第七届“飞思卡尔”杯全国大学生智能汽车竞赛

技术报告

杭州电子科技大学

HANGZHOU DIANZI UNIVERSITY

学　　校：　杭州电子科技大学

队伍名称：杭电摄像头一队

参赛队员：　肖乃瑶

沈骏

张学涌

带队老师：　陈龙　　高明煜

关于技术报告和研究论文使用授权的说明

本人完全了解第七届“飞思卡尔”杯全国大学生智能汽车竞赛关保留、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：肖乃瑶 沈骏 张学涌

带队教师签名：陈龙 高明煜

日　　期：2012/08/13

摘要

本文以第七届飞思卡尔智能车竞赛为背景，介绍了杭州电子科技大学的队员们为准备本次比赛所取得的成果。比赛采用大赛组委会提供的 1:10 仿真车模， 以 freescale 半导体公司的 MC9S12X128 单片机为核心，通过 CCD 摄像头检测车模的运行位置和方向，通过光电编码器获取车模运行速度。

在摄像头识别路线方案当中，黑线的处理显得尤为重要，我们沿用去年黑线处理的思想，采用动态阈值和区域搜索进行黑线的提取，这种算法经过大量的实验验证，最终得到我们的信赖。我们将更多的时间用在了控制方面，比如怎样使舵机打角和电子差速配合更加完美，为此我们进行了大量的实验，最终决定采用动态打角行和舵机 PD 控制，而速度使用了 PID 控制，效果理想。

为提高赛车的稳定性和速度，我们今年在软硬件方面使用了多套方案进行比较。沿用了去年的车载 FLASH 技术实时存储赛道信息。为更好的分析这些数据，我们用 MATLAB 软件编写了上位机程序来进行车模调试。在进行大量的数据分析以及调试之后，表明该系统设计方案可行。

关键字：飞思卡尔智能车,MC9S12X128,CCD 摄像头,动态阈值,PID 控制,FLASH 存储器,MATLAB

Abstract

In this report we will introduceour intelligent car from different aspects. The model car is appointed by the organizing committee, the micro-controller is MC9S12XS128 which come from Freescale Semiconductor, the image sensor is CCD camera and the speed sensor is photoelectric encoder which come from NEMICON.

The most important part of our car is software. How to recognize black lineaccurately is the main challenge. In order to solve this problem, we use some thoughts like dynamic threshold algorithm, region searchalgorithm and so on. We spent a lot of time to research how to control a steering servo and two DC motors and how to let them match each other. After a lot of tests, we decide to use PD algorithm to control the steering servo and use PID algorithm to control two DC motors. And the result is satisfactory.

For increasing speed and stability of our car, we use Flash to store track information and use upper computer which is compiled byMATLAB to do analysis and debugging.

Key Words:intelligentcar, MC9S12XS128, Freescale, CCD camera, PID, MATLAB

目录

第一章 引言

我们的智能车系统以飞思卡尔 16 位微控制 MC9S12XS128 作为核心控制单元，软件平台为 CodeWarrior IDE，车模采用大赛组委会统一提供的 A 型车模，赛道传感器选用 CCD 模拟摄像头，速度传感器选用光电编码器，电机驱动采用经优化的全桥电路，人机操作界面为“LCD+按键”模块，转向舵机采用 PD 控制，电机采用 PID 控制。

制作智能车的过程充满挑战和阻碍，从一开始的车模测量改造，零件设计加工到电路板的设计加工焊接，元器件的挑选购买，整车的装配，到漫长的软件调试阶段，无一不在考验和锻炼我们的动手实践能力和坚持不懈的毅力。经过几个月的努力，通过大量的实验和调试，我们解决了一系列的难题，例如小车的重启复位问题等，并成功制作出稳定可靠的智能车。

在该技术报告中，我们详细介绍了智能车系统的软硬件结构及设计开发过程。其中，第一章为引言部分；第二章主要介绍了小车的总体方案；第三章介绍了小车的机械改造及设计；第四章详细介绍了小车的硬件电路设计；第五章介绍了小车的软件设计和相关算法；第六章则介绍了我们在制作和调试智能车的过程中遇到的问题和实际解决方法。

智能车竞赛参赛学校众多，竞争异常激烈，要在这样的大赛中脱颖而出不仅仅需要可靠优秀的硬软件支持，还需要整个团队齐心协力，众志成城，另外不可或缺的是学校和老师的大力支持。经过这辛苦的几个月，我们不仅将小车的速度成功稳定在 3m/s 以上（中等难度赛道），更是学到了许许多多的专业知识、结识了一大帮志同道合的朋友。

第二章 智能车整体设计

2.1 系统结构框图

智能车系统主要包括以下模块：MC9S12XS128 最小系统、调试模块、转向舵机模块、电机及其驱动模块、速度反馈模块、摄像头模块、视频信号处理模块和电源管理模块（为所有其它模块供电）。智能车整体结构框图，如图 2.1 所示。

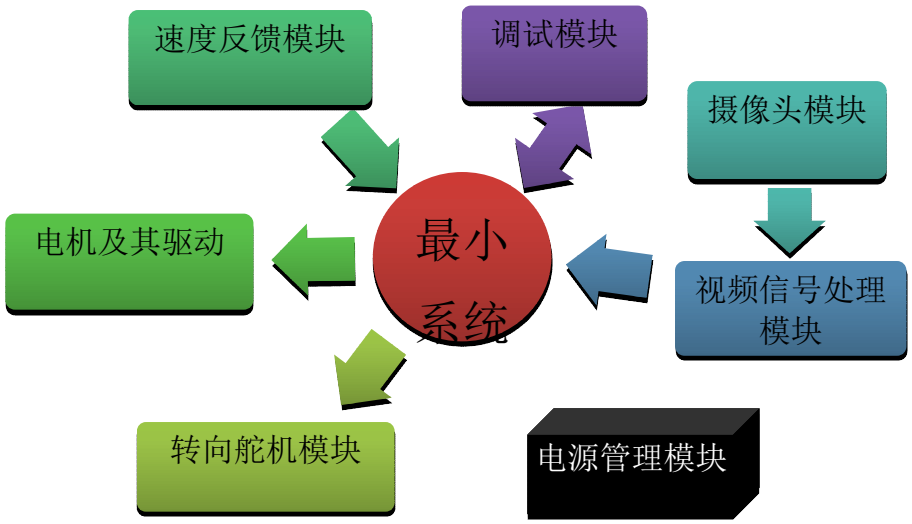


图 2.1 智能车整体结构框图

2.2 车模整体布局

根据以往多届比赛的经验，良好稳定的图像信息是智能车稳定的关键，优秀的机械架构是智能车驰骋赛道的有力保障，优良的辅助调试系统是高效率调试的前提，而其它部分则相对次要。本次比赛摄像头组采用了与第四届相同的车模，但前、后轮都采用了滚珠轴承。在制作小车的过程中，我们坚持一下原则：①重视车模机械改造及加工；②简化硬件电路；③尽量降低车模重心。

制作完成后的小车整体布局如图 2.2、图 2.3 所示。

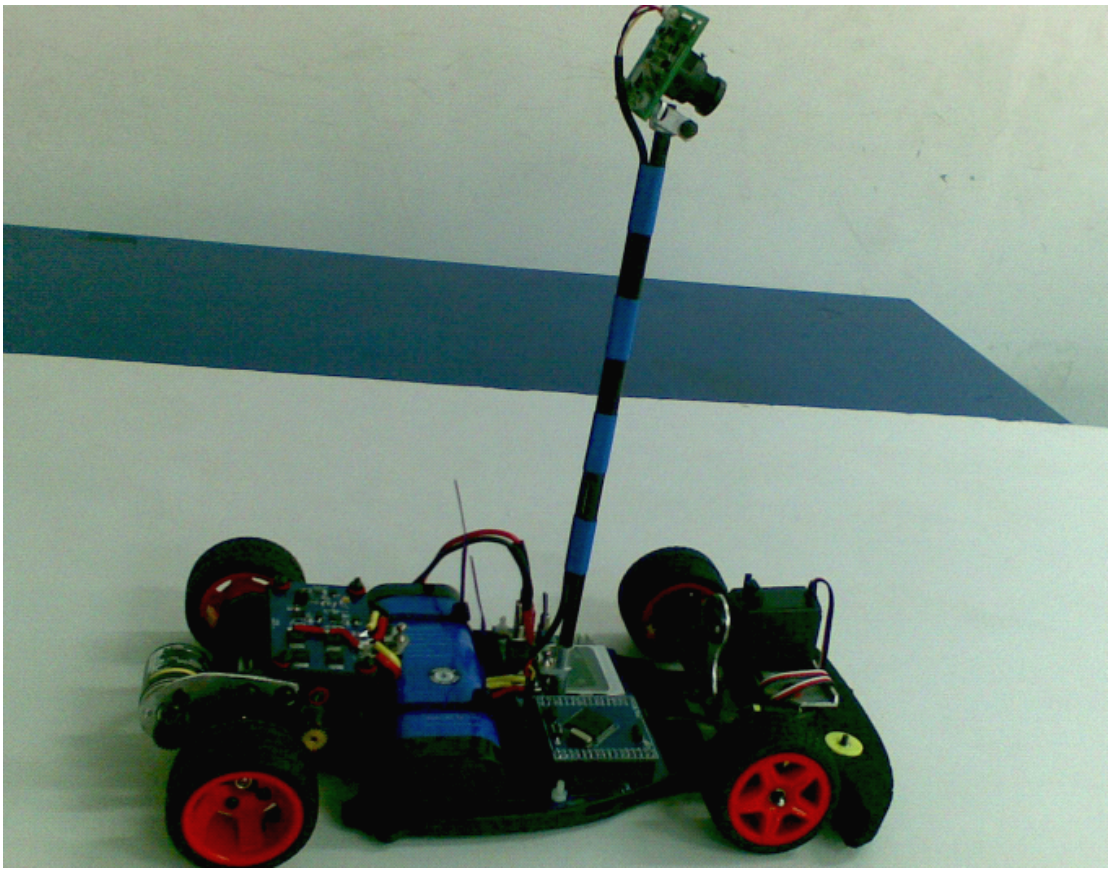


图 2.2 侧视图

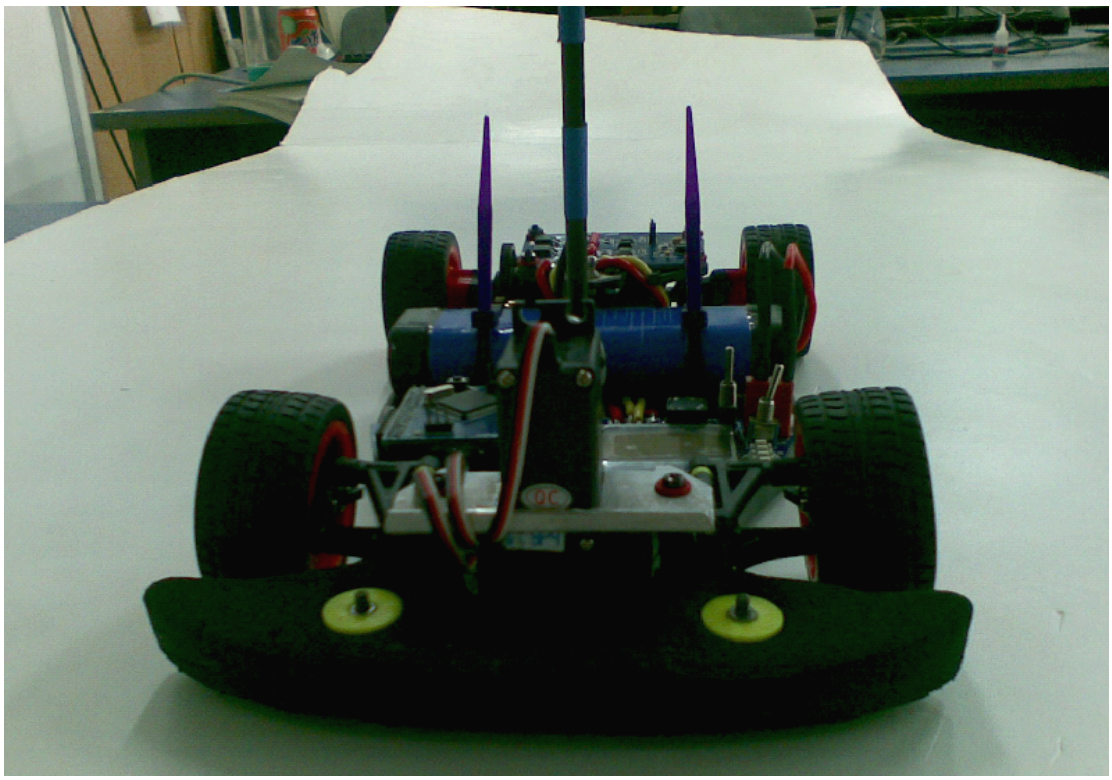


图 2.3 前视图

2.3 车模结构参数

本次大赛指定的摄像头组车模为 A 车模，车模型号 G768，生产商为东莞博思电子数码科技有限公司。该车模较与和第四届的车模相同，但从轴承改为了滚珠轴承，电机型号 RS380-ST/3545,舵机采用 FUTABA3010。

转向舵机 FUTABA3010，在 6V 工作电压下速度为 0.16 sec/60°，扭矩为 6.5 kg/cm，外形参数为 40 x 20 x 38mm，重量为 41g，内部为尼龙齿轮。

车模机械加工主要分为舵机支架、编码器支架和摄像头支架三个部分。将舵机竖直安装并架高；将光电编码器安装于后轮前方，直接与电机齿轮咬合；将摄像头竖直安装于车模中心位置。

完成之后的车模参数如表 2.1 所示。

表 2.1 车模机械参数

车模长度	300mm
车模宽度	170mm
车模高度	200mm
车模重量	995.7g

车模轴距	230mm
前轮轮距	140mm
后轮轮距	140mm

第三章 机械结构的安装与调整

3.1 舵机的安装

舵机转向是整个车模系统中延迟最大的一个环节，为了减小此时间常数，通过改变舵机的安装位置，加长力臂可以提高舵机的响应速度。鉴于往届经验以及本届车模舵机性能，我们进行多套方案的试验，诸如将舵机竖直、水平以及其它不同方向的摆放方法。考虑到舵机响应时间、稳定性以及虚位的诸多因素，我们最终选择竖直安装舵机，延长舵机臂杆至 33mm。具体安装见图 3.1。

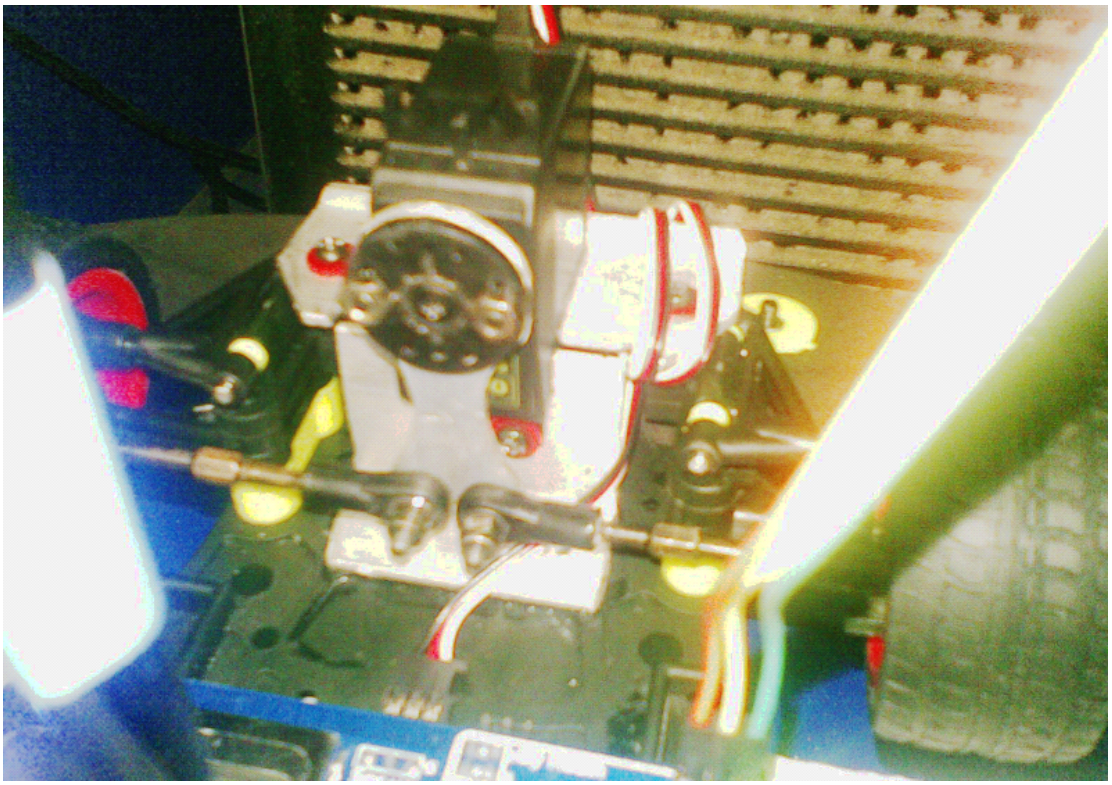


图 3.1 舵机的架法

3.2 底盘设计

今年的车模较去年的简洁，底盘结构与第四届车模相似。对于底盘，我们没有做过多的改造，只是适当调整了前后底盘的高度使车模整体的重心下降到合适的位置，即保证小车可以顺利过坡，且与赛道不会接触摩擦。用车模自带零配件即可调整底盘前后的高度。合适的重心在小车过弯性能和小车速度这两个方面上起了很大的影响。

3.3 摄像头的安装

摄像头是整辆车的眼睛，摄像头的安装是最重要的。摄像头的安装要求使得摄像头位于整个车模的中心位置，而且高度要适合于图像的采集和处理。通过多组对比试验，我们最后决定使用单杆结构来固定摄像头，单杆使得整个支架十分简约，但也会带来摄像头易抖动的问题。这时，单杆与底盘之间的连接可靠性就显得尤为重要。图 3.2 中是我们自行设计及加工的零件，它使得单杆牢牢地与底盘合为一体。

图 3.2 单杆-底盘连接件

另外，摄像头的高度也会影响其抖动，架的越高则越容易抖动。为保证图像的稳定可靠，也为了在一定程度上降低车模重心，我们架低了摄像头使之与地面的距离在 200mm 左右。摄像头与单杆之间的连接也十分重要，稳定牢靠的连接必不可少，连接件的重量也要控制在一定范围内，为此我们设计和加工了一种结构简单，质量轻巧的连接件，如图 3.3 所示。

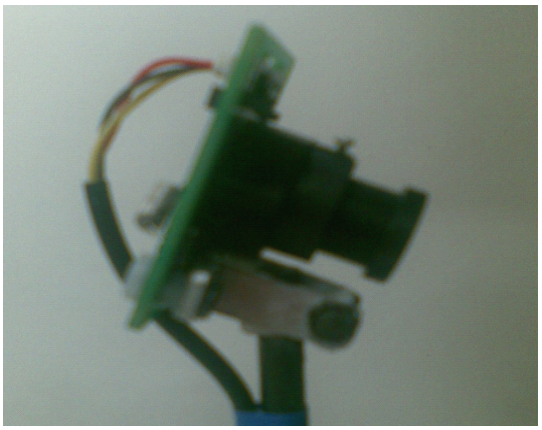


图 3.3 单杆-摄像头连接件

3.4 编码器的安装

速度传感器一般可以选择对射式光栅或光电编码器。对射式光栅的重量轻，阻力小精度也高，然而光栅暴露在外界容易受到外界光线或粉尘等的影响，导致计数不准确；而光电编码器就不存在此类问题。所以最后我们选择了微型的内密控光电编码器 OME-N 系列，该编码器线数为 157 线，符合我们的要求。编码器如图 3.4 所示。

图 3.4 内密控微型光电编码器

在安装编码器的时候要保证有合适的齿轮咬合。咬合完美的原则是：两个

传动齿轮轴保持平行，齿轮间的配合间隙要合适，过松容易打坏齿轮，过紧又会增加传动阻力；传动部分要轻松、顺畅，容易转动。判断齿轮传动是否调整好的一个依据是，听一下电机带动后轮空转时的声音。声音刺耳响亮，说明齿轮间的配合间隙过大，传动中有撞齿现象；声音闷而且有迟滞，则说明齿轮间的配合间隙过小，咬合过紧，或者两齿轮轴不平行，电机负载加大。调整好的齿轮传动噪音小，并且不会有碰撞类的杂音。图 3.5、图 3.6 为编码器的机械参数，根据这些参数我们自行设计和加工了轻巧的零件来使编码器与电机合适咬合在一起，如图 3.7 所示。

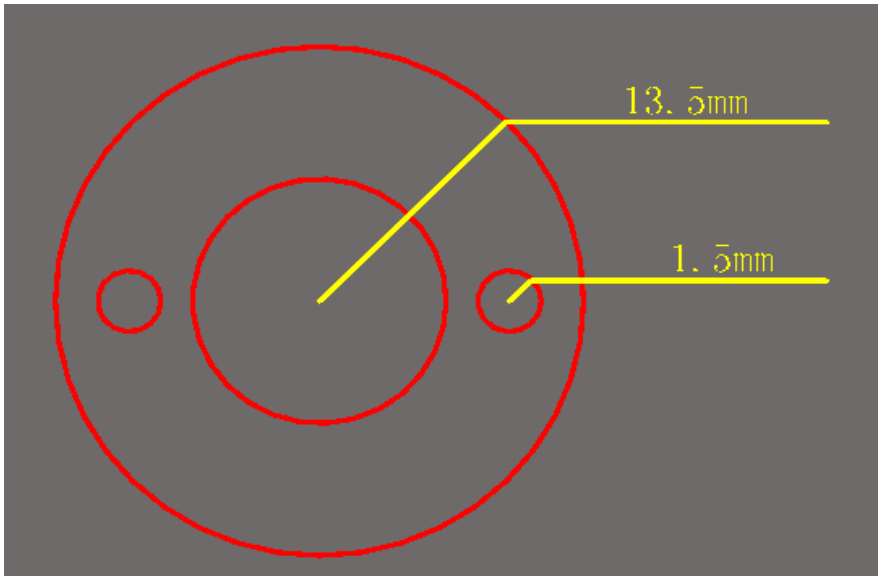


图 3.5 编码器安装孔定位图

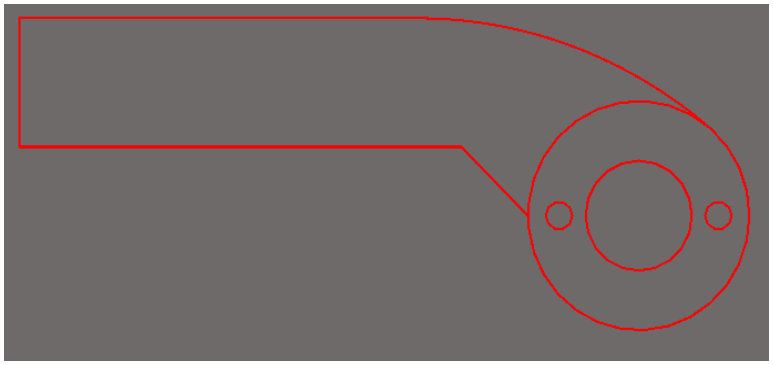


图 3.6 编码器外形参数

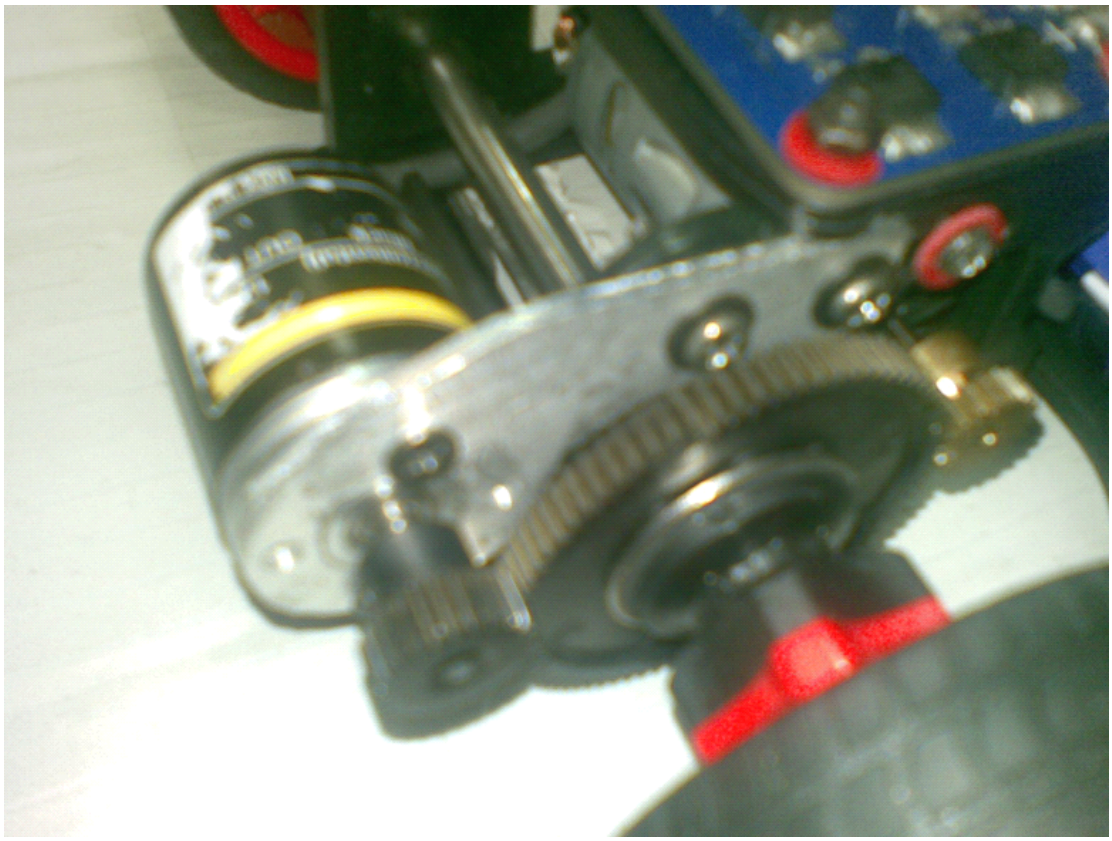


图 3.7 编码器的安装

3.5 前轮定位

3.5.1 主销后倾角

所谓主销后倾角，即是前轮的转向轴与铅垂线之间在纵向平面上的夹角。该角度以转向轴上端向后为正（小车行驶方向为前方），该角度具有使小车“直线行驶更稳定，前轮转向后自动回正”的功能，但是它增加了小车的转向阻力。所以需要调至适当的角度。当两个前轮的主销后倾角不一致时，小车会跑偏，且偏向后倾角比较小的一边。调整前轮黄色介子即可改变后倾角度。

3.5.2 主销内倾角

所谓主销内倾角，即是转向轴与铅垂线之间在横向平面上的角度。该角度会直接影响小车转向性能和稳定性，使该角度适当大一些可保证小车转向性和稳定性。

3.5.3 前轮前束

小车属于后轮驱动，故一般采取前轮前束。如果前轮拉杆长度、角度不合适则会造成小车转弯时车轮抖动的情况，这时需要调整好前束。

3.6 机械部分小结

小车机械方面的调整是相对比较繁琐的，一个设计不合理的零件往往会带来诸多问题，一个调整不当的前轮角度又有可能使小车跑得不流畅。如今，机械部分在智能车比赛中已经成为制胜的关键之一。只有有了好的机械，才能诞生一辆优秀的智能车。尤其是在小车高速奔驰时，机械的精良程度决定了最终的成绩。

总言之，智能车不是设计好硬软件就行的，机械的重要程度同样不可替代。

第四章 系统硬件的设计及实现

4.1 硬件设计方案

关于硬件设计，我们制定了一些目标原则：简约、可靠、美观。

简约。为了在硬件方面做到尽量少出问题，我们选用了成熟的设计方案，芯片选用可靠简单易用的，并充分利用硬件资源。

可靠。电路板进行单片化设计，将电源模块、调试模块、图像信号处理模块、速度反馈信息处理模块和各个接口电路设计在一块主板上。主板直接用螺丝螺母固定于底盘上。电路中做好了模拟部分和数字部分的隔离、各个电源的滤波，以及对干扰信号的屏蔽工作。各个接口连接可靠牢固。

美观。将液晶裸屏和按键直接安置在主板上，缩减了原本的调试模块电路板，使得车上的电路板数量得到了减少。主板和驱动板外形设计根据车模尺寸形状量身打造。减少了各个电路板之间的连线，使整车看上去简洁美观。主板效果见图 4.1。

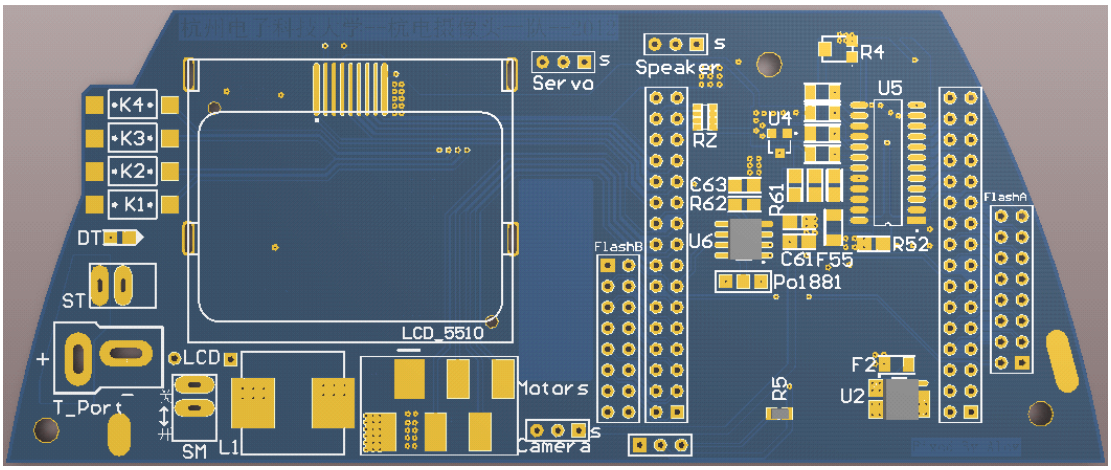


图 4.1 主板一览

4.2 电源电路设计

全部硬件电路的电源由 7.2V、2000mAh 的可充电镍镉电池提供。由于系统中的各个电路模块所需要的工作电压和工作电流各不相同，因此电源模块应该包括多种稳压电路，将充电电池电压转换成各个模块所需要的电压。

本系统所需的电压种类大致如下：

①3.3V，为 FLASH 模块、液晶供电；

②5V，为最小系统、图像信号处理电路、速度反馈信号处理电路供电；

③12V，为摄像头模块、栅极驱动芯片供电；

④电池电压，为 H 桥（电机）、舵机供电，其中电池电压经过一个二极管降压 0.7V 之后供给舵机。

电源分配示意图如下所示：

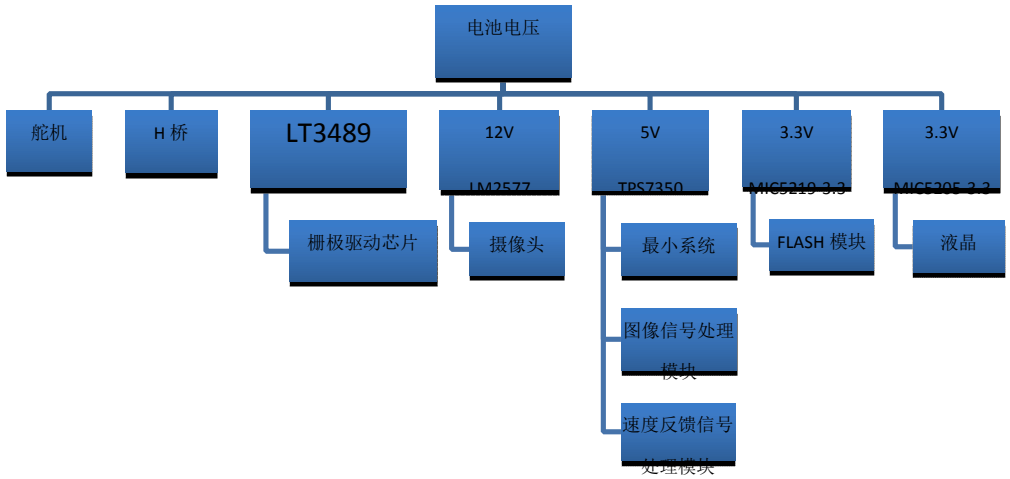


图 4.2 电压分配示意图

电源部分原理图如下图所示：

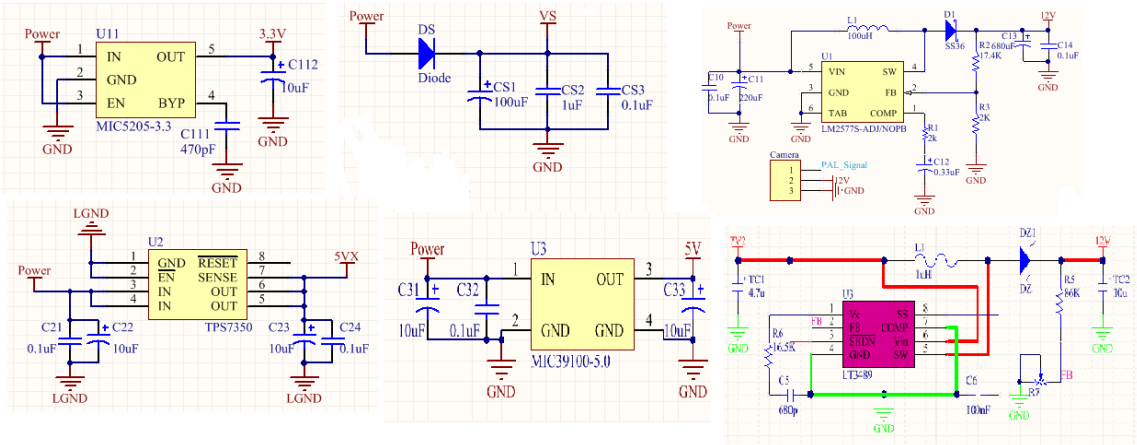


图 4.3 电源电路原理图

4.3 图像信号处理电路设计

摄像头输出的视频信号为模拟信号，而单片机可以处理的是数字信号，所以需要把模拟视频信号离散成数字信号，这就要用到模数转换技术。一般而言单片机都有 A/D 转换模块。我们所用的单片机为组委会指定的新一代单片机

MC9S12XS128。S12 内置了 2 个 10 位/8 位的 A/D 模块 ATD0 和 ATD1，通称为模数/转换器(ATD)。但是 S12 微控制器 ATD 的最高转换频率仅约为 2MHz，无法满足我们的要求，所以我们自己设计了外部高速 AD 电路将模拟视频信号先转换位数字信号后再通过 I/O 口送至单片机内部进行处理。我们所选用的 8 位高速 A/D 转换器为德州仪器生产的 TLC5510，并选择使用外部可调基准电压。该 AD 转换电路原理图如下所示：

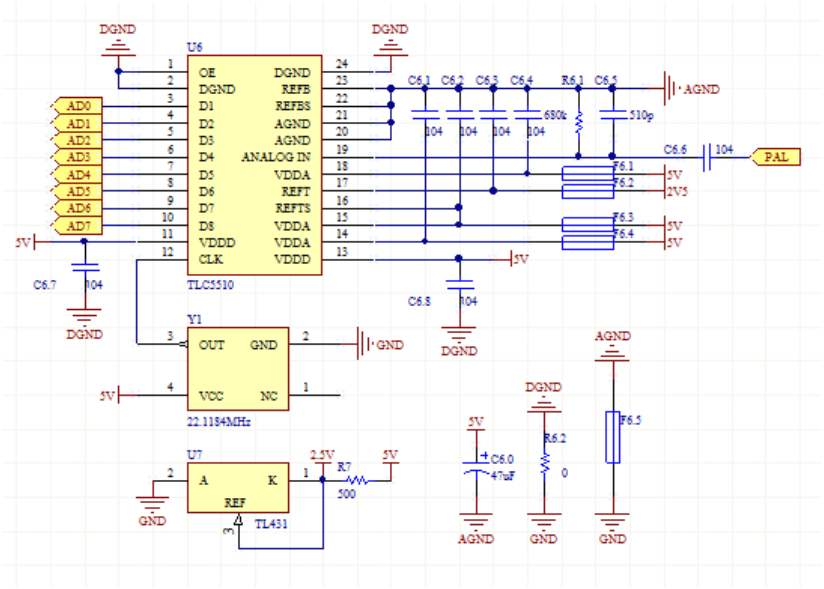


图 4.4 外部 AD 转换电路

为了采集图像信息，CPU 需要根据行、场同步信号启动 A/D 转换器，具体过程即在等待到行同步信号之后启动 AD 转换器，同时通过定时器设定 AD 采集的时间，然后等待下一次行中断并启动。由于视频信号的变化很快，所以需要另外设计同步分离电路。在本方案中，使用了 LM1881 视频同步分离集成芯片来获取奇场同步信号、偶场同步信号和行同步信号，并将此同步信号连接到单片机的中断输入端口。该视频信号同步分离电路如下图所示：

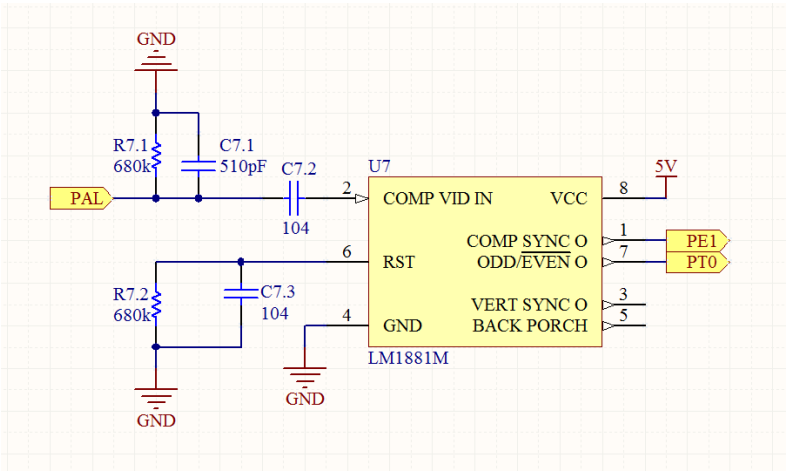


图 4.5 视频信号同步分离电路

4.4 速度反馈信号处理电路设计

由于本届的车模本届车模为单电机，所以编码器电路比较简单，只需供电和将编码器信号线上拉就行了。

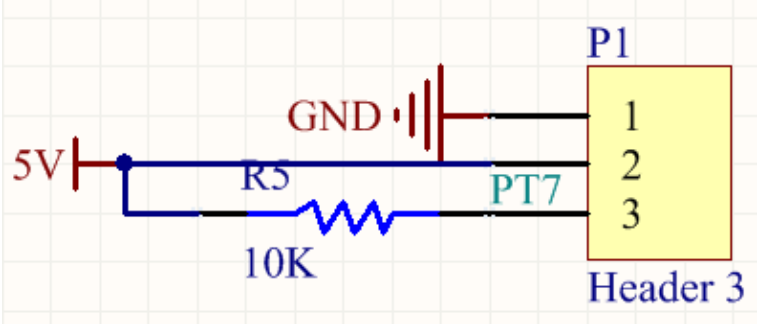


图 4.6 速度反馈信号处理电路

4.5 电机驱动电路设计

本届车模电机功率十分小，在实际工作时转速不快，加减速亦不明显，电机空载时(给 8.4V 工作电压)电流仅在 0.15~0.19A。电机本身的性能直接限制了小车的行驶速度和加减速性能，因此，对驱动电路的参数要求并不是十分严格，驱动电路的参数性能对电机的发挥并无明显的影响。本次设计的电机驱动电路较往届不同，使用了 NMOSFET 搭建的全桥。性能较 BTS7960 等集成驱动桥优秀，但在该车模上并无明显优势。为了减少驱动电路对主板的干扰，我们将驱动电路做成独立小板置于电机上方。驱动桥 PCB、原理图如下所示：

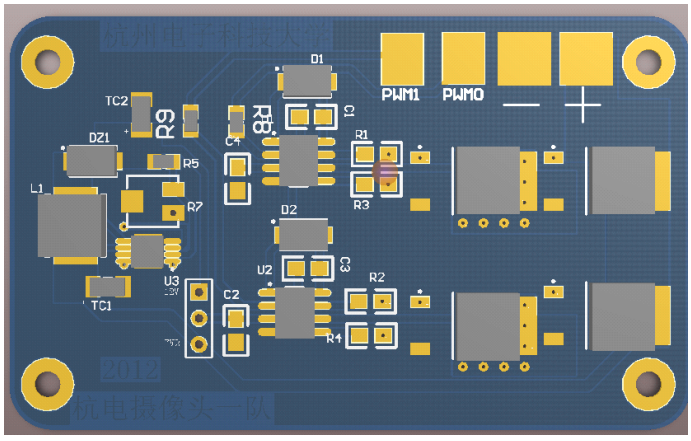


图 4.7 驱动电路板效果图

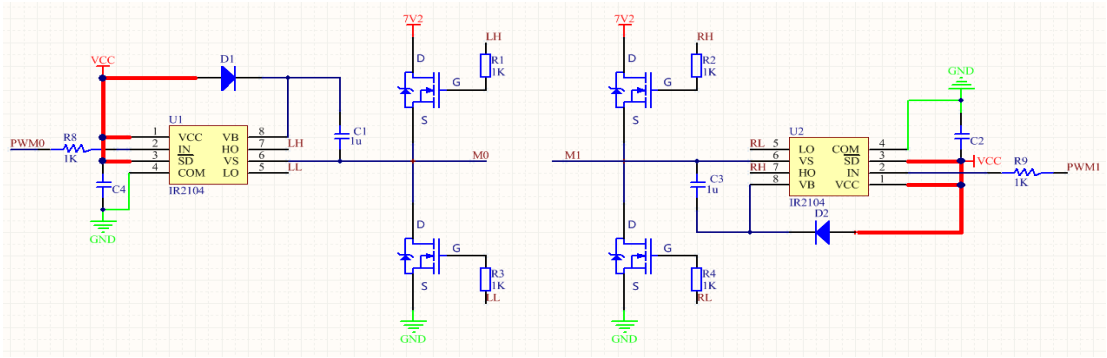


图 4.8 驱动电路原理图

PWM0、PWM1 分别代表 2 路 PWM 信号。栅极驱动芯片为国际整流器件公司生产的 IR2104，使用升压电路给该芯片提供 12V 左右的工作电压，使之可以顺利开启上下桥臂的 NMOS 管。每个电机分别由一个全桥来驱动，可实现电机的正逆转，而加在上桥臂 NMOS 管漏极上的电压为电池电压。

下面以一个全桥为例解释一下该电路的各个工作状态：

① PWM1 为高、PWM0 为低时，节点 RH 上电压为高、RL 为低、LH 为低、LL 为高，这时 Q2、Q3 开启，Q1、Q4 关断，电流流向为 M1 到 M0，此时电机正转。

②当 PWM1 为低、PWM0 为高时，节点 RH 上电压为低、RL 为高、LH 为高、LL 为低，这时 Q2、Q3 关断，Q1、Q4 开启，电流流向为 M0 到 M1，此时电机逆转。

4.6 调试电路设计

调试电路我们沿用了往届的“液晶+按键”模式。继承上届的经验，我们将液晶和按键直接放在了主板上，仍采用 nokia5110 裸屏加 4 按键的调试方式。这种设计减少了电路板的数量，并在一定程度上降低了车的重心，也增加了车的美观程度。该模块的原理图如下所示：

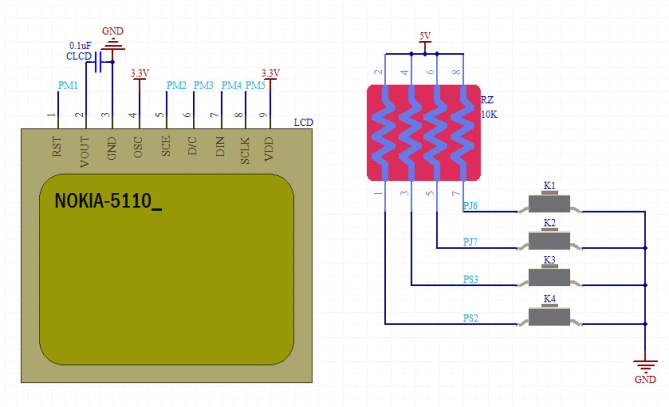


图 4.9 调试模块电路

4.7 编码器

本次使用的欧姆龙编码器输出信号为方波，所以不需要外部电路对其进行整形，可以直接传给单片机处理。

4.8 核心板电路设计

最小系统电路是整个系统的核心，它主要完成各种信号的处理、舵机电机的控制等工作。我们采用大赛主委会制定的 80 脚的 MC9S12XS128 单片机，在电路设计上，我们采用了模块化设计的思想，将 S12 的所有管脚引出并将最小系统做成单独一块电路板，极大的方便了我们调试小车。为了追求轻便小巧的车模系统，为了让最小系统符合我们的实际要求，我们采用了自制的 S12 核心板。经过大量测试，该核心板工作稳定，不易出现复位重启等现象，完全满足我们的要求。该核心板包含 S12 最小系统、时钟电路、BDM 接口电路、SPI 接口电路、电源滤波电路及复位电路。其原理图及 PCB 图如下所示：

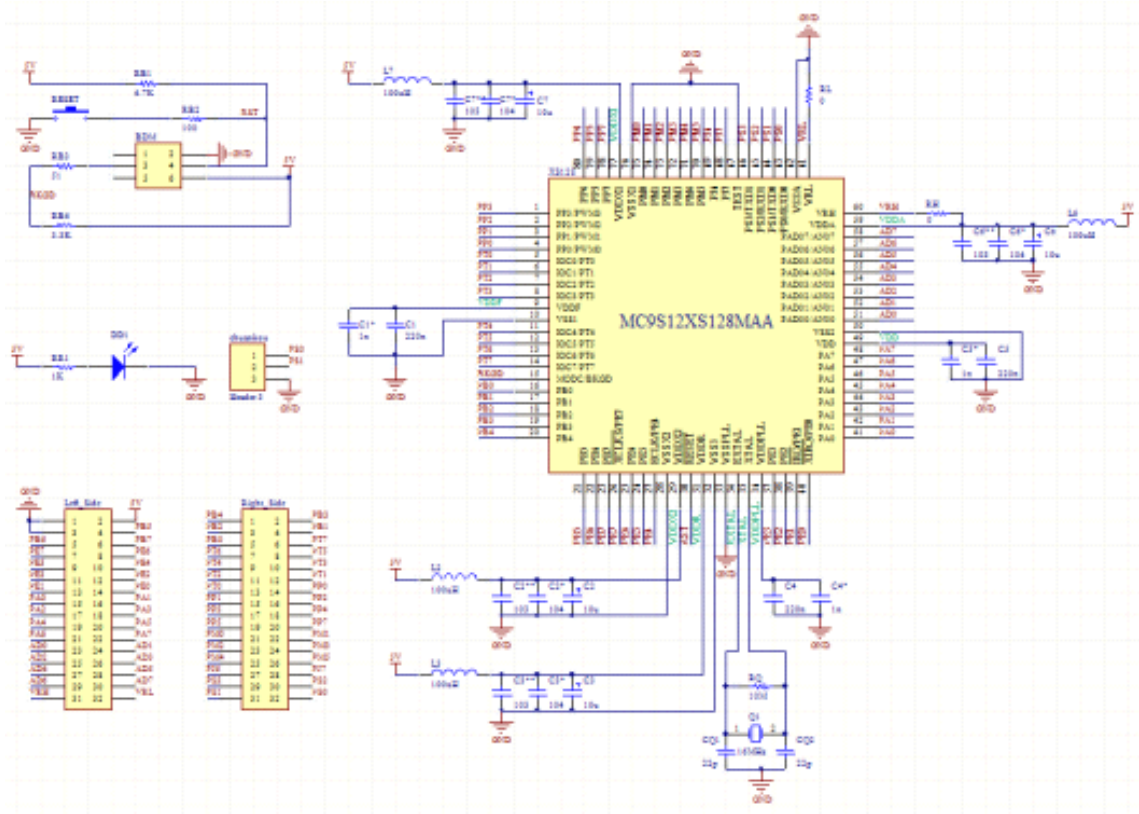


图 4.10 核心板电路原理图

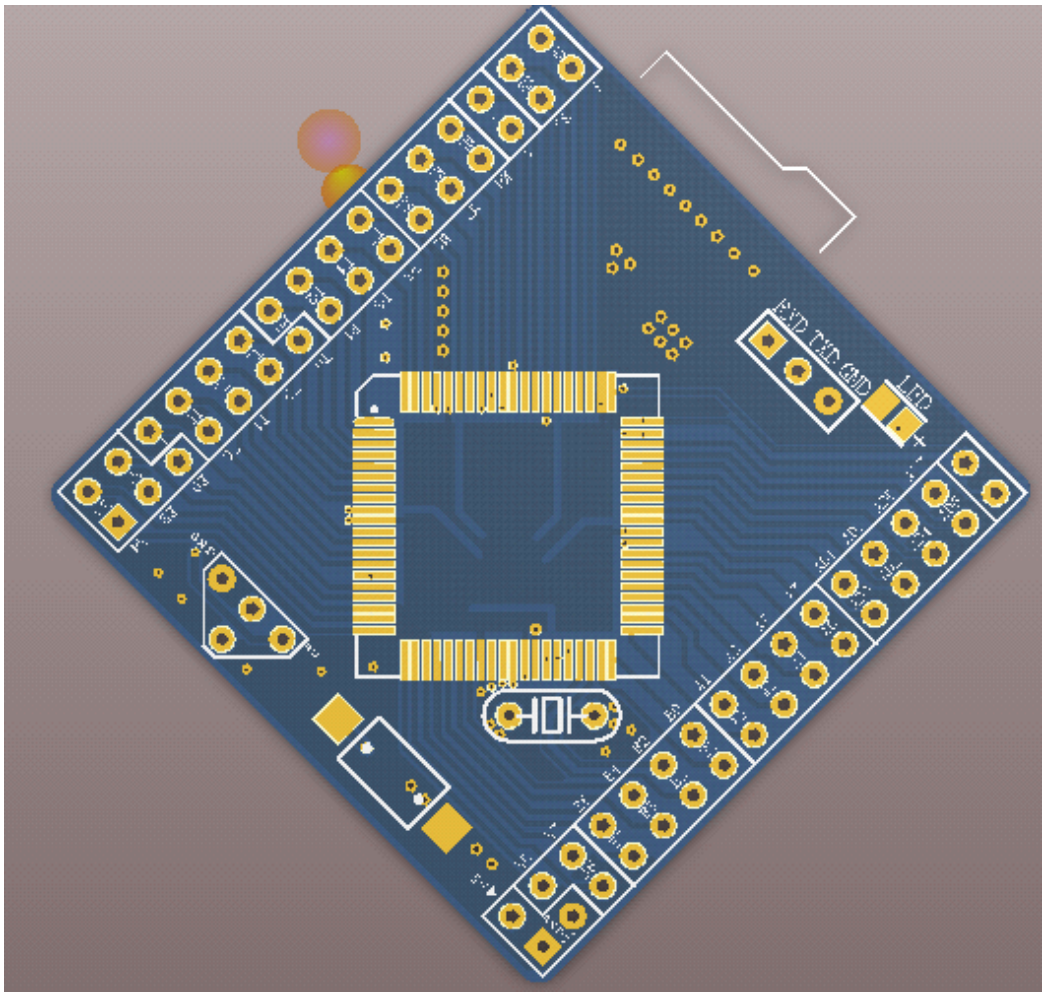


图 4.11 核心板 PCB 图

4.9 硬件电路部分小结

硬件电路的关键是稳定，智能车系统的正常工作即是以硬件的稳定为前提的。而整个电路系统中，电源部分的稳定性亦至关重要，只有各个电源供应稳定可靠纹波小才能保证其它电路模块的正常工作，尤其是单片机最小系统。稳定的电路板为后续的工作奠定了坚实的基础。在电机转速急减急加或舵机打大角度时，电池电压会产生大的波动，此时若 5V 稳不住单片机就会出现复位等问题。合理的使用滤波电容可以减少此类的影响，使电压稳定。

我们的电路中既包含模拟电路又包含数字电路，为提高系统稳定性，电路中区分了模拟地与数字地。为保证电路板的稳定性和可靠性，我们将设计好 的 PCB 文件送入工厂打样。各个电路板之间通过排针排母或杜邦线相连，保证了整体电路系统的稳定性。

第五章 系统软件的设计与实现

5.1 系统软件流程图

软件设计部分主要包括：图像采集、图像处理、赛道判断、舵机打角、电机控制以及速度反馈处理，系统流程图如图 5.1 所示。

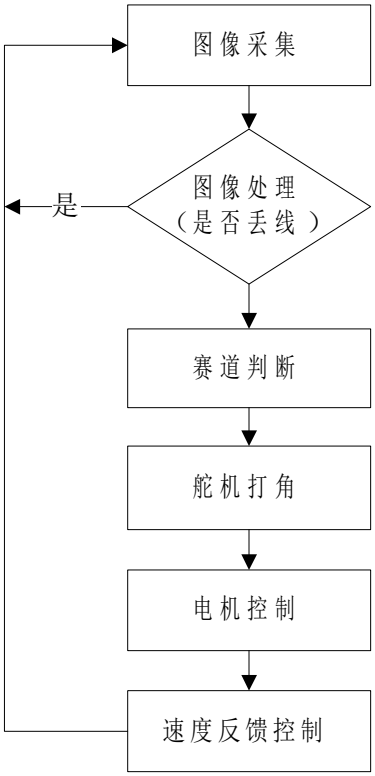


图 5.1 系统流程图

5.2 图像采集

图像采集部分主要有摄像头、视频分离芯片 LM1881。摄像头主要由镜头、CCD 图像传感器、图像信号形成电路、同步信号发生器、预中放、CCD 驱动器等外围电路组成。LM1881 用于分离视频信号中的行同步脉冲和和奇偶场同步脉冲以供单片机处理。

5.2.1 摄像头工作原理

我们比赛用的摄像头主要分为黑白和彩色两种。彩色摄像头对比度比黑白的要好，对赛道不同背景色有较好的抗干扰能力，但考虑到比赛时我们只关注黑线的提取而不关心图像彩色信息，所以我们只选用黑白摄像头。接下来，我们用眼睛来打比方以说明成像原理：当光线照射景物，景物上的光线反射通过人的晶状体聚焦，在视网膜上形成图像，之后视网膜的神经将信息传导到大脑，我们就能看见东西了。摄像头成像的原理和眼睛非常相似，光线照射景物，景物上的光线反射通过镜头聚焦，CCD 图像传感器就会感知到图像。而感知图像的具体过程是这样的：摄像头按一定的分辨率，以隔行扫描的方式采集图像上的点，当扫描到某点时，就通过图像传感芯片将该点处图像的灰度转换成与之一一对应的电压值，然后将此电压值通过视频信号端输出。如图 5.2 所示，摄像头连续地扫描图像上的一行，则输出就是一段连续的电压信号，该电压信号的高低起伏反映了该行图像的灰度变化。当扫描完一行，视频信号端就输出一个低于最低视频信号电压的电平（如 0.3V），并保持一段时间。这样相当于，紧接着每行图像信号之后会有一个电压“凹槽”，此“凹槽”叫做行同步脉冲，它是扫描换行的标志。然后，跳过一行后（因为摄像头是隔行扫描的），开始扫描新的一行，如此下去，直到扫描完该场的视频信号，接着会出现一段场消隐区。该区中有若干个复合消隐脉冲，其中有个持续时间远长于其它的消隐脉冲，称为场同步脉冲，它是扫描换场的标志。场同步脉冲标志着新的一场的到来，不过，场消隐区恰好跨在上一场的结尾和下一场的开始部分，得等场消隐区过去，下一场的视频信号才真正到来。摄像头每秒扫描 25 幅图像，每幅又分奇、偶两场，先奇场后偶场，故每秒扫描 50 场图像。奇场时只扫描图像中的奇数行，偶场时则只扫描偶数行。摄像头有两个重要的指标：分辨率和有效像素。分辨率实际上就是每场行同步脉冲数，这是因为行同步脉冲数越多，则对每场图像扫

描的行数也越多。事实上，分辨率反映的是摄像头的纵向分辨能力。有效像素常写成两数相乘的形，如“320x240”，其中前一个数值表示单行视频信号的精细程度，即行分辨能力；后一个数值为分辨率，因而有效像素=行分辨能力×分辨率。

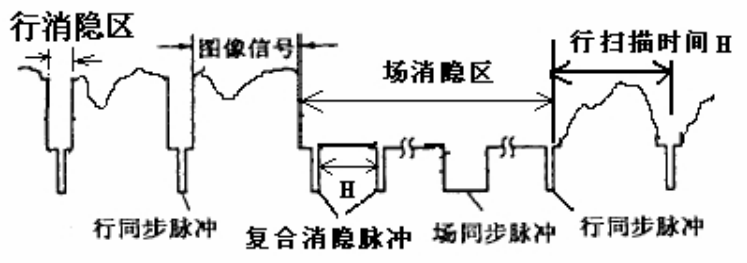


图 5.2 摄像头信号波形

5.2.2LM1881 工作原理

为了有效地对视频信号进行采样，首先要处理好的问题是如何提取出摄像头信号中的行同步脉冲、消隐脉冲和场同步脉冲，否则，单片机将无法识别所接收到的视频信号处在哪一场，也无法识别是在该场中的场消隐区还是视频信号区，更无法识别是在视频信号区的第几行。这里有两种可行的方法。第一，直接采用 A/D 转换进行提取。当摄像头信号为行同步脉冲、消隐脉冲或场同步脉冲时，摄像头信号电平就会低于这些脉冲模式之外时的摄像头信号电平。据此，可设一个信号电平阈值来判断 A/D 转换采样到的摄像头信号是否为行同步脉冲、消隐脉冲或场同步脉冲。第二，就是给单片机配以合适的外围芯片，此芯片要能够提取出摄像头信号的行同步脉冲、消隐脉冲和场同步脉冲以供单片机作控制之用。由于单片机的处理速度有限，而一些脉冲的间隔时间又较短，我们就采用了第二种方法进行信号提取。LM1881 视频同步信号分离芯片可从摄像头信号中提取信号的时序信息，如行同步脉冲、场同步脉冲和奇、偶场信息等，并将它们转换成 TTL 电平直接输给单片机的 I/O 口作控制信号之用。LM1881 的端口接线方式如图 5.3 所示。

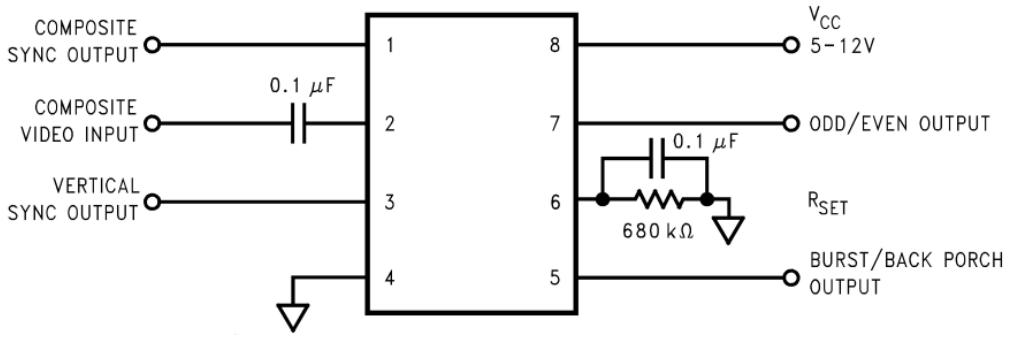


图 5.3 LM1881 端口描述

其中，引脚 2 为视频信号输入端，引脚 1 为行同步信号输出端（波形如图 5.4 中的 b）。 引脚 3 为场同步信号输出端，当摄像头信号的场同步脉冲到来时，该端将变为低电平，一般维持 230us，然后重新变回高电平（波形如图 5.4 中的 c）。引脚 7 为奇偶场同步信号输出端，当摄像头信号处于奇场时，该端为高电平，当处于偶场时，为低电平。事实上，不仅可以用场同步信号作为换场的标志，也可以用奇-偶场间的交替作为换场的标志。

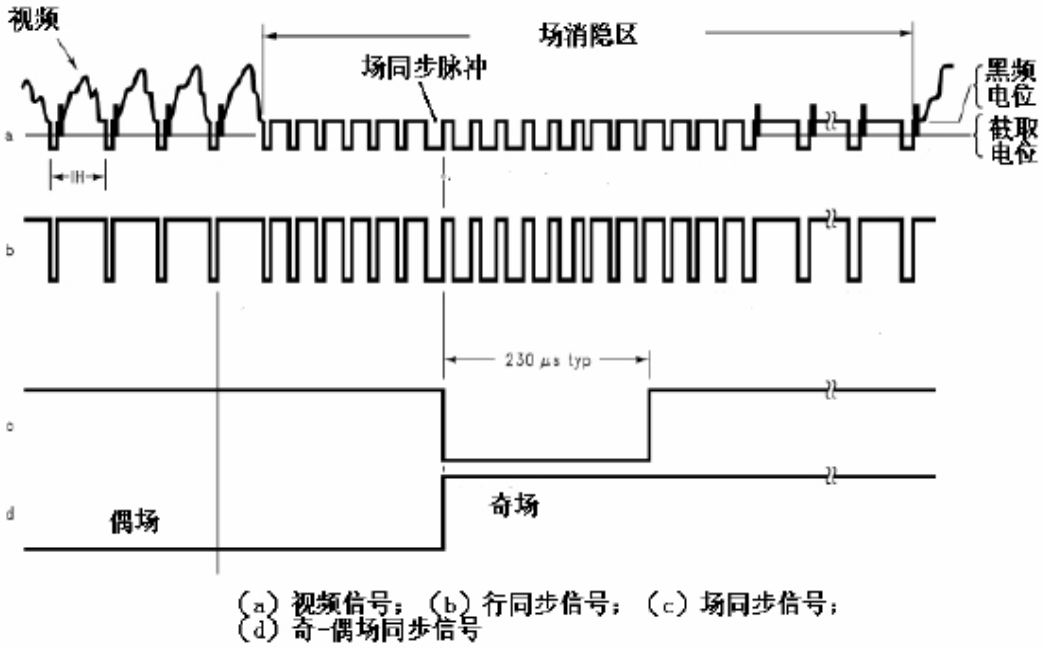


图 5.4 视频及其分离信号波形图

5.3 图像处理

图像处理的主要流程图如下所示：

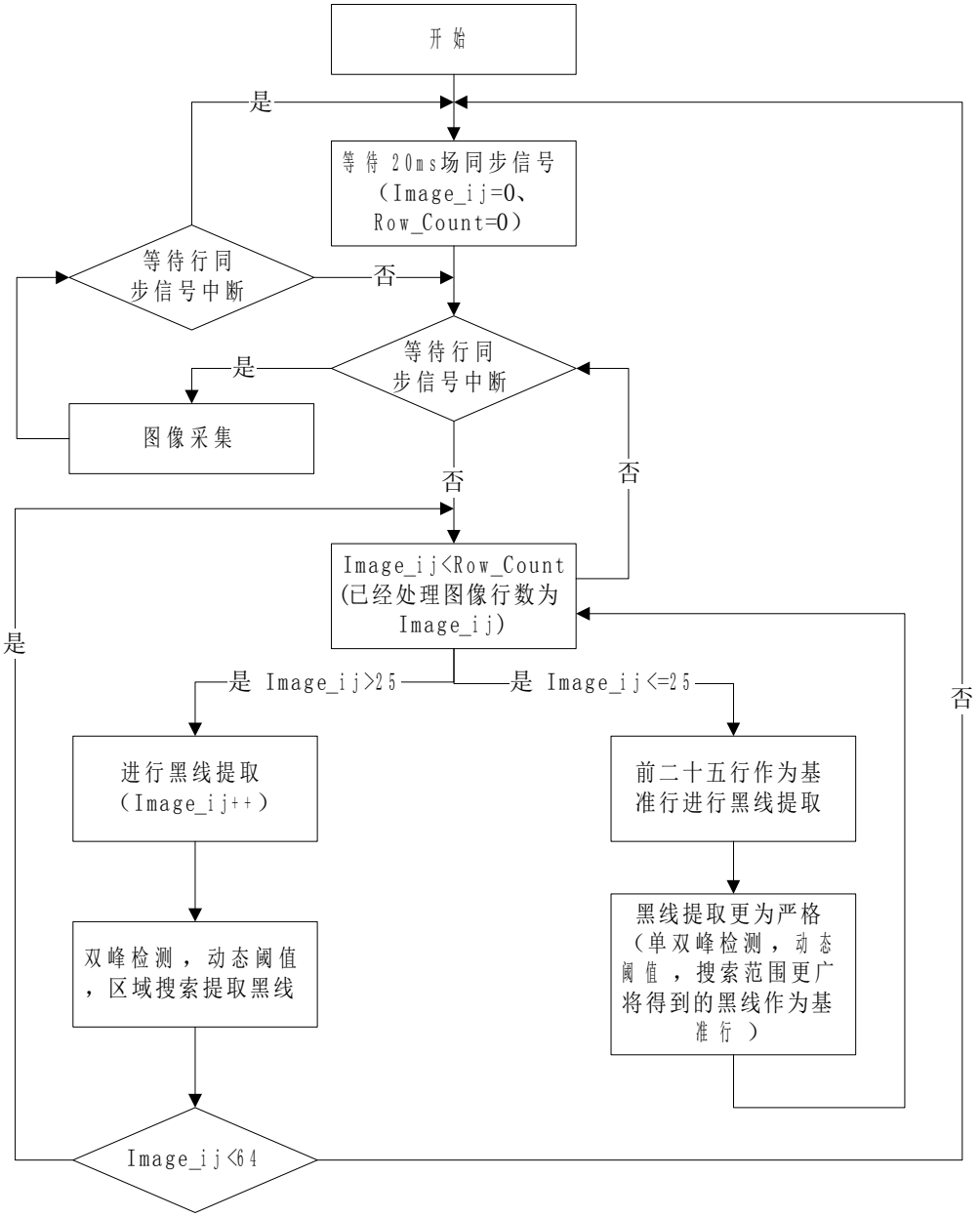


图 5.5 图像处理主要流程图

图像处理简单的来说就是根据摄像头传回来的视频信号中提取出黑线的位置。常用的黑线提取算法划分为二值化算法、直接边缘检测算法和跟踪边缘检测算法。

二值化算法的思路是：设定一个阈值 valve，对于视频信号矩阵中的每一行，从左至右比较各像素值和阈值的大小，若像素值大于或等于阈值，则判定该像素对应的是白色赛道；反之，则判定对应的是黑色的目标引导线。记下第一次和最后一次出现像素值小于阈值时的像素点的列号，算出两者的平均值，以此作为该行上目标引导线的位置。

直接边缘检测算法：采用逐行搜索的算法，首先找到从白色像素到黑色像素的下降沿和从黑色像素到白色像素的上升沿，然后计算上升沿和下降沿的位置差，如果大于一定的标准值，即认为找到了黑线，并可求平均值算出黑线的中心点。至于上升沿、下降沿的检测，可以通过上上次采样数与这次采样数的差值的绝对值是否大于一个阈值来判断，如果“是”且差值为负，则为上升沿；如果“是”且差值为正，则为下降沿。

跟踪边缘检测算法：由于黑色的目标引导线是连续曲线，所以相邻两行的左边缘点比较靠近。跟踪边缘检测正是利用了这一特性，对直接边缘检测进行了简化。其思路是若已寻找到某行的左边缘，则下一次就在上一个左边缘附近进行搜寻。这种方法的特点是始终跟踪每行左边缘的附近，去寻找下一列的左边缘，所以称为“跟踪”边缘检测算法。我们采用的是直接边缘检测算法，因为该方法抗环境光强变化干扰的能力更强，同时还能消除垂直交叉黑色引导线的干扰。由于智能车上安装的摄像头相对于赛道存在一定的倾斜角度，因此会造成采集到的赛道图像具有一定的梯形失真，即图像中的赛道远端窄、近端宽，远端图像不清晰而近端图像清晰可靠，如图 5.6 所示。

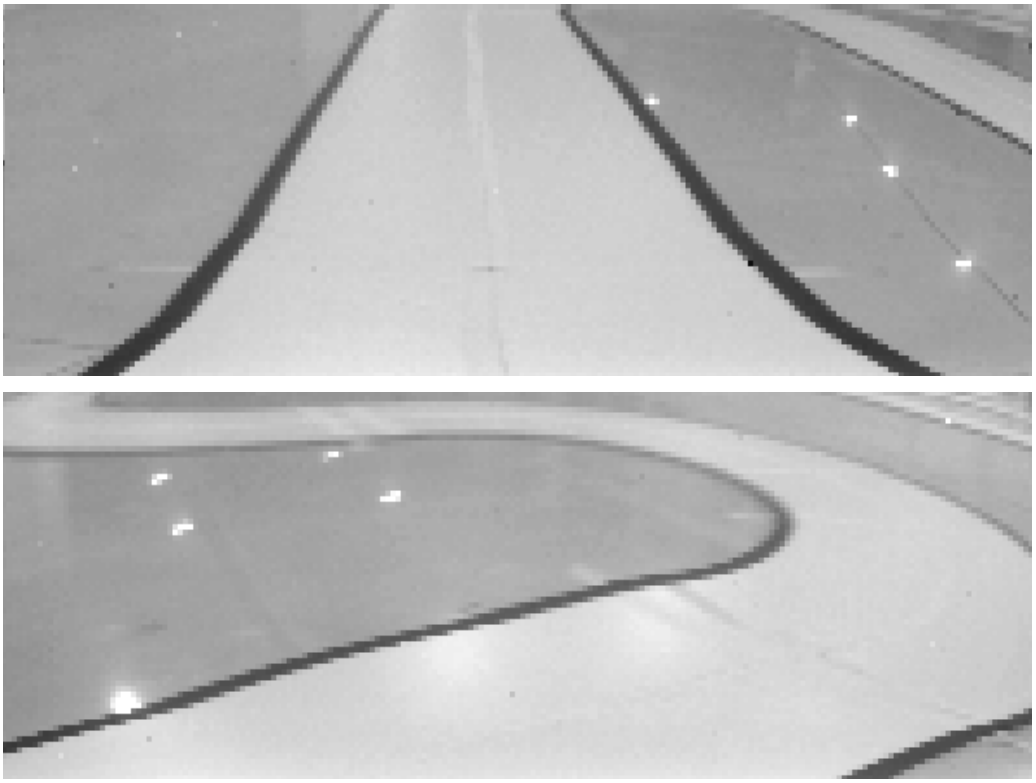


图 5.6 失真的图像

所以就将一场图像分为两部分，近端部分和远端部分。为了给单片机处理节约时间我们采用了全场动态范围来提取黑线即首先取得第一幅图像得到近处基准行黑线位置，在此基础上确定下一幅图像搜索范围由于黑先是连续变化的，远端部分黑线就根据前两行黑线位置的偏差量再加上一个固定范围来寻找。在图像滤波算法中，还应考虑以下几个方面：首先，根据图像模型去噪，例如，由于赛道的黑色引导线是绝对连续的，故两个中间有黑线的行之间不能有全白行（注意中间二字：如果黑线在边缘，则可能是由于摄像头的视野太窄或智能车身不正导致在过弯道时只能看到部分黑色引导线），这主要是解决光线对摄像头的反光问题；其次，在理想的情况下，根据赛道的黑色引导线的连续性，如果某一行求取的中心线位置与相邻的两行都相差很大，则可以认为该行数值错误，抛弃该行的数据或使用其前后两行数据的平均值来替代该错误数值用以校

正。这样一来提取出的黑线就比较可靠了，不过在实际调试中远处图像畸变比较大而且行距也大，偶尔还是可能会出现错误提取黑线，但是这对车体的控制影响不是很大。

摄像头车相对于电磁车与光电车最大的优势在于有较远的前瞻，所以图像精度就非常重要了即行分辨率和分辨率越高图取出来的黑线位置就越稳定可靠，这也是所有控制算法的基础。由于单片机内存只有 8K，存储图像数据的数组就不能开的太大，否则再加上一些全局变量，静态变量，临时变量，明显内存就不够用了。我们想到的第一个解决办法是买到更大内存的单片机可是这个没实现，经过尝试与多方设想我们采用动态内存刷新来提高图像精度然后再加上锁相环超频提高了图像的纵向与横向精度，从而使前瞻达到了原来的两倍多。为了使智能车适应能力更强抗干扰能力更好，我们将原来静态阈值改为动态阈值，使得每一行之间的阈值过度更加自然。不同的场合光线强度不一样从而阈值也不一样，相邻两行之间的灰度值不会相差很大。

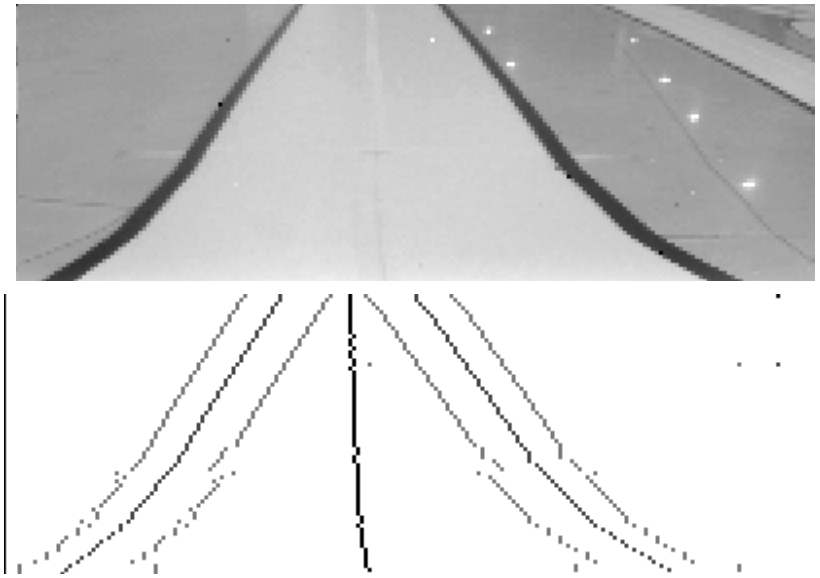
5.4 赛道判断

赛道判断就是在图像处理正确的基础上对提取出来的黑线进行分类，其中包括直道，弯道，S 弯，180°弯，坡道等等，代码如下：

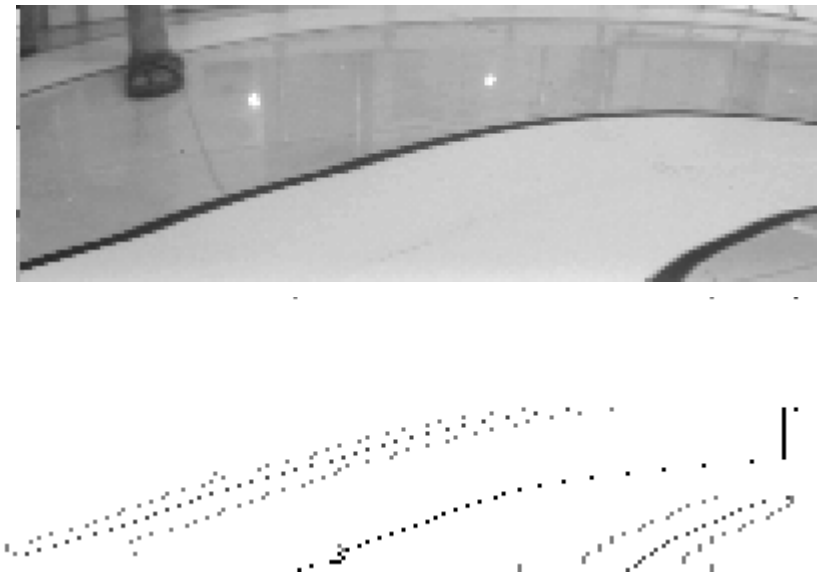
```
if(black_route[i]-black_route[i+1]>1)
{
S_right++; //S 弯右加加
}
else
{
if(black_route[i+1]-black_route[i]>1)
```

```
{
S_left++; //S  弯左加加
}
else
{
S_straight++;
}
}
```

其中数组 black_route[i]中存放每行黑线中心位置，通过比较 S_right、S_left、S_straight 的值再加上此时前瞻量 Effective_Rows 就很容易知道赛道类型如图 5.7 （含 7 张子图）所示。



（a）直道图像



（b）普通弯道图像

图 5.7 不同的赛道类型

由图 5.7 对比可知道我们将赛道分为三类。

第一类是前瞻 Effective_Rows 大以及 S_left、S_rights 两个量差不多，类型包括：弯与直道。

第二类是前瞻 Effective_Rows 不大但是 S_right、S_left 中一个比另一个明显大，类型包括：普通弯道，90°弯道，180°弯道和十字交叉线。

第三类是坡道，黑线丢失，起跑线以及其他特殊情况。

下面我来说一下坡道判断，坡道处理对所有的车都是一个比较棘手的问题，因为在过坡时可能出现黑线丢失和车速过快导致车体行进不稳，容易出赛道，经过大量数据分析和图像采集，我们研制出了两个方法对坡道进行判断。方法一：当车体上坡过程中距离坡顶越来越近而摄像头又看不到坡的另一端所以前瞻会连续减小再加上坡道位于直道上曲率变化小，抓住这两个特征基本上可以判断出来，但是该方法存在一个致命的缺点就是判断出坡道时车体已经几乎要到达坡顶了此时再减速的话已经来不及了所以我们采用第二个方法。方法二：车体在靠近坡道时前瞻会增加相比于直道上的同一前瞻黑线距离车体要近得多，根据图像远端窄、近端宽原则此时黑线宽度比同一前瞻直道上黑线宽度要

宽，于是我们取了车前一段区域内黑线，开了一个数组 Ramp_data[8]记录此段区域黑线的宽度，通过比较记录比较的记录下来的黑线宽度来确定是否是坡道，该方法优点是判断很准确而且可以提早判断出坡道，对于车体稳定过坡提供了保障，缺点是对图像精度要求比较高。

5.5 舵机打角控制

我们用的舵机打角控制比较简单，只根据光电式编码器反馈来的脉冲数转化为速度再用这个值乘上舵机反应滞后时间得出打角行位置，然后根据这个打角行位置与图像中心位置的偏差来计算出舵机的打角值。图 5.8 为舵机打角曲线图。 但是在后期调试中我们发现取车身前一段范围内黑线平均位置来确定舵机打角值这样赛车走的路径会得到很大程度上的优化。

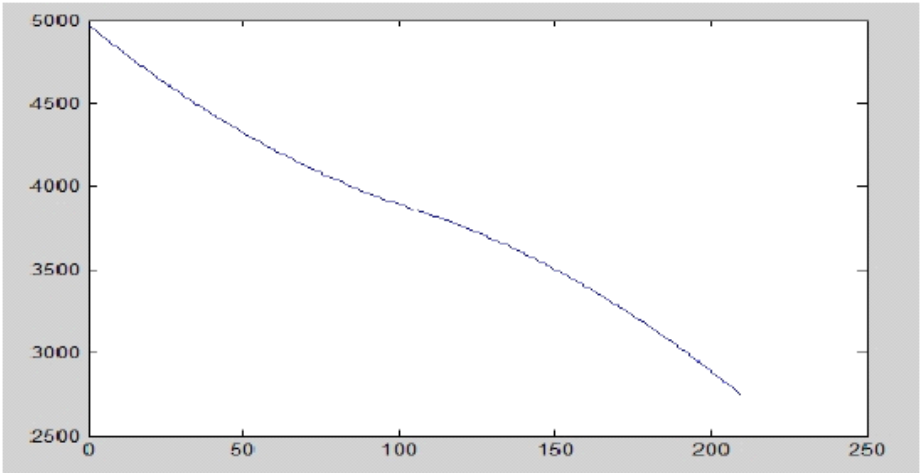


图 5.8 舵机打角曲线

5.6 电机控制以及速度反馈处理

速度控制策略的好坏就决定小车取得能否优秀成绩，如何给速度是关键，什么样的赛道配合什么样的速度。比如过弯时要提前减速使赛车在入弯前达到一个安全速度以免冲出赛道，过 S 弯时要同一般弯道和直道区分开，经过实际

测试我们将速度给定处理成二次曲线，这样达到的实际效果还是比较理想的。代码如下：

Speed_give=Speed_min+(Effective_Rows-40)*(Effective_Rows-40)*(Speed_max-Speed_min)/((57-40)*(57-40));

为了使赛车能快速达到给定速度，我们一开始采用的是 PID 控制，代码如下：

actual_given_PID=Kp*iError*7/10-Ki*LastError*7/10+Kd*PrevError;

PID 控制器系统原理框图如图 5.9 所示。将偏差的比例（KP）、积分（KI）和微分（KD）通线性组合构成控制量，对被控对象进行控制，KP、KI 和 KD 3 个参数的选取直接影响了控制效果。

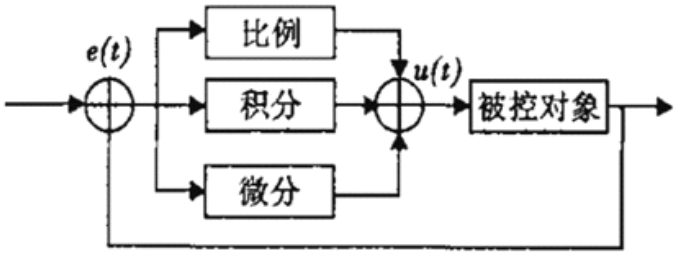


图 5.9PID 控制系统原理图

在经典 PID 控制中，给定值与测量值进行比较，得出偏差 e（t），并依据偏差情况，给出控制作用 u（t）。对连续时间类型，PID 控制方程的标准形式为，u（t）= Kp ［ e（t）+ ∫e（t）dtTI+ TDde（t）dt］。式中，u（t）为 PID 控制器的输出，与执行器的位置相对应；t 为采样时间；KP 为控制器的比例增益；e（t）为 PID 控制器的偏差输入，即给定值与测量值之差；TI 为控制器的积分时间常数；TD 为控制器的微分时间常数。

离散 PID 控制的形式为 u（k）= KP｛e（k）+Σe（k）T/TI+［e（k）-e（k - 1）］TD/T｝。式中，u（k）为第 k 次采样时控制器的输出；k 为采样序号，k = 0，1，2 ；e（k）为第 k 次采样时的偏差值；T 为采样周期；e（k - 1）为第（ k

- 1）次采样时的偏差值。

从系统的稳定性、响应速度、超调量和稳态精度等方面来考虑，KP、KI、KD 对系统的作用如下。

①系数 KP 的作用是加快系统的响应速度，提高系统的调节精度。KP 越大，系统的响应速度越快，系统的调节精度越高，但易产生超调，甚至导致系统不稳定；KP 过小，则会降低调节精度，使响应速度缓慢，从而延长调节时间，使系统静态、动态特性变坏。

②积分系数 KI 的作用是消除系统的稳态误差。KI 越大，系统的稳态误差消除越快，但 KI 过大，在响应过程的初期会产生积分饱和现象，从而引起响应过程的较大超调；若 KI 过小，将使系统稳态误差难以消除，影响系统的调节精度。

③微分作用系数 KD 的作用是改善系统的动态特性。其作用主要是能反应偏差信号的变化趋势，并能在偏差信号值变的太大之前，在系统引入一个有效的早期修正信号，从而加快系统的动作速度，减少调节时间。

KP、KI、KD 与系统时间域性能指标之间的关系如表 5.1 所示，图 5.10 是 PID 控制系统仿真结果。

表 5.1 PID 调节参数与系统时间域性能指标间的关系

参数名称	上升时间	超调量	过渡过程时间	静态误差
KP	减小	增大	微小变化	减小
KI	减小	增大	增大	消除
KD	微小变化	减小	减小	微小变化

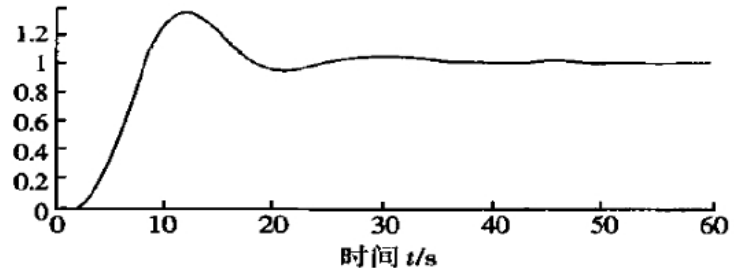


图 5.10 PID 控制系统仿真结果

经过很长时间的测试，我们最终采用了增量式 PID，因为通过实验证明这样能让电机有较好的加减速能力，下图是我们通过 Matlab 得出的速度跟踪曲线，虚线代表给定量，实线代表反馈量。通过图 5.11 可以看出此控制算法还是较为理想的。

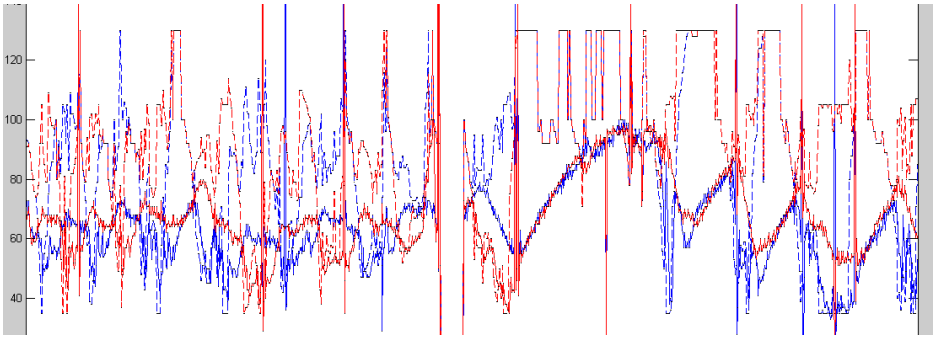


图 5.11 速度跟踪曲线

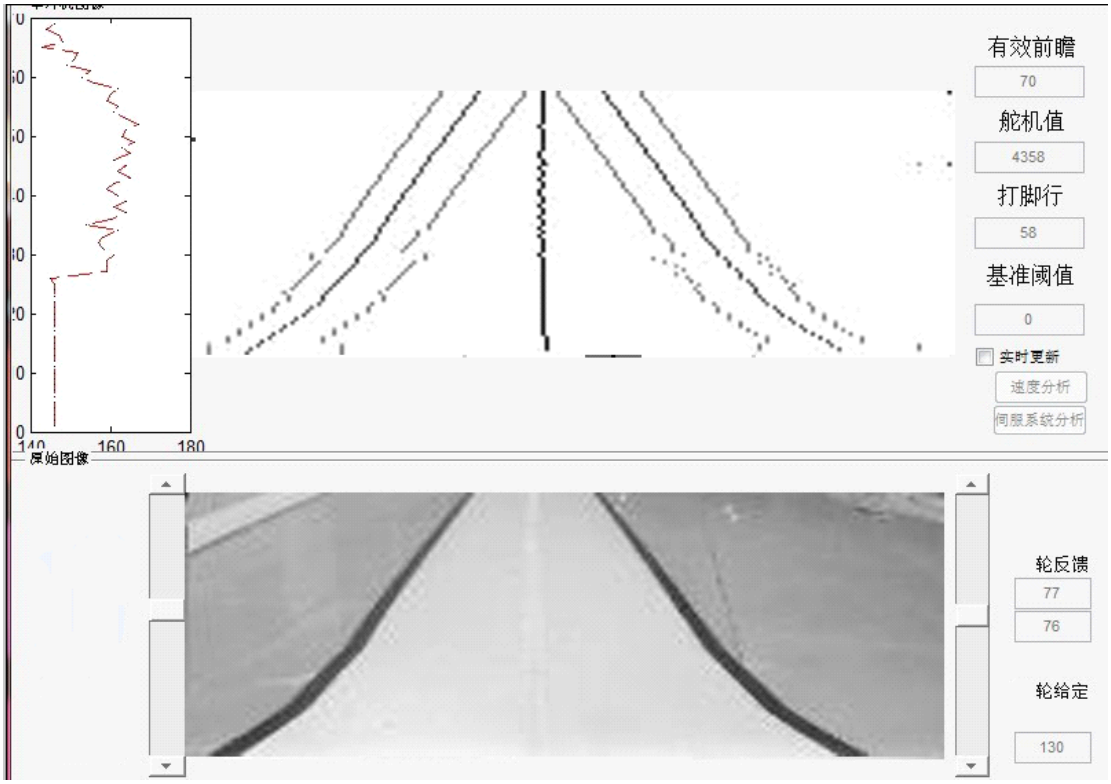
5.7 上位机调试软件的设计

我们在上届 matlab 上位机的基础上添加了双边图像的显示以及新增数据的处理，图 5.12 是上位机软件界面。

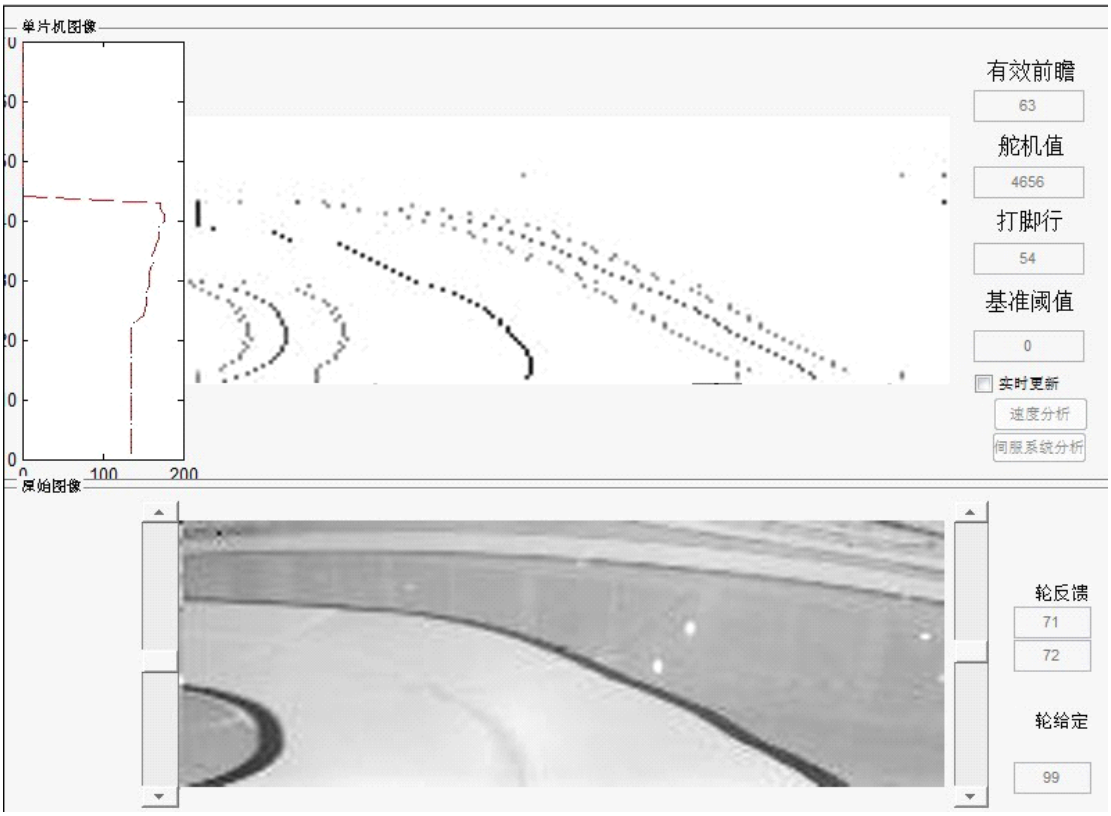
图 5.12 上位机软件界面

该上位机左上角两个空白部分分别接收原始图像和有效数据，两个空白框

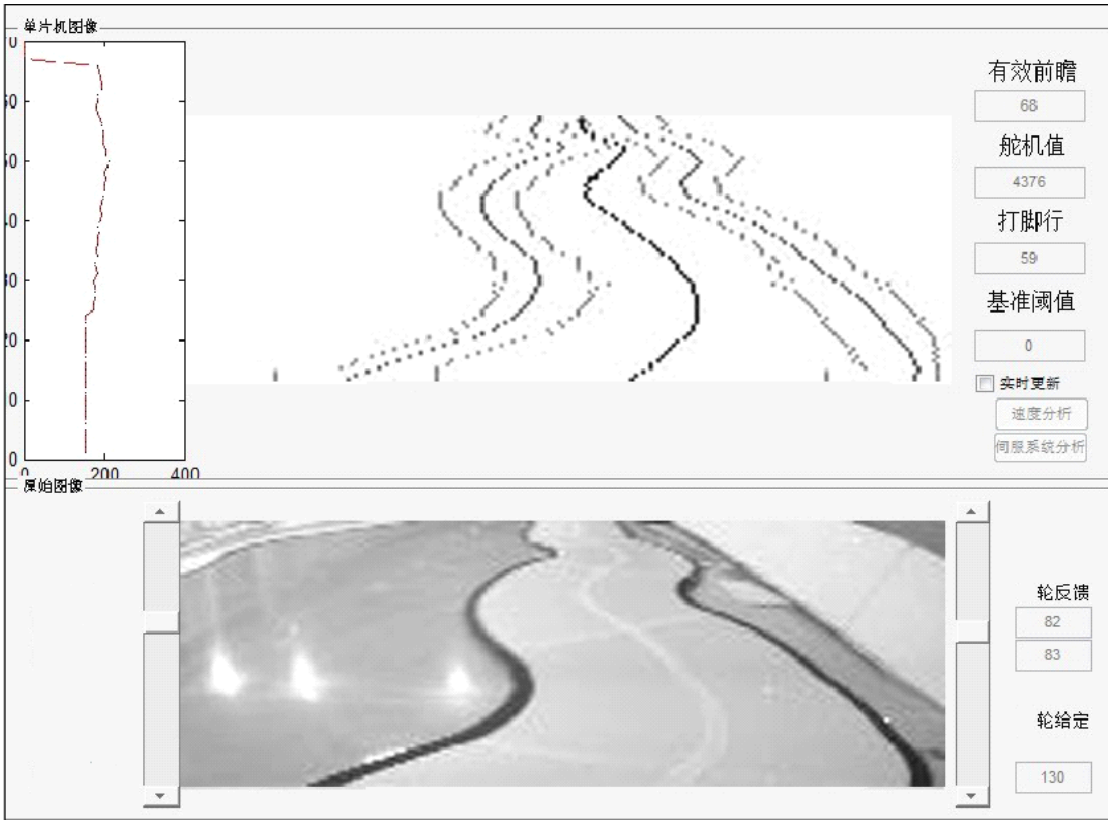
下面是有效数据的分析显示区，可以显示所有控制数据包括赛道类型，速度控制值，舵机控制值，进出弯判断等等。有效数据分析如图 5.13 （含 3 个子图）所示。



（a）直道数据分析


单片机图像
0
100
200
有效前瞻
63
舵机值
4656
打脚行
54
基准阈值
0
实时更新
速度分析
伺服系统分析
原始图像
轮反馈
71
72
轮给定
99

（b）弯道数据分析



（c）S 弯数据分析

图 5.13 有效数据分析图

5.8 Flash 调试模块设计

由于摄像头车模所需处理的信息量非常大，在调试过程中往往需要存储这些信息以供研究分析，而 S12 的存储空间远不能满足我们的要求。所以我们设计了外部 FLASH 来存储海量信息，这让我们在图像处理上有很多的发挥空间。通过 FLASH，我们存下了我们需要的图像信息和处理之后的信息，然后通过串口发送给上位机，这样我们就可以方便地分析出图像上的各种问题，以便在程序上作出修改。FLASH 模块的电路原理图如下所示。

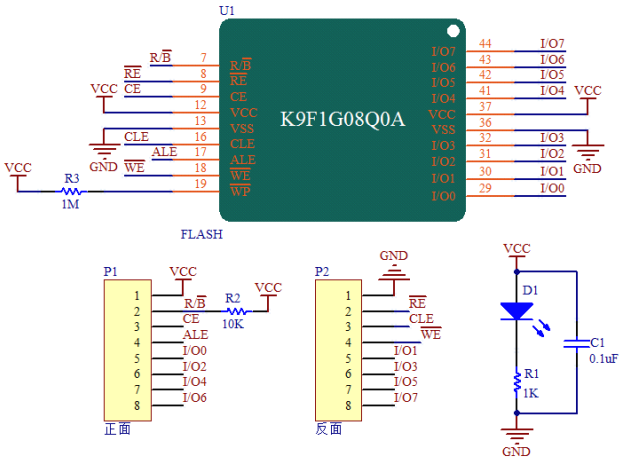


图 5.14 FLASH 模块原理图

第六章 总结与参数统计

6.1 总结

本报告详细介绍了我们为第六届全国大学生智能汽车大赛而准备的智能车系统方案。其中涉及机械部分、硬件电路部分和软件部分，机械部分包含舵机的安装、摄像头的安装和编码器的安装；硬件电路部分包含电源电路设计、图像处理电路设计、速度反馈信号处理电路设计、最小系统设计、液晶按键电路设计和电机驱动电路设计；软件部分包含图像处理、赛道判断、前轮舵机打角控制策略以及速度控制算法及实现。

纵观整个系统，我们在车模的机械硬件电路和软件上都做出了创新与改进:

①机械制作水平提高。我们在机械设计上力求简洁轻巧，重要的机械部分，例如摄像头的支架，采用车床加工，使机械零件的可靠性大大提高。

②硬件电路板设计合理美观。我们在设计电路板时力求做到布局合理、布线科学、外形美观。这样设计出来的电路板易于安装拆卸、在一定程度上降低了整车的重心、稳定可靠、为后期的调试节省了大量的返修时间以及使整车看上去简洁美观。

③软件设计更加成熟。成熟可靠的图像处理为智能车创造了一个良好的进步空间，大大方便了后期的调试。成熟的舵机电机控制算法使车模以高速顺利无误的完成比赛。

当然，由于时间和能力有限，我们制作的车模还存在很多需要改进的地方：

①测速传感器。我们在前期尝试过使用光栅来进行车速传感，但由于种种原因最后放弃了光栅使用了价格相对比较昂贵的光电编码器，光电编码器较之光栅好用但会增加车模重量。

②摄像头的选取。在前期，我们尝试过模拟 CMOS 摄像头、数字 CMOS 摄

像头，但都因为时间精力能力有限不了了之，最后还是沿用了之前的模拟 CCD 摄像头方案。CCD 摄像头外围电路复杂，故导致整个模块较重，且 CCD 工作电压在 12V 左右，故需要 DC-DC 升压电路去供电。CCD 摄像头虽然有图形动态特性好等优点，但有被 CMOS 摄像头代替的趋势。故在摄像头的选取上我们做的还不够好。

总结整个设计制作智能车的经过，我们学会了用软件设计零件、学会了使用车床铣床钻床锉刀、学会了在网上购买芯片、学会了用电路设计、学会了向工厂投板打样、学会了焊接各种贴片元器件、学会了汽车前轮的调教、学会了如何科学的给镍镉电池充放电、学会了如何进行自动化控制……到后期，我们基本上把实验室当作了家，日夜不分的调试车模。通过这次比赛，我们不仅得到了软硬件结合的锻炼机会，更多的是提高了自己的创新能力，相信这样的经历会对我们以后的工作和人生带来很大的帮助。

6.2 参数统计

表 6.1 车模技术参数统计

项目	参数
车模几何尺寸（长、宽、高）（毫米）	300　　170　　200
车模轴距/轮距（毫米）	230　　140　　140
电路电容总量（微法）	1116.323554
传感器种类及个数	编码器 1 个　CCD 摄像头 1 个
新增加伺服电机个数	0
赛道信息检测空间精度（毫米）	25
赛道信息检测频率（次/秒）	50
主要集成电路种类/数量	MC9S12XS128MAL、TPS7350、LM2577-ADJ、MIC5219-3.3、TLC5510、LM1881 各 1 片 LT3489*1、IR2104S*2、IRLR7843*4 片
车模重量（带有电池）（千克）	1.1

致谢

在为本次大赛制作智能车期间，我们遇到过很多问题，从最初的传感器选型与方案确定，到后来的软硬件联合调试。在解决一个个问题之后，我们发现，我们的技术在不断的成长，思想在不断的成熟。而在这过程中，离不开学校，老师和同学的支持。

首先，我们要感谢学校对这次比赛的重视，感谢学校教务处对我们比赛的大力支持。没有他们的支持，我们的车模绝对上不了赛场。

其次，我们要感谢高明煜教授和陈龙老师的悉心教导。没有他们在思想上的指导和整体的规划安排，我们很难有今天的成果。

最后，我们要感谢同实验室的其他同学，感谢我们同时工作在这么和谐的实验室。感谢他们在赛道制作和其它方面的帮助。

参考文献

[1]学做智能车 —— 挑战 “ 飞思卡尔 ” 杯 卓晴，黄开胜，邵贝贝，，北京：北京航空航天大学出版社， 2007

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

[3]Freescale 9S12 十六位单片机原理及嵌入式开发技术 孙同景主编 机械工业出版社 2008

[4]PID 调节规律和过程控制 仇慎谦编 江苏科学技术出版社 1987.12

[5]潘松，黄继业 现代数字电路基础教程 北京-科学出版社 2008

[6]Protel DXP 电路设计与制版实用教程 .2 版 李小坚 [等] 编著 人民邮电出版社 2009

[7]新型 PID 控制及其应用（第二版） 陶永乐编著 . ：机械工业出版社， 2002

[8]HCS12 微控制器原理及应用 王威 著 北京 - 北京航空航天大学出版社 2007

[9]电机自动控制系统 李宁，刘启新 著 北京 - 机械工业出版社， 2003

[10]谭浩强 C++程序设计 北京-清华大学出版社 2004

[11]赵先奎 汽车前轮定位 黑龙江 黑龙江出入境检验检疫局 2008

[12]科大中冶队第四届“飞思卡尔”杯全国大学生智能车大赛技术报告 北京科技大学 2009

附录程序源代码

```
#include<hidef.h>/* common defines and macros */
#include"derivative.h"/* derivative-specific definitions */
#include"MC9S12XS128.h"
#include"PLL.h"
#include"ECT.h"
#include"PWM.h"
#include"sci.h"
#include"5110.h"
#include"Key_Driver8.h"
#include"FLASH_K9F1G08.h"
#include"AD.H"

//******************宏定义*****************
#define ROW                         70
#define BOUND                       140
#define COLUMN                      200
#define IMAGE_CENTER                100
#define RollAhead                   PWMDTY4
#define RollBack                    PWMDTY5
#define SevorPwm                    PWMDTY67
//================================================

unsignedint     SevorCenter = 3830;
unsignedchar    LHelmAverageRoute = 20;//前十行黑线平均值
unsignedchar    RHelmAverageRoute = 180;//前十行黑线平均值
unsignedchar    HelmAverageRoute = 100;
//unsigned char    HelmPostion = 29;          //舵机打角参考行
unsignedchar    Debug = 10;
unsignedchar    HelmBlackRoute = 100;//舵机打角行的道路位置
unsignedchar    HelmD = 0 ;//舵机角度值D项
unsignedchar    LastLine = 0 ;//有效行数
unsignedchar    Effective_Forward_Looking = 0;
unsignedchar    HelmPostion = 50;
unsignedchar    SpeedDown = 0;
unsignedchar    SpeedUp     = 0;

//================图像算法拟合====================
```

```
//*****************采集信息*********************
unsignedchar    Hsync=0;              //行同步标志位
///***************
unsignedchar    *P_Pixels;                //图像指针
unsignedchar    *P_Pixel;
unsignedchar    Pixels[6002];             //图像数组缓存29行一场分三次
unsignedint     RowCount;                 //行计数
unsignedint     *P_ChooseRow;                 //选择采集指针
unsignedint     ChooseRow[70]=                //选择采集行
{
        64, 70, 76, 82, 88, 94,100,105,110,115,
       120,125,129,133,136,139,142,145,148,151,
       154,157,160,163,166,169,172,175,178,180,
       182,184,186,188,190,192,194,196,198,200,
       202,204,206,208,210,212,214,216,218,220,
       222,224,226,228,230,232,234,236,238,240,
       242,244,246,248,250,252,254,256,258,260
};
//================================================

//***************处理后道路信息************
unsignedchar LThreshold;
unsignedchar RThreshold;//阈值
unsignedchar Path_Information[650];       //前面70存储黑线中心，后70行存对应的行数，其余存相应的有用信息
unsignedchar LBenchmark = 0;                //Record the effective line
unsignedchar RBenchmark = 0;
unsignedchar Benchmark    = 0;
//*************************
//**********电机控制**********
//unsigned char    Effective_Forward_Looking ;//有效前瞻（记录一场中能处理出的黑线的行数）电机控制用
unsignedint     Max_Speed=135;
unsignedint     Min_Speed=135;
unsignedchar    Speed_Top_Row = 65;
unsignedchar    Speed_Bot_Row = 40;
unsignedchar    Speed_value[30];

//************flash存储指针***************
unsignedchar *P_WriteData ;
```

```
unsignedchar DebugImage=0;           //0:  不采集      1:采集          2:只采有效信息
//********************舵机打脚参数**************
unsignedint    A = 30;                        // 24                                    28
34                  28
unsignedint    B = 40;                                                  //38                  32
32
unsignedint    C = 20;
//unsigned int    D_xishu = 13;                                          //                  15
//unsigned char Cha_S = 14;
//**************(PID/1000)**************************

void delay_DJ(int deley_m)
{
int i,j;
for(i=0;i<deley_m;i++)
for(j=0;j<1000;j++);
}



//************************图像存到FLASH*****************************/
/*  卡1  卡2轮流存10行          */
/* 21*100=2112       */
/*****************************************************************/
void save_flash(void)
{
unsignedint flash_i=0 ;   //循环变量
staticunsignedint w_address2=0 ;   //写卡2地址
staticunsignedint w_address1=0 ;   //写卡1地址

staticunsignedchar first_time = 1;
staticunsignedchar time_2 = 1;
staticunsignedchar time_3 = 1;
staticunsignedchar time_4 = 1;

if(time_4<71)
      {

if(time_2==1)
                {
```

```
                P_WriteData = Pixels;
            }

//将图像存入flash//
if(first_time==1) //写flash1  卡1第一次进入
                {
                    Write_Command1(K9F1G_PAGE_PROGRAM1);
                    Write_Address1(0);
                    Write_Address1(0);

                    Write_Address1(w_address1&0xff);
                    Write_Address1((w_address1>>8)&0xff);            //!!!!!!!!
                    Write_Page1(P_WriteData,COLUMN);
                    P_WriteData = P_WriteData+COLUMN;
                }
else
                {
if(first_time<=10)
                    {
for(flash_i=100;flash_i>0;flash_i--)
                        {
                            WE_PORT1=0 ;
                            IO_PORT_OUT1 = *P_WriteData++;
                            WE_PORT1=1 ;

                            WE_PORT1=0 ;
                            IO_PORT_OUT1 =*P_WriteData++;
                            WE_PORT1=1 ;
                        }
if(first_time==10)              //    22行开始存卡2
                        {
                            Write_Command1(K9F1G_PAGE_PROGRAM2);   //
对第二页编程（打开第二页）
                            w_address1++ ;      //地址++  指向下一页
                        }
                    }

                }
if(first_time==11) //写flash2   22行开始
                {
```

```
                    Write_Command2(K9F1G_PAGE_PROGRAM1) ;
                    Write_Address2(0) ;
                    Write_Address2(0) ;

                    Write_Address2(w_address2&0xff) ;
                    Write_Address2((w_address2>>8)&0xff) ;
                    Write_Page2(P_WriteData,COLUMN) ;
                    P_WriteData = P_WriteData+COLUMN ;
                }
else
                {
if(first_time<=20&&first_time>11)
                    {
for(flash_i=100;flash_i>0;flash_i--)
                        {
                            WE_PORT2=0 ;
                            IO_PORT_OUT2 =*P_WriteData++;//上升沿写数据
                            WE_PORT2=1 ;

                            WE_PORT2=0 ;
                            IO_PORT_OUT2 =*P_WriteData++;//上升沿写数据
                            WE_PORT2=1 ;
                        }
//while(RB_PORT2==0);
if(first_time==20)
                        {
                            Write_Command2(K9F1G_PAGE_PROGRAM2) ;
//对第二页编程（打开第二页）
                            w_address2++ ;

                        }
                    }

                }

                first_time ++ ; //记录进入的次数

                time_2 ++;
                time_3 ++;
if(first_time==21)
```

```
                {
                    first_time=1;
                }
if(time_3==31||time_3==61||time_3==71)              //30行刷新
                {
                    time_2=1;
                }
if(time_3==71)              //第70行一幅结束
                {
                    time_3= 1;
                    first_time =1;
                }
                time_4 ++;
    }
else
    {
                Write_Command2(K9F1G_PAGE_PROGRAM1) ;
                 Write_Address2(0) ;
                 Write_Address2(0) ;

                 Write_Address2(w_address2&0xff) ;
                 Write_Address2((w_address2>>8)&0xff) ;
                 Write_Page2(Path_Information,630) ;
                 Write_Command2(K9F1G_PAGE_PROGRAM2) ;     //对第二
页编程（打开第二页）
                  w_address2++ ;

                  time_4 =1;
    }
}

//************************************
/*********************擦除所有的块************************/
void Erase_All_Block(void)
{
unsignedint i=0 ;
unsignedint j=0 ;
unsignedchar erase_result ;
//擦第一块
    Write_Command1(K9F1G_RESET) ;
```

```
//擦除块 ,同时将坏块的位置发上来
for(i=0;i<1024;i++)
      {
            erase_result=Erase_Block1(j);
while(RB_PORT1==0) ; //等待RB_PORT为1，标志就绪
if(erase_result==1)
            {
//Sci_Num(i,5) ;
// Sci_puts("\n");
            }
            j=j+64;     //64页
      }
//擦完第一块亮以下
// PORTK_PK1=0 ;

while(RB_PORT1==0) ; //等待RB_PORT为1，标志就绪
    LCD_write_char(9,0,'A');
//擦第二块
      Write_Command2(K9F1G_RESET) ;
//擦除块 ,同时将坏块的位置发上来
for(i=0;i<1024;i++)
      {
            erase_result=Erase_Block2(j);
while(RB_PORT2==0) ; //等待RB_PORT为1，标志就绪
if(erase_result==1)
            {
               LCD_write_shu(4,1,i/1000) ;
               LCD_write_shu(5,1,i%1000/100) ;
               LCD_write_shu(6,1,i%1000%100/10) ;
               LCD_write_shu(7,1,i%10) ;
//Sci_Num(i,5) ;
//    Sci_puts("\n");
            }
            j=j+64;
      }
//擦完第一块灭掉以下
//PORTK_PK1=1 ;
//    Send_data(100);
while(RB_PORT2==0) ; //等待RB_PORT为1，标志就绪
      LCD_write_char(10,0,'B');
```

```
}


//*************************************************************/
/*******************读出两块flash**********************
* 函数：Read_Flash_Image();        功能：读一场图像
*全局变量：无
*********************************************************/
void Read_Flash_Image(void)
{
//原始图像
staticunsignedint rd_address1 = 0 ;
staticunsignedint rd_address2= 0;
unsignedint i=64 ;
unsignedchar j=64;

//读原始图像

if(DebugImage==1)
            {
for(j=0;j<4;j++)                                        //一次2副
                  {
                        Write_Command1(K9F1G_PGAE_READ1) ;
                        Write_Address1(0) ;
                        Write_Address1(0) ;
                        Write_Address1(rd_address1&0xff);
                        Write_Address1((rd_address1>>8)&0xff);
                        Write_Command1(K9F1G_PAGE_READ2);
while(RB_PORT1==0); //等待RB_PORT为1，标志就绪
//等待25us
                        Read_Page1(Pixels,2000);

for(i=0;i<2000;i++)
                        {
                              Sci_Tx(Pixels[i] ) ;
//    Send_data(Pixels[i] );

                        }
                        rd_address1++ ;
                        Write_Command2(K9F1G_PGAE_READ1);
```

```
                        Write_Address2(0);
                        Write_Address2(0);
                        Write_Address2((rd_address2)&0xff) ;
                        Write_Address2(((rd_address2)>>8)&0xff) ;
                        Write_Command2(K9F1G_PAGE_READ2) ;
while(RB_PORT2==0) ; //等待RB_PORT为1，标志就绪
//等待25us
                        Read_Page2(Pixels,2000) ;

if(j<3)
                        {

for(i=0;i<2000;i++)
                            {
                                Sci_Tx(Pixels[i] ) ;
//Send_data(Pixels[i] ) ;
                            }
                        }
else
                        {
for(i=0;i<630;i++)
                            {
                                Sci_Tx(Pixels[i] ) ;
//Send_data(Pixels[i] ) ;
                            }
                        }
                       rd_address2++ ;
                    }
                }
if(DebugImage==2)
                {
                    Write_Command2(K9F1G_PGAE_READ1) ;
                    Write_Address2(0) ;
                    Write_Address2(0) ;
                    Write_Address2(rd_address1&0xff);
                    Write_Address2((rd_address1>>8)&0xff);
                    Write_Command2(K9F1G_PAGE_READ2);
while(RB_PORT2==0); //等待RB_PORT为1，标志就绪
//等待25us
                    Read_Page2(Pixels,200);
```

```
for(i=0;i<200;i++)
                        {
                                Sci_Tx(Pixels[i] ) ;
//Send_data(Pixels[i] );

                        }
                        rd_address1++ ;
                }


}


//*************************************************
void Get_Helm_Postion(void)
{

}
//#################################################################
//     函数: Get_First_threshold();
//     功能    首场动态区阈值
//
//
//#################################################################*/
void Get_First_threshold()
{
unsignedchar i;
unsignedchar j;
unsignedchar a;
unsignedint    threshold_sunmin=0;
unsignedint    threshold_sunmax=0;
unsignedchar temp;//冒泡法辅助
unsignedchar tempPixels[COLUMN];//存储第一行数据

while(!Hsync);

for (a=0; a<COLUMN-1; a++)
    {
        tempPixels[a] = Pixels[a];
```

```
    }
//冒泡法排序
for (i=0; i<COLUMN-1; i++)
    {
for (j=0; j<COLUMN-i-1; j++)
        {
if (tempPixels[j] > tempPixels[j+1])
            {
                temp = tempPixels[j];
                tempPixels[j] = tempPixels[j+1];
                tempPixels[j+1] = temp;
            }
        }
    }

for (i=0; i<8; i++)
    {
        threshold_sunmin = threshold_sunmin + tempPixels[i];
    }
for (j=0; j<8; j++)
    {
        threshold_sunmax = threshold_sunmax + tempPixels[COLUMN-j-1];
    }

    LThreshold = (threshold_sunmax + threshold_sunmin) >> 4;
    RThreshold = LThreshold;

}

/*********************************************************************

*********************************************************************
函数：medain_filter
中值滤波        2012-1-13
肖遥

*********************************************************************

*********************************************************************/
char medain_filter(char *a,char *b,char *c)
```

```
{
if (((*a - *b) ^ (*a - *c)) >> 7)
    {
return *a;
    }
if (((*b - *a) ^ (*b - *c)) >> 7)
    {
return *b;
    }
if (((*c - *a) ^ (*c - *b)) >> 7)
    {
return *c;
    }
}
//*********************************************************************
*
// 函数：search_center_go
// 搜索中心点作为搜索范围中心                2010-3-14    wzy
//
//*********************************************************************
*/
void search_center_go(void)
{
unsignedchar image_row = 0;//    Record The first Pixels ROW; A ROW    has 210
COLUMN

unsignedchar black_left_column = 0;// Record The starting point of the black line
unsignedchar black_right_column = 0;//      Record The end point of the black line

unsignedchar left_side = 0;
unsignedchar right_side = 0;

unsignedchar ten_black_l = 0;
unsignedchar ten_black_r = 0;

unsignedchar fieldDisposed = ROW;       //Field finished Signal
unsignedint    black_bound_sum = 0;     //Record    black line center
unsignedint    line_feed = 0;

unsignedchar row_left_variable = 0;//Loop variables is used to search the starting
```

```
point
unsignedchar row_right_variable = 0 ;//Loop variables is used to search the end point
unsignedchar threshold1_variable = 0;//Loop variables
unsignedchar row_i_variable = 0;       // Loop variables

unsignedchar turn_l_flag = 0;
unsignedchar turn_r_flag = 0;

unsignedchar tend_D = 0;

unsignedchar l_row_stop = 0;
unsignedchar r_row_stop = 0;
unsignedchar l_lost_cnt = 0;
unsignedchar r_lost_cnt = 0;

//==================赛道拟合参数====================
unsignedchar left_line = 0;
unsignedchar right_line = 0;
unsignedchar row_white_center = 100;
staticunsignedchar mid_value[140] = {0};

//==================阈值计算参数====================
unsignedint    threshold_b_w_sum = 0;
unsignedint    threshold_sum_l = 0;
unsignedint    threshold_sum_r = 0;
unsignedchar threshold_black = 0;
unsignedchar threshold_white = 0;
unsignedchar threshold_count_l = 0;
unsignedchar threshold_count_r = 0;
unsignedchar threshold = 0;

unsignedchar l_far_threshold = 120;
unsignedchar r_far_threshold = 120;

unsignedint    reflesh_i=0;
//==================搜索范围参数====================
staticunsignedchar range_ten_l = 100;
staticunsignedchar range_ten_r = 100;
unsignedchar range_l = 100;
unsignedchar range_r = 100;
```

```
unsignedchar search_range = 0;
//======================确定下一场参数====================
int rake = 0;
//======================十字处理参数======================
//    static unsigned char lost_flag = 0;
//    unsigned char cross_flag = 0;
//---------------清Path_Information数组，待存路径-----------
for (reflesh_i=0; reflesh_i<560; reflesh_i++)
    {
        Path_Information[reflesh_i] = 0;
    }
    LBenchmark = 0;
    RBenchmark = 0;
    Benchmark = 0;
    LThreshold = 25;
    RThreshold = 25;
//-----------------------------------------图像处理
-----------------------------------------------------
while(fieldDisposed)
    {
//==================Search for the number of black segments ,Record
information==================
while(!Hsync);          //行同步信号

for (image_row=0; image_row<30&&fieldDisposed>40; image_row++)
        {
            black_left_column = 0;
            black_right_column = 0;

            threshold_b_w_sum = 0;

            left_side = 0;
            right_side = 0;

            turn_l_flag = 0;
            turn_r_flag = 0;

for (row_left_variable=range_ten_l; row_left_variable>4; row_left_variable--)
            {
if (*(P_Pixel+line_feed+row_left_variable+1) >
```

```
*(P_Pixel+line_feed+row_left_variable))
                {
if
((*(P_Pixel+line_feed+row_left_variable+1)-*(P_Pixel+line_feed+row_left_variable)
>LThreshold)
&&(*(P_Pixel+line_feed+row_left_variable+2)-*(P_Pixel+line_feed+row_left_varia
ble-1)>LThreshold)
&&(*(P_Pixel+line_feed+row_left_variable+2)-*(P_Pixel+line_feed+row_left_varia
ble-2)>LThreshold))
                 {
                     left_side = 1;

break;
                 }
                }

             }
             black_left_column = row_left_variable;

for (row_right_variable=range_ten_r; row_right_variable<COLUMN-5;
row_right_variable++)
             {
if (*(P_Pixel+line_feed+row_right_variable-1) >
*(P_Pixel+line_feed+row_right_variable))
                {
if
((*(P_Pixel+line_feed+row_right_variable-1)-*(P_Pixel+line_feed+row_right_variab
le)>RThreshold)
&&(*(P_Pixel+line_feed+row_right_variable-2)-*(P_Pixel+line_feed+row_right_var
iable+1)>RThreshold)
&&(*(P_Pixel+line_feed+row_right_variable-2)-*(P_Pixel+line_feed+row_right_var
iable+2)>RThreshold))
                 {
                     right_side = 1;

break;
                 }
                }

             }
```

```
            black_right_column = row_right_variable;


/*      if (image_row == 0)
            {
                mid_value[image_row] = HelmBlackRoute;     //先给中点赋值，保证
数组中有30个值
            }
            else
            {
                mid_value[image_row] = mid_value[image_row-1];     //先给中点赋
值，保证数组中有30个值
            }

            if ((0 == left_side) && (0 == right_side) && (mid_value[image_row] <
130) && (mid_value[image_row] > 70))
            {
                lost_flag++;
                if (lost_flag > 3)
                {
                    cross_flag = 1;
                }
            }
            else
            {
                lost_flag = 0;
            }    */

if (1 == left_side)
            {
                Path_Information[LBenchmark] = black_left_column;
                Path_Information[LBenchmark+BOUND] = image_row + 1;
                Path_Information[LBenchmark+BOUND+BOUND] = range_ten_l;
                Path_Information[LBenchmark+BOUND+BOUND+BOUND] =
LThreshold;

/*      if (black_left_column > 6)
                {
                  if (Pixels[line_feed+black_left_column+image_row/10-6] <
LThreshold)
```

```
                    {
                      for (threshold1_variable=black_left_column;
                            threshold1_variable>black_left_column-6+image_row/10;
                            threshold1_variable--)
                      {
                        threshold_b_w_sum = threshold_b_w_sum +
Pixels[line_feed+threshold1_variable];
                      }
                      threshold_black = threshold_b_w_sum/(6-image_row/10);
                      threshold_b_w_sum = 0;

                      if ((black_left_column+10<COLUMN-10) &&
(Pixels[line_feed+black_left_column+10]>LThreshold))
                      {
                          for (threshold1_variable=black_left_column+2;
                                threshold1_variable<black_left_column+10;
                                threshold1_variable++)
                          {
                            threshold_b_w_sum = threshold_b_w_sum +
Pixels[line_feed+threshold1_variable];
                          }
                          threshold_white = threshold_b_w_sum >> 3;

                          if (threshold_white > threshold_black + 20)          //判断阈
值正确性
                          {
                              threshold = ((threshold_black+threshold_white)>>1);
//求出下一行黑白阈值的平均值
                              threshold_count_l++ ;
//记录进行阈值计算的个数
                              threshold_sum_l = threshold_sum_l + threshold ;
//阈值累加
                          }
                      }
                    }      */
if ((Path_Information[LBenchmark] >= LHelmAverageRoute)
&& (Path_Information[LBenchmark] - LHelmAverageRoute < 75))
                    {
                        LBenchmark = LBenchmark + 1; //记录基准黑线的个数
                        turn_l_flag = 1;
```

```
                    }
else
                    {
if ((LHelmAverageRoute > Path_Information[LBenchmark])
&& (LHelmAverageRoute - Path_Information[LBenchmark] < 75))
                        {
                            LBenchmark = LBenchmark + 1 ; //记录基准黑线的个数
                            turn_l_flag = 1;
                        }
                    }
            }



if (1 == right_side)
            {
                Path_Information[RBenchmark+ROW] = black_right_column;
                Path_Information[RBenchmark+ROW+BOUND] = image_row + 1;
                Path_Information[RBenchmark+ROW+BOUND+BOUND] =
range_ten_r;
                Path_Information[RBenchmark+ROW+BOUND+BOUND+BOUND]
= RThreshold;

/*      if (black_right_column < COLUMN - 7)
                {
                   if (Pixels[line_feed+row_right_variable-image_row/10+6] <
RThreshold)
                    {
                       for (threshold1_variable=black_right_column;

threshold1_variable<black_right_column-image_row/10+6;
                                threshold1_variable++)
                       {
                          threshold_b_w_sum = threshold_b_w_sum +
Pixels[line_feed+threshold1_variable];
                       }
                       threshold_black = threshold_b_w_sum/(6-image_row/10);
                       threshold_b_w_sum = 0;

                       if ((black_right_column - 10 > 10) &&
```

```
(Pixels[black_right_column-10] > RThreshold))
                        {
                            for (threshold1_variable=black_right_column-10;
                                 threshold1_variable<black_right_column-2;
                                 threshold1_variable++)
                            {
                               threshold_b_w_sum = threshold_b_w_sum +
Pixels[line_feed+threshold1_variable];
                            }
                            threshold_white = threshold_b_w_sum>>3;

                            if (threshold_white > threshold_black + 20)          //判断阈
值正确性
                            {
                                threshold = ((threshold_black+threshold_white)>>1);
//求出下一行黑白阈值的平均值
                                threshold_count_r++ ;
//记录进行阈值计算的个数
                                threshold_sum_r = threshold_sum_r + threshold ;
//阈值累加
                            }
                        }
                    }   */

if (Path_Information[RBenchmark+ROW] >= RHelmAverageRoute
&& Path_Information[RBenchmark+ROW] - RHelmAverageRoute < 75)
                    {
                        RBenchmark = RBenchmark + 1; //记录基准黑线的个数
                        turn_r_flag = 1;
                    }
else
                    {
if (RHelmAverageRoute>Path_Information[RBenchmark+ROW]
&&RHelmAverageRoute-Path_Information[RBenchmark+ROW] < 75)
                        {
                            RBenchmark = RBenchmark + 1 ; //记录基准黑线的个
数
                            turn_r_flag = 1;
                        }
                    }
```

```
            }


if (LBenchmark > 1 && 1 == turn_l_flag)
            {
if (Path_Information[LBenchmark-2] < Path_Information[LBenchmark-1])
                {
                    search_range =
(Path_Information[LBenchmark-1]-Path_Information[LBenchmark-2])/

(Path_Information[LBenchmark-1+BOUND]-Path_Information[LBenchmark-2+BO
UND]);
if (LBenchmark > 4)
                    {
if ((range_ten_l + search_range) > mid_value[Benchmark-1])
                            {
                                range_ten_l = mid_value[Benchmark-1];
                            }
else
                            {
                                range_ten_l = range_ten_l + search_range ;
                            }
                    }

                }
else
                {
                    search_range =
(Path_Information[LBenchmark-2]-Path_Information[LBenchmark-1])/

(Path_Information[LBenchmark-1+BOUND]-Path_Information[LBenchmark-2+BO
UND]);

if (LBenchmark > 4)
                    {
if ((range_ten_l - search_range) < 12)
                            {
                                range_ten_l = 12 ;
                            }
else
```

```
                        {
                            range_ten_l = range_ten_l - search_range ;
                        }
                    }

                }

            }

if (RBenchmark > 1 && 1 == turn_r_flag)
            {
if (Path_Information[RBenchmark+ROW-2] <
Path_Information[RBenchmark+ROW-1])
                {
                    search_range =
(Path_Information[RBenchmark+ROW-1]-Path_Information[RBenchmark+ROW-2])
/

(Path_Information[RBenchmark+ROW+BOUND-1]-Path_Information[RBenchmark
+ROW+BOUND-2]);

if (RBenchmark > 4)
                    {
if ((range_ten_r + search_range) >= COLUMN - 12)
                        {
                            range_ten_r = COLUMN - 12 ;
                        }
else
                        {
                            range_ten_r = range_ten_r + search_range ;
                        }
                    }


                }
else
                {
                    search_range =
(Path_Information[RBenchmark+ROW-2]-Path_Information[RBenchmark+ROW-1])
/
```

```
(Path_Information[RBenchmark+ROW+BOUND-1]-Path_Information[RBenchmark
+ROW+BOUND-2]);

if (RBenchmark > 4)
                    {
if ((range_ten_r - search_range) < mid_value[Benchmark-1])
                        {
                            range_ten_r = mid_value[Benchmark-1] ;
                        }
else
                        {
                            range_ten_r = range_ten_r - search_range ;
                        }
                    }

                }

            }

if ((1 == turn_l_flag) || (1 == turn_r_flag))
            {
if ((1 == left_side) && (1 == right_side) && (black_right_column -
black_left_column) > 60)
                {
                    row_white_center =    (black_right_column + black_left_column)
>> 1;
                }

elseif ((1 == left_side) && (0 == right_side))
                {
if (black_left_column    + (80 - image_row) > COLUMN - 2)
                    {
                        row_white_center = COLUMN - 2;
                    }
else
                    {
                        row_white_center = black_left_column    + (80 -
image_row) ;
                    }
```

```
                }

elseif ((0 == left_side) && (1 == right_side))
                {
if (black_right_column < (80 - image_row) + 3)
                  {
                        row_white_center = 3;
                  }
else
                  {
                        row_white_center = black_right_column - (80 - image_row);
                  }
                }

                mid_value[Benchmark] = row_white_center;
                Path_Information[Benchmark+BOUND+BOUND+BOUND+BOUND]
= row_white_center;
                mid_value[Benchmark+ROW] = image_row + 1;

if (mid_value[Benchmark] >= HelmAverageRoute && mid_value[Benchmark] -
HelmAverageRoute < 60)
                {
                     Benchmark =Benchmark + 1;
                }
elseif (HelmAverageRoute >    mid_value[Benchmark] && HelmAverageRoute-
mid_value[Benchmark] < 60)
                {
                     Benchmark = Benchmark + 1 ;
                }

            }

            line_feed = line_feed + COLUMN ; //处理完一行指向下一行
if (29 == image_row)
            {
              line_feed = 0;
            }

            fieldDisposed = fieldDisposed - 1;//每处理完一行数据就减1，当70行数据
减到60时说明所有采集的图象已经处理完
```

```
            Hsync = Hsync - 1;//行同步

if(DebugImage == 1)
            {
                save_flash();
            }

while(!Hsync);
        }


if (Benchmark>=4)
        {
            black_bound_sum = 0;

for(row_i_variable=0;row_i_variable<Benchmark;row_i_variable++)
            {
                black_bound_sum = black_bound_sum+ mid_value[row_i_variable] ;
            }

            HelmAverageRoute = black_bound_sum/Benchmark;
        }

if (LBenchmark >= 5)
        {
            black_bound_sum = 0;

for (row_i_variable=LBenchmark-4; row_i_variable<LBenchmark;
row_i_variable++)
            {
                black_bound_sum = black_bound_sum +
Path_Information[row_i_variable];
            }

            LHelmAverageRoute =    black_bound_sum >> 2;

/*    if (threshold_count_l >= 3)
            {
                LThreshold = threshold_sum_l / threshold_count_l;
            }       */
```

```
/****************搜索域的确定******************/
            rake = (int)LHelmAverageRoute - Path_Information[1] ;
//HelmAverageRoute

if (rake < 0)
            {
                rake = 0 - rake ;
            }

if ((Path_Information[1] + 10 + rake) > mid_value[Benchmark-1])
            {
                range_ten_l = mid_value[Benchmark-1] ;
            }
else
            {
                range_ten_l = Path_Information[1] + 10 + rake ;
            }
        }

if (RBenchmark >= 5)
        {
            black_bound_sum = 0;

for (row_i_variable=RBenchmark-4; row_i_variable<RBenchmark;
row_i_variable++)
            {
                black_bound_sum = black_bound_sum +
Path_Information[row_i_variable+ROW];
            }

            RHelmAverageRoute =    black_bound_sum >> 2;

/*    if (threshold_count_r >= 3)
            {
                RThreshold = threshold_sum_r / threshold_count_r;
            }         */

/****************搜索域的确定******************/
            rake = (int)RHelmAverageRoute - Path_Information[ROW+1] ;
```

```
//HelmAverageRoute

if (rake < 0)
            {
                rake = 0 - rake ;
            }

if((Path_Information[ROW+1] - 10 - rake) < mid_value[Benchmark-1])
            {
                range_ten_r = mid_value[Benchmark-1] ;
            }
else
            {
                range_ten_r = Path_Information[ROW+1] - 10 - rake ;
            }
        }


if (40 == fieldDisposed)
        {
if (Benchmark > 5)
            {
                range_l = mid_value[Benchmark-1];
                range_r = mid_value[Benchmark-1];
            }
else
            {
                range_l = 100;
                range_r = 100;
            }

            l_row_stop = 0;
            r_row_stop = 0;

            ten_black_l = LBenchmark ;
            ten_black_r = RBenchmark ;

for (image_row=30; image_row<70&&fieldDisposed>0; image_row++)
            {
                black_left_column = 0;
```

```
                black_right_column = 0;

                threshold_b_w_sum = 0;

                left_side = 0;
                right_side = 0;

                turn_l_flag = 0;
                turn_r_flag = 0;

switch(image_row/10)
                {
case 3: l_far_threshold = 20;
                        r_far_threshold = 20;
break;

case 4: l_far_threshold = 20;
                        r_far_threshold = 20;
break;

case 5: l_far_threshold = 10;
                        r_far_threshold = 10;
break;

case 6: l_far_threshold = 10;
                        r_far_threshold = 10;
break;
                }

/*      if (3 == image_row/10)
                {
                        point = 2;
                }
                else
                {
                        point = 1;
                }

                switch (image_row/10)
                {
```

```
                case 3: point = 2 ;

                case 4: point = 2 ;

                case 5: point = 1 ;

                case 6: point = 1 ;

                default : point = 1 ;
                }

                point = (69 - image_row)/10;
                if (point < 1)
                {
                      point = 1;
                }     */

for (row_left_variable=range_l; row_left_variable>3&&0==l_row_stop;
row_left_variable--)
                {
if (*(P_Pixel+line_feed+row_left_variable+1) >
*(P_Pixel+line_feed+row_left_variable)
&& *(P_Pixel+line_feed+row_left_variable+2) >
*(P_Pixel+line_feed+row_left_variable))
                    {
if
((*(P_Pixel+line_feed+row_left_variable+2)-*(P_Pixel+line_feed+row_left_variable)
>l_far_threshold)
&&(*(P_Pixel+line_feed+row_left_variable+2)-*(P_Pixel+line_feed+row_left_varia
ble-1)>l_far_threshold))
                     {
                         left_side = 1;

break;
                     }
                    }

                }
                black_left_column = row_left_variable;
```

```
for (row_right_variable=range_r;
row_right_variable<COLUMN-4&&0==r_row_stop; row_right_variable++)
                {
if (*(P_Pixel+line_feed+row_right_variable-1) >
*(P_Pixel+line_feed+row_right_variable)
&& *(P_Pixel+line_feed+row_right_variable-2) >
*(P_Pixel+line_feed+row_right_variable))
                  {
if
((*(P_Pixel+line_feed+row_right_variable-2)-*(P_Pixel+line_feed+row_right_variab
le)>r_far_threshold)
&&(*(P_Pixel+line_feed+row_right_variable-2)-*(P_Pixel+line_feed+row_right_var
iable+1)>r_far_threshold))
                   {
                       right_side = 1;

break;
                   }
                  }

                }
                black_right_column = row_right_variable;


if (1 == left_side)
                {
                   Path_Information[LBenchmark] = black_left_column;
                   Path_Information[LBenchmark+BOUND] = image_row + 1;
                   Path_Information[LBenchmark+BOUND+BOUND] = range_l;
                   Path_Information[LBenchmark+BOUND+BOUND+BOUND] =
l_far_threshold;

if ((black_left_column > Path_Information[LBenchmark-1] && black_left_column -
Path_Information[LBenchmark-1] < 8
                       || (ten_black_l == LBenchmark && (int)black_left_column -
Path_Information[LBenchmark-1] < 42))
                       || (black_left_column <= Path_Information[LBenchmark-1]
&& Path_Information[LBenchmark-1] - black_left_column < 8))
                {
/*    for (threshold1_variable=black_left_column;
```

```
                        threshold1_variable>black_left_column-(point+1);
                        threshold1_variable--)
                        {
                            threshold_b_w_sum = threshold_b_w_sum +
Pixels[line_feed+threshold1_variable];
                        }
                        threshold_black = threshold_b_w_sum/(point+1);
                        threshold_b_w_sum = 0;

                    if ((black_left_column+6<COLUMN-2) &&
(Pixels[line_feed+black_left_column+6]>l_far_threshold))
                    {
                        for (threshold1_variable=black_left_column+2;
                             threshold1_variable<black_left_column+6;
                             threshold1_variable++)
                        {
                          threshold_b_w_sum = threshold_b_w_sum +
Pixels[line_feed+threshold1_variable];
                        }
                        threshold_white = threshold_b_w_sum >> 2;

                        if ((threshold_white > threshold_black + 15)
                            || ((threshold_white > threshold_black +
8)&&(image_row > 50)))          //判断阈值正确性
                        {
                            l_far_threshold = ((threshold_black +
threshold_white)>>1);   //求出下一行黑白阈值的平均值

                            if (l_far_threshold < 90 + image_row)
                            {
                                l_far_threshold = 90 + image_row;
                            }
                        }
                    }    */
                    LBenchmark = LBenchmark + 1;
                    turn_l_flag = 1;
                    l_lost_cnt = 0 ;

if (LBenchmark > ten_black_l + 3)
                    {
```

```
if (Path_Information[LBenchmark-2] > Path_Information[LBenchmark-3])
                        {
if (black_left_column < Path_Information[LBenchmark-2]
&& Path_Information[LBenchmark-2] - black_left_column > 3)
                            {
                              LBenchmark = LBenchmark - 1;
                              turn_l_flag = 0;
                            }
                        }
else
                        {
if (black_left_column > Path_Information[LBenchmark-2]
&& black_left_column - Path_Information[LBenchmark-2] > 3)
                            {
                              LBenchmark = LBenchmark - 1;
                              turn_l_flag = 0;
                            }
                        }
                    }
                }

            }

if (1 == right_side)
            {
              Path_Information[RBenchmark+ROW] = black_right_column;
              Path_Information[RBenchmark+ROW+BOUND] = image_row + 1;
              Path_Information[RBenchmark+ROW+BOUND+BOUND] =
range_r;
              Path_Information[RBenchmark+ROW+BOUND+BOUND+BOUND]
= r_far_threshold;

if ((black_right_column >= Path_Information[RBenchmark+ROW-1] &&
black_right_column - Path_Information[RBenchmark+ROW-1] < 8)
                    || (black_right_column <
Path_Information[RBenchmark+ROW-1]
&& Path_Information[RBenchmark+ROW-1] - black_right_column < 8
                        || (ten_black_r == RBenchmark &&
(int)Path_Information[RBenchmark+ROW-1] - black_right_column < 42)))
              {
```

```
/*for (threshold1_variable=black_right_column;
                        threshold1_variable<black_right_column+(point+1);
                        threshold1_variable++)
                        {
                            threshold_b_w_sum = threshold_b_w_sum +
Pixels[line_feed+threshold1_variable];
                        }
                        threshold_black = threshold_b_w_sum/(point+1);
                        threshold_b_w_sum = 0;

                    if ((black_right_column-6>2) &&
(Pixels[line_feed+black_right_column-6]>r_far_threshold))
                    {
                        for (threshold1_variable=black_right_column-2;
                              threshold1_variable>black_right_column-6;
                              threshold1_variable--)
                        {
                          threshold_b_w_sum = threshold_b_w_sum +
Pixels[line_feed+threshold1_variable];
                        }
                        threshold_white = threshold_b_w_sum >> 2;

                        if (threshold_white > threshold_black + 15
                              || (threshold_white > threshold_black + 8)&&(image_row
> 50))          //判断阈值正确性
                        {
                            r_far_threshold = ((threshold_black + threshold_white)
>> 1);   //求出下一行黑白阈值的平均值

                            if (r_far_threshold < 100 + image_row)
                            {
                                r_far_threshold = 100 + image_row;
                            }
                        }
                    }   */
                    RBenchmark = RBenchmark + 1;
                    turn_r_flag = 1;
                    r_lost_cnt = 0;

if (RBenchmark > ten_black_r + 3)
```

```
                    {
if (Path_Information[RBenchmark+ROW-2] >
Path_Information[RBenchmark+ROW-3])
                        {
if (black_right_column < Path_Information[RBenchmark+ROW-2]
&& Path_Information[RBenchmark+ROW-2] - black_right_column > 3)
                            {
                                RBenchmark = RBenchmark - 1;
                                turn_r_flag = 0;
                            }
                        }
else
                        {
if (black_right_column >Path_Information[RBenchmark+ROW-2]
&& black_right_column    - Path_Information[RBenchmark+ROW-2] > 3)
                            {
                                RBenchmark = RBenchmark - 1;
                                turn_r_flag = 0;
                            }
                        }
                    }
                }

            }

if (0 == turn_l_flag)
            {
                l_lost_cnt = l_lost_cnt + 1;

if (l_lost_cnt > 2 && LBenchmark - ten_black_l > 2)
                {
                    l_row_stop = 1;
                }
            }

if (0 == turn_r_flag)
            {
                r_lost_cnt = r_lost_cnt + 1;

if (r_lost_cnt > 2 && RBenchmark - ten_black_r > 2)
```

```
                    {
                        r_row_stop = 1;
                    }
                }

if ((1 == turn_l_flag) || (1 == turn_r_flag))
                {
if ((1 == left_side) && (1 == right_side) && (black_right_column -
black_left_column) > 100 - image_row )
                    {
                       row_white_center = (black_left_column + black_right_column)
>> 1;
                    }
elseif ((1 == left_side) && (0 == right_side))
                    {
if (black_left_column    + (85 - image_row) > COLUMN - 2)
                        {
                            row_white_center = COLUMN - 2;
                        }
else
                        {
                            row_white_center = black_left_column    + (85 -
image_row);
                        }

/*if (Path_Information[LBenchmark-2] < Path_Information[LBenchmark-1])
                        {
                           search_range =
(Path_Information[LBenchmark-1]-Path_Information[LBenchmark-2])/

(Path_Information[LBenchmark-1+BOUND]-Path_Information[LBenchmark-2+BO
UND]);

                           if (mid_value[Benchmark-1] + search_range - 1 > COLUMN -
2)
                           {
                              row_white_center = COLUMN - 2;
                           }
                           else
                           {
```

```
                        row_white_center = mid_value[Benchmark-1] +
search_range - 1 ;
                    }

                }
                else
                {
                    search_range =
(Path_Information[LBenchmark-2]-Path_Information[LBenchmark-1])/

(Path_Information[LBenchmark-1+BOUND]-Path_Information[LBenchmark-2+BO
UND]);

                    if (mid_value[Benchmark-1] - search_range + 1 < 2)
                    {
                        row_white_center = 2;
                    }
                    else
                    {
                        row_white_center = mid_value[Benchmark-1] -
search_range;
                    }

                } */

            }

elseif ((0 == left_side) && (1 == right_side))
                {
if (black_right_column < (85 - image_row) + 3)
                    {
                        row_white_center = 3;
                    }
else
                    {
                        row_white_center = black_right_column - (85 - image_row);
                    }
/* if (Path_Information[RBenchmark+ROW-2] <
Path_Information[RBenchmark+ROW-1])
                    {
```

```
                search_range =
(Path_Information[RBenchmark+ROW-1]-Path_Information[RBenchmark+ROW-2])
/

(Path_Information[RBenchmark+ROW+BOUND-1]-Path_Information[RBenchmark
+ROW+BOUND-2]);

                if (mid_value[Benchmark-1] + search_range - 1 > COLUMN -
2)
                {
                   row_white_center = COLUMN - 2;
                }
                else
                {
                   row_white_center = mid_value[Benchmark-1] +
search_range;
                }

             }
             else
             {
                search_range =
(Path_Information[RBenchmark+ROW-2]-Path_Information[RBenchmark+ROW-1])
/

(Path_Information[RBenchmark+ROW+BOUND-1]-Path_Information[RBenchmark
+ROW+BOUND-2]);

                if (mid_value[Benchmark-1] - search_range + 1 < 2)
                {
                   row_white_center = 2;
                }
                else
                {
                   row_white_center = mid_value[Benchmark-1] -
search_range + 1;
                }

             } */
```

```
                    }


if (row_white_center > mid_value[Benchmark-1] && row_white_center -
mid_value[Benchmark-1] < 12
                            || 30 == image_row)
                    {
                        mid_value[Benchmark] = row_white_center;
                        mid_value[Benchmark+ROW] = image_row + 1;

Path_Information[Benchmark+BOUND+BOUND+BOUND+BOUND] =
row_white_center;

                        Benchmark = Benchmark + 1;
                    }
elseif (row_white_center <= mid_value[Benchmark-1] && mid_value[Benchmark-1]
- row_white_center < 12
                                || 30 == image_row)
                    {
                        mid_value[Benchmark] = row_white_center;
                        mid_value[Benchmark+ROW] = image_row + 1;


Path_Information[Benchmark+BOUND+BOUND+BOUND+BOUND] =
row_white_center;

                        Benchmark = Benchmark + 1;
                    }

                }

if (LBenchmark > ten_black_l)
                {
                    range_l = mid_value[Benchmark-1];
if (range_l < 8)
                    {
                        range_l = 8;
                    }
                }
```

```
if (RBenchmark > ten_black_r)
                {
                    range_r = mid_value[Benchmark-1];
if (range_r > COLUMN - 8)
                    {
                        range_r = COLUMN - 8;
                    }
                }

/******************十字加速判断******************/
if (LBenchmark > ten_black_l + 8 && RBenchmark > ten_black_r + 8 && SpeedUp
== 0
&& Path_Information[ten_black_r+ROW+BOUND] -
Path_Information[ten_black_r-1+ROW+BOUND] > 20
&& Path_Information[ten_black_l+BOUND] -
Path_Information[ten_black_l-1+BOUND] > 20)
                {
                    SpeedUp = 85;
                }


                line_feed = line_feed + COLUMN ;
if (59 == image_row)
                {
                  line_feed = 0;
                }

                fieldDisposed = fieldDisposed - 1;
                Hsync = Hsync - 1;//行同步

if(DebugImage == 1)
                {
                    save_flash();
                }

if (image_row < 69)
                {
while(!Hsync);
                }
```

```
            } // 30-70  行处理停止
        }
    }

if(DebugImage == 1)
    {
        save_flash();
    }

// LastLine = mid_value[Benchmark-1+ROW] - 1;
      Effective_Forward_Looking = mid_value[Benchmark-1+ROW]; //记录有效行

if (Effective_Forward_Looking > 64)
      {
        HelmPostion = 4*Effective_Forward_Looking/5;    //4/5            9/10
      }
elseif (Effective_Forward_Looking > 30)
      {
        HelmPostion = 41*Effective_Forward_Looking/35 -
Effective_Forward_Looking*Effective_Forward_Looking/175;    //6/5和1/150
15/14和1/350
      }
else
      {
        HelmPostion = Effective_Forward_Looking;
      }

if (Benchmark >= 9 && mid_value[ROW+Benchmark-1] > 15)
      {
unsignedchar i;

for (i=0; i<Benchmark; i++)
            {
if (mid_value[ROW+i] >= HelmPostion - 1)
                {
if (mid_value[ROW+i] == HelmPostion - 1)
                    {
                        HelmBlackRoute = mid_value[i] ;
                    }
else
```

```
                    {
                        HelmBlackRoute = mid_value[i-1] + (mid_value[ROW+i] -
HelmPostion+1) * (mid_value[i] - mid_value[i-1])/(mid_value[i+ROW] -
mid_value[i-1+ROW]);
                    }
break;
                }
            }
if (i == Benchmark)
            {
                HelmBlackRoute = mid_value[Benchmark-1] ;//+
(HelmPostion-mid_value[ROW+i-1]+1)* (mid_value[i-1] -
mid_value[i-2])/(mid_value[i-1+ROW] - mid_value[i-2+ROW]);
            }

for (i=0; i<Benchmark; i++)
            {
if( mid_value[ROW+i] >= 30)              //38      34
                {
                    HelmD = mid_value[i];
break;
                }
            }
if (i==Benchmark)
            {
                HelmD = mid_value[Benchmark-1];
            }

      }



/*     Path_Information[631] = Effective_Forward_Looking;
       Path_Information[632] = HelmPostion;
       Path_Information[633] = HelmBlackRoute;


    Write_LCD(1,2, Path_Information[Debug+ROW]);
    Write_LCD(1,1,Effective_Forward_Looking);
    Write_LCD(1,4, Path_Information[Debug+BOUND+BOUND+BOUND]);
```

```
    Write_LCD(1,5, Path_Information[Debug+ROW+BOUND+BOUND+BOUND]);
*/

}


//*********************************************************************
******/
//函数:Helm_Control();
//功能:舵机打角控制
/*********************************************************************/
void Helm_Control(void)
{
unsignedint sevorA = 0;
unsignedint sevorB = 0;
unsignedint sevorC = 0;
unsignedint pwm_temp = 0;
unsignedchar st_flag    = 0;
staticunsignedchar stc_flag      = 0;
staticunsignedchar last_Helmd = 0;

if (Effective_Forward_Looking >= 65)
      {
            sevorA = A + 3;
            sevorB = B - 3;
            sevorC = C ;
      }
elseif (Effective_Forward_Looking >= 60)
      {
            sevorA = A;
            sevorB = B + 8;
            sevorC = C + 5;
      }
else
      {
            sevorA = A - 2;
            sevorB = B ;
            sevorC = C + 2;
      }
```

```
if (Effective_Forward_Looking >= 69)
        {
            stc_flag = stc_flag + 1;
if (stc_flag > 2)
            {
                st_flag = 1;
            }
        }
else
        {
if (stc_flag > 8)
            {
                SpeedDown = 4;
            }
            stc_flag = 0;
        }

if (1 == st_flag)
        {
//      sevorA = A + 3;
            sevorB = 5;
            sevorC = 5;
        }
/******左边******/
if (HelmBlackRoute > IMAGE_CENTER)
        {
if (HelmD > last_Helmd)
            {
                pwm_temp =
SevorCenter+(sevorA)*(((HelmBlackRoute-IMAGE_CENTER)*(HelmBlackRoute-I
MAGE_CENTER))/500)+(sevorB*(HelmBlackRoute-IMAGE_CENTER))/10+sevor
C*(HelmD-last_Helmd);
            }
else
            {
                pwm_temp =
SevorCenter+(sevorA)*(((HelmBlackRoute-IMAGE_CENTER)*(HelmBlackRoute-I
MAGE_CENTER))/500)+(sevorB*(HelmBlackRoute-IMAGE_CENTER))/10;
            }
        }
```

```
/******中间******/
elseif (HelmBlackRoute == IMAGE_CENTER)
//Helm_Black_Route>=100&&Helm_Black_Route<=110
      {
          pwm_temp = SevorCenter;
      }
/******右边******/
else
      {
if (HelmD < last_Helmd)
          {
              pwm_temp =
SevorCenter-(sevorA)*(((IMAGE_CENTER-HelmBlackRoute)*(IMAGE_CENTER
-HelmBlackRoute))/500)-(sevorB*(IMAGE_CENTER-HelmBlackRoute))/10-sevorC
*(last_Helmd-HelmD) ;
          }
else
          {
              pwm_temp =
SevorCenter-(sevorA)*(((IMAGE_CENTER-HelmBlackRoute)*(IMAGE_CENTER
-HelmBlackRoute))/500)-(sevorB*(IMAGE_CENTER-HelmBlackRoute))/10 ;
          }
      }
if (pwm_temp > SevorCenter + 500)
      {
          pwm_temp = SevorCenter + 500;
      }
elseif (pwm_temp < SevorCenter - 500)
      {
          pwm_temp = SevorCenter - 500;
      }

      SevorPwm = pwm_temp;
      last_Helmd = HelmD;

// Path_Information[634] = (unsigned char) (SevorPwm);
// Path_Information[635] = (unsigned char) (SevorPwm>>8);
}
/*
    unsigned int pwm_temp = 0;
```

```
unsigned int sevorA = 0;
unsigned int sevorB = 0;
unsigned int sevorD = 0;
static unsigned char last_helm_d = IMAGE_CENTER ;//上次图像的D项

        if(Effective_Forward_Looking>=66)            //66
        {
            if(PointSign>0)
            {
                sevorA = A-10;     //0
                sevorB = B-18;
                if(MyCnt<=1&&(DebugImage!=5))
                {
                    //sevorD = D_xishu - 8;
                }
                else
                {
                    sevorD = 0;
                }
                 //PTM_PTM0 = 1;
            }
            else
            {
                sevorA = A+6;
                sevorB = B-8;
                sevorD = D_xishu-2;
                 //PTM_PTM0 = 0;
            }
        }
       else
       {
            //PTM_PTM0 = 0;
           if(Effective_Forward_Looking>=49)
           {
               sevorA = A;
               sevorB = B;
               sevorD = D_xishu;
              if(Effective_Forward_Looking>55)
               {
                   if(PointSign>0)
```

```
                        {
                            if(DebugImage!=5)
                            {
                                sevorA = A+4; //-2
                                sevorB = B-10;
                                sevorD =    D_xishu;//-4
                            }
                            else
                            {
                                 sevorA = A-2; //-2
                                sevorB = B-14;
                                sevorD =    D_xishu-4;//-4
                            }
                             //PTM_PTM0 = 1;
                        }
                        else
                        {
                            sevorA = A;
                            sevorB = B;
                        }
                    }
                }
                else
                {
                    sevorA = A;
                    sevorB = B;
                    sevorD = D_xishu;
                }
            }
            if(HelmBlackRoute>IMAGE_CENTER)
            {
                if(HelmD>last_helm_d)
                {
                    pwm_temp = SevorCenter+sevorA*(((HelmBlackRoute-
IMAGE_CENTER)*(HelmBlackRoute-
IMAGE_CENTER))/1000)+(sevorB*(HelmBlackRoute-
IMAGE_CENTER))/10+sevorD*(HelmD-last_helm_d) ;
                }
                else
                {
```

```
                    pwm_temp = SevorCenter+sevorA*(((HelmBlackRoute-
IMAGE_CENTER)*(HelmBlackRoute-
IMAGE_CENTER))/1000)+(sevorB*(HelmBlackRoute- IMAGE_CENTER))/10;
                }

                if(pwm_temp>=SevorCenter+450)
                {
                    PWMDTY01 = SevorCenter+450;
                }
                else
                {
                      PWMDTY01 = pwm_temp;
                }
            }
            //中间
            if(HelmBlackRoute==IMAGE_CENTER)
            {
                PWMDTY01 = SevorCenter;
            }
            //右打
            if(HelmBlackRoute < IMAGE_CENTER)
            {
                  if(HelmD>last_helm_d)
                 {
                      pwm_temp =
SevorCenter-(sevorA)*((( IMAGE_CENTER-HelmBlackRoute)*( IMAGE_CENTE
R-HelmBlackRoute))/1000)-((sevorB)*( IMAGE_CENTER-HelmBlackRoute))/10 ;
                 }
                 else
                  {
                       pwm_temp =
SevorCenter-(sevorA)*((( IMAGE_CENTER-HelmBlackRoute)*( IMAGE_CENTE
R-HelmBlackRoute))/1000)-((sevorB)*( IMAGE_CENTER-HelmBlackRoute))/10-s
evorD*(last_helm_d-HelmD);
                  }
                if(pwm_temp<=SevorCenter-450)
                {
                    PWMDTY01=SevorCenter-450;
                }
                else
```

```
                {
                    PWMDTY01 = pwm_temp;
                }moto

            }
            last_helm_d = HelmD ;


            //存储舵机值转换
            Path_Information[444] =    (unsigned char) (PWMDTY01);
            Path_Information[445] =    (unsigned char) (PWMDTY01>>8);
            //存储HelmD值
             Path_Information[443]= HelmD ;
}*/
void Init_Speed_value(void)
{
int i=0;
for(i=0;i<=Speed_Top_Row - Speed_Bot_Row;i++)
      {
            Speed_value[i] = Max_Speed - (int)((long)i*(long)i*(long)(Max_Speed -
Min_Speed)/((long)(Speed_Top_Row - Speed_Bot_Row)*(long)(Speed_Top_Row -
Speed_Bot_Row)));
      }

}


void MotoControl(void)
{
unsignedint getspeed = 0;

if(Effective_Forward_Looking >= Speed_Top_Row)
      {
            getspeed = Max_Speed;
      }
elseif(Effective_Forward_Looking > Speed_Bot_Row)
      {
            getspeed = Speed_value[Speed_Top_Row - Effective_Forward_Looking];
      }
else
```

```
        {
            getspeed = Min_Speed;
        }

if (SpeedDown != 0)
        {
            SpeedDown = SpeedDown - 1;

            getspeed = 0;
        }

if (SpeedUp != 0)
        {
            SpeedUp = SpeedUp - 1;

if (SpeedUp < 65)
            {
                getspeed = Max_Speed + 10;
            }
        }

        RollAhead = getspeed;
}
/*void Page_One(void)
{
   LCD_write_char(1,0,'S');//C
   LCD_write_char(2,0,'-');//Y
   LCD_write_char(3,0,'P');//-
   LCD_write_char(4,0,':');
   LCD_write_char(5,0,DebugImage);

   LCD_write_char(1,1,'H');
   LCD_write_char(2,1,'-');    //最高速
   LCD_write_char(3,1,'S');
   LCD_write_char(4,1,':');
   Write_LCD(5,1,Max_Speed);

   LCD_write_char(1,2,'L');
   LCD_write_char(2,2,'-');    //最低速
   LCD_write_char(3,2,'S');
```

```
    LCD_write_char(4,2,':');
    Write_LCD(5,2,Min_Speed);


    LCD_write_char(1,3,'H');
    LCD_write_char(2,3,'-');   //  直道入弯道速度
    LCD_write_char(3,3,'-');
    LCD_write_char(4,3,':');
    Write_LCD(5,3,H);

    LCD_write_char(1,4,'S');
    LCD_write_char(2,4,'-');   //打角行微调
    LCD_write_char(3,4,'C');
    LCD_write_char(4,4,':');
    write_four(5,4,SevorCenter);

    LCD_write_char(1,5,'R');
    LCD_write_char(2,5,'E');   //打角行微调
    LCD_write_char(3,5,'A');
    LCD_write_char(4,5,'D');
    LCD_write_char(5,5,'Y');
}

void Page_Two(void)
{
    LCD_write_char(1,0,'A');//C
    LCD_write_char(2,0,':');
    Write_LCD(3,0,A);

    LCD_write_char(1,1,'B');
    LCD_write_char(2,1,':');
    Write_LCD(3,1,B);

    LCD_write_char(1,2,'D');
    LCD_write_char(2,2,':');
    Write_LCD(3,2,D_xishu);

    LCD_write_char(1,3,'S');
    LCD_write_char(2,3,':');
    Write_LCD(3,3,Scratch_Time);
```

```
    LCD_write_char(1,5,'R');
    LCD_write_char(2,5,'E');    //打角行微调
    LCD_write_char(3,5,'A');
    LCD_write_char(4,5,'D');
    LCD_write_char(5,5,'Y');
}
void Sys_Set(void)
{
    unsigned char set_Y=0;
    unsigned char last_set_Y=0;
    unsigned char key=0;
    unsigned char run=0;
    unsigned char    page = 1;

    Page_One() ;
    LCD_write_char(0,set_Y,43);

    for(;run==0;)
    {
     key=KeyScan();
       if(key==4)
       {
          last_set_Y=set_Y;
          set_Y++;
          if(set_Y==6)
          {
                    LCD_clear();
                    if(page==1)
                    {
                        Page_Two() ;
                        page = 2;
                    }
                    else
                    {
                            Page_One() ;
                            page = 1;

                    }
```

```
                set_Y = 0;
    }


    LCD_clear_byte(0,last_set_Y);
     LCD_write_char(0,set_Y,43);
     delay_DJ(5000);
}
 if(key==3)
{
    last_set_Y=set_Y;

    if(set_Y==0)
     {
              LCD_clear();
                if(page==1)
             {
                 Page_Two() ;
                 page = 2;
             }
             else
             {
                     Page_One() ;
                     page = 1;
             }
             set_Y =    6;
     }
    set_Y--;
    LCD_clear_byte(0,last_set_Y);
    LCD_write_char(0,set_Y,43);
    delay_DJ(5000);
}

if(1==key)
{
        if(page==1)
        {
              switch(set_Y)
              {
                      case 0: DebugImage++;
```

```
                    if(DebugImage>5)
                            DebugImage=0 ;
                        if(DebugImage==5)
                        {
                                Scratch_Time = 1000;
                        }
                    LCD_write_char(5,0,DebugImage);break;
                  case 1: Max_Speed+=5;
                    Write_LCD(5,1,Max_Speed);break;
                    case 2: Min_Speed+=5;
                    Write_LCD(5,2,Min_Speed);break;
                    case 3: H +=1;
                    Write_LCD(5,3,H);break;
                    case 4: SevorCenter +=1;
                            PWMDTY01 = SevorCenter;
                      write_four(5,4,SevorCenter);break;
                    case 5: run=1;
             }
        }
        else
        {
            switch(set_Y)
            {
                    case 0: A++;
                    Write_LCD(3,0,A);break;
                  case 1: B++;
                    Write_LCD(3,1,B);break;
                    case 2: D_xishu++;
                    Write_LCD(3,2,D_xishu);break;
                    case 3:    Scratch_Time+=10;
                    Write_LCD(3,3,Scratch_Time);break;
                    case 4:
                    case 5: run=1;
             }
        }
         delay_DJ(5000);


}
```

```
   if(2==key)
{
      if(page==1)
      {

            switch(set_Y)
            {
             case 0:
                  if(DebugImage==0)
                  {
                        DebugImage=5 ;
                        Scratch_Time = 1000;
                  }
                  else
                  {
                        DebugImage--;
                  }
            LCD_write_char(5,0,DebugImage);break;
             case 1:Max_Speed-= 5;
               Write_LCD(5,1,Max_Speed);break;
             case 2: Min_Speed-= 5;
               Write_LCD(5,2,Min_Speed);break;
             case 3: H -= 1;
               Write_LCD(5,3,H);break;
             case 4: SevorCenter-= 1;
             PWMDTY01 = SevorCenter;
               write_four(5,4,SevorCenter);break;

             case 5: run=1;
            }
      }
      else
      {
            switch(set_Y)
            {
                  case 0: A--;
                     Write_LCD(3,0,A);break;
                  case 1: B--;
                    Write_LCD(3,1,B);break;
                  case 2: D_xishu--;
```

```
                    Write_LCD(3,2,D_xishu);break;
                case 3:    Scratch_Time-=10;
                    Write_LCD(3,3,Scratch_Time);break;
                case 4:
                case 5: run=1;
                }

            }
                delay_DJ(5000);
        }
    }
    LCD_write_char(1,5,'R');
    LCD_write_char(2,5,'U');
    LCD_write_char(3,5,'N');
    LCD_write_char(4,5,'I');
    LCD_write_char(5,5,'N');
    LCD_write_char(6,5,'G');
}               */

void PageOne(void)
{
        LCD_write_char(1,0,'C');       //采集图像信息
        LCD_write_char(2,0,'P');
        LCD_write_char(3,0,':');
        Write_LCD(4,0,Debug);
//Write_LCD2(0,8,PACNT);

        LCD_write_char(1,1,'C');       //采集有效数据信息
        LCD_write_char(2,1,'I');
        LCD_write_char(3,1,':');
        Write_LCD(4,1,DebugImage);
//Write_LCD2(1,4,ScratchTime);
//ScratchTime += 50;

        LCD_write_char(1,2,'H');
        LCD_write_char(2,2,':');
//Write_LCD2(2,3,DebugImageTwo);
        write_four(3,2,SevorCenter);

        LCD_write_char(1,3,'S');
```

```
        LCD_write_char(2,3,'D');
        Write_LCD(3,3,Max_Speed);

        LCD_write_char(1,4,'S');
        LCD_write_char(2,4,'X');
        Write_LCD(3,4,Min_Speed);


        LCD_write_char(1,5,'S');
        LCD_write_char(2,5,'T');
        LCD_write_char(3,5,'A');
        LCD_write_char(4,5,'R');
        LCD_write_char(5,5,'T');
}
/*********************************************************************
***/
//----------------------液晶第二页参数设置---------------------------//
/*********************************************************************
***/
void PageTwo(void)
{
        LCD_write_char(1,0,'A');
        LCD_write_char(2,0,':');
        Write_LCD(3,0,A);

        LCD_write_char(1,1,'B');
        LCD_write_char(2,1,':');
        Write_LCD(3,1,B);

        LCD_write_char(1,2,'D');
        LCD_write_char(2,2,':');
        Write_LCD(3,2,C);

        LCD_write_char(1,3,'S');
        LCD_write_char(2,3,'T');
        Write_LCD(3,3,Speed_Top_Row);

        LCD_write_char(1,4,'S');
        LCD_write_char(2,4,'B');
        Write_LCD(3,4,Speed_Bot_Row);
```

```
        LCD_write_char(1,5,'S');
        LCD_write_char(2,5,'C');
        write_four(3,5,Speed_Bot_Row);

        LCD_write_char(8,5,'R');
        LCD_write_char(9,5,'P');
//Write_LCD(5,10,rampspeed);

}

void PageView(void)
{
unsignedchar cnt = 0;
unsignedchar drop = 1;

      SevorPwm = SevorCenter;

while(drop)
      {
if(cnt <= 5)
        {
          PageOne();
          LCD_write_char(0,cnt,43);
        }
else
        {
          PageTwo();
          LCD_write_char(0,cnt - 6,43);
        }



if(KeyScan() == 1)
     {
         if(cnt > 0)
              cnt -= 1;
         else
              cnt = 0;
         while(KeyScan() == 1);
```

```
        LCD_clear();
    }


    if(KeyScan() == 2)
    {
        if(cnt > 12)
            cnt = 5;
        else
            cnt += 1;
        while(KeyScan() == 2);
        LCD_clear();
    }


    if(KeyScan() == 3)
    {
switch(cnt)
            {
case 0:
                                    Debug += 3;
break;//DebugImageOne = 1;break;

case 1:
                                    DebugImage += 1;//ScratchTime += 50;//
break;

case 2:
                                     SevorCenter += 1;
                                     SevorPwm = SevorCenter;
break;

case 3:
                                     Max_Speed += 5;
break;

case 4:
                                      Min_Speed += 5;
break;
```

```
case 5:
                                        LCD_clear();drop = 0;
break;

case 6:
                                        A += 1;
break;

case 7:
                                        B += 1;
break;

case 8:
                                        C += 1;
break;

case 9:
                                        Speed_Top_Row += 1;
break;

case 10:
                                        Speed_Bot_Row += 1;
break;

case 11:
                                        Speed_Bot_Row += 25;
break;
case 12:
//rampspeed       +=5;
break;

                }
while(KeyScan() == 3);
    }
    if(KeyScan() ==4)
    {
switch(cnt)
                {
case 0:
                                    Debug -= 3;
```

```
break;//DebugImageOne = 0;break;

case 1:
if((DebugImage - 1) < 0)
                                    {
                                        DebugImage = 6;
                                    }
else
                                    {
                                        DebugImage -= 1;
                                    }/**/
//ScratchTime -= 50;
break;

case 2:
                                    SevorCenter -= 1;
                                    SevorPwm = SevorCenter;
break;

case 3:
                                    Max_Speed -= 5;
break;

case 4:
                                    Min_Speed -= 5;
break;

case 5:
break;

case 6:
                                    A -= 1;
break;

case 7:
                                    B -= 5;
break;

case 8:
                                    C -= 1;
```

```
break;

case 9:
                                    Speed_Top_Row -= 1;
break;

case 10:
                                    Speed_Bot_Row -= 1;
break;
case 11:
                                    Speed_Bot_Row -= 25;
break;
case 12:
//rampspeed     -=5;
break;

              }
while(KeyScan() == 4);
     }
    }
}
/*****************************************************************
*********************************
      Velocity_feedback( );          当前速度：  speed=   100* now_pcnt/11400
（m/s）
舵机打角行应为speed*0.2（m）最高可达  650  脉冲~=5.7m/s
*****************************************************************
*********************************/
/*void Velocity_feedback(void)
{
   int lightPwm ;
   int rightPwm ;
   DDRA = 0x00;
   lightPwm = PWMDTY4 ;
   rightPwm = PWMDTY3 ;
  //反馈速度
   BackRightSpeed = PORTA;
   PTS_PTS3 =    1;                  //右轮置位

   BackLightSpeed = PACNT; //右轮计数
```

```
    PTS_PTS3 =    0;
    PACNT=0;                                    //左轮清零
    BackNowSpeed = ((BackRightSpeed+BackLightSpeed)>>1);

    if(RowCount==2)
     {
            BackSpeed[0] = BackRightSpeed;
            BackSpeed[2] = BackLightSpeed;
            Measure_Speed_1 = BackNowSpeed;
     }
     else
     {
             BackSpeed[1] = BackRightSpeed;
             BackSpeed[3] = BackLightSpeed;
             Measure_Speed_2 = BackNowSpeed;
     }

if(Motor_Stop==0)
{
                  //PID增量式
             lightPwm    = lightPwm +
speed_PID_calc_light(GetLightSpeed,BackLightSpeed);
             rightPwm = rightPwm +
speed_PID_calc_right(GetRightSpeed,BackRightSpeed);

                if(BackLightSpeed<=(7*GetLightSpeed/10))
              {
                       PWMDTY4 = 255;
              }
             else
             {
                  if(BackLightSpeed>=(13*GetLightSpeed/10))
                  {
                          PWMDTY4 = 0;
                  }
                  else
                  {
                       if(lightPwm>255)
                       {
                              PWMDTY4 = 255;
```

```
                    }
                    else if(lightPwm<0)
                    {
                          PWMDTY4 = 0;
                    }
                    else
                    {
                         PWMDTY4 = lightPwm;
                    }
               }
          }

          if(BackRightSpeed<=(7*GetRightSpeed/10))
           {
                    PWMDTY3 = 255;
           }
           else
           {
               if(BackRightSpeed>=(13*GetRightSpeed/10))
               {
                      PWMDTY3 = 0;
               }
               else
               {
                    if(rightPwm>255)
                    {
                          PWMDTY3 = 255;
                    }
                    else if(rightPwm<0)
                    {
                          PWMDTY3 = 0;
                    }
                    else
                    {
                         PWMDTY3 = rightPwm;
                    }
               }
           }
}
else
```

```
{
        PWMDTY3 = 0;
        PWMDTY4 = 0;
}

}*/
/* void Velocity_feedback(void)
{
   int lightPwm ;
   int rightPwm ;
   static unsigned char aDS = 8;
   static unsigned char motoStop = 0;
   DDRA = 0x00;
   lightPwm = PWMDTY4 - PWMDTY5 ;
   rightPwm = PWMDTY2 -PWMDTY3;
  //反馈速度
   BackRightSpeed = PORTA;
   PTS_PTS3 =    1;                    //右轮置位

   BackLightSpeed = PACNT; //右轮计数
   PTS_PTS3 =    0;
   PACNT=0;                                    //左轮清零
   BackNowSpeed = ((BackRightSpeed+BackLightSpeed)>>1);

   if(RowCount==2)
     {
        Measure_Speed_1 = BackNowSpeed;
        Path_Information[436] = GetLightSpeed;
        Path_Information[437] = GetRightSpeed;
        Path_Information[438] = BackLightSpeed;
        Path_Information[439] = BackRightSpeed;
     }
     else
     {
         Measure_Speed_2 = BackNowSpeed;
         Path_Information[446] = GetLightSpeed;
         Path_Information[447] = GetRightSpeed;
         Path_Information[448] = BackLightSpeed;
         Path_Information[449] = BackRightSpeed;
```

```
    }

if(Effective_Forward_Looking>60)
{
    aDS = 10;
}
else
{
    aDS = 8;
}
if(Motor_Stop>0)
{
                //PID增量式
            lightPwm    = lightPwm +
speed_PID_calc_light(GetLightSpeed,BackLightSpeed);
            rightPwm = rightPwm +
speed_PID_calc_right(GetRightSpeed,BackRightSpeed);

              if(BackLightSpeed<=(aDS*GetLightSpeed/10))
             {
                    PWMDTY4 = 255;
                    PWMDTY5 = 0;
             }
            else
            {
                if(BackLightSpeed>=(13*GetLightSpeed/10))
                {
                    PWMDTY4 = 0;
                    if(lightPwm<0)
                    {
                        if(lightPwm<0-255)
                        {
                            PWMDTY5 = 255;
                        }
                        else
                        {
                            PWMDTY5 = 0 - lightPwm;
                        }
                    }
                    else
```

```
                {
                        PWMDTY5 = 0;
                }
        }
        else
        {
                PWMDTY5 = 0;
                if(lightPwm>255)
                {
                        PWMDTY4 = 255;
                }
                else if(lightPwm<0)
                {
                        PWMDTY4 = 0;
                }
                else
                {
                        PWMDTY4 = lightPwm;
                }
        }
}

if(BackRightSpeed<=(aDS*GetRightSpeed/10))
 {
                PWMDTY2 = 255;
                PWMDTY3 = 0;
 }
 else
 {
        if(BackRightSpeed>=(13*GetRightSpeed/10))
        {
                PWMDTY2 = 0;
                if(rightPwm<0)
                {
                        if(rightPwm<0-255)
                        {
                                PWMDTY3 = 255;
                        }
                        else
                        {
```

```
                                PWMDTY3 = 0 - rightPwm;
                            }
                        }
                        else
                        {
                            PWMDTY3 = 0;
                        }
                    }
                    else
                    {
                        PWMDTY3 = 0;
                        if(rightPwm>255)
                        {
                            PWMDTY2 = 255;
                        }
                        else if(rightPwm<0)
                        {
                            PWMDTY2 = 0;
                        }
                        else
                        {
                            PWMDTY2 = rightPwm;
                        }
                    }
                }

                if(Motor_Stop==2)
                {
                    if(BackNowSpeed<20)
                    {
                        motoStop++;
                        if(motoStop>1)
                        {
                            Motor_Stop = 0;
                        }
                    }
                }
}
else
{
```

```
                if(BackNowSpeed>20)
                    {
                        motoStop = 0;
                        Motor_Stop = 2;
                    }
                    else
                    {
                         PWMDTY3 = 0;
                         PWMDTY2 = 0;
                         PWMDTY5 = 0;
                         PWMDTY4 = 0;
                    }
}
            //防止倒转
}   */
void Init_Port(void)
{
   IRQ_Init();
   Pll_Init();
   Sci_Init();
   LCD_Init();
   LCD_clear();
// PitInit();
   Pwm67_Init();     //舵机PWM
   SevorPwm = SevorCenter;
   Pwm2_Init() ;       //电机正PWM
   Pwm3_Init() ;
   Pwm4_Init() ;
   Pwm5_Init() ;
   Ad_Init() ;
   PageView();
//Sys_Set();
   Pulse_Accumulation_Init();
   DDRM_DDRM0 = 1;
   PTM_PTM0 = 0;

//-------FLASH  调试工具初始化----
//****flash IO口初始化  ***
if(DebugImage==1||DebugImage==2)
    {
```

```
            FLASH_IO_Init1() ;
            FLASH_IO_Init2() ;
            Erase_All_Block();
    }
if(DebugImage==3)
    {
            FLASH_IO_Init1() ;
            FLASH_IO_Init2() ;
            DebugImage=1;
for(;;)
            {
                Read_Flash_Image();
            }
    }
//-------SD卡调试工具初始化------
/* SD_Init();        //目前速度只能达到4k/T   严重滞后。采完整图像是不行的，
不过采有效信息是可以的
   LCD_write_char(7,0,'F');
   while(SD_Reset());        //复位是否成功成功：0      失败：1
   LCD_write_char(7,0,'C');       */
//-------------------------------------
   Ectinput0_Init() ;

}

//*****************************************************************/
//--------------------------主函数-----------------------------/
//*****************************************************************/
void main(void)
{
unsignedint sysTime = 0;
unsignedint time = 0;
unsignedint time1 =0;
   DDRA = 0x00;
   Init_Port();
   Init_Speed_value();
   TFLG1_C0F=1 ;              //清标志位
   TSCR1_TEN=0 ;              //关闭捕捉
   ATD0DIEN = 0xfe;
   DDR1AD0 &= 0x01;//配置为输入，用于外部AD采样
```

```
   DDRS_DDRS3 = 1;
   PTS_PTS3 =    1;
   PACNT=0 ; //脉冲计数清零
   PTS_PTS3 =    0;
for(sysTime=0;;sysTime++)
   {
        RowCount = 0 ;
        P_ChooseRow = ChooseRow;
        P_Pixels = Pixels;
        P_Pixel = Pixels;
        Hsync = 0;
        TSCR1_TEN = 1 ;//开启捕捉
while(TFLG1_C0F==0) ;   //查询ECT0,等待场同步信号的下降延或上升延到来
        EnableInterrupts;

if(sysTime<=5)
        {
// Get_First_threshold();
        }
else
        {
            search_center_go();
            Helm_Control();
if (sysTime <= 505)
            {
                MotoControl();
            }
else
            {
                RollAhead = 0;
            }

        }
while(RowCount<261);
        TFLG1_C0F = 1 ;
        TSCR1_TEN =0 ; //关掉捕
        DisableInterrupts;

//    Write_LCD(9,1,HelmBlackRoute);
```

```
    }



}
//*********************************************************
//行采集中断
//*********************************************************


#pragma CODE_SEG __NEAR_SEG NON_BANKED
interrupt 6 void IRQ_interrupt(void)
{
unsignedchar Point_c=0 ;
      RowCount++ ;        //每进入中断一次，行加1;

if(RowCount==*P_ChooseRow)     //采70行
      {
            P_ChooseRow=P_ChooseRow+1;
for(Point_c=95;Point_c>0;Point_c--)                        //前100点不要
            {
                  asm nop;
            }
for(Point_c=0;Point_c<200;Point_c++)                        //采集100个点
            {
                  *P_Pixels=PT1AD0;
                  P_Pixels++;
                  asm nop;
                  asm nop;
                  asm nop;
                  asm nop;
                  asm nop;
                  asm nop;
                  asm nop;
            }
            Hsync = Hsync + 1;
      }
else
      {
if ((RowCount == 181) || (RowCount == 241))          //30行刷新内存
```

```
            {
                P_Pixels = Pixels;
            }
        }

}

#pragma CODE_SEG DEFAULT
```