DOI:10.3969/j.issn.1001 - 4551.2018.10.019

基于 J2534 协议的诊断仪与 ECU 模拟器接口设计*

黄利权,赵治国*
(同济大学 汽车学院,上海 201804)

摘要:针对目前在汽车后市场故障诊断仪诊断内容开发领域出现的由于平台复杂而导致诊断内容开发工程师效率低下的问题,对开发平台结构的组成、ECU 模拟器软件的设计、J2534 协议接口的实现等方面进行了研究,提出了把调试平台简化为一台电脑,将 J2534 诊断接口协议应用到诊断软件和 ECU 模拟器的通信中的思路,实现了离线故障诊断的仿真;开发了 ECU 模拟器及其与诊断仪的通信接口,模拟器采用双进程模型和诊断仪进行了通信,其与已开发完成的故障诊断软件 DiagAnalyzer 间通过 J2534 协议所规定的 API 完成了通信。研究和测试结果表明:该 ECU 模拟器具备可行性和便捷性,诊断内容开发工程师只要在数据配置文件中写入特定的诊断数据,即可完成对特定车型诊断仪数据的快速开发;此外其可以和其他遵循 J2534 协议的故障诊断软件进行匹配和通信。

关键词:J2534 协议;ECU 模拟器;ISO15765;RPC 通信;故障诊断仪
中图分类号:TP273;U472.9 **文献标志码**:A **文章编号**:1001 - 4551(2018)10 - 1116 - 07

Design of interface between ECU simulator and fault diagnosis device based on J2534 protocol

HUANG Li-quan, ZHAO Zhi-guo
(School of Automotive Studies, Tongji University, Shanghai 201804, China)

Abstract: Aiming at the problem of low development efficiency of diagnostic engineers due to the complex composition of the platform in the development field of diagnostic content of diagnostic tool in automobile aftermarket, the composition of the development platform, the design of the ECU simulator and the implementation of the specified API in J2534 protocol were researched. The idea of simplifying the platform into one computer, applying J2534 protocol into the communication between diagnostic software and ECU simulator and implementing off-line diagnosis simulation were proposed. ECU simulator and the communication interface of diagnostic tool were developed, communication between ECU simulator and the diagnostic tool named DiagAnalyzer was completed by the specified API in J2534 protocol. The dual-process model was adapted to communicate with diagnostic tool by this simulator. The results indicate that this design scheme is feasible and convenient, diagnostic engineer could develop the diagnostic content for specified type of vehicle quickly after writing special diagnostic data into data configuration file, finally, it also could match and communicate with other diagnostic softwares complied with the J2534 interface protocol.

Key words: J2534 protocol; ECU simulator; ISO15765; RPC communication; diagnostic tool

0 引 言

当今汽车因装配诸多的电控单元(electric control unit, ECU)而带来的电控系统维修困难,而汽车故障诊断仪则用于检测汽车上安装的 ECU 出现各种软、硬件故障,故障诊断仪和汽车上各个 ECU 的诊断模块进行通信,读取 ECU 记录的各种故障数据和内容信息,控制 ECU 完成特定的动作测试程序。现有的开发平

收稿日期:2017 - 12 - 26
基金项目:国家自然科学基金资助项目(51675381)
作者简介:黄利权(1983 -),男,浙江杭州人,硕士研究生,主要从事汽车故障诊断软件开发和 ECU 软件开发方面的研究。E-mail:xspirate@126.com
通信联系人:赵治国,男,教授,博士生导师。E-mail:zhiguozhao@tongji.edu.cn

台通过以下两种形式来调试和验证各种诊断数据的正确性:(1)手持式诊断仪和 PC 机上的 ECU 软件模拟器进行通信;(2)手持式诊断仪和实验室内台架上的 ECU 软件模拟器进行通信。因为这两种开发模式需要操作手持式诊断仪,而当诊断内容开发工程师在处理数量巨大的诊断数据时,繁琐且慢速的仪器操作会导致开发效率的低下,需要设计可以在同一台 PC 机上运行故障诊断软件和 ECU 模拟器的解决方案,通过该 ECU 模拟器可以对故障诊断仪进行诊断数据的离线仿真。

另外,根据 ISO15765 规范,PC 机上的应用软件可以利用 J2534 API 作为访问 Pass-thru(故障诊断仪在 PC 机上的工作模块,在本文中代表了在 PC 机上运行的诊断软件)设备驱动程序的接口。该协议包括 J2534-1、J2534-2 和 J2534-3 等 3 个部分。其中 J2534-1 定义了用于对发动机电控单元排放相关模块再编程的 API 接口,这些接口不仅可以用于再编程,也可以用于其他的功能目标[1]。通过这些 API 接口应用程序可以控制 Pass-thru 设备,也可以控制 Pass-thru 设备和车辆之间的通信。Pass-thru 设备不需要解释所传输消息的内容,它允许任何的消息策略和消息结构被使用,只要它们能够被应用程序和 ECU 所理解。J2534 相关的 API 接口类型如下:通信建立和断开,消息读写,启动和停止周期性消息,设置可编程电压,启动和关闭消息过滤器,读取版本信息和取得 Last Error 等操作[2]。

为了测试特定车型的诊断数据,笔者进一步开发了同样能在 PC 机上运行的 ECU 模拟器,只需在配置文件中修改虚拟 ECU 的诊断数据,就可以在诊断仪和 ECU 之间建立通信,实现在一台电脑上通过 J2534 诊断协议来完成故障诊断的数据验证。

1　接口方案设计

1.1　ECU 模拟器模块

ECU 模拟器模块如图 1 所示。

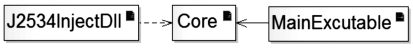

图 1　ECU 模拟器模块

ECU 模拟器包括 MainExcutable、Core 和 J2534InjectDll 3 个模块。各模块功能如下:

(1)MainExcutable 模块把操作接口和高级功能提供给了用户,并在 Core 模块提供功能的基础上升级而成;

(2)Core 模块完成了主要的接口模拟工作,处理来自于 MainExcutable 和 J2534InjectDll 的请求;

(3)J2534InjectDll 模块把来自于故障诊断仪的调用推送给 Core,并把来自于 Core 的响应推送给故障诊断仪。

模块结构图如图 2 所示。

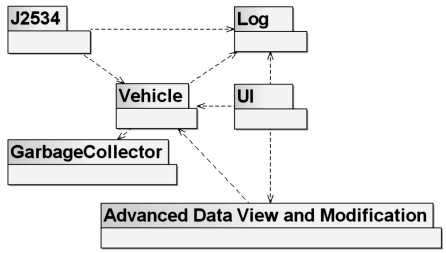

图 2　模块结构图

模拟器操作界面模块 UI 分别调用了 Log 模块、Vehicle 模块和高级 DataView 修改模块,用于显示和编辑诊断通信数据。而 J2534 模块则通过 Vehicle 模块取得车辆诊断协议配置信息和 ECU 相关的诊断模拟数据,用于和故障诊断仪的相应 API 接口通信。

1.2　J2534Inject. dll 模块

J2534Inject. dll 模块经过 Windows 操作系统注册后就能够被 J2534 应用程序所识别。J2534 应用程序首先加载 J2534Inject. dll,然后映射到它自己的地址空间,最后调用 J2534Inject. dll 中的 J2534 接口函数来完成诊断通信。

1.3　模拟器双进程模型

模拟器接口方案采用的双进程模型如图 3 所示。

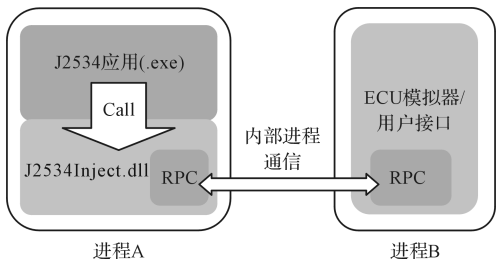

图 3　双进程模型

其把模拟器从 J2534 应用程序中隔离出来。J2534Inject. dll 驻留在以 J2534 为基础的诊断软件中,模拟器有它独立的进程。两个进程通过 Windows RPC 进行信息交换。

1.4　采用 RPC 通信机制的原因分析

模拟器模块运行成一个独立的进程,并构建了 J2534Inject. dll 用于提供 J2534 接口,因为该 DLL 在故障诊断仪的上下文中运行,需要使用 IPC(进程间数据交换)通信机制,当前 IPC 没有选择共享内存机制,而

是使用了基于 RPC(远程过程调用,即两个程序间通过传输协议进行通信)的 IPC[3]。本研究采用该机制使得开发更容易,不需要担心进程间通信的技术问题,而是让操作系统来完成该项工作,开发人员只需要把它视为本地调用即可。另外当通信发生在相同的物理机器上时,RPC 可以通过配置来使用 LPC(本地过程调用)方法完成消息推送。

1.5 相比已有的解决方案所具备的优点

传统的 ECU 模拟器方案中手持式诊断仪和 PC 机上的 ECU 软件模拟器进行通信,有如下的缺点:手持式诊断仪在操作时数据显示有一定延迟,且诊断内容开发工程师需要在仪器和 PC 机电脑之间反复切换操作,另外 ECU 模拟器和手持式诊断仪之间的通信需要有接线盒以及 USB 转接板等设备,在调试时有时会因为通信链路存在问题而需要排查原因。而采用了 J2534 诊断协议后,则可以完全避免以上问题。同时也可以通过开发脚本程序把诊断数据按照指定格式直接转化为该 ECU 模拟器的数据配置文件,用于诊断通信。

2 ECU 模拟器功能设计

2.1 用例图和功能描述

软件模拟器 UML 用例图如图 4 所示。

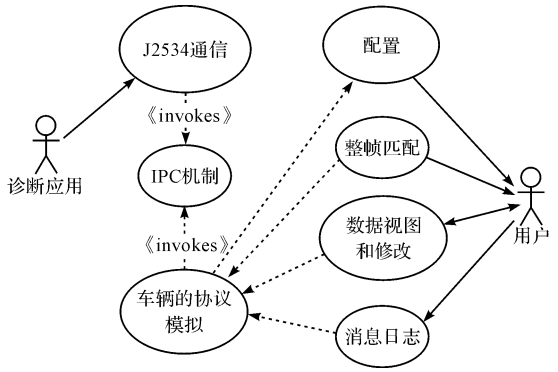

图 4 UML 用例图

诊断应用角色代表了诊断仪,而用户角色则代表了 ECU 模拟器操作者。诊断仪中的“J2534 通信”用例和“车辆协议模拟”用例通过 IPC 机制进行通信。同时模拟器的功能用例(整帧匹配、数据浏览和修改和消息日志)是依赖于“车辆协议模拟”用例来实现的。而“整帧匹配功能”用例则将匹配结果反馈给模拟器操作者。最后“配置”用例不同,则“车辆协议模拟”用例对应不同的诊断协议,即向操作者提供不同的协议模拟功能。

2.2 J2534 通信和 IPC 通信

用例向诊断工具提供了一个含有 J2534 接口的 J2534Inject. dll。诊断工具加载该 DLL,并通过它和模拟器进行通信。J2534Inject. dll 必须遵循 J2534 标准。

因为该模拟器和 J2534Inject. dll 并不在同一个进程中,必须有一种机制来完成模拟器和该 DLL 之间的 IPC 通信。IPC 机制使得诊断工具可以像调用自己的内部函数一样调用程序。

3 J2534 类软件设计

3.1 J2534 类图

因为类 J2534RpcInvoker 包含在 J2534InjectDll 里,含有该类的文件应该被加入 J2534InjectDll 的工程,而其他类属于 Core 工程。

J2534 软件包中的诊断仪接口需要遵循 J2534 标准。该软件包包含了用于通信的车辆程序接口和由诊断仪引发的 J2534 工作日志。

J2534 类图如图 5 所示。

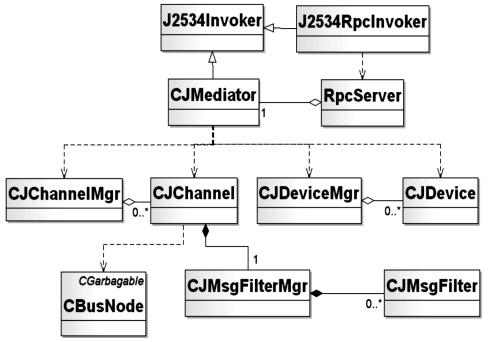

图 5 J2534 类图

其中,通信信道管理器 CJChannelMgr 中包含了多个信道 CJChannel,而每个信道则由若干个消息滤波管理器 CJMsgFilterMgr 所组成。同时每个信道 CJChannel 依赖于带有垃圾回收功能的总线节点类 CBusNode,总线节点是由相关诊断协议栈构成,而协议栈中不同层(物理层、数据链路层、网络层和应用层)由不同类型的诊断协议所组合而成[4]。另外 CJChannel 则负责主要的 J2534 通信,即负责收发消息并调用消息过滤管理器。它把类似 J2534 协议的请求转化为了内部的一些协议和数据组织方式。设备管理器 CJDeviceMgr 由多个 CJDevice 所组成,CJDevice 目前主要代表了故障诊断仪模型,它记录了引脚和相关的电压信息。

CJMediator 继承自 J2534Invoker,实现了 J2534 协议中包含的 API 接口,并且负责调度消息滤波管理器 CJMsgFilterMgr,信道 CJChannel,设备管理器 CJDeviceMgr 和设备 CJDevice。

3.2　J2534 时序图

J2534 序列图如图 6 所示。

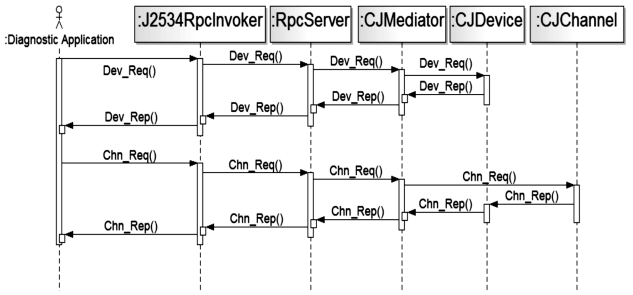

图 6　J2534 序列图

当诊断应用发起一个设备相关的调用时,完成如下工作:

(1)在 J2534InjectDll 中,J2534 接口的调用工作将被派遣给 J2534RpcInvoker,它也有类似于 J2534 的成员函数;

(2)J2534RpcInvoker 封装了诊断应用发过来的参数,然后再发送给 RPC 接口;

(3)在模拟器的处理流程中,当接收到一个来自于 RPC 客户端的消息时,类 RpcServer 代表的监听线程将被唤醒用于处理 RPC 调用;

(4)RpcServer 拆开接收到的参数,然后把它们转发给含有类似于 J2534 协议相关成员函数的 CJMediator;

(5)CJMediator 解释了 J2534 协议相关的接口并调用了 CJChannel 和 CJDevice 实例提供的方法,如果需要创建一个新的实例,或者需要释放一个已经存在的实例,那么 CJMediator 将指示 CJDeviceMgr 和 CJChannelMgr 分别完成或释放此类任务;

(6)在处理来自 J2534 协议相关接口的请求后,调用流程将逆向进行,参数需要被返回,并且函数的返回值需要被封装,最后需要由 RpcServer 和 J2534RpcInvoker 完成拆解;

(7)负责 RPC 通信机制的类 J2534RpcInvoker 包含于 J2534InjectDLL,其他则包含在 Core 模块里。

4　Vehicle 类软件设计

4.1　Vehicle 类图

Vehicle 类属于 Core 工程,类中的 CVehicle、CBusNode 和 CDiagProtocol 将被本模块以外的软件包所引用。

Vehicle 类图如图 7 所示。

主要内容解释如下:

(1)核心的处理类是 CVehicle,并通过它给出了

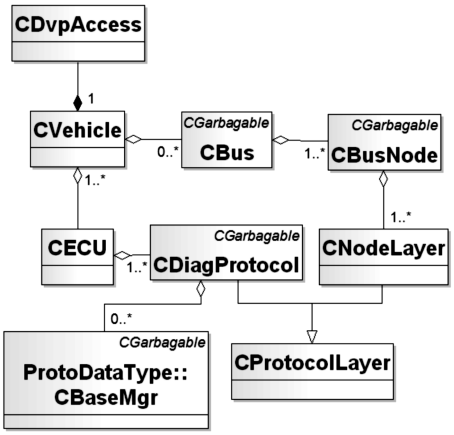

图 7　Vehicle 类图

CBus 和 CECU。一个 CBus(即某个车型的总线)是由多个 CBusNode 实例组成,同时一个 CECU 是由于多个 CDiagProtocol 实例组成;

(2)CBusNode 是 J2534 模块和 Vehicle 模块通信的桥梁;

(3)CDiagProtocol 带有面向 UI 接口。

4.2　Vehicle 时序图

当 J2534 通道从 RPC 接收到消息时,它将把消息推送到关联在它身上的 CBusNode 上。

Vehicle 消息处理时序过程如图 8 所示。

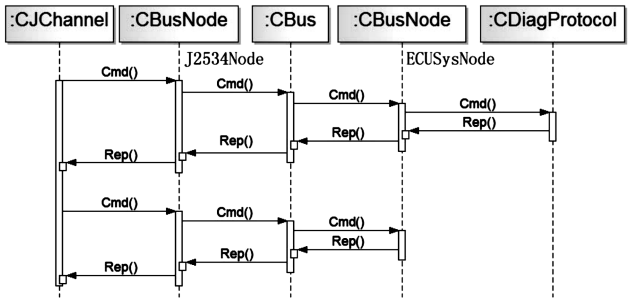

图 8　Vehicle 消息序列图

对于所有消息都按照如下过程处理:

(1)CJChannel 把消息放置到挂在它上面的 CBusNode 上(通过调用 J2534Node);

(2)J2534Node 把消息从诊断协议栈上层传到底层(上层和底层根据 ISO 协议分层定义),每一个存在层均按照协议规范封装消息(通常添加了报文头和校验码),最后底层把消息传到总线上(CBus 的实例);

(3)CBus 把它缓冲里的消息推送到其他 CBusNode 实例(这里为 EcuSysNode);

(4)当 EcuSysNode 的诊断协议栈底层接收到消息时,它把消息从底层往上层传递,每个协议栈中存在的协议层均按照协议解析消息(通常是远程报文头和校验码);

(5)如果一条消息能够在 EcuSysNode 的某一协议层中被处理,那么停止继续向上层传递消息,当前协议层将直接处理它并发送响应;如果消息不被诊断层以下的协议层处理,CDiagProtocol 将接收下,并有机会去处理它。在处理后它会发送报文响应,响应消息的处理路径按从 EcuSysNode 到 CBus,再到 J2534Node,最后到 CJChannel 的流程运转。

4.3 协议栈每层中命令处理激活算法

协议栈每一层的命令处理流程如图 9 所示。

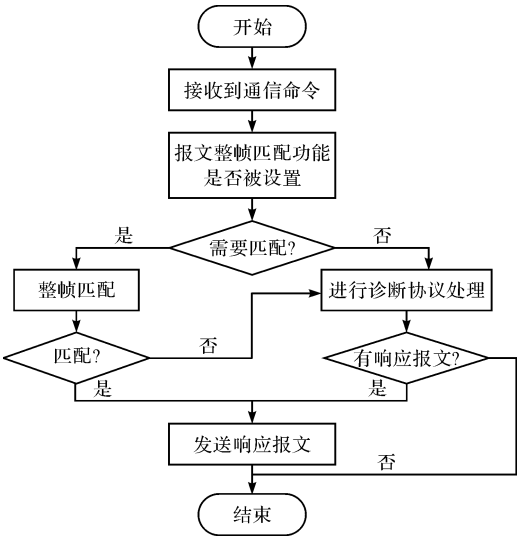

图 9 每层中命令处理的活动图

整帧的匹配操作有着最高优先级,如果它在每个工作的层中被包含,那么对于接收到的每个消息,整帧将被首先进行匹配。如果该消息没有匹配已存在的任何命令,则该层将进行下面的协议处理,如果发现相应的响应报文,则发送响应报文。

4.4 类 CProtocolLayer 设计

CProtocolLayer 类是消息的协议处理的基本定义,CProtocolLayer 是一个抽象类,它没有任何实例,一些方法被定义成纯虚函数。

这些诊断协议类继承自 CDiagProtocol,主要诊断协议子类型如表 1 所示。

表 1 继承自 CDiagProtocol 类的诊断协议类

序号	诊断协议子类
1	CHDC_CAN
2	CH9X
3	CKwp2000Diag

CHDC_CAN 类—代表了 ISO15765 协议;CH9X 类—代表了 Honda 系列协议;CKwp2000Diag 类—代表了 Keyword2000 协议

4.5 类 CNodeLayer 设计

本类继承自 CProtocolLayer,主要成员函数如表 2 所示。

表 2 CNodeLayer 类的主要成员函数

序号	成员函数名
1	SetLayerDown( )
2	UpdateSend( )
3	UpdateRecv( )
4	PutIntoTXBuf( )
5	GetFromRXBuf( )
6	Configure( )

(1)CNodeLayer 的实例是在协议栈的各层中,它有一个可选的上、下层,根据 ISO 下层提供服务给上层,所以每个 CNodeLayer 实例都有一个处理它下层的句柄,SetLayerDown( )方法是被用来设置此类句柄;

(2)CNodeLayerBottom 和 CNodeLayerTop 继承自该类,它们组成了 CBusNode 中的协议栈里的各个诊断协议层,例如 ISO/WD 15765 协议则是基于 CAN 总线的 Keyword2000 协议,即该协议栈的数据链路层是 ISO 11898-1 CAN 协议,上层是由 Keyword2000 或 UDS 的应用层移植而成[5];

(3)UpdateSend( )和 UpdateRecv( )在主循环中被用于消息处理;

(4)PutIntoTXBuf( )和 GetFromRXBuf( )是用于推动消息传递;

(5)Configure( )采用一个具体的配置类来完成参数配置、定义和检查配置参数的有效性。

CIso15765_2Layer 和 CKwp2000DLLayer 两个类继承自 CNodeLayer。

CIso15765_2Layer 是 ISO15765 协议的网络层。ISO/DIS 15765:1999 年出台 ISO/DIS 15765(Diagnostics on CAN-based on KWP2000),该诊断标准是基于 ISO 14230 在 CAN 线上的扩充,源于 K 线的诊断标准。其基本原理是把 KWP2000 的应用层诊断服务移植到 CAN 总线,它的数据链路层采用 ISO 11898-1[6]。

CKwp2000DLLayer 是 KWP2000 的数据链路层。ISO 14230:ISO 14230 于 1999 年出台,该诊断标准基于 K 线,波特率为 10.4 kb/s,用单线(K 线)通信,也可用双线(K 线和 L 线)通信。ISO14230 的头格式不是固定的,有 3 或 4 个字节,报文传输不用分包,最大可传 255 个字节数据,K 线本质上是一种半双工串行通信总线[7]。

4.6 协议数据类型介绍

协议数据类型相关类是包含在命名空间 ProtoDataType 中,并且继承自 CBaseMgr。它们被用来作为 CDiagProtocol 实例的扩展属性。包含 CBaseMgr 在内,都是派生自 CGarbagable 类。

继承自 CBaseMgr 类的协议数据类型如表 3 所示。

表 3　继承自 CBaseMgr 类的协议数据类型

序号	成员函数名
1	CUintData
2	CmdAndRes
3	CDataList
4	CMemBlock

CUintData—用于无符号整形数值属性;CmdAndRes—用于整帧匹配;CDataList—用于诊断协议的数据模型

CMemBlock 用于诊断协议的内存模型。

5　软件数据验证

5.1　ECU 模拟器端配置的诊断数据

数据验证所使用的模拟器配置文件中 A/C ECU 的 DID 数值设置相关的版本信息部分如表 4 所示。

表 4　模拟器配置文件中 A/C ECU 的 DID 数值设置——(读取版本信息)

	DID	DID 描述	DID 对应的数值 N
1	F010	车辆网络配置数据标识符	0×45,0×41,0×56,0×2D,0×36,0×44,0×42,0×2D,0×4C,0×32,0×30
2	F18C	ECU 序列号	0×45,0×44,0×43,0×37,0×2D,0×36,0×44,0×4C,0×32,0×30
3	F192	硬件版本号	0×48,0×30,0×2E,0×32,0×2E,0×31
4	F195	软件版本号	0×30,0×33,0×2E,0×30,0×31,0×2E,0×30,0×32

数据验证所使用的模拟器配置文件中 A/C ECU 的 DID 数值设置相关的数据流部分如表 5 所示。

表 5　模拟器配置文件中 A/C ECU 的 DID 数值设置——(读数据流)

	DID	DID 描述	单位	DID 对应的数据换算公式及测试数值
1	0×1226	加热器水温	℃	E=0.01*N(1 200)
2	0×122E	发动机转速	r/min	E=0.01*N(3 500)
3	0×1247	蒸发器温度	℃	E=N(10 048)
4	0×124B	散热器出口阀门请求温度	℃	E=0.01*N-82.48 (36 089)

模拟器端数据验证工作原理如下:ECU 模拟器根据表4 中的 ECU 版本信息构造相关的 DID(数据标识符)版本数据,根据表 5 中的数据换算公式构造相关的 DID 数据,并把构造的测试数据写入它的诊断数据配置文件 DVP(自定义的文件格式)中。在 ECU 模拟器和 DiagAnalyzer 进行诊断通信会话时,它将加载 DVP 文件中的诊断数据,通过故障诊断仪软件完成对 ECU 模拟器的测试。

5.2　诊断仪端显示的模拟测试结果

诊断仪显示的 A/C ECU 的 DID 数值设置相关的版本信息部分如表 6 所示。

表 6　诊断仪显示的 A/C ECU 的 DID 数值设置——(读取版本信息)

	DID	DID 描述	DID 对应的数值 E
1	F010	车辆网络配置数据标识符	EAB-6DB-L20
2	F18C	ECU 序列号	EDC7-6DL20
3	F192	硬件版本号	H0.2.1
4	F195	软件版本号	03.01.02

诊断仪显示的 A/C ECU 的 DID 数值设置相关的数据流部分如表 7 所示。

表 7　诊断仪显示的 A/C ECU 的 DID 数值设置——(读数据流)

	DID	DID 描述	单位	DID 对应的数值 E
1	0×1226	加热器水温	℃	12
2	0×122E	发动机转速	r/min	3 500
3	0×1247	蒸发器温度.	℃	28.00
4	0×124B	散热器出口阀门请求温度	℃	33.21

5.3　测试的 ECU 集合

该 ECU 模拟器可以对所有采用了 Keyword 2000 和 UDS 诊断协议[8] 的 ECU 进行诊断数据的仿真,同时为了充分验证程序的正确性,笔者采用了在北美地区销售的斯巴鲁某车型中的 ECU 集合进行了验证。测试 DataManager 所涉及的 ECU 集合如表 8 所示。

表 8　测试 DataManager 所涉及的 ECU 集合

	ECU 名称	协议	ECU 说明
1	Air Conditioner	CAN	车载空调
2	ABS VDC	CAN	汽车防抱死系统/车辆动态控制系统
3	Equipment	CAN	包括:Auto Light & Wiper,Power Window,Auto Headlight Leveling,Power Rear Gate
4	Auto Start Stop	CAN	自动启停控制
5	Body Integrated Module	CAN	BCM 控制模块
6	Electric Power Steering	CAN	电子助力转向系统
7	Keyless Access & Push Start	CAN	无线钥匙/引擎触控启动系统
8	Meter	CAN	该模块包括了空气温度、燃油效率修正值、驾驶员/乘员的安全带绑定状态、车速和发动机转速等测量指标
9	Occupant Detection System	CAN	驾驶员/乘客检测系统
10	Power Unit	K line	动力单元
11	SRS Airbag	K line	电子安全气囊
12	Tire Pressure Monitoring System	K line	胎压监控系统

每个 ECU 所测试的内容包括：DID 数值设置(ECU 版本号，软硬件版本号等)，显示数据流，读取和清除 DTC 故障码等项目。经过测试诊断结果显示内容与 ECU 模拟器中配置的故障码的故障描述及故障信息在个数和内容上完全一致，且诊断结果显示内容中的数据值不断刷新显示，即在执行中连续读取数据，诊断结果相对 ECU 配置的数据值具有较高的精确度，因此读取故障码诊断功能实现，且其诊断结果相对 ECU 模拟器的故障码和数据流更准确、可靠。

6 结束语

本研究开发了 ECU 模拟器及其与诊断仪通信接口，具有如下优点：通过该模拟器，开发工程师只要在软件的数据配置文件中根据故障诊断协议写入各种诊断数据，即可完成对特定车型诊断数据的开发和调试，只需要一台 PC 机就可以完成汽车上各个 ECU 的诊断数据开发；同时该 ECU 模拟器支持目前市场主流的故障诊断协议。另外，由于该 ECU 模拟器遵循 J2534 接口协议，可以和其他遵循该协议的故障诊断软件进行匹配和通信。

具体结论如下：

(1)该 ECU 模拟器支持 UDS 和 Keyword2000 等诊断协议，用户只需要按照指定格式修改 DVP 配置文件即可实现新 ECU 诊断数据的加载；

(2)通过使用 J2534Inject. dll，可以在电脑平台上完成 DiagAnalyzer 故障诊断仪和 ECU 模拟器之间的诊断数据通信，省略了硬件转换模块；

(3)在模拟器的 GUI 操作界面上开发了多种数据展示和修改方式，可以提高工作效率；

(4)在一个被模拟的 ECU 上可以设置多个信号，并且给特定信号指定最大值、最小值、回包中所处的位置等。

参考文献(References)：

[1] SAE J2534-2. Optional pass-thru features[S]. New York: SAE, 2010.

[2] SAE J2534-1. Recommended practice for pass-thru vehicle programming word[S]. New York: SAE, 2004.

[3] RICHLER J, NASARRE C. Windows 核心编程[M]. 5 版. 北京：清华大学出版社，2008.

[4] 罗 峰，孙泽昌. 汽车 CAN 总线系统原理设计与应用[M]. 北京：电子工业出版社，2010.

[5] ISO 15765-1. 2004 Road vehicles—diagnostics on controller area networks (CAN)—Part 1: general information[S]. Geneva: International Organization for Standardization, 2004.

[6] ISO 15765-3. 2004 Road vehicles—diagnostics on controller area networks (CAN)—Part 3: implementation of unified diagnostic services (UDS on CAN)[S]. Geneva: International Organtiation for Stomdardization, 2004.

[7] 常欣红，于金泳，刘志远. 汽车故障诊断标准 ISO15765 的网络层分析与实现[J]. 汽车技术，2006(9)：40-44.

[8] 罗 峰，苏 剑，袁大宏. 汽车网络与总线标准[J]. 汽车工程，2003(4)：372-376.

[**编辑**：李 辉]

本文引用格式：

黄利权，赵治国. 基于 J2534 协议的诊断仪与 ECU 模拟器接口设计[J]. 机电工程，2018，35(10)：1116 - 1122.

HUANG Li-quan, ZHAO Zhi-guo. Design of interface between ECU simulator and fault diagnosis device based on J2534 protocol[J]. Journal of Mechanical & Electrical Engineering, 2018, 35(10): 1116 - 1122.

《机电工程》杂志：http://www.meem.com.cn