

# 基于多功能车辆总线的动车组自动化调试系统的设计与实现

赵建博<sup>1</sup>, 沈华波<sup>1</sup>, 李超<sup>2</sup>

(1. 中车青岛四方机车车辆股份有限公司 技术工程部, 山东 青岛 266000;

2. 航天科工海鹰集团有限公司, 北京 100070)

**摘要:** 在动车组调试过程中, 需要实时获取车辆状态数据, 但由于车内系统接口的不完善, 导致车辆状态数据获取困难, 难以有效地辅助调试工作, 为了使调试过程中车辆数据可以得到实时的采集与分析, 并结合数据采集流程制定自动化调试流程, 基于多功能车辆总线的自动化调试系统可以在实现调试流程标准化的前提下, 实现系统与总线数据环路的实时交互, 并在调试流程自动执行的过程中, 实时获取并分析车辆状态, 通过验证分析, 该系统可以实现车内 12 个分系统将近 200 个状态参数的实时获取与分析, 并提高车辆调试效率 30% 以上。

**关键词:** 多功能车辆总线; 自动化调试系统; 数据交换

## Design and Realization of the Automatic Test System for EMU Train Based on MVB DataFlow

Zhao Jianbo<sup>1</sup>, Shen Huabo<sup>1</sup>, Li Chao<sup>2</sup>

(1. Engineering department of CRRC Qingdao Sifang Co., LTD, Qingdao 266000, China;

2. HIWING Group Co., LTD, CASIC, Beijing 100070, China)

**Abstract:** In the process of EMU train test, the real time data of train is needed, but because of the interface of the data system does not meet the demand, it is difficult for the test workers to get the feedback data while testing. In order to get the status data of the train and realize the automatic test system, By the method of automatic test solution based on the MVB data flow, the test system can extract the useful data from MVB data flow with the standard test process, after the analysis of the data, the technical status of the EMU train can be get exactly and rapidly, which can improve the quality and efficiency of the EMU train test.

**Keywords:** MVB (multifunction vehicle bus); automatic test system; data exchange

## 0 引言

在动车组的生产、调试过程中, 为了保证车辆的质量和各个分系统技术状态的完好性, 需要对车辆进行深入的调试与分析, 在常规的调试过程中, 由于车辆自身数据接口的限制, 难以实现调试数据的自动化采集, 随着车内信息采集技术的发展, 当前的车辆已经具有一定的数据采集功能, 尤其是在中国标准动车组中, 其 MVB 网络中采集并存储了大量的车辆状态、故障等信息<sup>[1]</sup>, 数据接口的不断完善, 促进了车辆调试工艺的改进, 为了提高调试过程的质量与效率, 需要充分利用车辆数据接口, 实现数据采集、分析、流程执行等过程的自动化。

本文提出了一种基于 MVB 数据车上状态在调试流程自动执行与结果解析的方法, 并实现基于 MVB 数据的自动化综合调试系统, 在车辆调试过程中, 综合调试系统作为现场端调试负责人的操作台, 是现场调试工作的核心系统。综合调试系统负责编辑调试流程和调试要领书, 并将安排调试任务在现场进行分配。当调试任务开始后, 综合调试试验台作为现场端的数据管理中心和状态监视中心, 实时接收现场调试的各种数据进

行显示, 同时监控整车的调试状态, 并及时调整操作工人用移动终端共同完成调试任务。还可以根据 MVB 采集的车上状态信息和故障信息, 根据工艺流程来运行试验过程, 利用任务属性和车上状态的标准化数据解析, 把调试要领书中需要获取的调试数据进行自动采集, 并根据调试流程进行调试的自动化执行<sup>[1]</sup>。并通过自动采集故障信息, 能够快速分析成调试履历与故障诊断的记录, 完成调试任务的记录与分析, 有效地提高调试效率和调试信息化水平。

## 1 系统结构及原理

综合调试系统的核心技术是数据的采集技术与调试自动化执行方式, 它是基于 MVB 数据车上状态在调试流程自动执行与结果解析的方法, 是要解决的目前在调试过程中的车上状态数据采集问题, 突破现有人工调试的效率瓶颈, 并能够提供调试流程的自动化执行所需要的数据基础, 完成按照调试工艺流程的自动化执行<sup>[2]</sup>。

本方法具体内容是针对车上的 MVB 数据传输上来的 10 个系统状态信息和故障信息进行具体的数据解析与数据转化。根据不同车型在管理平台设置其初始化信息, 同时编辑工艺流程并根据解析的数据与调试要领书中的表头信息进行校验, 并根据调试规则进行调试结果的匹配。实现调试流程的自动化执行, 并且进行调试数据的自动填充和故障信息的自动采集。并能够在信息系统中生成调试履历与故障诊断的记录。

收稿日期:2016-01-06; 修回日期:2016-03-05。

作者简介:赵建博(1985-),男,山东金乡人,工程师,主要从事动车组电气调试方面的研究。

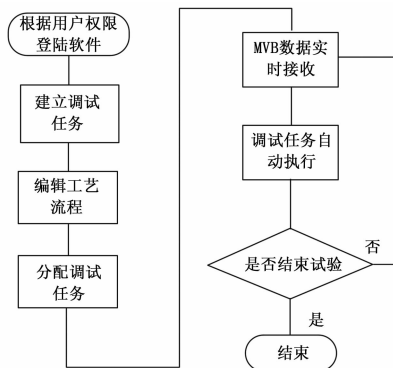


图 1 调试流程图

## 2 硬件设计

### 2.1 总体设计

本系统是针对车辆进行调试的综合调试信息管理系统，系统采用调试管理平台、现场操作试验台、分布式数据采集设备三级结构实现，管理平台主要实现系统中所有数据的存储与管理，实现调试任务的管理与维护。当调试任务开始后，综合调试系统作为现场端的数据管理中心和状态监视中心，实时接收现场调试的各种数据进行显示，同时监控整车的调试状态，并及时根据需求的变化调整调试任务。现场操作端为调试工人提供调试任务下载、查看、记录与反馈的操作环境，通过 MVB 采集设备对车上状态信息与故障信息配合对操作流程进行校验，支持按照操作提示完成相应步骤完成调试和自动执行调试流程进行调试，同时现场端调试人员采用便携式移动终端的方式进行质检记录与故障信息的提交。

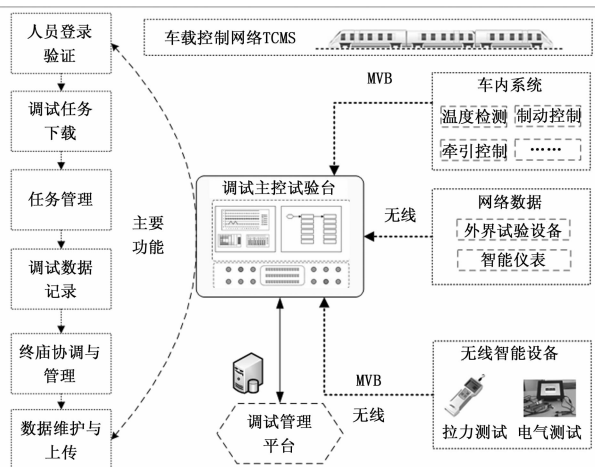


图 2 调试终端与调试主控试验台实现原理

### 2.2 模块设计

本系统主要包括任务执行模块、车上状态监控模块、流程控制与分析模块。任务执行模块主要进行调试任务的规划与制定，并根据实际的业务要求进行管理，车上状态监控模块主要实现车内 MVB 网络数据的实时采集，并根据对应的规则进行数据提取与分析，流程控制模块主要依据调试流程的逻辑关系进行设备的调用与控制，并实现对应数据的传输。

本系统在设计过程中需要实现车内多个分系统中不同类型数据的实时采集与分析，由于数据类型、数据协议种类很多，

且需要实现数据与调试流程的关联与实时交互，所以在系统设计过程中，需要针对数据与流程的关系，制定标准化的数据分析规则，并实现规则的定制与扩展，便于系统的扩展应用<sup>[8]</sup>。

#### 2.2.1 任务执行模块设计

任务执行模块的主要功能是建立调试任务，并对建立的调试任务，可以编辑调试工艺流程，并根据调试工艺流程进行调试要领书的任务分配。还可以同时对调试过程的每一个阶段进行监控与管理。并能够支持在线状态的自动化调试任务执行，通过对对工艺调试流程文本的解析、控制测试仪器输出、控制采集仪器采集数据，并与调试任务的特征值相匹配，调试结果的自动填充，给调试终端和综合调试系统传输数据并存储在数据库中，实现调试流程的自动化执行。

1) 调试任务管理：通过调试车间总线网络，对调试任务进行具体分配和管理，即调试负责人根据任务计划，当前与其建立联系的调试操作人员数量，进行具体调试任务的分配。

2) 调试流程管理：通过从综合调试系统下载下来的工艺文件、调试任务，对调试流程进行顺序变更与管理。

3) 调试执行：通过从 MVB 采集的数据，需要选择某一工艺流程，按照操作提示完成相应步骤，一个工艺流程结束后，可将试验数据通过无线网发送给手持终端和调试信息管理平台，通过解析后的车上状态信息与故障信息，会根据调试要领书中的项目信息、操作步骤信息、确认信息进行特征值的匹配，匹配后，在进行调试结果的验证，每一个调试结果（正则表达式）都会对应唯一的车上数据，在车上状态数据完全对特征值进行校验后，完成数据的自动匹配与结果的录入<sup>[4]</sup>。

4) 过程监控：系统可以根据分布式调试终端采集上来的调试人员信息、调试任务数据和故障信息等调试基础数据，经过综合调试系统中的算法分析与统计，以列表、柱状图、饼状图等形式，对调试任务的整体完成情况和各调试流程的具体完成情况进行展示。

5) 调试任务报表输出：综合调试系统在调试过程中采集的各种数据和相关信息，最终以调试要领书的形式进行展示和查看。经过现场操作负责人确认后，以调试要领书模板的形式进行输出打印，并签字确认。

#### 2.2.2 车上状态监控模块设计

车上状态监控主要是对分布式调试终端和车内总线发送的数据进行接收，经过数据判读后，以图形或者列表的形式显示在界面上，针对 MVB 采集的数据，进行数据分析，并通过数据协议解析后，分别按照可视化原则，进行了温度、状态、运行情况、速度、时间等信息进行界面展示。通过把实时收到的数据，按照位偏置和字偏置的顺序，对每一位进行单独的解析，并分别针对十六进制数、八进制数、十进制数、ASCII 码等数值进行转化解析。这些显示会给现场操作负责人提供决策依据。主要功能包括：协议配置、数据解析和显示内容配置等功能。具体描述如下：

1) 协议配置：主要是配置车内总线数据的传送协议，以配置文件的方式记录下来，供数据解析时使用。

2) 数据接收和解析：调用协议配置文件，将实时接收的 MVB 采集的数据进行解析，一方面进行存储，另一方面实时显示到界面上，供现场操作负责人进行监视。

3) 显示内容配置：用户可以配置多个显示界面上显示的数据，通过配置文件保存下来，方便用户监视数据使用。

### 2.2.3 流程控制与分析模块设计

本模块主要实现调试业务流程的控制与数据分析, 主要实现硬件设备与业务流程、规则数据的分析与交互。

1) 流程读取与控制: 读取调试业务中的流程文件, 并根据对应的规则进行硬件设备的调用, 控制系统按照正确的流程进行调试;

2) 数据交互: 根据调试业务数据需求与数据提取规则, 实现硬件设备与上层任务管理系统的交互;

3) 数据分析: 对实时采集的数据按照其校验规则进行分析, 从而对调试过程中的数据准确性进行判定。

本模块通过通用、可配置的数据通信接口实现数据的采集, 包括 TCP/IP、Http、MVB 等多种类型的接口兼容<sup>[5]</sup>, 在数据分析过程中, 采用数据库对当前的数据规则进行存储, 并根据调试任务数据中的规则关联关系进行调用与分析。

## 3 软件设计

基于综合调试技术的研究内容与实现目标, 构建综合调试系统, 主要包括综合调试管理平台、调试主控试验台、分布式调试终端、试验设备等 4 部分。

### 3.1 软件结构

综合调试系统主要由综合调试管理平台、调试主控试验台、分布式调试终端、试验设备等组成。其中调试管理平台和数据通信服务部署在中心服务器上, 管理人员、工艺人员和质量人员都可以在随时随地登录到管理平台, 进行相关工作和查看相关的信息。调试主控试验台部署在调试车间, 由调试组长进行操作, 进行调试任务的现场部署与车辆数据监控, 调试终端和试验测试设备主要是放在生产调试车间内, 供调试现场的作业人员工作时使用, 试验台与调试终端配合, 共同完成车辆的调试工作。

调试信息化系统借助有线网络和无线网络相结合的工作环境, 实现调试工艺文件的编辑、调试工作的分配和管理、调试信息的分析和调试工艺的优化。

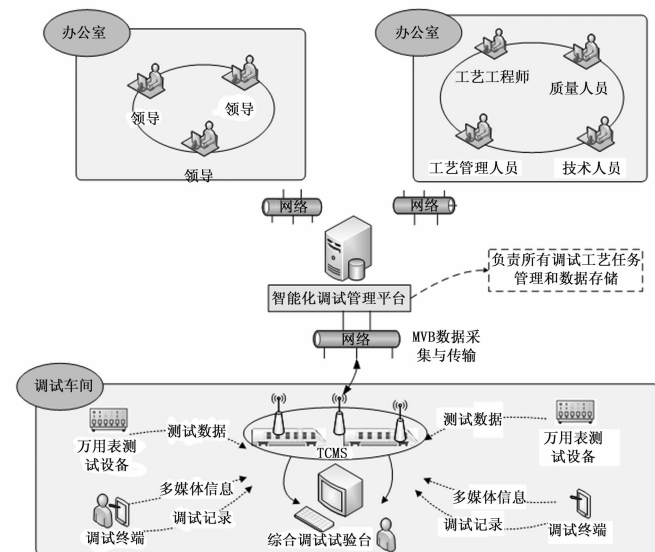


图 3 软件拓扑结构图

### 3.2 工作原理

根据项目实现目标、调试工艺需求, 综合调试系统采用中

心端、现场端的分层结构, 通过中心端、现场端的协同工作, 共同完成动车组调试自动化流程控制的目的。中心端负责调试计划、调试流程管理、过程监控、数据储存等操作, 现场端负责现场调试流程控制、调试命令发送接收、调试过程记录等, 整个系统采用 C/S、B/S 相结合的架构。

在系统的中心端部署了综合调试管理平, 为不同技术人员提供访问门户, 比如, 调试工艺人员可在中心应用信息系统中创建相关的工作任务、编制调试计划等; 调试负责人可进行调试过程进度查询等操作; 系统管理员可设置相关权限; 相关领导可查询调试完成进度情况、计划完成情况等。通过中心应用信息系统, 多名技术人员可同时进行协同访问, 协同工作, 有效提高了工作效率。管理平台通过数据库层与数据库系统进行交互, 实现数据的存储、提取。

现场端部署了调试主控试验台、分布式调试终端、试验设备。调试负责人使用调试主控试验台, 对现场操作人员在现场的分布式调试终端和试验设备, 通过无线网络实现调试流程的控制。调试负责人通过调试主控试验台查找相关的调试人员, 分配调试任务, 协助解决现场测试问题。操作人员利用分布式调试终端查询相关操作过程, 按照终端的操作提示, 完成操作步骤, 并记录在操作过程中产生的电压、电流值。也可以通过使用测试设备, 对一些测试数据进行现场采集与调试任务的自动填写。一次调试内容结束后, 系统自动/手动确认测试结果, 并进入到下一个调试环节。现场调试终端产生的中间调试数据信息, 将上传到中心端, 存入数据库中。

中心端与现场端之间通过通信服务来实现数据信息的传递、交互, 实现中心端的调试任务的下发、现场的调试过程数据的上传。

### 3.3 系统接口设计

#### 3.3.1 系统外部接口

为了提高系统的实用性, 并充分利用厂内现有数据资源, 提高调试工艺的智能化程度, 本系统需要如下的外部接口:

1) 调试终端与车载测试设备之间的接口: 实现车载测试设备的数据采集、记录与上传, 并通过数据与调试任务之间的关联, 实现调试过程数据记录的自动化。

2) 综合调试管理平台与公司内信息系统之间的接口: 如通过本接口实现人员信息、车辆信息、技术文件、调试记录在部门内管理平台与公司级平台(如 PDM、PLM 等)之间的数据关联和互相调用<sup>[6]</sup>。

3) 综合调试管理平台与公司内设备识别系统之间的接口: 实现对车辆位置、设备状态、人员信息等数据的获取和显示, 如通过 RFID、GPS 定位系统等<sup>[7]</sup>, 获得当前车辆的位置、设备安装状态、调试状态等。

#### 3.3.2 系统内部接口

本系统涉及管理平台、分布式调试终端、数据通信服务器、便携式测试设备 4 个平台, 在有线网络和无线网络的环境下进行交互, 其接口主要如下:

1) 综合调试管理平台与外部文件的接口: 实现调试工艺文件、车辆信息、技术资料等文件的导入和解析, 生成数字化的调试工艺信息, 包括可解析的调试任务(步骤、操作、确认项、判断规则等);

2) 综合调试管理平台与调试终端之间的接口: 实现调试任务的发布、下载、上传、数据匹配等过程, 通过标准的数据

通信协议接口实现本部分功能,将两者之间的所有的数据交互过程制定协议,并通过无线通信的方式实现;

3) 综合调试管理平台与便携式测试设备之间的接口:实现测试设备测试任务的发布、下载、记录与上传,并实现数据与调试任务之间的关联,需要以数字化后的调试任务的相关记录项为基础,将测试任务进行提取,并发布为测试设备可执行的任务,实现后续的测试和提交功能。

## 4 试验与结果分析

### 4.1 调试任务下载与维护

调试任务的下载与维护主要是实现综合调试系统对分配给自身的调试任务的下载与管理,调试任务可以通过无线局域网的方式实现数据的下载,当数据下载完毕之后,进行数据的存储。

### 4.2 调试任务执行

针对 MVB 采集的数据,需要把车上所有系统的状态特征信息按照车型、列号、车厢号、系统信息、故障信息、项目号的特征值提取出来,根据相关通信协议对数据进行解析,把相关的数据转化成调试结果需要的数据格式,再存储到数据库中或者写到 XML 文件中进行存储<sup>[8]</sup>,此时与综合调试系统下发的调试任务中的调试要领书特征值进行比对,通过匹配找到能够对应的调试要领书与车上状态数据或故障信息,然后调试状态数据会通过与数字化调试要领书中的调试结果正则表达式进行匹配,最后进行调试结果的自动填充,填充结束后,调试结果自动上传的数据库中并在综合调试系统中生成报表数据,完成调试后,车上状态数据会对下一个调试要领书的特征值进行校验,完成调试的自动化执行<sup>[9]</sup>。

如下的代码为针对要领书和相关数据进行存储的 XML 文件,在文件中实现了任务数据与调试结果数据的结构化存储。

```
<worksheet id="e4ba0ff7ad8a41bbb1bfb4ff1ff0456b" name="52
一监视器">
```

```
<sheetdata>
```

```
<row order="1">
```

```
<column id="E61A9B8DFABC425C97C81D91B484CA8F"
templateid="e4ba0ff7ad8a41bbb1bfb4ff1ff0456b" templateversion="1"
tagname="质检人员" startrowindex="1" startcolumnindex="1" end-
rowindex="1" endcolumnindex="2" ismergecell="Y" isedit="N" da-
tarule="" mergetype="Column" mergeid=""
E61A9B8DFABC425C97C81D91B484CA8F" rowmergenum="0"
columnmergenum="1" personcode="" time="" />
```

```
<column id="28F387B4113E44AAAE0E8CC137D447C"
templateid="e4ba0ff7ad8a41bbb1bfb4ff1ff0456b" templateversion="1"
tagname="车型" startrowindex="1" startcolumnindex="3" endrow-
index="1" endcolumnindex="4" ismergecell="Y" isedit="N" da-
tarule="" mergetype="Column" mergeid=""
28F387B4113E44AAAE0E8CC137D447C" rowmergenum="0" colum-
nmergenum="1" personcode="" time="" />
```

```
<column id="1D0B41BC919E44A78C21590C602E87F5" tem-
plateid="e4ba0ff7ad8a41bbb1bfb4ff1ff0456b" templateversion="1"
tagname="试验项目" startrowindex="1" startcolumnindex="5" end-
rowindex="1" endcolumnindex="6" ismergecell="Y" isedit="N" da-
tarule="" mergetype="Column" mergeid=""
1D0B41BC919E44A78C21590C602E87F5" rowmergenum="0" column-
mergenum="1" personcode="" time="" />
```

```
<column id="6A135D0B1C3A42AA82C1D9A5AACAA615"
templateid="e4ba0ff7ad8a41bbb1bfb4ff1ff0456b" templateversion="1"
tagname="项目页" startrowindex="1" startcolumnindex="7" end-
rowindex="1" endcolumnindex="7" ismergecell="N" isedit="N" da-
tarule="" mergetype="None" mergeid=""
6A135D0B1C3A42AA82C1D9A5AACAA615" rowmergenum="0"
columnmergenum="0" personcode="" time="" />
```

```
<column id="D5FE4E63ACA74A0CAED77C3F9A3D6148"
templateid="e4ba0ff7ad8a41bbb1bfb4ff1ff0456b" templateversion="1"
tagname="版本" startrowindex="1" startcolumnindex="8" endrowin-
dex="1" endcolumnindex="8" ismergecell="N" isedit="N" datarule
="" mergetype="None" mergeid=""
D5FE4E63ACA74A0CAED77C3F9A3D6148" rowmergenum="0"
columnmergenum="0" personcode="" time="" />
```

```
<column id="F30AB1A5097840F09F7D2EB3455D67D5" tem-
plateid="e4ba0ff7ad8a41bbb1bfb4ff1ff0456b" templateversion="1"
tagname="车种" startrowindex="1" startcolumnindex="9" endrowin-
dex="2" endcolumnindex="9" ismergecell="Y" isedit="N" datarule
="" mergetype="Row" mergeid=""
F30AB1A5097840F09F7D2EB3455D67D5" rowmergenum="1" colum-
nmergenum="0" personcode="" time="" />
```

```
<column id="1A4381A4B64D4CB6965B25446942DAE5" tem-
plateid="e4ba0ff7ad8a41bbb1bfb4ff1ff0456b" templateversion="1"
tagname="时速 250 公里速度级 & # xA;动车组(CRH2A 头型) & #
xA; & # xA;" startrowindex="2" startcolumnindex="3" endrowindex
="2" endcolumnindex="4" ismergecell="Y" isedit="N" datarule=""
mergetype="Column" mergeid=""
1A4381A4B64D4CB6965B25446942DAE5" rowmergenum="0" colum-
nmergenum="1" personcode="" time="" />
```

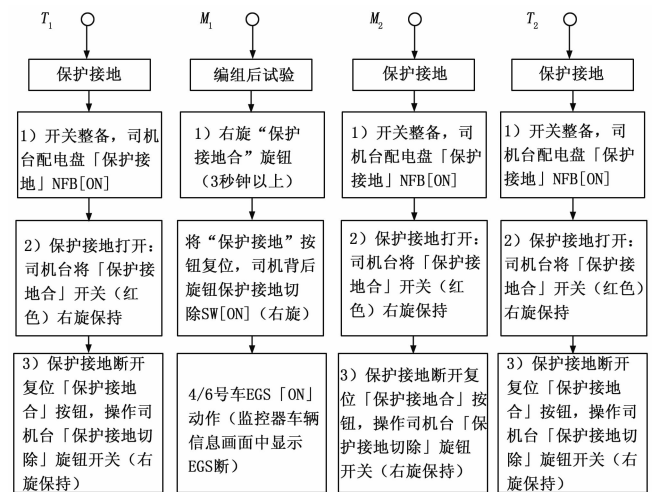


图 4 调试执行

### 4.3 数据采集与分析

系统应结合试验设备实际情况和调试现场实际需求,制订适合标准统一的数据通信协议和调试终端数据解析协议,标准的数据通信协议原则上应分为两级,第一级是针对数据帧的组织方式制定的数据协议,主要包括数据组帧算法、数据组织方式等,保证每台试验设备发送的数据都可以按照统一的方式进行提取。第二级是结合业务需求,将解析后的数据进行进一步处理,采用不同的帧头来进行标识,每一类帧都具有其对应的解析方式,通过对数据帧的解包,获取其发送数据的真实意

义。具体过程如下:需要选择某一工艺流程,按照操作提示完成相应步骤,一个工艺流程结束后,可将试验数据通过无线网发送给手持终端和综合调试系统,通过解析后的车上状态信息与故障信息,会根据调试要领书中的项目信息、操作步骤信息、确认信息进行特征值的匹配,然后在进行调试结果的验证,每一个调试结果(正则表达式)都会对应唯一的车上数据,在车上状态数据完全与特征值进行校验后,车上状态数据与调试结果数据进行自动匹配与结果的录入<sup>[10]</sup>。接着开始下一个工艺流程,直到试验结束。完成调试任务,这种自动化的调试方式大大提高了调试效率与质量。如下所示为数据处理代码:

```
public void GetUSBDateShow() {
    string port_address, port_address1 = string.Empty;
    Byte[] data = new Byte[10] { 0x55, 0xAA, 0xDD, 0x13,
    0x41, 0x48, 0x2C, 0x3C, 0x5F, 0x18 };
    string text = "0x198@21 2b 0 2 0 0 0 0 0 0 0 0 0 0 f1 78 0 10 84
ba 0 70 0 7d 59 58 0 0 0 0 32 32";//USBDateHelp.GetUSBData();
    string text1 = "0x598@aa 5b 1 0 1 0 4 0 0 0 0 0 0 0 71 78 0 0 79
b6 4b 75 0 98 54 54 0 0 0 0 0 0";
    if (text == "") {
        Page.RegisterStartupScript("key", "<script>alert('系统出
错,无法连接!!!')</script>");
        return;
    }
    else {
        port_address = text.Substring(0, text.IndexOf("@"));
        string textContent = text.Substring(text.IndexOf("@") + 1,
text.Length - text.IndexOf("@") - 1);
        string textContent1 = text1.Substring(text1.IndexOf("@")
+ 1, text1.Length - text1.IndexOf("@") - 1);
        port_address1 = text1.Substring(0, text1.IndexOf("@"));
        if (port_address == "0x198")
        {
            List<USBDateHelp.Code> list198 = new List<USBDate-
Help.Code>();
            string[] textarry198 = textContent.Split(';');
            int i198 = 0;
            while (i198 < textarry198.Length)
            {
                if (textarry198[i198] != "" && textarry198[i198] !=
null)
                {
                    USBDateHelp.Code codemodel = new USBDateHelp.Code
();
                    codemodel.ID = textarry198[i198];
                    codemodel.Value = textarry198[i198 + 1];
                    list198.Add(codemodel);
                }
                i198 = i198 + 2;
            }
            TC08H2 = TC01H1 = USBDateHelp.FormatInToDecimalist
(list198[0].ID + list198[0].Value);//充电机生命信号
            TC08Y2 = TC01Y1 = USBDateHelp.FormatInToDecimalist
(list198[9].ID + list198[9].Value) + "V";//充电机输入电压
            TC08L2 = TC01L1 = USBDateHelp.FormatInToDecimalist
(list198[12].ID + list198[12].Value) + "A";//充电机输出电流
```

```
string hexnum1198 = USBDateHelp.FormatInToDecimalist
(list198[12].Value);
if (int.Parse(hexnum1198) > 250)
{
    TC08W2 = TC01W1 = "温度" + USBDateHelp.Fault();//蓄电
池温度 1
    Image4.ImageUrl = Image1.ImageUrl = "../images/图标/灯.
gif";
}
else
    TC08W2 = TC01W1 = hexnum1198;//蓄电池温度 1
string hexnum2198 = USBDateHelp.FormatInToDecimalist
(list198[12].ID);
if (int.Parse(hexnum2198) > 250)
{
    TC08D2 = TC01D1 = "温度" + USBDateHelp.Fault();//蓄电
池温度 2
    Image4.ImageUrl = Image1.ImageUrl = "../images/图标/灯.
gif";
}
else
    TC08D2 = TC01D1 = hexnum2198;//蓄电池温度 2
    TC08Z2 = TC01Z1 = USBDateHelp.FormatInToDecimalist
(list198[10].ID + list198[10].Value) + "V";//充电机中间电压
    TC08S2 = TC01S1 = USBDateHelp.FormatInToDecimalist
(list198[11].ID + list198[11].Value) + "V";//充电机输出电压
    TC08X2 = TC01X1 = "";//蓄电池充电电流
    TC08C2 = TC01C1 = "";//锂电池组电量
}
if (port_address1 == "0x598") {
    List<USBDateHelp.Code> list598 = new List<USBDateHelp.
Code>();
    string[] textarry598 = textContent1.Split(';');
    int i598 = 0;
    while (i598 < textarry598.Length)
    {
        if (textarry598[i598] != "" && textarry598[i598] != null)
        {
            USBDateHelp.Code codemodel = new USBDateHelp.Code();
            codemodel.ID = textarry598[i598];
            codemodel.Value = textarry598[i598 + 1];
            list598.Add(codemodel);
        }
        i598 = i598 + 2;
    }
    TC08H1 = TC01H2 = USBDateHelp.FormatInToDecimalist
(list598[0].ID + list598[0].Value);//充电机生命信号
    TC08Y1 = TC01Y2 = USBDateHelp.FormatInToDecimalist
(list598[9].ID + list598[9].Value) + "V";//充电机输入电压
    TC08L1 = TC01L2 = USBDateHelp.FormatInToDecimalist
(list598[12].ID + list598[12].Value) + "A";//充电机输出电流
    string hexnum1598 = USBDateHelp.FormatInToDecimalist
(list598[12].Value);
    if (int.Parse(hexnum1598) > 250)
    {
```

(下转第 177 页)

