主教材课后习题参考答案

习题 1

1. 你是如何理解嵌入式系统的“嵌入性”的？

答：

嵌入式系统的嵌入性主要体现在把通用计算机系统嵌入到对象体系中，实现对其的智能化控制，其中嵌入一词，即指其软硬件可裁剪性，也表示该系统通常是更大系统中的一个完整的部分。

2．通用计算机系统和嵌入式计算机系统各自的技术要求与技术发展方向有何不同？

答：通用计算机系统的技术要求是高速、海量的数值计算，技术发展方向是总线速度的无限提升，存储容量的无限扩大。嵌入式计算机系统的技术要求是对象的智能化控制能力，技术发展方向是与对象系统密切相关的嵌入性能、控制能力与控制的可靠性。

3．什么是嵌入式系统？从两个方面 说明嵌入式系统的基本概念。

答：

见课本 P2—1.2.1 第三段

4．简述嵌入式系统与嵌入式设备的关系。

答：

嵌入式系统是以应用为中心，以计算机技术为基础，并且软硬件可裁剪，适用于应用系统对功能、可靠性、成本、体积、功耗有严格要求的专用计算机系统。嵌入式设备是指内部有嵌入式系统的产品、设备和装置等，是嵌入式系统所嵌入的宿主系统。

5．为什么说嵌入式系统是专用计算机应用系统？嵌入式系统由哪些部分组成？

答：

见课本 P2—1.2.2

6. 简述 MCU、MPU、DSP 和 SoC 之间的区别和联系。

答：MPU：具有 32 位以上的处理器，性能好，价格也相对较高；

MCU：是最大特点的单片化，体积小，成本和功耗都大为降低，外设较多，适合于控制；

DSP：专用于信号处理方面的处理器，具有很高的编译效率和指令的执行速度；

SoC：在单芯片上集成数字信号处理器、微控制器、存储器、数据转换器、接口电路等功能模块，可以直接实现信号采集、转换、存储、处理等功能。

联系：它们都是从通用计算机发展而来，相对于通用计算机都减小了体积，降低了成本和功耗，提高了专业性和可靠性。

7．嵌入式系统主要有哪些特点？

答：

（1）嵌入式系统通常及其注重成本
（2）嵌入式系统通常对实时性有要求
（3）嵌入式系统一般采用 EOS 或 RTOS
（4）嵌入式系统软件故障造成的后果较通用计算机更为严重
（5）嵌入式系统多为低功耗系统
（6）嵌入式系统经常在极端恶劣的环境下运行
（7）嵌入式系统的系统资源与通用计算机相比是非常少的
（8）嵌入式系统通常在 ROM 中存放所以程序的目标代码
（9）嵌入式系统可采用多种类型的处理器和处理器体系结构
（10）嵌入式系统需要有专用开发工具和方法进行设计
（11）嵌入式系统处理器包含专用调试电路

8．嵌入式操作系统一般如何分类？

答：

嵌入式系统的种类按形态可分为设备级、板级、芯片级，按应用分为工业应用和消费电子。

9. 说明使用嵌入式操作系统的优缺点。

答：

优点：程序设计和扩展容易，不需要大的改动就可以增加新的功能；通过将应用程序分成若干独立的模块，使程序设计过程大为简化；对实时性要求较高的事件都得到了快速、可靠的处理；充分利用了系统资源。

缺点：使用嵌入式操作系统需占用嵌入式处理机的硬件资源和部分内存，另外还需支付操作系统内核费用，不适合低成本的小型项目。

10．简述单片机的发展历史。

答：

见课本 P11—1.5.2

11．你是如何理解嵌入式系统应用的高低端？

答：
见课本 P11—1.5.3

12．简述单片机嵌入式系统的特点。

答：
(1).体积小、价格低、性能强大、速度快、用途广、灵活性强、可靠性高；
(2).存储器 ROM 和 RAM 是有严格分工的；
(3).采用面向控制的指令系统；
(4).输入/输出端口引脚通常设计有多种功能；
(5).品种规格的系列化；
(6).广泛的通用性。

13．简述单片机的技术指标。
答：
位数、存储器、I/O 口、速度、工作电压、功耗、温度、附加功能

14．通过查阅资料，谈谈单片机嵌入式系统产品开发和应用的发展趋势如何。

答：
开放型题目，答案不唯一。

15. 举出几个嵌入式系统应用的例子，通过查资料和独立思考，分析这些嵌入式系统产品主要由哪几部分组成，每个组成部分分别完成什么功能（提示：数码相机、办公类产品、工业控制类产品的例子等）。

答：
开放型题目，答案不唯一。

习题 2

1．将下列十进制数转化成等值的二进制、八进制和十六进制数。要求二进制数保留小数点后的 4 位有效数字。

（1）$(17)_{10}$；　（2）$(127)_{10}$；　（3）$(49)_{10}$；　（4）$(53)_{10}$；　（5）$(0.39)_{10}$；（6）$(25.7)_{10}$；（7）$(7.943)_{10}$；　（8）$(79.43)_{10}$。

解：

十进制	17	127	49	53	0.39	25.7	7.943	79.43
二进制	10001	1111111	110001	110101	0.0110	11001.1011	111.1111	**1001111.0111***
八进制	21	177	61	65	0.3075	31.5463	7.7426	117.3341
十六进制	11	7F	31	35	0.63D7	19.B333	7.F168	4F.6E14

*** 小数点后第 4 位 1 为进位得到，相当于四舍五入。**

2．将下列二进制数转化成等值的十六进制数和十进制数。

（1）$(10010111)_2$；（2）$(1101101)_2$；（3）$(101111)_2$；（4）$(111101)_2$；（5）$(0.10011)_2$；（6）$(0.01011111)_2$；（7）$(11.001)_2$；（8）$(1.1001)_2$。

解：

二进制	10010111	1101101	101111	111101	0.10011	0.01011111	11.001	1.1001
十六进制	97	6D	2F	3D	0.98	0.5F	3.2	1.9
十进制	151	109	47	61	0.59375	0.37109375	3.125	1.5625

3．将下列十进制数转换成 8421BCD 码，误差小于 10^{-3}。

（1）$(2004)_{10}$；（2）$(5308)_{10}$；（3）$(203)_{10}$；（4）$(85)_{10}$；（5）$(65.312)_{10}$；（6）$(3.4146)_{10}$；（7）$(0.8475)_{10}$；（8）$(999.675)_{10}$。

解：

十进制	2004	5308	203	85	65.312	3.4146	0.8475	999.675
8421BCD	0010 0000 0000 0100	1001 0011 0000 1000	0010 0000 0011	1000 0101	01100101.0011 0001 0010	0011.0100 0001 0100 0110	0000.1000 0100 0111 0101	1001 1001 1001.0110 0111 0101

4．写出：

（1）十进制数字 $(4590.38)_{10}$ 的 BCD 码，（2）$(100101010110.0100)_{BCD}$ 对应的十进数。

解：

十进制	4590.38	956.4
BCD	0100 0101 1001 0000.0011 1000	1001 0101 0110.0100

5．请将下列十六进制数转换为 ASCII 码。

（1）F；（2）A；（3）0；（4）7；（5）8；（6）C；（7）3；（8）4。

解：

（查主教材 P13 表 2.2）

（1）	（2）	（3）	（4）	（5）	（6）	（7）	（8）

F	A	0	7	8	C	3	4
70	65	48	55	56	67	51	52

6．写出下列字符串的 ASCII 码（用十六进制表示）。

（1）X = 3+5；　（2）China

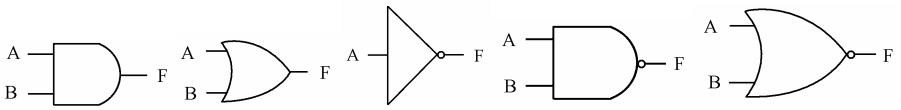
解：

（1）$(88\ 61\ 51\ 43\ 53)_{10}$ ⇔ $(58\ 3D\ 33\ 2B\ 35)_{16}$

（2）$(67\ 104\ 105\ 110\ 97)_{10}$ ⇔ $(43\ 68\ 69\ 6E\ 61)_{16}$

7．画出二输入与、或、非、与非和或非门的电路符号。

解：



8．写出三输入或门的真值表。

解：

除了输入为 0、0、0 的情况输出为 0 外，其余均输出为 1

输　入			输　出
A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

9．六输入或门真值表中有多少种输入逻辑组合?

答：

有 2^6=64 种输入逻辑组合。

10．表 2.26 所示是哪种逻辑门的真值表？写出它的表达式。

答：

异或非（同或）。

F= $\overline{A}\overline{B}+AB$

或

F= $\overline{A \oplus B}$

11．为什么 OC 门在应用时输出端需外接一个上拉负载电阻和电源？不接上拉负载电阻到电源会出现什么现象？

答：

OC 门即集电极开路的 TTL 门电路，由于 OC 门的输出端是开路的，即悬空的，故 OC 门在应用时输出端需要外接一个上拉负载电阻到电源。通过选择合适的电阻和电源电压，既可以保证输出的高、低电平合乎要求，又可使输出端三极管的负载电流不会过大。OC 门的输出端如果不接上拉负载电阻到电源，三极管的集电结则没有反偏，不能实现相应的逻辑关系。

如同 TTL 电路中的 OC 门那样，CMOS 门的输出电路结构也可以做成漏极开路的形式。称为漏极开路的门电路（OD 门），在 CMOS 电路中，这种输出电路结构经常用在输出缓冲/

驱动器当中，或者用于输出电平的变换，以及满足吸收大负载电流的需要。此外也可以用于线与逻辑。同样 OD 门在应用时输出端也需要外接一个上拉负载电阻到电源。

80C51 系列单片机的 P0 口即为 OD 门输出结构。

12．OC 门在单片机系统中主要作用是什么？

答：

（1）OC 门在单片机系统中主要有两个作用：线与和作为驱动器。几个 OC 门的输出端连在一起，输出可以实现与的功能（F=F1F2…Fn）。

（2）OC 门在单片机系统中，还常常作为控制执行机构。利用 OC 门可以控制一些较大电流的执行机构。

13．请列出优先编码器与普通编码器之间的区别。

答：

在普通编码器中，任何时刻只允许输入一个编码信号，否则输出将发生混乱。

在优先编码器电路中，允许同时输入两个以上的信号。不过在设计优先编码器时将所有的输入信号按优先顺序排队，当几个输入信号同时出现时，只对其中优先权最高的一个进行编码。

14．如图 2.35 是用两个 4-1 数据选择器组成的组合逻辑电路，试写出输出 Z 与输入 M、N、P、Q 之间的逻辑函数式。

答：$Y1=[QN\overline{M}+QNM]\cdot P$　　$Y2=[Q\overline{N}\overline{M}+Q\overline{N}M]\cdot\overline{P}$　　Z=Y1+Y2

15．什么是 RAM？什么是 ROM？试区分其性能和用途。

答：

RAM（random access memory）是随机访问存储器，RAM 存储器是断电时信息会丢失的存储器，但是这种存储器可以现场快速地修改信息，所以 RAM 存储器是可读/写存储器，一般都作为数据存储器使用，用来存放现场输入的数据，或者存放可以更改的运行程序和数据。

ROM（Read Only Memory）是只读存储器，ROM 只读存储器的特点是：其内容是预先写入的，而且一旦写入，使用时就只能读出不能改变，掉电时也不会丢失，通常用于存储程序或常数。

16．ROM 与 PROM 有何不同?

答：

ROM 通常指固定 ROM，又称 Mask ROM，需要存储的信息由 ROM 制造厂家写入，信

息存储可靠性最高，当用量很大时，单片成本最低。

PROM 即可编程 PROM，又称 OTP ROM，需要存储的信息由用户使用编程器写入，信息存储可靠性次之，单片成本较低，只能使用一次，目前已较少使用。

17．EEPROM 与 EPROM 之间有什么区别?

答：

EPROM 是 Erasable Programmable Read Only Memory 的缩写，因为其擦除方法是用紫外线照射，所以又称为可用紫外线擦除可多次编程的 ROM——UV-EPROM(Ultraviolet-Erasable Programmable Read Only Memory)。用户可多次改写内容，改写时需要宽度约为 50ms 的高电压编程脉冲，EPROM 芯片外壳上方有窗口，当用紫外线通过这个窗口照射时，写入的信息被擦除。为避免 EPROM 的内容在外来光线照射下慢性自动擦除，通常用一种不透光的标签粘贴在窗口上。

EEPROM 是电可擦写可编程只读存储器（Electrically Erasable Programmable Read-Only Memory），一种掉电后数据不丢失的存储芯片。EEPROM 可以在编程器等专用设备上直接擦除已有信息，重新编程（重写），是用户可更改的只读存储器（ROM），其可通过高于普通电压的作用在线擦除。不像 EPROM 芯片需要紫外线照射擦除，EEPROM 不需从计算机中取出即可修改。在一个 EEPROM 中，当计算机在使用的时候是可频繁地重编程的，EEPROM 的寿命是一个很重要的设计考虑参数。EEPROM 的一种特殊形式是闪存，可以直接使用 CPU 的工作电压来擦写和重编程。

18．试简要叙述 Flash EEPROM 的功能与特点。

答：

（1）低电压在线编程，使用方便，可多次擦写
（2）按块/按扇区擦除，按字节编程
（3）完善的数据保护功能

19.电源供电模块由哪些部分组成？

答：
见课本 P42—图 2.35

20.试分析各种稳压电路的优缺点。

答：
集成线性稳压电路
优点：单片集成稳压电路具有体积小、可靠性高、使用灵活、价格低廉等优点。
缺点：压差太大，郑家集成块的功耗

低压差线性稳压电路相对集成线性稳压电路压差小功耗低的特点

开关型稳压电路

优点：开关型稳压电路的功耗极低；开关管的高频通断特性以及串联滤波电感的使用对来自于电源的高频干扰具有较强大的抑制作用。

21．时钟电路的作用是什么？时钟脉冲频率越高，CPU 的处理速度就越快吗？

答：

时钟电路作用是产生时钟脉冲控制嵌入式处理器的工作

不一定，对于同一系列、相同体系结构的嵌入式处理器而言，时钟频率越高，CPU 的处理速度就越快。

22．内部时钟方式和外部时钟方式有什么特点？

答：

见课本 P45（1.内部时钟电方式，2 外部时钟方式。）

23．同步复位电路中，为何要使用施密特反相器？

答：

在现场干扰大、电压波动大的工作环境，并且，当系统有多个复位端时，使用施密特反相器能保证可靠的同步复位。

24．看门狗复位电路有什么特点？运用目前你所掌握的数字电路知识，设计一个高低复位电平的看门狗定时器。

答：

（1）电源测控：供电电压出现异常时提供预警指示或中断请求信号，方便系统实现异常处理。

（2）数据保护：当电源或系统工作异常时，对数据惊醒必要的保护，如写保护数据数据背反或切换后备电池

（3）看门狗定时器：当系统程序跑飞或死锁是复位。

（4）一定数量的 E^2PROM 串行存储器。

（5）日历时钟。

（6）其他的功能，如温度测控、短路测试等。

电路设计答案不唯一。

习题 3

1．请写出单片机应用系统的一般研制步骤和方法。

答：

虽然单片机的硬件选型不尽相同，软件编写也千差万别，但系统的研制步骤和方法是基本一致的，一般都分为总体设计、硬件电路的构思设计、软件的编制和仿真调试几个阶段。单片机应用系统的研制流程如图 3.1 所示。

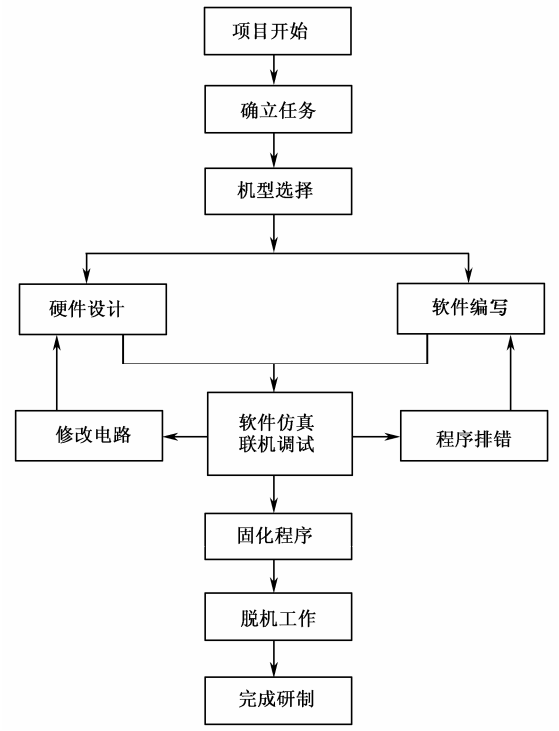


图 3.1　单片机应用系统的研制流程

2．总体设计要考虑哪些主要因素？

答：

① 确立功能特性指标
② 单片机的选型
③ 软件的编写和支持工具

3．简述单片机选型的注意事项。

答：

见课本 P51—3.1.1—2.单片机选型

4．单片机应用系统软、硬件分工要考虑哪些因素？

答：

单片机应用软件的设计与硬件的设计一样重要，没有控制软件的单片机是毫无用处的，它们紧密联系，相辅相成，并且硬件和软件具有一定的互换性，在应用系统中，有些功能既可以用硬件实现，也可以用软件完成。多利用硬件，可以提高研制速度、减少编写软件的工作量、争取时间、争取商机，但这样会增加产品的单位成本，对于以价格为竞争手段的产品不宜采用。相反，以软件代替硬件完成一些功能，最直观的优点是可以降低成本，提高可靠性，增加技术难度而给仿制者增加仿制难度；缺点是同时也增加了系统软件的复杂性，软件的编写工作量大，研制周期可能会加长，同时系统运行的速度可能也会降低等。因此在总体考虑时，必须综合分析以上因素，合理地制定某些功能硬件和软件的比例。

5．单片机应用系统软、硬件开发工具有哪些？

答：

单片机系统开发环境硬件由PC、单片机仿真器、用户目标系统、编程器和数条连接电缆组成。软件由 PC 上的单片机集成开发环境软件和编程器软件构成，前者为单片机仿真器随机软件，后者为编程器随机软件。

6．单片机仿真器的作用是什么？选择一个好的仿真器有哪些要求。

答：

单片机仿真器也称单片机开发系统。PC 通过仿真器和目标系统建立起一种透明的联系，程序员可以观察到程序的运行（实际上程序在仿真器中运行）和 CPU 内部的全部资源情况。也就是说，在开发环境中用户目标系统中的程序存储器是闲置的。我们调试的是仿真器中的程序，仿真器中的程序运行完全受仿真器的监控程序控制。仿真器的监控程序相当于 PC 的操作系统，该监控程序与 PC 上运行的集成开发环境相配合，使得我们可以修改和调试程序，并能观察程序的运行情况。

选择仿真器要求如下：

- 全地址空间的仿真；
- 不占用任何用户目标系统的资源；
- 必须实现硬断点，并且具有灵活的断点管理功能；
- 硬件实现单步执行功能；
- 可跟踪用户程序执行；
- 可观察用户程序执行过程中的变量和表达式；
- 可中止用户程序的运行或用户程序复位；
- 系统硬件电路的诊断与检查；

- 支持汇编和高级语言源程序级调试。

7．什么是 JTAG？为什么使用 JTAG 接口开发单片机，仿真更加贴近实际目标系统？

答：

JTAG（Joint Test Action Group，联合测试行动小组）是一种国际标准测试协议（IEEE 1149.1 兼容），主要用于芯片内部测试。标准的 JTAG 接口有 4 线：TMS、TCK、TDI、TDO，分别为模式选择、时钟、数据输入和数据输出线。JTAG 接口还常用于实现 ISP（In-System Programmable，在系统编程），对单片机内部的 Flash EEPROM 等器件进行编程。

在 JTAG 单片机仿真开发环境中，JTAG 适配器提供了计算机通信口到单片机 JTAG 接口的透明转换，并且不出借 CPU 和程序存储器给应用系统，使得仿真更加贴近实际目标系统。单片机内部已集成了基于 JTAG 的协议调试和下载程序。

8．请解释 ISP 和 IAP？具有 ISP 和 IAP 功能的单片机有什么好处？

答：

随着单片机技术的发展，出现了可以在线编程的单片机。这种在线编程目前有两种实现方法：在系统编程（ISP）和在应用编程（IAP）。ISP 一般通过单片机专用的串行编程接口对单片机内部的 Flash 存储器进行编程，而 IAP 技术是从结构上将 Flash 存储器映射为两个存储体，当运行一个存储体上的用户程序时，可对另一个存储体重新编程，之后将控制从一个存储体转向另一个。

利用 ISP 和 IAP，不需要编程器就可以进行单片机的实验和开发，单片机芯片可以直接焊接到电路板上，调试结束即为成品，甚至可以远程在线升级或改变单片机中的程序。

9．单片机系统的编程语言有哪几种？单片机的 C 语言有哪些优越性？

答：

单片机的高级语言包括：BASIC 语言、PL/M 语言和 C 语言。BASIC 语言主要应用在 MCS-51 系列单片机上，使用效果不是很理想，现在已经不再使用。PL/M 语言对硬件的控制能力和代码效率都很好，但局限于 Intel 公司的单片机系列，可移植性差。目前流行的单片机编程语言为 C 语言。

下面结合 80C51 介绍单片机 C 语言的优越性：

• 不懂得单片机的指令集，也能够编写完美的单片机程序；

• 无须懂得单片机的具体硬件，也能够编出符合硬件实际的专业水平的程序；

• 不同函数的数据实行覆盖，有效利用片上有限的 RAM 空间；

• 程序具有坚固性：数据被破坏是导致程序运行异常的重要因素。C 语言对数据进行了许多专业性的处理，避免了运行中间非异步的破坏；

• C 语言提供复杂的数据类型（数组、结构、联合、枚举、指针等），极大地增强了程序

处理能力和灵活性；

•提供 auto、static、const 等存储类型和专门针对 8051 单片机的 data、idata、pdata、xdata、code 等存储类型，自动为变量合理地分配地址；

• 提供 small、compact、large 等编译模式，以适应片上存储器的大小；

• 中断服务程序的现场保护和恢复，中断向量表的填写，是直接与单片机相关的，都由 C 编译器代办；

• 提供常用的标准函数库，以供用户直接使用；

• 头文件中定义宏、说明复杂数据类型和函数原型，有利于程序的移植和支持单片机的系列化产品的开发；

• 有严格的句法检查，错误很少，可容易地在高级语言的水平上迅速地被排掉；

•可方便地接受多种实用程序的服务：如片上资源的初始化有专门的实用程序自动生成；再如，有实时多任务操作系统可调度多道任务，简化用户编程，提高运行的安全性等等。

10.简述 Keil uVision2 集成开发环境的特点。

答：

uVision2 IDE 基于 Windows 的开发平台，包含一个高效的编辑器、一个项目管理器和一个 MAKE 工具。uVision2 IDE 支持所有的 Keil C51 工具，包括 C 语言编译器、宏汇编器、连接/定位器、目标代码到 HEX 的转换器。

uVision2 IDE 内嵌有多种符合当前工业标准的开发工具，可以完成工程建立、管理、编译连接、目标代码的生成、软件仿真、硬件仿真等完整的开发流程。尤其 C 语言编译工具在产生代码的准确性和效率方面达到了较高的水平，而且可以附加灵活的控制选项，在开发大型项目时非常理想。它的主要特性如下。

1） 集成开发环境
2） C51 编译器和 A51 汇编器
3） LIB51 库管理器
4） BL51 链接器/定位器
5） uVision2 软件调试器
6） uVision2 IDE 硬件调试器
7） RTX-51 实时操作系统

11. 简述 Keil uVision2 编译系统的存储模式。

答：

有三种 Memory Model

Small:变量存储在内部的 RAM 里。

Compact:变量存储在外部 RAM 里。使用 8 位间接寻址。

Large：变量存储在外部 RAM 里，使用 16 位间接寻址

一般使用 Small 来存储变量，此时单片机优先将变量存储在内部 RAM 里，如果内部 RAM 空间不够，才会存到外部 RAM 中。Compact 的方式要通知程序来指定页的高位地址。

Compace 模式适用于比较少的外部 RAM 的情况。Large 模式是指变量会优先分配到外部 RAM 里。

3 种存储方式 都支持内部 256B 和 64KB 的 RAM。因为变量存储到内部里，运算速度比存储在外部 RAM 要快得多。大部分的应用都选择 Small 模式。

12. ProteusVSM 都提供了哪些信号源和测试仪器？

答：

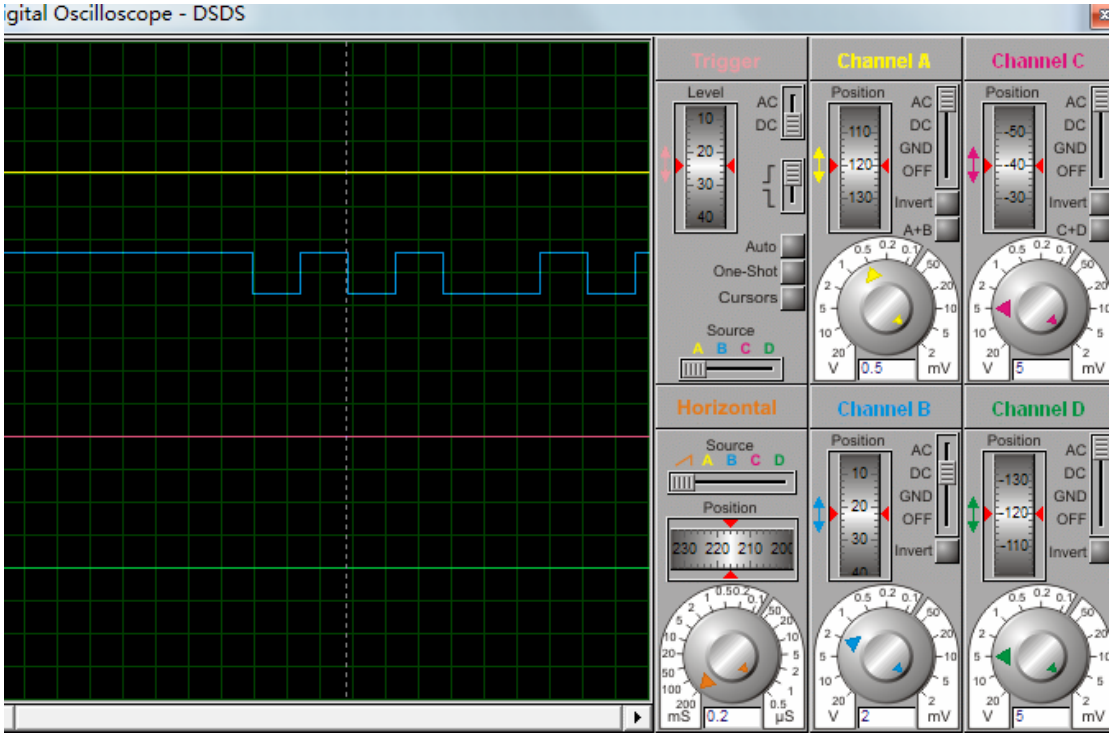
Proteus VSM 包含了大量的虚拟仪器，包括示波器、逻辑分析仪函数发生器、数字信号图案发生器、时钟计数器、虚拟终端及简单的电压表和电流表。

13. Proteus VSM 中的示波器可以同时测量几路信号？测量单片机引脚输出的周期 1kHz 的方波信号，示波器应如何设置?

答：

示波器可同时测量 4 路信号。

先点击虚拟仪器，选择 OSCILLOSCOPE 拉出示波器，连接对应的单片机引脚。双击示波器进行设置界面，选择对应的输入端如将单片机引脚连至 C，则选择 Channel C，然后在 Channel 选择相应的电压范围和时间间隙，其实最方便是中点击 Auto 按钮，具体操作与现实中的数字示波器相差无几。



14．在 Proteus ISIS 环境中使用 AT89C52 设计一个“走马灯”电路，并编写 C51 程序，然后在 μVision3 环境下编译调试。要求实现 Proteus VSM 与 μVision3 的联调。（答案不惟一！）

答：

Proteus 图如下

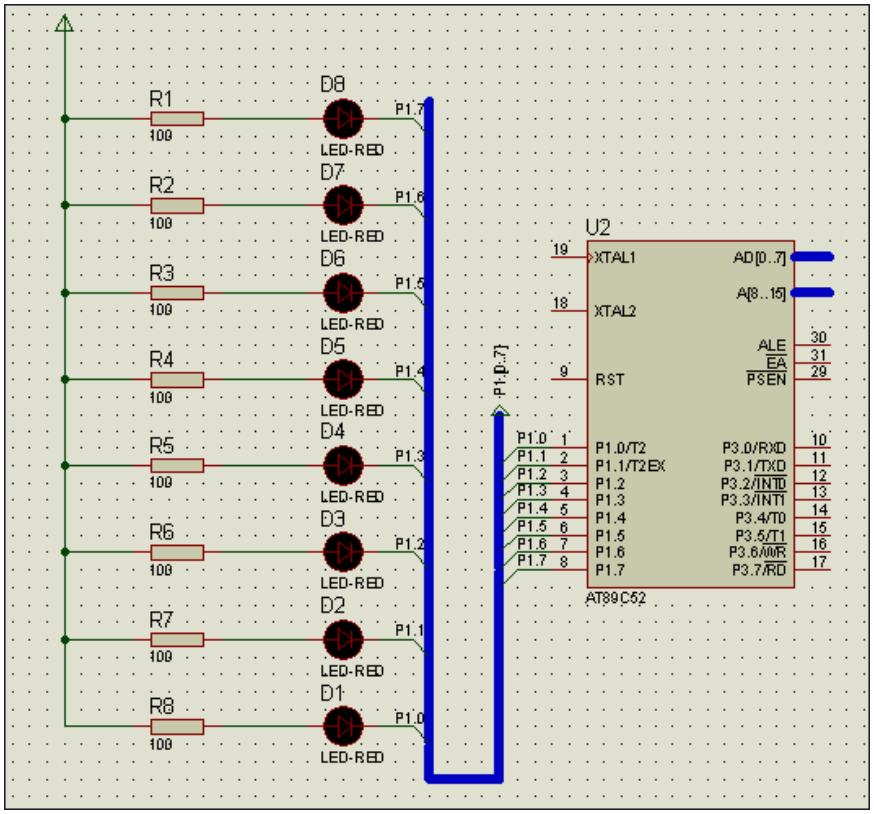


图 3-2 “走马灯”电路

参考程序如下：

```
#include <REG52.H>
#include <INTRINS.H>
#define LED_PORT P1

void time(unsigned int ucMs);/* 延时单位：毫秒 */
void main(void)
{
     LED_PORT= 0xfe;
     while(1)
     {
          LED_PORT = _crol_(LED_PORT,1);
          time(500);
```

```
    }
}

/*************************************************************
*  函数说明：延时 5us，晶振改变时只用改变这一个函数!
    1、对于 11.0592M 晶振而言，需要 2 个_nop_();
    2、对于 22.1184M 晶振而言，需要 4 个_nop_();
*  入口参数：无
*  返回：    无
*  创建日期：20010623
*  作者： 张齐
*************************************************************/
void delay_5us(void)//延时 5us，晶振改变时只用改变这一个函数!
{
    _nop_();
    _nop_();
    //_nop_();
    //_nop_();
}
/************ delay_50us ***************/
void delay_50us(void)//延时 50us
{
    unsigned char i;
    for(i=0;i<4;i++)
    {
        delay_5us();
    }
}
/******** 延时 100us ******************/
void delay_100us(void)//延时 100us
{
    delay_50us();
    delay_50us();
}

/*********** 延时单位：ms ******************/
void time(unsigned int ucMs)//延时单位：ms
{
    unsigned char j;
    while(ucMs>0){
        for(j=0;j<10;j++) delay_100us();
        ucMs--;
    }
```

```
}
```

习题 4

1．结合 MCS-51 系列单片机功能框图阐明其大致组成。

答：

MCS-51 系列单片机内部组成如图所示。

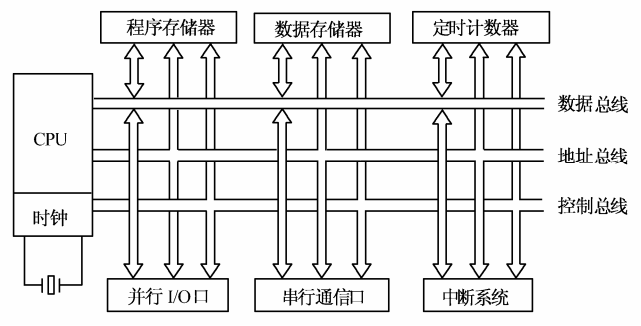


图 4.1　MCS-51 系列单片机内部组成

主要有 8031、8051、875l 三种机型，基于 HMOS 工艺，它们的指令系统与芯片引脚完全兼容，只是片内程序存储器（ROM, Read Only Memory）有所不同。

51 子系列的主要功能为：

- 8 位 CPU；
- 片内带振荡器及时钟电路；
- 128B 片内数据存储器；
- 4KB 片内程序存储器（8031/80C31 无）；
- 程序存储器的寻址范围为 64KB；
- 片外数据存储器的寻址范围为 64KB；
- 21B 特殊功能寄存器；
- 4×8 根 I/O 线；
- 1 个全双工串行 I/O 接口，可多机通信；
- 两个 16 位定时器/计数器；
- 中断系统有 5 个中断源，可编程为两个优先级；
- 111 条指令，含乘法指令和除法指令；
- 布尔处理器；
- 使用单＋5V 电源。

2．综述 80C51 系列单片机各引脚的作用。

答：

80C51 有 4 个 8 位并行 I/O 口,共 32 条端线： P0、P1、P2 和 P3 口。每一个 I/O 口都能

用作输入或输出。

用作输入时，均须先写入“1”；用作输出时，P0 口应外接上拉电阻。

P0 口的负载能力为 8 个 LSTTL 门电路； P1～P3 口的负载能力为 4 个 LSTTL 门电路。

在并行扩展外存储器或 I/O 口情况下：

P0 口用于低 8 位地址总线和数据总线(分时传送)

P2 口用于高 8 位地址总线，

P3 口常用于第二功能，

用户能使用的 I/O 口只有 P1 口和未用作第二功能的部分 P3 口端线。

（详细见主教材 P82~P85）

3．80C51 单片机内部包含哪些主要逻辑功能部件？各有什么主要功能？

答：

80C51 单片机内部包含含布尔（位）处理器的中央处理器、数据存储器和程序、并行输入/输出端口、中断系统、定时器/计数器，串行口、时钟电路、复位电路。

（详细见主教材 4.3 节 P85）

4．什么是 ALU？简述 MCS-51 系列单片机 ALU 的功能与特点。

答：

ALU 是用于对数据进行算术运算和逻辑操作的执行部件，由加法器和其他逻辑电路（移位电路和判断电路等）组成。在控制信号的作用下，它能完成算术加、减、乘、除和逻辑与、或、异或等运算以及循环移位操作、位操作等功能。

5．如何认识：80C51 存储器空间在物理结构上可划分为 4 个空间，而在逻辑上又可划分为 3 个空间？

答：

而 80C51 在物理结构上有 4 个存储空间：片内程序存储器、片外程序存储器、片内数据存储器和片外数据存储器。但在逻辑上，即从用户使用的角度上，80C51 有三个存储空间：片内外统一编址的 64KB 程序存储器地址空间（用 16 位地址）、256 B 片内数据存储器的地址空间（用 8 位地址）及 64KB 片外数据存储器地址空间（用 16 位地址）。在访问三个不同的逻辑空间时，应采用不同形式的指令（见指令系统），以产生不同的存储空间的选通信号。

6．什么是指令？什么是程序？简述程序在计算机中的执行过程。

答：

指令由操作码和操作数构成，分别表示何种操作和操作数的存储地址；

而程序则是：程序是可以连续执行,并能够完成一定任务的一条条指令的集合。

程序执行是由控制器控制的，控制器是 CPU 的大脑中枢，它包括定时控制逻辑、指令寄存器 IR、数据指针 DPTR 及程序计数器 PC、堆栈指针 SP、地址寄存器、地址缓冲器等。它的功能是对程序的逐条指令进行译码，并通过定时和控制电路在规定的时刻发出各种操作所需的内部和外部控制信号，协调各部分的工作，完成指令规定的操作。

7.什么是堆栈？堆栈有何作用？在程序设计时,有时为什么要对堆栈指针 SP 重新赋值？如果 CPU 在操作中要使用两组工作寄存器，你认为 SP 的初值应为多大？

答：

堆栈是个特殊的存储区，主要功能是暂时存放数据和地址，通常用来保护断点和现场。它的特点是按照先进后出的原则存取数据，这里的进与出是指进栈与出栈操作。

80C51 片内 RAM 的部分单元可以用做堆栈。有一个 8 位的堆栈指针寄存器 SP，专用于指出当前堆栈顶部是片内 RAM 的哪一个单元。80C51 单片机系统复位后 SP 的初值为 07H，也就是将从内部 RAM 的 08H 单元开始堆放信息。但是，80C51 系列的栈区不是固定的，只要通过软件改变 SP 寄存器的值便可更动栈区。为了避开工作寄存器区和位寻址区，SP 的初值可置为 2FH 或更大的地址值。

如果 CPU 在操作中要使用两组工作寄存器，如果不使用位变量，SP 的初值至少应为 0FH 或更大的值；如果使用位变量，SP 的初值至少应为 2FH 或更大的值；Keil C51 编译器会自动计算 SP 的初始设定值，无需编程者关心。

8．程序状态寄存器 PSW 的作用是什么？常用状态标识有哪几位？作用是什么？

答：

PSW 是 8 位寄存器，用做程序运行状态的标识。

表 4.7　PSW 寄存器各位名称及地址

地址	D7H	D6H	D5H	D4H	D3H	D2H	D1H	D0H
名称	C	AC	F0	RS1	RS0	OV	F1	P

当 CPU 进行各种逻辑操作或算术运算时，为反映操作或运算结果的状态，把相应的标识位置位或清 0。这些标识的状态，可由专门的指令来测试，也可通过指令读出。它为计算机确定程序的下一步运行方向提供依据。PSW 寄存器中各位的名称及地址如表 4.7 所示，下面说明各标识位的作用。

- P：奇偶标识。该位始终跟踪累加器 A 的内容的奇偶性。如果有奇数个 1，则置 P 为 1，否则清 0。在 80C5l 的指令系统中，凡是改变累加器 A 中内容的指令均影响奇偶标识位 P。
- F1：用户标识。由用户置位或复位。
- OV：溢出标识。有符号数运算时，如果发生溢出，OV 置 1，否则清 0。对于 1B 的

有符号数，如果用最高位表示正、负号，则只有 7 位有效位，能表示-128～+127 之间的数。如果运算结果超出了这个数值范围，就会发生溢出，此时，OV=1，否则 OV=0。在乘法运算中，OV=1 表示乘积超过 255；在除法运算中，OV=1 表示除数为 0。

- RS0、RSl：工作寄存器组选择位。用于选择指令当前工作的寄存器组。由用户用软件改变 RS0 和 RSl 的组合，以切换当前选用的工作寄存器组，单片机在复位后，RS0=RSl=0，CPU 自然选中第 0 组为当前工作寄存器组。根据需要，用户可利用传送指令或位操作指令来改变其状态，这样的设置为程序中快速保护现场提供了方便。
- F0：用户标识位，同 F1。
- AC：半进位标识。当进行加法（或减法）运算时，如果低半字节（位 3）向高半字节（位 4）有进位（或借位），AC 置 1，否则清 0。AC 也可用于 BCD 码调整时的判别位。
- CY：进位标识。在进行加法（或减法）运算时，如果操作结果最高位（位 7）有进位，CY 置 1，否则清 0。在进行位操作时，CY 又作为位操作累加器 C。

9．在 80C51 扩展系统中，片外程序存储器和片外数据存储器共处同一地址空间为什么不会发生总线冲突?

答：

在 80C51 扩展系统中，片外程序存储器和片外数据存储器虽然共处同一地址空间，但是在物理上是两个独立的存储空间，这两个空间都使用相同的 16 位地址线和 8 位数据线，分别为两个 64KB 的寻址空间，它们的选通控制信号不同。程序存储器使用 $\overline{\text{PSEN}}$ 作为取指令控制信号，数据存储器使用 $\overline{\text{WR}}$ 、$\overline{\text{RD}}$ 作为存取数据控制信号。所以不会发生总线冲突。

从指令周期角度来看，取指令周期访问片外程序存储器，$\overline{\text{PSEN}}$ 有效，执行指令周期则存取数据，$\overline{\text{WR}}$ 、$\overline{\text{RD}}$ 信号有效，访问的是片外数据存储器。

10．综述 P0、P1、P2 和 P3 口各有哪几种功能？

答：

参见主教材 P96~P100

11．阐明“准双向口”这一名词之所以要加“准”字的理由。

答：

① 80C51 的 32 条 I/O 线隶属于 4 个 8 位双向端口，每个端口均由锁存器（即特殊功能寄存器 P0～P3）、输出驱动器和输入缓冲器组成；

② P1、P2 和 P3 口均有内部上拉电阻，当它们用做通用 I/O 时，在读引脚状态时，各口对应的锁存器必须置 1，所以为准双向口；

③ P0口内部无上拉电阻，作为I/O口时，必须外部上拉电阻到电源。在读引脚状态时，各口对应的锁存器必须置1，所以为准双向口；

80C51的32条I/O在读引脚状态时，各口对应的锁存器必须置1，即先写1，保证锁存器的输出为1，然后再读引脚，方可读到正确的引脚状态。所以为准双向口。

12．何谓时钟周期、机器周期、指令周期？80C51的时钟周期、机器周期、指令周期是如何分配的？当振荡频率为8MHz时，一个单片机机器周期为多少微秒？

答：

为了便于对CPU时序进行分析，人们按指令的执行过程规定了几种周期，即时钟周期、状态周期、机器周期和指令周期，也称为时序定时单位，下面分别予以说明。

时钟周期也称为振荡周期，定义为时钟脉冲频率（fosc）的倒数，是计算机中最基本、最小的时间单位。

时钟周期经2分频后成为内部的时钟信号，用做单片机内部各功能部件按序协调工作的控制信号，称为状态周期，用S表示。这样一个状态周期就有两个时钟周期，前半状态周期相应的时钟周期定义为P1，后半周期对应的节拍定义为P2。

完成一个基本操作所需要的时间称为机器周期，也称CPU周期。80C51有固定的机器周期，规定一个机器周期有6个状态，分别表示为S1～S6，而一个状态包含两个时钟周期，那么一个机器周期就有12个时钟周期。

所以当振荡频率为8MHz时，

机器周期为 12×1/8MHZ=12×0.125μs=1.5μs

13．复位的作用是什么？有几种复位方法？复位后单片机的状态如何？

答：

复位是单片机的初始化操作。单片机系统在上电启动运行时，都需要先复位，其作用是使CPU和系统中其他部件都处于一个确定的初始状态，并从这个状态开始工作。

单片机的外部复位电路有上电自动复位和按键手动复位两种。

当80C51通电，时钟电路开始工作，在80C51单片机的RST（DIP40封装第9脚）引脚加上大于24个时钟周期以上的正脉冲，80C51单片机系统即初始复位。初始化后，程序计数器PC指向0000H，P0～P3输出口全部为高电平，堆栈指针写入07H，其他专用寄存器被清0。RST由高电平下降为低电平后，系统从0000H地址开始执行程序。

14．简述单片机的掉电保护和低功耗模式的区别。

答：

单片机具有一般的程序执行方式外，还具有两种低功耗运行方式：待机（或称空闲）方式和掉电（或称停机）方式，所以掉电保护是低功耗模式的一种。

PCON寄存器的PD位控制单片机进入掉电方式。当CPU执行一条置PCON.1位（PD）为1的指令后：

```
ORL PCON,#02H
```

单片机就进入掉电方式。在这种方式下，片内振荡器被封锁，一切功能都停止，只有片内 RAM 的 00H～7FH 单元的内容被保留，端口的输出状态值都保存在对应的 SFR 中，ALE 和 $\overline{\text{PSEN}}$ 都为低电平。

退出掉电方式的唯一方法是硬件复位，硬件复位 10 ms 即能使单片机退出掉电方式。复位后将所有的特殊功能寄存器的内容重新初始化，但内部 RAM 区的数据不变。

15. 何谓单片机最小系统？请分别画出由 80C32 单片机和 89C52 单片机组成的最小系统。

答：

单片机最小系统就是能使单片机工作的最少的器件构成的系统，是大多数控制系统必不可少的关键部分。80C32 单片机和 89C52 单片机组成的最小系统如图 4.2 和图 4.3 所示。

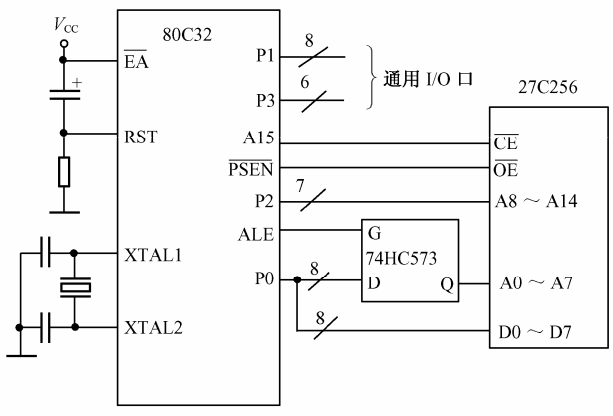


图 4.2　80C32 单片机单片机组成的最小系统

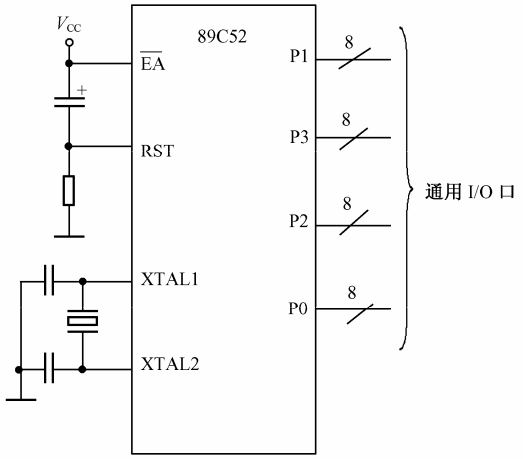
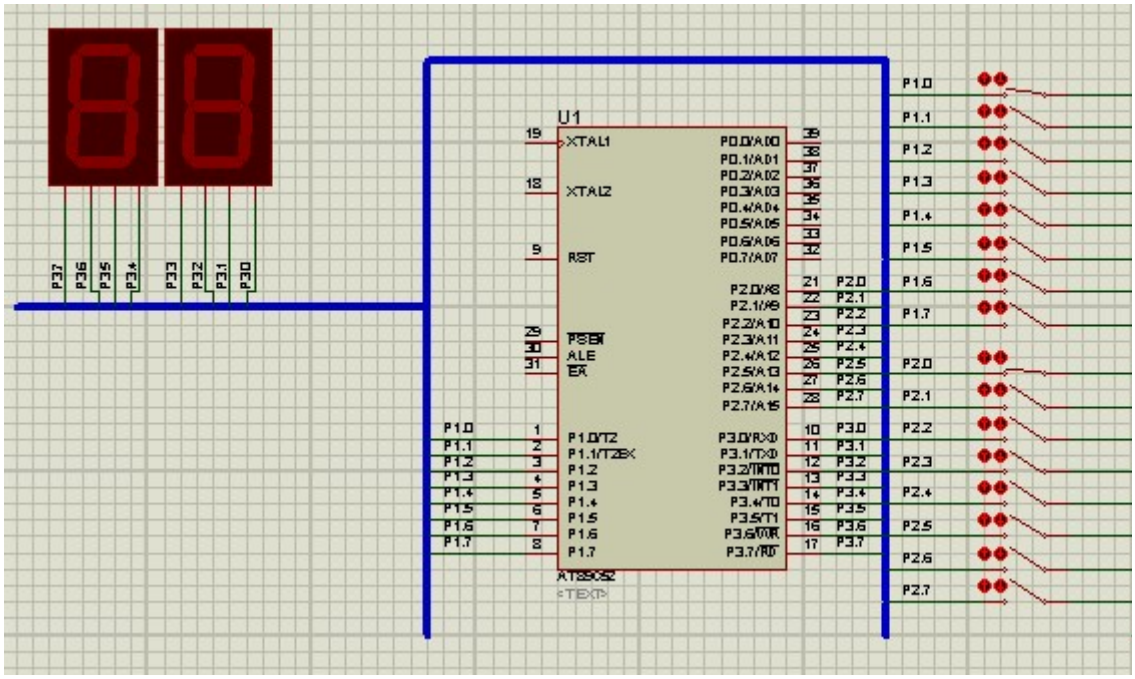


图 4.3　89C52 单片机组成的最小系统

16. 运用前面已掌握的知识，实现一个单片机硬件加法器：在 Proteus 中涉及仿真电路，从 AT89C52 的 P1 口和 P2 口输入两个数相加，然后在 P3 口显示计算结果。设计电路并编写程序实现。

答：

电路图如下：



代码如下：

```
#include<reg52.h>

void main()
{
    P3 = P1 + P2;
}
```

习题 5

1．80C51 的指令系统具有哪些特点？

答

80C51 指令系统由 49 条单字节指令、45 条双字节指令和 17 条三字节指令组成，这样可以提高程序存储器的使用效率。对于大多数算术、逻辑运算和转移操作，可选用短地址或长地址指令实现，以提高运算速度、编程效率和节省存储器单元。在 111 条指令中，有 64 条指令的执行时间为 12 个振荡器周期（1 个机器周期），45 条为 24 个振荡器周期（2 个机器周期），只有乘、除法指令需 48 个振荡周期（4 个机器周期）。当主频为 12MHz 时，典型指令的执行时间为 1μs，运算速度是比较快的。

2．80C51 单片机的指令系统按其功能可归纳为几大类？请写出各类名称。

答：

80C51 单片机指令系统操作码助记符按功能可分为五大类。

（1）数据传送类指令（7 种助记符）

（2）算术运算类指令（8 种助记符）

（3）逻辑运算类指令（10 种助记符）

（4）控制转移类指令（18 种助记符）

（5）位操作指令（1 种助记符）

3．何为寻址方式？80C51 单片机有哪些寻址方式？

答：

80C51 单片机共有 7 种寻址方式：

- 寄存器寻址
- 寄存器间接寻址
- 直接寻址
- 立即寻址
- 基址寄存器+变址寄存器的间接寻址
- 相对寻址
- 位寻址

4．什么是源操作数？什么是目的操作数？通常在指令中如何加以区分？

答：

在双操作数的指令中，指令执行后的结果放在其中的一个操作数中，这个操作数是目的操作数，另外一个是源操作数。80C51 指令系统中，不靠近指令操作码的是源操作数，源操作数可以是立即数。80C51 指令系统中，紧跟在指令操作码之后是目的操作数，算术运算和大多数逻辑运算其目的操作数必须是累加器 A，目的操作数不允许使用立即数寻址方式。

5．查表指令是在什么空间上的寻址操作？

答：

由于对程序存储器只能读而不能写，因此其数据传送是单向的，即从程序存储器读取数据，且只能向累加器 A 传送。这类指令共有两条，其功能是对存放于程序存储器中的数据表格进行查找传送，所以又称查表指令。

```
MOVC A,@A＋DPTR
MOVC A,@A＋PC
```

这两条指令都为变址寻址方式。前一条指令以 DPTR 作为基址寄存器进行查表，使用前可先给 DPTR 赋予任何地址，因此查表范围可达整个程序存储器的 64KB 空间。后一条指令以 PC 作为基址寄存器，虽然也提供 16 位基址，但其值是固定的。由于 A 的内容为 8 位无符

号数，所以这种查表指令只能查找所在地址以后256B范围内的常数或代码。

6．对80C51片内RAM的128～255字节区的地址空间寻址时，应注意些什么？对特殊功能寄存器，应采用何种寻址方式进行访问？

答：

此空间有2类不同的物理存储空间，一个是特殊功能寄存器区，一个是RAM区。

直接寻址访问特殊功能寄存器，间接寻址访问RAM区。

7．写出完成下列要求的C语言程序。

（1）将地址为4000H的片外数据存储单元内容，送入地址为30H的片内数据存储单元中。

答：

可使用绝对地址访问函数实现　　DBYTE[0x30] = XBYTE[0x4000];

（2）将地址为4000H的片外数据存储单元内容，送入地址为3000H的片外数据存储单元中。

答：

可使用绝对地址访问函数实现　　XBYTE [0x3000] = XBYTE[0x4000];

（3）将地址为0800H的程序存储单元内容，送入地址为30H的片内数据存储单元中。

答：

可使用绝对地址访问函数实现　　DBYTE [0x30] = CBYTE[0x0800];

（4）将片内数据存储器中地址为30H与40H的单元内容交换。

答：

```
unsigned char temp;
temp= DBYTE [0x40];
DBYTE [0x40]= DBYTE [0x30];
DBYTE [0x30]= temp;
```

此题答案不唯一，也可用指针运算或其他方案实现；

（5）将片内数据存储器中地址为30H单元的低4位与高4位交换。

答：

使用C51提供的本征函数_cror_实现；

```
#incluclе <intrins.h>
_cror_(DBYTE [0x30],4);
```

8．将30H、31H单元中的十进制数与38H、39H单元中的十进制数做十进制加法，其和送入40H、41H单元中，即（31H，30H）+（39H，38H）→（41H，40H）。

答：

```
#include <REG52.H>//片内寄存器定义
#include <absacc.h>
/********** main C **************/
void main (void)
{
unsigned int sum;
sum = DBYTE[0x30]+DBYTE[0x38];
if((sum&0x000f)>0x9) sum += 0x06; //个位十进制调整
if(sum>0x99) sum += 0x60;//十位十进制调整
DBYTE[0x40]= sum;//和低位字节
DBYTE[0x41] = (sum>>8) + DBYTE[0x31]+DBYTE[0x39];//和高位字节
if(DBYTE[0x41]>0x09) DBYTE[0x41] += 0x06;//百位十进制调整
while(1); /*  程序在此死循环  */
}
```

9．编写程序段完成下列乘法操作：（R4，R3）×（R5），（32H，31H，30H）。此式含义是将 R4、R3 中的双字节被乘数与 R5 中的字节乘数相乘，乘积存放在地址为 32H～30H 的 3 个存储单元中（（答案不惟一）。

解：

涉及到寄存器，用汇编实现：

```
ORG 0000H
MOV SP,#49H
MOV A,R3
MOV B,R5
MUL AB
MOV R0,#30H
MOV @R0,A     ;将低 8 位存入 30H
INC R0
MOV R1,B
MOV A,R4
MOV B,R5
MUL AB
ADD A,R1
MOV @R0,A;   将中间 8 位存入 31H
MOV A,B
ADDC A,#00H   ；将中间 8 位的进位加入 A 中
INC R0
MOV @R0,A   ;将高 8 位存入 32H
```

```
SJMP $ ;程序在此死循环
END
```

10．编写程序，用 30H 单元内容除以 40H 单元内容，商送入 50H 单元，余数送入 51H 单元。（答案不惟一）

解：

涉及直接地址，用汇编实现：

```
ORG 0000H
MOV SP,#49H
MOV A,30H   ;被除数
MOV B,40H;除数
DIV AB;A/B
MOV 50H,A ;商存 50H 单元
MOV 51H,B ;余存 51H 单元
SJMP $ ;程序在此死循环
END
```

11．已知：（30H）= 55H，（31H）= 0AAH。分别写出完成下列要求的指令，并写出 32H 单元的内容。

解：

（1）（30H）&（31H）→（32H）； DBYTE[0x32]= DBYTE[0x30]& DBYTE[0x31];0x00

（2）（30H）|（31H）→（32H）； DBYTE[0x32]= DBYTE[0x30]| DBYTE[0x31];0xff

（3）（30H）^（31H）→（32H）； DBYTE[0x32]= DBYTE[0x30]^DBYTE[0x31];0xff

12．十进制调整指令 DA 起什么作用？用在何处？

答：

十进制调整指令如下：

```
DA A
```

功能是把 A 中二进制码自动调整成二-十进制码（BCD 码）。用于对 BCD 码的加法结果进行调整。

13．80C51 指令系统中有了长跳转 LJMP、长调用 LCALL 指令，为何还设置了短跳转 AJMP、短调用 ACALL 指令？在实际使用时应怎样考虑？

答：

LJMP addr16，LCALL addr16，指令码中的目标地址均是 16 位的，所以可以指向 64KB 程序存储器空间任意位置。

AJMP addr11，ACALL addr11，指令码中的目标地址均是 12 位的，所以可以指向 2KB 程序存储器空间任意位置。

当使用的程序存储器空间在 2KB 之内或产生的代码长度在 2KB 之内时所有的跳转和调用可以使用 AJMP ，ACALL 指令。否则建议使用 LJMP，LCALL。高级语言编译器根据所选目标器件和代码情况自动生成。

14．写出下列短跳转指令中标号 L00 的取值范围。

37FFH　　AJMP　L00

答：

最大转移地址为：37FFH +2KB

15．设堆栈指针（SP）= 60H：

（1）2500H　　LCALL L00

……　　……

（2）2700H　　MOV A,#03H

……　　……

（3）2750H　　RET

执行（1）指令后：(SP)、((SP))、((SP-1))、(PC) 各为多少？执行(2)指令后：(SP)、(PC)为多少？若将(1)指令改为 ACALL L00，标号 L00 的取值范围是多少?

答：

执行（1）指令后，(SP)=62H，((SP))=25H，((SP-1))=03H，(PC)= L00；

执行（2）指令后，(SP)=60H，(PC)= 2702H；

若将(1)指令改为 ACALL L00，标号 L00 的取值范围是当前 PC 的高 5 位，加上低 11 位从全 0 变全 1；例如：2500H　ACALL　L00，当前 PC=2500H，PC 高 5 位是 00100，则 L00 取值范围是：[00100, 00000000000～00100, 11111111111]

16．为什么 SJMP 指令的 rel=$时，将实现单指令的无限循环？

答：

$表示本指令所处地址，该指令相当于：Here：SJMP Here

17．有程序如下：

```
CLR C
CLR RS1
CLR RS0
MOV A,#38H
MOV R0,A
MOV 29H,R0
SETB RS0
MOV R1,A
MOV 26H,A
MOV 28H,C
```

（1）区分哪些是伪操作指令？哪些是字节操作指令？

（2）写出程序执行后，片内 RAM 有关单元的内容。

（3）如 f_{osc}=12MHz，计算这段程序的执行时间。

答：

（1）（2）

CLR C；答：位操作指令　　（C）= 0

CLR RS1；答：位操作指令　（RS1）= 0

CLR RS0；答：位操作指令　（RS0）= 0

MOV A,#38H；答：字节操作指令　A=38H

MOV R0,A；答：字节操作指令 R0=(A)=38H

MOV 29H,R0；答：字节操作指令 (29H)=(R0)=38H

SETB RS0；答：位操作指令　（RS0）=1

MOV R1,A；答：字节操作指令　R1=(A)=38H

MOV 26H,A；答：字节操作指令 (26H)=(A)=38H

MOV 28H,C；答：位操作指令 (28H)=(C)=0

（3）如 fosc=12MHz，这段程序的执行时间=11μs

18．请用位操作指令，求下列逻辑方程：

（1） $P1.7 = ACC.0\&(B.0 + P2.0) + \overline{P3.0}$

（2） $PSW.5 = P1.0\&\overline{ACC} + B.6\&\overline{P1.4}$

（3） $PSW.5 = \overline{P1.7}\&B.4 + C + \overline{ACC}\&P1.0$

答：（1）P1_7=(ACC_0)&&((B_0||P2_0)||(~P3_0));

答：（2）PSW_5=(P1_0&&(~ACC))||(B_6&&(~P1_4))　//此处实际上是取 ACC.0

答：（3）PSW_5=(~(P1_7)&&B_4)||CY||((~ACC)&&P1_0)　//此处实际上是取 ACC.0

19．写出下列各条指令的机器码，并逐条写出依次执行每条指令后的结果和 PSW 的内容：

（1）CLR A

（2）MOV A, #9BH

（3）MOV B, #0AFH

（4）ADD A, B

答：

（1）CLR A　　//答案：机器码：E4　(A)=0 , (B)=0, PSW=0x00

（2）MOV A, #9BH　//答案：机器码：749B　(A)=0x9B , (B)=0, PSW：0x01

（3）MOV B, #0AFH //答案：机器码：75F0AF　(A)=0x9B , (B)=0xAF,　PSW：0x01

（4）ADD A, B　//答案：机器码：25F0　(A)=0x4A , (B)=0xAF,　PSW：0xc5

20．伪指令与汇编指令有何区别？说出常用的 5 种伪指令的作用。

答：

- 汇编指令，编译后产生机器码的指令；
- 伪指令，仅供汇编程序使用，编译后不产生机器码的指令。

常用的 5 种伪指令如下：

1．设置起始地址 ORG（Origin）

指令格式：ORG *nn*

作用：将 ORG *nn* 后的程序机器码或数据存放在以 *nn* 为首地址的存储单元中。如伪指令 ORG 1000H，是将目标程序从地址 1000H 处开始存放的。

2．定义字节 DB 或 DEFB（Define Byte）

指令格式：[LABEL] DB N1，N2，…，N*m*

作用：将 DB 后的 8 位字节数据 N1，N2，…，N*m* 依次存入以标号 LABEL 为首地址的存储单元中。若无标号，则 N1，N2，…，N*m* 依次存放在 DB 上一条指令之后的存储单元中。

3．定义字 DW 或 DEFW(Define Word)

指令格式：[LABEL] DW NN1，NN2，…，NN*m*

作用：将 DW 后的 16 位字数据 NN1，NN2，…，NN*m* 依次存放到以标号 LABEL 为首地址的存储单元中，若无标号，则 NN1，NN2，…，NN*m* 依次存放在 DW 上一条指令之后的存储单元中。

4．为标号赋值 EQU（Equate）

指令格式：LABEL EQU *nn*（或 *n*）

作用：将 16 位地址 *nn*（或 8 位地址 *n*）赋给标号 LABEL。

5．结束汇编 END

指令格式：END

作用：汇编程序编译源程序时，遇到伪指令 END，不管其后是否还有其他指令都将停止编译。

21．在单片机应用开发系统中，C 语言编程与汇编语言编程相比有哪些优势？

答：

汇编语言有执行效率高的优点，但其可移植性和可读性差，并且它本身就是一种编程效率低的低级语言，这些都使它的编程和维护极不方便，从而导致了整个系统的可靠性也较差。而使用 C 语言进行单片机应用系统的开发，有着汇编语言编程不可比拟的优势。

（1）编程调试灵活方便

（2）生成的代码编译效率高

（3）完全模块化

（4）可移植性好

（5）便于项目维护管理

22．在 C51 中有几种关系运算符？请列举。

答：

C51 中有 6 种关系运算符：

> 大于

< 小于

>= 大于等于

<= 小于等于

== 测试等于

!= 测试不等于

23．在 C51 中为何要尽量采用无符号的字节变量或位变量？

答：

采用无符号的字节变量或位变量可提高代码效率的方法就是减小变量的长度，使用 ANSI C 编程时，一般习惯于对变量使用 int 类型，而对于像 80C51 这类 8 位的单片机来说这是一种极大的浪费。80C51 单片机机器指令只支持字节和位变量，所以应该仔细考虑所声明的变量值的可能的取值范围，然后选择合适的变量类型。尽可能地选择变量类型为 char、unsigned char 或 bit，它们只占用 1B 或 1 位。

24．为了加快程序的运行速度，C51 中频繁操作的变量应定义在哪个存储区？

答：

局部变量和全局变量可以被定义在任何一个存储区中，根据前面的讨论，把经常使用的变量放在内部 RAM 中时，可使程序的速度得到提高。除此之外，还缩短了程序代码，因为外部存储区寻址的指令，相对要麻烦一些。考虑到存取速度，推荐读者按 data→idata→pdata→xdata 的顺序使用存储器，当然要记得在 idata 空间中留出足够的堆栈空间。

25．为何在 C51 中避免使用 float 浮点型变量？

答：

在 80C51 单片机系统上使用 32 位浮点数是得不偿失的，这样做会浪费单片机大量的存储器资源和程序执行时间。一定要在系统中使用浮点数的时候，可以通过提高数值数量级或使用整型运算代替浮点运算。在运算时，可以进行定点运算的尽量进行定点运算，避免进行浮点运算。尽量减少乘除法运算，如$*2^n$ 或$/2^n$，可以使用移位操作代替乘除法运算，这样不仅可以减少代码量，同时还能大大提高程序执行效率。处理 ints 和 longs 比处理 doubles 和 floats 要方便得多，代码执行起来会更快，C51 编译器也不用连接处理浮点运算的模块。

26．如何定义 C51 的中断函数？

答：

80C51 的中断系统十分重要，C51 编译器允许在 C 语言源程序中声明中断和编写中断服务程序，从而减轻了采用汇编程序编写中断服务程序的烦琐程度。通过使用 interrupt 关键字来实现。定义中断服务程序的一般格式如下：

```
void 函数名( ) interrupt n [using m]
```

关键字 interrupt 后面的 *n* 是中断号，*n* 的取值范围为 0～31。编译程序从 8*n*+3 处产生中断向量，即在程序存储器 8*n*+3 地址处形成一条长跳转指令，转向中断号 *n* 的中断服务程序。中断号对应着 IE 寄存器中的使能位，换句话说 IE 寄存器中的 0 位对应着外部中断 0，相应的外部中断 0 的中断号是 0。中断号 0～4 对应中断源的关系如主教材表 5.14 所示。

表 5.14　中断号和中断源的对应关系

中 断 号 *n*	中　断　源	中 断 向 量
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

using *m* 指明该中断服务程序所对应的工作寄存器组，取值范围为 0～3。指定工作寄存器组的缺点是所有被中断调用的过程都必须使用同一个寄存器组,否则参数传递会发生错误。通常不设定 using *m*，除非保证中断程序中未调用其他子程序。

设置一个定时器中断服务程序的例子如下：

```
#include <reg51.h>
#include <stdio.h>
#define RELOADTH 0x3C
#define RELOADTL 0xB0
extern unsigned int time0_counter;

void timer0(void) interrupt 1{
    TR0=0;                    /*停止定时器 0*/
    TH0=RELOADTH;             /* 50ms 后溢出*/
    TL0=RELOADTL;
    TR0=1;                    /*启动 T0*/
    time0_counter++;          /*中断次数计数器加 1*/
    printf("time0_counter=%05d\n", time0_counter);
}
```

使用 C51 编写中断服务程序，程序员无须关心 ACC、B、DPH、DPL、PSW 等寄存器的保护，C51 编译器会根据上述寄存器的使用情况在目标代码中自动增加压栈和出栈。

习题 6

1．80C51 有几个中断源，各中断标识是如何产生、如何复 0 的？CPU 响应中断时，其中断入口地址各是多少？

答：

80C5l 共有 3 类 5 个中断源，分别是 2 个外部中断源、2 个定时中断源、1 个串行口接收/发送中断源。

要实现中断，首先中断源要提出中断请求，单片机内中断请求的过程是特殊功能寄存器 TCON 和 SCON 相关状态位——中断请求标识位置 1 的过程，当 CPU 响应中断时，中断请求标识位才由硬件或软件清 0。

2 个外部中断源和 2 个定时中断源的标识由硬件自动复 0，串行口接收/发送中断由软件

清 0。

中断入口地址如下表：

表 6.6　80C51 中断向量地址分配

中　断　源	中断入口地址
外部中断 0	0003H
定时器 T0 中断	000BH
外部中断 1	0013H
定时器 T1 中断	001BH
串行口中断	0023H

2．在外部中断中，有几种中断触发方式？如何选择中断源的触发方式？

答：

外部中断源请求有两种触发方式：电平方式和脉冲方式，可通过特殊功能寄存器 TCON 中的控制位 IT0 和 IT1 定义。电平方式低电平有效，而脉冲方式则是脉冲的下降沿有效。一旦输入信号有效，特殊功能寄存器 TCON 中的中断标识位 IE0 或 IE1 被置 1，外部中断信号便向 CPU 发出了中断请求申请。

3. 80C51 提供哪几种中断？在中断管理上有何特点？什么是同级内的优先权管理？中断被封锁的条件有哪些？

答：

80C5l 共有 3 类 5 个中断源，分别是 2 个外部中断源、2 个定时中断源、1 个串行口接收/发送中断源。

中断管理是通过设置中断允许控制寄存器 IE 和中断优先级寄存器 IP 完成的。

中断源申请后，中断能否被响应，取决于 CPU 对中断源的开放或屏蔽状态，由内部的中断允许寄存器 IE 进行控制，IE 的地址是 A8H，位地址为 AFH～A8H，其内容如表 6.4 所示。

表 6.4　中断允许控制寄存器

位　地　址	AFH	AEH	ADH	ACH	ABH	AAH	A9H	A8H
位　符　号	EA	—	—	ES	ET1	EX1	ET0	EX0

IE 中与中断有关的共有 6 位，各位含义如下。

EA：CPU 中断允许总控制位。EA = l，CPU 开放中断，此时，每个中断源的中断允许或禁止，取决于各自的中断允许控制位；EA = 0，CPU 屏蔽所有中断，即中断总禁止。

EX0、EX1：外部中断允许控制位。EX0(EX1) = l，允许外部中断；EX0(EX1) = 0，禁止外部中断。

ET0、ET1：定时器/计数器溢出中断允许控制位。ET0(ET1)=1，允许中断；ET0(ET1)=0，禁止定时器/计数器中断。

ES：串行口中断允许控制位。ES=1，允许串行口中断；ES=0，禁止串行口中断。

以上中断控制为两级控制，即以 EA 实现中断总控，以各中断源的中断允许位实现分控。当总控制位为禁止时，不管分控位状态如何，整个中断系统被禁止。只有当总控制位为允许

状态时，中断的允许与禁止才能由各分控制位决定，参见图 6.3。

80C51 有两个中断优先级，每个中断源均可通过软件设置为高优先级或低优先级中断，实现 2 级中断嵌套。高优先级中断请求可以中断一个正在执行的低优先级中断服务，除非正在执行的低优先级中断服务程序设置了禁止某些高优先级的中断。正在执行的中断服务程序不能被另一个同级或低优先级的中断所中断；正在执行高优先级的中断服务程序，不能被任何中断源中断。一直执行到返回指令 RETI，返回主程序，而后再执行一条指令后，才能响应新的中断申请。

为实现以上功能，80C51 中断系统设有两个不可寻址的优先级状态触发器，一个指示 CPU 是否正在执行高优先级中断服务程序，而另一个指示 CPU 是否正在执行低优先级中断服务程序。前一个触发器的 1 状态屏蔽所有的中断申请，而后一个触发器的 1 状态屏蔽相同优先级的其他中断申请。

特殊功能寄存器 IP 为中断优先级控制寄存器，其地址为 B8H，位地址为 BFH～B8H，各位内容如表 6.5 所示。

表 6.5　中断优先级控制寄存器

位 地 址	BFH	BEH	BDH	BCH	BBH	BAH	B9H	B8H
位 符 号	—	—	—	PS	PT1	PX1	PT0	PX0

PX0：外部中断 0 中断优先级控制位。PX0＝l，外部中断 0 定义为高优先级中断；PX0＝0，为低优先级中断。

PT0：定时器 0 中断优先级控制位。PT0＝1，定时器 T0 中断定义为高优先级中断；PT0＝0，为低优先级中断。

PX1：外部中断 1 中断优先级控制位。PX1＝l，外部中断 1 定义为高优先级中断；PX1＝0，为低优先级中断。

PT1：定时器 1 中断优先级控制位。PT1＝1，定时器 T1 中断定义为高优先级中断；PT1＝0，为低优先级中断。

PS：串行口中断优先级控制位。PS=1 时，串行口中断定义为高优先级中断；PS=0 时，为低优先级中断。

当系统复位后，IP 的所有位被清 0，所有的中断源均被定义为低优级中断。IP 的各位都可以用程序置位和复位，也可以用位操作指令或字节操作指令更新 IP 的内容，以改变各中断源的中断优先级。

当同一优先级的几个中断源同时向 CPU 提出中断请求时，CPU 通过内部硬件查询逻辑电路，按查询顺序判定优先响应哪一个中断请求，其查询顺序为：外部中断 0、定时中断 0、外部中断 1、定时中断 1、串行中断。

4．在中断请求有效并开中断状况下，能否保证立即响应中断？有什么条件？

答：

条件是不出现下列情况：

- 一：CPU 正在处理同级的或更高优先级的中断。
- 二：当前的机器周期不是所执行指令的最后一个机器周期，即在当前指令完成之前，

CPU 不会响应任何中断请求。

- 三：正在执行的指令是 RETI 或访问 IE 或 IP 的指令。CPU 完成这类指令后，至少还要再执行一条指令才会响应新的中断请求，以便保证程序能够正确地返回。

5．中断响应中，CPU 应完成哪些自主操作？这些操作状态对程序运行有什么影响？

答：

CPU 响应中断时，先置位相应的高/低优先级状态触发器，指出 CPU 开始处理的中断优先级别，然后由硬件生成一条长调用指令 LCALL，其格式为：LCALL addr16。其中，addr16 是在程序存储区中与各中断请求对应的中断入口地址，也称为中断向量地址。

6．80C51 单片机内部设有几个定时器/计数器？它们各由哪些特殊功能寄存器所组成？有哪几种工作方式？

答：

80C51 的单片机内有 2 个 16 位可编程的定时器/计数器，它们具有 4 种工作方式，其控制字和状态均在相应的特殊功能寄存器中，通过对控制寄存器的编程就可以方便地选择适当的工作方式。

特殊功能寄存器包括加 1 计数器和控制寄存器：

16 位加 1 计数器 TH0、TL0 和 TH1、TL1；

定时控制寄存器（TCON）和工作方式控制寄存器（TMOD）；

有 4 种工作方式，分别是：

方式 0：13 位定时器 / 计数器

方式 1：16 位定时器 / 计数器

方式 2：初值自动重新装入的 8 位定时器 / 计数器

方式 3：仅适用于 T0，将其分为两个 8 位计数器。对 T1 停止计数

7．80C51 定时器/计数器方式 0 的 13 位计数器初值如何计算？已有方式 1 的 16 位计数为何还需要 13 位的计数方式？

答：

当 TMOD 中 M1M0＝00 时，定时器/计数器选定方式 0 进行工作。不妨以定时器/计数器 T0 为例解释。图 6.7 所示为 T0 工作在方式 0 下的逻辑结构图（定时器/计数器 1 与其完全一致）。在该工作方式下，TH0 的全部 8 位和 TL0 的低 5 位构成 13 位的加 1 计数器，计数器的最大值为 $2^{13}=8192$，而 TL0 的高 3 位处于闲置状态，这是出于与 MCS-48 单片机兼容性的考虑，因为 MCS-48 单片机的加 1 计数器是 13 位的。

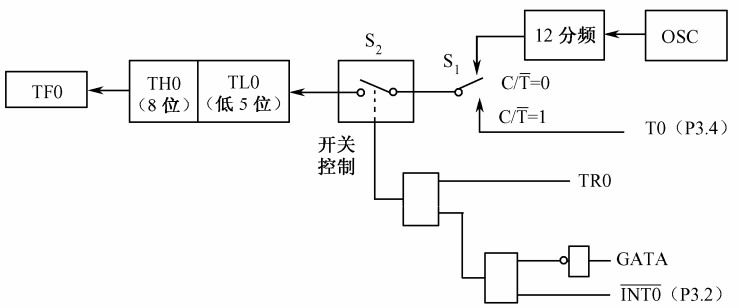


图 6.7　定时器/计数器 0 的工作方式 0 电路逻辑结构

计数器初值为：

$$N=8192-x$$

计数初值 x 是 TH0、TL0 设定的初值。$x=8191$ 时为最小计数值 l，$x=0$ 时为最大计数值 8192，即计数范围为 1～8192（2^{13}）。

8．定时器/计数器作定时用时，定时时间与哪些因素有关？作为计数用时，对外界计数频率有何限制？

答：

当 80C5l 内部的定时器/计数器被选定为定时器工作模式时，计数输入信号是内部时钟脉冲，此当需要高分辨率的定时，应尽量选用频率较高的晶振（80C5l 最高为 40 MHz）。

当定时器/计数器用作计数器时，计数脉冲来自外部输入引脚 T0 或 T1。当输入信号产生由 1 至 0 的跳变（即负跳变）时，计数器的值增 l。每个机器周期的 S5P2 期间，对外部输入进行采样。如在第一个周期中采得的值为 1，而在下一个周期中采得的值为 0，则在紧跟着的再下一个周期 S3P1 的期间，计数器加 1。由于确认一次下跳变需要花 2 个机器周期，即 24 个振荡周期，因此外部输入的计数脉冲的最高频率为振荡器频率的 1/24。例如，选用 6MHz 频率的晶体，允许输入的脉冲频率为 250kHz，如果选用 12MHz 频率的晶体，则可输入 500kHz 的外部脉冲。对于外部输入信号的占空比并没有什么限制，但为了确保某一给定的电平在变化之前能被采样一次，则这个电平至少要保持一个机器周期。故对输入信号的基本要求如图 6.12 所示，图中 T_{cy} 为机器周期。

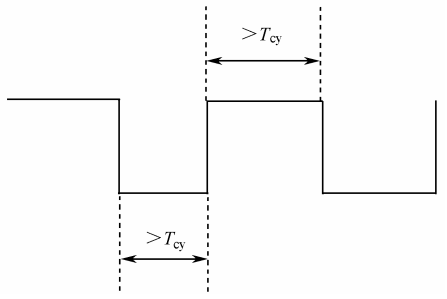


图 6.12　对输入信号的基本要求

9．定时器 T0 为方式 3 时，由于 TR1 位已经被 T0 占用，如何控制定时器 T1 的开启和关闭？

答：

在工作方式 3 下，定时器/计数器 T0 被拆成两个独立的 8 位计数器 TL0 和 TH0。其中，TL0 既可以计数使用，又可以定时使用，定时器/计数器 T0 的控制位和引脚信号全归它使用。其功能和操作与方式 0 或方式 1 完全相同，而且逻辑电路结构也极其类似，如图 6.10 所示。

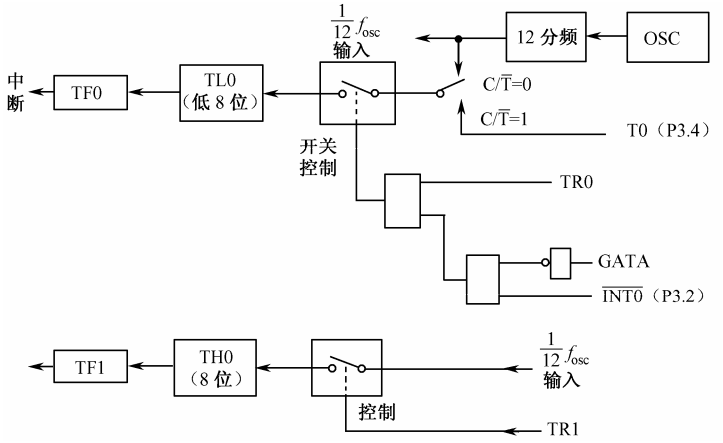


图 6.10　工作方式 3 下定时器/计数器 T0 被分成两个 8 位计数器的逻辑结构

与 TL0 的情况相反，对于 T0 的另一半 TH0，则只能作为简单的定时器使用。而且由于 T0 的控制位已被 TL0 独占，因此只好借用定时器/计数器 T1 的控制位 TR1 和 TF1，以计数溢出去置位 TF1，而定时的启动和停止则受 TR1 的状态控制。

由于 TL0 既能做定时器使用，也能做计数器使用，而 TH0 只能做定时器使用却不能做计数器使用，因此在工作方式 3 下，定时器/计数器 0 可以构成 2 个定时器或 1 个定时器、1 个计数器。

如果定时器/计数器 T0 已工作在工作方式 3 下，则定时器/计数器 T1 只能工作在方式 0、方式 1 或方式 2 下，它的运行控制位 TR1 及计数溢出标识位 TF1 已被定时器/计数器 0 借用，如图 6.11 所示。

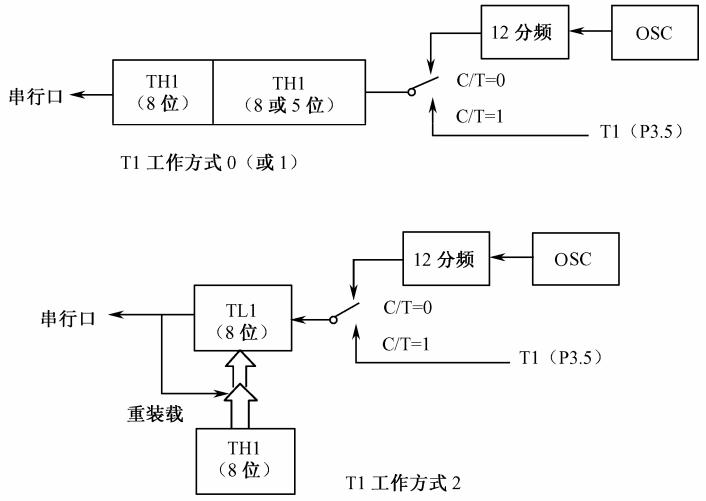


图 6.11　定时器/计数器方式 3 的 T1 结构

在这种情况下定时器/计数器 1 通常是作为串行口的波特率发生器使用，以确定串行通信的速率。因为已没有计数溢出标识位 TF1 可供使用，因此只能把计数溢出直接送给串行口。当作为波特率发生器使用时，只需设置好工作方式，便可自动运行。如果要停止工作，只需送入一个把它设置为方式 3 的方式控制字就可以了，因为定时器/计数器 1 不能在方式 3 下使用，如果硬把它设置为方式 3，则停止工作。

10．用 80C51 的定时器测量某正单脉冲的宽度，采用何种方式可以得到最大量程？若单片机晶振频率为 12MHz，求允许测量的最大脉冲宽度是多少？

答：

当 80C5l 内部的定时器/计数器被选定为定时器工作模式时，计数输入信号是内部时钟脉冲，每个机器周期产生一个脉冲位，计数器增 l。将定时器 0 或 1 设置为工作方式 1，即 16 位定时器方式，并且计数初值设置为 0x0000，计满 65536 定时脉冲后溢出，此时可以得到最大量程。

当采用 12MHz 频率的晶体时，每个机器周期为 1μs，测量的最大脉冲宽度也是 65536μs。

11．80C51 单片机系统中，已知单片机晶振频率为 6MHz，选用定时器 0 以方式 3 产生周期为 400μs 的等宽正方波连续脉冲，请编写由 P1.1 口输出此方波的程序。

解：

示例程序如下：

```
#include "REG51.H"
sbit rect_wave=P1^1;
void time_over(void);
void main(void){
TMOD=0x03;
TL0=160;
```

```
IE=0x00;
TR0=1;
for(;;){
      if(TF0)time_over();
  }
  }

void time_over(void)
{
TF0=0;
TL0=160;
rect_wave=!rect_wave;
}
```

12．串行通信操作模式有哪几种?各有什么特点?

答：

串行通信操作模式有 2 种，异步串行通信和同步串行通信。

异步串行通信（以下简称为异步通信）所传输的数据格式（也称为串行帧）由 1 个起始位、7 个或 8 个数据位、1～2 个停止位（含 1.5 个停止位）和 1 个校验位组成。起始位约定为 0，空闲位约定为 1。在异步通信方式中，接收器和发送器有各自的时钟，它们的工作是非同步的。图 6.17 所示为异步通信方式和异步通信数据格式示意图，数据格式是 1 个起始位、8 个数据位、1 个停止位，所传输的数据是 35H（00110101）。

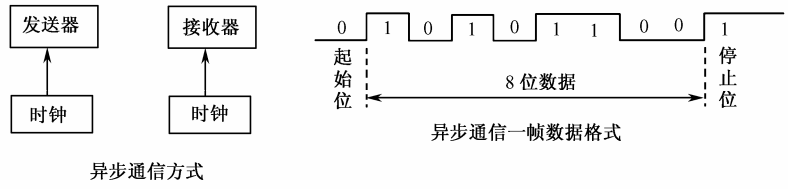


图 6.17　异步通信方式和异步通信数据格式示意图

异步通信的实质是指通信双方采用独立的时钟，每个数据均以起始位开始、停止位结束，起始位触发甲乙双方同步时钟。每个异步串行帧中的 1 位彼此严格同步，位周期相同。所谓异步是指发送、接收双方的数据帧与帧之间不要求同步，也不必同步。

同步串行通信（以下简称为同步通信）中，发送器和接收器由同一个时钟源控制。而在异步通信中，每传输一帧字符都必须加上起始位和停止位，占用了传输时间，在要求传送数据量较大的场合，速度就会慢得多。同步传输方式去掉了这些起始位和停止位，只在传输数据块时先送出一个同步头（字符）标识即可。图 6.18 所示为同步通信方式和同步通信数据格式示意图。由图 6.18 可知，同步通信所传输的数据格式（也称同步串帧）是由多个数据构成的，每帧有 2 个同步字符作为起始位以触发同步时钟开始发送或接收数据。空闲位需发送同步字符。因此，同步是指发送、接收双方的数据帧与帧之间严格同步，而不只是位与位之间

严格同步。

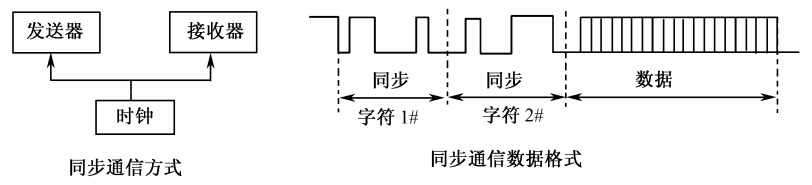


图 6.18　同步通信方式和同步通信数据格式示意图

同步传输方式比异步传输方式速度快，这是它的优势。但同步传输方式也有其缺点，即它必须要用一个时钟来协调收发器的工作，所以它的设备也较复杂。

由上所述可以得到推论：异步通信比较灵活，适用于数据的随机发送/接收，而同步通信则是成批数据传送的。异步传输一批数据，因为每个字节均有起始位和停止位控制而使发送/接收速度有所降低，一般适用于每秒 50～19200 位，而同步传输速度较快，可达每秒 80 万位。

13．异步串行通信时，通信双方应遵守哪些协定？一帧信息包含哪些内容?

答：

异步串行通信（以下简称为异步通信）所传输的数据格式（也称为串行帧）由 1 个起始位、7 个或 8 个数据位、1～2 个停止位（含 1.5 个停止位）和 1 个校验位组成。起始位约定为 0，空闲位约定为 1。在异步通信方式中，接收器和发送器有各自的时钟，它们的工作是非同步的。

图 6.17 所示为异步通信方式和异步通信数据格式示意图，数据格式是 1 个起始位、8 个数据位、1 个停止位，所传输的数据是 35H（00110101）。

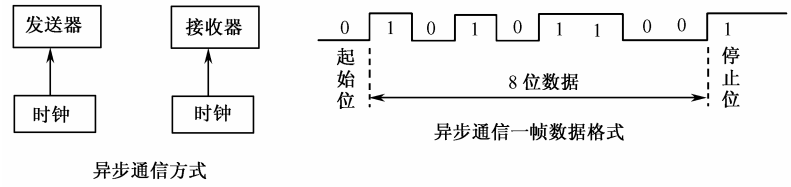


图 6.17　异步通信方式和异步通信数据格式示意图

14．串行通信时会出现哪些错误?用什么方法检查这些错误?

答：

异步通信时可能会出现帧格式错、超时错等传输错误。在具有串行口的单片机的开发中，应考虑在通信过程中对数据差错进行校验，因为差错校验是保证准确无误通信的关键。

常用差错校验方法有奇偶校验（80C51 系列单片机编程采用此法）、和校验及循环冗余码校验。

15．80C51 串行通信有哪几种工作方式？当并行口不够用时，如何实现串行口作并行口使用？

答：

80C51 串行通信有 4 工作方式，方式 0～方式 3

方式 0 下，串行口作为同步移位寄存器使用。这时用 RXD（P3.0）引脚作为数据移位的入口和出口，而由 TXD（P3.1）引脚提供移位脉冲。移位数据的发送和接收以 8 位为一帧，不设起始位和停止位，低位在前高位在后，利用这一特点，当并行口不够用时，可以通过串行口扩展并行口。具体方法见主教材 P214～P215。

16．什么是波特率？如何计算和设置 80C51 串行通信的波特率。

答：

波特率是串行通信中一个重要概念。波特率 BR 是单位时间传输的数据位数，即单位 bps（bit per second），1bps=1bit/s。波特率的倒数即为每位传输所需的时间。由上面介绍的异步串行通信原理可知，互相通信甲乙双方必须具有相同的波特率，否则无法成功的完成数据通信。发送和接收数据是由同步时钟触发发送器和接收器而实现的。发送/接收时钟频率与波特率有关，即：

$$f_{T/R} = n \times BR_{T/R}$$

式中，$f_{T/R}$ 为发/收时钟频率，单位为 Hz；$BR_{T/R}$ 为发/收波特率，单位为 bps；n 为波特率因子。

同步通信 n=1。异步通信 n 可取 1、16 或 64。也就是说，同步通信中数据传输的波特率即为同步时钟频率，而异步通信中，时钟频率可为波特率的整数倍。

① 方式 0 时波特率是固定的，为单片机晶振频率的 1/12，即 BR=f_{osc}/12。如果晶振频率用 f_{osc} 表示，按此波特率也就是一个机器周期进行一次移位。当 f_{osc} = 6MHz 时，波特率为 500kbps，即 2μs 移位一次；当 f_{osc} =12MHz 时，波特率为 1Mbps，即 1μs 移位一次。

② 方式 2 的波特率也是固定的，且有两种：一种是晶振频率的 1/32，另一种是晶振频率的 1/64，即 f_{osc}/32 和 f_{osc}/64。例如，用公式表示为：

$$BR = 2^{SMOD} \times f_{osc}/64$$

式中，SMOD 为 PCON 寄存器最高位的值，SMOD=1 表示波特率加倍。

③ 方式 1 和方式 3 的波特率是可变的，其波特率由定时器 1 的计数溢出（对 80C52 来说，也可以使用定时器 2 的计数溢出）决定，公式为：

$$BR = (2^{SMOD} \times T_d)/32$$

式中，SMOD 为 PCON 寄存器最高位的值，SMOD = 1 表示波特率加倍。而定时器 1 溢出率计算公式为：

$$T = f_{osc}/[12 \times (256 - TH1)]$$

使用定时器 1 的计数溢出，方式 1 和方式 3 的常用波特率如表 6.16 所示。

表 6.16　方式 1 和方式 3 的常用波特率

串行口工作方式	波　特　率	f_{osc}=6MHz			f_{osc}=12MHz			f_{osc}=11.0592MHz		
		SMOD	TMOD	TH1	SMOD	TMOD	TH1	SMOD	TMOD	TH1
方式 1 或方式 3	57600							1	20	FFH
	28800							1	20	FEH

	19200						1	20	FDH
	9600						0	20	FDH
	4800				1	20	F3H	0	20	FAH
	2400	1	20	F3H	0	20	F3H	0	20	F4H
	1200	1	20	E6H	0	20	E6H	0	20	E8H
	600	1	20	CCH	0	20	CCH	0	20	D0H
	300	0	20	CCH	0	20	98H	0	20	A0H

从表 6.16 中可以看出，当选择晶振 f_{osc}=11.0592MHz 时，波特率最为齐全，如无特别要求，通常在 80C51 单片机系统中选择晶振频率为 11.0592MHz。

设置定时器 2 为波特率发生器工作方式，定时器 2 的溢出脉冲经 16 分频后作为串行口发送脉冲、接收脉冲。发送脉冲、接收脉冲的频率称为波特率。其计算公式如下：

$$\mathrm{BR}=\frac{f_{\mathrm{osc}}}{32\times[65536-(\mathrm{RCAP2H,RCAP2L})]}$$

17．串行口控制寄存器 SCON 中 TB8、RB8 起什么作用？在什么方式下使用？

答：

TB8——发送数据位 8

在方式 2 和方式 3 时，TB8 是发送的第 9 位数据。在多机通信中，以 TB8 位的状态表示主机发送的是地址还是数据。TB8=0 为数据，TB8=1 为地址。该位由软件置位或复位。

RB8——接收数据位 8

在方式 2 或方式 3 时，RB8 存放接收到的第 9 位数据，代表着接收的某种特征，故应根据其状态对接收数据进行操作。

18．什么是串行接口的异步数据传输和同步数据传输？80C51 UART 中哪些方式是异步传输？哪些方式是同步传输？

答：

异步数据传输的实质是指通信双方采用独立的时钟，每个数据均以起始位开始、停止位结束，起始位触发甲乙双方同步时钟。每个异步串行帧中的 1 位彼此严格同步，位周期相同。所谓异步是指发送、接收双方的数据帧与帧之间不要求同步，也不必同步。

同步数据传输，发送器和接收器由同一个时钟源控制。

80C51 除了方式 0 是同步传输外，其余均为异步传输。

19．80C51 单片机内部串行口工作方式工作于方式 1、3 时波特率与 T1 的溢出率有关，什么是 T1 的溢出率?，如何计算定时器 T1 工作于方式 1 时的 T1 溢出率?

答：

计数器计满时（最后一个计数状态，对 16 位计数器一般为 65535），再来一个脉冲，会产生一个溢出信号，计数器复位至第一个状态（第一个计数状态，对 16 位计数器一般为 0），从 0～65535 这是满量程计数的情况。如果想不想满量程，则需要设置初值，如设置初值为 65536-1000=64536，则计数器输入 1000 个脉冲就会产生溢出信号。对 T1 而言，单位时间的

溢出次数即 T1 的溢出率。

方式 1 和方式 3 的波特率是可变的，其波特率由定时器 1 的计数溢出（对 80C52 来说，也可以使用定时器 2 的计数溢出）决定，公式为：

$$BR = (2^{SMOD} \times T_d) /32$$

式中，SMOD 为 PCON 寄存器最高位的值，SMOD = 1 表示波特率加倍。而定时器 1 溢出率计算公式为：

$$T = f_{osc}/[12\times(256-TH1)]$$

20．设置串行口工作于方式 3，波特率为 9600bps，系统主频为 11.0592MHz，允许接收数据，串行口开中断，初始化编程，实现上述要求。若将串口改为方式 1，应如何修改初始化程序?

答：

用公式 $Count = 256 - \frac{f_{osc} \times 2^{SMOD}}{384 \times BR}$ 计算 TH1 的初值。

设置串行口工作于方式 3：

```
/********** 初始化串口波特率 ************/
void initUart(void)/*初始化串口波特率，使用定时器 1*/
{
/* Setup the serial port for 9600 baud at 11.0592MHz */
   SCON  = 0xd0;  //串口工作在方式 3，此处 SM0SM1=03
   TMOD  = 0x20;
   PCON  = 0x0;
   TH1   = 0xfd;
   TCON  = 0x40;
}
```

设置串行口工作于方式 1：

```
/********** 初始化串口波特率 ************/
void initUart(void)/*初始化串口波特率，使用定时器 1*/
{
/* Setup the serial port for 9600 baud at 11.0592MHz */
   SCON  = 0x50;  //串口工作在方式 1，此处 SM0SM1=01
   TMOD  = 0x20;
   PCON  = 0x0;
   TH1   = 0xfd;
   TCON  = 0x40;
}
```

21. 使用串行口方式 3 进行双机通信，系统主频为 11.0592MHz，设置波特率为 19200bps，定时器 T1 工作于方式 2。A 机将地址为 3000H～30FFH 外部 RAM 中数据传送到 B 机地址为 4000H～40FFH 外部 RAM 中。请编写程序：① 采用查询方式，进行偶校验，② 采用中断

方式，不要校验。（答案不惟一，仅供参考!）

解：① 采用查询方式，进行偶校验，

A-甲机，B-乙机使用一个程序，P1.0悬空，为发送方，P1.0接地，为接收方。

程序一：

```
#include <REG52.H>   /* special function register declarations */
#include <stdio.h>   /* prototype declarations for I/O functions */
#include <intrins.h>
#include <Absacc.h>
#include <string.h>
#include <ctype.h>

#define byte unsigned char
#define uchar unsigned char
#define word unsigned int
#define uint unsigned int
#define ulong unsigned long
#define BYTE    unsigned char
#define WORD    unsigned int

#define TRUE  1
#define FALSE 0

void initUart(void);/*初始化串口*/
void time(unsigned int ucMs);//延时单位：ms

/* SEND_RECI_LINE＝1，设置发送; SEND_RECI_LINE＝0，设置接收*/
sbit SEND_RECI_CTRL = P1^0;
sbit CTRL_BUTTON = P1^7; /* CTRL_BUTTON＝0，设置错误校验和 */
void initUart(void);/*初始化串口波特率，使用定时器2*/
void send(uchar *d);    /*发送函数*/
void receive(uchar *d);  /*接收函数*/

xdata uchar sbuf[0x100] _at_ 0x3000;
xdata uchar rbuf[0x100] _at_ 0x4000;

/******** main 函数 *********/
void main (void) {
     time(1);/* 延时等待外围器件完成复位 */
     initUart();/* 初始化串口 */
     if(SEND_RECI_CTRL){  /*发送*/
          send(sbuf);
     }
     else{/*接收*/
          receive(rbuf);
```

```
     }
     while(TRUE){}
}
/********** 初始化串口波特率 ************/
void initUart(void)/*初始化串口波特率，使用定时器 1*/
{
/* Setup the serial port for 19200 baud at 11.0592MHz */
     SCON  = 0xd0;  //串口工作在方式 3，此处 SM0SM1=03
     TMOD  = 0x20;
     PCON  = 0x80;  //波特率 19200-0x80   波特率 9600-0x00
     TH1   = 0xfd;
     TCON  = 0x40;
}

void send(uchar *d) /*发送函数*/
{
uchar pf;
uint  i;
     do{
          ACC=0xaa;TB8=P;
          SBUF=0xaa;  /*发送联络信号*/
          while(TI==0){}TI=0;
          while(RI==0){}RI=0;
     }while((SBUF^0xbb)!=0);  /*乙机未准备好，继续联络*/
     time(500);
     do{
          pf=0;    /*清校验和*/
          for(i=0;i<255;i++){
               time(500);
               ACC=d[i];TB8=P;
               SBUF=d[i]; /*发送一个数据*/
               pf+=d[i];  /*求校验和*/
               while(TI==0){}TI=0;
          }
          if(!CTRL_BUTTON) pf++; /* CTRL_BUTTON＝0，设置错误校验和 */
          time(500);   /*显示校验和*/
          ACC=pf;TB8=P;
          SBUF=pf;             /*发送校验和*/
          while(TI==0){}TI=0;
          while(RI==0){}RI=0;
          time(500);
     }while(SBUF!=0);  /*回答出错，则重发*/
}
```

```
void receive(uchar *d)/*接收函数*/
{
uchar pf;
uint  i;
     do{
          while(RI==0){}RI=0;
          time(500);
   }while((SBUF^0xaa)!=0); /*判甲机请求否*/
     time(500);
     ACC=0xbb;TB8=P;
   SBUF=0xbb;   /*发应答信号*/
     while(TI==0){}TI=0;
   while(1){
     pf=0;  /*清校验和*/
          for(i=0; i<255; i++){
               while(RI==0){}RI=0;
               d[i]=SBUF;      /*接收一个数据*/
               ACC=d[i];if(P!=RB8) pf=0;//偶校验错误，置校验和=0
               pf+=d[i];  /*求校验和*/
          }
          while(RI==0){}RI=0;
          if((SBUF^pf)==0){/*比较校验和*/
               time(500);
               ACC=0x00;TB8=P;
               SBUF=0x00;/*校验和相同发"0x00"*/
               while(TI==0){}TI=0;
          }
          else{
          time(500);
               ACC=0xff;TB8=P;
               SBUF=0xff;while(TI==0){}TI=0; /*校验和不同发"0xff", 重新接收*/

          }
     }
}

/*******************************************************
*  函数说明：延时5us，晶振改变时只用改变这一个函数!
     1、对于11.0592M晶振而言，需要2个_nop_();
     2、对于22.1184M晶振而言，需要4个_nop_();
*  入口参数：无
*  返回：     无
*  创建日期：20010623
*  作者： 张齐
```

```
************************************************************/
void delay_5us(void)//延时 5us，晶振改变时只用改变这一个函数!
{
     _nop_();
     _nop_();
     //_nop_();
     //_nop_();
}
/************ delay_50us **************/
void delay_50us(void)//延时 50us
{
     unsigned char i;
     for(i=0;i<4;i++)
     {
          delay_5us();
     }
}
/******** 延时 100us ******************/
void delay_100us(void)//延时 100us
{
     delay_50us();
     delay_50us();
}

/*********** 延时单位：ms ******************/
void time(unsigned int ucMs)//延时单位：ms
{
     unsigned char j;
     while(ucMs>0){
          for(j=0;j<10;j++) delay_100us();
          ucMs--;
     }
}
```

② 采用中断方式，不要校验。

```
#include <REG52.H>   /* special function register declarations */
#include <stdio.h>   /* prototype declarations for I/O functions */
#include <intrins.h>
#include <Absacc.h>
#include <string.h>
#include <ctype.h>

#define byte unsigned char
#define uchar unsigned char
```

```
#define word unsigned int
#define uint unsigned int
#define ulong unsigned long
#define BYTE    unsigned char
#define WORD    unsigned int

#define TRUE  1
#define FALSE 0

void initUart(void);/*初始化串口*/
void time(unsigned int ucMs);//延时单位：ms

/* SEND_RECI_LINE＝1，设置发送; SEND_RECI_LINE＝0，设置接收*/
sbit SEND_RECI_CTRL = P1^0;
sbit CTRL_BUTTON = P1^7; /* CTRL_BUTTON＝0，设置错误校验和 */
void initUart(void);/*初始化串口波特率，使用定时器2*/
void send(uchar *d);    /*发送函数*/
void receive(uchar *d);  /*接收函数*/

xdata uchar sbuf[0x100] _at_ 0x3000;
xdata uchar rbuf[0x100] _at_ 0x4000;
xdata uchar* p_rbuf=&rbuf[0]; //指向接收缓冲去第一个字节地址

/******** main 函数 *********/
void main (void) {
     time(1);/* 延时等待外围器件完成复位 */
     initUart();/* 初始化串口 */
     if(SEND_RECI_CTRL){  /*发送*/
          send(sbuf);
     }
     while(TRUE){}/*中断接收*/
}
/********** 初始化串口波特率 ************/
void initUart(void)/*初始化串口波特率，使用定时器1*/
{
/* Setup the serial port for 19200 baud at 11.0592MHz */
     SCON  = 0xd0;  //串口工作在方式3，此处SM0SM1=03
     TMOD  = 0x20;
     PCON  = 0x80;  //波特率19200-0x80   波特率9600-0x00
     TH1   = 0xfd;
     TCON  = 0x40;
     if(!SEND_RECI_CTRL)
          IE=0x90; //如果是接收方，则打开串口中断
}
```

```
void send(uchar *d) /*发送函数*/
{
uint  i;
     for(i=0;i<255;i++){
                time(10);
                SBUF=d[i]; /*发送一个数据*/
                while(TI==0){}TI=0;
     }
}

/***********  串行口中断服务程序**************/
void serial0_int(void) interrupt 4
{
      *p_rbuf=SBUF;RI=0;
      p_rbuf++;
}
/*********************************************************
*  函数说明：延时 5us，晶振改变时只用改变这一个函数!
     1、对于 11.0592M 晶振而言，需要 2 个_nop_();
     2、对于 22.1184M 晶振而言，需要 4 个_nop_();
*  入口参数：无
*  返回：     无
*  创建日期：20010623
*  作者： 张齐
*********************************************************/
void delay_5us(void)//延时 5us，晶振改变时只用改变这一个函数!
{
     _nop_();
     _nop_();
     //_nop_();
     //_nop_();
}
/************ delay_50us **************/
void delay_50us(void)//延时 50us
{
     unsigned char i;
     for(i=0;i<4;i++)
     {
          delay_5us();
     }
}
/******** 延时 100us *****************/
void delay_100us(void)//延时 100us
```

```
{
     delay_50us();
     delay_50us();
}

/*********** 延时单位：ms ******************/
void time(unsigned int ucMs)//延时单位：ms
{
     unsigned char j;
     while(ucMs>0){
          for(j=0;j<10;j++) delay_100us();
          ucMs--;
     }
}
```

22．80C52 串行口按工作方式 1 进行串行数据通信。假定波特率为 1200bps，系统主频为 11.0592MHz，以中断方式传送数据，将本机中地址为 30H～4FH 内部 RAM 中内容传送到对方地址为 50H～6FH 内部 RAM 中去。请编写全双工通信程序。（答案不惟一，仅供参考!）

解：

程序示例如下：

```
#include <REG52.H>   /* special function register declarations */
#include <stdio.h>   /* prototype declarations for I/O functions */
#include <intrins.h>
#include <Absacc.h>
#include <string.h>
#include <ctype.h>

#define byte unsigned char
#define uchar unsigned char
#define word unsigned int
#define uint unsigned int
#define ulong unsigned long
#define BYTE    unsigned char
#define WORD    unsigned int

#define TRUE  1
#define FALSE 0

void initUart(void);/*初始化串口*/
void time(unsigned int ucMs);//延时单位：ms

/* SEND_RECI_LINE＝1，设置发送; SEND_RECI_LINE＝0，设置接收*/
sbit SEND_RECI_CTRL = P1^0;
```

```
sbit CTRL_BUTTON = P1^7; /* CTRL_BUTTON＝0，设置错误校验和 */
void initUart(void);/*初始化串口波特率，使用定时器 2*/
void send(uchar *d);    /*发送函数*/
void receive(uchar *d);  /*接收函数*/

idata uchar sbuf[0x10] _at_ 0x30;
idata uchar rbuf[0x10] _at_ 0x50;
idata uchar* p_rbuf=&rbuf[0]; //指向接收缓冲去第一个字节地址

/******** main 函数 *********/
void main (void) {
     time(1);/* 延时等待外围器件完成复位 */
     initUart();/* 初始化串口 */
     if(SEND_RECI_CTRL){  /*发送*/
          send(sbuf);
     }
     while(TRUE){}/*中断接收*/
}
/********** 初始化串口波特率 ************/
void initUart(void)/*初始化串口波特率，使用定时器 1*/
{
/* Setup the serial port for 1200 baud at 11.0592MHz */
     SCON  = 0x50;  //串口工作在方式 1
     TMOD  = 0x20;
     PCON  = 0x0;
     TH1   = 0xe8;  //波特率 1200
     TCON  = 0x40;
     if(!SEND_RECI_CTRL)
          IE=0x90; //如果是接收方，则打开串口中断
}

void send(uchar *d) /*发送函数*/
{
uchar  i;
     for(i=0;i<15;i++){
                time(10);
                SBUF=d[i]; /*发送一个数据*/
                while(TI==0){}TI=0;
     }
}

/***********  串行口中断服务程序***************/
void serial0_int(void) interrupt 4
{
```

```
      *p_rbuf=SBUF;RI=0;
      p_rbuf++;
}
/************************************************************
*  函数说明：延时5us，晶振改变时只用改变这一个函数!
     1、对于11.0592M晶振而言，需要2个_nop_();
     2、对于22.1184M晶振而言，需要4个_nop_();
*  入口参数：无
*  返回：     无
*  创建日期：20010623
*  作者： 张齐
************************************************************/
void delay_5us(void)//延时5us，晶振改变时只用改变这一个函数!
{
     _nop_();
     _nop_();
     //_nop_();
     //_nop_();
}
/************ delay_50us ***************/
void delay_50us(void)//延时50us
{
     unsigned char i;
     for(i=0;i<4;i++)
     {
          delay_5us();
     }
}
/******** 延时100us ******************/
void delay_100us(void)//延时100us
{
     delay_50us();
     delay_50us();
}

/************ 延时单位：ms ********************/
void time(unsigned int ucMs)//延时单位：ms
{
     unsigned char j;
     while(ucMs>0){
          for(j=0;j<10;j++) delay_100us();
          ucMs--;
     }
}
```

习题 7

1．为什么要对单片机系统进行扩展？系统扩展主要包括哪些方面？

答：

单片机内部资源有 RAM、I/O 接口、中断、定时器、串行接口等，但在构成实际的硬件系统时，如果单片机自身的资源还是不能满足要求，这时就要进行系统扩展。

单片机外部扩展资源包含：外部 RAM/ROM、键盘、显示、A/D、D/A、I/O 扩展、中断扩展、串行通信、总线驱动、电源监控、看门狗等一些最基本的模块，它们都是大多数单片机应用系统必不可少的关键部分。

详细见主教材 P230~231

2．请概述单片机应用系统中外部扩展资源的种类。

答：

① 外部程序存储器 ROM

当单片机内部程序存储器 ROM 容量无法满足应用系统要求时，需要在外部进行扩展。外部扩展的程序存储器种类主要有 EPROM、EEPROM 和 Flash EEPROM。目前大多数单片机生产厂家都提供大容量 Flash EEPROM 型号的单片机，其存储单元数量都达到了 64KB，能满足绝大多数用户的需要，且价格与 ROMLess 的单片机不相上下。用户没有必要再扩展外部程序存储器。

② 外部数据存储器 RAM

由于单片机的内部数据存储器容量较小，在需要大量数据缓冲的单片机应用系统中（如语音系统、商场收费 POS）仍然需要在外部扩展数据存储器。常用的外部数据存储器有静态随机存储器 RAM6264、RAM62256 和 RAM628128，但随机存储器不具备数据掉电保护特性，许多单片机应用系统采用 Flash EEPROM 作为数据存储器。

③ 并行 I/O 口资源扩展

I/O 口是单片机系统最宝贵的资源之一，单片机的外部扩展将占用大量 I/O 口资源，以 80C51 系列单片机为例，外部扩展占用 P0、P2 口 16 个 I/O 口和 P3 的 2 个 I/O 口，总共损失了至少 18 个 I/O 口。在较为复杂的控制系统（尤其是工业控制系统，如可编程控制器）中，经常需要扩展 I/O 口。常用的 I/O 接口芯片有 74HC 系列锁存器/寄存器、8255 和 8155 等。

④ 键盘和显示器

键盘和显示器提供了用户与单片机应用系统之间的人机界面，用户通过键盘向单片机系统输入数据或程序，而通过显示器用户可以了解单片机系统的运行状态。

⑤ 串行通信接口

单片机通常都提供了一个串行通信接口，且信号为 TTL 电平，为了方便单片机系统与 PC、打印机、其他外部设备等接口，往往需要扩展通用的 RS-232 通信接口。为了实现远距离通信，还要扩展 RS-485 通信接口。常用的 RS-232 接口芯片为 MAX232，常用的 RS-485 接口芯片为 MAX485。

当单片机系统需要更多的串行通信接口时，可以通过串行口芯片扩展，常用的串行口芯片有 8251、8250、16C554 等。

⑥ 模数转换 A/D

A/D 转换接口将外部设备输入的模拟量转换为计算机使用的数字量。常用的 A/D 转换芯片有 ADC0808/0809、ADC0816/0817、ADC1140、ADC71/76、AD574A 等。

⑦ 数模转换 D/A

D/A 转换接口将计算机的数字量转换为外部设备使用的模拟量。常用的 D/A 转换芯片有 DAC0832、DA7520、DAC1208、DAC1230 等。

⑧ 电源监控和硬件看门狗

在电源不稳定或有强大的干扰源时，系统经常会出现“程序跑飞”等异常情况，给系统的开发和实际应用带来极大的不便，严重时会使系统瘫痪，甚至发生工业事故。为此需要使用专用的电源监控复位芯片，人们常把此类电路称为硬件看门狗，当系统电压下降和“程序跑飞”时，它能发出复位信号，保证系统正常工作。常用的电源监控复位芯片有 CSI24C161、DS1232、X5045 等。

3．请简述 8255 芯片的定义、内部结构。

答：

8255 是 Intel 公司生产的可编程并行 I/O 接口芯片，有 3 个 8 位并行 I/O 口，是具有 3 个通道 3 种工作方式的可编程并行接口芯片（40 引脚）。其各口功能可由软件选择，使用灵活，通用性强。8255 可作为单片机与多种外设连接时的中间接口电路。8255 内部结构如图 7.5 所示。

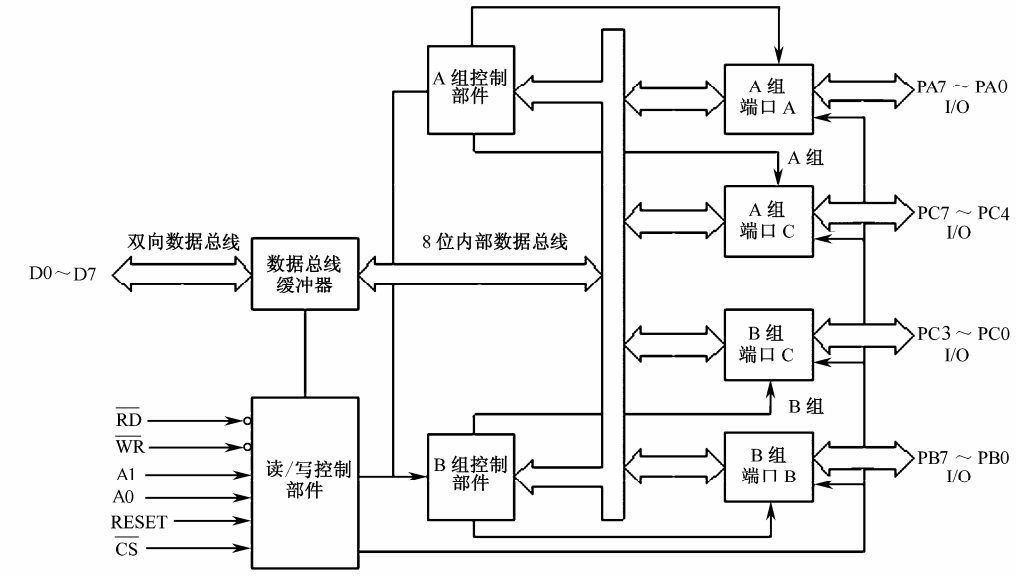


图 7.5　8255 内部结构

8255 作为主机与外设的连接芯片，必须提供与主机相连的 3 个总线接口，即数据线、地址线、控制线接口。同时必须具有与外设连接的接口 A、B、C 口。由于 8255 可编程，所以必须具有逻辑控制部分，因而 8255 内部结构分为 3 个部分：与 CPU 连接部分、与外设连接部分、控制部分。

8255 内部结构：由 4 部分组成：

（1）数据端口 A、B、C（称为 PA、PB、PC）：

端口 A 具有 1 个输入锁存器和 1 个输出锁存器/缓冲器。作输入或输出时，数据均受到锁存。所以可用作双向数据传输。

端口 B 只有 1 个数据输入缓冲器和 1 个输出锁存器/缓冲器。只能工作在输入或输出方式，不具有双向功能。

端口 C 每个 4 位的端口对应 1 个输入缓冲器和 1 个输出锁存器/缓冲器。

C 口只有在端口 A、B 都工作在简单 I/O 方式时，才具有 2 个 4 位的 I/O 功能。在更多情况下是作为 A、B 口的应答联络信号，分别为端口 A 和端口 B 提供控制信号和状态信号。

（2） A 组控制和 B 组控制

这两组控制电路接收 CPU 输出的控制字，以及读/写控制逻辑电路命令，决定两组端口的工作方式和读/写操作。

A 组控制电路控制端口 A 和端口 C 的高 4 位（PC7～PC4）的工作方式和读/写操作。

B 组控制电路控制端口 B 和端口 C 的高 4 位（PC3～PC0）的工作方式和读/写操作。

（3）读/写控制逻辑电路

接收及来自系统总线的信号 A1、A0（在 8086 系统中为 A2、A1）和控制总线的信号 ，将这些信号进行组合，以完成对数据、状态信息和控制信息的传输。

（4）数据总线缓冲器

双向三态的 8 位数据缓冲器，8255A 通过它与系统数据总线相连。用来传送输入/输出数据、CPU 发给 8255A 的控制字。

4．某单片机系统应用 8255 扩展 I/O 口，设其 A 口为方式 1 输入，B 口为方式 1 输出，C 口余下的引脚用于输出，试写出其方式控制字。

答：

根据 P238 图 7.8 可确定！

方式控制字：10110100

5．试分析 8255 实际可能有的各种置位/复位控制字。

答：

8255 的 C 口可进行位操作，即可对 8255 C 口的每一位进行置位或清 0 操作，该操作是通过设置 C 口置/复位字实现的。C 口置/复位字共 8 位，各位含义如图 7.9 所示。

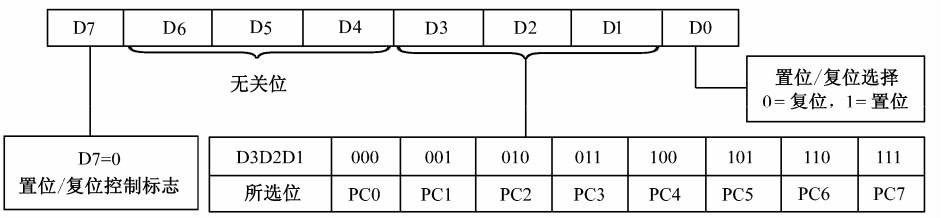


图 7.9　C 口置/复位控制字

由于 8255 的工作方式选择字与 C 口置/复位字共用一个控制寄存器，故特别设置 D7 为标识位，D7=0 表示控制字为 C 口置/复位字，D7＝1 表示控制字为 8255 工作方式选择字。D6D5D4 不用，常取 000。D3D2D1 为 C 口 8 个引脚 PC0～PC7 的选择位，D3D2D1＝000 选择 PC0，D3D2D1=001 选择 PC1，……，D3D2D1=111 选择 PC7。D0 为置位或清 0 选择位，D0=0 表示由 D3D2D1 选择的位清 0，D0=1 表示由 D3D2D1 选择的位置 1。C 口置/复位字必须输入 8255 控制寄存器。

6．什么是键盘的抖动？为什么要对键盘进行消抖动处理？如何消除键盘的抖动？

答：

在图 7.15 中，当按键 S 未被按下（即断开）时，P1.1 输入为高电平，S 闭合后，P1.1 输入为低电平。通常的按键所用的开关为机械弹性开关，当机械触点断开、闭合时，电压信号波形如图 7.15（b）所示。由于机械触点的弹性作用，一个按键开关在闭合时不会马上稳定地接通，在断开时也不会马上断开，因而在闭合及断开的瞬间均伴随有一连串的抖动。抖动时间的长短由按键的机械特性决定，一般为 5～10ms。这种抖动对于人来说是感觉不到的，但对单片机来说，则是完全可以感应到的，因为单片机的处理速度在微秒级。假如对按键不进行消抖处理，如通过键盘输入一个 1，单片机程序却已执行了多次输入 1 按键处理程序，其结果是认为输入了若干个 1。

图 7.15　按键输入与抖动波形

键抖动会引起一次按键被误读多次，为了确保单片机对键的一次闭合仅做一次处理，必须去除键抖动，在键闭合稳定时取键状态，并且必须判别到键释放稳定后再进行处理。按键的抖动，可用硬件或软件两种方法消除。

通常在键数较少时，可用硬件方法消除键抖动。RS 触发器为常用的硬件去抖电路，但单片机系统中常用软件法。软件消抖法很简单，如图 7.15 所示，就是在单片机获得 P1.1 口为低的信息后，不是立即认定按键已被按下，而是延时 10ms 或更长一些时间后再次检测 P1.1 口，如果仍为低，说明 S 的确按下了，这实际上是避开了按键按下时的抖动时间。而在检测到按键释放后（P1.1 为高）再延时 5～10ms，消除后沿的抖动，再对键值处理。不过一般情况下，通常不对按键释放的后沿进行处理，实践证明，也能满足一定的要求。当然，实际应用中，对按键的要求也是千差万别的，要根据不同的需要编制处理程序，以消除键抖动为原则。

7．单片机应用系统中有哪几种键盘类型？

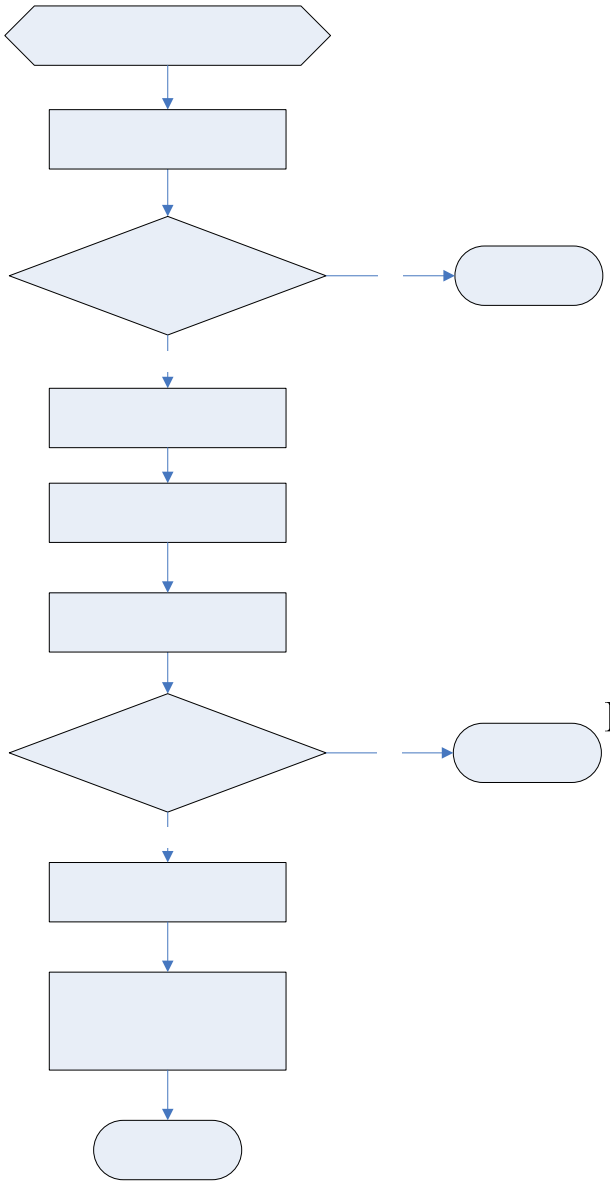
答：

键盘是一组按键的集合，键是一种常开型按钮开关，平时（常态）键的两个触点处于断

开状态，按下键时它们才闭合（短路），如图 7.15 所示。键盘分为编码键盘和非编码键盘。按键的识别由专用的硬件译码实现，并能产生键编号或键值的称为编码键盘，如 BCD 码键盘、ASCII 码键盘等，而缺少这种键盘编码电路要靠自编软件识别的键盘称为非编码键盘。在单片机组成的电路系统及智能化仪器中，用得更多的是非编码键盘，本节只讨论非编码键盘。

8. 对于行列矩阵形式非编码键盘中按键的识别通常采用两步扫描判别法。以图 7.17 所示的 4x4 键盘为例，画出两部扫描判别法识别按键流程图。

答



9. 编写一个程序，单片机外接 4x4 阵列式按键键盘和蜂鸣器，根据按键键值的不同是蜂鸣器响相应的次数。

10．简述 LED 显示器的静态与动态显示原理。

答：

LED 显示器是由发光二极管显示字段的显示器件，也可称为数码管。单片机系统中通常使用 8 段 LED 数码显示器，其外形及引脚如图 7.18（a）所示，由图可见 8 段 LED 显示器由 8 个发光二极管组成。其中 7 个长条形的发光二极管排列成“日”字形，另一个圆点形的发光二极管在显示器的右下角作为显示小数点用，其通过不同的组合可用来显示各种数字，包括 A～F 在内的部分英文字母和小数点“.”等字样。

LED 显示器有两种不同的形式：一种是 8 个发光二极管的阳极都连在一起，称为共阳极 LED 显示器；另一种是 8 个发光二极管的阴极都连在一起，称为共阴极 LED 显示器。如图 7.18（b）所示。

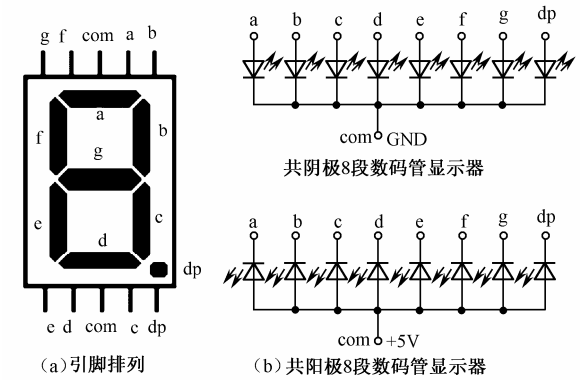


图 7.18　8 段 LED 数码显示器

共阴和共阳结构的 LED 显示器各笔画段名和安排位置是相同的，当二极管导通时，相应的笔画段发亮，由发亮的笔画段组合从而显示各种字符。8 个笔画段 dpgfedcba 对应于 1B（8 位）的 D7、D6、D5、D4、D3、D2、D1、D0，于是用 8 位二进制码就可以表示要显示字符的字形代码。例如，对于共阴极 LED 显示器，当公共阴极接地（为零电平），而阳极 dpgfedcba 各段为 01110011 时，显示器显示“P”字符，即对于共阴极 LED 显示器，“P”字符的字形码是 0x73。如果是共阳极 LED 显示器，公共阳极接高电平，显示“P”字符的字形代码应为 10001100（0x8C）。这里必须注意的是：很多产品为了方便接线，常不按规定的方法去对应字段与位的关系，这时字形码就必须根据接线自行设计了。

LED 显示器的显示方法有静态显示与动态显示两种。

数码管工作在静态显示方式时，共阴极（共阳极）的公共端 COM 连接在一起接地（电源）。

LED 动态显示的基本做法在于分时轮流选通数码管的公共端，使得各数码管轮流导通，在选

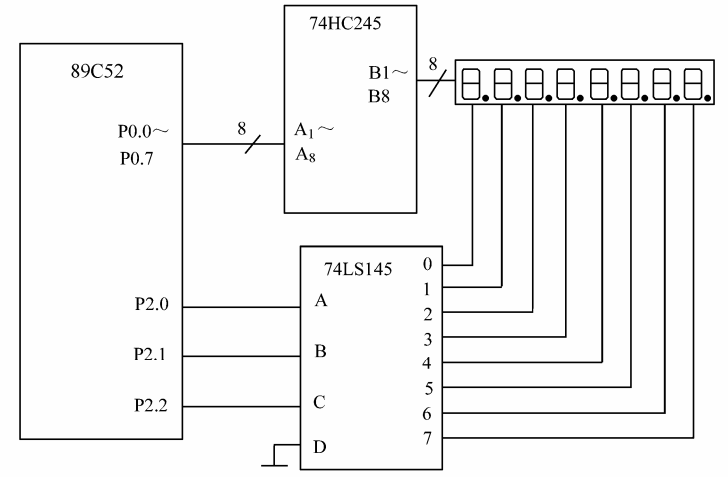
动态扫描显示接口是单片机系统中应用最为广泛的一种显示方式。其接口电路是把所有显示器的 8 个笔画段 a～dp 同名端并联在一起，而每个显示器的公共极 COM 各自独立地受 I/O 线控制，CPU 向字段输出口送出字形码时，所有显示器由于同名端并连接收到相同的字形码，但究竟是哪个显示器亮，则取决于 COM 端，而这一端是由 I/O 控制的，所以就可以自行决定何时显示哪一位了。而所谓动态扫描是指采用分时的方法，轮流控制各个显示器的 COM 端，使各个显示器轮流点亮。

在轮流点亮扫描过程中，每位显示器的点亮时间是极为短暂的（约 1ms），但由于人的视觉暂留现象及发光二极管的余辉效应，尽管实际上各位显示器并非同时点亮，但只要扫描的速度足够快，给人的印象就是一组稳定的显示数据，不会有闪烁感。

11. 图 7.19 中“串行口扩展 8 位 LED 显示器静态驱动电路”使用单片机的 RXD 和 TXD 引脚发送移位时钟和移位数据给 74HC164，如果单片机的 RXD 和 TXD 被占用，改用 P1.6 和 P1.7 发送移位时钟和移位数据给 74HC164，请编程实现。

```
/*******************发送串行数据到 74HC164**********************/
sbit DIN = P1^7;//移位寄存器数据引脚
sbit CLK = P1^6;//移位寄存器时钟引脚
void SendData(unsigned char shiftData)//发送串行数据到 74HC164
{
    uchar i;
    for(i = 0;i < 8;i++)
    {
        CLK = 0;
        DIN = shiftData & 0x01;
        CLK = 1;
        shiftData = shiftData>>1;
    }
}
/*******************更新 LED 数码管显示器**********************/
void DataToLed(uchar writeData)//更新 LED 数码管显示器
{
code unsigned char LEDValue[10]={0xfc,0x60,0xda,0xf2,0x66,0xb6,0xbe,0xe0,0xfe,0xf6};//共阴极数码编码表
unsigned char writeData_h,writeData_l;
    writeData_h = writeData/10;
    writeData_l = writeData%10;
    SendData( LEDValue[writeData_l] );
    SendData( LEDValue[writeData_h] );
    SendData( LEDValue[writeData_l] );
    SendData( LEDValue[writeData_h] );
}
```

12. 动态显示是指每隔一段时间循环点亮每隔 LED 数码管，每次只有一个 LED 点亮。根据人眼的视觉暂留效应，当循环点亮的速度很快时，可以认为各个 LED 是稳定现实的。根据这个原理，设一个 8 位 LED 数码管动态显示的电路和程序。



13．串行接口的芯片有哪些特点？为什么说串行接口芯片能节约单片机资源？

答：

串行接口的芯片以其使用简单、接口容易、与微机连线少等特点，在单片机应用系统，尤其是手持式信息设备中得到广泛应用。串行接口芯片一般只需 2～3 根线即可和单片机组成系统，可节约单片机资源 I/O。

14．简述 Flash EEPROM 存储器的特点。当 Flash EEPROM 存储器容量大于 64KB 时，在单片机系统如何能访问到所有空间？

答：

由于闪速存储器 Flash EEPROM 结合了 EEPROM 可电擦除的灵活性和 EPROM 大容量、低成本的优点，在电子词典、商场 POS 收费终端、考勤机等单片机应用系统中得到广泛的应用，随着制造工艺和材料的改进，闪速存储器与 EPROM 和 EEPROM、SRAM 及 DRAM 等存储器相比优势越来越明显。以 39SF040 为例说明齐特点。

39SF040 是一个容量为 512KB 的 CMOS 快速闪存，可块擦除、字节编程的 Flash EEPROM，引脚与 EPROM、EEPROM 兼容。它非常适合用在程序和数据重复写入的场合，对这种系统的应用来说，28SF040 可提高性能和稳定性。

1．39SF040 存储器的主要特点

- 容量 512KB
- 可全片或分扇区（每扇区 256B）擦除
- 字节写入时间 35μs
- 字节读出时间 90ns
- 页面擦除时间 2ms
- 单 5V 电压编程
- 可擦写次数达 100 000 次
- 数据保存期 100 年
- 读写状态电流 15mA
- 备用状态电流 5μA

● 可进行硬件及软件写保护

当 Flash EEPROM 存储器容量大于 64KB 时，在单片机系统中需增加高位地址线访问到所有空间。

1．使用单片机的 I/O 口控制高位地址线

图 7.13 中，单片机采用 89C52，39SF040 占有 CPU 的 0000～FFFFH 共 64KB 的地址空间，39SF040 共有 512KB 的空间，用 P1 口的 P1.2～P1.0 控制 39SF040 高位地址线，把 39SF040 分为 2^3 = 8 页（8×64KB= 512KB）。在对 39SF040 进行读写操作之前，首先确定页地址。当 P1.2～P1.0＝000 时选中第 0 页，对应 39SF040 的绝对地址空间为 00000～0FFFFH；当 P1.2～P1.0＝001 时选中第 1 页，对应 39SF040 的绝对地址空间为 10000～1FFFFH；……；当 P1.2～P1.0 = 111 时选中第 7 页，对应 39SF040 的绝对地址空间为 70000～7FFFFH。图中 74HC573 为地址锁存器，其输出为单片机系统低位地址线 A0～A7。

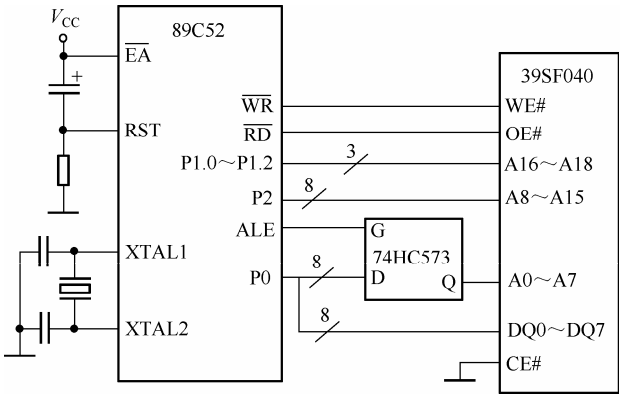


图 7.13　使用单片机的 I/O 口控制高位地址线

2．使用单片机的扩展 I/O 口控制高位地址线

图 7.14 中单片机采用 89C52，74HC138 译码给 39SF040 和 74HC574 的片选。当[A15，A14，$\overline{WR}$ & $\overline{RD}$]＝[0, 0, 0]时，$\overline{Y0}$＝0 选中 39SF040 进行读/写操作，对应 CPU 的地址空间为 0000～3FFFH 共 16KB。当[A15, A14, $\overline{WR}$ & $\overline{RD}$]＝[1, 1, 0]时，$\overline{Y6}$＝0 选中 74HC574，对应 CPU 的地址为 C000（实际上在此 C000～FFFFH 范围），执行写操作，将 D0～D4 打到 74HC574 的输出端 Q0～Q4 锁存，作为 39SF040 的高位地址线 A14～A18。此处需注意，因为 74HC574 为 CMOS 电路，未用的输入端要做接 VCC 或 GND 的处理。

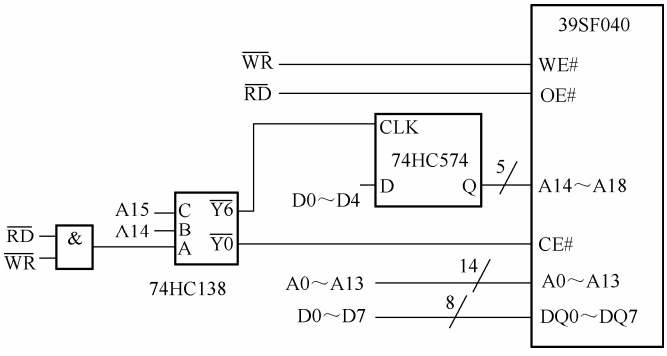


图7.14　使用单片机的扩展I/O口控制高位地址线

图7.14中，39SF040占有CPU的 0000～3FFFH共16KB的地址空间，39SF040共有512KB的空间，用74HC574的输出端Q4～Q0控制39SF040的高5位地址线A18～A14，把39SF040分为2^5=32页（32×16K=512K）。在对39SF040进行读写操作之前，首先确定页地址。当Q4～Q0=00000时选中第0页，对应39SF040的绝对地址空间为0000～3FFFH；当Q4～Q0=00001时选中第1页，对应39SF040的绝对地址空间为4000～7FFFH；……；当Q4～Q0=00011时选中第3页，对应39SF040的绝对地址空间为C000～FFFFH；……；当Q4～Q0=11111时选中第31页，对应39SF040的绝对地址空间为7C000～7FFFFH。图7.14中的74HC138、74HC574和与门也可由一片可编程逻辑器件PLD实现。

15. 简述E^2PROM存储器的特点。当Flash E^2PROM存储器容量大于64KB时，在单片机系统如何能访问到所有空间？

答：

串行E^2PROM是可在线电擦除和电写入的存储器，具有体积小、接口简单、数据保存可靠、可在线改写、功耗低等特点，而且为低电压写入。

16. 编写一个写I2C总线接口EEPROM的程序，并提供校验功能，当校验失败的时候提供报警，可用串口精灵发送要写入的数据给单片机。

17. 简述RS-232串行通信接口的工作原理。RS-232高低电平定义与TTL/COMS的高低电平定义有什么区别

答：

电气特性

EIA-RS-232C 对电器特性、逻辑电平和各种信号线功能都作了规定。

在TxD和RxD上：

逻辑1(MARK)=-3V～-15V

逻辑0(SPACE)=+3～+15V

在RTS、CTS、DSR、DTR和DCD等控制线上：

信号有效（接通，ON状态，正电压）=+3V～+15V

信号无效（断开，OFF状态，负电压)=-3V～-15V

以上规定说明了RS-232C标准对逻辑电平的定义。对于数据（信息码）：逻辑“1”（传号）的电平低于-3V，逻辑“0”（空号）的电平高于+3V；对于控制信号；接通状态（ON）即信号有效的电平高于+3V，断开状态(OFF)即信号无效的电平低于-3V，也就是当传输电平的绝

对值大于 3V 时，电路可以有效地检查出来，介于-3～+3V 之间的电压无意义，低于-15V 或高于+15V 的电压也认为无意义，因此，实际工作时，应保证电平在±(3～15)V 之间。

EIA RS-232C 与 TTL 转换：EIA RS-232C 是用正负电压来表示逻辑状态，与 TTL 以高低电平表示逻辑状态的规定不同。因此，为了能够同计算机接口或终端的 TTL 器件连接，必须在 EIA RS-232C 与 TTL 电路之间进行电平和逻辑关系的变换。实现这种变换的方法可用分立元件，也可用集成电路芯片。目前较为广泛地使用集成电路转换器件，如 MC1488、SN75150 芯片可完成 TTL 电平到 EIA 电平的转换，而 MC1489、SN75154 可实现 EIA 电平到 TTL 电平的转换。MAX232 芯片可完成 TTL←→EIA 双向电平转换。

18. 简述 RS-485 串行接口标准的特点

答：

1. RS-485 的电气特性，逻辑 1 以两线间的电压差为+2～+6V 表示；逻辑 0 以两线间的电压差-2～-6V 表示。接口信号电平比 RS-232C 降低了，这样就不易损坏接口电路的芯片，且该电平与 TTL 电平兼容，可方便与 TTL 电路连接。

2. RS-485 的数据最高传输率达 10Mbps

3. RS-485 接口采用平衡驱动器和差分接收器的组合，抗共模干扰能力增强，即抗噪声干扰性好。

4. RS-485 接口的最大传输距离标准为 1.2km，实际上可达 3KM。

19. RS-485 输入/输出信号为何不能同时进行？串行接口使用 RS-485 时，在程序中应注意什么？

答：

RS-485 串行总线接口标准以差分平衡方式传输信号，具有很强的抗共模干扰的能力，允许一对双绞线上一个发送器驱动多个负载设备。工业现场控制系统中一般都采用该总线标准进行数据传输，用户在开发一般的单片机应用系统时，利用单片机本身所提供的简单串行接口，加上总线驱动器如 MAX485 等组合成简单的 RS-485 通信网络。

RS-485 采用一对双绞线，输入/输出信号不能同时进行（半双工），MAX485 芯片的发送和接收功能转换是由芯片的 $\overline{RE}$ 和 DE 端控制的。$\overline{RE}$ =0 时，允许接收；$\overline{RE}$ =1 时，接收端 R 高阻。DE=1 时，允许发送；DE=0 时，发送端 A 和 B 高阻。在单片机系统中常把 $\overline{RE}$ 和 DE 接在一起用单片机的一个 I/O 线控制收发。

在程序中应注意空闲时 MAX485 处于接收状态。

20. 设计一个单片机程序，接收计算机通过串口发送的数据流，将其中的小写字符转换为大写字符，并会送给计算机

习题 9

1. 在 Keil uVision2 中支持两种模式的 RTX-51，即完全模式的 RTX-51 FULL 和最小模式的

RTX-51 Tiny

2. RTX-51 内核将有效的 CPU 时间分为时间片，然后将时间片合理地分配给多个任务。程序中每个任务执行预先定义好的时间片，然后切换到另一个任务的时间片上执行。

3. RTX-51 Tiny 作为 RTX-51 FULL 的一个子集，主要运行在没有外部数据存储器的 51 单片机系统中。

4. RTX-51 实时多任务操作系统运行于 80C51 硬件平台，其根据 80C51 的特点进行了特定的优化和限定。

5. 等待函数 os_wait 主要用于暂停当前任务，等待一个或多个事件发生。

6. 程序不要求有 main（）主函数。RTX-51 内核自动从任务（A）来开始执行
A. 0　　B. 1　　C.2　　D.main（）

7.如下哪些是 RTX-51 的功能（多选）（AC）
A.合理划分时间片　　B.CAN 通信　　C.4 种任务调度　　D.BITBUS 通信

8.RTX-51 Tiny 系统需要使用定时器（A）
A.0　　B.1　　C.0 或 1 均可以　　D.0 和 1 都需要

9.RTX-51 FULL 中优先级可以设置为（多选）（ABCD）
A.0　　B.1　　C.2　　D.3

10.简述 RTX-51 多任务系统与普通多任务循环的区别
答：
普通循环任务中，主函数循环依次执行单独的操作或任务。这种多任务循环当其中的某些任务需要等待较长事件才能返回的时候，就会造成系统的阻塞，需要较长时间才能执行下一个任务。
RTX-51 允许“准并行”同时执行几个任务。各个任务并非持续运行，CPU 执行事件被划分成若干时间片，每一个任务在预先定义好的时间片内得以执行。时间片到是正在执行的任务挂起，并使另一个任务开始执行。

11.简述 RTX-51 多任务系统的不同任务调度方式。
答：
RTX-51 支持 2 种不同的任务调度方式：循环任务切换、协作任务切换
循环法允许并行地执行若干任务。任务并非真的同时执行，而是分时间片执行的。由于时间片很短（几毫秒），看起来就好像任务在同时执行。

可以用 os_wait 或 os_switch_task 让 RTX-51 TINY 切换到另一个任务而不是等待任务的时间片用完。os_wait 函数挂起当前的任务（使之变为等待态）直到指定的事件发生（接着任务变为就绪态）。在此期间，任意数量的其他任务可以运行。

12.编写一个程序，包含三个任务，一个任务监控 P0 端口，另一个任务向 P2 端口写数据，第三个任务通过串口输出字符串。

13.参见例 9.2，要求红绿灯持续时间也可以设置，请编程实现。

14.参见例 9.2，将禁止通行按钮由 P2.7 改接到 INT0（P3.2），在 INT0 的终端服务中发送信号给 BLINKING，当按下按钮则禁止通行 30 秒，之后自动回复正常的交通灯控制，请编程实现。

15. 将第 8 张“单片机应用系统设计实例”电梯控制器的控制软件改为用 RTX-TINY 多任务实现。