第七届全国大学生“飞思卡尔”杯

智能汽车邀请赛

**技 术 报 告**



学 校：华东理工大学

队伍名称：Suprise

参赛队员：张泽瀚 徐林 赵钟

带队教师：凌志浩 杜红彬

论文使用授权的说明

本人完全了解第七届全国大学生“飞思卡尔”杯智能汽车邀请赛关保留、使用技术报告和研究论文的规定，即：参赛作品著作权归参赛者本人，比赛组委会和飞思卡尔半导体公司可以在相关主页上收录并公开参赛作品的设计方案、技术报告以及参赛模型车的视频、图像资料，并将相关内容编纂收录在组委会出版论文集中。

参赛队员签名：

带队教师签名：

日 期：

# 摘 要

本智能车以飞思卡尔公司的16位单片机MC9S12XS128B为核心控制器，利用CMOS视频传感器采集路况信息，配合传感器、电机、舵机、电池等组成的驱动电路进行信息处理，以达到路径识别的目的，控制模型车高速稳定地在跑道上行驶。

硬件电路部分主要采用BTS构成的H桥电路驱动直流电动机和舵机。所选电压稳压芯片29302和LM2940，可使在7.2-8.3伏电池供电的条件下为系统的各功能模块提供稳定、可靠的工作电源，为智能车的稳定工作提供了有力的保证。测速部分采用正交编码器和分频电路完成对速度的测量和反馈。

软件系统部分主要包括以下与路径识别系统相关模块的算法：(1)CMOS视频传感器采集数据的处理模块算法；(2)速度反馈以及用PID算法实现对即时速度的调节模块；(3)舵机调节模块算法；(4)基于上述三个模块的路径识别算法和起始线识别算法；(5)相应的调试函数。

本系统利用开发工具CodeWarrior进行编程开发，用BDM进行程序下载，利用串口传输的数据进行在线调试。这些工具的使用，使得软件的设计编程和调试工作得到了保证。

通过一系列的调试，本系统基本实现了路径识别的功能，在实际的测试中，小车也比较好的完成循线行驶的任务。

**关键字**：智能车，路径识别，PID，CMOS视频传感器，PWM

目 录

[摘 要 IV](#_Toc237499181)

[第一章 引言 1](#_Toc237499182)

[第二章 总体方案概要说明 2](#_Toc237499183)

[2.1 设计思想 2](#_Toc237499184)

[2.2 传感器的选择 2](#_Toc237499185)

[第三章 机械部分设计 4](#_Toc237499186)

[3.1 CMOS传感器的安装 4](#_Toc237499187)

[3.2 CMOS传感器支架结构 5](#_Toc237499188)

[3.3测速模块的安装 5](#_Toc237499189)

[3.4舵机的安装 6](#_Toc237499190)

[3.5前轮倾角的调整 7](#_Toc237499191)

[3.6 底盘高度和重心的调整 8](#_Toc237499192)

[3.7 齿轮传动机构及后轮差速的调整 8](#_Toc237499193)

[第四章 系统硬件设计 8](#_Toc237499194)

[4.1系统硬件总体结构 9](#_Toc237499195)

[4.2电源管理模块设计 10](#_Toc237499196)

[4.3直流电机驱动电路 11](#_Toc237499197)

[4.4转速测量电路 11](#_Toc237499198)

[第五章 软件设计 13](#_Toc237499199)

[5.1系统软件总体结构 13](#_Toc237499200)

[5.2初始化模块 14](#_Toc237499201)

[5.2.1时钟初始化 14](#_Toc237499202)

[5.2.2串口初始化 14](#_Toc237499203)

[5.2.3 PWM初始化 15](#_Toc237499204)

[5.3 路径识别原理及实现 16](#_Toc237499205)

[5.3.1 路径信息采集处理方法 16](#_Toc237499206)

[5.3.2 舵机转向控制算法 19](#_Toc237499207)

[5.3.3 直流电机的控制算法 20](#_Toc237499208)

[第六章 开发工具及制作调试过程 22](#_Toc237499209)

[6.1 开发工具 22](#_Toc237499210)

[6.2 制作调试过程 22](#_Toc237499211)

[第七章 结论 23](#_Toc237499212)

[7.1模型车的主要技术参数说明 23](#_Toc237499213)

[7.2 总结 23](#_Toc237499214)

[参考文献 24](#_Toc237499215)

[附录A：程序源代码 I](#_Toc237499216)

第一章 引言

随着现代科技的飞速发展，人们对智能化的要求已越来越高，而智能化在汽车相关产业上的应用最典型的例子就是汽车电子行业，汽车的电子化程度则被看作是衡量现代汽车水平的重要标志。同时，汽车生产商推出越来越智能的汽车，来满足各种各样的市场需求。

第七届“飞思卡尔”杯全国大学生智能汽车邀请赛就是在这个背景下举行的。大赛组委会为各支参赛队伍提供智能车车模、MC9S12XS128开发板、可充电镉镍电池组和舵机。参赛队伍需要学习并应用嵌入式软件开发工具Codewarrior进行在线开发和调试。这个大赛的综合性很强，涵盖了控制、模式识别、传感、电子、电气、计算机和机械等多个学科交叉的科技创意性比赛[1]。参赛队员需要在了解上述多学科知识的基础上，利用Codewarrior软件编程控制智能车对路径信息进行采集和处理，识别当前路径状况，进而控制舵机转相应的角度，驱动电机以合适的速度在跑道上行进。

本队制作的智能车以16位单片机MC9S12XS128B为控制系统核心，利用CMOS视频传感器采集路径信息，经过系统处理，识别当前路径情况，做出判断决策，从而给出相应的PWM信号，通过BTS构成的H桥电路驱动直流电机以合适的速度行驶，同时，控制舵机转出相应的角度。

本文先从总体上介绍了智能车的设计思想和方案论证，然后分别从机械、硬件、软件等方面的设计进行论述，重点介绍了芯片的选择和路径识别的方法，接着描述了智能车的制作及调试过程，其中包含本队在制作和调试过程中遇到的问题及其解决方法。

第二章 总体方案概要说明

2.1 设计思想

智能车主要由三个方面组成：检测系统，控制决策系统，动力系统。其中检测系统主要有两个选择：红外反射式红外传感器和CMOS视频传感器。控制决策系统采用大赛组委会提供的16位单片机MC9S12XS128作为主控芯片，动力系统主要控制舵机的转角和直流电机的转速。整体的流程为，检测系统采集路径信息，经过控制决策系统分析和判断，由动力系统控制直流电机给出合适的转速，同时控制舵机给出合适的转角，从而控制智能车稳定、快速地行驶。

2.2 传感器的选择

路径采集的传感器主要有两种：红外反射式红外传感器和CMOS视频传感器。根据上届比赛的经验，CMOS视频传感器比红外反射式红外传感器具有更好的前瞻性，利用这段前瞻的距离，有利于小车提前判断下一步的行驶路线和速度。因此，本队的智能车决定以CMOS视频传感器为采集路径的基础传感器。结合CMOS视频传感器，有两种方案可以选择。

方案一：基于CMOS传感器的路径识别。电荷藕合器件图像传感器CMOS（Charge Coupled Device），是使用一种高感光度的半导体材料制成，能把光线转变成电荷，再通过模数转换器芯片转换成数字信号。CMOS由许多感光单位组成，通常以百万像素为单位。当CMOS表面受到光线照射时，每个感光单位会将电荷反映在组件上，所有的感光单位所产生的信号加在一起，就构成了一幅完整的画面。基于CMOS传感器的路径检测方法具有探测距离远的优势，能够尽可能早的感知前方的路径信息进行预判断，再现路径的真实信息。而且，普通的CMOS传感器图像分辨率都在300线以上，它的分辨率远远大于光电管阵列的方法。通过镜头，CMOS传感器可以将车模前方很远的道路图像映射到CMOS器件中，从而得到车模前方很大范围内的道路信息。基于图像的道路参数识别，不仅可以识别道路的中心位置，同时还可以得到道路的方向、曲率等信息。基于CMOS传感器，通过图像处理的方式得到路径信息，可以有效进行车模运动控制，提高车模的路径跟踪精度和运行速度。但是，使用CMOS传感器的瓶颈是传感器具有20ms的固有延时，尽管它能体现出了模型车系统的前瞻性优势，但是20ms的固有延时会影响到小车对路面信息的精确控制。方案二：基于CMOS传感器和光电传感器的相互配合控制。CMOS传感器具有探测距离远的优势，能够尽可能早的感知前方的路径信息，与光电传感器阵列配合使用，具有远近结合的优势,且具有较高的稳定性和可靠性。但是本次大赛所要求的控制核心单片机MC9S12XS128，总线时钟最高25M，无法实现高级的图像算法和控制算法,且硬件电路复杂。

综合以上考虑，系统选择方案一，单纯地使用CMOS传感器来实现小车路径信息的检测。

第三章 机械部分设计

3.1 CMOS传感器的安装

CMOS传感器的安装分由以下几个部分组成：传感器安装高度、传感器俯仰角，以及传感器的架设。

CMOS传感器的视野范围与该传感器的安装高度有关,在相同的俯仰角情况下镜头安装高度增大,传感器的视野范围也随之增大,小车的预瞄距离也会相应的增加。从扩大视场角方面考虑，安装时可以适当增加镜头的安装高度。但是，随着镜头的升高，图像会变得越模糊，特别是两边的引导黑线，可能会因此而漏掉图像信息。所以，传感器也不能安装得太高。

传感器的俯仰角是镜头中心光轴与道路水平面的夹角，俯仰角越大，则采集到的路面信息距小车越接近，俯仰角越小，则传感器采集到的信息距离小车越远。但是，当传感器采集到的路面信息距离模型车太近时，不能体现使用CMOS传感器的前瞻性；当传感器采集的路面信息距离模型车太远时，采集到的图像显得很模糊，而且很容易采集到场地外的干扰信息，同时可能由于采集到的信息太靠前，使模型车提前量太多而撞上引导线旁边的立柱。图3.1为CMOS传感器的安装示意图。

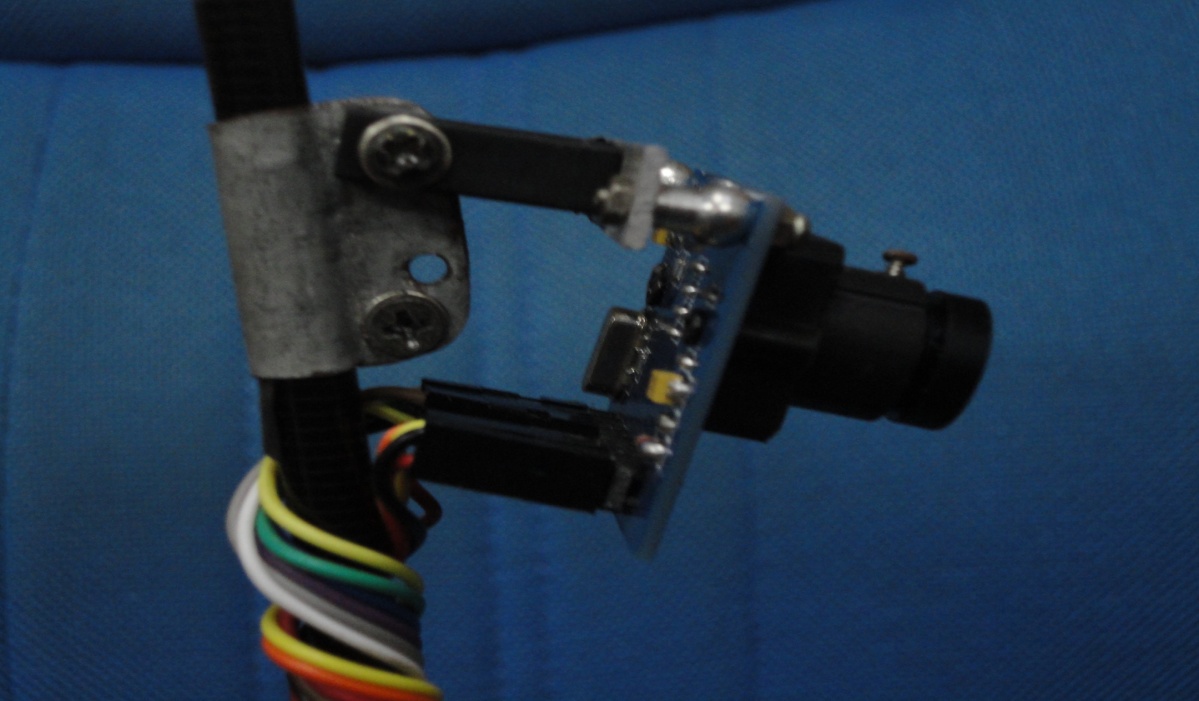


图3.1 CMOS传感器的安装

3.2 CMOS传感器支架结构

CMOS传感器架设在模型车舵机后面正中央处，架在舵机后面而没架在保险杠上，可以在一定程度上防止因模型车冲出场地后对镜头的碰撞。支架使用的是全铝合金的金属支架，使用螺丝固定在模型车的底板上。这样，CMOS传感器和车模始终保持同步的运动方式，从而，智能车可以获得同步稳定的路径信息。下图为CMOS传感器支架的安装实物图。



图3.2 CMOS传感器支架的安装

3.3测速模块的安装

测速传感器使用的集成编码器对模型车进行测速，固定在模型车的后转轴附近，小车的速度以脉冲个数的形式通过编码器发送给主板，单片机根据速度反馈来调整电机速度的控制量，测速模块安装见图3.3。



图3.3 测速模块安装

3.4舵机的安装

舵机通过一对连杆分别连接两前轮内侧固定点，通过两连杆在舵机转动力带动下的横向运动来控制两车轮的转向，我们采用最近流行的等臂立式舵机安装方法，提高了舵机响应速度，使舵机更加灵活。具体安装示意图如下：

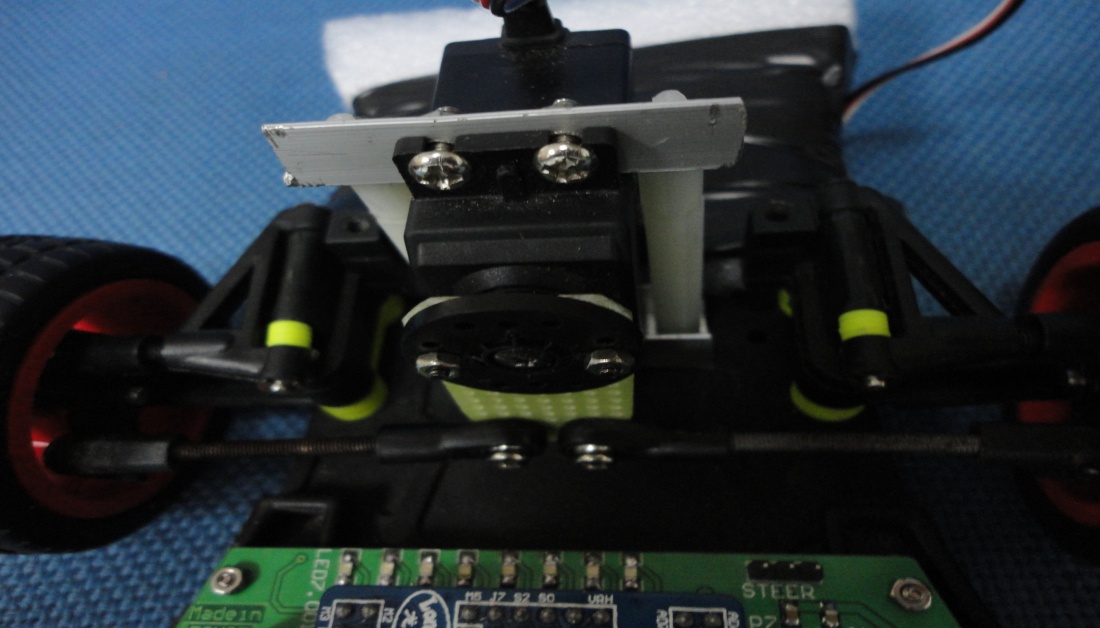


图3.4 舵机安装示意图

3.5前轮倾角的调整

我们在调试中发现：由于前轮轴和车轮之间的间隙较大，对车高速转向时的重心影响较大，会引起高速转向时车的转向不足。而且这里又是规则中严禁改动的部分，所以为了尽可能降低转向舵机负载，我们对前轮的安装角度，即前轮定位进行了调整。

前轮定位的作用是保障汽车直线行驶的稳定性、转向轻便和减少轮胎的磨损。前轮是转向轮，它的安装位置由主销内倾、主销后倾、前轮外倾和前轮前束等4个因素决定，反映了转向轮、主销和前轴等三者在车架上的位置关系。

主销内倾是指主销装在前轴略向内倾斜的角度，它的作用是使前轮自动回正。内倾角度越大时前轮自动回正的作用就越强烈，但转向时也越费力，轮胎磨损增大；反之，内倾角度越小时前轮自动回正的作用就越弱。我们没有对此参数进行调整。

主销后倾（Caster）是指主销装在前轴，上端略向后倾斜的角度。它使车辆转弯时产生的离心力所形成的力矩方向与车轮偏转方向相反，迫使车轮偏转后自动恢复到原来的中间位置上。由此，主销后倾角越大，车速越高，前轮稳定性也愈好。主销内倾和主销后倾都有使汽车转向自动回正，保持直线行驶的功能。不同之处是主销内倾的回正与车速无关，主销后倾的回正与车速有关，因此高速时后倾的回正作用大，低速时内倾的回正作用大。由于智能车比赛赛道逐渐向复杂型，多弯的赛道发展，所以我们更多的要求提升赛车弯道性能，故此处我们采用前1后3的安装方式。安装如下图：

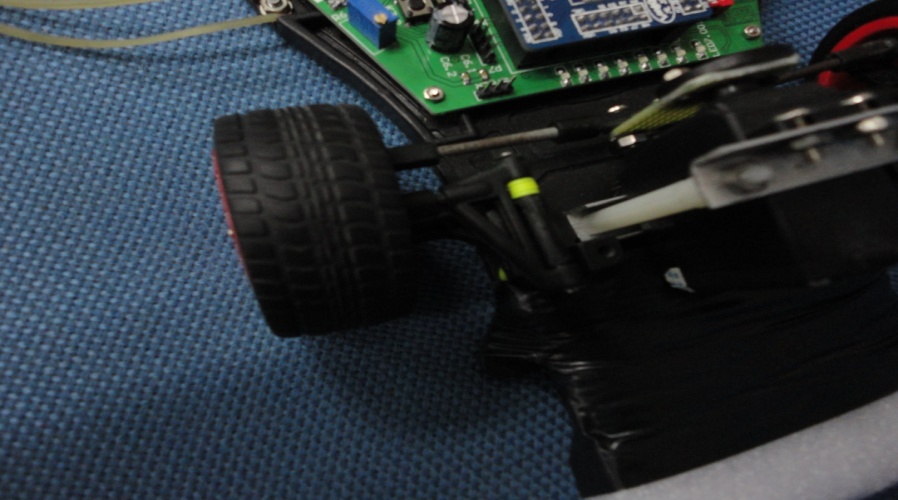


图3.5主销后倾安装示意图

前轮外倾角（Camber）对赛车的转弯性能有直接影响，它的作用是提高前轮的转向安全性和转向操纵的轻便性。前轮外倾角俗称“外八字”，如果车轮垂直地面一旦满载就易产生变形，可能引起车轮上部向内倾侧，导致车轮联接件损坏。此处可根据实际情况，调整5°左右的外倾角。

所谓前束（Toe-out）是指两轮之间的后距离数值与前距离数值之差，也指前轮中心线与纵向中心线的夹角。前轮前束的作用是保证汽车的行驶性能，减少轮胎的磨损。前轮在滚动时，其惯性力会自然将轮胎向内偏斜，如果前束适当，轮胎滚动时的偏斜方向就会抵消，轮胎内外侧磨损的现象会减少。此处一般调整为0°前束。

在实际调试中，我们发现适当增大内倾角的确可以增大转弯时车轮和地面的接触面积，从而增大车轮和地面的摩擦程度，使车转向更灵活，减小因摩擦不够而引起的转向不足的情况。

3.6 底盘高度和重心的调整

底盘适当降低，在可以过坡道的情况下，尽量降低底盘，从整体上降低车的重心，使车在转弯时可以更加稳定、快速。

此外智能车的电池占据了很大一部分重量，我们将电池后移将近2cm，将重心靠后增大了后轮抓地力。

3.7 齿轮传动机构及后轮差速的调整

车模后轮采用玩具电机驱动，由邀请赛主办方提供。齿轮传动机构对车模的驱动能力有很大的影响。齿轮传动部分安装不恰当，会增大电机驱动后轮的负载；齿轮配合间隙过松则容易打坏齿轮，过紧则会增加传动阻力。所以我们在电机安装过程中尽量使得传动齿轮轴保持平行，传动部分轻松、流畅，不存在卡壳或迟滞现象。

差速机构的作用是在车模转弯的时候，降低后轮与地面之间的滑动；并且还可以保证在轮胎抱死的情况下不会损害到电机。差速器的调整中要注意滚珠轮盘间的间隙，过松过紧都会使差速器性能降低，转弯时阻力小的车轮会打滑，从而影响车模的过弯性能。

第四章 系统硬件设计

硬件系统是整个智能车能够良好稳定运行的基础，选择不同的硬件，对系统的功能实现的影响是不同的。尽管很多芯片都能实现同样的功能，但是，对于不同的系统而言，芯片不同，实现的效果就不一样。有的芯片转换效率不够高，有的芯片散热效果不够好，或者某些芯片的输出不够稳定等等。这些问题都需要结合自己所设计的系统，选择最佳的电器元件。

下面将本车的硬件系统分为不同的模块分别介绍如下。

4.1系统硬件总体结构

首先，对硬件系统结构进行全面的了解，图4.1列出硬件系统的总体结构图。

充电电池

（7.2V 2A/h）

29302

输出6V

LM2940

输出5V

LM2940

输出5V

单片机

电机驱动芯片BTS桥电路

舵机

CMOS摄像头

（7.2V） 直流电机

编码器

图4.1 系统硬件总体框图

由上图可知，系统由一片MC9S12XS128作为主控制器，由电池经稳压后提供电，CMOS传感器和编码器由LM2940提供电源，电池直接驱动BTS7960，29302输出+6V电压供给舵机，CMOS传感器采集到数据后，直接传送给MCU。上面只是从整体上介绍系统硬件总体结构，为了更详尽的了解系统各部分的功能和工作原理，下面将分别对各个模块进行分析。

4.2电源管理模块设计

电源模块为系统其他各个模块提供所需的电源。设计中，除了需要考虑电压范围和电流容量等基本参数之外，还要在电源转换效率、降低噪声、防止干扰和电路简单等方案进行优化。可靠的电源方案是整个硬件电路稳定可靠运行的基础。

全部硬件电路的电源由7.2V，2A/h的可充电镍镉电池提供。由于电路中的不同电路模块所需要的工作电流容量各不相同，因此电源模块应该包含多个稳压电路，将充电电池电压转换成各个模块所需要的电压。

5V电压，给单片机、信号调理电路以及部分接口电路提供电源，电压要求稳定、噪声小，电流容量大于500mA。6V电压，为舵机提供工作电压。实际工作时，舵机所需要的工作电流一般在几十毫安左右，电压无需十分稳定。

7.2V电压。这部分直接取自电池两端电压，主要为后轮电机驱动模块提供电源。

5.0V电压。如果采用CMOS/CMOS图像传感器来进行道路检测，只需要5V工作电源。编码器也采用这个电源作为工作电压，为了保证电压稳压效果好另用一稳压芯片保证芯片和摄像头电压稳定，见图4.2。

图4.2 LM2940（5V）工作原理图

4.3直流电机驱动电路

用两片BTS7960构成的H桥芯片作为电机的驱动电路，通过4和8输出的PWM信号来控制电机。如图4.4所示。

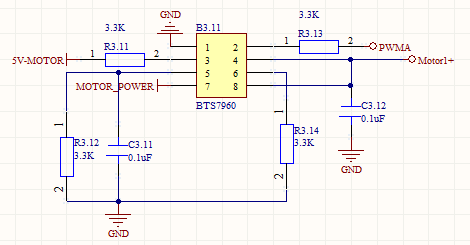


图4.4 H桥工作原理图

为了更快的减速，我们考虑了利用电机的倒转进行减速的过程，也就是：直流电机的反转刹车。当智能车需要刹车的时候，可以适当的进行倒转，来达到快速减速的过程。

4.4转速测量电路

为了使得模型车能够平稳地沿着赛道运行，除了控制前轮转向外，还需要控制小车的车速，使模型车在急转弯时速度不至于过快而冲出赛道，同时也使小车在直线段时以较快的速度行驶。所以要时刻把握当前小车的速度，并根据小车所处的位置来实时调整小车的速度。通过速度检测，可以消除或降低电池电压、电机传动摩擦力、道路摩擦力等的影响，使得小车在赛道上运行得更精确。

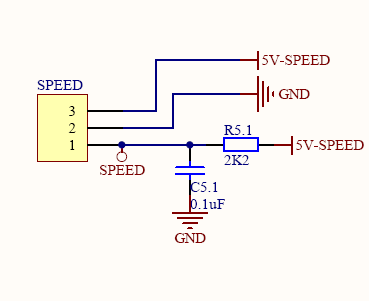


图4.5 测速

第五章 软件设计

为了体现程序的系统性和连贯性，本设计把程序分成各个模块进行分别处理，而各模块也有相应的数据接口，方便其他模块调用，这样，系统条理显得清晰。这些模块的设计与实现将在下面进行详细阐述。

5.1系统软件总体结构

该系统的软件结构主要分为：系统初始化模块、对CMOS采集数据的处理模块、速度控制和反馈处理模块、路径识别模块（对电机和舵机的控制）等，再有为了便于系统的调试，本系统设计了相关的调试函数，主要包括串口和定时模块。图5.1是本系统的总体结构图。



图5.1 系统软件总体结构框图

由上图可知，本程序运行的开始，须进行对系统的初始化。通过初始化的设置，系统才能按照预先的想法运行；接着，CMOS传感器采集数据，经过转换和相应处理，得出当前路径信息；同时，电机测速模块测得模型车当前的运行速度，反馈给系统；最后，路径识别系统综合利用当前路径信息和当前速度值作出相应的处理，控制电机和舵机以合适的方式运行。

5.2初始化模块

下面只介绍部分重要的初始化函数。

5.2.1时钟初始化

单片机的晶振是16MHz，MC9S12XS128在时钟初始化后可以通过锁相环将系统时钟系统时钟提高到80MHz，这里就用到了一些寄存器中的某些位，倍频公式如下：

公式1



其中，OSCCLK\_VALUE——系统的外部晶振；

REFDV\_VALUE——系统时钟分频系数，在初始化中，它的值为3；

SYNR\_VALUE——倍频系数，它的值为3。

而SYNR=SYNR\_VALUE，REFDV=REFDV\_VALUE，SYNR、REFDV就是寄存器中的对应位。

5.2.2串口初始化

可以看到，不论8位、16位还是32位单片机的最小系统都是通过异步串行口与人沟通的。使用串行通信接口SCI(Serial Communication Interface)通信是计算机与人对话最传统、最基本的方法，异步通信接口也成为通用异步接受器/发送器UART(Universal Asynchronous Receiver/Tansmitter)。

为了便于调试，本系统利用串口通信接口SCI实现数据的传输。

串口部分程序主要是初始化各寄存器，其中包括设置波特率、数据格式、接受发送功能使能、设置接受模式等。

串口初始化函数：SCI\_Init(uchar BUS\_CLK,unsigned long BR)，它有两个输入变量，其中BUS\_CLK是系统的时钟频率,BR为波特率。通过此函数可直接设置串口的时钟频率、波特率[5]。

5.2.3 PWM初始化

PWM(Pulse Width Modulation)即脉宽调制，脉宽调制波是一种可用程序来控制波形占空比、周期、相位的波形。它在电动机驱动、D/A变换等场合有着广泛的应用[6]。

PWM模块特点：

**☑** 8个带周期占空比可程控的PWM独立通道

**☑** 4个可程控选择的时钟源

**☑** 每个PWM通道有专用的计数器

**☑** PWM每个通道脉冲极性可以选择

**☑** 每个PWM通道可使能/禁止

**☑** 周期和占空比双缓冲

**☑** 每个通道有中心对齐和边缘对齐方式

**☑** 分辨率: 8位(8通道)，16位(4通道)

**☑** 带中断功能的紧急切断

PWM在本系统中主要用于驱动两个伺服电机，一个是电机，另一个是舵机，而控制这两个电机的PWM信号是经过路径识别和系统决策后给出的，这样就能实现小车的循线跑。

图5.2是PWM初始化程序流程图。



图5.2 PWM初始化程序流程图

5.3 路径识别原理及实现

5.3.1 路径信息采集处理方法

智能车通过CMOS视频传感器的行中断和场中断，把采集到的路径信息传送给MC9S12XS128。这样，这一场的数据就变成了一个292\*365的二维矩阵，其中的每个数字代表着一场图像中对应位置点的象素灰度值。但是，单片机处理这么多的数据会十分消耗资源，而且单片机的内存也不可能给这个存放图像数据的二维矩阵分配太多的存储空间。所以，为了节省单片机的资源和减少处理时间，就必须对每场的图像做一些处理，本队采取的方法是,选择实际需要的行分辨率和列分辨率；把一场图像按照的10cm分辨率分成40行；根据最远行看到黑线的点数不小于2点的标准，一行取100个点。

这样，该场图像的数据就变成了一个100\*40二维矩阵。图5.3就是一条直线的原始图像。

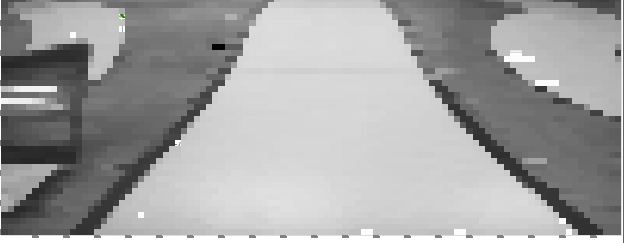


图5.3 直线赛道原始图像

接着，就要对CMOS视频传感器采集的视频数据进行路径判断处理，这时，需要确定视频阈值的大小。如果象素点灰度值小于该阈值，则确定该点为黑点，否则，确定其为白点。这样，就可以在一场数据中分辨出象素点是黑色还是白色。处理后的结果如图5.4所示。为了调试方便，把灰度值小于阈值的点对应的灰度值都置为0。

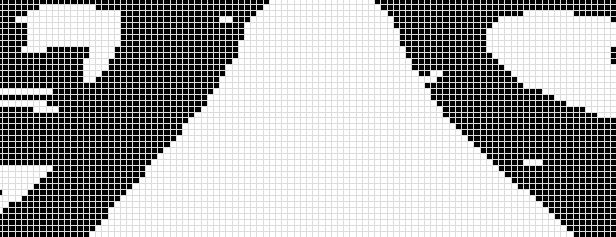


图5.4 处理后的直线赛道图像数据图

由图5.4知，黑点的灰度随着所在行的距离变远而变高，同时黑白点的灰度差值的绝对值也随着所在行的距离变远而变小。这说明图像在近处较之远处更黑白分明，因此本队采用最远处行的阈值作为整场图像的阈值。这样近处黑点灰度本来就小于远处，也就必然在阈值以下；同时由于近处黑白分明，白点的灰度很高会阈值。因此采用最远处行的阈值作为整场图像的阈值理论上是可行的，经实践检验也确实有效，只需根据经验确定阈值即可。

考虑到不同的光强会对阈值有影响，换了一个比赛场地之后原先的阈值有可能失效。本队采用自适应阈值算法作为拨码开关的一个选项，用作阈值的备用方案。本算法旨在适应整个赛场光照环境的变化，对于赛道局部的亮斑无能为力，而正规比赛场地基本上没有这种亮斑，使得本算法有一定的实用价值。

下面介绍一下本队的自适应阈值算法。先放好智能车，使导航线大致处于CMOS视频传感器视野的中心线上，取最远行中心的若干点进行分析（中心的点在导航线附近，是可靠的灰度信息），由双峰法得到阈值。

在确定某点是黑（白）点之后，就可以确定每行黑线中心位置和黑白变化次数。下面介绍一下具体的实现方法。通过实践测试观察，摄像头采集的数据中，近处的数据一直是比较准确的，因此，处理的顺序是从近向远。

每行有100个象素点，所以把这100个点分别标号，记为0～99。按照从右向左的顺序，每一点与阈值作比较，记下第一个点是黑（白）点。接下来只在有黑白变化处作相应处理。

在白到黑处，此黑点标号减上次黑到白的边界的差即白线长度，白线长度小于事先标定的干扰长度则认为前段是白点干扰，大于干扰长度则认为前段是白线。若是白线，则黑白变化次数自增，记下此边界。若是白点干扰，则各参数复原为被干扰前的状态。在黑到白处，作类似处理。一行处理完后，如果黑白变化次数为2，且白到黑的边界小于黑到白的边界则认为本行是可靠的赛道信息。

设白到黑边界的序号为A，黑到白边界的序号为B，将序号（B－A）/2，就可以得到该行黑点的中心位置坐标C，（B－A）即为该行的黑点个数，记为D。以上就是本场第一个有效行的数据，记本行为istart，为了提高系统处理的速度，降低系统资源的消耗，以后各行数据的处理方式与第一行有所区别。在处理后面的行时，可以根据上一行得出的导航线的右边界来确定下一行扫描的范围。例如，如果上一行的导航线的右边界A进坐标转换成是实际坐标Av,把 Av-10cm和Av-10cm的坐标分别转化为对应的点的标号，并分别记为jstart，jend。那么下一行就从jstart扫描到jend。这样，就可以大大减少待处理的数据，从而降低了数据处理的时间，加快了系统处理数据的速度。每次发现白到黑时将本点标号转化为实际坐标后减去前一行的右边界取绝对值，在一行中取这个绝对值的最小值。即离上一行导航线最近的黑线为本行的导航线，这是考虑的赛道的导航线是连续的，这样就使抗干扰能力增强了，即使扫描到起始线和十字交叉干扰，也能正确算出导航线中心位置。

经过上述图像处理后，可以获得两个主要的参数：每行黑点的中心位置坐标和每行的黑白变化次数。依据这两个参数，可以判断当前的路径信息。当发现一行的变化次数为0的时候，则说明本行没有看到导航线，本行的上一行为该场图像最后一个有效行，记为iend。

5.3.2 舵机转向控制算法

经过上面的图像处理方法，得到本场第一个有效行为istart，最后一个有效行iend。为了使智能车的路径更优化，需要把赛道分类。针对不同赛道选用不同的走线策略。由(istart,cent[istart])和(iend,cent[iend])两点确定一直线y=a\*i+b记为L。当有效行大于3行时，从istart+1开始到iend-1结束进行逐行处理。每行计算cent[i]-a\*i+b。如果扫描中发现有的行导航线在L左侧，有的在L右侧，因为只有S弯才会看到此现象，所以此时赛道为S弯，再根据bulb（bulb为一场中最大的cent[i]-a\*i+b）的大小可确定是大S弯还是小S弯。如不是S弯，再根据iend的大小来判断。iend大说明看到的图像长，那转弯处必定较远，记为远弯。相应的iend小则为近弯。如果有效行小于3行，再看istart是否为第0行。是则说明比较信息比较可靠，且弯很急记为急弯，否则信息不可靠记为不可靠信号。因为直道上不管以哪一行控制差别都不大，所以直到也可归为远弯。

得到了路径识别的结果，再确定不同路径的控制行。先按在直线上以每行作为控制行分别得到的转角相等为标准，确定每行的转角系数。这样同样大小的cent[i]所对应的转角随着i增大而减小。如果路径为小S弯，就以iend为控制行，以达到直走的效果。如果是大S弯，根据S玩的大小，分别以iend和ibulb控制行算得的转角按一定权值求平均值作为最终转角。如果是远弯，如果以iend行的黑线中值来控制舵机会使智能车过弯太提前，容易从弯内侧冲出赛道，为此要沿线走较合理，而ibulb正好是相当于直线与拐弯交界处，所以以ibulb为控制行。如果是近弯，以iend控制，会走内线，既不易冲出赛道又缩短路径。

如果是急弯，以iend控制转的最大。如果是不可靠信号，就以近处某一固定行控制，以达到循线走的效果。

为了更好的解决转向的问题，我们采用分段转角系数控制舵机。根据不同的路况信息来设定不同的转角系数。因为在直道或小S上转角系数太大，往往会导致舵机的抖动使得小车在直线上左右摇摆，所以如果小车行驶在直线上那么舵机的转角系数应该比较小；如果小车舵机行驶湾道上，那么转角系数应该同时变大否则很可能不能及时地过弯道。

5.3.3 直流电机的控制算法

根据路径识别的情况，如果当前路径为直道，则需要加速；若是弯道，则需要降速，而且根据不同的弯道速度也是有所区别，这个区别体现在速度设定上。我们根据iend、舵机转角、bulb等多个变量的大小来确定速度设定的策略。

当iend等于最远行时，说明弯道离得很远，速度可以设为很高的值。这时需要根据bulb来确定远处是弯道还是直道，bulb越大速度应该越小，bulb为0时速度为最高档。当iend不等于最远行时，速度需要降一个档，再根据iend越大速度越高，舵机转角越大，速度越低的原则，对速度做进一步微调。当iend小于4时，说明有冲出赛道的危险，速度再降一档，再根据iend越大速度越高，舵机转角越大，速度越低的原则，对速度做进一步微调。由于速度太低时，电机动力不足难以推动智能车前进。因此速度设定应该有个最低限幅值。当运算得到的速度设定小于该限幅值时，速度直接赋值为最低值。速度设定中用到的各参数都可用拨码开关来改变，这在实际比赛中有较大的实际意义。

以上只是速度设定的算法，实际的车速还需要闭环控制，用PID算法来实现稳定、准确、快速地跟随速度设定。下面介绍速度的控制算法。

系统利用测速模块反馈的当前速度值，和设定速度值来选择不同的P参数来进行调节，从而控制直流电机对当前路径进行快速反应,利用I的作用使得消除静差，加入D使得速度更加稳定。

本系统采用的是数字PID 控制，通过每一控制周期读入脉冲数来间接测得小车当前转速vi\_FeedBack，将vi\_ FeedBack与模糊推理得到的小车期望速度vi\_Ref比较，既而算得速度偏差error，再通过调用PID函数来获得速度的控制。

考虑到CMOS方案的控制周期较长，假设按2.5m/s的平均速度计算，则一个控制周期小车大概可以跑过5cm，如果按这种周期用上述PID调节速度，则会导致加速减速均过长的后果，严重的影响小车的快速性和稳定性。为了解决这个问题，可以在PID调速控制中适当加入Bang-Bang控制：根据error的大小，如果正大，则正转给全额占空比；如果负大，则自由停车或给一个反转的PWM，否则就采用PID来计算所需的占空比。

第六章 开发工具及制作调试过程

6.1 开发工具

开发工具使用的是大赛组委会提高的Codewarrior3.1、Codewarrior4.1开发环境。它能够为单片机MC9S12XS128提供与之配套的应用程序开发模块。在目标程序的下载方面，通过BDM与单片机之间的连接下载程序。BDM的使用参考了清华大学的《BDM\_for\_S12\_TBDML\_用户手册V1.3》。在调试方面，使用了MC9S12XS128的串口，利用串口线将MC9S12XS128和PC的串口相连，使用串口调试工具或PC的超级终端进行程序的调试。使用Protel DXP作为设计硬件电路的软件工具。在调试舵机转角的时候，使用Excel软件进行转角的调整和分析。

6.2 制作调试过程

在整体了解制作智能车所需要的各个模块之后，队员们进行分工，分别完成了各自所负责的模块，这其中主要包括视频采集及处理模块、测速及速度控制模块、舵机控制模块、电源的设计模块和智能车整体组装和摄像头的安装等的机械设计模块。

在完整组装智能车后，开始进入到调试阶段。本队主要利用串口和PCB板上的8个发光二极管进行调试。

在调试的过程中，我们也遇到了一些问题。例如在调试串口程序时，总会出现串口数据不能输出，或是程序运行后自己跳出的情况。经过调试，发现这些问题的出现与IRQ中断有关，因为，IRQ中断的优先级高，如果不关闭IRQ中断，系统总是要进入IRQ中断函数，而在调试串口程序时，系统运行总是被IRQ中断所打断，这时就会出现上面的问题。所以，在进行调试单个程序模块时，应适当考虑关闭IRQ中断（优先级高的中断）使能。

第七章 结论

7.1模型车的主要技术参数说明

|  |  |
| --- | --- |
| 项目 | 参数 |
| 检测方式（参赛组） | 摄像头组 |
| 车模几何尺寸（长、宽、高）（毫米） | 345、162.5、310 |
| 车模平均电流（匀速）（毫安） | 约14000 |
| 电路总电容量（微法） | 1580 |
| 传感器个数 | 1 |
| 新增加私服舵机个数 | 0 |
| 赛道信息检测频率（赫兹） | 50 |
| 车模重量 | 约1kg |

7.2 总结

本智能车以飞思卡尔公司的16位单片机MC9S12XS128B为核心控制器，利用CMOS视频传感器采集路况信息，配合传感器、电机、舵机、电池等组成的驱动电路进行信息处理，以达到路径识别的目的，控制模型车高速稳定地在跑道上行驶。

通过对智能车的仿真和实践测试，整个系统的方案是完全可行的，系统的控制策略和软硬件基本合理。运用灰度阈值自适应调整，获得准确的路径图像信息。

但本车的有些方面也是有待改进的，例如，在实际的测试中，时常会出现复位的状况，这是因为当某个芯片，尤其是主控芯片的供电不稳时，造成电路中的电流不稳定，如果电流过低，就会造成系统复位的情况。因此，在电源的设计上，要使每个芯片的供电独立分开，尽量减少它们之间的干扰。由此可见，电源的设计在整个系统的设计中占有很重要的地位，所以，在以后的研究中，要十分重视对电源的设计，从而达到提高整个系统稳定性的目的。

# 

# 参考文献

1．卓青，黄开胜，邵贝贝．学做智能车——挑战“飞思卡尔”杯[M]，北京：北京航空航天出版社，2007，1-2．

2．齐永利，鲁云峰，刘鸣．LM2907频率/电压转换器原理及应用[J]，国外电子元器件，2005，5．

3．刘淑英，蔡胜乐，王文辉．电路与电子学（第二版）[M]，北京：电子工业出版社，2003．

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

5．俞斯乐，侯正信，冯启明等．电视原理（第5版）[M]，北京：国防工业出版社，2004．

6．谭浩强．C语言程序设计（第二版）[M]，北京：清华大学出版社，2003．

7．MOTOROLA．MC68HC912DG128 Advance Information[M]，2000．

8．蔡萍，赵辉．现代检测技术与系统[M]，北京：高等教育出版社，2005．

9．张国范，顾树生，王明顺．计算机控制系统．北京：冶金工业出版社，2004．

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

11．孙增圻，张再兴．智能控制的理论与技术[J]，控制与决策，1996，2(01)．

12．王明顺，沈谋全．传感器与智能车路径识别[J]，电子产品世界，2007，4(04)．

13．王永军，李景华．数字逻辑与数字系统（第二版）[M]，北京：电子工业出版社，2003．

14．孙浩，程磊，黄卫华．基于HCS12的小车智能控制系统设计[J]，单片机与嵌入式系统应用，2007，2(03)．

15．孙涵，任明武，唐振民等．基于机器视觉的智能车辆导航综述[J]，公路交通科技，2005．

16．葛亚明，刘涛，王宗义．视频同步分离芯片LM1881及其应用[M]，哈尔滨工程大学,2004.

17．李兰友，万振凯，李静东．微型计算机原理与接口技术[M]，北京：北方交通大学出版社，2003．

18．陈杰，黄鸿.传感器与检测技术[M]，北京：高等教育出版社，2002.

19．Freescale Semiconductor Technical Data Document order number: MC33886 Rev 7.0, 07/2005

20．TPS7350Q LOW-DROPOUT VOLTAGE REGULATORS WITH INTEGRATE-D DELAYED RESET FUNCTION SLVS124F – JUNE 1995 – REVISED JANUARY 1999

21．姜志玲．简易智能电动车的设计[J]，华东交通大学学报，2005.

22．MAXIM +12V，120mA Flash Memory Programming Supply Rev2 1993.1

23．吴建平，殷战国，曹思榕等．红外反射式传感器在自主式寻迹机器人导航中的应用[J]，中国测试技术，2004，30(6)．

# 

# 附录A：程序源代码

#include <hidef.h> /\* common defines and macros \*/

#include <MC9S12XS128.h> /\* derivative information \*/

#pragma LINK\_INFO DERIVATIVE "mc9s12xs128"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*变量声明\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//采集头像信息

#define VIDEO\_WIDTH 100 //图像宽度

#define VIDEO\_HEIGHT 40 //图像高度

#define VIDEO\_INIT\_DATA 251 //图像各像素初始值

#define VIDEO\_INIT\_DATA\_8S 150 //图像各像素初始值

#define VIDEO\_INIT\_DATA\_16S 30000 //图像各像素初始值

#define VIDEO\_COLOR\_MIDDLE 120 //二值化阈值

//舵机参数

#define STEER\_PWM\_MIDDLE 1790 //舵机PWM中值

#define STEER\_RANGE 310

#define STEER\_PWM\_MIN STEER\_PWM\_MIDDLE-STEER\_RANGE //舵机PWM最小值 向右最大

#define STEER\_PWM\_MAX STEER\_PWM\_MIDDLE+STEER\_RANGE //舵机PWM最大值 向左最大

#define STEER\_PWM PWMDTY67//舵机PWM

//电机

#define MOTOR\_PWM\_MIN 300 //电机PWM最小值

#define MOTOR\_PWM\_MAX 700 //电机PWM最大值

#define MOTOR\_PWM\_MIDDLE 500 //电机PWM中值 此时停车

#define MOTOR\_FORWARD(pwm) PWMDTY0=pwm;PWMDTY1=0 //右轮前进

#define MOTOR\_BACKWARD(pwm) PWMDTY1=pwm;PWMDTY0=0 //右轮后退

//图像处理

#define BLACK\_BLOCK\_MAX\_NUM 5 //图像处理时候每行最都搜寻的黑块个数

#define BLACK\_BLOCK\_MIN\_LEN 2 //搜寻黑块最小长度，该值为基行黑块长度，向上会逐渐递减

#define BLACK\_BLOCK\_MAX\_LEN 15 //搜寻黑块最大长度，该值为基行黑块长度，向上会逐渐递减

#define ADJACENCY\_FRAME\_BASELINES\_LIMIT 30 //相邻两场基行平均值相差限制

//摄像头状态

#define VIDEO\_EVEN\_STATUS PORTE\_PE2

#define VIDEO\_WAIT 0x01

#define VIDEO\_ACQUISITION 0x02

//智能车状态

#define CAR\_RUN 0x01

#define CAR\_NORMAL\_STOP 0x02

#define CAR\_ERROR\_STOP 0x03

//道路状况

#define ROAD\_ZhiDao 0x0001 //直道

#define ROAD\_ZhiDaoRuWan 0x0002 //直道入弯

#define SetRoadStatus(s) RoadStatus|=s

#define UnsetRoadStatus(s) RoadStatus&=(~s)

#define HaveRoadStatus(s) (RoadStatus&s)

#define PrevHaveRoadStatus(s) (RoadStatusPrev&s)

#define ResetRoadStatus RoadStatus=0

#define HaltAndLedAlert for(;;){MOTOR\_LEFT\_FORWARD(0);MOTOR\_RIGHT\_FORWARD(0);STEER\_PWM = STEER\_PWM\_MIDDLE;Led1On;Led2On;Led3On;Led4On;Delay1ms(100);Led1Off;Led2Off;Led3Off;Led4Off;Delay1ms(100);}

#define AllLedOn Led1On;Led2On;Led3On;Led4On;Led5On;Led6On;Led7On;Led8On;Led9On;Led10On;Led11On;Led12On

#define AllLedOff Led1Off;Led2Off;Led3Off;Led4Off;Led5Off;Led6Off;Led7Off;Led8Off;Led9Off;Led10Off;Led11Off;Led12Off

//无线发送模块

//检查无线模块是否还有剩余数据未发送并且发送

#define CHECK\_LEFT\_AND\_SEND if(nRF905Index!=0){TxPacket();nRF905Index=0;Delay1ms(9);}

//立即发送

#define SEND\_NOW TxPacket();nRF905Index=0

//写入发送流

#define WRITE\_SEND\_BYTE(c) nRF905TxBuf[nRF905Index++]=c;if(nRF905Index==32){TxPacket();nRF905Index=0;Delay1ms(9);}

//写入命令字

#define WRITE\_COMMAND(c) WRITE\_SEND\_BYTE(0);WRITE\_SEND\_BYTE(255);WRITE\_SEND\_BYTE(c);WRITE\_SEND\_BYTE(0)

//分析参数

#define CM\_PER\_10000\_PULSE 453//712 //每10000个脉冲对应距离（单位cm）

#define BACKUP\_PULSE(int16s) int16s=PACNT

#define RESTORE\_PULSE(int16s) PACNT=int16s

//其他

#define ConfirmKey (Key3||Key1)

/\*函数声明结束\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*初始化程序\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Clock Init\*/

void TIMInit(void)

{

TIOS\_IOS7 = 0; //The corresponding channel acts as an input capture.

OC7M\_OC7M7 = 0; //

TCTL1 = 0; //OMx = 0 and OLx = 0

PACTL\_PAEN = 1; //Pulse Accumulator system enabled.

PACTL\_PAMOD = 0; //Event counter mode.

PACTL\_PEDGE = 0; //Falling edges on IOC7 pin cause the count to be incremented.

TSCR1\_TFFCA = 1; //Any access to the PACNT registers (0x0022, 0x0023) clears the PAOVF and PAIF flags in the PAFLG register (0x0021).

}

void Timerch0Init(void) //通道0产生10ms中断

{

TSCR2\_PR =5; //prescale factor is 32, bus clock/32=40Mhz/32=1250000

TIOS\_IOS0 =0x01; //0 channel output compare

TC0 =0x30D4;//channel 0 output compare holding register 0x30D4\*(1/1250000)=10ms

TCTL2 =0x02; //output low level

TSCR1\_TFFCA=1; // if set, TFLG1 can be cleared auto, manual clear is useless

TIE\_C0I=1; //Timer0 Interrupt enable

TSCR1\_TEN = 1; //Timer enable

}

INT16U GetPulse(void)

{

INT16U l\_Pulse;

l\_Pulse = PACNT;

//PACNT=10; //读出脉冲计数值

//PACNT = 0;

return l\_Pulse;

}

/\*TIM\_C END\*/

/\*PWM7 Init\*/

void PWM67Init(void)

{

PWMCTL\_CON67 = 1; //use 16 bit pwm channel

PWME\_PWME7 = 0; //Disable

PWMPRCLK\_PCKB = 3; //Clock A = Bus / PWMPRCLK\_PCKA;

PWMSCLB = 2; //Clock SA = Clock A /(2 \* PWMSCLA)

PWMCLK\_PCLK7 = 1; //选择 PWM3口的时钟源为Clock SA 若置为0，则时钟源为Clock A

PWMPOL\_PPOL7 = 1; //设置PWM波的一个周期开始时是高电平

PWMCNT67 = 0; //PWMDTY的计数器，用于改变PWM波电平。

PWMDTY67 = 0; //设置占空比，Duty Cycle = [PWMDTY / PWMPER] \* 100%

PWMPER67 = 18750; //设置PWM频率，PWM Frequency = Clock SA / PWMPER 初始值为 200Hz

PWME\_PWME7 = 1; //Enable

}

void PWM01Init(void)

{

PWMPRCLK\_PCKA = 3; //不使能总线分频，Clock A为总线频率 注意改变此初会影响舵机的分频特性

PWMPRCLK\_PCKB = 3; //不使能总线分频，Clock B为总线频率

//01 45 Clock A

//23 67 Clock B

PWMPRCLK\_PCKA = 3; //Clock A = Bus / PWMPRCLK\_PCKA;

PWMSCLA = 2; //Clock SA = Clock A /(2 \* PWMSCLA)

PWMPRCLK\_PCKB = 3; //Clock B = Bus / PWMPRCLK\_PCKB;

PWMSCLB = 2; //Clock SB = Clock B /(2 \* PWMSCLB)

PWMCTL\_CON01 = 0;

PWME\_PWME0 = 0;

PWMCLK\_PCLK0 = 1;

PWMPOL\_PPOL0 = 1;

PWMCNT0 = 0;

PWMDTY0 = 0;

PWMPER0 = 255;

PWME\_PWME0 = 1;

PWME\_PWME1 = 0;

PWMCLK\_PCLK1 = 1;

PWMPOL\_PPOL1 = 1;

PWMCNT1 = 0;

PWMDTY1 = 0;

PWMPER1 = 255;

PWME\_PWME1 = 1;

}

/\*IRQ Init\*/

#include "IRQ.h"

void IRQInit(void)

{

INT\_CFADDR = 0xF0; //Interrupt Request Configuration Address Register

INT\_CFDATA1 = 0x07; //IRQ have the highest priority.

IRQCR\_IRQEN = 0; //IRQ off

IRQCR\_IRQE = 1; // Failing sensitive only

}

/\*Keyboard Init\*/

#include "Keyboard.h"

//初始化按键键盘端口

void InitKeyboard(void)

{

Led1Init;

Led2Init;

Led3Init;

Led4Init;

Led5Init;

Led6Init;

Led7Init;

Led8Init;

Led9Init;

Led10Init;

Led11Init;

Led12Init;

Key1Init;

Key2Init;

Key3Init;

Key4Init;

Sw1Init;

Sw2Init;

Sw3Init;

Sw4Init;

Sw5Init;

}

/\* PLL Init\*/

void PLLInit(void)

{

CLKSEL = 0;

PLLCTL\_PLLON=1; // turn on PLL

#if FREQUENCE == BUS26PLL52

SYNR = 63|0xC0;

REFDV = 26|0x40;

#endif

#if FREQUENCE == BUS36PLL73

SYNR = 9|0xC0;

REFDV = 2|0x40;

#endif

#if FREQUENCE == BUS58PLL117

SYNR = 15|0xC0;

REFDV = 2|0x40;

#endif

#if FREQUENCE == BUS72PLL144

SYNR = 17|0xC0;

REFDV = 3|0x40;

#endif

#if FREQUENCE == BUS40PLL80

SYNR = 9|0xC0;

REFDV = 3|0x40;

#endif

#if FREQUENCE == BUS48PLL96

SYNR = 5|0xC0;

REFDV = 1|0x40;

#endif

#if FREQUENCE == BUS60PLL120

SYNR = 14|0xC0;

REFDV = 3|0x40;

#endif

#if FREQUENCE == BUS64PLL128

SYNR = 15|0xC0;

REFDV = 3|0x40;

#endif

#if FREQUENCE == BUS80PLL160

SYNR = 19|0xC0;

REFDV = 3|0x40;

#endif

POSTDIV = 0;

\_asm(nop);

\_asm(nop);

while(0 == CRGFLG\_LOCK);

//CLKSEL = 0x80;

CLKSEL\_PLLSEL =1;

}

/\* SCI Init\*/

void SCI0Init(void)

{

SCI0BDL = (INT8U)((BUS\_FREQUENCE / 16 / SCI0\_BAUDRATE));

SCI0BDH = 0x00;

SCI0CR1 = 0x00;

SCI0CR2 = 0x0C;

}

void SCI0SendByte(INT8U out\_byte)

{

while(SCI0SR1\_TC != 1);

while(SCI0SR1\_TDRE != 1);

SCI0DRL = out\_byte;

}

INT8U SCI0ReceiveByte(void)

{

while(1 == SCI0SR1\_RDRF);

return SCI0DRL;

}

void SCI0SendString(INT8U \*str)

{

while(\*str){

SCI0SendByte(\*str);

str++;

}

}

void SCISendCmd(INT8U c)

{

SCI0SendByte(1);

SCI0SendByte(255);

SCI0SendByte(c);

SCI0SendByte(1);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*延时部分\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

INT8U best;

void Delay1us(INT16U t)

{

while(t--);

}

void Delay5us(INT16U t)

{

INT8U i;

while(t--){

for(best = 0;best<30;i++);

}

}

void Delay1ms(INT16U t)

{

INT16U i;

while(t--){

for(i = 0;i<8200;i++);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 图像采集、处理以及控制策略 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include "Process.h"

#define DEBUG\_MODE 0

#define BDM\_ONLINE 0

#define STEP\_DEBUG 0

int a[4] = { 0x01, 0x04, 0x0A, 0x0B };

//返回变量是否在所在范围

#define VarLimit(var,min,max) (((var) >= (min))&&((var) <= (max)))

//摄像头采集相关变量

INT8U VideoStatus; //图像处理状态标志

INT8U VideoStopCollectFlag; //停止当前场的采集

INT8U VideoEvenFlag; //奇偶场标志

INT16S VideoOutputLine; //当前摄像头输出行

INT16S VideoCollectLine; //图像采集行

INT16S VideoCollectInterval; //图像采集行

INT8U Video[VIDEO\_HEIGHT][VIDEO\_WIDTH]; //摄像头数据

INT8U \*VideoP, \*VideoCollectP;

INT8S VIDEO\_HEIGHTSearchStart[VIDEO\_HEIGHT] = { 0 }; //扫描范围

INT8S VIDEO\_HEIGHTSearchEnd[VIDEO\_HEIGHT] = { 0 }; //扫描范围

//图像参数

INT8S PosStore1[VIDEO\_HEIGHT]; //存放每行黑线位置数组 交替存储

INT8S PosStore2[VIDEO\_HEIGHT];

INT16S PosStoreAdjusted1[VIDEO\_HEIGHT];

INT16S PosStoreAdjusted2[VIDEO\_HEIGHT];

INT8S PosStoreFlag = 1; //存放辅助标志 为1半场存放在1 为2本场存放在2

INT8S \*CurrentPos; //本场黑线位置

INT8S \*PrevPos; //上场黑线位置

INT16S \*CurrentPosAdjusted; //上场黑线位置

INT16S \*PrevPosAdjusted; //上场黑线位置

INT8U Dianji\_data;

INT16U Duoji\_data;

INT8U THRESHOLD = 155;

INT8U zuo\_danxian = 1, you\_danxian = 1;

INT16U image\_center[VIDEO\_HEIGHT] = { 0 };

INT16U sum = 0, dissum = 0;

INT8U zhongduan = 1;

INT8U count = 0;

INT8U b = 0;

INT8S PrlineLeft, PrlineRight;

INT8U Flag, PrcourtLeft, PrcourtRight, TmpFlag = 0,shiziflag=0;

//记录各行黑块信息

INT8S BlockPos[VIDEO\_HEIGHT][BLACK\_BLOCK\_MAX\_NUM]; //每行搜寻到的各黑块位置

INT8S \*BlockPosP;

INT8S BlockStart[VIDEO\_HEIGHT][BLACK\_BLOCK\_MAX\_NUM]; //每行搜寻到的各黑块的起始点

INT8S \*BlockStartP;

INT8S BlockEnd[VIDEO\_HEIGHT][BLACK\_BLOCK\_MAX\_NUM]; //每行搜寻到的各黑块的终止点

INT8S \*BlockEndP;

INT8U BlockSelectedIndex[VIDEO\_HEIGHT]; //每行最终黑线选取的黑块索引

INT8U BlockCount[VIDEO\_HEIGHT]; //每行搜索到的各黑块个数

INT8U ValidBottomVIDEO\_HEIGHT; //有效基行 越往低值越大

INT8U ValidTopVIDEO\_HEIGHT; //最顶端有效行位置 越往顶值越小

//电机参数 MOTOR\_PWM\_MIDDLE中值，大于MOTOR\_PWM\_MIDDLE为前进，小于MOTOR\_PWM\_MIDDLE为后退

INT16S MotorPWM = 0; //电机PWM

INT16S MotorPWMPrev = 500; //电机PWM

INT16S MotorFeedBackAvg = 0; //当前电机平均值 以0位基础

INT16S MotorMax = 0; //最大速度

INT16S MotorMin = 0; //最小速度

INT16S MotorMinReal = 0; //根据情况计算的真实速度

INT16U MotorStatusFlag = 0; //标定电机所处状况

INT16U MotorDiffFlag = 0; //标定电机差速所处状况

//舵机参数

INT16S SteerPWM = 0; //本场舵机PWM

INT16S SteerPWMPrev = STEER\_PWM\_MIDDLE; //上场舵机PWM

INT16U SteerMode = 0;

//智能车状态

INT8U CarStatus = 0;

INT16S ValidVIDEO\_HEIGHT = 0; //本场有效行数

INT16S ValidVIDEO\_HEIGHTNumPrev = 0;

INT16S OffsetAvg = 0; //本场图像平均值

INT16S OffsetAvgPrev = 0;

INT16S OffsetAvgAdjusted = 0; //本场图像平均值

INT16S OffsetAvgAdjusted1 = 0; //本场图像平均值

INT16S OffsetAvgAdjustedPrev = 0;

INT16S OffsetLeftMax = 0; //左边最大值

INT16S OffsetRightMax = 0; //右边最大值

INT8U OffsetLeftMaxIndex = 0; //左边最大值Index

INT8U OffsetRightMaxIndex = 0; //右边最大值Index

INT16S AngleTotalAdjusted = 0; //整体角度

INT16S AngleTotalAdjustedPrev = 0; //整体角度

INT16S AngleAdjustedTop1Of3 = 0;

INT16S AngleAdjustedMiddle1Of3 = 0;

INT16S AngleAdjustedBottom1Of3 = 0;

INT16S SlopeBottomHalf = 0; //斜率1

INT16S SlopeTopHalf = 0; //斜率2

INT16S AngleBottomHalf = 0;

INT16S AngleTopHalf = 0;

INT16S SlopeAdjustedVIDEO\_HEIGHTD[VIDEO\_HEIGHT]; //各行偏差

INT16S SlopeAdjustedLeftDMax = 0; //和顶部底部直线偏差最大值

INT16S SlopeAdjustedRightDMax = 0; //和顶部底部直线偏差最大值Index

INT8U SlopeAdjustedLeftDMaxIndex = 0; //和顶部底部直线偏差最大值

INT8U SlopeAdjustedRightDMaxIndex = 0; //和顶部底部直线偏差最大值Index

INT16U CrashedFrameNum = 0; //图像已经丢失的场数

INT8U IsInvalidFrame = 0; //本场是否为无效

INT8U IsInvalidFramePrev = 0; //上场是否为无效

//路况参数

INT16U RoadStatus = 0; //路况信息

INT16U RoadStatusPrev = 0;

//起点识别

INT8U StartLineFlag = 0;

INT8U StartLineFlagPrev = 0;

INT8U StartLineFlagPrevPrev = 0;

INT8U StartLineFlagPrevPrevPrev = 0;

INT8U StartLineLeftMaxNum = 0;

INT8U StartLineLeftMaxNumPrev = 0;

INT8U StartLineRightMaxNum = 0;

INT8U StartLineNum = 0;

INT8U StartLineRightMaxNumPrev = 0;

INT32U SprintDistance = 0;

INT32U SprintRunDistance = 0;

//运行分析参数

INT32U RunPulse = 0; //智能车前进距离脉冲数

INT32U RunTime = 0; //智能车行进时间 单位为毫秒

//动态调试

INT16S DynamicVar = 0;

INT8U LOOP = 0;

INT8U Temp = 0;

void SystemInit(void) {

//INT8U i;

//INT8S chooseNum;

DisableInterrupts; //disable interrupts

IRQCR\_IRQEN = OFF; //disable IRQ

PLLInit(); //init pll

PortInit(); //init port

IRQInit(); //init IRQ

SCI0Init();

TIMInit(); //init TIM

Led1Init;

Led2Init;

Led3Init;

Led4Init;

Led5Init;

Led6Init;

Led7Init;

Led8Init;

Led1Off;Led2Off;Led3Off;Led4Off;Led5Off;Led6Off;

Led7Off;Led8Off;

// Timerch0Init();

MotorPIDInit();

PWM01Init(); //init motor pwm

PWM67Init(); //init steer pwm

}

void LP\_image() {

INT8U \*a\_point, \*b\_point, \*c\_point;

INT8U a, b, c, d, i, j;

for (i = 0; i < VIDEO\_HEIGHT; i++) {

for (j = 1; j < VIDEO\_WIDTH; j++) {

a\_point = &Video[i][j - 1];

b\_point = &Video[i][j];

c\_point = &Video[i][j + 1];

a = \*a\_point;

b = \*b\_point;

c = \*c\_point;

if (a >= b) {

d = b;

b = a;

a = d;

}

if (a >= c) {

d = c;

c = a;

a = d;

}

if (b >= c) {

d = c;

c = b;

b = d;

}

\*(b\_point) = b;

}

}

}

int count\_centerline() {

INT8S a = 0, b = VIDEO\_WIDTH, i, j, TmpLeftStart, TmpLeftEnd, TmpRightStart, TmpRightEnd,shizi=0,shiziprev=100;

INT16U Suma = 0, Sumb = 0;

ValidVIDEO\_HEIGHT = 1;

PrlineLeft = PrcourtLeft;

PrlineRight = PrcourtRight;

TmpFlag = 0;

for (i = VIDEO\_HEIGHT - 1; i > 0; i--) {

if (PrlineLeft - 10 > 0) {

TmpLeftStart = PrlineLeft - 10;

} else {

TmpLeftStart = 0;

}

if (PrlineLeft + 10 < VIDEO\_WIDTH) {

TmpLeftEnd = PrlineLeft + 10;

} else {

TmpLeftEnd = VIDEO\_WIDTH;

}

if (PrlineRight - 10 > 0) {

TmpRightStart = PrlineRight - 10;

} else {

TmpRightStart = 0;

}

if (PrlineRight + 10 < VIDEO\_WIDTH) {

TmpRightEnd = PrlineRight + 10;

} else {

TmpRightEnd = VIDEO\_WIDTH;

}

if (Flag == 0) {

TmpLeftEnd = TmpRightStart = 50;

TmpLeftStart = 0;

TmpRightEnd = VIDEO\_WIDTH;

}

for (j = TmpLeftEnd; j > TmpLeftStart; j--) {

if (Video[i][j] < THRESHOLD) {

a = j;

Flag = 0;

break;

}

}

for (j = TmpRightStart; j < TmpRightEnd; j++) {

if (Video[i][j] < THRESHOLD) {

b = j;

Flag = 0;

break;

}

}

shizi=b-a;

if (shizi>shiziprev || (shiziflag && shizi==100 && shiziprev==100))

{

shiziflag=1;

a=PrlineLeft+1;

b=PrlineRight-1;

}

if (TmpFlag && ABS(PrlineLeft - a) > 8) {

ValidVIDEO\_HEIGHT = i + 1;

break;

}

if (TmpFlag && ABS(PrlineRight - b) > 8) {

ValidVIDEO\_HEIGHT = i + 1;

break;

}

PrlineLeft = a;

PrlineRight = b;

image\_center[i + 1] = (a + b) / 2;

if (i >= VIDEO\_HEIGHT - 3 && i <= VIDEO\_HEIGHT - 1) {

Suma = Suma + a;

Sumb = Sumb + b;

}

a = 0;

b = VIDEO\_WIDTH;

Flag = 1;

TmpFlag = 1;

shiziprev=shizi;

}

PrcourtLeft = (INT8U)(Suma / 3);

PrcourtRight = (INT8U)(Sumb / 3);

// Image\_Filte();

if (ValidVIDEO\_HEIGHT > 34)

return 0;

else

return 1;

}

void Car\_cotrol() {

INT8U i, tmpheight;

INT16U Duo\_kp = 0;

sum = 0;

dissum = 0;

if (20 < ValidVIDEO\_HEIGHT)

tmpheight = ValidVIDEO\_HEIGHT;

else

tmpheight = 20;

for (i = tmpheight; i < VIDEO\_HEIGHT; i++) {

sum = sum + image\_center[i];

}

sum = sum / (VIDEO\_HEIGHT - tmpheight);

for (i = ValidVIDEO\_HEIGHT + 3; i > ValidVIDEO\_HEIGHT; i--) {

dissum = dissum + image\_center[i];

}

dissum = dissum / 3;

/\*Duo\_kp = (INT16U)(ABS(sum - 50) \* ABS(sum - 50)) \* 110 / 100;

if (sum < 43) {

Duoji\_data = STEER\_PWM\_MIDDLE + Duo\_kp;

} else {

Duoji\_data = STEER\_PWM\_MIDDLE - Duo\_kp;

} \*/

if (sum < 50) {

Duoji\_data = STEER\_PWM\_MIDDLE + (INT16U)(ABS(sum - 50) \* ABS(sum - 50)) \* 40 / 100 + (50 - sum )\*3/\*(MotorPID\_FeedBackReal - MOTOR\_PWM\_MIDDLE)\*30/100\*/;

Led1On;Led8Off;

} else {

Duoji\_data = STEER\_PWM\_MIDDLE - (INT16U)(ABS(sum - 50) \* ABS(sum - 50)) \* 60 / 100 - (sum - 50)\*4/\*(MotorPID\_FeedBackReal - MOTOR\_PWM\_MIDDLE)\*30/100\*/;

Led8On;Led1Off;

}

if (ABS((INT8S)sum-50)>0 && ABS((INT8S)sum-50)<15)

{

Led2On;

Led3Off;

}

if (ABS((INT8S)sum-50)>=17)

{

Led3On;

Led2Off;

}

}

void PWMDuo\_Duty(INT16U duo\_duty) {

PWMDTY67 = duo\_duty;

}

void PWMdian\_Duty(INT8U duty) {

PWMDTY0 = duty;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function Name:SystemStart

Description: Start Sysmtem

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void SystemStart(void) {

{

INT8U i, tmpcount1 = 0, j;

INT16U tmpsss = 0, tmpcount = 0, tmpcount2 = 0, tmpflagg = 0,loop=0;

EnableInterrupts;

Flag = 0;

//数据循环处理

for (;;) {

//等待场切换

while (VIDEO\_EVEN\_STATUS == VideoEvenFlag)

;

VideoEvenFlag = VIDEO\_EVEN\_STATUS;

VideoStatus = VIDEO\_WAIT;

VideoStopCollectFlag = 0;

VideoOutputLine = 0; //输出行数

VideoCollectLine = 0; //采集行数

IRQCR\_IRQEN = ON; //开启中断

//Delay1ms(1000);

SetMotorFeedBack;

b = 0;

if (Sw1)

b = b + 1;

if (Sw2)

b = b + 2;

if (Sw3)

b = b + 4;

if (Sw4)

b = b + 8;

while (VideoCollectLine < VIDEO\_HEIGHT)

;

LP\_image();

shiziflag=0;

if (count\_centerline()) {

if (shiziflag) Led7On;

else Led7Off;

Flag = 1;

IsInvalidFrame = 0;

Car\_cotrol();

MotorPWM = MOTOR\_PWM\_MIDDLE + 85;

} else {

Flag = 0;

IsInvalidFrame=1;

if (IsInvalidFramePrev)

if (SteerPWMPrev>STEER\_PWM\_MIDDLE) Duoji\_data = SteerPWMPrev + 5;

else Duoji\_data = SteerPWMPrev - 5;

AllLedOff;

MotorPWM = 585;

}

if (Duoji\_data > STEER\_PWM\_MAX) {

Duoji\_data = STEER\_PWM\_MAX;

}

if (Duoji\_data < STEER\_PWM\_MIN) {

Duoji\_data = STEER\_PWM\_MIN;

}

SteerPWM = Duoji\_data;

if (Sw3) {

PWMDuo\_Duty(SteerPWM);

} else {

PWMDuo\_Duty(STEER\_PWM\_MIDDLE);

}

if (Sw4)

MotorPWM = 600;

MotorPID\_Ref = MotorPWM;

MotorPIDCalculate();

if (Sw5)

MotorPID\_Value = 500;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 保留可能用到的值 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

IsInvalidFramePrev = IsInvalidFrame;

SteerPWMPrev = SteerPWM;

//RoadStatusPrev=RoadStatus;

MotorPWMPrev = MotorPWM;

if (!IsInvalidFrame) {

ValidVIDEO\_HEIGHTNumPrev = ValidVIDEO\_HEIGHT;

}

if (loop<80) loop++;

// else

if (MotorPID\_Value >= MOTOR\_PWM\_MIDDLE) {

MOTOR\_FORWARD((INT8U)(MotorPID\_Value - MOTOR\_PWM\_MIDDLE));

} else {

MOTOR\_BACKWARD((INT8U)(MOTOR\_PWM\_MIDDLE - MotorPID\_Value));

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 传送数据 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

if (Sw1) {

SCI0SendByte(a[0]);

SCI0SendByte(a[1]);

}

SCISendCmd(3);

SCI0SendByte(sum);

SCI0SendByte(sum);

/\* SCI0SendByte(MotorPID\_Ref-MOTOR\_PWM\_MIDDLE);

SCI0SendByte(MotorPID\_Value - MOTOR\_PWM\_MIDDLE);

SCI0SendByte(MotorPID\_FeedBackReal - MOTOR\_PWM\_MIDDLE);

SCI0SendByte(MotorPID\_Ref-MOTOR\_PWM\_MIDDLE); \*/

SCI0SendByte(sum);

SCI0SendByte(sum);

SCI0SendByte(sum);

SCI0SendByte(sum);

SCISendCmd(4);

SCI0SendByte(HIGH\_BYTE\_OF\_INT(STEER\_PWM\_MIDDLE-SteerPWM));

SCI0SendByte(LOW\_BYTE\_OF\_INT(STEER\_PWM\_MIDDLE-SteerPWM));

SCI0SendByte(HIGH\_BYTE\_OF\_INT(STEER\_PWM\_MIDDLE - SteerPWM));

SCI0SendByte(LOW\_BYTE\_OF\_INT(STEER\_PWM\_MIDDLE - SteerPWM));

SCISendCmd(5);

SCI0SendByte(9);

SCI0SendByte(ValidVIDEO\_HEIGHT);

SCI0SendByte(sum);

SCI0SendByte(dissum);

SCI0SendByte(Duoji\_data / 100);

SCI0SendByte(Duoji\_data % 100);

SCI0SendByte(PrlineLeft);

SCI0SendByte(PrlineRight);

SCI0SendByte(PrcourtLeft);

SCI0SendByte(PrcourtRight);

if (!Sw1) {

SCISendCmd(1);

for (i = 0; i < VIDEO\_HEIGHT; i++) {

for (j = 0; j < VIDEO\_WIDTH; j++) {

SCI0SendByte(Video[i][j]);

}

}

}

VideoEvenFlag = VIDEO\_EVEN\_STATUS;

}

}

}

pragma CODE\_SEG \_\_NEAR\_SEG NON\_BANKED

\_\_interrupt void IRQ\_ISR(void) {

static INT8S i;

if (VIDEO\_WAIT == VideoStatus) {

VideoOutputLine++;

if (VideoOutputLine >= 50) {

VideoOutputLine = 5; //重置输出行数，以便采集

VideoCollectInterval = 4;

VideoStatus = VIDEO\_ACQUISITION;

}

} else if (VIDEO\_ACQUISITION == VideoStatus) {

VideoOutputLine++;

if (VideoStopCollectFlag) {

//停止采集 关闭中断

IRQCR\_IRQEN = OFF;

} else {

if (VideoOutputLine >= VideoCollectInterval) {

VideoCollectP = Video[VideoCollectLine];

for (i = 0; i < 35; i++) {

VideoCollectP[i] = VIDEO\_PORT;

}

VideoCollectP = Video[VideoCollectLine];

for (i = 0; i < VIDEO\_WIDTH; i++) {

VideoCollectP[i] = VIDEO\_PORT;

asm {

nop;

nop;

}

}

VideoCollectLine++;

if (VideoCollectLine >= VIDEO\_HEIGHT) {

//采集结束 关闭中断

IRQCR\_IRQEN = OFF;

}

VideoOutputLine = 0;

if (VideoCollectLine % 5 == 0) {

if (VideoCollectLine == 5) {

VideoCollectInterval = 2;

} else if (VideoCollectLine == 10) {

VideoCollectInterval = 3;

} else if (VideoCollectLine == 15) {

VideoCollectInterval = 4;

} else if (VideoCollectLine == 20) {

VideoCollectInterval = 5;

} else if (VideoCollectLine == 25) {

VideoCollectInterval = 6;

} else if (VideoCollectLine == 30) {

VideoCollectInterval = 7;

} else if (VideoCollectLine == 35) {

VideoCollectInterval = 7;

}

}

}

}

}

}

/\*interrupt 5 void timer0(void){

TFLG1\_C0F=1;

TC0=TCNT+0x30D4;

SetMotorFeedBack;

//baitou\_ctr();

} \*/