基于单片机的　LED旋转字的设计

摘　要：　如今的生活中，不管是银行、商场，还是饭馆、酒店，　　LED显示屏可谓是无处不在，广泛应用于我们的生产生活中。然而，传统　　LED 显示屏的不足也在一天天显露出来。一个就是传统　LED显示屏组成模块的器件数量较多，成本略高。还有一个就是传统　LED 显示屏显示在一个平面上，有一定视角限制。为了解决这两个不足，通过单片机控制被固定在电机上的发光二级管的亮灭，同时利用电机带动二极管旋转，并配合传感器触发的外部中断，从而稳定的显示出文字和图案。

关键词：LED显示屏　；　单片机；　视觉暂留；　旋转

LED rotating word design based on single chip microcomputer

Abstract：Today's life,whether banks,shopping malls,restaurants,hotels,LED display is ubiquitous,widely used in our life and production. However,traditional LED display also day by day out.One is the traditional LED display module device number is more,the cost is slightly higher. There is a traditional LED display on a plane,a certain angle limit.In order to solve the two issues,Through the control of the microcontroller ,is fixed on the motor emission level two tube light out,at the same time,the use of motor driven diode rotation,and cooperate with external sensor to trigger the interrupt,thus the stable display text and pattern.

Keywords：LED display; microcontroller; persistence of vision; rotation

目录

第 1 章绪论

1.1 LED 显示屏背景

LED 就是 light emitting diode ，发光二极管的英文缩写，简称 LED[1]。其显示屏是由 LED组成的点阵模块[2]。LED显示屏在生产生活中随处可见，银行、饭店的各种指示窗口，大街上的各种广告和活动的宣传窗口，工厂里各种仪器的指示板等等，都会用到 LED显示屏。作为一种信息传播的媒介载体，LED显示屏受到各领域的欢迎， 被广泛应用于各种服务和宣传场所， 尤其是在大城市里， 随着人口的增加和各种商业活动的宣传， LED显示屏的使用范围还将进一步扩大。目前， LED显示屏的推广率逐年增高，市场发展前景广阔，商业价值巨大。

1.2 国内外研究现状

早期的 LED光度低、色彩少，仅仅用在指示灯和显示板上。而随着电子信息技术的发展，各行各业对显示器材的要求， LED技术得到了快速的发展。现在的 LED不仅光度提高了，色彩也变得丰富了，应用的范围也更广泛了。而且，中小功率超高亮度 LED也已经出现，并且正在以惊人的速度被推广和应用。

在显示方面， LED已经广泛应用于各种产品的状态性能显示， 例如家用电器、工业设备等，越来越多的智能器件的显示屏都可以看到 LED的身影。随着其相关技术的快速发展，国外对 LED的使用越来越多。在改革开放后，随着经济的快速增长，工业化和信息化程度越来越高，人们的需求越来越大，LED显示屏在国内得到了快速发展， 应用范围和规模也在不断地扩大。 由此可以推断， LED 产业的未来，将根据用户的需求生产出更多种的 LED产品，多元化的 LED产品将出现在我们生活的每个角落。而且还将建立应用形式独立的应用领域， LED划分时代便出现。所以 LED有着更为美好的前景。

1.3 论文的设计目标和结构安排

毕业设计是学生在学生时代的最后一项作业，不仅检查学生是否掌握了本专业的知识，而且还培养了学生解决实际问题的能力。在本次设计中，我不仅温习了以前学习过的知识，而且还在查阅资料时学习了一些新的知识。

将理论和实际有效的结合起来，做到所学即为所用，培养了我的动手操作能力，对我以后的工作生活具有重要意义。

本次设计是以 STC89C52单片机为核心芯片，利用电机带动 16 个 LED旋转，随着 LED在不同时间下的显示状态，显示出所需要的文字。

本设计在绪论里主要介绍了有关 LED 显示屏的使用和发展现状。第二部分则对本次设计使用的核心芯片 STC89C52和旋转 LED显示屏做一个大致的了解。第三部分讲述了系统的硬件设计，主要介绍了系统的显示原理和对模块电路的剖析。第四部分介绍了系统模块程序运行流程。第五部分是系统调试。

第 2 章 STC89C52单片机和旋转　LED显示屏

2.1 单片机简介及应用

单片机是一种微型计算机系统，利用集成电路技术开发的一种电路芯片，功能多而且强大，被越来越多的运用到人们的生活中。由于人们日益增长的生活水平，科学技术的更新和发展，人们对电子产品的需求也越来越大，这在一定程度上促进了单片机技术的发展。而单片机的价格并没有水涨船高，反而跌得很厉害，　10 美元就可以购买到一个最高端的单片机了，而普通型号的单片机只需要　1 美元。

随着单片机技术和性能的不断发展和提高，不管是在最初的工业控制领域，还是现在的家用电器和医疗器械领域，甚至航空航天等领域，单片机都发挥着不可替代的作用，对我们的生产生活产生了重要影响。

2.2 STC89C52单片机的结构和管脚介绍

STC89C52 是一种耗能低、效率高、处理能力强大的　CMOS 8 位微控制器，它的内核仍然采用经典的　MCS-51，但做了很大的改进，增加了众多功能。自带 4K 字节的可编程可擦除的只读程序存储（　EPROM）空间和 512B 字节的随机存取数据存储（RAM）空间，还有功能强大的 8 位 CPU和可编程 Flash 单元等。还能够使用串口下载，简单方便，可应用于各种控制领域。图 2-1 是 STC89C52 单片机的基本功能方块图。

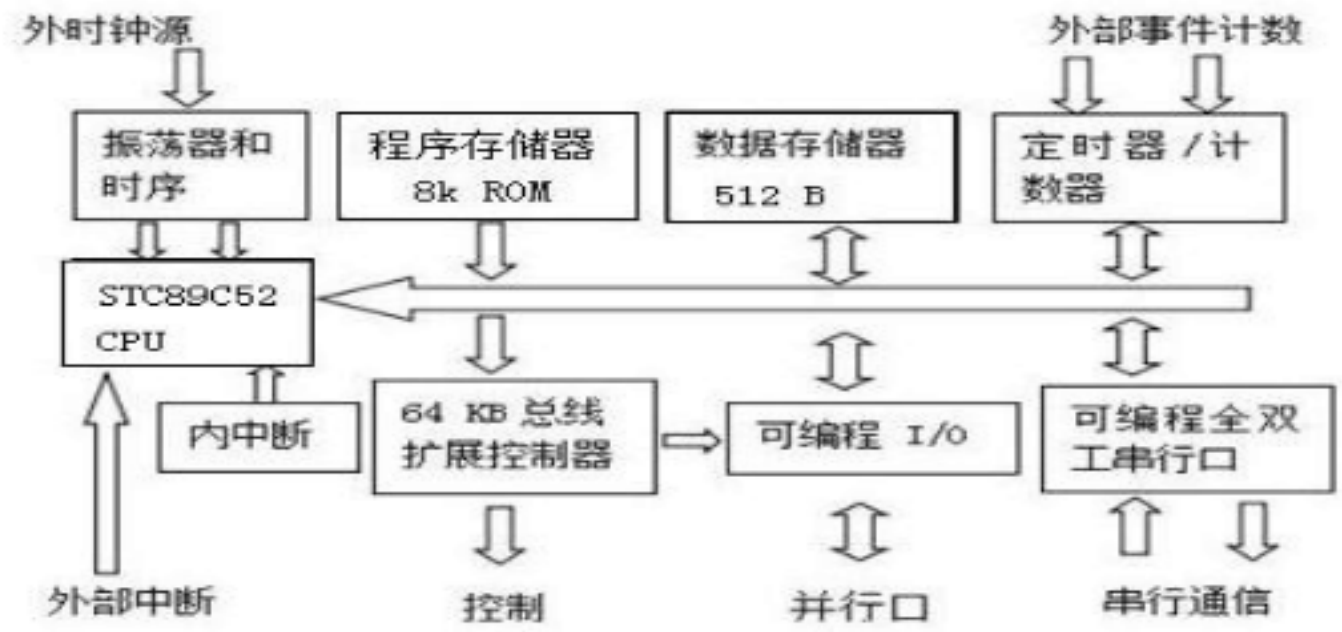


图 2-1 STC89C52 功能图

由于引脚只有　40 个，无法一一容纳其众多的功能，所以就把其中一些引脚开发了多种功能。图　2-2 是典型的 STC89C52单片机的引脚图。

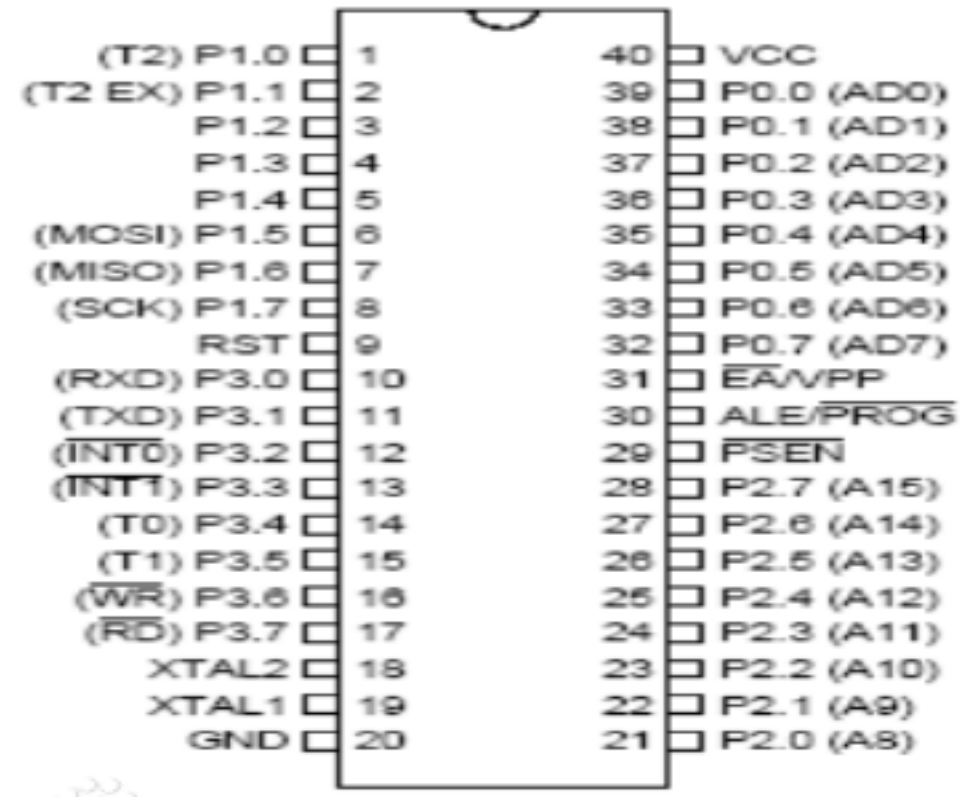


图 2-2 STC89C52 单片机管脚

VCC ：芯片的 40 引脚，是连接电源电压端口。

GND ：芯片的 29 引脚，是接地端口。

P0 口:P0 口指的是 P0.0-P0.7 端口，是芯片的 32-39 引脚，是一个 8 位漏极开路双向的 I/O 口[3]。当写入“ 1 ”时，端口可以作为高阻抗输入。另外，当把 P0口用作为数据 / 地址的第八位时，还可以访问外部 RAM和 ROM当 FLASH 编程时，此端口可以收到指令字节，当 Flash 核实程序时，指令字节又会被输出。需要了解的是， P0口在验证程序时，需要外接上拉电阻。

P1 口：P1口包括 P1.0-P1.7 端口，作为芯片的 1-8 引脚封装。 P1 口、P2口和 P3口一样，都是一个 8 位双向 I/O 口，都是由内部提供上拉电阻的[4]。在 P1 口端口写入 1 时，被上拉电阻拉为高电平，这时候端口可以用作输入。由于受到内部上拉电阻影响，端口写 0 时，被外部拉低，此时， P1口输出电流。

P2 口：P2口是指芯片 P2.0-P2.7 端口，包括 21-28 引脚。由于内部上拉电阻的影响，在端口写入 1 时，端口可以拉到高电平，然后 P2口可以用来作为一个输入端口。当端口写入 0 时，会被拉为低电平，这时端口将输出电流。

P3口：P3口包括 P3.0-P3.7 口，是 10-17 引脚。当 P3口被写入 1 后，可以作为输入端口使用，而其内部上拉电阻会将其拉为高电平，此时它的外部引脚被拉为低电平，端口将会输出电流。除了用于 I/O 口外， P3口还可做其他用处，如图 2-3。

P3 口管脚	其他功能
P3.0 RXD	串行数据输入口
P3.1 TXD	串行数据输出口
P3.2 INT0	外部中断 0
P3.3 INT1	外部中断 1
P3.4 T0	定时器 0 外部输入
P3.5 T1	定时器 1 外部输入
P3.6 /WR	外部数据存储器写通
P3.7 /RD	外部数据存储器读通

图 2-3 P3 口管脚的其他功能

RST: RST是芯片的 19 引脚，复位控制线。负责对单片机进行复位重置操作。

ALE/PROG ：此端口是芯片的 30 引脚， ALE地址锁存允许端， PROG编程脉冲输入端。

PSEN ：这个端口是 29 引脚，作为存储器读选通信号接口。

EA/VPP: 是芯片的 31 引脚，输入信号。

XTAL1: 该端口是芯片的 19 引脚，作为时钟电路的输入端 [5]。

XTAL2: 该端口是芯片的 18 引脚，作为时钟电路的输出端 [5]。

2.3 LED 显示屏特点

随着电子技术的快速发展， LED也被越来越广泛的应用。 LED显示屏有着众多的优点：

一是使用时限超长。业界对 LED的使用寿命检测的平均值达到 10 万小时，在未来，这一数值将有望达到 25 万小时。

二是众多可选择的色彩。 随着科学技术的不断发展， LED的颜色库也被丰富起来，从最初的红色发展到现在红、黄、绿、蓝等各种颜色，基本能够满足市场对 LED色彩的要求。

三是稳定可靠。 LED能够在其寿命期内，高效且稳定地进行工作。

四是安全性高。电压在 36V以下对人体都是安全的，不会对人体造成损

害。而 LED的工作电压是 6V-24V，工作电流也只有 10mA-20mA，属于弱电级工作器件。其电气安全性能在业界是有口皆碑的。

五是高效率而且节能环保。在亮度相同的条件下，普通白炽灯的耗电量是 LED的 10 倍。另外， LED也更加的环保，不像其他显示媒介存在有害金属污染等问题，也更符合社会对环保的要求和提倡。

六是体积小，便捷，灵活性好。单粒 LED的体积很小，大概只有 3-5 平方毫米那么大。而且 LED不仅可以低压供电，还可以高压供电。大大方便了工程应用，在工程应用领域得到了青睐。

七是方便控制。就目前的技术来说，已经能够对 LED的亮度、灰度、动态显示、分布等进行自由地控制，这在业界是无与伦比的。

八是优良的抗震抗干扰性能，可靠性高。现阶段市场上其它类型的电光源产品易碎、易坏且容易受干扰而失灵，而 LED却坚固、耐震、耐冲击且抗干扰，在使用过程中不易损坏，可靠性高。

九是较短的响应时间。 LED的响应时间是以毫秒为单位的， 在汽车刹车灯、相机闪光灯等应用上能够快速有效地接受指令并作出反应。

这些优势使得 LED显示屏在与传统媒介的较量中渐渐占了上风，传统的纸质媒介由于不环保、吸引力不足等原因已经正在被取消使用，而其他一些发光器件也因为可靠性低或者价格过高等原因转而使用性价比十足的 LED显示屏。

2.4 旋转 LED显示屏的简介

传统的 LED显示屏采用 16 行循环扫描显示文本和图像，扫描帧速率大于 60Hz，使得人眼感觉不画面闪动，因而看到的是一幅稳定的图像。然而，这样的显示屏在越来越多的应用中暴露了两个问题，第一，由于组成显示屏的 LED 模块所使用的器件数量多，导致制作显示屏的工作量增大 ，同时，成本也提高了。第二，由于 LED 模块被放置在一个平面上，观看显示屏的位置就会被局限于正面某个范围之内。如果应用在实际生活中，那么显示屏显示的信息就会受到影响，只有在范围以内的人能看到信息，而范围以外的人则看不到 LED所发布的信息，不利于信息的传播和推广。在图 2-4 中，我们看到 (a) (b) (c) (d) (e) 分别代表不同时刻 LED 的点亮状态， (f) 则为我们看到的完

整画面。在这种类型的 LED显示屏中，大多选用的是逐行扫描模式 [6]。

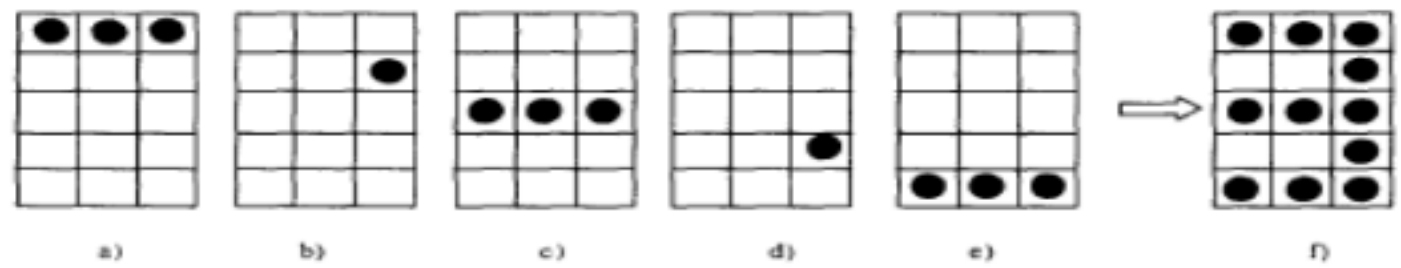


图 2-4 传统 LED显示屏的显示原理图

新的旋转柱式显示屏，使用旋转扫描方式，根据人的视觉暂留特性，由电动机带动模块 360 度旋转，以一定的速率刷新屏幕，再利用传感器控制二极管在各个位置的亮灭来呈现一幅完整的图像。这种扫描方式的显示器只有一列，不仅降低了成本，还扩大了可视范围，克服了传统 LED显示屏的不足，将 LED的发展推向了一个新的时期。图 2-5 中 a)b)c) 就是显示屏在不同时间的状态， d) 则是一幅完整画面。

图 2-5 旋转柱式显示屏的显示原理图

第 3 章　系统的硬件设计

3.1　显示原理

该系统的电路是由由电源供电电路、　LED显示电路、 电机驱动电路和数据处理电路构成的。利用人眼视觉暂留现象，利用电机驱动旋转扫描的方式，当每秒扫描　25 帧以上，人眼感觉不到闪烁能看到一个完整的画面。因此在电机设置时，转速需要大于每转　25 秒，周期可以设置为　0．04s/ 转，这样就能够看到 LED显示的完整画面。

3.2　系统供电

为了使系统能稳定的工作，需要要解决系统供电的问题。首先是电机供电，电机是采用　5V 电压供电的，由于电脑的　USB接口输出的电压也是　5V，所以只需要将电机的正负极用导线引出，然后接在　　USB 线上，通过电脑的　USB 接口供电，稳定方便。然后就是主板供电，由于设计中主板是固定在电机转子上的，并通过电机转动带动主板旋转，所以电机转子与主板是相对静止的，所以我们通过电机转子供电给主板。电路图如图　　3-8 所示。

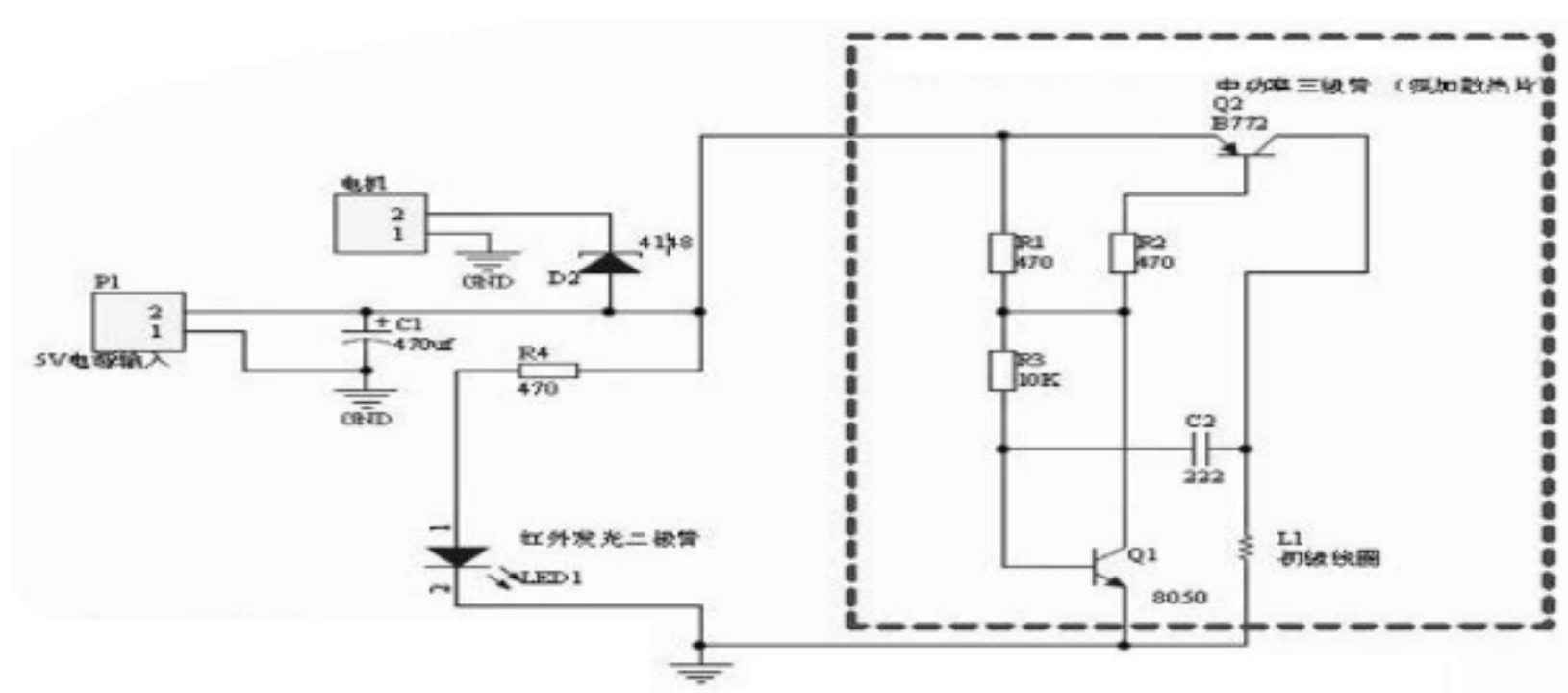


图 3-1　供电小板原理图

3.3　系统硬件框图和介绍

电源供电电路为整个系统提供稳定的电源，电机带动发光　　LED高速旋转，单片机系统负责信息采集、测速定位，它利用传感器测到的电机转速来确定LED的扫描时间，使二极管的亮灭配合电机的转速。　再利用人眼视觉暂留现象，我们就可以在　LED显示屏上看到一个完整的图像了。如图　　3-2 所示。

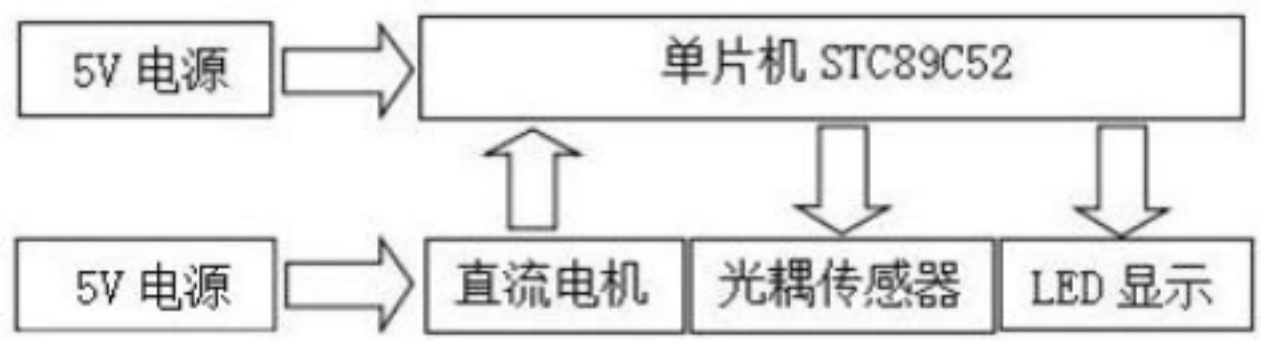


图 3-2　系统硬件框图

本设计的核心芯片采用　STC89C52单片机 LQFP-44封装，增加了 P4 口位寻址功能，性能更加强大，显示部分采用 0805LED，驱动选择高性能电机 fr370，系统主板如图 3-3 所示。为了更好地与计算机配合操作，将使用　TTL 串口下载器将内容发送到单片机。

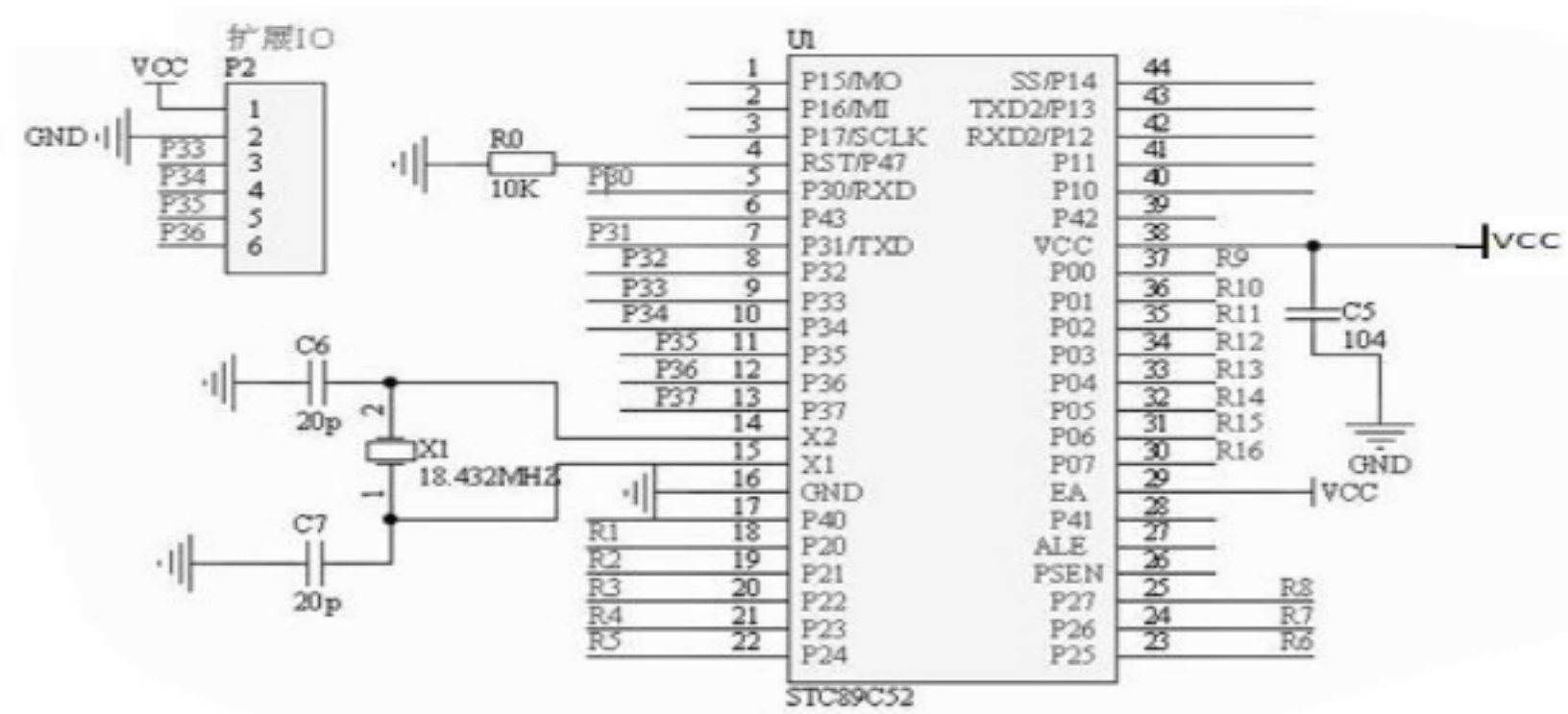


图 3-3　旋转 LED主板原理图

3.4　各模块的设计

3.4.1　传感器模块的设计

本设计的传感器模块采用　U型槽光耦合器设计，其响应速度快，易于单片机通信。如图 3-4 所示，当接通电源后，光敏三极管的集电极通过感应光线，输出高低电平，然后还会发送一个中断信号。

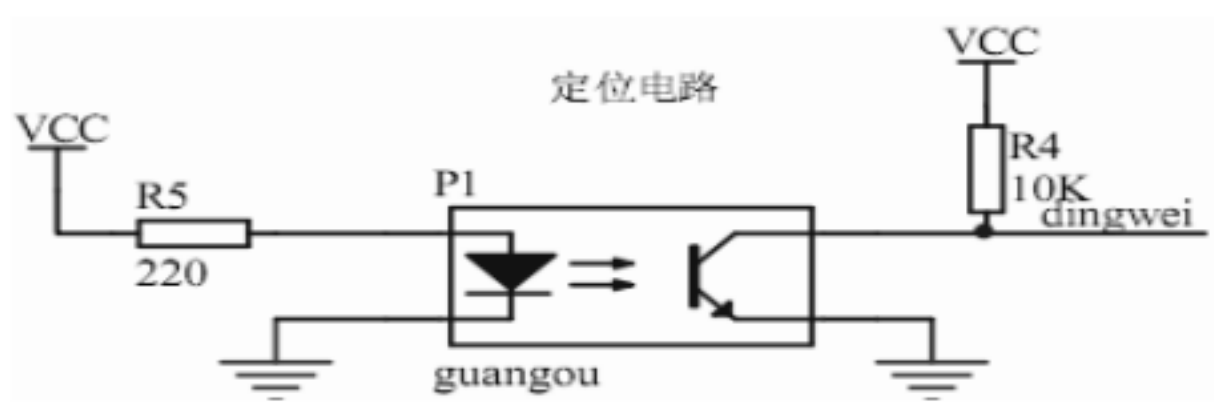


图 3-4 U 型槽光耦合器

3.4.2　显示模块的设计

由于 LED显示器的功耗低，接口控制简单，模块的接口信号又与操作指令广泛兼容，还能直接与单片机对接，可以很容易地实现各种操作，因此被广泛应用于各种测量及控制仪表仪器中。而要想在　　LED上显示汉字，首先要获得该汉字的点阵构成数据，然后将其写入显示存储器中，这是就可以显示我们所要的汉字了。旋转　LED显示通过同步控制发光二极管的位置和亮灭状态来实现图形显示[7]，可以 360 度观看。此次设计采用　16 个并联 LED，通过旋转扫描方式扫描。如图　3-5 所示。

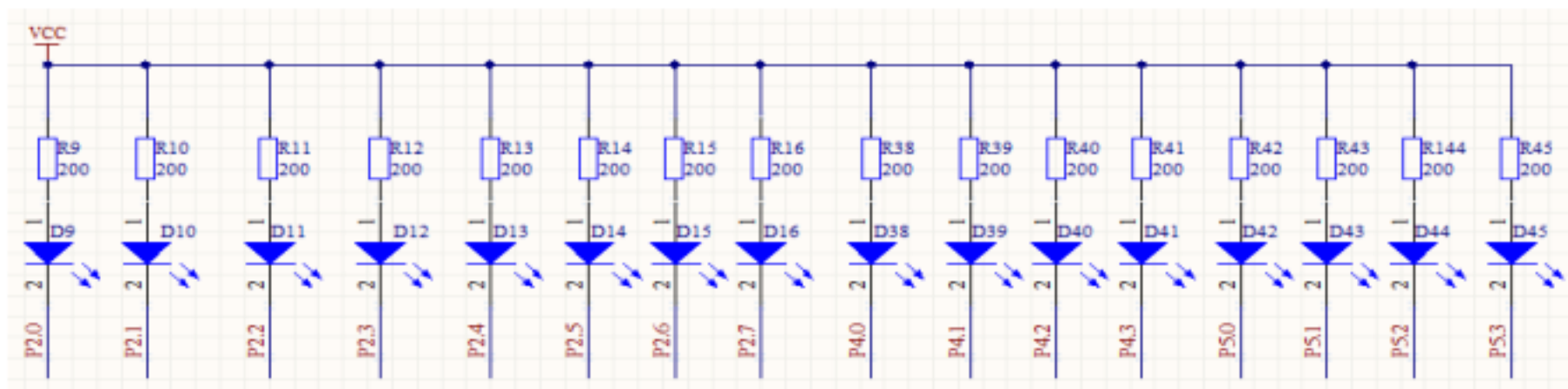


图 3-5 LED 模块

3.4.3　电源模块的设计

通过整流、滤波，普通的交流电压就可以得到直流电压。但是此时得到的直流电压还存在波纹，另外，由于交流电压的波动，及负载和温度的变化等影响，导致输出电压的纹波会变的更大，也就是说输出电流电压不稳定。为了获得稳定的输出电压，使负载得到稳定的输出电压，我们会在滤波电路与负载之间加入一个稳压电路。见图　3-6。

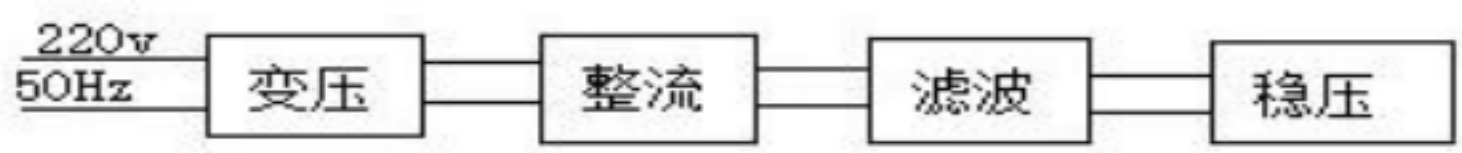


图 3-6　直流稳压电源的实现

3.5　电路设计

3.5.1　时钟电路

时钟电路是内置单片机电路，用于定时和计时。如图　　3-7 所示，把电容 C1、C2并联连接，分别接在　STC89C52的 XTALI 及 XTAL2端口，晶振　M则串连在电路中组成时钟电路。

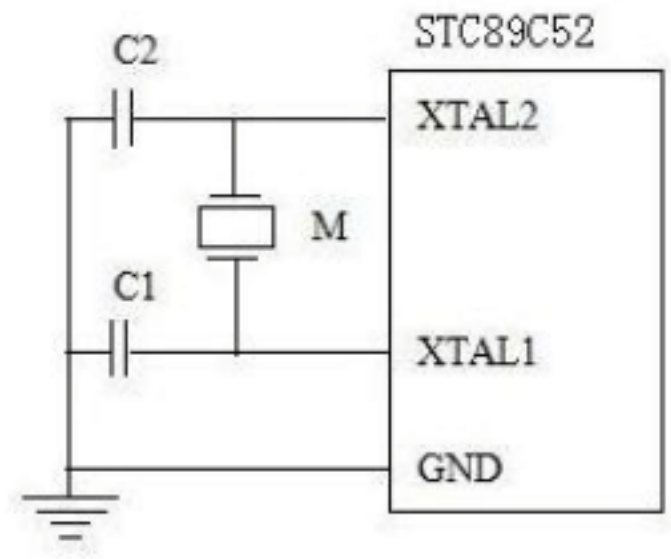


图 3-7 时钟电路

3.5.2 复位电路

复位电路的作用是在系统接通电源后，发出一个复位信号，并确定系统电源电压没有波动后再把发出的复位信号撤销。需要注意的是，由于刚接通电源时，各种误操作会导致电源抖动进而影响复位，所以在电源稳定后还需要过一段时间才可以撤销复位信号。本设计采用上电复位设计，如图 3-8 所示，电阻 R 和电容 C 串联，在系统通电后由于电容 C 需要充电，RST端由最初的高电平被慢慢拉为低电平，以实现复位操作[8]。

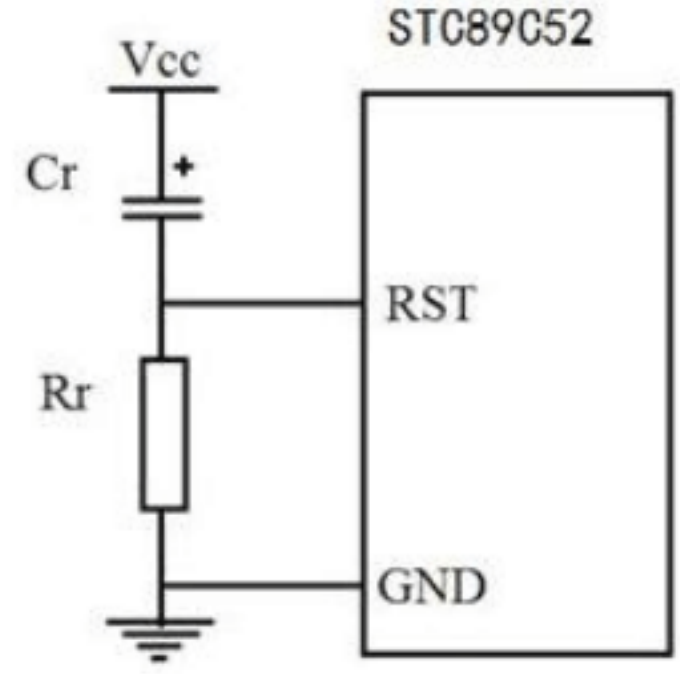


图 3-8 上电复位电路

3.5.3 驱动电路

驱动电路有两种输入方式。一种是并口输入方式，此方式占用的 I/O 口资源较多。另一种是串口输入方式，此方式占用的 I/O 口资源较少。所以选用串口输入方式。此时，进行行方向扫描的 P0 口作为 I/O 口使用，要加上拉电阻。

第 4 章　系统的软件设计

4.1　主程序流程图

主程序的功能是主板旋转后显示出设定的文字，如图　　4-1 所示。

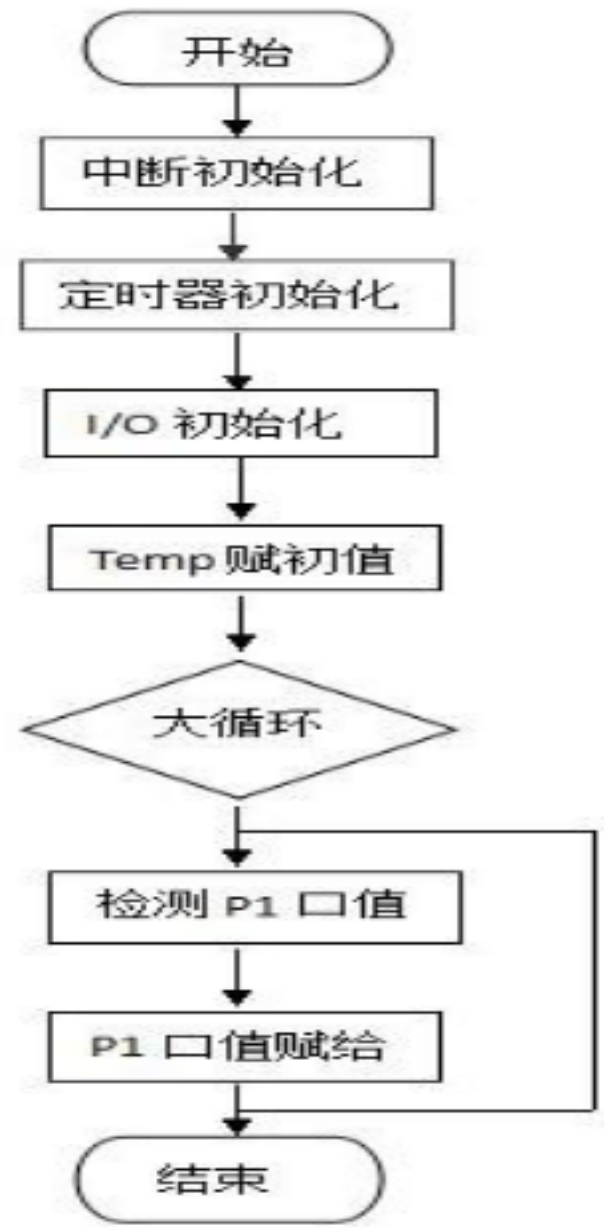


图 4-1 主程序流程图

4.2　各模块程序的流程图及功能说明

4.2.1 MAIN 函数部分

Main　函数又称主函数，　在程序主要负责对各个函数进行初始化设置，　　在没有其他中断来临时，　进入空指令死循环，　直到出现新的中断。　参看流程图 4-2。

4.2.2　外部中断 0 服务程序部分

外部中断 0 的作用主要是对显示更新、定位和对定时器　　T0 初值的校正。为了提供外部中断　0 的中断信号，我们将在硬件设计中添加一个光耦元件。在显示屏每旋转一周时就会就会产生一个中断信号给外部中断，中断服务程序此时就会把实际旋转一周的　T0 中断次数记录并保存下来，然后与我们的设置次数进行对比，以此数据来校正　　T0 的初始值。最后，为了更新显示和定位的，程序会把各个显示数据全部清零，跳出服务程序。具体流程如图　　4-3 所示。

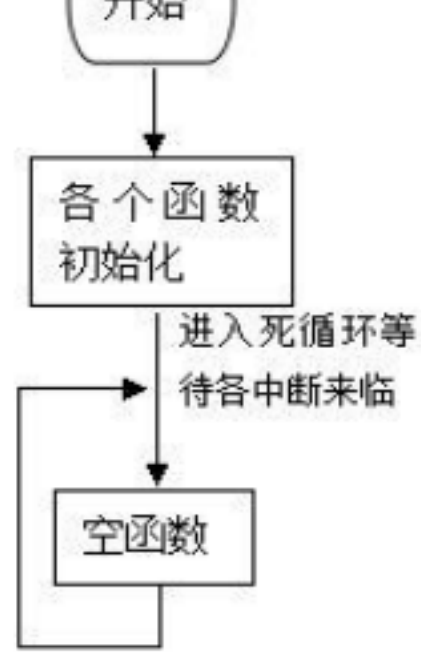


图 4-2 Main 函数程序流程图

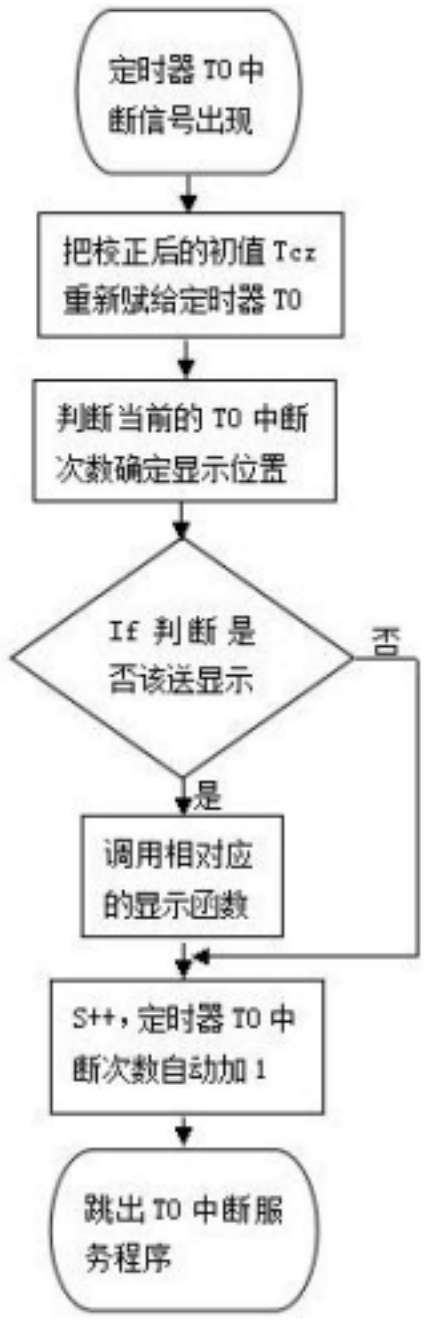
图 4-3 外部中断 0 程序流程图

4.2.3 定时器 T0 中断服务程序

在 T0 的中断服务程序中，在 T0 的中断信号出现时，系统会把初值赋给 T0，然后判断当前显示的位置，而后再决定是否发送该显示，在发送显示时，相应的显示函数也会启动，并且 T0 中断次数也会随之加一，最后跳出中断程序，中断结束。详细参看流程图 4-4。

4.2.4 定时器 T1 中断服务程序

由于 T0 的初始值是不固定的，所以我们不能再把 T0 当作时间运行的标准，否则时间会一时快一时慢，因此我们把 T1 作为时间运行的标准。我们给 T1 的定时为 20 毫秒，每当产生 50 个中断时，秒钟计数加 1。具体参看流程图 4-5。



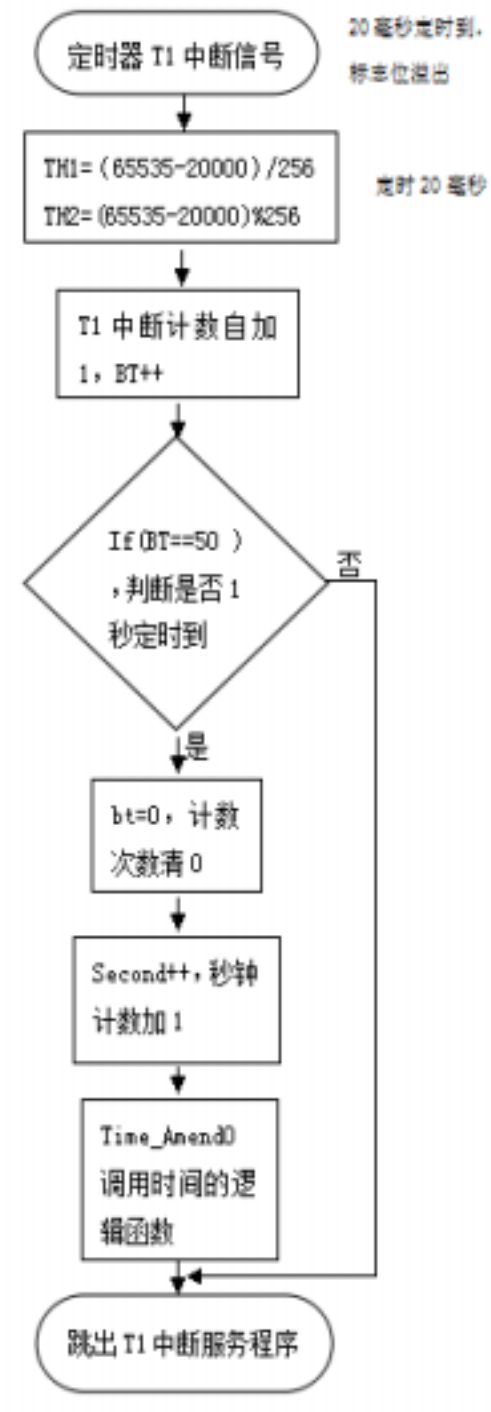


图 4-4 定时器 T0 中断服务程序流程图　　图 4-5 定时器 T1 中断服务程序流程图

4.3 系统软件介绍

4.3.1 ISP 软件

STA-ISP 软件界面简单，操作方便，是专门为 STC系列单片机设计的软件，如图 4-6 所示。首先，打开软件，在单片机类型下选中单片机型号，本设计中采用的是 STC89C51单片机。然后，在“ COM”中选择与设备管理器中相同的串口号，波特率一般保持默认。确认硬件连接正确后，单击“ 打开文件 ”来选择下载 HEX文件，然后点击“ download/ 下载 ”，并连接 VCC线。接着接通电源便即可加载可执行文件 HEX[9]。最后，当加载进度条显示 100%时，说明程序加载成功。再利用上位机发送汉字，操作简单。

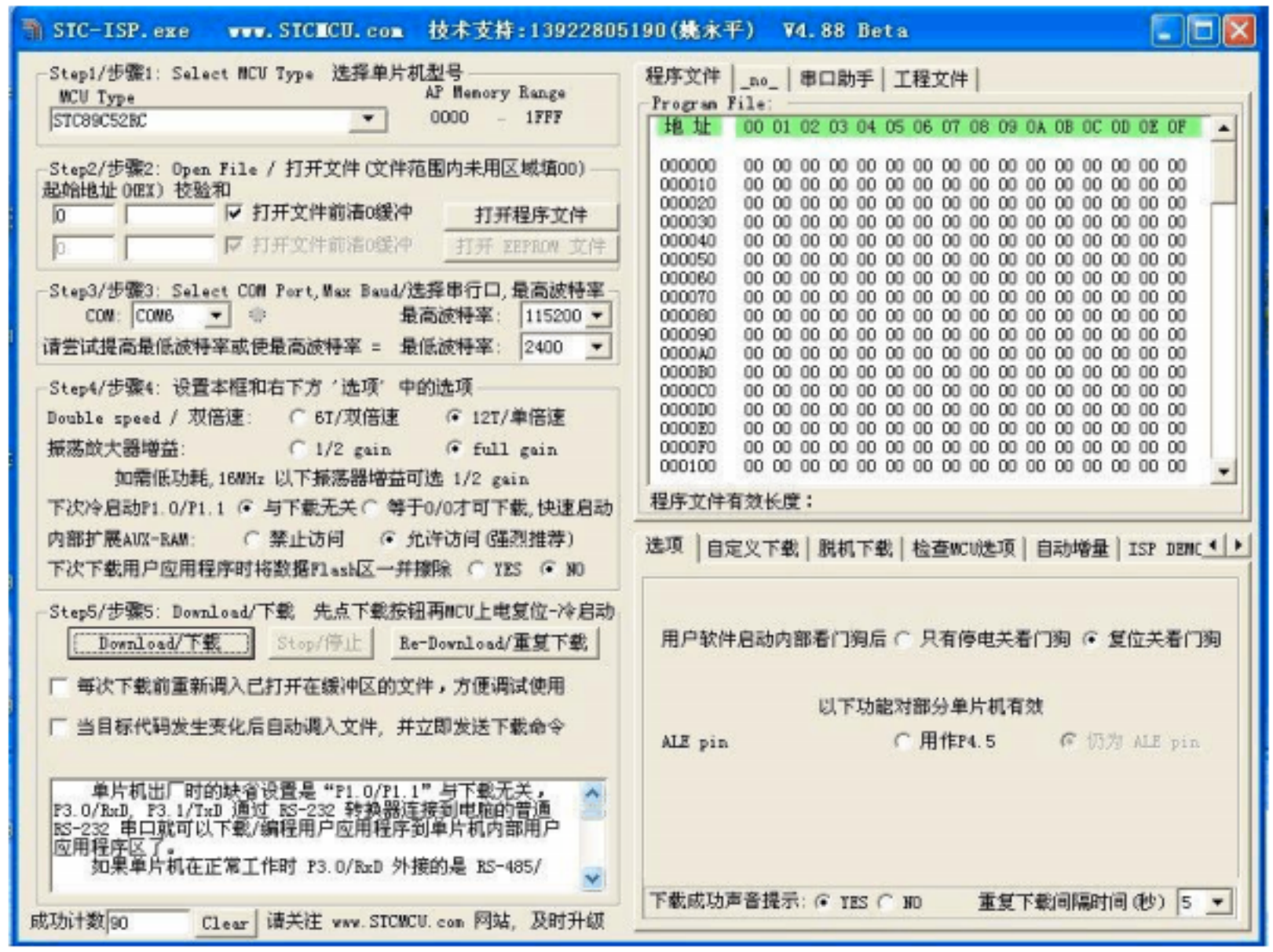


图 4-6 ISP 界面

4.3.2 ISP 软件流程图

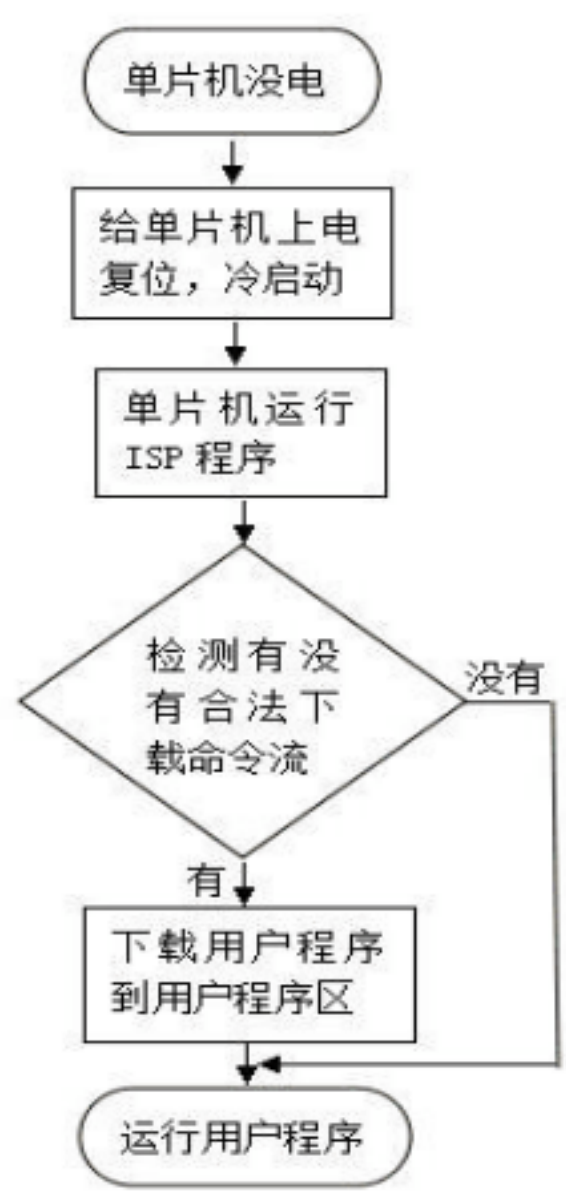


图 4-7 ISP 运行流程图

如图 4-7 所示，单片机通过上电复位后，进行冷启动。冷启动是在电机下载程序后，再把单片机和串口器的 VCC线相连。启动后，单片机就会运行

ISP 软件，并检测是否有下载命令。 如果没有， 将直接运行用户程序； 如果有，则先将用户程序下载到用户程序区，然后再运行程序。

第 5 章 系统调试

5.1 元件焊接

系统由主控板和供电板两部分组成，在元件焊接时需要注意，电阻焊接是不分正负极的，而其他元件是需要区分正负极的[10]，例如二极管和三极管，还有一些电容元件也是需要按照正负极来焊接的。另外，在焊接贴片元件时，需要先在电路板相应的地方焊锡，然后再焊接贴片元件。

5.2 系统调试

焊接完之后，需要对系统进行一个整体检查。首先检查元件的位置、类型是否焊接正确，然后检查每个元件的引脚是否有虚焊或接线短路的现象，并需要检查元件的正负极是否焊接正确。检查修正之后，给系统通电测试，采用分模块测试的方法，检查各个模块中的元件是否能正常工作，元件是否有过热现象。

模块检测完成后，对系统进行功能测试，对整个系统通电检测，检查 LED 显示屏是否能够正常运转。系统功能测试完毕之后，需要对系统进行调试工作，检查 LED 显示屏是否能够正确地显示文字，显示过程是否稳定。调试后的效果图如图 5-1 所示。

图 5-1 系统调试效果图

第 6 章 结论

本设计克服了传统 LED的两个缺点：一个就是传统 LED显示屏组成模块的器件数量较多，成本略高。还有一个就是传统 LED 显示屏显示在一个平面上，有一定视角限制。并且为了解决这两个不足，本设计推出了新型的柱式旋转 LED显示屏，不仅节约了制作器件，而且还可以 360 度观看。

新型的柱式旋转 LED显示屏利用人眼的视觉暂留现象（由于人眼观看景物时，大脑神经对光信号的反应会延迟一段短暂的时间，使得传入的视觉影像会停留一段时间，这一现象叫做人眼的视觉暂留现象。 ），通过单片机控制被固定在电机上的发光二级管的亮灭，同时利用电机带动二极管旋转，并配合传感器触发的外部中断，从而稳定的显示出文字和图案。

参考文献

[1] 童诗白，华成英 . 模拟电子技术基础 [M]. 高等教育出版社， 2006

[2] 周诗虎 . 单片机控制 LED 点阵显示屏 . 科技信息 ,2008.25

[3] 胡汉才 . 单片机原理及其接口技术 [M]. 北京：清华大学出版社 2004.2

[4] 薛峰，朱晓骏 . 单片机原理及应用（修订版） [M]. 北京：北京理工大学出版社， 2011.

[5] 赵嘉蔚，张家栋等 . 单片机原理与接口技术 [M]. 北京：清华大学出版 2010

[6] 靳桅 . 基于 51 系列单片机的 LED显示屏开发技术 [M]. 北京航空航天大学出版社 .2009

[7] 邱元瑞、温坚等 . 基于 TLC5947的旋转 LED屏显示控制器设计 [M]. 江西财经大学 2012

[8] 胡汉才 . 单片机在电子电路设计中的应用 [M]. 北京：清华大学出版社 2006

[9] 马忠梅 . 单片机的 C 语言应用程序设计 [M]. 北京航空航天大学出版社， 2003

[10] 陈大钦 . 电子技术基础实验 [M]. 高等教育出版社主编， 2008

[11] 清源计算机工作 . 室 Protel99 SE 原理图于 PCB设计 [M]. 机械工业出版社， 2001

[12] 谭浩强 . C 语言程序设计 [M]. 清华大学出版社， 2005

[13]Nathan. Single-Technology-Based Statistical Calibration for High-Performance Active-Matrix Organic LED Displays. 2007 , 284~294

[14]The silicon valley, 16 (2012)

[15]K.Alexander. Fundamentals of electric Circuits[M]

附录

显示程序如下：

```
for(i=0;i<16;i++) //          送 16 列       显示   这里只显示一个字。
{
  P2=zimo[i*2]; //          送数据低位显示
  P4=(zimo[i*2+1]);//         送数据高位显示       这里用了单片机   P4 和 P5 口
  P5=(zimo[i*2+1])>>4;//         这里行和列    都是 IO 口独立驱动的   LED
  DelayUs(200);//      延时让  LED亮起来   每列延时的时间
   P2=0XFF;
   P4=P5=0XFF;
 }
```

主程序如下：

```
#include<reg52.h>
unsigned char code
shuzi_0[8]={0x82,0x7C,0x7C,0x7C,0x7C,0x7C,0x82,0xfe};/*"0",0*/
Unsigned char code
shuzi_1[8]={0xfe,0x7e,0x7a,0x00,0x7e,0x7e,0xfe,0xfe};/*"1",1*/
unsigned char code
shuzi_2[8]={0x3a,0x3c,0x5c,0x5c,0x6c,0x6c,0x72,0xfe};/*"2",2*/
unsigned char code
shuzi_3[8]={0xba,0x7c,0x7c,0x6c,0x6c,0x6c,0x92,0xfe};/*"3",3*/
unsigned char code
shuzi_4[8]={0x9e,0xae,0xb6,0xba,0x00,0xbe,0xbe,0xfe};/*"4",4*/
unsigned char code
shuzi_5[8]={0xa0,0x6c,0x6c,0x6c,0x6c,0x6c,0x9c,0xfe};/*"5",5*/
unsigned char code
shuzi_6[8]={0x82,0x6c,0x6c,0x6c,0x6c,0x6c,0x9a,0xfe};/*"6",6*/
```

unsigned char code

shuzi_7[8]={0xfc,0xfc,0xfc,0x0c,0xf4,0xf8,0xfc,0xfe};/*"7",7*/

unsigned char code

shuzi_8[8]={0x92,0x6c,0x6c,0x6c,0x6c,0x6c,0x92,0xfe};/*"8",8*/

unsigned char code

shuzi_9[8]={0xb2,0x6c,0x6c,0x6c,0x6c,0x6c,0x82,0xfe};/*"9",9*/

unsigned char code

fuhao_A[8]={0xfe,0xfe,0xbb,0xbb,0xfe,0xfe,0xfe,0xfe};/*":",10*/

/**

**/

unsigned char code

hanzi_a[2][16]={0xf7,0x37,0x47,0x70,0x17,0x73,0xb7,0xbd,0xdd,0xdd,

0xed,0xe5,0xc9,0xdd,0xbf,0xff,

0xf7,0xe7,0xf7,0xf7,0x00,0xfb,0xbb,0xbd,0xbd,0xbd,0x81,0xbd,0xb

c,0x9d,0xbf,0xff};/* 轻*/

unsigned char code

hanzi_b[2][16]={0xff,0xfb,0xfb,0xfb,0xfb,0xfb,0xfb,0x03,0xfb,0xfb,

0xfb,0xfb,0xfb,0xfb,0xff,0xff,

0xdf,0xdf,0xdf,0xdf,0xdf,0xdf,0xdf,0xc0,0xdf,0xdf,0xdf,0xdf,0xd

f,0xdf,0xdf,0xff};/* 工*/

unsigned char code

hanzi_c[2][16]={0xfb,0xcb,0x3d,0xfb,0x3b,0xc3,0xdf,0xef,0xf0,0x17,

0xf7,0xf7,0xd7,0xe7,0xff,0xff,

0xef,0xf7,0xf9,0xfe,0x7d,0x73,0xbf,0xcf,0xf3,0xfc,0xf3,0xef,0x9

f,0x3f,0xbf,0xff};/* 欢*/

unsigned char code

hanzi_d[2][16]={0xbf,0xbd,0xbb,0x37,0xff,0x03,0xfb,0xfd,0x7d,0x03,

0xfb,0xfb,0xfb,0x01,0xfb,0xff,

0xff,0xbf,0xdf,0xe0,0xdf,0xb8,0xbd,0xbe,0xbf,0x80,0xbf,0xbd,0xb

b,0x9c,0xdf,0xff};/*　　迎*/

unsigned char code

hanzi_e[2][16]={0xbf,0xdf,0xef,0x03,0xdc,0xef,0x73,0x98,0xfb,0x0b,

0xfb,0xbb,0x6b,0x73,0xff,0xff,

　　0xbf,0xcf,0xff,0x88,0x7f,0x7e,0x77,0x6d,0x4b,0x7c,0x7f,0x1f,0xf

f,0xee,0x9f,0xff};/*　　您*/

unsigned int Tcz,s=0,bt=0;//s　　表示中断次数　，Tcz 表示定时器 TO的初值

unsigned char iDex=0,hiDex=0;　　// 控制字母和汉字的码值具体位数

unsigned char Hour=10,Minute=15,Second=45,Hour_Flag=0,buffer=0;

void hanzi_Show(unsigned char m);

void Time_Show(unsigned char m);　　// 数字 0-9 加：显示函数

void Time_Amend();　　// 时间逻辑控制函数

void Hour_Display();　　// 小时显示函数

void Minute_Display();　　// 分钟显示函数

void Second_Display();　　// 秒钟显示函数

void A_Display();　　// “ :”显示函数

void Init();　　// 初始化函数

int k=0;

void INT00(void)interrupt 0 //　　外部中断服务函数，定位更新显示和定

时器 0 初值校正

{

　　k=k+(s-180);　　//k　　为校正值，当定时中断过快， K 变大，Tcz

同时变大；

　　Tcz=Tcz+k;　　//Tcz　　是用来给定时器 T0 赋初值的，通过这个公式

来校正 T0 的初值,

　　iDex=0;　　//iDex　　值清零，防止调用显示子函数中时的值不同

步，出现乱码

　　hiDex=0;　　//hiDex　　值清零，防止调用显示子函数中时的值不同

步，出现乱码

```
    s=0;                                    // 定时器中断次数清零，  更新显示第二周
}
void  kjp_test(void)interrupt 1  //              定时器中断  0 中断服务程序。用于判
                                              断显示位置，和  //  控制显示
{     TH0=(-Tcz)/256;              //              把校正后的值给定时器    T0赋值
      TL0=(-Tcz)%256;
      if(buffer==0)                //                  判断  buffer    的值，确定显示上面内
容
      {                                       //  显示模式选择，   0 为数字式时钟
          if(iDex>=8){iDex=0;}
          if(s<16)                 //                  判断  T0 的终端次数，是否小于    16
             Hour_Display();       //              小于  16，调用显示 “ 小时值 ” 的子函
数
          if(s>=16&&s<24)          //          判断 T0 的中断次数是否大于   16 且小于  24
             A_Display();          //          是则调用显示 “ ： ” 的子函数
          if(s>=24&&s<40)          //          判断位置,
             Minute_Display();     //          调用显示 “ 分钟值 ” 的子函数
          if(s>=40&&s<48)          //          判断位置,
            A_Display();           //                  调用显示 “ ： ” 的子函数
          if(s>=48&&s<64)          //          判断位置
             Second_Display();     //          调用显示 “ 秒钟值 ” 的子函数
          if(s>=64)                //                  判断位置
          {
              P0=0xfe;P2=0xff;     //          关闭所有  LED
          }
      }
      if(buffer==1)                //                  判断  buffer    的值，确定显示内容，    1 为显
示 “ 轻工欢迎您 ”
      {
```

```
if(hiDex>=16)hiDex=0;
if(s<16)                    //      判断位置是否在  0-32 度之间
{
    hanzi_Show(0);          //   调用显示“ 轻 ”的子函数
}
if(s>=16&&s<32)             //          判断位置是否在  32-64 度之间
{
    hanzi_Show(1);          //          调用显示“ 工 ”的子函数
}
if(s>=32&&s<48)             //          判断位置是否在  64-96 度之间
{
    hanzi_Show(2);          //          调用显示“ 欢 ”的子函数
}
if(s>=48&&s<64)             //          判断位置是否在  96-128 度之间
{
    hanzi_Show(3);          //          调用显示“ 迎 ”的子函数
}
if(s>=64&&s<80)             //          判断位置是否在  128-160 度之间
{
    hanzi_Show(4);          //          调用显示“ 您 ”的子函数
}
if(s>=80)                   //          判断位置是否大于  160 度
{
        P0=0xff;P2=0xff;    //          关闭所有显示
    }
}
s++;                        //          定时器 T0 中断次数加 1
}
void Time_luoji()interrupt 3    //          定时器 T1 中断服务程序，用于控制
```

秒钟的跳动。

```
{     TH1=(65535-20000)/256;              //     给定时器  T1 赋初值，定时   20 毫秒
      TL1=(65535-20000)%256;
      bt++;                       //                  定时器中断次数加   1
if(bt==50)              //                判断  T1 中断次数是否到了   50，及判断一秒定
时到了没有
      {
            bt=0;             //              定时器  T1 中断次数清零，为下一秒做准备
            Second++;      //       秒钟值加  1
            Time_Amend();  //    调用时间控制的逻辑子函数，让秒钟到     60 分钟加
1。。。类推
      }
}
void Time_Show(unsigned char m)
{
      switch (m)
      {
            case 0 :P0=0xff; P0=shuzi_0[iDex];iDex++;P2=0xff;break; //
显示 1
            case 1 :P0=0xff; P0=shuzi_1[iDex];iDex++;P2=0xff;break; //              显
示 2
            case 2 :P0=0xff; P0=shuzi_2[iDex];iDex++;P2=0xff;break; //
显示 3
            case 3 :P0=0xff; P0=shuzi_3[iDex];iDex++;P2=0xff;break; //              显
示 4
            case 4 :P0=0xff; P0=shuzi_4[iDex];iDex++;P2=0xff;break; //
显示 5
            case 5 :P0=0xff; P0=shuzi_5[iDex];iDex++;P2=0xff;break; //              显
示 6
```

```
        case 6 :P0=0xff; P0=shuzi_6[iDex];iDex++;P2=0xff;break; //          显
示 7
        case 7 :P0=0xff; P0=shuzi_7[iDex];iDex++;P2=0xff;break; //          显
示 8
        case 8 :P0=0xff; P0=shuzi_8[iDex];iDex++;P2=0xff;break; //          显
示 9
        case 9 :P0=0xff; P0=shuzi_9[iDex];iDex++;P2=0xff;break; //          显
示 0
        case 10 :P0=0xff; P0=fuhao_A[iDex];iDex++;P2=0xff;break; //
显示：
    }
}
void hanzi_Show(unsigned char m)
{
switch (m)
    {
        case 0 :P0=0xff;
P0=hanzi_a[0][hiDex];P2=0xff;P2=hanzi_a[1][hiDex];hiDex++;break;
    // 轻
        case 1 :P0=0xff;
P0=hanzi_b[0][hiDex];P2=0xff;P2=hanzi_b[1][hiDex];hiDex++;break;
    // 工
        case 2 :P0=0xff;
P0=hanzi_c[0][hiDex];P2=0xff;P2=hanzi_c[1][hiDex];hiDex++;break;
    // 欢
        case 3 :P0=0xff;
P0=hanzi_d[0][hiDex];P2=0xff;P2=hanzi_d[1][hiDex];hiDex++;break;
    // 迎
        case 4 :P0=0xff;
```

```
P0=hanzi_e[0][hiDex];P2=0xff;P2=hanzi_e[1][hiDex];hiDex++;break;
        // 您
        }
}
void A_Display()            //                  显示“ ：”的子函数
{
        Time_Show(10);          //          调用显示数字子函数,
}
void Hour_Display()
{                                           //  显示“ 小时值 ”的子程序
        unsigned char m=0,n=0;
        m=Hour/10;                          //  把 hour 值求模，得到十位值
        n=Hour%10;              //              把 hour 值求余，得到各位值
        if(s<8)
            Time_Show(m);           //          调用显示数字子函数，显示十位。
        else
            Time_Show(n);       //          调用显示数字子函数，显示个位
}
void Minute_Display()           //  显示“ 分钟值 ”的子程序
{
        unsigned char m=0,n=0;
        m=Minute/10;                //              把 Minute 值求模，得到十位值
        n=Minute%10;                //              把 Minute 值求余，得到各位值
        if(s<32)
            Time_Show(m);           //              调用显示数字子函数，显示十位。
        else
            Time_Show(n);           //              调用显示数字子函数，显示个位
}
void Second_Display()           //                  显示“ 秒钟值 ”的子程序
```

```
{
    unsigned char m=0,n=0;
    m=Second/10;                    //                把 Second值求模，得到十位值
    n=Second%10;                    //                把 Second值求余，得到各位值
    if(s<56)
        Time_Show(m);               //                调用显示数字子函数，显示十位
    else
        Time_Show(n);               //                调用显示数字子函数，显示个位
}
void Time_Amend()                   //                时间控制值函数
{
    if(Second>=60)                  //  判断 Second值是否到 60,
    {
        Second=0;           // Second            值清零
        Minute++;           // Minute                自加 1
        if(Minute>=60)      //                判断 Minute 是否到了 60
        {
            Minute=0;       // Minute                清零
            Hour++;         // Hour                自加 1
            if(Hour>=24)    //                判断 Hour 是否到了 24
                Hour=0;     // Hour            归零
        }
    }
}
void Init()
{
```

TMOD=0x11; // 定时器 / 计数器的方式控制。将 T0 设为方式 2, 将 T1 设为方式 2

EA=1; // 中断允许控制

ET0=1; // 定时器 / 计数器 T0 的中断允许位

ET1=1; // 定时器 / 计数器 T1 的中断允许位

TH0=(65535-600)/256; // 定时初值， 5MS

TL0=(65535-600)%256;

TH1=(65535-50000)/256; // 定时器 1 初值

TL1=(65535-50000)%256;

TR0=1; // 定时器 / 计数器 T0 的启动

TR1=1;

EX0=1; //INT0 中断允许控制位

IT0=1;

IT1=1; // 控制寄存器 TCON

PT1=1; // 设置定时器 T1 中断先级为高，防止 T0 中出现，不处理 T1 中断服务程序

}

void main() //main 函数，初始化各数据，然后进入死循环等待各个中断

{

Init(); // 初始化函数

while(1) // 进入死循环

{

;

}

}