# 第十一届“恩智浦”杯全国大学生智能汽车竞赛

# 技术报告


杭州電子科技大学

HANGZHOU DIANZI UNIVERSITY

**学　　校：** 杭州电子科技大学

**队伍名称：** 杭电摄像头2队

**参赛队员：** 姚　鑫

汪商炯

王彬杰

**带队老师：** 陈　龙　曾　毓

# 关于技术报告和研究论文使用授权的说明

本人完全了解第十一届“恩智浦”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，比赛组委会和恩智浦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：姚　鑫、汪商炯、王彬杰

带队教师签名：陈　龙、曾　毓

日　　期：2016/08/15

# 关于技术报告和研究论文使用授权的说明

本人完全了解第十一届“恩智浦”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，比赛组委会和恩智浦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：姚　鑫、汪商炯、王彬杰

带队教师签名：陈　龙、曾　毓

日　　期：2016/08/15

# 摘要

本文介绍了杭州电子科技大学的队员们在准备此次比赛中的成果。本次比赛采用大赛组委会提供的1:16仿真车模，硬件平台采用带MK60DN512ZVLQ10单片机的K60环境，软件开发平台为Keil 5。

文中介绍了本次我们的智能车控制系统软硬件结构和开发流程，整个智能车涉及车模机械调整，传感器选型，电路设计，控制算法优化等多方面。整辆车的工作原理是在小车的控制周期中，由CCD摄像头采集赛道信息至主板进行预处理并递送分离后的信息到单片机，再由单片机读取信号进行分析处理，再由程序对赛道信息进行提取并选择最佳路径，通过对电机的精确控制从而实现小车在赛道上的高速行进。

为了进一步提高小车在运行时的稳定性和速度，我们组在软件方面使用了多套方案进行比较。使用了SD卡实时存储赛道信息。硬件上为了稳定的考虑，采用了以前比较稳定的方案，但是在电源部分做了调整，使得整车的电源裕度更大，硬件鲁棒性更强。为更好的分析调车数据，我们继承并且改进上届的上位机，用C#软件编写了新的上位机程序来进行车模调试，很大程度上提高了调车效率。在进行大量的实践之后，表明我们的系统设计方案完全是可行的。

关键字：飞思卡尔智能车,MK60DN512ZVLQ10,CCD 摄像头,PID 控制，存储器,C#上位机

## Abstract

This paper introduces the Hangzhou Dianzi university players in preparation for the achievements in this match.The game USES the competition committee provided for simulation models, MK60DN512ZVLQ10 microcontroller hardware platform using belt K60 environment, software platform for the Keil development environment.

This paper introduces the our smart car control system hardware and software structure and development process, the entire intelligent car involves the mechanical adjustment models, sensor selection, signal processing circuit design, control algorithm optimization and many other aspects.The working principle of the whole car is to extract the corresponding control cycle of car time slice, the corresponding time slice is used to control the balance of the body, leaving time slice is used to control speed and steering, from CCD camera in the track information to the mainboard LM1881 signal separation, and delivery information of the separated into single chip microcomputer, again by MCU read signal analysis and processing, using our own software program to extract and choose the best path of the track information, through the accurate control of the motor so as to realize the car on the track fascinating pretty fast!

In order to further improve the stability and speed of the car at run time, our group in the aspect of software USES the multiple sets of scheme comparison.Update the SD card technology real-time information storage track.Hardware to stabilize, the relatively stable solution before, but in the power part of the adjustment, makes the vehicle power supply margin larger, stronger robustness hardware.For a better analysis of the shunting data, we inherit and improve the previous PC, a new PC software written in c # program to adjustment of the models, greatly improve the efficiency of shunting.After a lot of practice, it shows that our system design scheme is feasible completely.



**Key words:** car NXP intelligence, MK60DN512ZVLQ10, CCD camera, PID control, memory, the c # upper machine,

# 目录

# 第一章. 引言

前期我们花了大量的时间去选择车模的架法，摄像头的选择，硬件方案的实验，大体上确定了车模的机械架法和硬件上的芯片选型，PCB布局等。中期是以软件调试为主，同时硬件也在完善，特别是在机械上我们花了很多心思，很多配件都是由自己设计制作的，实践证明在机械上的优化对软件的调车有很大的推动作用。后期软件算法已经确定，硬件上也很稳定，所以我们在原先的机械思路上更近一步，增加了主动悬挂。正是这种对智能车精益求精的追求，使得我们的智能车有了明显的提速。

在本文报告内容框架安排上，第一章是引言，主要对车模设计和报告的架构进行了概述；第二章是系统硬件电路的设计，详细介绍了电源部分、电机驱动等电路模块的设计思路及方案。第三章是机械架构调整，着重分析了摄像头的新式架法和电池的驾法，同时也对小车机械的整体布局和架车理念做了细致的解释和说明。第四章是软件系统设计，主要介绍了图像处理，车身打角控制和速度控制的模块编写，整体的模块调试和上位机的开发。第五章是开发智能车调试过程的说明，讲述了我们自己在调车训练时的方法和训练工具。第六章为车模的技术参数说明。第七章为总结部分。

# 第二章. 系统硬件电路设计

## 2.1. 硬件电路整体架构框图

我们的智能车硬件系统主要包括以下模块：电源管理模块、摄像头模块、图像处理模块、场行分离、电机驱动模块及测速模块、调试模块、电源管理模块、MK60DN512ZVLQ10最小系统。

智能车硬件电路整体结构框图，如图2.1所示。

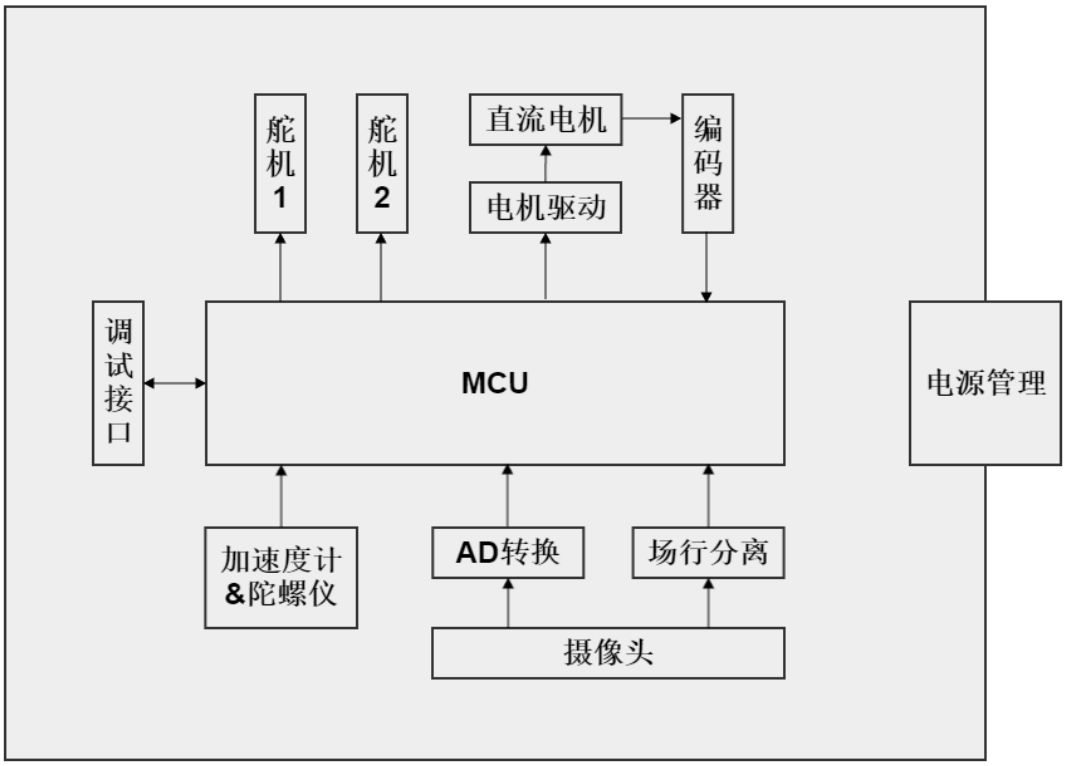



图 2.1　　硬件电路整体架构框图

## 2.2. 单片机外围电路设计

为了更好的配合程序，以及日常调试，我们自己设计了单片机外围，外围电路包括单片机时钟与复位电路，串口及供电电路，并且集成到主板上，以下是MCU部分的PCB图：

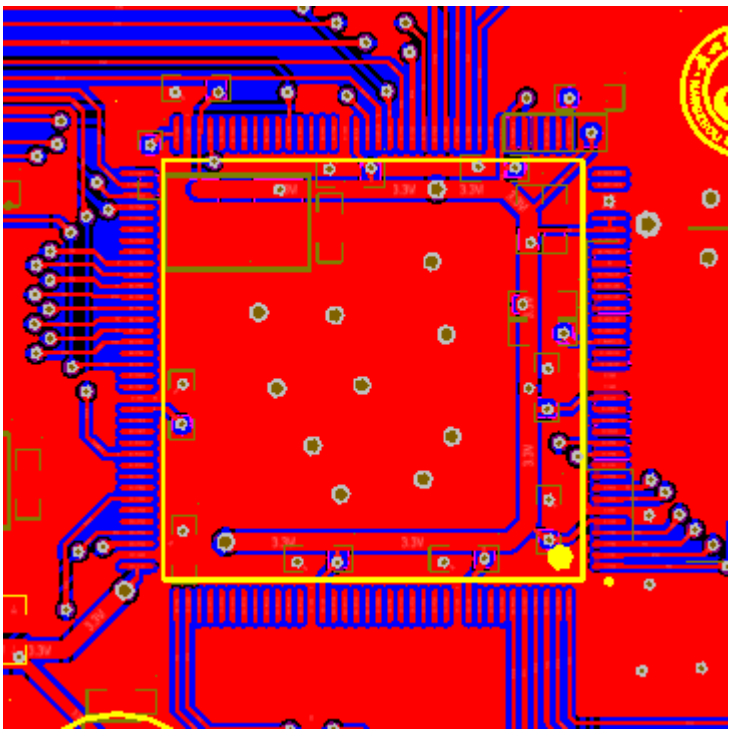


图 2.2 MCU PCB 图

## 2.3. 视频信号处理电路设计

我们采用Sony CCD摄像头。摄像头输出的模拟视频信号，首先通过场行信号分离芯片LM1881搭建的电路分离出场同步信号和行同步信号，并输入单片机进行处理，如图2.3.1所示。

我们使用了8位20M高速并行ADC(TLC5510)进行采样，由单片机软件二值化，实现动态阈值，如图2.3.2所示。

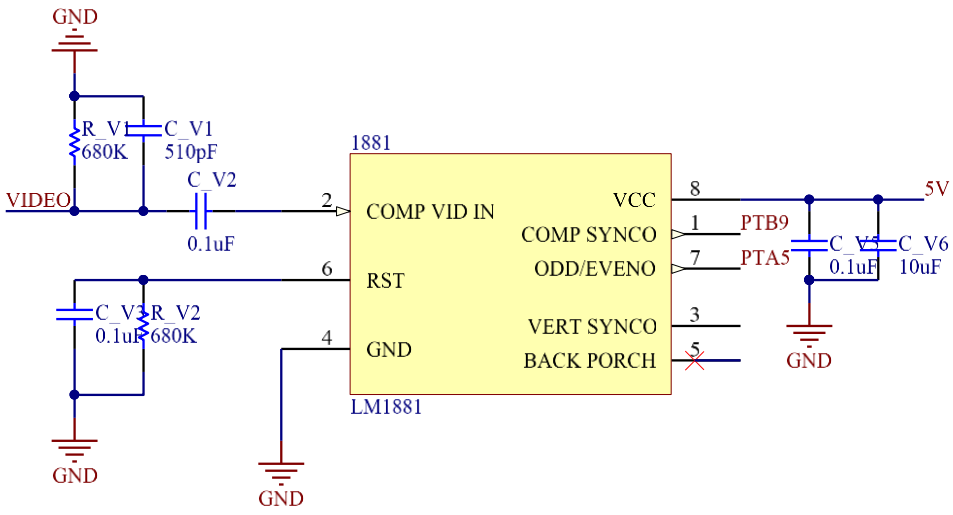



图 2.3.1　视频信号同步分离及处理

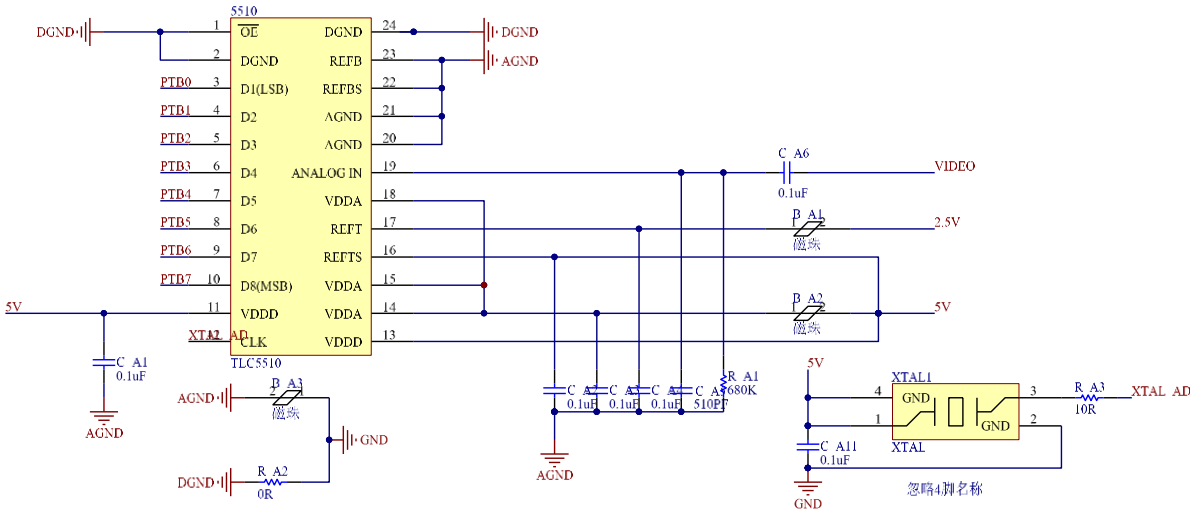



图 2.3.2　高速并行 ADC 采样电路图

## 2.4. 电源部分电路设计

电源分配框图：

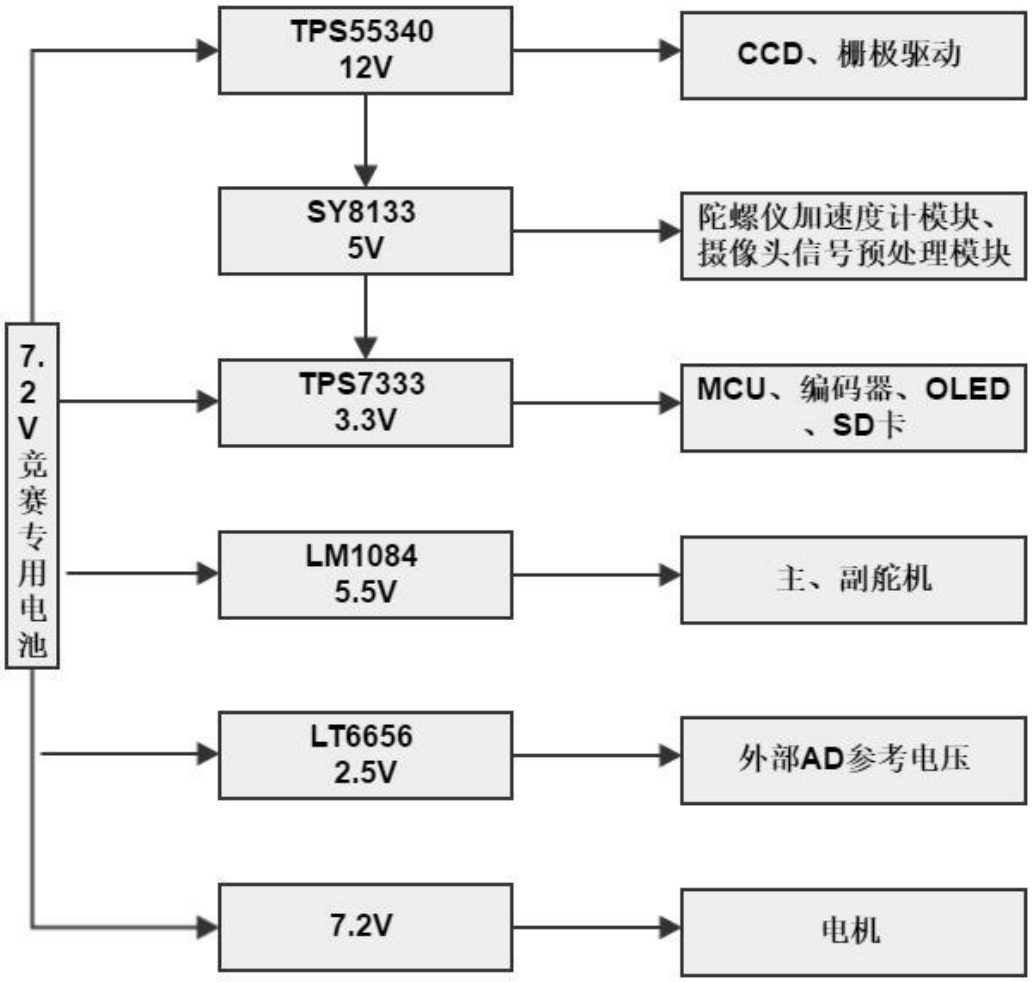



图 2.4 电源分配框图

整辆小车都是由 2000mAh 的镍镉电池提供，但是各种元器件所需的电压不同，这就需要我们用不同的电源芯片来生成不同的电压。

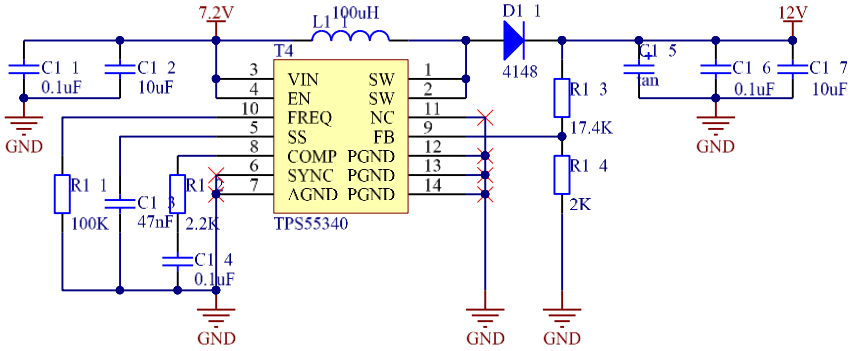



图 2.5.1　12V 电源电路原理图

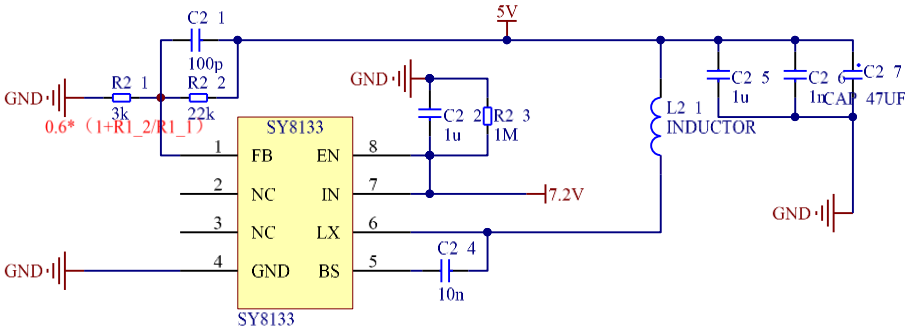



图 2.5.2　5V 电源电路原理图

## 2.5. 电机驱动电路设计

对于电机驱动电路，可有多种选择，像专用电机集成驱动芯片 MC33886,BTN7971，电调，MOS 管 H 桥。MC33886 使用方便，但内阻较大，电调驱动能力最好，但价格太高。MOS 导通电阻小，响应快，在本届驱动电路的设计中我们选用英飞凌 IR7843H 桥驱动电路来驱动电机。该芯片驱动能力较好，电路简洁，同时能够满足车模运行的需要。

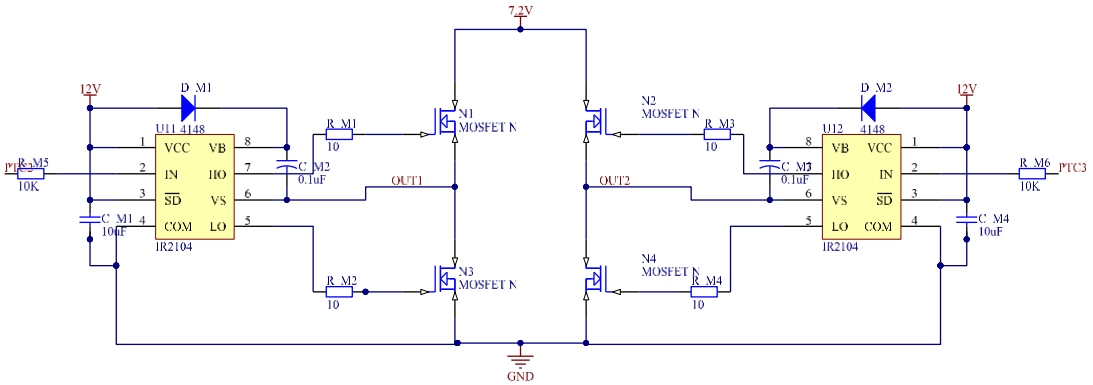



图 2.5　驱动原理图

## 2.6. 小车调试模块

为使软件上参数调节的方便，且兼顾硬件上的简洁与美观，我们特意将这部分设计在主板上，而且使用OLED显示模块来代替之前的NOKIA5110。OLED比NOKIA5110体积上更小，分辨率更高，功耗低。调试中，这个模块对参数的调试带来了极大的方便，可以减少烧录程序的次数提高调试效率。

## 2.7. 主板板级设计

由于需要高速处理 CCD 的模拟信号，PCB 的设计影响信号的完整性，所以需要合理的设计元器件的布局及走线，阻抗匹配。特别注意模拟信号与高速数字信号、电感，晶振之间的布局。

考虑到车子的机械结构，我们根据车模外形绘制了 PCB 的外形，并规划了接插件的位置。设计的 PCB 如图 2.9 所示

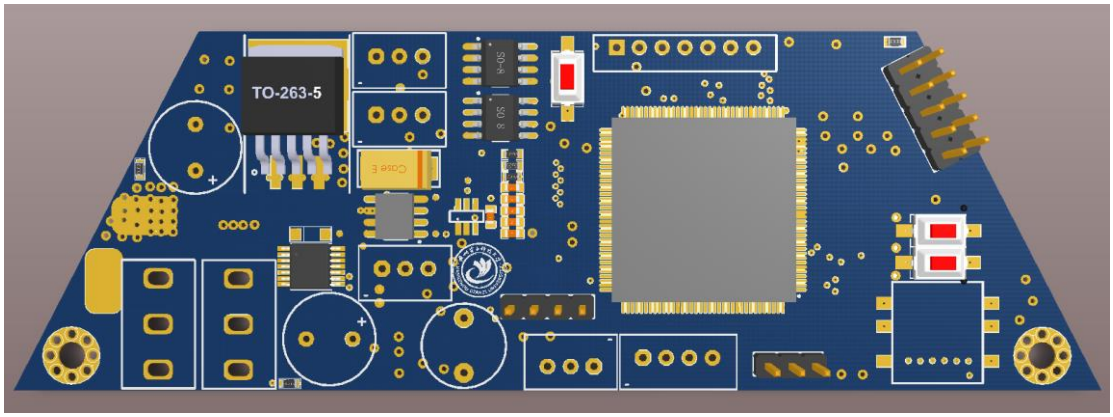



图 2.9 主板电路 PCB 图

## 2.8. 硬件电路部分总结

对于硬件电路部分，一定要做的足够稳定，这对于整个系统的稳定性占着至关重要的位置。我们在实验过程中，先分模块根据经典应用设计制作电路，确保实现功能和稳定性后，再将所有模块组合起来绘制PCB，再次制作电路板验证电路系统的可靠性，最后打样制作电路板。

# 第三章. 机械架构调整

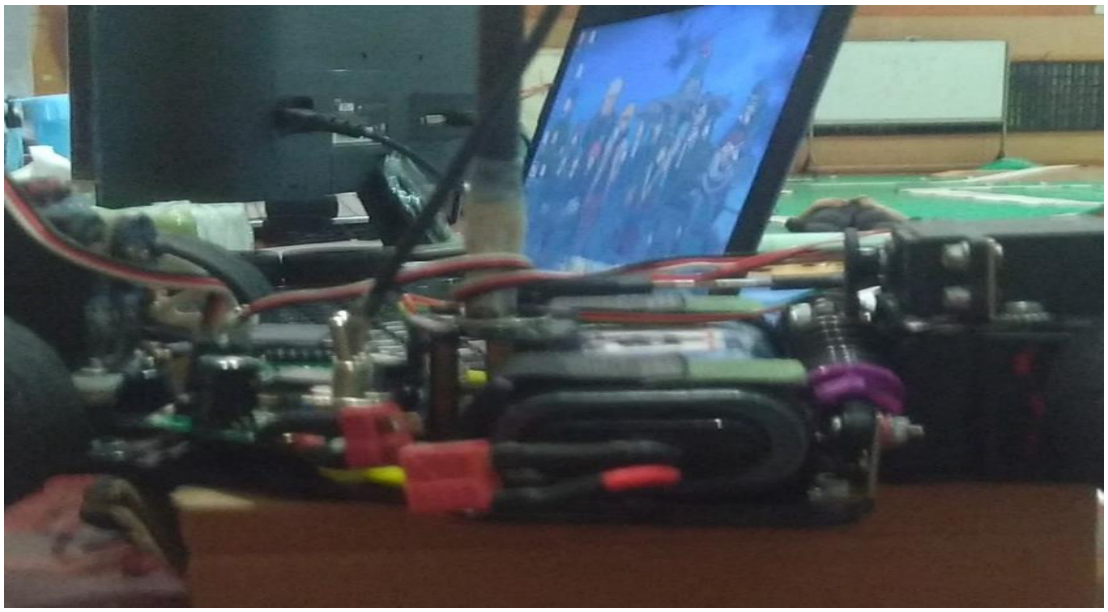
## 3.1. 摄像头的固定和安装

摄像头作为我们摄像头组最重要的传感器，它的固定和安装对小车的影响是十分巨大的，摄像头的布局和安装取决于系统方案，反过来又会影响系统的稳定性与可靠性以及软件的编写。

## 3.2. 摄像头固定与底座改装

由于本届车模为四轮车，所以摄像头架在车子的中间部分，介于电池和电路板之间，这样节省空间而且也不会让重心偏移太大，而摄像头的角度也很有讲究，角度低的时候能看到很远的赛道信息，但是图像较为模糊，不适合图像处理的编写，角度较高是，能看到的图像信息较少，但是分辨率明显更好，但因为某些特殊赛道的需要(如直角，十字，障碍等)，我们还是将摄像头的前瞻保持在2米以上以求最有效的信息，在程序的编写中，我们发现摄像头视野的宽广往往直接影响赛道信息提取的精准度，在浙江省赛上的成功尝试，也证明了这种架法的优越性。

图 3.1 摄像头架法



## 3.3. 摄像头与支撑杆之间的安装

摄像头与支撑杆之间的安装能否稳定是整个采集图像是否可靠地重要原之一。考虑到 CCD 摄像头本身就比较笨重，要架相对高的车就要尽量减轻附加固定结构的重量。为此我们也想了很多，最后我们采用了一种比较轻便的安装方法，其中碳棒为 6.0mm 的，但是直接用 6.0mm 的钻头的，打孔时加上机械抖动会放大孔径，所以我们采用了 5.9mm 的钻头进行打孔。这样碳棒卡的比较紧。另外固定螺丝的孔径为 3.0mm，但是这个孔径可以自己任意调节，只要便于固定，有适合的螺丝就可以。具体的固定如图 3.2 所示：

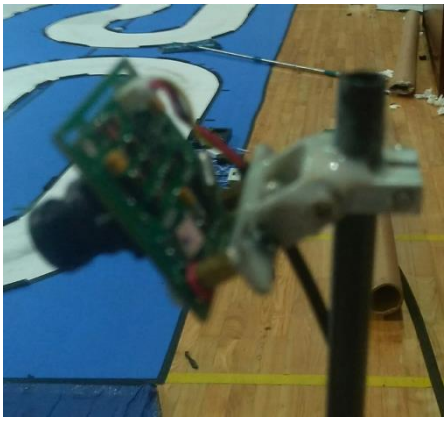


图 3.2 摄像头与碳棒之间固定

## 3.4. 编码器的安装

为使整个控制形成闭环，当前速度反馈是必须的。为了测出当前速度，我们用了两个磁矩式编码器，应用十分方便，编码器的固定用的是玻纤结构，足能稳定固定且重量轻。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。安装好的编码器如图 3.6 所示。固定钢片尺寸如图 3.7 所示

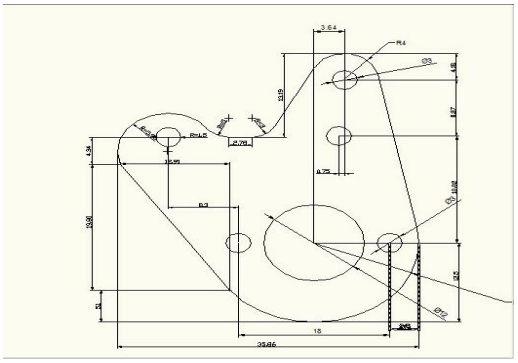


图 3.4 编码器固定片

## 3.5. 小车轮胎的定位调整

主销角度的调整，对整个小车的机械性能至关重要，角度调整的好有利于小车转弯，并且减小小车转弯时的抖动。主销后倾角是指在纵向平面内主销轴线与地面垂直线之间的夹角，它在车辆转弯时会产生与车轮偏转方向相反的回正力矩，使车轮自动恢复到原来的中间位置上。所以，主销后倾角越大，车速越高，自动回正的能力就越强 ,但是我们经过实际的测试调试车模时发现对于不同的车模，所要调节的后倾角度是不同的，主要是每辆车模的重量分配微小偏差所导致的。调节后倾角度有利于车模在转弯时的抓地力，这样车模就不容易打滑和出现推头现象。经过我们实际测试，在调节时调节时要把握好左边和右边打角的平衡，另外，后倾的角度也不是越大越好，基本在 1~3 度之间，调解时由车模决定。主销内倾角是指在横向平面内主销轴线与地面垂直线之间的夹角，它的作用也是使前轮自动回正。角度越大前轮自动回正的作用就越强，但转向时也就越费力，轮胎磨损增大；反之，角度越小前轮自动回正的作用就越弱。在查看资料时，我们发现通常汽车的主销内倾角不大于 8°，主销内倾的调整应该保持在一个合适的范围，“一般来说 0~8 度范围内皆可”。在实际测试中，

我们进行对车模内倾角度的不断微调使之适应车模，并在软件和机械配合下进行微小的调整。经过测试说明，在这范围内某个内倾的角度是车模的最佳适应状态。在微调的过程中还需注意车模前后的重量分配问题。对于我们的车模是稍稍重心偏后点，前后的重量分配还是比较均匀的。这样稍稍调到一定的角度，只要转弯抓地力足够即可。角度过大，轮胎对地面的接触面过小，假设轮胎的形变较大，或者弯道突然抖动，这样会使摩擦力突变，这样也是对车不利得，这跟车身抖动也是有一定关系的。在汽车的横向平面内，轮胎呈“八”字型时称为“负外倾”，而呈现“V” 字形张开时称为正外倾。在车模外倾角度的调整中，我们采用了正外倾的方式，通过车模跑的测试，提高前轮的转向安全性和转向操纵的轻便性。但是“V”字的角度不太大，基本在 1 度左右即可。但是一定不要太大，适当的放开一两圈就够了。但是调节前束是要根据具体车模来调节的，配合重心的匹配问题，我们的车模基本没有前束。这也是通过我们多次测试所的出来的结论。具体问题具体解决。每辆车都有各自的性能上的区别，采用硬软件和机械的配合调节才是最佳的调节方法。

虽然模型车的主销后倾角主销内倾角车轮外倾角和前束等均可以调整，但是由于车模加工和制造精度的问题，在通用的规律中还存在着不少的偶然性，一切是实际调整的效果为准。在调整的过程中，要注意各个方面的配合调节。另外，在调节时要注意轮胎的磨合，新的轮胎比较生硬，摩擦力也不是很好，经过磨合的轮胎比较柔软，相对来说抓地力也明显比新轮胎好。

## 3.6. 车模机械架构总结

上面我们已经介绍了车模的基本几个重要的机械架构部分。对于一辆架构优良的车来说，除了考虑以上几个部分还需要考虑一些细节，同时也要对车模的整体架构进行微小调整和重量的匹配。车模的重量均匀的分布在车模的四个轮子上对车模整体的平衡稳定都是有非常大的意义的。同时这也在一定程度上影响了左右的打角。另外要注意的是车模的底盘高度和重心问题，实践证明降低底盘高度，降低整个车模的整体的重心对转弯是有很大帮助的。但是同时也要考虑到车模是否过坡是擦到赛道。总之，具体的情况依车模而定。下图是我们车模的整体结构和布局。

图3.6 车模侧面

我们现在的车模结构总体均衡，构架轻巧，结构牢固。每一小块机械结构都是经过我们亲手设计打造。为这次比赛，我们不知换过多少方案，架构过多少次车模，机械结构是改过又改过的。

# 第四章. 软件系统设计

## 4.1. 软件控制程序的整体思路

软件的控制，就是让控制车模在符合大赛规则前提下，以最快的速度跑完整个赛道。不论上哪种方案，软件的总体框架总是相似的，我们追求的就是稳定至上，兼顾速度，以我们的方案，软件上就是图像采集、图像处理识别黑线、平衡控制，速度控制以及速度反馈。

## 4.2. 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。

## 4.3. 摄像头的工作原理

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。摄像头信号波形如图 4.1 所示。

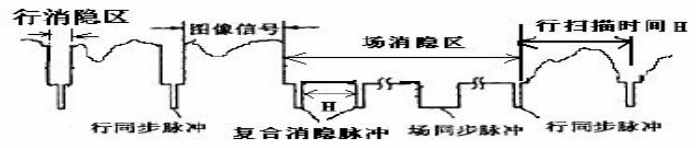



图4.1 摄像头信号波形

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。

### 4.2.3 LM1881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 4.2 所示。

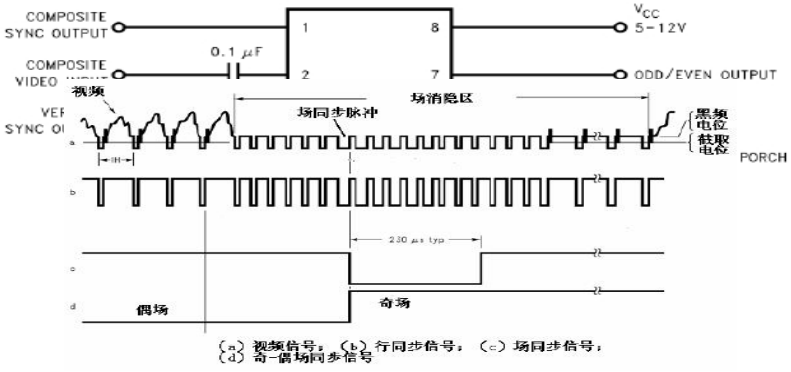



图4.2 摄像头信号波形

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。 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。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。

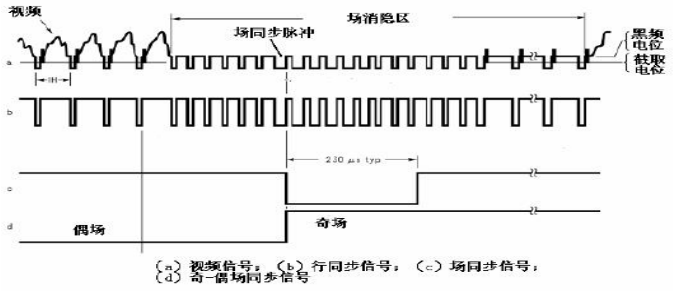



图4.3 视频及其分离信号波形图

### 4.2.4 黑线提取

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算法二值化算法的思路是：设定一个阈值 value，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。所以就将一场图像分为两部分，近端部分和远端部分

### 4.2.5 赛道中心拟合

因为是双边的黑线赛道，所以我们在正确提取黑线信息的基础上，外加赛道中心拟合函数模块，其具体思想为“两点求中线”。下图为上位机上的处理效果图

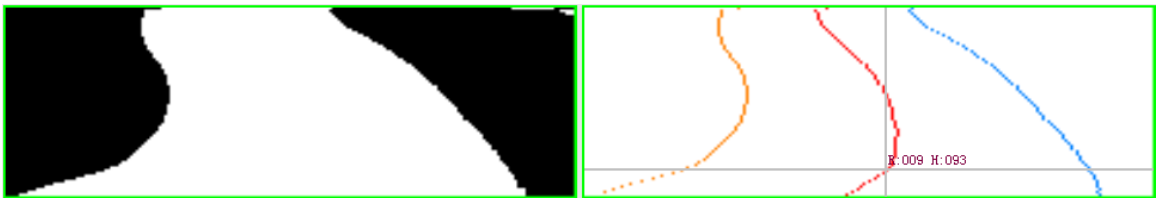



图4.4 赛道中心拟合图

## 4.4. 电机打角控制

我们采用的曲率打角，只根据光电式编码器反馈来的脉冲数转化为速度再叠加电机的相对反应时间，然后根据这打角行位置与图像中心位置的偏差通过一个二次函数来计算出所需打脚力度。图 4.5 为电机打角曲线图。

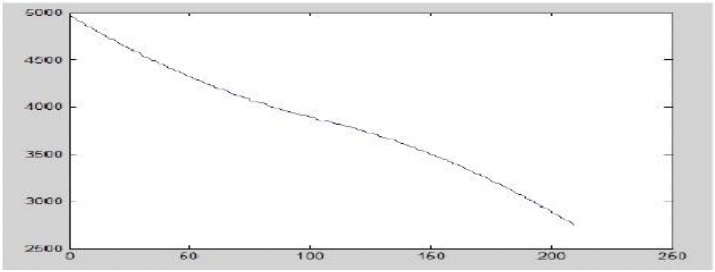



图 4.5 赛道中心拟合图

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打角值这样赛车走的路径会得到很大程度上的优化。

## 4.5. 速度控制

速度控制策略的好坏就决定小车能否取得优秀成绩，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，加减速控制就是让车模以最快的反应速度达到标定的速度，反应越快越好。程序每进入一次速度控制函数，首先读出于编码器连接的脉冲累加器计数的值，计算出当前速度，再与标定的速度比较，如果当前速度大于标定的速度再根据两速度的差值决定减速的力度；如果当前速度小于标定的速度，就根据两速度差值决定加速的急缓。减速就是递减电机正转的 PWM 值，如果速度差很大，便要启动电机反转刹车以迅速达到减速的目的。加速的过程是对电机正转的过程，这个过程对电机的驱动的承受能力是个很大的考验，而且它是 PWM 累加的过程，但是，要对 PWM 限定范围，加速的急缓只是每次累加的量大小不同。

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;
```

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。图 4.6 是 PID 控制系统仿真结果。

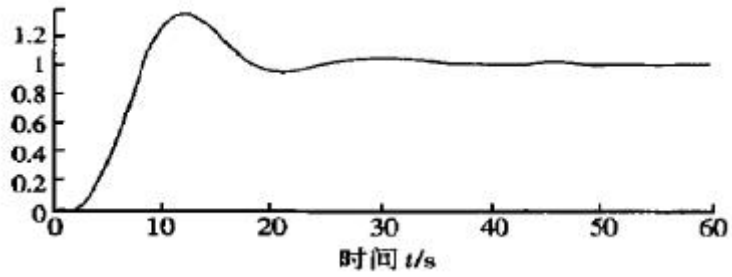



图 4.7 系统仿真结果

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。

## 4.6. 软件程序总结

软件部分是整个控制系统的核心。软件上主要有以下几个难点：

1) 怎样判断采样回来的赛道信息的有效性；

2) 怎样使速度控制配合好打角。

对于本车模的软件系统，我们充分发挥后架车的优势，超远的大前瞻和广

阔的视角带给我们不少创作灵感。对于车模来说，软件控制是核心，而对于软件来说，打角和速度控制都不算是核心，真正的核心应该是打角和速度控制的相互配合！本车模软件系统对速度的控制均以 PID 控制技术为原理，适当发挥再加以利用。

# 第五章. 智能车调试说明

前面几章一直是在为系统制定方案以及方案的细化。但整个系统的完善主要还是在系统的现场调试。在细分的每个模块中，大部分都涉及有众多参数，对这些参数的确定就需要软硬件联合调试。而这过程就需要一整套开发调试环境已工具。包括程序源代码的编辑以编译环境，参数调节与设定工具。

## 5.1. 上位机调试软件的设计

我们的上位机是本届完全自主开发的，图 5.1 是上位机软件界面。

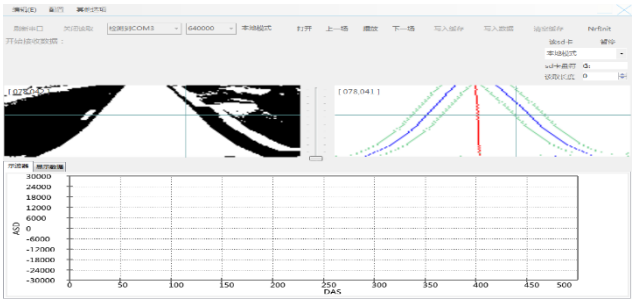


图5.1 上位机1

仿真模式：由于图像处理越来越复杂，对于图像仿真越来越重要，PC 仿真模式可以将单片机代码导入，更加直观看到图像处理结果。

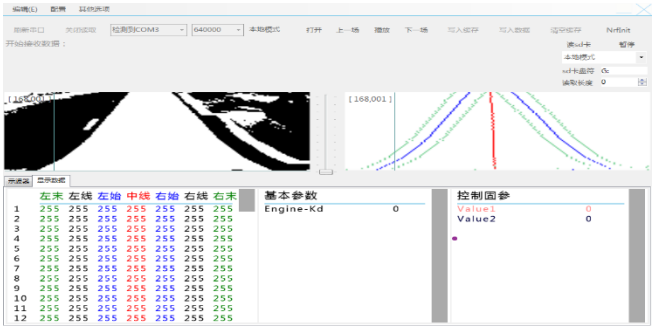


图5.2 上位机2

本地模式：小车的实际图像反馈对于调车非常重要，利用 SD 卡将小车行进中的图像以及处理信息保存，上传到上位机，以便常看。

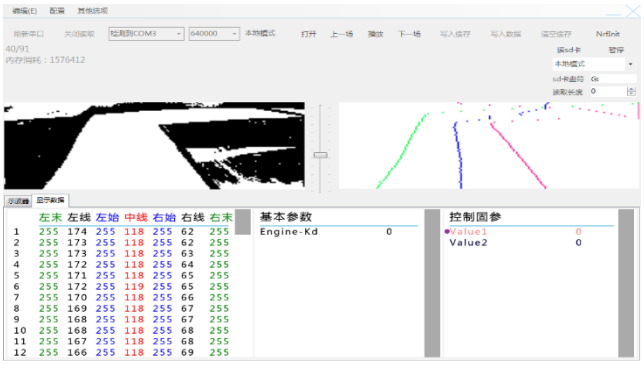



图 5.3 上位机 3

调试模式：为了提高调试效率，特别为上位机做了在线调试系统，信息一目了然。

## 5.2. Flash 存储模块

因为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 K60 的存储空间远不能满足我们的要求。所以我们设计了外部 SD 卡来存储海量信息，这让我们在图像处理上有很多的发挥空间。通过 SD 卡，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。

## 5.3. 现场调试

在完成基本程序之后，做的最多的工作就是调试，调试是一持续性的工作。之前说过，我们为调试的方便，特别制作了按键加液晶模块。写程序的时候，

自己想好的一些参数在静态测试的时候可能很好，但是当车在赛道上跑的时候，情况就不一样了。所以，车模必须在赛道上多跑以发现问题进而解决问题。

车模主板上的OLED和按键就为这种现场调试提供了很好地硬件平台，软件的同学在修改某个参数的时候就无需跑去电脑边上下载了，非常的方便。

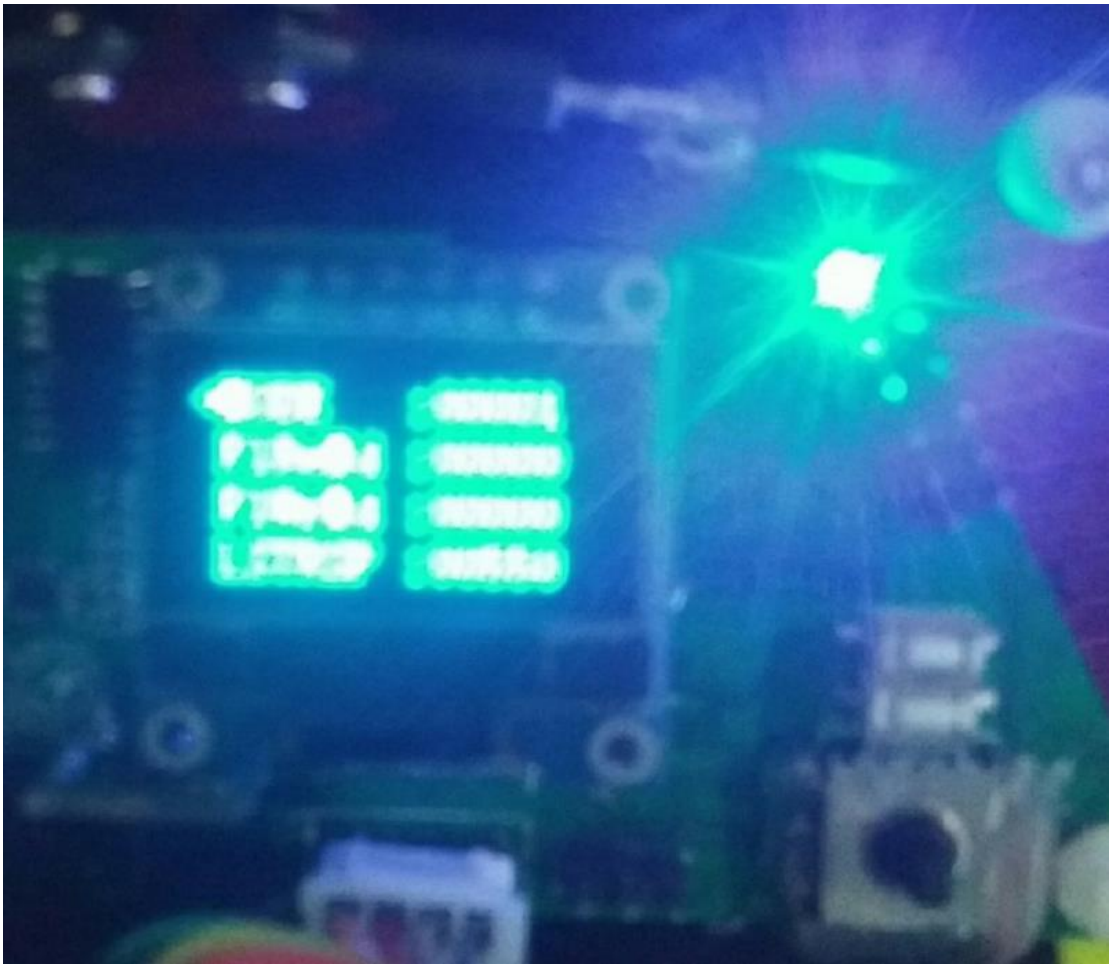


图 5.4 OLED

# 第六章. 车模技术参数说明

## 6.1. 车模主要技术参数

| 名称 | 参数 |
| --- | --- |
| 车模总质量 | 970g |
| 车模长度 | 304mm |
| 车模宽度 | 200mm |
| 车模高度 | 40mm |
| 电路电容器总容量 | 4483.71uF |
| 传感器种类及个数 | 编码器 2 个，CCD 摄像头 1 个，陀螺仪1个 |
| 外加伺服电机个数 | 0 个 |
| 赛道信息检测精度 | 22.5mm |
| 赛道信息检测频度 | 50（次 / 秒） |

表 6.1 车模部分主要技术.参数

## 6.2. 电路中芯片的种类及数量

| 芯片名称 | 使用数量 | 功能简述 |
| --- | --- | --- |
| MK60DN512ZVLQ10 | 1 | 微控制器 |
| TPS55340 | 1 | 开关电源芯片 |
| LM1084 | 1 | 低压差线性稳压器 |
| TPS7333 | 1 | 开关电源芯片 |
| SY8133 | 1 | 开关电源芯片 |

| | | |
|---|---|---|
| LM1881 | 1 | 场行分离 |
| Y5510 | 1 | ADC芯片 |
| IR7843 | 4 | H桥 |
| IR2104 | 2 | 栅极驱动 |
| LT6656-2.5 | 1 | 联基准电压芯片 |

表 6.2 电路中芯片的种类及数量

# 第七章. 总结

本文介绍了 CCD 摄像头智能小车的总体设计方案，从机械布局、架构设计，硬件系统设计和软件设计三个部分详细介绍了我们设计小车的方案和一些创新。在基本的设计流程中，我们也进行了不断地尝试，并且取得了较理想的成果。车模的设计要综合考虑各个方面的问题和可能遇到的问题。在设计的时候要胆大心细，求实创新。飞思卡尔举办到现在已经是第十届了。在技术和经验上都是比较成熟了，尤其是摄像头，比赛的差距更是非常小。想要在比赛中取得好成绩就要在平时注重机械、硬件、软件的综合配合调节和优化。而这一届我们的小车在软件、硬件和机械上都有了一些新的突破。摄像头的架构安装，与以前相比更加稳固，而且拆装方便；底盘高度的调节，PCB 板配合车模形状，液晶贴底，降低底盘高度的同时使整个车身的重量全部下降，相比以前，整个车身的重心要远远下降，这样在弯道就有明显的优势；芯片选型也较好，PCB 布局布线贴合车模车身设计，各个模块设计布线也是相对优化的。软件设计部分，相对以前已经有了的基础之上有了较大的优化。通过图像处理，速度反馈和速度控制，有效地让小车进行循迹。

# 第八章. 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，

我们技术上在不断成长，思想上不断成熟。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，我们要感谢指导老师的悉心教导。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。

# 第九章. 参考文献

[1] 孙同景，陈桂友 Freescale 9S12 十六位单片机原理及嵌入式开发技术，北京 -机械工业出版社，2008

[2] 邵贝贝. 单片机嵌入式应用的在线开发方法. 北京-清华大学出版社 2004年 10 月第 1 版

[3] 卓晴，黄开胜，邵贝贝 学做智能车 北京-北京航空航天大学出版社 2007

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

[5] 李宁，刘启新 电机自动控制系统 北京-机械工业出版社，2003

[6] 潘松，黄继业 现代数字电路基础教程 北京-科学出版社 2008

[7] 魏彪，盛新志 激光原理及应用 重庆-重庆大学出版社 2007

[8]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

[9] 赵先奎 汽车前轮定位 黑龙江 黑龙江出入境检验检疫局 2008

[10]刘锐宁 Visual C++编程之道 人民邮电出版社 2011

[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]姚领田 精通 MFC 程序设计 人民邮电出版社 2006

[13]刘锐宁 Visual C++编程之道 人民邮电出版社 2011

# 附录 A

小车程序：

```
int main(void)
{
    SystemClockSetup(ClockSource_EX50M,CoreClock_100M);          // 初 始化系统时钟  使用外部 50M 晶振  PLL 倍频到 100M
/************************
**初始化**
************************/
    MySD_Init();
    SD_ManyWT_Init();
    DMAInit();
    GPIOInit();
    FTMInit();
    DMAcnt_Init();
    Data_Uart_Init();
    DelayInit();
    OLED_Init();
    KEY_Init();


    while(1)
    {
        while(count <= 4)
    {
        count_f();    //count    加减

        OLED_Write_String(4,0,(uint8_t*)prm1);
        OLED_Write_Num4(10,0,prm1_val);

        OLED_Write_String(4,2,(uint8_t*)prm2);
```

```
OLED_Write_Num4(10,2,prm2_val);

OLED_Write_String(4,4,(uint8_t*)prm3);
OLED_Write_Num4(10,4,prm3_val);

OLED_Write_String(4,6,(uint8_t*)prm4);
OLED_Write_Num4(10,6,prm4_val);

OLED_Write_Char(0,colo[count%4],43);
switch(count%4)
{
    case 1:
        if(key3==0)
        {
            DelayMs(100);
            if(key3==0)
            {
                OLED_Clear();
                OLED_Write_Char(0,colo[count%4],43);
                prm1_val+=prm1_precision;
                OLED_Write_String(4,0,(uint8_t*)prm1);
                OLED_Write_Num4(10,0,prm1_val);
                OLED_Clear();
            }
        }
        if(key1==0)
        {
            DelayMs(100);
            if(key1==0)
            {
                OLED_Clear();
                OLED_Write_Char(0,colo[count%4],43);
```

```
                prm1_val-=prm1_precision;
                OLED_Write_String(4,0,(uint8_t*)prm1);
                OLED_Write_Num4(10,0,prm1_val);
            }
        }
    break;

    case 2:
        if(key3==0)
        {
            DelayMs(keydelay);
            if(key3==0)
            {
                OLED_Clear();
                OLED_Write_Char(0,colo[count%4],43);
                prm2_val+=prm2_precision;
                OLED_Write_String(4,2,(uint8_t*)prm2);
                OLED_Write_Num4(10,2,prm2_val);
            }
        }
        if(key1==0)
        {
            DelayMs(keydelay);
            if(key1==0)
            {
                OLED_Clear();
                OLED_Write_Char(0,colo[count%4],43);
                prm2_val-=prm2_precision;
                OLED_Write_String(4,2,(uint8_t*)prm2);
                OLED_Write_Num4(10,2,prm2_val);
            }
```

```
        }
    break;

    case 3:
        if(key3==0)
        {
            DelayMs(keydelay);
            if(key3==0)
            {
                OLED_Clear();
                OLED_Write_Char(0,colo[count%4],43);
                prm3_val+=prm3_precision;
                OLED_Write_String(4,4,(uint8_t*)prm3);
                OLED_Write_Num4(10,4,prm3_val);
            }
        }
        if(key1==0)
        {
            DelayMs(keydelay);
            if(key1==0)
            {
                OLED_Clear();
                OLED_Write_Char(0,colo[count%4],43);
                prm3_val-=prm3_precision;
                OLED_Write_String(4,4,(uint8_t*)prm3);
                OLED_Write_Num4(10,4,prm3_val);
            }
        }
    break;

    case 0:
        if(key3==0)
```

```
                {
                    DelayMs(keydelay);
                    if(key3==0)
                    {
                        OLED_Clear();
                        OLED_Write_Char(0,colo[count%4],43);
                        prm4_val+=prm4_precision;
                        OLED_Write_String(4,6,(uint8_t*)prm4);

//FTM_PWM_ChangeDuty(FTM2_CH0_PB18,prm4_val);
                        OLED_Write_Num4(10,6,prm4_val);
                    }
                }
                if(key1==0)
                {
                    DelayMs(keydelay);
                    if(key1==0)
                    {
                        OLED_Clear();
                        OLED_Write_Char(0,colo[count%4],43);
                        prm4_val-=prm4_precision;
                        OLED_Write_String(4,6,(uint8_t*)prm4);

//FTM_PWM_ChangeDuty(FTM2_CH0_PB18,prm4_val);
                        OLED_Write_Num4(10,6,prm4_val);
                    }
                }
            break;
        }

        if(key5 == 0)
        {
```

```
                DelayMs(keydelay);
                if(key5 == 0)
                {
                    if(mode <= 0)
                    {
                        mode = 0;
                    }
                    if(mode >= 6)
                    {
                        mode = 6;
                    }

                    switch(mode)
                    {
                        case 3:
                            OLED_Clear();
                            OLED_Write_String(0,0,(uint8_t*)"SDW");
                            while(1)
                            {
                                if(Pro_OK == 1)       //整场采集完成
                                {
                                        /**********************
    图像滤躁
    **********************/
    //  for(i = 2;i < 198;i ++)
    //  {
    //          if((RowNum >= 2) && (RowNum < 68))
    //          {
    //                  if      ((P_Pixels[RowNum][i]       ==      0)      &&
(P_Pixels[RowNum][(byte)(i - 1)] == 1)
    //                          && (P_Pixels[RowNum][(byte)(i + 1)] == 1) &&
(P_Pixels[RowNum - 1][i] == 1) && (P_Pixels[RowNum +1][i] == 1))
```

```
//                      {
//                              P_Pixels[RowNum][i] = 1;
//                      }
//              }
//   }

            /**********************
            第一场扫线
            **********************/
            if (flag2 <= 1)   //第一场
            {
                    for (i = 100; i < 198; i++)     //第一场从中间往两边找线
                    {
                            if      ((P_Pixels[RowNum][i]      ==      0)      &&
(P_Pixels[RowNum][(byte)(i + 1)] == 0))       //滤噪点/
                            {
                                    L_black[RowNum]  =  i;      //记录第一次扫
到左边线的位置
                                    break;
                            }
                    }


                    for (i = 100; i > 2; i--)
                    {
                            if      ((P_Pixels[RowNum][i]      ==      0)      &&
(P_Pixels[RowNum][(byte)(i -1)] == 0))
                            {
                                    R_black[RowNum] = i;     //记录第一次扫到
右边线的位置
                                    break;
                            }
```

```
                }
                Center = (byte)Sv_OutLine(((R_black[RowNum] + L_black[RowNum]) >>1));
                LCenter[RowNum] = Center;
        }


        /**********************
         *左右扫线
         *小弯道判定
         **********************/
        else
        {
                if (RowNum <= 1)
                {
                        L_start[RowNum] = LCenter[RowNum];
                        for(i = L_start[RowNum];i < 198;i++)
                        {
                                if ((P_Pixels[RowNum][i] == 0) && (P_Pixels[RowNum][(byte)(i + 1)] == 0))  //找到左边线
                                {
                                        L_black[RowNum] = i;
                                        L_flag = 0;
                                        break;
                                }
                                else
                                {
                                        L_flag = 1;    //标记左边丢线
                                }
                        }
                }
                if(RowNum >= 2)
```

```
                {
                        L_start[RowNum] = (byte)Sv_OutLine((L_black[RowNum - 1] - 15));
                        if(L_start[RowNum] < LCenter[RowNum])
                        {
                                L_start[RowNum] = LCenter[RowNum];
                        }
                        for (i = L_start[RowNum]; i < 198; i++)  //向左找线
                        {
                                if ((P_Pixels[RowNum][i] == 0) && (P_Pixels[RowNum][(byte)(i + 1)] == 0))  //找到左边线
                                {
                                        L_black[RowNum] = i;
                                        L_flag = 0;
                                        break;
                                }

                                else
                                {
                                        if(RowNum <= 65)
                                        {
                                                if ((i >= 197) && (P_Pixels[RowNum][i] == 1)
                                                                && ((P_Pixels[RowNum + 1][i - 25] == 1) || (P_Pixels[RowNum + 2][i - 25] == 1) ||(P_Pixels[RowNum + 3][i -28] == 1))
                                                                && ((P_Pixels[RowNum + 1][i] == 0) || (P_Pixels[RowNum + 2][i] == 0) || (P_Pixels[RowNum + 3][i] == 0)))
                                                                {

    L_smallcurve_rownum = (byte)(RowNum + 1);
                                                                L_smallcurve_flag = 1;
```

```
                                        }
                                    }

                                    if((i >= 196) && (P_Pixels[RowNum][i] == 1) && (RowNum <= 20))    //近处双边丢线//用来判定十字
                                    {
                                        L_nearlost_flag = 1;
                                    }

                                    L_flag = 1;    //标记左边丢线
                                }
                            }
                        }


                        if(RowNum <= 1)
                        {
                            R_start[RowNum] = LCenter[RowNum];
                            for(i = R_start[RowNum];i>2;i--)
                            {
                                if (P_Pixels[RowNum][i] == 0)
                                {
                                    R_black[RowNum] = i;
                                    R_flag = 0;
                                    break;
                                }
                                else
                                {

                                    R_flag = 1;      //标记右边丢线
                                }
```

```
                }
            }
            if(RowNum >= 2)
            {
                R_start[RowNum] = (byte)Sv_OutLine(R_black[RowNum - 1] + 15);
                if(R_start[RowNum] > LCenter[RowNum])
                {
                    R_start[RowNum] = LCenter[RowNum];
                }
                for (i = R_start[RowNum]; i > 0; i--)     //向右找线
                {
                    if (P_Pixels[RowNum][i] == 0)
                    {
                        R_black[RowNum] = i;
                        R_flag = 0;
                        break;
                    }
                    else
                    {
                        if (RowNum <= 65)
                        {
                            if ((i <= 3) && (P_Pixels[RowNum][i] == 1)
                                && ((P_Pixels[RowNum + 1][i + 25] == 1) || (P_Pixels[RowNum + 2][i + 25] == 1) || (P_Pixels[RowNum + 3][i + 25] == 1))
                                && ((P_Pixels[RowNum + 1][i] == 0) || (P_Pixels[RowNum + 2][i] == 0) || (P_Pixels[RowNum + 3][i] == 0)))
                                {

    R_smallcurve_rownum = (byte)(RowNum + 1);
```

```
                                                    R_smallcurve_flag
= 1;
                                                }
                                            }
                                            if((i <= 3) && (P_Pixels[RowNum][i]
== 1) && (RowNum <= 20))     //近处丢线//用来判定十字
                                            {
                                                R_nearlost_flag = 1;
                                            }

                                            R_flag = 1;       //标记右边丢线
                                        }
                                    }
                                }

                                if((RowNum >= 4) && (RowNum <= 40))
                                {
                                    if((L_nearlost_flag == 0) || (R_nearlost_flag == 0))
                                    {
                                        L_nearlost_flag = 0;
                                        R_nearlost_flag = 0;
                                    }

                                    if((L_nearlost_flag == 1) && (R_nearlost_flag == 1))
                                    {
                                        Bothside_lost_count++;
                                    }
                                }


                                Center      =      (byte)Sv_OutLine(((R_black[RowNum]      +
L_black[RowNum]) >> 1));
                                LCenter[RowNum] = Center;
```

```
}


/*****************************
 *小 S 判定
 *****************************/
if(S_curve_flag == 0)
{
    if ((RowNum >= 50) && (RowNum < 69))
    {
        if ((L_black[RowNum] <= L_black[RowNum - 1])
            && (L_black[RowNum - 1] > L_black[RowNum - 2])
            && (L_black[RowNum - 2] > L_black[RowNum - 3])
            && (L_black[RowNum - 3] > L_black[RowNum - 4]))
        {
            //setText 用户自定义("小 S 左极大尖角" + RowNum);
            S_leftmax_it_flag = 1;
            S_left_it_rownum = (byte)(RowNum - 1);
        }
        else
            if ((L_black[RowNum] >= L_black[RowNum - 1])
            && (L_black[RowNum - 1] < L_black[RowNum - 2])
            && (L_black[RowNum - 2] < L_black[RowNum - 3])
            && (L_black[RowNum - 3] < L_black[RowNum - 4]))
            {
                S_leftmin_it_flag = 1;
```

```
                                S_left_it_rownum = (byte)(RowNum - 1);
                        }

                    if ((R_black[RowNum] <= R_black[RowNum - 1])
                        && (R_black[RowNum - 1] > R_black[RowNum
- 2])
                        && (R_black[RowNum - 2] > R_black[RowNum
- 3])
                        && (R_black[RowNum - 3] > R_black[RowNum
- 4]))
                    {
                        S_rightmax_it_flag = 1;
                        S_right_it_rownum = (byte)(RowNum - 1);
                    }
                    else
                        if ((R_black[RowNum] >= R_black[RowNum -
1])
                        && (R_black[RowNum - 1] < R_black[RowNum
- 2])
                        && (R_black[RowNum - 2] < R_black[RowNum
- 3])
                        && (R_black[RowNum - 3] < R_black[RowNum
- 4]))
                        {
                            S_rightmin_it_flag = 1;
                            S_right_it_rownum = (byte)(RowNum -
1);
                        }
                }
            }

            /**********************
             *左边丢线
```

```
            *弯接直道判定
            **********************/
          if ((R_flag == 0) && (L_flag == 1) && (L_smallcurve_flag==0))
//左边丢线
          {
                if (RowNum <= 1)              //找右线
                {
                      R_start[RowNum] = LCenter[RowNum];
                      for (i = R_start[RowNum]; i > 2; i--)
                      {
                            if (P_Pixels[RowNum][i] == 0)
                            {
                                  R_black[RowNum] = i;
                                  R_flag = 0;
                                  break;
                            }
                      }
                }
                if (RowNum >= 2)
                {
                      if((RowNum >= 30) && (RowNum <= 55))
                      {
                            if(R_black[RowNum] >= R_black[RowNum -
1])
                            {
                                  if((R_black[RowNum]              -
R_black[RowNum - 1]) <= 3)                                  {
                                        L_straightway_count++;
                                  }
                                  if    ((R_black[RowNum    -    1]    -
R_black[RowNum]) >= 4)
                                  {
```

```
                                        L_straightway_it_count++;
                                }
                        }

                        if (R_black[RowNum] < R_black[RowNum - 1])
                        {
                                if((R_black[RowNum - 1] - R_black[RowNum]) <= 4)
                                {
                                        L_straightway_count++;
                                }
                        }
                }

                R_start[RowNum] = (byte)Sv_OutLine(R_black[RowNum - 1] + 10);
                for (i = R_start[RowNum]; i > 2; i--)     //向右找线
                {
                        if (P_Pixels[RowNum][i] == 0)
                        {
                                R_black[RowNum] = i;
                                R_flag = 0;
                                break;
                        }
                }
        }
        L_black[RowNum] = 198;     //默认左边线位置


        if(RowNum <= 2)
        {
                LCenter[RowNum]=(byte)((L_black[RowNum] +
```

```
R_black[RowNum])>>1);
                }
                if(RowNum >= 2 && RowNum <= 68)
                {
                        if(R_black[RowNum]  >  R_black[RowNum  -  1])
//左倾弯
                        {
                                Center                                  =
(byte)Sv_OutLine((LCenter[RowNum - 1] +
                                                        (R_black[RowNum]        -
R_black[RowNum - 1])));
                        }
                        else        //右倾弯
                                if (R_black[RowNum] < R_black[RowNum -
1])
                        {
                                Center                                  =
(byte)Sv_OutLine((LCenter[RowNum - 1] +
                                                        (R_black[RowNum  -  1]  -
R_black[RowNum])));
                        }
                }
                LCenter[RowNum] = Center;


        }
        if ((R_flag == 1) && (L_flag == 0) && (R_smallcurve_flag == 0))     //
右边丢线
        {
                if (RowNum <= 1)
                {
                        L_start[RowNum] = LCenter[RowNum];
                        for (i = L_start[RowNum]; i < 198; i++)
                        {
```

```
                    if (P_Pixels[RowNum][i] == 0)   //找到左边线
                    {
                        L_black[RowNum] = i;
                        L_flag = 0;
                        break;
                    }
                }
            }
            if (RowNum >= 2)
            {
                if ((RowNum >= 30) && (RowNum <= 55))           /
                {
                    if (L_black[RowNum] <= L_black[RowNum - 1])
                    {
                        if ((L_black[RowNum - 1] - L_black[RowNum]) <= 3)
                        {
                            R_straightway_count++;
                        }
                        if ((L_black[RowNum - 1] - L_black[RowNum]) >= 4)       //直道突变点
                        {
                            R_straightway_it_count++;
                        }
                    }
                    if (L_black[RowNum] > L_black[RowNum - 1])
                    {
                        if ((L_black[RowNum] - L_black[RowNum - 1]) <= 4)       //左线行差小于 4，计数累加
                        {
```

```
                                        R_straightway_count++;
                                }
                        }
                }
                L_start[RowNum] = (byte)Sv_OutLine((L_black[RowNum - 1] - 10));
                for (i = L_start[RowNum]; i < 198; i++)   //向左找线
                {
                        if (P_Pixels[RowNum][i] == 0)   //找到左边线
                        {
                                L_black[RowNum] = i;
                                L_flag = 0;
                                break;
                        }
                }

                if (RowNum <= 68)
                {
                        if ((L_black[RowNum - 1] >= L_black[RowNum]))
                        {
                                Center = (byte)Sv_OutLine((LCenter[RowNum - 1] -
(L_black[RowNum - 1] - L_black[RowNum])));
                        }
                        else
                        {
                                Center = (byte)Sv_OutLine((LCenter[RowNum - 1] -
                                        (L_black[RowNum] - L_black[RowNum - 1])));
```

```
                        }
                    }
                }
                R_black[RowNum] = 2;     //默认右边线的位置


                if(L_flag==1 && R_flag==1)
                {
                    //setText 用户自定义("左右均丢线" + RowNum);
                    Center = (byte)Sv_OutLine((LCenter[3] + LCenter[4]
+ LCenter[5] + LCenter[6])>>2);
                }


                LCenter[RowNum] = Center;
            }


            /*************************
             * 障碍
            * *************************/
            if( (RowNum < 55) && (RowNum >= 10) && (R_flag == 0) &&
(L_flag == 0))    //一边为直道另外一边检测到突变且突变向上连续几行边线连
续
            {
                if(L_black[RowNum] < L_black[RowNum - 1]) //判定远处障
碍
                {
                    if((L_black[RowNum - 1] - L_black[RowNum]) <= 2)
//连续直道计数
                    {
                        fence_left_count++;
                    }
```

```
                else
                    if ((L_black[RowNum - 1] - L_black[RowNum]) >= 6)        //检测到左边突变，直道计数清零
                    {
                        fence_left_count = 0;
                        fence_left_it_count++;
                        fence_start_rownum = RowNum;
                    }
            }

            if (L_black[RowNum] > L_black[RowNum - 1])      //近处障碍预判//左边障碍
            {
                if (((L_black[RowNum] - L_black[RowNum - 1]) >= 22)    //左线突变
                                                                )
                {
                    fence_left_count = 0;    //左边计数清零
                    near_left_fence_pre_flag = 1;
                    near_fence_end_rownum = RowNum;
                }
            }

            if((R_black[RowNum] > R_black[RowNum - 1]))
            {
                if ((R_black[RowNum] - R_black[RowNum - 1]) <= 2)
                {
                    fence_right_count++;
                }
                else
                    if((R_black[RowNum] - R_black[RowNum -
```

```
1]) >= 6)
                                    {
                                        fence_right_count = 0;
                                        fence_right_it_count++;
                                        fence_start_rownum = RowNum;
                                    }
                            }
                            if (R_black[RowNum - 1] > R_black[RowNum])
                            {
                                if (((R_black[RowNum - 1] - R_black[RowNum]) >=
22)
                                        )
                                {
                                    fence_right_count = 0;   //右边直道计数清零
                                    near_right_fence_pre_flag = 1;
                                    near_fence_end_rownum = RowNum;
                                }
                            }
                        }
                        if ((RowNum <= 40) && (RowNum > 2))
                        {
                            if ((L_black[RowNum] >= L_black[RowNum - 1])
                                && (L_black[RowNum - 1] < L_black[RowNum - 2])
                                && (L_black[RowNum - 2] < L_black[RowNum - 3]))
                            {
                                if ((P_Pixels[RowNum + 1][L_black[RowNum]] == 1)
                                    &&      (P_Pixels[RowNum      +
2][ L_black[RowNum]] == 1)
                                    &&      (P_Pixels[RowNum      +
3][L_black[RowNum]] == 1)
                                    &&      (P_Pixels[RowNum      +
4][L_black[RowNum]] == 1)
                                    &&      (P_Pixels[RowNum      +
```

```
5][L_black[RowNum]] == 1))          //尖角向上5个点为白点//尖角上一行中线找跳变
                                {
                                    pre_left_corner_flag = 1;
                                    left_corner_rownum = (byte)(RowNum - 1);
                                }
                            }
                        }

                        if ((RowNum <= 40) && (RowNum > 2))
                        {
                            if ((R_black[RowNum] <= R_black[RowNum - 1])
                                && (R_black[RowNum - 1] > R_black[RowNum - 2])
                                && (R_black[RowNum - 2] > R_black[RowNum - 3]))
                            {
                                if ((P_Pixels[RowNum + 1][R_black[RowNum]] == 1)
                                    && (P_Pixels[RowNum + 2][R_black[RowNum]] == 1)
                                    && (P_Pixels[RowNum + 3][R_black[RowNum]] == 1)
                                    && (P_Pixels[RowNum + 4][R_black[RowNum]] == 1)
                                    && (P_Pixels[RowNum + 5][R_black[RowNum]] == 1))
                                {
                                    pre_right_corner_flag = 1;
                                    right_corner_rownum = RowNum;
                                }
                            }
                        }
```

```
            if((RowNum > 4) && (RowNum < 65))
            {
                    if((P_Pixels[RowNum][(byte)Sv_OutLine(LCenter[RowNum] -
1)] == 0)
                            &&
(P_Pixels[RowNum][(byte)Sv_OutLine(LCenter[RowNum] + 1)] == 0))
                    {
                            cut_rownum = (byte)(RowNum - 3);
                            cut_flag = 1;
                    }
                    if(cut_flag == 1 && cut_rownum >= 10)
                    {
                            for (RowNum = (byte)(cut_rownum); RowNum < 70;
RowNum++)
                            {
                                    if ((L_flag == 1) && (R_flag == 0))
                                    {
                                            LCenter[RowNum] = 198;
                                            L_black[RowNum] = 198;
                                            R_black[RowNum] = 198;
                                            L_start[RowNum] = 198;
                                            R_start[RowNum] = 198;
                                    }
                                    if ((L_flag == 0) && (R_flag == 1))
                                    {
                                            LCenter[RowNum] = 1;
                                            L_black[RowNum] = 1;
                                            R_black[RowNum] = 1;
                                            L_start[RowNum] = 1;
                                            R_start[RowNum] = 1;
                                    }
```

```
                        if ((L_flag == 1) && (R_flag == 1))
                        {
                                LCenter[RowNum] = 100;
                                L_black[RowNum] = 100;
                                R_black[RowNum] = 100;
                                L_start[RowNum] = 100;
                                R_start[RowNum] = 100;
                        }
                        if ((L_flag == 0) && (R_flag == 0))
                        {
                                LCenter[RowNum] = 100;
                                L_black[RowNum] = 100;
                                R_black[RowNum] = 100;
                                L_start[RowNum] = 100;
                                R_start[RowNum] = 100;
                        }
                }
        }
}

if(RowNum <= 6)
{
        if(L_black[RowNum] < R_black[RowNum])
        {
                runout_count++;
        }
}

if(runout_count > 20)
{
        runout_flag = 0;            //保护关
```

```
        }

    }


    /*********************************************
    函数名：SignalProcess()

    功能：信息处理
    *********************************************/
    void SignalProcess()
    {
        Image_Init();//变量初始化
        if(flag2 == 1)
        {
            for (RowNum = 0; RowNum < Image_Height; RowNum++) //循环叠加
                {
                    ImageProcess();                    //图像行处理  内联函数
                }
            flag2++;
        }

        /*****************************************
         *图像场处理
         * ***************************************/
        for (RowNum = 0; RowNum < 70; RowNum++)        //循环叠加
        {
            ImageProcess();                            //图像行处理  内联函数
        }
```

```
        if(cut_flag == 0)
        {
            cut_rownum = 69;
        }

        i = (byte)Sv_OutLine((LCenter[2] + LCenter[3] + LCenter[4] +
LCenter[5] +
                    LCenter[6] + LCenter[7] + LCenter[8] + LCenter[9]) >>
3);
        for (RowNum = 2;RowNum < 65 ;RowNum++)
        {
            if (P_Pixels[RowNum][i] == 1 &&
P_Pixels[RowNum][(byte)Sv_OutLine(i + 1)] == 1 &&
                P_Pixels[RowNum][(byte)Sv_OutLine(i + 2)] == 1
&& P_Pixels[RowNum][(byte)Sv_OutLine(i -2)] == 1 &&
                P_Pixels[RowNum][(byte)Sv_OutLine(i - 1)] == 1)
            {
                z_speedup_count++;
            }
            if(cut_flag == 0 || cut_rownum >= 40)
            {
                if ((RowNum > 10) && (RowNum < 30))          //不
能影响小 S 的判定
                {
                    if ((LCenter[RowNum] >= 175) ||
(LCenter[RowNum] <= 25)
                        || (L_black[RowNum] >= 185) ||
(R_black[RowNum] <= 15))
                    {
                        z_nearboder_count++;
                    }
                }
```

```
                }
        }
        if (((cut_flag == 1) && (cut_rownum <= 50))
                || ((L_smallcurve_flag == 1) && (L_smallcurve_rownum >= 20)
&& (L_smallcurve_rownum < 48))
                || ((R_smallcurve_flag == 1) && (R_smallcurve_rownum >=
20) && (R_smallcurve_rownum < 48)))        //直道充分加速//防止弯道加速
        {
                z_speedup_count = 0;
        }
        if(z_speedup_count >= 40)     //直道允许检测停车标志
        {
                stop_signal_permission_flag = 1;
        }

        if(z_speedup_count >= 48)
        {
                z_speedup_flag = 1;
        }

        if(LCenter[60] > LCenter[15])
        {
                if((LCenter[60] - LCenter[15]) >= 22)
                {
                        z_speedup_flag = 0;
                }
        }
        else
            if(LCenter[15] > LCenter[60])
            {
                    if((LCenter[15] - LCenter[60]) >= 22)
                    {
```

```
                                z_speedup_flag = 0;
                        }
                    }
                if(z_nearboder_count >= 13)
                {
                        z_speedup_flag = 0;
                }


            /*****************************************
            *坡道
            * ****************************************/
                if(z_speedup_flag == 1)
                {
                        for(i = 39;i <= 55;i++)
                        {
                                if(L_black[i] > R_black[i])
                                {
                                        if((R_black[i] >= 4) && (L_black[i] <= 196))
                                        {
                                                if((L_black[i] - R_black[i]) > fix[i])
                                                {
                                                        if(((L_black[i]  -  R_black[i])
- fix[i]) >= 15)
                                                        {
                                                                ramp_fix_count++;
                                                        }
                                                }
                                        }
                                }
                                else
```

```
                {
                        ramp_fix_count = 0;
                        break;
                }
        }
}

if(ramp_flag == 1)
{
        for(i = (byte)(cut_rownum - 1);i > 5;i--)
        {
                if(L_black[i] < L_black[i - 1])
                {
                        if((L_black[i - 1] - L_black[i]) >= 5)
                        {
                                ramp_leftit_flag = 1;
                        }
                }
                else
                {
                        if((L_black[i] - L_black[i - 1]) >= 5)
                        {
                                ramp_leftit_flag = 1;
                                ramp_flag = 1;
                                ramp_leftit_rownum = i;
                        }
                }

                if(R_black[i] > R_black[i - 1])
                {
                        if((R_black[i] - R_black[i - 1]) >= 5)
                        {
```

```
                                        ramp_rightit_flag = 1;
                                        //setText 用户自定义("找到坡道右
边突变" + ramp_flag);
                                        ramp_flag = 1;
                                        ramp_rightit_rownum = i;
                                }
                        }
                        else
                        {
                                if((R_black[i - 1] - R_black[i]) >= 5)
                                {
                                        ramp_rightit_flag = 1;
                                        //setText 用户自定义("找到坡道右
边突变" + ramp_flag);
                                        ramp_flag = 1;
                                        ramp_rightit_rownum = i;
                                }
                        }

                        if((ramp_rightit_flag == 1) && (ramp_leftit_flag ==
1))
                        {
                                if(ramp_leftit_rownum >=
ramp_rightit_rownum)
                                {
                                        if((ramp_leftit_rownum -
ramp_rightit_rownum) <= 4)
                                        {
                                                ramp_flag = 1;
                                                break;
                                        }
                                }
```

```
                    }
                }
            }


            stop_count++;
            if (stop_signal_permission_flag == 1)
            {
                for (RowNum = 0; RowNum <= 20; RowNum++)
                {
                    if (RowNum <= 10)
                    {

    if((P_Pixels[RowNum][(byte)Sv_OutLine(L_black[RowNum] - 30)] == 0)
                                        &&
(P_Pixels[RowNum][(byte)Sv_OutLine(L_black[RowNum] - 5)] == 1))
                                    {
                                        if (P_Pixels[RowNum +
6][(byte)Sv_OutLine(L_black[RowNum] - 30)] == 1)
                                        {

    stop_left_black_dot_count++;
                                        }
                                    }

    if((P_Pixels[RowNum][(byte)Sv_OutLine(L_black[RowNum] - 40)] == 0)
                                        &&
(P_Pixels[RowNum][(byte)Sv_OutLine(L_black[RowNum] - 5)] == 1))
                                    {
                                        if (P_Pixels[RowNum +
6][(byte)Sv_OutLine(L_black[RowNum] - 40)] == 1)
                                        {
```

```
    stop_left_black_dot_count++;
                                        }
                                    }
                                    if
((P_Pixels[RowNum][(byte)Sv_OutLine(L_black[RowNum] - 50)] == 0)
                                        &&
(P_Pixels[RowNum][(byte)Sv_OutLine(L_black[RowNum] - 5)] == 1))



                                    if
((P_Pixels[RowNum][(byte)Sv_OutLine(R_black[RowNum] + 30)] == 0)
                                        &&
(P_Pixels[RowNum][(byte)Sv_OutLine(R_black[RowNum] + 5)] == 1))
                                    {
                                        if(P_Pixels[RowNum                  +
6][(byte)Sv_OutLine(R_black[RowNum] + 30)] == 1)
                                        {

    stop_right_black_dot_count++;
                                        }
                                    }
                                    if
((P_Pixels[RowNum][(byte)Sv_OutLine(R_black[RowNum] + 38)] == 0)
                                        &&
(P_Pixels[RowNum][(byte)Sv_OutLine(R_black[RowNum] + 5)] == 1))
                                    {
                                        if(P_Pixels[RowNum                  +
6][(byte)Sv_OutLine(R_black[RowNum] + 38)] == 1)
                                        {

    stop_right_black_dot_count++;
                                        }
                                    }
                                    if
((P_Pixels[RowNum][(byte)Sv_OutLine(R_black[RowNum] + 46)] == 0)
```

```
                                        &&
(P_Pixels[RowNum][(byte)Sv_OutLine(R_black[RowNum] + 5)] == 1))
                                    {
                                        if (P_Pixels[RowNum + 
6][(byte)Sv_OutLine(R_black[RowNum] + 46)] == 1)
                                        {

    stop_right_black_dot_count++;
                                        }
                                    }
                                }
                                else
                                {
                                    if
((P_Pixels[RowNum][(byte)Sv_OutLine(L_black[RowNum] - 45)] == 0)
                                        &&
(P_Pixels[RowNum][(byte)Sv_OutLine(L_black[RowNum] - 5)] == 1))
                                    {
                                        if (P_Pixels[RowNum + 
6][(byte)Sv_OutLine(L_black[RowNum] - 45)] == 1)
                                        {

    stop_left_black_dot_count++;
                                        }
                                    }
                                    if
((P_Pixels[RowNum][(byte)Sv_OutLine(L_black[RowNum] - 55)] == 0)
                                        &&
(P_Pixels[RowNum][(byte)Sv_OutLine(L_black[RowNum] - 5)] == 1))
                                    {
                                        if (P_Pixels[RowNum + 
6][(byte)Sv_OutLine(L_black[RowNum] - 55)] == 1)
                                        {

    stop_left_black_dot_count++;
```

```
                    }
                }
                if
((P_Pixels[RowNum][(byte)Sv_OutLine(L_black[RowNum] - 60)] == 0)
                    &&
(P_Pixels[RowNum][(byte)Sv_OutLine(L_black[RowNum] - 5)] == 1))
                {
                    if (P_Pixels[RowNum +
6][(byte)Sv_OutLine(L_black[RowNum] - 60)] == 1)
                    {

    stop_left_black_dot_count++;
                    }
                }



                if
((P_Pixels[RowNum][(byte)Sv_OutLine(R_black[RowNum] + 38)] == 0)
                    &&
(P_Pixels[RowNum][(byte)Sv_OutLine(R_black[RowNum] + 5)] == 1))
                {
                    if (P_Pixels[RowNum +
6][(byte)Sv_OutLine(R_black[RowNum] + 38)] == 1)
                    {

    stop_right_black_dot_count++;
                    }
                }
                if
((P_Pixels[RowNum][(byte)Sv_OutLine(R_black[RowNum] + 46)] == 0)
                    &&
(P_Pixels[RowNum][(byte)Sv_OutLine(R_black[RowNum] + 5)] == 1))
                {
                    if (P_Pixels[RowNum +
6][(byte)Sv_OutLine(R_black[RowNum] + 46)] == 1)
```

```
                                                {

    stop_right_black_dot_count++;
                                                }
                                            }
                                            if
((P_Pixels[RowNum][(byte)Sv_OutLine(R_black[RowNum] + 50)] == 0)
                                                &&
(P_Pixels[RowNum][(byte)Sv_OutLine(R_black[RowNum] + 5)] == 1))
                                            {
                                                if (P_Pixels[RowNum +
6][(byte)Sv_OutLine(R_black[RowNum] + 50)] == 1)
                                                {

    stop_right_black_dot_count++;
                                                }
                                            }
                                        }
                                    }
                                    if((stop_left_black_dot_count >= 1) &&
(stop_right_black_dot_count >= 1) && (stop_count >= 100))
                                    {
                                        stop_flag = 1;
                                        stop_signal_permission_flag = 0;
                                    }
                                }
                                if(stop_count >= 120)
                                {
                                    stop_count = 119;
                                }



    //          if(z_speedup_flag == 0)         //大回弯
    //          {
```

```
//                  if((cut_rownum <= 50) && (cut_rownum >= 40))
//                  {
//                          big_curve_count++;
//                  }
//                  else
//                  if(big_curve_count >= 1)
//                  {
//                          big_curve_count = 0;
//                  }
//                  if(big_curve_count >= 10)
//                  {
//                          big_curve_flag = 1;
//                  }

//          }
//          if(big_curve_flag == 1 )
//          {
//                  if((cut_rownum <= 52) && (cut_rownum >= 38))
//                  {
//                          big_curve_flag = 1;
//                  }else
//                  {
//                          big_curve_count = 0;
//                          big_curve_flag = 0;
//                  }
//          }
          if((fence_left_count >= 10) && (fence_right_it_count >= 2))
          {
                  if ((fence_start_rownum <= 54) && (fence_right_it_count < 4))
                  {
                          fence_flag = 1;
```

```
                    if(fence_flag == 1)        //防止停车线误判
                    {
                        {
                            if(L_black[i] > R_black[i])
                            {
                                if((L_black[i] - R_black[i])
< (fix[i]))
                                {
                                    if ((fix[i] -
(L_black[i] - R_black[i])) >= 15)
                                    {

    fence_fix_count++;
                                    }
                                }
                            }
                        }
                        if(fence_fix_count <= 1)
                        {
                            fence_flag = 0;
                            fence_handle_premission_flag = 1;
                        }
                    }
                    if((fence_flag == 1) &&
(fence_handle_premission_flag == 0))
                    {
                        far_right_fence_flag = 1;
                        for(i = (byte)(fence_start_rownum + 15);i >
10;i--)
                        {
                            LCenter[i] = (byte)(L_start[i] - 20);
                        }
                    }
```

```
                }
            }
            else
                if((fence_right_count >= 15) && (fence_left_it_count < 4))
                {
                    if((fence_start_rownum <= 54) && (fence_left_it_count >= 2))
                    {
                        fence_flag = 1;
                        if (fence_flag == 1)        //防止停车线误判
                        {
                            {
                                if (L_black[i] > R_black[i])
                                {
                                    if ((L_black[i] - R_black[i]) < (fix[i]))
                                    {
                                        if ((fix[i] - (L_black[i] - R_black[i])) >= 15)
                                        {
    fence_fix_count++;
                                        }
                                    }
                                }
                            }
                            if (fence_fix_count <= 1)
                            {
                                fence_flag = 0;
    fence_handle_premission_flag = 1;
                            }
```

```
                                        }
                                        if((fence_flag == 1) &&
(fence_handle_premission_flag == 0))
                                        {
                                                for (i = (byte)(fence_start_rownum +
15); i > 10; i--)
                                                {
                                                        LCenter[i] = (byte)(R_start[i]
+ 20);
                                                }
                                        }

                                }
                        }

                if ((fence_left_count >= 18) && (near_right_fence_pre_flag == 1)) //近
处右边障碍
                {
                        near_right_fence_flag = 1;
                        if ((near_fence_end_rownum <= 40))
                        {
                                fence_flag = 1;
                                if (fence_flag == 1)        //防止停车线误判
                                {
                                        {
                                                if (L_black[i] > R_black[i])
                                                {
                                                        if ((L_black[i] - R_black[i])
< (fix[i]))
                                                        {
                                                                if ((fix[i] -
(L_black[i] - R_black[i])) >= 15)
                                                                {
```

```
fence_fix_count++;
                                                    }
                                                }
                                            }
                                        }
                                        if (fence_fix_count <= 1)
                                        {
                                            fence_flag = 0;
                                            fence_handle_premission_flag = 1;
                                        }
                                    }
                                    if((fence_flag == 1) && (fence_handle_premission_flag == 0))
                                    {
                                        far_right_fence_flag = 1;
                                        for(i = (byte)(near_fence_end_rownum + 15);i > 10;i--)
                                        {
                                            LCenter[i] = (byte)(L_start[i] - 20);
                                        }
                                    }

                                }

                            }
                            else
                                if ((fence_right_count >= 15) && (near_left_fence_pre_flag == 1))
                                {
                                    near_left_fence_flag = 1;
                                    if (near_fence_end_rownum <= 40)
                                    {
```

```
                                    fence_flag = 1;
                                    if (fence_flag == 1)
                {
                                        {
                                            if (L_black[i] > R_black[i])
                                            {
                                                if    ((L_black[i]    -
R_black[i]) < (fix[i]))
                                                {
                                                    if  ((fix[i]  -
(L_black[i] - R_black[i])) >= 15)
                                                    {

    fence_fix_count++;
                                                    }
                                                }
                                            }
                                        }
                                        if (fence_fix_count <= 1)
                                        {
                                            fence_flag = 0;

    fence_handle_premission_flag = 1;
                                        }
                                    }
                                    if((fence_flag          ==          1)          &&
(fence_handle_premission_flag == 0))
                                    {
                                        for                 (i                  =
(byte)(near_fence_end_rownum + 15); i > 10; i--)
                                        {
                                            LCenter[i]  =  (byte)(R_start[i]
+ 20);
```

```
                                        }
                                }
                        }
                }

        /*****************************
        *4.13  右线太靠近左边是不判定小弯道//防止压路肩
        *****************************/
        if((L_smallcurve_flag == 1) && (fence_flag == 0))
        {
                for (j = L_smallcurve_rownum; j < L_smallcurve_rownum + 3;j ++)    //丢线行向上三行找左右线
                {
                        for (i = LCenter[j]; i < 198; i++)
                        {
                                if (P_Pixels[j][i] == 0)
                                {
                                        L_black[j] = i;
                                        break;
                                }
                                else
                                        if((i >=197) && (P_Pixels[j][i] == 1))
                                        {
                                                L_black[j] = 198;
                                        }
                        }
                        for (i = LCenter[j]; i > 2; i--)
                        {
                                if (P_Pixels[j][i] == 0)
```

```
                                {
                                        R_black[j] = i;
                                        break;
                                }
                        }
                }

                if ((L_black[L_smallcurve_rownum] > R_black[L_smallcurve_rownum]) && (R_black[L_smallcurve_rownum] <= 160))
                {
                        smallcurve_fix = (byte)(L_black[L_smallcurve_rownum] - R_black[L_smallcurve_rownum] + 5);
                        if ((L_black[L_smallcurve_rownum + 1] > R_black[L_smallcurve_rownum])
                                && (L_black[L_smallcurve_rownum + 2] > R_black[L_smallcurve_rownum]))
                        {
                                if (((L_black[L_smallcurve_rownum + 1] - R_black[L_smallcurve_rownum + 1]) <= smallcurve_fix) &&
                                        ((L_black[L_smallcurve_rownum + 1] - R_black[L_smallcurve_rownum + 2]) <= smallcurve_fix))
                                {
                                        L_repair_flag = 1;
                                }
                        }
                }
        }

        /**********************
          * ********************/
        if((R_smallcurve_flag == 1) && (fence_flag == 0))
        {
                for(j = R_smallcurve_rownum;j < R_smallcurve_rownum + 3;j
```

```
++)
                {
                        for(i=LCenter[j];i > 2;i --)
                        {
                                if(P_Pixels[j][i] == 0)
                                {
                                        R_black[j] = i;
                                        break;
                                }
                                else
                                        if((i <= 3) && (P_Pixels[j][i] == 1))
                                        {
                                                R_black[j] = 2;
                                        }
                        }
                        for(i=LCenter[j];i < 198;i ++)
                        {
                                if(P_Pixels[j][i] == 0)
                                {
                                        L_black[j] = i;
                                        break;
                                }
                        }
                }


                if      ((L_black[R_smallcurve_rownum]      >
R_black[R_smallcurve_rownum]) && (L_black[RowNum] >= 40))
                {
                        smallcurve_fix                              =
(byte)(L_black[R_smallcurve_rownum] - R_black[R_smallcurve_rownum]);
                        if    ((L_black[R_smallcurve_rownum    +    1]    >
R_black[R_smallcurve_rownum + 1])
```

```
                                    && (L_black[R_smallcurve_rownum + 2] > R_black[R_smallcurve_rownum + 2]))
                                {
                                    if (((L_black[R_smallcurve_rownum + 1] - R_black[R_smallcurve_rownum + 1]) <= smallcurve_fix) &&
                                        ((L_black[R_smallcurve_rownum + 2] - R_black[R_smallcurve_rownum + 2]) <= smallcurve_fix))
                                    {
                                        R_repair_flag = 1;
                                    }
                                }
                            }
                        }

                        /****************************************
                         * ****************************************/
                        if ((L_repair_flag == 1) && (cut_rownum > L_smallcurve_rownum) && (cut_flag == 1))
                        {
                            if((cut_rownum - L_smallcurve_rownum) > 3)
                            {
                                for (RowNum = L_smallcurve_rownum; RowNum >= 2; RowNum--)
                                {
                                    LCenter[RowNum] = (byte)Sv_OutLine((R_black[RowNum] + L_black[RowNum]) >> 1);
                                }
                            }
                        }
                        else
                            if((cut_flag == 0) && (L_repair_flag == 1))
                            {
                                for (RowNum = L_smallcurve_rownum; RowNum >= 2; RowNum--)
```

```
                    {
                        LCenter[RowNum] = (byte)Sv_OutLine((R_black[RowNum] + L_black[RowNum]) >> 1);
                    }
                }


            /*****************************************
             * ****************************************/
            if ((R_repair_flag == 1) && (cut_rownum > R_smallcurve_rownum) && (cut_flag == 1))
            {
                if((cut_rownum - R_smallcurve_rownum) > 3)
                {
                    //setText 用户自定义("截断补右小弯" + R_smallcurve_rownum);
                    for (RowNum = R_smallcurve_rownum; RowNum >= 2; RowNum--)
                    {
                        LCenter[RowNum] = (byte)Sv_OutLine((R_black[RowNum] + L_black[RowNum]) >> 1);
                    }
                }
            }
            else
                if((cut_flag == 0) && (R_repair_flag == 1))        //未截断时
                {
                    for (RowNum = R_smallcurve_rownum; RowNum >= 2; RowNum--)
                    {
                        LCenter[RowNum] = (byte)Sv_OutLine((R_black[RowNum] + L_black[RowNum]) >> 1);
                    }
```

```
                }


                /*************************
                * ***********************/
            if((L_straightway_count >= 25) || (R_straightway_count >= 25))
//出弯接直道拟中线
            {
                if ((L_straightway_it_count <= 4) && (R_straightway_it_count
<= 4))
                {                       for (RowNum = 55; RowNum >= 1;
RowNum--)
                    {
                        LCenter[RowNum]                         =
(byte)Sv_OutLine((R_black[RowNum] + L_black[RowNum]) >> 1);
                    }
                    curve_to_straightway_sppedup_flag = 1;          //
弯接直道加速
                }
            }


            /*************************
          * ***********************/
            if(S_curve_flag == 0)
            {
                if((S_leftmax_it_flag == 1) && (S_rightmax_it_flag == 1))
//出现极大尖角
                {
                    if(S_right_it_rownum >= S_left_it_rownum)
                    {
                        if((S_right_it_rownum - S_left_it_rownum)
<= 8)
                        {
                            S_curve_flag = 1;
```

```
                }
            }
            else
                if((S_left_it_rownum - S_right_it_rownum)
<= 8)
                {
                    S_curve_flag = 1;
                }
        }
        else
            if((S_leftmin_it_flag == 1) && (S_rightmin_it_flag ==
1))    //出现极小尖角
            {
                if (S_right_it_rownum >= S_left_it_rownum)
                {
                    if ((S_right_it_rownum -
S_left_it_rownum) <= 8)
                    {
                        S_curve_flag = 1;
                    }
                }
                else
                    if ((S_left_it_rownum -
S_right_it_rownum) <= 8)
                    {
                        S_curve_flag = 1;
                    }
            }
            else
                if ((S_leftmax_it_flag == 1) &&
(S_leftmin_it_flag == 1))
                {
                    S_curve_flag = 1;
```

```
                    }
                else
                    if((S_rightmax_it_flag == 1) &&
(S_rightmin_it_flag == 1))
                    {
                        S_curve_flag = 1;
                    }
        }
        if (S_curve_flag == 1)
        {
            if(cut_flag == 1)
            {
                if(cut_rownum <= 63)
                {
                    S_curve_flag = 0;
                }
            }
            else
                if(z_speedup_count <= 20)
                {
                    S_curve_flag = 0;
                }
                else
                    if(((L_smallcurve_flag == 1) &&
(L_smallcurve_rownum >= 20))
                    || ((R_smallcurve_flag == 1) &&
(R_smallcurve_rownum >= 20)))
                    {
                        S_curve_flag = 0;
                    }
        }
```

```
/****************************************
        * 将尖角的扫线范围锁定在尖角和尖角中线之间//ok//可能要加边线连续条件//neg//连续两点之差小于某个值//neg
        * 当尖角小于某个值时，扫线从 2 开始 //3-16//possi
        * 3.30  将尖角行范围由 50 行减小到 40 行
        * 3.30 在找尖角（不是跳变）的时候，应该要加一个 fix 保护!!!
        * 5.1 提高到 42
        * 5.8 障碍不判十字!!
       * ****************************************/
      if ((pre_left_corner_flag == 1) && (left_corner_rownum <= 42) && (fence_flag == 0))
      {
            if ((left_corner_rownum <= 42) && (left_corner_rownum > 30))
            {
                  for (j = (byte)(left_corner_rownum + 5); j <= 60; j++)
                  {
                        for (s = (byte)Sv_OutLine(L_black[left_corner_rownum] - ((L_black[left_corner_rownum]) >> 1));
                              s < L_black[left_corner_rownum]; s++)
                        {
                              if ((P_Pixels[j][s] == 0) && (P_Pixels[j][s + 1] != 1))
                              {
                                    leftjump_leftdot = s;
                                    left_corner_jump_flag = 1;
                                    leftjump_searchright_flag = 1;
                                    break;
                              }
                        }
```

```
                    }
                }
                else
                    if ((left_corner_rownum <= 30) && (left_corner_rownum > 20))
                    {
                        for (j = (byte)(left_corner_rownum + 8); j <= 60; j++)
                        {
                            for (s = (byte)Sv_OutLine(L_black[left_corner_rownum] - ((L_black[left_corner_rownum]) >> 1));
                                s < L_black[left_corner_rownum]; s++)
                            {
                                if ((P_Pixels[j][s] == 0) && (P_Pixels[j][s + 1] != 1))
                                {
                                    leftjump_leftdot = s;

    left_corner_jump_flag = 1;

    leftjump_searchright_flag = 1;
                left_corner_jump_rownum = j;
                                    break;
                                }
                            }
                            if (leftjump_searchright_flag == 1)
                            {
                                leftjump_end_rownum = j;
                                //setText 用户自定义("跳变行" + j);

                                break;
```

```
                                        }
                                    }
                                }
                                else
                                    if (left_corner_rownum <= 20)
                                    {
                                        for (j = (byte)(left_corner_rownum +
18); j <= 60; j++)
                                        {
                                            for (s =
(byte)Sv_OutLine(L_black[left_corner_rownum] -
((L_black[left_corner_rownum]) >> 1));
                                                s <
L_black[left_corner_rownum]; s++)
                                            {
                                                if ((P_Pixels[j][s]
== 0) && (P_Pixels[j][s + 1] != 1))
                                                {

    leftjump_leftdot = s;
                left_corner_jump_flag = 1;
                    leftjump_searchright_flag = 1;
                        left_corner_jump_rownum = j;      //
左跳变所在行
                                                    break;
                                                }
                                            }
                                            if (leftjump_searchright_flag
== 1)
                                            {

    leftjump_end_rownum = j;
                                                break;
                                            }
                                        }
```

```
                                        }
                }

                if (leftjump_searchright_flag == 1)          //找到左跳变后找右线//找到左右线之后加 fix 保护//neg
                {
                        if (leftjump_leftdot <= 50)             //左跳变小于 50 默认右线为 2
                        {
                                leftjump_rightdot = 2;
                                leftjump_rightconfirm_flag = 1; //确认找到右线标志 leftjump_rightdot);
                        }
                        else
                                if ((leftjump_leftdot > 50) && (leftjump_leftdot <= 199))
                                {
                                        for (j = (byte)(left_corner_jump_rownum + 3); j < left_corner_jump_rownum + 6; j++)          //跳变向上扫三行找右线
                                        {
                                                //在直入十字的情况下还是有 bug
                                                {
                                                        if ((P_Pixels[j][s] == 0) && (P_Pixels[j][s - 1] != 1))       //二次滤波
                                                        {
                                                                leftjump_rightdot = s;

    leftjump_rightconfirm_flag = 1;
                                                                break;
                                                        }
                                                        else
                                                                if(s >= 197)
                                                                {
```

```
leftjump_rightdot = s;

leftjump_rightconfirm_flag = 1;
                                                    break;
                                                }
                                            }
                                            if (leftjump_rightconfirm_flag == 1)
                                            {
                                                break;
                                            }

                                        }
                                    }

                }
                /*************************************
                 * 找右跳变
                * ***********************************/
                if ((pre_right_corner_flag == 1) && (right_corner_rownum <= 42) &&
(fence_flag == 0))
                {
                    if ((right_corner_rownum <= 42) && (right_corner_rownum >
30))
                    {
                        for (j = (byte)(right_corner_rownum + 5); j < 60; j++)
                        {
                            for (s = (byte)Sv_OutLine(((L_black[right_corner_rownum] +
R_black[right_corner_rownum]) >> 1));
                                s > R_black[right_corner_rownum];
s--)
                            {
```

```
                                    if ((P_Pixels[j][s] == 0) &&
(P_Pixels[j][s - 1] != 1))
                                    {
                                        rightjump_rightdot = s; //
记录右跳变
                                        right_corner_jump_flag = 1;
                                        break;
                                    }
                                }

                                if (rightjump_searchleft_flag == 1)
                                {
                                    rightjump_end_rownum = j;
                                    break;
                                }
                            }
                        }
                        else
                            if ((right_corner_rownum <= 30) &&
(right_corner_rownum > 20))
                            {
                                for (j = (byte)(right_corner_rownum + 20); j <
60; j++)
                                {
                                    for (s =
(byte)Sv_OutLine(((L_black[right_corner_rownum] +
R_black[right_corner_rownum]) >> 1));
                                        s >
R_black[right_corner_rownum]; s--)
                                    {
                                        {
                                            rightjump_rightdot
= s;  //记录右跳变
```

```
right_corner_jump_flag = 1;  //右跳变标志

rightjump_searchleft_flag = 1;

right_corner_jump_rownum = j;
                                                    break;
                                                }
                                            }

                                            if (rightjump_searchleft_flag == 1)
                                            {
                                                rightjump_end_rownum = j;
                                                break;
                                            }
                                        }
                                    }
                                    else
                                        if (right_corner_rownum <= 20)
                                        {
                                            for (j = (byte)(right_corner_rownum
+ 20); j < 60; j++)
                                            {
                                                for (s =
(byte)Sv_OutLine((((L_black[right_corner_rownum] +
R_black[right_corner_rownum]) >> 1)));
                                                    s >
R_black[right_corner_rownum]; s--)
                                                {
                                                    if ((P_Pixels[j][s]
== 0) && (P_Pixels[j][s - 1] != 1))
                                                    {

rightjump_rightdot = s;
            right_corner_jump_flag = 1;  //右跳变标志
```

```
rightjump_searchleft_flag = 1;

right_corner_jump_rownum = j;
                                                    break;
                                                }
                                            }

                                            if (rightjump_searchleft_flag == 1)

                                            {

rightjump_end_rownum = j;
                                                    break;
                                            }
                                        }
                                    }
                                }
                                if (rightjump_searchleft_flag == 1)
                                {
                                    if (rightjump_rightdot >= 150)        //右跳变大于 150  默认左线为 198
                                    {
                                        //Lcr[limit_right_jump_rownum].LBlack = 198;
                                        rightjump_leftdot = 198;
                                        rightjump_leftconfirm_flag = 1;
 rightjump_leftdot);
                                    }
                                    else
                                    {
                                        if ((rightjump_rightdot < 150) && (rightjump_rightdot >= 1))
                                        {
                                            for (j =
```

```
(byte)Sv_OutLine(right_corner_jump_rownum + 3); j < right_corner_jump_rownum
+ 6; j++)
                                    {
                                        for (s = (byte)(rightjump_rightdot +
15); s < 198; s++)
                                        {
                                            if (P_Pixels[j][s] == 0)       //
二次滤波
                                            {
                                                rightjump_leftdot =
s;

    rightjump_leftconfirm_flag = 1;
                                                break;
                                            }
                                            else
                                                if(s >= 197)
                                                {

    rightjump_leftdot = s;

    rightjump_leftconfirm_flag = 1; //确认找到左线标志

                                                    break;
                                                }
                                        }
                                        if (rightjump_leftconfirm_flag == 1)
                                        {
                                            break;
                                        }
                                    }
                                }
                            }
```

```
            }


            if ((left_corner_jump_flag == 1) && (left_corner_jump_rownum <= 60))
//确认跳变
            {
                    L_cross_flag = 1;
            }
            if ((right_corner_jump_flag == 1) && (right_corner_jump_rownum <=
60))
            {
                    R_cross_flag = 1;
            }

    /************************************************************
             * 十字处理
             * 尖角及跳变判定
             * 双边丢线计数大于某个值则判定为十字
             * 从近处 8 行中心均值向上找左右线

    ************************************************************/
            if (pre_right_corner_flag == 1)         //右尖角 fix 保护防止误判
            {
                    if(L_black[right_corner_rownum]                         >
R_black[right_corner_rownum])
                    {
                            rightcorner_fix                                 =
(byte)Sv_OutLine(L_black[right_corner_rownum]                           -
R_black[right_corner_rownum]);
                    }
                    rightcorner_upfix = (byte)((L_black[right_corner_rownum + 5]
- R_black[right_corner_rownum + 5]));
                    //setText 用户自定义("右尖角向上 5 行 fix 值" +
rightcorner_upfix);
```

```
                if(rightcorner_upfix >= rightcorner_fix)
                {
                        pre_right_corner_flag = (byte)(rightcorner_upfix - rightcorner_fix);
                        if((rightcorner_fix >= 40) && (pre_right_corner_flag >= 10))
                        {
                                pre_right_corner_flag = 1;
                        }
                        else
                        {
                                pre_right_corner_flag = 0;
                        }
                }
                else
                {
                        pre_right_corner_flag = 0;
                }
        }

        if (pre_left_corner_flag == 1)
        {
                if(L_black[left_corner_rownum] > R_black[left_corner_rownum])
                {
                        leftcorner_fix = (byte)Sv_OutLine(L_black[left_corner_rownum] - R_black[left_corner_rownum]);
                }
                leftcorner_upfix = (byte)((L_black[left_corner_rownum + 5] - R_black[left_corner_rownum + 5]));
                //setText 用户自定义("左尖角 fix" + leftcorner_fix);
                if(leftcorner_upfix >= leftcorner_fix)
```

```
                {
                        pre_left_corner_flag = (byte)(leftcorner_upfix - leftcorner_fix);
                        if((leftcorner_fix >= 40) && (pre_left_corner_flag >= 10))
                        {
                                pre_left_corner_flag = 1;
                        }
                        else
                        {
                                pre_left_corner_flag = 0;
                        }
                }
                else
                {
                        pre_left_corner_flag = 0;
                }
        }

         if ((L_cross_flag == 1) && (leftjump_rightconfirm_flag == 1) && (pre_left_corner_flag == 1))            //由左尖角补线
        {
                LCenter[leftjump_end_rownum] = (byte)Sv_OutLine((leftjump_leftdot + leftjump_rightdot) >> 1);
                left_corner_slope = ((float)(LCenter[leftjump_end_rownum] - LCenter[2])) / (leftjump_end_rownum - 2);
                for (i = 2; i < leftjump_end_rownum; i++)
                {
                        LCenter[i] = (byte)(LCenter[2] + (i - 2) * left_corner_slope);
                }
                if(left_corner_rownum <= 20)
                {
                        leftcorner_scanpermission_flag = 1; //左尖角补线后
```

```
允许扫线标志
                }
        }
        if ((R_cross_flag == 1) && (rightjump_leftconfirm_flag == 1) && (pre_right_corner_flag == 1))          //由右尖角补线
        {
                LCenter[rightjump_end_rownum] = (byte)Sv_OutLine((rightjump_leftdot + rightjump_rightdot) >> 1);
                right_corner_slope = ((float)(LCenter[rightjump_end_rownum] - LCenter[2])) / (rightjump_end_rownum - 2);
                for (i = 2; i < rightjump_end_rownum; i++)
                {
                        LCenter[i] = (byte)(LCenter[2] + (i - 2) * right_corner_slope);
                }
                if (right_corner_rownum <= 20)
                {
                        rightcorner_scanpermission_flag = 1; //右尖角补线后允许扫线标志
                }
        }

        if (Bothside_lost_count >= 20)
        {
                both_lost_startdot = (byte)Sv_OutLine((LCenter[2] + LCenter[3] + LCenter[4] + LCenter[5]
                                + LCenter[6] + LCenter[7] + LCenter[8] + LCenter[9]) >> 3);
                for (i = 3; i <= 46; i++)
                {
                        for (j = both_lost_startdot; j < 198; j++)
                        {
                                if ((P_Pixels[i][j] == 0) && (P_Pixels[i][j -
```

```
1] != 1))      //二次滤波
                        {
                                L_black[i] = j;
                                break;
                        }
                        if ((j >= 197) && (P_Pixels[i][j] == 1))
                        {
                                L_black[i] = 198;
                        }
                }
                for (j = both_lost_startdot; j > 2; j--)
                {
                        if ((P_Pixels[i][j] == 0) && (P_Pixels[i][j +
1] != 1))
                        {
                                R_black[i] = j;
                                break;
                        }
                        if ((j <= 3) && (P_Pixels[i][j] == 1))
                        {
                                R_black[i] = 2;
                        }
                }
                if (L_black[i] >= R_black[i]) //找到左右线符合赛道
fix,计数累加
                {
                if (both_lost_fix_count >= 2)
                {
                        both_lost_far_rownum = i;
                        both_lost_cross_flag = 1;
                        break;
                }
```

```
                }


                if ((both_lost_cross_flag == 1) && (both_lost_far_rownum > 2))
                {
                        bothlost_far_center = Sv_OutLine((R_black[both_lost_far_rownum] + L_black[both_lost_far_rownum]) >> 1);
    //                  both_lost_slope = ((float)(((R_black[both_lost_far_rownum] + L_black[both_lost_far_rownum]) >> 1) - LCenter[2])) / (both_lost_far_rownum - 2);
                        for (i = both_lost_far_rownum; i >= 2; i--)
                        {
    //                          LCenter[i] = (byte)(LCenter[2] + (i - 2) * both_lost_slope);
                                LCenter[i] = bothlost_far_center;
                        }
                }
        }


    /*************************************************************
             * 十字处理
             * 5.3  扫开始几行的线有 bug,不能沿用该场的左右线
            * 这种扫线方式应该注意和一半赛道区分，如果远处的行数比较近，则扫出的线符合一般赛道 fix 消除十字标志

    *************************************************************/
        if ((leftcorner_scanpermission_flag == 1) || (bothsidelost_scanpermission_flag == 1) || (rightcorner_scanpermission_flag == 1))
        {
                if(CrossScanLine_Stopflag == 0)
                {
                        if ((leftcorner_scanpermission_flag == 1) &&
```

```
(rightcorner_scanpermission_flag == 1))
                    {
                        for (RowNum = rightjump_end_rownum; RowNum <= 67; RowNum++)
                        {
                            CrossScanLine_mid();

                            CrossCut();

                            if (CrossScanLine_Stopflag == 1)
                            {
                                break;
                            }
                        }
                    }
                }

                if(CrossScanLine_Stopflag == 0)
                {
                    if (bothsidelost_scanpermission_flag == 1)
                    {
                        for (RowNum = both_lost_far_rownum; RowNum <= 67; RowNum++)
                        {
                            CrossScanLine_mid();

                            CrossCut();

                            if (CrossScanLine_Stopflag == 1)
                            {
                                break;
                            }
                        }
```

```
                }
            }

            if(CrossScanLine_Stopflag == 0)
            {
                if (leftcorner_scanpermission_flag == 1)
                {
                    for (RowNum = leftjump_end_rownum; RowNum <= 67; RowNum++)
                    {
                        CrossScanLine_mid();

                        CrossCut();

                        if (CrossScanLine_Stopflag == 1)
                        {
                            break;
                        }
                    }
                }
            }

            if(CrossScanLine_Stopflag == 0)
            {
                if (rightcorner_scanpermission_flag == 1)
                {
                    for (RowNum = rightjump_end_rownum; RowNum <= 67; RowNum++)
                    {
                        CrossScanLine_mid();

                        CrossCut();
```

```
                                        if (CrossScanLine_Stopflag == 1)
                                        {
                                                break;
                                        }
                                }
                        }
                }

}

/************************
*  弯道处理
************************/
if(z_speedup_flag == 0)         //弯道判定
{
        if ((cut_flag == 0) && (LCenter[55] > LCenter[45]))
        {
                if ((LCenter[55] - LCenter[45]) >= 20)
                {
                        curve_flag = 1;
                }
        }
        else
                if ((cut_flag == 0) && (LCenter[55] < LCenter[45]))
                {
                        if ((LCenter[45] - LCenter[55]) >= 20)
                        {
                                curve_flag = 1;
                        }
                }
}
```

```
        if((cut_flag == 1) && (cut_rownum <= Slowdown_Rownum) &&
(cut_rownum > 2))
        {
            curve_flag = 1;
        }

        if(cut_flag == 1)           //下位机防止中线窜道
        {
            for(RowNum = 5;RowNum < cut_rownum;RowNum ++)
            {
                if((L_black[RowNum] <= LCenter[RowNum]) ||
(R_black[RowNum] >= LCenter[RowNum]))
                {
                    LCenter[RowNum] = LCenter[RowNum - 1];
                }
            }
        }



        /************************
        **保护
        ************************/
        protect_count++;
        if(protect_count <=  start_protect_count)      //发车超调保护
        {
            z_speedup_flag = 0;
        }
        if(protect_count <= 200)      //前 200 场不检测停车线
        {
          runout_count = 0;
```

```
            stop_flag = 0;
        }
        else
        {
            protect_count = 210;
        }
                                        if(protect_flag == 1)       //保护
                                        {
                                            if(runout_flag == 1)
                                            {
                                                while(1)
                                                {
OLED_Write_String(0,0,(uint8_t*)"protect");
FTM_PWM_ChangeDuty(FTM0_CH4_PD4,0);
FTM_PWM_ChangeDuty(FTM0_CH3_PC4,0);
FTM_PWM_ChangeDuty(FTM2_CH0_PA10,servo_duty_max);
                                                }
                                            }
                                        }

                                        motor();

                                        if(ramp_flag == 1)
        {
                turn_row = 15;
        }
        if(cut_flag == 0)           //不截断正常打角行
        {
            turn_row = static_turnrow;
```

```
            Speed_Matching();
        }
        else
        {
            if((cut_rownum <= static_turnrow) && (cut_flag == 1) &&
(cut_rownum >= 4))
            {
                turn_row = cut_rownum - 1;
            }
            else
                if((cut_rownum > static_turnrow) && (cut_flag == 1))
                {
                    turn_row = static_turnrow;
                    Speed_Matching();
                }
        }
```

```
        if(turn_num > 0)      //如果这一场的打角行的中心值与上一场的打
角中心值相差大于某个值则打角值延续上一场
        {
            if(turn_num > LCenter[turn_row])
            {
                if((turn_num - LCenter[turn_row]) >= 25)
                {
                    temp = turn_num;
                }
            }
            else
```

```
                if(turn_num < LCenter[turn_row])
                {
                        if((LCenter[turn_row] - turn_num) >= 25)
                        {
                                temp = turn_num;
                        }
                }
}
turn_num = LCenter[turn_row];
temp = LCenter[turn_row];
error = (aim - temp);
if(curve_flag == 1)
{
        if(temp > aim)    //右转补偿
        {
                p += P_Cp;
        }
}

err = servo_duty_mid + ((p*error)/10) + servo_d*(error-err0)/10;
if(err > d_dt)
{
        if((err - d_dt) >= 750)
        {
                err = d_dt;
        }
        else
        {
                d_dt = err;
                err0 = error;
        }
}
```

```
else
{
        if(d_dt - err >= 750)
        {
                err = d_dt;
        }
        else
        {
                d_dt = err;
                err0 = error;
        }
}


p = nor_P;
servo_d = Nor_Servo_D;

if(err > servo_duty_max)       //往右打
{
    err = servo_duty_max;
}
else
    if(err < servo_duty_min)       //往左打
    {
        err = servo_duty_min;
    }
FTM_PWM_ChangeDuty(FTM2_CH0_PA10,err);       //舵机打角
d_dt = err;
                                SD_Write();
                                Pro_OK = 0;
                        }
```

```
            if(key3==0)
            {
                DelayMs(keydelay);
                if(key3==0)
                {
                    OLED_Clear();
                    break;
                }
            }

        }
    break;

    case 1:
        OLED_Clear();
        OLED_Write_String(0,0,(uint8_t*)"SDR");
        NVIC_DisableIRQ(DMA0_IRQn);
        NVIC_DisableIRQ(PORTA_IRQn);
        NVIC_DisableIRQ(PORTD_IRQn);
        MySD_Init();
        while(1)
        {
            SD_ReadData();
            UART_Send_Con();
            if(key3==0)
            {
                DelayMs(keydelay);
                if(key3==0)
                {
                    OLED_Clear();
                    break;
```

```
                }
            }
        }
    break;

    case 2:
        OLED_Clear();
        OLED_Write_String(0,0,(uint8_t*)"shangwj");
        while(1)
        {
            if(Pro_OK==1)       //整场采集完成
            {
                Pro_OK=0;
                UART_Send_Con();
            }
            if(key3==0)
            {
                DelayMs(keydelay);
                if(key3==0)
                {
                    OLED_Clear();
                    break;
                }
            }
        }
    break;

    case 0:
        OLED_Clear();
        OLED_Write_String(0,0,(uint8_t*)"RUN");
        while(1)
```

```
                {
                    if(Pro_OK == 1)       //整场采集完成
                    {
                        Pro_OK = 0;

                        SignalProcess();

                        if(protect_flag == 1)        //保护
                        {
                            if(runout_flag == 1)
                            {
OLED_Write_String(0,0,(uint8_t*)"Protect");
                                while(1)
                                {
FTM_PWM_ChangeDuty(FTM0_CH4_PD4,0);
FTM_PWM_ChangeDuty(FTM0_CH3_PC4,0);
FTM_PWM_ChangeDuty(FTM2_CH0_PA10,servo_duty_max);
                                }
                            }
                        }

                        motor();

                        Servo_Ctrl();

                        OLED_Write_Num5(2,2,turn_row);

                    }
                    if(key3 == 0)
```

```
                {
                    DelayMs(keydelay);
                    if(key3 == 0)
                    {
                        OLED_Clear();
                        break;
                    }
                }
            }
        break;

        case 4:
            OLED_Clear();
            OLED_Write_String(0,0,(uint8_t*)"StopLess");


            while(1)
            {
                if(Pro_OK == 1)       //整场采集完成
                {
                    Pro_OK = 0;

                    SignalProcess();

                    if(protect_flag == 1)       //保护
                    {
                        if(runout_flag == 1)
                        {

OLED_Write_String(0,0,(uint8_t*)"Protect");
                                while(1)
```

```
                                        {
FTM_PWM_ChangeDuty(FTM0_CH4_PD4,0);
FTM_PWM_ChangeDuty(FTM0_CH3_PC4,0);
FTM_PWM_ChangeDuty(FTM2_CH0_PA10,servo_duty_max);
                                        }
                                    }
                                }

                                stop_flag = 0;

                                motor();

                                Servo_Ctrl();


                            }
                            if(key3 == 0)
                            {
                                DelayMs(keydelay);
                                if(key3 == 0)
                                {
                                    OLED_Clear();
                                    break;
                                }
                            }
                        }

                    break;

                    case 5:
```

```
                    OLED_Clear();
                    OLED_Write_String(0,0,(uint8_t*)"CAME");
                    while(1)
                    {
                        if(Pro_OK == 1)
                        {
                            Pro_OK = 0;
                            trans(0,0);
                            Encoder_Pulse = DMACNT_GetValue(DMA_CH2);
                            Encoder_Dir = PCin(9);

    OLED_Write_Num5(2,2,Encoder_Pulse);

    OLED_Write_Num5(2,4,Encoder_Dir);
                            if(key3==0)
                            {
                                DelayMs(keydelay);
                                if(key3==0)
                                {
                                    OLED_Clear();
                                    break;
                                }
                            }
                        }

                    }

                break;

                case 6:
                    OLED_Clear();
```

```
                OLED_Write_String(0,0,(uint8_t*)"GryAcc");
                while(1)
                {
                        if(Pro_OK == 1)
                        {
                                Pro_OK = 0;

FTM_PWM_ChangeDuty(FTM0_CH4_PD4,0);

FTM_PWM_ChangeDuty(FTM0_CH3_PC4,0);     //反转
 //                                     trans(0,0);
 //
OLED_Write_Num5(2,0,Gyro_Data);
 //
OLED_Write_Num5(2,2,Accel_Data_2);
 //
OLED_Write_Num5(2,4,Accel_Data_1);
                                        if(key3==0)
                                        {
                                            DelayMs(keydelay);
                                            if(key3==0)
                                            {
                                                OLED_Clear();
                                                break;
                                            }
                                        }
                        }

                }

            break;

        }
```

```
            }
        }
    }

    while(count>4 && count<=8)
    {
        count_f();                              //count   加减
       OLED_Write_Char(0,colo[count%4],43);     //显示箭头
        OLED_Write_String(4,0,(uint8_t*)prm5);
        OLED_Write_Num4(10,0,prm5_val);

        OLED_Write_String(4,2,(uint8_t*)prm6);
        OLED_Write_Num4(10,2,prm6_val);

        OLED_Write_String(4,4,(uint8_t*)prm7);
        OLED_Write_Num4(10,4,prm7_val);

        OLED_Write_String(4,6,(uint8_t*)prm8);
        OLED_Write_Num4(10,6,prm8_val);
        switch(count%4)
        {
            case 1:
                if(key3==0)
                {
                    DelayMs(keydelay);
                    if(key3==0)
                    {
                        OLED_Clear();
                        prm5_val+=prm5_precision;
                        OLED_Write_String(4,0,(uint8_t*)prm5);
                        OLED_Write_Num4(10,0,prm5_val);
```

```
            }
        }
        if(key1==0)
        {
            DelayMs(keydelay);
            if(key1==0)
            {
                OLED_Clear();
                prm5_val-=prm5_precision;
                OLED_Write_String(4,0,(uint8_t*)prm5);
                OLED_Write_Num4(10,0,prm5_val);
            }
        }
        break;

    case 2:
        if(key3==0)
        {
            DelayMs(keydelay);
            if(key3==0)
            {
                OLED_Clear();
                prm6_val+=prm6_precision;
                OLED_Write_String(4,2,(uint8_t*)prm6);
                OLED_Write_Num4(10,2,prm6_val);
            }
        }
        if(key1==0)
        {
            DelayMs(keydelay);
            if(key1==0)
            {
```

```
                OLED_Clear();
                prm6_val-=prm6_precision;
                OLED_Write_String(4,2,(uint8_t*)prm6);
                OLED_Write_Num4(10,2,prm6_val);
            }
        }
        break;

    case 3:
        if(key3==0)
        {
            DelayMs(keydelay);
            if(key3==0)
            {
                OLED_Clear();
                prm7_val+=prm7_precision;
                OLED_Write_String(4,4,(uint8_t*)prm7);
                OLED_Write_Num4(10,4,prm7_val);
            }
        }
        if(key1==0)
        {
            DelayMs(keydelay);
            if(key1==0)
            {
                OLED_Clear();
                prm7_val-=prm7_precision;
                OLED_Write_String(4,4,(uint8_t*)prm7);
                OLED_Write_Num4(10,4,prm7_val);
            }
        }
```

```
            break;

        case 0:
            if(key3==0)
            {
                DelayMs(keydelay);
                if(key3==0)
                {
                    OLED_Clear();
                    prm8_val+=prm8_precision;
                    OLED_Write_String(4,6,(uint8_t*)prm8);
                    OLED_Write_Num4(10,6,prm8_val);
                }
            }
            if(key1==0)
            {
                DelayMs(keydelay);
                if(key1==0)
                {
                    OLED_Clear();
                    prm8_val-=prm8_precision;
                    OLED_Write_String(4,6,(uint8_t*)prm8);
                    OLED_Write_Num4(10,6,prm8_val);
                }
            }
            break;


    }
    if(key5==0)
    {
        DelayMs(keydelay);
```

```
            if(key5==0)
            {
                    OLED_Clear();
                    count = 1;
            }
        }

}

while(count > 8 && count <= 12)
{
    count_f();                              //count   加减
   OLED_Write_Char(0,colo[count%4],43);     //显示箭头
    OLED_Write_String(4,0,(uint8_t*)prm9);
    OLED_Write_Num4(10,0,prm9_val);

    OLED_Write_String(4,2,(uint8_t*)prm10);
    OLED_Write_Num4(10,2,prm10_val);

    OLED_Write_String(4,4,(uint8_t*)prm11);
    OLED_Write_Num4(10,4,prm11_val);

    OLED_Write_String(4,6,(uint8_t*)prm12);
    OLED_Write_Num4(10,6,prm12_val);
    switch(count%4)
    {
        case 1:
            if(key3==0)
            {
                DelayMs(keydelay);
                if(key3==0)
```

```
            {
                OLED_Clear();
                prm9_val+=prm9_precision;
                OLED_Write_String(4,0,(uint8_t*)prm9);
                OLED_Write_Num4(10,0,prm9_val);
            }
        }
        if(key1==0)
        {
            DelayMs(keydelay);
            if(key1==0)
            {
                OLED_Clear();
                prm9_val-=prm9_precision;
                OLED_Write_String(4,0,(uint8_t*)prm9);
                OLED_Write_Num4(10,0,prm9_val);
            }
        }
        break;

    case 2:
        if(key3==0)
        {
            DelayMs(keydelay);
            if(key3==0)
            {
                OLED_Clear();
                prm10_val+=prm10_precision;
                OLED_Write_String(4,2,(uint8_t*)prm10);
                OLED_Write_Num4(10,2,prm10_val);
            }
        }
```

```
        if(key1==0)
        {
            DelayMs(keydelay);
            if(key1==0)
            {
                OLED_Clear();
                prm10_val-=prm10_precision;
                OLED_Write_String(4,2,(uint8_t*)prm10);
                OLED_Write_Num4(10,2,prm10_val);
            }
        }
        break;

    case 3:
        if(key3==0)
        {
            DelayMs(keydelay);
            if(key3==0)
            {
                OLED_Clear();
                prm11_val+=prm11_precision;
                OLED_Write_String(4,4,(uint8_t*)prm11);
                OLED_Write_Num4(10,4,prm11_val);
            }
        }
        if(key1==0)
        {
            DelayMs(keydelay);
            if(key1==0)
            {
                OLED_Clear();
```

```
                prm11_val-=prm11_precision;
                OLED_Write_String(4,4,(uint8_t*)prm11);
                OLED_Write_Num4(10,4,prm11_val);
            }
        }
        break;

    case 0:
        if(key3==0)
        {
            DelayMs(keydelay);
            if(key3==0)
            {
                OLED_Clear();
                prm12_val+=prm12_precision;
                OLED_Write_String(4,6,(uint8_t*)prm12);
                OLED_Write_Num4(10,6,prm12_val);
            }
        }
        if(key1==0)
        {
            DelayMs(keydelay);
            if(key1==0)
            {
                OLED_Clear();
                prm12_val-=prm12_precision;
                OLED_Write_String(4,6,(uint8_t*)prm12);
                OLED_Write_Num4(10,6,prm12_val);
            }
        }
        break;
}
```

```
    if(key5==0)
    {
            DelayMs(60);
            if(key5==0)
            {
                    OLED_Clear();
                    count = 1;
            }
    }
}

while(count > 12 && count <= 16)
{
    count_f();                          //count   加减
  OLED_Write_Char(0,colo[count%4],43);    //显示箭头
    OLED_Write_String(4,0,(uint8_t*)prm13);
    OLED_Write_Num4(10,0,prm13_val);

    OLED_Write_String(4,2,(uint8_t*)prm14);
    OLED_Write_Num4(10,2,prm14_val);

    OLED_Write_String(4,4,(uint8_t*)prm15);
    OLED_Write_Num4(10,4,prm15_val);

    OLED_Write_String(4,6,(uint8_t*)prm16);
    OLED_Write_Num4(10,6,prm16_val);
    switch(count%4)
    {
        case 1:
            if(key3==0)
            {
```

```
            DelayMs(keydelay);
            if(key3==0)
            {
                OLED_Clear();
                prm13_val+=prm13_precision;
                OLED_Write_String(4,0,(uint8_t*)prm13);
                OLED_Write_Num4(10,0,prm13_val);
            }
        }
        if(key1==0)
        {
            DelayMs(keydelay);
            if(key1==0)
            {
                OLED_Clear();
                prm13_val-=prm13_precision;
                OLED_Write_String(4,0,(uint8_t*)prm13);
                OLED_Write_Num4(10,0,prm13_val);
            }
        }
        break;

    case 2:
        if(key3==0)
        {
            DelayMs(keydelay);
            if(key3==0)
            {
                OLED_Clear();
                prm14_val+=prm14_precision;
                OLED_Write_String(4,2,(uint8_t*)prm14);
                OLED_Write_Num4(10,2,prm14_val);
```

```
            }
        }
        if(key1==0)
        {
            DelayMs(keydelay);
            if(key1==0)
            {
                OLED_Clear();
                prm14_val-=prm14_precision;
                OLED_Write_String(4,2,(uint8_t*)prm14);
                OLED_Write_Num4(10,2,prm14_val);
            }
        }
        break;

    case 3:
        if(key3==0)
        {
            DelayMs(keydelay);
            if(key3==0)
            {
                OLED_Clear();
                prm15_val+=prm15_precision;
                OLED_Write_String(4,4,(uint8_t*)prm15);
                OLED_Write_Num4(10,4,prm15_val);
            }
        }
        if(key1==0)
        {
            DelayMs(keydelay);
            if(key1==0)
```

```
            {
                OLED_Clear();
                prm15_val-=prm15_precision;
                OLED_Write_String(4,4,(uint8_t*)prm15);
                OLED_Write_Num4(10,4,prm15_val);
            }
        }
        break;

    case 0:
        if(key3==0)
        {
            DelayMs(keydelay);
            if(key3==0)
            {
                OLED_Clear();
                prm16_val+=prm16_precision;
                OLED_Write_String(4,6,(uint8_t*)prm16);
                OLED_Write_Num4(10,6,prm16_val);
            }
        }
        if(key1==0)
        {
            DelayMs(keydelay);
            if(key1==0)
            {
                OLED_Clear();
                prm16_val-=prm16_precision;
                OLED_Write_String(4,6,(uint8_t*)prm16);
                OLED_Write_Num4(10,6,prm16_val);
            }
        }
```

```
                break;
        }
        if(key5==0)
        {
                DelayMs(60);
                if(key5==0)
                {
                        OLED_Clear();
                        count = 1;
                }
        }
}
while(count > 16 && count <= 20)
{
        count_f();                              //count   加减
      OLED_Write_Char(0,colo[count%4],43);     //显示箭头
        OLED_Write_String(4,0,(uint8_t*)prm17);
        OLED_Write_Num4(10,0,prm17_val);

        OLED_Write_String(4,2,(uint8_t*)prm18);
        OLED_Write_Num4(10,2,prm18_val);

        OLED_Write_String(4,4,(uint8_t*)prm19);
        OLED_Write_Num4(10,4,prm19_val);

        OLED_Write_String(4,6,(uint8_t*)prm20);
        OLED_Write_Num4(10,6,prm20_val);
        switch(count%4)
        {
            case 1:
                if(key3==0)
```

```
                {
                    DelayMs(keydelay);
                    if(key3==0)
                    {
                        OLED_Clear();
                        prm17_val+=prm17_precision;

//FTM_PWM_ChangeDuty(FTM0_CH4_PD4,prm17_precision);
                        OLED_Write_String(4,0,(uint8_t*)prm17);
                        OLED_Write_Num4(10,0,prm17_val);
                    }
                }
                if(key1==0)
                {
                    DelayMs(keydelay);
                    if(key1==0)
                    {
                        OLED_Clear();
                        prm17_val-=prm17_precision;

//FTM_PWM_ChangeDuty(FTM0_CH4_PD4,prm17_precision);
                        OLED_Write_String(4,0,(uint8_t*)prm17);
                        OLED_Write_Num4(10,0,prm17_val);
                    }
                }
                break;

            case 2:
                if(key3==0)
                {
                    DelayMs(keydelay);
                    if(key3==0)
                    {
```

```
                    OLED_Clear();
                    prm18_val+=prm18_precision;

//FTM_PWM_ChangeDuty(FTM0_CH3_PC4,prm18_precision);
                    OLED_Write_String(4,2,(uint8_t*)prm18);
                    OLED_Write_Num4(10,2,prm18_val);
                }
            }
            if(key1==0)
            {
                DelayMs(keydelay);
                if(key1==0)
                {
                    OLED_Clear();
                    prm18_val-=prm18_precision;

//FTM_PWM_ChangeDuty(FTM0_CH3_PC4,prm18_precision);
                    OLED_Write_String(4,2,(uint8_t*)prm18);
                    OLED_Write_Num4(10,2,prm18_val);
                }
            }
            break;

        case 3:
            if(key3==0)
            {
                DelayMs(keydelay);
                if(key3==0)
                {
                    OLED_Clear();
                    prm19_val+=prm19_precision;
```

```
//FTM_PWM_ChangeDuty(FTM2_CH0_PA10,prm19_val);
                        OLED_Write_String(4,4,(uint8_t*)prm19);
                        OLED_Write_Num4(10,4,prm19_val);
                    }
                }
                if(key1==0)
                {
                    DelayMs(keydelay);
                    if(key1==0)
                    {
                        OLED_Clear();
                        prm19_val-=prm19_precision;

//FTM_PWM_ChangeDuty(FTM2_CH0_PA10,prm19_val);
                        OLED_Write_String(4,4,(uint8_t*)prm19);
                        OLED_Write_Num4(10,4,prm19_val);
                    }
                }
                break;

            case 0:
                if(key3==0)
                {
                    DelayMs(keydelay);
                    if(key3==0)
                    {
                        OLED_Clear();
                        prm20_val+=prm20_precision;
                        OLED_Write_String(4,6,(uint8_t*)prm20);
                        OLED_Write_Num4(10,6,prm20_val);
                    }
                }
```

```
                if(key1==0)
                {
                    DelayMs(keydelay);
                    if(key1==0)
                    {
                        OLED_Clear();
                        prm20_val-=prm20_precision;
                        OLED_Write_String(4,6,(uint8_t*)prm20);
                        OLED_Write_Num4(10,6,prm20_val);
                    }
                }
                break;
        }
        if(key5==0)
        {
                DelayMs(60);
                if(key5==0)
                {
                        OLED_Clear();
                        count = 1;
                }
        }
    }   }

}

void assert_failed(uint8_t* file, uint32_t line)
{
    UART_printf("assert_failed:line:%d %s\r\n",line,file);   //断言失败检测
    while(1) ;
}
```