第七届“飞思卡尔”杯全国大学生

智能汽车竞赛

技 术 报 告

学校：东南大学

队伍名称：SEU3

参赛队员：赵行晟 吴昌盛 徐乃阳

带队老师：谈英姿 孙琳

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：

赵行晟

徐乃阳

吴昌盛

带队教师签名：谈英姿

日　　期：2012.8.13

目录

摘要：

本文以第七届全国大学生智能车竞赛为背景，使用 Freescale 半导体公司生产的 16 位单片机 MC9S12XS128 和摄像头 OV7620 的配合实现道路识别。数字摄像头采集数据由 MCU 进行采集，并且经过处理和计算得到道路信息，并且根据当前赛道类型和车模所处的位置以及速度计算出控制信息，控制车模以最快最优的路径进行行驶。本文设计的智能车系统是由软硬件和机械结构组成的整体，其硬件主要包括电源模块、电机驱动，速度测量、SD 卡调试、单片机模块等。

关键词：Freescale　MCU 摄像头 机械 最优路径

第一章 引言

1.1 智能车竞赛概况

“飞思卡尔杯”全国大学生智能汽车竞赛是教育部高等学校自动化专业教学指导委员会主办，以“立足培养、重在参与、鼓励探索、追求卓越”为指导思想的多学科专业交叉的创意性科技竞赛。该竞赛是面向全国大学生的探索性工程实践活动，涵盖控制、模式识别、传感技术、电子、电气、计算机、机械等多个学科知识，旨在促进高等学校素质教育，培养大学生的综合知识运用能力、基本工程实践能力和创新意识，激发大学生从事科学研究与探索的兴趣和潜能，倡导理论联系实际、求真务实的学风和团队协作的人文精神。大赛至今已举办四届，已吸引全国 26 个省(自治区)、直辖市的二百余所高校广泛参与。我校作为一所以理工科见长的综合性大学，从首届比赛开始便参与其中，至今已举办四届校内选拔赛，共有千余学生参与其中，从中选拔出的优秀队伍代表学校参加更高级别的比赛。

1.2 智能车研究内容

摄像头组智能车主要包含图像采集处理，道路识别以及基于二维图像的车体运动控制三个领域的内容。其中，图像采集处理主要包括片外采集电路设计及单片机采集时序的控制，道路识别主要是拟合算法及优化，而运动控制包含电机驱动电路及速度和方向的调节算法。

1.3 SEU3 整体设计思路

对于摄像头组，本届比赛相比以往有 2 点显著的变化，一是赛道变为 45 厘米，二是黑线变为在赛道两边。吸取各赛区优秀队伍的经验，我们更改了舵机的安装方式，将车模底盘进行加固，电路布局紧凑，前瞻距离超过 1.3 米，控制算法采用了最小二乘拟合。

1.4 SEU3 技术概括

机械方面，参照后轮主动悬挂的原理，将车模的后轮与车身固定，实现转弯时候的“被动悬挂”，帮助过完。摄像头置于车身靠前部分，并采用碳素杆固定，其高度为 350mm，前方可见视野范围是 100mm~1500mm。

下图为车模的整体外形图：

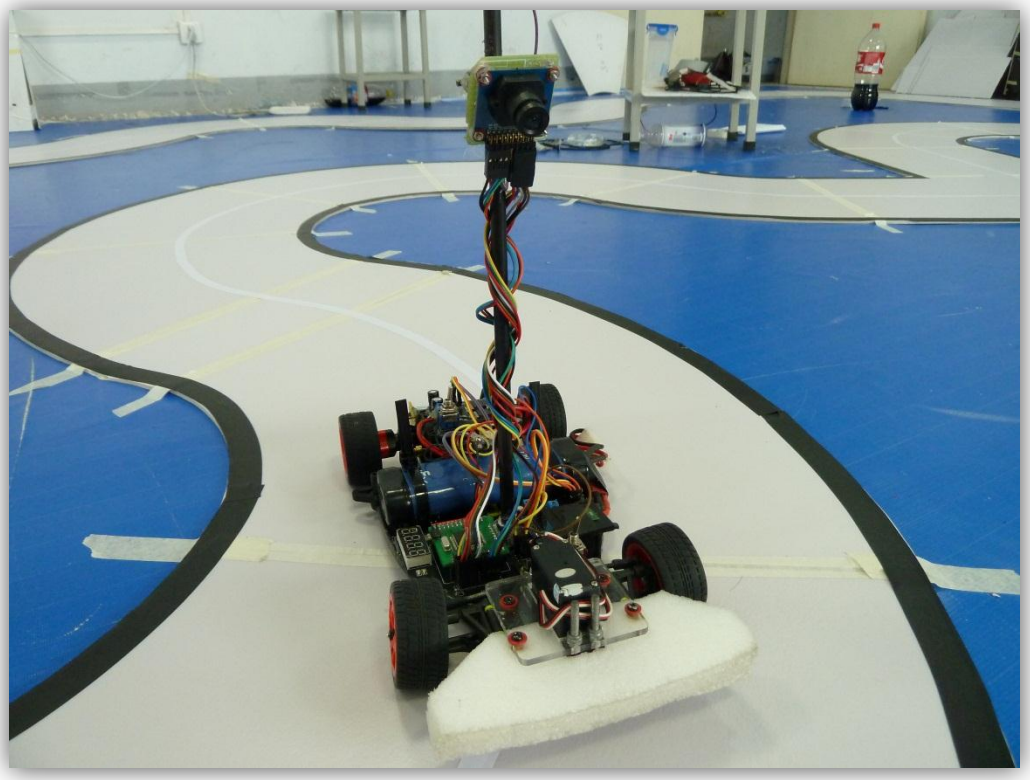


图 1.1 SEU3 外观

硬件电路方面，由于采用摄像头前置安装，前部分空余的空间较少，，在考虑到要尽量减少车体重量和降低整车重心的情况下，本文给车量身设计了一款比较小巧的电路板，如图 1.2 所示。

SEU3 采用数字摄像头 ov7620，因数字摄像头使用方便，减少了外围器件，从而减轻了车的重量。并且数字摄像头的动态性能较好，对于黑色敏感度较高，对于赛道提取有很大帮助。SEU3 还加入了 SD 卡模块，蓝牙无线模块，记录行驶过程中的重要数据，利用 SD 卡进行算法离线调试，无线模块进行在线参数调试，相比利用 BDM 进行在线调试，工作效率显著提高。

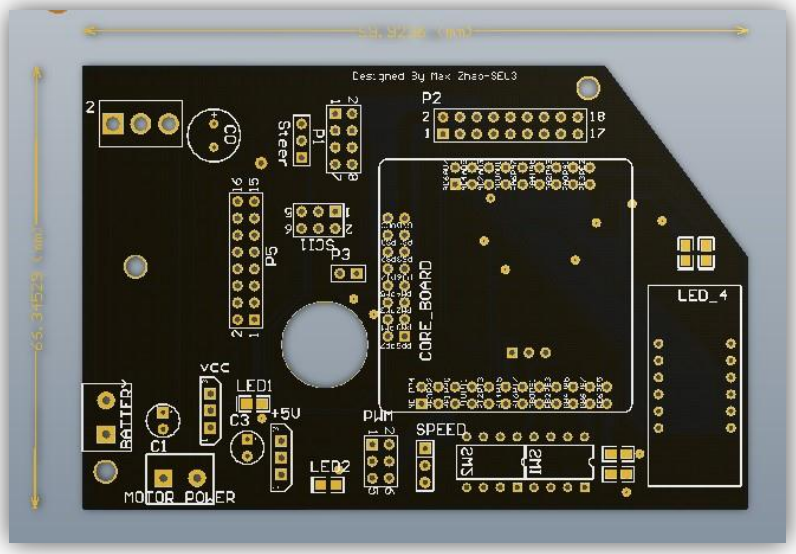


图 1.2 PCB 电路板

软件方面，SEU3 使用开窗边沿检测和阈值检测相结合的方法提取中点信息，利用最小二乘法对中点进行分段线性拟合，软件系统运行流程图如图 1.3。

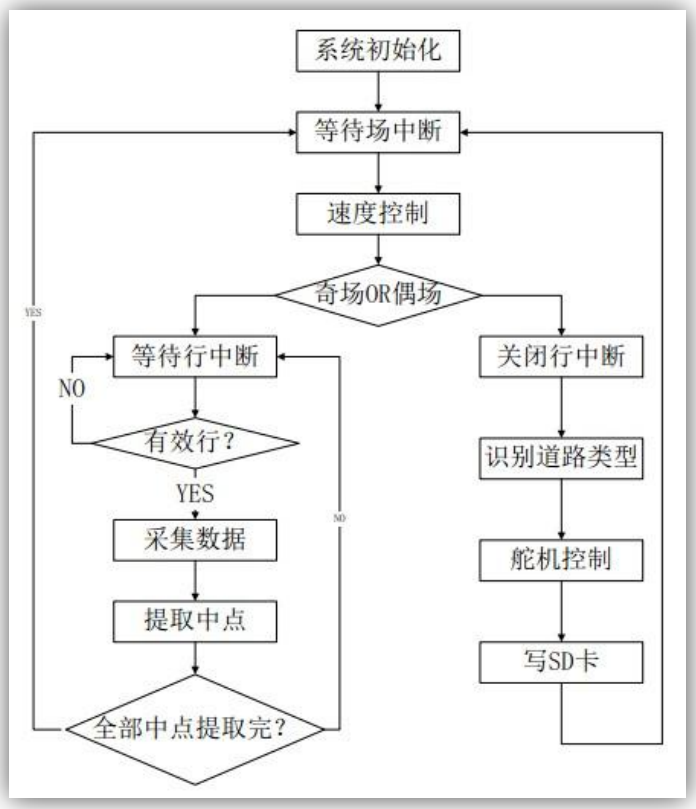


图 1.3 SEU3 算法流程图

第二章 机械设计

2.1 车模机械分析

2.1.1. 底盘结构——差速器

差速器位置应该处于传动轴与左右半轴的交汇点，从变速箱输出的动力在这里被分配到左右两个半轴。汽车在直线行驶时左右两个驱动轮的转速是相同的，但在转弯过时两边车轮行驶的距离不是等长的，因此车轮的转速肯定也会不同。差速器的作用就在于允许左右两边的驱动轮以不同的转速运行。

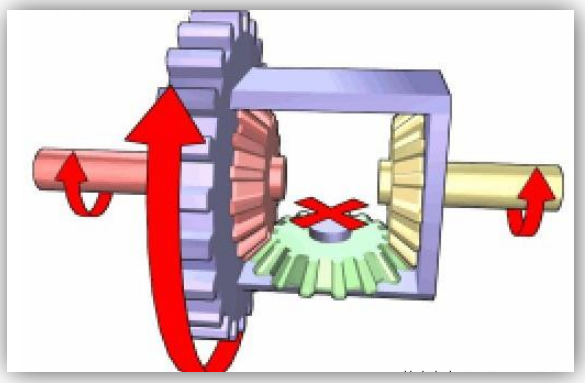


图 2.1 车辆直线行驶差速器状态

直线行驶时的特点是左右两边驱动轮的阻力大致相同。从发动机输出的动力首先传递到差速器壳体上使差速器壳体开始转动。接下来要把动力从壳体传递到左右半轴上，我们可以理解为两边的半轴齿轮互相在“较劲”，由于两边车轮阻力相同，因此二者谁也掰不过对方，因此差速器壳体内的行星齿轮跟着壳体公转同时不会产生自转，两个行星齿轮咬合着两个半轴齿轮以相同的速度转动，这样汽车就可以直线行驶了！

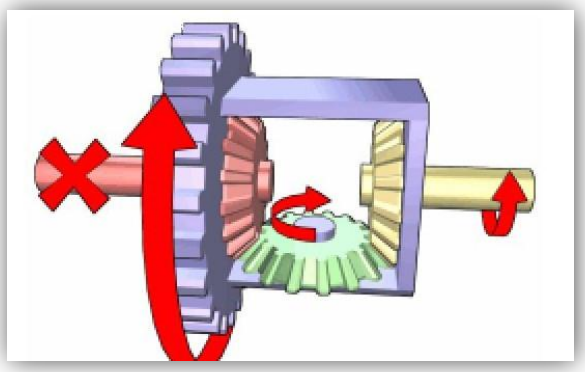


图 2.1 一侧车轮遇到阻力

假设车辆现在向左转，左侧驱动轮行驶的距离短，相对来说会产生更大的阻力。差速器壳体通过齿轮和输出轴相连，在传动轴转速不变情况下差速器壳体的转速也不变，因此左侧半轴齿轮会比差速器壳体转得慢，这就相当于行星齿轮带动左侧半轴会更费力，这时行星齿轮就会产生自传，把更多的扭矩传递到右侧半轴齿轮上，由于行星齿轮的公转外加自身的自传，导致右侧半轴齿轮会在差速器壳体转速的基础上增速，这样以来右车轮就比左车轮转得快，从而使车辆实现顺滑的转弯。

2.1.2. 外倾角

从车头望向车轮,车轮与铅垂线的夹角称为外倾角， 若轮胎上端向外倾斜即左右轮呈"\/"形, 称为正外倾角,向内倾斜为负外倾角。基本上，正外顷角的设定有较佳的灵活度，而负外顷角具较稳定的直进性。

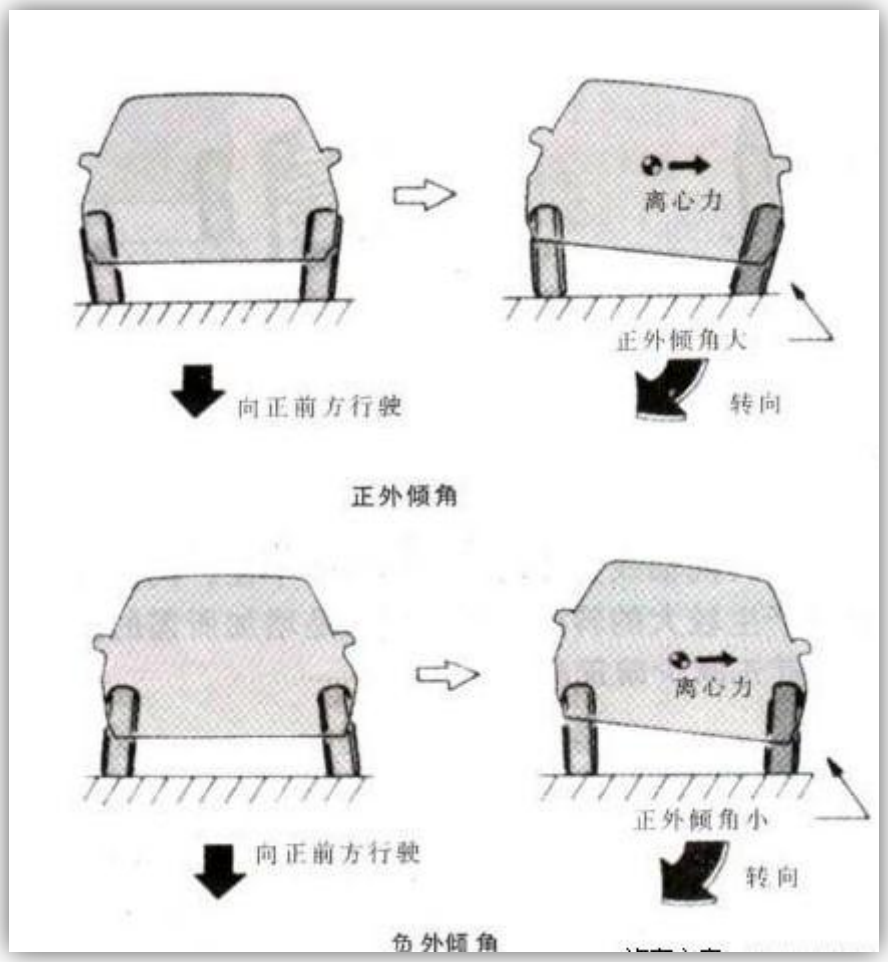


图 2.3 外倾角

2.1.3. 车身结构——轮距

轮距是车轮在车辆支承平面(一般就是地面)上留下的轨迹的中心线之间的距离。如果车轴的两端是双车轮时，轮距是双车轮两个中心平面之间的距离。

汽车的轮距有前轮距和后轮距之分，前轮距是前面两个轮中心平面之间的距离，后轮距是后面两个轮中心平面之间的距离，两者可以相同，也可以有所差别。

一般来说，轮距越宽，驾驶舒适性越高 此外，轮距还对汽车的总宽、总重、横向稳定性和安全性有影响。

2.2 机械调整——前后轮定位

A 车是后轮驱动，前轮是转向轮。由于 A 车本身较轻，在转弯处，因为离心力，会使得车的重心向外倾斜，当摩擦力不足时候，会使得车整体滑出赛道。

而调整外倾角可有调整车模的过弯特性。

外倾角越负，过弯越足，但是车模会出现过弯前轮上下抖动现象，影响过弯速度。外倾角越正，过弯越不足，但是过弯会比较平稳。综合考虑过弯特性以及稳定性，调整前轮外倾角为负 5° 左右。

因为 A 车底盘较低，从而在上坡时，后轮底盘会擦碰到赛道，影响车模正常行驶。所以本文里，将后轮的车轮降低，提高了后部底盘高度。

2.3 机械调整——舵机安装

A 车原来的舵机为横向安装，并且两个连杆是不等长的，为了使左右转对称，本文将舵机竖直安装于车头部分，并且使用两个等长的连杆，从而保证左右转充分对称。由于受舵机本身性能的影响，传动连杆需要尽可能的短，太长会使得力量不足。

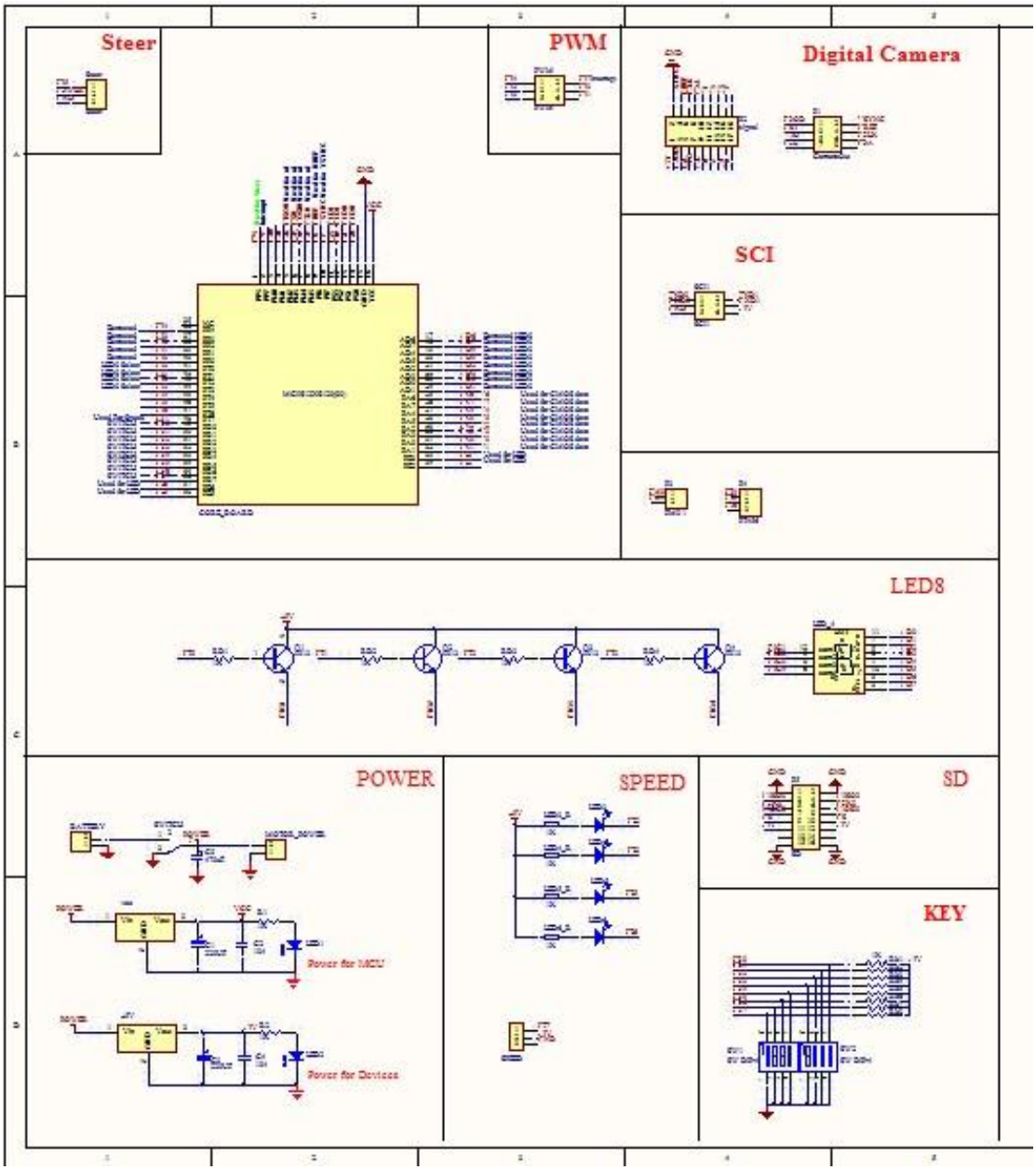
2.4 摄像头安装

摄像头的安装主要是根据采集所需来进行，为保证远处采集到的点畸变尽可能的小，本文里摄像头架得比较高，为了使整车重心尽可能小的受摄像头影响，采用质量较轻的碳纤维棒作为支撑材料。具体安装如图所示：

第三章 硬件电路设计

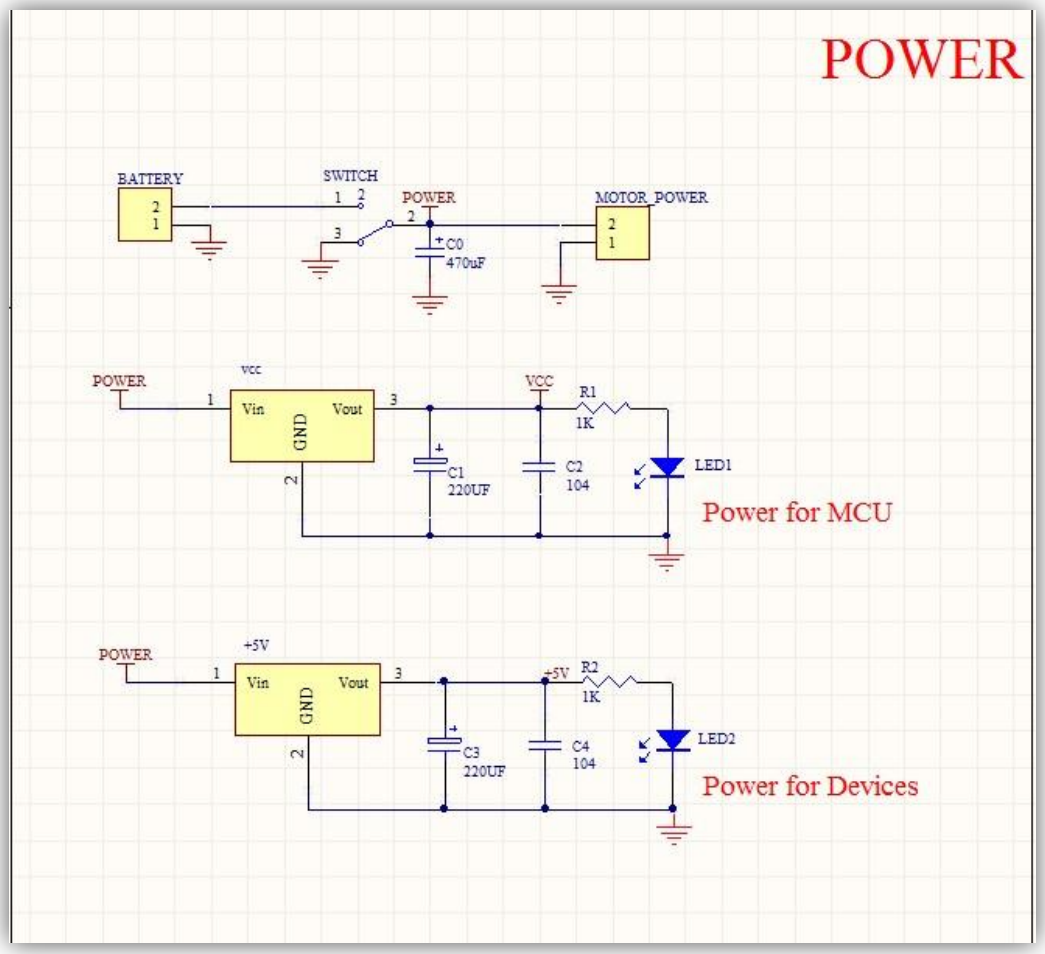
3.1 单片机系统板

控制系统主芯片为飞思卡尔 16 位单片机 MC9S12XS128



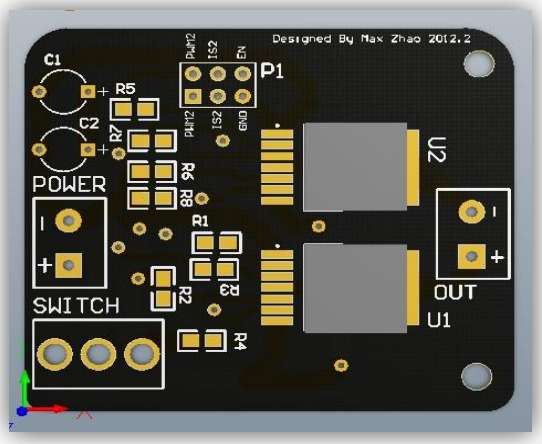
3.2 电源供电

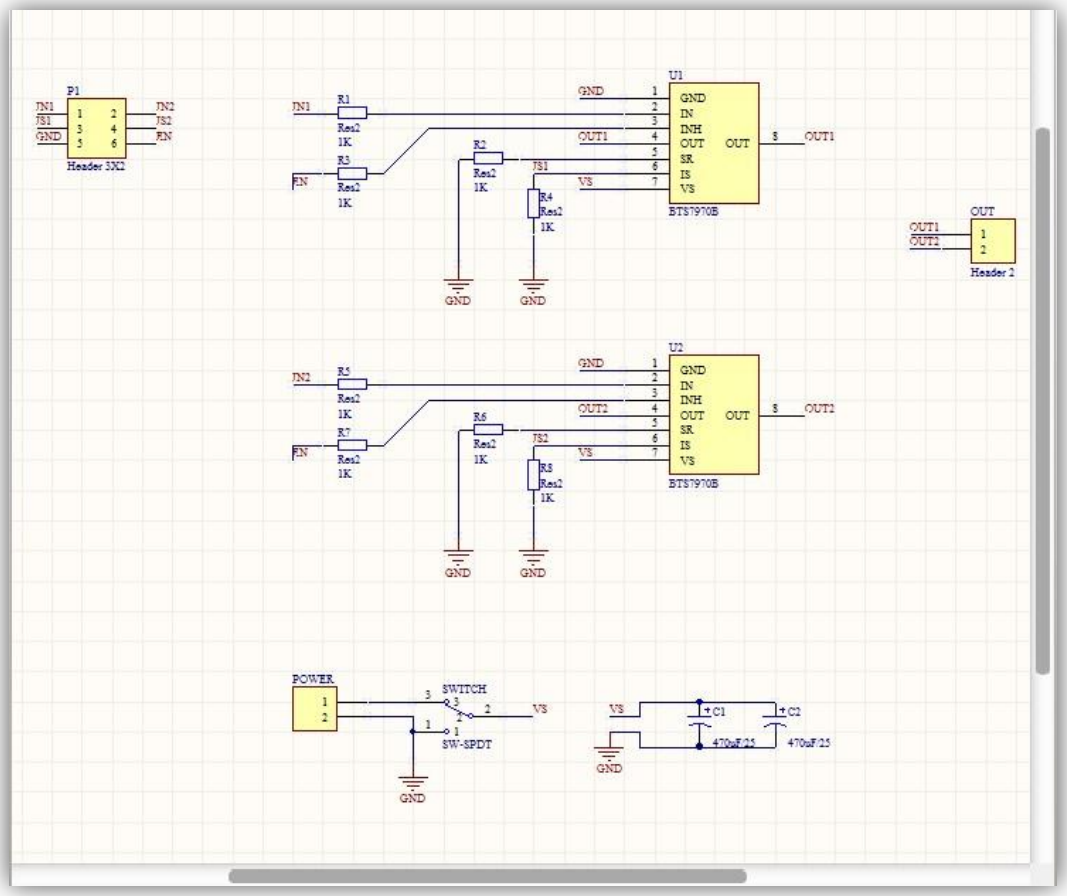
电源供电采用两块 2940，为了保证稳定性，单片机供电单独用一块 2940，其他器件供电单独采用一块 2940。



3.2 电机驱动电路

电机驱动芯片本文选择 BTS797，并搭建的 H 桥电路驱动电机。





3.3 舵机供电

为了增加舵机的响应速度，本文直接用电池给舵机供电。

第四章 控制软件设计

4.1 边线采集

4.1.1. 单行图像数据采集

为了能采集更多的点，我们利用配置锁相环寄存器将 XS128 的总线频率提高到 80MHz，使用 I/O 操作不断读取 PORTB 口状态，利用 IDE 仿真显示每 18 个 CPU 周期即 0.225us 可以采集一个点，一个行周期(64us)大约可以采集 230 个有效点。

4.1.2 整幅图像采集

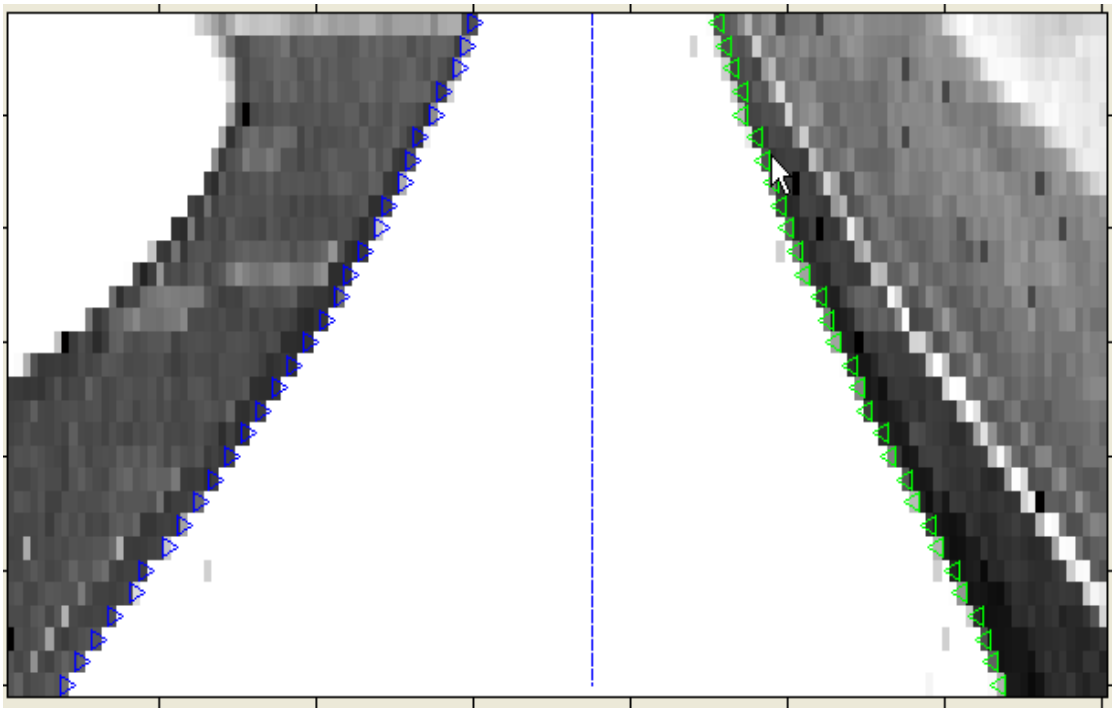


图4.1 赛道图像

摄像头分为奇偶场，我们采取的是一场采集图像，一场处理控制。因为存储空间有限，我们采集 30 行，140 列。

图 4.1 为使用 SD 卡保存的赛道图像，可以看出镜头存在梯形畸变，即近大远小，这对边线的定位不利，因此有必要对这种畸变进行处理。

我们采用相对简单的方法，统计出直道上左右线的畸变规律。利用 SD 卡采集足够多的图像样本（实际样本数为 50），之后由上位机程序对采到图像中的左右线变化进行计数并取平均值。

统计结果如下：

表 4.1 摄像头采集左线位置

0 行	1 行	2 行	3 行	4 行	5 行	6 行	7 行	8 行	9 行
56	55	55	55	53	50	49	46	44	42
10 行	11 行	12 行	13 行	14 行	15 行	16 行	17 行	18 行	19 行
40	38	36	34	31	30	28	26	24	23
20 行	21 行	22 行	23 行	24 行	25 行	26 行	27 行	28 行	29 行
21	20	18	17	15	12	11	11	10	9

表 4.2 摄像头采集左线位置

0 行	1 行	2 行	3 行	4 行	5 行	6 行	7 行	8 行	9 行
55	55	54	54	52	50	48	46	44	41
10 行	11 行	12 行	13 行	14 行	15 行	16 行	17 行	18 行	19 行
39	37	35	34	31	29	27	26	24	22
20 行	21 行	22 行	23 行	24 行	25 行	26 行	27 行	28 行	29 行
21	20	18	16	15	12	11	10	9	9

4.1.3 边线提取

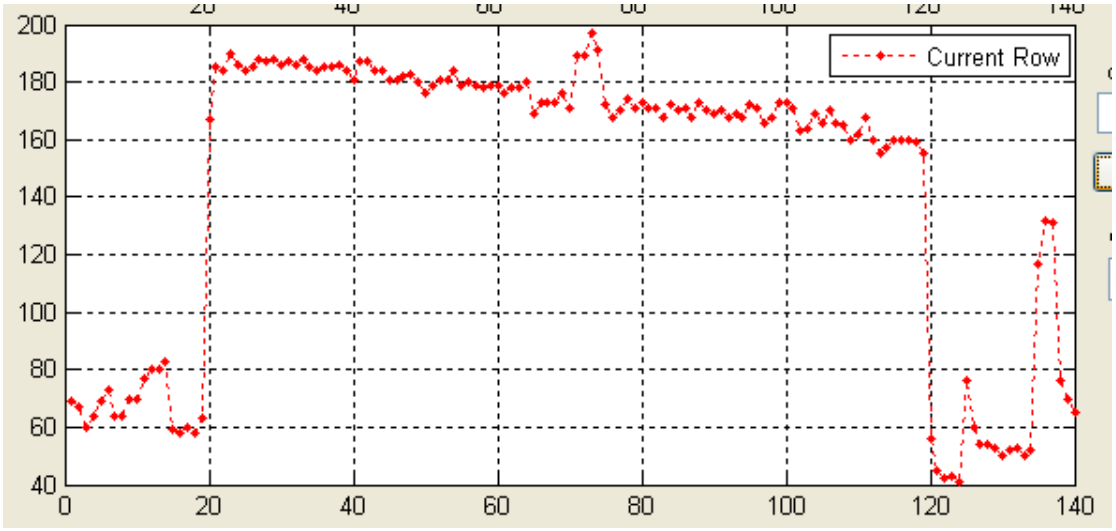


图4.2 一行灰度值变化

边线提取采用窗口搜索算法，该方法经测试可以有效对抗一般反光，其实现方法如下：在赛道中间设置一个窗口，向两边推进。检测窗口内是否有黑白或白黑跳变，如果有，则认为找到了边线并记录。

因为赛道内全部是白色，所以灰度值只会有小范围的波动。而在赛道边缘出现了黑线，灰度值一定会有一个较大的跳变，如图 4.2。这样只要阈值合适，即使有一定的反光，还是可以检测到跳变。

而且从赛道中间往两边检测，可以避免误识别其它赛道和赛道外的反光，提高了抗干扰能力。

这种方法比较困难的是，确保准确的从赛道中开始检测。我们采取的是根据上一场的赛道来推测。

我们提取的边线如下图 4.3，三角形就是我们每行找到的边线点：

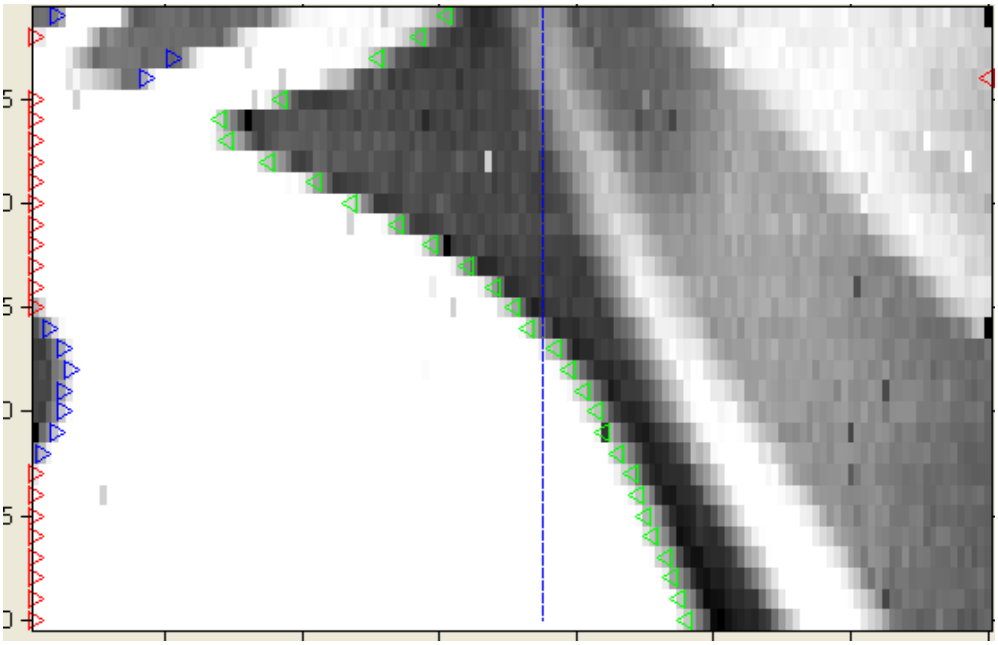


图4.3 边线提取

4.1.4 引导线提取

我们的引导线是根据找到的边线来计算得出的：

1)如果两条线都找到了，用左线位置加上右线位置再除以 2，即是引导线如图 4.4。

2)如果只找到了左线，则用左线减去表 1，得到引导线如图 4.5。

3)如果只找到了右线，则用右线加上表 2，得到引导线。

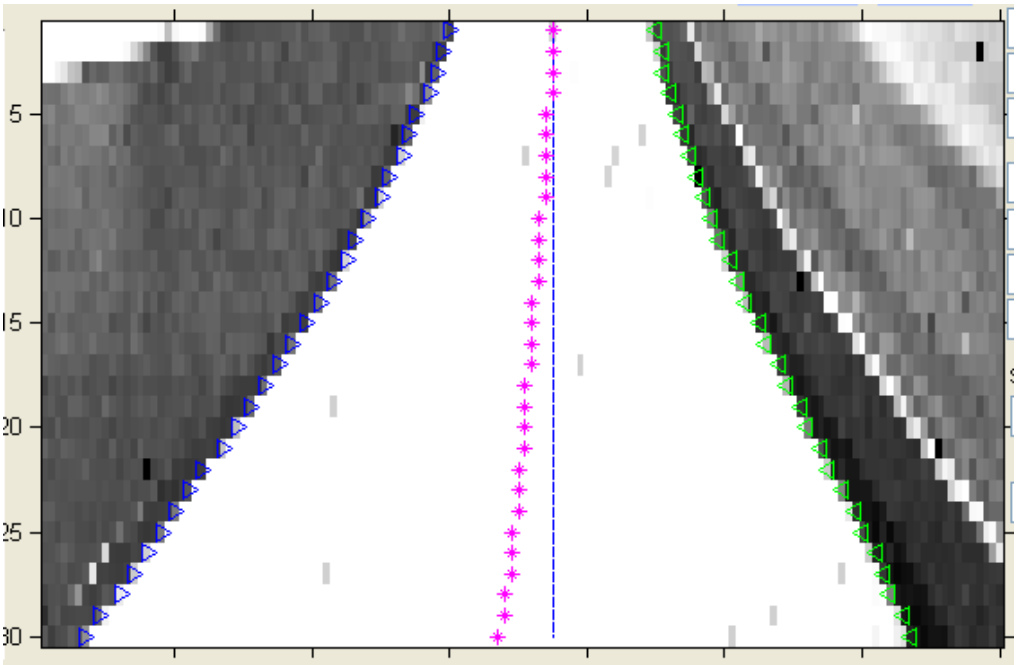


图 4.4 两边线提取引导线

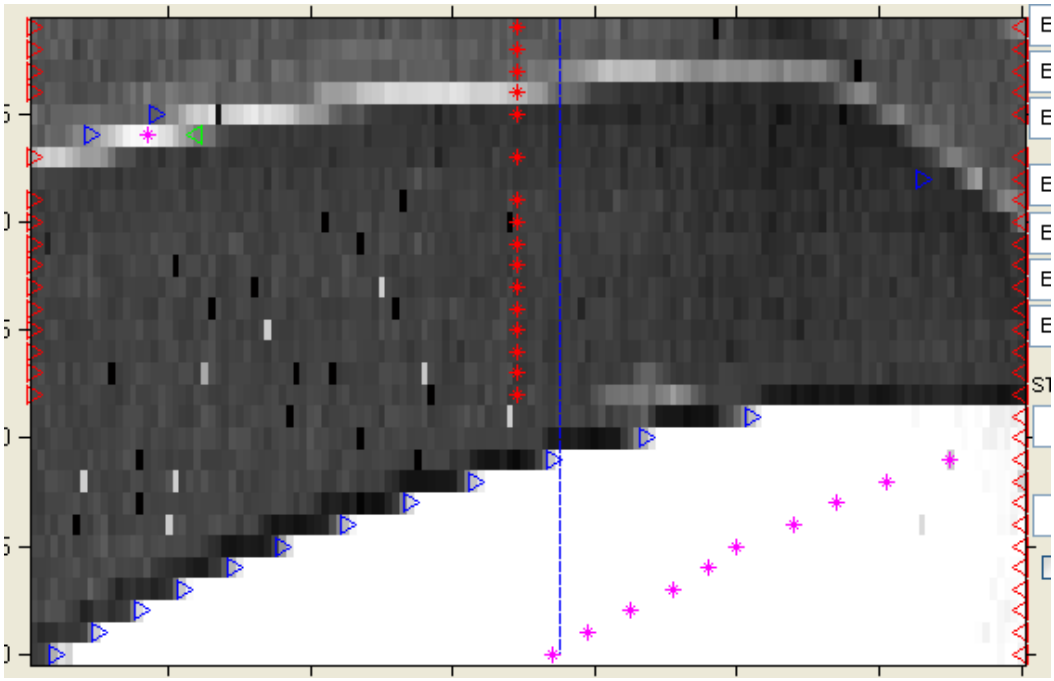


图 4.5 左边线提取引导线

4.1.5 起跑线识别

比赛规则中规定车模必须在跑完一圈之后，停在起始线后的三米区内，否则将罚时 1 秒，标准起跑线如图 4.6 所示。

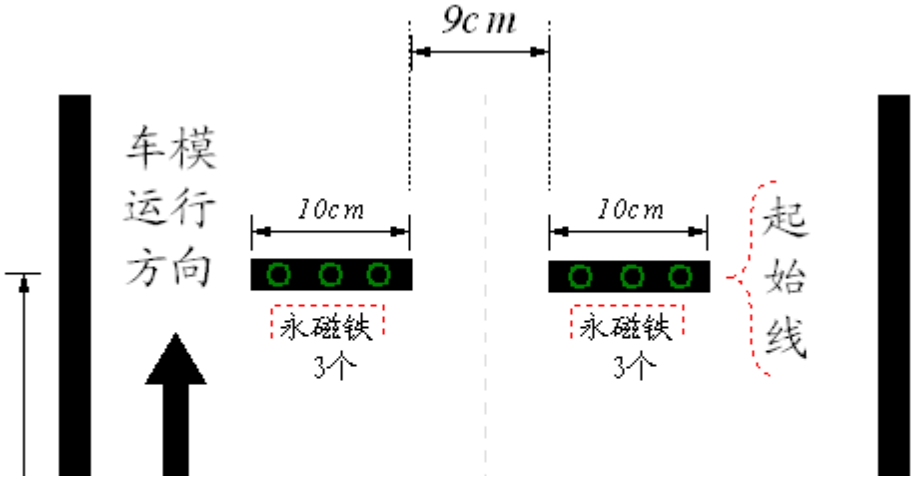


图 4.6 起跑线

根据起跑线的特点：在赛道内部，两个横向宽度远远大于边线宽度的黑线。

我们采取策略：起跑四秒后开始在 5-10 行里检测起跑线即终点线。有 5 个以上跳变，且左线或右线内侧，有一个宽度较大的黑线，则认为已经到了终点线，停车。

4.1.6 坡道识别

坡道识别可以直接利用边线宽度的变化来判断，当远处边线宽度大于正常直道宽度，判定为坡道，限定合适速度使得车辆在坡上不会发生飞跃的情况，同时也防止下坡过快而来不及转弯，图 4.7 为坡道图像.

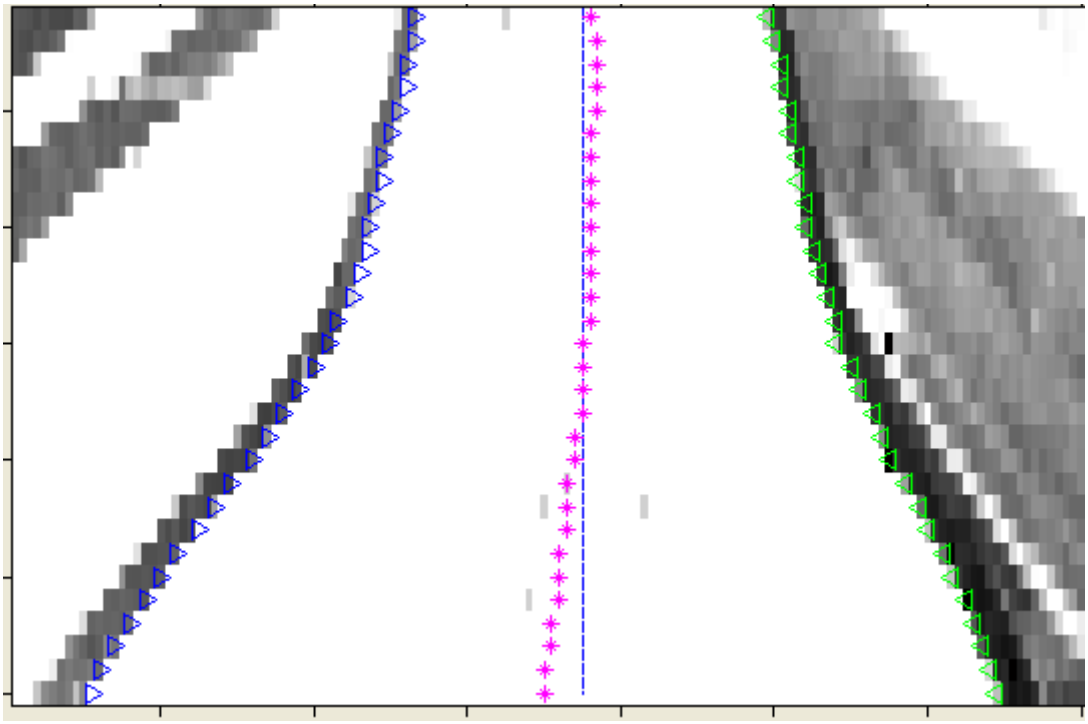


图 4.7 坡道

4.2 最小二乘法

最小二乘法是用来拟合直线最常用的方法，在采集了 30 行有效点后，我们采用这种方法对赛道进行拟合。对于最小乘法的介绍很多书上都有，这里就不对其原理进行敷述，只是对我们如何应用加以介绍：具体来说，选择与行平行方向作为 Y 轴，垂直方向作为 X 轴，对采集到的有效点进行拟合，分别得到整体斜率、整体截距、近处斜率、远处斜率、近处截距等数据，并在拟合过程中注意排除无效点和跳跃太大点（可能的噪点）。得到这些数据后，再进行下一步的赛道类型判断以及控制。

4.3 赛道类型识别

由于赛道类型变化多端，我们利用最小二乘法计算出来的各种斜率和截距，判断出 7 种不同的赛道类型，然后分别进行舵机和电机的给定赋值，以及舵机的 PD 控制参数赋值。

4.3.1 直道

如图 4.4 所示，为赛道类型 0，表示直道，此时速度最快，舵机转弯角度很小。判断条件为：当采集到的边点数很多，斜率很小，方差也很小，并且整体截距在某一范围内。如果远处边线变宽，则能判断出为上坡，如果此时速度太快，则减一下速。直道最快速度有上限，防止速度太快，入弯时不能立即减下来。

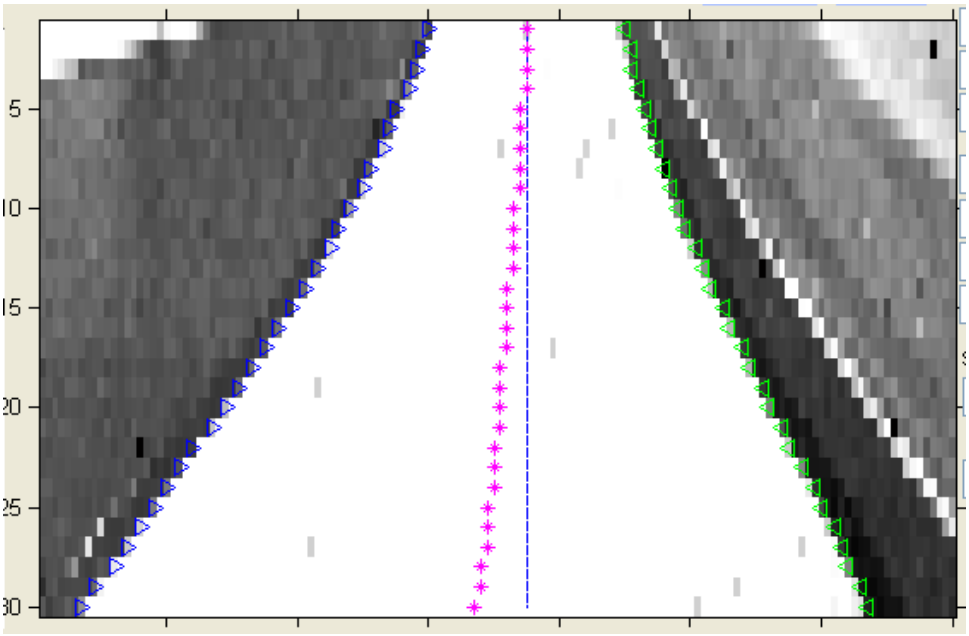


图4.4

4.3.2 直道入弯

如图 4.2 所示，为赛道类型 2，表示直道入弯，此时舵机转弯角度也很小，且要刹车，直道速度降到某一设定值为止。判断条件为：采集到的中点数很多，在前几场图像中判断出来的直道次数很多，且本场图像近处为直道，远处为弯道。

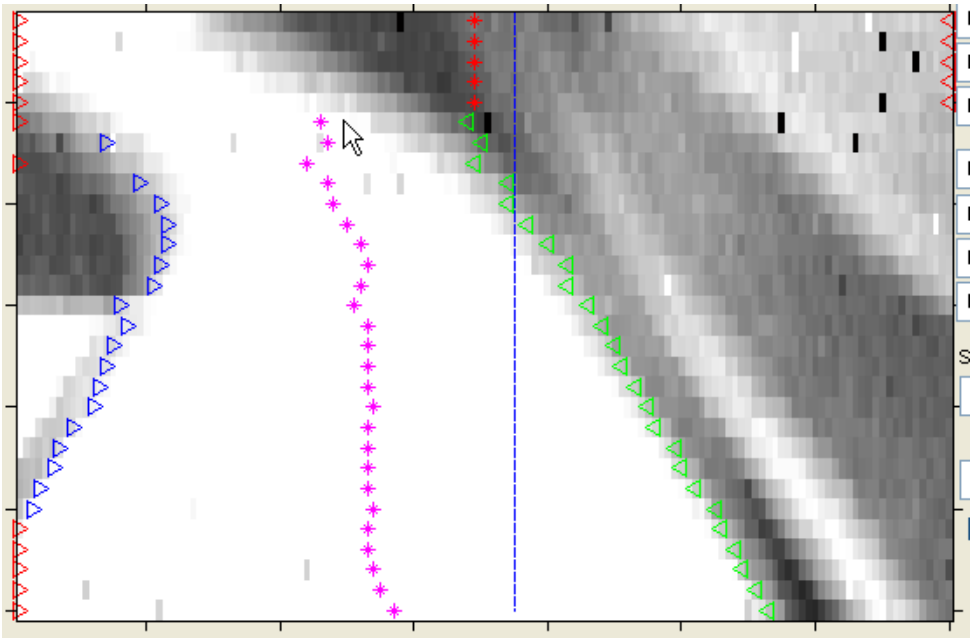


图 4.2

4.3.3 小 S 弯

如图 4.8 所示，为赛道类型 3，表示小 S，此时舵机转弯角度很小，基本不转，速度和直道一样。判断条件为：采集到的点数很多，总斜率偏小，方差不是太大，而且边线有 S 形。

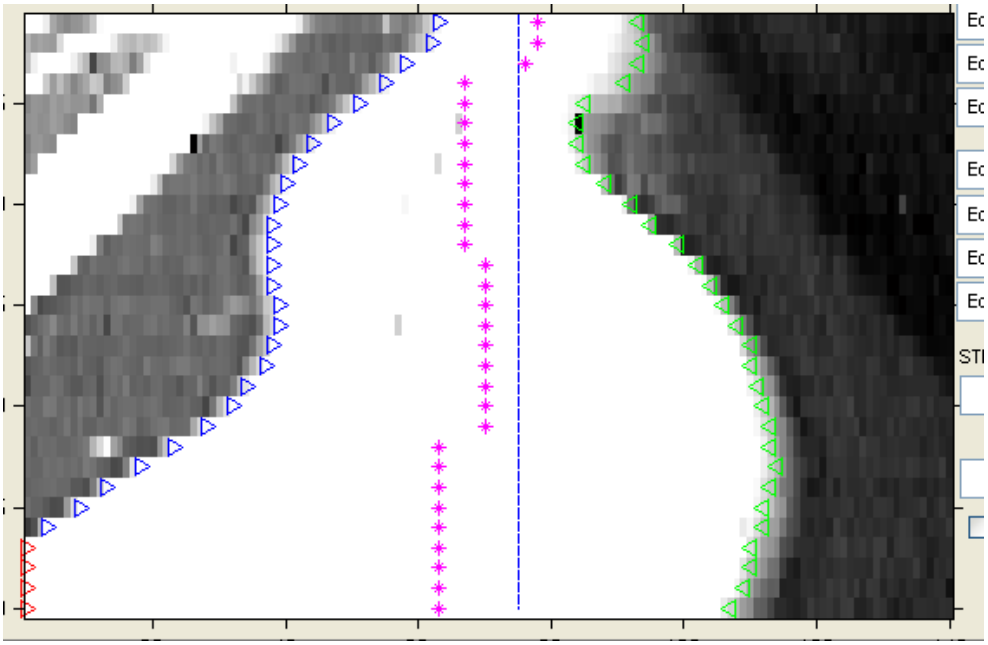


图 4.8

4.3.4 C 弯

如图 4.9 所示，为赛道类型 8，表示远处与近处呈相反方向转的弯。舵机转弯角度根据远断点斜率大小不同而定，速度较小。判断条件为：斜率很大，且两段的斜率和截距的绝对值都较大。

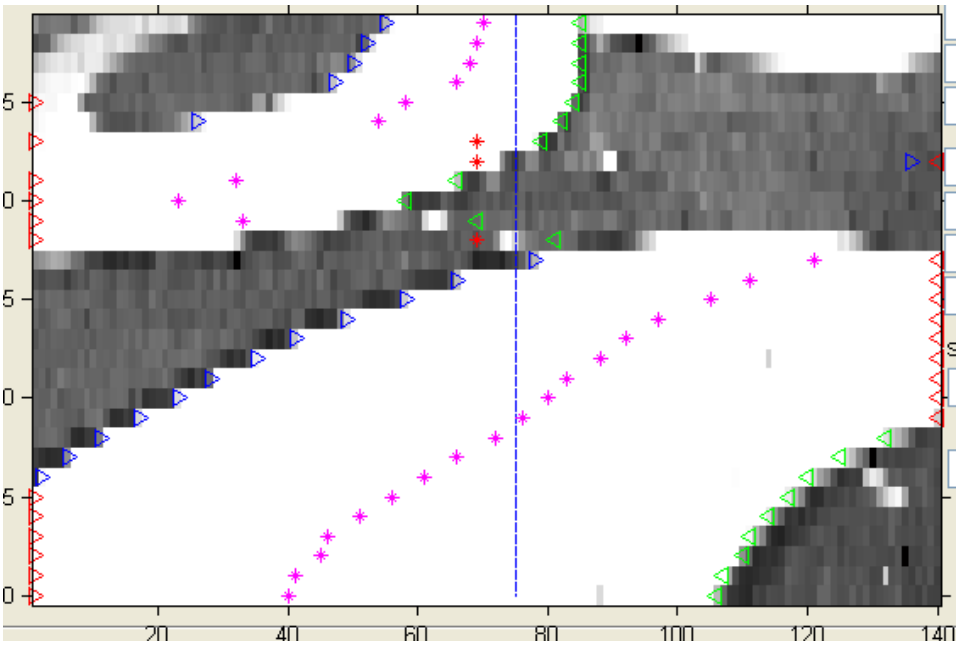


图 4.9

4.3.5 普通弯

如图 4.10 所示，为赛道类型 11，表示采集到中点数较多时，其它所有情况的弯。此时需要根据采集到的中点和斜率的大小来得出舵机转弯角度，速度低于直道大于大弯。判断条件：斜率一般大。

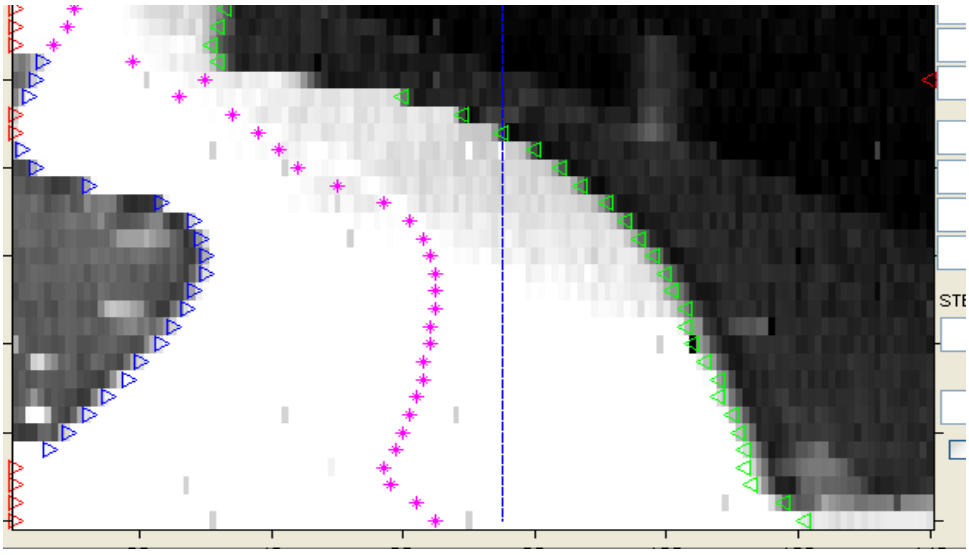


图 4.10

4.3.6 十字弯

如图4.11所示，为赛道类型5，表示赛道被分为前后两段，且中间是空白。此时舵机转角应根据正确方向打角。速度低于知道大于弯道。判断条件：有几行基本全白的行或者左右线出现了尖锐的折角。

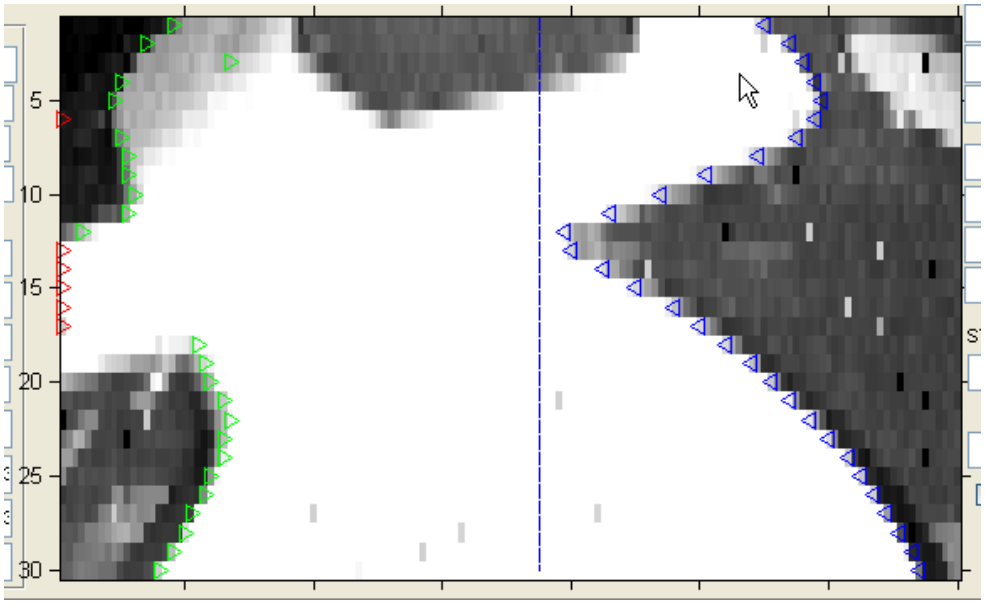


图 4.11

4.3.7 无引导线的弯

为赛道类型6，表示采集到的点数特别少，此时舵机转弯角度保持上一场的值不变。速度很低。

我们的每一个弯，都会分为左转或右转。用来给不同的参数，克服左右不对称。

4.4 运动控制

4.4.1 舵机控制

我们采取引导线偏差和斜率控制结合的方法，采用位置式的PD控制算法。具体公式为：

steer_out=steer_center+SteerE0+SteerKd*(SteerE0-SteerE1)-DeE0-Deviation_D*(DeE0-DeE1); （公式1）

各控制参数在各个赛道类型里面给出。在此算法中，给定值为 0，steer_out 初始值为 steer_center，赛道类型中计算出的舵机给定值即为此处的引导线误差 SteerE0，斜率误差 DeE0,且为了防止误差变化过大，导致下一次控制时迟迟不能将值拉回到所需要的大小，我们对误差进行了限幅，为保护舵机，对最终的舵机输出值也进行

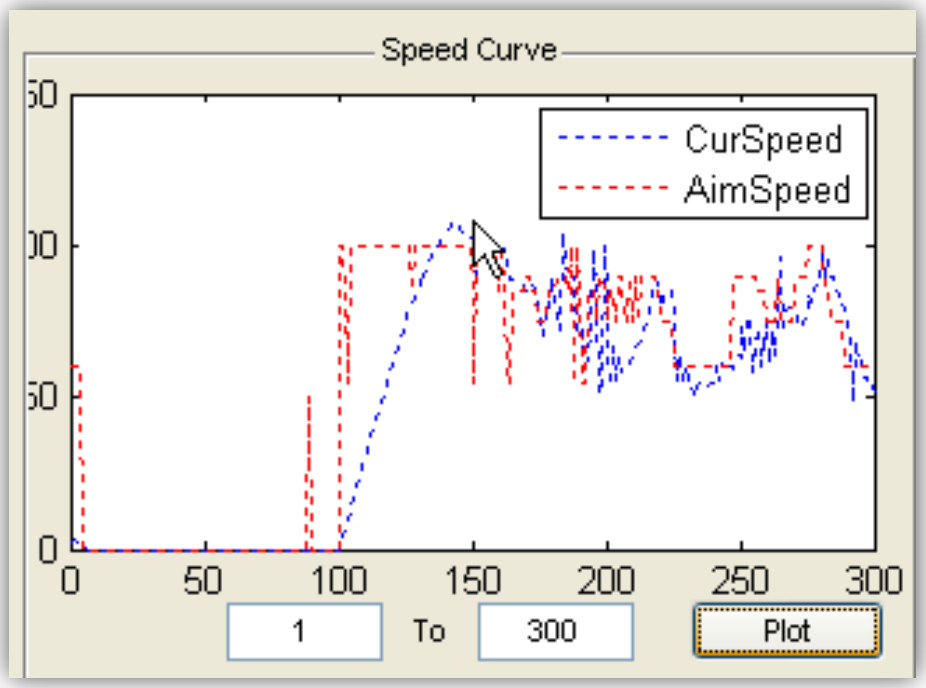
了限幅。

4.4.2 电机控制

由于速度变化并没有舵机输出变化频繁，且为了使速度能够稳定地达到给定值，我们采用了增量式的 PID 控制算法。具体公式为：

motor_out+=Mi*MotorE0+Mp*(MotorE0-MotorE1)+Md*(MotorE0-2*MotorE1+MotorE2);　　（公式 2）

此处我们同样对误差和最终输出给电机的值进行了限幅。为了防止检测到的速度突变导致的震荡，我们对误差进行了不同的处理，当有突变时，当场误差为给定值减去上一场所测得的值，否则为给定值减去本场所测得的值。电机响应曲线如图 4.12 所示。



第五章 开发及调试工具

5.1 开发平台

我们使用 Freescale CodeWarrior 5.0 for HCS12(X)系列开发工具作为我们车上程序的开发工具，其主界面如图 5.1。

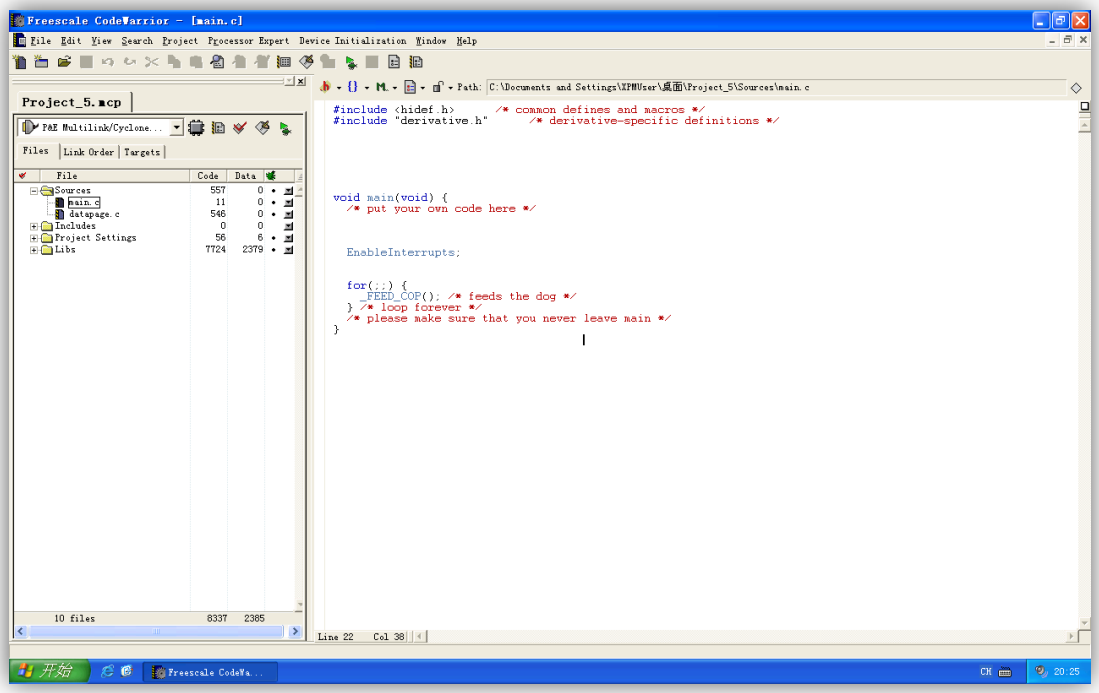


图 5.1 CodeWarrior 主界面

该版本 CodeWarrior 开发工具针对 HCS12(X)系列单片机，包含 XS128 内所有寄存器的定义，可以在编程的时候利用“Alt+.”一组快捷键进行自动提示，如图 5.2。

图 5.2 自动提示功能

CodeWarrior 内部还包含一个全片模拟器，可以方便地在 PC 机上进行仿真，仿真过程中看到响应指令或者函数的运行时间，帮助我们控制好单片机运行的时序，利用仿真功能，我们了解了 80MHz 总线频率下，采集 1 个点需要 18 个时钟周期等数据，图 5.3 展示了这一功能。

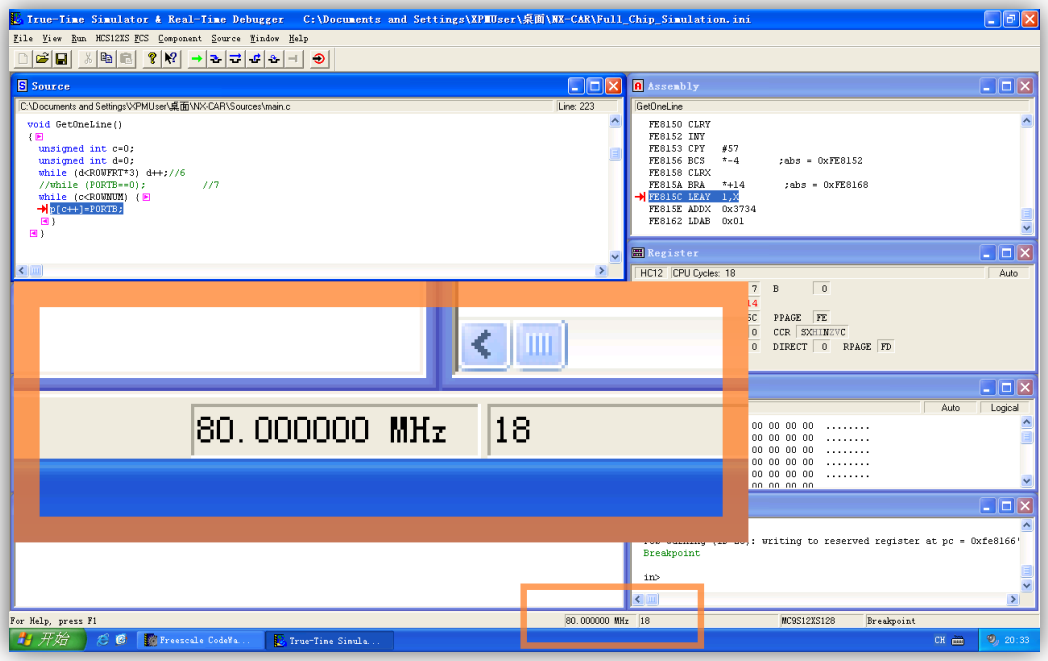


图 5.3 Full Chip Simulation 功能

5.2 SD 卡调试平台

对于我校来说，SD 卡的调试平台是首次使用，我们感觉十分必要而且十分有用。相比于无线调试系统，SD 卡调试平台更加适合摄像头组使用，因为图像数据如果用无线传输的话时间太长，加上相关的确认机制，无法在一个场周期内完成上一场图像的发送，而使用高速 SD 卡（如 SanDisk Ultra II 1G），单片机 SPI 总线可以工作在 20MHz 的速率上，此时一次写卡（5120 字节，10 块）只需要 7-8ms，远小于场间隔 20ms。

由于单片机内部 RAM 有限，我们采到的 40 行数据无法全部存在 5120 字节大小的缓冲区中，我们选择了图像信息只保留奇数行的方法，这样一共需要 4600 字节存储图像数据，剩下的 400 字节存储全部中点信息及车辆运行状况和拟合数据等，图 5.4 为 Matlab 编写的上位机界面。

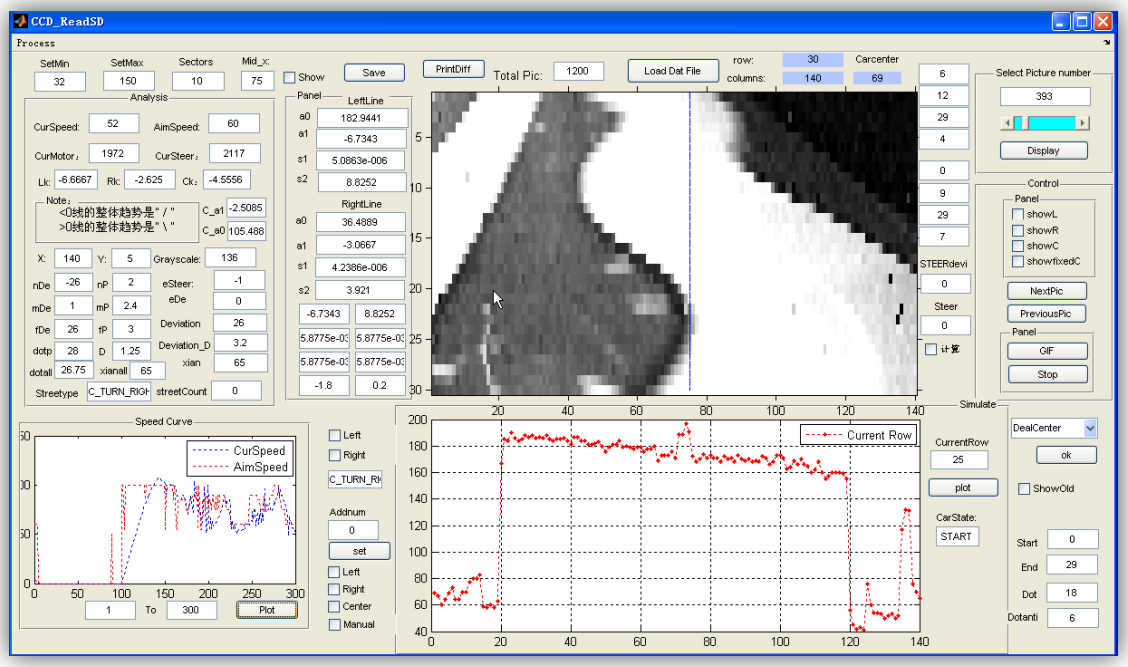


图5.4

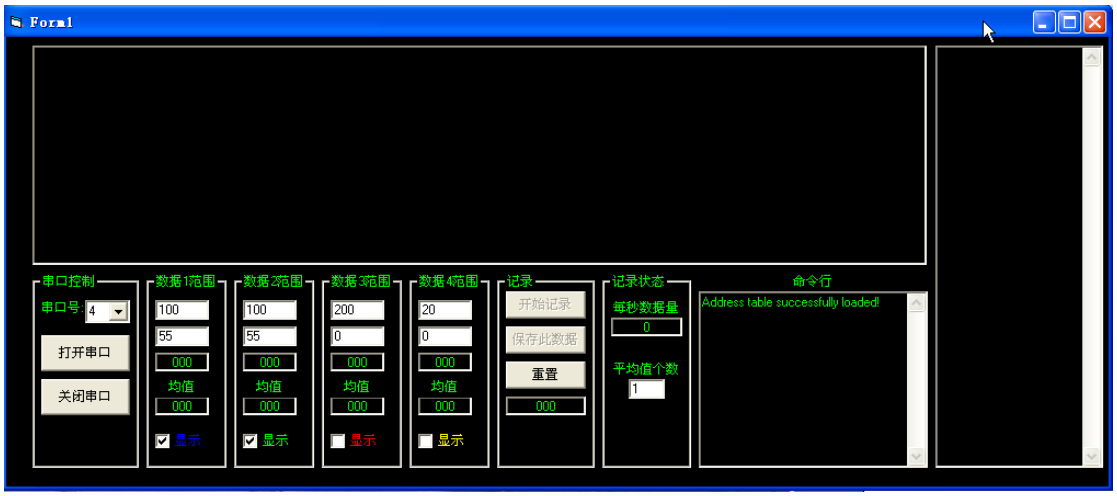
该平台可以显示每一场图像信息对应的中点信息、速度转向信息及拟合数据，还可对整体的速度、转向调节效果进行分析。

配合 Matlab 的上位机平台，我们还使用 VS2010 编写了的原始数据解释程序，可以更加自由的提取原始数据，帮助我们完成分析时序信息，算法仿真等 Matlab 不能完成的一些功能。

5.3 蓝牙无线调试平台

为了能够在线的调试参数，我们使用了蓝牙模块来无线在线修改参数。这样可以避免每次修改参数都需要重新下载程序的麻烦。而且在线调试 PID 效果会更加明显有效。

除了可以发给单片机修改参数外，还可以从单片机发回四个数据用曲线表示。上位机是用 VB 编写，上位机界面如图 5.5。



第六章 总结与展望

我们的队伍从去年开始就与智能车结缘，参加了第五届东南大学智能车竞赛，并通过校队的选拔，代表学校参加第七届全国大学生智能车竞赛华东赛区的比赛，获得了摄像头组第五名的成绩，顺利的进入到了全国赛的比赛中。在华东赛以及全国赛的备战过程中，我们在 SD 卡的调试平台的使用方面积累了不少经验，通过 SD 卡平台的使用，我们加快了对算法的验证与改进速度，这极大地提高了我们的工作效率。我们还开发出了无线上位机平台，实现了实时修改参数了功能，避免了频繁下载程序带了的麻烦，节约了时间，提高了进度，对车模机械结构的改造，由于经验的缺乏，我们只能在摸索中前进，在车模经历了多次由稳定到不稳定再到稳定的过程之后，在经历了多次无可奈何之后，在经历了多次尝试与失败后，最终车模处于一个很好的稳定状态，这也使得我们能最大限度的发掘算法的潜力。提升车模的速度。

调试平台的建立与不断完善，标志着我们的智能车研究进入了一个全新的环节，用好调试工具不仅可以发现程序运行中的问题和算法的错误，更可以在不改变硬件的情况下，使车模的速度趋于最优，真正实现闭环调试。在下一届校队备战智能车的时候，他们可以在进一步改良上位机的同时，进一步的优化车模的路径，改良车模的机械结构，做好速度控制，只有这样才能更好地发挥车模的潜力，提升车模的速度。

智能车比赛已经经历六届比赛，几乎每一年都有新的变化。明年或许车模会变，或许赛道会变，但是智能车团队不会变，智能车人追求卓越的信念不会改变，老队员与新队员的传承不会改变，不同学校之间的技术交流更加便利，愿每一个智能车团队在新的一年中都能取得更新的成果，愿智能汽车有朝一日走入人们的生活，为人类的生活增添新的色彩。

参考文献

[1] 田玉平.自动控制原理[M].北京:科学出版社.2006

[2] 卓晴等.学做智能车:挑战“飞思卡尔”杯[M].北京:北京航空航天大学出版社.2007

[3] 王威等.HCS 12 微控制器原理及应用[M].北京:北京航空航天大学出版社.2007

[4] 刘金琨.先进 PID 控制 MATLAB 仿真[M].北京:电子工业出版社.2004

[5] 罗华飞.MATLAB GUI 设计学习手记[M].北京: 北京航空航天大学出版社.2009

[6] 蔡兴旺.汽车构造与原理[M].北京:机械工业出版社.2010

[7] 张延伟.Protel DXP 电子电路设计技法范例[M].北京:清华大学出版社.2005

[8] 唐向宏等.MATLAB 及在电子信息类课程中的应用[M].北京:电子工业出版社.2009

[9] 华文、俞斌、翁华.基于 CCD 摄像头黑线提取算法[J].电子产品世界，2009，16(1)

[10] 维斯.数据结构与算法分析 C++描述[M].北京:人民邮电出版社.2007

[11] 王勤.计算机控制技术[M].南京:东南大学出版社.2003

致谢

在竞赛的准备期间，我们遇到了不少困难。从电路的设计到机械结构的调整，再到 SD 卡调试平台的建立，离不开学校教务处和智能车团队给予我们的支持和帮助。

首先，要感谢教务处方老师以及指导教师谈老师、孙老师的关心，没有她们的沟通协调，我们将缺乏训练场地和器材。

其次，要感谢上届校队的学长的悉心指导，没有他们的努力，我们的调试平台将进展缓慢，缺少他们的支持，我们难以超越去年的成绩。

最后，要感谢校队的所有成员，没有大家共同的努力，就不可能按时更换赛道，也不可能有竞争的环境让每个队伍不断前进

源代码

```
#include <hidef.h>          /* common defines and macros */
#include <MC9S12XS128.h>         /* derivative information */
#include <math.h>
#include "FAT16.h"
#include "PllInit.h"
#include "arithmetic.h"
#include "Init.h"
#include "SCI.h"
#pragma LINK_INFO DERIVATIVE "mc9s12xs128"
#define BUFFERSIZE       5120
#define row_num           30
#define col_num           140
#define steer_center    3000
#define car_center       70
#define steer_left       2650
#define steer_right     3350
#define motor_left       0
#define motor_right     2000
#define whiteRoad         150


//路况    Cur_street
#define STRIGHT         0
#define SMALL_TURN    1
#define C_TURN          2
#define SMOOTH_S       3
#define SEVERE_S       4
#define CROSSROAD_LEFT     5
#define LOSE_LINE     6
#define OTHERSTREET 7
#define C_TURN_LEFT    8
#define C_TURN_RIGHT 9
```

```
#define CROSSROAD_RIGHT      10
#define SMALL_TURN_LEFT      11
#define SMALL_TURN_RIGHT     12
#define CROSSROAD            13
//CCD_state
#define GETVSYNC        0
#define GETHREF         1
#define WAITVSYNC       2
#define WAITHREF        3
//Line_State
#define ONRIGHT 0
#define ONLEFT    1
#define NORMAL    2
//车内部状态 Car_state
#define STOP             0
#define START            1
#define INIT             2
#define LOST             3
#define P_WIDTH          8
#define BW_DELTA         30
#define LINE_EDGE      2    //----4-15----- 2->3
#define EndLine_Length    10
#pragma DATA_SEG DEFAULT

unsigned char    S12BUFFER[BUFFERSIZE];
unsigned char* IMAGE_P;
unsigned int*    CURSPEED_P;                    t

unsigned char * Car_Center;//2.26
int              * Center_P;                    /
int              * centerfixed_P;
unsigned char* Streetstate_P;
unsigned char* Carstate_P;
unsigned char* CCDstate_P;
```

```
unsigned int*    VsyncNum_P;
int *near_piancha_P;
int *mid_piancha_P;
int *far_piancha_P;
int *allpiancha_P;
float *steer_nearp_P;
float *steer_midp_P;
float *steer_farp_P;
float *steerKD_P;
int *e_steer_P;
unsigned char *C_Start_P;//2.27
unsigned char *C_End_P;
unsigned char *Dot_P;
unsigned char *Dotanti_P;

float *L_a0_P;
float *L_a1_P;
float *L_s1_P;
float *L_s2_P;
float *R_a0_P;
float *R_a1_P;
float *R_s1_P;
float *R_s2_P;
float *C_a0_P;
float *C_a1_P;
unsigned char *L_Count_P;
unsigned char *R_Count_P;

unsigned char *L_Start_P;
unsigned char *R_Start_P;
unsigned char *L_End_P;
unsigned char *R_End_P;
unsigned char *L_Count_anti_P;
unsigned char *R_Count_anti_P;
```

```
unsigned int    *streetcount_P;
float * L_a0_front_P,*L_a1_front_P,*L_s1_front_P,*L_s2_front_P;
float * R_a0_front_P,*R_a1_front_P,*R_s1_front_P,*R_s2_front_P;
float * L_a0_back_P,*L_a1_back_P,*L_s1_back_P,*L_s2_back_P;
float * R_a0_back_P,*R_a1_back_P,*R_s1_back_P,*R_s2_back_P;
unsigned char    SDReady=0;
unsigned char    SDError=0;
unsigned char    FILENAME[]="SMARTCARDAT";
unsigned char Car_state;
unsigned char Cur_street;
unsigned char Far_street;
unsigned char Last_street;
unsigned char CCD_state;
unsigned int    Cur_Speed;
unsigned char Line_State;
//////////////////////数码管//////////////////////
unsigned char LedCode[]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90};
unsigned char LedData[]={0,0,0,0};
unsigned char LedNum=0;

unsigned char VSync;
unsigned char *image;
int image_v_num=0;
int v_counter=0;                        /
unsigned int c=0;                       /
unsigned int All_VSync;
//
///////////////

unsigned char C_Start,C_End;
unsigned char L_Start,L_End;
unsigned char R_Start,R_End;
unsigned char L_Count;
unsigned char R_Count;
```

```
//***4.16****
unsigned char L_Count_anti;
unsigned char R_Count_anti;
//int center_line[row_num];
int center[row_num];                    /
int centerfixed[row_num];

unsigned char Lx[row_num];
unsigned char Rx[row_num];
unsigned char UnLx[row_num];
unsigned char UnRx[row_num];

unsigned char Lxlast[row_num];
unsigned char Rxlast[row_num];
float Lk,Rk;
float Ck;//center 斜率
float Cnk;//centerfixed 远处斜率
float Cfk;//centerfixed 近处斜率

int LRightEdge,LLeftEdge;
int LRightEdge1,LLeftEdge1;
int LRightEdge2,LLeftEdge2;
int LRightEdge3,LLeftEdge3;
unsigned char stopLx,stopRx;
unsigned char FindStartingLine;
//////////////////////舵机控制//////////////////
float steer_p_mid=5;
float steer_p_far=3;

float SteerKd=2;
int near_piancha;
int mid_piancha;
int far_piancha;
int allpiancha;
```

```
int SteerE0=0;
int SteerE1=0;
int SteerE2=0;
int e_steer;

unsigned char maxindex,minindex;
unsigned char dot ;
unsigned char dot_anti;
int Count_Num=0;
unsigned int time=0;
//////////////////////速度控制//////////////////////
unsigned char FullDuty;
unsigned int AIMSPEED=20;
unsigned int LASTAIMSPEED=20;
unsigned char STEERdeviation=50;
int motor_out=0;                    int MotorE2,MotorE1,MotorE0;
int Mi,Md,Mp;
unsigned char init_rti_ok=0;
unsigned char is_anti_motor;              /
float Motor_a,Cur_MotorUd,Last_MotorUd;/
int U1_0,U1_1,E_0,E_1,U_0,U_1,R_0;
float Fek0,Fek1,A,B;
unsigned int streetcount;   ////////////////////直线拟合/////////////////
float L_a0,L_a1;
float R_a0,R_a1;
float L_s1,L_s2;
float R_s1,R_s2;
float C_a1,C_a0;
float C_s1,C_s2;
float L_a0_front,L_a1_front,L_s1_front,L_s2_front;
float R_a0_front,R_a1_front,R_s1_front,R_s2_front;
float L_a0_back,L_a1_back,L_s1_back,L_s2_back;
float R_a0_back,R_a1_back,R_s1_back,R_s2_back;
unsigned char firstdot,seconddot;
```

```
unsigned char C_TURN_RIGHT_D;
unsigned char C_TURN_LEFT_D;
unsigned char CROSSROAD_RIGHT_D;
unsigned char CROSSROAD_LEFT_D;
unsigned char DEFAULT_D;

unsigned char SLOW_SPEED;
unsigned char LOSE_LINE_SPEED;
unsigned char STRIGHT_SPEED;
unsigned char C_TURN_SPEED;
unsigned char S_SPEED;
unsigned char DEAFUALT_SPEED;

const int selectRow[]=
{
   30,      37,      44,      51,      58,      65,      72,      79,      86,      93,
   100,     106,     112,     118,     124,     130,     135,     140,     145,     150,
   154,     158,     162,     166,     170,     174,     178,     181,     184,     187,
}   ;
unsigned char offset[ ]= {                  //每一行的 lp1,lp2 扫描偏移量
 40,      39,      38,      37,      36,      35,      34,      33,      32,      31,
 30,      29,      28,      27,      26,      25,      24,      23,      22,      21,
 20,      19,      18,      17,      16,      15,      14,      13,      12,      10,
};
const char DistanceR[ ]=                    //黑线距离中心的距离
{
    61,      60,      58,      56,      54,      52,      51,      48,      46,      44,
    42,      40,      38,      36,      35,      32,      31,      29,      27,      25,
    24,      22,      21,      20,      17,      16,      14,      13,      11,      10,

 };
const char DistanceL[ ] =
{
```

```
    61,    60,    58,    56,    54,    52,    51,    48,    46,    44,
    42,    40,    38,    36,    35,    32,    31,    29,    27,    25,
    24,    22,    21,    20,    17,    16,    14,    13,    11,    10,
};
#pragma CODE_SEG DEFAULT

void Initial_SDCard()//初始化成功，PE6 灯亮
{
  unsigned char temp;
  unsigned char retry;
  temp=1;
  retry=0;
  while(temp==1 && retry<10)
  {
    temp=SD_File_init(FILENAME);
    retry++;
  }
  SDReady=(temp==0)?1:0;//成功 SDReady=1；
}
void delay10us(int num)
{
  unsigned char i;
  int temp;
  for(i=0;i<num;i++)
  {
    temp=0;
    while(temp<665)temp++;
  }
}
void init (void)
{
  int i;
  initPLL80M();
  initPWM();
```

```
PJ_init();
ATD0_init();
initGPIO();
InitPACNT();  //脉冲计数初始化
InitECT0();   //TCNT 初始化
InitSCI(0);
/////////初始化 SD 卡的区域//////////////
IMAGE_P=S12BUFFER+2;                    //2
image=IMAGE_P;
S12BUFFER[0]=row_num;
S12BUFFER[1]=col_num;
CURSPEED_P=(unsigned int*)(S12BUFFER+4500);        //4500
AIMSPEED_P=CURSPEED_P+1;                              //4502   uint 16bits
CURMOTOR_P=AIMSPEED_P+1;                              //4504   uint
CURSTEER_P=CURMOTOR_P+1;                              //4506   uint
Lx_P=(unsigned char*)(S12BUFFER+4508);              //4508 Lx_P 4508-4557
Rx_P=(unsigned char*)(S12BUFFER+4558);              //4558-4607
Lk_P=(float*)(S12BUFFER+4608);                    //4608
Center_P=(int *)(S12BUFFER+4612);           //4612-4711
centerfixed_P=(int *)(S12BUFFER+4712);   //4712-4811
Car_Center=(unsigned char *)(S12BUFFER+4814);//2.26
C_Start_P=(unsigned char *)(S12BUFFER+4815);//2.27
C_End_P=(unsigned char *)(S12BUFFER+4816);
Dot_P=(unsigned char *)(S12BUFFER+4817);
Dotanti_P=(unsigned char *)(S12BUFFER+4818);
Cnk_P=(float *)(S12BUFFER+4820);//2.28
Streetstate_P=(unsigned char*)(S12BUFFER+4824);
Carstate_P=(unsigned char*)(S12BUFFER+4825);
CCDstate_P=(unsigned char*)(S12BUFFER+4826);
near_piancha_P=(int *)(S12BUFFER+4828);
mid_piancha_P=(int *)(S12BUFFER+4830);
far_piancha_P=(int *)(S12BUFFER+4832);
allpiancha_P=(int *)(S12BUFFER+4834);
steer_nearp_P=(float *)(S12BUFFER+4836);
```

```
steer_midp_P=(float *)(S12BUFFER+4840);
steer_farp_P=(float *)(S12BUFFER+4844);
steerKD_P=(float *)(S12BUFFER+4848);
e_steer_P=(int *)(S12BUFFER+4852);
L_Count_P=(unsigned char*)(S12BUFFER+4862);
R_Count_P=(unsigned char*)(S12BUFFER+4863);
Rk_P=(float*)(S12BUFFER+4864);//----4.15-----
Ck_P=(float *)(S12BUFFER+4868);
Cfk_P=(float *)(S12BUFFER+4872);
L_a0_P=(float *)(S12BUFFER+4876);//----4.15----
L_a1_P=(float *)(S12BUFFER+4880);
L_s1_P=(float *)(S12BUFFER+4884);
L_s2_P=(float *)(S12BUFFER+4888);
R_a0_P=(float *)(S12BUFFER+4892);
R_a1_P=(float *)(S12BUFFER+4896);
R_s1_P=(float *)(S12BUFFER+4900);
R_s2_P=(float *)(S12BUFFER+4904);
//5.4
L_Start_P=(unsigned char *)(S12BUFFER+4908);
L_End_P=(unsigned char *)(S12BUFFER+4909);
L_Count_anti_P=(unsigned char *)(S12BUFFER+4910);
R_Start_P=(unsigned char *)(S12BUFFER+4911);
R_End_P=(unsigned char *)(S12BUFFER+4912);
R_Count_anti_P=(unsigned char *)(S12BUFFER+4913);
//5.9
streetcount_P=(unsigned int*)(S12BUFFER+4914);

L_a0_front_P=(float *)(S12BUFFER+4916);
L_a1_front_P=(float *)(S12BUFFER+4920);
L_s1_front_P=(float *)(S12BUFFER+4924);
L_s2_front_P=(float *)(S12BUFFER+4928);

L_a0_back_P=(float *)(S12BUFFER+4932);
L_a1_back_P=(float *)(S12BUFFER+4936);
```

```
L_s1_back_P=(float *)(S12BUFFER+4940);
L_s2_back_P=(float *)(S12BUFFER+4944);

R_a0_front_P=(float *)(S12BUFFER+4948);
R_a1_front_P=(float *)(S12BUFFER+4952);
R_s1_front_P=(float *)(S12BUFFER+4956);
R_s2_front_P=(float *)(S12BUFFER+4960);

R_a0_back_P=(float *)(S12BUFFER+4964);
R_a1_back_P=(float *)(S12BUFFER+4968);
R_s1_back_P=(float *)(S12BUFFER+4972);
R_s2_back_P=(float *)(S12BUFFER+4976);

C_a0_P=(float*)(S12BUFFER+4980);
C_a1_P=(float*)(S12BUFFER+4984);

UnLx_P=(unsigned char*)(S12BUFFER+4988);
UnRx_P=(unsigned char*)(S12BUFFER+5018);
VsyncNum_P=(unsigned int *)(S12BUFFER+5119);
/////////状态初始化//////////////////
Car_state=INIT;         //状态初始化
Cur_street=STRIGHT; /NC;
All_VSync=0;
PWMDTY01=0;
PWMDTY23=0;
PWMDTY45=steer_center;
stopLx=0;
stopRx=0;
time=0;
init_rti_ok=0;
steer_p_near=2;                 steer_p_mid=2;
steer_p_far=2;
SteerKd=1.2;
Mp=17;
```

```
Mi=8;
Md=3;
is_anti_motor=0;
for(i=0;i<row_num;i++)
{
   Lxlast[i]=Lx[i]=car_center;
   Rxlast[i]=Rx[i]=car_center;
}
L_Start=R_Start=0;
L_Count=R_Count=0;
PORTA_PA4=1;
PORTA_PA5=1;
PORTA_PA6=1;
PORTA_PA7=1;
PWMDTY45=steer_center;
streetcount=0;
FullDuty=0;
VSync=0;
AIMSPEED=40;
RecBuf0[0]=6;
RecBuf0[1]=164;
RecBuf0[2]=3;
RecBuf0[3]=32;
RecBuf0[4]=1;
RecBuf0[5]=44;
RecBuf0[6]=0;
RecBuf0[7]=200;
RecBuf0[8]=0;
RecBuf0[9]=200;
RecBuf0[10]=1;
RecBuf0[11]=44;
RecBuf0[12]=0;
RecBuf0[13]=50;
RecBuf0[14]=19;
```

```
    RecBuf0[15]=136;
    RecBuf0[16]=19;
    RecBuf0[17]=136;
    RecBuf0[18]=23;
    RecBuf0[19]=112;
    RecBuf0[20]=23;
    RecBuf0[21]=112;
    RecBuf0[22]=19;
    RecBuf0[23]=136;
    RecBuf0[24]=7;
    RecBuf0[25]=208;
    RecBuf0[26]=0;
    RecBuf0[27]=0;
    RecBuf0[28]=23;
    RecBuf0[29]=112;
    RecBuf0[30]=19;
    RecBuf0[31]=136;
    RecBuf0[32]=23;
    RecBuf0[33]=112;
    RecBuf0[34]=19;
    RecBuf0[35]=136;
    RecBuf0[36]=19;
    RecBuf0[37]=136;
}

void DateToSD(void)
{
      int i;
     *CURSPEED_P=Cur_Speed;
     *AIMSPEED_P=AIMSPEED;
     *CURMOTOR_P=motor_out;
     *CURSTEER_P=steer_out;
     for(i=0;i<row_num;i++) {
          *(Lx_P+i)=Lx[i];
```

```
        *(Rx_P+i)=Rx[i];
        *(Center_P+i)=center[i];
        *(centerfixed_P+i)=centerfixed[i];
        *(UnLx_P+i)=UnLx[i];
        *(UnRx_P+i)=UnRx[i];
    }
    *Lk_P=Lk;
    *Rk_P=Rk;
    *Ck_P=Ck;
    *Streetstate_P=Cur_street;
    *Carstate_P=Car_state;
    *CCDstate_P=CCD_state;
    *near_piancha_P=near_piancha;
    *mid_piancha_P=mid_piancha;
    *far_piancha_P=far_piancha;
    *allpiancha_P=allpiancha;
    *steer_nearp_P=steer_p_near;
    *steer_midp_P=steer_p_mid;
    *steer_farp_P=steer_p_far;
    *steerKD_P=SteerKd;
    *e_steer_P=e_steer;
    *VsyncNum_P=All_VSync;
    *Car_Center=car_center; //2.26
    *C_Start_P=C_Start; //2.27
    *C_End_P=C_End;
    *Dot_P=dot;
    *Dotanti_P=dot_anti;
    *Cnk_P=Cnk;//2.28
    *Cfk_P=Cfk;
    *L_Count_P=L_Count;
    *L_Start_P=L_Start;
    *L_End_P=L_End;
    *L_Count_anti_P=L_Count_anti;
    *R_Start_P=R_Start;
```

```
*R_Count_P=R_Count;
*R_End_P=R_End;
*R_Count_anti_P=R_Count_anti;
*L_a0_P=L_a0;
*L_a1_P=L_a1;
*L_s1_P=L_s1;
*L_s2_P=L_s2;
*R_a0_P=R_a0;
*R_a1_P=R_a1;
*R_s1_P=R_s1;
*R_s2_P=R_s2;
*streetcount_P=streetcount;

*L_a0_front_P=L_a0_front;
*L_a1_front_P=L_a1_front;
*L_s1_front_P=L_s1_front;
*L_s2_front_P=L_s2_front;

*L_a0_back_P=L_a0_back;
*L_a1_back_P=L_a1_back;
*L_s1_back_P=L_s1_back;
*L_s2_back_P=L_s2_back;

*R_a0_front_P=R_a0_front;
*R_a1_front_P=R_a1_front;
*R_s1_front_P=R_s1_front;
*R_s2_front_P=R_s2_front;

*R_a0_back_P=R_a0_back;
*R_a1_back_P=R_a1_back;
*R_s1_back_P=R_s1_back;
*R_s2_back_P=R_s2_back;

*C_a0_P=C_a0;
```

```
    *C_a1_P=C_a1;
}
//////////////////////算法开始//////////////////
void quzaozongxiang()
{
   uchar i;
   uchar c=0;
   unsigned char *p;
   //纵向中值滤波
   for(c=0;c<col_num-1;c++){
        //p=image+i*col_num;
        for(i=0;i<row_num-2;i++){
        p=image+i*col_num;

//if(*(p+c+i*row_num)>=*(p+c+(i+1)*row_num)&&*(p+c+i*row_num)<=*(p+c+(i+2)*row_num)||(*(p+c+i*row_num)<*(p+c+(i+1)*row_num)&&*(p+c+i*row_num)>*(p+c+(i+2)*row_num))*(p+c+1)=*(p+c);
        if(*(p+c)>=*(p+c+col_num)&&*(p+c)<=*(p+c+2*col_num)||(*(p+c)<*(p+c+col_num)&&*(p+c)>*(p+c+2*col_num))) *(p+c+col_num)=*(p+c);
        //else if(*(p+c+2)>=*(p+c)&&*(p+c+2)<=*(p+c+1)||(*(p+c+2)<*(p+c)&&*(p+c+2)>*(p+c+1)))*(p+c+1)=*(p+c);
        else if(*(p+c+2*col_num)>=*(p+c)&&*(p+c+2*col_num)<=*(p+c+col_num)||(*(p+c+2*col_num)<*(p+c)&&*(p+c+2*col_num)>*(p+c+col_num)))*(p+c+col_num)=*(p+c+2*col_num);
        }
   }
   //横向中值滤波
  /* for(i=0;i<row_num-1;i++){
        p=image+i*col_num;
        for(c=0;c<col_num-2;c++){
```

```
if(*(p+c)>=*(p+c+1)&&*(p+c)<=*(p+c+2)||(*(p+c)<*(p+c+1)&&*(p+c)>*(p+c+2)))*(p+c+1)=*(p+c);
        else
if(*(p+c+2)>=*(p+c)&&*(p+c+2)<=*(p+c+1)||(*(p+c+2)<*(p+c)&&*(p+c+2)>*(p+c+1)))*(p+c+1)=*(p+c);
        }
   }*/

}
void quzaohengxiang()
{
   uchar i;
   uchar c=0;
   unsigned char *p;
  //横向中值滤波
   for(i=0;i<row_num-1;i++){
        p=image+i*col_num;
        for(c=0;c<col_num-2;c++){

if(*(p+c)>=*(p+c+1)&&*(p+c)<=*(p+c+2)||(*(p+c)<*(p+c+1)&&*(p+c)>*(p+c+2)))*(p+c+1)=*(p+c);
        else
if(*(p+c+2)>=*(p+c)&&*(p+c+2)<=*(p+c+1)||(*(p+c+2)<*(p+c)&&*(p+c+2)>*(p+c+1)))*(p+c+1)=*(p+c);
        }
   }
}
float Cal_L_k(void) []，
{
   unsigned char L,Start;
   unsigned char i;
   unsigned char half;
   int sum;
```

```
    sum=0;
    L=L_Count;Start=L_Start;
    if(L%2==0)
        half=L/2;
    else
            half=(L-1)/2;
    if(half<2)return 0;//点过少，认为丢失
        for(i=0;i<half && (i+Start+half)<row_num;i++)
        {
            sum=sum+Lx[i+Start+half]-Lx[i+Start];
        }
        return (float)(sum/((float)(half*i)));
}

float Cal_R_k(void) /{
    unsigned char R,Start;
    unsigned char i;
    unsigned char half;
    int sum;
    sum=0;
    R=R_Count;Start=R_Start;
    if(R%2==0)
        half=R/2;
    else
        half=(R-1)/2;
    if(half<2) return 0;
        for(i=0;i<half && (i+Start+half<row_num);i++)
        {
            sum=sum+Rx[i+Start+half]-Rx[i+Start];
        }
        return (float)(sum/((float)(half*i)));
}
/*float Cal_C_k(void) {
    unsigned char C,Start;
```

```
    unsigned char i;
    int sum;
    sum=0;
    C=dot;Start=C_Start;
    if(C<2) return 0;
        for(i=0;i<(C/2) && (i+Start+(C/2))<row_num;i++)
        {
            sum=sum+center[i+Start+(C/2)]-center[i+Start];
        }
        return (float)(sum/(C/2))/(float)(i);
}*/

float Cal_FixC_k(uchar start,uchar end)//计算修正中线 centerfixed 的斜率
{
    unsigned char i,C;
    int sum;
    sum=0;
    C=end-start+1;
    if(C==1) return 0;
        for(i=0;i<(C/2) &&(i+start+(C/2))<row_num;i++)
        {
            sum=sum+centerfixed[i+start+(C/2)]-centerfixed[i+start];
        }
        return (float)(sum/(C/2))/(float)(i);
}

void NiHeLine(int x)/ {
    unsigned char* xx;
    unsigned char Start,End,i;
    float a0,a1,s1,s2;
    float y_col[row_num],n    float    xhe,xhef,yhe,xfhe,xyhe;
    if(x==1){
         if(L_Count<4||L_Start>20)            {
              L_a0=0;
```

```
            L_a1=0;
            L_s1=0;
            L_s2=0;
            return;
        }
        Start=L_Start;              End=L_Start+L_Count-1;
        xx=Lx;
    }
    else if(x==2){
        if(R_Count<4 || R_Start>20)
        {
            R_a0=0;
            R_a1=0;
            R_s1=0;
            R_s2=0;
            return;
        }
        Start=R_Start;
        End=R_Start+R_Count-1;
        xx=Rx;
    }
    n=End-Start+1;
    xhe=0;xhef=0;yhe=0;
    xfhe=0;xyhe=0;

    for(i=Start;i<=End;i++){
        y_col[i]=(int)xx[i];
        xhe+=i;
        yhe+=y_col[i];
        xfhe+=i*i;                          xyhe+=y_col[i]*i;
       }                                    /   xhef=xhe*xhe;
    a0=(xfhe*yhe-xhe*xyhe)/(n*xfhe-xhef);
    a1=(n*xyhe-xhe*yhe)/(n*xfhe-xhef);
    {
```

```
        s1=0;s2=0;
        for(i=Start;i<=End;i++){
            s2+=(a0+i*a1-y_col[i])*(a0+i*a1-y_col[i]);
            s1+=(a0+i*a1-y_col[i]);
        }
        s2=s2/n;
        s1=s1/n;
    }
    if(x==1)
    {
        L_a0=a0;
        L_a1=a1;
        L_s1=s1;
        L_s2=s2;
    }
    if(x==2)
    {
        R_a0=a0;
        R_a1=a1;
        R_s1=s1;
        R_s2=s2;
    }
}
void NiHeLineCenter(int x)/ {
    unsigned char* xx;
    unsigned char Start,End,i;
    float a0,a1,s1,s2;
    float y_col[row_num],n;    float    xhe,xhef,yhe,xfhe,xyhe;
    if(x==1){

        Start=C_Start;                  End=C_Start+dot-1;
        xx=centerfixed;
    }
    else if(x==2){
```

```
        Start=C_End-dot_anti+1;
        xx=centerfixed;
    }
    n=End-Start+1;
    xhe=0;xhef=0;yhe=0;
    xfhe=0;xyhe=0;

    for(i=Start;i<=End;i++){
        y_col[i]=(int)xx[i];
        xhe+=i;
        yhe+=y_col[i];
        xfhe+=i*i;
        xyhe+=y_col[i]*i;
       }
    xhef=xhe*xhe;
    a0=(xfhe*yhe-xhe*xyhe)/(n*xfhe-xhef);
    a1=(n*xyhe-xhe*yhe)/(n*xfhe-xhef);
    {
       s1=0;s2=0;
       for(i=Start;i<=End;i++){
           s2+=(a0+i*a1-y_col[i])*(a0+i*a1-y_col[i]);
           s1+=(a0+i*a1-y_col[i]);
       }
       s2=s2/n;
       s1=s1/n;
    }

       C_a0=a0;
       C_a1=a1;
       C_s1=s1;
       C_s2=s2;
}
void NiHeLeftAndRight(unsigned char* xx,uchar Start,uchar End,uchar selectNum)
```

```
{
   uchar i;
   float a0,a1,s1,s2;
   float y_col[row_num],n;//n 记录连续的点
   float    xhe,xhef,yhe,xfhe,xyhe;

   n=End-Start+1;
   if(n<2)
   {
      if(selectNum==0)
      {
         L_a0_front=0;
         L_a1_front=0;
         L_s1_front=0;
         L_s2_front=0;
      }else if(selectNum==1)
               {
                  L_a0_back=0;
                  L_a1_back=0;
                  L_s1_back=0;
                  L_s2_back=0;
               } else if(selectNum==2)
                     {
                        R_a0_front=0;
                        R_a1_front=0;
                        R_s1_front=0;
                        R_s2_front=0;
                     }else if(selectNum==3)
                              {
                                 R_a0_back=0;
                                 R_a1_back=0;
                                 R_s1_back=0;
                                 R_s2_back=0;
                              }
```

```
    return ;
}
xhe=0;
xhef=0;
yhe=0;
xfhe=0;
xyhe=0;

for(i=Start;i<=End;i++){
     y_col[i]=(int)xx[i];
     xhe+=i;
     yhe+=y_col[i];
     xfhe+=i*i;
xhef=xhe*xhe;
a0=(xfhe*yhe-xhe*xyhe)/(n*xfhe-xhef);
a1=(n*xyhe-xhe*yhe)/(n*xfhe-xhef);
{
   s1=0;s2=0;
   for(i=Start;i<=End;i++){
        s2+=(a0+i*a1-y_col[i])*(a0+i*a1-y_col[i]);
        s1+=(a0+i*a1-y_col[i]);
   }
   s2=s2/n;
   s1=s1/n;
}
if(selectNum==0)
{
   L_a0_front=a0;
   L_a1_front=a1;
   L_s1_front=s1;
   L_s2_front=s2;
}else if(selectNum==1)
{
   L_a0_back=a0;
```

```
        L_a1_back=a1;
        L_s1_back=s1;
        L_s2_back=s2;
    } else if(selectNum==2)
    {
        R_a0_front=a0;
        R_a1_front=a1;
        R_s1_front=s1;
        R_s2_front=s2;
    }else if(selectNum==3)
    {
        R_a0_back=a0;
        R_a1_back=a1;
        R_s1_back=s1;
        R_s2_back=s2;
    }
}
uchar max_index_right;
void FindMaxRight()
{
    uchar i,j,temp;
    uchar max=Rx[R_Start];
    max_index_right=R_Start;
    for(i=R_Start;i<R_Start+R_Count;i++)
    {
        if(max<Rx[i])
        {
            max=Rx[i];
            max_index_right=i;
        }
    }
    NiHeLeftAndRight(Rx,R_Start,max_index_right,2);          //front
    if(R_Start+R_Count-max_index_right>5)
        NiHeLeftAndRight(Rx,max_index_right,max_index_right+4,3);    //back
```

```
    else
        NiHeLeftAndRight(Rx,max_index_right,R_Start+R_Count-1,3);
}
uchar min_index_left;
void FindMinLeft()
{
    uchar i,j,temp;
    uchar min=Lx[L_Start];
    min_index_left=L_Start;
    for(i=L_Start;i<L_Start+L_Count;i++)
    {
        if(min>Lx[i])
        {
            min=Lx[i];
            min_index_left=i;
        }
    }
    NiHeLeftAndRight(Lx,L_Start,min_index_left,0);
    if(L_Start+L_Count-min_index_left>5)
        NiHeLeftAndRight(Lx,min_index_left,min_index_left+4,1);
    else
        NiHeLeftAndRight(Lx,min_index_left,L_Start+L_Count-1,1);
}
{

     unsigned char temp1=0,temp2=0;
     unsigned char a1,b1,c1;
     signed char    Blackcnt=0;                          /
     LLeftEdge1=-1;
     LLeftEdge2=-1;
     LLeftEdge3=-1;
        if(Lxlast[L_Start]!=col_num && L_Start<15)
     {
       lp1=(Lxlast[L_Start]-offset[L_Start]>0)?Lxlast[L_Start]-offset[L_Start]:0;
```

```
 lp2=lp1+P_WIDTH;
}
else     {
    if(Rxlast[R_Start]!=0)
    {

lp1=(Rxlast[R_Start]+offset[R_Start]>col_num-P_WIDTH)?col_num-P_WIDTH:Rxlast[R_Start]
+offset[R_Start];
        lp2=lp1+P_WIDTH;
    }
    else
    {
       lp1=car_center;
       lp2=lp1+P_WIDTH;
    }
}
//--------------------------
for(;CurL<row_num;CurL++)
{
    LLeftEdge1=-1;
    LLeftEdge2=-1;
    LLeftEdge3=-1;
   p=image+CurL*col_num;     Blackcnt=0;
   if(CurL>0)
   {
       LastL=CurL-1;  //Lx[]=col_num                while(Lx[LastL]==col_num  &&LastL>0)
LastL--;
           if(Lx[LastL]!=col_num)
          if((int)(Lx[LastL]-offset[LastL]-(CurL-LastL)*2)>0)
           lp1=Lx[LastL]-offset[LastL]-(CurL-LastL)*fabs(L_a1);
          else
           lp1=0;
          lp2=lp1+P_WIDTH;
           }
```

```
                else /
                {
                  if(Rx[CurL]!=0)              {          }

        if(isLeftEdge)
              if(CurL==0)
                Lx[CurL]=LLeftEdge-1;                  else
                if(Lx[CurL-1]!=col_num)
                  {
                    if(fabs(Lx[CurL-1]-LLeftEdge)<25)
                      Lx[CurL]=LLeftEdge-1;                        else
                      Lx[CurL]=col_num;
                  }
                else
                   Lx[CurL]=LLeftEdge-1;
            }
        else          Lx[CurL]=col_num;                        isLeftEdge=0;
    }

void ScanLine(void)
{
   uchar i;
   Leftline();
   Rightline();
   for(i=0;i<row_num;i++)
   {
      UnLx[i]=Lx[i];
      UnRx[i]=Rx[i];
   }
}
void DealWithLeftAndRightLine(void)
{
     uchar i;
     uchar CurR,LastR;
```

```
LastR=0;
for(CurR=1;CurR<row_num;CurR++)     {
      if( Lx[LastR]!=col_num && Lx[CurR]!=col_num && fabs(Lx[LastR]-Lx[CurR])>40 )
          Lx[CurR]=col_num;
      if( Rx[LastR]!=0 && Rx[CurR]!=0 && fabs(Rx[LastR]-Rx[CurR])>40 )
          Rx[CurR]=0;
       LastR=CurR;
}
CalLeftDots();     L_Start L_Count L_End L_Count_anti
CalRightDots();
Lk=Cal_L_k();          Rk=Cal_R_k();
Fixline();



/*
if(L_Start+L_Count-1<R_Start && R_Start<10 && R_Count>=5)
{
    for(i=L_Start;i<L_Start+L_Count;i++)
       Lx[i]=col_num;
}
if(R_Start+R_Count-1<L_Start && L_Start<10 && L_Count>=5)
{
    for(i=R_Start;i<R_Start+R_Count;i++)
       Rx[i]=0;
}
*/
/*
for(i=R_Start+R_Count;i<row_num;i++)
   Rx[i]=0;
for(i=L_Start+L_Count;i<row_num;i++)
   Lx[i]=col_num;
*/
/*
if(R_Start>15    && L_Start<10)
```

```
      for(i=R_Start;i<R_Start+R_Count;i++)
         Rx[i]=0;
    if(L_Start>15    && R_Start<10)
      for(i=L_Start;i<L_Start+L_Count;i++)
         Lx[i]=col_num;
    */

    CalLeftDots();    i
    CalRightDots();
    NiHeLine(1);            NiHeLine(2);
    FindMaxRight();
    FindMinLeft();
}

void GetCenter(void)
{
   CenterLine();
}

void DealWithCenter(void)
{
   unsigned int sum=0,pnum=0,twolnum;
   unsigned int j1,j2,j3;
   unsigned char i;
    FixCenter();
   CalCenterDots( );
   Ck=Cal_FixC_k(C_Start,C_Start+dot-1);
   if(dot_anti>=dot+dot)
         NiHeLineCenter(2);
   else
         NiHeLineCenter(1);
   if(Cur_street!=CROSSROAD_LEFT          &&          Cur_street!=CROSSROAD_RIGHT
&&Cur_street!=CROSSROAD)
   {
```

```
        if(L_a1!=0 && R_a1!=0 )
        {
           if((L_a1<0 && R_a1<0 && R_a1-L_a1<0) ||(L_a1>0 && R_a1>0 &&L_a1-R_a1>0));
else
           if(L_Start<=20 && R_Start<=20)
           {
              C_a1=(L_a1+R_a1)/2;
              C_a0=(L_a0+R_a0)/2;
           } else
           if(L_Start<=10)
           {
              C_a1=L_a1*0.618;
              C_a0=L_a0-car_center;
           } else
           {
              C_a1=R_a1*0.618;
              C_a0=R_a0+car_center;
           }
        }
        else
           if(L_a1==0)
           {
              C_a1=R_a1*0.618;
              C_a0=R_a0+car_center;
           }
           else
           {
              C_a1=L_a1*0.618;
              C_a0=L_a0-car_center;
           }
     }
     else
     {
```

```
        if(Cur_street==CROSSROAD_RIGHT)     //十字弯右转
        {
            if(R_a1==0)
            {
               L_a1=L_a1_front;
               L_a0=L_a0_front;
               if(fabs(C_a1)-fabs(L_a1*0.618)<0.3)
               {
                  C_a1=L_a1_front*0.618;
                  C_a0=L_a0_front-car_center;
               }
            }
            else
            {
               L_a1=L_a1_front;
               L_a0=L_a0_front;
               if((L_a1<0 && R_a1<0 && R_a1-L_a1<0) ||(L_a1>0 && R_a1>0
&&L_a1-R_a1>0));
               else
               {
                  if(fabs(C_a1)-fabs((L_a1_front+R_a1)/2)<0.3)
                  {
                     C_a1=(L_a1_front+R_a1)/2;
                     C_a0=(L_a0_front+R_a0)/2;
                  }
               }
            }
        }
        if(Cur_street==CROSSROAD_LEFT) //十字弯左转
        {
            if(L_a1==0)
            {
               R_a1=R_a1_front;
               R_a0=R_a0_front;
```

```
                if(fabs(C_a1)-fabs(R_a1*0.618)<0.3)
                {
                  C_a1=R_a1_front*0.618;
                  C_a0=R_a0_front+car_center;
                }
              }
              else
              {
                R_a1=R_a1_front;
                R_a0=R_a0_front;
                if((L_a1<0 && R_a1<0 && R_a1-L_a1<0) ||(L_a1>0 && R_a1>0
&&L_a1-R_a1>0))                 else
                {
                  if(fabs(C_a1)-fabs((R_a1_front+L_a1)/2)<0.3)
                  {
                    C_a1=(R_a1_front+L_a1)/2;
                    C_a0=(R_a0_front+L_a0)/2;
                  }

                }
              }
        }
  }

  switch(Cur_street)
  {
    case SMOOTH_S:
    pnum=(C_Start+dot-1)/3;
    if(pnum==0)break;
          for(i=0,j1=0;i<pnum;i++){
                j1+=centerfixed[i];
          }
          j1=j1/pnum;
          for(i=pnum,j2=0;i<2*pnum;i++){
```

```
            j2+=centerfixed[i] ;
        }
        j2=j2/pnum;
        for(i=2*pnum,j3=0;i<3*pnum;i++){
            j3+=centerfixed[i] ;
        }
        j3=j3/pnum;
        sum=(j1+j2+j3)/3;
        j1=(j1+sum)/2;
        j2=(j2+sum)/2;
        j3=(j3+sum)/2;

        for(i=0;i<pnum;i++){
            centerfixed[i]=j1;
        }
        for(i=pnum;i<2*pnum;i++){
            centerfixed[i]=j2;
        }
        for(i=2*pnum;i<3*pnum;i++){
            centerfixed[i]=j3;
        }

    break;
    default:
    break;
  }
}

void Fixline(void)//2011-12-2
{
  uchar CurR,LastR,whiteCount=0;
  uchar *p,lp1,lp2,i;
  uchar isRight=0,isLeft=0;
  LastR=0;
```

```
for(CurR=1;CurR<row_num;CurR++)
{
    if( Lx[LastR]!=col_num && Lx[CurR]!=col_num && fabs(Lx[LastR]-Lx[CurR])>40 )
        Lx[CurR]=col_num;
    if( Rx[LastR]!=0 && Rx[CurR]!=0 && fabs(Rx[LastR]-Rx[CurR])>40 )
        Rx[CurR]=0;
     LastR=CurR;
}



        i=0;
   while(Lx[i]==col_num && i<row_num)i++;
   if(i<5 && Lx[i]!=col_num)       {
      lp1=Lx[i];
      p=image+i*col_num;/
      while(*(p+lp1)>whiteRoad && lp1>0 && whiteCount<=10)
      {
         whiteCount++;
         lp1--;
      }
      if(whiteCount>10 && Lk<0 && Lx[i]<40)        /              isRight=1;                }

   i=0;
   while(Rx[i]==0 && i<row_num)i++;
   if(i<5 && Rx[i]!=0)       {
      lp1=Rx[i];
      p=image+i*col_num;//p
      while(*(p+lp1)>whiteRoad && lp1<col_num)
          whiteCount++;
          lp1++;
      }
      if(whiteCount>10 && Rk>0 && Rx[i]>100)
          isLeft=1;
```

```
    for(CurR=0;CurR<row_num;CurR++)
    {
      if(Rx[CurR]>Lx[CurR] || fabs(Rx[CurR]-Lx[CurR])<15)        {
        if(isLeft    ||    (Rk>=0    &&    Lk>=0)    ||    Last_street==C_TURN_LEFT
||Last_street==SMALL_TURN_LEFT                Lx[CurR]=col_num;
        else    if(isRight    ||(Rk<=0    &&    Lk<=0)    ||Last_street==C_TURN_RIGHT
||Last_street==SMALL_TURN_RIGHT)        //                Rx[CurR]=0;
      }
    }

}//Fixline 结束


void CenterLine(void)/
    int CurL,i,LastL,LaterL;
  float chazhi;
    for(CurL=0;CurL<row_num;CurL++)
    {
        if(Lx[CurL]!=col_num && Rx[CurL]!=0)/        {
            center[CurL]=(Lx[CurL]+Rx[CurL])>>1;
            if(center[CurL]==0)
                center[CurL]=-1;
        }

        if(Lx[CurL]!=col_num && Rx[CurL]==0)//当只有左引导线
        {
            //******4.15********

                    center[CurL]=Lx[CurL]-DistanceL[CurL]-4;
                else
                    center[CurL]=-1;
            //计算 chazhi
        }
        if(Lx[CurL]==col_num && Rx[CurL]!=0
```

```
            {
                center[CurL]=Rx[CurL]+DistanceR[CurL];
            }

            if(Lx[CurL]==col_num && Rx[CurL]==0)/        {
          center[CurL]=0;
            }
       }
}//CenterLine()
//搜寻起跑线

void FixCenter(void)
{
    unsigned char CurR,LastL ;//记录当前行,和上一行
    LastL=0;
    centerfixed[0]=center[0];
    for(CurR=1;CurR<row_num;CurR++)
    {
        if(center[CurR]!=0  &&  center[LastL]!=0  &&  fabs(center[CurR]-center[LastL])>20  )
//2.25 10->15 2.27:13->20
        {
            centerfixed[CurR]=0;
        }
        else
        {
           centerfixed[CurR]=center[CurR];//
        }
        LastL=CurR;
    }
}

unsigned char FindMax(uchar start,uchar end)
{
   int max,i;
```

```
    unsigned char index;
    max=centerfixed[start];
    index=start;
    for(i=0;i<(end-start+1) && (start+i)<=end;i++)
    {
        if(max<centerfixed[start+i])
        {
            max=centerfixed[start+i];
            index=start+i
        }
    }
    return index;
}
unsigned char FindMin(uchar start,uchar end)
{
    int min,i;
    unsigned char index;
    min=centerfixed[start];
    index=start;
    for(i=0;i<(end-start+1) && (start+i)<=end;i++)
    {
        if(min>centerfixed[start+i])
        {
            min=centerfixed[start+i];
            index=start+i;
        }
    }
    return index;
}
void CalCenterDots(void)
{
            C_Start=0;    //center
            C_End=row_num-1;
            dot=0;
```

```
        dot_anti=0;

        while(centerfixed[C_Start]==0 && C_Start<row_num-1)C_Start++;
        while(centerfixed[C_End]==0 && C_End>0)C_End--;

        while(centerfixed[dot+C_Start]!=0 && dot<row_num-C_Start)dot++;
while(centerfixed[C_End-dot_anti]!=0 && C_End-dot_anti>=0)dot_anti++;

        //***4.16***"<=2" -> "<2"
        if(dot<2 && C_Start<row_num/2)
        {
          C_Start+=dot;
          dot=0;

          while(centerfixed[C_Start]==0 && C_Start<row_num-1)C_Start++;
          while(centerfixed[dot+C_Start]!=0  &&  dot<row_num-C_Start)dot++;//找到丢失黑线
的第一个点，用前面的点来控制舵机
        }
        if(dot_anti<2 && C_End>row_num/2)
        {
          C_End-=dot_anti;
          dot_anti=0;

          while(centerfixed[C_End]==0 && C_End>0)C_End--;
          while(centerfixed[C_End-dot_anti]!=0 && C_End-dot_anti>=0)dot_anti++;
        }
}
void CalLeftDots(void)
{
    L_Start=0;
    L_End=row_num-1;
    L_Count=0;
    L_Count_anti=0;
```

```
    while(Lx[L_Start]==col_num && L_Start<row_num-1) L_Start++;
    while(Lx[L_End]==col_num && L_End>0)L_End--;

    while(Lx[L_Start+L_Count]!=col_num && L_Start+L_Count<row_num)L_Count++;
    while(Lx[L_End-L_Count_anti]!=col_num && L_End-L_Count_anti>=0 )L_Count_anti++;

    if(L_Count<2 && L_Start<(row_num)/2)
    {
        Lx[L_Start]=col_num;            L_Start=L_Start+L_Count;
        L_Count=0;
        while(Lx[L_Start]==col_num && L_Start<row_num-1) L_Start++;
        while(Lx[L_Start+L_Count]!=col_num && L_Start+L_Count<row_num)L_Count++;
    }
    if(L_Count_anti<2 && L_End>row_num/2) //后半段连续点数少于 2,重新找
    {
      L_End=L_End-L_Count_anti;
      L_Count_anti=0;

      while(Lx[L_End]==col_num && L_End>0)L_End--;
      while(Lx[L_End-L_Count_anti]!=col_num                    &&
L_End-L_Count_anti>=0)L_Count_anti++;
    }

}
void CalRightDots(void)
{
    R_Start=0;
    R_End=row_num-1;
    R_Count=0;
    R_Count_anti=0;

    while(Rx[R_Start]==0 && R_Start<row_num-1) R_Start++;
    while(Rx[R_End]==0 && R_End>0)R_End--;
```

```
    while(Rx[R_Start+R_Count]!=0 && R_Start+R_Count<row_num)R_Count++;
    while(Rx[R_End-R_Count_anti]!=0 && R_End-R_Count_anti>=0 )R_Count_anti++;

    if(R_Count<2 && R_Start<row_num/2)
    {
        Rx[R_Start]=0;//清除那//清除那个单独的点   R_Start=R_Start+R_Count;
        R_Count=0;
        while(Rx[R_Start]==0 && R_Start<row_num-1) R_Start++;
        while(Rx[R_Start+R_Count]!=0 && R_Start+R_Count<row_num)R_Count++;
    }
    if(R_Count_anti<2 && R_End>row_num/2)
    {
      R_End=R_End-R_Count_anti;
      R_Count_anti=0;

      while(Rx[R_End]==0 && R_End>0)R_End--;
      while(Rx[R_End-R_Count_anti]!=0 && R_End-R_Count_anti>=0)R_Count_anti++;
    }
}

void SetAimSpeed(void)
{
  if(PORTB_PB0==0)
  {
      Mp=17;
      Mi=8;
      Md=3;
      steer_p_near=2;                //对 P 值赋初值
      steer_p_mid=2;                  //之前是 5;
      steer_p_far=3;
      SteerKd=0.5;
      C_TURN_RIGHT_D=50;
      C_TURN_LEFT_D=50;
      CROSSROAD_RIGHT_D=85;
```

```
    CROSSROAD_LEFT_D=85;
    DEFAULT_D=30;

    SLOW_SPEED=50;
    LOSE_LINE_SPEED=50;
    STRIGHT_SPEED=80;
    C_TURN_SPEED=55;
    S_SPEED=75;
    DEAFUALT_SPEED=65;
}
else if(PORTB_PB1==0)
{
    Mp=17;
    Mi=8;
    Md=3;
    steer_p_near=2.5;                //对 P 值赋初值
    steer_p_mid=2;                   //之前是 5;
    steer_p_far=2.5;
    SteerKd=0.618;
    C_TURN_RIGHT_D=62;
    C_TURN_LEFT_D=62;
    CROSSROAD_RIGHT_D=80;
    CROSSROAD_LEFT_D=80;
    DEFAULT_D=55;

    SLOW_SPEED=50;
    LOSE_LINE_SPEED=50;
    STRIGHT_SPEED=70;
    C_TURN_SPEED=65;
    S_SPEED=65;
    DEAFUALT_SPEED=65;

}
        switch(Cur_street)
```

```
{
   case C_TURN_RIGHT:
      STEERdeviation=C_TURN_RIGHT_D;
   break;
   case C_TURN_LEFT:
      STEERdeviation=C_TURN_LEFT_D;
   break;

   case CROSSROAD_RIGHT:
      STEERdeviation=CROSSROAD_RIGHT_D;
   break;
   case CROSSROAD_LEFT:
      STEERdeviation=CROSSROAD_LEFT_D;
   break;
   default:
      STEERdeviation=DEFAULT_D;
   break;
}

if(Cur_street==STRIGHT && Far_street==C_TURN)
{
   AIMSPEED=SLOW_SPEED;
}
else
   switch(Cur_street)
   {
      case LOSE_LINE:
         AIMSPEED=LOSE_LINE_SPEED;
         break;
      case STRIGHT:
         AIMSPEED=STRIGHT_SPEED;
         break;
      case C_TURN:
      case C_TURN_LEFT:
```

```
            case C_TURN_RIGHT:\
               AIMSPEED=C_TURN_SPEED;
               break;
            case SMOOTH_S:
               AIMSPEED=S_SPEED;
            break;
            default:
               AIMSPEED=DEAFUALT_SPEED;
            break;
         }

if(PORTB_PB5==0)//无线调试
{

         Mp=(RecBuf0[0]*256+RecBuf0[1])/100;
         Mi=(RecBuf0[2]*256+RecBuf0[3])/100;
         Md=(RecBuf0[4]*256+RecBuf0[5])/100;
         steer_p_near=(float)(RecBuf0[6]*256+RecBuf0[7])/100.0;
         steer_p_mid=(float)(RecBuf0[8]*256+RecBuf0[9])/100.0;
         steer_p_far=(float)(RecBuf0[10]*256+RecBuf0[11])/100.0;
         SteerKd=(float)(RecBuf0[12]*256+RecBuf0[13])/100.0;
         switch(Cur_street)
         {
            case C_TURN_RIGHT:
               STEERdeviation= (RecBuf0[14]*256+RecBuf0[15])/100;
            break;
            case C_TURN_LEFT:
               STEERdeviation= (RecBuf0[16]*256+RecBuf0[17])/100 ;
            break;

            case CROSSROAD_RIGHT:
               STEERdeviation= (RecBuf0[18]*256+RecBuf0[19])/100 ;
            break;
            case CROSSROAD_LEFT:
```

```
        STEERdeviation= (RecBuf0[20]*256+RecBuf0[21])/100 ;
    break;
    default:
        STEERdeviation= (RecBuf0[22]*256+RecBuf0[23])/100 ;
    break;
}

if(Cur_street==STRIGHT && Far_street==C_TURN)
{
    AIMSPEED= (RecBuf0[24]*256+RecBuf0[25])/100;
}
else
    switch(Cur_street)
    {
        case LOSE_LINE:
            AIMSPEED= (RecBuf0[26]*256+RecBuf0[27])/100;
            break;
        case STRIGHT:
            AIMSPEED= (RecBuf0[28]*256+RecBuf0[29])/100;
            break;
        case C_TURN:
        case C_TURN_LEFT:
        case C_TURN_RIGHT:
            AIMSPEED=    (RecBuf0[30]*256+RecBuf0[31])/100;
            break;
        case SMOOTH_S:
            AIMSPEED= (RecBuf0[32]*256+RecBuf0[33])/100;
        break;

        case OTHERSTREET:
            AIMSPEED= (RecBuf0[34]*256+RecBuf0[35])/100;
        break;

        default:
```

```
                AIMSPEED= (RecBuf0[36]*256+RecBuf0[37])/100;
            }
            if(RecBuf0[39]>=0 &&RecBuf0[39]<=3)
              Car_state=RecBuf0[39];
    }
}
void SetSteer(void)
{
    // PWMDTY45=pre_steer_out=0.888*steer_out+0.112*pre_steer_out;
    PWMDTY45=steer_out;
}

void speed_PID(void){
  Cur_Speed=PACNT;
  Cur_Speed=Cur_Speed/4;
  PACNT =0;
  //---------不完全微分------------------

  //-----------一般 PID--------------
  if(is_anti_motor && Cur_Speed)
  {

  }
      MotorE2=MotorE1;
      MotorE1=MotorE0;
      MotorE0=AIMSPEED-Cur_Speed;

motor_out+=Mi*MotorE0+Mp*(MotorE0-MotorE1)+Md*(MotorE0-2*MotorE1+MotorE2);
      if(motor_out>motor_right) motor_out=motor_right;
      if(motor_out<motor_left) motor_out=motor_left;
      PWMDTY01=motor_out;
      PWMDTY23=0;

}
```

```
unsigned int stoptime=0;
void protact()
{
   if(Cur_Speed<25)
        stoptime++;
   else
      stoptime=0;
   if(stoptime>12)
      Car_state=STOP;
}
uint time1,time2,time3;
void final_control(void)
{
   int temp=0,i;
   switch( Car_state)
   {
      case START:
      {
                  if(CCD_state==GETVSYNC )
               {

                        if(VSync%2==0)
                           {
                           time1=TCNT;
                           //quzaozongxiang();

                           ScanLine();
                           DealWithLeftAndRightLine();
                           CheckStreetType();
                           DealwithTheStreet();
                           GetCenter();
                           DealWithCenter();
                           CheckFarStreet();
                           SetAimSpeed();
```

```
                steer_PD();
                for(i=0;i<row_num;i++)
                {
                   Lxlast[i]=Lx[i];
                   Rxlast[i]=Rx[i];
                }
                protact();
                //speed_PID();
                 if(All_VSync>500        &&        PORTB_PB7==0)//20ms*200=4000ms
{
                      SearchEnd();
                   }

                CCD_state=WAITVSYNC;
                //----------LED----------

                Count_Num=Cur_Speed;
                LedData[0] = Count_Num/1000%10;
                LedData[1] = Count_Num/100%10;
                LedData[2] = Count_Num/10%10;
                LedData[3] = Count_Num%10;
                PTT= 0x01 << LedNum ;
                PT1AD0= LedCode[LedData[LedNum]];
                LedNum++;
                if(LedNum >= 4) LedNum = 0;

                time2=TCNT-time1;//time2，
                if(PORTB_PB5==0)
                {
                WriteToSCI(0,0x35);
                WriteToSCI(0,0xA4);
                WriteToSCI(0,0);
                WriteToSCI(0,AIMSPEED);
                WriteToSCI(0,0);
```

```
                    WriteToSCI(0,Cur_Speed);
                    WriteToSCI(0,steer_out/256);
                    WriteToSCI(0,steer_out%256);
                    WriteToSCI(0,0);
                    WriteToSCI(0,Cur_street);
                    }
                    if(SDReady)
                    {

                      DateToSD();
                      if(Data_Write(row_num,col_num,S12BUFFER,10)==1 )//1:
                        SDError++;
                      if(SDError>10)PORTE_PE3=0;
                      time3=TCNT-time1;//time3

                    }
                }
            }

            break;
    }// end of case START

case INIT:
{
    for(temp=0;temp<380;temp++)
    {
        delay10us(100);
    }
    Car_state=START;
    break;
}
case STOP:
{
    PWMDTY01=0;
```

```
PWMDTY23=0;
ScanLine();
DealWithLeftAndRightLine();
CheckStreetType();
DealwithTheStreet();
GetCenter();
DealWithCenter();
CheckFarStreet();
SetAimSpeed();
steer_PD();
if(PORTB_PB5==0)
        {
        WriteToSCI(0,0x35);
        WriteToSCI(0,0xA4);
        WriteToSCI(0,0);
        WriteToSCI(0,AIMSPEED);
        WriteToSCI(0,0);
        WriteToSCI(0,Cur_Speed);
        WriteToSCI(0, steer_out/256);
        WriteToSCI(0,steer_out%256);
        WriteToSCI(0,0);
        WriteToSCI(0,Cur_street);
        }
if(SDReady)
    {
        DateToSD();
        if(Data_Write(row_num,col_num,S12BUFFER,10)==1 )//1
          SDError++;
        if(SDError>10)PORTA_PA5=0;
        time3=TCNT-time1
    }
Count_Num=Cur_Speed;
        LedData[0] = Count_Num/1000%10;
        LedData[1] = Count_Num/100%10;
```

```
                        LedData[2] = Count_Num/10%10;
                        LedData[3] = Count_Num%10;
                        PTT= 0x01 << LedNum ;
                        PT1AD0= LedCode[LedData[LedNum]];
                        LedNum++;
                        if(LedNum >= 4) LedNum = 0;
            break;
        }
    }//end of switch
}


int oi;
void main(void) {
  /* put your own code here */
  int temp=0;
  DisableInterrupts;
  init();
  Initial_SDCard();
  EnableInterrupts;
  DateToSD();
  for(;;) {
    final_control();

  } /* wait forever */
  /* please make sure that you never leave this function */
}


#pragma CODE_SEG __NEAR_SEG NON_BANKED

void interrupt 7 RTI_INT(void) {

  speed_PID();
```

```
    //-------------------------------------------
    CRGFLG=0x80;
}

interrupt 24 void V_SYNC(void)
{
    //unsigned char temp=0;
    unsigned char *p;
    //--------------------------------
    if (PIFJ_PIFJ7==1)
    {
        PIEJ_PIEJ6=0;
        if(Car_state==START)
            speed_PID();
        image_v_num=0;
        v_counter=0;
        c=0;
        CCD_state=GETVSYNC;
        All_VSync++;
        VSync++;
        VSync=VSync%2;
        PIFJ_PIFJ7=1;
        if(VSync%2==0)
            PIEJ_PIEJ6=0;
        else
            PIEJ_PIEJ6=1;


    }
    else if (PIFJ_PIFJ6==1)
    {
      uchar temp=0;
      image_v_num++;
      if(v_counter<row_num)
```

```
        {
          c=0;//计数列数
          //temp=0;
          if(image_v_num==selectRow[v_counter])
          {
            p=image+v_counter*col_num;
            //while(temp<80)temp++;
            while (c<col_num)
            {
              *(p+c)=PORTA;
              c++;
            }
            v_counter++;


          }
          PIFJ_PIFJ6=1;
        }
        /*else
        {
          PIEJ_PIEJ6=0;
          PIEJ_PIEJ7=0;
          quzaozongxiang();
          PIEJ_PIEJ6=1;
          PIEJ_PIEJ7=1;
        } */

}
}
#pragma CODE_SEG DEFAULT
```