基于 J1939 协议的车辆故障诊断与 ECU 报文解析

汪志斌，吴长水，黄敏涛，冯琛

（上海工程技术大学 汽车工程学院，上海 201620）

摘要：在对汽车的故障诊断过程中，基于 SAE J1939 协议的 CAN 通信的 ECU 提供的发动机性能检测参数和整车网络通信数据，实现整车网络中多个 ECU 数据的共享；J1939 协议同时也支持故障的诊断，通过数据转换模块将接收的数据转换成串行数据（包含 CAN 的 ID 地址），诊断工具（手持终端）可以读取当前故障码 DM1 或清除当前故障码 DM11。本文提出了一种车辆故障诊断的研究策略，同时提出了一种基于 JAVA 语言的报文的解析方法，能够有效实时地实现对汽车发动机的故障检测。

关键词：CAN 通信；ECU；SAE J1939 协议

中图分类号：U46　　**文献标识码**：A

Automobile Failure Diagnosis and Analysis Trouble Code from ECU Based on J1939 Protocol

Wang Zhibin，Wu Changshui，Huang Mintao，Feng Chen

（College of Automotive Engineering，Shanghai University of Engineering Science，Shanghai 201620，China）

Abstract：In the process of vehicle fault diagnosis，the CAN communication ECU based on SAE J1939 protocol can provide engine performance detection parameters and vehicle network communication data to realize the sharing of multiple ECU data in vehicle network. The J1939 protocol also supports faulty diagnosis. The data is converted into the serial data（including the CAN ID address）through the data conversion module. The diagnostic tool（handheld terminal）can read the current fault code DM1 or clear the current fault code DM11. In this paper，a research method of vehicle diagnosis is proposed. At the same time，an analytical method based on JAVA language is proposed，which can effectively detect the fault of automobile engine in real-time.

Key words：CAN communication；ECU；SAE J1939 protocol

引　言

SAE J1939 协议专供卡车及其拖车、大客车等商用车使用，是用来支持分布在车辆各个不同位置的电控单元之间实现实时闭环控制功能的高速通信标准，包括通信层、物理层和数据链路层，以 CAN2.0B 为基础，数据传输速率可达 250 kbps。另外，它还定义了网络层和应用层的协议，是目前大型汽车中应用最广泛的应用层协议，同样能够实现故障诊断、故障处理。

1　SAE J1939 协议简介[3]

1.1　原理简介

SAE J1939 协议由美国汽车工程师协会（SAE）制定，目前在载货汽车及客车等重型车辆中广泛应用，其以 CAN2.0B 作为网络核心协议，在其基础上定义了网络层和协议层，遵循 7 层 OSI 网络结构，并对每个被实现的层使用不同的文件进行描述，通过提供一个标准的框架，使电控单元 ECU 之间可以实现网络互联通信而不需要额外的功能接口。表 1 介绍了 CAN2.0 的标准和扩展格式，及 J1939 协议所定义的格式。J1939 协议报文单元的具体格式如下所示：

PRIORITY	R	DP	PDU FORMAT	PDU SPECIFIC	SOURCE ADDRESS	DATA FIELD
3	1	1	8	8	8	0～64

可以看出，J1939 标识符包括：PRIORTY（优先权位）；R（保留位）；DP（数据页位）；PDU FORMAT（协议数据单元）；PDU SPECIFIC（扩展单元）和 SOURCE ADDRESS（源地址）。而报文单元还包括 64 位的数据场。

1.2　J1939 协议在系统 ECU 中的应用

基于 CAN 通信的 J1939 协议的 ECU 能提供发动机性能检测参数和整车网络通信，实现整车网络中多个 ECU

表 1 CAN2.0 的标准和扩展及 J1939 协议的格式

CAN 扩展帧格式	SOF	11 位标识符				SRR	IDE	18 Extension identifier		
J1939 帧格式	帧起始位置	优先权 3 位	R 位(保留)	数据页 DP	PF 格式 6 位	SRR 位	扩展标识	PF	PS 格式(8 位)	源地址 8 位
CAN	1	2～4	5	6	7～12	13	14	15～16	17～24	25～32
帧位置	/	28～26	25	24	23～18	/	/	17～16	15～8	7～9

的数据共享,同时 J1939 协议支持故障诊断,通过诊断工具可以读取或清除诊断故障码[4]。系统 ECU 主要采用两种通信方式:单帧通信和多帧通信。在检测和整车网络通信时主要采用单帧通信方式;在诊断时因灵活的故障码个数是可变的,因此单帧通信与多帧通信结合使用。

2 下位机对故障码(报文)的接收与发送

2.1 CAN 转蓝牙控制器模块

CAN 转蓝牙控制器模块(简称 BluetoothCAN)接收来自终端的命令(串行数据),转换成对应的 CAN 总线格式的数据帧,发送给发动机控制器 ECU。其次,BluetoothCAN 接收来自 ECU 的数据,转换成对应的串行数据(包含 CAN 的 ID 地址),发送给终端。

2.2 接收与发送设计

数据接收与发送使用 CAN 转蓝牙控制器模块,通过 OBD(On Board Diagnosis、车载诊断系统)接口(CAN 总线)和发动机控制单元(ECU)进行通信,读取来自 ECU[5] 的信息并进行处理,转换成蓝牙数据发送给手持终端;同样,手持终端发送数据信息给 BluetoothCAN,BluetoothCAN 转换成 CAN 报文信息发送给发动机控制器 ECU,实现手持终端与 ECU 的信息交互。原理示意图如图 1 所示。

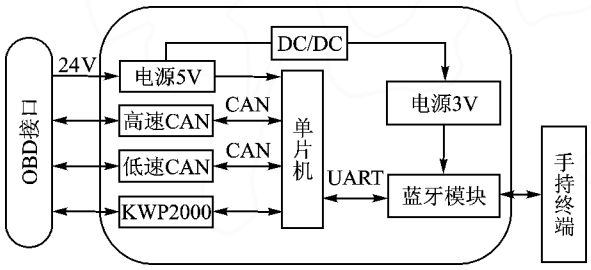

图 1 故障诊断过程

3 故障码的解析

3.1 诊断故障码定义

J1939 协议包括在线故障诊断功能,由诊断应用层定义。系统在进行故障诊断时主要采用 4 种消息帧实现故障诊断,分别为 DM1 发送当前故障码、DM2 发送先前故障码、DM3 清除先前故障码和 DM11 清除当前故障码。J1939 协议诊断故障码(DTC)由 4 个独立域构成,同时也是由 4 个字节(32 位)表示,这 4 个部分如表 2 所列。

表 2 DTC 的构成

顺 序	特殊位	位 数
a	可疑参数的编号(SPN)	19
b	故障模式标志(FMI)	5
c	可疑参数编号的转化方式(CM)	1
d	故障发生次数(OC)	7

其中 SPN 为发生故障的部件,FMI 为发生的故障类型,CM 为 SPN 转换排列规则,OC 为本故障的发生次数。OC 取值范围为 0～126,即使故障发生次数大于 126 时,OC 也保持为 126。

3.2 诊断故障码请求

J1939 协议规定请求参数组编号 PGN 为 59904,诊断工具地址为 0x2B,ECU 地址为 0x00。系统 ECU 在接收到诊断设备的请求后,在中断服务程序中对请求帧所请求的目标 PGN 内容进行判断,如果请求的是读取先前故障 DM2,那么系统 ECU 将向诊断工具发送先前故障码;如果请求的是清除先前故障码 DM3 或当前故障码 DM11,那么系统 ECU 将清除先前故障码或当前故障码。请求帧数据、请求参数组格式分别如表 3、表 4 所列。

表 3 请求帧格式

位定义	位 数	值
优先级	3	6
R	1	0
DP	1	0
PF	8	EA
PS	8	00
SA	8	2B
数据域/8 字节	1～3,4～8	请求参数组 PGN FF

表 4 请求参数组

字节 / 参数	byte1	byte2	byte3
DM2	CB	FE	00
DM3	CC	FE	00
DM11	D3	FE	00

其中 PGN 为 0x00 FECB 时为先前故障码,PGN 为 0x00 FECC 时为清除先前故障码,PGN 为 0x00 FED3 时

为清除当前故障码。

当诊断工具向 ECU 请求先前故障码或清除故障码时，诊断工具发送请求 PGN 和被请求的 PGN 给 ECU 电控单元，ECU 给诊断工具回复相应的数据。诊断工具与 ECU 之间的数据交互如图 2 所示。

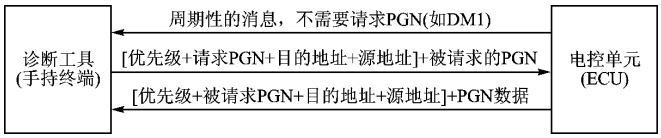

图 2　故障码的请求与发送

3.3　手持终端接收故障码

诊断时，当前故障和先前故障所包含的故障码有三种情况：无故障、一个故障和多个故障。无故障和一个故障时，使用单个 CAN 数据帧就可以发送全部数据，而当传输多个故障时，需要使用多个数据帧才能发送完毕，这时就需要使用 J1939 协议的传输协议功能。J1939 协议通信的核心是负责数据传输的传输协议，传输协议功能主要包含三个部分：消息的拆装、数据重组和连接管理。消息的拆装是指无法用单个 CAN 数据装载全部的数据，需要拆分为多个数据帧。当多个故障在使用传输协议传输时，第一个字节为数据帧编号，其后为故障灯状态，接着是所要发送的故障码 DTC，多个 DTC 按顺序填充到数据帧中，如果最后一个数据帧中的数据字节不到 7 位，那么就用 0xFF 来进行填充。数据重组就是将接收到的数据帧按照序列编号把多包消息的数据帧重新组合成原始数据，按照编号排列，第一个数据帧中的第 2、3 字节为故障灯状态，从第 4 个字节开始每 4 个字节为一个故障码，不足 4 个字节则从下一个数据帧的第 2 个字节开始读取。

当没有故障发生的时候或者单个故障发生时，此时 DM1 和 DM2 就可以直接使 CAN 数据帧发送相应的故障码数据；当有多个故障时，采用 BAM 多包方式发送，首先发送一条 BAM 公告信息，接着使用数据传输 PGN 多包发送故障码。

单个故障采用单包发送的方式，使用单个 CAN 数据帧发送相应故障码。数据格式如表 5 所列，DM1 表示当前故障，ID=0x18FE CA00；DM2 表示先前故障，格式与 DM1 相同。

表 5　单包数据格式

位定义	位　数	值
优先级	3	6
R	1	0
DP	1	0
PF	8	FE
PS	8	CA
SA	8	00
数据域/8 字节	1,2,3～6,7,8	XX FF DTC FF FF

多个故障采用多包发送方式，首先系统 ECU 发送 BAM 公告信息，接着发送多个数据帧。数据拆装时每个数据帧的第一个字节为数据帧编号，其余 7 个字节存放故障码数据，其中数据字节为故障码的有效字节数。BAM 信息格式、多包信息格式如表 6、表 7 所列。

表 6　BAM 信息格式

位定义	位　数	值	ID 号，位意义
优先级	3	7	0x1CECFF00
R	1	0	
DP	1	0	
PF	8	EC	
PS	8	FF	
SA	8	00	
数据域/8 字节	1,2～3,4,5,6,7,8	20 LSB MSB XX FF ///	控制字节 消息字节数 数据包个数 保留 打包消息 PGN

表 7　多包数据格式

位定义	位　数	位　值	位定义	位　数	位　值	位定义	位　数	位　值
优先级	3	7	优先级	3	7	优先级	3	7
R	1	0	R	1	0	R	1	0
DP	1	0	DP	1	0	DP	1	0
PF	8	EB	PF	8	EB	PF	8	EB
PS	8	FF	PS	8	FF	PS	8	FF
SA	8	00	SA	8	00	SA	8	00
数据域/8 字节	1,2,3,4～7,8	SN=1 LAMP 0xFF DTC1 DTC2 字节	数据域/8 字节	1,2～4,5～8	SN=2 DTC2 字节 2～4 DTC3……	数据域/8 字节	1,2～M…	SN=N DTC～M FF

4　故障码的解析方法

4.1　故障解析流程图

故障解析流程图如图 3 所示。

4.2　解析方法的程序设计

手持终端收到数据包时，要对数据包进行解析，显示出发动机中对应的故障，从而实现故障的诊断，方便故障维修者对其进行维修。故障码诠释如表 8 所列。

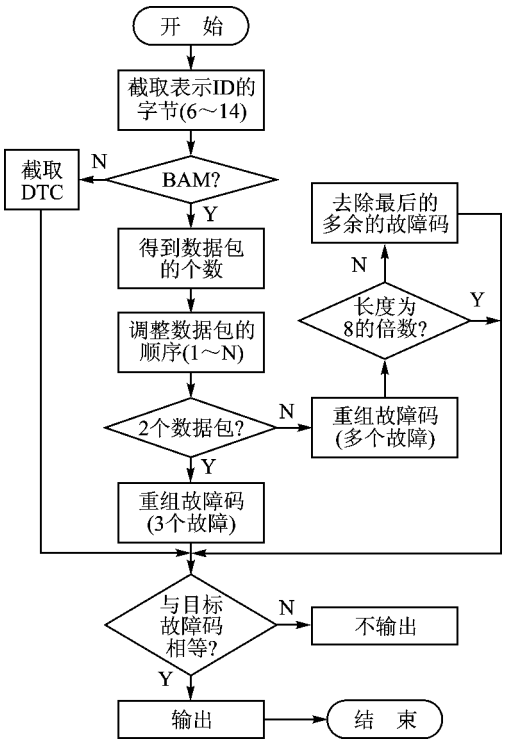

图 3 故障解析流程图

表 8 故障码诠释

部 件	描 述	J1939DTC
曲轴传感器	无曲轴信号故障	0x010C00BE
第一缸喷油电磁阀	喷油阀对地对短路	0x0104028B

在对报文的分析中，采用 JAVA 语言开发的 eclipse[6] 软件，提出一种解析报文的算法，能有效地解析接收的各种报文，显示对应的故障源。

由于终端接收到的数据报文为十六进制数，将接收到的报文转换为字符串进行处理。方法如下：

① 使用 length()函数[7]取得字符串长度。

② 根据字符串长度判断为单帧数据还是多帧数据。

③ 当为单帧数据时，使用 regionMatches()，截取字符串的有用故障码，与目标故障码进行比较，输出对应的故障源。

④ 当为多帧数据时，使用 substring()函数从 BAM 信息中截取数据包个数有用字符，并使用 Integer.parseInt()函数将其转换成十进制数，即数据包的个数。

⑤ 根据数据包的个数分为两帧数据和两帧以上数据，需要将故障码截取、拼凑，再与目标故障码进行比较，从而得到对应的故障源。

JAVA 代码解析方法如下：

```
package org.xs.date;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.*;
public class DateStore {
    //列举几种目标故障码
Static String J1939DTC[]=
{" 0x00000000"," 0x010C00BE",
" 0x010200BE"," 0x010C02D3"," 0x010202D3"};
//故障码对应的故障部件
static String 故障部件[]={" 无事件" ," 曲轴传感器" ," 曲轴传感器" ," 凸轮传感器" ," 凸轮传感器" };
//故障码对应的故障源
static String 故障源[]={" 无事件" ," 无曲轴信号故障" ," 曲轴信号异常" ," 无凸轮信号故" ," 凸轮信号异常故障" };
public static void main(String[] args)
//TODO Auto-generated method stub
        throws IOException{
//生成 BufferedReader()函数对象(手动从控制端输入标准数据
//帧进行模拟)
BufferedReaderbr =New BufferedReader(new InputStreamReader(System.in));
//新建一个字符串数组对象
    String str[]=new String[1];
//str[0]中存放手动输入的字符串故障码
    str[0]=br.readLine();
//DTC1 表示截取拼凑的故障码
    String DTC1=" ";
//XX 表示数据包的个数
    String XX;
//LSB 表示字符串的长度
    int LSB =str[0].length();
        switch(LSB){
//单帧数据处理
    case 32:
     for(int i=0;i<J1939DTC.length;i++){
        if( J1939DTC[i].regionMatches(2,str[0],18,8))
        System.out.println(" [" +故障部件[i]+"]" +"  " +故障源[i]); break;
//多帧数据处理
    default:
//从 BAM 中读取数据包的个数，截取表示数据包个数的字节
    XX=str[0].substring(20,22);
//将字符串转换成十进制
    int XXD=Integer.parseInt(XX,16);
        switch(XXD){
//2 帧处理数据
        case 2:
//将故障码拼凑起来，与目标故障码进行比较
    DTC1=str[0].substring(52,62)+str[0].substring(80,94);
    for(int j=0;j<J1939DTC.length;j++){
//k 表示 DTC1 的字节数
```

```
    int k=0;
    while(k<24){
      if(DTC1.regionMatches(k,J1939DTC[j],2,8)){
      System.out.println(" [" +故障部件[j]+"]" +"  " +故
障源[j]); }
       k=k+8;}}
    break;
    default:
DTC1=str[0].substring(52,62)+str[0].substring(80,94);
    int m=2;
    while(m<XXD){
//两帧以上数据的故障码拼凑公式
    DTC1=DTC1+str[0].substring(96+32*(m-2)+16,96
+32*(m-2)+16+14);
        m++; }
//判断 PC 是否为 8 的倍数
    if(DTC1.length()%8==0){
        for(int j=0;j<J1939DTC.length;j++){
            int k=0;
        while(k<10+14*(XXD-1)){
    if(DTC1.regionMatches(k,J1939DTC[j],2,8)){ System.
out.println(" [" +故障部件[j]+"]" +"" +故障源[j]);}
        k=k+8;}}}
    else if (DTC1.length()%8!=0){
        for(int j=0;j<J1939DTC.length;j++){
        int k=0;
        while(k<8*(DTC1.length()/8)){
          if(DTC1.regionMatches(k,J1939DTC[j],2,8)){
                System.out.println(" [" +故障部件[j]+"]
" +"" +故障源[j]);
}
    k=k+8;}}}}}}}}}
```

4.3 程序调试

手动输入：

①（单帧数据）FEFF0818FECA00FFFF010C00BEFFFFFF

输出：[曲轴传感器] 无曲轴信号故障

②（两帧数据）FFFFFF1CECFF00FFFFFF02FFFFFFFFFFFFFFFF18FECA00C00BE01FFFFFFFF18FECA00FF0200BE010C02D3FF

输出：[曲轴传感器] 无曲轴信号故障

[曲轴传感器] 曲轴信号异常

[凸轮传感器] 无凸轮信号故

③（六帧数据）FFFF01E4F00601FFFFFFFF18FECB00FFEFFFFFFFF18FECB00FF01E3F00501E4F0FFFFFFFF18FECB00FF0501E5F00501E6FFFFFFFF18FECB00FFF005010D029001FFFFFFFF18FECB00FF04029001050290FF

输出：

[第 6 缸喷油电磁阀] 喷油阀第 6 缸两端开路

[第 6 缸喷油电磁阀] 喷油阀第 6 缸对地短路

[第 6 缸喷油电磁阀] 喷油阀第 6 缸阻抗超限

[PCV 阀 1 电子驱动] PCV 阀 1 两端短路

[PCV 阀 1 电子驱动] PCV 阀 1 开路

[PCV 阀 1 电子驱动] PCV 阀 1 对地短路

[PCV 阀 1 电子驱动] PCV 阀 1 对电源短路

[PCV 阀 2 电子驱动] PCV 阀 2 两端短路

[PCV 阀 2 电子驱动] PCV 阀 2 开路

[PCV 阀 2 电子驱动] PCV 阀 2 对地短路

④ ……

以上只对部分故障码进行了模拟，此种方法适用于收到的所有(1～N)数据帧。

结 语

汽车故障诊断技术是汽车电子控制技术的重要组成部分。随着电子行业的逐渐发展，系统故障诊断方法变得更加智能，很多现代化的工具在汽车故障诊断上得到了应用，在实现故障诊断的设计过程中，方法各异。而本文是基于 J1939 协议，综合利用 ECU 与 CAN 转蓝牙通信[8]模块，手持终端(智能手机)接收蓝牙模块的报文信息，实现故障的有效诊断，相对于传统的诊断仪来说，更趋向于智能化的发展方向。

注：此论文受“大功率多点电喷气体发动机喷嘴阀及控制技术”项目资助。

参考文献

[1] 康拉德 赖夫.汽车电子学[M].3 版.李裕华，译.西安：西安交通大学出版社，2011.

[2] 罗富坤.汽车故障诊断技术[M].北京：化学工业出版社，2009.

[3] W 齐默尔曼，R 施密特加尔.汽车总线系统[M].邓萍，译.北京：机械工业出版社，2011.

[4] 樊永强.汽车故障诊断与排除[M].长沙：中南大学出版社，2011.

[5] Sandoval Leon，Jairo A.Study of Transit Bus Duty Cycle and its Influence on Fuel Economy and Emissions of Diesel-Electric Hybrids[J].Mechanical Engineering，2011：123-128.

[6] 霍尔泽.Eclipse 集成开发工具[M].O' Reilly Taiwan 公司，译.南京：东南大学出版社，2007.

[7] 史斌星，史佳.JAVA 基础及应用教程[M].北京：清华大学出版社，2007.

[8] 吴海东，梅海龙.汽车车载网络技术与检修[M].北京：北京理工大学出版社，2010.

汪志斌(硕士研究生)，主要研究方向为汽车故障诊断；吴长水(副教授)，主要研究领域为内燃机排放控制。

(责任编辑：薛士然 收稿日期：2017-06-23)