**编 号：**

**审定成绩：**

**重庆邮电大学**

**毕业设计（论文）**

|  |  |
| --- | --- |
| **设计（论文）题目：** | **单片机洗衣机控制系统** |

|  |  |
| --- | --- |
| **学 院 名 称 ：** | **计算机科学与技术学院** |
| **学 生 姓 名 ：** | **刘 磊** |
| **专 业 ：** | **计算机科学与技术** |
| **班 级 ：** | **0490602** |
| **学 号 ：** | **06100406** |
| **指 导 教 师 ：** | **杜 欢** |
| **答辩组 负责人 ：** | **丰江帆** |

**填表时间：2010年6月**

**重庆邮电大学教务处制**

**摘 要**

洗衣机是现在的家居生活必不可少的一部分，给人们带来许多方便。波轮式套筒洗衣机，即全自动洗衣机，以其自动化和智能化，受到人们的普遍欢迎。洗衣机正朝着节能、大容量、健康化、微型化、品种多样化等几个方向不断发展和完善。随着单片机技术的不断成熟，单片机以其可靠性高、控制功能强、环境适应性好、体积小等优点在家用电器中得到日益广泛的应用，单片机和家用电器的结合，开创了家用电器自动化、智能化的新时代。

本文就是运用STC89C52RC单片机实现控制功能来设计全自动洗衣机。本文所设计的全自动洗衣机具有洗涤、清洗、脱水等一般洗衣机具有的功能，以及对洗涤模式、清洗模式进行设置的控制功能。本文在设计时采用LCD1602液晶显示模块作为全自动洗衣机的显示窗口，采用12MHZ的晶振作为系统的时钟信号。在硬件设计时，电路原理图和PCB板图清晰反映了全自动洗衣机控制系统的硬件实现。在软件设计时，使用C51编程实现全自动洗衣机的基本功能。在系统调试时，通过RS-232串口将程序烧制到单片机开发板上，硬件调试全自动洗衣机的每一个功能模块。本次设计的全自动洗衣机系统具有基本功能完善、控制可靠、性价比高的优点。

**【关键词】**全自动洗衣机 STC89C52RC单片机 LCD1602液晶显示

**ABSTRACT**

In the modern family life, washing machine, which brings us lots of convenience, has become part of our daily activity. Wave-wheel-sleeve washing machine, also known as automatic washing machine, is generally welcomed because of its high automatization and intelligent. At the same time, great improvement of washing machine will be made in terms of safety, energy saving, high-capacity, miniaturization, variety diversity and so forth. In addition, single-chip has infiltrated into every corner of our daily life with the maturity of SCM technology, which is increasingly widely used in the household appliances because of its high reliability, strong control, environmental adaptability and the advantages of small size, and the combination of SCM and household electrical appliances make it promising to expect the automatization and intelligent of household appliances.

The product of this article is trying to take advantage of the function of STC89C52RC SCM to design automatic washing machine. This design of the automatic washing machine has the functions as the general features, such as washing, rinsing, drying, washing mode setting, rinsing mode setting and so on. In this article, LCD1602 display module is used as the display window of the automatic washing machine, and the crystal oscillator of 12MHZ as clock signal of the whole system. As for the hardware design, the circuit schematics and PCB board diagram have clearly demonstrated the hardware implementation; and for the software design, C51 program is utilized to actualize the basic function of the automatic washing machine; and for the system test, with RS-232 serial port adheres to SCM developing board, the hardware is possible to debug the every functional module. The advantages of this automatic washing machine are multifunction, reliable control system and cost-effective.

**【Key words】**automatic washing machine STC89C52RC SCM LCD1602 display

**目 录**

[前 言 1](#_Toc263721696)

[第一章 绪 论 2](#_Toc263721697)

[第一节 洗衣机概述 2](#_Toc263721698)

[第二节 单片机发展史简介 3](#_Toc263721699)

[一、单片机的发展历程 3](#_Toc263721700)

[二、单片机的发展趋势 3](#_Toc263721701)

[第三节 单片机的特点及应用概述 4](#_Toc263721702)

[一、单片机的特点 5](#_Toc263721703)

[二、单片机的应用领域 5](#_Toc263721704)

[三、单片机的应用发展方向 6](#_Toc263721705)

[第四节 单片机产品 7](#_Toc263721706)

[第五节 论文的主要工作 8](#_Toc263721707)

[第六节 本章小结 8](#_Toc263721708)

[第二章 开发工具及环境 9](#_Toc263721709)

[第一节 Protel 99 SE概述 9](#_Toc263721710)

[一、Protel99SE的系统组成 9](#_Toc263721711)

[二、Protel 99 SE的功能特性 10](#_Toc263721712)

[三、Protel 99 SE的设计环境 10](#_Toc263721713)

[第二节 C51概述 11](#_Toc263721714)

[一、C语言与C51 11](#_Toc263721715)

[二、C51程序开发流程 11](#_Toc263721716)

[第三节 Keil uVision3概述 12](#_Toc263721717)

[一、Keil uVision3介绍 12](#_Toc263721718)

[二、Keil uVision3开发环境 14](#_Toc263721719)

[第四节 本章小结 15](#_Toc263721720)

[第三章 概要设计 16](#_Toc263721721)

[第一节 全自动洗衣机的功能设计 16](#_Toc263721722)

[一、全自动洗衣机的简单工作流程 16](#_Toc263721723)

[二、子功能模块划分和工作原理 17](#_Toc263721724)

[第二节 全自动洗衣机的控制面板设计 19](#_Toc263721725)

[第三节 本章小结 20](#_Toc263721726)

[第四章 硬件设计 21](#_Toc263721727)

[第一节 主控模块 21](#_Toc263721728)

[一、选用51单片机的优势 21](#_Toc263721729)

[二、STC89C52RC引脚功能说明 22](#_Toc263721730)

[三、时钟电路 24](#_Toc263721731)

[第二节 控制面板模块 25](#_Toc263721732)

[第三节 显示模块 25](#_Toc263721733)

[第四节 蜂鸣器模块 26](#_Toc263721734)

[第五节 全自动洗衣机详细电路设计 27](#_Toc263721735)

[一、电路原理图 27](#_Toc263721736)

[二、印刷电路板 28](#_Toc263721737)

[第六节 本章小结 30](#_Toc263721738)

[第五章 软件设计及代码实现 31](#_Toc263721739)

[第一节 程序变量及端口说明 31](#_Toc263721740)

[一、80C52端口地址定义 31](#_Toc263721741)

[二、程序变量定义和引脚设定 32](#_Toc263721742)

[第二节 主函数运行流程 34](#_Toc263721743)

[第三节 子模块程序流程设计 36](#_Toc263721744)

[一、定时模块 37](#_Toc263721745)

[二、LCD显示模块 38](#_Toc263721746)

[三、按键扫描模块 40](#_Toc263721747)

[四、进水模块 41](#_Toc263721748)

[五、洗涤模块 42](#_Toc263721749)

[六、排水模块 43](#_Toc263721750)

[七、脱水模块 43](#_Toc263721751)

[八、清洗模块 43](#_Toc263721752)

[九、报警模块 44](#_Toc263721753)

[十、外部中断服务模块 45](#_Toc263721754)

[第四节 本章小结 46](#_Toc263721755)

[第六章 系统测试 47](#_Toc263721756)

[结 论 50](#_Toc263721757)

[致 谢 51](#_Toc263721758)

[参考文献 52](#_Toc263721759)

[附 录 53](#_Toc263721760)

[一、英文原文 53](#_Toc263721761)

[二、英文翻译 57](#_Toc263721762)

[三、网络文件表 60](#_Toc263721763)

[四、源代码 62](#_Toc263721764)

# 前 言

随着单片机技术日新月异的发展，单片机以其可靠性高、控制功能强、环境适应性好、体积小等优点在家用电器中得到日益广泛的应用。用单片机取代传统家电中的机械控制部件，正在使传统的家用产品走向智能化。将单片机应用在家用电器中，是一个机电一体化的系统设计和制造技术过程。随着家电系统的日益复杂化，面对节能、环保、智能化等多方面要求，对单片机的功能、性能等也提出了更高要求，由此也推动了单片机的发展。内置A/D、D/A、看门狗、计时器、LED/LCD/VFD驱动器等较高集成度的单片机，以其优越的性能正在引导着家电行业不断提高。

随着现代科技的进步及互联网络的普及，自动化和智能化产品已经开始进入社会和家庭。在家电行业中，自动化和智能化也是其发展的必然趋势，消费者对智能家电的兴趣与日俱增。全自动洗衣机作为居家生活必不可少的家用电器，能实现高度自动化和智能化，对人们生活的意义非凡。

单片机技术的不断成熟，引领了各个领域自动化和智能化的跳跃性发展。在家用电器领域，从电饭褒、洗衣机、电冰箱、空调机、彩电、其他[音响](http://baike.baidu.com/view/68529.htm)视频器材、再到[电子秤](http://baike.baidu.com/view/287613.htm)量设备，都离不开单片机控制。

本次毕业设计的主要任务就是开发全自动洗衣机单片机控制系统，实现洗衣机基本的自动化。在设计的过程中，涉及到硬件和软件两方面的具体内容。在硬件方面，需要设计全自动洗衣机控制系统的硬件电路图和PCB板图。在软件方面，需要设计全自动洗衣机控制系统的每一个功能，达到很好地对洗衣机各种硬件的控制。

在具体设计的过程中，充分利用了已经成熟的各种软件和程序开发语言。在硬件设计中，Protel 99 SE给电路原理图、PCB板图的设计提供了很大的方便。在软件设计中，C51给程序的开发减少了许多困难和难题，使功能实现起来更容易，代码更清晰易懂。Keil uVision3作为C51开发的IDE，也给设计的实现提供了便捷。

单片机控制系统和洗衣机的结合，将更好的满足人们对洗衣机自动化、智能化、安全性、健康性和节能的要求。

# 第一章 绪 论

本文是基于单片机的全自动洗衣机控制系统的设计。全自动洗衣机作为居家生活中必不可少的电器，给我们带来很多方便。单片机以其独特的优势在控制领域起着不可或缺的重要作用。本文对洗衣机和单片机进行简单的介绍。

## 第一节 洗衣机概述

1858年，一个叫汉密尔顿•史密斯的美国人在匹兹堡制成了世界上第一台洗衣机。从此揭开了洗衣机发展的帷幕，1937年第一台自动洗衣机问世。工业化的加速，加速了世界各国研制的步伐。至1955年，在引进英国喷流式洗衣机的基础之上，日本研制出独具风格、并流行至今的波轮式洗衣机。至此，波轮式、滚筒式、搅拌式在洗衣机生产领域三分天下的局面初步形成。70年代后期，以微处理器控制的全自动洗衣机在日本问世，开创了洗衣机发展史的新阶段。之后，随着科技的进一步发展，滚筒洗衣机已经成了大家耳濡目染的产品。

在现在的洗衣机市场中主要有三种类型的洗衣机：滚筒式洗衣机、波轮式洗衣机、搅拌式洗衣机。滚筒式洗衣机流行于欧洲、南美等主要穿毛、棉为主的地区,几乎走入100%的家庭。波轮式洗衣机主要流行于日本、中国、东南亚等地。搅拌式洗衣机在北美普遍使用。波轮式套桶洗衣机也就是是现在常说的全自动洗衣机。

在城市洗衣机市场，海尔仍为行业老大。在不同的细分市场，内外资品牌表现各异。双缸洗衣机市场，外资品牌基本不涉足。波轮洗衣机市场内外资品牌竞争激烈，2009年零售量前十名中内外资品牌各有五个，五个内资品牌零售量份额合计为59.41%，其中海尔以34.47%占据第一，小天鹅份额为13.75%；五个外资品牌零售量份额合计 35.07%，其中松下以13.48%领军，紧随其后的是三洋（10.44%）。滚筒洗衣机市场外资品牌占优，2009年零售量前十名中外资品牌占七个，零售量份额达到67.06%，西门子以26.94%独领风骚，市场地位稳固；三个内资品牌零售量份额合计为31.12%，海尔零售量份额为 23.81%，作为国内企业唯一具有竞争力的品牌，在滚筒洗衣机市场里与外资品牌一较高下。

现在的洗衣机真的是多种多样，双缸洗衣机已经慢慢走出来历史的舞台，滚筒洗衣机和波轮洗衣机将成为市场的主流。现在的全自动洗衣机将在高度自动化、健康化、节能、大容量、微型化、品种多样化等几个方面得到不断地发展和完善。总之，将来的洗衣机会朝着使用更方便、更加节能、更加个性化的方向发展。

## 第二节 单片机发展史简介

单片机诞生于20世纪70年代末，经历了SCM、MCU、SOC三大阶段。单片机作为微型计算机的一个重要分支，应用面很广，发展很快。自单片机诞生至今，已发展为上百种系列的近千个机种。目前，单片机正朝着高性能和多品种方向发展趋势将是进一步向着CMOS化、低功耗、小体积、大容量、高性能、低价格和外围电路内装化等几个方面发展。

### 一、单片机的发展历程

①SCM即单片微型计算机（Single Chip Microcomputer）阶段，主要是寻求最佳的单片形态嵌入式系统的最佳体系结构。“创新模式”获得成功，奠定了SCM与通用计算机完全不同的发展道路。在开创嵌入式系统独立发展道路上，Intel公司功不可没。

②MCU即微控制器（Micro Controller Unit）阶段，主要的技术发展方向是：不断扩展满足嵌入式应用时，对象系统要求的各种外围电路与接口电路，突显其对象的智能化控制能力。它所涉及的领域都与对象系统相关，因此，发展MCU的重任不可避免地落在电气、电子技术厂家。从这一角度来看，Intel逐渐淡出MCU的发展也有其客观因素。在发展MCU方面，最著名的厂家当数Philips公司。Philips公司以其在嵌入式应用方面的巨大优势，将MCS-51从单片微型计算机迅速发展到微控制器。因此，当我们回顾嵌入式系统发展道路时，不要忘记Intel和Philips的历史功绩。

③单片机是嵌入式系统的独立发展之路，向MCU阶段发展的重要因素，就是寻求应用系统在芯片上的最大化解决；因此，专用单片机的发展自然形成了SOC化趋势。随着微电子技术、IC设计、EDA工具的发展，基于SOC的单片机应用系统设计会有较大的发展。因此，对单片机的理解可以从单片微型计算机、单片微控制器延伸到单片应用系统。

### 二、单片机的发展趋势

①CMOS化：近年，由于CHMOS技术的进小，大大地促进了单片机的CMOS化。CMOS芯片除了低功耗特性之外，还具有功耗的可控性，使单片机可以工作在功耗精细管理状态。这也是今后以80C51取代8051为标准MCU芯片的原因。因为单片机芯片多数是采用CMOS（金属栅氧化物）半导体工艺生产。CMOS电路的特点是低功耗、高密度、低速度、低价格。采用双极型半导体工艺的TTL电路速度快，但功耗和芯片面积较大。随着技术和工艺水平的提高，又出现了HMOS（高密度、高速度MOS）和CHMOS工艺。CHMOS和HMOS工艺的结合。目前生产的CHMOS电路已达到LSTTL的速度，传输延迟时间小于2ns，它的综合优势在于TTL电路。因而，在单片机领域CMOS正在逐渐取代TTL电路。

②低功耗化：单片机的功耗已从Ma级，甚至1uA以下；使用电压在3~6V之间，完全适应电池工作。低功耗化的效应不仅是功耗低，而且带来了产品的高可靠性、高抗干扰能力以及产品的便携化。

③低电压化：几乎所有的单片机都有WAIT、STOP等省电运行方式。允许使用的电压范围越来越宽，一般在3~6V范围内工作。低电压供电的单片机电源下限已可达1~2V。目前0.8V供电的单片机已经问世。

④低噪声与高可靠性：为提高单片机的抗电磁干扰能力，使产品能适应恶劣的工作环境，满足电磁兼容性方面更高标准的要求，各单片厂家在单片机内部电路中都采用了新的技术措施。

⑤大容量化：以往单片机内的ROM为1KB~4KB，RAM为64~128B。但在需要复杂控制的场合，该存储容量是不够的，必须进行外接扩充。为了适应这种领域的要求，须运用新的工艺，使片内存储器大容量化。目前，单片机内ROM最大可达64KB，RAM最大为2KB。

⑥高性能化：主要是指进一步改进CPU的性能，加快指令运算的速度和提高系统控制的可靠性。采用精简指令集（RISC）结构和流水线技术，可以大幅度提高运行速度。现指令速度最高者已达100MIPS（Million Instruction Per Seconds），并加强了位处理功能、中断和定时控制功能。这类单片机的运算速度比标准的单片机高出10倍以上。由于这类单片机有极高的指令速度，就可以用软件模拟其I/O功能，由此引入了虚拟外设的新概念。

⑦小容量、低价格化：与上述相反，以4位、8位机为中心的小容量、低价格化也是发展动向之一。这类单片机的用途是把以往用数字逻辑集成电路组成的控制电路单片化，可广泛用于家电产品。

⑧外围电路内装化：这也是单片机发展的主要方向。随着集成度的不断提高，有可能把众多的各种处围功能器件集成在片内。除了一般必须具有的CPU、ROM、RAM、定时器/计数器等以外，片内集成的部件还有模/数转换器、DMA控制器、声音发生器、监视定时器、液晶显示驱动器、彩色电视机和录像机用的锁相电路等。

⑨串行扩展技术：在很长一段时间里，通用型单片机通过三总线结构扩展外围器件成为单片机应用的主流结构。随着低价位OTP（One Time Programmable）及各种类型片内程序存储器的发展，加之处围接口不断进入片内，推动了单片机“单片”应用结构的发展。特别是IC、SPI等串行总线的引入，可以使单片机的引脚设计得更少，单片机系统结构更加简化及规范化。

## 第三节 单片机的特点及应用概述

为适应嵌入式应用的需求，单片微控制器应运而生，发展极其迅速。从70年代至今，单片机发展成为一个品种齐全，功能丰富的庞大家庭。单片机是微型计算机的一个分支，是在一块芯片上集成了CPU、RAM、ROM存储器、I／O接口等而构成的微型计算机。因为它主要应用于工业测控领域，因此单片机在出现时，英特尔公司就给单片机取名为嵌入式微控制器。

### 一、单片机的特点

单片机是以工业测控对象、环境、接口特点出发向着增强控制功能，提高工业环境下的可靠性方向发展。主要特点如下：

①种类多，型号全：很多单片机厂家逐年扩大适应各种需要，有针对性地推出一系列型号产品，使系统开发工程师有很大的选择余地。大部分产品有较好的兼容性，保证了已开发产品能顺利移植，较容易地使产品进行升级换代。

②提高性能，扩大容量，性能价格比高：集成度已经达到300万个晶体管以上，总线速度达到数十微妙到几百纳秒，指令执行周期已经达到几微妙到数十纳秒，以往片外XRAM现已在物理上存入片内，ROM容量已经扩充到32K，64K，128K以致更大的空间，价格从几百到几元不等。

③增加控制功能，向真正意义上的“单片”机发展：把原本是外围接口芯片的功能集成到一块芯片内，在一片芯片中构造了一个完整的功能强大的微处理应用系统。

④低电压、低功耗：现在新型单片机的功耗越来越小，供电电压从5v降低到了3.2V，甚至1V，工作电流从毫安降到微安级。特别是很多单片机都设置了多种工作方式，这些工作方式包括等待、暂停、睡眠、空闲、节电等。

⑤C语言开发环境，友好的人机互交环境：大多数单片机都提供基于C语言开发平台，并提供大量的函数供使用，这使产品的开发周期、代码可读性、可移植性都大为提高。

### 二、单片机的应用领域

目前单片机渗透到我们生活的各个领域，几乎很难找到哪个领域没有单片机的踪迹。[导弹](http://baike.baidu.com/view/35448.htm)的导航装置，飞机上各种仪表的控制，计算机的网络通讯与数据传输，工业自动化过程的实时控制和数据处理，广泛使用的各种智能[IC卡](http://baike.baidu.com/view/4189.htm)，民用豪华轿车的安全保障系统，[录像机](http://baike.baidu.com/view/111680.htm)、[摄像机](http://baike.baidu.com/view/45678.htm)、全自动洗衣机的控制，以及程控玩具、电子宠物等等，这些都离不开单片机。更不用说自动控制领域的机器人、智能仪表、医疗器械了。因此，单片机的学习、开发与应用将造就一批计算机应用与智能化控制的科学家、工程师。

单片机广泛应用于仪器仪表、家用电器、医用设备、航空航天、专用设备的智能化管理及过程控制等领域，大致可分如下几个范畴：

①在[智能仪器](http://baike.baidu.com/view/1540405.htm)仪表上的应用

单片机具有体积小、功耗低、控制功能强、扩展灵活、微型化和使用方便等优点，广泛应用于仪器仪表中，结合不同类型的[传感器](http://baike.baidu.com/view/16431.htm)，可实现诸如电压、[功率](http://baike.baidu.com/view/44147.htm)、频率、湿度、温度、流量、速度、厚度、角度、长度、硬度、元素、压力等物理量的测量。采用单片机控制使得仪器仪表数字化、智能化、微型化，且功能比起采用电子或[数字电路](http://baike.baidu.com/view/635019.htm)更加强大。例如精密的测量设备（[功率计](http://baike.baidu.com/view/1608633.htm)，[示波器](http://baike.baidu.com/view/130973.htm)，各种分析仪）。

②在工业控制中的应用

用单片机可以构成形式多样的控制系统、数据采集系统。例如工厂流水线的智能化管理，电梯智能化控制、各种报警系统，与计算机联网构成二级控制系统等。

③在家用电器中的应用

可以这样说，现在的家用电器基本上都采用了单片机控制，从电饭褒、洗衣机、电冰箱、空调机、彩电、其他[音响](http://baike.baidu.com/view/68529.htm)视频器材、再到[电子秤](http://baike.baidu.com/view/287613.htm)量设备，五花八门，无所不在。

④在计算机网络和通信领域中的应用

现代的单片机普遍具备通信接口，可以很方便地与计算机进行数据通信，为在计算机网络和通信设备间的应用提供了极好的物质条件，现在的通信设备基本上都实现了单片机智能控制，从手机，电话机、小型[程控交换机](http://baike.baidu.com/view/84642.htm)、楼宇自动通信呼叫系统、列车无线通信、再到日常工作中随处可见的移动电话，集群移动通信，[无线电](http://baike.baidu.com/view/19599.htm)对讲机等。

⑤单片机在医用设备领域中的应用

单片机在医用设备中的用途亦相当广泛，例如医用呼吸机，各种分析仪，监护仪，超声诊断设备及病床呼叫系统等等。

⑥在各种大型电器中的模块化应用

某些专用单片机设计用于实现特定功能，从而在各种电路中进行模块化应用，而不要求使用人员了解其内部结构。如音乐集成单片机，看似简单的功能，微缩在电子芯片中（有别于磁带机的原理），就需要复杂的类似于计算机的原理。如：音乐信号以数字的形式存于存储器中（类似于ROM），由微控制器读出，转化为模拟音乐电信号（类似于声卡）。在大型电路中，这种模块化应用极大地缩小了体积，简化了电路，降低了损坏、错误率，也方便于更换。

⑦单片机在汽车设备领域中的应用

单片机在汽车电子中的应用非常广泛，例如汽车中的发动机控制器，基于CAN总线的汽车发动机智能电子控制器，GPS导航系统，ABS防抱死系统，制动系统等等。

此外，单片机在工商，金融，科研、教育，国防航空航天等领域都有着十分广泛的用途。

### 三、单片机的应用发展方向

①使用寿命长：这里所说的长寿命，一方面指用单片机开发的产品可以稳定可靠地工作十年、二十年，另一方面是指与微处理器相比的长寿命。随着半导体技术的飞速发展，MPU更新换代的速度约来约快。可以预见，一些成功上市的相对年轻的CPU核心，也会随着I/O功能模块的不断丰富，有着相当长的生存周期。新的CPU类型的加盟，使单片机队伍不断壮大，给用户带来了更多的选择余地。8位、16位、32位单片机共同发展是当前单片机发展的另一个动向之一。长期以来，单片机技术的发展是以8位机为主的。随着移动通讯、网络技术、多媒体技术等高科技产品进入家庭，32位单片机应用得到了长足的发展。以Mororola68K为CPU的32位单片机97年的销售量达到了八千万枚。过去认为由于8位单片机功能越来越强，32位机越来越便宜，使16位机单片机生存空间有限，而16位单片机的发展无论从品种和产量方面，近年来都有较大幅度的增长。

②速度越来越快：MUP发展中表现出来的速度越来越快是以时钟频率越来越高为标志的。而单片机则有所不同，为提高单片机抗干扰能力，降低噪音，降低时钟频率而不牺牲运算速度是单片机技术发展之追求。改善单片机的内部时序，在不提高时钟频率的条件下，使运算速度提高了很多。

③低噪声和高可靠性技术：在单片机应用中，可靠性是首要因素，位了扩大单片机的应用范围和领域，提高单片机自身的可靠性是一种有效的方法。今年来，单片机的生产厂家喜欢在单片机设计上采用了各种提高可靠性的新技术，这些新技术表现在如下几点：首先，EFT技术。EFT技术是一种抗干扰技术，它是指在振荡电路的正弦信号受到外界干扰时，其波形上会迭加各种毛刺信号，人使用Schmitt电路对其整形，则毛刺会成为触发信号干扰正常的时钟，在交替使用Schmitt电路和RC滤波电路时，就可以消除这些毛刺另其作用失效，从而保证系统的时钟信号正常工作。这样，就提高了单片机工作的可靠性。其次，低噪声布线技术及驱动技术。在传统的单片机中，电源及地线是在集成电路外壳的对称引脚上，一般是在左上、右下或右上左下的两块对称点上。这样，就使电源噪声穿过整块芯片，对单片机的内部电路造成干扰。现在，很多单片机都把地线和电源引脚安排在两条相邻的引脚上。这样，不仅降低了穿过整个芯片的电流，另外还在印制电路板上容易布置去耦电容，从而降低系统的噪声。

④OTP和掩膜：OTP是一次性写入的单片机。过去认为一个单片机产品的成熟是以投产掩膜型单片机为标志的。由于掩膜时需要一定的生产周期，而OTP型单片机价格不断下降，使得近年来直接使用OTP完成最终产品制造更为流行。它较之掩膜具有生产周期短、风险小的特点。近年来，OTP型单片机需量大幅度上扬，为适应这种需求许多单片机都采用了在编程技术（In System Programming）。为编程的OTP芯片可以采用裸片Bonding技术或表面贴技术，先焊在印刷版上，然后通过单片机上引出的编程线、串行数据、时钟线等对单片机编程。解决了在批量写OTP芯片时容易出现的芯片写入器接触不好的问题，使得OTP的裸片得到广泛使用，降低了产品的成本。编程线与I/O线共用，不增加单片机的额外引脚。

## 第四节 单片机产品

国内常用的单片机芯片系列主要有STC单片机、PIC单片机、EMC单片机、ATMEL单片机(51单片机)、PHLIPIS 51PLC系列单片机(51单片机)、HOLTEK单片机、TI公司单片机(51单片机)和松翰单片机（SONIX）等。

国外常用的单片机芯片系列主要有MOTOROLA单片机、Microchip单片机、SCENIX单片机、NEC单片机、东芝单片机、富士通单片机、Epson单片机、8051单片机、ZILOG单片机、NS单片机、三星单片机和华邦单片机等。

## 第五节 论文的主要工作

本文主要是对全自动洗衣机控制系统设计的阐述， 整个全自动洗衣机的控制系统搭建在一块51单片机开发板上，其设计流程包括以下几个阶段：

①对全自动洗衣机进行功能分析，划分功能子模块，对各子模块的实现原理进行详细阐述。

②采用STC89C52RC单片机芯片，利用Protel 99 SE设计全自动洗衣机的硬件电路图，并生成PCB板图，实现建立在51单片机开发板上的硬件电路。

③在Keil uVision3的开发环境中，用C51开发语言，实现全自动洗衣机控制系统的软件部分。

④利用STC\_ISP\_V480，将程序烧制到51单片机开发板上，调制、测试该全自动洗衣机控制系统的控制功能。

## 第六节 本章小结

本章主要介绍了在设计单片机洗衣机控制系统的过程中要涉及到的两方面的内容，即洗衣机有关知识和单片机的有关知识。对单片机的介绍主要从发展史和应用领域两个方面给予简单阐述。借此让读者能对单片机有一定认识。

# 第二章 开发工具及环境

在进行单片机应用系统的开发过程中，利用Protel能够对硬件电路进行方便的设计，C已经成为单片机开发中很重要的一种语言，本次设计采用C51进行51单片机应用系统的开发，开发环境选择Keil uVision3。

## 第一节 Protel 99 SE概述

Protel99SE是应用于Windows9X/2000/NT操作系统下的[EDA](http://baike.baidu.com/view/5822.htm)设计软件，采用设计库管理模式，可以进行联网设计，具有很强的数据交换能力和开放性及[3D](http://baike.baidu.com/view/4376.htm)模拟功能，是一个32位的设计软件，可以完成电路原理图设计，印制电路板设计和可编程逻辑器件设计等工作，可以设计32个信号层，16个电源--地层和16个机加工层。

### 一、Protel99SE的系统组成

按照系统功能来划分，Protel99SE主要包含以下两大部分和六个功能模块。

**1、电路工程设计部分**

①电路原理设计部分（Advanced Schematic 99）：电路原理图设计部分包括电路图编辑器（简称SCH编辑器）、电路图零件库编辑器（简称Schlib编辑器）和各种文本编辑器。本系统的主要功能是：绘制、修改和编辑电路原理图；更新和修改电路图零件库；查看和编辑有关电路图和零件库的各种报表。

②印刷电路板设计系统（Advanced PCB 99）：印刷电路板设计系统包括印刷电路板编辑器（简称PCB编辑器）、零件封装编辑器（简称PCBLib编辑器）和电路板组件管理器。本系统的主要功能是：绘制、修改和编辑电路板；更新和修改零件封装；管理电路板组件。

③自动布线系统（Advanced Route 99）：本系统包含一个基于形状（Shape-based）的无栅格自动布线器，用于印刷电路板的自动布线，以实现PCB设计的自动化。

**2、电路仿真与PLD部分**

①电路模拟仿真系统（Advanced SIM 99）：电路模拟仿真系统包含一个数字/模拟信号仿真器，可提供连续的数字信号和模拟信号，以便对电路原理图进行信号模拟仿真，从而验证其正确性和可行性。

②可编程逻辑设计系统（Advanced PLD 99）：可编程逻辑设计系统包含一个有语法功能的文本编辑器和一个波形编辑器（Waveform）。本系统的主要功能是；对逻辑电路进行分析、综合；观察信号的波形。利用PLD系统可以最大限度的精简逻辑部件，使数字电路设计达到最简化。

③高级信号完整性分析系统（Advanced Integrity 99）：信号完整性分析系统提供了一个精确的信号完整性模拟器，可用来分析PCB设计、检查电路设计参数、实验超调量、阻抗和信号谐波要求等。

### 二、Protel 99 SE的功能特性

①开放式集成化的设计管理体系

②超强功能的、修改与编辑功能

③强大的设计自动化功能

### 三、Protel 99 SE的设计环境

Protel 99 SE操作界面符合Windows的标准风格。如图2.1所示，为一个设计项目的设计窗口图。

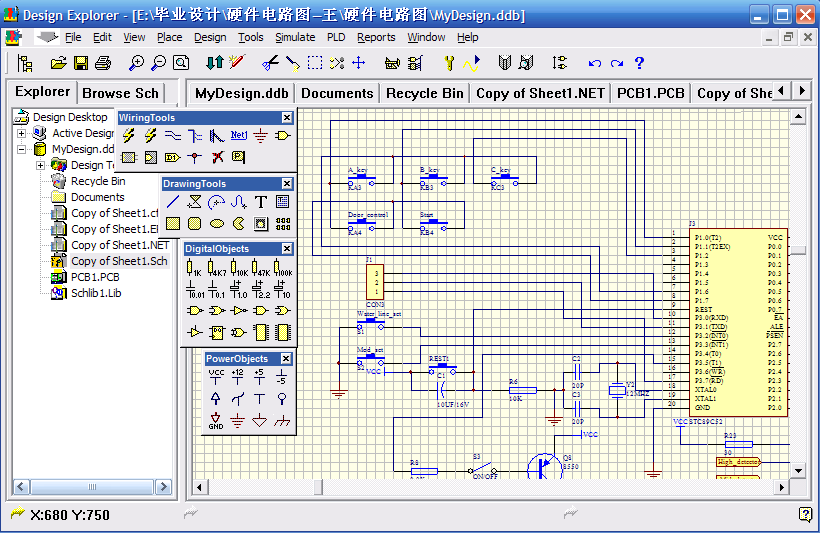


图2.1 Protel 99 SE工作环境

## 第二节 C51概述

C51是C语言的扩展，除了遵循一般C语言的规则外，还有其自身的特点。

### 一、C语言与C51

**1、C语言与C51**

C语言是一种使用非常方便的高级语言。所以，在单片机的开发应用中，除了使用汇编语言外，也逐渐引入了C语言。早在1985年便出现了80C51单片机的C语言，简称C51。

单片机C语言除了遵循一般C语言的规则外，还有其自身的特点。例如，增加了位变量数据类型（如bit、sbit）、中断服务函数（如interrupt n），对80C51单片机特殊功能寄存器的定义是C51所特有的。是对标准C语言的扩展。

**2、C语言编程的优点**

使用C语言与使用汇编语言相比，具有如下优点。

①不需要掌握8051系列单片机的指令集，只要了解单片机内部特殊功能寄存器的用途即可。

②编程人员不必考虑寄存器的分配和寻址方式等细节，这些都由编译器自动进行管理。

③利于结构化编程，易于维护。

④可使用与人的思维更相近的关键字和操作函数。

⑤由于C语言可实现模块化编程技术，并与硬件无关，这样可以将大量的例程直接调用，节省开发时间。

⑥程序的开发和调试的时间大大缩短，提高了效率。

**3、C语言和汇编语言混合编程**

C语言编程有明显的优势，但在有些场合下不适合使用C语言编程，例如，在对时序要求比较严格的场合用C语言编程比较困难，这时可以使用汇编语言。C语言可以和汇编语言混合编程。在实际编程中常常以C语言为主，汇编语言为辅，充分发挥各自的优势。

### 二、C51程序开发流程



图2.2 C51程序开发流程

C51程序开发流程与使用汇编语言基本相同，其流程图如上所示。但区别有以下两点。

①单片机若使用C语言开发，必须选择具有C语言编译能力的开发集成系统软件，如Keil C51单片机开发集成系统，既可以完成C语言的程序开发，也可以进行汇编语言的程序开发。

②编写C语言源程序时，文件名的后缀为.C。Keil C51通过检测源程序后缀来确定使用不同的编译器。如果检测到源程序后缀是.ASM，就自动使用A51汇编器；如果检测到源程序后缀是.C，就自动使用C51编译器。

## 第三节 Keil uVision3概述

Keil uVision3是可进行C51开发的集成开发环境，支持多种8051单片机。

### 一、Keil uVision3介绍

Keil uVision3是Keil Software公司推出的一款可用于多种8051MCU的集成开发环境(IDE)，除增加了源代码、功能导航器、模板编辑以及改进的搜索功能外，uVision3还提供了一个配置向导功能，加速了启动代码和配置文件的生成。此外其内置的仿真器可模拟目标MCU，包括指令集、片上外围设备及外部信号等。uVision3提供逻辑分析器，可监控基于MCU的I/O引脚和外设状态变化下的程序变量。uVision3提供对多种最新的8051类微处理器的支持。

如图2.3 Keil uVision3工作集的框图形象地阐述了单片机软件系统的开发流程：



图2.3 Keil uVision3工作集

①uVision3 IDE

uVision3集成开发环境集成了一个项目管理器，一个功能丰富、有错误提示的编辑器，

以及设置选项，生成工具，在线帮助。利用 uVision3创建你的源代码并把它们组织到一个能确定你的目标应用的项目中去。uVision3自动编译，汇编，连接你的嵌入式应用，并为你

的开发提供一个单一的焦点。

②C51编译器和A51汇编器

源代码由 uVision3 IDE创建，并被C51编译或A51汇编。编译器和汇编器从源代码生成可重定位的目标文件。

Keil C51编译器完全遵照ANSI C语言标准，支持C语言的所有标准特性。另外，直接支持8051结构的几个特性被添加到里面。

Keil A51宏汇编器能够支持8051及其派生系列的全部指令集。

③LIB51库管理器

LIB51库管理器允许你从由编译器或汇编器生成的目标文件创建目标库。库是一种被特

别地组织过并在以后可以被连接重用的对象模块。当连接器处理一个库时，仅仅那些被使用

的目标模块才被真正使用。

④BL51连接器/定位器

BL51连接器/定位器利用从库中提取的目标模块和由编译器或汇编器生成的目标模块创建一个绝对地址的目标模块。一个绝对地址目标模块或文件包含不可重定位的代码和数据。所有的代码和数据被安置在固定的存储器单元中。此绝对地址目标文件可以用来：写入EPROM 或其它存储器件，由uVision2调试器使用来模拟和调试，由仿真器用来测试程序。

⑤uVision2调试器

uVision2源代码级调试器是一个理想地快速，可靠的程序调试器。此调试器包含一个高速模拟器，能够让你模拟整个8051系统，包括片上外围器件和外部硬件。当你从器件库中选择器件时，这个器件的特性将自动配置。

uVision2调试器为你在实际目标板上测试你的程序提供了几种方法：安装MON51目标监控器到你的目标系统并且通过Monitor-51接口下载你的程序，利用高级GDI（AGDI）接口，把uVision2调试器绑定到你的目标系统。

⑥Monitor-51

uVision2调试器支持用Monitor-51进行目标板调试。此监控程序驻留在你的目标板的存储器里，它利用串口和uVision2调试器进行通信。利用Monitor-51，uVision2调试器可

以对你的目标硬件实行源代码级的调试。

⑦RTX51实时操作系统

RTX51实时操作系统是一个针对8051系列的多任务核RTX51实时内核从本质上简化了对实时事件反应速度要求高的复杂应用系统的设计，编程和调试。RTX51实时内核是完全集成到C51编译器中的，从而方便使用。任务描述表和操作系统的连接由 BL51 连接器/定位器自动控制。

### 二、Keil uVision3开发环境

Keil uVision3操作界面符合Windows的标准风格。如图2.4所示，为一个开发项目的设计窗口图。

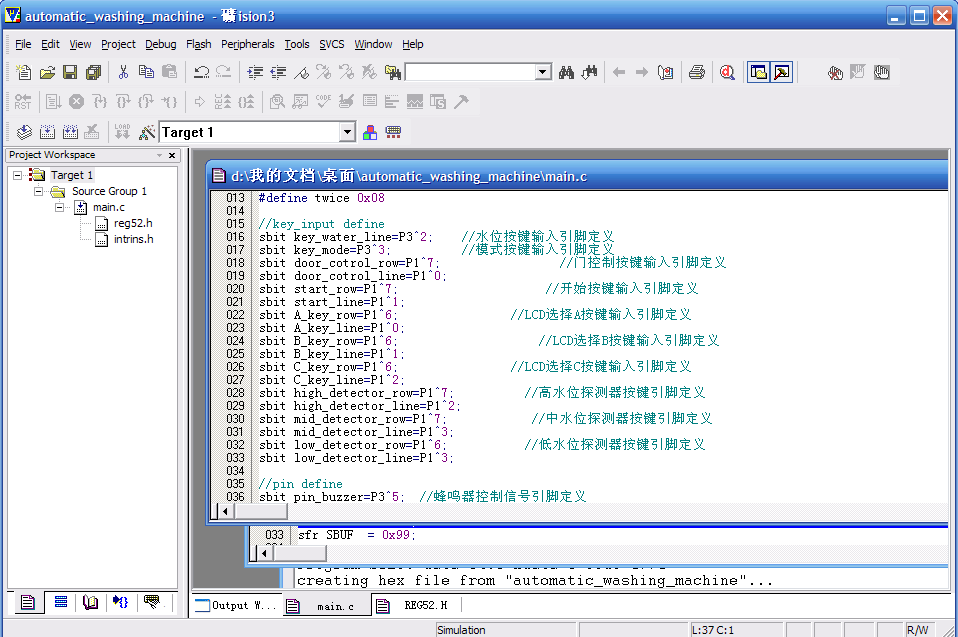


图2.4 Keil uVision3的开发环境

## 第四节 本章小结

本章主要介绍了在全自动洗衣机单片机控制系统的设计和实现过程中，用到的软件和实现语言。Protel99SE作为硬件电路的设计工具，其方便性和功能完善性，已经在业界得到了证实。C51在C语言的基础上通过一些扩展，能够高效地完成底层平台，即硬件方面的功能实现。Keil uVision也已成为被程序员所认可的IDE，成为单片机开发的重要选择。本次设计充分运用了这些成熟的知识和软件，为系统的成功开发奠定了基础。

# 第三章 概要设计

波轮式套筒洗衣机就是通常所说的全自动洗衣机，本章通过分析全自动洗衣机的工作流程，将全自动洗衣机的功能分解为一些功能子模块，对子模块的工作原理的阐述，来明确在硬件、软件实现时，需要达到的目标。

## 第一节 全自动洗衣机的功能设计

### 一、全自动洗衣机的简单工作流程



图3.1 全自动洗衣机的工作流程[7]

对图3.1全自动洗衣机的工作流程详细说明如下：

①按下电源按钮、开始按钮，并合上洗衣机上盖，开始运行。

②按照水位选择，开始进水直到（高、中、低）水位，关水，2秒后开始洗涤。

③洗涤时，正转28秒，停2秒，然后反转28秒，停2秒；正反循环（强洗20次，中洗10次，弱洗5次），然后开始排水。

④排空后脱水30秒。

⑤开始清洗，按照水位选择，开始进水直到（高、中、低）水位，关水, 2秒后开始洗涤。

⑥洗涤时正反转循环5次。

⑦循环完毕，开始排水。

⑧按照清洗选择，如果为一清，直接执行⑨；如果为二清，循环执行⑤-⑦。

⑨开始脱水120秒。

⑩脱水完成，报警3秒并自动停机。

### 二、子功能模块划分和工作原理

全自洗衣机的洗衣桶（外桶）和脱水桶（内桶）是以同一心安放的，内桶可以旋转，作为脱水用。内桶的周围有许多小孔，使内桶和外桶的水流相通，洗衣机的进水和排水分别由进水电磁阀和排水电磁阀来执行。进水时通过控制系统将进水电磁阀打开，经进水管将水注入到外桶。排水时，通过控制系统将排水电磁阀打开，将水由外桶排到机外。洗涤正转、反转由洗涤电动机驱动波盘的正、反转来实现，此时脱水桶并不旋转。脱水时，控制系统将离合器合上，由洗涤电动机带动内桶正转进行甩干。由此，可以将全自动洗衣机分为洗涤系统、脱水系统、进水系统、排水系统、传动系统、报警系统六个字模块。

**1、洗涤系统**

洗衣机的洗涤过程主要是在机械产生的排渗、冲刷等机械作用和洗涤剂的润湿、分散作用下，将污垢拉入水中来实现洗净的目的。

首先充满于波轮叶片间的洗涤液，在离心力的作用下被高速甩向桶壁，并沿桶壁上升。在波轮中心处，因甩出液体而形成低压区，又使得洗涤液流回波轮附近。这样，在波轮附近形成了以波轮轴线为中心的涡流。衣物在涡流的作用下，作螺旋式回转，吸入中心后又被甩向桶壁，与桶壁发生摩擦。又由于波轮中心是低压区，衣物易被吸在波轮附近，不断地与波轮发生摩擦，如同人工揉搓衣物，污垢被迫脱离衣物。

其次，当衣物被放进洗涤液之后，由于惯性作用运动缓慢，在水流与衣物之间存在着速度差，使得两者发生相对运动，水流与衣物便发生相对摩擦，这种水流冲刷力同样有助于污垢离开衣物。

再次由于洗衣涌形状的不规则，当旋转着的水流碰到桶壁后，其速度和方向都发生了改变，形成湍流。在湍流的作用下，衣物做无规则地运动并翻滚，其纤维不断被弯曲、绞纽扣拉长，衣物相互相摩擦，增大了洗涤的有效面积，提高衣物的洗净的均匀性。

全自动洗衣机波轮安装在洗涤桶的正中，托盘连接着盛水桶。洗涤系统的传动部分由风叶、皮带、皮带轮、和洗衣轴体组成。全自动洗衣机的洗涤桶其上部略大，下部略小，呈圆锥形。为了保证洗涤效果，洗涤桶的内壁上必需设计成凸形来增大摩擦力，达到满意的洗涤效果，当衣物与洗涤桶接触时，桶壁就产生像搓板那样的洗涤作用，而且能增强涡旋作用，提高洗涤率。

**2、脱水系统**

全自动洗衣机脱水时，控制系统将离合器合上，由洗涤电动机带动脱水桶正转进行甩干。脱水桶既要保证能离心脱水，又要能容纳洗涤的衣物。脱水系统由脱水桶、脱水定时器、控制开关、电动机、制动机构等组成。洗衣机具有盖的带锁装置，该锁定装置，包括洗衣机盖板、洗衣机箱体，还包括控制开关，与控制脱水的开关联动；电磁铁，与控制开关连接；洗衣机盖板翻口端内面在对应洗衣机箱体内壁处设有凹缘，凹缘上设有锁孔，洗衣机箱体上端设有所述电磁铁，电磁铁衔铁的伸缩口位于洗衣机箱体内壁，而且电磁铁通电后其衔铁伸出端恰好位于盖好的洗衣机盖板凹缘的锁孔内。由于电磁铁的控制开关与洗衣机控制脱水的开关联动，使洗衣机在脱水时电磁铁的衔铁能伸出，而且正好锁住洗衣机盖板凹缘的锁孔，使用户在脱水时不能打开洗衣机盖板，从而确保了洗衣机脱水时的操作安全。

**3、进水系统**

全自动洗衣机的进水系统采用水位压力开关和进水阀，由程序控制器调节。设有溢水口，其位置在盛水桶上口部。清洗时，它能让洗涤液中的泡沫和污水溢出，有利于清清。

全自动洗衣机水位开关一般有三档水位控制,即低水位、中水位、高水位控制。当进水阀注水，内桶水位增高到预选水位时，导通橡皮气膜受到内部空气的压缩而被顶出，中触片上跳，与上触片闭合。此时主电机导通，进水阀断开（开始时刻中触片与下触片闭合），并开始洗衣。当旋钮旋至低水位时，凸轮转动，但曲率半径较小。通过一定的机构，橡皮气膜的压簧被压缩而产生压力P1，压迫气膜。当桶内水量达到30L时，软管内的空气被压缩，产生空气压力F1，当F1>P1时,中触片上跳,与上触片闭合,主电动机动作,进水阀关闭。

**4、排水系统**

全自动洗衣机的排水系统由程序来控制排水电磁阀，牵引排水阀。排水阀主要同阀盖、阀芯弹簧、阀芯拉簧，橡皮阀和阀体组成。

排水电磁铁主要用来控制自动型洗衣机排水阀的开闭,在套桶式自动型洗衣机中同起到改变减速离合大的洗涤、脱水状态、排水电磁铁主要由线圈、磁轭、静铁芯、衔铁和短路铜环等组成。

**5、传动系统**

全自动洗衣机的传动系统设在洗衣机脱水桶的底部,主要由波轮、脱水桶、离合器、传动带、电动机及电磁阀组成。离合器是内外轴复合为一体的结构。离合器的内轴（洗涤轴），一端固定波轮，另一端固定离合套，离合套上固定大带轮，离合器外轴（离心轴）的一端固定离心桶（脱水桶），另一端通过抱簧与离合套连接。内外桶的联动或分动（即实现脱水或洗涤），是由拨叉控制抱簧和刹车盘来实现的。

离合器轴包括脱水轴和洗涤轴。洗涤或漂洗时，牵引电机处在断电状态，制动杆将制动带收紧，对脱水轴制动。棘爪将棘轮拔动一定角度使方丝离合簧旋松，所以电机做正转或反转转矩都不能传递给脱水轴，而只是经过行星减速器再输出给波轮，实现洗涤运转。

脱水时，牵引电机是通电状态，拉动制动杆使得制动带松开，棘轮和方丝离合器处于自由状态，大皮带轮顺时针旋转将方丝离合簧旋紧，使输入转矩有效地传递给脱水轴，实现高速脱水运转。如果大皮带轮逆时针转动，则方丝离合簧被拨松而不能传递转矩给脱水轴，所以脱水只能顺时针单向转动。

**6、报警系统**

洗涤完毕报警，主要还是由软件编程实现对蜂鸣器的控制，程序主要是运用定时器的定时功能实现。

## 第二节 全自动洗衣机的控制面板设计

全自动洗衣机控制面板[8][9]由LCD显示器和电源、开始、水位、模式、选择（A、B、C）等按钮组成。控制面板如图所示：



图3.2 全自动洗衣机的控制面板

按钮说明如下：

①电源：是全自动洗衣机的一般性电源开关，控制洗衣机的电源开与关，是总开关。

②开始：控制洗衣机洗涤开始。

③水位：是全自动洗衣机的设置按钮，进入系统的水位设置模式。如图5所示，可以通过选择按钮（A、B、C）设置水位为高水位、中水位或低水位。默认为中水位。



图3.3 水位设定LCD显示

④模式：是全自动洗衣机的设置按钮，进入系统的洗涤和清洗的模式设定。如图6所示，可以通过选择按钮（A、B、C）设置洗涤模式为强洗、中洗或弱洗。默认为强洗。如图7所示，可以通过选择按钮（A、B）设置清洗模式为一清或两清。默认为一清。



图3.4 洗涤模式设定LCD显示



图3.5 清洗模式设定LCD显示

⑤A、B、C按钮：作为全自动洗衣机在模式设定时的选择按钮。

⑥门控制：即全自动洗衣机的上翻盖，翻口端内面在对应洗衣机箱体内壁处设有凹缘，凹缘上设有锁孔，洗衣机箱体上端设有电磁铁，电磁铁衔铁的伸缩口位于洗衣机箱体内壁，而且电磁铁通电后其衔铁伸出端恰好位于盖好的洗衣机盖板凹缘的锁孔内。

## 第三节 本章小结

本章详细阐述了全自动洗衣机的工作原理，分析了其简单的工作流程，并设计了控制面板。全自动洗衣机的各个子模块，如洗涤模块、脱水模块、进水模块、排水模块、报警模块等，在控制上的需求即为下面两章软硬件设计所要达到的目标。

# 第四章 硬件设计

全自动洗衣机硬件系统框图如图4.1所示，系统核心是由51单片机充当的主控模块，主控模块处理控制面板发出的控制命令和水位探测器反馈的信号，然后对显示模块、进水电磁阀、洗涤接触器、排水离合器、脱水离合器、蜂鸣器等发出控制信息和数据信息。



图4.1 全自动洗衣机的硬件系统框图

## 第一节 主控模块

本系统采用STC公司生产的STC89C52RC单片机。STC89C52RC是宏晶科技推出的抗干扰、高速、低功耗的单片机，指令完全兼容传统的8051单片机，工作电压为5V，有8KB的Flash程序存储器，512B的RAM数据存储器，功能强大。STC89C52RC单片机可为你提供许多高性价的应用场合，可灵活的应用于各种控制领域。

### 一、选用51单片机的优势

51系列单片机优点之一是它从内部的硬件到软件有一套完整的按位操作系统，即位处理器，或布尔处理器。它的处理对象不是字或字节而是位。它不仅能对片内某些特殊功能寄存器的某位进行处理，如传送、置位、清零、测试等，还能进行按位的逻辑运算，其功能十分完备，使用起来得心应手。虽然其他种类的单片机也具有按位处理功能，但能进行按位逻辑运算的实属少见。51系列在片内RAM区间还特别开辟了一个双重功能的地址区间，十六个字节，单元地址20H～2FH，它既可作字节处理，也可作位处理，使用极为灵活。这一功能无疑给使用者提供了极大的方便，因为一个较复杂的程序在运行过程中会遇到很多分支，因而需建立很多标志位，在运行过程中，需要对有关的标志位进行置位、清零或检测，以确定程序的运行方向。而实施这一处理（包括位功能），只需用一条位操作指令即可。

51系列单片机的另一个优点是乘法和除法指令，这给编程也带来了便利。八位除以八位的除法指令，商为八位，精度嫌不够，用得不多。而八位乘八位的乘法指令，其积为十六位，精度还是能满足要求的，用的较多。作乘法时，只需一条指令就行了。很多的八位单片机都不具备乘法功能，作乘法时还得编上一段子程序调用，十分不便。

51系列单片机中，还有一条二进制十进制调整指令，能将二进制变为BCD码，这对于十进制的计量十分方便。而在其他的单片机中，则也需调用专用的子程序才行。

Intel公司51系列的典型产品是8051，片内有4K字节的一次性程序存储器（OTP）。Atmel公司就将其改为电可改写的闪速存储器（Flash），容许改写1000次以上，这给编程和调试带来极大的便利，其产品AT89C51、AT89C52…等成为了当今最流行的八位单片机[13]。

51系列的I/O脚的设置和使用非常简单，当该引脚作为输入脚使用时，只须将该引脚设置为高电平。当该引脚作为输出脚使用时，则为高电平或低电平均可。低电平时，吸入电流可达20mA，具有一定的驱动能力；而为高电平时，输出电流仅数10μA甚至更小，基本上没有驱动能力。

### 二、STC89C52RC引脚功能说明

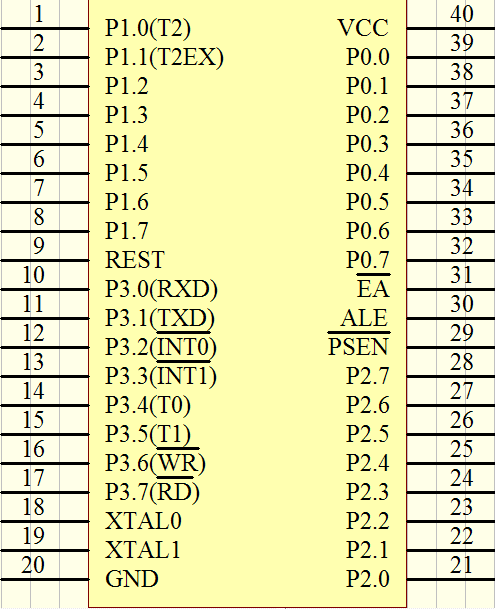


图4.2 STC89C52RC引脚图**[10]**

VCC：接电源+5V

GND：接地

P0口：P0口是一组8位漏极开路行双向I/O口，也既地址/数据总线复用口。可作为输出口使用时，每位可吸收电流的方式驱动8个TTL逻辑电路，对端口写“1”可作为高阻抗输入输出用。在访问外部数据存储器时，这组口线分时转换地址（低8位）和数据总线复用，在访问期间激活内部上拉电阻。在Flash编程时，P0口接收指令字节，而在程序校验时，输出指令字节，校验时，要求有上拉电阻。

P1口：P1口是一个内部提供上拉电阻的8位双向I/O口，P1口缓冲器能接收输出4TTL门电流。P1口管脚写入1后，被内部上拉为高，可用作输入，P1口被外部下拉为低电平时，将输出电流，这是由于内部上拉的缘故。在FLASH编程和校验时，P1口作为第八位地址接收。

P2口：P2口为一个内部上拉电阻的8位双向I/O口，P2口缓冲器可接收，输出4个TTL门电流，当P2口被写“1”时，其管脚被内部上拉电阻拉高，且作为输入。并因此作为输入时，P2口的管脚被外部拉低，将输出电流。这是由于内部上拉的缘故。P2口当用于外部程序存储器或16位地址外部数据存储器进行存取时，P2口输出地址的高八位。在给出地址“1”时，它利用内部上拉优势，当对外部八位地址数据存储器进行读写时，P2口输出其特殊功能寄存器的内容。P2口在FLASH编程和校验时接收高八位地址信号和控制信号。

P3口：P3口管脚是8个带内部上拉电阻的双向I/O口，可接收输出4个TTL门电流。当P3口写入“1”后，它们被内部上拉为高电平，并用作输入。作为输入，由于外部下拉为低电平，P3口将输出电流，这是由于上拉的缘故。P3口除可作为一般的I/O口线外，可作为STC89C52RC的一些特殊功能口，如下表4.1所示：

表4.1 P3端口的第二功能表

|  |  |  |
| --- | --- | --- |
| 引脚 | 第二功能 | 功能说明 |
| P3.0 | RXD | 串行数据接收端 |
| P3.1 | TXD | 串行数据发送端 |
| P3.2 | INT0 | 外部中断0申请信号线 |
| P3.3 | INT1 | 外部中断1申请信号线 |
| P3.4 | T0 | 定时器/计数器0计数输入端 |
| P3.5 | T1 | 定时器/计数器1计数输入端 |
| P3.6 | WR | 外部数据存储器写选通端 |
| P3.7 | RD | 外部数据存储器读选通端 |

REST：复位输出。当震荡器工作时，RST引脚出现两个机器周期以上高电平使机器复位。

ALE：当访问外部程序存储器或数据存储器时，ALE（地址锁存允许）输出脉冲用于锁存地址的低8位字节，即使不访问外部字节，ALE仍时钟震荡频率的1/6输出固定的正脉冲信号，因此它可对外输出时钟脉冲或用于定时目的。要注意的是：每次访问外部存储器时将跳过一个ALE脉冲。对Flash存储器编程期间，该引脚还要输入编程脉冲（PROG）。如有必要，可通过对特殊功能寄存器（SFR）区中的8EH单元的D0位置位，可禁止ALE操作。该位置位后，只有一条MOVX和MOVC指令可激活。此外，此引脚会被微弱拉高，单片机执行外部程序时，应该置ALE无效。

PSEN：程序存入允许（PSEN）输出的是外部程序存储器的读选通信号，当STC89C52RC由外部程序取指令（或数据）时，每个机器周期两次PSEN有效，既输出两个脉冲。在此期间，当访问外部数据存储器，这两次有效的PSEN信号不出现。

EA：外部访问允许。欲使CPU仅访问外部程序存储器（地址为0000H--FFFFH），EA端必须保持低电平（接地）。

XTAL1：震荡器反向放大器及内部时钟的输入端。

XAAL2：震荡器反向放大器的输出端。

### 三、时钟电路

时钟电路是计算机的心脏，它控制着计算机的工作节奏。本次设计选用的单片机时钟频率为12MHZ。图4.3所示为内部震荡器震荡电路**[5]**。

STC89C52RC中有一个构成内部震荡器的高增益反向放大器，引脚XTAL1和XTAL2分别是该放大器的输入端和输出端。这个放大器与作为反馈元件的片外石英或陶瓷震荡器一起构成自激震荡器震荡电路。外接石英晶体（或陶瓷震荡器）及电容C1、C2接在放大器的震荡回路中构成并联震荡电路。本次设计两个电容都选取20pF。

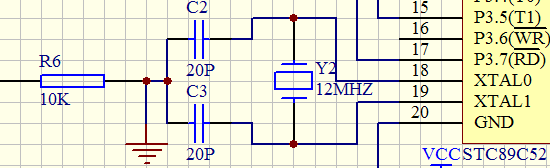


图4.3 时钟电路

## 第二节 控制面板模块

控制面板模块主要包括电源、开始、水位、模式等按键和选择按键（A、B、C）。在本次设计的过程中，对于门控制硬件，在实现的时候也以按键来代替信号的输入。如图11控制面板的按键电路所示：Water\_line\_set按键即为水位设置按键，Mod\_set按键即为模式设置按键，Door\_control即为门控制按键，Start即为开始按键，A\_key、B\_key、C\_key按键即为选择按键。

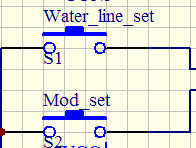
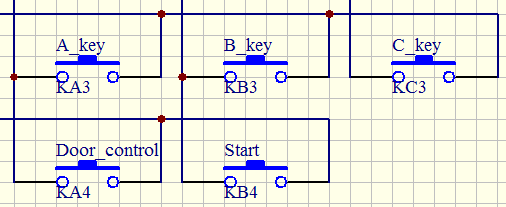


图4.4 控制面板的按键电路

## 第三节 显示模块

本设计选用的显示器件是LCD液晶显示器。1602A LCM是16字×2行的字符模块。如图4.5为LCD的引脚图。

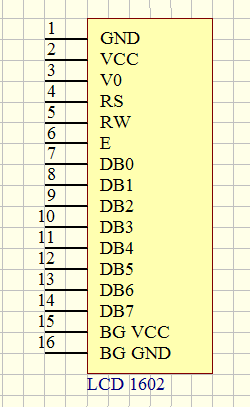


图4.5 LCD 1602引脚图

LCD引脚功能说明如表4.2：

表4.2 LCD 1602引脚功能说明

|  |  |  |
| --- | --- | --- |
| 引脚 | 符号 | 功能说明 |
| 1 | GND | 电源接地 |
| 2 | VCC | 电源正极 |
| 3 | V0 | 液晶显示偏压电压 |
| 4 | RS | 寄存器选择 0：指令寄存器 1：数据寄存器 |
| 5 | R/W | 读/写选择端 |
| 6 | E | 使能信号 |
| 7 | DB0 | 低4位三态、双向数据总线 |
| 8 | DB1 |
| 9 | DB2 |
| 10 | DB3 |
| 11 | DB4 | 高4位三态、双向数据总线  DB7也是一个Busy flag |
| 12 | DB5 |
| 13 | DB6 |
| 14 | DB7 |
| 15 | BG VCC | 背光源正极 |
| 16 | BG GND | 背光源负极 |

## 第四节 蜂鸣器模块

单片机的P3.5引脚通过限流电阻与三极管基极相接，三极管的集电极接有蜂鸣器。通过单片机以软件延时的方式产生方波。蜂鸣器模块电路如图4.6所示：

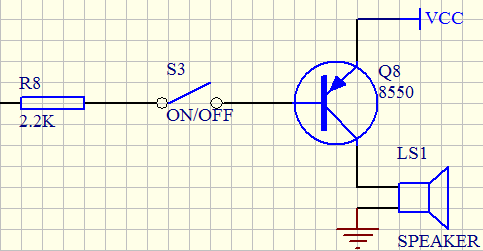


图4.6 蜂鸣器电路

## 第五节 全自动洗衣机详细电路设计

### 一、电路原理图

如图4.7所示全自动洗衣机的详细电路图[1][2][3][4]，输出端口J1的1、2、3号排针分别表示单片机对脱水离合器、排水离合器、进水电磁阀的控制信号，输出端口J2的1、2号排针分别表示单片机对正转洗涤接触器、反转洗涤接触器的控制信号，输入引脚High\_detector、Mid\_detector、Low\_detector分别是高水位探测器、中水位探测器、低水位探测器对单片机控制系统的信息反馈。

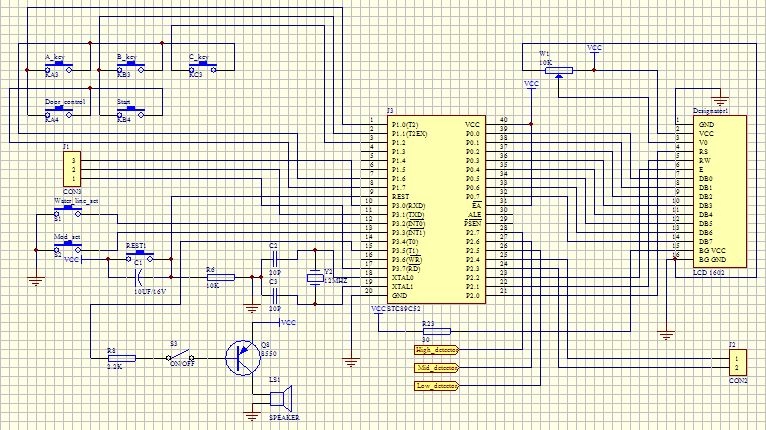


图4.7 全自动洗衣机详细电路图

表4.3是Protel自动生成的元件列表，其中包括了电路原理图中所使用的各种元器件，并对各种元器件的封装和功能信息进行了描述。

表4.3 电路原理图元件列表

|  |  |  |  |
| --- | --- | --- | --- |
| Part Type | Designator | Footprint | Description |
| 2.2K | R8 | AXIAL0.3 |  |
| 10K | R6 | AXIAL0.3 |  |
| 10K | W1 | TO-39 | Potentiometer |
| 10UF/16V | C1 | SIP2 | Capacitor |
| 12MHZ | Y2 | SIP2 | Crystal |
| 20P | C2 | SIP2 | Capacitor |
| 20P | C3 | SIP2 | Capacitor |
| 30 | R23 | AXIAL0.3 |  |
| 8550 | Q8 | SIP3 | PNP Transistor |
| CON2 | J2 | SIP2 | Connector |
| CON3 | J1 | SIP3 | Connector |
| KA3 | A\_key | RAD0.2 |  |
| KA4 | Door\_control | RAD0.2 |  |
| KB3 | B\_key | RAD0.2 |  |
| KB4 | Start | RAD0.2 |  |
| KC3 | C\_key | RAD0.2 |  |
| LCD 1602 | Designator1 | DIP16 |  |
| ON/OFF | S3 | RAD0.2 |  |
| S1 | Water\_line\_set | RAD0.2 |  |
| S2 | Mod\_set | RAD0.2 |  |
| SPEAKER | LS1 | RAD0.2 |  |
| STC89C52 | J3 | DIP40 |  |

### 二、印刷电路板

经过电器法则测试，得到测试错误报表Copy of sheet1.erc如下：

**Error Report For : Copy of Sheet1.Sch 9-Jul-2010 12:25:13**

**End Report**

由测试错误报表可知，电路原理图顺利的通过了电气法则测试。然后生成该电路原理图的网络表文件Copy of sheet1.net，该网络表文件见附录三。

最后进入PCB编辑器，规划电路板为65×75（mm×mm），加载元件封装库与网络表文件，自动布局元件，手动调整元件布局，自动布线，手工调整布线，得到如图4.8的PCB图。表4.4显示了引脚信息报表，提供各引脚的信息。

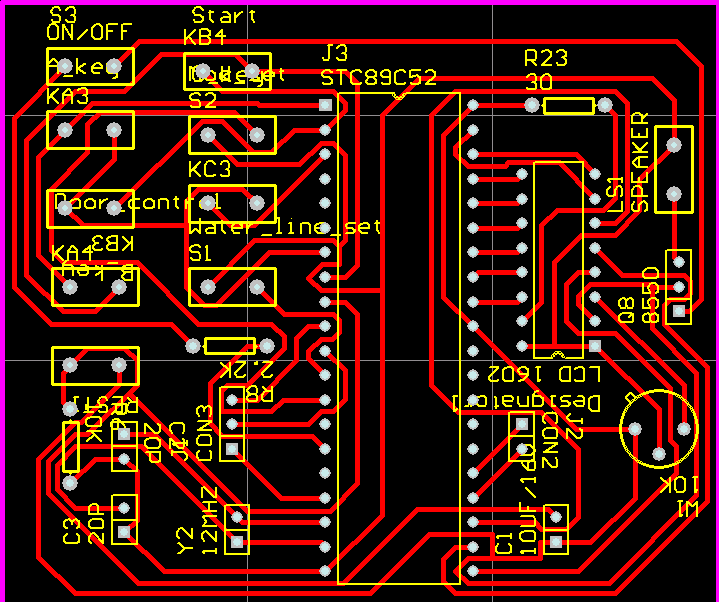


图4.8 全自动洗衣机PCB图

表4.4 引脚信息报表

|  |  |  |
| --- | --- | --- |
| A\_key-1 | A\_key-2 | B\_key-1 |
| B\_key-2 | C1-1 | C1-2 |
| C2-1 | C2-2 | C3-1 |
| C3-2 | C\_key-1 | C\_key-2 |
| Designator-1 | Designator-2 | Designator-3 |
| Designator-4 | Designator-5 | Designator-6 |
| Designator-7 | Designator-8 | Designator-9 |
| Designator-10 | Designator-11 | Designator-12 |
| Designator-13 | Designator-14 | Designator-15 |
| Designator-16 | Door\_contr-1 | Door\_contr-2 |
| J1-1 | J1-2 | J1-3 |
| J2-1 | J2-2 | J3-1 |
| J3-2 | J3-3 | J3-4 |
| J3-5 | J3-6 | J3-7 |
| J3-8 | J3-9 | J3-10 |
| J3-11 | J3-12 | J3-13 |
| J3-14 | J3-15 | J3-16 |
| J3-17 | J3-18 | J3-19 |
| J3-20 | J3-21 | J3-22 |
| J3-23 | J3-24 | J3-25 |
| J3-26 | J3-27 | J3-28 |
| J3-29 | J3-30 | J3-31 |
| J3-32 | J3-33 | J3-34 |
| J3-35 | J3-36 | J3-37 |
| J3-38 | J3-39 | J3-40 |
| LS1-1 | LS1-2 | Mod\_set-1 |
| Mod\_set-2 | Q8-1 | Q8-2 |
| Q8-3 | R6-1 | R6-2 |
| R8-1 | R8-2 | R23-1 |
| R23-2 | REST1-1 | REST1-2 |
| S3-1 | S3-2 | Start-1 |
| Start-2 | W1-1 | W1-2 |
| W1-3 | Water\_line-1 | Water\_line-2 |
| Y2-1 | Y2-2 |  |

## 第六节 本章小结

本章对全自动洗衣机进行了具体的设计，以STC89C52RC为微处理器，搭建蜂鸣器、LCD1602，从硬件层面上实现了设计。第一节是对STC89C52RC的详细介绍，第二节是对控制面板控制按键的设计，第三节对LCD显示做了描述，第四节对蜂鸣器报警模块给予设计，在这四个模块实现的基础上，第五节给出全自动洗衣机的详细电路设计图，以及PCB印刷电路板图。

# 第五章 软件设计及代码实现

硬件设计是本次项目实现的基础，但软件设计则是核心，软件承担着控制系统主要的控制工作。本章将详细对软件进行设计以及编程实现。

## 第一节 程序变量及端口说明

### 一、80C52端口地址定义

表5.1对各种I/O端口和特殊功能寄存器进行了详细的说明[6]，表5.2对程序状态字PSW每一位进行定义，表5.3对TCON定时/计数器控制/状态寄存器TCON的位进行定义，表5.4对中断允许控制寄存器IE的位进行定义，表5.5对中断优先权寄存器IP的位进行定义，表5.6对P3端口特殊功能位进行定义，表5.7对串行通信控制寄存器SCON的位进行定义。

表5.1 特殊功能寄存器的地址定义

|  |  |
| --- | --- |
| 特殊功能寄存器 | 地址 |
| P0 | 0x80 |
| P1 | 0x90 |
| P2 | 0xA0 |
| P3 | 0xB0 |
| PSW | 0xD0 |
| ACC | 0xE0 |
| B | 0xF0 |
| SP | 0x81 |
| DPL | 0x82 |
| DPH | 0x83 |
| PCON | 0x87 |
| TCON | 0x88 |
| TMOD | 0x89 |
| 特殊功能寄存器 | 地址 |
| TL0 | 0x8A |
| TL1 | 0x8B |
| TH0 | 0x8C |
| TH1 | 0x8D |
| IE | 0xA8 |
| IP | 0xB8 |
| SCON | 0x98 |
| SBUF | 0x99 |
| T2CON | 0xC8 |
| RCAP2L | 0xCA |
| RCAP2H | 0xCB |
| TL2 | 0xCC |
| TH2 | 0xCD |

表5-2 PSW程序状态字位表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CY | AC | F0 | RS1 | RS0 | OV | - | P |

表5-3 TCON定时/计数器控制/状态寄存器位表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| TF1 | TR1 | TF0 | TR0 | IE1 | IT1 | IE0 | IT0 |

表5-4 IE中断允许控制寄存器位表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| EA | - | ET2 | ES | ET1 | EX1 | ET0 | EX0 |

表5-5 IP中断优先权寄存器位表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| - | - | PT2 | PS | PT1 | PX1 | PT0 | PX0 |

表5-6 P3端口特殊功能位表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| RD | WR | T1 | T0 | INT1 | INT0 | TXD | RXD |

表5-7 SCON串行通信控制寄存器位表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| SM0 | SM1 | SM2 | REN | TB8 | RB8 | TI | RI |

### 二、程序变量定义和引脚设定

表5.8对程序中用到的变量的数据类型和变量作用进行描述。表5.9对位于控制面板模块的按键输入进行定义，有些按键的引脚定义，考虑到STC89C52RC的引脚有限，采用二维定位，即通过两个引脚来定位一个按键，如4×4个按键只需8个引脚就可以定位。表5.10对各种输入输出的信号进行定义，具体化到引脚所代表的信号。表5.11对LCD的引脚进行定义，其控制信号和数据通信的具体引脚得到分配。

表5.8 程序变量说明

|  |  |  |
| --- | --- | --- |
| 数据类型 | 变量名 | 变量作用描述 |
| bit | door\_cotrol | 门控制按键按下，该值变为0； |
| start | 开始按键按下，该值变为0； |
| A\_key | 选择键A按下，该值变为0； |
| B\_key | 选择键B按下，该值变为0； |
| C\_key | 选择键C按下，该值变为0； |
| uchar | water\_line | 水位存储单元，初始化为中水位 |
| wash\_mode | 洗涤模式存储单元，初始化为强洗 |
| rinse\_mode | 清洗模式存储单元，初始化为一清 |
| frq | 蜂鸣器警报鸣叫变量 |

表5.9 按键输入引脚定义（部分采用两引脚定位按键）

|  |  |  |  |
| --- | --- | --- | --- |
| 数据类型 | 变量名 | 对应引脚 | 变量作用描述 |
| sbit | key\_water\_line | P3^2 | 水位按键输入引脚定义 |
| key\_mode | P3^3 | 模式按键输入引脚定义 |
| door\_cotrol\_row | P1^7 | 门控制按键输入引脚定义 |
| door\_cotrol\_line | P1^0 |
| start\_row | P1^7 | 开始按键输入引脚定义 |
| start\_line | P1^1 |
| A\_key\_row | P1^6 | LCD选择A按键输入引脚定义 |
| A\_key\_line | P1^0 |
| B\_key\_row | P1^6 | LCD选择B按键输入引脚定义 |
| B\_key\_line | P1^1 |
| C\_key\_row | P1^6 | LCD选择C按键输入引脚定义 |
| C\_key\_line | P1^2 |

表5.10 输入输出信号的引脚定义

|  |  |  |  |
| --- | --- | --- | --- |
| 数据类型 | 变量名 | 对应引脚 | 变量作用描述 |
| sbit | pin\_buzzer | P3^5 | 蜂鸣器控制信号引脚定义 |
| pin\_high\_detector | P2^7 | 高水位探测器反馈信号引脚定义，低电平有效 |
| pin\_mid\_detector | P2^6 | 中水位探测器反馈信号引脚定义，低电平有效 |
| pin\_low\_detector | P2^5 | 低水位探测器反馈信号引脚定义，低电平有效 |
| pin\_clockwise\_washing | P2^4 | 正转洗涤接触器控制信号引脚定义 |
| pin\_anticlockwise\_washing | P2^3 | 反转洗涤接触器控制信号引脚定义 |
| pin\_infall | P3^0 | 进水电磁阀控制信号引脚定义,低电平有效 |
| pin\_scupper | P3^1 | 排水电磁阀控制信号引脚定义，低电平有效 |
| pin\_drier | P3^7 | 脱水离合器控制信号引脚定义，低电平有效 |

表5.11 LCD引脚定义

|  |  |  |
| --- | --- | --- |
| 数据类型 | LCD引脚名 | 对应STC89C52RC引脚 |
| sbit | rs | P2^0; |
| rw | P2^1; |
| e | P2^2; |
| DB0 | P0^0; |
| DB1 | P0^1; |
| DB2 | P0^2; |
| DB3 | P0^3; |
| DB4 | P0^4; |
| DB5 | P0^5; |
| DB6 | P0^6; |
| DB7 | P0^7; |

## 第二节 主函数运行流程

主函数包括执行全自动洗涤功能的主体函数和外部中断服务函数，在分析主函数的过程中，我们省去对电源的描述和对复位按键的设计，这是由于这次设计是基于一块51单片机开发板的，在开发板上已提供了电源按键和复位按键的设置。从图5.1的主函数流程图中我们可以清楚地把系统分为一些子模块，如按键扫描模块、进水模块、洗涤模块、排水模块、脱水模块、清洗模块、报警模块以及两个外部中断模块。



图5.1 主函数流程图

Main函数调用各个子功能模块，实现整个洗涤过程，并开启/关闭外部中断，通过响应外部中断或扫描按键来控制用户对洗涤流程的控制和洗涤参数的设置。如下是main实现的具体代码。

P0=0xff; //设置初始引脚值，变量值

P1=0xff;

P2=0xff;

P3=0xff;

IE=0x85; //开启外部中断

IP=0x00;

LCD\_display(0x82,dis1,0xC1,dis2);

keydown(); //扫描门控制键是否按下

while(door\_cotrol!=0){keydown();}

door\_cotrol=1; //恢复门控制键

keydown(); //扫描开始键是否按下

while(start!=0){keydown();}

IE=0x00; //关中断

start=1; //恢复开始键

LCD\_display(0x85,dis3,0xC0,dis0);

delay50ms(20);

LCD\_display\_move(0x83,dis4,13); //LCD显示进水

infall(water\_line);

delay50ms(40);

LCD\_display\_move(0x89,dis5,7); //LCD显示洗涤中

washing(wash\_mode);

scupper(water\_line);

delay50ms(40);

dry30\_120(30);

LCD\_display\_move(0x89,dis6,7); //LCD显示清洗

rinse(rinse\_mode);

delay50ms(40);

LCD\_display\_move(0x8d,dis8,3); //LCD显示脱水中

dry30\_120(120);

LCD\_display(0x85,dis7,0xC0,dis0); //LCD显示完成

buzzer();

## 第三节 子模块程序流程设计

全自动洗衣机子模块包括定时模块、LCD显示模块、按键扫描模块、进水模块、洗涤模块、排水模块、脱水模块、清洗模块、报警模块以及水位设置和模式设置外部中断服务模块[12]。

### 一、定时模块

在程序中的不同情况需要延时的时间段长度不同，因此设计了两种不同时间基数的延时子函数，delay250us的延时单位是250us，delay50ms的延时单位是50ms。

本设计采用12MHZ的晶振，故振荡周期为1/12us，由此可知一个机器周期为1us。

Delay250us函数延时功能使用T0的定时功能，工作模式为模式0，TL0初值为，TH0初值为。



void delay250us(uint i) // 该延时子函数延时250\*i(us)

{

TMOD=0x00;

TR0=1;

while(i!=0)

{

TH0=0x1F; //T0延时250us计数初值

TL0=0x06;

while(TF0!=1);

TF0=0;

i--;

}

TR0=0;

}

Delay50ms函数延时功能使用T0的定时功能，工作模式为模式1，TL0初值为，TH0初值为。



void delay50ms(uint i) //该延时子函数延时50\*i（ms）

{

TMOD=0x01;

TR0=1;

while(i!=0)

{

TH0=0x3C; //T0延时50ms计数初值

TL0=0xB0;

while(TF0!=1);

TF0=0;

i--;

}

TR0=0;

}

### 二、LCD显示模块

LCD显示的主程序流程图如图5.2所示，在程序中实现了两种显示函数，LCD\_display函数静态显示字符，LCD\_display\_move函数显示的字符左右移动循环两次。



图5.2 LCD主程序流程图

void LCD\_display(uchar pos1,uchar code first[],uchar pos2,uchar code second[]) //LCD显示调用子函数

{

uchar i;

lcd\_init(); // 初始化LCD

delay250us(10);

lcd\_pos(pos1); // 设置第一行的显示位置pos1

i = 0;

while(first[i] != '\0')

{ // 显示first

lcd\_wdat(first[i]);

i++;

}

lcd\_pos(pos2); // 设置第二行的显示位置pos2

i = 0;

while(second[i] != '\0')

{

lcd\_wdat(second[i]); // 显示second

i++;

}

delay250us(100);

}

LCD初始化子程序设定显示幕的基本参数，光标开/关和位置，清除LCD地显示内容。图5.3给出了初始化子程序的程序流程图。



图5.3 LCD初始化子程序

void lcd\_init() //LCD初始化设定

{

delay250us(50); //等待LCD电源稳定

lcd\_wcmd(0x38); //16\*2显示，5\*7点阵，8位数据

delay250us(15);

lcd\_wcmd(0x38);

delay250us(15);

lcd\_wcmd(0x38);

delay250us(15);

lcd\_wcmd(0x0c); //显示开，关光标

delay250us(15);

lcd\_wcmd(0x06); //移动光标

delay250us(15);

lcd\_wcmd(0x01); //清除LCD的显示内容

delay250us(15);

}

### 三、按键扫描模块

按键扫描模块有两个子函数，keyscan按键扫描函数和keydown判断按键是否按下和按键位置。

uchar keyscan(void) //按键扫描函数

{

uchar key;

P1=0x0F; //低四位输入

delay250us(1);

if(door\_cotrol\_line==0||A\_key\_line==0)

key=0;

else if(start\_line==0||B\_key\_line==0)

key=1;

else if(high\_detector\_line==0||C\_key\_line==0)

key=2;

else if(mid\_detector\_line==0||low\_detector\_line==0)

key=3;

else

key=16;

P1=0xF0; //高四位输入

delay250us(1); if(door\_cotrol\_row==0||start\_row==0||high\_detector\_row==0||mid\_detector\_row==0)

key=key+12;

else if(A\_key\_row==0||B\_key\_row==0||C\_key\_row==0||low\_detector\_row==0)

key=key+8;

else

key=16;

return key;

}

void keydown(void) //判断键是否按下

{

P1=0xF0;

while(P1==0xF0){}

switch(keyscan())

{

case 12:door\_cotrol=0;break;

case 13:start=0;break;

case 8:A\_key=0;break;

case 9:B\_key=0;break;

case 10:C\_key=0;break;

case 11:pin\_low\_detector=0;break;

case 14:pin\_high\_detector=0;break;

case 15:pin\_mid\_detector=0;break;

default:break;

}

}

### 四、进水模块

在进水模块的实现过程中，要涉及到对进水电磁阀和高、中、低水位探测器的控制。程序需要处理高、中、低水位探测器的信息反馈，考虑到硬件的限制，对于高、中、低水位探测器的信号反馈用按键的方式来代替。

如图5.4所示为进水的程序流程图，PIN\_INFALL设为0即给进水电磁阀发出开启的控制信号，PIN\_INFALL设为1即给进水电磁阀发出关闭的控制信号。PIN\_HIGH\_DETECTOR，PIN\_MID\_DETECTOR，PIN\_LOW\_DETECTOR分别为高、中、低水位探测器的信号反馈端口，低电平有效。



图5.4进水程序流程图

下面代码为进水至高水位的关键代码：

pin\_infall=0; //开启进水电磁阀，开始进水

LCD\_display(0x85,dis9,0xC0,dis0); //显示WAIT

keydown();

while(pin\_high\_detector)

{

keydown();

} //等待进水至高水位

pin\_high\_detector=1; //复位高水位探测器反馈

pin\_infall=1; //关闭进水电磁阀，停止进水

### 五、洗涤模块

在洗涤模块的实现过程中，要涉及到对正转洗涤接触器、反转洗涤接触器的控制。一个洗涤的基本单元包括一次正转和一次反转，每次变换中间间隔时间为2s，一次正转或一次反转时间为28s，所以一个基本单元的时间为60s。对于一次强洗，需要循环20次的基本洗涤单元，即需20分钟；对于一次中洗，需要循环10次的基本洗涤单元，即需10分钟；对于一次弱洗，需要循环5次的基本洗涤单元，即需5分钟。Pin\_clockwise\_washing、pin\_anticlockwise\_washing分别为正转洗涤接触器、反转洗涤接触器的控制引脚，低电平表示开启洗涤接触器，高电平表示关闭洗涤接触器。

下面为洗涤模块中强洗的关键代码。

for(i=0;i<20;i++)

{

pin\_clockwise\_washing=0; //开始正转洗涤接触器

delay50ms(560); //延时28s，即正转洗涤28s

pin\_clockwise\_washing=1; //关闭正转洗涤接触器

delay50ms(40); //延时2s

pin\_anticlockwise\_washing=0; //开始反转洗涤接触器

delay50ms(560);

pin\_anticlockwise\_washing=1;

delay50ms(40);

}

### 六、排水模块

在排水模块的实现过程中，涉及到对排水电磁阀的控制。Pin\_scupper为排水电磁阀的控制引脚，低电平表示开启，高电平表示关闭。对高水位、中水位、低水位的排水差异，只在排水时间上给予区别。高水位排水时间为6s，中水位排水时间为4s，低水位排水时间为2s。

下面是排水的关键代码。

pin\_scupper=0; //开启排水电磁阀

delay50ms(120); //6s

pin\_scupper=1; //关闭排水电磁阀

### 七、脱水模块

在脱水模块的实现过程中，主要涉及对脱水离合器的控制。Pin\_drier为脱水离合器的控制引脚，低电平表示开启，高电平表示关闭。在主程序的调用过程中，用到两个时间段的脱水，30s的脱水和120s的脱水。

下面为脱水模块的关键代码。

pin\_drier=0; //开启脱水离合器

delay50ms(20\*dry\_time); //延时dry\_time秒，即脱水dry\_time段时间

pin\_drier=1; //关闭脱水离合器

### 八、清洗模块

在清洗模块的实现过程中，涉及到进水和洗涤。一清只进行一次进水和洗涤，两清要重复一清流程两次。如图5.5所示，为清洗模块的程序流程图。



图5.5 清洗程序流程图

下面代码为清洗模块的关键代码。

pin\_clockwise\_washing=0; //开始正转洗涤接触器

delay50ms(560); //延时28s，即正转洗涤28s

pin\_clockwise\_washing=1; //关闭正转洗涤接触器

delay50ms(40); //延时2s

pin\_anticlockwise\_washing=0; //开始反转洗涤接触器

delay50ms(560);

pin\_anticlockwise\_washing=1;

delay50ms(40);

### 九、报警模块

在报警模块的实现过程中，涉及到蜂鸣器和定时/计数器T1。Pin\_buzzer为蜂鸣器方波输入引脚。

下面代码为报警模块的关键代码。

void buzzer() //蜂鸣器警报

{

TMOD=0x10;

frq=0x00;

TH1=0x00;

TL1=0xff;

TR1=1;

IE=0x88;

while(20000)

{

frq++;

delayms(1); //延时1ms

}

TR1=0;

}

void timer1() interrupt 3 using 1 //T1的中断服务程序

{

TH0=0xfe;

TL0=frq;

pin\_buzzer=~pin\_buzzer;

}

### 十、外部中断服务模块

STC89C52RC单片机有两个外部中断，水位设定模块使用外部中断0，模式设定模块使用外部中断1。

外部中断0的中断服务代码[15]如下：

void water\_line\_set() interrupt 0 using 2

{

LCD\_display(0x81,dis10,0xC1,dis11); //LCD显示水位设定：高（high）中（mid）低（low）

keydown();

if(A\_key==0)

water\_line=high;

else if(B\_key==0)

water\_line=mid;

else

water\_line=low;

LCD\_display(0x82,dis1,0xC1,dis2);

}

外部中断1的中断服务代码如下：

void mode\_set() interrupt 2 using 3

{

LCD\_display(0x81,dis12,0xC1,dis13);

//LCD显示洗涤模式设定：强（strong）中（middle）弱（feeble）

keydown();

if(A\_key==0)

wash\_mode=strong;

else if(B\_key==0)

wash\_mode=middle;

else

wash\_mode=feeble;

LCD\_display(0x81,dis14,0xC1,dis15); //LCD显示清洗模式设定：一清（once）二清（twice）

keydown();

if(A\_key==0)

rinse\_mode=once;

else

rinse\_mode=twice;

LCD\_display(0x82,dis1,0xC1,dis2);

}

## 第四节 本章小结

本章对全自动洗衣机的软件进行了详细的设计和实现。第一节是对程序变量和端口具体信息给予了详细介绍，第二节按照全自动洗衣机的执行流程实现主函数，第三节实现各个子模块。在第二节的主函数模块和第三节的定时模块、按键模块、LCD显示模块、进水模块、洗涤模块、排水模块、清洗模块、脱水模块、报警模块、外部中断服务模块等插入部分关键代码，详细阐述了实现方法。

# 第六章 系统测试

用RS-232串口数据线将开发板与PC机的串口相连接，通过软件STC-ISP将程序烧制到开发板的单片机里，然后启动单片机，调试各个功能。如图6.1所示为STC-ISP的主窗口。

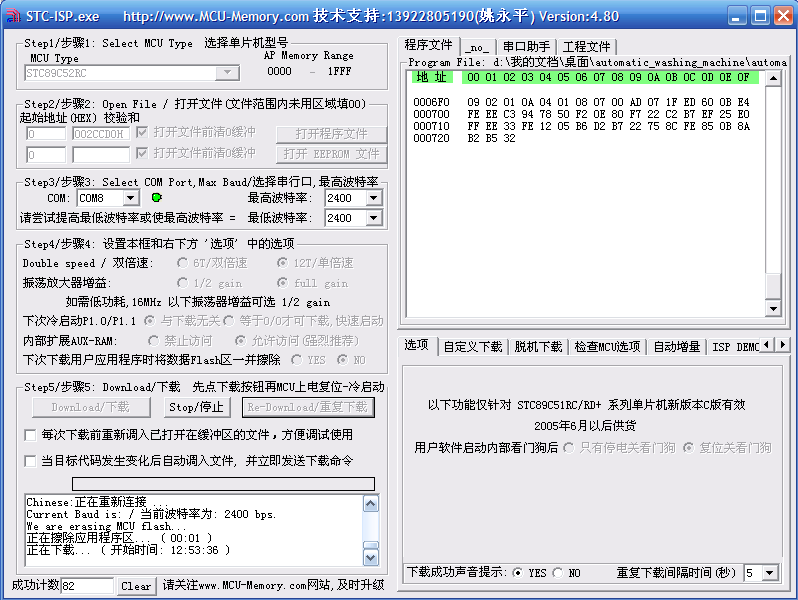


图6.1 STC-ISP的代码烧制环境

单片机供电后，全自动洗衣机的控制面板的主界面如图6.2所示。



图6.2 全自动洗衣机的主界面

然后进入水位设置模式，如图6.3所示。

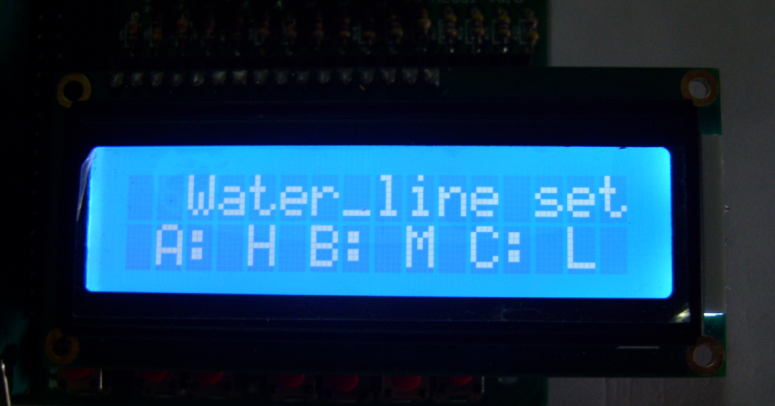


图6.3 水位设置界面

选择A来设置水位，然后进入模式设置界面，如图6.4、图6.5所示。

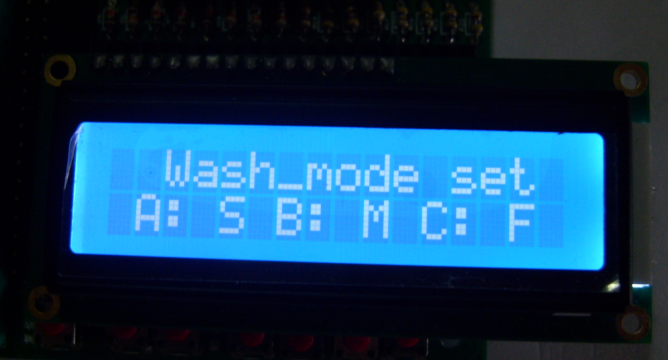
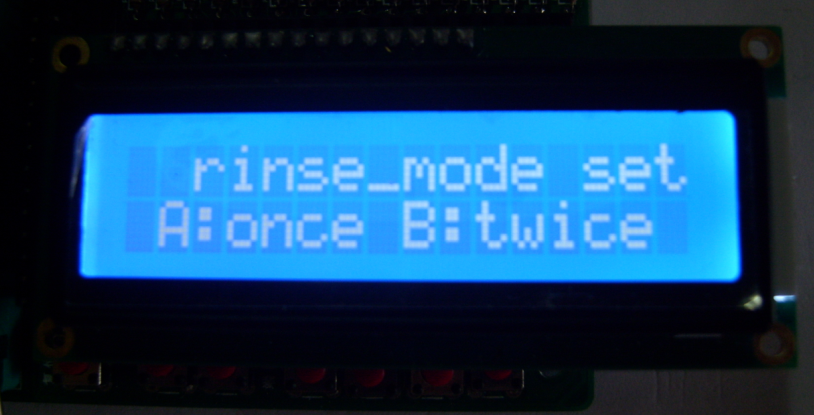


图6.4 洗涤模式设定界面 图6.5 清洗模式设定界面

洗涤模式选C，清洗模式选A，然后按门控制和开始按键，洗涤流程开始。



图6.6 开始界面 图6.7 进水界面



图6.8 等待水位探测器输入界面 图6.9 洗涤界面

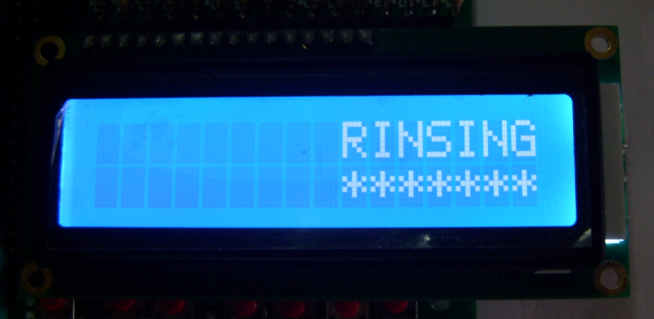
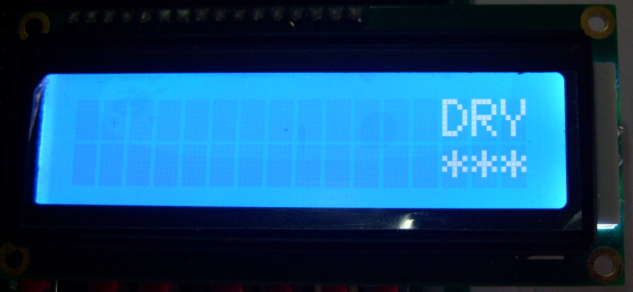


图6.10 清洗界面 图6.11 脱水界面



图6.12 洗涤完毕界面

# 结 论

本文充分地利用了51单片机提供的功能，使用C51开发出了全自动洗衣机的控制系统，本系统具备家用洗衣机所有的基本控制功能。

在本系统的硬件电路设计中，充分利用Protel提供的便捷功能，方便地进行电路原理图的设计，然后生成可行的PCB板图，最终得到简单、高效的电路。在软件设计中，采用模块化设计，将全自动洗衣机的功能分为一些子功能模块。在最后调试时，使用RS-232连接单片机与电脑，利用STC-ISP将烧制程序到单片机上，逐个功能进行调试。

本次设计的控制系统实现了全自动洗衣机的基本功能，比如洗涤、清洗、脱水等，但还不够完善。全自动洗衣机发展趋势是不断向着高度自动化、智能化、健康化、节能、大容量、微型化、安全化等方向发展。本次设计没有考虑这些细节功能。

首先，在全自动洗衣机的设计中应具有进水超时保护功能。洗衣机进水过程中，因水压过低或进水管连接口有杂质堵塞过滤网等原因致使进水时间过长，此时为防止进水阀线圈长时间通电而发热进而烧毁或引燃火灾事故，在控制系统中，设置一定的时间界限，当时间到达时间界限而水位仍未达到预定水位，则控制系统会发出指令使进水阀线圈断电，停止洗衣机进水动作执行。此时，控制系统应该通过LCD显示或蜂鸣器来提醒洗衣机使用者洗衣机进水有问题。

其次，在全自动洗衣机的设计中应具有排水超时保护功能。洗衣机排水一般通过排水电机或电磁铁的吸合来牵引排水阀门打开而使洗涤筒内的水流向筒外。有时因排水管未放下，或因排水口位置过高、或排水管内有异物堵塞，抑或结冰堵塞而导致洗衣机无法排水或排水不畅，但若排水时间稍长，控制系统应该发出停止排水的指令，以免排水电机或电磁铁线圈长时间通电而烧毁。此时，全自动洗衣机的控制系统，若排水从开始至一定时间后仍未下降到一定水位，就控制排水超时保护功能执行。并通过LCD显示或蜂鸣器来提醒洗衣机使用者洗衣机排水有问题。

最后，在全自动洗衣机的设计中应具有电机过热保护功能。洗衣机主电机工作时因各种原因会致使温升过高，这时，控制系统就会发出停止工作的指令，保护主电机不受损伤。

# 致 谢

经过近三个月的不断努力，我终于完成了自己的本科毕业设计论文。在毕业设计的程序开发和论文的撰写过程中，感谢老师和同学给我提供的帮助。

首先，我要衷心地感谢我的毕业设计指导老师杜欢老师在我完成毕业设计的过程中给予的指导和关心。在这段时间里，杜老师给予了我悉心的指导，帮助我清楚了单片机控制系统的开发流程。从硬件原理图的设计，到软件C51程序的开发，杜老师都耐心地指导我学习这些自己首次接触的知识。在论文撰写过程中，杜老师也仔细地给我指出论文中存在的缺点和欠缺的内容。

在这里，我还要向周围的同学表示感谢。浓厚的学习氛围激励我耐心地完成毕业设计，当遇到自己解决不了的问题时，他们总能够和我一起来讨论这些问题。

其次，感谢我的亲人，他们一直以来默默地关心和支持着我完成本科学业，是我不断成长和取得进步的动力源泉。

最后，向百忙中抽空来审阅我论文的老师们表示深深的感谢！

# 参考文献

1. 王彦平，任延群，危胜军等．Protel 99 电路设计指南［M］．北京：清华大学出版社，2000．
2. 李东生，张勇，许四毛等．Protel 99 SE 电路设计技术入门与应用［M］．北京：电子工业出版社，2002．
3. 梁恩主，梁恩维等．Protel 99 SE电路设计与仿真应用［M］．北京：清华大学出版社，2000．
4. 王正谋．Protel电路设计使用教程［M］．北京：电子工业出版社，20003．
5. 马家辰，孙玉德，张颖等．MCS-51单片机原理及接口技术［M］．哈尔滨：哈尔滨工业大学出版社，2000．
6. 王守中．51单片机开发入门与典型实例［M］．北京：人民邮电出版社，2007．
7. 曾璐，李明. 基于AT89C52单片机的洗衣机智能控制系统［J］．电子技术，2006，33(11):67-69．
8. 王琰. 基于MCS-51单片机的洗衣机控制系统设计［J］．自动化与仪器仪表，2008，(4):37-38．
9. 李圣良. 基于MCS-51单片机的洗衣控制系统［J］．兵工自动化，2006，25(10):77-79．
10. 郭强，刘志峰，张爱平，王建华. 基于STC89C52单片机的智能交通灯控系统设计［J］．机电一体化，2008，14(11):85-88，92.
11. 王一然．浅析MCS-51单片机I/O口的扩展［J］．光盘技术，2009，(6).
12. 陶孝海. 51单片机C语言编程技巧［J］．计算机时代，2009， (6):41-42．
13. 庄建清，徐玮. 51单片机毕业设计系列开发板［J］．电子制作，2009， (4):23-25．
14. 郑丽群. 谈8051单片机开发系统及开发过程［J］．中国新技术新产品，2009， (4):10-10．
15. 赵群. 简单的软硬兼施51单片机外部中断扩展［J］．科技资讯，2008， (35):24-25．

# 附 录

### 一、英文原文

**Embedded system**

An embedded system is a [computer system](http://en.wikipedia.org/wiki/Computer_system) designed to perform one or a few dedicated functions often with [real-time computing](http://en.wikipedia.org/wiki/Real-time_computing) constraints. It is embedded as part of a complete device often including hardware and mechanical parts. By contrast, a general-purpose computer, such as a [personal computer](http://en.wikipedia.org/wiki/Personal_computer), is designed to be flexible and to meet a wide range of end-user needs. Embedded systems control many devices in common use today. Embedded systems are controlled by one or more main processing cores that are typically either [microcontrollers](http://en.wikipedia.org/wiki/Microcontroller) or [digital signal processors](http://en.wikipedia.org/wiki/Digital_signal_processor). The key characteristic, however, is being dedicated to handle a particular task, which may require very powerful processors. Since the embedded system is dedicated to specific tasks, design engineers can optimize it to reduce the size and cost of the product and increase the reliability and performance. Some embedded systems are mass-produced, benefiting from [economies of scale](http://en.wikipedia.org/wiki/Economies_of_scale). In general, "embedded system" is not a strictly definable term, as most systems have some element of extensibility or programmability. For example, [handheld computers](http://en.wikipedia.org/wiki/Handheld_computer) share some elements with embedded systems such as the operating systems and microprocessors which power them, but they allow different applications to be loaded and peripherals to be connected. Moreover, even systems which don't expose programmability as a primary feature generally need to support software updates. On a continuum from "general purpose" to "embedded", large application systems will have subcomponents at most points even if the system as a whole is "designed to perform one or a few dedicated functions", and is thus appropriate to call "embedded".

**History**

In the earliest years of computers in the 1930–40s, computers were sometimes dedicated to a single task, but were far too large and expensive for most kinds of tasks performed by embedded computers of today. Over time however, the concept of [programmable controllers](http://en.wikipedia.org/wiki/Programmable_controllers) evolved from traditional [electromechanical](http://en.wikipedia.org/wiki/Electromechanical) sequencers, via solid state devices, to the use of computer technology.

One of the first recognizably modern embedded systems was the [Apollo Guidance Computer](http://en.wikipedia.org/wiki/Apollo_Guidance_Computer), developed by [Charles Stark Draper](http://en.wikipedia.org/wiki/Charles_Stark_Draper) at the MIT Instrumentation Laboratory. At the project's inception, the Apollo guidance computer was considered the riskiest item in the Apollo project as it employed the then newly developed monolithic integrated circuits to reduce the size and weight. An early mass-produced embedded system was the D-17 guidance computer for the [Minuteman missile](http://en.wikipedia.org/wiki/Minuteman_(missile)), released in 1961. It was built from [transistor](http://en.wikipedia.org/wiki/Transistor) [logic](http://en.wikipedia.org/wiki/Digital_circuit) and had a [hard disk](http://en.wikipedia.org/wiki/Hard_disk) for main memory. When the Minuteman II went into production in 1966, the D-17 was replaced with a new computer that was the first high-volume use of integrated circuits. This program alone reduced prices on quad [nand gate ICs](http://en.wikipedia.org/wiki/Sheffer_stroke#NAND_gate) from $1000/each to $3/each, permitting their use in commercial products.

Since these early applications in the 1960s, embedded systems have come down in price and there has been a dramatic rise in processing power and functionality. The first [microprocessor](http://en.wikipedia.org/wiki/Microprocessor) for example, the [Intel 4004](http://en.wikipedia.org/wiki/Intel_4004), was designed for [calculators](http://en.wikipedia.org/wiki/Calculator) and other small systems but still required many external memory and support chips. In 1978 National Engineering Manufacturers Association released a "standard" for programmable microcontrollers, including almost any computer-based controllers, such as single board computers, numerical, and event-based controllers.

As the cost of microprocessors and microcontrollers fell it became feasible to replace expensive knob-based [analog](http://en.wikipedia.org/wiki/Analog_electronics) components such as [potentiometers](http://en.wikipedia.org/wiki/Potentiometer) and [variable capacitors](http://en.wikipedia.org/wiki/Variable_capacitor) with up/down buttons or knobs read out by a microprocessor even in some consumer products. By the mid-1980s, most of the common previously external system components had been integrated into the same chip as the processor and this modern form of the [microcontroller](http://en.wikipedia.org/wiki/Microcontroller) allowed an even more widespread use, which by the end of the decade were the norm rather than the exception for almost all electronics devices.

The integration of microcontrollers has further increased the applications for which embedded systems are used into areas where traditionally a computer would not have been considered. A general purpose and comparatively low-cost microcontroller may often be programmed to fulfill the same role as a large number of separate components. Although in this context an embedded system is usually more complex than a traditional solution, most of the complexity is contained within the microcontroller itself. Very few additional components may be needed and most of the design effort is in the software. The intangible nature of software makes it much easier to prototype and test new revisions compared with the design and construction of a new circuit not using an embedded processor.

**Characteristics**

Embedded systems are designed to do some specific task, rather than be a general-purpose computer for multiple tasks. Some also have [real-time](http://en.wikipedia.org/wiki/Real-time_computing) performance constraints that must be met, for reasons such as safety and usability; others may have low or no performance requirements, allowing the system hardware to be simplified to reduce costs.

Embedded systems are not always standalone devices. Many embedded systems consist of small, computerized parts within a larger device that serves a more general purpose. For example, the [Gibson Robot Guitar](http://en.wikipedia.org/wiki/Gibson_Robot_Guitar) features an embedded system for tuning the strings, but the overall purpose of the Robot Guitar is, of course, to play music. Similarly, an embedded system in an [automobile](http://en.wikipedia.org/wiki/Automobile) provides a specific function as a subsystem of the car itself.

The program instructions written for embedded systems are referred to as [firmware](http://en.wikipedia.org/wiki/Firmware), and are stored in read-only memory or [Flash memory](http://en.wikipedia.org/wiki/Flash_memory) chips. They run with limited computer hardware resources: little memory, small or non-existent keyboard and/or screen.

1. **User interface**

Embedded systems range from no user interface at all — dedicated only to one task — to complex [graphical user interfaces](http://en.wikipedia.org/wiki/Desktop_operating_system#Graphical_user_interfaces) that resemble modern computer desktop operating systems. Simple embedded devices use [buttons](http://en.wikipedia.org/wiki/Push-button), [LEDs](http://en.wikipedia.org/wiki/LED), graphic or character [LCDs](http://en.wikipedia.org/wiki/LCD) (for example popular [HD44780 LCD](http://en.wikipedia.org/wiki/HD44780_Character_LCD)) with a simple [menu system](http://en.wikipedia.org/wiki/Menu_(computing)).

A more sophisticated devices use graphical screen with [touch](http://en.wikipedia.org/wiki/Touch_screen) sensing or screen-edge buttons provide flexibility while minimizing space used: the meaning of the buttons can change with the screen, and selection involves the natural behavior of pointing at what's desired. [Handheld systems](http://en.wikipedia.org/wiki/Mobile_device) often have a screen with a "joystick button" for a pointing device.

Some systems provide user interface remotely with the help of serial (e.g. [RS-232](http://en.wikipedia.org/wiki/RS-232), [USB](http://en.wikipedia.org/wiki/USB), [I²C](http://en.wikipedia.org/wiki/I%C2%B2C), etc.) or network (e.g. [Ethernet](http://en.wikipedia.org/wiki/Ethernet)) connection. In spite of installed client software and cables are needed this approach usually gives a lot of advantages: extends the capabilities of embedded system, avoids the cost of a display, simplifies [BSP](http://en.wikipedia.org/wiki/Board_support_package), allows to build rich user interface on PC.

1. **Processors in embedded systems**

Embedded processors can be broken into two broad categories: ordinary microprocessors and microcontrollers, which have many more peripherals on chip, reducing cost and size. Contrasting to the personal computer and server markets, a fairly large number of basic [CPU architectures](http://en.wikipedia.org/wiki/CPU_architecture) are used; there are [Von Neumann](http://en.wikipedia.org/wiki/Von_Neumann_architecture) as well as various degrees of [Harvard architectures](http://en.wikipedia.org/wiki/Harvard_architecture), [RISC](http://en.wikipedia.org/wiki/RISC) as well as non-RISC and [VLIW](http://en.wikipedia.org/wiki/VLIW); word lengths vary from 4-bit to 64-bits and beyond (mainly in [DSP](http://en.wikipedia.org/wiki/Digital_signal_processor) processors) although the most typical remain 8/16-bit. Many architectures come in a large number of different variants and shapes, many of which are also manufactured by several different companies.

A long but still not exhaustive list of common architectures are: [65816](http://en.wikipedia.org/wiki/65816), [65C02](http://en.wikipedia.org/wiki/65C02), [68HC08](http://en.wikipedia.org/wiki/68HC08), [68HC11](http://en.wikipedia.org/wiki/68HC11), [68k](http://en.wikipedia.org/wiki/68k), [8051](http://en.wikipedia.org/wiki/Intel_8051), [ARM](http://en.wikipedia.org/wiki/ARM_architecture), [AVR](http://en.wikipedia.org/wiki/Atmel_AVR), [AVR32](http://en.wikipedia.org/wiki/AVR32), [Blackfin](http://en.wikipedia.org/wiki/Blackfin), [C167](http://en.wikipedia.org/wiki/C167_family), [Coldfire](http://en.wikipedia.org/wiki/Coldfire), [COP8](http://en.wikipedia.org/wiki/COP8), [Cortus APS3](http://en.wikipedia.org/w/index.php?title=Cortus_APS3&action=edit&redlink=1), [eZ8](http://en.wikipedia.org/wiki/Zilog_Z8), [eZ80](http://en.wikipedia.org/wiki/EZ80), [FR-V](http://en.wikipedia.org/wiki/FR-V), [H8](http://en.wikipedia.org/wiki/Renesas_H8), [HT48](http://en.wikipedia.org/wiki/HT48FXX_Flash_I/O_type_series), [M16C](http://en.wikipedia.org/wiki/M16C), [M32C](http://en.wikipedia.org/wiki/M32C), [MIPS](http://en.wikipedia.org/wiki/MIPS_architecture), [MSP430](http://en.wikipedia.org/wiki/MSP430), [PIC](http://en.wikipedia.org/wiki/PIC_microcontroller), [PowerPC](http://en.wikipedia.org/wiki/PowerPC), [R8C](http://en.wikipedia.org/wiki/R8C), [SHARC](http://en.wikipedia.org/wiki/Super_Harvard_Architecture_Single-Chip_Computer), [ST6](http://en.wikipedia.org/wiki/ST6), [SuperH](http://en.wikipedia.org/wiki/SuperH), [TLCS-47](http://en.wikipedia.org/wiki/TLCS-47), [TLCS-870](http://en.wikipedia.org/wiki/TLCS-870), [TLCS-900](http://en.wikipedia.org/wiki/TLCS-900), [Tricore](http://en.wikipedia.org/wiki/Tricore), [V850](http://en.wikipedia.org/wiki/V850), [x86](http://en.wikipedia.org/wiki/X86_architecture), [XE8000](http://en.wikipedia.org/wiki/XE8000), [Z80](http://en.wikipedia.org/wiki/Z80), [AsAP](http://en.wikipedia.org/wiki/Asynchronous_array_of_simple_processors) etc.

1. **Computer boards**

[PC/104](http://en.wikipedia.org/wiki/PC/104) and PC/104+ are examples of standards for computer boards intended for small, low-volume embedded and ruggedized systems, mostly x86-based. These often use [DOS](http://en.wikipedia.org/wiki/DOS), [Linux](http://en.wikipedia.org/wiki/Linux), [NetBSD](http://en.wikipedia.org/wiki/NetBSD), or an embedded [real-time operating system](http://en.wikipedia.org/wiki/Real-time_operating_system) such as [MicroC/OS-II](http://en.wikipedia.org/wiki/MicroC/OS-II), [QNX](http://en.wikipedia.org/wiki/QNX) or [VxWorks](http://en.wikipedia.org/wiki/VxWorks). Sometimes these boards use non-x86 processors.

In certain applications, where small size and power efficiency are not primary concerns, the components used may be compatible with those used in general purpose x86 personal computers. Boards such as the VIA [EPIA](http://en.wikipedia.org/wiki/EPIA) range help to bridge the gap by being PC-compatible but highly integrated, physically smaller or have other attributes making them attractive to embedded engineers. The advantage of this approach is that low-cost commodity components may be used along with the same software development tools used for general software development. Systems built in this way are still regarded as embedded since they are integrated into larger devices and fulfill a single role. Examples of devices that may adopt this approach are [ATMs](http://en.wikipedia.org/wiki/Automated_teller_machines) and [arcade machines](http://en.wikipedia.org/wiki/Arcade_machines), which contain code specific to the application.

However, most ready-made embedded systems boards are not PC-centered and do not use the ISA or PCI busses. When a [System-on-a-chip](http://en.wikipedia.org/wiki/System-on-a-chip) processor is involved, there may be little benefit to having a standardized bus connecting discrete components, and the environment for both hardware and software tools may be very different.

One common design style uses a small system module, perhaps the size of a business card, holding high density [BGA](http://en.wikipedia.org/wiki/Ball_grid_array) chips such as an [ARM](http://en.wikipedia.org/wiki/ARM_architecture)-based [System-on-a-chip](http://en.wikipedia.org/wiki/System-on-a-chip) processor and peripherals, external [flash memory](http://en.wikipedia.org/wiki/Flash_memory) for storage, and [DRAM](http://en.wikipedia.org/wiki/Dynamic_random_access_memory) for runtime memory. The module vendor will usually provide boot software and make sure there is a selection of operating systems, usually including [Linux](http://en.wikipedia.org/wiki/Linux) and some real time choices. These modules can be manufactured in high volume, by organizations familiar with their specialized testing issues, and combined with much lower volume custom main boards with application-specific external peripherals. [Gumstix](http://en.wikipedia.org/wiki/Gumstix) product lines are a Linux-centric example of this model.

A common configuration for very-high-volume embedded systems is the [system on a chip](http://en.wikipedia.org/wiki/System_on_a_chip) which contains a complete system consisting of multiple processors, multipliers, caches and interfaces on a single chip. System on Chip can be implemented as an [application-specific integrated circuit](http://en.wikipedia.org/wiki/Application-specific_integrated_circuit) (ASIC) or using a [field-programmable gate array](http://en.wikipedia.org/wiki/Field-programmable_gate_array) (FPGA).

1. **Tools**

As for other software, embedded system designers use [compilers](http://en.wikipedia.org/wiki/Compiler), [assemblers](http://en.wikipedia.org/wiki/Assembly_language#Assembler), and [debuggers](http://en.wikipedia.org/wiki/Debugger) to develop embedded system software.

Software tools can come from several sources: Software companies that specialize in the embedded market; Ported from the [GNU](http://en.wikipedia.org/wiki/GNU) software development tools; Sometimes, development tools for a personal computer can be used if the embedded processor is a close relative to a common PC processor.

However, they may also use some more specific tools:

①Custom compilers and linkers may be used to improve optimization for the particular hardware.

②Utilities to add a checksum or [CRC](http://en.wikipedia.org/wiki/Cyclic_redundancy_check) to a program, so the embedded system can check if the program is valid.

③For systems using [digital signal processing](http://en.wikipedia.org/wiki/Digital_signal_processing), developers may use a math workbench such as [Scilab](http://en.wikipedia.org/wiki/Scilab) / [Scicos](http://en.wikipedia.org/wiki/Scicos), [MATLAB](http://en.wikipedia.org/wiki/MATLAB) / [Simulink](http://en.wikipedia.org/wiki/Simulink), [EICASLAB](http://en.wikipedia.org/wiki/EICASLAB), [MathCad](http://en.wikipedia.org/wiki/MathCad) [Mathematica](http://en.wikipedia.org/wiki/Mathematica),or [FlowStone DSP](http://en.wikipedia.org/w/index.php?title=FlowStone_DSP&action=edit&redlink=1) to simulate the mathematics. They might also use libraries for both the host and target which eliminates developing DSP routines as done in [DSPnano RTOS](http://en.wikipedia.org/wiki/DSPnano_RTOS) and [Unison Operating System](http://en.wikipedia.org/wiki/Unison_Operating_System).

④An embedded system may have its own special language or design tool, or add enhancements to an existing language such as [Forth](http://en.wikipedia.org/wiki/Forth_(programming_language)) or [Basic](http://en.wikipedia.org/wiki/BASIC_Stamp).

⑤Another alternative is to add a [real-time operating system](http://en.wikipedia.org/wiki/Real-time_operating_system) or [embedded operating system](http://en.wikipedia.org/wiki/Embedded_operating_system), which may have DSP capabilities.

### 二、英文翻译

**嵌入式系统**

嵌入式系统，是一种完全嵌入受控器件内部，为特定应用而设计的专用[计算机](http://zh.wikipedia.org/wiki/%E8%AE%A1%E7%AE%97%E6%9C%BA)系统，根据[英国电器工程师协会](http://zh.wikipedia.org/w/index.php?title=%E8%8B%B1%E5%9C%8B%E9%9B%BB%E5%99%A8%E5%B7%A5%E7%A8%8B%E5%B8%AB%E5%8D%94%E6%9C%83&action=edit&redlink=1)的定义，嵌入式系统为控制、监视或辅助设备、机器或用于工厂运作的装置。与[个人计算机](http://zh.wikipedia.org/wiki/PC)这样的通用[计算机](http://zh.wikipedia.org/wiki/%E8%AE%A1%E7%AE%97%E6%9C%BA)系统不同，嵌入式系统通常执行的是带有特定要求的预先定义的任务。由于嵌入式系统只针对一项特殊的任务，设计人员能够对它进行优化，减小尺寸降低成本。由于嵌入式系统通常进行大量生产，所以单个的成本节约，能够随着产量进行成百上千的放大。由于[个人数码助理](http://zh.wikipedia.org/wiki/%E4%B8%AA%E4%BA%BA%E6%95%B0%E7%A0%81%E5%8A%A9%E7%90%86)及[手持设备](http://zh.wikipedia.org/wiki/%E6%89%8B%E6%8C%81%E8%AE%BE%E5%A4%87)在硬件上设计的特性，即使在软件上的扩充性比其他装置来得好，但普遍被认为是嵌入式装置。但是这个定义也逐渐模糊。举例来说，英特尔公司的ATOM微处理器原本是为了移动互联网设备这一类的嵌入式系统设计的，但是现在更多的被应用于[上网本](http://zh.wikipedia.org/wiki/%E4%B8%8A%E7%BD%91%E6%9C%AC)，而上网本属于使用[Windows](http://zh.wikipedia.org/wiki/Windows)或者[Linux](http://zh.wikipedia.org/wiki/Linux)的[个人计算机](http://zh.wikipedia.org/wiki/%E4%B8%AA%E4%BA%BA%E8%AE%A1%E7%AE%97%E6%9C%BA)，因此嵌入式系统的定义又更模糊了。

**一、历史**

第一个被大家认可的现代嵌入式系统是[麻省理工学院](http://zh.wikipedia.org/wiki/%E9%BA%BB%E7%9C%81%E7%90%86%E5%B7%A5%E5%AD%A6%E9%99%A2)仪器研究室的[查尔斯.斯塔克.德雷珀](http://zh.wikipedia.org/w/index.php?title=%E6%9F%A5%E5%B0%94%E6%96%AF%C2%B7%E6%96%AF%E5%A1%94%E5%85%8B%C2%B7%E5%BE%B7%E9%9B%B7%E7%8F%80&action=edit&redlink=1)开发的[阿波罗导航计算机](http://zh.wikipedia.org/w/index.php?title=%E9%98%BF%E6%B3%A2%E7%BD%97%E5%AF%BC%E8%88%AA%E8%AE%A1%E7%AE%97%E6%9C%BA&action=edit&redlink=1)。在两次月球飞行中他们在[太空驾驶舱](http://zh.wikipedia.org/w/index.php?title=%E5%A4%AA%E7%A9%BA%E9%A9%BE%E9%A9%B6%E8%88%B1&action=edit&redlink=1)和[月球登陆舱](http://zh.wikipedia.org/w/index.php?title=%E6%9C%88%E7%90%83%E7%99%BB%E9%99%86%E8%88%B1&action=edit&redlink=1)都是用了这种[惯性导航系统](http://zh.wikipedia.org/wiki/%E6%83%AF%E6%80%A7%E5%AF%BC%E8%88%AA%E7%B3%BB%E7%BB%9F)。

在计划刚开始的时候，阿波罗导航计算机被认为是阿波罗计划风险最大的部分。为了减小尺寸和重量而使用的当时最新的单片[集成电路](http://zh.wikipedia.org/wiki/%E9%9B%86%E6%88%90%E7%94%B5%E8%B7%AF)加大了阿波罗计划的风险。

第一款大批量生产的嵌入式系统是1961年发布的民兵I导弹上的D-17自动导航控制计算机。它是由独立的[晶体管逻辑电路](http://zh.wikipedia.org/w/index.php?title=%E6%99%B6%E4%BD%93%E7%AE%A1%E9%80%BB%E8%BE%91%E7%94%B5%E8%B7%AF&action=edit&redlink=1)建造的，它带有一个作为主内存的[硬盘](http://zh.wikipedia.org/wiki/%E7%A1%AC%E7%9B%98)。当民兵II导弹在1966年开始生产的时候，D-17由第一次使用大量集成电路的更新计算机所替代。仅仅这个项目就将与非门集成电路模块的价格从每个1000美元降到低了每个3美元，使集成电路的商用成为可能。

民兵导弹的嵌入式计算机有一个重要的设计特性：它能够在项目后期对制导算法重新编程以获得更高的导弹精度，并且能够使用计算机测试导弹，从而减少测试用电缆和接头的重量。

这些二十世纪六十年代的早期应用，使嵌入式系统得到长足发展，它的价格开始下降，同时处理能力和功能也获得了巨大的提高。[英特尔4004](http://zh.wikipedia.org/wiki/Intel_4004)是第一款[微处理器](http://zh.wikipedia.org/wiki/%E5%BE%AE%E5%A4%84%E7%90%86%E5%99%A8)，它在[计算器](http://zh.wikipedia.org/wiki/%E8%AE%A1%E7%AE%97%E5%99%A8)和其他小型系统中找到了用武之地。但是，它仍然需要外部存储设备和外部支持芯片。1978年，[国家工程制造商协会](http://zh.wikipedia.org/w/index.php?title=%E5%9B%BD%E5%AE%B6%E5%B7%A5%E7%A8%8B%E5%88%B6%E9%80%A0%E5%95%86%E5%8D%8F%E4%BC%9A&action=edit&redlink=1)发布了可编程微控制器的“标准”，包括几乎所有以计算机为基础的控制器，如[单片机](http://zh.wikipedia.org/wiki/%E5%8D%95%E7%89%87%E6%9C%BA)，[数控](http://zh.wikipedia.org/w/index.php?title=%E6%95%B0%E6%8E%A7&action=edit&redlink=1)设备，以及基于事件的控制器。

随着微控制器和微处理器的价格下降，一些消费性产品用使用微控制器的数字电路取代如[分压计](http://zh.wikipedia.org/w/index.php?title=%E5%88%86%E5%8E%8B%E8%AE%A1&action=edit&redlink=1)和[可变电容](http://zh.wikipedia.org/w/index.php?title=%E5%8F%AF%E5%8F%98%E7%94%B5%E5%AE%B9&action=edit&redlink=1)这样的昂贵[模拟](http://zh.wikipedia.org/wiki/%E9%A1%9E%E6%AF%94%E8%A8%8A%E8%99%9F)组件成为可能。

到了二十世纪八十年代中期，许多以前是外部系统的组件被集成到了处理器芯片中，这种结构的微处理器得到了更广泛的应用。到了八十年代末期，微处理器已经出现在几乎所有的电子设备中。

集成化的微处理器使得嵌入式系统的应用扩展到传统计算机无法涉足的领域。对多用途和相对低成本的微控制器进行编程，往往可成为各种不同功能的组件。虽然要做到这一点，嵌入式系统比传统的解决方案要复杂，最复杂的是在微控制器本身。但是嵌入式系统很少有额外的组件，大部分设计工作是软件部分。而非物质性的软件不管是建立原型还是测试新修改相对于硬件来说，都要容易很多的，并且设计和建造一个新的电路不会修改嵌入式处理器。

**二、特性**

嵌入式系统在广义上说就是[计算机系统](http://zh.wikipedia.org/wiki/%E8%AE%A1%E7%AE%97%E6%9C%BA%E7%B3%BB%E7%BB%9F)，它包括除了以通用为目的计算机之外的所有计算机。从便携式音乐播放器到航天飞机的实时控制子系统都能见到嵌入式系统的应用。与通用计算机系统可以满足多种任务不同，嵌入式系统只能完成某些特定目的的任务。但有些也有实时性能的制约因素必须得到满足的原因，如安全性和可用性。除此之外其他功能可能要求较低或没有要求，使系统的硬件得以简化，以降低成本。对于大批量生产的系统来说，降低成本通常是设计的首要考虑。嵌入式系统通常需要简化去除不需要的功能以降低成本，设计师通常选择刚刚满足所需功能的硬件使目标最小化低成本的实现。

嵌入式系统并非总是独立的设备。许多嵌入式系统是以一个部件存在于一个较大的设备，它为设备提供更多的功能，使设备能完成更广泛的任务。例如，吉布森吉他机器人采用了嵌入式系统来调弦，但总的来说[吉布森吉他机器人](http://zh.wikipedia.org/w/index.php?title=%E5%90%89%E5%B8%83%E6%A3%AE%E5%90%89%E4%BB%96%E6%9C%BA%E5%99%A8%E4%BA%BA&action=edit&redlink=1)设计的目的绝不是调弦而是演奏音乐。同样的，[车载电脑](http://zh.wikipedia.org/w/index.php?title=%E8%BD%A6%E8%BD%BD%E7%94%B5%E8%84%91&action=edit&redlink=1)作为汽车的一个子系统，为它提供了导航，控制，车况反馈等功能。

部分为嵌入式系统编写的程序被称为[固件](http://zh.wikipedia.org/wiki/%E5%9B%BA%E4%BB%B6)，他们存储在[只读存储器](http://zh.wikipedia.org/wiki/%E5%8F%AA%E8%AF%BB%E5%AD%98%E5%82%A8%E5%99%A8)或[闪存芯片](http://zh.wikipedia.org/w/index.php?title=%E9%97%AA%E5%AD%98%E8%8A%AF%E7%89%87&action=edit&redlink=1)。他们运行在资源有限的计算机硬件：小内存，没有键盘，甚至没有屏幕。

**2.1、用户界面**

嵌入式系统用户界面包括了从没有用户界面，专注于单一任务到现在的类似于现代桌面操作系统的复杂[图形用户界面](http://zh.wikipedia.org/wiki/%E5%9B%BE%E5%BD%A2%E7%94%A8%E6%88%B7%E7%95%8C%E9%9D%A2)的各类界面。简单的嵌入式设备往往由按钮，[发光二极管](http://zh.wikipedia.org/wiki/%E5%8F%91%E5%85%89%E4%BA%8C%E6%9E%81%E7%AE%A1)，和仅仅能显示小字符或数字的显示器，以及简单的[菜单系统](http://zh.wikipedia.org/w/index.php?title=%E8%8F%9C%E5%8D%95%E7%B3%BB%E7%BB%9F&action=edit&redlink=1)组成。较为复杂的嵌入式系统拥有的完整的图形屏幕、触摸感应或屏幕边缘按钮在最大限度地使用空间的同时，提供了足够的灵活性：指点控制是很自然的非常理想的操控方式，这样的方式可以改变的屏幕意义。手持系统通常都有一个屏幕和作为定位装置的控制按键。大部分嵌入式系统通过“维护”或者“测试”接口来提供一个菜单，或者由一个[RS-232](http://zh.wikipedia.org/wiki/RS-232)接口提供的命令行界面。这样就可以在很少的显示花费的基础上提供大量的控制手段。然而，这也会让大部分消费者无法安装必要的电缆。

**2.2、嵌入式系统的处理器**

嵌入式处理器可以分为两大类：普通微处理器和微控制器，它有更多的片上外设来降低成本和尺寸。相较于个人电脑和服务器市场，相当多的基本架构的CPU大量使用；有冯诺依曼，以及哈佛不同程度的RISC以及非RISC和VLIW架构；不同字长从4位至64位，虽然仍是最典型的16位产品。大多数架构来在不同的变种和形状，其中许多是由几个不同的公司也大量生产。

以下是一份不甚详细的架构表单：[65816](http://en.wikipedia.org/wiki/65816)， [65C02](http://en.wikipedia.org/wiki/65C02)， [68HC08](http://en.wikipedia.org/wiki/68HC08)， [68HC11](http://en.wikipedia.org/wiki/68HC11)， [68k](http://en.wikipedia.org/wiki/68k)， [8051](http://en.wikipedia.org/wiki/Intel_8051)， [ARM](http://en.wikipedia.org/wiki/ARM_architecture)， [AVR](http://en.wikipedia.org/wiki/Atmel_AVR)， [AVR32](http://en.wikipedia.org/wiki/AVR32)， [Blackfin](http://en.wikipedia.org/wiki/Blackfin)， [C167](http://en.wikipedia.org/wiki/C167_family)， [Coldfire](http://en.wikipedia.org/wiki/Coldfire)， [COP8](http://en.wikipedia.org/wiki/COP8)， [Cortus APS3](http://en.wikipedia.org/w/index.php?title=Cortus_APS3&action=edit&redlink=1)， [eZ8](http://en.wikipedia.org/wiki/Zilog_Z8)， [eZ80](http://en.wikipedia.org/wiki/EZ80)， [FR-V](http://en.wikipedia.org/wiki/FR-V)， [H8](http://en.wikipedia.org/wiki/Renesas_H8)， [HT48](http://en.wikipedia.org/wiki/HT48FXX_Flash_I/O_type_series)， [M16C](http://en.wikipedia.org/wiki/M16C)， [M32C](http://en.wikipedia.org/wiki/M32C)， [MIPS](http://en.wikipedia.org/wiki/MIPS_architecture)， [MSP430](http://en.wikipedia.org/wiki/MSP430)， [PIC](http://en.wikipedia.org/wiki/PIC_microcontroller)， [PowerPC](http://en.wikipedia.org/wiki/PowerPC)， [R8C](http://en.wikipedia.org/wiki/R8C)， [SHARC](http://en.wikipedia.org/wiki/Super_Harvard_Architecture_Single-Chip_Computer)， [ST6](http://en.wikipedia.org/wiki/ST6)， [SuperH](http://en.wikipedia.org/wiki/SuperH)， [TLCS-47](http://en.wikipedia.org/wiki/TLCS-47)， [TLCS-870](http://en.wikipedia.org/wiki/TLCS-870)， [TLCS-900](http://en.wikipedia.org/wiki/TLCS-900)， [Tricore](http://en.wikipedia.org/wiki/Tricore)， [V850](http://en.wikipedia.org/wiki/V850)， [x86](http://en.wikipedia.org/wiki/X86_architecture)， [XE8000](http://en.wikipedia.org/wiki/XE8000)， [Z80](http://en.wikipedia.org/wiki/Z80)， [AsAP](http://en.wikipedia.org/wiki/Asynchronous_array_of_simple_processors)等。

**2.3、CPU平台**

PC/104和PC/104 +是小型、低容量的嵌入式标准系统的例子，主要是基于x86的。[MicroC/OS-II](http://en.wikipedia.org/wiki/MicroC/OS-II), [QNX](http://en.wikipedia.org/wiki/QNX) 或者[VxWorks](http://en.wikipedia.org/wiki/VxWorks)经常使用DOS、Linux、[NetBSD](http://en.wikipedia.org/wiki/NetBSD)，或者是嵌入式实时操作系统。有时，使用非x86处理器。

在某些应用中，小尺寸或功率效率不是主要问题，可能与使用通用的x86兼容的个人计算机的组件有关。VIA [EPIA](http://en.wikipedia.org/wiki/EPIA)有助于弥合被个人电脑兼容，而且高度集成、体积更小、或对他们有吸引力的嵌入式工程师其他属性的差距。这种方法的优点是廉价商品的组件可以使用相同的软件开发，可以与一般的软件开发工具一起使用。系统用这种方式建立仍然被视为嵌入式是因为它们都集成到更大的设备，完成一个角色。

然而，大多数现成的嵌入式系统不是以个人电脑为中心，不使用ISA或PCI总线。当一个系统级芯片处理器有关，硬件和软件工具的环境可能会非常不同。

一个常见的设计风格采用了小型系统模块，也许是一张名片大小，设计这样的一个ARM为基础的系统级片上系统处理器和外围设备，外部存储BGA封装闪存芯片，运行内存的DRAM芯片。该模块的供应商通常会提供启动软件，并确保有一个操作系统的选择，通常包括Linux和一些实时的选择。这些模块可大批量制造，有专门的测试组织，与特定功能外围设备相结合。 [Gumstix](http://en.wikipedia.org/wiki/Gumstix)生产线是该模型的以Linux为中心的例子。

大批量生产的嵌入式系统的通常配置是[系统单芯片](http://zh.wikipedia.org/wiki/%E7%B3%BB%E7%B5%B1%E5%96%AE%E6%99%B6%E7%89%87)，它是一个[专用集成电路](http://zh.wikipedia.org/w/index.php?title=Application-specific_integrated_circuit&action=edit&redlink=1)，CPU是外购的知识产权并且添加到集成电路设计中。一个类似的常用机制是使用[可编程门阵列](http://zh.wikipedia.org/w/index.php?title=Field-programmable_gate_array&action=edit&redlink=1)，所有的逻辑包括CPU部分都可以编程实现。许多时髦的[FPGA](http://zh.wikipedia.org/wiki/FPGA)都是为这个目的设计的。

**2.4、工具**

同典型的计算机程序员一样，嵌入式系统设计人员也使用[编译器](http://zh.wikipedia.org/wiki/%E7%BC%96%E8%AF%91%E5%99%A8)、[连结器](http://zh.wikipedia.org/wiki/%E9%80%A3%E7%B5%90%E5%99%A8)和[调试器](http://zh.wikipedia.org/wiki/%E8%AA%BF%E8%A9%A6%E5%99%A8)开发嵌入式系统软件。然而，他们也使用一些大多数程序员不熟悉的工具。

软件工具的来源有几种：专注于嵌入式系统市场的软件公司；从[GNU](http://zh.wikipedia.org/wiki/GNU)软件开发工具移植；有些情况下，如果嵌入式处理器与普通个人计算机处理器很近似的话也可以使用个人计算机开发工具。

嵌入式系统设计人员也使用一些不为普通计算机程序员所熟悉的软件工具：

①连结器通常是各种各样。对于大多数商业编程来说，连接器几乎总是最后才想起的部分，缺省设置也从来不变。与此相反，嵌入式连结器有完整、复杂的命令行语言是很普通的。经常有不同类型的内存，分别保存特殊的代码和数据。单独的数据结构能够放在特殊的地址，这样软件能够很方便地访问映射到内存的控制寄存器。嵌入式连结器经常是用于减小代码大小和运行时间的外部优化工具。例如，他们可能移动子程序的位置以使用较小的调用和跳转指令。它们经常带有管理[data overlays](http://en.wikipedia.org/wiki/data_overlays)和[band switch](http://en.wikipedia.org/wiki/band_switch)技术的特性，这些技术是在嵌入式软件经常使用的扩展廉价CPU的方法。

②另外一个常用的工具是一个在程序中添加代码或者[CRC](http://en.wikipedia.org/wiki/CRC)的工具程序，使用这个工具嵌入式系统能够在执行程序之前先进行程序数据检查。

③为[数字信号处理](http://zh.wikipedia.org/wiki/%E6%95%B0%E5%AD%97%E4%BF%A1%E5%8F%B7%E5%A4%84%E7%90%86)开发软件的嵌入式程序员经常使用[MatchCad](http://en.wikipedia.org/wiki/MatchCad)或者[Mathematica](http://en.wikipedia.org/wiki/Mathematica)这样的数学工具进行数学仿真。

④一些较少使用的工具有将数据文件转换成代码的工具，使用这种工具就可以在程序中包含任意类型的数据。

⑤少数一些项目为了特殊的可靠性或者[数字信号处理](http://zh.wikipedia.org/wiki/%E6%95%B0%E5%AD%97%E4%BF%A1%E5%8F%B7%E5%A4%84%E7%90%86)要求使用[同步编程语言](http://en.wikipedia.org/wiki/Synchronous_programming_language)。

### 三、网络文件表

[

A\_key

RAD0.2

KA3

]

[

B\_key

RAD0.2

KB3

]

[

C1

SIP2

10UF/16V

]

[

C2

SIP2

20P

]

[

C3

SIP2

20P

]

[

C\_key

RAD0.2

KC3

]

[

Designator1

DIP16

LCD 1602

]

[

Door\_control

RAD0.2

KA4

]

[

J1

SIP3

CON3

]

[

J2

SIP2

CON2

]

[

J3

DIP40

STC89C52

]

[

LS1

RAD0.2

SPEAKER

]

[

Mod\_set

RAD0.2

S2

]

[

Q8

SIP3

8550

]

[

R6

AXIAL0.3

10K

]

[

R8

AXIAL0.3

2.2K

]

[

R23

AXIAL0.3

30

]

[

REST1

RAD0.2

]

[

S3

RAD0.2

ON/OFF

]

[

Start

RAD0.2

KB4

]

[

W1

TO-39

10K

]

[

Water\_line\_set

RAD0.2

S1

]

[

Y2

SIP2

12MHZ

]

(

GND

C2-2

C3-1

Designator1-1

Designator1-16

J3-20

LS1-2

Mod\_set-1

R6-2

W1-2

Water\_line\_set-1

)

(

NetA\_key\_1

A\_key-1

Door\_control-1

J3-1

)

(

NetA\_key\_2

A\_key-2

B\_key-2

C\_key-2

J3-7

)

(

NetB\_key\_1

B\_key-1

J3-2

Start-1

)

(

NetC1\_2

C1-2

J3-9

R6-1

REST1-2

)

(

NetC2\_1

C2-1

J3-18

Y2-2

)

(

NetC3\_2

C3-2

J3-19

Y2-1

)

(

NetC\_key\_1

C\_key-1

J3-3

)

(

NetDoor\_control\_2

Door\_control-2

J3-8

Start-2

)

(

NetJ1\_1

J1-1

J3-17

)

(

NetJ1\_2

J1-2

J3-11

)

(

NetJ1\_3

J1-3

J3-10

)

(

NetJ3\_21

Designator1-4

J3-21

)

(

NetJ3\_22

Designator1-5

J3-22

)

(

NetJ3\_23

Designator1-6

J3-23

)

(

NetJ3\_24

J2-2

J3-24

)

(

NetJ3\_25

J2-1

J3-25

)

(

NetJ3\_26

J3-26

)

(

NetJ3\_27

J3-27

)

(

NetJ3\_28

J3-28

)

(

NetJ3\_32

Designator1-14

J3-32

)

(

NetJ3\_33

Designator1-13

J3-33

)

(

NetJ3\_34

Designator1-12

J3-34

)

(

NetJ3\_35

Designator1-11

J3-35

)

(

NetJ3\_36

Designator1-10

J3-36

)

(

NetJ3\_37

Designator1-9

J3-37

)

(

NetJ3\_38

Designator1-8

J3-38

)

(

NetJ3\_39

Designator1-7

J3-39

)

(

NetMod\_set\_2

J3-13

Mod\_set-2

)

(

NetQ8\_1

Q8-1

S3-2

)

(

NetQ8\_3

LS1-1

Q8-3

)

(

NetR8\_1

J3-15

R8-1

)

(

NetR8\_2

R8-2

S3-1

)

(

NetR23\_2

Designator1-15

R23-2

)

(

NetW1\_3

Designator1-3

W1-3

)

(

NetWater\_line\_set\_2

J3-12

Water\_line\_set-2

)

(

VCC

C1-1

Designator1-2

J3-31

J3-40

Q8-2

R23-1

REST1-1

W1-1

)

### 四、源代码

#include<reg52.h>

#include<intrins.h>

#define uchar unsigned char

#define uint unsigned int

#define high 0x01 //水位高、中、低

#define mid 0x02

#define low 0x03

#define strong 0x04 //洗涤模式强、中、弱

#define middle 0x05

#define feeble 0x06

#define once 0x07 //清洗模式一清、两清

#define twice 0x08

//key\_input define

sbit key\_water\_line=P3^2; //水位按键输入引脚定义

sbit key\_mode=P3^3; //模式按键输入引脚定义

sbit door\_cotrol\_row=P1^7; //门控制按键输入引脚定义

sbit door\_cotrol\_line=P1^0;

sbit start\_row=P1^7; //开始按键输入引脚定义

sbit start\_line=P1^1;

sbit A\_key\_row=P1^6; //LCD选择A按键输入引脚定义

sbit A\_key\_line=P1^0;

sbit B\_key\_row=P1^6; //LCD选择B按键输入引脚定义

sbit B\_key\_line=P1^1;

sbit C\_key\_row=P1^6; //LCD选择C按键输入引脚定义

sbit C\_key\_line=P1^2;

sbit high\_detector\_row=P1^7; //高水位探测器按键引脚定义

sbit high\_detector\_line=P1^2;

sbit mid\_detector\_row=P1^7; //中水位探测器按键引脚定义

sbit mid\_detector\_line=P1^3;

sbit low\_detector\_row=P1^6; //低水位探测器按键引脚定义

sbit low\_detector\_line=P1^3;

//pin define

sbit pin\_buzzer=P3^5; //蜂鸣器控制信号引脚定义

sbit pin\_high\_detector=P2^7; //高水位探测器反馈信号引脚定义，低电平有效

sbit pin\_mid\_detector=P2^6; //中水位探测器反馈信号引脚定义，低电平有效

sbit pin\_low\_detector=P2^5; //低水位探测器反馈信号引脚定义，低电平有效

sbit pin\_clockwise\_washing=P2^4; //正转洗涤接触器控制信号引脚定义

sbit pin\_anticlockwise\_washing=P2^3; //反转洗涤接触器控制信号引脚定义

sbit pin\_infall=P3^0; //进水电磁阀控制信号引脚定义，低电平有效

sbit pin\_scupper=P3^1; //排水电磁阀控制信号引脚定义，低电平有效

sbit pin\_drier=P3^7; //脱水离合器控制信号引脚定义，低电平有效

//LCD1602引脚设定

sbit rs=P2^0;

sbit rw=P2^1;

sbit ep=P2^2;

sbit DB0=P0^0;

sbit DB1=P0^1;

sbit DB2=P0^2;

sbit DB3=P0^3;

sbit DB4=P0^4;

sbit DB5=P0^5;

sbit DB6=P0^6;

sbit DB7=P0^7;

//LCD显示字符数组初始化

uchar code dis0[]={""};

uchar code dis1[]={"Auto washer"};

uchar code dis2[]={"Made in China"};

uchar code dis3[]={"START"};

uchar code dis4[]={"POURING WATER"};

uchar code dis5[]={"WASHING"};

uchar code dis6[]={"RINSING"};

uchar code dis7[]={"FINISH"};

uchar code dis8[]={"DRY"};

uchar code dis9[]={"WAIT"};

uchar code dis10[]={"Water\_line set"};

uchar code dis11[]={"A: H B: M C: L"};

uchar code dis12[]={"Wash\_mode set"};

uchar code dis13[]={"A: S B: M C: F"};

uchar code dis14[]={"rinse\_mode set"};

uchar code dis15[]={"A:once B:twice"};

bit door\_cotrol=1; //门控制按键按下，该值变为0；

bit start=1; //开始按键按下，该值变为0；

bit A\_key=1; //选择键A按下，该值变为0；

bit B\_key=1; //选择键B按下，该值变为0；

bit C\_key=1; //选择键C按下，该值变为0；

uchar water\_line=mid; //水位存储单元，初始化为中水位

uchar wash\_mode=strong; //洗涤模式存储单元，初始化为强洗

uchar rinse\_mode=once; //清洗模式存储单元，初始化为一清

uchar frq; //蜂鸣器警报鸣叫变量

void delay250us(uint i) // 该延时子函数延时250\*i(us)

{

TMOD=0x00;

TR0=1;

while(i!=0)

{

TH0=0x1F; //T0延时250us计数初值

TL0=0X06;

while(TF0!=1);

TF0=0;

i--;

}

TR0=0;

}

void delay50ms(uint i) //该延时子函数延时50\*i（ms）

{

TMOD=0x01;

TR0=1;

while(i!=0)

{

TH0=0x3C; //T0延时50ms计数初值

TL0=0xB0;

while(TF0!=1);

TF0=0;

i--;

}

TR0=0;

}

uchar keyscan(void) //按键扫描函数

{

uchar key;

P1=0x0F; //低四位输入

delay250us(1);

if(door\_cotrol\_line==0||A\_key\_line==0)

key=0;

else if(start\_line==0||B\_key\_line==0)

key=1;

else if(high\_detector\_line==0||C\_key\_line==0)

key=2;

else if(mid\_detector\_line==0||low\_detector\_line==0)

key=3;

else

key=16;

P1=0xF0; //高四位输入

delay250us(1);

if(door\_cotrol\_row==0||start\_row==0||high\_detector\_row==0||mid\_detector\_row==0)

key=key+12;

else if(A\_key\_row==0||B\_key\_row==0||C\_key\_row==0||low\_detector\_row==0)

key=key+8;

else

key=16;

return key;

}

void keydown(void) //判断键是否按下

{

P1=0xF0;

while(P1==0xF0){}

switch(keyscan())

{

case 12:door\_cotrol=0;break;

case 13:start=0;break;

case 8:A\_key=0;break;

case 9:B\_key=0;break;

case 10:C\_key=0;break;

case 11:pin\_low\_detector=0;break;

case 14:pin\_high\_detector=0;break;

case 15:pin\_mid\_detector=0;break;

default:break;

}

}

bit lcd\_bz()

{ // 测试LCD忙碌状态

bit result;

rs = 0;

rw = 1;

ep = 1;

\_nop\_();

\_nop\_();

\_nop\_();

\_nop\_();

result = (bit)(P0 & 0x80);

ep = 0;

return result;

}

void lcd\_wcmd(uchar cmd)

{ // 写入指令数据到LCD

while(lcd\_bz());

rs = 0;

rw = 0;

ep = 0;

\_nop\_();

\_nop\_();

P0 = cmd;

\_nop\_();

\_nop\_();

\_nop\_();

\_nop\_();

ep = 1;

\_nop\_();

\_nop\_();

\_nop\_();

\_nop\_();

ep = 0;

}

void lcd\_pos(uchar pos)

{ //设定显示位置

lcd\_wcmd(pos);

}

void lcd\_wdat(uchar dat)

{ //写入字符显示数据到LCD

while(lcd\_bz());

rs = 1;

rw = 0;

ep = 0;

P0 = dat;

\_nop\_();

\_nop\_();

\_nop\_();

\_nop\_();

ep = 1;

\_nop\_();

\_nop\_();

\_nop\_();

\_nop\_();

ep = 0;

}

void lcd\_init() //LCD初始化设定

{

delay250us(50); //等待LCD电源稳定

lcd\_wcmd(0x38); //16\*2显示，5\*7点阵，8位数据

delay250us(15);

lcd\_wcmd(0x38);

delay250us(15);

lcd\_wcmd(0x38);

delay250us(15);

lcd\_wcmd(0x0c); //显示开，关光标

delay250us(15);

lcd\_wcmd(0x06); //移动光标

delay250us(15);

lcd\_wcmd(0x01); //清除LCD的显示内容

delay250us(15);

}

void LCD\_display(uchar pos1，uchar code first[]，uchar pos2，uchar code second[]) //LCD显示调用子函数

{

uchar i;

lcd\_init(); // 初始化LCD

delay250us(10);

lcd\_pos(pos1); // 设置第一行的显示位置pos1

i = 0;

while(first[i] != '\0')

{ // 显示first

lcd\_wdat(first[i]);

i++;

}

lcd\_pos(pos2); // 设置第二行的显示位置pos2

i = 0;

while(second[i] != '\0')

{

lcd\_wdat(second[i]); // 显示second

i++;

}

delay250us(100);

}

void LCD\_display\_move(uchar pos，uchar code dischar[]，uint string) //LCD移动显示子函数

{

uchar i，j;

uint k=2;

lcd\_init(); // 初始化LCD

delay250us(10);

lcd\_pos(pos); // 设置第一行的显示位置pos1

i = 0;

while(dischar[i] != '\0')

{ // 显示first

lcd\_wdat(dischar[i]);

i++;

}

lcd\_pos(pos|0x40); // 设置第二行的显示位置pos2

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

{

lcd\_wdat('\*'); // 显示second

}

while(k)

{

for(j=0;j<(16-string);j++)

{

lcd\_wcmd(0x18);

delay50ms(2);

}

for(j=0;j<(16-string);j++)

{

lcd\_wcmd(0x1c);

delay50ms(2);

}

k--;

}

//delay250us(100);

}

void infall(uchar depth) //进水功能函数

{

switch(depth)

{

case high:pin\_infall=0; //开启进水电磁阀，开始进水

LCD\_display(0x85，dis9，0xC0，dis0);

keydown();

while(pin\_high\_detector)

{

keydown();

} //等待进水至高水位

pin\_high\_detector=1; //复位高水位探测器反馈

pin\_infall=1; //关闭进水电磁阀，停止进水

break;

case mid:pin\_infall=0;

LCD\_display(0x85，dis9，0xC0，dis0);

keydown();

while(pin\_mid\_detector)

{

keydown();

} //等待进水至中水位

pin\_mid\_detector=1; //复位中水位探测器反馈

pin\_infall=1;

break;

case low:pin\_infall=0;

LCD\_display(0x85，dis9，0xC0，dis0);

keydown();

while(pin\_low\_detector)

{

keydown();

} //等待进水至低水位

pin\_low\_detector=1; //复位低水位探测器反馈

pin\_infall=1;

break;

}

}

void washing(uchar washing) //洗涤功能函数

{

uint i;

switch(washing)

{

case strong: //强洗

for(i=0;i<20;i++)

{

pin\_clockwise\_washing=0; //开始正转洗涤接触器

delay50ms(560); //延时28s，即正转洗涤28s

pin\_clockwise\_washing=1; //关闭正转洗涤接触器

delay50ms(40); //延时2s

pin\_anticlockwise\_washing=0; //开始反转洗涤接触器

delay50ms(560);

pin\_anticlockwise\_washing=1;

delay50ms(40);

}

break;

case middle: //中洗

for(i=0;i<10;i++)

{

pin\_clockwise\_washing=0;

delay50ms(560);

pin\_clockwise\_washing=1;

delay50ms(40);

pin\_anticlockwise\_washing=0;

delay50ms(560);

pin\_anticlockwise\_washing=1;

delay50ms(40);

}

break;

case feeble:

for(i=0;i<5;i++) //弱洗

{

pin\_clockwise\_washing=0; //开始正转洗涤接触器

delay50ms(560); //延时28s，即正转洗涤28s

pin\_clockwise\_washing=1; //关闭正转洗涤接触器

delay50ms(40); //延时2s

pin\_anticlockwise\_washing=0; //开始反转洗涤接触器

delay50ms(560);

pin\_anticlockwise\_washing=1;

delay50ms(40);

}

break;

}

}

void scupper(uchar depth) //排水功能函数

{

switch(depth)

{

case high: //高水位排水

pin\_scupper=0; //开启排水电磁阀

delay50ms(120); //6s

pin\_scupper=1; //关闭排水电磁阀

break;

case mid:

pin\_scupper=0;

delay50ms(80); //4s

pin\_scupper=1;

break;

case low: //2s

pin\_scupper=0;

delay50ms(40);

pin\_scupper=1;

break;

}

}

void dry30\_120(uint dry\_time) //脱水功能函数

{

pin\_drier=0; //开启脱水离合器

delay50ms(20\*dry\_time); //延时dry\_time秒，即脱水dry\_time段时间

pin\_drier=1; //关闭脱水离合器

}

void rinse(uchar rinse)

{

uint i，rinse\_time=1;

if(rinse==twice)

rinse\_time=2;

for(i=1;i<=rinse\_time;i++)

{

infall(water\_line);

LCD\_display\_move(0x89，dis6，7); //LCD显示清洗

for(i=0;i<5;i++)

{

pin\_clockwise\_washing=0; //开始正转洗涤接触器

delay50ms(560); //延时28s，即正转洗涤28s pin\_clockwise\_washing=1; //关闭正转洗涤接触器

delay50ms(40); //延时2s

pin\_anticlockwise\_washing=0; //开始反转洗涤接触器

delay50ms(560);

pin\_anticlockwise\_washing=1;

delay50ms(40);

}

scupper(water\_line);

}

}

void delayms(unsigned char ms) // 延时子程序，蜂鸣器使用

{

unsigned char i;

while(ms--)

{

for(i = 0; i < 120; i++);

}

}

void buzzer() //蜂鸣器警报

{

TMOD=0x10;

frq=0x00;

TH1=0x00;

TL1=0xff;

TR1=1;

IE=0x88;

while(20000)

{

frq++;

delayms(1);

}

TR1=0;

}

void timer1() interrupt 3 using 1

{

TH0=0xfe;

TL0=frq;

pin\_buzzer=~pin\_buzzer;

}

void water\_line\_set() interrupt 0 using 2

{

LCD\_display(0x81，dis10，0xC1，dis11); //LCD显示水位设定：高（high）中（mid）低（low）

keydown();

if(A\_key==0)

water\_line=high;

else if(B\_key==0)

water\_line=mid;

else

water\_line=low;

LCD\_display(0x82，dis1，0xC1，dis2);

}

void mode\_set() interrupt 2 using 3

{

LCD\_display(0x81，dis12，0xC1，dis13); //LCD显示洗涤模式设定：强（strong）中（middle）弱（feeble）

keydown();

if(A\_key==0)

wash\_mode=strong;

else if(B\_key==0)

wash\_mode=middle;

else

wash\_mode=feeble;

LCD\_display(0x81，dis14，0xC1，dis15); //LCD显示清洗模式设定：一清（once）二清（twice）

keydown();

if(A\_key==0)

rinse\_mode=once;

else

rinse\_mode=twice;

LCD\_display(0x82，dis1，0xC1，dis2);

}

void main()

{

P0=0xff; //设置初始引脚值，变量值

P1=0xff;

P2=0xff;

P3=0xff;

IE=0x85; //开启外部中断

IP=0x00;

LCD\_display(0x82，dis1，0xC1，dis2);

keydown(); //扫描门控制键是否按下

while(door\_cotrol!=0){keydown();}

door\_cotrol=1; //恢复门控制键

keydown(); //扫描开始键是否按下

while(start!=0){keydown();}

IE=0x00; //关中断

start=1; //恢复开始键

LCD\_display(0x85，dis3，0xC0，dis0);

delay50ms(20);

LCD\_display\_move(0x83，dis4，13); //LCD显示进水

infall(water\_line);

delay50ms(40);

LCD\_display\_move(0x89，dis5，7); //LCD显示洗涤中

washing(wash\_mode);

scupper(water\_line);

delay50ms(40);

dry30\_120(30);

LCD\_display\_move(0x89，dis6，7); //LCD显示清洗

rinse(rinse\_mode);

delay50ms(40);

LCD\_display\_move(0x8d，dis8，3); //LCD显示脱水中

dry30\_120(120);

LCD\_display(0x85，dis7，0xC0，dis0); //LCD显示完成

buzzer();

}