

重庆邮电大学

---

硕士学位论文

---

基于ISO15765的车载CAN网络上位机诊断软件设计

---

姓名：姚燕

---

申请学位级别：硕士

---

专业：控制理论与控制工程

---

指导教师：李锐；程安宇

---

20110610

## 摘要

随着车载 CAN(Controller Area Network)网络技术的发展与广泛应用，车载 CAN 网络国际诊断标准 ISO15765 随之发布。但由于协议的新颖性与国外对此相关知识产权的保护，使得国内在此方面的技术研究相对薄弱，诊断工具主要是依靠国外高成本的设备。因此，研究设计基于国际诊断标准的具有自主知识产权的低成本 CAN 网络诊断工具具有重要意义。论文以某车型的网络结构与支持诊断的网络节点为被诊断对象，在对 ISO15765 体系结构进行深入研究的基础上，通过 VC 软件平台与 USBCANII 硬件采集卡设计低成本的车载 CAN 网络诊断上位机。论文的具体工作如下：

(1) 在研究车载 CAN 网络诊断协议体系结构的基础上，针对某车型的车载网络结构，提出了网络诊断的结构及诊断的功能需求，将诊断上位机分为相对独立的常规诊断模块(无下载功能)和下载诊断模块进行设计。

(2) 在深入理解诊断协议的基础上，实现了应用层 5 类诊断服务的设计。针对网络层的数据传输，设计符合 ISO15765-2 的数据封装算法，实现网络层的数据传输。

(3) 在实现应用层协议与网络层的数据传输的基础上，针对安全访问服务的实现过程，设计安全访问算法并建立安全算法动态链接库，利用 VC 软件平台与 USBCANII 硬件采集卡，设计车载 CAN 网络常规诊断模块的低成本诊断上位机。

(4) 在深入研究基于 ISO15765-3 的下载诊断流程的基础上，设计 S19 文件处理算法对 S19 文件进行处理，实现基于 CAN 网络诊断协议的在线下载流程。以 USBCSNII 为硬件下载器，利用 VC 软件平台设计在线下载上位机软件，来实现程序基于车载 CAN 网络的在线下载。

(5) 搭建测试平台，结合 CANoe 对所开发的诊断上位机进行测试，包括常规诊断上位机的测试以及下载功能的测试。

通过实际测试，验证所设计的低成本诊断上位机可实现与车载 CAN 网络进行基于 ISO15765 的诊断通信；并可实现程序通过 CAN 网络的在线下载；从而验证了所设计诊断上位机的可行性。

**关键词：**车载网络，CAN，ISO15765，诊断，测试

## Abstract

With the development and wide use of in-Vehicle CAN (Controller Area Network) network technology, complete in-vehicle CAN network international diagnostic specification has been published. However, because of the novelty of the specification and the protection of intellectual property rights of foreign country, our country is relatively weak on this technique, moreover, diagnostic tool is mainly dependent on high-cost foreign equipment. Therefore, the study on CAN network diagnostic tool based on ISO15765 with independent intellectual property rights is significant. In this paper, diagnostic structure of certain car and network nodes which can be diagnostic is chosen as the diagnostic object. After the analysis and deep study of ISO15765 architecture, low cost diagnostic upper computer of CAN network is designed by VC software and USBCANII hardware acquisition card. The main contributions of dissertation include the following:

(1)Basing on the research on diagnostic protocol architecture of in-vehicle CAN network, the structure of in-vehicle network and the diagnostic function needs are put forward for a certain vehicle model. The diagnostic upper computer design is divided into relatively independent of ordinary diagnostic module (without download function) and download diagnostic module.

(2)The major 5 kinds of diagnostic services of application layer are realized based on deep understanding of diagnostic protocol. For the network layer data transmission, data packing algorithm which meet the ISO 15765-2 is designed to achieve the network layer data package transmission.

(3)Basing on the realization of the application-layer protocol and the network layer data transmission algorithm. For the safety access realization, safety algorithm is designed and DLL is created. Low-cost CAN network diagnostic upper computer of ordinary diagnosis module is designed by VC software platform and USBCANII hardware acquisition card.

(4)The S19 file processing algorithm is designed to realize program on-line download through CAN network diagnostic specification, which is

based on ISO15765-3 download process. Upper computer of on-line download is designed by VC software platform and USBCANII hardware, by which program can be downloaded on-line through CAN network.

(5) Test platform is built to test the diagnostic upper computer combined with CANoe. The test includes ordinary diagnostic module test and download function test.

Test results show that diagnostic upper computer can communicate with CAN network through ISO15765, and S19 file can be downloaded on-line through CAN successfully. The feasibility of the designed diagnostic upper computer is validated.

**Keywords:** in-vehicle network, CAN, ISO15765, diagnosis, test

## 第一章 绪论

### 1.1 课题研究的背景与意义

CAN 网络技术起源于欧洲，最早运用于汽车的电子通讯系统上，专门装备高档车型，被公认为现代 B 级、C 级轿车和高技术含量代表的标志之一<sup>[1]</sup>。它具有极强的抗干扰和纠错能力，这项技术的最大优点是减少了线束的数量和控制器接口的引脚数，能够使多个控制器和各类型的传感器之间的数据通信联系起来，使整车线束布置更加紧凑。CAN 网络由于其非破坏性的网络仲裁机制、较高的通信速率和灵活可靠的通信方式<sup>[2]</sup>，在车载网络领域广受青睐，CAN 网络可以更简单、迅速地实现汽车控制、通信、在线诊断以及在线编程。由于其综合的优势，目前，在多种车用总线中，CAN 总线成为应用最广泛的总线。

由于车载 CAN 总线的广泛应用，车载 CAN 网络的诊断技术亦显得尤为重要。由于诊断系统独立于车载 CAN 网络，这使得系统的开发成本增加，内部网络变得复杂。为解决上述问题，欧洲汽车厂商推出一种基于 CAN 总线的诊断系统通信标准 ISO15765，它可满足 E-OBD(European-On Board Diagnosis) 的系统要求，ISO15765 以 ISO14229-1 定义的服务为基础，规定了基于 CAN 总线的诊断服务(UDS on CAN)，包括网络管理、网络定时、应用层定时等详细内容，使得该协议的适用性和可操作性更强，并与 ISO14230 应用层的服务和参数完全兼容<sup>[3]</sup>。基于 ISO15765 的车载网络诊断不仅可以读取故障码，还能实现车载网络 ECU (Electronic Control Unit) 的在线升级功能。通过在线升级功能，整车厂可以在 4S 店刷新控制器的软件，从而减少因软件缺陷造成的召回成本<sup>[4]</sup>。此外，还能实现读取、写入及控制 ECU 的信息。如读取 ECU 的版本信息、生产厂商以及控制 ECU 的输入输出信息等功能。ISO15765 符合现代汽车网络总线系统的发展趋势，已被许多汽车厂商采纳，并将成为未来汽车行业的通用诊断标准。

对于车载网络的诊断，国外各大著名汽车公司对诊断技术的研究较为成熟，其诊断通信协议也是由国外各大厂商制定，其相关的诊断产品和诊断工具也开发的较为成熟，比如德国 Vector 公司的诊断系列产品，以及 Mentor 公司的诊断系列产品。目前国内的汽车生产厂商及汽车零部件厂商与研发机构大都直接使用国外相对成熟的诊断系列产品。由于知识产权的保护，国外对车载网络的诊断技术几乎处于封锁状态，即使是中外合资的相关汽车厂商或零部件厂商其诊断技术也是不会对国内开放。综上原因，使得目前国内的车载网络诊断技术处于起步阶段。因

此，深入研究车载网络诊断技术，是摆在国内汽车设计公司和汽车生产厂商面前必须解决的实际问题。

## 1.2 车载网络诊断协议概述

对车载网络诊断技术的实现主要依托于对车载网络诊断协议的应用与实现。现对常用的车载网络的国际诊断协议进行概述如下：

### **ISO-9141:**

早在 1996 年国际标准化组织颁布基于 K 线的 ISO-9141 标准；该标准的应用时间不算太长。

### **ISO 14230:**

ISO 14230 于 1999 年出台，又称作 Keyword Protocol 2000(kwp2000)，该诊断标准是基于 K 线的，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，目前多用单线通信。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。过去比较常用，到目前为止，ISO14230 仍是许多汽车厂商采用的诊断通信标准<sup>[5]</sup>。

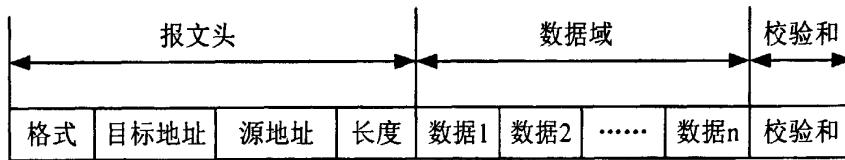


图 1.1 ISO 14230 K 线数据格式

### **ISO/DIS 15765:**

1999 年出台 ISO/DIS 15765(Diagnostics on CAN - based on KWP-2000)，此诊断标准是基于 ISO 14230 在 CAN 线上的扩充，源于 K 线的诊断标准。

### **ISO 15031:**

2001 年 6 月发布 ISO 15031(Communication for emissions-related diagnostics)，此诊断标准的出台主要针对排放系统相关的诊断，其中 ISO 15031-6 中，对故障诊断码的格式进行了详细规定。

### **ISO 15765:**

2001 年发布了 ISO 15765 (Diagnostics on CAN-based on UDS)，此诊断标准与基于 K 线的诊断标准不同，这是基于统一诊断服务的诊断。其中的 ISO15765-3、ISO15765-2 分别规定了应用层与网络层的实现，在 ISO15765-2 网络层中对 ECU

的在线上传下载进行了详细的规定。将帧类型分为单帧、第一帧、流控帧和后续帧，此诊断标准对报文进行打包传输，数据以流控帧的机制进行传输，一次最多可传输 4095 字节的数据。标准 CAN 帧格式<sup>[6]</sup>如图 1.2 所示。

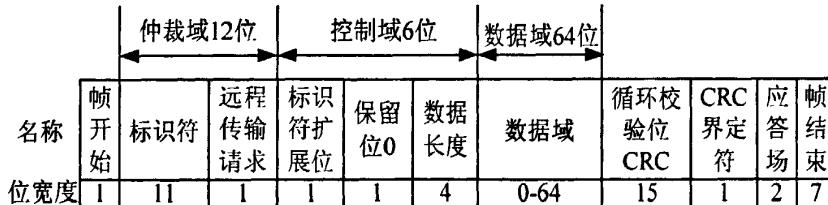


图 1.2 标准 CAN 帧格式

### ISO 14229-1:

2006 年发布统一诊断服务标准 ISO 14229-1 (Unified Diagnostic Services)。该国际标准只是规定应用层上诊断规范，该标准的制定是为了针对任何一种连续数据链路，不涉及网络及实现未。定义诊断系统的通用需求，为了实现这一点，该标准基于 OSI 基本参考模型，如表 1.1 所示，其通信系统为 7 层。其中映射到 CAN 线上的诊断为 ISO15765。在其它链路上的映射，如无线局域网、Flexray 等的诊断，也将按照 ISO14229-1(统一诊断标准 UDS) 执行<sup>[7]</sup>。

表 1.1 诊断规范与 OSI 的对应关系表

适用性	OSI 7 层	增强诊断服务（非放射相关）	
遵循	应用层（第 7 层）	ISO 14229-1 / ISO 15765-3 / ISO 11992-4	ISO 14229-1/更多标准
ISO/IEC	表达层（第 6 层）	--	--
7498 和	会话层（第 5 层）	ISO 15765-3 / ISO 11992-4	更多标准
ISO/IEC	传输层（第 4 层）	ISO 15765-2 / ISO 11992-4	更多标准
10731 的 7 层	网络层（第 3 层）	ISO 15765-2 / ISO 11992-4	更多标准
	数据链路层（第 2 层）	ISO 11898 / ISO 11992-1 / SAE J1939-15	更多标准
	物理层（第 1 层）	ISO 11898 / ISO 11992-1 / SAE J1939-15	更多标准

综述上述几种车载网络诊断协议，国内外汽车厂商使用较广泛的诊断协议为基于 K 线的 ISO 14230 和基于 CAN 线的 ISO15765 诊断协议，然而随着汽车中电控元数量的增加，如 ABS 系统、发送机电控系统、车身控制模块（BCM）系统、安全气囊等系统<sup>[8]</sup>，这些系统之间以及系统与汽车仪表之间都需要进行数据交换，若这么多的数据量仍采用导线对数据进行点对点传输，则大量导线的使用会导致系统内部繁杂，且增加成本。此外，由于 K 线不能满足数据链路层在网络管理以

及通信速率上的局限性，使得 K 线无法满足越来越复杂的车载诊断网络的需求<sup>[9]</sup>。随着 CAN 网络在汽车网络上的广泛应用，基于 CAN 网络的 ISO15765 的诊断则受到广泛应用，它符合现代汽车网络总线的发展趋势，逐渐被越来越多的汽车厂商使用，将成为未来汽车行业的通用诊断标准。

与基于 K 线的 ISO14230 相比，基于 CAN 线的诊断协议 ISO15765 具有以下的优势：

(1) CAN 总线的传输速率比 K 线要高很多，K 线的传输速率为 10.4kbit/s，而 CAN 线的最高波特率可达 1Mbit/s。

(2) K 线使用的是单线传输，CAN 线为双线采用差分信号传输，抗干扰能力强，且可靠性比 K 线好。

(3) CAN 总线可以构建比较复杂的网络结构，对于不同网络仍可通过网关实现诊断，即可实现不同网段的远程诊断，并且 CAN 网络的网络管理能力很强。

(4) 当诊断设备采用功能寻址对多个 ECU 进行诊断或诊断控制通信时，开发者不需考虑由于同时访问总线引起的总线冲突问题，因为 CAN 总线采用仲裁机制确保总线通信的正常进行。

(5) 利用 ISO15765 基于 CAN 线的诊断，其网络层对报文的传输进行了规范化的顺序控制以及流控制等，提高了报文传输的可靠性，并且其单次传输的报文数量可达 4096 字节。而 K 线在网络层没有定义，单次最多传输 255 字节的数据。

(6) 在 ISO15765-2 中，明确规定了基于 CAN 网络的 ECU 在线上传下载的诊断通信流程，将上传下载纳入诊断范围内，上传下载的内容按照 ISO15765-2 网络层传输规则进行传输，以实现更标准、更可靠的网络数据传输。

### 1.3 国内车载 CAN 网络诊断存在的问题

根据国内外车载网络诊断的研究现状与目前流行的车载网络诊断协议基础，现从以下三个方面阐述目前国内车载 CAN 网络诊断研究中存在的、并待解决的问题：

#### (一) 我国车载网络诊断技术自主知识产权方面

由于基于 CAN 网络技术的新颖性与诊断协议本身的复杂性，现阶段国内在基于 ISO15765 的车载网络诊断技术的开发与应用方面尚不成熟，在诊断开发与设计方面通常借助于国外的工具或产品。目前，国内汽车制造商与设备供应商大都使用德国 VECTO R 公司的 Candelastudio、Diva、CANdesc 等诊断系列工具，由于这些诊断工具受知识产权的保护，成本较高，其整套设备需花费几十万元。因此，有必要设计开发具有自主知识产权的基于 ISO15765 的车载网络诊断，突破国外在

汽车电子行业的技术垄断，开发具有自主知识产权的诊断产品和工具。

### （二）中小型汽车零部件厂商开发汽车诊断产品的成本与效率和质量方面

对某些中小型汽车零部件厂商而言，在开发具有诊断功能的车载 ECU 阶段后期，需对其进行测试，看所开发的 ECU 是否符合诊断协议，他们常用的测试方式是使用低成本的 CAN 网络测试工具，如周立功的 CANTEST 软件，逐个手动输入所需要的基于诊断协议的诊断通信报文，来测试 ECU 的响应情况。另外，传统的简易 CAN 网络监测工具，只是对 CAN 报文进行显示，而无法对诊断协议内容进行解析，面对大量的通信报文通过人工肉眼进行逐字节解析以判断是否符合诊断协议，此种形式的开发工作显然不能保障开发的效率与质量。对中小型企业来说，若他们耗费几十万元购买国外诊断工具则成本太高。因此，对国内中小型汽车零部件厂商而言，研究低成本的基于 ISO15765 国际诊断标准的诊断上位机工具，则可大大降低其产品开发成本，并确保产品开发的效率与质量。

### （三）基于 CAN 线的车载 ECU 在线下载的合理性与需求方面

以往 ECU 在线升级大都是通过串口或 BDM 口进行程序下载，但针对已装入汽车内通过 CAN 网络连接的 ECU，无疑使用 CAN 总线对程序进行在线下载，会比较方便也是需求所在。在 ISO15765-3 中，对基于 CAN 总线的在线下载进行了统一、详细的实现约束，使车载网络 ECU 在线下载从整车网络规划与网络管理的角度进行统一规划<sup>[10]</sup>。但由于 ISO15765 在线下载技术的新颖性与技术本身的复杂性，目前在国内核心的学术研究期刊上尚未看到相关研究领域的报道，国内汽车厂商也尚未普遍使用此种方式对 ECU 进行程序的在线下载。因此，其下载工具未产业化，实际应用中符合要求的下载工具不多，或被作为大型工具的组件使用如(CANoe)，但其成本较高并且灵活性较差。将程序下载通过 CAN 网络进行网络统一规划与管理，并使 ECU 通过 CAN 网络传输是在线下载的趋势所在，因此有必要研究基于 ISO15765 的 CAN 网络诊断的车载 ECU 程序在线下载技术。

## 1.4 论文主要研究工作

根据上节所述的在车载 CAN 网络中诊断研究中所提出的问题，本文针对以上问题展开了相关的研究工作以及相关问题的解决。主要通过研究车载 CAN 网络国际标准 ISO15765，对基于 CAN 网络的车载诊断进行研究、分析，设计基于 ISO15765 的车载 CAN 网络的上位机诊断。论文的主要研究工作的相互关系见图 1.3，具体工作如下：

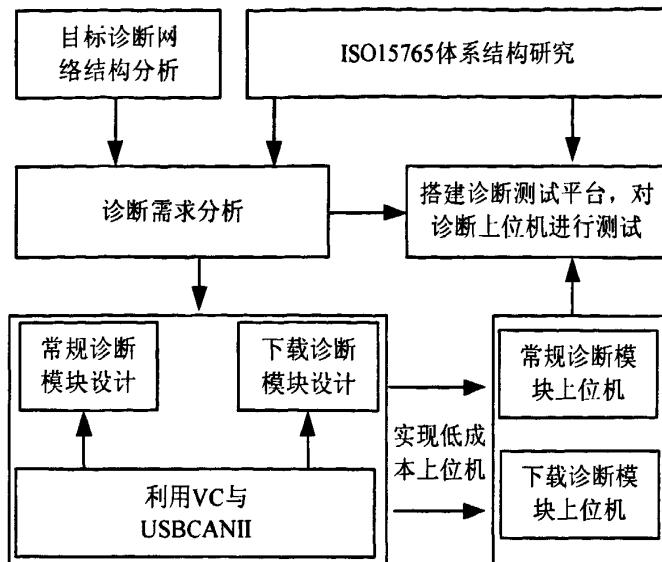


图 1.3 论文主要研究工作的相互关系

1. 通过深入研究基于车载 CAN 网络的国际诊断标准 ISO15765 的应用层与网络层协议，针对特定车载网络对象，设计车载网络诊断结构，并进行诊断需求分析。
2. 实现基于 ISO15765 的诊断上位机的应用层与网络层协议。
3. 设计基于 ISO15765 的车载 CAN 网络诊断上位机。针对安全访问服务的实现过程，设计安全访问算法并建立安全算法动态链接库，利用 VC 软件平台与 USBCANII 硬件采集卡，设计车载 CAN 网络常规诊断模块的诊断上位机。
4. 设计基于 ISO15765 的在线下载上位机。对下载诊断模块分别设计其预编程阶段与编程阶段，以及上位机对 S19 文件的解析处理。利用 VC 软件平台及 USBCANII 硬件采集卡实现下载器上位机的设计。
5. 根据诊断结构设计诊断测试平台，结合 CANoe 与 ISO15765 诊断协议对所设计的诊断上位机进行测试。

## 1.5 本章小结

本章首先介绍了课题研究背景与意义，然后概述了车载网络诊断协议，重点分析比较了 ISO14230 与 ISO15765 协议，并分析基于 CAN 网络的 ISO15765 协议的优势。在此基础上分析并概述了国内车载 CAN 网络诊断存在并待解决的问题。最后提出了本文的研究工作。

## 第二章 车载 CAN 网络诊断协议与诊断需求分析

本章首先对车载 CAN 网络国际诊断标准协议 ISO15765 体系统结构进行分析，并对其应用层与网络层进行详细解析。在此基础上，结合某车型网络结构及诊断功能进行需求分析，提出车载网络的诊断结构，以及基于 ISO15765 的车载 CAN 网络诊断功能需求。后续研究工作在此章的理论基础与需求分析上进行设计。

### 2.1 ISO15765 协议体系结构

车载 CAN 网络诊断协议体系结构如图 2.1 所示，分为应用层、网络层、数据链路层和物理层<sup>[1]</sup>。诊断首先由符合 ISO14229-1 或 ISO15765-3 的应用层开始，经过网络层 ISO15765-2 实现数据的传输、打包、解包等，再由网络层映射到数据链路层 ISO11898-1，转化为有效的 CAN 数据帧，最后到物理层的传输介质。其中 ISO 15765-4 规定了与排放相关的诊断内容。以下两节对应用层和网络层的协议进行详细分析。

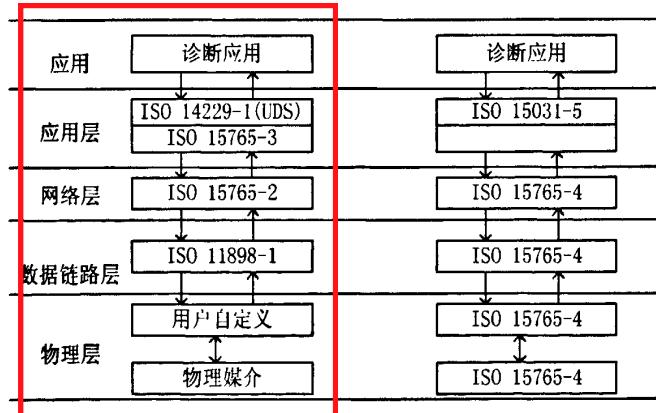


图 2.1 CAN 网络诊断协议体系结构

#### 2.1.1 应用层协议分析

如图 2.1 诊断协议体系结构所示，应用层是有关诊断的应用，ISO14229-1 是对任何一种数据链路而言，为了建立一个通用的诊断需求而建立的标准，ISO15765 是 CAN 网络在 ISO14229-1 上的具体实现。在 ISO14229-1 中规定了 25 种诊断服务，其中每种又定义了功能不同的子服务来实现具体不同服务内容。

应用层协议数据单元 (A\_PDU) 具有如图 2.2 的格式。

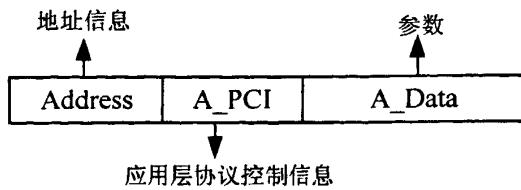


图 2.2 应用层协议数据单元格式

应用层协议数据单元包含地址信息，应用层的协议控制信息和应用层的数据参数。其中地址信息又详细分为源地址(SA)、目标地址(TA)和远程地址(RA)等，在 29 位的 CAN 报文 ID 中，以上地址有明确数值，在 11 位的 CAN 报文 ID 一般没有明确数值。对应用层协议数据单元中每项的内容详细描述如下<sup>[12]</sup>：

```

A_PDU (
    SA,
    TA,
    TA_type,
    [RA,]
    A_Data = A_PCI + parameter 1, ...
)

```

①SA：表示源地址，参数 SA 用来对客户端和服务器标识符编码，并且用来表示客户端或服务器的物理位置/实际位置。

②TA：表示目标地址，参数 TA 用来对客户端和服务器标识符编码。有两种不同的寻址方式：物理寻址和功能寻址。

③RA：表示远程地址，只有采用远程诊断的时候才使用此地址，RA 用来扩展可用地址范围。在诊断范围里，RA 仅用来执行本地服务器和远程服务器概念。远程地址标识其本身地址范围，并且独立于主网络上的地址。

④A\_Data：一串数据字节，用来定义每个独立的应用层服务。A\_Data 字符串起始于 A\_PCI，后面是具体的子服务或参数。

⑤A\_PCI：表示为应用层协议控制信息，可表示为：A\_PCI (SI)，SI “是参数服务标识符”，ISO-14229 中给出了具体服务代码，各服务代码如表 2.1 所示。

表 2.1 服务标识符代码

服务标识符 (SI)	服务类型 (bit 6)	出处
00 - 0F	OBD服务请求	ISO 15031-5
10 - 3E	ISO 14229-1 服务请求	ISO 14229-1
3F	不使用	保留

40 – 4F	OBD服务响应	ISO 15031-5
50 – 7E	ISO 14229-1主动服务响应	ISO 14229-1
7F	否响应服务标识符	ISO 14229-1
80	不使用	ISO 14229-1保留
81 – 82	不使用	ISO 14230保留
83 – 88	ISO 14229-1服务请求	ISO 14229-1
89 – 9F	服务请求	保留以便将来扩展使用
A0 – B9	服务请求	汽车制造厂商定义
BA – BE	服务请求	系统提供方定义
BF	不使用	文献保留
C0	不使用	ISO 14229-1 保留
C1 – C2	不使用	ISO 14230 保留
C3 – C8	ISO 14229-1主动服务响应	ISO 14229-1
C9 – DF	主动服务响应	保留以便将来扩展使用
E0 – F9	主动服务响应	汽车制造厂商定义
FA – FE	主动服务响应	系统提供方定义
FF	不使用	文献保留

## 2.1.2 网络层协议分析

如图 2.1 诊断协议体系结构所示，基于 CAN 的诊断体系结构的网络层实现协议为 ISO15765-2，该协议是为了建立一种基于 CAN 链路的通用诊断需求，网络层负责将数据从发送方传给接收方，保证数据传输的可靠性。对多包数据进行打包、解包，实现应用层数据与 CAN 数据帧之间的转换。发送/接收可高达 4095 字节的数据。

网络层的协议数据单元是在应用层协议数据单元的基础上又添加了网络层的协议控制信息，用于控制数据的发送方式。网络层的协议数据单元的帧格式如图 2.3 所示。

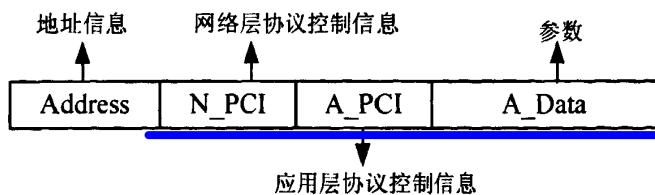


图 2.3 网络层的协议数据单元格式

## 网络层数据传输及帧格式

基于 ISO15765 的数据传输规则分为两种：单帧传输和多帧传输<sup>[13]</sup>。

### 单帧传输：

若要发送的字节不超过 6 个或 7 个字节时采用单帧传输的方式，如图 2.5 所示。

其中采用扩展寻址或混合寻址时不超过 6 个字节；采用常规寻址时不超过 7 个字节。有关寻址方式将在本节后续内容进行阐述。单帧的网络层协议控制信息如图 2.4 所示。

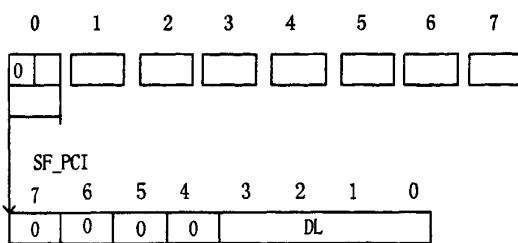


图 2.4 单帧的协议控制信息

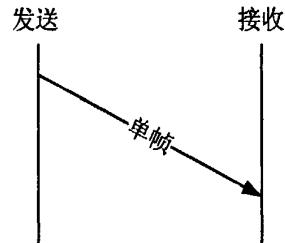


图 2.5 单帧数据传输

其中，在 N-PCI 中，前四位表示帧类型，0 表示单帧(SF)，1 表示第一帧(FF)，2 表示后续帧(CF)，3 表示流控帧(FC)。

### 多帧传输：

若传输的数据超过 6 个或 7 个字节时，则采用多帧传输的方式。如图 2.6 所示，即将要传送的数据分成多帧传送，首先发送第一帧(FF)，再通过流控制(FC)来控制后续帧(CF)的每次发送数量与时间间隔，对每个 CF 帧都有序号（0 到 F 循环编号，第一个后续帧序号除外为 1）。

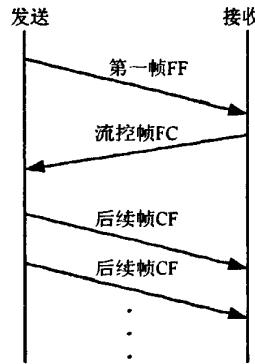


图 2.6 多帧数据传输规则

在此以传送 19 字节的数据为例详细说明多帧传输的传输规则及多帧传输时所用各种帧类型，及其协议控制信息。

首先，如图 2.7，传送第一帧 FF，第一帧的协议控制信息如图 2.8 所示，由两字节组成，第一字节的高四位用 1 表示为第一帧，后 12 个字节表示传输数据量的

大小。最多传输的数据量为 fff(4095 字节)。

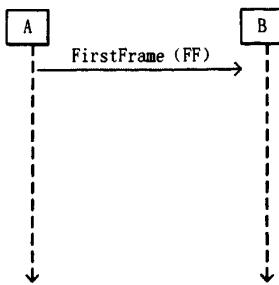


图 2.7 第一帧数据传输

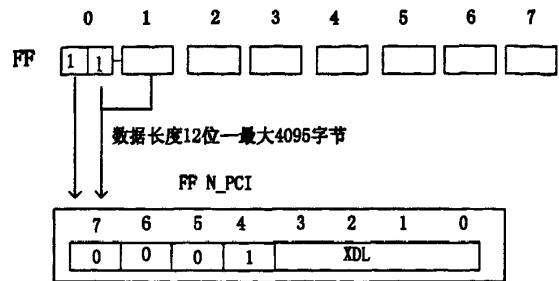


图 2.8 第一帧数据的协议控制信息

接收到第一帧后，应回应流控帧 FC 以控制数据以何种方式控制发送，如图 2.9，流控帧的协议控制信息由三字节组成，如图 2.10，第一字节高四位用 3 表示流控帧，第四位表示状态，0 表示允许继续发送，1 表示等待发送，2 表示请求发送数据量超过 ECU 的 buffer 大小。第二字节表示允许一次发送后续帧的数量。第三字节表示后续帧发送的时间间隔。若后两字节都为 0，则表示后续帧可任意每次最大量帧发送。

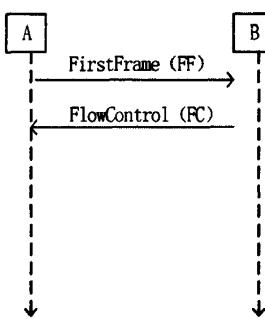


图 2.9 流控帧的传输

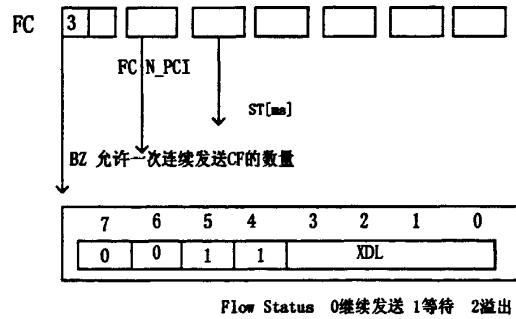


图 2.10 流控帧的协议控制信息

接收到流控帧后，后续帧会根据流控帧的信息进行发送，如图 2.11，后续诊断的协议控制信息如图 2.12 所示：协议控制信息占一字节，用高四位的 2 表示后续帧(CF)，第四位的 SN 表示后续帧的编号，从 0 开始到 F 循环编号，第一个后续帧从 1 开始编号，因为 0 被分配给第一帧 FF，但在 FF 的 N\_PCI 中并不包含一个明确的 SN，但应看做分段消息的第一个。

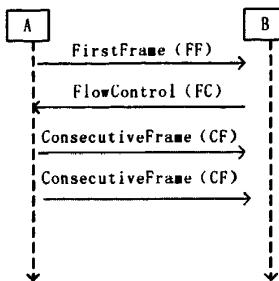


图 2.11 后续帧的传输

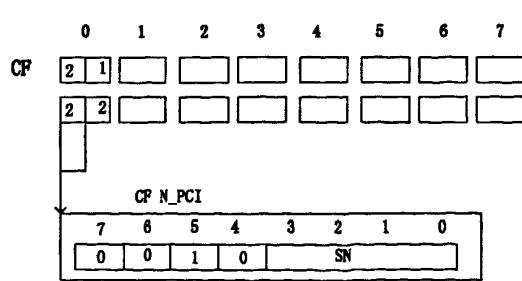


图 2.12 后续帧的协议控制信息

### 诊断帧 4 种地址格式

将网络层协议数据单元 N-PDU 映射到 CAN 数据帧的不同位置，构成 4 种地址格式：常规寻址（用于 11 位 CANID），常规固定寻址（用于 29 位 CANID），扩展寻址（11 位 CANID），混合寻址（11 或 29 位 CANID）<sup>[14]</sup>。其中，混合寻址仅用于远程寻址方式（被诊断节点处于不同的网段）。以下对四种地址格式进行分析：

#### 常规寻址

常规寻址（Normal addressing），仅用于 11 位 CAN ID，网络层地址信息 N\_AI 映射到 CANID，但没有规定 N\_AI 与 CAN ID 的具体映射关系。

#### 常规固定寻址

常规固定寻址（Normal fixed addressing），仅用于 29 位 CANID，完整定义了 N\_AI 如何映射到 29 位 CANID。此种寻址方式，明确规定了地址信息与如何映射到 CANID。其中，在此寻址方式下，数据域的第一字节为远程地址。在 29 位 ID 中，前三位用 110 表示为诊断帧，25 和 24 位按 J1939 规定都为 0，16 到 23 位表示在此种寻址方式下是物理寻址还是功能寻址，都对应不同的值，后 16 位分别表示目标地址与源地址。

#### 扩展寻址

扩展寻址（Extended addressing），仅用于 11 位 ID，其第一数据字节需放置目标地址信息，第二数据字节为网络层的协议控制信息 N\_PCI。

#### 混合寻址

混合寻址（Mixed addressing）有 11 位 ID 与 29 位 ID，用于远程寻址，即被诊断网络处于不同网段，此种寻址方式下的第一数据字节为扩展地址信息。表 2.2 为 29 位 ID 的混合寻址方式。

表 2.2 29 位 ID 地址类型为物理地址的混合寻址方式

N_PDU 类型	29 位 CANID								CAN 帧数据域							
	28~26	25	24	23~16	15~8	7~0	1	2	3	4	5	6	7	8		
SF	110bin	0	0	206dec	N_TA	N_SA	N_AE	N_PCI	N_Data							
FF	110bin	0	0	206dec	N_TA	N_SA	N_AE	N_PCI	N_Data							
CF	110bin	0	0	206dec	N_TA	N_SA	N_AE	N_PCI	N_Data							
FC	110bin	0	0	206dec	N_TA	N_SA	N_AE	N_PCI	N/A							

#### 数据映射

从应用层到网络层再到数据链路层的协议数据单元映射关系如图 2.13 所示。应用层的 A\_PCI 和 A\_Data 映射到网络层的 N\_Data，网络层 N\_PCI 和 N\_Data 映射到数据链路层的 Data。即到最终 CAN 数据域的数据依次为 {N\_PCI, A\_PCI, A\_Data}，首先是网络层的协议控制信息，接下来是应用层的协议控制信息，最后

是要传送的数据。假设要发送一个读故障码服务，采用单帧传送，则最终 CAN 网络上的数据为{02, 19, 03}，02 为网络层协议控制信息 N\_PCI，表示单帧携带 2 字节数据；19 为应用层协议控制信息 A\_PCI，即服务 ID(SID)，表示读故障码服务；03 为其子服务，表示读取所有故障码。

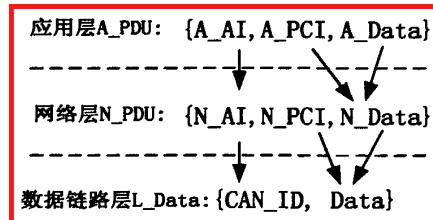


图 2.13 各层协议数据单元的映射关系

## 2.2 诊断网络结构与需求分析

对车载网络的诊断，应根据目标诊断网络要确定诊断的结构，此节中首先分析汽车诊断的三种诊断结构，然后在此基础上结合目标诊断网络设计车载网络诊断的结构，基于所设计的结构，再进行后续诊断相关功能的设计。

根据不同的车身网络层次结构和客户端（诊断设备）的接入位置，汽车网络诊断大体上可分为三种诊断结构<sup>[15]</sup>：第一种是客户端（诊断设备）与服务器（ECU）在同一个网络，诊断设备直接对 ECU 进行诊断，如图 2.14 所示。

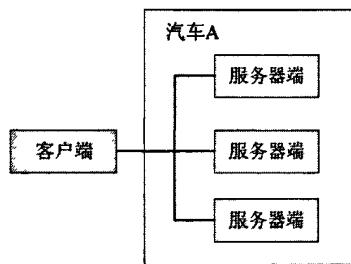


图 2.14 汽车诊断结构 1

第二种是客户端（诊断设备）通过网关与服务器（ECU）相连接，诊断设备的诊断报文通过网关转换后再发送给被诊断 ECU 进行诊断通信，如图 2.15 所示。

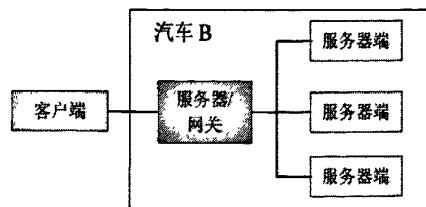


图 2.15 汽车诊断结构 2

第三种是主网络下面有子网络，客户端（诊断设备）在主网络中，而需要诊断的服务器（ECU）在子网络中，此时客户端（诊断设备）与服务器（ECU）的通信是通过兼有网关功能的服务器实现的，此种诊断主要为远程诊断方式。如图 2.16 所示：

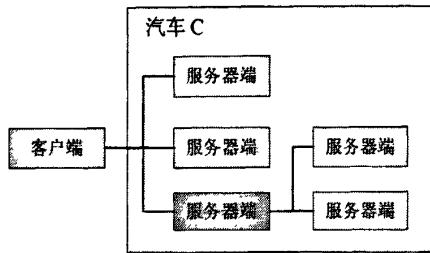


图 2.16 汽车诊断结构 3

现选取目前广泛应用的以 CAN 网络为主干网的某车型的车载网络结构为目标网络进行研究，对车载网络的诊断结构进行需求分析。被诊断的目标网络<sup>[16]</sup>如图 2.17 所示，整体网络结构以 CAN 网络作为主干网，仪表节点（IP）、ABS 节点、车身控制模块（BCM）节点等连接到主干网上，LIN 子网络节点通过兼有网关功能的 BCM 节点连接到 CAN 主干网上，则诊断接口可在主干网 CAN 线上接出，如 2.17 图中所示用虚连接 CAN 线并圈出的“Tester Interface”。

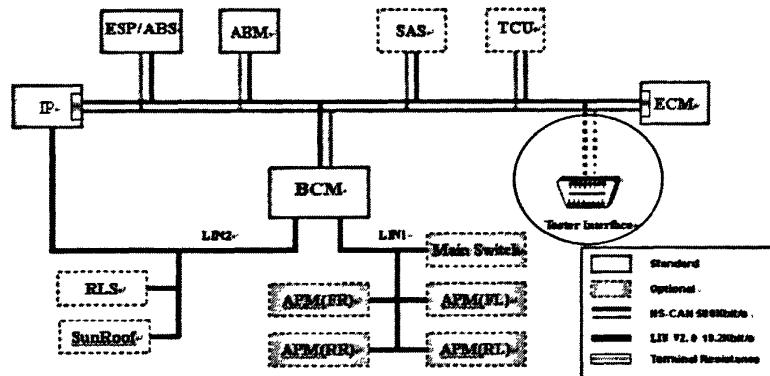


图 2.17 某车型的网络结构

根据 2.2.1 节中的三种汽车诊断结构，则目标网络的诊断结构可采用第一种汽车诊断结构，客户端（诊断设备）与服务器（ECU）在同一个网络，诊断设备直接对 ECU 进行诊断。

诊断网络结构可设计为如图 2.18 结构：诊断设备连接到 CAN 网络直接对网络节点进行诊断。对于 LIN 子网络节点的诊断，诊断设备可通过 CAN 网络的兼有网关功能的 BCM 节点对其进行间接诊断。

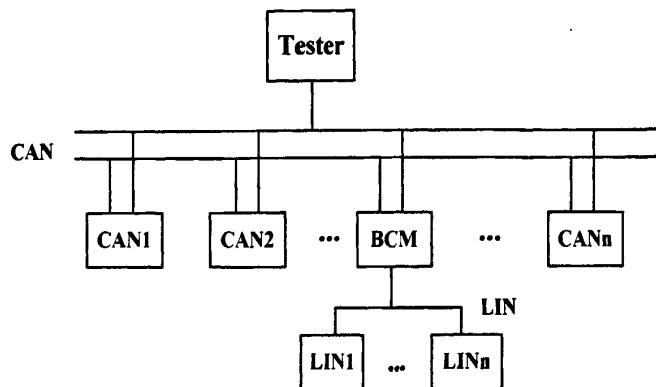


图 2.18 车载诊断系统网络结构

## 2.3 诊断服务及功能需求分析

根据 2.2 节中 CAN 网络诊断协议体系结构分析以及目标网络诊断结构的需求分析，在本节中将对整个网络的诊断服务及其实现进行需求分析与设计<sup>[17]</sup>，包括寻址类型的需求分析、诊断服务的需求分析以及诊断上位机诊断功能的需求分析

### 寻址类型需求分析：

针对 2.1.2 节中的常规寻址、常规固定寻址、扩展寻址和混合寻址四种寻址方式。现对四种寻址方式进行如下分析与选择：

首先，混合寻址一般用于远程诊断，在此不做选择。

扩展寻址的地址信息将位于 CAN 报文的数据域，此寻址方式有占用和浪费数据帧资源的缺陷，在此也不选择此种寻址方式。

对常规固定寻址而言，虽然其源地址、目标地址以及其物理和功能寻址等清晰的在 29 位 ID 中给以映射，但针对目前汽车厂商广泛使用的 11 位 ID 而言相对不统一。

综上分析，本文诊断网络的寻址方式采用 11 位 CAN 报文的 ID 的常规寻址方式。对同一个 ECU 的物理寻址与功能寻址采用不同的两对固定预先分配好的 ID。

### 诊断服务需求分析：

诊断服务应包含基于 ISO14229-1 与 ISO15765-3 应用层的诊断服务<sup>[18]</sup>，具体包括：诊断会话控制（10）、ECU 复位（11）、清除故障码（14）、读取故障码（19）、读取数据（22）、由地址读取内存（23）、读 Scaling 数据（24）、安全访问（27）、通信控制（28）、周期读数据（2A）、动态定义数据（2C）、写入数据（2E）、输入输出控制（2F）、例程控制（31）、请求下载（34）、请求上传（35）、传输数据（36）、请求传输退出（37）、写入内存（3D）、诊断仪在线（3E）、访问定时参数（83）、

安全数据传输（84）、控制 DTC 设置（85）、链路控制（87）。

此外，还应包含扩展的诊断服务实施，即用户自定义的诊断服务。

#### 诊断功能需求分析：

对于基于 ISO15765 的诊断上位机功能分为两部进行实现：常规诊断模块和下载诊断模块，这样会方便诊断服务及诊断下载的实施，使两块相对独立的功能模块执行更加方便灵活。

对常规诊断模块的功能需求分析如用例图<sup>[19]</sup>2.19 所示，应包括如下功能：

- (1) 诊断用户可对设备进行波特率、滤波、发送模式、CAN 路选择等参数的硬件配置；
- (2) 用户可进行基于 ISO15765 的诊断服务的实施，具体诊断服务的内容在上述诊断服务需求分析中已经给以详细说明；
- (3) 用户可进行非诊断报文即正常报文的发送，比如任意非诊断报文的发送，报文的循环发送以及不同帧类型的发送等；
- (4) 用户需清楚的观察到诊断通信报文数据，并需将通信报文给以解析处理，解析成清晰的诊断报文，比如通信报文为何种诊断帧类型、何种诊断服务，报文的 ID，发送接收方向等；
- (5) 用户可对通信报文进行保存及刷新当前显示报文；
- (6) 可打开关闭设备。

对下载诊断模块的功能需求分析如用例图 2.20 所示，应包括如下功能：

- (1) 用户可对下载器进行波特率、CAN 路选等的硬件配置；
- (2) 用户可执行预编程阶段的诊断服务，如通信控制、终止 DTC 设置等；
- (3) 用户可执行编程阶段的诊断服务，诊断会话控制、安全访问、读取 ECU 信息、下载数据等服务；
- (4) 用户可观察到通信报文内容、可保存及刷新报文等功能；
- (5) 打开关闭下载器设备。

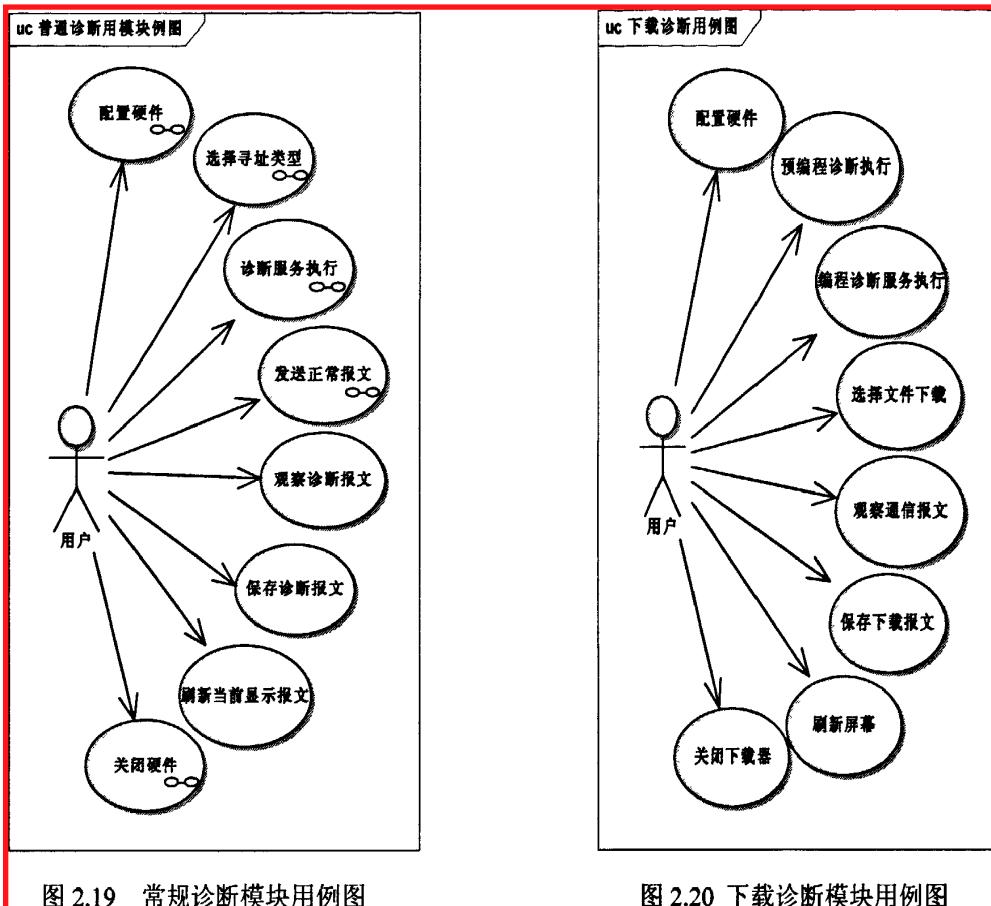


图 2.19 常规诊断模块用例图

图 2.20 下载诊断模块用例图

## 2.4 本章小结

本章首先分析了基于 ISO15765 的车载 CAN 网络诊断协议体系结构，然后对汽车诊断结构及某车型网络的网络结构进行分析，提出了车载网络的诊断结构需求。在前两者基础上对基于 ISO15765 的常规诊断模块及下载诊断模块进行了较为详细的功能需求分析。

## 第三章 基于 ISO15765 的上位机常规诊断模块设计

基于第二章提出的需求分析，在此章中对常规诊断模块的功能实现进行详细的设计。主要包括以下几个内容：整个软件系统的设计；CAN 数据采集卡的接口函数定义；ISO15765 诊断协议的逐层实现；常规诊断模块的辅助功能的设计与实现。上述几项内容都将在此章中进行详细设计与阐述。

### 3.1 诊断系统功能设计

整个常规诊断模块的功能设计如图 3.1 所示。PC 端上位机软件通过 USB 口与 CAN 数据采集卡相接，并通过接口函数实现与 CAN 卡的通信；CAN 卡通过其 CAN 口与被诊断 CAN 网络连接实现基于 ISO15765 的通信<sup>[20-22]</sup>。

对 PC 端诊断软件主要功能模块及之间的关系与参数传递说明如下：硬件配置功能模块，实现通信网络波特率、滤波等的配置；通信功能模块，通过 CAN 卡的接口函数以及硬件配置模块所传递的通信参数与诊断模块所传递的参数可实现相关通信；诊断功能模块，可实现基于 ISO15765 的诊断功能实施，主要包含应用层的诊断服务执行；报文显示功能模块，对诊断通信报文进行解析，包括诊断帧的类型、诊断服务类型、诊断帧 ID、诊断数据、诊断发送与接收的方向等信息。此外，文件存储功能模块以及屏幕刷新功能模块，可实现对诊断通信报文的分析与处理。诊断上位机与被诊断 ECU 通过诊断协议 ISO15765 进行诊断通信。

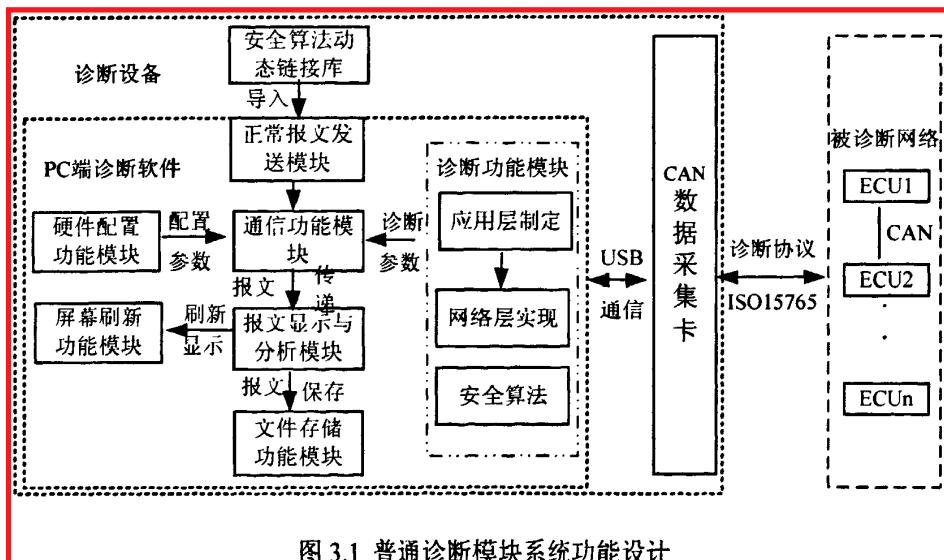


图 3.1 普通诊断模块系统功能设计

以下几节内容将对上述所设计的常规诊断系统模块的每个功能模块逐一进行实现。

### 3.2 CAN 数据采集卡及其接口函数

对系统 CAN 数据采集卡的选择，采用低成本周立功 USBCAN-II 实现 CAN 报文的发送与接收<sup>[23]</sup>，USBCAN-II 智能 CAN 接口卡可通过 USB 口实现与 PC 端的通信，此数据采集卡集成了 2 路 CAN 总线接口，其通信过程支持 CAN 协议，通过这两路或其中的一路 CAN 接口可将其作为 CAN 网络节点连接到 CAN 网络<sup>[24]</sup>。

USBCAN-II 板卡支持的动态链接库中主要用到的结构体有两个：VCI\_CAN\_OBJ 结构体和 VCI\_INIT\_CONFIG 结构体。现对主要的两个结构进行说明如下：

VCI\_CAN\_OBJ 结构体主要在发送函数和接收函数中被用来传送 CAN 信息帧。包含 CAN 报文帧的 ID、数据长度、数据、帧类型等参数。结构体定义及参数如下：

```
typedef struct _VCI_CAN_OBJ {
    UINT      ID;
    UINT     TimeStamp; //有无时间戳判断
    BYTE     TimeFlag; //时间
    BYTE     SendType; //发送类型
    BYTE     RemoteFlag; //远程帧判断
    BYTE     ExternFlag; //扩展帧判断
    BYTE     DataLen; //数据长达
    BYTE     Data[8]; //数据
    BYTE     Reserved[3];
} VCI_CAN_OBJ, *PVCI_CAN_OBJ;
```

VCI\_INIT\_CONFIG 结构体定义了初始化 CAN 的参数配置。此结构体在初始化函数中被用来初始化 CAN。主要参数有验收码、屏蔽码、用于配置波特率的两个定时器参数等。结构体定义及参数如下：

```
typedef struct _INIT_CONFIG {
    DWORD    AccCode; //验收码
    DWORD    AccMask; //屏蔽码
    DWORD    Reserved; //
    UCHAR   Filter; //滤波方式
    UCHAR   Timing0; //定时器 0
```

```

    UCHAR Timing1; //定时器 1
    UCHAR Mode; //发送模式
} VCI_INIT_CONFIG, *PVCI_INIT_CONFIG;

```

动态链接库中主要用到的接口库函数有 VCI\_StartCAN、VCI\_OpenDevice、  
VCI\_CloseDevice、VCI\_Transmit 和 VCI\_Receive<sup>[25]</sup>。函数功能描述如下：

VCI\_StartCAN, 此函数用以启动 CAN, 函数原型描述如下:

DWORD \_\_stdcall VCI\_StartCAN(DWORD DevType, DWORD DevIndex,  
DWORD CANIndex);

VCI\_OpenDevice, 此函数用以打开设备, 函数原型描述如下:

DWORD \_\_stdcall VCI\_OpenDevice(DWORD DevType, DWORD DevIndex,  
DWORD Reserved);

VCI\_CloseDevice, 此函数用来关闭硬件设备。函数原型描述如下: DWORD  
\_\_stdcall VCI\_CloseDevice(DWORD DevType, DWORD DevIndex);

VCI\_InitCan, 此函数用以初始化指定的 CAN, 函数原型描述如下:

DWORD \_\_stdcall VCI\_InitCan(DWORD DevType, DWORD DevIndex,  
DWORD CANIndex, PVCI\_INIT\_CONFIG pInitConfig);

VCI\_Transmit, 此函数用于发送报文, 返回值为实际发送的 CAN 报文帧数。  
函数原型描述如下:

ULONG \_\_stdcall VCI\_Transmit(DWORD DevType, DWORD DevIndex,  
DWORD CANIndex, PVCI\_CAN\_OBJ pSend, ULONG Len);

VCI\_Receive, 此函数从指定的设备读取数据, 函数原型描述如下:

ULONG \_\_stdcall VCI\_Receive(DWORD DevType, DWORD DevIndex,  
DWORD CANIndex, PVCI\_CAN\_OBJ pReceive, ULONG Len, INT WaitTime=-1);

VCI\_ReadErrInfo 函数来获取错误码。

接口库函数使用流程如图 3.2 所示:

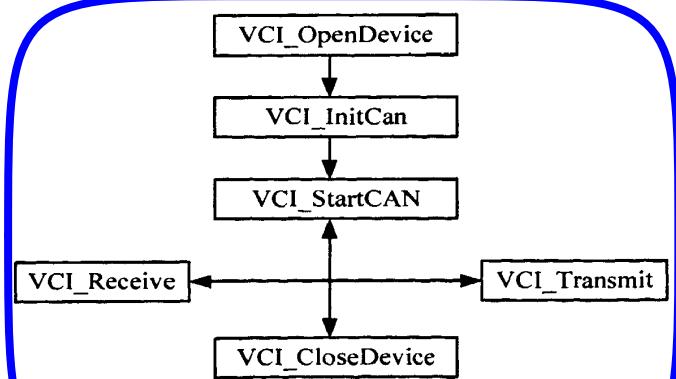


图 3.2 USBCANII 接口库函数使用流程

### 3.3 诊断功能模块设计

如 3.1 节图 3.1 所示, PC 端诊断软件功能模块中, 诊断功能模块为诊断系统主要的功能模块, 此功能模块主要实现基于 ISO15765 车载 CAN 网络国际诊断标准的诊断服务的实施, 包括 ISO14229-1 和 ISO15765-3 中应用层的实现, 以及基于 CAN 网络 ISO15765-2 网络层的实现, 还有相关的安全访问算法设计与实现。在此节中将对此功能模块所设计的功能进行具体设计。

诊断上位机的诊断功能模块设计过程中, 其核心功能为 ISO15765 诊断协议的实现, 诊断上位机软件对诊断协议的实现过程见图 3.3<sup>[26,27]</sup>。在 3.3.1 节与 3.3.2 节中将分别详细阐述应用层与网络层协议的实现。

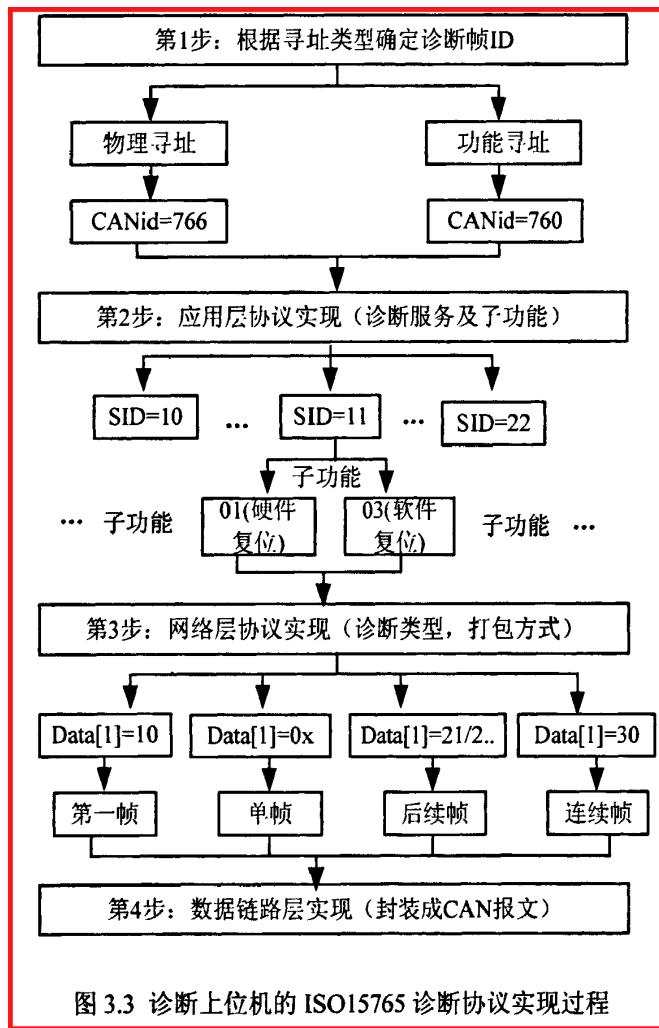


图 3.3 诊断上位机的 ISO15765 诊断协议实现过程

物理寻址: 每个ECU的物理寻址各不相同, 物理寻址的报文只有一个ECU会接收处理;

功能寻址: 同一总线上具备诊断的ECU功能寻址是相同的, 所有ECU都会接收并处理。

网络传输层: 根据应用层需要发送数据的字节数(是否<=7字节) 将数据封装成[单帧]或者[首帧+若干连续帧]的方式发送至总线。

### 3.3.1 应用层诊断服务设计

综述 ISO14229-1 应用层协议，该协议推荐规定了 5 大类诊断服务，主要有诊断/通信管理、数据传输、读故障信息、在线编程、功能/元件测试。其中，诊断/通信管理主要包括诊断会话控制、安全访问、通信控制等 10 种诊断服务；数据传输包括读写数据的 7 种诊断服务；读故障信息包括读故障信息和清除故障信息 2 种诊断服务；在线编程相关的包括请求下载、数据传输等 3 种诊断服务；功能/元件测试包括例程控制和输入输出控制 2 种诊断服。所有诊断服务在 2.3 节中的诊断服务需求分析中已详细列出，本设计将每项诊断服务以树形控件主节点的形式供用户选择，每个诊断服务请求以双击子节点的形式发送<sup>[28]</sup>，如图 3.4 和 3.5 所示。



图 3.4 诊断服务设计



图 3.5 诊断服务子服务设计

参考 2.1.1 节中应用层的帧格式，主要诊断服务的制定及其具体实现对每项诊断服务的详述如下<sup>[29]</sup>。

#### 诊断会话控制服务 (SID=0x10)

该诊断服务简称 DSC 诊断服务，SID 值为 10，客户端通过该诊断服务切换服务器的诊断会话模式。主要规定了三种会话模式：默认模式、扩展模式和编程模

式。其中，用户还可以自定义其它会话模式。

**默认模式：**ECU 上电后自动进入默认模式，在默认模式下，ECU 支持一些基础的诊断服务，而非所有的诊断服务。

**扩展模式：**扩展模式下，ECU 支持所有的诊断服务。

**编程模式：**该模式主要应用于上传下载等一些需要改变 ECU 内部数据的服务使用。

除了默认模式的所有模式统称为非默认模式，当 ECU 处于非默认模式时，如果在规定时间内（一般为 5s 钟以内）ECU 无任何诊断服务执行，则自动回到默认模式下，将不再支持非默认模式下的诊断服务，此时一般用循环发送“诊断仪在线”服务的方式以保持 ECU 处于非默认模式下。

诊断会话控制的应用层协议实现如表 3.1 所示。

表 3.1 诊断会话控制应用层数据

	数据字节	描述（所有值是 16 进制数）	字节 (Hex)	备注
A_Data	#1	请求诊断会话控制 SID	10	
	#2	诊断会话类型	01	

### 电控单元复位服务 (SID=0x11)

该诊断服务简称 ECUR 诊断服务，SID 值为 11，客户端通过该诊断服务命令服务器（电控单元）复位。协议推荐三种复位方式—硬件复位，点火钥匙复位和软件复位。

**硬件复位：**向 ECU 发送硬件复位请求时，ECU 执行硬件复位。

**软件复位：**向 ECU 发送软件复位请求时，ECU 执行软件复位。

**点火钥匙复位：**向 ECU 发送点火钥匙复位请求，ECU 响应点火钥匙复位。

若服务器可以执行复位，则应首先发送肯定响应报文，再执行复位。

电控单元复位的应用层协议实现见表 3.2，其中，01 为其子功能，表示硬件复位，其它子功能定义如表 3.3 所定义。

表 3.2 电控单元复位应用层数据

	数据字节	描述（所有值是 16 进制数）	字节 (Hex)	备注
A_Data	#1	电控单元复位 SID	11	
	#2	复位类型	01	子功能，见表 3.3

表 3.3 电控单元复位子功能数值定义

数值	描述	备注
0x00	保留	
0x01	硬件复位	
0x02	点火钥匙复位	
0x03	软件复位	
0x4~0x7F	保留	

**清除故障诊断码服务 (SID=0x14)**

该诊断服务简称 CDTCI 诊断服务，SID 值为 14，客户端通过该诊断服务清除服务器中存储的诊断信息。客户端通过发送 DTC 群组 ID 来指示服务器所要清除的是哪种类型的 DTC 信息。协议规定一次只请求清除一组 DTC 信息，有关 DTC 群组的定义见表 3.5，故请求报文只为单帧。

清除故障诊断码应用层协议实现如表 3.4 所示。

表 3.4 清除诊断信息应用层数据

	数据字节	描述 (所有值是 16 进制数)	字节 (Hex)	备注
A_Data	#1	清除诊断信息服务 SID	14	DTC 群组 ID 定义见表 3.5
	#2	DTC 群组 ID	C0	
	#3		00	
	#4		00	

表 3.5 所设计的 DTC 群组列表

DTC 群组 ID	描述	备注
000000	排放系统的 DTC 群组 ID	
000001	动力系统的 DTC 群组 ID	
400000	底盘系统的 DTC 群组 ID	
300000	车身系统的 DTC 群组 ID	
C00000	网络系统的 DTC 群组 ID	
FFFFFF	全部系统的 DTC 群组 ID	

**读取 DTC 信息服务 (SID=0x19)**

该诊断服务简称 RDTCI 诊断服务。SID 值为 19，客户端通过该诊断服务读取

车辆内部所有服务器或一组服务器存储的 DTC 信息。服务器支持客户端获得的 DTC 信息<sup>[30]</sup>。子功能描述如下：

01 子功能，获得与客户端定义的 DTC 状态屏蔽码相匹配的 DTC 数目（0x19 0x01），该子功能用于获得与客户端定义的 DTC 状态掩码相匹配的 DTC 数量。

02 子功能，获得与客户端定义的 DTC 状态屏蔽码相匹配的所有 DTC 列表（0x19 0x02），该子功能用于获得与客户端定义的 DTC 状态掩码相匹配的 DTC 列表。

**03、04 子功能，获得与客户端定义的 DTC 和状态屏蔽码相关的 DTCSnapshot 记录信息（0x19 0x03、0x19 0x04）**，该子功能用于获得 DTCSnapshot 记录，响应报文包括 DTCSnapshot 记录数量和记录信息等。

06 子功能，获得与客户端定义的 DTC 相关的扩展数据（0x19 0x06），该子功能用于获得 DTC 的一个或全部扩展数据。

**0A 子功能，获得服务器支持的所有 DTC 的状态（0x19 0x0A）**，该子功能用于获得服务器支持的所有 DTC 的状态信息。响应报文包含 DTC 有效状态掩码、DTC 和对应的状态记录。

各子功能的应用层实现如表 3.6、表 3.7、表 3.8 和表 3.9。

表 3.6 01/02 子服务应用层数据

	数据字节	描述（所有值是 16 进制数）	字节（Hex）	备注
A_Data	#1	电控单元复位 SID	19	
	#2	DTC 数量	01	
		DTC 列表	02	
	#3	DTC 状态屏蔽码	00-ff	

表 3.7 03/04 子服务应用层数据

	数据字节	描述（所有值是 16 进制数）	字节（Hex）	备注
A_Data	#1	电控单元复位 SID	19	
	#2	DTC 快照记录	03	
		DTC 快照记录	04	
	#3	DTC 状态屏蔽码高位	00-ff	
	#4	DTC 状态屏蔽码中位	00-ff	
	#5	DTC 状态屏蔽码低位	00-ff	
	#6	DTC 快照记录数量	00-ff	

表 3.8 06 子服务应用层数据

	数据字节	描述 (所有值是 16 进制数)	字节 (Hex)	备注
A_Data	#1	电控单元复位 SID	19	
	#2	DTC 扩展数据记录	06	
	#3	DTC 状态屏蔽码高位	00-ff	
	#4	DTC 状态屏蔽码中位	00-ff	
	#5	DTC 状态屏蔽码低位	00-ff	
	#6	DTC 扩展数据记录量	00-ff	

表 3.9 06 子服务应用层数据

	数据字节	描述 (所有值是 16 进制数)	字节 (Hex)	备注
A_Data	#1	电控单元复位 SID	19	
	#2	报告支持的 DTC	0A	

### 读取数据服务 (SID=0x22)

该诊断服务简称 RDBI 诊断服务。SID 值为 22，客户端通过该诊断服务可以读取服务器中指定数据标识符对应的数据，数据标识符以及对应的记录数据由用户或供应商自己定义。

数据 ID 均由两个字节组成，在请求报文中，可一次包含多个所需数据的 ID(即一次可以请求读取多项 ECU 数据)。

请求报文中可以只请求一个数据，也可以请求多组数据。当请求多组数据时，响应报文以：ID1，数据，ID2，数据……等依次类推的形式存放数据到相应的字节中，常使用多帧方式发送。

该服务的请求和响应报文均可为单帧，也可为多帧。

读取数据服务的应用层协议实现如表 3.10 所示。

表 3.10 读取数据应用层数据

	数据字节	描述 (所有值是 16 进制数)	字节 (Hex)	备注
A_Data	#2	读取数据 SID	22	
	#3	数据标识符 1 (DID)	F1	可以读取一个或多个 ID
	#4		8C	
	#5	数据标识符 2 (DID)	F1	
	#6		99	

### 安全访问服务 (SID=0x27)

该诊断服务简称 SA 诊断服务。SID 值为 27，客户端通过该诊断服务，请求或写入某些因保密、排放以及人身安全相关的受限数据或和诊断服务等。

安全访问服务分两步进行，第一步，诊断仪请求种子，ECU 将响应一个随机种子。第二步，诊断仪依据加密算法对种子进行处理，得到密钥，将密钥发送至 ECU，ECU 内部也已定义好了规定的算法，将接收到的密钥与自己算出来的密钥相比较，若一致，则 ECU 解锁成功，发送肯定响应；若不一致，则 ECU 解锁失败，发送否定响应。

加密算法与随机种子字节数由用户自定义。扩展模式与编程模式下的服务都需要进行安全访问服务。某些需要 ECU 先解锁才能执行的服务若没有先执行该服务，将发送否定响应。先判断诊断仪是否请求过种子，若诊断仪直接发送密钥，则要发送请求次序错误的否定响应报文。3 次非法密钥后，服务器需要锁定 10 秒后才允许再次进行安全访问。

安全访问服务的应用层协议实现如表 3.11、和表 3.12 所示，该服务的请求和响应报文可为单帧，也可为多帧。

表 3.11 安全访问应用层数据——请求种子

	数据字节	描述 (所有值是 16 进制数)	字节 (Hex)	备注
A_Data	#2	安全访问 SID	27	
	#3	请求种子子功能	01	安全等级一

表 3.12 安全访问应用层数据——发送密钥

	数据字节	描述 (所有值是 16 进制数)	字节 (Hex)	备注
A_Data	#2	安全访问 SID	27	
	#3	发送密钥子功能	02	安全等级一
	#4	密钥	YY	密钥所占字节数量由安全算法决定
	#5	密钥	XX	

同一安全访问过程请求种子与发送密钥的子功能类型值不同，一般以加 1 递增的方式成对使用，如 01 与 02，03 与 04，05 与 06，在次应用层定义中，定义了三种安全等级算法。

表 3.13 安全算法等级

数值	描述	备注
0x01	安全等级一	等级一请求种子
0x02	安全等级一	等级一发送密钥
0x03	安全等级二	等级二请求种子
0x04	安全等级二	等级二发送密钥
0x05	安全等级三	等级三请求种子
0x06	安全等级三	等级三发送密钥

### 通信控制服务 (SID=0x28)

该诊断服务简称 CC 诊断服务, SID 值为 28, 客户端通过该诊断服务禁止或者允许服务器非诊断报文的发送和接收。该诊断服务通常和“控制 DTC 设置”(85)服务联合使用, 实现 ECU 下载预编程阶段时的诊断通信控制, 终止正常通信以确保编程阶段程序的高速率下载。

通信控制服务的应用层协议实现如表 3.14。

表 3.14 通信控制应用层数据

	数据字节	描述 (所有值是 16 进制数)	字节 (Hex)	备注
A_Data	#2	通信控制 SID	28	
	#3	通信控制子功能—控制类型	00	子功能, 见表 3.15
	#4	通信类型	01	通信类型, 见表 3.16

表 3.15 通信控制子功能——控制类型数值定义

数值	描述	备注
0x00	允许接收和发送	允许服务器报文的发送和接收
0x01	允许接收禁止发送	允许服务器接收报文, 禁止发送报文
0x02	禁止接收允许发送	禁止服务器接收报文, 允许发送报文
0x03	禁止接收和发送	禁止服务器发送和接收报文
0x4~0x7F	保留	

表 3.16 通信类型数值定义

数值	描述	备注
0x01	常规报文	控制常规报文的通信
0x02	网络管理报文	控制网络管理报文的通信
0x03	常规报文和网络管理报文	控制常规报文和网络管理报文的通信

**写入数据服务 (SID=0x2E)**

该诊断服务简称 WDBI 诊断服务， SID 值为 2E。客户端通过该诊断服务可以写入服务器中指定数据标识符对应的数据。

该服务与传输数据服务 (SID=0x36) 的区别在于：写入数据是对固定的标识符所表示的数据进行写入，例如修改 ECU 序列号，写入编程时间等；而传输数据是向服务器内存中存入一段数据，具体功能由用户自定义。

- ① 该诊断服务仅用于编程模式。
- ② 请求报文以：ID1，要写入的数据，ID2，要写入的数据……的形式存放数据到相应的字节中，常使用多帧方式发送。

写入数据服务的应用层协议实现如表 3.17。

表 3.17 写入数据应用层数据

第一帧	数据字节	描述（所有值是 16 进制数）	字节 (Hex)	备注
A_Data	#1	写入数据请求报文 SID	2E	该数据由用户自定义，此例示意 5 个字节的数据
	#2	数据标识符 (DID)	F1	
	#3		8C	
	#4	数据	XX	
	#5		XX	
	#6		XX	
	#7		XX	
	#8		XX	

**例程控制服务 (SID=0x31)**

该诊断服务简称 RC 诊断服务， SID 值为 31。客户端通过该服务控制服务器的例程：启动、停止例程以及请求例程结果。一个例程通过 2 字节的例程 ID 定义。

例程包括可配置循环程序、安全系统程序等等，通过相应的标识符对其进行启动、停止、请求结果的控制。

请求报文一次只能请求控制一个例程（即请求报文中一次只包含一个例程 ID）。只有当启动例程激活后，才能进行停止例程或请求例程结果，否则服务器将返回否定响应。

例程控制服务的应用层协议实现如表 3.18。

表 3.18 例程控制应用层数据

	数据字节	描述(所有值是 16 进制数)	字 节 (Hex)	备注
A_Data	#2	例程控制 SID	31	
	#3	例程控制子功能	01	子功能，见表 3.19
	#4	例程 ID	FF	例程 ID
	#5		00	
标识符 0xFF00 为擦除内存。				

表 3.19 例程控制子功能数值定义

数值	描述	备注
0x00	保留	
0x01	启动例程	该参数用于服务器启动和例程 ID 相对应的例程
0x02	停止例程	该参数用于服务器停止和例程 ID 相对应的例程
0x03	请求例程结果	服务器返回例程结果
其他	保留	

### 诊断设备在线服务 (SID=0x3E)

该诊断服务简称 TP 诊断服务，客户端通过该诊断服务维持和服务器在非默认模式下的链接。

该服务用于维持服务器的会话状态。如前所述，当服务器处于非默认状态下(即扩展模式和编程模式)，若在规定时间内无动作，将自动回到默认模式。客户端为了使服务器不自动跳回，将周期性的发送该服务，以维持和服务器在非默认模式下的链接。

该服务可以为功能寻址，即保持与多个 ECU 的链接。通常，该服务是以正定响应方式进行请求，即 ECU 接收该服务请求时不向客户端进行响应。

诊断仪在线服务的应用层协议实现如表 3.20。

表 3.20 诊断设备在线应用层数据

	数据字节	描述(所有值是 16 进制数)	字节(Hex)	备注
A_Data	#2	诊断设备在线 SID	3E	
	#3	诊断设备在线子功能	80	子功能见表 3.13.2

表 3.21 诊断设备在线子功能数值定义

数值	描述	备注
0x00	需要肯定响应报文	
0x80	不需要肯定响应报文	通常使用该子功能, 请求报文为周期发送

### 控制 DTC 设置服务 (SID=0x85)

该诊断服务简称 CDTCS 诊断服务, SID 值为 85。客户端通过该诊断服务停止或恢复诊断故障码的设置。该服务常与上述“通信控制”服务 (SID 为 28) 服务连用。终止故障诊断码的设置, 预防正常通信被终止后非正常故障码的产生。

控制 DTC 设置服务的应用层协议实现如表 3.22 所示。

表 3.22 控制 DTC 设置请求报文

	数据字节	描述(所有值是 16 进制数)	字节(Hex)	备注
A_Data	#2	控制 DTC 设置 SID	85	
	#3	控制 DTC 设置子功能	01	子功能见表 3.23

表 3.23 控制 DTC 设置子功能数值定义

数值	描述	备注
0x00	保留	
0x01	打开 DTC 设置	可以设置 DTC
0x02	关闭 DTC 设置	不可以设置 DTC
其他	保留	

### 3.3.2 网络层数据打包及算法设计

在数据打包封装过程中, 应用层数据经网络层后会加入网络层的协议控制信息, 实现应用层数据到网络层协议数据单元的打包封装; 网络层数据经数据链路

层进一步封装后，实现报文在某种链路上的封装，作为 CAN 网络，在此数据链路层上则封装成 CAN 报文的形式。

首先，在应用层数据（A-data）的基础上，加上网络层的协议控制信息 N\_PCI，封装成网络层的协议数据单元 N\_PDU，在对应用层数据进行网络层数据打包封装时，将对应用层数据分两种方式进行封装，及单帧数据的封装与多帧数据的封装。如 2.1.2 节所示，在不使用混合寻址的方式下，若应用层数据少于等于 7 个字节则用单帧形式进行处理，若大于 7 个字节则用多帧的形式对应用层数据进行处理。

单帧数据封装方式为：首先，统计应用层数据的数量，统计出来之后，根据 2.1.2 节中单帧的帧格式，在应用层数据之前加入统计出来的应用层数据数量作为网络层的协议控制信息，即完成单帧数据从应用层协议数据单元到网络层协议数据单元的封装。

在上节 3.3.2 中“清除故障码请求”的应用层数据如表 3.24 所示。

表 3.24 清除故障码应用层数据

	数据字节	描述（所有值是 16 进制数）	字节（Hex）	备注
A_Data	#1	清除诊断信息服务 SID	14	
	#2	DTC 群组 ID	C0	
	#3		00	
	#4		00	

其应用层数据为 14 C0 00 00 共 4 个字节，则需在应用层数据前加入 04 作为其网络层的协议控制信息，经网络层封装的数据则变为如表 3.25 所示。

表 3.25 清除故障码网络层打包数据

	数据字节	描述（所有值是 16 进制数）	字节（Hex）	备注
N_PCI	#0	网络层协议控制信息	04	表示四字节数据
A_Data	#1	清除诊断信息服务 SID	14	
	#2	DTC 群组 ID	C0	
	#3		00	
	#4		00	

多帧数据封装方式为：首先统计应用层数据的数据量，如图 3.6，假设一共有 A\_databyt 个字节的数据，则根据 2.1.2 节中第一帧的帧格式，A\_databyt 将作为第一帧的协议控制信息的前两个字节。假设 A\_databyt 需发送帧的数量为

`Frm_amount`, 根据网络层的帧格式, 第一帧只携带 6 个字节的数据, 后续帧携带 7 字节的数据, 则需打包帧的数量应按如下方式计算: 应用层数据量 `A_databyt` 加 1 再除以 7, 若所计算的值为整数, 则需发送帧的数量为 `Frm_amount`, 即公式 1 所示; 若不为整数, 应在所得的整数值的基础上加再 1, 即为发送帧的数量为 `Frm_amount`, 如公式 3.2 所示, 其中, 除式的余数为最后一帧所携带的字节数量, 假设为 `m`。

应用层所有字节数量如下:

1	2	3	...	<code>A-databyt</code>
---	---	---	-----	------------------------

图 3.6 所发送字节

打包帧的数量计算公式:

$$\text{Frm\_amount} = (\text{A\_databyt} + 1) / 7 \quad (3.1)$$

或  $\text{Frm\_amount} = (\text{A\_databyt} + 1) / 7 + 1 \quad (3.2)$

假设发送帧数量 `Frm_amount` 的值小于等于 16, 则应用层数据应打包成如图 3.7 形式, `Frm` 为 `Frm_amount` 的 16 进制值;

<code>A_databyt</code>	1	2	...	
21			...	
22			...	
.			...	
2 <code>Frm</code>	1	...	<code>m</code>	...

图 3.7 发送帧数量小于 16 的数据打包方式

若 `Frm_amount` 的值大于 16, 则后续帧第一字节协议控制信息的值应为 0x20~0x2F 循环编号, 假设打包循环次数为 `0Fcycnum`, 则其计算公式如下两种, 若结果为整数则为公式 3.3; 否则为公式 3.4, 取结果的整数值再加 1, 其余数值再调用上述公式 1 或公式 2 进行剩余帧发送数量的计算。此时应按如图 3.8 方式进行打包。

以 0-F 共 16 帧数据为循序周期, 进行循环打包次数的计算公式为:

$$0Fcycnum = (\text{A\_databyt} + 1) / (16 * 7) \quad (3.3)$$

$$0Fcycnum = (\text{A\_databyt} + 1) / (16 * 7) + 1 \quad (3.4)$$

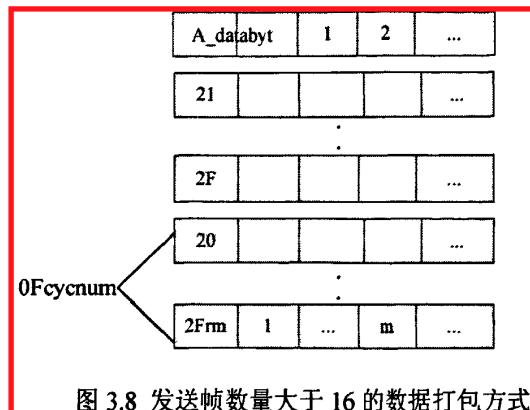


图 3.8 发送帧数量大于 16 的数据打包方式

多帧数据封装打包算法流程如图 3.9 所示。首先统计要发送数量字节量的大小 A\_databyt，然后代入公式 1 计算帧数量 Frm\_amount，若不为整数取其整数值再加 1；接下来将所发送数量 A\_databyt 代入公式 3 计算循环打包次数，若不为整数，其值取整数再加 1；开始循环发送报文，并记录循序发送次数，发送完一个周期将计数减 1，在一个发送周期内，记录每帧报文的计数，每发送一帧，记录数加 1，直到记录数加到 16，一个周期结束后，进入下一个发送周期，循环发送。直到循环发送记录数为 0 即数据发送完毕。

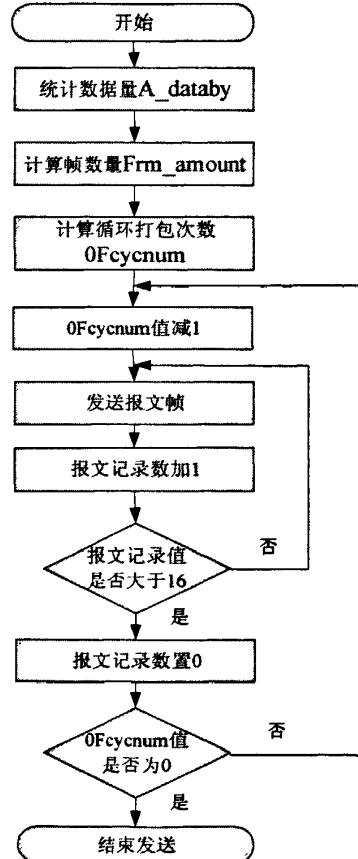


图 3.9 多帧数据封装打包算法流程

按照上述多帧数据的打包算法，对在上节“写入如数据”服务应用层数据进行打包传输，“写入如数据”服务应用层数据如表 3.26 所示。

表 3.26 写入如数据服务应用层数据

第一帧	数据字节	描述(所有值是 16 进制数)	字节 (Hex)	备注
A_Data	#1	写入数据请求报文 SID	2E	
	#2	数据标识符 (DID)	F1	0xF18C—ECU 序列号
	#3		8C	
	#4	数据	XX	5 个字节的数据
	#5		XX	
	#6		XX	
	#7		XX	
	#8		XX	

此服务的应用层数据量为 8 字节，即  $A\_databyt=8$ ，则需发送报文帧的数量  $Frm\_amount$  应为  $(A\_databyt+1)/7+1 = (8+1)/7+1=1+1=2$ ，余数为 2，即最后一帧所携带数据量为 2 个字节。循环打包次数  $0Fcycnum=(A\_databyt+1)/(16*7)+1=0+1=1$ ，使用一个循环周期即可发送完毕。经网络层的多帧数据打包封装后的第一帧数据应如表 3.27 所示。

表 3.27 写入如数据服务网络层第一帧数据

第一帧	数据字节	描述(所有值是 16 进制数)	字节 (Hex)	备注
N_PCI	#0	第一帧的第一字节	10	
	#1	第一帧的第二字节	08	
A_Data	#2	写入数据请求报文 SID	2E	
	#3	数据标识符 (DID)	F1	0xF18C—ECU 序列号
	#4		8C	
	#5	数据	XX	前三个字节数据
	#6		XX	
	#7		XX	

超出六个字节应打包为后续帧如表 3.28 所示。

表 3.28 写入如数据服务网络层后续帧数据

后续帧	数据字节	描述(所有值是 16 进制数)	字节(Hex)	备注
N_PCI	#0	后续帧的第一帧	21	
A_Data	#1	数据	XX	剩余两字节
	#2		XX	

### 3.3.3 安全访问及算法设计

安全访问过程本质见图 3.10，首先，测试仪向 ECU 发送请求种子请求，ECU 返回种子，测试仪根据接收到的种子和安全算法计算出密钥，再向 ECU 发送密钥请求，ECU 再根据收到的密钥判断密钥是否正确，并给予响应<sup>[31]</sup>。

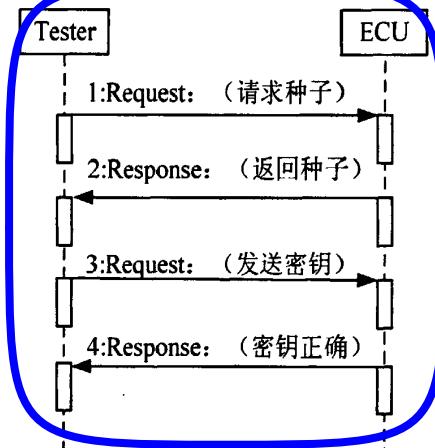


图 3.10 安全访问过程

对 ECU 进行安全访问前，首先应使 ECU 进入某种会话模式下，然后才对 ECU 进行相关的后续诊断服务实施。并对 ECU 进行安全访问服务的诊断协议实现执行流程，如图 3.11 所示。

首先，ECU 进行非默认模式会话请求，需注意的是，ECU 上电时默认的诊断会话状态为默认会话模式，因此，需要进行诊断会话模式跳转才能执行安全访问服务。然后，进行安全等级选择并请求种子，根据接收到的种子及安全访问算法发送密钥，收到 ECU 正定响应后则 ECU 被解锁。其中，安全算法可自行设计，在此设计了三种安全等级的安全算法，分别为等级 1、等级 2 和等级 3，算法依次复杂。

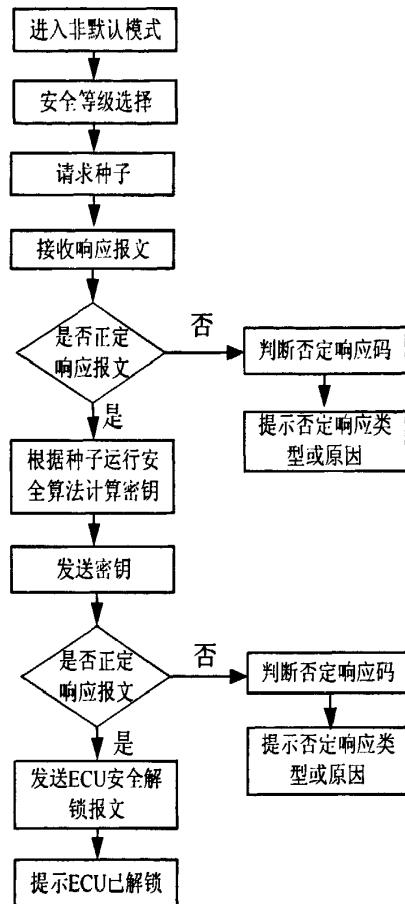


图 3.11 安全访问执行流程图

安全算法是以动态链接库的形式提供给上位机供上位机调用。在此设计的三种安全等级的安全算法。安全算法执行流程图如图 3.12 所示，首先获取种子，然后根据选择的安全算法等级选择不同的安全算法。

三个等级的安全算法设计如下：

安全等级一算法设计：

当选择安全等级为 1 的安全访问请求时，此时使用两字节随机种子，安全算法为，将两字节种子种子的高低字节互换，即得到密钥的值。若服务器响应的 16 进制随机种子为 15 F1，则当选取安全等级为 1 的安全算法后，所算得的密钥应为 16 进制的 F1 15。

安全等级二算法设计：

当选择安全等级为 2 的安全访问请求时，此时使用两字节随机种子，安全算法为，将两字节种子的值加 9，即得到密钥的值。若服务器响应的 16 进制随机种子为 15 F1，则当选取安全等级为 2 的安全算法后，所算得的密钥应为 16 进制的 15 FA。

### 安全等级三算法设计：

当选择安全等级为 2 的安全访问请求时，此时使用两字节随机种子，安全算法为，将两字节种子的值进行按位取反，即得到密钥的值。若服务器响应的 16 进制随机种子为 15 F1，则当选取安全等级为三的安全算法后，所算得的密钥应为 16 进制的 EA 0E。

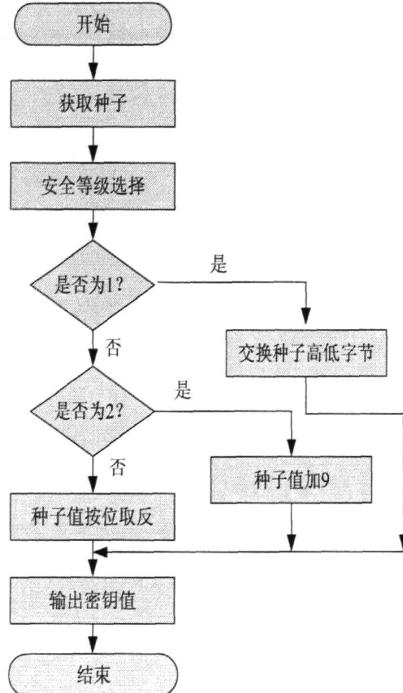


图 3.12 安全算法流程图

动态链接库采用动态添加的方式如图 3.13 所示，通过点击“浏览”按钮选择动态链接库 dll 文件对安全算法动态链接库进行动态导入<sup>[32]</sup>，导入后将显示动态链接库的路径，如图 3.14 所示。

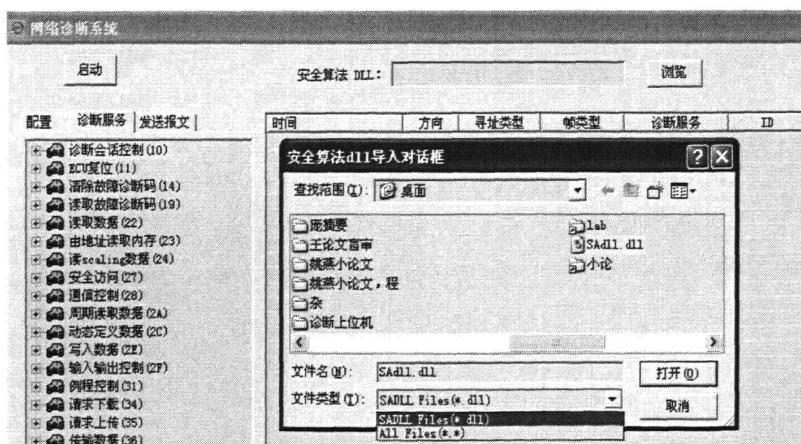


图 3.13 安全算法动态链接库的导入

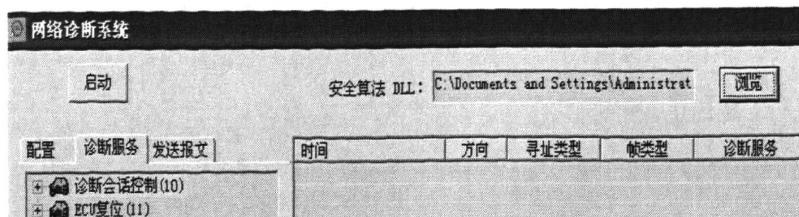


图 3.14 导入安全算法动态链接库后显示

导入安全算法动态链接库后，在执行安全访问服务时，根据所得的种子与请求的安全等级将自动调用安全算法动态链接库计算密钥值。

**动态调用动态链接库函数计算密钥代码如下：**

```
HINSTANCE hInst; //定义一个实例句柄变量
hInst = LoadLibrary(dllpath);
typedef unsigned char* (*ADDPROC)(const unsigned char* seed, const unsigned int
SecurityLevel); //函数指针
ADDPROC genertkey = (ADDPROC)GetProcAddress(hInst,MAKEINTRESOURCE(1));
key=genertekey(seed,frameinfo[i].Data[2]);
```

## 3.4 辅助功能模块设计

辅助功能模块主要作为诊断功能的辅助模块，包括硬件的配置功能模块、报文的显示与分析功能模块、正常报文发送功能模块、文件存储功能模块。

### 3.4.1 硬件配置模块设计

硬件配置模块需实现对硬件的 CAN 路选择、硬件通信波特率及发送模式等的配置。硬件配置的参数主要通过初始化函数结构体 VCI\_INIT\_CONFIG 实现对硬件相关通信参数的配置。结构体将在 VCI\_InitCan 函数中被填充，其中结构体内部需初始化的配置参数为如下所示的结构体变量。

```
typedef struct _INIT_CONFIG {
    DWORD AccCode; //验收码
    DWORD AccMask; //屏蔽码。
    DWORD Reserved; //保留。
    UCHAR Filter; //滤波方式。
    UCHAR Timing0; //定时器 0 (BTR0)
```

```

UCHAR Timing1; //定时器 1 (BTR1)
UCHAR Mode; //模式
} VCI_INIT_CONFIG, *PVCI_INIT_CONFIG;

```

配置参数通过“配置”页面的参数设置来实现参数的人工配置，其参数配置页面<sup>[33]</sup>如图 3.15 所示。

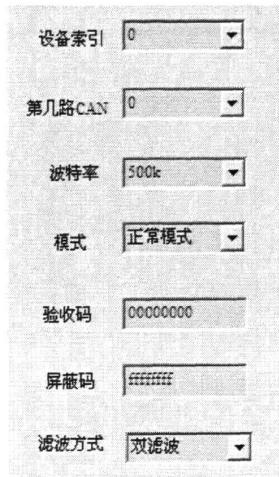


图 3.15 硬件配置模块设计

设备索引，第几路 CAN，波特率，模式和滤波方式以组合框的形式给用户以选择不同的选项；验收码和屏蔽码以编辑框的形式供用户随即输入所需屏蔽的报文 ID，其中，验收码为 00000000，屏蔽码为 ffffffff，表示不对任何报文进行屏蔽。设备索引默认为 0；第几路 CAN 可选择 0 或 1，分别选择第一路或第二路 CAN；模式可选择正常模式与只听模式两种，正常模式下可发送或接收报文，只听模式下只能接收不能发送报文。滤波方式可选择双虑波或单滤波模式。波特率可选择 5k~1000k 范围，波特率是通过定时器 0 和定时器 1 的值来设置的，比如波特率为 500kbps 时，定时器 0 和定时器 1 的值分别为十六进制的 0x00 和 0x1C，波特率值与两个定时器值对应关系如表 3.29 所示。

表 3.29 波特率值与两个定时器值映射关系表：

CAN 波特率	定时器 0	定时器 1
5kbps	0xBF	0xFF
10kbps	0x31	0x1C
20kbps	0x18	0x1C
40kbps	0x87	0xFF
50kbps	0x09	0x1C

80kbps	0x83	0xFF
100kbps	0x04	0x1C
125kbps	0x03	0x1C
200kbps	0x81	0xFA
250kbps	0x01	0x1C
400kbps	0x80	0xFA
500kbps	0x00	0x1C
666kbps	0x80	0xB6
800kbps	0x00	0x16
1000kbps	0x00	0x14

响应函数下，配置页面下通过组合框变量的下拉框选择响应的波特率。部分波特率设置代码如下：

```

baudrate=m_page1.m_Rate.GetCurSel();
switch(baudrate)
{
    case 1: //100k
        init_config.Timing0=0x04;
        init_config.Timing1=0x1c;
        break;
    case 2: //125k
        init_config.Timing0=0x03;
        init_config.Timing1=0x1c;
        break;
    case 3: //250k
        init_config.Timing0=0x01;
        init_config.Timing1=0x1c;
        break;
    case 4: //500k
        init_config.Timing0=0x00;
        init_config.Timing1=0x1c;
        break;
}

```

将配置页面人工配置的相关参数传递给结构体 VCI\_INIT\_CONFIG 的每一个变量，再将初始化结构体传递给设备初始化函数 VCI\_InitCan 既可以实现对硬件的初始化处理。

### 3.4.2 报文显示与分析模块设计

报文显示与分析模块，实现对发送和接收的报文内容进行详细分析及显示，见图 3.16，包括时间栏，报文传输方向，诊断诊断的类型，诊断服务类型，报文的 ID，报文的长度，以及报文的数据。

在 VC 中以报表控件的方式分栏显示，使得其显示相对清晰。

时间	方向	诊断网络	帧类型	诊断服务	ID	长度	数据

图 3.16 报文显示与分析模块设计

时间栏显示的为报文显示的当前系统时间，用 MFC 封装的 CTime 类的 GetCurrentTime 方法可获得系统的当前时间。

```
CTime time= CTime::GetCurrentTime();
str = time.Format("%Y/%m/%d %H:%M:%S");
```

方向栏的显示，当在接收线程时，接收到的报文即将显示栏设置为“接收”，但当在接收线程里判断接收的报文，并响应报文，则应将显示栏设置为“发送”；当双击树形控件发送诊断服务请求时，将显示栏设置为“发送”。

对帧类型的判断，诊断帧有单帧、第一帧、流控帧和后续帧四种帧类型，取诊断报文的第一字节的高四位进行判断，若值为 0 则为单帧，值为 1 为第一帧，值为 2 为后续帧，值为 3 为流控帧。

对诊断服务的判断，对诊断服务种类进行判断与显示前，首先进行帧类型的判断，若为单帧则取第二个字节进行判断，若为第一帧则取第三个字节进行判断，若为流控帧或后续帧，则不对诊断服务种类进行判断与显示，因为此两类帧不含有诊断服务信息。其判断流程如图 3.17 所示。

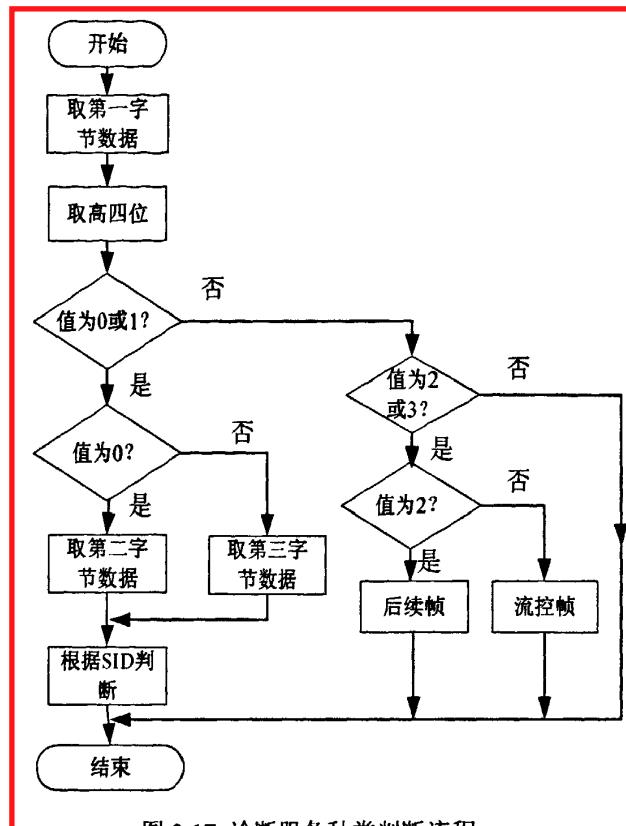


图 3.17 诊断服务种类判断流程

部分实现代码如下：

```

switch(data1)
{
case 0:
    str="单帧";
    box->SetItem(index,3,LVIF_TEXT,str,0,0,0,0);
    diagserv(frameinfo[i].Data[1],a); //单帧判断诊断服务类型
    box->SetItem(index,4,LVIF_TEXT,a,0,0,0,0);
    break;
case 1:
    str="第一帧";
    box->SetItem(index,3,LVIF_TEXT,str,0,0,0,0);
    diagserv(frameinfo[i].Data[2],a); //第一帧判断诊断服务类型
    box->SetItem(index,4,LVIF_TEXT,a,0,0,0,0);
    break;
case 2:
    str="后续帧";
}
  
```

```

box->SetItem(index,3,LVIF_TEXT,str,0,0,0,0);
break; //后续帧不予判断

case 3:
str="流控帧";
box->SetItem(index,3,LVIF_TEXT,str,0,0,0,0);
break; //流控帧不与判断

```

诊断服务判断函数通过取出诊断服务的 SID 来判断诊断服务种类，并将诊断服务类型传给 Cstring 类型的 str1 参数，然后给以显示。

判断诊断服务种类函数实现部分如下：

void CTabcontrDlg::diagserv(unsigned char data11,CString &str1) //引用

报文显示与分析模块的效果如图 3.18 所示：

时间	方向	诊断网络	帧类型	诊断服务	ID	长度	数据
2011/03/22 11:48:08	发送	CAN网诊断	单帧	诊断会话请求	00000766	08	02 10 01 55 55 55 55 55
2011/03/22 11:48:08	接收	CAN网诊断	单帧		00000746	08	06 50 01 00 32 13 88 00
2011/03/22 11:48:13	发送	CAN网诊断	单帧	读取数据	00000766	08	03 22 f1 87 55 55 55 55
2011/03/22 11:48:13	接收	CAN网诊断	第一帧		00000746	08	10 12 62 f1 87 33 38 32
2011/03/22 11:48:13	发送	CAN网诊断	流控帧		00000766	08	30 00 00 55 55 55 55 55
2011/03/22 11:48:13	接收	CAN网诊断	后连接帧		00000746	08	21 30 31 30 30 58 4a 5a
2011/03/22 11:48:13	接收	CAN网诊断	后续帧		00000746	08	22 30 38 41 00 00 00 00

图 3.18 读数据诊断服务过程的诊断通信报文

### 3.4.3 正常报文发送模块设计

当需要发送正常通信报文以检测或测试网络的某种响应，此时不需要发送诊断报文，因此需要设计一个发送正常报文的页面以实现上位机的灵活性，此外，在此页面下通过手动输入报文，可实现某些诊断服务的测试，比如诊断服务过长

正常报文发送模块可实现对任意 CAN 报文的发送，可单次发送，或循环发送，以实现对网络的某些测试功能。报文发送页面的主要配置参数有，帧类型的选择，报文 ID 输入，报文数据输入，报文的循环发送，发送方式的选择，如图 3.19 所示。

帧类型和发送方式以组合框的形式进行设置。帧类型可选择标准帧和扩展帧；发送方式有正常发送、单次发送、自发自收和单次自发自收。正常发送方式下需两个节点以上进行报文的发送与接收，自发自收发送方式下支持一个节点的自发自收。

循法发送选项可以对报文进行每隔一定的时间进行循环发送，或每次发送固定数量帧的方式发送，其中每隔一定时间发送报文时的时间单位为毫秒 (ms)，并且两种方式不能同时选择，只能选择其中一种循环发送方式进行发送。

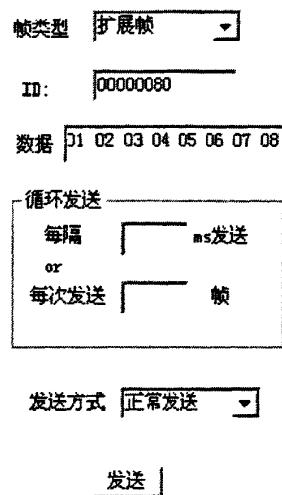


图 3.19 正常报文发送模块设计

在数据输入时，需对数据的输入进行校验，当输入错误时应对其进行提示，判断数据格式流程如图 3.20 所示，首先取编辑框 ID 内容和数据编辑框内容，若编辑框内容为空则弹出提示对话框“请输入数据”；取编辑框 ID 数据的长度，若大于 8，则弹出提示对话框“ID 超出范围”，ID 最长为 29 位，用 32 位表示即可；取编辑框数据长度，若大于 24，则弹出提示对话框“数据长度超过范围，最大为 8 个字节”。

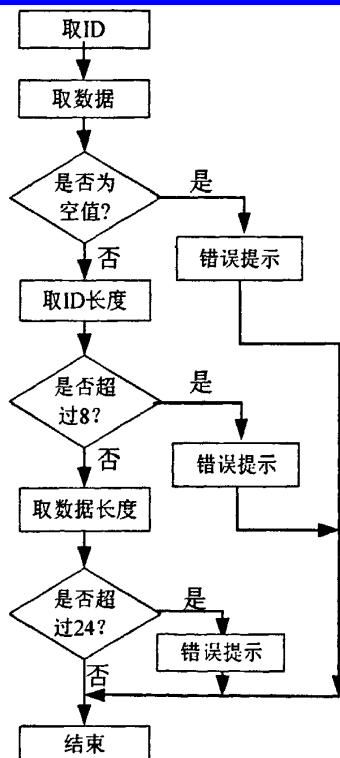


图 3.20 数据输入格式判断

### 3.4.4 文件存储模块设计

诊断通信的报文的内容可直接反应被诊断 ECU 的状况，因此有必要对诊断通信报文进行保存，以分析被诊断 ECU 的具体信息。使用 CStdioFile 类来创建一个 TXT 文件对诊断报文进行保存。用列表控件变量的 GetItemText 方法获得每行数据元素的内容，将数据按行取出后放到一个 CString 类型的变量中，即将每一行的时间、方向、帧类型、ID、数据等放到 CString 变量中，然后用 CstdioFile 的 WriteString 方法将内容按行写入文件。下对读数据诊断服务过程的诊断通信报文保存成如下 txt 文档的形式：

时间	方向	诊断网络	帧类型	诊断服务	ID	长度	数据
2011/03/22 11:48:08	发送	CAN网诊断	单帧	诊断会话请求	00000766	08	02 10 01 55 55 55 55 55
2011/03/22 11:48:08	接收	CAN网诊断	单帧		00000746	08	06 50 01 00 32 13 08 00
2011/03/22 11:48:13	发送	CAN网诊断	单帧	读取数据	00000766	08	03 22 F1 87 55 55 55 55
2011/03/22 11:48:13	接收	CAN网诊断	第一帧		00000746	08	18 12 62 F1 87 33 38 32
2011/03/22 11:48:13	发送	CAN网诊断	遥控帧		00000766	08	30 00 00 55 55 55 55 55
2011/03/22 11:48:13	接收	CAN网诊断	后续帧		00000746	08	21 38 31 38 38 58 4a 5a
2011/03/22 11:48:13	接收	CAN网诊断	后续帧		00000746	08	22 38 38 41 00 00 00 00

图 3.20 读数据诊断服务过程的诊断通信报文保存

## 3.5 本章小结

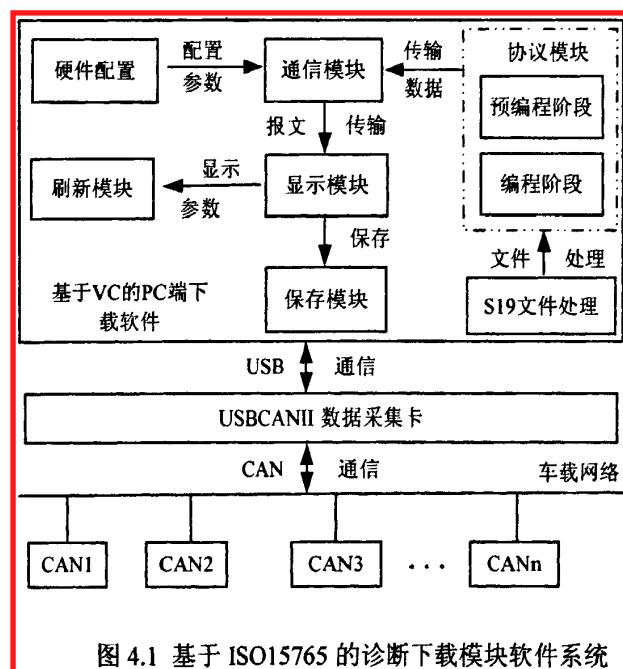
本章设计了常规诊断模块的功能实现，在主要的诊断功能模块中设计了应用层的诊断服务、应用层数据到网络层的数据打包算法，以及安全访问服务的安全访问算法等诊断协议的实现。此外，还设计了上位机软件的硬件配置模块、报文显示与分析模块、正常报文发送模块以及文件存储等模块，以实现常规诊断模块的基本功能及其扩展功能。

## 第四章 基于 ISO15765 的上位机下载诊断模块设计

本章在第二章下载诊断模块的功能需求以及第三章的诊断协议实现基础上,以飞思卡尔 S12 系列单片机为研究对象,以 USBCANII 为下载器硬件,以 VC 软件作为其开发平台,研究分析基于 ISO15765 的上传下载诊断协议通信流程,对下载诊断模块进行详细的功能设计,主要包括整个软件系统的功能设计、下载过程预编程阶段的诊断功能设计、编程阶段的诊断设计,以及对所下载文件—S19 文件的程序算法处理。最终实现了低成本的基于 CAN 线的在线下载器设计。

### 4.1 基于 ISO15765 的下载系统

下载器系统所设计的整个系统功能结构与第三章的普通诊断模块软件系统的功能结构类似,诊断网络结构也是采用客户端与服务器直接相连并处于同一网络的诊断结构。硬件采集卡仍为 USBCANII,其硬件的配置与 CAN 网络通信已在第三章中得以实现。此时可将正常报文发送模块和报文分析模块去掉,主要解决的问题是文件下载功能模块的实现,即预编程阶段功能模块,编程阶段功能模块,以及 S19 文件处理模块。



## 4.2. S19 文件解析及算法设计

所需下载的文件有两种格式，S19 格式的文件和HEX 格式的文件，本章主要针对飞思卡尔单片机所生成的 S19 格式文件进行相关解析与处理。因所生成的 S19 文件中，含有不需下载到 FLASH 中的无用信息，因此需对 S19 格式的文件进行相关处理。

### 4.2.1 S19 文件格式解析

S19 格式文件是 Freescale CodeWarrior 编译器生成的后缀名为.S19 的程序文件，它是一段能直接被烧写进 MCU 的 ASCII 码，全称为 Motorola format for EEPROM programming。

S19 文件中，每行最多有 78 个字节，一字节由两字符组成，即每行最多有 156 个字符。 S19 文件每行的格式为如下形式：都由类型、计数、地址、数据和校验和组成<sup>[34]</sup>。

S19 文件格式如下所示：

type	count	address	data	checksum
------	-------	---------	------	----------

type(类型)：2 个字符。用来描述记录的类型 (S0, S1, S2, S3, S5, S7, S8, S9)。

count(计数)：2 个字符。用来组成和说明了一个 16 进制的值，显示了在记录中剩余成对字符的计数。

address(地址)：4 或 6 或 8 个字节（字符）。用来组成和说明了一个 16 进制的值，显示了数据应该装载的地址，这部分的长度取决于载入地址的字节数。2 个字节的地址占用 4 个字符，3 个字节的地址占用 6 个字符，4 个字节的地址占用 8 个字符。

data(数据)：0—64 字符。用来组成和说明一个代表了内存载入数据或者描述信息的 16 进制的值。

checksum(校验和)：2 个字符。这些字符当被配对并换算成 16 进制数据的时候形成了一个最低有效字符节，该字符节用来表达作为补充数据，地址和数据库的字符对所代表的（字节的）补码的 byte 总和。即计数值、地址场和数据场的若干字符以两个字符为一对，将它们相加求和，和的溢出部分不计，只保留最低两位字符 NN，checksum =0xFF-0xNN。

S0 Record：记录类型是“S0”(0x5330)。地址场没有被用，用零置位(0x0000)。

数据场中的信息被划分为以下四个子域：

name(名称): 20 个字符, 用来编码单元名称

ver(版本): 2 个字符, 用来编码版本号

rev(修订版本): 2 个字符, 用来编码修订版本号

description(描述): 0-36 个字符, 用来编码文本注释

此行表示程序的开始, 不需烧入 memory。

S1 Record: 记录类型是“S1”(0x5331)。地址场由 2 个字节地址来说明。数据场由可载入的数据组成。

S2 Record: 记录类型是“S2”(0x5332)。地址场由 3 个字节地址来说明。数据场由可载入的数据组成。

S3 Record: 记录类型是“S3”(0x5333)。地址场由 4 个字节地址来说明。数据场由可载入的数据组成。

S5 Record: 记录类型是“S5”(0x5335)。地址场由 2 字节的值说明, 包含了先前传输的 S1、S2、S3 记录的计数。没有数据场。

S7 Record: 记录类型是“S7”(0x5337)。地址场由 4 字节的地址说明, 包含了开始执行地址。没有数据场。此行表示程序的结束, 不需烧入 memory。

S8 Record: 记录类型是“S8”(0x5338)。地址场由 3 字节的地址说明, 包含了开始执行地址。没有数据场。此行表示程序的结束, 不需烧入 memory。

S9 Record: 记录类型是“S9”(0x5339)。地址场由 2 字节的地址说明, 包含了开始执行地址。没有数据场。此行表示程序的结束, 不需烧入 memory。

根据不同的描述信息, 在以上三种不同的结束行中选择一种使用

例如有如下 S19 文件

S123C000CF1400790011CC09395B105A124A8046304A8000300001C01BCB731400  
07340027

则此行的地址为两字节 C000, count=23 表示 23 后有 35 个字节的数据 (包括地址和校验和)。

#### 4.2.2 S19 文件处理算法设计

对如上格式的文件, 由于文件中含有一些无用信息, 并且要对所发送的数据以 CAN 报文的形式进行打包处理, 因此, 要对原始 S19 文件进行处理, 算法如图 4.2 所示。

首先, 打开文件, 用 ReadString 方法将文件按行读取到 CString 类型的 cstmp 变量中, 此时要把无用的字节信息去除, 比如开头行、末尾行以及每行的开头标

识等，开头行通过 `cstmp[1]` 是否为‘0’判断，末尾行通过 `cstmp[1]` 是否为‘8’或是否为‘9’判断。通过所设计 `S19Data` 类的 `Convert` 函数对读取的每行数据进行处理。然后将处理后的每行数据添加到以 `S19Data` 类对象为元素的链表中，对 `m_S19DataList` 链表类型的声明为：`CList<S19Data,S19Data&> m_S19DataList;` 循环处理每行数据，并将处理好的数据添加到链表，以备发送。

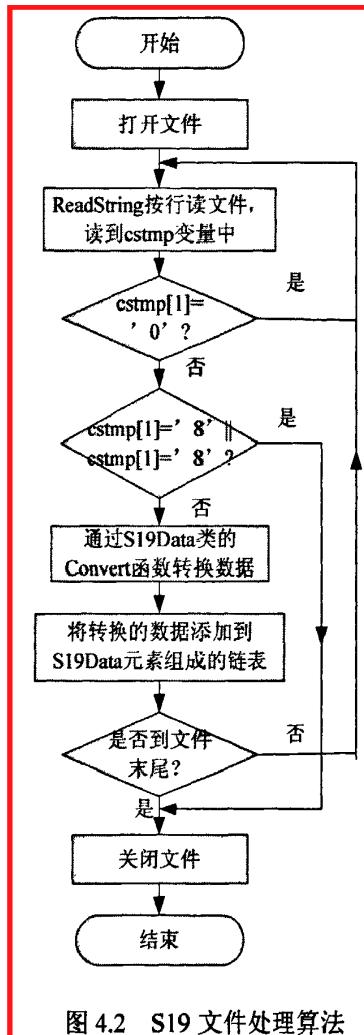


图 4.2 S19 文件处理算法

对链表中数据的读取，以及转换成以 8 字节数据为单位的 CAN 报文帧并发送的代码如下：

```

ps=m_S19DataList.GetHeadPosition(); //找到链表的初始位置
S19Data s19=m_S19DataList.GetAt(ps); //取初始元素
for (int i=0;i<s19.m_Length;i+=8) //以 8 字节为单位发送数据
{
    memset(data,0,8);
    if(s19.m_Length-i>8)
  
```

```

        memcpy(data,s19.m_Data+i,8);
    else
    {
        memcpy(data,s19.m_Data+i,s19.m_Length-i);
    }
    memcpy(&frameinfo.Data,data,8);
    if(VCI_Transmit(4,0,0,&frameinfo,1)==0)
    {
        AfxMessageBox("发送失败");
        break;
    }
}
m_S19DataList.GetNext(ps); //找到链表中下个元素的位置;

```

对文件每行元素进行处理的 S19Data 类的设计如下：

```

class S19Data
{
public:
void Convert(CString cstr);
int m_Length; //数据长度
BYTE* m_Data; //存放转换过来的数据
S19Data();
virtual ~S19Data();
};

```

整型 m\_Length 用来存放每行数据的数据长度，以统计数据量。 **BYTE** 型

m\_Data 用来存放转换过来的数据，以备发送。Convert 函数将从文件读取的每行数据转换成 **BYTE** 型数据放于 m\_Data 中， S19Data 类中 Convert 函数处理文件每行数据算法流程如图 4.3 所示。

首先将 **CString** 类型的元素转换成 **char\*** 型 **record** 变量中以便处理，然后取此行的数据长度，即第三和第四个元素并将其转换成整形放于 m\_length 变量中，在发送数据时会将 m\_length 变量作为循环条件对数据进行循环发送。然后判断为 S1 类型还是 S2 类型的数据，若为 S1 型，则将其地址前补 0，补充为三字节的地址长度，此时因多加一字节的地址，则为 m\_Data 分配 m\_length+1 大小的内存空间用于存储转换过来的 **BYTE** 型数据。若为 S2 则无须加 1，S2 时即为三字节的地址。循环转换数据并存放到 m\_Data 中。

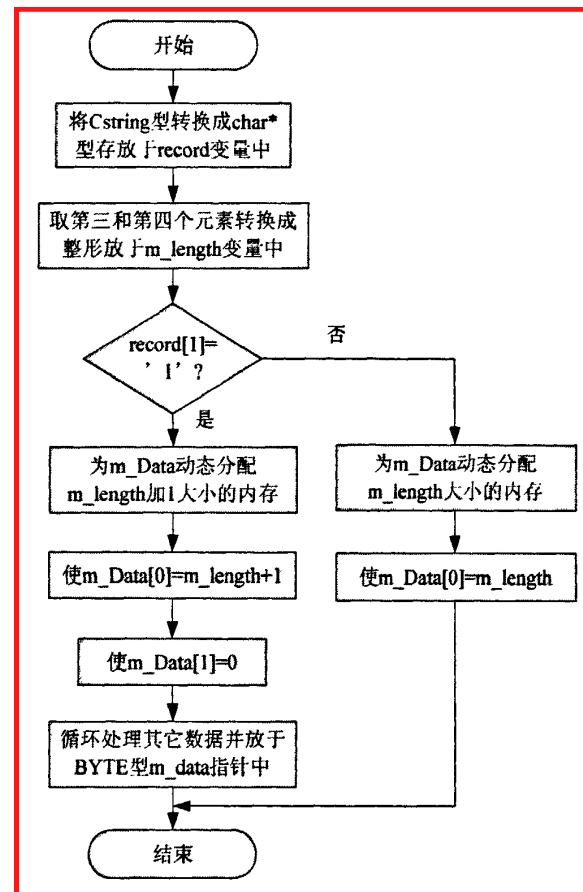


图 4.3 Convert 函数处理文件每行数据算法流程

通过对上传下载诊断协议的研究与实现以及对所下载文件的处理与协议实现，选取硬件 USBCANII 实现网络报文的发送与接收，软件选用 VC 作为开发平台，实现对下载器的开发。所开发的下载器界面如图 4.6 所示，首先实现对硬件的配置，有通信波特率及滤波等的设置。分别对预编程阶段与编程阶段的诊断服务的实现做成按钮的形式进行诊断报文的报送。

### 4.3 基于 ISO15765 的上位机下载软件设计

在 ISO15765-3 中，对在线下载分为预编程阶段和编程阶段两个主要阶段进行实施。针对下载过程的两个主要阶段，本节分别对预编程阶段和编程阶段进行详细分析与设计。

#### 4.3.1 预编程阶段实现

预编程阶段是做 ECU 编程阶段的准备工作，在此阶段要进入扩展诊断会话模

式来终止对故障码的检测，终止正常报文的通信等，以保证编程下载阶段的正常进行。

预编程阶段诊断服务执行流程如图 4.4 所示，执行步骤如下：（括号内为应发送的诊断通信报文）<sup>[35]</sup>

- a.发送扩展模式请求 (02 10 03)，发送方式为功能寻址，使所有 ECU 都进入扩展模式下。因为 ECU 一上电是进入默认模式下，此时需对 ECU 进行模式跳转，跳转到扩展模式下执行终止 DTC 等诊断服务。
- b.诊断仪在线服务 (02 3e 80)，此设计采用每隔 3s 发送一次以确保 ECU 处于非默认模式，维持正常诊断会话，若 5s 内 ECU 未收到诊断报文，ECU 将自动跳回默认模式。
- c.发送终止 DTC 设置服务 (02 85 10)，发送方式仍为功能寻址，使所有 ECU 都停止检测故障码，预防在诊断通信方式下非正常故障码的产生，因为此时正常通信被终止。
- d.进行通信控制服务请求 (03 28 00 01)，用于终止正常通信，保证数据在下载时的传输速率以避免被干扰。

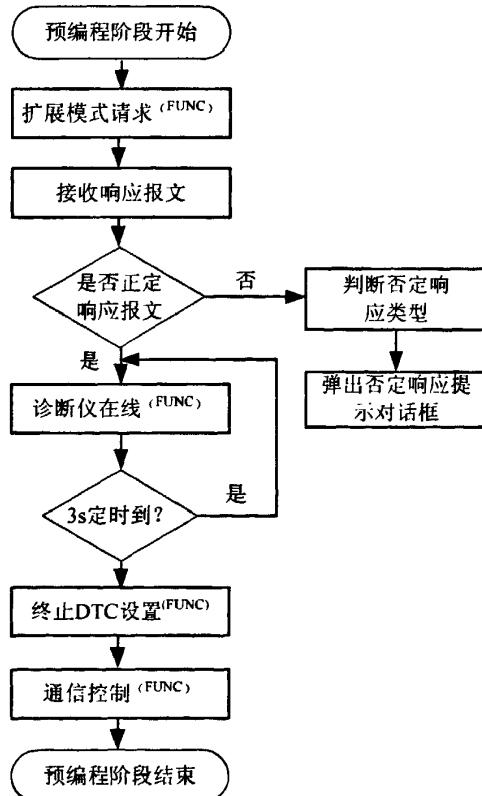


图 4.4 预编程阶段诊断服务执行流程

由于采用常规寻址类型，假设某个 ECU 的请求报文物理地址为 766，功能地

址为 760，响应报文 ID 为 7a6。则预编程阶段基于 ISO15765 的诊断通信报文如下：

760	Rx	02 3E 80 00 00 00 00 00 00
766	Rx	02 10 03 00 00 00 00 00 // 切到扩展诊断模式
7a6	Tx	06 50 03 00 14 03 E8 00 //P2 = 20ms , P2* = 1000ms
760	Rx	02 85 82 00 00 00 00 00
760	Rx	03 28 81 01 00 00 00 00

#### 4.3.2 编程阶段实现

编程阶段首先要使 ECU 进入编程会话模式下，然后对 ECU 解锁才能进行后续下载工作，主要下载三段程序，首先下载擦除程序，再下载写入程序，最后下载应用程序。传输完毕后要对所传输数据进行校验，最后要对 ECU 进行复位<sup>[36]</sup>，编程阶段的诊断服务执行流程图 4.5 所示，步骤如下：（括号内为发送的诊断通信报文，其中\*表示第一帧 PCI、DID 或参数等信息）

- a. 编程模式会话请求 (02 10 02)，使 ECU 进入编程模式下，可进行程序下载服务。此模式下仍每隔 3s 发送诊断仪在线服务以维持诊断会话模式。
- b. 安全访问服务执行 (27)，ECU 处于安全保护状态下，需对其进行解锁才能进行数据的写入与下载。
- c. 请求下载 (\*\* \* 34 00 33 \* \* \* \* \* \*），告知 ECU 要发送数据的起始地址与数据量大小，此时要从 S19 文件中取出所需的起始地址，并计算出所要发送的数据总量。
- d. 传输数据 (\*\* \* 36 \*\*），根据传输协议对数据进行打包发送，此时应将 S19 文件中所要发送的应用层数据加入传输层协议控制信息以备打包传输。
- e. 请求传输退出 (01 37)，要传送数据传输完后，要发送请求传输退出服务退出传输。
- f. ECU 复位 (02 11 01)，程序下载并校验成功后对 ECU 进行复位，有软件复位与硬件复位两种方式。

其中安全访问服务在排放与安全系统下是强制执行的，需对 ECU 进行解锁，其访问方法是上位机首先选择一安全等级的请求种子服务，ECU 接收种子请求后返回一随机种子。上位机再根据所得随机种子与安全算法将得到密钥发送给 ECU，ECU 也根据相同安全算法计算再来判断上位机的密钥是否正确，若判断不正确向 ECU 发送否定响应码，ECU 仍处于锁定状态，若正确发送正定响应 (02 67) 给 ECU 解锁。ECU 解锁后可对其写入数据及下载程序。

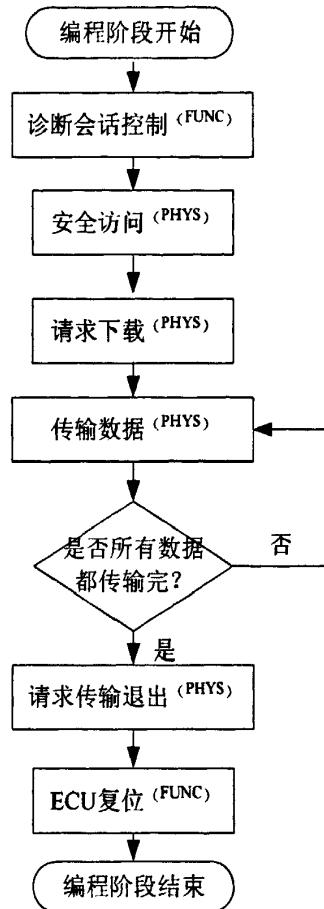


图 4.5 ECU 编程阶段流程图

此外，对某个 ECU 下载完成后，应将整个网络恢复到正常状态下，使能正常通信，使能 DTC 的在线监测，使所有 ECU 回到默认模式下。此软件右侧界面为对通信报文的检测，可对诊断通信报文进行实时的检测与显示，同时可将通信报文保存起来以对通信过程及所下载文件进行分析。

与目标 ECU 相接，执行预下载阶段、下载阶段的诊断服务，以及 S19 格式的应用程序下载，并对诊断服务执行过程中出现的否定响应给出具体否定码的解析提示，通过反复测试与修改，最终形成相对友好的下载器界面，见图 4.6。

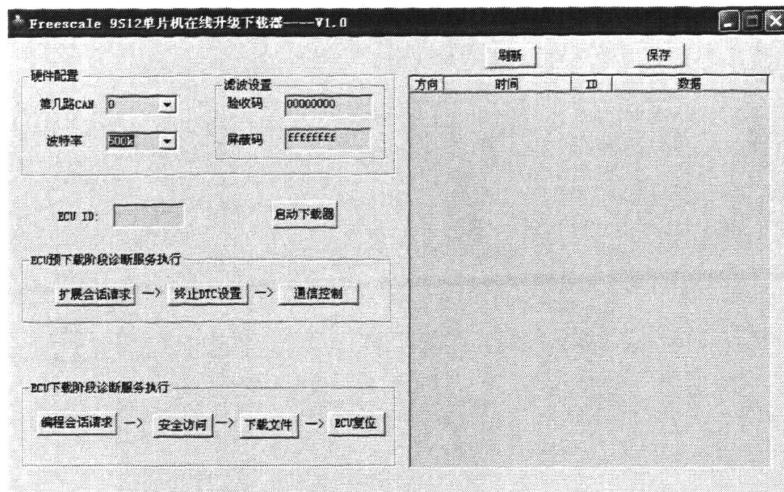


图 4.6 下载器诊断服务实施

#### 4.4 本章小结

综上所述,本章针对基于车载 CAN 网络 ECU 在线升级下载过程需符合诊断协议 ISO15765, 实现基于 ISO15765 的下载流程, 对 S19 文件设计相关算法对其进行处理, 将文件内容处理成 CAN 报文的形式通过 CAN 总线发送给 ECU, 实现了通过 CAN 网络进行程序在线下载的下载器软件设计。

## 第五章 诊断上位机测试

在第三章和第四章对常规诊断模块与下载诊断模块的设计完成之后，本章主要的工作是对所设计两个模块的测试。首先根据诊断网络结构搭建诊断测试平台，然后对常规诊断模块与下载模块分别进行测试。对常规诊断模块的测试主要测试诊断上位机的诊断服务实施是否符合车载 CAN 网络的国际诊断标准 ISO15765，并结合 CANoe 所生成的诊断上位机来完成对比测试，验证所开发诊断上位机的有效性。下载模块的测试主要针对所设计的上位机预编程阶段诊断服务实施的测试，以及 S19 文件通过 CAN 线下载的功能测试。

### 5.1 诊断测试平台搭建

根据第二章图 2.18 所设计诊断系统结构，设计诊断网络测试平台的网络结构如图 5.1 所示，采用诊断设备与被诊断网络 ECU 进行直接相连的诊断网络结构，报文采集工具 USBCANII 通过 USB 口与 PC 端相连，与基于 VC 平台开发的诊断软件作为诊断设备节点。将 CANoe 接入网络可作为整个诊断网络的检测设备，支持 CAN 通信的仪表节点、BCM 节点倒车雷达节点通过 CAN 线连接组成被诊断网络，并接入 12V 电源为被诊断 ECU 供电。将 USBCANII 的 CAN1 口连接到被诊断网络组成诊断系统。

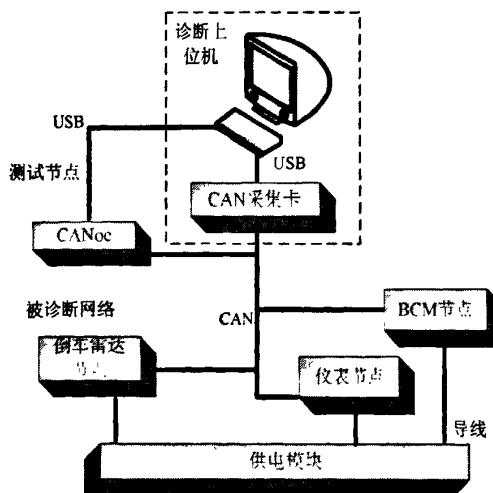


图 5.1 诊断测试网络结构

对诊断测试系统平台中所设计的主要节点进行说明如下：

诊断设备节点：所开发的基于 VC 软件平台与 USBCANII 硬件接口的诊断上位机，所开发的诊断设备通过 CAN 口接入网络后，可对 CAN 网络执行基于 ISO15765 的诊断服务实施。

仪表节点：被诊断仪表节点支持的基于 ISO15765 的诊断服务有：诊断会话请求服务（10）、ECU 复位（11）、清除故障码（14）、读取故障码（19）、读取数据（22）、安全访问（27）、通信控制（28）、写入数据（2E）、诊断仪在线（3E）、控制 DTC 设置（85）等。可作为被测试诊断上位机的诊断响应节点与诊断上位机进行诊断通信。同时，支持基于 CAN 线的程序在线下载功能。

CANoe 节点：CANoe 作为功能强的网络设计仿真及其诊断测试工具，接入网络中导入 CANdelastudio 生成的诊断库文件后，可作为诊断上位机对 ECU 进行诊断。同时，可作为通信网络的检测工具，实时检测通信网络上的通信报文。因此利用 CANoe 作为诊断通信网络的检测工具具有其可靠性与合理性。

倒车雷达节点与 BCM 节点：此两个节点连接到网络，均可支持基于所开发诊断上位机的基于 CAN 线的程序在线下载功能。

按照上图诊断网络结构连接实物如下，USBCANII 通过 USB 口接到 PC 端作为诊断上位机，CANoe 也通过 USB 口接到 PC 端作为网络的测试节点。依次连接好每个网络节点。第一次使用所开发的诊断设备时，应安装 USBCANII 硬件驱动，以及所开发的软件，然后启动诊断设备与诊断软件。启动 CANoe，给 ECU 供电，则整个诊断网络测试平台搭建成功。

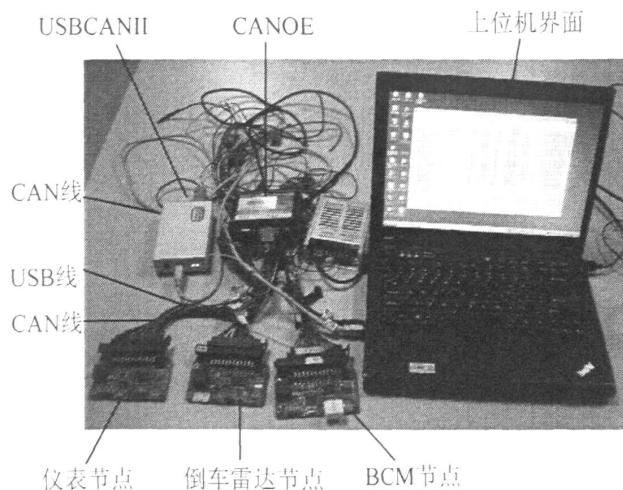


图 5.2 诊断系统测试平台

## 5.2 上位机常规诊断模块测试

对诊断测试平台搭建好后，首先对常规诊断模块进行测试，主要包含诊断协议应用层实现的测试、网络层协议的测试，以及各项诊断服务的实施测试。应用层的测试目的是看每项诊断诊断服务的 SID、其后的子服务或参数是否符合 ISO14229-1 或 ISO15765-3； 网络层的测试主要测试诊断报文是否按照 ISO15765-2 的数据传输规则进行打包传输； 诊断服务的执行测试主要测试诊断服务报文能否正确的被发送或实施。在此测试中，选取支持诊断功能的仪表 ECU 经行诊断，此 ECU 内部有 11 个故障码。其中，其诊断的物理地址 ID 为 0x766，对此仪表 ECU 进行的诊断服务测试有：诊断仪在线、诊断会话请求、安全访问、读故障码、清除故障码。

测试原理：对被测 ECU 进行安全解锁，需让 ECU 进入非默认模式，且 ECU 进入非默认模式后，需启动诊断仪在线服务，以 3s 为周期循环发送此报文以维持 ECU 处于非默认模式。对 ECU 进行安全等级为 1 的安全访问请求，发送 (27 01)，得到种子后，调用安全算法动态链接库并运算得到密钥值，发送给 ECU，ECU 得以解锁。执行“读故障码”服务，分别执行 01 与 02 子服务，分别读取故障码的数量与具体故障码的值。

测试对象：常规诊断模块上位机软件

测试项目：诊断仪在线、诊断会话请求、安全访问、读故障码、清除故障码

测试工具：支持诊断的仪表 ECU 节点，车载电源（12V），Vector CANoe 7.2，计算机（windows XP 系统）

测试步骤：

- 1、打开电源，打开所开发的诊断上位机
- 2、选择诊断上位机通道，选择通信波特率
- 3、在测试设备界面中输入本诊断报文 ID 为 0x766
- 4、双击诊断服务请求页面的“诊断仪在线”
- 5、双击诊断服务请求页面的“扩展模式请求”，执行 03 子服务
- 6、执行“安全访问”，分别执行 01 与 02 子服务
- 7、执行“读故障码”服务，分别执行 01 与 02 子服务
- 8、执行“清除故障码”服务，执行“清除所有 DTC”子服务
- 9、从报文显示与分析模块记录诊断通信报文
- 10、打开 CANoe，导入 CANdelastudio 生成的诊断库文件
- 11、依次执行上述诊断服务，在 Trace 窗口中记录诊断通信报文

## 12、对所开发诊断上位机与 CANoe 下的诊断通信报文进行对比分析

所开发的诊断上位机的诊断服务实施的诊断通信报文见图 5.3, CANoe 下导入诊断库后执行的诊断服务的诊断通信报文见图 5.4。

配置	诊断服务	发送报文	时间	方向	寻址类型	帧类型	诊断服务	ID	长度	数据
诊断会话控制(10)			2011/05/25 23:22:45	发送	物理寻址	单帧	诊断仪在线	00000766	08	02 3e 8d 55 55 55 55 55
(01)默认模式			2011/05/25 23:22:45	接收	物理寻址	单帧	应答	00000765	08	02 7a 00 00 00 00 00
(02)编程模式			2011/05/25 23:22:48	发送	物理寻址	单帧	诊断会话请求	00000766	08	02 10 03 55 55 55 55 55
(03)扩展模式			2011/05/25 23:22:48	接收	物理寻址	单帧	用	00000765	08	04 50 03 00 32 13 88 00
ECU定位(11)			2011/05/25 23:22:48	发送	物理寻址	单帧	诊断仪在线	00000766	08	02 3e 8d 00 00 00 00 00
消除故障诊断码(14)			2011/05/25 23:22:50	接收	物理寻址	单帧	安全访问	00000765	08	02 27 01 AA 55 00 00 00
(01)排放系统DTC			2011/05/25 23:22:50	发送	物理寻址	单帧	算法	00000766	08	04 27 02 00 00 00 00 00
动力系统DTC			2011/05/25 23:22:50	接收	物理寻址	单帧	调用	00000765	08	04 27 03 00 00 00 00 00
底盘系统DTC			2011/05/25 23:22:50	发送	物理寻址	单帧	诊断仪在线	00000766	08	02 3e 8d 00 00 00 00 00
车身系统DTC			2011/05/25 23:22:51	接收	物理寻址	单帧	诊断仪在线	00000765	08	03 7f 27 10 00 00 00 00
网络系统DTC			2011/05/25 23:22:52	发送	物理寻址	单帧	故障码请求	00000766	08	03 19 01 ff 55 55 55 55
全部系统DTC			2011/05/25 23:22:53	接收	物理寻址	单帧	故障码响应	00000765	08	04 59 01 7b 01 00 00 00
读取故障诊断码(19)			2011/05/25 23:22:54	发送	物理寻址	单帧	诊断仪在线	00000766	08	02 3e 8d 00 00 00 00 00
(01)状态故障码匹配DTC数目			2011/05/25 23:22:54	接收	物理寻址	单帧	故障码请求	00000765	08	03 19 02 ff 55 55 55 55
(02)状态故障码匹配DTC列表			2011/05/25 23:22:54	发送	物理寻址	单帧	故障码响应	00000766	08	04 24 59 01 7b c1 00 87
(03)DTC快照记录			2011/05/25 23:22:54	接收	物理寻址	单帧	诊断仪在线	00000765	08	30 90 00 55 55 55 55 55
(04)DTC快照数量			2011/05/25 23:22:54	发送	物理寻址	单帧	故障码请求	00000766	08	21 87 c1 00 88 87 c1 00
(05)DTC相关扩展数据			2011/05/25 23:22:54	接收	物理寻址	单帧	故障码响应	00000765	08	21 91 87 c1 00 92 87 c1
(06)读取所有DTC			2011/05/25 23:22:54	发送	物理寻址	单帧	第一帧	00000766	08	23 00 93 87 c1 00 94 87
读取数据(22)			2011/05/25 23:22:54	接收	物理寻址	单帧	故障码请求	00000765	08	24 c1 00 95 87 c1 00 96
由地址读取内存(23)			2011/05/25 23:22:55	发送	物理寻址	单帧	故障码响应	00000766	08	25 01 24 31 11 02 24 00
读scaling数据(24)			2011/05/25 23:22:55	接收	物理寻址	单帧	诊断仪在线	00000765	08	02 3e 8d 00 00 00 00 00
安全访问(27)			2011/05/25 23:22:55	发送	物理寻址	单帧	清除故障码请求	00000766	08	04 14 ff ff ff 55 55 55
(01)请求种子			2011/05/25 23:22:55	接收	物理寻址	单帧	清除故障码响应	00000765	08	01 54 00 00 00 00 00 00
(03)读取种子			2011/05/25 23:22:58	发送	物理寻址	单帧				

图 5.3 所开发诊断上位机界面

New_ECU_1 - Diagnostics Console		Trace						
		Time	Chn	ID	Name	Dir	DLC	Data
		11.65...	1	7...	Tester Present...	Tx	2	7E 00
		12.65...	1	766	<OTP>	Tx	2	02 10 03 [00 00 00 00 00]
		12.65...	1	1...	Extended Dia...	Tx	2	10 03
		12.65...	1	7A6	<OTP>	Rx	6	06 50 03 00 32 13 88 [00]
		12.65...	1	5...	Extended Dia...	Rx	6	10 03 00 32 13 88
		13.49...	1	766	<OTP>	Tx	2	02 27 01 [00 00 00 00 00]
		13.49...	1	2...	Seed Level #...	Tx	2	27 01
		13.49...	1	7A6	<OTP>	Rx	4	04 67 01 AA 55 [00 00 00]
		13.49...	1	5...	Seed Level #...	Rx	4	67 01 AA 55
		14.24...	1	766	<OTP>	Tx	4	04 27 02 55 AA [00 00 00]
		14.24...	1	2...	Key Level #1...	Tx	4	27 02 55 AA
		14.24...	1	7A6	<OTP>	Rx	2	02 67 02 [00 00 00 00 00]
		14.24...	1	5...	Key Level #1...	Rx	2	67 02
		15.09...	1	766	<OTP>	Tx	3	03 19 01 FF [00 00 00 00]
		15.09...	1	1...	Fault Memory...	Tx	3	19 01 FF
		15.09...	1	7A6	<OTP>	Rx	6	06 59 01 7B 01 00 0B [00]
		15.09...	1	5...	Fault Memory...	Rx	6	59 01 7B 01 00 0B
		16.18...	1	766	<OTP>	Tx	3	03 19 02 FF [00 00 00 00]
		16.18...	1	1...	Fault Memory...	Tx	3	19 02 FF
		16.18...	1	7A6	<OTP>	Rx	47	10 2F 59 02 7B C1 00 87
		16.18...	1	766	<OTP>	Tx	8	30 00 00 [00 00 00 00 00]
		16.18...	1	7A6	<OTP>	Rx	8	21 87 C1 00 88 87 C1 00
		16.18...	1	7A6	<OTP>	Rx	8	22 91 87 C1 00 92 87 C1
		16.18...	1	7A6	<OTP>	Rx	8	23 00 93 87 C1 00 94 87
		16.18...	1	7A6	<OTP>	Rx	8	24 C1 00 95 87 C1 00 96
		16.18...	1	7A6	<OTP>	Rx	8	25 87 C1 00 97 87 31 11
		16.18...	1	7A6	<OTP>	Rx	8	26 01 24 31 11 02 24 00

图 5.4 CANoe 下的诊断上位机界面

### 测试结果分析:

由上述所开发的诊断上位机的报文显示与分析模块来看, 对应用层而言, 实现了诊断服务基于 ISO14229-1 与 ISO15765-3 的执行, 包括安全算法过程, 读取、清除故障码等服务; 网络层协议而言, 实现了基于 ISO15765-2 的诊断帧传输, 对单

帧数据的传输进行了网络层的封装，对超过 7 个数据的数据传输，通过多帧数据的打包算法，以及通过流控帧以多帧的机制进行传输。并将诊断报文封装成 CAN 报文的形式进行传输。实现了对车载 CAN 网络 ECU 进行基于 ISO15765 体系结构的诊断。

### 5.3 上位机下载诊断模块测试

对下载模块进行测试时，平台的搭建与 5.1 节中平台的搭建类似，只是 ECU 对象不同，诊对某款特定车用 ECU 芯片进行下载操作。现对飞思卡尔单片机生成的 S19 文件进行下载测试，测试描述如下：

测试案例：飞思卡尔 S19 文件程序经过 CAN 线的下载测试

测试原理：首先，在文件下载之前，上位机对 ECU 进行预编程阶段及编程阶段的诊断服务实施。然后，上位机软件对 S19 文件进行处理，将不需发给 ECU 的信息处理掉，将文件以行为单位进行发送，并且将每行数据处理成 CAN 报文帧的形式进行发送。ECU 每接收完一行变进行响应，上位机接收到响应报文后，继续发下一行数据，直到所有数据被发送完毕，所下载程序被 ECU 成功启动，说明下载成功。

测试对象：下载器上位机软件

测试模块：下载诊断模块

测试项目：下载诊断测试

测试工具：支持下载诊断功能的 CAN 网络倒车雷达 ECU 节点，车载电源（12V），Vector CANoe 7.2，计算机（windows XP 系统）

测试步骤：

- 1、打开电源，打开下载系统和监测系统软件
- 2、选择 CANoe 通道，配置 CANoe 通信波特率
- 3、选择诊断上位机通道，选择通信波特率
- 4、在测试设备界面中输入本诊断报文 ID 为 0x7df
- 5、进行预编程阶段的诊断服务实施
- 6、进行编程阶段的诊断服务实施
- 7、下载 S19 文件
- 8、对 ECU 进行复位

如图 5.5 所示，对 ECU 进行完相关诊断会话后，进行 S19 文件的导入。

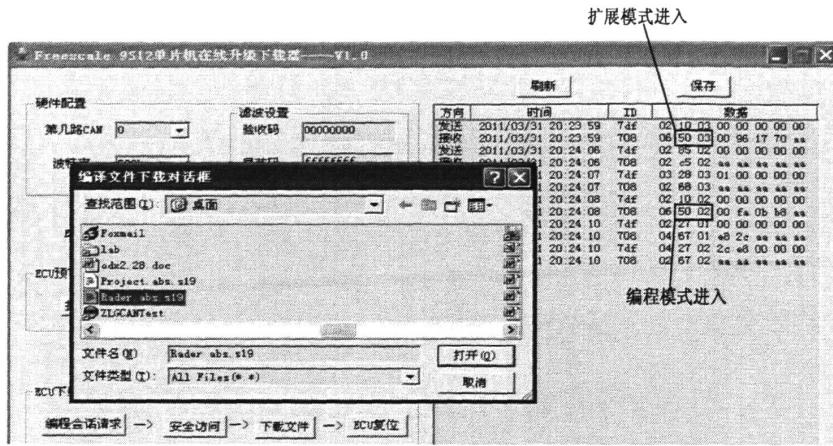


图 5.5 S19 文件在线下载

所导入的 S19 原文件如图 5.6 所示，第一行为文件开始提示行，第二行 S1 表示类型，16 进制 23 表示其后有 35 字节数据，C000 表示地址，其后为所要发送的数据，其中，图中框出的 16 个字符的数据为要下载的其中一小段数据。

This screenshot shows a Windows Notepad window with the title 'Rader.abs.s19 - 记事本'. The content of the window is a raw binary dump of an S19 file. The first few lines of the dump are:

```

S02D 0000453A5C8 007B4EF5C85B9B385323031312D3F
6F6A6563742E616 起始地址 → 须下载的其中16个字符数据
S123C 0000CF05004A8905384A880F83800001C011C502050000040FF

```

The '起始地址' (Start Address) is highlighted with an arrow pointing to the value '6F6A6563742E616'. The text '须下载的其中16个字符数据' (The first 16 bytes of data to be downloaded) is also indicated by an arrow.

图 5.6 所需下载的 S19 原文件

经上位机对 S19 原文件进行解析后以 CAN 报文的形式发送如图 5.11 所示。将不满三字节的地址前加 00 统一为三字节的地址，数据加 1 为 24，地址前加 00 为 00 C0 00，在原文件中的字符型数据经上位机处理为以 8 字节为一帧的 CAN 报文数据，不满 8 字节时末尾以 00 填充。如原文件中框出的 16 个字符数据“01C011C50205000”上位机处理为 8 字节的 CAN 报文数据“01 C0 11 C5 02 05 00 00”进行发送，见图 5.7 所框出的 CAN 报文。

方向	时间	ID	数据
发送	2011/03/31 16:34:40	7af	02 10 03 00 00 00 00 00 01
接收	2011/03/31 16:34:40	708	06 50 03 00 96 17 70 a
发送	2011/03/31 16:34:44	7af	02 85 02 00 00 00 00 00 01
接收	2011/03/31 16:34:44	708	02 c5 02 a a a a a a a a
发送	2011/03/31 16:34:45	7af	03 28 03 01 00 00 00 00 00
接收	2011/03/31 16:34:45	708	02 68 03 a a a a a a a a
发送	2011/03/31 16:34:46	7af	02 10 02 00 00 00 00 00 01
接收	2011/03/31 16:34:46	708	06 50 02 00 f a 08 b a
发送	2011/03/31 16:34:47	7af	02 27 01 00 00 00 00 00 01
接收	2011/03/31 16:34:47	708	04 67 01 a 2 c a a a a a
发送	2011/03/31 16:34:47	7af	04 27 02 2 c a 00 00 00 00
接收	2011/03/31 16:34:47	708	02 67 02 a a a a a a a a
发送	2011/03/31 16:35:13	7af	24 00 c0 00 c f 05 00 4
发送	2011/03/31 16:35:13	7af	88 05 38 4 a 80 68 38 00
发送	2011/03/31 16:35:13	7af	01 c0 11 c5 02 05 00 00
发送	2011/03/31 16:35:13	7af	40 ff 01 01 01 01 01 01 0
发送	2011/03/31 16:35:13	7af	01 01 01 01 00 00 00 00 0
接收	2011/03/31 16:35:13	000	02 76 00 00 00 00 00 00 0
发送	2011/03/31 16:35:13	7af	24 00 c0 20 01 01 01 0
发送	2011/03/31 16:35:13	7af	01 01 01 81 81 81 81 81 8
发送	2011/03/31 16:35:13	7af	81 81 81 81 81 01 01 01 0
发送	2011/03/31 16:35:13	7af	01 01 01 01 01 01 01 01 0
发送	2011/03/31 16:35:13	7af	01 01 01 01 00 00 00 00 0
接收	2011/03/31 16:35:13	000	02 76 00 00 00 00 00 00 0
发送	2011/03/31 16:35:13	7af	24 00 c0 40 01 01 01 0
发送	2011/03/31 16:35:13	7af	ff ff 00 00 00 00 00 00 0
发送	2011/03/31 16:35:13	7af	00 00 00 00 00 00 00 00 01

图 5.7 经下载器处理的 S19 文件

### 测试结果分析：

由测试结果可见，所设计的上位机可执行基于 ISO15765 的预编程阶段及编程阶段的诊断服务实施。原始 S19 格式的程序文件经所设计的上位机处理后转化为标准 CAN 报文的形式进行发送，程序经过 CAN 网络进行在线下载，下载到倒车雷达 ECU 后，程序成功运行。

## 5.4 本章小结

本章通过搭建测试系统平台，对所设计的基于 ISO15765 的诊断上位机进行了测试，测试模块包括常规诊断模块与下载模块。其中，对常规诊断模块的测试包括 5 项诊断服务的实施测试，并与 CANoe 导入 CANdelastudio 所生成的诊断库进行对比测试。结果表明所开发常规诊断上位机的有效性。对下载模块的测试通过下载一个实际程序对所设计的诊断模块进行测试。可实现基于 ISO15765 的诊断上位机通过 CAN 线的下载。从而验证所开发的基于 ISO15765 国际诊断标准的车载 CAN 网络诊断上位机的可行性。

## 第六章 总结与展望

### 6.1 论文研究工作总结

车载网络是近十年逐步兴起的汽车电子技术，CAN 网络作为目前最流行的车载网络技术，国内外汽车生产商针对 CAN 网络的研究开发工作也逐步展开，在国外基于国际诊断标准的 CAN 网络诊断研究相对成熟，在我国的各大汽车厂商及车载 ECU 供应商，基于车载网络国际标准 ISO14229-1 和 ISO15765 的车载网络诊断研究工作尚处于起步阶段。ISO14229-1 协议于 2006 年刚出现，至今只有几年时间，并且协议中的专业术语难以理解，内容也较为复杂，以至于在我国大部分汽车厂商和 ECU 供应商等尚未普遍被理解与应用，这也是我国在对于车载网络诊断的研究工作进度较慢的原因所在，车载网络诊断研究工作的技术突破点在于对国际诊断协议流程的理解与实现。

本文以国家核高基项目和重庆市节能环保汽车专项为背景，研究开发基于 ISO14229-1 与 ISO15765 的车载 CAN 网络国际诊断协议，对国际诊断标准应用层协议 ISO14229-1 以及基于 CAN 网络的 ISO15765-3 进行详细解析与设计，并研究其基于 ISO15765-2 的 CAN 网络层诊断协议。本论文的主要研究开发工作和取得的结论如下：

- (1) 对目前汽车诊断的研究背景和意义进行了阐述，对汽车诊断方法及车载网络诊断协议等相关技术进行了总结，在此基础上提出了本文的研究工作。
- (2) 分析被诊断车载网络对象结构及特征，设计车载网络的诊断结构，诊断报文的地址格式、确定不同诊断服务的寻址方式类型等。
- (3) 实现基于 ISO14229-1 与 ISO15765-3 车载 CAN 网络诊断协议的应用层设计，将 5 大类诊断服务的 SID 以及其参数按照国际标准进行设计。
- (4) 实现基于 CAN 网络 ISO15765-2 的网络层诊断协议，对应用层数据进行基于 ISO15765-2 的封装，实现数据的单帧与多帧传输。
- (5) 对普通诊断模块，利用 VC 软件开发平台和 USBCANII 硬件采集卡，实现了基于国际诊断标准低成本的诊断上位机，所开发的诊断上位机可实现基于 ISO15765 的车载 CAN 网络的诊断。可脱离成本较高的国外诊断系列工具，对支持国际诊断标准的 ECU 进行诊断。
- (6) 对下载诊断模块，在国际诊断标准中将上传下载进行了详细的规定，利用 VC 软件开发平台和 USBCANII 硬件采集卡，实现了基于国际诊断标准的低成

本、方便灵活的基于 CAN 线的下载器，装入车内的 ECU 不需拆卸下来，利用所开发的下载器通过 CAN 线即可实现车载 ECU 的在线下载功能。

## 6.2 论文研究展望

本论文主要研究实现了基于国际诊断标准 ISO15765，并且实现了基于 ISO 15765 的车载诊断上位机，及诊断下载上位机。论文存在以下几个方面可以进行后续扩展和研究。

(1) 论文从诊断协议的应用层到网络层以及到数据链路层实现了车载网络的诊断，对应用层数据都是自行设计的。如果有统一的诊断库文件，则可将诊断上位机设计成基于统一诊断库的诊断设备，对不同 ECU 进行诊断只需导入 ECU 的诊断数据库文件即可。这样不再需要自行设计诊断协议，只需取诊断库中的诊断描述数据即可。

(2) 有关下载器模块的上位机设计，仍需对其进行功能扩展，可实现对多种芯片的设计。另外，其数据传输部分，一次可进行更多数据量的传输，而非单行的诊断数据的传输，以提高数据下载速度。

## 致谢

在本文完成的过程中，我经历了许许多多，有亲人的无私支持，有老师的悉心培育，有同学的热情帮助，还有身边朋友的丝丝关怀等等；借此机会，我要感谢那些一直在我身边支持我帮助我鼓励我的亲朋好友，感谢大家陪伴我一路走过这研究生的三年时光。

本论文是在李锐副教授和程安宇副教授的悉心指导下完成的。在攻读硕士期间，李老师和程老师给了我展现自身价值的机会，并在学习上和生活上都给我了极大的关怀。三年来，本人在两位老师的指导下完成了科研课题的研究开发工作，使我各方面的能力均得到了锻炼和提高，在此谨向两位老师表示衷心地感谢。还要感谢我的父母，感谢你们为我做的一切，让我能安心学习研究，为未来铺垫更好的基础。

最后再次对我的导师李锐副教授送上最真挚的感谢，不忘记时常当面教导我，教育我做人的道理，指导我做学问的思想。在这三年里，不论有多忙多累，李老师从来不曾延迟回复我的邮件。在每一份李老师为我审阅的材料，仍可见李老师详细而精炼的批语，指导我的步伐。同时也感谢程安宇老师在平时的科研过程中对我的关心和指导。

感谢我们诙谐幽默的罗洪平老师，在科研之余，为我们带来愉悦，让我体验实验室温暖的氛围。感谢实验室的所有师弟师妹。

## 参考文献

- [1]怀琳. 关于汽车 CAN 总线技术的研究[J]. 轿车情报. 2008: 162-163
- [2]NICOLAS NAVET, YEQIONG SONG, FRANÇOISE SIMONOT-LION, AND CÉDRIC WILWERT. Trends in Automotive Communication Systems[C]. IEEE, VOL. 93, NO. 6, JUNE 2005
- [3]张宏, 詹德凯, 林长加. 基于 CAN 总线的汽车故障诊断系统研究与设计[J]. 汽车工程. 2008.30(10): 934
- [4]郭锐. 车载 LIN 网络诊断技术的研究与实现[D]. 重庆邮电大学硕士论文, 2007
- [5]田晓川, 王励明等. 2 种汽车诊断协议对比浅析[J]. 汽车电器. 2008(12).6-8
- [6]ISO 11898-1:Controller area network (CAN) Part 1[S]. 2003
- [7]ISO 14229-1:Road vehicles-Unified diagnostic services[S]. 2005:ix
- [8]王立东, 车载监测诊断 GUI 的设计与实现[D]. 电子科技大学硕士论文. 2008:1
- [9]马英, 阴晓峰, 张德旺. 基于 CAN 的汽车电控系统故障诊断技术[C]. 2008 中国汽车工程学会年会论文集. 2008.SAE-C2008E417
- [10]张毅华, 张剑锋, 沈延. 汽车诊断仪平台整合[J]. 上海汽车. 2010:29-32
- [11]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[J].2006(9):40-43
- [12]ISO 14229-1:Road vehicles-Unified diagnostic services[S]. 2005:1-10
- [13]ISO 15765-2: Diagnostics on Controller Area Networks (CAN) -Part 2[S]. 2004:17~22
- [14]ISO 15765-2: Diagnostics on Controller Area Networks (CAN) -Part 2[S]. 2004:28~30
- [15]ISO 14229-1:Road vehicles-Unified diagnostic services[S]. 2005:11
- [16]Requirement Specification.1001-201-001RS01 Diagnostic Communication. CHB-011 Project[S]. 2010:5
- [17]王天军. 汽车电子通信诊断系统的设计与实现[D]. 大连理工大学硕士论文.2008.11:24
- [18]ISO 14229-1:Road vehicles-Unified diagnostic services[S]. 2005
- [19]Michael Blaba, James Rumbaugh 著. 车皓阳, 杨眉译. UML 面向对象建模与设计(第 2 版)[M]. 人民邮电出版社. 2006
- [20]丁志华, 罗峰, 孙泽昌. 基于 CANoe 的汽车故障诊断系统研制[J]. 汽车工程. 2007 29(5): 449-452

- [21]Jittiwut Suwatthikul, Ross McMurran and R. Peter Jones. Automotive Network Diagnostic Systems[C]. IEEE. 2006
- [22]Feng Luo, Mang Mo, Juxiao Chen, Zechang Sun. Fault Diagnosis Systems Development for Fuel Cell Vehicle[C]. IEEE Vehicle Power and Propulsion Conference (VPPC). Harbin, China. 2008
- [23]邵斌等. 混合动力汽车 CAN 网络信号监测与故障诊断系统的开发[J]. 汽车技术. 2009(2):46-49
- [24]USBCAN 数据手册 V2.00
- [25]CAN-bus 通用测试软件及接口函数库使用手册
- [26]杨会, 鲁统利, 王天军. 基于 GMLAN 的汽车诊断通信仿真[J]. 汽车工程. 2010(10): 902-904
- [27]王明玉, 董浩, 王建. 基于 GPRS 的车载故障诊断网关的设计与实现[J]. 汽车工程. 2009(10): 999-1003
- [28]CANdelaStudio 用户使用手册. 2009: 43
- [29]ISO 14229-1:Road vehicles-Unified diagnostic services[S]. 2005
- [30]CAN 总线诊断标准 CAN diagnosis requirements specification[S]. 2009:49
- [31]ISO 14229-1:Road vehicles-Unified diagnostic services[S]. 2005:61-65
- [32]孙鑫, 余安萍著. VC++深入详解[M]. 电子工业出版社. 2006: 723-725
- [33]Renjun Li, Chu Liu, Feng Luo. A Design for Automotive CAN Bus Monitoring System[C]. Vehicle Power and Propulsion Conference (VPPC). Harbin, China. 2008
- [34] <http://blog.csdn.net/finewind/archive/2010/04/14/5483554. aspx>
- [35]ISO 15765-3: Diagnostics on Controller Area Networks (CAN) -Part 3[S]. 2004: 61-63
- [36]ISO 15765-3: Diagnostics on Controller Area Networks (CAN) -Part 3[S]. 2004:64-66

## 攻读硕士学位期间所参与科研工作和所发表论文

### 1、参与的课题和科研工作

- [1] 国家核高基重大专项—“汽车电子控制器嵌入式软件平台研发及产业化”(No.2009ZX01038-002-002-2)。
- [2] 重庆市高技术产业重大产业技术开发项目—“节能环保汽车专项”(No.CSTS2008AA6025)。
- [3] 重庆市自然科学基金项目《汽车 ABS/TCS 分级智能控制方法与关键技术研究》(CSTC,2008BB2407)。
- [4] 重庆市教委科技项目《全路面机动车辆安全制动分级智能控制研究》(KJ080501)。

### 2、所发表学术论文

- [1] Anyu Cheng, Yan Yao, Zhihui Duan, Anjian Zhou and Wei Hong. ECU Loader Design of in-vehicle CAN Network Based on ISO15765[C]. IEEE,ICIST. Nanjing, China.2011.4.

# 基于ISO15765的车载CAN网络上位机诊断软件设计

作者：姚燕

学位授予单位：重庆邮电大学

本文链接：[http://d.wanfangdata.com.cn/Thesis\\_Y1989855.aspx](http://d.wanfangdata.com.cn/Thesis_Y1989855.aspx)