# 诊断模块接口说明

1. **意义**  
   为了指导开发工程师，正确的使用诊断模块，快速开发出满足车厂要求的诊断功能。
2. **诊断模块介绍**

此诊断模块根据ISO-14229-1文档，并结合部分车厂的文档进行开发，使用面向对象的思路进行设计，将模块需要处理的所有事情封装在模块内部，留出模块处理过程接口和配置接口供调用接口的工程师使用。通过调用配置接口，可以配置我们想要的功能。通过调用处理过程接口，诊断模块便能提供诊断服务，无需其他操作，便能实现诊断功能，开发起来方便快捷。

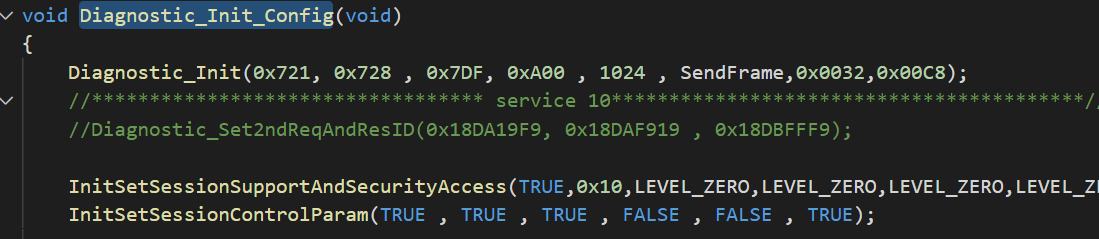
使用过程中的复杂之处在于配置，需要根据具体项目的诊断需求进行具体配置。以下详细介绍。

1. **快速入门(保证CAN驱动OK)**

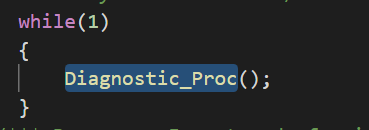
1.CAN报文接收完成，调用接口传入UDS模块，如下图：

****

2.UDS配置参考 DiagnosticDemo.c-> Diagnostic\_Init\_Config()函数，如下图：



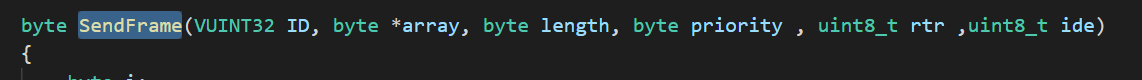
3.主循环调用Diagnostic\_Proc函数，如下图：



4.一毫秒定时中断调动 Diagnostic\_1msTimer，为UDS模块提供定时基准

C:\Users\ADMINI~1\AppData\Local\Temp\1637332783(1).png

5.确认实现CAN报文发送接口，注意次函数指针在步骤2初始化时候传入



6.实例配置代码

**UDSDemo \** **CODE \DiagnosticDemo.c**

1. **接口与配置说明**

1、**类型定义**

**1）、安全等级定义**

typedef enum{

LEVEL\_ZERO = 7,//安全等级0，当一个服务不需要安全解锁时，使用此安全等级。

LEVEL\_ONE = 1,//安全等级1，当一个服务可以在安全等级1时，使用此安全等级。

LEVEL\_TWO = 2,//安全等级2，当一个服务可以在安全等级2时，使用此安全等级。

LEVEL\_THREE = 4,//安全等级3，当一个服务可以在安全等级3时，使用此安全等级。

LEVEL\_FOUR = 8,//安全等级4，工厂模式会话使用此安全等级，用户零部件商下线配置。

LEVEL\_UNSUPPORT = 0,//不支持，当一个服务在某个会话模式不支持时，使用此等级。

}SecurityLevel;

**2）、复位类型，参考ISO-14229-1中11服务复位类型的定义。**

typedef enum{

HARD\_RESET = 1,//硬件复位

KEY\_OFF\_ON\_RESET = 2,//关开钥匙复位

SOFT\_RESET = 3,//软件复位

ENABLE\_RAPID\_POWER\_SHUTDOWN = 4,//预留，一般不使用

DISABLE\_RAPID\_POWER\_SHUTDOWN = 5,//预留，一般不使用

}EcuResetType;

**3）、DTC类型定义。**

typedef enum{

ISO15031\_6DTCFORMAT= 1,

ISO14229\_1DTCFORMAT = 2,

SAEJ1939\_73DTCFORMAT = 3,

}DTCFormatIdentifier;

**4）、诊断故障状态定义**

typedef enum{

PASSED,//测试通过

IN\_TESTING,//测试未完成

FAILED,//测试失败

}DTCTestResult;

**5)、DID类型定义**

typedef enum{

EEPROM\_DID,//静态存储器DID，存储在EEPROM中的DID使用此类型

REALTIME\_DID,//实时DID，存储在RAM中，会实时改变的数据使用此类型

IO\_DID,//输入输出控制DID，需要通过2F服务控制的DID使用此类型

}DIDType;

**6）、DID的读写属性**

typedef enum{

READONLY = 1,//只读

WRITEONLY = 2,//只写

READWRITE = 3,//可读写

}ReadWriteAttr;

**7)、通信控制参数**

typedef enum{

ERXTX,//enableRxAndTx

ERXDTX,//enableRxAndDisableTx

DRXETX,//disableRxAndEnableTx

DRXTX,//disableRxAndTx

//ERXDTXWEAI,//enableRxAndDisableTxWithEnhancedAddressInformation

//ERXTXWEAI,//enableRxAndTxWithEnhancedAddressInformation

}CommulicationType;

**8）、通信控制的控制对象参数**

typedef enum{

NCM = 1,//application message

NWMCM,//network manage message

NWMCM\_NCM,//application and netwrok manage message

}communicationParam;

**9）、子功能在会话的支持情况**

typedef enum{

SUB\_DEFAULT = 1,//sub function supported in default session

SUB\_PROGRAM = 2,//sub function supported in program session

SUB\_EXTENDED = 4,////sub function supported in extedned session

SUB\_FACTORY = 8,//sub funcion supported in factory session,

SUB\_ALL = 7,//sub function supported in both of three session

}SubFunSuppInSession;

**10)、诊断故障码的等级**

typedef enum{

LEVEL\_A,

LEVEL\_B,

LEVEL\_C,

}DTCLevel;

**11）、输入输出控制接口函数原型**

typedef uint8\_t (\*IoControl)(uint8\_t ctrl, uint8\_t param);

Ctrl表示控制类型:

0：归还控制权到ECU

1：恢复默认状态

2：冻结当前

3：短时调整

只有当ctrl为3时存在param，param参数根据具体的DID而不同，如当控制开关时，可以表示为：

0：关

1：开，

如控制档位时

1:1档

2:2档

....

**12）、安全解锁算法接口函数原型**

typedef uint32\_t (\*SecurityFun)(uint32\_t);

此原型表示一个函数指针，有一个uint32\_t型的参数，表示种子。返回uint32\_t型值，表示根据算法算出的秘钥

**13）、DTC的检测接口函数原型**

typedef DTCTestResult (\*DetectFun)(void);

无参数，需要返回DTCTestResult 类型的值，2表示测试失败，0表示测试通过，1表示正在测试（测试未完成）。

**14）、复位接口函数原型。**

typedef void (\*ResetCallBack)(EcuResetType);

参数

EcuResetType：复位类型，取值范围，1-5，分别表示硬件复位，key-off-on复位，软件复位。通常用1和3。

**15）、通信控制接口函数原型**

typedef void (\*CommCallBack)(CommulicationType , communicationParam);

参数：

CommulicationType ：参考1.7定义

communicationParam：参考1.8定义

**16）、CAN发送接口函数原型**

typedef uint8\_t (\*SendCANFun)(uint32\_t ID, uint8\_t \*array, uint8\_t length, uint8\_t priority);

注：以上所有接口函数原型供诊断开发时使用，开发时必须提供以上接口，否则诊断模块将无法正常工作。

#define USE\_MALLOC 0//1：使用动态内存分配，0：不使用动态内存分配。

#define USE\_J1939\_DTC 0//建议不修改

当时不使用动态内存分配时候，存在以下参数，可调节。

/\*======================== buf size config ================================\*/

#define MAX\_DTC\_NUMBER 35//最大DTC个数

#define MAX\_DID\_NUMBER 70//最大DID个数

#define MAX\_SNAPSHOT\_NUMBER 10//最大快照信息个数

#define MAX\_GROUP\_NUMBER 5//最大DTC组个数

/\*======================== buf size config ================================\*/

**2、接口函数**

**1）、诊断基本配置函数**

void Diagnostic\_Init(uint32\_t requestId, uint32\_t responseId, uint32\_t funRequestId, uint16\_t EEPromStartAddr, uint16\_t EEpromSize,SendCANFun sendFun,uint16\_t p2CanServerMax, uint16\_t p2ECanServerMax);

requestId:诊断仪请求ID（物理寻址）

responseId：ECU响应ID（物理寻址）

funRequestId：功能寻址请求ID

EEPromStartAddr：诊断模块可使用的EEPROM起始

EEpromSize：诊断模块可使用的EEPROM的大小

sendFun：诊断模块发送CAN报文使用的函数指针

p2CanServerMax：诊断的响应时间参数限制（未发送78响应时，具体可参数项目诊断规范）

p2ECanServerMax：诊断的响应时间参数限制（发送了78响应后，具体可参数项目诊断规范）

**2）、诊断额外支持的请求和响应ID**

void Diagnostic\_Set2ndReqAndResID(uint32\_t requestId1, uint32\_t responseId1,uint32\_t funRequestId1);

requestId1：诊断仪第二请求ID（物理寻址）

responseId1：ECU第二响应ID（物理寻址）

funRequestId1：第二功能寻址请求ID

**3）、诊断模块释放接口**

void Diagnostic\_DelInit(void);

此接口会处理释放内存，保存故障码的操作，一定要在休眠之前调用。

**4）、诊断模块报文接收函数**

void Diagnostic\_RxFrame(uint32\_t ID,uint8\_t\* data,uint8\_t IDE,uint8\_t DLC,uint8\_t RTR);

此函数需要在接收中断中调用，如果不调用,诊断模块将无法收到任何报文，无法提供任何服务。参数：

ID：报文ID，可以是11位和29位ID

Data：报文数据指针

IDE：参考S12G手册

DLC：报文长度

RTR：参考S12G手册

**5）、诊断模块时间基数函数**

void Diagnostic\_1msTimer(void);

此函数需要在1毫秒的RTI中断中调用，如不调用，诊断模块所有与超时相关的功能将不能工作（包括多帧响应，S3超时等）。

**6）、添加安全算法的函数**

bool InitAddSecurityAlgorithm(SecurityLevel level, SecurityFun AlgoritthmFun,byte SeedSubFunctionNum,byte KeySubFunctionNum , uint8\_t\* FaultCounter,uint8\_t FaultLimitCounter , uint32\_t UnlockFailedDelayTimeMS,SubFunSuppInSession SubFuntioncSupportedInSession,uint8\_t KeySize);

此函数的功能是为诊断模块的添加安全算法，最多支持三个等级的安全算法，如果不添加安全算法，27服务将没有正响应。参数：

Level：能使用LEVEL\_ONE，LEVEL\_TWO，LEVEL\_THREE，不能使用LEVEL\_ZERO和LEVEL\_UNSUPPORT

AlgoritthmFun：安全解锁算法函数，参考三.1.12。

SeedSubFunctionNum：此算法支持的请求种子的子功能，如“27 01”中的“01”

KeySubFunctionNum ：此算法支持的发送秘钥的子功能，如“27 02”中的“02”

FaultCounter：预留参数，设置为NULL

FaultLimitCounter：解锁失败次数限制，超时此次数时，启用延时

UnlockFailedDelayTimeMS：解锁失败后延时时间，单位为毫秒，如3000表示3秒

SubFuntioncSupportedInSession：子功能在会话模式的支持情况，可以是SUB\_PROGRAM ，SUB\_EXTENDED，也可以都支持，使用按位或的方式SUB\_PROGRAM | SUB\_EXTENDED。

KeySize：seed和可以的长度，可以设置为2或者4。设置为2时只使用高生成种子的高两个字节，解锁算法生成的秘钥也需要放到高两个字节。设置为4时将使用所有字节。

**7）、初始化工厂模式安全算法函数**

void InitFactorySecuriyAlgorithm(void);

无参数，此函数内部会调用InitAddSecurityAlgorithm函数，添加安全算法，算法包含于内部，如不进行此初始化，工厂模式将无法解锁。

**8）、配置服务的函数**

bool InitSetSessionSupportAndSecurityAccess(bool support ,uint8\_t service,uint8\_t PHYDefaultSession\_Security, uint8\_t PHYProgramSeesion\_Security, uint8\_t PHYExtendedSession\_Security, uint8\_t FUNDefaultSession\_Security, uint8\_t FUNProgramSeesion\_Security, uint8\_t FUNExtendedSession\_Security);

Support：只能为TRUE，如果为FALSE和未配置一样会有11否定响应。

Service：服务名称，如0x10，0x11，0x27等（一次只能使用一个）

PHYDefaultSession\_Security：服务在物理寻址默认会话支持的安全访问等级。

PHYProgramSeesion\_Security：服务在物理寻址编程会话支持的安全访问等级。

PHYExtendedSession\_Security：服务在物理寻址扩展会话支持的安全访问等级。

FUNDefaultSession\_Security,：服务在功能寻址默认会话支持的安全访问等级。

FUNProgramSeesion\_Security：服务在功能寻址编程会话支持的安全访问等级。

FUNExtendedSession\_Security：服务在功能寻址扩展会话支持的安全访问等级。

注意：以上6个参数，

如果支持，不需要安全解锁，则使用LEVEL\_ZERO,

如果不支持，则使用LEVEL\_UNSUPPORT，

如果需要安全解锁等级1才能支持，则使用 LEVEL\_ONE，

如果需要安全解锁等级2才能支持，则使用LEVEL\_TWO，

如果需要安全解锁等级3才能支持，则使用LEVEL\_THREE，

如果同时支持多个安全等级，则只用按位或的方式，如LEVEL\_TWO|LEVEL\_THREE

**9）、添加DID的接口函数**

void InitAddDID(uint16\_t DID , uint8\_t DataLength , uint8\_t\* DataPointer , DIDType DidType , IoControl ControlFun , ReadWriteAttr RWAttr,uint16\_t EEaddr, bool SupportWriteInFactoryMode);

DID：DID数字，如：0xF190

DataLength：DID的数据长度，如F190为17。

DataPointer：DID数据指针，此指针由应用程序提供，当类型为EEPROM\_DID时，此参数设为NULL，类型为IO\_DID并且不需要读时，也可设置为NULL。

DidType:DID的类型可以是EEPROM\_DID,REALTIME\_DID,IO\_DID。

ControlFun：输入输出控制的函数指针，当类型不为IO\_DID时，此参数设置为NULL。

RWAttr：读写属性

EEaddr：DID的eeprom地址只有DidType为EEPROM\_DID时有效，当此参数为0时，诊断模块将自动分配eeprom地址，因此如果不需要手动指定地址，将此值设置为0即可。

SupportWriteInFactoryMode:是否支持在工厂模式可写。

注意：工厂模式的会话模式为0x71，需要先切换到10 03扩展会话，才能切换到工厂模式会话，工厂模式写DID数据需要先27解锁，分别是27 71请求种子，27 72发送秘钥。工厂模式解锁算法包含在诊断模块内部，对客户不可见。

**10）、添加故障码的接口函数**

#if USE\_J1939\_DTC

void Diagnostic\_DM1MsgEnable(bool dm1en);

bool InitAddDTC(uint32\_t DTCCode,DetectFun MonitorFun,byte DectecPeroid, byte ValidTimes,DTCLevel dtcLevel,uint32\_t spn, uint8\_t fmi);

#else

bool InitAddDTC(uint32\_t DTCCode,DetectFun MonitorFun,byte DectecPeroid, byte ValidTimes,DTCLevel dtcLevel);

#endif

灰色部分可以不作关注

DTCCode：诊断故障代码，如0x910223，诊断模块只使用低24位，高8位设置为零。

MonitorFun：故障检测函数指针。

DectecPeroid：故障检测周期，此参数暂未使用，可以设置为0.

ValidTimes：故障有效次数，记录历史故障码的故障出现次数，当在历史故障和当前故障码同时置位时，设置为1，当历史故障码需要多个点火循环才能置位时，可设置为大于等于2的数。2表示需要两个点火周期，3表示3个，类推。

dtcLevel：故障等级，可以设置为LEVEL\_C

**11）、添加快照信息接口函数**

void InitAddDTCSnapShot(uint8\_t recordNumber , uint16\_t ID , uint8\_t\* datap , uint8\_t size);

recordNumber：快照信息记录号，如1，表示全局快照，2，表示局部快照。

ID：此快照的ID，如0x9102表示快照车速信息。

Datap：此快照记录的内存指针，需要是能表示实时状态（如实时车速）的内存指针。

Size：此快照的大小，字节数。

**12）、设置故障扩展信息-老化计数器的扩展信息号的接口函数**

void InitSetAgingCounterRecordNumber(uint8\_t RecordNumer);

RecordNumer：老化计数器信息的序号（需要参考诊断规范中19 06的响应信息，一般范围1-4）

**13）、设置故障扩展信息-已老去计数器的扩展信息号的接口函数**

void InitSetAgedCounterRecordNumber(uint8\_t RecordNumer);

RecordNumer：已老去计数器信息的序号（需要参考诊断规范中19 06的响应信息，一般范围1-4）

**14）、设置故障扩展信息-故障发生次数计数器的扩展信息号的接口函数**

void InitSetOccurrenceCounterRecordNumber(uint8\_t RecordNumer);

RecordNumer：故障发生次数计数器信息的序号（需要参考诊断规范中19 06的响应信息，一般范围1-4）

**15）、设置故障扩展信息-故障待定计数器的扩展信息号的接口函数**

void InitSetPendingCounterRecordNumber(uint8\_t RecordNumer);

RecordNumer：故障待定计数器信息的序号（需要参考诊断规范中19 06的响应信息，一般范围1-4）

**16）、设置支持的故障位的接口函数**

void InitSetDTCAvailiableMask(uint8\_t AvailiableMask);

AvailiableMask：故障位，如0x09表示支持当前位和历史位

**17）、设置DTCgroup的接口函数**

void InitAddDTCGroup(uint32\_t Group);

Group：14服务的group，目前支持支0xFFFFFF（仅低24位有效），清除所有故障码。

**18）、配置11服务的接口函数**

void InitSetSysResetParam(bool support01 , bool support02 , bool support03 , bool support04 ,

bool support05 , ResetCallBack callback, bool supressPosResponse);

support01:11服务是否支持01子功能，TRUE:支持，FALSE：不支持

support02:11服务是否支持02子功能，TRUE:支持，FALSE：不支持

support03:11服务是否支持03子功能，TRUE:支持，FALSE：不支持

support04:11服务是否支持04子功能，TRUE:支持，FALSE：不支持

support05:11服务是否支持05子功能，TRUE:支持，FALSE：不支持

Callback:复位接口函数指针，由应用提供，诊断模块只调用，具体的复位动作需要应用根据参数执行。

supressPosResponse:是否支持抑制响应，TRUE：支持，FALSE：不支持

**19）、配置28服务的接口函数**

void InitSetCommControlParam(bool supportSubFun00, bool supportSubFun01 , bool supportSubFun02 , bool supportSubFun03 , bool supportType01, bool supportType02, bool supportType03, CommCallBack callback, bool supressPosResponse);

supportSubFun00：是否支持00子功能-使能接收和发送，TRUE：支持，FALSE：不支持

supportSubFun01：是否支持01子功能-使能接收关闭发送，TRUE：支持，FALSE：不支持

supportSubFun02：是否支持02子功能-关闭接收使能发送，TRUE：支持，FALSE：不支持

supportSubFun03：是否支持03子功能-关闭接收和发送，TRUE：支持，FALSE：不支持

supportType01：是否支持控制参数01-一般通信报文，TRUE：支持，FALSE：不支持

supportType02：是否支持控制参数02-网络管理报文，TRUE：支持，FALSE：不支持

supportType03：是否支持控制参数03-通信报文和网络管理报文，TRUE：支持，FALSE：不支持

Callback：通信控制接口函数指针，由应用提供，诊断模式只负责调用，控制逻辑由应用实现。

supressPosResponse：是否支持抑制响应，TRUE：支持，FALSE：不支持

**20）、配置10服务的接口函数**

void InitSetSessionControlParam(bool supportSub01, bool supportSub02,bool supportSub03, bool sub02SupportedInDefaultSession, bool sub03SupportedInProgramSession, bool supressPosResponse);

supportSub01：是否支持01子功能-默认会话，TRUE：支持，FALSE：不支持

supportSub02：是否支持02子功能-编程会话，TRUE：支持，FALSE：不支持

supportSub03：是否支持03子功能-拓展会话，TRUE：支持，FALSE：不支持

sub02SupportedInDefaultSession：在默认会话是否支持02子功能，TRUE：支持，FALSE：不支持

sub03SupportedInProgramSession：在编程会话是否支持03子功能，TRUE：支持，FALSE：不支持

supressPosResponse：是否支持抑制响应，TRUE：支持，FALSE：不支持

**21）、配置3E服务的接口函数**

void InitSetTesterPresentSupress(bool supressPosResponse);

supressPosResponse：是否支持抑制响应，TRUE：支持，FALSE：不支持

**22）、配置85服务的接口函数**

void InitSetDTCControlSupress(bool supressPosResponse);

supressPosResponse：是否支持抑制响应，TRUE：支持，FALSE：不支持

**23）、配置当前会话模式DID的接口函数**

void InitSetCurrentSessionDID(uint16\_t m\_DID);

由于此数据在诊断模块，应用无法得到，所以使用此接口即可。此函数内部会添加DID。

**24）、配置CAN数据库DID的接口函数**

void InitSetCanDataBaseVersionDID(uint16\_t m\_DID);

由于此数据在诊断模块，应用无法得到，所以使用此接口即可。此函数内部会添加DID。

**25）、配置CAN诊断版本DID的接口函数**

void InitSetCanDiagnosticVersionDID(uint16\_t m\_DID);

由于此数据在诊断模块，应用无法得到，所以使用此接口即可。此函数内部会添加DID。

**26）、配置网络管理版本DID的接口函数**

void InitSetCanNMVersionDID(uint16\_t m\_DID);

由于此数据在诊断模块，应用无法得到，所以使用此接口即可。此函数内部会添加DID。

**27）、配置CAN驱动版本DID的接口函数**

void InitSetCanDriverVersionDID(uint16\_t m\_DID);

由于此数据在诊断模块，应用无法得到，所以使用此接口即可。此函数内部会添加DID。

**28）、加载所有诊断模块数据的接口函数**

void Diagnostic\_LoadAllData(void);

需要先 配置好DID，安全算法，DTC后才能调用此接口函数，此接口函数回从EEPROM中读取所有需要的数据。

**29）、配置车架号的接口函数**

void Diagnostic\_ConfigVIN(uint8\_t length, uint8\_t\* data);

/\*\*\*\*\*\*\*\*\*\*\*\*set netwrok layer parameters\*\*\*\*\*\*\*\*/

**30）、设置网络层参数的接口函数**

void Diagnostic\_SetNLParam(uint8\_t TimeAs, uint8\_t TimeBs, uint8\_t TimeCr, uint8\_t TimeAr, uint8\_t TimeBr, uint8\_t TimeCs, uint8\_t BlockSize, uint8\_t m\_STmin, uint8\_t FillData);

TimeAs：网络层定时参数AS

TimeBs：网络层定时参数BS

TimeCr：网络层定时参数CR

TimeAr：网络层定时参数AR

TimeBr：网络层定时参数BR

TimeCs：网络层定时参数CS

BlockSize：网络层参数BloskSieze（BS）

STmin：网络层定时参数STmin

FillData：未使用字节的填充数据

**31）、诊断处理过程的接口函数**

void Diagnostic\_Proc(void);

此函数时最终实现诊断功能的函数，需要放到主循环不停的调用，如有需要，可以设置定时调用，最大定时为1MS。