中文摘要

摘要：电池管理系统是混合动力汽车中的关键部件，是电池和整车的联接纽带。在对电池状态进行监控的同时，还需估算电池的剩余电量，调整电池的充放电功率限制，为整车能量的流动提供依据。为使动力电池合理利用，需要对系统参数进行大量标定试验。

本文借鉴了汽车发动机电控系统标定方法，提出了一种基于CCP协议的电池管理系统的标定方案。该方案主要由电池管理系统、USB－CAN通信模块及主控PC机的标定平台软件组成，并采用国际通用的CCP作为电池管理系统和PC机的标定通信协议，由于CCP基于地址通信，可以提高了系统的扩展性和通用性。

论文简要介绍了CCP协议及标定系统组成，完成了针对长安混合动力汽车的电池管理系统硬件系统设计。在此基础之上，着重讨论了CCP驱动程序与接口程序的开发方法，以及实时操作系统μC/OS-II在电池管理系统软件设计中的开发方法。电池管理系统所采用的微控制器是Freescale 16位芯片MC9S12DT128，软件开发是在集成开发环境CodeWarrior下完成。采用UML建模工具对标定平台软件的功能需求和动态行为进行了建模，并在Visual C++ 6.0下完成了软件的开发。

最后对电池管理系统进行初步标定试验，结果表明将CCP协议引入电池管理系统可以方便对系统进行标定，实现对电池的合理利用。

关键词：混合动力汽车；电池管理系统；CCP 协议；标定；μC/OS-II

分类号：U469.72

ABSTRACT

ABSTRACT: Battery management system which is the linking ligament of battery and HEV is the key part for HEV. It monitors the state of battery, estimates the state of charge and adjusts the battery charge-discharge power limit. All these are the foundation of the car's energy flow.

The calibration program based on CCP protocol is developed which is mainly composed of the battery management, USB-CAN communication convert module and the calibration platform software. The CCP protocol can improve the system's scalability and versatility.

The paper outlines the CCP protocol and the composition of calibration system. The battery system's hardware for ChangAn HEV is designed. On this basis, the paper introduces how to design CCP driver and software development for battery management using μC/OS-II. The MCU MC9S12DT128 is the core controller of the battery management system, while software design is competed in Integrated Development Environment CodeWarrior. UML modeling tool is used to model for calibration software platform functional requirements and dynamic behavior, and Visual C++ 6.0 is used to complete software.

In order to verify the calibration system function, the calibration tests have been done. The result indicates that the CCP protocol can make the calibration of the system facilely and help the battery is used more reasonable.

KEYWORDS: HEV; Battery Management System; CCP; calibration; μC/OS-II

CLASSNO: U469.72

独创性声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作和取得的研究成果，除了文中特别加以标注和致谢之处外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得北京交通大学或其他教育机构的学位或证书而使用过的材料。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作了明确的说明并表示了谢意。

学位论文作者签名：盛大双 签字日期：2009 年 6 月 25 日

学位论文版权使用授权书

本学位论文作者完全了解北京交通大学有关保留、使用学位论文的规定。特授权北京交通大学可以将学位论文的全部或部分内容编入有关数据库进行检索，并采用影印、缩印或扫描等复制手段保存、汇编以供查阅和借阅。同意学校向国家有关部门或机构送交论文的复印件和磁盘。

（保密的学位论文在解密后适用本授权说明）

学位论文作者签名：盛大双　　导师签名：

签字日期：2009年6月15日　　签字日期：2009年7月2日

致谢

本论文的工作是在我的导师姜久春教授的悉心指导下完成的，姜久春教授严谨的治学态度和科学的工作方法给了我极大的帮助和影响。在此衷心感谢两年来姜久春老师对我的关心和指导。

姜久春教授悉心指导我们完成了实验室的科研工作，在学习上和生活上都给予了我很大的关心和帮助，在此向姜久春老师表示衷心的谢意。

张维戈教授和王占国老师对于我的科研工作和论文都提出了许多的宝贵意见，在此表示衷心的感谢。

在实验室工作及撰写论文期间，实验室里的师兄弟师姐妹对我论文的研究工作给予了热情帮助，在此向他们表达我的感激之情。

另外也感谢我的家人，他们的理解和支持使我能够在学校专心完成我的学业。

1 绪论

1.1 引言

交通能源与环境保护是 21 世纪全球面临的重大挑战,人们在不断地寻求清洁的代用燃料，以改善日益恶化的交通排放现状，电动汽车逐渐成为各国政府和汽车制造商关注的焦点，人们不遗余力地开展电动汽车的研究，试图使其成为新世纪汽车的发展主流。当代融合多种高新技术而兴起的纯蓄电池电动汽车、混合动力电动汽车及燃料电池汽车正在引发世界汽车工业的一场革命，展现了汽车工业的光明未来[1]。

1.2 HEV 和 BMS 在国内外的发展状况

国际电工委员会(IEC) 电动汽车技术委员会将混合动力汽车（Hybrid Electric Vehicle, HEV）定义为：有一种以上能量转换器提供驱动动力的混合型电动汽车,也可简单定义为将电力驱动和辅助动力单元(Auxiliary Power Unit, APU) 合用到一辆车上。混合动力汽车采用适当的燃料转换装置(如内燃机)、储能装置和电动机作为混合动力源，在严密的控制策略控制下，使燃料转换装置、储能装置和电机在驱动工况下尽可能工作在高效率、低排放区域，在汽车制动工况下，通过发电机或电机工作象限的调整回收部分制动能量，从而大大改善汽车在不同工况行驶时的燃油经济性能、尾气排放性能及其他使用性能。混合动力汽车结合了传统内燃机汽车和纯电动汽车的优点，续航里程不受限制，而且对于传统汽车的改动并不大，产业化生产的投入相比燃料电池汽车也少得多[2]。

20 世纪 90 年代以来，各大汽车公司通过使用先进的开发手段，已在较短的开发周期内相继推出了采用不同布置形式和控制策略的 HEV 产品。1991 年德国大众汽车公司第一次推出了“Chico”混合动力电动微型汽车。自 1995 年起，世界各大汽车生产厂商已将研究的重点转向了混合动力汽车的研究开发。日本丰田汽车公司是走在混合动力汽车研究发展前沿的汽车公司，开发的混合动力汽车已达到实用化水平。1997 年，丰田推出了世界上第一款批量生产的混合动力汽车 Prius，其后又在 2001 年相继推出了混合动力的面包车和皇冠轿车，2003 年推出第二代 Prius，在日本、美国和欧洲上市。截至 2006 年 4 月，丰田 Prius 混合动力车全球销售 504700 辆，成为到目前为止市场表现最好的混合动力汽车。日本的本田汽车公司于 1999 年独立研制开发的 Insight 混合动力车也已经实现量产。继 Insight 后，

2002 年本田公司又推出 Dualnote 运动型概念车。

国内 90 年代末期开始了混合动力汽车的研发工作。1999 年，清华大学与厦门金龙联合汽车工业有限公司合作研制成功国内第一辆混合动力轻型客车。2001 年底，国家“863”电动汽车科技攻关项目正式启动，第一批项目中主要是混合动力车，目前仍在进行当中。一汽和东风汽车集团联合各高校和研究所，在各自客车底盘上，研发混合驱动的公共汽车和大型客车。此外，东风电动汽车公司还承担了混合动力轿车的研究开发。目前东风电动车辆股份有限公司开发出混合动力汽车。其中，EQ7200HEV 型混合动力轿车以风神蓝鸟轿车为平台，以满足未来城市公务、出租用车需求为目标；EQ6110V 型混合动力城市公交车采用混联方案，专为 2008 年北京奥运会公交用车而开发。北京嘉捷博大电动车有限公司是国内最早进行混合动力技术和相关产品的研究和开发实践的公司，2001 年以来自主开发了串联式电动混合动力技术并且成功研制出中国第一辆混合动力电动大客车和世界第一辆混合动力电动机场摆渡车。天津清源电动车辆有限公司开发出混合动力中型客车，排放达到欧III标准，燃油经济性提高 15%以上，适于城市和城际之间的公共交通。深圳明华环保汽车有限公司开发出混合动力电动轻型客车，采用并联式混合动力系统，内燃机采用达到欧 II 排放标准的柴油机。第一汽车集团公司、美国电动车(亚洲)公司、汕头国家电动汽车试验示范区三方共同合作推出一款混合动力轿车——CA7180AE。我国通过国家“八五”、“九五”以及“十五”电动汽车的科技攻关，在 HEV 方面已经积累了一定的技术基础和经验，正向商品化阶段迈进[3]。在“十一五”中，部分汽车公司的混合动力大巴和轿车已经成功在北京奥运会交付使用。

2009 年 2 月，财政部、科技部初步选择了北京、上海、重庆等 13 个城市，率先在公交、出租、公务、市政和邮政等公共交通领域试点推广使用以混合动力为主的新能源汽车，在一些有条件的大城市，也适当推广应用部分纯电动和燃料电池汽车，这将进一步促进电动汽车的发展。

电动汽车电池管理系统(Battery Management System，简称 BMS)是电动汽车中一个越来越重要的部分，是一个处于监控运行及保护电池关键技术中的核心地位，能给出电池剩余电量和功率强度预测、进行智能充电和电池安全诊断等功能集合的综合系统。从有关资料来看，美国一直站在世界汽车技术领域的最前列，在电动汽车的电池管理系统的研究方面也走在世界各国的前列。通用汽车公司的 BMS 采用了一个微电脑，对电池组进行管理。监测和控制蓄电池组的充放电工作状态，提高电池的充放电性能，预测蓄电池组的荷电状态和剩余能量。美国也在开展基于智能电池模块(Smart battery module, 简称 SBM)的电池管理系统的研究。即在一个电池模块中装入一个微控制器并集成相关外围电路，然后封装为一个整

体，多个智能电池模块再与一个主控制模块相连，加以其它辅助设备，就构成了一个基于智能电池的管理系统。该 BMS 系统成功实现了对每个电池模块的状态监测、模块内电池电量均衡和电池保护等功能。

在欧洲，法国是电动汽车应用发展较快的国家。法国电动汽车计划(EDF)设计了一个随车电池管理系统来管理其电动汽车上的密封铅酸电池组，其主要功能有：电池寿命的记录、充电监测、行驶过程中电池组的管理、剩余电量显示，防止对电池的有害使用，收集电池信息从而确定如何合理使用电池和更换电池。在德国，西门子公司在其开发的电动汽车上安装了一个电池管理系统，电动汽车充电时，电池管理系统能跟踪电池充电特性，控制充电器对电池进行优化充电，电池管理系统对电池的工作状态进行监测，检测电池组的电量消耗，并将有关信息传送到仪表板上的仪表和信号指示装置上。

在日本，本田公司在电动汽车上安装的电池管理系统包括：管理控制模块、车载充电器、惯性控制开关、高压系统安全检测装置等。该系统对电池的状态进行监控，并根据电池的状态控制车载充电器的充电过程，当动力电池组高压端与车体有接触时，管理系统发出报警信号，当电动汽车发生碰撞时，管理系统切断电源，以保证安全。

我国在十五和十一五期间设立了电动汽车重大专项，经过几年的发展之后，在电池管理系统方面取得了很大的突破，与国外的水平也较为接近。各高校和动力电池公司等单位承担了混合动力汽车中电池系统研究的相关课题，并取得了良好的效果，在一汽，东风，上汽等多个汽车厂家均进行了装车，部分实现了量产。北京交通大学研究了包括国家电动汽车运行试验示范区、东风汽车集团、北京公共交通控股有限公司、北京 121 示范线、北京奥运用电动大巴等电动车电池管理系统，具有性能稳定、可靠性好、检测精度高、功能齐备、方便耐用等优点。经过大量的研究发展，电池管理系统已经从监控系统逐渐向管理系统转变，在功能、可靠性、实用性、安全性等方面都有了很大提高，检测精度提高，优化了整车通讯，数据处理方面，增加了电池故障的实时分析能力，对电池的滥用进行预警和报警，对故障进行定位，为电池的维护提供便利，增加了电池的均衡控制能力，但在电池的建模，SOC 和 SOH 的计算和评价方面，充放电算法等方面还需要进一步研究[4][5]。

1.3 基于 CCP 协议标定的优势

混合动力整车控制系统是一个很复杂的控制系统，为了使混合动力系统协调工作，并达到最佳的综合性能，必须对整车控制器的控制参数进行相应的修改和

优化，使混合动力系统按照最优的控制参数运行，这个过程称为标定。

传统的汽车标定只是涉及到发动机电子控制系统（ECU），这是因为传统的汽车最主要的核心部件就是发动机，发动机的控制性能优劣直接决定了汽车性能。ECU 是计算机技术在发动机控制技术的典型应用，采用微型计算机数字控制的燃油喷射系统和电子点火系统，使发动机的控制变得快速和准确，从根本上解决了发动机控制的难题。现在，发动机电控系统已成为现代发动机的标准配置。发动机电控系统的控制效果，很大程度上取决于其内部控制参数的优劣。为使发动机达到最佳性能，必须对这些控制参数进行精确的标定，获取其综合最优值，使发动机按照最优控制参数运行。

混合动力汽车在传统汽车基础上，还需要两大重要部件：电机和辅助储能单元，这就增加了整车控制器需要控制的参数。而作为储能单元的电池，它需要自己的电池管理系统以优化电池使用和延长电池寿命。电池管理系统为了实现优化电池使用和延长电池寿命的目标，需要对其内部的一些参数针对具体使用的电池进行优化，即需要进行系统标定。

CCP 协议首先在欧洲成为标准，不仅 ASAP 的很多成员厂商均有支持它的 ECU 产品，Vector 公司的 CANape, dSPACE 公司的 CalDesk, ETAS 公司的 INCA CCP 以及 ATI 公司的 VISION CCP 及等标定和测试工具都支持 CCP 协议，德国 BOSCH 公司新一代发动机管理系统的标定也将全面采用 CCP 协议。CCP 协议的不断完善和发展，显示出了强大优势，使得标定通讯标准的统一成为可能[6]。

1.4 研究内容

本课题的研究目的是针对电池管理系统内部控制参数优化的实际应用需要，主要研究内容包括：

(1)、在广泛查阅文献资料的基础上，提出电池管理标定系统的整体结构设计方案。

(2)、设计适用于混合动力车辆的车载电池管理系统的硬件电路的设计，确保各功能模块的准确测量，保证测量精度，具有较高的可靠性和稳定性，有较强的抗干扰性能。

(3)、实现 CCP 标定驱动程序的开发和实时操作系统 μC/OS-II 在电池管理系统的移植，完成基于它们的应用程序开发。开发适合电池管理系统标定用的上位机软件。

(4)、对动力电池的功率限制进行初步标定试验。

1.5 本章小结

本章首先介绍了课题研究背景，结合对电池管理系统的国内外研究现状进行了综述，介绍了课题的研究目的和研究工作的内容。

2 CCP 协议概述

CCP 协议是标定平台软件和标定型 ECU 连接所需的软件接口协议，是在 CAN 数据帧的数据域中扩展的应用层协议，定义了数据域内数据的具体含义，因此完全遵循 CAN2.0B 标准。CCP 由欧洲的自动测量系统标准化协会(ASAM)提出并制定，属于 ASAP 标准体系的一部分，接口的标准化使得标定系统更为通用，功能也更加强大。本课题采用的 CCP 协议版本为 1999 年发布的 2.1 版。

2.1 CAN 总线协议

2.1.1 基本概念

CAN (Controller Area Network)即控制器局域网络，属于工业现场总线的范畴。与一般的通信总线相比，CAN 总线的数据通信具有突出的可靠性、实时性和灵活性。由于其良好的性能及独特的设计，CAN 总线越来越受到人们的重视。它在汽车领域上的应用是最为广泛的，世界上一些著名的汽车制造厂商，如 BENZ(奔驰)、BMW(宝马)、PORSCHE(保时捷)、ROLLS-ROYCE(劳斯莱斯)和 JAGUAR(美洲豹)等都采用了 CAN 总线来实现汽车内部控制系统与各检测和执行机构间的数据通信。

CAN 总线是一种串行数据通信协议，通信介质可以是双绞线、同轴电缆或光导纤维，通信速率可达 1Mbps。CAN 总线通信接口中集成了 CAN 协议的物理层和数据链路层功能，可完成对通信数据的成帧处理，包括位填充、数据块编码、循环冗余检验、优先级判别等项工作。CAN 协议的一个最大特点是废除了传统的站地址编码，而代之以对通信数据块进行编码。采用这种方法的优点可使网络内的节点个数在理论上不受限制，数据块的标识码可由 11 位或 29 位二进制数组成，因此可以定义 2^{11} 或 2^{29} 个不同的数据块，这种按数据块编码的方式，可以非常灵活地控制节点的通信对象和通信来源，比如使不同的节点同时接受到相同的数据，这一点在分布式控制系统中非常有用。数据段长度最多为 8 个字节，可满足通常工业领域中控制命令、工作状态及测试数据的一般要求。同时，8 个字节不会过长占用总线时间，从而保证了通信的实时性。CAN 协议采用 CRC 检验并可提供相应的错误处理功能，保证了数据通信的可靠性[7][8]。CAN 具有下列主要特性：

(1)、通信方式灵活：网络上任意一个节点均可以在任意时刻主动地向网络上的其它节点发送信息，而不分主从；

(2)、多优先级：通过报文标识符，可以为 CAN 节点赋予不同的优先级，满足系统的实时要求，数据最快可在 134 微秒内得到传输；

(3)、CAN 采用非破坏性总线仲裁技术：当两个节点同时向网络上发送数据时，优先级低的节点主动停止数据发送，而优先级高的节点可不受影响地继续传输数据，大大地节约了总线仲裁冲突时间，在网络负载很重的情况下也不会出现网络瘫痪；

(4)、CAN 系统内任意两个节点之间的传输距离与其位速率有关，其最大速率可达到 1Mbps，最远距离可达 10km。CAN 的总线数值为两种互补的逻辑值："隐性"（"DOMAINNANT"）和"显性"（"RECESSIVE"）。在"隐性"状态下 CANH 和 CANL 都被固定在平均电压电平，VDIFF 近似为 0，而"显性"状态以大于最小阈值的差分电压表示。但需要注意的是当"隐性位"和"显性位"同时发送时，最后的值将为"显性"，这也是故障界定和错误检测的根本基础；

(5)、CAN 网络可以无破坏性地给予优先权的仲裁，各个节点之间依据优先权进行总线访问。在保证灵活性的同时，错误检测和出错信令方面都有独特的要求，使全系统的数据相兼容，即使在发送期间丢失仲裁或因遭遇破坏，数据帧都可自动重新发送，甚至在严重出错的情况下，系统可以很容易的区分暂时错误和永久性故障节点以及故障节点的自动脱离；

(6)、CAN 可以点对点、点对多点(成组)及全局广播方式传送接收数据；

(7)、CAN 总线通信格式采用短帧格式，传输时间短，受干扰概率较低，具有极好的检错效果，并且每帧信息都有 CRC 校验及其他检错措施，保证了数据通信的可靠性。

2.1.2 CAN 网络分层结构

CAN 按照 ISO/OSI 标准模型，CAN 结构分为数据链路层和物理层，其中数据链路层又分为：逻辑链路控制子层（LLC，Logical Link Control）和媒体访问控制子层（MAC，Medium Access Control）；物理层又分为：物理信令（PLS，Physical Signaling）、物理媒体附属装置（PMA，Physical Medium Attachment）和媒体相关接口（MDI，Medium Dependent Interface）。

LLC 子层的主要功能是报文滤波、超载通知和恢复管理，它为数据传送和远程数据请求提供服务，确认由 LLC 子层接收的报文实际已被接收，并为恢复管理和通知超载提供信息。在定义目标处理时，存在许多灵活性。MAC 子层是 CAN 协议的核心，MAC 子层的功能主要是数据的传送规则，亦即控制帧结构，执行仲裁、错误检测、出错标定和故障界定，它描述由 LLC 子层接收到的报文和对 LLC

子层发送的认可报文。在开始一次新的发送时，MAC 子层需要确定总线是否开放或者是否马上开始接收。MAC 子层出称为故障界定的一个管理实体监控，它具有识别永久故障或短暂扰动的自检机制。位定时特性也是 MAC 子层的一部分，MAC 子层不存在修改的灵活性。物理层的功能是有关全部电气特性在不同节点间的实际传送，在一个网络内，物理层的所有节点必须是相同的，然而，物理层的选择存在很大的灵活性。CAN 技术规范 2.0B 定义了数据链路中的 MAC 子层和 LLC 子层的一部分，并描述了与 CAN 有关的外层。物理层定义信号怎样进行发送，并涉及到位定时、位编码和同步的描述。在这部分技术规范中，未定义物理层中的驱动器/接收器特性，以便允许根据具体应用，对发送媒体和信号电平进行优化[9]。

2.1.3 CAN 协议报文

2.1.3.1 CAN 总线位数值的表示

总线位数值表示如图 2-1 所示，根据 ISO11898 以及 SAE J2284 定义的标准，CAN 总线的传输介质由两根传输线组成，其中一根为高电平传输线 CAN_H，其对地电压为 VCAN_H，另一根为低电平传输线 CAN_L，其对地电压为 VCAN_L，它们之间的差值被称为差分电压 Vdiff，即 Vdiff=VCAN_H−VCAN_L。满足条件 0.9V＜Vdiff＜5.0V 时，代表逻辑数字“0”，当前传送的数据位被称为“显性（Dominate）”位，当−1.0V＜Vdiff＜0.5V 时，代表逻辑数字“1”，当前传送的数据位被称为“隐性（Recessive）”位。当“显性”位和“隐性”位同时发送时，最后总线数值为“显性”[9]。

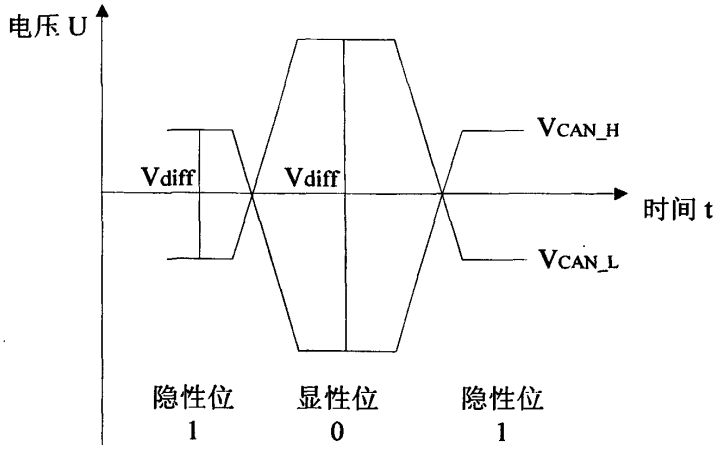


图 2-1 CAN 总线位数值表示

Figure 2-1 Representation of bit of CAN

2.1.3.2 CAN 协议报文帧格式

CAN 协议以报文的格式传递消息，每个报文都携带有标识符 ID，它标识了该报文的优先权。CAN 总线各个节点不分主从，都可以主动地发送消息。如果两个或多个节点同时向总线发送报文，则CAN 总线采用无破坏性的仲裁技术进行仲裁，优先权高的报文获得总线的使用权，而优先权低的报文会主动地停止发送，并重新检测总线状态，在总线再次空闲后，自动地重发报文。CAN 系统中，节点不使用有关系统的任何信息（如站地址等）。报文中的标识符 ID 不指出报文的目的地址，而是表述数据的类型和含义，通常这些数据的类型根据它们在控制中的重要性和实时性要求来划分。

在 CAN 系统中，每一个报文被称作一帧，数据在节点之间发送和接收以四种不同类型的帧出现和控制，其中：

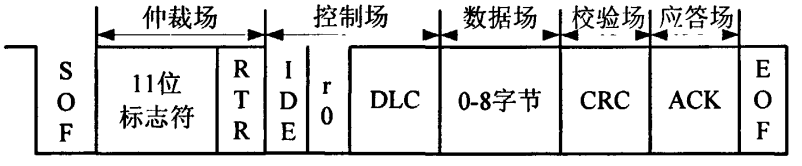
(1)、数据帧：数据帧携带数据，将数据从发送器传至接收器；

(2)、远程帧：由节点发送，用于请求发送具有相同标识符的数据帧；

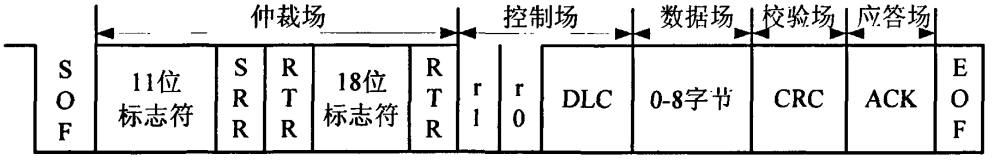
(3)、出错帧：可由任何节点发送，用于检测总线错误；

(4)、超载帧：用于提供先前和后续的数据帧或远程帧之间的附加延时。

其中数据帧和远程帧以帧间空间与先前帧隔开，其中最常用的是数据帧，针对数据帧，CAN 总线协议规定了标准信息帧和扩展信息帧，ISO-11898 中分别在定义了这两种模式，帧格式分别如图 2-2 所示：



(a) 标准数据帧格式



(b) 扩展数据帧格式

图 2-2 CAN 总线帧格式

Figure 2-2 Structure of standard frame and extended frame of CAN

图中(a)为标准帧格式，(b)为扩展帧格式，两种模式的主要区别在于扩展帧的标示符是 29 位，扩展帧格式可以标识更多的信息，这种格式下的 CAN 总线上可以存在的不同标示符信号个数几乎不受限制。图中帧格式的各个位场定义如下：

(1)、SOF：帧起始，标示帧的开始，由一个显性位构成，用于硬同步。

(2)、仲裁场：标示帧的优先级，标准帧的仲裁场由 11 位的标识符和 RTR(远程请求位)位组成；扩展帧的仲裁场由 29 位的标识符、SRR(替代远程请求位)、

RTR 位和 IDE 位组成。

(3)、控制场：由 6 位组成，DLC 是数据长度码共 4 位，表示数据场的长度，使用数目为 0~8。

(4)、数据场：传输的数据，最多 8 个字节。

(5)、校验场：包含 CRC(循环冗余校验码)序列，后随 CRC 界定符。

(6)、应答场：由 2 位组成，包括应答间隙和应答界定符，接受方成功接收数据帧后会发出应答信号。

(7)、EOF：帧结束，由 7 个隐性位组成的标志序列界定。

2.1.4 CAN 总线错误处理及故障界定

CAN 总线协议中也定义了错误类型和界定方式，主要有 5 中错误类型：

(1)、位错误(Bit Error)：某个单元向总线上送出一位同时也在监视总线，当监视到总线位值与送出的位值不同时，则在该位时刻检测到一个位错误。但是在仲裁场的填充位流期间或者应答期间送出隐性位而检测到显性位时，不认为是位错误。当发送器发出一个“认可错误”标志但检测到显性位时，也不认为是位错误。

(2)、填充错误(Stuff Error)：在应当使用位填充方法进行编码的报文中，出现了第 6 个连续相同的位电平时，将检测到一个位填充错误。

(3)、CRC 错误(CRC Error)：CRC 序列包括了发送器计算的 CRC 结果。接收器计算 CRC 的方法与发送器相同。如果计算结果和接收到的 CRC 序列结果不符，则检测到一个 CRC 错误。

(4)、格式错误(Form Error)：如果一个固定格式的位域含有一个或多个非法位，则检测到一个格式错误。对于接收器，帧末尾最后一位期间的显性位不被当作帧错误。

(5)、应答错误(Acknowledgment Error)：只要在应答间隙期间所监视的位不为显性，发送器就会检测到一个应答错误。

检测到错误条件的站通过发送“错误标志”(Error Flag)来表示错误。当任何站检出位错误、填充错误、形式错误或应答错误时，由该站在下一位开始发送出错标志当检测到 CRC 错误时，出错标志在应答界定符后面那一位开始发送，除非其他出错条件的错误标志已经开始发送。

在 CAN 总线网络中，任何一个节点单元可能处于下列 3 种故障状态之一，即错误激活(Error Active)、错误认可(Error Passive)和总线关闭。“错误激活”单元可以正常参与总线通信，并在检测到错误时发出“激活错误”标志；“错误认可”

单元不允许发送“错误激活”标志，它参与总线通信，在检测到错误时只发出“认可错误”标志，发送以后，“错误认可”单元将在起动下一个发送之前处于等待状态；“总线关闭”单元不允许对总线有任何影响(比如关闭输出驱动器)。

2.2 CCP 协议

2.2.1 ASAM MCD 标准模型

为了对 ECU 和测量、标定、诊断等工具间的接口实现标准化，ASAM 组织提出了 MCD 模型的概念，并按 MCD 的层次结构制定了一个完整的 ASAP 标准体系，如图 2-3 所示。

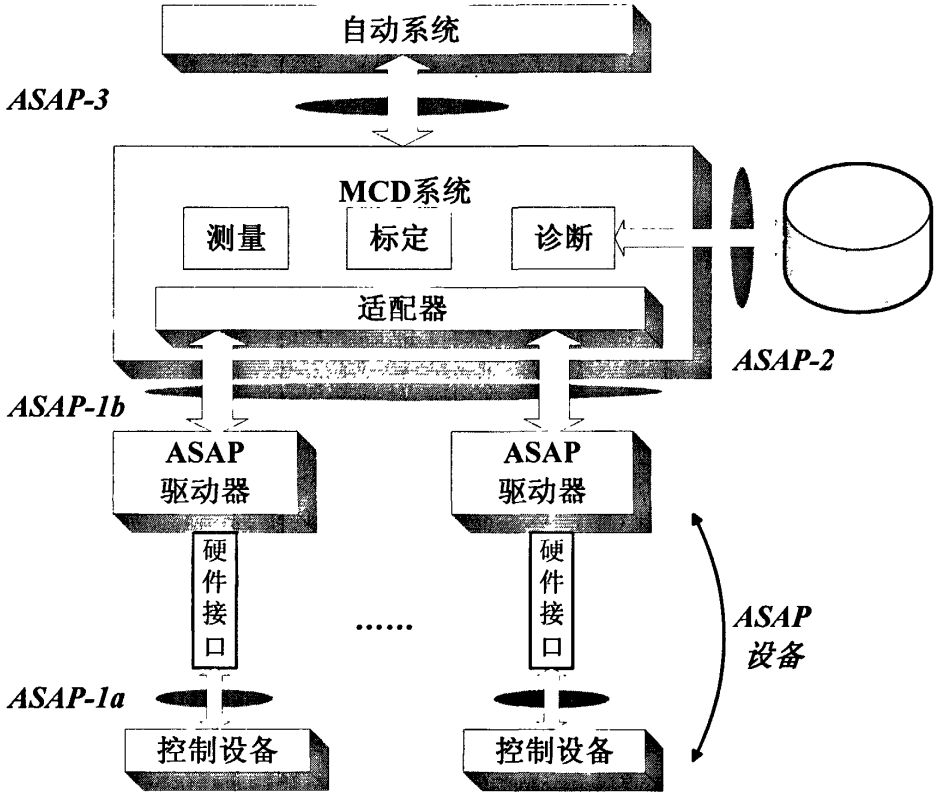


图 2-3 ASAP 标准模型体系结构
Figure 2-3 Structure of ASAP

围绕这个模型，在包括软件、硬件和信息交换兼容性需求的基础上，定义了 ASAP1，ASAP2，ASAP3 标准。ASAP1 作为应用层同控制设备之间接口的标准，定义了应用测量标定系统（MCS）和 ECU 之间的物理和逻辑连接，分为 1a 标准和 1b 标准。其中 1a 是 ECU 和 MCS 之间的物理层和协议层，CCP 就是从属于这

个标准的协议。1b 则定义了 1a 和 MCS 之间的逻辑连接，将不同原理的设备统一起来，诸如：不同的总线（CAN 总线、诊断总线等），不同的终端设备（ECU、其它测量设备）。ASAP2 标准对 ECU 功能和接口及标定信息进行了标准和规范化的数据库，MCS 对控制器的参数标定和数据测量都是基于这个按该标准生成的数据库。ASAP3 定义了 MCS 系统和用户之间的接口，使用户可以通过调用标准化函数同 MCS 系统进行数据和命令交互来实现测量、标定和诊断的功能[10]。

2.2.2 CCP 协议通信方式

CCP 协议属于 CAN 总线的应用层协议。最新的 CCP 协议 2.1 版本支持 CAN2.0B(11 位标准帧或 29 位扩展帧)，同时兼容 CAN2.0A。CCP 是一种基于主从通信模式的协议，系统中只有一个主机，它可以连接 CAN 总线上的一个或多个从机。由主机（标定系统）发出所有命令，从机（待标定 ECU）根据主机的指示完成相应的操作[11][12]。如图 2-4 所示：

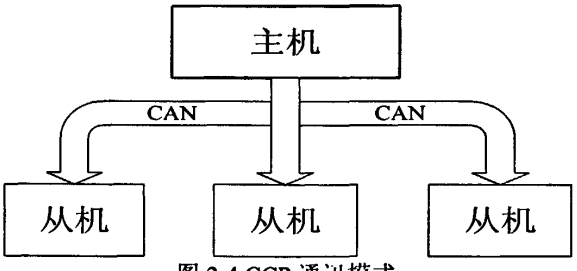


图 2-4 CCP 通讯模式

Figure 2-4 CCP communication model

主机通过从机分配的地址实时地建立主、从机之间的逻辑连接，连接在另一从机被选中或发出断开当前从机连接命令之前一直有效。建立完成主机与从机的逻辑连接以后，它们之间的数据传输均由主机控制，从机负责对主机命令的处理并作出命令响应，包括命令正确执行以后的数据结果返回和命令执行失败的错误代码返回。在本文系统中，主机即为 PC 机上的标定平台软件，从机为电池管理系统。

2.2.4 CCP 协议报文帧

CCP 协议主要依赖两个 CAN 消息帧：主机发送给从机的 CRO(Command Receive Object，命令接收消息)和从机发送给主机的 DTO(Data Transmission Object，数据发送消息)。CCP 协议通信以 CAN 帧为基本单位，根据 CAN 总线规范，发送和接收各使用一个 ID，共占用 CAN 总线报文两个 ID 标志符，所有的收

发的数据都被打包成最多出 8 个字节组成的数据场的报文。CRO 与 DTO 占用数据帧中数据场，最大长度不得超过 8 个字节。下图是 CCP 主从机之间的工作交互图。

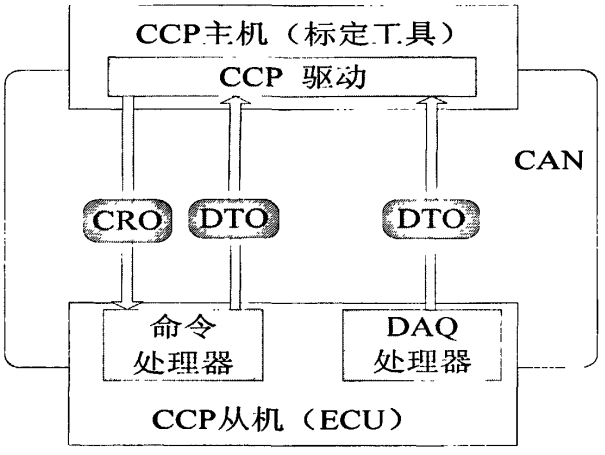


图 2-5 CCP 主从机交互图

Figure 2-5 CCP master and slaver interaction diagram

命令接受消息 CRO 用于主设备向从设备发送的命令，CRO 使用每个数据字节表征基于命令的信息项目。如图 2-6 所示，CRO 数据场的第 1 个字节为命令代码 CMD(Command Code)，CCP 协议共规定了 28 条命令。从设备通过 CMD 代码判断主设备请求并执行相应的命令和做出相应的响应。数据场的第 2 个字节是命令计数器 CTR(Command Counter)，用于跟踪当前系统命令已发生的次数，也可以在主机中跟踪从设备的命令是否已经得到响应。剩余 6 个字节均为命令参数，每条命令有各自对应的命令参数，从设备接收到 CRO 后返回一个 DTO 帧。

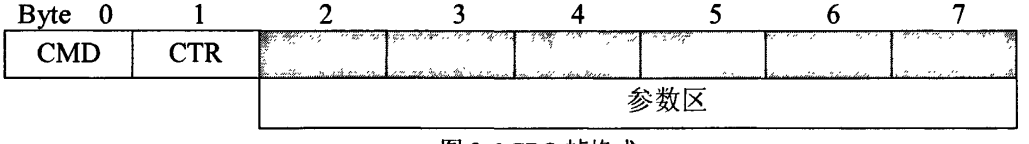


图 2-6 CRO 帧格式

Figure 2-6 CRO frame format

CRO 的数据长度总是定为 8 个字节，由于不同命令所需要的参数不同，长度也不同，对于不用的字节可以填入任何值。

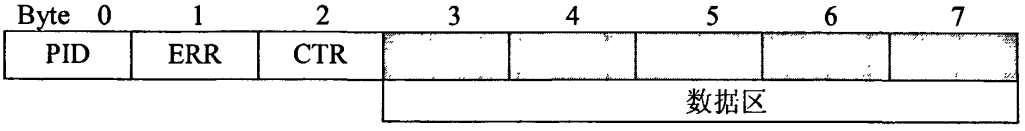
数据发送消息 DTO 是从机的返回消息，CCP 将 DTO 分为了三类：

(l)、命令返回消息(CRM)：作为 CRO 的应答发送给主机，消息的结构如图 2-7(a)所示。

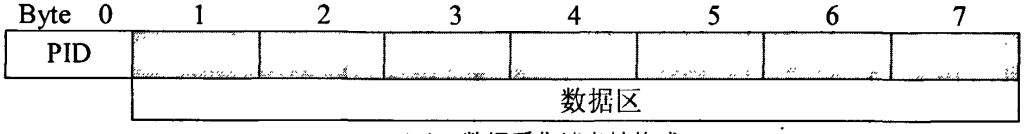
(2)、事件消息 (Event Message)：当从机内部状态出现异常错误或更改其他服务时，由从机向主机报告事件的详细信息并唤醒相应的错误机制进行处理；它的消息的结构和命令返回消息的结构相同。

(3)、数据采集消息(Data Acquisition Message)：在 DAQ 测量模式下，该消息

用于从机按照设定的周期自行向主机上传所需采集的数据，消息的结构如图 2-7(b)所示。



（a）、CRM/事件消息帧格式



（b）、数据采集消息帧格式

图 2-7 DTO 报文帧格式

Figure 2-7 DTO message frame format

在上述的消息帧格式中它们有一个共同的起始字节 PID(Packet ID)，它是用来指示 DTO 的类型，含义如表 2-1 所示。ERR 在 CRM 中是命令返回代码，在事件消息中是错误代码。CTR 为命令计数器，该字节数值与其对应的 CRO 的 CTR 值相对应，在事件消息中无意义。数据采集消息帧仅以 PID 作为消息帧的区分标志，而且 PID 也是该数据采集消息帧的 ODT 编号，用于分段传输和组合 DAQ 数据，剩余的 7 个字节全部作为传送给 PC 端所需要的监测数据之用。

表 2-1 PID 含义表

Table 2-1 PID meaning

PID	DTO 消息类型
0xFF	命令返回消息（CRM）
0xFE	事件消息（Event Message）
0~0xFD	数据采集消息（DAQ）

2.2.4 CCP 数据获取方式

CCP 协议中，主机发给从机的命令可以分为两种：一般控制命令和数据请求命令，分别对应了两种数据测量模式：一种是查询模式（Polling），另一种就是数据采集模式（DAQ-ODT）。

查询模式采用的是一问一答的方式，所有来自主机的命令都需要所选从机的 CRM 报文的响应（含有命令返回代码或者错误代码）。主机与从机的逻辑连接初始化以后，主机与从机之间的数据通讯都是由主机命令进行控制。在这种模式下，当标定系统的上位机需要知道某一地址、某一长度的数据的时候，预先在系统中设置好该地址、长度、监测周期和监测模式等信息后，由上位机按照该周期通过

CAN 总线向从机发送获取指定地址和长度的数据请求命令，然后从机就返回一个对应区域和长度的数据至上位机。

这种查询模式实现比较容易，这个过程的数据交换容易控制，可靠性比较高。但是从机每上传一次数据都必须先由主机发送一个命令帧，这样不仅浪费时间，还会占用 CAN 总线的带宽，增加 CAN 总线上的负载率。虽然 CAN 总线的速率很高，这样频繁往来的监测数据也难免会造成 CAN 总线堵塞和帧丢失的情况。

分析查询模式，不难发现，如果能够让从机自己按照预先设定的周期传送对应区域和长度的数据至上位机，就可以减少因主机发送数据请求命令给总线带宽带来的负面影响。于是 CCP 协议在查询模式之外还定义另外一种数据采集模式（DAQ-ODT）。

DAQ-ODT 模式不需要主机向从机发送数据请求命令来实现数据的上传，而是从机根据预先主机给从机的配置信息按照一定的周期，或触发了一定的事件自行的将所需要采集的数据发送给上位机，这样就可以实现上面降低总线负载率的目标，或者在相同的负载率下能够传递更多的数据。但这种在多通道采集时实现起来相对比较复杂。

在 DAQ 模式下，需要对所需要采集的数据进行配置，所有的配置信息都保存在对象描述表（Object Descriptor Table，ODT）当中，配置过程的完成是在 Polling 模式下进行的。在配置过程结束后，从机就可以根据主机的配置信息确定要上传给主机的数据以及相应的上传周期，自行上传数据，无须主机进一步通过命令来控制。ODT 表是 DAQ 模式实现的基础，其组织结构如图 2-7 所示。

DAQ 和 ODT 在数据结构的表现形式是以数组，也就是说在从机内存里可以建立多个 DAQ 对象，每个 DAQ 对象又可以由多个 ODT 对象组成，主机需要获取的数据信息都存放在 ODT 中。图 2-8 中所示 ODT 结构中，只是示意性的画出一个 DAQ List，在这个 DAQ List 中含有三个 ODT 对象。每个 ODT 对象最多能够包含有 7 个元素，每个元素包括的内容有：数据所在的内存地址，数据长度。这是因为每个 ODT 的数据信息是放在一个 DTO 中进行传输，二者是一一对应的，不能一个 ODT 由多个 DTO 来完成数据传输，也不能一个 DTO 来完成多个 ODT 的数据传输。每个 DTO 的数据区只有 7 个字节，所以每个 ODT 能够至多包含 7 个单字节的数据信息（数据所在的内存地址和数据长度）。当所要获取的数据长度不止一个字节时，ODT 中所能包括的元素个数就会少于 7 个。ODT 存储数据信息也是需要占用从机内存空间，实际应用中为了可以节省 ODT 所占的内存空间，可以省略数据信息的长度，让数据信息的长度取默认值：1 个字节，此时由上位机来完成多字节变量的拆分与拼装。

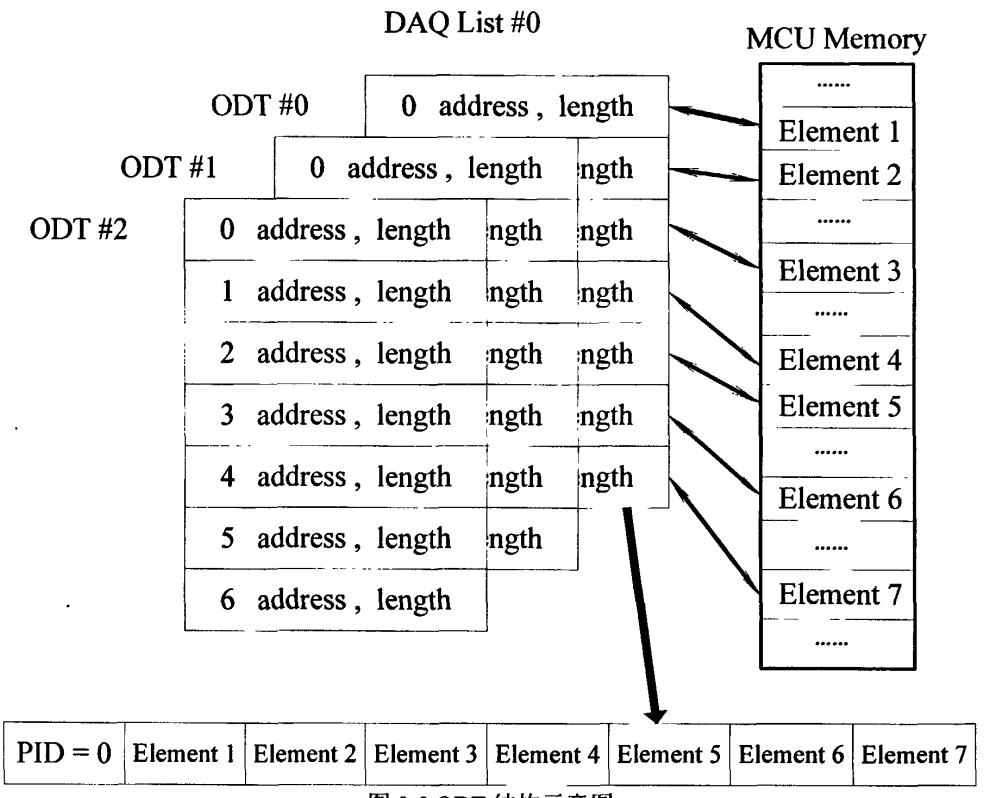


图 2-8 ODT 结构示意图

Figure 2-8 ODT structure

每个 ODT 都有一个唯一的编号，即上文提到的 PID，这样系统至多可以拥有 253 个 ODT 对象。CCP 协议既支持同步周期性采集，也支持基于事件的非同步采集。其实现是依靠不同事件通道进行采集的原理。多张同周期或同事件的 ODT 将构成一个 DAQ 列表(DAQ List)。每个 DAQ 列表可以放入不同的通道进行数据采集，也可以将采集周期为相同倍数的 DAQ 列表放入相同的通道进行数据采集，系统可以同时拥有和激活多个 DAQ 列表。虽然理论上可以同时测量所有已定义的 DAQ 列表变量，但由于 DAQ 模式的多通道采集消耗系统资源较大，实际应用中根据系统的性能不同，所能同时采集的 ODT 上限也不同，这需要在实际试验中确定一个合适值[6][13][14]。

DAQ List 的数目以及每个 DAQ List 中 ODT 的数目由从机实现。主机对 ODT 的配置命令依次完成以下的动作：(1)、获取从机所实现的 DAQ List 的数目及 ODT 数目。(2)、向相应的 ODT 元素中赋予内容，即数据存储地址和数据长度。(3)、针对不同的 DAQ List 赋予事件通道(Event Channel)，同一个 DAQ List 中不同的 ODT 享有同一个 Event Channel。Event Channel 由从机程序实现，从机可以按照一定的周期(如 100ms)发布事件，与 Event Channel 对应的事件会触发从机上传相应 DAQ List 中所有 DTO 对应的 DAQ-DTO 报文。

ODT配置完毕以后，从机就按照所规定的触发事件自行上传DAQ类型的DTO报文(DAQ-DTO)。DAQ-DTO报文的数据区由8个字节组成，第一字节为PID，标识了这帧报文所对应的ODT，而后的七个字节对应该ODT所指向数据内容，从而完成数据内容的上传。

2.2.5 CCP的错误处理机制

CCP协议总共定义了28条不同的命令，标定系统的所有功能的实现都是这些不同命令的组合。CCP协议中有11条命令是标定系统所必须实现，另外17条命令可以根据系统实际需要进行取舍。CCP协议命令一览表如附表所示。

除了丰富的命令，CCP协议还拥有完整的错误处理机制。在表2-2中给出了反映系统各异常状态的命令返回代码信息。错误类别含义为：C0警告，C1干扰，C2可恢复的错误，C3不可恢复的错误。

表2-2 CCP命令返回代码

Table 2-2 Return code of CCP command

错误代码	描述	错误类别
0x00	没有错误	—
0x01	DAQ处理器过载	C0
0x10	命令处理器忙	C1
0x11	DAQ处理器忙	C1
0x12	内部超时	C1
0x18	请求Key错误	C1
0x19	请求状态错误	C1
0x20	冷启动出错	C2
0x21	标定数据初始化出错	C2
0x22	DAQ列表初始化出错	C2
0x23	数据更新错误	C2
0x30	未知命令	C3
0x31	命令语法错误	C3
0x32	参数超出范围	C3
0x33	访问被拒绝	C3
0x34	过载	C3
0x35	访问被锁定	C3
0x36	资源/功能不可用	C3

系统的任务的状态图及错误处理机制如图2-9所示：

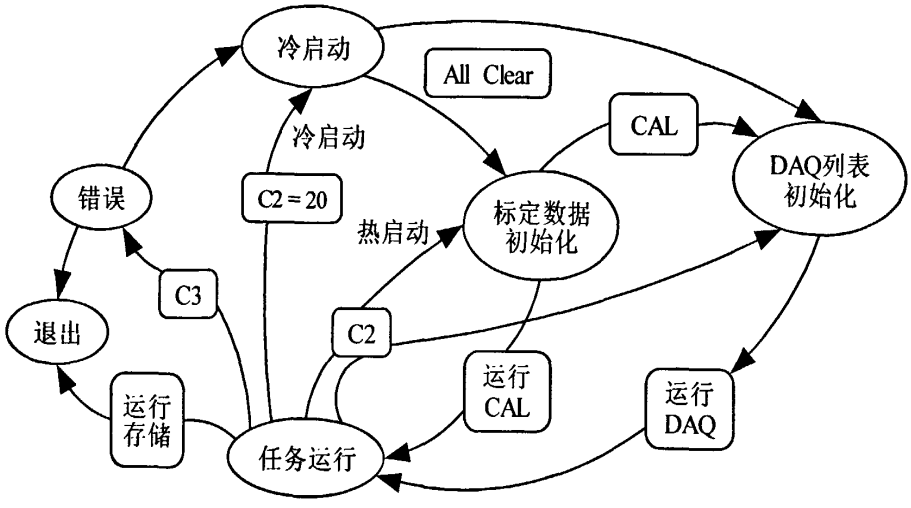


图 2-9 CCP 错误处理机制
Figure 2-9 Error handle of CCP

其中冷启动执行如下操作：首先使用 CONNECT 命令建立主从机之间的逻辑连接，然后通过 GET_SEED 命令登陆从机命令处理进程，最后使用 EXCHANGE_ID 命令完成自动配置任务，以及数据自动更新[11]。

2.3 本章小结

本章主要对 CAN 通信总线作了简要介绍，对基于 CAN 通信的标定协议 CCP 的通信方式、报文帧格式、数据获取方式及错误处理机制作了详细的解读和分析。

3 电池管理标定系统组成

电池管理标定系统的开发工作主要包括三个部分：标定对象——电池管理系统；标定系统的软件开发，包括 CCP 协议的驱动程序，电池管理系统支持 CCP 协议的应用程序及支持 CCP 协议应用的标定平台软件；以及最后的标定试验，以验证标定系统的正确性和可靠性。

3.1 标定系统的功能分析

3.1.1 标定系统基本功能

标定系统需要具备以下的基本功能：

(1)、数据的采集，能够完成电池管理系统测量和控制的信号的实时采集，从而完成动力电池的工作状态的监控。

(2)、数据的显示，电池管理系统将现场采集得到的数据传递到 PC 机标定平台软件进行显示。软件提供数字、图形等多种方式的显示，方便标定人员对动力电池的状态作直观的评价，作为后续标定工作的依据。

(3)、数据的修改，通过标定平台软件，标定人员能够方便的对电池管理系统进行控制，并对其内的控制参数值进行修改，以适应不同标定目的的需要。

(4)、数据的存储，包括标定过程中实时监测到的参数值存储，以及标定完成后试验结果的存储。

3.1.2 自动化标定

随着标定技术的不断提高，自动化标定已成为研究的热点。其基本原理是按照预先设定的模型或算法由标定系统自行完成标定过程，无需人工干预，从而大大减少了标定人员的工作量。自动化标定特别适用于在汽车具有特殊使用要求以及相关法规的特定规定下，用来确定汽车内部各控制单元的最优控制参数值。

本课题的标定系统可以完成标定系统的基本功能，在目前尚不能完成自动化标定的工作。系统可以实现电池管理系统的在线标定的功能，同时还要充分考虑到系统通用性，即可以让标定平台对其他由控制器支持的系统也能完成标定系统的基本功能。

3.2 标定系统总体结构

标定系统的硬件组成结构如图 3-1 所示，主要包括标定用的 PC 机、USB-CAN 通讯单元及标定的对象——电池管理系统。

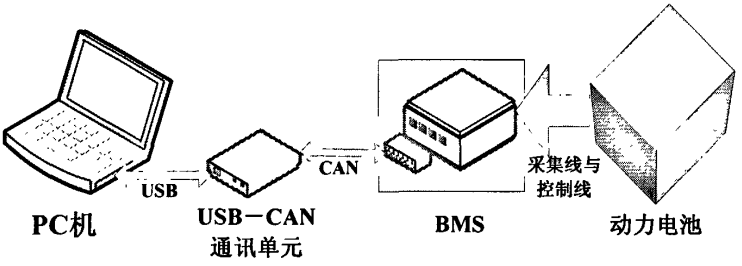


图 3-1 电池管理标定系统组成框图

Figure 3-1 Components of calibration system of BMS

CCP 协议是属于 CAN 总线的应用层协议，是基于 CAN 的应用。目前不管是台式 PC 机还是便携式 PC 机都不支持 CAN 接口，必须在标定用的 PC 机与电池管理系统之间插入一个通讯转换单元，来实现二者之间的通讯和命令交互。现实应用中，台式 PC 机比较常用的接口是 RS232 串口、USB 接口和 PCI 接口，但便携式 PC 机对 RS232 串口的支持也越来越少，许多 RS232 应用也是通过 USB 接口来实现的。PCI 接口的通讯速度快，但是需要额外的安装操作，使用并不是很方便，并且在上车测试时基本上都是使用便携式 PC 机，如果在便携式 PC 机使用 PCI 接口就会显得更加困难。而 USB 接口不管在台式 PC 上还是在便携式 PC 上都有很好的支持，而且也可以保证通讯的速度，使用方便。本课题所采用的就是 USB 接口，通过 USB-CAN 通讯单元实现 PC 机与电池管理系统的数据交换。

PC 机作为标定系统的最上层，对整个标定系统统一控制和管理，并完成数据分析和存储。PC 机同电池管理系统的数据交换就是通过上面提到的 USB-CAN 通讯单元实现连接的。

3.3 电池管理系统硬件设计

标定系统所要标定的对象是项目研发的混合动力汽车电池管理系统，整个标定系统设计的对象原型就是长安汽车 CV8 型混合动力汽车电池管理系统。此次电池管理系统的硬件设计较之前的管理系统有了较大改动，包括系统可靠性方面、电磁兼容性方面、操作灵活性方面等。硬件平台也是后面章节阐述软件开发的基础。

3.3.1 电池管理系统设计要求

动力电池是混合动力汽车的重要组成部分之一，在电池本身能够达到系统要求的前提下，对动力电池的状态监控及管理是混合动力汽车的重要研究内容之一。电池管理系统已经不仅仅只是电池状态的监控单元，而是需要参与到整车控制当中，为整车控制器的控制策略提供实施的依据，并对动力电池充放电加以控制以防止过充和过放对动力电池性能和寿命造成影响[15]。因此混合动力汽车对电池管理系统有如下要求：

(1)、实时准确地监控动力电池的基本状态：混合动力汽车在方案设计之初就会确定系统所需要的动力电池的容量，该容量不仅反应在电池单体的安时数上，同时也反应在电池要达到的电压等级上。混合动力汽车多采用镍氢电池，因此动力电池通常只需要将多个相同安时数的电池单体进行串联起来以达到所需电压等级，这样会使最后的电池包是由很多个电池单体串联而组成的。电池管理系统需要实时监控的基本参数括电池电压和电池温度，这样就会带来一个问题：是不是需要给每一个电池单体单独进行电压和温度检测。理论上讲，这种方式是最理想，但是实际应用中这种方式是不可取的。因为这样做的话，会导致电池包里面有很多的数据采集线束，同时会增大电池管理系统体积、复杂成度，同时还会降低电池管理系统的可靠性和电池包安全性。目前采用的较为合理的方式是这样的：将一定数量的电池单体分成一组，对这一组完成电压采集，称之为模块电压。这样就可以将电池包中的电池分成多个模块进行参数监控。电池的温度可以根据电池包的结构合理选择几个典型的采样点即可。除此之外，电池管理系统还需要实时监控动力电池的端电压和流过电池电流。

(2)、动力电池高压绝缘测量：在混合动力汽车里，动力电池属于高压部分，因此需要考虑其安全性。根据国标相关规定，在绝缘性能低于安全范围时需要进行绝缘报警，防止伤害人身安全。

(3)、较为准确地估算动力电池组的荷电状态（SOC）：SOC 是整车控制策略当中的一个重要信息，整车的一个重要控制目标就是控制电池的 SOC 在一定的范围之内，长安混合动力汽车所要控制的 SOC 目标范围为：30%～70%。

(4)、动力电池故障分析：动力电池是由多个电池单体串联组成的，实际测量是由多个模块实现的。因此任意时刻流过每个单体的电流是相同的。故障电池就可以通过模块电池电压反应出来。另外，电池温度也可以很好的表征电池当前的状态，通常如果所采温度点的温度过高就很有可能说明当前电池有故障。这个时候需要做出故障报警。

(5)、完善的通讯机制：电池管理系统需要通过 CAN 总线同整车控制器进行数据交互，另外还需要独立的用于标定的 CAN 总线。

(6)、可靠的握手机制：为了提高整车的可靠性和安全性，系统设计新添加了

握手机制，它是由硬件实现的一条单总线。它作为整车 CAN 通讯的补充，防止 CAN 通讯间歇中断或 CAN 通讯直接失灵，也可以用来防止电池管理系统工作异常带来的危险。

除了这些混合动力汽车对电池管理系统的要求，作为电池管理系统的设计者还需要分析实车运行中电池的数据，以进一步改进和提高电池管理系统整体性能。目前系统设计还采用比较成熟的 RS232 通讯接口，实车运行时也可以用自己开发的行车记录仪通过 RS232 接口来采集电池管理系统的运行数据。

3.3.2 电池管理系统总体结构设计

电池管理系统结构主要分为集散式和集中式，两种结构方式各有优缺点，集中式测量具有可靠性高，成本低的优点，但是检测速度相对慢；集散式测量速度相对较快，但成本较高，系统较为复杂，电池箱内走线较多，不利于系统的维护。采用哪种结构方式主要取决于所应用的车型以及电池包的数量和排布方式[16]。长安混合动力汽车所采用的电池包只有一个，而且空间结构的设计也比较紧凑。电池管理系统所采用的结构是集中式结构。系统的硬件设计框图如下图所示：

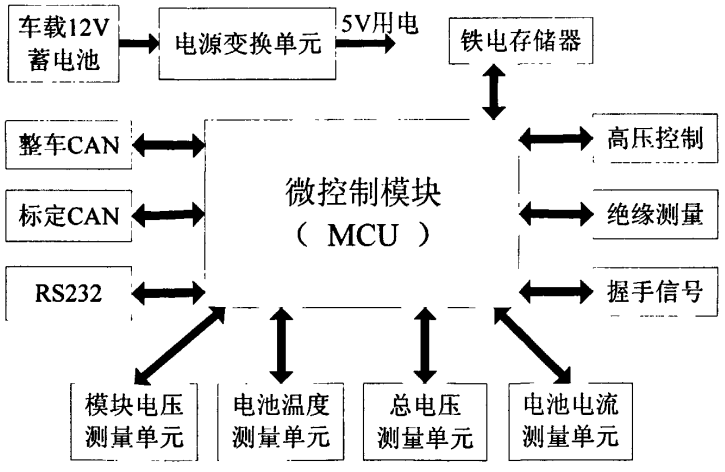


图 3-2 电池管理系统硬件设计框图
Figure 3-2 Hardware structure of BMS

3.3.3 硬件系统设计

3.3.3.1 控制核心

本课题选用 Freescale S12 系列单片机，具体型号是 MC9S12DT128。该系列单片机是以 CPU12 内核（Star Core）为核心的 16 位单片机，采用 Motorola 的第

三代快闪存储器，容量 32K-512KB，具有快速编程能力、灵活的保护与安全机制，有利于软件版权的保护，而且擦除和写入无需外加高电压。MC9S12DT128 采用 5V 供电，总线速度可达 25MHz，主要用于工业控制，特别适合用在汽车上。这是因为该系列单片机有丰富的 I/O 模块和工业控制专用的通讯模块。

(1)、在支持大容量存储器扩展的同时，内部集成了 128K 字节的 FLASH EEPROM，8K 字节的 RAM；

(2)、具有类似流水线的指令队列，用于缓冲指令代码。

(3)、两个 SCI 通讯接口，两个 SPI 通讯接口，一路 I^2C 通讯接口，三路 CAN 通讯接口；

(4)、八通道 16 位增强捕捉定时器，八通道 8 位脉宽调制器（PWM），两个八通道十位分辨率的模数转换器（ADC）；

(5)、同时该系列单片机支持 BDM(Background Debug Mode)，BDM 具备基本的调试功能，包括资源访问及运行控制，与指令挂牌及断点逻辑配合，可以实现许多重要的开发功能，例如系统开发、在线检测、现场检测和芯片程序的固化。从另一个角度来看，BDM 固件相当于一个小监控程序，所谓 BDM 激活实际上就是 CPU 挂起用户程序，将控制权交给这个监控程序。BDM 控制逻辑与外部宿主开发系统之间通过 BKGD 引脚实现双向串行通讯，这种单线通信方案使得用于开发支持的引脚数减到最少[21][22]。

3.3.3.2 模块电压测量方案设计

模块电压测量方案采用差模测量方式，其测量结构示意图如图 3-3 所示：

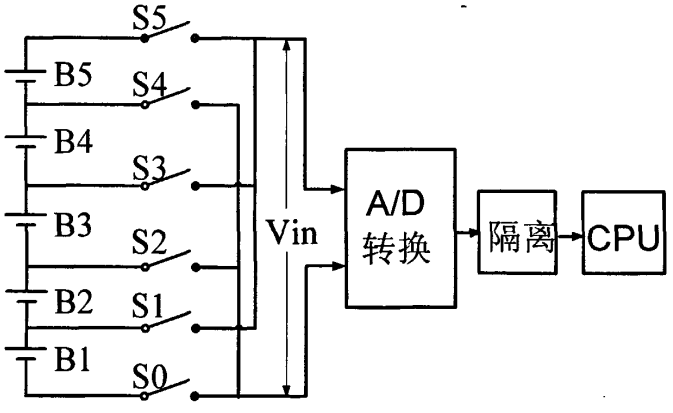


图 3-3 电压测量结构示意图

Figure 3-3 Structure of voltage measure

电池包中每个模块由 12 个单体组成，标称额定电压是 14.4V，每个模块通过电子开关控制其经过采样电路进入 A/D 转换器。电池电压测量过程为：当测量电

池 B1 的电压时，则闭合开关 S0 、S1（其余开关全部断开），这时候 Vin=VB1，经过 A/D 转换芯片完成模数转换，再由 CPU 通过隔离光藕获取转换的结果；下一时刻，需要测量电池 B2 的电压，这时只需要先断开 S0，然后闭合 S2，此时 Vin=-VB2。依次类推，依次可测量其它电池的电压，并且此种测量方法各个电池互相独立，不存在误差累积的问题，所以测量精度更高。另外，高压电池侧和低压 CPU 侧通过光藕隔离，避免了高压侧对低压侧的威胁。

在上面的电池切换的过程中，会发现接入到 A/D 中的电压 Vin 是正负交替变化的，因此对 A/D 提出了新的要求。方案设计中采用了美国 MAXIM 公司的 MAX110 芯片，它是一种双通道串行 Σ—△模数转换器，采用内部自校准和过采样技术，具有精度高、功耗低、抗干扰能力强、无需外围元件、易与各种微控制器和微处理器接口等特点。A/D 在设计中采用双电源供电，两个通道中的一个负责正电压的转换，另一个通道负责负电压的转换。负责负电压转换的通道在信号进入 A/D 之前已经将其变换成正信号，方法就是交换信号输入线的位置。通过这样处理以后，A/D 在进行转换时，所转换的信号就都是正信号了。通过采用双电源供电，可以保证芯片信号输入引脚的绝对电压在电源电压范围之内。另外，A/D 的参考电压源是单电源供电方式，这样就有必要将正负信号统一转换成正信号以后再加到 A/D 里面去。

A/D 的参考电压源是 1.8V，加入到 A/D 的信号范围-1.8V 到+1.8V 之间，而模块电压的额定标称是 14.4 V，实际电压范围是 12V 到 20V 左右，因此需要在模块电压与 A/D 输入之间加入采样电路，方案中采用的是电阻分压的方式以满足加入到 A/D 的电压信号范围的要求。电子开关采用松下的光控 MOS，它可以很好的完成模块电压逐次切入 A/D 的工作。

3.3.3.2 温度测量方案设计

工业应用中，温度测量比较常用的方式就是采用热电偶和热电阻来实现测量的：热电偶测量范围广，测量误差低，但由于输出电压与热端和冷端的温差有关，在实际使用时需要进行冷端补偿，需要使用专门的滤波放大电路，并需要设计专门的冷端温度检测电路；热电阻指利用金属的电阻率随温度变化的特性做成的测温器件。在中间温度区热电阻呈现线性特性，在低温或高温区热电阻和温度呈非线性，温度响应速度较慢，只适合一般精度要求不高的场合。二者都是模拟信号，都需要单独引线通过信号调理电路进行信号调理，然后通过 A/D 进行模数转换，最后由控制核心来处理测量结果。

而在电池包里面，温度的采样点比较多，如 CV8 型混合动力汽车需要 6 个温度点，如果采用上述的热电偶和热电阻的方式来测量温度，会使电池管理系统的设计复杂化，灵活性变差，同时需要在电池包里面布置的温度采样线会很多，这

就会使电池包里面空间利用变得更加窘迫。

方案设计中采用智能化温度传感器，它把温度传感器，外围电路，A/D 转换器，微控制器和接口电路集成到一个芯片中，具有温度测量，温度控制和与微处理器数据通讯的能力。本系统使用的是美国 DALLAS 公司推出的一种单总线数字式温度传感器 DS18B20。温度测量范围为-55℃～+125℃，可编程 9-12 位 A/D 转换精度，测温分辨率可达 0.0625℃，由于电池管理系统设计选择 9 位的分辨率，温度测量精度为 0.5℃，转换周期 93.75ms。每个 DS18B20 在出厂时都用激光进行了调校，具有独一无二的 64 位序列号，所以多个传感器可以同时挂在一条数据传输总线上，64 位序号值存放在 DSl820 内部的 ROM(只读存贮器)中，开始 8 位是产品类型编码(DSl820 编码均为 28H)接着的 48 位是每个器件唯一的序号，最后 8 位是前面 56 位的 CRC(循环冗余校验)码。CPU 只需一根端口线就能与多个 DS18B20 通信，占用 CPU 端口少，可节省大量的引线和逻辑电路。以上特点使得 DS18B20 非常适用于温度测量网络的建立。

DS18B20 采用单总线的方式进行温度测量结果的访问，这种特殊的数字接口对 I/O 口的操作时序要求比较严格，同时对总线信号边沿的上升时间和下降时间都有特别的要求，这就需要在实际使用当中不仅要注意合理的硬件设计，还要做好抗干扰的设计，合理的布线长度和屏蔽措施能够提高温度测量的可靠性。

3.3.3.3 电流测量方案设计

精确测量电流是准确估算电池 SOC 的基本条件之一。由于霍尔型的电流传感器在测量范围的低端线性度很难保证，在电池管理系统的设计中采用分流器来测量电流，在整个量程内都可以保持比较好的线性度。但使用分流器带来的一个问题是温度系数。对于霍尔型的电流传感器而言，一般都把温度漂移的补偿封装在传感器内，使用时不需要过多考虑温飘的问题。分流器的工作原理为电阻分压，在持续大电流工作条件下会出现发热的情况，进而改变分流器的阻值，影响测量精度。实际工作中，分流器安装在电池包中，处在电池的风冷系统中，同时分流器的选择考虑了足够的余量，可以基本保证工作时分流器的阻值变化很小，给测量带来的误差可以忽略不计。

测量使用的芯片为 CS5460A，作为一款电表专用的测量芯片，CS5460A 芯片还可以测量能量和功率。在电池管理系统的设计中，将 CS5460A 的电压测量通道加入恒压源信号，电流测量通道则测量分流器信号，这样测得的能量为电流与时间的积分，即电池电量的计量单位“安时（AH）”，可直接用于 SOC 的计算。该芯片的量程为 250mV，分流器采用国内标准的 FL—2 型分流器，其满量程信号为 75mV，二者可以不需要额外的调理电路来放大信号来完成电流的测量。

由于分流器直接将高压信号引入到电池管理系统，需要将电流信号与低压系

统进行隔离。为了不影响测量的线性度，设计中没有对分流器输入的电流信号进行隔离，而是将 CS5460A 与 CPU 之间的通讯进行了隔离，数字信号的隔离对测量没有影响。这样的优点是保证了测量的精确度，但降低了 CPU 与 CS5460A 之间的通讯速率，可以通过系统软件上的合理设计减少加入隔离电路对访问速率的影响。

3.3.3.4 总电压和绝缘测量方案设计

总电压测量对实时性要求比较高，整车控制器需要电池管理系统每 10ms 就上报一次总电压测量值，因此，总电压信息不能通过将模块电压累加的方式来实现。同上述测量方案，总电压是高压系统，需要做好系统隔离。总电压测量可以采用专门的电压传感器来测量，但这会使整个系统的体积变大，占用较多的 PCB 空间或电池管理系统空间，成本上的投入也比较大。

系统设计采用了传统的电阻分压的设计方案，通过隔离运放来实现系统的隔离，然后将隔离运放输出的信号进行调理，再送入 CPU 内部自带的 A/D 转换器进行模数转换。

在我国的国标 GB/T 18384.1-18384.3-2001《电动汽车安全要求》中，对电动汽车的绝缘状况的定义、测量方法及安全要求都做了明确的规定。电动汽车的绝缘状况以直流正负母线对地的绝缘电阻来衡量。国标中动力蓄电池的绝缘电阻定义为：如果动力蓄电池与地（电底盘）之间的某一点短路，最大（最坏情况下的）泄漏电流所对应的电阻。电动汽车的国际标准和我国国家标准规定：绝缘电阻值除以电动汽车直流系统标称电压 U，结果应大于 100Ω/V，才符合安全的要求。这个值是通过下列条件来选定的：“如果人或其他动物构成动力电池系统与地之间的外部电路，最坏的情况下泄漏电流不超过 2mA，这是人体没有感觉的阀值”。

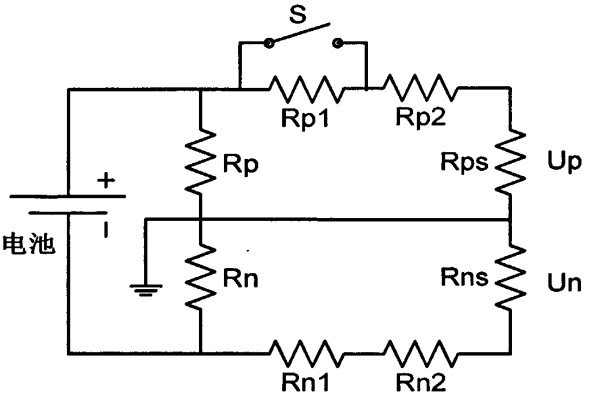


图 3-4 绝缘测量原理图

Figure 3-4 Insulation measurement schematic

设计中采用的方案如图 3-4 所示，其中，Rp、Rn 分别是动力电池的正负母线对混合动力汽车底盘的等效绝缘电阻，Rp1、Rp2、Rps、Rn1、Rn2、Rns 和开关 S

是管理系统设计的绝缘测量电路，Rps、Rns 是用来采样的电阻。测量原理：

(1)、当绝缘等效电阻 Rp、Rn 很大，即整车绝缘性能良好时，系统等效电路可以认为没有 Rp、Rn，电池就是通过 Rp1、Rp2、Rps、Rn1、Rn2、Rns 构成回路的，这时无论开关 S 是闭合还是断开，在采样电阻 Rps、Rns 分得的电压 Up 和 Un 的比值恒定：Up/Un = Rps/Rns。

(2)、当绝缘电阻 Rp、Rn 只有一个很大，另外一个却比较小，即发生不对称绝缘故障时，系统在开关 S 断开的情况下，在采样电阻 Rps、Rns 分得的电压 Up 和 Un 的比值关系已经失去了 Rps/Rns 的关系，不对称绝缘越严重采样电阻分得的电压比例关系越偏离 Rps/Rns。

(3)、当绝缘电阻 Rp、Rn 都比较小，即发生对称性绝缘故障时，如开关 S 断开，则在采样电阻 Rps、Rns 分得的电压比例关系同绝缘良好时的比例关系相近；而在开关 S 闭合时，采样电阻 Rps、Rns 分得的电压比例就失去了绝缘良好时的比例关系。

因此，可以通过比较在不同开关 S 状态下采样电阻分得的电压比例关系就可以确定当前的绝缘性能。分压电阻上的电压采集也是通过 CPU 内部自带的 A/D 转换器进行的。测量回路设计在选择电阻时必须考虑以下几个方面的因素：(1)、基本不影响被测系统原有的绝缘性能；(2)、考虑电阻功率和耐压的要求，可以通过采用多个大功率电阻串联的方式来实现。

3.3.3.5 通讯和握手机制设计

系统设计要求有两路 CAN 通讯接口：一路用于和整车通讯和故障诊断，一路用于标定参数。CAN 通讯由 CPU 中的 MSCAN 模块控制器和 CAN 驱动芯片 PCA82C250 组成，MSCAN 由收发引擎、报文过滤与缓冲、控制与状态管理、时钟、低通滤波器等组成，服从 CAN2.0A/B 协议，支持标准和扩展帧格式，0～8 字节数据段长度，通讯速度达到 1Mb/s；PCA82C250 是 CAN 控制器和物理总线间的接口，它是专用的 CAN 驱动芯片，提供对总线的差动发送和接收功能。

系统设计时充分考虑灵活性，加入一路串口通讯接口。CPU 内部自带两个 SCI 通讯模块，只需要同 CAN 通讯设计一样加入 RS232 电平转换芯片就可以实现串口通讯。CPU 的 SCI 是一种采用 NRZ 格式的异步串行通信接口，由发送器，接受器，波特率产生电路及其他辅助逻辑构成，可以选择 8 或 9 个数据位，发送和接收的奇偶校验位可以选择是否由硬件生成。RS232 电平转换芯片可以实现 TTL 电平和 RS232 电平之间的转换，然后通过 9 芯串行口和 PC 机进行通信。

握手机制是为了系统可靠性而添加设计的，整车网络结构如图 3-5 所示：

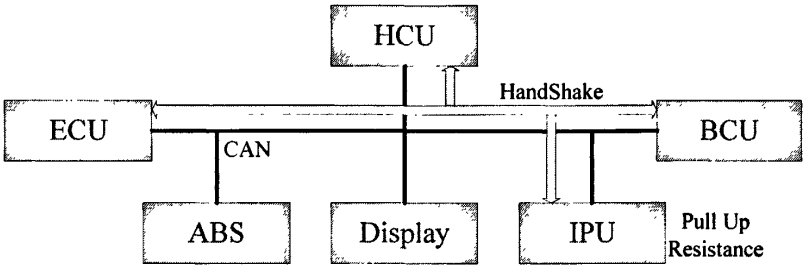


图 3-5 整车网络结构示意图

Figure 3-5 Network architecture diagram of the car

握手机制是通过单总线的结构实现的，该总线上挂有四个单元，分别是整车控制器，电控单元，电机控制器和电池控制单元（即电池管理系统）。四个单元都要时刻检测这条握手信号的电平，正常情况下这条总线上的电平是高电平，HCU 会通过 CAN 总线向 IPU 和 BCU 发送测试握手信号的命令，当 IPU 和 BCU 接收到这个信号就会主动将总线电平拉低 30ms 到 60ms，然后将总线拉高回到正常的高电平。如果 HCU 发送了测试握手信号的命令以后没有检测到总线被拉低，就会报系统三级故障，车辆会立即停止运行。如果总线被拉低以后在 150ms 后仍没有被重新拉高，HCU 也会报系统三级故障，车辆停止立即停止运行。电池管理系统硬件设计如图 3-6 所示：

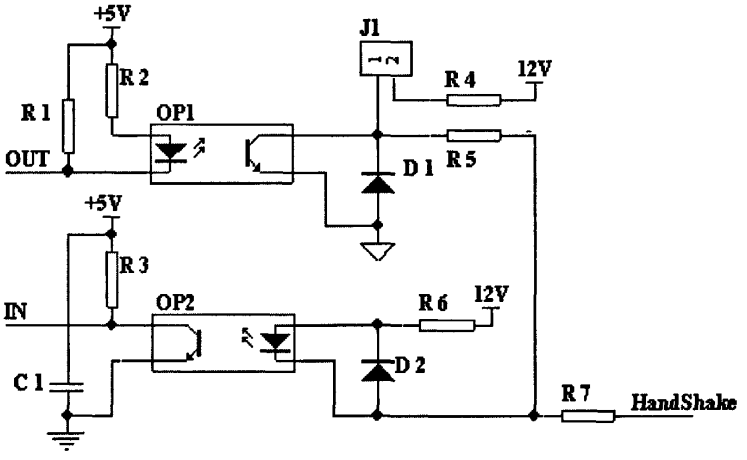


图 3-6 系统握手机制硬件原理图

Figure 3-6 Hardware schematic of system handshake signal

系统通过光藕的方式实现外部单总线同控制核心的隔离，当总线被拉低时，光藕 OP2 会导通，IN 变为低电平；当需要控制总线时，可以通过光藕 OP1 的导通和断开来控制总线上的电平高低。CPU 利用两个 IO 口完成握手信号的状态检测和控制总线上的电平高低，构成了一个闭环检测系统，从而实现握手机制。

3.4 本章小结

本章主要对标定系统进行分析，提出了电池管理标定系统的方案设计，介绍了电池管理标定系统总体结构。并针对混合动力汽车的动力电池管理系统的要求设计了满足系统要求的电池管理系统硬件系统。

4 CCP 协议驱动程序与应用程序开发

在使用标定工具对电池管理系统进行基于 CCP 协议的标定工作前，必须在上位机和电池管理系统上都要有对 CCP 协议的驱动支持，只有二者都能够很好的支持CCP 协议驱动程序才可以实现标定上位机同电池管理系统进行基于 CCP 协议的通讯。因此，对于上位机和电池管理系统的程序开发可以分成三个部分：

(1)、CAN 驱动程序（CAN driver），实现上位机和电池管理系统的 CAN 通讯功能，它为 CCP 驱动程序提供服务；

(2)、CCP 驱动程序（CCP driver），它是这个标定系统设计的核心，用于解析和执行 CCP 命令，实现标定上位机对电池管理系统的标定功能；

(3)、应用程序，包括上位机程序和电池管理系统的应用程序。

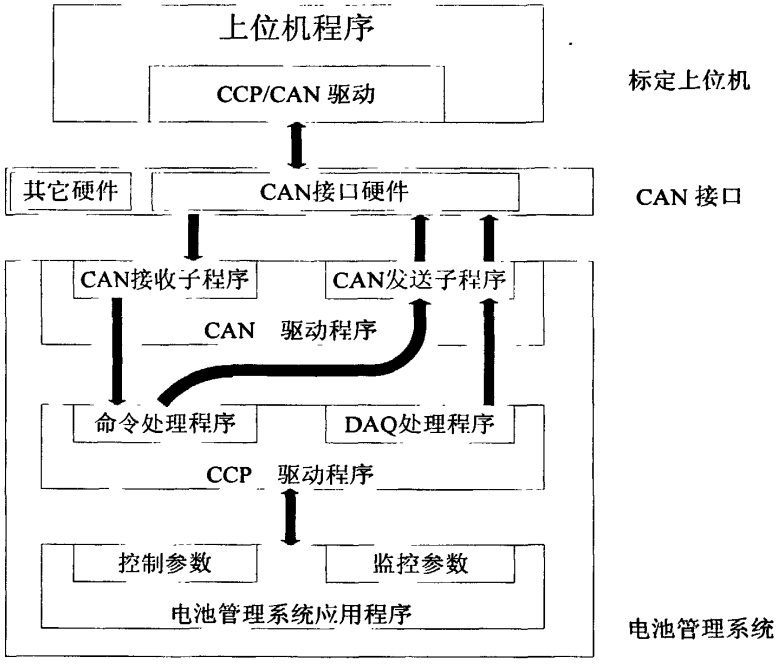


图 4-1 CCP 协议驱动程序与应用程序结构框图

Figure 4-1 CCP driver and application program structure diagram

CCP 协议驱动程序与应用程序结构框图如图 4-1 所示，在标定的过程中，各个程序模块的信息交换过程如下：

(1)、主控单元（上位机）通过 CAN 设备发送支持 CCP 协议的 CAN 信息命令包；

(2)、经过 CAN 通讯接口电路，CAN 信息命令包到达从设备（电池管理系统）的 CAN 接收子程序；

(3)、从设备 CCP 驱动程序处理来自 CAN 接收子程序中的信息，将处理结果通过 CAN 发送程序经通讯接口电路返回给主设备；

(4)、如果是 DAQ 命令，则出 DAQ 处理程序按照之前的配置向主设备返回 DAQ 数据信息。DAQ 处理程序从应用程序待监测的变量中组织应答数据，然后调用 CAN 驱动程序的发送模块，将数据以 DAQ-DTO 的形式自行发送至主设备。

4.1 CAN 驱动程序

CAN 驱动程序是针对 MC9S12DT128 而设计的 CAN 驱动程序。MC9S12DT128 内部集成了 CAN 控制器模块 MSCAN，可以实现基于 CAN2.0A/B 规范的基本要求[21]。CAN 驱动程序需要稳定地实现 CAN 信息的接收和发送功能，它包括三个基本的模块：CAN 控制初始化模块、CAN 数据接收模块及 CAN 数据发送模块。

4.1.1 CAN 初始化模块

MC9S12DT128 的 CAN 模块初始化需要按照一定的步骤来完成模块寄存器的配置，包括配置 CAN 控制器的工作模式、通讯速率、ID 验收过滤及开中断操作。其中 CAN 通讯速率必须与制定的协议一致。ID 验收过滤则用于屏蔽与整车控制器节点无关的报文，减少控制器的中断响应次数和时间开销，提高控制器的工作效率。为了使从设备尽可能不遗漏它应该接收到 CAN 总线上的报文，以接收中断的方式来接收 CAN 总线上相关的报文。图 4-2 是 CAN 控制器初始化流程图：

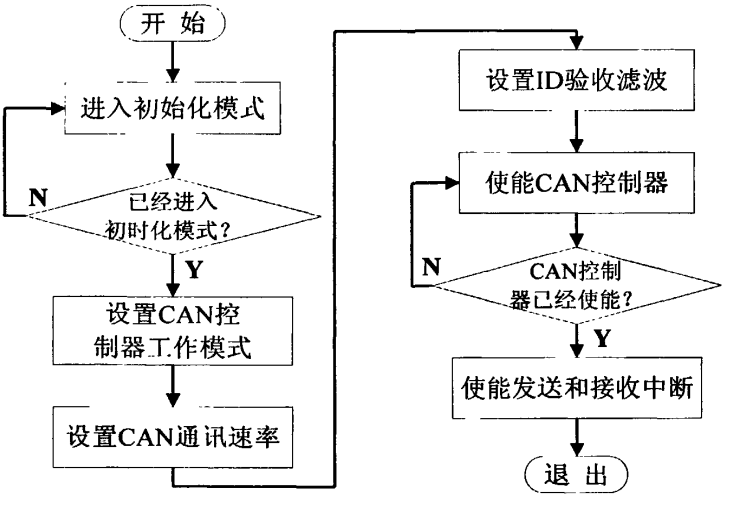


图 4-2 CAN 控制器初始化程序流程图

Figure 4-2 Initial program flow chart of CAN controller

4.1.2 CAN 数据接收与发送模块

CAN 数据接收与发送模块通过中断的方式来实现的，这种采用前后台的方式可以充分利用从设备控制核心的资源，发送程序还会由 CCP 程序主动对其进行调用。接收模块在 CAN 接收到数据时将数据转存，然后让 CCP 驱动程序对转存的数据进行解析和处理；发送程序则将解析和处理的结果发送给标定上位机，MSCAN 内部具有三个发送缓冲区，发送程序只有在发送缓冲器区至少有一个为空的情况下进行发送操作。接收与发送模块的程序流程图如图 4-3 所示：

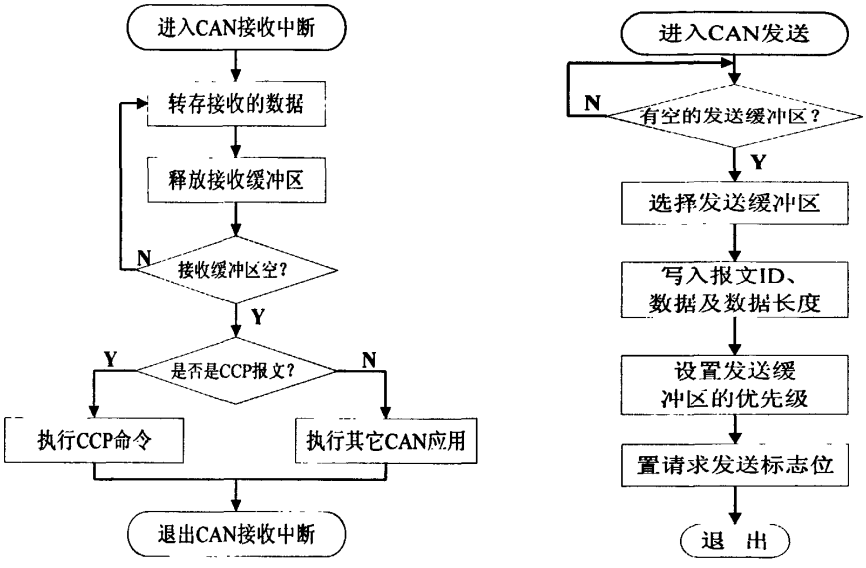


图 4-3 CAN 驱动数据发送和接收程序流程图

Figure 4-3 Transeive and receive flow chart diagram of CAN driver

4.2 CCP 驱动程序

CCP 驱动程序是 CCP 协议的真正实现者，使整个标定系统设计的核心。

4.2.1 CCP 驱动程序基本函数

根据 CCP 协议规范，实现 CCP 的基本功能，需要包括两个命令处理模块：命令处理模块和 DAQ 处理模块。这些实现在程序主要模块 CCP.c 中，其中包含了如表 4-1 所示的 CCP 驱动程序基本函数：

表 4-1 CCP 驱动程序基本函数

Table 4-1 Basic function of CCP driver

命令分类	函数名	说明
系统命令	ccpInit( )	使用 CCP 必须事先调用这个命令初始化 CCP 驱动程序。
	ccpGetSeed( )	获得协议规定 seed，然后根据该 seed 按协议计算访问资源的密匙（key），通过 ccpUnlock 获取资源的访问权限。
	ccpUnlock( )	以获得的密匙获取资源的访问权限。
	ccpSet_S_Status( )	设置从机的会话状态
	ccpGet_S_Status( )	获取从机的会话状态
	ccpCommand( )	这是 CCP 驱动程序中最主要的函数，每次接收到 CRO 命令的时候，都必须调用这个函数。其中包含了实现 CCP 驱动功能的所有 28 个命令，并且通过这些命令再调用其它服务函数，用于实现逻辑连接、解锁、数据上传、下载及 Flash 擦除和写操作等功能。
CAL 命令	ccpSetMTA( )	设置需要标定参数的物理地址，由从机根据物理地址所处的范围来采取对 RAM 区、EEPROM 区及 CPU 自带的 Flash 区不同的操作。
	ccpDnload ( )	由标定平台向从机下载标定的数据
	ccpUpload ( )	由从机向标定平台上传标定参数的数据
	ccpMove ( )	控制从机不同存储区的数据转移
DAQ 命令	ccpGetDAQSize( )	获取从机控制内部所配置的 DAQ 列表的数量以及每个列表所包含的 ODT 数量。
	ccpSetDAQPtr( )	在 DAQ 列表初始化时，选中所要设置元素所在的 DAQ 列表、ODT 及在该 ODT 中所处的位置。
	ccpWriteDAQ( )	在 DAQ 列表初始化时，设置元素的物理地址及数据长度
	ccpStartStop( )	用来控制 DAQ 运行状态，开始、暂停及退出
	ccpDaq( )	这是 DAQ 模式实现的主要命令。按照一定的事件触发（如周期触发），采集预先设定需要采集的数据，并通过 CAN 发送 DTO 消息到标定平台，主要使用在监测功能。

4.2.2 CCP 驱动程序核心函数 ccpCommand( )

图 4-4 为 ccpCommand()函数流程图，该函数作为 CCP 驱动程序的核心函数，CCP 协议中所有需要实现的功能都是在这个函数中执行的。该函数负责解释并执

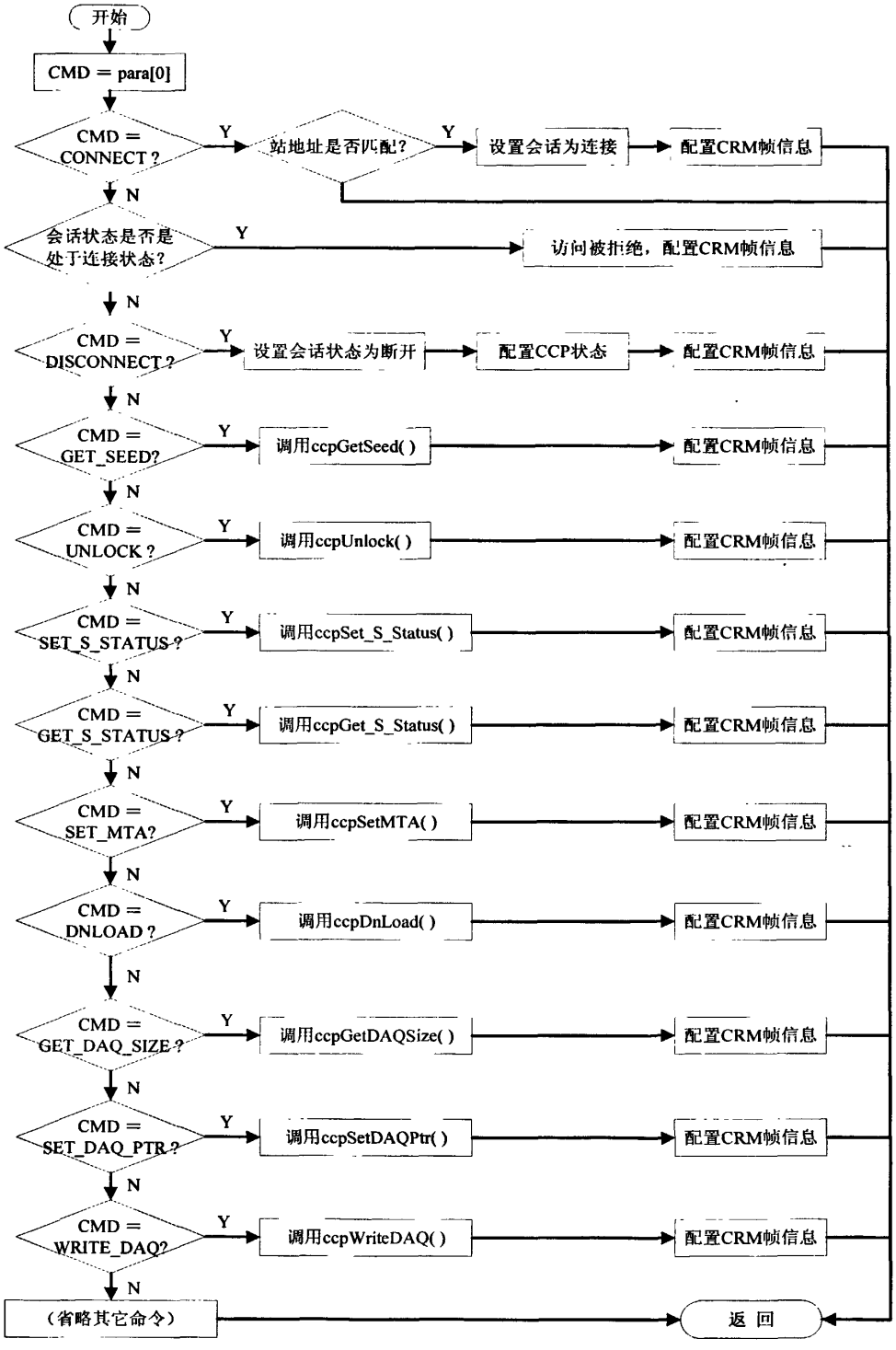


图 4-4 ccpCommand( )函数流程图

Figure 4-4 Flow chart of ccpCommand ( )

行收到的 CRO 命令，其输入量为指向 CAN 数据帧接受缓冲区首字节地址的指针，

通过分析该 CAN 帧的第一个字节（byte0）即命令代码字节（CMD）从而判断该 CRO 帧的功能，在建立好逻辑连接的前提下，采用 switch 结构[25]将任务分配至具体的模块，再调用 CCP 驱动程序中的其它如 ccpSetMTA( )，ccpGetDAQSize( )和 ccpStartStop( )等函数，进行数据处理并组织需要反馈的数据，实现逻辑连接、解锁、数据上传、下载等功能。执行完毕使用 CRM_DTO 返回命令执行情况至上位机，至此命令处理模块完成处理一个 CRO 命令。

4.2.3 DAQ 模式的实现

DAQ 模式的实现包括四个步骤：主机与从机建立逻辑连接，主机获取从机的 DAQ 资源大小，主机根据需要监控的数据完成从机 DAQ 的数据信息的初始化及启动 DAQ 过程。

主机与从机建立逻辑连接是整个标定系统得以运行的基础，主机与从机的各种命令的交互都是以逻辑连接已经被建立为前提，如果主从机没能完成主从机的逻辑建立，则从机对主机发送过来的 CCP 协议命令都会作出拒绝访问的响应。

DAQ 模式的下一步就是要获取从机能够支持的 DAQ 列表的数目及每个表所支持的 ODT 的数目，其中每个 ODT 需要根据标定平台的需要完成初始化，即给 ODT 设置监控参数的地址及数据长度等信息。CCP 驱动在设计时提供三个结构体来描述 DAQ 列表和 ODT 信息及 ODT 的元素信息：

ODT 的元素信息结构体：

```
typedef struct
{
    CCP_DAQBYTEPTR ptr; //ptr 是字节型指针，描述元素的内存地址
    CCP_BYTE size;      //size 是字节型变量，描述元素的长度
}CCP_Element;
```

ODT 信息结构体：

```
typedef struct
{
    CCP_Element Element[7]; //每个 ODT 内最多能够包含 7 个元素
}CCP_ODT;
```

DAQ 列表结构体：

```
typedef struct
{
    CCP_ODT     ODT[CCP_MAX_ODT]; //该列表含有的最大 ODT 数目
    CCP_WORD    prescaler;        //预分频系数
    CCP_WORD    cycle;            //内部使用
    CCP_BYTE    eventChannel;     //事件通道号
```

```
    CCP_BYTE    last;     //记录该列表实际拥有的 ODT 数目
    CCP_BYTE    flags;    //当前 DAQ 运行状态标识
}CCP_DAQ_List;
```

从机是用一维数组的方式来组织 DAQ 列表，通过上面的结构体描述，可以发现每个 DAQ 列表可以拥有 ODT 最大数目是相同，它采用宏定义的方式。实际被利用的 ODT 的数目是通过 last 记录下来的[23][24]。因此当主机需要获取从机 DAQ 资源大小的时候就会把 DAQ 列表的数目和每个 DAQ 列表最大 ODT 数目反馈给主机。即执行 ccpGetDAQSize()。

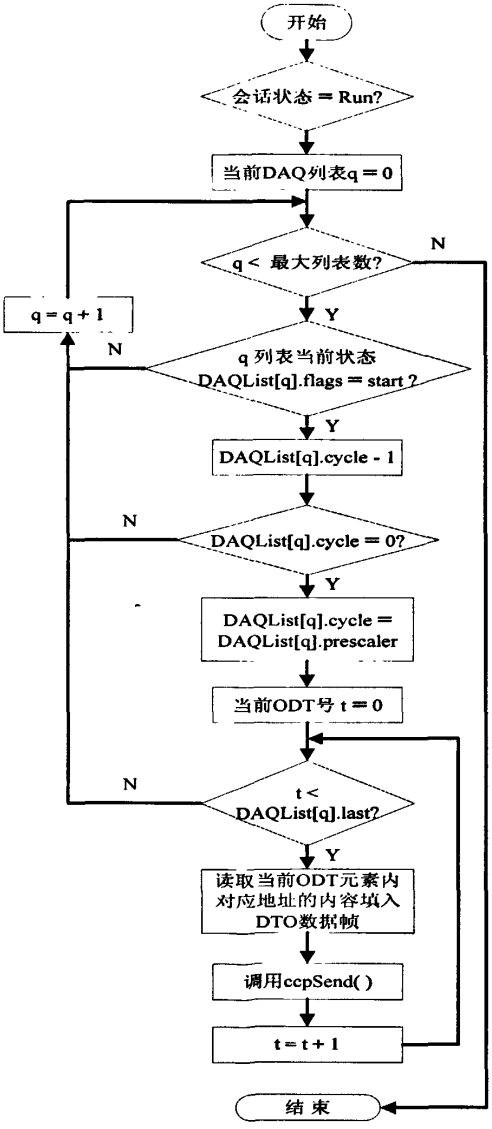


图 4-5 DAQ 函数 ccpDaq() 的程序流程图

Figure 4-5 Flow chart of ccpDaq( ) of DAQ

主机根据获取的从机 DAQ 资源的大小，将需要监控的参数的物理地址和长度

信息发送给从机，以完成对从机 ODT 的元素的初始化。针对不同的 DAQ 列表赋以事件通道(EventChannel)，同一个 DAQ 列表中不同的 ODT 享有同一 EventChannel。EventChannel 由从机程序实现，从机按一定周期发布事件，与 EventChannel 对应的事件会促发从机上传相应 DAQ 列表中所有 ODT 对应的 DAQ-DTO 消息，因此 EventChannel 实际上表示的就是上传周期。

最后就是由标定平台控制 DAQ 的运行状态，由 flags 标志记录当前的 DAQ 的运行的状态。从机的应用程序会周期性调用 ccpDaq()函数，由该函数根据当前 DAQ 初始化信息执行 DAQ 模式。ccpDaq()执行流程图如图 4-5 所示。

4.2.4 数据标定功能

数据标定功能的实现就需要对被标定系统即从机的内存和外部存储设备进行访问，能够对其进行读写操作，并能够完成数据的保存。一般还要求被标定的参数的修改结果在系统断电或重新启动时都不会丢失。本系统采用的是飞思卡尔 MC9S12DT128 单片机，同时还配备有专门的数据存储单元 FM24C64。飞思卡尔 MC9S12DT128 单片机内部的存储单元由三个部分组成：RAM、EEPROM 及 Flash。系统设计时之所以单独配置铁电存储器 FM24C64，是因为单片机内部自带的 EEPROM 的擦除与写入速度比较慢，在擦除与写入时必须间隔 20ms 以后才能对 EEPROM 进行下一步的操作，擦除和写入是两个独立的过程，擦除的最小单位是 4 个字节，写入的最小单位是 2 个字节；铁电存储器的特点就是读写速度快，最高总线访问频率为 1MHz，在高噪声的环境下，数据读写可靠性高，应用灵活。

针对电池管理系统这一应用，标定所选用的方案是：系统各种参数都是保存在铁电存储器中，上电后从存储器中将这些参数的信息读入到单片机内部的 RAM 区，标定平台针对参数所在的 RAM 区进行标定操作，等所需参数完成了标定以后将标定结果回写到铁电存储器当中。其中部分参数在每一个系统当中都是一样的，在完成了这部分参数的标定以后，在以后程序设计中将这部分参数直接写到单片机内部的 Flash 中。因此系统标定的目标主要有两个：单片机内部的 RAM 区和外部的铁电存储器。

MC9S12DT128 单片机单片模式下的存储区域分布如图 4-6 所示，在默认的系统配置下，EEPROM 的存储空间是没有被分配的，即让 RAM 的物理分配从$400 开始分配存储空间。$0～$400 这 1KB 的存储空间被分配给单片机的寄存器，它的存在也覆盖了 RAM 本来从$0 开始的 8KB 字节 RAM 存储空间的开始的 1KB 字节的存储空间，也就说系统实际使用的 RAM 空间大小是 7KB 字节。

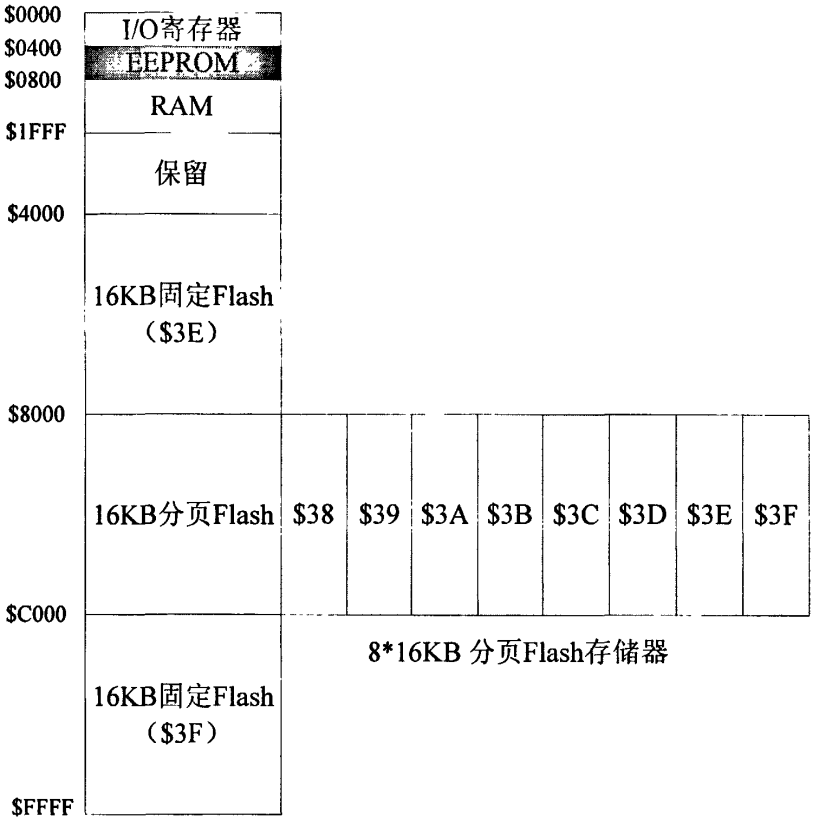


图 4-6 MC9S12DT128 内部存储器空间分配图

Figure 4-6 Memory distribution of MC9S12DT128

由于系统标定的目标包括两个部分，除了上述的标定方案，系统还可对外置的铁电存储器也像 RAM 一样进行读写操作，即将 FM24C64 模拟成 RAM。系统实际使用的铁电存储器的空间并不是很大，小于 1KB，因此可以将针对 FM24C64 的擦写函数的接口和针对 RAM 的写操作的函数接口统一起来，在函数 ccpWriteMTA() 中通过需要操作的内存地址来判断是对 FM24C64 还是对 RAM 的操作。当对空间 $400～$1FFF 进行访问操作时可以断定是对 RAM 的操作，而把对 FM24C64 的存储空间进行操作的地址分配在$2000～$3FFF。ccpWriteMTA( )函数程序流程图如图 4-7 所示。

值得注意的是，在设计电池管理系统的应用程序时应该将系统需要进行标定的参数作为全局变量放在 RAM 中，因为这样变量的存储状态是静态的，有固定的分配空间，它们的生存周期同整个应用程序的生存周期。如果将参数作为局部变量的话，它们的存储空间是动态分配的，没有固定的存储位置，这样标定平台是无法完成对这些参数的修改。

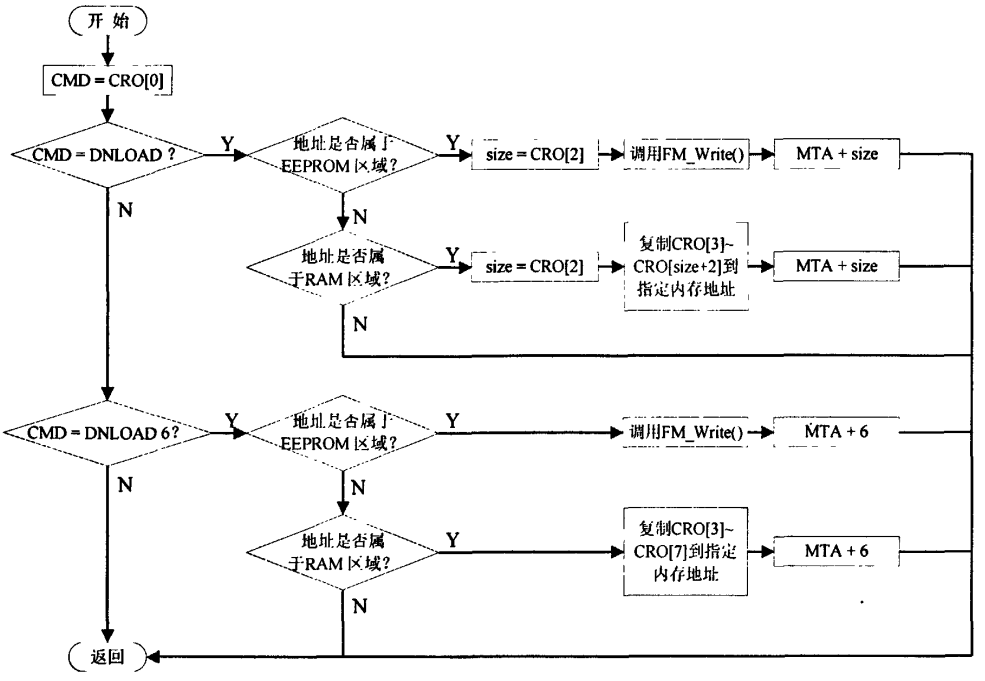


图 4-7 ccpWriteMTA( )函数流程图

Figure 4-7 Flow chart of ccpWriteMTA ( )

4.3 基于 CCP 协议应用

根据前一章节对电池管理系统所要完成功能的描述，可以从中看出该系统需要完成的任务比较多：模块电压的测量、模块温度、总电压的测量、动力电池上流过的电流、电池包绝缘性能检测、电池荷电状态（SOC）在线估算、高压控制、握手机制、串口通讯及整车 CAN 通讯，另外还有一项辅助功能，测量车载 DC/DC 变换器的输入电流（该电流也是由动力电池提供的）。传统的应用程序的开发是采用前后台或超循环系统(Super-Loops)的编程模式，对于这种前后台的编程模式难以在多个复杂任务之间进行协调管理，开发起来比较困难，不利于以后系统程序的开发与维护。整个应用程序就是一个大的循环程序，在循环中调用函数完成相应的操作，这部分称为后台行为(background)，利用中断服务程序(ISR)来处理异步事件，这部分称为前台行为(foreground)。前后台系统中后台行为的响应时间是与整个循环的执行时间有关的，对于某一个特定部分的准确时间是不能确定的，所以它不能保证实时性。为了保证系统的实时性，系统一般需要采用一个实时操作系统内核[26]。

嵌入式实时操作系统（RTOS）在目前的嵌入式应用中使用越来越多，尤其在功能复杂、系统庞大的应用中显得愈来愈重要。这主要是嵌入式操作系统自身特

点所决定的：首先，嵌入式实时操作系统提高了系统的实时性和可靠性。其次，提高了开发效率，缩短了开发周期。在嵌入式实时操作系统环境下，开发一个复杂的应用程序，通常可以按照软件工程中的解耦原则，将整个程序分解为多个任务模块。每个任务模块的调试、修改几乎不影响其他模块。另外，嵌入式实时操作系统可以充分的发挥 CPU 的应用潜能[27]。

因此，本课题在开发电池管理系统应用程序的时候，将实时操作系统 μC/OS-II 引入进来，采用的版本是 V2.52。

4.3.1 μ C/OS-II 的特点

(1)、源代码公开

它的源代码公开，可以从 μC/OS 官方网站上获取全部源码以及在各种体系结构上的移植范例。通过和其它源码公开的商业实时内核相比较，μC/OS-II 的源码清晰易读、结构协调，注释详尽有序，这也是 μC/OS-II 能成功的得到广泛应用的一个原因。

(2)、可移植

绝大部分 μC/OS-II 的源码是用移植性很强的 ANSI C 写的。和微处理器硬件相关的那部分是用汇编语言写的。汇编语言写的部分已经压到最低限度，使得 μC/OS-II 便于移植到其他微处理器上。条件是，只要该微处理器有堆栈指针，有 CPU 内部寄存器入栈、出栈指令。另外，使用的 C 编译器必须支持内嵌汇编或者该 C 语言可扩展、可连接汇编模块，使得关中断、开中断能在 C 语言程序中实现。μC/OS-II 可以在绝大多数 8 位、16 位、32 位以至 64 位微处理器、微控制器、数字信号处理器(DSP)上运行。

(3)、可裁剪

可以只使用 μC/OS-II 中应用程序需要的某些系统服务。也就是说某方案设计可以只使用很少几个 μC/OS-II 系统调用，而另外一个方案则可以使用几乎所有 μC/OS-II 的功能。这样可以定制方案设计中的 μC/OS-II 所需的存储空间(RAM 和 ROM)，这种可裁剪性是利用条件编译实现的。只要在用户的应用程序中定义哪些 μC/OS-II 中的功能是应用程序需要运行的就可以了。程序和数据两部分的存储用量已被尽可能的压低。

(4)、多任务

μC/OS-II 可以管理 64 个任务（最新的版本可以管理 256 个任务），其中应用程序最多可以有 56 个任务，另外 8 个任务是给系统预留的。每个任务分配的优先级必须是不同的，这意味着 μC/OS-II 不支持时间片轮转调度法(Round-robin

Scheduling)，时间片轮转调度法可用于调度相同优先级的任务。

(5)、任务执行时间可确定

全部 μC/OS-II 的函数调用与服务的执行时间具有其可确定性。也就是说，全部 μC/OS-II 的函数调用与服务的执行时间是可知的。进而言之，μC/OS-II 系统服务的执行时间不依赖于应用程序任务的多少。

(6)、任务栈

每个任务都有自己单独的栈空间，μC/OS-II 允许每个任务有不同的栈空间。以便灵活得控制应用程序对 RAM 的需求量。通过使用 μC/OS-II 的栈空间校验函数，还可以确定每个任务到底需要多少栈空间[28]。

4.3.2 μ C/OS-II 操作系统结构

μC/OS-II 是一个基于优先级调度的抢占式的实时内核，并在这个内核之上提供最基本的系统服务，如信号量，邮箱，消息队列，内存管理，中断管理等[29][30]。

4.3.2.1 任务管理

μC/OS-II 中最多可以支持 64 个任务，分别对应优先级 0 ~ 63，其中 0 为最高优先级，63 为最低级，系统保留了 4 个最高优先级的任务和 4 个最低优先级的任务，所以用户可以使用的任务数有 56 个。μC/OS-II 提供了任务管理的各种函数调用，包括创建任务，删除任务，改变任务的优先级，任务挂起和恢复等。系统初始化时会自动产生两个任务：一个是空闲任务，它的优先级最低，该任务仅给一个整形变量做累加运算；另一个是系统统计任务，它的优先级为次低，该任务负责统计当前 CPU 的利用率，该任务也可以通过系统配置而取消。

4.3.2.2 时间管理

μC/OS-II 的时间管理是通过定时中断来实现的，该定时中断一般为 10 毫秒到 100 毫秒发生一次，时间频率取决于用户对硬件系统的定时器编程来实现。中断发生的时间间隔是固定不变的，该中断也成为一个时钟节拍。μC/OS-II 要求用户在定时中断的服务程序中，调用系统提供的与时钟节拍相关的系统函数，例如中断级的任务切换函数，获取系统时间函数。

4.3.2.3 内存管理

在 ANSI C 中是使用 malloc 和 free 两个函数来动态分配和释放内存。但在嵌入式实时系统中，多次这样的操作会导致内存碎片，且由于内存管理算法的原因，malloc 和 free 的执行时间也是不确定。μC/OS-II 中把连续的大块内存按分区管理。每个分区中包含整数个大小相同的内存块，但不同分区之间的内存块大小可以不同。用户需要动态分配内存时，系统选择一个适当的分区，按块来分配内存。释

放内存时将该块放回它以前所属的分区，这样能有效解决碎片问题，同时执行时间也是固定的。

4.3.2.4 任务间通信与同步

对一个多任务的操作系统来说，任务间的通信和同步是必不可少的。μC/OS-II 中提供了 4 种同步对象，分别是信号量，邮箱，消息队列和事件标志。所有这些同步对象都有创建，等待，发送，查询等接口用于实现任务间的通信和同步。

4.3.2.5 任务调度

μC/OS-II 采用的是可剥夺型实时多任务内核。可剥夺型的实时内核在任何时候都运行处于任务就绪态的最高优先级的任务。μC/OS-II 的任务调度是完全基于任务优先级的抢占式调度，也就是高优先级的任务一旦处于就绪状态，则立即抢占正在运行的低优先级任务的处理器资源。为了简化系统设计，μC/OS-II 规定所有任务的优先级不同，因为任务的优先级也同时唯一标志了该任务本身。

4.3.2.6 μC/OS-II 的组成部分

μC/OS-II 可以大致分成核心、任务处理、时间处理、任务同步与通信、CPU 的移植等 5 个部分。

内核部分（OSCore.c）是操作系统处理的核心，包括操作系统的初始化、操作系统的运行控制、中断进出的前导、时钟节拍、任务调度、事件处理等多部分。能够维持系统基本工作的部分都在这里。

任务处理部分(OSTask.c)，任务处理部分中的内容都是与任务的操作密切相关的。包括任务的建立、删除、挂起、恢复等等。因为 μC/OS-II 是以任务为基本单位调度的，所以这部分内容也相当重要。

时钟部分(OSTime.c)，μC/OS-II 中的最小时钟单位是 time tick(时钟节拍)。任务的调度也会发生在时钟节拍中断中，任务延时等操作是在这里完成的。

任务同步和通信部分包括信号量、邮箱、邮箱队列、事件标志等部分，主要用于任务间的互相联系和对共享资源的访问。

与 CPU 的接口部分指 μC/OS-II 针对所使用的 CPU 的移植部分。由于 μC/OS-II 是一个通用性的操作系统，所以对于关键问题上的实现，还是需要根据具体 CPU 的具体内容和要求作相应的修改。这部分内容由于牵涉到 SP 等系统指针，所以通常用汇编语言编写。主要包括中断级任务切换的底层实现、任务级任务切换的底层实现、时钟节拍的产生和处理、中断的相关处理部分等内容。

4.3.3 基于 μ C/OS-II 的 CCP 协议应用

4.3.3.1 μC/OS-II 的移植

所谓移植，就是使一个实时内核能在某个微处理器或微控制器上运行。为了方便移植，大部分的 μC/OS-II 代码是用 C 语言写的，但仍需要用 C 和汇编语言写一些与处理器相关的代码，这是因为 μC/OS-II 在读写处理器寄存器时只能通过汇编语言来实现。由于 μC/OS-II 在设计时就已经充分考虑了可移植性，所以 μC/OS-II 的移植相对是比较容易的。要使 μC/OS-II 正常运行，处理器必须满足以下要求[29]：

(1)、处理器的 C 编译器能产生可重入代码；

(2)、用 C 语言就可以打开和关闭中断；

(3)、处理器支持中断，并且能产生定时中断(通常在 10 至 100Hz 之间)；

(4)、处理器支持能够容纳一定量数据(可能是几千字节)的硬件堆栈；

(5)、处理器有将堆栈指针和其它 CPU 寄存器读出和存储到堆栈或内存中的指令。

电池管理系统采用的控制器是飞思卡尔的 MC9S12DT128，软件开发环境是 Metrowerks 公司的 CodeWarrior 3.1。系统提供的能允许同时被多个任务所调用，而不会通过函数中变量的藕合而引起任务之间的干扰的函数，叫做可重入函数。CodeWarrior 支持高级语言，可通过在公共函数中只使用局部变量来产生可重入函数，因为局部变量存储在任务的堆栈中，所以可以保证不同的任务在调用同一个函数时不会发生冲突。若必须使用全局变量，则需要对使用的全局变量做必要的保护。CodeWarrior 3.1 编译器能够满足移植条件的前两个。μC/OS-II 由于可以定制内核大小、可裁减，用户可以只选用对其应用程序有用的那一部分，内核目标代码可以裁减小到 1.5KB，大到 10KB 以上，同时也会占用一定的 RAM 空间。MC9S12DT128 单片机有 8KB 片内 RAM，128KB 片内 Flash，能够满足运行实时内核的要求。该单片机具有丰富的中断源，含有将堆栈指针和其它 CPU 寄存器读出和存储到堆栈或内存中的指令，满足上述的移植条件。

要实现 μC/OS-II 向 CPU12 的移植，需要做两方面的工作：一是重新定义内核的大小和功能；二是为内核编写与硬件相关的代码。μC/OS-II 的文件结构如图 4-8 所示。

可以看到，μC/OS-II 与 CPU 类型无关的 C 代码文件 UCOS_II.C 包括很多文件，它们是 μC/OS-II 的内核和很多功能函数，其中前面 3 个文件是实时内核、任务管理和时钟节拍，这 3 个文件是一定要用的。后面 6 个功能函数用于任务间通信，应用程序中可能只用到其中的几个，不用的可以不包含进去，以避免编译时生成没用的代码。这部分代码与 CPU 类型无关，在移植时，这些文件可不必改动。配置文件 OS_CFG.C 需要根据应用配置，主要作用是确定 μC/OS-II 提供的系统功能函数中，应用程序用哪些、不用哪些，这个文件移植时根据需要进行修改。

与 CPU 类型有关的代码文件主要有 3 个：OS_CPU.H、OS_CPU_A.ASM 和

OS_CPUC.C。OS_CPU.H 文件定义用于特定 CPU 的数据类型、定义相关的宏。OS_CPU_A.ASM 是用汇编语言写的与硬件相关的代码，OS_CPUC.C 文件是用 C 语言写的与硬件相关的代码。由于移植使用的 C 交叉编译工具在 C 代码中可以插入汇编语句，在移植中，将这 2 个文件合成为 1 个文件。

产生时钟节拍的定时中断来自单片机内部但并非来自 CPU12 内部，可以用实时时钟产生定时中断，也可以用单片机片内的外设模块定时器来产生定时中断，这部分代码显然与硬件相关，移植时要自己写[29]。

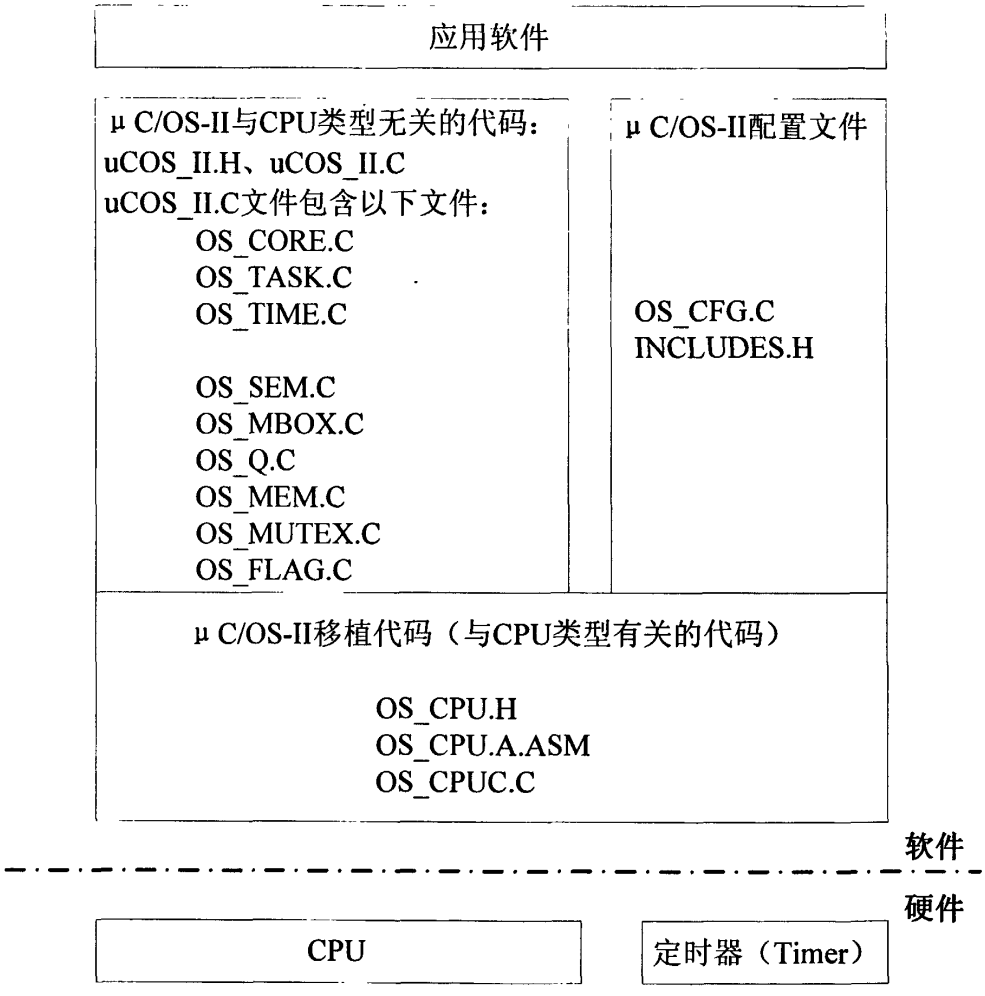


图 4-8 μC/OS-II 的文件结构

Figure 4-8 File structure of μC/OS-II

移植第一步：重新定义内核的大小和功能：

公共头文件 INCLUDES.H，这个文件会被所有的 C 源程序引用。里面包含了一些头文件，其中最主要的是三个头文件“os_cpu.h”、“os_cfg.h”、“ucos_ii.h”需要根据应用修改的是文件 OS_CFG.H，这个文件用于配置内核的属性。这里面

可以对系统服务进行裁减，不编译不需要的代码，使系统精简并且减少存储器用量。OS_CPU.H 用于设置与微控制器的 CPU 核心相关的属性，包括各种数据类型、堆栈宽度和增长方向以及任务调度函数 OS_TASK_SW( )的宏定义。接下来定义了堆栈宽度和增长方向。绝大多数的微处理器和微控制器的堆栈是从上往下长的。但是某些处理器是用另外一种方式工作的。μC/OS-II 被设计成两种情况都可以处理，只要在结构常量 OS_STK_GROWTH 中指定堆栈的生长方式就可以了。其中置 OS_STK_GROWTH 为 0 表示堆栈从下往上长；置 OS_STK_GROWTH 为 1 表示堆栈从上往下长。MC9S12DT128 选用的是第二种堆栈方式。OS_TASK_SW() 是在 μC/OS-II 从低优先级任务切换到最高优先级任务时被调用的。OS_TASK_SW() 总是在任务级代码中被调用的。任务切换只是简单的将处理器寄存器保存到将被挂起的任务的堆栈中，并且将处于就绪态优先级最高的任务从堆栈中恢复出来。在 μC/OS-II 中，处于就绪状态的任务的堆栈结构看起来就像刚发生过中断并将所有的寄存器保存到堆栈中的情形一样。换句话说，μC/OS-II 要运行处于就绪状态的任务必须要做的事就是将所有处理器寄存器从任务堆栈中恢复出来，并且执行中断的返回。为了切换任务可以通过执行 OS_TASK_SW()来产生中断。大部分的处理器会提供软中断或是陷阱(TRAP)指令来完成这个功能，它们处理函数的向量地址必须指向汇编语言函数 OSCtxSw()。OS_CPU.H 中还需要给出 2 个关于临界段函数的宏定义，它们的源代码如下：

```
#if        OS_CRITICAL_METHOD = = 3
#define    OS_ENTER_CRITICAL()   {asm tpa;  asm sei;  asm staa cpu_sr}
#define    OS_EXIT_CRITICAL()    {asm ldaa cpu_sr;   asm tap}
#endif
```

OS_ENTER_CRITICAL()是进入临界段宏定义，在要进入临界段时先保存进入临界段代码前的 CCR 寄存器，即保存中断标志位的状态，在调用 OS_EXIT_CRITICAL()退出临界段时，将 CCR 寄存器还原至临界段进入之前的状态。二者必须配对使用。

接下来就是编写与硬件相关的代码：

由于使用的是 CodeWarrior 交叉编译器，可以将图 4-8 所示的 OS_CPUC.C 和 OS_CPU_A.ASM 放在同一个文件中。涉及到的主要几个函数是：

(1)、时钟中断服务子程序 OSTickISR( )；

(2)、任务堆栈初始化函数 OSTaskStkInit( )；

(3)、让优先级最高的就绪态任务开始运行函数 OSStartHighRdy( )；

(4)、中断级任务切换函数 OSIntCtxSw( )；

(5)、任务级任务切换函数 OSCtxSw( )；

(6)、相关接口函数。

这一部分涉及到最重要的就是 MC9S12DT128 的任务堆栈管理。MC9S12DT128 在发生中断时，会自动地将程序计数器 PC，索引寄存器 Y、X，累加寄存器 A、B，条件码寄存器压入堆栈，其堆栈结构如图 4-9 所示。

MC9S12DT128 内有 128KB 的 Flash，作为 16 位的 CPU，它的寻址空间为 64K，为了能够访问者 128K 的 Flash，S12 采用了页面管理的机制，将地址范围在 0x8000~0xBFFF 的空间划分成不同的页，设置一个专门的页寄存器 PPAGE 纪录当前程序使用的是那一页。而这个页寄存器不同上述其它几个寄存器，在发生中断时不会自动入栈，因此需要通过额外的堆栈操作来模拟 PPAGE 入栈，方法是在自动完成相关寄存器的入栈以后紧接着通过压栈操作将 PPAGE 放入自动入栈的寄存器之后；当中断退出时，需要通过出栈操作将 PPAGE 先弹出，然后执行中断返回将自动入栈的寄存器依次弹出。从而人为的实现 PPAGE 的自动入栈操作。最终的整个堆栈的结构如图 4-9 所示。时钟中断服务子程序 OSTickISR( )除了调用操作系统时钟节拍处理函数，还需要注意的就是对页寄存器的堆栈操作。

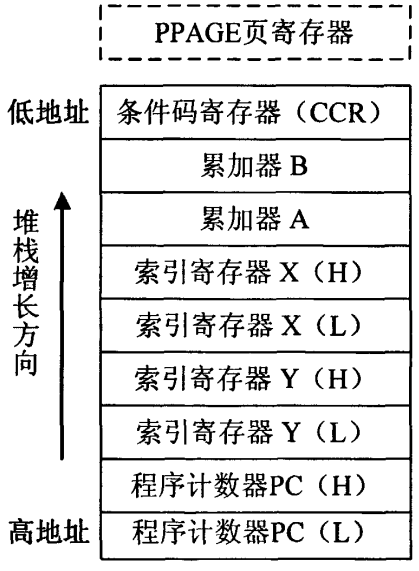


图 4-9 中断时寄存器的入栈结构图

Figure 4-9 Stack structure for interruption

OSTaskStkInit( )是用来在建立任务时初始化任务堆栈的函数。由于任务的开始运行不是通过被某一个程序调用而开始的，而是内核做任务调度时，让这个任务运行才开始运行。如果任务调度让某一个任务开始运行，它会从该任务的堆栈中获取该任务上一次运行的 PC 值和各种寄存器信息，通过中断返回继续执行，就像这个任务刚刚发生过中断一样。OSTaskStkInit( )就是为了在内核让该任务第一次开始运行时找到任务的入口，因此它需要做的就是将任务的入口地址等信息填入到

该任务的堆栈中，模拟该任务发生过中断一样将 CPU 寄存器入栈。任务堆栈初始化时堆栈中各寄存器在堆栈中的位置必须与寄存器的入栈结构一致，任务堆栈的初始化函数的源代码如下：

```
void *OSTaskStkInit (void (*task)(void *pd), void *pdata, void *ptos, INT16U opt)
{
    INT16U   *stk;
    stk   =   INT16U *)ptos;                    // 装载堆栈指针
    *--stk = opt;                               // 保留
    *--stk = (INT16U)((INT32U)task>>8);         // PC 寄存器
    *--stk = (INT16U)((INT32U)task>>8);         // PC 寄存器
    *--stk = (INT16U)(0x0);                     // Y 寄存器
    *--stk = (INT16U)(0x0);                     // X 寄存器
     ((INT8U *)stk)--;                          // A 寄存器
     *(INT8U *)stk = (INT8U)(((INT16U)pdata)>>8);
     ((INT8U *)stk)--;                          // B 寄存器
     *(INT8U *)stk = (INT8U)(pdata);
     ((INT8U *)stk)--;                          // CCR 寄存器
     *(INT8U *)stk = (INT8U)(0x00);
     ((INT8U *)stk)--;                          // PPAGE 寄存器
     *(INT8U *)stk = (INT8U)task;
     return ((void *)stk);                      //返回栈顶地址
}
```

因为 MC9S12DT128 工作在分页方式下时，它的任务起始地址一共由 3 个字节组成，其中低 8 位是它的分页地址，高 16 位是入口地址，所以可以通过上述方法从任务函数名来提取任务的入口地址和页地址。当然，也可以根据 C 语言函数调用规则从堆栈中提取任务入口地址和页地址，而不是简单的利用任务函数名来提取。

操作系统完成初始化以后，开始执行系统内优先级最高的就绪任务，是通过调用 OSStartHighRdy( )函数实现的，它将 CPU 的堆栈指针 SP 的值，改成优先级最高的就绪态任务的堆栈指针的值，该值是其任务控制块的第一个参数；然后修改内核的状态，由非运行态“0”改为运行态“1”；恢复任务代码所在的寄存器页面的值以换入即将运行的任务代码，换出被挂起的任务的代码；然后执行中断返回指令 RTI 以开始运行这个任务。前面是接口函数的调用，用户可以插入一些自定义的代码。

```
void    OSStartHighRdy(void)
{
   OSTaskSwHook();                  // 接口函数
   asm
   {
```

```
        ldx     OSTCBCur
        lds     0，x            // 将就绪态任务堆栈的栈顶赋给 SP
        ldaa    OSRunning
        inca
        staa    OSRunning       // OSRunning = 1
        pula
        staa    $30             // 从堆栈中提取 PPAGE
        nop
        rti
    }
}
```

OSCtxSw( )和 OSIntCtxSw( )是内核用来进行任务切换的，分别对应任务级的任务切换和中断级的任务切换。如果当前运行的程序使一个比它优先级高的任务进入了就绪态，系统做任务级的任务切换，将使当前运行的任务挂起，让高优先级的任务运行。任务级的切换是通过执行软中断指令来实现的。OSCtxSw( )实际上就是软中断服务子程序，软中断服务子程序的向量地址必须指向 OSCtxSw( )，MC9S12DT128 的软中断服务子程序的向量地址为$FFF6。OSCtxSw( )会依次完成以下几个步骤：把被挂起任务的断点指针保存到任务堆栈中；把 CPU 通用寄存器的内容保存到任务堆栈中；把即 PPAGE 寄存器的值保存到任务堆栈中；把被挂起任务的任务堆栈指针当前值保存到该任务的任务控制块；任务切换接口函数；获得待运行任务的任务控制块；使 CPU 通过任务控制块获得待运行任务的任务堆栈指针；恢复待运行任务的 PPAGE 寄存器；通过执行中断返回指令把待运行任务堆栈中通用寄存器的内容恢复到 CPU 的通用寄存器；使 CPU 获得待运行任务的断点指针，新任务开始恢复运行。OSCtxSw( )的源程序如下：

```
void    OSCtxSw(void)
{
    asm
    {
        ldaa    $30
        psha                        //保存页寄存器到任务堆栈
        ldx     OSTCBCur
        sts     0，x                // 保存任务的栈顶到任务块中
    }
    OSTaskSwHook();                 // 接口函数
    OSTCBCur = OSTCBHighRdy;        // 获取最高就绪态任务块
    OSPrioCur = OSPrioHighRdy;      // 获取最高就绪态任务的优先级
    asm
    {
        ldx     OSTCBCur
```

```
        lds     0，x                    // 获取最高就绪态任务的堆栈指针
        pula
        staa    $30                     // 恢复最高就绪态任务的页寄存器
        nop
        rti                             // 执行中断返回，让该任务恢复运行
    }
}
```

中断服务子程序在执行完中断服务以后会调用 OSIntExit()，如果 OSIntExit()发现中断激活了更高优先级的任务，则调用中断级的任务切换函数 OSIntCtxSw()。由于中断时包括 PPAGE 寄存器的所有 CPU 寄存器都保存过了，故不需要再保存。所以 OSIntCtxSw()函数的代码实际上是任务级任务切换函数的后半部分。它的源代码如下：

```
void    OSIntCtxSw(void)
{
    OSTaskSwHook();                     // 接口函数
    OSTCBCur = OSTCBHighRdy;            // 获取最高就绪态任务块
    OSPrioCur = OSPrioHighRdy;          // 获取最高就绪态任务的优先级
    asm
    {
        ldx     OSTCBCur
        lds     0，x                    // 获取最高就绪态任务的堆栈指针
        pula
        staa    $30                     // 恢复最高就绪态任务的页寄存器
        nop
        rti                             // 执行中断返回，让该任务恢复运行
    }
}
```

4.3.3.2 系统应用程序的设计

在完成了实时操作系统的移植以后，需要根据电池管理系统设计的功能，对系统所要完成的功能进行任务划分，并对每个任务根据其实时性要求和有效利用 RAM 的原则分配一定的优先级。

从第 3 章的介绍可知，可以将系统任务这样划分：模块电压测量任务、电池温度测量任务、电池包实时总电压和电流测量任务、动力电池荷电状态(SOC)在线估算任务、电池包绝缘性能在线检测任务、车载 DC/DC 变换器输入电流数据处理任务、握手机制实现任务、标定 CAN 通讯处理任务以及 RS232 串口通讯数据处理任务。高压控制是由整车 CAN 通讯的命令交互来控制的，整车 CAN 通讯的帧数据都是定时发送的，间隔时间为 10ms 和 1s 两种，因此系统设计了两个定时器，一个为系统提供时钟节拍，另一个为整车 CAN 数据发送提供时间基准。因此系统

还需要几个中断处理函数：系统时钟节拍中断，CAN 帧数据发送定时中断、整车 CAN 通讯发送与接收中断、RS232 串口发送与接收中断；另外，标定 CAN 的数据接收也是采用中断的方式进行的。DC/DC 变换器的输入电流是通过霍尔传感器测量的，它的输出是一定频率范围的脉宽调制（PWM）信号，因此系统还需要脉冲边沿捕捉中断。

任务划分完毕以后，还必须对其进行合理的优先级分配。任务优先级分配一般根据任务的实时性要求进行划分，并同时需要综合系统的资源要求进行合理安排，故将优先级是按顺序分配，这样可以减少在分配任务块的时候造成内存资源的浪费。最后的任务划分和优先级划分结果如表 4-1 所示。

表 4-1 系统任务划分及任务优先级分配

Table 4-1 Task partitioning and Priority of system

任务名	优先级	任务描述
Task_RS232	4	系统串口通讯任务
Task_CCP	5	CCP 标定任务
Task_Handshake	6	实现握手机制
Task_SumvAndCurr	7	实时采集电池包的总电压和电流
Task_Isolation	8	在线测量电池包的绝缘性能
Task_DC_Curr	9	实时测量 DC/DC 变换器的输入电流
Task_SOC	10	在线估算电池包的荷电状态
Task_ModuleVoltage	11	采集电池包模块电池的电压
Task_Temperature	12	采集电池包的温度

任务程序的结构本质上是一个死循环的程序代码。应用程序的多任务设计时还需考虑各个任务间的数据交换，同时还要处理中断与任务之间的数据交换的问题，也就是说需要完成任务与任务之间、任务与中断处理函数之间的数据通讯。实现任务间通讯最简便的办法是使用共享数据结构，另外还可以使用消息机制，μC/OS-II 的消息机制提供信号量、邮箱服务、消息队列和事件标志。

系统在任务设计时，用到的任务通讯机制主要包括三个方面：共享数据结构、信号量及邮箱服务。每一个任务对共享数据进行访问时，都采用了临界段的保护措施对共享数据进行保护；串口接受中断与串口数据处理采用了信号量的通讯机制实现；DC/DC 输入电流脉冲捕捉与电流的处理同样也采用了信号量的通讯机制；标定 CAN 的接收中断与 CCP 命令处理任务采用了邮箱服务的机制，其执行流程图如图 4-10 所示。

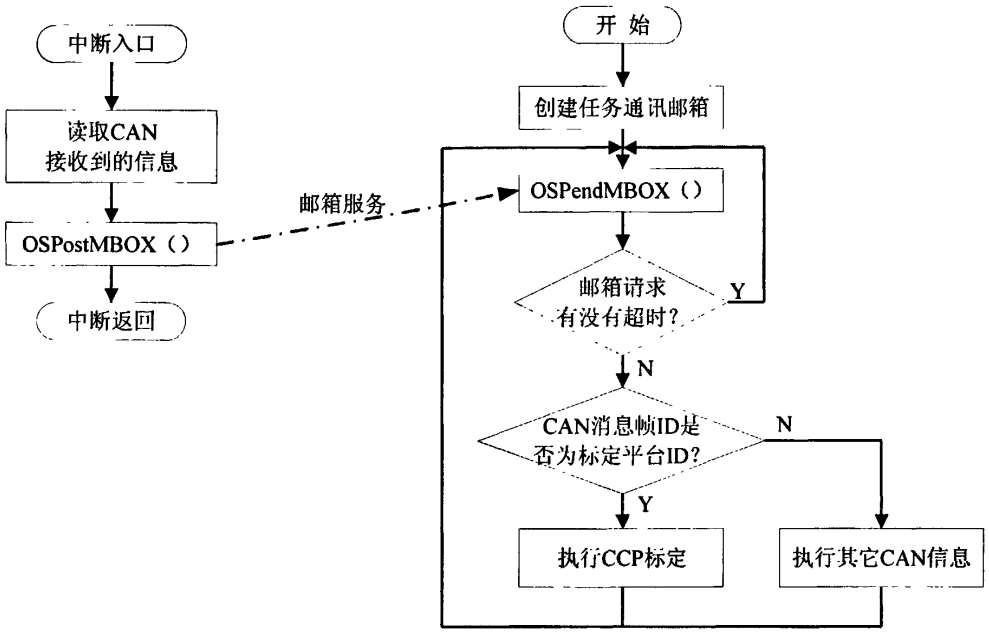


图 4-10 标定 CAN 接收中断与 CCP 命令处理任务的邮箱服务通讯流程图
Figure 4-10 Flow chart of CAN receive interruption and CCP command progress using mail box

4.4 标定平台软件开发

4.4.1 标定平台软件的功能分析

标定平台作为整个标定系统的主导者，它控制着整个标定系统的数据流向、数据处理和显示。标定平台软件需要支持以下几个功能：

(1)、界面操作指令处理：有着良好的人机交互接口，方便的操作控制界面；

(2)、数据通讯：能够完成 CCP 协议标定命令的帧数据的打包和解包；

(3)、数据显示：为标定人员提供所有数据的信息显示功能；

(4)、任务管理：任务的建立、修改和删除，包括添加/删除需要监测和标定的参数对象，修改参数和窗口的显示属性；

(5)、试验数据管理：接收数据以及数据的存储；

(6)、参数值修改：标定人员在标定窗口上的参数值修改操作；

(7)、命令数据发送：向从机发送标定人员的控制命令。

4.4.2 标定平台软件的动态行为建模

系统的动态行为模型有交互作用图（时序图和协作图）、状态图、动态图描

述。本小节对系统主要的三个部分的动态行为进行了交互作用图描述。

1、通讯设备启动与通讯链路配置：

标定平台软件需要通过 USB-CAN 设备对从机进行数据通讯，因此系统需要对 CAN 卡设备进行控制。同时标定平台还必须明确自己和从机的 ID 号及从机的分站号，以建立整个标定链路的通讯。它的交互作用描述图如图 4-11 所示。

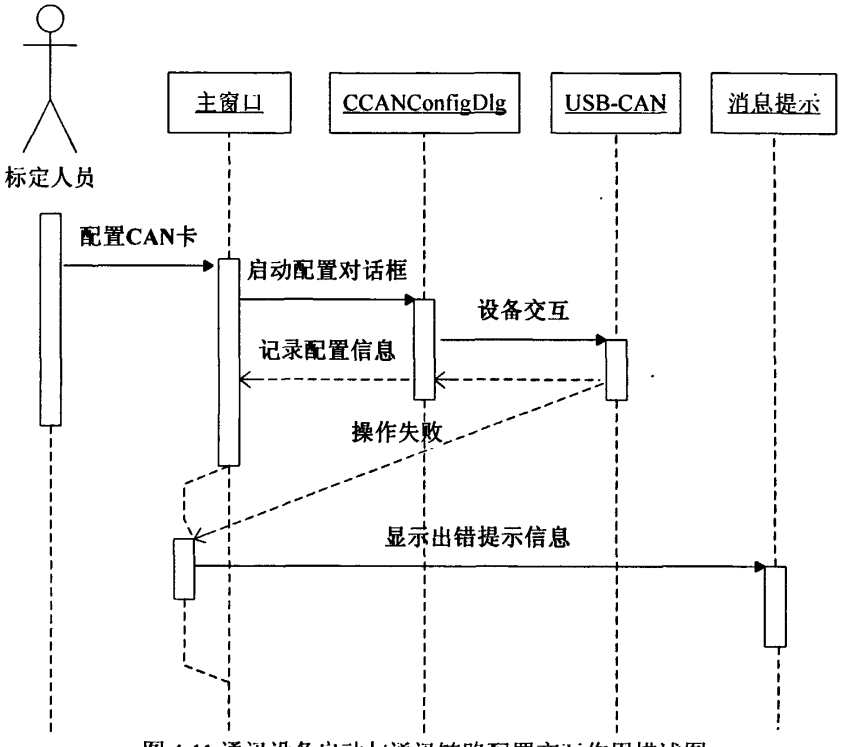


图 4-11 通讯设备启动与通讯链路配置交互作用描述图

Figure 4-11 Interaction description of communication device start and communication link configuration

2、DAQ 模式数据监控

CCP 协议支持的数据监控模式有两种，一种是 polling 模式，另外一种是 DAQ 模式。DAQ 模式的实现比较复杂，它的交互作用描述图如图 4-12 所示。标定平台通过与从机建立逻辑连接之后，对需要监控的参数进行 DAQ 信息的初始化，然后就可以对 DAQ 模式进行启停控制。在 DAQ 模式，标定平台还需将接收到的 DAQ 数据进行处理分析，并将其显示出来。需要数据保存的时候，可以将监控到的数据保存到数据库当中。

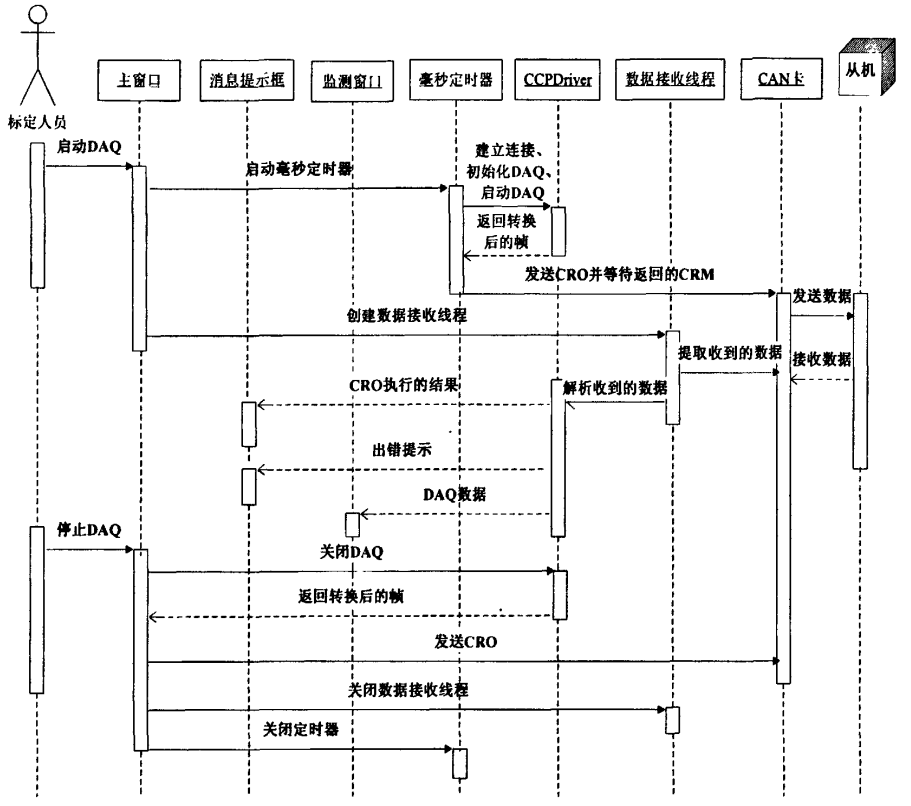


图 4-12　DAQ 模式交互作用描述图

Figure 4-12 Interaction description of DAQ mode

3、数据标定

图 4-13 是在线数据标定的交互作用描述图。进行数据标定时标定平台必须先同从机建立逻辑连接，然后通过标定界面对需要标定的参数进行标定操作。

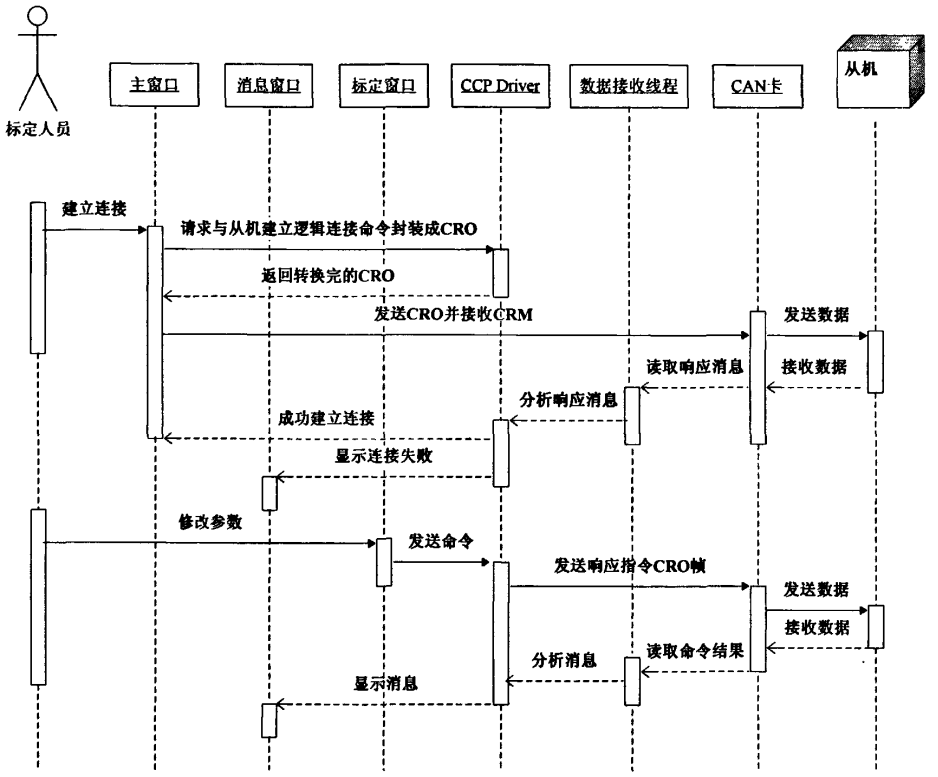


图 4-13 标定模式交互作用描述图

Figure 4-13 Interaction description of calibration mode

4.4.3 标定平台软件的编程实现

Windows 作为一个提供功能强大的应用程序接口(API)编程的操作系统，方便了许多程序员的开发工作。但由于 API 函数实在太多了，而且名称很乱，如果从零开始构架一个所需显示的窗口就需要上百行的代码。因此传统的 Win32 开发(即直接使用 Windows 的 API 函数)，对于程序员来说是比较困难的。相对的，MFC 是面向对象程序设计与应用程序框架的完美结合，它将传统的 API 进行了分类封装，并且预先为程序员创建了几种程序框架。MFC 使得软件的开发过程变得更为容易，也使得软件的程序结构更为清晰[31]。因此，本文最终选择了基于 MFC 的 SDI 框架来开发标定平台软件，本节将对程序主体流程以及通信流程控制，关键技术点进行说明。

4.4.3.1 程序主体

Windows 属于消息驱动，操作系统的所有功能都由消息来触发，并依靠对消息的响应和处理来完成。MFC 同样提供了一套支持 Windows 消息机制的技术：消息映射和命令传递。

Windows 进行资源分配的最小单位是进程，进行 CPU 时间分配的最小单位是线程。进程是应用程序的运行实例，拥有自己的地址空间。每个进程拥有一个主线程，同时还可以建立其他的线程，进程中的线程共享进程的资源，处于并行执行状态，这就是多线程的基本概念。本文在实现标定平台软件时就采用了多线程技术[32]。

软件运行最初，程序只有系统自己创建的一个主线程，相当于 C 程序中 main 函数，主线程依靠消息驱动维持正常的运行，它不断从消息队列中获取来自用户界面操作等消息，然后进行相应的处理，循环反复执行直至接收到退出应用程序消息。标定人员点击界面上的菜单、按钮等控件，发出不同的消息，然后软件完成这些消息的处理响应。软件主要涉及到对 CAN 卡的操作，监控参数的配置，DAQ 操作，监控数据的处理、显示、保存，标定操作。在 DAQ 模式下，系统创建了一个专门用来接收 DAQ 数据的接收线程，如果该模式下需要数据保存功能，系统能够还另外创建了一个数据收集线程，防止接收线程因数据量过大导致接收缓存溢出。主线程，数据收集线程都共用了数据接收线程的缓冲区，为此数据接收线程开辟了两个缓冲区交替使用，防止主线程在处理缓冲区数据时而阻塞了接收线程，导致数据的丢失。线程之间会进行共享资源的访问和操作，所以需要利用 MFC 的线程同步函数来实现资源的共享访问。当退出 DAQ 模式时，数据线程和数据收集线程会自动退出并结束自己的生命期。软件操作界面如图 4-14 所示。

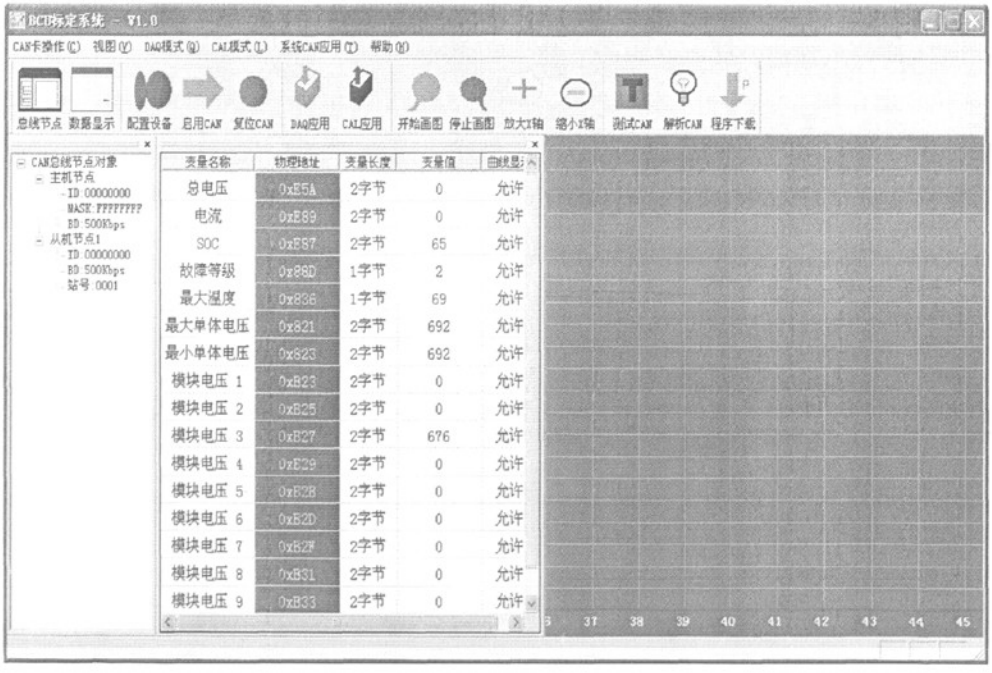


图 4-14 标定平台软件操作界面
Figure 4-14 Software interface of calibration platform

4.4.3.2 通信流程控制

1、监控通信

(1)、开始通信

首先执行通信初始化操作，包括：与从机建立逻辑连接、对从机控制器中的DAQ 列表进行配置等。初始化完成后，开启定时器并创建数据接收线程，如果需要数据保存就创建数据收集线程。图 4-15 说明了这个操作流程，并对整个数据收发命令作了描述。

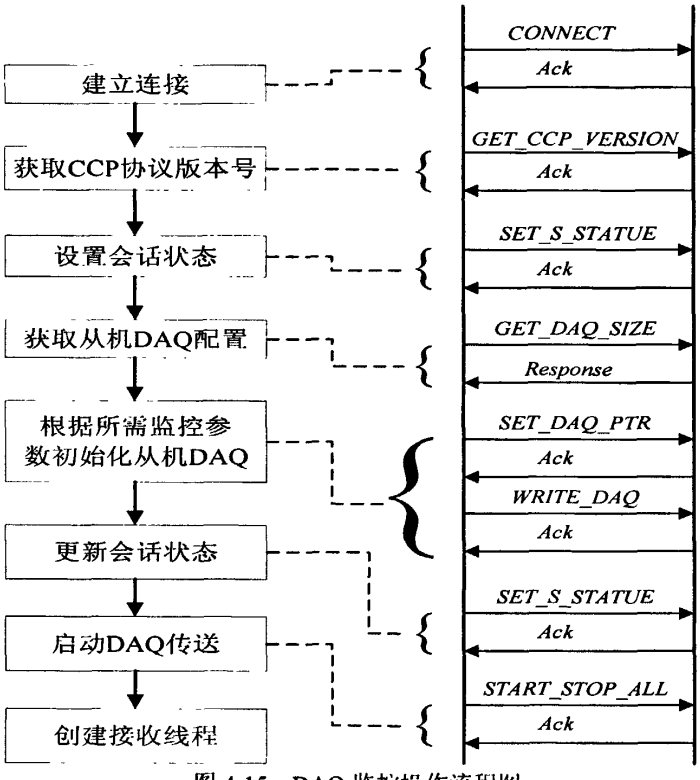


图 4-15 DAQ 监控操作流程图

Figure 4-15 DAQ operation flow chart

(2)、停止监控

停止监控的操作流程比较简单，如图 4-16 所示：

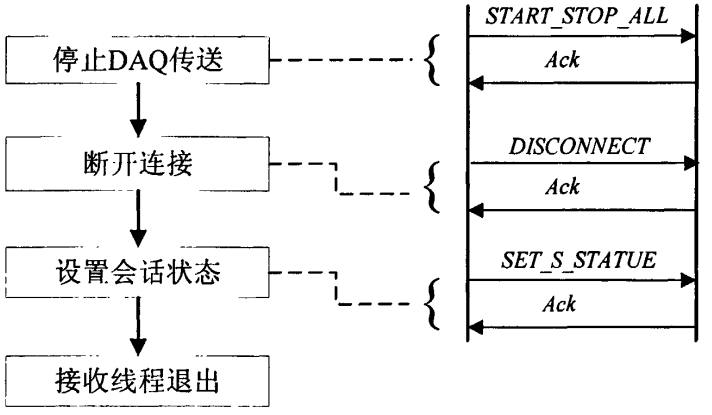


图 4-16　DAQ 停止监控流程图
Figure 4-16 DAQ stop operation flow chart

2、标定通信

此时，已与 ECU 建立了连接，标定人员可以实时监测到参数值，并根据监测结果进行相应的标定操作，修改后的参数值将直接下载到标定内存。标定过程中所有控制参数的改动，都是对 RAM 区的修改。试验结束时，由标定人员发出写固化命令，此时再将标定内存的数据回写到铁电存储器中进行固化。操作流程图如图 4-17 所示。

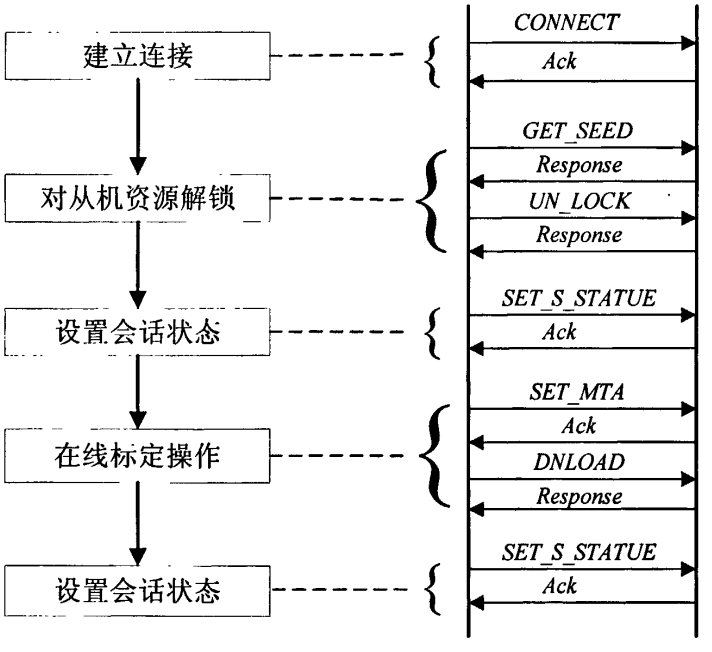


图 4-17　标定通信流程图
Figure 4-17 Flow chart of Calibration operation

4.5 本章小结

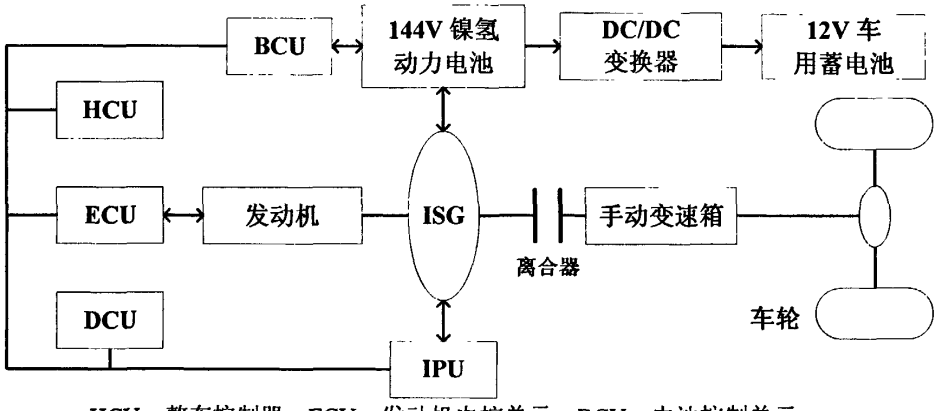
本章主要电池管理系统 CAN 驱动程序和 CCP 驱动程序的设计，结合系统自身的应用需要将实时操作系统 μC/OS-II 移植到系统中，并完成基于它们的电池管理标定系统的应用程序开发。最后，还将标定平台上位机软件的设计进行介绍。

5 标定试验

5.1 整车控制系统

5.1.1 整车控制系统

如图 5-1 为长安 HEV 整车控制系统。



HCU—整车控制器；ECU—发动机电控单元；BCU—电池控制单元；
IPU—智能功率单元；ISG—起动发电一体机；DCU—显示控制单元

图 5-1 长安 HEV 整车控制系统
Figure 5-1 The control system of ChangAn HEV

整车控制器根据驾驶员的操纵信息，实时对 HEV 的整车运行工况进行控制，对动力流进行调配和控制，进行驱动模式的切换，并高效率的回收制动能量。车辆的显示控制系统，除保持传统车辆的各种行车数据和发动机的控制数据的传递和显示功能外，增加了对电力系统的蓄电池、ISG、驱动电动机和 DC/DC 变换器的控制数据的显示。

发动机电控单元根据动力总成模块发出的指令，经过 CAN 总线控制发动机的子系统操纵电子气门的开度，调控发动机的功率输出，实现启动和停车控制。

长安混合动力汽车是一种串联式中度混合动力轿车，在发动机输出轴上装配有 ISG，发动机和 ISG 的动力混合后通过变速器变速后带动车辆行驶。ISG 子系统控制的控制单元 IPU 根据整车控制策略来转换 ISG 的状态，在起动时蓄电池提供电能供给 ISG，ISG 进入电动状态带动发动机快速起动，发动机启动后转换 ISG 为发电状态，将发动机多余的机械能量转换为电能储存到蓄电池中。在车辆加速或爬坡时，蓄电池供给 ISG 电能，ISG 进入电动状态为发动机提供辅助动力。在车辆制动或下坡时，ISG 进入发电状态回收制动或下坡时反馈的能量。蓄电池子系统

根据蓄电池 SOC 的变化控制蓄电池的充电或放电。

电池控制单元，也就是电池管理系统，它负责动力电池的监控，状态估计，高压控制，通过通信总线向整车提供电池的实时运行数据，方便整车控制器采取合适的控制策略。

5.1.2 混合动力轿车运行模式

停机模式。混合动力轿车停车时，发动机关闭，电机也停止工作，传动系统中没有能量的传递，动力总成各部件均不工作。

启动模式。发动机最初处于关闭状态，当驾驶员将钥匙拧到启动档或空档时踩下离合踏板并同时踩下加速踏板或踩下离合器踏板并挂档时，控制系统判断混合动力轿车为启动模式。在这个阶段，发动机与电机同时产生扭矩驱动车身。由于发动机的工作原理决定了发动机在转速较低的时候效率比较低，排放的尾气中污染物含量较高，而且扭矩也比正常工作时要小，需要电机来产生大扭矩满足整车的功率需求，这样才能减小发动机的负荷，尽快使发动机达到额定转速，达到降低油耗和减少尾气排放的目的。

行驶模式。混合动力轿车所需的驱动行驶转矩可以由发动机单独提供，并由主控制器控制发动机工作在较佳的燃油经济区或低排放区。如果动力电池管理系统监测到动力电池的 SOC 小于设定的下限值时，发动机还提供一部分额外功率驱动电机，电机发电给动力电池充电。车辆加速时，由于发动机控制输入和发动机输出之间存在较大的时间滞后，而电机控制输入与输出的时间滞后小，可能需要动力电池提供能量，由电机给车辆助力，使车辆快速达到期望的车速，当发动机能提供足够的动力，电机就停止助力；上坡时，若发动机不能提供足够的动力，此时电机也通过动力电池提供能量产生助力。助力的前提条件：SOC 大于设定的下限值，并且混合动力轿车处于急加速状态。

制动、减速模式。当加速踏板释放或制动踏板踩下时，动力系统进入制动能量回馈工作方式。控制器根据制动踏板位置和车速，给电机发送负转矩指令，电机依据主控制器指令，执行发电操作，电机作为发电机向动力电池充电，由此产生部分制动转矩，此时动力电池回收整车的部分惯性能量，即能量回馈制动。电机制动转矩的大小受到动力电池的最大允许充电电流的限制。制动能量回馈的前提条件：SOC 小于设定的上限值。

怠速模式。混合动力轿车的发动机工作在怠速模式时，主控制器在空调按钮未按下，发动机温度高于一定值，并且在怠速状态超过一定时间时会关闭发动机。在下列情况下，发动机不会停机：当变速器的档位不是空档时；空调按钮按下；

踩下加速踏板，且发动机温度低于一定值；动力电池电量 SOC 低（小于 30%）。

5.2 BMS 参数对整车控制策略的影响

对于整车控制策略来说，它最关心动力电池三个参数：电池 SOC、电池最大充放电功率和动力电池故障等级。其它电池参数诸如电池总电压、电流、温度等参数只是作为整车控制策略的参考。

电池 SOC，即电池荷电状态，反应了电池当前具有的实际容量。 电池 SOC 是整车的控制策略的关键。只有能够准确的估算电池 SOC 才能让混合动力汽车的节油的控制目标得以实现。整车控制策略会控制电池 SOC 范围在合理的范围，一般电池的 SOC 在 50%时电池的内阻最小，输出的能量效率也是最高的。在 20%~80%电池内阻变化不大，如果超出这个范围内阻变化很大，若让电池不工作在合理的范围，电池的利用率就不会太高，还会对电池的使用寿命和性能造成极大的影响。

最大充放电功率。最大充放电功率是指在电池的当前状态下，电池可以向整车提供的最大放电功率，或者可以承受的整车对其进行的最大充电功率。整车的控制策略中根据最大充放电功率来控制当前对电池的充放电电流，调整能量在整车电气设备中的流动。由于目前电池管理系统不能够直接控制电池输出功率大小，只能通过对电池电压，SOC, 温度的测量来判断电池状态，并以此状态判断电池可以承受的充放电功率，因此电池管理系统需要综合考虑不同电池电压、SOC、温度等条件对应的最大充放电功率曲线然后进行拟合得出最终的充放电功率：在电压、SOC、温度较高时，限制最大充电功率；在电压、SOC、温度较低时，限制最大放电功率，取三条曲线的最低值作为当前动力电池的最大充放电功率上报给整车控制器。

故障的判断及阀值的设定。对于电池本身来说，当电池的单电压过高或过低、电池温度发生剧烈的变化、电池单体之间电压和温度严重不一致可以作为电池故障判断的依据，根据电池本身特性可以对电池进行预报警和故障报警。故障判断应遵循“不误报，报的准确及时，尽量多报”的原则，分等级报故障让整车可以采取一定的措施来应对电池出现的问题。

5.3 标定试验

长安 CV8 型混合动力汽车采用的是湖南神舟科技股份有限公司生产的镍氢动力电池，额定容量 144V，额定容量 6Ah，电压变化范围 96V～198V，工作温度为

－30℃～60℃。

正如上一节所说，对于整车控制策略来说，它最关心电池的三个参数：电池SOC、电池最大充放电功率和动力电池故障等级。其它电池参数诸如电池总电压、电流、温度等参数只是作为参考。为了优化整车的性能，需要对电池电池最大充放电功率这个参数做好系统匹配，而电池SOC需要管理系统采取合理的算法进行估算，电池故障等级则由电池本身外特性和应用环境所决定。

从电池外特性来看，可以得到一些基本的规律：电池端电压越高，电池允许的放电功率越大，而充电功率越小；电池端电压越低，电池允许的放电功率越小，而充电功率越大；电池SOC越高，电池允许的放电功率越大，而充电功率越小；电池SOC越低，电池允许的放电功率越小，而充电功率越大。运用这些规律的主要目的一方面是防止电池过充和过放，另一方面是由于镍氢电池自身功率和充放电电流限制。电池电压和SOC还不能完全反应当前电池允许的充放电功率。之所以需要电池电压和SOC单独限制电池的充放电功率，是因为SOC和电压有一定的关系，表现在电池SOC越高端电压越高，但是他们不是线性关系，在不同的充放电状态相同SOC的端电压是不一样的。图5-2是神舟电池20摄氏度3单体SOC-OCV曲线。

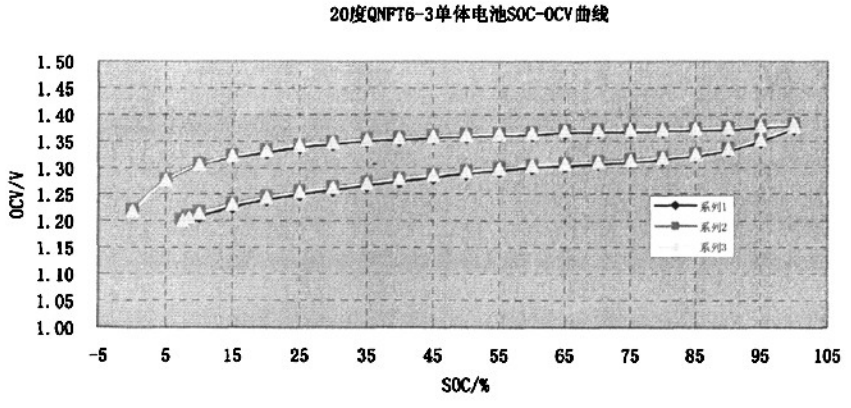
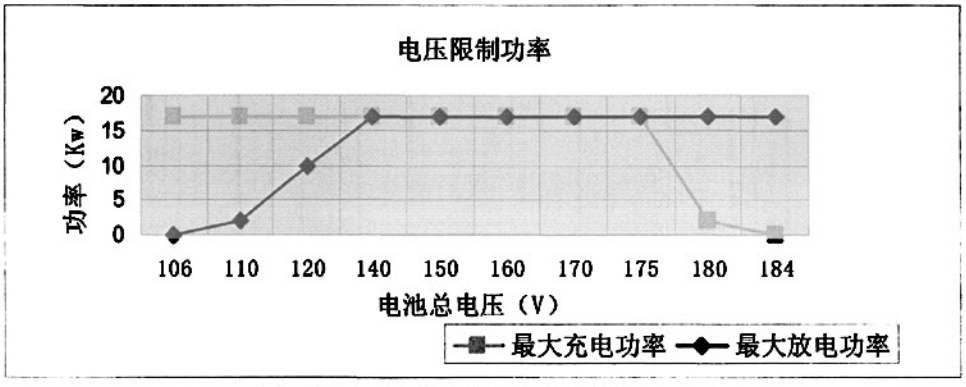


图5-2 6Ah镍氢电池3单体SOC-OCV曲线
Figure 5-2 The SOC-OCV curve of 6Ah Ni-MH battery

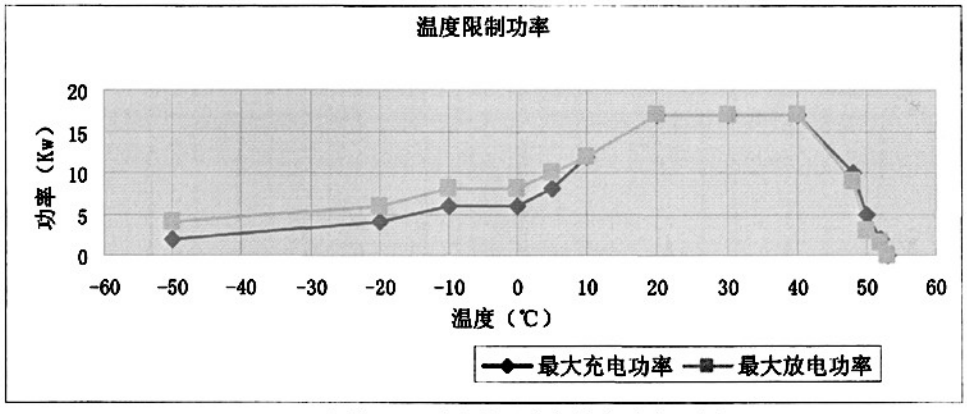
充放电功率还同电池所处环境及自身温度有很大关系。电池的内阻一般比较小，而且不是常数。它会随电池的SOC、温度、使用时间等发生变化。在常温下，电池内部的接触电阻和电池负极的方应阻力是内阻的主要来源，当温度比较低时，电负极反应阻力是电池阻力的主要来源：温度越低，电阻力越大。同时电解液的导电率与温度成正比，温度越低，电导率越低，导致电解液传荷能力降低[34][35]。当温度比较高时，虽然电池电极反应容易进行，但是充电效率低，充电不完全以致电池容量减少，且会缩短电池工作时间，甚至会导致电池漏碱。根据这些分析，

温度是实车运行中功率限制的一个重要因素。

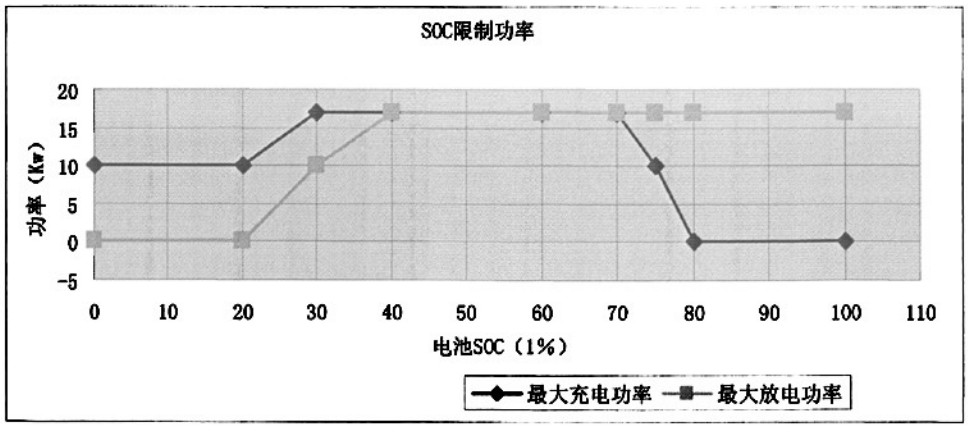
下图是经过实车试验整理后的试验结果：



(a) 电池总电压对应的最大充放电功率限制



(b) 电池 SOC 对应的最大充放电功率限制



(c) 电池温度对应的最大充放电功率限制

图 5-3 最大充放电功率与电池电压、SOC 和温度间的曲线图

Figure 5-3 The maximum charge and discharge power curve of battery voltage, SOC and temperature

通过分析这些实车运行数据，可以很好的与之前的功率控制的理论分析相吻合：在电压比较高，或电池 SOC 较高时，可以认为电池现在的电量比较充足，这个时候电池可以向外输出较大功率，而为了防止电池过充则充电功率会限制的比较低，以此类推电压比较低或 SOC 较低时可以认为电池现在的电量不足，需要充电，即此时的充电功率较大而为了防止电池过放将放电功率限制的较低。温度在限制电池充放电功率则是充分考虑自身的温度特性，在保证电池安全工作的前提下而使电池利用率达到最高。

5.4 本章小结

本章主要介绍了长安混合动力汽车控制系统以及BMS参数对整车控制策略的影响，并对动力电池的最大功率限制进行了标定试验。通过这些试验，表明标定系统可以较好的完成基本的标定功能。

6 结论

全文总结

随着国家政策大力推广节能与新能源汽车示范，尤其是对混合动力汽车大力扶持，会进一步推进我国新能源的发展步伐。同时也反映了新能源汽车在能源短缺的今天的重要性。作为电动汽车的关键部件，电池管理系统的重要地位已经凸现出来，它已经不是简单的电池状态监控部件，而是一个电池能量管理与性能分析的重要角色，它已经开始参与整车控制策略当中，联系着整车能量的分配，对整车性能有着重要的影响。

正是由于电池管理系统在整车控制策略所占有的重要地位，随着它的发展它所要优化控制的参数将会越来越多，这也将会使它像发动机电控单元一样，需要对各种系统参数进行标定试验并优化。为此，本文借对长安 CV8 型混合动力汽车电池管理系统研发的机会对电池管理标定系统进行研究，以期为以后电池管理系统的标定提供一个良好的工具。

本课题的研究工作总结如下：

(1)、在总结前人对电池管理系统的研究后，针对长安混合动力汽车对电池管理系统的功能要求，设计了电池管理系统整套系统的软硬件方案并得以实现。

(2)、深入学习和研究了 CCP 协议，通过引入 MCD 模型构建了电池管理系统标定总体解决方案。结合 CCP 协议工作机制和电池管理系统硬件平台的特点，开发了标定平台和硬件平台的 CCP Driver。

(3)、鉴于电池管理系统功能的复杂性和方便系统任务的扩展，将实时操作系统 μC/OS-II 引入到系统软件设计，并成功将它移植到电池管理系统的硬件平台上。在此基础之上，将 CCP 驱动程序作为系统应用程序的一部分完成系统标定功能，同时利用 Visual C++开发标定平台上位机软件。

(4)、利用设计的标定方案完成电池最大充放电功率限制的标定试验。

课题展望

车载电池管理系统是混合动力汽车的一个重要部件，也是当前研究的热点，无能是它自身功能，还是它相应的配套应用工具都是不够完善的。课题开发了电池管理标定系统作为电池管理系统应用的配套工具，虽然设计的目标基本完成，但在实际应用发现还需要对一些方面进行补充和完善。鉴于这些问题，需要对以

下几个方面作进一步完善和研究：

(1)、在完成电池管理系统的硬件方案设计后，还需要对提高电池管理系统的SOC估算准确度方面做更多的工作，同时对电池还需要加入健康评价即电池SOH的评价，它需要对电池的内阻大小，自放电率、电池一致性、电池工作电流、电池发热、容量衰退以及电池的工作日志进行综合分析，为进一步发挥电池性能提供有利依据。

(2)、目前开发的CCP协议驱动程序CCP Driver完成了协议主要功能，尚还有部分命令因为在系统开发中没有用到就没有实现，但为了以后系统应用的扩展还需要完成这部分命令的实现。

(3)、本课题开发系统软件设计，将实时操作系统μC/OS-II引入进来，但还需要进一步优化，以求系统达到更好的实时性。

(4)、标定平台的功能尚不完善，部分ASAP标准没有完全实现，这是下一步标定平台软件的重要工作。

参考文献

[1] 胡骅, 宋慧. 电动汽车. 北京. 人民交通出版社. 2003

[2] 陈清泉, 孙逢春. 混合动力汽车基础, 北京: 北京理工大学出版社, 2001

[3] 戎喆慈. 混合动力汽车现状与发展. 农业装备与车辆工程. 2008. 7

[4] 姜久春. 电池管理系统的概况和发展趋势. 新材料产业. 2008. 7

[5] 魏学哲. 车用电池及其管理系统分析. 汽车电子. 2006. 7

[6] 陈振辉. 基于 CCP 协议标定混合动力车整车控制器. 上海交通大学硕士论文. 2007

[7] 饶运涛等. 现场总线 CAN 原理与应用技术. 北京. 北京航天航空大学出版社. 2003

[8] 史久根等. CAN 现场总线系统设计技术. 北京. 国防工业出版社. 2004

[9] BOSCH. CAN Specification Version 2.0. 1991. 9

[10] 李计镕, 钟再敏. 车载控制器匹配标定 ASAP 标准综述. 汽车技术. 2004. 10. 1-4

[11] H.KLiebknecht. CCP/CAN Calibration Protocol Rev.2.1. 1999

[12] Kim Lemon. Introduction to CAN Calibration Protocol. Version 2.0. 2003

[13] ShiWei Yang, Lin Yang, and Bin Zhuo. Developing a Multi-node Caliberation System for CAN Bus Based Vehicle. IEEE 2006. 199-203

[14] 李雅博, 张俊智, 甘海云等. 基于 CCP 协议的 HEV 用 ECU 标定系统设计. 汽车工程. 2004. 26(4). 375-378

[15] 姜久春, 牛利勇, 张欣. 混合动力汽车电池管理系统的研究. 高技术通讯. 2004. 6. 75-77

[16] Wang Jun-xi, YANG Lin, FENG Jing, ZHUO Bin. Development of a new calibration System for Electronic Control Units Based on CCP. Transaction of CSICE. 2005. 23 (2). 147-154

[17] Jiapeng Wen, Jiuchun Jiang. Battery Management System For the Charge Mode of Quickly Exchanging Battery Package. IEEE Vehicle Power and Propulsion Conference (VPPC), September 3-5, 2008

[18] Bharath Pattipati, Krishna Pattipati. Automotive Battery Management Systems. IEEE AUTOTESTCON 2008

[19] Do Y J, Baek H L, Sun W K. Development of battery management system for nickel-metal hydride batteries in electric vehicle applications. Journal of Power Sources. 2002

[20] Jiaxi Qiang, Lin Yang, Guoqiang Ao, Hu Zhong, Battery Management System for Electric Vehicle Application, Vehicular Electronics and Safety, IEEE International Conference on13-15 Dec. 2006

[21] 邵贝贝. 单片机嵌入式应用的在线开发方法. 北京. 清华大学出版社. 2004

[22] Motorola, Inc. MC9S12DT128 Device User Guide V02.10. 2004. 2

[23] 邱宝梅, 熊键, 王平. 一种对标定数据的综合处理新方法. 自动化技术与应. 2008. 28(7). 132-134

[24] 宋国民. 基于 CCP 的电控柴油机标定系统开发. 柴油机. 2004. 5. 10-12

[25] 谭浩强. C 程序设计教程. 北京. 中国科学技术出版社, 1994

[26] 赵立业, 张激, 游夏. 实时操作系统的性能分析和评估. 计算机工程. 2008. 34(8). 283-285

[27] 吴智强. 基于嵌入式实时操作系统的电动汽车充电站网络监控系统的研究. 北京. 北

京交通大学. 2007

[28] 任哲. 嵌入式实时操作系统 uC/OS_II 原理及应用. 北京. 北京航空航天大学出版社. 2005

[29] (美)Jean J.Labrosse 著, 邵贝贝译. uC/OS_II 源码公开的实时嵌入式操作系统(第二版). 北京. 中国电力出版社. 2003

[30] http://www.ucos-ii.com

[31] 孙鑫, 余安萍. VC++深入详解. 北京. 电子工业出版社. 2006

[32] (美) Jeffrey Richter 著, 王建华译. Windows 核心编程. 北京. 机械工业出版社. 2000

[33] 张贵强, 张付军等. ECU 的多任务、多数据结构标定系统的研究开发. 车辆与动力技术. 2004. 2. 22-26

[34] Kim Junbom, NguyenTV, and White RE.J. electrochem soc. 1994. 141. 333-337

[35] 李芳. 镍氢动力电池峰值输出功率测试方法. 中南大学硕士论文. 2007

附录 A

附表 CCP 命令代码一览表

Table of CCP Command Codes

命令名	命令代码	描述	响应超时限制(ms)
CONNECT	0x01	建立逻辑连接	25
GET_CCP_VERSION	0x1B	获取 CCP 协议版本号	25
EXCHANGE_ID	0x17	主从机之间互相交换 ID 号	25
GET_SEED	0x12	请求对从机受保护的资源的受控码	25
UNLOCK	0x13	对从机受保护的资源解锁	25
SET_MTA	0x02	当内存区数据传送设定基地址指针	25
DNLOAD	0x03	将主机数据下载到从机内存	25
DNLOAD_6	0x23	主机下载 6 个字节数据到从机内存	25
UPLOAD	0x04	从机向主机上传数据	25
SHORT_UP	0x0F	从机向主机上载制定地址区数据	25
SELECT_CAL_PAGE	0x11	设置 MTA 所处的活动数据内存页	25
GET_DAQ_SIZE	0x14	获取从机 DAQ 资源	25
SET_DAQ_PTR	0x15	设定 DAQ 数据元素存储位置	25
WRITE_DAQ	0x16	设定该元素的详细信息	25
START_STOP	0x06	启动或停止制定 DAQ 列表	25
DISCONNECT	0x07	断开连接	25
SET_S_STATUS	0x0C	设置主从间的会话状态	25
GET_S_STATUS	0x0D	获取从机会话状态	25
BUILD_CHKSUM	0x0E	内存块数据校验	30000
CLEAR_MEMORY	0x10	擦除从机 FLASH 从 MTA 开始的存储区	30000
PROGRAM	0x18	向 FLASH 写入指定长度的数据	100
PROGRAM_6	0x22	向 FLASH 写入 6 个字节的数据	100
MOVE	0x19	从机内存块数据转存	30000
TEST	0x05	测试从机是否支持 CCP 协议	25
GET_ACTIVE_CAL_PAGE	0x09	获取当前活动的数据内存页	25
START_STOP_ALL	0x08	启动或停止所有 DAQ 列表	25
DIAG_SERVICE	0x20	系统诊断服务	500
ACTION_SERVICE	0x21	执行系统请求服务	5000