论文编码：TP39

首都师范大学
Capital Normal University

首都师范大学本科生毕业设计论文

多功能智能车的设计与实现
——智能车的改进与实现

Design and Implementation of Multifunctional Intelligent Car
- Improve and Implementation of Intelligent Car

论文作者_____
院　　系　信息工程学院
专　　业　电子信息工程
学　　号_____
指导教师_____
完成日期　2014 年 4 月

首都师范大学本科生毕业设计论文

摘　　要

智能车是现代电子计算机技术等最新成果和机械成果相结合的产物。智能车在军事、科学研究上十分重要，如美国的“勇气号”、“好奇号”火星探测车以及我国的“玉兔”号月球探测车等。这些智能车往往可靠性极高，功能复杂；而智能车在民用领域以玩具为主，这些智能车往往可靠性低、功能单一。本文设计了一种基于分布式处理器方案的智能车系统构架及智能车，使其能够灵活进行软硬件裁剪，面向中端应用领域。

本文主要做了以下工作：

1.设计了智能车系统构架，采用主从机的分布式处理方案，使智能车能够灵活的进行软硬件裁剪。

2.对现有的分布式处理器通讯方案进行分析。并在此基础上改进，得到一种在效率与可靠性之间取得平衡的设计方案。该方案协议简单，传输高效，具有较高可靠性。

3.对直流有刷电机工作原理进行分析，提出启动门限电压的概念。并对现有的电机控制方案进行分析，在此基础上改进。改进方案具有节能性好、电机调速平滑等特点。

4.设计硬件电路解决 H 电桥中的死区时间问题。

5.将上述功能以及摄像机云台、电源系统、通讯系统、车身等模块实物实现。

6.制作出智能车实体。

实验证明该系统构架能够进行灵活的软硬件裁剪；系统构架及通讯协议高效、可靠；电机控制平滑良好。

关键词：智能车;系统构架;分布式处理通讯;电机控制

首都师范大学本科毕业设计论文

Abstract

Intelligent car is a product combined with the newest achievements of modern computer technology and machinery. Intelligent car plays an important role in the military and scientific research field, such as NASA's Spirit and Curiosity Mars rovers and China's rabbit lunar rover. These smart cars are often high reliability and have complex functions. Intelligent cars, which are used as toys in civilian field, often have low reliability and a few functions. This paper devised an intelligent car for general applications whose hardware and software can be tailored flexibly by using distributed processor architecture.

Major descriptions:

1. Designed an intelligent car system by using master-slave distributed processing method. This makes the hardware and software of intelligent car can be tailored flexibly.

2. Analyzed existing distributed communications methods. Then found a method to balance the design between efficiency and reliability.

3. Analyzed DC brush motor working theory, and proposed the concept of Start threshold voltage. Analyzed the existing motor controls method, and improved it. The improved method has better energy conservation and stepless speed control.

4. Designed a hardware circuit to solve the dead time problems in H Bridge.

5. Implemented those functions above as well as, camera pan and tilt, power systems, communication systems, car body and other modules.

6. Manufactured the entitative intelligent car.

Experimental results show that the software and hardware can be tailored flexibly on this system architecture. System architecture and communication protocol are efficient and reliable. Motor's speed controls stepless.

Key words: intelligent car; system architecture; distributed Processing communication; motor controls

首都师范大学本科毕业设计论文

目　录

第 1 章 绪论 1
1.1 引言 1
1.2 智能车研究背景和意义 1
1.3 智能车在国内外研究现状 2
1.3.1 国内研究现状 2
1.3.2 国外研究现状 3
1.3.3 总体现状 3
1.4 本文研究内容 3
1.5　论文结构 4
第 2 章 智能车系统指标与构架 5
2.1 智能车设计指标 5
2.1.1 常用的系统指标 5
2.1.2 系统设计指标 5
2.2 智能车总体系统构架 6
2.2.1 常用的系统构架 6
2.2.2 本文的系统构架 8
2.2.3　本系统构架性能分析 11
第 3 章 本文提出的三种改进设计 13
3.1 分布式处理器通讯协议 13
3.1.1 常用的多处理器通讯方案 13
3.1.2 多处理器通信模型改进设计 14
3.1.3 多处理器通讯改进设计性能分析 19
3.2 电机控制模型 21
3.2.1 常用的电机控制方案 21
3.2.2 电机运动分析 22
3.2.3 电机控制改进设计 24
3.2.4 电机控制改进设计性能分析 29
3.3 硬件防死区时间 31
3.3.1 死区产生 31
3.3.2 常用的专用芯片死区解决方案 32
3.3.3 硬件电路死区解决改进设计 32
第 4 章　系统实现 35
4.1 微控制器、处理器选型 35
4.1.1 电机、伺服舵机控制器选型 35
4.1.2　主处理器选型 36
4.2 驱动库 37
4.3 多处理器通信协议栈实现 38
4.3.1 协议栈程序特性 38
4.3.2　协议栈功能结构框图 38
4.4　电机驱动方案实现 40

首都师范大学本科毕业设计论文

4.4.1 电机驱动方案程序特性 41
4.4.2 程序功能结构图 41
4.5 电机控制器的编程实现 42
4.6 伺服舵机控制器的编程实现 42
4.7 电机控制电路的实现 43
4.8 私服舵机控制电路的实现 44
4.9 电源系统的实现 45
4.9.1 线性稳压电源 45
4.9.2 开关稳压电源 46
4.10 智能车车身的实现 47
4.10.1 电机驱动装置 47
4.10.2 行动方式 48
4.10.3 智能车成品实物 48
第 5 章 性能结果评估 50
5.1 基本指标实现总结 50
5.2 功能指标实现总结 50
5.3 各模块性能总结 50
第 6 章 总结 52
参考文献 53
致 谢 55
附录一：智能车电机控制(应用层)-传输协议 56
附录二：智能车私服舵机控制(应用层)-传输协议 57
附录三：UART.h 头文件 58
附录四：COM_UART.h 头文件 61
附录五：MOTOR.h 头文件 65

首都师范大学本科毕业设计论文

第 1 章 绪论

1.1 引言

智能车(Intellignet Car)是现代电子计算机技术等最新成果和机械成果相结合的产物。智能车上一般加载有以微处理器为控制核心的电子系统，配合外围电路驱动执行部件(如电机)进行行走和相应动作。同时结合外部诸多传感器获取周围环境数据(如图像、GPS 坐标、温度等参数)，传送至处理核心进行相应的处理，并采取适当行动以躲避障碍物、自主寻路等。

智能车是属于电子计算机和机械科学的交叉产物。智能车的发展水平表明了一个国家的机械和电子产业的综合实力。智能车的研究和发展将会对智能汽车产业的发展起到重要作用。随着人工智能、信息论、控制技术以及决策理论等多门学科的相互融合智能车、智能汽车将会更加智能，能够做到良好的人机交互、自主驾驶、绿色节能等[1]。

智能车还在科学考察、军事侦察、危险作业等领域发挥重要作用。因体积小巧并能携带诸多传感器、执行机构和无线通讯系统，智能车能够前往一些人类所不能涉足的地方(例如核辐射污染区域、外太空考察等)，为人类的科学发展、生命财产安全作出重要贡献。

1.2 智能车研究背景和意义

智能车思想雏形最早出现于上世纪 50 年代，在美国一个仓库内用于物品自动传输的车辆引导系统。进入 80 年代，随着集成电路和计算机技术的飞速发展，智能机器人的相关探索研究正在飞速发展[2]。到了 90 年代末，21 世纪初，巨大规模集成电路的成熟和人工智能、机器决策等相关理论的完善让智能车的发展进入了新纪元。这一时期的智能车可谓种类繁多，百花齐放。

现代智能车依据需要和功能大致可以分为以下几个种类：1.作为深空探测器到其他行星进行科学考察(例如美国发射的火星探测车好奇号与我国发射的月球探测车玉兔号)。这类智能车造价昂贵、结构复杂、为专用目的而设计。2.作为地质探测、考古等科学考察。3.无人侦察、遥控作战等作为军事用途的智能车。4.在危险、极端环境下进行作业的智能车。5.灾后救援的救援智能车。6.在公路上自动驾驶躲避障碍物的智能交通智能汽车。7.作为玩具、模型、安保等用途的民用小型智能车。

智能车已经融入了人类社会的生产生活、军事科研等各个领域中，并发挥着重要作用。智能车的发展将会对人类文明的进步起到重要促进作用，对国家的经济生产繁荣、军事力

量强大、科学水平提高有重要意义。正因如此各国的高校、研究机构、公司都在研究功能更强大、性能更好，结构更优的智能车。

1.3 智能车在国内外研究现状

1.3.1 国内研究现状

清华大学早在 1986 年就开始了我国智能车研究，并研究至今开发出用激光雷达、摄像头作为传感器的的无人驾驶智能车[3]。1989 年，国防科技大学研制出我国首辆智能小车，长 100cm，宽 60cm，重 175kg。1992 年国防科技大学研制出第一辆由计算机、传感器、液压系统构成的无人智能车。该车可以人工驾驶也可以自主驾驶[4]。随后我国又开展了中欧合作研究无人驾驶智能车。我国还举办了许多智能车竞赛。

2011 年 7 月国防科技大学研究出红旗 HQ3 智能车。该车依靠自身摄像头等传感器结合先进的算法实现了从长沙到武汉 286 公里的自主驾驶。这标志着我国智能车在环境识别、智能决策和控制方面都有了新的突破[4]。

2013 年 12 月中国在西昌卫星发射中心成功将载着“玉兔”号智能车的嫦娥三号探测器送入轨道。该智能车重 140kg，长 150cm，宽 100cm，高 110cm，由移动、结构与机构、导航与控制、综合电子、电源、热控、测控与数传和有效载荷等分系统组成。该智能车携带全景相机、测月雷达、红外成像光谱、X 射线谱仪等设备[5]。该探测车的出现标志着我国在深空无人远程遥控智能车领域又向前迈出一大步！(如图 1.1 所示)

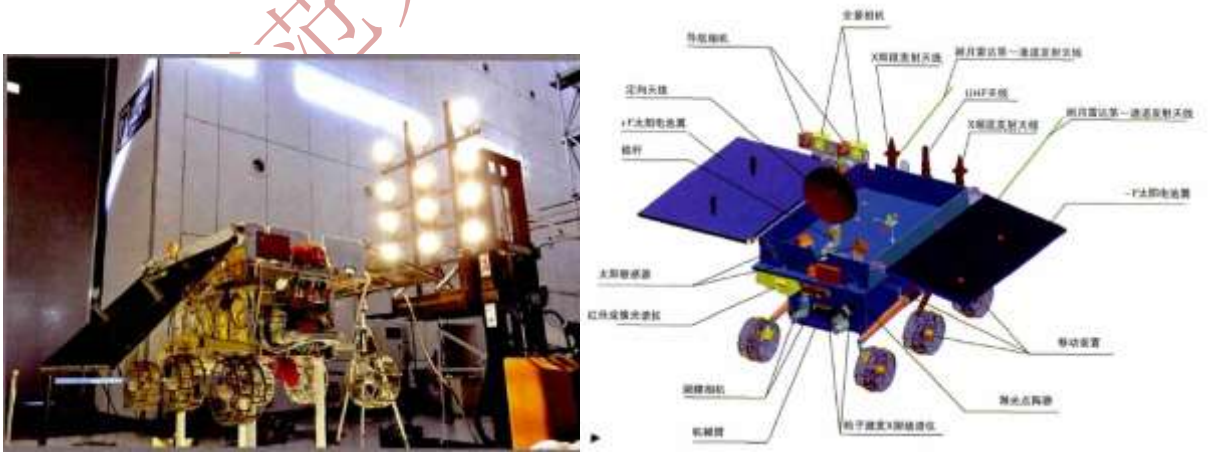


图 1.1 月兔智能车实物图(左)　月兔智能车功能图(右)

而我国民用领域的智能汽车却还处于试验阶段，尚未量产。军用智能无人战车还在研发和试运行阶段，科学考察用智能车已经开始投入使用，而民用智能车更多则是处于模型

玩具阶段。各大高校正积极研究智能车自助寻路、壁障、驾驶等领域。

1.3.2 国外研究现状

20 世纪 70 年代发达国家便开始了智能车的研究，美国和德国走在前列。随后荷兰、法国、意大利、日本等国都相继研发。到 2010 年，美国谷歌研制出智能无人驾驶汽车，能够根据其自身携带的雷达、镜头、激光测距仪实现自主驾驶[4]。

美国曾发射数辆深空探测智能车。其中 2004 年 1 月着陆的勇气号火星探测智能车和 2013 年着陆的好奇号火星探测智能车标志着美国的深空无人远程遥控智能车领域已经走在了世界前沿。其中好奇号火星探测智能车是核动力火星车，并带有大量设备。

美国智能车领域发展蓬勃。智能汽车已经完成研发阶段，正在测试并准备量产。美军已使用能远程无线操控的智能车作战，使人员伤亡将至最低。同样在民间智能车也大显身手，从科学考察、仓储物流、治安维护、巡逻安保等领域都有它的身影。日本每年也举办大大小小各种形式的智能车竞赛，鼓励在校生积极参与、制作、了解智能车。

1.3.3 总体现状

总体而言智能车主要分为两个级别。一个是高端应用，另一个是低端应用。

在高端应用(如深空探测、军事作战、科学考察)中，智能车往往为某一特定目的而设计，所以一旦配置完成，若要更换上边的传感器等设备将十分麻烦。而且由于这类智能车要求可靠性极高，所以造价昂贵、系统结构复杂。这就导致了难以维护和升级、对维护升级人员素质要求高。

在低端应用(如作为模型玩具的遥控车)中，智能车有限的设备只能实现几种功能。这类智能车作为商品往往要控制成本，所以这类智能车结构简单，没有体系标准，无法进行升级维护。

1.4 本文研究内容

本课题设计并实现一辆智能车，研究内容：

1.设计了可扩展的智能车构架，采用分布式处理器方案。

2.本文提出并实现了一套基于 UART 总线的轻量级多处理器间通讯协议。

3.本文提出了更节能高效的电机控制方法与模型。

4.本文设计了硬件电路。

5.组装并实现整个智能车系统。

1.5 论文结构

第 1 章绪论简述了智能车的发展史和现状，并提出了本文主要研究的内容。

第 2 章是本智能车的指标分析和构架提出。

第 3 章是本文提出了三点改进方案：本文提出了一种轻量级、高效可靠的主从模式分布式处理器通讯协议；本文对电机控制提出改进，引入了电压门限、定步长控制、刹车状态等概念；设计硬件电路解决 H 电桥的死区时间问题。

第 4 章是智能车各系统的软、硬件实现。

第 5 章是智能车的性能试验总结。

第 6 章是总结。

首都师范大学本科毕业设计论文

第 2 章 智能车系统指标与构架

2.1 智能车设计指标

指标对系统设计十分重要，它明确了系统所需性能，为系统整体设计方案指明方向。

2.1.1 常用的系统指标

以下介绍常用衡量智能车性能的基本指标：

1.负荷量(kg)：智能车所能承载的最大重量。

2.最高速度(m/s)：智能车最高能达到的速度。额定功率不变，速度越快拉力越小。如果速度过快，智能车将因拉力不足而无法爬坡。公式 2.1 是功率、速度、拉力关系。

$$W = Fv \tag{2.1}$$

其中 W 是输出功率，F 是拉力，v 是速度。

3.电机输出功率(W)：智能车电机装置对外做功的功率。输出功率越大，智能车的运动性能越好。但同样意味着耗电量将增加，待机时间将减少。

4.电池容量(J)：智能车蓄电池所含电能量。电池容量越大，智能车待机时间越长、活动范围也就越大，但同样会导致重量的增加。

5.待机功率(W)：智能车待机时所消耗的电功率。

6.最大功率(W)：智能车所有设备打开，并全速运行时所消耗的电功率。

2.1.2 系统设计指标

本智能车为多功能智能车，故智能车设计时应考虑到能够搭载许多设备传感器，智能车的负载量和电池容量应增大。表 2.1 给出本智能车基本指标。

表 2.1 本设计智能车指标

指标	参数
负荷量	大于 3kg
最高速度	大于 2m/s
电机输出功率	大于 20W
电池容量	100kJ
待机功率	小于 10W
最大功率	小于 200W

除上述基本指标外，本智能车应完成如下功能：

1.本智能车搭载摄像机以及摄像机云台，该摄像机云台系统能够旋转并将实时图像传

首都师范大学本科毕业设计论文

回。其有效无线距离在开阔地带应大于 300m。

2.本智能车有电机控制器，能够实现电机正反转、调速。并且实现缓启动、缓变速。并让电机发挥最好性能。

3.本智能车具有可裁剪的特性。本智能车系统构架要支持传感器、各个模块的方便灵活添加与移除。

4.本智能车采用分布式处理器方案，故应设计一套高效总线使得各个处理器可以正常通讯。

5.本智能车可以远程操控。

2.2 智能车总体系统构架

智能车系统构架包括智能车系统结构和智能车系统设计思想。系统构架对智能车功能的发挥、智能车的特点和性能有巨大影响。为完成智能车的模块可裁剪性指标，要依照智能车系统特点设计系统构架和模块划分。先从宏观把握智能车整体结构，并让其各个模块之间分工明确，良好协作。

2.2.1 常用的系统构架

当前民用智能车的常用系统构架方案分为如下几大类：

1.使用单处理器为核心的系统构架[6][7][8][9](如图 2.1、2.2 所示)。

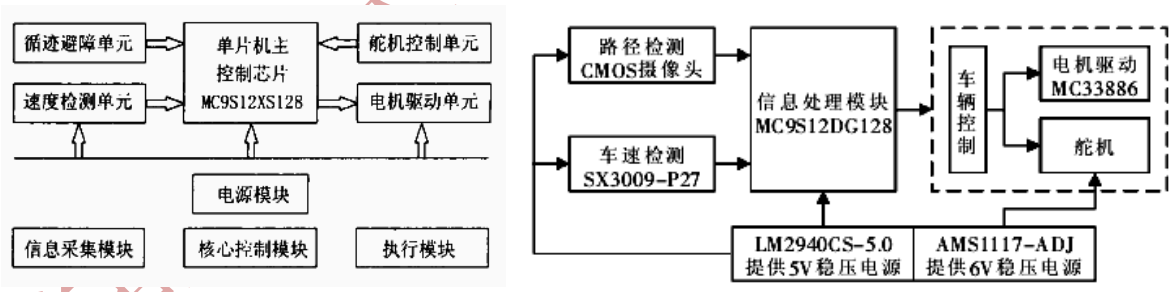


图 2.1　文献[6]的系统构架(左)　文献[7]的系统构架(右)

本科毕业设计论文

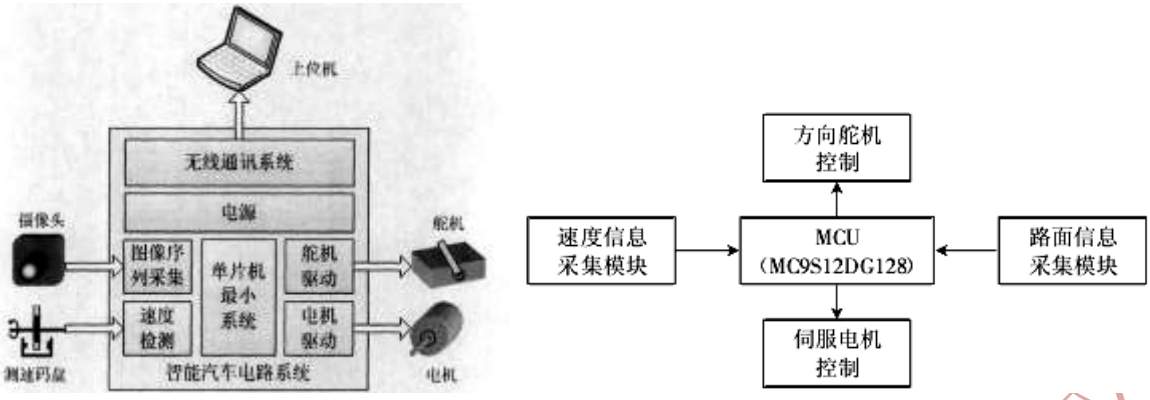


图 2.2　文献[8]的系统构架(左) 文献[9]的系统构架(右)

这类系统构架的特点是使用一块处理器或单片机来完成整个智能车的系统控制。

其优点是：

A.构架设计简单易于实现；

B.编程方便只需在一片处理器上编程便可完成整个功能。

C.对于简单功能而言开发周期短、开发效率高。

其缺点是：

A.功能单一不利于拓展，后期二次开发升级的困难。

B.单片微处理器系统结构需要众多外设，这导致芯片的尺寸和选型受限制。

C.单机微控制器主频、总线位宽不高、计算能力低下、系统资源不足，无法移植操作系统达到“智能化”和进行高密度复杂运算。

D.单处理器需要响应大量中断请求，系统负担重，无法有更多时间执行高级逻辑运算分析程序。

E.系统聚合度高，系统的风险增大。一旦该处理器损坏或程序出现异常，整个系统失控、各个模块异常，轻则死机无响应，重则发生车毁事故。

2.使用复杂总线连接的多处理器系统构架[10]。(如图 2.3 所示)

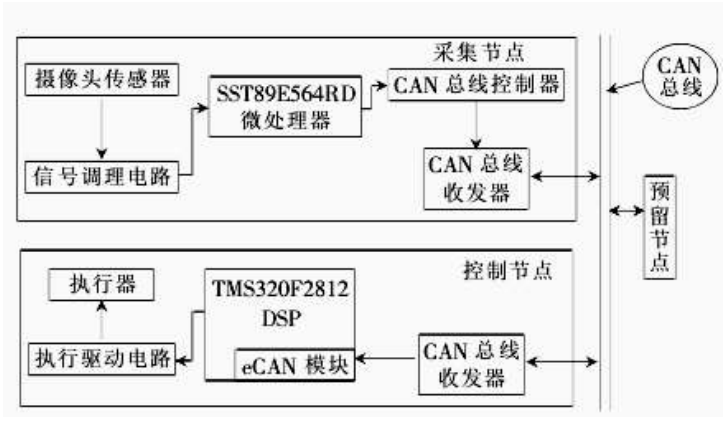


图 2。3　文献[10]中采用的基于 CAN 总线多处理器系统构架

CAN(Controller Area Network)总线，是一种现场总线，已通过 ISO11898 及 ISO11519 进行了标准化，在欧洲已是汽车网络的标准协议。其高性能和可靠性已被认同，并被广泛地应用于工业自动化、船舶、医疗设备、工业设备等方面。它的出现为分布式控制系统实现各节点之间实时、可靠的数据通信提供了强有力的技术支持。

这类系统构架使用高级复杂 CAN 总线互联并使用多个处理器来对不同的模块进行分别控制和处理。

其优点是：

A.传输速率高、可靠性高。

B.中断和检测事件响应及时，系统反应性高。

C.系统模块间的耦合低，系统可靠性增强，系统性事故率降低。

D.系统可靠性、可维护性增强、升级性强。

其缺点是：

A.高级总线结构、协议复杂，开发难度大，低端处理器很难装下整个协议栈。

B.对中小型系统而言增加了系统复杂度和系统成本。

C.复杂 CAN 总线存在不足：消息传输时间在不同填充位下概率不同[11]。Last-But-One-Bit 错误。不同节点状态间相互转换延迟不同造成的不可预测性。Babbling Idiot 错误导致信道出错堵塞等。

2.2.2 本文的系统构架

上述两种常用系统构架各有其优缺点，为了达到一种平衡和折中的方案，本文提出了自己的系统构架和通讯方案。

本文设计为了达到：

- 模块之间低耦合，并且易于维护
- 模块可拓展性可裁剪性强
- 各模块间与主机通讯高效快捷可靠

因此本智能车系统构架定位于快速较高可靠性的中端级别系统，使用了分布式处理器系统，并且设计了一种主从间高效可靠通讯的协议栈。

本智能车分为如下几个模块部分，其模块关系图如图 2-4 所示。

首都师范大学本科毕业设计论文

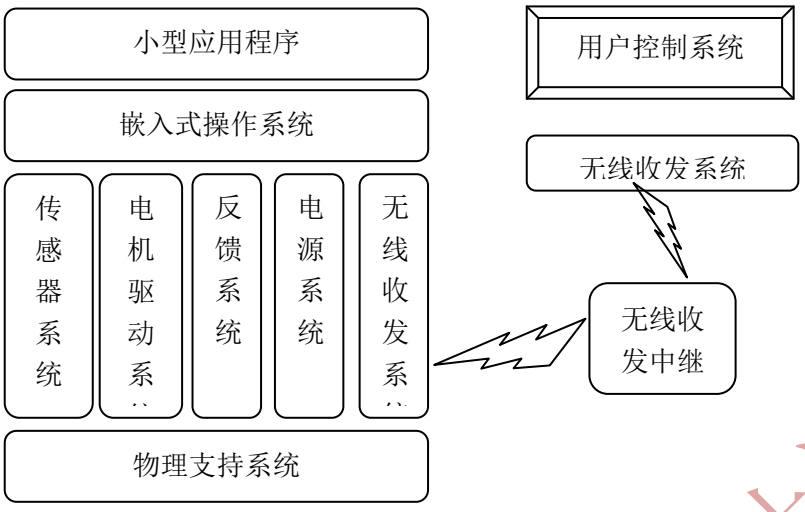


图 2- 4　本文智能车系统结构图

基于模块化划分的思想让智能车的每个部分分工独立明确，让各模块之间低耦合。各个模块各司其职，保证系统稳定运行。下边简述各模块的功能。

嵌入式系统及小型应用程序：

该系统使用 MCU 或 ARM 芯片，其上可搭载小型实时操作系统，保证多程序模块的相互独立和协作及并发执行。这些任务包括智能车正常通讯、处理传感器采集的数据。操作系统上可以运行小型应用程序，扩展了智能车的高级软件功能。

传感器系统：

传感器系统由众多传感器(如陀螺仪、GPS、摄像、温度采集等)组成，这为实时掌控当前处环境和车体状态、精确的自动控制奠定了基础。因为传感器是通过总线变成统一接口方式与主机交互，故该系统是可以依据实际需要进行裁剪的。

电机驱动系统：

驱动系统由电机、齿轮组、驱动电路构成为智能车前进提供动力。电机系统模块的微处理器结合本文提出的控制模型和电路精确控制着电机运转。这些电机可以是智能车动力电机，也可以是其他设备电机。驱动系统可以按照设计要求进行硬件剪裁。

反馈系统：

用于实时数据反馈、实现较强的人机交互性，或依据外部环境自动的做出反应。同样可以依据设计需求进行裁剪。

多供电开关电源系统:

由电池和开关稳压系统组成。电池作为电源为智能车提供能源。开关电源降压电路高效率(大于 90%)。采用多开关电源稳压模块为各个模块提供其所需的不同电压。数字系统电源系统与模拟电源系统分离。

首都师范大学本科毕业设计论文

物理系统：

物理系统主要是车的骨架和机械装置。这些骨架和机械装置是保证智能车运行和执行任务的基础平台，是控制设备发出命令的最终执行体。

无线通讯系统：

该系统是智能车和用户控制器的通讯和数据交互。其带有中继节点，可以进一步延长通讯传输距离。

用户系统：

用户系统与智能车可以进行联机操控，用户能够实时监测到智能车的状态与记录智能车返回的数据。同时，用户也可以对智能车进行实时操控。

图 2.5 是本文为达到上述要求而提出的智能车系统构架。

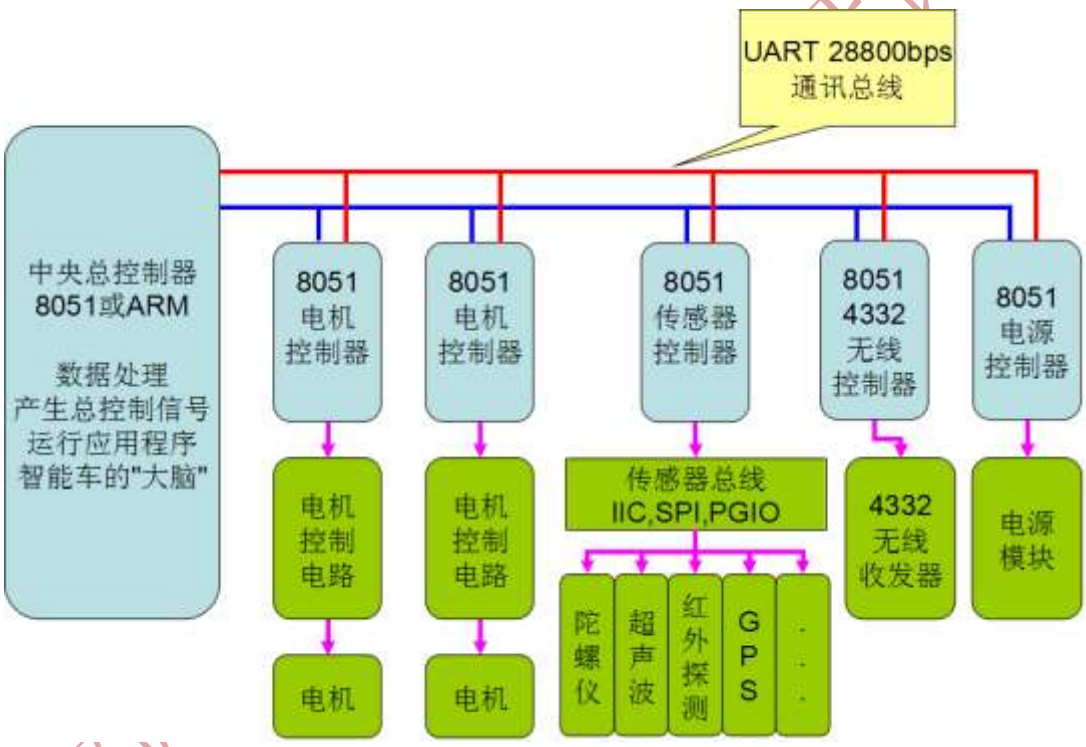


图 2.5　智能车系统构架图

本系统基于一条波特率为 28800bps(可调为 115200bps 或更高)的全双工 UART 总线将 ARM 主机处理器和各个 8051 微控制器模块连接起来。而每个模块微控制器专注其控制模块的控制和相关计算功能。这样做的优点如下：

- 良好的模块隔离：每个模块有专一处理器进行控制和运算，从而达到硬件和程序上的模块划分和隔离。
- 主处理器能发挥更多潜能：运算速度强劲的 ARM 主处理器只需通过总线发布命令不再局限于模块控制细节(如中断处理)，有更多的时间和资源进行高密度复杂逻辑运算发

挥处理器潜能。

- 系统性能最大：该系统构架将 32 位多级流水 ARM 主处理器和外设资源丰富的 8051 低端处理器各取所长相互配合在一起。构成的混合系统同时满足了实时性和高运算性，从而将系统性能发挥到最大。
- 开发维护容易：分布式处理和轻量级总线让模块间的通讯编程变得容易和简单。模块所有的控制运算程序均写入该模块控制器中，并通过标准通讯协议和控制字与主机相连。独立开发、升级模块时，仅需要保持控制字协议一致即可。
- 系统可靠性增强：彼此分离、独立的模块一旦发生程序异常、失效不会牵连其他模块。主机失效后，由于通讯中断，子模块可进入预定保程序防止进一步灾害发生。

若主处理器支持多 UART 接口，则可以依照大数据量、高实时性、普通数据等标准分成不同总线，从而进一步提高系统性能(如图 2.6 所示)。

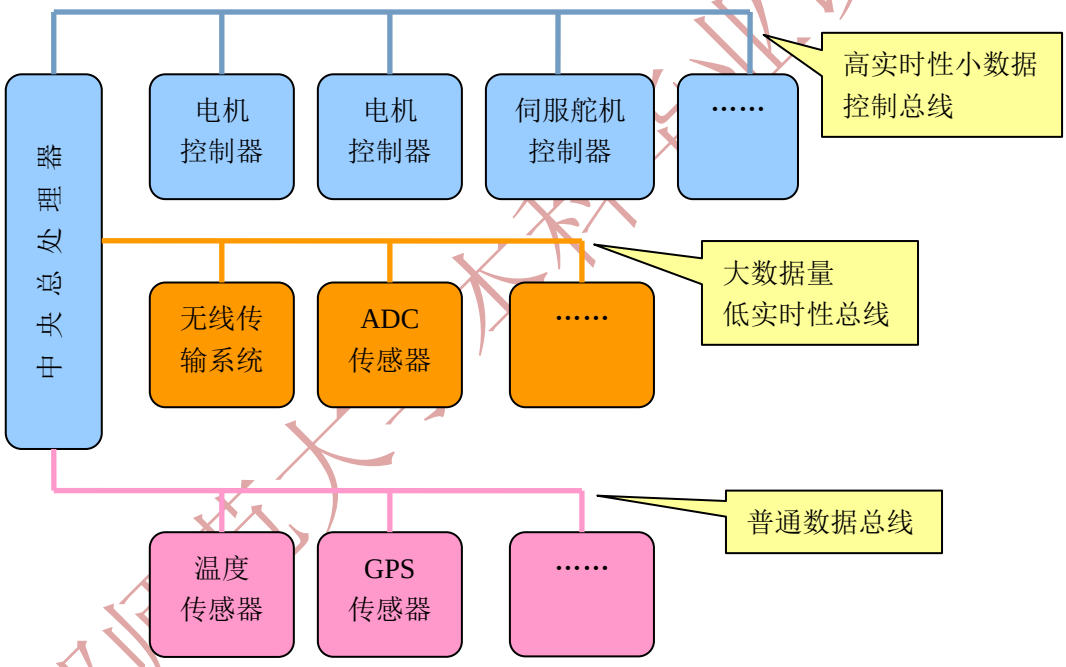


图 2.6　多总线构架

使用多重总线结构能够使智能车整体性能得到进一步提升，能够满足智能车上各个模块对于不同通讯的需求，能满足高实时性小数据通讯，能满足低实时性但大数据量的通讯，也能满足普通的传感器通讯。

2.2.3　本系统构架性能分析

本文制作智能车暂采用本文提出的第一种单总线多分布式处理器。

分析条件见表 2.2

表 2.2 构架分析条件设定

条件	参数
波特率(Br)	115200 bps
主处理器运行速度(OS_m)	72 MIPS
微控制器运行速度(OS_s)	20 MIPS
传输协议报文长度(M_t)	3 Byte
物理帧长度(F_p)	11 bit/Byte
最长容忍时限($T_{\max}$)	2 ms
平均数据长度(DL_{ave})	3 Byte
处理每字节所需指令(IPB)	400 I

计算物理层字节传输速率：

$$TS_p = \frac{Br}{F_p} = \frac{115200bps}{11bit/Byte} \approx 10472.7Byte/S \tag{2.2}$$

主处理器处理每字节指令时间：

$$T_{mPB} = \frac{IPB}{OS_m} = \frac{400I}{72MIPS} \approx 0.006ms \tag{2.3}$$

微控制器处理每字节指令时间：

$$T_{sPB} = \frac{IPB}{OS_s} = \frac{400I}{20MIPS} \approx 0.02ms \tag{2.4}$$

平均协议包长度：

$$PL_{ave} = DL_{ave} + M_t = 3Byte + 3Byte = 6Byte \tag{2.5}$$

公式 2.6 是微控制器传输处理数据平均所需时间 TP_t：

$$TP_t = \frac{PL_{ave}}{TS_p} + T_{sPB} = \frac{6Byte}{10472.7Byte/S} + 0.02ms \approx 0.593ms \tag{2.6}$$

可见传输并处理指令平均时间远小于容忍上限 2ms，故系统实时性可以得到保障。

综上分析，本系统构架既能保障实时性这一重要基本要求，并且在此之上能够做到模块化和模块间的良好交互。故本设计能满足设计指标中的可裁剪和实时性要求。

首都师范大学本科毕业设计论文

第 3 章 本文提出的三种改进设计

依据本智能车系统设计要求并结合常用方案的特性，本文在以下三处进行了改进并提出自己的方案：

- 轻量级的可靠分布式处理器通讯协议
- 带有缓启动、刹车功能基于状态机的电机控制方案
- 硬件解决 H 电桥的电机控制死区问题

3.1 分布式处理器通讯协议

在分布式多处理器系统中同构、异构的处理器之间需要数据交换保证彼此的协同性以便共同完成系统功能。

3.1.1 常用的多处理器通讯方案

目前民用设备上的常用分布式处理器(也包括多处理器)通讯方案如下：

- 基于高级复杂 CAN 总线的分布式处理器通讯方案[12]。

  CAN 总线主要应用于汽车控制等领域。

  其优点：

  - 可靠性极强
  - 传输性能好

  其缺点是：

  - 协议栈大，占用系统资源多，不适合小型微控制器
  - 处理延迟大，系统状态和运算复杂，小型低性能处理器处理延迟大
  - 编程复杂度高
  - 一次最多仅能传送 8 字节数据

- 基于共享存储器的设计方案[13]。

  将各个处理器接入到统一共享存储器中，彼此之间通过互斥访问在共享存储器中存放数据来通讯。这种方案更适合 SOC 系统。

  其优点：

  - 传输协议简单

首都师范大学本科毕业设计论文

- 通讯速率高

其缺点：

- 互斥访问读写导致硬件设计复杂、需要大量互斥标记
- 传输距离近

■ 基于复杂网络的现场总线[14]。

复杂的控制协议和格式来完成主机与从机的可靠通讯。其优缺点如 CAN 总线。

■ 基于专用硬件的通讯方案[15]。

设计一种专用的通讯管理单元完成多处理器、分布式处理器通讯。

其优点：

- 传送高效
- 传输速率快
- 可靠性强
- 节省软件资源

其缺点：

- 设计复杂度高
- 通用性差

■ 基于同/异步串口的多处理器直接通讯[16]。

直接使用数据链路层 UART 协议传输。

其优点：

- 使用简单
- 效率极高，因为直接使用数据链路层传输

其缺点：

- 可靠性极差，无差错控制编码(校验码)

3.1.2 多处理器通信模型改进设计

基于本智能车系统构架方案，本文采用分布式处理器主从机通信方法。本文设计一种介于高级复杂总线与无差错控制协议两者之间的多处理器通讯协议模型。有如下特性：

- 轻量级：报文长度更短，传输效率更高。占用系统资源少。状态机模型简化。
- 可靠：具有差错控制编码(校验功能)，能检测到传输中的差错。支持超时检测。
- 地址分组：按地址发送数据，按地址分组并能分组广播数据。
- 高效：使用更简化的协议状态机，快速完成响应。

首都师范大学本科毕业设计论文

- 可裁减：可以配置是否开启 CRC 校验，以节省系统资源。

3.1.2.1 协议栈模型

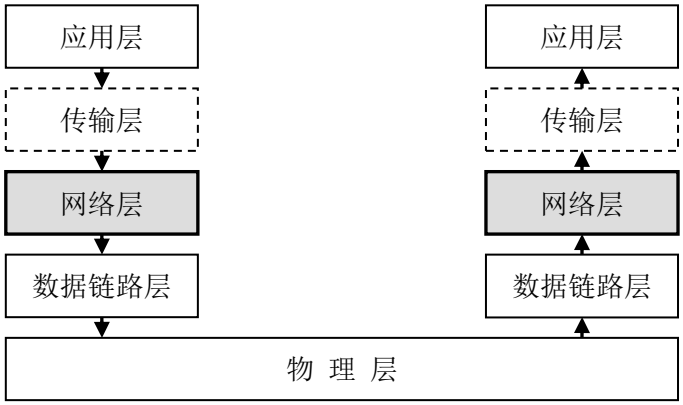


图 3.1 本文定义的传输 5 层协议

如图 3.1 所示，本文采用 5 层传输协议，并对网络层协议、模型进行修改。

3.1.2.2 本文提出网络层协议

图 3.2 是本文提出的轻量级网络层报文协议：

字节	0	1	2	3	……	n+1	n+2
内容	地址 Ad	数据长度 n	数据 1	数据 2	……	数据 n	CRC(可选)

图 3.2 本文网络层数据报文结构

数据包由四部分组成：地址、数据长度、数据、CRC。本文网络层是对等传输协议，即从主机发送到从机、从机发送到主机均使用上述统一格式。本协议控制信息长度为 3 字节(CRC 开启),2 字节(CRC 禁用)。数据最大传输数量为 255 字节。

■ 地址

一个 8bit 的地址信息。当数据包由主机发送到从机时，地址指明了目的从机编号。当数据包由从机发送到主机时，地址是该从机编号，以便主机区分数据包源。发送地址信息的时候需要物理帧中地址/数据标志位为 1，以表明其为地址数据。

地址有如下规则：

1.地址 00H 是主机地址

2.地址 0FFH 是全网广播地址，即对全体从机发送数据，这个地址通常用于紧急情况下的全模块复位或进入紧急状态指令

3.其他地址可作为从机编号使用

4.地址可分组使用

本文提出可分组地址的概念，引入了掩码的使用。其优点如下：

- 可将若干相关模块分为一组进行广播，而其他组的模块不受影响
- 广播使组内所有模块同时接收到数据，实现模块组同步动作(如电机同步运行)
- 广播可以减少数据传输量，提高通讯效率

分组与地址、掩码规则如下：

假设一从机地址为 AD_s，其设定的地址掩码为 AM_s，&运算符表示按位与，~运算符表示按位取反。

则其组地址号：

$$GA_s = AD_s \,\&\, AM_s \tag{3.1}$$

其组内地址编号：

$$SubA_s = AD_s \,\&\, (\sim AM_s) \tag{3.2}$$

当从机接收到的地址 AD_{rec} 使公式 3.3 为真时，表示该该地址对从机有效，从机应该接受此数据包。

$$(AD_{rec} = FFH) \vee (AD_{rec} = AD_s) \vee (AD_{rec} = ((AD_s \,\&\, AM_s) | (AD_s \,\&\, \sim AM_s))) \quad \text{(公式 3.3)}$$

假如从机 A 的地址为 0A5H,从机 B 的地址为 0ACH，两台从机的掩码均为 0F0H。依据公式 3.1，从机 A 组地址为 0A5H & 0F0H = 0A0H，从机 B 组地址为 0ACH & 0F0H = 0A0H，所以从机 A 和从机 B 是属于同一组中。依据公式 3.2，从机 A 组内编号为 0A5H & ~0F0H=05H，从机 B 的组内编号为 0ACH & ~0F0H=0CH。依据公式 3.3，同时能时从机 A 和从机 B 接收数据的地址为：0FFH 与 0AFH。

■ 数据长度

该字段占用一个字节，指明后边将追随多少字节数据。该字段最大有效值为 255，故一个数据包最大可传送 255 个字节的数据。

■ 数据

N 个字节要传输的数据。

■ **CRC**

可选字段，在对于可靠性要求不是很高的系统中，为了节约计算资源提高带宽利用率，可以裁剪掉 CRC 校验代码。

CRC 即循环冗余校验码(Cyclic Redundancy Check)，用于通讯中的纠错检错。使用的生成多项式为常用的 CCITT 标准[17]： $x^8 + x^7 + x^3 + x^2 + 1$。

CRC 校验是对前边地址+数据长度字段+数据整体的校验。

3.1.2.3 接收状态机模型

首都师范大学本科毕业设计论文

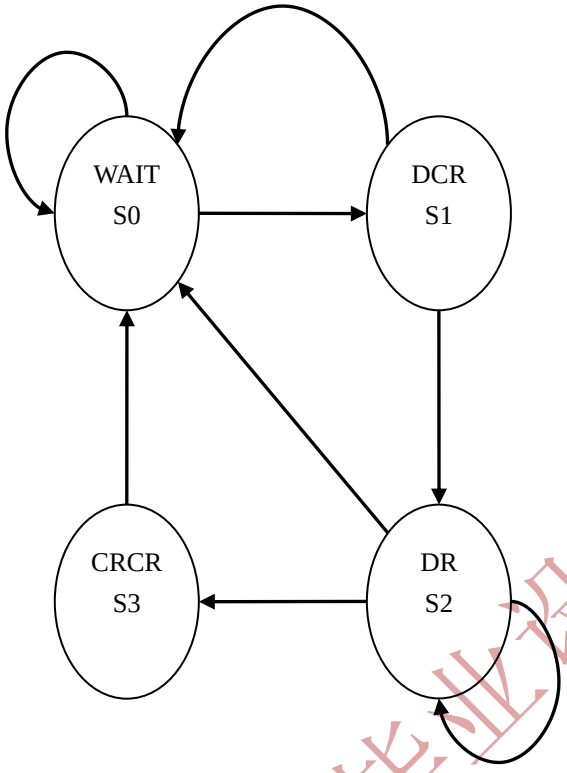


图 3.3　接收状态机模型

■ WAIT S0 (空闲等待状态)

接收方的启动状态，没有数据时的等待状态。该状态下接收方一直在等待一个有效的地址。若在此状态下接收方接收到的不是地址而是数据，则自动忽略此数据，并维持在等待状态(此时的数据是给别的从机发送的)。若从机收到了地址不满足公式 3.3，则维持在等待状态(此时主机呼叫的不是该从机)。若从机收到地址且满足公式 3.3 或者主机收到任意一地址后则进入态等待接收数据长度字段 DCR S1 状态。

■ DCR S1 (等待接收数据长度字段)

这个状态下表示接收方正在等待接收数据长度信息字段。若此时收到一个地址数据，则状态转移到 WAIT S0(在该数据包未完成完整传送的前提下，数据又开始了新的传输，上一个数据包发生异常)，并同时告知处理器出现了接收错误异常(标记相应的状态位)。若在此等待时间超过时限，则状态转移到 WAIT S0(发生了接收超时异常)，并通知处理器出现了接收超时错误。若正常接收到数据字节，则记录下作为数据包长度字段，并转移至等待数据接收状态 DR S2。

■ DR S2 (等待数据接收)

该状态下接收方等待接收有效的数据。若此时收到一个地址数据，则状态转移到 WAIT S0，并同时告知处理器出现了接收错误异常；若等待时间超过时限，则状态转移到 WAIT S0，

并通知处理器出现了接收超时错误；若接收到有效数据，但目前缓冲区中的数据尚未达到纪录的数据长度，则表明此时数据尚未接收完整，需要继续接收。则将收到的数据送入缓冲区，并清零接收超时计时器，维持在 DR S2 状态，等待下一个数据的到来；若此时数据已经全部接受完，则将当前接收的数据放入缓冲区中，如果 CRC 校验被使能则进入等待 CRC 校验字 CRCR S3 状态。如果 CRC 校验未被使能则进入 WAIT S0 状态，等待下一个数据包的接收，并告知处理器已经接收完一个数据包，等待处理器做出处理。

■ CRCR S3 (等待 CRC 校验字)

该状态下接收方正在等待 CRC 校验字的接收。若此时收到一个地址数据，则状态转移到 WAIT S0，并同时告知处理器出现了接收错误异常；若在此等待时间超过时限，则状态转移到 WAIT S0，并通知处理器出现了接收超时错误；若收到有效数据则进行 CRC 校验，并转移到 WAIT S0 状态，如果 CRC 校验没有出现错误，则告知处理器已经接收完一个数据包，等待处理器做出处理。若 CRC 校验出现错误(数据传输出现错误、异常)则丢弃数据包并告知处理器出现了 CRC 校验错误异常。

3.1.2.4 发送方状态机模型

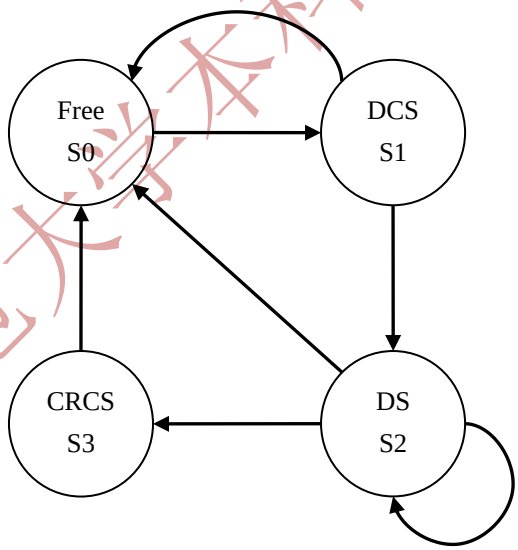


图 3.4 发送方状态机模型

■ Free S0 (空闲状态)

发送方的启动状态。表示准备完毕等待发送。若此时有数据包来临并告知发送方需要发送，则发送方以地址数据发送出去，并转移到状态 DCS S1 等待发送数据长度字段。

■ DCS S1 (等待发送数据长度)

发送方要发送数据长度字段；若接收到硬件通知的发送完成中断或标记，则发送数据

首都师范大学本科毕业设计论文

长度字段，并转移到等待发送数据状态 DS S2；若发送超时计时器超时且未能收到硬件的发送完成中断(发送失败)，则转入状态 Free S0，并通知处理器出现发送超时异常。

- DS S2 (等待发送数据)

发送方要发送数据内容。若接收到硬件通知的发送完成中断或标记，则发送数据，如果此时数据已经全部发送完毕且 CRC 校验被使能，则转移到等待发送 CRC 校验状态 CRCS S3，若此时数据已经全部发送完毕且 CRC 校验未被使能，则进入空闲状态 Free S0，并通知处理器发送完成，若此时数据尚未全部发送完毕，则清零发送超时计时器，并保持状态 DS S2。若发送超时计时器超时且未能收到硬件的发送完成中断，则转入状态 Free S0，并通知处理器出现发送超时异常。

- CRCS S3 (等待发送 CRC 校验)

此状态下发送方将要发送 CRC 校验字。若接收到硬件通知的发送完成中断或标记，则发送 CRC 校验字，发送完成后进入空闲状态 Free S0，并通知处理器发送完成。若发送超时计时器超时且未能收到硬件的发送完成中断,则转入状态 Free S0，并通知处理器出现发送超时异常。

3.1.3 多处理器通讯改进设计性能分析

- CAN 总线、本文通讯模型、基于同/异步串口的多处理器直接通讯模型，性能比较[18][19](如表 3.1 所示)：

表 3.1 几种方案性能比较

比较项目	CAN 总线	本文模型	UART 直接通讯
复杂度	高	低	无
传输速率	较快	快	最快
校验方式	CRC16	CRC8	无
可靠性	高	较高	最差
报文长度	67bit	3Byte	0
发送数据数量	≤8Byte	≤255Byte	1Byte
效率	较高	高	最高
节点数量	理论上无限	小于 255	-
状态数量	>18	8	-
选定节点广播	支持	支持	不支持
协议栈大小	几十 KB 不等	<0.75KB	-

- 效率比较

平均效率：$\eta = \frac{DL}{ML} = \frac{DL}{ML + M_t}$ (3.4)

式中 DL 是数据长度， ML 是数据包长度， M_t 是报文长度。

图 3.5 是三种方式效率比较，可以看出直接传送方案效率最高；CAN 总线传送方案效率较低，且标准 CAN 总线最多一次只能传送 8 个字节；本文方案效率较高，而且最多一次可以传送 255 个字节。

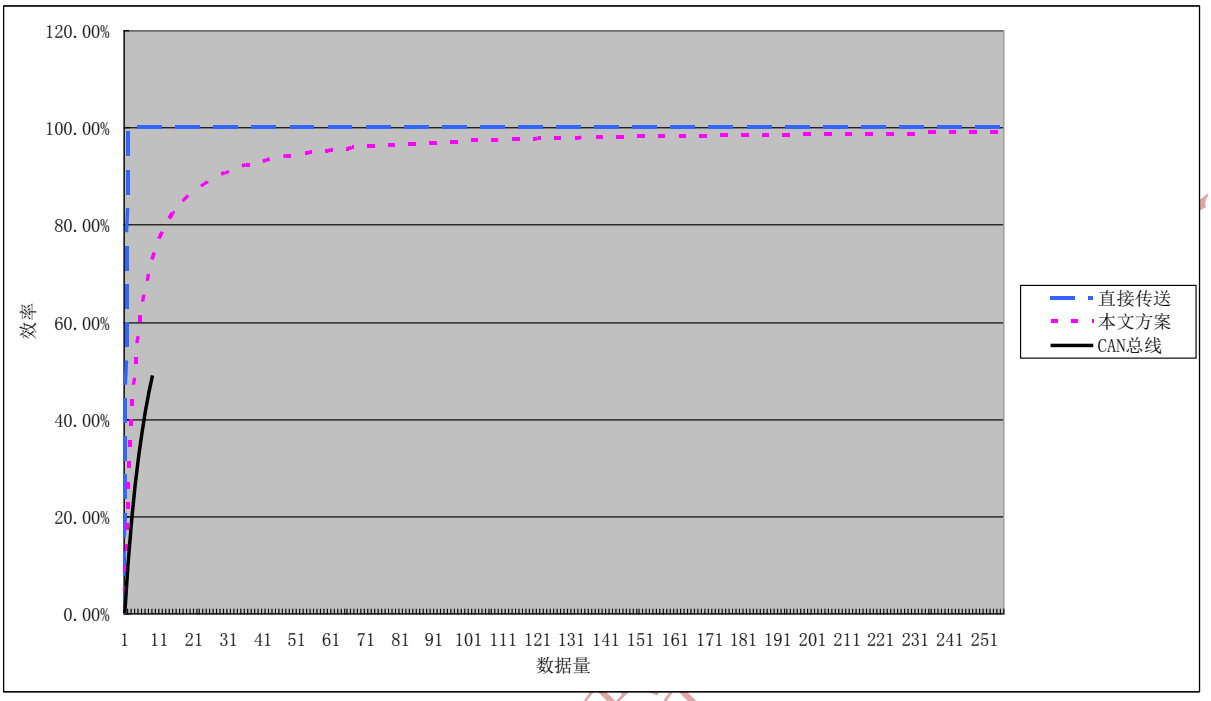


图 3.5　三种方案效率比较

■　可靠性比较

可靠性指出现 n 比特错误后，能被纠正发现的概率。图 3.6 给出三种可靠性方案比较[20][21]示意图。图中是假设传送 64bit 时出现的比特级错码数量。使用 CRC-16 的 CAN 总线可靠性最高；不使用差错控制方式的直接传输几乎没有可靠性；本文方案具有较高可靠性。

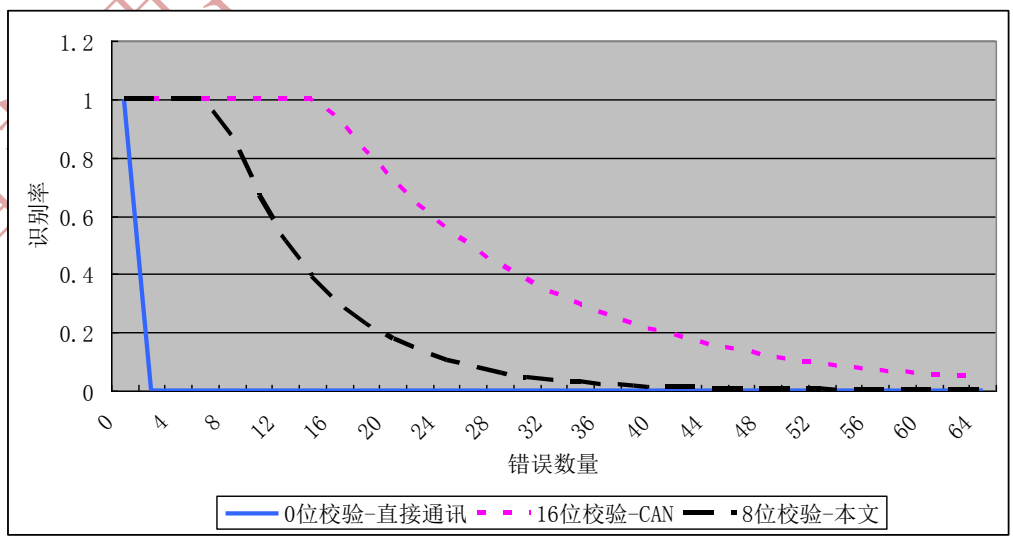


图 3.6　三种方案检错率示意图

*注：CRC 校验码可以检测出所有奇数个错误，故上图中只显示偶数错误对应的检测率。

3.2 电机控制模型

电机控制模型的合理与否将关系到电机的能耗和使用寿命。本文采用直流有刷电机作为智能车驱动电机，因为其调速、启动性能好，堵转力矩大[22]、大范围内平滑地调速、制动及时等优点。

3.2.1 常用的电机控制方案

目前常用的直流有刷电机控制方案如下：

- 基于闭环反馈电路的自动控制系统

  使用精密传感器反馈系统和传递函数进行闭环控制。其优点：

  - 控制精度高
  - 控制效果好

  其缺点：

  - 系统设计复杂，要设计传感器相应电路和对应传递函数
  - 成本较高，要购买相应传感器和较高性能处理器

- 基于固定分段的缓变速控制[23]

  将目标速度与当前速度分为固定 N 份，逐份递增或递减达到缓变速。其优点

  - 控制相对简单，编程实现相对容易
  - 电路简单，不用构成闭环系统，电路没有传感器部分

  其缺点：

  - 浪费能源，控制中存在不合理因素导致能源浪费
  - 电机寿命短，控制中存在不合理因素导致电机寿命缩短

- 无变速直接控制

  无控制方式。直接输出目标速度。其优点：

  - 控制简单，编程实现最容易
  - 电路简单

  其缺点：

  - 能源浪费严重
  - 电机寿命严重缩短

首都师范大学本科毕业设计论文

3.2.2 电机运动分析

有刷直流电机由定子及转子两部分组成。通电后，转子上形成磁场，定子和转子的磁极之间有一个夹角，在定转子磁场的相互吸引下，使电机旋转。如下图所示：

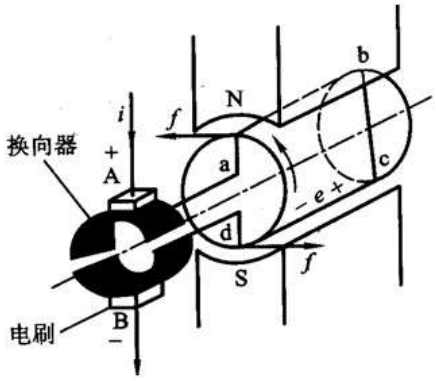


图 3.7　电机工作原理示意图

带电导体在磁场中受安培力：$F = nilB\sin < i, B >$ (3.5)

因为在电机匀强磁场中，<i,B>=90，故公式简化为：$F = nilB$ (3.6)

转子旋转后，运动导线切割磁感线产生反电动势：$E' = nBlv\sin < B, l >$ (3.7)

因为在电机匀强磁场中，<i,B>=90，故公式简化为：$E' = nBlv$ (3.8)

设电机两端电势 E，内阻为 R。电机没有旋转时电流：$i = \frac{E}{R}$ (3.9)

此时电机热功率为：$W_Q = \frac{E^2}{R}$ (3.10)

假设电机的外部负载扭力 f 。

让电机启动的条件是 $F > f$，此时电压满足 $E_{\min} > \frac{fR}{nlB}$ (3.11)

由公式 3.11 以及 3.10 可知，若电机两端电压小于 $\frac{fR}{nlB}$ 时全部的电功率都转化成为热功率，电机十分危险且效率为 0，轻则过热发烫，重则烧毁，所以设计电机控制时应避免电压处于此门限电压之内。

电机运动时，存在反电动势 E'，此时流过电机的电流 $i = \frac{E - E'}{R}$ (3.12)

电机在电动势 E 下达到平稳运行时满足 $F = f$ ，其速度与电势成一次函数关系：

$$v = \frac{1}{nlB}E - \frac{fR}{n^2 l^2 B^2} \quad (3.13)$$

此时电机功率：$W = fv = \frac{f}{nlB}E - \frac{f^2 R}{n^2 l^2 B^2}$ (3.14)

此时效率为：$\eta = 1 - \frac{fR}{EnlB}$ (E> $E_{\min}$) (3.15)

图 3.8 示意了电机两端电压与电机效率的关系，当电压升高时，电机效率也会逐渐提升。

首都师范大学本科毕业设计论文

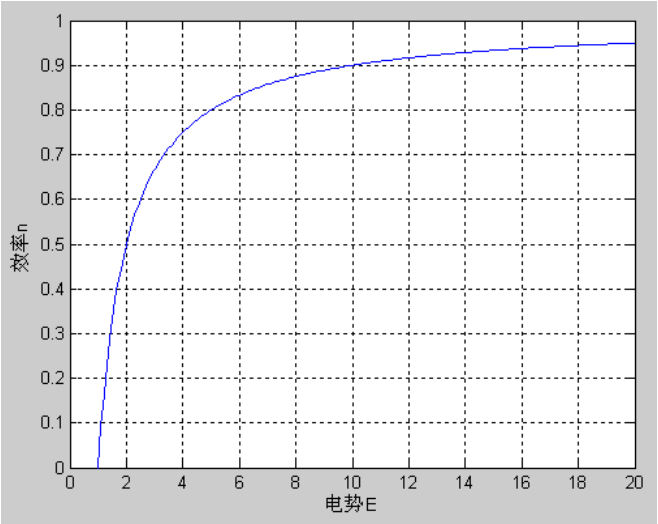


图 3.8　电势 E 与效率 n 关系示意图

平稳运行时的电机热功率 $W_Q = \dfrac{((\dfrac{R-nlB}{R})E-f)^2}{R}$ (3.16)

若电机以速度 v 稳定运行时，改变两端电势为 E_2，由于速度不能突变，随瞬时速度仍然维持在 v，而热功率 $W'_Q = \dfrac{(E_2-\dfrac{nlBE}{R}-f)^2}{R}$ (3.17)

图 3.9 示意了电机两端电压突然改变值引起的电功率。当电机两端瞬时改变电压幅度越大时，瞬时热功率越大。

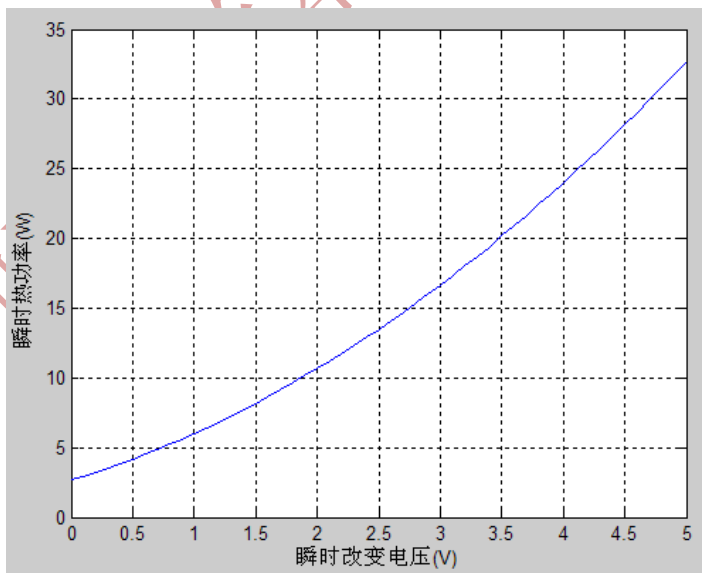


图 3.9　瞬时电压差引起的电热功率改变示意图

从公式 3.17，图 3.9 得出：电机稳定运行时，突然剧烈改变电机两端电压会极大增加热功率，既浪费了电能又损害了电机。图 3.10 灰色部分即表示突然大幅度改变电压后随时间变化的功率情况，其中灰色表示浪费的热功率。

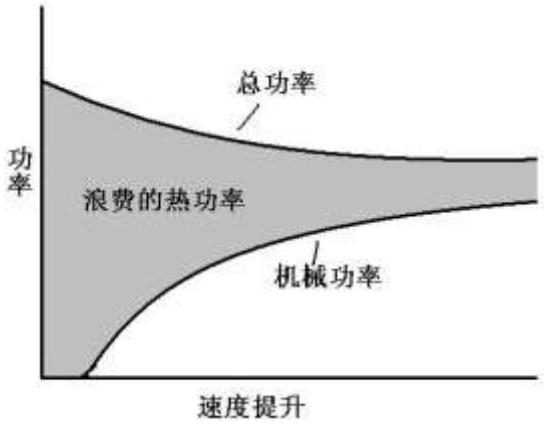


图 3.10　突然改变电压后的功率情况示意图

3.2.3 电机控制改进设计

本文借鉴缓变速思路，结合分析出的电机启动门限电压 E_{min} 设计出一种基于状态机控制的有刷电机定步长缓变速控制模型。相比一般开环有刷电机控制方案有如下特性：

- 更节能：因为考虑到启动门限电压和采用定步长变速，所以更节省电能。
- 电机寿命长：本模型能有效减少电机发热和速度突变以延迟电机寿命。
- 状态机中增加刹车状态：能让电机在紧急情况下迅速停止，防止意外发生。
- 更安全：未在规定时限内收到主机驱动信息则自动停止电机，防止意外发生。

3.2.3.1 启动门限

增加最低启动电压门限 E_{min}。当电机两端电压小于 E_{min} 时电机不转动且电能全部转化成热能浪费掉，并且损害电机。

本文模型中电机在启动或变速过程中不会出现 0-E_{min} 的中间电压值，电机两端电压范围被限定为$\{0\}\cup(E_{min},E_{in}]$以减少不必要的电能浪费。这样的优点有：

- 节能
- 延长电机寿命

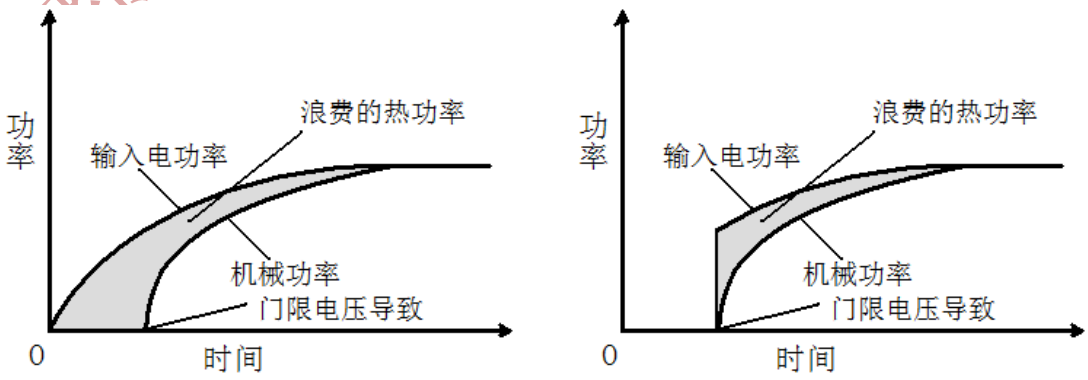


图 3.11　常规模型的电机启动(无门限电压控制)示意图(左) 本文电机启动示意图(右)

沈阳师范大学本科毕业设计论文

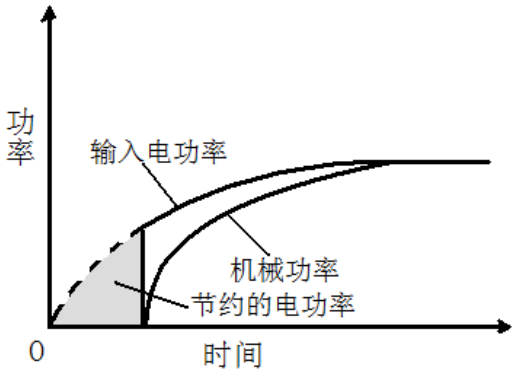


图 3.12　本文门限电压控制节约的电能示意图

图 3.11 可知：常规电机启动方案输入功率从 0 开始随时间缓慢增长，而本文则是从门限电压 E_{min} 开始输入。本文方案节省了图 3.12 阴影部分中的电能。门限电压方案节约了能源，减少了电机发热延长电机使用寿命。

3.2.3.2 定步长调速

采用定步长进行调速控制。需要调速时，每隔固定时间段电机两端电压增加或减少固定值 S。相比传统每隔固定时间段电机两端电压变化 $\frac{\text{目标电压}-\text{当前电压}}{k}$ 的固定份数调速方案，本文改进方案有如下优点：

- 节能：在大幅度调速中能节省更多电能。
- 调速平滑：在大幅度调速中能让电机速度变化平滑。
- 调速快：在小幅度调速中反应快，调速灵敏。

■ 大幅调速情况

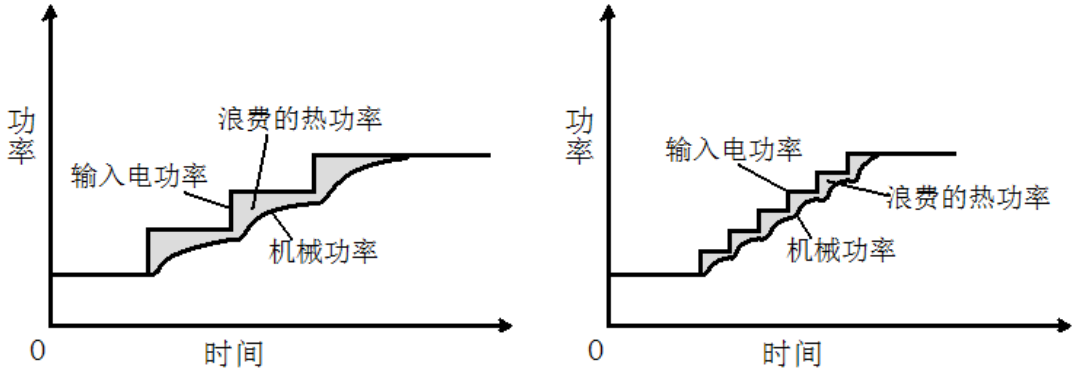


图 3.13　大幅度调速下固定分数调速示意图(左)　本文定步长调速示意图(右)

图 3.13 中，相比左图，右图中灰色部分面积明显要小。说明本文定步长调速方案在应对大幅度调速情况下能节省更多电能，减少电机发热，延长电机寿命。

对比两幅图的机械功率曲线可以看出，右图比左图曲线更加平滑。说明本文定步长调速方案在应对大幅度调速情况下能让电机调速更平滑。

■ 小幅度调速情况

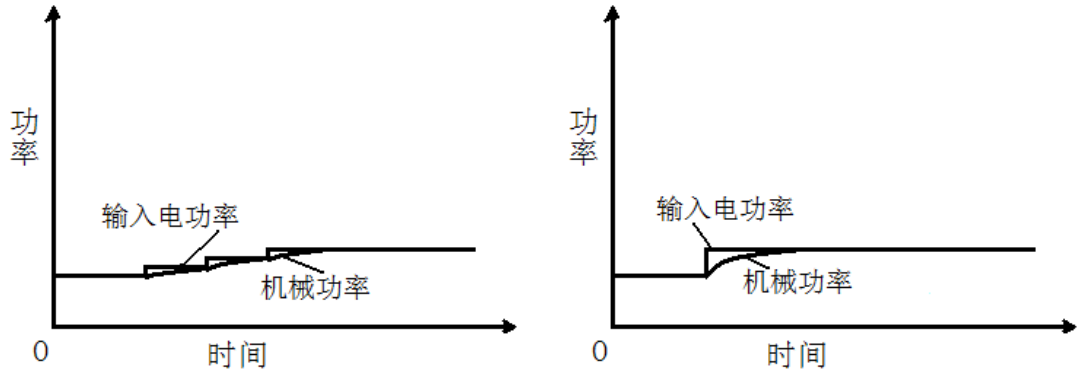


图 3.14 小幅度调速下固定分数调速示意图(左)　本文定步小调速示意图(右)

图 3-14 中在小幅度调速时相比左图，右图所耗时间明显要少，说明在小幅调速时定步长调速响应快。

3.2.3.3　超时控制器

本文控制方案中设定了一个倒计时器，主机需要每隔一定时间向电机控制器发送信号。若超过预定时间而没有信号从主机传来，控制器将让电机进入预定安全状态。

本方案可以避免当主机程序异常导致控制失效，但电机继续运行造成的灾害。

3.2.3.4 电机控制状态机

本文基于 8 种状态的状态机进行电机控制，且加入紧急制动(刹车)状态。该状态机模型的优点有：

- **安全性高**：任何运行状态都能转入紧急制动状态以避免紧急情况造成的伤害。
- **控制精细**：将电机在不同方向时的加减速都区别开，可应对变向调速。

电机控制状态机图如下：

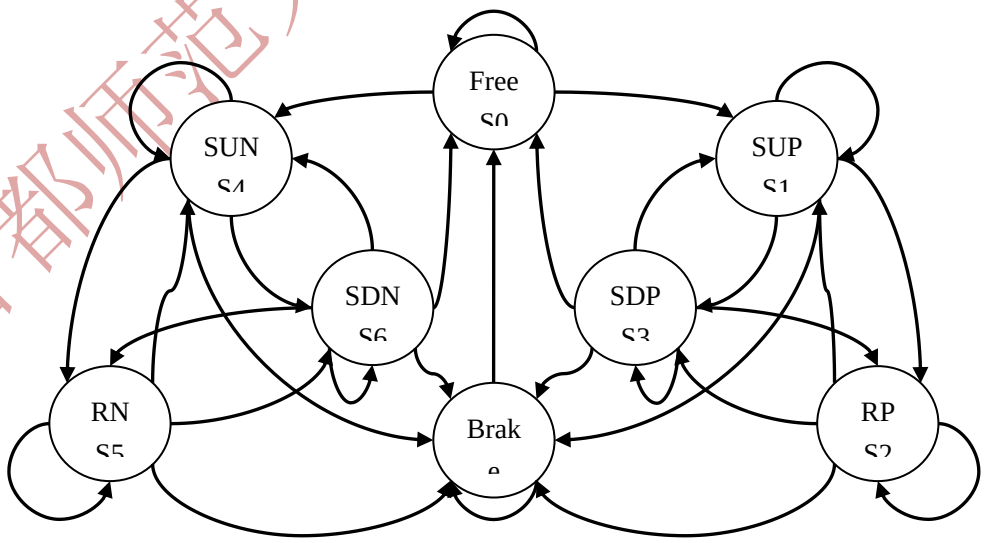


图 3.15　本文电机控制器状态图

其中各个状态和含义如表 3.2 所示：

首都师范大学本科毕业设计论文

表 3. 2　电机控制状态机各状态说明

状态	含义
Free(S0)空闲状态	此状态下电机不转动，待机
SUP(S1)正转加速状态	此状态下电机正转并逐渐加速
RP(S2)正转稳定状态	电机正转稳定运行且无需加速
SDP(S3)正转减速速状态	此状态下电机正转并逐渐减速
SUN(S4)正转加速状态	此状态下电机反转并逐渐加速
RN(S5)正转稳定状态	电机反转稳定运行且无需加速
SDN(S6)正转减速速状态	此状态下电机反转并逐渐减速
Brake(S7)刹车	此时电机处于刹车状态

下边将说明几个标记及其含义。Timer：倒计时器，若 Timer=0 表示发生超时，电机需要停止。PWM_C：当前 PWM 值，在加减速时需要不断改变。PWM_T：目标 PWM 值，即最终要到达的 PWM 值。DIR_C：当前电机运行方向。DIR_T：电机目标运行方向。若此方向与当前电机运行方向相反，电机需要减速然后调整方向在加速。

各个状态直接每隔一个固定的时间间隔 T 进行一次状态切换和检查。各个状态之间转换关系、条件、动作如表 3.3 所示(其中“阈值”指启动门限电压对应的 PWM 值)：

表 3.3　电机控制器状态转移表

源状态	目标状态	条件	动作
S0	S0	PWM_T<=阈值，构不成启动条件	无
S0	S1	PWM_T>阈值且 DIR_T=正且 Timer>0	令 DIR_C=正
S0	S4	PWM_T>阈值且 DIR_T=负且 Timer>0	令 DIR_C=负
S1	S1	PWM_T>阈值且 DIR_T=正 且 Timer>0 且 PWM_C< PWM_T	按不超过定步长加速
S1	S2	PWM_T>阈值且 DIR_T=正 且 Timer>0 且 PWM_C>= PWM_T	令 PWM_C= PWM_T 且进入稳定运行模式
S1	S3	PWM_T<阈值或 DIR_T=负或 Timer=0	无
S1	S7	刹车信号出现	刹车并进入刹车状态
S2	S2	PWM_T>阈值且 PWM_C = PWM_T 且 Timer>0 且 DIR_T=正	无
S2	S1	PWM_T>阈值且 PWM_C < PWM_T 且 Timer>0 且	无

		DIR_T=正	
S2	S3	PWM_T<阈值或 DIR_T=负 或 Timer=0 或 PWM_C > PWM_T	无
S2	S7	刹车信号出现	刹车并进入刹车状态
S3	S3	PWM_T<=阈值或 DIR_T=负 或 Timer=0 或 PWM_C > PWM_T	按不超过定步长减速
S3	S2	PWM_T>阈值且 PWM_C <= PWM_T 且 Timer>0 且 DIR_T=正	令 PWM_C= PWM_T 且进入稳定运行模式
S3	S1	PWM_T>阈值且 PWM_C < PWM_T 且 Timer>0 且 DIR_T=正	无
S3	S0	PWM_C<=阈值或 DIR_T=负或 Timer=0	无
S3	S7	刹车信号出现	刹车并进入刹车状态
S4	S4	PWM_T>阈值且 DIR_T=负 且 Timer>0 且 PWM_C< PWM_T	按不超过定步长加速
S4	S5	PWM_T>阈值且 DIR_T=负 且 Timer>0 且 PWM_C>= PWM_T	令 PWM_C= PWM_T 且进入稳定运行模式
S4	S6	PWM_T<阈值或 DIR_T=正或 Timer=0	无
S4	S7	刹车信号出现	刹车并进入刹车状态
S5	S5	PWM_T>阈值且 PWM_C = PWM_T 且 Timer>0 且 DIR_T=负	无
S5	S4	PWM_T>阈值且 PWM_C < PWM_T 且 Timer>0 且 DIR_T=负	无
S5	S6	PWM_T<阈值或 DIR_T=正 或 Timer=0 或 PWM_C > PWM_T	无
S5	S7	刹车信号出现	刹车并进入刹车状态

首都师范大学本科毕业设计论文

S6	S6	PWM_T<=阈值或 DIR_T=正 或 Timer=0 或 PWM_C > PWM_T	按不超过定步长减速
S6	S5	PWM_T>阈值且 PWM_C<= PWM_T 且 Timer>0 且 DIR_T=负	令 PWM_C= PWM_T 且进入稳定运行模式
S6	S4	PWM_T>阈值且 PWM_C < PWM_T 且 Timer>0 且 DIR_T=负	无
S6	S0	PWM_C<=阈值或 DIR_T=正或 Timer=0	无
S6	S7	刹车信号出现	刹车并进入刹车状态
S7	S7	无	无
S7	S0	收到启动信号	解除刹车状态

3.2.4 电机控制改进设计性能分析

本段将从三方面(整体特性、电机启动、大幅度调速)来比较直接控制、固定份数调速、本文方案这三种方案。

■ 整体特性

表 3.4 三种电机调速方案比较

比较项目	本文方案	固定份数	直接控制
控制方式	开环	开环	开环
节能性	好	较好	差
电机发热	少	较少	较多
变速平滑性	平滑	较为平滑	不平滑
大幅调速用时	较长	较短	短
小幅调速用时	短	较长	短

■ 电机启动

首都师范大学本科毕业设计论文

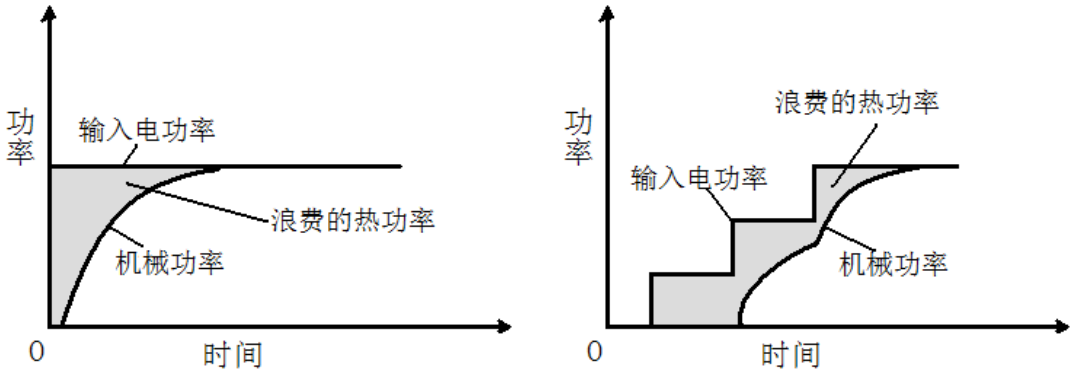


图 3.16　直接控制电机启动示意图(左)　固定份数调速电机启动示意图(右)

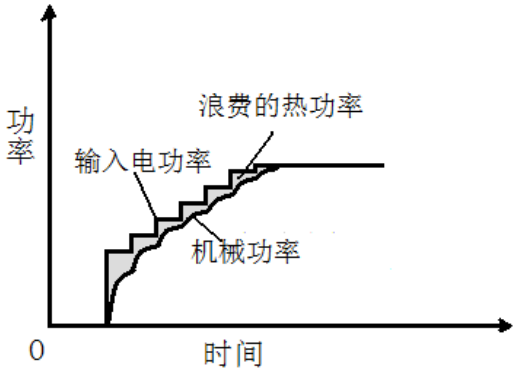


图 3. 17　本文方案电机启动示意图

图 3.16 和图 3.17 中可看出：考虑了启动电压门限和采用定步长调速后，本文电机启动方案效率高，电机调速平滑。

■　大幅度调速

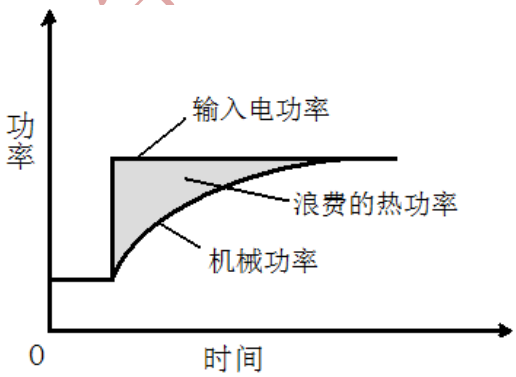


图 3.18　直接控制的大幅度调速示意图

对比图 3.13 和图 3.17 以及图 3.18 可以看出：本文提出的定步长调速方案节省能源且调速平滑。

首都师范 学本科生 设计论文

3.3 硬件防死区时间

基于 H 电桥的电机控制电路中，最大问题就是防止死区时间的出现。出现死区时间轻则电路发热、短路，重则电路、电源烧毁。

3.3.1 死区产生

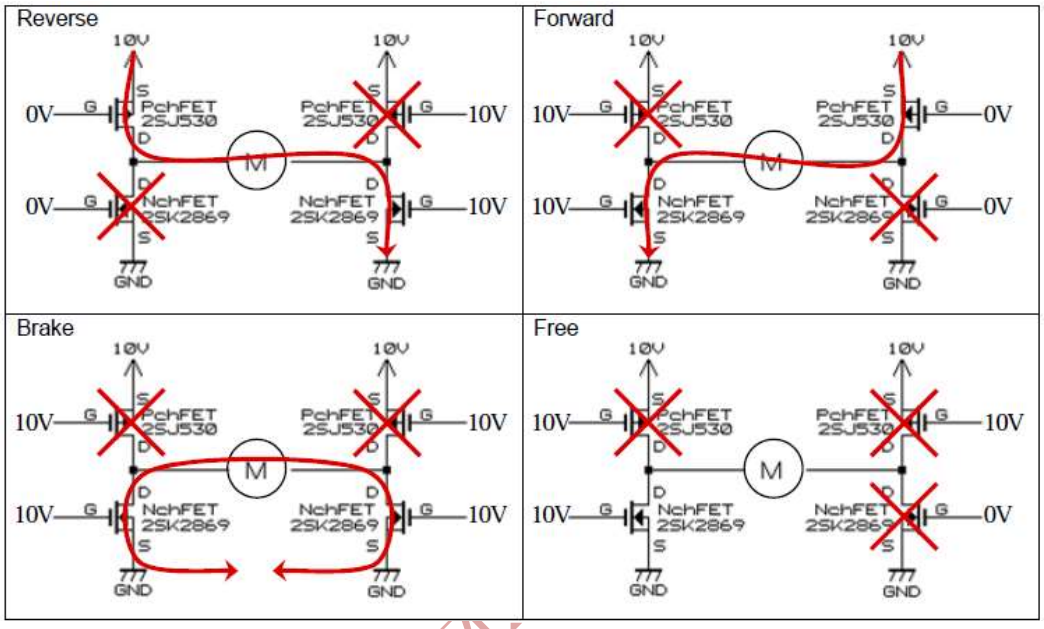


图 3.19　H 前桥电机控制原理 反转(左上) 正转(右上) 制动(坐下) 空闲(右下)

图 3.19 中叉号表示不通电。上下左右不同的开关管处于导通或截止状态，就能改变流入电机的电流方向，控制电机的正反转。

若出现图 3.20 中同一侧导通的情况，则电路处于短路状态。该情况被禁止出现。

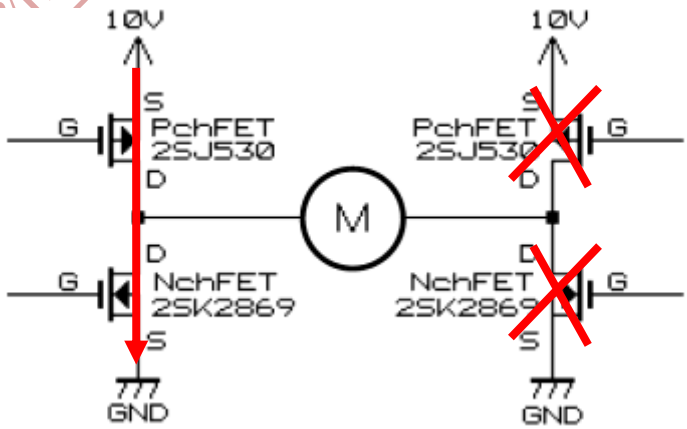


图 3.20　H 电桥一侧导通引起的短路

H 电桥的上下开关管其电气特性参数是不同的，见下表：

表 3.5　本文采用 H 电桥器件电器特性

型号	导通延迟	截止延迟

J530	87ns	200ns
K2869	120ns	225ns

J530 和 K2869 的导通、截止延迟不同，在 H 电桥变化导通、截止状态时会出现如下图 3.21 所示情况：

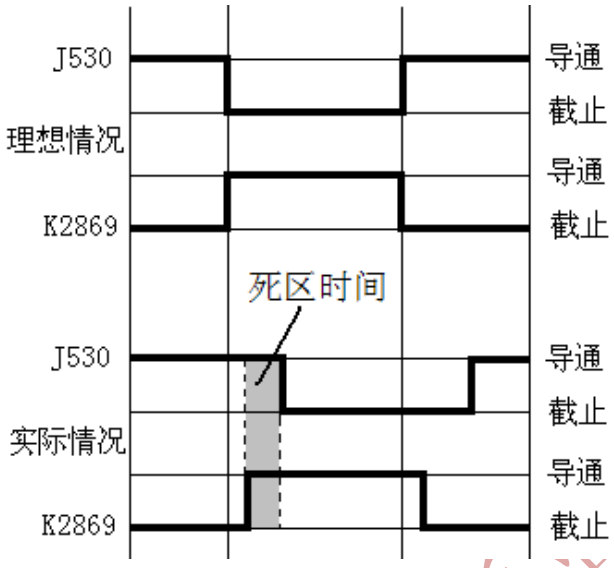


图 3.21　延迟引发的死区

图 3.21 中灰色部分就是由于两个开关管导通、截止延迟不同而出现两开关管均为导通状态。此时电路短路，电流极大，时间一长会烧坏电路。

3.3.2 常用的专用芯片死区解决方案

常用的死区解决方案是采用专用多路带死区延迟的互补 PWM 输出控制器或含此外设的微控制器如 STM32 等专为电机控制设计的微控制器。

3.3.3 硬件电路死区解决改进设计

本智能车设计时采用了低成本的 8051 芯片，并不含有带死区延迟的互补 PWM 输出。故本文设计了一种使用施密特触发反相器延迟和逻辑电路组成的 H 电桥防死区电路。

本设计优点有：

- 延迟性能好：延迟设定在 40us 左右，远远大于死区时间 225ns。
- 安全性高：采用硬件组合逻辑电路保证 H 电桥不会出现短路状态。
- 紧急制动：设计了应对紧急情况时的刹车功能

■ 总电路图

首都师范大学本科毕业论文

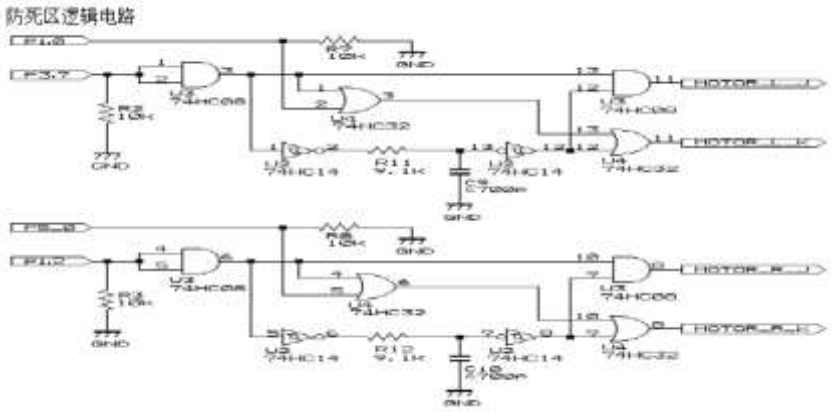


图 3.22　防死区延迟、逻辑电路图

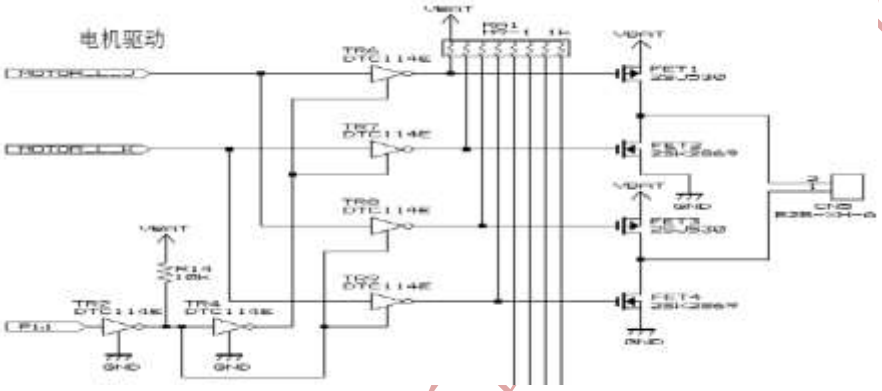


图 3. 23　H 电桥电机驱动电路图

■　延迟器

本文使用器件 74HC14(施密特触发反相器)和 RC 电路实现延迟。(如下图 3.24 所示)

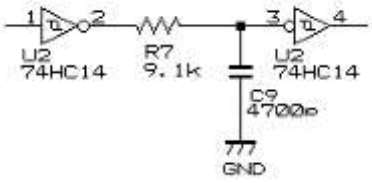


图 3.24　延迟器部分电路图

其中的 RC 部分，其时间常数 $\tau = RC = 9.1K \times 4.7 \times 10^{-9} = 42.77\mu s$　　(3.18)

使用 74HC14 的原因是要将缓慢变化的电压变为数字逻辑。

本电路部分延迟达 42.77us，远远超过了 225ns 的器件延迟，所以能过达到效果。所以能做到等 H 电桥上半开关管完全关闭后在打开下半开关管，避免了死区的产生。

■　逻辑电路

本模块使用与门、或门电路保证 H 电桥的开关管仅处于图 3.19 中的有效状态，避免图 3.20 中短路的发生。

表 3.6 为电路逻辑真值表：

表 3.6　电机控制模块组合逻辑电路真值表

P1.0 刹车	P3.7 速度	P1.1 方向	FET1	FET2	FET3	FET4	H 电桥
-	1	0	截止	导通	导通	截止	正转
-	1	1	导通	截止	截止	导通	反转
0	0	-	截止	导通	截止	导通	刹车
1	0	0	截止	导通	截止	截止	自由
1	0	1	截止	截止	截止	导通	自由

首都师范大学本科毕业设计论文

第 4 章　系统实现

本章从选型开始，到电路设计、软件设计，直至最终完成系统。本文将说明如下部分的设计和实现：

- 微控制器、处理器选型
- 驱动库编写
- 本文提出的多处理器通信协议栈实现
- 本文提出的电机驱动方案实现
- 电机控制器的编程实现
- 伺服舵机控制器的编程实现
- 电机控制电路的实现
- 私服舵机控制电路的实现
- 电源系统的实现
- 智能车车身的实现

(*本智能车无线视频系统、传感器系统、无线系统由同组其他同学完成)

4.1 微控制器、处理器选型

微控制器、处理器选型关系到系统性能和价格，不同的微控制器有不同的外设，需要依据不同模块具体所需功能进行选型。

4.1.1 电机、伺服舵机控制器选型

- 功能需求分析：

  至少两路 PWM 信号输出功能，PWM 用于控制电机调速和伺服角度

  至少 6 个强电流推挽输出通用 I/O 管脚，用于控制电机运行

  至少 1 个全双工 UART 收发器，用于总线通讯

  使用外置高精度晶振作为时钟源，保证通讯质量

  体积尽可能小，节约电路板空间

  指令执行速度快，能高效执行数据校验以及控制功能

- 经过仔细比较后选用芯片：STC12C5052。其特点有[24]：

首都师范大学本科毕业设计论文

- 2 路内置 PWM 发生器
- 1T 指令模式：CPU 速度是普通 8051 体系的 12 倍
- 采用 8051 内核：技术成熟开发容易
- I/O 口支持强推挽上拉输出
- 内置全双工 UART 收发器
- 体积尺寸小：仅为 20 管脚
- 可靠性高：内置 MAX810 复位电路，硬件看门狗，RC 振荡器
- 成本低：单片成本不足￥5.00

■ 芯片管脚图及最小系统事例[24](引自官方手册)

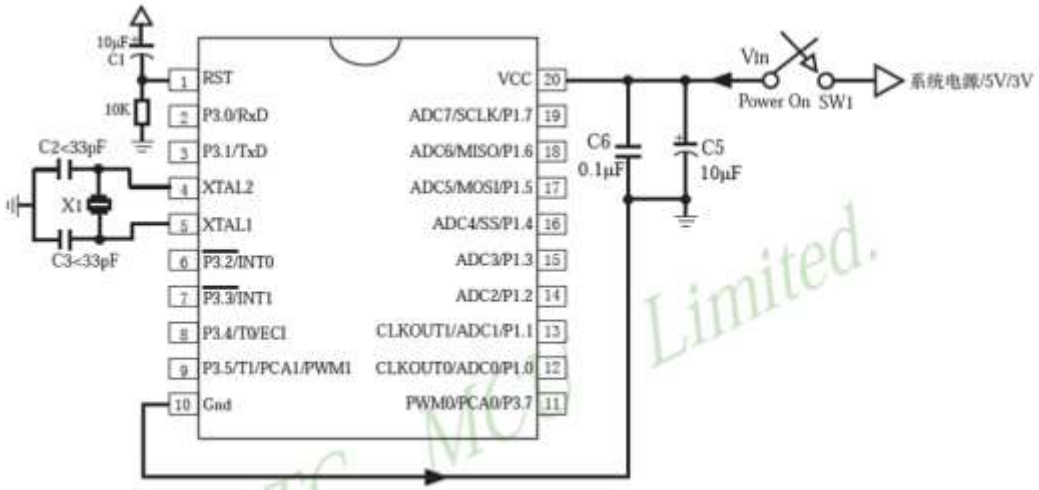


图 4. 1　STC12C5052 管脚和最小系统电路图

本方案使用外部晶振 X1=22.1184MHz。由于 STC12C5052 有 1T 指令模式，其时钟相当于普通 8051 结构的 12 倍，并且缩短了许多指令的执行周期。故其指令执行速度能达到 17MIPS。

该芯片有 5KB 的程序存储空间，256Byte 的内存，并有内部 EEPROM 可以存储数据。该芯片工作温度范围宽泛-40~85 度。有 15 个普通 I/O 口。

4.1.2　主处理器选型

■ 功能需求分析：

- 运算性能强
- 至少 1 个全双工 UART 收发器，用于总线通讯
- 使用外置高精度晶振作为时钟源，保证通讯质量
- 芯片可用资源大

- 可靠性高

■ 经过分析比较后主控制器选用芯片：STM32F103C8T6。其特点有[25]：

- 内置硬件乘法、除法器，运算速度快
- 使用高效 ARM-CortexM3 的 32 位 RISC 内核，支持多级流水
- 时钟频率可达 72MHZ
- 系统资源大：64KB 程序存储空间，20KB RAM 空间
- 可靠性高：双看门狗，双时钟
- 通信接口丰富：有多路 SUART、SPI、IIC 等通讯接口

■ 芯片管脚图

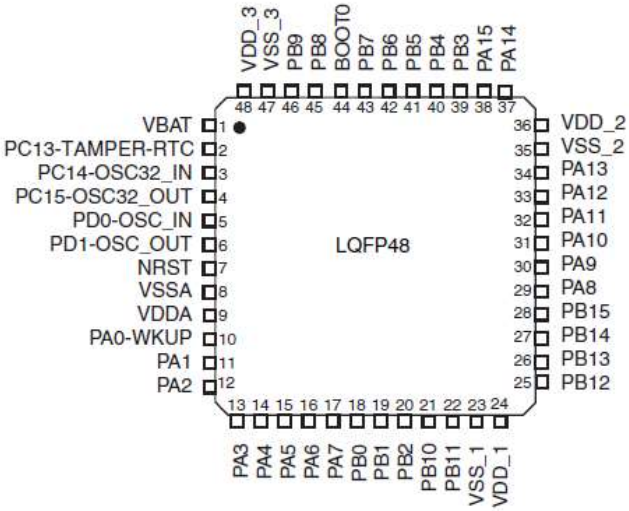


图 4- 2　STM32F103C8T6 芯片管脚图

4.2 驱动库

驱动库中驱动函数包含了大量关于芯片设备寄存器的底层操作。通过驱动库可以将繁琐的硬件底层细节和计算细节屏蔽，让后续开发和协议栈移植变得十分容易。

本文基于 STC12 系列芯片编写了如下模块的驱动库：

■ PCA/PWM 捕获/脉宽调制模块驱动

该驱动库特点：

- 可按通道设定子模块工作模式，配置模式更方便
- 可按通道设定并自动完成计算 PWM 输出有关参数
- 可自动计算 16 位捕获值，返回参数兼容 C 标准
- 可设定不同通道的触发模式

- Timer 定时器、计时器模块驱动

  该驱动库特点：

  - 能同步启动/停止 Timer0 和 Timer1 通道
  - 能按通道设定定时器工作模式，且更方便
  - 按照频率自动计算定时器值，免除手工计算的麻烦
- UART 通用串行通讯接口驱动
  - 设定 UART 串口工作模式更方便
  - 列出常用标准波特率方便查询
  - 自动计算波特率对应时钟，免除手工计算的烦恼
  - 快速高效：大量使用宏定义，编译后执行效率高，符合 C 标准
- ADC 模数转换器驱动
  - 自动计算 ADC 的 10 位返回值，并返回 C 标准的 short 数据
  - 能设定 ADC 的采样速率
  - 能查询 ADC 采样标记

4.3 多处理器通信协议栈实现

编程实现了本文提出的分布式多处理器通讯协议栈

4.3.1 协议栈程序特性

本文实现的多处理器通讯协议栈有如下特性：

- 多缓冲区：接受和发送均可配置多缓冲区，每个缓冲区可配置长度
- 可裁剪：CRC 校验等极为耗费计算资源的部分可以剪裁掉
- 灵活配置：多种系统参数均可配置，可灵活选择主从模式
- 高效性：部分函数使用宏定义，可以提高执行效率
- 可移植：所有与硬件相关函数均用宏定义，移植时方便替换
- 可靠性：基于状态该机、超时倒计时器，保证正常运行
- 轻量级：全功能编译仅占 0.75KB，精简版<500B

4.3.2 协议栈功能结构框图

协议栈功能结构框图讲述了程序间各个部分和资源的相互关系和调度模型。(如下图

首都师范大学本科毕业设计论文

4.3,4.4 所示)：

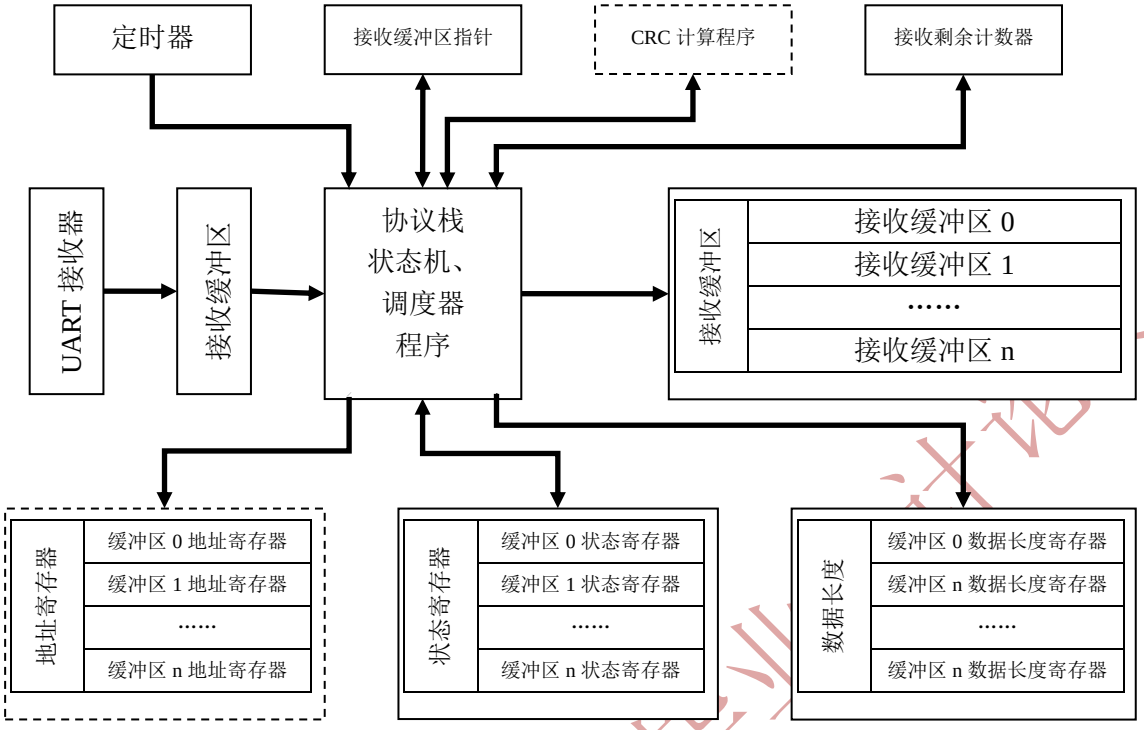


图 4.3　协议栈接收模块程序功能结构框图

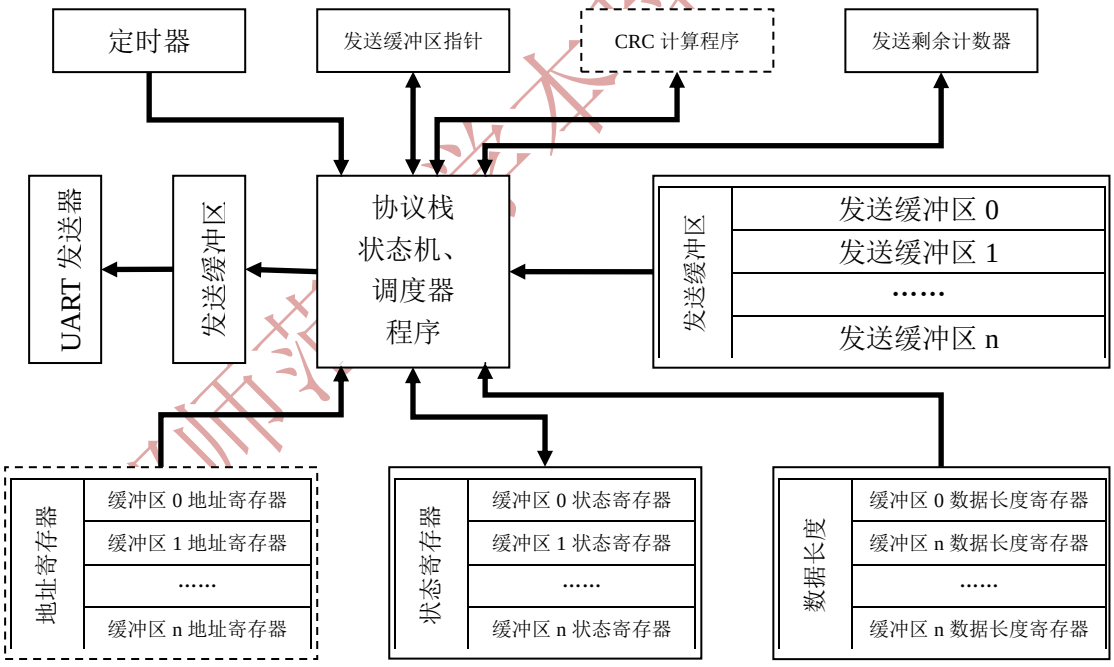


图 4. 4　协议栈发送模块程序功能结构框图

图 4.3,4.4 中的各个模块功能如下：

协议栈状态机、调度器程序：该模块严格按照状态机执行，对外部事件(接收数据/发送完成/定时器超时)进行状态转移，并做出相应动作调用相应程序、资源。调度程序中含有缓冲池的相关信息和缓冲池查询程序。

定时器：周期性为协议栈状态机、调度器程序模块提供信号，用于超时判断。

接收/发送缓冲区指针：用来指向存放/发送的数据地址，每存储/读取一个字节后递增，使用指针能提高程序效率。

CRC 计算程序：可依据需求裁剪(在对可靠性要求不高的情况下可以裁剪)，用于计算 CRC 校验码的程序段。

接收/发送剩余计数器：标记还剩多少字节未接收/发送。

UART 接收/发送器：用于总线通讯中接收/发送数据的硬件外设。

接收/发送缓冲区：用于缓存刚接收/待发送的数据。

接收/发送缓冲池：用来存放接收到/待发送的数据。由 n 个(可修改)长度为 x(可修改)的连续字节构成的二维数组。当接收数据/发送数据时，协议栈状态机、调度器程序会自动搜索可用缓冲区并执行动作。

地址寄存器：主机程序才有。用于记录接收/发送的从机地址。

数据长度：接收到的/待发送的数据长度。

状态寄存器：用于存储接收/发送缓冲区的状态。状态见表 4.1 及表 4.2：

表 4.1　接收缓冲区状态及含义

名称	值	含义
EMPTY	0x01	接收缓冲区为空,即可以将接收的数据存入其中
RECEIVING	0x02	接收缓冲区正在被使用,处于接收状态
COMPLETED	0x04	接收缓冲区接收完成,包括CRC校验
TIMEOUT	0x08	接收缓冲区超时错误,即没有在指定时间间隙内连续完成传输
CRCERROR	0x10	接收缓冲区CRC校验错误
RECEIVECOUNTERROR	0x20	接收缓冲区溢出

表 4.2　发送缓冲区状态及含义

名称	值	含义
EMPTY	0x01	发送缓冲区为空,即程序可以写入
SEDING	0x02	发送缓冲区正在使用,处于发送状态
COMPLETED	0x04	发送缓冲区发送完成,包括CRC校验发送
TIMEOUT	0x08	发送缓冲区发送超时,未能在有效时间内发送完全部数据

4.4　电机驱动方案实现

编程实现了本文提出的电机驱动方案。

首都师范大学本科毕业设计论文

4.4.1 电机驱动方案程序特性

本文实现的电机驱动方案程序有如下特性：

- 灵活配置：门限 PWM，定步长 PWM 等参数可依据不同情况配置
- 系统资源少：占用少量 RAM 就可完成控制
- 接口简单：使用少量函数便可完成电机控制
- 可刹车：有专门的紧急制动(刹车)函数，可在紧急情况时制动。
- 可同步启动电机
- 有超时倒计时器

4.4.2 程序功能结构图

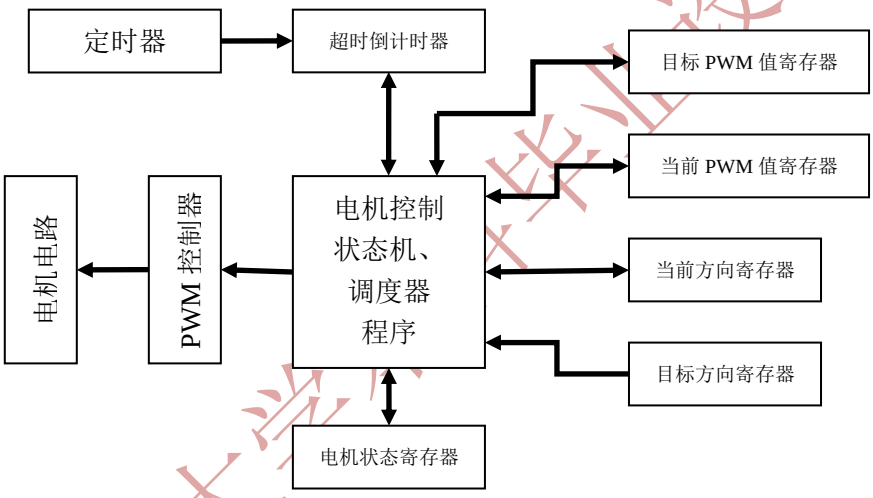


图 4.5 电机驱动程序功能结构框图

上图中各个模块功能如下：

电机控制状态机、调度器程序：该模块严格按照状态机执行，对外部事件(定时器超时、刹车信号)进行状态转移，并做出相应动作调用相应程序、资源。

定时器：周期性提供信号，使超时倒计时器减一。

超时倒计时器：每次定时器倒计时结束后都会减一，当减为 0 时关闭电机。可以调用函数重新装载倒计时器值。

目标 PWM 值寄存器：设定的 PWM 目标值，是状态转移的重要依据之一。

当前 PWM 值寄存器：当前向电机输出的 PWM 值，是状态转移的重要依据之一。

当前方向寄存器：当前电机运转方向，是状态转移的重要依据之一。

目标方向寄存器：设定的电机运转方向，是状态转移的重要依据之一。

电机状态寄存器：用于存储当前电机状态，其值和含义如表 4-3 及表 4-4 所示：

表 4.3　电机状态及含义

名称	值	含义
DIR_P	0x01	电机运行在正方向
DIR_N	0x02	电机运行在负方向
RUN	0x04	电机处于运行状态
BRAKE	0x08	电机处于刹车状态
TIMEUP	0x10	电机控制器倒计时器超时
ERROR	0x80	电机运行出现异常

4.5　电机控制器的编程实现

电机控制器程序结构图(如图 4-6)：

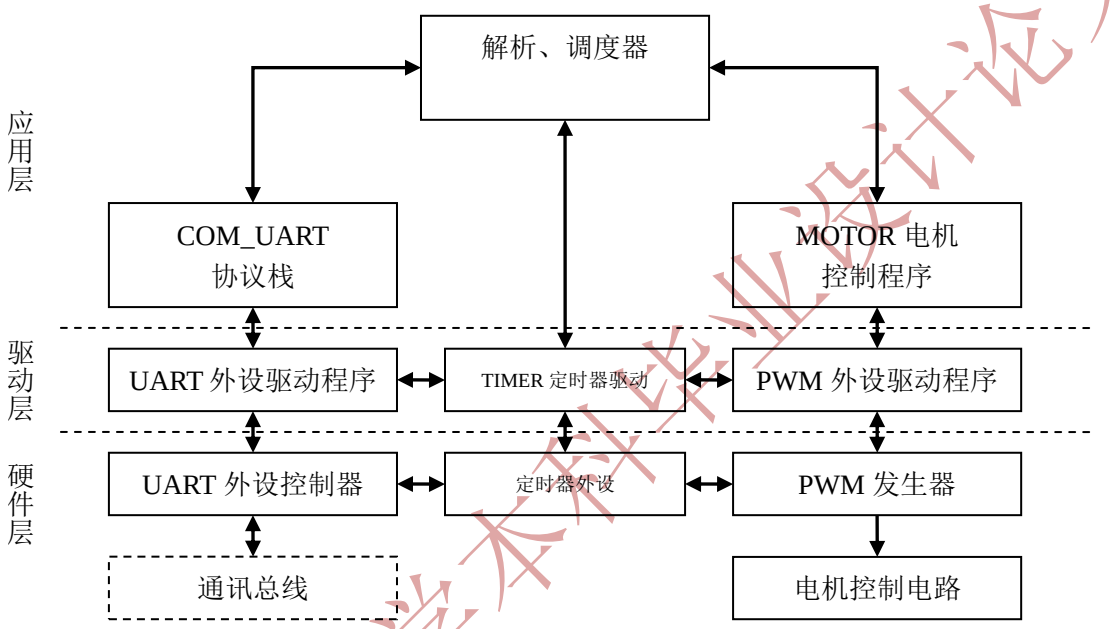


图 4.6　电机控制器程序结构图

采用分层模型编程，分为硬件层、驱动层、应用层。其中每一层对上屏蔽底层细节，控制着下层所有的细节和具体操作。其中驱动层是之前编写的硬件寄存器操作相关函数。

本编程中，将所有模块链接起来，并通过电机控制模块解析、调度器进行管理。当从 COM_UART 协议栈中收到数据包后，解析、调度器分析数据包，并调动程序、资源以执行指令。必要的话通过 COM_UART 向主机发送数据包。

4.6　伺服舵机控制器的编程实现

伺服舵机控制器程序结构图(如图 4-7)：

首都师范大学本科毕业设计论文

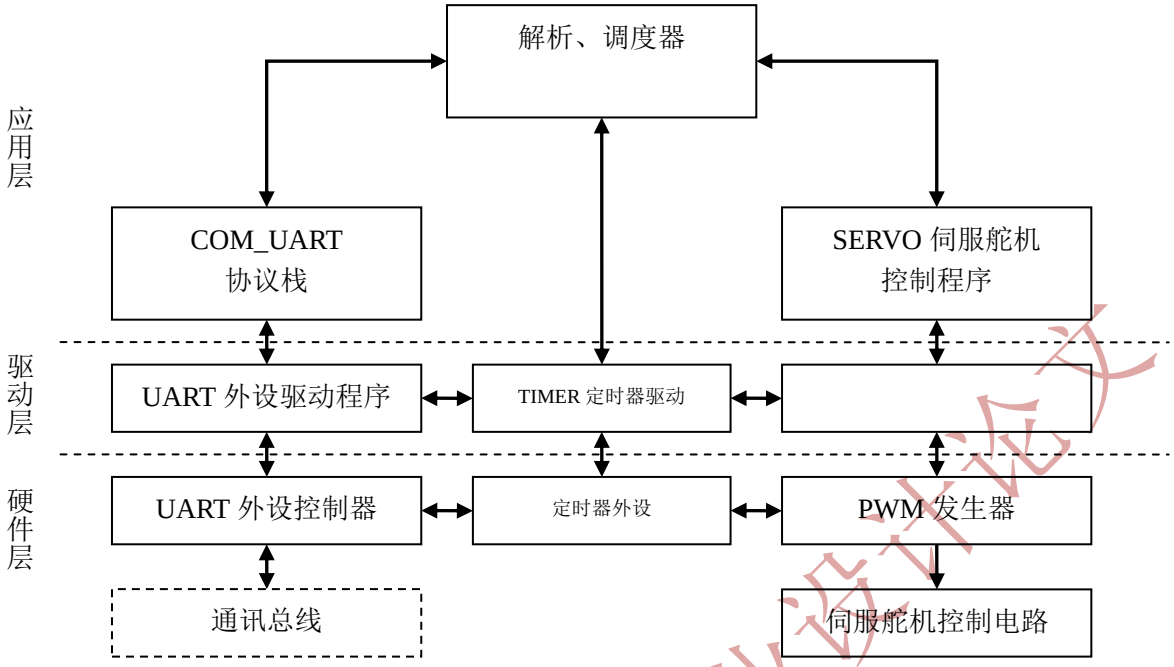


图 4.7　伺服舵机控制器程序结构图

编程思路同 4.5 电机控制器的编程实现类似。将应用层的电机控制程序模块换为伺服舵机控制程序模块，并针对伺服舵机控制模块指令格式重写了解析器和调度器。

4.7　电机控制电路的实现

依照第 3 章中的电路原理图设计出的电路 PCB 图：

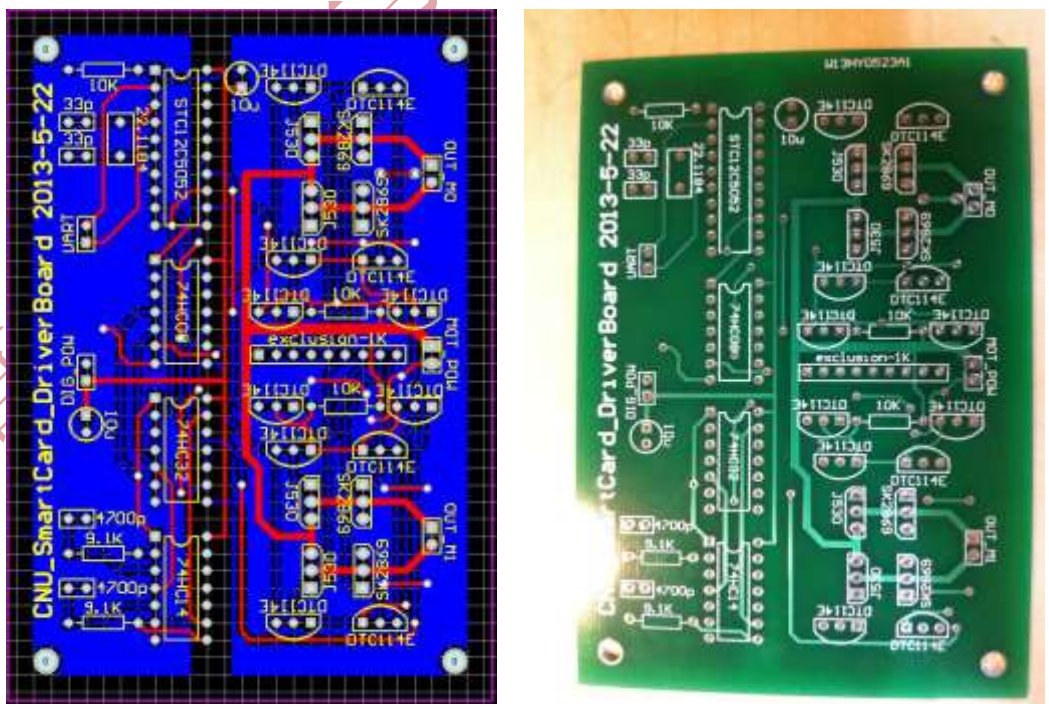


图 4.8　电机控制模块 PCB 图(左)　PCB 实物图片(右)



图 4.9　焊接后的电机控制模块实物图

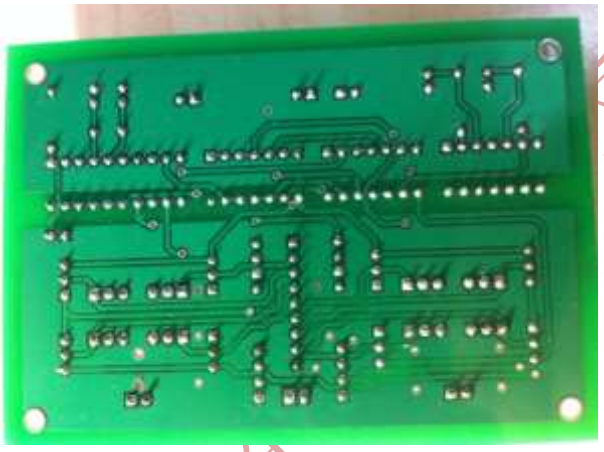


图 4.10　电路板实物-背面

4.8 私服舵机控制电路的实现

由于伺服舵机与微处理器连线简单，此处不再附上电路原理图。焊接好的私服舵机控制电路和部分电源系统电路(焊在同一电路板上)：

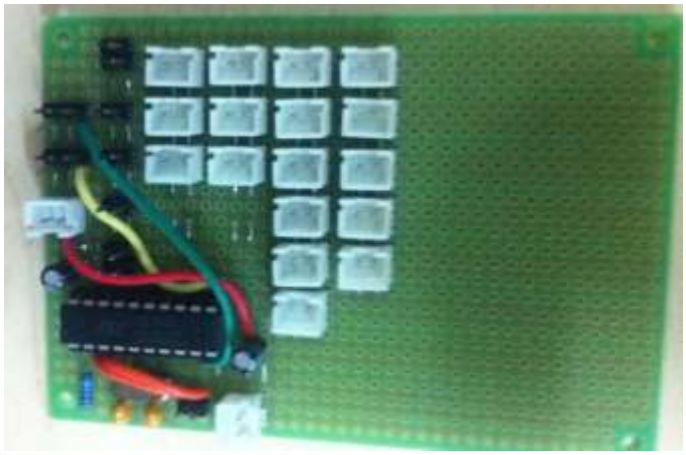


图 4.11　私服舵机控制电路实物图

首都师范大学 设计论文

4.9 电源系统的实现

本文先后使用过线性稳压电源和开关稳压电源，并最终确定使用开关稳压电源。因为其转换效率高，适合大功率用电。在数字电源等小功率且需要精密稳压的地方使用线性稳压电源。

本智能车使用锂离子集合物电池作为电源，其两端有效电压范围 15V~18.8V，电容量达 2Ah，放电速率达 25c。

本智能车各模块需要电源电压如下：

- 12V：视频无线传输系统
- 6V：电机驱动系统
- 5V(功率)：伺服舵机控制系统
- 5V(数字电源)：所有微控制器部分和传感器系统
- 3.3V(数字电源)：无线传输系统

4.9.1 线性稳压电源

使用线性稳压电源有如下优点[26]：

- 电源稳定度及负载稳定度极高
- 输出波纹电压较小
- 瞬态响应速度快
- 线路结构简单、可靠性高、故障率低、理解和维修起来简单容易
- 没有开关干扰,隔离性能好

其缺点如下：

- 功耗大、效率低(其效率一般只有 45%左右)
- 体积大、重量重(不能微小型化)
- 必须有较大容量的滤波电容

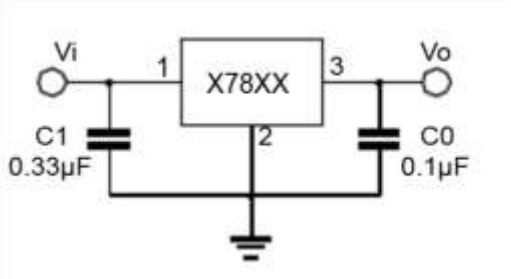


图 4.12　线性稳压电源电路连接图

首都师范大学本科毕业设计论文

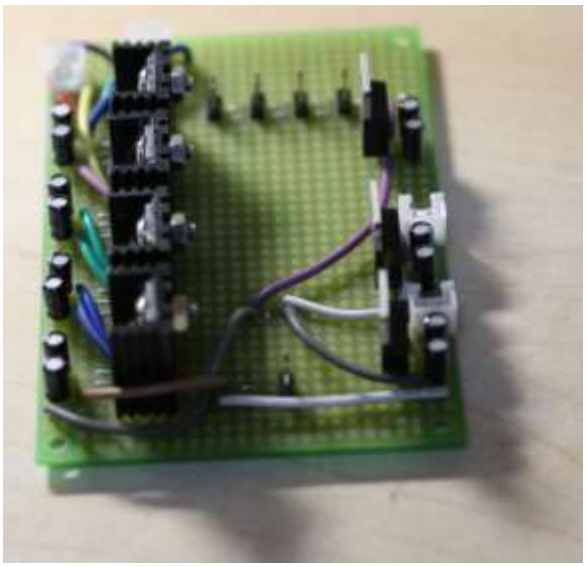


图 4.13　本智能车曾采用的线性稳压电源实物图

4.9.2　开关稳压电源

使用开关电源有如下优点：

- 功耗小、效率高
- 体积小、重量轻
- 稳压范围宽
- 滤波效率高

其缺点有：

- 开关干扰较为严重

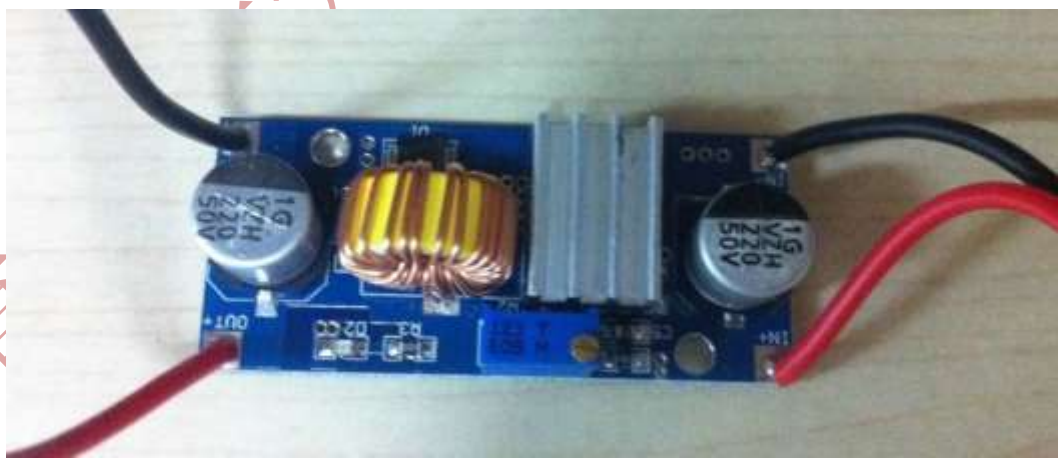


图 4.14　本文采用的开关稳压电源模块

省大学本科毕业设计论文

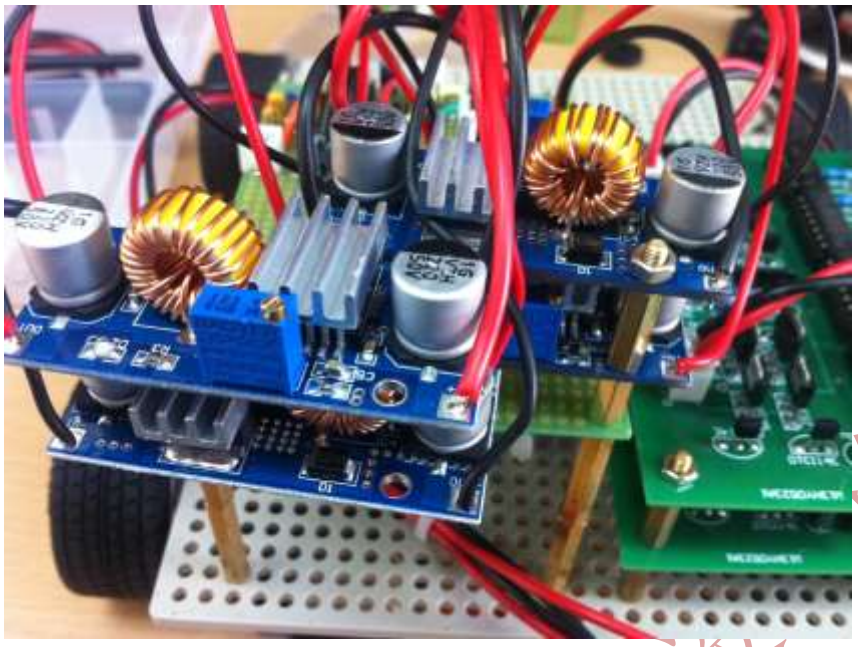


图 4.15　本文最终电源系统

4.10　智能车车身的实现

本小节将说明智能车的车身如何制作、最终系统如何组装到一起。

4.10.1 电机驱动装置

电机使用的是 260 型号的有刷直流电机。齿轮箱使用的是 Tamiya 72002 的高扭力轮组。(如图 4.16、图 4.17)

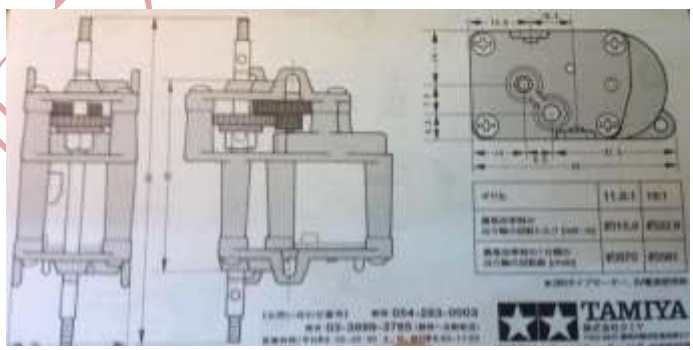


图 4.16 驱动装置原理图

首都师范大学本科毕业设计论文



图 4.17　驱动装置实物图

4.10.2　行动方式

本车采用四轮驱动行走，其优点是输出功率大，可以原地转弯。(如图 4.18 所示)：

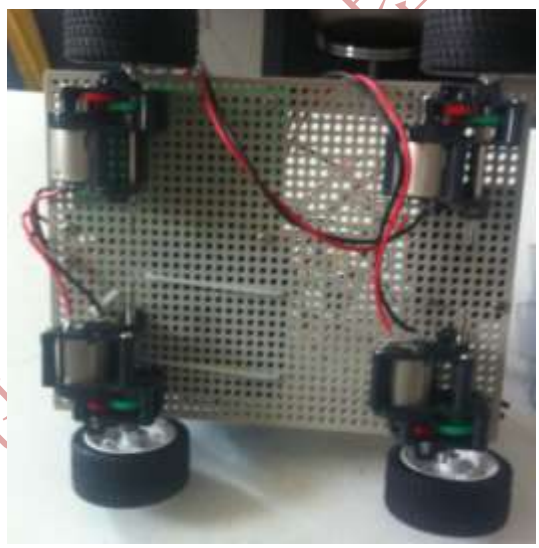


图 4.18　车身四轮驱动

4.10.3 智能车成品实物

将所有模块系统组装后，本智能车完成图如下：

首都师范 毕业设计论文

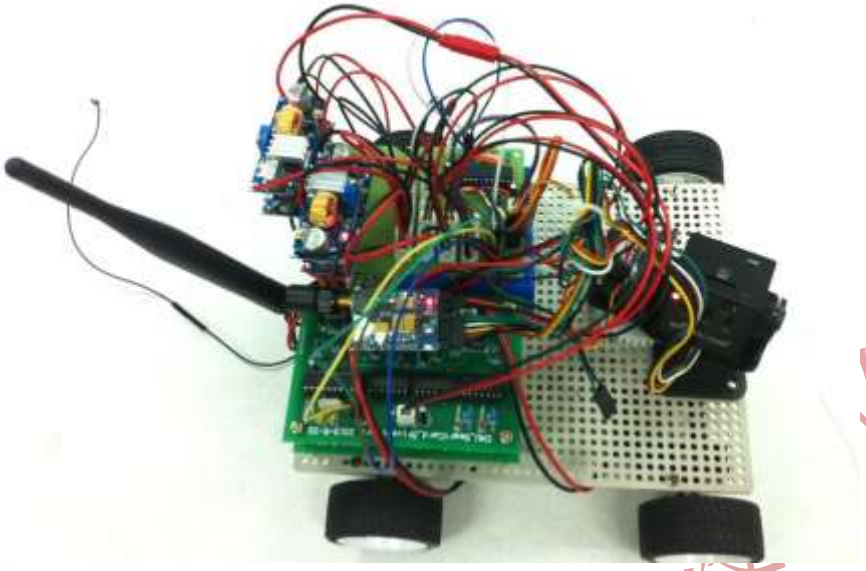


图 4.19　智能车成品实物图

首都师范大学本科毕业设计论文

第 5 章　性能结果评估

本章说明了本设计最终的性能，并与原始系统设计指标相比较。

5.1 基本指标实现总结

- 负荷量：经检测本智能车有效负荷量>3.7kg 并且在此基础上仍能保持最高速度的 60%。
- 最高速度：经测定在负荷量为 2kg 左右时，最高速度可达 5m/s。
- 电机输出功率：一部电机工作在 6V，其内阻为 0.5 欧姆，带载工作电流为 1.5A。此时输出功率为 7.875W，四部电机共能输出 31.5W 功率。
- 电池容量：本智能车设计使用有效电压为 15V-18.8V 的 2AH 锂离子聚合物电池，其有效电容量大于 108kJ。
- 待机功率：当关闭全部设备(包括视频传输、摄像机、电机等)其耗电量小于 8W。
- 最大功率：开启视频传输设备并且电机全部运行时，智能车耗电小于 90W。

经分析，本智能车设计符合系统指标要求。

5.2 功能指标实现总结

1.本智能车搭载了摄像头，该摄像机能够实时传回图像，同时将图像记录在其内存卡中。摄像机带有云台，能 180 度旋转。开阔地测量，距离能到 350m 以上(与天线尺寸有关)。

2.本智能车电机控制器采用状态机模型，能够实现电机的正反转和调速。提出了门限启动电压的概念，并采用定步长调速法，从而节省了更多能源，延长了电机寿命。

3.本智能车构架具有可裁剪的特性，智能车系统构架支持传感器、电机、无线等模块的灵活添加与删除。添加和删除模块不会对智能车主电路造成过大影响。

4.改进了一种分布式处理器通讯方案，使其适合中端级别应用。该方案具有效率高、可靠性较强、执行效率高、接口简便等特性。能保证各个主从处理器间的相互通讯。

5.3 各模块性能总结

- 开关电源模块

首都师范大学本科毕业设计论文

本模块体积小巧效率高。输出功率可达 50W，效率高达 86%以上，比线性稳压电源方案的 40%要节约能源。

- 改进的多处理器通信协议栈程序

  代码小巧，资源需求小。在 8051 体系结构处理器上，精简版本(4 缓冲区，每缓冲区 8 字节)仅占不到 500Byte 程序空间，不超过 48ByteRAM 空间。

  执行效率高：禁用 CRC 校验。接收/发送每字节处理程序需执行不到 400 机器周期。启用 CRC 后不到 600 机器周期。

  通讯效率高：在 28800bps 下，每秒可传送 400 数据包(数据包长度为 6Byte 时)。

- 改进的电机控制程序及模块

  程序高效：2 台电机的控制模块程序仅为 1.5KB，占用 RAM 空间小于 16Byte。

  有门限电压启动。

  调速平滑。

  有效防止死区时间的出现。

- 伺服舵机控制程序及模块

  能按角度、PWM 设定，并能自动换算。

  实现有上下限 PWM 设定，防止伺服舵机损坏。

  能实现水平 270 度，垂直 120 度视野覆盖。

首都师范大学本科毕业设计论文

第 6 章　总结

在经过不断的努力和摸索后，终于实现了原定设计的智能车系统。在不断的实践中，我不断学习更多理论知识并将其应用，学会了因地制宜的分析和解决问题。知道了实际工程中对待问题的原则和方法。在设计本系统的过程中既有快乐也有艰辛，付出了许多汗水和努力同时也收获知识和经验，并主要完成了以下工作：

1. 智能车系统的需求指标分析。
2. 智能车系统的构架与结构提出和改进
3. 改进了多处理器通讯的轻量级协议栈
4. 改进了电机控制模型，使其更节能。
5. 改进了硬件电路以防止死区时间的产生。
6. 编程实现了上述协议栈、模型。
7. 设计并制作上述模块的实物电路。
8. 构建智能车系统。
9. 模块间整合、调试和测试。

在进行各项任务时，时常会碰到许多问题。在解决问题中学会了查阅资料、官方手册资料等重要参考信息，并学会了实际排查电路故障、设计电路板的相关知识与技能经验。

首都师范大学本科毕业设计论文

参考文献

[1] 王国胤,陈乔松,王进.智能车技术探讨.[J].计算机科学,2012,39,1-8.
[2] 张伟.无人操作智能车的运动控制系统研究.[D].西安:西安工业大学,2012
[3] 刘大维.基于嵌入式单片机的智能车控制系统研究.[D].华南理工大学,2012
[4] 戴意愿.智能车远程调度系统设计与研究.[D].湖南大学,2012
[5] 司马光.灵活高效的玉兔号月球车.[N].国际太空.420
[6] 康孝胜,王冰.基于 MC9S12 的智能车系统设计.[J].信息技术,2010.2,19-22,27
[7] 林涛,马腾炜,陈亚勋,赵秀峰,马鑫.基于 MCU 的自动寻迹智能车控制系统的设计.[J].自动化与仪表.2012.11,1-4
[8] 张兆惠.基于微处理器的智能车控制系统开发与研究.[D].大连理工大学.2008
[9] 丁芳,贾翔宇,李科伟,杨明.模糊算法在智能车控制中的应用.[J].中国民航大学学报.2009.2,27 卷 1 期,27-30
[10] 尹占芳,张莹.基于 CAN 总线的智能车的设计.[J].微型机与应用.2012,31 卷 9 期,24-25,28
[11] 陈曦.CAN 总线实时性和可靠性若干问题的研究.[D].天津大学电气与自动化工程学院.2010
[12] 满庆丰.CAN 总线的应用与发展.[J].电子技术应用.1994-12. 2-4
[13] 宋秀兰,吴晓波.多处理器通信机制设计.[J].浙江工业大学学报.2010-8,38 卷 4 期.426-429.
[14] 梁阿磊,赵玉源,钟凯,白英彩,韩江洪.基于 LONWORKS 网络的多处理器智能节点设计.[J].计算机研究与发展.2000-4,37 卷 4 期.453-457
[15] 陈远.一种嵌入式通用多处理器通信管理单元设计.[J].电讯技术.2008-7,48 卷 7 期.34-38
[16] 谷畅霞,李天阳,陶建中.支持多处理器通信的同/异步串口的设计.[J].微电子学.2012-2,42 卷 1 期.107-110,114
[17] http://zh.wikipedia.org/wiki/CRC32 .[DB].引自维基百科.2014-4
[18] 唐金海.CAN 总线分析及应用编程.[D].内蒙古大学.2012
[19] 汪瞳,刘玉明,杨楚平,朱齐丹. CAN 总线协议实现方法研究[J].应用科技.2007-7,34 卷 7 期.54-57
[20] 张平安.16 位循环冗余校验码(CRC)的原理和性能分析.[J].山西科技.2005,5 期.123-125
[21] Ranjan Bose(印).信息量、编码与密码学.[M].工业机械出版社.2010-9
[22] 王穆移,马瑞卿,孙银川,徐蕾.基于 CPLD 的有刷和无刷直流电动机混合驱动技术研究.[J].微电机.2009,42 卷 7 期.40-43
[23] 吕浩,吕世公,钱焕裕,徐翔.用于直流电机驱动的控制算法设计.[J].电子测量技术.2010-7,33 卷 7 期.46-50
[24] STC12C2052AD.pdf.[EB/OL]. http://www.stcmcu.com/datasheet/stc/STC-AD-PDF/STC12C2052AD.pdf .2010-4.
[25] CD00161566.pdf.[EB/OL]. http://www.stmicroelectronics.com.cn/st-web-ui/static/active/en/resource/technical/document/datasheet/CD00161566.pdf .2010-4
[26] 王水平,武芒,彭学愚.开关电源及其发展动态.[J].电子科技.1996-4,2 期.3-10
[27] 三浦宏文(日).机电一体化实用手册[M].科学出版社.2007
[28] Abraham I. Pressman,Keith Billings,Taylor Morey(美).开关电源设计(第三版).[M].电子工业出版社.2013

首都师范大学本科毕业设计论文

[29] Raj Kamal(印).微控制器架构、编程、接口和系统设计.[M].机械工业出版社.2009
[30] Joseph Yiu(英).ARM Cortex-M3 权威指南.[M].北京航空航天大学出版社.2012

首都师范大学本科毕业设计论文

致　谢

感谢潘巍老师对我大学四年的关心和指导，带领我参加各种竞赛和学术活动，让我受益匪浅。感谢本组组员申娇在毕设智能车上的大力支持。感谢电子信息工程系的袁慧梅、宋宇、赵晓旭、闵子健、付小雁、丁瑞、张永祥老师在大学四年期间对我的教育、指导和帮助。同样感谢信工学院的老师和领导们在我大学四年期间对我的帮助和教育。

感谢父母对我的养育之恩。是父母无私的奉献和对我的支持让我完成自己的大学学业，向人生理想迈进。感谢我的亲属们在我成长过程中对我的照顾和支持。

最后还要感谢我的同学和朋友们，正是他们的关心和支持，我才能顺利完成论文和大学学业。

转眼即将毕业，大学生活也将结束。我将承载着老师、家长、同学、朋友们的关怀与希望继续奋斗！

首都师范大学本科毕业设计论文

附录一：智能车电机控制(应用层)-传输协议

一、传输格式

主机发送至控制器

字节	0	1(可选)	2(可选)	3(可选)
内容	控制字	复位字(必须总是 0A5H)/速度值	时间值字	方向字

从机返回格式

字节	0	1(可选)	2(可选)	3(可选)	4(可选)
内容	标记字	状态值	速度值	时间值	方向

二、控制字

位	7	6	5	4	3	2	1	0
标记	RST	W/R	BRAKE/START SR	SPEED	TIMER	DIR	MR	ML

标记	说明
RST	为 1 的时候代表复位，随后跟随复位关键字 0A5H
W/R	W/R=1 表示执行写入操作，将对应设定内容写入到寄存器中。 W/R=0 表示读取操作，将指定内容的当前寄存器值按顺序发送到主机，此时 MR 和 ML 不能同时为 1
BRAKE/START	W/R=1 时： 当 BRAKE/START =1 时执行刹车。BRAKE/START=0 是启动 start，即执行一次重装值
SR	W/R=0 时： SR=1，表示读取当前状态字
SPEED	速度值 0~FF
TIMER	定时器重装值
DIR	方向，00 为 DIR=0，FF 为 DIR1
MR	为 1 时，表示该设定对右电机(电机 1)有效
ML	为 1 时，表示该设定对左电机(电机 0)有效

首都师范大学本科毕业设计论文

附录二：智能车私服舵机控制(应用层)-传输协议

一、传输格式

主机发送至控制器

字节	0	1(可选)	2(可选)
内容	控制字	复位字(必须总是 0A5H)/设定字	角度/PWM 值字

从机返回格式

字节	0	1(可选)	2(可选)
内容	标记字	状态字	角度/PWM 值字

二、控制字

位	7	6	5	4	3	2	1	0
标记	RST	W/R	-	SR	SV	DEGREE/PWM	S1	S0

标记	说明
RST	为 1 的时候代表复位，随后跟随复位关键字 0A5H
W/R	W/R=1 表示执行写入操作，将对应设定内容写入到寄存器中。 W/R=0 表示读取操作，将指定内容的当前寄存器值按顺序发送到主机，此时 S1 和 S0 不能同时为 1。
SR	W/R=1 时：写入的是设定字(Set Register) W/R=0 时：SR=1，表示读取当前状态字
SV	SV(Set Value)=1 时，表示设定角度/PWM 值，后边随着角度/PWM 值字
DEGREE/PWM	DEGREE/PWM=1 时表示数据为角度，为 0 时表示数据为 PWM
S1	为 1 时，表示该设定对伺服舵机 1 有效
S0	为 1 时，表示该设定对伺服舵机 0 有效

三、设定字

位	7	6	5	4	3	2	1	0
标记	-	-	-	-	INTERVAL_DIS	INTERVAL_EN	DISABLE	ENABLE

INTERVAL_DIS：禁用间隙工作模式

INTERVAL_EN：启用间隙工作模式

ENABLE：启用私服舵机

DISABLE：禁用私服舵机

首都师范大学本科毕业设计论文

附录三：UART.h 头文件

文件名：UART.h
功能：UART 物理层驱动

```
//本文件是串口通讯 UART 的驱动文件头文件
//第一版日期:2014-3-24 10:01:19
//作者:杨博

#ifndef _UART
#define _UART

//基本定义
#define INT_UART                4               //串口 1 的中断号
//================配置文件
//编译方式配置
#define UART_CONFIG_COMPILE_MOD_MACRO     0x01      //编译模式-快速编译,即用宏代替函数,而非按照函数编译
#define UART_CONFIG_COMPILE_MOD_FUN       0x02      //编译模式-函数编译,使用正常函数调用方式编译
#define UART_CONFIG_COMPILE_MOD           UART_CONFIG_COMPILE_MOD_MACRO
      //选择编译模式

//================寄存器模式位定义
//SCON 串口控制寄存器  位定义
#define SCON_RI         0x01            //接收中断标志位(需要手动清零)
#define SCON_TI         0x02            //发送中断标志位(需要手动清零)
#define SCON_RB8        0x04            //数据接收第 9 位(方式 2、3)
#define SCON_TB8        0x08            //数据发送第 9 位(方式 2、3)
#define SCON_REN        0x10            //串口接收使能
#define SCON_SM2        0x20            //多机通讯控制,SM2=1 时第 9 位 RB8=1 时才接收数据(方式 2、3),否则丢弃.方式 1 时接收到有效停止位时中断
#define SCON_SM1        0x40            //工作方式 SM1 位
#define SCON_SM0        0x80            //工作方式 SM0 位

//================模式定义
//SCON_SM 模式定义
#define SCON_SM_0    (0x00)             //串口模式 0,同步移位寄存器模式.TXD 为脉冲,RXD 为收发数据总线.波特率为 SYS_CLK/12
#define SCON_SM_1    (SCON_SM1)         //串口模式 1,10 位异步收发模式.波特率=T1 溢出频率/32
#define SCON_SM_2    (SCON_SM0)         //串口模式 2,11 位异步收发(有 RB8,TB8).波特率=SYS_CLK/64
#define SCON_SM_3    (SCON_SM0|SCON_SM1)  //串口模式 3,11 位异步收发(有 RB8,TB8).波特率=T1 溢出频率/32
#define SCON_SM_MASK   (SCON_SM0|SCON_SM1)  //用于指明 SCON 中 SM 所占位

//================定义常用量
//常用波特率声明
#define UART_BPS_300      300           //常用波特率(比特率) 300bit/s
#define UART_BPS_600      600           //常用波特率(比特率) 600bit/s
```

首都师范大学本科毕业设计论文

```
#define UART_BPS_1200   1200        //常用波特率(比特率) 1200bit/s
#define UART_BPS_1800   1800        //常用波特率(比特率) 1800bit/s
#define UART_BPS_2400   2400        //常用波特率(比特率) 2400bit/s
#define UART_BPS_3600   3600        //常用波特率(比特率) 3600bit/s
#define UART_BPS_4800   4800        //常用波特率(比特率) 4800bit/s
#define UART_BPS_7200   7200        //常用波特率(比特率) 7200bit/s
#define UART_BPS_9600   9600        //常用波特率(比特率) 9600bit/s
#define UART_BPS_14400  14400       //常用波特率(比特率) 14400bit/s
#define UART_BPS_19200  19200       //常用波特率(比特率) 19200bit/s
#define UART_BPS_28800  28800       //常用波特率(比特率) 28800bit/s

//指明不同来源下的分频系数
#define UART_DIV_MOD_0      12          //串口模式 0 下分频为 SYS_CLK/12
#define UART_DIV_MOD_1      32          //串口模式 1 下分频为 T1 溢出频率/32
#define UART_DIV_MOD_2      64          //串口模式 2 下分频为 SYS_CLK/64
#define UART_DIV_MOD_3      32          //串口模式 3 下分频为 T1 溢出频率/32,同模式 1

//================参数计算公式
#define UART_T1_FREQ(UART_BPS,UART_DIV)  ((unsigned long)(UART_BPS)*(UART_DIV))
    //计算在当前比特率(波特率)下,T1  定时器的溢出频率.此宏需要两个参数,并且返回为 unsigned
long 类型
    //UART_BPS 是欲需要的波特率,UART_DIV 是当前模式下的分频系数,例如串口模式 1 下,分频系
数为 32,若启用了双倍波特率,则改为 16
#define UART_T1_FREQ_STD(UART_BPS)
    UART_T1_FREQ(UART_BPS,UART_DIV_MOD_1)
    //标准模式(即没启用波特率加倍)下的直接计算 T1 溢出频率公式

//================函数声明
void UART_SETMOD(unsigned char MOD);
//设定 UART 的模式,MOD 是模式,即可设定工作在模式 0、1、2、3

//下面的函数有编译模式之分
//函数编译模式下执行
#if UART_CONFIG_COMPILE_MOD == UART_CONFIG_COMPILE_MOD_FUN
void UART_SETES(bit UART_ES);
//设定 UART 是否使能中断,UART_ES 是使能中断 1:使能,0:除能
bit UART_READES();
//读取 UART 是否使能中断,返回 ES 值
void UART_SETREN(bit UART_REN);
//设定 UART 是否允许接收,UART_REN=1:允许接收数据,0:不允许
bit UART_READREN();
//读取 UART 接收允许,返回 REN
void UART_SETTI(bit UART_TI);
//设定 UART 的 TI 发送中断标志位.应在软件清零 TI 时使用,UART_TI=1 则会强制挂起 UART 中断
    //在 UART 中断中,先判断是否是 TI 置位,若是表示为发送中断,需手动将 TI 清零.
bit UART_READTI();
//读取 UART 的发送中断标志位,返回 TI
void UART_SETRI(bit UART_RI);
//设定 UART 的 RI 接收中断标志位.应在软件清零 RI 时使用,UART_RI=1 则会强制挂起 UART 中断
    //在 UART 中断中,先判断是否是 RI 置位,若是表示为接收中断,需手动将 RI 清零.
bit UART_READRI();
//读取 UART 的接收中断标志位,返回 RI
void UART_SETSM2(bit UART_SM2);
```

首都师范大学本科毕业设计论文

```
//设定 UART 的 SM2 多机通讯控制位.SM2=1 时第 9 位 RB8=1 时才接收数据(方式 2、3),否则丢弃.方式 1 时接收到有效停止位时中断
bit UART_READSM2();
//读取 UART 的 SM2 多机通讯控制位,返回 SM2
void UART_SETTB8(bit UART_TB8);
//设定 UART 中第 9 位发送比特 TB8(方式 2、3),可作为校验位(双机通讯),地址/数据分离信号(多机通讯)
bit UART_READRB8();
//读取 UART 中第 9 位接收的比特 RB8(方式 2、3 中位 TB8 发送,方式 1 中 SM2=0 时接收到停止位)
void UART_SENDDATA(unsigned char DATA);
//8 位数据发送模式.从 UART 上发送一个字节的数据,DATA 为发送的 8bit 数据.
void UART_SENDDATA_9(unsigned char DATA,bit SEND_9);
//9 位数据发送模式.从 UART 上发送一个字节+1 比特的数据,DATA 为发送的 8bit 数据,SEND_9 为数据位第 9 位.
unsigned char UART_READDATA();
//数据读取函数,从 UART 上读取一个字节的数据
#endif

//函数编译模式下执行,使用宏
#if UART_CONFIG_COMPILE_MOD == UART_CONFIG_COMPILE_MOD_MACRO

#define UART_SETES(UART_ES) ES=(UART_ES)
#define UART_READES() ES
#define UART_SETREN(UART_REN) REN=(UART_REN)
#define UART_READREN() REN
#define UART_SETTI(UART_TI) TI=(UART_TI)
#define UART_READTI() TI
#define UART_SETRI(UART_RI) RI=(UART_RI)
#define UART_READRI() RI
#define UART_SETSM2(UART_SM2) SM2=(UART_SM2)
#define UART_READSM2() SM2
#define UART_SETTB8(UART_TB8) TB8=(UART_TB8)
#define UART_READRB8() RB8
#define UART_SENDDATA(DATA) SBUF=(DATA)
#define UART_SENDDATA_9(DATA,SEND_9) (TB8=(SEND_9),SBUF=(DATA))
#define UART_READDATA() SBUF
#endif

#endif
```

首都师范大学本科毕业设计论文

附录四：COM_UART.h 头文件

文件名：COM_UART.h
功能：COM_UART 通讯协议栈

```
//本文件是STC系列单片机 UART串口通讯协议栈 函数文件
//第一版日期：2014-4-2 11:34:20
//作者:杨博

#ifndef _COM_UART
#define _COM_UART

//================Config 配置
//常量定义
#define COM_UART_CONFIG_COMPILE_IDENTITY_MASTER     1       //本机处于主机地位
#define COM_UART_CONFIG_COMPILE_IDENTITY_SLAVE      2       //本机处于从机地位

//以下部分可裁剪
#define COM_UART_CONFIG_COMPILE_CRC_ENABLE          0       //是否开启CRC校验
编译,可裁剪部分
#define COM_UART_CONFIG_COMPILE_IDENTITY
    COM_UART_CONFIG_COMPILE_IDENTITY_SLAVE
    //确定本机地位是主机还是从机
#define COM_UART_CONFIG_BUFFER_LENGTH               8       //缓冲区字节长度
#define COM_UART_CONFIG_BUFFER_COUNT_SEND           2       //发送缓冲区数
量
#define COM_UART_CONFIG_BUFFER_COUNT_RECEIVE        2       //接收缓冲区数量
#define COM_UART_CONFIG_UART_INT                    INT_UART  // 对 应 UART 的
中断号

//CRC相关
#define COM_UART_CRC_GEN                    0x8D        //使用的CRC校验多项式

//函数替换定义
#define COM_UART_SEND_DATA(DATA)    UART_SENDDATA_9(DATA,0)     //发送数据字节
#define COM_UART_SEND_ADDRESS(DATA)             UART_SENDDATA_9(DATA,1)
    //发送地址字节
#define COM_UART_READ_DATA()            UART_READDATA()             // 读 取
一个字节的数据
#define COM_UART_READ_ADDRESS()             UART_READDATA()             //
读取一个字节的地址
#define COM_UART_WAIT_ADDRESS()             UART_SETSM2(1)
    //进入等待地址模式
#define COM_UART_WAIT_DATA()            UART_SETSM2(0)              //
进入等待数据模式

//本机地址**********一定要修改的
#define COM_UART_CONFIG_ADDRESS                 0x01    //本机地址,对于从机
非常重要
#define COM_UART_CONFIG_ADDRESSMASK                 0xF0    //地址掩码
```

首都师范大学本科毕业设计论文

```
//================变量类型声明&操作
typedef unsigned char COM_UART_IDATA_POINTER;                    //定义了一个字节的idata指针
#define COM_UART_IDATA_LOAD(DATA,POINTER)  (DATA)=(*((unsigned char idata *)(POINTER)))
//定义了一个加载,将POINTER所指示的idata内存数据载入DATA中
#define COM_UART_IDATA_STORE(DATA,POINTER)  (*((unsigned char idata *)(POINTER)))=(DATA)
//定义了一个存储,将POINTER所指示的idata内存存入DATA数据

//================变量声明
extern unsigned char idata COM_UART_SEND_BUFFER[COM_UART_CONFIG_BUFFER_COUNT_SEND][COM_UART_CONFIG_BUFFER_LENGTH];
    //发送缓冲区
extern unsigned char idata COM_UART_RECEIVE_BUFFER[COM_UART_CONFIG_BUFFER_COUNT_RECEIVE] [COM_UART_CONFIG_BUFFER_LENGTH];
    //接收缓冲区
extern unsigned char idata COM_UART_SEND_SR[COM_UART_CONFIG_BUFFER_COUNT_SEND];  //发送缓冲区状态寄存器
extern unsigned char idata COM_UART_RECEIVE_SR[COM_UART_CONFIG_BUFFER_COUNT_RECEIVE]; //接收缓冲区状态寄存器
extern unsigned char idata COM_UART_SEND_DATACOUNT[COM_UART_CONFIG_BUFFER_COUNT_SEND];  //发送缓冲区待发送数据长度
extern unsigned char idata COM_UART_RECEIVE_DATACOUNT[COM_UART_CONFIG_BUFFER_COUNT_RECEIVE];  //接收缓冲区接收数据长度
#if COM_UART_CONFIG_COMPILE_IDENTITY == COM_UART_CONFIG_COMPILE_IDENTITY_MASTER
    extern unsigned char idata COM_UART_SEND_ADDRESS[COM_UART_CONFIG_BUFFER_COUNT_SEND]; //发送数据地址,主机发送前设置的从机地址
    extern unsigned char idata COM_UART_RECEIVE_ADDRESS[COM_UART_CONFIG_BUFFER_COUNT_RECEIVE];// 接收数据地址,确定主机是从那台从机接收的
#endif

//位标记变量
extern bit COM_UART_SEND_SRCHANGE;              //发送缓冲区状态寄存器发生变化,可能是发送完成\发送失败等因素
extern bit COM_UART_RECEIVE_SRCHANGE;           //接收缓冲区状态寄存器发生变化,可能是接收完成\接收失败(CRC错误\超时)等因素

//功能开关变量
#if COM_UART_CONFIG_COMPILE_CRC_ENABLE == 1
    extern bit COM_UART_CRC_ENABLE;             //标记是否启用CRC校验
#endif

//================模式位声明

//接收SR状态寄存器位定义
#define COM_UART_RECEIVE_SR_EMPTY               0x01        //接收缓冲区为空,即可以将接收的数据存入其中
#define COM_UART_RECEIVE_SR_RECEIVING           0x02        //接收缓冲区正在被使用,处于接收状态
```

首都师范大学本科毕业设计论文

```
#define COM_UART_RECEIVE_SR_COMPLETED          0x04        //接收缓冲区接收完成,包括CRC校验
#define COM_UART_RECEIVE_SR_TIMEOUT            0x08        //接收缓冲区超时错误,即没有在指定时间间隙内连续完成传输
#define COM_UART_RECEIVE_SR_CRCERROR           0x10        //接收缓冲区CRC校验错误
#define COM_UART_RECEIVE_SR_RECEIVECOUNTERROR  0x20        //接收缓冲区溢出

//发送SR状态寄存器位定义
#define COM_UART_SEND_SR_EMPTY                 0x01        //发送缓冲区为空,即程序可以写入
//********注明当COM_UART_SEND_SR_EMPTY无效时,表示可以发送该缓冲区
#define COM_UART_SEND_SR_SEDING                0x02        //发送缓冲区正在使用,处于发送状态
#define COM_UART_SEND_SR_COMPLETED             0x04        //发送缓冲区发送完成,包括CRC校验发送
#define COM_UART_SEND_SR_TIMEOUT               0x08        //发送缓冲区发送超时,未能在有效时间内发送完全部数据

//=================宏函数
#define COM_UART_READ_SEND_SR(INDEX)     (COM_UART_SEND_SR[(INDEX)])
//读取发送状态寄存器,INDEX是状态寄存器组编号
#define COM_UART_SET_SEND_SR(INDEX,SRDATA)     (COM_UART_SEND_SR[(INDEX)]=(SRDATA))
//设定发送状态寄存器,INDEX是状态寄存器组编号,SRDATA是指定状态
#define COM_UART_READ_RECEIVE_SR(INDEX) (COM_UART_RECEIVE_SR[(INDEX)])
//读取接收状态寄存器,INDEX是状态寄存器组编号
#define COM_UART_SET_RECEIVE_SR(INDEX,SRDATA)     (COM_UART_RECEIVE_SR[(INDEX)]=(SRDATA))
//设定接收状态寄存器,INDEX是状态寄存器组编号,SRDATA是指定状态
#define COM_UART_READ_RECEIVE_DATACOUNT(INDEX) (COM_UART_RECEIVE_DATACOUNT[(INDEX)])
//读取接收缓冲区字节数,INDEX是数据长度寄存器组编号
#define COM_UART_SET_SEND_DATACOUNT(INDEX,DATACOUNT) (COM_UART_SEND_DATACOUNT[(INDEX)]=(DATACOUNT))
//设定发送缓冲区字节数,INDEX是数据长度寄存器组编号,DATACOUNT是字节数
#if COM_UART_CONFIG_COMPILE_IDENTITY == COM_UART_CONFIG_COMPILE_IDENTITY_MASTER
    #define COM_UART_SET_SEND_ADDRESS(INDEX,ADDRESS) (COM_UART_SEND_ADDRESS[(INDEX)]=(ADDRESS))
    //设定发送目标地址,即接收从机地址.INDEX是地址寄存器组编号,ADDRESS是接收从机地址
    #define COM_UART_READ_RECEIVE_ADDRESS(INDEX) (COM_UART_RECEIVE_ADDRESS[(INDEX)])
    //读取接收缓冲区地址,即从哪个从机发送的数据.INDEX是地址寄存器组编号
#endif
#if COM_UART_CONFIG_COMPILE_CRC_ENABLE == 1
    #define COM_UART_SET_CRC_ENABLE(CRC_EN) ((COM_UART_CRC_ENABLE)=(CRC_EN))
    //设定是否开启CRC校验使能,CRC_EN=1开启,CRC_EN=0禁止
    #define COM_UART_READ_CRC_ENABLE() (COM_UART_CRC_ENABLE)
    //读取CRC校验使能,返回CRC_ENABLE
#endif
//=================函数声明
void COM_UART_INI();
```

```
//COM_UART通讯协议栈模块初始化,包括变量赋值\状态寄存器复位
void COM_UART_TICK();
//COM_UART协议栈时钟,用于判断发送\接收超时,建议每隔2ms以上调用
void COM_UART_SEND_START();
//调用此函数开始发送,此函数会找到数组中非EMPTY的数组发送,从第0开始,一直到最后若都没有,则无动作
unsigned char COM_UART_GETEMPTY_SENDBUFFER();
//找到一个空的SENDBUFFER缓冲区,这样程序可以用来写入数据了,若没找到则返回0xFF
unsigned char COM_UART_GETCOMPLETED_RECEIVEBUFFER();
//找到一个已经接收完成的接收缓冲区,若没找到则返回0xFF
#if COM_UART_CONFIG_COMPILE_IDENTITY == COM_UART_CONFIG_COMPILE_IDENTITY_SLAVE
    void COM_UART_READY_SENDBUFFER(unsigned char SENDBUFFER_INDEX,unsigned char DATACOUNT);
    //就绪一个发送缓冲区,使得指定的缓冲区处于就绪状态,并设定欲发送的数据长度
        //SENDBUFFER_INDEX,发送缓冲区编号,DATACOUNT发送数据长度
#endif
#if COM_UART_CONFIG_COMPILE_IDENTITY == COM_UART_CONFIG_COMPILE_IDENTITY_MASTER
    void COM_UART_READY_SENDBUFFER(unsigned char SENDBUFFER_INDEX,unsigned char DATACOUNT,unsigned char ADDRESS);
    //就绪一个发送缓冲区,使得指定的缓冲区处于就绪状态,并设定欲发送的数据长度
        //SENDBUFFER_INDEX,发送缓冲区编号,DATACOUNT发送数据长度,ADDRESS是要发送到的从机地址
#endif
void COM_UART_CLEAN_RECEIVEBUFFER(unsigned char RECEIVEBUFFER_INDEX);
//清除一个接收缓冲区,包括其SR状态寄存器,指示为空,表示可以接收数据,应在程序读取完数据后清除
    //RECEIVEBUFFER_INDEX,接收缓冲区编号

//====================模块内部使用函数
//CRC校验函数
#if COM_UART_CONFIG_COMPILE_CRC_ENABLE == 1
    void COM_UART_CALCULATE_CRC_SEND(unsigned char NEW_DATA);
    //用于计算发送缓冲区的CRC值,NEW_DATA是移入的新字节,并对CRC产生影响
        //使用已有的COM_UART_SEND_CRC变量存储CRC校验字
    void COM_UART_CALCULATE_CRC_RECEIVE(unsigned char NEW_DATA);
    //用于计算接收缓冲区的CRC值,NEW_DATA是移入的新字节,并对CRC产生影响
        //使用已有的COM_UART_RECEIVE_CRC变量存储CRC校验字
#endif

#endif
```

首都师范大学本科毕业设计论文

附录五：MOTOR.h 头文件

文件名：MOTOR.h
功能：电机控制器头文件

```
//本文件是 STC 系列单片机 智能车 H 电桥电机控制 驱动头文件
//第一版日期：2014-3-31 14:45:54
//作者:yangkkb
#ifndef _MOTOR
#define _MOTOR

//===========位寄存器定义
//注明:本端口定义需要依据不同的芯片端口和 PCB 板进行修改
sbit MOTOR_PWM_0=P3^7;          //左电机(0 号电机),即 0 端口的 PWM 输出管脚
sbit MOTOR_PWM_1=P3^5;          //右电机(1 号电机),即 1 端口的 PWM 输出管脚
sbit MOTOR_DIR_0=P1^0;       //左电机(0 号电机)的运行方向,1 为正方向,0 为负方向
sbit MOTOR_DIR_1=P1^2;       //右电机(1 号电机)的运行方向,1 为正方向,0 为负方向
sbit MOTOR_FREE_0=P1^1;          //左电机(0 号电机)的刹车状态,1 为自由,0 为制动
sbit MOTOR_FREE_1=P1^3;          //右电机(1 号电机)的刹车状态,1 为自由,0 为制动

//===========Config 配置声明
#define MOTOR_SPEED_STEP   10              //电机调速步进尺度

//===========变量声明
//当前用于电机内部调整的变量
extern unsigned char MOTOR_TIMER_0;   //电机 0 的倒数定时器,用于记录工作多少时间片,为 0 时电机停止工作
extern unsigned char MOTOR_TIMER_1;   //电机 1 的倒数定时器,用于记录工作多少时间片,为 0 时电机停止工作
extern unsigned char MOTOR_PWM_0_CURRENT;  //电机 0 的当前 PWM 值,用于调速模块用,即实现变速 PWM
extern unsigned char MOTOR_PWM_0_TARGET;    //电机 0 的目标 PWM 值,用于调速模块用,即源程序目标设定的 PWM 值
extern unsigned char MOTOR_PWM_1_CURRENT;  //电机 1 的当前 PWM 值,用于调速模块用,即实现变速 PWM
extern unsigned char MOTOR_PWM_1_TARGET;    //电机 1 的目标 PWM 值,用于调速模块用,即源程序目标设定的 PWM 值
extern unsigned char MOTOR_SR_0;        //电机 0 状态寄存器,供外界读取电机状态
extern unsigned char MOTOR_SR_1;        //电机 1 状态寄存器,供外界读取电机状态
extern unsigned char MOTOR_SM_0;        //电机 0 的状态机
extern unsigned char MOTOR_SM_1;        //电机 1 的状态机

//定义由外部设置的变量缓冲,用作暂存
extern unsigned char MOTOR_TIMER_0_RELOAD;       //用于外界设定 TIMER0 的重载值,在 START 函数作用下重载到 TIMER_0 中,并启动控制
extern unsigned char MOTOR_TIMER_1_RELOAD;       //用于外界设定 TIMER1 的重载值,在 START 函数作用下重载到 TIMER_1 中,并启动控制

//定义位变量
extern bit MOTOR_ENABLE_0;                  //电机 0 是否使能,1 使能此时可以对电机操作,0 除能电
```

首都师范大学本科毕业设计论文

```
机自动变为 STOP 状态,并且不接受任何启动指令
extern bit MOTOR_ENABLE_1;                    //电机 1 是否使能,1 使能此时可以对电机操作,0 除能电机自动变为 STOP 状态,并且不接受任何启动指令

//===========电机模式定义
//正负定义
#define MOTOR_DIR_P              1            //电机正方向
#define MOTOR_DIR_N              0            //电机负方向

//SR 状态寄存器位声明
#define MOTOR_SR_DIR_P      0x01              //电机运行方向,1 表示为正方向,即电流从 1 号引脚流入 2 号引脚,DIR 管脚为高电平
#define MOTOR_SR_DIR_N           0x02         //电机运行方向,1 表示为反方向,即电流从 2 号引脚流入 1 号引脚,DIR 管脚为低电平
#define MOTOR_SR_RUN        0x04              //电机是否处于运行状态,1 为运行状态,0 为停止或刹车,具体参考 SR_BRAKE
#define MOTOR_SR_BRAKE           0x08         //电机是否处于刹车状态,1 为刹车状态,BRAKE=0 且 RUN=0 表示为停止状态
#define MOTOR_SR_TIMEUP          0x10         //电机控制器倒计时器超时,即倒计时器为 0,此时关闭属于定时自动关闭
#define MOTOR_SR_ERROR           0x80         //电机运行出现异常,可能导致电机控制失效,需要重新初始化

//SR_DIR 模式组合
#define MOTOR_SR_DIR_NONE        0x00         //电机没有运行,故没有方向
//#define MOTOR_SR_DIR_MASK      (MOTOR_SR_DIR_P|MOTOR_SR_DIR_P)     //标记处 SR 的 DIR 位所占位,粗心错误,由于电机反向失败检查查出 2014-4-13 1:18:10
#define MOTOR_SR_DIR_MASK        (MOTOR_SR_DIR_N|MOTOR_SR_DIR_P)     //标记处 SR 的 DIR 位所占位

//状态机模式定义
#define MOTOR_SM_FREE                0        //状态机 0,电机处于空闲状态
#define MOTOR_SM_DIRP_SPEEDUP        1        //状态机 1,电机处于正向加速状态
#define MOTOR_SM_DIRP_RUN            2        //状态机 2,电机处于正向匀速运行状态
#define MOTOR_SM_DIRP_SLOWDOWN       3        //状态机 3,电机处于正向减速状态
#define MOTOR_SM_DIRN_SPEEDUP        4        //状态机 4,电机处于负向加速状态
#define MOTOR_SM_DIRN_RUN            5        //状态机 5,电机处于负向匀速运行状态
#define MOTOR_SM_DIRN_SLOWDOWN       6        //状态机 6,电机处于负向减速状态
#define MOTOR_SM_BRAKE               7        //状态机 7,电机处于刹车状态
#define MOTOR_SM_ERROR               0xFF     //状态机?,电机控制器处于异常状态,此时电机停止运行

//===========定义常量
#define MOTOR_CH0       0x01          //电机 0 通道
#define MOTOR_CH1       0x02          //电机 1 通道

//===========函数声明
void MOTOR_INI();
//电机程序初始化,包括设置变量和初始化引脚值
void MOTOR_RESET();
//出现异常时调用复位函数重新复位电机控制状态,包括 PWM 重置,变量重置
void MOTOR_SETENABLE(unsigned char CHS,bit ENABLE);
```

首都师范大学本科毕业设计论文

```
//设定电机模块是否被使能,CHS 是通道,可以同时选择多个通道,ENABLE 是使能信息 1 使能,0 除能
bit MOTOR_READENABLE(unsigned char CH);
//读取电机相应使能,CH 是通道,只能选择一个,返回 ENABLE 表示是否使能
void MOTOR_SETTIMER(unsigned char CHS,unsigned char TIME);
//设定电机定时器重装(预装)值定时器,CHS 是通道,可以同时选择多个通道,TIME 是时间,单位为时间片
unsigned char MOTOR_READTIMER(unsigned char CH);
//读取电机定时器重装(预装)值寄存器,CH 是通道,只能选择一个,返回字节即定时器寄存器
unsigned char MOTOR_READTIMERREMAIN(unsigned char CH);
//去读正在倒计时的计时器寄存器值,即剩余电机时间.CH 是通道,只能选择一个,返回字节即计时器寄存器
void MOTOR_SETDIR(unsigned char CHS,bit DIR);
//设定电机方向,CHS 是通道,可以同时选择多个通道,DIR 是方向 1 为正方向,0 为负方向
bit MOTOR_READDIR_TARGET(unsigned char CH);
//读取目标方向值,即设定的 PWM 值,CH 是通道,只能选择一个,返回方向,1 表示正方向
bit MOTOR_READDIR_CURRENT(unsigned char CH);
//读取目标方向值,即当前的 PWM 值,CH 是通道,只能选择一个,返回方向,1 表示正方向
void MOTOR_SETPWM(unsigned char CHS,unsigned char PWM);
//设定电机的预期 PWM,CHS 是通道,可以同时选择多个通道,PWM 是希望的 PWM 值
unsigned char MOTOR_READPWM_TARGET(unsigned char CH);
//读取目标 PWM 值,即设定的 PWM 值.CH 是通道,只能选择一个,返回字节即目标 PWM
unsigned char MOTOR_READPWM_CURRENT(unsigned char CH);
//读取当前 PWM 值,.CH 是通道,只能选择一个,返回字节即当前 PWM
unsigned char MOTOR_READSR(unsigned char CH);
//读取当前电机状态,CH 是通道,只能选择一个,返回对应 SR 状态寄存器值
void MOTOR_START(unsigned char CHS);
//开始启动电机,CHS 是通道,可以同时选择多个通道
    //注明:相应电机要先使能,且定时器重装\预装值要设定
void MOTOR_STOP(unsigned char CHS);
//停止电机(减速停止,并非刹车),CHS 是通道,可以同时选择多个通道
    //注明:此函数不需要电机先使能
void MOTOR_BRAKE(unsigned char CHS);
//刹刹车函数(紧急停车),CHS 是通道,可以同时选择多个通道
    //注明:此函数不需要电机先使能,此函数修改 SR 和 SM 状态并清零计时器
void MOTOR_TIMER_TICK();
//电机时钟节拍,用于计时器倒计时使用,建议调用间隔大于等于 10ms
void MOTOR_SM_TICK();
//电机控制器状态机&调速器时钟节拍,用于控制状态转移和电机缓进调速,建议调用时间大于 100ms
#endif
```

首都师范大学本科毕业设计论文