摘 要

车载信息系统是汽车上电子设备的综合，它主要包括汽车信息显示和故障诊断两大类。汽车信息显示可以在汽车行驶的过程中，通过车载的显示屏显示电子地图、汽车所在位置。汽车故障诊断系统既可以通过车载显示屏显示汽车各个关键部位的实时状态，如轮胎的温度压力值、发动机是否正常、水温是否正常等等，同时将实时数据经车载网络传输到故障诊断仪器，完成故障诊断的任务。

故障诊断系统的基础是汽车内部的车载网络。目前，现有的具备故障诊断功能的汽车的车载网络包括车身网络 CAN 网(Controller Area Network)和故障诊断网络 K 线、L 线。CAN 网和 K 线、L 线是完全独立的两个网络。CAN 网的传输速率达到 256Kbps，属于高速网，是专用于连接发动机、悬架、牵引控制等高速设备，可以保证数据采集的实时性。K 线、L 线是 KWP2000（Keyword Protocol 2000）即故障诊断专用通讯协议的物理层，专门用于收集汽车各个部件的故障信息，然后向上送给 KWP2000 的应用层。

本文论述的车载故障诊断系统，使用车身网络 CAN 网取代故障诊断专用的 K 线、L 线网络，将两种独立的网络合而为一，进一步简化车内网络的复杂程度。以车身网络 CAN 传输各个部件的实时数据，通过 CAN/USB 网关实现 CAN 数据帧同 USB 数据帧的转换，然后经过 USB 将 CAN 网的数据传到笔记本上的故障诊断程序，程序使用国际上通用的故障诊断代码，保证了系统的通用性。

故障诊断程序对经由 USB 传输过来的故障代码进行分析诊断，确定故障的原因和位置，使维修人员能够快速方便的进行修理。将此技术应用于汽车故障诊断，可以大大降低汽车维修的难度和费用。

关键词：车载信息系统，故障诊断，USB，CAN

Abstract

Carborne information system is the integration of all the electronic equipments of the car, it mainly consists of carborne information display system and malfunction diagnosis system. The carborne information display system can display the electronical map and the position of the car etc, on the LCD when the car is running. The malfunction diagnosis system can display the real-time status of all the key parts on LCD, such as the pressure of the tires, whether the engine is ok, whether the temperature is ok etc. At the same time, it can also transmit the data through the carborne network, using the special diagnose devices examine the car through the interface of the car.

The base of the malfunction diagnosis system is carborne network in the car. Now, in the existing cars that have the malfunction diagnosis function, the familiar networks are car body network CAN (Controller Area Network) and malfunction diagnosis network K line and L line. CAN network and K and L line network are two absolute network. The CAN network can transmit at a speed of 256Kbps, it belongs to the high-speed network and is special for connecting the high-speed equipments such as engine, draw controller etc. It can ensure the real-time feature of the data gathering. K line and L line are the physics layer of the KWP2000 malfunction diagnose special communication protocol, it is used specially to gather the malfunction information of all parts, then sends these information to the application layer of KWP2000.

The malfunction diagnosis system that discussed in the paper uses car body network instead of the K and L line network, it unites the two abstract network to one, predigests the inner network more. It uses car body network transmit the real-time data of all parts, uses the CAN/USB gateway realize the conversion between CAN data frames and USB data frames, then transmit the CAN data to malfunction-diagnosis program that in the portable computer through USB. The program uses the universal malfunction diagnosis codes in the world, it can ensure the currency of the system.

The malfunction-diagnosis program in the portable computer is used to

analyze the standard malfunction codes to ensure the reason and location of the malfunction. It can make the maintenance workers maintain the cars quickly and conveniently. Using this technology in the malfunction diagnose of the cars can reduce the difficulty and expense of automobile reparation greatly.

Key words: Carborne Information System, Malfunction diagnosing, USB, CAN

第 1 章 绪论

1．1 课题的提出和研究背景

从 2000 年开始，中国的汽车工业进入了一个高速发展阶段，如此同时，轿车作为私家车的主体，迅速进入千家万户。随着汽车的普及，大量的汽车已经或正在进入维修期。各种各样品牌的国产和进口车，其内部构造有着很大的区别，如果无法确定故障的准确位置，维修工作将是十分困难的。

针对这种情况，发达国家早在上世纪九十年代便开始基于信息显示、故障诊断的车载信息系统的研究。目前，已经形成了一套以汽车传感器、车载网络、KWP2000[1]为基础的车载故障诊断系统。整个故障诊断系统网络独立与 CAN 网络，采用 KWP2000 协议为故障诊断定义的 K 线和 L 线[2]连接各个汽车部件，通过 K 线接口同外部的专用的故障诊断仪器相连。

国内近年来也开始这方面的研究，具备车载故障诊断功能的汽车[3]，使用的故障诊断设备，也都采用 KWP2000 作为故障诊断协议。本文提出的基于 USB 的车载信息系统，是以车身 CAN[4]网络（在 CAN 节点下可以包含 LIN[5]网络）为基础的车载故障诊断系统。使用国际通用的故障代码，使用 CAN 网络的物理层和数据链路层取代 K 线和 L 线构成故障诊断网络，来连接各个汽车部件。系统的对外接口并未采用 K 线接口，不使用专用的故障诊断仪器，而是使用 PC 机上通用的 USB 接口，使用配备专门故障诊断程序的 PC 机或笔记本代替专用的故障诊断仪器。这种设计的目的在于，可以使该系统尽量少的受制于国外的技术壁垒，易于开发自己的故障诊断仪器，有利于国内的车载故障诊断系统的快速发展。

1．2 车载信息系统

汽车的出现是作为功能单一的交通工具。随着科学技术的发展和人们对汽车的安全性、舒适性的要求的提高，汽车内的电子设备，如：ABS、安全气囊、电控门窗等等越来越多。这些设备刚出现时是作为一个单独的部件增加到汽车上去的，它们之间似乎并没有太多的联系。但是随着网络技术和计算机技术的不断发展，不同设备之间建立起了复杂的关联，于是人们便将各种设备的操作部件集中到一个平台上，由一个统一的操作平台来完成对不同设备的操作，在此基础上便逐渐产生了车载信息系统。现在的车载信息系统的功能已经远远超出了单一的控制，实现了将整台汽车的所有部件，包括发动机、轮胎等等连接成一个网络，在信息平台上测控汽车不同部件的工作状态。

1．3 车载信息系统国内外发展的现状

车载信息系统是一种新型的汽车电子设备。目前，国外一些知名的汽车生产厂商在这方面已经做了多年的研究，形成了比较成熟的产品。车载信息系统，主要包括汽车信息显示和故障诊断两大类。汽车信息显示通过车载显示屏显示各种图形、图像信息，如：电子地图、汽车位置等。汽车故障诊断系统通过汽车内部的网络传输各个部件的故障信息，通过故障诊断接口对外部传输数据，专用的故障诊断仪器通过这个接口获得数据，对汽车整车进行检测，找出发生故障的位置和原因，并通过特定的符号显示出来告诉修理人员如何进行快速、正确的修理。

汽车故障诊断系统[6][7][8]主要由两部分组成：数据采集系统和数据分析系统。数据采集系统主要完成对车辆运行过程中各个工作部件的工作状态数据的采集，数据采集是通过各种各样的汽车传感器实现的；数据分析系统包括车载部分和车外部分（本文论述的重点）即前面提到的专用故障诊断仪器。故障诊断仪器通过汽车对外部的接口获得由传感器给出的各个部件工作状态的参数，然后使用专用的诊断程序对采集到的数据进行详细处理，以得到部件的故障位置、原因和解决方案，程序需要完成大量的数据处理工作。

国内在车载信息系统方面，是近几年才开始的。目前主要的研究集中在车载信息显示，如3G，在这方面的研究开展的比较广泛。而在汽车故障诊断系统方面却发展缓慢，远远落后与国外。

1．4 本文主要的研究内容及技术路线

本文论述的基于USB的车载信息系统重点着重于故障诊断系统，该系统主要由车上的数据采集部分和以笔记本电脑为平台的故障数据分析部分组成。本文研究的内容包括：

1．在汽车（以轿车为对象）上构建CAN/USB网关，使得通过汽车传感器采集到的数据可以经过车载网络汇总到CAN/USB网关。

2．通过USB接口，设计在PC机（以笔记本为对象）上运行符合通用故障诊断代码的故障诊断程序，包括数据获得、数据处理、结果显示等。

USB（Universal Serial Bus）[9][10][11]是为了适应计算机的广泛应用、人们对串行通信提出的更高要求，而开发出来的一种兼容高速和低速，为广大用户提供可共享、可扩充、使用方便的串行总线。USB[12][13]具有快速、双向、同步传输、支持热拔插、易于与PC机接口等特点。基于以上优点，使得其在接口方面的使用极其方便。USB可以连接多个不同的外部设备，最高传输率可达12Mb/s，这使得PC机与车载信息系统的结合成为可能。

车载故障诊断系统的构成如图1－1所示。

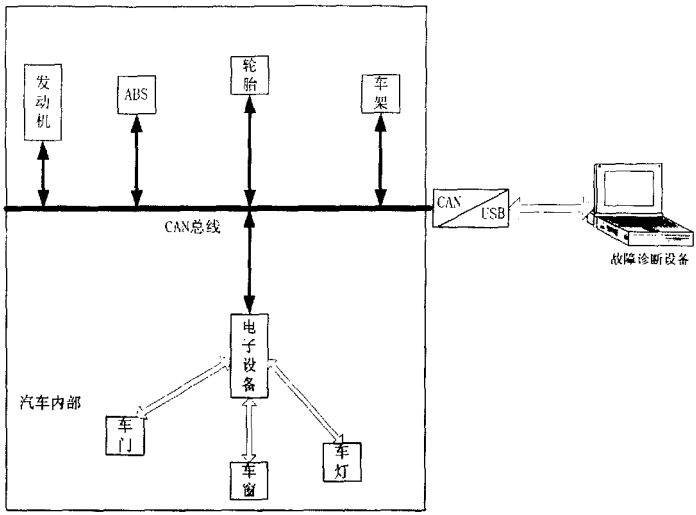

图 1－1 结构原理图

在现代汽车的内部所有的部件，包括车门、车窗、车灯、发动机、ABS、轮胎、刹车系统、转向系统等等都通过总线形式连接在一起，通过一个共同的 CAN/USB 网关将数据传输到汽车的外部。根据汽车上不同部件对操作的实时性要求的不同，连接的总线可以分为 CAN 总线和 LIN 总线两种。其中，CAN 总线用于连接对实时性要求很高的部件，例如：发动机、刹车系统等；LIN 总线用于连接对实时性要求不是很高的部件，例如：车门、车窗等。CAN 和 LIN 总线采集到不同部件的数据后，统统汇总到 CAN/USB 网关上，然后通过 USB 接口同外部设备通信。

当需要对一辆汽车进行故障诊断时，将一台装有故障诊断程序的笔记本电脑通过 USB 接口同汽车内部的总线相连接，便可以获得各个部件的数据，然后由故障诊断程序对收集到的数据进行分行，以确定汽车的故障出在哪里，并给出适当的解决方案。

在车载故障诊断系统中，采用 USB[14]接口的目的在于：CAN 网络代表了汽车网络的发展趋势[15]，通过 CAN/USB 网关可将汽车各个部件的信息传送到 PC 机，利用 PC 机所具有的诸如高速信息处理、丰富的软件支持、良好的网络功能等特点，而不必专门为车载信息系统开发专门的故障诊断设备，进而可以降低成本、提高工作效率。PC 机强大的网络功能，提供了在现场无法排除故障的情况下，将故障数据传送到故障维修数据中心交由专家远程诊断的能力。

基于 USB 的车载故障诊断系统分为数据采集部分和数据分析部分两部分进行设计。数据采集部分以车身 CAN 网络为基础，数据采集部分的构成涉及

到汽车传感器、车内 CAN 网络和 CAN/USB 网关等硬件方面和数据的传输格式、操作系统等软件方面。数据分析部分的硬件平台是一台笔记本电脑，软件平台是基于 Win2000＋Delphi7.0[116]的可视化界面。笔记本电脑通过 USB 接口获得整车的各个工作部件的状态参数，并提交诊断程序处理，最终给出故障位置和处理方法。

由于整个系统的构成比较复杂，本文以汽车轮胎压力检测为研究对象，构建数据采集平台；以车身 CAN 网络构建网络平台；采用国际通用的故障诊断代码，配合 Windows 平台下的 Delphi 构建软件平台[117]。系统设计采取先硬件后软件的技术路线，针对以上三个对象开展工作。

1．5 课题的研究意义

表 1－1 汽车总线协议发展概况

Hersteller - Modell	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008
Audi/VW	ISO 9141	ISO 9141	ISO 9141	ISO 9141	ISO 9141	ISO 9141	ISO9141 KWP2000	ISO9141 KWP2000	ISO9141 KWP2000	ISO9141 KWP2000 CAN	ISO9141 KWP2000 CAN	ISO9141 KWP2000 CAN	CAN
Bentley	ISO 9141	ISO 9141	ISO 9141	ISO 9141	ISO 9141	ISO 9141	ISO 9141	ISO 9141	ISO 9141 (50%) KWP2000 (50%)	ISO 9141 (35%) KWP2000 (65%)	ISO 9141 (20%) KWP2000 (80%)	CAN	CAN
BMW - Mini	N/A	N/A	N/A	N/A	N/A	N/A	KWP2000	KWP2000	KWP2000	KWP2000			
BMW 3-Serie, 5-Serie, X5, Z3, Z4, Z8, 740i, 740iL, 750iL	ISO 9141/2	ISO 9141/2	ISO 9141/2	ISO 9141/2	ISO 9141/2	ISO 9141/2	ISO 9141/2	ISO 9141/2	ISO 9141/2	ISO 9141/2			
BMW 5-Serie, nächste Generation (E60)	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	KWP2000	KWP2000			
BMW 745i, 745Li, 765Li	N/A	N/A	N/A	N/A	N/A	N/A	KWP2000	KWP2000	KWP2000	KWP2000			
Chrysler	ISO 9141-2 (100%)	ISO 9141-2 (100%)	ISO 9141-2 (95%) J1850-VPW (5%)	ISO 9141-2 (85%) J1850-VPW (15%)	ISO 9141-2 (75%) J1850-VPW (25%)	ISO 9141-2 (35%) J1850-VPW (65%)	ISO 9141-2 (15%) J1850-VPW (85%)	ISO 9141-2 (5%) J1850-VPW (95%)	CAN (5%) J1850-VPW (95%)	CAN (15%) J1850-VPW (85%)	CAN (35%) J1850-VPW (65%)	CAN (85%) J1850-VPW (15%)	CAN (100%)

表 1-1 是由奥迪、宝马、克莱斯勒等多家大汽车公司共同制定的汽车总线协议的发展目标。从表中可以看出，到目前 2005 年为止，KWP2000 和 CAN 协议在汽车上是共存的。而到 2008 年，所有的汽车厂商都将汽车内部总线统一到 CAN 协议上来，用 CAN 协议取代专用的汽车故障诊断专用通讯协议 KWP2000，即使用一个统一通用的协议来代替各种专用协议，用 CAN 的数据帧格式来完成故障诊断工作，简化汽车内部的总线种类，最终以一种总线形式承担起汽车内部连接各种部件、传输数据、故障诊断等所有的功能。

由此表可以看出，本文论述的车载故障诊断系统没有采用汽车故障诊断专用的 K 线、L 线网络和专用的 K 线接口，而是采用通用的车身 CAN 网络取代专用的 K 线、L 线网络，并且用 CAN/USB 网关将汽车信息统一到 PC 机上的

设计是符合汽车总线发展趋势的，也体现了本系统的创新点所在。

车载故障诊断系统在国内的研究远远落后于国外先进水平，国内厂商生产的汽车基本上不具备故障诊断的能力，使得汽车的维修工作和成本十分庞大。研究本系统意义在于提高国产汽车的智能化程度，减少汽车维修的难度和费用。

第 2 章　传统故障诊断系统的基本构架

2．1 系统构成综述

一个典型的、使用 KWP2000 故障诊断专用通讯协议的车载信息显示、故障诊断系统，可以分为数据采集部分、数据传输部分、数据接口部分、故障诊断专用仪器部分。

数据采集部分由各种各样的汽车传感器组成，数据传输部分是专用的车载网络，数据接口是同车载网络配套的标准接口，故障诊断专用仪器接收到汽车传感器的数据对故障进行诊断。

2．2 数据采集

2．2．1 汽车传感器

近年来从半导体集成电路技术发展而来的微电子机械系统(MEMS) 技术日渐成熟，利用这一技术制作各种能敏感和检测力学量、磁学量、热学量、化学量和生物量的微型传感器[18]，这些传感器的体积和能耗小，可实现许多全新的功能，便于大批量和高精度生产，单件成本低，易构成大规模和多功能阵列，非常适合在汽车上应用。

汽车传感器作为汽车电子控制系统的信息源，是汽车部件数据采集的关键部件。汽车传感器对温度、压力、位置、转速、加速度和振动等各种信息进行实时、准确的测量和控制。当前，一辆国内普通家用轿车上大约安装了近百个传感器，而豪华轿车上的传感器数量 200 多只。各式各样的汽车传感器构成了车载信息系统的基础。汽车上常用的传感器有发动机传感器、胎压监测传感器、一氧化碳传感器、NOx 净化传感器、车辆防盗倾斜传感器、汽车油压传感器等等。

2．2．2 实时性要求部件

高实时性要求部分主要是指发动机控制器、供油系统、供气系统，以及各种同安全相关的设备。发动机在工作的过程中，要求空气的进气量和燃油的喷射量的比值，即空燃比要达到一定的比值，才能保证油料燃烧充分、提供出最大的功率、排放出来的废气最少。发动机的工况不同，空燃比就不同，因此就要求将发动机控制器、供油系统、供气系统，及其外围器件反映出来的各种数据实时的汇总到故障诊断系统。各种同安全相关的设备，主要包括

轮胎压力检测 TPMS、轮胎防暴死系统 ABS 等等。这些设备直接关系到汽车和车上人员的安全，所以其反映出来的数据必须实时的传输给故障诊断系统做出判断。

2．2．3 其它部件

其它部件主要包括车门、车窗、车灯，后视镜等部件。这些部件的操作时间和指令到来时间之间存在一定的延时，是完全可以容忍的，并不要求严格的实时同步。因此这些部件反映出来的各种数据可以非实时的汇总到信息系统。

2．3 数据传输

数据传输部分是指的连接高实时性要求部件和其它部件的总线。CAN 总线的传输速率高，成本相对 LIN 较高，用于连接实时性要求部件。LIN 总线的传输速率低，成本低，用于连接无实时性要求的部件。K 线、L 线网络专门用于故障诊断。

2．3．1 汽车内使用的几种总线

基于车载网络的连接标准主要有：IDB(Intelligent Transportation System)，LIN(Local Interconnect Network)，CAN(Controller Area Network)，SAEJ1939 等。各种网络连接用于不同层次和目的，为了实现信息的集成和状态监控主要是实现基于 CAN/LIN 的网络信息传输与网络协议转换，同时其网络结构与设计必须满足实时性要求。现代汽车用于构成内部车身网络主要使用 CAN 和 LIN 两种协议，故障诊断使用基于 SAE 的 K 线、L 线网路。

◆CAN 总线[19][20]的数据传输速率是 256Kbps，在车载网络中属于高速网，主要用于发动机、悬架、牵引控制的连网。CAN，即控制器局域网，是国际上应用最广泛的现场总线之一。最初，CAN 被设计作为汽车环境中的微控制器通讯[21]，在各个车载电子控制装置 ECU 之间交换信息，形成汽车电子控制网络。比如：发动机管理系统、变速箱控制器、仪表装备、电子主干系统中，均嵌入 CAN 控制装置。

一个由 CAN 总线构成的单一网络中，每个节点有一个 ID 号。实际应用中，节点数目受网络硬件的电气特性所限制。例如，当使用 Philips P82C250 作为 CAN 收发器时，同一网络中允许挂接 110 个节点。CAN 可提供高达 256Kbps 的数据传输速率，这使实时控制变得非常容易。另外，硬件的错误检测特性也增强了 CAN 的抗电磁干扰能力。

CAN 的规范定义了模型的最下面两层：数据链路层和物理层。应用层协

议可以由 CAN 用户定义成适合特别工业领域的任何方案。已在工业控制和制造业领域得到广泛应用的标准是 DeviceNet，这是为 PLC 和智能传感器设计的。

在汽车工业，许多制造商都应用他们自己的标准[22]。CAN 能够使用多种物理介质，例如双绞线、光纤等。最常用的就是双绞线。信号使用差分电压传送，两条信号线被称为“CAN_H”和“CAN_L”，静态时均是 2.5V 左右，此时状态表示为逻辑“1”，也可以叫做“隐性”。用 CAN_H 比 CAN_L 高表示逻辑“0”，称为“显形”，此时，通常电压值为：CAN_H = 3.5V 和 CAN_L = 1.5V。

CAN 总线发展到今天，派生出一个新的名词——CAN-BUS[23]。CAN-BUS 顾名思义就是车载 CAN 总线。CAN 总线最早被应用于汽车电子系统的通讯上，起源于欧洲，专门用于高档轿车[24]。CAN-BUS 总线技术的最大的优点是减少线束的数量和控制器接口的引脚数，与此同时可以更简单、迅速的实现在线编程、在线诊断，甚至多个控制器同时工作等新功能。CAN-BUS 技术中的通讯节点是控制器、智能传感器或智能执行单元。在现代轿车上，只需要很少的几个节点，就可以实现全车数据共享。

◆LIN总线[25]的数据传输速率是20Kbps,属于低速网，主要用于车身的连网。LIN 的目标是为现有汽车网络(例如CAN总线)提供辅助功能，因此LIN总线是一种辅助的总线网络，在不需要CAN总线的带宽和多功能的场合，比如智能传感器和制动装置之间的通讯使用LIN总线可大大节省成本。

LIN技术规范中，除定义了基本协议和物理层外，还定义了开发工具和应用软件接口。LIN通讯是基于SCI(UART)数据格式，采用单主控制器/多从设备的模式仅使用一根12V信号总线和一个无固定时间基准的节点同步时钟线。

这种低成本的串行通讯模式和相应的开发环境，已经由LIN协会制定成标准。LIN的标准化将为汽车制造商以及供应商，在研发应用系统时降低成本。

典型的LIN总线应用[26]是汽车中的联合装配单元，如门、方向盘、座椅、空调、照明灯、湿度传感器、交流发电机等。对于这些成本比较敏感的单元，LIN可以使那些机械元件，如智能传感器、制动器或光敏器件得到较广泛的使用。这些元件可以很容易的连接到汽车网络中，并得到十分方便的维护和服务。在LIN实现的系统中，通常将模拟信号量用数字信号量所替换，这将使总线性能优化。

◆K线、L线网络是汽车内部转为故障诊断设计的，同车身网络独立的网络，它负责连接汽车上的ECU，将ECU的信息传输到故障诊断仪器。其连接原理图如图2－1所示。

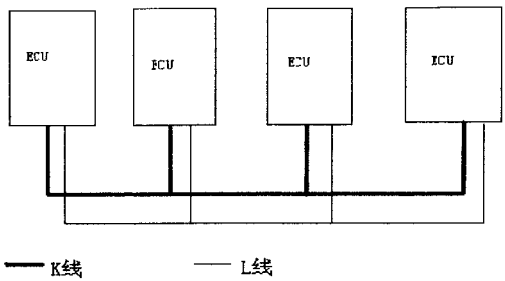

图 2－1 K、L线连接原理图

2．3．2 汽车故障诊断协议 KWP2000

OBD-II[27]是美国 1995 年广泛采用的第二代诊断标准，其数据传输线有两个标准。一个是欧洲标准即 ISO，另一个是美国统一标准即 SAE 标准，该协议主要用于发动机故障诊断。目前开始广泛推广的 KWP2000 通信诊断协议是 OBD-II 的改进和扩充，其对汽车故障的诊断范围更加广泛，几乎涉及所有电控系统的状态反馈。这些诊断协议已成为各种轿车的必备工能。

KWP2000 是目前汽车故障诊断领域应用较为普遍的串行通讯协议，规定了车载电子控制单元（例如电子燃油喷射系统、自动变速箱、防暴死系统等）与上层设备通过串行数据线实现串行通讯的通用要求。

KWP2000 由物理层、数据链路层、应用层三个层构成。物理层实现诊断服务，用于配置硬件系统，指导接口电路的设计。KWP2000 的链路层定义通讯消息基本格式如图 2－2 所示。

Fmt	Tgt	Src	Len	Slt	Data	CS

图 2－2 帧结构

表中各参数含义如下:Fmt :帧字节;Tgt :目标地址;Src :源地址;Len :附加长度字节; Sld :功能识别字节;Data :数据字节;CS:校验和。KWP2000 协议采用消息结构进行传递信息,可分为请求消息、指示消息和响应消息,其中响应消息可分正响应和负响应,所有这些消息都具有相同结构。

KWP2000 的应用层定义了服务标识符的字节编码及十六进制数值；诊断服务请求与相应参数的字节编码；标准参数的十六进制数值。KWP2000 的三层结构如图 2－3 所示。

系统的物理层是汽车内部 K 线、L 线网络，收集以‘1’‘0’数据流形式传输的各个部件的数据；数据链路层采用 KWP2000 规定的数据格式对物理层

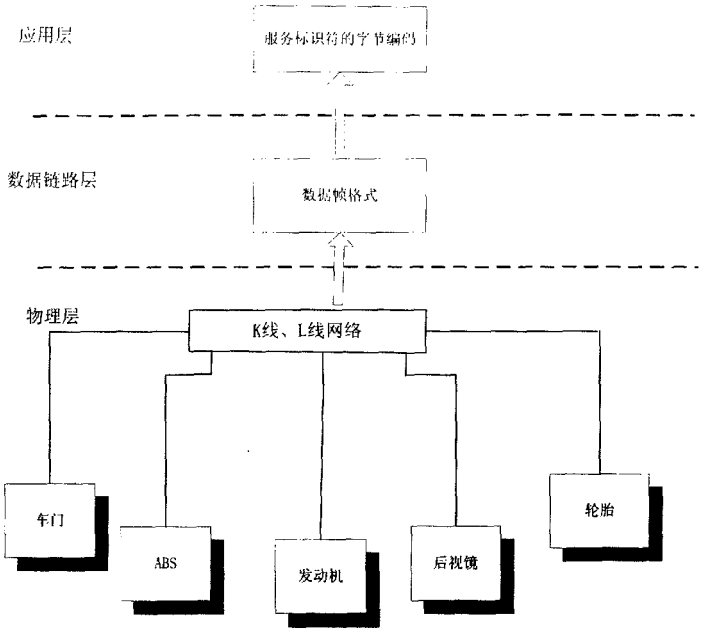

图 2—3 KWP2000 结构图

收集到的数据进行打包，形成一定格式的数据帧；应用层接收数据帧，按照KWP2000 采用定义的应用层服务标识符的字节编码、诊断服务请求与相应参数的字节编码，对数据帧进行处理，得出最终的结果。

独立于车身网络设计 K 线、L 线网络用于汽车故障诊断，一方面增加了成本，另一方面使得汽车内部网络更加复杂。车身网络有许多各 CAN 节点构成，其中已经包括了各种 ECU。将故障诊断代码同车身网络进行整合，直接用车身网络进行汽车故障诊断不失是一种好的设想，本文便是针对这一点展开的。

2．4 数据接口

数据输出接口是整个车载故障诊断系统将汇总数据输入输出的接口。根据 SAE 规定的 OBD 标准，车辆行业使用 K、L 线制进行诊断或标定[28]，L 线单向传输，只在系统初始化时传递从诊断设备到车辆总线的 ECU 地址，系统联接成功后，L 线恒定为高状态（无信息传递）。因此，为使系统设计简单，实际应用中大部分不使用 L 线。K 线可双向传递数据，系统初始化时先传递 ECU 地

址,联接成功后用于信息交换,接口转换典型芯片为Motorola 公司的33290。

MC33290 是专门为车辆诊断而开发的双向、半双工通讯接口芯片,该芯片可实现单片机串口与 K 线电平转换,符合诊断系统 ISO9141 规范 ,K 线信号输出具有以下主要特点:

工作电压范围 8 V～18 V ;
使用环境温度为- 40 ℃～125 ℃;
与单片机 CMOS 电平无缝连接;
具有对地线短路保护;
最大传输速率超过 50 K;
支持大电流;
抗 8. 0 kV 静电保护;

工作时电源电压无电流输出。芯片引脚数为 8 ,各引脚定义如表 2—1

表 2—1 引脚说明

引 脚	定 义
1	K线电平电压输入(VB)
2	空
3	地线
4	K- 线
5	MCU 串口发送输入
6	MCU 串口接收输出
7	5 V 电源
8	片选信号

2. 5 故障诊断仪器

诊断设备一般通过专用诊断接口(SAE2J1962) 与车辆总线进行数据交换。诊断设备由第三方开发,并符合 SAE 标准规定,诊断接口主要完成诊断设备与车辆 ECU 通讯信号转换。

如果 K 线、L 线网络用车身网络取代，那么 KWP2000 协议的物理层不再遵循其自身的数据传输线标准，诊断设备也需要相应的改进。本文中用 USB 来统一诊断设备同车身网络的接口。

第 3 章 基于 USB 的车载信息系统的体系结构

本章论述基于 USB 的车载故障诊断系统，用 USB 接口取代专用诊断接口(SAE2J1962)，使得汽车通过 CAN 节点传送的信息能够与 PC 机兼容，诊断设备不需要第三方开发，进而降低系统成本。

3．1 系统的整体结构

本文论述的车载故障诊断系统由 CAN 网络、CAN/USB 网关、USB 接口等几部分构成。CAN 网络收集连接在各个 CAN 节点上的所有汽车部件的数据，然后将数据汇总到 CAN/USB 网关。USB 接口位于车身 CAN 网络的 CAN/USB 网关上，通过网关实现 USB 数据结构同 CAN 数据结构的相互转换。其结构示

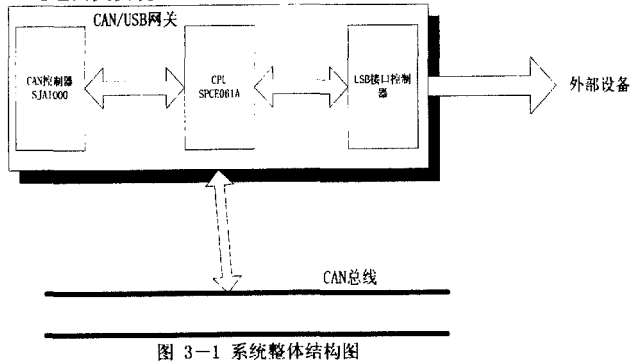

图 3－1 系统整体结构图

意图如图 3－1 所示。CAN/USB 网关由 USB 接口控制器 PDIUSBD12、CPU、CAN 控制器 SJA1000 构成。PDIUSBD12、SJA1000 之间通过 CPU 实现数据格式的转换，PDIUSBD12 实现 CAN/USB 网关同 PC 的接口，SJA1000 实现 CAN/USB 网关同汽车内 CAN 总线的接口。

3．2 CAN 网的基本结构

3．2．1 CAN 控制器

SJA1000[29][30]是一种独立控制器用于移动目标和一般工业环境中的区域网络控制。它是PHILIPS半导体PCA82C200 CAN控制器BasicCAN的替代产品而且它增加了一种新的工作模式PeliCAN,这种模式支持具有很多新特性的CAN2.0B 协议。SJA1000为28脚封装，它实现了CAN2.0协议的物理层和数据

链路层，其管脚图见图3－2所示。其中MODE用于选择芯片工作于Inter或Motorola模式，AD0-AD7为地址数据复用线，ALE为Inter模式下的地址选通信号。

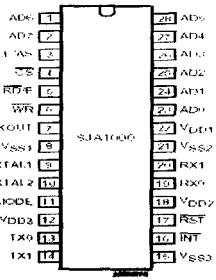

图 3－2 SJA1000管脚图

3．2．2 CAN 节点

CAN 控制器 SJA1000 构成的 CAN 节点同 CAN 总线连接时，使用 CAN 收发器 82C250，82C250 同 SJA1000 之间使用高速光耦 6N137 进行隔离，连接使用平行双股线或双绞线。

CAN 节点的硬件原理图如图 3－3 所示。

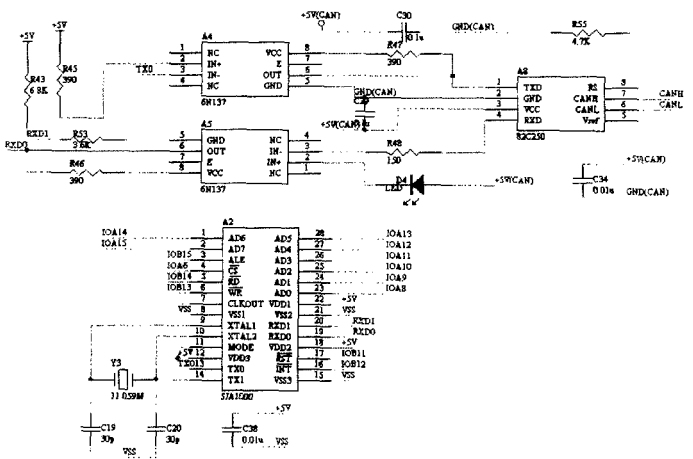

图 3－3 CAN 节点原理图

3．3 USB 接口

PDIUSBD12[10] 是一款性价比很高的 USB 器件，它通常用作微控制器系统中实现与微控制器进行通信的高速通用并行接口，它还支持本地的 DMA 传输。

PDIUSBD12 完全符合 USB1.1 版的规范，连同 LazyClock 输出，可以满足使用 ACPI OnNOW 和 USB 电源管理的要求，低的操作功耗可以应用于使用总线供电的外设。

此外它还集成了许多特性，包括 SoftConnetTM GoodLinkTM，可编程时钟输出，低频晶振和终止寄存器集合，所有这些特性都为系统显著节约了成本，同时使 USB 功能在外设上的应用变得容易。

PDIUSBD12 的功能描述，其功能框图如图 3－4 所示：

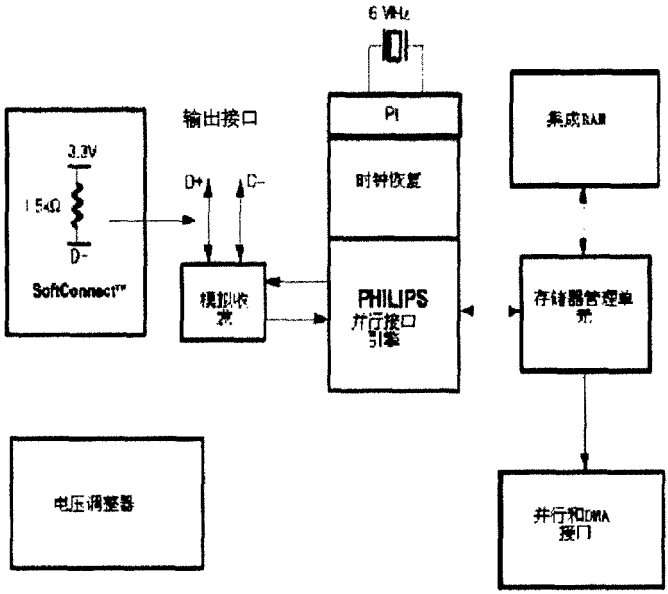

图 3－4　PDIUSBD12 功能框图

1．模拟收发器

集成的收发器接口可通过终端电阻直接与 USB 电缆相连。

2．电压调整器

片内集成了一个 3.3V 的调整器，用于模拟收发器的供电。该电压还作为输出连接到外部 1.5kΩ 的上拉电阻，可选择 PDIUSBD12 提供的带 1.5kΩ

内部上拉电阻的软件连接技术。

3. PLL

片内集成了 6M 到 48M 时钟乘法 PLL。这样就可使用低成本的 6M 晶振 EMI，也随之降低 PLL 的工作，不需要外部元件。

4. 时钟恢复

位时钟恢复电路使用 4X 过采样规则，从进入的 USB 数据流中恢复时钟，它能跟踪 USB 规定范围内的抖动和频漂。

5. Philips 串行接口引擎 PSIE

Philips SIE 实现了全部的 USB 协议层，完全由硬件实现而不需要固件的参与。该模块的功能包括同步模式的识别，并行/串行转换，位填充/解除填充，CRC 校验/产生，PID 校验/产生，地址识别和握手评估/产生。

6. SoftConnectTM

与 USB 的连接是通过 1.5kΩ 上拉电阻将 D+ 用于高速 USB 器件置为高实现的。1.5kΩ 上拉电阻集成在 PDIUSBD12 片内，默认状态下不与 VCC 相连，连接的建立通过外部/系统微控制器发送命令来实现。这就允许系统微控制器在决定与 USB 建立连接之前，完成初始化时序。USB 总线连接可以重新初始化，而不需要拔出电缆。PDIUSBD12 在连接可以建立之前，会检测 USB VBUS 是否可用，VBUS 可通过 EOT_N 管脚进行检测。需要注意的是内部电阻的误差 25% 大于 USB 规格的 5%，但用于连接的 VSE 电压规格仍然有足够的余量。SoftConnectTM 是 Philips 半导体一项尚未获批准的专利技术。

7. GoodLinkTM

GoodLinkTM 技术可提供良好的 USB 连接指示，在枚举中 LED 指示根据通信的状况间歇闪烁。当 PDIUSBD12 成功地枚举和配置后，LED 指示将一直点亮，随后与 PDIUSBD12 之间成功的传输带应答，将关闭 LED。处于挂起状态时，LED 将会关闭。该特性为 USB 器件集线器和 USB 通信状态，提供了用户友好的指示作为一个诊断工具。它对隔离故障的设备是很有用的。该特性降低了现场支持和热线的成本。

8. 存储器管理单元 MMU 和集成 RAM

在以 12M/s 的速率传输并与微控制器并口相连时，MMU 和集成 RAM 作为 USB 之间速度差异的缓冲区。这就允许微控制器以它自己的速率对 USB 信息包进行读写。

9. 并行和 DMA 接口

一个普通的并行接口定义成易于使用，快速而且可以与主流的微控制器直接接口。对一个微控制器而言，PDIUSBD12 看起来就象一个带 8 位数据总线和一个地址位占用 2 个位置的存储器件。PDIUSBD12 支持多路复用和非复用的地址和数据总线，还支持主端点与本地共享 RAM 之间直接读取的 DMA 传输，支持单周期和突发模式的 DMA 传输。

USB 硬件构成原理图如图 3－5 所示。PDIUSBD12 芯片的数据线同 CPU 的 IOA8-IOA15 连接，中断管脚同 CPU 的 IOB2 连接，读、写、复位信号分别同 CPU 的 IOB7、IOB6、IOB3 连接。供电电源使用 5V，不使能 DMA 方式。外部元件包括 6M 晶振，USB 物理接口等等。PDIUSBD12 管脚功能说明如下表所示。

表 3－1 PDIUSBD12 管脚说明

管脚	类型	符号	描述
1	IO2	DATA<0>	双向数据位 0
2	IO2	DATA<1>	双向数据位 1
3	IO2	DATA<2>	双向数据位 2
4	IO2	DATA<3>	双向数据位 3
5	P	GND	地
6	IO2	DATA<4>	双向数据位 4
7	IO2	DATA<5>	双向数据位 5
8	IO2	DATA<6>	双向数据位 6
9	IO2	DATA<7>	双向数据位 7
10	I	ALE	地址锁存使能
11	I	CS_N	片选（低有效）
12	LOD4	SUSPEND	器件处于挂起状态
13	O2	CLKOUT	可编程时钟输出
14	OD4	INT_N	中断（低有效）
15	I	RD_N	读选通（低有效）
16	I	WR_N	写选通（低有效）
17	O4	DMREQ	DMA 请求
18	I	DMACK_N	DMA 应答（低有效）
19	I	EOT_N	DAM 传输结束（低有效）
20	I	RESET_N	复位
21	OD8	GL_N	GoodLinkLED 指示器
22	I	XTAL1	晶振连接端 1
23	O	XTAL2	晶振连接端 2。如果采用外部时钟信号取代晶振，可连接 XTAL1，XTAL2 应当悬空
24	P	Vcc	电源电压，要使器件工作在 3.3V，对 Vcc 和 Vout3.3 脚都提供 3.3V。
25	A	D-	USB D-数据线
26	A	D+	USB D+数据线
27	P	Vout3.3	3.3V 调整输出。
28	I	A0	地址位。A0=1 选择命令指令，A0=0 选择数据。

注：　O2：　2mA 驱动输出　　OD4：　4mA 驱动开漏输出
　　OD8：　8mA 驱动开漏输出　　IO2：　4mA 输出

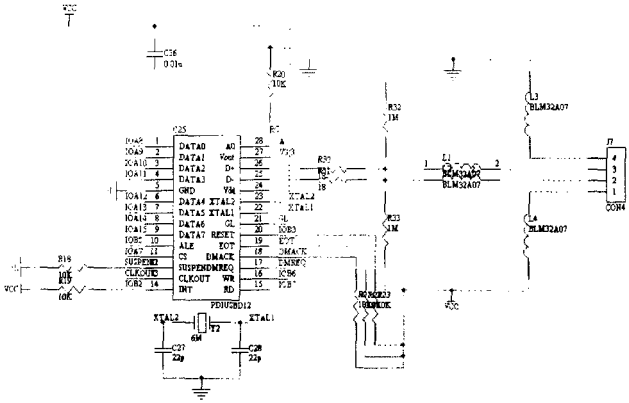

图 3－5 USB 硬件原理图

3．4 CAN/USB 网关的结构

3．4．1 网关使用的 CPU

CPU 硬件接口原理图如图 3－6 所示。

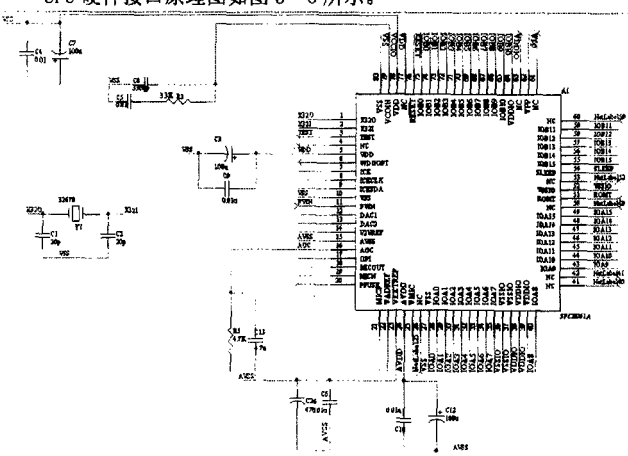

图 3－6 CPU 硬件原理图

本系统使用的 CPU 是台湾凌阳公司生产的 16 位单片机 SPCE061A[32]。该芯片为 80 脚 LQFP 贴片封装，SPCE061A 在 2.6V~3.6V 工作电压范围内的工作速度范围为 0.32MHz~49.152MHz，较高的工作速度使其应用领域更加拓宽。2K 字 SRAM 和 32K 字闪存 ROM 仅占一页存储空间，大容量的 Flash 使得嵌入式操作系统的使用成为可能（本系统使用 μC/OS-II 作为操作系统，整个操作系统下载到 Flash 内，大约占 25K 的程序空间）。2 个 16 位可编程定时器/计数器(可自动预置初始计数值)，32 位通用可编程输入/输出端口，14 个中断源可来自定时器 A / B 时基，2 个外部时钟源输入，可按键唤醒。32 位可编程的多功能 I/O 端口，两个 16 位定时器/计数器，32768Hz 实时时钟，低电压复位/监测功能，系统处于备用状态下(时钟处于停止状态)，耗电小于 2μA@3.6V，较小的能耗正适合车载电子的要求。

3．4．2 CAN/USB 网关的设计

由 PDIUSBD12、SJA1000、SPCE061A 构成的 CAN/USB 网关的硬件原理图如图 3－7 所示。

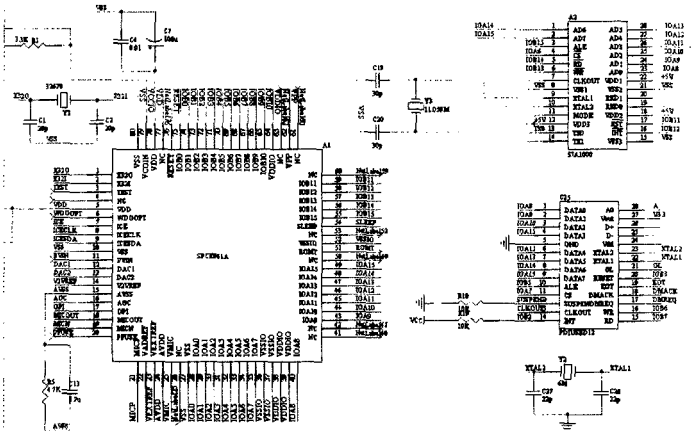

图 3－7 CAN/USB 网关的结构原理图

图中主要给出了 PDIUSBD12、SJA1000、SPCE061A 之间的连接关系，忽略了各个芯片的外围器件。CPU 使用其 IOA 口作为数据线，IOB 口作为控制线，通过控制线控制 SJA1000 和 PDIUSBD12 的片选信号、读写信号、复位信号、中断信号、地址选通信号等等。

第 4 章 基于 USB 的车载信息系统的软件实现

4．1 系统的整体软件流程

汽车上 USB 接口工作在一种被动的方式[33][34]。当进行故障诊断的时候，外部笔记本上的故障诊断程序通过 USB 接口向汽车上的 CAN/USB 网关发送命令，汽车上的 CAN/USB 网关收到命令并由 CPU 识别命令，进而 CPU 再通知 CAN 控制器应该传输哪个 CAN 节点的数据到 CAN/USB 网关，然后通过 USB 接口将数据传输到笔记本上的故障诊断程序进行分析处理。其软件流程图如图 4－1 所示。

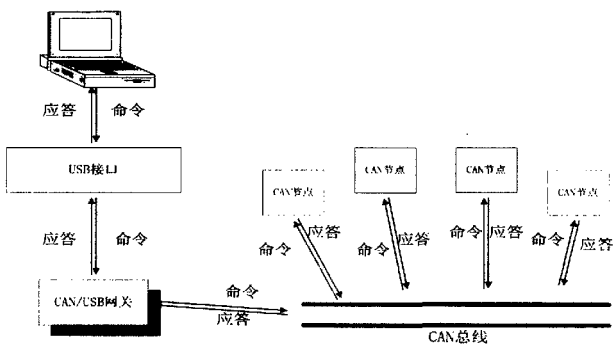

图 4－1 系统工作流程图

4．2 CAN 网络的帧传输

CAN的数据链路层[35]的MAC子层是CAN协议的核心。它把接收到的报文提供给LLC子层，并接收来自LLC子层的报文。MAC子层负责报文分帧、仲裁、应答、错误检测和标定。MAC子层也被称作故障界定的管理实体监管。此故障界定为自检机制，以便把永久故障和短时扰动区别开来。LLC子层涉及报文滤波、过载通知、以及恢复管理。数据链路层的结构如下图所示。

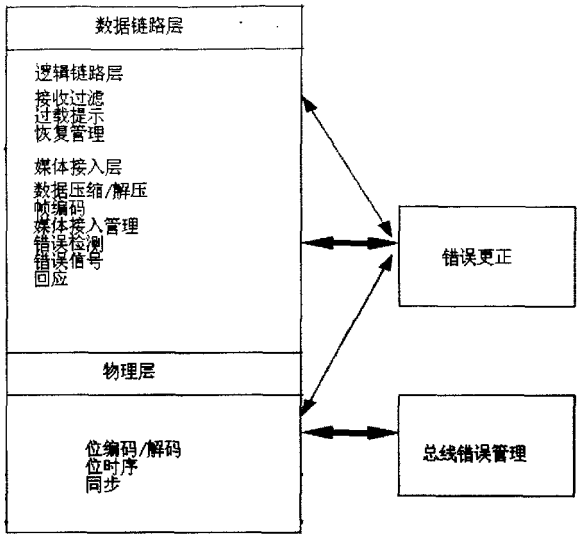

图 4－2 CAN数据链路层结构

4．2．1 帧格式

有两种不同的帧格式，不同之处为识别符场的长度不同：具有11位识别符的帧称之为标准帧。而含有29位识别符的帧为扩展帧。

4．2．2 帧类型

报文传输由以下4个不同的帧类型所表示和控制：

数据帧：数据帧将数据从发送器传输到接收器。

远程帧：总线单元发出远程帧，请求发送具有同一识别符的数据帧。

错误帧：任何单元检测到总线错误就发出错误帧。

过载帧：过载帧用以在先行的和后续的数据帧（或远程帧）之间提供一附加的延时。

数据帧和远程帧可以使用标准帧及扩展帧两种格式。它们用一个帧间空间与前面的帧分隔。四种帧类型中，远程帧、错误帧、过载帧都属于命令帧，其帧格式都比较简单，本文重点论述的是封装有KWP2000数据帧的CAN数据帧的帧格式。

4．2．3 数据帧

数据帧由7 个不同的位场组成：帧起始（Stsrt of Frame）、仲裁场（Arbitration Frame）、控制场（Control Frame）、数据场（Data Frame）、CRC 场（CRC Frame）、应答场（ACK Frame）、帧结尾（End of Frame）。数据场的长度可以为0。其结构图如下。

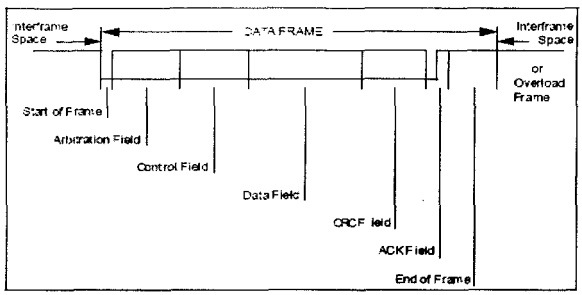

图 4－3 CAN 数据帧结构

◆帧起始

－ 帧起始（SOF）标志数据帧和远程帧的起始，仅由一个“显性”位组成。只在总线空闲时才允许站开始发送信号。所有的站必须同步于首先开始发送报文的站的帧起始前沿。

◆仲裁场

－ 仲裁场包括29位识别符、SRR 位、IDE 位、RTR 位。其识别符由ID28...ID0。

◆控制场

－ 控制场由6个位组成。包括四个位的数据长度代码和两个保留位：r1 和r0。现在根据需要将两个保留位r1和r2分别定义为DIV和ED。控制场格式如图4－4所示。

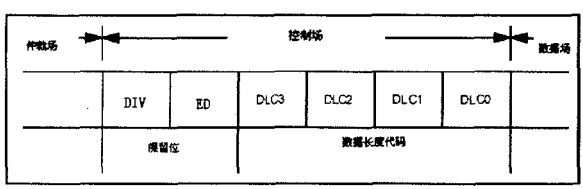

图 4－4 控制场格式

DIV为‘1’表示数据帧进行了拆分，为‘0’表示没有进行拆分。

ED为‘1’表示此数据帧后面还有帧，此帧不是最后一帧；为‘0’表示此数据帧后面没有帧，此帧是最后一帧数据了。

◆数据场

— 数据场由数据帧里的发送数据组成。它可以为0～8个字节，每字节包含了8个位，首先发送MSB。数据场内的8个字节中，用于存放汽车某个部件的故障代码。如果一个完整的故障代码超过了8个字节，则要进行拆分。

数据场结构图如图4－5所示。

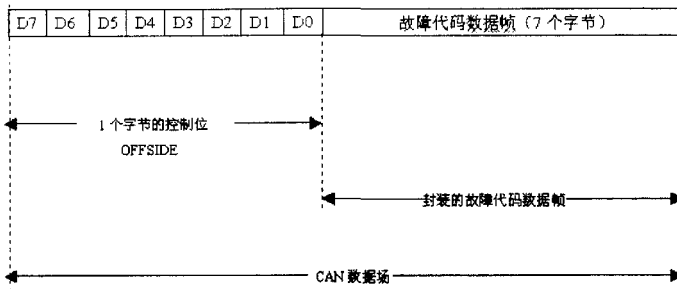

图 4－5 CAN数据场格式

◆CRC场

— CRC场包括CRC序列（CRC SEQUENCE），其后是CRC界定符（CRC DELIMITER）。

◆应答场

— 应答场长度为2个位，包含应答间隙（ACK SLOT）和应答界定符（ACK DELIMITER）。在ACK场（应答场）里，发送站发送两个“隐性”位。当接收器正确地接收到有效的报文，接收器就会在应答间隙（ACK SLOT）期间（发送ACK信号）向发送器发送一“显性”位以示应答。

◆帧结尾

— 每一个数据帧由一标志序列定界。这个标志序列由 7 个“隐性”的位组成。

4．3 USB 固件程序的设计

根据 USB1.1 协议的规定，USB 接口是工作在被动工作方式下。

本系统设计中，将 USB 接口芯片 PDIUSBD12 的中断输出管脚同 CPU 的外部中断管脚相连，这样每当上位机发命令过来时，都将使 PDIUSBD12 芯片产生中断输出信号，该信号触发 CPU 的外部中断，在 CPU 外部中断的中断服务程序中[36]，CPU 读 PDIUSBD12 的中断寄存器来得知到来的是什么中断，根据不同类型的中断，调用不同的子程序。其工作流程如图 4－6 所示。由于对 PDIUSBD12 的操作是一种不能重入的操作，所以必须要在这一次操作完成后才能进行下一次对 PDIUSBD12 的操作。因此，对 PDIUSBD12 的操作采用了一个信号量来管理。这种信号量的管理方式就像用一把钥匙开多个门，一次只能开一扇门，开门的先后由任务的优先级决定。

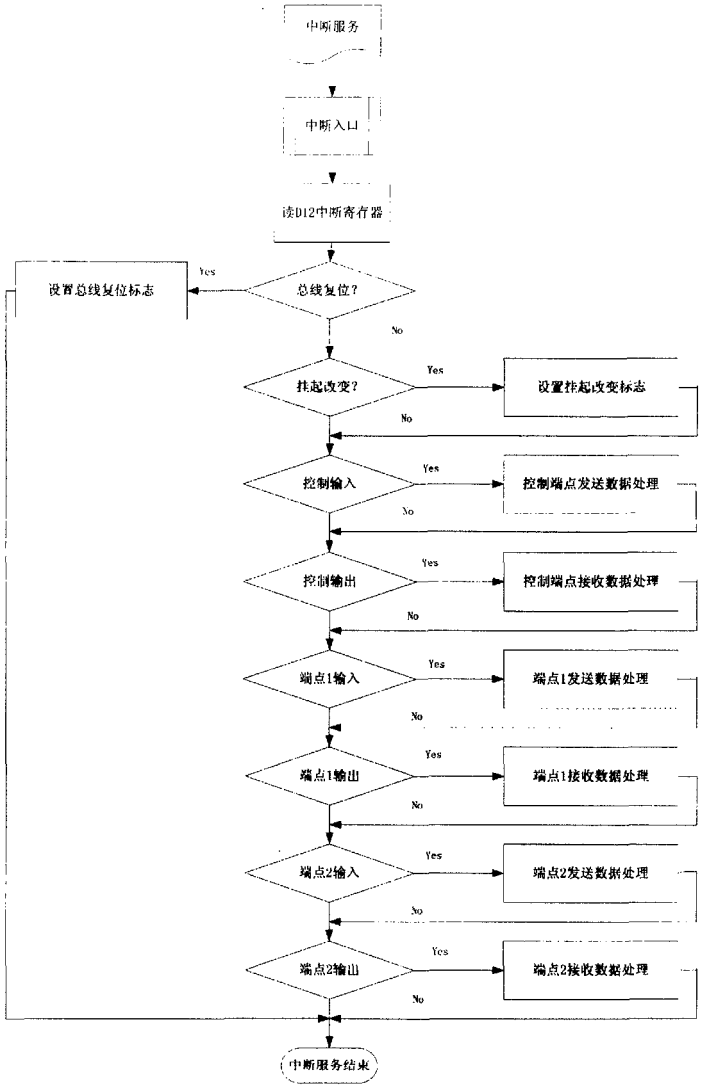

图 4—6 USB 中断服务程序流程图

4．4 CAN/USB 网关的软件设计

4．4．1 嵌入式操作系统μC/OS－II

CAN/USB 网关由实时嵌入式操作系统μC/OS－II 完成其各项工作，包括系统初始化程序、USB 固件程序和 SJA1000 控制程序。程序重包括多个任务：USB 中断处理任务、USB 请求处理任务、SJA1000 发送任务、SJA1000 接收任务等等，所以使用操作系统的多任务调度机制，将可以很好的完成系统的需要。

1）μC/OS－II 概述

目前，实时操作系统已广泛应用于工业控制的各个领域。μC/OS-II[37]作为一个实时内核，由于其源码公开、代码规范，广受开发人员的喜爱。

μC/OS－II 包括任务调度、时间管理、内存管理、资源管理（信号量、邮箱、消息队列）四大部分，没有文件系统、网络接口、输入输出界面。有 64 个优先级，系统占用 8 个，用户可创建 56 个任务，每个任务均有一个独有的优先级。由于其内核为抢先式，所以总是处于运行态最高优先级的任务占用 CPU。系统提供了丰富的 API 函数，实现进程之间的通信以及进程状态的转化。不支持时间片轮转。

2）μC/OS－II 的体系结构

μC/OS-II 的软件体系结构如图 4－7 所示。从图 4－7 中可以看到，如果要使用μC/OS-II，必须为其编写 OS_CPU.H、OS_CPU_C.C、OS_CPU_A.ASM 三个文件。这三个文件是与芯片的硬件特性有关的，它们主要提供任务切换与系统时钟的功能。其它文件用 C 写成，它们为系统提供任务管理、任务之间通信、时间管理以及内存管理等功能。

μC/OS-II 工作的基本思路就是“近似地每时每刻总是让优先级最高的就绪任务处于运行状态”。为了保证这一点，它在调用系统 API 函数、中断结束、定时中断结束时总是执行调度算法。原作者通过事先计算好数据，简化了运算量，通过精心设计就绪表结构，使得延时可预知。任务的切换是通过模拟一次中断实现的。

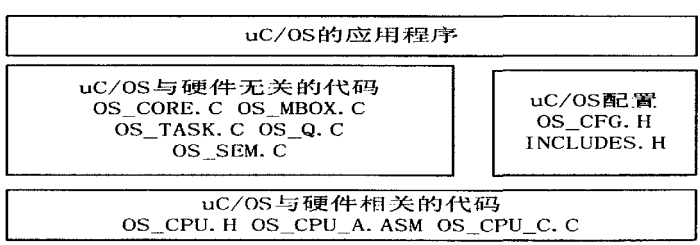

图 4－7　μC/OS-II 的体系结构

3) μC/OS-II 的任务调度原理

μC/OS-II 工作核心原理是：近似地让最高优先级的就绪任务处于运行状态。

操作系统将在下面情况中进行任务调度：调用API函数(用户主动调用)，中断(系统占用的时间片中断 OsTimeTick(),用户使用的中断)。操作系统调度算法的整体思路如下：

(1)在调用 API 函数时，有可能引起阻塞,如果系统 API 函数察觉到运行条件不满足，需要切换就调用 OSSched()调度函数，这个过程是系统自动完成的，用户没有参与。OSSched()判断是否切换，如果需要切换，则此函数调用 OS_TASK_SW()。这个函数模拟一次中断(在 51 里没有软中断，我用子程序调用模拟，效果相同)，好象程序被中断打断了，其实是 OS 故意制造的假象，目的是为了任务切换。既然是中断，那么返回地址(即紧邻 OS_TASK_SW()的下一条汇编指令的 PC 地址)就被自动压入堆栈，接着在中断程序里保存 CPU 寄存器(PUSHALL)……。堆栈结构不是任意的，而是严格按照μC/OS-II 规范处理。OS 每次切换都会保存和恢复全部现场信息(POPALL)，然后用 RETI 回到任务断点继续执行。这个断点就是 OSSched()函数里的紧邻 OS_TASK_SW()的下一条汇编指令的 PC 地址。切换的整个过程就是，用户任务程序调用系统 API 函数，API 调用 OSSched()，OSSched()调用软中断 OS_TASK_SW()即 OSCtxSw，返回地址(PC 值)压栈，进入 OSCtxSw 中断处理子程序内部。反之，切换程序调用 RETI 返回紧邻 OS_TASK_SW()的下一条汇编指令的 PC 地址，进而返回 OSSched()下一句，再返回 API 下一句，即用户程序断点。因此，如果任务从运行到就绪再到运行，它是从调度前的断点处运行。

(2)中断会引发条件变化，在退出前必须进行任务调度。μC/OS-II 要求中断的堆栈结构符合规范，以便正确协调中断退出和任务切换。前面已经说到任务切换实际是模拟一次中断事件，而在真正的中断里省去了模拟(本身就是中断嘛)。只要规定中断堆栈结构和μC/OS-II 模拟的堆栈结构一样，就能保证在中断里进行正确的切换。任务切换发生在中断退出前，此时还没有返回中断断点。仔细观察中断程序和切换程序最后两句，它们是一模一样的，POPALL+RETI。即要么直接从中断程序退出，返回断点；要么先保存现场到 TCB，等到恢复现场时再从切换函数返回原来的中断断点(由于中断和切换函数遵循共同的堆栈结构，所以退出操作相同，效果也相同)。用户编写的中断子程序必须按照μC/OS-II 规范书写。任务调度发生在中断退出前，是非常及时的，不会等到下一时间片才处理。OSIntCtxSw()函数对堆栈指针做了简单调整，以保证所有挂起任务的栈结构看起来是一样的。

在为μC/OS-II 编写任务切换代码时需要注意的是：μC/OS-II 在每次发生中断后都会产生任务调度，但在中断结束后进行的任务切换，不能调用普通任务切换函数，这是因为在中断过程中往往伴随将 CPU 的状态寄存器压栈操作。以凌阳单片机为例，在中断后，芯片将 PC 和 SR 寄存器的值压入堆栈，因此，在中断结束后进行的任务切换中必须对堆栈指针进行调整。

μC/OS-II 的移植相对比较简单，只与 4 个文件相关：汇编文件（OS_CPU_A.ASM)、处理器相关 C 文件（OS_CPU.H、OS_CPU_C.C）和配置文

件（OS_CFG.H）。操作系统移植的过程，就是针对不同的 CPU 修改以上四个文件的过程。

4．4．2 在 SPCE061A 上移植 μC/OS-II

1）移植涉及到的四个文件

OS_CPU_A.ASM 内主要包括几个同移植紧密相关的子程序：OSStartHighRdy、OSCtxSw、OSIntCtxSw、OSTickISR。OSStartHighRdy 对处于就绪态的优先级最高的任务开始运行；OSCtxSw 对任务进行调度；OSIntCtxSw 在中断中对任务进行调度；OSTickISR 提供始终节拍。这几个子程序必须要用汇编语言写，因为它们是直接同硬件相关的。

OS_CPU.H 内主要包括用＃define 语句定义的、与处理器相关的常数、宏以及类型。

OS_CPU_C.C 内主要是用来描述芯片堆栈的设计。

OS_CFG.H 内主要包括内核编译相关的参数。

2）μC/OS-II 在 SPCE061A 上的移植过程

首先，根据 SPCE061A 的要求对 OS_CFG.H 内的配置选项进行调整。

然后，在 OS_CPU_C.C 中设计适合 SPCE061A 使用的堆栈。SPCE061A 的寄存器组里有八个 16 位的用户寄存器，分别是 SP 和 R1-R7。其中 SP 是堆栈指针寄存器，R5 是基址指针寄存器，R6 是状态寄存器，R7 是程序指针寄存器，其余 R1-R5 是一般寄存器。所以设计的堆栈结构如下：

◇void *OSTaskStkInit(void (*task)(void *pd), void *pdata, void *ptos,

```
INT16U opt)
{
   OS_STK* stk;
   stk=(OS_STK*)ptos;
   *stk++=0x0007;                          /*堆栈的长度*/
   *stk++=*((INT16U*)task+1);              /* PC (R7)  */
   *stk++=0x0000;                            /* SR (R6) */
   *stk++=0x0000;                          /* R5  */
   *stk++=0x0000;                          /* R4  */
   *stk++=0x0000;                          /* R3  */
   *stk++=0x0000;                          /* R2  */
   *stk++=0x0000;                          /* R1  */
   return (void*)ptos;
}
```

操作系统中的每一个任务，在其被创建时都会有自己的堆栈（实际上是一块存储空间），结构同上。

接下来，使用 SPCE061A 的汇编语言编写 OS_CPU_A.ASM 内的

OSStartHighRdy、OSCtxSw、OSIntCtxSw、OSTickISR 等几个子程序。

其中，OSStartHighRdy 子程序的功能是：使处于就绪态的优先级最高的任务运行。OSCtxSw 子程序的功能使：实现任务调度。OSIntCtxSw 子程序的功能使：在中断服务程序中实现任务调度。OSTickISR 子程序的功能：为操作系统自身提供时钟。

```
◇_OSStartHighRdy:
    CALL _OSTaskSwHook
OSCtxSw_in:
    // OSTCBStrPtr -->R1
    // 通过在编译器的变量窗口查看数据结构 OSTCB 可知：以下三句语句的含义
    // R1 为任务堆栈的头指针
    R1=[_OSTCBCur]
    // R2 中为任务堆栈头地址存放的第一条内容，是堆栈的长度
    R1=[R1]
    R2=[R1]
    // R3 为系统堆栈的高位地址+1
    R3=OSStkStart
RESTORE_STACK:
    R1+=1
    R3-=1
    R4=[R1]
    [R3]=R4
    R2-=1
    JNZ RESTORE_STACK
    // 调整堆栈指针
    R3-=1
    SP=R3
    // OSRunning =TRUE
    R1=0x0001
    [_OSRunning]=R1
    POPALL
    INT FIQ
    INT IRQ
    RETI
◇_OSCtxSw:
    // 将所有寄存器压栈
    PUSHALL
OSIntCtxSw_in:
    // 求出系统堆栈的长度，并将其存入 R2
    R1=SP
```

```
    R2=OSStkStart
    R1+=1
    R2=R2-R1
    // R1 <-- OSTCBStkPtr
    // R1 为任务堆栈的头指针
    R1=[_OSTCBCur]
    // 首先将系统堆栈长度保存在任务堆栈中
    R1=[R1]
    [R1]=R2
    // 得到堆栈的起始地址
    R3=OSStkStart
    // 保存系统堆栈到任务堆栈
save_stack:
    R3-=1
    R1+=1
    R4=[R3]
    [R1]=R4
    R2-=1
    JNZ save_stack
    CALL _OSTaskSwHook
    R1=[_OSTCBHighRdy]
    [_OSTCBCur]=R1
    R1=[_OSPrioHighRdy]
    [_OSPrioCur]=R1
    JMP OSCtxSw_in
◇_OSIntCtxSw:
   //R1=SP
   //R1+=0x0007    //***本移植对所有编译器通用，此处对多压栈的偏移
量，不需要作任何调整***，
    //SP=R1
    JMP OSIntCtxSw_in
◇_F_TimeISR:
    PUSHALL
    CALL _OSIntEnter
    r1=[_OSIntNesting]
    cmp R1,0x0001
    Jne OUT
    R1=[_OSTCBCur]  //本移植代码不依赖编译器，对所有的编译器通用
    r1=sp
    [_OSTCBCur]=r1
OUT:r1=0x1000
```

```
[P_INT_Clear]=R1
r1=0x1000
[P_INT_Ctrl]=R1
CALL _OSTimeTick
CALL _OSIntExit
POPALL
reti
```

通常，不同的编译器对CPU操作，在产生中断时，堆栈压栈的个数是不相同[98]的，这样就使得μC/OS-II在使用不同编译器对芯片进行移植时，在_OSIntCtxSw子程序内都需要一个特定的偏移量来对堆栈进行调整。在本系统中，对μC/OS-II进行移植时，借鉴了Jean J.Labrosse先生在μC/OS－2.51版中对其作的改进——在TimeISR子程序中加上了一句R1=[_OSTCBCur]，使得μC/OS-II的移植不再依赖任何编译器。

4．4．3 CAN/USB网关的软件流程

在完成了以上工作后，μC/OS-II的内核便成功移植到了SPCE061A上，接下来要做的事便是在内核的基础上写芯片各个部分的驱动程序和最终的应用程序。CAN/USB网关在收到笔记本发来的命令后，对SJA1000进行操作和处理完SJA1000后，向笔记本发送数据时，就要涉及到CAN数据格式同USB数据格式之间的转换。转换过程的实质是对CAN、USB数据格式的去封装和加封装的过程，示意图如图4－8所示。

整个CAN/USB网关的工作流程如图4－9所示。笔记本向PDIUSBD12芯片发送命令，PDIUSBD12收到命令后产生中断信号，中断信号触发CPU的中断服务程序，CPU判断到中断服务程序是端口输入/输出程序时，即跳转到相应的子程序。端口输入/输出子程序负责对CAN控制器SJA1000的读/写操作。当笔记本要求读取某个CAN节点的数据，则调用端口输入子程序，按照PDIUSBD12的FIFO容量，每次读取8个字节的数据通过USB送往笔记本；要求对某个CAN节点发送数据时，则调用端口输出子程序。

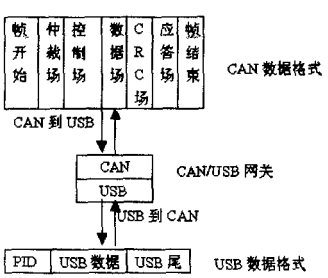

图 4－8 CAN/USB数据格式转换

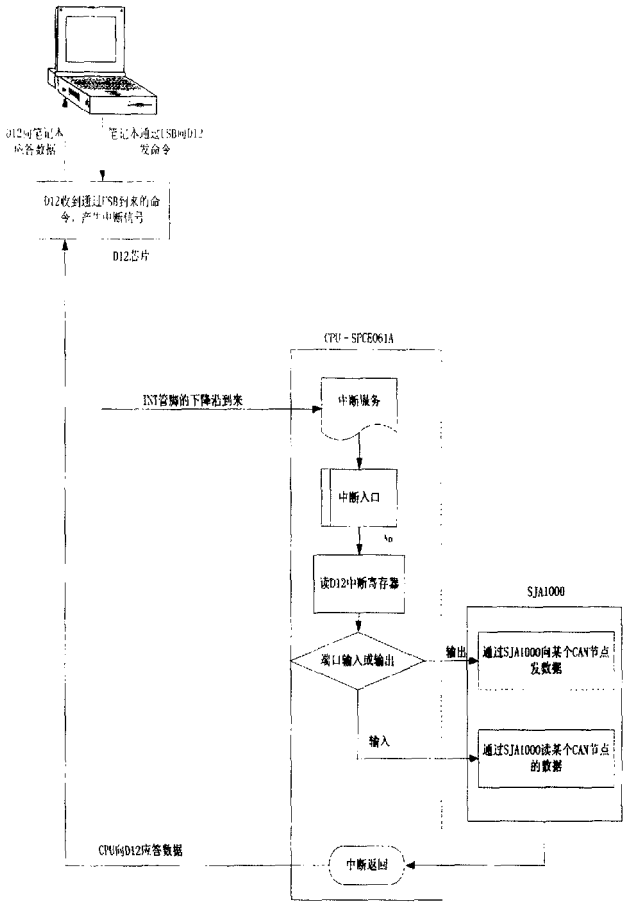

图 4－9 CAN/USB 网关工作流程

4．5 通用故障诊断代码

故障诊断代码是依照 KWP2000 应用层规定的故障代码设计的[39][40]，是目前国际上通用的，现在将其应用于 CAN 的应用层，将来可以用全新的 CAN 上层协议来取代。故障诊断代码定义了在 SSF 14230，SAE J1979 中被车辆制

造商或系统供应者定义的服务标志符数值的不同范围。如表 4—2 所示。

表 4—2 服务标志符约定数值

服务标志符 16 进制数	服务种类 （bit 6）	定义来源
00-0F	请求	SAE J1979
10-1F	请求（bit 6=0）	SSF 14230-3
20-2F		
30-3E		
3F	不做应用	文档保留
40-4F	响应	SAE J1979
50-5F	对服务（$10-$3E）的肯定响应(bit 6=1)	SSF 14230-3
60-6F		
70-7E		
7F	否定响应	SSF 14230-3
80	请求“ESC”-字码	ISO 14230-3
81-8F	请求（bit 6=0）	SSF 14230-2
90-9F	请求（bit 6=0）	为将来扩展预留
A0-B9	请求（bit 6=0）	由车辆制造商定义
BA-BF	请求（bit 6=0）	由系统供应者定义
C0	肯定响应“ESC”-字码	ISO 14230-3
C1-CF	肯定响应（bit 6=1）	SSF 14230-2
D0-DF	肯定响应（bit 6=1）	为将来扩展预留
E0-F9	肯定响应（bit 6=1）	由车辆制造商重定义
FA-FF	肯定响应（bit 6=1）	由系统供应者重定义

此表中以 16 进制数表示的服务标志符，同数据链路层中数据字节内的 SID 服务识别字节对应。不同的 SID 值代表不同的服务请求，故障诊断程序必须要符合此应用层标准，才能识别不同的 16 进制代码所代表的不同的故障信息。

在设计故障诊断程序时，仍然使用国际上通用的故障代码，使得程序具有通用性和实用性。以 PC 机或笔记本为硬件平台运行故障诊断程序构成的故障诊断设备，无需专门开发硬件平台，可以大大降低开发成本，易于实现设备的升级维护。使用 USB 接口进行数据的传输，可以满足高实时性设备的要求，实现数据的同步显示。

第 5 章 典型应用——汽车胎压检测（TPMS）

5．1 典型应用概述

汽车胎压检测作为 CAN 总线中的一个节点，采集轮胎的压力、温度值后，一方面将数据送车载显示器，另一方面通过 CAN/USB 网关将数据送交笔记本上的故障诊断程序分析。故障诊断程序按照 KWP2000 应用层规定的 16 进制故障代码识别各种故障，由 CAN 数据帧封装好的 KWP2000 数据帧传送各种故障代码。典型应用构成原理图如图 5－1 所示。

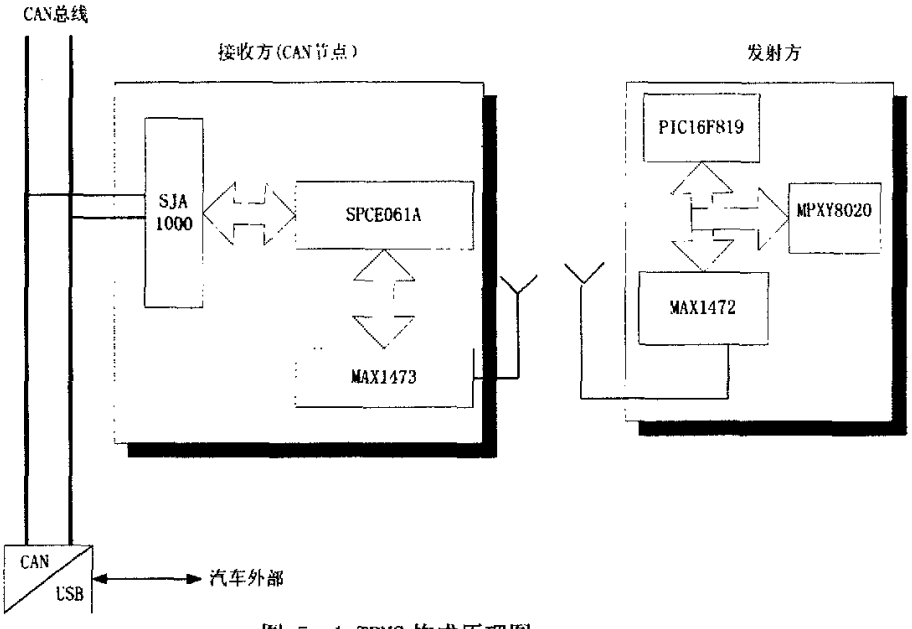

图 5－1 TPMS 构成原理图

5．2 TPMS 的发展现状

TPMS[41] 是汽车轮胎压力监视系统“Tire Pressure Monitoring System”的英文缩写形式，主要用于在汽车行驶时实时的对轮胎气压进行自动监测，对轮胎漏气和低气压进行报警，以保障行车安全。

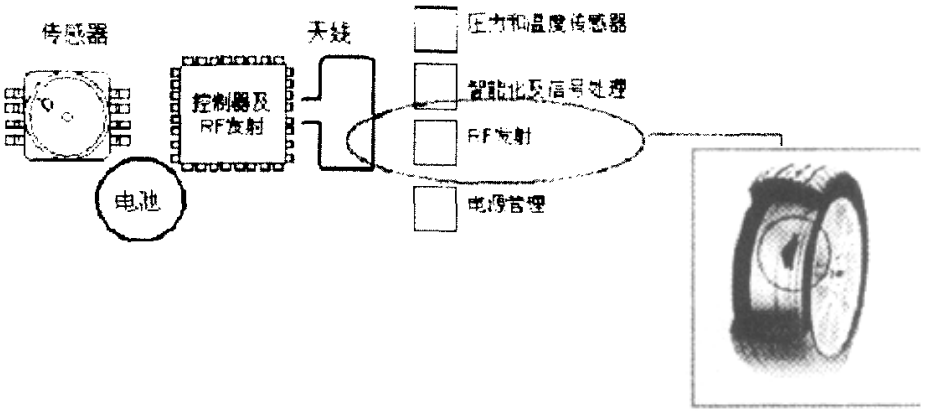

图 5－2　TPMS 结构

目前，TPMS 主要分为两种类型[42]，一种是 Wheel-Speed Based TPMS（简称：WSB TPMS，或称为间接式 TPMS ），这种系统是通过汽车 ABS 系统的转速传感器来比较轮胎之间的转速差别，以达到监视胎压的目的，该类型系统的主要缺点是无法对两个以上轮胎同时缺气的状况和速度超过 100 公里/小时的情况进行判断。另一种是 Pressure-Sensor Based TPMS(简称:PSB TPMS，或称为直接式 TPMS)，这种系统是利用安装在每一个轮胎里的压力传感器来直接测量轮胎的气压，并对各轮胎气压进行显示及监视，当轮胎气压太低或有渗漏时，系统会自动报警。PSB TPMS 从功能和性能上均优于 WSB TPMS。

国内多数汽车厂家目前还没有进行这方面的研究，随着中国经济的持续发展，汽车越来越多地进入普通家庭， TPMS 在国内的应用前景是十分广阔的。

5．3 数据采集部分设计

5．3．1 器件选择

汽车胎压检测系统作为保障汽车安全的部件，在汽车的整体构造中占有十分重要的地位，这也是作者选用 TPMS 作为论文内容的原因。

汽车胎压检测系统由于位于汽车轮胎内部，使得其设计思路由许多不同于其他设计的独到之处。例如：胎压检测系统一旦安装人车胎内部，便不再取出来，直到最终随轮胎一起报废，这就使得系统在功耗、能源供应、数据传输、对恶劣工作环境的承受能力等方面必须要作特别的考虑。因此，胎压检测系统采用了“无线数据发送、低功耗、小体积、稳定性好”的设计思路。根据以上设计思路，在选择相应的器件时，首先要考虑最大限度的降低功耗，然后到达比较高的稳定性，接下来尽量缩小体积。为此，本系统的数据采集

部分选择了 Microchip 公司高性价比的 8 位微控制器 PIC16F819、Motorola 公司的温度压力传感器 MPXY8020A。

PIC16F819[43]采用 20 脚贴片封装，在线方式进行程序调试、烧写；可采用内部晶振，工作频率可选 31K－8MHz；工作电压从 2V－5.5V；基本工作时功耗：内部晶振、2V 电压、32KHz、7uA；睡眠状态、2V 电压、0.7uA。其管脚分部如图 5－3 所示。

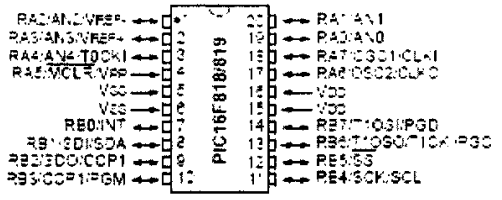

图 5－3 16F819 管脚图

MPXY8020[44]共有八个管脚，采用贴片封装，管脚分部如图 5－4 所示。其中 s1, s0, clk, data 为输入脚，/rst, out 为输出脚。s1, s0 组合控制工作模式，clk, data 改变压力的门限值，/rst 每 52 分钟输出一次复位信号，out 每 3 秒钟输出一次唤醒信号。MPXY8020 有四种工作状态：空闲态、测量压力、测量温度、输出。其中测量压力最耗电为：1.5mA，持续时间 500us。

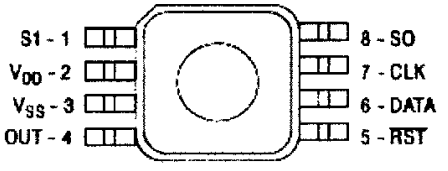

图 5－4 MPXY8020 管脚图

PIC16F819 同 MPXY8020A 的硬件连接原理图如图 5－5 所示。

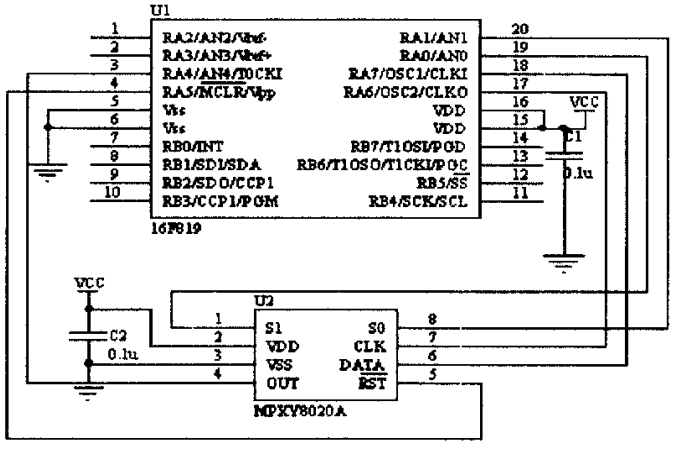

图 5－5 数据采集部分原理图

5．3．2 软件流程

压力温度传感器MPXY8020A测到的数字化的压力值和温度值并不是直接由MPXY8020A的管脚输出的，而是CPU通过特定的测量程序获得的。该测量程序的流程如图 5－6 所示。在程序的开头首先定义两个变量，一个是比较值 temp 并付初值 0x80，一个存放结果 result 并附初值 0。程序开始，将 temp 同 result 相或后送 MPXY8020。然后读 MPXY8020 的 out 脚的状态，如果是 0 则为 high，将 temp 右移一位；如果是 1 则为 low，将 result 同 temp 相或，再将 temp 右移一位。然后程序从头开始，如此循环 8 次，得到的 result 的值便是测量到的压力的值。

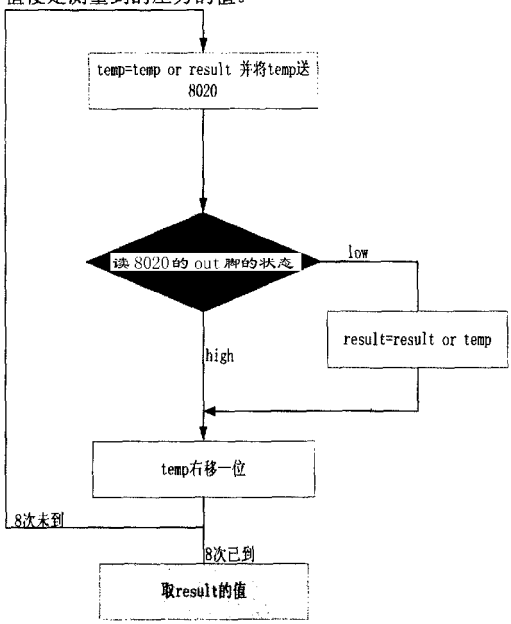

图 5－6 MPXY8020 工作流程图

MPXY8020 测量压力的一个完整的工作流程为：

1）置 S0、S1 为 00，使 MPXY8020 进入空闲态，通过 Clk，Data 向 MPXY8020 输入任意值清空其内部计数器。

2）置 S0、S1 为 10，使 MPXY8020 进入测量压力态。

3）150ms 后，置 S0、S1 为 11，使 MPXY8020 进入输出态，调用上述的测量程序，获得数字化的压力值。

测量温度的过程与此相同。

5．4 数据传输部分设计

5．4．1 硬件设计

硬件部分由发射部分、接收部分构成。发射部分位于汽车轮胎内部，接收部分是一个 CAN 节点，同 CAN 总线相连。

1．发射方硬件构成原理

发射器件使用 MAXIM 公司提供的超外差 ASK 调制射频发射芯片 MAX1472[45]。其外部只需很少的器件便可构成射频发射机，所以其使用比较简单，且发射功率为 10db 功耗低，适用于各种对功耗有特殊要求的器件使用。与 MAX1472 配合使用的 CPU 选择 Microchip 公司一款高性价比的 8 位微控制器 PIC16F819。MAX1472 的主要管脚的功能如表 5－1 所示。

表 5－1 管脚说明

XTAL1	晶振连接脚 1
GND	芯片地
PAGND	放大器地
PAOUT	放大器输出
ENABLE	发射使能
DATA	发射数据输入
Vdd	芯片电源
XTAL2	晶振连接脚 2

其中 Enable 管脚为高电平，则 MAX1472 进入发射状态。DIN 管脚输入串行数据。XTAL1 和 XTAL2 连接不同的晶振，用于提供不同的载波频率。PAOUT 连接由 LC 构成的匹配网络。

发射的载波频率由 MAX1472 所用的晶振决定，晶振同发射载波频率间的关系是：$f_{XTAL}=32\times$晶振频率

如果选择的发射载波频率为 315MHz，则所使用晶振为 9.8437MHz；如发射载波频率为 433MHz，则所使用晶振为 13.56MHz。

MAX1472 使用的天线要符合 50 欧阻抗匹配，天线可以采用 1/4 波长的鞭状天线，也可以使用 PCB 布线作为天线。

另外，对于高频而言，其难点在于电路板的设计。针对高频板，在设计时必须注意一下几点：

1）所有的电源旁边都要加防抖动电容。

2）电源和地的布线要尽量短，线要宽，所有的地线直接同电路板背面的铺铜层相连，不要走线。

3）天线的长度时波长的 1/4，本系统采用 433MHz 的发射载波，所以其波长约为 69 厘米，而天线的长度为 140 毫米。在布天线时，天线的背面不能铺

铜，天线布线的宽度为 2 毫米。

4）发射端要求阻抗匹配，发射频率由 LC 振荡产生，所以在初次调试的时候，使用可调电容以达到理想的载波频率。

发射方硬件原理图如图 5－7 所示。

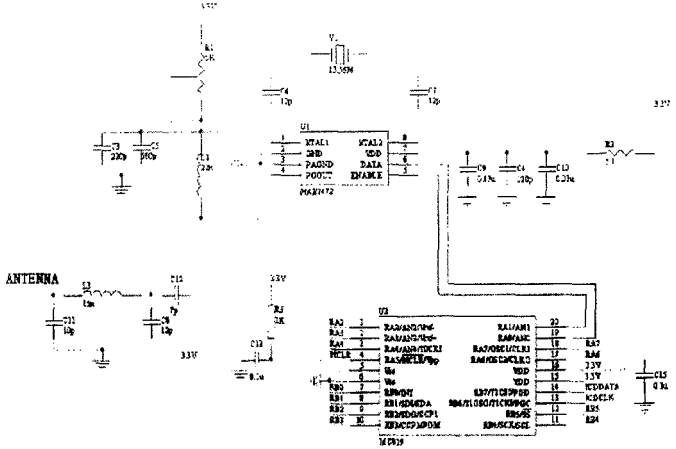

图 5－7　发射端原理图

2．发射方能耗问题的解决

发射方作为一个整体的话，就必须考虑其能耗。发射方一旦安装进车胎内部就不会再取出，直到和车胎一起报废。如何使发射方能构在车胎内工作很长的时间（一般 5－8 年），是本系统设计过程中的一个难点。

目前经常使用的低功耗设计方法主要有以下几种：

1．使用离心开关，发射装置在汽车运行过程中才开始工作，汽车停止便停止工作；

2．软件设计，使 CPU 在常态处于休眠状态，工作时才从休眠状态唤醒。

3．选择低功耗芯片。

经过多方面的综合考虑，包括成本、实用性等方面，最终采用了可更换电池的方案。本系统将标准的汽车车胎打气装置设计为 TPMS 的一部分，在安装的时候，将 TPMS 安装在车胎钢圈的打气孔处，温度压力传感器和 RF 发射部分安装到车胎的内部，打气嘴同时也是电池接口露在车胎外部，打气嘴的盖子同时也就是电池。这样一来，TPMS 在使用到一定时间后，如一年左右，便可以方便的更换电池了。

3．接收方硬件构成原理

接收方是一个 CAN 节点。同样使用 MAXIM 公司提供的 ASK 调制的超外差射频接收芯片 MAX1473[46]，CPU 使用 SPCE061A。061A 要完成接收数据还原，

控制 SJA1000 发送 CAN 数据的功能。MAX1473 的需要外接一个 10.7MHz 的低通滤波器，晶振的频率同接收载波的频率的关系为：

$f_{RECEIVE} = (f_{XTAL} \times 32) + 10.7MHz$

如果接收载波的频率为 315MHz，则晶振的频率为 9.509MHz；接收载波的频率为 433.92MHz，则晶振的频率为 13.2256MHz。

MAX1473 的主要管脚如表 5－2 所示。

表 5－2 MAX1473 管脚说明

DATAOUT	接收数据输出
LNAOUT	放大器输出
XTALSEL	晶振选择
/PWRDN	芯片工作状态选择
LNAIN	放大器输入
IFIN+	10.7MHz 差分低通滤波器输入
MIXOUT	混频器输出
AGND	模拟地

其中 IFIN+、MIXOUT、AGND 连接 10.7MHz 的低通滤波器，用于 ASK 数据的解调。XTALSEL 用于选择接收的载波频率是 433.98MHz 还是 315MHz。DATAOUT 输出经过 ASK 解调后的数据方波。LNAIN 连接接收天线，输入经过 ASK 调制的数据波形。

MAX1473 同样要求 50 欧阻抗匹配的天线，可以使用 PCB 布线作为天线 MAX1473 只需要很少量的外围器件就可以工作，接收方硬件构成原理图如图 5－8 所示。

5．4．2 软件实现

1．发射方

发射部分的编程主要是对 PIC16F819 的操作。程序由 PIC16F819 初始化、数据发射两部分组成。

对 PIC16F819 的初始化包括设置内部晶振的工作频率、PA 口的状态。数据发射部分包括：设置数据帧格式、发送位‘0’或‘1’子程序、发送一个字节数据子程序和循环发送多个字节数据子程序。程序流程图如图 5－9 所示。程序工作原理：16F819 向 MAX1472 发送数据是串行进行的，发送‘1’即将 I/O 口置高，发‘0’即将 I/O 口置低。编程时首先编写将 I/O 口置高和置低的子程序，然后由置高、置低子程序构成发送一个字节数据的子程序，再接着使用循环语句不断调用发送一个字节数据的子程序，直到将一帧数据发送完毕。

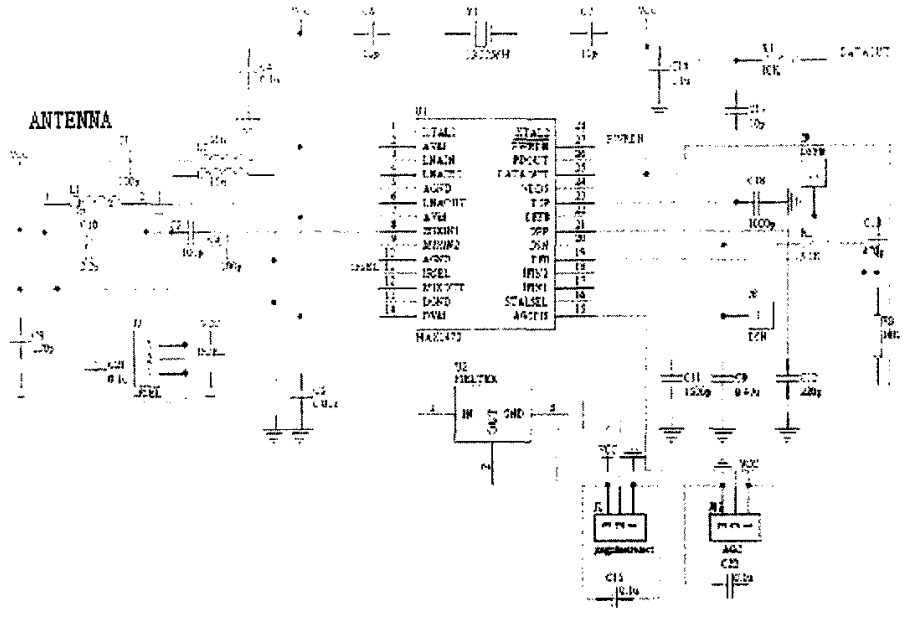

图 5—8 接收方原理图

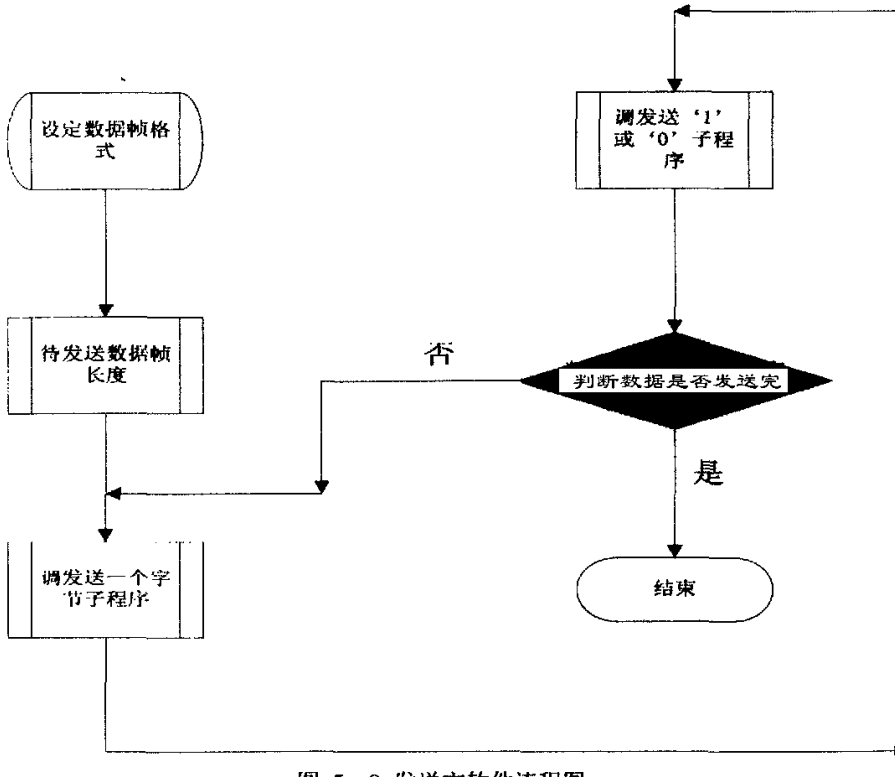

图 5—9 发送方软件流程图

2．发送帧的结构

发送帧由帧头、设备 ID、压力值、温度值、效验和、停止位构成。其结构如图 5－10 所示。

帧头	设备 ID	压力值	温度值	效验和	停止位
8bit	32bit	8bit	8bit	8bit	2bit

图 5－10 发送帧结构

帧头 8 位用来确认收到的帧是我们想要的帧，而不是任意帧。

设备 ID32 位用来区分本车上的四个轮胎和其他车上的轮胎。

压力值就是 MPXY8020 测量到的压力值。

温度值就是 MPXY8020 测量到的温度值。

效验和是用来确认数据帧在发送的过程中是否被破坏，即收到的帧值是否正确。

停止位用来表示该数据帧已结束。

3．接收方

接收部分程序主要是微控制器通过 DATAOUT、/PWRDN 等管脚对 MAX1473 进行控制。/PWRDN 管脚为低，MAX1473 进入工作状态，从 DATAOUT 管脚输出经过 ASK 解调后的代表‘1’‘0’的方波，CPU 需要做的工作就是将方波还原成数字量。

CPU 对 MAX1473 输出的方波进行还原的程序的工作流程图如图 5－11 所示。

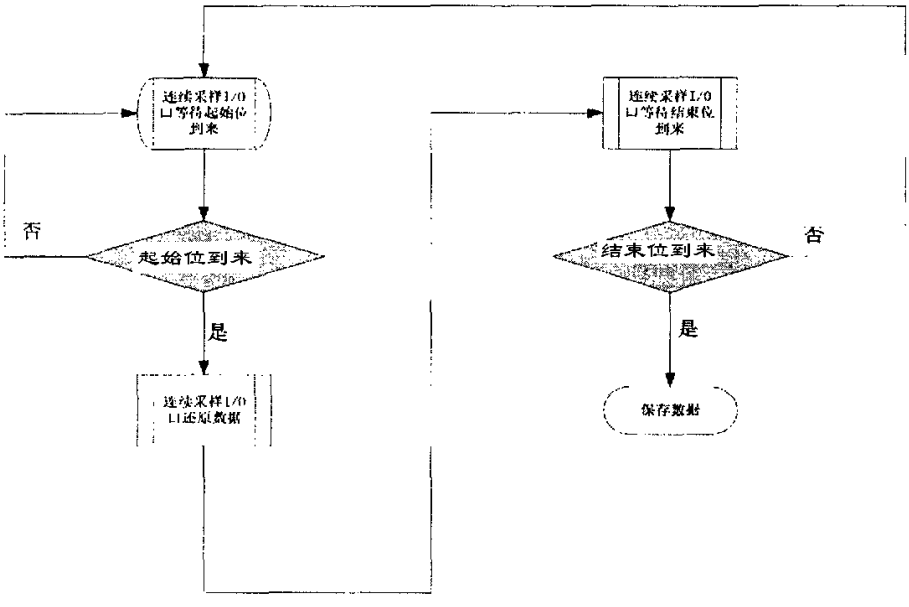

图 5－11 接收方软件流程图

程序工作原理：本程序中使用的是重复采样，即使用 10 倍与数据传输速率的采样速率对 I/O 口到来的方波进行采样。如果连续采样的 5－8 个数

据都是 1，则认为到来的数据是 1；如果连续采样的 5－8 个数据都是 0，则认为到来的数据是 0，以此来将接收到的波形还原成数字量。本程序中数据的帧格式是 1 字节起始位、2 字节数据位、1 字节停止位。

TPMS 设计中有两大难点，一是高频板设计，二是能耗问题的解决。本系统在能耗问题上，采用了一种低成本的解决方案——电池外置，更换电池。轮胎压力和温度数据通过无线进行发送时，要区分同一辆车上的不同轮胎，还要区分不同车上的轮胎，另外还要能抗拒汽车工作过程中的其他干扰。

5．5 应用层软件设计

5．5．1 软件界面设计

应用层软件的工作是对从 USB 口传输过来的数据进行处理，将最终的分析诊断结果以图形、数字或表格的形式表现出来。应用层软件采用 DELPHI7.0[47]配合 Pascal 语言编写，DELPHI 以面向对象和易于设计各种界面而见长[48]，软件 TPMS 部分的界面如图 5－12 所示。

界面分为四个部分，每个部分代表一个车胎的状况。车胎的状况分为压力和温度两个数据指标，两个指标分别都用进度条和数字显示出来，如果数据经过分析后得到的结果是压力、温度正常，则绿框闪动并显示正常字样，否则红框闪动并显示报警字样。

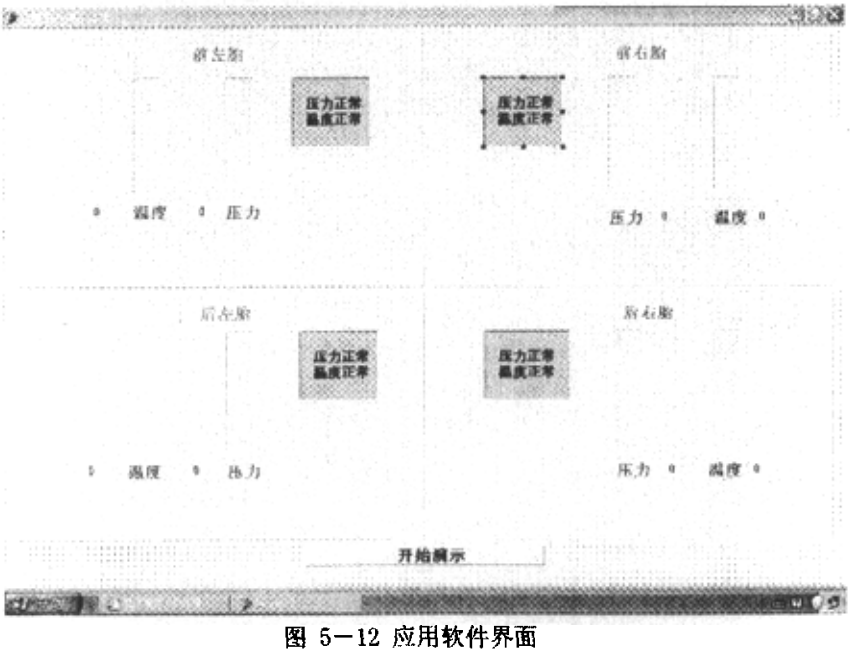

图 5－12 应用软件界面

5．5．2 系统应用协议的实现

应用程序符合通用故障诊断代码的要求。笔记本内的故障诊断程序同车载信息系统通讯使用 CAN/USB 网关在 CAN 数据格式和 USB 数据格式间进行转换。CAN 的帧格式除了数据帧，另外还包括三种命令帧：远程帧、错误帧、过载帧。

针对汽车轮胎压力检测来说，在数据帧中，仲裁场内的 29 位 ID 号用于区别本车的四个轮胎；控制场内的 DLC0-DLC3 决定 CAN 数据帧的数据场的长度，DIV、ED 决定封装在 CAN 数据帧内的一个完整数据帧是否拆分和是否发送完毕；数据场用于存放经过拆分的完整数据帧；效验场使用硬件判断数据帧在传输过程中是否损坏。

轮胎压力检测系统的发射端，按照图 5－10 的数据帧格式将测量到的压力、温度值发送给接收端。接收端收到数据帧后，经 CPU 处理后，得到压力和温度值，并将得到的数值同正常值进行比较。根据比较所得结果，按照表 4－2 所示 16 进制故障代码，向 CAN 总线上发送 CAN 数据帧。表 4－2 中，A0-B9 部分的代码是由汽车制造商定义的，这部分代码便可用来进行故障诊断。发送到 CAN 总线上的 CAN 数据帧，通过 CAN/USB 网关后，转换成 USB 数据格式，经 USB 接口传输到笔记本上的故障诊断程序。

故障诊断程序首先根据 CAN 数据帧的 CRC 场，判断数据帧是否正确；然后读取 CAN 数据帧的仲裁场的 ID 号，判断该数据帧是来自哪个轮胎；再然后读取控制场，判断此数据帧内的数据帧是否进行了拆分，及此帧是否是最后一帧；最后读取数据场内的偏移量和数据帧，如果数据帧进行了拆分，根据偏移量指定的位置对数据帧进行重组。

获得了完整的数据帧后，便可以读取数据帧内的数据，根据不同的代码，确定故障的类型和位置。

汽车胎压检测的工作流程如图 5－13 所示。

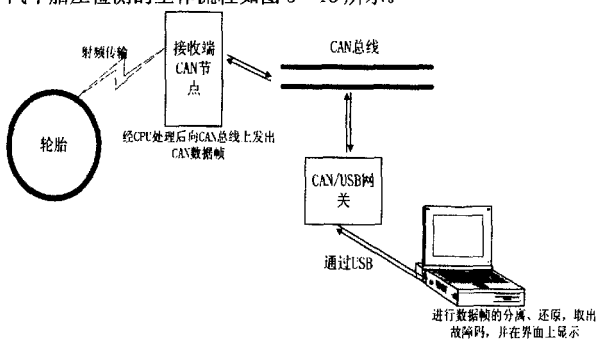

图 5－13 工作流程图

5．5．3 软件工作流程

汽车胎压检测部分故障诊断程序的工作流程如图 5－14 所示。

汽车上的各个轮胎作为 CAN 节点，通过 CAN 总线相连，并根据 CAN 协议被赋予不同的物理地址。当要获得某个轮胎的实时数据时，先发送命令帧指向这个部件，然后轮胎节点再发送应答帧回应相应的数据。CAN/USB 网关在发送命令时，负责将 USB 数据格式转换成 CAN 数据格式，让 SJA1000 能够识别；在接收应答帧时，负责将 CAN 数据格式转换成 USB 数据格式，让应用程序能够识别。应用程序从接收到的 USB 格式应答帧内部，分解出 16 进制故障代码，根据不同的代码在界面上显示不同的状态，如：压力、温度的进度条，绿框闪动或红框闪动。

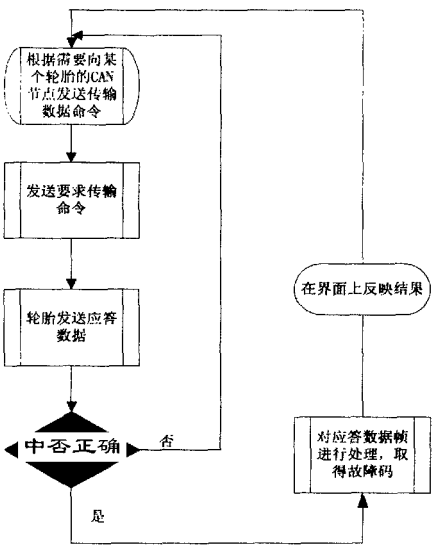

图 5－14 应用软件流程图

第6章 总结

现代汽车正在向智能化、通用化的方向发展，车载信息系统将成为汽车上的造价最高、最核心的标准部件，最终其内部总线将由现在的多种总线共存形式发展为单独的CAN总线形式，汽车故障的诊断和维修将由现在的以人工为主发展到以故障自诊断、自维修为主。

本文论述的基于USB的车载故障诊断系统，取消了其原有的专用故障诊断网络K线和L线，取而代之的是未来汽车上通用的CAN网络，并且使用PC机上通用的USB接口取代KWP2000协议规定的专用诊断接口(SAE2J1962)，使得汽车内部的网络进一步简化，故障诊断仪器不需要第三方开发，而是使用普通的笔记本电脑加上故障诊断程序来取代。该系统适应了汽车发展的主流趋势，简化了现有汽车的内部网络，随身携带的笔记本电脑上的故障诊断程序可以实时的反映汽车的工作状态，及时对异常做出反映，将故障在最初出现时便给予排除，把对汽车造成的损害降低到最小的程度。

参考文献

[1] 冯静. 基于KWP2000的电控发动机标定工具开发. 内燃机工程，2002.4. 1～4
[2] 宋国民. 基于K线的电控系统诊断平台开发. 现代车用动力，2004.5. 1～4
[3] 许峰. 汽车故障自诊断与故障诊断仪研究. 电子技术应用，2000.1. 25～28
[4] Titus. J. The CAN makes small networks easy. ECN. Vol.48, no.3, pp.69～70. Mar. 2004
[5] LIN规范 V1.2. http://www.zlgmcu.com
[6] 张蕾，董恩国，李泳鲜. 电子信号在汽车故障诊断中的应用. 汽车技术，2002.6. 38～42
[7] 胡玲. 汽车故障诊断专家系统软件的设计与研究. 电子技术应用， 2000.3. 17～19
[8] 董久悦. 轿车电控系统的检修. 黑龙江科学技术出版社，2004.2 105～138
[9] 刘丁. USB在数据采集系统中的应用. 电子技术应用，2000.4. 37～39
[10] 刘炎. 通用串行总线(USB)原理及接口设计. 电子技术应用，2000.12. 56～57
[11] Universal Serial Bus Specification Revision 1.0. Compaq DEC IBM Intel Mecrosoft NEC Northern Telecom, 1996.1.15.
[12] Lynn, Kevin. Universal Serial Bus (USB) power management. WESCON CONF REC, 1998. 194～201.
[13] Wong.T, Understanding USB. Electronics World, 1999.11. 890～900
[14] Insam.E, USB oscilloscope for the PC. Electronics World, 2002.5. 54～58
[15] Song.Y Q, Simonot-Lion. F, Belissent, P. VACANS. a tool for the validation of CAN-based applications, 1997.10. 381～390
[16] 张增强，武向辉. Delphi6入门与提高. 人民邮电出版社，2002.6. 1～261
[17] LNS DDE Server User's Guide. ECHELON Corporation, 2001. 10～154
[18] 王远. 别克(BUICK)轿车发动机故障诊断. 汽车技术，2003.10. 38～41
[19] The calculation method of fiber-CAN bus network node number. Vol. 49, no.5, 2003.11. 629～632.
[20] Fonseca. J A, Almeida. L M. Using a planning scheduler in the CAN network, 1999.10. 815～821.
[21] 顾洪军. 工业企业网与现场总线技术及应用. 人民邮电出版社，2002. 13～55
[22] Ekiz. H, Kutlu. A, Baba. M D, Powner. E T. Performance analysis of a CAN/CAN bridge, 1996.10. 181～188.
[23] 张继东，张卫东，许晓鸣. 现场总线与以太网互联研究. 计算机工程，2002.5. 19～21
[24] 刘越琪. CAN总线在雪铁龙C5轿车中的应用及检修. 汽车技术，2003.11. 36～39
[25] 侯树梅，张云龙，苏剑. 一种新型汽车车身低端通讯总线LIN. 汽车技术，2003.11. 5～8
[26] LIN介绍. http://www.zlgmcu.com
[27] OBD-II规范. http://www.obdii.com/
[28] 宋国民，季晓华. 基于K线的电控系统诊断平台开发. 现代车用动力，2004.5.

[29] SJA1000 datasheet. http://www.zlgmcu.com/philips/philips-can.asp, 1～12
[30] 韩党群. CAN控制器SJA1000及其应用.电子技术应用，2003.1. 66～69
[31] PDIUSBD12 datasheet. Philips Corporation 1～35
[32] 李晶皎. 凌阳16位单片机应用.北京航空航天大学出版社，2003.11 149～210
[33] Insam.E, USB oscilloscope for the PC.Electronics World, 2002.5. 54～58
[34] Data acquisition has USB future. Control and Instrumentation, 2000.4. 20～21
[35] BOSCH_CAN_V20_cn. BOSCH Corporation 1～30
[36] 周立功. USB固件编程与驱动开发.北京航空航天大学出版社，2003.2. 1～40
[37] Jean J.Labrosse. MicroC/oS-II The Real-Time Kernel, 2003.5. 73～396
[38] Joseph.C. Personal Computer World. Universal solution, 1999.1. 303～324
[39] SSF 14230 Road Vehicles - Diagnostic Systems
[40] SSF 14230 Road Vehicles - Diagnostic Systems. Swedish Recommended Practice, 1998.6
[41] 韩建保. 汽车轮胎气压电子实时监测系统. 汽车技术，2002.7. 1～3
[42] 未来汽车轮子上的信息世界. 金属世界，2003.1.
[43] PIC16F819 datasheet. Microchip Corporation 1～24
[44] MPXY8020A datasheet. Motorola Corporation 1～32
[45] MAX1472 datasheet. MAXIM Corporation 1～17
[46] MAX1473 datasheet. MAXIM Corporation 1～18
[47] 伊立民. Delphi7开发实例.电子工业出版社，2002.11 39～243
[48] 吴炜煜. 面向对象分析设计与编程. 清华大学出版社，2000. 103～153

攻读硕士学位期间发表的学术论文：

[1] 廖传书，韩屏.基于USB的无源身份认证的实现。微机发展，2004.11
[2] 廖传书，韩屏.基于LabVIEW的USB数据采集系统的实现。工业控制计算机,2004.7
[3] 廖传书，韩屏.基于智能控制的汽车点火检测平台的设计。微计算机信息，2005.8

致谢

本论文是在导师廖传书副教授的悉心教诲和无微不至的关怀下完成的。从论文的选题，论文的构架及写作等方面，廖老师都给予了极大的帮助。在研究生学习期间，廖老师不仅授业，更是传道、解惑；始终给予我严格的要求、充分的信任、热情的鼓励和全面锻炼的诸多机会。在学习和学术研究中，我深受廖老师严谨的治学态度和勤奋的敬业精神的熏陶和激励，从中收益良多；在生活和为人方面，廖老师更是言传身教，勉励我们积极上进；在此，谨向廖传书老师表示最诚挚的谢意。

此外，在论文的写作过程中还受到了黄涛、卢珞先、郑建彬等多位老师的无私帮助，在此，对他们表示最诚挚的感谢。

我的父母和家人支持并期待我完成学业，并为此承担了很多的生活压力。在论文得以顺利完成之际，我愿与他们分享其中的甘苦，并感谢亲人们对我的殷切期望和无私无悔的支持。

感谢所有关心和帮助过我的师长、同学、朋友和亲人们！

作者：韩屏

2005 年 5 月 22 日