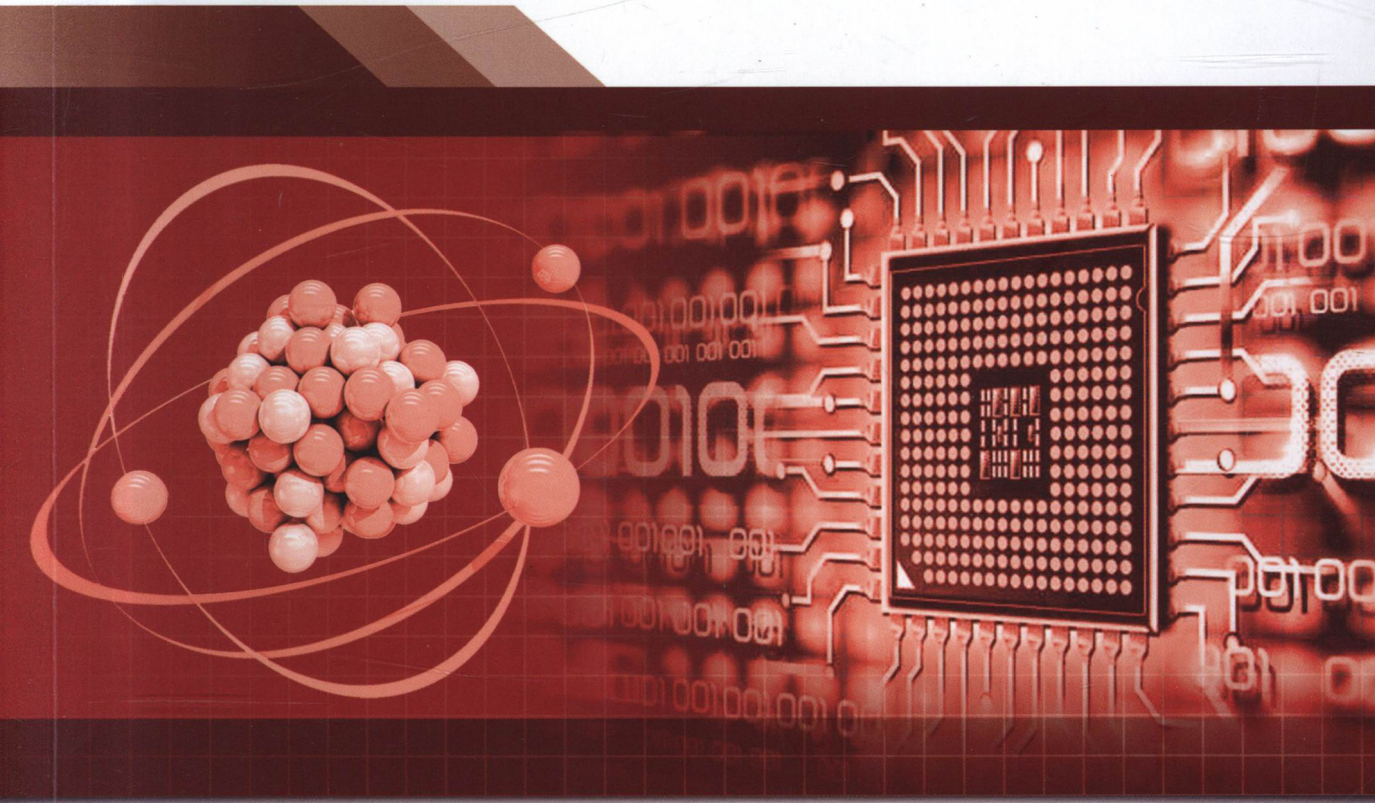
北京劳动保障职业学院国家骨干校建设资助项目

单片机控制技术及应用



李广军　纪娜 ◎ 主编

机械工业出版社
CHINA MACHINE PRESS

北京劳动保障职业学院国家骨干校建设资助项目

单片机控制技术及应用

主　编　李广军　纪　娜
参　编　程晓辉　蒙　娟　宁玉红
　　万　捷　孔　杰　赵海静

机 械 工 业 出 版 社

本书以80C51单片机为例，讲述了单片机控制技术及应用。内容包括单片机基础知识、结构与原理、指令系统、汇编语言设计、中断系统、定时器/计数器、串行通信、单片机的并行扩展技术及单片机A-D、D-A转换接口技术等。

本书可作为高职高专电子、通信、电气、机电专业单片机课程教材，也可供从事单片机应用的工程技术人员参考，非常适合单片机爱好者自学使用。

图书在版编目（CIP）数据

单片机控制技术及应用/李广军等主编. —北京：机械工业出版社，2013.11
北京劳动保障职业学院国家骨干校建设资助项目
ISBN 978-7-111-44921-8

Ⅰ.①单… Ⅱ.①李… Ⅲ.①单片微型计算机－计算机控制－高等职业教育－教材 Ⅳ.①TP368.1

中国版本图书馆CIP数据核字（2013）第282894号

机械工业出版社（北京市百万庄大街22号 邮政编码100037）
策划编辑：罗 莉 责任编辑：罗 莉
版式设计：常天培 责任校对：陈延翔
封面设计：赵颖喆 责任印制：李 洋
北京华正印刷有限公司印刷
2014年1月第1版第1次印刷
184mm×260mm·12印张·295千字
0001—3000册
标准书号：ISBN 978-7-111-44921-8
定价：39.00元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

电话服务	网络服务
社 服 务 中 心：(010)88361066	教材网：http://www.cmpedu.com
销 售 一 部：(010)68326294	机工官网：http://www.cmpbook.com
销 售 二 部：(010)88379649	机工官博：http://weibo.com/cmp1952
读者购书热线：(010)88379203	**封面无防伪标均为盗版**

前　言

本书从社会需要和高等职业教育的特点出发，确定了教学目标：掌握单片机基本结构和原理、MCS-51 系列单片机指令系统、汇编语言程序设计方法、单片机存储系统、输入/输出接口电路、定时器/计数器、中断系统、串行通信技术及模拟量输入/输出接口技术等。

本书以 80C51 单片机为例，讲述了单片机控制技术及应用。内容包括单片机基础知识、结构与原理、指令系统、汇编语言设计、中断系统、定时器/计数器、串行通信、单片机的并行扩展技术及单片机 A-D、D-A 转换接口技术等。

根据高职高专教育注重培养学生实践动手能力的要求，本书以单片机应用为主线，理论与实践紧密结合，注重对单片机资源应用方法的总结，使初学者更容易理解和掌握；每章包含相应的例题，以便使学生更深入学习理论知识，加强实际操作的练习，并安排了一定量的习题与思考题，方便读者练习和提高。

本书可作为高职高专电子、通信、电气、机电专业单片机课程教材，也可供从事单片机应用的工程技术人员参考，非常适合单片机爱好者自学使用。

本书由北京劳动保障职业学院机电工程系教师团队编写。李广军编写第 1 章和第 2 章；纪娜编写第 3 章；程晓辉编写第 4 章；蒙娟编写第 5 章；宁玉红编写第 6 章；万捷编写第 7 章；孔杰、赵海静编写第 8 章。全书由李广军统稿。

由于作者水平所限，书中可能会存在某些错误和不妥之处，恳请广大读者批评指正。

作　者

目　录

第 1 章　计算机运算基础

【学习目的】

1. 了解计算机中的数制。
2. 掌握数制之间的转换。
3. 熟悉计算机中数的表示方法及数的表示形式。

电子计算机是一种能对信息加工处理的机器，具有记忆、判断和运算的能力。

1.1　计算机中的数制及数制的转换

十进制是人们生活中普遍使用的计数制，但计算机都是以二进制形式进行算术运算和逻辑运算操作的，微型计算机也不例外。因此，对于用户在键盘上输入的十进制数字和符号命令，微型计算机必须先把它们转换成二进制形式进行识别、运算和处理，然后再把二进制形式的运算结果转换为人们容易识别的十进制数字和符号，并在显示器上显示出来。

上述过程都是由计算机自动完成的，在微型计算机中除了用到二进制和十进制的数制外，经常会用到八进制和十六进制的计数制，为了使读者弄清计算机中数制转换的原理，先对计算机中常用的数制和数制的转换进行讨论。

1.1.1　计算机中的数制

所谓数制是指数的制式，是人们利用符号计数的一种科学方法。数制有很多种，微型计算机中常用的数制有十进制、二进制、八进制和十六进制等。

1. 十进制（Decimal）

十进制是人们生活中普遍使用的数制，它用 0、1、2、…、9 这十个数来描述。十进制数的主要特点如下：

① 它有 0 ~ 9 十个不同的数，这是构成所有十进制数的基本符号。

② 它是逢 10 进位的。十进制数在计数过程中，当它的某位计满 10 时就要向它邻近的高位进 1。

2. 二进制（Binary）

二进制是在计算机系统中使用的数制，它用 0、1 这两个数来描述。二进制数的主要特点如下：

① 任何二进制数都是由 0、1 这两个数组成。

② 二进制数的基数为 2，它遵循逢 2 进 1 的进位计数原则。

3. 十六进制（Hexadecimal）

十六进制是计算机指令代码和数据以及软件工具中经常使用的数制，它用 0、1、…、9 和 A、B、…、F 这十六个数和字母来描述。十六进制数的主要特点如下：

① 任何一个十六进制数都是由0、1、…、9和A、B、…、F这十六个数和字母构成。

② 十六进制数的基数为16，它遵循逢16进1的进位计数原则。

为方便起见，现将部分十进制、二进制和十六进制数的对照表列于表1-1。

表1-1 部分十进制、二进制和十六进制数的对照表

整数			小数		
十进制	二进制	十六进制	十进制	二进制	十六进制
0	0000	0	0	0	0
1	0001	1	0.5	0.1	0.8
2	0010	2	0.25	0.01	0.4
3	0011	3	0.125	0.001	0.2
4	0100	4	0.0625	0.0001	0.1
5	0101	5	0.03125	0.00001	0.08
6	0110	6	0.015625	0.000001	0.04
7	0111	7			
8	1000	8			
9	1001	9			
10	1010	A			
11	1011	B			
12	1100	C			
13	1101	D			
14	1110	E			
15	1111	F			
16	10000	10			

1.1.2 数制间的转换

在计算机中都是以二进制数进行算术运算和逻辑运算操作的，而人们习惯使用十进制数，计算机会自动对不同数制的数进行转换。下面学习不同数制的数是如何转换的。

1. 二进制数和十进制数间的转换

（1）二进制数转换成十进制数

二进制数转换成十进制数只要把欲转换数按权展开后相加即可。例如：

$$11010.01\text{B} = 1\times2^4 + 1\times2^3 + 1\times2^1 + 1\times2^{-2} = 26.25$$

（2）十进制数转换成二进制数

本转换过程是上述转换过程的逆过程，但十进制整数和小数转换成二进制整数和小数的方法是不相同的，现分别进行介绍。

1）十进制整数转换成二进制整数

常用的是“除2取余法”。用2连续去除要转换的十进制数，直到商小于2为止，然后把各次余数按最后得到的为最高位、最早得到的为最低位，依次排列起来所得到的数便是所求的二进制数。

【例 1-1】　100D = _____B　　余数

2 | 100　　0（最低位）

2 | 50　　0

2 | 25　　1

2 | 12　　0

2 | 6　　0

2 | 3　　1

2 | 1　　1（最高位）

0

答案：100D = 1100100B

2）十进制小数转换成二进制小数

通常采用“乘 2 取整法”。用 2 连续去乘要转换的十进制小数，直到所得积的小数部分为 0 或满足所需精度为止，然后把各次整数按最先得到的为最高位、最后得到的为最低位，依次排列起来所对应的数便是所求的二进制小数，现结合实例加以介绍。

【例 1-2】　0. 625D = _____B

乘 2 取整　　整数部分

0. 625

× 2

1. 250　　1

0. 25

× 2

0. 50　　0

× 2

1. 0　　1

答案：0. 625D = 0. 101B

2. 二进制数与十六进制数的转换

1）二进制数转换为十六进制数

采用四位二进制数合成为一位十六进制数的方法，以小数点为界分成左侧整数部分和右侧小数部分；整数部分从小数点开始，向左每 4 位二进制数一组，不足 4 位在数的前面补 0；小数部分从小数点开始，向右每 4 位二进制数一组，不足 4 位在数的后面补 0，然后每组用十六进制数码表示，并按序相连即可。

【例 1-3】　把 111010. 011110B 转换为十六进制数。

0011　1010.　0111　1000 = 3A. 78H

3　A　7　8

2）十六进制数转换为二进制数

将十六进制数的每位分别用 4 位二进制数码表示，然后它们按序连在一起即为对应的二进制数。

【例 1-4】 把 2BD4H 和 20.5H 转换为二进制数

2BD4H = 0010 1011 1101 0100B

20.5H = 0010 0000.0101B

3. 十六进制数与十进制数的转换

（1）十六进制数转换成十进制数

十六进制数转换成十进制数的方法和二进制数转换成十进制数的方法类似，将十六进制数按权展开后求和即得到十进制数。

【例 1-5】 将十六进制数 3DF2H 转换成十进制数。

$3DF2H = 3 \times 16^3 + 13 \times 16^2 + 15 \times 16^1 + 2 \times 16^0 = 15858$

（2）十进制数转换成十六进制数

① 十进制整数转换成十六进制整数与十进制整数转换成二进制整数类似，十进制整数转换成十六进制整数可以采用“除 16 取余法”。用 16 连续去除要转换的十进制整数，直到商数小于 16 为止，然后把各次余数按逆得到顺序排列起来所得的数，便是所求的十六进制数。

② 十进制小数转换成十六进制小数的方法类似十进制小数转换成二进制小数，常采用“乘 16 取整法”。把欲转换的十进制小数连续乘以 16，直到所得乘积的小数部分为 0 或达到所需精度为止，然后把各次整数按相同的得到顺序排列起来所得的数，便是所求的十六进制小数。

1.2 计算机中数的表示方法

在讨论计算机如何对有符号数或无符号数进行运算和处理之前，先弄清计算机中数的表示方法是十分必要的。在计算机中，小数和整数都是以二进制形式表示的，但对小数点通常有定点和浮点两种表示方法。小数点采用定点表示法的称为定点机，采用浮点表示法的叫做浮点机。

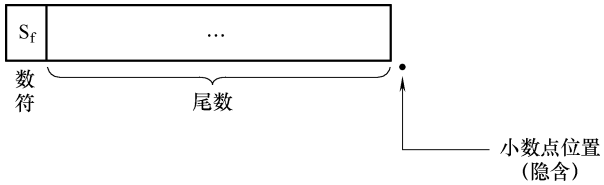
1.2.1 定点机中数的表示方法

在定点计算机中，二进制数的小数点位置是固定不变的，小数点位置可以固定在数值位之前，也可以固定在数值位之后。前者称为定点小数计算机，后者叫做定点整数计算机。

在理论和习惯上，小数点固定在中间位置比较合适，但因为它所能表示的数既有整数又有小数部分，会给数在数制间替换带来麻烦，故这种方法通常并不为计算机设计师们所采用。

1. 定点整数表示法

在采用定点整数表示法的计算机中，小数点位置被固定在数值位之后。因此，这种计算机在实际运算前应先把参加运算的数（二进制形式）按适当比例替换成纯整数，并在运算后把结果操作数按同一比例还原后输出。设 N 为某一定点二进制整数，其表示形式为



其中，S_f 为数符，$S_f=0$ 表示 N 为正数，$S_f=1$ 表示 N 为负数。

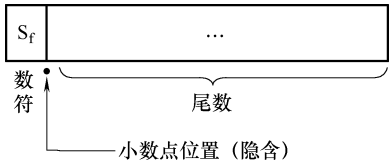
数的表示形式在大多数计算机中都是采用定点整数法，MCS-51 单片机也是一种定点整数计算机。因此，MCS-51 单片机只能对二进制整数进行直接运算和处理，它在遇到二进制小数时，必须把该小数按比例扩大成二进制整数后进行处理，并在处理完后再按同样比例缩小后进行输出。

定点整数表示法的优点是，运算规则简单，它所能表示的数的范围没有相同位数的浮点法大。例如，一个 16 位的二进制定点整数 N，若它的 S_f 占 1 位、尾数占 15 位，则它所能表示的原码数的范围为

$$|N|\leqslant 11\cdots11=1\ 00\cdots00-1=2^{15}-1$$，近似形式为 $-2^{15}\leqslant N2^{15}$

2. 定点小数表示法

在采用定点小数表示法的计算机中，小数点的位置被固定在数值位之前。因此，这种计算机在实际运算前应首先把参加运算的二进制整数按适当比例替换成纯小数，并在运算结束后把结果操作数（纯小数）按同样比例逆替换后输出。设 N 为定点小数，其表示形式为



其中，S_f 为数符，$S_f=0$ 表示 N 为正数，$S_f=1$ 表示 N 为负数。

定点小数表示法的优点是，运算规则简单，但它所能表示的数的范围较小。例如，一个 16 位的二进制小数 N，若它的 S_f 占 1 位、尾数占 15 位，则它所能表示的原码数范围为

$$|N|\leqslant 0.11\cdots11=1-0.00\cdots01=1-2^{-15}$$

即 $-(1-2^{-15})\leqslant N\leqslant(1-2^{-15})$

1.2.2　浮点机中数的表示方法

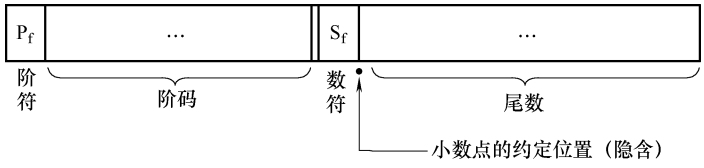
在采用浮点表示的二进制数中，小数点位置是浮动的、不固定的。

通常，任意一个二进制数都可以写为

$$N=2^P\times S$$

式中，S 为二进制数 N 的尾数，代表了 N 的实际有效值；P 为 N 的阶码，可以决定小数点的具体位置。例如：$N=101.11\text{B}=2^3\times0.10111\text{B}$。

因此，任何一个浮点数 N 都由阶码和尾数两部分组成。阶码部分包括阶符和阶码，尾数部分由数符和尾数组成，其形式为



其中，P_f 为阶符，$P_f=0$ 表示阶码为正，$P_f=1$ 表示阶码为负；S_f 为数符，$S_f=0$ 表示该数为正数，$S_f=1$ 表示该数为负数；小数点的约定位置在尾数之前，实际位置是浮动的，由阶码

决定。浮点法的优点是，数的表示范围大。浮点表示法的缺点是，运算规则复杂，通常要对阶码和尾数分别运算。

1.2.3 二进制数的运算

二进制数的运算分为二进制整数运算和二进制小数运算两种类型，但运算法则完全相同。由于大部分计算机中数的表示方法均采用定点整数表示法，故这里仅介绍二进制整数运算，二进制小数运算与它相同。

二进制数的运算比较简单，包括算术运算和逻辑运算。算术运算包括加、减、乘、除运算；逻辑运算包括逻辑乘、逻辑加、逻辑非和逻辑异或等。

1. 二进制数的算数运算

（1）加法运算

运算规则为

$$0+0=0;\ 0+1=1+0=1;\ 1+1=10\text{（向高位进位）}$$

（2）减法运算

运算规则

$$0-0=0;\ 1-0=1;\ 1-1=0;\ 0-1=1\text{（向高位借 1）}$$

（3）乘法运算

运算规则

$$0\times0=0;\ 0\times1=1\times0=0;\ 1\times1=1$$

（4）除法运算

除法是乘法的逆运算。与十进制类似，二进制除法也是从被除数最高位开始，查找出够减除数的位数，并在其最高位处上商 1 和完成它对除数的减法运算，然后把被除数的下一位移到余数位置上。若余数不够减除数，则上商 0，并把被除数的再下一位移到余数位置上；若余数够减除数，则上商 1 并进行余数减除数。这样反复进行，直到全部被除数的各位都下移到余数位置上为止。

2. 逻辑运算

计算机处理数据时常常要用到逻辑运算。逻辑运算由专门的逻辑电路完成。下面介绍几种常用的逻辑运算。

（1）逻辑乘运算

逻辑乘又称逻辑与，常用∧算符表示。逻辑乘运算法则为

$$0\wedge0=0;\ 1\wedge0=0\wedge1=0;\ 1\wedge1=1$$

（2）逻辑加运算

逻辑加又称逻辑或，常用算符∨表示。逻辑加的运算规则为

$$0\vee0=0;\ 1\vee0=0\vee1=1;\ 1\vee1=1$$

（3）逻辑非运算

逻辑非运算又称逻辑取反，常采用“-”运算符表示。逻辑非的运算规则为

$$\overline{0}=1;\ \overline{1}=0$$

（4）逻辑异或

逻辑异或又称为半加，是不考虑进位的加法，常采用算符表示。逻辑异或的运算规则为

$$0 \oplus 0 = 0;\ 1 \oplus 1 = 0;\ 1 \oplus 0 = 0 \oplus 1 = 1$$

1.3　计算机中数的表示形式

在现代微型机中，其内部运算器通常由一个补码加法器、n 位寄存器/计数器组和移位控制电路等组成，但它恰能进行各种算术运算和逻辑操作。这就是说，补码加法器既能做加法又能将减法运算变为加法来做，从而大大简化了运算器内部的电路设计。这应归功于人们长期以来对计算机中码制的研究。

机器数是指数的符号和值均采用二进制的表示形式。因此，机器数在定点和浮点机中的表示形式各不相同。为了方便起见，这里的机器数均指在定点整数机中的表示形式。即最高位是符号位（0 表示正数，1 表示负数），其余位为数值位，小数点约定在数值位之后。在计算机中，机器数有原码、反码、补码、变形原码、变形反码、变形补码和移码等多种形式。

1.3.1　机器数的原码、反码和补码

原码、反码和补码是机器数的三种基本形式，它和机器数的真值不同。机器数的真值定义为采用 + 和 - 表示的二进制数符号，并非是真正的机器数。例如，+76 的机器数真值为 +1001100B，原码形式为 01001100B（最高位的 0 表示正数）；-76 的真值为 -1001100B，原码为 11001100B（最高位的 1 表示负数）。

1. 原码

原码表示法是机器数的一种简单的表示法。这种表示法数的最高位为符号位（用 0 或 1 来表示），其余位为数值位，符号位的 0 表示该数为正数，符号位为 1 表示它是负数。通常，一个数的原码可以先把该数用方括号括起来，并在方括号右下角加个“原”字来标记。设有一数为 x，则原码表示可记作 $[x]_{原}$。

【例 1-6】 设 X1 = +1100B，X2 = -1100B，请分别写出它们在 8 位微机中的原码。

解：其原码记为

$$[X1]_{原} = [00001100]_{原}$$

$$[X2]_{原} = [10001100]_{原}$$

原码表示数的范围与二进制位数有关。当用 8 位二进制来表示小数原码时，其表示范围为

最大值为 0.1111111，其真值约为（0.99）D

最小值为 1.1111111，其真值约为（-0.99）D

当用 8 位二进制来表示整数原码时，其表示范围为

最大值为 01111111，其真值为（127）D

最小值为 11111111，其真值为（-127）D

在原码表示法中，对 0 有两种表示形式：

$$[+0]_{原} = 00000000$$

$$[-0]_{原} = 10000000$$

2. 反码

在微型计算机中，二进制数的反码可由原码得到，有正数的反码和负数的反码之分：正数的反码和原码相同；负数反码的符号位和负数原码的符号位相同，数值位是它的数值位的按位取反。反码的标记方法和原码类似，只要在被括数方括号的右下角添加一个“反”字即可。设有一数为 x，则反码表示可记作 $[x]_{反}$。

【例 1-7】 设 X1 = +1010110，X2 = -1001010，请分别写出它们在 8 位微机的原码和反码。

解：由于正数的反码就是其原码；负数的反码是符号位不变，数值位是它的数值位的按位取反，所以有

$[X1]_{原} = 01010110$

$[X1]_{反} = [X1]_{原} = 01010110$

$[X2]_{原} = 11001010$

$[X2]_{反} = 10110101$

反码通常作为求补过程的中间形式，即在一个负数的反码的末位上加 1，就得到了该负数的补码。

3. 补码

补码的概念是在计算机中经常会遇到的，二进制数的补码可由反码得到，如果是正数，则该机器数的补码与原码一样；如果是负数，则该机器数的补码是对它的原码（除符号位外）各位取反，并在末位加 1 而得到的。补码的标记方法和原码、反码类似，只要在被括数方括号的右下角添加一个“补”字即可。设有一数 X，则 X 的补码表示记作 $[X]_{补}$。

【例 1-8】 设 X1 = +1010B，X2 = -01010B，试分别写出它们在 8 位微机中的原码、反码和补码形式。

解：由原码、反码和补码的定义得

$[X1]_{原} = 00001010B$　　$[X2]_{原} = 10001010B$

$[X1]_{反} = 00001010B$　　$[X2]_{反} = 11110101B$

$[X1]_{补} = 00001010B$　　$[X2]_{补} = 11110110B$

补码的优点是可以将减法运算转换为加法运算，其符号位可以连同数值位一起运算。这样非常有利于计算机的实现。

【例 1-9】 45H，-55H，用补码运算的方法求两数之和。

解：$[45H]_{补} - [-55H]_{补} = [-10H]_{补}$

1.3.2 计算机中二进制数的单位表示

在计算机中使用的二进制数共有 3 个单位，从小到大依次为位、字节和字。

1. 位（bit）

这里所说的位是指二进制数的位。位是数的最小单位，bit 是位的英文名称，读作“比特”。

在计算机中位仅有 0 和 1 两个数值，表示两种状态。

2. 字节（Byte）

8 位二进制数称为一个字节，其英文名称是 Byte，在使用时常用大写字母 B 表示。字节

是最基本的数据单位，计算机中的数据、代码、指令、地址多以字节为单位。

3. 字（Word）

字是一台计算机上所能并行处理的二进制数，字的位数（或长度）称之为字长。字长必须是字节的整数倍。例如，MCS-51 单片机字长为 8 位，MCS-96 单片机字长为 16 位，还有 32 位、64 位字长的计算机。

1.3.3　计算机使用二进制数的原因

为什么在计算机中要使用二进制数呢？其原因主要有以下几点：

1. 易于实现

在计算机中，数是用不同的物理状态来表示的。因为二进制数只有两个数字 0 和 1，用两种物理状态就可以表示出来。而两种相反的物理状态在技术上极易实现。例如，开关的接通与断开，晶体管的导通与截止，电平的高低，脉冲的有无等。

对于这样两种截然相反的物理状态，不但易于实现，而且状态稳定可靠。而两种以上的物理状态，不但难以实现，而且稳定性也差。

2. 运算简单

因为二进制数只有两个数字，所以对二进制数的运算比人们熟悉的十进制数的运算要简单得多，而运算简单将有利于简化计算机的电路结构。

3. 具有逻辑属性

由于二进制数的 0 和 1 正好与逻辑值的“假（F）”和“真（T）”相对应，因此可以使用二进制数实现逻辑运算，从而使逻辑代数运算成为可能。

4. 可靠性高

由于二进制数用两种截然相反的物理状态表示，十分稳定。因此二进制数的处理、存储和传送都最为可靠。

1.4　计算机中使用的编码

计算机中除了使用数以外，还使用编码。可以把编码分为两类：一类是数的编码，另一类是文字符号的编码。因为是在计算机中使用，所以编码必须是二进制数。

1. 二一十进制编码

在计算机中最常用的是用二进制数给十进制数编码，即通常所说的二-十进制编码。若要给一位十进制数编码，则须用 4 位二进制数。在二-十进制编码中最常用的是 BCD 码。

BCD 码共有 10 个编码，即二进制数 0000～1001，分别对应十进制数 0～9。例如，十进制数 3 的 BCD 码是 0011；9 的 BCD 码是 1001；39 的 BCD 码是把 3 和 9 的 BCD 码连在一起，即 00111001，正好为 1 个字节。BCD 码的特点是，4 位之内为二进制关系，每 4 位之间为十进制关系。

定义 BCD 码是为了便于在计算机中使用人们最熟悉的十进制数，特别是在输入与输出操作中。例如，从键盘输入的十进制数到计算机中就变为 BCD 码形式，当然这需要有相应的转换程序。有了十进制数的输入和输出，在计算机中就会存在十进制数的存储和计算。但十进制数计算存在调整问题，即所谓的十进制调整，以解决 BCD 码运算时因进位和借位产

生的偏差。

2. ASCII 码

文字符号代码用于在计算机中表示西文字符、汉字以及各种符号，最常用的文字符号代码是 ASCII 码和汉字国标码。这里只介绍 ASCII 码。

ASCII 代表的是“美国信息交换标准代码”。它原是美国的字符代码标准，于 1968 年发表，由于使用广泛，早已被国际标准化组织确定为国际标准，成为计算机领域中最重要的代码。

ASCII 码表见表 1-2。

表 1-2 ASCII 码表

$b_6b_5b_4$ / $b_3b_2b_1b_0$	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	、	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	·	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

ASCII 码中字符和功能符号共计 128 个：其中字符 94 个，包括十进制数字 10 个，英文小写字母 26 个，英文大写字母 26 个，标点符号及专用符号 32 个，功能符 34 个（字符区首尾两个符号 SP 和 DEL 一般归入功能符）。由于 $2^7=128$，因此 128 个字符和功能符使用 7 位二进制数就可以进行编码，此编码即为 ASCII 码。

ASCII 码表是一个 16 行 ×8 列的矩阵，其中行为编码中的后 4 位二进制数（$b_3b_2b_1b_0$），列为编码中的前 3 位二进制数（$b_6b_5b_4$），合在一起为 7 位二进制编码。例如，字符 A 的编码为 1000001。

为了方便，常用十进制数或十六进制数来表示 ASCII 码。例如，字符 A 的 ASCII 码用十进制数表示为 65，用十六进制数表示为 41H。

7 位 ASCII 码结构是基本 ASCII 码，由于在计算机中常用字节（8 位）来表示数据。因此，为凑成一个字节，应在 ASCII 码的最高位补 1 个 0。

本 章 小 结

在计算机中常用的数制有十进制、二进制和十六进制。不同数制之间的转换都有一定的规则，如二进制数转换成十六进制数采用“四位合一位”法，十六进制数转换成二进制数采用“一位分四位”法，十进制整数转换成二进制整数采用“除 2 取余”法，十进制整数转换成十六进制整数采用“除 16 取余”法，十进制小数转换成二进制小数采用“乘 2 取整”法，十进制小数转换成十六进制小数采用“乘 16 取整”法。

有符号二进制数有三种表示法，即原码、反码和补码。在计算机中有符号数一般用补码表示，无论是加法还是减法都可采用加法运算，而且是连同符号位一起进行的，运算的结果仍为补码。

BCD 码的加减法运算与十进制运算规则相同，但必须对运算结果进行修正。ASCII 码是国际通用的标准编码，采用 7 位二进制编码，分为图形字符和控制字符两类，共 128 个字符。

习　　题

1. 十进制和二进制数各有什么特点？请举例加以说明。
2. 为什么微机要采用二进制？
3. 把下列十进制数转换为二进制数和十六进制数：
① 135　② 0. 625　③ 47. 6875　④ 0. 94　⑤ 111. 111　⑥ 1995. 12
4. 把下列二进制数转换为十进制数和十六进制数：
① 11010110B　② 1100110111B　③ 0. 1011B
④ 0. 10011001B　⑤ 1011. 1011B　⑥ 111100001111. 11011B
5. 把下列十六进制数转换成十进制数和二进制数：
① AAH　② BBH　③ C. CH　④ DE. FCH　⑤ ABC. DH　⑥ 128. 08H
6. 先把下列十六进制数变成二进制数，然后分别完成逻辑乘、逻辑加和逻辑异或操作，应写出竖式：
① 33H 和 BBH　② ABH 和 7FH　③ CDH 和 80H　④ 78H 和 0FH
7. 先把下列各数变成 8 位二进制数（含符号位），然后按补码运算规则求 \ [X + Y \ ]补及其真值：
① X = +46　② X = +78　③ X = +112　④ X = −51
8. 请写出下列各十进制数在 8 位定点整数机中的原码、反码和补码形式（最高位为符号位）：
① X = +38　② X = +76　③ X = −54　④ X = −115
9. 写出下列各数的 BCD 码：
① 47　② 59　③ 1996　④ 1997. 6
10. 用十六进制形式写出下列字符的 ASCII 码：
① AB8　② STUDENT　③ COMPUTER　④ GOOD

第 2 章　80C51 的结构及原理

【学习目的】

1. 了解 80C51 的内部结构。
2. 掌握 80C51 引脚信号功能定义。
3. 掌握 80C51 的存储器空间分配及各 I/O 口的特点。
4. 掌握 80C51 的复位电路、时钟电路及指令时序。

美国英特尔（intel）公司生产的 MCS-51 系列单片机具有典型的硬件结构和完善的指令系统，为单片机的发展奠定了良好的基础。众多单片机芯片的厂商都以它为基核，开发出具有不同功能和应用场合的单片机，为单片机的使用者提供了更多的选择余地。本书所介绍的 80C51 型单片机是 MCS-51 系列单片机的典型产品。

2.1　单片机的概述

2.1.1　单片机的定义

微型计算机的发展，促进了工业控制领域的进步。目前，已有各种各样的微型计算机控制系统在工业中得到应用。一个微型计算机包括微处理器（CPU）、存放程序指令存储器（ROM）和存放数据的存储器（RAM）、输入/输出端口（I/O 端口）及时钟、计数器和中断等。它们经过地址总线（Address Bus）、数据总线（Data Bus）和控制总线（Control Bus）的连接以及输入/输出端口与外围设备连接，构成微机系统。由于单片微处理器集成了计算机的主要器件，所以把单片微处理器看做一个微型计算机，简称单片机（Single Chip Microcomputer）。

2.1.2　单片机的特点

单片机是微型计算机的一个重要分支，它是工业控制和智能化系统中应用最多的一种微机。它的最大特点是设计者可以根据自己的实际需要开发、设计一个单片机系统，更加方便、灵活，且成本低。

单片机就是计算机，由于计算机的特殊结构，其具有如下特点：

① 体积小、重量轻。

② 电源单一、功耗低（突出特点）。

许多单片机可在 2.2V 的电压下工作，有的能在 1.2V 或 0.9V 电压下工作，功耗降为 μA 级。

③ 功能强、价格低，有优异的性能价格比。

④ 全部集成在芯片上，布线短、合理、集成度高。

⑤ 数据大部分在单片机内传递，运行速度快，抗干扰能力强，可靠性高。

2.1.3　单片机的发展

单片机自问世以来，性能不断提高和完善，不仅能满足很多应用场合的需要，而且具有集成度高、功能强、速度快、体积小、功耗低、使用方便、性能可靠、价格低廉等特点，因此，在工业控制、智能仪器仪表、数据采集和处理、通信系统、网络系统、汽车工业、国防工业、高级计算器具、家用电器等领域的应用日益广泛，并且正在逐步取代现有的多片微机应用系统，单片机的潜力越来越被人们所重视。特别是当前用 CMOS 工艺制成的各种单片机，由于功耗低，使用的温度范围大、抗干扰能力强、能满足一些特殊要求的应用场合，更加扩大了单片机的应用范围，也进一步促进了单片机技术的发展。

单片机的发展主要经历了 3 个阶段（美国 intel 公司）。

第 1 阶段（1971 ~ 1978 年）：初级单片机阶段，以 MCS-48 系列为代表。有 4 位、8 位 CPU，并行 I/O 口，8 位定时器/计数器，无串行口，中断处理比较简单，RAM、ROM 容量较小，寻址范围不超过 4KB。

第 2 阶段（1978 ~ 1983 年）：单片机普及阶段，以 MCS-51 系列为代表。是 8 位 CPU，片内 RAM、ROM 容量加大，片外寻址范围可达 64KB，增加了串行口，多级中断处理系统，16 位定时器/计数器。

第 3 阶段（1983 年以后）：16 位单片机阶段，以 MCS-96 系列为代表。是 16 位 CPU，片内 RAM、ROM 容量进一步增大，增加了 A-D 和 D-A 转换器，8 级中断处理功能，实时处理能力更强。它允许用户采用面向工业控制的专用语言，如 C 语言等。

总之，单片机发展可归结为以下几个方面：

① 增加字长，提高数据精度和处理的速度。

② 改进制作工艺，提高单片机的整体性能。

③ 由复杂指令集 CISC 转向简单指令集 RISC 技术。

④ 多功能模块集成技术，使一块“嵌入式”芯片具有多种功能。

⑤ 微处理器与 DSP 技术相结合。

⑥ 融入高级语言的编译程序。

⑥ 低电压、宽电压、低功耗。

目前，国际市场上 8 位、16 位单片机系列已有很多，32 位的单片机也已经进入了实用阶段。随着单片机技术的不断发展，新型单片机还将不断涌现，单片机技术正以惊人的速度向前发展着。

2.1.4　单片机的应用

单片机是在芯片上集成了微型计算机所需的 CPU、存储器、输入/输出部件和时钟电路等。因此它具有体积小、使用灵活、成本低、易于产品化、抗干扰能力强、可在各种恶劣环境下可靠地工作等特点。特别是它应用面广，控制能力强，使它在工业控制、智能仪表、外设控制、家用电器、机器人、军事装置等方面得到了广泛的应用。单片机主要可用于以下几方面：

1. 家用电器

已广泛应用于家用电器的自动控制中，如洗衣机、空调机、电冰箱、彩色电视机、录像机、VCD、音响设备和手机等。单片机的使用提高了家用电器的性能和质量，降低了家用电器的生产成本和销售价格。

2. 智能卡

尽管目前使用的各种卡主要是磁卡和 IC 卡，但是，带有 CPU 和存储器的智能卡，已经并将日益广泛用于金融、通信、信息、医疗保健、社会保险、教育、旅游、娱乐和交通等各个领域。

3. 智能仪器仪表

单片机体积小、耗电少，被广泛用于各类仪器仪表，如智能电表、智能流量计、气体分析仪、智能电压电流测试仪和智能医疗仪器等。单片机使仪器仪表走向了智能化和微型化，使仪器仪表的功能和可靠性大大提高。

4. 网络与通信

许多型号的单片机都有通信接口可方便地进行机间通信，也可方便地组成网络系统，如单片机控制的无线遥控系统、列车无线通信系统和串行自动呼叫应答系统等。

5. 工业控制

单片机可以构成各种工业测控系统、数据采集系统，如数控机床、汽车安全技术检测系统、报警系统和生产过程自动控制等。

2.2 80C51 单片机的逻辑结构

2.2.1 80C51 单片机的组成

单片机芯片上集成了微型计算机的 5 个组成部分，单片机的片内结构框图如图 2-1 所示。它主要由以下几部分组成：8 位的微处理器 CPU，内含布尔处理器，一个片内振荡器和时钟电路；数据存储器 RAM，程序存储器 ROM；定时器/计数器；串行通信接口；64KB 总线扩展控制器和 4 个并行 I/O 接口等。MCS-51 单片机片内总体结构框图如图 2-1 所示。

2.2.2 中央处理单元

中央处理单元（CPU）是 MCS-51 单片机核心部分，主要由运算器和控制器两部分构成。它的工作过程是，取指令→分析指令→根据指令的功能控制单片机的各功能部件执行指定的运算或操作，如此循环往复。

1. 运算器

运算器由算术/逻辑运算单元 ALU、累加器 ACC、寄存器 B、暂存寄存器、程序状态字寄存器 PSW 组成。它所完成的任务是实现算术与逻辑运算、位变量处理和数据传送等操作。

1）算术/逻辑运算单元 ALU 可以实现 8 位数据的加、减、乘、除算术运算和与、或、异或、循环、取反等逻辑运算，同时还具有一般微处理器所不具备的位处理功能。

2）累加器 ACC 的作用是向 ALU 提供操作数和存放运算的结果。同一般微机一样，MCS-51 单片机在结构上也是以累加器 ACC 为中心，大部分指令的执行都要通过累加器

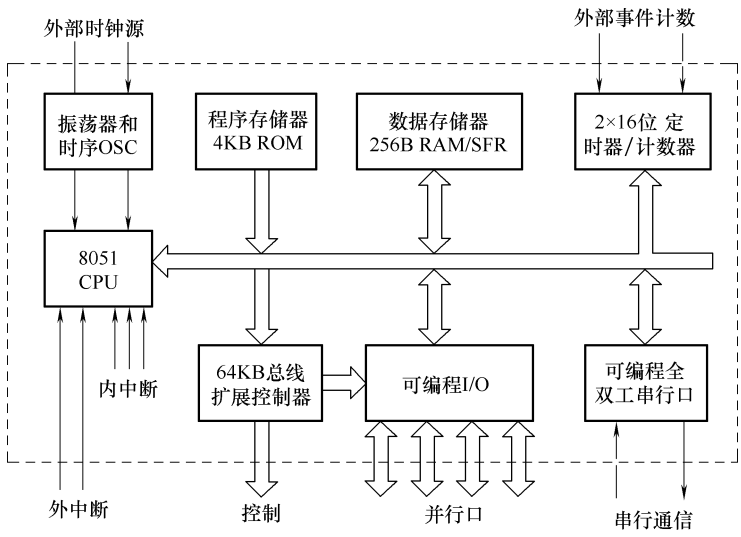


图 2-1　MCS-51 单片机片内总体结构框图

ACC 进行。在运算时将累加器 ACC 中的一个操作数经暂存器 2 送至 ALU，与另一个来自暂存器 1 的操作数在 ALU 中进行运算，运算后的结果又送回累加器 ACC。

3）寄存器 B 作为通用的寄存器使用。另外在乘、除运算时用来存放一个操作数，也用来存放运算后的一部分结果。

4）暂存寄存器用来暂时存放数据总线或其他寄存器送来的操作数。它作为 ALU 的数据输入源，向 ALU 提供操作数。

5）程序状态字寄存器 PSW 用来保存 ALU 运算结果的特征位，如结果是否有溢出、是否有进位/借位等。这些特征位可以作为控制程序转移的条件，供程序判别和查询。

2. 控制器

控制器是由指令寄存器 IR、指令译码器 ID、定时及控制逻辑电路和程序计数器 PC 等部件组成。

1）程序计数器 PC 是一个 16 位的计数器。它总是存放着下一个要取指令的 16 位存储单元地址。即 CPU 将 PC 的内容作为地址，从内存中取出指令码或含在指令中的操作数。每取完一个字节后，PC 的内容自动加 1，为取下一个字节做好准备。在执行转移、子程序调用指令和中断响应时例外，PC 的内容不再加 1，而是由指令或中断响应过程自动给 PC 置入新的地址。

2）指令寄存器（IR）保存当前正在执行的一条指令。执行一条指令，先要把它从程序存储器取到指令寄存器中，然后进行译码，并形成相应指令的微操作信号。

3）定时与控制是微处理器的核心部件，它的任务是控制取指令、执行指令、存取操作数或运算结果等操作，以及向其他部件发出各种微操作控制信号、协调各部件的工作。

2.2.3　存储器

MCS-51 系列单片机片内有程序存储器 ROM 和数据存储器 RAM，不同型号的单片机片内存储器的容量稍有不同，内部结构框图如图 2-2 所示。

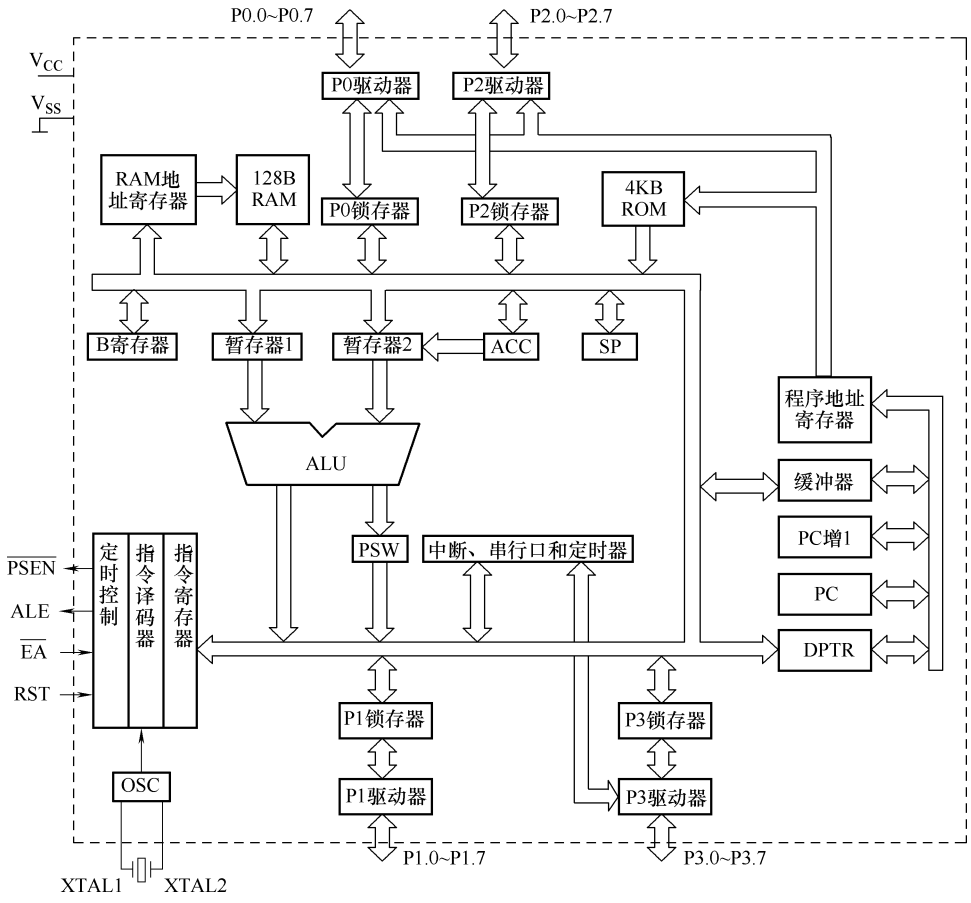


图 2-2 MCS-51 单片机内部结构框图

8051 及 8751 的片内程序存储器容量为 4KB，用于存放程序和表格常数；8051/8751/8031 片内数据存储器均为 128B，用于存放运算的中间结果、数据暂存以及数据缓冲等。在这 128B 的 RAM 中，有 32B 可指定为工作寄存器，和片内 RAM 排在一个队列里统一编址。

2.2.4 输入/输出接口

MCS-51 系列单片机有四个 8 位并行接口（P0～P3），它们都是双向的，每个接口的 8 条 I/O 线均可以进行数据的输入/输出。端口的详细介绍请看本章第 5 节。

2.3 80C51 单片机的引脚及功能

2.3.1 80C51 单片机的引脚封装

MCS-51 系列单片机通常有两种封装：一种是双列直插式封装，另一种是方形封装。图 2-3 所示为双列直插式封装（DIP）形式。

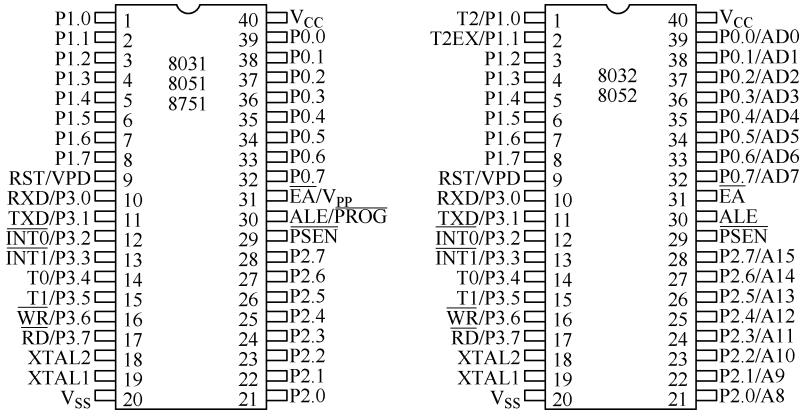


图 2-3　双列直插式封装形式

2.3.2　80C51 单片机引脚及功能

这种封装形式的 80C51 单片机共有 40 根引脚，分为电源线、端口线和控制线三类。

1. 电源线

V_{CC}、V_{SS}（㊵、⑳脚）：电源接入引脚，MCS-51 系列单片机采用 +5V 的电源电压。使用时 V_{CC}接电源的正极（+5V），V_{SS}接电源负极（0V）。

2. 输入/输出端口引脚

P0，P1，P2 和 P3。

四个并行端口都可以作为普通的 I/O 端口使用，除 P1 口外其他三个端口都具有第二功能。

（1）P0 口（P0.7 ~ P0.1，㊴ ~ ㉜脚）

P0 口具有两种功能，分别适用于不同的情况。第一种情况是 P0 口作为通用 I/O 端口使用，P0.0 ~ P0.7 用于传送 CPU 输入/输出数据。这时，输出数据可以得到所存，不需要外接专用锁存器，输入数据可以得到缓冲，增加了数据输入的可靠性。第二种情况是单片机使用了外部存储器，P0.0 ~ P0.7 在 CPU 访问外部存储器时，P0 口作为低 8 位地址/数据的复用总线使用。在此期间，P0 口内部上拉电阻有效。P0 口是一个漏极开路的 8 位准双向 I/O 端口。每位能驱动 8 个 LS 型 TTL 负载。

（2）P1 口（P1.7 ~ P1.0，① ~ ⑧脚）

P1 口作为通用 I/O 端口使用，与 P0 口的第一功能类似，用于传送 CPU 的输入/输出数据。在 P1 口作为输入口使用时，应先向 P1 口锁存器（地址 90H）写入全 1，此时 P1 口引脚由内部上拉电阻拉成高电平，然后再读入 P1 口的数据。P1 口的每一位能驱动（灌入或输出电流）4 个 LS 型 TTL 负载。

（3）P2 口（P2.0 ~ P2.7，㉑ ~ ㉘脚）

P2 和 P0 类似，也有两种功能，分别适用于不同的情况。第一种情况和以上两组引脚的第一功能相同，作为通用 I/O 口使用，用于传送 CPU 输入/输出数据。第二种情况与 P0 口的第二功能配合使用，用于输出片外存储器的高 8 位地址，以构成片外存储器的 16 位地址。但不能像 P0 口那样还可以用来传送存储器的读写数据。P2 口是一个带内部上拉电阻的 8 位

准双向 I/O 端口，每一位能驱动 4 个 LS 型 TTL 负载。

（4）P3 口（P3.0 ~ P3.7，⑩ ~ ⑰脚）

P3 口是一个带内部上拉电阻的 8 位准双向 I/O 端口，也具有两种功能，第一功能与其余三个端口一样，可以作为通用 I/O 来用。第二功能作为控制来用，功能如下：

P3.0——串行数据输入引脚 RXD。

P3.1——串行数据输出引脚 TXD。

P3.2——外部中断 0（$\overline{\text{INT0}}$）的中断请求信号输入引脚。

P3.3——外部中断 1（$\overline{\text{INT1}}$）的中断请求信号输入引脚。

P3.4——定时器 0（T0）外部计数脉冲输入引脚。

P3.5——定时器 1（T1）外部计数脉冲输入引脚。

P3.6——片外数据存储器写选通控制信号输出引脚$\overline{\text{WR}}$。

P3.7——片外数据存储器读选通控制信号输出引脚$\overline{\text{RD}}$。

3. 控制线

（1）XTAL1、XTAL2（⑱、⑲脚）

外部晶体振荡器的接入引脚。

（2）RST/VPD（⑨脚）

此引脚有两个功能：一是作为复位信号输入端 RST 使用，高电平有效，正常工作时，当此输入端保持两个机器周期（24 个振荡周期）的高电平时，就可以完成复位操作；此引脚的第二功能是作为备用电源的输入端 V_{PD}，当主电源 V_{CC} 发生故障，降低到低电平规定值时，将备用电源自动接入，为 RAM 提供备用电源，以保证存储在 RAM 中的信息不丢失，从而使复位后能继续正常运行。

（3）ALE/$\overline{\text{PROG}}$（㉚脚）

此引脚有两个功能：一是输出地址锁存控制信号 ALE，高电平有效；二是作为片内 EPROM 编程写入脉冲输入端。

当单片机上电正常工作后，ALE 引脚不断向外输出地址锁存信号，其频率为振荡器频率 f_{OSC} 的 1/6。CPU 访问片外存储器时，P0 端口输出片外存储器的低 8 位地址的同时在 ALE/$\overline{\text{PROG}}$输出一个高电平脉冲，其下降沿用于这个 8 位地址所存到外部专用地址锁存器，此时 P0 口即可以传送片外存储器的读写数据。在对片内带有 4KB EPROM 的单片机进行固化程序时，作为编程脉冲输入端$\overline{\text{PROG}}$使用。

（4）$\overline{\text{PSEN}}$（㉙脚）

程序存储器输出允许信号，低电平有效。在访问片外扩展的程序存储器时，此端定时输出负脉冲，作为片外存储器的读选通信号。$\overline{\text{PSEN}}$同样可驱动 8 个 LS 型 TTL 负载。

（5）$\overline{\text{EA}}/V_{PP}$（㉛脚）

此引脚有两个功能：一是作为内/外部程序存储器设置信号输入端；二是作为片内 EPROM 编程写入电压输入端使用。

当$\overline{\text{EA}}$引脚接高电平时，CPU 访问片内程序存储器的顺序是由内到外，即 CPU 首先从片内程序存储器开始执行程序，当 PC 的值超过 0FFFH 时，将自动转去执行片外程序存储器内的程序。

当输入信号$\overline{\text{EA}}$引脚接低电平（接地）时，CPU 只访问外部程序存储器，执行外部程序

存储器中的程序，而不管是否有片内程序存储器。对于无片内 ROM 的 8031 或 8032 型号的单片机，在使用时需要通过外部电路扩展程序存储器，此时必须将$\overline{EA}$引脚接地。

此引脚的第二功能 V_{PP}是对 8751 片内 EPROM 固化编程时，作为施加较高编程电压（一般 12～21V）的输入端。

2.4　80C51 单片机的存储器组织

80C51 单片机的存储器在物理结构上分为程序存储器和数据存储器，共有四个存储空间：分别是片内程序存储器、片内数据存储器、片外程序存储器和片外数据存储器。存储空间分布如图 2-4 所示，从使用者的角度看，80C51 单片机存储器的地址空间为三类：程序存储器、数据存储器、特殊功能寄存器。

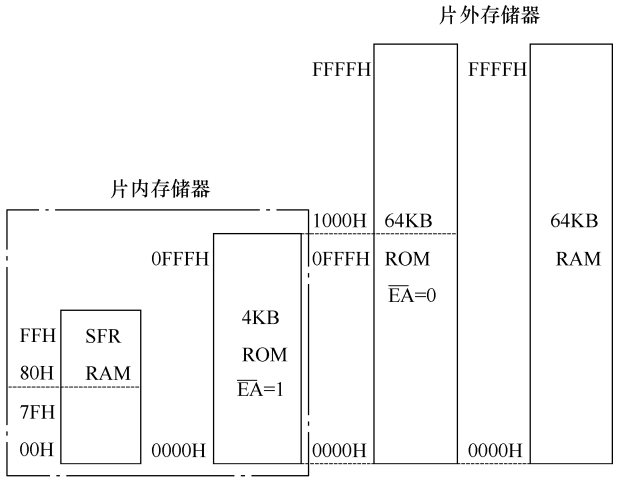


图 2-4　80C51 存储器物理结构

2.4.1　80C51 单片机的程序存储器

80C51 单片机的程序存储器最大配置为 64KB，用于存放编好的程序和表格常数。由图 2-4 所示可知，程序存储器由两个部分组成：片内程序存储器 ROM，8051/8751 的容量为 4KB，地址为 0000H～0FFFH；片外程序存储器最多可扩至 64KB，地址为 1000H～FFFFH，片内外统一编址。并且通过$\overline{EA}$引脚进行设置。

1. $\overline{EA}$引脚的作用

当引脚$\overline{EA}$接高电平（$\overline{EA}$ = 1）时，80C51 单片机程序计数器在 0000H～0FFFH 范围内（即前 4KB 地址）执行片内 ROM 中的程序；当指令地址超过 0FFFH 后，就自动转向执行片外 ROM 中的程序。80C51 单片机从片内程序存储器和片外程序存储器取值时执行速度相同。

当引脚$\overline{EA}$接低电平（$\overline{EA}$ = 0）时，80C51 单片机片内 ROM 不起作用，CPU 只能从片外扩展的程序存储器 ROM 中取指令执行，片外扩展的程序存储器 ROM 的地址从 0000H 开始编址。这种接法特别适用于无片内程序存储器的 8031 型号的单片机。

CPU 访问片内、片外 ROM 时，使用 MOVC 指令。

2. 程序存储器中的保留单元

程序存储器的某些单元是留给系统使用的，见表 2-1。由于 80C51 单片机上电复位后，程序计数器的内容为 0000H，所以 CPU 总是从 0000H 开始执行程序。存储单元 0000H ~ 0002H 用做 80C51 单片机上电复位后引导程序的存放单元。使用时通常在这三个单元中存放一条转移指令（LJMP　MAIN），那么程序就被引导到转移指令指定的程序段去执行。

表 2-1　保留的存储单元

存 储 单元	功　　能
0000H	上电或复位后，引导程序入口地址
0003H	外部中断 0（$\overline{\text{INT0}}$）中断服务程序的入口地址
000BH	定时器 0（T0）中断服务程序的入口地址
0013H	外部中断 1（$\overline{\text{INT1}}$）中断服务程序的入口地址
001BH	定时器 1（T1）中断服务程序的入口地址
0023H	串行口中断服务程序的入口地址
002BH	增强型单片机定时器/计数器 2 溢出或 T2EX 负跳变中断服务程序入口地址

80C51 单片机有 5 个中断源，在程序存储器中规定了 5 个中断服务程序的入口，并且在每个中断向量之间有 8 个单元提供用户使用。例如，外部中断 0 引脚$\overline{\text{INT0}}$（P3.2）有效时，向 CPU 发出中断申请，CPU 响应$\overline{\text{INT0}}$的中断请求后自动将$\overline{\text{INT0}}$中断服务程序的入口地址 0003H 装入 PC，程序就自动转向 0003H 单元开始执行。如果事先在 0003H ~ 000AH 存入转移到$\overline{\text{INT0}}$中断服务程序的转移指令，则程序就被自动转移到指定的中断服务程序空间去执行。

2.4.2　80C51 单片机的数据存储器

数据存储器 RAM 用于存放运算的中间结果、数据暂存和缓冲、标志位等。数据存储器空间也分成片内和片外两大部分，片内存储器空间为 256B，地址范围为 00H ~ 0FFH；片外数据存储器空间可扩展为 64KB，地址范围为 0000H ~ 0FFFFH。下面分别进行介绍。

1. 片外 RAM

片外数据存储器通过硬件电路可以扩展为 64KB，地址范围为 0000H ~ 0FFFFH。使用时通过“MOVX”指令进行数据存取。

2. 片内 RAM

片内数据存储器共有 128B，地址范围为 00H ~ 7FH。它们又分为三个部分：工作寄存器区、位寻址区和用户区。

1）地址为 00H ~ 1FH 的 32B 定义为工作寄存器区，并且分成四组来使用，称为 0 组、1 组、2 组和 3 组。每组有 8 个工作寄存器，每个字节定义为一个工作寄存器，分别用 R0 ~ R7 表示。每组寄存器均可选作 CPU 的当前工作寄存器组。若程序中并不需要四组寄存器，其余可以作一般 RAM 单元使用。CPU 复位后，选中第 0 组寄存器为当前的工作寄存器。通过对程序状态字 PSW 中 RS1、RS0 的设置，可以选择其他组为当前工作寄存器，见表 2-2。

2）地址为 20H ~ 2FH 的单元为“位寻址区”，这 16B 既可以按字节寻址，也可以按位寻址。这 16B 总共 128bit，每 1bit 都有唯一的位地址，可通过位寻址方式访问其各位，其位地址分布见表 2-3。

表 2-2　工作寄存器地址表

组		R7	R6	R5	R4	R3	R2	R1	R0	RS1 RS0
0 组	工作寄存器的地址	07H	06H	05H	04H	03H	02H	01H	00H	0 0
1 组		0FH	0EH	0DH	0CH	0BH	0AH	09H	08H	0 1
2 组		17H	16H	15H	14H	13H	12H	11H	10H	1 0
3 组		1FH	1EH	1DH	1CH	1BH	1AH	19H	18H	1 1

表 2-3　RAM 位寻址区地址表

字节地址	MSB	位　地　址						LSB
2FH	7FH	7EH	7DH	7CH	7BH	7AH	79H	78H
2EH	77H	76H	75H	74H	73H	72H	71H	70H
2DH	6FH	6EH	6DH	6CH	6BH	6AH	69H	68H
2CH	67H	66H	65H	64H	63H	62H	61H	60H
2BH	5FH	5EH	5DH	5CH	5BH	5AH	59H	58H
2AH	57H	56H	55H	54H	53H	52H	51H	50H
29H	4FH	4EH	4DH	4CH	4BH	4AH	49H	48H
28H	47H	46H	45H	44H	43H	42H	41H	40H
27H	3FH	3EH	3DH	3CH	3BH	3AH	39H	38H
26H	37H	36H	35H	34H	33H	32H	31H	30H
25H	2FH	2EH	2DH	2CH	2BH	2AH	29H	28H
24H	27H	6H	25H	24H	23H	22H	21H	20H
23H	1FH	1EH	1DH	1CH	1BH	1AH	19H	18H
22H	17H	16H	15H	14H	13H	12H	11H	10H
21H	0FH	0EH	0DH	0CH	0BH	0AH	09H	08H
20H	07H	06H	05H	04H	03H	02H	01H	00H

3）地址为 30H ~ 7FH 的单元为用户区，用户可以在这个区域存取数据，也可以定义为堆栈区。

对于 128B 的 RAM 区，CPU 访问时可以采用字节地址或位地址方式。如果访问位地址区，就用位寻址方式，否则可以采用直接寻址或间接寻址方式。

位寻址能力是 80C51 单片机的一个重要特点。这些可寻址位，通过执行指令可直接对某一位操作，如置 1、清 0 等，可用作软件标志位或用于位（布尔）处理。

2. 4. 3　80C51 单片机的特殊功能寄存器

80C51 单片机片内 RAM 的地址空间 80H ~ FFH 为特殊功能寄存器（SFR）区，有 21 个特殊功能寄存器，它们离散地分布在 80H ~ FFH 的 RAM 空间中。访问特殊功能寄存器只允许使用直接寻址方式，这些特殊功能寄存器见表 2-4 所示。

在表 2-4 中，有 21 个特殊功能寄存器，有些特殊功能寄存器的符号地址上标有“※”号，它表示该特殊功能寄存器既可以位寻址也可以字节寻址。表中有 10 个寄存器是既可以

表 2-4 80C51 单片机特殊功能寄存器表

符号地址	名　　称	字节地址
B	B 寄存器	F0
* ACC	累加器	E0H
* PSW	程序状态字寄存器	D0H
* IP	中断优先级控制寄存器	B8H
* P3	P3 口锁存寄存器	B0H
* IE	中断允许控制寄存器	A8H
* P2	P2 口锁存寄存器	A0H
SBUF	串行口数据缓冲器	99H
* SCON	串行口控制寄存器	98H
* P1	P1 口锁存寄存器	90H
TH1	定时器/计数器 1 的高字节	8DH
TH0	定时器/计数器 1 的低字节	8CH
TL1	定时器/计数器 0 的高字节	8BH
TL0	定时器/计数器 0 的低字节	8AH
TMOD	定时器/计数器工作方式寄存器	89H
* TCON	定时器/计数器控制寄存器	88H
PCON	电源控制寄存器	87H
DPH	数据指针寄存器高字节	83H
DPL	数据指针寄存器低字节	82H
SP	堆栈指针寄存器	81H
* P0	P0 口锁存寄存器	80H

位寻址也可以字节寻址，特征是它们的字节地址正好能被 8 整除，其地址分布见表 2-5。下面介绍部分特殊功能寄存器，其余将在后续章节中讲述。

表 2-5 特殊功能寄存器地址表

SFR	MSB	位地址/定义						LSB	字节地址
B									F0
ACC	E7H	E6H	E5H	E4H	E3H	E2H	E1H	E0H	E0H
	ACC. 7	ACC. 6	ACC. 5	ACC. 4	ACC. 3	ACC. 2	ACC. 1	ACC. 0	
PSW	D7H	D6H	D5H	D4H	D3H	D2H	D1H	D0H	D0H
	CY	AC	F0	RS1	RS0	OV	F1	P	
IP	BFH	BEH	BDH	BCH	BBH	BAH	B9H	B8H	B8H
	/	/	/	PS	PT1	PX1	PT0	PX0	
P3	B7H	B6H	B5H	B4H	B3H	B2H	B1H	B0H	B0H
	P3. 7	P3. 6	P3. 5	P3. 4	P3. 3	P3. 2	P3. 1	P3. 0	

（续）

SFR	MSB	位地址/定义						LSB	字节地址
IE	AFH	AEH	ADH	ACH	ABH	AAH	A9H	A8H	A8H
	EA	/	/	ES	ET1	EX1	ET0	EX0	
P2	A7H	A96H	A95H	A4H	A3H	A2H	A1H	A0H	A0H
	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0	
SBUF									99H
SCON	9FH	9EH	9DH	9CH	9BH	9AH	99H	98H	98H
	SM0	SM1	SM2	REN	TB8	RB8	TI	RI	
P1	97H	96H	95H	94H	93H	92H	91H	90H	90H
	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0	
TH1									8DH
TH0									8CH
TL1									8BH
TL0									8AH
TMOD	GATE	C/T	M1	M0	GATE	C/T	M1	M0	89H
TCON	8FH	8EH	8DH	8CH	8BH	8AH	89H	88H	88H
	TF1	TR1	TF0	TR0	IT1	IE1	IT0	IE0	
PCON	SMOD	/	/	/	GF1	GF0	PD	IDL	87H
DPH									83H
DPL									82H
SP									81H
P0	87H	86H	85H	84H	83H	82H	81H	80H	80H
	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0	

1）累加器 ACC 是 80C51 单片机最常用的 8 位特殊功能寄存器；在执行运算指令时，许多指令的操作数取自于 ACC，许多运算中间结果也存放于 ACC 中。在指令系统中用 A 作为累加器 ACC 的助记符。

2）寄存器 B 在乘、除指令中使用最多。乘法指令的两个操作数分别取自 A 和 B，乘积存于 B 和 A 两个 8 位寄存器中。详见乘除法指令，在其他指令中，B 可作为一般通用寄存器或一个 RAM 单元使用。

3）程序状态字寄存器 PSW，它的 8 位包含了程序执行后的状态信息，供程序查询和判断之用。其各位含义如下：

CY：进位、借位标志位，有进位、借位时 CY = 1，否则 CY = 0。

AC：辅助进位、借位标志位，当高半字节与低半字节间有进位或借位时 AC = 1，否则 AC = 0。

F0：用户标志位，由用户自己定义。

RS1、RS0：当前工作寄存器组选择位，工作寄存器地址表见表 2-2。

OV：溢出标志位，有溢出时 OV = 1，否则 OV = 0。

P：奇偶标志位，存于 ACC 中的运算结果有奇数个 1 时 P = 1，否则 P = 0。

4）8 位堆栈指针寄存器 SP，它总是指向栈顶。80C51 单片机的堆栈通常设在 30H ~ 7FH 这一段 RAM 中。在系统上电或复位时，SP 的初始值为 07H，可以在初始化程序中重新设置。

80C51 单片机堆栈操作遵循“先进后出，后进先出”的原则：入栈操作时，首先执行 SP 加 1 操作，然后数据入栈存入 SP 指向的单元；出栈操作时，先将 SP 指向单元的数据弹出，然后执行 SP 减 1 操作，这时 SP 指向的单元是新的栈顶。由此可见，80C51 单片机的堆栈区是向地址增大的方向生成的。

5）16 位数据指针寄存器 DPTR，通常作为地址寄存器使用，用于存放 16 位地址，也可以分成两个 8 位寄存器 DPH 和 DPL 使用。可以对片外 64KB 范围的 RAM/ROM 数据进行间接寻址或变址寻址操作。

6）并行 I/O 端口锁存器 P0 ~ P3，它们都可以进行位寻址和字节寻址。每一条 I/O 线均可以作为输入和输出使用，输出时可以锁存数据，输入时可以进行数据缓冲。

2.5 80C51 单片机的并行 I/O 口

80C51 单片机有 4 个 8 位的并行 I/O 口 P0、P1、P2 和 P3。各口均由口锁存器、输出驱动器和输入缓冲器组成。4 个并行口既有字节地址又有位地址。对各个并行口锁存器的读写，就可以实现口的输入/输出操作。4 个并行口的功能有所不同，结构也存在一些差异，但每个口的位结构是相同的。所以，口结构的介绍均以其位结构进行说明。

2.5.1 P0、P2 口的结构

80C51 单片机中，使用内部含有程序存储器的单片机，基本上不需要外部扩展程序存储器和数据存储器，这时 P0、P2 口可用作通用的输入/输出口。若使用内部无程序存储器型号的单片机，就需要通过外部电路扩展程序存储器和数据存储器，此时，P0、P2 口作为总线接口使用，即 P0 口为分时复用的低 8 位地址/数据总线，P2 口作为高 8 位地址总线。

1. P0 口的结构

P0 口的位结构如图 2-5 所示。P0. X 位由一个输出锁存器、一个转换开关 MUX、两个三态输入缓冲器 1、2、输出驱动电路（T_1、T_2）和一个与门 4 及一个反相器 3 组成，图中控制信号 C 的状态决定转换开关的位置。当 C = 0 时，开关处于图中所示位置；当 C = 1 时，开关拨向反相器输出端位置。

（1）P0 作为通用 I/O 口使用

当单片机系统没有外部扩展 ROM/RAM 时，P0 用作通用 I/O 口使用。在这种情况下，单片机硬件逻辑自动控制 C = 0，MUX 开关处于当前位置。另外，C = 0，与门 4 的输出为“0”，使输出驱动器的上拉场效应晶体管 T_1 处于截止状态。因此，输出驱动级工作在需要外接上拉电阻的漏极开路方式。

作输出口时，CPU 执行口的输出指令，内部数据总线上的数据在“写锁存器”信号的作用下由 D 端进入锁存器，经锁存器的 $\overline{Q}$ 端送至场效应晶体管 T_2，再经 T_2 反相，在 P0. X

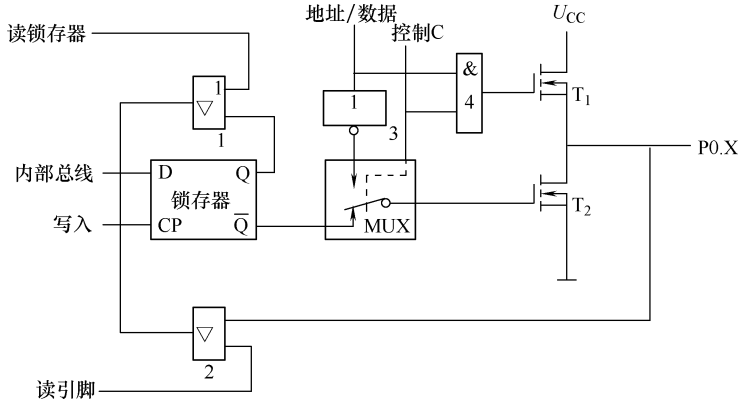


图 2-5　P0 口的位结构

引脚出现的数据正好是内部总线的数据。

作输入口时，数据可以读自口的锁存器，也可以读自口的引脚。这要根据输入操作采用的是“读锁存器”指令，还是“读引脚”指令来决定。

CPU 在执行“读—修改—写”类输入指令时（如 ANL　P0，A），内部产生的“读锁存器”操作信号，使锁存器 Q 端数据进入内部数据总线，在与累加器 A 进行逻辑运算之后，结果又送回 P0 口锁存器并出现在引脚上。读端口锁存器可以避免因外部电路原因造成的误读。

CPU 在执行“MOV”类输入指令时（如 MOV　A，P0），内部产生的操作信号是“读引脚”。注意，在执行该类输入指令前要先给锁存器写入“1”，目的是使场效应晶体管 T_2 截止，从而使引脚处于悬浮状态，可以作为高阻抗输入。否则，在作为输入方式之前曾向锁存器输出过“0”，T_2 导通会使引脚钳位在“0”电平，使输入高电平“1”无法读入。所以，P0 口在作为通用 I/O 口时，属于准双向口。

（2）P0 作为地址/数据总线口使用

当系统扩展片外 ROM/RAM 时，P0 作为地址/数据总线使用。在这种情况下，单片机内硬件逻辑自动使 MUX 开关控制端 C = 1，则 MUX 开关接到反相器 3 的输出端，这时与门 4 的输出由地址/数据线的状态决定。

CPU 在执行输出指令时，低 8 位地址和数据信息分时地出现在地址/数据总线上。若地址/数据总线的状态为“1”，则场效应晶体管 T_1 导通、T_2 截止，引脚状态为“1”；若地址/数据总线的状态为“0”，则场效应晶体管 T_1 截止、T_2 导通，引脚状态为“0”。可见 P0. X 引脚的状态正好与地址/数据线的信息相同。

CPU 在执行输入指令时，首先低 8 位地址信息出现在地址/数据总线上，P0. X 引脚的状态与地址/数据总线的地址信息相同。然后，CPU 自动地使转换开关 MUX 拨向锁存器，并向 P0 口写入 FFH，同时“读引脚”信号有效，数据经缓冲器进入内部数据总线。

由此可见，P0 口作为地址/数据总线使用时是一个真正的双向口。

2. P2 口的结构

P2 口的位结构如图 2-6 所示，P2. X 位由一个输出锁存器、一个转换开关 MUX、两个三态输入缓冲器、输出驱动电路和一个反相器组成。图中控制信号的状态决定转换开关的位

置。当控制信号为 0 时，开关处于图中所示位置；当控制信号为 1 时，开关拨向地址线位置。

由图可见，P2 口的输出驱动电路与 P0 口不同，只使用一个场效应晶体管，其内部设有上拉电阻。

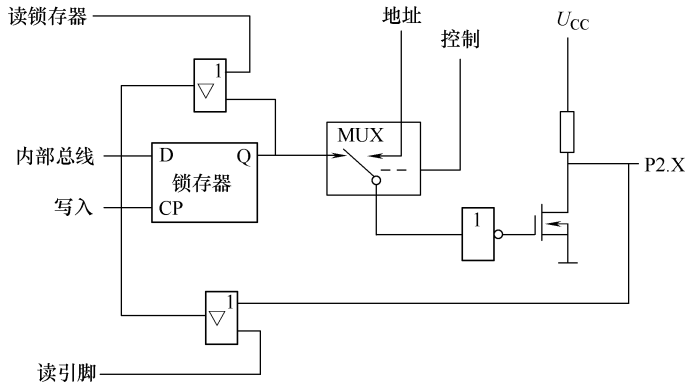


图 2-6　P2 口的位结构

（1）P2 作为通用 I/O 口使用

当单片机系统没有外部扩展 ROM/RAM 时，P2 可以作为通用 I/O 口使用。

CPU 在执行输出指令时，内部数据总线的数据在“写锁存器”信号的作用下由 D 端进入锁存器，经反相器反相后送至场效应晶体管 T_2，再经 T_2 反相，在 P2. X 引脚出现的数据正好是内部总线的数据。

P2 口用作输入时，数据可以读自口的锁存器，也可以读自口的引脚。这要根据输入操作采用的是“读锁存器”指令还是“读引脚”指令来决定。

CPU 在执行“读一修改一写”类输入指令时（如 ANL　P2，A），内部产生的“读锁存器”操作信号，使锁存器 Q 端数据进入内部数据总线，在与累加器 A 进行逻辑运算之后，结果又送回 P2 口的锁存器并出现在引脚。

CPU 在执行“MOV”类输入指令时（如 MOV　A，P2），内部产生的操作信号是“读引脚”。应在执行输入指令前对锁存器写入“1”，目的是使场效应晶体管 T_2 截止，从而使引脚处于高阻抗输入状态。所以，P2 口在作为通用 I/O 口时，属于准双向口。

（2）P2 口作为地址总线接口使用

当需要在单片机外部扩展程序存储器或数据存储器时，单片机内部硬件逻辑自动控制 MUX 开关接向地址线，这时 P2. X 引脚的状态正好与地址线输出的信息相同。

2. 5. 2　P1、P3 口的结构

P1 口是 80C51 单片机唯一的单功能口，仅能用作通用的数据输入/输出口。P3 口是双功能口，除具有数据输入/输出功能外，还具有特殊的第二功能。

1. P1 口的结构

P1 口的位结构如图 2-7 所示。由图可见，P1 口由一个输出锁存器、两个三态输入缓冲器和输出驱动电路组成。其输出驱动电路与 P2 口相同，内部设有上拉电阻。P1 口是通用的

准双向 I/O 口。输出高电平时，能向外提供拉电流负载。用作输入口使用时，在读入数据前必须向口锁存器写入“1”。

2. P3 口的结构

P3 口的位结构如图 2-8 所示。P3 口由一个输出锁存器、三个输入缓冲器（1、2、4），其中缓冲器 1、2 为三态门，场效应晶体管输出驱动电路和一个与非门（3）组成。输出驱动电路内部设有上拉电阻。

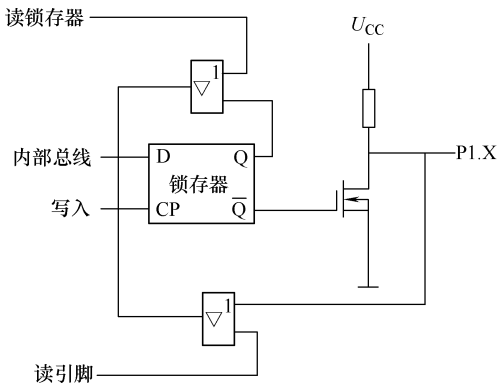


图 2-7　P1 口的位结构

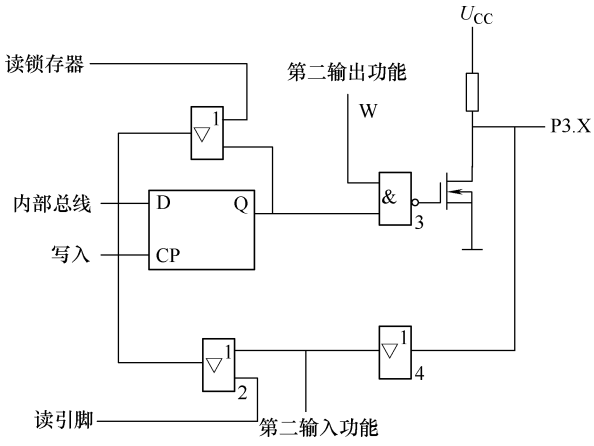


图 2-8　P3 口的位结构

（1）P3 作为通用 I/O 口使用（第一功能）

单片机内部的硬件自动将第二功能输出线的 W 置 1。这时，对应的口线为通用 I/O 口方式。作为输出时，锁存器的状态（Q 端）与输出引脚的状态相同；作为输入时，也要先向口锁存器写入 1，使引脚处于高阻输入状态。输入的数据在“读引脚”信号的作用下，进入内部数据总线。所以，P3 口在作为通用 I/O 口时，也属于准双向口。

（2）P3 口的第二功能

当 CPU 不对 P3 口进行字节或位寻址时，单片机内部硬件自动将口锁存器的 Q 端置 1。这时，可以使用 P3 口的第二功能。P3 口第二功能的各引脚的定义如下：

P3.0：串行口输入信号引脚（RXD）。

P3.1：串行口输出信号引脚（TXD）。

P3.2：外部中断 0 输入信号引脚（$\overline{\text{INT0}}$）。

P3.3：外部中断 1 输入信号引脚（$\overline{\text{INT1}}$）。

P3.4：定时/计数器 0 的外部输入信号引脚（T0）。

P3.5：定时/计数器 1 的外部输入信号引脚（T1）。

P3.6：片外数据存储器“写”选通输出信号引脚（$\overline{\text{WR}}$）。

P3.7：片外数据存储器“读”选通输出信号引脚（$\overline{\text{RD}}$）。

有些应用场合若把其中的几条口线设为第二功能，另外几条口线设为第一功能使用。这时宜采用位寻址方式。

2.5.3　并行口的负载能力

P0、P1、P2、P3 口的输入和输出电平与 CMOS 电平和 TTL 电平均兼容。

P0 口的每一位口线可以驱动 8 个 LSTTL 负载。在作为通用 I/O 口时，由于输出驱动电路是开漏方式，驱动时需外接上拉电阻；当作为地址/数据总线使用时，接口线输出不是开漏方式，无需外接上拉电阻。

P1、P2、P3 口的每一位能驱动 4 个 LSTTL 负载。它们的输出驱动电路内部设有上拉电阻，所以可以方便地由集电极开路（OC 门）电路或漏极开路电路所驱动，而无需外接上拉电阻。

由于单片机口线仅能提供几毫安的电流，当作为输出驱动一般晶体管的基极时，应在口与晶体管的基极之间串接限流电阻。

2.5.4 并行口的应用举例

1. 并口作为输出口的应用

利用 P1 口作为输出口控制发光二极管的应用电路，如图 2-9 所示。将 8 个发光二极管（LED）的阳极接在一起，并将它们接在 +5V 电源端，这种接法也称为共阳极接法。发光二极管的阴极通过一个限流电阻分别接到 P1 口的 8 个引脚上。由此可见，若要控制 8 个发光二极管的点亮与熄灭，只要控制 P1 口 8 个引脚的输出电平即可达到目的。当 P1 口的每个引脚输出高电平时，发光二极管均不导通，此时的发光二极管不亮。而当 P1 口的每个引脚输出低电平时，发光二极管导通，发光二极管被点亮。

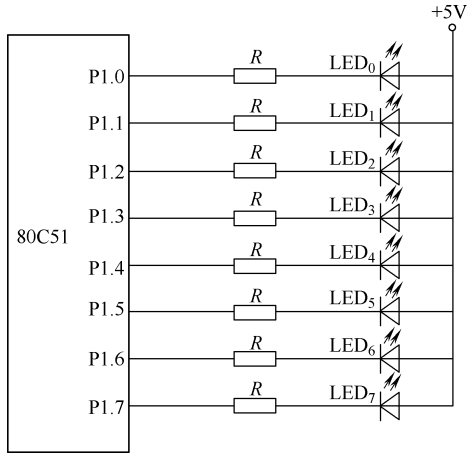


图 2-9 P1 口输出控制 LED 电路

2. 并口作为输入、输出口的应用

并口作为输入、输出口的应用如图 2-10 所示。由图可见，P1 口分别连接了 4 个开关和 4 个发光二极管，组成输入/输出应用电路。4 个开关 S_1 ~ S_4 的一端连在一起并接低电平（接电源地），另一端分别接在 P1 口的 P1.4 ~ P1.7 引脚上，同时通过 4 个上拉电阻接到高电平（+5V）。当对 P1 口进行读操作时，可以读入开关的当前状态，开关状态与 P1 口读入的数字量关系如下：

若开关均处于打开状态，从 +5V 电源→电阻 *R*→开关 S→电源地端，由于开关 S 是打开的不能形成电流回路，电阻 *R* 上无压降产生，所以 P1.4 ~ P1.7 引脚的电平值为 +5V（高电平），此时从 P1.4 ~ P1.7 引脚读入的数字量值为“1111B”。

当开关均处于闭合状态，从 +5V 电源→电阻 *R*→开关 S→电源地端，由于开关 S 是闭合的能够形成电流回路，电阻 *R* 上产生压降（压降

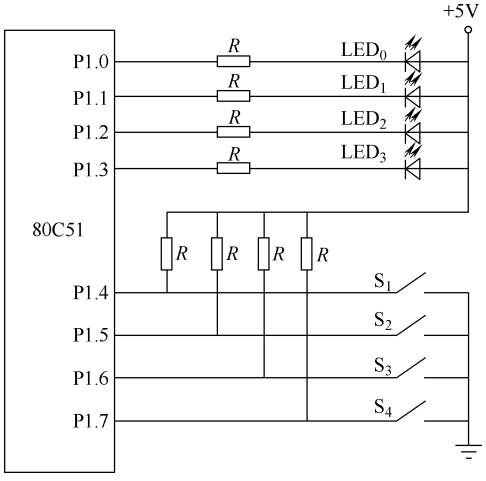


图 2-10 P1 口输入、输出控制电路

为 +5V），开关 S 是闭合后将 P1.4 ~ P1.7 引脚的与电源地连接（低电平），此时从 P1.4 ~ P1.7 引脚读入的数字量值为“0000B”。

P1.0 ~ P1.3 引脚连接 4 个 LED，组成输出控制电路，4 个 LED 的点亮与熄灭控制如前例所述。在此例中，将读入的开关状态，直接从 P1.0 ~ P1.3 引脚输出，可以控制 LED 点亮与熄灭。即改变开关的状态，就可以控制 LED 的状态，通过编程可以实现控制的目的。

在本例中，要注意在读入开关状态之前，要对 P1 口写“1”（使输出端的场效应晶体管截止），然后再读入开关状态。关于并行口的编程参考本书第 3 章内容。

2.6　80C51 单片机的时钟与复位

单片机本身是一个复杂的同步时序系统，为保证同步工作方式的实现，单片机必须有时钟信号，以使其系统在时钟信号的控制下按时序协调工作。而所谓时序，则是指指令执行过程中各信号之间的相互时间关系。

复位是单片机的硬件初始化操作。系统复位后，单片机系统才能开始正常工作。

2.6.1　80C51 单片机的时钟

1. 振荡电路

80C51 芯片中的高增益反相放大器，其输入端为引脚 $XTAL_1$，输出端为引脚 $XTAL_2$。通过这两个引脚在芯片外并联石英晶体振荡器和两只电容器（电容 C_1 和 C_2 一般取 30pF）。石英晶体为感性元件，与电容构成振荡回路，为片内放大器提供正反馈和振荡所需的相移条件，从而构成一个稳定的自激振荡器，如图 2-11 所示。

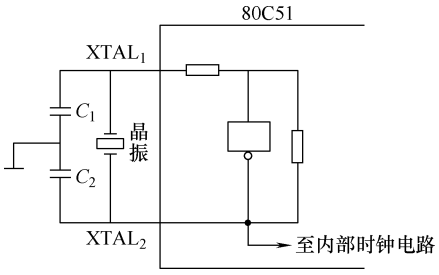


图 2-11　80C51 单片机自激振荡电路

除使用石英晶体振荡器外，若对时钟频率要求不高，还可以用电感或陶瓷振荡器，但使用陶瓷振荡器时要把电容的容量稍微提高一些。

2. 分频电路

振荡电路产生的振荡信号并不直接为单片机所用，而要进行分频，经分频后才能得到单片机各种相关的时钟信号，如图 2-12 所示。

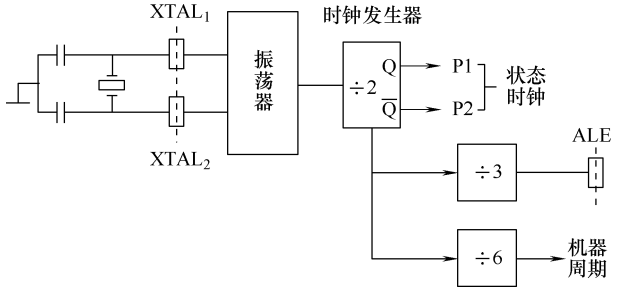


图 2-12　80C51 单片机的时钟电路框图

振荡脉冲经二分频后作为系统的时钟信号（注意时钟信号与振荡脉冲之间的二分频关系，否则会造成概念上的错误），在二分频的基础上再进行三分频产生 ALE 信号，在二分频的基础上再进行六分频得到机器周期信号。

3. 晶振频率

晶振频率是指晶体振荡器的振荡频率，也就是振荡电路的脉冲频率，所以也称振荡频率。80C51 的晶振频率范围一般为 1.2 ~ 33MHz。随着技术的发展，单片机的晶振频率还在逐步提高，如现在一些高速芯片的晶振频率已达 40MHz。晶振频率是单片机的一项重要性能指标。因为单片机的时钟信号是通过振荡信号分频得到的，所以晶振频率直接影响着时钟信号频率。晶振频率高，系统的时钟频率就高，单片机运行速度也就快。晶振频率不但影响速度，而且对单片机的工作电流也有一定的影响，所以在选择晶振频率时，要兼顾速度、功耗和线路工艺。

4. 从外部引入脉冲信号驱动时钟电路

高频振荡信号除了由振荡电路产生外，还可以从外部脉冲源直接引入。直接引入外部脉冲信号的情况多发生在由多片单片机组成的系统中，因为统一从一个外部脉冲源引入脉冲信号，可以保证各单片机之间时钟信号的同步。对于 80C51 芯片，外部脉冲信号经 $XTAL_1$ 引脚注入，但同时要把 $XTAL_2$ 引脚悬空，其连接电路如图 2-13 所示。

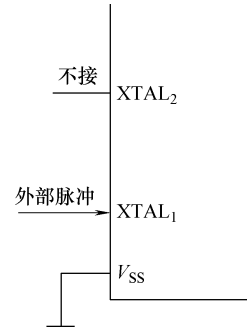


图 2-13　外部脉冲源接法

实际使用时，引入的脉冲信号应为高低电平持续时间大于 20ns 的矩形波，且脉冲频率应低于 12MHz。注意，尽管 80C51 与 8051 兼容，但当使用外部脉冲信号驱动芯片的时钟电路时，应注意它们之间的差别。80C51 的外部脉冲信号经 $XTAL_1$ 引脚接入，而 8051 则是经 $XTAL_2$ 引脚接入。两种芯片之所以有如此差别，是芯片内部的原因，80C51 的时钟电路是由 $XTAL_1$ 引脚信号驱动的，而 8051 则是由 $XTAL_2$ 引脚信号驱动的。

2.6.2　80C51 单片机的定时单位

80C51 的时序就是 80C51 在执行指令时所需控制信号的时间顺序。80C51 单片机的时序定时单位从小到大依次为，时钟周期（状态周期）、机器周期和指令周期。

1. 时钟周期

把晶振脉冲的周期定义为节拍（用 P 表示）。晶振脉冲经过二分频后，得到的振荡脉冲的周期就是单片机的时钟周期（即一个时钟周期是晶振周期的 2 倍），时钟周期也称为状态（用 S 表示）。这样，一个状态就包含两个节拍，具前半周期对应的拍节叫节拍 1（P1），后半周期对应的节拍叫节拍 2（P2）。

2. 机器周期

80C51 采用定时控制方式，因此它有固定的机器周期。规定一个机器周期的宽度为 6 个状态，并依次表示为 S1 ~ S6。由于一个状态又包括两个节拍，因此，一个机器周期总共有 12 个节拍，分别记作 S1P1、S1P2、…、S6P2。由于一个机器周期共有 12 个晶振周期，因此机器周期就是晶振脉冲的十二分频。

当晶振脉冲频率为 12MHz 时，一个机器周期为 1μs；当晶振脉冲频率为 6MHz 时，一个机器周期为 2μs。

3. 指令周期

指令周期是最大的时序定时单位，执行一条指令所需要的时间称为指令周期。它一般由若干个机器周期组成。不同的指令，所需要的机器周期数也不相同。通常，包含一个机器周期的指令称为单周期指令，包含两个机器周期的指令称为双周期指令。指令的运算速度与指令所包含的机器周期有关，机器周期数越少的指令执行速度越快。80C51 单片机通常可以分为单周期指令、双周期指令和四周期指令三种。四周期指令只有乘法和除法指令两条，其余均为单周期和双周期指令。

2.6.3　80C51 单片机的复位方式与初始化状态

复位是单片机的硬件初始化操作。经复位操作后，单片机系统才能开始正常工作。

1. 复位方式

80C51 有复位信号引脚 RST，用于从外界引入复位信号。复位操作比较简单，只有两种复位方式：加电复位和手动复位。

（1）加电复位

加电复位是指通过专用的复位电路产生复位信号。它是系统的原始复位方式，发生在开机加电时，是系统自动完成的。加电复位是任何单片机系统都具有最基本的功能。

（2）手动复位

手动复位也应通过专用的复位电路实现。在单片机系统中，手动复位是必须具有的功能。在调试或运行程序时，若遇到死机、死循环等情况，手动复位是摆脱这种尴尬局面的最常用方法。这时，手动复位所完成的是一次重新启动操作。

在实际系统中，总是把加电复位电路和手动复位电路结合在一起，形成一个既能加电复位，又能手动复位的公用复位电路。另外，目前已经出现了专用的复位芯片，如 Maxim 公司推出的 MAX813L。该芯片具有 4 项基本功能：加电复位、手动复位、看门狗和掉电监视。

2. 初始化状态

复位操作有：为一些专用寄存器设置初始状态、PSW 清 0、PC 被赋值为 0000H，以及为芯片的某些引脚设置电平状态等内容。复位操作后，部分专用寄存器（SFR）的初始化状态见表 2-6。

表 2-6　部分专用寄存器（SFR）的初始化状态

寄　存　器	内　　容	寄　存　器	内　　容
PC	0000H	TCON	00H
ACC	OOH	TL0	00H
PSW	00H	TH0	00H
SP	07H	TL1	00H
DPTR	0000H	TH1	00H
P0 ~ P3	0FFH	SCON	00H
IP	× × ×00000B	SBUF	不定
IE	0 ×000000B	PCON	0 × × × × × × ×0B
TMOD	00H		

完成复位操作共需 24 个状态周期。复位结束后，单片机从地址 0000H 开始执行程序。对于专用寄存器的复位状态，值得关注的是，PC 为 0000H，SP 为 07H，各 I/O 口锁存器为 FFH，SBUF 状态不定，其他寄存器大多被置为 00H。此外，复位操作还对单片机的个别引脚信号有影响。例如，把 ALE 和$\overline{\text{PSEN}}$信号变为无效状态，即 ALE = 0，$\overline{\text{PSEN}}$ = 1。

2.6.4 80C51 单片机的复位电路

复位电路用于产生复位信号，通过 RST 引脚送入单片机，进行复位操作。复位电路的好坏直接影响单片机系统工作的可靠性。

1. 复位电路的分类

目前，在单片机系统中共使用过 4 种类型的复位电路：积分电路型、微分电路型、比较器型和看门狗型。其中前 3 种是在芯片外面用分立元器件或集成电路芯片搭建的；而最后一种位于芯片内部，是单片机芯片的一部分。对于片外复位电路，无论哪种类型，加电复位和手动复位都是必不可少的基本功能。下面把最常用的积分电路型和微分电路型复位电路做一简单说明。

（1）积分型

积分型复位电路是在积分电路的基础上形成的，用于产生低电平复位信号。图 2-14 所示为最基本的积分型复位电路及其演化过程。

（2）微分型

微分型复位电路是在微分电路的基础上形成的，用于产生高电平复位信号。图 2-15 所示为最基本的微分型复位电路。

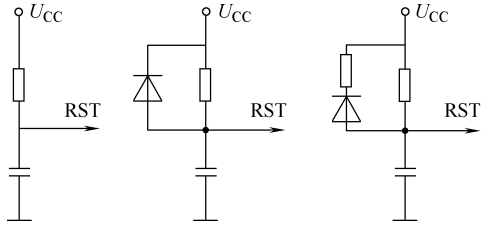


图 2-14 积分型复位电路

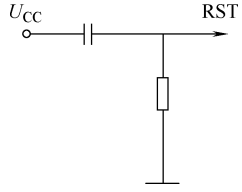


图 2-15 微分型复位电路

2. 80C51 单片机的基本复位电路

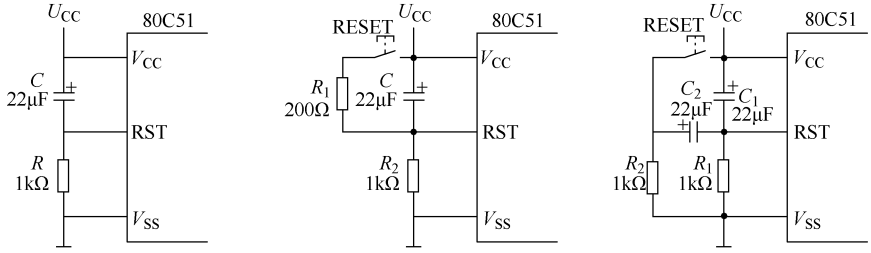
80C51 基本复位电路共有上电复位、按键电平复位和按键脉冲复位 3 种。其中上电自动复位是通过电容充电来实现的，比较简单的上电复位电路如图 2-16a 所示。只要电源 U_{CC} 的上升时间不超过 1ms，就可以实现自动上电复位，即接通电源即可完成系统的复位初始化。

手动复位是通过按键实现的，有电平方式和脉冲方式两种。其中按键电平复位是通过使复位端经电阻与 U_{CC} 电源接通而实现的，电路如图 2-16b 所示。而按键脉冲复位则是利用 *RC* 微分电路产生的正脉冲来实现的，电路如图 2-16c 所示。

上述电路图中的电阻、电容参数适用于 6MHz 晶振，能保证复位信号高电平持续时间大于两个机器周期。

3. 80C51 芯片内复位电路

80C51 的 RST 引脚是复位信号的输入端。复位信号 RST 是高电平有效，其有效时间应



a) 简单的复位电路　　b) 电平方式复位电路　　c) 按键脉冲复位电路

图 2-16　80C51 单片机的基本的复位电路

持续 24 个振荡脉冲周期（即 2 个机器周期）以上。若使用频率为 6MHz 的晶振，则复位信号持续时间应超过 4μs 才能有效。产生芯片内复位信号的电路逻辑如图 2-17 所示。

可见，整个复位电路包括芯片内、外两部分。外部电路产生的复位信号 RST 送施密特触发器，再由片内复位电路在每个机器周期的 S5P2 时刻对施密特触发器的输出进行采样，最后才得到内部复位操作所需要的信号。

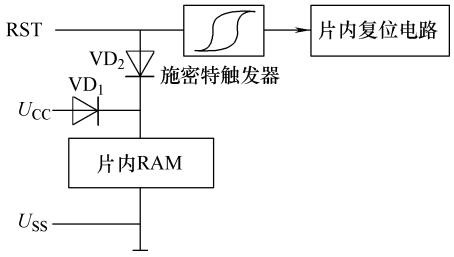


图 2-17　芯片内复位信号的电路

本 章 小 结

80C51 单片机的存储器在物理上设计成程序存储器和数据存储器两个独立的空间。片内程序存储器容量为 0KB/4KB/8KB，中断向量设置在 0003H ~ 0023H 区域内。片内数据存储器为 128B，分成工作寄存器区、位寻址区和用户区，以及 128B 的特殊功能寄存器区。CPU 使用不同指令和寻址方式对其进行相应的操作。

单片机通过复位电路实现上电和复位操作，使单片机进入初始化状态。复位后，PC 内容为 0000H，P0 ~ P3 口内容为 FFH，SP 内容为 07H，SBUF 内容不定，IP、IE 和 PCON 的有效位为 0，其余的特殊功能寄存器的状态均为 00H。

80C51 单片机的时钟信号有内部时钟方式和外部时钟方式两种。内部的各种微操作都以晶振周期为时序基准。一个机器周期包含 12 个晶振周期（或 6 个时钟周期）。指令的执行时间称作指令周期。

80C51 单片机有 4 个 8 位的并行 I/O 口：P0 ~ P3 口。各口均由接口锁存器、输出驱动器和输入缓冲器组成。4 个并行口均可以作为普通 I/O 口使用，具有数据输入/输出功能，注意作为输入口使用时先写 1，然后再读入数据。在需要外部扩展程序存储器和数据存储器时，P0 口作为分时复用的低 8 位地址/数据总线，P2 口作为高 8 位地址总线，P3 口的 P3.6/P3.7 提供“读写”控制信号。P1 口是唯一的单功能口，仅能用作通用的数据输入/输出口。P3 口是双功能口，除具有数据输入/输出功能外，每一个接口还具有不同的第二功能，如 P3.0、P3.1 是串行输入/输出口等。

习　题

1. 80C51 单片机存储器的组织采用何种结构？存储器地址空间如何划分？各地址空间的地址范围和容量如何？在使用上有何特点？

2. 80C51 单片机的 P0～P3 口在结构上有何不同？在使用上有何特点？

3. 如果 80C51 单片机晶振频率分别为 6MHz、11.0592MHz、12MHz 时，机器周期分别为多少？

4. 80C51 单片机的片内、片外存储器如何选择？

5. 80C51 单片机的 PSW 寄存器各位标志的意义如何？

6. 80C51 单片机的当前工作寄存器组如何选择？

7. 80C51 单片机复位后的状态如何？复位方法有几种？

8. 80C51 单片机系统初始化状态，PC、ACC、PSW、P0～P3、TMOD、DPTR、SCON、TCON 等寄存器处于什么状态？

第 3 章　80C51 的指令系统

【学习目的】

学习和应用单片机一个很重要的环节就是理解并熟练掌握它的指令系统。通过本章的学习应达到以下的学习目标：

1. 了解机器语言、汇编语言和高级语言的特点。
2. 掌握汇编语言指令的基本格式，熟悉机器语言指令的格式。
3. 理解 80C51 的 7 种寻址方式及相应的寻址空间，并能实际应用。
4. 熟记 80C51 的 111 条汇编语言指令的形式。
5. 熟悉每条指令的功能、操作的对象和结果；并会根据不同的实践需要选择合适的指令。

3.1　指令概述

指令是 CPU 用来执行某种操作的命令。一条指令只能完成有限的功能，为了使计算机能够完成一定复杂的功能就需要一系列的指令，计算机能够执行的各种指令的集合称为它的指令系统。计算机的总体功能是由指令系统来体现的，一般来说，若一台计算机的指令越丰富、寻址方式越多，且每条指令执行速度都较快，那么它的总体功能就越强。不同型号的计算机的指令系统也不相同。

3.1.1　汇编语言

在计算机中，所有的指令、数据都是用二进制代码来表示的。这种用二进制代码表示的指令系统称为机器语言（Machine Language），用机器语言编写的程序称为机器语言程序或“目标程序”（Object Program）。为了书写方便，二进制代码常用十六进制代码表示。对于计算机，机器语言能被直接识别并快速执行。但对于使用者，这种用机器语言编写的程序很难识别和记忆，容易出错。为了克服这些缺点，出现了汇编语言和高级语言。

用英文字符来代替机器语言，这些英文字符被称为助记符。用助记符表示指令系统的语言称为汇编语言（Assembly Language）。它由字母、数字和符号组成，又称“符号语言”。由于助记符一般都是操作功能的英文缩写，这样使程序易写、易读和易改。可见汇编语言仍是一种面向机器的语言，和 CPU 类别密切相关，不同 CPU 的机器有不同的汇编语言。本章介绍的 80C51 系列单片机程序都是汇编语言形式的。

但是计算机不能直接识别在汇编语言中出现的各种字符，需要将其转换成机器语言，通常把这一转换（翻译）工作称为汇编。汇编可以由查表的形式手工完成，也可由专门的程序来进行，这种程序称为汇编程序。汇编后得到的机器语言程序称为目的程序或目标程序，原来的汇编语言程序称为源程序。

由于汇编语言是一种面向机器的语言，因此受到机器种类的限制，不能在不同类型的计

算机上通用，这样就出现了高级语言，如 BASIC、PASCAL、C 语言等。高级语言是一种面向过程的语言，这种语言更接近英语和数字表达式，易被一般用户掌握。高级语言是独立于机器的，在编程时，用户不需要对机器的硬件结构和指令系统有深入的了解。高级语言直观、易学，通用性强，易于移植到不同类型的机器上去。

计算机对高级语言不能直接识别和执行，需要转换为机器语言，因此它的执行速度比机器语言和汇编语言慢，且占用内存空间大。

因汇编语言运行速度快，占用内存空间小，且易读易记，所以在工业控制中广泛采用的是汇编语言。本章就用 80C51 单片机的汇编语言来描述其指令功能。

3.1.2 指令格式

1. 汇编语言指令的格式

汇编语言指令的一般格式如下：

［标号:］ 操作码助记符 ［操作数］ ［；注释］

其中每条指令必须有操作码助记符，带［ ］的为可选项，可有可无。

标号是表示该指令位置的符号地址，代表该指令第一个字节所存放的存储器单元的地址。它是以英文字母开始的由 1～8 个字母或者数字组成的字符串，并以“:”结尾。通常在子程序入口或者转移指令的目标地址才标号。

操作码助记符是表示指令功能的英文缩写。它是指令的核心部分，不能省略。例如，ADD 是加法的助记符，MOV 是传送的助记符。

操作数是表示指令操作所需要的操作数或者操作数的地址。指令的操作数可以是 1 个、2 个或者 3 个，也可以没有。例如，NOP 指令就没有操作数。操作数之间以“,”分隔，操作码与操作数之间以空格分隔。

注释字段是用户给该条指令或该程序的功能说明，是为了方便阅读程序的一种标注。注释以“;”开始。注释不影响该指令的执行。

2. 机器语言指令的格式

机器语言指令是一种二进制代码，它包括两个基本部分：操作码和操作数。操作码规定了指令操作的性质，操作数则表示指令操作的对象。在 80C51 的指令系统中，有单字节、双字节和三字节共 3 种指令，它们分别占有 1～3 个程序存储器的单元。机器语言指令格式如图 3-1 所示。

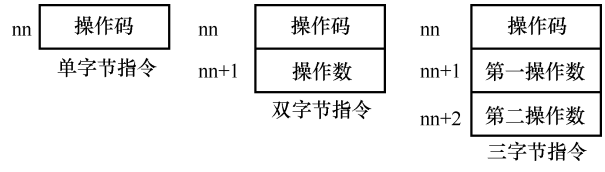


图 3-1 机器语言指令的格式示意图

本章第 3 节中每条指令都有较详细的机器语言格式说明。

3.1.3 常用符号

在描述 80C51 指令系统的功能时，规定了一些描述寄存器、地址及数据等的符号，其意义如下：

Rn 当前选中的工作寄存器组 R0～R7（n 为 0～7）。它在片内数据存储器中的地址由 PSW 中的 RS1、RS0 确定，可以是 00H～07H（第 0 组）、08H～0FH（第 1 组）、10H～17H（第 2 组）、18H～1FH（第 3 组）。

Ri　当前选中的工作寄存器组中可作为地址指针的 2 个工作寄存器 R0、R1（i 为 0 或 1）。它在片内数据存储器中的地址由 RS0、RSl 确定，分别为 00H、01H；08H、09H；10H、11H；18H、19H。

#data　8 位立即数，即包含在指令中的 8 位常数。

#data16　16 位立即数，即包含在指令中的 16 位常数。

direct　8 位片内 RAM 单元（包括 SFR）的直接地址。对于 SFR，此地址可以直接用它的名称来表示，如 ACC（此时不能用 A 代替）、PSW、P0 等。

addr11　11 位目的地址。用于 ACALL 和 AJMP 指令中，目的地址必须放在与下一条指令第 1 个字节同一个 2KB 程序存储器地址空间之内。

addr16　16 位目的地址。用于 LCALL 和 LJMP 指令中，目的地址范围在 64KB 程序存储器地址空间。

rel　补码形式的 8 位地址偏移量，用于相对转移指令中。偏移量以下一条指令第 1 个字节地址为基值，偏移范围为 -128 ~ +127。

bit　片内 RAM 或特殊功能寄存器的直接寻址位地址。

@　在间接寻址方式中，表示间址寄存器的符号。

/　在位操作指令中，表示对该位先取反，再参与操作，但不影响该位原值。

以下符号仅出现在指令注释或功能说明中：

X　片内 RAM 的直接地址（包含位地址）或寄存器。

(X)　在直接寻址方式中，表示直接地址 X 中的内容；在间接寻址方式中，表示由间址寄存器 X 指出的地址单元。

((X))　在间接寻址方式中，表示由间址寄存器 X 指出的地址单元中的内容。

←　在指令操作流程中，将箭头右边的内容送入箭头左边的单元内。

在本章指令注释中，表示寄存器 Rn 或累加器 A、寄存器 B 等中的内容时均不加括号。源操作数中的间址内容用((Ri))表示，但是目的操作数中送入某间址单元用（Ri）表示。注意，不是表示 Ri 中的内容，而是表示 Ri 间址单元里的内容。

3.2　寻址方式

执行任何一条指令都需要使用操作数（空操作除外）。寻址方式就是指在寻找操作数所在地址的方式。在这里，地址泛指一个立即数、某个存储单元或者某个寄存器等。80C51 系列单片机有以下 7 种寻址方式。

3.2.1　立即寻址

立即寻址指在该指令中直接给出参与操作的常数（称为立即数）。立即数前冠以“#”，以便与直接地址相区别。

【例 3-1】　传送指令：MOV　A，#5AH

这条指令的功能是把立即数 5AH 送入到累加器 A 中。指令机器代码为 74H、5AH，双字节指令。在程序存储器中占的地址为 0100H 和 0101H（存放指令的起始地址是任意假设的）。该指令的执行过程如图 3-2a 所示。

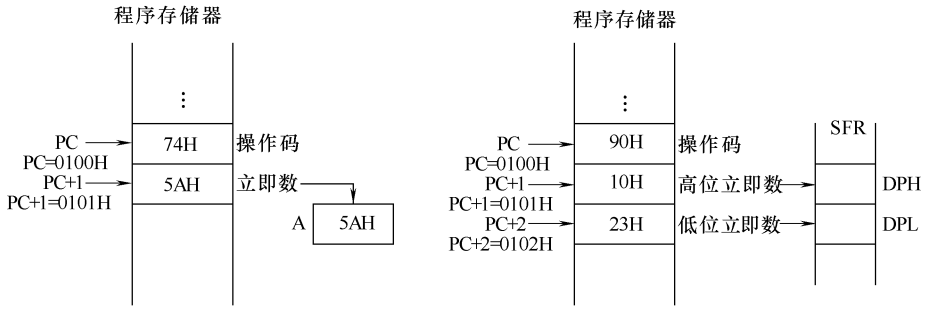
在 80C51 系列指令系统中还有一条 16 的立即数传送指令，即

MOV　DPTR，#data16

该指令是把 16 位立即数 data16 送入数据指针 DPTR 中。DPTR 由两个特殊功能寄存器 DPH 和 DPL 组成。立即数的高 8 位送入 DPH 中，低 8 位送入 DPL 中。

【例 3-2】 16 位传送指令：MOV　DPTR，#1023H

这条指令的功能是把 16 位的立即数送入 DPTR 中。其中高字节 10H 送入 DPH 中，低字节 23H 送入 DPL 中。指令的机器代码为 90H、10H、23H，是三字节指令，在程序存储器中占的地址为 0100H、0101H 及 0102H。该指令的执行过程如图 3-2b 所示。



a) MOV A, #5AH　　b) MOV DPTR, #1023H

图 3-2　立即寻址示意图

3.2.2　直接寻址

直接寻址就是在指令中直接给出操作数所在存储单元的地址，该地址指出了参与操作的数据所在的字节地址或者是位地址。在 80C51 单片机中，直接地址只能用来表示特殊功能寄存器。内部数据存储器和位地址空间。其中，特殊功能寄存器和位地址空间只能用直接寻址方式来访问。

【例 3-3】 传送指令：MOV　A，30H

这条指令的功能是把内部 RAM30H 单元的内容送入 A 中（注意，内部 RAM 地址为 30H 单元中的内容可以是 00H～0FFH 范围内的任意一个数）。指令代码为 E5H、30H，为双字节指令。在程序存储器中占的空间及寻址示意图如图 3-3 所示。

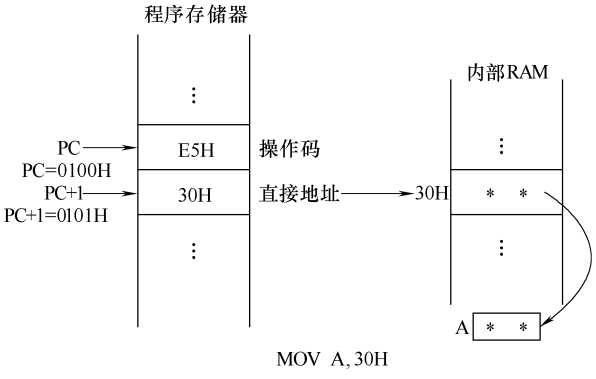


图 3-3　直接寻址示意图

3.2.3　寄存器寻址

在指令中指出某个寄存器（Rn，A，B 和 DPTR 等）中的内容作为操作数，这种寻址方式称为寄存器寻址。采用寄存器寻址可以获得较高的运算速度。

【例 3-4】 传送指令：MOV　A，R5

这条指令的功能是把寄存器 R5 的内容送入到累加器 A 中。指令的代码为 0EDH，单字节指令。其寻址示意图如图 3-4 所示。

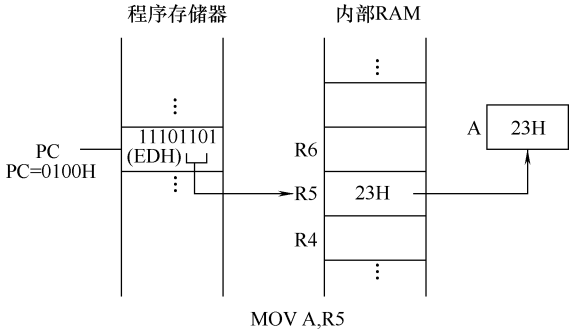


图 3-4　寄存器寻址示意图

3.2.4　寄存器间接寻址

寄存器间接寻址是指把指令中指定的寄存器的内容作为操作数的地址，把该地址对应单元中的内容作为操作数。这种寻址方式适于访问内部 RAM 和外部 RAM。可以看出，在寄存器寻址中寄存器的内容作为操作数，但是寄存器间接寻址方式中，寄存器中存放的是操作数的地址。即指令操作数的获得是通过寄存器间接得到的。为了区别寄存器寻址和寄存器间接寻址，在寄存器间接寻址中应在寄存器名称的前面加间址符“@”。

在访问内部 RAM 的 00H ~ 7FH 地址单元时，用当前工做寄存器 R0 或 R1 做地址指针来间接寻址。对于栈操作指令 PUSH 和 POP，则用堆栈指针 SP 进行寄存器间接寻址。

在访问外部 RAM 的页内 256 个单元（00H ~ FFH）时，用 R0 或 R1 工作寄存器来间接寻址。在访问外部 RAM 整个 64K（0000H ~ FFFFH）地址空间时，用数据指针 DPTR 来间接寻址。

【例 3-5】　传送指令：MOV　A，@R1

这条指令属于寄存器间接寻址。它的功能是将寄存器 R1 的内容（设 R1 = 75H）作为地址，再将片内 RAM 的 75H 单元的内容（设（75H）= 37H）送入累加器 A 中。指令中在寄存器名前冠以“@”，表示寄存器间接寻址，称之为间址符。指令代码为 0E7H，单字节指令。其寻址示意图如图 3-5 所示。

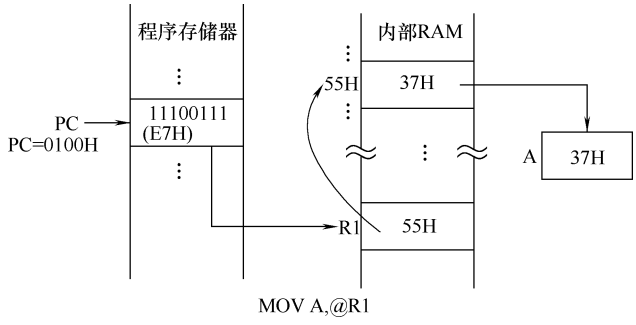


图 3-5　寄存器间接寻址示意图

3.2.5　变址寻址

变址寻址以程序计数器 PC 或数据指针 DPTR 作为基地址寄存器，以累加器 A 作为变址寄存器，把两者的内容相加形成操作数的地址（16 位）。这种寻址方式用于读取程序存储器中的常数表。

【例 3-6】　查表指令：MOVC　A,@ A + DPTR

这条指令的其功能是把 DPTR 的内容作为基地址，把累加器 A 中的内容作为地址偏移量，两者相加后得到 16 位地址，把该地址对应的程序存储器 ROM 单元中的内容送到累加器 A 中。指令代码为 93H，单字节指令。其寻址过程示意图如图 3-6 所示。

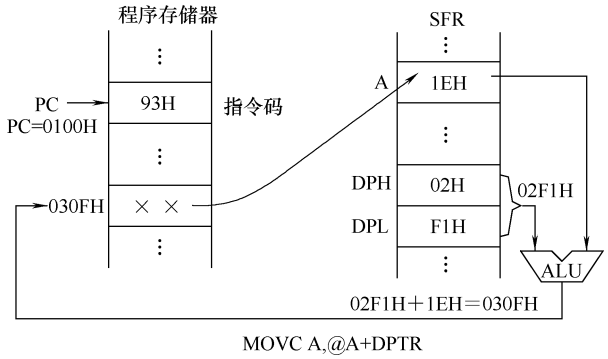


图 3-6　变址寻址示意图

3.2.6　相对寻址

相对寻址以程序计数器 PC 的当前值作为基地址，与指令中给定的相对偏移量 rel 进行相加，把所得之和作为程序的转移地址。这种寻址方式用于相对转移指令中。指令中的相对偏移量是一个 8 位带符号数，用补码表示。

【例 3-7】　累加器 A 内容判零指令：JZ　30H

这条指令是以累加器 A 的内容是否为零作为条件的相对转移指令，指令代码为 60H、30H，为两字节指令。其功能为：当 A = 0 时，条件满足，则程序执行发生转移 PC←PC + 2 + rel；当 A≠0 时，条件不能满足，则程序顺序执行 PC←PC + 2。其寻址示意图如图 3-7 所示。

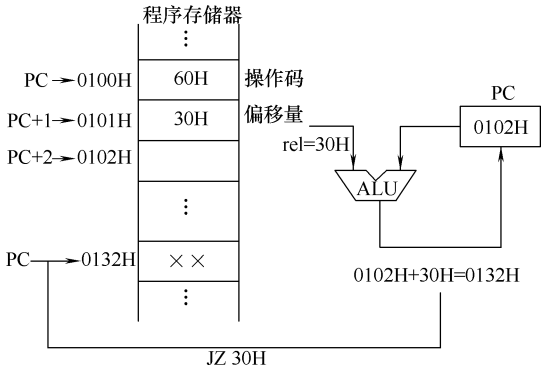


图 3-7　相对寻址示意图

80C51 指令系统中，相对寻址指令多数为 2 字节指令，执行完相对寻址指令后，当前的 PC 值应该为这条指令首字节所在单元的地址值（源地址）加 2，所以偏移量应该为

$$rel = 目的地址 - (源地址 + 2)$$

但也有一些是 3 字节的相对寻址指令（如 CJNE　A，direct，rel），那么执行完这条指令后，当前的 PC 值应该为本指令首字节所在单元的地址值加 3，所以偏移量为

$$rel = 目的地址 - (源地址 + 3)$$

相对偏移量 rel 是一个带符号的 8 位二进制数，以补码形式出现。因此，程序的转移范围在相对 PC 当前值的 -128 ~ +127 个字节单元之间。

3.2.7　位寻址

80C51 单片机中设有独立的位处理器。位操作指令能对内部 RAM 中的位寻址区和某些有位地址的特殊功能寄存器进行位操作。也就是说可对位地址空间的每个位进行位变量传送、状态控制、逻辑运算等操作。

【例 3-8】　位传送指令：MOV　C，04H

这条指令的功能是把位地址 04H 中的内容传送到 Cy 中（即把内部 RAM 的 20H 单元的

D_4 位（位地址为 04H）的内容传送到位累加器 C 中）。指令代码为 0A2H、04H，为双字节指令。其寻址示意图如图 3-8 所示。

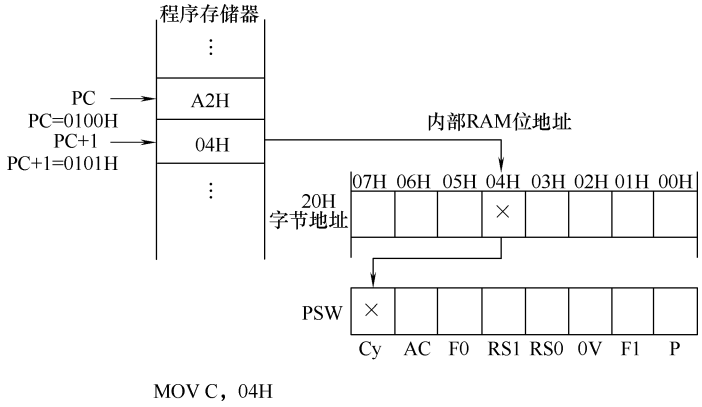


图 3-8　位寻址示意图

上面介绍了 80C51 指令系统的 7 种寻址方式。实际上许多指令本身包含着两个或三个操作数，这时往往就具有几种类型的寻址方式。这里面重点讨论的是源操作数的寻址方式。

【例 3-9】 传送指令：MOV　　A，　　#4FH
　　寄存器寻址　　立即寻址

这条指令的功能是把 4FH 这个立即数送入到累加器 A 中。其中源操作数为立即寻址，目标操作数为寄存器寻址。

比较不相等则转移指令：CJNE　　A，　　30H，　　NEXT
　　寄存器寻址　　直接寻址　　相对寻址

这是条件转移指令中的比较不相等则转移指令，其功能为比较累加器 A 的内容与直接地址 30H 的内容是否相等，如果不相等则加上偏移量 rel 转移到 NEXT 位置，如果相等则顺序执行。

3.2.8　寻址空间

80C51 单片机指令系统一共有 7 种寻址方式，每种寻址方式都有自己使用的变量和适用的寻址空间，见表 3-1。根据不同的存储器或者存储器中不同的位置分别采用不同的寻址方式，这是 80C51 单片机指令系统的特点，在以后学习的过程中应注意区分。

表 3-1　80C51 中的寻址方式和寻址空间

序　　号	寻 址 方 式	使用的变量	寻 址 空 间
1	立即寻址		程序存储器
2	直接寻址		片内 RAM 低 128B 特殊功能寄存器
3	寄存器寻址	R0 ~ R7、A、B、DPTR、CY	
4	寄存器 间接寻址	@ R0、@ R1、SP	片内 RAM
		@ R0、@ R1、@ DPTR	片外 RAM

（续）

序　　号	寻 址 方 式	使用的变量	寻 址 空 间
5	相对寻址	PC + 偏移量	程序存储器
6	变址寻址	@ A + PC、@ A + DPTR	程序存储器
7	位寻址		片内 RAM 中的位寻址区可以位寻址的特殊功能寄存器位

3.3　80C51 的指令系统

80C51 的指令系统使用了 7 种寻址方式，共有 111 条指令。若按字节数分类，则单字节指令 49 条，双字节 46 条，3 字节指令 16 条；若按运算速度分类，则单周期指令 64 条，双周期 45 条，4 周期指令 2 条。由此可见，80C51 指令系统在占用存储空间和运行时间方面，效率都比较高。按照指令的功能来分类，80C51 指令系统可分为下面的 5 类：

① 数据传送类指令（28 条）。

② 算术运算类指令（24 条）。

③ 逻辑运算类指令（25 条）。

④ 控制转移类指令（17 条）。

⑤ 位操作类指令（17 条）。

3.3.1　数据传送指令

数据传送指令把第二个“源操作数”中的数据传送到第一个“目的操作数”中去，而“源操作数”的内容保持不变。这类指令在程序中占有较大的比重，是一种最基本最常用的操作。

1. 对内部 RAM 和 SFR 之间的数据传送指令

80C51 内部 RAM 和特殊功能寄存器 SFR 各存储单元之间的数据传送，通常是通过 MOV 指令来实现的，这类指令称为内部 RAM 和 SFR 的一般数据传送指令。其传送操作示意图如图 3-9 所示。

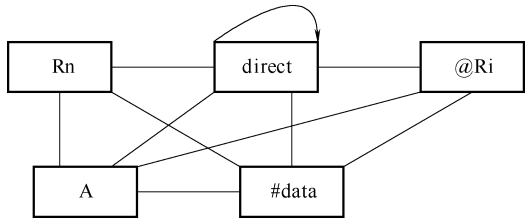


图 3-9　内部 RAM 和 SFR 之间的传送操作示意图

（1）以累加器为目的操作数的指令（见表 3-2）

表 3-2　以累加器为目的操作数的指令

指 令 名 称	汇 编 格 式	操　　作	机　器　码	机 器 周 期
以累加器 A 为目的操作数	MOV　A，Rn	A←(Rn)	E8H ~ EFH	1
	MOV　A，direct	A←(direct)	85H direct	1
	MOV　A，@ Ri	A←((Ri))	E6H ~ E7H	1
	MOV　A，#data	A←data	74H data	1

这类指令是把源操作数送入目的操作数 A 中，源操作数的寻址方式分别为寄存器寻址、直接寻址、寄存器间接寻址和立即寻址。

【例 3-10】 若 R1 = 21H，（21H）= 55H，执行指令“MOV　A，@ R1”后的结果为，A = 55H，而 R1 的内容和 21H 单元的内容均不变。

（2）以寄存器为目的操作数的指令（见表 3-3）

表 3-3　以寄存器为目的操作数的指令

指令名称	汇编格式	操　作	机器码	机器周期
以寄存器 Rn 为目的操作数	MOV　Rn，A	Rn←A	F8H ~ FFH	1
	MOV　Rn，direct	Rn←(direct)	A8H ~ AFH direct	2
	MOV　Rn，#data	Rn←data	78H ~ 7FH data	1

这类指令是把源操作数送入目的操作数 Rn 中，源操作数的寻址方式分别为寄存器寻址、直接寻址和立即寻址。

【例 3-11】 若（50H）= 45H，R5 = 33H，执行指令“MOV　R5，50H”后的结果为，R5 = 45H，50H 单元的内容不变。

（3）以直接地址为目的操作数的指令（见表 3-4）

表 3-4　以直接地址为目的操作数的指令

指令名称	汇编格式	操　作	机器码	机器周期
以直接地址为目的操作数	MOV　direct，A	(direct)←A	F5H direct	1
	MOV　direct，Rn	(direct)←Rn	88H ~ 8FH	2
	MOV　direct2，direct1	(direct2)←(direct1)	85H direct1 direct2	2
	MOV　direct，@ Ri	(direct)←((Ri))	86H ~ 87H direct	2
	MOV　direct，#data	(direct)←data	75H direct data	2

这类指令的功能是把源操作数送入目的操作数 direct 中，源操作数的寻址方式分别为寄存器寻址、直接寻址、寄存器间接寻址和立即寻址。

直接地址之间的直接传送指令生产机器码时源地址在前，目的地址在后。如“MOV 40H，41H”对应的机器码为 85H、41H、40H。

【例 3-12】 若 R0 = 50H，（50H）= 6AH，（70H）= 2FH，执行指令“MOV　70H，@ R0”后的结果为，（70H）= 6AH，R0 中的内容和 50H 单元的内容不变。

（4）以寄存器间接地址为目的操作数的指令（见表 3-5）

表 3-5 以寄存器间接地址为目的操作数的指令

指令名称	汇编格式	操作	机器码	机器周期
以寄存器间接地址为目的操作数	MOV @Ri，A	(Ri)←A	F6H～F7H	2
	MOV @Ri，direct	(Ri)←(direct)	A6H～A7H direct	2
	MOV @Ri，#data	(Ri)←data	76H～77H data	1

这类指令的功能是把源操作数送入目的操作数@Ri 中，源操作数的寻址方式分别为寄存器寻址，直接寻址和立即寻址。提醒一下读者，在这类指令中，目的操作数@Ri 中的表示方法为单括号，并不表示 Ri 中的内容，而是表示间址寄存器 Ri 的地址单元。

【例 3-13】 若 R1＝30H，(30H)＝22H，A＝34H，执行指令“MOV @R1，A”后的结果为，(30H)＝34H，R1 和 A 当中的内容不变。

（5）16 位数据的传送指令（见表 3-6）

表 3-6 16 位数据的传送指令

指令名称	汇编格式	操作	机器码	机器周期
16 位数据传送	MOV DPTR，#data16	DPH←data15～8 DPL←data7～0	90H data15～8 data7～0	2

这条指令是唯一的一条 16 位传送指令，通常用来给 DPTR 赋初值。源操作数的寻址方式为立即寻址。读者可以参看本章寻址方式中【例 3-2】。

有关内部 RAM 和 SFR 之间的数据传送指令的说明：

① 指令操作数为 Rn 时，属于寄存器寻址。80C51 内部 RAM 区中有 4 组工作寄存器，每组由 8 个寄存器组成，用户可通过改变 PSW 中的 RS0 和 RS1 这两位来切换当前工作寄存器组。

② 指令中操作数为@Ri 时，属于寄存器 R0 或 R1 间接寻址。它根据操作码字节中 D_0 位 i 的取值为 0 或 1，来决定以哪个寄存器进行间接寻址。

③ 直接寻址的数传指令比较丰富，使得内部数据存储器各单元之间的数传十分方便。特别令人感兴趣的是直接地址到直接地址的传送。直接地址 direct 是 8 位地址，原则上寻址范围为 00H～FFH。对 80C51 来说，内部 RAM 地址空间是 00H～7FH。而 21 个 SFR 离散地分布在 80H～FFH 的地址空间中，有许多单元是无定义的。因此 direct 不能为无定义的内部数据存储单元的地址，否则将得不到正确的结果。

④ 数据传送指令都不影响标志位（除目的操作数是累加器 A 时会根据传送来的数据改变奇偶标志外）。

2. 累加器 A 与外部数据存储器传送指令

CPU 与外部 RAM 的数据传送指令，其助记符为 MOVX，其中的 X 就是单词 external（外部）的第二个字母，表示访问外部 RAM。这类指令共有 4 条，见表 3-7。

表 3-7　累加器 A 与外部数据存储器传送指令

指令名称	汇编格式	操　作	机　器　码	机器周期
累加器 A 与外部 RAM 的数据传送	MOVX　A，@DPTR	A←((DPTR))	E0H	2
	MOVX　@DPTR，A	((DPTR))←A	F0H	2
	MOVX　A，@Ri	A←((Ri))+((P2))	E2H～E3H	2
	MOVX　@Ri，A	((Ri))+((P2))←A	F2H～F3H	2

这组指令的功能是，在累加器 A 与外部 RAM 或扩展 I/O 口之间进行数据传送，且仅为寄存器间接寻址。80C51 只能用这种方式与连接在扩展 I/O 口的外部设备进行数据传送。

前两条指令以 DPTR 作为外部 RAM 的 16 位地址指针，由 P0 口送出低 8 位地址，由 P2 口送出高 8 位地址，寻址能力为 64KB。后 2 条指令用 R0 或 R1 作外部 RAM 的低 8 位地址指针，由 P0 口送出地址码，P2 口的状态不受影响，寻址能力为外部 RAM 空间 256 个字节单元。

【例 3-14】 若 DPTR=1020H，外部 RAM（1020H）=54H，执行指令“MOVX　A，@DPTR”的结果为 A=54H，DPTR 的内容和外部 RAM1020H 单元的内容不变。

【例 3-15】 若 P2=03H，R1=40H，A=7FH，执行指令“MOVX　@R1，A”后的结果为外部 RAM（0340H）=7FH，P2 和 R1 及 A 中内容不变。

【例 3-16】 把外部数据存储器 2040H 单元的内容送入内部寄存器 R2 中。

```
MOV　DPTR，#2040H
MOVX　A，@DPTR
MOV　R2，A
```

3. 累加器 A 与程序存储器的传送指令

80C51 指令系统提供了两条累加器 A 与程序存储器的数传指令，指令助记符采用 MOVC，其中 C 就是单词 code（代码）的第一字母，表示读取 ROM 中的代码。这是两条极为有用的查表指令，见表 3-8。

表 3-8　累加器 A 与程序存储器的传送指令

指令名称	汇编格式	操　作	机　器　码	机器周期
查表	MOVC　A，@A+PC	PC←PC+1 A←(A+PC)	83H	2
	MOVC　A，@A+DPTR	A←(A+DPTR)	93H	2

第一条指令为单字节指令，CPU 读取本指令后，PC 已执行加 1 操作，指向下一条指令的首字节地址。该指令以 PC 作为基址寄存器，累加器 A 的内容为无符号整数，两者相加得到一个 16 位地址，把该地址指出的程序存储器单元的内容送到累加器 A 中（见表 3-8）。

【例 3-17】 设 A=35H，执行指令“1000H：MOVC　A，@A+PC”后的结果：

首先把累加器 A 中的内容加上本条指令执行后的 PC 值 1001H，然后将程序存储器 1036H 单元的内容送入累加器 A 中，即 $A\leftarrow(1036H)_{ROM}$。

本指令的优点是不改变 PC 的状态，仅根据累加器 A 的内容就可以取出表格中的数据。缺点是表格只能存放在该查表指令后面的 256 个单元之内，表格的长度受到限制，而且表格

只能被一段程序所使用。

第二条指令以 DPTR 作为基址寄存器，累加器 A 的内容作为无符号数，两者相加后得到一个 16 位地址，把该地址指出的程序存储器单元的内容送到累加器 A 中。

【例 3-18】 设 DPTR = 2010H，A = 40H，执行指令“MOVC A，@ A + DPTR”后的结果：

首先把累加器 A 与 DPTR 的内容相加得 2050H，然后将程序存储器中 2050H 单元中的内容送入累加器 A 中，即 A←(2050H)$_{ROM}$。

本查表指令的执行结果只与 DPTR 和 A 的内容有关，与该指令存放的地址及表格存放的地址无关。因此表格的长度和位置可以在 64KB 的程序存储器空间任意改变，而且一个表格可以被多个程序段共享。

【例 3-19】 把外部数据存储器 2042H 的内容送入内部 RAM 的 50H 中。

方法一：

```
MOV   DPTR，#2042H
MOVX  A，@ DPTR；外部 RAM 2042H 单元的内容中送入 A 中
MOV   50H，A；由 A 送入内部 RAM 50H 中
```

方法二：

```
MOV   P2，#20H；地址的高 8 位由 P2 口送出
MOV   R0，#42H
MOVX  A，@ R0；把外部 RAM 2042H 的内容送入 A 中
MOV   50H，A；由 A 送入内部 RAM 50H 中
```

【例 3-20】 把程序存储器 0150H 单元的内容取出送到外部 RAM 的 1070H 中。

```
MOV   DPTR，#0150H
MOV   A，#00H
MOVC  A，@ A + DPTR；程序存储器 0150H 的内容取到 A 中
MOV   DPTR，#1070H
MOVX  @ DPTR，A；A 的内容送入外部 RAM 1070H 单元中
```

4. 数据交换指令

数据传送指令还提供了 4 条数据交换指令，包括 3 条字节交换指令和 1 条半字节交换指令。

（1）字节交换指令（见表 3-9）

表 3-9 字节交换指令

指令名称	汇编格式	操作	机器码	机器周期
字节交换	XCH A，Rn	A⇆Rn	C8H ~ CFH	1
	XCH A，direct	A⇆（direct）	C5H direct	1
	XCH A，@ Ri	A⇆((Ri))	C7H	1

字节交换指令的功能是将累加器 A 的内容与内部 RAM 中任何一个单元的内容相互交换。其操作也可表示为

【例 3-21】 若 A = 7AH，R1 = 45H，(45H) = 39H，执行指令“XCH　A，R1”后的结果为，A = 45H，R1 = 7AH。

若 A = 7AH，R1 = 45H，(45H) = 39H，执行指令 XCH　A，@ R1 后的结果为 A = 39H，(45H) = 7AH，R1 = 45H。

（2）低半字节交换指令（见表 3-10）

表 3-10　低半字节交换指令

指令名称	汇编格式	操　作	机器码	机器周期
低半字节交换	XCHD　A，@ Ri	$A_{0\sim3} \leftrightarrows ((Ri))_{0\sim3}$	D6H ~ D7H	1

这条指令的功能是将累加器 A 的低 4 位与 Ri 间接寻址单元的低 4 位相互交换，而各自的高 4 位维持不变。其操作表示为

【例 3-22】 设 A = 59H，R0 = 45H，(45H) = 7AH，执行指令“XCHD　A，@ R0”后的结果为，A = 5AH，R0 = 45H（不变），(45H) = 79H。

5. 堆栈操作指令

在 80C51 内部 RAM 中可以设定一个 LIFO（后进先出）或 FILO（先进后出）区域作为堆栈，在 SFR 中有一堆栈指针（8 位寄存器），它指出栈顶的位置。在 80C51 指令系统中，有两条用于数据传送的栈操作指令，见表 3-11。

表 3-11　堆栈操作指令

指令名称	汇编格式	操　作	机器码	机器周期
进栈	PUSH direct	SP←SP + 1 (SP)←(direct)	C0H direct	2
出栈	POP direct	(direct)←((SP)) SP←SP - 1	D0H direct	2

堆栈是在 RAM 中设定的存储区，栈底是固定的，栈顶是浮动的，所有的信息存入和取出都是在浮动的栈进行的。存取数据依据“先入后出、后入先出”的规则。堆栈技术在子程序嵌套时常用于保存断点，在多级中断时用来保存断点和现场等。用堆栈指令也可以实现内部 RAM 单元之间的数据传送和交换。

【例 3-23】 在中断处理时堆栈指令用于保护现场和恢复现场。设 SP = 60H，中断服务程序的一般结构如下：

```
PUSH　ACC　　;　SP←SP + 1 (SP = 61H)
```

```
                    (61H)←ACC
PUSH  PSW     ;  SP←SP+1 (SP=62H)
                    (62H)←PSW
      ⋮       ;  中断处理
POP   PSW     ;  PSW←(62H)
                    SP←SP-1 (SP=61H)
POP   ACC     ;  A←(61H)
                    SP←SP-1 (SP=60H)
RETI          ;  中断返回
```

【例 3-24】 设 (30H)=51H，(40H)=6AH，将内部 RAM 的这两个单元的内容交换。

```
PUSH  30H     ; 30H 单元的内容进栈
PUSH  40H     ; 40H 单元的内容进栈
POP   30H     ; 将栈顶元素弹出，送入 30H 单元
POP   40H     ; 再将下一个元素出栈，送入 40H 单元
```

执行结果为，(30H)=6AH，(40H)=51H。

3.3.2 算术运算指令

80C51 的算术运算指令也比较丰富，包括加、减、乘、除法指令，数据运算功能较强。

80C51 的算术运算指令，仅直接执行 8 位数的算术操作。指令的执行结果将使 PSW 中的进位标志 Cy、半进位标志 AC 和溢出标志 OV 置位或复位，只有加 1 和减 1 指令不影响这些标志，乘除指令不影响 AC 标志位。注意，无论执行何种指令，PSW 中的奇偶标志 P 总是表示累加器 A 的奇偶性。

1. 加法指令

(1) 不带进位的加法指令 (见表 3-12)

表 3-12 不带进位的加法指令

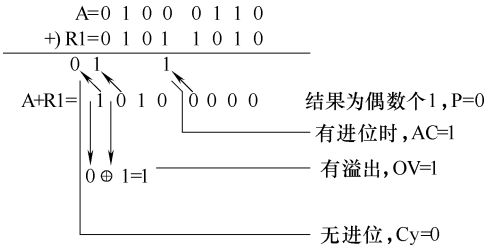
指令名称	汇编格式	操作	机器码	机器周期
不带进位加法	ADD A，Rn	A←A+Rn	28H~2FH	1
	ADD A，direct	A←A+(direct)	25H direct	1
	ADD A，@Ri	A←A+((Ri))	26H~27H	1
	ADD A，#data	A←A+data	24H	1

这类指令的功能是把所指出的字节变量加到累加器 A 中去，运算结果存放在累加器 A 中。

加法指令对 PSW 各标志位产生影响，在相加的结果中，如果 D_7 有进位，则 Cy=1，否则 Cy=0；如果 D_3 有进位，则 AC=1，否则 AC=0；如果 D_6 有进位而 D_7 没进位，或者 D_7 有进位而 D_6 没进位，则 OV=1，否则 OV=0；如果相加结果在 A 中 1 的个数为奇数，则 P=1，否则 P=0。

【例 3-25】 设 A=46H，R1=5AH，试分析执行指令“ADD A，R1 ；A←A+R1”

后的结果以及对标志位的影响。



结果为，A = A0H，R1 = 5AH（不变）。

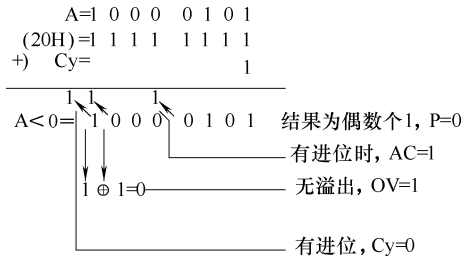
（2）带进位的加法指令（见表 3-13）

表 3-13　带进位的加法指令

指令名称	汇编格式	操　作	机器码	机器周期
带进位加法	ADDC　A，Rn	A←A + Rn + Cy	38H ~ 3FH	1
	ADDC　A，direct	A←A + (direct) + Cy	35H direct	1
	ADDC　A，@Ri	A←A + ((Ri)) + Cy	36H ~ 37H	1
	ADDC　A，#data	A←A + data + Cy	34H data	1

这类指令的功能是同时把所指出的字节变量、进位标志 Cy 和累加器 A 的内容相加，相加后的结果存放在累加器 A 中。

【例 3-26】 设 A = 85H，(20H) = FFH，Cy = 1，试分析执行指令“ADDC　A，20H；A←A + (20H) + Cy”后的结果以及对标志位的影响。



结果为，A = 85H，(20H) = FFH（不变）。

【例 3-27】 编写程序计算 1234H + 0FE7H，将和的高 8 位存入 31H 中，低 8 位存入 30H 中。

两个 16 位数相加可以分为 2 步：第 1 步先加低 8 位，第 2 步对高 8 位相加。由于高 8 位相加时候需要考虑到低 8 位可能产生的进位，第 2 步必须用带进位的加法指令。

```
MOV　A，#34H
ADD　A，#0E7H
MOV　30H，A
MOV　A，#12H
```

ADDC A，#0FH

MOV 31H，A

（3）加 1 指令（见表 3-14）

表 3-14 加 1 指令

指令名称	汇编格式	操作	机器码	机器周期
加 1	INC A	A←A + 1	04H	1
	INC Rn	Rn←Rn + 1	08H ~ 0FH	1
	INC direct	(direct)←(direct) + 1	05H direct	1
	INC @Ri	(Ri)←((Ri)) + 1	06H ~ 07H	1
	INC DPTR	DPTR←DPTR + 1	A3H	2

这类指令的功能是把操作数指定单元的内容加 1（提醒读者加 1 指令是单操作数的指令），除奇偶标志外，操作结果不影响 PSW 中的标志位。若原来单元的内容为 FFH 时，加 1 后将溢出为 00H。

【例 3-28】 将累加器 A 的内容加 1，有以下两种方法：

INC A ；单字节指令，只影响奇偶标志 P，不影响其他标志位。

或 ADD A，#01H ；双字节指令，影响 PSW 各标志位（Cy，OV，AC，P）。

从标志位状态和指令长度来看，这两条指令是不等价的。

2. 减法指令

减法指令 80C51 指令系统中仅有带借位的减法和减 1 指令。

（1）带借位减法指令（见表 3-15）

表 3-15 带借位减法指令

指令名称	汇编格式	操作	机器码	机器周期
带借位减法	SUBB A，Rn	A←A - Rn - Cy	98H ~ 9FH	1
	SUBB A，direct	A←A - (direct) - Cy	95H direct	1
	SUBB A，@Ri	A←A - ((Ri)) - Cy	96H ~ 97H	1
	SUBB A，#data	A←A - data - Cy	94H data	1

这组指令的功能是从累加器 A 中减去指定变量及借位 Cy 的值，差值存放在累加器 A 中。

在 80C51 指令系统中没有提供不带借位减法指令，但在“SUBB”指令之前加一条“CLR C”指令先将 Cy 清零，可以实现不带借位减法的功能。

带借位的减法指令对 PSW 各标志位产生影响。在相减时，如果 D_7 位有借位，则 Cy = 1，否则 Cy = 0；若 D_3 位有借位，则 AC = 1，否则 AC = 0；如果 D_6 位需借位而 D_7 位不需借位，或者 D_7 位需借位而 D_6 位不需借位，则 OV = 1，否则 OV = 0；如果 A 中（相减的差）1 的个数为奇数，则 P = 1，否则 P = 0。

【例 3-29】 设 A = C9H，R2 = 54H，Cy = 1，试分析执行指令“SUBB A，R2 ；A←

A－R2－Cy”后的结果以及对标志位的影响。

```
  A=1 1 0 0 1 0 0 1
-)R2=0 1 0 1 0 1 0 0
-)Cy=
  A<=0 1 1 1 0 1 0 0
```

结果为，A＝74H，R2＝54H（不变）；Cy＝0，AC＝0，OV＝1，P＝0。

本例中，若看作两个无符号数相减，差为 74H，是正确的；若看作两个带符号数相减，则从负数减去一个正数，结果为正数是错误的，OV＝1 表示运算有溢出。

（2）减 1 指令（见表 3-16）

表 3-16　减 1 指令

指令名称	汇编格式	操　作	机器码	机器周期
减 1	DEC　A	A←A－1	14H	1
	DEC　Rn	Rn←Rn－1	18H～1FH	1
	DEC　direct	(direct)←(direct)－1	15H direct	1
	DEC　@Ri	(Ri)←((Ri))－1	16H～17H	1

减 1 指令的功能是，将操作数指定单元的内容减 1。除奇偶标志外，操作结果不影响 PSW 的标志位。若原单元的内容为 00H，减 1 后下溢为 FFH。其他情况与加 1 指令类同。

3. 乘法指令（见表 3-17）

表 3-17　乘法指令

指令名称	汇编格式	操　作	机器码	机器周期
乘法	MUL　AB	BA←A×B	A4H	4

这条指令的功能是把累加器 A 和 B 寄存器中的两个 8 位无符号数相乘，把 16 位乘积的低 8 位存放在累加器 A 中，高 8 位存放寄存器 B 中。如果乘积大于 255（即 FFH），则溢出标志位 OV 置 1，否则为 0；进位标志总是为 0；半进位标志 AC 保持不变；奇偶标志 P 仍按 A 中的 1 的奇偶性来确定。

【例 3-30】 设 A＝32H（即 50），B＝60（即 96），执行指令“MUL　AB；BA←A×B”后的结果：

乘积为 12C0H（即 4800）＞FFH（即 255）

A＝C0H，B＝12H

标志位 Cy＝0，OV＝1，P＝0

4. 除法指令（见表 3-18）

表 3-18　除法指令

指令名称	汇编格式	操　作	机器码	机器周期
除法	DIV　AB	A（商）　$\Leftarrow\frac{A}{B}$ B（余数）	84H	4

这条指令的功能是把累加器 A 中的 8 位无符号整数除以寄存器 B 中的 8 位无符号数，所得的商（为整数）存放在 A 中，余数存放在 B 中，标志位 Cy 和 OV 均被清 0。但是，如果除数 B = 00H 时，执行该指令，A、B 的内容无法确定，且溢出标志 OV 置 1，Cy 仍为 0。执行除法指令时，半进位标志 AC 不受影响，奇偶标志 P 仍按 A 的内容而定。

【例 3-31】 设 A = FFH（255），B = 12H（18），执行指令“DIV AB；A B←A ÷ B”后的结果：

商 A = 0EH（14），余数 B = 03H（3）

标志位 Cy = 0、OV = 0、P = 1

5. 十进制调整指令（见表 3-19）

表 3-19 十进制调整指令

指令名称	汇编格式	操作	机器码	机器周期
二-十进制度调整	DA A	将 A 的内容转换为 BCD 码	D4H	1

这是一条专用于 BCD 码加法的指令。此指令的功能是在累加器 A 进行 BCD 码加法运算后，根据 PSW 中标志位 AC、Cy 的状态以及 A 中的结果，将 A 的内容进行“加 6 修正（通过加 00H、06H、60H、或 66H 到累加器上）”，使其转换成压缩的 BCD 码形式。

【例 3-32】 设 A = 45H（01000101B），表示十进制数 45 的压缩 BCD 码；R5 = 78H（01111000B），表示十进制数 78 的压缩 BCD 码。执行下列指令：

ADD A，R5 ；使 A = BDH（10111101），Cy = 0，AC = 0

DA A ；使 A = 23H（00100011），Cy = 1

结果为，A = 23H，Cy = 1，相当于十进制数 123。

在 80C51 指令系统中这条十进制调整指令不能对减法指令的结果进行修正。

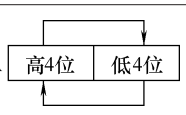
3.3.3 逻辑运算指令

80C51 的逻辑运算指令可分为四大类：对累加器 A 单独进行逻辑操作，对字节变量的逻辑与、逻辑或、逻辑异或操作。指令中的操作数都是 8 位，它们在进行逻辑运算操作时都不影响除奇偶标志外的其他标准位。其中逻辑与、逻辑或、逻辑异或操作指令可以实现对某些字节变量的清零、置 1、取反功能。

1. 对累加器 A 单独进行的逻辑操作

1）清零、取反与半字节交换指令（见表 3-20）

表 3-20 清零、取反与半字节交换指令

指令名称		汇编格式	操作	机器码	机器周期
简单逻辑操作	清零	CLR A	A←0	E4H	1
	取反	CPL A	A←$\overline{A}$	F4H	1
	半字节交换	SWAP A		C4H	1

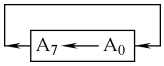
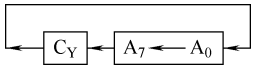
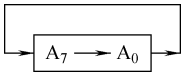
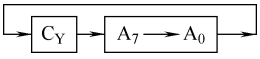
清零指令是将累加器 A 中的所有位全部置 0。

取反指令是将累加器 A 中的内容按位取反，即原来为 1 变为 0，原来为 0 变为 1。

半字节交换指令是将累加器 A 的两个半字节（高 4 位和低 4 位）内容交换。

2）循环移位指令（见表 3-21）

表 3-21　循环移位指令

指令名称			汇编格式	操　作	机器码	机器周期
循环移位	左移	左环移	RL　A		23H	1
		带进位左环移	RLC　A		33H	1
	右移	右环移	RR　A		03H	1
		带进位右环移	RRC　A		13H	1

“RL　A”和“RLC　A”指令都使 A 中的内容逐位左移一位，但 RLC　A 将使 CY 连同 A 的内容一起左移循环，A_7 进入 CY，CY 进入 A_0。

“RR　A”和“RRC　A”指令的功能类似“RL　A”和“RLC　A”，仅是 A 中数据移位的方向向右。

【例 3-33】 若 A = 24H，执行“RL　A”后，A = 48H。

若 A = 24H，CY = 1，执行“RLC　A”后，A = 49H，CY = 1。

若 A = 24H，执行“RR　A”后，A = 12H。

若 A = 24H，CY = 1，执行“RRC　A”后，A = 92H。

2. 逻辑与运算指令（见表 3-22）

表 3-22　逻辑与运算指令

指令名称	汇编格式	操　作	机器码	机器周期
逻辑与	ANL　A，Rn	A←A∧(Rn)	58H ~ 5FH	1
	ANL　A，direct	A←A∧(direct)	55H direct	1
	ANL　A，@Ri	A←A∧((Ri))	56H ~ 57H	1
	ANL　A，#data	A←A∧data	54H data	1
	ANL　direct，A	(direct)←(direct)∧A	52H direct	1
	ANL　direct，#data	(direct)←(direct)∧data	53H direct data	2

这组指令的功能是进行逻辑与运算，前 4 条指令的功能是把源操作数与累加器 A 的内容相与，结果送入目的操作数 A 中；后 2 条指令的功能是把源操作数与直接地址指定的单元内容相与，结果送入直接地址指定的单元。

通过逻辑与运算指令的功能可以实现一个字节里面的某些位清零（与 0 相与）和某些位不变的效果（与 1 相与）。

【例 3-34】 已知寄存器 R5 = 59H，把 R5 内容的低 4 位清零，高 4 位保持不变。

```
MOV  A，R5
ANL  A，#0F0H；高 4 位都与 1 相与达到不变效果，低 4 位都与 0 相与达到清零效果
MOV  R5，A
```

3. 逻辑或运算指令（见表 3-23）

表 3-23 逻辑或运算指令

指令名称	汇编格式	操作	机器码	机器周期
逻辑或	ORL A，Rn	A←A ∨ (Rn)	48H ~ 4FH	1
	ORL A，direct	A←A ∨ (direct)	45H direct	1
	ORL A，@Ri	A←A ∨ ((Ri))	46H ~ 47H	1
	ORL A，#data	A←A ∨ data	44H data	1
	ORL direct，A	(direct)←(direct) ∨ A	42H direct	1
	ORL direct，#data	(direct)←(direct) ∨ data	43H direct data	2

这组指令的功能是进行逻辑或运算，前 4 条指令的功能是把源操作数与累加器 A 的内容相或，结果送入目的操作数 A 中；后 2 条指令的功能是把源操作数与直接地址指定的单元内容相或，结果送入直接地址指定的单元。

通过逻辑或运算指令的功能可以实现一个字节里面的某些位置 1（与 1 相或）和某些位不变的效果与 0 相或）。请读者分析一下如果把【例 3-34】中的与运算指令换成或运算指令会出现什么样的结果？

4. 逻辑异或指令（见表 3-24）

表 3-24 逻辑异或指令

指令名称	汇编格式	操作	机器码	机器周期
逻辑异或	XRL A，Rn	A←A ⊕ (Rn)	68H ~ 6FH	1
	XRL A，direct	A←A ⊕ (direct)	65H direct	1
	XRL A，@Ri	A←A ⊕((Ri))	66H ~ 67H	1
	XRL A，#data	A←A ⊕ data	64H data	1

（续）

指令名称	汇编格式	操　作	机　器　码	机器周期
逻辑异或	XRL　direct，A	(direct)←(direct)⊕A	62H direct	1
	XRL　direct，#data	(direct)←(direct)⊕data	63H direct data	2

这组指令的功能是进行逻辑异或运算，前 4 条指令的功能是把源操作数与累加器 A 的内容相异或，结果送入目的操作数 A 中；后 2 条指令的功能是把源操作数与直接地址指定的单元内容相异或，结果送入直接地址指定的单元。

这些逻辑运算指令，除了带进位位 Cy 循环移位指令只影响 Cy 和 P 标志位外，其余的逻辑运算都不会影响 PSW 的各标志位。

通过逻辑异或运算指令的功能可以实现一个字节里面的某些位取反 1（与 1 相异或）和某些位不变的效果与 0 相异或）。请读者分析一下如果把【例 3-34】中的与运算指令换成异或运算指令会出现什么样的结果？

3.3.4　控制转移指令

1. 无条件转移指令（见表 3-25）

无条件转移指令功能是，当程序执行无条件转移指令时，程序就无条件地转移到该指令所提供的地址去。下面将分别介绍各种无条件转移指令的功能。

表 3-25　无条件转移指令

指令名称	汇编格式	操　作	机　器　码	机器周期
绝对无条件转移	AJMP　addr11	PC←PC+2 $PC_{10\sim0}$←$addr_{10\sim0}$ $PC_{15\sim11}$不变	$a_{10}a_9a_800001$ $a_7\sim a_0$	2
长转移	LJMP　addr16	PC←$addr_{15\sim0}$	00000010 $a_{15}\sim a_8$ $a_7\sim a_0$	2
相对转移	SJMP　rel	PC←PC+2 PC←PC+rel	10000000 rel	2
间接转移	JMP　@A+DPTR	PC←A+DPTR	01110011	2

（1）绝对无条件转移指令 AJMP　addr11

这是两字节指令。指令中包含 addr11 共 11 位地址码，转移的目标地址必需和 AJMP 指令的下一条指令首字节位于程序存储器的同一 2KB 区内。指令执行过程是，先将 PC 值加 2，然后把指令中给出的 11 位地址 addr11（$a_{10}\sim a_0$）送入 PC 的低 11 位（$PC_{10}\sim PC_0$），PC 的高 5 位（$PC_{15}\sim PC_{11}$）保持不变，形成新的目标地址 $PC_{15}\sim PC_{11}a_{10}\sim a_0$，程序随即转移到该地址处。应当注意，即 PC+2 后的 PC 值和目标地址的高 5 位 $a_{15}\sim a_{11}$应该相同。这里，PC 就是指向 AJMP 指令首字节单元的指针。

AJMPaddr11 指令的机器码如下：

$a_{10}a_9a_8$00001	$a_7a_6a_5a_4a_3a_2a_1a_0$
页号　操作码	页内地址

该指令为 2 字节指令，第一字节的低 5 位 00001 是这条指令特有的操作码。指令中给出的 11 位地址 addr11（a_{10} ~ a_0）被分成两部分，分放在指令机器码的两个字节中，第一字节的高 3 位 $a_{10}a_9a_8$，它指出大小为 2KB 的转移范围内的页号（每一 2KB 区分 8 页）；第二字节为 a_7 ~ a_0，它指出页内地址。如表 3-26 所示。

表 3-26　AJMP 指令转移分布地址

操　作　码	a_{10}　a_9　a_8　地址页码	a_7 ~ a_0　（页码地址）
01H	0　0　0　0 页	00H ~ FFH
21H	0　0　1　1 页	00H ~ FFH
41H	0　1　0　2 页	00H ~ FFH
61H	0　1　1　3 页	00H ~ FFH
81H	1　0　0　4 页	00H ~ FFH
A1H	1　0　1　5 页	00H ~ FFH
C1H	1　1　0　6 页	00H ~ FFH
E1H	1　1　1　7 页	00H ~ FFH

绝对转移指令仅为 2 个字节指令，却能提供 2K 范围的转移空间，它比相对转移指令的转移范围大得多。但是要求 AJMP 指令的转移目标地址和 PC + 2 的地址处于同一 2KB 区域内，故受一定的限制。

【例 3-35】 设“AJMP　27BCH”存放在 ROM 的 1FFEH 和 1FFFFH 两地址单元中。在执行该指令时，PC + 2 指向 2000H 单元。转移目标地址 27BCH 和 PC + 2 后指向的单元地址 2000H 在同一 2KB 区，因此能够转移到。页号为 7，指令机器码为 E1H，BCH。

如果把存放在 1FFEH 和 1FFFFH 两单元的绝对转移指令改为 AJMP　1F00H，这种转移是不能实现的，因为转移目标地址 1F00H 和指令执行时 PC + 2 后指向的下一单元地址 2000H 不在同一 2KB 区。

（2）长转移指令 LJMP　addr16

这条指令执行时把指令操作数提供的 16 位目标地址 a_{15} ~ a_0 装入 PC 中，即 PC = a_{15} ~ a_0。所以用长转移指令可以跳到 64KB 程序存储器的任何位置。长转移指令本身是 3 字节指令。

（3）相对转移指令 SJMP　rel

这是一条无条件相对转移指令，转移的目标地址为

目标地址 = 源地址 + 2 + rel

源地址是 SJMP 指令操作码所在的地址。相对偏移量 rel 是一个用补码表示的 8 位带符号数。转移范围为 - 128 ~ + 127 共 256 个单元，即从（PC - 126）~（PC + 129）。

相对转移指令是两字节指令：首字节为操作码，第二字节为相对偏移量。在执行此指令时 PC + 2 指向下一条指令的首字节地址，因此转移目标地址可以在 SJMP 指令的下条指令首

字节前 128 个字节和后 127 个字节之间。

若偏移量 rel 取值为 FEH，则目标地址等于源地址，相当于动态暂停，程序“终止”在这条指令上。动态暂停指令在调试程序时很有用。MCS-51 没有专用的停止指令，若要求动态暂停可以用 SJMP 指令来实现：

```
HERE：SJMP　HERE　；动态停机（80H，FEH）
```

或写为

```
SJMP　$　；“$”表示本指令首字节所在单元的地址，使用它可省略标号
```

这是一条死循环指令，如果系统的中断是开放的，那么指令“SJMP $”实际上是在等待中断。当有中断申请后，CPU 转至执行中断服务程序。中断返回时，仍然返回到这条死循环指令，继续等待中断，而不是返回到该指令的下一条指令。这是因为执行“SJMP $”后，PC 仍指向这条指令，中断的断点就是这条指令的首字节地址。

（4）间接转移指令 JMP　@ A + DPTR

这条指令的功能是把累加器 A 中的 8 位无符号数与数据指针 DPTR 的 16 位数相加，相加之和作为下一条指令的地址送入 PC 中，不改变 A 和 DPTR 的内容，也不影响标志。

间接转移指令采用变址方式实现无条件转移，其特点是转移地址可以在程序运行中加以改变。例如，当把 DPTR 作为基地址且确定时，根据 A 的不同值就可以实现多分支转移，故一条指令可完成多条条件判断转移指令功能。这种功能称为散转功能，所以间接转移指令又称为散转指令。

2. 条件转移指令

条件转移指令是依某种特定条件转移的指令。条件满足时转移（相当于执行一条相对转移指令）；条件不满足时，则按顺序执行下面一条指令。

80C51 的条件转移目标地址位于条件转移指令的下一条指令首字节地址前 128 个字节和后 127 个字节之间，即转移可以向前也可以向后，转移范围为 – 128 ~ + 127，共 256 个单元。在执行条件转移指令时，PC 已指向下一条指令的第一字节地址，如果条件满足，再把相对偏移量 rel 加到 PC 上，计算出转移目标地址。

80C51 的条件转移指令非常丰富，包括累加器判零转移、比较不相等转移和减 1 不为零转移共 3 组。

（1）累加器判零转移指令（见表 3-27）

表 3-27　累加器判零转移指令

指令名称		汇编格式	操　作	机器码	机器周期
判断 A = 0? 转移	零转移	JZ　rel	PC←PC + 2，若 A = 0，则 PC←PC + rel 若 A≠0，顺序执行	60H rel	2
	非零转移	JNZ　rel	PC←PC + 2，若 A≠0，则 PC←PC + rel 若 A = 0，顺序执行	70H rel	2

这两条指令的功能是对累加器 A 的内容为零和不为零进行判断并实现转移。读者分析一下在实际应用当中何时选用为零转移，何时选用不为零转移。

【例 3-36】 已知 A = 01H，分析执行如下指令后的结果。

JZ LOOP1；因为 A≠0，则程序顺序执行

DEC A；A 的内容减 1 后变成 0

JZ LOOP2；因为 A = 0，则程序转移到 LOOP2 处执行

（2）减 1 不为 0 转移指令（见表 3-28）

表 3-28 减 1 不为 0 转移指令

指令名称	汇编格式	操作	机器码	机器周期
减 1 不为 0 转移	DJNZ Rn，rel	PC←PC + 2，Rn←Rn - 1 当 Rn≠0，则 PC←PC + rel 当 Rn = 0，则结束循环，程序往下执行	D8H ~ DFH rel	2
	DJNZ direct，rel	PC←PC + 3，(direct)←(direct) - 1 当 (direct) ≠0，则 PC←PC + rel 当 (direct) = 0，则结束循环，程序往下执行	D5H direct rel	2

减 1 不为 0 转移指令，是把源操作数减 1，结果送回到源操作数中去；如果结果不为 0 则转移。源操作数有寄存器寻址和直接寻址两种方式，允许用户把内部 RAM 单元用作程序循环计数器。

【例 3-37】 由单条 DJNZ 指令来实现软件延时。

LOOP：DJNZ R1，LOOP ； 2 个机器周期

可写成省略标号的形式：

DJNZ R1，$ ；“$”表示本条指令首字节地址

其中，R1 的取值范围为 00H ~ FFH，可实现延时 2 ~ 512 个机器周期（当时钟频率 12MHz 时，一个机器周期为 1μs）。

（3）比较不相等转移指令（见表 3-29）

表 3-29 比较不相等转移指令

指令名称	汇编格式	操作	机器码	机器周期
比较不相等转移	CJNE A，direct，rel	PC←PC + 3，若 (direct) < A，则 PC←PC + rel，且 Cy←0 若 (direct) > A，则 PC←PC + rel，且 Cy←1 若 (direct) = A，则顺序执行，且 Cy←0	B5H direct rel	2
	CJNE A，#data，rel	PC←PC + 3，若 data < A，则 PC←PC + rel，且 Cy←0 若 data > A，则 PC←PC + rel 且 Cy←1 若 data = A，则顺序执行且 Cy←0	B4H data rel	2

（续）

指令名称	汇编格式	操　作	机　器　码	机器周期
比较不相等转移	CJNE Rn，#data，rel	PC←PC + 3，若 data < Rn，则 PC←PC + rel，且 Cy←0 若 data > Rn，则 PC←PC + rel 且 Cy←1 若 data = Rn，则顺序执行，且 Cy←0	B8H ~ BFH data rel	2
	CJNE @ Ri，#data，rel	PC←PC + 3 若 data < ((Ri))，则 PC←PC + rel，且 Cy←0 若 data > ((Ri))，则 PC←PC + rel 且 Cy←1 若 data = ((Ri))，则顺序执行，且 Cy←0	B6H ~ B7H data rel	2

比较不相等转移指令是 80C51 指令系统中具有 3 个操作数的指令组。比较不相等转移指令的功能是比较前两个无符号操作数的大小，若不相等，则转移；否则顺序往下执行。如果第一个操作数大于或等于第二个操作数，则 Cy 清 0，否则 Cy 置 1。指令执行结果不影响其他标志位和所有的操作数。这组指令为 3 字节指令，因此转移目标地址应是 PC 加 3 以后再加偏移量 rel 所得的 PC 的值。即

目标地址 = 源地址 + 3 + rel

源地址是比较转移指令所在位置的首字节地址。

除了比较转移指令外，3 字节的条件转移指令还包括 3 条位判断转移指令和 1 条减 1 不为 0 循环转移指令，其余的都是 2 字节指令。因此在计算偏移量时要特别注意。

【例 3-38】 写出完成下列要求的指令组合。

累加器 A 的内容不等于 55H 时把 A 的内容加上 5；累加器 A 的内容等于 55H 时把 A 的内容减去 5。指令组合如下：

```
CJNE   A，#55H，NEXT1；A 的内容和 55H 比较不相等转移到 NEXT1
CLR    C；顺序执行说明 A 的内容与 55H 相等
SUBB   A，#05H；完成减 5 操作
SJMP   LAST；跳到结束
NEXT1：ADD   A，#05H；若不相等完成加 5 操作
LAST：SJMP  $ ；动态暂停
```

3. 子程序调用与返回指令（见表 3-30）

表 3-30　子程序调用与返回指令

指令名称	汇编格式	操　作	机　器　码	机器周期
绝对调用	ACALL addr11	PC←PC + 2，SP←SP + 1 (SP)←$PC_{7\sim0}$，SP←SP + 1 (SP)←$PC_{15\sim8}$ $PC_{10\sim0}$←$addr_{10\sim0}$ $PC_{15\sim11}$不变	$a_{10}a_9a_8$10001 $addr_{7\sim0}$	2

（续）

指令名称	汇编格式	操　作	机器码	机器周期
长调用	LCALL addr16	PC←PC+3，SP←SP+1 (SP)←$PC_{7\sim0}$，SP←SP+1 (SP)←$PC_{15\sim8}$，PC←$addr_{15\sim0}$	00010010 $addr_{15\sim8}$ $addr_{7\sim0}$	2
子程序返回	RET	$PC_{15\sim8}$←((SP))，SP←SP－1 $PC_{7\sim0}$←((SP))，SP←SP－1	22H	2
中断返回	RETI	$PC_{15\sim8}$←((SP))，SP←SP－1 $PC_{7\sim0}$←((SP))，SP←SP－1	32H	2

绝对调用指令 ACALL 是一条两字节指令。该指令提供了 11 位目标地址 addr11，产生调用地址的方法和绝对转移指令 AJMP 产生转移地址的方法相同，ACALL 是在同一 2KB 区范围内调用子程序的指令。指令执行过程是，执行 ACALL 指令时，PC 加 2 后获得了下条指令的地址，然后把 PC 的当前值压栈（栈指针 SP 加 1，PCL 进栈，SP 再加 1，PCH 进栈）。最后把 PC 的高 5 位和指令给出的 11 位地址 addr11 连接组成 16 位目标地址（$PC_{15}\sim PC_{11}a_{10}\sim a_0$），并作为子程序入口地址送入 PC 中，使 CPU 转向执行子程序。因此，所调用的子程序入口地址必需和 ACALL 指令下一条指令的第一个字节在同一个 2KB 区域的程序存储器空间。

【例 3-39】 当 ACALL 指令所在的首地址为 2000H 时，可调用子程序的入口地址范围是多少？如果 ACALL 指令首地址位于 07FEH 时呢？

ACALL 指令的下一条指令的首地址应该为 2002H，其地址的高 5 位是 00100，因此，可调用子程序的入口地址范围是 2000H～27FFH。

如果 ACALL 指令位于 07FEH 和 07FFH 两地址单元时，执行该指令时 PC 加 2 后的值为 0800H，使其高 5 位是 00001，因此，可调用子程序的入口地址范围是 0800H～0FFFH。

长调用指令 LCALL 是一条可以在 64KB 程序存储器内调用子程序的指令，它是三字节指令。指令执行过程是，把 PC 加 3 获得的下一条指令的地址进栈（先压入低字节，后压入高字节），进栈操作使 SP 加 1 两次，接着把指令的第二和第三字节（$a_{15}\sim a_8$，$a_7\sim a_0$）分别装入 PC 的高位和低位字节中，然后从该地址 addr16（$a_{15}\sim a_0$）开始执行子程序。

两条返回指令的功能都是从堆栈中取出断点地址，送给 PC，并从断点处开始继续执行程序。RET 应放在一般子程序的末尾，而 RETI 应放在中断服务子程序的末尾。在执行 RETI 指令时，还将清除 80C51 中断响应时所置位的优先级状态触发器，开放中断逻辑，使得已申请的较低级中断源可以响应，但在 RETI 指令执行完之后，至少要再执行一条指令才能响应这个中断。

4. 空操作指令（见表 3-31）

表 3-31　空操作指令

指令名称	汇编格式	操　作	机器码	机器周期
空操作	NOP	PC←PC+1	00H	1

空操作指令也是一条控制指令，它控制 CPU 不作任何操作，只是消耗该指令的执行时

间。在执行 NOP 指令时，仅使 PC 加 1，时间上消耗了 12 个时钟周期，不作其他操作。常用于等待、延时等。

3.3.5　位操作指令

另外，80C51 单片机有丰富的位操作指令，这些指令与位操作部件结合在一起，可以把大量的硬件组合逻辑用软件来代替，这样可以方便地应用于各种逻辑控制，这是 80C51 指令系统的一大特色。

80C51 单片机内部有一个性能优异的位处理器，实际上是一个一位的微处理器，它有自己的位变量操作运算器、位累加器（借用进位标志 Cy）和存储器（位寻址区中的各位）等。80C51 指令系统加强了对位变量的处理能力，具有丰富的位操作指令，可以完成以位变量为对象的传送、运算、控制转移等操作。位操作指令的操作对象是内部 RAM 的位寻址区，即字节地址为 20H～2FH 单元中连续的 128 位（位地址为 00H～7FH），以及特殊功能寄存器中可以进行位寻址的各位。

下面说明一下关于位地址 bit 表示方式。在汇编语言级指令格式中，位地址有多种表示方式：

① 直接（位）地址方式，如 20H；7FH 等。

② 字节地址．位方式，如 23H.0，表示字节地址为 23H 单元中的 D_0 位。

③ 寄存器名．位方式，如 ACC.7，但不能写成 A.7。

④ 位定义名方式，如 RS0。

⑤ 用伪指令 BIT 定义位名方式，如 F1、BIT、PSW.1。经定义后，允许在指令中用 F1 来代替 PSW.1。

对于不同的汇编程序，位地址的表示方式不尽相同，可参考有关说明。

下面介绍位变量传送、控制和运算指令。

1. 位变量传送指令（见表 3-32）

表 3-32　位变量传送指令

指令名称	汇编格式	操　作	机器码	机器周期
位变量传送	MOV C，bit	C←(bit)	A2H bit	1
	MOV bit，C	(bit)←C	92H bit	2

这两条指令可以实现位地址单元与位累加器之间的数据传送（注意传送的位数是 1 位）。

2. 位变量修改指令（见表 3-33）

表 3-33　位变量修改指令

指令名称		汇编格式	操　作	机器码	机器周期
位变量修改	位清零	CLR C	C←0	C3H	1
		CLR bit	(bit)←0	C2H bit	1

（续）

指令名称		汇编格式	操作	机器码	机器周期
位变量修改	位求反	CPL C	C←$\overline{C}$	B3H	1
		CPL bit	(bit)←($\overline{bit}$)	B2H bit	1
	位置1	SETB C	C←1	D3H	1
		SETB bit	(bit)←1	D2H Bit	1

这些指令可以完成位地址单元与位累加器的清零、置位、取反操作。

3. 位逻辑运算指令（见表3-34）

表3-34 位逻辑运算指令

指令名称		汇编格式	操作	机器码	机器周期
位逻辑运算	逻辑与	ANL C，bit	C←C∧(bit)	82H bit	2
		ANL C，/bit	C←C∧($\overline{bit}$)	B0H bit	2
	逻辑或	ORL C，bit	C←C∨(bit)	72H bit	2
		ORL C，/bit	C←C∨($\overline{bit}$)	A0H bit	2

这些指令可以实现位地址单元内容或者内容取反后的值与位累加器的内容相与或者相或运算，操作结果送位累加器C中。

4. 位条件转移指令（见表3-35）

表3-35 位条件转移指令

指令名称		汇编格式	操作	机器码	机器周期
位条件转移	Cy=1转移	JC rel	PC←PC+2，若Cy=1，则 PC←PC+rel 若Cy=0，顺序执行	40H rel	2
	Cy=0转移	JNC rel	PC←PC+2，若Cy=0，则 PC←PC+rel 若Cy=1，顺序执行	50H rel	2
	(bit)=1转移	JB bit，rel	PC←PC+3，若(bit)=1，则 PC←PC+rel 若(bit)=0，顺序执行	20H bit rel	2
	(bit)=0转移	JNB bit，rel	PC←PC+3，若(bit)=0，则 PC←PC+rel 若(bit)=1，顺序执行	30H bit rel	2
	(bit)=1位清零转移	JBC bit，rel	PC←PC+3，若(bit)=1，则 (bit)←0，PC←PC+rel 若(bit)=0，顺序执行	10H bit rel	2

【例 3-40】　设 x、y、z 为位地址，用指令实现下列逻辑表达式。

$$(x)\oplus(y)=(\bar{x})(y)+(x)(\bar{y})\Rightarrow(z)$$

可由以下 8 条指令来实现：

```
MOV   C, y
ANL   C, /x   ; C←(x̄) ∧ (y)
MOV   z, C    ; 暂存
MOV   C, x
ANL   C, /y   ; C←(x̄) ∧ (ȳ)
ORL   C, z    ; C←(x̄)⊕(y)
MOV   z, C    ; 存结果
```

本 章 小 结

单片机常用的编程语言是汇编语言，它具有占存储空间少、运行速度快、实时性强等优点，但是缺乏通用性。高级语言面向过程，具有直观、易学、通用性强的优点，但是单片机直接能够识别和执行的是机器语言，需要经过汇编的过程变成机器语言。

80C51 共有 111 条指令，其指令执行时间短、字节少、位操作指令非常丰富。按指令长度分类，可分为 1 字节、2 字节和 3 字节指令。按指令执行时间分类，可分为 1 个机周期、2 个机周期和 4 个机周期指令。

1. 80C51 指令基本格式由标号、操作码、操作数和注释组成。其中标号和注释为选择项，可在可无；操作数可以为 0 ~ 3 个；操作码为必须项，代表了指令的操作功能。

2. 80C51 指令系统有七种寻址方式：立即寻址、直接寻址、寄存器寻址、寄存器间接寻址、变址寻址、相对寻址和位寻址。所谓寻址就是寻找操作数的地址。

3. 80C51 的指令系统按指令功能分类，可分为数据传送类（29 条）、算术运算类（24 条）、逻辑运算及移位类（24 条）、控制转移类（17 条）和位操作类（17 条）五大类指令。下面分类总结一下：

（1）数据传送指令

1）内部 RAM 和特殊功能寄存器 A、Rn、@ Ri、direct 之间可用 MOV 指令互相传送数据。注意，工作寄存器 Rn 之间不能直接传送。

2）读写外部 RAM 要用 MOVX 指令间址传送。

3）读 ROM 要用 MOVC 指令。

4）堆栈操作：入栈 PUSH；出栈 POP。

5）字节交换可在 A 与 Rn、@ Ri、direct 之间进行；低半字节交换只能在 A 与@ Ri 之间进行；高低 4 位互换只能在 A 中进行。

（2）算术运算指令

1）加减法必须由 A 与另一加减数之间进行；运算结果存在 A 中，有进（借）位时，Cy1；无进（借）位时，Cy 清 0，另有带 Cy 的加法指令，减法指令必须带 Cy。

2）加 1 减 1 可在 A、Rn、@ Ri、direct 中执行，另有 DPTR 加 1 指令。注意，加 1 减 1 指令不影响 Cy。

3）乘除法必须在 A 与 B 之间进行，积低位和商存在 A 中，积高位和余数存在 B 中。

4）BCD 码调整指令用于 BCD 码加法，须紧跟在加法指令之后。

（3）逻辑运算及移位指令

1）逻辑运算有“与”、“或”和“异或”运算指令，逐位进行，目的寄存器可以是 A 或 direct。

2）循环移位必须在 A 中进行，分为带或不带 Cy、左移或右移指令。

3）字节（8 位）清零和取反必须在 A 中进行。

（4）控制转移类指令

1）无条件转移指令可分为长转移、短转移、相对转移和间接转移四种。长转移 LJMP 转移范围是 64KB；短转移 AJMP 转移范围是 2KB；相对转移 SJMP 转移范围是 －128 ～ ＋127。使用 AJMP 和 SJMP 指令应该注意转移目标地址是否在转移范围内，若超出范围，程序将出错。

间接转移也称散转指令，属变址寻址，以 DPTR 为基址，由 A 的值来决定具体的转移地址。

2）条件转移指令可分为判 C 转移、判 bit 转移、判 A 转移、减 1 非 0 转移和比较转移指令。满足条件，则转移；不满足条件，则程序顺序执行。

3）调用指令根据其调用子程序范围分为长调用和短调用两种，其特点类似长转移和短转移指令。长调用可调用 64KB 范围内的子程序；短调用只能调用与当前 PC 值同一 2KB 范围内的子程序。

4）返回指令对应于调用指令，分为子程序返回和中断返回两种，两者不能混淆。其功能都是从堆栈中取出断点地址，送入 PC，使程序从主程序断点处继续执行。

5）空操作指令的功能仅使 PC 加 1，常用于在延时或等待程序中时间“微调”。

（5）位操作类指令

1）位传送只能在 Cy 与 bit 之间进行，bit 与 bit 之间不能直接传送。

2）位修正分置 1、清 0 和取反，只能由 Cy 或 bit 进行。

3）位逻辑运算只有“与”、“或”两种指令，无位“异或”指令。

80C51 单片机的指令系统见表 3-36 所示。

表 3-36　80C51 单片机指令表

数据传送指令表					
序号	助　记　符	操 作 功 能	字　节　数	机器周期数	备　　注
1	MOV A，Rn	寄存器内容送累加器	1	1	
2	MOV A，direct	直接寻址字节内容送累加器	2	1	
3	MOV A，@ Ri	片内 RAM 内容送累加器	1	1	
4	MOV A，#data	立即数送累加器	2	1	
5	MOV Rn，A	累加器内容送寄存器	1	1	
6	MOV Rn，direct	直接寻址字节内容送寄存器	2	2	
7	MOV Rn，#data	立即数送寄存器	2	1	

（续）

数据传送指令表

序号	助　记　符	操 作 功 能	字　节　数	机器周期数	备　　注
8	MOV direct，A	累加器送直接寻址字节	2	1	
9	MOV direct，Rn	寄存器内容送直接寻址字节	2	2	
10	MOV direct，@ Ri	片内 RAM 内容送直接寻址字节	2	2	
11	MOV direct1，direct2	直接寻址字节内容送另一直接寻址字节	3	2	
12	MOV direct，#data	立即数送直接寻址字节	3	2	
13	MOV @ Ri，A	累加器内容送片内 RAM	1	1	
14	MOV @ Ri，direct	直接寻址字节内容送片内 RAM	2	2	
15	MOV @ Ri，#data	立即数送片内 RAM	2	1	
16	MOV DPTR，#data	16 位立即数送数据指针寄存器	3	2	
17	MOVX A，@ Ri	片外 RAM 内容送累加器（8 位地址）	1	2	
18	MOVX A，@ DPTR	片外 RAM 内容送累加器（16 位地址）	1	2	
19	MOVX @ Ri，A	累加器内容送片外 RAM（8 位地址）	1	2	
20	MOVX @ DPTR，A	累加器内容送片外 RAM（16 位地址）	1	2	
21	MOVC A，@ A + DPTR	相对数据指针内容送累加器	1	2	
22	MOVC A，@ A + PC	相对程序计数器内容送累加器	1	2	
23	XCH A，Rn	累加器与寄存器交换内容	1	1	
24	XCH A，@ Ri	累加器与片内 RAM 交换内容	1	1	
25	XCH A，direct	累加器与直接寻址字节交换内容	2	1	
26	XCHD A，@ Ri	累加器与片内 RAM 交换低半字节内容	1	1	
27	SWAP A	累加器交换高半字节与低半字节内容	1	1	
28	PUSH direct	直接寻址字节内容压入堆栈栈顶	2	2	
29	POP direct	堆栈栈顶内容弹出到直接寻址字节	2	2	

算术操作类指令汇总表

1	ADD A，Rn	寄存器与累加器内容相加	1	1	
2	ADD A，@ Ri	片内 RAM 与累加器内容相加	1	1	
3	ADD A，direct	直接寻址字节与累加器内容相加	2	1	
4	ADD A，#data	立即数与累加器内容相加	2	1	
5	ADDC A，Rn	寄存器与累加器与进位位内容相加	1	1	
6	ADDC A，@ Ri	片内 RAM 与累加器与进位位内容相加	1	1	
7	ADDC A，direct	直接寻址字节与累加器与进位位内容相加	2	1	
8	ADDC A，#data	立即数与累加器与进位位内容相加	2	1	

（续）

算术操作类指令汇总表					
序号	助 记 符	操 作 功 能	字 节 数	机器周期数	备 注
9	SUBB A，Rn	累加器内容减寄存器与进位位内容	1	1	
10	SUBB A，@ Ri	累加器内容减片内 RAM 与进位位内容	1	1	
11	SUBB A，direct	累加器内容减直接寻址字节与进位位内容	2	1	
12	SUBB A，#data	累加器内容减立即数与进位位内容	2	1	
13	INC A	累加器内容加 1	1	1	
14	INC Rn	寄存器内容加 1	1	1	
15	INC @ Ri	片内 RAM 内容加 1	1	1	
16	INC direct	直接寻址字节内容加 1	2	1	
17	INC DPTR	数据指针寄存器内容加 1	1	2	
18	DEC A	累加器内容减 1	1	1	
19	DEC Rn	寄存器内容减 1	1	1	
20	DEC @ Ri	片内 RAM 内容减 1	1	1	
21	DEC direct	直接寻址字节内容减 1	2	1	
22	DA A	累加器内容十进制调整	1	1	
23	MUL AB	累加器内容乘寄存器 B 内容	1	4	
24	DIV AB	累加器内容除寄存器 B 内容	1	4	
逻辑运算类指令汇总表					
1	ANL A，Rn	寄存器内容与累加器内容	1	1	
2	ANL A，@ Ri	片内 RAM 内容与累加器内容	1	1	
3	ANL A，direct	直接寻址字节内容与累加器内容	2	1	
4	ANL A，#data	立即数与累加器内容	2	1	
5	ANL direct，A	累加器内容与直接寻址字节内容	2	1	
6	ANL direct，#data	立即数与直接寻址字节内容	3	2	
7	ORL A，Rn	寄存器内容或累加器内容	1	1	
8	ORL A，@ Ri	片内 RAM 内容或累加器内容	1	1	
9	ORL A，direct	直接寻址字节内容或累加器内容	2	1	
10	ORL A，#data	立即数或累加器内容	2	1	
11	ORL direct，A	累加器内容或直接寻址字节内容	2	1	
12	ORL direct，#data	立即数或直接寻址字节内容	3	2	
13	XRL A，Rn	寄存器内容异或累加器内容	1	1	
14	XRL A，direct	片内 RAM 内容异或累加器内容	1	1	
15	XRL A，@ Ri	直接寻址字节内容异或累加器内容	2	1	
16	XRL A，#data	立即数异或累加器内容	2	1	

（续）

逻辑运算类指令汇总表

序号	助　记　符	操 作 功 能	字　节　数	机器周期数	备　　注
17	XRL direct，A	累加器内容异或直接寻址字节内容	2	1	
18	XRL direct，#data	立即数异或直接寻址字节内容	3	2	
19	CPL A	累加器内容取反	1	1	
20	CLR A	累加器内容清零	1	1	
21	RL A	累加器内容向左环移一位	1	1	
22	RR A	累加器内容向右环移一位	1	1	
23	RLC A	累加器内容带进位位向左环移一位	1	1	
24	RRC A	累加器内容带进位位向右环移一位	1	1	

转移类指令汇总表

1	AJMP addr11	绝对转移（2KB 地址内）	2	2	
2	LJMP addr16	长转移（64KB 地址内）	3	2	
3	SJMP rel	相对短转移（－128 ~ ＋127B 地址内）	2	2	
4	JMP @ A＋DPTR	间接转移	1	2	
5	JZ rel	累加器内容为零转移	2	2	
6	JNZ rel	累加器内容不为零转移	2	2	
7	CJNE A，direct，rel	累加器内容与直接寻址字节内容不等转移	3	2	
8	CJNE A，#data，rel	累加器内容与立即数不等转移	3	2	
9	CJNE Rn，#data，rel	寄存器内容与立即数不等转移	3	2	
10	CJNE @ Ri，#data，rel	片内 RAM 内容与立即数不等转移	3	2	
11	DJNZ Rn，rel	寄存器内容减 1 不为零转移	3	2	
12	DJNZ direct，rel	直接寻址字节内容减 1 不为零转移	3	2	
13	ACALL addr11	绝对调子程序（2KB 地址内）	2	2	
14	LCALL addr16	长调子程序（64KB 地址内）	3	2	
15	RET	返回主程序	1	2	
16	RETI	中断返回主程序	1	2	
17	NOP	空操作	1	1	

位操作类指令汇总表

1	MOV C，bit	直接寻址位内容送进位位	2	1	
2	MOV bit，C	进位位内容送直接寻址位	3	1	
3	CPL C	进位位取反	1	1	
4	CPL bit	直接寻址位取反	2	1	
5	CLR C	进位位清零	1	1	
6	CLR bit	直接寻址位清零	2	1	
7	SETB C	进位位置位	1	1	

（续）

位操作类指令汇总表

序号	助 记 符	操 作 功 能	字 节 数	机器周期数	备 注
8	SETB bit	直接寻址位置位	2	1	
9	ANL C，bit	直接寻址位内容与进位位内容	2	2	
10	ANL C，/bit	直接寻址位内容的反与进位位内容	2	2	
11	ORL C，bit	直接寻址位内容或进位位内容	2	2	
12	ORL C，/bit	直接寻址为内容的反或进位位内容	2	2	
13	JC rel	进位位为 1 转移	2	2	
14	JNC rel	进位位不为 1 转移	2	2	
15	JB bit，rel	直接寻址位为 1 转移	3	2	
16	JNB bit，rel	直接寻址位不为 1 转移	3	2	
17	JBC bit，rel	直接寻址位为 1 转移且该位清 0	3	2	

习 题

1. 80C51 单片机有哪几种寻址方式？各寻址方式所对应的寄存器或存储器空间如何？

2. 说明下列指令中源操作数和目的操作数的寻址方式。

（1）ADD A，30H　　（2）MOV 30H，20H

（3）MOV A，@R0　　（4）MOVX @R1，A

（5）SJMP $　　（6）MOV R0，#20H

（7）ORL C，00H　　（8）MOV DPTR，#2000H

（9）MOVC A，@A+PC　　（10）ANL 20H，#30H

（11）ANL C，/30H　　（12）CPL C

（13）CPL A　　（14）CPL 20H

（15）ADD A，@R1　　（16）MOVC A，@A+DPTR

（17）DJNZ R0，rel　　（18）SETB 00H

（19）CJNE A，#00H，rel　　（20）INC DPTR

3. 访问特殊功能寄存器 SFR 可以采用哪些寻址方式？

4. 访问内部 RAM 单元可以采用哪些寻址方式？

5. 访问外部 RAM 单元可以采用哪些寻址方式？

6. 访问外部程序存储器可以采用哪些寻址方式？

7. 完成某种操作可以采用几条指令构成的指令序列实现，试写出完成以下每种操作的指令序列。

（1）将 R0 的内容传送到 R1。

（2）内部 RAM 单元 60H 的内容传送到寄存器 R2。

（3）外部 RAM 单元 1000H 的内容传送到内部 RAM 单元 60H。

（4）外部 RAM 单元 1000H 的内容传送到寄存器 R2。

（5）外部 RAM 单元 1000H 的内容传送到外部 RAM 单元 2000H。

8. 若（R1）=30H，（A）=40H，（30H）=60H，（40H）=08H，试分析执行下列程序段后上述各单元内容的变化。

MOV A，@R1

MOV　@R1，40H

MOV　40H，A

MOV　R1，#7FH

9. 若（A）=E8H，（R0）=40H，（R1）=20H，（R4）=3AH，（40H）=2CH，（20H）=0FH，试写出下列各指令独立执行后有关寄存器和存储单元的内容？若该指令影响标志位，试指出 Cy、AC、和 OV 的值。

（1）MOV　A，@R0

（2）ANL　40H，#0FH

（3）ADD　A，R4

（4）SWAP　A

（5）DEC　@R1

（6）XCHD　A，@R1

10. 分步写出下列程序每条指令的运行结果。

（1）MOV　SP，#40H

MOV　A，#20H

MOV　B，#30H

PUSH　A

PUSH　B

POP　A

POP　B

（2）MOV　A，#83H

MOV　R0，#47H

MOV　47H，#34H

ANL　A，#47H

ORL　47H，A

XRL　A，@R0

（3）MOV　R0，#00H

MOV　A，#20H

MOV　B，#0FFH

MOV　20H，#0F0H

XCH　A，R0

XCH　A，B

XCHD　A，@R0

（4）MOV　A#45H

MOV　R5，#78H

ADD　A，R5

DA　A

11. 若（50H）=40H，试写出执行以下程序段后累加器 A、寄存器 R0 及内部 RAM 的 40H、41H、42H 单元中的内容各为多少？

MOV　A，50H

MOV　R0，A

MOV　A，#00H

MOV　@R0，A

MOV　A，#3BH

MOV 41H，A

MOV 42H，41H

12. 试编写程序，将内部 RAM 的 20H、21H、22H 三个连续单元的内容依次存入 2FH、2EH 和 2DH 单元。

13. 试编写程序，完成两个 16 位数的减法

7F4DH－2B4EH

结果存入内部 RAM 的 30H 和 31H 单元，30H 单元存差的高 8 位，31H 单元存差的低 8 位。

14. 试编写程序，将内部 RAM 的 20H、21H 单元的两个无符号数相乘，结果存放在 R2、R3 中，R2 中存放高 8 位，R3 中存放低 8 位。

15. 若（CY）=1，（P1）=10100011B，（P3）=01101100B。试指出执行下列程序段后，CY、P1 口及 P3 口内容的变化情况。

MOV P1.3，C

MOV P1.4，C

MOV C，P1.6

MOV P3.6，C

MOV C，P1.0

MOV P3.4，C

第 4 章　汇编语言程序设计

【学习目的】

1. 了解汇编语言程序设计的特点。
2. 掌握汇编语言程序的基本结构及简单应用的设计方法。
3. 用循环程序设计一个控制信号灯的程序。
4. 用循环程序设计一个控制步进电动机的程序。
5. 用分支程序设计一个控制汽车信号灯的程序。
6. 用分支程序设计一个控制水塔水位的程序。

一个完整的单片机应用系统是合理的硬件与完善的软件的有机组合，二者缺一不可。所谓的软件设计就是指程序的设计。程序实际上是一系列计算机指令的有序集合。我们把利用计算机的指令系统来合理地编写出解决某个问题的程序的过程称之为程序设计。

4.1　程序设计概述

汇编语言是一种面向机器的程序设计语言，常因机器的不同而有差别。本章以 80C51 单片机为例来介绍汇编语言的构成。

4.1.1　程序设计语言

1. 机器语言（Machine Language）

这是一种用二进制代码“0”和“1”表示指令和数据的程序设计语言。计算机只能识别二进制代码，这种语言是能被计算机直接识别和执行的机器级语言。

特点：机器语言能够被计算机立即识别并加以执行，具有执行速度快、占用内存少等优点。但对于使用者来说，用机器语言编写程序具有编写难、识别难、记忆难、查错难、交流难等缺点。

2. 汇编语言（Assembly Language）

汇编语言是一种用助记符来表示的面向机器的程序设计语言。不同的机器所使用的汇编语言一般是不同的。但计算机的 CPU 不能直接识别汇编语言，所以计算机不能立即执行汇编语言程序。用汇编语言编写的源程序，在由计算机执行之前，必须将它翻译成机器语言程序。

特点：这种语言弥补了机器语言的不足，用汇编语言编写程序比用机器语言方便、直观、易懂、易用、易记，可以编写出结构紧凑、运行时间精确的程序，所以这种语言非常适合于实时控制的需要。

3. 高级语言（High-Level Language）

高级语言是面向过程并能独立于计算机硬件结构的通用程序设计语言，是一种接近人类语言和数学表达式的计算机语言，如 BASIC、FORTRAN、COBOL、PASCAL、C 语言等。高

级语言不能被计算机直接识别和执行，需要用编译程序或解释程序将高级语言编写的源程序翻译为机器语言。

特点：它比汇编语言易学、易懂，具有通用性强、易于移植等优点。高级语言的语句功能强，它的一条语句往往相当于许多条指令，因而用于翻译的程序要占用较多的存储空间，而且执行时间长，且不易精确掌握，故在高速实时控制中一般是不适用的。

4.1.2 汇编语言源程序的编辑与汇编

在目前单片机的开发应用中，经常采用 C 语言和汇编语言共同编写程序。要想很好地掌握和应用单片机首先要掌握汇编语言。

汇编语言是面向机器的程序设计语言，对于 CPU 不同的单片机，其汇编语言一般是不同的。用汇编语言编写的程序称为汇编语言源程序。

汇编语言源程序是由汇编语言语句构成的。汇编语言语句可分为两大类：指令性语句和指示性语句。

指令性语句是由指令组成的由 CPU 执行的语句。

指示性语句是由伪指令组成的，它不被 CPU 执行，用来告诉汇编程序如何对程序进行汇编的指令；由于它不能生成机器语言，故又被称为伪指令语句。

1. 指令性语句格式

[标号:] 操作码助记符　[目的操作数] [,源操作数] [;注释]

① 每条汇编语句一般由若干部分组成，每一部分称为一个字段。

② 每个字段之间应该严格地用分界符加以分隔。

③ 分界符包括冒号、空格符、逗号、分号等。标号段与操作码之间要加冒号“:”；操作码与操作数之间要用空格相隔；各操作数之间要用逗号“,”相隔；操作数与注释段之间要加分号“;”相隔。

2. 伪指令的指示性语句格式

[标号:] 伪操作　操作数 [,操作数,……] [;注释]

① 伪指令不是真正的指令，是在汇编时供汇编程序识别的指令，又称为汇编指令。

② 它不属于指令系统，也无对应的机器码，只是用来对汇编过程进行某种控制。利用伪指令告诉汇编程序如何进行汇编，为编程提供方便。

3. 常用的指示性语句

① ORG（Origin）汇编起始指令

ORG 是程序汇编起始地址定位伪指令。

功能：规定对汇编语言源程序进行汇编时，目的程序在程序存储器中存放的起始地址。

格式：[标号:] ORG　16 位地址或标号

注意：在一个源程序中，可多次使用 ORG 指令，以规定不同程序段的起始位置，地址应从小到大顺序排列，不允许重叠。

例如：ORG　1000H

MOV　A，#12H　　；该指令的机器码是 74H、12H

ADD　A，#34H　　；该指令的机器码是 24H、34H

在上述源程序中，第一条指令的首字节 74H 存放到程序存储器的 1000H 地址单元中，

其他字节和后续指令的数据顺序存放到后面的存储单元中。

② END（End）汇编结束指令

END 是汇编语言程序结束伪指令。

功能：表示程序已结束，汇编程序对 END 后面的指令不再汇编。

格式：[标号:] END

注意：在一个源程序中，只能有一条 END 指令，而且必须放在整个程序的末尾。

③ EQU（Equate）赋值指令。

EQU 是赋值（也称等值）伪指令。

功能：把操作数段中的数据或地址赋值给标号字段中的字符名称。

格式：字符名称　EQU　　数值或汇编符号

注意：字符名称必须先赋值后使用，故 EQU 指令通常放在源程序的开头。EQU 可定义 8 位或 16 位的数据或地址，

```
例如：  ABC     EQU    30H     ; AB 与 30H 等值
        ACB     EQU    R3      ; AC 与 R3 等值
        MOV   A, ABC           ; 把片内 RAM 30H 单元中的数据送入 A 中
        MOV   A, ACB           ; 把 R3 中的数据送入累加器 A 中
```

④ DATA（Data）数据地址赋值指令

DATA 是数据地址赋值伪指令。

功能：把操作数段中的表达式的值赋给标号字段中的字符名称。

格式：字符名称　DATA　　表达式

注意：DATA 指令功能与 EQU 指令类似，它们的主要区别如下：

① DATA 定义的字符名称可以先使用后定义，DATA 指令可以放在源程序的任何位置。

② DATA 只能用来定义 8 位的数据或地址。

③ EQU 可以把汇编符号赋给字符名称，而 DATA 只能把数据赋给字符名称。

④ DATA 可以把表达式的值赋给字符名称，这个表达式是可以进行求值运算的。

⑤ XDATA 数据地址赋值指令。

XDATA 是数据地址赋值伪指令。

功能：把操作数段中的表达式的值赋给标号字段中的字符名称。

格式：字符名称　XDATA　　表达式

注意：XDATA 指令功能与 DATA 指令类似，它们的主要区别是 XDATA 可定义 16 位的数据或地址。

⑥ BIT（Bit）位地址赋值指令

BIT 是位地址赋值伪指令。

功能：把位地址赋给字符名称。

格式：字符名称　BIT　位地址

```
例如：AB    BIT     30H     ; AB 与 30H 等值
      AC    BIT     P1.0    ; AC 与 P1.0 等值
      MOV     C,AB          ; 把位地址区 30H 单元中的数据送入位累加器 C 中
      CLR     AC            ; 把 P1.0 中的内容清零
```

⑦ DB（Define Byte）定义字节指令

DB 是定义字节伪指令。

功能：从程序存储器指定地址单元开始存放若干个字节的数值或 ASCII 码字符。

格式：[标号:] DB　字节数据或 ASCII 码字符

注意：多个字节数据或 ASCII 码字符之间要用逗号相隔，DB 指令常用于定义 8 位的数据常数表。

例如：ORG　1000H

TAB：DB　　50H，60，'A'

　　DB　　01010111B，'6'

⑧ DW（Define Word）定义字指令

DW 是定义字伪指令。

功能：从程序存储器指定地址单元开始存放若干个字的数值。

格式：[标号:] DW　字节数据或 ASCII 码字符

注意：多个字数据之间要用逗号相隔，DW 指令常用于定义 16 位的地址表。

例如：ORG　1000H

TAB：　DW　20H，50H，00H，60H

⑨ DS（Define Space）定义存储空间指令

DS 是定义存储空间伪指令。

功能：从程序存储器指定地址单元开始，保留表达式的值所规定的存储单元。

格式：[标号:] DS　表达式

例如：ORG　1000H

TAB：DS　06H

　　DB　25H，35H

在上述源程序中，程序存储器从 1000H 单元开始保留 6 个单元，1006H 单元存放 25H，1007H 单元存放 35H。

4. 汇编语言源程序的汇编

汇编语言源程序必须要转换为机器码（即目的程序），计算机才能执行，这个转换过程称为汇编。

汇编语言源程序的汇编可分为手工汇编和机器汇编两类。

① 手工汇编是指用人脑通过查指令编码表把汇编语言源程序翻译成机器码的过程。

② 机器汇编是用机器代替人脑并由专门的程序来进行的，这种程序称为汇编程序（不同的指令系统汇编程序不同）。机器汇编由计算机自动完成，汇编程序把用汇编语言编写的源程序翻译成由机器语言表示的目的程序。

③ 反汇编是在分析程序存储器已有的程序时，将机器语言翻译成汇编语言的转换过程。

源程序、汇编程序和目的程序之间的关系如图 4-1 所示。

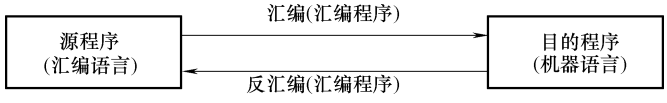


图 4-1　源程序、汇编程序和目的程序之间的关系

5. 汇编语言源程序的编辑

汇编语言源程序一般在微机上借助编辑软件进行编写，可供使用的编辑工具有许多，如行编辑软件、屏幕编辑软件等。

4.1.3　汇编语言程序的基本结构

汇编语言程序具有 4 种结构形式：顺序程序结构、循环程序结构、分支程序结构和子程序结构。

1. 顺序程序

顺序程序结构是一种最简单、最基本的程序结构，又称为简单程序结构或直线程序结构。程序按顺序一条一条地执行指令，程序流向不变，如图 4-2 所示。

2. 循环程序

循环程序是把需要多次重复执行的某段程序，利用条件转移指令反复转向执行，可减小整个程序的长度，优化程序结构，如图 4-3 所示。

循环程序一般由循环初始化、循环处理、循环控制和循环结束四部分组成。

3. 分支程序

分支程序根据条件进行判断来决定程序的执行，如满足条件则进行程序转移，如不满足条件就顺序执行程序。判断是通过条件转移指令实现的。分支程序又分为单分支结构和多分支结构，如图 4-4 所示。

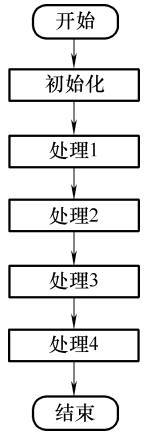


图 4-2　顺序程序结构

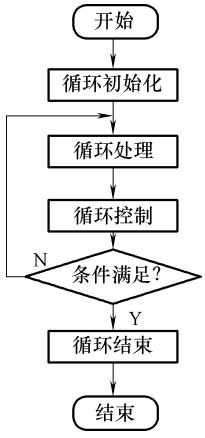


图 4-3　循环程序结构

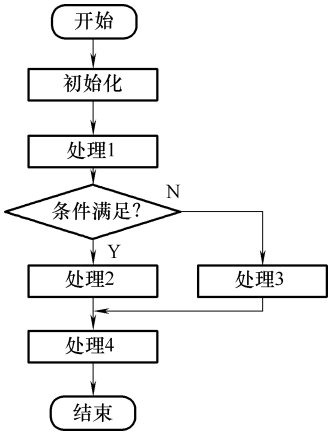


图 4-4　分支程序结构

4. 子程序

子程序是指完成某一确定任务并能被其他程序反复调用的程序段。使用子程序可以减小整个程序的长度，实现模块化程序结构。

4.1.4　程序设计方法和技巧

1. 程序设计的一般步骤

① 分析工作任务，明确要达到的工作目的、技术指标等。

② 确定解决问题的算法。算法就是如何将实际问题转化成程序模块来处理，要对不同

的算法进行分析、比较，找出最适宜的算法。

③ 画程序流程图。其图形的符号规定均与高级语言流程图相同，如桶形框表示程序的开始或结束，矩形框表示需要进行的工作，菱形框表示需要判断的事情，指向线表示程序的流向等。

④ 分配内存工作单元，确定程序与数据的存放地址。

⑤ 编写源程序。

⑥ 上机调试、修改源程序。

2. 程序设计的一般原则

① 按照尽可能使程序简短和缩短运行时间两个原则来编写程序。

② 应用程序一般都由一个主程序（包括若干个功能模块）和多个子程序构成，即采用模块化的程序设计方法。

③ 每一功能模块或子程序都能完成一个明确的任务，实现某个具体功能，如检测输入信号、码制转换、输出控制信号、发送数据、接收数据、延时、显示、打印等。

3. 模块化程序设计方法的特点

① 单个模块结构的程序功能单一，易于编写、调试和修改。

② 对程序的局部修改，可以使无关的部分保持不变。

③ 程序可读性好，便于功能扩展和版本升级。

④ 对于使用频繁的子程序可以建立子程序库，便于多个模块调用。

⑤ 可实现多人同时进行程序的编写和调试工作，缩短程序编写时间。

4. 划分模块应遵循的原则

① 高内聚性。每个模块应具有独立的功能，能产生一个明确的结果。

② 低耦合性。模块之间的控制耦合应尽量简单，数据耦合应尽量少。控制耦合是指模块进入和退出的条件及方式，数据耦合是指模块间的信息交换（传递）方式、交换量的多少及交换的频繁程度。

③ 模块长度适中。模块语句的长度在 20 ~ 100 条的范围较合适。模块太长时，分析和调试比较困难，失去了模块化程序结构的优越性；过短则模块的连接太复杂，信息交换太频繁。

5. 程序设计的一般技巧

① 尽量采用循环结构和子程序结构。这样可以使程序的总容量大大减少、提高程序的效率、节省内存。

② 尽量少用无条件转移指令。这样可以使程序条理更加清楚，从而减少错误。

③ 对于通用的子程序，除了用于存放子程序入口参数的寄存器外，子程序中用到的其他寄存器的内容应压入堆栈，即保护现场。一般不必把标志寄存器压入堆栈。

④ 在中断处理程序中，除了要保护中断处理程序中用到的寄存器外，还要保护标志寄存器。

⑤ 用累加器传递入口参数或返回参数比较方便，在子程序中一般不必把累加器内容压入堆栈。

4.2　顺序程序设计

顺序程序设计方法：

顺序结构程序是最简单、最基本的程序。要设计出高质量的程序需要掌握一定的技巧，需要熟悉指令系统，正确地选择指令，掌握程序设计的基本方法和技巧，以达到提高程序执行效率、减少程序长度、最大限度地优化程序的目的。

顺序程序的特点和设计方法。

① 结构比较单一和简单，按程序编写的顺序依次执行，中间没有任何分支，程序流向不变。

② 数据传送指令使用得较多，没有控制转移类指令。

③ 作为复杂程序的某个组成部分，如循环结构程序中需多次重复执行的某段程序（称为循环处理）。

【例 4-1】 有两个 6 位 BCD 码分别存放在片内 RAM 的 30H、31H、32H 单元和 40H、41H、42H 单元，求它们的和并将和存放到片内 RAM 的 50H、51H、52H 单元。

解： 设定片内 RAM 30H 单元存放高位，片内 RAM 32H 单元存放低位，其他单元与之类同。BCD 码加法运算后，要用 DA 指令进行调整。

```
地址      机器码          程序                    注释
                          ORG   0000H
0000H     02 00 30        LJMP  MAIN
                          ORG   0030H
0030H     E5 32     MAIN: MOV   A，32H          ；低 2 位被加数送入累加器 A
0032H     25 42           ADD   A，42H
0034H     D4              DA    A               ；BCD 码调整
0035H     F5 52           MOV   52H，A          ；存放和的低 2 位
0037H     E5 31           MOV   A，31H          ；中 2 位被加数送入累加器 A
0039H     35 41           ADDC  A，41H          ；加上低位的进位
003BH     D4              DA    A               ；BCD 码调整
003CH     F5 51           MOV   51H，A          ；存放和的中 2 位
003EH     E5 30           MOV   A，30H          ；高 2 位被加数送入累加器 A
0040H     35 40           ADDC  A，40H          ；加上中位的进位
0042H     D4              DA    A
0043H     F5 50           MOV   50H，A
0045H     80 FE           SJMP  $
                          END
```

【例 4-2】 有一个 16 位二进制负数的补码存放在片内 RAM 的 30H、31H 单元，求它的原码的绝对值并将它存放到片内 RAM 的 40H、41H 单元。

解：设定片内 RAM 30H 单元存放高位，片内 RAM 31H 单元存放低位，其他单元与之类同。补码取反后要加 1，绝对值要去掉符号位。

```
地址      机器码          程序                    注释
                          ORG   0000H
0000H   02 00 30          LJMP   MAIN
                          ORG   0030H
0030H   E5 31       MAIN: MOV   A，31H          ；低 8 位补码送入累加器 A
0032H   F4                CPL   A               ；低 8 位取反
0033H   24 01             ADD 1，#01H           ；补码取反后要加 1
0035H   F5 41             MOV   41H，A          ；存放原码绝对值的低 8 位
0037H   E5 30             MOV   A，30H          ；高 8 位补码送入累加器 A
0039H   F4                CPL A                 ；高 8 位取反
003AH   34 00             ADDC A，#00H          ；加上低 8 位的进位
003CH   54 7F             ANL A#7FH             ；去掉最高位符号位
003EH   F5 40             MOV 40H，A            ；存放原码绝对值的高 7 位
0040H   80 FE             SJMP  $               ；暂停
                          END
```

4.3 循环程序设计

4.3.1 循环程序设计方法

1. 循环程序的结构

循环程序的结构一般包括以下几部分：

循环初始化——进入循环处理前必须要有的一个环节，用于完成循环前的准备工作。循环初始化包括给工作寄存器（或其他存储单元）设置计数初值、地址指针、数据块长度等。

循环处理——需要多次重复执行的程序段。循环处理是循环程序的核心，用于完成主要的计算和操作任务。

循环控制——用条件转移指令控制循环是否继续。每循环一次，根据循环结束条件进行一次判断；当满足条件时，停止循环，继续执行其他程序；否则，再作循环。

循环结束——用于存放循环程序的执行结果，同时恢复相关工作单元的初值。

2. 循环程序的编写方法

循环程序一般有以下两种编写方法：

① 先循环处理后循环控制（即先处理后判断），其流程图如图 4-5 所示。

② 先循环控制后循环处理（即先判断后处理），其流程图如图 4-6 所示。

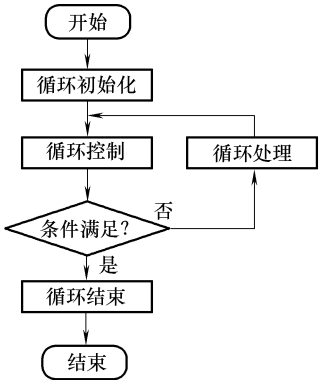


图 4-5　先循环处理后循环控制流程图

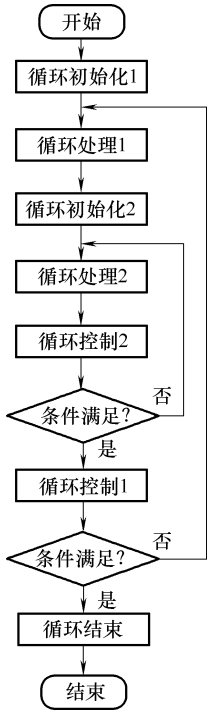


图 4-6　先循环控制后循环处理流程图

循环处理和循环控制构成循环体，若循环程序的循环体内不再包含其他循环程序，则称为单重循环程序；若循环程序的循环体内包含有其他循环程序，则称为多重循环程序，又称为循环嵌套。多重循环程序中的各重循环不能有交叉，不能从外循环跳入内循环，只能外循环内嵌套内循环。两重循环程序流程如图 4-6 所示。

3. 循环程序的特点和设计方法

程序结构紧凑，占用存储单元较少，程序中间有分支，循环程序本质上是分支程序的一种特殊形式。

DJNZ 指令使用得较多，凡是分支程序中可以使用的控制转移类指令，循环程序一般都可以使用。

循环控制的形式有多种。计数循环是最常用的一种，它先预置计数初值，再用 DJNZ 指令控制循环次数；条件循环也是较常用的一种，它先预置结束循环的条件，再用 CJNE 指令、JZ 指令或 JB 指令控制循环的结束。

4.3.2　循环程序设计实例

【例 4-3】 片内 RAM 中存放有 10 个数据，首地址为 30H，编程将数据块传送到片外 RAM 以 1000H 为首地址的存储单元。

解：该程序是单重循环程序，片内 RAM 首地址 30H、片外 RAM 首地址 1000H 和数据块长度 10 都是循环初始化的内容。

循环控制是对数据块长度进行判断，每传送一个数据，存放数据块长度的寄存器减 1；10 个数据传送完，存放数据块长度的寄存器内容正好为零，退出循环。

```
地址      机器码          程序                          注释
                          ORG   0000H
0000H   02 01 00          LJMP   MAIN
                          ORG   0100H
0100H   79 30       MAIN:MOV   R1，30H          ；置片内 RAM 地址指针 30H
0102H   90 10 00          MOV   DPTR，#1000H    ；置片外 RAM 地址指针 1000H
0105H   7A 0A             MOV   R2，#10H        ；数据块的长度
0107H   E7          LOOP:MOV A，@R1           ；从片内 RAM 取数据
0108H   F0                MOVX   @DPTR，A       ；数据传送到片外 RAM
0109H   09                INC R1                ；修改片内 RAM 地址指针
010AH   A3                INC DPTR              ；修改片外 RAM 地址指针
010BH   DA FA             DJNZ R2，LOOP         ；循环次数未到 10 次，转移
010DH   80 FE             SJMP  $               ；暂停
                          END
```

【例 4-4】 P1 口为输出口，控制 8 盏灯（P1 口输出低电平时灯被点亮）。编写程序，使灯按以下规律显示：同一时间只有两盏灯点亮，用 P1.7、P1.6 口控制的灯开始，每盏灯闪烁 5 次，再移向下两盏灯，同样闪烁 5 次，循环往复，延时时间 1s。晶振频率为 6MHz。

解：主程序是双重循环程序，循环移位是外循环，灯闪烁 5 次是内循环，内循环程序不能与外循环程序交叉。

延时 1s 采用三重循环程序。晶振频率为 6MHz 时，机器周期为 2μs，延时程序的延时时间计算方法如下：

$$\{1+[1+(1+(1+1+2)\times125+2)\times200+2]\times5+2\}\times2\mu s$$
$$=1006036\mu s=1.006036s$$

主程序：

```
地址      机器码          程序                          注释
                          ORG   0000H
0000H   02 00 30          LJMP   MAIN
                          ORG   0030H
0030H   74 5F       MAIN:MOV   A，#5FH          ；灯点亮初始状态
0032H   79 05         LP1:MOV   R1，#5           ；循环闪烁次数
0034H   F5 90         LP2:MOV   P1，A            ；两盏灯点亮
0036H   12 01 00          LCALL   DELAY         ；延时 1s
0039H   75 90 FF          MOV   P1，#0FFH       ；灭
003CH   12 01 00          LCALL   DELAY         ；延时 1s
003FH   D9 F3             DJNZ   R1，LP2        ；循环闪烁次数不够 5 次，继续
```

```
0041H   03          RR   A          ; 右移一位
0042H   03          RR   A          ; 再右移一位
0043H   80 ED       SJMP LP1
```

延时子程序：

地址	机器码	程序			注释
			ORG	0100H	
0100H	7B 05	DELAY:	MOV	R3，#5	；延时 1s 的循环次数
0102H	7C C8	DEL3:	MOV	R4，#200	；延时 200ms 的循环次数
0104H	7D 7D	DEL2:	MOV	R5，#125	；延时 1ms 的循环次数
0106H	00	DEL1:	NOP		
0107H	00		NOP		
0108H	DD FE		DJNZ	R5，DEL1	
010AH	DC F9		DJNZ	R4，DEL2	
010CH	DB F5		DJNZ	R3，DEL3	
010EH	22		RET		；子程序返回
			END		

【例 4-5】 P1 口为输出口，控制步进电动机的四相绕组。编写程序，控制步进电动机每 2s 正向转动一步。晶振频率为 6MHz。

解： 步距角（°）$\theta_b = 360/mZ$

电动机转速（r/min）$n = 60f/mZ$

式中　f——脉冲频率（Hz 或步/s）；

m——拍数，本例中为 4；

Z——转子齿数，本例中为 5。

拍数 $m=4$，若使用的步进电动机转子齿数 Z 为 5，则步距角 $\theta_b = 18°$。题目要求步进电动机每 2s 正向转动一步，即 $T=2$s，则 $f=0.5$Hz，电动机转速 $n=1.5$r/min。

用三重循环设计 2s 的循环程序。晶振频率为 6MHz 时，机器周期为 2μs，延时程序的延时时间计算方法如下：

$$\{1 + [1 + (1 + 1 + 2 \times 123 + 2) \times 200 + 2] \times 20 + 2\} \times 2\mu s$$
$$= 2000126\mu s = 2.000126s$$

P1.0 ~ P1.3 口为输出口，分别控制步进电动机的四相绕组，步进电动机的控制状态与 P1 口的控制码的对应关系见表 4-1。

表 4-1　步进电动机的控制状态与 P1 口控制码的关系

控制状态	P1 口控制码	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0
						D 相	C 相	B 相	A 相
A 相、B 相绕组通电	03H	0	0	0	0	0	0	1	1
B 相、C 相绕组通电	06H	0	0	0	0	0	1	1	0
C 相、D 相绕组通电	0CH	0	0	0	0	1	1	0	0
D 相、A 相绕组通电	09H	0	0	0	0	1	0	0	1

主程序：

地址	机器码	程序	注释
		ORG 0000H	
0000H	02 00 30	LJMP MAIN	
		ORG 0030H	
0030H	75 90 03	MAIN: MOV P1, #03H	; AB 相通电
0033H	12 01 00	LCALL DELAY	
0036H	75 90 06	MOV P1, #06H	; BC 相通电
0039H	12 01 00	LCALL DELAY	
003CH	75 90 0C	MOV P1, #0CH	; CD 相通电
003FH	12 01 00	LCALL DELAY	
0042H	75 90 09	MOV P1, #09H	; DA 相通电
0045H	12 01 00	LCALL DELAY	
0048H	80 E6	SJMP MAIN	; 重复循环

延时子程序：

地址	机器码	程序	注释
		ORG 0100H	
0100H	7B 14	DELAY: MOV R3, #20	; 延时 2s 的循环次数
0102H	7F C8	LP1: MOV R7, #200	; 延时 100ms 的循环次数
0104H	7E 7B	LP2: MOV R6, #123	; 延时 0.5ms 的循环次数
0106H	00	NOP	
0107H	DE FE	LP3: DJNZ R6, LP3	
0109H	DF F9	DJNZ R7, LP2	
010BH	DB F5	DJNZ R3, LP1	
010DH	22	RET	
		END	; 程序结束

【例 4-6】 片内 RAM 从 50H 单元开始存放了 10 个无符号数，编程将它们按由小到大的顺序排列。

解： 数据排序的方法有很多，本例采用常用的冒泡排序法，又称为两两比较法，程序流程图如图 4-7 所示。

想象把 10 个数纵向排列，自上而下将存储单元相邻的两个数进行比较，若前数大于后数，则存储单元中的两个数互换位置；若前数小于后数，则存储单元中的两个数保持原来位置。按同样的原则依次比较后面的数据，直到该组数据全部比较完，经过第 1 轮的比较，最大的数据就像冒泡一样排在了存储单元最末的位置上。经过 9 轮冒泡，便可完成 10 个数据

的排序。

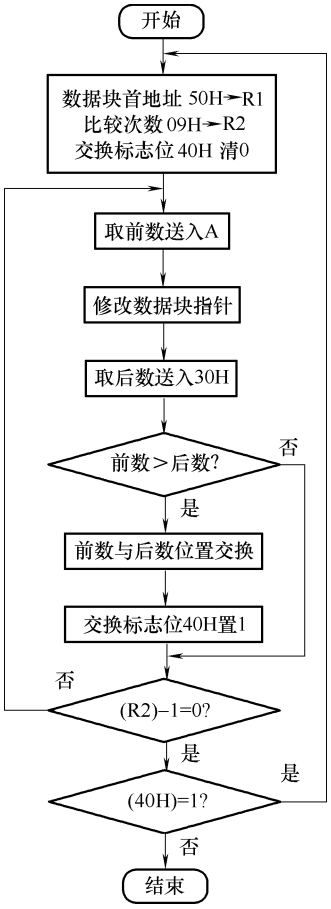


图 4-7　冒泡排序程序流程图

在实际排序中，10 个数不一定要经过 9 轮排序冒泡，可能只要几次就可以了。为了减少不必要的冒泡次数，可以设计一个交换标志，每一轮冒泡的开始将交换标志位清 0，在该轮数据比较中若有数据位置互换，则将交换标志位置 1；每轮冒泡结束时，若交换标志位仍为 0，则表明数据排序已完成，可以提前结束排序。

```
                        ORG   0000H
0000H   02 01 00        LJMP   MAIN
                        ORG   0100H
0100H   79 50     MAIN: MOV   R1，#50H        ；置数据块首地址
0102H   7A 09           MOV   R2，#09H        ；置每次冒泡比较次数
0104H   C2 40           CLR   40H             ；交换标志位清 0
0106H   E7        LOOP1: MOV   A，@R1         ；取前数
```

```
0107H   09              INC   R1
0108H   87 30           MOV   30H, @R1        ; 取后数
010AH   B5 30 00        CJNE A, 30H, LOOP2    ; 比较前数与后数的大小
010DH   40 07    LOOP2: JC    LOOP3           ; 若前数<后数则转移，不互换
010FH   F7              MOV   @R1, A          ; 大数存放到后数的位置
0110H   19              DEC   R1
0111H   A7 30           MOV   @R1, 30H        ; 小数存放到前数的位置
0113H   09              INC   R1              ; 恢复数据指针，准备下一次比较
0114H   D2 40           SETB  40H             ; 有互换，标志位置1
0116H   DA EE    LOOP3: DJNZ  R2, LOOP1       ; 若一次冒泡未完，继续进行比较
0118H   20 40 E5        JB    40H, MAIN       ; 若有交换，继续进行下一轮冒泡
011BH   80 FE           SJMP  $
                        END
```

4.4 分支程序设计

4.4.1 分支程序设计方法

1）分支程序是根据程序的要求改变程序的执行顺序，并根据条件对程序的流向进行判断的程序结构。

2）分支程序一般有两个或两个以上的出口。

3）分支程序又分为单分支和多分支结构。

4）分支程序的特点和设计方法。

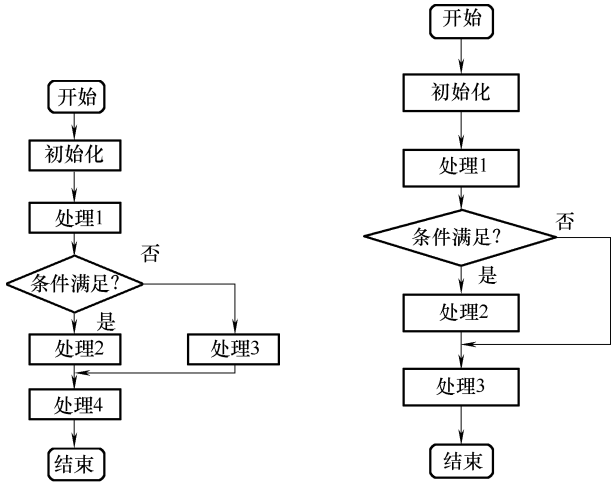
① 程序中有转移指令包括无条件转移、条件转移和散转指令。

② 分支的出口有两个以上时，形成散转程序，一般用散转指令来实现，设计方法有4种：转移指令表法、地址偏移量表法、转向地址表法和利用RET指令法。

③ 单分支程序一般有一个入口、两个出口，一般用无条件转移和条件转移指令来实现，结构形式有如下两种：

一种是当条件满足时，执行处理程序2，否则执行处理程序3。分支程序流程图如图4-8a所示。

另一种是当条件满足时，跳过处理程序2，直接执行处理程序3，否则顺序执行处理程序2和处理程序3。分支程序流程图如图4-8b所示。



a) 分支程序形式一　　b) 分支程序形式二

图 4-8　分支程序流程图

④ 分支程序允许嵌套，即一个分支接一个分支，形成树状多分支结构。多分支程序流程图如图 4-9 所示。

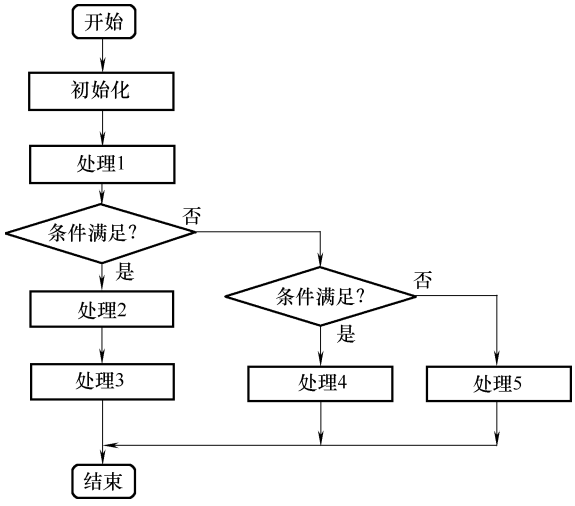


图 4-9　多分支程序流程图

4.4.2　分支程序设计实例

【例 4-7】　设计一个水塔水位控制系统，晶振频率为 6MHz。设计要求如下：

① 在水塔内三个不同的高度分别安装了一根固定不动的金属棒，正常情况下，塔内水位应保持在虚线之内，水位控制原理如图 4-10 所示。

② A 棒处于水位上限，B 棒处于水位下限。当水位低于水位下限时，自动起动水泵电动机给水塔供水；直到塔内水位达到水位上限，自动停止水泵电动机。

③ 塔内水位从水位上限下降到水位下限的过程中，水泵电动机不会自动起动。

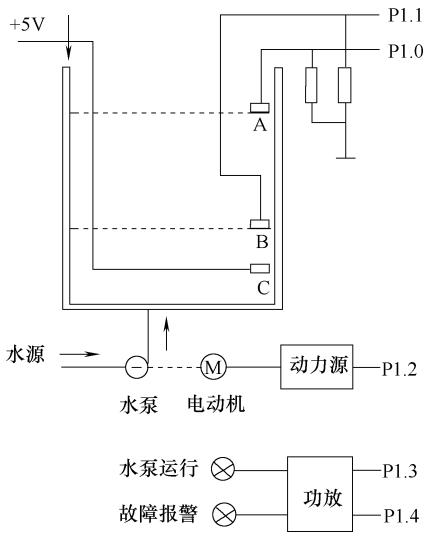


图 4-10 水位控制原理

④ 水塔进水时，要有信号灯指示；水位检测发生故障时，要有故障灯指示并使水塔水位控制系统停止工作。

解：当塔内水位处于水位下限以下时，A、B 棒通过电阻接地。

由于水的导电作用，当塔内水位达到水位下限时，B 棒接通 +5V；当塔内水位达到水位上限时，A 棒也接通 +5V。

水位上限信号输入至 P1.0 口，水位下限信号输入至 P1.1 口，P1.2 口输出控制信号以控制水泵电机的起动（P1.2 = 0）和停止（P1.2 = 1），P1.3 口输出显示信号以指示水泵电动机的运行状态（P1.3 = 0 时点亮），P1.4 口输出故障信号以指示水位检测系统故障状态（P1.4 = 0 时点亮）。

水位控制信号与水泵电动机控制状态的对应关系见表 4-2 所示。

表 4-2 水位控制信号与水泵电动机控制状态的对应关系

P1.1	P1.0	控 制 状 态	P1.2	P1.3	P1.4
0	0	水泵电动机起动	0	0	1
0	1	故障报警	1	1	0
1	0	维持原来状态			1
1	1	水泵电动机停止	1	1	1

为了防止电动机频繁起停，在起动或停止电动机后最少要维持这一状态 20s，这可以采用延时程序来实现。

主程序：

```
                    ORG   0000H
0000H   02 00 30    LJMP  MAIN
```

```
                        ORG   0030H
0030H   43 90 03   MAIN:ORL   P1, #03H        ; 水位信号输入端做读入准备
0033H   F5 90              MOV   A, P1         ; 读入水位检测信号
0035H   30 E1 08           JNB   ACC.1, QDZB   ; P1.1=0，转移至起动准备
0038H   20 E0 16           JB    ACC.0, TZDJ   ; P1.0=1，转移至停止电动机
003BH   12 01 00     YS:LCALL   DELAY          ; 延时 20s
003EH   80 F0              SJMP  MAIN
0040H   30 E0 08   QDZB:JNB   ACC.0, QDDJ      ; P1.0=0，转移至起动电动机
0043H   D2 92              SETB  P1.2          ; 停止电动机
0045H   D2 93              SETB  P1.3          ; 关闭电动机运行指示
0047H   C2 94              CLR   P1.4          ; 打开水位检测故障指示
0049H   80 FE              SJMP  $
004BH   C2 92      QDDJ:CLR   P1.2             ; 起动电动机
004DH   C2 93              CLR   P1.3          ; 打开电动机运行指示
004FH   80 EA              SJMP  YS
0051H   D2 92      TZDJ:SETB  P1.2             ; 停止电动机
0053H   D2 93              SETB  P1.3
0055H   80 E4              SJMP  YS
```

延时子程序：

```
地址      机器码          程序                    注释
                        ORG   0100H
0100H   7B C8    DELAY:MOV   R3, #200    ; 延时 20s 的循环次数
0102H   7F C8      LP1:MOV   R7, #200    ; 延时 100ms 的循环次数
0104H   7E 7B      LP2:MOV   R6, #123    ; 延时 0.5ms 的循环次数
0106H   00                 NOP
0107H   DE FE      LP3:DJNZ  R6, LP3
0109H   DF F9              DJNZ  R7, LP2
010BH   DB F5              DJNZ  R3, LP1
010DH   22                 RET
                           END             ; 程序结束
```

4.5 子程序设计

4.5.1 子程序设计方法

1）子程序是指完成某一专门任务并能被其他程序反复调用的程序段。调用子程序的程序称为主程序或调用程序。使用子程序的过程称为调用子程序。子程序执行完后返回主程序的过程称为子程序返回。

2）主程序和子程序是相对的，同一程序既可以作为另一程序的子程序，也可以有自己的子程序。也就是说，子程序是允许嵌套的，嵌套深度和堆栈区的大小有关。

3）采用子程序能使整个程序结构简单，缩短程序设计时间，减少对存储空间的占用。

4）子程序的特点和设计方法：

① 子程序具有通用性和独立性，以满足所有调用程序实现资源共享。

② 子程序的第一条指令的地址称为子程序的入口地址，该指令前应有标号。

③ 合理地确定子程序的参数传递方式。入口参数是子程序需要的原始参数，由主程序通过相关的工作寄存器、特殊功能寄存器、片内 RAM 或堆栈等传送给子程序；出口参数是根据入口参数执行子程序后获得的结果，由子程序通过相关的工作寄存器、特殊功能寄存器、片内 RAM 或堆栈等传递给主程序。

④ 在主程序中可以用调用指令调用子程序，在子程序末尾用 RET 返回指令从子程序返回主程序。

⑤ 根据需要保护现场和恢复现场。在子程序的开始，使用压栈指令把需要保护的内容压入堆栈；在返回主程序前，使用弹出指令把堆栈中保护的内容送回原来的存储单元中。

⑥ 子程序中有可能要使用累加器 A 或工作寄存器，在子程序使用它们之前，把它们中可能存有的主程序的中间结果保存起来，这一过程称为保护现场。在子程序执行完并将返回主程序之前，再将这些中间结果取出，送回到累加器 A 或原来的工作寄存器中，这一过程称为恢复现场。

⑦ 子程序中应尽量使用相对转移指令而不使用其他转移指令，以便子程序放在内存的任何区域都能被主程序调用。

⑧ 要正确地设置堆栈指针，以避免堆栈区与工作寄存器或其他存储单元发生冲突。

5）传送子程序参数的方法。

① 利用寄存器或片内 RAM 传送参数。可以把入口参数存放到寄存器或片内 RAM 中传送给子程序，也可以把出口参数存放到寄存器或片内 RAM 中传送给主程序。

② 利用寄存器传送参数的地址。把存放入口参数的地址通过寄存器传送给子程序，子程序根据寄存器中存放入口参数的地址便可找到入口参数并对它们进行相应操作；出口参数的地址也可通过寄存器传送给主程序。

③ 利用堆栈传送参数。可以用压栈指令 PUSH 把入口参数压入堆栈传送给子程序，也可以使用压栈指令 PUSH 把出口参数压入堆栈传送给主程序。

4.5.2　子程序设计实例

【例 4-8】 将片内 RAM 的 20H ~ 24H 单元中的 1 位十六进制数转换成 ASCII 码，并分别存放到片内 RAM 的 30H ~ 34H 单元中。

解：ASCII 码是有一定规律的编码，如十六进制数的 0 ~ 9 的 ASCII 码为该数值加上 30H，分别为 30H ~ 39H；十六进制数的 A ~ F 的 ASCII 码为该数值加上 37H，分别为41H ~ 46H。

主程序：

```
地址      机器码          程序                    注释
                        ORG   0000H
0000H   02 01 00        LJMP   MAIN
                        ORG   0100H
0100H   7C 05     MAIN:MOV   R4, #05H      ; 数据块的长度
0102H   78 20          MOV   R0, #20H      ; 存放十六进制数首地址
0104H   79 30          MOV   R1, #30H      ; 存放 ASCII 码首地址
0106H   E6          LP1:MOV   A, @R0       ; 取十六进制数
0107H   12 01 50       LCALL   HAC         ; 调用代码转换程序
010AH   F7             MOV   @R1, A        ; 存放 ASCII 码
010BH   08             INC   R0
010CH   09             INC   R1
010DH   DC F7          DJNZ   R4, LP1
010FH   80 FE          SJMP   $
```

子程序：

```
地址      机器码          程序                    注释
                      ORG   0150H       ; 子程序从地址 0150H 开始存放
0150H   C0 D0    HAC:PUSH   PSW         ; 保护现场
0152H   54 0F          ANL   A, #0FH     ; 屏蔽掉高 4 位
0154H   C0 E0          PUSH   ACC        ;
0156H   C3             CLR   C
0157H   94 0A          SUBB   A, #0AH    ; 比较 A 中内容的大小
0159H   D0 E0          POP   ACC
015BH   40 02          JC   LP2          ; (A) <10 时，转移
015DH   24 07          ADD   A, #07H
015FH   24 30     LP2:ADD   A, #30H
```

```
0161H   D0 D0          POP   PSW        ；恢复现场
0163H   22             RET               ；子程序返回
                       END
```

4.6 查表程序设计

4.6.1 查表程序设计方法

1）在单片机的实际应用中，经常要对一些数据进行函数运算，如求二次方、正弦函数等，为了提高单片机执行程序的速度，一般将某函数的全部函数值按一定的规律编成表格存放到程序存储器中。

2）查表程序就是根据某数据的函数运算要求，按索引号从程序存储器中查找与之相对应的函数值的程序结构。

3）设计查表程序时，主要通过两条查表指令实现查表功能。

4）查表程序的特点和设计方法：

① 查表指令“MOVC　A，@A＋DPTR”的查表过程比较简单。查表时首先需要把数据表格起始地址存入 DPTR，然后把所查数据的索引值送入累加器 A 中，最后使用查表指令“MOVC A，@A＋DPTR”完成查表。

② 查表指令“MOVC　A，@A＋PC”的查表过程相对复杂一些。查表时首先使用传送指令把所查数据的索引值送入累加器 A，然后用“ADD　A，#data”指令对累加器 A 进行修正。data 值由该式确定，PC＋data＝数据表格的首地址。其中，PC 是“MOVC　A，@A＋PC”的下一条指令的地址。因此，data 值实际等于查表指令和数据表格之间的字节数。最后使用查表指令“MOVC A，@A＋PC”完成查表。

4.6.2 查表程序设计实例

【例 4-9】 变量 a、b 均为小于 10 的正整数，编程计算 $c=a^2+b^2$，其中变量 a、b 分别存放在片内 RAM 的 51H 和 52H 单元中，计算结果 c 存放到片内 RAM 的 53H 单元。

解：首先把二次方表格起始地址 0150H 存入 DPTR，然后把要查数据 a 或 b 的索引值送入累加器 A 中，最后使用查表指令“MOVC　A，@A＋DPTR”完成查表。

主程序：

```
                       ORG   0000H
0000H   02 01 00       LJMP   MAIN
                       ORG   0100H
0100H   90 01 50   MAIN:MOV   DPTR，#0150H   ；平方表格首地址
0103H   E5 51          MOV   A，51H          ；取数 a 送到 A 中作为索引值
0105H   93             MOVC   A，@A＋DPTR ；查数 a 的二次方
```

```
0106H    F8                  MOV   R0, A              ; 数 a 的二次方送到 R0 中暂存
0107H    E5 52               MOV   A, 52H             ; 取数 b 送到 A 中作索引值
0109H    93                  MOVC  A, @A + DPTR       ; 查数 b 的二次方
010AH    28                  ADD   A, R0              ; 求二次方和
010BH    F5 53               MOV   53H, A
010DH    80 FE               SJMP  $
                             ORG   0150H
0150H    00 01 04 09  TABLE: DB    0, 1, 4, 9         ; 0 ~ 9 二次方表
0154H    10 19 24            DB    16, 25, 36
0157H    31 40 51            DB    49, 64, 81
                             END
```

本 章 小 结

1. 程序设计语言按语言结构可分为三大类：机器语言、汇编语言和高级语言。在目前单片机的开发应用中，经常采用 C 语言和汇编语言共同编写程序。

2. 汇编语言是面向机器的程序设计语言，对于 CPU 不同的单片机，其汇编语言一般是不同的。在进行汇编语言源程序设计时，必须严格遵循汇编语言的格式和语法规则。

3. 汇编语言源程序是由汇编语言语句构成的。汇编语言语句可分为两大类：指令性语句和指示性语句。指令性语句一般由标号、操作码、操作数和注释四个字段组成，指示性语句也包括标号、操作码、操作数和注释四个字段。

4. 汇编语言源程序的汇编可分为手工汇编和机器汇编两类。机器汇编可以使用汇编程序进行汇编或反汇编。

5. 汇编语言程序具有顺序结构、循环结构、分支结构和子程序结构四种结构形式。实际的应用程序一般都由一个主程序和多个子程序构成，即采用模块化的程序设计方法。

6. 程序设计的原则是尽可能使程序简短和缩短运行时间，设计的关键首先是根据实际问题和所选用的单片机的特点来合理地确定解决问题的算法，然后是将工作任务细分成易于理解和实现的小模块。

7. 在程序设计时，要注意顺序程序、循环程序、分支程序、查表程序和子程序的特点和设计方法。要设计出高质量的程序还需要掌握一定的技巧，通过多读、多看一些实用程序可以积累一定的设计经验。

习　　题

1. 程序设计语言有哪几类？各有什么特点？
2. 汇编语言有哪两类语句？各有什么特点？
3. 汇编语言源程序有哪两类汇编？各采用什么方法来实现？
4. 汇编语言程序设计一般分哪几个步骤？

5. 有两个 4 位十六进制数分别存放在片内 RAM 的 20H、21H 单元和 30H、31H 单元内，请编程求它们的和，并将和存放到片内 RAM 的 40H、41H 单元。

6. 有一个 16 位二进制负数的原码存放在片内 RAM 的 60H、61H 单元内，请编程求它的补码，并将它存放到片内 RAM 的 70H、71H 单元。

7. 片内 RAM 中存放有 20 个数据，首地址为 40H，请编程将数据块传送到片外 RAM 以 50H 为首地址的存储单元中。

8. 片外 RAM 中存放有 20 个数据，首地址为 40H，请编程将数据块传送到片外 RAM 以 5000H 为首地址的存储单元中，同时将片外 RAM 以 40H 为首地址的 20 个存储单元中的内容全部清零。

9. 片内 RAM 的 30H ~ 34H 单元中存放有 5 个十六进制数，请编程计算这 5 个数的算术平均值，结果存放到片内 RAM 的 35H 单元。

10. 请分别编写延时 1min、1h 的子程序，晶振频率为 12MHz。

11. 自变量 X 为一无符号数，存放在片内 RAM 的 30H 单元，函数 Y 存放在 31H 单元。请编写满足如下关系的程序：$X \leqslant 95$ 时，$Y=1$；$95 < X < 105$ 时，$Y=2$；$X \geqslant 105$ 时，$Y=3$。

12. 在片内 RAM 从 30H 单元开始存放了 50 个数据，请编程找出某一关键值并将该值在片内 RAM 存储单元的地址存放到片内 RAM 的 70H 单元。

第 5 章　80C51 的中断系统及定时器/计数器

【学习目的】

1. 熟悉中断的基本概念。
2. 掌握 80C51 中断系统及定时/计数器的结构。
3. 能够编写中断、定时/计数器的程序。

中断是 CPU 与 I/O 设备之间数据传送的一种控制方式。80C51 单片机具备一套完善的中断系统，包含 5 个中断源、2 个优先级。

为了满足定时和计数的功能，80C51 单片机配置了 2 个 16 位的定时器/计数器，这对于单片机应用系统具有非常重要的使用价值。

5.1　80C51 单片机的中断系统

中断是指计算机在执行某一程序的过程中，由于计算机系统内、外的某种原因，而必须中止原程序的执行，转去执行相应的处理程序，待处理结束之后，再回来继续执行被中止的原程序的过程。

采用了中断技术后的计算机，可以解决 CPU 与外设之间速度匹配的问题，使计算机可以及时处理系统中许多随机的参数和信息，同时，它也提高了计算机处理故障与应变的能力。

5.1.1　80C51 单片机中断的概念

（1）中断

为了说明中断的概念，先看一个日常生活中可能经历的中断过程：你在看书→电话铃响了→你在书上做个记号，走到电话旁→拿起电话和对方通话→门铃响了→你让打电话的对方稍等一下→你去开门，并在门旁与来访者交谈→谈话结束，关好门，回到电话机旁，拿起电话，继续通话→通话完毕，挂上电话→从做记号的地方起继续读书。

从看书到接电话，是一次中断过程，而从打电话到与门外来访者交谈，则是在中断过程中发生的又一次中断，即所谓的中断嵌套。在日常生活中为什么会发生上述这一系列中断现象呢？是因为你在某一时刻要面对着 3 项任务：看书、打电话和接待来访者。但一个人不可能同时完成 3 项任务，因此，只好采用中断的方法，穿插着去做。

也就是说，当中央处理器 CPU 正在处理某件事情的时候，外界发生了紧急事件请求，要求 CPU 暂停当前的工作，转而去处理这个紧急事件。处理完毕后，再回到原来被暂停的地方，继续原来的工作，这样的过程称为中断。

（2）中断嵌套

即指在中断过程中又发生了新中断的现象。

（3）中断源

向 CPU 发出中断请求的来源。

（4）中断请求或中断申请

中断源要求 CPU 为其服务的请求。

（5）中断查询

指 CPU 通过测试各中断控制寄存器中各标志位的状态，以确定有没有中断请求发生以及是哪一个中断源提出中断请求的过程。

（6）中断响应

指 CPU 对中断源提出的中断请求的接受，发生在中断查询之后。

（7）中断处理或中断服务

中断处理就是执行中断服务程序。

（8）中断返回

中断返回是指 CPU 在执行完中断服务程序后，返回原来暂停的地方（断点），继续执行原来程序的过程。

5.1.2 80C51 单片机的中断源及中断向量

在计算机系统中，中断可以由各种硬件设备产生，以便请求服务或报告故障等。此外，中断也可由处理器自身产生，如程序错误或对操作系统的请求作出响应等。计算机的中断服务需求是以中断请求（Interrupt Request）的形式提出来的，不管是来自硬件的还是来自软件的中断请求，凡是中断请求的来源都统称为中断源。80C51 的中断系统具有 5 个中断源，即 2 个外部中断、2 个定时器中断和 1 个串行中断。

1）INT0，外部中断 0。可由 IT0（TCON.0）选择其为低电平有效还是下降沿有效。当 CPU 检测到 P3.2 引脚上出现有效的中断信号时，中断标志 IE0（TCON.1）置 1，向 CPU 申请中断。

2）INT1，外部中断 1。可由 IT1（TCON.2）选择其为低电平有效还是下降沿有效。当 CPU 检测到 P3.3 引脚上出现有效的中断信号时，中断标志 IE1（TCON.3）置 1，向 CPU 申请中断。

3）TF0（TCON.5），片内定时/计数器 T0 溢出中断请求标志。当定时/计数器 T0 发生溢出时，置位 TF0，并向 CPU 申请中断。

4）TF1（TCON.7），片内定时/计数器 T1 溢出中断请求标志。当定时/计数器 T1 发生溢出时，置位 TF1，并向 CPU 申请中断。

5）RI（SCON.0）或 TI（SCON.1），串行口中断请求标志。当串行口接收完一帧串行数据时置位 RI 或当串行口发送完一帧串行数据时置位 TI，向 CPU 申请中断。

各中断源的中断向量见表 5-1。中断向量（Interrupt Vector）其实就是程序存储器的一个地址，表明一个中断的服务程序从这里开始存放。中断发生后要通过它引导 CPU 转向相应的中断服务。正因为它具有指向性，所以称其为中断向量（或中断矢量）。

在 80C51 的中断系统中，外部中断是由外部原因引起的，共有两个中断源，即外部中断 0 和外部中断 1。它们的中断请求信号分别由引脚 INT0（P3.2）和 INT1（P3.3）引入。

表 5-1　80C51 单片机的中断向量

中 断 名 称	中 断 向 量
外部中断 0（INT0）	0003H
定时器 0 中断（TF0）	000BH
外部中断 1（INT1）	0013H
定时器 1 中断（TF1）	001BH
串行口中断（RI）	0023H

外部中断请求有两种信号方式：电平方式和脉冲方式。两种信号方式可通过有关控制位进行定义。

电平方式的中断请求是低电平有效。只要单片机在中断请求引入端（INT0）或（INT1）上采样到有效的低电平信号，即为中断请求。

脉冲方式的中断请求则是脉冲的下降沿有效。在两个相邻机器周期所进行的两次采样中，若前一次为高，后一次为低，即为中断请求信号。为此，脉冲方式的中断请求信号的高、低电平状态都应至少维持一个机器周期，才能确保负脉冲的跳变能被采样到。

定时器中断是为满足定时或计数的需要而设置的。在单片机芯片内部有 2 个定时器/计数器 T0 和 T1，所以定时器中断也有 2 个：定时器 1 中断和定时器 0 中断。当计数器溢出时，表明定时时间到或计数值满，这时内部电路就产生中断请求。由于这种中断请求是在芯片内部发生的，因此，在芯片上没有对应的中断请求引入端。

串行中断包含串行发送中断和串行接收中断。它们对应同一个中断向量 0023H。串行中断是为串行数据传送而设置的。每当串行口发送或接收完一帧串行数据时，就产生相应的中断请求。同样因为中断请求是在芯片内部自动发生的，所以也不需在芯片上设置中断请求引脚。

5.2　80C51 单片机中断系统的结构及控制

5.2.1　80C51 单片机中断系统的结构

80C51 单片机的中断系统有 5 个中断源，2 个优先级，可实现二级中断嵌套。由片内中断允许寄存器 IE 控制 CPU 是否响应中断请求；由中断优先级寄存器 IP 安排各中断源的优先级，同一优先级内各中断源同时提出中断请求时，由内部的查询逻辑确定其响应次序。

80C51 单片机的中断系统由中断请求标志位（在相关的特殊功能寄存器中）、中断允许寄存器 IE、中断优先级寄存器 IP 及内部硬件查询电路组成，如图 5-1 所示。该图从逻辑上描述了 80C51 单片机中断系统的整体工作机制。

5.2.2　80C51 单片机中断系统的控制

这里所说的中断控制是指提供给用户使用的中断控制手段。具体到 80C51，中断控制的

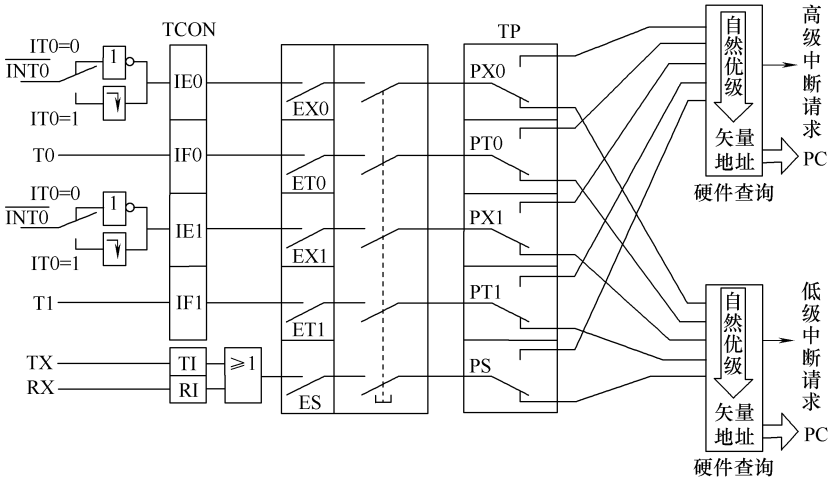


图 5-1 80C51 单片机中断系统结构图

内容共有 4 项：中断允许控制、中断请求标志、中断优先控制和外中断触发方式控制。这些控制内容分布在 4 个控制寄存器：中断允许寄存器、定时器控制寄存器、串行控制寄存器和中断优先级寄存器。中断控制是通过硬件实现的，但需进行软件设置。

1. 定时器控制寄存器 TCON

定时器控制寄存器 TCON 的作用是控制定时器的启动与停止，并保存 T0、T1 的溢出中断标志和外部中断的中断标志。该寄存器的地址是 88H，对应的位地址是 88H ~ 8FH。虽然该寄存器名称为定时器控制寄存器，但是多数位都是为中断控制而设置的。位定义及位地址见表 5-2。

表 5-2 定时器控制寄存器位定义及位地址

位地址	8FH	8EH	8DH	8CH	8BH	8AH	89H	88H
符号	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

说明：

1）TF1（TCON. 7）：定时器 1 溢出标志位。定时器 1 启动计数后，从初值开始进行加 1 计数；当定时器 1 计满溢出时，由硬件自动使 TF1 置 1，并申请中断。该标志一直保持到 CPU 响应中断后，才由硬件自动清 0。也可用软件查询该标志，并由软件清 0。

2）TR1（TCON. 6）：定时器 1 启停控制位。

3）IT1（TCON. 2）：外部中断 1 触发方式选择位。

当 IT1 = 0 时，外部中断 1 为电平触发方式。在这种方式下，CPU 在每个机器周期的 S5P2 期间对 INT1（P3. 3）引脚采样，若采到低电平，则认为有中断申请，硬件自动使 IE1 置 1；若为高电平，认为无中断请求或中断请求已撤除，硬件自动使 IE1 清 0。在电平触发方式中，CPU 响应中断后硬件不能自动使 IE1 清 0，也不能由软件使 IE1 清 0，所以在中断返回前必须撤销 INT1 引脚上的低电平，否则将再次响应中断造成出错。

当 IT1 = 1 时，外部中断 1 为边沿触发方式。CPU 在每个机器周期的 S5P2 期间采样 INT1（P3. 3）引脚。若在连续两个机器周期采样到先高电平后低电平，则认为有中断请求，

硬件自动使 IE1 置 1，此标志一直保持到 CPU 响应中断时才由硬件自动清 0。在边沿触发方式下，为保证 CPU 在两个机器周期内检测到先高后低的负跳变，输入高低电平的持续时间至少要保持 12 个时钟周期。

4）IE1（TCON. 3）：外部中断 1 请求标志位。IE1 = 1 表示外部中断 1 向 CPU 申请中断。当 CPU 响应外部中断 1 的中断请求时，由硬件自动使 IE1 清 0（边沿触发方式）。

5）TF0（TCON. 5）：定时器 0 溢出标志位。其功能同 TF1。

6）TR0（TCON. 4）：定时器 0 启、停控制位。其功能同 TR1。

7）IE0（TCON. 1）：外部中断 0 请求标志位。其功能同 IE1。

8）IT0（TCON. 0）：外部中断 0 触发方式选择位。其功能同 IT1。

2. 串行口控制寄存器 SCON

SCON 是一个用于串行数据通信控制的寄存器，其中只有 98H 和 99H 这两位与中断有关，已用黑体字表示出来。该寄存器的位定义及位地址见表 5-3。

表 5-3　串行口控制寄存器位定义及位地址

位地址	9FH	9EH	9DH	9CH	9BH	9AH	99H	98H
符号	SM0	SM1	SM2	REN	TB8	RB8	TI	RI

说明：

1）TI（SCON. 1）：串行发送中断请求标志。CPU 将一个字节数据写入发送缓冲器 SBUF 后启动发送，每发送完一帧数据，硬件自动使 TI 置 1。但 CPU 响应中断后，硬件并不能自动使 TI 清 0，必须由软件使 TI 清 0。

2）RI（SCON. 0）：串行接收中断请求标志。在串行口允许接收时，每接收完一帧数据，硬件自动使 RI 置 1。但 CPU 响应中断后，硬件并不能自动使 RI 清 0，必须由软件使 RI 清 0。

3. 中断允许寄存器 IE

该寄存器用于控制是否允许使用中断。中断允许寄存器地址为 A8H，位地址为 A8H ~ AFH。寄存器位定义及位地址见表 5-4。

表 5-4　中断允许寄存器位定义及位地址

位地址	AFH	AEH	ADH	ACH	ABH	AAH	A9H	A8H
符号	EA	/	/	ES	ET1	EX1	ET0	EX0

说明：

1）EA（IE. 7）：CPU 中断总允许位。EA = 1，CPU 开放中断。每个中断源是被允许还是被禁止，分别由各中断源的中断允许位确定；EA = 0，CPU 屏蔽所有的中断要求，称为关中断。

2）ES（IE. 4）：串行口中断允许位。ES = 1，允许串行口中断；ES = 0，禁止串行口中断。

3）ET1（IE. 3）：定时器 1 中断允许位。ET1 = 1，允许定时器 1 中断；ET1 = 0，禁止定时器 1 中断。

4）EX1（IE. 2）：外部中断 1 中断允许位。EX1 = 1，允许外部中断 1 中断；EX1 = 0，

禁止外部中断 1 中断。

5）ET0（IE.1）：定时器 0 中断允许位。ET0 = 1，允许定时器 0 中断；ET0 = 0，禁止定时器 0 中断。

6）EX0（IE.0）：外部中断 0 中断允许位。EX0 = 1，允许外部中断 0 中断；EX0 = 0，禁止外部中断 0 中断。

4. 中断优先级寄存器 IP

各中断的优先级通过中断优先级控制寄存器 IP 设定。该寄存器地址为 B8H，位地址为 B8H ~ BFH，其位定义及位地址见表 5-5。

表 5-5 中断优先级寄存器位定义及位地址

位地址	AFH	AEH	ADH	ACH	ABH	AAH	A9H	A8H
符号	/	/	/	PS	PT1	PX1	PT0	PX0

说明：

1）PS（IP.4）：串行口中断优先级控制位。PS = 1，串行口为高优先级中断；PS = 0，串行口为低优先级中断。

2）PT1（IP.3）：定时器 1 中断优先级控制位。PT1 = 1，定时器 1 为高优先级中断；PT1 = 0，定时器 1 为低优先级中断。

3）PX1（IP.2）：外部中断 1 中断优先级控制位。PX1 = 1，外部中断 1 为高优先级中断；PX1 = 0，外部中断 1 为低优先级中断。

4）PT0（IP.1）：定时器 0 中断优先级控制位。PT0 = 1，定时器 T0 为高优先级中断 PT0 = 0，定时器 0 为低优先级中断。

5）PX0（IP.0）：外部中断 0 中断优先级控制位。PX0 = 1，外部中断 0 为高优先级中断；PX0 = 0，外部中断 0 为低优先级中断。

5.2.3 80C51 单片机中断系统的优先级控制

中断优先级越高，则响应优先权就越高。当 CPU 正在执行中断服务程序时，又有中断优先级更高的中断请求产生，这时 CPU 就会暂停当前的中断服务转而处理高级中断请求，待高级中断处理程序完毕再返回原中断程序断点处继续执行，这一过程称为中断嵌套。

1. 中断优先级定义原则

80C51 的中断优先级控制比较简单，只划分为高、低两个优先等级，因此，就存在如何为一个具体中断定义优先等级的问题。下面是一些可供参考的基本原则：

1）中断的轻重缓急程度。例如，电源故障有使整个系统瘫痪的危险，必须及时处理，所以应安排为高优先级；而那些仅影响局部故障的中断或操作性中断（如输入/输出中断）应安排为低优先级。

2）中断设备的工作速度。快速设备需要及时响应，否则将有丢失数据的危险，所以应安排为高优先级。

3）中断处理的工作量。尽量把处理工作量小的中断安排为高优先级，因为处理工作量小，占用 CPU 的时间短。

4）中断请求发生的频繁程度。可以考虑将那些很少请求单片机干预的事件产生的中断

安排为高优先级。

2. 80C51 单片机中中断响应的优先级原则

1）当高、低优先级中断请求同时出现时，高优先级中断请求被响应。

2）如果同级的多个中断请求同时出现，则按照自然优先级顺序响应。自然优先级：外部中断 0→定时器 0 中断→外部中断 1 一定时器 1 中断→串行中断。

3. 80C51 单片机的中断嵌套

CPU 在响应某一中断源中断请求而进行中断处理时，若有中断优先级更高的中断源发出中断请求，CPU 会暂停正在执行的中断服务程序，转向执行中断优先级更高的中断源的中断服务程序，等处理完这个高优先级的中断请求后，再返回来继续执行被暂停的中断服务程序。这个过程称为中断嵌套，如图 5-2 所示。

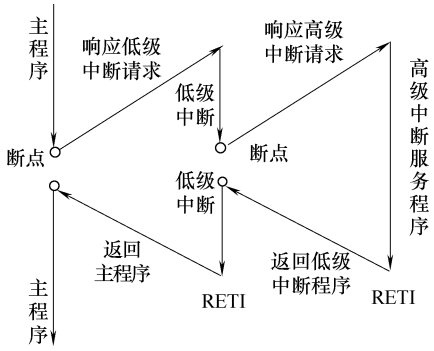


图 5-2　中断嵌套的示意图

5.3　80C51 单片机中断处理过程

5.3.1　中断响应的条件

CPU 响应中断必须首先满足以下三个基本条件：

1）有中断源发出中断请求。

2）中断总允许位 EA = 1。

3）请求中断的中断源的中断允许位为 1。

CPU 执行程序过程中，中断系统会在每个机器周期的 S5P2（第 5 状态第 2 拍节）对各个中断源进行采样，采样值在下一个机器周期内按优先级和内部顺序依次查询。若某个中断标志在上一个机器周期的 S5P2 时被置为 1，它将于现在的查询周期中被及时发现。接着 CPU 便执行一条由中断系统提供的硬件指令 LCALL，转向中断向量的特定地址单元，进入响应的中断服务程序。

在满足以上条件的基础上，若有下列任何一种情况存在，中断响应都会受到阻断：

1）CPU 正在执行一个同级或高优先级的中断服务程序。

2）正在执行的指令尚未执行完。

3）正在执行中断返回指令 RETI 或者对专用寄存器 IE、IP 进行读/写的指令。CPU 在执

行完上述指令之后，要再执行一条指令，才能响应中断请求。

若由于上述条件的阻碍中断未能得到响应，当条件消失时该中断标志却已不再有效，那么该中断将不被响应。也就是说，中断标志曾经有效，但未被响应，查询过程在下个机器周期将重新进行。

5.3.2 中断响应的过程

从中断请求发生到中断被响应，再转向执行中断服务程序去完成中断所要求的操作，是一个完整的中断处理过程。下面介绍 80C51 单片机的中断响应过程。

1. 外部中断请求采样

只有外部中断请求才有采样问题，因为它们来自单片机芯片的外部，而且是随机的，只有通过采样才能知道是否有中断请求信号到来。

采样是在每个机器周期的 S5P2（第 5 状态第 2 拍节）对芯片引脚 INT0（P3.2）和 INT1（P3.3）进行的，根据采样结果来设置定时器控制寄存器 TCON 外部中断标志位的状态，从而把外部中断请求锁定在这个寄存器中。对于电平方式的外部中断请求，采样到低电平即为有效的中断请求，应把 IE0（或 IE1）置 1；对于脉冲方式的外部中断请求，若两个相邻机器周期的采样结果为先高电平后低电平，则为有效的中断请求信号，应把 IE0（或 IE1）置 1。

2. 中断查询

因为中断发生是随机的，无法事先预知，所以必须主动检测，这一过程称为中断查询。中断查询是查看是否有中断请求发生并确定是来自哪一个中断源的中断请求。中断查询操作是由 CPU 逐个检测定时器控制寄存器 TCON 和串行控制寄存器 SCON 中各中断标志位的状态而实现的，因为所有中断请求最终都要汇集到这两个寄存器中。其中，外部中断的中断请求是通过采样得到的；而定时中断和串行中断的中断请求就发生在芯片内部，若有中断发生就通过硬件把相应的标志位直接置位。

80C51 单片机是在每一个机器周期的最后一个状态 S6 进行中断查询，查询按优先级顺序进行。具体为先高级中断后低级中断，同级中断按“外部中断 0→定时器 0 中断→外部中断 1→定时器 1 中断→串行中断”的顺序进行。如果查询到有标志位为 1，则表明有中断请求发生，接着就从相邻的下一个机器周期的 S1 状态开始进行中断响应。

中断请求汇集及查询如图 5-1 所示。由于中断请求是随机发生的，是 CPU 无法预知的，因此，在程序执行过程中，中断查询要在指令执行的每个机器周期进行一遍。

3. 中断响应

中断响应就是接受中断源提出的中断请求。在一次中断查询之后，当发现有中断请求时，紧接着就进行中断响应。

中断响应的主要内容是由硬件自动生成一条长调用指令，指令格式为“LCALL addr16”。这里的 addr16 就是程序存储器中断区中相应中断的入口地址，在 80C51 单片机中，这些入口地址已由系统设定。例如，对于外部中断 0 的响应，产生的长调用指令为“LCALL 0003H”。

生成 LCALL 指令后，紧接着由 CPU 执行。首先将程序计数器 PC 的内容压入堆栈以保护断点，再将中断入口地址装入 PC，使程序执行转向相应的中断入口地址。

中断响应是有条件的，并不是查询到的中断请求都能立即响应。当存在下列情况之一时，中断响应将被封锁：

1）CPU 正处在为一个同级或高级的中断服务中。因为当一个中断被响应时，要把对应的优先级触发器置位，即封锁了低级和同级中断的响应。

2）查询中断请求的机器周期不是当前指令的最后一个机器周期。作此限制的目的在于使当前指令执行完毕后，才能进行中断响应，以确保指令的完整执行。

3）当前指令是返回指令（RET，RETI）或访问 IE、IP 的指令。因为 80C51 中断系统规定，在执行完这些指令之后，还应再继续执行一条指令，然后才能响应中断。

80C51 对中断查询的结果不作记忆，由于上述这些原因而被拖延的查询结果将不复存在。其后将按新的查询结果进行中断响应。

4. 中断响应的快慢

如果中断查询的机器周期恰好是指令的最后一个机器周期，则最快只需 3 个机器周期就可以转向中断服务程序的入口。其中，查询占 1 个机器周期，在这个机器周期结束后中断即被响应，生成 LCALL 指令。执行这条长调用指令需要 2 个机器周期。

实际上中断响应不一定这么顺利，下面看一个中断响应最慢的情况。如果中断查询刚好是开始执行 RET、RETI 或访问 IE、IP 的指令，则需把当前指令执行完再继续执行一条指令后，才能进行中断响应。这些指令中最长执行时间需 2 个机器周期。而如果接着再执行的指令恰好是 MUL（乘）或 DIV（除）指令，则又需 4 个机器周期。再加上执行长调用指令 LCALL 所需的 2 个机器周期，从而形成了 8 个机器周期的最长响应时间。

一般情况下，中断响应时间的长短无需考虑，只有在精确定时的应用场合才认真对待中断响应时间，以保证定时的精确控制。

5.3.3　中断服务程序

当单片机接收到一个中断请求信号后，就暂停它的当前操作，保存其工作状态，并将控制权转交给中断服务程序，以便通过执行中断服务程序（Interrupt Handler）来完成该中断所对应的操作内容。

1. 主程序中的中断初始化

中断都是在运行主程序时发生的，是主程序的随机事件。是否允许发生及如何发生，都应该在主程序中预先设置，这就是中断初始化。中断初始化的内容包括堆栈设置、中断系统总开放、中断允许设置、中断请求方式设置（外部中断）和中断优先级设置等。

现以外部中断 0 为例进行说明，外部中断 0 的中断地址区从 0003H 开始，假定外部中断 0 的中断服务程序入口地址标号为 EXINT0，则转向中断服务程序的设置和中断初始化的程序代码如下：

```
        ORG     0000H
        AJMP    MAIN            ；转向主程序
        ORG     0003H
        AJMP    EXINT0          ；转向外部中断 0 服务子程序
MAIN:   MOV     TCON，#01H      ；中断请求方式设置：脉冲触发
```

```
        MOV     IE，#81H        ；中断开发，外部中断0允许
        MOV     IP，#01H        ；外部中断0为高优先级，其余为低优先级
        MOV     SP，#03H        ；设置堆栈
        ……      ……
EXINT0：                        ；外部中断0服务程序
        ……      ……
```

使用与中断相关的控制寄存器时，既可以按寄存器名称又可以按寄存器地址，此外对位状态的设置还可以使用位操作指令。例如，设置外中断0为高优先级，其余为低优先级的位操作指令为

```
MOV   IP，#00H        ；清中断优先级寄存器
SETB  PX0             ；外部中断0为高优先级
```

有时在主程序空闲时，需要等待中断请求出现，为此可在需要等待的地方使用一条SJMP指令进行设置，即

```
THERE：SJMP  THERE
```

2. 中断服务流程

所有计算机的中断服务流程都十分相似，单片机也不例外。80C51单片机的中断服务流程如图5-3所示。流程图表明，只有在一条指令全部执行完之后，才能响应中断请求，以确保指令的完整执行。下面对中断服务流程中的一些问题进行说明。

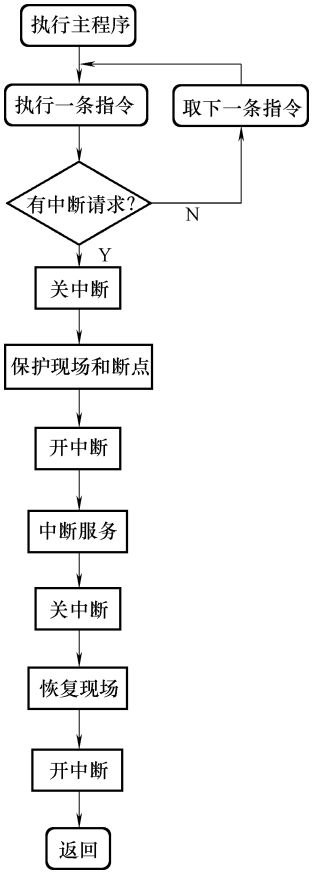


图5-3 80C51单片机的中断服务流程图

1）现场保护和现场恢复。所谓现场就是指中断时刻单片机中存储单元内的数据或状态。为了使中断服务程序的执行不破坏这些数据或状态，就要把它们送入堆栈中保存起来，以免在中断返回后影响主程序的运行。这就是现场保护，现场保护一定要完成于中断处理程序之前。

中断服务结束后，在返回主程序之前，应把保存的现场内容从堆栈中弹出，以恢复相关存储单元的原有内容，这就是现场恢复。现场恢复一定要在中断处理程序之后进行。

80C51中有堆栈操作指令“PUSH direct”和“POP direct”，主要是供现场保护和现场恢复的。至于要保护哪些现场内容，应该由用户根据中断处理程序的情况来决定。

2）关中断和开中断。在一个多中断源系统中，为保证重要中断能执行到底，不被其他中断所嵌套，除采用设定高优先级之外，还可以采用关中断的方法来解决。即在现场保护之前先关闭中断系统，彻底屏蔽其他中断，待中断处理完成后再打开中断系统。即使中断处理可以被嵌套，但现场保护和恢复不允许打扰，以免影响现场保护和恢复工作，为此应在现场保护和现场恢复程

序段的前后进行关、开中断。这样做可以在除现场保护和现场恢复的片刻外，仍然为系统保留中断嵌套功能。

对于80C51 单片机，中断的关和开可通过 CLR 和 SETB 指令复位、置位中断允许控制寄存器中的有关位来实现。

3）中断处理。中断处理是中断服务程序的核心内容，中断要做的事全在其中体现。

4）中断返回。中断服务程序的最后一条指令必须是中断返回指令 RETI。CPU 执行这条指令时，把响应中断时置位的优先级触发器复位，再从堆栈中弹出断点地址送入程序计数器 PC，以便从断点处重新执行被中断的主程序。

【例 5-1】 P1 口做输出口，正常时控制 8 只灯（P1 口输出低电平时灯被点亮）每隔 0.5s 全亮全灭一次；按下开关 1，8 只灯从右向左依次点亮；按下开关 2，8 只灯从左向右依次点亮。

解：开关 1 的低电平脉冲信号作为外部中断信号由 INT0（P3.2）引脚输入，开关 2 的低电平信号作为外部中断信号由 INT1（P3.3）引脚输入。中断允许寄存器 IE 中相应的 EA、EX1、EX0 位设置为 1。

外部中断 0 为低优先级，IP 中的 PX0 位设置为 0；外部中断 1 为高优先级，IP 中的 PX1 位设置为 1。

外部中断 0 的中断触发方式设为边沿触发，控制位 IT0 应设置为 1；外部中断 1 的中断触发方式设为电平触发，控制位 IT1 应设置为 0。

主程序：

```
                        ORG   0000H          ；程序入口
0000H   02  00  30      LJMP  MAIN           ；转向主程序
                        ORG   0003H          ；外部中断 0 的入口地址
0003H   02  01  00      LJMP  INT0           ；转向外部中断 0 中断服务程序
                        ORG   0013H          ；外部中断 1 的入口地址
0013H   02  02  00      LJMP  INT1           ；转向外部中断 1 中断服务程序
                        ORG   0030H
0030H   75  81  30  MAIN:MOV  SP，#30H
0033H   75  A8  85      MOV   IE，#85H       ；允许外部中断 0、外部中断 1 中断
0036H   75  B8  04      MOV   IP，#04H       ；外部中断 1 为高优先级
0039H   75  88  01      MOV TCON，#01H       ；外部中断 0 为边沿触发
003CH   74  00          MOV   A，#00H
003EH   F5  90      LP1:MOV   P1，A
0040H   12  03  00      LCALL DELAY
0043H   F4              CPL   A
0044H   80  F8          SJMP  LP1
```

外部中断 0 中断服务程序：

```
                        ORG   0100H
0100H   C0  E0      INT0:PUSH   A          ; 外部中断 0 中断服务程序
0102H   C0  D0           PUSH   PSW
0104H   C2  D4           CLR    RS1        ; 选择第 1 组工作寄存器
0106H   D2  D3           SETB   RS0
0108H   7A  07           MOV    R2, #07H
010AH   74  FE           MOV    A, #0FEH   ; 灯点亮的初始状态
010CH   F5  90    NEXT0:MOV    P1, A
010EH   12  03  00       LCALL  DELAY
0111H   23               RL     A          ; 点亮左边一盏灯
0112H   DA  F8           DJNZ R2, NEXT0
0114H   D0  D0           POP    PSW
0116H   D0  E0           POP    A
0118H   32               RETI
```

外部中断 1 中断服务程序：

```
                        ORG   0200H
0200H   C0  E0      INT1:PUSH   A          ; 外部中断 1 中断服务程序
0202H   C0  D0           PUSH   PSW
0204H   D2  D4           SETB   RS1        ; 选择第 2 组工作寄存器
0206H   C2  D3           CLR    RS0
0208H   7A  07           MOV    R2, #07H
020AH   74  7F           MOV    A, #7FH    ; 灯点亮的初始状态
020CH   F5  90    NEXT1:MOV    P1, A
020EH   12  03  00       LCALL  DELAY
0211H   03               RR     A          ; 点亮右边一盏灯
0212H   DA  F8           DJNZ   R2, NEXT1
0214H   D0  D0           POP    PSW
0216H   D0  E0           POP    A
0218H   32               RETI
```

延时子程序：

```
                        ORG   0300H
0300H   7B  FA    DELAY:MOV   R3，#250        ；延时子程序
0302H   7A  F8     DEL2:MOV   R2，#248
0304H   00                NOP
0305H   DA  FD     DEL1:DJNZ  R2，DEL1
0307H   DB  F9           DJNZ  R3，DEL2
0309H   22                RET                ；子程序返回
                          END
```

5.4　80C51 单片机的定时器/计数器

80C51 单片机内部设有两个 16 位可编程定时器/计数器，简称为定时器 0T0 和定时器 1T1。

16 位的定时器/计数器分别由两个 8 位寄存器组成，T0 由 TH0 和 TL0 构成，T1 由 TH1 和 TL1 构成。每个寄存器均可单独访问，这些寄存器是用于存放定时初值或计数初值的。

有一个 8 位的定时器方式寄存器 TMOD 和一个 8 位的定时器控制寄存器 TCON。这些寄存器之间是通过内部总线和控制逻辑电路连接起来的。定时器/计数器的工作方式、定时时间和启停控制通过指令确定这些寄存器的状态来实现。TMOD 主要用于设定定时器的工作方式，TCON 主要用于控制定时器的启动与停止，并保存 T0、T1 的溢出和中断标志。80C51 单片机的定时器/计数器的结构如图 5-4 所示。

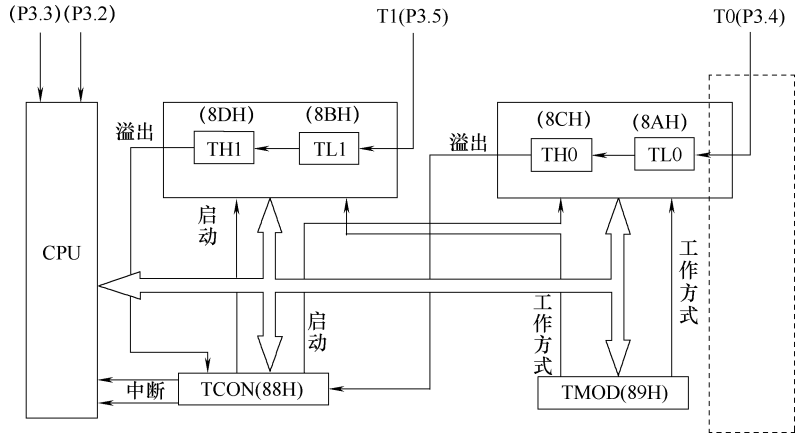


图 5-4　80C51 单片机的定时器/计数器

5.4.1　定时器/计数器的原理

16 位的定时器/计数器实质上是一个加 1 计数器，可实现定时和计数两种功能，其功能由软件控制和切换。定时器属硬件定时和计数，是单片机中效率高而且工作灵活的部件。

在定时器/计数器开始工作之前，CPU 必须将一些命令（称为控制字）写入定时器/计数器。将控制字写入定时器/计数器的过程叫定时器/计数器的初始化。

在初始化程序中，要将工作方式控制字写入定时器方式寄存器（TMOD），工作状态控制字（或相关位）写入定时器控制寄存器（TCON），赋定时/计数初值给 TH0（TH1）和 TL0（TL1）。

1. 定时器/计数器的定时功能

计数器的加 1 信号由振荡器的 12 分频信号产生，即每过一个机器周期计数器加 1，直至计满溢出。

定时器的定时时间与系统的时钟频率有关。因一个机器周期等于 12 个时钟周期，所以计数频率应为系统时钟频率的 1/12。如果晶振频率为 12MHz，则机器周期为 1μs。通过改变定时器的定时初值，并适当选择定时器的长度（8 位、13 位或 16 位），可以调整定时时间。

2. 定时器/计数器的计数功能

通过外部计数输入引脚 T0（P3.4）和 T1（P3.5）对外部信号计数，外部脉冲的下降沿将触发计数。计数器在每个机器周期的 S5P2 期间采样引脚输入电平，若一个机器周期 S5P2 期间采样值为 1，下一个机器周期 S5P2 期间采样值为 0，则计数器加 1，再下一个机器周期 S3P1 期间，新的计数值装入计数器。

因检测一个由 1 至 0 的跳变需要两个机器周期，故外部信号的最高计数频率为时钟频率的 1/24。如果晶振频率为 12MHz，则最高计数频率为 0.5MHz。虽然对外部输入信号的占空比无特殊要求，但为了确保给定电平在变化前至少被采样一次，外部计数脉冲的高电平与低电平保持时间均需在一个机器周期以上。

5.4.2 定时器/计数器的应用

定时器/计数器具有定时和计数两种功能，应用范围如下。

1）定时与延时控制方面，可产生定时中断信号，以设计出各种不同频率的信号源；产生定时扫描信号，对键盘进行扫描以获得控制信号，对显示器进行扫描以不间断地显示数据。

2）测量外部脉冲方面，对外部脉冲信号进行计数可测量脉冲信号的宽度、周期，也可实现自动计数。

3）监控系统工作方面，对系统进行定时扫描，当系统工作异常时，使系统自动复位，重新启动以恢复正常工作。

5.4.3 定时器/计数器的控制

1. 定时器控制寄存器 TCON

定时器控制寄存器 TCON 的作用是控制定时器的启动与停止，并保存 T0、T1 的溢出和中断标志。TCON 寄存器地址 88H，位地址为 8FH ~ 88H。该寄存器位定义及位地址见表 5-6。

表 5-6　TCON 的格式

位地址	8FH	8EH	8DH	8CH	8BH	8AH	89H	88H
符号	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

说明：

（1）TF1（TCON.7）：定时器 1 溢出标志位。当定时器 1 计满溢出时，由硬件自动使 TF1 置 1，并申请中断。对该标志位有两种处理方法：一种是以中断方式工作，即 TF1 置 1 并申请中断，响应中断后执行中断服务程序，并由硬件自动使 TF1 清 0；另一种以查询方式工作，即通过查询该位是否为 1 来判断是否溢出，TF1 置 1 后必须用软件使 TF1 清 0。

（2）TR1（TCON.6）：定时器 1 启停控制位。

① GATE = 0 时，用软件使 TR1 置 1 即启动定时器 1，若用软件使 TR1 清 0 则停止定时器 1。

② GATE = 1 时，用软件使 TR1 置 1 的同时外部中断 INT1 的引脚输入高电平才能启动定时器 1。

（3）TF0（TCON.5）：定时器 0 溢出标志位。其功能同 TF1。

（4）TR0（TCON.4）：定时器 0 启停控制位。其功能同 TR1。

（5）IE1（TCON.3）：外部中断 1 请求标志位。

（6）IT1（TCON.2）：外部中断 1 触发方式选择位。

（7）IE0（TCON.1）：外部中断 0 请求标志位。

（8）IT0（TCON.0）：外部中断 0 触发方式选择位。

2. 定时器方式寄存器 TMOD

定时器方式寄存器 TMOD 的作用是设置 T0、T1 的工作方式，该寄存器定义及位地址见表 5-7。

表 5-7　TMOD 的格式

位地址	B7H	B6H	B5H	B4H	B3H	B2H	B1H	B0H
符号	GATE	$C/\overline{T}$	M1	M0	GATE	$C/\overline{T}$	M1	M0

说明：

（1）GATE：门控位。

GATE = 0 时，软件启动定时器，即用指令使 TCON 中的 TR1（TR0）置 1 即可启动定时器 1（定时器 0）。

GATE = 1 时，软件和硬件共同启动定时器，即用指令使 TCON 中的 TR1（TR0）置 1 时，只有外部中断 INT0（INT1）引脚输入高电平时才能启动定时器 1（定时器 0）。

（2）$C/\overline{T}$：功能选择位。

$C/\overline{T}$ = 0 时，以定时器方式工作。

$C/\overline{T}$ = 1 时，以计数器方式工作。

（3）M1、M0：方式选择位。定时器工作方式选择位定义见表 5-8。

表 5-8 定时器工作方式选择位定义

M1 M0	工作方式	功能描述
0 0	方式 0	13 位计数器
0 1	方式 1	16 位计数器
1 0	方式 2	自动重装初值 8 位计数器
1 1	方式 3	定时器 0：分为两个独立的 8 位计数器
		定时器 1：无中断的计数器

3. 中断允许控制寄存器（IE，见表 5-9）

表 5-9 中断允许控制寄存器

位地址	AFH	AEH	ADH	ACH	ABH	AAH	A9H	A8H
符号	EA	/	/	ES	ET1	EX1	ET0	EX0

其中与定时器/计数器有关的是定时器/计数器中断允许控制位 ET0 和 ET1。

ET0（ET1）=0，禁止定时器中断。

ET0（ET1）=1，允许定时器中断。

以上为 80C51 单片机定时系统提供给用户使用的硬件内容，共有 4 个 8 位定时器 TH0、TH1、TL0、TL1，及上述 3 个控制寄存器。

5.5 定时器/计数器的四种工作方式及其应用

定时器/计数器共有四种工作方式，两个定时器/计数器 T0 和 T1 并不是全部具备这四种工作方式。定时器/计数器 T0 有四种工作方式（方式 0、1、2、3），T1 有三种工作方式（方式 0、1、2）。前三种工作方式，T0 和 T1 除了所使用的寄存器、有关控制位、标志位不同外，其他操作完全相同。我们在使用的时候可以结合实际要求来选择工作方式，通过定时器方式寄存器 TMOD 来进行设定。

5.5.1 定时器/计数器的初始化

1. 定时器/计数器的初始化步骤

定时器/计数器是一种可编程部件，在使用定时器/计数器前，一般都要对其进行初始化，以确定其以特定的功能工作。初始化的步骤如下：

（1）确定定时器/计数器的工作方式，确定方式控制字，并写入 TMOD。

（2）预置定时初值或计数初值，根据定时时间或计数次数，计算定时初值或计数初值，并写入 TH0、TL0 或 TH1、TL1。

（3）根据需要开放定时器/计数器的中断，给 IE 中的相关位赋值。

（4）启动定时器/计数器，给 TCON 中的 TR1 或 TR0 置 1。

2. 定时初值或计数初值的计算方法

不同工作方式的定时初值及计数初值的计算方法见表 5-10 所示。

表 5-10　不同工作方式的定时初值及计数初值的计算方法

工作方式	计数位数	最大计数值	最大定时时间	定时初值计算公式	计数初值计算公式
方式 0	13	$2^{13}=8192$	$2^{13}\times T_{机}$	$X=2^{13}-T/T_{机}$	$X=2^{13}-$计数值
方式 1	16	$2^{16}=65536$	$2^{16}\times T_{机}$	$X=2^{16}-T/T_{机}$	$X=2^{16}-$计数值
方式 2	8	$2^{8}=256$	$2^{8}\times T_{机}$	$X=2^{8}-T/T_{机}$	$X=2^{8}-$计数值

【例 5-2】 用定时器 0 方式 0，定时 5ms，以中断方式工作，进行程序初始化设计，晶振频率为 6MHz。

解：用定时器 0 方式 0 时，定时器/计数器方式寄存器 TMOD 低 4 位中的 M1M0 应取 00；可设定为软件启动定时器，故 GATE 取 0；因用定时功能，C/T 取 0；定时器方式寄存器 TMOD 高 4 位为无关位，一般都取 0，所以 TMOD 应为 00H。

晶振频率为 6MHz，则

$$T_{机}=12/f_{osc}=[12/(6\text{MHz})]=2\mu\text{s}$$

定时初值为

$$X=2^{13}-T/T_{机}=2^{13}-5\times10^{3}/2=8192-2500=5692$$
$$=163\text{CH}=1011000111100\text{B}$$

因 TL0 的高 3 位未用，对计算出的定时初值 X 要进行修正，即在低 5 位前插入 3 个 0，修正后的定时初值

$$X=10110001\ 00011100\text{B}=\text{B11CH}$$

定时器以中断方式工作，故将中断总允许位 EA 和定时器 0 的中断允许位 ET0 置 1。

参考程序：

```
75  89  00      MOV   TMOD, #00H     ; 置定时器 0 为工作方式 0
75  8C  B1      MOV   TH0, #0B1H     ; 定时初值的高 8 位
75  8A  1C      MOV   TL0, #1CH      ; 定时初值的低 8 位
D2  AF          SETB  EA             ; 开放中断总允许位
D2  A9          SETB  ET0            ; 开放定时器 0 的中断允许位
D2  8C          SETB  TR0            ; 启动定时器 0
```

5.5.2　方式 0 及应用实例

1. 电路逻辑结构

不同工作方式下定时器/计数器的逻辑结构有所不同。工作方式 0 是 13 位计数结构，计数器由 TH0 的全部 8 位和 TL0 的低 5 位构成，TLO 的高 3 位不用。图 5-5 所示为定时器/计数器 0 工作方式 0 的逻辑结构。

在工作方式 0 下，计数脉冲既可以来自芯片内部，也可以来自外部。来自内部的是机器周期脉冲，图中 OSC 是振荡器的英文 Oscillator 的缩写。晶振脉冲经 12 分频后，即为单片机的机器周期脉冲。来自外部的计数脉冲由 T0（P3.4）引脚输入，计数脉冲由控制寄存器

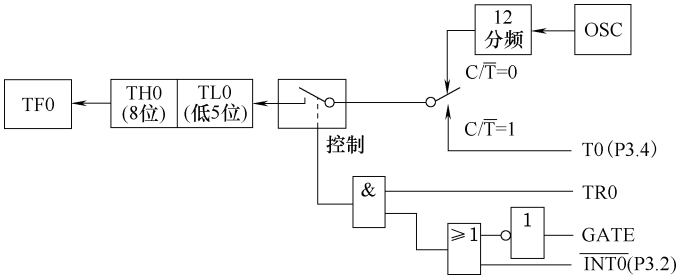


图 5-5 定时器/计数器 0 工作方式 0 的逻辑结构

TMOD 的 C/T 位进行控制。当 C/T = 0 时，接通机器周期脉冲，计数器每个机器周期进行一次加 1，这就是定时器工作方式；当 C/T = 1 时，接通外部计数引脚 T0（P3.4），从 T0 引入计数脉冲输入，这就是计数工作方式。

不管是哪种工作方式，当 TL0 的低 5 位计数溢出时，向 TH0 进位；而全部 13 位计数溢出时，向计数溢出标志位 TF0 进位，将其置 1。

2. 启停控制

定时器/计数器的启停控制有两种方法：一种是纯软件方法；另一种是软件和硬件相结合的方法。两种方法由门控位（GATE）的状态进行选择。

当 GATE = 0 时，为纯软件启停控制。GATE 信号反相为高电平，经“或”门后，打开了“与”门，这样 TR0 的状态就可以控制计数脉冲的通断，而 TR0 位的状态又是通过指令设置的，所以称为软件方式。当把 TR0 设置为 1，控制开关接通，计数器开始计数，即定时器/计数器工作；当把 TR0 清 0 时，开关断开，计数器停止计数。

当 GATE = 1 时，为软件和硬件相结合的启停控制方式。这时计数脉冲的接通与断开决定于 TR0 和 INT0 的“与”关系，而 INT0 是 P3.2 引脚引入的控制信号。由于 P3.2 引脚信号可控制计数器的启停，所以可利用 80C51 的定时器/计数器进行外部脉冲信号宽度的测量。

3. 定时利计数范围

使用工作方式 0 的计数功能时，计数值的范围是 1 ~ 2^{13}。使用工作方式 0 的定时功能时，定时时间的计算公式为

$$（2^{13} - 计数初值） \times 机器周期$$

又因为晶振脉冲经 12 分频后，得到单片机的机器周期脉冲，所以机器周期脉冲的周期是晶振脉冲周期的 12 倍，从而可得定时时间的计算公式又为

$$（2^{13} - 计数初值） \times 晶振周期 \times 12$$

其时间单位与晶振周期或机器周期的时间单位相同，为 μs。若晶振频率为 6MHz，则最小定时时间为

$$[2^{13} - （2^{13} - 1）] \times \frac{1}{6\text{MHz}} \times 12 = 2 \times 10^{-6}\text{s} = 2\mu\text{s}$$

最大定时时间为

$$[2^{13} - 0] \times \frac{1}{6\text{MHz}} \times 12 = 16384 \times 10^{-6}\text{s} = 16384\mu\text{s}$$

4. 应用实例

【例 5-3】 P1.0 口输出周期为 1 ms（频率 1kHz）的方波，采用定时器 1 方式 0 设计程序，晶振频率为 12MHz。

解：根据题意，只要使 P1.0 口每隔 500μs 取反一次即可得到周期为 1 ms 的方波，因而 T1 的定时时间为 500μs。

用定时器 1 方式 0 时，定时器/计数器方式寄存器 TMOD 高 4 位中的 M1M0 应取 00；可设定为软件启动定时器，故 GATE 取 0；因为用定时功能，C/T 取 0；定时器方式寄存器 TMOD 低 4 位为无关位，一般都取 0，所以 TMOD 应为 00H。

晶振频率为 12MHz，则

$$T_{机} = 12/f_{osc} = \frac{12}{12\text{MHz}} = 1\mu\text{s}$$

定时初值

$$X = 2^{13} - T/T_{机} = 2^{13} - 500/1 = 8192 - 500 = 7692$$
$$= 1\text{E0CH} = 1111000001100\text{B}$$

因 TL1 的高 3 位未用，对计算出的定时初值 X 要进行修正，即在低 5 位前插入 3 个 0，修正后的定时初值 $X = 1111000000001100\text{B} = \text{F00CH}$

参考程序：

```
地址      机器码          程序              注释
                    ORG   0000H
0000H   02  00  50      LJMP  MAIN
                    ORG   0050H
0050H   D2  90      MAIN:SETB  P1.0        ; 置 P1.0 初始状态
0052H   75  89  00      MOV   TMOD, #00H   ; 置定时器 1 为工作方式 0
0055H   75  8D  F1      MOV   TH1, #0F0H   ; 置 500μs 定时初值
0058H   75  8B  0C      MOV   TL1, #0CH
005BH   D2  8E          SETB  TR1          ; 启动定时器 1
005DH   10  8F  02    LP1:JBC   TF1, LP2   ; 查询计数溢出
0060H   80  FB          SJMP  LP1          ; 未到 500μs 继续计数
0062H   75  8D  F1    LP2:MOV   TH1, #0F0H ; 重新置 500μs 定时初值
0065H   75  8B  0C      MOV   TL1, #0CH
0068H   B2  90          CPL   P1.0         ; 输出取反
006AH   80  E4          SJMP  LP1          ; 重复循环
                    END
```

5.5.3　方式 1 及应用实例

方式 1 是 16 位计数结构的工作方式，计数器由 TH0 的全部 8 位和 TL0 的全部 8 位构成。

它的逻辑电路和工作情况与方式 0 完全相同，所不同的是计数器的位数。

使用工作方式 1 的计数功能时，计数值的范围是 1 ~ 65536（即 2^{16}）。使用工作方式 1 的定时功能时，定时时间计算公式为

$$(2^{16} - \text{计数初值}) \times \text{机器周期}$$

或

$$(2^{16} - \text{计数初值}) \times \text{晶振周期} \times 12$$

定时时间单位与晶振周期或机器周期的时间单位相同，为 μs。若晶振频率为 6MHz，则最小定时时间为

$$[2^{16} - (2^{16} - 1)] \times \frac{1}{6\text{MHz}} \times 12 = 2 \times 10^{-6}\text{s} = 2\mu\text{s}$$

最大定时时间为

$$[2^{16} - 0] \times \frac{1}{6\text{MHz}} \times 12 = 131072 \times 10^{-6}\text{s} = 131072\mu\text{s} \approx 131\text{ms}$$

在方式 1 下，以定时器 0 为例，定时器/计数器是一个由 TH0 中的 8 位和 TL0 中的 8 位组成的 16 位加 1 计数器。

方式 1 与方式 0 基本相似，最大的区别是方式 1 的加 1 计数器位数是 16 位。

5.5.4 方式 2 及应用实例

工作方式 0 和工作方式 1 有一个共同特点，就是计数溢出后计数器全为 0。因此，循环定时应用时就需要反复设置计数初值。这不但影响定时精度，而且也给程序设计带来麻烦。工作方式 2 就是针对此问题而设置的，该方式是定时器自动重装载的操作方式，在这种方式下，它的工作过程与方式 0、方式 1 基本相同，只不过在溢出的同时，将 8 位二进制初值自动重装载，即在中断服务子程序中不需要编程送初值。

1. 电路逻辑结构

在工作方式 2 下，16 位计数器被分为两部分，TL 作为计数器使用，TH 作为预置寄存器使用，初始化时把计数初值分别装入 TL 和 TH 中。当计数溢出后，由预置寄存器 TH 以硬件方法自动给计数器 TL 重新加载。图 5-6 所示为定时器/计数器 0 在工作方式 2 下的逻辑结构。

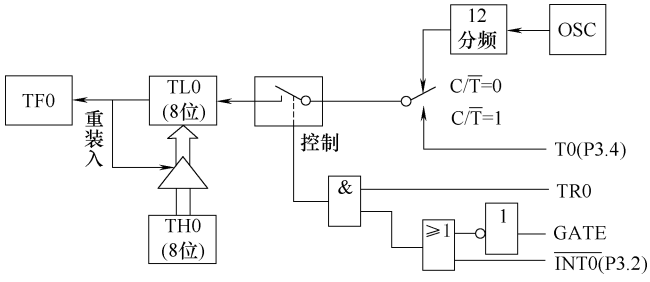


图 5-6 定时器/计数器 0 工作方式 2 的逻辑结构

初始化时，8 位计数初值同时装入 TL0 和 TH0 中。当 TL0 计数溢出时，置位 TF0，并用保存在预置寄存器 TH0 中的计数初值自动加载 TL0，然后开始重新计数，如此重复。这样不但省去了用户程序中的重装指令，而且也有利于提高定时精度。但这种工作方式是 8 位计数

结构，计数值有限，最大只能到 255。

2. 循环定时应用

【例 5-4】 P1.1 输出脉冲宽度调制（PWM）信号，即脉冲频率为 1kHz、占空比为 2:5 的矩形波，以控制直流电动机按一定的速度转动，晶振频率为 6MHz。

解： 频率为 1kHz、周期为 1ms、占空比为 2:5 的 P1.1 输出的矩形波的波形如图 5-7 所示。

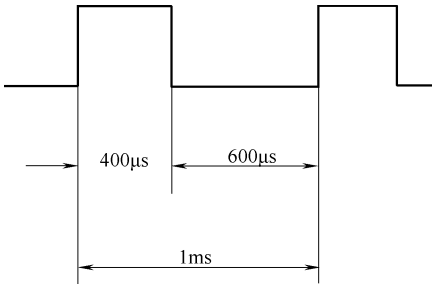


图 5-7　矩形波的波形

对 P1.1 取反时，由于高、低电平的时间不同，可找出一个时间基准，如 100μs、200μs。本例设定时间基准 200μs，即定时时间为 200μs。

定时初值为

$$X = 2^8 - T/T_{机} = 28 - 200/2 = 256 - 100 = 156 = 9CH$$

高电平的软件计数为 2，低电平的软件计数为 3。定时器以查询方式工作。

主程序：

```
                        ORG   0000H
0000H   02 00 30        LJMP  MAIN
                        ORG   000BH        ; 定时器 0 的中断入口地址
000BH   02 00 50        LJMP  INT0
                        ORG   0030H
0030H   D2 91     MAIN: SETB  P1.1         ; 设置 P1.1 初始状态
0032H   7A 02           MOV   R2, #02H     ; 给 R2 赋高电平计数值
0034H   75 89 02        MOV   TMOD, #02H   ; 定时器 0 工作方式 2
0037H   75 8C 9C        MOV   TH0, #9CH    ; 置 200μs 定时初值
003AH   75 8A 9C        MOV   TL0, #9CH
003DH   D2 AF           SETB  EA
003FH   D2 A9           SETB  ET0
0041H   D2 8C           SETB  TR0
0043H   80 FE           SJMP  $            ; 动态暂停
```

定时器0中断服务程序：

```
                        ORG   0050H
0050H   DA 0B     INT0: DJNZ  R2, EXIT
0052H   B2 91           CPL   P1.1          ; P1.1 取反
0054H   30 91 04        JNB   P1.1, L1      ; 判断 P1.1 电平高低
0057H   7A 02           MOV   R2, #02H      ; 若 P1.1 =1，给 R2 赋高电平计数值
0059H   80 02           SJMP  EXIT
005BH   7A 04       L1: MOV   R2, #03H      ; 若 P1.1 =0，给 R2 赋低电平计数值
005DH   32        EXIT: RETI
                        END
```

【例5-5】 定时器0外部输入端P3.4作为计数脉冲输入端，利用手控单脉冲信号作为计数输入脉冲，编写控制程序，每输入10个脉冲，工作寄存器R0的内容加一，晶振频率为6MHz。

解： 定时器以中断方式工作。用定时器0方式2时，定时器/计数器方式寄存器TMOD低4位中的M1M0应取10。可设定为软件启动定时器，故GATE取0。因用计数功能，C/T取1。定时器方式寄存器TMOD高4位为无关位，一般都取0，所以TMOD应为06H。计数初值为

$$X = 2^8 - \text{计数值} = 2^8 - 10 = 256 - 10 = 246 = \text{F6H}$$

参考程序：

```
                        ORG   0000H
0000H   02 00 50        LJMP  MAIN
                        ORG   000BH         ; 定时器0的中断入口地址
000BH   02 02 00        LJMP  SER0          ; 转向中断服务程序
                        ORG   0050H
0050H   78 00     MAIN: MOV   R0, #00H
0052H   75 89 06        MOV   TMOD, #06H    ; 置计数器0为工作方式2
0055H   75 8C F6        MOV   TH0, #0F6H    ; 置10次计数初值
0058H   75 8A F6        MOV   TL0, #0F6H
005BH   D2 AF           SETB  EA
005DH   D2 A9           SETB  ET0
005FH   D2 8C           SETB  TR0
0061H   80 FE           SJMP  $
                        ORG   0200H
```

```
0200H    08          SER0:INC   R0          ；中断服务程序
0201H    32               RETI
                          END
```

5.5.5　方式 3

1. 电路逻辑结构

在方式 3 下，定时器 0 分为两个独立的 8 位加 1 计数器 TH0 和 TL0。其中 TL0 既可用于定时，也能用于计数；TH0 只能用于定时。定时器/计数器 0 方式 3 逻辑结构如图 5-8 所示。

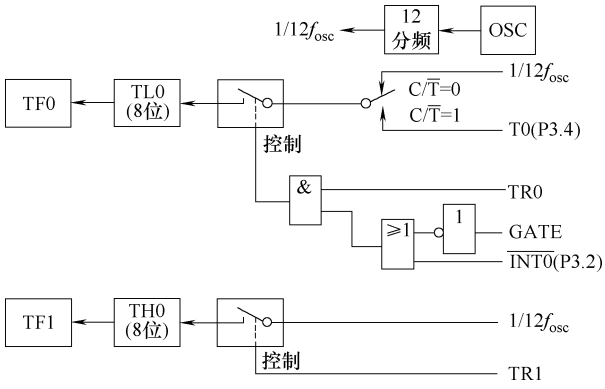


图 5-8　定时器/计数器 0 方式 3 逻辑结构

加 1 计数器 TL0 占用了 T0 除 TH0 外的全部资源，与定时器/计数器 T0 相关的各个控制位和引脚信号均由它使用；与方式 2 相比，只是不能自动将定时初值或计数初值再装入 TL0，而必须用程序来完成。加 1 计数器 TH0 只能用于简单的内部定时功能，它占用了原 T1 的控制位 TR1 和 TF1，同时占用了 T1 中断源。

工作方式 3 下定时器/计数器 0 的另一半是 TH0，只能作简单的定时器使用。而且由于寄存器 TCON 的定时器 0 的控制位已被 TLO 独占，因此，只能借用定时器 1 的控制位 TR1 和 TF1 为其服务。即用计数溢出置位 TF1，而定时的启停则受 TR1 的状态控制。

由于 TLO 既能作定时器使用，也能作计数器使用，而 TH0 只能作定时器使用，所以在工作方式 3 下，定时器/计数器 0 可以分解为 2 个 8 位定时器或 1 个 8 位定时器和 1 个 8 位计数器。

2. 工作方式 3 下的定时器/计数器 1

T1 不能工作在方式 3 下，因为在 T0 工作在方式 3 下时，T1 的控制位 TR1、TF1 和中断源被 T0 占用。T1 可工作在方式 0、方式 1、方式 2 下，但其输出直接送入串行口。设置好 T1 的工作方式，T1 就自动开始计数；若要停止计数，可将 T1 设为方式 3。其使用方法如图 5-9 所示。

如果定时器/计数器 0 已经工作在工作方式 3，则定时器/计数器 1 只能工作在方式 0、方式 1 或方式 2 下，因为它的运行控制位 TR1 及计数溢出标志位 TF1 已被定时器/计数器 0

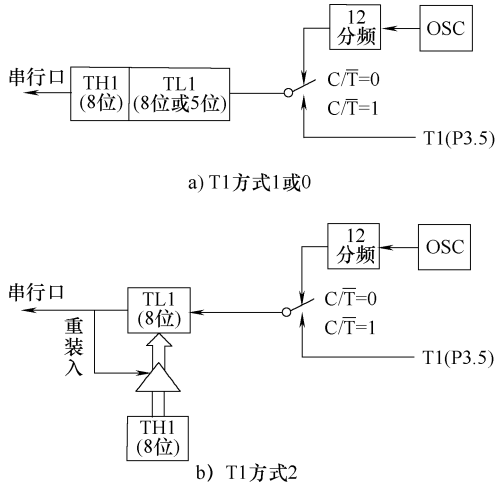


图 5-9 工作方式 3 下的定时器/计数器 1 使用方法

借用。其使用方法如图 5-9 所示。

这时，定时器/计数器 1 通常是作为串行口的波特率发生器使用。因为已没有计数溢出标志位 TF1 可供使用，因此只能把计数溢出直接送到串行口。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。若要停止工作，只需向工作方式选择寄存器 TMOD 送入一个能把它设置为方式 3 的控制字就可以了。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，就会停止工作。

本 章 小 结

本章讨论了 80C51 单片机的定时器/计数器和中断系统的结构、工作原理及其应用。重点是应用。

80C51 单片机具有完备的中断系统，具有 5 个中断源，可设置两个中断优先级。中断的控制与管理通过特殊功能寄存器 TCON、SCON、IE 和 IP 完成。中断处理过程可以分为 3 步：中断响应、中断处理和中断返回。中断控制的实现也要通过编写控制程序完成。

80C51 单片机内部有两个 16 位的可编程定时器/计数器，即 T0 和 T1（8032/8052 有 3 个）。无论是作为定时器实现定时，还是作为计数器实现计数，其实质都是对脉冲信号进行加法计数。只不过定时器是对内部脉冲信号进行计数，而计数器是对外部脉冲信号进行计数。要完成定时或计数控制必须编写初始化程序。编写程序时要完成下列工作：设置 T0 或 T1 的工作方式，为 T0 或 T1 设置定时或计数初始值，启动 T0 或 T1 开始计数，采用查询或中断方式判断是否溢出等。这些工作都是通过软件完成的。

习 题

1. 什么是中断和中断系统？其主要功能是什么？
2. MCS-51 的外部中断有哪两种触发方式？它们对触发脉冲或电平有什么要求？
3. 80C51 有几个中断源？各中断标志是如何产生的？又是如何复位的？CPU 响应各中断时，其中断入

口地址是多少？

4. 某系统有三个外部中断源 1、2、3，当某一中断源变低电平时便要求 CPU 处理，它们的优先处理次序由高到低为 3、2、1，处理程序的入口地址分别为 2000H、2100H、2200H。试编写主程序及中断服务程序（转至相应的入口即可）。

5. 外部中断源有电平触发和边沿触发两种触发方式，这两种触发方式所产生的中断过程有何不同？怎样设定？

6. 80C51 单片机定时器有哪几种工作方式？它们有何区别？

7. 试用 T1 对外部事件计数。要求每计数 100，就将 T1 改成定时方式，控制 P1.7 口输出一个脉宽为 10ms 的正脉冲，然后又转为计数方式，如此反复循环。设晶振频率为 12MHz。

8. 利用 T0 从 P1.0 口输出周期为 1s，脉宽为 20ms 的正脉冲信号，晶振频率为 12MHz。试设计程序。

9. 要求从 P1.1 口输出 1000Hz 方波，晶振频率为 12MHz。试设计程序。

10. 利用 T0 产生定时时钟，由 P1 口控制 8 个指示灯。编一个程序，使 8 个指示灯依次闪动，闪动频率为 1Hz/s。

第 6 章　80C51 单片机的串行通信

【学习目的】

1. 掌握串行通信基础知识。
2. 了解常用的串行通信总线标准。
3. 掌握 MCS-51 单片机的串行通信的基本原理。

串行通信是一种能把二进制数据按位传送的通信，故它所需传输线数量少，特别适用于分级、分层和分布式控制系统以及远距离通信。

6.1　串行通信概述

在通信领域内，有两种数据通信方式：并行通信和串行通信。随着计算机网络化和微机分级分布式应用系统的发展，通信的功能越来越重要。通信是指计算机与外界的信息传输，既包括计算机与计算机之间的传输，也包括计算机与外部设备，如终端、打印机和磁盘等设备之间的传输。

并行通信：一条信息的各位数据被同时传送的通信方式称为并行通信。并行通信的特点：各数据位同时传送，传送速度快、效率高，但有多少数据位就需多少根数据线，因此传送成本高，且只适用于近距离（相距数米）的通信。图 6-1 所示为并行通信方式示意图。

串行通信：一条信息的各位数据被逐位按顺序传送的通信方式称为串行通信。串行通信的特点：数据位传送按位顺序进行，最少只需一根传输线即可完成，成本低但送速度慢。串行通信的距离可以从几米到几千米。图 6-2 所示为串行通信方式示意图。

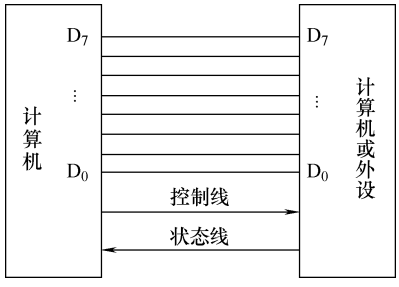


图 6-1　并行通信方式示意图

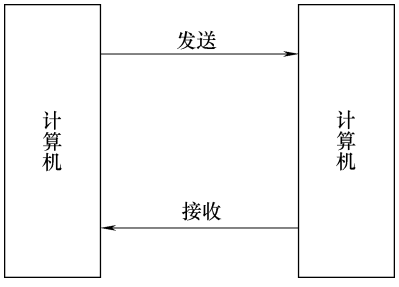


图 6-2　串行通信方式示意图

6.1.1　异步通信和同步通信

串行通信按位顺序进行传输，它又分为异步传输（Asynchronous Transmission）和同步传输（Synchronous Transmission）两种方式，一般称为异步串行通信和同步串行通信。

1. 波特率

波特率的定义为每秒钟传送二进制数码的位数，也称比特数，单位是 bit/s 即位/秒。波特率是串行通信的重要指标，用于表征数据传输的速度。波特率越高，数据传输速度越快，但和字符的实际传输速率不同。字符的实际传输速率是指每秒内所传字符帧的帧数，和字符帧格式有关。例如，一个字符帧包含 11 个字符，则字符的实际传输速率为（1200/11）帧/s = 109. 09 帧/s。

每位的传输时间定义为波特率的倒数。例如，波特率为 1200bit/s 的通信系统，其每位的传输时间应为

$$T_d = \left(\frac{1}{1200}\right)\text{s} = 0.833\text{ms}$$

波特率还和信道的频带有关。波特率越高，信道频带越宽，因此波特率也是衡量通道频宽的重要指标。通常，异步通信的波特率在 50 ~ 9600bit/s 之间。波特率不同于发送时钟和接收时钟，它通常是时钟频率的 1/16 或 1/64。

2. 异步串行通信

在异步通信中，数据通常是以字符（或字节）为单位组成字符帧传送的。字符帧由发送端一帧一帧地发送，通过传输线被接收设备一帧一帧地接收。发送端和接收端可以有各自的时钟来控制数据的发送和接收，这两个时钟源彼此独立、互不同步。

在异步通信中，接收端是依靠字符帧格式来判断发送端是何时开始发送及何时结束发送。平时，发送线为高电平（逻辑“1”），每当接收端检测到传输线上发送过来的低电平逻辑“0”（字符帧中起始位）时，就知道发送端已开始发送，每当接收端接收到字符帧中的停止位时，就知道一帧字符信息已发送完毕。图 6-3 所示为异步串行通信的帧格式。

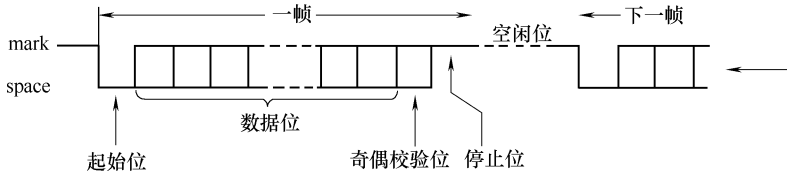


图 6-3　异步串行通信的帧格式

异步串行通信的帧格式说明：

字符帧也叫数据帧，由起始位、数据位、奇偶校验位和停止位四部分组成。各部分结构和功能分述如下。

① 起始位：位于字符帧开头，只占 1 位，始终为逻辑“0”（低电平），用于向接收设备表示发送端开始发送一帧信息。

② 数据位：紧跟起始位之后，用户根据情况可取 5 位、6 位、7 位或 8 位，低位在前（左），高位在后（右）。若所传数据为 ASCII 码，则常取 7 位。

③ 奇偶校验位：位于数据位后，仅占一位，用来表征串行通信中采用奇校验还是偶校验，由用户根据需要决定。

④ 停止位：位于字符帧末尾，为逻辑“1”（高电平），通常可取 1 位、1. 5 位或 2 位，用于向接收端表示一帧字符信息已发送完毕，也为发送下一帧字符作准备。

⑤ 位时间：一个格式位的时间宽度。

⑥ 帧（Frame）：从起始位开始到停止位结束的全部内容称为一帧，帧是一个字符的完整通信格式。

异步串行通信是一帧接一帧进行的，传输可以是连续的，也可以是断续（间歇）的。连续的异步串行通信，是在一个字符格式的停止位之后立即发送下一个字符的起始位，开始一个新的字符传输，即帧与帧之间是连续的。而断续的异步串行通信，则是在一帧结束之后并不接着传输下一个字符，不传输时维持数据线的“1”（高电平）状态，使数据线处于空闲；其后，新的字符传输可以在任何时刻开始，并不要求整数倍的位时间。

3. 同步串行通信

异步通信以字符为单位，为实现发送和接收双方的协调，需要有开始和结束标志，为此每个字符的帧格式中都要包含起始位和停止位。传送过程中，起始位和停止位的不断重复将会占用大量的通信时间。为提高传送速度，把数据传输按相等的时间间隔分块进行，在数据块的开始加一些特殊字符，作为发送和接收双方的同步标志。由于数据块的位数较多，为防止错位，在发送数据时一般同时给出时钟信号，以保持接收与发送的同步，这就是同步串行通信。同步串行通信的数据传送格式如图 6-4 所示。

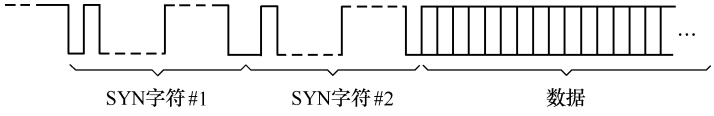


图 6-4　同步串行通信的数据传送格式

与异步串行通信比较，同步串行通信的数据传输效率高，但其通信双方对同步的要求也高，因此，同步串行通信的数据格式有如下特点和要求：

① 在数据块传输的开始使用同步字符串，作为发送和接收双方同步的标志，而在结束时不需要同步标志。

② 数据字符之间不允许有间隔，当线路空闲或没有数据可发时，可发送同步字符串。

③ 数据块内各字符的格式必须相同。

显然，同步串行通信比异步串行通信的传送速度快，但同步串行通信要求收发双方在整个数据传输过程中始终保持同步，这将对硬件提出更高的要求，实现起来难度大一些；而异步串行通信只要求在每帧的短时间内保持同步即可，实现起来容易得多。所以同步串行通信适用于数据量大、对速度要求比较高的串行通信场合。

6.1.2　串行通信的数据传送方式

在串行通信中，数据是在两个站之间传送的。按照数据传送方向，串行通信可分为单工、半双工和全双工三种传送方式，如图 6-5 所示。

1. 单工方式

通信线的一端接发送器，另一端接接收器，它们形成单向连接，只允许数据按照一个固定的方向传送。单工形式的串行通信只需要一条数据线。

2. 半双工方式

系统中的每个通信设备都由一个发送器和一个接收器组成，通过收发开关接到通信线

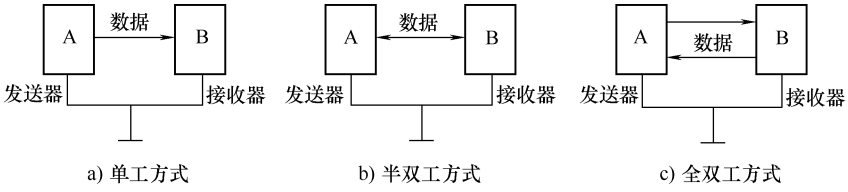


图 6-5　串行通信的数据传送方式

上。数据能够实现双方向传送，但任何时刻只能由其中的一方发送数据，另一方接收数据。因此半双工形式既可以使用一条数据线，也可以使用两条数据线。其收发开关并不是实际的物理开关，而是由软件控制的电子开关，通信线两端通过半双工协议进行功能切换。

3. 全双工方式

系统的每端都含有发送器和接收器，数据可以同时在两个方向上传送。因此全双工形式的串行通信需要两条数据线。

尽管许多串行通信接口电路具有全双工功能，但在实际应用中，大多数情况下只工作于半双工方式，即两个工作站通常并不同时收发。这种用法并无害处，虽然没有充分发挥效率，但简单、实用。

6.2　80C51 串行通信

为了实现串行通信，需要有硬件电路以解决串行数据传输中的一系列协调问题，这些硬件就是串行接口电路或简称串行口。

6.2.1　80C51 单片机串行口结构

串行口主要由发送寄存器、接收寄存器和移位寄存器组成。通常把实现异步通信的串行口称为通用异步接收器/发送器（Universal Asynchronous Receiver/Transmitter，UART），把实现同步通信的串行口称为通用同步接收器/发送器（Universal Synchronous Receive/Transmitter，USRT），而把实现同步和异步通信的串行口称为通用同步异步接收/发送器（Universal Synchronous Asynchronous Receiver/ Transmitter，USART）。

80C51 的串行口，虽然是既能实现同步通信，又能实现异步通信的全双工串行口，但在单片机的串行数据通信中，最常用的是异步方式，因此常把它写为 UART。它的寄存器结构如图 6-6 所示。

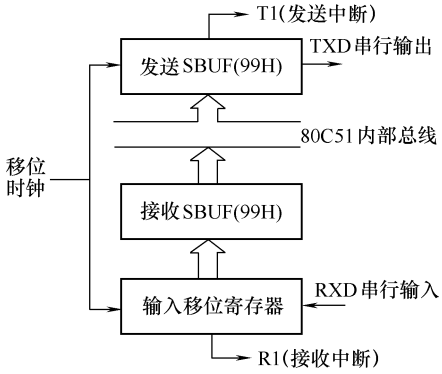


图 6-6　80C51 的串行口寄存器结构

发送寄存器（发送器）SBUF（TX）为 8 位只写寄存器，地址为 99H，写入该寄存器的数据将从 TXD 引脚发送出去。接收寄存器（接收器）SBUF（RX）用于在接收状态下存放从 RXD 引脚接收到的数据，只供 CPU 读取。接收寄存器为 8 位寄存器，地址也是 99H。

鉴于任何时刻 CPU 只能执行发送或接收指令中的一种，因此这两个寄存器共用一个地址，并且在程序

中可以不必区分 SBUF（TX）和 SBUF（RX），统一使用 SBUF 表示即可。在通信过程中，一旦 SBUF（TX）变空或 SBUF（RX）变满，便向 CPU 发出中断请求。串行口有两个相互独立的数据引脚，即发送引脚 TXD 和接收引脚 RXD，分别供数据发送和接收使用。但这两个引脚都是与 I/O 口线复用的，TXD 与 P3.1 共用引脚，RXD 与 P3.0 共用引脚。

串行口的主要功能是实现数据的串行化/反串行化，串行化是把并行数据转变为串行数据，而反串行化则是把串行数据转变为并行数据。串行口的数据发送是一个串行化过程，在这一过程中，把写入发送寄存器的并行数据，按帧格式要求插入格式信息（起始位、奇偶位和停止位），构成一个串行位串，经 TXD 引脚串行送出。

串行口的数据接收是一个反串行化过程。在反串行化过程中，串行数据通过引脚 RXD 进入，经移位寄存器移位把帧中的格式信息滤除而保留数据位，从而在接收缓冲器中得到并行数据，并送上内部数据总线。在串行接收通路中，移位寄存器和接收缓冲器构成了双缓冲结构，以避免在数据接收过程中出现帧重叠错误。所谓帧重叠错误就是在接收下一帧数据时，前一帧的数据还没有被读走。

6.2.2 80C51 单片机串行口控制机制

80C51 串行口通过控制寄存器、中断功能和波特率设置实现串行通信控制，此处先介绍前两项内容。

1. 串行口控制寄存器 SCON

SCON 用来控制串行口的工作方式和状态，它可以是位寻址。在复位时所有位被清零，单元地址为 98H。

其格式见表 6-1。

表 6-1 串行口控制寄存器 SCON 的格式

位地址	9F	9E	9D	9C	9B	9A	99	98
位符号	SM0	SM1	SM2	REN	TB8	RB8	TI	RI

各位定义如下：

- SM0、SM1：串行口工作方式选择位（见表 6-2）。

表 6-2 SM0、SM1 串行口工作方式选择位

SM0 SM1	工作方式	功能说明	波特率
0 0	方式 0	8 位同步移位寄存器	$f_{osc}/12$
0 1	方式 1	10 位 UART	由定时器控制
1 0	方式 2	11 位 UART	$f_{osc}/32$ 或 $f_{osc}/64$
1 1	方式 3	11 位 UART	由定时器控制

- SM2：多机通信控制位。

因为多机通信是在方式 2 和方式 3 下进行的，因此 SM2 主要用于方式 2 和方式 3。

➢ SM2 = 1，则只有在接收到的第 9 位数据（RB8）为 1 时，才将接收到的前 8 位数据送

入 SBUF，并置位 RI 产生中断请求；否则将接收到的前 8 位数据丢弃。

➢ SM2 = 0，则不论第 9 位数据是 0 还是 1，都将前 8 位数据装入 SBUF 中，并产生中断请求。在方式 0 时，SM2 必须为 0。

● REN：允许接收控制位。该位由软件置位或复位。

➢ REN = 0 时，禁止串行口接收。

➢ REN = 1 时，允许串行口接收。

● TB8：发送数据位。

➢ 在方式 2 或方式 3 时，TB8 是发送数据的第 9 位，根据发送数据的需要由软件置位或复位。

➢ 可作为奇偶校验位（单机通信）。

➢ 可在多机通信中作为发送地址帧或数据帧的标志位。多机通信时，一般约定：发送地址帧时，设置 TB8 = 1；发送数据帧时，设置 TB8 = 0。在方式 0 和方式 1 中，该位未用。

● RB8：接收数据位。

➢ 在方式 2 和方式 3 时，存放接收数据的第 9 位。

➢ 可以是约定的奇偶校验位。

➢ 可以是约定的地址/数据标志位，可根据 RB8 被置位的情况对接收到的数据进行某种判断。在多机通信时，若 RB8 = 1，说明收到的数据为地址帧；RB8 = 0，说明收到的数据为数据帧。在方式 1 下，若 SM2 = 0，则 RB8 用于存放接收到的停止位方式。在方式 0 下，该位未用。

● TI：发送中断标志位，用于指示一帧数据发送完否。

➢ 在方式 0 下，发送电路发送完第 8 位数据时，TI 由硬件置位。

➢ 在其他方式下，TI 在发送电路开始发送停止位时置位。这就是说，TI 在发送前必须由软件复位，发送完一帧后由硬件置位。因此，CPU 查询 TI 状态便可知一帧信息是否已发送完毕。

● RI：接收中断标志位，用于指示一帧信息是否接收完。

➢ 在方式 1 下，RI 在接收电路接收到第 8 位数据时由硬件置位。

➢ 在其他方式下，RI 是在接收电路接收到停止位的中间位置时置位的，RI 也可供 CPU 查询，以决定 CPU 是否需要从“SBUF（接收）”中提取接收到的字符或数据。RI 也由软件复位。

在进行串行通信时，一帧发送完后，必须用软件来设置 SCON 的内容。当由指令改变 SCON 的内容时，改变的内容在下一条指令的第一个周期的 S1P1 状态期间才锁存到 SCON 寄存器中，并开始有效。如果此时已开始进行串行发送，那么 TB8 送出去的仍是原有的值而不是新值。

在进行串行通信时，一帧发送完毕后，发送中断标志置位，向 CPU 请求中断；当一帧接收完毕时，接收中断标志置位，也向 CPU 请求中断。若 CPU 允许中断，则要进入中断服务程序。CPU 事先并不能区分是 RI 请求中断还是 TI 请求中断，只能在进入中断服务程序后，通过查询来区分，然后进入相应的中断处理。

2. 电源控制寄存器 PCON

PCON 主要是为 CHMOS 型单片机的电源控制设置的专用寄存器，单元地址为 97H，不

能位寻址。PCON 的格式见表 6-3。

表 6-3 电源控制寄存器 PCON 的格式

位地址	D7	D6	D5	D4	D3	D2	D1	D0
位符号	SMOD	/	/	/	GF1	GF0	PD	IDL

各位定义如下：

- SMOD：串行口波特率的倍增位。

在 HMOS 单片机中，该寄存器中除最高位之外，其他位都是虚设的。在单片机工作在方式 1、方式 2 和方式 3 时，有：

➢ SMOD = 1，串行口波特率提高一倍。

➢ SMOD = 0，则波特率不加倍。系统复位时 SMOD = 0。

- GF1、GF0：通用标志位，由软件置位、复位。
- PD：掉电方式控制位，PD = 1 则进入掉电方式。
- IDL：待机方式控制位，IDL = 1 则进入待机方式。

6.3 80C51 单片机串行口的工作方式

80C51 串行口可设置四种工作方式，由 SCON 中的 SM0、SM1 进行定义。

6.3.1 串行工作方式 0

串行工作方式 0 是把串行口作为同步移位寄存器使用，实现串行数据的输入/输出。移位数据的传输以 8 位为一组，低位在前、高位在后。

利用串行工作方式 0，加上“并入串出”或“串入并出”芯片的配合，80C51 的串行口可实现数据的并行输入/输出，如图 6-7 所示。

“并入串出”芯片 74165 用于把并行输入数据通过移位形成位串，传送给串行口；而“串入并出”芯片 74164 则接收串行口的串行数据，通过移位形成 8 位并行数据输出。

在方式 0 下，串行数据的发送和接收都使用 RXD（P3.0）引脚，而 TXD（P3.1）引脚则用来为“并入串出”或“串入并出”芯片中的移位寄存器提供移位脉冲，控制数据移位速率。该移位脉冲由串行口提供，直接送到两种移位寄存器的时钟输入端（通常称为 CLK）。

为控制并行数据的输入和移位，“并入串出”芯片要有输入选通和启动移位的信号，即图 6-7 所示的移位/加载信号。先把此信号置为 0 以加载并行数据，随后再变为 1 以启动移位，把数据位直接移向串行口的 RXD 引脚。实际应用中，该信号通常用软件方法产生，通过一条口线（图中为 P*x*.*x*）送出。

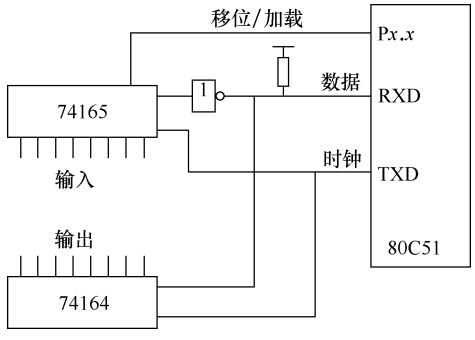


图 6-7 串行口方式 0 实现数据并行输入/输出

方式 0 的帧格式都是纯数据位，不用附加起始位、停止位和校验位，数据移位按低位在

前、高位在后的顺序进行。输入的并行数据在串口 SBUF 中。串行数据接收需要有允许接收的控制，具体由 SCON 寄存器的 REN 位实现。REN = 0，禁止接收；REN = 1，允许接收。当软件置位 REN 时，即开始从 RXD 端输入数据（低位在前）；当接收到 8 位数据时，中断标志 RI 置位；再通过查询或中断方法把移入串口的数据读走，然后就可以对“并入串出”芯片加载新数据了。

要用 80C51 的串行口进行并行数据输出，先把数据字节通过 RXD 引脚串行传送给“串入并出”芯片，并在 TXD 引脚移位脉冲的控制下，把接收的串行数据通过移位得到并行数据，移位停止后 8 位并行数据留在其中，等待被其他设备取走。

如果“串入并出”芯片没有并行输出的选通控制功能，可在数据输出端加三态门，通过选通控制，实现并行数据“齐步”输出。当串行口把 8 位数据全部移出后，SCON 的中断标志 TI 被自动置 1。通知 CPU 以中断或查询的方法把并行数据取走。

工作方式 0 时，移位操作（串入或串出）的波特率是固定的，为单片机晶振频率的 1/12，若晶振频率用 f_{osc} 表示，则波特率为 $f_{osc}/12$。按此波特率的一个机器周期进行一次移位，若 $f_{osc}=6$MHz，则波特率为 500kbit/s，即 2μs 移位一次。若 $f_{osc}=12$MHz，则波特率为 1Mbit/s，即 1μs 移位一次。

6.3.2　串行工作方式 1

串行工作方式 1 是 10 位为 1 帧的异步串行通信方式，这种工作方式是为双机通信而准备的。帧格式包括 1 个起始位、8 个数据位和 1 个停止位。

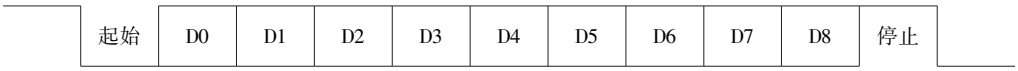
方式 1 的数据发送是由一条写发送寄存器（SBUF）的指令开始，随后在串行口由硬件自动加入起始位和停止位，构成完整的帧格式，然后在移位脉冲的作用下，由 TXD 端串行输出。一个字符帧发送完后，使 TXD 输出线维持在状态 1 下，并将 SCON 寄存器的 TI 位置 1，通知 CPU 可以接着发送下一个字符。

接收数据时，SCON 的 REN 位应处于允许接收状态，即 REN = 1。在此前提下，在串行口采样 RXD 端，当采样到从 1 到 0 的状态跳变时，就认为已接收到起始位。随后在移位脉冲的控制下，把接收到的数据位移入接收寄存器中。直到停止位到来之后置位中断标志位 RI，通知 CPU 从 SBUF 取走接收到的一个字符。

其实，上述有关数据传送过程的说明不但适用于串行工作方式 1，同样也适用于工作方式 2 和工作方式 3。

6.3.3　串行工作方式 2 和 3

串行工作方式 2 和 3 都是 11 位为一帧的串行通信方式，即 1 个起始位、9 个数据位和 1 个停止位。其帧格式为



在这两种工作方式下，字符还是 8 个数据位，只不过增加了一个第 9 数据位（D8），它是一个可编程位，其功能由用户设定。

在发送数据时，应预先在串行口控制寄存器 SCON 的 TB8 位中把第 9 个数据位的内容准

备好。可使用如下指令完成：

```
SETB TB8      ；TB8 位置 1
CLR TB8       ；TB8 位置 0
```

发送数据 D0 ~ D7 由 MOV 指令向 SBUF 写入，而 D8 位的内容则来自 SCON 的 TB8 位，在发送移位过程中插入到 8 位数据之后成为第 9 个数据位。这两种工作方式的数据接收过程也与方式 1 基本类似，不同点仍在第 9 个数据位上，串行口把接收到的前 8 个数据位移入 SBUF，而把第 9 数据位送 SCON 的 RB8。

串行工作方式 2 和 3 是为多机通信而准备的。两者的工作过程相同，差别仅在于波特率的设置，方式 2 的波特率是固定的；而方式 3 的波特率可由用户根据需要设定，设定方法与方式 1 相同。不同的波特率设置可以应对多种通信环境。

6.4 串行通信数据传输速率

传输速率（Transfer Rate）用于说明信息传输的快慢，是串行通信的一项重要技术指标，以单位时间内传输信息的单位数表示。此处讨论的是串行通信中的传输速率。

6.4.1 传输速率的表示方法

1. 传输速率的术语

单片机应用中涉及的有关传输速率的术语有以下几点：

① 波特（Baud）。波特本是一名法国工程师的名字，在通信技术中，每秒 1 次的信号变化称为 1 波特。波特原本是表示电信设备传输速率的单位，后来又用于表示调制解调器的数据传输速率。

② 波特率（Baud Rate）。波特率是每秒钟信号变化的次数。在单片机的串行数据传输中，信号变化都反映在二进制位上，因此就以波特率表示串行数据的传输速率。

③ 比特率（Bit Rate）。比特率也称为位速率，即每秒钟传输二进制数的位数。

在一般的单片机串行通信中，波特率与比特率的概念是一样的，但在高速串行通信中，由于一个事件的编码往往不止 1 位，因此波特率与比特率就不一样了，例如事件按 4 位编码，如果数据传输的波特率是 2400Baud/s，则比特率就是 9600bit/s。

2. 单片机中使用的波特率

单片机使用波特率作为串行通信传送速率的单位。每秒传送 1 个格式位就是 1 波特，即 1 波特 = 1bit/s（位/秒）。

在串行数据传输中，波特率除表明数据传输速率外，还可以表示串行口中移位脉冲频率的高低，因为串行数据发送和接收的速率是由移位脉冲决定的。波特率高表明移位脉冲频率高，串行数据传输速度就快；反之，波特率低表明移位脉冲频率也低，串行数据传输速度就慢。波特率的数值差异很大。例如，在 RS-232C 标准中规定，允许波特率为每秒 50 ~ 19200bit/s。在实际的串行数据传输应用中，应根据速度要求、线路质量及设备情况等因素选定波特率。

6.4.2 80C51 单片机波特率的设置

串行口的通信波特率恰到好处地反映了串行传输数据的速率。通信波特率的选用，不仅

和所选通信设备、传输距离有关，还受传输线状况所制约。用户应根据实际需要加以正确选用。

1. 串行工作方式 0 下的波特率

在串行工作方式 0 下，串行口通信的波特率是固定的，其值为

$$\text{波特率} = f_{osc}/12$$

式中，f_{osc}表示外部振荡器频率。$f_{osc}/12$ 即外部振荡脉冲的 12 分频，而外部振荡脉冲 12 分频产生一个机器周期。因此，在串行工作方式 0 下，每个机器周期产生一个移位脉冲，进行一次串行移位。因为其波特率固定，所以在串行工作方式 0 时，不存在设置波特率的问题。

2. 串行工作方式 2 下的波特率

在串行工作方式 2 下，波特率也是固定的，但有两个数值。其计算公式为

$$\text{波特率} = f_{osc} \times 2^{smod}/64$$

这就是说，若 SMOD = 0，则所选波特率为$f_{osc}/64$，即晶振频率的 1/64；若 SMOD = 1，则波特率为$f_{osc}/32$，即晶振频率的 1/32。

3. 串行工作方式 1 和 3 下的波特率

在串行工作方式 1 和 3 下，波特率不是固定的，可以根据需要设置。具体地说，80C51 是以定时器 T_1 作为波特率发生器，以其溢出脉冲产生串行口的移位脉冲。因此，在这两种工作方式中，通过计算 T_1 的计数初值就可以实现波特率的设置。假定定时器的计数初值为 X，则计数溢出周期为

$$(12/f_{osc}) \times (256 - X)$$

溢出率为溢出周期的倒数，则波特率计算公式为

$$\text{波特率} = (2^{smod}/32) \times (\text{定时器 1 溢出率}) = (2^{smod}/32) \times \{f_{osc}/[12 \times (256 - X)]\}$$

根据上述波特率计算公式，得出计数初值的计算公式为

$$X = 256 - (f_{osc} \times 2^{smod}) / (384 \times \text{波特率})$$

以定时器 T_1 作波特率发生器是由系统决定的，用户只需先把波特率确定下来，再计算出定时器的计数初值，然后通过初始化程序装入 T_1 即可。还要注意的是，当定时器 T_1 作波特率发生器使用时，应选择定时方式 2，因为在方式 2 下定时器具有自动加载功能。

6.5　80C51 单片机串行通信的应用

6.5.1　单片机串行口扩展并行输入输出口的应用

1. 将串行口作为并行输入口使用

串行口在方式 0 下，通过外接一个“并入串出”的 8 位移位寄存器（74LS165 或 CD4014），可以作为并行输入口使用。例如，通过外接 CD4014 将 8 路开关状态从串口读入单片机的 20H 存储单元，要求控制开关 Kc 断开（Kc = 1）时，8051 处于等待状态；Kc 闭合（Kc = 0）时，8051 开始输入。其电路如图 6-8 所示。

CD4014 是一个 8 位串入/并入-串出移位寄存器，CP 为同步移位脉冲输入端，P1 ~ P8 为并行输入端，Q8 为串行输出端。P/$\overline{S}$为控制端，控制规则如下：

若 P/$\overline{S}$ = 0，则 CD4014 为串行输入（该输入端图中未画出）；

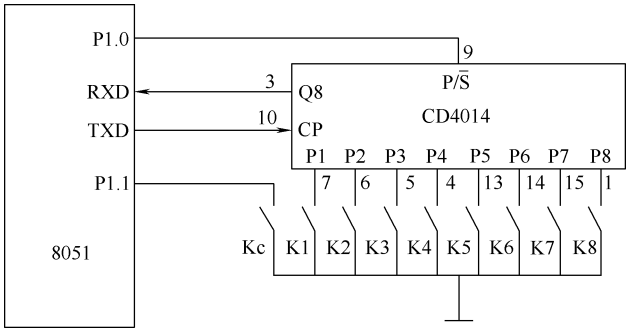


图 6-8 将串行口作为并行输入口使用

若 $P/\overline{S}=1$，则 CD4014 为并行输入。开关 Kc 用于提供控制信号，当 Kc 闭合时，表示要求单片机读入开关量。

只要在程序中对 P1.1 引脚进行查询，发现 P1.1 =0（即开关 Kc 闭合），便通过 P1.0 使 CD4014 的 $P/\overline{S}=1$，然后再启动单片机串口方式 0 接收过程，即可将 CD4014 并行输入的开关状态通过串口输入到单片机中。

应用程序：

```
        ORG 0200H
        CLR ES              ；关串口中断，使用查询方式
START:  JB P1.1，$          ；若 Kc 未闭合，则等待
        SETB P1.0           ；令 CD4014 并行输入开关量
        NOP
        NOP
        CLR P1.0            ；令 CD4014 停止并行输入，准备串行输出
        MOV SCON，#10H      ；设置串口为方式 0
        JNB RI，$           ；若未接收完，则等待
        CLR RI              ；接收完。清 RI
        MOV A，SBUF         ；将开关量读入单片机的 A 中
        MOV 20H，A          ；将开关量存入单片机 20H 单元
        ……                  ；转入其他程序
        SJMP START          ；准备下一次读入开关量
        END
```

2. 将串行口作为并行输出口使用

在串行工作方式 0 下，串行口通过外接一个“串入并出”的 8 位移位寄存器 74LS164（或 CD4094），可以作为并行输出口使用。

例如，使用 CD4094 的并行输出端接 8 支发光二极管（LED），利用它的串入并出功能，

把 LED 从左向右依此点亮，并反复循环之。假设 LED 为共阴极型。

硬件电路：

串行口与 LED 的连接电路如图 6-9 所示。

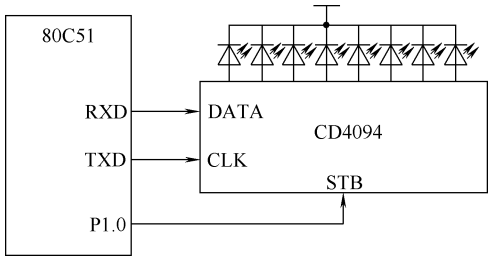


图 6-9　串行口与 LED 的连接电路

在图 6-9 中，8051 单片机串行口工作于方式 0。CD4094 是一种 8 位串行输入、并行输出的同步移位寄存器，CLK 为同步脉冲输入端。STB 为控制端，控制规则如下：

若 STB = 0，则 8 位并行数据输出端关闭，但串行数据从 DATA 输入；若 STB = 1，则 DATA 输入端关闭，8 位数据并行输出。

应用程序：

当串行口把 8 位状态码串行移位输出后，TI 置 1，如把 TI 作状态查询标志，则可使用查询法进行程序设计。

```
        ORG     0200H
        MOV     SCON, #00H  ; 串行口方式 0 工作
        CLR     ES
        MOV     A, #80H     ; 发光二极管从左亮起
DELR:   CLR     P1.0        ; 关闭（关）并行输出
        MOV     SBUF, A     ; 串行输出
        JNB     TI, $       ; 状态查询
        SETB    P1.0        ; 开启并行输出
        ACALL   DELAY       ; 状态维持（DELAY 延时子程序）
        CLR     TI          ; 清发送中断标志
        RR      A           ; 发光右移
        AJMP    DELR        ; 继续
        END
```

6.5.2　单片机与单片机之间的通信

有两个单片机子系统，它们能独立地完成主系统的某一个功能，且两个子系统具有一定的信息交换需求，这时就可以利用串行通信的方式将两个子系统连接起来。

1. 硬件连接

两个单片机子系统如果在同一个电路板或处于同一个机箱内，这时将两个单片机的TXD和RXD引出线相连就可以，如图6-10所示。当进行通信的两台单片机距离较远（5～15m）时，两台单片机之间则不宜直接连接。此时，通常采用RS-232C接口进行点对点的通信连接，如图6-11所示。

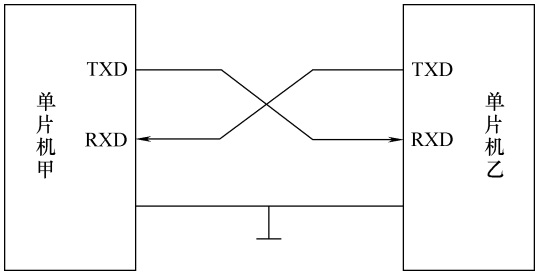


图6-10 短距离双机通信的硬件连接

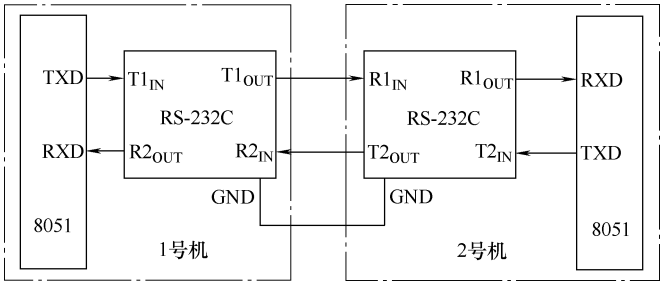


图6-11 较远距离（5～15m）双机通信的硬件连接

2. 通信协议

1号机是发送方，2号机是接收方。1号机发送时，先发送一个“E1”联络信号，2号机收到后回答一个“E2”应答信号，表示同意接收。1号机收到应答信号“E2”后，开始发送数据，每发送一个数据字节都要计算“校验和”。假定数据块长度为16B，起始地址为40H，一个数据块发送完毕后立即发送“校验和”。2号机接收数据并转存到数据缓冲区，起始地址也为40H，每接收到一个数据字节便计算一次“校验和”。当接收完一个数据块后，再接收1号机发来的“校验和”，并将它与本机求出的校验和进行比较。

若校验和进行比较时，两者相等，说明接收正确，2号机回答00H；若两者不相等，说明接收不正确，2号机回答FFH，请求重发。1号机接到00H后结束发送。若收到的答复不是00H，则重新发送数据一次。双方约定采用串行工作方式1进行通信，波特率为2400Baud/s。T1工作在定时器方式2，晶体振荡器频率选用11.0592MHz，PCON寄存器的SMOD位为0，通过计算或查表，可得定时器T1的初值为（TH1）=（TL1）=0F4H。

3. 应用程序

```
ASTART：CLR   ES    ；关串口中断，使用查询方式控制收发
        MOV   TMOD，#20H    ；置定时器T1为方式2
```

```
        MOV   TH1, #0F4H    ; 装载定时初值, 波特率为 2400Baud/s
        MOV   TL1, #0F4H
        MOV   PCON, #00H    ; 置 SMOD = 0
        SETB  TR1   ; 启动定时器 T1
        MOV   SCON, #50H    ; 置串口为方式 1, 允许接收, 清 TI、RI
ALOOP1: MOV   SBUF, #0E1H   ; 发联络信号
        JNB   TI, $     ; 等待一帧发完
        CLR   TI    ; 发完清 TI, 允许再发
        JNB   RI, $     ; 等待 2 号机的应答信号
        CLR   RI    ; 收到应答, 清 RI, 允许再接收
        MOV   A, SBUF   ; 读 2 号机应答信号至 A
        XRL   A, #0E2H  ; 判断 2 号机是否准备完毕
        JNZ   ALOOP1 ; 2 号机未准备好, 继续联络
ALOOP2: MOV   R0, #40H; 2 号机准备好, 设定数据块指针初值
ALOOP2: MOV   R0, #40H; 2 号机准备好, 设定数据块指针初值
        MOV   R7, #10H  ; 设定数据块长度
        MOV   R6, #00H  ; 校验和单元清 0
ALOOP3: MOV   SBUF, @R0  ; 发送一个字节数据
        MOV   A, R6     ; 取出校验和
        ADD   A, @R0    ; 求校验和
        MOV   R6, A     ; 保存校验和
        INC   R0    ; 数据块指针加 1
        JNB   TI, $     ; 等待一帧发完
        CLR   TI    ; 一帧发完, 清 TI, 允许再发
        DJNZ  R7, ALOOP3    ; 整个数据块是否发送完毕
        MOV   SBUF, R6   ; 发送校验和
        JNB   TI, $
        CLR   TI    ; 发完, 清 TI
        JNB   RI, $     ; 等待 2 号机的应答信号
        CLR   RI    ; 收到应答, 清 TI
        MOV   A, SBUF   ; 读 2 号机应答信号至 A
        JNZ   ALOOP2    ; 2 号机应答“错误”, 转重新发送
        RET     ; 2 号机应答“正确”, 返回主程序
```

2 号机源程序:

```
BSTART: CLR   ES   ; 关串口中断, 使用查询方式控制收发
        MOV   TMOD, #20H   ; 置定时器 T1 为方式 2
        MOV   TH1, #0F4H   ; 装载定时初值, 波特率为 2400Baud/s
        MOV   TL1, #0F4H
        MOV   PCON, #00H   ; 置 SMOD = 0
```

```
        SETB   TR1      ；启动定时器 T1
        MOV    SCON, #50H   ；置串口为方式 1，允许接收，清 T1、RI
BLOOP1：JNB    RI, $       ；等待 1 号机的联络信号
        CLR    RI       ；收到 1 号机联络信号，清 RI
        MOV    A, SBUF     ；读 1 号机联络信号到 A
        XRL    A, #0E1H    ；判断是否为 1 号机联络信号
        JNZ    BLOOP1      ；不是 1 号机联络信号，再等待
        MOV    SBUF, #0E2H    ；是 1 号机联络信号，发应答信号
BLOOP2：JNB     TI, $
        CLR    TI       ；发完，清 TI
        MOV    R0, #40H    ；准备接收数据，设定数据块指针初值
        MOV    R7, #10H    ；设定数据块长度
        MOV    R6, #00H    ；清校验和单元
BLOOP3：JNB     RI, $       ；等待接收数据
        CLR    RI       ；收到清 RI
        MOV    A, SBUF     ；读入接收数据至 A
        MOV    @R0, A      ；保存接收数据
        INC    R0       ；数据指针加 1
        ADD    A, R6       ；求校验和
        MOV    R6, A       ；保存校验和
        DJNZ    R7, BLOOP3     ；判数据块是否接收完毕
        JNB    RI, $       ；完毕，接收 1 号机发来的校验和
        CLR    RI       ；收到，清 R2
        MOV    A, SBUF     ；读入校验和至 A
        XRL    A, R6       ；比较校验和
        JZ   DONE       ；校验和相等，发“正确”标志
        MOV    SBUF, #0FFH    ；校验和不相等，发“错误”标志
        SJMP    BLOOP2    ；重新接收
DONE：MOV    SBUF, #00H    ；发“正确”标志
        RET         ；返回主程序
```

本 章 小 结

MCS-51 系列单片机内部有一个全双工的异步串行通信 I/O 口，该串行口的波特率和帧格式可以编程设定。MCS-51 串行口有四种工作方式：方式 0、1、2、3。帧格式有 10 位、11 位。方式 0 和 2 的传送波特率是固定的；方式 1 和 3 的传送波特率是可变的，由定时器的溢出率决定。

单片机串行口可以扩展并行输入/输出口的应用，单片机与单片机之间以及单片机与 PC 之间都可以进行通信，本章节只介绍了单片机扩展并行输入/输出口的应用和单片机与单片

机之间的通信。

习　题

1. 串行数据传送与并行数据传送相比的主要优点和用途是什么？

2. 简述 MCS-51 单片机串行口四种工作方式的接收和发送数据的过程。

3. 串行口各工作方式的波特率如何确定？

4. 若晶体振荡器频率为 11.0592MHz，串行口工作于方式 1，波特率为 4800bit/s，写出用 T1 作为波特率发生器的方式控制字和计数初值。

5. 利用单片机串行口扩展并行输入接口电路如图 6-12 所示。试编写程序完成将 SW1 ~ SW8 的状态反应在 P0 口所接的 LED 上（如 SW1 闭合时 L7 应点亮）。

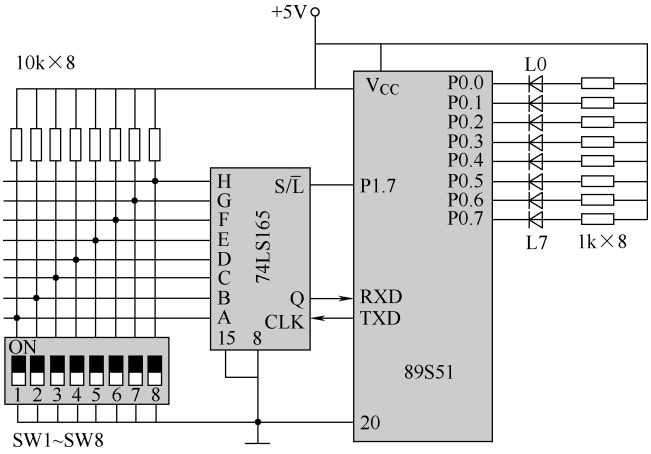


图 6-12　单片机串行口扩展并行输入接口电路

第 7 章　并行扩展技术

【学习目的】

1. 了解数据总线、地址总线和控制总线的构成。
2. 掌握片外扩展程序存储器和数据存储器的方法。
3. 掌握扩展 I/O 的方法和可编程芯片 8255 及 8155 的应用。

7.1　扩展概述

系统扩展是指单片机内部各功能部件不能满足应用系统要求时，在片外连接相应的外围芯片以满足应用系统要求。80C51 系列单片机有很强的外部扩展能力，扩展电路及扩展方法较典型、规范。一个单片机应用系统是以单片机作为核心部件的，但其硬件资源还远不能满足实际需求。通常还需要进行一些必要的扩展。包括以下几项：

① 扩展程序存储器，以存放较大控制程序和数据表格等；

② 扩展数据存储器，以解决大量数据的存储问题；

③ 扩展 I/O 端口，以解决单片机对外 I/O 端口线复用问题；

④ 扩展键盘、显示器和打印机等，以解决数据输入、输出和人机交互信息等接口问题。

一个实际的单片机应用系统往往具有图 7-1 所示的一般结构，需要根据实际情况进行系统扩展。

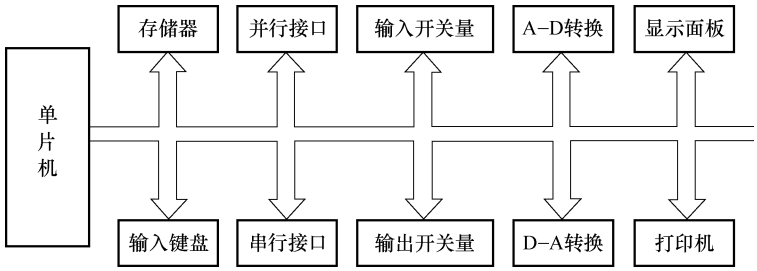


图 7-1　单片机系统扩展一般结构

对于单片机系统扩展的方法有并行扩展法和串行扩展法两种。并行扩展法是指利用单片机本身具备的三组总线（AB、DB、CB）进行的系统扩展，早几年构成单片机应用系统的扩展方法基本上都是并行的三总线扩展。近几年，由于集成电路设计、工艺和结构的发展，串行扩展法得到了很快发展，它利用 SPI 三线总线和 I^2C 双线总线进行串行系统扩展。本教材只对并行扩展技术加以介绍。

7.1.1　单片机并行扩展总线

单片机系统扩展是以单片机芯片为核心进行的，存储器扩展中包括程序存储器和数据存储器，其余所有扩展内容统称为 I/O 扩展。单片机并行扩展系统结构如图 7-2 所示。

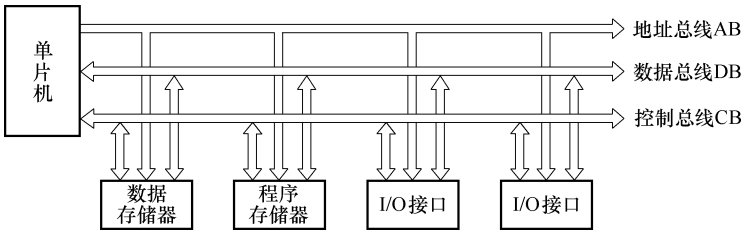


图 7-2　单片机并行扩展系统结构

由扩展系统结构图可知，扩展是通过系统总线进行的。所谓总线就是连接单片机各扩展部件的一组公共信号线，是系统共享的通路，通过总线把各扩展部件连接起来，以进行数据、地址和控制信号的传送。

1. 并行扩展总线的组成

所谓总线，就是单片机连接扩展器件的一组公共信号线，按其功能通常把这些总线分为三组：地址总线、数据总线和控制总线。每组总线由若干条导线组成，具体数目根据功能决定，一般地址总线的数量最多，数据总线固定为 8 根。

（1）地址总线（Address Bus，AB）

地址总线用于传送单片机发出的地址信号，以便对号入座地对 ROM、RAM 及 I/O 口进行选择，以选中相应的单元（字节），然后才能对它进行操作。地址总线的传输是单向的，即只能由单片机向外发出地址信号。地址总线数目决定着可以直接访问的存储单元的数目。例如 10 条地址线组成的地址总线，可以访问 1KB 的外部 ROM 和 RAM 存储单元，每增加一条线，可访问空间翻一番。MCS-51 系列单片机最多可以构造 16 条地址线，也就访问 64KB 的存储空间，对于单片机来说，64KB 将是一个很大的数目了。地址总线由单片机 P0 口提供低 8 位地址 A0 ~ A7，P2 口提供高 8 位地址 A8 ~ A15。

（2）数据总线（Data Bus，DB）

数据总线是用于单片机与外部存储器之间或单片机与外部 I/O 口之间进行数据传送的一组信号线，单片机系统数据总线的数目，与单片机字长是一致的，都是 8 位，所以数据总线也就是 8 条。数据总线是双向的，既可以由单片机向外部输出数据，也可以由外部向单片机输入数据。数据总线由 P0 口提供，用 D0 ~ D7 表示。

（3）控制总线（Control Bus，CB）

控制总线是单片机发出的一组控制命令信号线，是单片机决定对外部器件做什么操作的命令线。一般说来，控制总线是单向的，是单片机向外部发出的。控制总线包括片外系统扩展用控制线和片外信号对单片机的控制线。系统扩展用控制线有 ALE、$\overline{\text{PSEN}}$、$\overline{\text{EA}}$、$\overline{\text{WR}}$、$\overline{\text{RD}}$。

2. 80C51 单片机并行扩展总线

虽然系统扩展需要地址总线和数据总线，但在单片机芯片上并没有为此提供专用的地址

总线引脚和数据总线引脚，实际扩展时都是用 I/O 口线来充当地址线和数据线。80C51 单片机并行扩展总线的构成如图 7-3 所示。

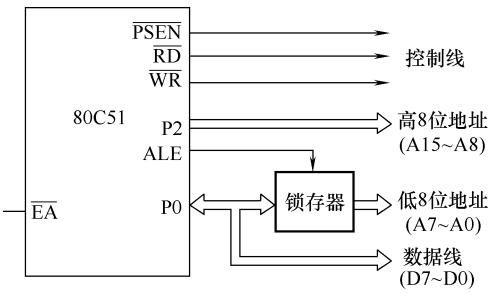


图 7-3　80C51 单片机并行扩展总线的构成

（1）以 P0 口的 8 位口线充当低位地址线/数据线

低位地址线是指低 8 位地址 A7 ~ A0，而数据线为 D7 ~ D0。在第 2 章的内容中，P0 口作为低 8 位地址/数据的复用总线使用，可以提供地址、数据线两种作用，既传送地址又传送数据，所以要采用分时技术对它上面的地址和数据进行分离。

使用分时技术被分离出的是低 8 位地址。因为 CPU 对扩展系统的操作总是先送出地址，然后再进行数据读/写操作，所以应把首先出现的地址分离出来，以便腾出总线供其后的数据传送使用。为保存分离出的地址，需另外增加一个 8 位锁存器，并以 ALE 作为锁存控制信号。因为从时序上看，在 CPU 送出地址时，ALE 信号正好有效。为了与 ALE 信号相适应，应选择高电平或下降沿选通的锁存器，如 74LS373 等。

低 8 位地址进入锁存器，经另一途径提供给扩展系统。在其后的时间里，P0 口线即作为数据线使用，进行数据传送。数据总线用 D0 ~ D7 表示。数据总线是并连到多个连接的外围芯片的数据线上，而在同一时间里只能够有一个是有效的数据传送通道。哪个芯片的数据通道有效，则由地址线控制各个芯片的片选线来选择。

（2）以 P2 口的口线做高位地址线

同样在第 2 章的内容中，P2 口除了作为数据输入/输出口来用，还可以与 P0 口的第二功能配合使用，用于输出片外存储器的高 8 位地址来用，以构成片外存储器的 16 位地址，使单片机外扩展的寻址范围达到 64KB 单元。在实际应用中，高位地址线是根据需要从 P2 口中引出，需要用几位就引出几条。若外扩展容量小于 256 个单元，则不需要高位地址线。

（3）控制总线

控制总线包括片外系统扩展用控制线和片外信号对单片机的控制线。系统扩展用控制线有 ALE、$\overline{\text{PSEN}}$、$\overline{\text{EA}}$、$\overline{\text{WR}}$、$\overline{\text{RD}}$。

ALE：输出 P0 口上地址与数据隔离信号，用于锁存 P0 口输出的低 8 位地址的控制线。通常，ALE 在 P0 口输出地址期间出现低电平，用这个低电平信号的上升沿控制锁存器来锁存地址数据。

$\overline{\text{PSEN}}$：输出，用于读片外程序存储器（EPROM）中的数据。“读”取 EPROM 中数据（指令）时，不能用$\overline{\text{RD}}$信号，而只用$\overline{\text{PSEN}}$信号。

EA：输入，用于选择片内或片外程序存储器。当$\overline{\text{EA}}$ = 0 时，只访问外部程序存储器。当$\overline{\text{EA}}$ = 1 时，先访问内部程序存储器，内部程序存储器全部访问完之后，再访问外部程序存储器。

$\overline{\text{WR}}$、$\overline{\text{RD}}$：输出，用于片外数据存储器（RAM）的读、写控制。当执行片外数据存储器操作指令 MOVX 时，自动生成$\overline{\text{RD}}$、$\overline{\text{WR}}$控制信号。

可以看出，尽管 80C51 单片机有 4 个并行 I/O 口，共 32 条口线，但由于系统外扩展的

需要，仅剩 P1 口以及 P3 口的部分口线可供数据 I/O 使用。

7.1.2　并行扩展系统的 I/O 编址和芯片选取

把扩展芯片接入单片机系统，数据线和控制信号线的连接比较简单，而地址线的连接则比较复杂，因为地址线的连接涉及 I/O 编址和芯片的选取问题。

1. 单片机外扩展地址空间

单片机的外扩展地址空间，与它的存储器系统有关。80C51 单片机存储器系统与外扩展地址空间结构如图 7-4 所示。

在 80C51 单片机系统中，有两个并行存在且相互独立的存储器系统，即程序存储器系统和数据存储器系统。在程序存储器系统中，包括 4KB 的芯片内程序存储器和 64KB 的外扩展地址空间，其中外扩展地址空间供扩展程序存储器使用。在数据存储器系统中，包括由通用寄存器和专用寄存器等占用的芯片内 256 个 RAM 单元以及 64KB 的外扩展地址空间，其中外扩展地址空间供数据存储器和 I/O 扩展使用。

程序存储器系统和数据存储器系统的外扩展地址空间大小相同，但外扩展程序存储器 ROM 的起始地址与单片机芯片是否有片内程序存储器有关。如果没有片内程序存储器，外扩展 ROM 的地址从 0000H 开始，如果有片内程序存储器，则外扩展 ROM 的地址从 1000H 开始。而外扩展 RAM 的起始地址与单片机芯片内 RAM 单元的存在毫无关系，都是从 0000H 开始。

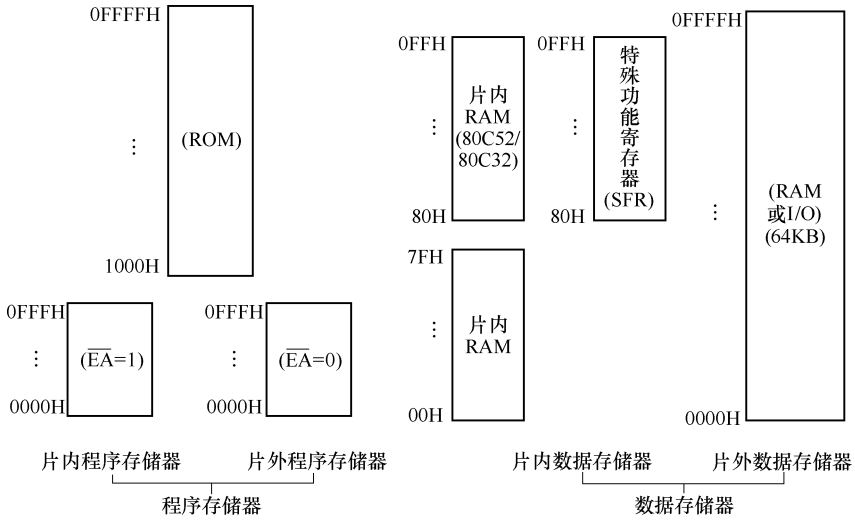


图 7-4　80C51 单片机存储器系统与外扩展地址空间结构

2. 片选技术

进行单片机系统扩展，首先要解决寻址问题，即如何找到要访问的扩展芯片以及芯片内的目标单元。因此，寻址应分芯片选择和芯片内目标单元选择两个层次。由于芯片内单元的选择问题已在各自的芯片内解决，外扩展时只需把扩展芯片的地址引脚与系统地址总线中对应的低位地址线连接起来即可，芯片内自有译码电路完成单元寻址。所以外扩展系统的寻址问题主要集中在芯片的选择上。

为进行芯片选择，扩展芯片上都有一个甚至多个片选信号引脚（常用名为$\overline{CE}$或$\overline{CS}$）。所以寻址问题的主要内容就归结到如何产生有效的片选信号。常用的芯片选择方法（即寻址方法），有线选法和译码法两种。

（1）线选法寻址

所谓线选法，就是直接以位地址信号作为芯片的片选信号。使用时只需把地址线与扩展芯片的片选信号引脚直接连接即可。线选法寻址的最大特点是简单，但只适用于规模较小的单片机系统。假定单片机系统分别扩展了程序存储器芯片 2716、数据存储器芯片 6116、并行接口芯片 8255、键盘/显示器接口芯片 8279 和 D-A 转换芯片 0832，则采用线选法寻址的扩展片选连接示意图如图 7-5 所示。

I/O 口线 P2.7～P2.3（即高位地址线）分别连接到 2716、6116、8255、8279 和 0832 的片选信号引脚。信号为低电平状态时芯片被选中。

（2）译码法寻址（见图 7-6）

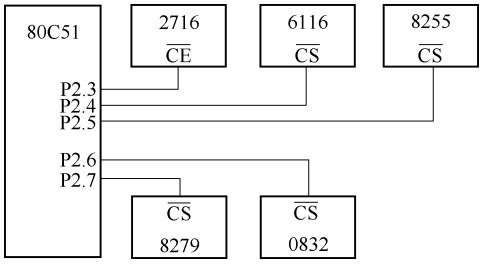


图 7-5　扩展片选连接示意图

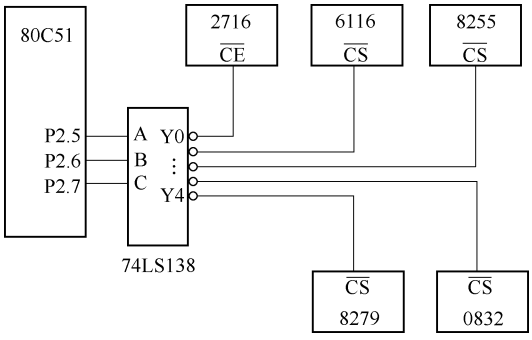


图 7-6　译码法寻址片选示意图

所谓译码法，就是使用译码器对高位地址进行译码，以其译码输出作为扩展芯片的片选信号。这是一种最常用的寻址方法，能有效地利用存储空间，适用于大容量、多芯片的系统扩展。

同样是扩展程序存储器芯片 2716、数据存储器芯片 6116、并行接口芯片 8255、键盘/显示器接口芯片 8279 和 D-A 转换芯片 0832，采用经 74LS138 译码后可产生 8 种状态输出，只需其中的 5 个分别连接 2716、6116、8255、8279 和 0832 的片选信号引脚。可见，译码法能提高系统的寻址能力，但增加了硬件开销。

7.2　存储器扩展技术

7.2.1　存储器的类型

存储器扩展要用到存储器芯片，首先介绍各类存储器。

1. 只读存储器（Read Only Memory，ROM）

程序存储器扩展使用只读存储器。所谓只读，从字面上理解就是只可以从里面读出数据，而不能写进去，ROM 就是单片机用来存放程序的地方。只要让存储器满足一定的条件

就能把数据预先写进去（用指令编写好程序，再将程序编译成机器码 hex 文件，用编程器写入单片机集成电路中）。

根据编程方式的不同，只读存储器有以下五类。

（1）掩膜只读存储器

掩膜只读存储器编程是由半导体制造厂商完成的，即在生产过程中编程。因编程过程是掩膜工艺，因此，称为掩膜 ROM 或 Mask ROM。掩膜 ROM 制造完成后，用户不能更改其内容。这种 ROM 芯片存储结构简单、集成度高，但由于掩膜工艺成本较高，因此只适合于大批量生产。

掩膜 ROM 只供固化软件使用，市场上并没有这种 ROM 芯片出售，存储器扩展也不会涉及它。

（2）可编程只读存储器（PROM）

可编程只读存储器，只能写一次，不能重新擦写，习惯上把带这种程序存储器的单片机称为 OTP 型单片机。存储器容量单位 1KB = 1024B；1MB = 1024KB；1GB = 1024MB。

（3）紫外线擦除的可编程只读存储器（EPROM）

它里面的内容写上去之后，如果觉得不满意，可以用一个特殊的方法去掉后重写，就是用紫外线照射，这种芯片可以擦除的次数也是有限的，约几十次，电脑上的 BIOS 芯片采用的就是这种结构的存储器。

美国 intel 公司的 27 系列产品是 EPROM 的典型芯片，按存储容量不同有多种型号，如 2716（2K×8）、2732（4K×8）、2764（8K×8）、27128（16K×8）、27256（32K×8）等。型号名称后面的数字表示位存储容量。

（4）电擦除可编程只读存储器（EEPROM）

这种存储器可以直接用电擦写，比较方便数据的改写。读/写功能与 RAM 存储器相似，只是写入速度慢一些，但断电后却能保存信息。比较典型的 EEPROM 芯片有 2816、2816A、2817、2817A 和 2864A 等。

（5）闪速存储器（Flash ROM）

这一种快速存储式只读存储器，这种程序存储器的特点是既可以电擦写，而且掉电后程序还能保存，编程寿命可以达到一千次左右，可以反复烧写的。目前新型的单片机都采用这种程序存储器。

2. 读/写存储器

了解了 ROM，再来讲另外一种存储器，叫随机存取存储器（Random Access Memory，RAM），也叫内存。它是一种既可以随时改写，也可以随时读出里面数据的存储器。

按其工作方式，RAM 又分为静态 RAM（SRAM）和动态 RAM（DRAM）两种。静态 RAM 只要电源加电信息就能保存；而动态 RAM 使用的是动态存储单元，需要不断进行刷新以便周期性地再生才能保存信息。动态 RAM 的集成密度高，集成同样的位容量，动态 RAM 所占芯片面积只是静态 RAM 的 1/4；此外动态 RAM 的功耗低，价格便宜。但扩展动态存储器要增加刷新电路，因此适应于大型系统，在单片机系统中使用不多。

7.2.2 程序存储器扩展

程序存储器采用只读存储器，因为这种存储器在电源关断后，仍能保存程序（我们称此特性为非易失性的），在系统上电后，CPU 可取出这些指令重新执行。

1. 锁存器

受引脚数的限制，P0 口兼用作数据线和低 8 位地址线，为了将它们分离出来，需在单片机外部增加地址锁存器。目前，常用的地址锁存器芯片有 74LS373、74LS573 等。

（1）锁存器 74LS373

它是一种带三态门的 8D 锁存器，其引脚排列如图 7-7 所示，内部结构如图 7-8 所示。MCS-51 与 74LS373 锁存器的连接电路如图 7-9 所示。

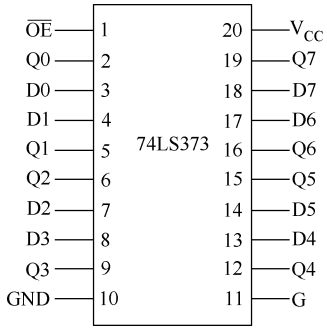


图 7-7 锁存器 74LS373 的引脚排列

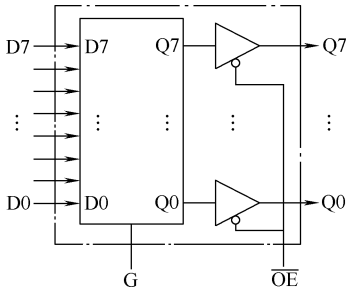


图 7-8 74LS373 的内部结构

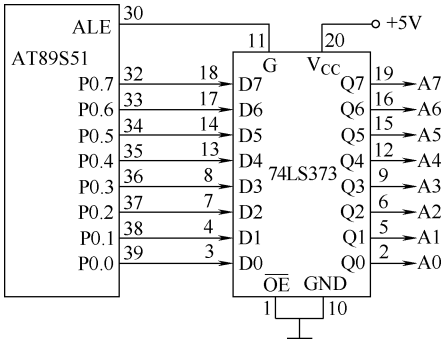


图 7-9 MCS-51 单片机 P0 口与 74LS373 的连接电路

引脚说明：

◆ D7 ~ D0：8 位数据输入线，

◆ Q7 ~ Q0：8 位数据输出线。

◆ G：数据输入锁存选通信号。当加到该引脚的信号为高电平时，外部数据选通到内部锁存器，负跳变时，数据锁存到锁存器中。

$\overline{OE}$：数据输出允许信号，低电平有效。当该信号为低电平时，三态门打开，锁存器中数据输出到数据输出线。当该信号为高电平时，输出线为高阻态。

74LS373 锁存器功能见表 7-1。

表 7-1　74LS373 锁存器功能表

$\overline{OE}$	G	D	Q
0	1	1	1
0	1	0	0
0	0	×	不变
1	×	×	高组态

（2）锁存器 74LS573

它也是一种带有三态门的 8 位锁存器，功能及内部结构与 74LS373 完全一样，只是其引脚排列与 74LS373 不同。图 7-10 所示为 74LS573 的引脚排列。

如图 7-10 所示，与 74LS373 相比，74LS573 的输入 D 端和输出 Q 端依次排列在芯片两侧，为绘制印制电路板提供了方便。

引脚说明：

◆ D7 ~ D0：8 位数据输入线。

◆ Q7 ~ Q0：8 位数据输出线。

◆ G：数据输入锁存选通信号，该引脚与 74LS373 的 G 端功能相同。

◆ $\overline{OE}$：数据输出允许信号，低电平有效。当该信号为低电平时，三态门打开，锁存器中数据输出到数据输出线。当该信号为高电平时，输出线为高阻态。

2. 程序存储器扩展

（1）2716 芯片介绍

下面以过去常用的最简单的 2716 芯片为例进行原理说明。2716 的引脚排列如图 7-11 所示。

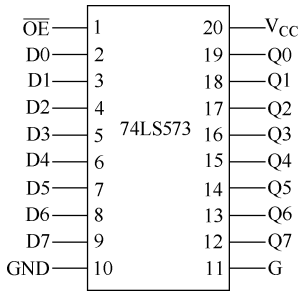


图 7-10　74LS573 的引脚

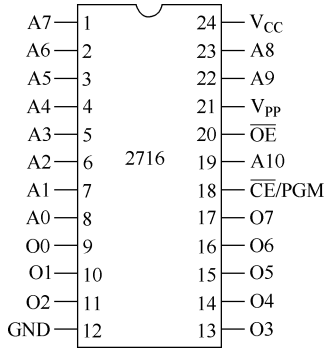


图 7-11　2716 的引脚排列

1）主要引脚功能

A10 ~ A0：11 位地址，它的数目由芯片的存储容量决定，用于进行单元选择。

O7 ~ O0：数据读出。

$\overline{CE}$/PGM：双重功能控制线，当使用时，它为片选信号$\overline{CE}$，低电平有效；当芯片编程时，它为编程控制信号 PGM，用于引入编程脉冲。

$\overline{OE}$：输出允许信号，当$\overline{OE}$ = 0 时，输出缓冲器打开，被寻址单元的内容可以被读出。

V_{PP}：编程电源。当芯片编程时，该端加 +25V 编程电压；当使用时，该端加 +5V 电源。

2）2716 的工作方式（见表 7-2）

表 7-2 2716 的工作方式表

引 脚 方 式	$\overline{CE}$/PGM	$\overline{OE}$	V_{PP}	$O_7 \sim O_0$
读出	低	低	+5V	程序读出
未选中	高	×	+5V	高阻
编程	正脉冲	高	+25V	程序写入
程序检验	低	低	+25V	程序读出
编程禁止	低	高	+25V	高阻

① 读出方式

CPU 从 EPROM 中读取代码，为单片机应用系统的工作方式。此时$\overline{CE}$、$\overline{OE}$均为低电平，V_{PP}端加 +5V。

② 维持方式

即未选中状态，此时$\overline{CE}$为高电平，数据输出为高阻状态，功耗下降 75%，处于低功率维持状态。

③ 编程方式

把程序代码固化到 EPROM 中。V_{PP}端加 +25V 电压，$\overline{OE}$高电平。每当$\overline{CE}$/PGM 端出现脉冲时，写入一个存储单元信息。

④ 编程校验方式

即检查编程写入的信息是否正确，通常紧跟编程之后。V_{PP}端加 +25V，$\overline{CE}$及$\overline{OE}$为低电平。

⑤ 编程禁止方式

2716 不但可单片编程，也允许多片同时编程，好把同样信息并行写入多片 2716 中。多片编程时，若要写入各片的数据不尽相同，可使某片或某几片芯片处于编程状态或编程禁止状态，当$\overline{CE}$/PGM 信号加低电平时，该芯片处于编程禁止状态，不写入数据。

（2）存储器扩展的主要内容

单片程序存储器扩展是只扩展一片 ROM 芯片，这是最简单的程序存储器扩展。下面以扩展 2716 芯片为例进行说明，其扩展连接图如图 7-12 所示。

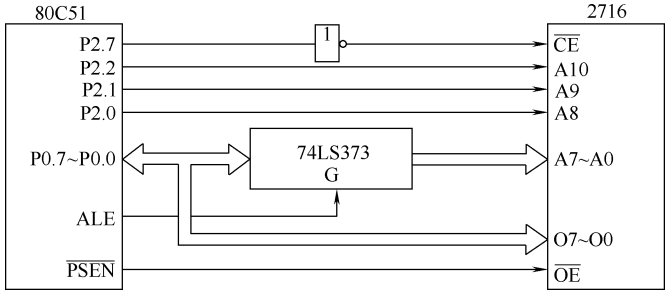


图 7-12 扩展连接图

存储器扩展的主要内容是地址线、数据线和控制线的连接。低位地址线的连接与存储芯片的容量有关。2716 的存储容量为 2KB，需 11 位地址（A10 ~ A0）进行存储单元编址。为此先把芯片的 A7 ~ A0 引脚与地址锁存器的 8 位地址输出对应连接，再把 A10 ~ A8 引脚与 P2 口的 P2.2 ~ P2.0 相连。这样，2716 芯片内存储单元的寻址问题就解决了。由于这是一个小规模存储器扩展系统，采用线选法进行片选，只需在剩下的高位地址线中取一位 P2.7 与 2716 的$\overline{CE}$端相连即可。

数据线的连接比较简单，只要把存储芯片的数据输出引脚与单片机 P0 口对应连接就可以了。对于控制信号，程序存储器的扩展只涉及$\overline{PSEN}$（外部程序存储器读选通），把该信号连接到 2716 的$\overline{OE}$引脚，用于存储器读出选通。

存储器单元地址分析：

上述三总线的连接过程中，地址总线只需 11 根（A0 ~ A10），占据了 P0 口的 8 根口线（P0.0 ~ 0.7）和 P2 口的 3 根口线（P2.0 ~ 2.2），片选线占据了 P2 口的 1 根口线（P2.7）。因此，P2 口还剩 4 根口线（P2.3 ~ 2.6）没有用，一般来说其状态是任意的。为了便于分析，通常假设没有用到的高位地址线 A11 ~ A14（P2.3 ~ 2.6）处于一种确定状态（如全部为“0”状态），则扩展的程序存储器芯片 Intel 2716 的地址范围分析见表 7-3。

表 7-3　存储器单元地址表

地址线	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
口线	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0
最低地址	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
最高地址	1	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1

由于 P2.6 ~ P2.3 的状态与 2716 芯片的寻址无关，所以在该芯片被寻址，P2.6 ~ P2.3 可以为任意状态，即从 0000 ~ 1111 共有 16 种状态组合。表明 2716 芯片对应着 16 个地址区间，即 8000H ~ 87FFH、8800H ~ 8FFFH、9000H ~ 97FFH、9800H ~ 9FFFH、A000H ~ A7FFH、A800H ~ AFFFH…。在这些地址区间内都能访问到 2716，这就是线选法存在的地址区间重叠问题。

7.2.3　数据存储器扩展

数据存储器扩展使用 RAM 芯片。现以 intel 6116 实现单片数据存储器扩展为例进行说明。

1. intel 6116 芯片

6116 芯片的存储容量为 2KB，该芯片为双列直插式封装，引脚排列如图 7-13 所示，工作方式见表 7-4。

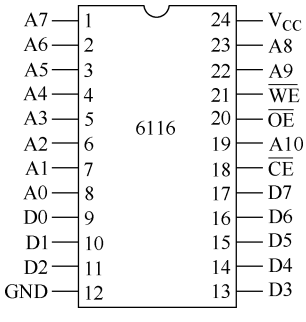


图 7-13　intel 6116 芯片引脚排列

主要引脚功能：

- A10 ~ A0：地址线。
- D7 ~ D0：数据线。
- $\overline{CE}$：片选信号。
- $\overline{OE}$：数据输出允许信号。
- $\overline{WE}$：写选通信号。

◆ V_{CC}：电源（+5V）。
◆ GND：地。
工作方式：
◆ 未选中
◆ 禁止
◆ 读出
◆ 写入

表 7-4 6116 的工作方式表

状　态	$\overline{CS}$	$\overline{OE}$	$\overline{WE}$	D7 ~ D0
未选中	1	×	×	高阻
禁止	0	1	1	高阻
读出	0	0	1	数据读出
写入	0	1	0	数据写入

2. 数据存储器扩展连接

数据存储器扩展与程序存储器扩展在数据线、地址线的连接上是完全相同的。不同之处在于控制信号，程序存储器的扩展使用$\overline{PSEN}$作为读选通信号，而数据存储器的扩展使用$\overline{RD}$和$\overline{WR}$分别作为读、写选通信号。使用一片 6116 实现 2KB RAM 扩展的电路连接图如图 7-14 所示。

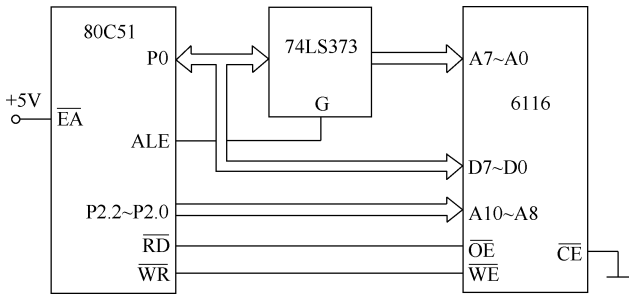


图 7-14 使用一片 6116 实现 2KB RAM 扩展的电路连接图

在扩展连接中，以$\overline{RD}$信号接 6116 的$\overline{OE}$引脚，以$\overline{WR}$信号接$\overline{WE}$引脚，进行 RAM 芯片的读/写控制。由于假定系统只有一片 6116，不需要片选信号，所以把$\overline{CE}$引脚直接接地。

为了便于分析，通常假设没有用到的高位地址线 A11 ~ A15（P2.3 ~ 2.7）全部为“0”状态，这样的话，6116 的地址范围是 0000H ~ 07FFH。则扩展的数据存储器芯片 Intel 6116 的地址范围分析见表 7-5。

表 7-5 Intel 6116 的地址范围分析表

地址线	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
口线	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0
最低地址	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
最高地址	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1

7.3　单片机并行 I/O 接口扩展

7.3.1　单片机并行 I/O 接口扩展基础知识

1. I/O 接口电路的功能

外部设备的速度十分复杂，必须通过 I/O 接口电路实现。

（1）速度协调

面对各种设备的速度差异，单片机无法按固定的时序以同步方式进行 I/O 操作，只能以异步方式进行，也就是只有在确认设备已为数据传送做好准备的前提下才能进行 I/O 操作。为此需要接口电路产生状态信号或中断请求信号，表明设备是否做好准备。即通过接口电路来进行单片机与外部设备之间的速度协调。

（2）输出数据锁存

由于 CPU 与外设速度的不一致，需要有接口电路把输出数据先锁存起来，待输出设备为接收数据做好准备后，再传送数据。这就是接口电路的数据锁存功能。

（3）数据总线隔离

总线上可能连接着多个数据源（输入设备）和多个数据负载（输出设备）。一个源和负载的数据传送正在进行时，所有其他不参与的设备在电性能上必须与总线隔开。这就是接口电路的总线隔离功能。

为了实现总线隔离，需要有接口电路提供具有三态缓冲功能的三态缓冲电路。

（4）数据转换

外部设备种类繁多，不同设备之间的性能差异很大，信号形式也多种多样。单片机只能使用数字信号，如果外部设备所提供或需要的不是电压形式的数字信号，就需要有接口电路进行转换，其中包括 A-D 转换和 D-A 转换等。

（5）增强驱动能力

通过接口电路为输出数据提供足够的驱动功率，以保证外部设备能正常、平稳地工作。

2. I/O 接口的特点

外部设备和 I/O 操作的复杂性，使接口电路成为单片机与外部设备之间必不可少的界面，通过接口电路居中协调和控制，保证外部设备的正常工作。有关 I/O 接口的特点可归结为如下 3 点：

（1）异步性

平时单片机与外部设备按各自的时序并行工作，只有在需要时外部设备才通过接口电路接受单片机的控制。

（2）实时性

单片机对外部设备的控制以查询或中断方式进行，以便最大限度地实现控制的实时化。

（3）与设备无关性

接口芯片不一定是专用的，同一个接口芯片通过软件设置可为多种设备实现接口。

3. 接口的分类

按数据传输方式的不同，接口有并行与串行之分，即并行接口与串行接口。

本章重点介绍的是并行接口。

4. I/O 编址技术

为了对 I/O 接口电路中的寄存器（端口）进行读/写操作，就需要对它们进行编址，所以就出现了 I/O 编址问题。有两种 I/O 编址方式：统一编址方式和独立编址方式。在 80C51 单片机系统中，采用统一编址方式。

所谓统一编址方式，就是把 I/O 接口中的寄存器与外扩展的数据存储器中的存储单元同等对待，合在一起使用同一个 64KB 的外扩展地址空间。I/O 和存储器的统一编址，使得 I/O口也采用 16 位地址编址，并使用数据存储器读/写指令进行 I/O 操作，而不需要专门的 I/O 指令。80C51 单片机系统中，采用统一编址方式。

所谓独立编址方式，就是把 I/O 与存储器分开进行编址。这样，在一个单片机系统中就形成了两个独立的地址空间：存储器地址空间和 I/O 地址空间。

独立编址方式的优点是两个地址空间相互独立、界限分明，但同时也存在许多麻烦并增加系统开销，所以独立编址方式在单片机中较少采用。

5. 单片机 I/O 控制方式

（1）无条件方式

无条件传送也称为同步程序传送。只有那些能一直为 I/O 操作做好准备的设备，才能使用无条件传送方式。在进行无条件 I/O 操作时，无需测试设备的状态，可以根据需要随时进行 I/O 操作。

无条件传送适用于两类设备的 I/O 操作：一类是具有常驻的或变化缓慢的数据信号的设备，如机械开关、指示灯、发光二极管、数码管等；另一类则是工作速度非常快，足以和单片机同步工作的设备，如 D-A 转换器（D-AC）。

（2）查询方式

查询方式又称有条件传送方式，在 I/O 操作前，要检测设备的状态，只有在确认设备已“准备好”的情况下，单片机才能执行 I/O 操作。检测也称为“查询”，所以就把这种有条件的 I/O 控制方式称为查询方式。

为实现查询方式的 I/O 控制，需要由接口电路提供设备状态，接口电路中的状态寄存器或状态位就是为此而准备的，查询方式只适用于规模比较小的单片机系统。

（3）中断方式

中断方式与查询方式的主要区别在于如何知道外部设备是否为 I/O 操作做好准备。采用中断方式进行 I/O 控制时，当设备做好准备之后，就向单片机发出中断请求。单片机接收到中断请求之后作出响应，暂停正在执行的原程序，而转去执行中断服务程序，通过执行中断服务程序完成一次 I/O 操作，然后程序返回，单片机再继续执行被中断的原程序。

中断方式效率较高，所以在单片机系统中被广泛采用。但中断请求是一种不可预知的随机事件，所以实现起来对单片机系统的硬件和软件都有较高的要求。

7.3.2 8255 可编程并行接口扩展

8255 是美国 intel 公司产品，因其工作方式和操作功能等可通过程序进行设置和改变，称为可编程接口芯片。

1. 8255 硬件逻辑结构

8255 的全称是“可编程并行输入/输出接口芯片”，具有通用性强且使用灵活等优点，可用于实现 80C51 系列单片机的并行 I/O 口扩展。

8255 是一个 40 引脚的双列直插式集成电路芯片，其引脚排列如图 7-15 所示。

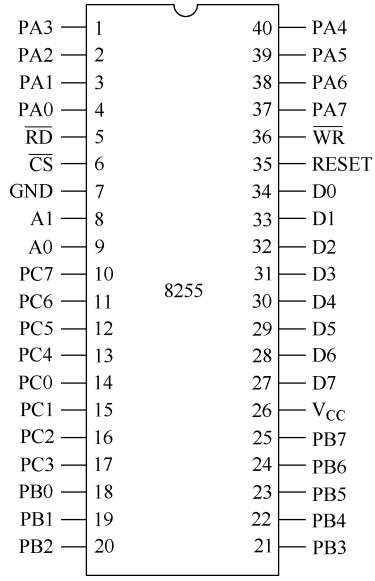


图 7-15　8255 芯片引脚排列

8255A 引脚说明：

① RESET：复位信号，输入高电平有效。

② $\overline{CS}$：片选信号，输入低电平有效。

③ $\overline{RD}$：读信号，输入低电平有效。

④ $\overline{WR}$：写信号，输入低电平有效。

⑤ D0 ~ D7：三态双向数据总线。

⑥ PA0 ~ PA7：端口 A 数据线，双向。

按功能可把 8255 的内部结构分为 3 个逻辑电路部分，分别为口电路、总线接口电路和控制逻辑电路，如图 7-16所示。

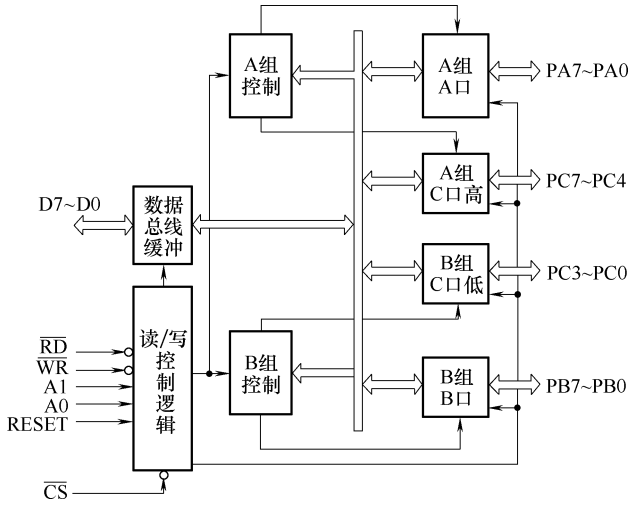


图 7-16　8255 的逻辑结构图

（1）口电路

8255 共有 3 个 8 位口，其中 A 口和 B 口是单纯的数据口，作为数据输入/输出使用；而 C 口则既可以作数据口使用，又可以作控制口使用，主要用于实现 A 口和 B 口的控制功能。在使用中常把 C 口分为两部分，即 C 口高位部分（PC7 ~ PC4）和 C 口低位部分（PC3 ~ PC0）。数据传送中，A 口所需的控制信号由 C 口高 4 位（PC7 ~ PC4）提供，把 A 口和 C 口高位部分合在一起称为 A 组；同理，把 B 口和 C 口低位部分合在一起称为 B 组。

（2）总线接口电路

总线接口电路用于实现 8255 和单片机芯片的信号连接。其中包括：

① 数据总线缓冲器。数据总线缓冲器为 8 位双向三态缓冲器，可直接与系统数据总线相连，与 I/O 操作有关的数据、控制字和状态信息都是通过该缓冲器进行传送的。

② 读/写控制逻辑。读/写控制逻辑用于实现 8255 的硬件管理，其内容包括：芯片的选择，口的寻址以及规定各端口和单片机之间的数据传送方向等。相关的控制信号有：$\overline{CS}$、$\overline{RD}$、$\overline{WR}$、A0、A1、RESET。其中，A0、A1 低位地址信号，用于端口选择，8255 共有 4 个可寻址的端口。当 RESET 复位信号是高电平时，芯片复位后，控制寄存器清 0，各端口被置为输入方式。8255 端口选择及读/写控制见表 7-6。

表 7-6 8255 端口选择及读/写控制

$\overline{CS}$	A1	A0	$\overline{RD}$	$\overline{WR}$	选择端口	端口操作
0	0	0	0	1	A 口	读端口 A
0	0	1	0	1	B 口	读端口 B
0	1	0	0	1	C 口	读端口 C
0	0	0	1	0	A 口	写端口 A
0	0	1	1	0	B 口	写端口 B
0	1	0	1	0	C 口	写端口 C
0	1	1	1	0	控制寄存器	写控制命令
1	×	×	×	×	—	数据总线缓冲器输出端呈高阻抗

（3）A 组和 B 组控制电路

A 组控制和 B 组控制合在一起构成 8255 的控制电路，其中包括一个 8 位控制寄存器，用于存放编程命令和实现各口操作控制。

（4）中断控制电路

8255 逻辑电路中还包含一个中断控制电路。中断控制电路中对应 A、B 两个口各有一个中断触发器，即触发器 A 和触发器 B，用于对中断的允许和禁止进行控制，置位为允许，复位为禁止。对两个触发器的置位和复位控制是通过口 C 的有关位进行的，具体划分是，在输入方式下，PC4 对应触发器 A，PC2 对应触发器 B；在输出方式下，PC6 对应触发器 A，PC2 对应触发器 B。

2. 8255 的工作方式

8255 共有 3 种工作方式：方式 0、方式 1 和方式 2。

（1）方式 0（基本输入/输出方式）

方式 0 适用于无条件数据传送。两个 8 位口（A 口和 B 口）和两个 4 位口（C 口高位部分和 C 口低位部分）都可以分别或同时设置为方式 0。

（2）方式 1（选通输入/输出方式）

方式 1 是选通输入/输出方式。8255 的“选通”是通过信号的“问”与“答”，以联络方式（或称握手方式）实现的。所以这种数据传送方式是有条件的，适用于以查询或中断方式进行控制。

在方式 1 下，A 口和 B 口是数据口，C 口是控制口，用于传送和保存数据口所需要的联络信号都有具体的定义。这些联络信号见表 7-7。

表 7-7　C 口联络信号表

C 口位线	方式 1		方式 2	
	输入	输出	输入	输出
PC7		$\overline{\text{OBFA}}$		$\overline{\text{OBFA}}$
PC6		$\overline{\text{ACKA}}$		$\overline{\text{ACKA}}$
PC5	IBFA		IBFA	
PC4	$\overline{\text{STBA}}$		$\overline{\text{STBA}}$	
PC3	INTRA	INTRA	INTRA	INTRA
PC2	$\overline{\text{STBB}}$	$\overline{\text{ACKB}}$		
PC1	IBFB	$\overline{\text{OBFB}}$		
PC0	INTRB	INTRB		

在该方式下，A 口和 B 口的联络信号都是 3 个。在具体应用中，如果只有一个口按方式 1 使用，需占用 11 位（8 + 3 = 11）口线，剩下的 13 位口线可按其他方式使用；如果两个口都按方式 1 使用，则只剩下 2 位口线可做它用。

（3）方式 2（双向数据传送方式）

方式 2 是在方式 1 的基础上加上双向传送功能，只有 A 口才能选择这种工作方式，这时 A 口既能输入数据又能输出数据。如果把 A 口置于方式 2 下，则 B 口只能工作于方式 0。方式 2 适用于查询或中断方式的双向数据传送。在这种方式下需使用 C 口的 5 位口线作控制线。

3. 8255 的编程内容

8255 是可编程接口芯片，主要编程内容是两条控制命令，即工作方式命令和 C 口位置位/复位命令。

（1）工作方式命令

工作方式命令用于设定各数据口的工作方式及数据传送方向。命令的最高位 D7 是标志位，其状态固定为 1。其命令格式如图 7-17 所示。

对工作方式命令有如下两点说明：

◆ A 口有 3 种工作方式，而 B 口只有两种工作方式。

◆ 在方式 1 和方式 2 下，对 C 口的定义（输入或输出）不影响作为联络信号使用的 C 口各位的功能。

（2）C 口位置位/复位命令

在方式 1 和方式 2 下，C 口用于定义控制信号和状态信号，因此 C 口的每一位都可以进行置位或复位。对 C 口各位的置位或复位是由位置位/复位命令进行的。8255 的位置位/复位命令格式如图 7-18 所示。

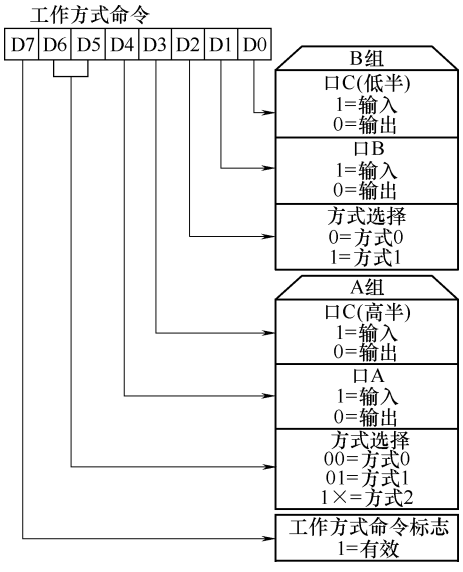


图 7-17　8255 工作方式的命令格式

其中，D7 为该命令的标志，其状态固定为 0。在使用时，该命令每次只能对 C 口中的一位进行置位或复位。

（3）接口与初始化编程

8255 初始化的内容就是向控制字寄存器写入命令。例如，若对 8255 各口作如下设置：A 口方式 0 输入；B 口方式 1 输出；C 口高位部分为输出，低位部分为输入。设控制寄存器地址为 0003H。按各口的设置要求，工作方式命令字为 10010101，即 95H。则初始化程序段应为

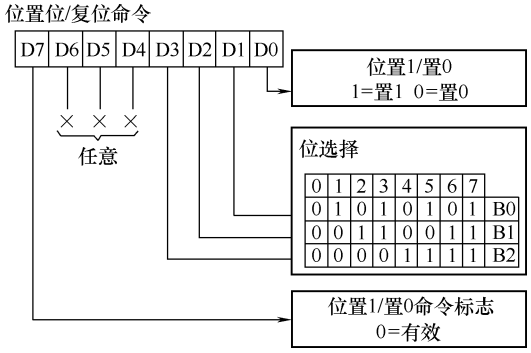


图 7-18　8255 的位置位/复位命令格式

```
MOV R0，#03H
MOV 1，#95H
MOV@R0，A
```

4. 8255 的 I/O 控制方式

8255 中可以使用无条件方式、查询方式和中断方式共 3 种 I/O 控制方式。

（1）无条件方式

以方式 0 进行数据输入/输出，就是无条件传送方式。

（2）查询方式

在方式 1 和方式 2 下，都可以使用查询方式进行数据传送。数据输入时，供查询的状态信号是 IBF（对应 A 口为 IBFA，B 口为 IBFB），因为传送这些信号的口线分别为 PC5 和 PC1，所以查询时就是对输入这些口线的状态进行测试。数据输出时，供查询的状态信号是 OBF（对应 A 口为 OBFA，B 口为 OBFB），被测试的口线为 PC7 和 PC1。

（3）中断方式

在方式 1 和方式 2 下，都可以使用中断方式进行数据传送。中断请求信号是 INTR（对应 A 口为 INTRA，B 口为 INTRB），传送中断请求信号的口线分别为 PC3 和 PC0。

7.3.3　8155 可编程并行接口扩展

8155 是一种多功能可编程外围扩展接口芯片，它也有三个可编程 I/O 端口（端口 A、B、C）。与 8255A 的区别在于 PC 口是 6 位，同时还有一个可编程 14bit 定时器/计数器和 256B 的 RAM，能方便地进行 I/O 扩展和 RAM 扩展。其内部逻辑结构如图 7-19 所示。

1. 8155 引脚

8155 芯片为 40 引脚双列直插封装，单一的 +5V 电源，其引脚排列如图 7-20 所示。各引脚说明如下：

AD0 ~ AD7：8 位三态地址/数据线。当 ALE 为高电平，作地址线；ALE 下降沿时，锁存有关信号；ALE 低电平时，作数据线（$\overline{RD}$、$\overline{WR}$均无效时作高阻状态）。

IO/$\overline{M}$：I/O 端口和 RAM 选择信号输入端口。当为 1 时，选择 I/O 端口；当为 0 时，选择 RAM 单元。

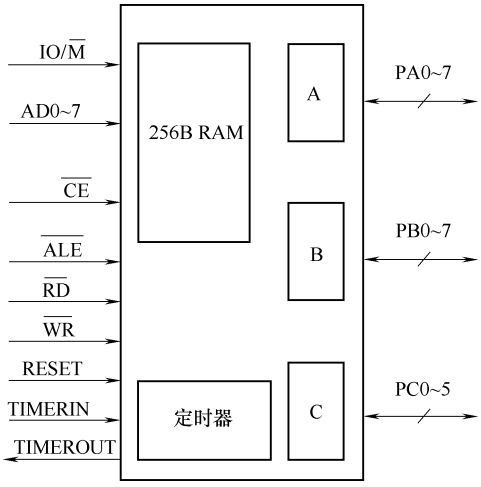


图 7-19　8155 的内部逻辑结构

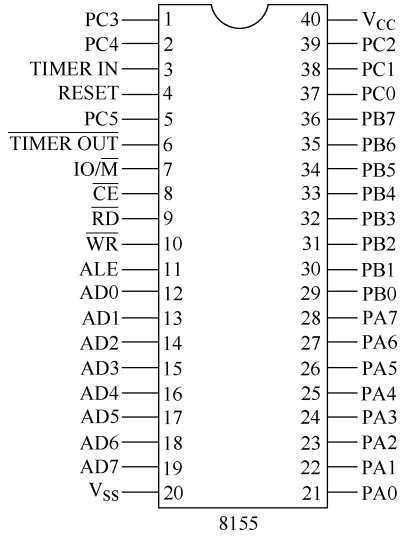


图 7-20　8155 芯片引脚排列

$\overline{CE}$：片选信号输入线，低电平有效。

ALE：地址锁存允许信号，输入端口。高电平时将低 8 位地址写入 8155 的内部锁存器中，当 ALE 信号处于下降沿时，将 AD0 ~ AD7、$\overline{CE}$、IO/$\overline{M}$信号都锁存到 8155 的内部锁存器。

$\overline{RD}$：读选通信号，低电平有效，输入。有效时，根据 IO/$\overline{M}$信号状态选择读 I/O 口还是读 RAM。

$\overline{WR}$：写选通信号，低电平有效，输入。有效时，根据 IO/$\overline{M}$信号状态选择写 I/O 还是写 RAM。

TIMERIN：定时器时钟信号输入端。

$\overline{TIMEROUT}$：定时器输出。当计数器减到 0，则根据设定的工作方式输出一个方波还是一个脉冲。

RESET：复位控制信号线，高电平有效。

PA0 ~ PA7：A 口的 8 位 I/O 线。

PB0 ~ PB7：B 口的 8 位 I/O 线。

PC0 ~ PC7：C 口的 6 位 I/O 线。

V_{CC}：电源线，+5V。

V_{SS}：线路地。

2. 8155 的 RAM 和 I/O 端口寻址

8155 的 RAM 单元与一般外部数据存储器的使用基本一样，唯一区别是事先要使 IO/$\overline{M}$为低电平。A、B、C 各端口可工作于不同的工作方式，使用前要进行初始化（写命令字到命令口）。8155 的 RAM 单元和端口地址选择见表 7-8。

（1）当 IO/$\overline{M}$ = 0 时，选中 8155 内部 RAM，其地址范围为 00H ~ FFH。

（2）当 IO/$\overline{M}$ = 1 时，选择命令/状态寄存器、A 口、B 口、C 口、定时器低 8 位、定时器高 6 位。

（3）命令/状态寄存器是同一地址的两个独立寄存器，由读写信号加以区分，命令寄存器只能写入，状态寄存器只能读出。

表 7-8 端口地址分配

$\overline{CE}$	IO/$\overline{M}$	A2	A1	A0	所选端口
0	1	0	0	0	命令/状态寄存器
0	1	0	0	1	A 口
0	1	0	1	0	B 口
0	1	0	1	1	C 口
0	1	1	0	0	定时器低 8 位
0	1	1	0	1	定时器高 6 位
0	0	×	×	×	RAM 单元

3. 8155 的工作方式

（1）8155 作片外 RAM（256B）

① 在片选信号$\overline{CE}$有效的情况下，IO/$\overline{M}$信号为高电平，8155 作片外 RAM 使用。

② 地址的高 8 位由片选信号确定，地址的低 8 位为 00H ~ FFH。

（2）8155 作 I/O 口

8155 有两种工作方式：基本 I/O 和选通 I/O。

① 当 8155 的 PA 口、PB 口、PC 口工作在基本 I/O 方式下，可用于无条件 I/O 操作。基本输入时执行“MOVX A，@ DPTR”类指令，基本输出时执行“MOVX@ DPTR，A”类指令。

② 当 8155 的 PA 口工作在选通 I/O 方式下时，PC 口低三位作 PA 口联络线，其余位作 I/O 线，PB 口定义为基本 I/O；PA 口和 PB 口也可同时定义为选通 I/O，此时 PC 作 PA 口、PB 口联络线。

③ 在 8155 操作前，须由 CPU 向命令寄存器送命令字，设定其工作方式，命令字只能写入不能读出。

④ 8155 内部还有一个状态寄存器，可以锁存 8155 I/O 口和定时器的当前状态，供 CPU 查询，状态寄存器和命令寄存器共用一个地址，只能读出不能写入。因此可以认为 8155 的 00H 口是命令/状态口，CPU 往 00H 写入的是命令字，而从中读出的是状态字。

4. 8155 的控制字及其工作方式

（1）命令寄存器

A、B、C 各端口可工作于不同的工作方式，使用前要进行初始化（写命令字到命令口）。8155 命令寄存器格式见表 7-9。

表 7-9 8155 命令寄存器格式

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
TM2	TM1	IEB	IEA	PC2	PC1	PB	PA

各位功能说明如下：

1）TM2 TM1

① 00，代表空操作。

② 01，代表停止计时。

③ 10，代表时间到则停止计时。

④ 11，代表置入定时器方式控制字和计数初值后，立即启动计时。若正在计时，溢出后按新的定时器方式和计数初值计时。

2）IEB

0 代表禁止 PB 口中断；1 代表允许 PB 口中断。

3）IEA

0 代表禁止 PA 口中断；1 代表允许 PA 口中断。

4）PC2 PC1：

① 00 代表 PA、PB 均为基本 I/O 方式，PA 和 PB 输入/输出由 D_1D_0 确定，PC 口输入。

② 01 代表 PA、PB 均为基本 I/O 方式，PA 和 PB 输入/输出由 D_1D_0 确定，PC 口输出。

③ 10 代表 PA 选通 I/O，PB 为基本 I/O 方式，输入/输出由 D_1D_0 确定；PC0 ~ PC2 为 PA 口联络线；PC3 ~ PC5 输出。

④ 11 代表 PA、PB 均为选通方式，输入 \ 输出由 D_1D_0 确定；PC0 ~ PC2 为 PA 口联络线；PC3 ~ PC5 为 PB 口联络线。

5）PB：0 代表输入；1 代表输出。

6）PA：0 代表输入；1 代表输出。

（2）状态寄存器

状态字只能读不能写，所以 8155 的命令字和状态字共用一个地址。当对命令/状态字进行写操作时，写进去的是命令，当对命令/状态字进行读操作时，读出来的是状态。状态字用于寄存各端口及定时器/计数器的工作状态。

8155 状态寄存器格式见表 7-10。

表 7-10　8155 状态寄存器格式

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
×	TIMER	INTEB	BEF	INTRB	INTEA	ABF	INTRA
未用	定时中断 1：计数器溢出标志 0：读出状态或复位	B 口中断允许位 1：允许 0：禁止	B 口缓冲器满标志 0：空 1：满	B 口中断请求标志 0：空 1：无	A 口中断允许位 0：禁止 1：允许	A 口缓冲器满标志 0：空 1：满	A 口中断请求标志 0：无 1：有

5. 8155 内部定时器及使用

定时器共有 4 种工作方式，由定时器长度字高字节中的 M2、M1 两位状态决定，定时器长度字的低 14 位用于给定时器设置初值。定时器长度字格式如图 7-21 所示。

现对定时器在 4 种工作方式下的$\overline{\text{TIMER OUT}}$输出波形分述如下：

① 在 M2M1 = 00 时，定时器在计数的后半周期内使$\overline{\text{TIMER OUT}}$线上输出低电平（一个矩形波）。矩形波周期和定时器长度字初值有关：若定时器长度字初值为偶数，则$\overline{\text{TIMER OUT}}$线上的矩形波是对称的；若它为奇数，则矩形波高电平持续期比低电平的多一

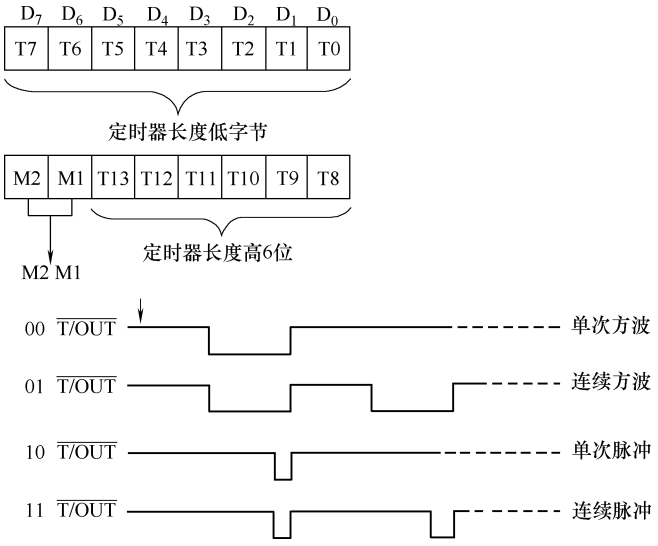


图 7-21 定时器长度字格式

个计数脉冲时间。

② 在 M2M1 = 01 时，定时器每当减“1”到全“0”时，都能自动装入定时器长度字初值，故$\overline{\text{TIMER OUT}}$线上输出连续矩形波。矩形波周期也与定时器长度字初值的设定有关。

③ 在 M2M1 = 10 时，定时器每当减“1”到全“0”时，便会在$\overline{\text{TIMER OUT}}$线上输出一个单脉冲。

④ 在 M2M1 = 11 时，定时器每当减“1”变为全“0”时，都能自动装入定时器长度字初值，故$\overline{\text{TIMER OUT}}$线上能输出一串重复脉冲。重复脉冲的频率也和定时器长度字初值有关。

8155 对定时器的控制是由命令字中的 D_7D_6 两位状态决定的（见图 7-21）。现把这两位对定时器的控制分述如下：

$D_7D_6 = 00$ 时，无操作。即 $D_7D_6 = 00$ 的命令字对定时器工作不产生影响。

$D_7D_6 = 01$ 时，停止计数。若定时器原为停止状态，则它继续停止计数；若定时器正在运行，则 $D_7D_6 = 01$ 的命令字送给 8155 后便能立即停止定时器的减“1”计数。

$D_7D_6 = 10$ 时，计满后停止。若定时器原为停止状态，则它继续停止计数；若定时器正在运行，则 8155 收到 $D_7D_6 = 10$ 的命令字后，必须等到定时器回零时才会停止计数。

$D_7D_6 = 11$ 时，开始计数。若定时器原为停止状态，它收到 $D_7D_6 = 11$ 的命令字后立即开始计数；若定时器正在运行，则它在回零后立即按新输入的定时器长度字开始计数。

在定时器计数期间，CPU 随时可以读出定时器中的状态，以了解定时器的工作情况。

7.4 键盘接口技术

键盘是单片机应用系统中人机交流不可缺少的输入设备。键盘由一组规则排列的按键组成，一个按键实际上是一个开关元件。键盘通常使用机械触点式按键开关，其主要功能是把机械上的通断转换为电气上的逻辑关系（1 和 0）。

键盘的分类：

1）按照键盘的形式来分的话，可分为独立式按键和矩阵式键盘。

◆ 独立式键盘：按键之间相互独立，每个按键的工作不会影响其他按键。

◆ 矩阵键盘：排列成矩阵形式，在行列交叉点上对应一个按键。

2）按照键盘的功能分来的话，又可以分为编码键盘和非编码键盘。

◆ 编码键盘：能自动识别按下的键并产生相应代码，以并行或串行方式送给 CPU。它使用方便、接口简单、响应速度快，但较贵。

◆ 非编码键盘：通过软件来确定按键并计算键值。这种方法没有编码键盘速度快，但它价格便宜，因此得到了广泛的应用。

编码键盘主要是用硬件来实现对键的识别，非编码键盘主要是由软件来实现键盘的定义与识别。非编码键盘只简单地提供行和列的矩阵，其他工作均由软件完成。由于其经济实用，较多地应用于单片机系统中。本章中主要介绍非编码的独立键盘和矩阵键盘。

7.4.1　键盘的结构

1. 独立式按键

其结构如图 7-22 所示，特点是每个按键单独占用一根 I/O 口线，每个按键工作不会影响其他 I/O 口线的状态；多用于所需按键不多的场合；可采用 JNB（或 JB）来查询哪一个按键按下，并转向相应的功能处理程序。

2. 矩阵式键盘

单片机系统中，若使用按键较多时，通常采用矩阵式键盘，结构如图 7-23 所示。由图可知，一个 4 ×4 的行、列结构，可以构成一个含有 16 个按键的键盘，节省了很多 I/O 口。

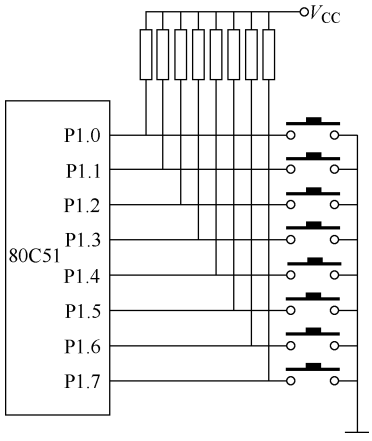


图 7-22　独立键盘结构及与单片机的接口

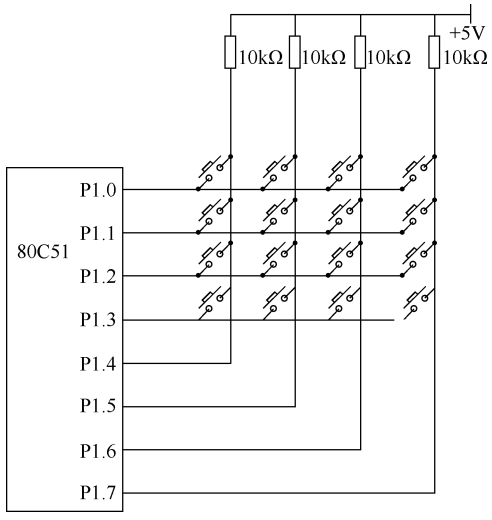


图 7-23　矩阵键盘结构及与单片机的接口

控制方式：先判断是否有键按下。如有，再判断哪一键按下，并得到键码值，然后根据键码值转向不同的功能程序。矩阵式结构键盘比独立式按键要复杂，识别也要复杂一些。最常用的识别方法是键盘扫描法。

3. 键盘设计需解决的几个问题

（1）按键的确认

键盘实际上是一组按键开关的集合，其每一个按键就是一个开关量输入装置。键的闭合与否，通过电平状态（高或低）的检测，便可确定相应按键是否已被按下。例如，高电平表示断开，低电平表示闭合。

（2）重键与连击的处理

实际按键操作中，若无意中同时或先后按下两个以上的键，系统确认哪个键操作是有效的完全由设计者的意志决定。

① 以按下时间的长短为准；

② 以最先按下的键为当前按键；

③ 也可以将最后释放的键看成是输入键。

通常总是采用单键按下有效，多键同时按下无效的原则（若系统设有复合键，当然应该另当别论）。

（3）按键防止抖动

1）按键抖动产生的原因

多数键盘的按键均采用机械弹性开关。其主要功能是把机械上的通断转换成为电气上的逻辑关系。也就是说，它能提供标准的 TTL 逻辑电平，以便与通用数字系统的逻辑电平相容。

一个电信号通过机械触点的断开、闭合过程，完成高、低电平的切换。由于机械触点的弹性作用，一个按键开关在闭合及断开的瞬间必然伴随有一连串的抖动（触点在闭合和断开瞬间的电接触情况不稳定，造成了电压信号的抖动现象）。其波形如图 7-24 所示。抖动过程的长短由按键的机械特性决定，一般为 5～10ms（或 10～20ms）。

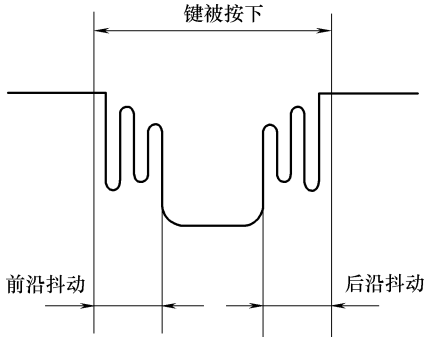


图 7-24　按键抖动信号波形

在触点抖动期间检测按键的通与断状态，可能导致判断出错。即按键一次按下或释放被错误地认为是多次操作，这种情况是不允许出现的。

2）键盘消除抖动的措施

为了克服按键触点机械抖动所致的检测误判，必须采取去抖动措施，可从硬件、软件两方面予以考虑。在键数较少时，可采用硬件去抖；而当键数较多时，采用软件去抖。

① 硬件防抖技术。通过硬件电路消除按键过程中抖动的影响是一种广为采用的措施。这种作法工作可靠，且节省机时。硬件防抖技术包括两种：滤波防抖电路、双稳态防抖电路。

a. 滤波防抖电路。利用 *RC* 积分电路对于干扰脉冲的吸收作用，只要选择好时间常数，就能在按键抖动信号通过此滤波电路时，消除抖动的影响。滤波防抖电路如图 7-25 所示。

当键 K 未按下时，电容 *C* 两端电压均为 0，非门输出为 1。

当 K 刚按下时，由于 C 两端电压不可能产生突变，尽管在触点接触过程中可能出现抖动，只要适当选取 R_1、R_2 和 C 值，即可保证电容 C 两端的充电电压波动不超过非门的开启电压（TTL 为 0.8V），非门的输出将维持高电平。

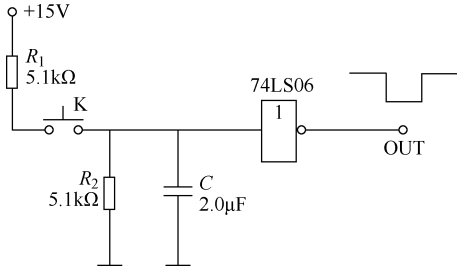


图 7-25　滤波防抖电路

同理，当触点 K 刚断开时，由于电容 C 经过电阻 R_2 放电，C 两端的放电电压波动不会超过门的关闭电压，因此门的输出也不会改变。

只要 R_1、R_2 和 C 的时间常数选取得当，确保电容 C 由稳态电压充电到开启电压，或放电到关闭电压的延迟时间等于或大于 10ms，该电路就能消除抖动的影响。

b. 双稳态防抖电路。在硬件上可采用在键输出端加 R-S 触发器（双稳态触发器）或单稳态触发器构成去抖动电路。图 7-26 所示是一种由 R-S 触发器构成的双稳态防抖动电路，当触发器翻转时，触点抖动不会对其产生任何影响。键盘输出经双稳态电路之后变为规范的矩形方波。

按键未按下时，A = 0、B = 1，输出 Q = 1，按键按下时，因按键的机械弹性作用的影响，使按键产生抖动，当开关没有稳定到达 B 端时，因与非门 2 输出为 0 反馈到与非门 1 的输入端，封锁了与非门 1，双稳态电路的状态不会改变，输出保持为 1，输出 Q 不会产生抖动的波形。当开关稳定到达 B 端时，因 A = 1、B = 0，使 Q = 0，双稳态电路状态发生翻转。

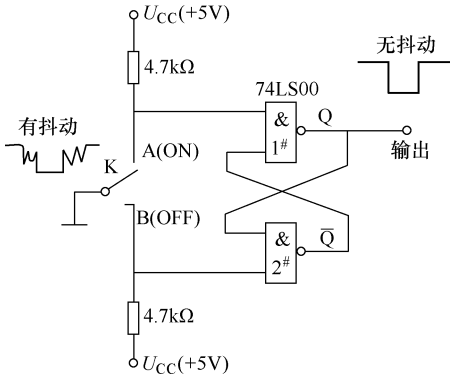


图 7-26　双稳态防抖电路

当释放按键时，在开关未稳定到达 A 端时，因 Q = 0，封锁了与非门 2，双稳态电路的状态不变，输出 Q 保持不变，消除了后沿的抖动波形。当开关稳定到达端 A 时，因 A = 0、B = 1，使 Q = 1，双稳态电路状态发生翻转，输出 Q 重新返回原状态。可见，键盘输出经双稳态电路之后，输出已变为规范的矩形方波。

② 软件防抖方法。如前所述，若采用硬件防抖电路，则 N 个键就必须配有 N 个防抖电路。因此，当键的个数比较多时，硬件防抖将无法胜任。在这种情况下，可以采用软件的方法进行防抖。当第一次检测到有键按下时，先用软件延时 10 ~ 20ms（具体时间应视所使用的按键进行调整），而后再确认该键电平是否仍维持闭合状态电平。若保持闭合状态电平；则确认此键确已按下，从而消除了抖动的影响。

7.4.2　键码和键盘的扫描

1. 键码

键盘上的每个键都担负一项处理功能，而处理功能是通过软件实现的，所以键盘接口必须有软件配合。为此，键盘上每个键都对应有一个处理程序段，键的功能是通过运行这个程

序段实现的。为了在程序中能顺利地分支到键处理程序段，就需要对键进行编码，称为键码，以便能按键码进行程序分支。键的编码没有统一标准，只要能实现键处理程序的正确分支就可以，因此，就存在多种多样的键编码方法。

2. 键盘的扫描

（1）键盘的原理及编码

为了弄清键盘的扫描方法，先来了解键盘的编码方法。以一个 8 行、4 列的矩阵键盘为例，如图 7-27 所示。

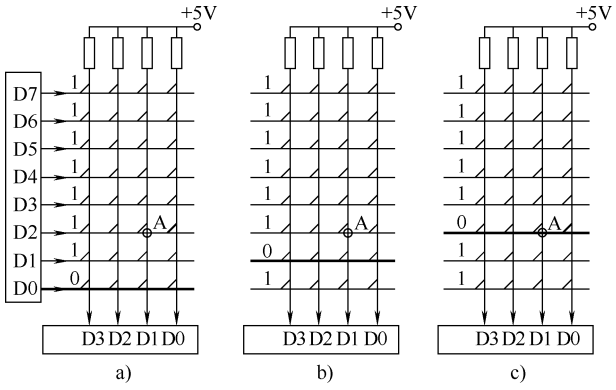


图 7-27　键盘扫描示意图

键盘上有行线和列线之分，本键盘共有 8 条行线 4 条列线。在行线和列线的交点处有一个键，由于行线与列线分别与键的不同端相连，平时键处于断开状态，所以行线和列线互不相通。接口时，行线一端接输出口，另一端悬空；而列线一端经电阻接 +5V 电源，另一端接输入口。由于列线通过电阻与 +5V 电源相连，所以列线的初始状态为高电平。

键盘最常用的编码方法是以键在键盘矩阵中的位置，从 0 开始按自然数顺序进行编码，键码以十六进制数表示。例如，表 7-11 所列的是图 7-27 所示键盘上各键的键码，其值从 00H ~ 1FH。

表 7-11　8 × 4 键盘键码

7FH	1FH	17H	0FH	07H
BFH	1EH	16H	0EH	06H
DFH	1DH	15H	0DH	05H
EFH	1CH	14H	0CH	04H
F7H	1BH	13H	0BH	03H
FBH	1AH	12H	0AH	02H
FDH	19H	11H	09H	01H
FEH	18H	10H	08H	00H
	F7H	FBH	FDH	FEH

（2）扫描方法

通常把键盘上被按下的键称为闭合键。为了识别闭合键，即判定键盘上有没有键被按下

以及哪个键被按下，有行扫描法和线反转法两种方法可供选用。在单片机中常用的是行扫描法，简称扫描法。这里介绍的键盘扫描是由软件实现的。软件方法键盘扫描是在扫描程序驱动下进行的，所以扫描过程也就是扫描程序的执行过程。

开始前，通过程序反复不断地进行闭合键查找，即看看键盘中是否有闭合键，为此，应先使行线输出口输出全为 0，再读回列线状态。若列线状态为全 1，则表明没有键被按下；若不为全 1，则表明有键被按下。因为当有键被按下时，由于行线与列线在闭合键交点处接通，使穿过闭合键的那条列线变为低电平。发现闭合键后才接着进行键盘扫描，判定闭合的是哪个键；若无闭合键，就返回去重复进行闭合键的查找。

键盘扫描过程是依次使行线中的每一条输出低电平，接着输入列线状态进行有无闭合键的判定。假定图 7-27 中 A 键被按下，键盘扫描的过程是，先经输出口在行线上输出 FEH，然后输入列线，测试列线状态中是否有 0（见图 7-27a）；若没有，再经输出口输出 FDH，再测试列线状态（见图 7-27b）……；直到行线输出为 FBH 时，列线中有状态为 0 的位，列线状态为 FB（假定输入口中没用的引脚为高电平），说明在该列线上有闭合键（见图 7-27c）。

发现闭合键后，扫描并未结束。因为还要判定是否还有其他键被同时按下，所以扫描还应继续下去，直至最后在行线上输出 7FH 为止。

（3）键盘扫描程序流程

从行扫描到键码生成的键盘扫描程序流程如图 7-28 所示。

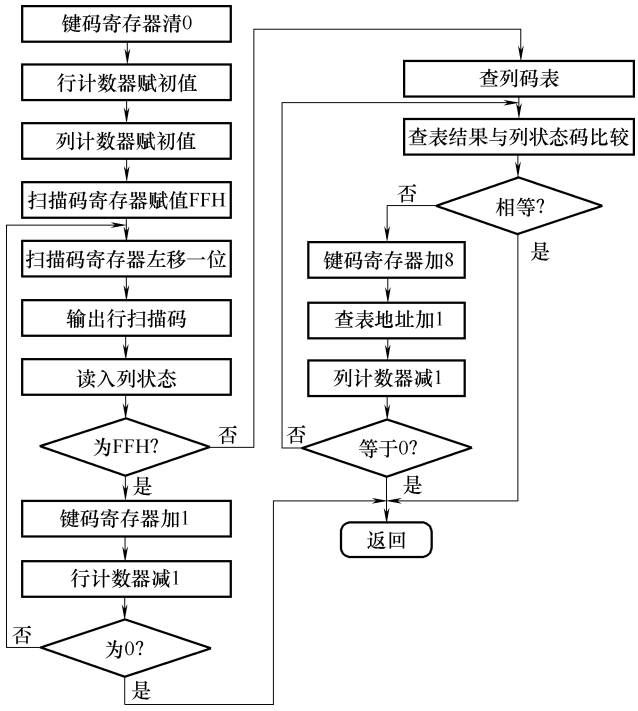


图 7-28　键盘扫描程序流程

从行的方面看，为形成行扫描码设置一个扫描码寄存器。参照图 7-27 所示电路，由于

扫描是从最下行开始的，所以扫描码寄存器赋初值 FEH。以后其他各行的扫描码可以通过扫描寄存器左移一位来形成。从列的方面看，为了与扫描过程中读回的列状态进行比较，可预先把各列的状态码写成一个数据表，称为列码表。以便每次扫描与读回的列状态进行比较。

为了生成和保存键码，还设置了一个键码寄存器，并赋初值 00H。行计数器和列计数器用于控制扫描。

7.4.3 用 8255 实现键盘接口

1. 接口电路逻辑图

以 8255 作 8×4 键盘的接口为例。A 口为输出口，接键盘行线；C 口为输入口，以 PC3～PC0 接键盘的 4 条列线；电路如图 7-29 所示。

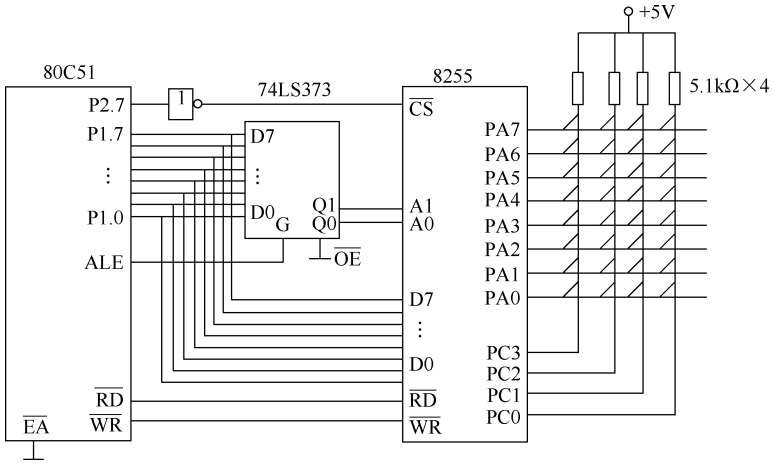


图 7-29 矩阵键盘接口电路

如图 7-29 所示，假定 A 口地址为 8000H，则 B 口地址为 8001H，C 口地址为 8002H，控制寄存器地址为 8003H。

2. 判断有无闭合键的子程序

判断有无闭合键的子程序为 KS，以供在键盘扫描程序中调用。执行 KS 子程序的结果是，有闭合键，则（A）≠0；无闭合键，则（A）=0。程序如下：

```
KS: MOV DPTR, #8000H
    MOV A, #00H          ; A 口送 00H
    MOVX@ DPTR, A
    INC DPTR
    INC DPTR             ; 建立 C 口地址
    MOVX A, @ DPTR       ; 读 C 口
    CPL A                ; A 取反，若无键按下，则全为 0
    ANL A, #0FH          ; 屏蔽 A 高半字节
    RET
```

3. 键盘扫描程序

在单片机应用系统中常是键盘和显示器同时存在，因此可以把键盘程序和显示程序配合起来使用，即把显示程序作为键盘程序中的一个延时子程序使用。这样既不耽误显示驱动，又可以起到键盘定时扫描的作用。假定本系统中显示器驱动程序为 DIR，执行时间约为 6ms。键盘扫描程序如下，程序中 R2 为扫描码寄存器，R4 为行计数器。

```
KEY:   ACALL  KS
       JNZ    LK1
       ACALL  DIR
       AJMP   KEY
LK1:   ACALL  DIR
       ACALL  DIR
       ACALL  KS
       JNZ    LK2
       ACALL  DIR
       AJMP   KEY
LK2:   MOV  R2, #FEH
       MOV  R4, #00H
LK4:   MOV DPTR, #8000H
       MOV  A, R2
       MOV  @DPTR, A
       INC DPTR
       INC DPTR
       MOVX A, @DPTR
       JB   ACC.0, LONE
       MOV  A, #00H
       ALMP  LKP
LONE:  JB   ACC.1, LTWO
       MOV  A, #08H
       ALMP  LKP
LTWO:  JB   ACC.2, LTHR
       MOV  A, #10H
       ALMP  LKP
LTHR:  JB   ACC.3, NEXT
       MOV  A, #18H
LKP:   ADD  A, R4
       PUSH  ACC
LK3:   ACALL  DIR
       ACALL  KS
       JNZ    LK3
```

```
        POP     ACC
        RET
NEXT：INC   R4
        MOV   A，R2
        JNB   ACC.7，KND
        RL     A
        MOV   R2，A
        AJMP   LK4
KND：AJMP   KEY
```

键盘扫描程序的运行结果是把闭合键的键码放在累加器 A 中。键码的计算公式为

键码 = 列状态码起始值 + 行号

7.5 LED 显示接口技术

在小型控制装置和数字化仪器仪表中，往往只要几个简单的数字显示或字符状态便可满足现场的需求，而显示数码的 LED 因其成本低廉、配置灵活，与计算机接口方便等特点在小型微机控制系统中得到极为广泛的应用。

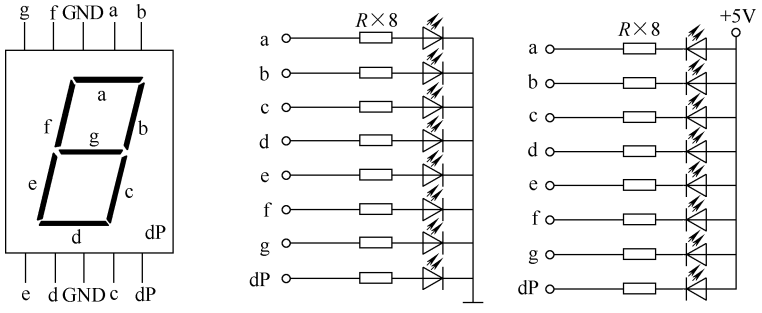
本节将讨论 LED 显示器及其接口电路与相应程序。

7.5.1 LED 显示器概述

LED 是利用 PN 结把电能转换成光能的固体发光器件，根据制造材料的不同可以发出红、黄、绿、白等不同色彩的可见光来。LED 的伏安特性类似普通二极管，正向压降约为 2V 左右，工作电流一般在 10 ~ 20mA 之间较为合适。

1. 结构

由条形 LED 组成“8”字形的 LED 显示器，也称数码管。通过数码管中 LED 的亮暗组合，可以显示多种数字、字母以及其他符号。数码管有 7 段数码管和 8 段数码管之分。7 段数码管由 7 个 LED 组成，而 8 段数码管则是在 7 段发光二极管的基础上再加一个圆点形 LED，用于显示小数点，其外形如图 7-30a 所示。



a) 符号和引脚排列　b) 共阴极接法　c) 共阳极接法

图 7-30　数码管结构图

8 段 LED 显示器有共阴极和共阳极两种结构，内部结构原理如图 7-30b、c 所示。

共阴极 LED 的所有发光管的阴极并接成公共端 COM，当共阴极 LED 的 COM 端接地，则某个 LED 的阳极加上高电平时，则该管有电流流过因而点亮发光；而共阳极 LED 的所有发光管的阳极并接成公共端 COM。当共阳极 LED 的 COM 端接高电平，则某个发光管的阴极加上低电平时，则该管有电流流过因而点亮发光。

2. 原理

（1）段码

所谓段码就是为数码管显示提供的各段状态组合，即字形代码。7 段数码管的段码为 7 位，8 段数码管的段码为 8 位，用一个字节即可表示。在段码字节中代码位与各段 LED 的对应关系如下：

段码	D7	D6	D5	D4	D3	D2	D1	D0
段名	dp	g	f	e	d	c	b	a

码段的数值除了与要显示的数字或字符有关外，还与 LED 显示器公共引脚的接法有关（共阳接法和共阴接法）。以 8 段数码管为例，显示十六进制的码段值见表 7-12。

表 7-12　十六进制数码段表

显示	十六进制代码		显示	十六进制代码	
	共阴极	共阳极		共阴极	共阳极
0	3FH	C0H	9	6FH	90H
1	06H	F9H	A	77H	88H
2	5BH	A4H	b	7CH	83H
3	4FH	B0H	C	39H	C6H
4	66H	99H	d	5EH	A1H
5	6DH	92H	E	79H	86H
6	7DH	82H	F	71H	8EH
7	07H	F8H	H	76H	89H
8	7FH	80H	P	F3H	8CH

（2）LED 显示器显示方法

1）静态显示。静态显示，是由单片机一次输出显示后，就能保持该显示结果，直到下次送新的显示模型为止。

这种显示占用机时少、显示可靠，但它使用元器件多，且电路比较复杂，因而成本比较高。随着大规模集成电路的发展，目前已经研制出具有多种功能的显示器件，如锁存器、译码器、驱动器、显示器四位一体的显示器件，用起来比较方便。

2）动态显示。动态显示，就是微机定时地对显示器件扫描。在这种方法中，显示器件分时工作，每次只能有一个器件显示，但由于人视觉的暂留现象，所以仍感觉所有的器件都在显示。例如，许多单片机的开发系统及仿真器上的六位显示器就采用这类显示方法。

其优点是使用硬件少、因而价格低，但占用机时长，只要微机不执行显示程序，就立刻停止显示。

7.5.2 LED 显示器接口

1. LED 静态显示接口技术

在智能化仪器及微型机控制系统中，为了使操作者随时都能监视生产过程，而又不占有 CPU 的很多时间，人们更喜欢采用静态显示电路。它主要是用于 BCD 码显示。图 7-31 所示为 6 位 BCD 码静态显示电路原理图。

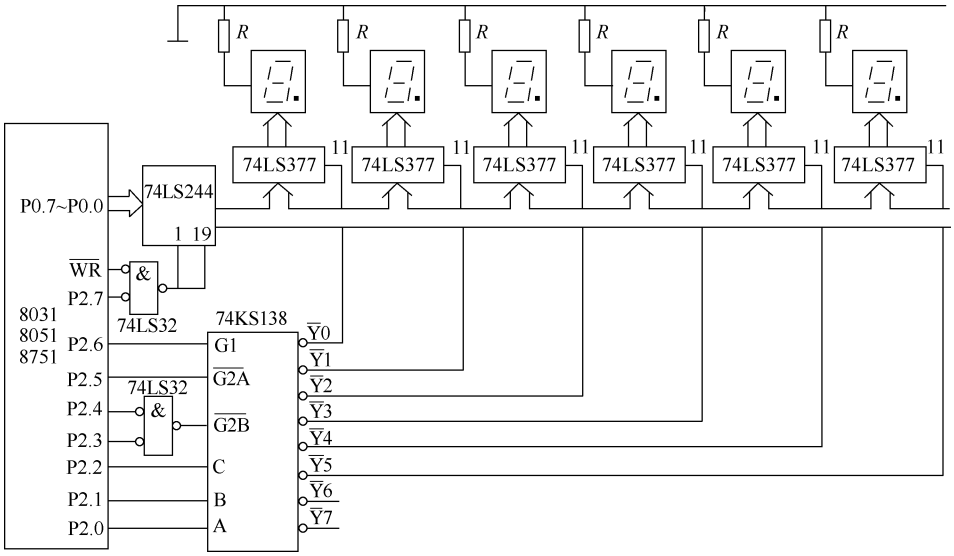


图 7-31 用锁存器连接的 6 位 BCD 码静态显示电路原理图

（1）电路分析

① 如图 7-31 所示，74LS244 为总线驱动器，6 位数字显示共用同一组总线。

② 每个 LED 显示器均配有一个锁存器（74LS377），用它来锁存待显示的数据。当被显示的数据从数据总线经 74LS244 传送到各锁存器的输入端后，到底哪一个锁存器选通，取决于地址译码器 74LS138 各输出位的状态。

③ 总线驱动器 74LS244 由$\overline{WR}$和 P2.7 控制，当$\overline{WR}$和 P2.7 同时为低电平时，74LS244 打开，将数据总线上的数据传送到各个显示器的锁存器 74LS377 上。

④ 在图 7-31 所示的显示系统中，地址的确定见表 7-13。

表 7-13 地址的确定

地址线	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
口线	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0
显示 1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
显示 2	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0
显示 3	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0
显示 4	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0
显示 5	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0
显示 6	0	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0

从左到右各显示位的地址依次为 4000H、4100H、4200H、4300H、4400H、4500H。

静态显示电路的最大优点是只要不送新的数据，则显示值不变，且微机不用像动态显示那样不间断地扫描，因而节省了大量机时，适用于工业过程控制及智能化仪器中。

（2）程序设计

根据图 7-31 所示电路可写出 6 位静态显示程序，由于接口电路中显示模型输出地址和位选信号可一次选中，故只要一次输出即可显示一位。

```
        ORG   0000H
SIXDPY: MOV   R0, #30H              ; 建立显示缓冲区地址指针
        MOV   33H, #03H             ; 设置循环次数
        MOV   DPTR, #4000H          ; 指向最左边一位
LOOP:   MOV   A, @R0                ; 取 BCD 码高 4 位送去显示
        ANL   A, #0F0H
        RR    A
        RR    A
        RR    A
        RR    A
        ADD   A, #10H
        MOVC  A, @A+PC              ; 取字形码
        MOVX  @DPTR, A
        MOV   A, @R0
        ANL   A, #0FH
        INC   DPH
        ADD   A, #08H
        MOVC  A, @A+PC
        MOVX  @DPTR, A
        INC   R0
        INC   DPH
        DJNZ  33H, LOOP
        RET
SEGTAB DB  3FH, 06H, 5BH, 4FH       ; 0, 1, 2, 3
       DB  66H, 6D, 7DH, 07H        ; 4, 5, 6, 7
       DB  7FH, 6FH, 77H, 7CH       ; 8, 9, A, b
       DB  39H, 5EH, 79H, 71H       ; c, d, E, F
       DB  80H, 40H, 00H, 73H       ; ., -, 空, P
```

2. LED 动态显示接口技术

目前国内生产的许多单片单板机，包括一些开发系统及仿真器，均采用动态显示。这种显示方法的最大优点就是电路简单、价格便宜，适合于大批量生产。图 7-32 所示为单板机或仿真器中常用的一种并行 6 位动态显示电路。

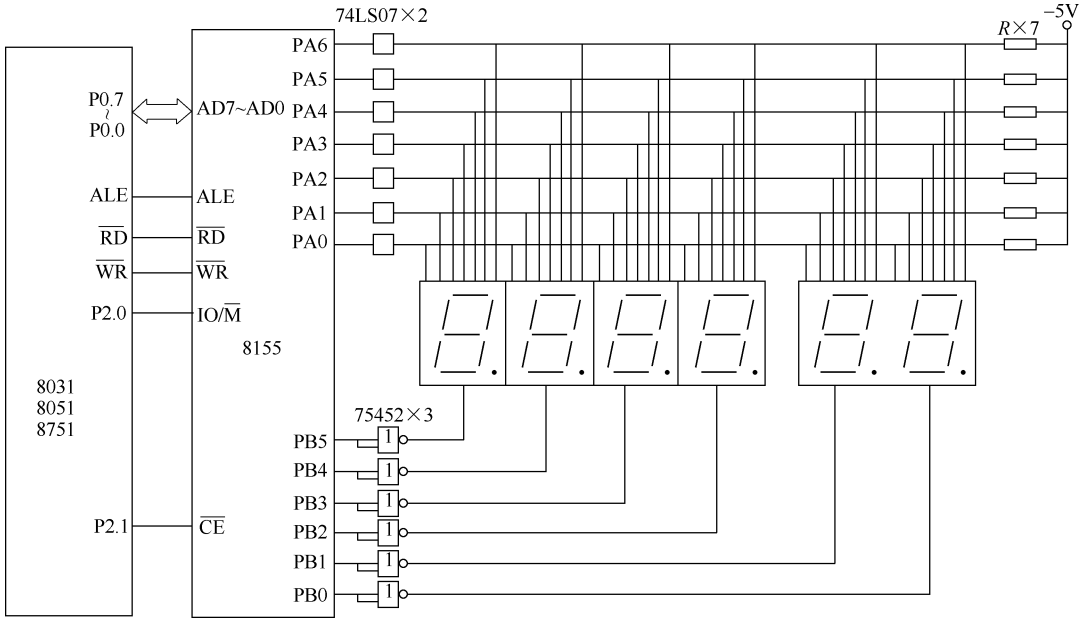


图 7-32　6 位动态显示电路

（1）电路分析

① 用 8155 的 PA 口输出显示码，PB 口用来输出位选码。

② 74LS07 为 6 位驱动器，它为 LED 提供一定的驱动电流，由于一片 74LS07 只有 6 个驱动器，故七段数码管需要 2 片进行驱动。

③ 8155 的 PB 口经 75452 缓冲器/驱动器反相后，作为位控信号。

④ 75452 内部包括两个缓冲器/驱动器，它们各有两个输入端。需要 3 片为 6 位数码管提供位选信号。

（2）显示原理

设显示缓冲区为 30H～35H，则完成对 8155 初始化后取出一位要显示的数（十六进制数），利用软件译码的方法求出待显示的数对应的七段显示码，然后由 PA 口输出，并经过 74LS07 驱动器放大后送到各显示器的数据总线上。到底哪一位数码管显示，主要取决于位选码。只有位选信号 $PB_i = 1$（经驱动器变作低电平）时，对应位上的选中段才发光。若将各位从左至右依次进行显示，每个数码管连续显示 1ms，显示完最后一位数后，再重复上述过程，这样，人们看到的就好像 6 位数“同时”显示一样。

这种动态 LED 显示接口由于各个数码管共用一个段码输出口，分时轮流通电，从而大大简化了硬件线路，降低了成本。

（3）设计流程（见图 7-33）

（4）程序设计

```
        ORG   0000H
DISPLY: MOV     A，#30H          ；设 8155  A 口、B 口均为输出方式
        MOV   DPTR，#FD00H
        MOV   @ DPTR，A
```

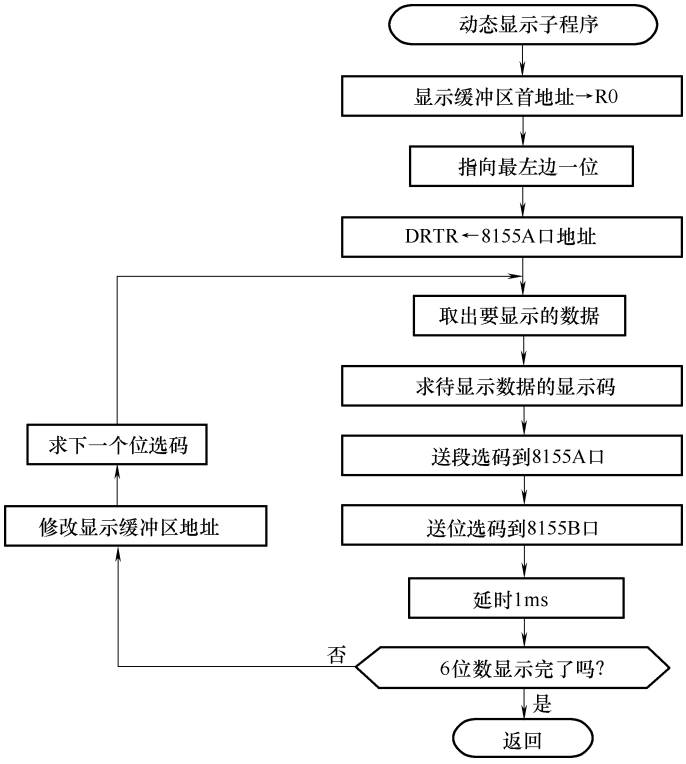


图 7-33　动态显示子程序流程图

```
        MOV   R0，#30H          ；显示缓冲区首地址送 R0
        MOV   R2，#20H          ；位选码指向最左一位
DISPY1：MOV   A，@R0            ；取出要显示的数
        MOV   DPTR，#SEGTAB     ；指向换码表首址
        MOVC  A，@A+DPTR        ；取出显示码
        MOV   DPTR，#0FD01H     ；从 8155 A 口输出显示码
        MOV   @DPTR，A
        MOV   A，R2             ；从 8155 B 口输出位选码
        INC   DPTR
        MOV   @DPTR，A
        ACALL D1MS              ；延时 1ms
        MOV   A，R2
        JNB   ACC.0，DISPY2     ；6 位未显示完，继续显示
        RET
DISPY2：INC   R0                ；求下一位待显示的数的存放地址
        MOV   A，R2             ；求下一个位选码
        RRA   A
        MOV   R2，A
        AJMP  DISPY1
```

```
D1MS:   MOV   R3, #7DH          ; 延时 1ms
DL1:    NOP
        NOP
        DJNZ  R3, DL1
        RET
SEGTAB  :  DB  3FH              ; 对应于字符 0
           DB  06H              ; 对应于字符 1
           DB  5BH              ; 对应于字符 2
           DB  4FH              ; 对应于字符 3
           DB  66H              ; 对应于字符 4
           DB  6DH              ; 对应于字符 5
           DB  7DH              ; 对应于字符 6
           DB  07H              ; 对应于字符 7
           DB  7FH              ; 对应于字符 8
           DB  67H              ; 对应于字符 9
           DB  77H              ; 对应于字符 A
           DB  7CH              ; 对应于字符 B
           DB  39H              ; 对应于字符 C
           DB  5EH              ; 对应于字符 D
           DB  79H              ; 对应于字符 E
           DB  71H              ; 对应于字符 F
```

本 章 小 结

并行扩展方式一般采用总线并行扩展，即数据传送由数据总线完成，地址总线负责外围设备的寻址，而控制总线来完成传输过程中的传输控制。

在并行总线扩展中，主要介绍了两种寻址方式：线选法与译码法。该方式的扩展实例是数据存储器与程序存储器的扩展。单片机常采用数据存储与程序存储相互独立的哈佛结构体系，故存储器的扩展要分别考虑。通过本章的学习要学会画总线扩展电路图，对常用的程序存储器及数据存储器的型号和特性要熟练掌握。

在 I/O 口并行扩展中，概略地讲了简单 I/O 口并行扩展所用 TTL 芯片；重点讲述了 I/O 口扩展时所用的美国 intel 公司的可编程芯片：8255A 和 8155。

8255A 是一个用在美国 intel 公司的微型计算机系统中的可编程外围接口设备（PPI），与美国 intel 公司的所有微处理器都兼容。它主要是作为外围设备和微型计算机总线间的 I/O 组成中的接口。由于 8255A 可以通过软件来设置芯片的工作方式，因此用 8255A 连接外部设备时，通常不需要再附加外部电路，给使用带来很大的方便。

8155 是 intel 公司的另一种多功能可编程外围扩展接口芯片，它也有三个可编程 I/O 端口（端口 A、B、C），与 8255A 的区别在于 PC 口是 6 位，一个可编程 14bit 定时/计数器和 256B 的 RAM，能方便地进行 I/O 扩展和 RAM 扩展，芯片采用 40 针双列直插式封装。对于

这部分内容，要掌握芯片的引脚特性、内部结构以及工作方式的特点等。

键盘可分为独立式键盘和矩阵式（也叫行列式）键盘两种，MCS-51 可方便地与这两种键盘接口。独立式键盘配置灵活，软件识别简单，但占用 I/O 口线多，不适合较多按键的键盘。矩阵式键盘占用 I/O 口线少，节省资源。矩阵式键盘一般采用扫描方式识别按键，软件设计相对复杂，但只要学会调用本章实例所提供的子程序，用起来就很简单。使用机械式按键时，应注意去抖。

LED 数码显示器有共阴极和共阳极两种结构，共阴极 LED 的所有发光管的阴极并接成公共端 COM；而共阳极 LED 的所有发光管的阳极并接成公共端 COM。在进行数码和符号显示过程中，共阳接法和共阴接法的数码管的码段是不一样的，需要依据其结构进行分析。显示方式分为两种：动态显示和静态显示。需要掌握动态显示和静态显示电路的设计方法与程序设计方法。

习　题

1. 并行总线包含哪几部分？各有什么特点？
2. 只读存储器包含哪几类？各有什么特点？
3. 常用的芯片选择方法（即寻址方法）有哪两种？这两种方法各有什么特点？
4. 2716 芯片有哪些工作方式？
5. I/O 编址技术包含哪两类？各有什么特点？
6. 8255 的 I/O 控制方式有哪些？各有什么特点？
7. 8255 和 8155 分别有几种工作方式，其特点是什么？
8. 机械式按键组成的键盘应如何消除按键抖动？独立式按键和矩阵式按键分别具有什么特点？
9. 在用共阳极数码管显示的电路中，如果直接将共阳极数码管换成共阴极数码管，能否正常显示？为什么？应采取什么措施？
10. LED 显示器静态显示和动态显示分别具有什么特点？实际设计时应如何选择使用？

第 8 章　单片机 A-D 及 D-A 转换接口

【学习目的】

1. 了解微型计算机系统模拟量输入输出接口技术。
2. 掌握模拟量输入接口（A-D 转换）结构、原理及应用。
3. 模拟量输出接口（D-A 转换）结构、原理及应用。

在微型机控制系统与智能化仪器中，被测物理量（如温度、压力、流量、液位、成分、位移、速度等）都是模拟量，而计算机只能接收数字量，所以在上述系统中，必须首先把传感器（有时需要通过变送器）输出的模拟量转换成数字量，然后再送到计算机进行数据处理，以便实现控制或进行显示。能够变模拟量为数字量的器件称作模-数转换器（简称 A-D 转换器）。

同理，经计算机处理后的数字量输出，不能直接用以控制执行机构。这是由于大多数执行机构，如电动执行机构、气动执行机构以及直流电机等，只能接收模拟量。为此，还必须把数字量变成模拟量，即完成数-模转换（简称 D-A 转换）。

由此可得

- A-D 转换器：模拟量→数字量的器件。
- D-A 转换器：数字量→模拟量的器件。

8.1　模拟量输入接口（A-D 转换）

8.1.1　A-D 转换器概述

模拟量输入通道的任务是将模拟量转换成数字量。能够完成这一任务的器件，称之为 A-D 转换器，简称 A-D 转换器。和 D-A 转换器一样，A-D 转换器也做成单片型双列直插式封装芯片。

1. 分类

（1）按转换原理分

① 计数器式：结构简单、转换速度慢，现在基本不用。

② 双积分式：精度高、抗干扰能力强，但速度较慢，常用于数字电压表。

③ 逐次逼近型：速度较快、结果比较简单，是计算机应用最多的一种。

④ 并行 A-D：速度最快，但结构复杂，价格高，一般用于军事。

⑤ V/F 变换：结构简单、成本低，适合于远程应用。

（2）按位数分

① 8 位，10 位、12 位、16 位等。

② 位数越高，其分辨率也越高，但价格也越贵。

（3）按结构分

① 单一的 A-D 转换器（如 ADC 0801、AD673 等）。

② 内含多路开关的 A-D 转换器（如 ADC0809，AD 7581 均带有 8 路多路开关）。

③ 多功能 A-D 转换芯片，AD 363 就是一种典型芯片。其内部具有 16 路多路开关、数据放大器、采样-保持器及 12 位 A-D 转换器，其本身就已构成一个完整的数据采集系统。

（4）按输出方式

① 串行 A-D 转换器，如 MAX195。

② 并行 A-D 转换器。

2. A-D 转换器的主要技术指标

（1）分辨率。以输出二进制的位数表示分辨率，位数越多，误差越小，转换精度越高。

（2）相对精度。相对精度是指实际的各个转换点偏理想特性的误差。在理想的情况下，所有的转换点应当在一条直线上。

（3）转换速度。它是指完成一次转换所需的时间。转换时间是指由启动转换命令到转换结束信号开始有效的时间间隔。

（4）电源抑制。在输入电压不变的前提下，当转换电路的供电电源电压发生变化时，对输出也会产生影响。这种影响可用输出数字量的绝对变化量来表示。

此外，尚有功率损耗、温度系数、输入模拟电压范围以及输出数字信号的逻辑电平等指标。

3. 通常使用的逐次逼近式典型 A-D 转换器芯片

ADC0801 ~ ADC0805 型 8 位 MOS 型 A-D 转换器，美国国家半导体公司生产。它是目前最流行的中速廉价型产品，片内有三态数据输出锁存器，单通道输入，转换时间约 100μs 左右。

ADC0808/0809 型 8 位 MOS 型 A-D 转换器，可实现 8 路模拟信号的分时采集，片内有 8 路模拟选通开关，以及相应的通道地址锁存用译码电路，其转换时间为 100μs 左右。

ADC0816/0817。这类产品除输入通道数增加至 16 个以外，其他性能与 ADC0808/0809 型基本相同。

8.1.2　8 位 A-D 转换器芯片及与 80C51 单片机的接口

8 位 A-D 转换芯片以 ADC0809 为例进行说明。8 位 A-D 转换芯片以 ADC0809 为例进行说明。ADC0809 是 ADC08 × × 系列中的一员，ADC08 × × 是美国国家半导体公司的一个A-D 转换芯片系列，具有多种芯片型号，其中包括 8 位 8 通道 CMOS 型芯片 ADC0808 和 ADC0809 以及 8 位 16 通道 CMOS 型芯片 ADC0816 和 ADC0817 等。

1. ADC0809 芯片

（1）结构及原理

ADC0808/0809 都是带有 8 位 A-D 转换器、8 路多路开关，以及与微型计算机兼容的控制逻辑的 CMOS 组件，其转换方法为逐次逼近型。其原理框图如图 8-1 所示。

图 8-1 所示的多路开关可选通 8 个模拟通道，允许 8 路模拟量分时输入，共用一个 A-D

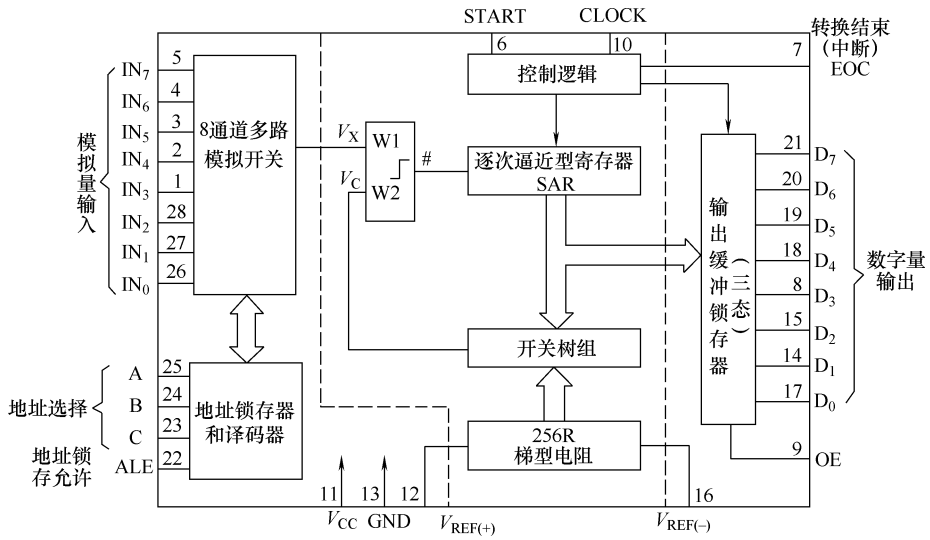


图 8-1 ADC0809 原理框图

转换芯片进行转换。地址锁存与译码电路完成对 A、B、C 3 个地址位进行锁存和译码，其译码输出用于通道选择。8 位 A-D 转换器是逐次逼近式，由控制与时序电路、逐次逼近寄存器、树状开关以及 256R 电阻阶梯网络等组成。输出锁存器用于存放和输出转换得到的数字量。

（2）芯片引脚

ADC0809 芯片为 28 引脚双列直插式封装，其引脚排列如图 8-2 所示。

各引脚功能如下：

$IN_7 \sim IN_0$：8 个模拟量输入端。ADC0809 对输入模拟量的要求主要有：信号单极性，电压范围为 0～5V。

START：启动信号。A-D 转换启动脉冲输入端，输入一个正脉冲（至少 100ns 宽）使其启动（脉冲上升沿使 0809 复位，下降沿启动 A-D 转换）。

EOC：转换结束信号。当 A-D 转换结束后，发出一个正脉冲，表示 A-D 转换完毕。此信号可用作 A-D 转换是否结束的检测信号，或向 CPU 申请中断的信号。

OE：输出允许信号。当此信号被选中时，允许从 A-D 转换器的锁存器中读取数字量。此信号可作为 ADC0808/0809 的片选信号，高电平有效。当 A-D 转换结束时，此端输入一个高电平，才能打开输出三态门，输出数字量。

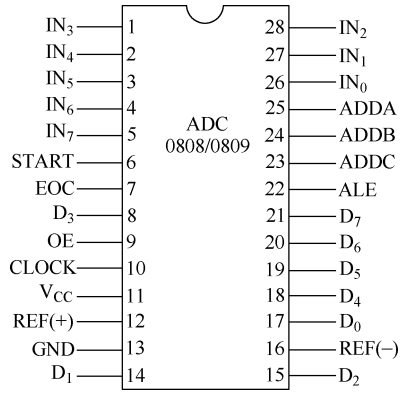


图 8-2 ADC0809 引脚排列

CLOCK：实时时钟，可通过外接 RC 电路改变时钟频率。

ALE：地址锁存允许，高电平有效。当 ALE 为高电平时，允许 C、B、A 所示的通道被选中，并把该通道的模拟量接入 A-D 转换器。

C、B、A：通道号选择端子。C 为最高位，A 为最低位，选择方法见表 8-1。

表 8-1　通道号选择表

ALE 锁存信号	C	B	A	接通的通道
1	0	0	0	IN0
1	0	0	1	IN1
1	0	1	0	IN2
1	0	1	1	IN3
1	1	0	0	IN4
1	1	0	1	IN5
1	1	1	0	IN6
1	1	1	1	IN7

$D_7 \sim D_0$：数字量输出端。为三态缓冲输出形式，可以与单片机的数据线直接相连。D_0 为最低位，D_7为最高位

$V_{REF(+)}$、$V_{REF(-)}$：参考电压端子。用以提供 D-A 转换器权电阻的标准电平。对于一般单极性模拟量输入信号，$V_{REF(+)} = +5V$，$V_{REF(-)} = 0V$。

V_{CC}：电源端子，接 +5V。

GND：接地端。

（3）ADC0808/0809 的技术指标

① 单一电源，+5V 供电，模拟输入范围为 0 ~ 5V。

② 分辨率为 8 位。

③ 最大不可调误差：ADC0808 < ±1/2LSB$\left(1\text{LSB} = \frac{1}{256}\right)$。

ADC0809 < ±1LSB

④ 功耗为 15mW。

⑤ 转换速度取决于芯片的时钟频率。时钟频率范围为 10 ~ 1280kHz，当 CLOCK 等于 500kHz 时，转换速度为 128μs。

⑥ 可锁存三态输出，输出与 TTL 兼容。

⑦ 无需进行零位及满量程调整。

⑧ 温度范围为 -40 ~ +85℃。

总之，ADC0808/0809 具有较高的转换速度和精度、受温度影响较小、能较长时间保证精度、重现性好、功耗较低，且具有 8 路模拟开关，所以用于过程控制是比较理想的器件。

2. ADC0809 的接口技术

（1）模拟量输入信号的连接

A-D 转换器 ADC0809 所要求接收的模拟量大都为 0 ~ 5V 的标准电压信号，用户可通过改变外接线路来改变量程。

（2）数字量输出引脚的连接

A-D 转换器数字输出引脚和单片机的连接方法与其内部结构有关。对于内部未含输出锁存器的 A-D 转换器来说，一般通过锁存器或 I/O 接口与微机相连，常用的接口及锁存器有美国 intel 公司的 8155、8255、8243 以及 74LS273、74LS373、8212 等。ADC0809 转换器

内部含有数据输出锁存器时，可直接与微型机相连。有时为了增加控制功能，也采用I/O接口连接。

（3）A-D转换器的启动方式

任何一个A-D转换器在开始转换前，都必须加一个启动信号，才能开始工作。芯片不同，要求的启动方式也不同。一般分脉冲启动和电平启动两种。脉冲启动型芯片，只要在启动转换输入引脚加一个启动脉冲即可，如ADC0809、ADC80、AD574A等均属于脉冲启动转换芯片，往往用$\overline{WR}$及地址译码器的输出Y_i经过一定的逻辑电路进行控制。

（4）转换结束信号的处理方法

微型机检查判断A-D转换结束的方法有以下三种：

① 中断方式，将转换结束标志信号接到微型机的中断申请引脚（如$\overline{INT0}$、$\overline{INT1}$）或允许中断的I/O接口的相应引脚上（8255）。当转换结束时，即提出中断申请，微机响应后，在中断服务程序中读取数据。这种方法使A-D转换器与微机的工作同时进行，因而节省机时，常用于实时性要求比较强或多参数的数据采集系统。

② 查询方式，把转换结束信号经三态门送到CPU数据总线或I/O接口的某一位上，微型机向A-D转换器发出启动信号后，便开始查询A-D转换是否结束，一旦查询到A-D转换结束，则读出结果数据。这种方法的程序设计比较简单，且实时性也比较强，是应用最多的一种方法。特别是在单片机系统中，因为它具有很强的位处理功能，因而，使用起来更加方便。

③ 软件延时方法，其具体作法是，微型机启动A-D转换后，就根据转换芯片完成转换所需要的时间，调用一段软件延时程序（为保险起见，通常延时时间略大于A-D转换过程所需的时间）；延时程序执行完以后，A-D转换也已完成，即可读出结果数据。这种方法可靠性比较高，不必增加硬件连线，但占用CPU的机时较多，多用在CPU处理任务较少的系统中。

（5）参考电平的连接

在A-D转换器中，参考电平的作用是供给其内部D-A转换器的标准电源，它直接关系到A-D转换的精度，因而对该电源的要求比较高，一般要求由稳压电源供电。不同的A-D转换器，参考电源的提供方法也不一样。

（6）时钟的连接

A-D转换器的另一个重要连接信号是时钟，其频率是决定芯片转换速度的基准。整个A-D转换过程都是在时钟作用下完成的（见图2-16所示）。

时钟的提供方法：

① 芯片内部提供，经常外接RC电路来提供。

② 一种是由外部时钟提供，提供如下方法：

◆ 可以用单独的振荡器；

◆ 更多的则是用CPU时钟经分频后，送至A-D转换器的相应时钟端子。

（7）接地问题

模拟地和数字地要分别连接。然后，再把这两种“地”用一根导线连接起来。

3. ADC0809与80C51接口

A-D转换器芯片与单片机的接口是数字量输入接口，其原理与并行I/O输入接口相同，

需要有三态缓冲功能，即 A-D 转换器芯片必须通过三态门“挂上”数据总线。ADC0809 芯片已具有三态输出功能，因此，ADC0809 与 80C51 的接口比较直接，如图 8-3 所示。

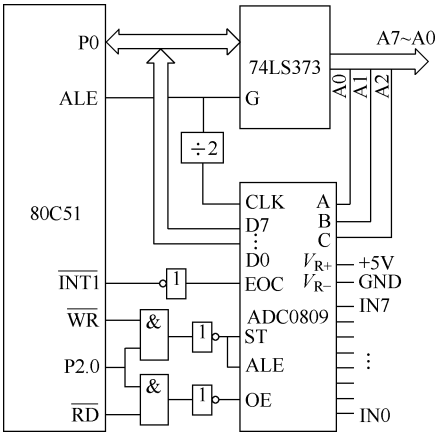


图 8-3　ADC0809 与 80C51 接口电路图

如图 8-3 中所示，8 路模拟通道选择信号 A、B、C 分别接最低 3 位地址 A0、A1、A2（即 P0.0、P0.1、P0.2），而地址锁存允许信号 ALE 由 P2.0 控制，则 8 路模拟通道的地址为 FEF8H ~ FEFFH。因转换结束信号 EOC 高电平有效，所以经反相器与 $\overline{\text{INT1}}$ 引脚相连。在进行 A-D 转换之前，必须先用“MOVX　@DPTR，A”指令启动 A-D 转换。此时，$\overline{\text{WR}}$ = 0，P2.0 也是低电平，于是 A-D 转换开始。当 A-D 转换完成后，EOC 变为高电平，随之 80C51 单片机的 $\overline{\text{INT1}}$ 变成低电平，向单片机提出中断申请。若中断得到响应，便进行读操作（$\overline{\text{RD}}$ = 0，P2.0 = 0），读出 A-D 转换结果。

【例 8-1】　设有一个 8 路巡回检测系统，其采样数据依次存放在外部 RAM A0H ~ A7H 单元中，按图 8-3 的接口电路，ADC0809 的 8 个通道地址为 FEF8H ~ FEFFH。试进行程序设计。

执行一条“MOVX　@DPTR，A”指令，产生 $\overline{\text{WR}}$ 信号，使 ALE 和 START 有效，就可以启动一次 A-D 转换。但一次启动只能进行一个通道的转换，8 个通道的 A-D 转换需按通道顺序逐个进行。为此，在程序中应当有改变通道号的指令，并且每改变一次就执行一次启动 A-D 转换指令。据此数据采样的参考程序如下：

```
        ORG   0000H              ; 主程序入口地址
        AJMP  MAIN               ; 跳转主程序
        ORG   0013H              ; INT1中断入口地址
        AJMP  INT1               ; 跳转中断服务程序
        ORG   0030H
MAIN:   MOV   R0, #A0H           ; 数据暂存区首址
        MOV   R2, #08H           ; 8 路计数初值
        SETB  IT1                ; INT1边沿触发
        SETB  EA                 ; 开中断
        SETB  EX1                ; 允许INT1中断
        MOV   DPTR, #FEF8H       ; 通道首地址
        MOVX  @DPTR, A           ; 启动 A-D 转换
HERE:   SJMP  HERE               ; 等待中断
        ORG   0060H
INT1:   MOVX  A, @DPTR           ; 读 A-D 转换结果
        MOV   @R0, A             ; 存数
        INC   DPTR               ; 更新通道
        INC   R0                 ; 更新暂存单元
```

```
        DJNZ   R2,    NEXT      ；巡回未完继续
        RETI                    ；返回
NEXT:   MOVX   @DPTR, A         ；启动下一通道 A-D 转换
        RETI
```

8.1.3 高于 8 位 A-D 转换器芯片及与 80C51 单片机的接口

和 D-A 转换器一样，在一些微机控制系统中，往往精度要求比较高，因此需要更多位数的 A-D 转换器，如 10 位、12 位 A-D 转换器等。由于位数不同，所以其与 CPU 的接口及程序设计方法也不同。

下边主要以 12 位 A-D 转换器 AD574 为例讲一下高于 8 位的 A-D 转换器与 8 位单片机的接口方法。

AD574A 是美国模拟器件公司的产品，由于芯片内有三态数据输出缓冲器，所以接口时无需外加三态缓冲器。由于内部的缓冲器为 12 位，所以其转换数据既可以一次读出，也可以分两次读出。AD574A 与 80C51 的接线如图 8-4 所示。

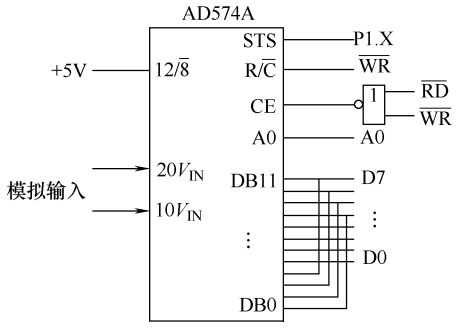


图 8-4 AD574A 与 80C51 的连线

AD574A 有两个模拟输入端，其中 $10V_{IN}$ 的电压范围是 0～10V，$20V_{IN}$ 的电压范围是 0～20V。在 80C51 系统中使用 AD574A 芯片，其转换数据应该分两次读出，并按高 8 位和低 4 位分次。分次读出由 A0 控制，该引脚一般接地址线的最低位 A0。A0 = 0 时，读高 8 位；A0 = 1 时，读低 4 位。其他的相关控制信号如下：

R/$\overline{C}$：读/启动转换信号。R/$\overline{C}$ = 0 时为启动转换信号；R/$\overline{C}$ = 1 时为读信号。接口时可通过 80C51 的写命令$\overline{WR}$进行控制。

STS：转换结束信号。转换期间为高电平，转换结束时下跳为低电平。转换结束信号为输出的状态信号，供单片机查询或中断使用。图 8-4 中，STS 与 P1 口的一根口线相连，这是使用查询方法读取转换数据。

12/$\overline{8}$：输出位数选择信号。12/$\overline{8}$ = 1 时，为 12 位输出，12/$\overline{8}$ = 0 时，为 8 位输出。使用时可接电源或地。

CE：允许信号，高电平有效。参与启动转换和读数据的控制。

8.2 模拟量输出接口（D-A 转换）

8.2.1 D-A 转换器概述

模拟量输出通道主要完成数字量到模拟量的转换，简称 D-A 转换。D-A 转换器的输出多数为电流形式，如 DAC0832、AD7522 等。有些芯片内部设有放大器，直接输出电压信号，如 AD558、AD7224 等。电压输出型又有单极性输出和双极性输出两种形式。

1. 分类

D-A 转换器输入的是数字量，经转换后输出的是模拟量。D-A 转换器集成电路芯片种类很多。

1）按输入的二进制数的位数分类，有 8 位、10 位、12 位和 16 位等。

2）按输出是电流还是电压分类，分为电压输出器件和电流输出器件。

2. D-A 转换器的主要技术指标

有关 D-A 转换器的技术性能指标很多，如绝对精度、相对精度、线性度、输出电压范围、温度系数、输入数字代码种类（二进制或 BCD 码）等。

3. D-A 转换器与接口有关的技术性能指标

1）分辨率。D-A 转换的分辨率是指最小输出电压（对应的输入二进制数为 1）与最大输出电压（对应的输入二进制数的所有位全为 1）之比。例如，8 位数的分辨率为 1/256≈0.004，10 位数分辨率为 1/1024，约等于 0.001。由此可见数字量位数越多，分辨率也就越高。分辨率通常用数字输入信号的位数表示，有 8 位、10 位、12 位等。

2）建立时间。也称稳定时间，它是指从数字量输入到建立稳定的输出电流的时间，是描述 D-A 转换速率的一个重要参数。

3）转换精度。由于转换器内部的误差等原因，当送一个确定的数字量给 D-A 转换器后，它的实际输出值与该数值应产生的理想输出值之间会有一定的误差，它就是 D-A 转换器的精度。

8.2.2　8 位 D-A 转换器芯片及与 80C51 单片机的接口

1. DAC0832 芯片

DAC 0832 是美国数据公司的 8 位 D-A 转换器，与微处理器完全兼容。器件采用先进的 CMOS 工艺，因此功耗低，输出漏电流误差较小；而其间的特殊的电路结构可与 TTL 逻辑输入电平兼容；采用 20 引脚、双列直插式集成电路芯片；主要参数为，分辨率 8 位，电流稳定时间 1μs，电流输出，与 TTL 电平兼容，功耗 20mW。

（1）结构及原理

DAC 0832 D-A 转换器的内部，具有两级输入数据缓冲器和一个 R-2R T 形电阻网络，其原理框图如图 8-5 所示。

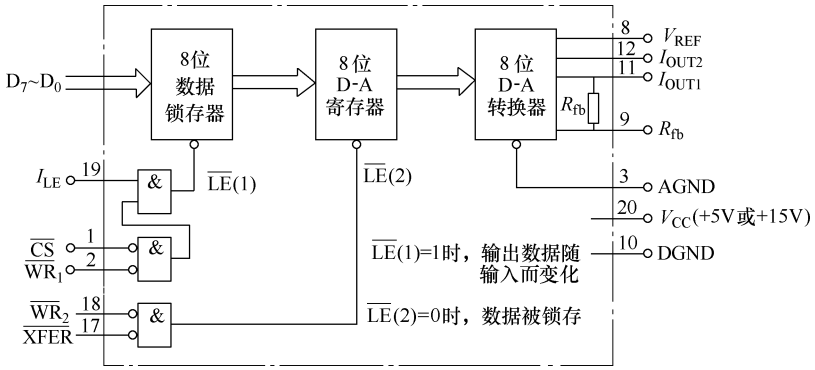


图 8-5　DAC 0832 原理框图

DAC0832 内部结构框图如图 8-6 所示。从图中可见，在 DAC0832 中有两个数据缓冲器：输入寄存器和 DAC 寄存器。其控制端分别受 I_{LE}、$\overline{CS}$、$\overline{WR_1}$和$\overline{WR_2}$、$\overline{XFER}$的控制。

（2）芯片引脚

DAC0832 芯片为 20 引脚双列直插式封装，其引脚排列如图 8-6 所示。

各引脚功能如下：

$D_7 \sim D_0$：8 位的数据输入端，D_7为最高位。

I_{OUT1}：模拟电流输出端 1，当 DAC 寄存器中数据全为 1 时，输出电流最大，当 DAC 寄存器中数据全为 0 时，输出电流为 0。

I_{OUT2}：模拟电流输出端 2，I_{OUT2}与 I_{OUT1}的和为一个常数，即 $I_{OUT1} + I_{OUT2} =$ 常数。

R_f：反馈电阻引出端，DAC0832 是电流输出，为了取得电压输出，需在电压输出端接运算放大器。DAC0832 内部已经有反馈电阻，所以 R_5端可以直接接到外部运算放大器的输出端，这样相当于将一个反馈电阻接在运算放大器的输出端和输入端之间。

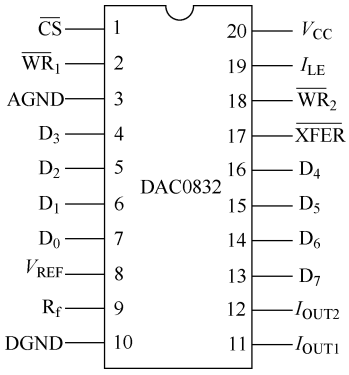


图 8-6　DAC0832 芯片引脚排列

V_{REF}：参考电压输入端，此端可接一个正电压，也可接一个负电压，它决定 0 ~ 255 的数字量转化出来的模拟量电压值的幅度，V_{REF}范围为 −10 ~ +10V。V_{REF}端与 D-A 内部 T 形电阻网络相连。

$\overline{CS}$：片选信号（低电平有效）。

I_{LE}：输入锁存允许信号（高电平有效）。

$\overline{WR_1}$：输入锁存器写选通信号（低电平有效）。

当$\overline{WR_1}$为低电平时，将输入数据传送到输入锁存器；当$\overline{WR_1}$为高电平时，输入锁存器中的数据被锁存；只有当I_{LE}为高电平，且$\overline{CS}$和$\overline{WR_1}$同时为低电平时，方能将锁存器中的数据进行更新。以上三个控制信号联合构成第一级输入锁存控制。

$\overline{WR_2}$：DAC 寄存器写选通信号（低电平有效）。该信号与信号$\overline{XFER}$配合，可使锁存器中的数据传送到 DAC 寄存器中进行转换。

$\overline{XFER}$：数据传送控制信号（低电平有效）。该信号与$\overline{WR_2}$信号联合使用，构成第二级锁存控制。

V_{CC}：芯片供电电压，范围为 +5 ~ 15V。

AGND：模拟量地，即模拟电路接地端。

DGND：数字量地。

2. DAC0832 有三种不同的工作方式

在使用时，可以通过对控制引脚的不同设置，采用双缓冲方式（两级输入锁存），也可以用单缓冲方式（只用一级输入锁存，另一级始终直通），或者接成完全直通的形式。所以 DAC0832 有三种不同的工作方式：直通方式、单缓冲方式、双缓冲方式。

（1）直通方式

当 ILE 为高电平，$\overline{CS}$、$\overline{WR_1}$、$\overline{WR_2}$和$\overline{XFER}$都接数字地时，DAC 处于直通方式，8 位数字量一旦到达 $D_7 \sim D_0$输入端，就立即加到 8 位 D-A 转换器，被转换成模拟量。直通方式连

接如图 8-7 所示。

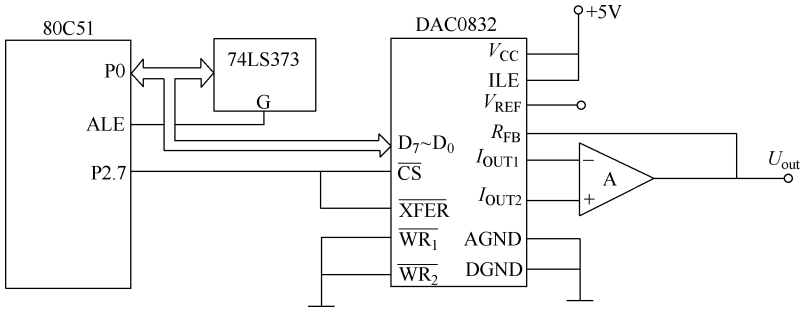


图 8-7　直通方式连接

（2）单缓冲方式

单缓冲方式就是使 DAC0832 的两个输入寄存器中有一个处于直通方式，而另一个处于受控的锁存方式，或者说两个输入寄存器同时受控的方式。

在实际应用中，如果只有一路模拟量输出，或虽有几路模拟量但并不要求同步输出的情况，就可采用单缓冲方式。单缓冲方式连接如图 8-8 所示。

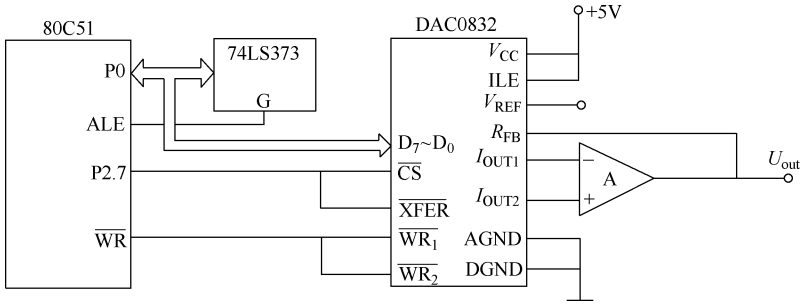


图 8-8　单缓冲方式连接

（3）双缓冲方式的接口与应用

双缓冲方式就是把 DAC0832 的两个锁存器都接成受控锁存方式。双缓冲方式用于多路 D-A 转换系统，以实现多路模拟信号同步输出的目的。为了实现对寄存器的控制，应当给两个寄存器各分配一个地址，以便能单独进行操作。图 8-9 所示为使用地址译码输出分别接$\overline{CS}$和$\overline{XFER}$实现的。由 80C51 的$\overline{WR}$为$\overline{WR_1}$和$\overline{WR_2}$提供写选通信号。这样就完成了两个寄存器都可控的双缓冲接口方式。

由于两个寄存器各占据一个地址，因此，在程序中需要使用两条传送指令，才能完成一个数字量的模拟转换。假定输入寄存器地址为 0EH，DAC 寄存器地址为 0FH，则完成一次 D-A 转换的程序段如下：

```
MOV R0，#0EH        ；装入输入寄存器地址
MOVX @R0，A         ；转换数据送入寄存器
INC R0              ；产生 DAC 寄存器地址
MOVX @R0，A         ；数据通过 DAC 寄存器
```

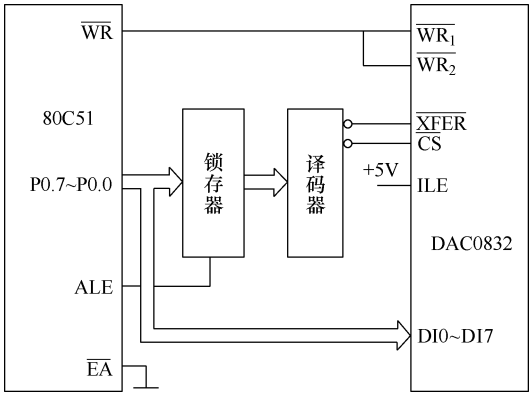


图 8-9　双缓冲方式连接图

下面通过几道例题来介绍 DAC0832 三种不同的工作方式在实际中的应用。

【例 8-2】　直通方式产生锯齿波电压信号。电路如图 8-10 所示。

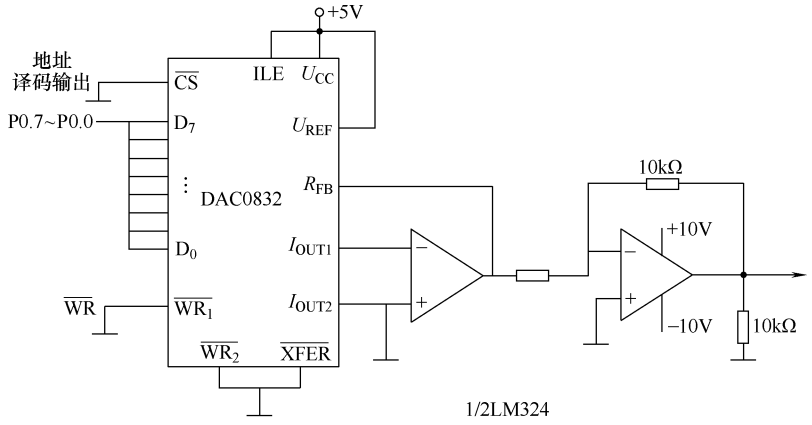


图 8-10　直通方式产生锯齿波电压信号电路图

集成运放在电路中的作用是把 DAC0832 输出电流转换为电压。即实现电流电压转换。锯齿波电压信号随时间变化而上升，达到最大值后，又从 0 开始上升，再到最大值如此循环下去。因此，只要让 DAC0832 输入的数字量也如此变化就可使输出端输出锯齿波。

对锯齿波的产生作如下说明：

① 程序每循环一次，（R0）加 1，因此实际上锯齿波的上升沿是由 256 个小阶梯构成的。但由于阶梯很小，所以看上去就如图所表示的线性增长锯齿波。

② 延迟时间不同，波形周期不同，锯齿波的斜率就不同。

参考程序如下：

```
        ORG 0000H
        MOV R0，#00H   ；置转换初值
DAC：   MOV P0，R0     ；送数据到 P0 口，DAC0832 同时进行转换
        INC R0         ；转换数字量加 1，当加到最大值 0FFH 时，再加 1，R0 变为 0
        ACALL  DELAY   ；延时量决定锯齿波周期
```

```
        AJMP DAC
DELAY：……（略）          ；延时子程序
        END
```

【例 8-3】　单缓冲方式产生锯齿波。电路如图 8-11 所示。

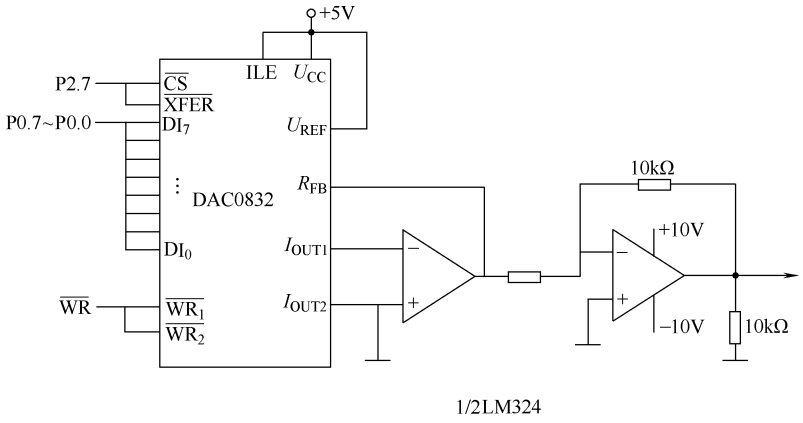


图 8-11　单缓冲方式产生锯齿波电压信号电路图

$\overline{WR_2}=0$ 和$\overline{XFER}=0$，因此 DAC 寄存器处于直通方式。而输入寄存器处于受控锁存方式，$\overline{WR_1}$接 8051 的$\overline{WR}$，ILE 接高电平，CS 接 P2.7 故输入寄存器地址为 07FFFH。

软件设计思路与【例 8-2】相同，只是改为用“MOVX　@DPTR，A”来发送数据和启动转换。

单缓冲方式产生锯齿波的源程序如下：

```
        MOV　DPTR，#7FFFH      ；指向 0832 地址
        MOV R0，#00H           ；置转换数字初值
DA1：   MOV A，R0
        MOVX@ DPTR，A          ；启动转换
        INC R0                 ；转换数字量加 1
        ACALL　DELAY           ；延时
        AJMP DA1
DELAY：MOV　R7，#7DH           ；延时子程序
DL1：   NOP
        NOP
        DJNZ R7，DL1
        RET
```

【例 8-4】　DAC0832 同步波形输出正弦波、锯齿波。DAC0832 与单片机的接口电路如图 8-12 所示。

DAC0832（1）输入寄存器地址为 0BFFFH，DAC0832（2）输入寄存器地址为 0DFFFH，DAC0832（1）和 DAC0832（2）的 DAC 寄存器地址均为 7FFFH。正弦波的产生由各采样点数据依次进行 D-A 转换得到。

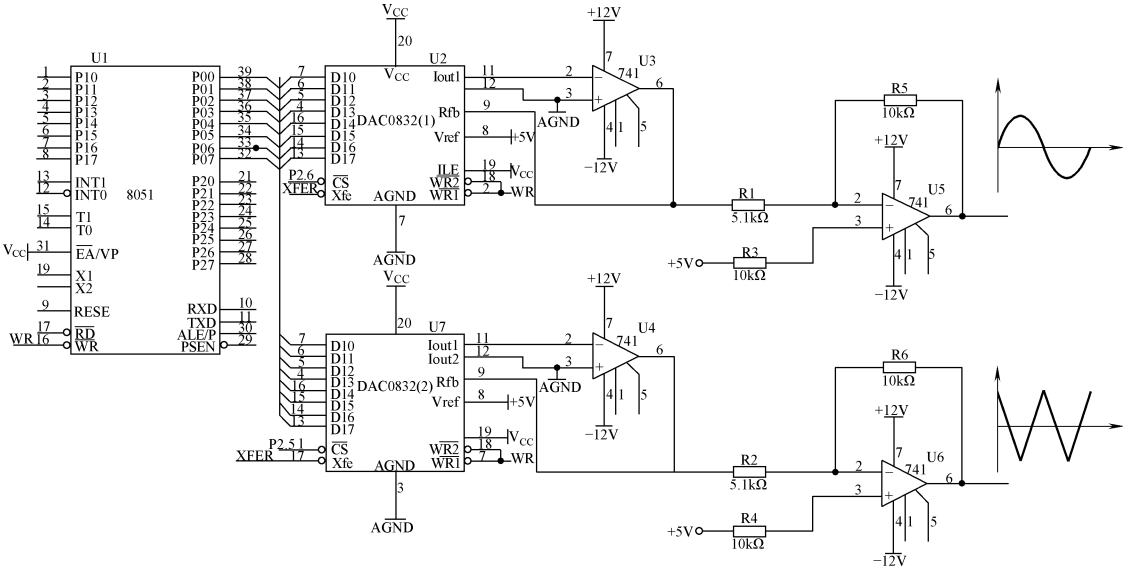


图 8-12　DAC0832 同步波形输出正弦波、锯齿波图

双缓冲方式同步波形输出正弦波参考程序：

```
        ORG   0000H
START:  MOV   R1, #255          ; 256 个取样点
        MOV   R2, #0            ; 锯齿波初值
LOOP:   MOV   DPTR, #0DFFFH     ; DAC0832（2）输入寄存器地址
        MOV   A, R2
        MOVX  @DPTR, A          ; 锯齿波送 DAC0832（2）
        MOV   DPTR, #DTAB       ; 取信号数据表首地址
        MOVC  A, @A+DPTR        ; 查表取正弦波信号数据
        MOV   DPTR, #0BFFFH     ; DAC0832（2）输入寄存器地址
        MOVX  @DPTR, A          ; 输出正弦波信号到 DAC0832（1）
        MOV DPTR, #7FFFH        ; DAC0832（1）DAC0832（2）DAC 寄存器地址
        MOVX  @DPTR, A          ; 同时启动两个 0832 转换
        INC   R2
        DJNZ  R1, LOOP
        SJMP  START
DTAB:   DB 80H, 83H, 86H, 8DH, 90H, 96H, 99H, 9CH      ; 正弦数据表
        DB 9FH, 0A2H, 0A5H, 0A8H, 0ABH, 0AEH
        DB 0B1H, 0B4H, 0B7H, 0BAH, 0BCH, 0BFH, 0C2H, 0C5H
        DB 0C7H, 0CAH, 0CFH, 0D1H, 0D4H, 0D6H, 0D8H
        DB 0DAH, 0DDH, 0DFH, 0E1H, 0E3H, 0E5H, 0E7H, 0E9H
        DB 0EAH, 0ECH, 0EEH, 0EFH, 0F1H, 0F2H, 0F4H, 0F5H
        DB 0F6H, 0F7H, 0F8H, 0F9H, 0FAH, 0FBH, 0FCH, 0FDH
```

```
DB 0FDH，0FEH，0FFH，0FFH，0FFH，0FFH，0FFH，0FFH
DB 0FFH，0FFH，0FFH，0FFH，0FFH，0FFH，0FEH，0FDH
DB 0FDH，0FCH，0FBH，0FAH，0F9H，0F8H，0F7H，0F6H
DB 0F5H，0F4H，0F2H，0F1H，0EFH，0EEH，0ECH，0EBH
DB 0EAH，0E9H，0E7H，0E5H，0E3H，0E1H，0DFH，0DDH
DB 0DAH，0D8H，0D6H，0D4H，0D1H，0CFH，0CCH，0CAH
DB 0C7H，0C5H，0C2H，0BFH，0BCH，0BAH，0B7H，0B4H
DB 0B1H，0AEH，0ABH，0A8H，0A5H，0A2H，9FH，9CH
DB 99H，96H，93H，90H，8DH，89H，86H，83H
DB 80H，80H，7CH，79H，76H，72H，6FH，6CH
DB 69H，66H，63H，60H，5DH，5AH，57H，55H
DB 51H，4EH，4CH，48H，45H，43H，40H，3DH
DB 3AH，38H，35H，33H，30H，2EH，2BH，29H
DB 27H，25H，22H，20H，1EH，1CH，1AH，18H
DB 16H，15H，13H，11H，10H，0EH，0DH，0BH
DB 0AH，09H，08H，07H，06H，04H，03H，02H
DB 02H，01H，00，00，00，00，00，00
DB 00，00，00，00，00，00，01H，02H
DB 02H，03H，04H，05H，06H，07H，08H，09H
DB 0AH，0BH，0DH，0EH，10H，11H，13H，15H
DB 16H，18H，1AH，1CH，1EH，20H，22H，25H
DB 27H，29H，2BH，2EH，30H，33H，35H，38H
DB 3AH，3DH，40H，43H，45H，48H，4CH，4EH
DB 51H，55H，57H，5AH，5DH，60H，63H，66H
DB 69H，6CH，6FH，72H，76H，79H，7CH，80H
END
```

8.2.3　高于 8 位 D-A 转换器芯片及与 80C51 单片机的接口

为了提高转换精度，可选用更多位数的 D-A 转换器，如 10 位、12 位、16 位。其转换原理与 8 位 D-A 转换器基本一样，不同的是在与数据线位数较少的微型计算机进行接口连接时，数据要分两次或三次输入。以 12 位 D-A 转换器 AD667 来介绍高于 8 位 D-A 转换器芯片。

1. 12 位 D-A 转换器 AD667

AD667 是一个完整的 12 位 D-A 转换器，片内含两级数据输入锁存器，且具有建立时间短和精度高的特点。图 8-13 所示为 AD667 的原理结构。该芯片的总线逻辑由 4 个独立寻址的锁存器组成。它们分为两级，第一级包括 3 个 4 位寄存器，可以直接从 4 位、8 位、12 位微型计算机总线获得数据。一旦全 12 位数据被装入第一级，便一起被置入第二级的 12 位 D-A 转换器。这种双缓冲结构避免了产生虚假的模拟量输出值。

如图 8-13 所示，内部锁存器分别由 AD667 的地址线 $A_3 \sim A_0$ 及片选信号控制，所有控

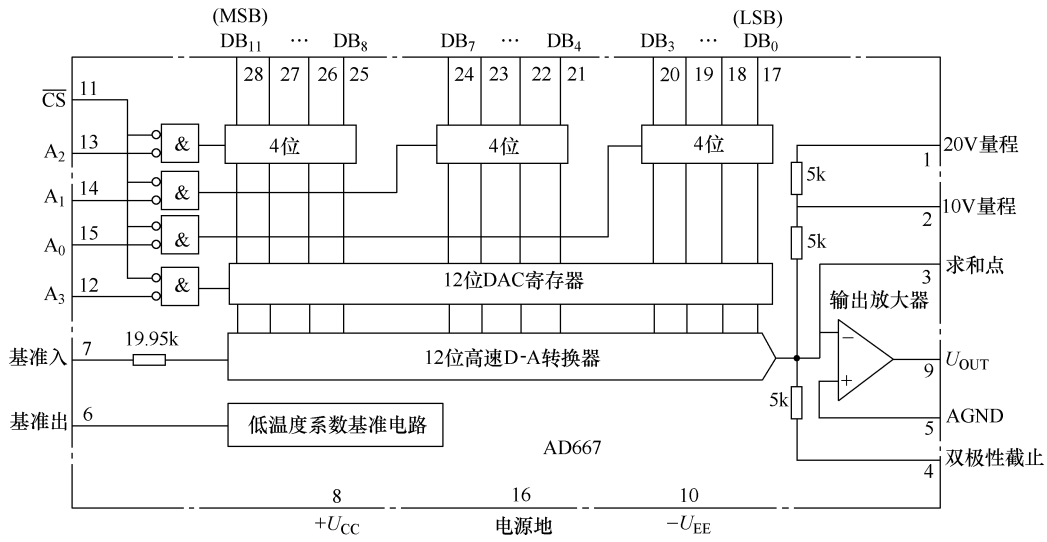


图 8-13 AD667 的原理结构

制信号均为低电平有效，见表 8-2。

表 8-2 AD667 真值表

$\overline{CS}$	A_3	A_2	A_1	A_0	操 作
1	×	×	×	×	无操作
×	1	1	1	1	无操作
0	1	1	1	0	选通第一级低 4 位寄存器
0	1	1	0	1	选通第一级中 4 位寄存器
0	1	0	1	1	选通第一级高 4 位寄存器
0	0	1	1	1	从第一级向第二级置数
0	0	0	0	0	所有锁存器均透明

AD667 允许有两个以上的锁存器同时被选通，因此它允许与 4 位、8 位、12 位微型计算机接口进行连接。

2. 高于 8 位 D-A 转换器及其接口

下面以 AD667 为例介绍 12 位 D-A 转换器与 8051 的接口连接及程序设计的方法，设接口电路如图 8-14 所示。

与 DAC0832 相似，AD667 也由两级缓冲器组成。主要差别在于 AD667 的第一级由 3 个 4 位寄存器组成。如图 8-13 所示，待转换的数字量分低 8 位和高 4 位两步传入 AD667。由 P2 口产生的高 8 位地址线控制 D-A 转换器的片选信号及输入寄存器的选通信号。当 P2.1 = 0、P2.0 = 1 时，选通低 8 位；反之 P2.1 = 1、P2.0 = 0 时，选通高 4 位和第二级 12 位 DAC 寄存器。当然，上述两种控制都必须在 CS = 0 的前提之下才生效。

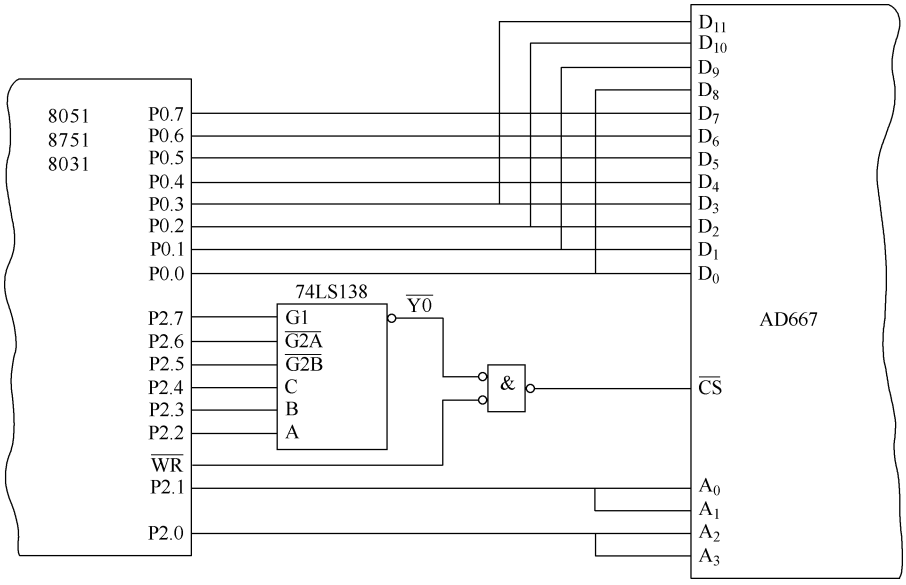


图 8-14　12 位 D-A 转换器 AD667 与 8051 的接口

本 章 小 结

A-D 和 D-A 转换器是计算机与外界联系的重要途径。本章介绍了 D-A 转换芯片 DAC0832 的工作原理，并详细介绍了 DAC0832 直通方式、单缓冲方式和双缓冲方式的接口及应用。A-D 转换技术主要介绍 ADC0809 与 MCS-8051 的接口电路，叙述了 A-D 转换后两者间的数据传送方式，即定时传送方式、查询方式和中断方式。还通过 8 路模拟量输入巡回检测系统实例，详细介绍了两者间数据传送的编程方法。

各模块技术的学习最终是为了设计有实际用途的单片机应用系统。本章设计了两个实用性很强的单片机应用系统实例。由此可以使读者将所学知识加以系统化并用于实践。

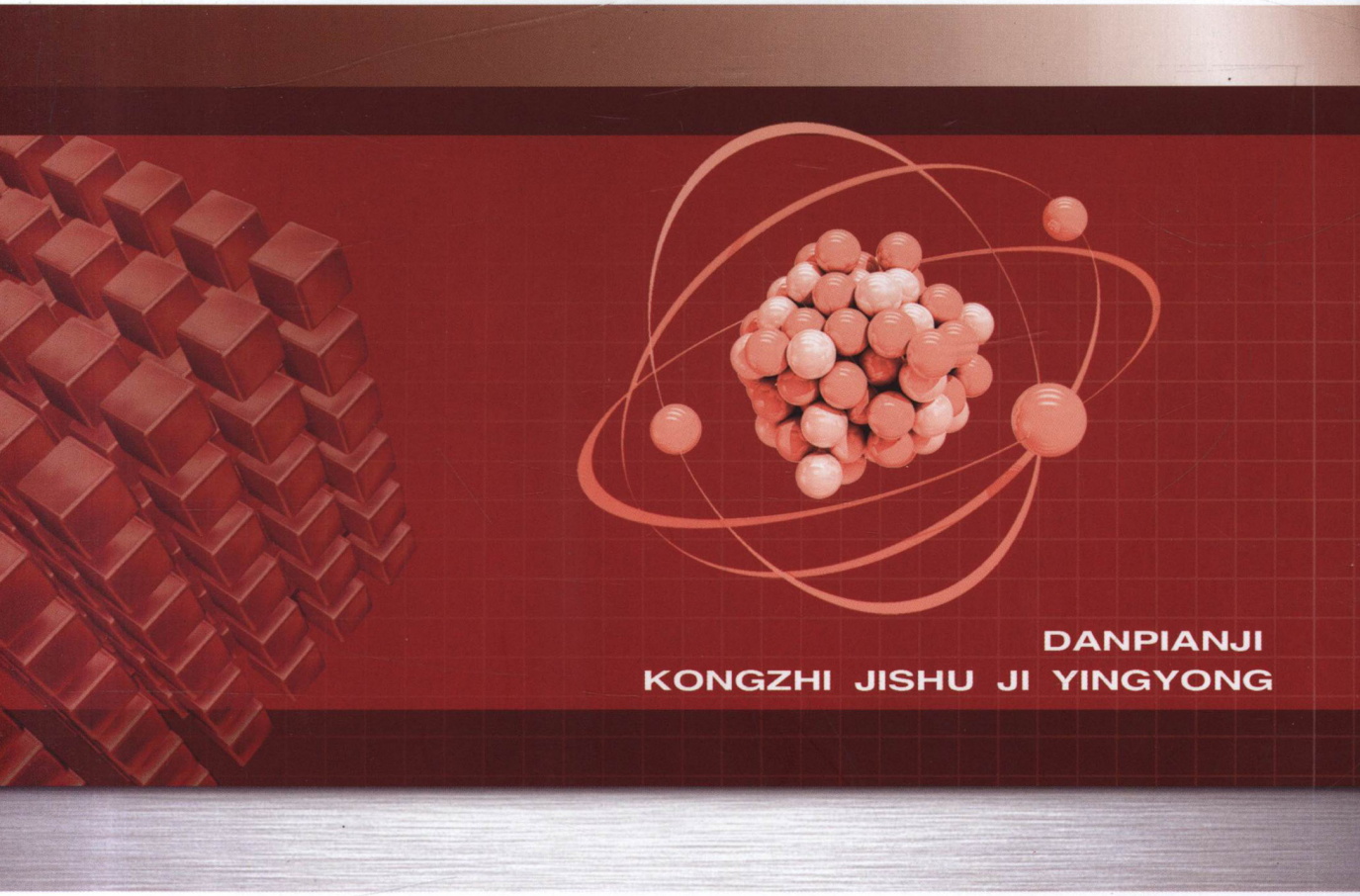
习　　题

1. D-A 与 A-D 转换器的主要功能是什么？

2. DAC0832 有哪三种不同的工作方式？

3. DAC0832 与 8051 单片机接口时有哪些控制信号？作用分别是什么？ADC0809 与 8051 单片机接口时有哪些控制信号？作用分别是什么？

4. 使用 DAC0832 时，单缓冲方式如何工作？双缓冲方式如何工作？软件编程有什么区别？



地址：北京市百万庄大街22号
邮政编码：100037
电话服务
社服务中心：010-88361066
销售一部：010-68326294
销售二部：010-88379649
读者购书热线：010-88379203
网络服务
教材网：http://www.cmpedu.com
机工官网：http://www.cmpbook.com
机工官博：http://weibo.com/cmp1952
封面无防伪标均为盗版

上架指导： 工业技术／电气工程／单片机
ISBN 978-7-111-44921-8
策划编辑◎罗 莉/ 封面设计◎赵颖喆

ISBN 978-7-111-44921-8
9 787111 449218 >
定价：39.00元