肖　婧　编著

单片机系统设计与仿真

——基于Proteus

北京航空航天大学出版社

单片机系统设计与仿真

——基于 Proteus

肖　婧　编著

北京航空航天大学出版社

内 容 简 介

本书介绍了5大类共12个功能各异且非常实用的单片机控制系统的设计方法及过程，读者既能学习到单片机系统进行仿真设计的全部过程及基本方法，同时也可以掌握常用控制器件的应用知识。

本书内容丰富、通俗、实用，适合于有一定基础的单片机初学者的自学及实践，可用作高等院校学生的教材，也可用作相关科研人员、培训人员的参考资料。

图书在版编目(CIP)数据

单片机系统设计与仿真：基于 Proteus / 肖婧编著.
-- 北京：北京航空航天大学出版社，2010.8
ISBN 978-7-5124-0159-4

Ⅰ. ①单… Ⅱ. ①肖… Ⅲ. ①单片微型计算机－系统设计－应用软件，
Proteus②单片微型计算机－系统仿真－应用软件，
Proteus Ⅳ. ①TP368

中国版本图书馆 CIP 数据核字(2010)第 139013 号

单片机系统设计与仿真
——基于 Proteus
肖 婧 编 著
责任编辑 董立娟
*
北京航空航天大学出版社出版发行
北京市海淀区学院路 37 号(邮编：100191) http://www.buaapress.com.cn
发行部电话：(010)82317024 传真：(010)82328026
读者信箱：emsbook@gmail.com 邮购电话：(010)82316936
北京市松源印刷有限公司印装 各地书店经销
*
开本：787 mm×960 mm 1/16 印张：16.75 字数：375 千字
2010 年 8 月第 1 版 2010 年 8 月第 1 次印刷 印数：4 000 册
ISBN 978-7-5124-0159-4 定价：32.00 元

前　言

单片机也被称作“微控制器”、“嵌入式微控制器”、“单片微控制器”。它不是完成某一个逻辑功能的芯片，而是把一个计算机系统集成到一个芯片上。从1974年世界上第一台单片微型计算机诞生至今，虽然仅历经30多年的发展历程，但如今单片机已在生产生活的多个领域得到了广泛的应用。那么，单片机系统设计究竟是怎样进行的，初学者又能否快速掌握这一技术呢？

本书就是在这样的背景之下应运而生的。它从单片机系统设计的相关知识入手，结合12个从实际生活中模拟到的单片机应用系统的具体设计，引导读者由浅入深地学习与掌握单片机系统设计的方法；同时，也可为今后进行更为复杂系统的设计打下良好的基础。

在章节划分上，本书主要分为4章。第1章介绍了单片机系统设计的内涵，其中包括了单片机系统设计前的知识储备以及系统设计的过程分析等。第2章介绍了单片机系统设计的工具，其中主要介绍了Proteus仿真软件的使用方法以及单片机C语言编程的方法。第3章为单片机系统设计初体验。此章由理论到实际，结合一个简单的单片机系统，介绍了初学单片机系统仿真设计的实际过程。第4章为单片机系统设计的实战章节。此章从显示、温度控制、电机控制、声音控制、通信控制5个方面详细介绍了12个功能各异、有一定实际应用价值的单片机应用系统的具体设计过程。这12个应用系统包括4方向实用交通控制系统、基于点阵LED显示屏的实时电子万年历显示器、LCD奥运宣传牌设计、多路智能温度测控系统、模拟自动恒温控制系统、模拟电梯显示控制系统、智能电机转速控制显示系统、多功能音乐播放器、智能防盗密码锁报警系统、基于单片机的红外遥控系统、双机串行通信系统、基于单片机的简易智能信号源发生器等。同时，本书还将设计中涉及的相关器件使用原理进行了一定的介绍。此外，在附录中还为读者提供了多个可自学体验的系统设计题目、PCB布线的实用方法等相关应用资料。

本书在编写的过程中，得到了湖南商学院领导和有关同志的支持和帮助，在此对他们表示衷心的感谢。

由于编者时间仓促，水平有限，书中必有疏漏及错误之处，敬请读者批评指正。

有兴趣的读者可以发送电子邮件到：jx882003@yahoo.com.cn，与作者进一步交流；也可以发送电子邮件到：xdhydcd5@sina.com，与本书策划编辑进行交流。

编　者

2010年4月

目 录

第 1 章

单片机系统设计的内涵

1.1 概 述

单片机也称作“微控制器”、“嵌入式微控制器”、“单片微控制器”。它不是完成某一个逻辑功能的芯片，而是把一个计算机系统集成到一个芯片上。从 1974 年世界上第一台单片微型计算机诞生至今，虽然仅历经 30 多年的发展历程，但如今已在生产生活的多个领域得到了广泛的应用。

那么，单片机系统设计究竟是怎样进行的呢？设计的内涵是怎样的呢？下面就先从单片机系统设计前的准备工作说起。

1.2 单片机系统设计前的准备工作

在过去初学单片机时，我们基本掌握了一些有关芯片的基本结构及编程等相关知识，但这离实际的使用还有不少的差距。现今社会中，不仅仅利用单一的芯片，而是要将单片机应用到具体的系统设计之中。为了更好地实现后面的系统设计，必须在设计之前完成一定的准备工作。

1.2.1 设计前的知识储备

(1) 使用的芯片及设备知识

单片机适用于电子玩具、工业控制、民用电器、机电一体化产品、航天航海等众多领域，进行系统设计时必然要用到许多芯片以及设备器件，如单片机、显示器件、按键与键盘、电机、电源电路器件、复位及保护电路器件、编程器等。此外，单片机的开发应用还涉及硬件扩展接口和各类传感器。这些芯片器件的结构、外部连线及使用方法等知识都是系统设计前必须了解的。只有对芯片及设备的内部结构与使用方法都非常了解，才有可能做好系统的设计。当然，目前有许多的应用手册可供设计者查阅。因此，在具体应用时，也可根据实际情况去翻查手册。不过基本的应用常识应当预先学习，否则即便是查手册，也可能出现不知从何处入手的情况。

(2) 应用设计的编程语言学习

进行单片机系统设计，必须了解并熟练掌握单片机设计的编程语言，否则设计就无从谈起。目前进行设计的语言，常用的就是汇编语言与 C 语言。初学单片机时可学习使用汇编语言来进行设计，但学到一定程度时，就应从设计效率出发来学习使用 C 语言进行单片机系统设计。C 语言是简洁、高效、而又最贴近硬件的高级编程语言，许多厂商在推出新的单片机产品时也纷纷配套 C 语言编译器。笔者建议读者在掌握汇编语言单片机编程方法的基础上，应学习与掌握单片机的 C 语言编程方法。之后将两种语言融汇贯通，最终实现可交叉使用来完成混合编程。

本书的设计采用的是 C 语言编程，还为读者介绍了一些 C51 单片机编程的设计经验，希望读者能有所借鉴。

(3) 设计所需的环境了解

进行系统设计时，可能用到许多设计方法及设计工具。比如利用实际的芯片，直接做成实际的硬件系统；或是利用一定的软件环境先进行模拟仿真设计，设计成功后再转化到实际的系统设计中。不论是采用何种设计方式，相关的设计环境是必须要了解的，这就包括编程环境的掌握和设计仿真软件环境的了解等。换言之，就是应当掌握设计中所用工具的使用方法。比如编程软件，C 语言的编程工具可采用 C 专用的编程工具软件(如 Franklin C51、KeilL C51 等)，也可采用对汇编及 C 语言都可适用的编程工具(如 WAVE)等。设计工具可用到仿真设计软件(如 Proteus)，此外，还有芯片烧写器等。

1.2.2　学会分析任务及总结经验

完成设计前的知识储备之后，接下来就必须学习分析设计任务。人们见到的设计任务往往就是一些抽象的描述语言。比如要求设计一个交通控制系统，只会提到控制的交通灯数量情况如何、各种灯亮灭的时间有多久等实际系统使用时的功能实现效果要求。系统由哪些组成部分、具体应用到哪些电路元件、设计中采用何种设计的语言、设计中采用哪些软件等内容，在初始的设计任务描述中是不会也不可能出现的。因此，人们就将抽象的设计任务转变为适于设计的具体设计思路内容。

初次开发时由于没经验，可能要经过多次反复才能完成项目。每完成一次设计，就会得到较大的收获和积累，如硬件设计方面、软件设计方面以及设计经验方面的积累等。

在单片机硬件开发设计中应注意以下几个方面：

① 单片机应用开发者在学习好基本单片机系统设计的方法后，应学习应用最新单片机(MCU)进行设计。因为新型 MCU 在很多方面具有旧 MCU 所不具备的优点，因此其应用相对也会更为广泛。新型 MCU 的优势表现在时钟频率的进一步提高(如从 6 MHz 提高到 33 MHz)、指令执行速度的提高(从 12 个机器周期到 6 个机器周期，甚至到 1 个机器周期)、处理器相关功能的提高(如增加了数学处理、模糊控制等)、内部程序存储器和数据存储器容量的

进一步扩大(ROM扩大到64 KB,RAM扩大到2 KB)、A/D和D/A转换器的内部集成、LCD显示等功能模块的内部集成以及外部扩展功能的增强等。如NXP的P89C884单片机内部有64 KB Flash(快闪存储器)、3个计数器、33 MHz时钟、6个机器周期执行一条指令、I^2C总线、ISP/IAP等。

② C语言是普及最广泛的程序设计语言,既有高级语言的各种特点,又可对硬件进行操作,并可进行结构化程序设计,用其编写的程序较容易移植。目前已有专为单片机设计的C语言编译器,如Franklin C51、Keil C51,它们可生成简捷可靠的目标代码,在代码效率和代码执行速度上完全可以和汇编媲美。

③ 扩展了RS-232等标准串口以后,单片机可和PC机通信,对于众多测控方面的人机对话、报表输出、集成控制等功能进行优势互补。如果芯片支持ISP/IAP功能,则还可以进行在线仿真和远程调试远程软件升级。例如,Dallas的1位总线、NXP的I^2C总线等接口均配有较多的专用扩展接口,接口扩展十分方便,所配软件有标准模式,也较容易编写。

④ 扩展接口的开发尽可能采用PSD、FPGA(或CPLD)等器件。这类器件都有开发平台的支持,开发难度较小,开发出的硬件性能可靠、结构紧凑、利于修改、保密性好;这也是硬件接口开发的趋势。如Altera公司生产的EPM7128S应用较广,在中国市场也容易买到;WSI推出的新型可编程的单片机外围器件PSD813F,把单片机外围电路中的许多功能模块组合在一起,为用户提供体积更小、成本更低、开发更快的解决方案。

有时开发一个单片机应用项目,在仿真调试完成后系统运行正常,而接入现场后不能正常运行或运行时好时坏,脱离现场后又一切正常,这种现象就涉及可靠性问题。解决这种问题可以从以下几个方面考虑:

- 选择性能好、抗干扰能力强的供电系统,尽量少地从电源引入干扰;
- 设计电路板时排除可能引起干扰的因素,合理布线,避免高频信号的干扰;
- 选择较好的接地方式,如模拟地和数字地采用一点接地方式,驱动大电流信号时采用光电隔离;
- 数据采集时进行数字滤波处理,常用的数字滤波方式有程序判断滤波、中位值滤波、算术平均滤波、递推平均滤波、防脉冲干扰平均值滤波和一阶滞后滤波等。

由于干扰可能是不同的原因引起,在设计时要根据项目应用场所分析可能出现的干扰,有目的地设计抗干扰电路。

1.3　单片机系统设计的过程

掌握了系统设计所必须采用的基本器件使用方法之后,将一同来了解单片机系统设计的基本过程,如图1.3.1所示。

1. 确定设计方案

无论是怎样的设计，功能复杂或是功能简单，确定设计方案是设计前必须进行的第一项工作。设计者应首先对设计的可行性进行分析，即分析设计所需的知识面、硬件/软件功能的实现难易程度等是否在设计者的可承受范围之内，还有就是设计时所需的设计条件是否都已齐备等。

当确认设计的可行性之后，就应分析系统实现的整体功能以及相关性能指标。有一些测控系统实现的功能可能有许多种，设计时必须确定这一次设计时具体应实现哪几项功能。同时，在确认实现功能时，就必须确定相应的性能指标。比如显示达到几位，精度达到多少，检测对象是什么，检测误差范围怎样等。

之后就可根据系统实现的功能，结合自身的技术条件以及经济条件等确定具体设计方案。设计方案中应包括设计时所要实现各项功能对应的设计思路，设计者在这一步骤中将抽象的设计功能任务转化为实际的、便于操作易懂的设计思路描述性语言。总体方案是整个系统实现的关键，若方案确定有误，那么无论后面的设计如何，整体系统的设计功能都不可能达到最初的设计要求。因此，设计方案在分析时务必详尽、具体。

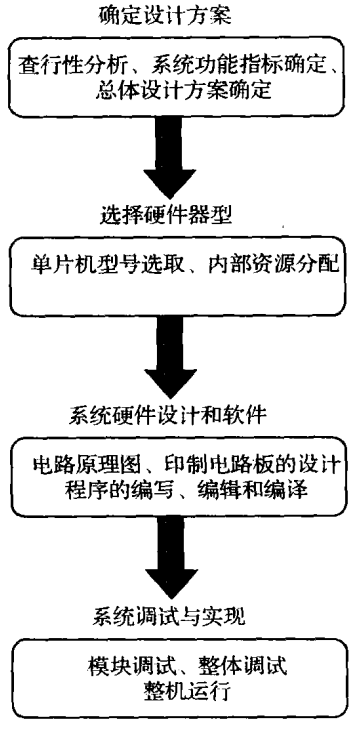

图 1.3.1　单片机系统设计过程示意图

2. 选择硬件类型

确定了总体设计方案后，应选择合适的硬件类型。这主要是选择合适的单片机型号，同时确定单片机内部资源的总体分配情况。此外，单片机的一些基本外围电路也应一并确定，比如晶振电路、复位电路、电源电路等。

目前，单片机种类繁多，型号也是五花八门。选用时应从设计方案的角度出发，选择适合自身设计、同时经济成本又不是太高的型号为宜。当然，如果设计对于运行速度、封装外形、今后扩展等方面有特别的要求，同时对于成本方面又没有特别的考量，那么设计者就可考虑选用一些目前性能很不错但价格相对较高的单片机型号。

3. 系统硬件和软件的设计

对系统进行具体设计是单片机系统设计的核心步骤，包括硬件设计和软件设计两个方面。硬件设计主要包括设计电路原理图、设计印制电路板(PCB)或用万用板直接焊出实验板等操作工序。软件设计则主要是根据硬件电路原理图，结合设计的功能要求进行软件程序的编写、

编辑和编译，并最终将程序写入单片机中。

在硬件设计中，电路原理图的设计是基础也是关键。在这一设计环节中，必须根据设计的功能要求，结合使用的单片机型号，合理选用各种硬件电路元器件。然后，根据各器件的控制应用基础知识进行线路的连接。绘制电路原理图时，除了考虑系统设计实现的功能外，还必须考虑电气规则以及今后实际硬件工作时是否会有干扰等问题。

在软件设计时，设计者先选择好设计的编程语言种类，如采用C语言或汇编语言等，然后选择一个相应的编程工具软件。此后，设计者应根据设计功能要求以及单片机内部资源的分配情况，先列出程序设计的思路并画出程序实现的基本流程框图。然后，再根据具体绘制好的硬件电路原理图，结合各器件引脚控制的电平信号特点来进行具体程序的编写。编写程序时，应为今后的调试有所考虑，如写详细的注释等。

4. 系统调试与实现

在前面的设计环节都完成之后，接下来就是系统的调试与运行实现阶段了。

在系统调试中，一般先对硬件及软件进行分模块的调试，然后再进行系统的整体调试。在分模块调试中，对于硬件，主要看硬件电路整体连线情况、电路供电情况以及信号的输入与输出是否都在设计的指标要求范围之内；对于软件，主要看主程序以及各子程序的各条语句是否存在语法输入错误、实现功能上是否符合设计流程的要求、程序运行时是否能按照要求进行、是否存在不能正常启停程序等问题。

各模块均调试完成后，会把程序写入单片机中，然后组装连接各硬件模块，看系统通电运行后的各模块配合情况以及整体实现功能。有时会发现模块分开调试时都很正常，但组装起来整体调试就会出现各种不同的问题。其实，这往往就是在设计电路原理图以及印制电路板时缺乏经验，或考虑欠缺导致的。出现问题后，应先分析问题出现的状态，看其是出在什么环节上。一般情况下按照先查找软件问题再查找硬件问题的顺序去查找问题出现的原因。这是因为相对而言，软件问题不需要对整体设计进行调整，设计调整难度相对小很多。当软件修改仍不能解决问题时，就只能考虑根据电路原理图去查找硬件方面的问题了。

当所有故障、设计障碍等均被排除之后，应再将程序通过编程器烧写固化到单片机中，并将系统放置到最终的使用现场进行实际运行且工作一段时间。当系统能够稳定地完成设计所有的功能时，就可认定系统已达到设计的各项要求了。

1.4　软件程序编写规范

随着社会的进步，系统设计功能不断增强，相应的软件程序代码也就越来越复杂，源文件也越来越多。对于设计开发者而言，除了保证程序运行的正确性和提高代码的运行效率外，规范风格的程序编码会对软件的升级、修改、维护带来极大的方便，也能保证程序员不陷入“代码泥潭”中无法自拔。设计一个成熟的软件产品或是程序文件，除了有详细丰富的设计文档之

外，必须在编写代码时就有条不紊，细致严谨。

本书总结了 11 条编码规范，包含了程序排版、注释、命名、可读性、变量、程序效率、质量保证、代码编译、代码测试和版本控制等内容，希望读者能从中掌握到编写软件程序时的一些基本规则。

(1) 排　版

关键词和操作符之间加适当的空格。在相对独立的程序块与块之间加空行。

较长的语句、表达式等要分成多行书写。划分出的新行要进行适当的缩进，使排版整齐，语句可读。长表达式要在低优先级操作符处划分新行，操作符放在新行之首。循环、判断等语句中若有较长的表达式或语句，则要进行适应的划分。若函数或过程中的参数较长，则也要进行适当的划分。

不允许把多个短语句写在一行中，即一行只写一条语句。函数或过程的开始、结构的定义及循环、判断等语句中的代码都要采用缩进风格。

C/C++语言是用大括号“{”和“}”界定一段程序块的，编写程序块时“{”和“}”应各独占一行并且位于同一列，同时与引用它们的语句左对齐。在函数体的开始、类的定义、结构的定义、枚举的定义以及 if、for、do、while、switch、case 语句中的程序都要采用如上的缩进方式。

(2) 注　释

注释要简单明了。边写代码边写注释，修改代码同时修改相应的注释，以保证注释与代码的一致性。要在必要的地方注释，注释量要适中。注释的内容要清楚、明了，含义准确，防止注释二义性。保持注释与其描述的代码相邻，即注释的就近原则。

对代码的注释应放在其上方相邻位置，不可放在下面。对数据结构的注释应放在其上方相邻位置，不可放在下面；对结构中每个域的注释应放在此域的右方；同一结构中不同域的注释要对齐。

变量、常量的注释应放在其上方相邻位置或右方。全局变量要有较详细的注释，包括对其功能、取值范围、哪些函数或过程存取它以及存取时注意事项等的说明。

在每个源文件的头部要有必要的注释信息，包括文件名、版本号、作者、生成日期、模块功能描述(如功能、主要算法、内部各部分之间的关系、该文件与其他文件关系等)、主要函数或过程清单及本文件历史修改记录等。

在每个函数或过程的前面要有必要的注释信息，包括函数或过程名称、功能描述、输入、输出及返回值说明、调用关系及被调用关系说明等。

(3) 命　名

对于较短的单词可通过去掉“元音”形成缩写；而较长的单词可取单词头几个字符的优先级，并用括号明确表达式的操作顺序，避免使用默认优先级。

(4) 可读性

首先，避免使用不易理解的数字，用有意义的标识来替代。其次，不要使用难懂的技巧性

很高的语句。然后，源程序中关系较为紧密的代码应尽可能相邻。

(5) 变　量

去掉没必要的公共变量。

构造仅有一个模块或函数可以被修改或创建、而其余有关模块或函数只能被访问的公共变量，防止多个不同模块或函数都可以修改、创建同一公共变量的现象。

仔细定义并明确公共变量的含义、作用、取值范围及公共变量间的关系。明确公共变量与操作此公共变量的函数或过程的关系，如访问、修改及创建等。向公共变量传递数据时，要十分小心，防止赋予不合理的值或越界等现象发生。防止局部变量与公共变量同名。

仔细设计结构中元素的布局与排列顺序，使结构容易理解、节省占用空间，并减少引起误用现象。结构的设计要尽量考虑向前兼容和以后的版本升级，并为某些未来可能的应用保留余地(如预留一些空间等)。

留心具体语言及编译器处理不同数据类型的原则及有关细节。严禁使用未经初始化的变量。声明变量的同时对变量进行初始化。编程时，要注意数据类型的强制转换。

(6) 函数及过程

函数的规模尽量限制在 200 行以内，且一个函数最好仅完成一个功能。要为简单功能编写函数。而函数的功能应该是可以预测的，也就是只要输入数据相同就应产生同样的输出。

尽量不要编写依赖于其他函数内部实现的函数。避免设计多参数函数，不使用的参数从接口中去掉。用注释详细说明每个参数的作用、取值范围及参数间的关系。要检查函数所有参数输入的有效性以及函数所有非参数输入的有效性，如数据文件、公共变量等。

函数名应准确描述函数的功能。避免使用无意义或含义不清的动词为函数命名。

函数的返回值要清楚、明了，让使用者不容易忽视错误情况。明确函数功能，精确(而不是近似)地实现函数设计。

减少函数本身或函数间的递归调用。编写可重入函数时，若使用全局变量，则应通过关中断、信号量(即 P、V 操作)等手段对其加以保护。

(7) 可测性

编写代码之前，应预先设计好程序调试、测试的方法、手段以及各种调测开关及相应测试代码(如打印函数等)。

在进行集成测试/系统联调之前，要构造好测试环境、测试项目及测试用例，同时仔细分析并优化测试用例，以提高测试效率。

(8) 程序效率

编程时要经常注意代码的效率。在保证软件系统的正确性、稳定性、可读性及可测性的前提下，提高代码效率。但不能一味追求代码效率，而对软件的正确性、稳定性、可读性及可测性造成影响。

编程时，要随时留心代码效率；优化代码时，要考虑周全。要仔细构造或直接用汇编语言

编写调用频繁或性能要求极高的函数。可考虑通过对系统数据结构划分与组织的改进，以及对程序算法的优化来提高空间效率。

尽量减少循环嵌套层次。在多重循环中，应将最忙的循环放在最内层。应避免循环体内含判断语句，并将循环语句置于判断语句的代码块之中。

尽量用乘法或其他方法代替除法，特别是浮点运算中的除法。

(9) 质量保证

在软件设计过程中必须构筑软件质量，这主要包括：

① 代码质量保证优先原则。

- 正确性，指程序要实现设计要求的功能。
- 稳定性、安全性，指程序稳定、可靠、安全。
- 可测试性，指程序要具有良好的可测试性。
- 规范/可读性，指程序书写风格、命名规则等要符合规范。
- 全局效率，指软件系统的整体效率。
- 局部效率，指某个模块/子模块/函数的本身效率。
- 个人表达方式/个人方便性，指个人编程习惯。

② 只引用属于自己的存储空间。防止引用已经释放的内存空间。过程/函数中分配的内存，在过程、函数退出之前要释放。

③ 过程/函数中申请的(为打开文件而使用的)文件句柄在过程/函数退出前要关闭。要防止内存操作越界。

④ 时刻注意表达式是否会上溢、下溢。认真处理程序所能遇到的各种出错情况。

⑤ 系统运行之初要初始化有关变量及运行环境，防止引用未经初始化的变量。另外，要对加载到系统中的数据进行一致性检查。

⑥ 严禁随意更改其他模块或系统的有关设置。也不能随意改变与其他模块的接口，必须在充分了解系统的接口之后，再使用系统提供的功能。

⑦ 要时刻注意易混淆的操作符。当编完程序后，应从头至尾检查一遍这些操作符。不使用与硬件或操作系统关系很大的语句，而使用建议的标准语句。

⑧ 使用第三方提供的软件开发工具包或控件时，首先要注意充分了解应用接口、使用环境及使用时注意事项。其次不能过分相信其正确性。另外，除非必要，否则不要使用不熟悉的第三方工具包与控件。

(10) 代码编译

编写代码时要注意随时保存并定期备份，防止由于断电、硬盘损坏等原因造成代码丢失。

同一项目组内，最好使用相同的编辑器及相同的设置选项，并统一编译开关选项。合理地设计软件系统目录，方便开发人员使用。打开编译器的所有报警开关，然后再对程序进行编译。

使用工具软件(如 Visual SourceSafe)对代码版本进行维护。

(11) 代码测试与维护

代码的单元测试要求至少达到语句覆盖,而且要跟踪每一条语句,并观察数据流及变量的变化。其次,清理、整理或优化后的代码也要经过审查及测试。当代码版本进行升级时,必须经过严格的测试方可进行使用。

1.5 单片机控制板的设计原则

(1) 元器件布局的考虑

元器件布局时,应该把相互有关的元件尽量放得近一些。例如,时钟发生器、晶振、CPU的时钟输入端都易产生噪音,在放置的时候应该把它们靠近些。对于那些易产生噪声的器件、小电流电路、大电流电路、开关电路等,应尽量使其远离单片机的逻辑控制电路和存储电路(ROM、RAM);如有可能,则将这些电路另外制成电路板,会更加有利于抗干扰,提高电路工作时的可靠性。

(2) 去耦电容的设置

尽量在关键元件,如 ROM、RAM 等芯片旁边安装去耦电容。实际上,印制电路板走线、引脚走线和接线等都可能含有较大的电感效应。大的电感可能在 VCC 走线上引起严重的开关噪声尖峰,防止其产生的唯一方法是在 VCC 与电源地之间安放一个 0.1 μF 的去耦电容。如果电路板上使用的是表面安装零件,则可以用片状电容直接紧靠着元件,在 VCC 引脚上固定。最好是使用瓷片电容,因为这种电容有较低的静电损耗(ESL)和高频阻抗,另外这种电容在温度和时间上的介质稳定性也很不错。尽量不要使用钽电容,因为它在高频下的阻抗较高。

另外在安放去耦电容时需要注意以下几点:

① 在印制电路板的电源输入端跨接 100 μF 左右的去耦电容;如果体积允许,电容量大一些更好。

② 原则上每个集成电路芯片的旁边都需要放置一个 0.01 μF 的瓷片电容;如果电路板的空隙太小而放置不下,可以每 10 个芯片左右放置一个 1～10 μF 的钽电容。

③ 对于抗干扰能力弱的、关断时电流变化大的元件和 RAM、ROM 等存储元件,应该在电源线(VCC)和地线之间接入去耦电容。

④ 电容的引线不要太长,特别是高频旁路电容不能带引线。

(3) 接地线的处理

在这种电路中地线的种类有很多(有系统地、屏蔽地、逻辑地、模拟地等),地线布局是否合理,将决定整个电路的抗干扰能力。在设计地线和接地点的时候,应该考虑以下问题:

① 逻辑地和模拟地要分开布线,不能合用,将它们各处的地线分别与相应的电源地线相连。设计时,模拟地线应尽量加粗,而且尽可能加大引出端的接地面积。一般来讲,对于输入

输出的模拟信号，与单片机电路之间最好通过光耦进行隔离。

② 在设计逻辑电路的印制板时，其地线应构成闭环形式，可提高电路的抗干扰能力。

③ 地线应尽量粗。如果地线很细，则地线电阻将会很大，造成接地电位随电流的变化而变化，致使信号电平不稳、电路的抗干扰能力下降。在布线允许的情况下，要保证主要地线的宽度至少在 2～3 mm，元件引脚上的接地线应在 1.5 mm 左右。

④ 要注意接地点的选择。当板上信号频率低于 1 MHz 时，由于布线和元件之间的电磁感应影响很小，而接地电路形成的环流对于干扰的影响较大，所以要采用一点接地，使其不形成回路。当电路板上信号频率高于 10 MHz 时，由于布线的电感效应明显，地线阻抗变得很大，此时接地电路形成的环流就不再是主要问题了，所以应采用多点接地，尽量降低地线阻抗。

⑤ 电源线的设置除了要根据电流的大小尽量加粗线宽度外，布线时还应使电源线、地线的走线方向与数据线的走线方向一致。此外，还可用地线将电路板的底层没有走线的地方铺满，这些方法都有助于增强电路的抗干扰能力。

(4) 数据线宽度选择

数据线的宽度也应尽可能宽，以减小阻抗。至少不小于 0.3 mm，如果采用 0.46 mm 或者更大，则更为理想。

(5) 电路板过孔的处理

由于电路板的过孔会带来约 10 pF 的电容效应，对于高频电路，将会引入太多的干扰，所以在布线的时候应该注意。

1.6　本章小结

本章主要介绍了单片机系统设计的内涵、单片机系统设计前的准备工作、系统设计的过程、软件程序编写的基本规范以及单片机控制设计的原则等。通过本章的学习，读者可基本了解单片机系统设计所应包含的内容，为后面具体的系统设计方法学习打下基础。

第2章

单片机系统设计工具介绍

在了解了一些单片机系统设计的基本知识后，读者可通过本章的学习了解单片机系统设计时所需使用的一些工具以及C51编程时的有益经验。通过本章的介绍，读者可以基本掌握利用Proteus仿真设计软件进行电路原理图绘制的方法及过程。同时，还介绍了一些C51编程的经验及解错方法供读者借鉴。

2.1 单片机设计仿真所需软件

单片机系统仿真设计时，常用到的是两类软件，即仿真软件和编程软件。编程软件主要完成程序语言的编写、编辑以及编译等任务，而仿真软件则完成虚拟硬件电路的设计、测试及运行等任务。

(1) 仿真软件

进行系统设计时可能需要进行多次系统器件的修改或功能的调整，若每次都用实际的硬件电路器件来制作，无疑会耗费过多的设计成本。如果在系统设计最终方案确定前采用仿真软件来代替实际硬件电路器件，那么，一方面由于仿真软件的便捷性将使得用户可以在更短时间内确定其设计方案；同时另一方面，由于仿真设计时所需使用的硬件较少，系统设计成本将会因此减少很多。

目前，常用的单片机仿真设计软件是Proteus软件，后面的章节将对其功能以及使用方法等进行较为详细的介绍。

(2) 编程软件

无论是单片机仿真设计还是实际硬件电路设计，单片机编程都是必不可少的。编程软件是完成程序的编写、编辑、编译、调试等功能的软件。目前的编程软件种类很多，比如WAVE、Keil等，常用的是Keil软件。在这类软件中，用户可实现汇编语言或是C语言程序语句的编写，同时进行源程序的编译、调试，直到得到最终系统所需的.hex文件。