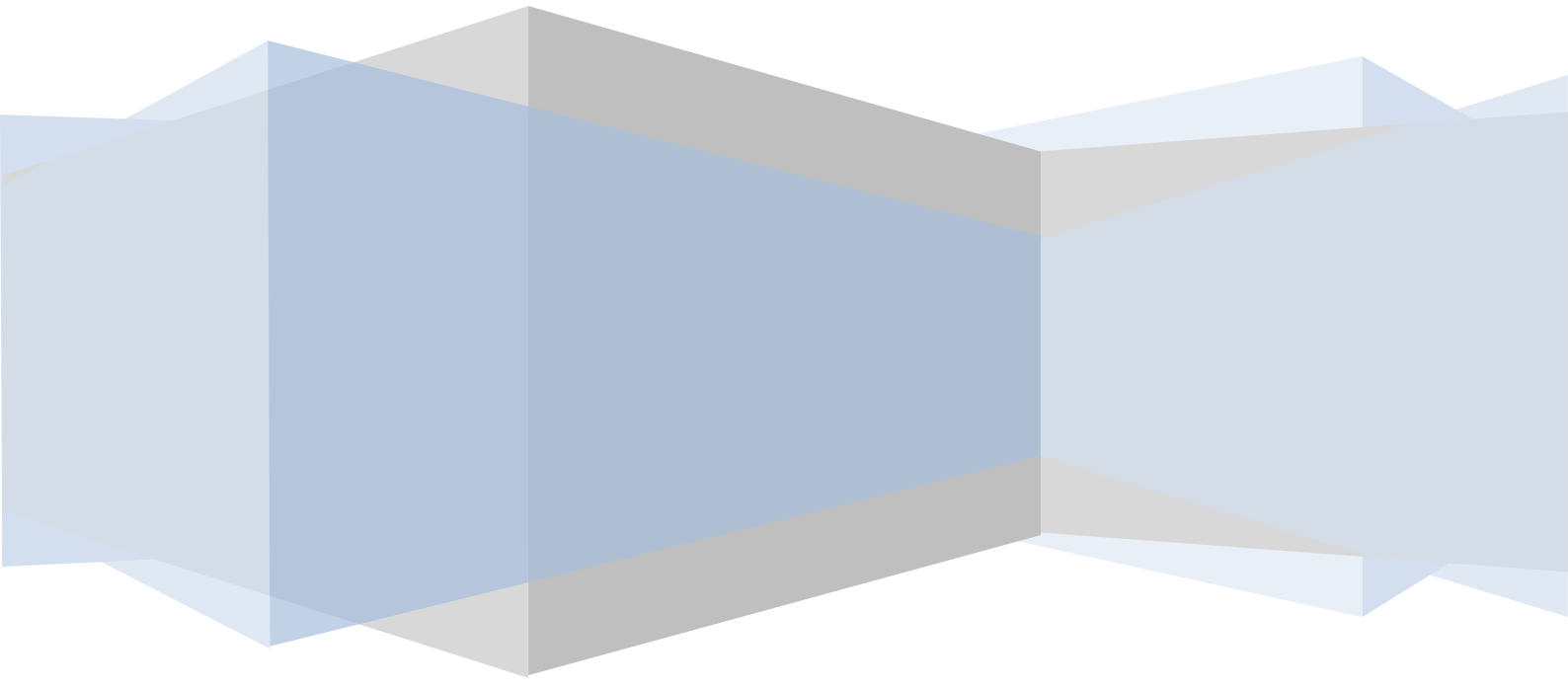
浙江大学光电信息工程学系

颜色识别系统设计报告

光电检测技术及系统课程项目

项目成员：张津 马超璇



目录

摘要

本文采用 C8051f020 单片机结合 TCS3200D 颜色传感器来进行色彩的识别功能。通过 C8051f020 的 I/O 口依次选通 TCS3200D 颜色传感器的 R、G、B 三个通道，分别对反射光的红、绿、蓝三种色光光强进行测量，然后合成该颜色的 RGB 值在 LCD 上显示。本文还根据 HSV（Hue, Saturation, Value）颜色模型计算了该颜色的 HSV 值进行显示。另外，本文还自行设计了通过颜色的 RGB 值和 HSV 值计算感官颜色的算法，并编写了程序。本系统因此不仅能够显示颜色的 RGB 值和 HSV 值，而且能够判断颜色的大致分类，例如红、绿、蓝、黄、紫、黑、白等。经测试，颜色识别器的识别快速准确，而且其具有体积小、成本低等优点，可用于大批量的工业生产。

关键字：色彩识别 TCS3200D 颜色传感器 C8051f020 单片机 LCD

1 绪论

1.1 研究背景

随着现代工业生产向高速化、自动化方向的发展，颜色识别广泛应用于各种工业检测和自动控制领域，而生产过程中长期以来由人眼起主导作用的颜色识别工作将越来越多地被相应的颜色传感器所替代。如：各种物体表面颜色识别（产品包装色标检测，产品外表特征颜色的检测，液体溶液颜色变化过程的检测与控制，等等）。

标准的颜色测量方法是采用光谱光度测色仪，通过测量样品的三刺激值，从而得到样品的颜色。目前，基于各种原理的颜色识别传感器有两种基本类型：一是 RGB(红绿蓝)颜色传感器，检测的是三刺激值；二是色标传感器，检测被测物体与标准颜色的色差。这类装置许多是漫反射型、光束型和光纤型的，封装在各种金属和聚碳酸酯外壳中。

目前的颜色传感器通常是在独立的光电二极管上覆盖经过修正的红、绿、蓝滤光片，然后对输出信号进行相应的处理，才能将颜色信号识别出来；有的将两者集合起来，但是输出模拟信号，需要一个 A/D 电路进行采样，对该信号进一步处理，才能进行识别，增加了电路的复杂性，并且存在较大的识别误差，影响了识别的效果。

本文所用的颜色传感器 TCS3200D，不仅能够实现颜色的识别与检测，与以前的颜色传感器相比，还具有许多优良的新特性。TCS3200D 这种可编程的彩色光到频率转换器适合于色度计测量应用领域，如彩色打印、医疗诊断、计算机彩色监视器校准以及油漆、纺织品、化妆品和印刷材料的过程控制和色彩配合。

1.2 颜色传感器的分类

目前，用于颜色识别的传感器有两种基本类型。其一是色标传感器，其二是 RGB 颜色传感器，也是本文接下来研究的重点。

1） 色标传感器：色标传感器常用于检测待测色标或物体上的斑点，通过与非色标区（或背景）相比较来实现色标检测，而不是对颜色进行直接测量。光源垂直于目标物体安装，而接收器与物体成锐角方向安装，让它只检测来自目标物体的散射光，从而避免传感器直接接接收反射光。此类传感器又分为两种，一类是以白炽灯为光源，以白炽灯为基础的传感器用有色光源检测颜色，这种白炽灯发射包括红外在内的各种颜色的光，因此用这种光源的传感器可在很宽范围上检测颜色的微小变化。另外，自炽灯传感器的检测电路通常都十分简单，因此可获得极快的响应速度。然而，白炽灯不允许振动和延长使用时同，因此不适用于有严重冲击和振动的场合。一类是以单色光源，使用单色光源(即绿色或红色 LED)的色标传感器就其原理来说并不是检测颜色，它是通过检测色标对光束的反射或吸收量与周围材料相比的不同而实现检测的 所以，颜色的识别要严格与照射在目标上的光

谱成分相对应。在单色光源中．绿光 LED(565mm)和 LED((660mm)各有所长 绿光 LED 比白炽灯寿命长，并且在很宽的颜色范围内比红光源灵敏度高。红光 LED 对有限的颜色组合有响应．但它的检测距离比绿光 LED 远。通常红光源传感器的检测距离是绿光源传感器的 6～8 倍。

2） RGB 颜色传感器：RGB 颜色传感器对相似颜色和色调的检测可靠性较高。它是通过测量构成物体颜色的三基色的反射比率实现颜色检测的，由于这种颜色检测法精密度极高，所以 RGB 传感器能准确区别极其相似的颜色，甚至相同颜色的不同色调。一般 RGB 传感器都有红、绿、蓝三种光源，三种光通过同一透镜发射后被目标物体反射。光被反射或吸收的量值取决于物体颜色。RGB 传感器有两种测量模式 一种是分析红、绿、蓝光的比例 因为检测距离无论怎样变化，只能引起光强的变化，而三种颜色光的比例不会变．因此，即使在目标有机械振动的场合也可检测。第二种模式是利用红绿蓝三基色的反射光强度实现检测目的。利用这种模式可实现微小颜色判别的检测，但传感器会受目标机械位置的影响。

1.3 本项目的主要工作及意义

本文是以 C8051f020 单片机为核心，采用 TCS3200D 颜色传感器和 LCD 液晶模块建立起来的。文章从以下几个方面介绍该测量系统：

- 整个系统的设计原理，包括对三原色的感应原理和 TCS3200D 颜色传感器识别原理的分析；
- 系统实现。包括单片机和各模块之间的连接、整体系统的搭建、操作界面的设计等；
- 算法设计与实现。结合 RGB 值与 HSV 值自行设计了感官颜色（红、黄、绿、蓝、黑、白、紫等）判别的算法与程序实现；
- 实验验证。通过实验验证算法的有效性。

本文研究的色彩识别系统的意义在于利用经济的系统整合与设计，实现了颜色的自动检测与识别，降低了色彩识别的难度，而且检测结果有很高的准确性和可信性，将有利于自动化行业以及其它相关行业的发展，也可用于艺术工作者随时随地记录颜色的需要。另外，文中实现的颜色识别系统也可用于机器人视觉识别系统。

2 颜色识别及颜色传感器技术介绍

2.1 色彩识别原理和模型

所谓颜色模型就是指某个三维颜色空间中的一个可见光子集，它包含某个颜色域的所有颜色。例如，RGB 颜色模型就是三维直角坐标颜色系统的一个单位正方体。颜色模型的用途是在某个颜色域内方便的指定颜色，由于每一个颜色域都是可见光的子集，所以任何一个颜色模型都无法包含所有的可见光。在大多数的彩色图形显示设备一般都是使用红、绿、蓝三原色，我们的真实感图形学中的主要的颜色模型也是 RGB 模型，但是红、绿、蓝颜色模型用起来不太方便，它与直观的颜色概念如色调、饱和度和亮度等没有直接的联系。

颜色模型主要有 HSV、RGB、HSI、CHL、LAB、CMY 等。它们在不同的行业各有所指，但在计算机技术方面运用最为广泛。本文主要应用了 RGB 模型和 HSV 模型。

2.1.1 RGB 颜色模型

RGB(red,green,blue)颜色空间最常用的用途就是显示器系统，彩色阴极射线管,彩色光栅图形的显示器都使用 R、G、B 数值来驱动 R、G、B 电子枪发射电子，并分别激发荧光屏上的 R、G、B 三种颜色的荧光粉发出不同亮度的光线，并通过相加混合产生各种颜色；扫描仪也是通过吸收原稿经反射或透射而发送来的光线中的 R、G、B 成分，并用它来表示原稿的颜色。RGB 色彩空间称为与设备相关的色彩空间,因为不同的扫描仪扫描同一幅图像，会得到不同色彩的图像数据；不同型号的显示器显示同一幅图像，也会有不同的色彩显示结果。显示器和扫描仪使用的 RGB 空间与 CIE 1931 RGB 真实三原色表色系统空间是不同的，后者是与设备无关的颜色空间。btw：Photoshop 的色彩选取器(Color Picker)。可以显示 HSB、RGB、LAB 和 CMYK 色彩空间的每一种颜色的色彩值。

根据三基色原理，用基色光单位来表示光的量，则在 RGB 颜色空间，任意色光 F 都可以用 R、G、B 三色不同分量的相加混合而成：

F＝r[R]+g[G]+b[B]

如图 2.1.1 所示，RGB 颜色空间还可以用一个三维的立方体来描述。

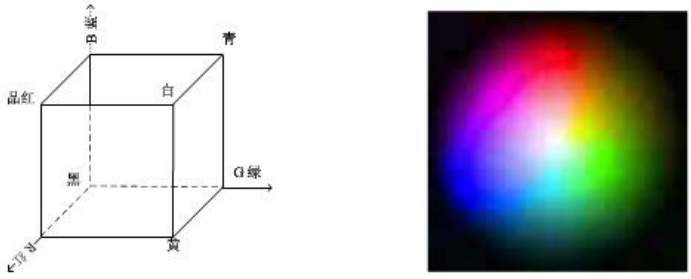


图 2-1-1RGB 颜色模型

我们可知自然界中任何一种色光都可由 R、G、B 三基色按不同的比例相加混合而成，当三基色分量都为 0（最弱）时混合为黑色光；当三基色分量都为 k（最强）时混合为白色光。任一颜色 F 是这个立方体坐标中的一点，调整三色系数 r、g、b 中的任一系数都会改变 F 的坐标值，也即改变了 F 的色值。RGB 颜色空间采用物理三基色表示，因而物理意义很清楚，适合彩色显像管工作。然而这一体制并不适应人的视觉特点。因而，产生了其他不同的颜色空间表示法。

2.1.2 HSV 颜色模型

HSV (色相 hue,饱和度 saturation, 明度 value), 也称 HSB (B 指 brightness)是艺术家们常用的，这是因为与加法减法混色的术语相比，使用色相，饱和度等概念描述色彩更自然直观。H 指 hue（色相），S 指 saturation（饱和度），V 指 value(色调)。

- 色相（H）是色彩的基本属性，就是平常所说的颜色名称，如红色、黄色等。
- 饱和度（S）是指色彩的纯度，越高色彩越纯，低则逐渐变灰，取 0-100%的数值。
- 明度（V），亮度（L），取 0-100%。

该模型使用色相（X 轴）、饱和度（Y 轴）和明度（Z 轴）表示。

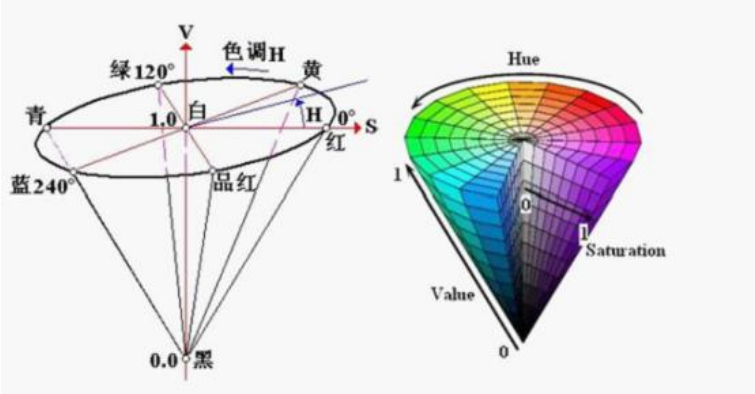


图 2-1-2 HSV 颜色模型

HSV 颜色空间的模型对应于圆柱坐标系中的一个圆锥形子集，可以用一个圆锥空间模型来描述。圆锥的顶面对应于 V=1。它包含 RGB 模型中的 R=1，G=1，B=1 三个面，所代表的颜色较亮。色彩 H 由绕 V 轴的旋转角给定。红色对应于角度 0°，绿色对应于角度 120°，蓝色对应于角度 240°。在 HSV 颜色模型中，每一种颜色和它的补色相差 180°。饱和度 S 取值从 0 到 1，所以圆锥顶面的半径为 1。HSV 颜色模型所代表的颜色域是 CIE 色度图的一个子集，这个模型中饱和度为百分之百的颜色，其纯度一般小于百分之百。在圆锥的顶点(即原点)处，

V=0,H 和 S 无定义，代表黑色。圆锥的顶面中心处 S=0，V=1,H 无定义，代表白色。从该点到原点代表亮度渐暗的灰色，即具有不同灰度的灰色。对于这些点，S=0,H 的值无定义。可以说，HSV 模型中的 V 轴对应于 RGB 颜色空间中的主对角线。在圆锥顶面的圆周上的颜色，V=1，S=1,这种颜色是纯色。HSV 模型对应于画家配色的方法。画家用改变色浓和色深的方法从某种纯色获得不同色调的颜色，在一种纯色中加入白色以改变色浓，加入黑色以改变色深，同时加入不同比例的白色，黑色即可获得各种不同的色调。

2.2 RGB 值测量

基于以上两种模型，识别颜色有很多种算法。其中对测量颜色的准确性有很大影响的是白平衡算法，这也是不同测量技术中所不可或缺的算法。

本文中的 TCS3200D 将分别选通红、绿、蓝滤波器时的输出光强值转换为频率信号量化输出 R、G、B 值，进而计算出颜色。由于在不同测量条件下所用参考白光并不是标准的白光，其 R、G、B 三个分量并不完全相同，故每次测量前都要重新进行白平衡调整才能保证测量的准确度。

所谓白平衡，就是首先测量出基准光源的 RGB 值，并将此 RGB 值设定为响应信道的参考 1。然后将测量出的在标准光源下的光强值分别与标准光源的 RGB 光强值求比即可，用公式表示如下 2.2.1-3。

$$R = P_{物红}/P_{源红} \quad 2.2.1$$

$$G = P_{物绿}/P_{源绿} \quad 2.2.2$$

$$B = P_{物蓝}/P_{源蓝} \quad 2.2.3$$

由于 RGB 模型中颜色坐标在 0~255 之间，故将所比结果乘以 255 即可得测量颜色的标准 RGB 值。

2.3 颜色传感器技术

2.3.1 颜色检测的难点

影响颜色检测准确度的参数主要有：参考光源、光源方位、传感器性能等。

- 光源的影响
  - ① 光源本身 RGB 值非均匀性的影响；
  - ② 参考光源中混有太阳光和外界杂散光后的影响。
- 光源方位和测量方位的影响

两者都会影响进入传感器的光强值从而影响测量结果。故实验中需注意选择最佳

的测量位置和方位，方能保证最精确的测量。

2.3.2 颜色传感器

常用的 RGB 颜色传感器的测量原理为：在三个光电二极管上贴上三基色滤波片，依次测量反射光的强度测量各颜色的成分。该测量方法的精度极高，能正确区分极其相似的颜色。

它有两种测量模式：一种是分析红、绿、蓝光的比例。这种模式的测量结果不会因测量距离的改变而变化；另一种是分析三基色的光强。这种方法受测量时的距离影响很大，可实现微小颜色判别的检测。

2.4 本章小结

本章主要对色彩识别和识别传感器的一般原理进行了介绍，介绍了 RGB 和 HSV 色彩模型，并对白平衡以及 RGB 值测量的原理进行了分析。最后对颜色测量过程中可能存在的难点进行了总结。

3 颜色识别系统的硬件设计

本项目主要是基于 C8051f020 单片机、TCS3200D 颜色传感器和 LCD12864 模块进行系统搭建和测量的。下面分别做简要介绍。

3.1 C8051f020 单片机简介

C8051f020 单片机是完全集成的混合信号系统级 MCU 芯片（SOC），它使用 Cygnal 的专利 CIP-51 微处理器内核，CIP-51 与 MCS-51 指令集完全兼容。它采用流水线结构，与标准额 8051 结构相比指令执行速度有很大的提高。CIP-51 提供了 22 个中断源，允许大量的模拟和数字外设中断微控制器，因而有更高的执行效率。具有 64 个 I/O 引脚，每个端口都可以配置成推挽或漏极开路输出。C8051f020 MCU 内部有一个 SMBUS/I^2C 结构、两个具有增强型波特率配置的全双工 UART 和一个增强型 SPI 结构，每种串行总线完全用硬件实现，都能向 CIP-51 产生中断。它内部有一个 12 位的 ADC0，该子系统包括一个 9 通道的可编程模拟多路选择器（AMUX0），一个可编程增益放大器（PGA0）和一个 100ksps、12 位分辨率的逐次逼近器型 ADC，ADC 中继承了跟踪保持电路和可编程窗口检测器：一个 8 位 ADC1，包括一个 8 通道的可配置模拟多路开关（AMUXI），一个可编程增益放大器（PGA1）和一个 500ksps、8 位分辨率的逐次逼近寄存器型 ADC。两个 12 位 DAC 转换器，每个 DAC 都具有灵活的输出更新机制，允许无缝的满意变化，并支持无抖动输出更新。C8051f020 还有 5 个通用的 16 位定时器和 5 个捕捉比较模块的可编程计数器/定时器阵列。

3.2 TCS3200D 颜色识别原理简介

3.2.1 TCS3200D 芯片的结构框图与特点

TCS3200D 是 TAOS 公司推出的可编程彩色光到频率的转换器，它把可配置的硅光电二极管与电流频率转换器集成在一个单一的 CMOS 电路上，同时在单一芯片上集成了红绿蓝（RGB）三种滤光器，是业界第一个有数字兼容接口的 RGB 彩色传感器，TCS3200D 的输出信号是数字量，可以驱动标准的 TTL 或 CMOS 逻辑输入，因此可直接与微处理器或其他逻辑电路相连接，由于输出的是数字量，并且能够实现每个彩色信道 10 位以上的转换精度，因而不再需要 A/D 转换电路，使电路变得更简单，图 3.2.1 是 TCS3200D 的引脚和功能框图。

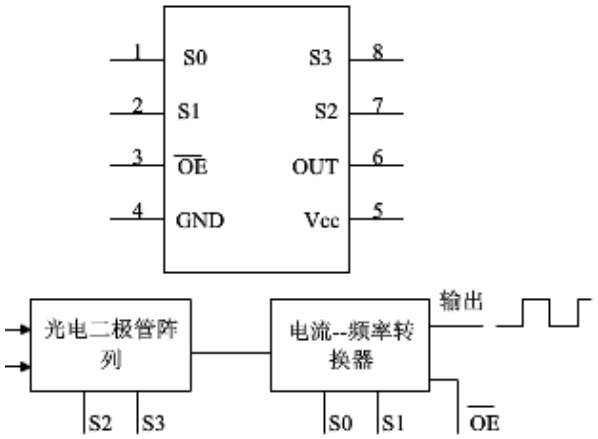


图 3-2-1 TCS3200D 的引脚和功能框图

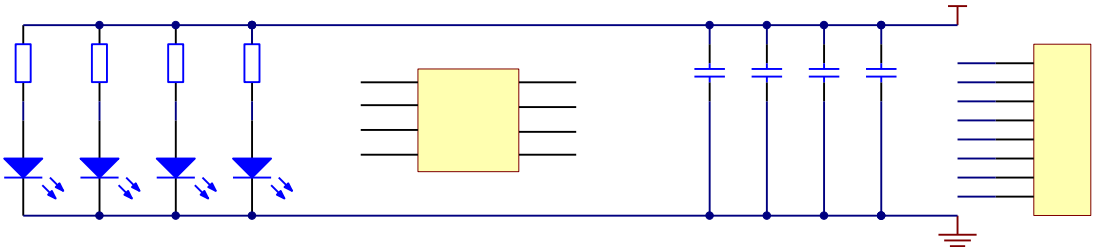


图 3-2-2　Protues 模拟 TCS3200D 的原理图

图 3.2.1 中，TCS3200D 采用 8 引脚的 SOIC 表面贴装式封装，在单一芯片上集成有 64 个光电二极管，这些二极管分为四种类型，其 16 个光电二极管带有红色滤波器；16 个光电二极管带有绿色滤波器；16 个光电二极管带有蓝色滤波器，其余 16 个不带有任何滤波器，可以透过全部的光信息，这些光电二极管在芯片内是交叉排列的，能够最大限度地减少入射光辐射的不均匀性，从而增加颜色识别的精确度；另一方面，相同颜色的 16 个光电二极管是并联连接的，均匀分布在二极管阵列中，可以消除颜色的位置误差。工作时，通过两个可编程的引脚来动态选择所需要的滤波器，该传感器的典型输出频率范围从 2Hz－500kHz，用户还可以通过两个可编程引脚来选择 100％、20％或 2％的输出比例因子，或电源关断模式。输出比例因子使传感器的输出能够适应不同的测量范围，提高了它的适应能力。例如，当使用低速的频率计数器时，就可以选择小的定标值，使 TCS3200D 的输出频率和计数器相匹配。

从表 3.2.2 可知：当入射光投射到 TCS230 上时，通过光电二极管控制引脚 S2、S3 的不同组合，可以选择不同的滤波器；经过电流到频率转换器后输出不同频率的方波（占空比是 50％），不同的颜色和光强对应不同频率的方波；还可以通过输出定标控制引脚 S0、S1，选择不同的输出比例因子，对输出频率范围进行调整，以适应不同的需求。

表 3.2.2 S0、S1 及 S2、S3 的组合选项

S0	S1	输出频率定标	S2	S3	滤波器类型
L	L	关断电源	L	L	红色
L	H	2%	L	H	蓝色
H	L	20%	H	L	无
H	H	100%	H	H	绿色

下面简要介绍 TCS3200D 芯片各个引脚的功能选项。

S0、S1	选择输出比例因子或电源关断模式
S2、S3	选择滤波器的类型
OE	频率输出使能引脚
OUT	频率输出引脚
GND	接地引脚
VCC	工作电压引脚

3.2.2 TCS3200D 识别颜色的原理

首先介绍一些光与颜色的基本知识。

（1）三原色的感应原理

通常所看到的物体颜色，实际上是物体表面吸收了照射到它上面的白光（日光）中的一部分有色成分，而反射出的另一部分有色光在人眼中的反应。白色是由各种频率的可见光混合在一起构成的，也就是说白光中包含着各种颜色的色光（如红 R、黄 Y、绿 G、青 V、蓝 B、紫 P）。根据德国物理学家赫姆霍兹（Helinholtz）的三原色理论可知，各种颜色是由不同比例的三原色（红、绿、蓝）混合而成的。

（2）TCS3200D 识别颜色的原理

由三原色感应原理可知，如果知道构成各种颜色的三原色的值，就能够知道所测试物体的颜色。对于 TCS3200D 来说，当选定一个颜色滤波器时，它只允许某种特定的原色通过，阻止其他原色的通过。例如：当选择红色滤波器时，入射光中只有红色可以通过，蓝色和绿色都被阻止，这样就可以得到红色光的光强；同时，选择其他的滤波器，就可以得到蓝色光和绿色光的光强。通过这三个值，就可以分析投射到 TCS3200D 传感器上的光的颜色。

（3）白平衡和颜色识别原理

对于 TCS3200D 的光传感器来说，它对这三种基本色的敏感性是不相同的，导致 TCS3200D 的 RGB 输出并不相等，因此在测试前必须进行白平衡调整，使得 TCS230 对所检测的"白色"中的三原色是相等的。

进行白平衡调整是为后续的颜色识别作准备。在本装置中，白平衡调整的具体步骤和方法如下：将一张白纸放置在传感器的上方，根据前面所介绍的方法，

依次选通红色、绿色和蓝色滤波器，分别测得红色、绿色和蓝色的值，然后就可计算出需要的 3 个调整参数。

当 TCS230 识别颜色时，就用这 3 个参数对所测颜色的 R、G 和 B 进行调整。这里有两种方法来计算调整参数：1、依次选通三颜色的滤波器，然后对 TCS230 的输出脉冲依次进行计数。当计数到 255 时停止计数，分别计算每个通道所用的时间，这些时间对应于实际测试时 TCS230 每种滤波器所采用的时间基准，在这段时间内所测得的脉冲数就是所对应的 R、G 和 B 的值。2、设置定时器为一固定时间（例如 10ms），然后选通三种颜色的滤波器，计算这段时间内 TCS230 的输出脉冲数，计算出一个比例因子，通过这个比例因子可以把这些脉冲数变为 255。在实际测试时，室外同样的时间进行计数，把测得的脉冲数再乘以求得的比例因子，然后就可以得到所对应的 R、G 和 B 的值。

本次项目设计我们应用的是方法 2。

应用中需要注意一下问题：

①颜色识别时要避免外界光线的干扰，否则会影响颜色识别的结果，最好把传感器、光源等放置在一个密闭、无反射的箱子中进行测试。

②对光源没有特殊的要求，但是光源发出的光要尽量集中，否则会造成传感器之间的相互干扰。

③ 当第 1 次使用 TCS3200D 时，或 TCS3200D 识别模块重启、更换光源等情况时，都需要进行白平衡调整。

3.3 液晶显示器显示原理简介

本项目所用 LCD 为 128*64 点阵的液晶显示器。其相关的指令格式和功能为：

- LCD 的寄存器选择：通过 RS 和 RW 引脚共同决定。选择情况见表 3.3.1。

表 3.3.1 LCD 内部寄存器选择表

RS	RW	寄存器及操作
0	0	指令寄存器写入
0	1	忙标志和地址计数器
1	0	数据寄存器写入
1	1	数据寄存器读出

- 总共有 11 条指令，它们的格式和功能如下表 3.3.2。

表 3.3.2 控制指令表

序号	指令	RS	RW	D7	D6	D5	D4	D3	D2	D1	D0
1	清显示	0	0	0	0	0	0	0	0	0	1
2	光标返回 0 0	0	0	0	0	0	0	1	*		
3	置输入模式	0	0	0	0	0	0	0	1	I/D	S
4	显示开/关控制	0	0	0	0	0	0	1	D	C	B
5	光标或字符移位	0	0	0	0	0	1	S/C	R/L	*	*
6	置功能	0	0	0	0	1	DL	N	F	*	*
7	置字符发生存储器地址	0	0	0	1	字符发生存储器地址					
8	置数据发生存储器地址	0	0	1	显示数据发生存储器地址						
9	读忙标志或地址	0	1	BF	计数器地址						
10	写数到 CGRAM 或 DDRAM	1	0	要写的数据内容							
11	从 CGRAM 或 DDRAM 读数	1	1	读出的数据内容							

3.4 颜色识别系统各部分的连接

3.4.1 TCS3200D 颜色传感器与 51 单片机的连接

将脉冲信号输出端 OUT 与单片机外中断 1 P3.5 口相连，S2/S3 与单片机 P3.0 和 P3.1 口相连。具体连线对应关系如下表：

表 3.4.1 TCS3200D 与 51 单片机的接口

EO	S0/S1	S2	S3	OUT	VCC/GND
P3.2	VCC	P3.0	P3.1	P3.5	VCC/GND

- P3.2 口作为频率输出使能端口，清零时使能输出；
- P3.0 和 P3.1 口作为颜色通道选择的控制端口，相关通道对应关系见表 3.2.2;
- P3.5 口（配置为外中断 1 的输入端口）与颜色传感器的输出口 OUT 连接。

3.4.2 LCD1602 与 51 单片机的连接

LCD 输出口与单片机引脚的连接对应关系为：

|DB0-----P5.0 | DB4-----P5.4 | RW-------P6.6 |

|DB1-----P5.1 | DB5-----P5.5 | RS--------P6.7 |

|DB2-----P5.2 | DB6-----P5.6 | E----------P6.5 |

|DB3-----P5.3 | DB7-----P5.7 |

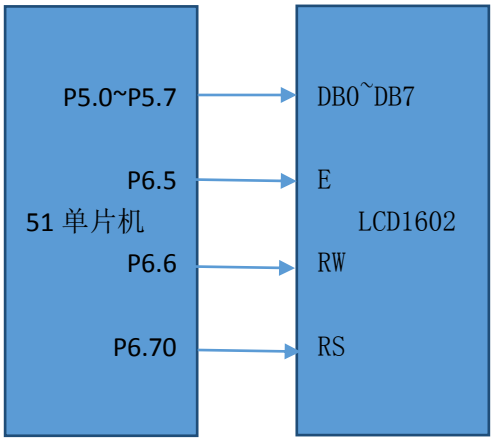


图 3.4.2 LCD 与 51 单片机的连接

3.5 本章小结

本章主要介绍了系统所用各功能模块的原理和模块之间的连接。以此硬件系统为依托我们进行了后续的软件编写和测量。

4 颜色识别系统的软件设计

4.1 系统软件框图

本系统通过 LCD 与按键实现与用户的人机交互。使用时，按 0 键进行白平衡校准，即帮助系统进行白色定义；按 1 键则进行实时的颜色测量。界面显示 RGB 值、HSV 值以及感官颜色判断的结果。同时，在用户的使用过程中，本系统还将用提示来辅助用户的测量。

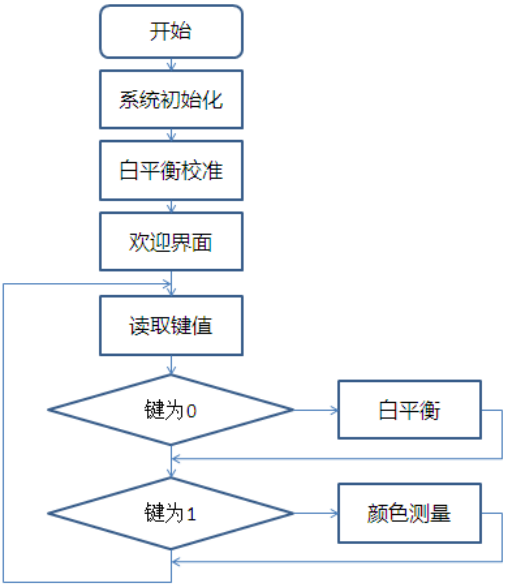


图 4-1 系统软件结构框图

4.2 白平衡

本设计主要应用单片机的定时器/计数器 T0 和 T1。其中 T0 用作定时器定时 10ms，作为颜色测量的基准时间；T1 用作计数器，对由光强信号转化输出的频率信号进行计数测量。白平衡所测得的各颜色分量的强度将作为之后颜色测量的基准使用。具体函数内容请见附录 1 中的白平衡子程序 void baipingheng()。

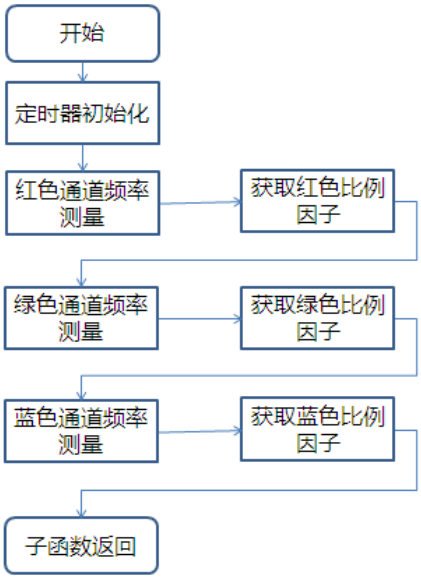


图 4-2 白平衡子程序流程图

4.3 RGB 值测量

在白平衡的基础上，将颜色传感器芯片置于待测颜色上（由于芯片本身是根据光强信息进行测量输出的，所以本芯片对测量距离的要求较高，测量时应保持距离不变），通过用 T0 定时 10ms，单片机控制依次选通红、绿、蓝 3 色滤波器，将 T1 的计数值分别与前述白平衡时相应 RGB 因子相比并与 255 相乘，即可输出该颜色的 RGB 值。具体函数内容请见附录 1 中的 RGB 测量子程序 void celiang()。

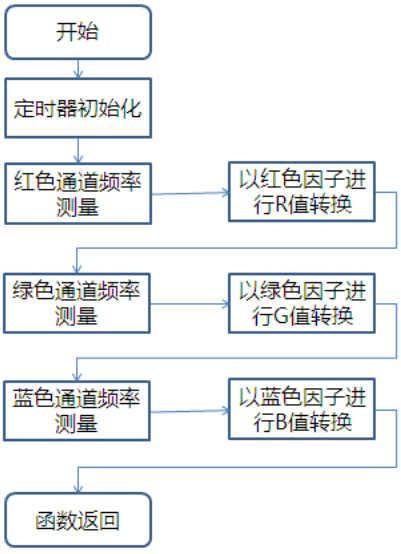


图 4-3 RGB 测量子程序流程图

4.4 HSV 算法

RGB 和 HSV 是对颜色的不同描述模型，它们之间存在一定的换算关系，具体算法为：

max=max(R,G,B)

min=min(R,G,B)

if R = max, H = (G-B)/(max-min)

if G = max, H = 2 + (B-R)/(max-min)

if B = max, H = 4 + (R-G)/(max-min)

H = H * 60

if H < 0, H = H + 360

V=max(R,G,B)

S=(max-min)/max

据此算法编写相应的 C51 程序即可实现转换。具体函数内容请见附录 1 中的 RGB 转 HSV 程序 void hsv(uint r, uint g, uint b)。

4.5 感官颜色判断算法

虽然颜色和 RGB 值之间存在着一一对应的关系，即不同颜色对应着不同的 RGB 值。但是由 RGB 反推颜色存在较大难度，没有成型的算法，因此我们在此块设计时遇到了较大障碍。

经过反复摸索，对不同颜色的输出 RGB 值进行比较总结，我们归纳出了自己的常见颜色的主观判别的算法，并成功实现了对红、绿、蓝、黄、红、黑和白等基本颜色的准确判别。同时，我们的算法还允许较大的容差，即对不同色度值的相同颜色（如深红、红、玫红、粉红等）都能实现准确判别。具体函数内容请见附录 1 中的颜色判别程序 void colordefine0(uintrr,uintgg,uint bb)

以下是算法所基于的颜色特征：

白色：亮度较大且各分量基本相等；

黑色：亮度很小且各分量基本相等；

红色：红色分量远大于蓝色和绿色分量，且蓝色和绿色分量基本相等；

绿色：绿色分量大于蓝色分量和红色分量；

蓝色：蓝色分量大于红色分量和绿色分量；

黄色：红色分量很大，绿色分量也较多，且绿色分量远大于蓝色分量；

紫色：蓝色分量大于绿色分量，红色分量也大于绿色分量，且红色分量与蓝色分量相当。

4.6 本章小结

本章对测量系统的各个模块的软件设计流程和方法做了介绍，包括 TCS3200D 颜色传感器的白平衡和测量方法等相关软件算法等。从而实现了整个系统的搭建。

5 色彩识别器系统的测试实验

5.1 色彩识别的测试实验

在上述硬件和软件设计的基础上，我们进行了颜色识别的测试。为了保证测试时将距离的影响降到最低，我们将颜色传感模块进行了封装，从而保证其在测试时距北侧物体表面的距离保持恒定。

测试结果可成功识别各种颜色（红、绿、蓝、黄、红、紫、黑、白等），并输出其对应的 RGB 值、HSV 值和最终判别的感官颜色。

实验测定表明，本设计对不同颜色的 RGB 值输出精度很高，在保证实验条件的基础上，测试 RGB 值的误差很小，从而完美实现了最基本的功能模块。

5.2 实验结果分析

由于测试时对测试距离和杂散光等外界因素进行了很好的限制和消除，本设计的 RGB 值输出具有很高的精度，无论是测量本身的稳定性（多次对同一颜色测量的输出值）还是对不同颜色输出的准确性（不同颜色的测定 RGB 值）都达到了很好的效果。

由于 HSV 的转化与 RGB 值之间只是软件算法的关系，且它们之间的转换关系已经很成熟，所以有最初 RGB 值转换的 HSV 值也同样具有很高的精度和准确度。

对于感官颜色的判定，最终本设计可以成功过识别出红、绿、蓝、黄、红、紫、黑、白等颜色并成功在 LCD 上显示，并且具有很高的分辨力和色度容差。由于算法是我们不断归纳总结的，所以判别的细度还有很大的提升空间。

5.3 本章小结

本章对设计系统进行了实际检验，验证了系统的可行性和分辨力。最终结果为不尽可以在 LCD 上显示输出 EGB 和 HSV 值，还能对红、绿、蓝、黄、红、紫、黑、白等基本颜色做出准确的感官判定并以汉字的形式输出，从而很好的实现了本设计预期实现的功能。

6 系统设计优势和不足

6.1 优势及实现原因分析

本次设计以极其低廉的成本和较好的便携性实现了颜色识别系统的成功搭建，具有测定颜色准确性高的优点，同时 LCD 输出结果由于加了感官化判定颜色的输出，因此很人性化。

这样的结果基于我们对以下影响因素的成功解决：

- 待测环境光源的影响；

  解决方案：①模块本身集成了 4 个 LED 灯；②每次测定开始前进行白平衡，同时在测试过程中可以随时通过按键 0 进行重新的白平衡；③将模块进行了封闭化处理.

- 测量距离的影响；

  解决方案：将模块封装在小盒子里，从而保证了测量距离的稳定不变性。

6.2 有待提高的问题

- 由于本系统对距离很敏感，故测定平面物体的准确性较高，而对表面曲度很大的物体的测量仍有一定的误差；
- 本系统对反射式不透明物体的颜色测定精度横高，但不能测定透明物体的透射光颜色；
- 由 RGB 值转换成的感官颜色算法有待进一步完善以达到更好的分辨力。

7 体会与收获

本次大项目设计集合了我们在微机课上学得的单片机基础和光电检测技术课上学得的器件特性和测试原理，从器件的评估选择、硬件系统的搭建、软件程序的编写到最后的调试改进，整个过程都让我们收获颇丰。

一方面我们将检测中学得的检测原理、外界干扰分析和元件性能评估等知识应用在了实际中，从而加深了对理论的理解，更加清晰地了解了实际系统的构架；另一方面通过编写程序和不断地改进算法，我们提升了自己的软件编程能力和问题解决能力。这些都对以后的学习和工作有很大帮助。

8 参考文献

[1] 张智博, 王艳, 殷天明. 基于 TCS230 的颜色识别系统设计[J]. 2010 年西南三省一市自动化与仪器仪表学术年会论文集, 2010.

[2] 胡建民. 颜色传感器 TCS230 及颜色识别电路[J]. 单片机与嵌入式系统应用, 2006, 4: 40-41.

[3] BINGQIONG P, JIANYI M. 基于 TCS230 传感器的高精度颜色识别系统设计[J]. 2008.

附录 1 程序代码

```
/*实现功能:对颜色传感器输出 RGB、HSV 并判别出感官颜色*/
#include <c8051f020.h> // SFR 声明
#include <stdio.h>
#include <intrins.h>
#include <string.h>
#include <math.h>
#define      uchar unsigned char
#define      uint   unsignedint
/*******端口设置********/
#define   RS   0x80   // 选择（或 D/I) RS=0 命令 RS＝1 数据
#define   RW   0x40     // 读写（或 R/W) RW=0 写入 RW＝1 读出
#define   E    0x20   // 配合 RS、RW 对寄存器操作 E=1 读出 E 下降沿写入
#define   RST 0x10
#define   PSB 0x08   // ST2920 接口方式选择 PSB=0 串行 PSB=1 4b/8b 并行
#define   set(cbit) P6|=cbit
#define   clr(cbit) P6&=~cbit
//=================颜色传感模块连接====================
sbit      tcs230_s2=P3^0;//TCS230 S2 接单片机 P3.0
sbit      tcs230_s3=P3^1;//TCS230 S3 接单片机 P3.1
sbit      tcs230_en=P3^2; //TCS230 EN(E0)接 P3.2
//**************函数声明*************************************
void     SYSCLK_Init (void);
void     PORT_Init (void);
void     LCD_INIT(void); //LCD 初始
void     Clr_DDRAM(void);//清屏程序
void      LCD_WR_DATA(unsigned char temp);//LCD 模块写数据
void      LCD_WR_COMMAND(unsigned char command); //LCD 模块写指令
void      LCD_READ_BF (void);//读 LCD 模块的忙标
void      Disp_String(unsigned char x,unsigned char y,unsigned char
*str,unsignedchar str_length);//写字符串程序
void      Delay1us(unsigned char us);//1uS 基准延时程序
void      delay_ms(long intms);//1mS 基准延时程序
void      hsv(uint r, uint g, uint b);//RGB 转 HSV 程序
void      baipingheng();//白平衡子程序
```

```
  void       celiang();//实际颜色程序
  void       colordefine0(uintrr,uintgg,uint bb);//颜色判别程序
  voidchushihua(void);
  voidKey_Read(unsigned char byte);
  uintxdataryz,gyz,byz;//分别定义红色因子绿色因子蓝色因子
  uintrb,gb,bb;//RGB 值
  uinth,s,v;
  uchar      tab1[]={'0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'};
  //**********************主程序*****************************
  unsigned char LineR[]={'R',0x30,0x30,0x30};
  unsigned char LineG[]={'G',0x30,0x30,0x30};
  unsigned char LineB[]={'B',0x30,0x30,0x30};
  ucharLineHSV[]={'H',0x30,0x30,0x30,' ',' ','S',0x30,0x30,0x30,' ',' '
','V',0x30,0x30,0x30};
  ucharLinecolor[]={0xD1,0xD5,0xC9,0xAB,0xCE,0xAA,0xA1,0xC3,0xA1,0xAB,0x
A1,0xAB,0xA1,0xAB };
  //颜色为:
  ucharLinecolor0[]={0xD1,0xD5,0xC9,0xAB,0xCE,0xAA,0xA1,0xC3,0xA1,0xAB,0
xA1,0xAB,0xA1,0xAB};
  ucharjiemian0[]={0xA1,0xEE,0xD1,0xD5,0xC9,0xAB,0xCA,0xB6,0xB1,0xF0,0xA
1,0xEE};
  ucharjiemian1[]={0x2A,0x2A,0x2A,0x2A,0x2A,0x2A,0x2A,0x2A,0x2A,0x2A,0x2
A,0x2A,0x2A,0x2A,0x2A,0x2A};
  uchar jiemian2[]={0xB2,0xE2,0xC1,0xBF,0xC7,0xE8,0xB0,0xB4,0x31};
  ucharjiemian3[]={0xC2,0xED,0xB3,0xAC,0xE8,0xAF,0xA1,0xAB,0xD5,0xC5,0xB
D,0xF2};
  ucharkey_value=0xFF;
  unsigned char KeyTab[16] ={ 0xEE,0xED,0xEB,0xE7,
                              0xDE,0xDD,0xDB,0xD7,
                              0xBE,0xBD,0xBB,0xB7,
                              0x7E,0x7D,0x7B,0x77
  };
  unsigned char KeyCode=0xFF;
  bitkey_flag=0;
  uchar key;
```

```
main()
{
    WDTCN=0xDE;
    WDTCN=0xAD;
    TMOD=0x51;//设定 T0 以工作方式 1 定时 10 毫秒
SYSCLK_Init();
PORT_Init();
baipingheng();
    LCD_INIT();//LCD 初始
    EA = EX0 = 1; IT0 = 1; //  允许外部中断 0  边沿触发
chushihua();
while(key_flag==0);
Clr_DDRAM();

while(1)
    {
Key_Read(KeyCode);
if(key==0x00)
        {
         key_flag=0;
         Disp_String(0,0,"   白平衡校准      ",16);
         Disp_String(0,1,"                   ",16);
         Disp_String(0,2,"按其余任意键开始",16);
         Disp_String(0,3,"                   ",16);
         while(key_flag==0);
         baipingheng();
         Disp_String(0,2,"白平衡校准完成   ",16);
         Disp_String(0,3,"      请开始测量   ",16);
        }

    if(key==0x01)
        {
    celiang();//颜色测试
hsv(rb,gb,bb);//转换 HSV
```

```
            colordefine0(rb,gb,bb);//颜色判别
        LineR[1]=rb/100+0x30;
        LineR[2]=rb/10%10+0x30;
        LineR[3]=rb%10+0x30;
        LineG[1]=gb/100+0x30;
        LineG[2]=gb/10%10+0x30;
        LineG[3]=gb%10+0x30;
        LineB[1]=bb/100+0x30;
        LineB[2]=bb/10%10+0x30;
        LineB[3]=bb%10+0x30;
    LineHSV[1]=h/100+0x30;
    LineHSV[2]=h/10%10+0x30;
    LineHSV[3]=h%10+0x30;
            LineHSV[7]=s/1000+0x30;
    LineHSV[8]=s%1000/100+0x30;
    LineHSV[9]=s%100/10+0x30;
    LineHSV[10]=s%10+0x30;
    LineHSV[13]=v/100+0x30;
    LineHSV[14]=v/10%10+0x30;
    LineHSV[15]=v%10+0x30;
            Disp_String(0,0,"       颜色识别       ",16);
            Disp_String(0,1,LineR,4);
        Disp_String(6,1,LineG,4);
        Disp_String(12,1,LineB,4);
        Disp_String(0,2,LineHSV,16);
    Disp_String(0,3,Linecolor0,14);
                    }
          }
    }
    //-----------------------------------------------------------------------
    // Int0 中断服务程序
    //-----------------------------------------------------------------------
    //  得到 KeyCode 键值 0～F,键盘无效则返回-1;
    void INT0_ISR (void) interrupt 0 using 3
    {
```

```
    key_flag=1;
    KeyCode = P7; //  读键盘高 4 位；
    if(KeyCode != P7) {
        KeyCode=0xFF;
        return;
    }
    P7 ^= 0xFF;
    Delay1us(20);
    KeyCode |= P7;
    P7 ^= 0xFF;
}
//-----------------------------------------------------------------------------
//  键值读取程序
//-----------------------------------------------------------------------------
voidKey_Read(unsigned char byte)
{
    unsigned char i;

    for(i=0;i<16;i++)
        if(KeyTab[i]==byte)
        {
        key=i;
        break;
          }
}
voidchushihua(void)
{
inti;
for(i=0;i<0x010f;i++)
Disp_String(0,1,"欢迎",16);
delay_ms(50000);
for(i=0;i<0x010f;i++)
Disp_String(0,1,"   欢迎   ",16);
delay_ms(50000);
for(i=0;i<0x010f;i++)
```

```
Disp_String(0,1,"       欢迎       ",16);
delay_ms(50000);
for(i=0;i<0x010f;i++)
Disp_String(0,1,"          欢迎          ",16);
delay_ms(50000);
Disp_String(0,2,"       WELCOME       ",16);
delay_ms(100000);

Disp_String(2,0,jiemian0,12);
Disp_String(0,1,jiemian1,16);
Disp_String(0,2,"校准[0]  测量[1]",15);
Disp_String(3,3,"By Zhang & Ma",13);
}

voidSYSCLK_Init (void)
{
inti; //  延时计数器
      OSCXCN = 0x67; //  开启外部振荡器 11.0592MHz 晶体
      for (i=0; i< 256; i++) ; //  等待振荡器启振
while (!(OSCXCN & 0x80)) ; //  等待晶体振荡器稳定
      OSCICN = 0x88; //  选择外部振荡器为系统时钟源并允许丢失时钟检测器
}

voidPORT_Init (void)
{
      XBR0 = 0xa7; //  使能 UART0 等
      XBR1 = 0x1f;
      XBR2 = 0x44; //  使能数据交叉开关和弱上拉
    P3MDOUT&=~0x01; // P3.0 漏开  18B201.c 6/13
    P7 = 0xF0;
}

//*********************************************************
//测量子程序
voidceliang()
```

```
{
    //*********求 R 值*************************************
      TH0=(65536-10000)/256;
      TL0=(65536-10000)%256;
TH1=0;
TL1=0;
tcs230_s2=0;
      tcs230_s3=0;//选择红色滤光器
      tcs230_en=0;
TR0=1;//10 毫秒开始计时
TR1=1;//开始计数
while(TF0==0);//等待定时器溢出
TF0=0;//清楚定时器 0 溢出标志
TR0=0;//关闭定时 0
TR1=0;
rb=(unsigned long)(TH1*256+TL1)*255/ryz;
if(rb>255)rb=255;//判断 RGB 值是否合法*/
//***********求 B 值*************************************
TH0=(65536-10000)/256;
     TL0=(65536-10000)%256;
TH1=0;
TL1=0;
tcs230_s2=0;
tcs230_s3=1;//选择蓝色滤光器
TR0=1;//10 毫秒开始计时
TR1=1;//开始计数
while(TF0==0);//等待定时器溢出          0
TF0=0;//清楚定时器 0 溢出标志
TR0=0;//关闭定时 0
TR1=0;
bb=(unsigned long)(TH1*256+TL1)*255/byz;
if(bb>255)bb=255;//判断 RGB 值是否合法
//***********求 G 值*************************************
TH0=(65536-10000)/256;
     TL0=(65536-10000)%256;
```

```
TH1=0;
        TL1=0;
        tcs230_s2=1;
        tcs230_s3=1;//选择绿色滤光器
        TR0=1;//10 毫秒开始计时
        TR1=1;//开始计数
        while(TF0==0);//等待定时器溢出
        TF0=0;//清楚定时器 0 溢出标志
        TR0=0;//关闭定时 0
        TR1=0;
        tcs230_en=1;
gb=(unsigned long)(TH1*256+TL1)*255/gyz;
        if(gb>255)gb=255;//判断 RGB 值是否合法  */
}
//************************************************************
//白平衡子程序
voidbaipingheng()
{
//***************求取红色因子************************
        TH0=(65536-10000)/256;
        TL0=(65536-10000)%256;
        TH1=0;
        TL1=0;
        tcs230_s2=0;
        tcs230_s3=0;//选择红色滤光器
        tcs230_en=0;
        TR0=1;//10 毫秒开始计时
        TR1=1;//开始计数
        while(TF0==0);//等待定时器溢出
        TF0=0;//清楚定时器 0 溢出标志
        TR0=0;//关闭定时 0
        TR1=0;
ryz=TH1*256+TL1;//其实这里的比例因子应该为 255/(TH1*256+TL1)
  //***************求取蓝色因子************************
        TH0=(65536-10000)/256;
```

```
    TL0=(65536-10000)%256;
        TH1=0;
        TL1=0;
        tcs230_s2=0;
        tcs230_s3=1;//选择蓝色滤光器
        TR0=1;//10 毫秒开始计时
        TR1=1;//开始计数
        while(TF0==0);//等待定时器溢出
        TF0=0;//清楚定时器 0 溢出标志
        TR0=0;//关闭定时 0
        TR1=0;
  byz=TH1*256+TL1;//其实这里的比例因子应该为 255/(TH1*256+TL1)
      //***************求绿红色因子************************
        TH0=(65536-10000)/256;
    TL0=(65536-10000)%256;
        TH1=0;
        TL1=0;
        tcs230_s2=1;
        tcs230_s3=1;//选择绿色滤光器
        TR0=1;//10 毫秒开始计时
        TR1=1;//开始计数
        while(TF0==0);//等待定时器溢出
        TF0=0;//清楚定时器 0 溢出标志
        TR0=0;//关闭定时 0
        TR1=0;
  gyz=TH1*256+TL1;//其实这里的比例因子应该为 255/(TH1*256+TL1)
      tcs230_en=1;
  }
  /*============================================================
=
   RGB 转 HSV 颜色空间
  ============================================================*
/
  voidhsv(uint r, uint g, uint b)
  {
```

```
uint max,min,temp1,temp2,sub;
temp1 = r>g?r:g;
max=temp1>b?temp1:b;
temp2 = r<g?r:g;
max=temp2<b?temp2:b;

sub=max-min;
if(max==r)
h=(g-b)/sub;
else if(max==g)
h=2+(b-r)/sub;
else
h=4+(r-g)/sub;
h=h*60;
if(h<0)
h=h+360;

v=max;
s=sub/max;
}
/*============================================================
=
颜色判别程序
============================================================*/
void       colordefine0(uintrr,uintgg,uint bb)
{
int r=rr,g=gg,b=bb;
if((r>185)&&(g>185)&&(b>185))
{//白色
Linecolor0[8]= 0xB0;
Linecolor0[9]= 0xD7;
Linecolor0[10]=0xC9;
Linecolor0[11]=0xAB;
}
else if((r<25)&&(g<25)&&(b<25)&&((r-b)<3))
```

```
{//黑色
Linecolor0[8]= 0xBA;
Linecolor0[9]= 0xDA;
Linecolor0[10]=0xC9;
Linecolor0[11]=0xAB;
}
else if (r>(b+70)&&r>(g+70)&&abs(g-b)<30)
{   //红色
Linecolor0[8]=0xBA;
Linecolor0[9]=0xEC;
Linecolor0[10]=0xC9;
Linecolor0[11]=0xAB;
}
else if(b>(r+10) &&b>g)
{ //蓝色
Linecolor0[8]=0xC0;
Linecolor0[9]=0xB6;
Linecolor0[10]=0xC9;
Linecolor0[11]=0xAB;
}
else if(r>150 && g>80&& (g-b)>40)
{//黄色
Linecolor0[8]=0xBB;
Linecolor0[9]=0xC6;
Linecolor0[10]=0xC9;
Linecolor0[11]=0xAB;
}
else if(b>g&&r>(g+10))
{ //紫色
Linecolor0[8]=0xD7;
Linecolor0[9]=0xCF;
Linecolor0[10]=0xC9;
Linecolor0[11]=0xAB;
}
else if ((r-g)<20 &&g>(b+10))
```

```
{    //绿色
Linecolor0[8]=0xC2;
Linecolor0[9]=0xCC;
Linecolor0[10]=0xC9;
Linecolor0[11]=0xAB;
}
else
{
Linecolor0[8]='?';
Linecolor0[9]='?';
Linecolor0[10]=0xC9;
Linecolor0[11]=0xAB;
}
}
/*============================================================
=
  LCM 初始化
 ============================================================*
/
void LCD_INIT(void)
{
    set(PSB);     //选择并口方式
    set(RST);     //复位
    clr(RST);
    set(RST);
    LCD_WR_COMMAND(0x30);//8 位并行
    LCD_WR_COMMAND(0x01);//清除显示 DDRAM
    LCD_WR_COMMAND(0x06);//光标右移 AC 加 1
    LCD_WR_COMMAND(0x0C);//整体显示 ON，光标显示关，光标位置不反白
闪烁
    LCD_WR_COMMAND(0x02);//地址归零
    Clr_DDRAM();
}
/*============================================================
=
```

```
  读忙标志
  ================================================================*
/
  void LCD_READ_BF (void)
  {
      unsigned char temp;
      clr(RS);
      set(RW);
      while(1)
      {
            P5=0xFF;
            set(E);
            Delay1us(30);
            temp=P5;
            clr(E);
            if((temp&0x80)==0) break; //忙标志为 0，方可接收新的指令
      }
  }
  /*================================================================
=  写数据到指令寄存器
  ================================================================*
/
  void LCD_WR_COMMAND(unsigned char command)

  {
      LCD_READ_BF();
      clr(RS);
      clr(RW);
      P5=command;
      set(E);
      Delay1us(3);
      clr(E);
  }
  /*================================================================
=
```

```
写数据到数据寄存器
============================================================*/
void LCD_WR_DATA(unsigned char temp)
{
    LCD_READ_BF();
    set(RS);
    clr(RW);
    Delay1us(1);
    P5=temp;
    set(E);
    Delay1us(3);
    clr(E);
}
/*============================================================
清屏
============================================================*
/
voidClr_DDRAM(void)
{
        LCD_WR_COMMAND(0x01);//清除显示 DDRAM
}
/*============================================================
=
显示字符串，x 为横坐标，y 为纵坐标
============================================================*
/
void Disp_String(unsigned char x,unsigned char y,unsigned char *str,unsigned char
str_length) //为了与反白相对应，x 位置以字符为单位
{
unsigned char i;
    switch(y){
        case 0: LCD_WR_COMMAND(0x80+x/2); break;
        case 1: LCD_WR_COMMAND(0x90+x/2); break;
        case 2: LCD_WR_COMMAND(0x88+x/2); break;
        case 3: LCD_WR_COMMAND(0x98+x/2);
```

```
    }
    if(x%2) LCD_WR_DATA(' ');
    for(i=0;i<str_length;i++) LCD_WR_DATA(*str++);
}
/*============================================================
=
延时 1us
============================================================*
/
void Delay1us(unsigned char us) using 3
{
    for(;us;us--)
        _nop_(), _nop_(), _nop_();
}
/*============================================================
=
延时 1ms
============================================================*/
voiddelay_ms(long intms)
{
longinti;
    for(i=0; i<ms; i++) {
            Delay1us(250);
            Delay1us(250);
            Delay1us(250);
            Delay1us(250);
    }
}
```