摘　要

汽车防抱死制动系统(Anti-lock Braking System，简称ABS)是一种主动安全装置，它能够提高汽车的安全性能，减少交通事故的发生率，现在已经成为了许多车辆的必备装置。近年来我国先后出台了相关法规与标准，强制要求相关的客车与货车汽车安装ABS。现在我国的ABS研究速度依然落后于国外，其核心技术包括控制算法以及具体实现的软件和硬件条件也同样依赖国外产品，所以研发一种高效的具有完整知识产权的ABS系统对于中国汽车工业走向成熟是必要的。本论文主要是研究开发基于嵌入式Linux的燃料电池电动公交车ABS系统。具体开展了以下几个方面的研究工作：

从控制器实际设计及实施的角度研究了三种比较实用的ABS控制算法，即传统的逻辑门限控制、PID控制和模糊控制。首先论述了三种控制方法的原理，然后采用国际流行的MATLAB/SIMULINK软件对三种控制方法作了模拟研究，并对模拟结果作了分析与评价。

通过搭建嵌入式ABS系统，为控制算法提供了应用平台，并深入研究了电动公交车ABS系统的组成与结构，同时，根据电动公交车的实际工况要求，设计了控制单元的电路；剖析了嵌入式Linux操作系统的移植过程，包括Boot Loader的移植、Linux内核的移植、文件系统的移植、驱动程序的设计。

针对汽车防抱死制动系统(ABS)的强非线性，利用前面对于3种普遍应用的ABS控制算法的分析与模拟的结果，采用基于滑移率的模糊控制策略建立了ABS的仿真模型。通过实际ABS系统部分的设计，其中主要包括数据采集模块，控制模块，执行模块，最后，以实际工况的数据进行模拟，得到较满意的结果。

基于ARM9微处理器和嵌入式Linux操作系统开发了燃料电池电动公交车ABS系统。进行了ABS系统设计，描述了实现电子控制单元中硬件电路和软件逻辑的设计过程。通过试验，验证了系统的有效性和合理性，为电动公交车的主动安全控制装置集成化研究提供了有利的依据，也为今后在此课题基础上的构建基于嵌入式系统的整车集成控制奠定了积累相关的技术材料。

关键词：电动公交车，ABS，Linux，模糊控制

Research of ABS in Fuel Cell Electrical Bus

Abstract

Automobile Anti-lock Braking System is an active safety device.It can raise the rate of the safety performance and reduce accident . Now ABS becomes a very necessary device equipped in many vehicles. In recent years some interralated legislation has been compulsively drawn up to call for the setting of ABS on bus and wangon. In China,the research of ABS is still behind that of other country,and the core technology inside ABS including control algorithm and the realization of hardware and software depends on foreign product. What the paper focuses on is the basis of development of the fuel-cell electric bus ABS system based on embedded technology. Detail and main conclusion are showed as follows:

Studied 3 kinds of more practical ABS control methods from the physically design and implement of controller, namely traditional control of the logic threshold, PID control and fuzzy control. Discussed them and come up with the right way, then adopted MATLAB/ SIMULINK software to simulate these 3 kinds of control methods, and evaluate the imitating result.

According to build ABS based on embeded system, provides an applied environment for control methods, furthermore studied the constitute and structure of ABS system on electric bus, according to the real condition, designed the circuit of each unit; and analyzed the transplantation process of embeded Linux deeply, including the transplantation of Boot Loader, kernel of Linux, file system, the design of driver.

Aiming at strong non-line of Anti-lock Braking System (ABS), according to the results of above annlysis about 3 kinds of control methods, By the whole design of ABS, mainly including data collection module, control module, execution module, and the simulation with data in practical situation ,received a relative ideal result at last.

Developed ABS system of the fuel cell electric bus according to ARM9 microprocessors. Carried on the design of ABS system, described the process that carrys out the design of hardware circuit and software logic in the unit. Through feasible test, verified the usefulness and rationality of system, provide a beneficial basis for the research that the active safety

control of electric bus will be integrated in future, and accumulate the technology material for building up the whole integrated control system based on embedded system after this project in the future.

Key Words: Electric Bus , ABS, Linux , Fuzzy Control

独 创 性 说 明

本人郑重声明：所呈交的论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写的研究成果，也不包含为获得沈阳工业大学或其他教育机构的学位或证书所使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中做了明确的说明并表示了谢意。

签名：_____ 日期：_____

关于论文使用授权的说明

本人完全了解沈阳工业大学有关保留、使用学位论文的规定，即：学校有权保留送交论文的复印件，允许论文被查阅和借阅；学校可以公布论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存论文。

（保密的论文在解密后应遵循此规定）

签名：_____ 导师签名：_____ 日期：_____

1 绪论

1.1 课题来源

本学位论文基于沈阳骐骥电动车研究所合作项目“新型燃料电池电动公交车”，主要是研究汽车防抱死制动系统ABS(anti-lock brake system)在燃料电池电动公交车上的具体实现。

1.2 课题背景

1.2.1 燃料电池电动汽车的发展

近年来，世界石油资源逐渐减少，国际油价持续攀升，电动汽车的发展日益受人关注[1]。燃料电池汽车，即以燃料电池作为动力系统的汽车，具有节能、转换效率高、不需要石油燃料、排放达到零污染、车辆性能接近内燃机汽车、结构简单及运行平稳的优点，近年来成为各国研究的焦点[2]。世界各大汽车公司已相继推出以甲醇或汽油为燃料的燃料电池汽车，但在关键技术上还没有取得全面突破。从20世纪50年代开始，中国一直进行燃料电池相关技术的研究，但直到20世纪90年代，全球环境署支持在中国进行燃料电池公共汽车示范，中国对其产生了浓厚兴趣。从那时起，中国在此方面有了很大进步，且有些技术达到世界先进水平[3]。国内很多科研机构和汽车企业纷纷研究以燃料电池为动力的客车、小型轿车和公共汽车，他们希望能够在未来的10年内将燃料电池车投入使用。

目前在我国，应用ABS系统的公交车很少，主要是因为公交车整体价格相对较低，车内辅助配套设施少，加之ABS系统成本高。城市公共交通车辆及其运行组织管理的优劣，已是一个国家一个城市的发达与文明程度的标志。本课题主要是在燃料电池电动公交车的相关技术还相对处于不成熟的阶段的前提下，基于沈阳骐骥电动车研究所合作项目“新型燃料电池电动公交车”，研究开发一套适合这种公交车的ABS系统。

1.2.2 汽车防抱死制动系统ABS技术的起源与发展

20世纪初，原始的ABS装置就安装在铁路机车上，用于减少车轮的磨损。在30年代，机械式ABS开始在火车和飞机上应用，德国博世(Bosch)公司在1936年第一个获

得通过电磁式车轮转速传感器得到车轮转速的制动防抱死专利权，这是ABS系统发展史上的一个里程碑[4]。

在第二次世界大战的末期，ABS装置被用于喷气式飞机上，目的在于飞机着陆时，防止车轮抱死和严重磨损，并保持直线行驶性能。50年代末期，Good Year等公司开发出的防抱制动装置，根据车轮的减速情况，阶段性地控制液压，并采用了初期的电子计算机，使得ABS的性能得到了很大的改善，而后随着汽车动力性能的不断提高,人们对汽车的制动安全性能提出了越来越高的要求. 研究人员对制动器不断改进，以此保证为速度不断提高的汽车提供充足的制动力. 然而,由于汽车运动状态的复杂性和车轮与地面的之间附着力的非线性等原因，当汽车在高速行驶中制动或弯道上紧急制动以及在冰雪路面等复杂路况下制动时，经常出现车轮因抱死拖滑而导致制动距离加长,同时还有车身侧滑调头、失去方向控制能力等现象,严重威胁汽车、乘客和行人的安全。1954年美国Toad公司首次把民航机上的ABS应用在林肯牌轿车上，这次试验虽然以失败而告终，但揭开了汽车应用ABS的序幕。鉴于当时的电子工业和机械工业的发展水平所限，ABS的控制性能和可靠性较差，而且价格很高。

随着电子技术的发展，在60年代后期和70年代初期，一些电子控制的制动防抱系统开始进入产品化阶段。这一时期的ABS系统采用的控制器是模拟电路，驱动装置为电磁阀，控制系统反应速度慢，控制精度低，未达到预期的控制效果[5]。进行70年代后期，数字式电子技术和大规模集成电路的发展为ABS的实用化奠定了基础。Bosch公司在1978年开发出数字式电子控制ABS，揭开了现代防抱制动系统发展的序幕。自从80年代中期以来，ABS控制器不断更新换代，体积越来越小，功能越来越强，控制逻辑更加合理，并具有自适应性和故障自诊断功能，ABS向着提高性能成本比的方向发展。90年代以后，ABS技术已日趋成熟，制造成本不断降低，使得ABS迅速普及。目前，最著名的ABS开发公司有Bosch, ITTTeves、Delco和Lucas等[6-7]。

目前，在美国、西欧、日本等发达国家和地区，ABS已经成为轿车的标准设备，装车率达到100在大型客车和货车上，ABS的应用也日益普及。随着车辆动力学、计算机技术和电子技术的发展，ABS一方面向提高性能价格比方向发展；另一方面，与驱动防滑装置(ASR)相集成，并与主动悬架、电子转向控制等系统构成综合控制装置[8]。

我国对 ABS 的研究开始于 20 世纪 80 年代初，现在已进入产品试制和在车辆上试装的阶段，ABS 的研究项目被列入“九五”科技攻关计划。目前，国内 ABS 的研究单位也有不少，如:东风汽车公司、交通部重庆公司路研究所、重庆宏安 ABS 有限公司、陕西博华、西安公路学院和清华大学等。国产 ABS 的主要问题是路面识别不够理想、可靠性较差和性能价格比较差，增强国产 ABS 的竞争能力，迫切需要对 ABS 的关键技术(ECU 和控制软件等)进行深入的研究。

1.2.3 汽车 ABS 的分类与组成

ABS 的种类可分机械式和电子式两种，机械式的结构简单，主要是一个机械阀，利用阀体内一个橡胶气囊对刹车压力的反馈来不断放松，制动，从而达到轮胎不抱死的结果，目前一些国产皮卡和低档客车大部分采取了这种装置。工作原理简单，它没有传感器来反馈路面摩擦力和轮速等信号，完全依靠预先设定的数据来工作[9-11]。也就是说，不管车在积水路面，结冰路面或是泥泞路面和良好的水泥沥青路面，它的工作方式都是一样的。如果在水泥沥青路面行驶的话，制动效果就会大打折扣啦。

而电子式不然，它由轮速传感器，线束，ECU，ABS 液压泵，指示灯等等部件构成。其中 ECU 单元是 ABS 的核心，它能根据每个轮的摩擦力、转速、转弯角度、车身倾斜度等来发出相应的控制信号，由 ECU 分配刹车力度频率，能对每只轮施加不同的制动力度，从而达到科学合理分配制动力的效果。

机械式只是用部件的物理特性去机械的动作，而电子式是运用电脑对各种数据进行分析运算从而得出结果的。本课题是完全以电子式的 ABS 为研究基础，以 ECU（电子控制单元）为研究重点，把先进的嵌入式技术应用到电动公交车的制动系统上，从而实现适合不同路面的公交车 ABS 制动系统。

1.2.4 嵌入式系统在汽车 ABS 中的应用

目前，嵌入式系统(Embedded System)作为计算机应用的一个重要领域，已深入到社会的方方面面，不仅广泛渗透到社会、经济、军事、交通、通信等相关行业，而且深入到家电、娱乐、艺术、社会文化等各个领域，掀起了一场数字化技术革命。 嵌入式系统是以应用为中心、以计算机技术为基础、软件硬件裁剪、适应应用系统对功能、可靠性、成本、体积、功耗严格要求的专用计算机系统[12-13]。它主要山嵌入式微处理器、

相关支撑硬件、嵌入式操作系统以及应用软件系统等部分组成，集软硬件于一体，用于实现对其他设备的控制、监视和管理等功能。

32位嵌入式CPU价格的下降和性能指标的提高，为嵌入式系统的广泛应用提供了可能。那么，限制嵌入式系统发展的瓶颈就突出地表现在软件方面。尽管从上世纪八十年代末开始，已经陆续出现了一些嵌入式操作系统(比较著名的有 Vxwork、pSOS、Neculeus和WindowsCE等)，但这些专用操作系统都是商业化产品，其高昂的价格使许多生产低端产品的小公司望而却步；而且，源代码的封闭性也大大限制了开发者的积极性。嵌入式系统需要的是一套高度简练、界面友善、质量可靠、应用广泛、易开发、多任务，并且价格低廉的操作系统。如今，业界已经达成共识：即嵌入式Linux是大势所趋。 嵌入式 Linux 操作系统以价格低廉、功能强大、易于移植等特点而正在被广泛采用，并已成为一种新兴力量。

嵌入式系统同通用型计算机系统相比，具有以下特点：

（1）嵌入式系统通常是面向特定应用的，嵌入式 CPU 与通用型的最大不同就是，嵌入式CPU大多用在为特定用户群设计的系统中，它通常都具有低功耗、小体积、高集成度等特点，能够把通用CPU中许多板卡完成的任务集成在芯片内部，从而有利于嵌入式系统设计趋于小型化，因此，器件的移动能力大大增强，同时跟网络的耦合也越来越紧密[14]。

（2）嵌入式系统的硬件和软件都必须高效地设计，量体裁衣，去除冗余，力争在同样的硅片面积上实现更高的性能，这样才能在具体应用中对处理器的选择更具有竞争力。

（3）因为嵌入式系统和具体应用有机地结合在一起，它的升级换代也和具体产品同步进行，所以，嵌入式系统产品一旦进入市场，一般都有较长的生命周期。

（4）为了提高执行速度和系统可靠性，嵌入式系统中的软件一般都固化在只读存储器芯片或单片机之中，而不是存储于磁盘等载体中。

（5）嵌入式系统开发采用独特的宿主机—目标机模式，在这个环境下调试好口标机的硬件和软件，才能使目标机(最终的嵌入式系统)脱离开发环境，独立运行。

现代制动防抱死装置多是电子计算机控制，这也反映了现代汽车制动系向电子化方向发展。ABS（制动防抱死系统）的核心 ECU（电子控制单元）大多采用 MCU(微控制器)也就是单片机作为硬件基础，从未来的发展方向上看，为了实现 ABS 的快速响应、低成本、系统稳定性好、与 CAN(控制区域网络)通信效率高等特性，研究开发一款由高性价比的有稳定的实时性强的操作系统和处理速度快、多可扩充接口的微处理器构成的 ABS 是有必要的[15-16]。

1.3 课题研究的目的和意义

ABS 系统在保持汽车制动时的方向稳定性和有限度地缩短制动距离，提高汽车制动安全性方面作用明显，使得全世界对 ABS 的应用都非常重视。ABS 作为现代汽车的一项关键性技术，它具有广阔的发展前景。

目前国际上 ABS 在汽车上的应用越来越广泛，已成为绝大多数类型的汽车的标准装备，北美和西欧的各类客车和轻型货车 ABS 的装备率已达 90%以上，轿车 ABS 的装备率在 60%左右，运送危险品的货车 ABS 的装备率为 100%[17-19]。我国有许多轿车也已装备 ABS 系统，GB12676-1999《汽车制动系统结构、性能和试验方法》对汽车的制动性能要求严格，具有关于汽车制动时车轮不抱死的强制要求。我国汽车工业发展规划中把 ABS 技术开发应用列为第一条。但都未有自主的知识产权。

面对激烈竞争的国际和国内的汽车市场以及我国汽车工业的现状，研究汽车 ABS，可以完善对 ABS 在汽车应用上的认识，尤其是对电动公交车的上的应用。是为今后自主开发并实现沈阳骐骥电动车研究所的燃料电池电动公交汽车项目奠定了一定的理论基础。电动公交车是未来的城市中的主要交通工具，是很多人市内出行的首选，它的制动系统的可靠程度关系到很多乘客人身安全。因此，通过严谨的态度，广泛的查阅，深入的研究，细致的归纳，准确的数据最终完成对电动客车 ABS 刹车系统的研究将是具有重要的社会意义和经济意义的。

1.4 课题研究的主要内容

本研究内容主要包括以下几个方面:

（1）分析国内外 ABS 技术的当前情况，尤其是电动公交车上 ABS 中 ECU 的设计，剖析其整体优缺点。

（2）构建嵌入式 ABS 系统，为控制算法的实施提供有力的平台。对 ABS 制动系统主要组成部分的原理和模型进行的研究。分析 ABS 制动系统的特点、附着系数与滑移率的关系曲线对现有的几种 ECU（电子控制单元）的控制策略进行比较。分析它们各自的侧重点。

（3）研究电动公交车上 ABS 的设计方法；由于每款车的 ABS 各有各的特点，从电动公交车的车型，使用环境，用途，结合发动机、悬挂、变速箱、电脑管理系统等等综合考虑，如何提出适合电动公交车上安装的一套 ABS 设计方案。

（4）对防抱死制动系统 ABS 的电子控制单元 ECU 的故障监控功能进行研究。因为 ABS 都具有自诊断功能，能够对系统的工作情况进行监测，一旦发现存在影响系统正常工作的故障时将自动地关闭 ABS，并将 ABS 警示灯点亮，向驾驶发出警示信号，汽车的制动系统仍然可以像常规制动系统一样进行制动。所以，故障监控功能的设计对于 ABS 的维护是必要的。

（5）进行 Linux 在 ARM 平台上的移植，并通过 CAN 总线实现与电动车其它控制模块的通信。由于本车是基于 CAN 总线的分布式控制系统，ABS 系统需要通过总线进行数据采集，然后根据预先设置的控制策略判断汽车的刹车情况，从而得到相应的控制信号，又通过总线发送控制指令到各个执行单元。此外还有由自检产生的故障信息也要及时的传输到中央控制模块。

1.5 课题研究的主要难点和关键技术

（1）路面识别是一个重要的问题，这在于需要根据路面来确定加、减速度门限。不同的路面防抱特征是不同的，需要根据路面来确定合适的门限值，同时对参考速度的计算也要求有一个合适的参考减速度，而参考减速度是山路况来决定的。若增加对制动压力及制动速度的测量将使辩识路面变得比较容易，但这样对ABS产品是不实际的[20-22]。由于没有额外的传感器测试更多的参数，加之在防抱的过程中由于路面的不平度，信号的干扰等原因，防抱循环并不能象理想的那样形成完整的循环，所以利用现有的系统辩识路面是一个比较棘手的问题。

（2）ABS 控制策略的比较，还要与电动公交车的实际内部工作原理以及应用场合相联系，考虑采用何种控制策略解决问题最优。

（3）存在于测控系统内部的干扰，具有随机性，采用硬件抗干扰措施，只能抑制某些干扰，但仍有一些干扰会侵入系统而引起系统不时地出现一些功能性故障:如控制开关不起作用，产生误动作或不按程序设定的逻辑顺序动作，测试结果不能正常输出等。由于故障的特点是暂时、间歇和随机地，用硬件解决比较困难。因此，对于单片机测控系统来讲，除了采取硬件抗干扰方法外，还要采取必要的软件抗干扰措施。

（4）对于嵌入式系统设计过程的详细研究，包括：系统的移植与裁减，硬件的选配，程序的调试与编写等都将是未来要解决的难题。

1.6 课题研究的前景

ABS 系统和其他汽车部件的结合是发展方向。汽车控制是一个系统工程问题。例如，其底盘就包括制动，转向和悬架等子系统。这些子系统控制的简单叠加并不能获得良好的综合性能，因为许多性能指标是冲突的，所以存在整体最优化的问题。防抱死系统的控制成为汽车综合控制的一个方面。所以，ABS 研究工作需要与其他部件综合起来寻求整体优化。

ABS 系统控制与主动悬架系统(A-SUS)的综合，优点体现在消除干扰。主动悬架系统不仅能控制车辆转弯时的姿态变化，而且能调节前后轮的侧偏刚度[23]。因控制系统的快速性（微分环节）太强，积分环节略弱致使系统发生超调震荡引起车辆“点头”有必要依靠 ABS 系统保证较好的舒适性和制动稳定性。

ABS 系统控制与系统转向系统(4WS)的结合既有助于提高车辆的行驶稳定性;也有利于缩短制动距离。众所周知，在车辆紧急制动时 ABS 系统能防止车轮拖地，提高方向可控性和缩短制动距离。然而在有些情况下，方向稳定性和短的制动距离之间存在矛盾，对于不同的摩擦系数路面，在 ABS 设计者中普遍存在一种观点:即认为在这种情况下应该优先考虑方向稳定性；而不是最短距离，这样就会导致制动距离的延长。为了克服以上矛盾，需要另外增加 A-4WS 系统来保证方向稳定性，而 ABS 系统只解决制动距离的问题。A-4WS 出现时的车辆横摆角速度，并与控制器内算出的理想车辆横摆角速度相比较，

然后令后车轮产生一转向角，以消除实际与理想横摆角速度之间的偏差。即使在低召路面上制动，也可以获得良好的稳定性能。

电子稳定系统 ESP (Electronic Stability Program)是对人们熟知的 EDS(电子差速锁),TCS(牵引力控制)系统的进一步扩展，具有很高的集成度，从而降低了轿车的重量和价格。标准化的接口使用户可以根据自己的需要选装相应功能，也使之适用于多种排量的发动机和多种驱动形式。它不仅整合了 ABS 和 ASR(加速防滑系统)的所有功能，而且还能在车轮自由滑转以及极限操纵下保持车辆的稳定性；它可以比 ABS 和 ASR 更好地利用轮胎和路面间的附着潜能，在改善车辆转向能力和稳定性的同时，进一步改善驱动能力和缩短制动距离。

现代制动防抱死装置多是电子计算机控制，这也反映了现代汽车系向电子化方向发展。ABS 已成为当今世界上公认的提高汽车安全性必不可少的系统。目前在我国生产的部分车型中已有安装 ABS 系统的，集成了先进的嵌入式技术，同时借助于电子控制技术制动防抱死系统反应更灵敏、成本更低、安装更方便[24-25]。今后的 ABS，一是技术性能提高，使其更加完善；二是进一步简化系统，使之小型轻量化。这两项将是汽车 ABS 今后的发展方向。可以预计 21 世纪汽车的发展将是电子控制的时代，汽车在电子系统控制下将变得更加清洁、安全与舒适。

2 电动汽车 ABS 控制算法的研究

汽车ABS实质上是一种制动力的自动调节装置。通过调节制动力,将汽车轮胎在制动过程中的滑移率控制在预定范围内,从而提高汽车的制动效能及汽车在制动过程中的横向稳定性和方向可操纵性等。所以ABS是一个典型的控制系统。

2.1 概述

汽车制动过程具有明显的非线性、时变性和不确定性等特点,防抱制动系统(ABS)的控制成为突出的难点。目前，在实际的ABS系统中普遍采用基于经验的逻辑门限控制方法，由于缺乏足够的理论指导，逻辑门限的方法在选择门限值、车速测量及路面识别上存在困难，基于滑移率的鲁棒控制（健壮性控制）系统,在系统稳定性和抗干扰能力上有所提高,但鲁棒控制需要知道模型传递函数误差的上限,选择加权函数具有一定的难度[26-28]。还有基于滑移模式对ABS 进行控制的方法,具有这种滑移模式控制律在车轮防抱系统中可获得较高的制动效率，但如何选定参数以及消除相轨迹在沿曲线滑移过程中存在的抖动现象是一个有待进一步研究的课题。论文主要是对传统逻辑门限控制、PID控制及模糊控制进行了模拟研究。

2.2 汽车 ABS 控制的原理

在驾驶员、汽车和环境三者所组成的闭环系统中，汽车与环境之间的最基本联系是轮胎与路面之间的作用力。由于汽车行驶状态主要是由轮胎与路面之间的纵向作用力与横向作用力决定的，因此车轮与路面之间的作用力必然要受到轮胎与路面之间附着力的限制。由经验可知，如果驾驶员断续地踩制动踏板，就可以防止车轮抱死。但一般驾驶员要掌握这种制动技巧是非常困难的,惟一能够自动控制制动过程的就是ABS系统。ABS系统就是最大限度地利用轮胎与路面的纵向和横向附着系数，从而在制动过程中增强汽车的稳定性，防止侧滑和摆尾，同时在紧急制动过程中保持转向操纵能力，有效利用纵向附着力可以缩短汽车制动距离，同时也使轮胎的磨损大为减轻。

2.3 ABS 的控制方法

汽车ABS的控制方法主要有逻辑门限值控制方法、滑动模态变结构控制方法、PID 控制方法和鲁棒控制方法等。其中逻辑门限值控制方法使用历史最长,但是它的控制逻辑

复杂,不同路况下各种门限值及保压时间一般是经过反复实验得出的经验数值,没有十分明确的理论依据,对系统的稳定性等品质无法评价,而且由于控制过程中逻辑门限总是处于波动状态,因此控制效果不太好,制动距离也稍长;滑动模态变结构控制可获得较高的制动效率,但是在换节线附近切换时,由于系统的惯性,在滑动运动中叠加了一个抖动,如何选定参数及消除相轨迹在沿曲线滑移过程中存在的抖动现象还有待进一步研究;对于PID 控制,只要现场整定的PID参数合适,就会得到较好的控制效果,但其性能效果仍有待改进和提高;鲁棒控制在系统稳定性和抗干扰能力上有所提高,但鲁棒控制需要知道模型传递误差的上限,选择加强函数具有一定的难度。因此探索一种有效的控制方法一直是ABS发展的关键。

模糊控制采用类似于人脑的模糊推理方法,遵循一定的控制规则,结合实际经验,对系统进行动态调控,具有不依赖对象的数学模型、便于利用人的经验知识、鲁棒性好、简单实用等优点,适应于ABS这种变工况非线性系统。本文对模糊控制方法用于ABS控制进行模拟研究。

2.3.1 逻辑门限控制

这种控制方法是基于车轮角速度变化与制动力矩、附着系数和滑移率的变化有强烈的敏感性,在制动过程中, 车轮抱死总是出现在dw/dt相当大的时刻。因此可以预选一个角速度门限值,当实际的角速度超过此门限值时, 控制器发出指令,开始释放制动力矩使车轮得以加速旋转;再新选一个角加速度门限值,当车轮的角速度达到此门限值时,控制器又发出指令, 使制动力矩开始增大,车轮作减速运动。所以可以采用一个车轮角速度传感器作为单轮信号,同时在电子控制器中设置合理的加、减速度门限值, 就可以实现防抱制动的循环。

防抱门限大多选择加、减速度作为主要门限,以滑移率(由滑移率来了解汽车车轮是否已抱死)作为辅助门限。因为单独的加减速门限有很大的局限性, 在初始和高速紧急制动情况下,有可能使防抱制动逻辑在后续的控制中失效, 因此需要将角速度和滑移率这2个门限结合起来,以识别不同路况。逻辑门限控制方法的缺点在于控制系统中的各种门限及保压时间(holdup time)都是从反复试验中得出的经验数据, 而无充分的理论依据, 对系统的稳定性等品质无法评价。

2.3.2 PID控制

经典的PID控制的设计建立在试凑的基础上，只要现场整定PID参数合适，就会得到较好的控制效果，定义期望的滑移率ST与实际滑移率之差为PID的输入，由PID控制算法算出控制气压值反馈给制动系统，构成典型的反馈控制，系统结构如图2.1所示。

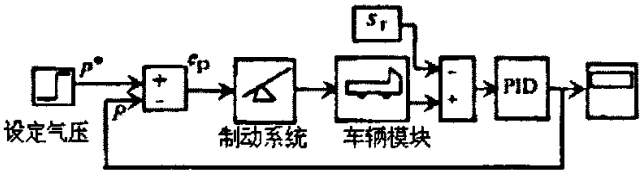

图 2.1 PID控制的防抱死制动系统

Fig. 2.1 The ABS system controlled by PID

2.3.3 模糊控制

模糊控制通过模糊逻辑和近似推理方法，把人的经验形式化、模糊化，使之成为计算机可以接受的控制模型，让计算机代替人来进行有效的实时控制。模糊控制是基于经验规则的控制，与系统的模型无关，有很好的鲁棒性和控制规则灵活性，控制规则符合人的思维规律[29]。

模糊控制防抱死系统如图2.2所示，其控制过程是，制动时由制动踏板输出一个阶跃输入，而输入给制动系统一个设定值制动气压，进入求和模块，另一个进入求和模块的是来自控制器的输出变量，求和后输出给伺服系统的是实际给定系统的目标压力值，通过限值模块，输入给气压系统及气动伺服阀系统，求得系统的动态制动力矩，它输入给车辆模块使车辆进行制动，车辆模块输出为车辆的实际滑移率，它输入给模糊控制器模块，实际滑移率与设定期望滑移率构成误差，由模糊控制算法算出控制气压值反馈给制动系统。

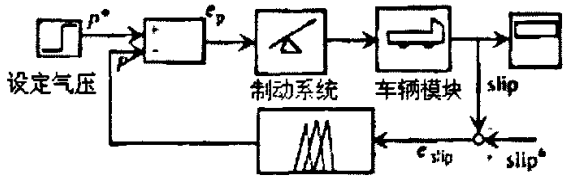

图2.2 模糊控制防抱死系统

Fig. 2.2 The fuzzy control system of ABS

2.4 模糊控制 ABS 的算法设计

车轮滑移率定义为：

$$S=1-\omega R/v \tag{2.1}$$

式中：S为车轮滑移率； ω为车轮角速度； R为车轮滚动半径；v为车辆速度。

由上式可知，滑移率由车辆速度和车轮角速度确定。因此，模糊控制ABS以车轮角速度和车体纵向加速度为输入信号。其中车轮角速度可由电磁式轮速传感器直接测得；车辆速度的获取则采用间接的方法，由车轮的角速度和加速度构造车辆的参考速度。

图2.3为运用模糊逻辑方法对车辆参考速度进行估算的结构框图。输入为4个由电磁式轮速传感器测得的角速度信号和1个由电容式加速度传感器测得的车辆加速度信号，输出为车速值。

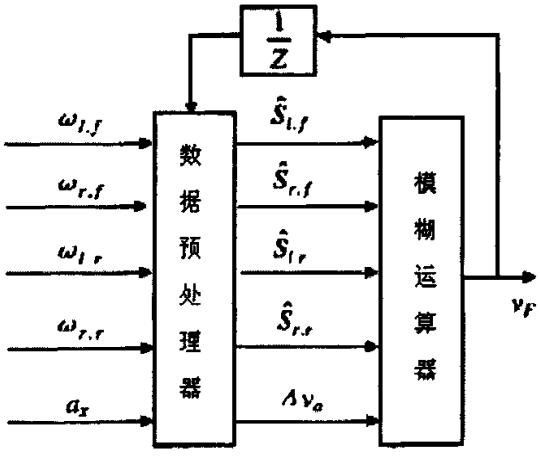

图 2.3 车速估算框图

Fig. 2.3 The structure figured out of vehicle speed

输入信号经过数据预处理模块中的低通滤波器后,由下列公式分别计算出4 个车轮的参考滑移率和整车的速度变化率:

$$S(k) = \frac{v_F(k-1) - \omega(k-1) \times R}{v_F(k-1)} \times 100\% \tag{2.2}$$

$$\Delta v_a(k) = \frac{[a_x(k) - a_e(k)] \times T}{v_F(k-1)} \times 100\% \tag{2.3}$$

式中: S(k)为车轮的参考滑移率; Δva(k)为整车的速度变化率; ae(k)为加速度修正值; T为延迟时间。模糊逻辑运算器的输入变量为4个车轮的参考滑移率和整车的速度变化率,将输入变量进行模糊化,用语言变量来表示,可分为4级:负小(NS); 零(ZO); 正小(PS) ; 正大(PL)。其隶属函数形状如图2.4所示。模糊运算器的输出变量为各个输入变量的权值ki 和ka 。将输出变量用语言变量来表示,可分为3 级:小(S); 中(M); 大(L)。其隶属函数形状如图2.5所示。

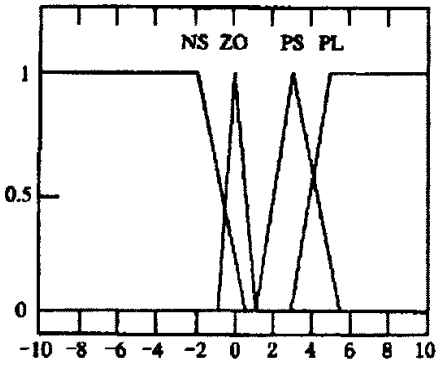

图 2.4 输入变量隶属函数

Fig. 2.4 The subjection function of input value

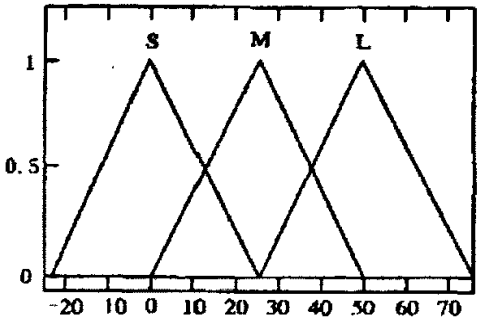

图 2.5 输出变量隶属函数

Fig. 2.5 The subjection function of output value

由下式可求出车辆的参考速度:

$$v_F(k) = \{ \sum_{i=1}^{4} k_i \omega_i(k) R + k_a [v_F(k-1) - Ta_{x,corrected}(k)] \} / (\sum_{i=1}^{4} k_i + k_a) \tag{2.4}$$

ABS 模糊控制系统如图2.6所示。其控制过程:制动时由制动踏板输出一个阶跃输入,而输入给制动系统一个设定值制动气压,进入求和模块,另一个进入求和模块的是来自控制器的输出变量,求和后输出给伺服系统的是实际给定系统的目标压力值; 通过限值

模块，输入给气压系统及气动伺服阀系统，求得系统的动态制动力矩，它输入给车辆模块，使车辆进行制动；车辆模块输出为车辆的实际滑移率，它输入给模糊控制器模块，实际滑移率与设定期望滑移率构成误差，由模糊控制算法算出控制气压值反馈给制动系统，使汽车达到良好的制动效果和操纵稳定性。

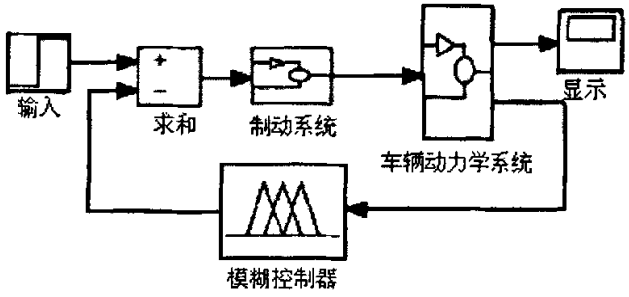

图 2.6 ABS 控制系统的仿真模型

Fig. 2.6 The simulation model of ABS control system

2.5 算法的仿真与分析

为了对上述三种控制方法进行模拟，采用MATLAB/SIMULINK国际控制界流行的软件，不仅可用于车辆动力学的模拟，而且可进行系统验证，该软件特别适合复杂系统的研究，它提供的各种控制工具箱可以方便地实现上述控制思想的仿真[30]。采用MATLAB中的SIMULINK软件分别建立车辆、轮胎、制动器及控制系统的模块图，构成如图2.6所示的仿真模型。对系统进行模拟，用ode45高阶法解微分方程，采用变步长算法，初始步长为0.2，最大步长为0.5，绝对误差为1×10^{-6}，相对误差为1×10^{-3}，制动时车轮初速度为80km/h。模糊控制采用MATLAB 中的FUZZY专用工具箱进行模糊控制研究，控制轮速较平稳，只有很微小的波动，趋于理想的控制状态。

基于上述三种控制方法进行了仿真计算，为突出控制规律的研究，采用单轮车辆模型。现以电动公交车的单轮系统为模型，计算参数如下：

$J = 20kg\cdot m^2$, $g = 9.8m/s^2$, $N = 18\ 125N$, $C^* = 21\times10^{-3}Nm/Pa$, $V_0 = 22m/s$,

$m = 1850\ kg$, $L_h = 0.8$(取干路面), $L_g = 0.7$。

图2.7和图2.8为逻辑门限控制的车速、轮速模拟和滑移率的模拟，加减速度采用门限值控制，轮速必然有一定的波动。单独的加减速度门限有很大的局限性，在初始高速紧急刹车，有可能使防抱控制逻辑在后继的控制中失败，如在高L路面出现过分减压而在低L路面出现抱死等情况。由于以加减速度为主要控制对象，从模拟结果可以看出滑移率有较大的波动。在制动的前段由于考虑到要让车轮充分制动，所以滑移率偏高呈现抱死趋向，后段就较平稳。

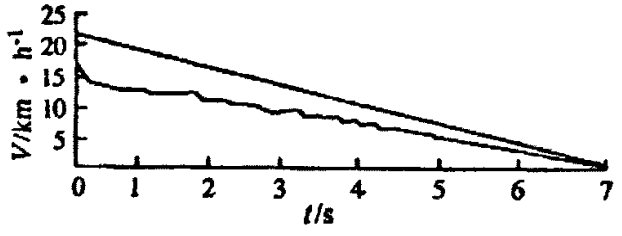

图2.7 逻辑门限控制的车速轮速模拟

Fig. 2.7 The simulation of vehicle wheel speed controlled by logic gate limit

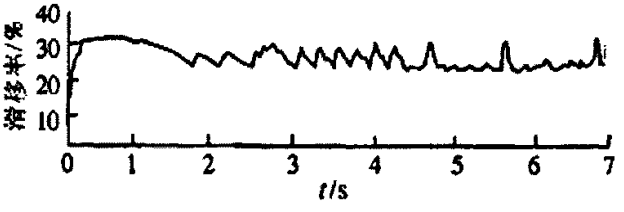

图2.8 逻辑门限控制的滑移率模拟

Fig. 2.8 The simulation of slip ratio controlled by logic gate limit

PID的参数可设计为$P = K(bS_t - S_n)$，其中S_t为期望滑移率，在系统中$S_t = 0.25$，S_n为实际滑移率，b为适应不同路况的加权系数；取为1，K为比例系数，设为2；系统模拟结果如图2.9和图2.10所示。

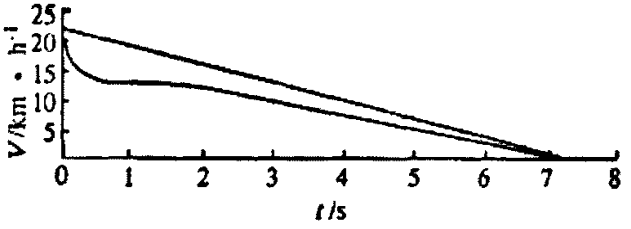

图2.9 PID控制的车速轮速模拟

Fig. 2.9 The simulation of vehicle wheel speed controlled by PID

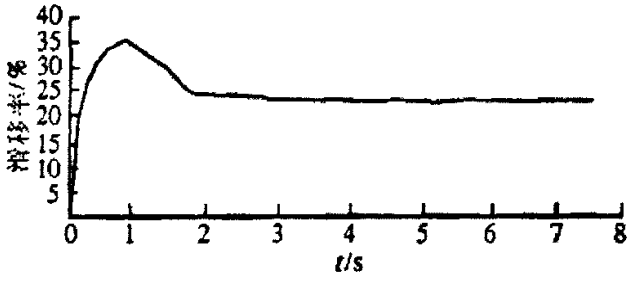

图2.10 PID控制的滑移率模拟

Fig. 2.10 The simulation of slip ratio controlled by PID

PID控制防抱系统模拟结果表明，在制动初期滑移率有较大超调，而整个过程比较平稳。在模拟过程中发现，如果改变系统的结构参数，如将气动回路传递函数增益扩大1倍，则对系统输出结果有较大影响，甚至影响整个系统的稳定，在调整过程中还发现，适应低L值的PID参数对高L值路面情况的调节控制效果不佳，反之亦然[31-34]。

模糊控制采用基于滑移率的模糊控制，对系统进行模拟用ode45高阶法解微分方程，采用变步长算法，初始步长为0.2，最大步长为0.5，绝对误差1×10^{-6}，相对误差1×10^{-3}，制动时车轮初速度为80km/h。

基于滑移率的模糊控制轮速比较平稳，只有很微小的波动趋于理想的控制状态，图2.11是车速轮速模拟结果。

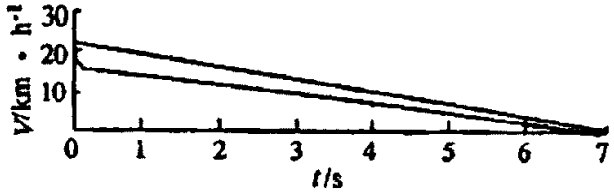

图2.11 滑移率控制的车速轮速模拟

Fig. 2.11 The simulation of vehicle wheel speed controlled by slip ratio

滑移率表现很平稳,整个制动过程,经过最初的增压制动后基本上都维持在25%期望值处,上下波动很微小,制动过程非常平稳,如图2.12所示。

图2.12 滑移率控制的滑移率模拟

Fig. 2.12 The simulation of slip ratio controlled by slip ratio

模糊控制器制动压力输出比较灵敏,制动初期由于制动力的迅速增大,轮速急速降低而车速仍维持在较高水平。如果不迅速调整将很容易形成抱死。由模拟的结果可以看出,基于滑移率的模糊控制有较理想的制动防抱效果。它最主要的优点除了系统本身有较强的鲁棒性外,在所有路面上都能确保车辆旋转恢复到稳定区域。

通过对三种控制方法的研究和仿真设计,可以看出汽车ABS防抱制动系统都能将滑移率控制在理想值附近,提高了汽车的制动性能。相比而言,模糊控制方法的原理简单,容易实现,只要赋予控制器足够的控制能力,就能很好地适应路况及车型结构参数的变化,虽然模糊控制现在还处于理论研究阶段,并没有转化为实际的产品,但是,模糊控制的防抱制动系统必定是未来的发展方向。因此,从研究的前瞻性上考虑,本文在后面的系统实际中设计中,将会运用这种方法作为控制器里的控制策略。

3 电动汽车ABS嵌入式系统的总体设计

3.1 ABS嵌入式系统设计需求分析

目前嵌入式系统技术正在积极建立一个更具联通性的用户基础，无论是网络，还是无线技术，这基础需要那些更具兼容性的互联式设备。此外，应用设备硬件已变得更为廉价，同时也具有更强的通用性，这就使得开发商通过更丰富的功能区分其产品。

倘若我们换一种角度来看，当许多信息设备跟汽车产业融合的过程中，未来会有更多智能应用设备的市场需求出现在车上，而汽车制造商不得不随着市场变化而开始缩短开发周期，重视服务特性的多样化；最后，所得到结果就是，汽车制造商开始倾向使用商业化嵌入式设备，这些产品中内置的组件，可以快速地组装在车内各个智能系统设备中。随着汽车行驶速度的提高，道路行车密度的增大，对于汽车行驶安全性能的要求也越来越高。汽车的防抱死制动系统(ABS)应运而生，它是以传统制动系统为基础，采用电子控制技术，在制动时防止车轮抱死的一种机电一体化系统。

ABS系统设计中主要考虑以下几个问题：首先，由于ABS系统直接关系车辆的安全性能，因而它的故障问题显得极为重要，系统必须保证能及时检测故障并准确判断故障点；其次，ABS系统通常包含电磁阀等感性负载，驱动电流很大，需要适当的驱动电路；此外，为了便于ABS系统与车辆上其他系统进行通信，系统需要预留通信接口。

汽车防抱死制动系统(ABS)发源于20世纪60年代，普及推广运用于80年代，直到90年代末期才在国内合资生产的中高档轿车上安装使用。而在2001年第3届北京国际车用设备展览会上，ABS装置在国内主要电动公交车厂家展出的众多豪华客车上初露面容。

电动公交车由于乘员多，大中型电动公交车可载30~50,车质量大，可达50t以上，高速公路上行驶车速多在100km/h以上，应该说，电动公交车的安全可靠性相对其它车种要求更严格。ABS可以提高电动公交车制动性能，有效防止电动公交车制动侧滑、跑偏，保证制动时操纵稳定性，因此随着我国电动公交车装备技术的提高，ABS在电动公交车上推广运用是指日可待的。欧共体和日本早在1991年就在电动公交车安全法规中规定，行驶在高速公路上的大型电动公交车必须安装ABS。我国交通部在有关电动公交车

定级标准中也相应规定：定为高级的电动公交车也必须装备ABS。2003年我国各厂家推出的中高档电动公交车上均实际装备了ABS，而不仅仅作为炫耀的商业技术卖点了。

我国现行电动公交车上装用的ABS具有以下几个特点：

（1）技术起点高。早期价廉的机械式ABS被摒弃，现均为电控式ABS并配装了缓速器。高档电动公交车多采用了ABS/ASR系统，ASR是ABS性能的延伸和扩充，可防止驱动轮在湿滑路面上滑转，这表明厂家有意识采用最新的ABS。

（2）型号繁多。ABS有引进各国电动公交车底盘直接配装的；有国内合资厂定点生产的。国内合资生产的ABS由于价廉质优渐占主流。

（3）性能可靠。我国大中型电动公交车基本实现了柴油化，且制动系统均为双管路，前盘后鼓式渐趋上风，所以现有电动公交车上采用的ABS为4轮各装一个传感器气源式四控制回路，其可靠性好。客运单位实际使用ABS后，反映良好，ABS故障率低，少数故障多是因维修和维护原因造成的。这表明电动公交车上的ABS获得了客运单位和驾驶员的欢迎和认同。

3.2 嵌入式系统的分析与架构

基本上，嵌入式系统是指在系统运算过程中重要的基础架构，也是嵌入在对象主要架构体系中，能够完成指定功能的专用计算机系统。嵌入式系统有体积小、低功耗、集成度高、子系统间能通信融合等优点，正当汽车技术发展及微处理器技术的持续进步，嵌入式系统设备也在车用电子技术中开始获得广泛应用。

一般来说，从车身控制、底盘控制、发动机管理、主被动安全系统到车载用娱乐、信息系统都能够见到嵌入式技术的身影。

那么，嵌入式设备为何越来越受到车用电子厂商的重视？汽车嵌入式系统是嵌入式系统向实时多任务管理、网络耦合与通信的高端应用过渡的产物，大大提高了汽车电子系统的实时性、可靠性和智能化程度。除了具备普通嵌入式系统的共有特性外，它还具有以下几项特点：

（1）提供智能汽车未来强大的网络通信功能。

（2）实时多任务处理的支持能力。

（3）强大储存区的保护功能。

（4）实时操作系统支持。

（5）提高系统集成度、降低了成本。

（6）超低功耗。

（7）适应高温、潮湿、振动和电磁辐射。

（8）支持软件多线程结构、增强软件抗干扰性。

（9）强大储存区的保护功能。

基于创造新型电动汽车为主要出发点，要如何将车上嵌入式系统技术能够融合概念、芯片、系统与软硬件技术，才能设计出安全、高效及智能化的汽车电子系统。一段时间的发展后，车用嵌入式系统在低阶发展到高阶的过程，每一个发展阶段都有其主要特点。这可从微处理器越来越强大的处理功能，开发出新理论、新方法和新的关键技术等趋势窥出所以然。

原先，从 4 位和低档 8 位微处理器为核心，将 CPU 和外围电路集成到一个芯片上，配置了外部并行总线、串行通讯接口、SFR 模块系统。硬件结构和功能相对单一、处理效率低、存储容量小、软件结构也比较简单，不需要嵌入操作系统，由于该系统主要用在特性简单、数据处理量小和实时性要求不高的控制场合，如：雨刷、车灯系统、仪表盘及电动门窗等。

到了第二阶段，便以高档 8 位和 16 位处理器为主要核心，可集成较多外部接口功能单元，如：A/D 转换、PWM、PCA、Watchdog、高速 I/O 口等，配置了芯片间的串行总线；软件结构比较复杂，程序数据量有明显增加。能够完成简单的实时任务，目前在智能型车辆的电控系统中，也获得了广泛应用，如：ABS 系统、智能安全气囊、主动悬架及发动机管理系统等。

现阶段来说，以性能极高的 32 位甚至 64 位嵌入式处理器为核心，在对容量大但离散时间信号要求快速处理的场合，必须使用 DSP 作为协处理器，并满足汽车系统不断扩展的嵌入式应用需求外，还必须提高处理速度，增加存储容量与集成度。

在嵌入式操作系统的支持下，具有实时多任务处理能力，同时与网络的耦合更为紧密。车用嵌入式技术在汽车电子上的高端应用，满足了现代汽车电控系统功能不断扩展、逻辑渐趋复杂、子系统间通信频率不断提高的要求，代表着汽车电子技术的发展趋势。

其主要应用，如：混合动力总成、底盘综合控制、汽车定位导航、车辆状态记录与监控等领域。

电动汽车持续燃烧，这使得车内的系统设备，必须具备更强大软、硬件结构。硬件结构包括：嵌入式处理器和外围设备，软件包括应用软件和操作系统。软件构造则是负责：数据结构、算法和通讯协议实现汽车电子控制策略，硬件则为软件提供了运行平台，执行具体控制。

汽车嵌入式硬件系统集成度越来越高，一般为模块化结构。在高性能 CPU 核心外通过 CAN 总线扩展实时时钟模块、SRAM（静态随机内存）及大容量 FLASH，配置 CAN 总线与 USB 通信模块，无缝集成 PWM 输出、多通道串口、A/D 转换接口与统一的高速缓冲存储器，支持 RISC 技术、多级流水线技术与在片调试技术。

系统的实时处理能力、可靠性和网络通信能力大大增强。另外，在通讯接口方面，支持相应的通信组网协议软件和物理层驱动软件，提供容错数据传输能力和更大通讯频宽。

现代汽车电子系统从单一控制逐渐发展到多变量多任务协调控制，软件越来越庞大，越来越复杂，使得汽车嵌入式系统需要寻找新的软件解决方案。汽车嵌入式系统软件的典型结构采用基于标准化接口和通讯协议的模块化软件设计，系统内部通讯由交互层直接完成，保障应用程序间的信息传送。

网络层拥有数据流处理能力，是不同系统层面间信息交换的中间接口，能最大程度地整合系统资源。嵌入式实时操作系统摒弃了传统操作系统的前后台模式，使用总线驱动层和硬件抽象层管理 I/O 端口，合理分配 CPU 资源，采用基于优先级的事件管理策略，通过应用程序接口（API）调用应用程序，根据邮箱、消息队列和信号量机制综合管理中断、系统行为和任务分配等。

面对到汽车的应用环境多变，硬件稳定性、可靠性要求很高，还必须要考虑设备的延展性。因此，要如何选择一项具有优势的嵌入式设备控制器，就显得相当重要，使控制器能够长时间持续在恶劣环境下，能够因应高振动、高冲击、低温、高温、温度变化剧烈和高湿度等环境问题。

另外，在内部运算方面，也要能具有可编程逻辑控制器、比例放大器、模拟量输入A/D模块、继电器输出功能于一身的高性能机械专用控制器。

（1）取代传统控制的比例放大器/电路，可同时驱动12片电液比例阀，在液压系统的设计成本上有绝对的优势。

（2）必须具有较大电流输出端口，才能取代传统电路中的继电器功能。

（3）控制器的模拟量输入端口要具备不同信号的处理功能，不论是电阻信号、电流信号还是电压信号都有能力进行侦测，并利用软件编程进行设定。

（4）在逻辑控制功能方面，要能解决过去使用各种类的继电器和模块很难解决的逻辑运算及数学运算。

（5）控制器内部采用数个16位高性能微处理器及大容量的内存空间使其具有极强的数字处理能力，可完成很多以前必须由计算机高级语言来编程的复杂算法，如直行纠偏多PID控制、发动机油泵复合控制、姿态模糊控制、自动驾驶等等。

（6）控制器是基于CANBUS总线开发的产品，当系统节点较多或信号传输距离较远时可以使用不止一个控制器组网，节省接线、提高可靠性、并且灵活性和可扩展性也需要增强。

车用嵌入式系统具有卓越的性能，其优越性逐渐被汽车界所认可。不过，未来车用嵌入式系统，将以通讯接口、缩短时程、统一标准、应用扩张等趋势来发展。

（1）通讯接口：随着汽车局域网技术和智能交通技术的发展，车用嵌入式系统平台将会形成以C级或D级网络为基础的整车分布式控制系统和以无线通信为基础的远程高频网络通信系统。

（2）缩短时程：汽车嵌入式系统将会应用FPGA/CPLD（在线可编程门阵列）等技术，发展高弹性、低成本、且确实可行的技术解决方案，相较于目前使用的传统硬件解决方案，这些可编程逻辑组件可改善产品上市时程的问题。而系统由分布式可编程互连逻辑单元构成，单元之间可以交换信息，大量运算由硬件直接完成，整个架构能够更灵活应用在车上，设备的集成度也会高于目前的状况。

（3）统一标准：在系统开发上，遵循市场通用汽车电子系统开放平台和统一的标准。为了提高软硬件通用性，加快开发速度，降低成本，嵌入式系统必须要在最快的时

间内，建立统一标准的开发平台；以目前的趋势来看，欧洲所颁布的 OSEK / VDX 标准中 MODISTARC 规范，将是汽车嵌入式系统开发平台的发展趋势。

（4）应用扩张：车用嵌入式系统的应用范围，在短期之内应该会从高级进口车款扩展至较低阶车款及国产车上。

3.3 ABS 控制系统的组成设计

ABS的电路由电子控制器(ECU) 、车速传感器、液压调节器、ABS继电器、故障指示灯、诊断连接器、开关信号与电源等电路组成。制动时，踩下制动踏板，制动开关闭合，ECU实时采集车速转感器送来的车轮转速信号，同时对这些信号进行分析判断，若发现某车轮有抱死趋势时，就给ABS继电器和液压调节器发出控制信号，以调节制动管路压力，防止车轮抱死。ECU还具有自诊断功能，当检测出有故障发生时，就点亮ABS故障指示灯或制动故障警告灯，并存储故障代码，当用诊断连接器检查故障时，可输出故障代码。

3.4 ABS 控制系统的结构详细设计

3.4.1 电子控制器(ECU)

电路ECU是ABS的核心，整个电路封装在一个金属盒内，它负责各种信号及数据的采集、处理、分析、判断以及输出控制信号，记忆故障代码，进行自诊断等功能。

3.3.2 车速传感器电路

该系统采用4个磁脉冲式车速传感器来检测4个车轮的转速，ECU通过计算单位时间内传感器送来的脉冲数就能计算出车轮的转速，对转速微分就可获得车轮角加速度。每个传感器有2根线与ECU相连(见图3.1)。

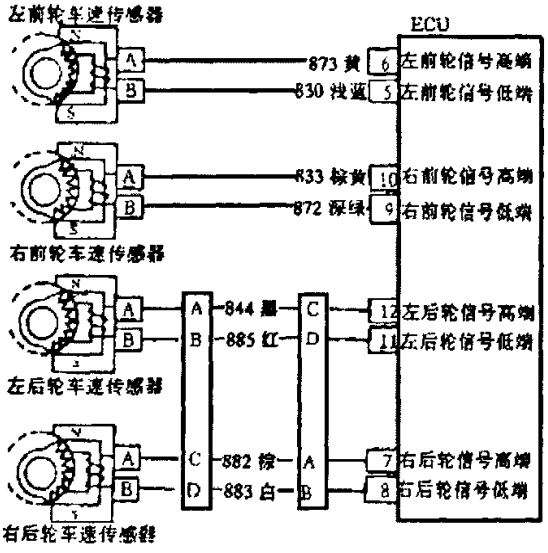

图3.1 车速传感器电路

Fig. 3.1 The circuit of vehicle speed sensor

3.4.3 ABS继电器及液压调节器电路

ABS的继电器与液压调节器分为独立的两体，分别装在发动机舱内的不同位置。当点火开关闭合时(见图3.2)，由点火开关送来的电源电压经25A熔断丝接到继电器的5脚，再通过继电器线圈接到ABSECU 的22脚上。当22脚上输出高电平时，继电器电磁线圈两端无电压，1脚与与4脚间的触点断开，继电器4脚上无电压输出。当22脚输出低电平时，继电器电磁线圈得电，继电器1脚与4脚间的触点接通，蓄电池电源经继电器1脚，触点，4脚，ECU的A脚，给ECU送去继电器的闭合信号，同时蓄电池电源经继电器4脚到左、右前轮电磁线圈的B端，此时如果ECU的19脚和20脚均输出低电平，则左、右前轮电磁线圈均得电，如果这时左、右前轮的电磁阀也得电，则电磁阀就会在电磁线圈的电磁力作用下，由原来的常开状态转为关闭状态，从而关闭前轮缸到主缸间的主油路。由图3.2可见，左、右前轮的电磁阀的B端均搭铁，其A端分别接ECU的24脚和4脚，如果此时ECU的24脚、4脚均输出高电平，则左、右前轮电磁阀就通电，电磁阀关闭。

ABS液压调节器内的3个液压调节电动机分别有2根线与ECU相连，其中一根为高电压端，另一根为低电压端，当低电压端输出低电平且高电压端输出高电平时，电动机开始正

向旋转，反之，电动机反向旋转。ECU还可调节2根线间的电压差，从而调节了电动机的电流，也就调节了电动机的转速。

3.4.4 点火开关与故障指示灯电路

由图3.3可见，该部分电路主要由点火开关、制动指示灯、ABS灯、指示灯驱动电子组件、熔断丝组成。点火开关信号通过25A熔断丝接到ECU的14脚上，同时接到指示灯电路的熔断丝上和ABS继电器的5脚上。制动指示灯右侧接ECU的21脚和制动液面传感器，当制动液面过低时，该传感器对搭铁短路，从而将制动指示灯点亮，并将制动液面低信号送给ECU的21脚，表示制动系统出现严重故障，应停车修理。ABS灯亮表示防抱死功能出现故障，而常规制动性能正常，汽车可继续行驶，当ECU检测到影响防抱死功能有故障时，就从23脚上给指示灯驱动电子组件的D端发出低电平控制信号，该信号经驱动电子组件的电流放大作用后，通过驱动电子组件的E端将ABS灯点亮。

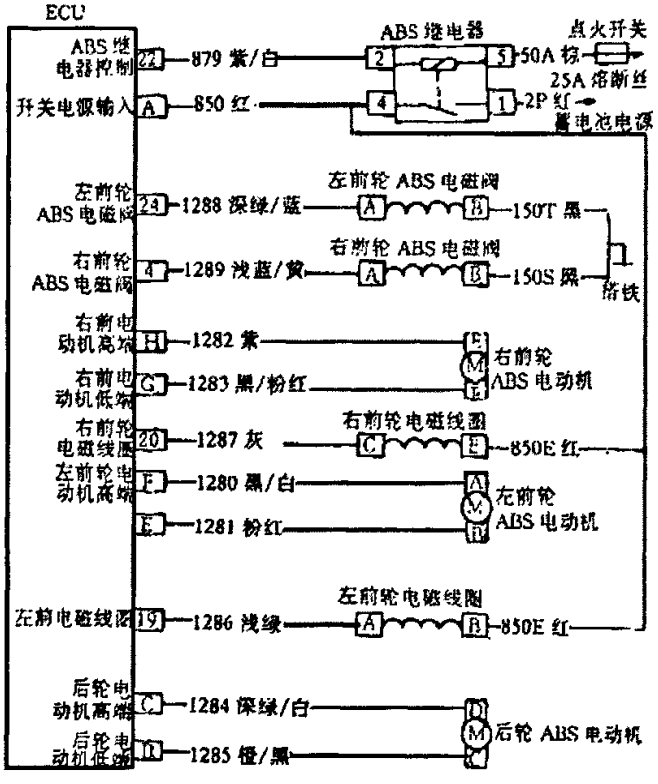

图3.2 ABS继电器及液压调节器电路

Fig. 3.2 ABS relay and the circuit of hydraulic pressure adjuster

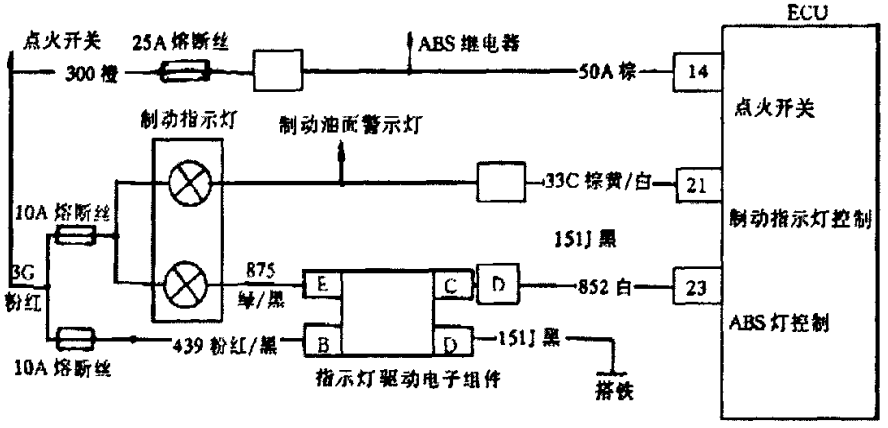

图3.3 点火开关与故障指示灯电路

Fig. 3.3 Ignition switch and the circuit of hitch indicator

3.4.5 制动开关与诊断信号输出电路

由图3.4可见，制动开关的B端通过20A的熔断丝接蓄电池正极，A和C端接ECU的13脚，当制动开关被踏下时，ECU的13脚就会收到一个制动开始信号(高电平)。ECU的2脚是诊断信号输出脚，ECU内存储的故障代码通过2脚送到诊断连接器和旅行电脑或发动机控制电脑。

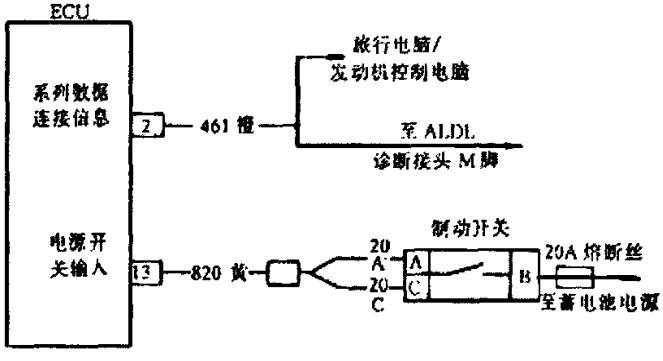

图3.4 制动开关与诊断信号输出电路

Fig. 3.4 Trig switch and the output circuit of diagnosis signal

3.4.6 ECU的电源电路

图3.5可见,ECU的电源是从15脚通过熔断丝接到起动机的电源上,并通过15脚给ABS继电器的1脚(供电端)供电。ECU的B端为搭铁端。

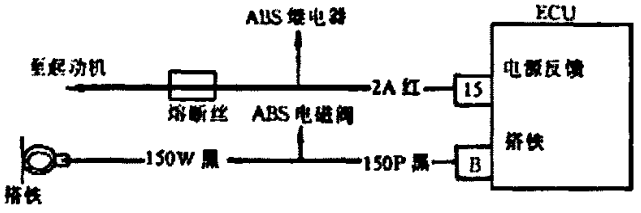

图3.5 ECU的电源电路

Fig. 3.5 The power circuit of ECU

4 嵌入式 Linux 系统在 ARM 平台的移植

嵌入式 Linux 是按照嵌入式操作系统的要求而设计的一种小型操作系统，它由一个 Kernel(内核)及一些根据需要进行定制的系统模块组成。Kernel 一般只有几百 kB 左右，即使加上其它必须的模块和应用程序，所需的存储空间也很小。它具有多任务、多进程的系统特征，有些还具有实时性。一个小型的嵌入式 Linux 系统只需要引导程序、Linux 微内核、初始化进程 3 个基本元素。运行嵌入式 Linux 的 CPU 可以是 x86、Alpha、Sparc、MIPS、PPC 等。与这些芯片搭配的主板都很小，通常只有一张 PCI 卡大小，有的甚至更小。嵌入式 Linux 所需的存储器不是软磁盘、硬盘、Zip 盘、CD-ROM、DVD 这些众所周知的常规存储器，它主要使用 Rom、CompactFlash、M-Systems 的 DiskOnChip、Sony 的 MemoryStick、IBM 的 MicroDrive 等体积极小(与主板上的 BIOS 大小相近)，且存储容量不太大的存储器。它的内存可以使用普通的内存，也可以使用专用的 RAM。 与其它嵌入式操作系统相比，Linux 的源代码是开放的，不存在黑箱技术。Linux 作为一种可裁剪的软件平台系统，很可能发展成为未来嵌入式设备产品的绝佳资源。Linux 与生俱来的优秀网络血统更为今后的发展铺平了一条宽广平坦的大路。因此，在保持 Linux 内核系统更小、更稳定、更具价格竞争力等优势的同时，对系统内核进行实时性优化，更加使之能够适应对工业控制领域高实时性的要求。这也正是嵌入式 linux 操作系统在嵌入式工控系统中的发展所在。同时也使 Linux 成为嵌入式操作系统中的新贵。标准的 Linux 内核通常驻留在内存中，每一个应用程序都是从磁盘装载内存上执行。当程序结束后，它所占用的内存就被释放，程序就被卸载了。而在一个嵌入式系统里，可能没有磁盘。有两种途径可以消除对磁盘的依赖，一是在一个简单的系统里，当系统启动后，内核和所有的应用程序都存在内存里。这是大多数传统的嵌入式系统的工作模式，同样 Linux。第二种就是 linux 所特有的功能，因为 Linux 已经有能力“加载”和“卸载”程序，因此，一个嵌入式系统就可以利用它来节省内存。一个比较典型的系统有大约 8MB 到 16MB 的闪存和 8MBRAM，而闪存可以被用作文件系统。用闪存驱动程序作为从闪存到文件系统的界面就是一种选择。当然，也可以用一个闪存磁盘。用闪存来摆脱系统对一个磁盘的需求(依赖)具有 DiskOnChip 技术以及 CmopactFlash 卡等方式。

用来连接 FlashMemory 和文件系统的程序都以文件形式存储在 Flash 文件中，需要时可以装入内存，这种动态的、根据需要加载的能力是支持其它一系列功能的重要特征。它能使初始化代码在系统引导后被释放。实际上，Linux 同样还有很多内核外运行的公用程序，这些程序通常在初始化时运行一次，以后就不再运行。而且，这些公用程序可以用它们相互共有的方式一个接一个地按顺序运行。这样，相同内存空间可以被反复使用以“召入”每一个程序，就象系统引导一样。这样可以节省内存，特别是那些配置一次以后就不再更改的系统堆栈。如果将 Linux 可加载模块的功能包括在内核里，驱动程序和应用程序就都可以被加载。由于它可以检查硬件环境并且为硬件装上相应的软件，从而消除了用一个程序占用许多 FlashMemory 来处理多种硬件的复杂性。另外，软件的升级更加模块化，可以在系统运行时在 Flash 上升级应用程序和加载驱动程序，其配置信息和运行时间参数可以作为数据文件储存在 Flash 中。

为了提高软件的开发效率和可扩展性，燃料电池 ABS 系统的软件开发建立在嵌入式 Linux 系统之上，因此开发的第一步就是进行嵌入式 Linux 系统的移植，移植是指基于一种体系结构的代码经过修改可以到另一种不同的体系结构上运行[35]。嵌入式 Linux 广泛的支持许多不同体系结构的计算机，是一个可移植性很好的操作系统。

4.1 嵌入式 Linux 内核的移植

实现嵌入式 Linux 内核的移植主要进行三个部分的工作，首先是为内核的移植做准备，包括内核、交叉编译工具的下载以及交叉编译环境的建立；其次是加载启动程序 Bootloader 的移植，最后是 Linux 内核的编译与分析。主要过程见图 4.1。

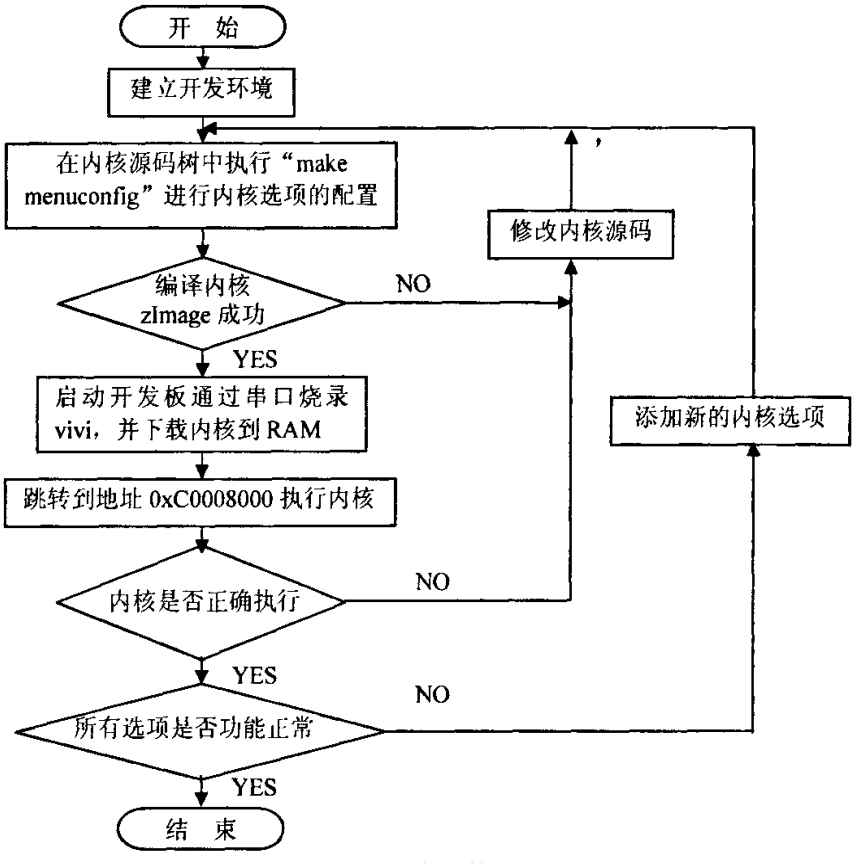

图 4.1 Linux 内核移植流程

Fig. 4.1 Linux kernel porting flow

从图中可以看出，内核的移植是一个循环往复的过程，内核的移植是后面的文件系统移植，驱动程序的移植的基础，也会对整个嵌入式系统的性能产生影响。本节主要通过具体交叉编译环境的搭建，实现了启动代码 vivi 和 Linux 内核的移植。

4.1.1 内核和交叉编译环境准备

进行 Linux 系统移植的第一步就是选择一个合适的内核版本进行下载，还要打上合适的补丁，由于本文的系统使用的开发板是基于 ARM 的，所以可以从 ARMlinux 的官方站点 http://www.arm.linux.org.uk/ 下载 ARMlinux 的源码包，也可以从 http://www.kernel.org/下载标准 Linux 内核，再给它打上该版本内核的 ARMlinux 补丁，

本系统采用的是 ARM Linux2.6.12 内核，因为它加入了多种微控制器的支持，如本课题所采用的 S3C2410 微处理器，另外，2.6 内核提高了时性能,系统的移植更加方便。

内核源码包下载完毕后，接下来就要建立交叉编译环境。所谓的交叉编译是指交叉编译是嵌入式开发过程中的一项重要技术，它的主要特征是某机器中执行的程序代码不是在本机编译生成，而是由另一台机器编译生成，一般把前者称为目标机，后者称为主机[36-37]。需要交义编译的原因有两个二，首先，在项目的起始阶段，目标平台尚未建立，因此需要做交叉编译，以生成本文所需要的 Boot Loader（启动引导代码）以及操作系统核心；其次，当目标平台能启动之后，由于目标平台上资源的限制，编译大型程序时，依然可能需要用到交叉编译[38-39]。

Linux 下的交叉编译环境重要包括以下几个部分：

(1) 针对目标系统的编译器 gcc

(2) 针对目标系统的二进制工具 binutils

(3) 目标系统的标准 c 库 glibc

(4) 目标系统的 Linux 内核头文件

首先要做的是安装 Linux 2.6.12 的头文件，然后编译安装 Binutils-2.15，接着编译安装 Gcc-3.4.2 的 C 编译器，再编译安装 Glibc-2.3.5，最后编译安装 Gcc-3.4.2 的 C、C++ 编译器。具体的制作过程可见参考文献[36]。

针对实时性的调整是虚拟内存机制的屏蔽。经过分析发现，虚拟内存是导致 Linux 实时性不强的原因之一。在控制中，一些任务要满足一定的实时性要求，屏蔽内核的虚拟内存管理机制可以增强 Linux 的实时性。当要更改内核的某项机制时，一般不必大规模地写代码，可采用条件编译的方法。同时由于 linux 系统对应用进程采用的是公平的时间分配调度算法，但这一算法也不能保证系统的实时性要求，因此要求对其进行更改。更改途径有两种：一是通过 POSIX，二是通过底层编程。笔者是通过 linux 的实时有名管道(FIFO)的特殊队列来处理实时任务的先后顺序。实际上，实时有名管道就象实时任务一样从不换页，因而可以大大减少由于内存翻页而造成的不确定延时。

4.1.2 Boot Loader 的移植

简单地说，Boot Loader 就是在操作系统内核运行之前运行的一段小程序。通过这段小程序，可以初始化硬件设备、建立内存空间的映射图，从而将系统的软硬件环境带到一个合适的状态，以便为最终调用操作系统内核准备好正确的环境[40]。最终，Boot Loader 把操作系统内核映像加载到 RAM 中，并将系统控制权交给它。

通常，Boot Loader 对硬件的依赖性非常强，特别是对于嵌入式系统。因此，在嵌入式系统里建立一个通用的 Boot Loader 几乎是不可能的。本文以 vivi 为例对 Boot Loader 的结构进行分析。vivi 是韩国 mizi 公司开发的 Boot Loader，适用于 ARM9 处理器。和大多数 Boot Loader 一样，vivi 具有两种工作模式，即启动加载模式和下载模式[25]。启动加载模式可以在一段时间后（这个时间可更改）自行启动 Linux 内核，这是 vivi 的默认模式。在系统的开发过程中通常使用下载模式，在下载模式下，vivi 为用户提供一个命令行接口，通过接口可以使用 vivi 提供的一些命令，见表 4.1：

表 4.1 vivi 常用命令

Tab. 4.1 Vivi command in common use

命 令	功 能
Load	把二进制文件载入 Flash 或 RAM
Part	操作 MTD 分区信息。显示、增加、删除、复位、保存 MTD 分区
Param	设置参数
Boot	启动系统
Flash	管理 Flash，如删除 Flash 的数据

通过 vivi 提供的这些命令用户可以方便的进行 flash 分区，加载更新文件（如：系统内核、根文件系统）以及其他一些参数的设置。

vivi 的代码包括 arch，init，lib，drivers 和 include 等几个目录，共 200 多条文件。

arch：此目录包括了所有 vivi 支持的目标板的子目录，例如本文用到的 S3C2410 开发板目录。

drivers：其中包括了引导内核需要的设备的驱动程序（MTD 和串口）。MTD 目录下分 map、nand 和 nor 三个目录。

init：这个目录只有 main.c 和 version.c 两个文件。和普通的 C 程序一样，vivi 将从 main 函数开始执行。

lib：一些平台公共的接口代码，比如 time.c 里的 udelay( )和 mdelay( )。

include：头文件的公共目录，其中的 S3C2410.h 定义了这块处理器的一些寄存器。Platform/smdk2410.h 定义了与开发板相关的资源配置参数，只需要修改这个文件就可以配置目标板的参数，如波特率、引导参数、物理内存映射等。

vivi 的运行也可以分为两个阶段 stage1 和 stage2：

stage1 完成含依赖于 CPU 的体系结构硬件初始化的代码。包括禁止看门狗计时器、禁止所有中断、初始化系统时钟、初始化串口、复制自身到 RAM 等，最后跳转到 C 代码的入口 main( )函数。相关代码集中在 head.S（\vivi\arch\S3C2410 目录下）。

stage2 从 main( )函数开始，同一般的 C 语言程序一样，该函数在/init/main.c 文件中，总共可以分为 8 个步骤。

(1) 函数开始，通过 putstr(vivi_banner)打印出 vivi 的版本。vivi_banner 在/init/version.c 文件中定义

(2) 对开发板进行初始化（board_init 函数），board_init 是与开发板紧密相关的，这个函数在/arch/S3C2410/smdk.c 文件中。开发板初始化主要完成两个功能，时钟初始化（init_time( )）和通用 IO 口设置（set_gpios( )）。

(3) 内存映射初始化和内存管理单元的初始化工作，该工作分别由 mem_map_init( )和 mmu_init( )完成，这两个函数都在/arch/S3C2410/mmu.c 文件中。

(4) 初始化堆栈，heap_init( )。（定义在\vivi\lib\heap.c 文件中）

(5) 初始化 mtd 设备，mtd_dev_init( )。定义在/drivers/mtd/maps/S3C2410_flash.c 中。

(6) 初始化私有数据，init_priv_data( )。（定义在\vivi\lib\priv_data\rw.c 文件中）

(7) 初始化内置命令，init_builtin_cmds( )。通过 add_command 函数，加载 vivi 内置的几个命令。

(8) 启动 boot_or_vivi( )启动成功后，将通过 vivi_shell( )启动一个 shell（如果配置了 CONFIG_SERIAL_TERM），此时 vivi 的任务完成。

通过上面的分析可以得到 vivi 的启动代码执行流程图如图 4.2 所示。

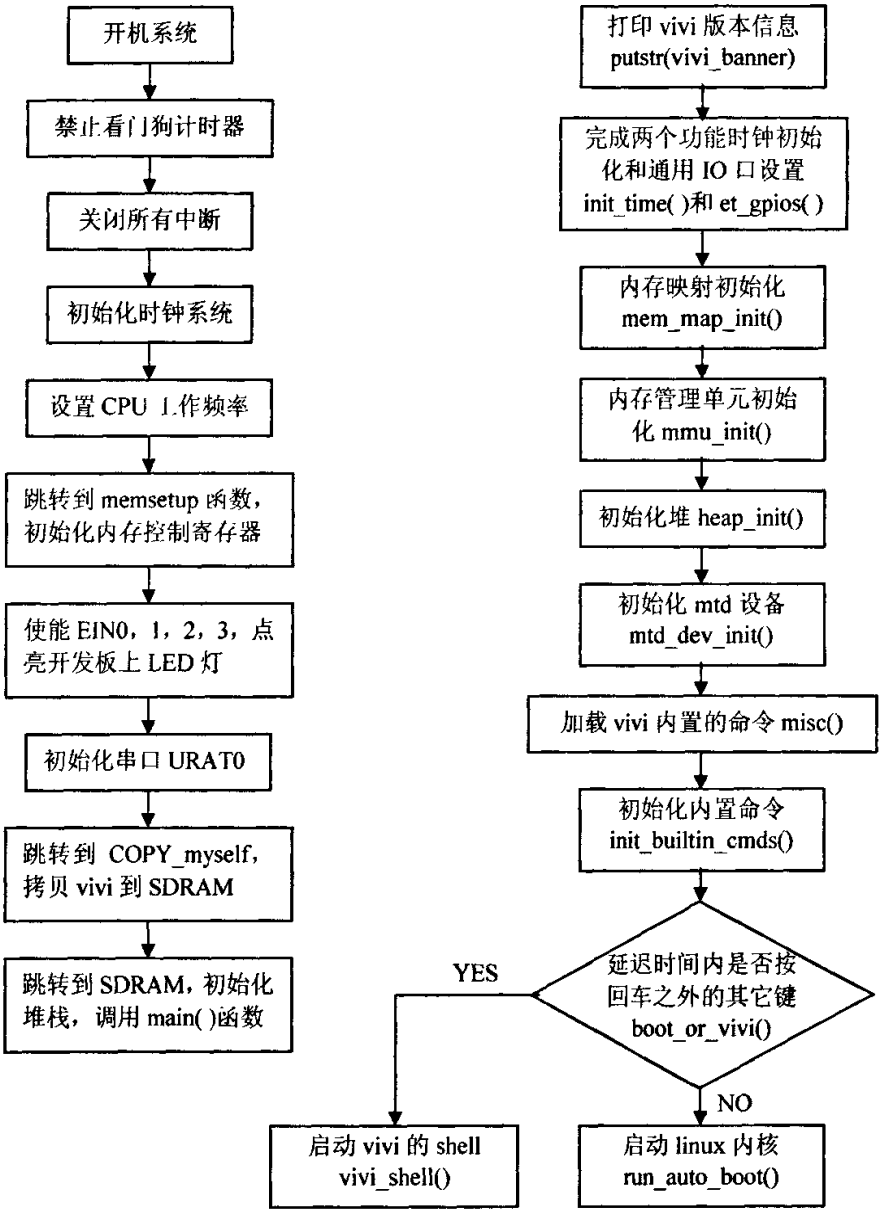

图 4.2 head.s 和 main.c 代码执行流程

Fig. 4.2 Head.s and main.c code flow of execution

针对本系统的硬件特点要对 vivi 源代码进行重新的编译和配置，首先要对源代码包解压：

#cd

#tar zxvf /mnt/cdrom/gzsd-vivi.tar.gz

其次进入 vivi 目录重新编译

#cd vivi-gzsd

#make distclean

#make menuconfig

这时会看 vivi 的配置界面，在该界面的 System Type 下选择 cs8900，增加 vivi 对网络的支持，然后后保存退出。最后运行#make，在此目录下会得到 vivi 这个文件这就是本文要用的启动加载程序映像（大约 80k）。

4.1.3 内核结构分析与移植

系统在 Boot Loader 的引导下完成自举，接下来就可以方便地下载调试和运行所需要移植的 Linux 操作系统了。多数情况下对内核的移植需要在前人工作的基础上修改已有的代码。这就需要对嵌入式 Linux 的体系结构、内核源代码树以及内核调试技术熟悉。下面就来分析一下 Linux 系统的内核结构。

Linux 内核主要由进程管理，内存管理，文件系统，网络功能，设备控制 5 个子系统组成[39]。这五部分有着复杂的调用关系，移植内核的时候需要改动的就是进程管理、内存管理和设备管理中被独立出来的与硬件相关的那部分代码。

一般在内核的每个目录下都有一个 depend 文件和一个 makefile 文件。这两个文件都是编译时使用的辅助文件。其中 makefile 文件中指出了编译时需要用到的编译器，也是移植内核过程中不可缺少的。

下图 4.3 为本系统所使用的 Linux2.6.12 内核的目录树结构，其中虚线圈起来的部分就是移植工作中需要修改的部分。下面分别对该内核的目录加以说明：

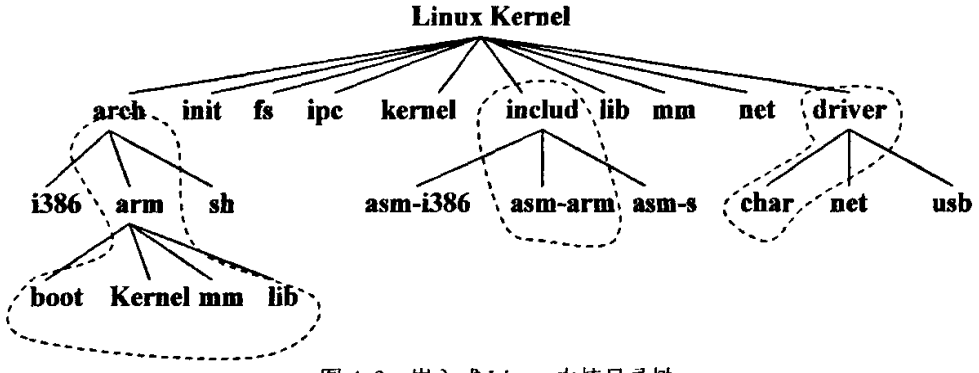

图 4.3 嵌入式 Linux 内核目录树

Fig. 4.3 Embedded Linux kernel directory tree

(1) arch 目录

Linux 系统能支持如此多平台的部分原因是因为内核把源代码清楚地划分为体系结构无关部分和体系结构相关部分。arch 目录包含了体系结构相关部分的内核代码。其中的每一个目录都代表一种硬件平台，比如本文使用的 ARM 平台和 PC 使用的 i386。对于任何平台，都必须包括以下一个目录。

boot：包括启动内核所使用的部分或全部平台特有代码。

kernel：存放支持体系结构特有的（如 SMP）特征的实现。

lib：存放高速的体系结构特有的通用函数（如 strlen）的实现。

mm：存放体系结构特有的内存管理程序的实现。

移植工作的重点就是移植 arch 目录下的文件。

(2) drivers 目录

该目录下保存了所有设备驱动程序。它占整个内核发行版本代码的一半以上。有些驱动程序是与硬件平台无关的而有些是相关的。

(3) fs 目录

该目录下列出了 Linux 支持的所有文件系统。目前 Linux 已经支持包括 ntfs 在内的多种文件系统。一般说来，文件系统与硬件无关。

(4) include 目录

这里包括了编译核心所需要的大部分头文件，例如与平台无关的头文件在include/Linux 子目录下。不同的平台需要的头文件会有所不同，因此该目录和 arch 目录一样，按平台划分了多个子目录，如 asm-arm 目录等。

(5) init 目录

init 目录下包括核心的初始化代码，有 main.c 和 version.c 两个文件。这是研究核心如何工作的好起点。

(6) ipc 目录

ipc 目录包含了核心进程间的通信代码。

(7) kernel 目录

内核管理的核心代码就在这里，与处理器结构相关的代码都放在 arch/*/kernel 目录下。

(8) lib 目录

该目录包含与平台无关的诸如 strlen 和 memcpy 之类的通用函数。

(9) mm 目录

该目录包含了所有的内存管理代码，与具体硬件体系结构相关的内存管理代码位于 arch/*/mm 目录下。

(10) net 目录

net 目录是核心的网络部分代码，其每个子目录对应与网络的一个方面。

(11) 其他目录

还有两个没有提到的目录是 documentation 和 scripts 目录。documentation 目录存放许多文档，非常详细。script 目录主要在配置内核时用到，存放了配置内核的一些脚本文件，比如“make menuconfig”命令。

下面具体介绍一下本系统使用的 Linux 2.6.12 内核在 S3C2410 开发板上的移植工作。

(1) 内核修改

Linux 2.6.12 内核加入了对 S3C2410 芯片的支持，不再需要任何补丁文件。如果是 Linux2.4 内核在 S3C2410 上的移植是相当繁琐的，因为它没有提供对该处理器的支持，

需要对新的硬件平台进行定义，如：在 arch/ARM/mach-S3C2410/*下添加 S3C2410 平台的初始化函数，在 include/asm-ARM/arch-S3C24101 下添加 S3C2410 寄存器和板子的定义，在 arch/ARM/kernel/ Makefile 下添加对 S3C2410 处理器的支持等。而这里只需要修改内核源码中 Makefile 的交叉编译选项 ARCH=arm，CROSS_COMPILE=arm-linux-。再有就是针对硬件配置,在 arch/arm/mach-S3C2410/devs.c 或者 smdk2410.c 中添加 FLASH 的分区信息 s3c_nand_info，如表 4.2。

表 4.2 NAND FLASH 分区表

Tab. 4.2 NAND FLASH partition table

分区名	起始地址	终止地址	大 小
Vivi	0x00000000	0x00020000	128k
Param	0x00020000	0x00010000	64k
Kernel	0x00030000	0x001c0000	1M+768k
Root	0x00200000	0x00200000	2M
Usr	0x00200000	0x03cf8000	60M+992k

然后在 s3c_device_nand 中增加 .dev={.platform_data= &s3c_nand_info}，在 arch/arm/mach-S3C2410/mach-smdk2410.c 中的__initdata 部分增加&s3c_device_nand,使内核在启动时初始化 NAND FLASH 信息。

(2) 内核配置

对内核进行适当的配置是一个量体裁衣的过程。由于 2.6 内核会根据本地系统配置进行初始设置,可以导入内核源码默认 S3C2410 的配置文件,方便加载内核基本配置,然后再选择所需选项。

cp arch/arm/configs/smdk2410_defconfig.config

make menuconfig

在配置时,大部分选项可以使用其缺省值,只有小部分需要根据用户不同需要选择。选择的原则是将与内核其它部分关系较远部分且不经常使用的部分功能代码编译成为可加载模块,有利于减小内核的长度,减小内核消耗的内存,简化该功能相应的环境改变时对内核的影响。主要是进行以下几项配置：选择处理器类型；对 MTD 配置选择支持 MTD 设备驱动以及 NAND FLASH 驱动；选择支持要用到的各类文件系统（DEVFS、

TMPFS、CRAMFS、YAFFS、EXT2、NFS，本课题选择的是 CRAMFS 和 YAFFS）；选择网络设备和协议，本系统加载了网络芯片 CS8900 支持。

(3) 内核的编译

通常在 Linux 中，内核映像分为压缩的内核映像和未压缩的内核映像，压缩的内核映像通常名为 zImage，位于 arch/arm/boot 目录中。而未压缩的内核映像通常名为 vmlinux，位于源码树的根目录中。为了适应嵌入式的需要,采用压缩方式编译：

#make zImage

这种方式下，编译好的内核映象文件先被烧写到 Flash 内，启动时由引导程序将内核从 Flash 里加载到系统 RAM 中解压,然后运行。本课题所生成的内核映象不到 1M，因而这种压缩方式编译出来的内核非常适合嵌入式系统使用。

4.2 根文件系统的移植

文件系统是基于被划分的存储设备上的逻辑上单位上的一种定义文件的命名、存储、组织及取出的方法。如果一个 Linux 没有根文件系统，它是不能被正确的启动的。因此，需要为 Linux 创建根文件系统。

Linux 的根文件系统可能包括如下目录（或更多的目录）：

(1) /bin (binary)：包含着所有的标准命令和应用程序；

(2) /dev (device)：包含外设的文件接口，在 Linux 下，文件和设备采用同种方法访问的，系统上的每个设备都在/dev 里有一个对应的设备文件；

(3) /etc (etcetera)：这个目录包含着系统设置文件和其他的系统文件，例如/etc/fstab(file system table)记录了启动时要 mount 的 filesystem；

(4) /home：存放用户主目录；

(5) /lib(library)：存放系统最基本的库文件；

(6) /mnt：用户临时挂载文件系统的地方；

(7) /proc：linux 提供的一个虚拟系统，系统启动时在内存中产生，用户可以直接通过访问这些文件来获得系统信息；

(8) /root：超级用户主目录；

(9) /sbin：这个目录存放着系统管理程序，如 fsck、mount 等；

(10) /tmp(temporary)：存放不同的程序执行时产生的临时文件；

(11) /usr(user)：存放用户应用程序和文件。

采用 BusyBox 是缩小根文件系统的好办法，因为其中提供了系统的许多基本指令但是其体积很小。从网上下载并编译好 busybox 后，将其放入/bin 目录，若要使用其中的命令，只需要建立 link，如：

#ln –s ./busybox ls

#ln -s ./busybox mkdir

目前嵌入式 Linux 常用的文件系统有 CRAMFS、JFFS、JFFS2、YAFFS 等，为保护系统的基本设置不被更改，本文采用的是 CRAMFS 格式的根文件系统，CRAMFS（compressed ROM file system）是 Linus Torvalds 本人开发的一个适用于嵌入式系统的小型经压缩、只读(Read-Only)文件系统，它是只具备最基本特征的文件系统。CRAMFS 文件系统在启动后时它并不需要一次性地将文件系统中的所有内容都解压缩到内存之中，而只是在系统需要访问某个位置的数据的时侯，马上计算出该数据在 cramfs 中的位置，将其实时地解压缩到内存之中，然后通过对内存的访问来获取文件系统中需要读取的数据。这样节约了嵌入式系统有限的存储资源。CRAMFS 文件系统制作过程如下：

(1) 首先要安装 CRAMFS 工具 mkcramfs；

(2) 制作 CRAMFS 文件系统 image。建立一个目录，将需要放到文件系统的文件 copy 到这个目录，运行"mkcramfs 目录名 image 名"就可以生成一个 cramfs 文件系统的 image 文件。例如：目录名为 rootfs，则正确的命令为：

$mkcramfs rootfs rootfs.ramfs

(3) 复制文件系统映像到指定的存储体上。本文在建立 TFTPserver 的基础上使用 vivi 提供的 net tftp 命令将根文件系统烧录映像（大小 1.2M）到 nand flash 的 root 分区。

4.3 设备驱动程序的移植

Linux 系统内核通过设备驱动程序与外围设备进行交互，设备驱动程序是 Linux 内核的一部分，它是一组数据结构和函数，这些数据结构和函数通过定义接口控制一个或多个设备。对用户程序而言，设备驱动程序封装了控制设备的具体技术细节，对各种不同设备提供一致的接口。本章这一小节就以课题中 CAN（Controller Area Network 控制

局域网）网络的通讯接口 CAN 控制器的驱动程序为例来分析 Linux 设备驱动程序的移植。

燃料电池电动公交车 ABS 系统和中央控制器通讯是通过 CAN 网络实现的。而本系统所使用的 S3C2410 开发板本身不带有 CAN 控制器，需要自己扩展，这里采用上一章介绍的带有 SPI 接口的独立的 CAN 控制器 MCP2515。

如图 4.4 所示，CAN 控制器在 Linux 下的驱动程序为 CAN 总线应用程序访问 MCP2515 提供了一座桥梁。

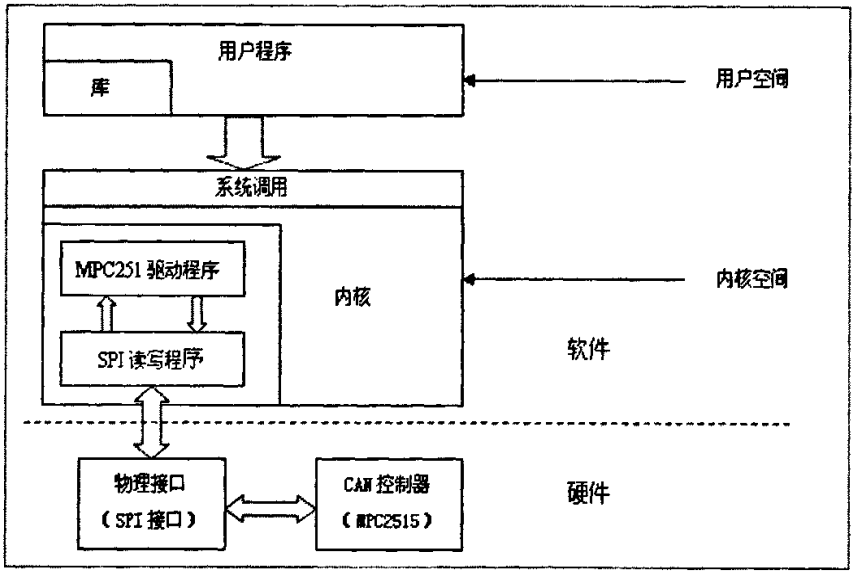

图 4. 4 CAN 控制器驱动程序在系统中的地位

Fig. 4.4 Station of CAN controller driver in the system

Linux 有四种设备驱动：字符型设备驱动、块设备驱动、终端设备驱动和流设备驱动。字符型设备的驱动一个字节接一个字节的将信息从用户到设备（反之亦然）[40]。CAN 控制器属于字符型设备，它的驱动程序的模块化程序设计可以分为以下四个主要组成部分[41-43]：

4.3.1 初始化加载和卸载部分

设备驱动程序通常以模块方式加入内核，因而一定会包含加载和卸载两个模块，加载模块在系统初始化时调用。

4.3.2 服务于 I/O 请求的 CAN 设备基本入口点

这部分是设备驱动程序对应用程序的接口，也是应用程序进行系统调用唯一可见的部分。在 Linux 系统中每一个设备都用一个特殊设备文件名来表示。这个抽象界面的主体就是一个抽象的 file_operations 数据结构。根据系统需求，CAN 设备驱动程序实现了部分重要的设备方法，具体定义如下：

```
struct file_operations Fops = {
    .open = device_open,
    .release = device_release,
    .read = device_read,
    .write = device_write
};
```

4.3.3 中断服务子程序

Linux 中的中断处理程序很有特色，它的一个中断处理程序分为两个部分：上半部和下半部[44]。这样划分是考虑到中断处理的效率。上半部的功能是"登记中断"，它把设备驱动程序中断例程的下半部挂到该设备的下半部执行队列中去，由于它执行时完全屏蔽其它中断，这部分程序要求短小快速。而对于那些中断事件的处理是比较复杂的，Linux 引入了下半部的概念[45]。下半部和上半部最大的不同是下半部是可中断的，而上半部是不可中断的。

5 ABS 模糊控制器的设计

5.1 ABS 模糊控制器的设计

汽车防抱死制动系统(ABS)的控制目标是把车轮的滑移率限制在对应最大路面附着系数的范围之内，从而使车辆获得最大的地面制动力，同时具有较好的横向稳定性。由于轮胎是车辆与地面作用的物质载体，两者之间的力作用特性决定了车辆的运动。轮胎的力特性呈非线性关系，这给防抱死系统的控制带来了困难。汽车防抱死制动系统的控制方法有多种，包括逻辑门限值控制方法、最优控制方法、滑模变结构控制方法以及鲁棒控制方法等。在现售的汽车防抱死制动系统中，大多采用逻辑门限值控制方法。采用该方法时，需要确定汽车车轮的加减速度、纵向滑移率和制动压力的变化等参数。逻辑门限值控制方法是基于经验知识的，它对经验知识的处理仍是基于二值逻辑来进行数据信息的优化和判断。而对于经验知识的处理采用语言信息描述会更加准确。模糊控制正好适合对人类的经验知识用语言描述并进行准确的处理。对防抱死制动系统这种具有时变、非线性的复杂系统采用模糊控制时，为了获得良好的控制效果，必须要求有完善的控制规则。这些控制规则是人们对被控制过程的模糊信息的归纳和操作经验的总结。在模糊控制器的设计中，模糊规则的获取和控制器参数的调整都无系统的方法，主要依靠控制专家的经验和设计者的反复实验确定。根据第2章ABS控制算法比较的结果，本文仍选着模糊控制的方法。

5.1.1 电动公交车 ABS 系统数学模型的建立

(1) 汽车动力学模型

基于单轮的1/4汽车模型可满足各位置车轮单独ABS控制参数理论设计的要求，由如下车体、车轮力矩平衡式和滑移率 λ 的定义式描述。

$$mv = -f_b - f_w \tag{5.1}$$

$$I_w\omega = -T_b + f_{bR} + f_{RR} \tag{5.2}$$

$$\lambda = (v - \omega R)/v \tag{5.3}$$

式中m为1/ 4汽车质量， I_w为车轮转动惯量，R为车轮半径，λ 为滑移率，v为车速，ω 为车轮转速，T_b为制动力矩，f_b为地面制动力，f_w为汽车风阻，fR为汽车滚阻。

（2） 轮胎力计算模型

为了比较准确地模拟不同附着系数下轮胎路面之间的相互作用，采用Magic Formula计算轮胎作用力，即

$$Fx = Dsin\{ Carctan[B (1 - E) (\lambda + S h) + Earctan (B\lambda + BS h)]\} + Sv \qquad (5.4)$$

式中Fx为轮胎纵向力，B、C、D、E、Sh和Sv为特征系数，必须通过试验获得。

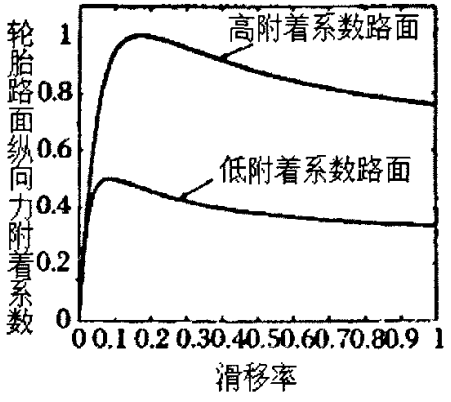

图5.1 不同路面附着系数的变化

Fig. 5.1 The change of accrete coefficient on different road

利用Magic Formula式计算得到的不同路面附着系数随滑移率的变化曲线如图4.1。ABS的功能是把滑移率控制在附着系数的峰值点附近。

（3） 制动系统模型

ABS液压系统是一本质非线性系统，建立其精确的数学模型相当困难。将液压系统理想化为无滞后的线性压力变化过程，会给ABS动力学仿真带来很大的误差。通过试验数据，采用回归分析的方法，建立了ABS液压系统的辨识模型，比较准确地描述了制动轮缸压力随制动主缸压力的变化关系。辨识模型的形式为：

$$\frac{dPw}{dt} = \frac{1}{Ce\,\mathrm{Re}}(Pm - Pw)^{k} u1(t - \tau_{vp}) - \frac{1}{CeR'e}(Pm - Pw)^{k'} u1(t - \tau'_{vp}) \qquad (5.5)$$

式中Pw 和Pm 分别为轮缸和主缸的制动压力， Ce为等效液容， Re和R′ e分别为增压和减压时的等效液阻，κ 和κ ′ 分别为增压和减压时的节流指数，t为系统工作时间，

τvp和τ′vp分别为增压和减压时的电磁阀和管路传输滞后时间，u1和u2为电磁阀控制指令信号。系统增压时，取u1 = 1， u2 = 0；系统减压时，取u1 = 0，u2 = 1；系统保压时，取u1 = 0，u2= 0 。

轮缸压力作用于活塞，并通过制动器产生制动力矩。计算制动力矩时必须考虑制动器动态特性的影响。根据盘式制动器的结构，得其数学模型用传递函数表示为：

$$\frac{T_b(s)}{P_w(s)} = \frac{K_d A_w r_d}{1 + \frac{2s\zeta\omega_n + s^2}{\omega_n^2}} \tag{5.6}$$

式中Tb为制动力矩，Pw 轮缸制动压力， Kd为效能因数， A w为活塞面积，rd有效制动半径，ζ和ωn分别为制动器的阻尼系数和固有频率，均通过试验获得。

5.1.2 ABS控制策略及模糊控制器的设计

成熟的ABS产品几乎都采用逻辑门限的控制方式，利用车轮加减角速度门限及参考滑移率的组合，构成控制逻辑，把滑移率调整在峰值附着系数附近波动。文中参照Bosch公司公开发表的逻辑门限控制策略，设计控制逻辑如图5.2。

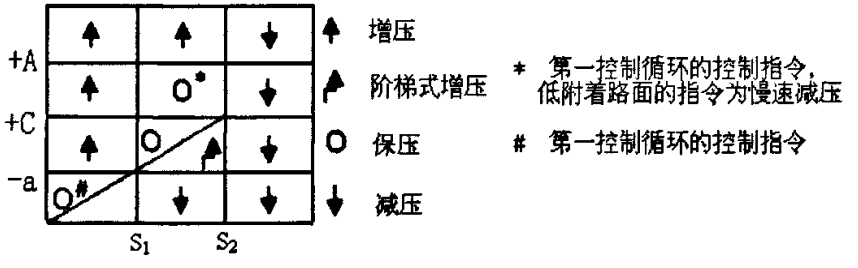

图5.2 ABS限辑门限控制规律

Fig. 5.2 The rules of ABS controlled by logic gate limit

图中-a为车轮角减速度门限值，+C和+A分别第一、第二角加速度门限值，S_1和S_2分别为第一、第二参考滑移率门限值，而且高附着路面和低附着路面各参数的取值不同。参数的不同取值，将获得不同的控制效果。在ABS控制器的开发过程中，通过大量实车试验调整的方法来获得合适的参数设计值，以致于设计周期长，耗费大。

采用基于滑移率的控制方式，以车轮滑移率参考值的跟踪为控制目标。通过设定参考滑移率，设计模糊控制器调节车轮的制动压力，使车轮的实际滑移率跟踪设定值。由于各种路面的最优滑移率在8%～25%之间，故选定参考滑移率值为20%。利用模糊逻辑进行ABS控制的结构框图如图5.3所示。

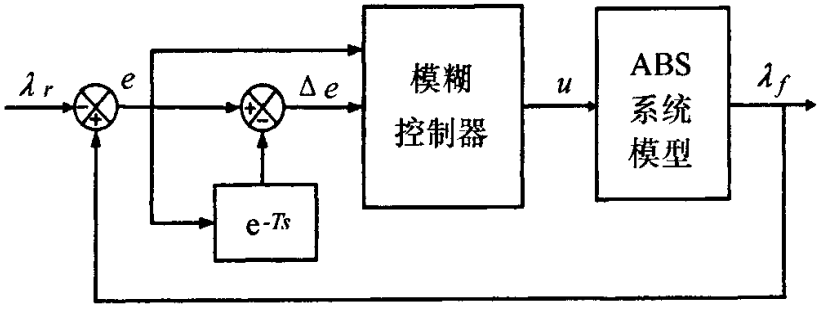

图5.3 ABS模糊控制框图

Fig. 5.3 The fuzzy control frame of ABS

模糊控制器的输入量为实际滑移率λf 与参考滑移率λr 的差值和差值的变化，即

$$e(t)=\lambda f(t)-\lambda r(t) \tag{5.10}$$

$$\Delta e(t)=e(t)-e(t-T) \tag{5.11}$$

式中， T为设计参数，表示采样时间。图5.3中的e^{-Ts}为δ(t-T)的拉氏变换，图中加该项后表示误差信号前一采样时刻对应的值即e(t-T)。模糊控制器的输出量为压力的变化量u 。描述输入、输出控制量的语言值模糊子集选取为:{NB 、NS 、ZE、PS 、PB}其中: NB =负大； NS = 负小； ZE = 零； PS = 正小； PB = 正大。滑移率差值的论域为E ，将其变化范围- 20%～20%量化为9个等级，量化因子Ke = 0.2，则有E = {-4 ，-3 ，-2 ，-1 ，0 ，1 ，2 ，3 ，4} 。滑移率差值变化的论域为ΔE ，将其变化范围- 10 %～10 %量化为9 个等级，量化因子KΔe = 0.4 ，则有ΔE = {-4 ， -3 ，-2 ， -1 ，0 ，1 ，2 ，3 ，4} 。输出量的论域为U ，将它的变化范围- 1～1 也量化为9 个等级， 其比例因子为Ku = 0.25 ， 则有U ={-4 ，-3 ，-2 ，-1 ，0 ，1 ，2 ，3 ，4} 。输入、输出控制量的隶属度函数的形状如图4.4所示，依据专家经验建立反映滑移率及其变化量与控制压力之间关系的模糊控制规则见表5.1所示。

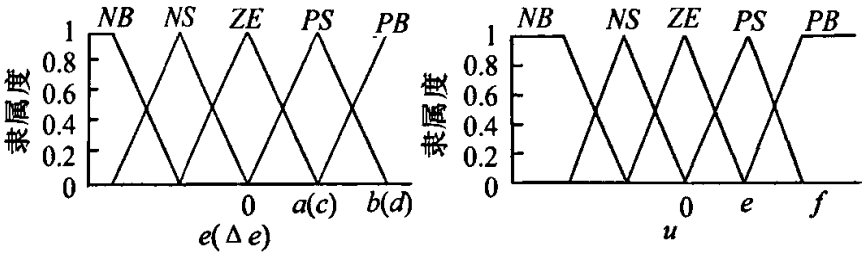

图5.4 输入、输出量的隶属度函数

Fig. 5.4 The subjection function of input and output value

表5.1 用语言变量表示的模糊控制规则表

Tab. 5 .1 The fuzzy control rule table indicated by language variable

ΔE	E				
	NB	NS	ZO	PS	PB
NB	PB	PB	PS	PS	ZO
NS	PB	PS	PS	ZO	ZO
ZO	PS	PS	ZO	ZO	NS
PS	PS	ZO	ZO	NS	NS
PB	ZO	ZO	NS	NS	NB

本文采用Mamdani（Max-Min）推理进行逻辑运算,采用重心法进行解模糊计算。

6 ABS系统软硬件的设计与测试

6.1 设计背景

现在世界上很多汽车公司和零部件生产厂家都在致力于提高汽车行驶的安全性能，特别是致力于提高汽车行驶的主动安全性能，这已经成为汽车电子化发展的一个重要方面。

汽车制动防抱死系统（Antilock Braking System），简称ABS，是汽车主动安全装置的代表，其作用是在制动过程中防止车轮抱死，提高车辆在制动过程中的方向稳定性、转向控制能力和缩短制动距离，使汽车制动更为安全有效。

基于ABS系统，可以开发出更多的车辆电子控制系统。例如电子制动力分配系统（Electronic braking force distribution，EBD），汽车行驶稳定性控制系统（Electronic stability program,ESP)，下坡辅助控制系统(Downhill assist control，DAC），坡道起步辅助控制系统（Hill-start assist control，HAC），转向制动控制系统（Corneringbrake control，CBC），汽车信息记录仪（Event data recorder，EDR）等。

汽车ABS系统是一个新的汽车主动安全行驶的发展方向，ABS的执行机构可完全满足ABS系统的需要，传感器方面只需添加探测主车与目标车辆信息的车距传感器，在ABS ECU的基础上，对硬件电路进行相应的扩充和软件控制逻辑的有机融合，就可实现ABS系统的进一步扩展。

2005年，我们承担了沈阳骐骥电动车研究所项目“燃料电池电动公交车ABS系统仿真研究”，进行了ABS系统的研究和开发。以前采用Intel的80C196KB芯片，功能不足，后改用了ARM9芯片，功能完善，性能可靠，实车控制效果好。

6.2 ABS系统设计

在燃料电池电动公交车上，开发一种客车ABS系统的ECU，并通过预留相关的通讯和控制接口，初步实现ABS系统的功能。该集成系统主要包括电控单元ECU、传感器、执行机构三个部分。图6.1为汽车ABS系统示意图。

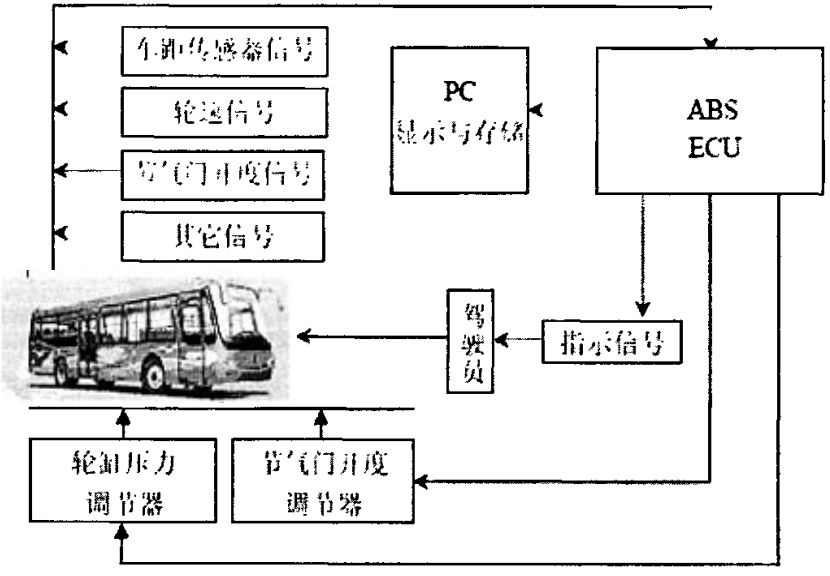

图6.1 ABS系统示意图

Fig. 6.1 The sketch map of ABS system

当汽车正常行驶时，ABS系统ECU实时采集和处理传感器信号，包括四个轮速信号、蓄能器压力信号、节气门开度信号、加速踏板开度信号、制动踏板开关信号、节气门怠速开关信号、与目标车辆间的相对距离和相对速度信号，并根据其所提供的信息，选用不同的控制方式对汽车进行控制。控制的方式包括以调节车轮轮缸压力为目标的制动干预模式和以调节发动机输出力矩为目标的节气门开度控制模式，各子系统功能的实现就是对以上两种控制方式适当组合和合理控制的结果。

ABS系统具有实时故障诊断功能，当系统出现故障时，执行机构及时复位，故障报警灯点亮，提醒驾驶员注意和及时处理。对于集成系统的ECU和传感器部分将在后面详细介绍，这里仅对集成系统的执行机构作以简单描述。

针对某些样车4×2 前轮驱动方式和交叉双制动管路布置方案，基于原车装配的Teves MKII型ABS压力调节器，进行了车辆制动系统的设计改造，添加了外部液压动力源，通过工作模式电磁阀22、23实现ABS制动方式和主动制动干预方式的切换，如图6.2所示。

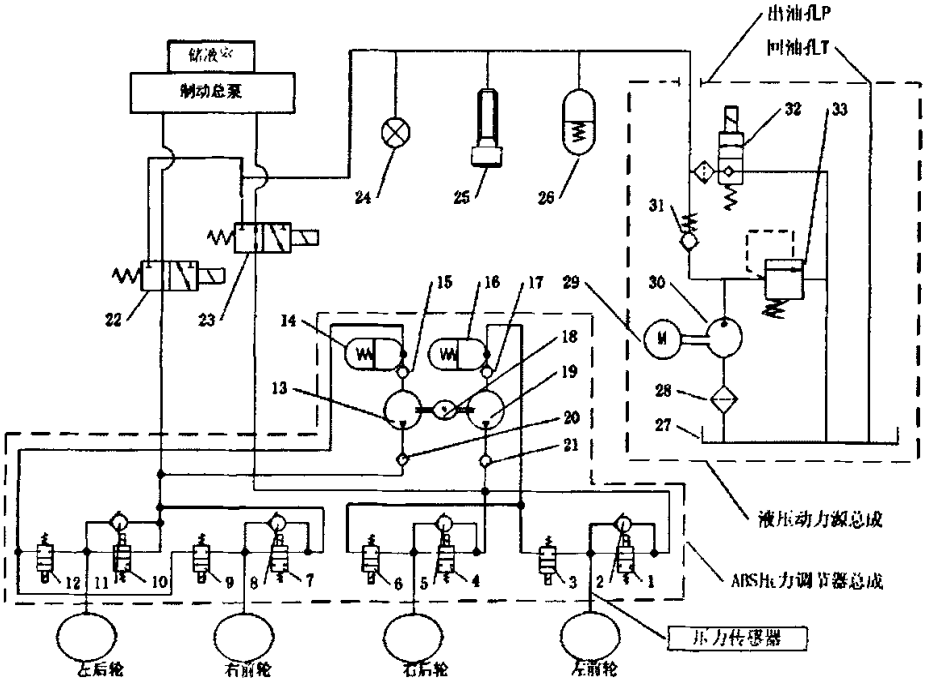

图6.2 ABS液压系统原理图

Fig. 6.2 The schematic diagram of ABS hydraulic pressure system

1、左前轮进油电磁阀；2、左前轮卸压单向阀；3、左前轮回油电磁阀；4、右后轮进油电磁阀；5、右后轮卸压单向阀；6、右后轮回油电磁阀；7、右前轮进油电磁阀；8、右前轮卸压单向阀；9、右前轮回油电磁阀；10、左后轮进油电磁阀；11、左后轮卸压单向阀；12、左后轮回油电磁阀；13、19、ABS 回油泵；14、16、ABS 低压储液器；15、17、20、21、ABS 回油单向阀；18、ABS 电机；22、23、工作模式电磁阀；24、压力表；25、压力传感器；26、蓄能器；27、压力继电器；28、油滤；29、ASR 电机；30、液压泵；31、液压泵单向阀；32、卸荷电磁阀；33、安全阀

针对ABS功能要求的发动机输出力矩调节，采用了步进电机控制节气门的改装方案，如图6.3所示。集成系统ECU根据加速踏板传感器的输入信号，运用PID算法，驱动步进电机对节气门进行伺服随动调节；当它判断需要进行发动机输出力矩调节时，计算出节气门需要转过的目标角度，发出控制指令，驱动步进电机使节气门转到目标开度。

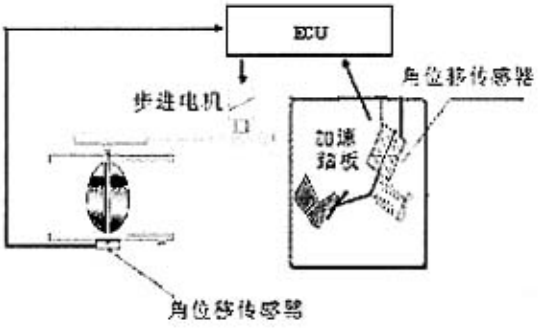

图6.3 发动机输出力矩节气门开度调节方案图

Fig. 6.3 The project diagram adjusted by solar term opening of engine output moment

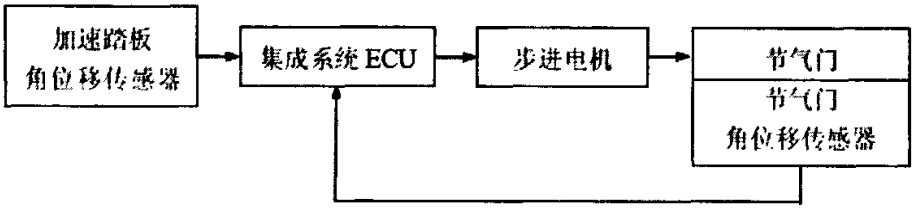

图6.4 发动机输出力矩节气门开度调节工作原理图

Fig. 6.4 The schematic diagram adjusted by solar term opening of engine output moment

集成系统的执行机构具备以下特点和功能：

1）改造后的汽车ABS系统的执行机构，不影响原车ABS执行机构的正常工作；

2）能够进行驱动轮制动干预控制和发动机输出力矩调节，能够很好的配合ASR驱动防滑控制；

3）添加车距传感器后，结合各个车轮的独立车轮制动干预控制和发动机输出力矩调节，有助于进行ACC自适应巡航控制；

4）集成系统具备外部的动力源，可主动对四个车轮独立进行制动干预控制，使得系统有很好的扩展性。在现有集成系统执行机构的基础上，仅通过修改和完善电控系统的硬件电路和软件逻辑就能实现更多的集成控制功能，例如，坡道起步辅助控制系统HAC (Hill-start assist control)，下坡辅助控制系统DAC(Downhill assist control)，转向制动控制系统CBC (Cornering braking control)等；

5）改造后的汽车ABS系统的执行机构，不影响汽车的所有使用性能，如执行机构发生故障，仍可保证车辆具有常规的制动功能。

6.3 ABS系统硬件电路

ABS系统ECU的硬件结构框图如图6.5所示，包括以ARM9芯片为核心的最小系统，传感器处理电路、执行机构驱动电路，以及通讯接口电路。

6.3.1 主控制芯片ARM9

MCU是整个系统的核心，它负责数据的采集和处理，所有的逻辑运算以及最终控制的实现。ABS系统对其运算能力、存储空间、I/O接口都有较高的要求。经过调研，我们选择了ARM公司的ARM9芯片。它有很强的运算能力，丰富的IO接口和充裕的存储空间，该芯片资料丰富，性价比高，最高处理速度为203MHz,是同ARM处理器中较快的一款。它采用的是RISC架构，因而具有如下特点：体积小，功耗低，成本低，性能高；支持Thumb（16位）/ARM（32位）双指令集；大量使用寄存器，使指令执行速度更快；寻址方式灵活简单，执行效率高；指令长度固定。

另外，利用ADS 1.2（ARM Developer Suite） 集成开发环境,通过背景调试方式（Background debug mode，BDM），下载控制程序和修改相关参数，在不干扰目标程序运行情况下，实时监测各寄存器和存储器，实现了控制程序的板上在线调试，从而提高了集成系统的开发效率和试验的方便性，缩短了试验周期。

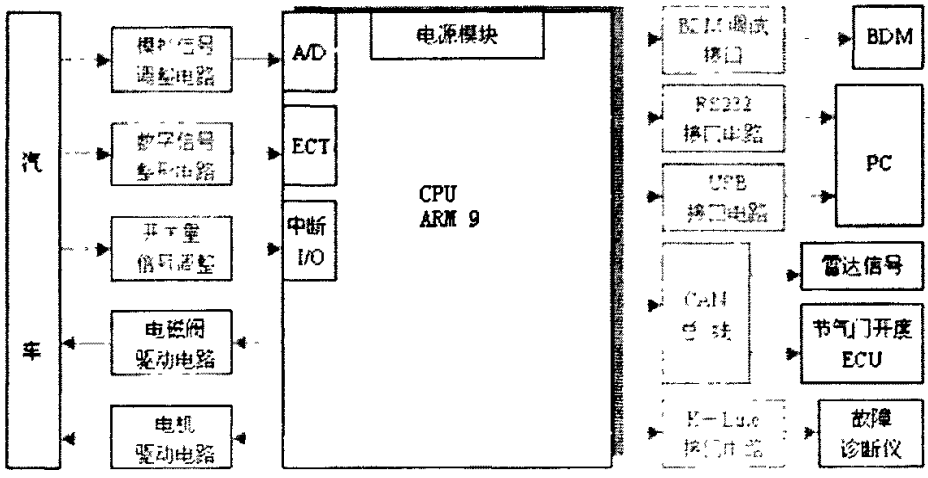

图6.5 系统ECU结构框图

Fig. 6.5 The structure of the ECU system

6.3.2 外围电路

硬件外围电路部分采用了模块化设计，即把电控单元划分成不同的模块，将比较成熟固定的模块组成一块单独的电路板，研究过程中变化较大的模块组成另一块单独的电路板，板与板之间通过IO扩展线进行通信。这种设计具有以下优点：

(1) 有利于试验过程中对系统的维护和扩展。试验过程中有很多不确定的因素，有时某一部分损坏需要进行更换，有时为了试验的需要还要另外添加硬件电路，如果把整个系统做在一块电路上，一旦需要更改电路或者对系统进行扩展，就必须重新设计整块电路，这样会造成时间和金钱上的浪费。采用模块化设计，只需在相应模块上要改变或添加即可，避免了许多重复的劳动，缩短了设计周期。

(2) 从电气特性上看，模块化设计有更好的电气特性。各个模块之间是隔离的，这样各个模块之间就不会有相互影响。例如，驱动模块是这些模块中最大的干扰源，尤其对A/D转换和、MCU的工作影响很大。为此在设计中，对驱动模块单独供电，与其有关的信号线采取了隔离措施，这样就不会对其它模块部分造成电气干扰影响。根据ABS系统的特点和开发经验，将整个电控单元分为了A板和B板，两块电路板之间通过IO 总线扩展插槽连接。

1） A板介绍

A板主要包括主控芯片ARM9及其最小系统外围电路、通讯接口电路和主要的数据采集电路。

ARM9最小系统包括电源模块、外围复位电路、时钟晶振电路、工作模式选择等。

通讯接口电路包括BDM接口，两路SCI串口通讯接口电路，其中一路通过硬件跳线选择可连接K-LINE故障诊断驱动芯片MC33199或与外部设备通讯，另一路预留扩展USB通讯；利用CAN驱动芯片PCA82C250引出两路CAN通讯节点，一路预留给车距信号通讯，另一路与其它要求高速数据传输的控制单元通讯。

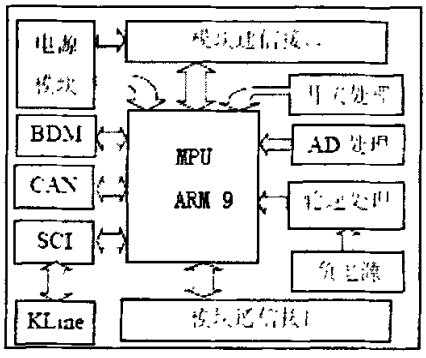

图6.6 A板电路结构示意图

Fig. 6.6 The sketch map of A board circuit

ABS系统设计了数字量、模拟量和开关量采集处理电路，考虑到集成系统的扩展性，设计采集信号如表6.1所示。将设计比较成熟固定的轮速信号、模拟信号处理电路放在A板上，而把其它使能控制信号及其显示电路和开关量信号处理电路放在B板上。

表6.1 集成系统采集信号

Tab. 6.1 The collection signal of the integration system

名称	通道	型号	备注
车轮轮速	4	电磁式	A板
蓄能器	1	MSP-300-250-B-5-W-1	A板
节气门开度	1	电位式	A板
加速踏板开度	1	电位式	A板
制动踏板开关	1		B板
节气门怠速开关	1		B板
ASR使能	1		B板
手刹开关	1		B板

2） B板介绍

B板主要包括执行机构驱动电路和开关量信号处理电路。

系统驱动的执行机构主要包括ABS压力调节器的四个常开进油阀（驱动电流3.6A），四个常闭出油电磁阀（驱动电流2.4A），两个工作模式切换电磁阀（驱动电流2.2A），一个ABS 电动机驱动开关（驱动电流8.4A），一个ASR电动机控制开关（驱动电流1.8A）。ASR、HAC子系统的使能信号及工作指示灯部分的相关电路也放在该电路板上。

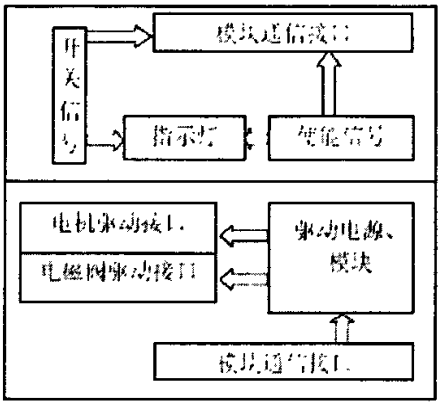

图6.7 数据采集板主要结构示意图

Fig.6.7 The sketch map of the data collection board

6.4 ABS 系统软件设计

汽车ABS系统中每一子系统都是复杂的控制系统，集成后形成的集成化系统更是一个非常复杂的控制系统，软件系统是其核心部分，它决定着整个系统运行的好坏、控制效果的优劣。软件系统的设计并不是把很多子控制系统简单的累加，而是要把它们有机的融合，还要考虑到软件运行的实时性、可靠性，控制算法的优化等问题。

汽车ABS系统的软件模块主要由系统初始化模块、启动自检模块、主控制模块、数据采集模块、数据处理模块、参考车速计算模块、路面识别模块、车辆运动状态识别模块、控制决策和执行机构动作模块、故障诊断模块、通讯模块等几大部分组成，各模块由主控制模块按任务管理机制实时进行统一调度，分配运行时间，进行数据和信号的交换。总体框图如图6.8所示。

6.4.1 系统初始化模块

系统初始化模块在系统上电复位时对系统进行初始化。初始化内容包括MCU内部的时钟、各端口设置、串行通讯接口、模拟和数字通道、看门狗定时器、系统变量等，以保证MCU 正常运行。对集成系统的执行机构进行复位，确保车辆的安全行驶。

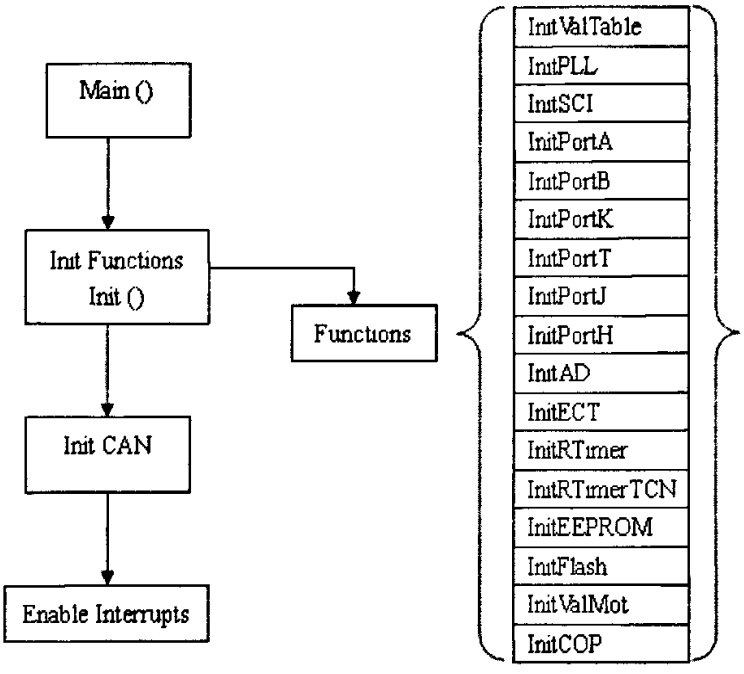

图6.8 系统初始化模块

Fig.6.8 The initial module for the system

入口函数Main()首先调用系统初始化函数Init( )，对系统进行初始化，Init( )包括以下函数:

InitValTable:对一些变量和中断向量表进行初始化。

InitPLL:对MCU内核和外部总线运行频率进行设定。

InitSCI:串口初始化。目前只使用了串口1，所以只对串口1进行初始化。

InitPortA:设置PA口数据传输的方向。

InitPortB:设置PB口数据传输的方向。

InitPortK:设置PK口数据传输的方向。

InitPortT:设置PT口数据传输的方向。

InitPortJ:设置PJ口数据传输的方向。

InitPortH:设置PH口的数据传输方向。

InitAD:设置AD转换模式。

InitECT:初始化超强定时器。

InitRTimer:初始化硬件时钟。

InitTimeTCNT:初始化时钟计数器。

InitEEPROM:寄存器初始化。

InitFlash:初始化Flash。

InitValMot:执行机构初始化。

执行机构的初始化就是要在ABS起作用之前，压力调节器的阀和电机都不能动作，即常开阀保持打开，常闭阀保持关闭，电机不运转，阀初始化只要把阀的总控制MOSFET管关闭即可。

InitCOP:看门狗定时器初始化。

看门狗主要用于检测软件运行出错，在软件中的适当位置加上清除看门狗定时器的指令，如果在监视定时器溢出之前不对看门狗定时器清零，则表示软件运行出错，需要产生系统复位。目前并没有用到看门狗的这个功能，只是把看门狗用做软件复位，当需要软件复位时，只需要打开看门狗的功能，把看门狗定时器的时间设置为只有一个时钟周期，因此，系统会立即复位。

6.4.2 系统启动自检模块

为了使系统安全运行，系统在复位和初始化完毕后要进行自检，对系统的关键软、硬件部分进行静态检测，以判断系统的软、硬件工作是否正常。在系统自检期间，故障警示灯将会自动点亮，如果电子控制装置发现系统中存在故障，电子控制装置将会以故障代码的形式存储故障情况，故障警示灯会持续点亮，执行机构复位至常规状态；如果未发现系统中存在故障，在自检过程结束后，故障警示灯在1.5秒后会自动熄灭，表示集成系统处于正常运行状态。

系统启动自检的内容主要包括对已存故障信息的复查检测和对制动干预执行机构的检测，在轮速大于15km/h 后进行轮速信号的检测。如果已存故障由轮速信号故障引起，启动自检时故障警示灯将持续点亮，直到轮速大于15km/h后进行复查检测，如故障排除，故障警示灯会自动熄灭，记录该故障为临时性故障。

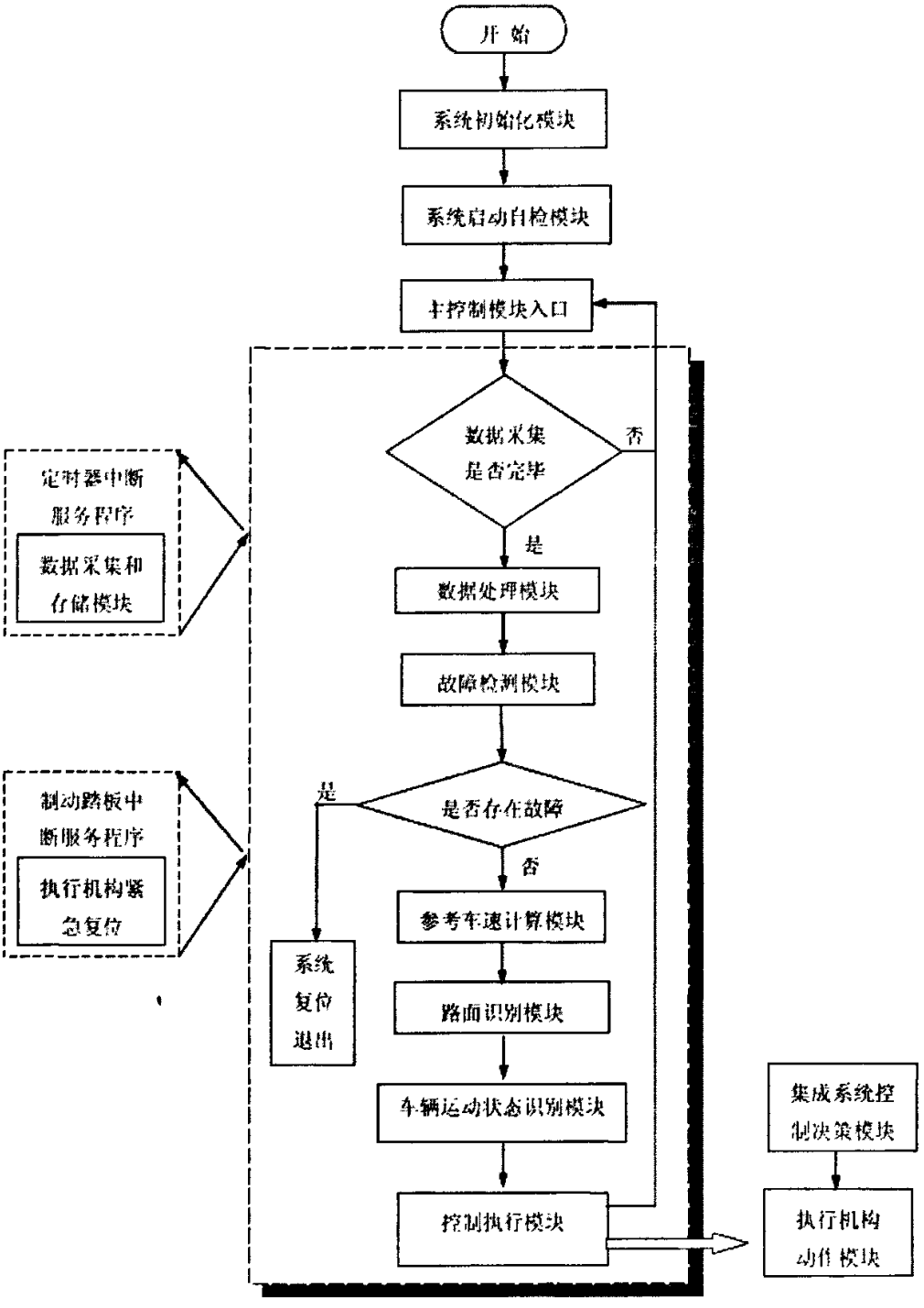

图6. 9 ABS系统软件结构简图

Fig. 6.9 The software structure sketch of ABS system

6.4.3 主控制模块

主控制模块为ABS系统的控制主程序，见图6.9阴影虚线框。主控制模块自身做无限的循环，连续调用故障诊断模块、数据处理模块、参考车速计算模块、路面识别模块和车辆运动状态识别模块，实时进行车辆运动状态和外界环境的判断。

6.4.4 数据处理模块

完成对采集到的数字和模拟信号进行滤波和平滑处理，剔除奇异点和干扰信号，保证采集信号的准确性。这里主要针对四个车轮的轮速信号进行的异点剔除和三点滑动平均法处理。

6.4.5 故障诊断模块

故障诊断模块在集成系统运行的过程中实时检测执行机构各电磁阀和电机的故障反馈端，车轮轮速信号。轮速的检测采用下式：

$$\|\omega_{lf}-\omega_{rf}|-|\omega_{lr}-\omega_{rr}\|\leq\lambda_1 \tag{6.1}$$

式中，ω_{lf}，ω_{rf}，ω_{lr}，ω_{rr}分别代表左前轮、右前轮、左后轮、右后轮的转动角速度（rad/s），λ_1为设定好的阀值。式（6.1）满足时认为轮速传感器工作正常；式（6.1）不满足时，再通过对同轴车轮转速差和同侧前后轮转速差的分析比较，就可以识别出发生故障的车轮轮速传感器，电控单元以故障代码的形式存储故障情况。

6.4.6 参考车速计算模块

参考车速计算模块用来计算车身速度。由于成本等方面的原因，一般车辆上是不安装车身速度传感器的，集成系统控制的目标就是将车轮的纵向滑动率（滑移率或滑转率）控制在合理的范围内，见式（6.2）和式（6.3）。因此参考车速确定的精确与否将直接影响到控制效果的优劣，各个厂家有各自的方法，且属于生产厂家的核心商业机密不对外公布。课题组对于参考车速进行了大量分析研究，采用了自适应卡尔曼滤波的参考车速确定方法，计算的参考车速结果存储在参考车速缓冲区，用于制动工况和和驱动工况滑动率的计算，以及ACC系统控制中主目标车辆绝对车速的计算。

$$S_{xi}=\frac{\omega_i\cdot r-v}{\omega_i\cdot r}\times 100\% \tag{6.2}$$

$$S_{Bi}=\frac{v-\omega_i\cdot r}{v}\times 100\% \tag{6.3}$$

式中，v代表计算的参考车速，km/h；ωi代表各车轮转动角速度，rad / s；S_{Ai}和S_{Bi}分别代表对应车轮的滑转率和滑移率；r代表车轮半径。当判断制动踏板已经踩下，进行滑移率的计算，否则进行滑转率的计算，同一车轮的滑移率和滑转率存储在相同地址的缓冲区中。

6.4.7 路面识别模块

路面识别方法是集成系统控制中的关键技术，它的准确识别对于参考车速的准确确定和合理的逻辑门限值的选取都有重要的意义。我们在大量试验数据的基础上通过深入的理论分析和控制逻辑设计，实现了较为实用的路面识别。

6.4.8 车辆运动状态识别模块

车辆运动状态识别模块用来区分直线行驶还是弯道行驶，左右后轮的速度差包含了转向信息。在车辆直线行驶工况下，左右后轮的轮速差应当不超过一定限度。计算两个后轮转速差Δω（rad/s），并设定判断参数λ2，λ2的确定考虑到使用过程中左、右后轮轮胎胎压变化和轮胎磨损程度的不同。当Δω>λ2时判断车辆为转向行驶，否则为直线行驶。

6.4.9 控制执行模块

系统控制执行模块包括二个子模块，控制逻辑模块和执行机构动作模块。通过整理后的角速度和滑移率作为输入量，依据ABS模糊控制逻辑模块的模糊规则做出逻辑判断和发出控制指令，控制执行机构动作模块，控制模块主要是通过电磁阀产生制动信号，迫使液压装置加压，达到制动效果。

其具体方法是由于S3C2410X将系统空间分为8组，每组的大小是128MB，共1GB。因此它的内存足够大，无需考虑内存之忧。在芯片中如何实现控制规则表是整个系统的关键，本系统采用二维矩阵形式存储控制器规则表，假设二维矩阵为control[15][15]。在S3C2410X只能存储数字量，故在内存中模糊语言用十六进制表示：输入量E,DE量化后分别为x，y，且x,y＝｛-6,-5,-4,-3,-2,-1,0,1,2,3,4,5,6｝，在ARM中则对应a,b=｛0,1,2,3,4,5,6,7,8,9,10,11,12｝。假设量化后x=-5，y=2,在ARM中对应a=1,b=8查表时只需查control[1][8]则这样表示无论从数据管理还是从查表方便而言，都具有极大的优势。这种离线计算、在线查表的模糊控制方法比较容易满足实时控制的要求。

这种离线计算采用 MATLAB 计算出控制表，然后以二维矩阵形式存储在 ROM 中，供在线查表。

6.4.10 数据采集中断服务程序

数据采集模块由定时中断服务程序完成。集成系统采用数据采集和主控制程序分离的结构型式。由于ABS电磁阀动作响应时间通常不小于7ms，再加上制动系统的机械滞后，传统的数据采集和控制周期通常不能低于电磁阀动作响应时间。为了使数据处理更为方便和准确，提高数据采样频率是一种有效的手段之一，因此，集成系统采用了3ms周期的定时中断数据采集程序。

为了满足集成系统的需要，目前采集的数据主要包括：四个车轮转速信号，蓄能器压力信号，节气门开度信号、加速踏板开度信号。考虑到数据采集模块属于实时模块，对实时性要求较高，所以在设计软件时，尽可能的利用硬件资源，减少MCU 的占用时间。

ARM9有5个16位定时器，每个定时器都可以通过软件选择作为输入捕捉或者输出比较，MCU自动将输入捕捉寄存器中的时间值放入保持缓冲器中，同时将主计时器的当前时间值放入输入捕捉寄存器中，因此基于测周法可较为容易地实现四个轮速信号的采集。

节气门开度信号和加速踏板开度信号、蓄能器压力信号都是模拟信号，它的采集由ARM9内部的AD转换模块自动完成。设置了模拟信号AD转换的完毕中断，在中断处理程序中存储转换结果。

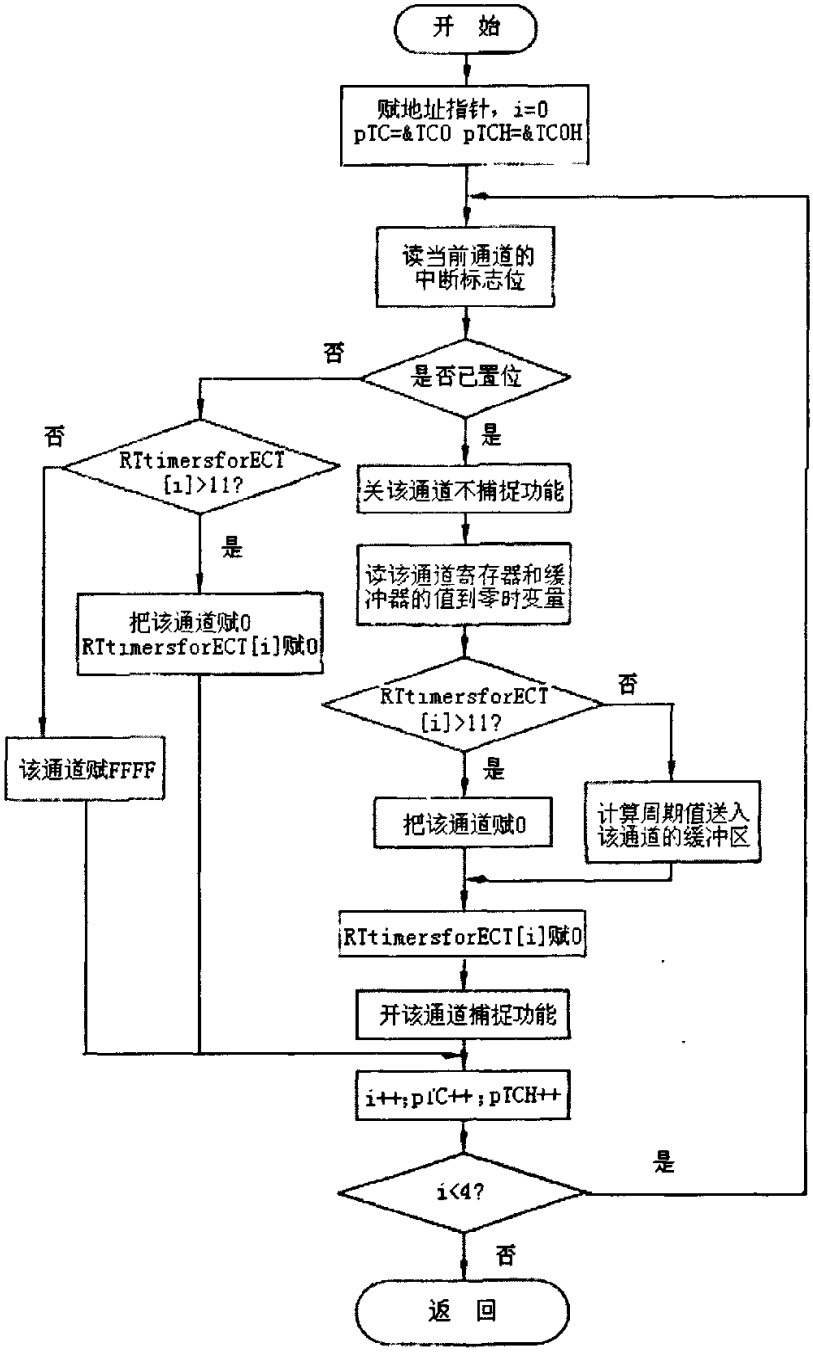

图6. 10 数据采集模块

Fig.6.10 The module for data collection

6.4.11 制动踏板中断服务程序

考虑到驾驶员制动动作与主动制动控制的紧急切换，设计了制动踏板中断服务程序。无论系统处于何种控制方式和控制状态，当采集到制动踏板上升沿触发的中断时，集成控制系统立即进入制动踏板中断服务程序，对执行机构模块进行复位，恢复常规制动方式，不干预驾驶员的制动动作，保证了车辆行驶的安全性。系统执行机构复位延时一段时间后退出中断，继续运行主控制模块循环控制。

6.4.12 通讯模块

集成系统通讯模块包括实时通讯和非实时通讯两部分。其中实时通讯主要指通过CAN总线与车距传感器间的通讯，由于车距尚未购买到合适的车距传感器，这部分仅预留了相关的硬件和软件接口；非实时通讯主要指通过串行通讯与外部故障诊断设备的通讯。

在上位机进行通信时，必须保证要有相同的波特率。串行初始化选用串口工作方式1UART，允许接收，串行口被设置为波特率可变的8位异步通信通信接口。波特率的定义为每秒钟传送的二进制数码的位数(亦称比特数)，单位是bit/s，波特率是串行通信的重要指标，用于表征数据传输的速度。波特率越高，数据传输速度越快，但和字符的实际传输速率不同。字符实际传输速率是指每秒内所传字符帧的帧数，和字符帧的格式有关。在串行方式1中，1帧信息为10位，如果采用波特率为57600bit/s的通信系统，在与上位机的一次通信中要传递三帧数据，则字符实际传输速率为57600/30=1920帧/秒。而ABS每秒动作10-20次，也就是说每秒发送数据大概是20帧左右，接收数据大概为40帧，所以57600bit/s是完全可以满足传送的需要。另外，为了让上位机程序中计算出来的轮速尽可能和硬件产生的模拟轮速相近，就应该在1秒内增加轮速发送的次数，以便及时的反映出轮速的变化，所以也要求采用比较高的波特率才能满足要求。波特率设计选用11.0592MHZ的振荡频率。

ARM9串口初始化设置程序如下：

```
#define HwBaseAddress 0x80000000
#define HwControl 0x00000100
#define HwUartControl 0x000004C0
#define HwStatus 0x00000140
```

```
#define HwUartData 0x00000480
// 硬件串口寄存器的定义
#define HwUartControlParityEnable 0x00002000
#define HwUartControlParityEven 0x00004000
#define HwUartControlTwoStopBits 0x00008000
#define HwUartControlFifoEnable 0x00010000
#define HwUartControlDataLength 0x00060000
#define HwUartControlDataLength5 0x00000000
#define HwUartControlDataLength6 0x00020000
#define HwUartControlDataLength7 0x00040000
#define HwUartControlDataLength8 0x00060000
#define HwStatusUartTxBusy 0x00000800
#define HwStatusUartRxFifoEmpty 0x00400000
#define HwStatusUartTxFifoFull 0x00800000
// UARTEnable 设置串口 UART,并使能
long UARTEnable(long lPort, long lDataRate, long lDataBits, long lStopBits,
        long lParity, long lEvenParity)
{
  unsigned char *pucPtr = (unsigned char *)HwBaseAddress;   //硬件的基地址
  long lRates[12] = { 115200, 76800, 57600, 38400, 28800, 19200, 14400, 9600,
            4800, 2400, 1200, 110 };                    //串口波特率
  long lDivisors[12] = { 1, 2, 3, 5, 7, 11, 15, 23, 47, 95, 191, 2094 };
  long lIdx, lConfig;
// 赋波特率值
  for(lIdx = 0; lIdx < 12; lIdx++)
  {
    if(lRates[lIdx] == 57600)
    {
      break;
    }
  }
  if(lIdx == 12)
  {
    return(0);
  }
```

```
    lConfig = lDivisors[lIdx];
    // 设置有效的数据位宽度
    switch(lDataBits)
    {
      case 5:
      {
        lConfig |= HwUartControlDataLength5;
        break;
      }
      case 6:
      {
        lConfig |= HwUartControlDataLength6;
        break;
      }
      case 7:
      {
        lConfig |= HwUartControlDataLength7;
        break;
      }
      case 8:
      {
        lConfig |= HwUartControlDataLength8;
        break;
      }
    default:
      {
        return(0);
      }
    }
  // 设置停止位个数
    if(lStopBits == 2)
    {
      lConfig |= HwUartControlTwoStopBits;
    }
    else if(lStopBits != 1)
    {
```

```
        return(0);
    }
// 设置奇偶位校验
    if(lParity)
    {
        lConfig |= HwUartControlParityEnable;
// 偶数位
        if(lEvenParity)
        {
// 改变奇数位为偶数位 (默认是奇数位).
            lConfig |= HwUartControlParityEven;
        }
    }
// 设置,使能 UART.
// 打开 UART.
    *((unsigned long *)(pucPtr + HwControl)) |= HwControlUartEnable;
// 设置 UART.
    *((unsigned long *)(pucPtr + HwUartControl)) =
    lConfig | HwUartControlFifoEnable;
    }
//*****************************************************************************
// UARTDisable 关闭 UART.
//*****************************************************************************
void UARTDisable(long lPort)
{
    unsigned char * volatile pucPtr = (unsigned char *)HwBaseAddress;
// 如果 UART 已经关闭,返回
    if(!lPort1Enabled)
    {
        return;
    }
// 检查传送数据的 FIFO 是否为空,若不为空,循环等待.
    while(*((unsigned long *)(pucPtr + HwStatus)) & HwStatusUartTxBusy)
    {
// 关闭 UART.
```

```
    *((unsigned long *)(pucPtr + HwControl)) &= ~HwControlUartEnable;
// 标记 UART 为关
    lPort1Enabled = 0;
    }
}
//*****************UARTSendChar 发送一个字符到串口 UART***********
void UARTSendChar(long lPort, char cChar)
{
    unsigned char * volatile pucPtr = (unsigned char *)HwBaseAddress;
// 循环等待直到传送数据的 UART FIFO 为空.
    while(*((unsigned long *)(pucPtr + HwStatus)) & HwStatusUartTxFifoFull)
    {
    }
// 写字符到串口 UART.
    pucPtr[HwUartData] = cChar;
}
// UARTReceiveChar 从串口 UART 接收字符
char UARTReceiveChar(long lPort)
{
    unsigned char * volatile pucPtr = (unsigned char *)HwBaseAddress;
// 循环等待直到接收数据的 UART FIFO 内有数据
    while(*((unsigned long *)(pucPtr + HwStatus)) & HwStatusUartRxFifoEmpty)
    {
    }
// 从串口 UART 读出数据,并返回
    return(pucPtr[HwUartData]);
    }
// UARTCharReady 判断是否在串口有数据等待接收
long UARTCharReady(long lPort)
{
    unsigned char * volatile pucPtr = (unsigned char *)HwBaseAddress;
// 检查是否有数据在 UART FIFO 等待
    if(*((unsigned long *)(pucPtr + HwStatus)) & HwStatusUartRxFifoEmpty)
    {
// 没有数据,返回 0
```

```
        return(0);
    }
// 有数据,返回 1
    return(1);
}
```

上位机的串口初始化通过 VC++中提供的 MSComm 控件可以实现串行端口传输和接收数据，为仿真界面提供串行通信功能。在对话框界面中添加 MSComm 控件，并将生成的 mscomm.h 添加到 abssimulinkDlg.h 中，在 abssimulinkDlg.h 文件中添加一个 CMSComm 类的变量 m_Com。然后进行串口初始化设置，在 abssimulinkDlg.cpp 文件的 CAbssimulinkDlg::OnInitDialogo PAI 数中添加如下的代码:

```
//串口初始化开始
DWORD style=WS_VISIBLE|WS_CHILD;
if (!m_Com.Create(NULL,style,CRect(0,0,0,0),this,IDC_COMMCTRL))
{    /*TRACEO*/
AfxMessageBox("Failed to create OLE CommunicationsControl\n");
return -1; //fail to create
}
m_Com.SetCommPort(2); //选择 COM2
m_Com.SetInBufferSize(1024);//设置输入缓冲区的大小，Bytes
m_Com.SetOutBufferSize(512);//设置输入缓冲区的大小，Bytes
if(!m_Com.GetPortOpen())      //打开串口
m_Com.SetPortOpen(TRUE);
m_Com.SetInputMode(1); //设置输入方式为二进制方式
m_Com.SetSettings("57000,n,8,1 "); //设置波特率等参数
m_Com.SetRThreshold(1);//为 1 表示有一个字符引发一个事件
m_Com.SetInputLen(0);
m_Com.GetInput();    //先预读缓冲区以清除残留数据
```

6.5 测试结果与分析

利用所开发的采集系统和本文所设计的 ABS 控制系统，利用 Matlab 进行了典型工况 ABS 模拟仿真，验证其控制效果，具体 Simulink 仿真模型设计见 2.4。

ABS 仿真试验是在制动初速度为 50km/h 的条件下进行，图 6.11 为带有 ABS 控制的直线制动过程。图中显示了车辆左前轮速度变化的过程，在整个制动过程中与车速都能够

比较好地逼近，车轮的滑移率也被控制在比较理想的区域内，保证了车辆制动过程中方向的稳定性。

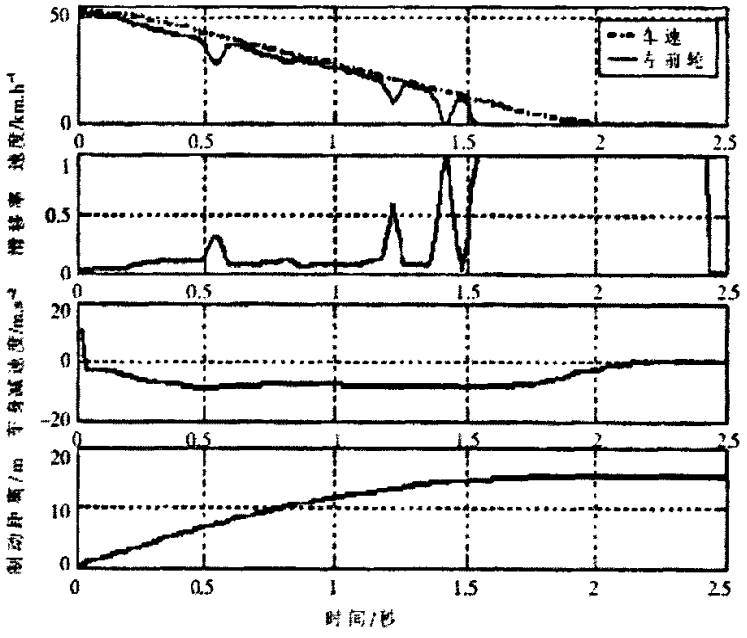

图 6.11 带有 ABS 控制的直线制动过程

Fig. 6.11 The beeline trig process of the control by ABS

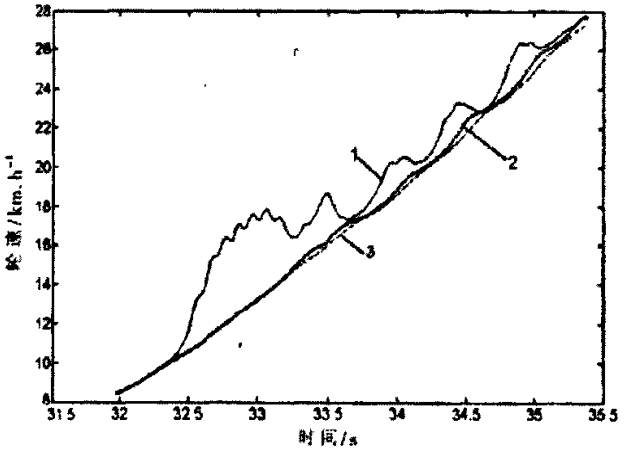

图 6.12 左右两侧驱动轮轮速比较

Fig. 6.12 The compare of the left and right driving wheel speed

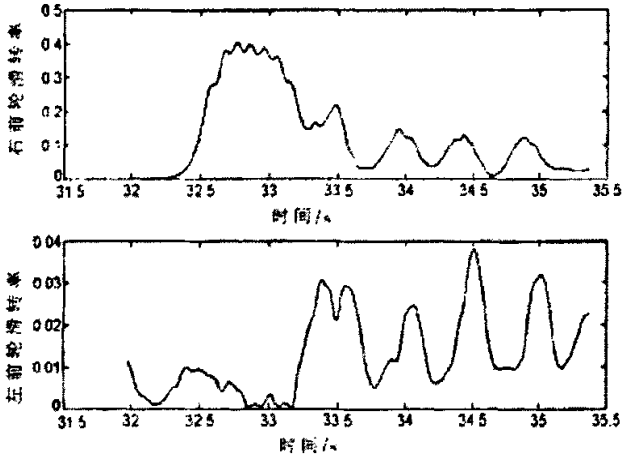

图 6.13 左右两侧驱动轮滑转率比较

Fig. 6.13 The compare of slip ratio of the left and right driving wheel

为了进一步完善验证 ABS 系统功能，我们还进行了 ASR 仿真试验，选择一挡对开路面起步过程，右侧车轮下是低附着系数的情况。图 6.12 为起步过程中左右两侧驱动轮转速对比，其中曲线 1 为右侧驱动轮车速，曲线 3 为左侧驱动轮车速，曲线 2 为非驱动轮车速，可近似认为是车身速度。可以看到起步初期，右侧驱动轮发生明显的滑转，在 33.5 秒附近 ASR 开始调节，车轮的滑转明显改善。图 6.13 为起步过程中左右两侧驱动轮滑转率的变化曲线，更明显地反映出进行 ASR 控制后，右侧滑转驱动轮的滑转率被控制在较为理想的范围内。同时，由于制动干预的影响，左侧驱动轮的滑转率略有上升。

通过实车试验，说明 ABS 系统中的各模块功能都取得了比较理想的控制效果，为其它底盘主动安全控制系统的集成创造了条件。

6.6 小结

本章论述了基于ARM9微处理器开发的燃料电池电动公交车ABS系统。进行了ABS系统的设计，着重描述了如何实现电子控制单元中硬件电路和软件逻辑的集成。所开发的集成化电控系统具有良好的实用性和扩展性，为电动公交车的主动安全控制装置集成化研究打下了基础。

7 结论

本文以新型的燃料电池电动公交车为应用平台，研究了国内外先进的ABS技术，设计了一个燃料电池电动公交车ABS控制系统。在系统的设计过程中，针对电动公交车的实际工况要求，主要完成了下面工作：

(1) 从控制器实际设计及实施的角度研究了3种比较实用的ABS控制算法，即传统的逻辑门限控制、PID控制和模糊控制。首先论述了3种控制方法的原理，然后采用国际流行的SIMULINK/MATLAB 软件对3种控制方法作了模拟研究，并对模拟结果作了分析与评价。

(2) 基于嵌入式技术，搭建了电动公交车ABS嵌入式系统的平台，为控制算法的实现提供了有力平台。剖析了嵌入式Linux操作系统在ARM平台的移植过程，以Linux2.6内核为例，对内核重新裁剪和编译，成功移植到S3C2410开发板上。对S3C2410开发板进行了CAN接口扩展，满足了燃料电池电动公交车ABS系统对开发环境的要求，为高效、可靠的嵌入式软件开发奠定了基础。

(3) 深入研究了电动公交车ABS系统的组成与结构，同时，根据电动公交车的实际工况要求，设计了各个控制单元的电路；运用功能概念，对汽车防抱死制动系统(ABS)的作用及ABS汽车的紧急制动过程进行了分析。针对汽车防抱死制动系统(ABS)的强非线性，采用基于滑移率的模糊控制策略建立了ABS的仿真模型。设计结果可供工程实践参考。

(4) 设计了基于ARM9微处理器开发的电动公交车ABS系统。进行了ABS控制系统的设计，描述了如何实现电子控制单元中硬件电路和软件逻辑的集成。所开发的集成化电控系统具有良好的实用性和扩展性，为电动公交车的主动安全控制装置集成化研究打下了基础。

电动公交车ABS系统的设计是一个非常复杂的过程，由于课题的设计周期比较短，本人经验不足，因此，系统在实时性、安全性方面仍需改进，进一步的工作如下：

(1) 进一步完善电动公交车ABS控制系统的结构设计，结合电动公交车具体的运行工况需求不断的改进。

(2) 通过不断的试验优化，设计完整的电动公交车 ABS 控制系统的测试方案，进而，在试验台和试验车上完成测试的具体过程。

(3) 根据试验的结果，改进电动公交车 ABS 控制系统的软硬件设计，再次试验，得出最优的模型，装车实际测试。

参 考 文 献

[1] 郭晓汾.燃料电池车辆.重型汽车,2005,(1):27~29.

[2] 任恒山.现代汽车概论.北京：人民交通出版社，2005.

[3] 余志杰.汽车理论.北京:机械工业出版社,1999, 5.

[4] 魏朗,王囤.现代汽车制动防抱死系统实用技术.北京:人民交通出版社，2001，6.

[5] 王田苗.嵌入式系统设计与实例开发一基于 ARM 微处理器与 NC/OS-II 实时操作系统(第 2 版).北京:清华大学出版社，2003.

[6] Zadeh L A. Fuzzy Sets, Informat Control, 1965，8:P338~353M Bauer, Masayoshi Tomizuka. Fuzzy Logic Traction Controllers and Their Effect on Longitudinal Vehicle Platoon Systems. California PATH Research Report, UCB-ITS-PRR-95-14.

[7] Peter M. de Koker, J. Gouws, L. Pretorius. Fuzzy Control Algorithm forAutomotive Traction Control Systems. IEEE 00550996.

[8] 汪立亮.实现代汽车自动防抱死制动系统的原理与检修.北京:电子工业出版社 2000. 1.

[9] Wayne Wolf.嵌入式计算系统设计原理.北京：机械工业出版社，2002.

[10] 桑楠.嵌入式系统原理及应用开发技术.北京：北京航空航天大学出版社，2002.

[11] 蔡德聪，周德泽工业.控制计算机实时操作系统.北京：清华大学出版社，2002.

[12] Jean J. Labrosse.嵌入式系统构件.北京：机械工业出版社，2002.

[13] Jean J. Labrosse.嵌入式实时操作系统 UC/OS.北京：中国电力出版社，2001.

[14] Bakker E，Nyborg L. Pacehka H.B. Tyre Modelling for Use in Vehicle Dynamics Studies, SAE paper, No.870481，1987。

[15] 余志生.汽车理论.北京：机械工业出版社，1999. 5.

[16] 程军.汽车防抱死制动系统的理论和实践.北京：北京理工大学出版社，1999.9.

[17] Fangjun Jiang, Zhiqiang Gao. An Application of Nonlinear PID Control to aClass of Truck ABS Problems.

[18] Luca Piancastelli, Giuseppe Sarubbi. An EESP Control System For The Recover OF The Initial Direction After An Initial spin. XII ADM InternationalConference — Grand Hotel -Rimini — Italy -Sept. 5th-7th,2001.

[19] Terry D. Day] L. Daniel Metz, The Simulation of Driver Inputs Using a Vehicle Driver Model. SAE TECHNICAL PAPER SERIES 2000-01 — 1313.

[20] Terry D. Day, Sydney G. Roberts, A Simulation Model for Vehicle Braking Systems Fitted with ABS. SAE TECHNICAL PAPER SERIES 2002-01-0559.

[21] Garrick Forkenbrock, Mark Flick, W. Riley Garrott, A Comprehensive Light Vehicle Antilock Brake System Test Track 'Performance Evaluation. SAE TECHNICAL PAPER SERIES 1999-01 — 1287.

[22] Constantin von Altrock, Fuzzy Logic in Automotive Engineering. Circuit Cellar INK Issue88 November 1997, P1~9.

[23] 韩曾晋. 自适应控制. 北京：清华大学出版社，2003, 2.

[24] 崔龙，周启明，江文瑞. “基于 ARM 的实时测控系统开发平台”，单片机与嵌入式系统应用，2003, 1: 47~50.

[25] 肖踞雄. “嵌入式系统硬件抽象层的建立及软件的可移植性设计”，单片机与嵌入式系统应用，2003, 1: 11~13

[26] John Chatzakis, Kostas Kalaitzakis, Nicholas C. Voulgaris, and Stefanos N. Manias. Designing a New Generalized Battery Management System. IEEE Transactions On Industrial Electronics, VOL. 50 NO. 5. OCTOBER 2003.

[27] Hydrogen Fuel Cell Engines and Related Technologies Course Manual.

[28] J. T. Pukrushpan, A. G. Stefanopoulou, H. Peng. Control of Natural Gas Catalytic Partial Oxidation for Hydrogen Generation in Fuel Cell Aplications. IEEE American Control Conference, 2003.

[29] 李士勇. 模糊控制•神经控制和智能控制论. 哈尔滨：哈尔滨工业大学出版社, 1998.

[30] 汪光阳. 基于模糊工具箱和 SIMULINK 的模糊控制系统计算机仿真. 安徽工业大学学报 , 2001, 18(1):57-61.

[31] V. JIRICNY, S. SIU, A. ROY and J.W. EVANS. Regeneration of zinc particles for zinc-air fuel cells in a spouted-bed electrode. Journal of Applied Electrochemistry , 2000: 647-656.

[32] Fritz G. Will. Recent advances in zinc/air batteries. Proceedings of the 13th Annual Battery Conference on Applications and Advances, 1998.

[33] Jonathan Goldstein , Ian Brown, Binyamin Koretz. New developments in the Electric Fuel Ltd. Zinc/air system. Journal of Power Sources, 1999: 171 - 179.

[34] J. T. Pukrushpan, A. G. Stefanopoulou, H. Peng. Controlling Fuel Cell Breathing: Initial Results on the Oxygen Starvation Problem. IEEE Control Systems Magazine, April 2004.

[35] Wookey. Porting LINUX kernel to new hardware platform. http://www.intel.com/. 2006.

[36] 尤盈盈，孟利民. 构建嵌入式 linux 交叉编译环境 .计算机与数字工程 ，2006 年 06 期:30-32.

[37] 陈渝，李明，杨晔. 源码开放的嵌入式系统软件分析与实践——基于 SkyEye 和 ARM 开发平台. 北京：北京航空航天大学出版社，2005.

[38] 刘军芳，李众立，胡和智. 基于 s3c2410 开发板的 Boot Loader 的启动分析. 微计算机信息，2006，22(6):201-203.

[39] 魏平，夏良正，王岩. Linux 体系结构及嵌入式 Linux 的移植方法. 东南大学学报(自然科学版)，2004，34(S1):126-130.

[40] Daniel P. Bovet Marco Cesati. Understanding the Linux Kernel(2nd Edition). O'Reilly Media, Inc. 2002.

[41] 陶志东，周纯杰，宋明权等. 基于 S3C2410 的 CAN 总线通信设计与开发. PLC&FA，2006，(1).

[42] 杨红科. 基于 CAN 总线的液位模糊控制系统的设计及研究:(硕士学位论文). 南京:南京航空航天大学，2005.

[43] Ronald G. Landman. Design and Analysis of CANNetworks for Vehicles, SAE Technical Paper Series, 2000-01-2585.

[44] 孙天泽，袁文菊，张海峰. 嵌入式设计及 Linux 驱动开发指南——基于 ARM9 处理器. 北京：电子工业出版社，2005.

[45] 陈铁军，仇洪冰. 基于 S3C2410 的嵌入式 Linux 的移植方法. 桂林电子工业学院学报，2006，26(4):259-263.

在学研究成果

在学期间发表的论文：

[1] 张扬，申朝，曾碚凯，姜岩. CAN 总线技术在电动公交车上的实现. 沈阳工业大学学报,2006(10):563-567.

致　谢

首先，我要衷心感谢我的导师曾碚凯副教授。在研究生学习期间，曾老师给我在学习和科研上提供耐心细致的指导。特别在论文写作过程中，更得到曾老师的悉心指导。曾老师严谨的治学态度，勤奋的工作作风和对新领域的探索精神，是我一生学习的榜样。

同时感谢电气工程学院的梁中华教授，给我的论文提出了许多宝贵的意见和建议。还要感谢在研究生期间曾给予我帮助和支持的其他各位老师和同学。感谢沈阳工业大学研究生部及信息学院的各位老师和领导对我的帮助与支持。

最后，我还要感谢我的家人，他们默默的支持、理解和无微不至的关怀与鼓励，使我能够安心地并完成学业。