

# TsCAN API 编程指导 Python 版

V1.2



## 目 录

1. 什么情况下需要此文档? .....	3
2. 添加库文件.....	3
1. Python 语言: .....	3
3. 数据类型定义.....	4
1. TLIBCAN: CAN 总线发送数据类型(CANFD 同理).....	4
2. 调用示例: 创建一个发送报文(CANFD 同理).....	4
3. 调用示例: 创建一个接收报文(CANFD 同理).....	4
4. TLIBLIN: LIN 总线发送数据类型.....	5
5. 调用示例: 创建一个 LIN 发送报文.....	5
6. 调用示例: 创建一个 LIN 接收报文.....	5
4. 接口函数介绍.....	6
1. initialize_lib_tsmaster.....	6
2. tsapp_connect.....	6
3. tsapp_disconnect_AHeadle.....	6
4. tsapp_disconnect.....	6
5. tsapp_configure_baudrate_can.....	7
6. tsapp_configure_baudrate_canfd.....	7
7. tsapp_configure_baudrate_lin.....	7
8. tsapp_set_node_functiontype.....	7
9. tsapp_transmit_can_async.....	7
10. tsapp_transmit_can_sync.....	8
11. tsapp_transmit_canfd_async.....	8
12. tsapp_transmit_canfd_sync.....	8
13. tsapp_transmit_lin_async.....	8
14. tsapp_transmit_lin_sync.....	8
15. tsapp_receive_can_msgs.....	8
16. tsapp_receive_canfd_msgs.....	9
17. tsapp_receive_lin_msgs.....	9
18. tsapp_register_event_can.....	9
19. tsapp_register_event_canfd.....	9
20. tsapp_register_event_lin.....	10
5. 示例工程.....	10

# 1. 什么情况下需要此文档？

用户基于 python 编程语言，对上海同星智能科技有限公司的 TSCAN 系列工具（TC1001, TL1001, TC1011, TC1014, TC1026）进行二次开发的时候，需要参考本文档，调用 API 函数来实现对设备的程序控制。

## 2. 添加库文件

### 1. Python 语言：

要实现对同星硬件设备（CAN\CANFD\LIN）的操作，需要基于 libTSCAN.dll 动态链接库文件。该文件集成了上海同星公司对 TS 系列工具设备在 Win32(WIN64)平台上的所有 API 接口。WIN32 平台 libTSCAN.dll(x86)的运行,还需要依赖 libTSH.dll(x86)，binlog.dll(x86)以及 liblog.dll(x86)；WIN64 平台 libTSCAN.dll(x64)的运行则需要依赖 libTSH.dll(x64)。因此，需要把上述几个库文件都添加到可执行文件路径下面，如下所示：

binlog.dll	2019-10-10 20:47	应用程序扩展	340 KB
liblog.dll	2019-11-30 19:15	应用程序扩展	51 KB
libTSCAN.dll	2019-12-08 20:51	应用程序扩展	17,223 KB
libTSH.dll	2019-11-21 11:15	应用程序扩展	30 KB
mfc140.dll	2017-02-07 23:09	应用程序扩展	4,596 KB
msvcrt140.dll	2017-02-07 23:09	应用程序扩展	428 KB
test.exe	2019-09-05 17:15	可执行文件	2 KB

#### 1. 相关动态链接库

要在 python 平台调用 dll 内部的接口函数，需要在工程中调用 TScanAPI.py。该文件中主要定义了使用 API 所需要用到的数据结构类型以及函数指针类型，如下所示：

X64	2021/11/25 16:37	文件夹	
X86	2021/11/24 15:01	文件夹	
DLL.zip	2021/11/25 16:37	ZIP 文件	14,231 KB
TScanAPI.py	2021/11/24 15:07	JetBrains PyChar...	10 KB
TScanAPIDemo.py	2021/11/25 18:04	JetBrains PyChar...	4 KB
TScanAPIDemo-pyqt5.py	2021/11/24 14:58	JetBrains PyChar...	5 KB

#### 2. 引用头文件

开发人员可以根据 TScanAPI.py 定义，直接引用 api 函数进行开发。也可以根据数据结构定义，动态载入函数指针，详细情况见例程。如上图中的 TScanAPIDemo.py 以及 TScanAPIDemo-pyqt5.py（基于 pyqt5 可视化）。

## 3. 数据类型定义

### 1. TLIBCAN: CAN 总线发送数据类型(CANFD 同理)

```
class TLIBCAN(Structure):
    _pack_ = 1
    _fields_ = [("FIdxChn", c_uint8),
                 ("FProperties", c_uint8), # 定义can数据类型 1:标准数据帧 3:标准远
                                           #程帧 5: 扩展数据帧 7: 扩展远程帧
                 ("FDLC", c_uint8),
                 ("FReserved", c_uint8),
                 ("FIdentifier", c_int32),
                 ("FTimeUs", c_uint64),
                 ("FData", c_uint8 * 8),
                 ]
```

### 2. 调用示例: 创建一个发送报文(CANFD 同理)

```
msg = TScanAPI.TLIBCAN()
msg.FIdxChn = 0
msg.FIdentifier = 0x100
msg.FProperties = 5 #5: 扩展数据帧
msg.FDLC = 8
FData = [0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x17]
for i in range(len(FData)):
    msg.FData[i] = FData[i]
TScanAPI.tsapp_transmit_can_async(obj1, msg) #obj1:headle
```

### 3. 调用示例: 创建一个接收报文(CANFD 同理)

```
list = []
for i in range(16):
    item = TScanAPI.TLIBCAN()
    list.append(item)
Size = c_int(16)
chn = c_ubyte(0) #0表示通道1; 1表示通道2; 以此类推
txrx = TScanAPI.READ_TX_RX_DEF.TX_RX_MESSAGES.value
TScanAPI.tsapp_receive_can_msgs(obj1, list, size, chn, txrx)
```

#### 4. TLIBLIN: LIN 总线发送数据类型

```
class TLIBLIN(Structure):
    _pack_ = 1
    _fields_ = [("FIdxChn", c_ubyte),
                ("FErrCode", c_ubyte),
                ("FProperties", c_ubyte),
                ("FDLC", c_uint8),
                ("FIdentifier", c_ubyte),
                ("FChecksum", c_ubyte),
                ("FStatus", c_ubyte),
                ("FTimeUs", c_ulonglong),
                ("FData", c_uint8 * 8),
                ]
```

#### 5. 调用示例：创建一个 LIN 发送报文

```
msg = TScanAPI.TLIBLIN()
msg.FIdxChn = 0
msg.FIdentifier = 0x11
msg.FProperties = 1    #1: 表示发送    0: 表示接收
msg.FDLC = 8
FData = [0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x17]
for i in range(len(FData)):
    msg.FData[i] = FData[i]
```

需要注意的是：LIN 发送报文是作为主节点，故需要先将 LIN 硬件设置为主节点；

`tsapp_set_node_funtiontype(AHeadle, chn, AFunctiontype)`

#AFunctiontype :T\_LIN\_NODE\_FUNCTION.T\_MASTER\_NODE.value为设置主节点

`TScanAPI.tsapp_transmit_lin_async(objl, msg)` #objl:headle

#### 6. 调用示例：创建一个 LIN 接收报文

需要注意的是：对于 LIN 总线主节点，即使是自身接收报文，也需要主动发送帧头。即构造一个接收报文，填充方向属性 istx 为接收，然后调用发送函数把这一帧报文发送出去，就实现了帧头的发送。

```
Msg1 = TScanAPI.TLIBLIN()
Msg1.FIdxChn = 0
Msg1.FIdentifier = 0x10
Msg1.FProperties = 0    #1: 表示发送    0: 表示接收
Msg1.FDLC = 8
TScanAPI.tsapp_transmit_lin_async(objl, Msg1) #objl:headle
```

```
list = []
for i in range(16):
    item = TScanAPI.TLIBLIN()
```

```
list.append(item)
Size = c_int(16)
chn = c_ubyte(0)    #0表示通道1；1表示通道2；以此类推
txrx = TScanAPI.READ_TX_RX_DEF.TX_RX_MESSAGES.value
tsapp_receive_lin_msgs(objl, list, size, chn, txrx)
```

## 4. 接口函数介绍

### 1. initialize\_lib\_tsmaster

函数名称	Initialize_lib_tsmaster
功能介绍	初始化函数，所有 API 调用需先调用该函数，否则无效
调用位置	调用 API 前，需先调用该函数
输入参数	
返回值	

### 2. tsapp\_connect

函数名称	Tsapp_connect(ADeviceSerial, AHeadle)
功能介绍	连接 TSCAN 工具，并获取该工具的唯一句柄
调用位置	使用 TSCAN 工具之前，先调用此函数连接设备
输入参数	ADeviceSerial: != NULL, 获取指定序列号的设备 ==NULL, 获取任意处于连接状态的设备 UInt32 类型的设备句柄
返回值	==0:连接成功 ==5:设备已经连接

### 3. tsapp\_disconnect\_AHeadle

函数名称	Tsapp_disconnect_AHeadle(AHeadle)
功能介绍	根据设备句柄，断开该 TSCAN 设备
调用位置	不需要使用设备，调用此函数断开设备连接
输入参数	设备句柄
返回值	==0:断开设备成功

### 4. tsapp\_disconnect

函数名称	Tsapp_disconnect()
功能介绍	断开所有连接
调用位置	不需要使用设备，调用此函数断开设备连接
输入参数	
返回值	==0:断开设备成功

5. `tsapp_configure_baudrate_can`

函数名称	<code>tsapp_configure_baudrate_can(AHeadle, chn, ARateKbps, A120)</code>
功能介绍	根据设备句柄，设置 can 波特率
调用位置	<code>tsapp_disconnect</code> 使用连接硬件之后
输入参数	设备句柄，can 通道，波特率，是否激活终端电阻：1 激活 0 不激活
返回值	<code>==0</code> :断开设备成功

6. `tsapp_configure_baudrate_canfd`

函数名称	<code>tsapp_configure_baudrate_canfd(AHeadle, chn, ARateKbps, ADataKbps, AControlType, AControlMode, A120)</code>
功能介绍	根据设备句柄，设置 canfd 波特率
调用位置	<code>tsapp_disconnect</code> 使用连接硬件之后
输入参数	设备句柄，canfd 通道，仲裁波特率，数据波特率，CANFD 类型，模式，是否激活终端电阻
返回值	<code>==0</code> :断开设备成功

7. `tsapp_configure_baudrate_lin`

函数名称	<code>tsapp_configure_baudrate_lin(AHeadle, chn, ARateKbps)</code>
功能介绍	根据设备句柄，设置 lin 波特率
调用位置	<code>tsapp_disconnect</code> 使用连接硬件之后
输入参数	设备句柄，lin 通道，波特率
返回值	<code>==0</code> :断开设备成功
示例	

8. `tsapp_set_node_functiontype`

函数名称	<code>tsapp_set_node_functiontype(AHeadle, chn, AFunctiontype)</code>
功能介绍	根据设备句柄，设置 lin 类型（主节点/从节点）
调用位置	<code>tsapp_disconnect</code> 使用连接硬件之后
输入参数	设备句柄，lin 通道，lin 类型
返回值	<code>==0</code> :断开设备成功

9. `tsapp_transmit_can_async`

函数名称	<code>tsapp_transmit_can_async(AHeadle, msgcan)</code>
功能介绍	异步发送 can 报文
调用位置	<code>tsapp_disconnect</code> 使用连接硬件之后
输入参数	设备句柄，TLIBCAN
返回值	<code>==0</code> :断开设备成功

#### 10. tsapp\_transmit\_can\_sync

函数名称	<code>tsapp_transmit_can_sync(AHeadle, msgcan, ATimeoutMS)</code>
功能介绍	同步发送 can 报文
调用位置	<code>tsapp_disconnect</code> 使用连接硬件之后
输入参数	设备句柄, TLIBCAN, 超时判断
返回值	<code>==0</code> :断开设备成功

#### 11. tsapp\_transmit\_canfd\_async

函数名称	<code>tsapp_transmit_can_async(AHeadle, msgcanfd)</code>
功能介绍	异步发送 can 报文
调用位置	<code>tsapp_disconnect</code> 使用连接硬件之后
输入参数	设备句柄, TLIBCANFD
返回值	<code>==0</code> :断开设备成功

#### 12. tsapp\_transmit\_canfd\_sync

函数名称	<code>tsapp_transmit_can_sync(AHeadle, msgcanfd, ATimeoutMS)</code>
功能介绍	同步发送 can 报文
调用位置	<code>tsapp_disconnect</code> 使用连接硬件之后
输入参数	设备句柄, TLIBCANFD, 超时判断
返回值	<code>==0</code> :断开设备成功

#### 13. tsapp\_transmit\_lin\_async

函数名称	<code>tsapp_transmit_lin_async(AHeadle, msglin)</code>
功能介绍	异步发送 can 报文
调用位置	<code>tsapp_disconnect</code> 使用连接硬件之后
输入参数	设备句柄, TLIBLIN
返回值	<code>==0</code> :断开设备成功

#### 14. tsapp\_transmit\_lin\_sync

函数名称	<code>tsapp_transmit_can_sync(AHeadle, msglin, ATimeoutMS)</code>
功能介绍	同步发送 lin 报文
调用位置	<code>tsapp_disconnect</code> 使用连接硬件之后
输入参数	设备句柄, TLIBLIN, 超时判断
返回值	<code>==0</code> :断开设备成功

#### 15. tsapp\_receive\_can\_msgs



函数名称	<a href="#">tsapp_receive_can_msgs(AHeadle, msgscan, msgscansize, Achn, ARxTx)</a>
功能介绍	接收 can 报文
调用位置	tsapp_disconnect 使用连接硬件之后
输入参数	设备句柄, TLIBCAN[], 存储接收 can 报文的数组大小, can 通道, 0: 仅接收; 1 接收发送
返回值	==0:断开设备成功

#### 16. [tsapp\\_receive\\_canfd\\_msgs](#)

函数名称	<a href="#">tsapp_receive_canfd_msgs(AHeadle, msgscanfd, msgscanfdsize, Achn, ARxTx)</a>
功能介绍	接收 canfd 报文
调用位置	tsapp_disconnect 使用连接硬件之后
输入参数	设备句柄, TLIBCANFD[], 存储接收 canfd 报文的数组大小, lin 通道, 0: 仅接收; 1 接收发送
返回值	==0:断开设备成功

#### 17. [tsapp\\_receive\\_lin\\_msgs](#)

函数名称	<a href="#">tsapp_receive_lin_msgs(AHeadle, msgslin, msgslinsize, Achn, ARxTx)</a>
功能介绍	接收 lin 报文
调用位置	tsapp_disconnect 使用连接硬件之后
输入参数	设备句柄, TLIBLIN[], 存储接收 lin 报文的数组大小, lin 通道, 0: 仅接收; 1 接收发送
返回值	==0:断开设备成功

#### 18. [tsapp\\_register\\_event\\_can](#)

函数名称	<a href="#">tsapp_register_event_can(AHeadle, Acallback)</a>
功能介绍	注册 can 回调事件
调用位置	tsapp_disconnect 使用连接硬件之后
输入参数	设备句柄, 回调函数: <a href="#">Acallback= TScanAPI.OnTx_RxFUNC_CAN(OnPreRxCANEvent)</a>
返回值	==0:断开设备成功

#### 19. [tsapp\\_register\\_event\\_canfd](#)

函数名称	<a href="#">tsapp_register_event_lin(AHeadle, Acallback)</a>
功能介绍	注册 canfd 回调事件
调用位置	tsapp_disconnect 使用连接硬件之后
输入参数	设备句柄, 回调函数: <a href="#">Acallback= TScanAPI.OnTx_RxFUNC_CANFD(OnPreRxCANFDEvent)</a>
返回值	==0:断开设备成功

20. tsapp\_register\_event\_lin

函数名称	tsapp_register_event_lin(AHeadle, Acallback)
功能介绍	注册 lin 回调事件
调用位置	tsapp_disconnect 使用连接硬件之后
输入参数	设备句柄，回调函数： OnRxCANEvent = TScanAPI.OnTx_RxFUNC_LIN(OnPreRxLINEvent)
返回值	==0:断开设备成功

5. 示例工程

本工程演示了调用 API，实现加载 DLL，连接设备，注册回调函数，设置波特率，发送 can 报文、canfd 报文，接收 can 报文、canfd 报文等过程，运行效果图：

```
G:\pythonProject2\Scripts\python.exe G:/Python32/TSMsater_32/TScanAPIDemo.py
连接成功
波特率设置成功
can报文发送成功
canfd报文发送成功
回调事件发送接受can.FIdentifier =0x100
回调事件发送接受can.FIdentifier =0x100
接收到1通道报文id= 0x100
接收到1通道报文id=0x100
接收到1通道报文id=0x101

进程已结束，退出代码为 0
```

为了同步监测 can、canfd 报文通讯情况，可以打开同星公司的 TSMaster 软件，打开 can\canfd Trace 窗口，监测 can、canfd 报文通讯过程，如下所示：

