教　案

2025-2026 学年第一学期

课程名称　单片机与接口技术

专业班级　机电一体化技术（现代学徒制）241

总学时数　64　学时

任课教师　吴佳楷

课程基本信息

<table>
<tr><td>课程名称</td><td colspan="4">单片机与接口技术</td></tr>
<tr><td>课程性质</td><td colspan="2">专业必修</td><td>学分</td><td>4.0</td></tr>
<tr><td>学　时</td><td colspan="4">总学时：64 学时，其中：课堂讲授 64 学时；课内实验 0 学时。</td></tr>
<tr><td>开课部门</td><td colspan="2">机电工程系</td><td>任课教师</td><td>吴佳楷</td></tr>
<tr><td>授课
专业、班级</td><td colspan="2">机电一体化技术（现代学徒制）241</td><td>开课学期</td><td>2025-2026 第一学期</td></tr>
<tr><td>成绩评定</td><td colspan="2">平时成绩占 50 %；期末成绩占 50 %</td><td>考核方式</td><td>考试</td></tr>
<tr><td rowspan="2">选用教材</td><td>书　　名</td><td>主 编</td><td colspan="2">出版社</td><td>出版日期</td></tr>
<tr><td>单片机应用技术（C51 版）</td><td>庄乾成</td><td colspan="2">机械工业出版社</td><td>2024.1</td></tr>
<tr><td>本课程在本专业人才培养方案中的地位和作用</td><td colspan="4">单片机应用技术通常是电子工程、自动化、计算机科学与技术等相关专业的一门核心课程，因为它涵盖了硬件设计与软件编程的基础知识，是连接理论与实践的重要桥梁。随着物联网、嵌入式系统、智能控制等领域的发展，单片机技术成为了掌握最新技术趋势的关键之一。单片机应用技术课程让学生有机会将学到的理论知识应用到实践中，通过实验和项目加深对硬件电路和软件编程的理解。</td></tr>
<tr><td>本课程
教学目标</td><td colspan="4">通过学习，学生应掌握单片机的基本原理、内部结构及工作模式，理解单片机的指令系统和编程方法，能够编写基本的控制程序能够设计简单的单片机外围电路，并理解其工作原理。</td></tr>
<tr><td>素质（思政）
内容与要求</td><td colspan="4">将思政元素融入《单片机与接口技术》课程，旨在培养学生的爱国情怀、科学精神、辩证思维以及社会责任感。
一、结合我国的发展历程，讲述单片机技术在我国工业现代化过程中的贡献，激发学生的爱国情怀；</td></tr>
</table>

	二、鼓励学生探索新技术，培养创新意识。例如，通过介绍一些成功的单片机应用案例，让学生了解到持续学习和技术革新对于个人和社会的重要性； 三、讲解我国相关领域科学家和技术人员为国家科技进步做出的贡献，鼓励学生为国家的发展贡献力量。
学生用主要参考资料	庄乾成，单片机应用技术（C51 版），机械工业出版社，2024.

第一章　单片机概述（4 学时）

本章主要内容

介绍单片机基础知识、发展历史、应用领域以及发展趋势。

8 位单片机的主流机型，MCS-51 系列单片机及其兼容的单片机（统称为 8051 单片机）

对目前流行的 8051 单片机的代表性机型：美国 ATMEL 公司的 AT89C5x/AT89S5x 系列单片机及代表性产品 AT89S51 详细介绍。结构清晰，易掌握，初学者入门机型。

简要介绍其它类型的单片机。

本章课时分配　本章分为 2 讲，共 4 学时。

1.1~1.6 单片机简介、发展历史、特点、应用、发展趋势

教学目标：（教学 2 学时）

了解单片机基础知识、发展历史、应用领域以及发展趋势。

教学重点

1．单片机的概念与结构；

2. STC 系列单片机

教学难点

1．单片机的构成、各种总类的单片机

素质（思政）内容与要求：

思政融入点：结合我国的发展历程，讲述单片机技术在我国工业现代化过程中的贡献，激发学生的爱国情怀。

案例：通过讲解单片机在农业中的应用，帮助学生认识到科技进步对于解决食品安全、环境保护等社会问题的重要性。

教学手段：讲授

主要内容：

1.1　单片机简介

一片半导体硅片集成：中央处理单元（CPU）、存储器（RAM、ROM）、并行 I/O、串行 I/O、定时器/计数器、中断系统、系统时钟电路及系统总线的微型计算机。

具有微型计算机属性，因而被称为单片微型计算机，简称单片机。

主要应用测控领域。单片机处于测控系统的核心地位并嵌入其中，所以国际上通常把单片机称为嵌入式控制器（EMCU，Embedded MicroController Unit），或微控制器（MCU，MicroController Unit）。

我国习惯于使用“单片机”这一名称。

1.2　单片机的发展历史

按处理二进制位数主要分为：4 位单片机、8 位单片机、16 位单片机和 32 位单片机。

发展大致分为 4 个阶段。

第一阶段（1974 年～1976 年）：单片机初级阶段。因工艺限制，双片形式且功能较简单。1974 年 12 月，仙童公司推出了 8 位的 F8 单片机，实际只包括了 8 位 CPU、64B RAM 和 2 个并行口。

第二阶段（1976 年～1978 年）：低性能单片机阶段。1976 年 Intel 的 MCS-48 单片机（8 位）极大地促进了单片机变革和发展，1977 年 GI 公司推出 PIC1650，但这个阶段仍处于低性能阶段。

第三阶段（1978 年～1983 年）：高性能单片机阶段。1978 年，Zilog 公司推出 Z8 单片机，1980 年，Intel 公司在 MCS-48 系列基础上推出 MCS-51 系列，Mortorola 推出 6801 单片机。使单片机性能及应用跃上新台阶。

第四阶段（1983 年～现在）：8 位单片机巩固发展及 16 位单片机、32 位单片机推出阶段。

1.3 单片机的特点

单片机是集成电路技术与微型计算机技术高速发展的产物。**体积小、价格低、应用方便、稳定可靠**，因此，给工业自动化等领域带来了一场重大革命和技术进步。

由于体积小，很容易地嵌入到系统之中，以实现各种方式的检测、计算或控制，这一点，一般微机根本做不到。

1.4 单片机的应用

软硬件结合、体积小，容易嵌入到各种应用系统中。得到广泛应用。

1．工业检测与控制

主要应用：工业过程控制、智能控制、设备控制、数据采集和传输、测试、测量、监控等。在工业自动化领域中，机电一体化技术将发挥愈来愈重要的作用，在这种集机械、微电子和计算机技术为一体的综合技术（如机器人技术）中，单片机发挥着非常重要作用。

2．仪器仪表

目前对仪器仪表的自动化和智能化要求越来越高。单片机的使用有助于提高仪器仪表的精度和准

确度，简化结构，减小体积而易于携带和使用，加速仪器仪表向数字化、智能化、多功能化方向发展。

3．消费类电子产品

例如，洗衣机、电冰箱、空调机、电风扇、电视机、微波炉、加湿机、消毒柜等。嵌入了单片机后，功能和性能大大提高，并实现智能化、最优化控制。

4．通信

在调制解调器、各类手机、传真机、程控电话交换机、信息网络及各种通讯设备中，单片机也已得到广泛应用。

5．武器装备

现代化武器装备，如飞机、军舰、坦克、导弹、鱼雷制导、智能武器装备、航天飞机导航系统，都有单片机嵌入其中。

6．各种终端及计算机外部设备

计算机网络终端（如银行终端）及计算机外部设备（如打印机、硬盘驱动器、绘图机、传真机、复印机等）中都使用了单片机作为控制器。

7．汽车电子设备

已广泛应用在各种汽车电子设备中，如汽车安全系统、汽车信息系统、智能自动驾驶系统、卫星汽车导航系统、汽车紧急请求服务系统、汽车防撞监控系统、汽车自动诊断系统及汽车黑匣子等。

8．分布式多机系统

在较复杂多节点测控系统中，常采用分布式多机系统。

1.5　单片机的发展趋势

单片机发展趋势将是向大容量、高性能化，外围电路内装化等方面发展。为满足不同用户要求，各公司竞相推出能满足不同需要的产品。

1．CPU 的改进

2．存储器的发展

3．片内 I/O 的改进

4．低功耗

5．外围电路内装化

6．编程及仿真的简单化

综上所述，单片机正在向多功能、高性能、高速度（时钟达 40MHz）、低电压（2.7V 即可工作）、低功耗、低价格（几元钱）、外围电路内装化以及片内程序存储器和数据存储器容量不断增大的方向发展。

1.6 MCS-51 系列与 AT89C5x 系列单片机

20 世纪 80 年代以来，单片机发展非常迅速，其中 Intel 公司的 MCS-51 系列单片机是一款设计成功、易于掌握并在世界范围得到广泛使用的机型。

1.6.1 MCS-51 系列单片机

MCS 是 Intel 公司单片机的系列符号，如 MCS-48、MCS-51、MCS-96 系列单片机。

MCS-51 系列单片机主要包括

基本型：8031/8051/8751（低功耗型 80C31/80C51/87C51）

增强型：8032/8052/8752。

已为我国广大技术人员所熟悉和掌握。上世纪 80 年代和 90 年代，MCS-51 系列是在我国应用最为广泛的机型之一。

MCS-51 系列品种丰富，经常使用的是基本型和增强型

本节作业：

课本 P13 页

一、

二 1、2、

四、1、2、3

1.7~1.9 各种衍生品种的 8051 单片机 、嵌入式 DSP 处理器

教学目标：（教学 2 学时）

了解各种衍生品种的 8051 单片机、简单了解嵌入式 DSP 处理器。

教学重点

1、STC 系列单片机的主要性能与特点。

2．嵌入式处理器的概念；

教学难点

1．嵌入式处理器的概念

素质（思政）内容与要求：

思政融入点：强调精益求精的态度，通过讲解单片机的不同系列在实际应用中的细节处理，培养学生认真负责的工作态度。

案例：通过讲解单片机在不同邻域的应用，帮助学生了解和掌握单片机对我国工业发展的重要性。

教学手段：讲授

主要内容：

1.7 各种衍生品种的 8051 单片机

除 AT89S5x 系列单片机外，世界各器件厂家推出的以 8051 为内核、各种集成度高、功能强的单片机，也得到广大用户青睐。

1.7.1 STC 系列单片机

STC 系列具有我国独立自主知识产权，功能与抗干扰性强的增强型 8051 单片机，多种子系列，几百个品种，以满足不同需要。其中的 STC12C5410/STC12C2052 系列的主要性能及特点如下。

（1）高速：传统 8051 为每个机器周期为 12 个时钟，而 STC 可为每机器周期 1 个时钟，指令执行速度大大提高，速度比普通 8051 快 8～12 倍。

（2）宽工作电压：5.5～3.8V，2.4～3.8V（STC12LE5410AD 系列）

（3）12KB/10KB/8KB/6KB/4KB 片内 Flash 程序存储器，擦写次数 10 万次以上。

（4）512B 片内的 RAM 数据存储器。

（5）可在线编程（ISP）/在应用可编程（IAP），无需编程器/仿真器，可远程升级。

（6）8 通道 10 位 ADC，4 路 PWM 输出。

（7）4 通道捕捉/比较单元，也可用来再实现 4 个定时器或 4 个外部中断。

（8）2 个硬件 16 位定时器，兼容 8051 定时器。4 路 PCA 还可再实现 4 个定时器。

（9）硬件看门狗（WDT）。

（10）高速 SPI 串口。

（11）全双工异步串行口(UART)，兼容普通 8051 的串口。

（12）通用 I/O 口（27/23/15 个），复位后为：准双向口/弱上拉（与 8051 的 I/O 接口相似）。可设置成四种模式：准双向口/弱上拉，推挽/强上拉，仅为输入/高阻，开漏，每个 I/O 口驱动能力均可达到 20mA，但整个芯片最大不可超过 55mA。

（13）超强抗干扰能力与高可靠性：

高抗静电；

通过 2kV/4kV 快速脉冲干扰的测试（EFT 测试）；

宽电压，不怕电源抖动；

宽温度范围：−40℃～+85℃；

I/O 口经过特殊处理；

片内的电源供电系统、时钟电路、复位电路、看门狗电路均经过特殊处理；

（14）采取了降低单片机时钟对外部电磁辐射的措施：

可禁止 ALE 输出；

如选每个机器周期为 6 个时钟，外部时钟频率可降一半；

单片机时钟振荡器增益可设为 Gain。

（15）超低功耗设计

掉电模式：典型功耗＜0.1μA；

空闲模式：典型功耗为 2mA；

正常工作模式：典型功耗为 4mA～7mA；

掉电模式可由外部中断唤醒，适用于电池供电系统，如水表、气表、便携设备等。

1.9 其它的嵌入式处理器简介

以各类嵌入式处理器为核心的嵌入式系统的应用，已成为当今电子信息技术应用的一大热点。

嵌入式处理器按体系结构主要分为如下几类：嵌入式微控制器（单片机）、嵌入式数字信号处理器（简称 DSP）及嵌入式微处理器。

1.9.1 嵌入式 DSP 处理器(DSP)

嵌入式数字信号处理器（Digital Signal Processor，DSP)，简称数字信号处理器（DSP）。

非常擅长于高速实现各种数字信号处理运算（如数字滤波、FFT、频谱分析等）的嵌入式处理器。由于对 DSP 硬件结构和指令进行了特殊设计，使其能高速完成各种数字信号处理算法。

1981 年，美国 TI（Texas Instruments)）公司研制出了著名的 TMS320 系列的首片低成本、

高性能的 DSP 处理器芯片：TMS320C10，使 DSP 技术向前跨出了意义重大的一步。

1.9.2 嵌入式微处理器

嵌入式微处理器（Embedded MicroProcessor Unit，EMPU）的基础是通用计算机中的 CPU。虽在功能和标准微处理器基本一样，但由于只保留和嵌入式应用有关的功能，这样可大幅度减小系统体积和功耗，同时在工作温度、抗电磁干扰、可靠性等方面一般都做了各种增强处理。

这里要对“嵌入式系统”这个名称作进一步说明。从更广泛意义讲，凡是系统中嵌入了“嵌入式处理器”，如单片机、DSP、嵌入式微处理器，都称其为“嵌入式系统”。

但目前较为流行的说法是，仅把“嵌入”嵌入式微处理器的系统，称为“嵌入式系统”。目前“嵌入式系统”还没有一个严格和权威定义，但通常所说的“嵌入式系统”，多指后者。

本节作业：

课本 P14 页

三

四、4、5、6

第二章　AT89S52 单片机的片内硬件结构（8 学时）

本章主要内容

本章介绍 AT89S52 单片机的片内硬件结构。

解并熟知 AT89S52 单片机的片内硬件结构，以及片内外设资源的工作原理与基本功能，重点掌握 AT89S52 单片机的存储器结构、常见的特殊功能寄存器的基本功能以及复位电路与时钟电路的设计，掌握单片机最小系统的概念。

本章课时分配　本章分为 2 讲，共 8 学时。

2.1~2.4 AT89S52 单片机的片内硬件结构

教学目标：（教学 4 学时）

了解 AT89S52 单片机的片内硬件结构，以及片内外设资源的工作原理与基本功能。

教学重点

1. 掌握 AT89S52 单片机的存储器结构；
2. 常见的特殊功能寄存器的基本功能
3. AT89S52 的引脚功能

教学难点

1. 常见的特殊功能寄存器的基本功能

素质（思政）内容与要求：

思政融入点：讲解 AT89S52 单片机的硬件结构时，强调每部分设计的精密与复杂，引导学生认识到任何一项技术成果的背后都是无数工程师辛勤工作的结果。

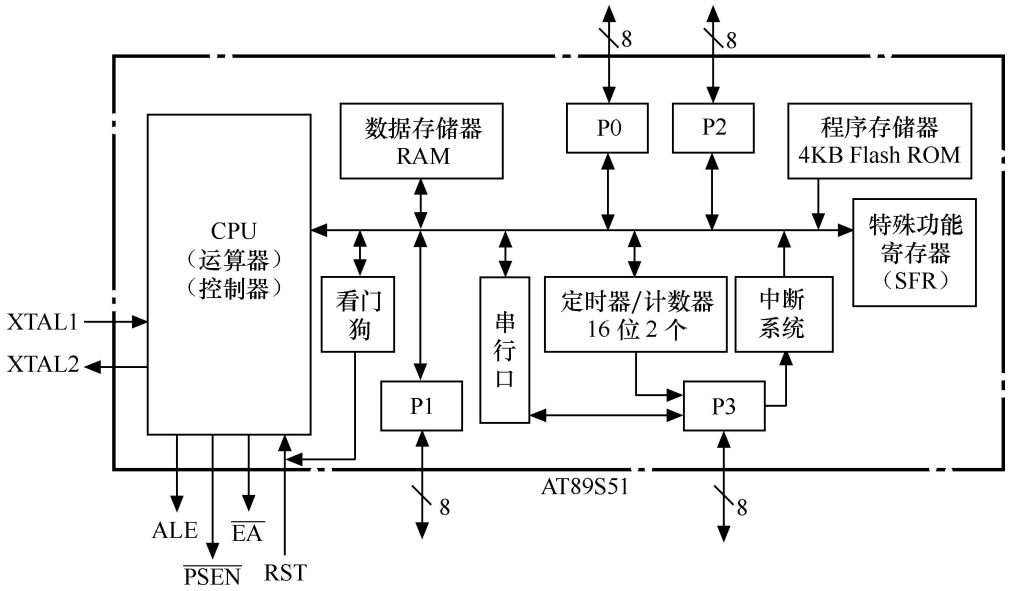
案例：介绍一些国内企业或研发团队在单片机设计与制造过程中精益求精的故事，如某国产单片机品牌在研发过程中克服重重困难，最终推出性能优异的产品。

教学手段：讲授

主要内容：

2.1　AT89S51 单片机的硬件组成

片内硬件组成结构如图 2-1 所示。把作为控制应用所必需的基本功能部件都集成在一个尺寸有限的集成电路芯片上。



有如下外设部件和特性：

（1）8 位微处理器（CPU）；
（2）数据存储器（128B RAM）；
（3）程序存储器（4KB Flash ROM）；
（4）4 个 8 位可编程并行 I/O 口（P0 口、P1 口、P2 口和 P3 口）；
（5）2 个可编程的 16 位定时器/计数器（T0、T1），AT89S52 增加了 T2
（6）1 个全双工的异步串行口；
（7）中断系统具有 5 个中断源、5 个中断向量；
（8）特殊功能寄存器（SFR）26 个；
（9）1 个看门狗定时器；
（10）低功耗模式有空闲模式和掉电模式。

片内各外设部件：

（1）CPU（微处理器）
8 位，与通用 CPU 基本相同，同样包括了运算器和控制器两大部分，还有面向控制的位处理功能。
（2）数据存储器（RAM）
片内为为 128B，片外最多可扩 64KB。片内 128B 的 RAM 以高速 RAM 的形式集成，可加快单片机运行的速度和降低功耗。
（3）程序存储器（Flash ROM）
用来存储程序。AT89S51 片内有 4KB 的 Flash 存储器，AT89S52 片内有 8KB 的 Flash 存储器；S53/S54/S55 片内集成了 12KB/16KB/20KB 的 Flash 存储器，如果片内程序存储器容量不够，片外最多可外扩至 64KB 程序存储器，即“片内+片外”的程序存储器总容量不超过 64KB。

（4）中断系统
具有 5 个中断源，2 级中断优先权。

（5）定时器/计数器
片内有 2 个 16 位的定时器/计数器，具有 4 种工作方式。

（6）串行口

1 个全双工的异步串行口，4 种工作方式。可进行串行通信，扩展并行 I/O 口，可与多个单片机构成多机系统。

（7）P0 口、P1 口、P2 口和 P3 口　　4 个 8 位并行 I/O 口。

（8）特殊功能寄存器（SFR）

共有 26 个特殊功能寄存器，用于 CPU 对片内各外设部件进行管理、控制和监视。特殊功能寄存器实际上是片内各外设部件的控制寄存器和状态寄存器，这些特殊功能寄存器映射在片内 RAM 区的 80H～FFH 的地址区间内。

（9）1 个看门狗定时器 WDT

当单片机由于干扰而使程序陷入死循环或跑飞状态时，可引起单片机复位，使程序恢复正常运行。

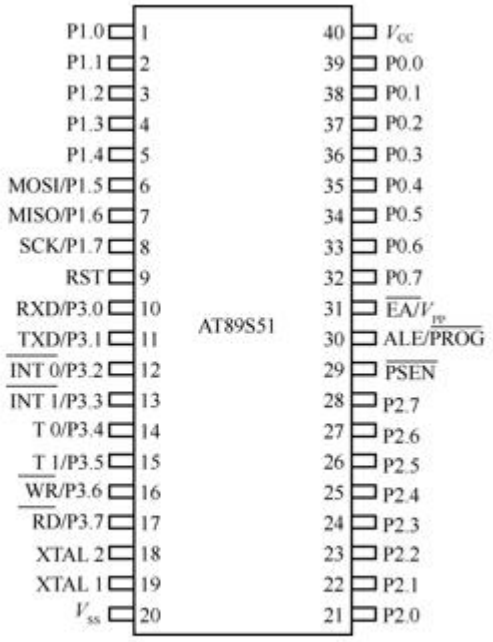
AT89S51 完全兼容 AT89C51 单片机，使用 AT89C51 单片机的系统，在保留原来软硬件的基础上，可用 AT89S51 直接代换。

2.2 AT89S51 的引脚功能

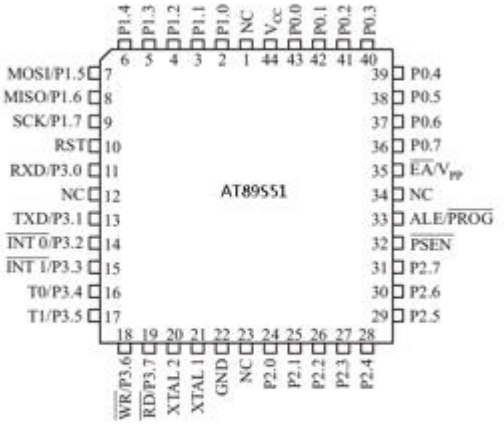
AT89S51 与各种 8051 单片机的引脚是兼容的。

目前，AT89S51 多采用 40 引脚的 DIP 封装（双列直插），见图 2-2（a），以及 44 引脚的 PLCC 和 TQFP 封装方式的芯片，见图 2-2（b）和图 2-2（c）。

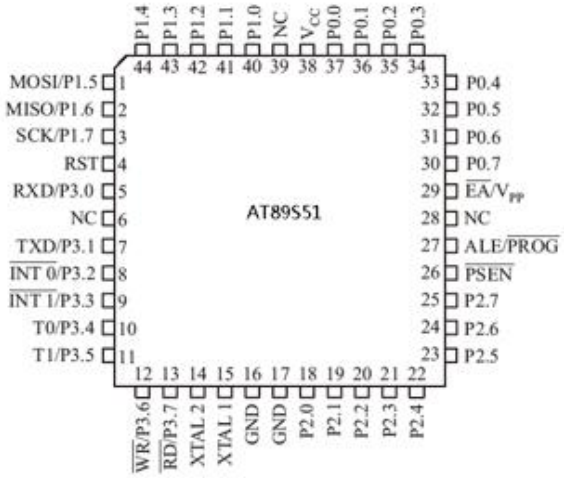
44 引脚的 PLCC 和 TQFP 封装方式的芯片，有 4 只引脚是无用的，标为“NC”。



(a)DIP封装的引脚分布



(b) PLCC封装的引脚分布



(c) TQFP封装的引脚分布

图2-2 AT89S51各种封装方式的引脚

2.2.3 并行 I/O 口引脚

1. P0 口：P0.7～P0.0 引脚

漏极开路的双向 I/O 口。当 AT89S51 扩展外部存储器及 I/O 接口芯片时，P0 口作为地址总线（低 8 位）及数据总线的分时复用端口。

P0 口也可作为通用 I/O 口使用，但需加上拉电阻，这时为准双向口。P0 口可驱动 8 个 LS 型 TTL 负载。

2. P1 口：P1.7～P1.0 引脚

准双向 I/O 口，具有内部上拉电阻，可驱动 4 个 LS 型 TTL 负载。

P1 口是完全可提供给用户使用的准双向 I/O 口。

P1.5/MOSI、P1.6/MISO 和 P1.7/SCK 也可用于对片内 Flash 存储器的串行编程和校验，它们分别是串行数据输入、串行数据输出和移位脉冲引脚。

3. P2 口：P2.7～P2.0 引脚

准双向 I/O 口，具有内部上拉电阻，可驱动 4 个 LS 型 TTL 负载。

当 AT89S51 扩展外部存储器及 I/O 口时，P2 口作为高 8 位地址总线用，输出高 8 位地址。

P2 口也可作为通用的 I/O 口使用。

4. P3 口：P3.7～P3.0

准双向 I/O 口，具有内部上拉电阻。

P3 口的第一功能是作为通用的 I/O 口使用，可驱动 4 个 LS 型 TTL 负载。

P3 口还可提供第二功能。第二功能定义见表 2-1，应熟记。

综上所述，P0 口可作为总线口，为双向口。作为通用的 I/O 口使用时，为准双向口，这时需加上拉电阻。P1 口、P2 口、P3 口均为准双向口。

表 2-1　P3 口的第二功能定义

引　脚	第 二 功 能	说　　明
P3.0	RXD	串行数据输入口
P3.1	TXD	串行数据输出口
P3.2	$\overline{INT0}$	外部中断 0 输入
P3.3	$\overline{INT1}$	外部中断 1 输入
P3.4	T0	定时器 0 外部计数输入
P3.5	T1	定时器 1 外部计数输入
P3.6	$\overline{WR}$	外部数据存储器写选通输出
P3.7	$\overline{RD}$	外部数据存储器读选通输出

2.3　AT89S51 的 CPU

CPU 由运算器和控制器构成。

2.3.1　运算器

对操作数进行算术、逻辑和位操作运算。主要包括算术逻辑运算单元 ALU、累加器 A、位处理器、程序状态字寄存器 PSW 及两个暂存器等。

1．算术逻辑运算单元 ALU

可对 8 位变量逻辑运算（与、或、异或、循环、求补和清零），还可算术运算（加、减、乘、除）

2．累加器 A

使用最频繁的寄存器，可写为 Acc。“A”与“Acc” 书写上的差别，将在第 3 章介绍。位于片内的特殊功能寄存器区。

3．程序状态字寄存器 PSW

PSW（Program Status Word）位于片内特殊功能寄存器区，字节地址为 D0H。

包含了程序运行状态的信息，其中 4 位保存当前指令执行后的状态，供程序查询和判断。格式如图 2-4 所示。

	D7	D6	D5	D4	D3	D2	D1	D0	
PSW	Cy	Ac	F0	RS1	RS0	OV	-	P	D0H

2.3.2　控制器

任务识别指令，并根据指令的性质控制单片机各功能部件，从而保证单片机各部分能自动协调地工作。

控制器包括：程序计数器、指令寄存器、指令译码器、定时及控制逻辑电路等。功能是控制指令的读入、译码和执行，从而对各功能部件进行定时和逻辑控制。

程序计数器 PC 是一个独立的 16 位计数器，不可访问。单片机复位时，PC 中内容为 0000H，从程序存储器 0000H 单元取指令，开始执行程序。

PC 工作过程：CPU 读指令时，PC 的内容作为所取指令的地址，程序存储器按此地址输出指令字节，同时 PC 自动加 1。

2.4 AT89S51 存储器的结构

存储器的结构特点之一是将程序存储器和数据存储器分开（哈佛结构），并有各自的访问指令。

存储器空间可分为 4 类。

1.程序存储器空间

片内和片外两部分。

片内 4KB Flash ，编程和擦除完全是电气实现。可用通用编程器对其编程，也可在线编程。

当片内 4KB Flash 存储器不够用时，可片外扩展，最多可扩展至 64KB 程序存储器。

2.数据存储器空间

片内与片外两部分。

片内有 128B RAM。

片内 RAM 不够用时，在片外可扩展至 64KB RAM 。

3.特殊功能寄存器 SFR （Special Function Register）

片内各功能部件的控制寄存器及状态寄存器。综合反映了整个单片机基本系统内部实际的工作状态及工作方式。

4.位地址空间

共有 211 个可寻址位，构成了位地址空间。位于片 RAM 区字节地址 20H～2FH（共 128 位）和特殊功能寄存器区（片内 RAM 区字节地址 80H～FFH 区间内，共定义了 83 个可寻址位）。

64KB 程序存储器空间中有 5 个特殊单元分别对应于 5 个中断源的中断入口地址，见表 2-3。

通常这 5 个中断入口地址处都放一条跳转指令跳向对应的中断服务子程序，而不是直接存放中断服务子程序。

表 2-3　AT89S51 各中断源的中断入口地址

中　断　源	入 口 地 址
外部中断 0	0003H
定时器/计数器 T0	000BH
外部中断 1	0013H
定时器/计数器 T1	001BH
串行口	0023H

作为对 AT89S51 存储器结构的总结，图 2-8 为各类存储器的结构图。从图中可清楚看出各类存储器在存储器空间的位置。

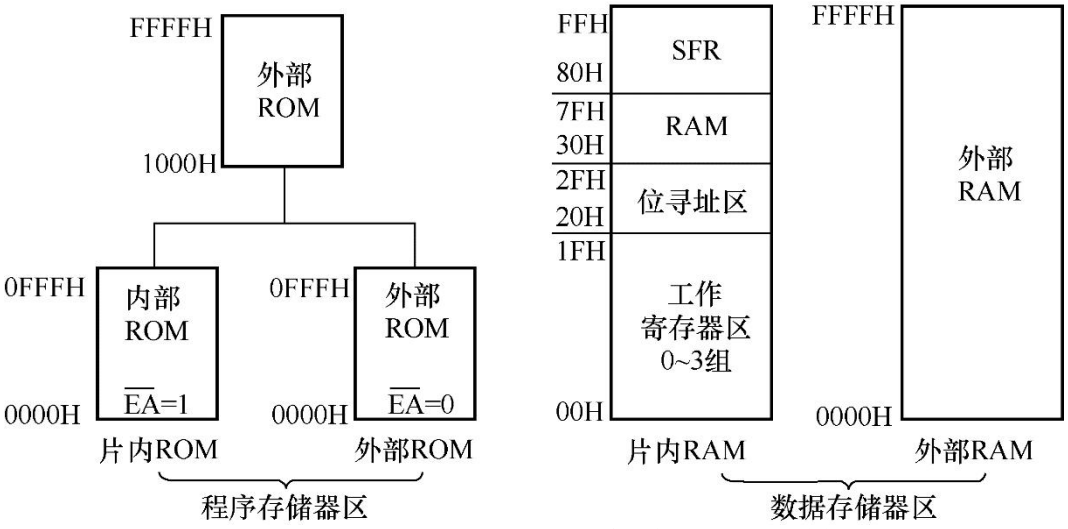


图 2-8　AT89S51 单片机的存储器结构

本节作业：

课本 P42 页

四、1~2

2.5 ~2.10 AT89S52 单片机的最小系统

教学目标：（教学 4 学时）

掌握单片机最小系统的概念。

教学重点

1．AT89S52 单片机的最小系统；

教学难点

1．AT89S52 单片机的最小系统；

素质（思政）内容与要求：

思政融入点：讲解 AT89S52 单片机最小系统的构成时，强调每一个元件的重要性，无论是电源电路还是复位电路，都需精心设计和调试。

案例：工程师在设计单片机最小系统时，经过多次试验优化电路布局，最终达到了既定的功能目标。通过这个例子，让学生明白精益求精的工作态度对于技术进步的重要性。

教学手段：讲授

主要内容：

2.5 AT89S51 的并行 I/O 端口

4 个双向的 8 位并行 I/O 端口，分别记为 P0、P1、P2 和 P3，其中输出锁存器属于特殊功能寄存器。端口的每一位均由输出锁存器、输出驱动器和输入缓冲器组成，4 个端口按字节输入/输出外，也可位寻址。

2.5.1 P0 口

P0 口是一个双功能的 8 位并行端口，字节地址为 80H，位地址为 80H～87H。端口的各位具有完全相同但又相互独立的电路结构，P0 口某一位的位电路结构如图 2-9 所示。

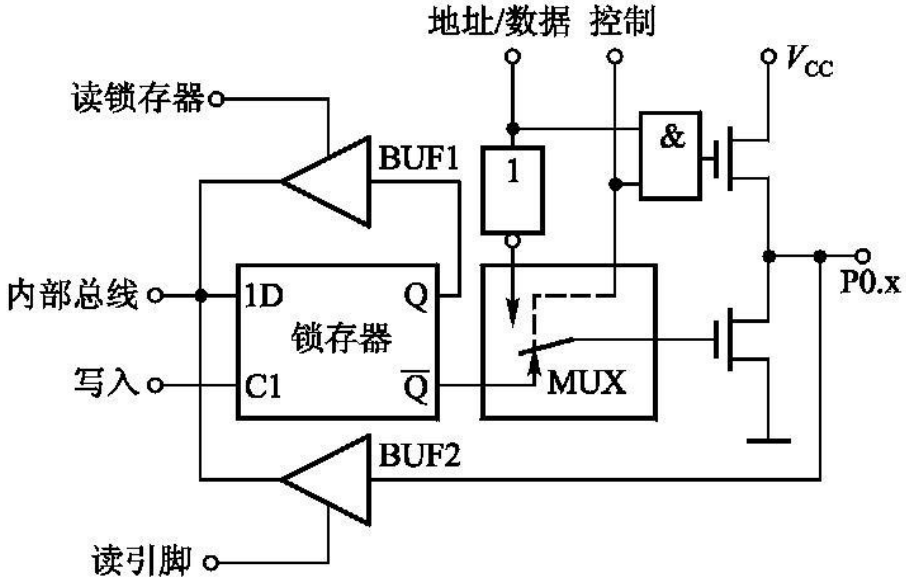


图 2-9 P0 口某一位的位电路结构

综上所述，P0 口具有如下特点：

（1）当 P0 口用作地址/数据总线口使用时，是一个真正的双向口，用作与外部扩展的存储器或 I/O 连接，输出低 8 位地址和输出/输入 8 位数据。

（2）当 P0 口用作通用 I/O 口使用时，各引脚需要在片外接上拉电阻，此时端口不存在高阻抗的悬浮状态，因此是一个准双向口。

如果单片机片外扩展了 RAM 和 I/O 接口芯片，P0 口此时应作为复用的地址/数据总线口使用。如果没有外扩 RAM 和 I/O 接口芯片，此时即可作为通用 I/O 口使用。

2．P1 口总结

P1 口由于有内部上拉电阻，没有高阻抗输入状态，故为准双向口。作为输出口时，不需要在片外接上拉电阻。

P1 口“读引脚”输入时，必须先向 P1 口的锁存器写入 1。

2．P2 口总结

P2 口作为高 8 位地址总线使用时，可输出外部存储器或 I/O 的高 8 位地址，与 P0 口输出并经锁存器的锁存的低 8 位地址一起构成 16 位地址，共可寻址 64KB 的片外地址空间。当 P2 口作为高 8 位地址输出口时，输出锁存器的内容保持不变。

P2 口作为通用 I/O 口使用时，为准双向口，功能与 P1 口一样。

一般情况下，P2 口大多作为高 8 位地址总线口使用，这时就不能再作为通用 I/O 口。如果不作为地址总线口使用，可作为通用 I/O 口使用。

2．P3 口总结

P3 口内部有上拉电阻，不存在高阻抗输入状态，故为准双向口。

由于 P**3 口每一引脚有第一功能与第二功能**，究竟是使用哪个功能，完全是由单片机执行的指令控制来自动切换的，用户不需要进行任何设置。

引脚输入部分有两个缓冲器，第二功能的输入信号取自缓冲器 BUF3 的输出端，第一功能的输入信号取自缓冲器 BUF2 的输出端。

2.6 时钟电路与时序

时钟电路产生 AT89S51 工作时所必需的控制信号，在时钟信号的控制下，严格按时序执行指令。

执行指令时，CPU 首先到程序存储器中取出需要执行的指令操作码，然后译码，并由时序电路产生一系列控制信号完成指令所规定的操作。

CPU 发的时序信号两类，一类用对片内各个功能部件控制，用户无须了解；另一类用于对片外存储器或 I/O 口的控制，这部分时序对于分析、设计硬件接口电路至关重要。

2.6.1 时钟电路设计

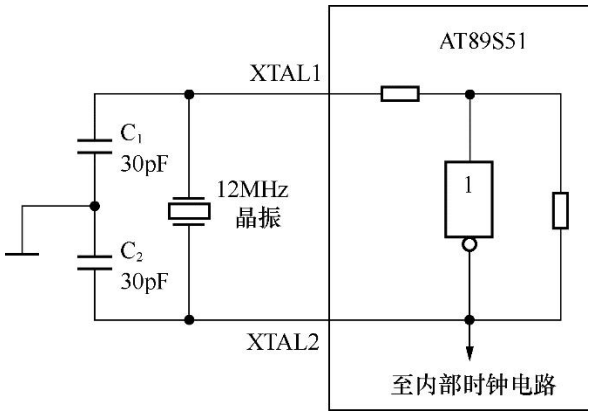


图 2-13　内部时钟方式电路

电路中的电容 C1 和 C2 的典型值通常选择为 30pF。晶体振荡频率通常选择 6MHz、12MHz（可得到准确的定时）或 11.0592MHz(可得到准确的串行通信波特率)的石英晶体。

2．外部时钟方式

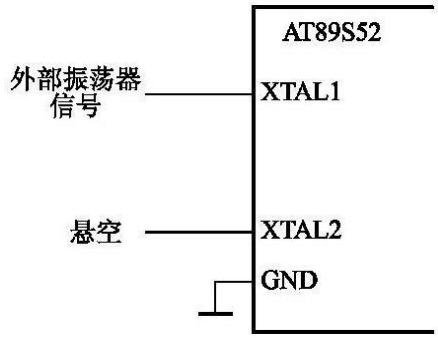


图 2-14　AT89S51 的外部时钟方式电路

3．时钟信号的输出

当使用片内振荡器，XTAL1、XTAL2 引脚还能为应用系统中的其他芯片提供时钟，但需增加驱动能力。引出的方式有两种，如图 2-15 所示。

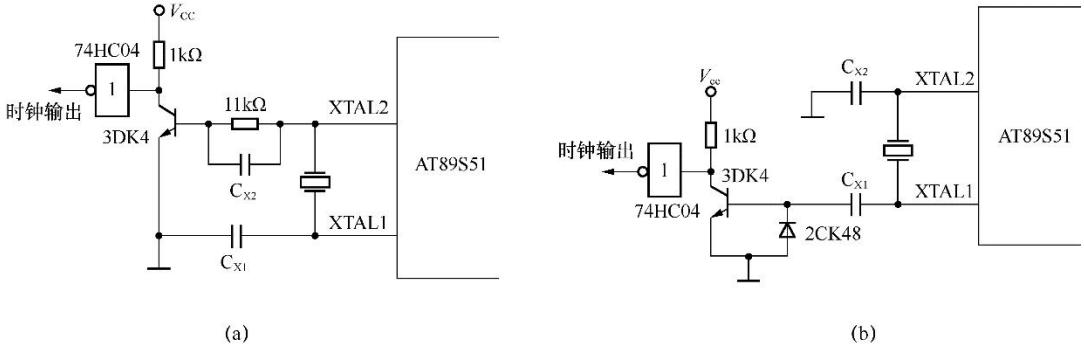


图 2-15　时钟信号的两种引出方式

2.7　复位操作和复位电路

单片机的初始化操作，给复位脚 RST 加上大于 2 个机器周期（即 24 个时钟振荡周期）的高电平就使 AT89S51 复位。

2.7.1　复位操作

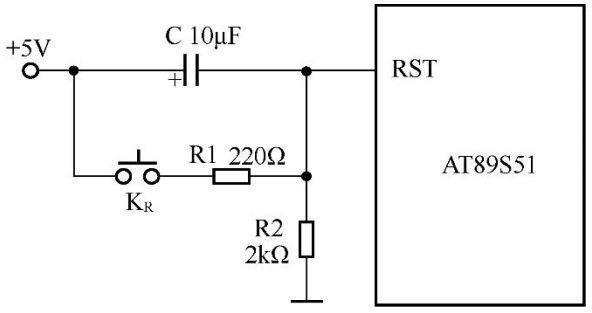
复位时，PC 初始化为 0000H，程序从 0000H 单元开始执行。

除系统的正常初始化外，当程序出错（如程序跑飞）或操作错误使系统处于死锁状态时，需按复位键使 RST 脚为高电平，使 AT89S51 摆脱“跑飞”或“死锁”状态而重新启动程序。

表 2-7　　复位时片内各寄存器的状态

寄 存 器	复位状态	寄 存 器	复位状态
PC	0000H	TMOD	00H
Acc	00H	TCON	00H
PSW	00H	TH0	00H
B	00H	TL0	00H
SP	07H	TH1	00H
DPTR	0000H	TL1	00H
P0～P3	FFH	SCON	00H
IP	×××0 0000B	SBUF	×××××××××B
IE	0××0 0000B	PCON	0××× 0000B
DP0H	00H	AUXR	××××0××0B
DP0L	00H	AUXR1	××××××××0B
DP1H	00H	WDTRST	××××××××B
DP1L	00H		

2.7.2　复位电路设计



2.8　AT89S51 单片机的最小应用系统

AT89S51 本身片内有 4KB 闪烁存储器，128B 的 RAM 单元，4 个 I/O 口，外接时钟电路和复位电路即构成了一个 AT89S51 单片机最小应用系统，如图 2-18 所示。

该最小应用系统只能作为小型的数字量的测控单元。

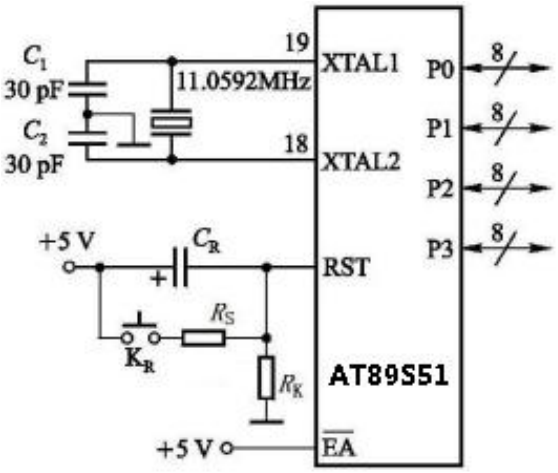


图 2-18　　AT89S51 单片机的最小应用系统

本节作业：

课本 P40-41 页

一、二、三

第三章　C51 语言编程基础（8 学时）

本章主要内容

本章介绍 8051 单片机的 C51 语言，以及如何使用 C51 语言集成化开发平台 Keil μVision3，进行 C51 程序设计与开发。

本章课时分配　本章分为 2 讲，共 8 学时。

3.1-3.2 C51 编程语言简介与程序设计基础

教学目标：（教学 4 学时）

了解用 C51 进行单片机编程的优点，与标准 C 语言比较的不同之处。

掌握 C51 的数据类型与存储类型，以及特殊寄存器。

教学重点

1. C51 的数据类型与存储类型；
2. C51 的特殊寄存器。

教学难点

1. C51 的数据类型与存储类型；
2. C51 的特殊寄存器。

素质（思政）内容与要求：

思政融入点：在教授 C51 语言的基本语法和编程技巧时，可以强调编写高质量、可维护代码的重要性，体现新时代工匠精神。

案例：引导学生通过不断地修改和完善自己的程序，完成一个高效且稳定的控制系统，培养工匠精神。

教学手段：讲授

主要内容：

3.1　C51 编程语言简介

用于 8051 单片机编程的 C 语言，在标准 C 基础上针对 8051 硬件特点进行扩展，并向 8051 上移植，经多年努力，C51 已成为公认的高效、简洁的 8051 单片机的实用高级编程语言。与 8051 汇编语言相比，C51 语言在功能上、结构性、可读性、可维护性上有明显优势，易学易用。

3.1.1　C51 语言与 8051 汇编语言比较

与 8051 汇编语言相比， C51 有如下优点。

（1）可读性好。C51 语言程序比汇编语言程序的可读性好，编程效率高，程序便于修改、维护以及升级。

（2）模块化开发与资源共享。用 C51 开发的程序模块可不经修改，直接被其他工程所用，使得开发者能够很好地利用已有的大量标准 C 程序资源与丰富的库函数，减少重复劳动，同时也有利于多个工程师进行协同开发。

（3）可移植性好。为某种型号单片机开发的 C 语言程序，只需把与硬件相关的头文件和编译链接的参数进行适当修改，就可方便地移植到其他型号的单片机上。例如，为 8051 单片机编写的程序通过改写头文件以及少量的程序行，就可方便地移植到 PIC 单片机上。

（4）生成的代码效率较高。当前较好的 C51 语言编译系统编译出来的代码效率只比直接使用汇编语言低 20%左右，如果使用优化编译选项，最高可达到 90%左右。

3.1.2 C51 语言与标准 C 语言的比较

C51 语言与标准 C 语言间有许多相同地方，但也有自身特点。不同的嵌入式 C 语言编译系统之所以与标准 C 语言有不同的地方，主要是由于它们所针对的硬件系统不同。对于 8051 单片机，目前广泛使用的是 C51 语言。

C51 语言基本语法与标准 C 相同，是在标准 C 的基础上进行适合 8051 内核单片机硬件的扩展。深入理解 C51 语言对标准 C 语言的扩展部分以及它们的不同之处，是掌握 C51 语言的关键之一。

C51 语言与标准 C 语言一些差别如下。

（1）库函数不同。
（2）数据类型有一定区别。
（3）C51 语言变量存储模式与标准 C 语言中变量存储模式数据不一样。
（4）数据存储类型不同。
（5）标准 C 语言没有处理单片机中断的定义
（6）头文件不同
（7）程序结构的差异

3.2 C51 语言程序设计基础

3.2.1 C51 语言中的数据类型与存储类型

1. 数据类型

数据是单片机操作的对象，具有一定格式的数字或数值，数据的不同格式就称为数据类型。

Keil C51 支持的基本数据类型见表 3-1。

针对 8051 的硬件特点，C51 在标准 C 基础上，扩展了 4 种数据类型（见表 3-1 中最后 4 行）。

注意，扩展的 4 种数据类型，不能使用指针来对它们存取。

表 3-1　Keil C51 支持的数据类型

数据类型	位数	字节数	值 域
signed char	8	1	−128～+127，有符号字符变量
unsigned char	8	1	0～255，无符号字符变量
signed int	16	2	−32 768～+32 767，有符号整型数
unsigned int	16	2	0～65 535，无符号整型数
signed long	32	4	−2 147 483 648～+2 147 483 647，有符号长整型数
unsigned long	32	4	0～+4 294 967 295，无符号长整型数
float	32	4	±1.175494E-38~±3.402823E+38
double	32	4	±1.175494E-38~±3.402823E+38
*	8～24	1～3	对象指针
bit	1		0 或 1
sfr	8	1	0～255
sfr16	16	2	0～65 535
sbit	1		可进行位寻址的特殊功能寄存器的某位的绝对地址

2．C51 的扩展数据类型

下面对扩展的 4 种数据类型说明。

（1）位变量 bit 的值可以是 1（true），也可是 0（false）。

（2）特殊功能寄存器 sfr。8051 单片机的特殊功能寄存器分布在片内数据存储区的地址单元 80H～FFH 之间，“sfr”数据类型占用一个内存单元。利用它可访问 8051 单片机内部的所有特殊功能寄存器。

例如：sfr P1=0x90 这一语句定义了 P1 端口在片内的寄存器，在程序后续的语句中可以用“P1=0xff”，使 P1 的所有引脚输出为高电平的语句来操作特殊功能寄存器。

（3）特殊功能寄存器 sfr16。

“sfr16”数据类型占用两个内存单元，用于操作占两个字节的特殊功能寄存器。例如：“sfr16 DPTR=0x82”语句定义了片内 16 位数据指针寄存器 DPTR，其低 8 位字节地址为 82H，高 8 位字节地址为 83H。在程序的后续语句中就可对 DPTR 进行操作。

（4）特殊功能位 sbit。

sbit 是指 AT89S51 片内特殊功能寄存器的可寻址位。例如：

```
sfr  PSW=0xd0;          //定义 PSW 寄存器地址为 0xd0
sbit OV=PSW^2;          //定义 OV 位为 PSW.2
```

上述第一条指令定义了特殊功能寄存器 PSW，第二条指令定义了特殊功能寄存器 PSW 中的可寻址位，符号“^”前是特殊功能寄存器名字，“^”后的数字定义可寻址位在特殊功能寄存器中的位置，取值必须是 0～7。

注意，不要把 bit 与 sbit 相混淆。bit 定义普通的位变量，只能是二进制的 0 或 1。sbit 是定义特殊功能寄存器的可寻址位，值是可以进行位寻址的特殊功能寄存器的某位的绝对地址，例如，PSW 寄存器 OV 位的绝对地址 0xd2。

上面的例子还涉及到 C51 注释的写法问题，C51 的注释写法有两种：

（1）//……，两个斜杠后面跟着的为注释语句，本写法只能注释一行，当换行时，必须在新行上重新写两个斜杠。

（2）/*……*/，一个斜杠与星号结合使用，本写法可注释任一行，即斜杠星号与星号斜杠之间的所有文字都作为注释，即注释有多行时，只需在注释的开始处，加斜杠星号，在注释的结尾处，加上星号斜杠即可。

头文件引用举例如下：

```
#include<reg51.h>        //包含 8051 单片机的头文件
void   main(void)
{
    TL0=0xf0;      //给 T0 低字节 TL0 设置时间常数，已在 reg51.h 中定义
    TH0=0x3f;          //给 T0 高字节 TH0 设置时间常数，已在 reg51.h 中定义
    TR0=1;         //启动定时器 0
    ……
}
```

3.2.3 C51 语言的绝对地址访问

1．绝对宏

编译器提供了一组宏定义对 code、data、pdata 和 xdata 空间进行绝对寻址。

程序中用“#include<absacc.h>”来对 absacc.h 中声明的宏来访问绝对地址，包括 CBYTE、CWORD、DBYTE、DWORD、XBYTE、XWORD、PBYTE、PWORD，具体使用参见 absacc.h 头文件。其中：

【例 3-2】片内 RAM、片外 RAM 及 I/O 定义的程序如下：

```
#include<absacc.h>
#define PORTA XBYTE[0xFFC0]
                    //将 PORTA 定义为外部 I/O 口，地址为 0xFFC0，长度 8 位
#define NRAM   DBYTE[0x50]
                    //将 NRAM 定义为片内 RAM，地址为 0x50，长度 8 位
main(   )
{
    PORTA=0x3d;      //将数据 3DH 写入地址为 0xffc0 的外部 I/O 端口 PORTA
中
    NRAM=0x01;           //将数据 01H 写入片内 RAM 的 0x40 单元
}
```

表 3-7　关系运算符及其说明

符　号	说　明	举例（设 a=2,b=3）
>	大于	a> b；//返回值为 0
<	小于	a< b；//返回值为 1
>=	大于等于	a>=b；//返回值为 0
<=	小于等于	a<=b；//返回值为 1
==	等于	a==b；//返回值为 0
!=	不等于	a!=b；//返回值为 1

4．位运算

位运算符及其说明见表 **3-8**。

表 3-8　位运算及其说明

符　　号	说　　明	举例
&	按位逻辑与	0x19&0x4d=0x09
|	按位逻辑或	0x19 | 0x4d =0x5d
^	按位异或	0x19^0x4d =0x54
～	按位取反	x=0x0f，则～x=0xf0
<<	按位左移(高位丢弃，低位补 0)	y=0x3a，若 y<<2，则 y=0xe8
>>	按位右移(高位补 0，低位丢弃)	w=0x0f，若 w>>2，则 w=0x03

C51 提供 **3** 种形式的 **if** 语句：

形式 **1**

if (表达式**)　{**语句**}**

例如：

if (x>y)　{max=x; min=y;}

即如果 **x>y**，则 **x** 赋给 **max**，**y** 赋给 **min**。如果 **x>y** 不成立，则不执行大括号中的赋值运算。

形式 **2**

if (表达式**)　{**语句 **1;}　else {**语句 **2;}**

例如：

```
if (x>y)
{max=x; }
else {max=y;}
```

本形式相当于双分支选择结构。

形式 **3**

```
if (表达式 1) {语句 1;}
else  if (表达式 2) {语句 2;}
else  if (表达式 3) {语句 3;}
……
else  {语句 n;}
```

例如：

```
if (x>100) {y=1;}
else  if (x>50) {y=2;}
else  if (x>30) {y=3;}
else  if (x>20) {y=4;}
else  {y=5;}
```

本形式相当于串行多分支选择结构。

在 if 语句中又含有一个或多个 if 语句，这称为 if 语句的嵌套。应当注意 if 与 else 的对应关系，else 总是与它前面最近的一个 if 语句相对应。

（2）switch 语句。if 语句只有两个分支可选择，而 switch 语句是多分支选择语句。switch 语句的一般形式如下：

```
switch  (表达式 1)
{
    case  常量表达式 1:{语句 1;}break;
    case  常量表达式 2:{语句 2;}break;
    ……
    case  常量表达式 n:{语句 n;}break;
    default:{语句 n+1;}
}
```

【例 3-6】在单片机程序设计中，常用 switch 语句作为键盘中按键按下的判别，并根据按下键的键号跳向各自的分支处理程序。

```
input:  keynum=keyscan( )
switch(keynum)
{
    case 1:key1( ); break;    //如果按下键为 1 键，则执行函数 key1( )
    case 2:key2( ); break;    //如果按下键为 2 键，则执行函数 key2( )
    case 3:key3( ); break;    //如果按下键为 3 键，则执行函数 key3( )
    case 4:key4( ); break;    //如果按下键为 4 键，则执行函数 key4( )
    ……
    default:goto input
}
```

例子中的 keyscan( )是另行编写的一个键盘扫描函数，如有键按下，该函数就会得到按下键的键值，将键值赋予变量 keynum。如果键值为 2，则执行键值处理函数 key2( )后返回；如果键值为 4，则执行 key4( )函数后返回。

2．循环控制语句

许多实用程序都包含循环结构，熟练掌握和运用循环结构的程序设计是 C51 语言程序设计的基本要求。

实现循环结构的语句有以下 3 种：while 语句、do-while 语句和 for 语句。

（1）while 语句。语法形式为：

```
while(表达式)
    {
        循环体语句;
```

```
}
```

表达式是 while 循环能否继续的条件，如果表达式为真，就重复执行循环体语句；反之，则终止循环体内的语句。

while 循环结构特点：循环条件测试在循环体开头，要想执行重复操作，首先必须进行循环条件的测试，如条件不成立，则循环体内的重复操作一次也不能执行。

例如：

```
while((P1&0x80)= =0)
{ }
```

while 中的条件语句对 AT89S8051 单片机的 P1 口的 P1.7 位进行测试，如果 P1.7 为低（0），则由于循环体无实际操作语句，故继续测试下去（等待），一旦 P1.7 的电平变高（1），则循环终止。

（2）do-while 语句。语法形式为：

```
do
{
   循环体语句;
}
while(表达式);
```

【例 3-7】实型数组 sample 存有 10 个采样值，编写程序段，要求返回其平均值（平均值滤波）。程序如下：

```
float avg(float *sample)
     {
        float sum=0;
    char n=0;
    do
       {
         sum+=sample[n];
             n++;
         } while(n<10);
         return(sum/10);
      }
```

（3）基于 for 语句的循环。3 种循环常用的是 for 循环。不仅可用于循环次数已知的情况，也可用于循环次数不确定而只给出循环条件情况，完全可替代 while 语句。

for 循环的一般格式为

```
for(表达式 1;表达式 2;表达式 3)
{
    循环体语句;
}
```

for 是关键字，括号中常含有 3 个表达式，各表达式间用“；”隔开。这 3 个表达式可以是任意形式的表达式，通常主要用于 for 循环控制。紧跟在 for()之后的循环体，在语法上要求是 1 条语句；若在循环体内需要多条语句，应用大括号括起来组成复合语句。

⑤ 没有循环体的 for 语句。

例如：

```
int a=1000;
for(t=0;t<a;t++)
{;}
```

本例典型应用就是软件延时。可用循环结构来实现，即循环执行指令，消磨一段已知的时间。指令的执行时间是靠一定数量的时钟周期来计时的，如果使用 12MHz 晶振，则 12 个时钟周期花费的时间为 1μs。

【例 3-8】编写一个延时 1ms 程序。

```
void delayms( unsigned char int j)
{
    unsigned char i;
    while(j- -)
    {
        for(i=0;i<125;i++)
        {;}
    }
}
```

3．break 语句、continue 语句和 goto 语句

在循环体执行中，如满足循环判定条件的情况下跳出代码段，可使用 break 语句或 continue 语句；如要从任意地方跳转到代码某地方，可使用 goto 语句。

（1）break 语句

循环结构中，可使用 break 语句跳出本层循环体，马上结束本层循环。

【例 3-11】执行如下程序段。

```
void   main(void )
{   int i,sum;
    sum=0;
    for(i=1;i<=10;i++)
    {   sum=sum+i;
        if(sum>5) break;
        print( “sum=%d\n” , sum); //通过串口向计算机屏幕输出显示 sum 值    }
}
```

本例如没有 break 语句，程序将进行 10 次循环；当 i=3 时，sum 的值为 6，此时，if 语句的表达式“sum>5”的值为 1，于是执行 break 语句，跳出 for 循环，从而提前终止循环。

因此在一个循环程序中，既可通过循环语句中的表达式来控制循环是否结束，还可直接通过 break 语句强行退出循环结构。

（2）continue 语句

作用及用法与 break 语句类似，区别：当前循环遇到 break，是直接结束循环，若遇上 continue，则是停止当前这一层循环，然后直接尝试下一层循环。可见，continue 并不结束整个循环，而仅仅是中断这一层循环，然后跳到循环条件处，继续下一层的循环。

当然，如果跳到循环条件处，发现条件已不成立，那么循环也会结束。

本节作业

课本 P68 页

一、1、2、3

三、1-2

四、1

3.3 C51 语言的函数

教学目标：（教学 4 学时）

了解 C51 的函数类型，区别主函数与子函数、库函数；

教学重点

1．C51 的函数类型；

教学难点

C51 的函数类型；

素质（思政）内容与要求：

思政融入点：鼓励学生在学习函数的过程中，尝试创新，编写具有特色的函数，解决实际问题。

案例：引导学生编写通用的函数，可实现反复调用而不必重复代码，这不仅简化了程序结构，也方便了后期的维护和升级。

教学手段：讲授

主要内容：

3.3 C51 语言的函数

函数是一个完成一定相关功能的执行代码段。在高级语言中，函数与另外两个名词“子程序”和“过程”用来描述同样的事情。在 C51 语言中使用的是函数这个术语。

C51 语言中函数的数目是不限制的，但是一个 C51 程序必须至少有一个函数，以 main 为名，称为主函数，主函数是唯一的，整个程序从这个主函数开始执行。

C51 语言还可建立和使用库函数，可由用户根据需求调用。

3.3.1 函数的分类

从结构上分，C51 语言函数可分为主函数 main( ) 和普通函数两种。而普通函数又划分为两种：标准库函数和用户自定义函数。

1．标准库函数

标准库函数是由 C51 编译器提供的。编程者在进行程序设计时，应该善于充分利用这些功

能强大、资源丰富的标准库函数资源，以提高编程效率。

用户可直接调用 C51 库函数而不需为这个函数写任何代码，只需要包含具有该函数说明的头文件即可。例如调用输出函数 printf 时，要求程序在调用输出库函数前包含以下的 include 命令：

#include <stdio.h>

2. 用户自定义函数

用户自定义函数是用户根据需要所编写的函数。从函数定义的形式分为：无参函数、有参函数和空函数。

（1）无参函数

此种函数在被调用时，既无参数输入，也不返回结果给调用函数，只是为完成某种操作而编写的函数。

无参函数的定义形式为：

返回值类型标识符　函数名（）

{

　函数体；

}

无参函数一般不带返回值，因此函数的返回值类型的标识符可省略。

例如函数：main（），为无参函数，返回值类型的标识符可省略，默认值是 int 类型。

（2）有参函数

调用此种函数时，必须提供实际的输入函数。有参函数的定义形式为：

返回值类型标识符　函数名（形式参数列表）

形式参数说明

{

　函数体；

}

【例 3-15】定义一个函数 max( )，用于求两个数中的大数。

```
int a,b
int max(a, b)
{
        if（a>b）return（a）;
        else   return（b）;
}
```

程序段中，a、b 为形式参数。return（ ）为返回语句。

（3）空函数

此种函数体内是空白的。调用空函数时，什么工作也不做，不起任何作用。定义空函数的目的，并不是为了执行某种操作，而是为了以后程序功能的扩充。先将一些基本模块的功能函数定义成空函数，占好位置，并写好注释，以后再用一个编好的函数代替它。这样整个程序的结构清晰，可读性好，以后扩充新功能方便。

空函数的定义形式为：
返回值类型标识符　函数名（）
{　　}
例如：
float min(　　)
{　　}　　/*空函数，占好位置*/

3.3.2　函数的参数与返回值

1.函数的参数

C 语言采用函数之间的参数传递方式，使一个函数能对不同的变量进行功能相同的处理，从而大大提高了函数的通用性与灵活性。

函数之间的参数传递，由主函数调用时主调函数的实际参数与被调函数的形式参数之间进行数据传递来实现。

被调用函数的最后结果由被调用函数的 return 语句返回给调用函数。

函数的参数包括形式参数和实际参数。

（1）形式参数：函数名后面括号中的变量名称为形式参数，简称形参。
（2）实际参数：在函数调用时，主调函数名后面括号中的表达式称实际参数，简称实参。

在 C 语言的函数调用中，实际参数与形式参数之间的数据传递是单向进行的，只能由实际参数传递给形式参数，而不能由形式参数传递给实际参数。

实际参数与形式参数的类型必须一致，否则会发生类型不匹配的错误。被调用函数的形式参数在函数未调用之前，并不占用实际内存单元。只有当函数调用发生时，被调用函数的形式参数才分配给内存单元，此时内存中调用函数的实际参数和被调用函数的形式参数位于不同的单元。

在调用结束后，形式参数所占有的内存被系统释放，而实际参数所占有的内存单元仍保留并维持原值。

2. 函数的返回值

函数返回值是通过 return 语句获得的。一个函数可有一个以上的 return 语句，但是多于一个的 return 语句必须在选择结构（if 或 do/case）中使用（例如前面求两个数中的大数函数 max( )的例子），因为被调用函数一定只能返回一个变量。

函数返回值的类型由返回值的标识符来指定。例如在函数名之前的 int 指定函数的返回值的类型为整型数（int）。若没有指定函数的返回值类型，默认返回值为整型类型。

当函数没有返回值时，则使用标识符 void 进行说明。

3.3.3　函数的调用

在一个函数中需要用到某个函数的功能时，就调用该函数。调用者称为主调函数，被调用者称为被调函数。

1. 函数调用的一般形式

函数调用的一般形式：

函数名　{实际参数列表}；

若被调函数是有参函数，则主调函数必须把被调函数所需的参数传递给被调函数。传递给被

调函数的数据称为实际参数（简称实参），必须与形参的数据在数量、类型和顺序上都一致。实参可以是常量、变量和表达式。实参对形参的数据是单向的，即只能将实参传递给形参.

2. 函数调用的方式

主调用函数对被调用函数的调用有以下 3 种方式。

（1）函数调用语句

函数调用语句把被调用函数的函数名作为主调函数的一个语句。例如：

```
print_message(  );
```

此时，并不要求函数返回结果数值，只要求函数完成某种操作。

（2）函数结果作为表达式的一个运算对象

函数结果作为表达式的一个运算对象，例如：

```
result=2*gcd(a,b);
```

被调用函数以一个运算对象出现在表达式中。这要求被调用函数带有 return 语句，以便返回一个明确的数值参加表达式的运算。被调用函数 gcd 为表达式的一部分，它的返回值乘 2 再赋给变量 result。

（3）函数参数

函数参数即被调用函数作为另一个函数的实际参数。例如：

```
m=max(a,gcd(u,v));
```

其中，gcd(u,v)是一次函数调用，它的值作为另一个函数的 max( )的实际参数之一。

3. 对调用函数的说明

在一个函数调另一个函数调用另一个函数时，须具备以下条件：

（1）被调用函数必须是已经存在的函数（库函数或用户自定义的函数）。

（2）如果程序中使用了库函数，或使用了不在同一文件中的另外自定义函数，则应该在程序的开头处使用#include 包含语句，将所有的函数信息包含到程序中来。

例如，#include<stdio.h>，将标准的输入、输出头文件 stdio.h（在函数库中）包含到程序中来。

在程序编译时，系统会自动将函数库中的有关函数调入到程序中去，编译出完整的程序代码。

3.3.4 中断服务函数

由于标准 C 没有处理单片机中断的定义，为能进行 8051 的中断处理，C51 编译器对函数定义进行了扩展，增加了一个扩展关键字 interrupt。

使用 interrupt 可将一个函数定义成中断服务函数。由于 C51 编译器在编译时对声明为中断服务程序的函数自动添加了相应的现场保护、阻断其他中断、返回时自动恢复现场等处理的程序段，因而在编写中断服务函数时可不必考虑这些问题，这就为用户编写中断服务程序提供了极大方便。

中断服务函数的一般形式为：

```
函数类型  函数名（形式参数表） interrupt n  using n
```

关键字 interrupt 是中断号，对于 51 单片机，n 取值为 0~4。

关键字 using 后的 n 是所选择的寄存器组，using 是一个选项，可省略。

如果不用关键字 using 指明寄存器组，中断函数中的所有工作寄存器的内容将被保存到堆栈中。

3.3.5 变量及存储方式

1. 变量

（1）局部变量

是某一个函数中存在的变量，它只在该函数内部有效。

（2） 全局变量

在整个源文件中都存在的变量。有效区间是从定义点开始到源文件结束，其中的所有函数都可直接访问该变量。如果定义前的函数需要访问该变量，则需要使用 extern 关键词对该变量进行说明，如果全局变量声明文件之外的源文件需要访问该变量，也需要使用 extern 关键词进行说明。由于全局变量一直存在，占用了大量的内存单元，且加大了程序的耦合性，不利于程序的移植或复用。

全局变量可以使用 static 关键词进行定义，该变量只能在变量定义的源文件内使用，不能被其他源文件引用，这种全局变量称为静态全局变量。如果一个其他文件的非静态全局变量需要被某文件引用，则需要在该文件调用前使用 extern 关键词对该变量声明。

2. 变量的存储方式

单片机的存储区间，可以分为程序存储区、静态存储区和动态存储区 3 个部分。数据存放在静态存储区或动态存储区。其中全局变量存放在静态存储区，在程序开始运行时，给全局变量分配存储空间；局部变量存放在动态存储区，在进入拥有该变量的函数时，给这些变量分配存储间。

3.3.6 宏定义与文件包含

在 C51 程序设计中要经常用到宏定义、文件包含与条件编译。

1. 宏定义

宏定义语句属于 C51 语言的预处理指令，使用宏可以使变量书写简化，增加程序的可读性、可维护性和可移植性。宏定义分为简单的宏定义和带参数的宏定义。

简单的宏定义格式如下：

#define 宏替换名 宏替换体

#define 是宏定义指令的关键词，宏替换名一般用大写字母来表示，而宏替换体可以是数值常数、算术表达式、字符和字符串等。宏定义可以出现在程序的任何地方，例如宏定义：

#define uchar unsigned char

在编译时可由 C51 编译器把“unsigned char”用“uchar”来替代

例如，在某程序的开头处，进行了 3 个宏定义：

```
#define uchar unsigned char          /*宏定义无符号字符型变量方便书写*/
```

2. 文件包含

是指一个程序文件将另一个指定的文件的内容包含进去。文件包含的一般格式为：

#include <文件名> 或 #include “文件名”

3.3.7 库函数

C51 语言的强大功能及其高效率在于提供了丰富的可直接调用的库函数。库函数可以使程序代码简单、结构清晰、易于调试和维护。

下面介绍几类重要的库函数。

(1) 特殊功能寄存器包含文件 reg51.h 或 reg52.h。reg51.h 中包含所有的 8051 的 sfr 及其位定义。reg52.h 中包含所有 8052 的 sfr 及其位定义，一般系统都包含 reg51.h 或 reg52.h。

(2) 绝对地址包含文件 absacc.h：该文件定义了几个宏，以确定各类存储空间的绝对地址。

(3) 输入/输出流函数位于 stadio.h 文件中。流函数默认 8051 的串口来作为数据的输入/输出。如果要修改为用户定义的 I/O 口读写数据，例如，改为 LCD 显示，可以修改 lib 目录中的 getkey.c 及 putchar.c 源文件，然后在库中替换它们既可。

(4) 动态内存分配函数，位于 stdlib.h 中。

(5) 能够对方便地对缓冲区进行处理的缓冲区处理函数位于 string.h 中。其中包括复制、移动、比较等函数。

本节作业

课本 P68 页

二

四、2、3、4

第四章　开发与仿真工具（4 学时）

本章主要内容

本章为实训实操，介绍编程软件 Keil 与仿真软件 proteus。

本章课时分配　本章分为 2 讲，共 4 学时。

4.1 Keil C51 的使用

教学目标：（教学 2 学时）

1、了解 Keil 软件的基本概况；

2、清楚 Keil 软件的窗口功能；

3、掌握 keil 软件新建程序、编程、调试、生成 HEX 文件的流程。

教学重点

1．Keil 软件新建程序、编程、调试；

教学难点

1．Keil 软件新建程序、编程、调试；

素质（思政）内容与要求：

思政融入点：鼓励学生在使用仿真工具时，不仅仅是模拟现有的设计，而是尝试提出自己的创新思路。

案例：引导同学们共同完成了一个基于单片机的自动化控制系统项目。在这个过程中，团队成员利用各自的工具进行设计、仿真和测试。

教学手段：实操演练

主要内容：

4.1　Keil C51 的使用

4.1.1　Keil C51 简介

Keil C51 是用于 8051 单片机的 C51 语言编程的集成开发环境，它由德国 Keil software 公司（已被 ARM 公司收购）开发，是使用 C51 语言开发编程所必须掌握的软件开发工具。

Keil C51 集编辑、编译、仿真等功能于一体，它具有强大的软件调试功能，生成的程序代码运行速度快，所需的存储器空间小，完全可与汇编语言相媲美，是目前 8051 单片机应用开发中的最优秀软件开发工具之一。Keil C51 集成了文件编辑处理、编译链接、项目（Project）管理窗口、工具引用、仿真软件模拟器以及 Monitor51 硬件目标调试器等多种功能，可在 Keil C51 开发环境中极为简便地进行操作。

4.1.2　基本操作

1．软件安装与启动

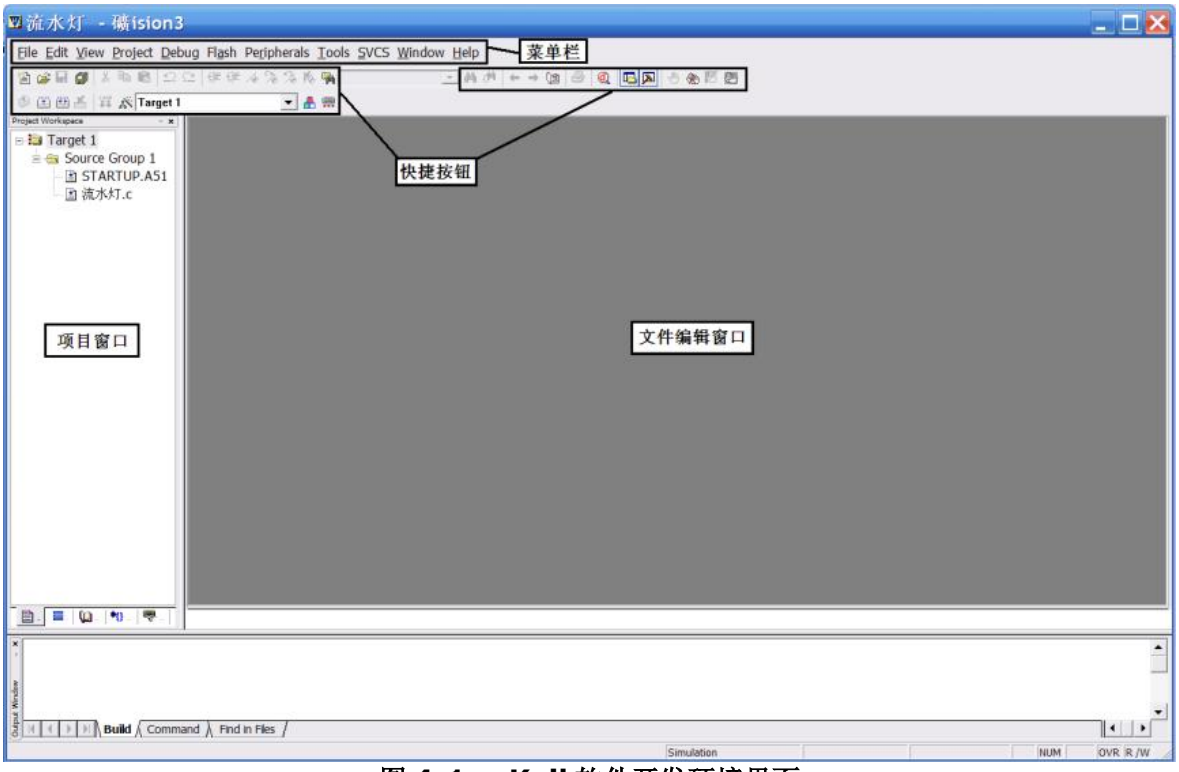


图 4-1　Keil 软件开发环境界面

2. 创建项目

一、新建文件夹

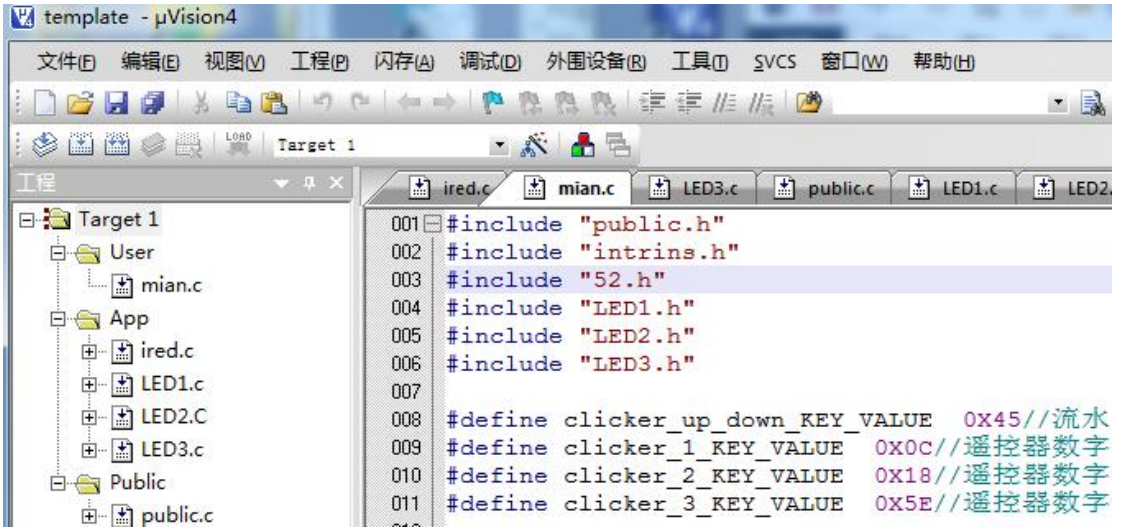
在想保存的位置（如桌面），新建一个文件夹 ，命名如：项目 1

二、打开软件

双击 

打开 Keil4 软件，出现如下空空的界面：

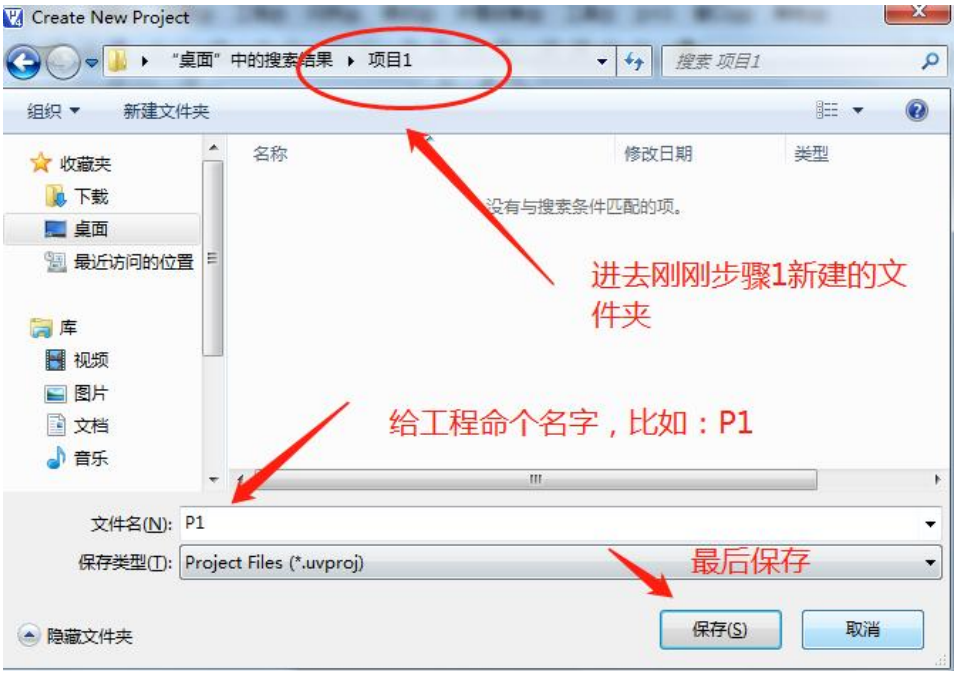
如果出现已经存在的工程，可以把工程关闭：

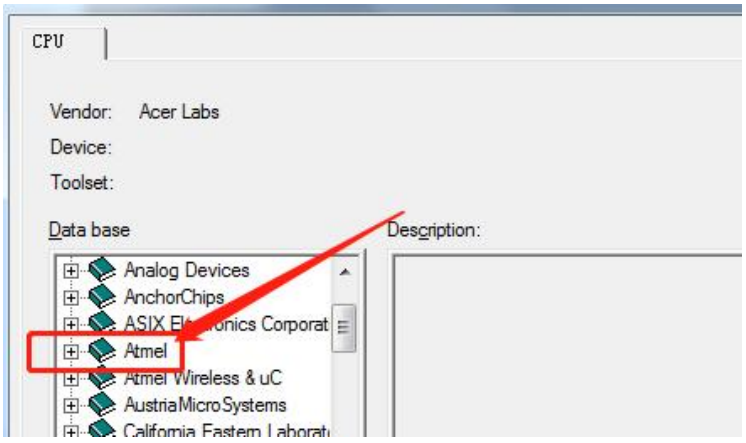


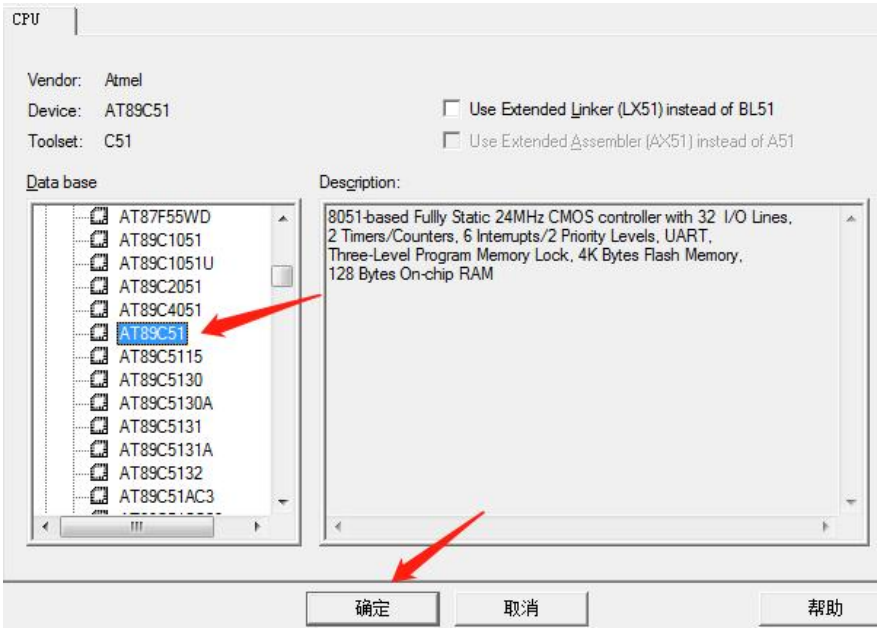
点击工程，关闭工程

三、新建工程

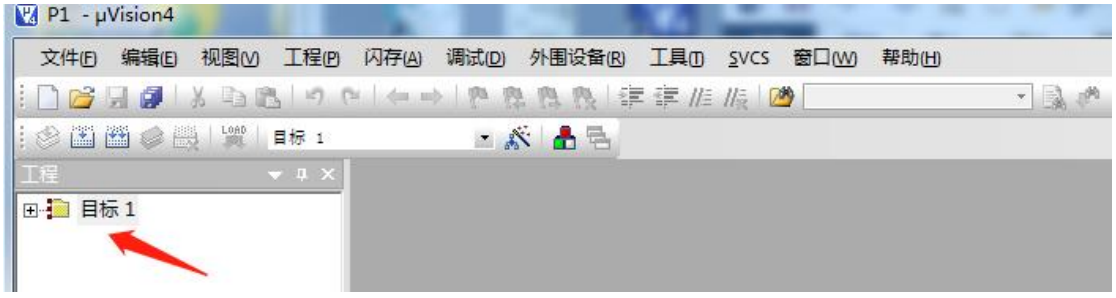
点击“工程”—>“New uVision Project”，新建工程


Create New Project
"桌面" 中的搜索结果 ▸ 项目1
搜索 项目1
组织
新建文件夹
收藏夹
下载
桌面
最近访问的位置
库
视频
图片
文档
音乐
名称
修改日期
类型
没有与搜索条件匹配的项。
进去刚刚步骤1新建的文件夹
给工程命个名字，比如：P1
文件名(N): P1
保存类型(T): Project Files (*.uvproj)
最后保存
隐藏文件夹
保存(S)
取消


CPU
Vendor: Acer Labs
Device:
Toolset:
Data base
Description:
Analog Devices
AnchorChips
Atmel
Atmel Wireless & uC
AustriaMicroSystems



可以看到左边多了个工程，说明工程新建成功：

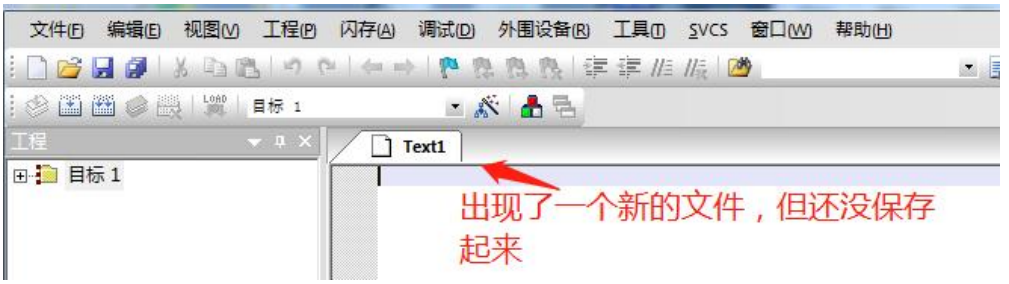


四、新建 C 语言文件

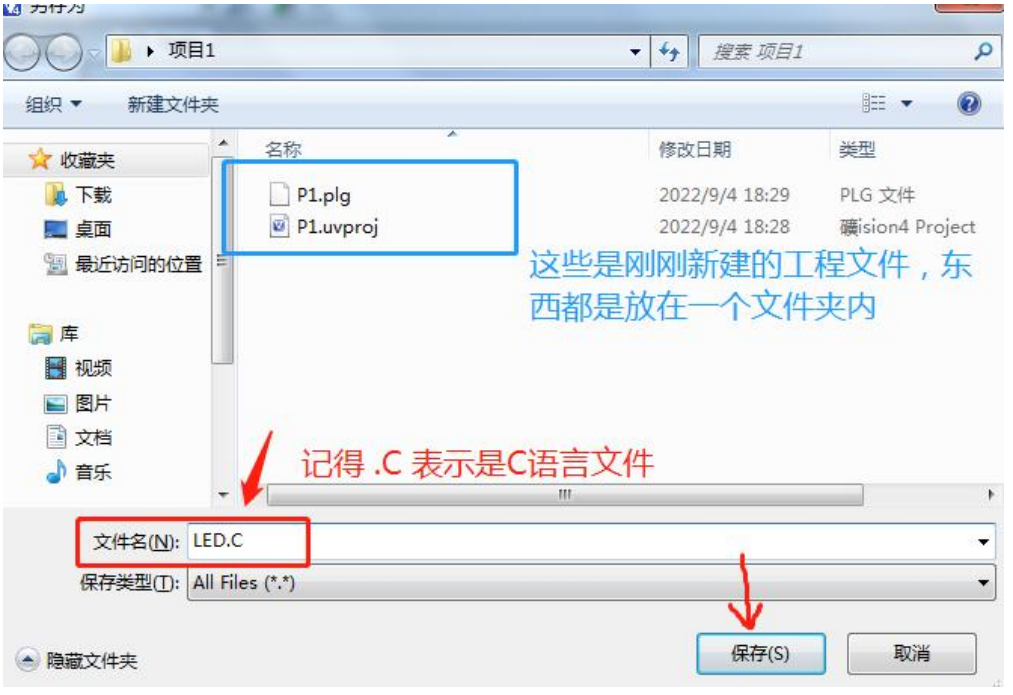
先新建一个文件

方法 1

方法 2

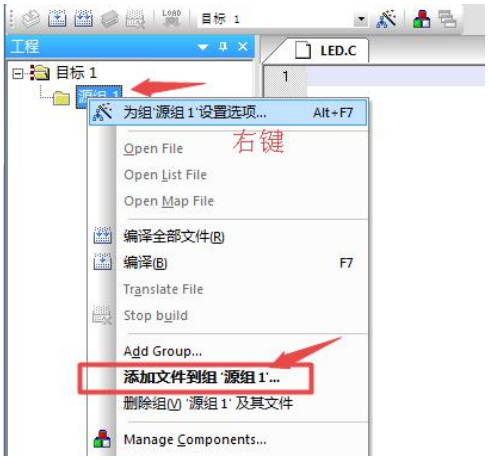


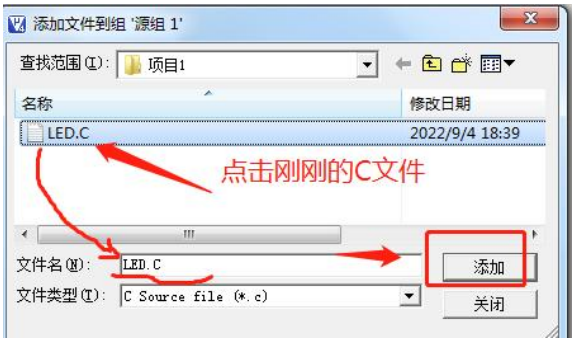
点击保存文件，**命名记得加后缀.C**



把**保存好的 C 文件**添加到工程文件里

右键（源组 1），点击添加





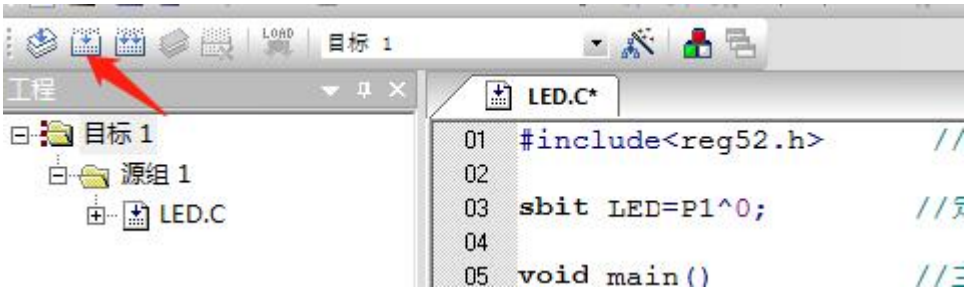
五、输入 C 语言程序，并调试。自己试下手打，或者直接复制过去

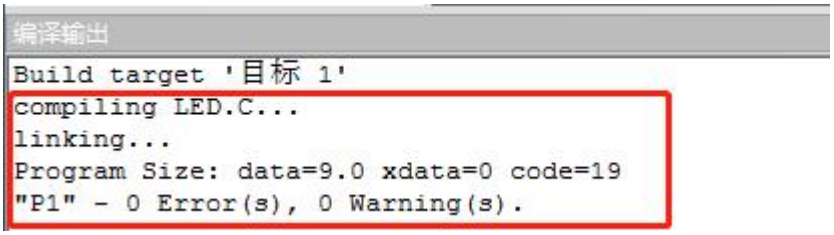
```
#include<reg52.h>

sbit LED=P2^0;

void main()
{
    while(1)
    {
        LED=0;
    }
}
```

LED.C

```
01  #include<reg52.h>       //C52单片机的头文件
02
03  sbit LED=P2^0;          //定义P2.0的引脚名为LED，用来控制LED亮不亮
04
05  void main()             //主程序
06  {
07      while(1)            //一直循环
08      {
09          LED=0;          //点亮灯
10      }
11  }
```

输入程序后，点击调试：

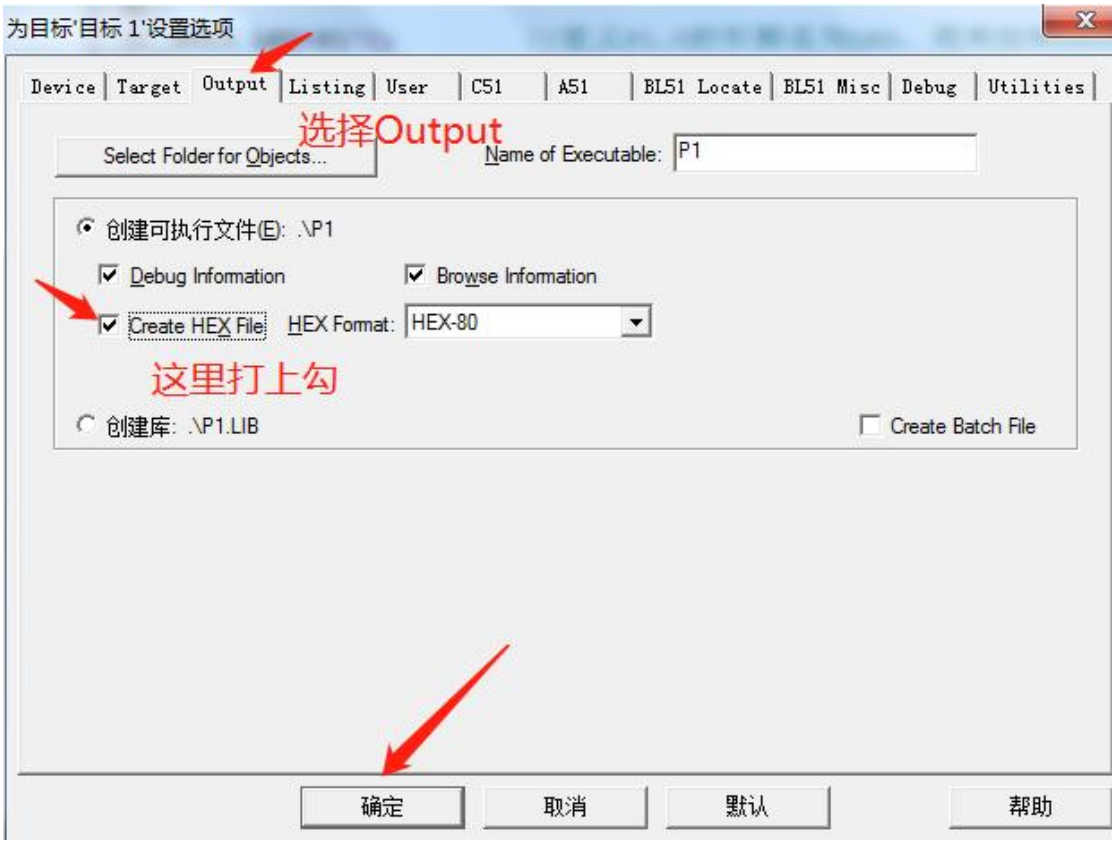




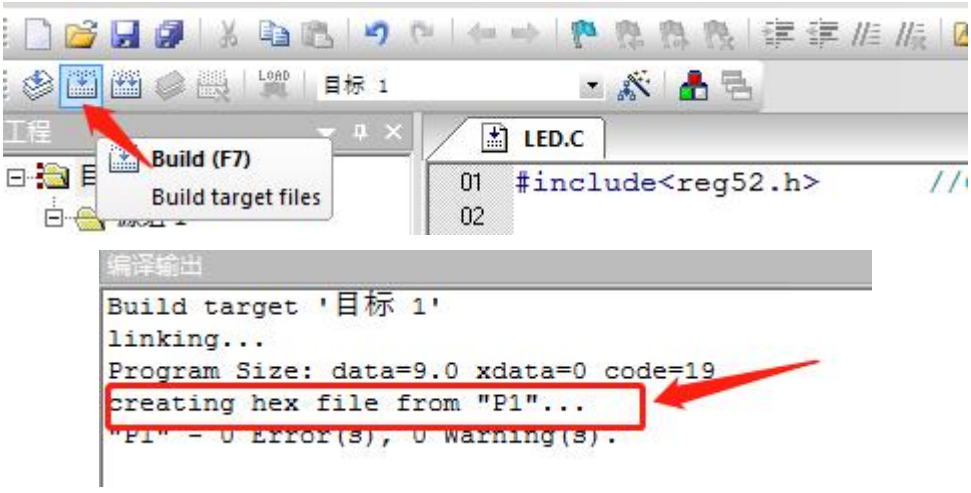
出现以上 0 错误，表示程序 OK。

六、生成 HEX 文件

点击这个很像魔法棒的东西，



再次运行一次，可以看到生成了 HEX 文件。



本节作业

根据以上步骤，完成 Keil 软件新建程序的任务。

4.2 Proteus 虚拟仿真工具介绍

教学目标：（教学 2 学时）

1、了解 Proteus 软件的基本概况；

2、清楚 Proteus 软件的窗口功能；

3、掌握 Proteus 软件搭建仿真电路的方法，并进行仿真。

教学重点

1．Proteus 软件搭建仿真电路的方法，并进行仿真；

教学难点

1．Proteus 软件搭建仿真电路的方法，并进行仿真；

素质（思政）内容与要求：

思政融入点：鼓励学生在使用仿真工具时，不仅仅是模拟现有的设计，而是尝试提出自己的创新思路。

案例：引导同学们共同完成了一个基于单片机的自动化控制系统项目。在此过程中，团队成员利用各自的工具进行设计、仿真和测试。

教学手段：实操演练

主要内容：

4.2 Proteus 虚拟仿真工具介绍

Proteus 是英国 Lab center Electronics 公司在 1989 年推出的完全使用软件手段来对单片机应用系统进行虚拟仿真的软件工具。

4.2.1 Proteus 简介

Proteus 是目前世界上唯一的支持嵌入式处理器的虚拟仿真平台，它除了可仿真模拟电路、数字电路外，还可仿真 8051、PIC12/16/18 系列、AVR 系列、MSP430 等各主流系列单片机，以及各种外围可编程接口芯片。此外，它还支持 ARM7、ARM9 等型号的嵌入式微处理器的仿真。

由于 Proteus 的虚拟仿真，因此用户不需要用户硬件样机，就可直接在 PC 上对单片机系统进行虚拟仿真，将系统的功能及运行过程形象化，可以像焊接好的电路板一样看到单片机系统的执行效果。

4.2.2 Proteus ISIS 的虚拟仿真

Proteus ISIS（智能原理图输入）界面是用来绘制单片机系统的电路原理图，它还可直接实现单片机系统的虚拟仿真，可产生声、光及各种动作等逼真的效果。当电路连接无误后，单击单片机芯片载入经调试编译后生成的.hex 文件，单击仿真运行按钮，即可检验电路硬件及软件的设计正确与否。

Proteus 软件在 PC 上安装完后，单击桌面的 ISIS 运行界面图标，就会出现图 4-21 所示的 Proteus ISIS 原理电路图绘制界面（以汉化 7.5 版本为例）。

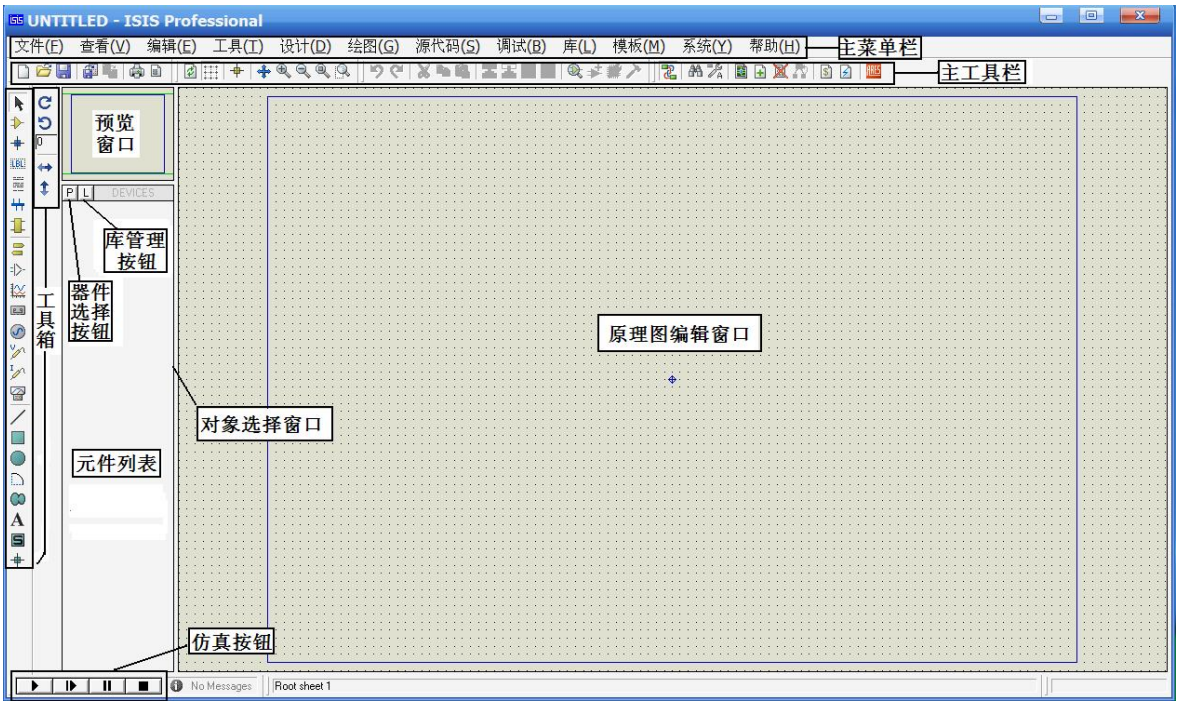


图 4-21　Proteus 的 ISIS 界面

整个 ISIS 界面分为若干个区域，由原理图编辑窗口、预览窗口、工具箱、主菜单栏、主工具栏等组成。

4.2.2.1　ISIS 各窗口简介

ISIS 界面主要有 3 个窗口：原理图编辑窗口、预览窗口和对象选择窗口。

1．原理图编辑窗口

用来绘制电路原理图、电路设计、设计各种符号模型的区域，图 4-21 所示的方框内为可编辑区，元件放置、电路设置都在此框中完成。

2．预览窗口

可对选中的元器件对象进行预览，同时可对原理图编辑窗口预览，如图 4-22 所示。它可显示两个内容：

（1）如果单击某个元件列表中的元件时，预览窗口会显示该元件的符号。

（2）当鼠标指针落在原理图窗口时（即放置元件到原理图编辑窗口后或在原理图编辑窗口中单击鼠标后），它会显示整张原理图的缩略图，并会显示一个绿色的方框，方框里面的内容就是当前原理图窗口中显示的内容。单击绿色方框中的某一点，就可拖动鼠标来改变绿色方框的位置，从而改变原理图的可视范围，最后在绿色方框内单击鼠标，绿色方框就不再移动，使得原理图的可视范围也就固定了。

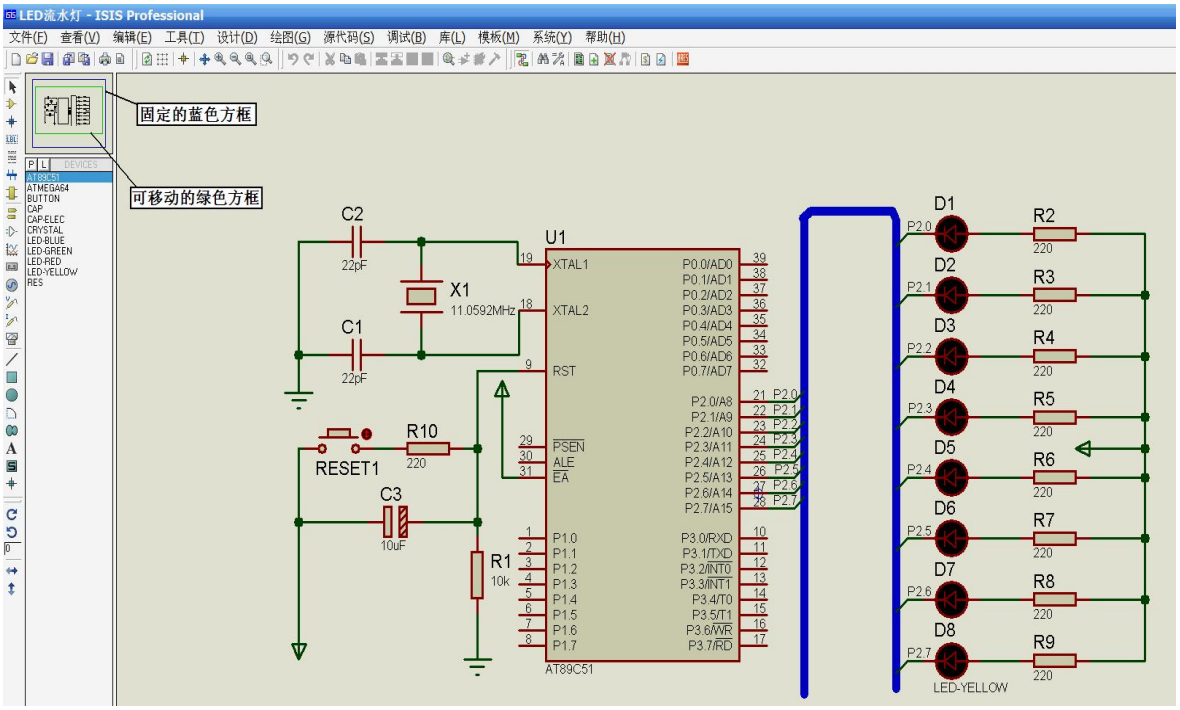


图 4-22　预览窗口调整原理图的可视范围

4.2.2.2　主菜单栏

图 4-21 最上面一行为主菜单栏，包含如下命令：文件、查看、编辑、工具、设计、绘图、源代码、调试、库、模板、系统和帮助。单击任意菜单命令后，都将弹出其下拉的子菜单命令列表。

1. 文件（File）菜单

文件菜单包括项目的新建设计、打开设计以及打印等操作，如图 4-24 所示。ISIS 下的文件主要是设计文件（Design Files），其文件扩展名为“.DSN”。它包括一个单片机硬件系统的原理电路图及其所有信息，用于虚拟仿真。

图4-23　元件列表

图4-24　文件菜单

下面介绍文件菜单下的“新建设计”命令。

单击【文件】→“新建设计”，会出现一张空的 A4 纸模板。新设计的默认名为“UNTITLED.DSN”，本命令会把该设计以这个名字存入磁盘文件中，文件的其他选项也会使用它作为默认名。

如果想进行新的设计，需要给这个设计命名，可单击【文件】→“保存设计”，输入新的文件名保存即可。

2．工具（Tools）菜单

工具菜单如图 4-25 所示。本菜单中的“自动连线（W）”命令文字前的快捷图标，在绘制电路原理图时出现，按下该图标即进入电路原理图的自动连线状态。

3．调试（Debug）菜单

调试菜单如图 4-26 所示，它主要完成单步运行、断点设置等功能。

图4-25 工具菜单　　图4-26 调试菜单

4.3.3 主工具栏

主工具栏位于主菜单下面，以图标形式给出，栏中共有 38 个快捷图标按钮：

每一个图标按钮都对应一个具体的菜单命令，主要目的是为了快捷方便地使用这些命令。下面把 38 个图标分为 4 组，简要介绍快捷图标命令的功能。

图标  的功能如下：

	新建一个设计文件
	打开一个已存在的设计文件
	保存当前的电路图设计
	将一个局部文件导入 ISIS 中
	将当前选中的对象导出为一个局部文件
	打印当前设计文件
	选择打印的区域

4.2.2.5　元件列表

如图 4-27 所示，元件列表用于挑选元件、终端接口、信号发生器、仿真图表等。

挑选元件时，单击 “P” 快捷图标，这时会打开挑选元件的对话框，在对话框中的“关键字”里面输入要检索的元器件的关键词，例如要选择使用 AT89C51，就可以直接输入。输入后能够在中间的“结果”栏里面看到搜索的元器件的结果。

在对话框的右侧，还能够看到选择的元器件的仿真模型以及 PCB 参数，选择了元件 AT89C51 后，双击 AT89C51，该元件就会在左侧的元件列表中显示，以后用到该元件时，只需在元件列表中选择即可。

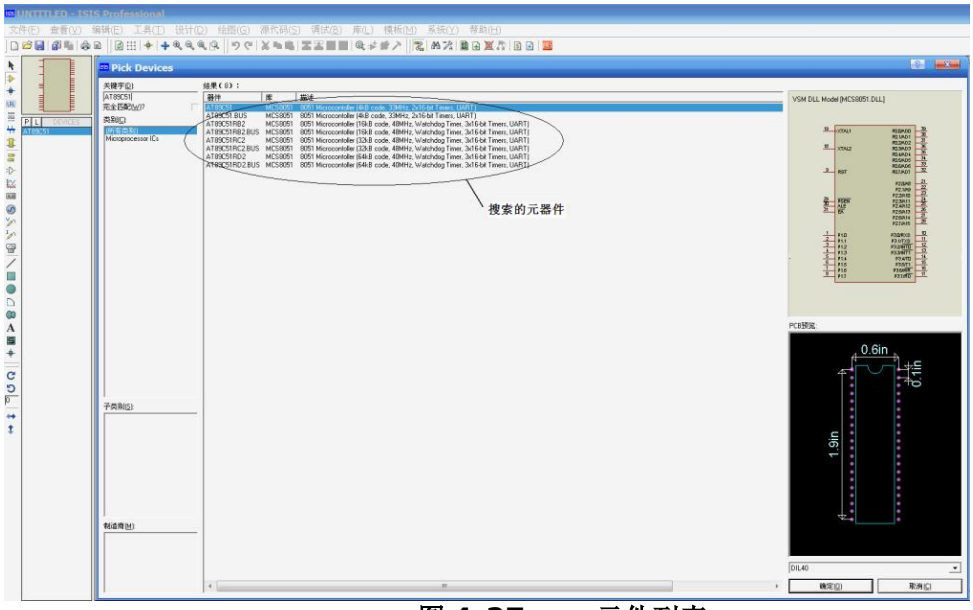


图 4-27　　元件列表

4.2.4　虚拟设计仿真举例

Proteus 环境下的一个单片机系统的原理电路虚拟设计与仿真需要 3 个步骤。

（1）Proteus ISIS 环境下的电路原理图设计。

（2）在 Keil C51 平台上进行源程序的输入、编译与调试，并最终生成目标代码文件（*.hex 文件）。

（3）调试与仿真，在 Proteus 环境下将目标代码文件（*.hex 文件）加载到单片机中，并对系统进行虚拟仿真。

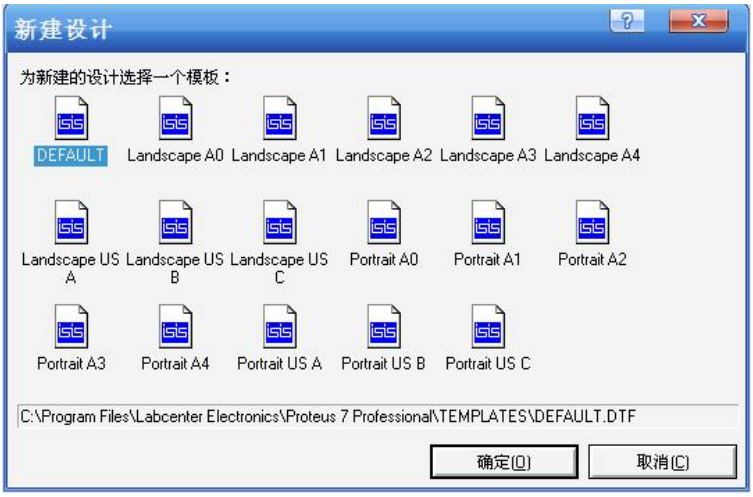
下面以“流水灯”的设计为例，介绍如何使用 Proteus。

1．建立新设计文件

单击主菜单的【文件】→“新建设计”选项来新建一个文件。如果选择新建设计文件，会弹出图 4-38 所示的“新建设计”窗口，窗口中有多种模板，单击要选的模板图标，再单击“确定”按钮，即建立一个该模板的空白文件。

如果直接单击“确定”按钮，即选用系统默认的“DEFAULT” 模板。如果用工具栏的快

捷图标按钮  来新建文件，就不会出现图 4-38 所示的窗口，而直接选择系统默认的模板。



2．保存文件

在建立了一个新的文件，第一次保存该文件时，选择菜单栏【文件】→“另存为（A）”选项，即弹出图 4-39 所示的“保存 ISIS 设计文件”窗口，在该窗口选择文件的保存路径和文件名“流水灯”，后，单击“保存”，则完成了设计文件的保存。

这样就在“实验 1（流水灯）”子目录下建立了一个文件名为“流水灯”的新的设计文件。

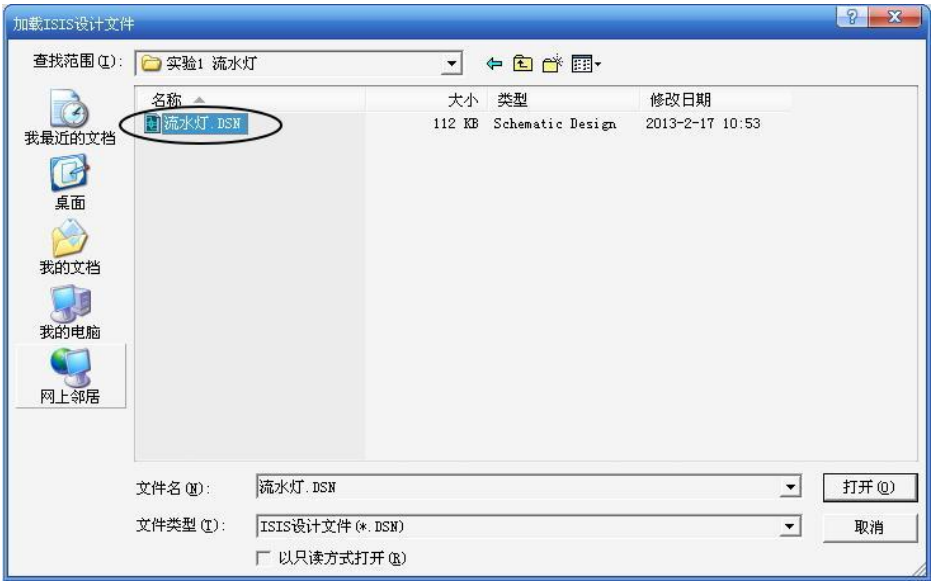
图 4-39　“保存 ISIS 设计文件”窗口

如果不是第一次保存，可选择菜单栏【文件】→“保存设计（S）”选项，或直接单击快捷图标按钮即可。

3．打开已保存的设计文件

选择菜单栏【文件】→“打开设计（O）”，或直接单击快捷按钮，将弹出图 4-40 所

示的“加载 ISIS 设计文件”窗口。单击需要打开的文件名，再单击“打开”按钮即可。



4.2.4.2　选择需要的元件到元件列表

电路设计前，要把设计“流水灯”电路原理图中需要的元器件列出，如表 4-3 所示。然后根据表 4-3 选择元件到元件列表中。观察图 4-21，左侧的元件列表中没有一个元件，单击左侧工具栏中的按钮 ，再单击器件选择按钮 就会出现“Pick Devices”窗口，在窗口的“关键字”栏中，输入 AT89C51，此时在“结果”栏中出现“元件搜索结果列表”，并在右侧出现“元件预览”和“元件 PCB 预览” ，如图 4-41 所示。

在“元件搜索结果列表”中双击所需要的元件 AT89C51，这时在主窗口的元件列表中就会添加该元件。用同样的方法可将表 4-3 中所需要选择的其他元件也添加到元件列表中。

表4-3　　流水灯所需元件列表

元件名称	型号	数量（个）	Proteus的关键字
单片机	AT89C51	1	AT89C51
晶体振荡器	12MHz	1	CRYSTAL
二极管	蓝色	8	LED-BLUE
二极管	绿色	8	LED-GREEN
二极管	红色	8	LED-RED
二极管	黄色	8	LED-YELLOW
电容	24pF	4	CAP
电解电容	10μF	1	CAP-ELEC
电阻	240Ω	10	RES
电阻	10kΩ	1	RES
复位按钮		1	BUTTON

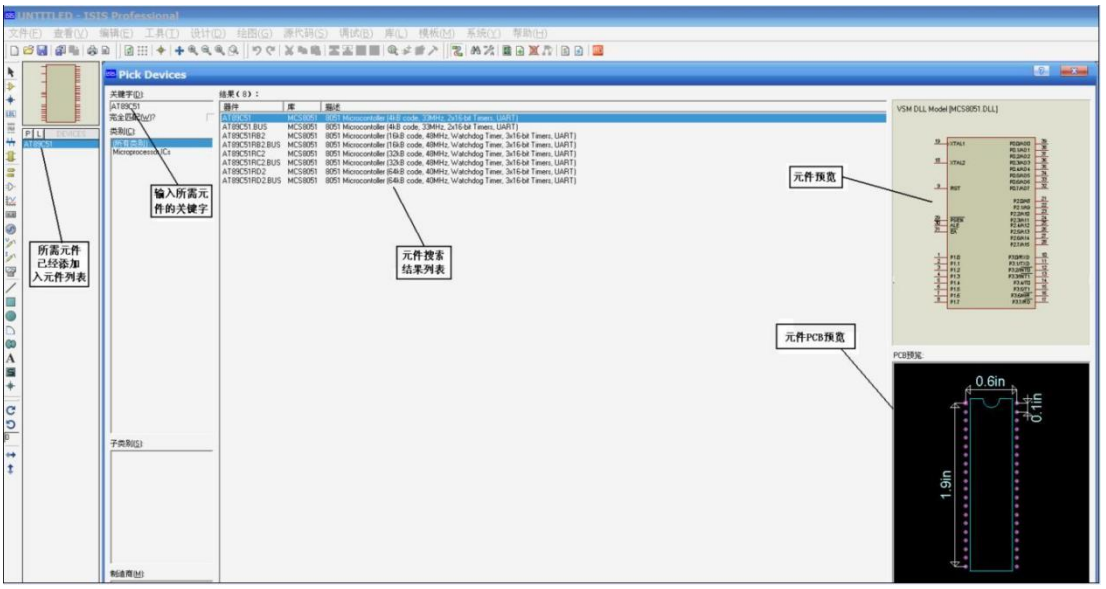


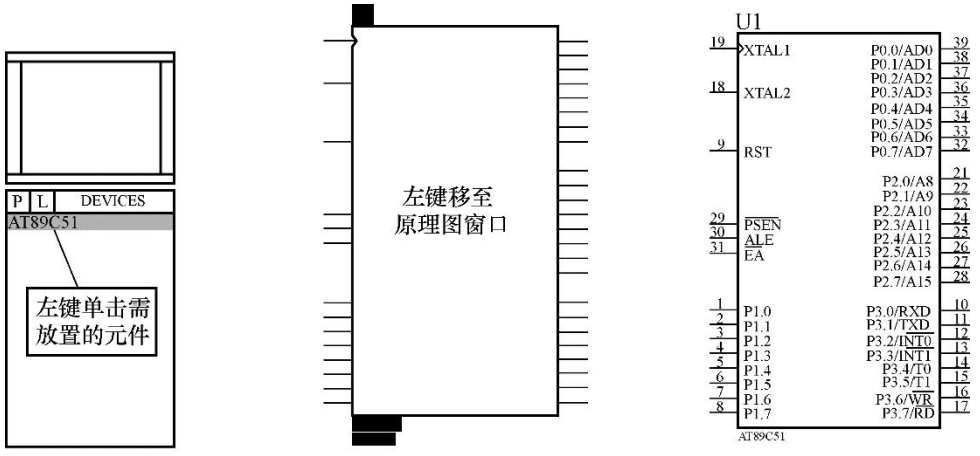
图 4-41　　“Pick Devices”窗口

所有元件选取完毕后，单击窗口右下方的“确定”按钮，即可关闭“Pick Devices”窗口，回到主界面进行原理图绘制。此时的“流水灯”设计的元件列表如图 4-23 所示。

4.2.4.3　放置元件并连接电路

1．元件的放置、调整与编辑

（1）元件的放置。单击元件列表中所需要放置的元器件，然后将鼠标移至原理图编辑窗口中单击一下，此时就会在鼠标指针处有一个粉红色的元器件，移动鼠标选择合适的位置，单击一下左键，此时该元件就被放置在原理图窗口了。例如选择放置单片机 AT89C51 到原理图编辑窗口，具体步骤如图 4-42 所示。



(a)　　(b)　　(c)

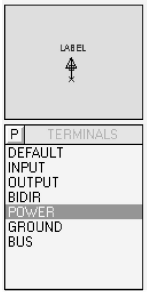
图 4-42　元件放置的操作步骤

若要删除已放置的元件，用鼠标左键单击该元件，然后按 Delete 键删除元件，如果进行了误删除操作，可以单击快捷按钮 恢复。

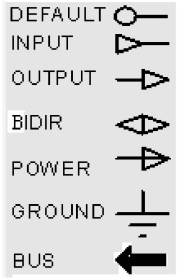
一个单片机系统电路原理图设计，除了元器件还需要各种终端，如电源、地等，单击工具栏

中的快捷按钮，就会出现各种终端列表，单击元件终端中的某一项，上方的窗口中就会出现该终端的符号，如图 4-43（a）所示。此时可选择合适的终端放置到电路原理图编辑窗口中去，放置的方法与元件放置相同。图 4-43（b）为图 4-43（a）列表中各项对应的终端符号。当再次单击按钮 时，即可切换回到用户自己选择的元件列表，如图 4-23 所示。

根据上述介绍，可将所有的元器件及终端放置到原理图编辑窗口中去。



(a)



(b)

（2）元件位置的调整。

a．改变元件在原理图中的位置，用鼠标左键单击需调整位置的元件，元件变为红颜色，移动鼠标指针到合适的位置，再释放鼠标即可。

b．调整元件的角度，用右键单击需调整的元件，会出现图 4-44 所示的菜单，操作菜单中的命令选项即可。

（3）元件参数设置。

用鼠标双击需要设置参数的元件，就会出现“编辑元件”窗口。下面以单片机 AT89C51 为例，双击 AT89C51，出现“编辑元件”窗口。

设计者可根据设计的需要，双击需要设置参数的元件，进入“编辑元件”窗口自行完成原理图中各元件的参数设置。

2．电路元件的连接

（1）两元件间绘制导线。在元件模式快捷按钮与自动布线器快捷按钮按下时，两个元件导线的连接方法是：先单击第一个元件的连接点，移动鼠标，此时会在连接点引出一根导线。如果想要自动绘出直线路径，只需单击另一个连接点。如果设计者想自己决定走线路径，只需在希望的拐点处单击鼠标左键。需要注意的是，拐点处导线的走线只能是直角。在自动布线器快捷按钮松开时，导线可按任意角度走线，只需要在希望的拐点处单击鼠标左键，把鼠标指针拉向目标点，拐点处导线的走向只取决于鼠标指针的拖动。

（2）连接导线连接的圆点。单击连接点按钮，会在两根导线连接处或两根导线交叉处添加一个圆点，表示它们是连接的。

（3）导线位置的调整。对某一绘制的导线，要想进行位置的调整，可用鼠标左键单击导线，导线两端各有一个小黑方块，单击右键出现菜单，如图 4-45 所示。单击“拖曳对象”，即可拖曳导线到指定的位置，也可进行旋转，然后单击导线，这就完成了导线位置的调整。

（4）绘制总线与总线分支。

总线绘制：单击工具栏的图标按钮，移动鼠标到绘制总线的起始位置，单击鼠标左键，便可绘制出一条总线。如想要总线出现不是 90° 角的转折，此时自动布线器快捷按钮 应当松开，总线即可按任意角度走线，只需要在希望的拐点处单击鼠标左键，把鼠标指针拉向目标点，在总线的终点处双击鼠标左键，即结束总线的绘制。

总线分支绘制：总线绘制完后，有时还需绘制总线分支。为使电路图显得专业和美观，通常要把总线分支画成与总线成 45° 角的相互平行的斜线，如图 4-46 所示。注意，此时一定要把自动布线器快捷按钮松开，总线分支走向只取决于鼠标指针的拖动。

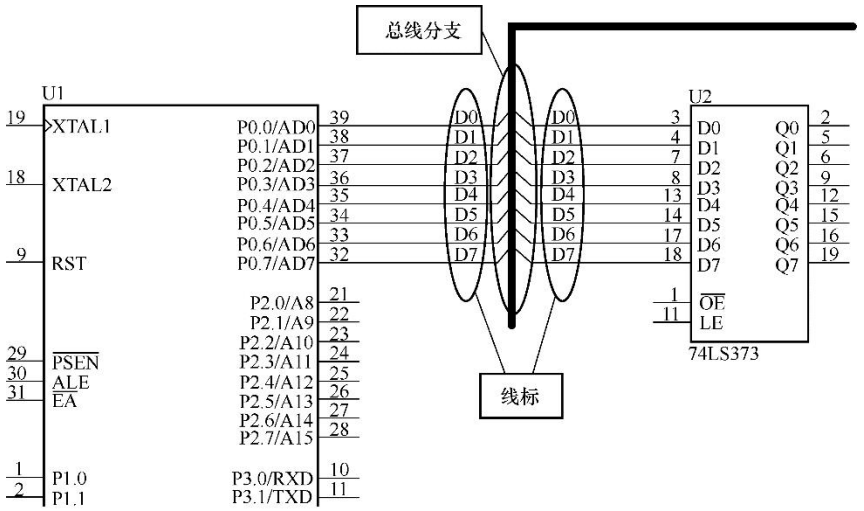


图 4-46　　总线与总线分支及线标

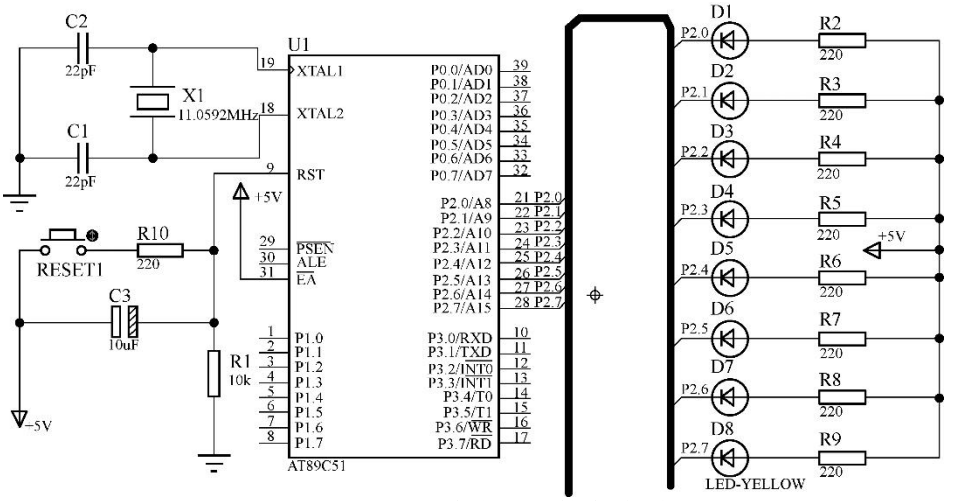
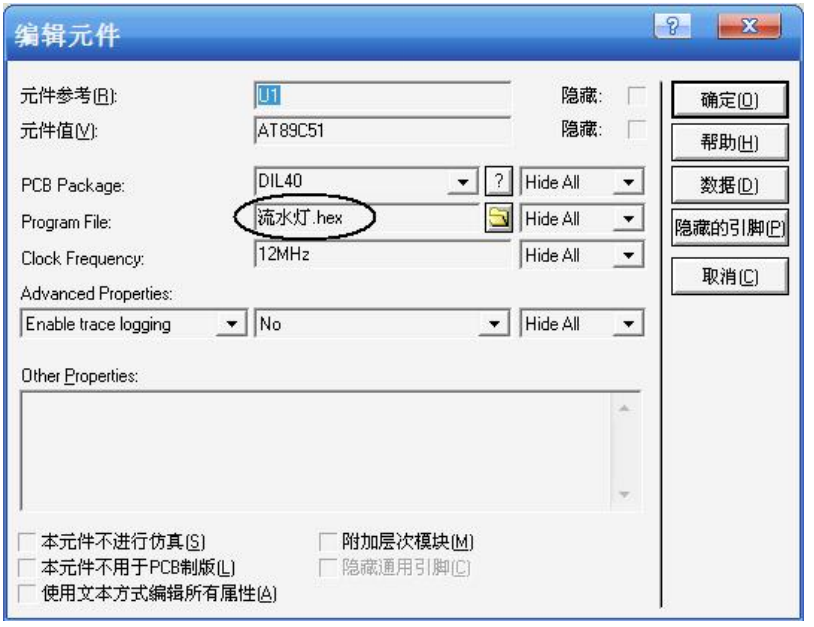


图 4-48　　“流水灯”的电路原理图



2. 仿真运行

单击 Proteus ISIS 界面中的快捷命令按钮  即可运行程序。图 4-21 左下角的各种仿真运行命令按钮功能如下：

 ：运行程序。

 ：单步运行程序。

 ：暂停程序。

 ：停止运行程序。

本节作业

根据以上步骤，完成流水灯电路仿真。

第五章　单片机与开关、键盘以及显示器件的接口设计（18 学时）

本章课时分配　　本章分为 6 讲，共 18 学时。

5.1　单片机控制发光二极管显示

教学目标：（教学 4 学时）

1、掌握单片机外接发光二极管的接线

2、掌握 c51 编程方法点亮 LED 灯

教学重点

1．单片机控制发光二极管显示

教学难点

1．单片机控制发光二极管显示

素质（思政）内容与要求：

思政融入点：鼓励学生在控制 LED 显示的过程中尝试新的方法或设计，比如使用不同的控制算法来实现动态效果。

案例：引导同学们利用单片机控制 LED 制作了一个具有创意的显示装置，如动态文字滚动显示板或 LED 艺术墙。

教学手段：讲授+实训

主要内容：

发光二极管介绍

发光二极管常用来指示系统工作状态，制作节日彩灯、广告牌匾等。大部分发光二极管工作电流 1~5mA 之间，其内阻为 20~100Ω。电流越大，亮度也越高。

为保证发光二极管正常工作，同时减少功耗，限流电阻选择十分重要，若供电电压为+5V，则限流电阻可选 1~3kΩ。

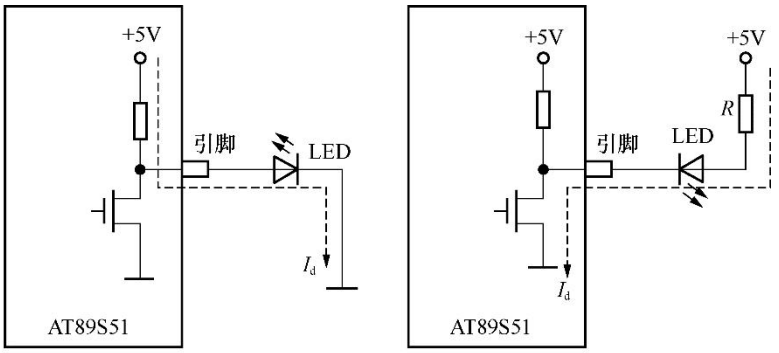
5.1.1　单片机与发光二极管的连接

第 2 章已介绍，P0 口作通用 I/O 用，由于漏极开路，需外接上拉电阻。而 P1～P3 口内部有 30kΩ左右上拉电阻。

下面讨论 P1～P3 口如何与 LED 发光二极管驱动连接问题。

单片机并行端口 P1～P3 直接驱动发光二极管，电路见图 5-1。

与 P1、P2、P3 口相比，P0 口每位可驱动 8 个 LSTTL 输入，而 P1～P3 口每一位驱动能力，只有 P0 口一半。



(a) 不恰当的连接：高电平驱动　　(b) 恰当的连接：低电平驱动

图 5-1　发光二极管与单片机并行口的连接

当 P0 口某位为高电平时，可提供 400μA 的拉电流；当 P0 口某位为低电平（0.45V）时，可提供 3.2mA 的灌电流，而 P1～P3 口内有 30kΩ左右上拉电阻，如高电平输出，则从 P1、P2 和 P3 口输出的拉电流 Id 仅几百μA，驱动能力较弱，亮度较差，见图 5-1（a）。

如果端口引脚为低电平，能使灌电流 Id 从单片机外部流入内部，则将大大增加流过的灌电流值，见图 5-1（b）。AT89S51 任一端口要想获得较大的驱动能力，要用低电平输出。

如果一定要高电平驱动，可在单片机与发光二极管间加驱动电路，如 74LS04、74LS244 等。

5.1.2　I/O 端口的编程举例

单片机的 I/O 端口 P0~P3 是单片机与外设进行信息交换的桥梁，可通过读取 I/O 端口的状态来了解外设的状态，也可向 I/O 端口送出命令或数据来控制外设。

对单片机 I/O 端口进行编程控制时，需要对 I/O 端口的特殊功能寄存器进行声明，在 C51 的编译器中，这项声明包含在头文件 reg51.h 中，编程时，可通过预处理命令 #include<reg51.h>，把这个头文件包含进去。下面通过一个例子介绍如何对 I/O 端口编程实现对发光二极管亮灭的控制。

【例 5-1】 制作流水灯，原理电路见图 5-2，8 个发光二极管 LED0～LED7 经限流电阻分别接至 P1 口的 P1.0～P1.7 引脚上，阳极共同接高电平。编写程序来控制发光二极管由上至下的反复循环流水点亮，每次点亮一个发光二极管。

参考程序：

```
#include <reg51.h>
#include <intrins.h>          //包含移位函数_crol_(   )的头文件
#define uchar unsigned char
#define uint unsigned int
void   delay(uint i)        //延时函数
{
    uchar t;
    while (i--)
  {
        for(t=0;t<120;t++);
    }
}
```

```
void   main(   )                //主程序
{
    P1=0xfe;                    //向 P1 口送出点亮数据
    while (1)
    {
        delay( 500 );           //500 为延时参数，可根据实际需要调整
        P1=_crol_(P1,1) ;       // 函数_crol_(P1,1)把 P1 中的数据循环左移 1 位
    }
}
```

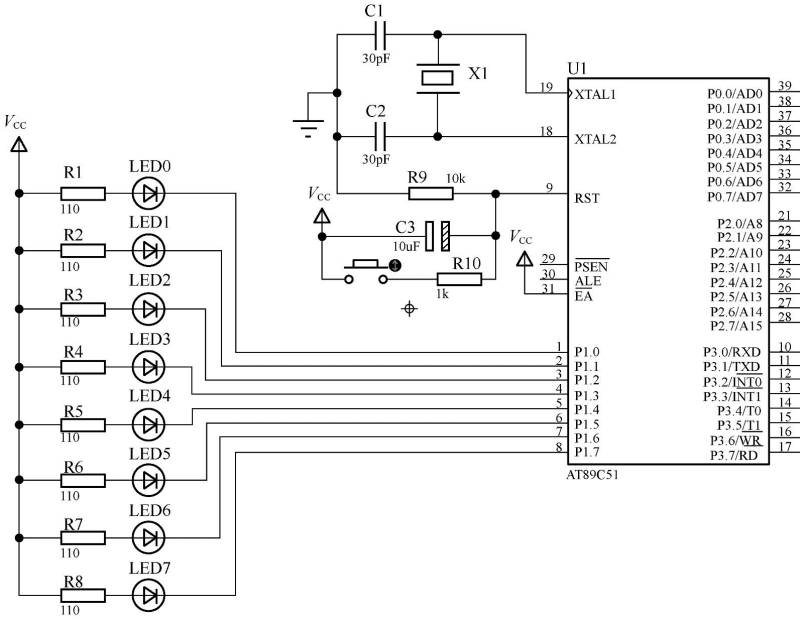


图 5-2　单片机控制的流水灯

【例 5-2】电路见图 5-2，制作由上至下再由下至上反复循环点亮显示的流水灯，3 种方法实现。

（1）数组的字节操作实现

建立 1 个字符型数组，将控制 8 个 LED 显示的 8 位数据作为数组元素，依次送 P1 口。参考程序：

```
#include <reg51.h>
#define uchar unsigned char
uchar tab[ ]={ 0xfe , 0xfd , 0xfb , 0xf7 , 0xef , 0xdf , 0xbf , 0x7f , 0x7f , 0xbf ,
0xdf , 0xef , 0xf7 ,  0xfb , 0xfd , 0xfe };
                                        /*前 8 个数据为左移点亮 数据，后 8 个为右移
点亮数据*/
void   delay(   )
{
    uchar i,j;
    for(i=0; i<255; i++)
    for(j=0; j<255; j++);
}
```

```
void   main(   )                    //主函数
{
    uchar i;
    while (1)
    {
        for(i=0;i<16; i++)
        {
            P1=tab[i];  //向 P1 口送出点亮数据
            delay( );    //延时，即点亮一段时间
        }
    }
}
```

【例 5-3】 如图 5-3，单片机的 P1.4～P1.7 接 4 个开关 S0～S3，P1.0～P1.3 接 4 个发光二极管 LED0～LED3。

编程将 P1.4～P1.7 上的 4 个开关状态反映在 P1.0～P1.3 引脚控制的 4 个发光二极管上，开关闭合，对应发光二极管点亮。

例如 P1.4 引脚上开关 S0 状态，由 P1.0 脚上 LED0 显示，P1.6 引脚上开关 S2 状态，由 P1.2 脚的 LED2 显示。

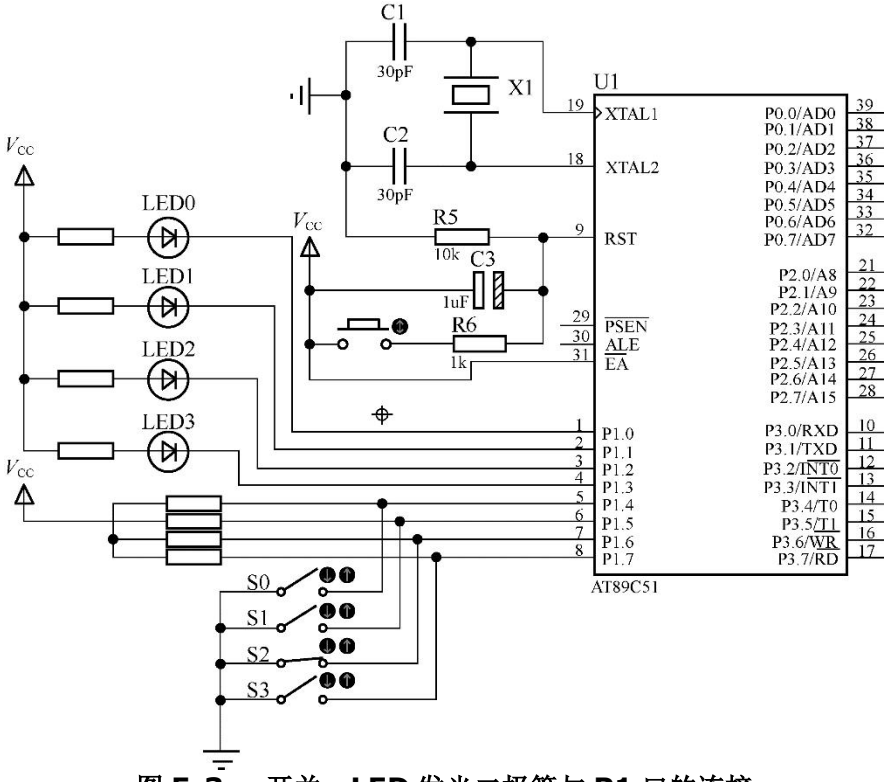


图 5-3　开关、LED 发光二极管与 P1 口的连接

参考程序如下：

```
#include <reg51.h>
#define uchar unsigned char
void   delay(   )                    //延时函数
{
    uchar i,j;
    for(i=0; i<255; i++)
```

```
    for(j=0; j<255; j++);
}
void   main(   )                    //主函数
{
while (1)
    {
        unsigned char temp;     //定义临时变量 temp
    P1=0xff;        //P1 口低 4 位置 1，作为输入;高 4 位置 1，发光二极管熄灭
    temp=P1&0xf0;           //读 P1 口并屏蔽低 4 位，送入 temp  中
    temp=temp>>4;                //temp 内容右移 4 位，P1 口高 4 位移至低 4 位
    P1=temp;        // temp 中的数据送 P1 口输出
    delay(   );
 }
```

作业：

实操练习例题，完成画图，打代码，仿真。

5.2　开关状态检测

教学目标：（教学 4 学时）

1、掌握单片机开关的接线方法

2、检测一个开关处于闭合状态还是打开状态

教学重点

1．单片机检测开关闭合打开的方法

教学难点

1．单片机检测开关闭合打开的方法

教学手段：讲授+实训

主要内容：

5.2.1　开关检测案例 1

用 I/O 端口来进行开关状态检测，开关一端接到 I/O 端口引脚上，并通过上拉电阻接+5V 上，开关另一端接地，当开关打开时，I/O 引脚为高电平，当开关闭合时，I/O 引脚为低电平。

【例 5-3】 如图 5-3，单片机的 P1.4～P1.7 接 4 个开关 S0～S3，P1.0～P1.3 接 4 个发光二极管 LED0～LED3。

编程将 P1.4～P1.7 上的 4 个开关状态反映在 P1.0～P1.3 引脚控制的 4 个发光二极管上，开关闭合，对应发光二极管点亮。

例如 P1.4 引脚上开关 S0 状态，由 P1.0 脚上 LED0 显示，P1.6 引脚上开关 S2 状态，由 P1.2 脚的 LED2 显示。

参考程序如下：

```
#include <reg51.h>
#define uchar unsigned char
void   delay(   )                          //延时函数
{
    uchar i,j;
    for(i=0; i<255; i++)
    for(j=0; j<255; j++);
}
void   main(   )                //主函数
{
while (1)
    {
        unsigned char temp;     //定义临时变量 temp
    P1=0xff;       //P1 口低 4 位置 1，作为输入;高 4 位置 1，发光二极管熄灭
    temp=P1&0xf0;          //读 P1 口并屏蔽低 4 位，送入 temp  中
    temp=temp>>4;                //temp 内容右移 4 位，P1 口高 4 位移至低 4 位
    P1=temp;        // temp 中的数据送 P1 口输出
    delay(   );
 }
```

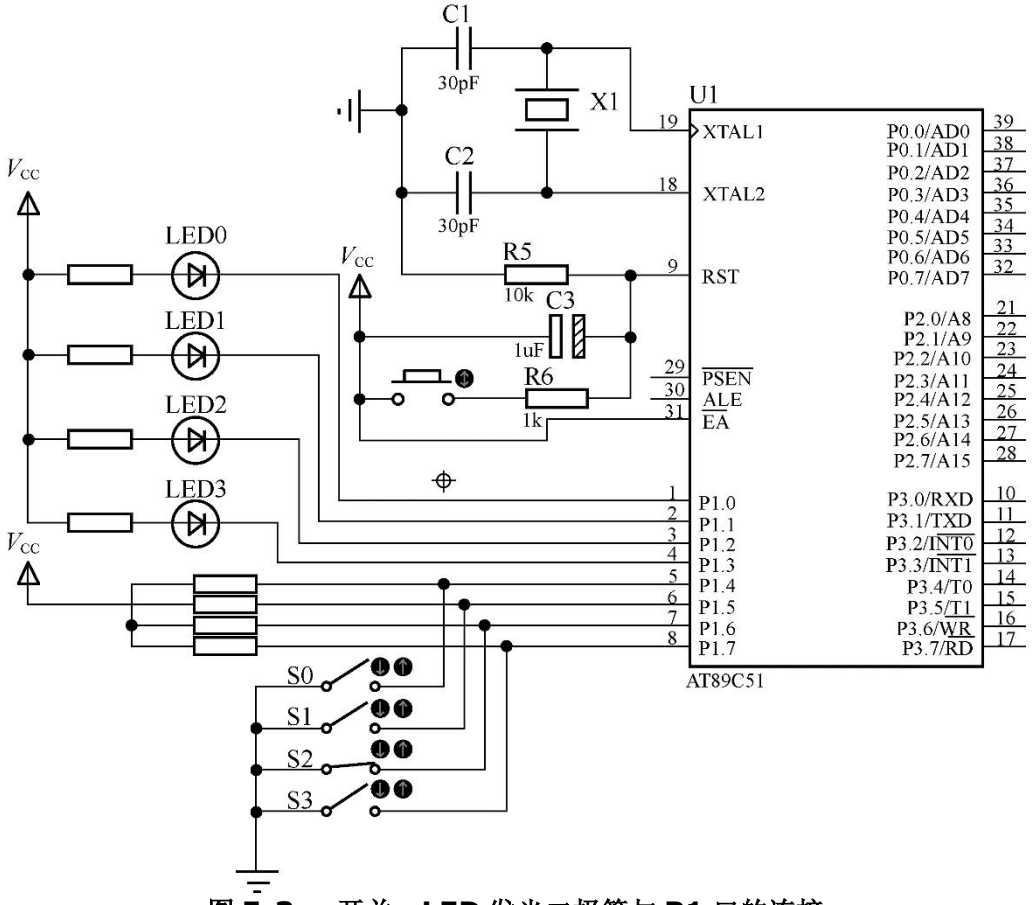


图 5-3　开关、LED 发光二极管与 P1 口的连接

5.2.2　开关检测案例 2

【例 5-4】 如图 5-4，P1.0 和 P1.1 引脚接有两只开关 S0 和 S1，两引脚上的高低电平共 4 种组合，4 种组合分别点亮 P2.0～P2.3 引脚控制的 4 只 LED，即 S0、S1 均闭合，LED0 亮，其余灭；S1 闭合、S0 打开，LED1 亮，其余灭；S0 闭合、S1 打开，LED2 亮，其余灭；S0、S1 均打开，LED3 亮，其余灭。编程实现此功能。

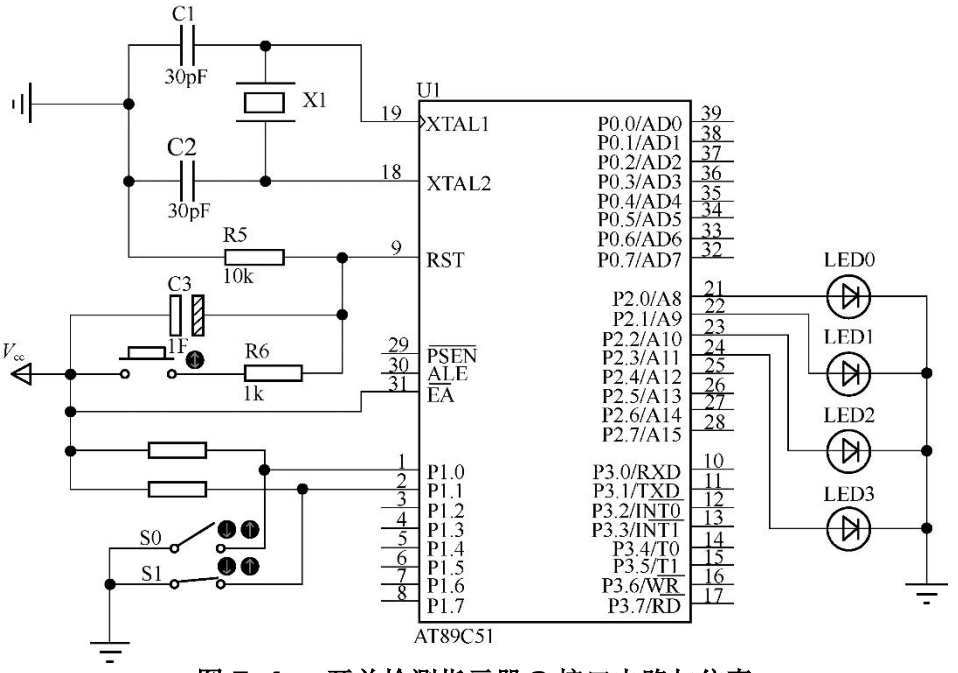


图 5-4　开关检测指示器 2 接口电路与仿真

参考程序：

```
#include <reg51.h>              // 包含头文件 reg51.h
void   main(   )               //主函数 main( )
{
    char state;
    do
    {
        P1=0xff;                // P1 口为输入
        state=P1;           // 读入 P1 口的状态，送入 state
        state=state&0x03;                   // 屏蔽 P1 口的高 6 位
        switch (state)          // 判 P1 口低 2 位开关状态
        {
            case 0: P2=0x01; break;         // 点亮 P2.0 脚 LED
            case 1: P2=0x02; break;         //点亮 P2.1 脚 LED
            case 2: P2=0x04; break;         // 点亮 P2.2 脚 LED
            case 3: P2=0x08; break;         //点亮 P2.3 脚 LED
        }
    }while ( 1 );
}
```

作业：

实操练习例题，完成画图，打代码，仿真

5.3　单片机控制 LED 数码管的显示

教学目标：（教学 4 学时）

1、掌握数码管的内部原理

2、能用单片机控制数码管显示对应数字

教学重点

1．LED 数码管显示原理

2．单片机控制数码管显示的代码编程

教学难点

1．LED 数码管显示原理

2．单片机控制数码管显示的代码编程

素质（思政）内容与要求：

思政融入点：通过讲解单片机控制 LED 数码管显示的实际应用场景，帮助学生认识到技术对于社会发展的积极作用。

案例：鼓励学生在完成基本功能后，尝试编写一些具有创新性的显示效果，如倒计时功能、闹钟提醒等。激发同学们的创造力。

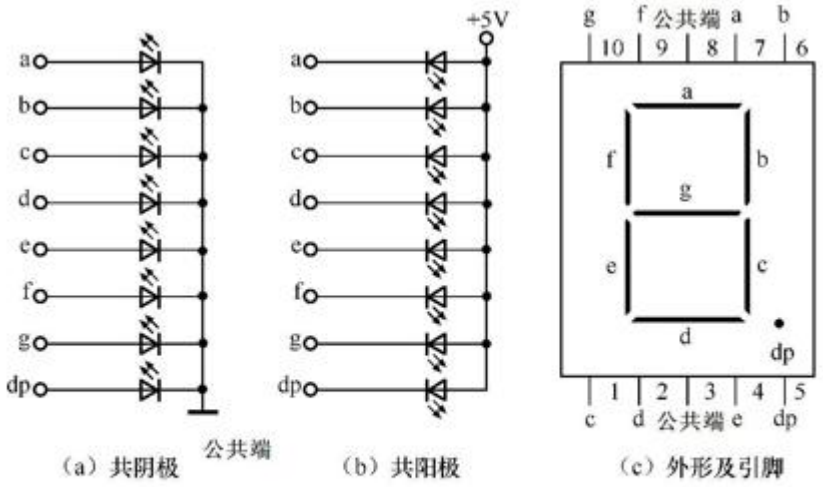
教学手段：讲授+实训

主要内容：

5.3.1　LED 数码管显示原理

LED 数码管：　“8”字型，7 段（不包括小数点）或 8 段（包括小数点），每段对应一个发光二极管，共阳极和共阴极两种，见图 5-5。共阳极数码管的阳极连接在一起，接+5V；共阴极数码管阴极连在一起接地。

对于共阴极数码管，当某发光二极管阳极为高电平时，发光二极管点亮，相应段被显示。同样，共阳极数码管阳极连在一起，公共阳极接+5V，当某个发光二极管阴极接低电平时，该发光二极管被点亮，相应段被显示。



（a）共阴极　（b）共阳极　（c）外形及引脚

图5-5　8段LED数码管结构及外形

为使LED数码管显示不同字符，要把某些段点亮，就要为数码管各段提供一字节的二进制码，即字型码（也称段码）。习惯上以“a”段对应字型码字节的最低位。各字符段码见表5-1。

表5-1　LED数码管的段码

显示字符	共阴极字型码	共阳极字型码	显示字符	共阴极字型码	共阳极字型码
0	3FH	C0H	C	39H	C6H
1	06H	F9H	d	5EH	A1H
2	5BH	A4H	E	79H	86H
3	4FH	B0H	F	71H	8EH
4	66H	99H	P	73H	8CH
5	6DH	92H	U	3EH	C1H
5	7DH	82H	T	31H	CEH
7	07H	F8H	y	6EH	91H
8	7FH	80H	H	76H	89H
9	6FH	90H	L	38H	C7H
A	77H	88H	“灭”	00H	FFH
b	7CH	83H	…	…	…

【例 5-5】利用单片机控制一个 8 段 LED 数码管先循环显示单个偶数：0、2、4、6、8，再显示单个奇数：1、3、5、7、9，如此反复循环显示。

本例原理电路及仿真结果，见图 5-6。

参考程序如下：

```
#include "reg51.h"
#include "intrins.h"
#define uchar unsigned char
#define uint unsigned int
#define out P0
```

```
    uchar code
seg[]={0xc0,0xa4,0x99,0x82,0x80,0xf9,0xb0,0x92,0xf8,0x90,0x01};
                                        //共阳极段码表
    void delayms(uint);
    void main(void)
    {
    uchar i;
    while(1)
        {
        out=seg[i];
        delayms(900);
        i++;
        if(seg[i]==0x01)i=0;    // 如段码为 0x01，表明一个循环显示已结束
        }
    }
    void delayms(uint j)                    // 延时函数
    {
    uchar i;
    for(;j>0;j--)
        {
            i=250;
            while(--i);
            i=249;
            while(--i);
        }
    }
```

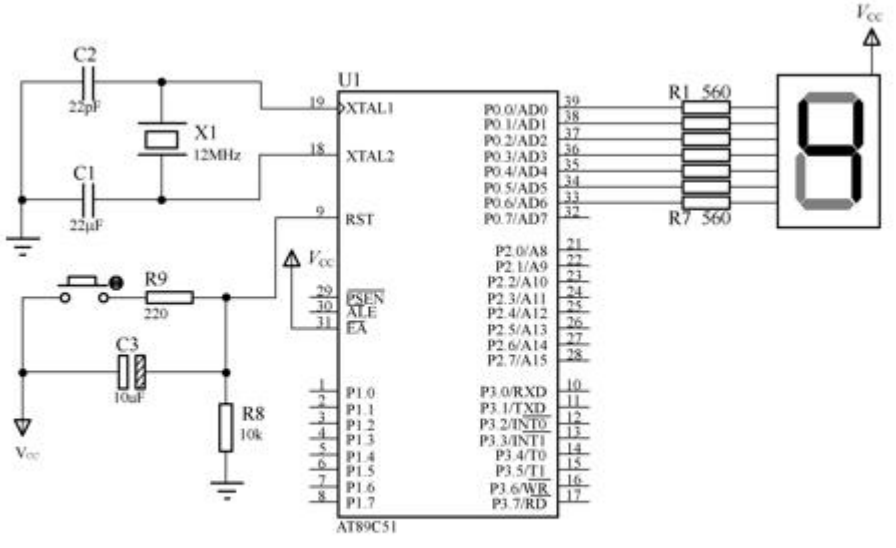


图5-6　控制数码管循环显示单个数字的电路及仿真

5.3.2　LED 数码管的静态显示与动态显示

两种显示方式：静态显示和动态显示。

1. 静态显示方式

无论多少位 LED 数码管，都同时处于显示状态。

多位 LED 数码管工作于静态显示方式时，各位共阴极（或共阳极）连接在一起并接地（或接+5V）；每位数码管段码线（a～dp）分别与一个 8 位 I/O 口锁存器输出相连。如果送往各个 LED 数码管所显示字符的段码一经确定，则相应 I/O 口锁存器锁存的段码输出将维持不变，直到送入下一个显示字符段码。静态显示方式显示无闪烁，亮度较高，软件控制较易。

图 5-7 为 4 位 LED 数码管静态显示电路，各数码管可独立显示，只要向控制各位 I/O 口锁存器送相应显示段码，该位就能保持相应的显示字符。

这样在同一时间，每一位显示的字符可各不相同。静态显示方式占用 I/O 口端口线较多。图 5-7 电路，要占用 4 个 8 位 I/O 口（或锁存器）。如数码管数目增多，则需增加 I/O 口数目。

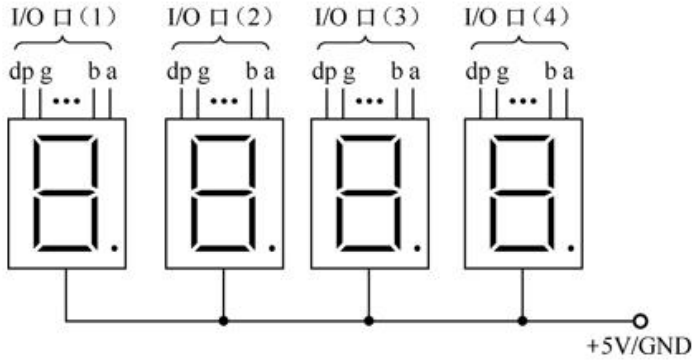


图5-7　4位LED静态显示的示意图

【例 5-6】单片机控制 2 只数码管，静态显示 2 个数字“27”。 原理电路见图 5-8。

单片机用 P0 口与 P1 口，分别控制加到两个数码管 DS0 与 DS1 的段码，而共阳极数码管 DS0 与 DS1 的公共端（公共阳极端）直接接至+5V，因此数码管 DS0 与 DS1 始终处于导通状态。利用 P0 口与 P1 口带有的锁存功能，只需向单片机 P0 口与 P1 口分别写入相应的显示字符“2”和“7”的段码即可。

由于一个数码管就占用一个 I/O 端口。如果数码管数目增多，则需增加 I/O 口，但软件编程要简单的多。

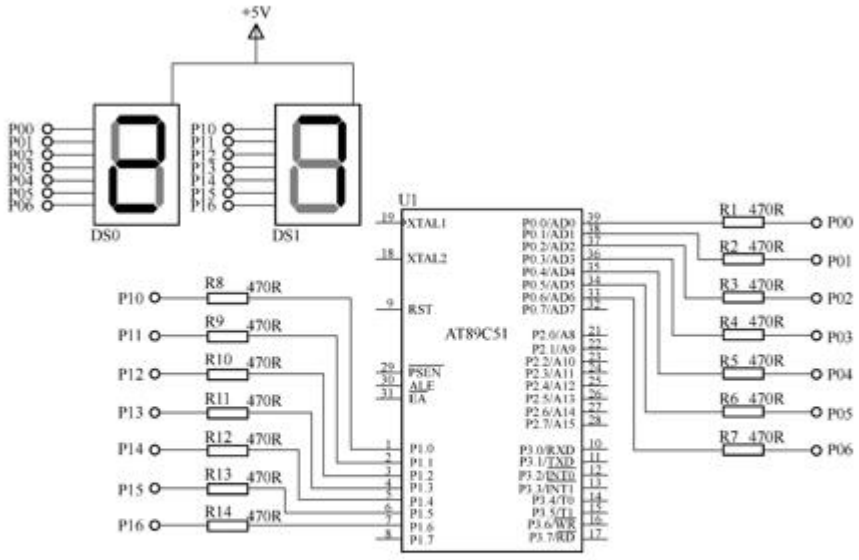


图5-8　2位数码管静态显示的原理电路与仿真

参考程序如下：

```
#include<reg51.h>      //包含 8051 单片机寄存器定义的头文件
void main(void)
{
   P0=0xa4;          //将数字"2"的段码送 P0 口
   P1=0xf8;          //将数字"7"的段码送 P1 口
   while(1)       //无限循环
   ;
}
```

2. 动态显示方式

显示位数较多时，静态显示所占的 I/O 口多，这时常采用动态显示。为节省 I/O 口，通常将所有显示器段码线相应段并联在一起，由一个 8 位 I/O 口控制，各显示位公共端分别由另一单独 I/O 口线控制。

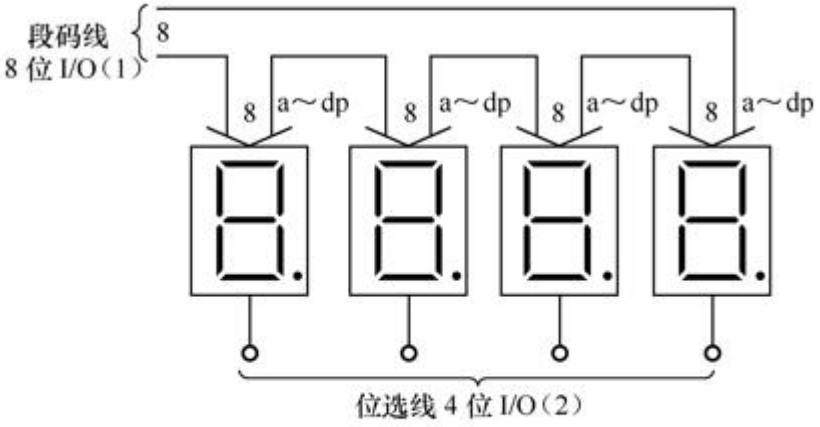


图5-9　4位LED数码管动态显示示意图

图 5-9：4 位 8 段 LED 动态显示器电路示意图。其中单片机发出的段码占用 1 个 8 位 I/O（1）端口，而位选控制使用 I/O（2）端口中 4 位口线。

动态显示就是单片机向段码线输出欲显示字符的段码。每一时刻，只有 1 位位选线有效，即选中某一位显示，其他各位位选线都无效。每隔一定时间逐位轮流点亮各数码管（扫描方式），由于数码管余辉和人眼的“视觉暂留”作用，只要控制好每位数码管显示时间和间隔，则可造成“多位同时亮”的假象，达到同时显示效果。

各位数码管轮流点亮的时间间隔（扫描间隔）应根据实际情况定。发光二极管从导通到发光有一定的延时，如果点亮时间太短，发光太弱，人眼无法看清；时间太长，产生闪烁现象，且此时间越长，占用单片机时间也越多。另外，显示位数增多，也将占用单片机大量时间，因此动态显示实质是以执行程序时间来换取 I/O 端口减少。下面是动态显示实例。

【例 5-7】 8 只数码管，分别滚动显示单个数字 1～8。程序运行后，单片机控制左边第 1 个数码管显示 1，其他不显示，延时之后，控制左边第 2 个数码管显示 1，其他不显示，直至第 8 个数码管显示 8，其他不显示，反复循环上述过程。

动态显示电路见图 5-10，P0 口输出段码，P2 口输出扫描的位控码，通过由 8 个 NPN 晶体管的位驱动电路对 8 个数码管位控扫描。

由于数码管余辉和人眼 “视觉暂留”作用，只要控制好每位数码管显示的时间和间隔，则可造成“多位同时亮” 假象，达到同时显示效果。

本例使我们了解动态扫描显示实际过程。

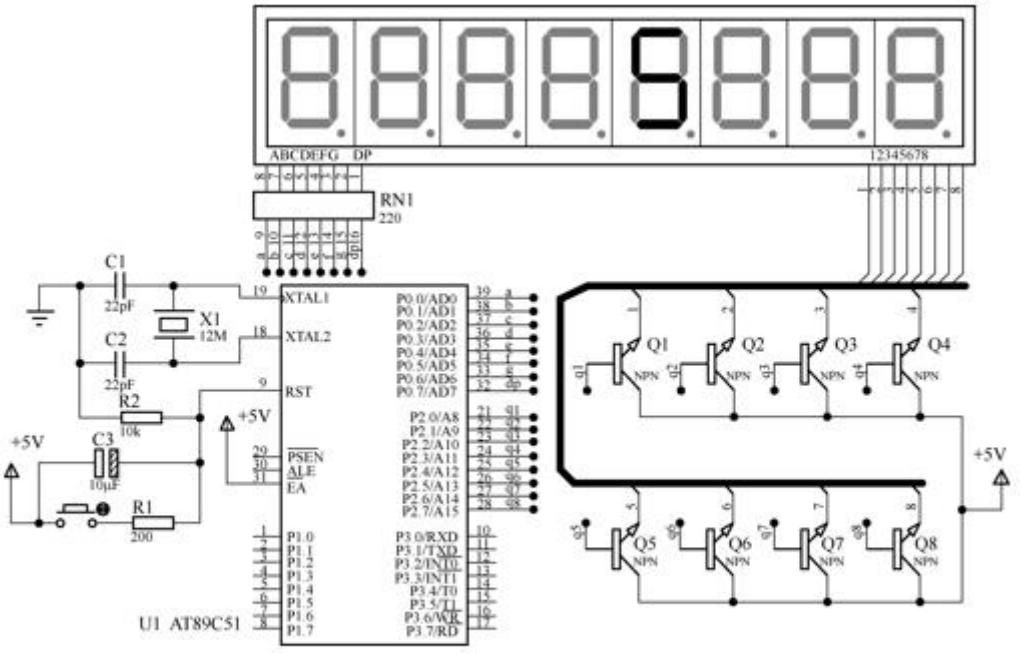


图5-10　8只数码管分别滚动显示单个数字1~8

参考程序如下：

```
#include<reg51.h>
#include<intrins.h>
#define uchar unsigned char
#define uint unsigned int
uchar code
dis_code[]={0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90,0x88,0xc0};
                                        //共阳数码管段码表
void   delay(uint t)                    //延时函数
{
    uchar i;
    while(t--) for(i=0;i<200;i++);
}
void   main()
{
    uchar i,j=0x80;
     while(1)
     {
    for(i=0;i<8;i++)
    {
            j=_crol_(j,1);    //_crol_(j,1)为将对象 j 循环左移 1 位
            P0=dis_code[i]; //P0 口输出段码
            P2=j;          //P2 口输出位控码
            delay(180);       //延时，控制每位显示的时间
    }
       }
}
```

作业：

实操练习例题，完成画图，打代码，仿真

5.4　LED 点阵显示器结构与显示原理

教学目标：（教学 6 学时）

1、掌握 LED 点阵显示器结构与显示原理

2、能用单片机控制 16*16LED 点阵显示流水字体

教学重点

1．LED 点阵显示器结构与显示原理

2．单片机控制 LED 点阵的代码编程

教学难点

1．LED 点阵显示器结构与显示原理

2．单片机控制 LED 点阵的代码编程

素质（思政）内容与要求：

思政融入点：鼓励学生在学习 LED 点阵显示原理的过程中，尝试创新设计，如使用不同的驱动方式或显示模式。

案例：鼓励学生讨论他们的作品如何应用于实际生活中，帮助人们更好地传递信息或美化环境。

教学手段：讲授+实训

5.4.1　LED 点阵显示器结构与显示原理

由若干个发光二极管按矩阵方式排列而成。阵列点数可分为 5×7、5×8、6×8、8×8 点阵；按发光颜色可分为单色、双色、三色；按极性排列可分为共阴极和共阳极。

1.　LED 点阵结构

以 8×8LED 点阵显示器为例，外形见图 5-11，内部结构见图 5-12，由 64 个发光二极管组成，且每个发光二极管是处于行线（R0~R7）和列线（C0~C7）之间交叉点上。

2. LED 点阵显示原理

显示的字符由一个个点亮的 LED 所构成。

由图 5-12 点亮点阵中一个发光二极管条件：对应行为高电平，对应列为低电平。如在很短时间内依次点亮很多个发光二极管，LED 点阵就可显示一个稳定字符、数字或其他图形。控制 LED 点阵显示器显示，实质就是控制加到行线和列线上编码，控制点亮某些发光二极管（点），从而显示出由不同发光点组成的各种字符。

16×16 LED 点阵显示器的结构与 8×8LED 点阵显示模块内部结构及显示原理是类似的，只不过行和列均为 16。16×16 是由 4 个 8×8 LED 点阵组成，且每个发光二极管也是放置在行线和列线的交叉点上，当对应某一列置 0 电平，某一行置 1 电平时，该发光二极管点亮。

下面以显示字符“子”为例，见图 5-13。

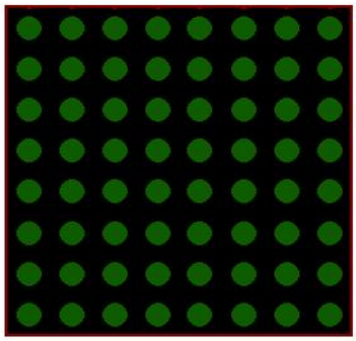


图5-11　8×8 LED点阵显示器外形

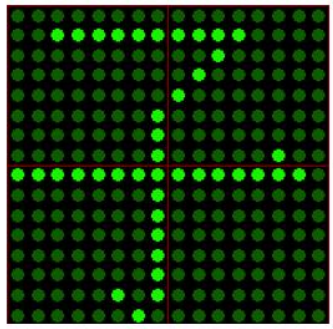


图5-13　16×16 LED点阵显示器显示字符“子”

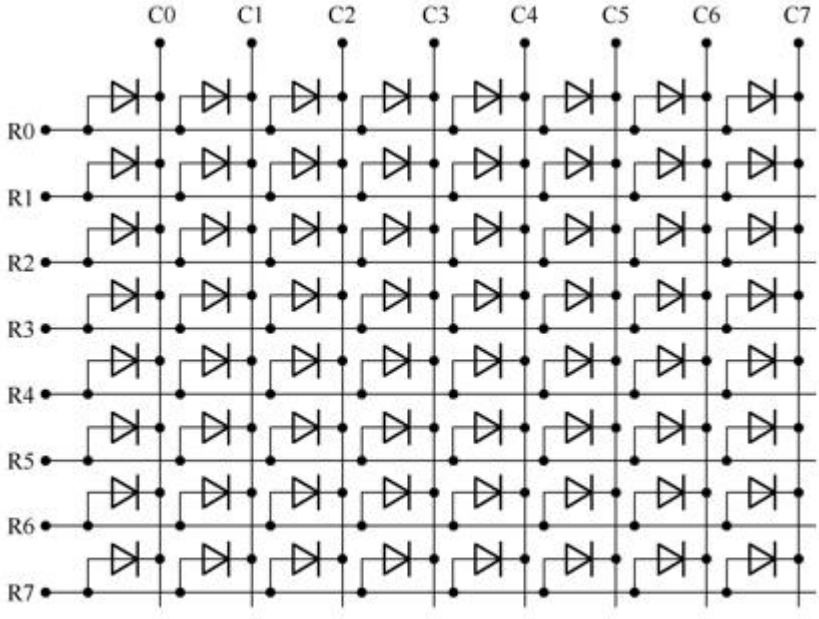


图5-12　8×8LED点阵显示器（共阴极）的结构

显示过程如下：

先给 LED 点阵的第 1 行送高电平（行线高电平有效），同时给所有列线送高电平（列线低电平有效），从而第 1 行发光二极管全灭；

延时一段时间后，再给第 2 行送高电平，同时给所有列线送“1100 0000 0000 1111”，列线为 0 的发光二极管点亮，从而点亮 10 个发光二极管，显示出汉字“子”的第一横；

延时一段时间后，再给第 3 行送高电平，同时加到列线的编码为“1111 1111 1101 1111”，点亮 1 个发光二极管；

……；

延时一段时间后，再给第 16 行送高电平，同时给列线送“1111 1101 1111 1111”，显示出汉字“子”的最下面的一行，点亮 1 个发光二极管。然后再重新循环上述操作，利用人眼视觉暂留效应，一个稳定字符“子” 显示出来，见图 5-13。

5.4.2　控制 16×16 LED 点阵显示屏的案例

单片机控制 16×16 点阵显示屏显示字符案例。

【例 5-8】如图 5-14，利用单片机及 74LS154（4-16 译码器）、74LS07、16×16 LED 点阵显示屏来实现字符显示，编写程序，循环显示字符“电子技术”。

图中 16×16 LED 点阵显示屏 16 行行线 R0~R15 电平，由 P1 口低 4 位经 4-16 译码器 74HC154 的 16 条译码输出线 L0~L15 经驱动后的输出来控制。16 列列线 C0~C15 的电平由 P0 口和 P2 口控制。剩下问题是如何确定显示字符的点阵编码，以及控制好每一屏逐行显示的扫描速度（刷新频率）。

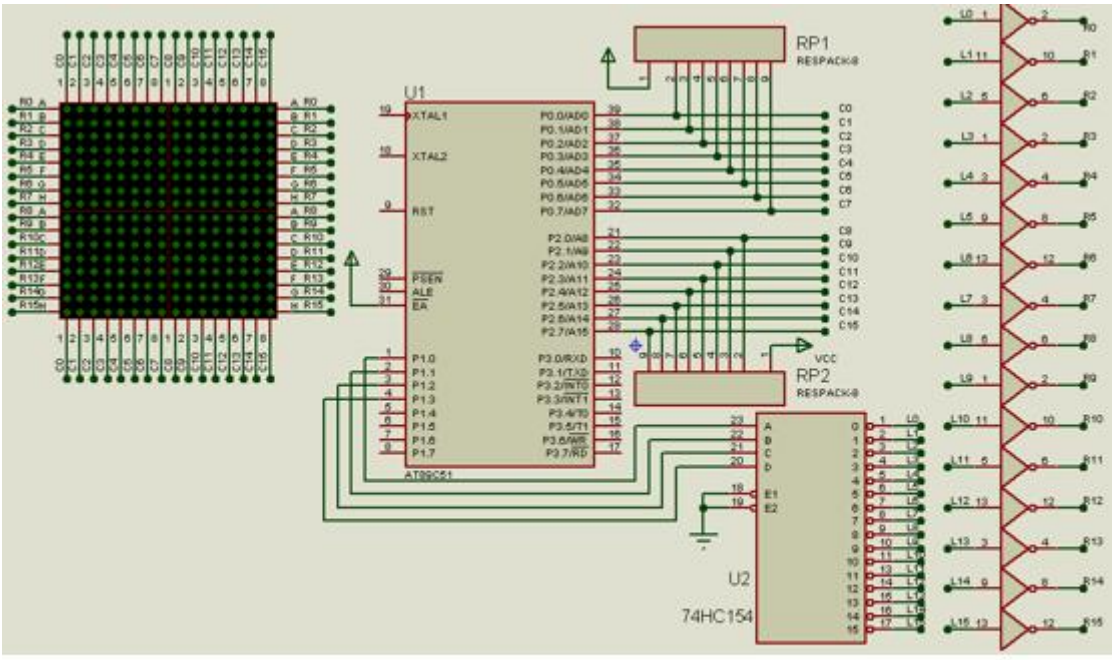


图5-14　控制16×16LED点阵显示器（共阴极）显示字符

参考程序如下：

```
#include<reg51.h>
#define uchar unsigned char
#define uint unsigned int
#define out0 P0
#define out2 P2
#define out1 P1
void delay(uint j)      //延时函数
{
    uchar i=250;
    for(;j>0;j--)
    {
        while(--i);
        i=100;
    }
 }
uchar code string[]= {
//汉字“电” 16×16 点阵列码
0x7F,0xFF,0x7F,0xFF,0x7F,0xFF,0x03,0xE0,0x7B,0xEF,0x7B,0xEF,0x03,0xE0,0x7B,0xEF,0x7B,0xEF,0x7B,0xEF,0x03,0xE0,0x7B,0xEF,0x7F,0xBF,0x7F,0xBF,0xFF,0x00,0xFF,0xFF
```

```
    //汉字“子” 16×16 点阵列码
    0xFF,0xFF,0x03,0xF0,0xFF,0xFB,0xFF,0xFD,0xFF,0xFE,0x7F,0xFF,0x7F,
0xFF,0x7F,0xDF,0x00,0x80,0x7F,0xFF,0x7F,0xFF,0x7F,0xFF,0x7F,0xFF,0x
7F,0xFF,0x5F,0xFF,0xBF,0xFF
    //汉字“技” 16×16 点阵列码
    0xF7,0xFB,0xF7,0xFB,0xF7,0xFB,0x40,0x80,0xF7,0xFB,0xD7,0xFB,0x
67,0xC0,0x73,0xEF,0xF4,0xEE,0xF7,0xF6,0xF7,0xF9,0xF7,0xF9,0xF7,0xF
6,0x77,0x8F,0x95,0xDF,0xFB,0xFF
    //汉字“术”的 16×16 点阵的列码
    0x7F,0xFF,0x7F,0xFB,0x7F,0xF7,0x7F,0xFF,0x00,0x80,0x7F,0xFF,0x3
F,0xFE,0x5F,0xFD,
    0x5F,0xFB,0x6F,0xF7,0x77,0xE7,0x7B,0x8F,0x7C,0xDF,0x7F,0xFF,0x7
F,0xFF,0xFF,0xFF,
     };
    void main()
    {
        uchar i,j,n;
        while(1)
         {
            for(j=0;j<4;j++)          //共显示 4 个汉字
            {
    for(n=0;n<40;n++)           //每个汉字整屏扫描 40 次
    {
            for(i=0;i<16;i++)         //逐行扫描 16 行
            {
            out1=i%16;              //输出行码，
            out0=string[i*2+j*32];       //输出列码到 C0~C7，逐行扫描
            out2=string[i*2+1+j*32]; //输出列码到 C8~C15，逐行扫描
            delay(4);               //显示并延时一段时间
            out0=0xff;              //列线 C0~C7 为高电平，熄灭发光二极管
            out2=0xff;              /列线 C8~C15 为高电平，熄灭发光二极管
             }
          }
     }
        }
    }
```

扫描显示时，单片机通过 P1 口低 4 位经 4-16 译码器 74HC154 的 16 条译码输出线 L0~L15 经驱动后的输出来控制，逐行为高电平，来进行扫描。由 P0 口与 P2 口控制列码的输出，从而显示出某行应点亮的发光二极管。

以显示汉字“子”为例，说明显示过程。由上面程序可看出，汉字“子”的前 3 行发光二级管的列码为“0xFF,0xFF,0x03,0xF0,0xFF,0xFB,……”

第一行列码为：“ 0xff,0xff”，由 P0 口与 P2 口输出，无点亮的发光二极管。第二行列码为：“0x03,0xf0”，通过 P0 口与 P2 口输出后，由图 5-13 看出，0x03 加到列线 C7~ C0 的二进制编码为“0000 0011”，这里要注意加到 8 个发光二极管上的对应位置。

按照图 5-12 和图 5-14 连线关系，加到从左到右发光二极管应为 C0~ C7 的二进制编码为“1100 0000”，即最左边的 2 个发光二极管不亮，其余的 6 个发光二极管点亮。

同理，P2 口输出的 0xF0 加到列线 C15~ C8 的二进制编码为“1111 0000”，即加到 C8~ C15 的二进制编码为“0000 1111”，所以第二行的最右边的 4 个发光二极管不亮，如图 5-13 所示。对应通过 P0 口与 P2 口输出加到第 3 行 16 个发光二极管的列码为“0xFF,0xFB,”，对应于从左到右的 C0~ C15 的二进制编码为“1111 1111 1101 1111”，从而第 3 行左边数第 11 个发光二极管被点亮，其余均熄灭，如图 5-13 所示。其余各行点亮的发光二极管，也是由 16×16 点阵的列码来决定。

作业：

实操练习例题，完成画图，打代码，仿真

第六章　中断系统（12 学时）

本章课时分配　　本章分为 4 讲，共 12 学时,其中 2 讲为实操练习。

6.1~6.9　AT89S51 中断技术概述

教学目标：（教学 12 学时）

1、掌握单片机 AT89S51 中断系统结构

教学重点

1．AT89S51 中断系统结构类型与个数

教学难点

1．AT89S51 中断系统结构类型与个数

素质（思政）内容与要求：

思政融入点：通过讲解中断技术在实际应用中的意义，帮助学生认识到技术对于社会发展的积极作用。

案例：鼓励学生在完成基本功能后，尝试编写一些具有创新性的中断处理逻辑，如根据不同类型的中断事件采取不同的响应措施。

教学手段：讲授

主要内容：

6.1　AT89S51 中断技术概述

中断技术主要用于实时监测与控制，要求单片机能及时地响应中断请求源提出的服务请求，并快速响应与及时处理。

当中断请求源发出中断请求时，如中断请求被允许，单片机暂时中止当前正在执行的主程序，转到中断服务处理程序处理中断服务请求，处理完中断服务请求后，再回到原来被中止的程序之处（断点），继续执行被中断的主程序。

图 6-1 显示单片机对外设中断服务请求整个中断响应和处理过程。

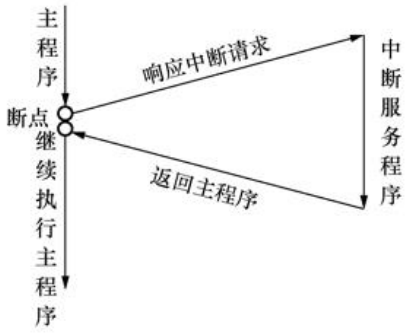


图6-1　中断响应和处理过程

如没有中断系统，单片机大量时间可能会浪费在查询是否有服务请求的定时查询操作上，即不论是否有服务请求，都必须去查询。

采用中断技术完全消除查询方式的等待，大大提高单片机工作效率和实时性。

6.2　AT89S51 中断系统结构

中断系统结构见图 6-2。中断系统有 5 个中断请求源（简称中断源），2 个中断优先级，可实现 2 级中断服务程序嵌套。每一中断源可用软件独立控制为允许中断或关闭中断状态；每一个中断源的优先级均可用软件设置。

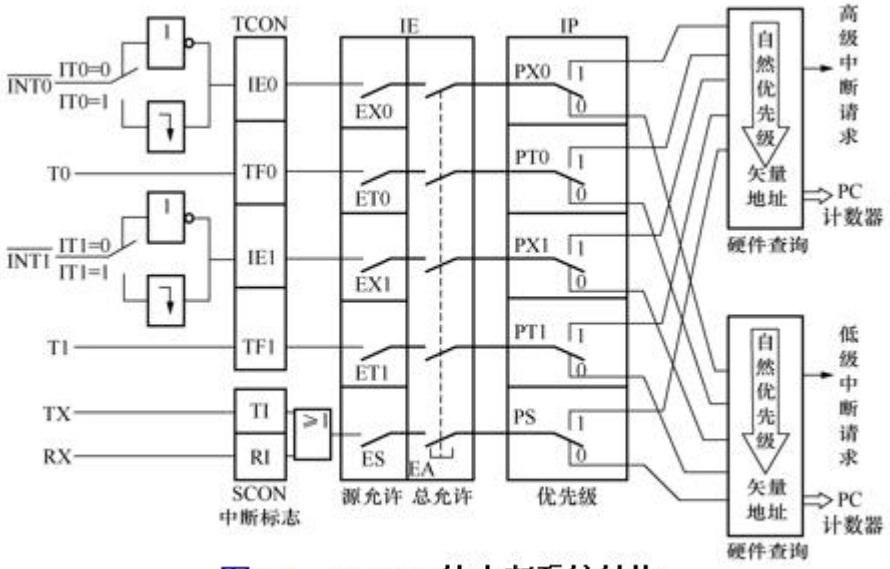


图6-2　AT89S51的中断系统结构

6.2.1　中断请求源

由图 6-2，中断系统共有 5 个中断请求源，它们是：

（1）INT0*—外部中断请求 0，外部中断请求信号（低电平或负跳变有效）由 INT0*引脚输入，中断请求标志为 IE0。

（2）INT1*—外部中断请求 1，外部中断请求信号（低电平或负跳变有效）由 INT1*引脚输入，中断请求标志为 IE1。

（3）定时器/计数器 T0 计数溢出的中断请求，标志为 TF0。

（4）定时器/计数器 T1 计数溢出的中断请求，标志为 TF1。

（5）串行口中断请求，标志为发送中断 TI 或接收中断 RI。

6.2.2 中断请求标志寄存器

5 个中断请求源的中断请求标志分别由特殊功能寄存器 TCON 和 SCON 相应位锁存（见图 6-2）。

1．TCON 寄存器

为定时器/计数器的控制寄存器，字节地址为 88H，可位寻址。既包括定时器/计数器 T0、T1 溢出中断请求标志位 TF0 和 TF1，也包括两个外部中断请求的标志位 IE1 与 IE0，还包括两个外部中断请求源的中断触发方式选择位。TCON 格式见图 6-3。

	D7	D6	D5	D4	D3	D2	D1	D0	
TCON	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	88H
位地址	8FH	—	8DH	—	8BH	8AH	89H	88H	

图6-3 特殊功能寄存器TCON的格式

2．SCON 寄存器

串行口控制寄存器，字节地址为 98H，可位寻址。SCON 的低二位锁存串口的发送中断和接收中断的中断请求标志 TI 和 RI，格式见图 6-4。

	D7	D6	D5	D4	D3	D2	D1	D0	
SCON	—	—	—	—	—	—	TI	RI	98H
位地址	—	—	—	—	—	—	99H	98H	

图6-4 SCON中的中断请求标志位

6.3 中断允许与中断优先级的控制

实现中断允许控制和中断优先级控制分别中断允许寄存器 IE 和中断优先级寄存器 IP 实现。下面介绍两个特殊功能寄存器。

6.3.1 中断允许寄存器 IE

各中断源开放或屏蔽，是由片内中断允许寄存器 IE 控制。IE 字节地址为 A8H，可进行位寻址，格式见图 6-5。

	D7	D6	D5	D4	D3	D2	D1	D0	
IE	EA	—	—	ES	ET1	EX1	ET0	EX0	A8H
位地址	AFH	—	—	ACH	ABH	AAH	A9H	A8H	

图6-5 中断允许寄存器IE的格式

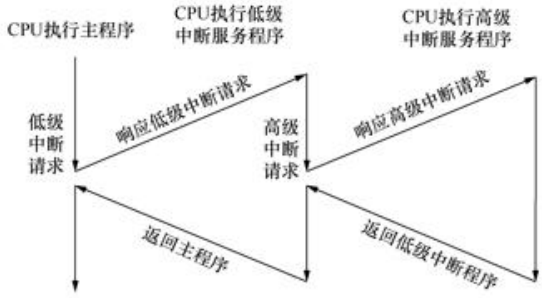


图6-6 两级中断嵌套过程

各中断源的中断优先级关系，可归纳为下面两条基本规则：

（1）低优先级可被高优先级中断，高优先级不能被低优先级中断。

（2）任何一种中断（不管是高级还是低级）一旦得到响应，不会再被它的同级中断源所中断。如果某一中断源被设置为高优先级中断，在执行该中断源的中断服务程序时，则不能被任何其他的中断源的中断请求所中断。

AT89S51 片内有一个中断优先级寄存器 IP，字节地址为 B8H，可位寻址。只要用程序改变其内容，即可进行各中断源中断优先级设置，IP 寄存器格式见图 6-7。

	D7	D6	D5	D4	D3	D2	D1	D0	
IP	—	—	—	PS	PT1	PX1	PT0	PX0	B8H
位地址	—	—	—	BCH	BBH	BAH	B9H	B8H	

图6-7 IP寄存器的格式

表 6-1 同级中断的查询次序

中 断 源	中 断 级 别
外部中断 0 T0 溢出中断 外部中断 1 T1 溢出中断 串行口中断	最高 ↓ 最低

6.4 响应中断请求的条件

一个中断源中断请求被响应，须满足以下必要条件：

（1）总中断允许开关接通，即 IE 寄存器中的中断总允许位 EA=1。

（2）该中断源发出中断请求，即该中断源对应的中断请求标志为“1”。
（3）该中断源的中断允许位=1，即该中断被允许。
（4）无同级或更高级中断正在被服务。

中断响应就是 CPU 对中断源提出的中断请求的接受，当查询到有效的中断请求时，满足上述条件时，紧接着就进行中断响应。

中断响应过程：

首先由硬件自动生成一条长调用指令“LCALL addr16”。即程序存储区中相应的中断入口地址。例如，对于外部中断 1 的响应，硬件自动生成的长调用指令为：

LCALL 0013H

生成 LCALL 指令后，紧接着就由 CPU 执行该指令。首先将程序计数器 PC 内容压入堆栈以保护断点，再将中断入口地址装入 PC，使程序转向响应中断请求的中断入口地址。各中断源服务程序入口地址是固定的，见表 6-2。

其中两个中断入口间只相隔 8 字节，一般情况下难以安放一个完整的中断服务程序。

表6-2　　中断入口地址表

中断源	中断入口地址
外部中断0	0003H
定时器/计数器T0	000BH
外部中断1	0013H
定时器/计数器T1	001BH
串行口中断	0023H

6.8　中断函数

为直接使用 C51 编写中断服务程序，C51 中定义了中断函数。这在第 3 章中已简要介绍。由于 C51 编译器在编译时对声明为中断服务程序的函数自动添加相应现场保护、阻断其他中断、返回时自动恢复现场等处理的程序段，因而在编写中断函数时可不必考虑这些问题，减小编写中断服务程序烦琐程度。

中断服务函数的一般形式为：

函数类型　函数名（形式参数表）interrupt n　using n

关键字 interrupt 后面的 n 是中断号，对于 8051 单片机，n 的取值为 0～4，编译器从 8×n+3 处产生中断向量。AT89S51 中断源对应的中断号和中断向量见表 6-3。

AT89S51 内部 RAM 中可使用 4 个工作寄存器区，每个工作寄存器区包含 8 个工作寄存器（R0～R7）。关键字 using 后面的 n 用来选择 4 个工作寄存器区。using 是一选项，如不选，中断函数中的所有工作寄存器内容将被保存到堆栈中。

表6-3　8051单片机的中断号和中断向量

中断号n	中 断 源	中断向量（8×n+3）
0	外部中断0	0003H
1	定时器0	000BH
2	外部中断1	0013H
3	定时器1	001BH
4	串行口	0023H
其他值	保留	8×n+3

关键字 using 对函数目标代码的影响如下：

在中断函数的入口处将当前工作寄存器区内容保护到堆栈中，函数返回前将被保护的寄存器区内容从堆栈中恢复。使用 using 在函数中确定一个工作寄存器区须十分小心，要保证任何工作寄存器区的切换都只在指定的控制区域中发生，否则将产生不正确的函数结果。

例如，外中断 1（）中断服务函数如下：

```
void int1( ) interrupt 2 using 0//中断号 n=2，选择 0 区工作寄存器区
```

中断调用与标准 C 的函数调用是不一样的，当中断事件发生后，对应的中断函数被自动调用，即没有参数，也没有返回值，会带来如下影响。

（1）编译器会为中断函数自动生成中断向量。

（2）退出中断函数时，所有保存在堆栈中的工作寄存器及特殊功能寄存器被恢复。

（3）在必要时特殊功能寄存器 Acc、B、DPH、DPL 以及 PSW 的内容被保存到堆栈中。

编写中断程序，应遵循以下规则：

（1）中断函数没有返回值，如果定义一个返回值，将会得到不正确结果。建议将中断函数定义为 void 类型，明确说明无返回值。

（2）中断函数不能进行参数传递，如果中断函数中包含任何参数声明都将导致编译出错。

（3）任何情况下都不能直接调用中断函数，否则会产生编译错误。因为中断函数的返回是由汇编语言指令 RETI 完成的。RETI 指令会影响 AT89S51 硬件中断系统内的不可寻址的中断优先级寄存器的状态。如没有实际中断请求情况下，直接调用中断函数，也就不会执行 RETI 指令，其操作结果有可能产生一个致命错误。

（4）如在中断函数中再调用其他函数，则被调用的函数所用的寄存器区必须与中断函数使用的寄存器区不同。

6.9　中断系统应用举例

本节通过几个例程介绍有关中断应用程序的编写。

6.9.1　单一外中断的应用

【例 6-1】 在单片机 P1 口上接有 8 只 LED。在外部中断 0 输入引脚（P3.2）接一只按钮开关 K1。要求将外部中断 0 设置为电平触发。程序启动时，P1 口上的 8 只 LED 全亮。每按一次按钮开关 K1，使引脚接地，产生一个低电平触发的外中断请求，在中断服务程序中，让低 4 位的 LED 与高 4 位的 LED 交替闪烁 5 次。然后从中断返回，控制 8 只 LED 再次全亮。

原理电路及仿真结果见图 6-9。

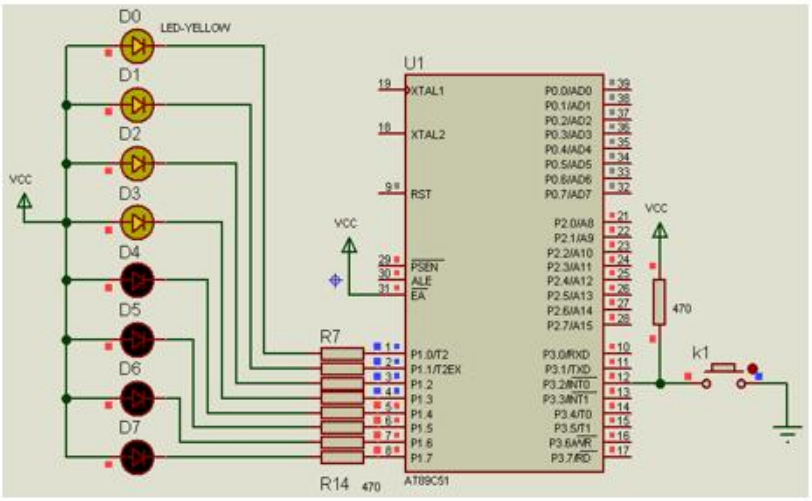


图6-9　利用中断控制8只LED交替闪烁1次的电路

参考程序如下：

```
    #include <reg51.h>
    #define uchar   unsigned char
    void Delay(unsigned int i)   //延时函数 Delay( )，i 形式参数，不能赋初值
    {
        unsigned int j;
        for(;i > 0;i--)
        for(j=0;j<333;j++)          //晶振为 12MHz，j 选择与晶振频率有关
        {;}             //空函数
    }
void   main( )          //主函数
    {
        EA=1;           //总中断允许
        EX0=1;          //允许外部中断 0 中断
        IT0=1;          //选择外部中断 0 为跳沿触发方式
        while(1)        //循环

      { P1=0;}          // P1 口的 8 只 LED 全亮
    }
  void int0( )  interrupt 0   using 0      //外中断 0 的中断服务函数
    {
        uchar   m;
        EX0=0;          //禁止外部中断 0 中断
        for(m=0;m<5;m++)   //交替闪烁 5 次
        {
            P1=0x0f;        //低 4 位 LED 灭，高 4 位 LED 亮
     Delay(400) ;       //延时
        P1=0xf0;        //高 4 位 LED 灭，低 4 位 LED 亮
        Delay(400);         //延时
        EX0=1;          //中断返回前，打开外部中断 0 中断
```

}

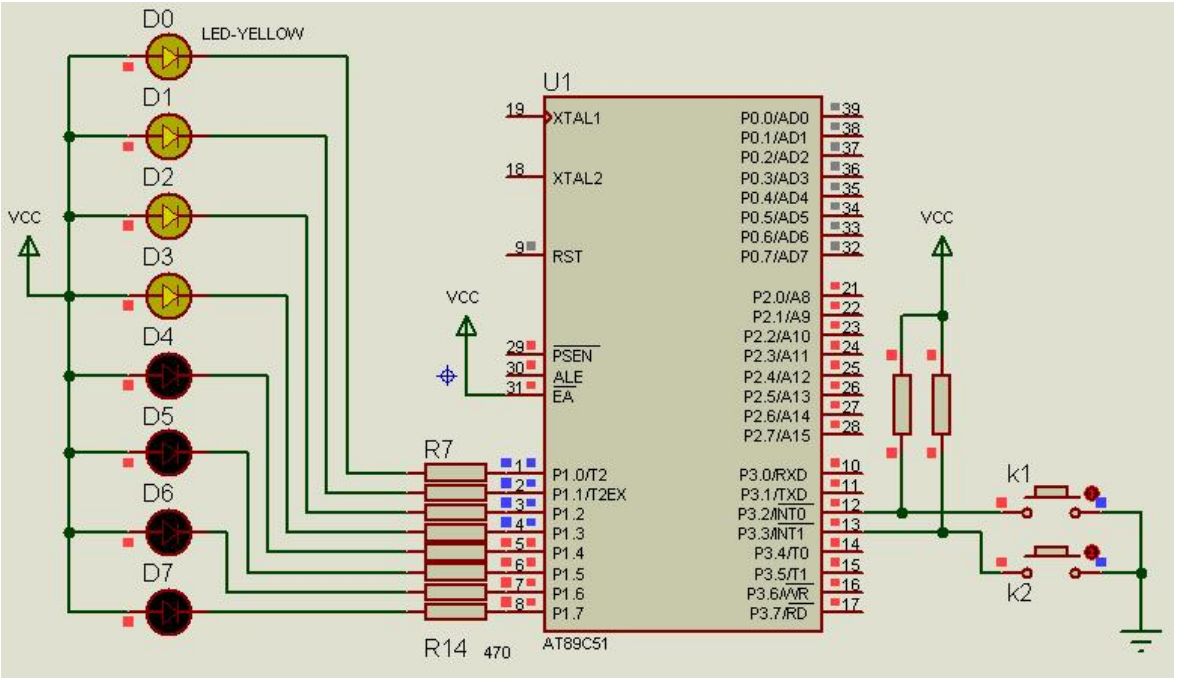
本例程包含两部分，一部分是主程序段，完成中断系统初始化，并把 8 个 LED 全部点亮。另一部分是中断函数部分，控制 4 个 LED 交替闪烁 1 次，然后从中断返回。

作业：

（1）实操练习例题，完成画图，打代码，仿真

（2）作业题：

在单片机 P1 口上接有 8 只 LED。在外部中断 0 输入引脚（P3.2）接有一只按钮开关 K1。在外部中断 1 输入引脚（P3.3）接有一只按钮开关 K2。要求 K1 和 K2 都未按下时，P1 口的 8 只 LED 呈流水灯显示，仅 K1（P3.2）按下再松开时，上下各 4 只 LED 交替闪烁 10 次，然后再回到流水灯显示。如果按下再松开 K2（P3.3）时，P1 口的 8 只 LED 全部闪烁 10 次，然后再回到流水灯显示。设置两个外中断的优先级相同。



两个外中断控制 8 只 LED 显示的电路

第七章　定时器/计数器的工作原理及应用（10 学时）

本章课时分配　　本章分为 4 讲，共 10 学时,其中 2 讲为实操练习。

教学目标：（教学 10 学时）

1、掌握定时器计数器的结构

2、掌握用定时器溢出中断的方法

3、用 C51 编程完成定时器中断

教学重点

1．定时器溢出中断的方法

教学难点

1．定时器溢出中断的方法

素质（思政）内容与要求：

思政融入点：鼓励学生在学习定时器/计数器时，尝试提出新的应用方案或优化现有系统。

案例：鼓励学生在完成基本功能后，尝试编写一些具有创新性的定时提醒逻辑，如根据不同的时间段采取不同的提醒方式。

教学手段：讲授+实训

7.1 定时器/计数器的结构

AT89S51 定时器/计数器结构见图 7-1，定时器/计数器 T0 由特殊功能寄存器 TH0、TL0 构成，T1 由特殊功能寄存器 TH1、TL1 构成。

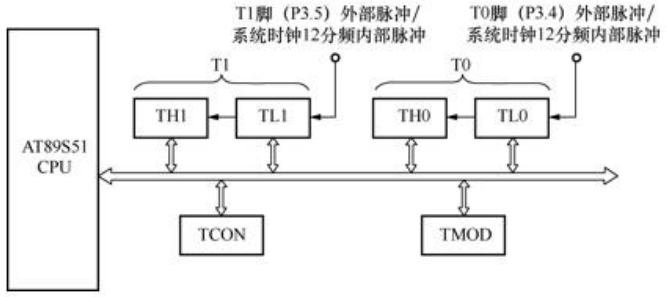


图7-1 定时器/计数器结构框图

T0、T1 都有定时器和计数器两种工作模式，两种模式实质都是对脉冲信号进行计数，只不过计数信号来源不同。

计数器模式是对加在 T0（P3.4）和 T1（P3.5）两个引脚上的外部脉冲进行计数（见图 7-1）；

定时器模式是对系统时钟信号经 12 分频后的内部脉冲信号（机器周期）计数。由于系统时钟频率是定值，可根据计数值计算出定时时间。两个定时器/计数器属于增 1 计数器，即每计一个脉冲，计数器增 1。

T0、T1 具有 4 种工作方式（方式 0、1、2 和 3）。

图 7-1 特殊功能寄存器 TMOD 用于选择定时器/计数器 T0、T1 的工作模式和工作方式。特殊功能寄存器 TCON 用于控制 T0、T1 的启动和停止计数，同时包含了 T0、T1 状态。

计数器起始计数从初值开始。单片机复位时计数器初值为 0，也可给计数器装入 1 个新的初值。

7.1.1 工作方式控制寄存器 TMOD

TMOD 用于选择定时器/计数器的工作模式和工作方式，字节地址为 89H，不能位寻址，格式见图 7-2。

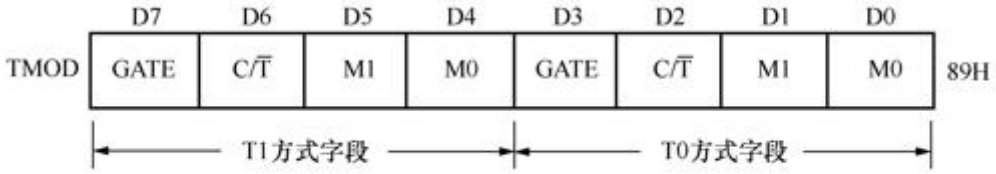


图7-2 寄存器TMOD格式

2）M1、M0—工作方式选择位

M1、M0 4 种编码，对应于 4 种工作方式的选择，见表 7-1。

表7-1 M1、M0工作方式选择

M1 M0	工作方式
0 0	方式0，为13位定时器/计数器
0 1	方式1，为16位定时器/计数器
1 0	方式2，为8位的常数自动重新装载的定时器/计数器
1 1	方式3，仅适用于T0，此时T0分成2个8位计数器，T1停止计数

7.1.2 定时器/计数器控制寄存器 TCON

TCON 字节地址 88H，位地址为 88H～8FH。格式见图 7-3。

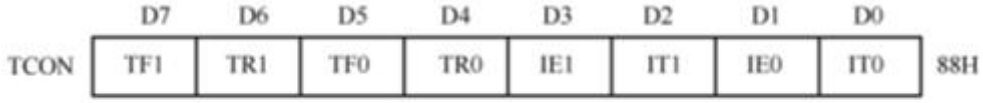


图7-3 TCON格式

7.4 定时器/计数器的编程和应用

4 种工作方式中，方式 0 与方式 1 基本相同，只是计数位数不同。方式 0 为 13 位，方式 1 为 16 位。由于方式 0 是为兼容 MCS-48 而设，计数初值计算复杂，所以在实际应用中，一般不用方式 0，常采用方式 1。

7.4.1 P1 口控制 8 只 LED 每 0.5s 闪亮一次

【例 7-1】在 AT89S51 的 P1 口上接有 8 只 LED，原理电路见图 7-13。采用 T0 方式 1 的定时中断方式，使 P1 口外接的 8 只 LED 每 0.5s 闪亮一次。

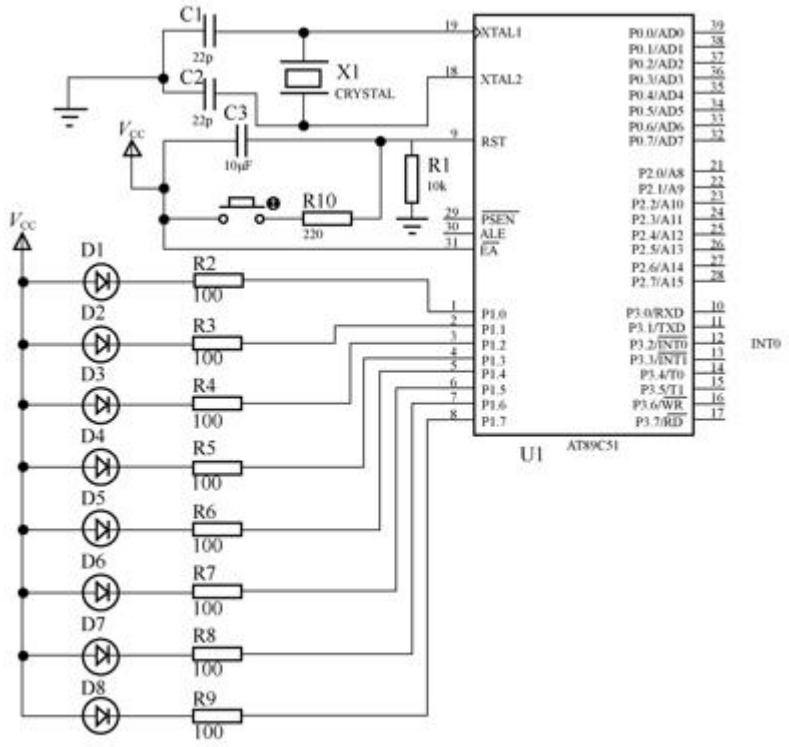


图7-13 方式1定时中断控制LED闪亮

1）设置 TMOD 寄存器

T0 工作在方式 1，应使 TMOD 寄存器的 M1、M0=01；应设置 C/T*=0，为定时器模式；对 T0 的运行控制仅由 TR0 来控制，应使相应的 GATE 位为 0。定时器 T1 不使用，各相关位均设为 0。所以，TMOD 寄存器应初始化为 0x01。

（2）计算定时器 T0 的计数初值

设定时时间 5ms（即 5 000μs），设 T0 计数初值为 *X*，假设晶振的频率为 11.059 2MHz，则定时时间为：

定时时间=(2^{16}−X)×12/晶振频率

则 5 000=(2^{16} −X) ×12/11.059 2

得 X = 60 928

转换成十六进制：0xee00，其中0xee装入TH0，0x00装入TL0。

（3）设置 IE 寄存器

本例由于采用定时器 T0 中断，因此需将 IE 寄存器中的 EA、ET0 位置 1。

（4）启动和停止定时器 T0

将定时器控制寄存器 TCON 中的 TR0=1，则启动定时器 T0；TR0=0，则停止定时器 T0 定时。

参考程序：

```
#include<reg51.h>
char   i=100;
void main ()
```

```
{
    TMOD=0x01;              //定时器 T0 为方式 1
    TH0=0xee;           //设置定时器初值
    TL0=0x00;
    P1=0x00;                //P1 口 8 个 LED 点亮
    EA=1;                   //总中断开
    ET0=1;                  //开 T0 中断
    TR0=1;                  //启动 T0
    while(1);               //循环等待
    {
        ;
    }
}
void timer0() interrupt 1               //T0 中断程序
{
    TH0=0xee;               //重新赋初值
    TL0=0x00;
    i--;                //循环次数减 1
    if(i<=0)
    {
        P1=~P1;             //P1 口按位取反
        i=100;              //重置循环次数
    }
}
```

7.4.2　计数器的应用

【例 7-2】如图 7-14，T1 采用计数模式，方式 1 中断，计数输入引脚 T1（P3.5）上外接按钮开关，作为计数信号输入。按 4 次按钮开关后，P1 口的 8 只 LED 闪烁不停。

（1）设置 TMOD 寄存器

T1 工作在方式 1，应使 TMOD 的 M1、M0=01；设置 C/T*=1，为计数器模式；对 T0 运行控制仅由 TR0 来控制，应使 GATE0=0。定时器 T0 不使用，各相关位均设为 0。所以，TMOD 寄存器应初始化为 0x50。

（2）计算定时器 T1 的计数初值

由于每按 1 次按钮开关，计数 1 次，按 4 次后，P1 口的 8 只 LED 闪烁不停。因此计数器初值为 65 536−4=65 532，将其转换成十六进制后为 0xfffc，所以，TH0=0xff，TL0=0xfc。

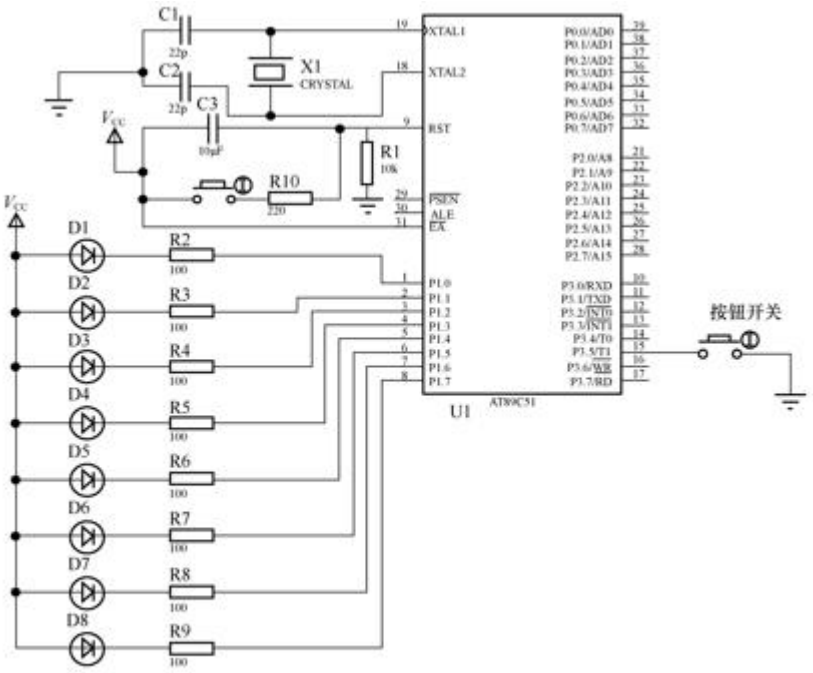


图7-14 由外部计数输入信号控制LED的闪烁

3）设置 IE 寄存器

本例由于采用 T1 中断，因此需将 IE 寄存器的 EA、ET1 位置 1。

（4）启动和停止定时器 T1

将寄存器 TCON 中 TR1=1，则启动 T1 计数；TR1=0，则停止 T1 计数。

参考程序如下：

```
    #include <reg51.h>
    void Delay(unsigned int i)        //定义延时函数 Delay( )，i 是形
                                      //式参数，不能赋初值

{
    unsigned int j;
    for(;i>0;i--)                     //变量 i 由实际参数传入一个值                    //因此 i
不能赋初值
    for(j=0;j<125;j++)
    {;}                               //空函数
 }
void   main( )                        //主函数
 {
    TMOD=0x50;                        //设置定时器 T1 为方式 1 计数
    TH1=0xff;                         //向 TH1 写入初值的高 8 位
    TL1=0xfc;                         //向 TL1 写入初值的低 8 位
    EA=1;                             //总中断允许
    ET1=1;                            //定时器 T1 中断允许
    TR1=1;                            //启动定时器 T1
    while(1) ;                        //无穷循环，等待计数中断
 }
void T1_int(void)   interrupt 3       //T1 中断函数
{
```

```
    for(;;)                 //无限循环
    {
        P1=0xff;            //8 位 LED 全灭
        Delay(500) ;            //延时 500ms
        P1=0;               //8 位 LED 全亮
        Delay(500);             //延时 500ms
    }
}
```

7.4.3 控制 P1.0 产生周期为 2ms 的方波

【例 7-3】假设系统时钟为 12MHz，设计电路并编写程序实现从 P1.0 引脚上输出一个周期为 2ms 的方波，见图 7-15。

要在 P1.0 上产生周期为 2ms 的方波，定时器应产生 1ms 的定时中断，定时时间到则在中断服务程序中对 P1.0 求反。使用定时器 T0，方式 1 定时中断，GATE 不起作用。

本例的原理电路见图 7-16。其中在 P1.0 引脚接有虚拟示波器，用来观察产生的周期 2ms 的方波。

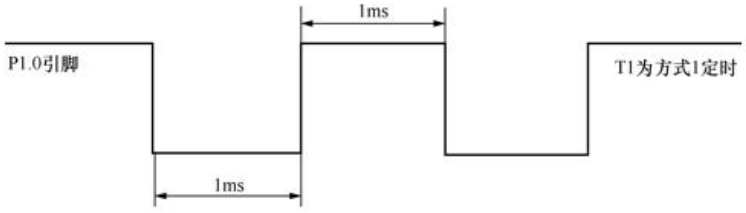


图7-15 定时器控制P1.0输出一个周期2ms方波

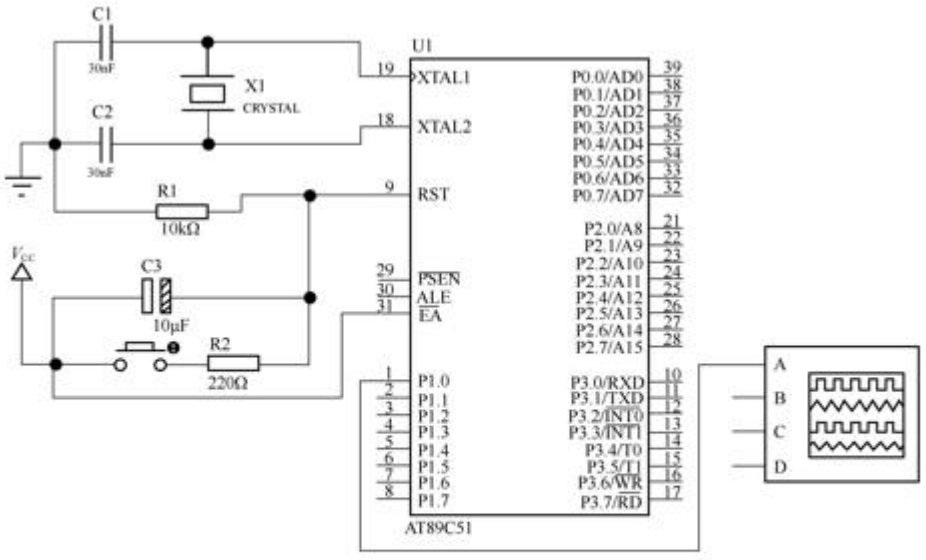


图7-16 定时器控制P1.0输出周期2ms的方波的原理电路

下面来计算 T0 初值 X ：

设 T0 的初值为 X，有

$$(2^{16}-X)\times 1\times 10^{-6}=1\times 10^{-3}$$

即　　$65\ 536-X=1\ 000$

得 X=64 536，化为 16 进制数就是 0xfc18。将高 8 位 0xfc 装入 TH0，低 8 位 0x18 装入 TL0。

参考程序如下：

```
#include <reg51.h>          //头文件 reg51.h
sbit P1_0=P1^0;             //定义特殊功能寄存器 P1 的位变量 P1_0
void main(void)          //主程序
{
    TMOD=0x01;        //设置 T0 为方式 1
    TR0=1;          //接通 T0
    while(1)        //无限循环
    {
    TH0=0xfc;       //置 T0 高 8 位初值
    TL0=0x18;       //置 T0 低 8 位初值
    do{}while(!TF0);          //TF0 为 0 原地循环，为 1 则 T0 溢出，往下执行
    P1_0=!P1_0;       // P1.0 状态求反
    TF0=0;          //TF0 标志清零
    }
}
```

仿真时，右键单击虚拟数字示波器，出现下拉菜单，点击“Digital oscilloscope”选项，就会在数字示波器上显示 P1.0 引脚输出周期为 2ms 方波，如图 7-17 所示。

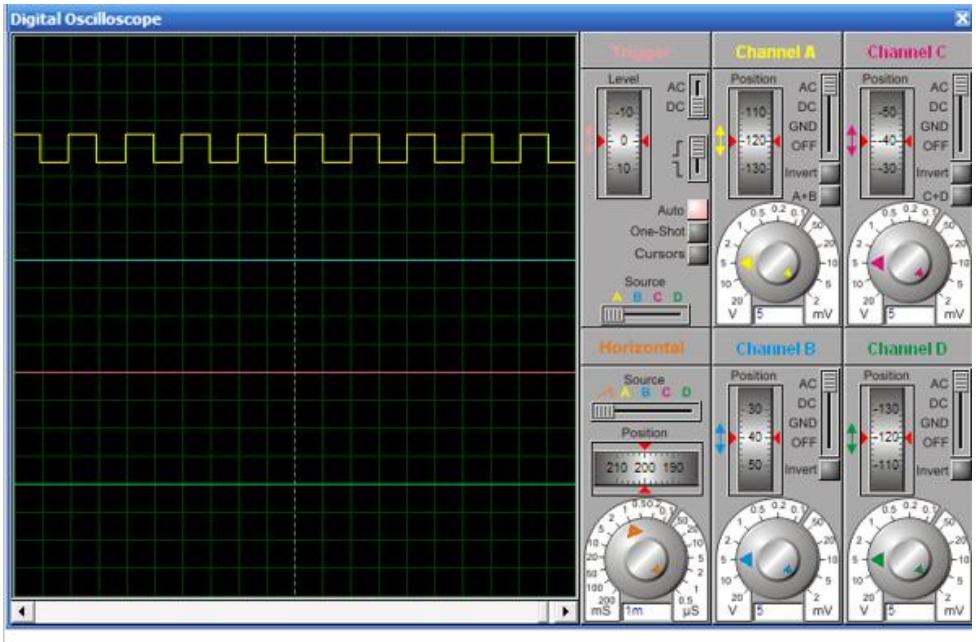


图7-17　虚拟数字示波器显示的2ms的方波波形

作业：

LED 数码管秒表的制作

用 2 位数码管显示计时时间，最小计时单位为“百毫秒”，计时范围 0.1～9.9s。当第 1 次按一下计时功能键时，秒表开始计时并显示；第 2 次按一下计时功能键时，停止计时，将计时的时间值送到数码管显示；如果计时到 9.9s，将重新开始从 0 计时；第 3 次按一下计时功能键，秒表清 0。再次按一下计时功能键，则重复上述计时过程。

原理电路见图