第六届"飞思卡尔"杯
全国大学生智能汽车竞赛
技 术 报 告

上海交通大学

SHANGHAI JIAO TONG UNIVERSITY

学　　校：上海交通大学

队伍名称：Cyber++

参赛队员：龚路　王浩　黄炫圭

带队教师：王冰　王春香

关于技术报告和研究论文使用授权的说明

本人完全了解第六届“飞思卡尔”杯全国大学生智能汽车邀请赛关保留、使用技术报告和研究论文的规定，即：参赛作品著作权归参赛者本人，比赛组委会和飞思卡尔半导体公司可以在相关主页上收录并公开参赛作品的设计方案、技术报告以及参赛模型车的视频、图像资料，并将相关内容编纂收录在组委会出版论文集中。

参赛队员签名：_____

带队教师签名：_____

日　　期：_____

摘要

本文以第六届全国大学生智能车竞赛为背景，介绍了智能赛车控制系统的软硬件结构和开发流程。该比赛采用大赛组委会统一指定的由东莞市博思电子数码科技有限公司提供的C型车模，以Freescale半导体公司生产的16位DSC MC56F8366为核心控制器，在CodeWarrior IDE 开发环境中进行软件开发，要求赛车在未知道路上沿着黑线以最快的速度完成比赛。整个系统涉及车模机械结构调整、传感器电路设计及信号处理、控制算法和策略优化等多个方面。为了提高智能赛车的行驶速度和可靠性，对比了不同方案的优缺点，并结合PC调试平台进行了大量底层和上层测试，最终确定了现有的系统结构和各项控制参数。赛车采用模拟摄像头对赛道进行检测，通过边缘提取获得黑线位置，用PD方式对舵机进行反馈控制。同时通过速度传感器获取当前速度，采用优化后的PID控制实现速度闭环。

关键词：Freescale，智能车，摄像头，PID

ABSTRACT

In the background of the 6th National Intelligent Car Contest for College Students, this article introduces the software and hardware structures and the development flow of the vehicle control system. This contest adopting C-type car model provided by Dongguan Bosi Technology Co., Ltd. prescribed by the contest organization committee, using the 16-bit DSC MC56F8366 produced by Freescale Semiconductor Company as the core controller, developing under the CodeWarrior IDE, requires the car finish the race in the fastest speed. The whole system includes the aspects of the mechanism structure adjustment, the sensor circuit design and signal process, control algorithm and strategy optimization etc. In order to increase the speed and the reliability of the car, the advantage and disadvantage of the different schemes are compared, and a great number of the bottom layer and the upper layer tests are carried on combined with the PC simulation platform. At last, the current system structure and each control parameters are determined. It captures the road information through a camera, then abstracts the black line position using edge detector. After that, PD feedback control is used on the steering. At the same time, the system obtains the current speed using two speed sensors, so that it can realize the feedback control of the speed by an PID controlling algorithm. According to the pre-judge inform and the memorial inform, it allocates the speed properly.

Key words: Freescale，intelligent vehicle, camera, PID

目录

第一章 引言

1.1 智能车研究背景

1.1.1 发展历史

智能小车系统是迷你版的智能汽车，二者在信息提取，信息处理，控制策略及系统搭建上有很多相似之处，可以说智能小车系统将为智能汽车提供很好的试验和技术平台，推动智能汽车的发展。摄像头作为很多智能汽车的信息输入的传感器具有其他很多传感器所不具有的优势，如对物体的特征提取，视野宽广等，因此基于摄像头的智能小车系统的研究将推动智能汽车的发展。

智能汽车是未来汽车的发展方向，将在减少交通事故、发展自动化技术、提高舒适性等许多方面发挥很重要的作用；同时智能汽车是一个集通信技术，计算机技术，自动控制，信息融合技术，传感器技术等于一身的行业，它的发展势必促进其他行业的发展，在一定程度上代表了一个国家在自动化智能方面的水平[1]。汽车在走过的 100 多年的历史中，从没停止过智能化的步伐，进入 20 世纪 90 年代以来，随着汽车市场竞争激烈程度的日益加剧和智能运输系统（ITS）地兴起，国际上对于智能汽车及其相关技术的研究成为热门，一大批有实力有远见的大公司、大学和研究机构开展了这方面的研究。很多美国、日本和欧洲等国家都十分重视并积极发展智能车系统，并进行了相关实验，取得了很多成就。我国的相关研究也已经开展，清华大学成立了国内最早的研究智能汽车和智能交通的汽车研究所，在汽车导航、主动避撞、车载微机等方面进行了广泛而深入的研究，2000 年上海智能交通系统进入实质性实施阶段，国防科大研制出第四代无人驾驶汽车，西北工业大学、吉林交通大学、重庆大学等也展开了相关研究[2]。这一新兴学科正在吸引越来越多的研究机构和学者投入其中。

1.1.2 智能车研究方向

1.驾驶员行为分析（Driver Behavior Analysis）任务：研究驾驶员的行为方式、精神状态与车辆行驶之间的内在联系；目的：建立各种辅助驾驶模型，为智能车辆安全辅助驾驶或自动驾驶提供必要的数据，如对驾驶员

面　　表情的归类分析能够判定驾驶员是否处于疲劳状态，是否困倦瞌睡等；

2.环境感知（Environmental Perception）主要是运用传感器融合等技术，来获得车辆行驶环境的有用信息，如车流信息、车道状况信息、周边车辆的速度信息、行车标志信息等；

3.极端情况下的自主驾驶（Autonomous Driving on Extreme courses）主要研究在某些极端情况下，如驾驶员的反应极限、车辆失控等情况下的车辆自主驾驶；

4.规范环境下的自主导航（Autonomous Navigation on Normal environment）主要研究在某些规范条件下，如有人为设置的路标或道路环境条件较好，智能车辆根据环境感知所获得的环境数据，结合车辆的控制模型，在无人干预下，自主地完成车辆的驾驶行为。

5.车辆运动控制系统（Vehicle Motion Control Systems）研究车辆控制的运动学、动力学建模、车体控制等问题；

6.主动安全系统（Active，Safety Systems）主要是以防为主，如研究各种情况下的避障、防撞安全保障系统等；

7.交通监控、车辆导航及协作（Traffic Monitoring，Vehicle Navigation and coordination）主要研究交通流诱导等问题；

8.车辆交互通信（Inter—Vehicle Communication）研究车辆之间有效的信息交流，主要是各种车辆间的无线通信问题；

9.军事应用（Military Applications）研究智能车辆系统在军事上的应用；

10.系统结构（System Architectures）研究智能车辆系统的结构组织问题；

11.先进的安全车辆（Advanced Safety Vehicles）研究更安全、具有更高智能化特征的车辆系统。

1.1.3 智能车的应用前景

智能车系统有着极为广泛的应用前景。结合传感器技术和自动驾驶技术可以实现汽车的自适应巡航并把车开得又快又稳、安全可靠；汽车夜间行驶时，如果装上红外摄像头，就能实现夜晚汽车的安全辅助驾驶；他也可以工作在仓库、码头、工厂或危险、有毒、有害的工作环境里，此外他还能担当起无人值

守的巡逻监视、物料的运输、消防灭火等任务。在普通家庭轿车消费中，智能车的研发也是很有价值的，比如雾天能见度差，人工驾驶经常发生碰撞，如果用上这种设备，激光雷达会自动探测前方的障碍物，电脑会控制车辆自动停下来，撞车就不会发生了。

1.2 飞思卡尔智能汽车大赛介绍

1.2.1 飞思卡尔智能车大赛简介

飞思卡尔公司开发嵌入式解决方案的历史可追溯到 50 多年前，现在，已发展成为在 20 多个国家设有业务机构，拥有 20,000 多名员工的实力强大的独立企业。

飞思卡尔公司专门为汽车、消费电子、工业品、网络和无线应用提供“大脑”。他们无比丰富的电源管理解决方案、微处理器、微控制器、传感器、射频半导体、模块与混合信号电路及软件技术已嵌入在全球使用的各种产品中。并拥有雄厚的知识产权，其中包括 6,200 多项专利。

为加强大学生实践、创新能力和团队精神的培养，促进高等教育教学改革，受教育部高等教育司委托(教高司函[2005]201 号文，附件 1)，由教育部高等自动化专业教学指导分委员会（以下简称自动化分教指委）主办全国大学生智能汽车竞赛。该竞赛以智能汽车为研究对象的创意性科技竞赛，是面向全国大学生的一种具有探索性工程实践活动，是教育部倡导的大学生科技竞赛之一。该竞赛以“立足培养，重在参与，鼓励探索，追求卓越”为指导思想，旨在促进高等学校素质教育，培养大学生 的综合知识运用能力、基本工程实践能力和创新意识，激发大学生从事科学研究与探索的兴趣和潜能，倡导理论联系实际、求真务实的学风和团队协作的人文精神， 为优秀人才的脱颖而出创造条件。

该竞赛由竞赛秘书处为各参赛队提供/购置规定范围内的标准硬软件技术平台，竞赛过程包括理论设计、实际制 作、整车调试、现场比赛等环节，要求学生组成团队，协同工作，初步体会一个工程性的研究开发项目从设计到实现的全过程。该竞赛融科学性、趣味性和观赏性为一体，是以迅猛发展、前景广阔的汽车电子为背景，涵盖自动控制、模式识别、传感技术、电子、电气、计算机、机械与汽车等多学科专业的创意性比赛。该竞赛规则透明，评价标准客观，坚持公开、公平、公正的原则，保证竞赛向健康、普及，持续的方向发展。

该竞赛以飞思卡尔半导体公司为协办方，得到了教育部相关领导、飞思卡尔

公司领导与各高校师生的高度评价，已发展成全国 30 个省市自治区近 300 所高校广泛参与的全国大学生智能汽车竞赛。2008 年起被教育部批准列入国家教学质量与教学改革工程资助项目中科技人文竞赛之一（教高函[2007]30 号文）。

全国大学生智能汽车竞赛原则上由全国有自动化专业的高等学校（包括港、澳地区的高校）参赛。竞赛首先在各个分赛区进行报名、预赛，各分赛区的优胜队将参加全国总决赛。每届比赛根据参赛队伍和队员情况，分别设立光电组、摄像头组、创意组等多个赛题组别。每个 学校可以根据竞赛规则选报不同组别的参赛队伍。全国大学生智能汽车竞赛组织运行模式贯彻“政府倡导、专家主办、学生主体、社会参与”的 16 字方针，充分调动各方面参与的积极性。

全国大学生智能汽车竞赛一般在每年的 10 月份公布次年竞赛的题目和组织方式，并开始接受报名，次年的 3 月份进行相关技术培训，7 月份进行分赛区竞赛，8 月份进行全国总决赛。

1.2.2 比赛规则介绍

选手须使用竞赛秘书处统一指定的竞赛车模套件，采用飞思卡尔半导体公司的8 位、16 位微控制器作为核心控制单元，自主构思控制方案进行系统设计，包括传感器信号采集处理、电机驱动、转向舵机控制以及控制算法软件开发等，完成智能车工程制作及调试，于指定日期与地点参加各分赛区的场地比赛，在获得决赛资格后，参加全国决赛区的场地比赛。参赛队伍的名次（成绩）由赛车现场成功完成赛道比赛时间为主，技术报告、制作工程质量评分为辅来决定。大赛根据车模检测路径方案不同分为电磁、光电与摄像头三个赛题组。车模通过感应由赛道中心电线产生的交变磁场进行路经检测的属于电磁组；车模通过采集赛道图像（一维、二维）进行进行路经检测的属于摄像头组；车模通过采集赛道上少数孤立点反射亮度进行路经检测的属于光电组。

1.2.2.1 器材限制规定

1. 须采用统一指定的车模。本届比赛指定采用三种车模，分别用于三个赛题组：

A 型车模：东莞市博思电子数码科技有限公司提供，限定光电组比赛使用。

B 型车模：北京科宇通博科技有限公司提供。限定电磁组使用。

C 型车模：东莞市博思电子数码科技有限公司提供。限定摄像头组使用。

2. 须采用飞思卡尔半导体公司的 8 位、16 位处理器作为唯一的微控制器。

3.参加电磁赛题组不允许使用传感器获取道路的光学信息进行路径检测；

参加光电赛题组中不允许传感器获取道路图像信息进行路径检测；

参加摄像头赛道组可以使用光电管作为辅助检测手段。

4.其他事项

如果损毁车模中禁止改动的部件，需要使用相同型号的部件替换；

车模改装完毕后，尺寸不能超过：250mm 宽和 400mm 长。

1.2.2.2 有关赛场的规定

1.赛道基本参数（不包括拐弯点数、位置以及整体布局），详见官网通知；

2.比赛赛道实际布局将在比赛当日揭晓，在赛场内将安排采用制作实际赛道的材料所做的测试赛道供参赛队进行现场调试。

1.2.2.3 裁判及技术评判

竞赛分为分赛区和全国总决赛两个阶段。其中全国总决赛阶段是在竞赛组委会和秘书处指导下，由决赛承办学校成立竞赛执行委员会，下辖技术组、裁判组和仲裁委员会，统一处理竞赛过程中遇到的各类问题。

所有竞赛组织委员会工作人员，包括技术评判组及现场裁判组人员均不得参与任何针对个别参赛队的指导或辅导工作（提供微控制器培训除外），不得泄露任何有失竞赛公允的信息。

在分赛区预赛阶段中，裁判以及技术评判由各分赛区组委会参照上述决赛阶段组织实施。

1.2.2.4 分赛区、决赛区比赛规则

分赛区和全国总决赛的比赛规则相同，都具有电磁组、光电组和摄像头组比赛。三个赛题组在同一个场馆同时进行比赛，所遵循的比赛规则是相同的。三个赛题组分别独立进行成绩排名。

分赛区和全国总决赛的现场比赛均包括预赛和决赛两个阶段。下面列出的现场预赛、决赛阶段的比赛规则适用于各分赛区及全国总决赛的三个赛题组。

1.初赛与决赛规则

1）初赛规则

i.比赛场中有三条赛道。

ii.参赛队根据比赛题目分为三个组，并以抽签形式决定组内比赛次序。

iii.比赛分为两轮，三组同时在三个赛道上进行比赛，每支队伍可以在每轮比

赛之前有 10 分钟的现场调整时间。在此期间，参赛队伍只允许对赛车的硬件（不包括微控制器芯片）进行调整。第二轮比赛在同一赛道沿逆向进行。

iv.在每轮比赛中，选手首先将赛车放置在起跑区域内赛道上，赛车至少静止两秒后自动启动。

v.每辆赛车在赛道上跑一圈，以计时起始线为计时点，跑完一圈后赛车需要自动停止在起始线后三米之内的赛道内，如果没有停止在规定的区域内，比赛计时成绩增加 1 秒。

vi.每辆赛车以在两个单轮成绩中较好的一个作为赛车最终预赛成绩；计时由电子计时器完成并实时显示。

vii.根据参赛队伍数量，由比赛组委会根据成绩选取一定比例的队伍晋级决赛。

viii.晋级决赛的赛车在决赛前有 10 分钟的调整时间。在此期间，参赛队伍只允许对赛车的硬件（不包括微控制器芯片）进行调整。技术评判组将对全部晋级的赛车进行现场技术检查，如有违反器材限制规定的(指本规则之第一条)当时取消决赛资格，由后备首名晋级代替。

ix.由裁判组申报组织委员会批准公布决赛名单。

x.全部车模在整个比赛期间都统一放置在车模的展示区内。

2）决赛阶段规则

i. 参加决赛队伍按照预赛成绩进行排序，比赛顺序按照预赛成绩的倒序进行。

ii. 决赛的比赛场地使用一个赛道。决赛赛道与预赛赛道形状不同，占地面积增大，赛道长度增加。

iii.每支决赛队伍只有一次比赛机会，在跑道上跑一圈，比赛过程与要求同预赛阶段。

iv.计时由电子计时器完成并实时显示。

v.预赛成绩不记入决赛成绩，只决定决赛比赛顺序。没有参加决赛阶段比赛的队伍，预赛成绩为最终成绩，参加该赛题组的排名。

2.比赛过程规则

按照比赛顺序，裁判员指挥参赛队伍顺序进入场地比赛。同一时刻，一个场地上只有一支队伍进行比赛。

在裁判员点名后，每队指定一名队员持赛车进入比赛场地。参赛选手有 60

秒的现场准备时间。准备好后，裁判员宣布比赛开始，选手将赛车放置在起跑区，赛车应在起跑区静止两秒钟以上，然后自动出发。赛车应该在 30 秒之内离开出发区，沿着赛道跑完一圈。由计时起始线两边传感器进行自动计时。赛车跑完一圈且自动停止后，选手拿起赛车离开场地，将赛车放回指定区域。

如果比赛完成，由计算机评分系统自动给出比赛成绩。

3．比赛犯规与失败规则

比赛过程中，由比赛现场裁判根据统一的规则对于赛车是否冲出跑道进行裁定。赛车前两次冲出跑道时，由裁判员取出赛车交给比赛队员，立即在起跑区重新开始比赛。选手也可以在赛车冲出跑道后放弃比赛。

比赛过程中出现下面的情况，算作模型车冲出跑道一次。

i．裁判点名后，60 秒之内，参赛队没有能够进入比赛场地并做好比赛准备；

ii．比赛开始后，赛车在 30 秒之内没有离开出发区；

iii．赛车在离开出发区之后 60 秒之内没有跑完一圈；

比赛过程中如果出现有如下一种情况，判为比赛失败：

i．赛车冲出跑道的次数超过两次；

ii．比赛开始后未经裁判允许，选手接触赛车；

iii．决赛前，赛车没有通过技术检验。

如果比赛失败，则不计成绩。

4. 比赛禁止事项：

i．不允许在赛车之外安装辅助照明设备及其它辅助传感器等；

ii．选手进入比赛场地后，除了可以更换电池之外，不允许进行任何硬件和软件的修改；

iii．比赛场地内，除了裁判与 1 名队员之外，不允许任何其他人员进入场地；

iv．不允许其它干扰赛车运动的行为；

v．不允许车模设计方案抄袭，参赛队伍的车模设计的硬软件需要相互之间有明显的不同。

5．比赛组织说明：

1)现场正式比赛前，每个参赛队伍都有现场环境适应性调试阶段。调试跑道与比赛跑道形状不一样。

2)比赛开赛之前，所有车模都由比赛组委会收集并存放在同一保管区域，直到比赛结束。

3)在比赛期间，大赛组委会技术处将根据情况对参赛车模进行技术检查。如果违反了比赛规则的禁止事项，大赛组委会有权取消参赛队伍的成绩。

4)为了便于进行技术交流全国总决赛之后，获得全国总决赛特等奖、一等奖的车模将由全国竞赛组委会代为保管两年。

1.2.2.5. 其他

1.比赛过程中有其他作弊行为的，取消比赛成绩；

2.参加预赛并晋级决赛的队伍人员不允许改变。

3.本规则解释权归竞赛秘书处和比赛组织委员会所有。

第二章　模型车设计制作思路及实现的技术方案概要说明

2.1 智能车整体框架

智能车是在车模结构的框架上，搭上硬件结构，通过 MC56F8366 单片机的处理能力，将传感器采集到的信息处理分析后得出运算结果，指挥电机和舵机做出适应赛道及战术策略的响应的一套系统。其硬件结构框架如下图：

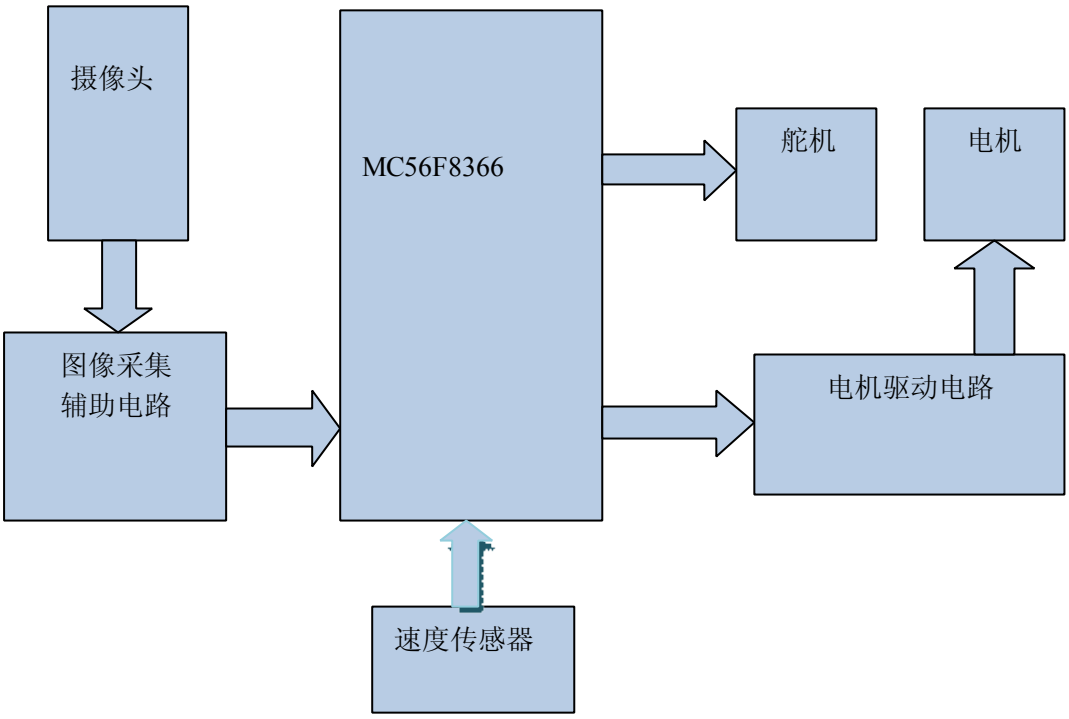


图 1-1 智能车硬件框图

2.2 赛道信息识别

赛道信息的识别是车辆控制的基础，为了提高行驶稳定性和优化行车线路，很多设计都把重点放在了使传感器感知更远的距离上、获取更多的信息。现有赛道信息检测方案总体上有两大类：光电传感器方案和摄像头方案。前者电路设计简单、信息检测频率高，但检测范围、精度有限且能耗较大，信息量小；

后者获取的赛道信息丰富，但电路设计和软件处理较复杂，且信息更新速度较慢。关于光电管方案，采用"线型检测阵列"的方法，以 4 个发光二极管和 1 个光敏二极管构成一个检测点，将多个检测点成直线排列。共排成两排，两排都安置在智能车底盘前端。然后采用巡回检测方式来寻找黑色导引线。"线型检测阵列"寻线装置，采用了单片机来实现对光电管检测信号的处理、分析，并实现对电机和舵机控制脉冲的产生。虽然采用光电传感器构成"线型检测阵列"的方案简单易行，但是将其应用于智能车竞赛中仍有一定局限性。例如，它对赛道的检测精度低，即使采用多个光电管，通常也只能确定 10-20 个状态，这对赛车转向的平滑控制是非常不利的。此外，线型检测阵列的探测距离较短，如果简单地将检测阵列元件向车模前方伸出安装，又将增加舵机的转向负载，另外光电管的耗电量也非常显著。基于前述对"线型检测阵列"寻线能力局限性的考虑，选择采用摄像头作为寻线传感器。一方面，摄像头所能检测的赛道信息远多于"线型检测阵列"所能检测的信息，有利于区分各种道路类型；另一方面，摄像头检测范围调整灵活，可以提供足够远的预判距离。实际上，通过"超频"和提高代码效率，并选择合适的图像处理算法，使用比赛规定的单片机完全可以对低线数黑白摄像头的视频信号进行采样和处理，有效识别出导引线的位置和相关几何信息。

2.3 车体控制

智能车应根据前方道路类型和当前车体位置误差及速度，对舵机转角和驱动电机 PWM 波的占空比进行及时调整。关于这方面的控制方案有 PID 控制、Bang-Bang 控制和模糊控制等。前四年我校参赛队的技术报告中均对舵机 PID 控制进行了深入研究，文中列出了不同 P、D 参数下赛车的平均速度和调整程度，并指出当 P 参数适中、D 参数较小时，性能最佳。此外,该文还介绍了 S12 单片机模糊控制指令的应用。虽然理论上模糊控制在非线性场合存在优势，但实际测试表明 PD 控制已能满足需求，所以舵机控制仍采 PD 控制。电机速度闭环控制使用 PID 控制。

第三章　模型车机械设计说明

3.1 车模参数

车模采用大赛主委会规定的 C 型车模，技术参数如表 2-1.车模实物如图 2-1。

表 3-1 车模参数

类别	参数
长*宽*高(mm)	280×160×60
前轮距（mm）	140
后轮距（mm）	140
轮胎宽度 （mm）	24
轮胎直径(mm)	50
轮胎材料	ABS
驱动方式	后轮驱动

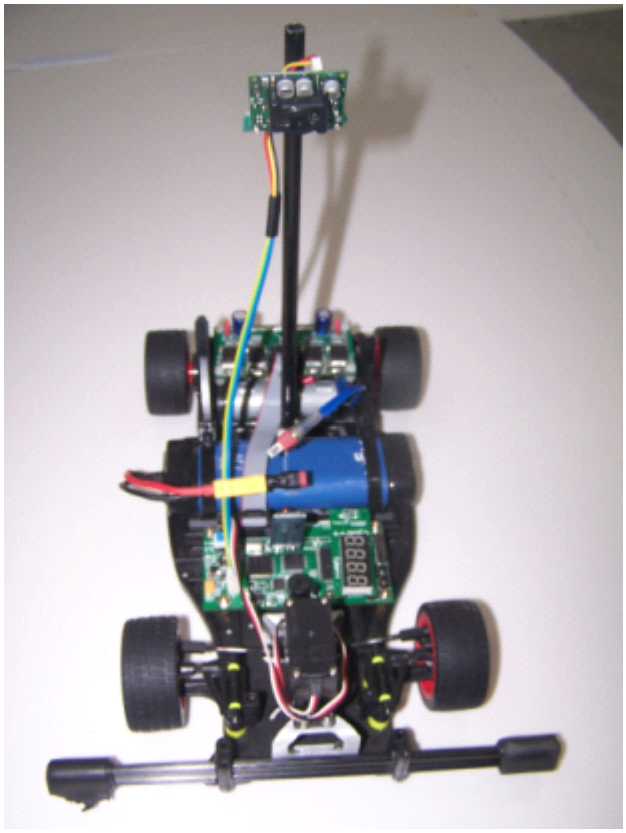


图 3-1 车模实物图

3.2 车模转向机构调整

车模转向由前轮舵机经过连杆将转动变为平行四边形结构的转动，从而实现了前轮的转动。默认的结构中，舵机侧卧放置，由于舵机的转动中心不在侧向的中心，造成了转动中心偏向一侧的情况，同时转动中心距离地盘比较小，造成舵机臂不能太长。以上两个原因导致了在完成左右同等角度的转向是，舵机需要分别向左右打出不等但都比较大的角度，这将会导致左右转向的响应速度上的差以及总体效率的地下。考虑到本届比赛所使用的 Futaba S3010 舵机在中心附近的执行效率最高，同时舵机的力矩远大于默认安装所需要的力矩，我们考虑加长舵机臂并且让舵机转动中心位于车身左右中心上，加长舵机臂使得在转动相同角度过程中，舵机需要转过的角度大大减小，提高了响应速度。但是加长舵机臂要有度，若太长，舵机提供不了转向所需要的力矩，轻者使小车转向不足，重者会导致舵机烧毁。在反复测试之后，我们舵机转轴中心到连杆连接部分的中心距离为 2.7cm。为了让舵机处于车身中心，我们自己设计了一套支架，将舵机卧向安装，及转动轴向下的方式来安装舵机。如图 3-2

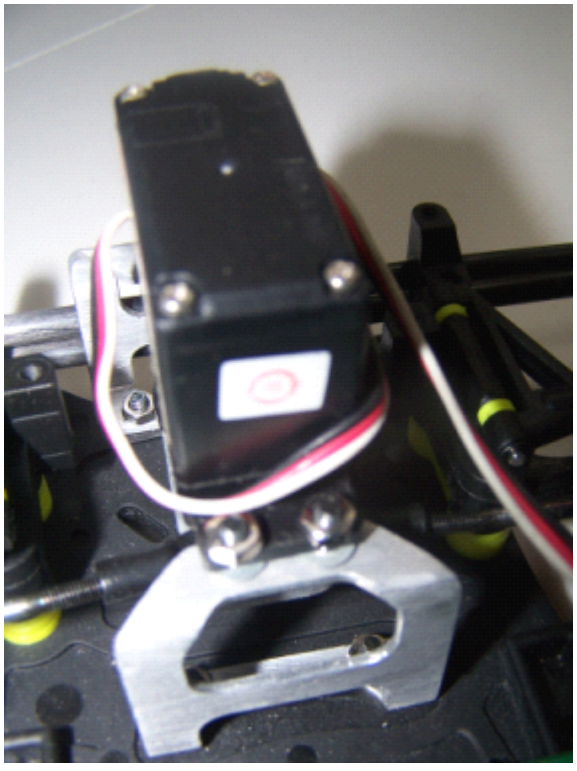


图 3-2 舵机安装方式

该方案的优点是：

一、可以根据需要选择舵机输出杆的长度，从而获得所需要的灵敏度，但是舵机输出杆的长度也不能太长，因为这会对舵机的输出力矩有较高的要求，太长会烧坏舵机。

二、效率较高，舵机的单边（例如取左边）效率,对于长连杆方案来说，左轮角在增大的同时，右轮角在减小，而且角是在 0-45 度之间，同时的变动也比较小（45 度-135 度），因此长连杆的效率变动较小且效率较高。

三、转角大，由于长连杆方案中舵机输出杆的转动在同一个平面内，当其到达极限位置时，转角比平行四边形方案要大。

四、转向更灵敏，因为放大倍数较平行四边形方案要大。

3.3 前轮调整

小车在调试过程中，转向轮定位参数是很重要的因素，它通常不易被察觉，但是却有着较大的危害。如果取得不恰当，那么将造成转向不灵活，效率低以及转向轮侧滑等问题，使得小车性能下降，加速轮胎的磨损。

转向轮定位参数包括：主销内倾角、主销后倾角、转向轮外倾角及转向轮前束。这其中最重要的就是转向轮外倾角和转向轮前束。

主销内倾是指主销装在前轴略向内倾斜的角度，它的作用是使前轮自动回正，如图 3-5。内倾角度越大前轮自动回正的作用就越强烈，但转向时也越费力，轮胎磨损增大；反之，角度越小前轮自动回正的作用就越弱[3]。

主销后倾（Caster）是指主销装在前轴，上端略向后倾斜的角度，如图 3-4。它使车辆转弯时产生的离心力所形成的力矩方向与车轮偏转方向相反，迫使车轮偏转后自动恢复到原来的中间位置上。由此主销后倾角越大，车速越高，前轮稳定性也愈好。主销内倾和主销后倾都有使汽车转向自动回正，保持直线行驶的功能。不同之处是主销内倾的回正与车速无关，主销后倾的回正与车速有关，因此高速时后倾的回正作用大，低速时内倾的回正作用大[3]。

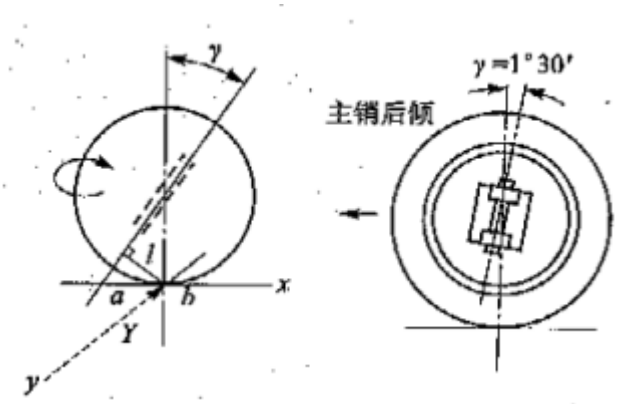


图 3-3 主销后倾角[3]

车轮外倾角（Camber）是指从前放看前轴时，轮胎的中心平面不是垂直的，而是上面向外倾斜一个角度，如图 3-5。设置转向轮的外倾角是为了平衡和协调因为车重造成的前轮内倾倾向，使轮胎和路面呈垂直接触的最佳状态[4]。

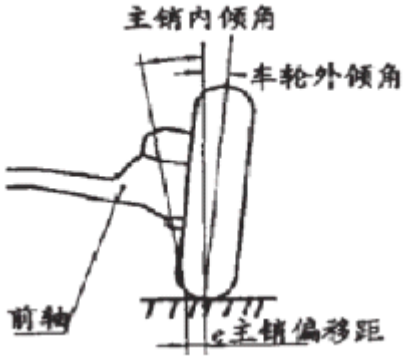


图 3-4 主销内倾角和车轮外倾角[4]

转向轮前束(Toe out)是指同一轴两端车轮轮辋内侧轮廓线的水平直径的端点为等腰梯形的顶点，底边为车轮轴线。等腰梯形两底边长度之差为前束。如图 3-6 所示，当梯形前低边小于后底边时，前束为正（A<B），反之为负。车轮的水平直径与纵向平面之间的夹角为前束角。正的前束角在车轮中心产生向内的侧向力，而正的外倾角在车轮中心产生向外的侧向力，因此前束角的作用是与外倾角协调，保持车轮做纯滚动和直线行驶，从而减少轮胎磨损，提高汽车的操纵稳定性[4]。

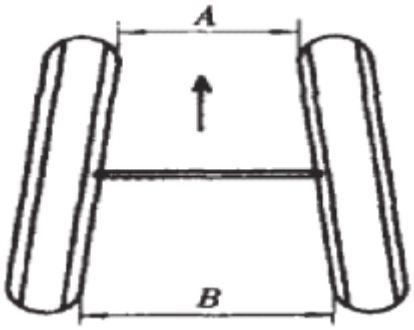


图 3-5 前束[4]

经过测试对车模的外倾角，前束进行了调整，如表 3-2。

表 3-2 转向轮调整

类别	参数
转向轮前束角（度）	1
转向轮外倾角(度)	1

3.4 编码器安装

光电编码器是智能小车速度反馈元件，其安装位置应该充分考虑测速的准确性和防干扰。由于本届 C 型车模有两个电机，且我们选用的编码器体积较大，给按照造成了极大的不便，在尝试了各种可能性后，采用了如图 3-6 的安装方式。将两个编码器安装于车模尾部，与车轮传动齿轮耦合。

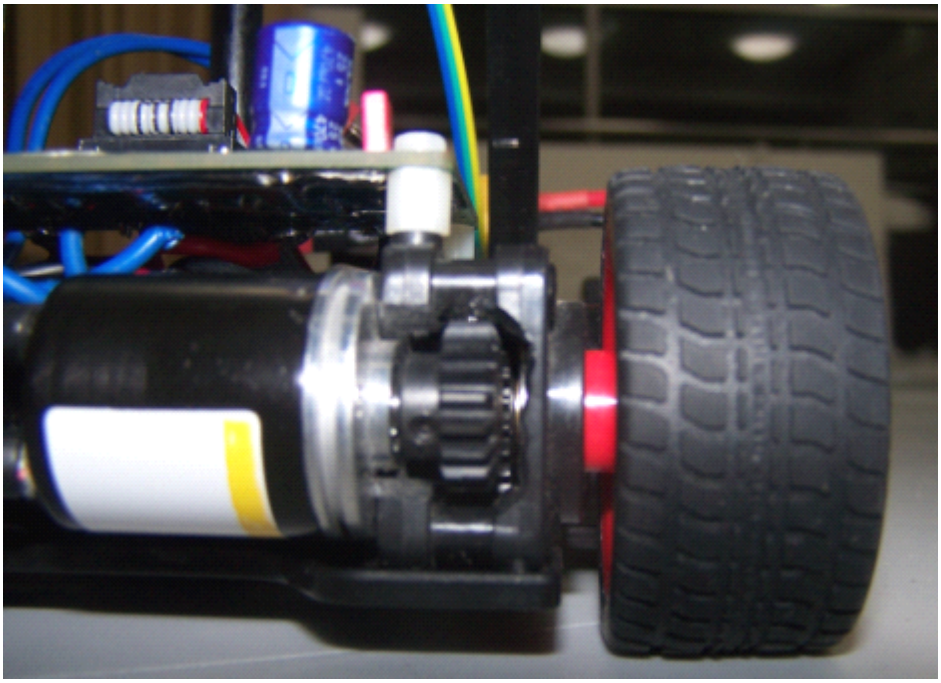


图 3-6 编码器安装

3.5 本章小结

机械结构是智能小车的基础，其首先决定了小车的性能，特别是转向和加速性能，本章主要通过对转向机构的分析以及对汽车理论知识的学习和应用，找到了影响小车转向范围和效率的因素并进行了改进，使得小车的转向灵敏度得到提高，左右转向极限提高若干度，同时舵机效率也得到较大提高。

第四章　控制电路设计说明

硬件部分组成：主板、显示板、驱动板、摄像头、旋转编码器、伺服电机和直流电动机。

4.1 主板

我们的主板集成了智能车控制所需的所有电路，因此其稳定性和性能至关重要。

4.1.1 CPU 选型

我们首先进行了 CPU 的选型。根据今年的比赛规则，推荐使用 Freescale 9S12XS128 作为微控制器，也可以使用 16 位 DSC 或 8 位 MCU 芯片。因此，我们首先考察 9S12XS128，以下简称 S12。

S12 是一款常用的 16 位 MCU，有丰富的片内资源，往届绝大部分参赛队也使用了此款 MCU，因此其资料丰富。然而，S12 的片内 RAM 只有 8KB，不能完全满足摄像头图像采集时保存多个帧缓冲区的需要；S12 的最高 CPU 总线频率只有 40MHz，不少参赛队将其超频至 80MHz 甚至更高进行使用，然而这是以牺牲稳定性为代价的。今年摄像头组使用 C 型车模，后轮采用两个独立的电机，为了对它们分别进行精确的速度控制，必然需要使用两个旋转测速传感器。然而，S12 只有一路 16 位硬件计数器，不能满足需求。采用中断进行软件计数会消耗大约 10%的 CPU 资源；采用外部计数电路会导致电路臃肿，体积膨胀；采用时分复用计数端口的思路则会牺牲速度测量的准确性。此外，S12 不具有外部总线，为了对摄像头信号进行采集，只能使用片内的 AD 转换器，其转换速率较低，导致图像每行的点数不能满足需求。如果需要输入片外数字信号，则有两种思路：使用同步信号和不使用同步信号。使用同步信号会造成速度过慢的问题，不使用同步信号则需要将指令周期与收到数据的周期进行精确的对齐，稍有差错就会采集到无效的数据。在这样的环境下，一些学校提出的硬件二值化方案是一种很好的选择。

由于以上的原因，我们认为 S12 并不能很好地满足摄像头组的需求，转而考察 16 位 DSC 系列芯片。DSC 芯片主要分为 DSP5685x 和 MC56F8xxx 两种，它们的

区别在于前者未集成片内 Flash 和一些其它的资源，而后者则集成了。然而，由于集成了片内 Flash 等电路，后者的主频只能达到 60MHz，而前者能达到 120MHz。经过仔细分析与讨论，我们决定使用型号为 MC56F8366 的 DSC，它的主频能达到 60MHz，且官方的对比文档表明其同主频性能远高于 S12。

MC56F8366，以下简称 DSC，拥有丰富的片内资源。其中 32KB Data RAM、正交解码器、外部总线很好地满足了我们的需求。值得注意的是，MC56F8366 使用 3.3V 电源，工作时电流高达 200mA，计算得到其功耗为 660mW。如果使用 LDO 等线性稳压器件，假设电源电压为 8V，则损耗功率为 940mW，从功耗、发热量等角度来考虑都是无法忍受的，因此需要使用开关电源。

4.1.2 3.3V 电源

我们选择了 TI 公司的 PTR08100 模块为整个系统提供 3.3V 电源。其实物图与电路图如图所示。

图 4-1 PTR08100 模块

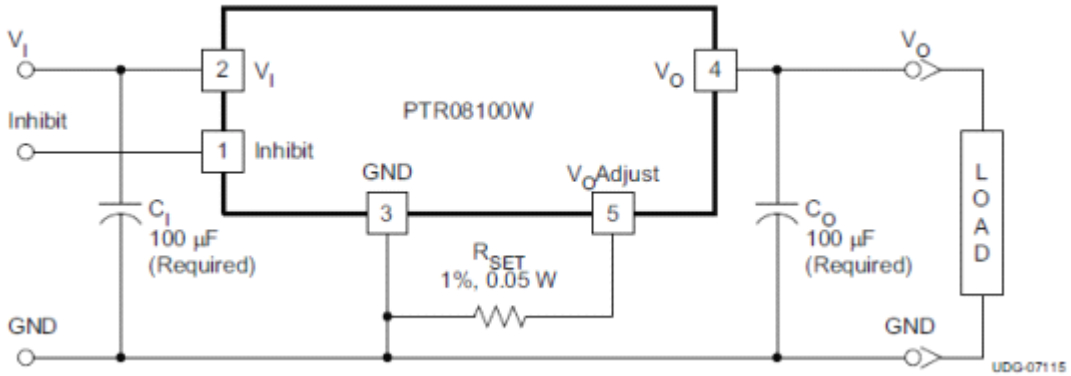


图 4-2 PTR08100 电路图

PTR08100 的输入电压范围为 4.5-14V，输出电压范围为 0.6-5.5V，最小允许

压降为 0.6V，最大输出电流高达 10A，最高转换效率能达到 96%，能很好地满足我们的需求。值得注意的是 RSET 的阻值会直接影响 PTR08100 的输出电压，需要选用精度较高的电阻。输出电压过高或过低都会造成系统无法正常工作。

4.1.3 10.6V 电源

我们选用的 CCD 摄像头需要 9-12V 的电压，有必要使用升压电路对其进行供电。我们选用了 NS 的 LM2731 升压芯片为摄像头提供 10.6V 的电源。LM2731 采用 5-pin SOT-23 封装，体积小巧。然而其开关频率高达 1.6MHz，最大输出电流为 1.8A，能够在给摄像头提供稳定电源的同时，缩减输出端并联电容的容值，从而减小车身的重量。电路图如下图所示：

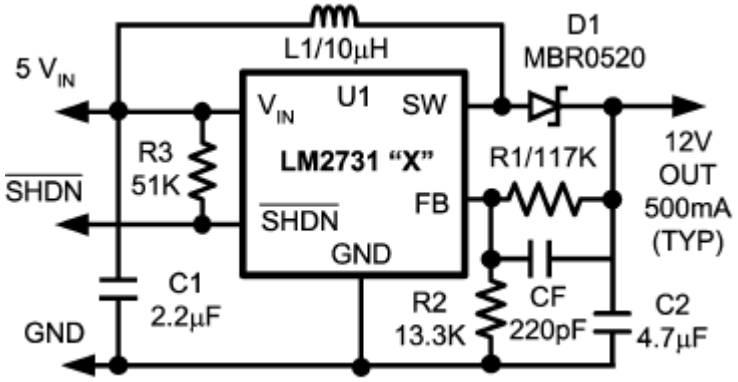


图 4-3 LM2732 升压电路图

4.1.4 模拟视频前端

我们选用的摄像头为模拟摄像头，需要将其输出的模拟信号转换为数字量。我们选用了 TI 公司的 TVP5146 芯片。这是一款称为模拟前端 (Analog Front-End) 的芯片，在一块芯片内集成了同步信号分离、比较电压选取和 AD 转换。输出的信号和常见的数字摄像头模块类似，包含行场同步信号、有效视频指示信号（AVID）、10-20 位数据线和一个数字时钟。其功能框图如下：

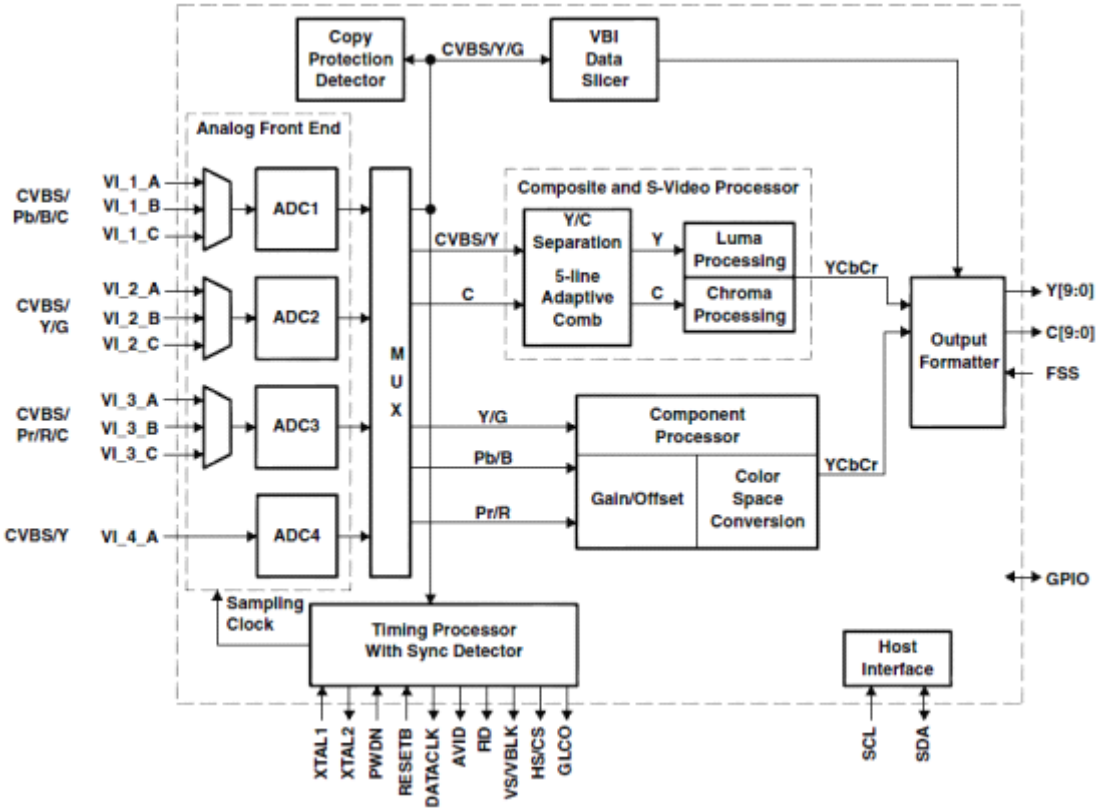


图 4-4 TVP5146 功能框图

4.1.5 1.8V 电源

该芯片核心的工作电压为 1.8V，电流约为 100mA。从缩小体积与重量的角度考虑，我们选用了型号为 LM1117-1.8 的 LDO 为其进行供电，输入为 PTR08100 输出的 3.3V 电源。LM1117 最大输出电流为 800mA，最低压降约为 1V，且输出电压的稳定性非常好，恰好能满足我们的需求。电路图如下：

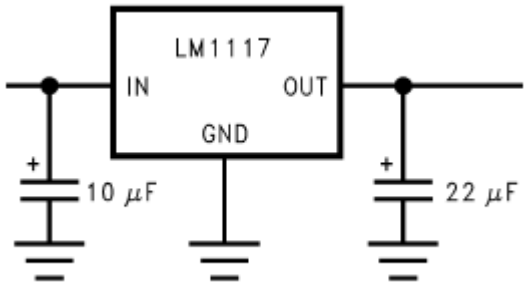


图 4-5 LM1117 电路图

4.1.6 FIFO 和数字分频

为了对 AFE 芯片输出的数字视频信号进行快速读取，需要用到 DSC 的外部总线。然而，DSC 和 AFE 有各自的独立的时钟信号，因此需要在中间加入一个 FIFO。我们选择了 TI 公司的型号为 SN74V235 的 FIFO 芯片，其最高支持频率高达 133MHz，读取时间短至 5ns，能很好地满足我们的需求。为了缩短控制延迟，我们选择按行进行缓冲而不是按列，由 FIFO 造成的延迟短至 64us。因此需要将 AFE 输出的有效视频指示信号连接至 FIFO 芯片的写入使能端。AFE 输出的一行的像素个数约为 690 个，实际并不需要这么高的精度。同时为了编程和调试方便，我们在 RAM 中存储了 3 个帧缓冲区，大大提高了编程的效率（一晚上的时间就搞定了视频采集），然而限制了图像的分辨率。经过各种考虑，我们决定使用 172x40 的图像分辨率。因此，我们使用 74LVC169 计数器芯片对 AFE 输出的视频信号进行 4 分频。

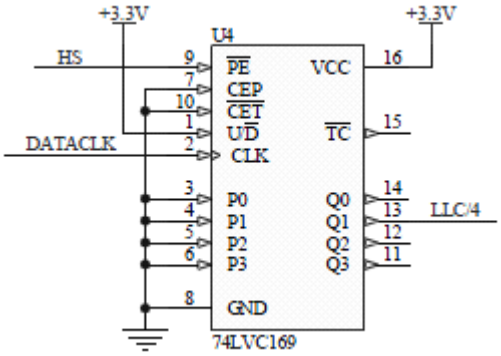


图 4-6 74LVC169 电路图

使用时钟分频来降低分辨率的好处在于电路简单，坏处在于会产生毛刺。根据香农-奈奎斯特采样定理，对信号进行降采样（down-sampling）时，必须先进行低通滤波，否则会产生毛刺。我们使用调节镜头焦距的方式进行近似的低通滤波，将镜头焦距调整至恰好能看清近处的物体而不产生毛刺处并固定。

4.1.7 按钮和蜂鸣器

为了调试方便，我们设计了按钮和蜂鸣器电路，能很好地满足调试的需求。

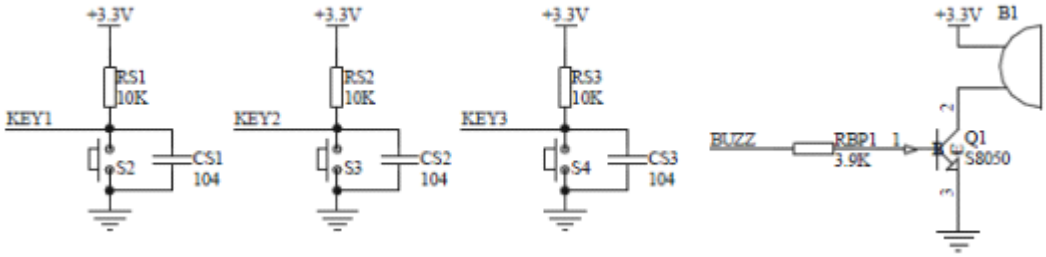


图 4-7 按钮、蜂鸣器电路图

4.1.8 外围接口

主板需要和一些外部设备进行连接，因此需要留出一些接口。由于这是一个私人使用的小系统，接口不需要遵循公共的标准，只要定义清晰、插拔方便、不容易混淆或反插即可。

需要留出的接口如下：

一个 JTAG 接口，14pin，用于连接 JTAG、OSBDM 等烧写器对单片机进行编程。

两个两相编码器接口，4pin，用于连接两个两相增量式光电旋转编码器。

一个带电源的 SPI 接口，6pin，用于连接支持 SPI 总线协议的外围设备。

一个带电源的 SCI 接口，4pin，用于连接支持 SCI 总线协议的外围设备。

一个带电源输入、2 路 GPIO 及 4 路 PWM 信号输出的接口，10pin，用于连接电机驱动板，并从电机驱动板获取电源。

一个带电源和一路 PWM 的接口，3pin，用于连接舵机。

4.2 显示板

由于主板体积限制，LED 数码管找不到足够大的容身之地，我们单独做了一小块电路板，使用四位 LED 数码管进行状态指示。

我们选择了 MAX6954 芯片对数码管进行驱动。MAX6954 的电源电压范围 2.7-5.5V，内置时钟产生电路，使用 SPI 总线协议与主板进行通信，传输速率高达 26Mbps。只要配置好该芯片内部的寄存器，就能自动对四个七段 LED 数码管进行扫描显示，具有连线少，驱动效果好等优点。这部分电路图如下：

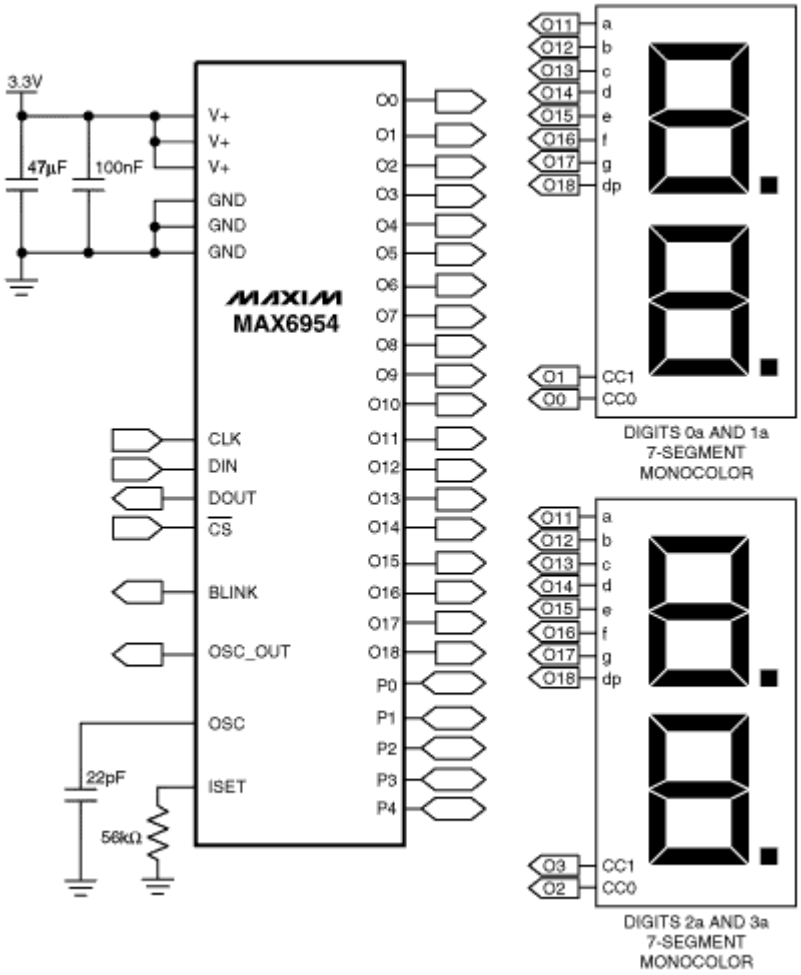


图 4-8 MAX6954 电路图

值得注意的是，由于 MAX6954 支持很高的时钟速率，因此也很容易受到干扰。测试表明，如果使用较长的排线进行连接，会出现传输不正常的情况。我们直接使用排针将显示板插在主板上，工作较为稳定。

4.3 驱动板

单片机输出使用两种模拟电平来表示 0 和 1 的数字信号，然而输出电流很小，无法直接连接直流电动机之类的大电流器件，因此需要使用驱动电路。为了能让直流电动机在四个象限内工作，需要使用 H 桥电路。

我们在之前的主板电路上投入了大量的开发时间。为了缩短开发周期，我们使用了成熟的 BTS7960B 芯片进行电机驱动。一片 BTS7960B 是一个半桥，使用两片 BTS7960B 芯片组成一个 H 桥驱动一个电机。C 车模有两个电机，这样，

就需要四片 BTS7960B。

BTS7960B 的负载电流可以达到 43A，而内阻为 16mΩ。BTS7960 是一款针对电机驱动应用的完全集成的大电流半桥芯片。它在一个封装中集成了一个 N 通道场效应管下桥臂和 P 通道场效应管上桥臂以及一个控制集成电路。由于上桥臂采用的是 P 通道开关，对于电荷泵的需求也就不复存在了，因此电磁干扰减至了最小。同时因为该芯片内部的驱动控制集成电路具有逻辑电平输入，使得与微控制器的接口变得很方便，而且该驱动集成电路还具有电流检测诊断、转换率调整、死区时间生成以及过热、过压、欠压、过流和短路保护等功能。

针对一个电机的 H 桥驱动电路图如图所示：

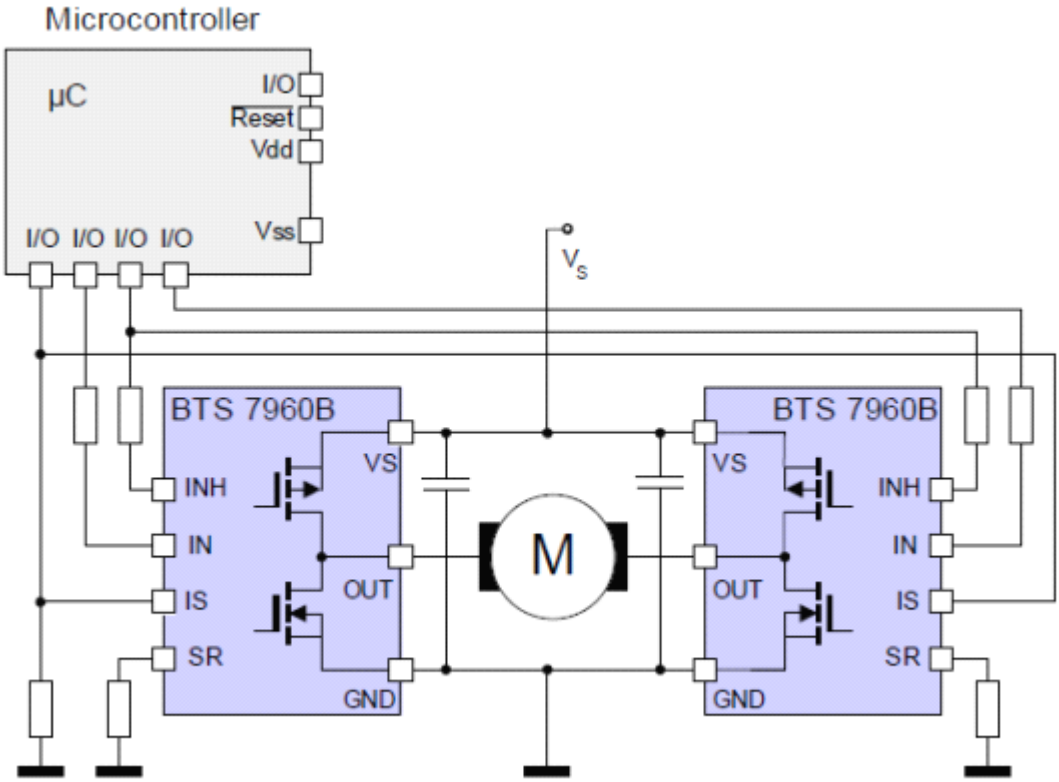


图 4-9 BTS7960 功能示意图

上图中两片 BTS7960B 组成的电路有三路输入信号，分别对应使能和两个半桥。我们将使能信号连接至 DSC 的 IO 口，半桥控制信号连接至 DSC 的 PWM 端口，通过调节输出 PWM 信号的占空比来实现对电机力矩的控制，上层通过一个闭环反馈实现速度控制。

4.4 摄像头

摄像头是智能小车系统信息提取关键，其信息输出信息的好坏将首先决定小车的性能。因此摄像头的选取必须慎重，既要保证图像质量好，满足后续处理和赛道识别的要求，又要考虑到单片机采集和处理的能力。对于由单片机组成的小系统来说，摄像头的分辨率不是越高越好。因为这样只会徒增单片机的负担。

摄像头由镜头、图像传感芯片和外围电路构成。而图像传感芯片是摄像头最重要的部分。

摄像头按图像传感器分可分为 CCD 图像传感器和 CMOS 图像传感器，CCD 图像传感器通常比 CMOS 图像传感器高 10 倍的感光度（ISO）。人眼能看到 1Lux 照度（满月的夜晚）以下的目标，CCD 图像传感器通常能看到比人眼略好，大约能看到在 0.1~3Lux 照度以下的目标，是 CMOS 图像传感器感光度的 3 到 10 倍。CMOS 图像传感器的感光度一般在 6Lux 到 15Lux 的范围内，CMOS 图像传感器有固定比 CCD 图像传感器高 10 倍的噪音，固定的图案噪音始终停留在屏幕上好像那就是一个图案，因为 CMOS 图像传感器在 10Lux 以下基本没用。但是 CMOS 图像传感器可以将所有逻辑和控制环都放在同一个硅芯片块上，可以使摄像头变得简单，因此 CMOS 图像传感器可以做得非常小、便于携带。同时 CMOS 图像传感器非常快速，比 CCD 图像传感器要快 10 到 100 倍，而且 CMOS 传感器不需要复杂的处理过程，可直接将图像半导体产生的电子转变成电压信号。

而根据摄像头输出信号的形式摄像头分为数字摄像头和模拟摄像头。数字摄像头是一种数字视频的输入设备，利用光电技术采集影像，而不像视频采集卡那样首先用模拟的采集工具采集影像，再通过专用的模数转换组件完成影像的输入。数字摄像头的优点是使用简单，输出即为数字信号。模拟摄像头多为 CCD 的，按不同档次分辨率不同。与数字摄像头同级的模拟摄像头一般有较高的分辨率，较好的实时性。模拟摄像头要与单片机相连必须先经过视频解码芯片进行解码，然后再交予单片机进行处理。

我们的主板上具有模拟视频输入接口，因此我们选择模拟摄像头。经过一些比较，我们选择了一款模拟针孔 CCD 摄像头。如图：

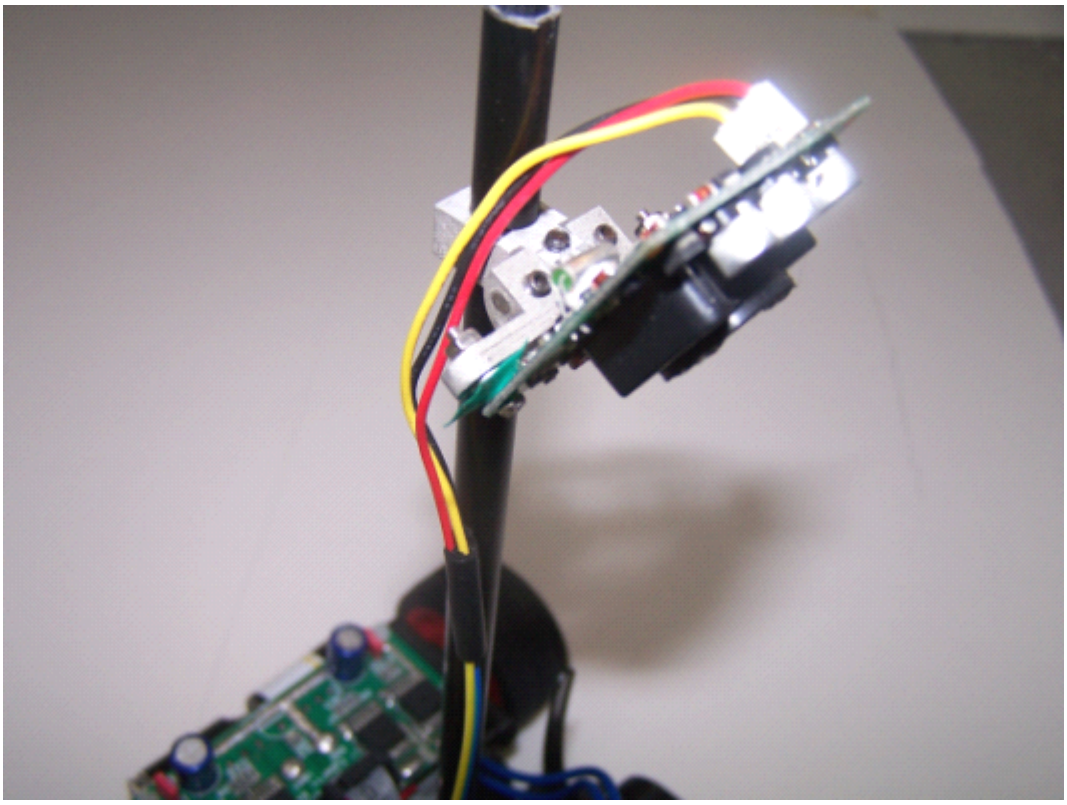


图 4-10 摄像头

针孔摄像头的优点在于镜头较轻，且采光面积小，容易适应光照较强的环境，缺点在于图像对比度较差。采集到的图像如图所示。

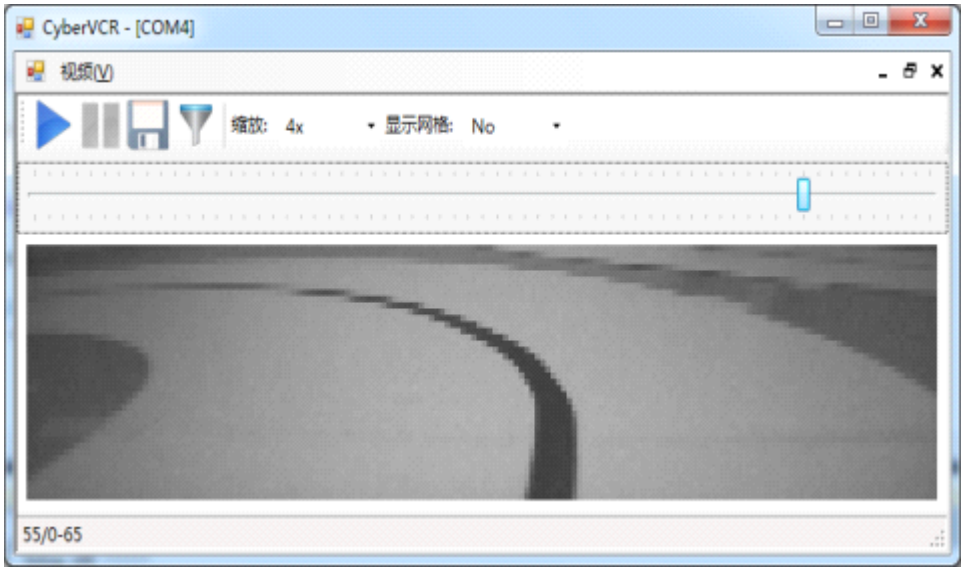


图 4-11 摄像头采集图像

4.5 旋转编码器

我们的车模的两侧后轮被两个独立的直流电动机进行驱动，在转弯时，内侧轮的速度应低于外侧轮的速度。为了分别精确地测得两轮的速度，需要使用旋转式测速传感器。

我们选用了浙江温州欧姆龙 OMRON 生产的两相光电旋转编码器，型号为 E6A2-CW3C 500P/R，旋转一圈时，两路输出均可以产生 500 个周期的方波，它们之间存在正负 1/4 个周期的相位差，分别对应正转和反转的情况。

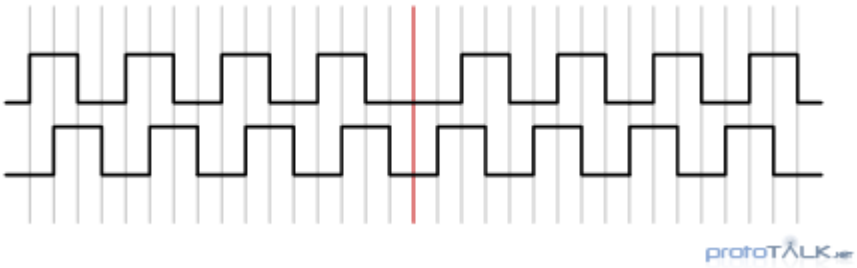


图 4-12 两相光电编码器输出示意图[14]

图片中显示了两相编码器输出信号的波形。注意中间的红线处经历了一次从正转到反转的过程。

将该编码器直接连接至主板上的两相编码器输入接口，由 DSC 内置的正交解码器（Quadrature Decoder）进行计数，由于每个周期存在四个边沿，可以进行四次计数，因此编码器正转一圈计数值增加 2000，反转一圈计数值减少 2000，测量的精度能满足我们的需要。正交解码器的逻辑真值表（Truth Table）如下：

表 4-1 正交解码器的逻辑真值表[14]

Current State			Next State			protoTALK.net
CH A	CH B		CH A	CH B		Direction
HIGH	HIGH		HIGH	LOW		Reverse
			LOW	HIGH		Forward
HIGH	LOW		HIGH	HIGH		Forward
			LOW	LOW		Reverse
LOW	HIGH		LOW	LOW		Forward
			HIGH	HIGH		Reverse
LOW	LOW		HIGH	LOW		Forward
			LOW	HIGH		Reverse

4.6 伺服电机

伺服电机，下称舵机，负责控制前轮的转向，对整车的沿线行驶起到了重要的作用。根据今年的比赛规则，控制前轮转向用的舵机必须使用 Futaba S3010。

Futaba S3010 由日本双叶公司出品，S 表示舵机，3 表示它用的是三级马达，0 表示是泛用型，10 指此舵机为模拟电路控制舵机。当工作电压为 6.0V 时，该舵机的扭矩为 6.5kg.cm，转动速度为 0.16s/60deg。

该舵机接受脉冲信号作为控制信号。舵机每捕捉到一次脉冲，就根据脉冲的宽度，通过内置的闭环控制系统旋转到相应的位置。脉冲之间的间隔不能太短，否则会导致舵机工作不正常，通常取 10-20ms。不少参赛队使用单片机的 PWM 模块来控制舵机，使单片机输出一个 50-100Hz 的 PWM，通过调节 PWM 的占空比来实现舵机的控制。

然而，经过分析，我们发现这种控制方法不利于采用摄像头传感器的车模进行寻线行驶。摄像头传感器频率通常为 50-60Hz，每 20ms 左右获得一幅完整的图像，时间常数较大。一幅图像处理完成后，通常需要改变舵机的转角。如果使用 PWM 来控制舵机，由于周期固定，改变占空比的时刻会位于一个 PWM 周期中间的某个位置。假设一个 PWM 周期长达 20ms，改变占空比的时刻位于 PWM 周期中任一位置的概率相等，那么可以发现，平均的控制延迟徒增了 10ms。

考虑到舵机控制频率和摄像头的信号频率相近，如果能让舵机和摄像头同步，就能缩短控制延迟，提高控制的效果。为此，我们将舵机连接至 DSC 的四时钟（Quad Timer）模块而非 PWM 模块。该模块共有 12 种工作模式，其中的 Pulse Output Mode 能满足我们的需求，输出一次指定宽度的脉冲。

4.7 直流电机

采用规则中指定的两个 RN260-CN 38-18130 直流电机为后轮提供动力。

第五章　控制软件设计说明

5.1 DSC 系统片内资源

MC56F8366 DSC 芯片作为 MC56F83xx 系列 16 位 DSC，把 DSP 和 MCU 的功能集中到一个单一的、高效的架构上，包括 16 位 56800E 内核、512KB 程序 Flash 存储器、32KB 的数据 Flash 存储器、32KB 的启动装载 Flash 存储器、4KB 的程序 RAM、32KB 的数据 RAM、至多两组 6 通道脉宽调制模块（PWM）、四组 4 通道 12 位模数转换器、温度传感器、至多两路正交解码器、至多两个 FlexCAN 总线接口、两个异步串行通信接口、至多两个串行外围接口、至多 4 路通用正交计时器、看门狗、JTAG 实时调试接口、至多 62 路通用 I/O 端口。该 DSC 在 60MHz 的频率下性能指标可以达到 60MIPS。

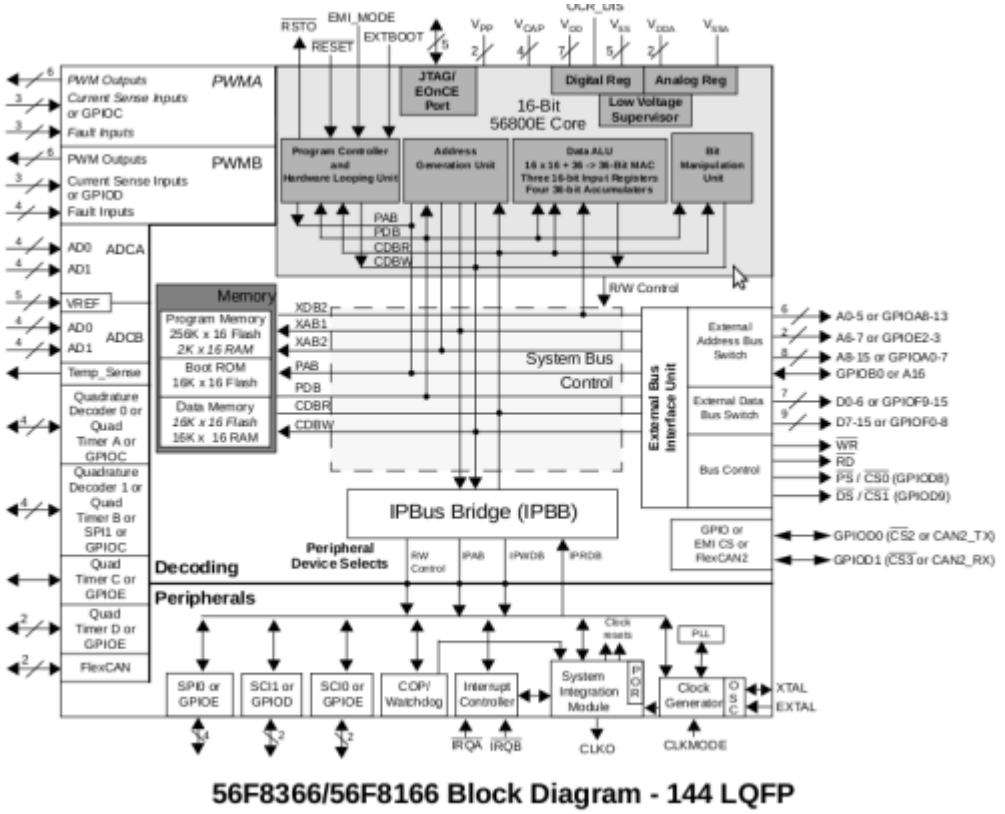


图 5-1 MC56F8366 功能模块图

具备了以上这些卓越的性能，MC56F8366 完全能够满足我们设计的需要，

并使得电路更加紧凑。以下将首先介绍系统设计中用到的各个软件功能模块的设置，然后再讨论黑线提取及车体控制算法。

5.2 底层组件

我们使用Processor Expert来生成与DSC片内资源交互的底层代码。Processor Expert是一个用于开发、配置、优化、集成和分发基于Freescale平台的软件组件的开发系统。它集成于Freescale公司的CodeWarrior开发环境中，支持S08/RS08、S12(X)、Coldfire、Coldfire+、Kinetis、DSC 56800/E以及一些Power架构的处理器。

使用了Processor Expert，我们很快就配置好了底层的DSC片内资源，配置好的软件界面截图如下。

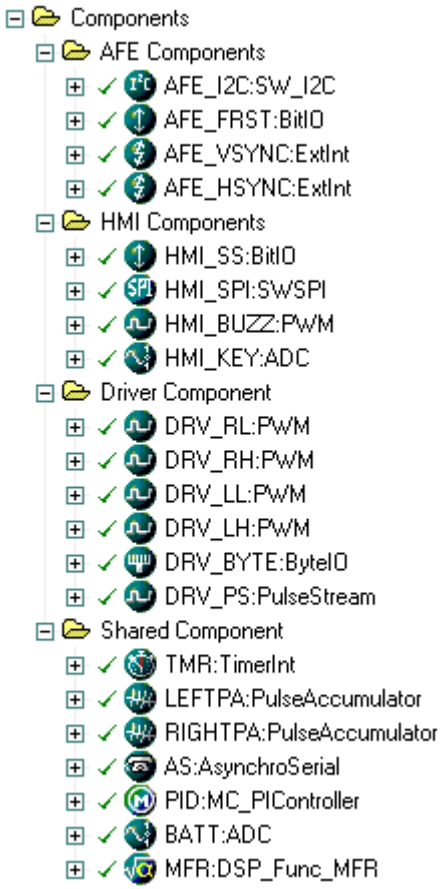


图 5-2 PE资源配置

使用了Processor Expert，整个系统结构十分清晰，缩短了开发周期，减小了配置寄存器出错的概率，提高了系统的可靠性。从图中可以看出，整个系统的

底层资源分为 AFE、HMI、Driver 和其它共享组件。

5.3 黑线位置提取

得到一场视频信号后，可以从中提取出黑线位置、是否为起始线等各种信息。这些信息可以被用于对舵机和电机等执行器件进行控制。

采集到的图像分辨率为 172x40，主要包含前景和背景两部分。前景是指赛道上的黑色部分，包括黑线和起始线。背景指赛道未贴黑线的白色部分，以及赛道外的其它部分。

将数字图像分为多个子区域被称为图像分割。图像分割的目的是简化或改变图像的表示形式，使得图像更容易理解和分析。[13]图像分割通常用于定位图像中的物体和边界（线，曲线等）。更精确地，图像分割是对图像中的每个像素加标签的一个过程，这一过程使得具有相同标签的像素具有某种共同视觉特性。

现已有许多各种用途的图像分割算法，包括但不限于聚类法、直方图法、边缘检测、区域生长、水平集方法。对于图像分割问题没有统一的解决方法，这一技术通常要与相关领域的知识结合起来，这样才能更有效地解决该领域中的图像分割问题。

我们选择了边缘检测的方法来进行黑线提取。在尝试了各种各样的边缘提取算子（包括但不限于 Sobel、Laplacian、Canny、Canny-Deriche[11]、Compass[12]）之后，针对智能车竞赛的图像特性，使用的边缘检测算法是简单的一阶微分，在离散状况下即一阶差分。

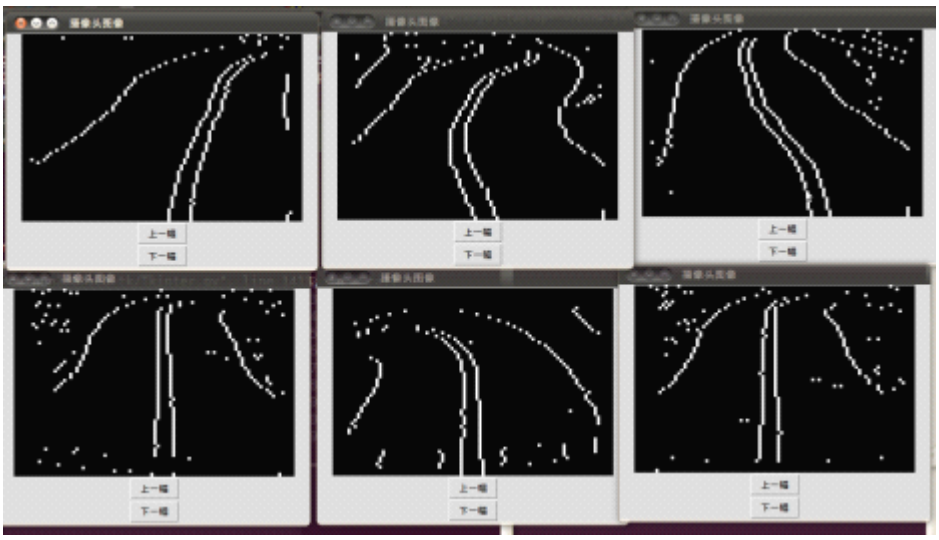


图 5-3 边缘提取结果

一阶差分最大的问题就是对噪声的敏感。尽管 Sobel 算子可以在竖直方向上对图像进行一定程度上的模糊，但是我们所需要的是水平方向上的边缘信息，意义不大。而 Canny 算子及其各种简化版、改进版在水平方向上的高斯模糊确实可以减小对噪声的敏感。但是，即使对于计算能力，特别是乘法运算能力远超 S12 的 DSC 芯片来说，这样的计算量还是过大，特别是考虑了之后其它的算法之后。最后，在改善图像质量后，选择了一阶差分的方法。考虑到黑线可能存在边缘模糊，隔三个点进行差分。查找边缘使用的阈值是动态给定的，使用的是一整行差分的平均值。实验证明，这种动态阈值能较好地适应各种光线情况，特别能适应光线不均匀的情况。

边缘检测可能存在的问题是，对于一个边缘造成多次检出。对于一个边缘只能有一个反应，这也是 Canny 提出的对于边缘检测算子的三个性能指标之一。[10]这个问题对于后面的黑线位置提取的成功率影响并不大，但是可能影响到提取出来的黑线位置，造成控制的不准确。在参考了 Canny 算子后，我们使用了其中的非极大值抑制思想。[10]

在进行全行边缘检测运算之后，对于检测出的由上升沿和下降沿组成的候选黑线，使用一种记分的算法来进行筛选。记分算法根据黑线的宽度、位置、位置的变化率、位置的二阶导来对黑线进行评分，再用一个固定的阈值来进行评判。实践证明可以很好地选择出正确的黑线。识别出来的黑线的中心位置以及所给的分数如图所示。

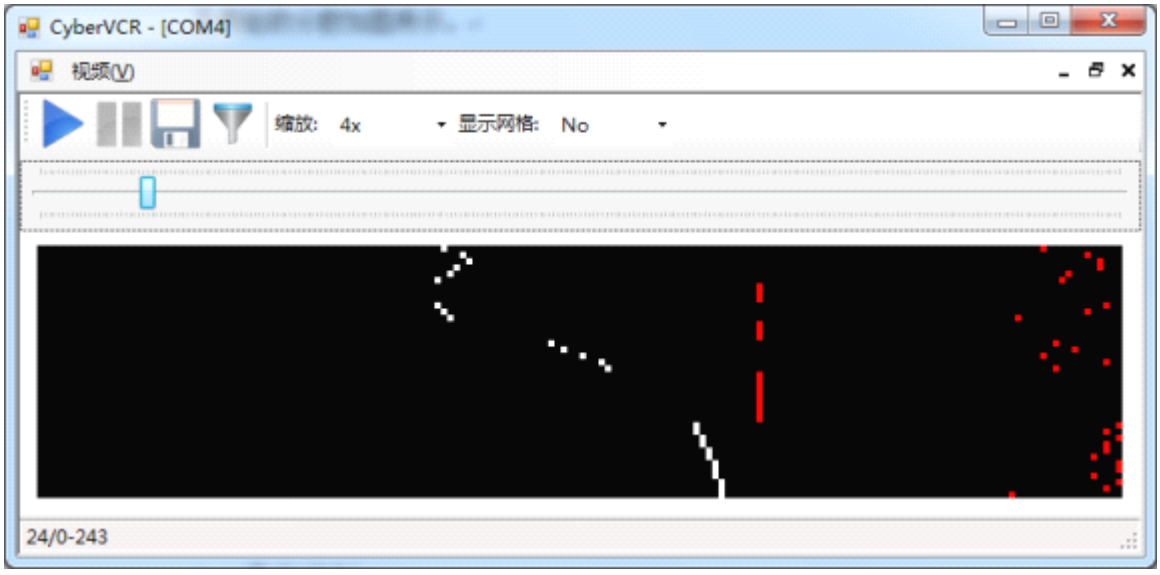


图 5-4 黑线中心及分数

对于今年新增加的虚线，采用的是容忍丢线的方法。即设定一个容忍数值，如果在这个范围内用记分算法能找到黑线就连上，使用线性插值补全其中的点。而容忍数值的确定，与当前行数、找到黑线的位置都有关。

剩下的需要从视频信号中提取出来的信息就是起始线。起始线的检测非常简单，就是在扫描边沿的时候顺便记下符合起始线宽度要求的黑线，在最终找到黑线后判断起始线与黑线两端的距离是否符合要求。在测试中发现，由于切弯的问题，在经过起始线时小车有可能偏离黑线中心很多，此时看不到完整的起始线。所以最后再记录了比较靠近边缘的边沿，并根据黑线中心、另一边是否有确定的起始线标志来判断起始线。起始线图像看起来是这样的：

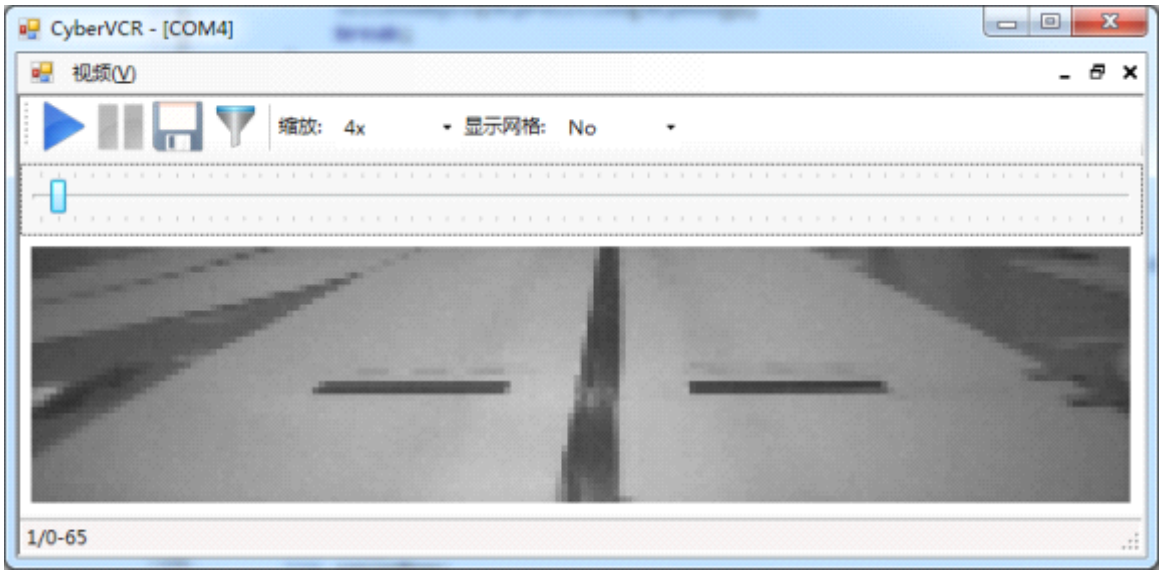


图 5-5 起始线图像

5.4 速度控制

分层是解决系统复杂性的有效方法。我们的速度控制算法分为顶层和底层，顶层负责根据传感器的信息给定一个速度，底层负责通过闭环反馈来进行速度控制。

5.4.1 顶层速度控制

我们采用了一种十分简单的顶层速度控制算法，即两段法。在采集到的图像中，如果黑线能一直延伸到最远行，就给高速，否则就给低速。低速设置为赛

道上曲率最大弯道的极限速度。实践表明使用这种方法进行速度分配，直道、单向弯道和 S 弯道均能安全通过。

然而，这种方法过于保守，在通过曲率较大的弯道时，有可能也会给低速。我们又对两段法进行了一些改进。增加了对控制行数变化的前馈，形成了三段速度。这是一个保守和激进的折中，给定过慢或过快的速度都会造成成绩降低。

我们还尝试了一些基于曲率的算法，发现在现有的图像分辨率（172x40）中，很难提取到有用的信息，最终放弃了这一方案。

5.4.2 底层速度控制

底层速度采用的是简单的 PID 控制。今年 C 车的电机性能相当差，而我们认为如果使用 bang-bang 控制将会造成电机性能的进一步下降，极大的降低了加速度和最高速度。而且由于电机加速性能差，需要更加平滑的控制。在经过一定的实验，综合各方面的因素之后，选择了 PID 控制。

考虑到电机的非线性特性，我们在根据 PID 得出的 PWM 值后，进行了一个非线性的二次曲线的转换。经测试，在加入这个转换后，在正反两个方向上，电机都能在更多的时间打满，从而能使加减速性能得到提升。

5.4.3 速度获取

在每一速度控制周期开始，读取由通用正交计时器实现的脉冲累加器中的数值，与 10ms 前得到的数值相减，得到 10ms 时间里共有多少个脉冲数累积，从而由可求得赛车速度值。由于速度控制周期实际上只有 0.5ms，这种做法相当于对 20 个周期内的速度进行了均值滤波，可以得到比较平滑的结果，减小由于增量式两相光电旋转编码器质量问题造成的读数抖动。

由于使用的是两相编码器配合正交计时器模块，可以从最后相减得出的数值直接判别出正反转。

5.5 舵机控制

5.5.1 舵机顶层控制量选定

简单沿线行驶策略的目标是控制舵机使得赛车尽可能沿着导引线前进。所以舵机顶层控制量选用的是某一行黑线的中心位置，控制行的选取与速度进行线性耦合。

5.5.2 舵机顶层控制量修正

对于 S 弯道的最佳行驶路线是沿着中心线行驶，这样可以大大提高赛车速度，缩短行驶时间。如何能做到这一点呢？一般的想法是将 S 弯和普通弯道、直道区分开来。这就要进行模式识别与决策。但是，识别存在着出错的可能，万一识别出错，小车将会很容易冲出赛道。而且，使用判断并分开进行单独控制的方法，与采用统一的方法进行 S 道直冲、普通弯道切弯相比，显得比较丑陋。

所以我们最后采用的是较为简洁、优美、统一的方法，通过对顶层控制量的修正，来一并完成这几项任务。修正使用的信息是控制行之上的黑线位置的加权平均值。

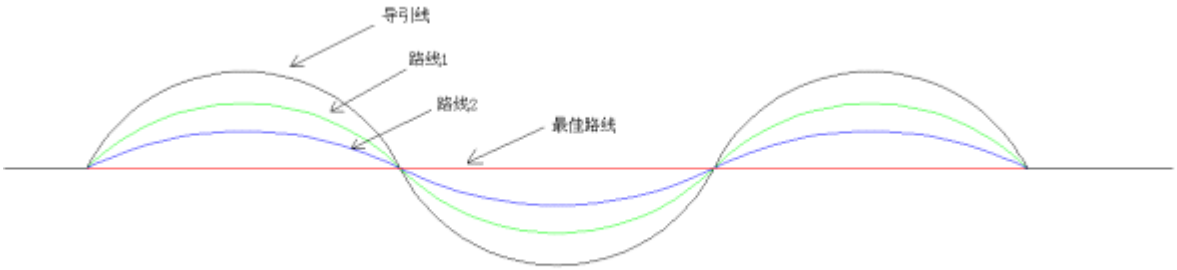


图 5-6　S 道路线图

第六章 开发工具、制作调试过程说明

6.1 软件调试平台

Codewarrior 是由 Metrowerks 公司提供的专门面向 Freescale 所有 MCU 与 DSP 嵌入式应用开发的软件工具。其中包括集成开发环境 IDE、处理器专家、全芯片仿真、可视化参数显示工具、项目工程管理、C 交叉编译器、汇编器、链接器以及调试器。其中在本设计方案中最为重要的部分就是集成开发环境 IDE 以及调试器，所以接下来将主要介绍该两部分。

6.1.1 Codewarrior IDE 功能介绍

CodeWarriorIDE 能够自动地检查代码中的明显错误，它通过一个集成的调试器和编辑器来扫描你的代码，以找到并减少明显的错误，然后编译并链接程序以便计算机能够理解并执行你的程序。每个应用程序都经过了使用像 CodeWorrior 这样的开发工具进行编码、编译、编辑、链接和调试的过程。

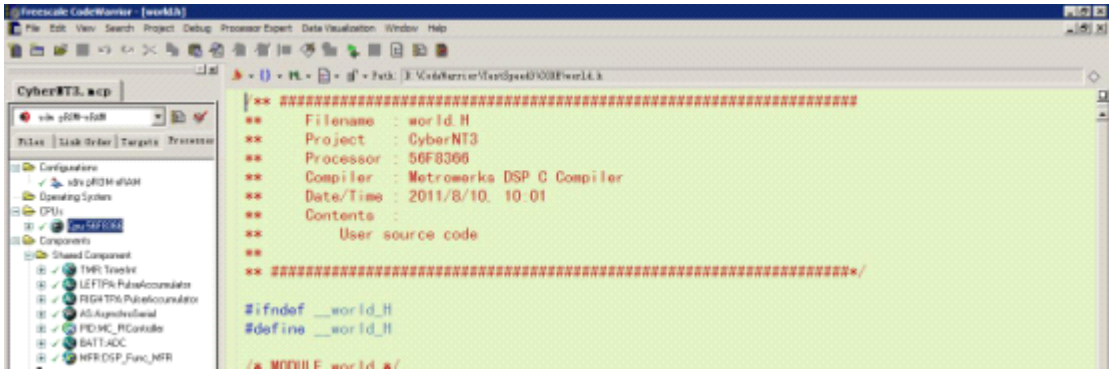


图 6-1 CodeWarrior for DSC56800E v8.3

6.1.2 Codewarrior IDE 基本使用方法

运行“开始菜单—>所有程序—>Freescale CodeWarrior—>CodeWarrior for DSC56800E v8.3—>CodeWarrior IDE”，选择“File—>New”，出现如下的对话框

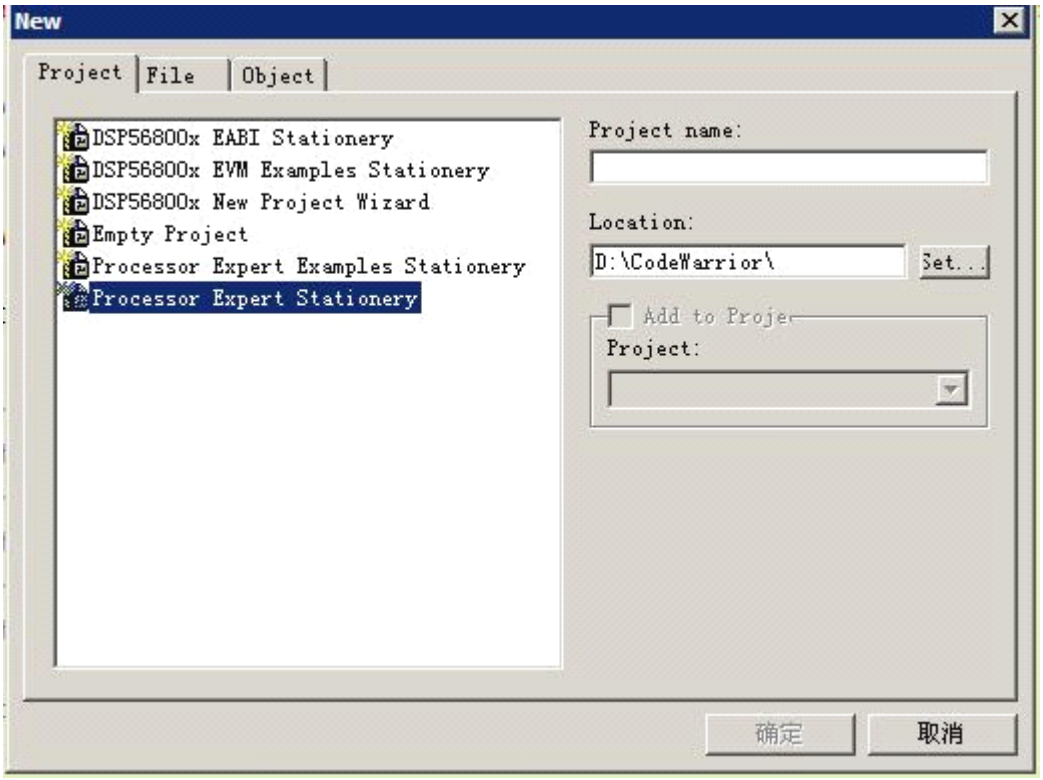


图 6-2

因为我们使用 PE 进行底层的编写，所以选择其中的 Processor Expert Stationery 选项。此后选择 MC56F8366，按照提示选取期望的选项。直至建立工程文件。

新版本的 CodeWarrior for 56800E 集成了烧写、调试环境，无需使用挂载的 HIWAVE，大大方便了程序的烧写与调试。

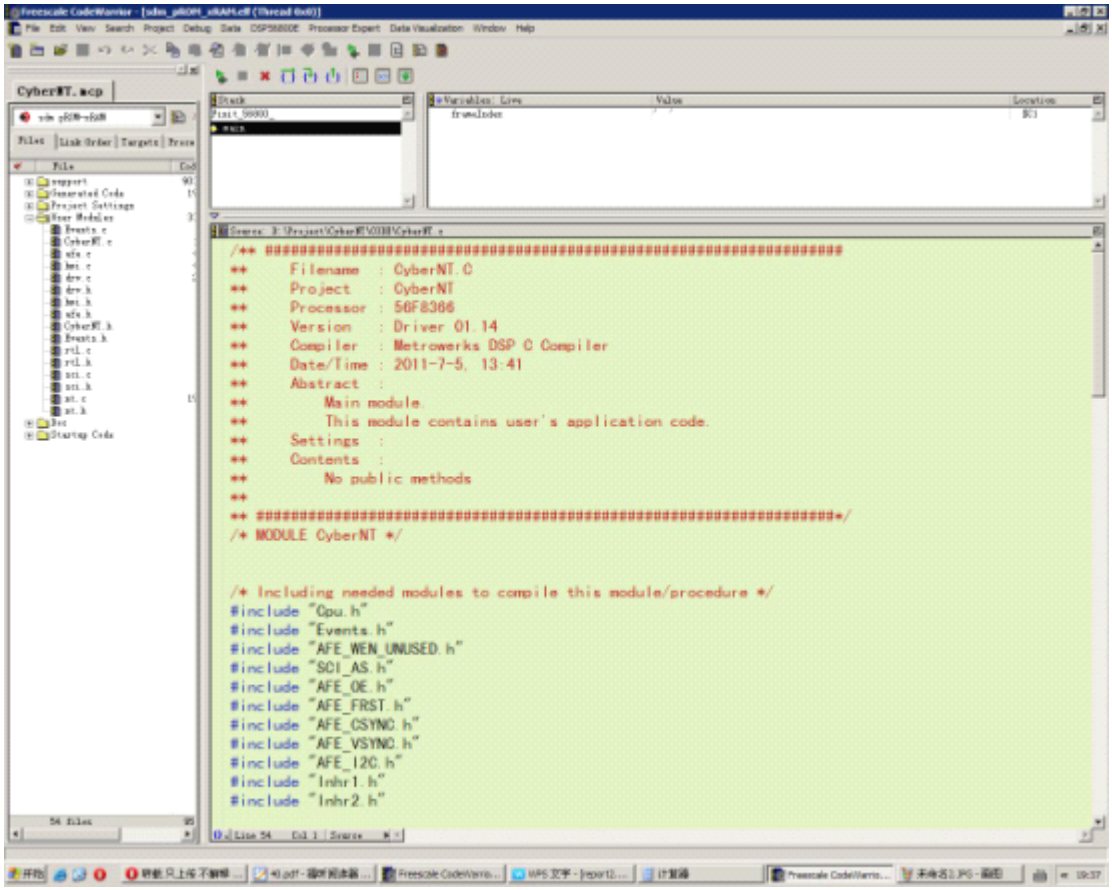


图 6-3 程序调试环境

6.1.3 Processor Expert 的使用

由于 DSC 的初始化设置与 MCU 有所不同，为了减小开发周期，减小消耗在软件底层的时间从而增加算法实现的时间，我们使用了 Processor Expert 来编写软件底层。

Processor Expert 是一个用于开发、配置、优化、集成和分发基于 Freescale 平台的软件组件的开发系统。它集成于 Freescale 公司的 CodeWarrior 开发环境中，支持 S08/RS08、S12(X)、Coldfire、Coldfire+、Kinetis、DSC 56800/E 以及一些 Power 架构的处理器。

Processor Expert 的使用相当方便，相当傻瓜化。能智能的显示出 DSC 所用到的管脚，在管脚冲突时会自动进行提示。

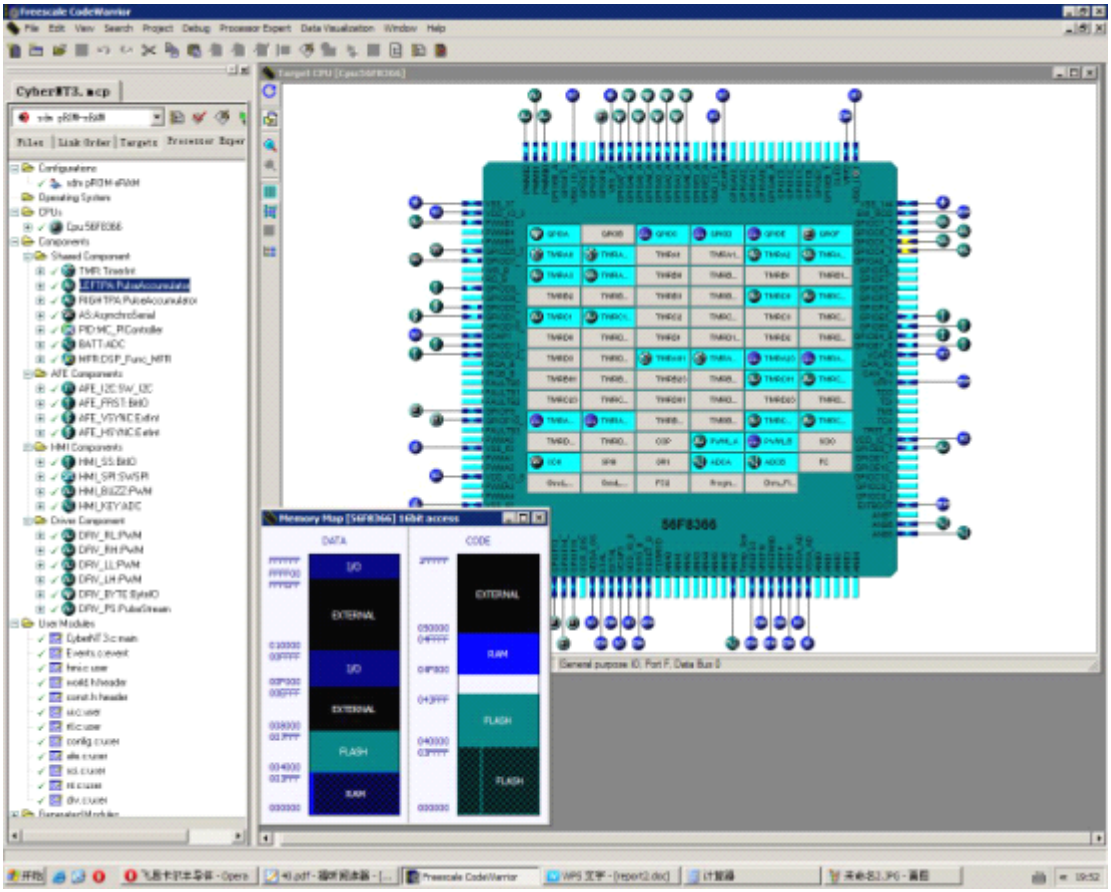


图 6-4 PEDSC 模块、管脚使用图示

6.2 调参数工具

我们使用 LED、按钮和转轮子的方法来进行参数的调整。在软件中层，使用 PE 提供的软件 SPI 编写了 HMI 模块，驱动 LED 显示。在软件顶层，使用面向对象的方法进行 UI 的编写，极大的方便了参数的增加、删除、修改，对于其他功能的扩展也十分方便。使用此 UI 模块，我们可以很方便的加入了显示电池电压的功能。

图 6-5 LED 及按钮

6.3 上位机平台

实际调试过程中为了获得更多的底层数据、信号信息，有必要开发、使用一些辅助的调试工具和方法。我们使用了不同的软件编写了不同的上位机工具，进行各种模块的调试。

6.3.1 CyberVCR

为了方便调试图像与黑线提取模块，我们使用 Visual Basic 编写了 CyberVCR 上位机软件，能通过蓝牙串口将发回来的图像数据或黑线数据显示出来，并支持前进、后退、放大、缩小、保存等功能。

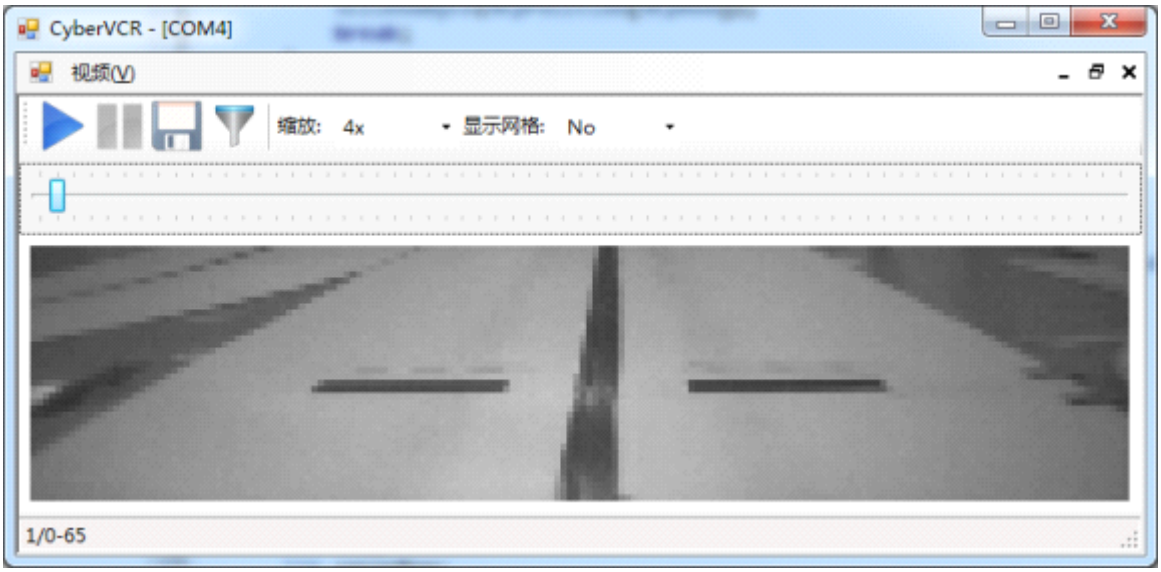


图 6-6 CyberVCR 界面

6.3.2 CyberSpeed

为了方便对速度与方向控制进行调试，我们用 Python 结合 PySerial、matplotlib 模块编写了实时显示并保存小车目标速度、当前速度、当前 PWM、舵机 PWM 数据的上位机软件。同时，该软件还能对保存的数据进行进一步的分析，得出不同的数据图像。

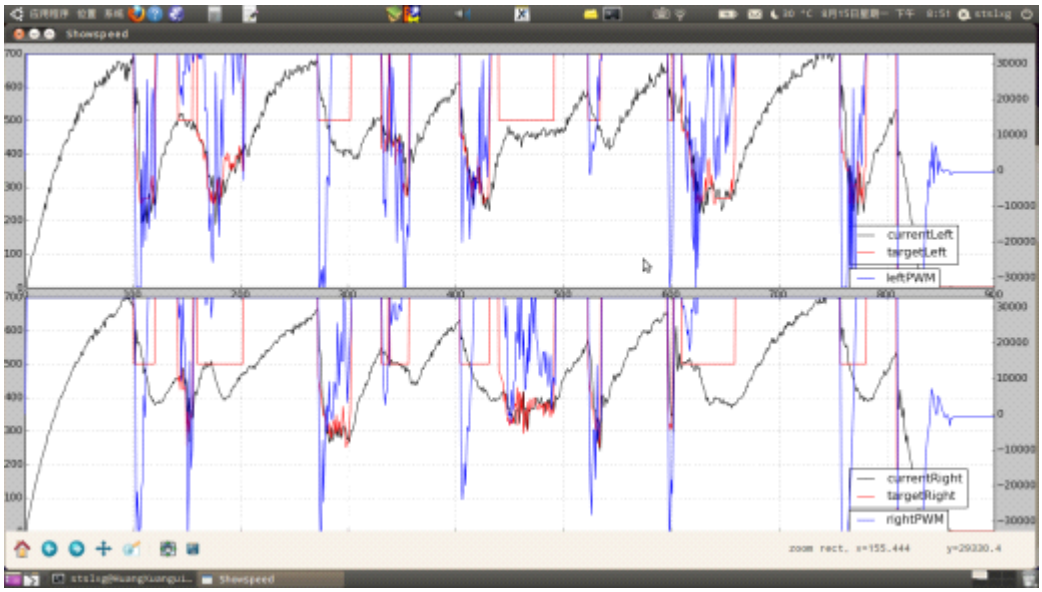


图 6-7 数据图像之一

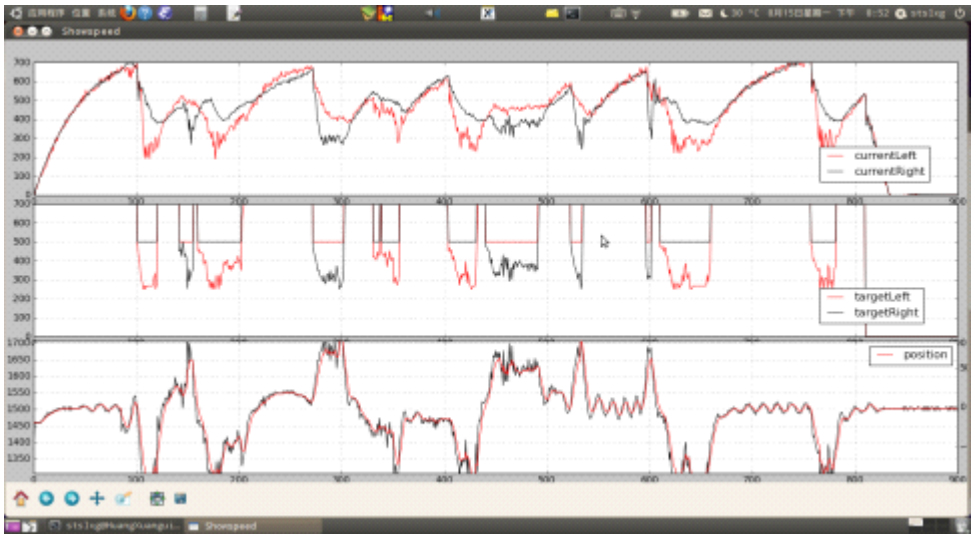


图 6-8 数据图像之二

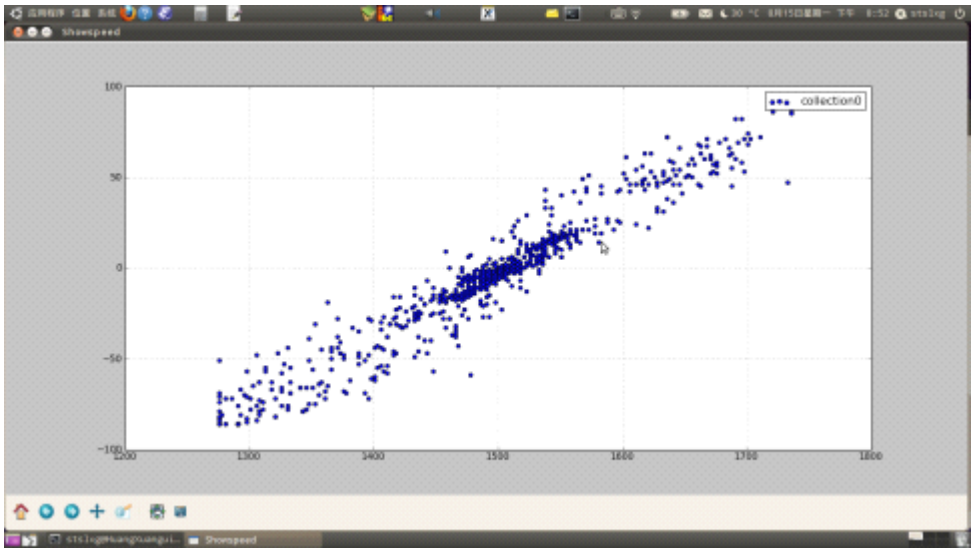


图 6-9 数据图像之三

6.6 计时器

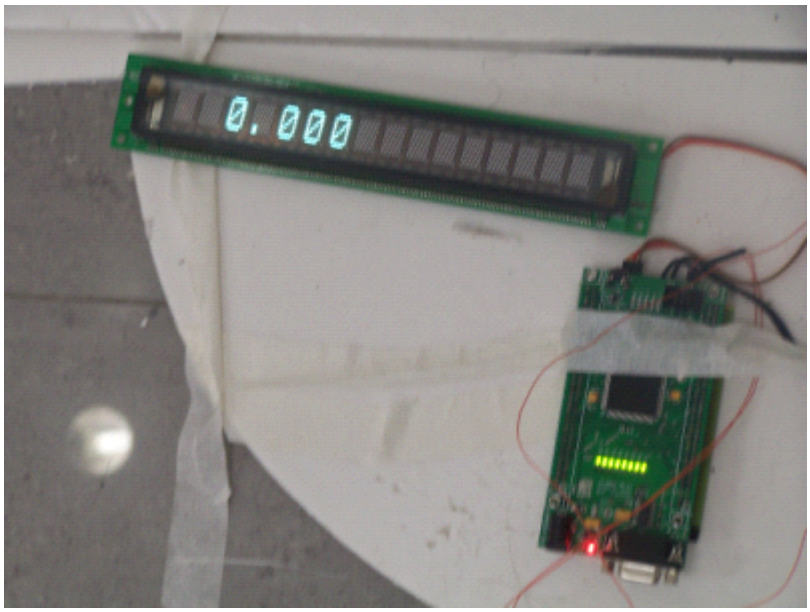
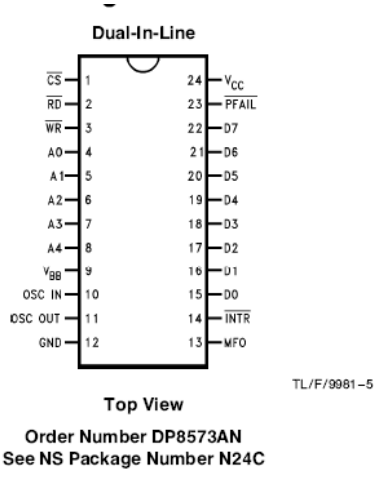


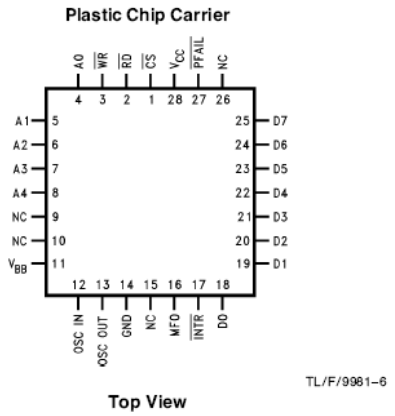
图 6-10 计时器显示

为了方便显示成绩，我们用废旧的清华 S12 培训板的主板制作了计时器。发射管使用的是 180K 调制激光。为了避免被光电组的激光干扰，我们所使用的激光是密码编码激光，即设定一个 16 位的密码，发射管按照这个密码发射编码激光，接收管接收后对激光信号进行解码，只有当解码出来的结果与密码相同时才认为激光管中间无物体存在。

计时器芯片是 DP8573ARTC 芯片。引脚定义如图



Top View
Order Number DP8573AN
See NS Package Number N24C



Top View
Order Number DP8573AV
See NS Package Number V28A

图 6-11 芯片引脚

定时器精度为 0.001 秒。完成计时器程序大致分为两个部分，一个是计时功能实现，另一个就是计时器初始化（端口初始化和显示），最后将光电触发功能加入计时器。为了防止小车通过时多次触发，加入了延时程序（在计时开始后程序主体经过 1s 后才再次响应其他触发），最终的程序相当稳定。

计时器操作方法：

1) 接线：发射管接 VIN01 和 VIN02 不分极性

接收管直接焊在底板上。

2) 显示屏将白色接口插上即可

3) 操作：接上电源，正常工作，注意对射装置要对准。若不正常工作，断开电源在插上。

注意事项：最好不要长时间通电，防止板子过热。

第七章　模型车主要技术参数说明

7.1 改造后的车模总体重量，长宽尺寸

表 7-1 改造后车模参数

类别	参数
车模总体重量(kg)	1.5
车长（mm）	280
车宽（mm）	160
电路功耗(W)	4
电容总量（uF）	1600
传感器个数	1
传感器型号	某针孔摄像头
附加伺服电机个数	0
赛道信息检测精度	1cm
信息检测频率	80MHz

参考文献

[1] Morita T,et al.An Approach to the Intelligent Vehicle[J].1993 IEEE Intelligent Vehicles Symposium, 1993：426-432.

[2] 胡海峰，史忠科，徐德文.智能汽车发展研究[A].西安：西北工业大学，2004

[3] 杜浩,褚利文,张宇腾. 第三届全国大学生“飞思卡尔”杯智能汽车竞赛上海交大speedstar 队技术报告[R]. 上海:上海交通大学,2008.

[4] 史建鹏.汽车转向轮前束与车轮外倾角的设计匹配[A].湖北:东风汽车公司,2005

[5] 韩飞,陈放,戴春博. 第三届全国大学生“飞思卡尔”杯智能汽车竞赛上海交大cybersmart 队技术报告[R]. 上海:上海交通大学,2008.

[6] 梁昆,王韬,丁丁. 第四届全国大学生“飞思卡尔”杯智能汽车竞赛上海交大 SmartStar 队技术报告[R].上海：上海交通大学，2009.

[7] 卓晴，王珽，王磊. 基于面阵 CCD 的赛道参数检测方法. 电子产品世界,2006(4):141-143.

[8] 鲍晓东，张仙妮. 智能交通系统的现状及发展. 交通论坛,2006 年 8 月:15-18.

[9] 党宏社，韩崇昭，段战胜. 智能车辆系统发展及其关键技术概述. 公路交通科技 2002 年 12 月 Vol.19-No.6

[10] Canny John. A Computational Approach to Edge Detection[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1986:679 - 698

[11] R Deriche.Using Canny's criteria to derive a recursively implemented optimal edge detector[J].Springer, International journal of computer vision, 1987

[12] Ruzon M.A., Tomasi C.Color edge detection with the compass operator.IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1999.

[13] Tinku Acharya 等. 数字图像处理——原理与应用.清华大学出版社,2007

[14] Understanding Quadrature Encoding, http://prototalk.net/forums/showthread.php?t=78

[15] 56F8300 Peripheral User Manual, Freescale Semiconductor Co.,Ltd.

致谢

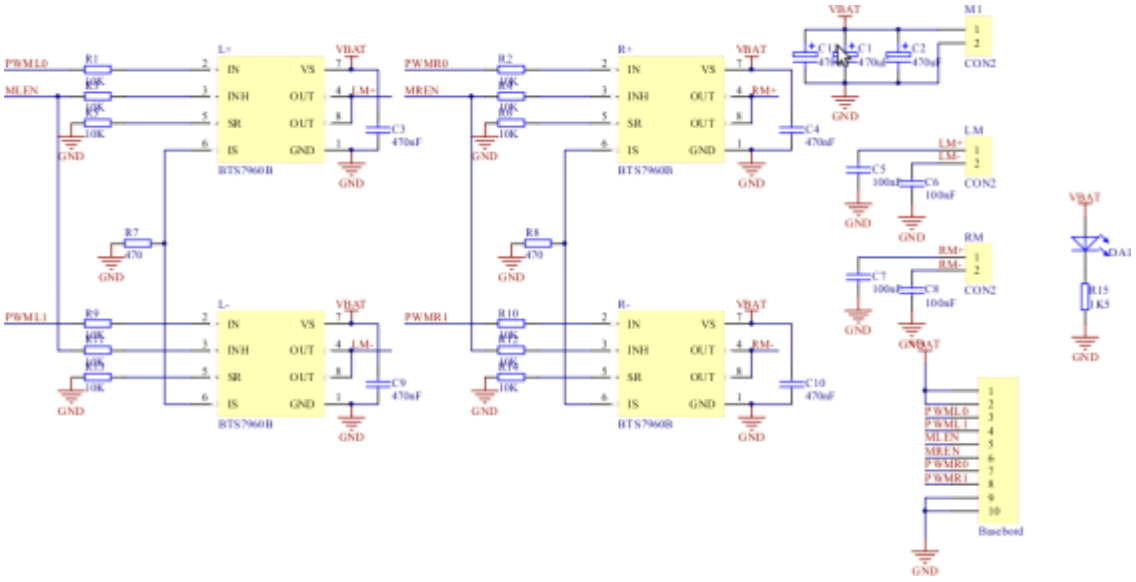
在这次智能车竞赛的准备过程中，我们遇到了很多的困难与挑战。从最初对车模结构的了解与摸索，到对赛车跑动的算法设计，以及对电路原理图的设计绘制与 PCB 的制作，很多事物对我们是新的，很多问题对我们来说是非常有挑战的，我们自身的努力固然重要，但是我们队各个难关的克服，都离不开其他人的帮助。我们要感谢始终支持我们工作的领导、老师、学长和同学，是他们的帮助促成了我们这支队伍的成长，并让我们进入了全国总决赛。

首先，我们要感谢杨明老师、王冰老师和王春香老师，感谢他们对我们工作的悉心指导，他们为我们的工作提供了各种全新的思路，为我们出谋划策，并帮助我们制定了各种方案性的决策，还帮助我们设计和制作了电路结构。没有他们的指导，我们的工作将会进展缓慢，他们帮助我们形成了自己的核心竞争力。

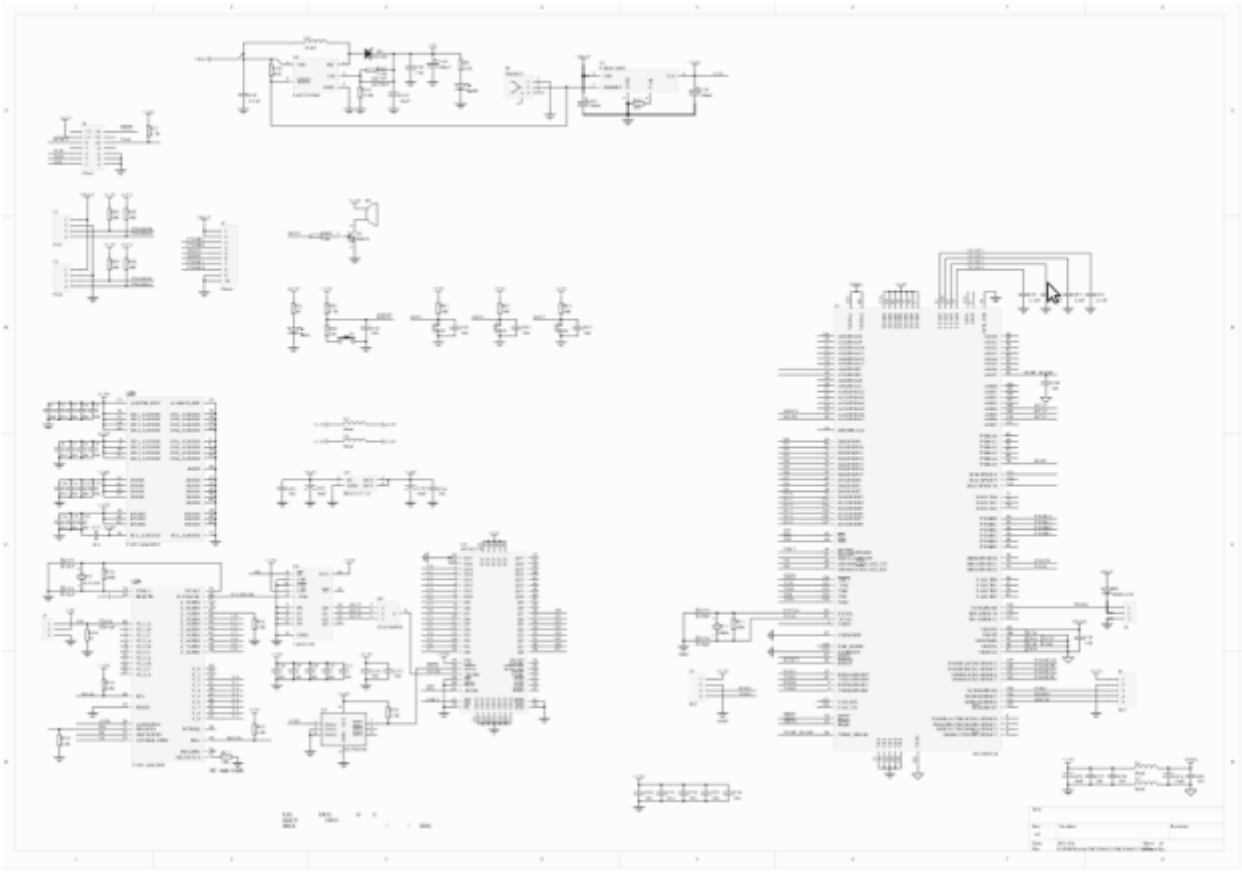
其次，我们还要感谢在实验室帮助我们克服各种困难的学长和实验室其他组别的同学。特别是梁昆、邓刚学长向我们介绍了上上一届、上一届的经验，给我们很大的启发。在车模结构方面，其他组别的同学帮我们解决了很多困难，我们非常感谢他们。

最后，我们要感谢学校和电子工程及电气工程学院领导的支持，他们为我们提供了一个良好的实验室，一个良好的工作环境，还有进行试验所需的经费和各种物质条件。没有他们提供的物质支持，我们的工作将很难顺利进行。

附录 A　　Cyber++ 驱动板原理图



附录 B　Cyber++核心电路板原理图



附录 C　程序代码

主程序

```
/** ###################################################################
**         Filename    : CyberNT3.C
**         Project     : CyberNT3
**         Processor : 56F8366
**         Version     : Driver 01.14
**         Compiler    : Metrowerks DSP C Compiler
**         Date/Time : 2011/8/10, 9:56
**         Abstract    :
**              Main module.
**              This module contains user's application code.
**         Settings    :
```

```
**         Contents  :
**               No public methods
**
** ###################################################################*/
/* MODULE CyberNT3 */


/* Including needed modules to compile this module/procedure */
#include "Cpu.h"
#include "Events.h"
#include "DRV_LH.h"
#include "DRV_LL.h"
#include "DRV_RH.h"
#include "DRV_RL.h"
#include "DRV_BYTE.h"
#include "DRV_PS.h"
#include "AFE_I2C.h"
#include "Inhr4.h"
#include "Inhr5.h"
#include "AFE_FRST.h"
#include "AFE_HSYNC.h"
#include "AFE_VSYNC.h"
#include "PID.h"
#include "BATT.h"
#include "MFR.h"
#include "HMI_SS.h"
#include "HMI_SPI.h"
#include "Inhr1.h"
#include "Inhr2.h"
#include "Inhr3.h"
#include "HMI_BUZZ.h"
#include "HMI_KEY.h"
#include "TMR.h"
#include "LEFTPA.h"
#include "RIGHTPA.h"
#include "AS.h"
/* Including shared modules, which are used for whole project */
#include "PE_Types.h"
#include "PE_Error.h"
#include "PE_Const.h"
```

```
#include "IO_Map.h"
#include "world.h"

static UI_OBJECT ui;

static U8 BatteryTest(U8 dummy)
{
    U16 x;
    UNREFERENCED_PARAMETER(dummy);

    if (BATT_Measure(TRUE) == ERR_OK
        && BATT_GetValue16(&x) == ERR_OK) {
        return (U8)(((U32)x * 100) >> 16);
    } else {
        return 99;
    }
}

static void ServoTest(U8 dummy0, U8 dummy1)
{
    U8 key[2];
    U16 servo = SERVO_MIDDLE;

    UNREFERENCED_PARAMETER(dummy0);
    UNREFERENCED_PARAMETER(dummy1);

    DrvEnableServoControl();
    while (1) {
        HmiShowNumber(servo / 8);
        HmiQueryKeyStatus(key);
        if (key[HMI_KEY_A]) {
            if (servo != 1200 * 8) {
                servo -= 8;
            }
        } else if (key[HMI_KEY_B]) {
            if (servo != 1800 * 8) {
                servo += 8;
            }
        }
        DrvSetValueServo((S16)servo - SERVO_MIDDLE);
```

```
            Cpu_Delay100US(500);
        }
}

void main(void)
{
        /*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! ***/
        PE_low_level_init();
        /*** End of Processor Expert internal initialization.                    ***/

        DrvEnableServoControl();
        HmiInitSystem();
        AfeInitSystem();
        UiInitializeObject(&ui);
        CfgMountUiObject(&ui);
        UiRegisterEntryObject(&ui, 0, &BatteryTest, NULL, 0, 0);
        UiRegisterEntryObject(&ui, 1, NULL, &ServoTest, 0, 0);
        DrvDisableServoControl();

        while (1) {
            UiPromptObject(&ui);
            HmiShowNumberEx(3, 0x0e);
            Cpu_Delay100US(INTERACTIVE_DELAY);
            HmiShowNumberEx(2, 0x0e);
            Cpu_Delay100US(INTERACTIVE_DELAY);
            HmiShowNumberEx(1, 0x0e);
            Cpu_Delay100US(INTERACTIVE_DELAY);
            NtSystemEntry();
        }

        for(;;) {}
}

/* END CyberNT3 */
/*
** ###################################################################
**
**         This file was created by Processor Expert 3.00 [04.35]
**         for the Freescale 56800 series of microcontrollers.
**
```

```
** ###################################################################
*/
```

hmi 程序

```
/** ###################################################################
**         Filename    : hmi.C
**         Project     : CyberNT3
**         Processor : 56F8366
**         Compiler    : Metrowerks DSP C Compiler
**         Date/Time : 2011/8/10, 10:00
**         Contents    :
**             User source code
**
** ###################################################################*/

/* MODULE hmi */

#include "world.h"

static HMI_HANDLER *volatile HmipHandler = NULL;
static volatile U8 HmipKeyStatus[2] = {0, 0};

static inline U8 HmipSendRecv(U8 high, U8 low)
{
    U8 data;
    HMI_SS_ClrVal();
    HMI_SPI_SendChar(high);
    HMI_SPI_SendChar(low);
    HMI_SS_SetVal();
    HMI_SPI_RecvChar(&data);
    return data;
}

static inline void HmipWriteRegister(U8 address, U8 data)
{
    HmipSendRecv(address, data);
}

static inline U8 HmipReadRegister(U8 address)
```

```
{
    HmipSendRecv((U8)(address | 0x80), 0);
    return HmipSendRecv(0, 0);
}

void HmiInitSystem(void)
{
    HmipWriteRegister(0x00, 0x00);
    HmipWriteRegister(0x07, 0x01);
    Cpu_Delay100US(INTERACTIVE_DELAY);
    HmipWriteRegister(0x07, 0x00);
    HmipWriteRegister(0x02, 0x08);
    HmipWriteRegister(0x20, 0x20);
    HmipWriteRegister(0x21, 0x20);
    HmipWriteRegister(0x22, 0x20);
    HmipWriteRegister(0x23, 0x20);
    HmipWriteRegister(0x08, 0xFF);
    HmipWriteRegister(0x09, 0x0F);
    HmipWriteRegister(0x06, 0x5F);
    HmipWriteRegister(0x04, 0x01);
}

void HmiShowNumberEx(U16 number, U8 mask)
{
    U8 reg;

    for (reg = 0x23; reg >= 0x20; --reg) {
        HmipWriteRegister(reg, (U8)((mask & 1) ? 0x20 : 0x00));
        mask >>= 1;
    }
    for (reg = 0x2b; reg >= 0x28; --reg) {
        HmipWriteRegister(reg, (U8)((number % 10) | ((mask & 1) ? 0x80 : 0x00)));
        number /= 10;
        mask >>= 1;
    }
}

HMI_HANDLER *HmiAttachHandler(HMI_HANDLER *handler)
{
    HMI_HANDLER *old = HmipHandler;
```

```
    HmipHandler = handler;
    return old;
}

void HmiQueryKeyStatus(U8 status[2])
{
    status[0] = HmipKeyStatus[0];
    status[1] = HmipKeyStatus[1];
}

static volatile U8 HmipSingleKey;

static void HmipSingleKeyHandler(U8 key)
{
    HmipSingleKey = key;
}

U8 HmiWaitForSingleKey(void)
{
    HMI_HANDLER *old;
    U8 key;

    HmipSingleKey = 0xff;
    old = HmiAttachHandler(&HmipSingleKeyHandler);
    while ((key = HmipSingleKey) == 0xff) {}
    HmiDetachHandler(old);
    return key;
}

void HmiIrqHandler(void)
{
    U16 data[2];
    U8 key;
    HMI_HANDLER *handler = HmipHandler;

    if (HMI_KEY_GetValue16(data) == ERR_OK) {
        for (key = 0; key < 2; ++key) {
            if (data[key] < KEY_AD_THRESHOLD) {
                if (HmipKeyStatus[key] == 0) {
                    if (handler != NULL) {
```

```
                    (*handler)(key);
                }
                HmipKeyStatus[key] = 1;
            }
        } else {
            HmipKeyStatus[key] = 0;
        }
    }
}
HMI_KEY_Measure(FALSE);
}

/* END hmi */
```

ui 程序

```
/** ###################################################################
**     Filename    : ui.C
**     Project     : CyberNT3
**     Processor : 56F8366
**     Compiler    : Metrowerks DSP C Compiler
**     Date/Time : 2011/8/11, 12:36
**     Contents    :
**         User source code
**
** ###################################################################*/

/* MODULE ui */

#include "world.h"

void UiInitializeObject(UI_OBJECT *object)
{
    object->lowLimit = 0;
    object->highLimit = 0;
}

void UiRegisterEntryObject(UI_OBJECT *object, U8 index,
    UI_READ_CALLBACK *readCallback, UI_WRITE_CALLBACK *writeCallback,
```

```
    U8 lowLimit, U8 highLimit)
{
    UI_ENTRY *entry;

    if (object->lowLimit == object->highLimit) {
        object->lowLimit = index;
        object->highLimit = (U8)(index + 1);
    } else if (index < object->lowLimit) {
        for (entry = &object->entry[index + 1];
             entry != &object->entry[object->lowLimit];
             ++entry)
        {
            entry->readCallback = NULL;
            entry->writeCallback = NULL;
        }
        object->lowLimit = index;
    } else if (index >= object->highLimit) {
        for (entry = &object->entry[object->highLimit];
             entry != &object->entry[index];
             ++entry)
        {
            entry->readCallback = NULL;
            entry->writeCallback = NULL;
        }
        object->highLimit = (U8)(index + 1);
    }
    entry = &object->entry[index];
    entry->readCallback = readCallback;
    entry->writeCallback = writeCallback;
    entry->lowLimit = lowLimit;
    entry->highLimit = highLimit;
}

#define UI_KEY_UP (0)
#define UI_KEY_DOWN (1)
#define UI_KEY_CLEAR (2)
#define UI_KEY_SUBMIT (3)

static volatile U8 UipKeyBuffer;
```

```
static void UipKeyHandler(U8 key)
{
    UipKeyBuffer = key;
}

static U8 UipWaitForSingleKey(void)
{
    HMI_HANDLER *old;
    U8 key = 0xff, temp;
    U16 pulsData[2];
    U16 pulsData0[2];
    S16 delta;

    RtlQueryPulseAccumulator(pulsData0);
    UipKeyBuffer = 0xff;
    old = HmiAttachHandler(&UipKeyHandler);

    while (key == 0xff) {
        RtlQueryPulseAccumulator(pulsData);
        delta = (S16)(pulsData[1] - pulsData0[1]);
        temp = UipKeyBuffer;
        if (delta > ROTATE_LIMIT) {
            key = UI_KEY_UP;
        } else if (delta < -ROTATE_LIMIT) {
            key = UI_KEY_DOWN;
        } else if (temp == HMI_KEY_B) {
            key = UI_KEY_CLEAR;
        } else if (temp == HMI_KEY_A) {
            key = UI_KEY_SUBMIT;
        }
    }
    HmiDetachHandler(old);
    return key;
}

void UiPromptObject(UI_OBJECT *object)
{
    U8 key = object->lowLimit, value;
    UI_ENTRY *entry;
    U8 canRead, canWrite;
```

```
UiPromptObject_Loop:
    entry = &object->entry[key];
    if (key >= object->lowLimit && key < object->highLimit) {
        canRead = (entry->readCallback != NULL);
        canWrite = (entry->writeCallback != NULL);
    } else {
        canRead = 0;
        canWrite = 0;
    }

    if (canRead) {
        value = (*(entry->readCallback))(key);
        HmiShowNumberEx((U16)key * 100 + value, 0x40);
    } else {
        HmiShowNumberEx((U16)key * 100, 0x43);
    }

    switch (UipWaitForSingleKey()) {
    case UI_KEY_UP:
        if (key + 1 < object->highLimit) {
            ++key;
        }
        break;
    case UI_KEY_DOWN:
        if (key > object->lowLimit) {
            --key;
        }
        break;
    case UI_KEY_CLEAR:
        HmiShowNumberEx(0, 0x0f);
        return;
    case UI_KEY_SUBMIT:
        if (!canWrite) {
            break;
        }
        if (entry->lowLimit >= entry->highLimit) {
            value = entry->lowLimit;
            (*(entry->writeCallback))(key, value);
            break;
```

```
        }
        if (canRead) {
            if (value < entry->lowLimit) {
                value = entry->lowLimit;
            } else if (value >= entry->highLimit) {
                value = (U8)(entry->highLimit - 1);
            }
        } else {
            value = entry->lowLimit;
        }

        while (1) {
            HmiShowNumberEx((U16)key * 100 + value, 0x10);
            switch (UipWaitForSingleKey()) {
            case UI_KEY_UP:
                if (value + 1 < entry->highLimit) {
                    ++value;
                }
                break;
            case UI_KEY_DOWN:
                if (value > entry->lowLimit) {
                    --value;
                }
                break;
            case UI_KEY_CLEAR:
                goto UiPromptObject_Loop;
            case UI_KEY_SUBMIT:
                (*(entry->writeCallback))(key, value);
                goto UiPromptObject_Loop;
            }
        }
        break;
    }
    goto UiPromptObject_Loop;
}

/* END ui */
```

config 程序

```
/** ###################################################################
**         Filename    : config.C
**         Project     : CyberNT3
**         Processor : 56F8366
**         Compiler    : Metrowerks DSP C Compiler
**         Date/Time : 2011/8/11, 12:42
**         Contents    :
**               User source code
**
** ###################################################################*/

/* MODULE config */

#include "world.h"

U8 CfgData[100];

static U8 CfgpReadCallback(U8 key)
{
    return CfgData[key];
}

static void CfgpWriteCallback(U8 key, U8 data)
{
    CfgData[key] = data;
}

void CfgMountUiObject(UI_OBJECT *ui)
{
#define MOUNT_ENTRY(idx, dfl, low, high) \
    do { \
        CfgData[idx] = dfl; \
        UiRegisterEntryObject(ui, idx, &CfgpReadCallback, &CfgpWriteCallback, low, high);
\
    } while (0)

    MOUNT_ENTRY(CFG_CONTROL_LEVEL, 2, 0, 3);
    MOUNT_ENTRY(CFG_SPEED_LOW, 25, 0, 100);
```

```
    MOUNT_ENTRY(CFG_SPEED_HIGH, 50, 0, 100);
    MOUNT_ENTRY(CFG_HIGHSPEED_ROW, 38, 0, ROWS);
    MOUNT_ENTRY(CFG_SERVO_ROW_LOW, 18, 0, ROWS);
    MOUNT_ENTRY(CFG_SERVO_ROW_HIGH, 23, 0, ROWS);
    MOUNT_ENTRY(CFG_TRACK_WIDTH, 50, 0, 100);
    MOUNT_ENTRY(CFG_SERVO_P, 40, 0, 100);
    MOUNT_ENTRY(CFG_SERVO_D, 40, 0, 100);
    MOUNT_ENTRY(CFG_TRANSFORM_NEAR, 30, 0, 100);
    MOUNT_ENTRY(CFG_STARTLINE_DISTANCE, 20, 0, 100);
    MOUNT_ENTRY(CFG_MOTOR_P, 40, 0, 100);
    MOUNT_ENTRY(CFG_MOTOR_I, 40, 0, 100);
    MOUNT_ENTRY(CFG_MOTOR_D, 40, 0, 100);
}

/* END config */
```

AFE 程序

```
/** ###################################################################
**      Filename    : afe.C
**      Project     : CyberNT3
**      Processor : 56F8366
**      Compiler    : Metrowerks DSP C Compiler
**      Date/Time : 2011/8/11, 14:21
**      Contents    :
**          User source code
**
** ###################################################################*/

/* MODULE afe */

#include "world.h"

volatile AFE_FRAME AfeFrameBuffer[2];
volatile U8 AfeFrameIndex;

static U8 AfepSlaveAddress = 0x5c;
static U16 AfepRowCounter;

static inline void AfepSwitchSlave(void)
```

```
{
    AfepSlaveAddress = (U8)(AfepSlaveAddress == 0x5c ? 0x5d : 0x5c);
    AFE_I2C_SelectSlave(AfepSlaveAddress);
}

static void AfepWriteRegister(U8 reg, U8 data)
{
    U8 buffer[2];
    U16 snd;

    buffer[0] = reg;
    buffer[1] = data;

    while (1) {
        if (AFE_I2C_SendBlock(buffer, 2, &snd) != ERR_OK) {
            AfepSwitchSlave();
            continue;
        }

        return;
    }
}

void AfeInitSystem(void)
{
    // reset
    AfepWriteRegister(0xe8, 0x02);
    AfepWriteRegister(0xe9, 0x00);
    AfepWriteRegister(0xea, 0x80);
    AfepWriteRegister(0xe0, 0x01);
    AfepWriteRegister(0xe8, 0x60);
    AfepWriteRegister(0xe9, 0x00);
    AfepWriteRegister(0xea, 0xb0);
    AfepWriteRegister(0xe0, 0x01);
    AfepWriteRegister(0xe0, 0x00);
    AfepWriteRegister(0x03, 0x01);
    AfepWriteRegister(0x03, 0x00);

    // input/output format
    AfepWriteRegister(0x00, 0x00);
```

```
    AfepWriteRegister(0x32, 0x00);
    AfepWriteRegister(0x33, 0x41);
    AfepWriteRegister(0x34, 0x11);
    AfepWriteRegister(0x35, 0xee);
    AfepWriteRegister(0x36, 0xaf);

    // events
    AFE_VSYNC_Enable();
    AFE_HSYNC_Enable();
}

void AfeVsyncHandler(void)
{
    U8 index = AfeFrameIndex;
    AfeFrameBuffer[index & 1].ready = 0xff;
    ++index;
    AfeFrameBuffer[index & 1].index = index;
    AfeFrameBuffer[index & 1].ready = 0;
    AfepRowCounter = 0;
    AfeFrameIndex = index;
}

void AfeHsyncHandler(void)
{
    volatile AFE_FRAME *frame;
    volatile U8 row;
    volatile U8 *p, *q;

    if (AfepRowCounter >= 80 && AfepRowCounter < 280 && AfepRowCounter % 5 == 0) {
        frame = &AfeFrameBuffer[AfeFrameIndex & 1];
        row = (U8)((AfepRowCounter - 80) / 5);
        p = &frame->data[row][0];
        q = p + COLS;

        // ignore 6 black pixels
        *(volatile U8 *)0x10000;
        *(volatile U8 *)0x10000;
        *(volatile U8 *)0x10000;
        *(volatile U8 *)0x10000;
        *(volatile U8 *)0x10000;
```

```
            *(volatile U8 *)0x10000;

            while (p != q) {
                *p++ = *(volatile U8 *)0x10000;
                *p++ = *(volatile U8 *)0x10000;
                *p++ = *(volatile U8 *)0x10000;
                *p++ = *(volatile U8 *)0x10000;
            }

            frame->ready = row;
        }

        AFE_FRST_ClrVal();
        ++AfepRowCounter;
        AFE_FRST_SetVal();
}

/* END afe */
```

sci 程序

```
/** ###################################################################
**         Filename    : sci.C
**         Project     : CyberNT3
**         Processor : 56F8366
**         Compiler    : Metrowerks DSP C Compiler
**         Date/Time : 2011/8/11, 15:13
**         Contents    :
**             User source code
**
** ###################################################################*/

/* MODULE sci */

#include "world.h"

static volatile U8 ScipEventBuffer = 0;

void SciIrqHandler(void)
{
```

```
    U8 c;

    while (AS_RecvChar(&c) == ERR_OK) {
        ScipEventBuffer = c;
    }
}

U8 SciReadEvent(void)
{
    U8 result = ScipEventBuffer;
    ScipEventBuffer = 0;
    return result;
}

void SciSendByte(U8 data)
{
    while (AS_SendChar(data) != ERR_OK) { /* spin */ }
}

void SciSendData(U8 *data, U16 size)
{
    U16 snd;

    while (size != 0) {
        snd = 0;
        if (AS_SendBlock(data, size, &snd) == ERR_OK) {
            data += snd;
            size -= snd;
        }
    }
}

/* END sci */
```

drv 程序

```
/** ###################################################################
**      Filename    : drv.C
**      Project     : CyberNT3
**      Processor : 56F8366
```

```
**         Compiler    : Metrowerks DSP C Compiler
**         Date/Time : 2011/8/11, 21:34
**         Contents    :
**               User source code
**
** ###################################################################*/

/* MODULE drv */

#include "world.h"

#define HISTORY_MAX (20)

typedef struct _DRV_MOTOR_CONTEXT {
      S16 expect;
      S16 speed;
      U16 index;
      U16 history[HISTORY_MAX];
      mc_sPIDparams param;
} DRV_MOTOR_CONTEXT;

static volatile U8 DrvpMotorEnabled = 0;
static volatile U8 DrvpServoEnabled = 0;
static DRV_MOTOR_CONTEXT DrvpMotorContext[2];
static S16 DrvpServoValue = 0;
static U16 DrvpServoCounter = 0;

static inline void DrvpSetPwmMotor(S16 left, S16 right)
{
      U16 l = (U16)left + 32768U;
      U16 r = (U16)right + 32768U;
      DRV_LH_SetRatio16(l);
      DRV_LL_SetRatio16(l);
      DRV_RH_SetRatio16(r);
      DRV_RL_SetRatio16(r);
}

static inline void DrvpInitializePidController(mc_sPIDparams *param)
{
      param->ProportionalGain = CFG(MOTOR_P) << 3;
```

```
    param->ProportionalGainScale = 0;
    param->IntegralGain = CFG(MOTOR_I);
    param->IntegralGainScale = 4;
    param->DerivativeGain = CFG(MOTOR_D) << 2;
    param->DerivativeGainScale = 0;
    param->PositivePIDLimit = 32767;
    param->NegativePIDLimit = -32768;
    param->IntegralPortionK_1 = 0;
    param->InputErrorK_1 = 0;
}

void DrvEnableMotorControl(void)
{
    DRV_MOTOR_CONTEXT *mc;
    U16 data[2];
    U16 i, j;

    RtlQueryPulseAccumulator(data);

    for (i = 0; i < 2; ++i) {
        mc = &DrvpMotorContext[i];
        mc->expect = 0;
        mc->speed = 0;
        mc->index = 0;
        for (j = 0; j < HISTORY_MAX; ++j) {
            mc->history[j] = data[i];
        }
        DrvpInitializePidController(&mc->param);
    }

    DrvpSetPwmMotor(0, 0);
    DRV_BYTE_PutVal(0xff);
    DrvpMotorEnabled = 1;
}

void DrvDisableMotorControl(void)
{
    DRV_BYTE_PutVal(0x0);
    DrvpMotorEnabled = 0;
}
```

```
void DrvSetSpeedMotor(S16 left, S16 right)
{
    DrvpMotorContext[0].expect = left;
    DrvpMotorContext[1].expect = right;
}

S16 DrvQueryCurrentSpeed(void)
{
    return (DrvpMotorContext[0].speed + DrvpMotorContext[1].speed) / 2;
}

void DrvEnableServoControl(void)
{
    DrvpServoValue = 0;
    DrvpServoCounter = 0;
    DrvpServoEnabled = 1;
}

void DrvDisableServoControl(void)
{
    DrvpServoEnabled = 0;
}

void DrvSetValueServo(S16 value)
{
    DrvpServoValue = RtlLimitS16(value, -SERVO_RANGE, SERVO_RANGE);
}

static inline void DrvpUpdateCurrentSpeed(U16 index, U16 current)
{
    DRV_MOTOR_CONTEXT *mc = &DrvpMotorContext[index];
    mc->speed = (S16)(current - mc->history[mc->index]);
    mc->history[mc->index] = current;
    if (++mc->index >= HISTORY_MAX) {
        mc->index = 0;
    }
}

static inline void DrvpMotorControl(void)
```

```
{
    DRV_MOTOR_CONTEXT *mc;
    U16 current[2];
    S16 pwm[2];

    RtlQueryPulseAccumulator(current);

#define ACTION(idx) \
    do { \
        mc = &DrvpMotorContext[(idx)]; \
        mc->speed = (S16)(current[(idx)] - mc->history[mc->index]); \
        mc->history[mc->index] = current[(idx)]; \
        if (++mc->index >= HISTORY_MAX) { \
            mc->index = 0; \
        } \
        pwm[(idx)] = PID_controllerPIDtype1(mc->expect, mc->speed, &mc->param); \
    } while (0)

    ACTION(0);
    ACTION(1);

    DrvpSetPwmMotor(pwm[0], pwm[1]);
}

static inline void DrvpServoControl(void)
{
    if (++DrvpServoCounter >= SERVO_FREQUENCY) {
        DRV_PS_SetClockTicks16((U16)(SERVO_MIDDLE + DrvpServoValue));
        DRV_PS_Enable();
        DrvpServoCounter = 0;
    }
}

void DrvIrqHandler(void)
{
    if (DrvpServoEnabled) {
        DrvpServoControl();
    }

    if (DrvpMotorEnabled) {
```

```
            DrvpMotorControl();
        }
    }

    /* END drv */
```

nt 程序

```
/** ###################################################################
**         Filename    : nt.C
**         Project     : CyberNT3
**         Processor : 56F8366
**         Compiler    : Metrowerks DSP C Compiler
**         Date/Time : 2011/8/11, 15:22
**         Contents    :
**               User source code
**
** ###################################################################*/

/* MODULE nt */

#include "world.h"

#define THRESHOLD_MIN (32)
#define INITIAL_LINE_WIDTH (11)
#define SCORE_MAX (COLS / 3)
#define LINE_WIDTH_MAX (20)
#define LOSE_LIMIT (18)

#define MAX_STARTLINE_ROW (20)
#define STARTLINE_WIDTH_MIN (23)
#define STARTLINE_WIDTH_MAX (48)
#define STARTLINE_STEP_MIN (8)
#define STARTLINE_STEP_MAX (18)

// global variables
static volatile U8 NtpIsRunning;
static volatile U8 NtpIsStopping;
static S32 NtpCurrentDistance;
#define CURRENT_DISTANCE (S16)(NtpCurrentDistance / 3400)
```

```
static S16 NtpStopDistance;
static U16 NtpPreviousAccumulator[2];
static U8 NtpFrameIndex;
static volatile AFE_FRAME *NtpFrame;
static U8 NtpSendFrame;
static U8 NtpRow;
static S16 NtpPredictPosition;
static U8 NtpServoHistoryIndex;
static S16 NtpServoHistory[3];

// memorial variables
static S16 NtpLineWidth;
static S16 NtpLineFirstDerivative;
static U8 NtpLastValidRow;
static S16 NtpLosingCounter;

// result variables
static U8 NtpWidth[ROWS];
static U8 NtpPosition[ROWS];

static U8 NtpLeftStartLineCount, NtpRightStartLineCount;
static U8 NtpLeftStartLine, NtpRightStartLine;

static U8 NtpMaybeLeftStartLineCount, NtpMaybeRightStartLineCount;
static U8 NtpMaybeLeftStartLine, NtpMaybeRightStartLine;

// controller context
static mc_sPIDparams NtpServoPidCtx;

// tables
S16 NtpTransformTable[ROWS];

static void NtpKeyHandler(U8 key)
{
    if (key == HMI_KEY_A) {
        NtpIsRunning = 0;
    }
}

static inline U8 NtpCheckforStartLine(void)
```

```
{
#define MAX_STARTLINE_LENGTH (5)
#define MAX_STARTLINE_GAP (3)
#define MAX_CONFIRM_STARTLINE_ROW (18)
    if(NtpLeftStartLineCount && NtpRightStartLineCount
        && NtpLeftStartLineCount <= MAX_STARTLINE_LENGTH &&
NtpRightStartLineCount <= MAX_STARTLINE_LENGTH) {
        if (RtlAbsS16(NtpLeftStartLine - NtpRightStartLine) <= MAX_STARTLINE_GAP) {
            return 1;
        }
    }
    if (NtpMaybeLeftStartLineCount && NtpRightStartLineCount
        && NtpMaybeLeftStartLineCount <= MAX_STARTLINE_LENGTH &&
NtpRightStartLineCount <= MAX_STARTLINE_LENGTH
        && NtpMaybeLeftStartLine < MAX_CONFIRM_STARTLINE_ROW) {
        if (RtlAbsS16(NtpMaybeLeftStartLine - NtpRightStartLine) <=
MAX_STARTLINE_GAP) {
            return 1;
        }
    }
    if (NtpLeftStartLineCount && NtpMaybeRightStartLineCount
        && NtpLeftStartLineCount <= MAX_STARTLINE_LENGTH &&
NtpMaybeRightStartLineCount <= MAX_STARTLINE_LENGTH
        && NtpMaybeRightStartLine < MAX_CONFIRM_STARTLINE_ROW) {
        if (RtlAbsS16(NtpLeftStartLine - NtpMaybeRightStartLine) <=
MAX_STARTLINE_GAP) {
            return 1;
        }
    }
    return 0;
}


static inline void NtpInitRunning(void)
{
    S16 row;

    NtpIsRunning = 1;
    NtpIsStopping = 0;
    NtpCurrentDistance = 0;
```

```
    NtpStopDistance = 32767;
    RtlQueryPulseAccumulator(NtpPreviousAccumulator);
    NtpFrameIndex = AfeFrameIndex;
    NtpPredictPosition = COLS / 2 - 1;
    NtpServoHistoryIndex = 0;
    NtpServoHistory[0] = 0;
    NtpServoHistory[1] = 0;
    NtpServoHistory[2] = 0;

    for (row = 0; row < ROWS; ++row) {
        NtpTransformTable[row] = RtlScaleS16(RtlSquareS16(ROWS - row),
            0, ROWS * ROWS, 1000, CFG(TRANSFORM_NEAR) * 100);
    }

    NtpServoPidCtx.ProportionalGain = CFG(SERVO_P);
    NtpServoPidCtx.ProportionalGainScale = 1;
    NtpServoPidCtx.IntegralGain = 0;
    NtpServoPidCtx.IntegralGainScale = 0;
    NtpServoPidCtx.DerivativeGain = CFG(SERVO_D);
    NtpServoPidCtx.DerivativeGainScale = 0;
    NtpServoPidCtx.PositivePIDLimit = SERVO_RANGE;
    NtpServoPidCtx.NegativePIDLimit = -SERVO_RANGE;
    NtpServoPidCtx.IntegralPortionK_1 = 0;
    NtpServoPidCtx.InputErrorK_1 = 0;
}

static inline void NtpProcessRow(void)
{
    static S16 edge[COLS];
    U32 sum;
    U8 *data;
    U8 col;
    S16 t;
    S16 threshold;
    U8 rising[5];
    U8 risingIndex;
    U8 i;

#define STARTLINE_MAX (20)
#define MAYBE_STARTLINE_MAX (5)
```

```
    U8 startLine[STARTLINE_MAX + 1][2];
    U8 startLineIndex;
    U8 maybeLeftStartLine[MAYBE_STARTLINE_MAX + 1],
maybeRightStartLine[MAYBE_STARTLINE_MAX + 1];
    U8 maybeLeftStartLineIndex, maybeRightStartLineIndex;
    static U8 noStartLine = 0;

    S16 width, pos, firstDeriv;
    S32 score;
    S32 bestScore = SCORE_MAX;
    S16 bestWidth, bestPos, bestFirstDeriv;
    S16 row;

    if (NtpLosingCounter > LOSE_LIMIT) {
        goto NtpProcessRow_SkipScanning;
    }

    sum = 0;
    data = (U8 *)&NtpFrame->data[NtpRow][0];
    while (NtpFrame->ready <= NtpRow) { /* spin wait */ }
    for (col = 1; col < COLS - 2; ++col) {
        t = (S16)data[col - 1] - (S16)data[col + 2];
        edge[col] = t;
        if (t >= 0) {
            sum += t;
        } else {
            sum -= t;
        }
    }
    threshold = (S16)(sum / (COLS - 3));
    if (threshold < THRESHOLD_MIN) {
        threshold = THRESHOLD_MIN;
    }

    rising[0] = 0;
    rising[1] = 0;
    rising[2] = 0;
    rising[3] = 0;
    rising[4] = 0; // end mark
    risingIndex = 0;
```

```
for (i = 0; i < STARTLINE_MAX + 1; ++i) {
    startLine[i][0] = 0;
    startLine[i][1] = 0;
}
startLineIndex = 0;

for (i = 0; i < MAYBE_STARTLINE_MAX + 1; ++i) {
    maybeLeftStartLine[i] = 0;
    maybeRightStartLine[i] = 0;
}
maybeLeftStartLineIndex = 0;
maybeRightStartLineIndex = 0;

edge[0] = 0;
edge[COLS - 2] = 0;

for (col = 1; col < COLS - 2; ++col) {
    if (edge[col] >= threshold) {
        if (edge[col] >= edge[col - 1] && edge[col] > edge[col + 1]) {
            rising[risingIndex] = col;
            if (++risingIndex == 4) {
                risingIndex = 0;
            }

            // maybe right startline
            maybeRightStartLine[maybeRightStartLineIndex] = col;
            if (++maybeRightStartLineIndex == MAYBE_STARTLINE_MAX) {
                maybeRightStartLineIndex = 0;
            }
        }
    } else if (edge[col] <= -threshold) {
        if (edge[col] < edge[col - 1] && edge[col] <= edge[col + 1]) {
            // maybe left startline
            if (maybeLeftStartLineIndex < MAYBE_STARTLINE_MAX) {
                maybeLeftStartLine[maybeLeftStartLineIndex] = col;
                ++maybeLeftStartLineIndex;
            }
            for (i = 0; rising[i] != 0; ++i) {
                width = (S16)col - (S16)rising[i];
```

```
                    // startline
                    if (STARTLINE_WIDTH_MIN <= width && width <=
STARTLINE_WIDTH_MAX){
                        startLine[startLineIndex][0] = rising[i];
                        startLine[startLineIndex][1] = col;
                        if (++startLineIndex == STARTLINE_MAX) {
                            startLineIndex = 0;
                        }
                    }

                    if (width > LINE_WIDTH_MAX) {
                        continue;
                    }
                    pos = ((S16)rising[i] + (S16)col) / 2;

                    // scoring
                    if (NtpLastValidRow == 0xff) {
                        firstDeriv = 0;
                        score = RtlSquareS32(width - NtpLineWidth) * 2
                            + RtlAbsS16(pos - NtpPredictPosition);
                    } else {
                        firstDeriv = (pos - NtpPosition[NtpLastValidRow]) / (NtpRow -
NtpLastValidRow);
                        score = RtlSquareS32(width - NtpLineWidth) * 2
                            + RtlAbsS16(firstDeriv)
                            + RtlSquareS32(firstDeriv - NtpLineFirstDerivative);
                    }

                    if (score < bestScore) {
                        bestScore = score;
                        bestWidth = width;
                        bestPos = pos;
                        bestFirstDeriv = firstDeriv;
                    }
                }
            }
        }
    }

NtpProcessRow_SkipScanning:
```

```
    if (bestScore < SCORE_MAX) {
        // current row found
        NtpLineWidth = (NtpLineWidth + bestWidth) / 2;
        NtpLineFirstDerivative = (NtpLineFirstDerivative + bestFirstDeriv) / 2;

        // interpolation
        if (NtpLastValidRow != 0xff) {
            for (row = NtpLastValidRow + 1; row < NtpRow; ++row) {
                NtpWidth[row] = (U8)RtlScaleS16(row, NtpLastValidRow, NtpRow,
                    NtpWidth[NtpLastValidRow], bestWidth);
                NtpPosition[row] = (U8)RtlScaleS16(row, NtpLastValidRow, NtpRow,
                    NtpPosition[NtpLastValidRow], bestPos);
            }
        }

        NtpWidth[NtpRow] = (U8)bestWidth;
        NtpPosition[NtpRow] = (U8)bestPos;
        NtpLastValidRow = NtpRow;

        // bestPos: [1, COLS-3 = 169]

        NtpLosingCounter = RtlScaleS16(NtpRow, 0, ROWS - 1, 0, LOSE_LIMIT);
        if (NtpLosingCounter < LOSE_LIMIT) {
            NtpLosingCounter = RtlScaleS16(
                RtlAbsS16((S16)bestPos - (COLS/2 - 1)),
                0, COLS/2 - 2, NtpLosingCounter, LOSE_LIMIT);
        }
    } else {
        // current row lost
        NtpWidth[NtpRow] = 0;
        NtpPosition[NtpRow] = 0;
        ++NtpLosingCounter;
    }


#define MAX_NO_STARTLINE (4)
    // startline
    if (NtpPosition[NtpRow] == 0) {
        noStartLine = MAX_NO_STARTLINE;
    } else {
```

```
            if (noStartLine > 0) {
                --noStartLine;
            }
        }
        if (NtpRow < MAX_STARTLINE_ROW && NtpPosition[NtpRow] != 0 && noStartLine
== 0) {
            // left
            for (i = 0; startLine[i][0] != 0; ++i) {
                if ((S16)NtpPosition[NtpRow] - (S16)NtpWidth[NtpRow] / 2 -
(S16)STARTLINE_STEP_MAX <= (S16)startLine[i][1]
                && (S16)startLine[i][1] <= (S16)NtpPosition[NtpRow] - (S16)NtpWidth[NtpRow]
/ 2 - (S16)STARTLINE_STEP_MIN) {
                    ++NtpLeftStartLineCount;
                    if (NtpLeftStartLineCount == 1) {
                        NtpLeftStartLine = NtpRow;
                    }
                    goto HasLeftStartLine;
                }
            }
            //maybe left
            for (i = 0;maybeLeftStartLine[i] != 0; ++i) {
                if ((S16)NtpPosition[NtpRow] - (S16)NtpWidth[NtpRow] / 2 -
(S16)STARTLINE_STEP_MAX <= (S16)maybeLeftStartLine[i]
                && (S16)maybeLeftStartLine[i] <= (S16)NtpPosition[NtpRow] -
(S16)NtpWidth[NtpRow] / 2 - (S16)STARTLINE_STEP_MIN
                && (S16)maybeLeftStartLine[i] - (S16)STARTLINE_WIDTH_MAX <= 1) {
                    ++NtpMaybeLeftStartLineCount;
                    if (NtpMaybeLeftStartLineCount == 1) {
                        NtpMaybeLeftStartLine = NtpRow;
                    }
                    break;
                }
            }
HasLeftStartLine:
            // right
            for (i = 0; startLine[i][0] != 0; ++i) {
                if ((S16)NtpPosition[NtpRow] + (S16)NtpWidth[NtpRow] / 2 +
(S16)STARTLINE_STEP_MIN <= (S16)startLine[i][0]
                && (S16)startLine[i][0] <= (S16)NtpPosition[NtpRow] + (S16)NtpWidth[NtpRow]
/ 2 + (S16)STARTLINE_STEP_MAX) {
```

```
                    ++NtpRightStartLineCount;
                    if (NtpRightStartLineCount == 1) {
                        NtpRightStartLine = NtpRow;
                    }
                    goto HasRightStartLine;
                }
            }
            //maybe right
            for (i = 0; maybeRightStartLine[i] != 0; ++i) {
                if ((S16)NtpPosition[NtpRow] + (S16)NtpWidth[NtpRow] / 2 +
(S16)STARTLINE_STEP_MIN <= (S16)maybeRightStartLine[i]
                && (S16)maybeRightStartLine[i] <= (S16)NtpPosition[NtpRow] +
(S16)NtpWidth[NtpRow] / 2 + (S16)STARTLINE_STEP_MAX
                && (S16)maybeRightStartLine[i] + (S16)STARTLINE_WIDTH_MAX >= COLS
- 2) {
                    ++NtpMaybeRightStartLineCount;
                    if (NtpMaybeRightStartLineCount == 1) {
                        NtpMaybeRightStartLine = NtpRow;
                    }
                    break;
                }
            }
HasRightStartLine:
            i = i;
        }

        switch (NtpSendFrame) {
        case 'g':
            SciSendData((U8 *)&NtpFrame->data[NtpRow], COLS);
            break;
        case 'b':
            SciSendByte(NtpPosition[NtpRow]);
            SciSendByte((U8)bestScore);
            break;
        case 'r':
            SciSendData((U8 *)&NtpFrame->data[NtpRow], COLS);
            SciSendByte(NtpPosition[NtpRow]);
            break;
        }
}
```

```
static inline S16 NtpGetScaledPosition(S16 row)
{
    if (NtpPosition[row] == 0) {
        return 0;
    }
    return (S16)((S32)(NtpPosition[row] - (COLS/2 - 1)) * NtpTransformTable[row] / 1000) +
(COLS/2 - 1);
}

static void NtpProcessFrame(void)
{
    S16 servo = 0;
    U8 frameIndex;
    U8 predictCount;
    S16 predictSum;
    S16 leftSpeed, rightSpeed;
    S16 position;
    S16 row;
    S16 servoRow;
    S32 fixSum, fixCount;

    NtpLineWidth = INITIAL_LINE_WIDTH;
    NtpLineFirstDerivative = 0;
    NtpLastValidRow = 0xff;
    NtpLosingCounter = 0;

    NtpLeftStartLineCount = 0;
    NtpRightStartLineCount = 0;

    NtpMaybeLeftStartLineCount = 0;
    NtpMaybeRightStartLineCount = 0;

    while ((frameIndex = AfeFrameIndex) == NtpFrameIndex) { /* spin wait */ }
    NtpFrameIndex = frameIndex;
    NtpFrame = &AfeFrameBuffer[frameIndex & 1];
    NtpSendFrame = SciReadEvent();

    predictSum = 0;
    predictCount = 0;
```

```
    for (NtpRow = 0; NtpRow < ROWS; ++NtpRow) {
        NtpProcessRow();
#define PREDICT_ROW (10)
        if (NtpRow < PREDICT_ROW && NtpPosition[NtpRow] != 0) {
            predictSum += NtpPosition[NtpRow];
            ++predictCount;
        }
    }

    if (predictCount != 0) {
        NtpPredictPosition = (NtpPredictPosition + predictSum / predictCount) / 2;
    }

    servoRow = RtlScaleS16(RtlLimitS16(DrvQueryCurrentSpeed(), 0, 800),
        0, 800, CFG(SERVO_ROW_LOW), CFG(SERVO_ROW_HIGH));
    position = NtpGetScaledPosition(servoRow);
    if (position != 0) {
        fixSum = fixCount = 0;
        for (row = servoRow + 1; row < ROWS; ++row) {
            if (NtpPosition[row] != 0) {
                fixSum += (S32)NtpGetScaledPosition(row) * row * row;
                fixCount += (S32)row * row;
            }
        }
        if (fixCount != 0) {
            position = RtlLimitS16((S16)(fixSum / fixCount),
                position - CFG(TRACK_WIDTH), position + CFG(TRACK_WIDTH));
        }
        position -= (COLS / 2 - 1);
    } else {
        position = (NtpServoHistory[0] + NtpServoHistory[1] + NtpServoHistory[2]);
    }
    if (position < -COLS/2) {
        position = -COLS/2;
    } else if (position > COLS/2) {
        position = COLS/2;
    }
    NtpServoHistory[NtpServoHistoryIndex] = position;
    if (++NtpServoHistoryIndex == 3) {
        NtpServoHistoryIndex = 0;
```

```
    }

    if (NtpLeftStartLineCount || NtpRightStartLineCount || NtpMaybeLeftStartLineCount ||
NtpMaybeRightStartLineCount) {
        HmiEnableBuzz();
    } else {
        HmiDisableBuzz();
    }

    if (!NtpIsStopping) {
        if (CURRENT_DISTANCE > CFG(STARTLINE_DISTANCE) * 10 &&
NtpCheckforStartLine()) {
            NtpIsStopping = 1;
            NtpStopDistance = CURRENT_DISTANCE + 8;
        }
    }

    if (CURRENT_DISTANCE >= NtpStopDistance) {
        leftSpeed = rightSpeed = 0;
    }else if (NtpPosition[CFG(HIGHSPEED_ROW)] != 0) {
        leftSpeed = rightSpeed = CFG(SPEED_HIGH) * 20;
    } else {
        leftSpeed = rightSpeed = CFG(SPEED_LOW) * 20;
    }

    servo = PID_controllerPIDtype1(0, position, &NtpServoPidCtx);
    if (servo > 0) {
        rightSpeed = RtlScaleS16(servo, 0, SERVO_RANGE, rightSpeed,
            rightSpeed / 2);
    } else if (servo < 0) {
        leftSpeed = RtlScaleS16(-servo, 0, SERVO_RANGE, leftSpeed,
            leftSpeed / 2);
    }

    DrvSetValueServo(servo);
    DrvSetSpeedMotor(leftSpeed, rightSpeed);
}

static void NtpMeasureDistance(void)
{
```

```
    U16 data[2];
    volatile S16 delta;

    RtlQueryPulseAccumulator(data);
    delta = (S16)(data[0] - NtpPreviousAccumulator[0]);
    NtpCurrentDistance += delta;
    delta = (S16)(data[1] - NtpPreviousAccumulator[1]);
    NtpCurrentDistance += delta;

    NtpPreviousAccumulator[0] = data[0];
    NtpPreviousAccumulator[1] = data[1];
}

void NtSystemEntry(void)
{
    HMI_HANDLER *oldHandler;

    NtpInitRunning();
    if (CFG(CONTROL_LEVEL) >= 1) {
        DrvEnableServoControl();
        if (CFG(CONTROL_LEVEL) >= 2) {
            DrvEnableMotorControl();
        }
    }
    oldHandler = HmiAttachHandler(&NtpKeyHandler);
    while (NtpIsRunning) {
        NtpProcessFrame();
        NtpMeasureDistance();
        HmiShowNumberEx((U16)CURRENT_DISTANCE, 0x20);
    }
    HmiDetachHandler(oldHandler);
    DrvDisableMotorControl();
    DrvDisableServoControl();
}

/* END nt */
```