：报文数据 |
| 示例 | msgName = "message1"<br>singalName = "signalName"<br>app = win32com.client.Dispatch("TSMaster.TSApplication")<br>com = app.TSCOM()<br>com.set_signal_value_canfd(ACANFD, msgName, singalName, value) |

## 10.160 get_signal_value_canfd

| | |
|---|---|
| 函数名称 | void get_signal_value_canfd(<br>[in] PCANFD AInCANFD,<br>[in] BSTR AMsgName,<br>[in] BSTR ASgnName,<br>[out] double* AValue); |
| 功能介绍 | 根据 Message 名称，信号名称，从 CANFD 报文中获取信号值 |
| 调用位置 | 加载数据库过后，任意位置（应用连接状态， 断开状态） 都可以调用此函数。 |
| 输入参数 | "ACANFD"：原始 CANFD 报文<br>"AMsgName"：报文名称<br>"ASgnName"：信号名称 |
| 返回值 | "AValue"：信号值 |
| 示例 | msgName = "message1"<br>singalName = "signalName"<br>app = win32com.client.Dispatch("TSMaster.TSApplication")<br>com = app.TSCOM()<br>com.get_signal_value_canfd(ACANFD, msgName, singalName, value) |

## 10.161 set_signal_vale_can_verbose

| | |
|---|---|
| 函数名称 | void set_signal_value_can_verbose(<br>[in] long AIdxChn,<br>[in] BSTR AInDatas, |

| | |
|---|---|
| | [in] BSTR AMsgName,<br>[in] BSTR ASgnName,<br>[in] double AValue,<br>[out] BSTR* AOutDatas); |
| 功能介绍 | 设置 CAN 信号值 |
| 调用位置 | |
| 输入参数 | AIdxChn:通道索引<br>AInDatas:报文数据<br>AMsgName:报文名<br>ASgnName:信号名<br>AValue:信号值 |
| 返回值 | AOutDatas：输出数据 |
| 示例 | |

## 10.162 get_signal_value_can_verbose

| | |
|---|---|
| 函数名称 | void get_signal_value_can_verbose(<br>[in] long AIdxChn,<br>[in] BSTR AInDatas,<br>[in] BSTR AMsgName,<br>[in] BSTR ASgnName,<br>[out] double* AValue); |
| 功能介绍 | 获取 CANFD 信号值 |
| 调用位置 | |
| 输入参数 | AIdxChn：通道索引<br>AInDatas:报文数据<br>AMsgName:报文名<br>ASgnName:信号名 |
| 返回值 | AValue:信号值 |
| 示例 | |

## 10.163 set_signal_value_canfd_verbose

| | |
|---|---|
| 函数名称 | void set_signal_value_canfd_verbose(<br>[in] long AIdxChn,<br>[in] BSTR AInDatas,<br>[in] BSTR AMsgName,<br>[in] BSTR ASgnName,<br>[in] double AValue,<br>[out] BSTR* AOutDatas); |
| 功能介绍 | 设置 CANFD 信号值 |
| 调用位置 | |
| 输入参数 | AIdxChn:通道索引<br>AInDatas:报文数据<br>AMsgName:报文名<br>ASgnName:信号名<br>AValue:信号值 |
| 返回值 | AOutDatas：输出数据 |
| 示例 | |

## 10.164 get_signal_value_canfd_verbose

| | |
|---|---|
| 函数名称 | void get_signal_value_canfd_verbose(<br>[in] long AIdxChn,<br>[in] BSTR AInDatas,<br>[in] BSTR AMsgName,<br>[in] BSTR ASgnName,<br>[out] double* AValue); |
| 功能介绍 | 获取 CANFD 信号值 |
| 调用位置 | |
| 输入参数 | AIdxChn:通道索引<br>AInDatas: 报文数据<br>AMsgName:报文名<br>ASgnName:信号名 |

| | |
|---|---|
| 返回值 | AValue 信号值 |
| 示例 | |

## 10.165 load_flexray_db

| | |
|---|---|
| 函数名称 | void load_flexray_db(<br>[in] BSTR AFRFile,<br>[in] BSTR ASupportedChannels,<br>[out] long* AId); |
| 功能介绍 | 加载 FlexRay 数据库 |
| 调用位置 | |
| 输入参数 | AFRFile:数据库路径<br>ASupportedChannels:加载通道 |
| 返回值 | AId:数据库 ID |
| 示例 | load_flexray_db("C:\\TSMaster\\bin\\Data\\Demo\\Databases\\<br>PowerTrain_v2.xml","0,1",frId); |

## 10.166 unload_flexray_db

| | |
|---|---|
| 函数名称 | void unload_flexray_db([in] long AId); |
| 功能介绍 | 卸载 FlexRay 数据库 |
| 调用位置 | |
| 输入参数 | AId:数据库 ID |
| 返回值 | 无 |
| 示例 | unload_flexray_db(frId); |

## 10.167 unload_flexray_dbs

| | |
|---|---|
| 函数名称 | void unload_flexray_dbs(); |

| | |
|---|---|
| 功能介绍 | 卸载所有 FlexRay 数据库 |
| 调用位置 | |
| 输入参数 | 无 |
| 返回值 | 无 |
| 示例 | unload_flexray_dbs(); |

## 10.168 get_flexray_db_count

| | |
|---|---|
| 函数名称 | void get_flexray_db_count([out] long* ACount); |
| 功能介绍 | 获取加载的 FlexRay 数据库计数 |
| 调用位置 | |
| 输入参数 | |
| 返回值 | 获取加载的 FlexRay 数据库计数 |
| 示例 | get_flexray_db_count(count); |

## 10.169 get_flexray_db_properties_by_address

| | |
|---|---|
| 函数名称 | void get_flexray_db_properties_by_address(<br>[in] BSTR AAddr,<br>[out] long* ADBIndex,<br>[out] long* ASignalCount,<br>[out] long* AFrameCount,<br>[out] long* AECUCount,<br>[out] int64* ASupportedChannelMask,<br>[out] BSTR* AName,<br>[out] BSTR* AComment); |
| 功能介绍 | 按地址获取 FlexRay 数据库信息 |
| 调用位置 | |
| 输入参数 | AAddr:数据库地址 |

| | |
|---|---|
| 返回值 | ADBIndex:数据库索引<br>ASignalCount:信号数量<br>AFrameCount:报文数量<br>AECUCount:ECU 数量<br>ASupportedChannelMask:加载数据库的通道掩码<br>//1011 表示通道 124 都加载了数据库<br>AName:数据库名<br>AComment:注释 |
| 示例 | get_flexray_db_properties_by_address("0/PowerTrain", out index,<br>sigalCount, frameCount, ecuCount, channel,name, comment); |

## 10.170 ger_flexray_db_properties_by_index

| | |
|---|---|
| 函数名称 | void get_flexray_db_properties_by_index(<br>[in] long ADBIndex,<br>[out] long* ASignalCount,<br>[out] long* AFrameCount,<br>[out] long* AECUCount,<br>[out] int64* ASupportedChannelMask,<br>[out] BSTR* AName,<br>[out] BSTR* AComment); |
| 功能介绍 | 按索引获取 FlexRay 数据库信息 |
| 调用位置 | |
| 输入参数 | ADBIndex:数据库索引 |
| 返回值 | ASignalCount:信号数量<br>AFrameCount:报文数量<br>AECUCount:ECU 数量<br>ASupportedChannelMask:加载数据库的通道掩码<br>//1011 表示通道 124 都加载了数据库<br>AName:数据库名<br>AComment:注释 |
| 示例 | get_flexray_db_properties_by_index(index, sigalCount,<br>frameCount, ecuCount, channel, name, comment); |

## 10.171 get_flexray_ecu_properties_by_address

| | |
|---|---|
| 函数名称 | void get_flexray_ecu_properties_by_address(<br>[in] BSTR AAddr,<br>[out] long* ADBIndex,<br>[out] long* AECUIndex,<br>[out] long* ATxFrameCount,<br>[out] long* ARxFrameCount,<br>[out] BSTR* AName,<br>[out] BSTR* AComment); |
| 功能介绍 | 按地址获取 ECU 信息 |
| 调用位置 | |
| 输入参数 | AAddr:ECU 地址 |
| 返回值 | ADBIndex:数据库索引<br>AECUIndex:ECU 索引<br>ATxFrameCount:发送报文数量<br>ARxFrameCount:接收报文数量<br>AName:ECU 名<br>AComment:注释 |
| 示例 | get_flexray_ecu_properties_by_address("0/PowerTrain/BLU",<br>index, ecuIndex, txFromCount, rxFromCount,name, comment); |

## 10.172 get_flexray_ecu_properties_by_index

| | |
|---|---|
| 函数名称 | void get_flexray_ecu_properties_by_index(<br>[in] long ADBIndex,<br>[in] long AECUIndex,<br>[out] long* ATxFrameCount,<br>[out] long* ARxFrameCount,<br>[out] BSTR* AName,<br>[out] BSTR* AComment); |
| 功能介绍 | 按索引获取 ECU 信息 |
| 调用位置 | |

| | |
|---|---|
| 输入参数 | ADBIndex:数据库索引<br>AECUIndex:ECU 索引 |
| 返回值 | ATxFrameCount:发送报文数量<br>ARxFrameCount:接收报文数量<br>AName:ECU 名<br>AComment:注释 |
| 示例 | get_flexray_ecu_properties_by_index(index, ecuIndex,<br>txFromCount, rxFromCount, name, comment); |

## 10.173 get_flexray_frame_properties_by_address

| | |
|---|---|
| 函数名称 | void get_flexray_frame_properties_by_address(<br>[in] BSTR AAddr,<br>[out] long* ADBIndex,<br>[out] long* AECUIndex,<br>[out] long* AFrameIndex,<br>[out] VARIANT_BOOL* AIsTx,<br>[out] long* AFRChannelMask,<br>[out] long* AFRBaseCycle,<br>[out] long* AFRCycleRepetition,<br>[out] VARIANT_BOOL* AFRIsStartupFrame,<br>[out] long* AFRSlotId,<br>[out] int64* AFRCycleMask,<br>[out] long* ASignalCount,<br>[out] long* AFRDLC,<br>[out] BSTR* AName,<br>[out] BSTR* AComment); |
| 功能介绍 | 按地址获取 flexray 报文信息 |
| 调用位置 | |
| 输入参数 | AAddr:报文地址 |
| 返回值 | ADBIndex:数据库索引<br>AECUIndex:ECU 索引<br>AFrameIndex:报文索引<br>AIsTx:报文方向 |

| | |
|---|---|
| | AFRChannelMask:FR 通道掩码 AB 通道<br>AFRBaseCycle:BaseCycle<br>AFRCycleRepetition:循环重复<br>AFRIsStartupFrame:启动帧<br>AFRSlotId:SlotID<br>AFRCycleMask:周期掩码<br>ASignalCount:信号数量<br>AFRDLC:数据长度<br>AName:报文名<br>AComment:注释 |
| 示例 | get_flexray_frame_properties_by_address("0//PowerTrain/C//BrakeControl/BrakePressure",index, ecuIndex, frameIndex, framelIndex,true,frChannel,baseCycle,cycleRepetition,true,slotId, cycleMask, sigalCount, 8, name, comment) |

## 10.174 get_flexray_frame_properties_by_index

| | |
|---|---|
| 函数名称 | void get_flexray_frame_properties_by_index(<br>[in] long ADBIndex,<br>[in] long AECUIndex,<br>[in] long AFrameIndex,<br>[in] VARIANT_BOOL AIsTx,<br>[out] long* AFRChannelMask,<br>[out] long* AFRBaseCycle,<br>[out] long* AFRCycleRepetition,<br>[out] VARIANT_BOOL* AFRIsStartupFrame,<br>[out] long* AFRSlotId,<br>[out] int64* AFRCycleMask,<br>[out] long* ASignalCount,<br>[out] long* AFRDLC,<br>[out] BSTR* AName,<br>[out] BSTR* AComment); |
| 功能介绍 | 按索引获取 flexray 报文信息 |
| 调用位置 | |

| | |
|---|---|
| 输入参数 | ADBIndex:数据库索引<br>AECUIndex:ECU 索引<br>AFrameIndex:报文索引<br>AIsTx:报文方向是否为发送 |
| 返回值 | AFRChannelMask:FR 通道掩码 AB 通道<br>AFRBaseCycle:BaseCycle<br>AFRCycleRepetition:循环重复<br>AFRIsStartupFrame:启动帧<br>AFRSlotId:SlotID<br>AFRCycleMask:周期掩码<br>ASignalCount:信号数量<br>AFRDLC:数据长度<br>AName:报文名<br>AComment:注释 |
| 示例 | get_flexray_frame_properties_by_index(index, ecuIndex,<br>frameIndex,true, frChannel, baseCycle, cycleRepetition,true,<br>slotId, cycleMask, sigalCount,8, name, comment); |

## 10.175 get_flexray_signal_properties_by_address

| | |
|---|---|
| 函数名称 | void get_flexray_signal_properties_by_address(<br>[in] BSTR AAddr,<br>[out] long* ADBIndex,<br>[out] long* AECUIndex,<br>[out] long* AFrameIndex,<br>[out] long* ASignalIndex,<br>[out] VARIANT_BOOL* AIsTx,<br>[out] TTSSignalType* ASignalType,<br>[out] TTSFlexRayCompuMethod* ACompuMethod,<br>[out] VARIANT_BOOL* AIsIntel,<br>[out] long* AStartBit,<br>[out] long* AUpdateBit,<br>[out] long* ALength,<br>[out] double* AFactor,<br>[out] double* AOffset, |

| | |
|---|---|
| | [out] double* AInitValue,<br>[out] BSTR* AName,<br>[out] BSTR* AComment);<br>[out] long* AFRDLC,<br>[out] BSTR* AName,<br>[out] BSTR* AComment); |
| 功能介绍 | 按地址获取 flexray 信号信息 |
| 调用位置 | |
| 输入参数 | AAddr:信号地址 |
| 返回值 | ADBIndex:数据库索引<br>AECUIndex:ECU 索引<br>AFrameIndex:报文索引<br>ASignalIndex:信号索引<br>AIsTx:报文方向<br>ASignalType:信号类型<br>ACompuMethod:计算方法<br>AIsIntel:字节序 true:英特尔 false:摩托罗拉<br>AStartBit:起始位<br>AUpdateBit:更新位<br>ALength:长度<br>AFactor:放大因子<br>AOffset:偏移量<br>AInitValue:初始值<br>AName:信号名<br>AComment:注释 |
| 示例 | get_flexray_signal_properties_by_address("0//PowerTrain/C//BrakeControl/BrakePressure",index, ecuIndex, frameIndex, signalIndex,true,signalType,compuMethod,false,startBit,updateBit,lengthFactor,offset,initValue,name,comment) |

## 10.176 get_flexray_signal_properties_by_index

| | |
|---|---|
| 函数名称 | void get_flexray_signal_properties_by_index(<br>[in] long ADBIndex, |

| | |
|---|---|
| | [in] long AECUIndex,<br>[in] long AFrameIndex,<br>[in] long ASignalIndex,<br>[in] VARIANT_BOOL AIsTx,<br>[out] TTSSignalType* ASignalType,<br>[out] TTSFlexRayCompuMethod* ACompuMethod,<br>[out] VARIANT_BOOL* AIsIntel,<br>[out] long* AStartBit,<br>[out] long* AUpdateBit,<br>[out] long* ALength,<br>[out] double* AFactor,<br>[out] double* AOffset,<br>[out] double* AInitValue,<br>[out] BSTR* AName,<br>[out] BSTR* AComment); |
| 功能介绍 | 按索引获取 Flexray 信号信息 |
| 调用位置 | |
| 输入参数 | ADBIndex:数据库索引<br>AECUIndex:ECU 索引<br>AFrameIndex:报文索引<br>ASignalIndex:信号索引<br>AIsTx:报文方向，是否为发送端 |
| 返回值 | ASignalType:信号类型<br>ACompuMethod:计算方法<br>AIsIntel:字节序 true:英特尔 false:摩托罗拉<br>AStartBit:起始位<br>AUpdateBit:更新位<br>ALength:长度<br>AFactor:放大因子<br>AOffset:偏移量<br>AInitValue:初始值<br>AName:信号名<br>AComment:注释 |
| 示例 | get_flexray_signal_properties_by_index(index,ecuIndex,frameIndex, |

| | |
|---|---|
| | signalIndex,true,signalType,compuMethod,false,startBit,updateBit,length<br>Factor,offset,initValue,name,comment) |

## 10.177 get_flexray_db_id

| | |
|---|---|
| 函数名称 | void get_flexray_db_id(<br>[in] long AIndex,<br>[out] long* AId); |
| 功能介绍 | 按索引获取 FlexRay 数据库 ID |
| 调用位置 | |
| 输入参数 | AIndex:数据库索引 |
| 返回值 | AId:数据库 ID |
| 示例 | get_flexray_db_id（index，Id） |

## 10.178 diag_can_create

| | |
|---|---|
| 函数名称 | VARIANT_BOOL diag_can_create(<br>[out] long* pDiagModuleIndex,<br>[in] long AChnIndex,<br>[in] VARIANT_BOOL ASupportFDCAN,<br>[in] long AMaxDLC,<br>[in] long ARequestID,<br>[in] VARIANT_BOOL ARequestIDIsStd,<br>[in] long AResponseID,<br>[in] VARIANT_BOOL AResponseIDIsStd,<br>[in] long AFunctionID,<br>[in] VARIANT_BOOL AFunctionIDIsStd); |
| 功能介绍 | 创建 can 诊断模块 |
| 调用位置 | |
| 输入参数 | pDiagModuleIndex :诊断模块索引<br>AChnIndex :通道索引<br>ASupportFDCAN：支持的 CANFD |

| | |
|---|---|
| | AMaxDLC:最大 DLC 长度<br>ARequestID:请求 ID<br>ARequestIDIsStd:请求 ID 是否标准<br>AResponseID:响应 ID<br>AResponseIDIsStd :响应 ID 是否标准<br>AFunctionID:功能 ID<br>AFunctionIDIsStd :功能 ID 是否标准 |
| 返回值 | ==0：执行成功<br>其他值：查询错误码 |
| 示例 | int pDiagModuleIndex = 0 ;<br>diag_can_create(&pDiagModuleIndex,0,ASupportFDCAN,AMaxDLC,<br>ARequestID,true,AResponseID,true,AFunctionID,true) ; |

## 10.179 diag_set_fdmode

| | |
|---|---|
| 函数名称 | VARIANT_BOOL diag_set_fdmode(<br>[in] long ADiagModuleIndex,<br>[in] VARIANT_BOOL AFDMode,<br>[in] VARIANT_BOOL ASupportBRS,<br>[in] long AMaxDLC); |
| 功能介绍 | 设置 CANFD 模块 |
| 调用位置 | |
| 输入参数 | ADiagModuleIndex:诊断模块索引<br>AFDMode:FD 模式<br>ASupportBRS:BRS 模式<br>AMaxDLC:最大 DLC |
| 返回值 | ==0：执行成功<br>其他值：查询错误码 |
| 示例 | diag_set_fdmode(diagnosticIndex,true,true,15); |

## 10.180 diag_can_delete

| | |
|---|---|
| 函数名称 | VARIANT_BOOL diag_can_delete([in] long ADiagModuleIndex); |

| | |
|---|---|
| 功能介绍 | 删除 can 诊断 |
| 调用位置 | |
| 输入参数 | ADiagModuleIndex：诊断模块索引 |
| 返回值 | ==0：成功<br>其他值：失败 |
| 示例 | diag_can_delete() ; |

## 10.181 diag_can_delete_all

| | |
|---|---|
| 函数名称 | void diag_can_delete_all(); |
| 功能介绍 | 删除所有 can 诊断 |
| 调用位置 | |
| 输入参数 | 无 |
| 返回值 | ==0：成功<br>其他值：失败 |
| 示例 | diag_can_delete_all() ; |

## 10.182 tp_can_request_and_get_response

| | |
|---|---|
| 函数名称 | VARIANT_BOOL tp_can_request_and_get_response(<br>[in] long ADiagModuleIndex,<br>[in] BSTR ARequest,<br>[out] BSTR* AResponse); |
| 功能介绍 | 请求 id 请求数据并获取响应数据 |
| 调用位置 | |
| 输入参数 | ADiagModuleIndex :uds 模块的句柄<br>AReqArray:指向请求数据源的数据指针 |
| 返回值 | AResponse：应答结果 |
| 示例 | byte reqDataArray= {0x10,0x03} ;<br>byte responseArray [10] ; |

| | |
|---|---|
| | int responseArraySize = 10;<br>if(tp_can_request_and_get_response(udsHandle, reqDataArray, 2, responseArray, &responseArraySize) == 0x00)<br>{<br>app.log_text("send diagnostic payload and get response success!",lvlOK) ;<br>} |

## 10.183 tp_can_request_and_get_response_functional

| | |
|---|---|
| 函数名称 | VARIANT_BOOL tp_can_request_and_get_response_functional(<br>[in] long ADiagModuleIndex,<br>[in] BSTR ARequest,<br>[out] BSTR* AResponse); |
| 功能介绍 | 发送 CAN 诊断请求并获取应答 |
| 调用位置 | |
| 输入参数 | ADiagModuleIndex:诊断模块索引<br>ARequest:请求 |
| 返回值 | AResponse：应答 |
| 示例 | |

# 11. 附件

# 12. 常见异常解答

TOSUN同星


# 汽车电子工具链，国产领导品牌

同星智能成立于2017年，一直专注于研发国产自主可控的汽车电子基础工具链产品，也是该领域国产领导品牌。

同星智能的核心软件TSMaster及配套硬件设备，具备嵌入式代码生成、汽车总线分析、仿真、测试及诊断、标定等核心功能，覆盖了汽车整车及零部件研发、测试、生产、试验、售后全流程。

全球企业用户超4000家，用户覆盖：汽车整车厂、零部件供应商、芯片厂商、设备/服务供应商、工程机械、航空航天及舰船军工等领域。

扫码关注
获取软件下载链接

## 软件

- UDS诊断
- ECU刷写
- CCP/XCP标定
- 嵌入式代码生成
- 应用发布/加密发布
- 记录与回放
- 图形化编程
- 剩余总线仿真
- C/Python脚本
- 总线监控/发送
- SOMEIP和DoIP

## 硬件

- 1/2/4/8/12通道CAN FD/CAN转USB工具
- 1/2/6通道LIN转USB工具
- 10通道CAN FD/CAN转以太网工具
- 多通道Flexray/CAN FD转USB工具
- 多通道车载以太网/CAN FD转USB工具
- 车载以太网介质转换工具（T1转Tx）
- 多通道CAN FD/Ethernet/LIN记录仪

## 解决方案

- EOL测试设备
- FCT测试设备
- 汽车“四门两盖”试验解决方案
- 线控底盘测试解决方案
- 电机性能/耐久试验解决方案
- 新能源产线设备解决方案
- 总线一致性测试解决方案
- 信息安全解决方案

021-59560506
www.tosunai.com
上海 | 广州 | 北京 | 成都 | 台北 | 斯图加特