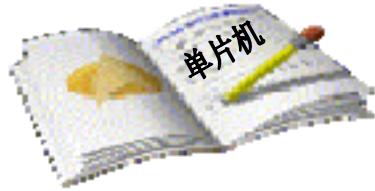


“单片机原理及应用”

课程教案



武汉科技学院电子信息工程学院

2007 年 2 月 8 日

**1、课程性质：**专业技术课

**2、考核方式：**闭卷考试

**3、教材：**《单片机原理与应用及 C51 程序设计》

编著：谢维成等 清华大学出版社

**4、教学目的：**

通过理论授课与上机实践，使学生掌握单片机的基本原理与应用，让学生了解单片机的内部结构、各硬件部分的工作原理及使用方法和单片机应用系统的组成原理，掌握单片机的汇编语言或 C 语言的指令功能、编程方法及软件开发技术，通过实例介绍单片机系统常用接口、扩展电路及其 C 语言应用程序设计，使学生较为熟练地掌握一种单片机产品的应用开发技术，从而有能力进一步对其他单片机产品的应用系统从事研制和开发工作。



# 目 录

第 1 章	单片机概述 .....	4
第 2 章	单片机硬件结构 .....	14
第 3 章	指令系统 .....	38
第 4 章	汇编语言程序设计 .....	69
第 5 章	单片机存储器扩展 .....	98
第 6 章	中断与定时系统 .....	120
第 7 章	I/O 扩展及应用 .....	146
第 8 章	串行数据通信 .....	182
第 9 章	数/模及模/数转换器接口 .....	200
第 10 章	单片机应用及开发技术 .....	211

# 第1章 单片机概述

## 一、教学要求：

了解：计算机的发展、分类、特点与应用，单片机的概念、发展及应用领域，以及典型单片机系列的基本情况。

## 二、教学内容：

- 1. 1 计算机的发展、分类、特点与应用
- 1. 2 单片机的概念
- 1. 3 单片机的发展
- 1. 4 单片机的应用

## 三、教学重点：单片机的概念。

## 四、教学难点：单片机的应用。

## 五、建议学时：2 学时。

## 六、教学内容：

### 1.1 单片机的概念

#### 1.1.1 单片机的名称

单片微机是早期 Single Chip Microcomputer 的直译，它忠实地反映了早期单片微机的形态和本质。

单片微型计算机简称单片机（Single Chip Microcomputer），又称微控制器（Microcomputer Unit）。将计算机的基本部件微型化，使之集成在一块芯片上。片内含有 CPU、ROM、RAM、并行 I/O、串行 I/O、定时器/计数器、中断控制、系统时钟及总线等。

随后，按照面向对象、突出控制功能，在片内集成了许多外围电路及外设接口，突破了传统意义的计算机结构，发展成 microcontroller 的体系结构，目前国外已普遍称之为微控制器 MCU（Micro Controller Unit）。

鉴于它完全作嵌入式应用，故又称为嵌入式微控制器 Embedded Microcontroller)。

### 1.1.2 通用单片机和专用单片机

根据控制应用可分为：通用型和专用型两大类。

#### 1、早期——通用型单片微机。

通过不同的外围扩展来满足不同的应用对象要求。

#### 2、随着应用领域的不断扩大出现了专门为某一类应用而设计的单片机

——专用型单片微机。

目的：降低成本、简化系统结构、提高可靠。

如：用于计费率电表、用于电子记事簿的单片机等。

### 1.1.3 单片机与单片机系统

单片机通常是指芯片本身，集成的是一些基本组成部分。是典型的嵌入式系统的主  
要构成单元，只能作为嵌入式应用，即嵌入到对象环境、结构、体系中作为其中的一个  
智能化控制单元。

如：洗衣机、电视机、VCD、DVD 等家用电器，打印机、复印机、通信设备、智  
能仪表、现场总线控制单元等。

单片机系统是在单片机芯片的基础上扩展其它电路或芯片构成的具有一定应用功  
能的计算机系统。

单片机应用系统中包括了满足对象（如洗衣机）要求的全部硬件电路和应用软件。  
构成各种嵌入式应用的电路系统，统称为单片机应用系统。

单片微机应用系统结构通常分三个层次，即单片机、单片机系统、单片机应用系统。

单片机：通常是应用系统的主机，设计单片机应用系统时，为所选择的单片机系列  
器件。

单片机系统：单片微机资源的扩展，外围接口电路进入片内，最终向单片应用系统  
集成发展。——最终产品的目标系统，除了硬件电路外，还须嵌入系统应用程序。按照

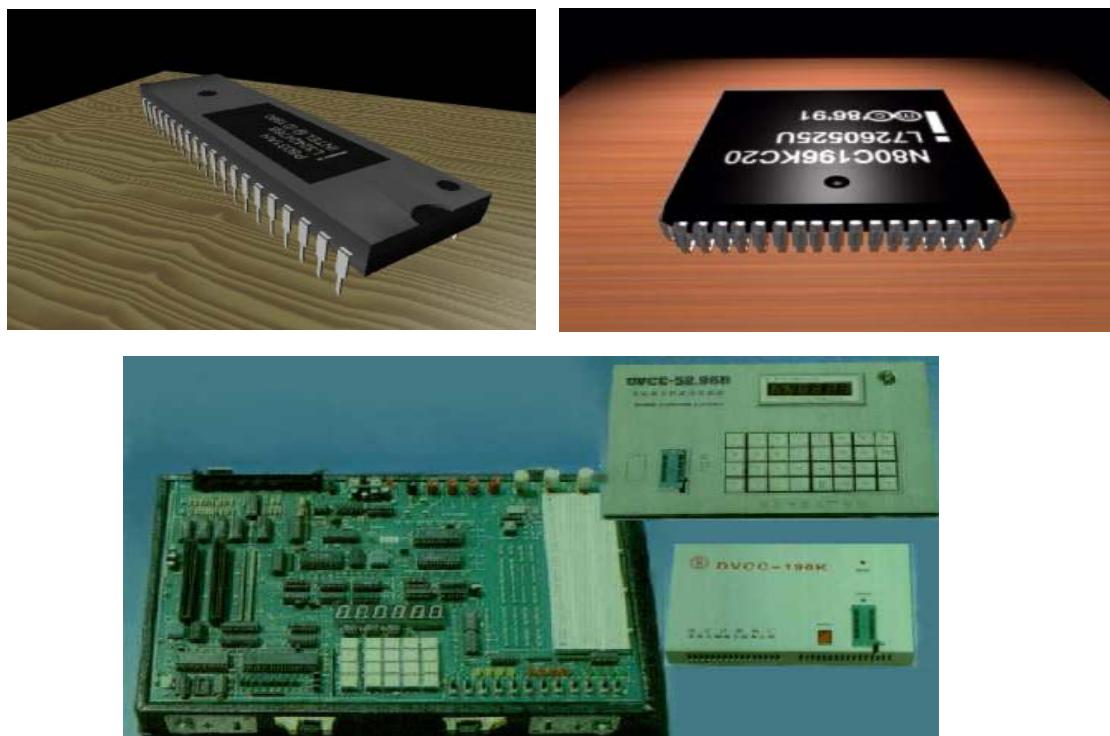
所选择的单片机，以及单片机的技术要求和嵌入对象对单片机的资源要求构成单片机系统。

**单片机应用系统：**按照单片机要求在外部配置单片机运行所需要的时钟电路、复位电路等，构成了单片机的最小应用系统。在单片机中 CPU 外围电路不能满足嵌入对象功能要求时，在单片机外部扩展 CPU 外围电路，如存储器、定时器/计数器、中断源等，形成能满足具体嵌入应用的一个计算机系统。

#### 1.1.4 单片机应用系统与单片机开发系统

单片机开发系统是单片机的开发调试的工具，有单片单板机和仿真器。实现单片机应用系统的硬、软件开发。

MDS（微型机开发系统）、ICE（在线仿真器）



#### 1.1.5 单片机的程序设计语言和软件

有三类  $\left\{ \begin{array}{l} \text{机器语言 (Machine Language)} \\ \text{汇编语言 (Assemble)} \\ \text{高级语言 (High Level Language)} \end{array} \right.$

### 机器语言：

单片机应用系统只使用机器语言（指令的二进制代码，又称指令代码）。机器语言指令组成的程序称目标程序。

MCS-51 两个寄存器相加的机器语言指令：00101000

### 汇编语言：

与机器语言指令一一对应的英文单词缩写，称为指令助记符。汇编语言编写的程序称为汇编语言程序。

MCS-51 两个寄存器相加汇编语言指令：ADD A, R0

### 高级语言：

高级语言源程序 C-51、C、PL/M51 等。

简单——控制程序不太长。

复杂——多种多样的控制对象，少有现成程序借鉴。

简单系统——不含管理和开发功能。

复杂系统——实时系统，需要监控系统（甚至实时多任务操作系统）。

{ 编译型高级语言可生成机器代码；  
解释型高级语言必须在解释程序支持下直接解释执行。

因此，只有编译型高级语言才能作为微机开发语言。

### 不同计算机语言的应用：

源程序通过编译得到机器能执行的目标程序。

汇编语言程序可以高效率利用计算机资源，目标程序占用内存少，执行速度快，适合于自动测控系统反应快速、结构紧凑的要求。实际应用中，常与 C 语言配合使用。

高级语言程序容易掌握，通用性好，但编译程序系统开销大，目标程序占用内存多，且执行时间比较长，多用于科学计算、工业设计、企业管理。

## 1.2 单片机的发展

### 1.2.1 单片机发展概述

#### 一、电子计算机的发展历史

1、第一代（1946—1958）：电子管计算机。

用于：科学计算

2、第二代（1958—1964）：晶体管计算机。

用于：科学计算、数据处理、工业控制

3、第三代（1964—1971）：集成电路计算机、网络。

用于：科学计算、数据处理、工业控制、事务管理。

4、第四代（1971— ）：大规模集成电路计算机。

用于：计算量极大的高尖技术及国民经济领域出现了微型机。

5、第五代：智能型计算机正在研制中。

用于：模拟人的智能，识别图像、语言和物体，联想、推理、解答问题，使用自然语言进行会话处理。

#### 二、微型计算机的发展历史

微型机算计的核心部分：微处理器的发展已经历了五代。

第一代（1971—1973）：4位→8位（初级）

第二代（1973—1975）：8位（初级）

第三代（1975—1978）：初级8位单片机

Intel MCS—48系列单片机

第四代（1978—80年代中期）：高档8位单片机

Intel MCS—51系列单片机→16位、32位

第五代（80年代中期至今）：→64位

➤ 1976-  ：初级8位单片机 Intel MCS-48系列

➤ 1980- : 高档 8 位单片机 Intel MCS-51 系列:

—51 子系列: 8031/8051/8751

—52 子系列: 8032/8052/8752

低功耗型 80C31

高性能型 80C252

廉价型 89C2051/1051

➤ 1983- : 16 位单片机 Intel MCS-96 系列: 8098/8096、80C198/80C196

32 位单片机 80960



MCS—48 (从 1976 年起):

低档型: 8021、8022

基本型: 8048、8748、8038

改进型: 8049、8749、8039 和 8050、8750、8040

MCS—51 (从 1980 年起):

基本型: 8051、8751、8031

改进型: 8052、8752、8032

低功耗型: 80C51、87C51、80C31

高性能型: 83C252、87C252、80C252

早期产品: 8X9X (8096)

MCS—96 (从 1983 年起):

改进型: 8X9XBH、8X9XJF

新产品: 8098 (准)

强功能型: 80C196、80C198 (准)

### 1.2.2 MCS-51 单片机系列

MCS-51 系列基本产品型号:

8051、8031、8751 称为 51 子系列。

不同型号 MCS-51 单片机 CPU 处理能力和指令系统完全兼容, 只是存储器和 I/O 接口的配置有所不同。

硬件配置基本配置:

1. 8 位 CPU
2. 片内 ROM/EPROM、RAM
3. 片内并行 I/O 接口
4. 片内 16 位定时器/计数器
5. 片内中断处理系统
6. 片内全双工串行 I/O 口

MCS-51 系列单片机的 3 种基本产品:

**8051:** 片内含有掩膜 ROM 型程序存储器, 只能由生产厂家代为用户固化, 批量大、永久保存、不修改时用。

**8751:** 片内含 EPROM 型程序存储器, 用户可固化, 可用紫外线光照射擦除; 但价格高。

**8031:** 片内无程序存储器, 可在片外扩展, 方便灵活, 价格便宜。

MCS-51 产品型号硬件配置									
型号	ROM	RAM	晶振	中断源	定时器	A/D	PWM	外部总线	掉电方式
8051	4KB	128B	12MHZ	5	2X16	无	无	8	有
8052	8KB	256B	12MHZ	5	2X16	无	无	8	有
80C51GB	8KB	256B	12MHZ	15	3X16	8X8	有	8	有
89C2051	2KB(OP)	256B	12MHZ	5	2X16	1X8	有	0	有

### 1.2.3 80C51 单片机系列

INTEL 公司先后推出了三个系列的单片机：

MCS—48 系列

MCS—51 系列

MCS—96 系列典型产品：

8096            8098            (准)

80C196            80C198            (准)

新一代 80C51 增加了一些外部接口功能单元，如 A/D，PCA，WDT 等。

PHILIPS：80C51

ATMEL (Flash ROM): AT89c51

CHMOS: 低功耗，高速度和高密度 (HMOS)，待机和掉电保护

## 1.3 单片机的应用

### 1.3.1 单片机应用的特点

单片机的应用很广泛，特点很多，仅从应用的角度来看：

计算机的控制应用分为：

1、控制系统离线应用：控制系统的计算机辅助设计（控制系统 CAD）

2、控制系统在线应用：计算机控制系统→使用单片机

工业控制领域与通用计算机系统不同的要求:

(1)面对控制对象。面对物理量传感变换的信号输入;

面对人机交互的操作控制;

面对对象的伺服驱动控制。

(2)嵌入到工控应用系统中的结构形态。

(3)工业现场环境中可靠性品质。

(4)突出控制功能。对外部信息及时捕捉;

对控制对象能灵活地实时控制;

有突出控制功能的指令系统,

如 I/O 口控制、位操作、丰富的转移指令等。

### 1.3.2 单片机的应用领域

• 工业自动化方面:

力、热、速度、加速度、位移。

• 仪器仪表:

降低成本、简化系统结构、提高可靠性。

• 家用电器:

小家电中要求小型价廉、程序容量不大。

• 信息和通信产品:

PDA 则要求大容量存储、大屏幕 LCD 显示、极低功耗等。

• 军事装备方面:

可靠性、极低功耗。

单片机的应用领域:

1、用单片机构成智能化产品:

■ 在智能仪器仪表中的应用;

- 在家用产品中的应用；
- 在医疗仪器中的应用；
- 在计算机外部设备中的应用。

## 2、单片机在工业测控领域中的应用：

- 过程控制：  
数控铣床、步进控制、生产流水线等；
- 数据采集；
- 信号处理；
- 旧设备的改造。

## 小结

- 1、单片机即单片微型计算机，是将计算机主机（CPU、内存和 I/O 接口）集成在一小块硅片上的微型机。
- 2、单片机为工业测控而设计，又称微控制器。  
具有三高优势（集成度高、可靠性高、性价比高）。
- 3、主要应用于工业检测与控制、计算机外设、智能仪器仪表、通讯设备、家用电器等。  
特别适合于嵌入式微型机应用系统。
- 4、单片机开发系统有单片单板机和仿真器。实现单片机应用系统的硬、软件开发。

## 第2章 单片机硬件结构

### 一、教学要求：

了解：单片机内部所包含的硬件资源及其功能特点和使用方法，注意几个概念：振荡周期、时钟周期、机器周期和指令周期的意义及它们之间的关系。

掌握：单片机芯片的内部组成及存储器结构，特别是片内 RAM 和四个并行 I/O 口的使用方法。

理解：单片机时钟电路与时序、输入输出口以及引脚的使用。注意“地址重叠”的问题，注意程序状态字 PSW 中各位的含义。

### 二、教学内容：

- 2.1 单片机逻辑结构及信号引脚
- 2.2 单片机的内部存储器
- 2.3 单片机并行输入/输出电路
- 2.4 单片机时钟电路与时序
- 2.5 单片机工作方式

### 三、教学重点：

单片机芯片的内部组成及存储器结构，特别是片内 RAM 和四个并行 I/O 口的使用方法。

### 四、教学难点：

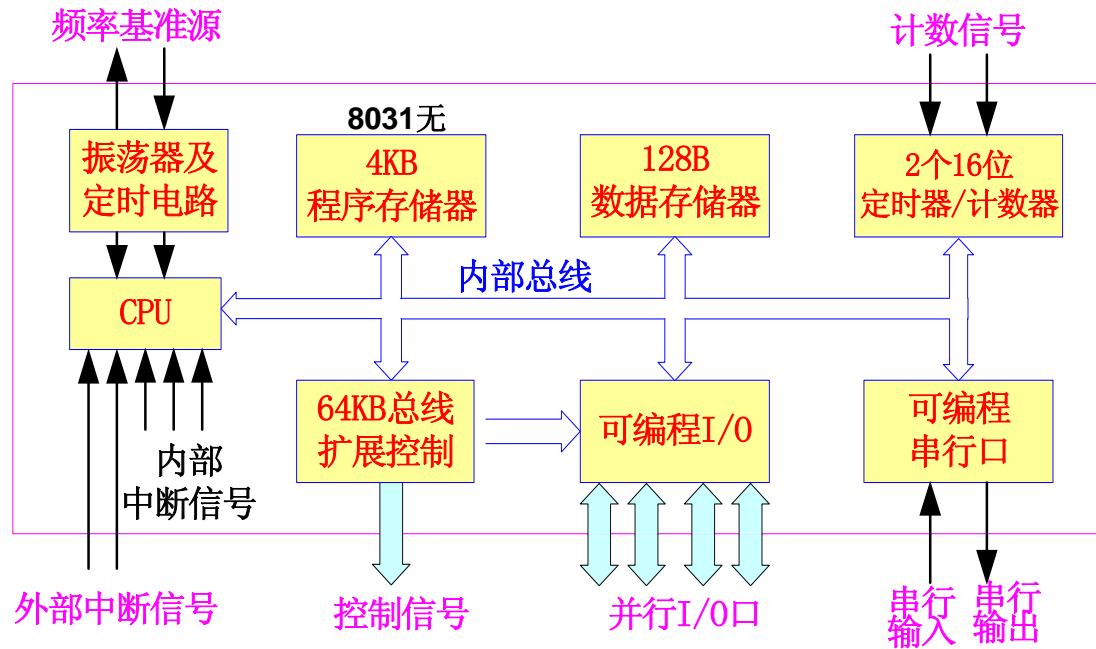
“地址重叠”的问题，注意程序状态字 PSW 中各位的含义。

五、建议学时：4 学时。

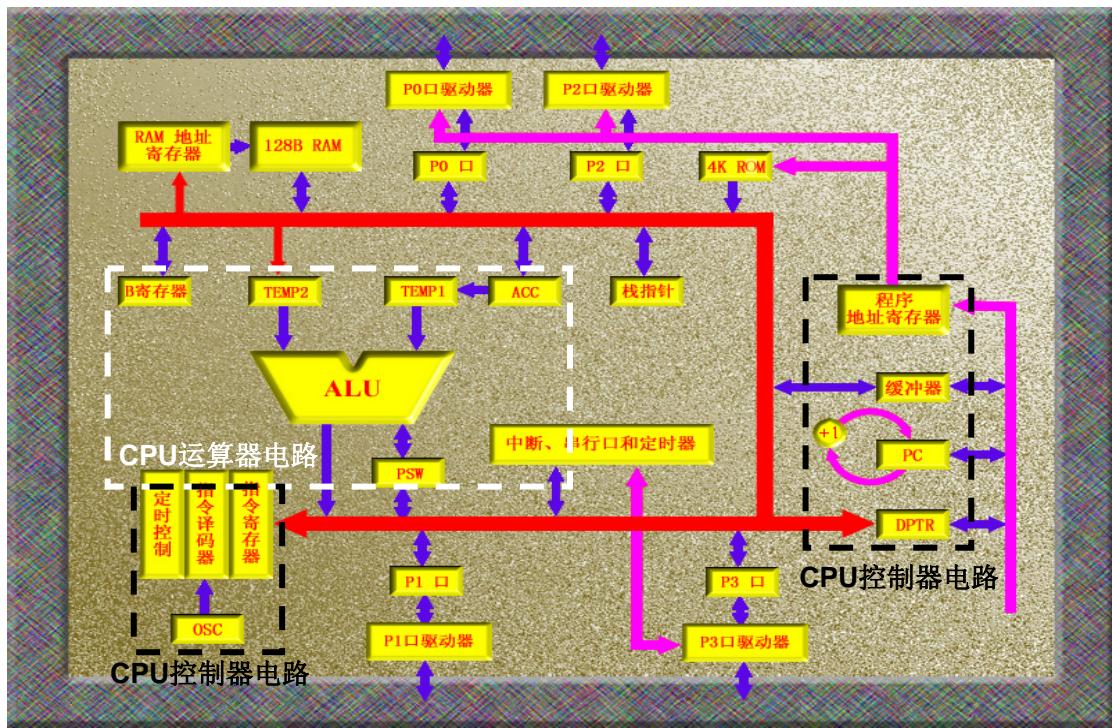
### 六、教学内容：

#### 2.1 逻辑结构及信号引脚

##### 2.1.1 结构框图



### 2.1.2 内部逻辑结构



### MCS-51 CPU

CPU 内部结构：

(1) 运算器电路：算术逻辑单元 ALU、累加器 ACC、寄存器 B、程序状态字 PSW 和

2 个暂存器等。

算术逻辑运算单元 ALU (8 位) :

$+$ 、 $-$ 、 $\times$ 、 $\div$  算术运算，与、或、非、异或逻辑运算，循环移位、位处理。

(2) 控制器电路：程序计数器 PC、PC+1 寄存器、指令寄存器、指令译码器、定时与控制电路等。

### 2.1.3 信号引脚

#### 1、I/O 口线功能

4 个 8 位并行 I/O 接口引脚 P0.0~P0.7、P1.0~P1.7、P2.0~P2.7 和 P3.0~P3.7 为多功能引脚，可自动切换用作数据总线、地址总线、控制总线和 I/O 接口外部引脚。

#### 2、控制线

ALE：地址锁存允许信号端

$\overline{PSEN}$ ：外部程序存储器读选通信号端

$\overline{EA}/V_{PP}$ ：程序存储器选择信号端/编程电源输入端

RST/V<sub>PD</sub>：复位信号端和后备电源输入端。输入 10ms 以上高电平脉冲，单片机复位。

$V_{PD}$  使用后备电源，可实现掉电保护。

复位电路：

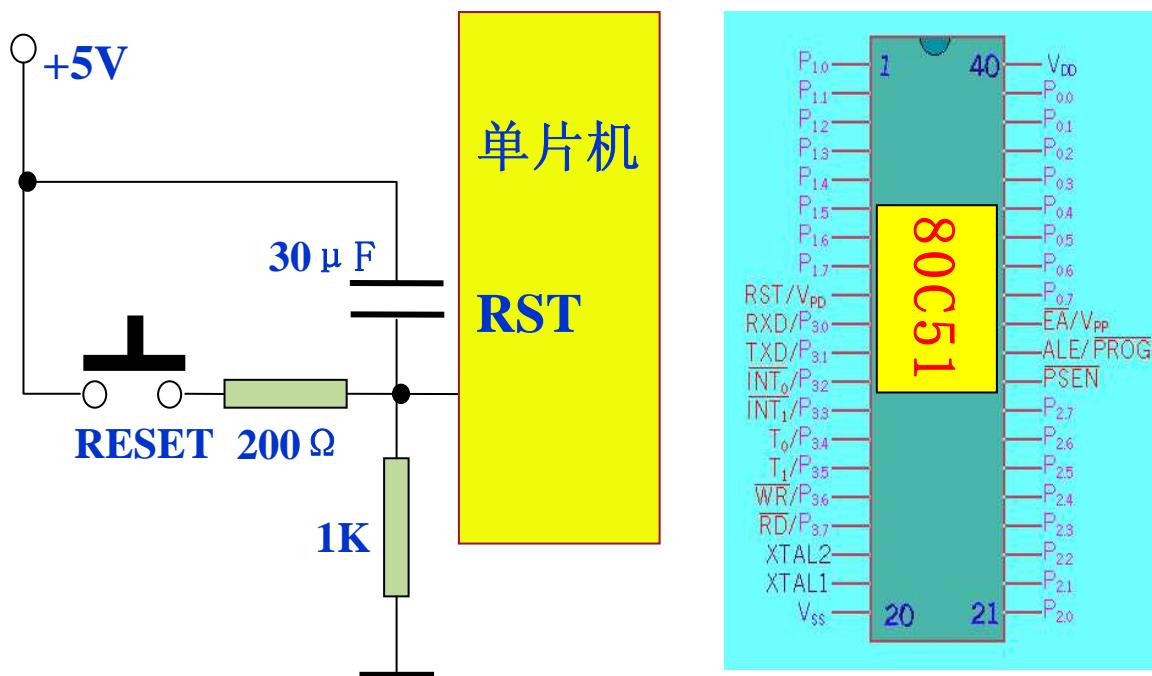
(1) 上电复位

(2) 外部信号复位

#### 3、电源及时钟引线

工作电源：V<sub>CC</sub>、V<sub>SS</sub>

时钟输入：XTAL1、XTAL2。



## 2.2 内部存储器

### 2.2.1 内部数据存储器低 128 单元

低 128 单元是单片机的真正 RAM 存储器。

分为三个区域：

#### 1. 寄存器区：

4 组寄存器（寄存器阵列）。

即 4 个工作寄存器 0 区~3 区。每组 8 个寄存单元（每单元 8 位），以 R0~R7 作寄存器名，暂存运算数据和中间结果。字节地址为 00H~1FH。

用 PSW 中的两位 PSW.4 和 PSW.3 来切换工作寄存器区，选用一个工作寄存器区进行读写操作。

#### 2. 位寻址区：

字节地址为 20H~2FH，既可作 RAM，也可位操作。

共有 16 个 RAM 单元，共 128 位，位地址为 00H~7FH。

### 3. 用户 RAM 区：

32 个单元，地址为 30H~7FH，在一般应用中常作堆栈区。

7FH	7F 7E 7D 7C 7B 7A 79 78	127
2FH	77 76 75 74 73 72 71 70	47
2EH	6F 6E 6D 6C 6B 6A 69 68	46
2DH	67 66 65 64 63 62 61 60	45
2CH	5F 5E 5D 5C 5B 5A 59 58	44
2BH	57 56 55 54 53 52 51 50	43
2AH	4F 4E 4D 4C 4B 4A 49 48	42
29H	47 46 45 44 43 42 41 40	41
28H	3F 3E 3D 3C 3B 3A 39 38	40
27H	37 36 35 34 33 32 31 30	39
26H	2F 2E 2D 2C 2B 2A 29 28	38
25H	27 26 25 24 23 22 21 20	37
24H	1F 1E 1D 1C 1B 1A 19 18	36
23H	17 16 15 14 13 12 11 10	35
22H	0F 0E 0D 0C 0B 0A 09 08	34
21H	07 06 05 04 03 02 01 00	33
20H		32
1FH		31
18H	寄存器区 3	
17H	寄存器区 2	
10H	寄存器区 1	
0FH	寄存器区 0	
08H		
07H		
00H		

RS0 RS1 的组合关系

RS1	RS0	寄存器组	片内 RAM 地址
0	0	第 0 组	00H~07H
0	1	第 1 组	08H~0FH
1	0	第 2 组	10H~17H
1	1	第 3 组	18H~1FH

工作寄存器地址表

组	RS1 RS0	R0	R1	R2	R3	R4	R5	R6	R7

0	0 0	00H	01H	02H	03H	04H	05H	06H	07H
1	0 1	08H	09H	0AH	0BH	0CH	0DH	0EH	0FH
2	1 0	10H	11H	12H	13H	14H	15H	16H	17H
3	1 1	18H	19H	1AH	1BH	1CH	1DH	1EH	1FH

## 2.2.2 内部数据存储器高 128 单元（也称特殊功能寄存器）

内部 RAM 的高 128 单元——专用寄存器（SFR）区 地址为 80H~FFH

表 2-6 MCS-51 系列单片机的特殊功能寄存器表

符 号	名 称	地 址
* ACC	累加器	E0H
* B	B 寄存器	F0H
* PSW	程序状态字	D0H
SP	栈指针	81H
DPTR	数据指针(包括指针高 8 位 DPH 和低 8 位 DPL)	83H(高 8 位),82H(低 8 位)
* P0	P0 口锁存寄存器	80H
* P1	P1 口锁存寄存器	90H
* P2	P2 口锁存寄存器	A0H
* P3	P3 口锁存寄存器	B0H
* IP	中断优先级控制寄存器	B8H
* IE	中断允许控制寄存器	A8H
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
TH0	定时器/计数器 0(高字节)	8CH
TL0	定时器/计数器 0(低字节)	8AH
TH1	定时器/计数器 1(高字节)	8DH
TL1	定时器/计数器 1(低字节)	8BH
* SCON	串行口控制寄存器	98H
SBUF	串行数据缓冲器	99H
PCON	电源控制及波特率选择寄存器	87H

注: 凡是标有“\*”号的 SFR 既可按位寻址, 也可直接按字节寻址。

## 1、SFR (80H~FFH) 介绍:

有 2 套地址:

字节地址: 只 21 个有效 (其中仅 11 个有位地址);

位地址: 只 83 位有效, 其字节地址可被 8 整除。

专用寄存器: A、B、PSW、DPTR、SP。

I/O 接口寄存器:

P0、P1、P2、P3、SBUF、TMOD、TCON、SCON 等。

### (1) 程序计数器 PC (16 位):

CPU 总是按 PC 的指示读取程序。PC 是一个 16 位的计数器。其内容为将要执行的指令地址 (即下一条指令地址), 可自动加 1。因此 CPU 执行程序一般是顺序方式。当发生转移、子程序调用、中断和复位等操作, PC 被强制改写, 程序执行顺序也发生改变。

复位时, PC=0000H。

### (2) 累加器 Acc (8 位):

需要 ALU 处理的数据和计算结果多数要经过累加器 A。

### (3) 寄存器 B (8 位):

与 A 累加器配合执行乘、除运算。也可用作通用寄存器。

### (4) 程序状态字 PSW (8 位):

存放 ALU 运算过程的标志状态。

位序	<b>B<sub>7</sub></b>	<b>B<sub>6</sub></b>	<b>B<sub>5</sub></b>	<b>B<sub>4</sub></b>	<b>B<sub>3</sub></b>	<b>B<sub>2</sub></b>	<b>B<sub>1</sub></b>	<b>B<sub>0</sub></b>
位符号	<b>C<sub>Y</sub></b>	<b>AC</b>	<b>F<sub>0</sub></b>	<b>RS<sub>1</sub></b>	<b>RS<sub>0</sub></b>	<b>OV</b>	<b>F<sub>1</sub></b>	<b>P</b>

### (5) 数据指针 DPTR (16 位):

存放片外存储器地址, 作为片外存储器的指针。可分成两个 8 位寄存器 DPH、DPL

使用。

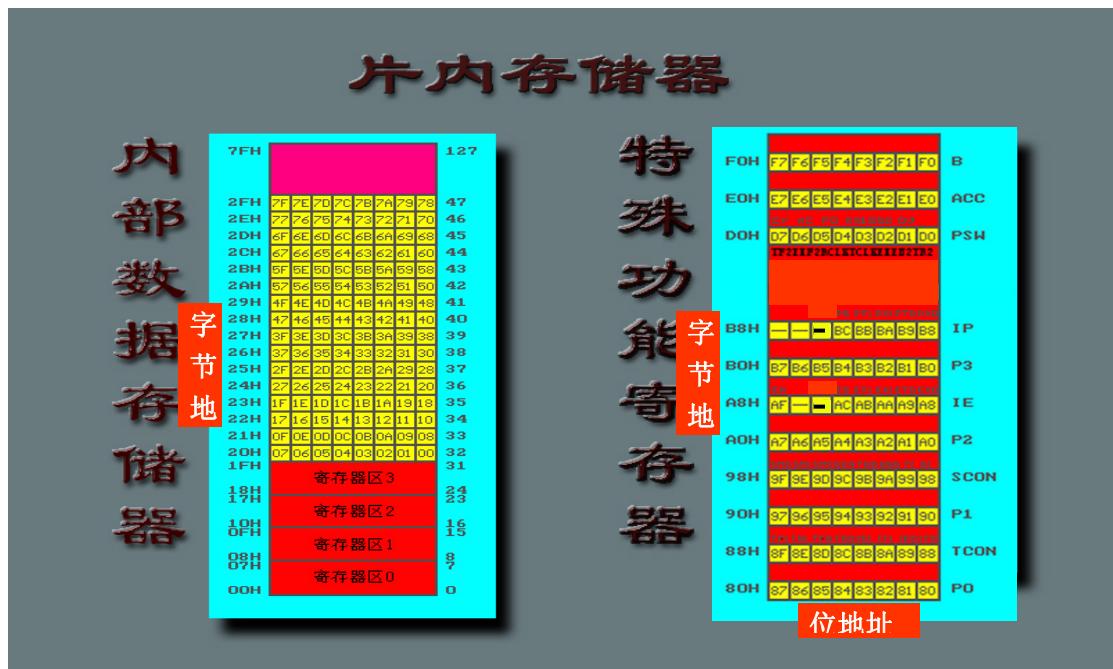
## 2. 专用寄存器的字节寻址

注意：

(1) 21 个可字节寻址的专用寄存器是不连续地分散在内部 RAM 高 128 单元之中，共 83 个可寻址位。尽管还剩余许多空闲单元，但用户并不能使用。  
在 22 个专用寄存器中，唯一一个不可寻址的 PC。PC 不占据 RAM 单元，它在 (2) 物理上是独立的，因此是不可寻址的寄存器。

(3) 对专用寄存器只能使用直接寻址方式，书写时既可使用寄存器符号，也可使用寄存器单元地址。

MCS-51 的寄存器在片内 RAM 都有映像地址。使用时，既可用寄存器名，也可用对应单元地址。



7FH	用户RAM区 (堆栈、数据缓冲)	FFH	
30H		F0H	B
2FH	位寻址区 (位地址00H~7FH)	E0H	ACC
20H		D0H	PSW
1FH	R7 第3组	B8H	IP
18H	R0 工作寄存器区	B0H	P3
17H	R7 第2组	A8H	IE
10H	R0 工作寄存器区	A0H	P2
0FH	R7 第1组	99H	SBUF
08H	R0 工作寄存器区	98H	SCON
07H	R7 第0组	90H	P1
00H	R0 工作寄存器区	8DH	TH1
		8CH	TH0
		8BH	TL1
		8AH	TL0
		89H	TMOD
		88H	TCON
		87H	PCON
		83H	DPH
		82H	DPL
		81H	SP
		80H	P0

片内RAM地址空间

### 2.2.3 堆栈操作

#### 1、堆栈类型：

向上生长型（向地址增大的方向生成）：MCS—51 系列

向下生长型（向地址较低的方向生成）：MCS—96 系列

#### 2、堆栈指针 SP（8 位）：

MCS—51 系列的堆栈是按“先进后出”原则存取数据的存储区。

MCS—51 堆栈设在片内 RAM 区。

数据入栈时：先 SP 自动加 1，后写入数据，SP 始终指向栈顶地址。

——“先加后压”

数据出栈时：先读出数据，后 SP 自动减 1，SP 始终指向栈顶地址。

——“先弹后减”

复位时 SP=07H。但在程序设计时应将 SP 值初始化为 30H 以后，以免占用宝贵的寄存器区和位地址区。

#### 2.2.4 内部程序存储器

80C51 内有 4KB ROM，其地址为 0000H~0FFFH（内部 ROM）。

其中 0000H~0002H 是系统的启动单元。

系统复位后 (PC)=0000H，开始取指令执行程序。

如果不从 0000H 开始，应存放一条无条件转移指令，以便直接转去执行指定的程序。

作用：

(1) 用来存放固化了的用户程序，取指地址由程序计数器 PC 给出，PC 具有自动加 1 的功能；

(2) 固化一片数据区，存放被查询的表格和参数等。

中断入口：0003H~0023H

0003H~000AH 外部中断 0 (INT0) 中断地址区

000BH~0012H 定时器/计数器 0 (T0) 中断地址区

0013H~001AH 外部中断 1 (INT1) 中断地址区

001BH~0022H 定时器/计数器 1 (T1) 中断地址区

0023H~002AH 串行 (RI/TI) 中断地址区

中断服务程序存放方法：

(1) 从中断地址区首地址开始，在中断地址区中直接存放；

(2) 从中断地址区首地址开始，存放一条无条件转移指令，以便中断响应后，通过中断地址区，再转到中断服务程序的实际入口地址区去。

程序存储器保留的单元：

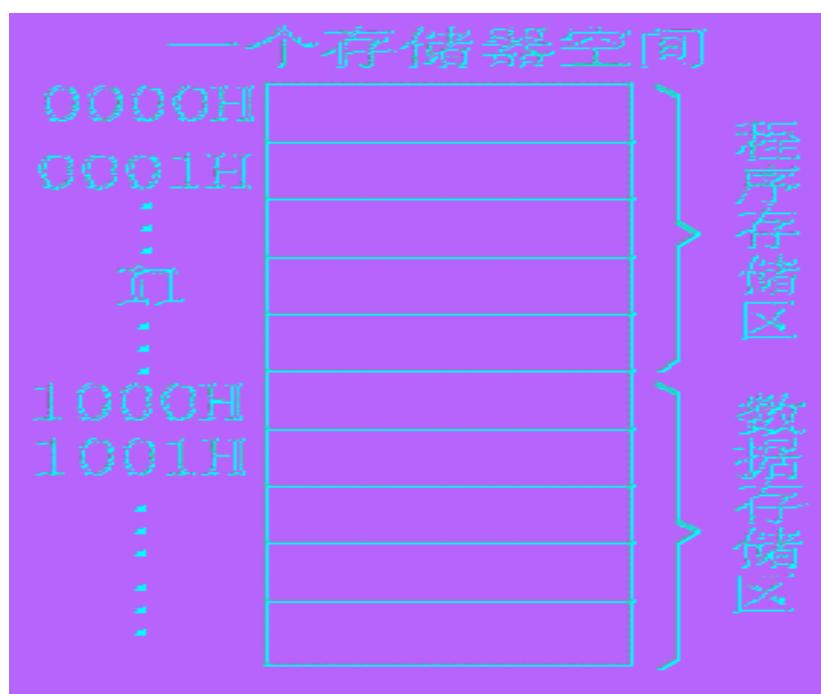
保留的存储单元

存储单元	保留目的
0000H~0002H	复位后初始化引导程序
0003H~000AH	外部中断 0
000BH~0012H	定时器 0 溢出中断
0013H~001AH	外部中断 1
001BH~0022H	定时器 1 溢出中断
0023H~002AH	串行口中断
002BH	定时器 2 中断 (8052 才有)

### 2.2.5 存储器结构特点

普林斯顿结构：

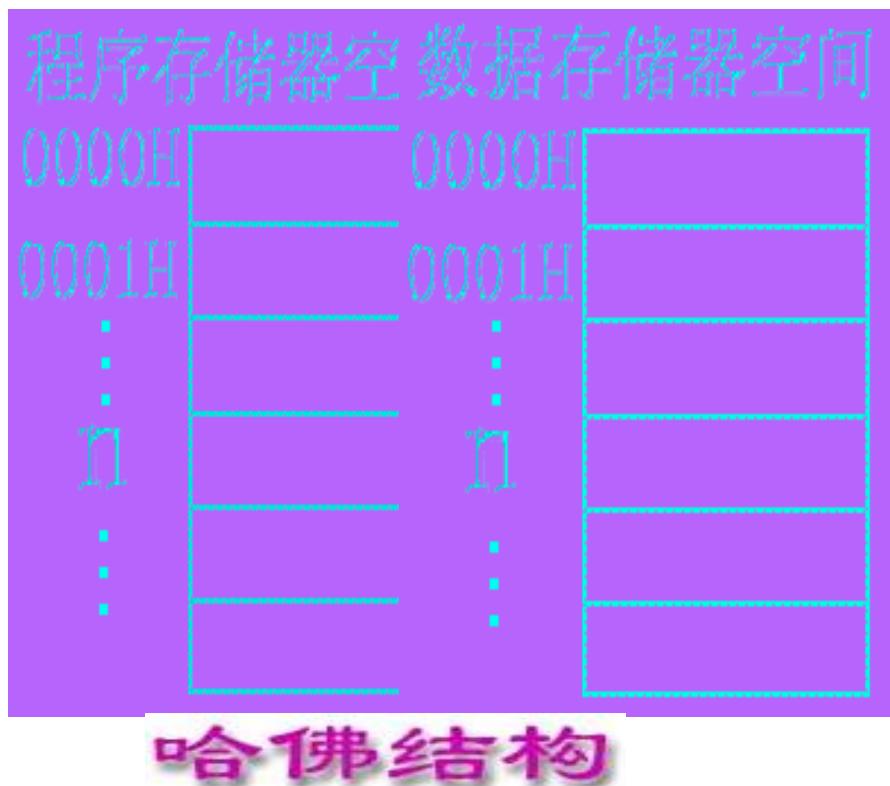
程序和数据共用一个存储器逻辑空间，统一编址。



普林斯顿结构

哈佛结构：

程序与数据分为两个独立存储器逻辑空间，分开编址。



物理上 4 个存储器地址空间：

片内程序存储器

片外程序存储器

片内数据存储器

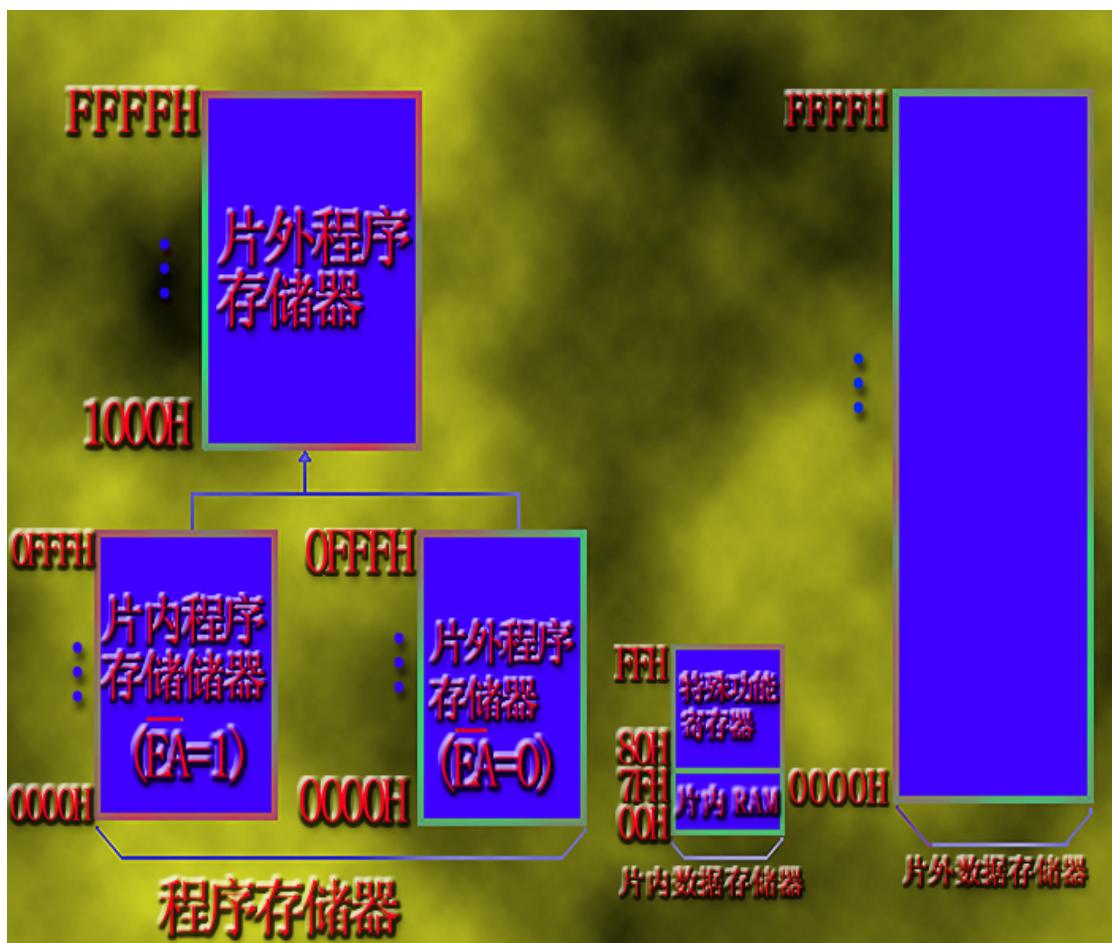
片外数据存储器

逻辑上 3 个存储器地址空间：

64KB 程序存储器

256B 片内数据存储器

64KB 片外数据存储器



存储器:

MCS—51 的程序存储器与数据存储器是分开的（属于哈佛结构），地址空间重叠，最大可扩展到 64KB。

## 1、程序存储器 ROM

### (1) 8031 内部无程序存储器

由于 8031 无片内程序存储器，需外接，因此， $\overline{EA}$  端必须外接低电平。

### (2) 8051、8751 内部有 4KB ROM/EPROM:

$\overline{EA}=0$ ，使用外部程序存储器；

$\overline{EA}=1$ ，使用内部程序存储器 4KB 空间，当 PC 的值超过 4KB 范围时，自动转向外部程序存储器。

## 2、数据存储器 RAM

- (1) 内部 RAM 中低 128B, 00H~7FH;
- (2) 外部 RAM, 可扩至 64KB, 0000H~FFFFH

### 2.3 并行输入/输出口电路

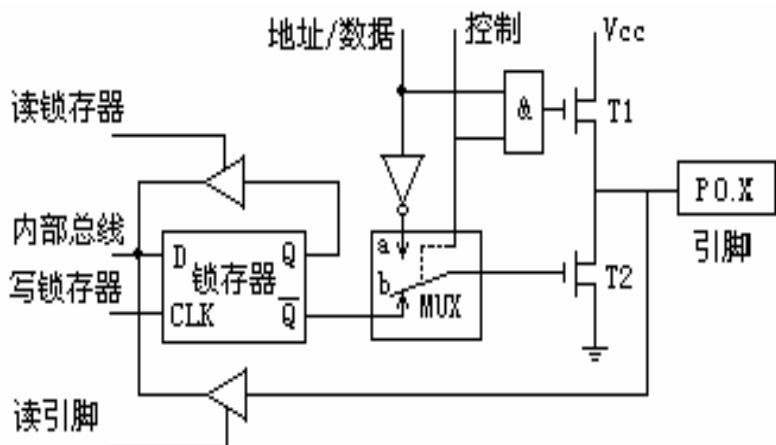
MCS—51 共有四个 8 位的双向并行 I/O 口, 分别记作 P0、P1、P2 和 P3。实际上它们已被归入专用寄存器之列。

口是一个综合概念, 是一个集数据输入缓冲、数据输出驱动及锁存等多项功能为一体的 I/O 电路。对于口有时也称为端口。

#### P0 口:

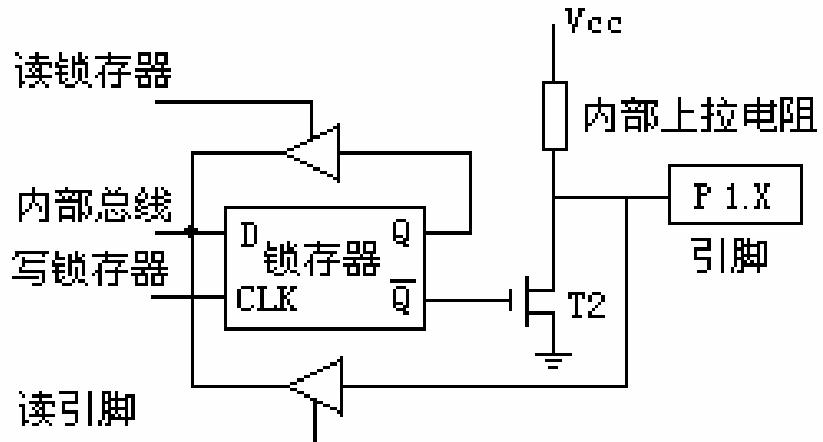
一个数据输出锁存器和两个三态数据输入缓冲器。

一个多路转接电路 MUX 在控制信号的作用下, MUX 可以分别接通锁存器输出或地址/数据线。当作为通用的 I/O 口使用时, 内部的控制信号为低电平, 封锁与门将输出驱动电路的上拉场效应管 (FET) 截止, 同时使 MUX 接通锁存器~Q 端的输出通路。



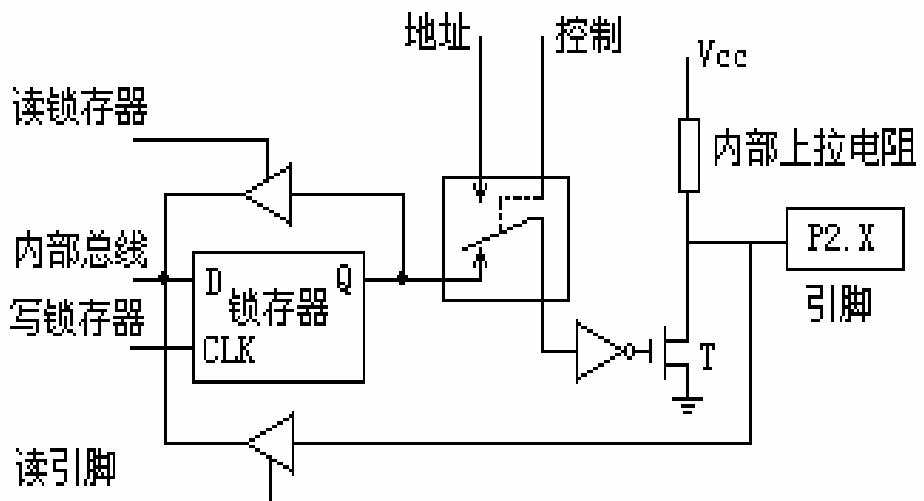
#### P1 口:

作通用 I/O 口使用, 所以在电路结构上与 P0 口有一些不同之处。首先它不再需要多路转接电路 MUX, 其次是电路的内部有上拉电阻。与场效应管共同组成输出驱动电路。作为输出口使用时, 已能向外提供推拉电流负载, 无需再外接上拉电阻。



### P2 口:

P2 口电路中比 P1 口多了一个多路转换电路 MUX，这又正好与 P0 口一样。P2 口也可以作为通用 I/O 口使用。这时多路转接开关倒向锁存器的 Q 端。但通常应用情况下，P2 口是作为高位地址线使用，此时多路转接开关应倒向相反方向。

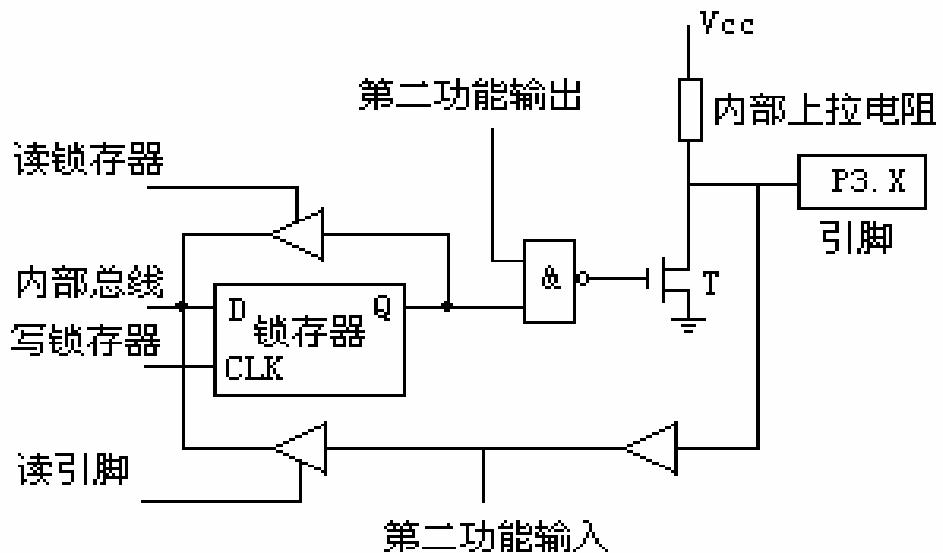


### P3 口:

P3 口的特点在于为适应引脚信号第二功能的需要。

对于第二功能为输出的信号引脚，当作为 I/O 使用时，第二功能信号引线应保持高电平，与非门开通，以维持从锁存器到输出端数据输出通路的畅通。

当输出第二功能信号时，该位的锁存器应置“1”，使与非门对第二功能信号的输出是畅通的，从而实现第二功能信号的输出。



P3 各口线与第二功能表

口线	替代的第二功能
P3.0	RXD (串行口输入)
P3.1	TXD (串行口输出)
P3.2	INT0 (外部中断 0 输入)
P3.3	INT1 (外部中断 1 输入)
P3.4	T0 (定时器 0 的外部输入)
P3.5	T1 (定时器 1 的外部输入)
P3.6	WR (片外数据存储器“写选通控制”输出)
P3.7	RD (片外数据存储器“读选通控制”输出)

### 端口小结:

(1) 系统总线:

地址总线 (16 位): P0 (地址低 8 位)、P2 口 (地址高 8 位)

数据总线 (8 位): P0 口 (地址/数据分时复用, 借助 ALE)

控制总线 (6 根): P3 口的第二功能和 9、29、30、31 脚;

(2) 供用户使用的端口: P1 口、部分未作第二功能的 P3 口;

(3) P0 口作地址/数据时, 是真正的双向口, 三态, 负载能力为 8 个 LSTTL 电路; P1~P3 是准双向口, 负载能力为 4 个 LSTTL 电路。

(4) P0~P3 在用作输入之前必须先写“1”, 即:

$(P0) = FFH \sim (P3) = FFH$  。

## 2.4 电路与时序

### 2.4.1 时钟电路

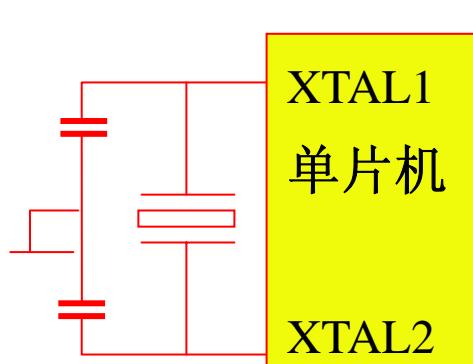
振荡源 (OSCillation)

时钟频率范围要求在  $1.2MHz \sim 12MHz$  之间。

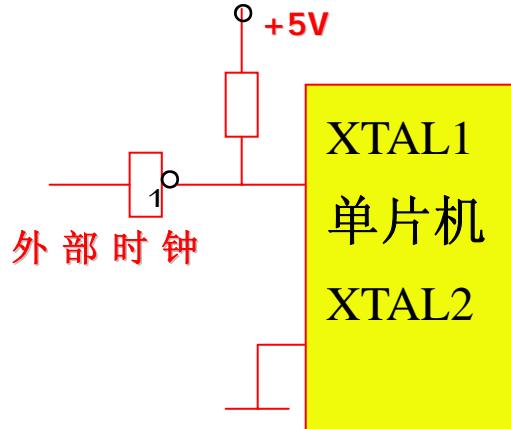
1. 内部时钟方式: 内部一个高增益反相放大器与片外石英晶体或陶瓷谐振器构成了一个自激振荡器。

晶体振荡器的振荡频率决定单片机的时钟频率。

2. 外部时钟方式: 外部振荡器输入时钟信号。



内部时钟方式



外部时钟方式

#### 2.4.2 时序定时单位

时钟周期：振荡频率的倒数。

机器周期：完成一个基本操作所需要的时间。

一个机器周期由 12 个时钟周期组成。

指令周期：一条指令的执行时间。

以机器周期为单位：可包含 1 个~4 个机器周期。

思考题：

设应用单片机晶振频率为 12MHz，问机器周期为多少？指令周期分别为多少？

解：  $f_{osc}=12MHz$

$$MC=12/f_{osc}$$

$$=12/12MHz$$

$$=1 \mu s$$

#### 2.4.3 典型指令时序

MCS—51 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。由于一个状态又包括两个拍节，因此一个机器周期总共有 12 个拍节，分别记作 S1P1、S1P2……S6P2。

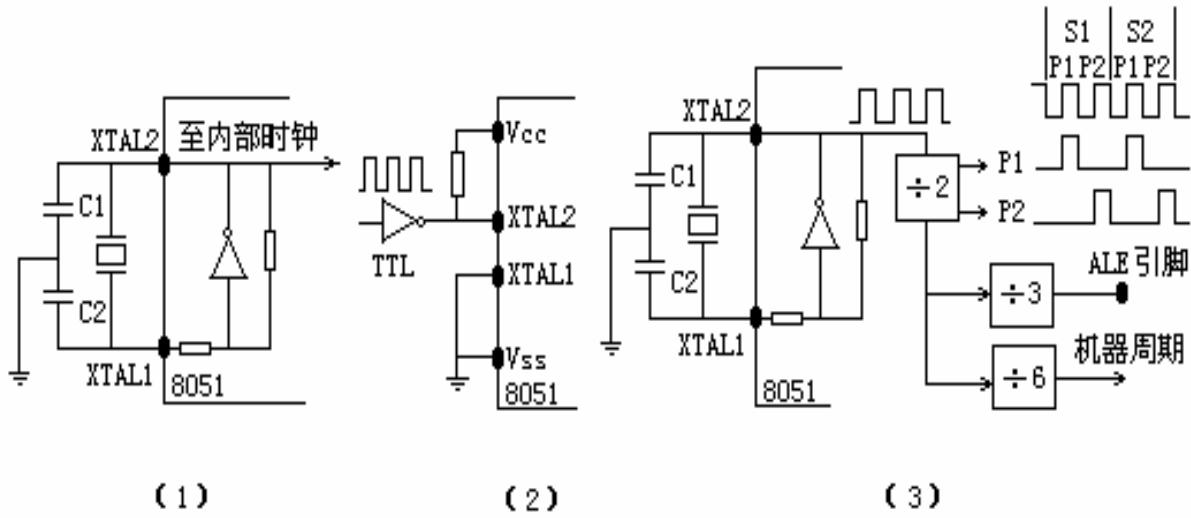
振荡脉冲并不直接使用，由 XTAL2 端送往内部时钟电路 (fosc):

经过 2 分频，向 CPU 提供 2 相时钟信号 P1 和 P2 ( $f_{拍节} = 1/2 f_{osc}$ );

再经 3 分频，产生 ALE 时序 ( $f_{ALE} = 1/6 f_{osc}$ );

经过 12 分频，成为机器周期信号 ( $MC = 12 / f_{osc}$ )，如下图所示。

需要指出的是，CPU 的运算操作在 P1 期间，数据传送在 P2 期间。



片内振荡器和时钟信号的产生

通常，每个机器周期，ALE 两次有效，第 1 次发生在 S1P2 和 S2P1 期间，第 2 次在 S4P2 和 S5P1 期间。

单周期指令的执行 始于 S1P2，这时操作码被锁存到指令寄存器内，读出下字节（应为下一个操作码）是不予考虑的，且程序计数器 PC 并不增量。

访问外部数据存储器的指令 MOVX 的时序，它是一条 单字节双周期指令。在第 1 机器周期 S5 开始时，送出外部数据存储器的地址，随后读或写数据。读写期间在 ALE 端不输出有效信号，在第 2 机器周期，即外部数据存储器已被寻址和选通后，也不产生取指操作。

MCS—51 单片机时序：

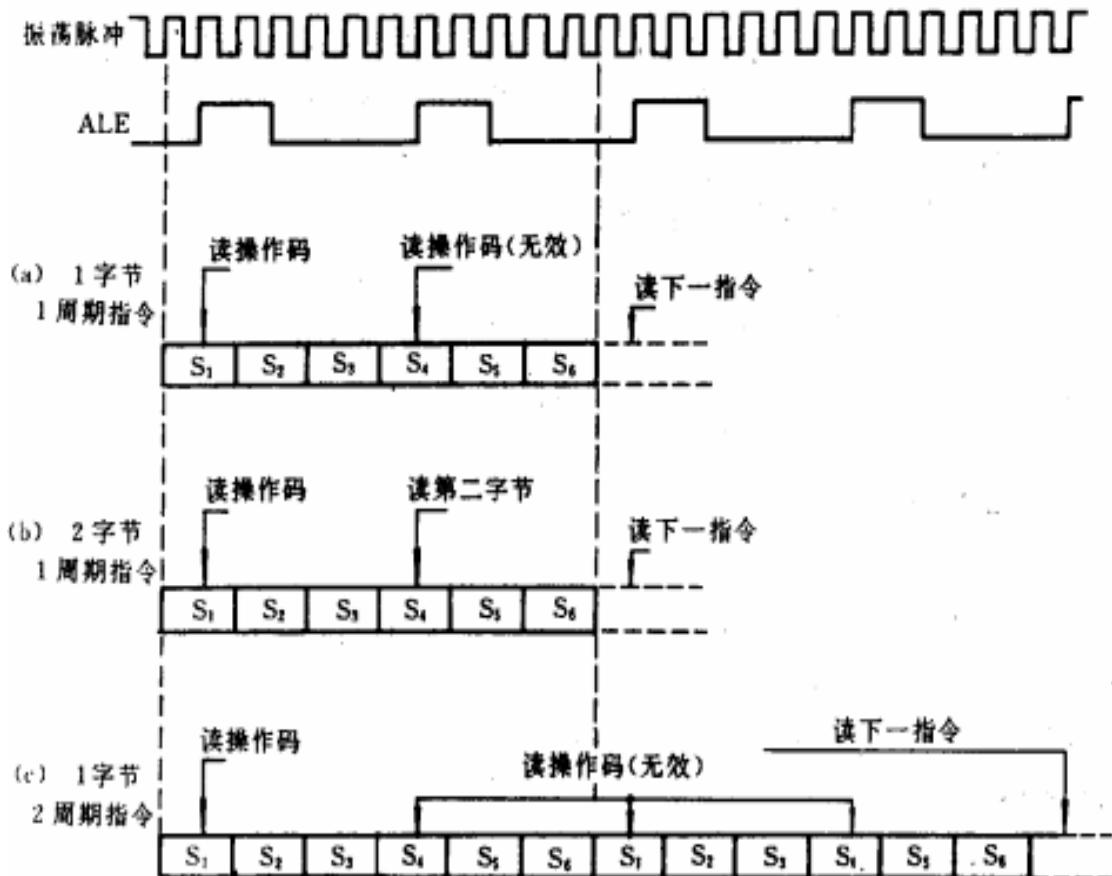


图 1-13 MCS-51 的取指/执行时序

可通过测量 ALE 确定 CPU 是否工作，ALE 有时钟的特点。

## 2.5 工作方式

复位、程序执行、单步执行、掉电保护、低功耗以及 EPROM 编程和校验等六种工作方式。

### 2.5.1 复位方式和复位电路

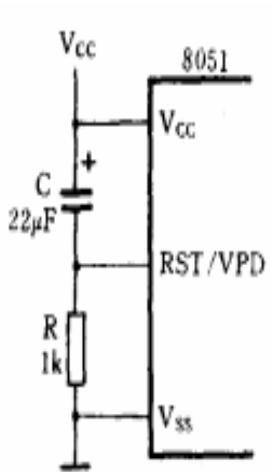
RST 引脚是复位信号的输入端，复位信号是高电平有效，其有效时间应持续 24 个振荡脉冲周期（即二个机器周期）以上。

例：若使用频率为 6MHz 的晶振，则复位信号持续时间应超过\_\_\_\_才能完成复位操作。

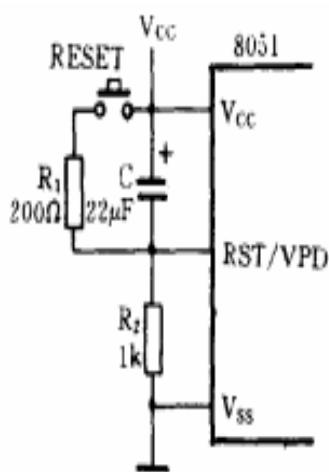
解： fosc=6MHz

$$MC=12/fosc=12/6MHz=2\mu s \quad t=4\mu s$$

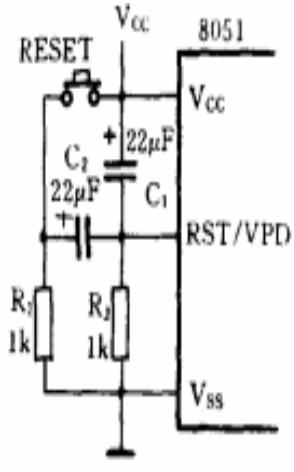
上电自动复位和按键手动复位：



(a) 上电复位



(b) 按键电平复位



(c) 按键脉冲复位

- 上电自动复位——通过电容充电来实现的，Vcc 的上升时间不超过 1ms，就可以实现自动上电复位。
- 按键脉冲复位——利用 RC 微分电路产生的正脉冲来实现的。

#### 一些寄存器的复位状态

寄存器	复位状态	寄存器	复位状态
PC	0000H	TCON	00H
ACC	00H	TL0	00H
PSW	00H	TH0	00H
SP	07H	TL1	00H
DPTR	0000H	TH1	00H
P0~P3	FFH	SCON	00H
IP	$\times \times 000000B$	SBUF	不定
IE	$0 \times 000000B$	PCON	$0 \times \times \times 0000B$
TMOD	00H		

### 2.5.2 程序执行方式

程序执行方式是单片机的基本工作方式。

由于复位后  $PC=0000H$ ，因此程序执行总是从地址  $0000H$  开始，但一般程序并不是真正从  $0000H$  开始，为此就得在  $0000H$  开始的单元中存放一条无条件转移指令，以便跳转到实际程序的入口去执行。

### 2.5.3 掉电保护方式

单片机系统在运行过程中，如发生掉电故障，将会丢失 **RAM** 和寄存器中的程序和数据，其后果有时是很严重的。

掉电保护处理——先把有用信息转存，然后再启用备用电源维持供电。

信息转存：

所谓信息转存是指当电源出现故障时，应立即将系统的有用信息转存到内部 **RAM** 中。信息转存是通过中断服务程序完成的。

系统中设置一个电压检测电路，一旦检测到电源电压下降，立即通过  $INT0/1$  产生外部中断请求，中断响应后执行中断服务程序，并将有用信息送内部 **RAM** 中保护起来，即通常所说的“掉电中断”。

掉电后时钟电路和 **CPU** 皆停止工作，只有内部只 **RAM** 单元和专用寄存器继续工作，以保持其内容。

### 2.5.4 80C51 的低功耗方式

8051 掉电保护方式实际上就是低功耗方式。

CHMOS 的 80C51 却有两种低功耗方式。

即待机方式和掉电保护方式。

待机方式和掉电方式都是由专用寄存器 **PCON**（电源控制寄存器）来控制的。

1、待机方式：

待机方式——振荡器仍然运行。并向中断逻辑、串行口和定时器/计数器电路提供

时钟，CPU 不能工作，与 CPU 有关的如 SP、PC、PSW、ACC 以及全部通用寄存器也都被“冻结”在原状态。

中断方法退出待机方式。中断的同时，PCON.0 被硬件自动清 0，单片机就退出待机方式而进入正常工作方式。其实在中断服务程序只需中安排一条 RETI 指令，就可以使单片机恢复正常工作后返回断点继续执行程序。

## 2、掉电保护方式：

PCON 的 PD 位控制单片机进入掉电保护方式。因此对于象 80C51 这样的单片机。在检测到电源故障时，除进行信息保护外、还应把 PCON.1 位置“1”，使之进入掉电保护方式。此时单片机一切工作都停止，只有内部 RAM 单元的内容被保存。

80C51 单片机除进入掉电保护方式的方法与 8051 不同之外，还有备用电源由 VCC 端引入的特点。VCC 正常后，硬件复位信号维持 10ms 即能使单片机退出掉电方式。

## 最小应用系统：

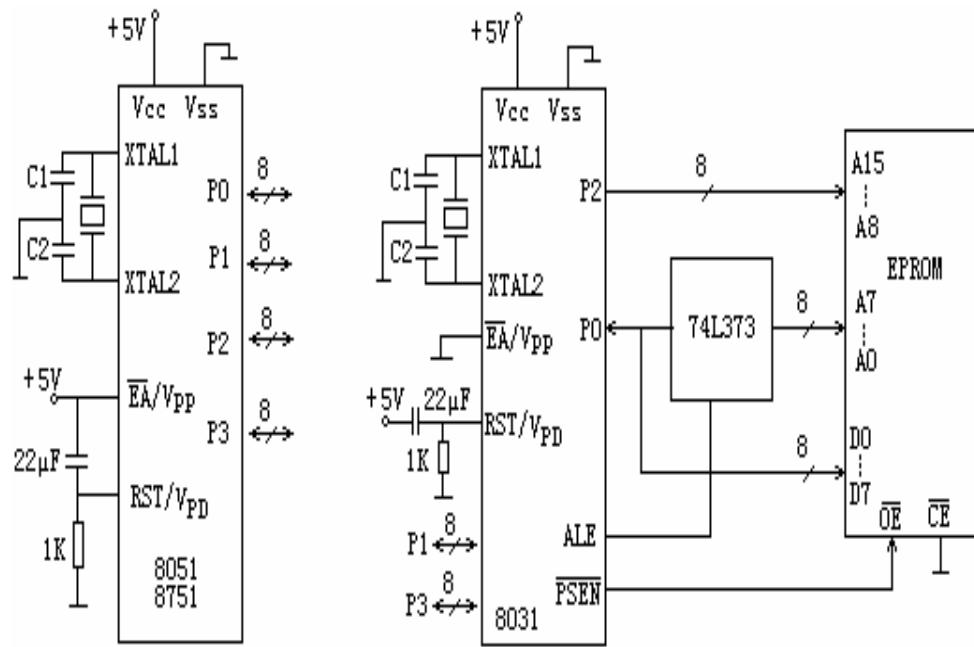
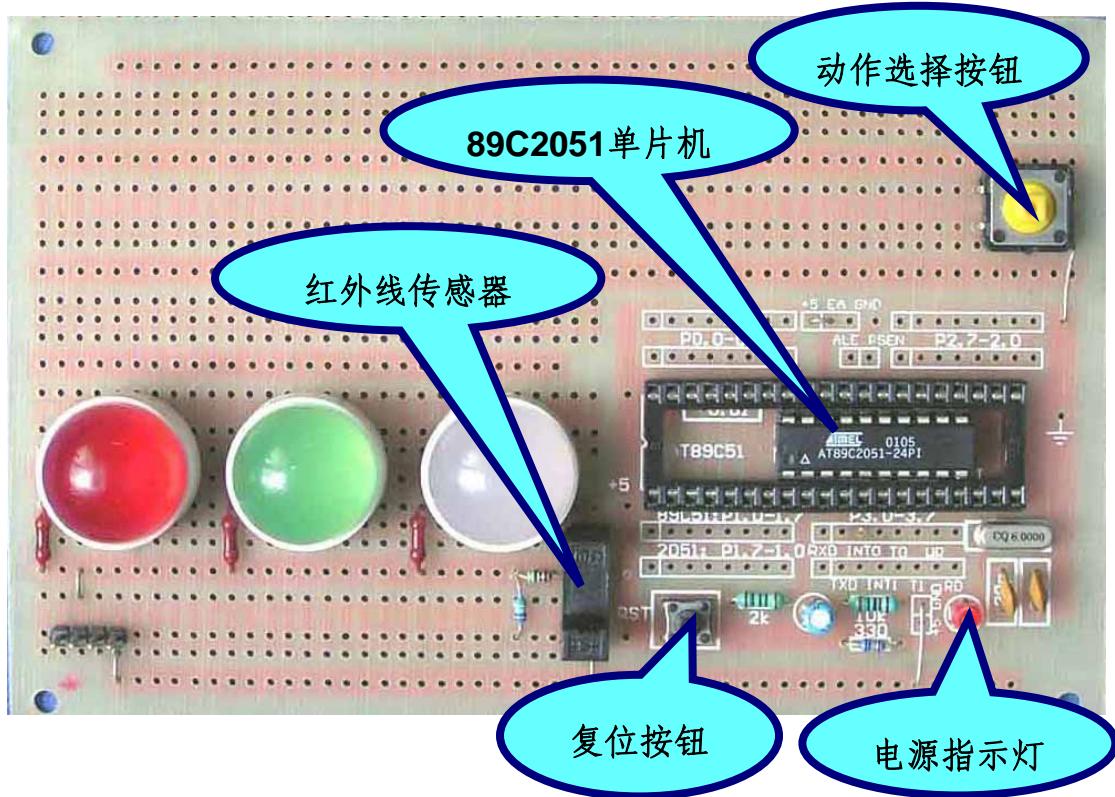


图 2-6 最小应用系统

基本的单片机控制电路板:



小结:

- 1、介绍单片机的编程结构和内部逻辑结构及性能。
- 2、学习了单片机存储器结构特点、内部数据存储器和程序存储器的结构和工作原理。
- 3、单片机的 4 个 8 位并行端口 P0、P1、P2 和 P3 各自的口电路逻辑和功能。
- 4、单片机的时钟电路和时序以及单片机的 6 种工作方式。

练习题:

- (一) 问答题
- (二) 判断题
- (三) 填空题
- (四) 选择题

## 第3章 指令系统

### 一、教学要求：

了解：单片机的寻址方式和指令系统功能，特别是其位寻址功能。

掌握：各种寻址方式，常用指令的功能和使用方法及汇编语言程序设计方法。注意几个中断入口地址在程序存储器中的位置，注意 16 位数据指针 DPTR 和两个 8 位数据 R0、R1 指针的使用方法。

### 二、教学内容：

3.1 单片机指令格式和寻址方式

3.2 单片机指令分类介绍

3.3 单片机指令汇总

### 三、教学重点：

各种寻址方式，常用指令的功能和使用方法及汇编语言程序设计方法。

### 四、教学难点：

注意几个中断入口地址在程序存储器中的位置，注意 16 位数据指针 DPTR 和两个 8 位数据指针 R0、R1 的使用方法。

五、建议学时：3 学时。

### 六、教学内容：

#### 3.1 指令格式和寻址方式

##### 一、汇编语言指令格式：

[标号：]操作码 操作数 1, 操作数 2 [; 注释]

换行表示一条指令结束。

例： LOOP: MOV A, #40H ; 取参数

1、标号：指令的符号地址。

2、操作码：指明指令功能。

3、操作数：指令操作对象。

4、注释行：说明指令在程序中的作用。

操作码和操作数是指令主体。

MOV—move 传送

XCH—exchange 交换

ANL—and logic 与逻辑运算

XRL—exclusive or 异或运算

MUL—multiply 乘法

RR—rotate right 右循环

SJMP—short jump 短跳转

RET—return 子程序返回

二、机器语言指令格式：

操作码 [操作数 1] [操作数 2]

有单字节、双字节和三字节指令。

汇编语言指令中操作码和操作数是指令主体，称为指令可执行部分，指令表中可查出对应指令代码。

举例：

汇编语言： 机器语言：

MOV A, R0 E8H

MOV R6, #32H 7E 32H

MOV 40H, #64H 75 40 64H

三、指令寻址方式：

(一) 操作数类型：

位 (bit) — 位寻址区中的一位二进制数据

字节 (Byte) — 8 位二进制数据

字 (Word) — 16 位双字节数据

## (二) 寻址方式:

### 1、立即寻址方式:

指令中给出实际操作数据 (立即数), 一般用于为寄存器或存储器赋常数初值。

举例:

8 位立即数: MOV A, #40H ; A $\leftarrow$ 40H

16 位立即数: MOV DPTR, #2100H ; DPTR $\leftarrow$ 2100H

### 2、直接寻址方式:

指令操作数是存储器单元地址, 数据放在存储器单元中。

MOV A, 40H ; A $\leftarrow$ (40H)

例: 设存储器两个单元的内容如图所示, 执行指令 MOV A, 40H 后 A = ?



直接寻址方式对数据操作时, 地址是固定值, 而地址所指定的单元内容为变量形式。

### 思考题:

直接寻址方式指令和立即寻址方式指令的形式有什么不同?

### 3、寄存器寻址方式:

指令操作数为寄存器名, 数据在寄存器中。

例: MOV A, R0 ; A $\leftarrow$  (R0)

设指令执行前 A=20H, R0=40H, 执行指令后, A= ?, R0 = ?

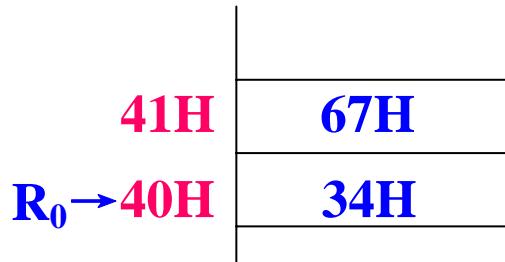
### 4、寄存器间接寻址方式:

指令的操作数为寄存器名，寄存器中为数据地址。

存放地址的寄存器称为间址寄存器或数据指针。

例：MOV A, @R0 ; A←((R0))

设指令执行前 A=20H, R0=40H, 地址为 40H 存储器单元内容如图所示。执行指令后，A= ? R0 = ? (40H)= ?



5、变址间接寻址方式：

数据在存储器中，指令给出的寄存器中为数据的基地址和偏移量。

数据地址 = 基地址 + 偏移量。

说明：1、只对程序存储器；

2、指令形式：MOVC A, @A+DPTR

MOVC A, @A+PC

JMP @A+DPTR

例：MOVC A, @A+DPTR ; A←((A+DPTR))

设指令执行前 A=09H, DPTR=2000H, 存储器单元内容如图所示。执行指令后，  
A= ? DPTR= ?

6、位寻址方式：

指令给出位地址。一位数据在存储器位寻址区。

(1) 内部 RAM 中的位寻址区：字节地址为 20H~2FH；

(2) 专用寄存器的可寻址位：11 个 (83 位)

表示方法：1) 直接使用位地址；如：PSW 的位 6 可表示为 0D6H

- 2) 位名称表示; 或 AC
- 3) 字节地址加位数表示; 或 0D0H.6
- 4) 专用寄存器符号加位数表示。或 PSW.6

例: MOV C, 40H; Cy← (位地址 40H)

设指令执行前 Cy=1, 位地址 40H 存储器单元如图, 执行指令后, Cy= ?



7、相对寻址方式:

目的地址=转移指令地址+转移指令字节数+rel (rel 为偏移量)

当前 PC 值加上指令中规定的偏移量 rel, 构成实际的操作数地址。

例: SJMP rel

操作: 跳转到的目的地址 = 当前 16 位 PC 值 + rel

注意:

- 1) “当前 PC 值” 指程序中下一条指令所在的首地址, 是一个 16 位数;
- 2) 符号 “rel” 表示 “偏移量”, 是一个带符号的单字节数, 范围是: -128~+127(80H~7FH)。

在实际编程中, “rel” 通常用标号代替。

## 3.2 指令分类介绍

指令功能分类:

数据传送、数据操作、布尔处理、程序控制。

### 3.2.1 数据传送指令

实现寄存器、存储器之间的数据传送。

一、内部传送指令： 片内数据存储器数据传送。

二、外部传送指令： 片外数据存储器数据传送。

三、交换指令： 片内数据存储器数据传送。

四、堆栈操作指令： 片内数据存储器数据传送。

五、查表指令： 程序存储器数据传送。

(一)内部传送指令：

实现片内数据存储器中数据传送。

指令格式： MOV 目的操作数，源操作数

寻址方式：立即寻址、直接寻址、寄存器寻址、寄存器间址。

```
MOV A, Rn      ; A←(Rn), Rn=R0~R7  
MOV A, direct  ; A←(direct)  
MOV A, @Ri     ; A←((Ri)), Ri=R0、R1  
MOV A, #data   ; A←data  
MOV Rn, direct ; Rn←(direct)  
MOV @Ri, direct ; (Ri)←(direct)  
MOV direct1, direct2 ; (direct1)←(direct2)  
MOV DPTR, #d1d2 ; DPTR←d1d2
```

指令机器码：

11101rrr E8~EF

11100101 n E5 n

1110011i E6、E7

01110100 d74d

10101rrr n

1010011i n

85 n1 n2

90 d1 d2

习题：找出配对指令，实现反向传送。

例：顺序执行下列指令序列，求每一步执行结果。

MOV A, #30H

MOV 4FH, A

MOV R0, #20H

MOV @R0, 4FH

MOV 21H, 20H

习题：用两种寻址方式实现，将片内 RAM60H 单元的数据传送给累加器 A。

解： MOV A, #60H (×)

或 MOV A, 60H (✓)

结果 A=32H

或 MOV R0, 60H

MOV A, @R0 (×)

或 MOV R0, #60H (✓)

MOV A, @R0

地址	内容
...	
60H	32H
...	
32H	58H

说 明：

1. 一条指令中不能同时出现两个工作寄存器：

非法指令： MOV R1, R2

MOV R2, @R0

2. 间址寄存器只能使用 R0、R1。

非法指令: **MOV A, @R2**

3. SFR 区只能直接寻址, 不能用寄存器间接寻址。

非法指令: **MOV R0, #80H**

**MOV A, @R0**

4. 指令表(P70): B: 指令字节数, M: 机器周期数

只有指令表中的指令才有对应指令代码, 计算机才能执行。编程时, 不能随意创造发明指令。

(二) 外部 RAM 传送指令: (MOVX)

实现片外数据存储器和 A 累加器之间的数据传送。

指令格式: **MOVX 目的操作数, 源操作数**

寻址方式: 片外数据存储器用寄存器间址方式。

1、DPTR 作 16 位数据指针, 寻址 64KB 片外 RAM 空间:

**MOVX A, @DPTR ; A←((DPTR)) (读)**

**MOVX @DPTR, A ; (DPTR)←A (写)**

2、Ri 作 8 位数据指针, 寻址 256B 片外 RAM 空间 (页内寻址):

**MOVX A, @Ri ; A←((P2Ri)) (读)**

**MOVX @Ri, A ; (P2Ri)←A (写)**

例: 实现片外数据存储器数据传送 (2000H)@(2100H)。

**MOV DPTR, #2000H**

**MOVX A, @DPTR**

**MOV DPTR, #2100H**

**MOVX @DPTR, A**

片外数据存储器不能直接寻址。下列为非法指令:

**MOVX A, 2000H**

MOVX 2100H, 2000H

思考题：为什么对 DPTR 的数据传送使用内部传送指令？

习题：将片外 RAM 0000H 单元的数据传送到片内 RAM 的 60H 单元。

### (三) 外部 ROM 传送指令 (查表指令): (MOVC)

实现从程序存储器读取数据到 A 累加器，只能使用变址间接寻址方式。

多用于查常数表程序，可直接求取常数表中的函数值。

#### 1. DPTR 为基址寄存器：

MOVC A, @A+DPTR ; A $\leftarrow$ ((A+DPTR)) (读)

查表范围为 64KB 程序存储器任意空间，称为远程查表指令。

#### 2. PC 为基址寄存器：

MOVC A, @A+PC ; A $\leftarrow$ ((A+PC)) (读)

常数表只能在查表指令后 256B 范围内，称为近程查表指令。

P49:

例 1：以查表方法把累加器中的十六进制数转换为 ASCII 码，并送回累加器中。

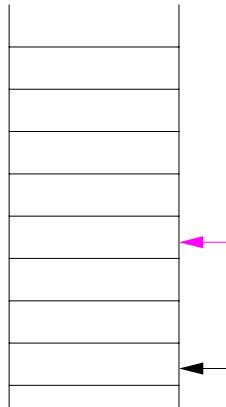
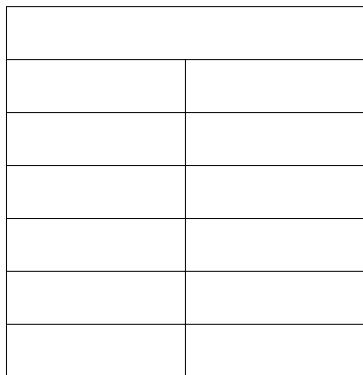
程序如下：

指令地址	源程序
	ORG 2000H
2000	HBA: INC A
2001	MOVC A, @A+PC
2002	RET
2003	DB 30H
2004	DB 31H
2005	DB 32H

.....

2011 DB 45H

2012 DB 46H



### 例 2:

查表法求  $Y=X^2$ 。设  $X(0 \leq X \leq 15)$  在片内 RAM 20H 单元中，要求查表求  $Y$  存入片内 RAM 21H 单元。

## 十六进制与ASCII码对应

### 方法 1:

程序: ORG 1000H

0 30H

SQU: MOV DPTR, #3000H ; 确定表首地址 (基址) 31H

MOV A, 20H ; 取 X (变量: 偏移量)

MOVC A, @A+DPTR ; 查表求  $Y=X^2$  ...

MOV 21H, A ; 保存 Y (结果) 45H

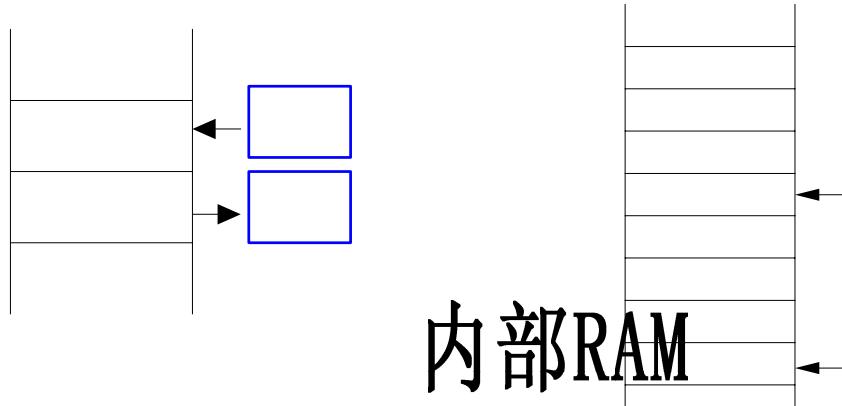
RET ; 子程序结束 46H

..... ; 其它程序段

ORG 3000H ; 常数表格首地址

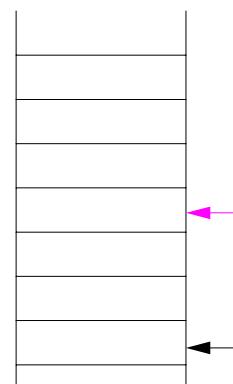
TAB: DB 00, 01, 04, 09, ... , 225 ; 平方表

END



### 方法 2:

指令地址	源程序		
	ORG 21H	Y (结果)	Y A
1000H	SQU: MOV A, 20H	; 取 X	
1002H	ADD A, #3	X (变量)	X A
1004H	MOVC A, @A+PC	; 查表求 Y=X <sup>2</sup> (PC=1005H)	
1005H	MOV 21H, A	• • • ; 存结果	
1007H	RET	; 子程序结束	
1008H	TAB: DB 00, 01, 04 ...	; 平方表	
100BH	DB 09, ... , 225		



思考题：当  $0 \leq X \leq 255$  时，如何用查表法编程求  $Y=X^2$ 。

### (四) 交换指令：

实现片内 RAM 区的数据双向传送。

## 1. 字节交换指令

XCH A, Rn ; A $\longleftrightarrow$ (Rn)

XCH A, @Ri ; A $\longleftrightarrow$ ((Ri))

XCH A, direct ; A $\longleftrightarrow$ (direct)

片内 RAM	
地址	内容
2BH	35H
2AH	38H
...	
20H	

例：设 A=29H，执行指令 XCH A, 2AH 后，A= ? ， (2AH)= ?

习题：将片内 RAM60H 单元与 61H 单元的数据交换。

XCH 60H, 61H; ←对吗？

不对!!

## 2. 半字节交换指令：

XCHD A, @Ri ; A0~3 «((Ri))0~3

SWAP A ; A4~7 « A0~3

例：将片内 RAM 2AH 和 2BH 单元中的 ASCII 码转换成压缩式 BCD 码存入 20H 单元。

单字节 BCD	
0000	千位
0000	百位
0000	十位
0000	个位

片内 RAM	
地址	内容
2BH	35H
2AH	38H
...	
20H	

MOV A, #0	A	00000000	00H		
MOV R <sub>0</sub> , #2AH				R <sub>0</sub>	00111000 38H
MOV R <sub>1</sub> , #2BH				R <sub>1</sub>	00110101 35H
XCHD A, @R <sub>1</sub>	A	00000101	05H	R <sub>1</sub>	00110000 30H
SWAP A	A	01010000	50H		
XCHD A, @R <sub>0</sub>	A	01011000	58H	R <sub>0</sub>	00110000 30H
XCH A, 20H	20H	01011000	58H		

习题：交换片内 RAM 40H 单元和 41H 单元的低半字节。

(五) 堆栈操作指令：

入栈指令：PUSH direct ; SP $\leftarrow$ SP+1, (SP) $\leftarrow$ (direct)

出栈指令：POP direct ; (direct) $\leftarrow$ (SP), SP $\leftarrow$ SP-1

“先加后压” “先弹后减”

例：设 A=02H, B=56H, 执行下列指令后, SP= ? , A= ? , B= ?

SBR: MOV SP, #30H ; 设栈底

PUSH A

PUSH B

MOV A, #00H

MOV B, #01H

...

POP B

POP A

练习：

说明程序执行过程中, SP 的内容及堆栈中内容的改变过程。

程序如下：

MOV SP, #30H

```
MOV A, #20H
MOV B, #30H
PUSH A
PUSH B
.....
POP A
POP B
```

习题：找出指令错误并改正：

1. MOV A, #1000H ; A←1000H (A 装 1 个字节数)
2. MOVX A, 1000H ; A←(1000H)片外 RAM(DPTR、Ri)
3. MOVC A, 1000H ; A←(1000H)片外 ROM(DPTR、PC)
4. MOVX 60H, A ; 片外 RAM(60H)←A (应为 MOV)
5. MOV R0, 60H ; 片内 RAM: (61H)←(60H)  
MOV 61H, @R0 (片内 RAM 可直接寻址)
6. XCH R1, R2 ; R1←R2(必须有 A 参加)
7. MOVX DPTR, #2000H ; DPTR←2000H(应为 MOV)
8. MOVX 60H, @DPTR ; 片内 RAM←片外 RAM (必须有 A 参加)

### 3.2.2 算术运算指令

与数据传送指令不同，多数算术运算指令会影响标志位的状态，即 CPU 执行算术运算指令后，根据数据操作情况自动设置标志位的状态。

MCS-51 的程序状态字寄存器 PSW 为标志寄存器。

其格式如下：字节地址为 D0H

位序	$B_7$	$B_6$	$B_5$	$B_4$	$B_3$	$B_2$	$B_1$	$B_0$
位符号	$C_Y$	AC	$F_0$	$RS_1$	$RS_0$	OV	$F_1$	P

### 1. 标志位(自动设置状态):

#### 1) Cy: 进位标志位

保存运算后最高位的进位/借位状态, 当有进位/借位,  $Cy=1$ , 否则  $Cy=0$ 。

#### 2) AC: 辅助进位标志位

保存低半字节的进位/借位状态, 当 D3 产生进位/借位,  $AC=1$ , 否则  $AC=0$ 。用于十进制调整。

#### 3) OV: 溢出标志位

$OV = Cy7 \oplus Cy6$ , 补码运算产生溢出  $OV=1$ , 否则  $OV=0$ 。

#### 4) P: 奇偶标志位

反映累加器 A 中数据的奇偶性。当 1 的个数为奇数,  $P=1$ , 否则  $P=0$ 。

### 2. 用户选择位(编程设置状态)

#### 1) F0、F1 : 用户自定义标志位。

#### 2) RS1、RS0 :

工作寄存器区选择位。

复位时,  $PSW=00H$

RS1	RS0	工作寄存器
0	0	0 区
0	1	1 区
1	0	2 区
1	1	3 区

例：复位后，设置使用工作寄存器 3 区，其余标志位不变。

解：MOV PSW, #18

位序	<b>B<sub>7</sub></b>	<b>B<sub>6</sub></b>	<b>B<sub>5</sub></b>	<b>B<sub>4</sub></b>	<b>B<sub>3</sub></b>	<b>B<sub>2</sub></b>	<b>B<sub>1</sub></b>	<b>B<sub>0</sub></b>
位符号	<b>C<sub>Y</sub></b>	<b>AC</b>	<b>F<sub>0</sub></b>	<b>RS<sub>1</sub></b>	<b>RS<sub>0</sub></b>	<b>OV</b>	<b>F<sub>1</sub></b>	<b>P</b>

算术运算指令：

完成片内 RAM 和 A 中数据的加减乘除运算。

一. 加减指令（读-修改-写）：

1. 加法指令：

1) 不带进位加法：ADD A, 源操作数

如：ADD A, R2 ； A←A+R2，影响 Cy、OV、AC、P

例 1：已知 A=3BH, PSW=0，执行指令 ADD A, #3BH 后

求：A=76H ? Cy=0 ? OV=0 ? AC=1 ? P=1 ?

PSW=? (01000001=41H)

2) 带进位加法：ADDC A, 源操作数

如：ADDC A, R2 ； A←A+R2+Cy，影响 Cy、OV、AC、P

例：A=9AH, R2=E3H, PSW=0，执行指令 ADC A, R2 后

求：A=? Cy=? OV=? AC=? P=?

PSW=? (10000100=84H)

带进位加法指令 ADC 用于多字节运算：

例：设双字节数 X 存在片内 RAM 41H、40H 单元，Y 存在 42H、43H 单元，编程求 Z=X+Y，并存入片内 RAM 单元 44H、45H、46H。（比较 P52 例 3.3）

MOV A, 40H

ADD A, 42H

```

MOV 44H, A
MOV A, 41H
ADDC A, 43H
MOV 45H, A
MOV A, #00H
ADDC A, #00H
MOV 46H, A
RET

```

片内 RAM	
地址	内容
46H	Z <sub>H</sub>
45H	Z <sub>M</sub>
44H	Z <sub>L</sub>
43H	Y <sub>H</sub>
42H	Y <sub>L</sub>
41H	X <sub>H</sub>
40H	X <sub>L</sub>

BCD 调整指令: BCD: Binary-Coded-Decimal (二进制编码的十进制)

DA A ; 把 A 中按二进制相加后的结果调整成按 BCD 数相加的结果

调整原因: 1、相加结果大于 9, 进入无效编码区;

2、相加结果有进位, 跳过无效编码区。

调整方法: 进行加“6”修正。

十进制加法指令: 仅对加法结果进行调整 ADD A, 源操作数

DA A

进位十进制加 1 法指令:

ADDC A, 源操作数

DA A

作业: BCD 码加法编程。

设 X、Y 为 4 位压缩 BCD 码, 求 Z=X+Y。

2. 减法指令:

SUBB A, 源操作数 ; 带借位减法指令

如: SUBB A, R2 ; A←A-R2-Cy,

; 影响 Cy、OV、AC、P

例: A= 5AH, R2= 5AH, Cy= 0, 执行下列指令

SUBB A, R2

求: A= ? Cy= ? OV= ? P= ? AC= ?

习题: 编程求双字节减法。设 X、Y 存在片内 RAM60H 起始单元, 计算 Z=X-Y。

思考: 有不带借位的减法指令吗?

3. 增量、减量指令:

INC 单操作数

如: INC R2 ; R2←R2+1

DEC 单操作数

如: DEC R2 ; R2←R2-1

INC DPTR ; DPTR←DPTR+1

不影响标志位状态。

注意: 没有指令 DEC DPTR

可用指令 DEC DPL 代替

4. 乘除指令:

MUL AB ; BA←A×B, Cy←0,

; 当积高字节 B=0, OV←0; B≠0, 则 OV←1

DIV AB ; A÷B, A←商, B←余数, Cy←0,

; 当除数 B=0, OV←1; B≠0, 则 OV←0

例: A= 96(60H), B= 192(C0H), 执行指令 MUL AB 后,

求: A= ? B= ? Cy= ? OV= ? P= ?

解:  $96 \times 192 = 18432(4800H)$

例: A= 156(F6H), B= 13(0DH), 执行指令 DIV AB 后

求: A= ? B= ? Cy= ? OV= ? P= ?

解:  $156 \div 13 = 18(12H)$ , 余数=  $12(0CH)$ 。

思考题: 如何实现多字节数据的乘除运算。

### 3.2.3 逻辑运算指令

一、单操作数指令 (A 累加器为操作数):

1、A 清 0 指令: CLR A ; A $\leftarrow$ 0

2、A 取反指令: CPL A ; A $\leftarrow$ A

3、循环移位指令:

1) 8 位循环指令:

RL A ; A 循环左移一位

RR A ; A 循环右移一位

2) 9 位循环指令:

RLC A ; 带 Cy 循环左移一位

RRC A ; 带 Cy 循环右移一位

例: 设 A= 11000101 B, Cy= 0, 分别执行下列单条指令:

CPL A 求: A= ? Cy= ?

RL A 求: A= ? Cy= ?

RLC A 求: A= ? Cy= ?

用 9 位循环指令实现多字节移位:

例: 编程将寄存器 R6R5 中的双字节数 X 左移一位。

CLR C

MOV A, R5

RLC A

MOV R5, A

MOV A, R6

RLC A

MOV R6, A

思考题：如何将寄存器 R6R5 中的双字节数 X 右移一位。

二、双操作数逻辑运算指令(对位逻辑运算):

ANL、ORL、XRL

例：A=01××××××B，×表示随机状态，为 1 或 0，下述一组指令执行后，A 的值如何？

XRL A, #0C0H ; 将累加器 A 的内容 D7、D6 取反

ORL A, #03H ; 将累加器 A 的内容 D1、D0 置 1

ANL A, #0E7H ; 将累加器 A 的内容 D4、D3 清 0

解：执行上述指令后，A=10×00×11B。

习题 1：如何将累加器 A 中的数据高 4 位清 0，低位不变？

习题 2：如何将寄存器 R2 中的数据奇数位取反，偶数位不变？

### 3.2.4 布尔变量操作指令

对片内 RAM 中位寻址区操作。位累加器 Cy 和位地址 bit。

一. 位传送：

MOV C, bit ; Cy←(bit)

MOV bit, C ; (bit)←Cy

例：将位地址 20H 的一位数传送到位地址 30H 中：

MOV C, 20H

MOV 30H, C

26H	1	0	1	1	0	1	0	1
25H	1	0	0	0	0	1	1	0
24H	0	1	1	1	0	0	0	0

## 二. 位清 0、置 1、取反(CLR、SETB、CPL):

CLR C ; Cy $\leftarrow$ 0

CLR 40H ; (位地址 40H) $\leftarrow$ 0

## 三. 逻辑运算(ANL、ORL):

ANL C, 40H ; C $\leftarrow$ C $\wedge$ (40H)

ANL C,  $\sim$ 40H ; C $\leftarrow$ C $\wedge$ (40H)

例: 设 Cy=1, (位地址 40H)=1, 执行指令

ANL C,  $\sim$ 40H 后, Cy= ? , (位地址 40H)= ?

位地址表示法:

位地址 40H, 位寄存器 F0, 字节加位 ACC.0

习题: 设累加器 A 中数据为 29H=00101001B, Cy=0,

执行指令 ORL C, 0E3H 后, Cy= ?

### 3.2.5 转移指令

转移指令通过改写 PC 的当前值, 从而改变 CPU 执行程序的顺序, 使程序发生跳转。

按转移条件分类:

1)无条件转移:

执行无条件转移指令, 程序无条件转移到指定处。

2)条件转移:

指令中给出转移条件, 执行指令时, 先测试条件: 若满足条件, 则程序发生转移; 否则, 仍顺序执行程序。

按转移方式分类:

1)绝对转移: 指令给出转移目的的绝对地址 d2d1, 执行指令后, PC $\leftarrow$ d2d1。

例: 地址 源程序

1000H LJMP 2000H(长转移)

1003H ...

...

2000H ... ; 转移目的指令

2) 相对转移: 指令给出转移目的与转移指令的相对

偏移量 rel, 执行指令后,  $PC \leftarrow PC + rel$ 。

目的地址=PC+字节数+rel

例: 地址 源程序

1000H SJMP 02(短转移) (2字节)

...

1004H ... ; 转移目的指令

1000H	SJMP
	02
1004H	

一、无条件转移指令:

1. 长转移指令:

LJMP addr16(d2d1) ;  $PC \leftarrow d2d1$

指令机器码: 02 d2 d1

指令转移范围: 64KB

2. 绝对转移指令:

AJMP addr11 ;  $PC \leftarrow PC + 2$

(2个字节) ;  $PC10 \sim 0 \leftarrow addr11$

$PC15 \sim 11$  不变

指令机器码:  $addr11 \sim 9 \quad 00001 \quad addr8 \sim 1$

指令转移范围: 2KB(32个)

转移时要求转移前后保持  $PC15 \sim 11$  不变。

例： 1030H AJMP 100H

指令机器码： 21 00H

目的地址为： 1100H

3. 短转移指令：

SJMP rel ; PC→PC+2, PC←PC+rel

指令机器码： 80H rel

相对偏移量 rel 为带符号的 8 位补码数。

rel=[目的地址—(源地址+2)]补

rel 为正数： 向前转移； rel=地址差-2

rel 为负数： 向后转移。 rel=FEH+地址差

指令转移范围： 前 126～后 129 字节即-126D～+129D

相对偏移量 rel 的计算通式：

rel = [目的地址—(转移指令地址+指令字节数)]补

= [目的地址—PC 当前值]补

(1) 由偏移量 rel 计算目的地址；

(2) 由目的地址计算偏移量 rel。

编程时，用标号代替转移目的地址，转移指令的操作数交给汇编程序计算。

LJMP NEXT (AJMP NEXT/SJMP NEXT)

...

NEXT:

例 1： 计算转移指令的目的地址。

(1) 835AH SJMP 35H

解： rel=35H = 0011 0101B 为正数，因此程序向前转移。

目的地址 = 835AH + 02H + 35H

$$= 8391H$$

(2) 835AH SJMP 0E7H

解: rel=0E7H = 1110 0111B 为负数, 因此程序向后转移。

$$\text{目的地址} = 835AH + 02H + 0E7H$$

$$= 8343H$$

例 2: 计算转移指令的相对偏移量 rel, 并判断是否超出转移范围?

指令地址 源程序

2130H SJMP NEXT

...

2150H NEXT: MOV A, R2

$$\text{相对偏移量 rel} = 2150H - (2130H + 2)$$

$$= 001EH = +30D \text{ (未超出转移范围),}$$

$$rel = 1EH \text{ 求出指令机器码: } 80\ 1EH$$

例 3: 求原地踏步指令的指令代码:

HERE: SJMP HERE (或 SJMP \$)

$$\text{相对偏移量} = [2000H - (2000H + 2)]\text{补} = FEH,$$

$$rel = FEH \text{ 求出指令代码为: } 80\ FEH$$

习题: 计算程序中转移指令的相对偏移量 rel, 并判断是否超出转移范围。

地址 源程序

2130H LOOP: ...

...

21B0H SJMP LOOP

$$\text{相对偏移量} = [2130H - (21B0H + 2)]\text{补} = (-130)\text{ 补}$$

rel 已超出转移范围 (-126D ~ +129D)

#### 4. 间接转移指令(多分支转移指令):

JMP @A+DPTR ; PC $\leftarrow$ ((A+DPTR))

指令机器码 73H, 指令转移范围 64KB。

应用: 处理功能键。

要求不同功能键执行不同程序段。设每个功能键对应一个键值  $X(0 \leq X \leq FH)$ 。

设  $X$  已存入片内 RAM 的 40H 单元中。

若  $X=0$ , 则执行程序段 FUNC0;

若  $X=1$ , 则执行程序段 FUNC1;

... 。

KEY: MOV DPTR, #KTAB

MOV A, 40H

ADD A, A

JMP @A+DPTR

KTAB: AJMP FUNC0

AJMP FUNC1

...

FUNC0: ...

FUNC1: ...

#### 二、条件转移指令:

条件转移指令形成程序的分支, 赋予计算机判断决策能力。

转移条件: 1)标志位的状态;

2)位地址中的状态。

#### 1、A 判零转移指令:

JZ rel ; PC $\leftarrow$ PC+2, ; 若 A=00H, PC $\leftarrow$ PC+rel(转移),

若  $A \neq 00H$ , PC 不变(不转移)

2、判 Cy 转移指令：

JC rel ; Cy=1 则转移, Cy=0 不转移  
JNC rel ; Cy=0 则转移, Cy=1 不转移

3、判位转移指令：

JB bit, rel ; (bit)=1 转移, 否则不转移  
JNB bit, rel ; (bit)=0 转移, 否则不转移

4、判位清 0 转移指令：

JBC bit, rel ; (bit)=1 转移, 且 (bit)=0, 否则不转移

5、比较不相等转移指令：

CJNE 操作数 1, 操作数 2, rel  
CJNE A, direct, rel ;  $PC \leftarrow PC + 3$   
; 若  $A = (direct)$ , 顺序执行, 且  $Cy = 0$ 。  
若  $A \neq (direct)$ , 则  $PC \leftarrow PC + rel + 3$   
且当  $A > (direct)$ ,  $Cy = 0$ ,  
当  $A < (direct)$ ,  $Cy = 1$ 。

相当于两个操作数相减, 仅影响标志状态, 不保存结果。

6、循环转移指令：

DJNZ 操作数, rel  
DJNZ R2, rel ;  $PC \leftarrow PC + 2$ ,  $R2 \leftarrow R2 - 1$   
; 若  $R2 \neq 0$ ,  $PC \leftarrow PC + rel$ ,  
; 若  $R2 = 0$ , PC 不变。

例：用于循环结构程序。设要求程序循环执行 100 次。

MOV R2, #100 ; 设循环计数器初值

```

LOOP:      ...          ; 多次循环程序段

DJNZ    R2, LOOP      ; 循环控制

...
; 循环结束

```

例：设单片机的晶振频率为 6MHz，编写一段延时程序约 100ms 的子程序。

```

Delay:    MOV    R7, #64H    ; 设循环计数器初值(100 次)

LOOP:    MOV    R6, #FAH    ; 循环 250 次( $250 \times 4 = 1\text{ms}$ )

DJNZ    R6, $          ; 循环控制

DJNZ    R7, LOOP

RET

```

$$T = 12/6 \text{ MHz} = 2 \mu \text{ s}$$

$$\begin{aligned}
t &= 2 \mu \text{ s} + 100 \times (2 \mu \text{ s} + 1\text{ms} + 2 \times 2 \mu \text{ s}) + 4 \mu \text{ s} \\
&= 100.606 \text{ ms}
\end{aligned}$$

习题：当循环计数器初值为 0，循环次数有多少？

### 3.2.6 子程序调用和返回指令

子程序调用和返回指令也使程序发生转移。

子程序调用过程：

与转移指令不同：转移时，先用堆栈保存当前地址。

一、长调用指令：

```

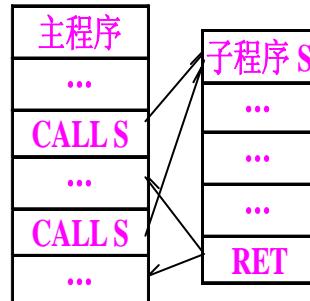
LCALL  addr16    ; PC  $\leftarrow$  PC+3,
                  ; SP  $\leftarrow$  SP+1, (SP)  $\leftarrow$  PC0~7;
                  ; SP  $\leftarrow$  SP+1, (SP)  $\leftarrow$  PC8~15;
                  ; PC  $\leftarrow$  addr16

```

addr16 为子程序起始地址，编程时可用标号代替。

指令机器码: 12 addr8~15 addr0~7

指令调用范围: 64KB



二、绝对调用指令:

ACALL addr11 ; PC $\leftarrow$ PC+2  
; SP $\leftarrow$ SP+1, (SP)  $\leftarrow$ PC0~7,  
SP $\leftarrow$ SP+1, (SP) $\leftarrow$ PC8~15  
; PC10~0 $\leftarrow$ addr11  
PC15~11 不变

addr11 为子程序首地址

指令机器码: addr11~9 10001 addr8~1

指令调用范围 2KB。

三、子程序返回指令:

RET ; PC15~8 $\leftarrow$ (SP), SP $\leftarrow$ SP-1,  
PC7~0 $\leftarrow$ (SP), SP $\leftarrow$ SP-1

指令机器码: 22H

RET 指令从堆栈弹出保存的 PC 地址, 实现子程序返回。

例: 子程序嵌套:

MAIN: MOV SP, #30H ; 设置栈底

...

LCALL SUB ; 调用子程序

...

SUB: ... ; 子程序段  
 ...  
 RET ; 返回主程序

程序存储器	
2000H	MOV SP, #30H
	...
208FH	LCALL 2100H
	...
2100H	...
	...
2150H	LCALL 2200H
	...
21FFH	RET
2200H	...
	...
2250H	RET

注意：

- 1、子程序起始指令要使用标号，用作子程序名。
- 2、执行返回指令 RET 之前，保证栈顶内容为主程序返回地址，以便正确返回主程序。

常用格式：

MAIN: ... ; 主程序  
 LCALL SUBR ; 调用 SUBR  
 ...  
 ...  
 SUBR: ... ; 子程序首地址  
 ...  
 RET ; 子程序返回

### 3.2.7 I/O 口访问指令使用说明

- 1、可以按口寻址，进行字节操作；

如：MOV Pm, A

可以按口线寻址，进行位操作。

如：MOV Pm.n, C

2、没有专门的输入/输出指令，均使用 MOV 传送指令来完成：

输入：用 MOV 指令把各口线的引脚状态读入；

输出：用 MOV 指令把输出数据写入各口线电路的锁存器。

3、在进行引脚数据输入操作之前，必须先向电路中的锁存器写入“1”，使 FET 截止，以避免锁存器为“0”状态时对引脚读入的干扰。

## 小结：

1、指令的寻址方式、格式、功能和使用方法。

7 种寻址方式：寄存器寻址；

寄存器间接寻址；

直接寻址；

立即寻址；

变址寻址；

相对寻址；

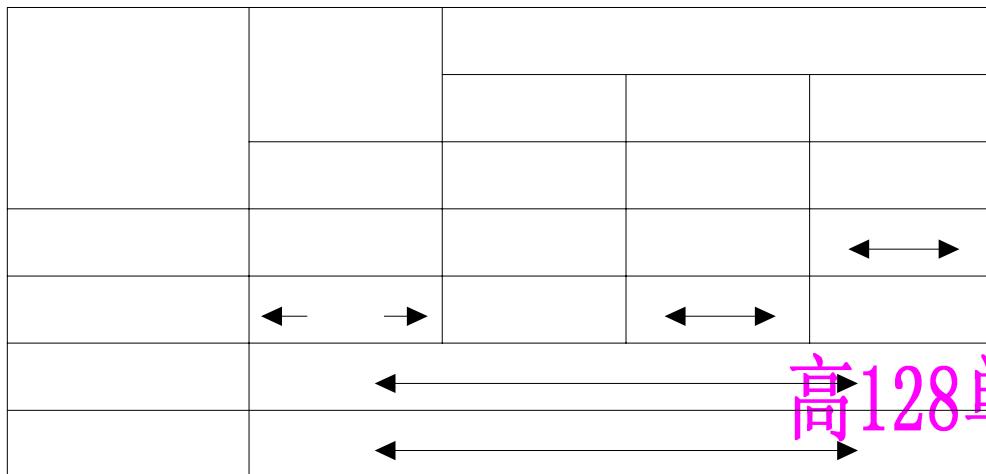
位寻址。

2、指令分类介绍：详见 P70 表。

3、寻址方式：

(1) 对程序存储器（内、外）：只能变址寻址 MOVC

(2) 对内部数据存储器：MOV

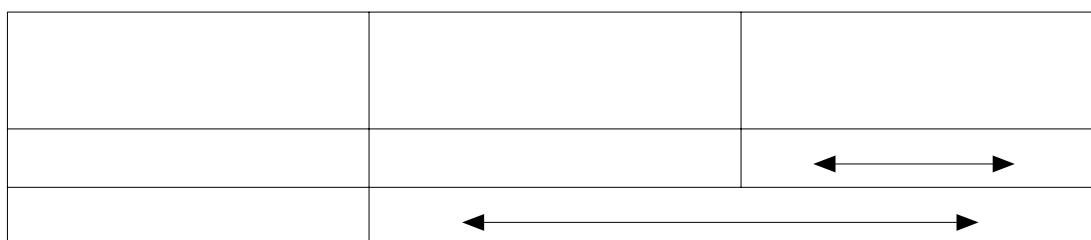


高128单元

(SFR)

用户R

(3) 对外部数据存储器: 只能寄存器间址 MOVX



位寻址方式

部分

练习题:

(一) 填空题

(二) 选择题

直接寻址方式

间接寻址方式

寄存器间接寻址方式

高地址单元  
(0FFFFH)

以R<sub>0</sub>或R<sub>1</sub>作间址寄存器

以DPTR作间址寄存器

## 第4章 汇编语言程序设计

### 一、教学要求：

掌握：单片机汇编语言程序的基本结构以及加、减、乘、除运算及数制转换程序设计。

理解：查表、差值运算、译码等简单程序设计方法。

### 二、教学内容：

- 4. 1 汇编语言程序设计概述
- 4. 2 单片机汇编语言程序的基本结构形式
- 4. 3 单片机汇编语言程序设计举例
- 4. 4 单片机汇编语言的伪指令
- 4. 5 单片机汇编语言源程序的编辑和汇编

### 三、教学重点：

单片机汇编语言程序的基本结构及程序设计方法。

### 四、教学难点：

加、减、乘、除运算及数制转换程序设计。

### 五、建议学时：4学时。

### 六、教学内容：

#### 4-1 汇编程序约定

##### 汇编语言程序：

用汇编语言编写的、完成特定功能的指令序列。

##### 汇编程序：

能将汇编语言源程序转换成机器语言目标程序的系统软件。

汇编语言程序到机器语言程序的转换过程称为汇编。

##### 1、手工汇编：人工查指令表汇编。用于设计短小程序或调试程序的场合。

2、机器汇编：用汇编程序进行汇编。

源程序使用机器汇编要考虑汇编程序的约定：

1) 按指令格式和语法规则编写程序。

常数的表示：

十进制数：20

十六进制数：87H, 0F0H

二进制数：01011001B

字符：‘H’

字符串：“Hello”

2) 使用伪指令提供汇编信息。

汇编的主要任务：

1) 确定程序中每条汇编语言指令的指令机器码。

2) 确定每条指令在存储器中的存放地址。

3) 提供错误信息。

4) 提供目标执行文件 (\*.OBJ/\*.HEX)和列表文件 (\*.LST)。

一、汇编语言指令类型：

1. 机器指令：

指令系统中的全部指令，每条指令有对应的机器代码。

2. 伪指令：

汇编控制指令，仅提供汇编信息，没有指令代码。

3. 宏指令：

宏汇编功能：将需要多次反复执行的程序段定义成一个宏指令名（宏定义），编程时，可在程序中使用宏指令名来替代一段程序（宏调用）。

宏定义过程：

宏指令名 MACRO 形式参数

... ; 定义程序段

ENDM

宏调用过程:

...

宏指令名 实际参数

...

宏指令名 实际参数

二、汇编控制指令(伪指令):

常用伪指令及其功能:

1.ORG—起始地址指令: 指明程序和数据块起始地址。

指令地址 机器码 源程序

ORG 2000H

2000H 78 30 MAIN: MOV R0, #30H

2002H E6 MOV A, @R0

...

ORG 3000H

3000H 23 DB 23H, 100, 'A'

3001H 64

3002H 41

2.DB—定义字节型常数指令。

例: DB 12H, 100, 'A'

3. DW — 定义字型常数指令。

例: DW 1234H, 5678H

4. EQU — 等值。为标号或标识符赋值。

X1 EQU 2000H

X2 EQU 0FH

...

MAIN: MOV DPTR, #X1

ADD A, #X2

5. END — 结束汇编指令。

例: START: ...

...

END START

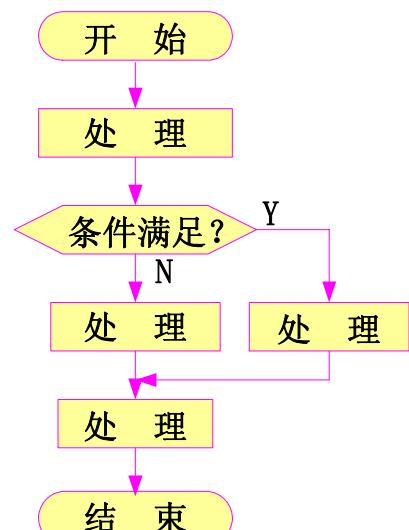
## 4-2 汇编语言程序设计步骤

- 一、确定方案和计算方法;
- 二、了解应用系统的硬件配置、性能指标;
- 三、建立系统数学模型, 确定控制算法和操作步骤;
- 四、画程序流程图;  
表示程序结构和程序功能。
- 五、编制源程序。

- 1.合理分配存储器单元和了解 I/O 接口地址。
- 2.按功能设计程序, 明确各程序之间的相互关系。
- 3.用注释行说明程序, 便于阅读、修改和调试。

### 常用程序结构:

顺序程序、分支程序、循环程序、子程序。



### 4-3 顺序程序

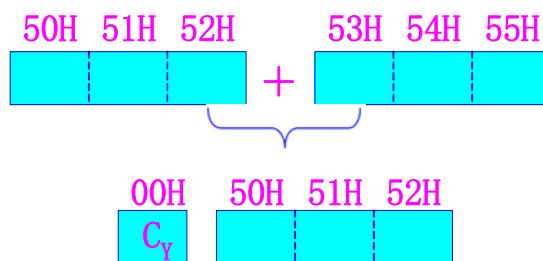
顺序程序又称简单程序，程序走向只有一条路径。

例：双字节变补程序(设数据在 R4R5 中):

```
MOV A, R5      ; 取低字节
CPL A
ADD A, #01H    ; 低字节变补
MOV R5, A
MOV A, R4      ; 取高字节
CPL A
ADDC A, #00H   ; 高字节变补
MOV R4, A
```

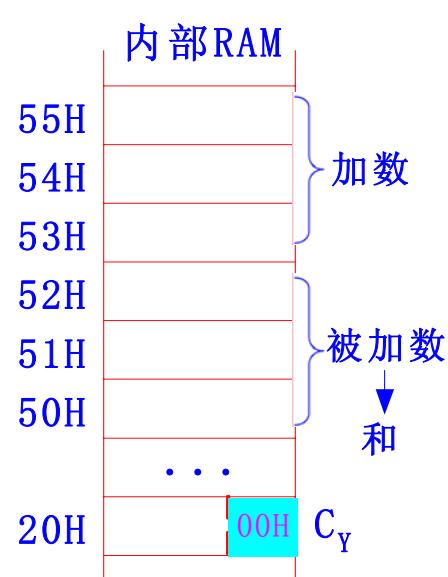
#### P84: 例题 (顺序)

[例 4.1] 三字节无符号数相加，其中被加数在内部 RAM 的 50H、51H 和 52H 单元中；加数在内部 RAM 的 53H、54H 和 55H 单元中；要求把相加之和存放在 50H、51H 和 52H 单元中，进位存放在位寻址区的 00H 位中。



程序：

```
MOV R0, #52H
MOV R1, #55H
MOV A, @R0
ADD A, @R1
```



```
MOV    @R0 , A
DEC    R0
DEC    R1
MOV    A, @R0
ADDC  A, @R1
MOV    @R0, A
DEC    R0
DEC    R1
MOV    A, @R0
ADDC  A, @R1
MOV    @R0, A
CLR    A
ADDC  A, # 00H
MOV    R0, # 00H
MOV    @R0, A
```

例：压缩式BCD码分解成为单字节BCD码。

```
MOV    R0, #40H      ; 设指针
MOV    A, @R0       ; 取一个字节
MOV    R2, A        ; 暂存
ANL    A, #0FH      ; 高半字节清 0
INC    R0
MOV    @R0, A        ; 保存数据个位
MOV    A, R2
SWAP  A            ; 十位换到低半字节
```

```

ANL A, #0FH
INC R0
MOV @R0, A      ; 保存数据十位

```

片内 RAM	
42H	0 十
41H	0 个
40H	

#### 4-4 分支程序

由条件转移指令构成程序判断框部分，形成程序分支结构。

##### 4-4-1 单重分支程序

一个判断决策框，程序有两条出路。

两种分支结构：如右图。

例：求 R2 中补码绝对值，正数不变，负数变补。

```

MOV A, R2
JNB ACC.7, NEXT ; 为正数？为 0 跳
CPL A           ; 负数变补
INC A
MOV R2, A
NEXT: SJMP NEXT ; 结束

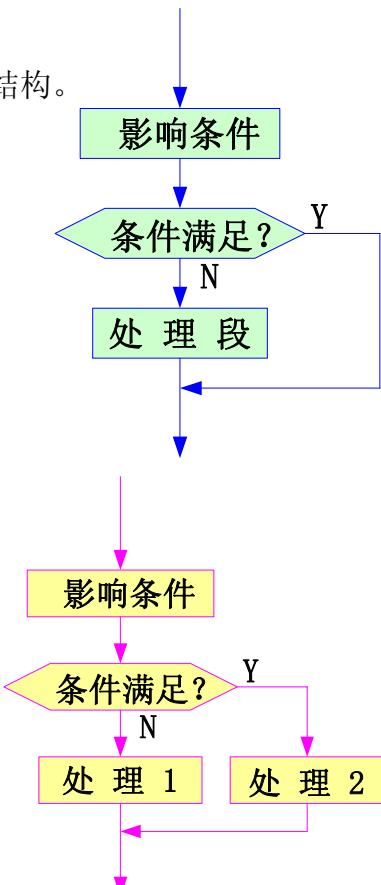
```

P85: 例题（单分支）

[例 4.2]假定在外部 RAM 中有 ST1、ST2 和 ST3 共 3 个连续单元，其中 ST1 和 ST2 单元中分别存放着两个 8 位无符号二进制数，要求找出其中的大数并存入 ST3 单元中。

START: CLR C

MOV DPTR, #ST1



```

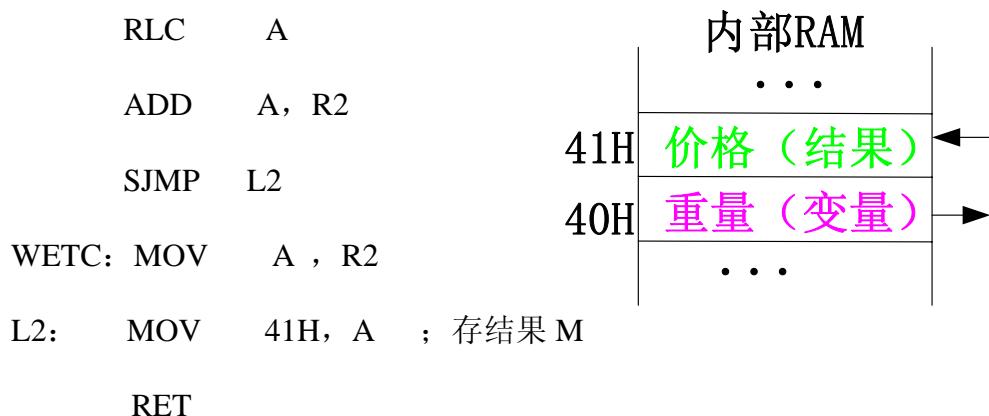
        MOVX    A, @DPTR
        MOV     R2, A
        INC     DPTR
        MOVX    A, @DPTR
        SUBB    A, R2
        JNC    BIG1
        XCH    A, R2
        BIG0: INC     DPTR
        MOVX    @DPTR, A
        RET
        BIG1: MOVX    A, @DPTR
        SJMP    BIG0

例：行李计价：当  $G \leq 5$ ,  $M = G \times 3$ ;  

当  $G > 5$ ,  $M = G \times 3 + (G - 5) \times (5 - 3)$  。

FRT: MOV    A, 40H           ; 取行李重量计价单位 G
      MOV    R3, A
      MOV    B, #03H           ;  $M = G \times 3$ 
      MUL    AB
      MOV    R2, A           ; 暂存 3G
      MOV    A, R3           ; 取回 G
      CJNE   A, #05H, L1      ;  $G = 5 ? G \neq 5$  跳 L1
      SJMP   WETC
      L1:  JC    WETC           ; 是，转至 WETC (C=1 即  $G < 5$  跳)
      SUBB   A, #05H          ; 否则  $M = 3G + 2(G - 5)$ 

```



#### 4-4-2 多重分支程序

一、多次使用条件转移指令，形成两个以上判断框。

例：求符号函数  $Y=SGN(X)$

$$SGN(X) = \begin{cases} +1 & (\text{当 } X > 0) \\ 0 & (\text{当 } X = 0) \\ -1 & (\text{当 } X < 0) \end{cases}$$

SYMB: MOV A, 40H ; 取 X

JZ STOR ; X=0 跳，Y=X

JB ACC.7, MINUS ; X<0 (A.7=1 跳)

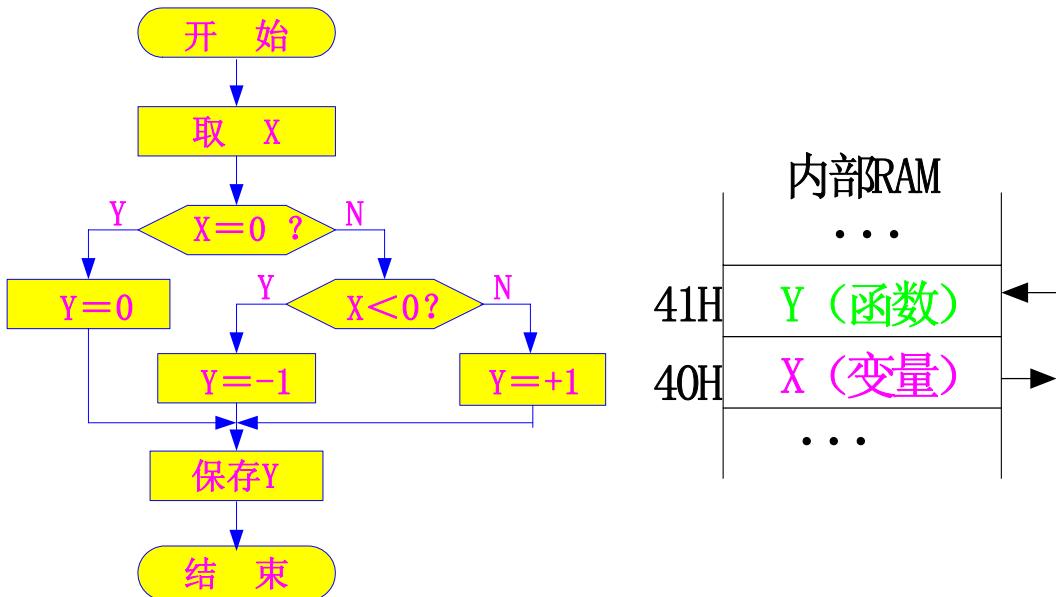
MOV A, #01H ; X>0, Y=+1

SJMP STOR

MINUS: MOV A, #0FFH ; X<0, Y=-1

STOR: MOV 41H, A ; 保存 Y

RET



### P86: 例题 (多分支)

[例4.3]某温度控制系统，采集的温度值 (Ta) 放在累加器A中。此外，在内部RAM54H单元存放控制温度下限值 (T54)，在55H单元存放控制温度上限值 (T55)。

若  $Ta > T55$ ，程序转向 JW (降温处理程序)；

若  $Ta < T54$ ，则程序转向 SW (升温处理程序)；

若  $T55 \geq Ta \geq T54$ ，则程序转向 FH (返回主程序)。

思路:  $Ta=T55?$   $\begin{cases} =: \text{去FH} \\ \neq: \begin{cases} Ta > T55: \text{去JW} \\ Ta < T55: Ta=T54? \begin{cases} =: \text{去FH} \\ \neq: \begin{cases} Ta < T54: \text{去SW} \\ Ta > T54: \text{去FH} \end{cases} \end{cases} \end{cases} \end{cases}$

算法: (1)  $Ta > T55$ : 去JW

(2)  $Ta < T54$ : 去SW

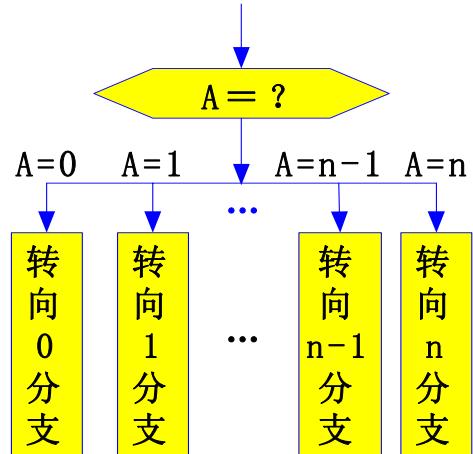
(3)  $T55 \geq Ta \geq T54$ : 去FH

程序：

```

CJNE      A, 55H, LOOP1
AJMP      FH
LOOP1: JNC      JW
CJNE      A, 54H, LOOP2
AJMP      FH
LOOP2: JC       SW
FH:       RET

```



二、按分支号转移。

如：当分支号=0，程序转移到ADDR0处；

当分支号=1，程序转移到ADDR1处；

...。

1、利用查地址表法：（P87）

[例4.4]有BR0、BR1、BR2和BR3共4个分支程序段，各分支程序段的功能依次是从内部RAM256B范围取数、从外部RAM低256B范围取数、从外部RAM4KB范围取数和从外部RAM64KB范围取数。

并假定R0中存放取数地址低8位地址，R1中存放高8位地址，R3中存放分支序号值。

假定以BRTAB作差值表首地址，BR0\_BRTAB ~ BR3\_BRTAB为差值。

分析：差值表=分支入口地址-该表首址

```

程序: MOV A, R3
      MOV DPTR, #BRTAB
      MOVC A, @A+DPTR
      JMP @A+DPTR
BRTAB: DB BR0_BRTAB

```

3003H	60H
3002H	40H
3001H	20H
3000H	10H
首址	DPTR

```

DB      BR1_BRTAB

DB      BR2_BRTAB

DB      BR3_BRTAB

BR0:  MOV   A, @R0
      SJMP  BRE

BR1:  MOVX A, @R0
      SJMP  BRE

BR2:  MOV   A, R1
      ANL   A, #0FH
      ANL   P2, #0F0H
      ORL   P2, A
      MOVX A, @R0

BR3:  MOV   DPL, R0
      MOV   DPH, R1
      MOVX A, @DPTR

BRE:  SJMP  $

```

2、使用转移指令表法。用分支转移指令 AJMP ...

**P88: 例题 (查转移指令表)**

[4.5]假定键盘上有 3 个操作键, 功能说明如下表:

```

MOV   DPTR, #3000H
CLR   C
RLC   A

```

JMP @A+DPTR

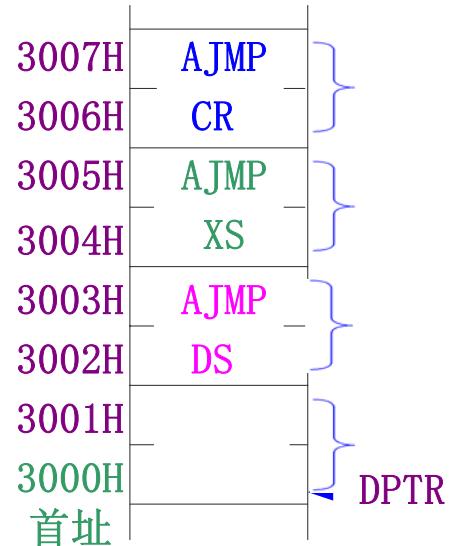
AJMP DS

键功能	键值	处理程序
读数据	01	DS
写数据	02	XS
插入	03	CR

AJMP XS

AJMP CR

...



3、利用堆栈操作法： (P88)

设分支号已存入 A。

MTJS: MOV DPTR, #TAB ; 取分支入口地址表首地址

CLR C ; 分支号 × 2

RLC A

MOV R2, A

MOVC A, @A+DPTR ; 取分支地址低位

PUSH A ; 入栈保存

MOV A, R2

INC A

MOVC A, @A+DPTR ; 取分支地址高位

PUSH A ; 入栈保存

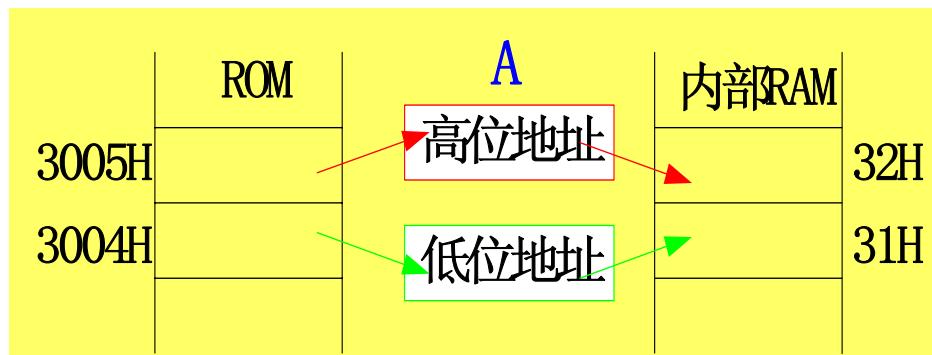
RET ; 分支地址 → PC, 转移

TAB: DW ADDR0 ; 分支程序入口地址表

DW ADDR1

...

ADDR0: ... ; 程序段 0 ...



## 4-5 循环程序

包含多次重复执行的程序段，循环结构使程序紧凑

### 4-5-1 循环程序的构成

各个环节任务：

一、初始化部分：

循环准备工作。如：清结果单元、设指针、设循环控制变量初值等。

二、循环体：

循环工作部分：

需多次重复处理的工作。

循环控制部分：

1.修改指针和循环控制变量。

2.检测循环条件：满足循环条件，继续循环，否则退出循环。

三、结束部分：

处理和保存循环结果。

允许 0 次循环的循环结构：在循环工作之前检测循环条件。

### 4-5-2 单重循环

简单循环结构：循环体中不套循环。

例：求 n 个单字节数据的累加，设数据串已在 43H 起始单元，数据串长度在 42H 单元，

累加和不超过 2 个字节。

```

SUM:  MOV  R0, #42H      ; 设指针
      MOV  A, @R0
      MOV  R2, A      ; 循环计数器←n
      CLR  A      ; 结果单元清 0
      MOV  R3, A
      ADD1: INC  R0      ; 修改指针
      ADD  A, @R0      ; 累加
      JNC  NEXT      ; 处理进位 (C=0 跳)
      INC  R3      ; 有进位, 高字节加 1
      NEXT: DJNZ R2, ADD1      ; 循环控制: 数据是否加完?
      MOV  40H, A      ; 循环结束, 保存结果
      MOV  41H, R3
      RET

```

片内 RAM	
...	...
	Xn
...	...
43H	X1
42H	n
41H	SUM <sub>H</sub>
40H	SUM <sub>L</sub>

循环控制方法: 计数控制、特征标志控制。

一、计数控制: (参看 P89[例 4.6])

设循环计数器, 控制循环次数。正计数和倒计数两种方式。

例: 为一串 7 位 ASCII 码数据的 D7 位加上奇校验, 设数据存放在片外 RAM 的 2101H 起始单元, 数据长度在 2100H 单元。

```

MOV  DPTR, #2100H
      MOVX A, @DPTR
      MOV  R2, A
      NEXT: INC  DPTR
      MOVX A, @DPTR

```

片外 RAM	
...	...
2102H	01101000
2101H	00101101
2100H	n

```

ORL A, #80H
JNB P, PASS      ; P=0: 1 的个数为偶数, 跳
MOVX @DPTR, A
PASS: DJNZ R2, NEXT
DONE: SJMP DONE

```

## 二、特征控制:

设定循环结束标志实现循环控制。(参看 P89[例 4.7])

例: 找正数表最小值。正数表存在片外 RAM 中以 LIST 为起始单元, 用-1 作为结束标志。

```

START: MOV DPTR, #LIST      ; 数表首地址
       MOV B, #127        ; 预置最小值
NEXT:  MOVX A, @DPTR      ; 取数
       INC DPTR        ; 修改指针
       CJNE A, #-1, NEXT1 ; 是否为数表结尾? (A≠-1 跳)
       SJMP DONE        ; 循环结束
NEXT1: CJNE A, B, NEXT2 ; 比较 (A≠B 跳)
NEXT2: JNC NEXT        ; C=0 跳
       MOV B, A        ; 保存较小值
       SJMP NEXT
DONE: SJMP DONE

```



习题: 统计一班考试为 100 分和不及格人数, 成绩单在 41H 起始单元。

### 4-5-3 多重循环

循环体中套循环结构。以双重循环使用较多。

例: 将内存一串单字节无符号数升序排序。 (参看 P102[例 4.14])

步骤：

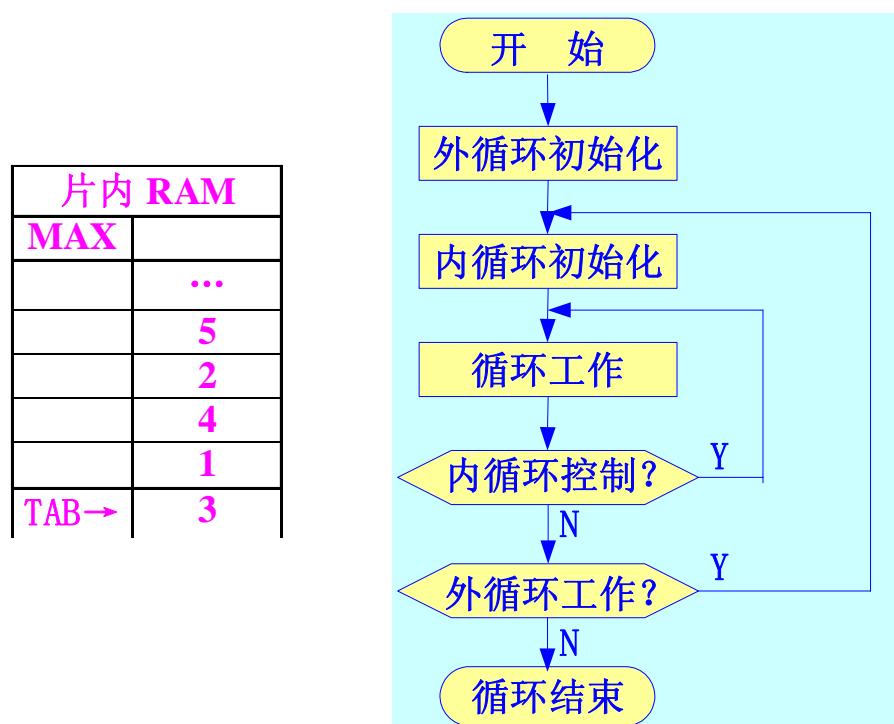
每次取相邻单元的两个数比较，决定是否需要交换数据位置。

第一次循环，比较N-1次，取到数据表中最大值。

第二次循环，比较N-2次，取到次大值。

...

第N-1次循环：比较一次，排序结束。



SORT: MOV A, #N-1 ; N个数据排序

MOV R4, A ; 外循环次数

LOOP1: MOV A, R4

MOV R3, A ; 内循环次数

MOV R0, #TAB ; 设数据指针

LOOP2: MOV A, @R0 ; 取二数

MOV B, A

```

INC      R0
MOV      A, @R0
CJNE    A, B, L1      ; 比较 (A≠B跳)
L1: JNC    UNEX      ; A≥B, 不交换
DEC      R0      ; 否则交换数据
XCH      A, @R0
INC      R0
MOV      @R0, A
UNEX: DJNZ  R3, LOOP2  ; 内循环结束?
DJNZ    R4, LOOP1  ; 外循环结束?
RET

```

软件延时程序:

用循环程序将指令重复多次执行, 实现软件延时。

1、单循环定时程序: (参看 P98)

```

MOV    R5, #TIME
LOOP: NOP
NOP
DJNE  R5, LOOP

```

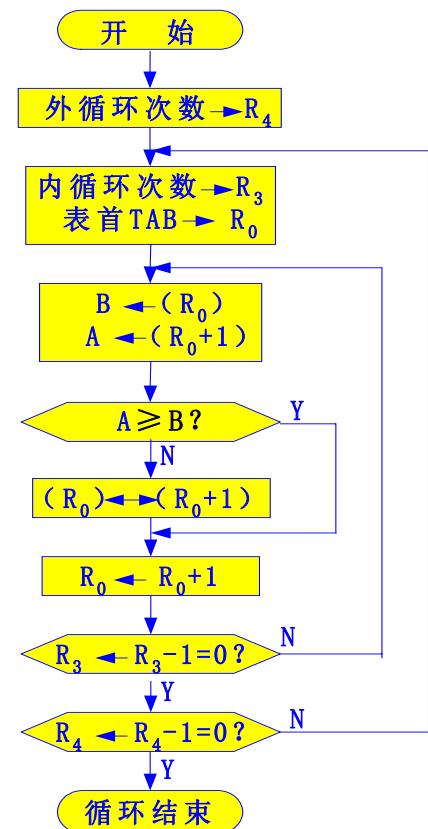
设:  $f_{osc}=6MHz$ , 则  $T=12/6MHz=2\mu s$

$$\begin{aligned}
t &= (1+4 \times TIME) \times T \\
&= 2+8 \times TIME \quad (\mu s)
\end{aligned}$$

2、多重循环定时:

用循环程序将指令重复多次执行, 实现较长时间的延时。

试计算延时程序的执行时间。(参看 P99)



源程序	指令周期(M)	指令执行次数
DELAY: MOV R6, #64H (=100)	1	1
I1: MOV R7, #0FFH (=255)	1	100
I2: DJNZ R7, I2	2	$100 \times 255$
DJNZ R6, I1	2	100
RET	2	1

延时时间计算: (设时钟  $f_{osc}=12MHz$ )  $M=1 \mu s$

$$t = (1 \times 1 + 1 \times 100 + 2 \times 100 \times 255 + 2 \times 100 + 2 \times 1) \times M = 51.303 ms$$

习题:

DELAY: MOV R6, #100  
 D1: MOV R7, #10  
 D2: NOP  
 DJNZ R7, D2  
 DJNZ R6, D1  
 RET

计算延时程序的执行时间 (设时钟  $f_{osc}=6MHz$ ,  $M=2 \mu s$ )。

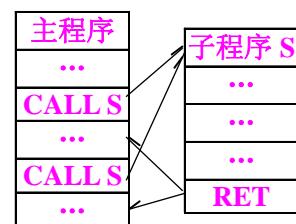
$$t=6.606 ms$$

## 4-6 子程序

子程序: 能完成某项特定功能的独立程序段, 可被反复调用。

### 4-6-1 子程序设计

- 一、子程序入口用标号作为子程序名。
- 二、调用子程序之前设置好堆栈。
- 三、用返回指令 RET 结束子程序, 并保



证堆栈栈顶为调用程序的返回地址。

四、子程序嵌套须考虑堆栈容量。

五、提供足够的调用信息：

如：子程序名、子程序功能、入口参数和出口参数、子程序占用的硬件资源、子程序中调用的其他子程序名。

#### 4-6-2 子程序的类型

按子程序与主程序之间传递参数的方式分类。

入口参数：调用子程序之前，需要传给子程序的参数。

出口参数：子程序送回调用程序的结果参数。

设计子程序应满足通用性的要求，不针对具体数据编程。

如：1. 子程序功能为求单字节数的立方：

$A \leftarrow A^3$ ，入口参数和出口参数为 A。

2. 子程序功能为求单字节数的 n 次方：

$(41H)(42H) \leftarrow (40H)^A$ ，入口参数为(40H)和 A，

出口参数为(42H)(41H)。

选用不同的参数传递方式：

1. 寄存器传送参数；

2. 存储器传送参数；

3. 堆栈传送参数。

例：将R4R5R6中三个字节数据对半分解，变成6个字节，存入显示缓冲区 (DISMEM0~DISMEM5)。

1) 子程序UFOR1的功能：将A累加器中单字节数据，对半分解成两个字节，存入R0所指向的相邻两个单元。

UFOR1: MOV @R0, #00H

```

XCHD A, @R0          ; 保存低半字节
INC  R0              ; 修改指针
MOV  @R0, #00H
SWAP A
XCHD A, @R0          ; 保存高半字节
RET

```

2) 调用子程序UF0R1之前, 将待分解的内容送A, 存放地址送R0。

RAM 显示缓冲区	
地址	数据
<b>DISMEM 5</b>	0 R6 <sub>H</sub>
<b>DISMEM 4</b>	0 R6 <sub>L</sub>
<b>DISMEM 3</b>	0 R5 <sub>H</sub>
<b>DISMEM 2</b>	0 R5 <sub>L</sub>
<b>DISMEM 1</b>	0 R4 <sub>H</sub>
<b>DISMEM 0</b>	0 R4 <sub>L</sub>

R<sub>0</sub>→

RAM	
0	A <sub>H</sub>
0	A <sub>L</sub>

例: 比较两个数据串是否完全相等, 若完全相等, A=0;

否则, A=FFH。

```

PMT: MOV  R2, A          ; 设数串长度
CHC: MOV  A, @R0          ; 各取数串中的一个数
      MOV  42H, @R1
      CJNE A, 42H, NOM    ; 是否相等? 不相等转移
      INC  R0              ; 相等, 修改指针
      INC  R1
      DJNZ R2, CHC        ; 全部比较完?
      MOV  A, #00H          ; 完全相等
      SJMP PEND

```

NOM: MOV A, #0FFH ; 不完全相等

PEND: RET

例：查表求出数据的ASCII码，再以字符形式输出。

- 1) 子程序HEXASC功能：取出堆栈中数据，查表将低半字节转换成ASCII码送累加器A。
- 2) 分别将待转换数据入栈，然后调用子程序HEXASC。（参看P96）

```
MOV SP, #30H
PUSH 40H           ; 入口参数入栈
LCALL HEXASC
POP A
...
HEXASC: DEC SP    ; 跳过返回地址
DEC SP
POP A             ; 取入口参数
...               ; 查表求ASCII码
PUSH A            ; 保存出口参数
INC SP            ; 指向返回地址
INC SP
RET
DB '0', '1', ... ; ASCII码表
```

RAM	
...	
41H	23
40H	01

...	
33H	
32H	
31H	
30H	X

## 4-7 算术运算程序

### 4-7-1 多字节加减运算程序

1. 多字节加法子程序,  $Z=X+Y$ 。(参看 P90)

```
ADDS: CLR    C
      MOV    R2, #23H
      LOOP: MOV    A, @R0
             ADDC  A, @R1      ; 加一字节
             MOV    @R0, A      ; 存和一字节
             INC    R0          ; 修改指针
             INC    R1
             DJNZ  R2, LOOP    ; 全部字节加完?
             RET
```

	<b>Y<sub>H</sub></b>
	...
<b>R1→</b>	<b>Y<sub>L</sub></b>
	<b>X<sub>H</sub></b>
	...
<b>R0→</b>	<b>X<sub>L</sub></b>

习题 1: 编写十进制多字节加法子程序,  $Z=X+Y$ 。

习题 2: 编写多字节减法子程序,  $Z=X-Y$ 。

思考题: 修改程序使运算结果保存到其他存储单元。

### 4-7-2 多字节无符号数乘除运算

相加计算多字节乘法程序。步骤:

1.部分积清零。

2.检测乘数各位,

为1则部分积对位加被乘数, 否则加0。

3.对位方法: 被乘数左移或部分积右移。

(参看P92)

例: 双字节相乘  $R2R3 \times R6R7 @ R4R5R6R7$

解: 部分积清零, 右移检测乘数, 决定部分积是否加被乘数, 部分积右移对位。

相减计算多字节除法程序。步骤：

- 1.对齐高位被除数试减除数。
- 2.若够减商上 1，不够减商上 0 且恢复余数。
- 3.余数左移或除数右移对位。
- 4.循环重复前 3 步，直至取够相应位数的商。

“四舍五入”：得到余数后，判断余数乘 2 后是否大于除数，若大于除数则商再加“1”；

否则不加。

(参看 P94)

例：  $R2R3R4R5 \div R6R7@R4R5$  (余数@R2R3)

解：1.判断  $R2R3 < R6R7$ ？使商不大于 16 位。

- 2.被除数左移 1 位，试减除数。
- 3.若够减，商加 1 并保留余数。

#### 4-7-3 代码转换程序

(一) 十六进制数转换为ASCII码；

(二) ASCII码转换为十六进制数。

0~9的ASCII码：30~39H；A~F的ASCII码：41~46H。

十六进制数转换为ASCII码：

HASC: CJNE A, #0AH, N

N: JNC N1 (C=0跳N1)

ADD A, #30H

SJMP SE

N1: ADD A, #37H

SE: RET

ASCII码转换为十六进制数：

```
AHEX: CLR    C
      SUBB   A, #30H
      CJNE   A, #0AH, N
N:     JC     N1
      SJMP   AE
N1:    SUBB   A, #11H
      CJNE   A, #06H, N2
N2:    JNC    ERR
      ADD    A, #0AH
      SJMP   AE
ERR:   MOV    A, #0FFH
AE:    RET
```

P96: 例题 (数制转换)

[例4.11]在内部RAM的hex单元中存有2位十六进制数，试将其转换为ASCII码，并存放于asc和asc+1两个单元中。设 (hex) =7BH

```
MOV    SP, #3FH
MAIN: PUSH   hex
      ACALL  HASC
      POP    asc
      MOV    A, hex
      SWAP   A
      PUSH   ACC
      ACALL  HASC
```

POP asc+1

子程序 (HASC) :

```
HASC: DEC SP
      DEC SP
      POP ACC
      ANL A, #0FH
      ADD A, #7  (A=12H)
      MOVC A, @A+PC
      PUSH ACC
      INC SP
      INC SP
      RET
```

ASCTAB:DB "0, 1, 2, 3, 4, 5, 6, 7"

DB "8, 9, A, B, C, D, E, F"

### (三) BCD码与二进制数之间的转换

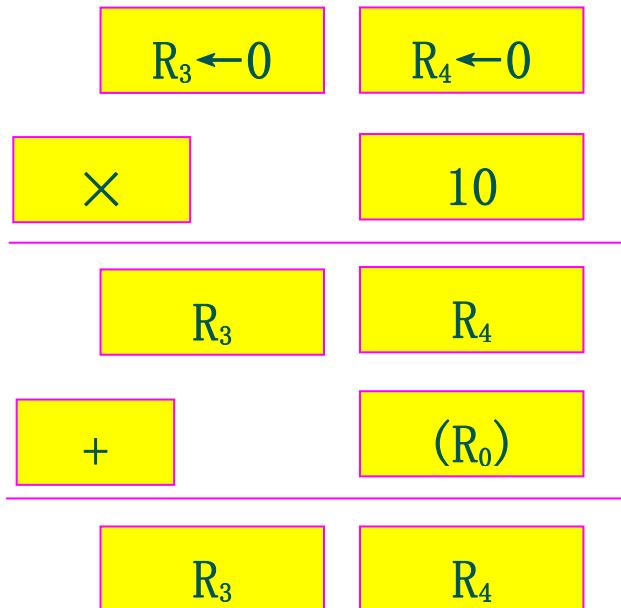
有乘法和除法两种转换方式。

#### 1. BCD 码转换为二进制数:

$$\begin{aligned} D &= d_{n-1} \times 10^{n-1} + d_{n-2} \times 10^{n-2} + \dots + d_1 \times 10^1 + d_0 \times 10^0 \\ &= (((d_{n-1} \times 10 + d_{n-2}) \times 10 + d_{n-3}) \times 10 + \dots d_1) \times 10 + d_0 \end{aligned}$$

“整数十翻二”：从最高位开始，按二进制运算法则循环。

“乘十加次低位”：  $B = B \times 10 + b_i$  。



## 2. 二进制数转换为 BCD 码:

$$B = b_{n-1} \times 2^{n-1} + b_{n-2} \times 2^{n-2} + \dots + b_1 \times 2^1 + b_0 \times 2^0$$

$$= ((b_{n-1} \times 2 + b_{n-2}) \times 2 + b_{n-3}) \times 2 + \dots + b_1 + b_0$$

“整数二翻十”：从最高位开始，按十进制运算法则循环。

“乘二加次低位”：  $D = D \times 2 + d_i$  。

除法转换式：

$$b_{n-1} \times 2^{n-1} + b_{n-2} \times 2^{n-2} + \dots + b_0 = d_{m-1} \times 10^{m-1} + d_{m-2} \times 10^{m-2} + \dots + d_0$$

两边同时除基数，两边的整数或小数应该分别相等。

除基数，取出 1 位余数，得到的商继续除基数取余数。

循环“除基取余”操作，得到转换进制的各位系数。

编程习题要求：

1. 为程序标明必要注释。

2. 上机调试通过。

3.给出程序执行前设定的数据，并记录程序执行后的结果数据。

4.要求写明数据的存储单元。

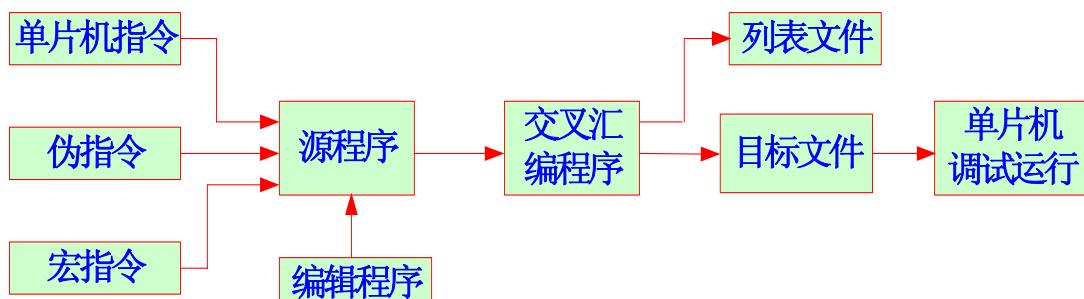
## 小结：

1、有三类 

- 机器语言 (Machine Language)
- 汇编语言 (Assemble)
- 高级语言 (High Level Language)

不同计算机语言的应用：

源程序通过编译得到机器能执行的目标程序。



汇编语言程序可以高效率利用计算机资源，目标程序占用内存少，执行速度快，适合于自动测控系统反应快速、结构紧凑的要求。实际应用中，常与 C 语言配合使用。

高级语言程序容易掌握，通用性好，但编译程序系统开销大，目标程序占用内存多，且执行时间比较长，多用于科学计算、工业设计、企业管理。

语言处理方式 

- 编译执行方式：将源程序编译成机器语言的目标，然后执行，生成并保留目标程序。
- 解释执行方式：逐条语句边解释边执行，即每解释一条语句就执行一条语句。

应用   
 高级语言：在科学计算方面（要求机器内存大，因编译后的目标程序比汇编语言的源程序大）。  
 汇编语言：在实时控制中。

## 2、宏指令与子程序的区别：

宏指令：在汇编语言的源程序中，若有一段程序要多次使用，为了使在源程序中不重复书写这样的一段程序，可用一条宏指令来代替，由汇编程序在汇编时产生所需要的代码。

优点：可简化源程序。

与子程序不同：宏指令并没有简化目标程序，而子程序可以。

## 3、伪指令：ORG、DB、DW、EQU、END 等。

## 4、程序设计步骤。

## 5、常用程序结构：顺序结构、分支结构、循环结构

### （1）顺序程序：直线程序或简单程序

（2）分支程序   
 单重分支  
 多重分支   
 多次使用条件转移指令  
 按分支号转移   
 地址表法  
 转移指令表法  
 过堆栈操作

### （3）循环程序：

构成：初始化、循环体、控制变量的修改、循环次数的控制。

延时；

控制   
 记数控制  
 特征标志控制

## 第5章 单片机存储器扩展

### 一、教学要求：

掌握：单片机系统扩展技术及应用系统设计方法。学会程序存储器和数据存储器的扩展方法。注意片内 RAM 和系统地址空间的使用分配以及一些常用扩展芯片的接口方法和访问控制方法。

### 二、教学内容：

- 5. 1 单片机系统扩展及结构
- 5. 2 单片机存储器扩展与编址技术
- 5. 3 单片机程序存储器扩展
- 5. 4 单片机数据存储器扩展
- 5. 5 存储器综合扩展
- 5. 6 单片机存储器系统的特点和使用

### 三、教学重点：

单片机系统扩展技术及应用系统设计方法，程序存储器和数据存储器的扩展方法。

### 四、教学难点：

片内 RAM 和系统地址空间的使用分配以及一些常用扩展芯片的接口方法和访问控制方法。

### 五、建议学时：4 学时。

### 六、教学内容：

#### 5-1 系统扩展及结构

单片机芯片内具有 CPU、ROM、RAM、定时器/计数器及 I/O 口。但在实际应用中、大多数情况下仅靠片内资源是不够的。

——资源性扩展：包括存储器扩展和 I/O 扩展。

如何扩展？

扩展功能如何实现?

扩展部件如何连接?

整个扩展系统以单片机为核心, 通过总线把各扩展部件连接起来, 各扩展部件“挂”在总线上。

所谓总线, 就是连接系统中各扩展部件的一组公共信号线。

包括: 地址总线 (AB);

数据总线 (DB);

控制总线 (CB)。

存储器的连接

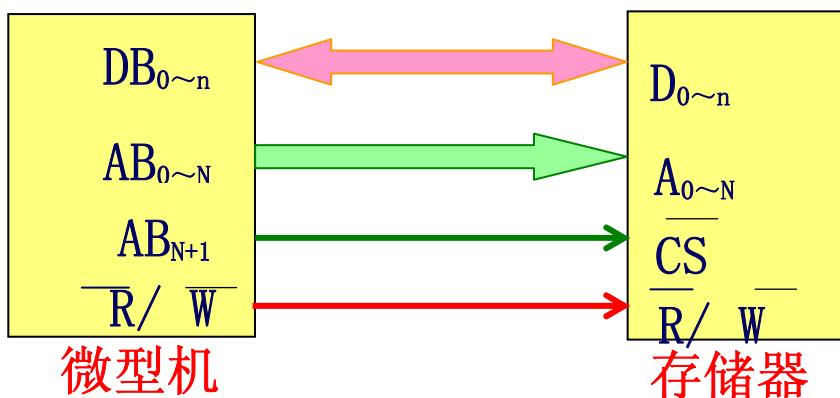
存储器与微型机三总线的连接:

1、数据线  $D_0 \sim n$  连接数据总线  $DB_0 \sim n$

2、地址线  $A_0 \sim n$  连接地址总线低位  $AB_0 \sim n$ 。

3、片选线 CS 连接地址总线高位  $AB_{n+1}$ 。

4、读写线 OE、WE(R/W)连接读写控制线 RD、WR。

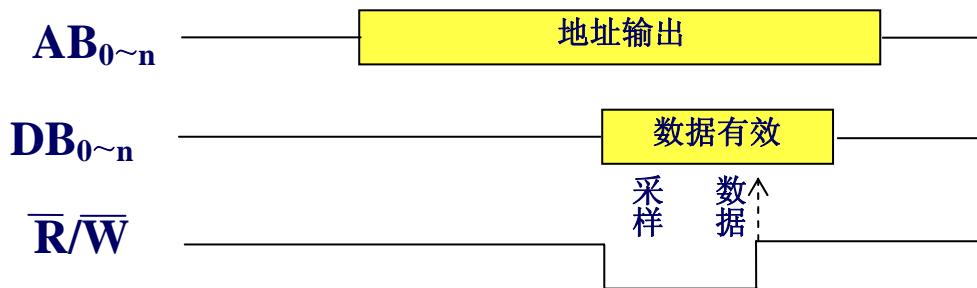
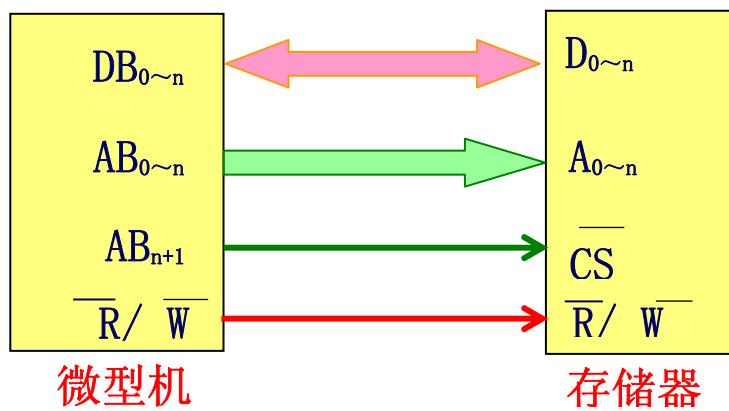


存储器与单片机的连接

存储器与微型机三总线的一般连接方法和存储器读写时序:

1. 数据总线与地址总线

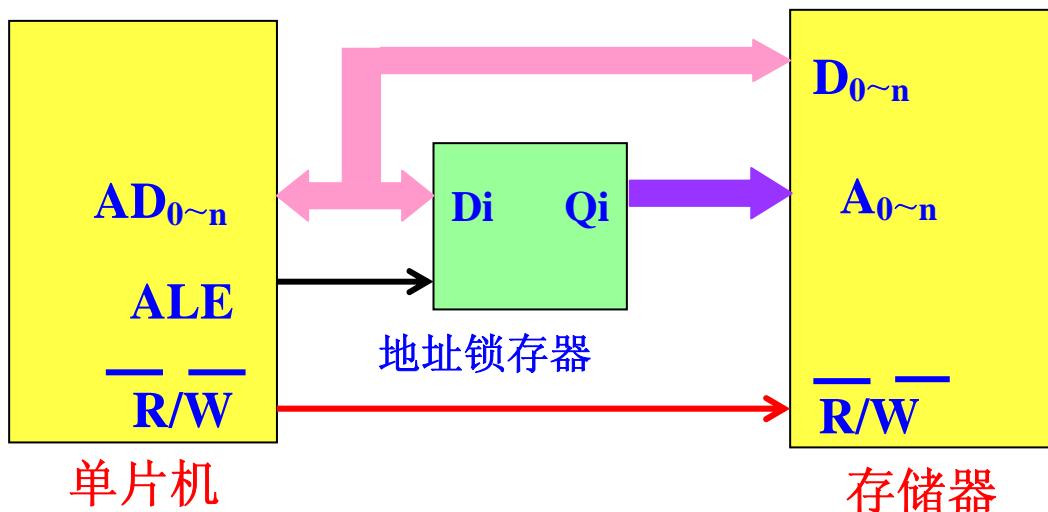
为两组独立总线。

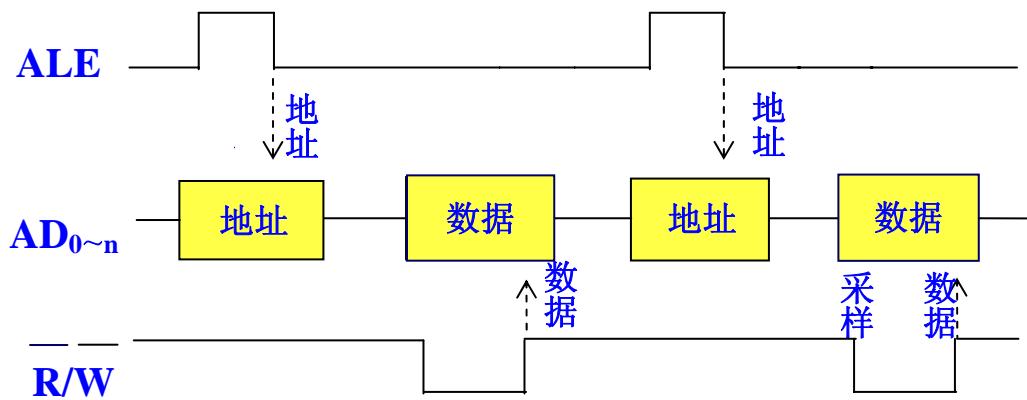


存储器与单片机的连接

## 2.微型机复用总线结构:

数据与地址分时共用一组总线。





### 5-1-1 单片机扩展的实现

- 单片机扩展的首要问题就是构造系统总线，然后再往系统总线上“挂”存储芯片或I/O接口芯片。
- “构造”总线——芯片本身并没有提供地址线和数据线。

具体的构造方法说明如下：

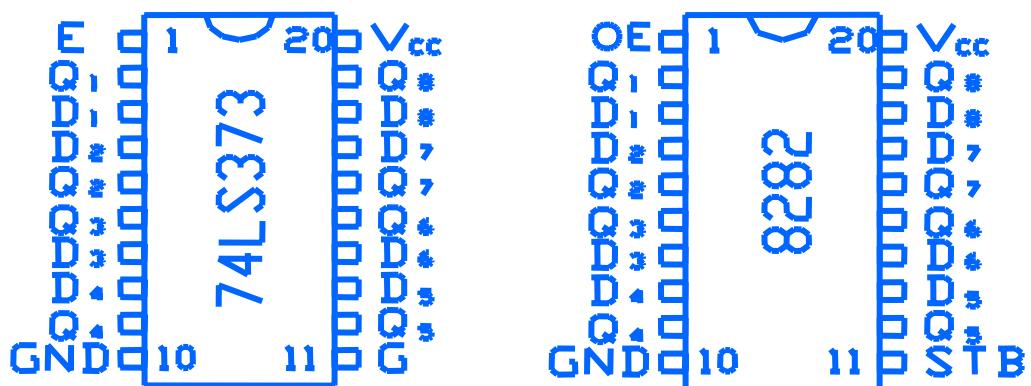
- 以P0口的8位口线作地址/数据线。
- 复用技术——地址和数据进行分离。

为此在构造地址总线时要添加一个8位锁存器。先把这低8位地址送锁存器暂存，然后就由地址锁存器给系统提供低8位地址，而把P0口线作为数据线使用。

- 以P2口的口线作高位地址线。
- 由P2口提供高8位，再加上P0口提供的低8位——64KB。

但实际应用系统中，地址高位并不固定为8位，而根据需要从P2口中引出。

8位地址锁存器：74LS373、8282等。



74LS373、8282 功能		
锁存	输出允许	输出
<b>G(STB)</b>	<b>OE</b>	<b>Qi</b>
1	0	Di
0	0	不变
X	1	Z

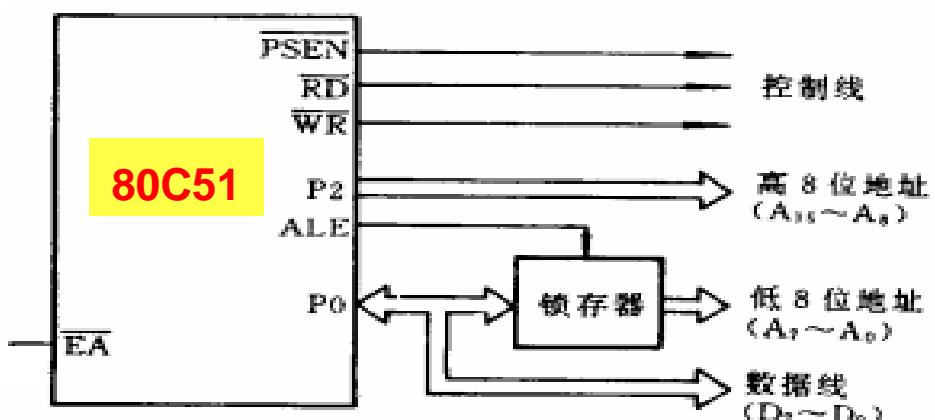


图5.2 单片机扩展总线构造图

控制信号：构成扩展系统的控制总线。

1. ALE 作地址锁存的选通信号，以实现低 8 位地址的锁存。
2. PSEN 作扩展程序存储器的读选通信号。
3. EA 作内外程序存储器的选通信号。

4. RD 和 WR 作扩展数据存储器和 I/O 端口的读写选通信号。

MCS-51 用于扩展存储器的外部总线信号：

P0.0~0.7: 8 位数据和低 8 位地址信号, 复用总线 AD0~7。

P2.0~2.7: 高 8 位地址信号 AB8~15。

ALE: 地址锁存允许控制信号。

$\overline{PSEN}$ : 片外程序存储器读选通信号。

$\overline{EA}$ : 内外程序存储器选择。

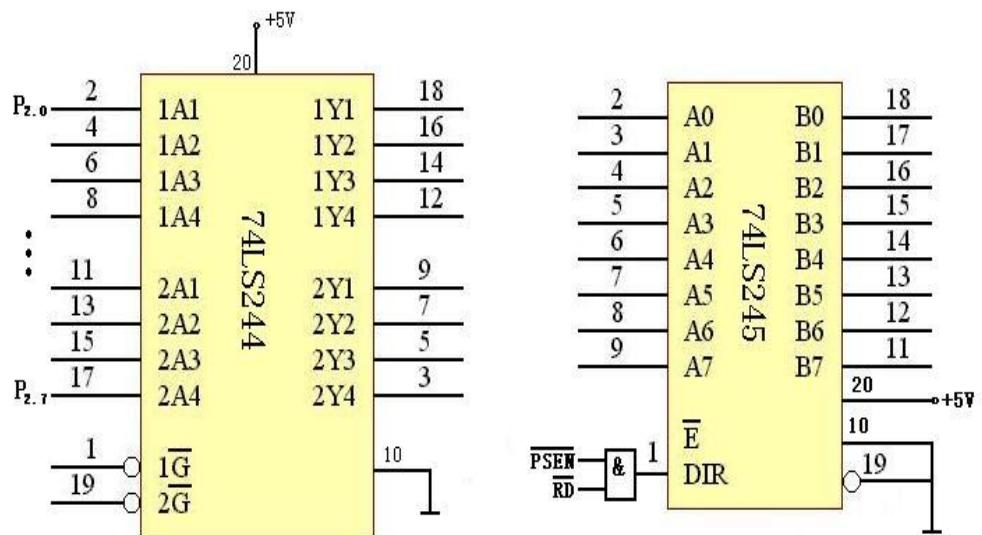
$\overline{RD}$ : 片外数据存储器读控制信号。

$\overline{WR}$ : 片外数据存储器写控制信号。

### 5-1-2 总线扩展驱动

当单片机外接芯片较多, 超出总线负载能力, 必须加总线驱动器。

- 单向驱动器 74LS244 用于地址总线驱动;
- 双向驱动器 74LS255 用于数据总线驱动。

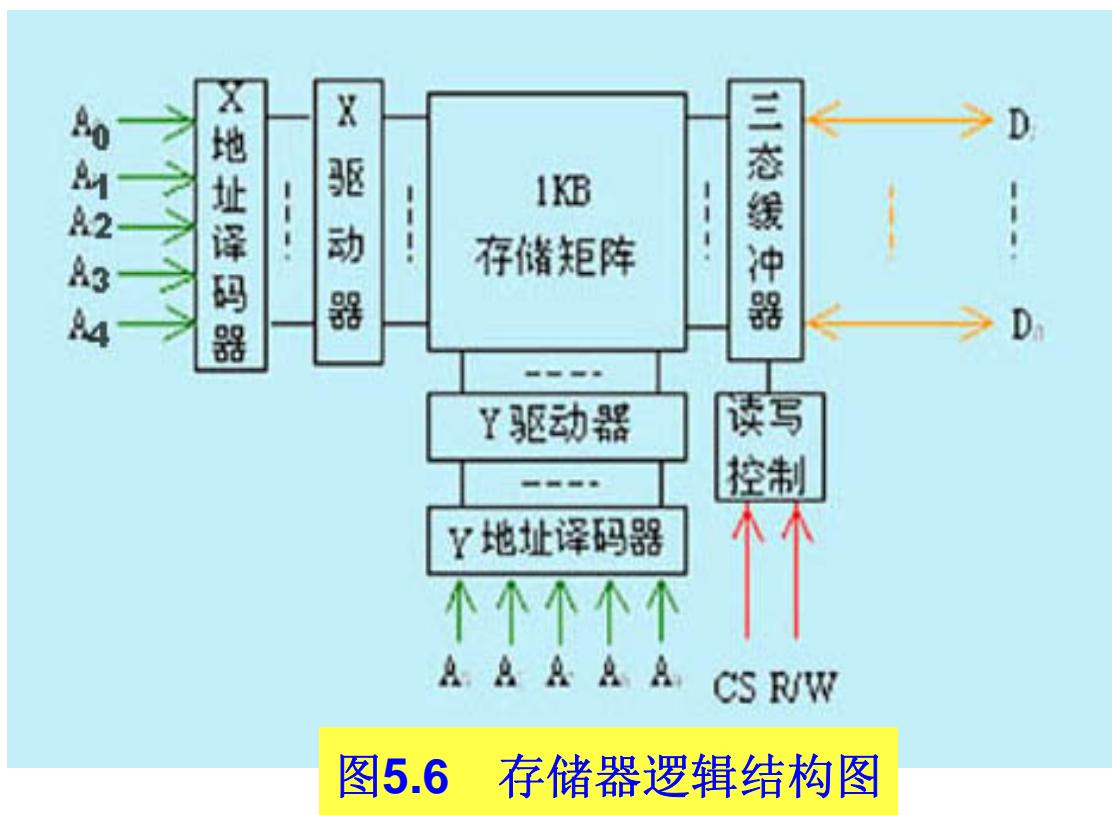


MSC-51 P2口的单向总线扩展

MSC-51 P0口的单向总线扩展

## 5-2 存储器扩展及编址技术

存储器结构框图



存储器内部为双向地址译码，以节省内部引线和驱动器。

如：1K容量存储器，有10根地址线。

单向译码需要1024根译码输出线和驱动器。

双向译码X、Y方向各为32根译码输出线和驱动器，总共需要64根译码线和64个驱动器。

存储器外部信号引线：

D0~7数据线：传送存储单元内容。根数与单元数据位数相同。

A0~9地址线：选择芯片内部一个存储单元。根数由存储器容量决定。

CS片选线：选择存储器芯片。当CS信号无效，其它信号线不起作用。

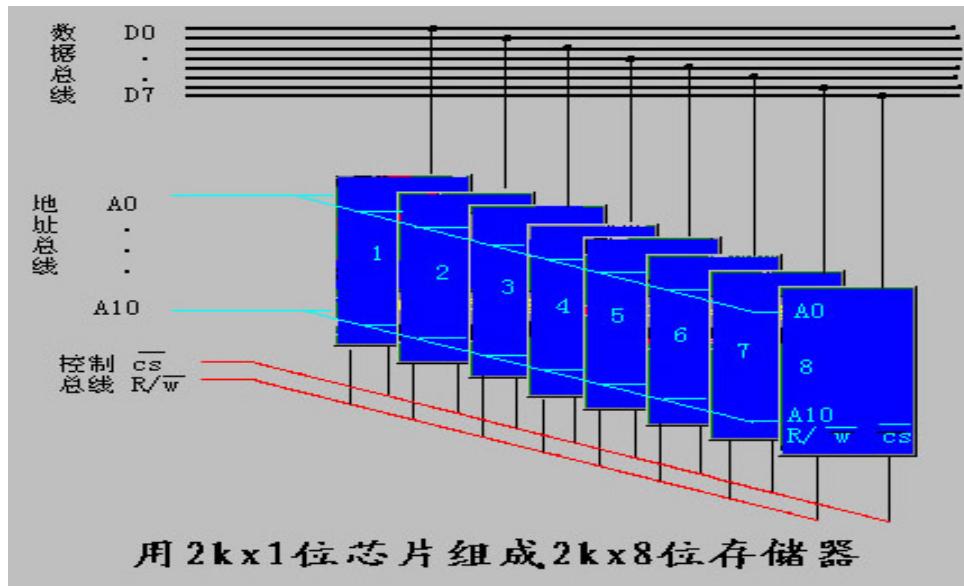
R / W(OE/WE)读写允许线：打开数据通道，决定数据的传送方向和传送时刻。

### 5-2-1 存储器芯片的扩展

- 用多片存储器芯片组成微型计算机系统所要求的存储器系统。
- 要求扩充后的存储器系统引出线符合微型计算机的总线结构要求。

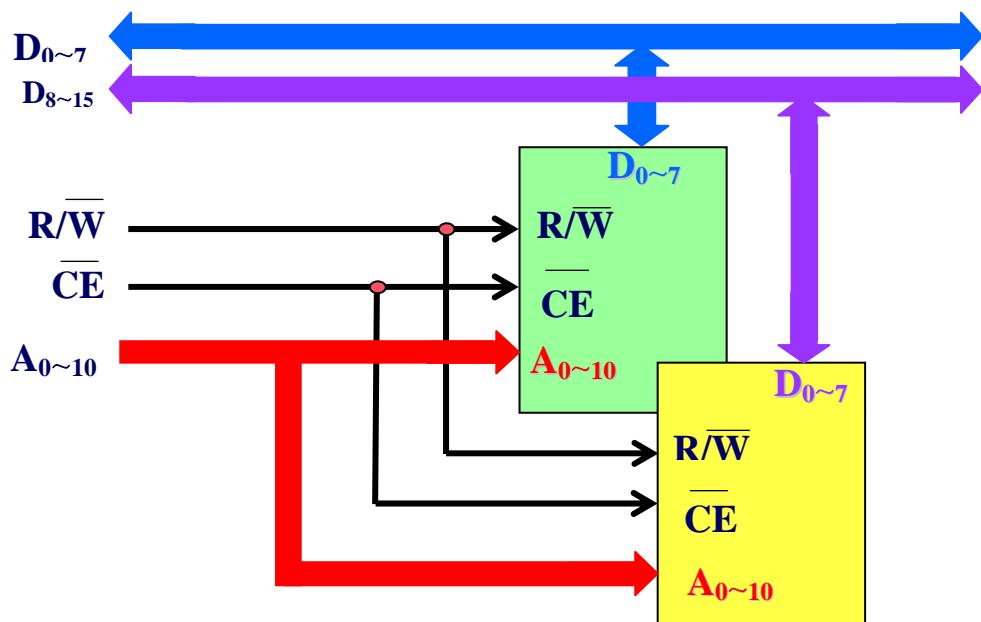
## 一、扩充存储器位数：

例 1：用  $2K \times 1$  位存储芯片组成  $2K \times 8$  位存储系统。



当地址、片选和读写信号有效，可并行存取 8 位信息。

例 2：用  $2K \times 8$  位存储器芯片组成  $2K \times 16$  位存储器系统。



地址、片选和读写引线并联后引出，数据线并列引出。

## 二、扩充存储器容量：

- 地址线、数据线和读写控制线均并联。

- 为保证并联数据线上没有信号冲突，必须用片选信号区别不同芯片的地址空间。

片选方法：

1. 线选法：

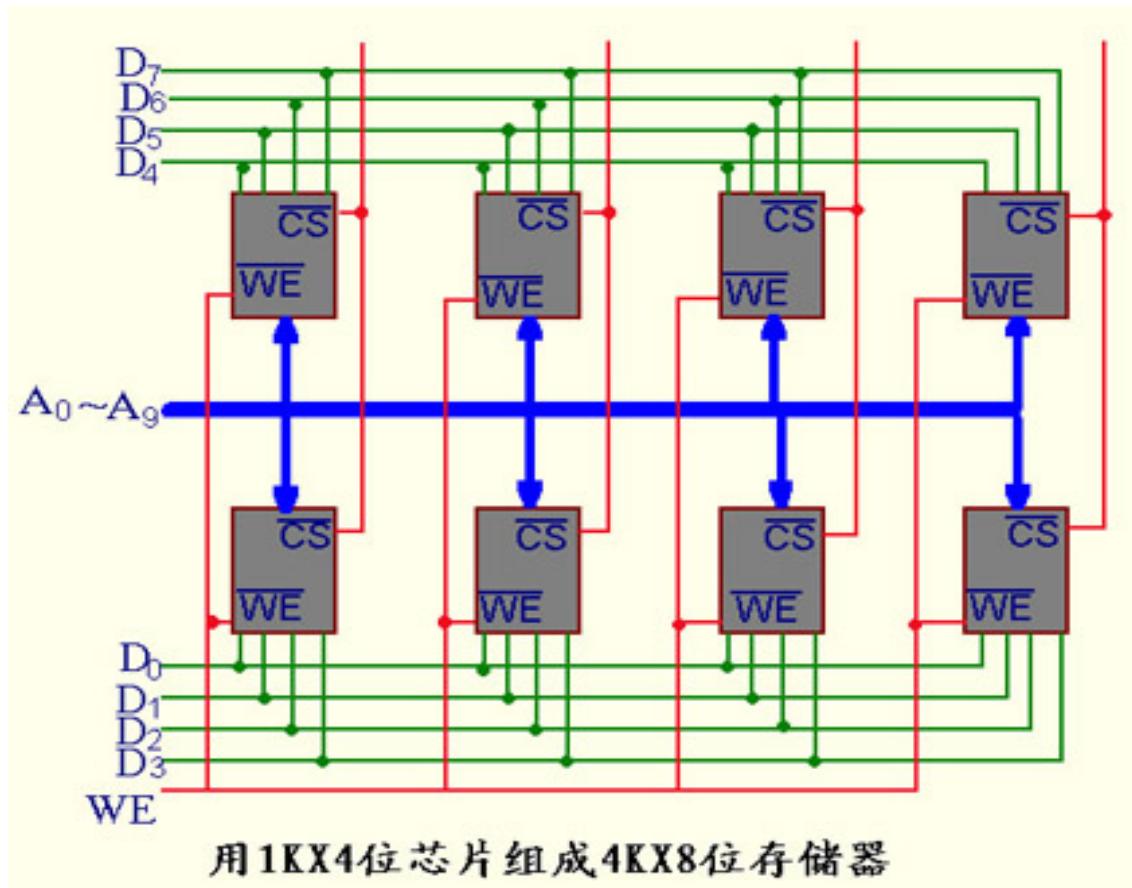
微型机剩余高位地址总线直接连接各存储器片选线。

2. 译码片选法：

微型机剩余高位地址总线通过地址译码器输出片选信号。

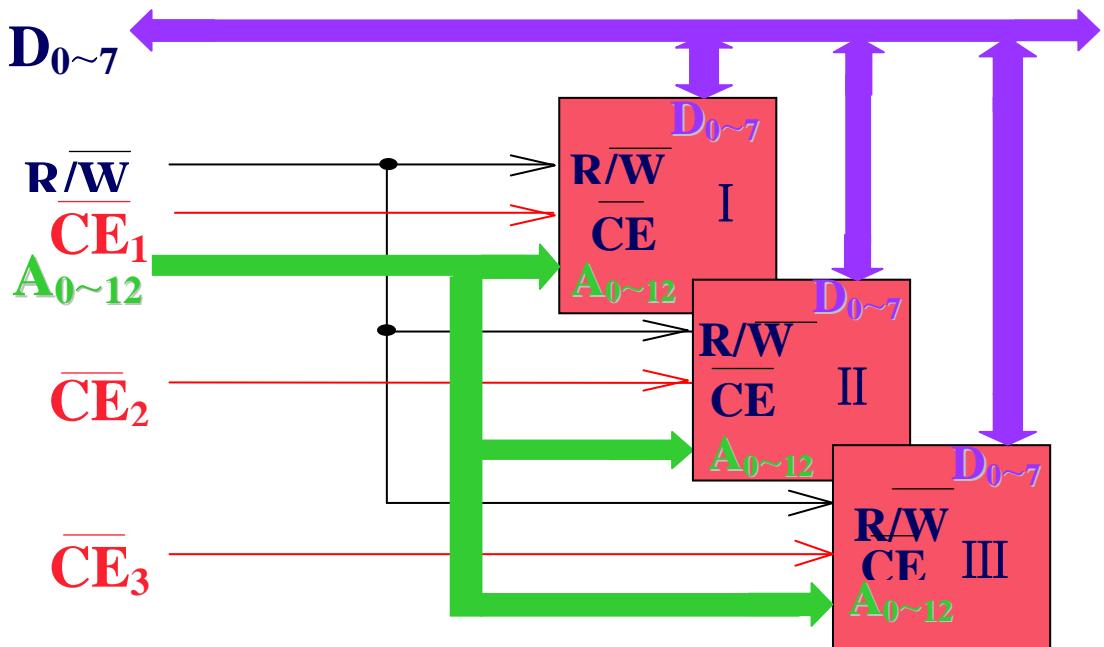
多片存储器芯片组成大容量存储器连接常用片选方法。

例 1：用  $1K \times 4$  位存储器芯片组成  $4K \times 8$  位存储器系统。



例 2：三片 8KB 的存储器芯片组成 24KB 容量的存储器。

设 CE1、CE2、CE3 分别连接微型机的高位地址总线 AB13、AB14、AB15。



确定各存储器芯片的地址空间：

ABi: 15141312 11109 8 7 6 5 4 3 2 1 0~15141312 11109 8 7 6 5 4 3 2 1 0

I : 1100 0000 0000 0000~1101 1111 1111 1111=C000H~DFFFH

II : 1010 0000 0000 0000~1011 1111 1111 1111=A000H~BFFFH

III: 0110 0000 0000 0000~0111 1111 1111 1111=6000H~7FFFH

### 5-2-2 存储器扩展的编址技术

- 所谓存储器编址，就是使用系统提供的地址线，通过适当的连接，最终达到一个编址唯一地对应存储器中一个存储单元的目的。
- 存储器编址分两个层次：（见 P119）
  - 存储芯片的选择；
  - 芯片内部存储单元的选择。
- 存储器映像则研究各部分存储器在整个存储空间中所占据的地址范围，以便为存储器的使用提供依据。

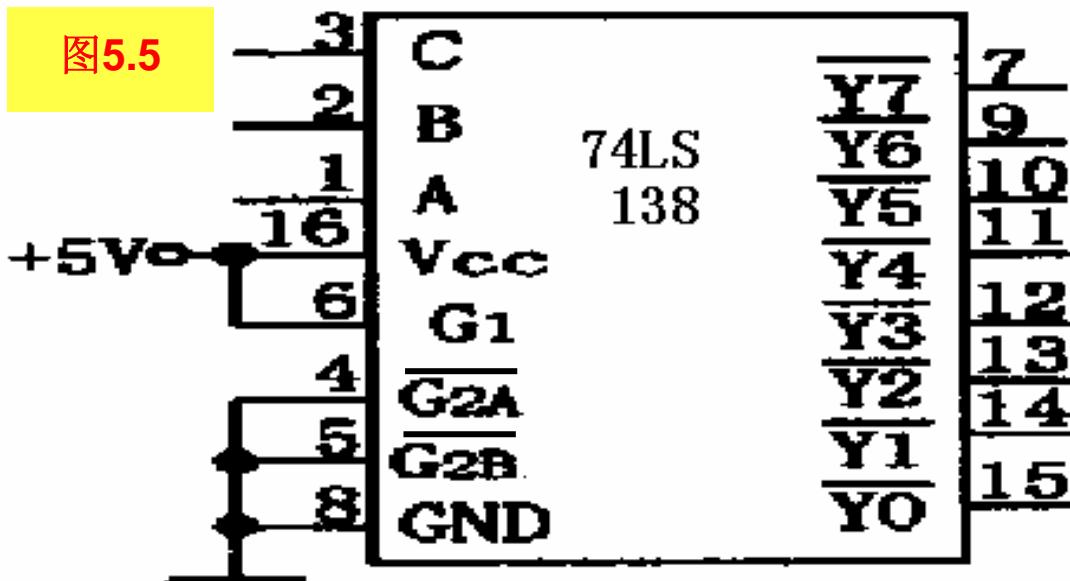
线选法：

- ❖ 直接以系统的地址位作为存储芯片的片选信号；

- ◆ 优缺点：简单明了，且不需增加电路。但存储空间的使用是断续的，不能有效地利用空间，扩充容量受限，只适用于小规模系统的存储器扩展。

译码法：

- ◆ 对系统的高位地址进行译码，以其译码输出作为片选信号。
- ◆ 有效地利用存储空间，适用于大容量多芯片扩展。
- ◆ 常用的译码芯片有：74LS139(双 2-4 译码器)、74LS138(3-8 译码器)和 74LS154 (4-16 译码器)等。



- G1 /G2A /G2B (使能端): 当 G1=“1”， G2A=G2B=“0”时， 3/8 译码器进入译码状态，这时 Y0~Y7 只有一位是低电平，其余全为高电平。译码无效时， Y0~Y7 全为高电平，无效。
- C 、 B、 A: 译码器输入 (C 为高位)。
- Y0~Y7 : 译码器输出，低电平有效。

3-8 地址译码器：74LS138

表5-1 74LS138 功能

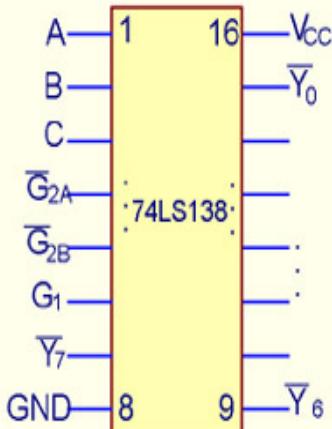
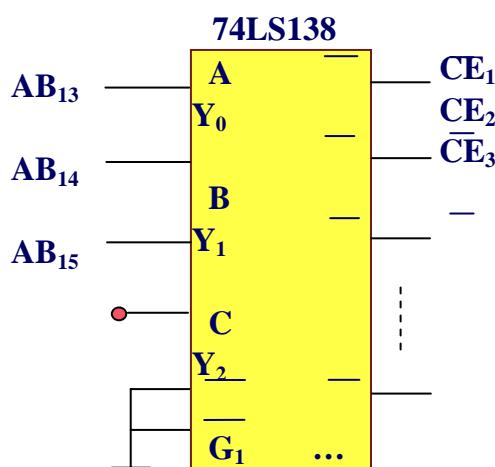


图5.5 74LS138引脚

$G_1 G_{2A} G_{2B}$	C B A	输出
1 0 0	0 0 0	$Y_0=0$ , 其余的为 1
1 0 0	0 0 1	$Y_1=0$ , 其余的为 1
1 0 0	0 1 0	$Y_2=0$ , 其余的为 1
1 0 0	0 1 1	$Y_3=0$ , 其余的为 1
1 0 0	1 0 0	$Y_4=0$ , 其余的为 1
1 0 0	1 0 1	$Y_5=0$ , 其余的为 1
1 0 0	1 1 0	$Y_6=0$ , 其余的为 1
1 0 0	1 1 1	$Y_7=0$ , 其余的为 1

3-8 地址译码器：74LS138

$Y_0$ 、 $Y_1$ 、 $Y_2$  分别连接三片存储器的片选端  $CE_1$ 、 $CE_2$ 、 $CE_3$



各片存储器芯片分配地址：

I : 0000H~1FFFH;

II : 2000H~3FFFH;

III: 4000H~5FFFH。

### 5-3 程序存储器扩展

工作时, ROM 中的信息只能读出, 要用特殊方式写入(固化信息), 失电后可保持信息不丢失。

#### 1.掩膜 ROM: 不可改写 ROM

由生产芯片的厂家固化信息。在最后一道工序用掩膜工艺写入信息, 用户只可读。

#### 2.PROM: 可编程 ROM

用户可进行一次编程。存储单元电路由熔丝相连, 当加入写脉冲, 某些存储单元熔丝熔断, 信息永久写入, 不可再次改写。

#### 3.EPROM: 可擦除 PROM

用户可以多次编程。编程加写脉冲后, 某些存储单元的 PN 结表面形成浮动栅, 阻挡通路, 实现信息写入。用紫外线照射可驱散浮动栅, 原有信息全部擦除, 便可再次改写。

#### 4.EEPRROM: 可电擦除 PROM

既可全片擦除也可字节擦除, 可在线擦除信息, 又能失电保存信息, 具备 RAM、ROM 的优点。但写入时间较长。

#### 扩展程序存储器电路

常用 EPROM 芯片:

Intel 2716 (2K×8 位)

2732 (4KB)

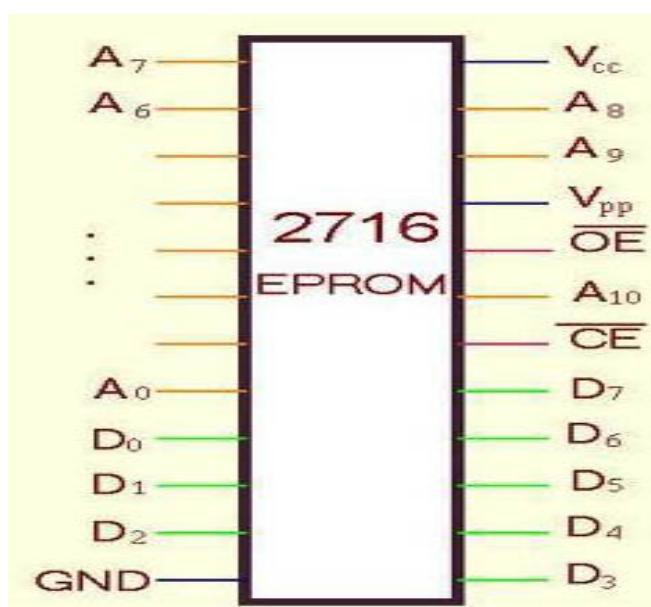
2764 (8KB)

27128(16KB)

27256(32KB)

27512(64KB)

- CE/PGM——片选低电



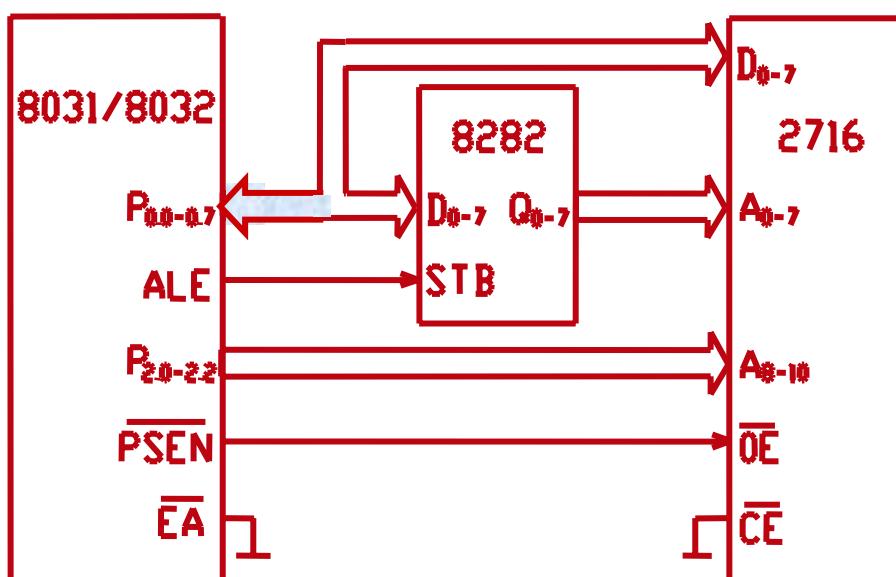
平有效。当编程时引入编程脉冲。

- OE——（输出允许）有效时输出缓冲器打开，被寻址单元才能被读出。
- VPP——编程时加十 25V 编程电压电源。

表 5-3 2716 工作方式

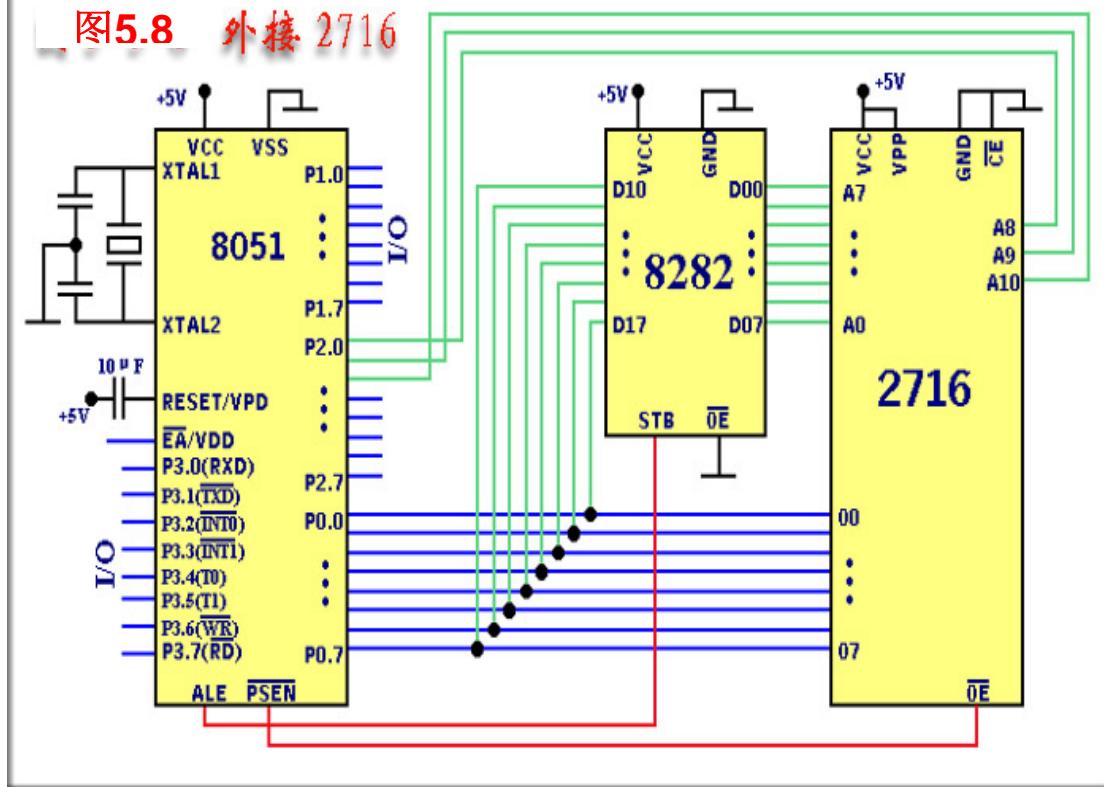
工作方式		V <sub>pp</sub>	CS	OE	D <sub>0~7</sub>
工 作	读	+5v	0	0	D <sub>0~7</sub>
	待机	+5v	0	1	Z
	禁止	+5v	1	0	Z
编 程	写入	+25v	1	1	D <sub>IN</sub>
	校验	+25v	0	0	D <sub>0~7</sub>
	禁止	+25v	0	1	Z

8031/8032 扩展 2KB EPROM Intel 2716（总线形式）

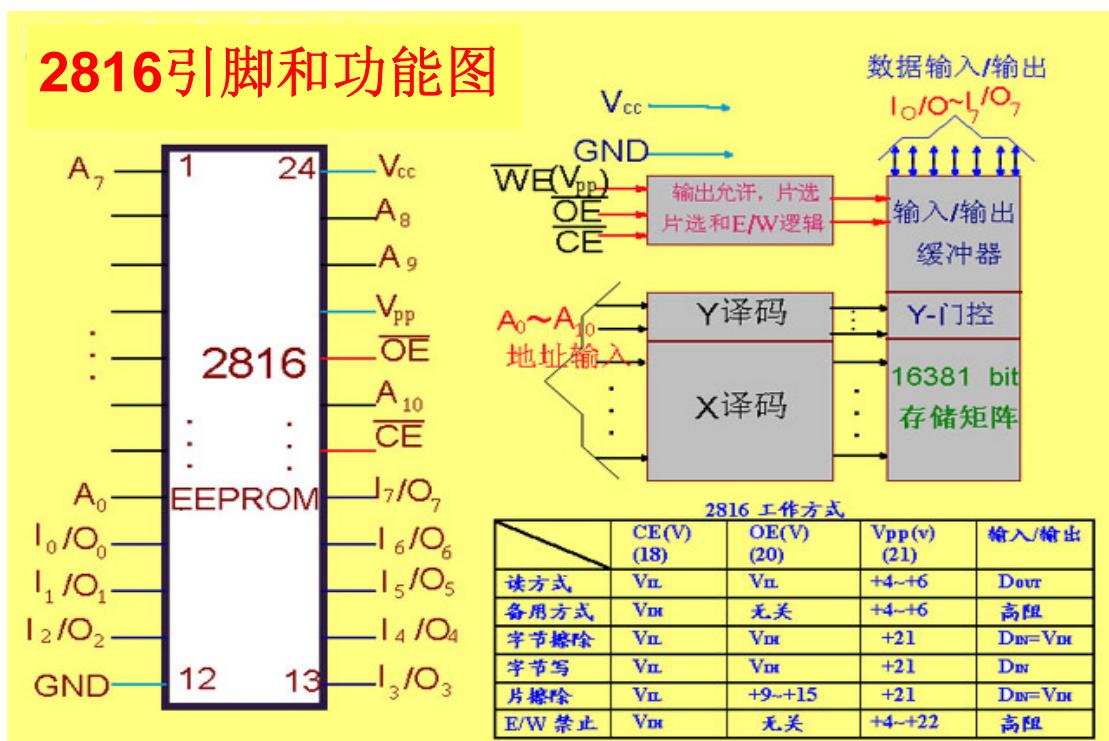


- 最低地址：8000H；
- 最高地址：87FFH。
- 地址范围：8000H~87FFH。

图5.8 外接2716



EEPROM 2816、2817

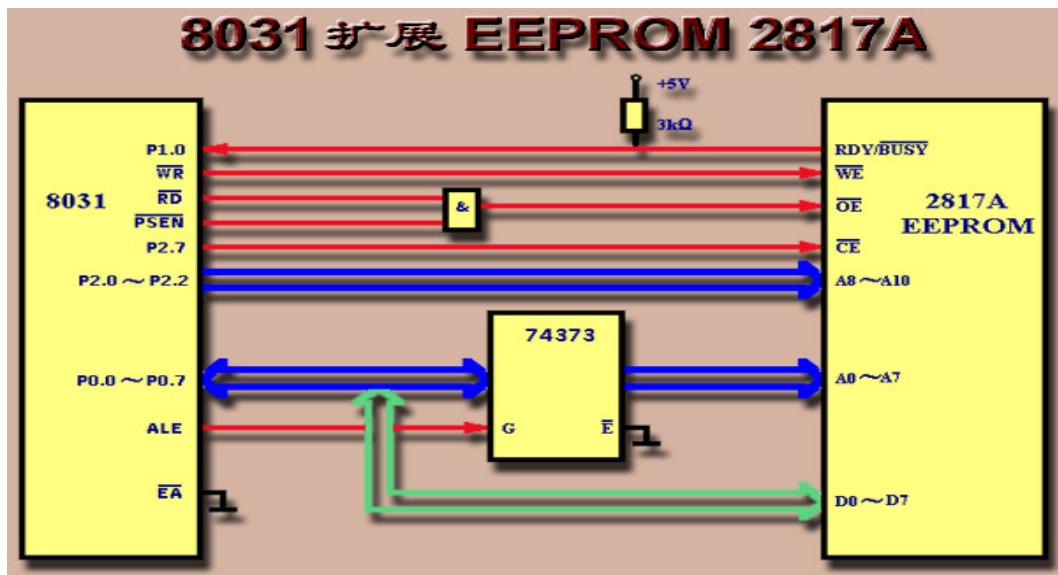


单片机外接 EEPROM 电路的存储器电路

EEPROM 既能作为程序存储器又能作数据存储器。

将程序存储器与数据存储器的空间合二为一。

片外存储器读信号= PSEN · RD



#### 5-4 数据存储器扩展

扩展数据存储器电路常用 RAM 芯片：

Intel 6116(2KB)、6264(8KB)、62256(32KB)等。

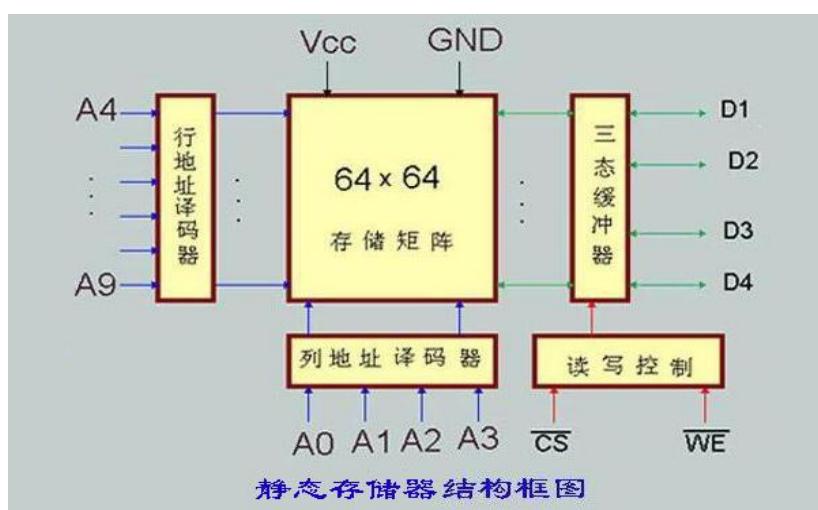


图 5.10 静态 RAM2114 引脚图

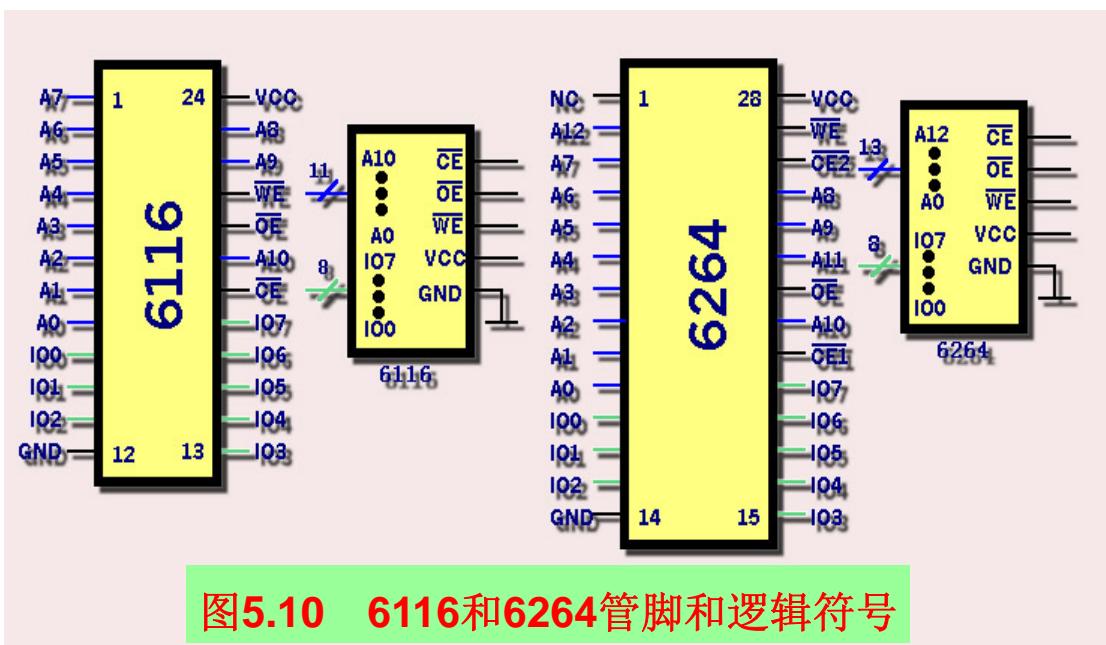


图5.10 6116和6264管脚和逻辑符号

工作方式	$\overline{CS}$	$\overline{OE}$	$\overline{WE}$	$D_i$
读	0	0	1	$D_{OUT}$
写	0	1	0	$D_{IN}$
禁止	1	×	×	Z

8031 (8051) 扩展 2KB RAM Intel 6116。

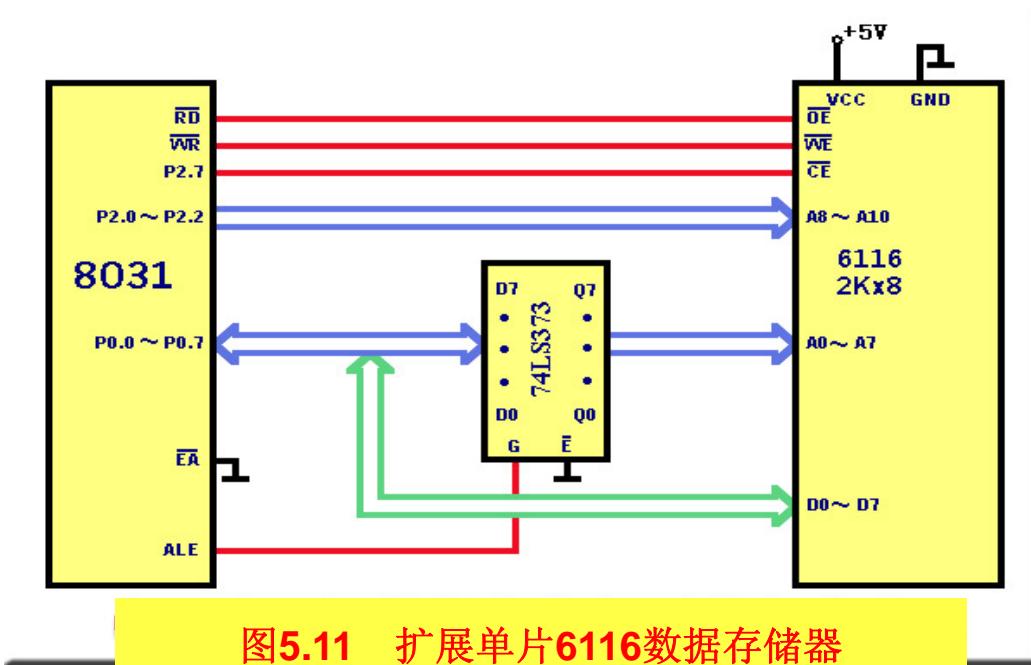


图5.11 扩展单片6116数据存储器

## 5-5 存储器综合扩展

数据存储器和程序存储器的综合扩展。

### 1、同时扩展数据存储器和程序存储器：

程序存储器的读操作有 PSEN 信号控制，数据存储器的读和写分别由 RD 和 WR 信号控制。不会造成操作上的混乱。

### 2、通过扩展可读写存储器：

(1) 利用 EEPROM 芯片扩展；(速度较慢)

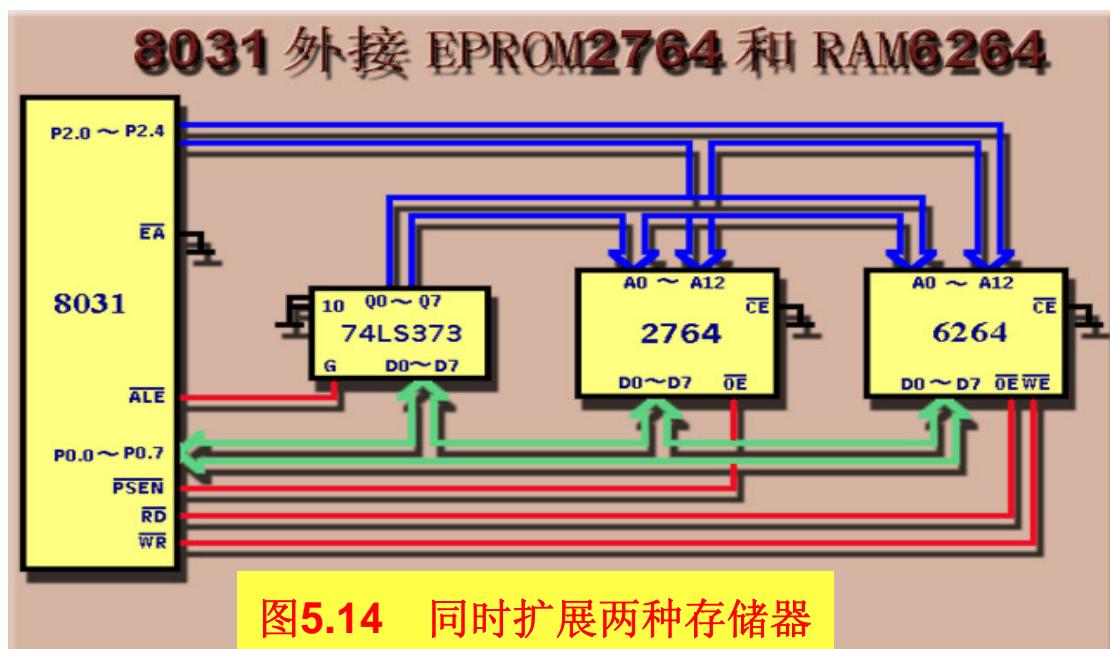
如：可扩展 2816 或 2817 等。

(2) 改造 RAM 存储芯片。(见 P131)

如：可改造 6116 等。

#### 5-5-1 同时扩展程序存储器和数据存储器

单片机连接 8KB EPROM 2764 和 8KB RAM 6264 各一片。

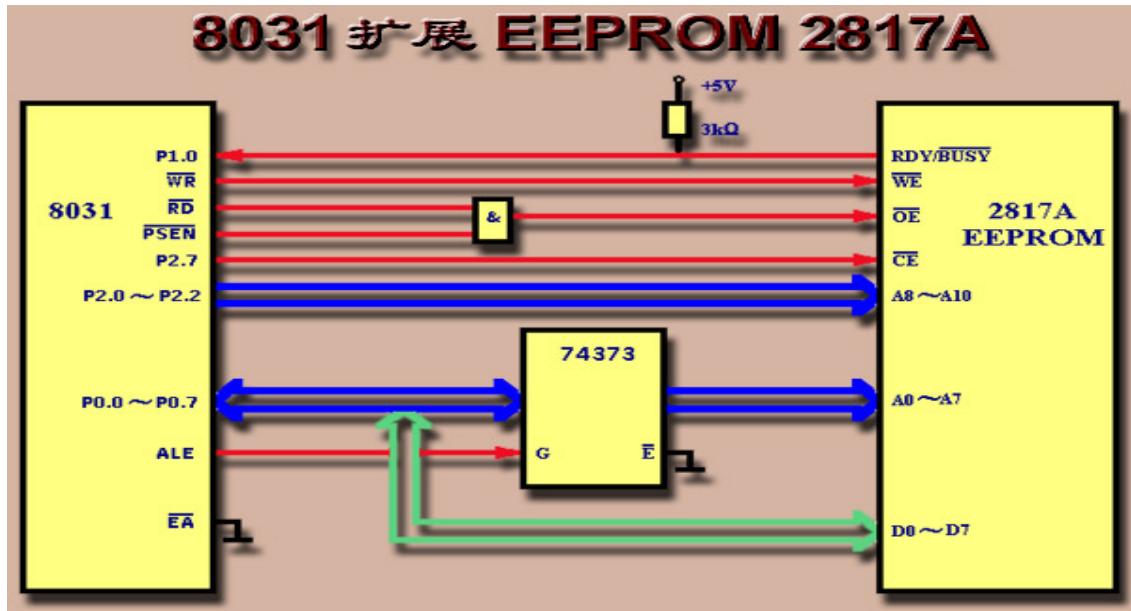


#### 5-5-2 扩展既可读又可写的程序存储器

EEPROM 既能作为程序存储器又能作数据存储器。

将程序存储器与数据存储器的空间合二为一。

片外存储器读信号= PSEN · RD



## 5-6 存储器系统的特性和使用

哈佛(Har-yard) 结构, 即将程序和数据存储器截然分开, 各有自己的寻址方式, 寻址空间和控制信号。 80C51 单片微机的存储器映像图。

一、特点: 复杂性:

- 1、程序存储器与数据存储器同时存在;
- 2、内外存储器同时存在;
- 3、存储器地址空间的重叠和连续。

二、使用:

存储器地址空间的区分和衔接:

- ① 在物理上设有 4 个物理存储空间:

程序存储器: 片内程序存储器;

片外程序存储器;

数据存储器: 片内数据存储器;

片外数据存储器。

② 在逻辑上设有 3 个逻辑存储空间：

1、内外程序存储器统一编址，形成一个完整的空间；

2、内外数据存储器分开编址，都是从“0”单元开始。

1、存储空间的区分：

(1) 内部程序存储器与数据存储器的区分；

(2) 外部程序存储器与数据存储器的区分；

(3) 内外数据存储器的区分。

	内 部	外 部
数据 存储器	MOV 指令	MOVX 指令 $\overline{RD}$ 、 $\overline{WR}$ 选通
程序 存储器	MOVC 指令 $\overline{EA}=1$	MOVC 指令 $\overline{PSEN}$ 选通 $\overline{EA}=0$

2、内外程序存储器的衔接。

内外 ROM 衔接形式

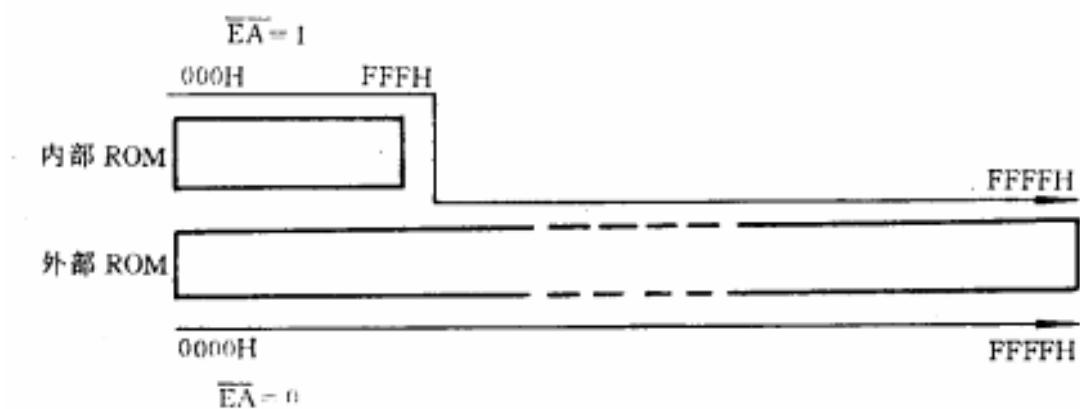


图3-15 内外程序存储器衔接示意图

## 小结:

- ❖ 构造系统总线，然后再往系统总线上“挂”存储芯片或 I/O 接口芯片。
- ❖ 超出总线负载能力，必须加总线驱动器。
- ❖ 复用技术——地址和数据进行分离，需用地址锁存器。
- ❖ 程序存储器可以分为片内和片外两部分，处理器访问片内和片外程序存储器，可由 EA 引脚所接的电平来确定。
- ❖ 存储器可分为 4 个物理存储空间和 3 个逻辑存储空间。
- ❖ 存储器扩展：程序存储器扩展；
  - 数据存储器扩展；
  - 综合扩展。
- ❖ 存储器系统的特点（哈佛结构）和编址技术：
  - 编址——线选法、译码法。
- ❖ 存储器的区分与衔接。

## 练习题:

(一) 问答题

(二) 填空题

(三) 选择题

# 第6章 中断与定时系统

## 一、教学要求：

掌握：计算机中断的概念，单片机中断系统的结构、中断源、中断特殊功能寄存器、中断响应过程；定时/计数器系统的电路结构、特殊功能寄存器及功能和使用方法。

理解：单片机中断、定时和计数的应用。

## 二、教学内容：

- 6. 1 单片机中断系统
- 6. 2 单片机的定时器/计数器
- 6. 3 单片机外部中断源的扩展
- 6. 4 定时器/计数器与中断综合应用举例

## 三、教学重点：

单片机中断系统的结构、中断源、中断特殊功能寄存器、中断响应过程；定时/计数器系统的电路结构、特殊功能寄存器及功能和使用方法。

## 四、教学难点：

单片机中断、定时和计数的应用。

五、建议学时：6 学时。

## 六、教学内容：

### 6-1 单片机中断系统

#### 6-1-1 中断技术

中断系统是计算机的重要指标之一。

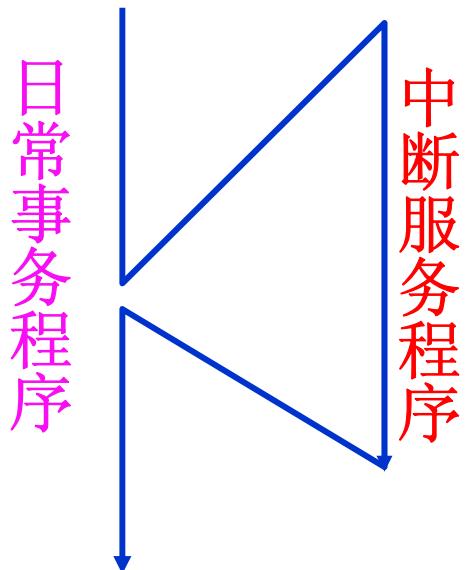
## 一、中断概念：

某人看书 执行主程序 中断过程

电话铃响 中断信号  $\overline{INT_x} = 0$  中断请求

暂停看书 暂停执行主程序 中断响应

书中作记号	当前 PC 值入栈	保护断点
电话谈话	执行中断程序	中断服务
继续看书	返回主程序	中断返回



### 二、两种中断：

#### 1. 可屏蔽中断：

可程控“开中断/关中断”。软件设置允许/禁止 CPU 响应中断。

#### 2. 非屏蔽中断：

不可程控“关中断”。有中断请求信号，CPU 必须响应。

### 三、中断源：

能发出中断请求信号的各种事件。

如 I/O 设备、定时时钟、系统故障、软件设定等。

有 3 类共 5 个中断：

#### 1、外中断 2 个：INT0、INT1

由引脚 INT0 (P3.2) 和 INT1 (P3.3) 引入。

#### 2、定时中断 2 个：T0、T1

无引入端，请求在芯片内部发生。以记数溢出信号作为中断请求，去置位一个溢出

标志位。

### 3、串行中断 1 个： RI/TI

无引入端，请求在芯片内部发生。接收或发送完一帧串行数据时，就产生一个中断请求。

INTR(Interrupt Request): 可屏蔽中断请求

NMI (Non-Maskable Interrupt): 非屏蔽中断

## 四、中断优先级控制原则和控制逻辑：

中断优先级是为中断嵌套服务的。

### 1、优先级控制原则：

(1) 低优先级中断不能打断高优先级的中断服务；但高优先级中断请求信号可以打断低优先级的中断服务，从而实现中断嵌套。

(2) 如果一个中断请求已被响应，则同级的其它中断服务将被禁止。即同级中断不能嵌套。

(3) 如同级的多个中断请求同时出现，则按 CPU 查询次序确定哪个中断请求被响应。

查询次序为： INT0→T0→INT1→T1→RI/TI。

### 2、控制逻辑：

(1) 利用中断优先级控制寄存器；

(2) 2 个不可寻址的优先级状态触发器：状态“0”或“1”。

## 五、寻找中断源和确定优先级：

寻找中断源：每个中断源对应一个中断服务程序。

多个中断源按优先级别排队。硬件上排队顺序： DMA、NMI、INTX。

### 1、软件查询方式：

(1) 中断源查询电路：

(2) 软件查询程序：

```

INTS: MOV A, P1           ; 读中断源寄存器

JB ACC.0, SV1  ; 查询高级中断请求

JB ACC.1, SV2  ; 查询低级中断请求

...

```

```

SV1: ... ; 中断服务程序 1

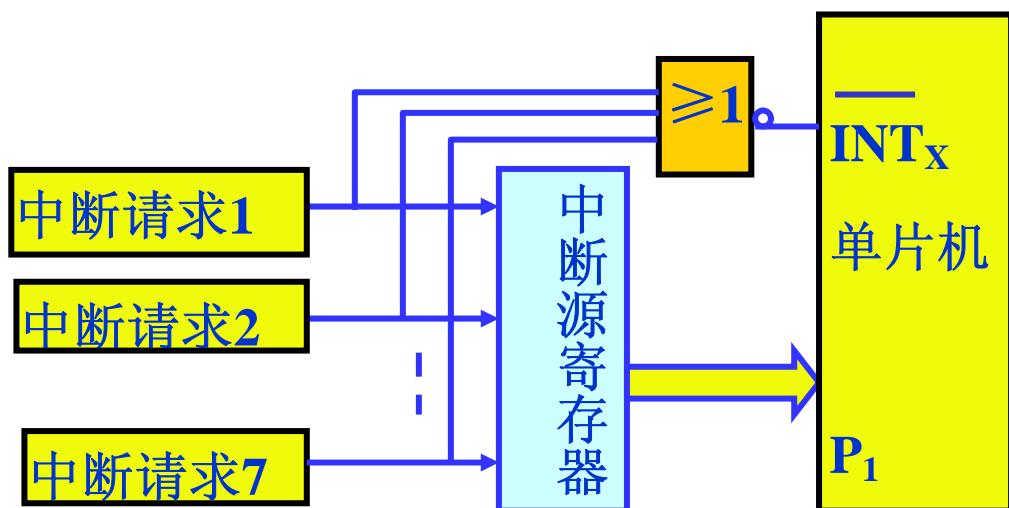
...

```

```

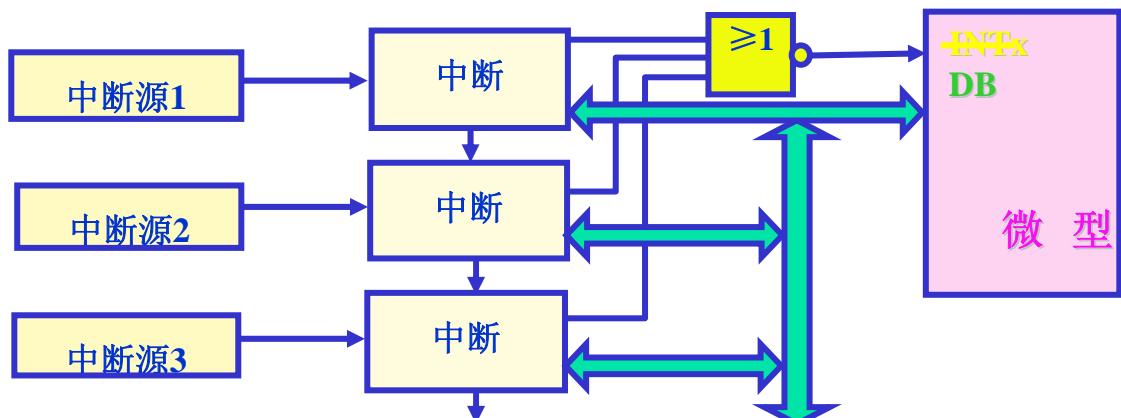
SV2: ... ; 中断服务程序 2

```



2、硬件查询方式：

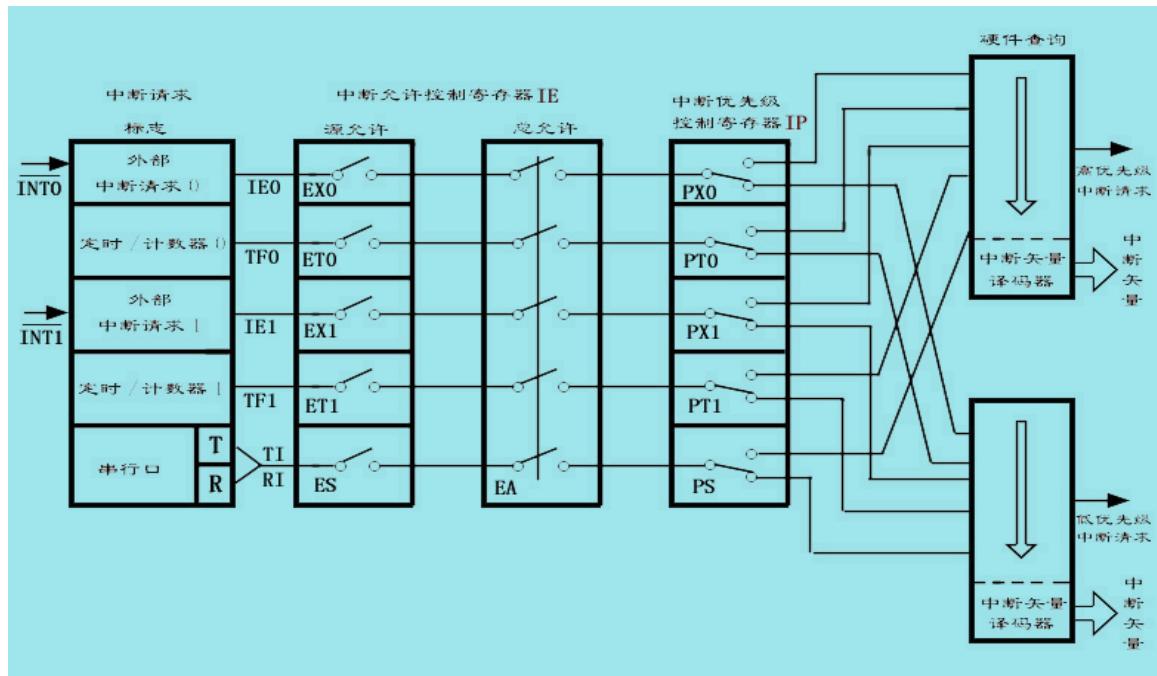
硬件优先级排队和中断向量锁存电路。



中断向量：提供中断服务程序入口地址信息的地址。

## 6-1-2 中断系统控制

### 一、MCS-51 中断系统内部结构



#### 1、中断源信号：

2 个外部中断源信号：INT0、INT1；

2 个定时器 T0、T1 溢出中断请求：TF0、TF1；

1 个串行口数据发送、接收结束中断请求：TI、RI。

#### 2、中断允许控制：

总允许开关：EA；

源允许开关：ES、ET1、EX1、ET0、EX0。

#### 3、2 级中断优先级控制：

优先级选择开关：PS、PT1、PX1、PT0、PX0。

### 二、中断控制寄存器：

寄存器名称		D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
定时器控制 寄存器	TCON (88H)	TF1		TF0		IE1	IT1	IE0	IT0
	位地址	8FH	8EH	8DH	8CH	8BH	8AH	89H	88H
串行口控制 寄存器	SCON (98H)							TI	RI
	位地址	9FH	9EH	9DH	9CH	9BH	9AH	99H	98H
中断允许 寄存器	IE (A8H)	EA			ES	ET1	EX1	ET0	EX0
	位地址	AFH			ACH	ABH	AAH	A9H	A8H
中断优先级 寄存器	IP (B8H)				PS	PT1	PX1	PT0	PX0
	位地址				BCH	BBH	BAH	B9H	B8H

1. 中断标志位： TF1、TF0、IE1、IE0、RI 、 TI

登记各中断源请求信号： =1， 有中断请求； =0， 无中断请求。

CPU 响应中断后， 该中断标志自动清零。 TI， RI 标志必须软件清零。

2. 外部中断触发方式选择位： IT0、IT1

=1： 负边沿触发中断请求； =0： 低电平触发中断请求。

3. 中断允许控制位： EA、ES、ET1、EX1、ET0、EX0

=1 开中断； =0 关中断。

例： 允许 CPU 响应 INT0 的中断请求。

SETB EX0

SETB EA

4. 中断优先级控制位： PS、PT1、PX1、PT0、PX0

2 级优先级： =1 为高优先级； =0 为低优先级。

同一优先级别按内部查询顺序排列优先级：

高 INT0、T0、INT1、T1、TI/RI 低。

### 6-1-3 中断处理过程

一、 中断响应条件：

1. 有中断请求信号；

2. 系统处于开中断状态。

## 二、中断响应过程:

1. 关中断: 屏蔽其它中断请求信号。
2. 保护断点: 将断点地址压入堆栈保存, 即当前 PC 值入栈。
3. 寻找中断源: 中断服务程序入口@PC, 转入中断服务。
4. 保护现场: 将中断服务程序使用的所有寄存器内容入栈。
5. 中断处理: 执行中断源所要求的程序段。链接中断处理
6. 恢复现场: 恢复被使用寄存器的原有内容。
7. 开中断: 允许接受其它中断请求信号。
8. 中断返回: 执行 RETI 指令, 栈顶内容→PC, 程序跳转回断点处。

RETI= RET 指令 + 通知 CPU 中断服务已结束

## 中断处理:

- 1、中断采样——仅对外中断 (INT0、INT1) 请求信号;
- 2、中断查询: 单片机在每一个机器周期的最后一个状态 S6,

按优先级顺序对中断请求标志位进行查询, 即先查询高级中断后再查询低级中断, 同级中断按“INT0→T0→INT1→T1→RI/TI”的顺序查询。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。

由于中断请求是随机发生的, CPU 无法预先得知, 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中不停地重复进行。

- 3、中断响应: 当查询到有效的中断请求时, 就进行中断响应。其主要内容是由硬件自动生成一条长调用指令 LCALL。其格式为: LCALL addr16, addr16 即是由系统设定的 5 个中断程序的入口地址。

各中断源中断服务程序的入口地址。

如下表:

中 断 源	中 断 入 口 地 址
$\overline{\text{INT}_0}$	0003H
$T_0$	000BH
$\overline{\text{INT}_1}$	0013H
$T_1$	001BH
RI/TI	0023H

### 三、中断响应阻断：

1. CPU 正处在为一个同级或高级的中断服务中。即当有同级或高级中断服务。
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。  
即当 CPU 未执行完一条指令。
3. 当前执行返回指令 RET/RETI 或访问 IE、IP 的指令后，不能立即响应中断，还应再执行一条指令，然后才能响应中断。  
程序单步执行就是利用此原理，借助单片机的外部中断功能来实现的。

### 四、中断响应周期时序：

每个机器周期的最后一个状态采样中断标志位，若有中断请求，将在下一个机器周期的第一个状态按优先级顺序进行中断查询。

#### 中断响应时间：

正常中断响应时间为 3~8 个机器周期，如果有同级或高级中断服务，将延长中断响应时间。

3T： 中断请求标志查询： 1T

产生、执行 LCALL： 2T

8T： 执行 RET/RETI (访问 IP/IE): 2T

主程序中 MUL/DIL 指令： 4T

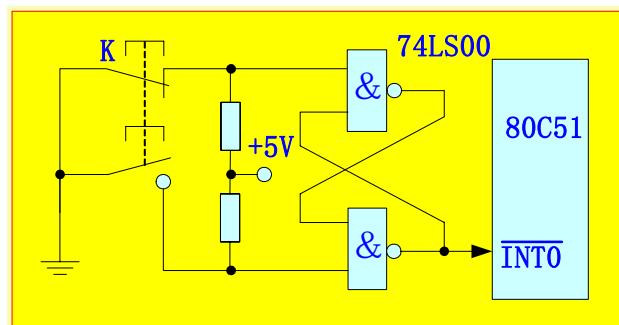
执行 LCALL 指令: 2T

单步工作方式:

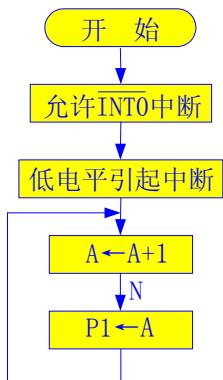
按一次键执行一条主程序的指令。

1、建立单步执行的外部控制电路。

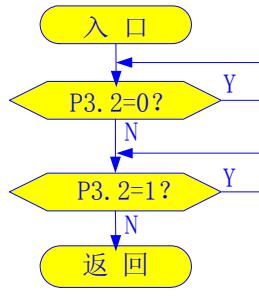
如图:



2、编写外部中断的中断服务程序:



主程序流程图



INT0中断服务  
程序流程图

ORG 0000H

SJMP START

ORG 0003H ; 中断程序

JNB P3.2, \$ ; =0 则“原地踏步”

JB P3.2, \$ ; =1 则“原地踏步”

RETI

START: MOV IE, #81H ; 主程序

```
MOV TCON, #00H
LOOP: INC A
      MOV P1, A
      SJMP LOOP
      END
```

#### 6-1-4 中断请求的撤消

中断响应后，TCON 或 SCON 中的中断请求标志应及时清除。否则就意味着中断请求仍然存在。

##### 1、定时中断硬件自动撤除定时：

中断响应后，硬件自动把标志位(TF0/TF1)清 0，因此定时中断的中断请求是硬件自动撤除的。不需要用户干预。

##### 2、脉冲方式外部中断请求的撤消：硬件自动撤除

外部中断的撤消包括两项内容：

(1) 中断标志位的置“0”：中断响应后由硬件电路自动完成；

(2) 外中断请求信号的撤消：随脉冲信号过后消失随即自动撤消。

##### 3、电平方式外部中断请求的撤消：自动与强制撤除

通过硬件自动地使标志位(IE0 或 IE1)清 0。电平请求方式光靠清除中断标志，并不能彻底解决中断请求的撤除问题。需在中断响应后把中断请求输入端从低电平强制改为高电平。

D 触发器的直接置位端 SD (得到负脉冲)：

ORL P1, #01H ; P1 输出高电平

ANL P1, #0FEH ; P1 输出低电平

软硬件相结合完成：硬件——自动清标志位 (IE0 或 IE1 清 0)；

软件——撤消中断请求信号 (由低电平改为高电平)。

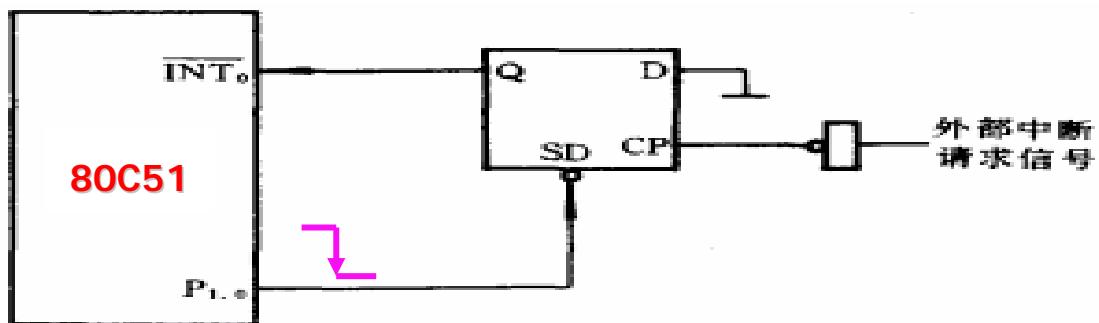


图6.2 电平方式外部中断请求的撤除电路

中断请求的撤消

4、串行中断请求的撤除：由软件方法完成

串行中断的标志位是 TI 和 RI，但对这两个中断标志不进行自动清 0。因为在中断响应后还需测试这两个标志位的状态，以判定是接收操作还是发送操作，然后才能清除。所以串行中断请求的撤除也应使用软件方法，在中断服务程序中进行。需由用户完成。

## 6-2 外部中断源的扩展

一、多中断源系统：

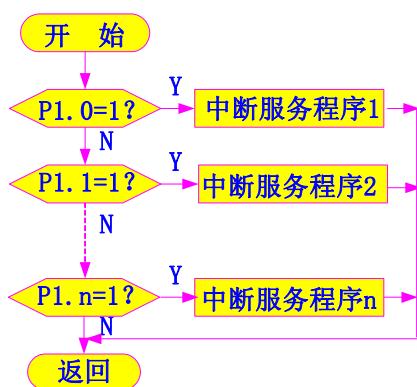
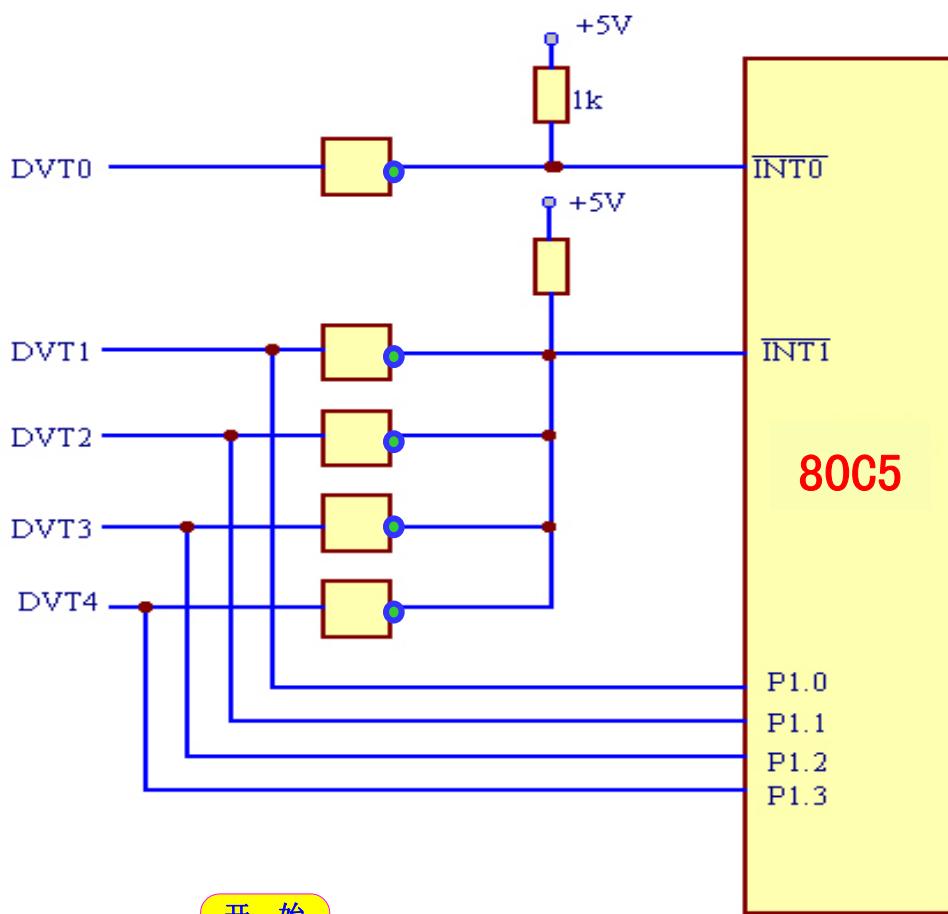
当外部中断源多于中断输入引脚时，可采取以下措施：

- 1、用定时器计数输入信号端 T0、T1 作外部中断入口引脚；
- 2、用串行口接收端 RXD 作外部中断入口引脚。
- 3、用一个中断入口接受多个外部中断源，并加入中断查询电路。

应把  $\overline{INT_x}$  设置为电平触发方式；

在中断服务程序中进行扩展中断源的查询；

查询顺序就是扩展中断源的优先级顺序。



## 二、处理外部中断举例：

要求每次按动按键，使外接发光二极管 LED 改变一次亮灭状态。

解：  $\overline{INT_0}$  输入按键信号， P1.0 输出改变 LED 状态。

1、跳变触发：每次跳变引起一次中断请求。

ORG 0000H ; 复位入口 AJMP MAIN

ORG 0003H ; 中断入口 AJMP PINT0

```

ORG 0100H ; 主程序

MAIN: MOV SP, #40H ; 设栈底

SETB EA ; 开总允许开关

SETB EX0 ; 开 INT0 中断

SETB IT0 ; 负跳变触发中断

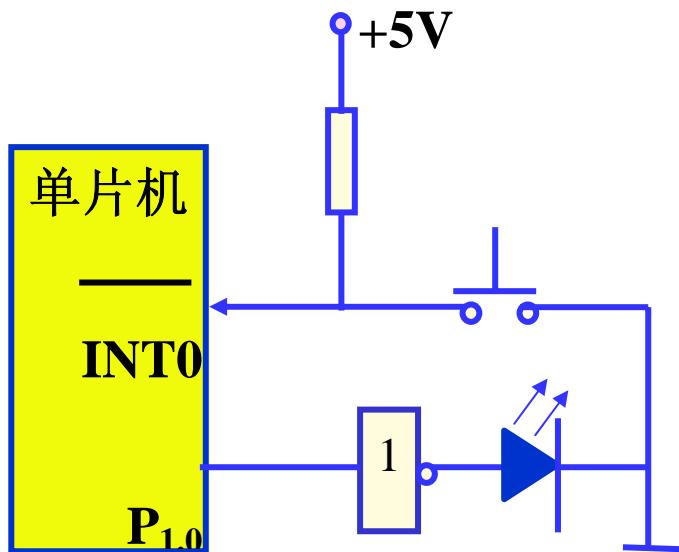
H: SJMP H ; 执行其它任务

ORG 0200H ; 中断服务程序

PINT0: CPL P1.0 ; 改变 LED

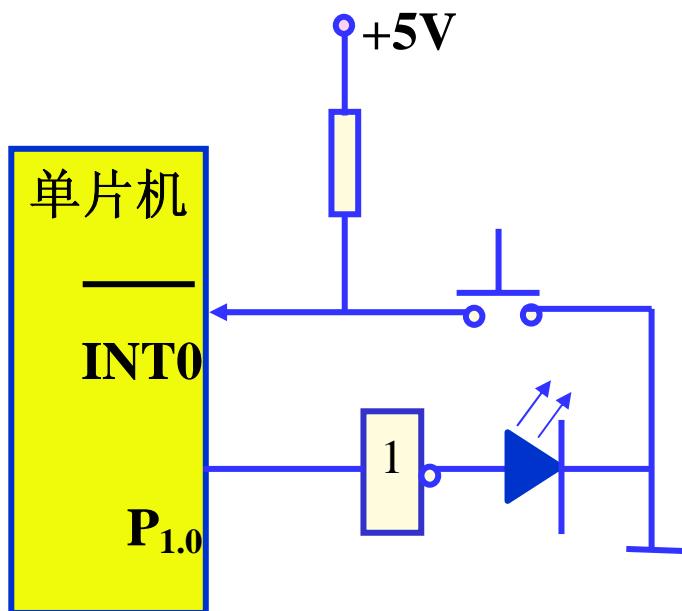
RETI ; 返回主程序

```



2、电平触发：可避免一次按键引起多次中断响应。

- (1) 软件等待按键释放 (撤消低电平);
- (2) 硬件清除中断信号 (标志位)。



```

ORG 0000H ; 复位入口
AJMP MAIN

ORG 0003H ; 中断入口
AJMP PINT0

ORG 0100H ; 主程序

MAIN: MOV SP, #40H ; 设栈底
      SETB EA ; 开总允许开关
      SETB EX0 ; 开 INT0 中断
      CLR IT0 ; 低电平触发中断

H: SJMP H ; 执行其它任务

ORG 0200H ; 中断服务程序

PINT0: CPL P1.0 ; 改变 LED

WAIT: JNB P3.2, WAIT ; 等按键释放 (P3.2 即 INT0)

RETI ; 返回主程序

```

### 6-3 定时器/计数器

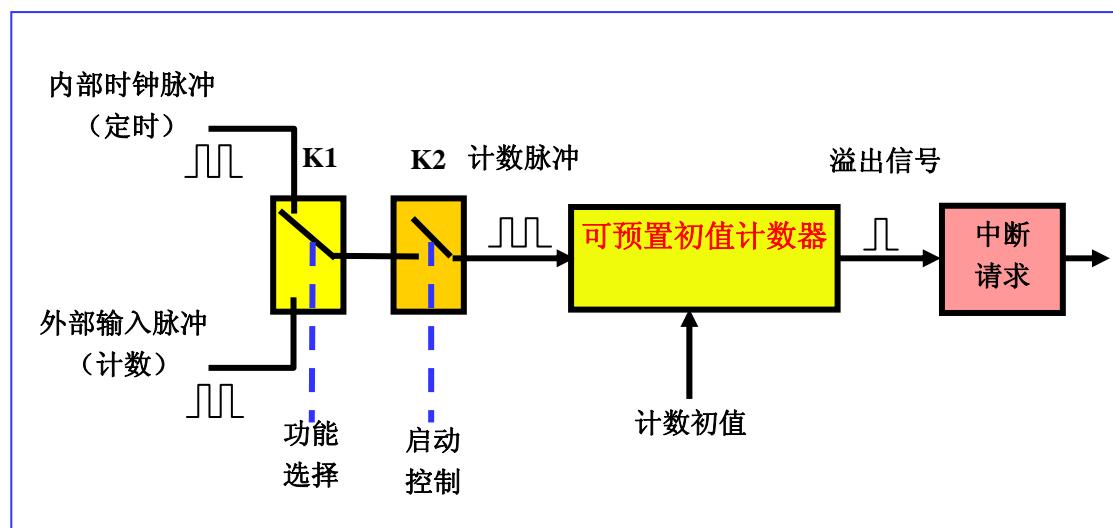
#### 一、定时方法:

- 1、硬件定时: 较长时间; 不够灵活、方便;
- 2、软件定时: 编循环程序, 时间精确; 占用 CPU;
- 3、可编程定时器定时: 对系统时钟脉冲记数, 灵活、方便。

#### 二、定时器/计数器的工作原理:

定时器/计数器中的核心部件为可预置初值计数器。预置初值后开始计数, 直至计数值回 0 或产生溢出, 可申请中断。

计数器有加 1 计数或减 1 计数两种形式。



6-3-1 MCS-51 定时器/计数器

2 个可独立控制的 16 位定时器/计数器:

加法计数器 T0、T1

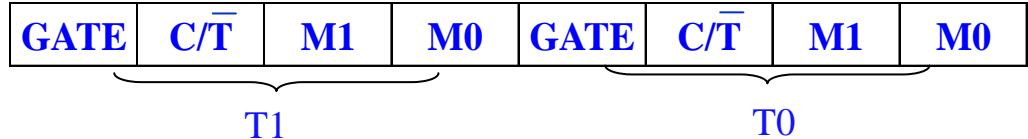
定时器初始化编程包括:

- 1.功能选择 (定时/计数);
- 2.位数选择 (8/13/16 位);
- 3.启动方式选择 (内部启动/外部启动);
- 4.启动控制 (启动/停止);

5. 恢复初值方式 (自动重装/软件重装)。

一、定时器控制、状态寄存器:

1、TMOD 定时器方式寄存器 (89H):



1) 功能选择位 C/T:

=0: 定时功能, 计数内部机器周期脉冲;

=1: 计数功能, 计数引脚 T0(T1)输入的负

2) 方式选择位 M1、M0: 4 种工作方式: 13/16/8 位

M1 M0	方 式	功 能 描 述
0 0	0	13位
0 1	1	16位
1 0	2	8位自动重装
1 1	3	T <sub>0</sub> 为2个8位

3) 门控方式选择位 GATE:

=0, 非门控方式(内部启动):

TR<sub>x</sub>=1, 启动定时器工作;

TR<sub>x</sub>=0, 停止定时器工作。

=1, 门控方式(外部启动):

TR<sub>x</sub>=1 且引脚 INT<sub>x</sub>=1 才启动。

确定定时器工作方式指令:

MOV TMOD, #方式字

例: 设 T0 用方式 2 非门控定时,

T1 用方式 1 门控计数。

MOV TMOD, #?

2、TCON 定时器控制/状态寄存器：

1) 启动控制位 TR0、TR1：

=0, 停止定时器工作；

=1, 启动定时器工作。

例：启动 T0： SETB TR0

2) 溢出中断标志位 TF0、TF1：

定时器溢出使  $TF_x=1$ , 引起中断请求, CPU 响应  $T_x$  中断后, 系统自动将  $TF_x$  清 0。

当然, 也可用软件检测  $TF_x$ , 这时必须软件清 0。

WAIT: JBC TF0, NEXT ; 检测 T0 是否溢出

SJMP WAIT ; 未溢出, 继续检测

NEXT: ... ; 溢出, TF0 清 0, 处理溢出

3、可预置初值的 16 位加 1 计数器 TH0、TL0、TH1、TL1：

如：预置 T0 初值指令：

MOV TH0, #XH

MOV TL0, #XL

求  $THX$ 、 $TLX$  的方法：

例 1：T0 运行于计数器状态, 工作于方式 1 (16 位方式), 要求外部引脚出现 3 个脉冲后, TH0、TL0 全回 0 (以便申请中断)。求计数初值 C。

解： $C = (0003H)$  求补 = FFFDH

例 2：

T0 运行于定时器状态, 时钟振荡周期为 12MHZ, 要求定时  $100 \mu s$ 。求不同工作方式时的定时初值 C。

解:  $f_{osc}=12MHz$

$T=1 \mu s$

$X=100 \mu s / 1 \mu s = (100)D=64H$

方式 0(13 位方式):  $C=(64H)$  求补=0 0000 0110 0100 B+1

=1F9CH

方式 1(16 位方式):  $C=(64H)$  求补=0000 0000 0110 0100 B+1

=FF9CH

方式 2、3(8 位方式):  $C=(64H)$  求补=0110 0100 B+1

=9CH

注意: 工作方式 0 的初值装入方法:

1F9CH=0001 1111 1001 1100 B

可见,  $TH0=FCH$ ,  $TL0=1CH$

MOV TH0, #0FCH

MOV TL0, #1CH

二、定时器工作方式:

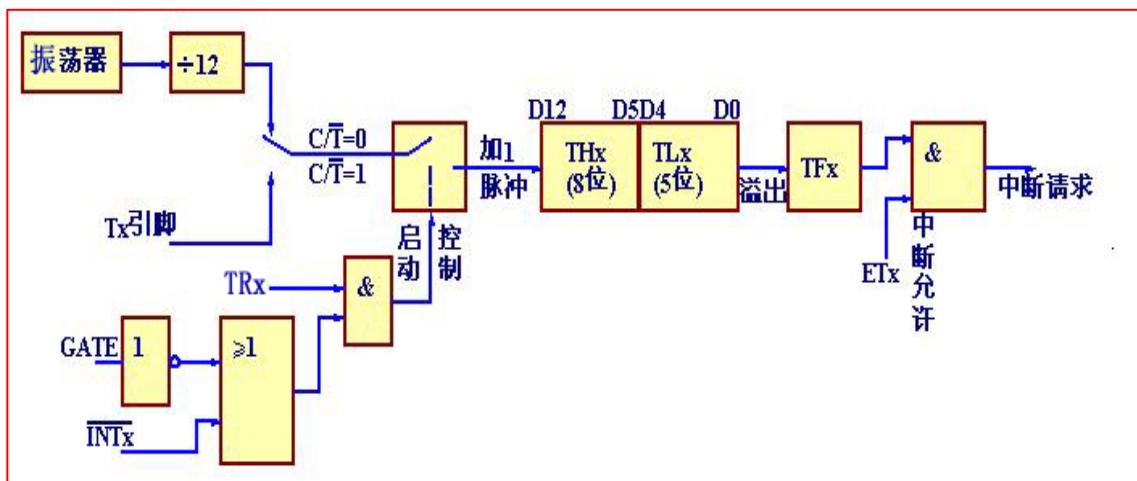
由方式选择位 M1、M0 设定。

1、方式 0: 13 位定时/计数器

$THx8$  位和  $TLx$  低 5 位组成加 1 计数器。

计数外部脉冲个数: 1~8192(213)

定时时间( $T=1\mu s$ ):  $1\mu s \sim 8.19ms$

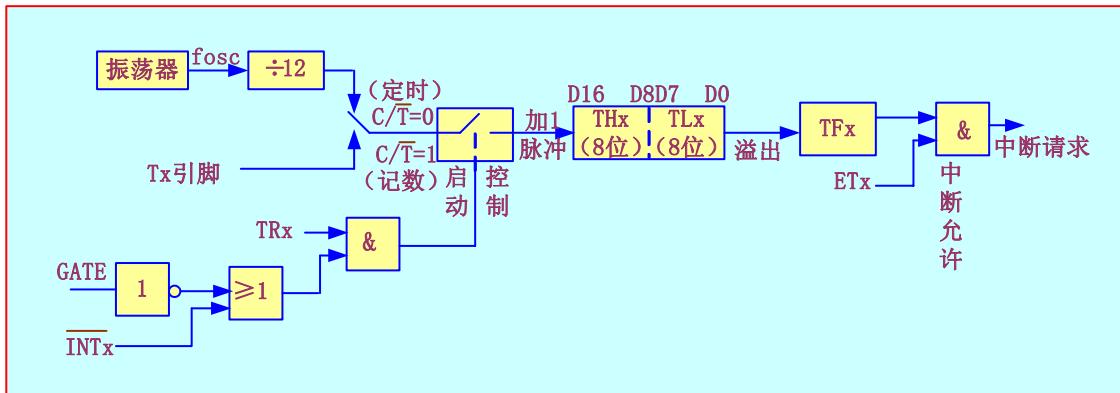


## 2、方式 1：16 位定时/计数器

THx8 位和 TLx8 位组成 16 位加 1 计数器。

计数外部脉冲个数：1~65536(216)

定时时间( $T=1\mu s$ )：  $1\mu s \sim 65536 \times T = 65.54ms$



## 3、方式 2：自动恢复初值 8 位定时/计数器。

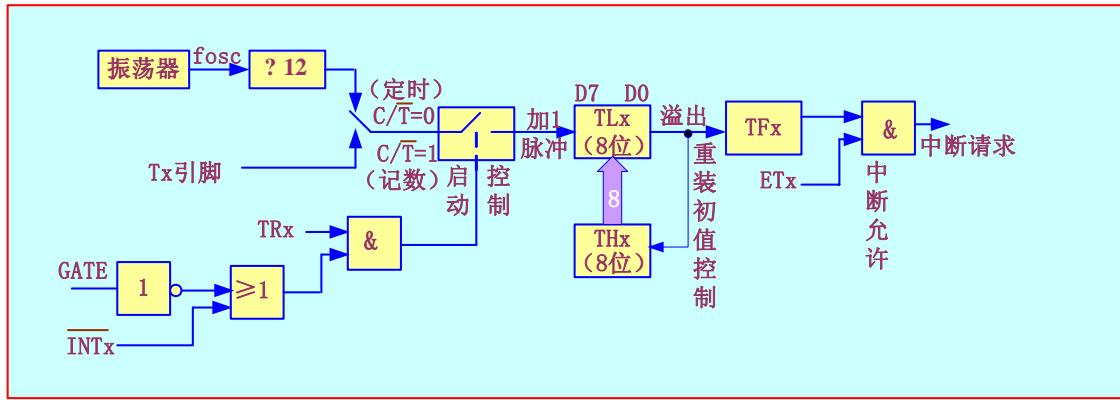
TLx 为 8 位加 1 计数器，

THx 为 8 位初值暂存器。

用于需要重复定时和计数の場合。

最大计数值：256 (28)

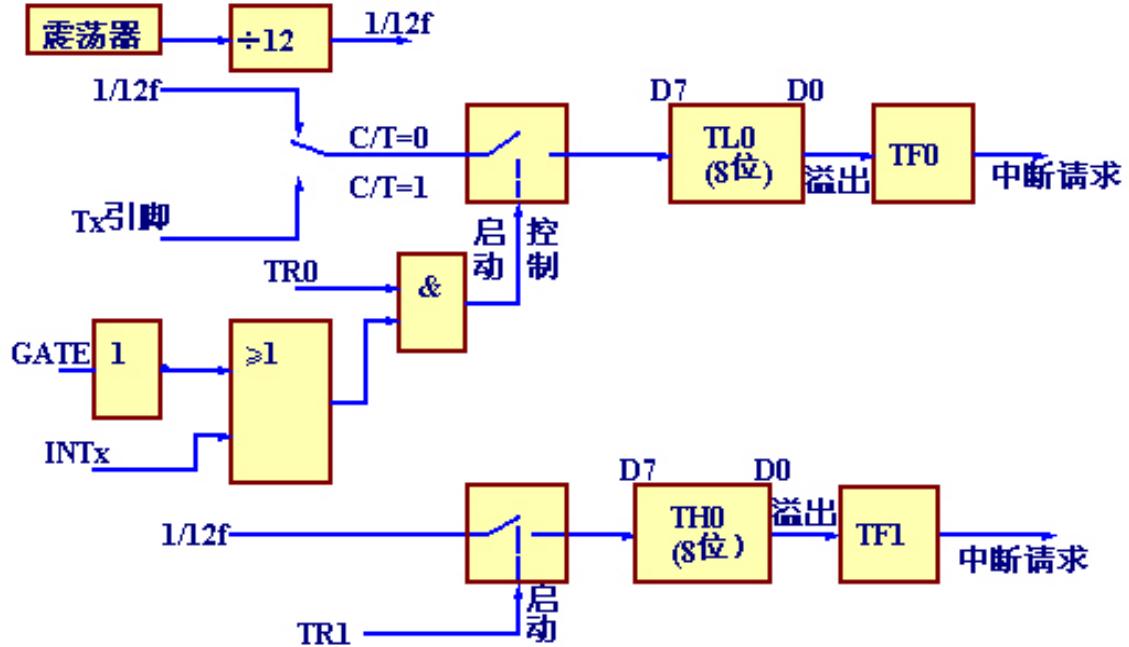
最大定时时间( $T=1\mu s$ )：  $256\mu s$



#### 4、方式 3：T0 分成 2 个 8 位定时器

TL0 定时/计数器和 TH0 定时器。

- TL0 占用 T0 控制位：C/T，TR0，GATE；
- TH0 占用 T1 控制位：TR1；
- T1 不能使用方式 3 工作。



### 三、MCS-51 定时器的应用：

#### 1. 计数功能：

- 生产线上产品计数：每个产品通过得到一个脉冲信号，计数器记录脉冲个数，

当计数值与设定值相等，启动包装机器。

- 检测转速：电机转动一圈发出一个脉冲，计数器记录一秒时间内脉冲个数，显示转速。

## 2. 定时功能：

- 用于实时控制：定时采样、定时启动等。

当定时时间与设定值相等，执行规定操作。

定时器初始化编程：

使用定时器工作之前，先写入控制寄存器，确定好定时器工作方式。

初始化编程格式：

```
MOV TMOD, # 方式字      ; 选择方式
MOV THx, #XH              ; 装入 Tx 时间常数
MOV TLx, #XL
(SETB EA)                 ; 开 Tx 中断
(SETB ETx)
SETB TRx                  ; 启动 Tx 定时器
```

需考虑：1. 按实际需要选择定时/计数功能；

2. 按时间或计数长度选择工作方式；

3. 计算时间常数：

计算时间常数 X(计算初值)：

计数功能：  $X = 2^n - \text{计数值}$  (n: 8/13/16)

定时功能：  $X = 2^n - t/T$  (t: 定时时间；

T: 机器周期)

## 4. 溢出处理编程格式：

1) 查询方式：先查询定时器溢出标志，再进行溢出处理。

```

...           ; 定时器初始化

WAIT: JBC    TFx, PT      ; 检测溢出标志

SJMP    WAIT

PT:  MOV    THx, #XH      ; 重装时间常数

MOV    TLx, #XL

...           ; 溢出处理

SJMP    WAIT

```

2) 中断方式: 初始化后执行其他任务, 中断服务程序处理溢出。

```

ORG    0000H

LJMP    MAIN

ORG    000BH(001BH)      ; Tx 中断入口

LJMP    PTS

MAIN:   ...           ; 初始化后执行其他程序

PTS:   ...           ; 溢出中断服务程序

MOV    THx, #XH      ; 重装时间常数

MOV    TLx, #XL

RETI

```

练习: 80C51 单片机选择 16 位加 1 计数器。

按要求选择功能和初值。

1、要求检测到 100 个脉冲, 发中断请求, 通知 CPU。

选计数功能, 计数初值为 156 。

2、要求定时每隔  $100\mu\text{s}$  时间, 发一次中断请求。

设内部时钟周期  $1\mu\text{s}$

选定时功能, 计数初值为 156 。

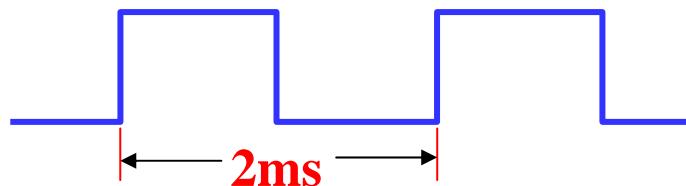
## 6-4 定时器/计数器的扩展

80C51 单片机内有 2 个 16 位的 T0 和 T1，如在应用中，定时器/计数器的数量或功能不能满足要求，则可在外部扩展。

扩展的芯片主要有 8253 (最高时钟频率为 2MHz) 和 8254 (最高时钟频率为 8MHz) 等。它们均为减法计数。各有 3 个独立的 16 位计数器 T0、T1 和 T2，它们既可用于计数，又可用于定时，并可运行在 0 模式~5 模式共 6 种不同的工作模式中。

## 6-5 定时器/计数器与中断综合应用举例

例 1：由 P1.0 输出方波信号，周期为 2ms，设  $f_{osc}=12MHz$ 。



解：每隔 1ms 改变一次 P1.0 的输出状态。

用 T1 非门控方式 1 定时。

计算时间常数： $X = 216 \cdot t/T = 216 \cdot 1000/1$

$$= FC18H$$

(1) 查询方式：

START: MOV TMOD, #10H

MOV TL0, #18H

MOV TH0, #0FCH

SETB TR0

LOOP: JBC TF0, PTF0

SJMP LOOP

PTF0: CPL P1.0

```
MOV      TL0, #18H
MOV      TH0, #0FCH
SJMP    LOOP
```

(2) 中断方式:

```
ORG    0000H
AJMP   MAIN
ORG    001BH
AJMP   PT0INT
ORG    0100H
MAIN:  MOV    SP, #60H
        MOV    TMOD, #10H
        MOV    TL0, #18H
        MOV    TH0, #0FCH
        SETB   EA
        SETB   ET0
        SETB   TR0
HERE:  SJMP   HERE
PT0INT: CPL   P1.0
        MOV    TL0, #18H
        MOV    TH0, #0FCH
        RETI
```

例 2: P1.7 驱动 LED 亮 1 秒灭 1 秒地闪烁, 设时钟频率为 6MHz。

长定时方法: 增加一个软件计数器或一个硬件计数器。

$T=2\mu s$ ,  $X=5 \times 105$  个  $T$ , 而最大只能 65536 个  $T$ ,

不能满足要求，必须借助硬件计数器或软件循环。

T0 定时初值：（方式 1）

$t=10\text{ms}$ ,  $X=5000\text{D}=1388\text{H}$ ,  $C=(1388\text{H})$  补= $EC78\text{H}$

T1 计数初值：（方式 2）

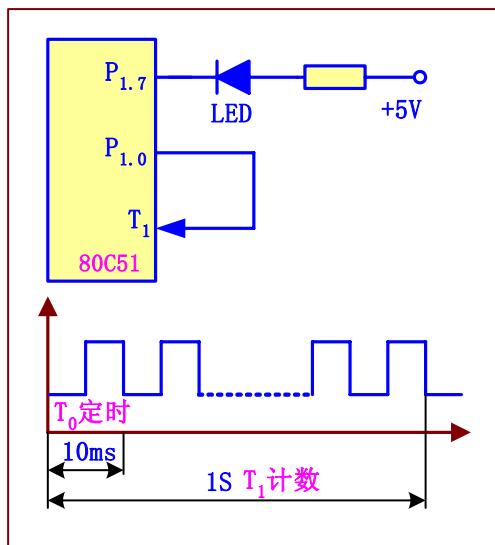
$X=100\text{D}$ ,  $C=(64\text{H})$  补= $9\text{CH}$

本题采用硬件方式：

T0 定时，定时 10ms；

T1 计数 T0 的定时跳变信号 P1.0 的负跳变次数，计满 100 个跳变为 1 秒。（查询方法）

```
START: MOV TMOD, #61H
        MOV TL1, #9CH
        MOV TH1, #9CH
        CLR P1.0
        SETB TR1
LOOP1: CPL P1.7
LOOP2: MOV TL0, #78H
        MOV TH0, #0ECH
        SETB TR0
LOOP3: JBC TF0, LOOP4
        SJMP LOOP3
LOOP4: MOV TL0, #78H
        MOV TH0, #0ECH
        CPL P1.0
        JBC TF1, LOOP1
        SJMP LOOP2
```



## 小结

1、5个中断源及其对应入口地址：

外中断2个：INT0、INT1

定时中断2个：T0、T1

串行中断1个：RI/TI

2、中断优先原则：分2个优先级；

中断源	中断入口地址
INT <sub>0</sub>	0003H
T <sub>0</sub>	000BH
INT <sub>1</sub>	0013H
T <sub>1</sub>	001BH
RI/TI	0023H

同级： INT0→T0→INT1→T1→RI/TI。

3、中断响应过程：

4、中断系统内部结构：

5、中断请求的撤消：3类中断请求撤消的异同点。

6、中断源的扩展。

7、定时器/记数器的工作原理、4种工作方式。

8、定时器/记数器与中断的综合应用。

9、利用定时器/计数器形成中断源：

把定时器/计数器 T0 (T1) 设置成记数状态，并把计数器初值设置成只要加 1 个外部脉冲后计数器就回 0 的数值时，在打开了中断的条件下，外部信号只要出现 1 个脉冲周期就可引起中断请求。

当计数器设置成方式 0 时，计数器初值应为 FF1FH；

当计数器设置成方式 1 时，计数器初值应为 FFFFH；

当计数器设置成方式 2/3 时，计数器初值应为 FFH。

方式 2 能自动重装初值；

方式 0、1、3 不能，必须需重装。

## 第7章 I/O扩展及应用

### 一、教学要求:

掌握: 单片机扩展 8255、8155、8279、键盘、显示器和打印机的接口技术和方法。

了解: 8255、8155 和 8279 等芯片的结构原理及与单片机的接口技术。

### 二、教学内容:

7.1 单片机为什么需要 I/O 扩展

7.2 单片机简单 I/O 扩展

7.3 8255 可编程通用并行接口芯片

7.4 8155 带 RAM 和定时器/计数器的可编程并行接口芯片

7.5 8279 可编程键盘/显示器接口芯片

7.6 单片机键盘接口芯片

7.7 单片机显示器接口芯片

7.8 单片机打印机接口技术

### 三、教学重点:

单片机扩展 8255、8155、8279、键盘和显示器的接口技术和方法。

### 四、教学难点: 单片机的接口技术。

### 五、建议学时: 6 学时。

### 六、教学内容:

#### 7-1 为何要扩展 I/O

原因: 1、单片机本身接口功能有限;

2、控制应用中的复杂接口要求: 速度差异大、设备种类繁多、数据信号形式多种多样。

I/O 设备必须通过 I/O 接口与计算机连接。

I/O 接口的功能:

1、速度协调:

锁存数据、传送联络信号。

2、数据格式转换:

并—串转换、A/D、D/A 转换。

3、电平转换:

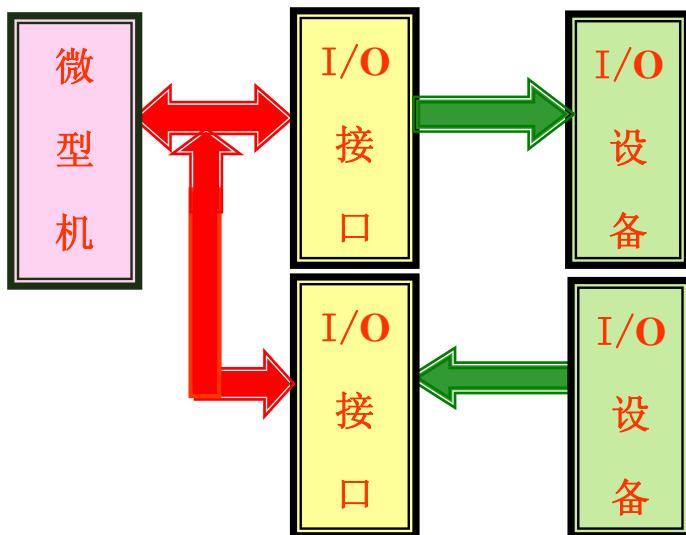
电平幅值或正/负逻辑转换。

4、隔离:

多个设备信号通过接口三态门隔离干扰信号。

5、输入三态缓冲:

6、输出锁存驱动: 驱动多个逻辑部件或大功率执行部件。



### 7-1-1 I/O 接口信号及编址技术

一、CPU 与 I/O 之间接口信号:

每个 I/O 接口分配有对应的 I/O 地址。

1、数据信息:

微型机与外设交换的数据，经接口数据缓冲器传送。

2、状态信息:

反映外设工作状态的信号。

### 3、控制信息：

设定 I/O 电路工作方式的信号。

数据总线传递三种信息，用不同 I/O 地址区别：

输入/输出数据缓冲器共用一个 I/O 地址：数据端口。

状态/控制寄存器共用一个 I/O 地址：控制/状态端口。

## 二、I/O 编址方式：

需要编址的子系统：存储器和接口电路。

### 1、存储器地址方式：统一编址

I/O 接口共用存储器的地址空间，每个 I/O 端口视为一个存储单元。

### 2、专用 I/O 地址方式：独立编址

有专用 I/O 控制信号和 I/O 指令。I/O 接口独立编址，不占用存储器的地址空间。如的 Z80。

MCS-51、96 为存储器地址方式（统一编址）。

MCS-51 单片机有片内 I/O 接口和扩展 I/O 接口。

片内 I/O 接口寄存器在 SFR 中，使用片内数据存储器空间，扩展 I/O 接口使用片外数据存储器地址空间：

输出指令：

片内寻址：MOV P1, A

输入指令：

MOV A, P1

片外寻址：MOVX @DPTR, A

MOVX A, @DPTR

MOVX @R0, A

MOVX A, @R0

## 7-1-2 I/O 控制方式

### 一、无条件传送（同步程序传送）方式：

已知 I/O 设备准备就绪，可直接进行数据传送。

适用：1、具有常驻的或变化缓慢的数据信号的设备。

如：指示灯、数码管等；

2、工作速度快，足以和单片机同步工作的设备。

如：DAC 等。

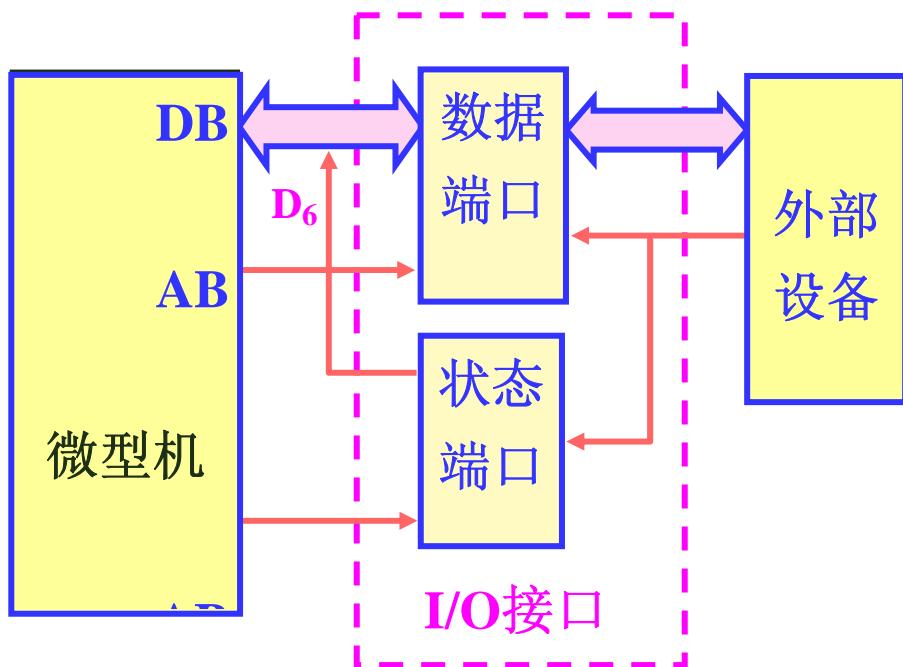
二、查询方式：

（有条件传送方式）

先查询 I/O 设备当前状态，若准备就绪，则交换数据，否则循环查询状态。

1.硬件查询电路：

设置状态锁存和数据锁存电路。



2.软件查询程序：

先输入状态，决定是否进行数据传送。

INPUT: MOV DPTR, #STATUS ; 状态口地址

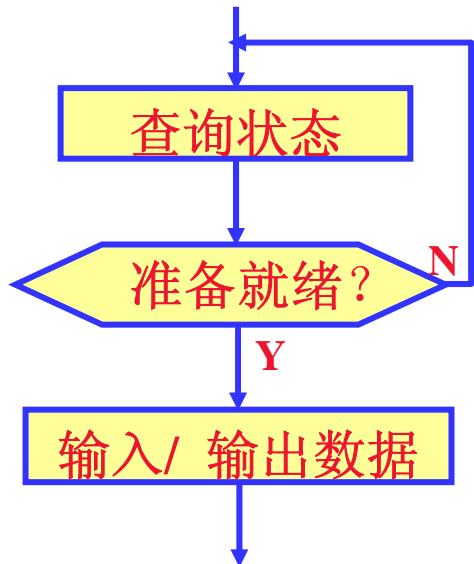
WAIT: MOVX A, @DPTR

JB ACC.6, WAIT

```
MOV      DPTR, #DATA      ; 数据口地址  
MOVX    A, @DPTR
```

查询方式只适用：

单道作业、规模比较小的单片机系统。



三、中断方式（程序中断方式）：

大多数时间计算机与外设并行工作，计算机不必因等待而浪费资源。当外设准备就绪，向 CPU 发出中断请求信号。CPU 暂停当前程序，执行 I/O 操作。当 I/O 操作结束，CPU 仍继续被中断的工作。

四、直接存储器存取方式传送：(DMA—Direct Memory Access)

用于计算机与高速外设进行大批量数据交换，由 DMA 控制器接管总线控制权，RAM 与外设之间直接数据传输，不需 CPU 的介入。

## 7-2 单片机并行接口

片内接口寄存器在 SFR 中的映象地址：

- 1、I/O 数据锁存器：P0、P1、P2、P3、SBUF。
- 2、I/O 控制/状态寄存器：IE、IP、TCON、TMOD、SCON、PCON。

并行 I/O 接口：

用于微型机与外部设备之间并行传送数据。

### 7-2-1 MCS-51 的并行接口

4 个 8 位双向并行 I/O 接口：

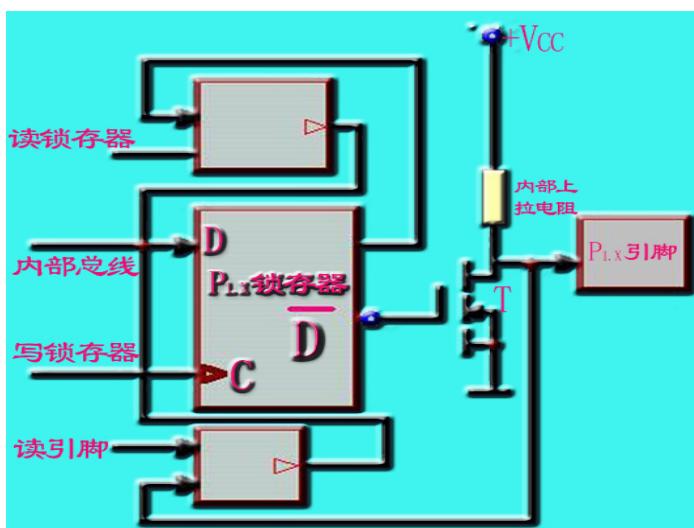
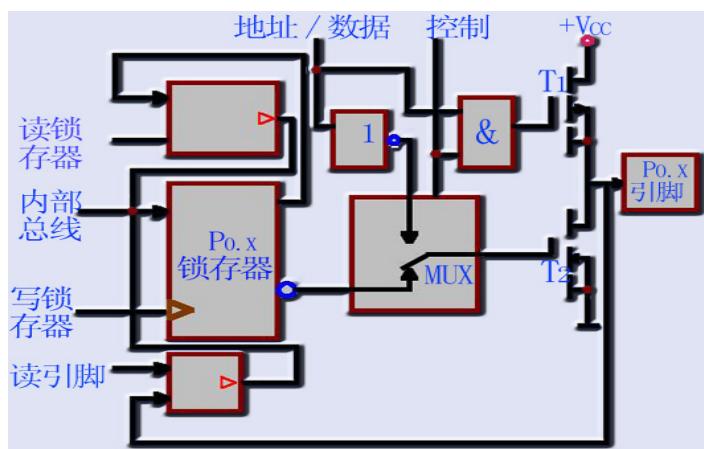
P0.0 ~ P0.7、P1.0 ~ P1.7、P2.0 ~ P2.7、P3.0 ~ P3.7。

均为多功能 I/O 接口，CPU 按当前操作自动进行功能切换。

#### 一、并行接口内部结构：

输出锁存器、输出驱动器、输入缓冲器及多路功能切换电路。

- 1) 输出操作：输出数据经过内部总线暂存到输出锁存器中，经  
过输出驱动器送到 I/O 引脚上。输出锁存器的内容可读入修改。
- 2) 输入操作：I/O 引脚输入数据经过输入缓冲器送到内部总线上。



## 二、并行接口的三种操作：

1.输出锁存：输出将使数据写入输出锁存器。

输出指令：MOV P1, A

MOV P1.0, C

2.输入三态：输入从 I/O 引脚上输入信号，读信号打开，引脚信号通过下三态门进入内部总线。为保证可靠输入，先写入“1”。

MOV P1, #0FFH ; 使输出驱动器截止

MOV A, P1 ; 输入 P0~P3 的复位状态均为 FFH，自动处于输入状态。

3.读-修改-写：修改输出锁存器的内容。锁存器中的数据通过上三态门进入内部总线，修改后再写入到锁存器中。

读-修改-写指令：并行口为目的操作数的指令：

如：ANL P1, A

## 三、并行口的使用：

1.P0 口：并行双向接口或系统总线 DB0~7/AB0~7。

P1、P2 和 P3 为准双向口。

2.P1 口：称为用户 I/O 接口。对片内 EEPROM 编程时，用作 EEPROM 低 8 位地址信号线。

3.P2 口：双向 I/O 接口或高 8 位地址总线 AB8~15。

对读写片外存储器后，引脚仍恢复输出锁存器的内容。

可用于读写片外数据存储器：

MOV P2, #20H

MOV R0, #00

MOVX A, @R0

4.P3 口：双向并行接口和第二功能：

串行接口引脚：TXD、RXD

中断输入引脚: INT0、INT1

定时器输入引脚: T0、T1

读写控制线: RD、WR

#### 四、接口负载能力:

P0 驱动 8 个 TTL 电路, P1, P2, P3 可驱动 4 个 TTL 电路。

#### 五、应用举例:

例: 用 4 个发光二极管对应显示 4 个开关的开合状态。

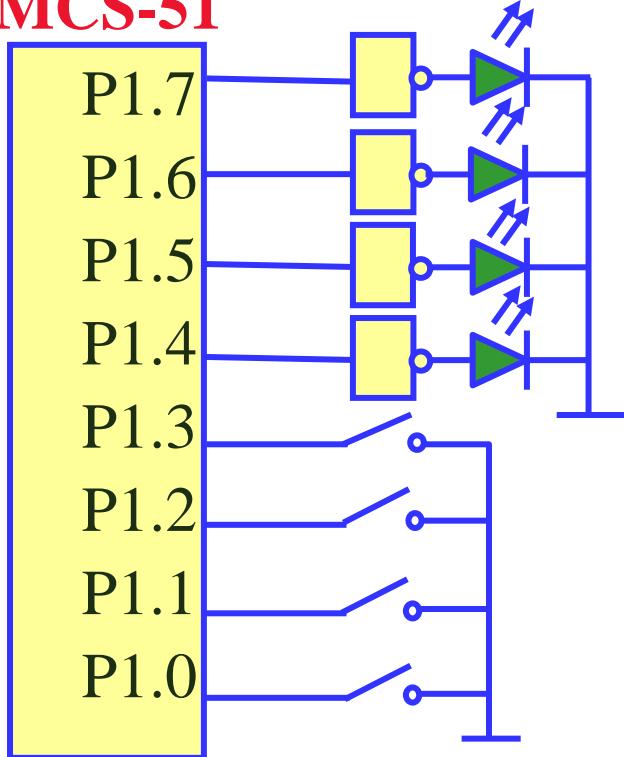
如 P1.0 合则 P1.4 亮, 其余依此类推。

##### 1. 无条件传送方式: (见 P166)

指示灯立即反映开关状态。

```
ORG      0000H
AJMP    MAIN
ORG      0100H
MAIN:    MOV     A, #0FFH
          MOV     P1, A      ; 熄发光二极管
          MOV     A, P1      ; 输入开关状态
          SWAP   A
          MOV     P1, A      ; 开关状态输出
SJMP    MAIN
```

# MCS-51



## 2. 中断传送方式: (见 P167)

先设好开关状态, 然后发出中断请求信号, 改变指示灯亮灭状态。

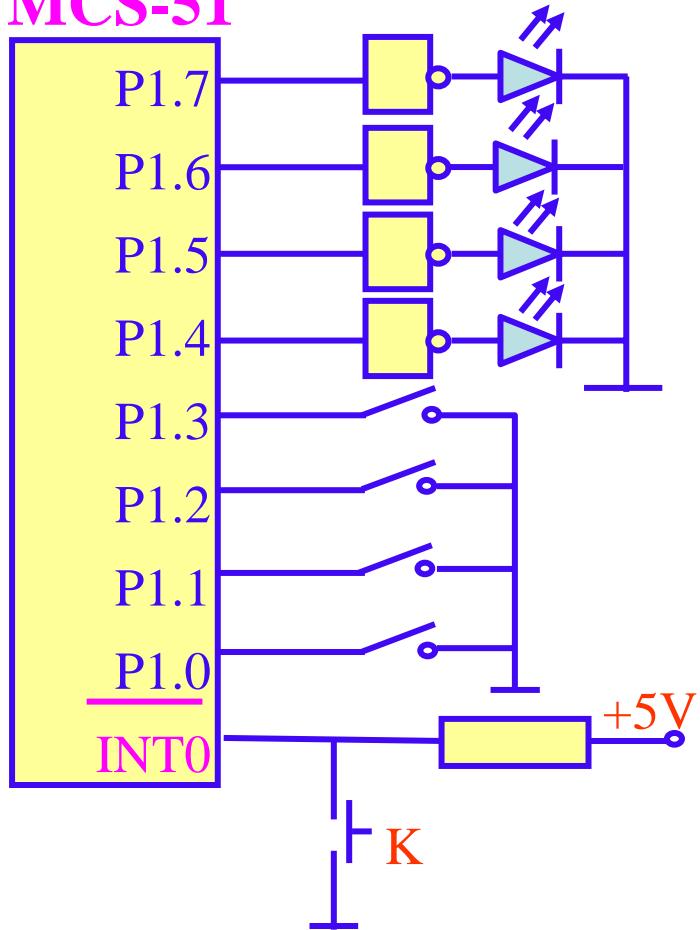
```
ORG      0000H
AJMP    MAIN
ORG      0003H
AJMP    IOINT
ORG      0100H
MAIN:   SETB    IT0      ; 脉冲边沿触发
        SETB    EX0      ; 外部中断 0 允许
        SETB    EA       ; 总中断允许
HERE:   SJMP    HERE     ; 等待中断
ORG      0500H
IOINT:  MOV     A, #0FFH ; 中断程序
```

```

MOV      P1, A      ; 熄发光二极管
MOV      A, P1      ; 输入开关状态
SWAP    A
MOV      P1, A      ; 开关状态输出
RETI

```

## MCS-51



### 7-3 可编程通用并行接口芯片 8255

8255 用于扩展单片机并行 I/O 接口。

#### 7-3-1 结构与引脚

一、结构：

1.3 个 8 位并行 I/O 接口 PA、PB 和 PC:

包含 I/O 数据锁存器，控制寄存器和状态寄存器。

2.2 组控制: A 组: PA 和 PC0~3; B 组: PB 和 PC4~7:

3.3 种工作方式:

方式 0 (基本 I/O 方式): A 口、B 口、C 口均为数据 I/O。输出锁存，输入三态，不用联络信号。

适用于无条件或查询方式的数据传送。

方式 1 (选通 I/O 方式): A 口和 B 口用于数据 I/O，输入/输出均锁存，C 口用于传送联络信号，读 C 口可了解外设当前状态。

适用于查询或中断方式的数据 I/O。

方式 2 (双向数据传送方式): A 口为数据 I/O，B 口只能为方式 0，C 口用作 A 口双向传送的联络信号线。

适用于查询或中断方式的数据 I/O。

二、引脚 (40 脚):

1. 数据线:

D0~7: 传送计算机与 8255 之间的数据、控制字和状态字。

PA0~7 PB 0~7 PC0~7: 传送 8255 与外设之间的数据和联络信息，PC0~7 可用作数据线或联络线。

2. 地址线:

CS: 片选线

A1、A0: 口选线，寻址 PA，PB，PC 数据口和控制口。

3. 读写控制线:

RD、WR 控制计算机与 8255 之间的信息传送和流向。

4. 复位线:

RESET 高电平复位，使内部寄存器全部清零。

三、例题：

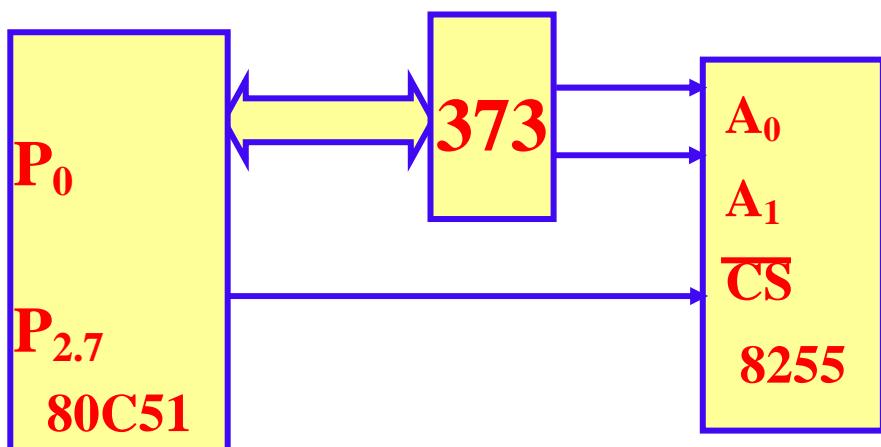
例：求 8255 口地址：

解： A 口 (7F00H);

B 口 (7F01H);

C 口 (7F02H);

控制口 (7F03H)。



### 7-3-2 8255 编程规定

8255 初始化编程：往控制口写入控制字，确定 8255 工作方式。

（见 P184，应用见 P218）

方式选择控制字：D7=1

C 口置位/复位控制字：D7=0

控制字	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
方式	1	A 组方式	PA	PC <sub>4~7</sub>	方式	PB	PC <sub>0~3</sub>	
置/复位	0				位选择		1 / 0	

例：8255PA 口方式 0 输出单片机片内 RAM 数据，PB 口方式 1 输入。

PIOS: MOV DPTR, #7F03H ; 控制口地址

MOV A, #86H ; 写控制字

```

MOVX @DPTR, A           ; 设工作方式
MOV DPTR, #7F00H         ; PA 数据口地址
MOV A, @R0                ; 取 RAM 的数据
MOVX @DPTR, A           ; 由 PA 口输出
...

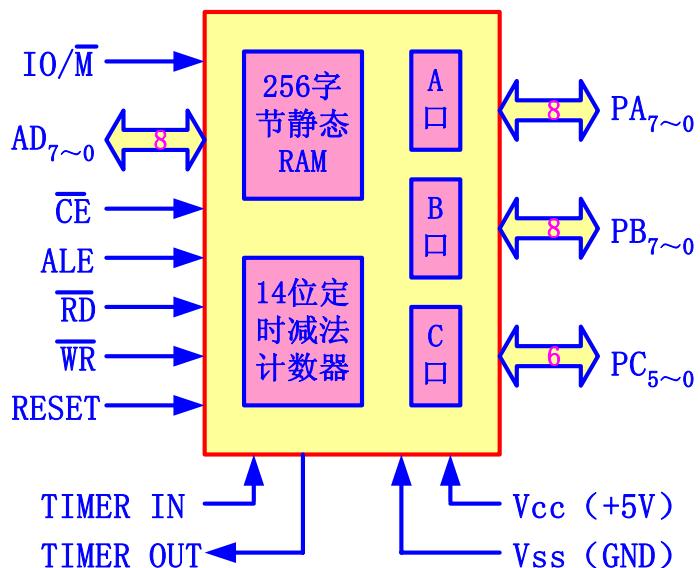
```

#### 7-4 可编程多功能接口芯片 8155

比 8255 功能更强： 3 个并行口 PA0~7、PB0~7、PC0~5；  
 256 字节 RAM； 1 个 14 位减法定时器。

##### 7-4-1 结构和引脚

###### 一、8155 逻辑结构：



###### 二、引脚功能（40 脚）：

###### 1. 数据线：

AD0~7：传送 8155 与计算机之间的数据、控制字和状态字。

PA0~7、PB0~7、PC0~5：传送 8155 与外设之间的信息。

###### 2. 地址线：

CE: 片选信号。

IO/M: 选择口/RAM 单元。(分开编址)

AD0~7: 6 个 I/O 口和 256 字节 RAM 地址。

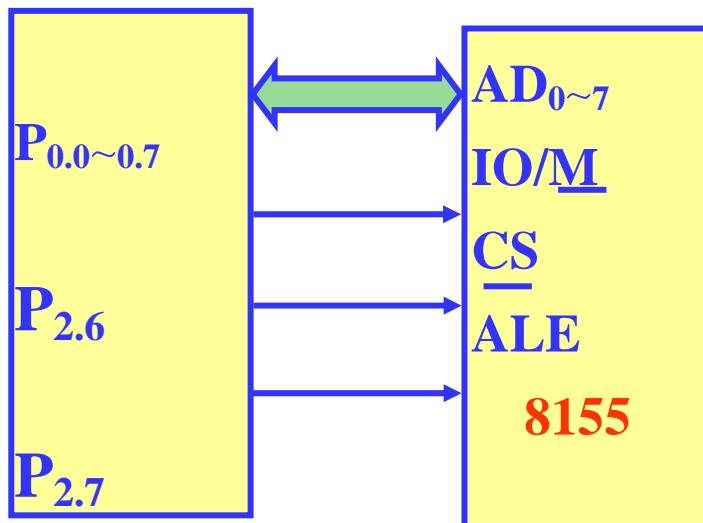
3. 控制线:

ALE: 地址锁存信号。

RD、WR: 读、写控制信号。

4. 定时器输入/输出线: TIMERIN、TIMEROUT (减法记数)。

5. 复位线: RESET。



#### 7-4-2 8155 的工作方式

一、 8155 的工作方式:

3 个 I/O 口: PA、PB、PC

PA、PB: 都为 8 位, 通用数据口, 只 I/O 2 种工作方式;

PC: 6 位, 数据口或控制口, 有 4 种工作方式:

无条件方式传送数据: 输入方式、输出方式;

中断方式传送数据时: PA 口控制端口方式、PA 和 PB 口控制端口方式。

二、 8155 与单片机的连接和编址 (分开编址):

1、只需使用 AD2~AD0 即可实现编址。

8155 共有：256 个 RAM 单元：00H（08H）～FFH；

6 个可编址的端口：命令/状态寄存器：00H；

PA 口 : 01H；

PB 口 : 02H；

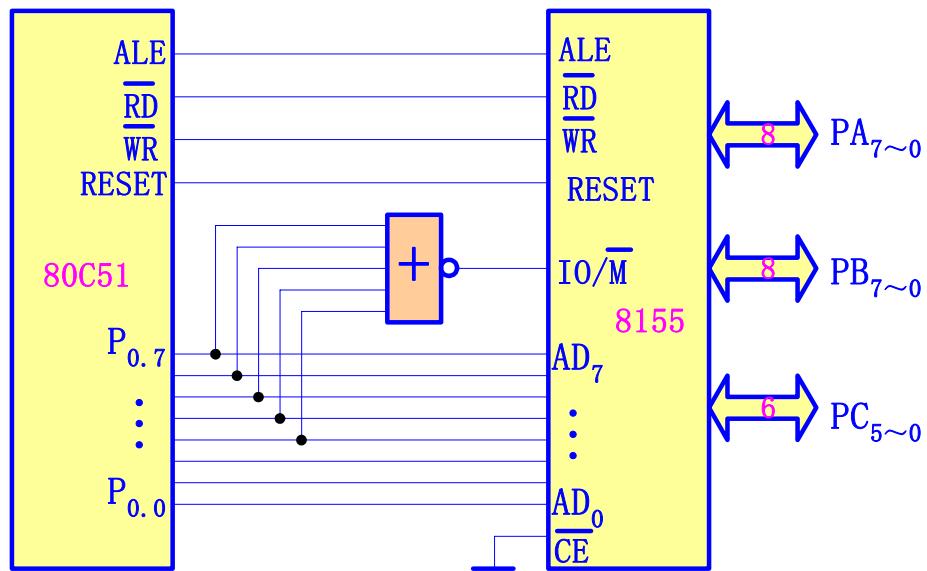
PC 口 : 03H；

定时器/计数器低 8 位：04H；

定时器/计数器高 8 位：05H。

2、连接：

(1) 用或非门产生 IO/M 信号：



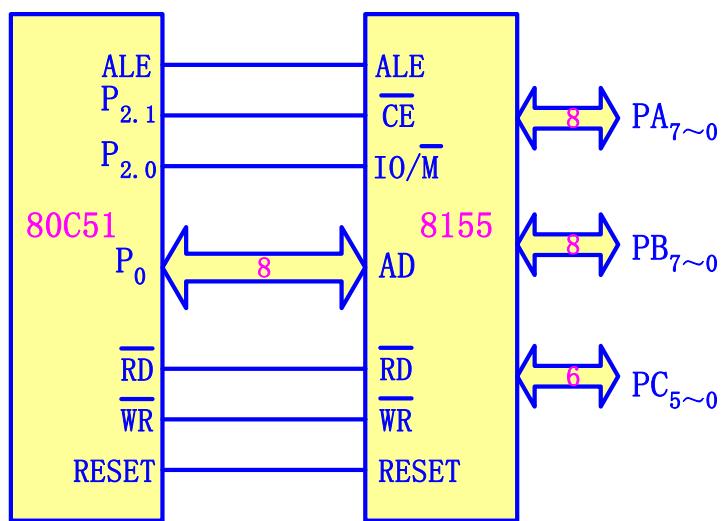
把 P0.7～P0.3 或非后作为 IO/M 信号（损失了 8 个 RAM 单元）。

当 P0.7～P0.3=00000, IO/M=1 时：6 个可编址端口依次为 00H～05H。

当 P0.7～P0.3 为其它, IO/M=0 时, 对应内部 RAM 地址范围为 08H～FFH。

只适用：系统中仅有单片 8155 的情况，可将 CE 接地。

(2) 以高位地址直接作为 IO/M 信号：



将 P2.0 接 IO/M 信号，将 P2.1 接片选信号 CE。

当 IO/M=1 时：可编程端口地址范围依次为 0100H~0105H。

当 IO/M=0 时： 对应内部 RAM 地址范围为 0000H~00FFH。

适用：有多片 I/O 扩展及存储器扩展的较大单片机系统。

### 7-4-3 控制寄存器及使用

## 一、控制寄存器：

1、命令字：只能写不能读；

2、状态字：只读。

3、定时器/计数器结构：14位减法计数器（TH和TL）。



M2M1 两位定义如下：

M2M1=00：单个方波

M2M1=01：连续方波

M2M1=10：单个脉冲

M2M1=11：连续脉冲

二、使用：

1、8155 的定时器/计数器与 MCS-51 单片机内部的定时器/计数器的异同点：

		MCS—51 内部定时器/计数器	8155 的定时器/计数器
不 同 点	记数原则	加法记数	减法记数
	工作方式	4 种：方式 0～方式 3	1 种：14 位记数（4 种输出）
	信号源	定时：由内部提供固定频率的脉冲	定时和记数都由外部 TIMER IN 提供记数脉冲
	工作原理	记数溢出自动置位 TF 位，供用户以 查询（或中断）方式使用	记数溢出时向 TIMER OUT 输出一个脉冲（方波）信号
相 同 点		同样具有定时和记数两种功能	

2、8155 初始化：

例：要求 8155 对计数脉冲进行千分频，即计数 1000 后，TIMER OUT 端电平状态变化，并重新置数以产生连续方波。PA 口为输入方式，PB 口为输出方式，PC 口为输入方式，禁止中断。

解：1、确定计数初值：（P2.1 接片选）

1000D=03E8H，且 M2M1=01，

则 TL=0E8H，其端口地址为 0FD04H；

TH=43H，其端口地址为 0FD05H。

2、确定命令字：

计数器		B口	A口	C口		B口	A口
装入后启动		不允许中断		输入		输出	输入
D7	D6	D5	D4	D3	D2	D1	D0
1	1	0	0	0	0	1	0

因此，命令字为 0C2H。命令/状态寄存器地址为 0FD00H。

初始化程序：

```

MOV      DPTR, #0FD00H ; 命令/状态寄存器地址
MOV      A, #0C2H        ; 命令字
MOVX    @DPTR,A         ; 装入命令字
MOV      DPTR, #0FD04H ; 计数器低 8 位
MOV      A, #0E8H        ; 低 8 位计数值
MOVX    @DPTR, A         ; 写入计数器低 8 位
INC     DPTR             ; 计数器高 8 位地址
MOV      A, #43H          ; 高 8 位计数值
MOVX    @DPTR, A         ; 写入计数值高 8 位

```

## 7-5 8279 可编程键盘/显示器接口芯片

8279 是专用接口芯片 (40 脚)：

自动完成键盘扫描输入和 LED 动态扫描输出；

有自动消抖和多个按键处理功能；

有一个  $16 \times 8$  的显示用 RAM 存储区。

一、信号引脚：

1、数据线 (DB0~DB7)：传送数据、命令和状态；

2、片选线 (CE)：低电平有效；

3、区分信息的特征位 (A0)：A0=0：I/O 为数据；

A0=1: 输入命令, 输出状态;

4、读/写选通线 (RD/WR): 读出状态/写入命令;

5、中断请求信号 (IRQ):

6、键扫描返回输入线 (RL0~RL7):

7、扫描输出线 (SL0~SL3):

8、显示段数据输出线 (OUTA0~3 高位、OUTB0~3 低位):

9、外时钟输入端 (CLK):

内时钟频率=外时钟频率/定时值 (分频系数)

二、8279 的寄存器:

1、命令寄存器: 8 位, 特征位为 D7D6D5。

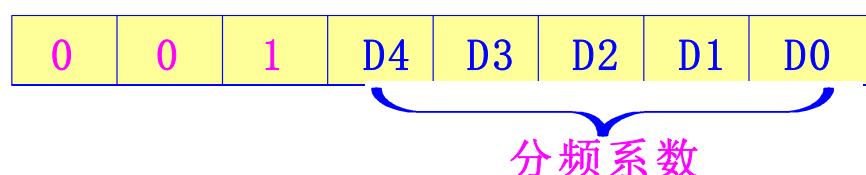
特征位D <sub>7</sub> D <sub>6</sub> D <sub>5</sub>	操作
0 0 0	键盘/显示器设置命令
0 0 1	内部时钟设置命令
0 1 0	读FIFO RAM命令
0 1 1	读显示缓冲命令
1 0 0	写显示数据命令
1 0 1	写入和消隐屏蔽命令
1 1 0	清除命令
1 1 1	结束中断/错误方式设置命令

2、状态寄存器: 8 位

3、数据寄存器: 8 位

内部时钟设置命令:

特征位 D7D6D5=001:



8279 需使用专用的内部时钟信号（或定时信号），其内部时钟信号是由外部输入的时钟信号经分频产生的。

80C51 的 ALE  $\longleftrightarrow$  8279 的 CLK

外部时钟从 CLK 引脚输入，而分频系数由本命令设定，具体数字在 D4~D0 中：

D4 ~D0 为分频系数 2~31。

例：已知 80C51 的  $f_{osc}=12MHz$ ，要求 8279 的内部时钟信号频率为 100kHz，求分频系数 N。

解： $f_{osc}=12MHz$

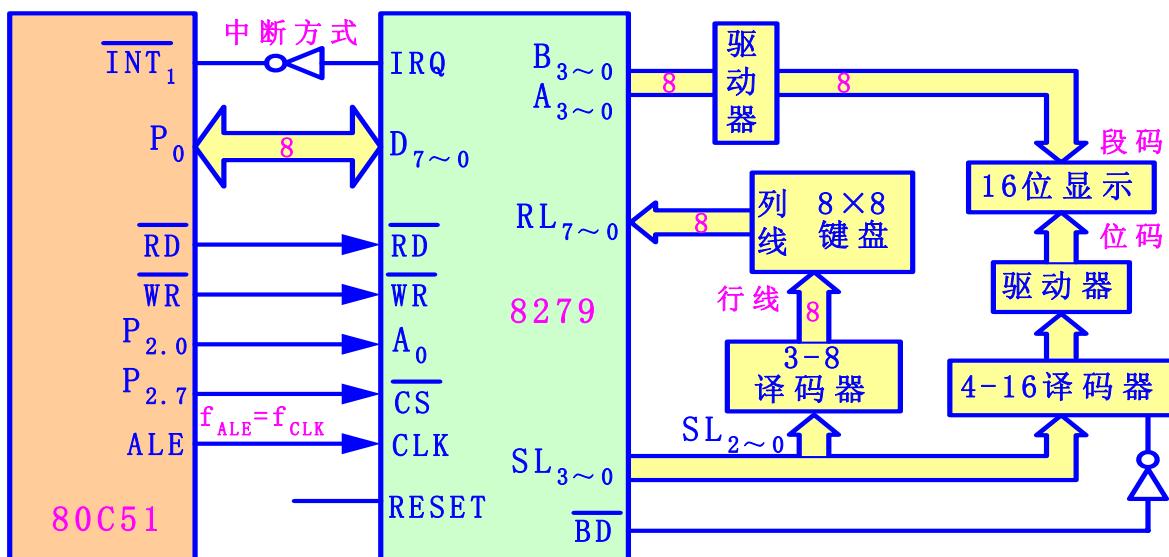
$$f_{ALE}=1/6 f_{osc}=2MHz = f_{CLK}$$

$$N=2MHz/100kHz=20D=10100B$$

因此，时钟编程命令送 00110100B=34H。

三、8279 的接口应用：

1、接口及地址：P198 图 7.30



P2.0=1，对应命令/状态寄存器；

P2.0=0，对应数据寄存器。

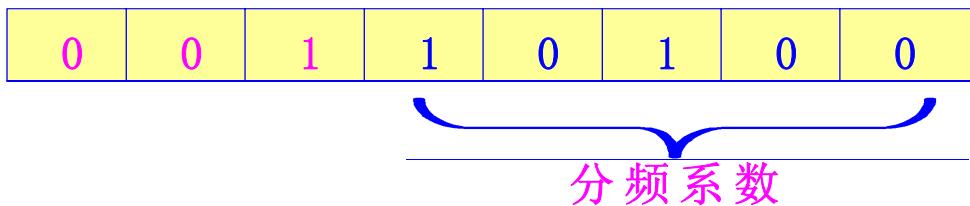
P2.7=0：8279 被选通，则

命令/状态寄存器的地址为 7FFFH(A=1);

数据寄存器的地址为 7EFFH (A=0)。

8279 初始化程序:

```
CSH:  MOV      DPTR, #7FFFH      ; 指向命令/状态口地址
      MOV      A, #0D1H          ; 清除命令 (见 P196)
      MOVX    @DPTR, A          ; 送入命令
      WAIT: MOVX    A, @DPTR      ; 读入命令 (等 8279 清除结束)
      JB      ACC.7, WAIT      ; 读 8279 状态, 为 1, 写入无效
      MOV      A, #00H          ; 键盘/显示器工作方式 (见 P195)
      MOVX    @DPTR, A
      MOV      A, #34H          ; N=34H (见 P196)
      MOVX    @DPTR, A
      MOV      IE, #84H          ; 开中断 (允许中断)
      RET
```



fosc=12MHz

fALE=1/6 fosc=2MHz = fCLK

N=2MHz/100kHz=20D=10100B (=34H)

显示器更新子程序:

```
DIR:      MOV      DPTR, #7FFFH      ; P2.7=0, A0=1
      MOV      A, #90H  ; P196 (5) 写显示 RAM 命令, 从 0 地址开始
      MOVX    @DPTR, A
```

```

MOV R0, #78H           ; 显示缓冲器首地址送 R0
MOV R7, #08H           ; 8 个
MOV DPTR, #7EFFH       ; 数据口地址 (P2.7=0, A0=0)
RD:   MOV A, @R0         ; 取显示数据
      ADD A, #05H         ; 加偏移量
      MOVC A, @A+PC       ; 查表转换为段数据送 8279
      MOVX @DPTR, A
      INC R0
      DJNZ R7, RD
      RET
SEG:  DB 3FH, 06H, 5BH, 4FH, 66H, 6DH, 7DH, 07H,
      ... (共阴极); 根据硬件线路设计字型数据

```

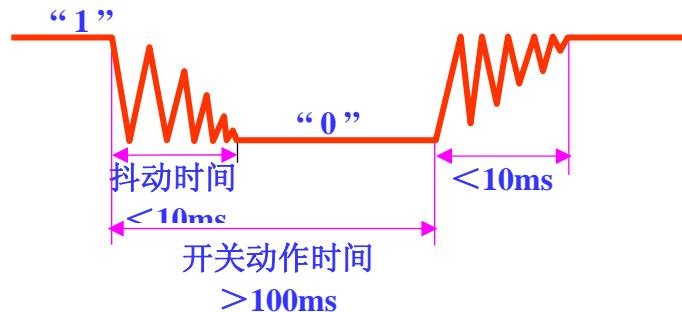
## 7-6 键盘接口技术

一、键盘处理程序任务：

1. 键输入：

检查键盘是否有键被按下，消除按键抖动。确定被按键的键号，获取键号。

硬件电路消除抖动或软件消除抖动。



## 2. 键译码：

键号为键盘位置码，根据键号查表得出被按键的键值。

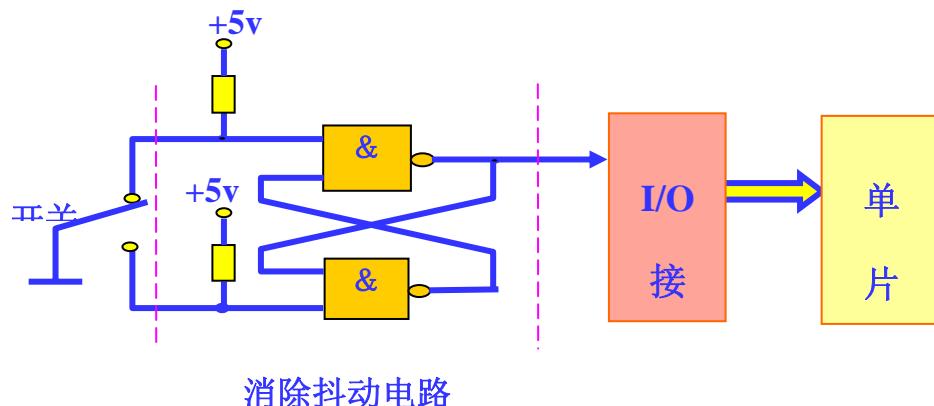
键值：数字键 0~9；

字符键 0AH~0FH；

功能键 10H~

键码=行首键号+列号（键号必须依次是从左至右书写）

或键码=列首键号+行号（键号必须依次是从上至下书写）

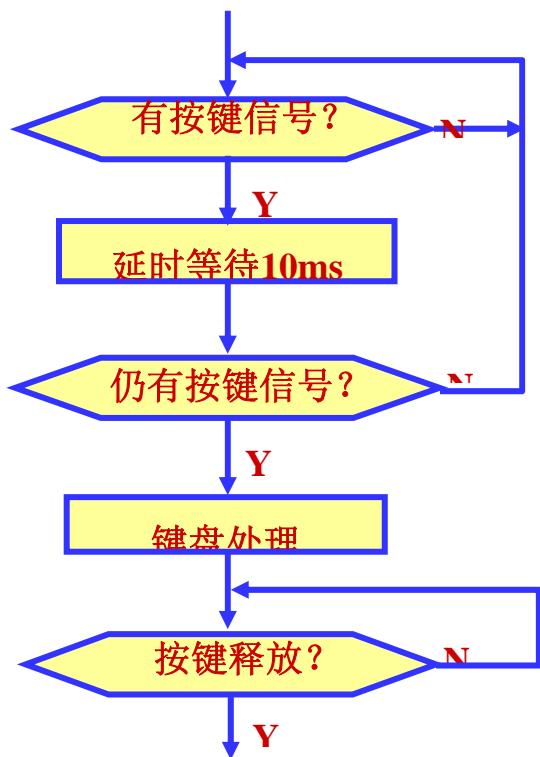


## 3. 键处理：

根据键值转移到不同程序段。

若键值属于数字、字符键，则调用显示数字和字符的子程序。

若键值属于功能键，则进行多分支转移，执行各个功能程序段。



## 二、键盘接口方法:

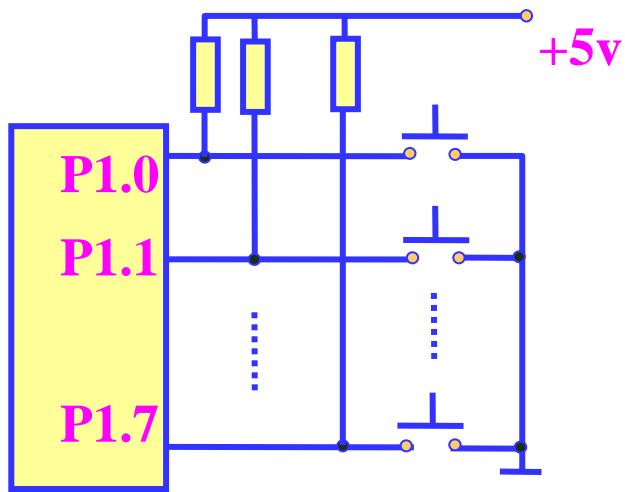
- 1、利用单片机本身的并行口；
- 2、利用单片机本身的串行口；
- 3、利用通用接口芯片 8155、8255 等；
- 4、利用专用接口芯片 8279 等。

## 三、键盘接口的控制方式:

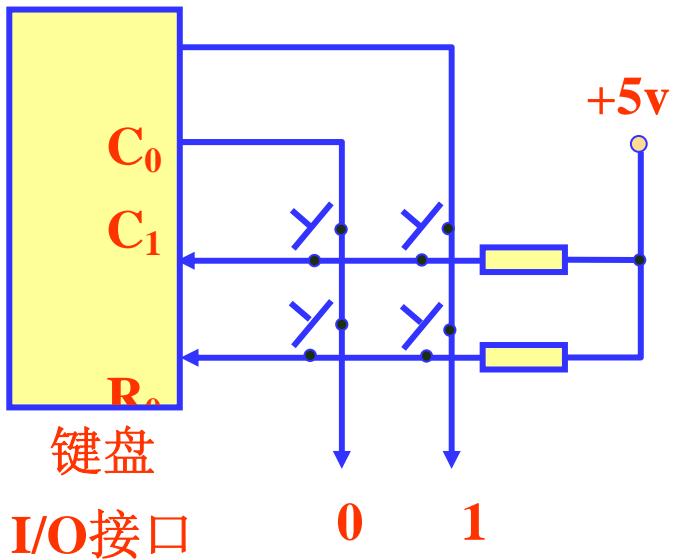
- 1、随机方式：每当 CPU 空闲时执行键盘扫描程序。
- 2、中断方式：键的按下引起中断后，单片机对键盘进行扫描。
- 3、定时方式：单片机定时地对键盘进行扫描。

## 四、键盘接口形式:

- (1) 独立式键盘电路：



(2) 矩阵式键盘:



(1) 独立式键盘电路: 每个按键单独占有一根 I/O 接口引线。

(2) 矩阵式键盘电路:

1) 扫描法:

列线输出, 行线输入。

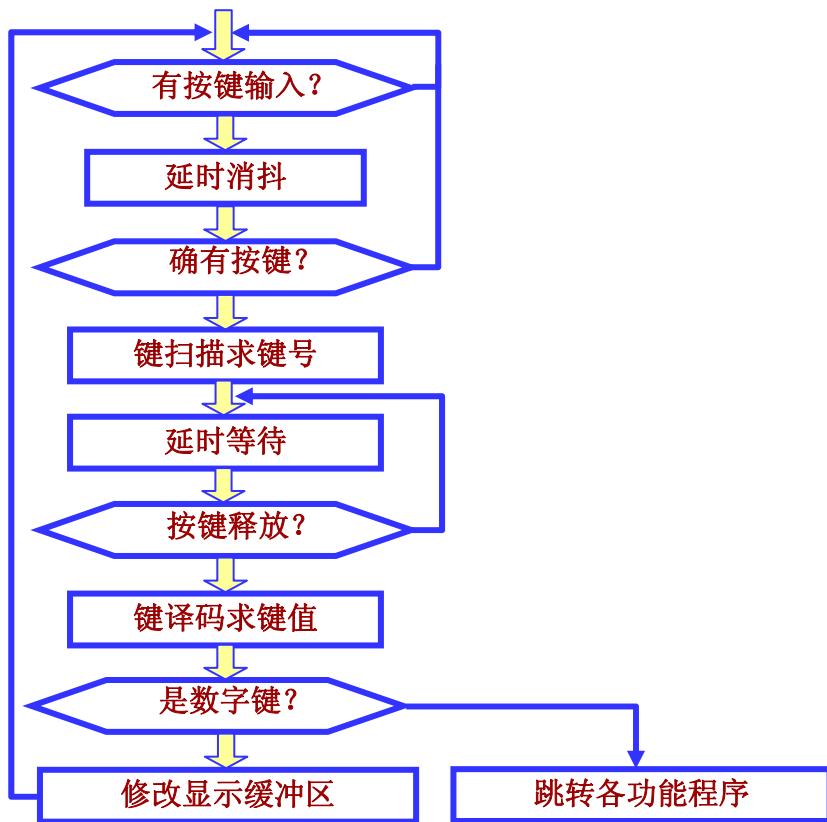
列线逐列输出 0, 某行有按键, 行线输入就为 0;

若无按键, 行线输入全部为 1。

2) 反转法:

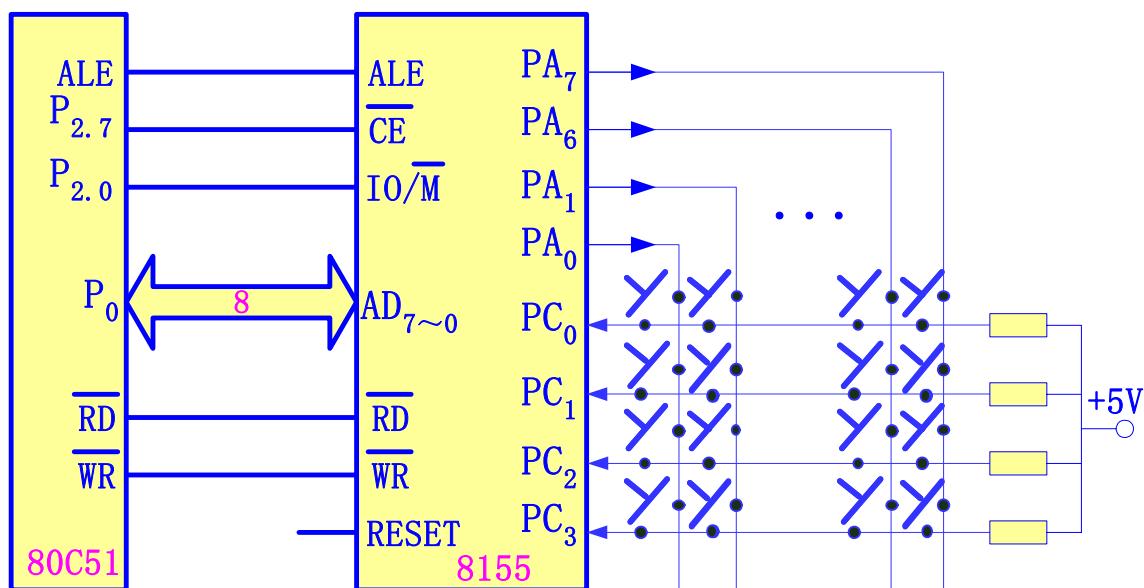
行列线交换输入、输出，两步获取按键键号。

按键处理程序：



五、接口应用：

1、使用 8155 作键盘接口：P204 图 7.35



P2.7=0, P2.0=1:

A 口地址为: 0101H;

C 口地址为: 0103H。

判定有无键闭合的子程序:

KS1: MOV DPTR, #0101H

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, 无键按下则全“0”

ANL A, #0FH ; 屏蔽 A 高半字节 (C 口只 4 位)

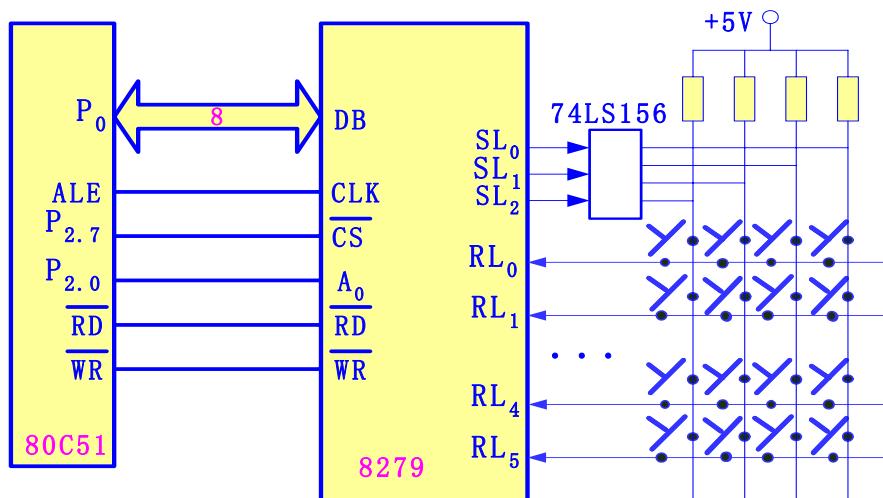
RET

执行 KS1 子程序的结果是: 有闭合键则 (A)  $\neq 0$ ;

无闭合键则 (A) = 0。

CPL A; 负逻辑不直观, 常采取行列线加反相器或软件求反的方法把键盘改成正逻辑。

2、使用 8279 作键盘接口: P207 图 7.36

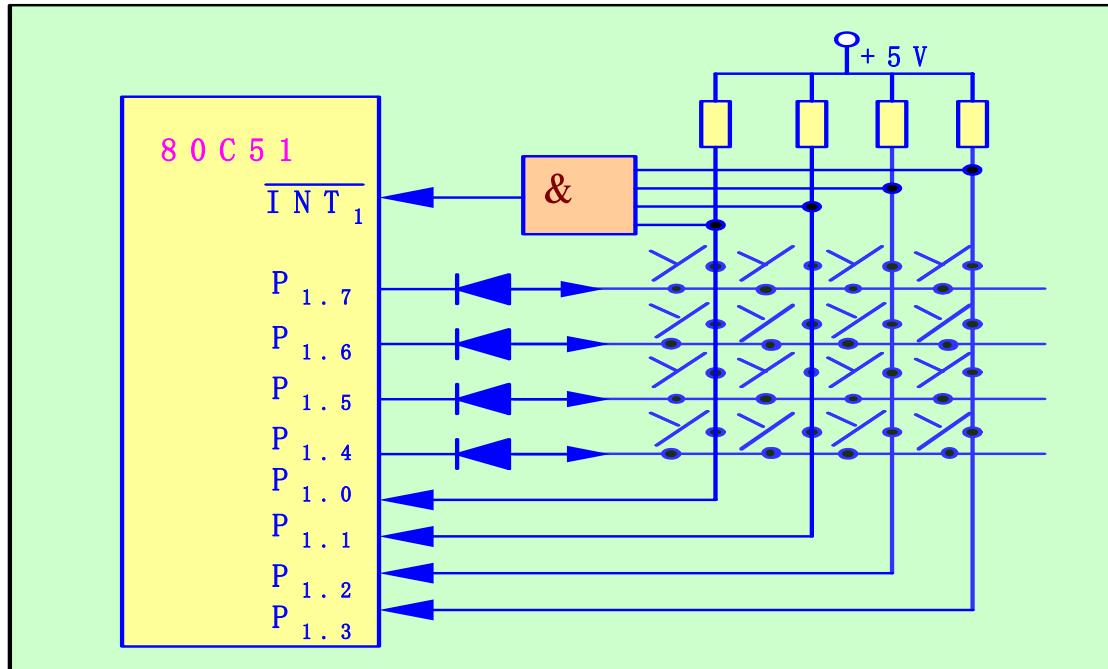


P2.7=0 时：

A0=P2.0=1：命令口地址为：7FFFH；

A0=P2.0=0：数据口地址为：7EFFH。 (程序略)

3、使用中断方式作键盘接口：



当有键按下时，INT1 为低，向 CPU 发出中断申请，在中断服务程序中除完成键识别、键功能处理外，仍须有清除键抖动等功能。

## 7-7 显示器接口技术

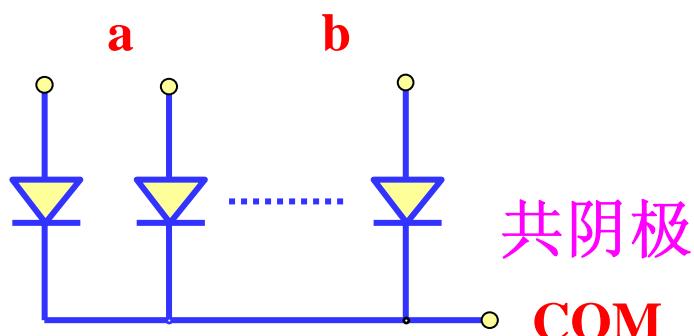
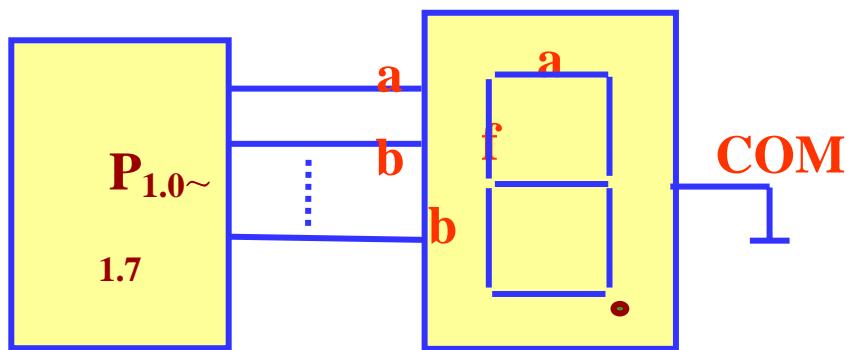
LED 显示器用于显示工业控制参数、过程状态。

一、LED 显示原理：

共阴极 LED 和共阳极 LED。

当 LED 字段引线与数据线连接，每个显示字形对应一个字形码。

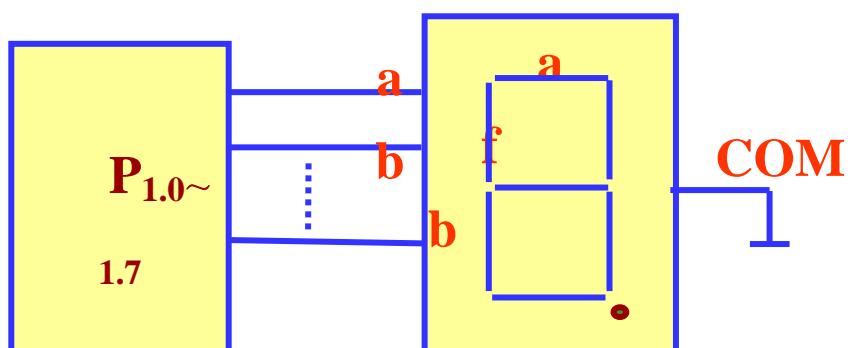
LED (Light Emitting Diode)



代码位	D7	D6	D5	D4	D3	D2	D1	D0
显示段	h	g	f	e	d	c	b	a

## 二、显示程序任务：

- 1、设置显示缓冲区，存放待显示数据和字符（位置码）。
- 2、显示译码：程序存储器中建立字形码常数表，查表得出对应数据和字符的字形码。
- 3、输出显示：输出字形码到显示端口。



例：MOV DPTR, #WTAB ; 指向字形码表首地址

MOV A, @R0 ; 取显示缓冲区中数据

MOVC A, @A+DPTR ; 查表显示译码

```

MOV  P1, A           ; 输出显示
...
WTAB: DB    3FH, 06H, 5BH    ; 字形码表
...

```

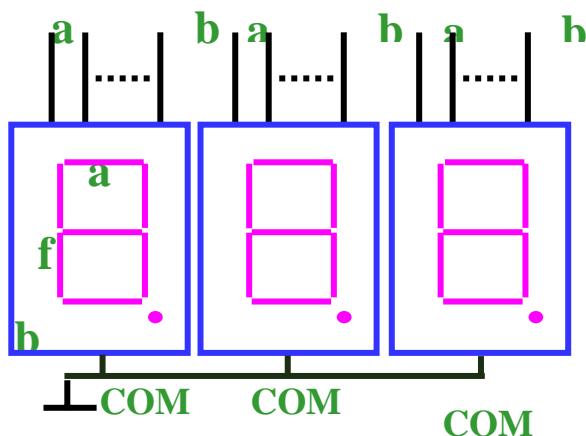
### 三、LED 接口电路：

显示多位数据的两种电路：

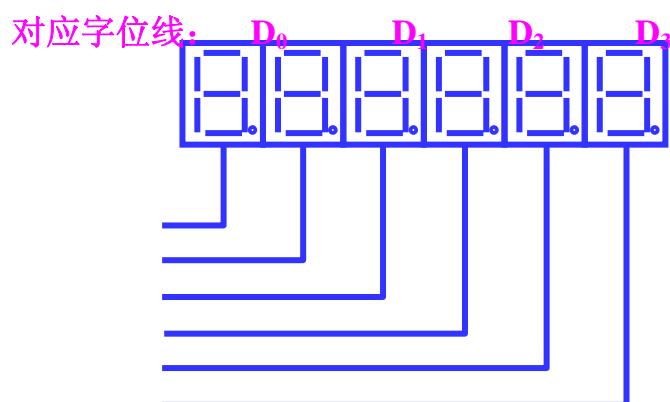
静态显示和动态显示。

#### 1. 静态显示：

多位 LED 共用一个 8 位字段口（共阴极或共阳极），各位 LED 公共端用字位口控制，扫描输出显示不同字形。

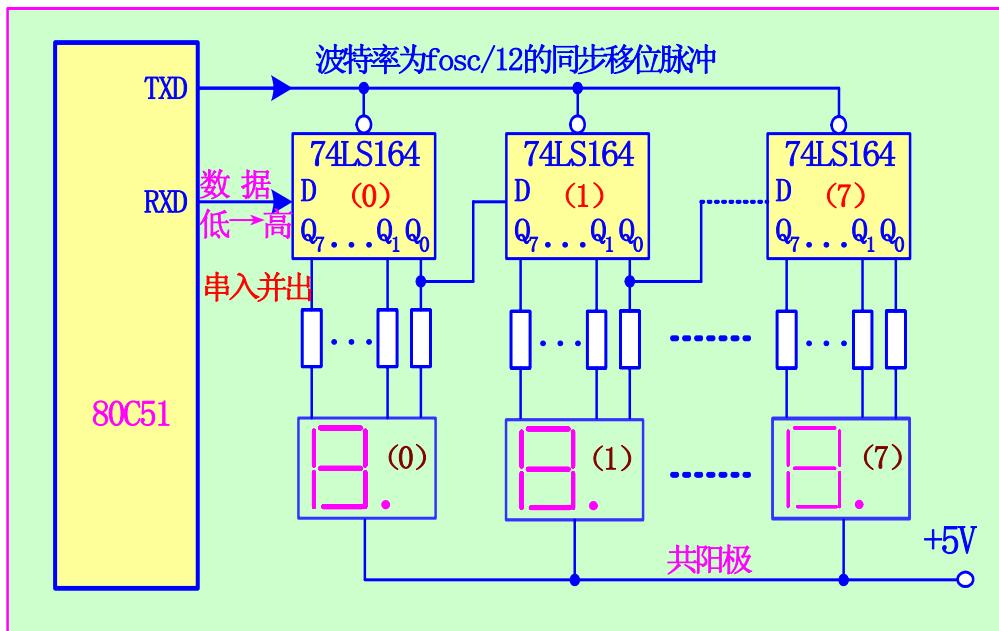


显示缓冲区与多位 LED 对应关系：每个 LED 需要一个 8 位并行口。



特点：显示亮度大，硬件和软件都较简单，应用广泛。

例：显示 80C51 片内 RAM 中以 30H 为首地址的 8 位字形数的程序：



静态显示电路

程序：

```

DIR: PUSH ACC           ; 保护现场
      PUSH DPH
      PUSH DPL
      MOV R2, #08H          ; 显示 8 个数
      MOV R0, #30H          ; 显示缓冲区地址送入 R0
DL0:   MOV A, @R0          ; 取要显示的数作查表偏移量
      MOV DPTR, #TAB        ; 指向字形码表首
      MOVC A, @A+DPTR      ; 查表得字形码
      MOV SBUF, A           ; 发送显示
DL1:   JNB TI, DL1        ; 等待发送完一桢数据
      CLR TI                ; 清标志，准备继续发送
      INC R0                ; 更新显示单元

```

```

DJNZ R2, DL0      ; 重复显示所有数码管

POP DPL          ; 恢复现场

POP DPH

POP ACC

RET

TAB    DB 0C0H, 0F9H, 0A4H, 0B0H, 99H ; 0, 1, 2, 3, 4
      DB 92H, 82H, 0F8H, 80H, 90H, 88H ; 5, 6, 7, 8, 9, A
      DB 83H, 0C6H, 0A1H, 86H, 8EH    ; B, C, D, E, F

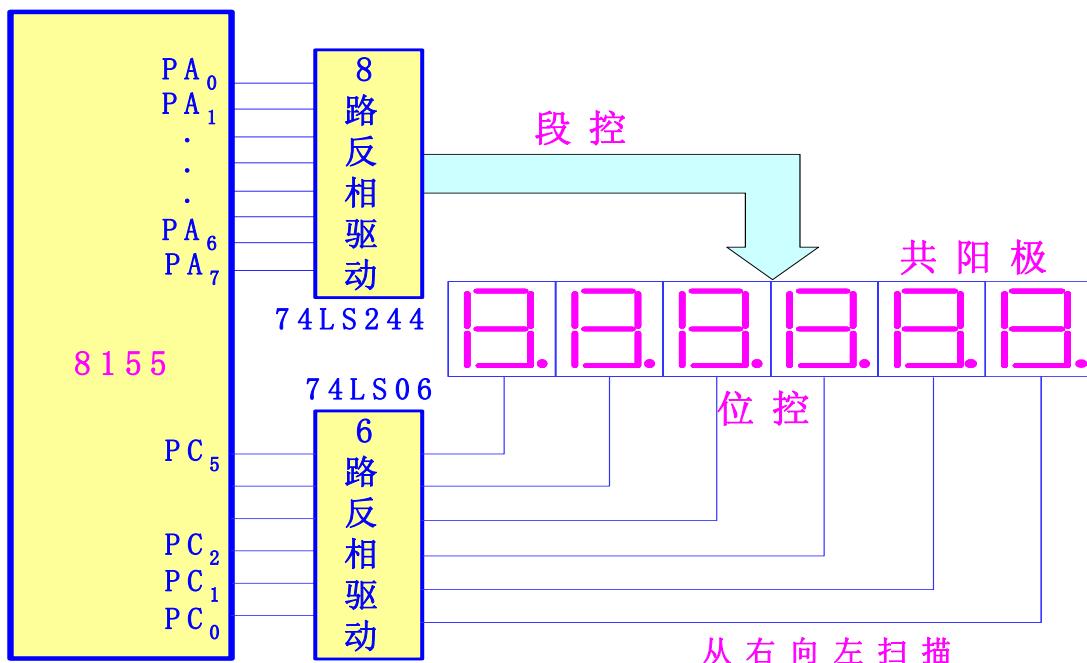
```

## 2. 动态显示:

分时轮流选通数码管的公共端，使得各数码管轮流导通，在选通相应 LED 后，即在显示字段上得到显示字形码。

特点：提高数码管的发光效率，可简化硬件线路。

## 3、用 8155 作 LED 显示器接口：(请见 P211)



可假定地址：段控：0101H；位控：0103H。

在内部 RAM 中设置显示缓冲区，其单元个数与 LED 显示位数相同。

设 6 个显示器的缓冲单元是 7AH~7FH。

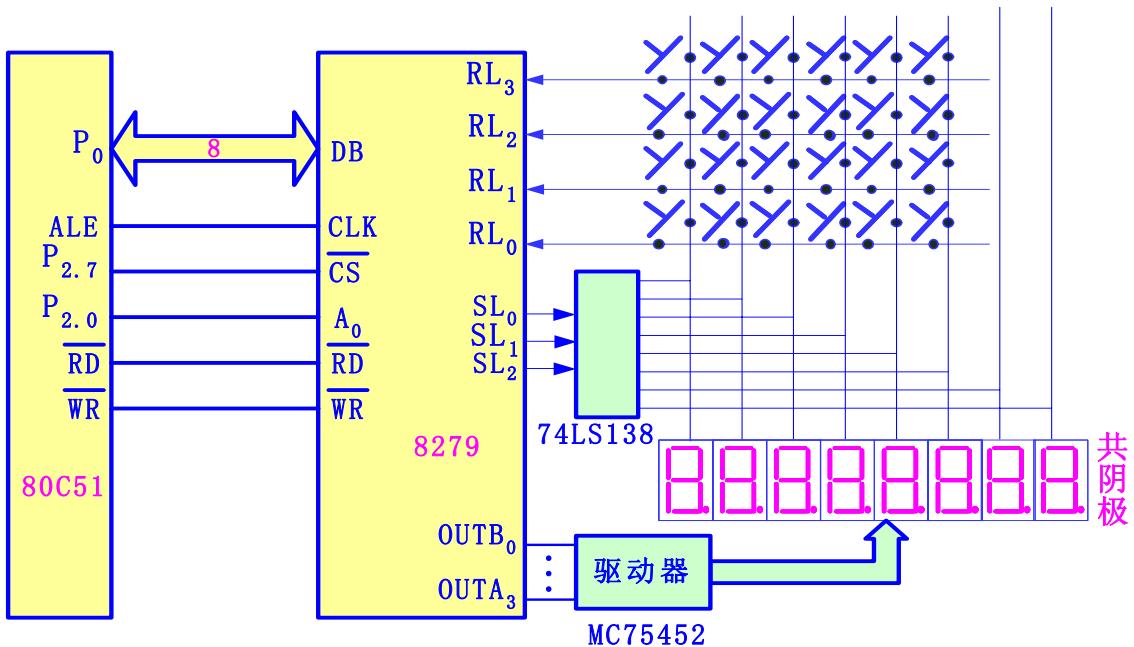
LED 显示程序：

```
DIS:  MOV  R0, #7AH          ; 指向显示缓冲区起始单元
      MOV  R3, #01H          ; 从右数第一位显示器开始
      MOV  A, R3            ; 取位控码初值

DLP:  MOV  DPTR, #0103H      ; 指向字位口 (PA 口)
      MOVX @DPTR, A          ; 输出字位码, 显示其中 1 位
      MOV  DPTR, #0101H      ; 段控码地址
      MOV  A, @R0            ; 取一个显示数据
      ADD  A, #0BH           ; 查表偏移量
      MOVC A, @A+PC          ; 取出字形码
      MOVX @DPTR, A          ; 输出字形码
      ACALL DLY1MS          ; 延时 1ms
      INC  R0               ; 指向显缓区下一单元
      MOV  A, R3            ; 修改字位码
      RL   A                ; 显示下一位
      MOV  R3, A
      JNB ACC.5, DLP         ; 未显示到最左边 LED, 继续显示
      RET                   ; 全部扫描一遍, 结束

DTAB: DB  0C0H, 0F9H, 0A4H      ; 字形代码表 (共阳极)
      DB  0B0H, 99H,
      DLY1MS: ...           ; 延时 1ms 子程序
```

4、用 8279 作 LED 显示器接口：



P2.7=0 时：

A0=P2.0=1：命令口地址为：7FFFH；

A0=P2.0=0：数据口地址为：7EFFH。 (程序略)

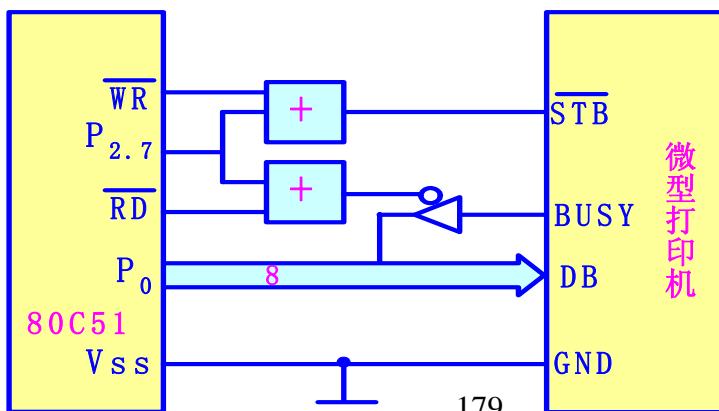
#### 四、LCD 显示：

LCD (Liquid Crystal Display)：

它是一种被动式的显示器，即液晶本身并不发光，而是利用液晶经过处理后能改变光线通过方向的特性，而达到白底黑字或黑底白字显示的目的。

#### 7-8 打印机接口技术

1、不用扩展接口的打印机连接：P217 图 7.44

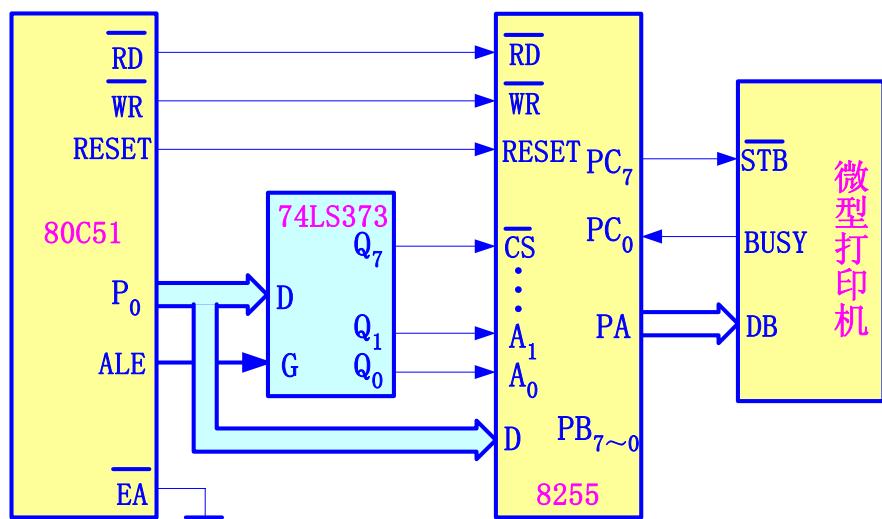


在打印机中只有一个数据寄存器，用于寄存打印数据。

P2.7 选通 80C51 的 RD 和 WR: (只能使用查询方式)

数据口的地址为: 7FFFH。

2、使用 8255 作打印机接口: P218 图 7.45



地址:  $CS \leftrightarrow Q_7$  ( $P_0.7 = 0$ ): (可以是查询方式)

A1A0= (Q1Q0)	00:	A 口: 7CH
	01:	B 口: 7DH
	10:	C 口: 7EH
	11:	控制口: 7FH

3、打印驱动程序: (查询方式)

```
MOV      R0, #7FH ; 控制寄存器地址 (P0.7 = 0)  
MOV      A, #88H ; 工作方式控制字  
MOVX    @R0, A ; 写入工作方式控制字  
TP:    MOV      R0, #7EH ; C 口地址  
TP1:   MOVX    A, @R0 ; 读 C 口  
JB      ACC.7, TP1 ; BUSY=1 (忙), 继续查询
```

```
MOV      R0, #7CH      ; A 口地址
MOV      A, @R1        ; 取缓冲区数据
MOVX    @R0, A        ; 打印数据送 8255
INC      R1          ; 指向下一单元
MOV      R0, #7FH      ; 控制口地址
MOV      A, #00H      ; 输出 STB 脉冲
MOVX    @R0, A
MOV      A, #01H
MOVX    @R0, A
DJNZ    R2, TP        ; 数据长度减 1, 不为 0 继续
RET
```

## 小结

- 1、接口的功能（链接）、编址方式和控制方式（无条件、查询、中断和 DMA）。
- 2、单片机本身 4 个并行 I/O 口的使用。
- 3、通用并行接口芯片 8255 的结构、工作方式、编址和应用。
- 4、通用并行接口芯片 8155 的内部结构、功能、工作方式、编址和应用，以及与 8255 的异同。
- 5、键盘/显示器专用接口芯片 8279 的结构、内部时钟的设定、编址和应用。
- 6、键盘、显示器和打印机的接口原理和实现方法。

## 第8章 串行数据通信

### 一、教学要求：

掌握：串行口结构与工作原理。

理解：串行通信方式。

了解：波特率设计，串行口应用。

### 二、教学内容：

8.1 串行通讯基础知识

8.2 单片机的串行口及控制寄存器

8.3 单片机串行通信工作方式

### 三、教学重点：串行口结构与工作原理。

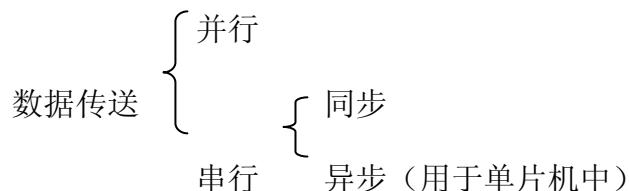
### 四、教学难点：波特率设计与串行口应用。

### 五、建议学时：3学时。

### 六、教学内容：

8-1 串行通信基础知识

#### 一、串行通信基本原理：



	并 行 数 �据 传 送	串 行 数 据 传 送
原 理	各数据位同时传送	数据位按位顺序进行
优 点	传送速度快、效率高	最少只需一根传输线即可完成：成本低
缺 点	数据位数→传输线根数：成本高	速度慢
应 用	传送距离<30米，用于计算机内部	几米~几千公里，用于计算机与外设之间

## 二、串行通信的基本方式:

### (一)异步通信:

以字符为传送单位用起始位和停止位标识每个字符的开始和结束字符，间隔不固定，只需字符传送时同步即可。

异步通讯常用格式：一个字符帧



异步通信的双方需要两项约定：

#### 1.字符格式:

一帧字符位数的规定：数据位，校验位，起始位和停止位。

#### 2.波特率(位/秒)和传送速率的规定：

例：要求每秒传送 120 个字符，每帧为 10 位。

解：  $B=120 \times 10=1200$  波特 每位 0.83ms

数据位传送速率=  $120 \times 8=960$  位/秒

### (二)同步通信:

以一串字符为一个传送单位，字符间不加标识位，在一串字符开始用同步字符标识，硬件要求高，通讯双方须严格同步。

## 三、串行接口功能:

(1) 发送器：并→串数据格式转换，添加标识位和校验位，一帧发送结束，设置结束标志，申请中断。

(2) 接收器：串→并数据格式转换，检查错误，去掉标识位，保存有效数据，设置接收结束标志，申请中断。

(3) 控制器：接收编程命令和控制参数，设置工作方式：同步/异步、字符格式、波特率、校验方式、数据位与同步时钟比例等。

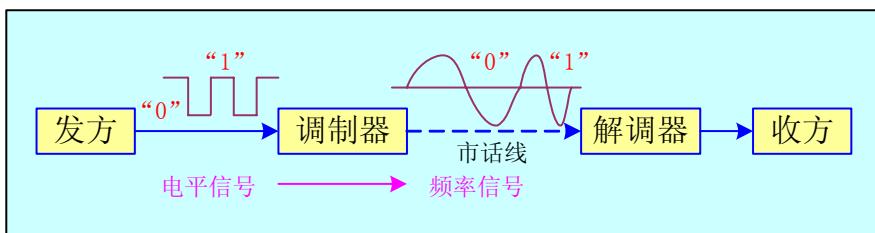
## 四、串行数据传送方向:

- 1、单工通讯：数据单向传送。（1条数据线，单向）
- 2、半双工通讯：数据可分时双向传送。（2条数据线，双向）
- 3、全双工通讯：可同时进行发送和接收。（1条或2条数据线，双向）

五、异步串行通信的信号形式：

1、远距离直接传输数字信号，信号会发生畸变，因此要把数字信号转变为模拟信号再进行传送。可利用光缆、专用通信电缆或电话线。

方法：通常使用频率调制法（频带传送方式）。



通常：“1”：1270Hz 或 2225Hz；

“0”：1070Hz 或 2025Hz。

2、因通信时（有干扰）信号要衰减，所以常采用 RS232 电平负逻辑，拉开“0”和“1”的电压档次，以免信息出错：

**TTL正逻辑：**

“0”： **0**  
—2.4V;

**RS232负逻辑：**

“0”： +3V—+25V;  
“1”： -3V—-25V。

## 8-2 串行口及控制寄存器

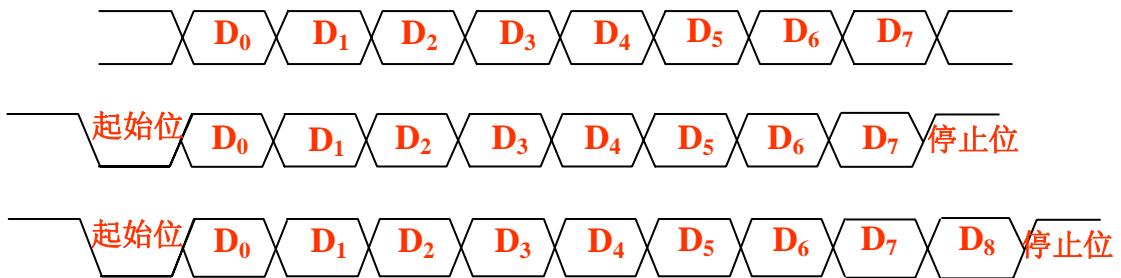
### 8-2-1 MCS-51 串行接口

1 个全双工串行接口，可同时进行发送和接收。

串行接口输入/输出引脚：TXD(P3.1)、RXD(P3.0)

数据格式：按不同方式，一帧数据位数 8/10/11

发送/接收时，数据皆低位在前。



一帧字符发送/接收结束，置位标志位(TI/RI)，并申请串行中断。

中断控制：中断允许位 ES、总允许 EA；

中断入口：0023H。

一、串行接口控制：

1.数据缓冲器 SBUF：

发送 SBUF 和接收 SBUF 共用一个地址 99H。

1) 发送 SBUF 存放待发送的 8 位数据，写入 SBUF 将同时启动发送。发送指令：

MOV SBUF, A

2) 接收 SBUF 存放已接收成功的 8 位数据，供 CPU 读取。读取串行口接收数据指令：

MOV A, SBUF

2.节电控制寄存器 PCON：

SMOD (PCON.7)：波特率加倍控制位。

SMOD=1，波特率加倍；

SMOD=0，则不加倍。

3.串行口控制/状态寄存器 SCON(98H)：

<b>SM0</b>	<b>SM1</b>	<b>SM2</b>	<b>REN</b>	<b>TB8</b>	<b>RB8</b>	<b>TI</b>	<b>RI</b>
------------	------------	------------	------------	------------	------------	-----------	-----------

**SM0、SM1：**选择串行口 4 种工作方式。

**SM2：**多机控制位，用于多机通讯。

**REN：**允许接收控制位， $REN=1$ ，允许接收；

$REN=0$ ，禁止接收。

**TB8：**发送的第 9 位数据位，可用作校验位和地址/数据标识位。

**RB8：**接收的第 9 位数据位或停止位。

**TI：**发送中断标志，发送一帧结束， $TI=1$ ，必须软件清零；

**RI：**接收中断标志，接收一帧结束， $RI=1$ ，必须软件清零。

### 8-3 串行通信工作方式

**SM0、SM1** 选择四种工作方式。

(1) 方式 0：同步移位寄存器方式

用于扩展并行 I/O 接口。

1. 一帧 8 位，无起始位和停止位。

2. **RXD：**数据输入/输出端。

**TXD：**同步脉冲输出端，每个脉冲对应一个数据位。

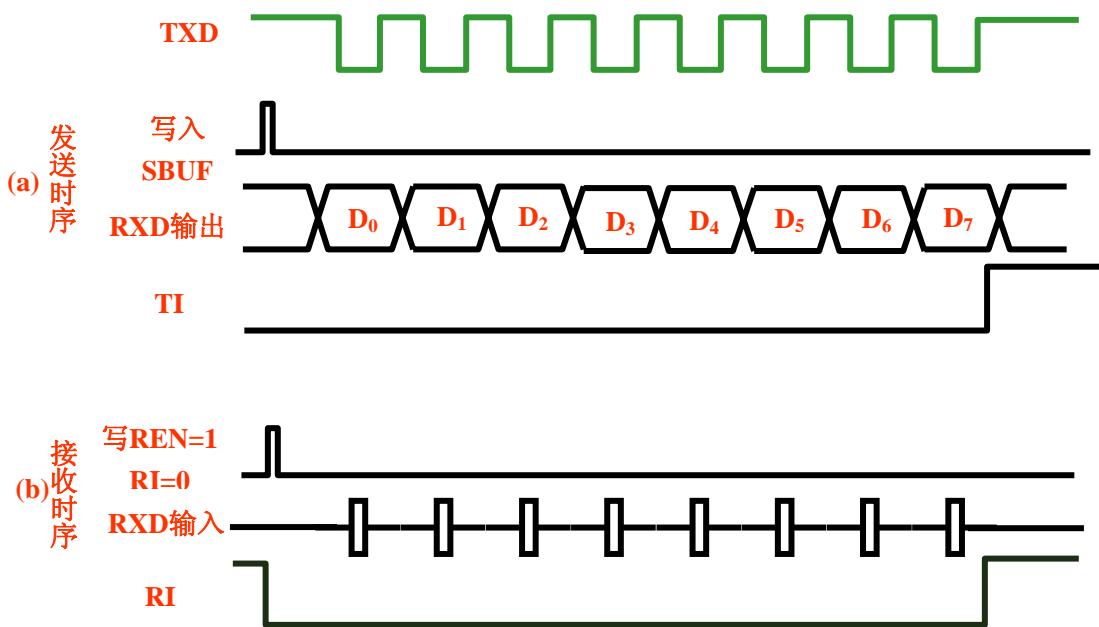
3. 波特率  $B = fosc/12$

如： $fosc=12MHz$ ,  $B=1MHz$ , 每位数据占  $1\mu s$ 。

4. 发送过程：写入 **SBUF**，启动发送，一帧发送结束， $TI=1$ 。

接收过程： $REN=1$  且  $RI=0$ ，启动接收，一帧接收完毕， $RI=1$ 。

时序图：



## (2) 方式 1: 8 位数据异步通讯方式

1. 一帧 10 位: 8 位数据位, 1 个起始位(0), 1 个停止位(1)。
2. RXD: 接收数据端。

TXD: 发送数据端。

3. 波特率: 用 T1 作为波特率发生器,  $B=(2SMOD/32) \times T1$  溢出率。

4. 发送: 写入 SBUF, 同时启动发送, 一帧发送结束, TI=1。

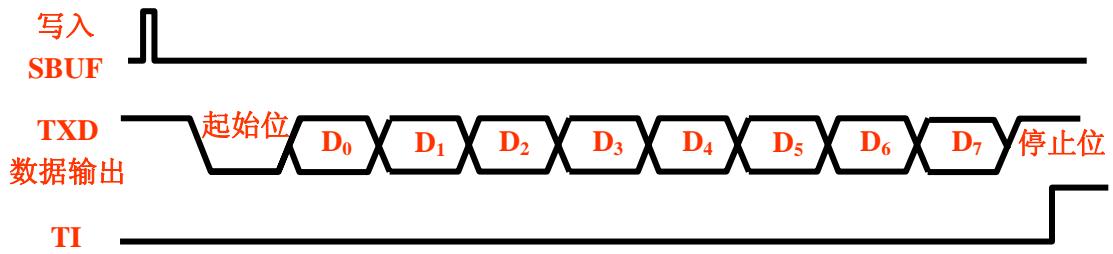
接收: REN=1, 允许接收。

接收完一帧, 若 RI=0 且停止位为 1 (或 SM2=0), 将接收数据装入 SBUF, 停止位装入 RB8, 并使 RI=1; 否则丢弃接收数据, 不置位 RI。

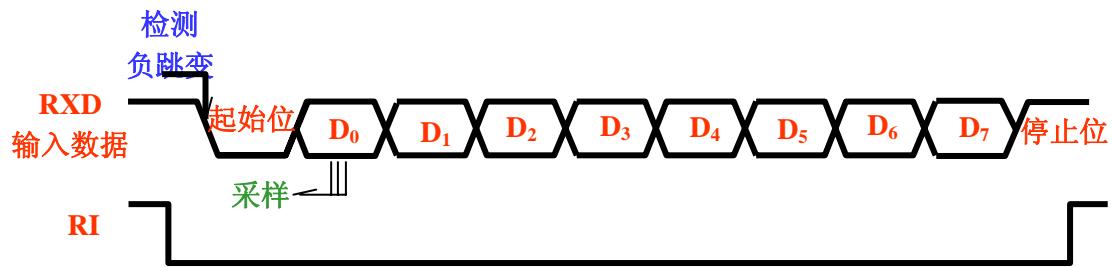
当 REN=1, CPU 开始采样 RXD 引脚负跳变信号, 若出现负跳变, 才进入数据接收状态, 先检测起始位, 若第一位为 0, 继续接收其余位; 否则, 停止接收, 重新采样负跳变。

数据采样速率为波特率 16 倍频, 在数据位中间, 用第 7、8、9 个脉冲采样 3 次数据位, 并 3 中取 2 保留采样值。

时序图：



(a) 发送时序图



(b) 接收时序图

### (3) 方式 2 和方式 3： 9 位数据异步通讯方式

1. 一帧为 11 位：9 位数据位，1 个起始位(0)，1 个停止位(1)。第 9 位数据位在 TB8/RB8 中，常用作校验位和多机通讯标识位。

2. RXD：接收数据端，

TXD：发送数据端。

3. 波特率： 方式 2：  $B=(2SMOD/64) \times fosc$  。

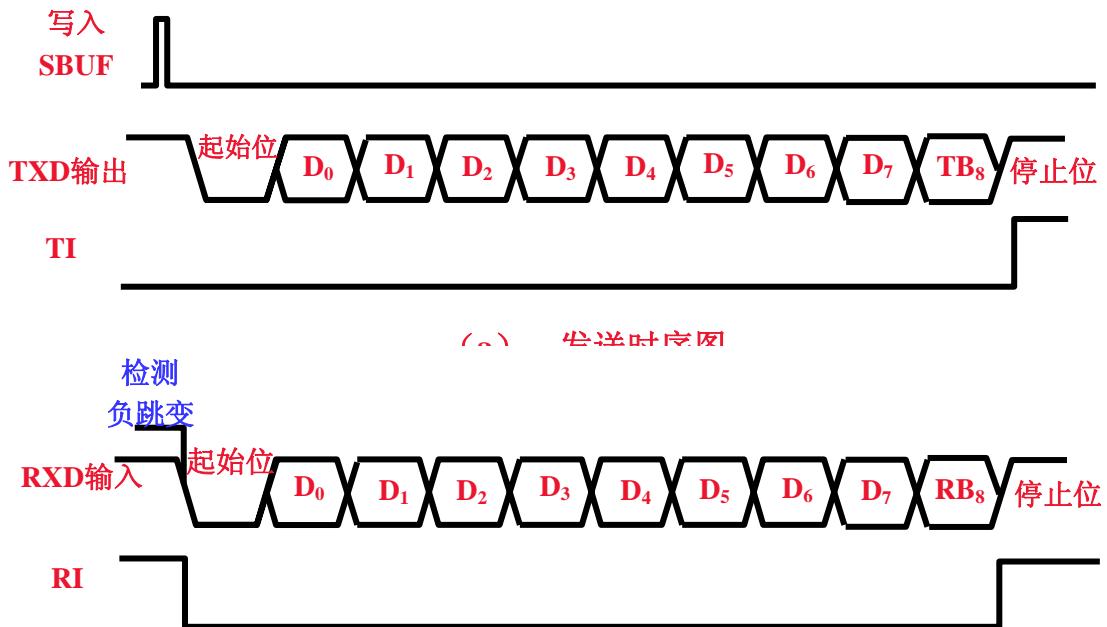
方式 3：  $B=(2SMOD/32) \times T1$  溢出率 。

4. 发送：先装入 TB8，写入 SBUF 并启动发送，发送结束，TI=1。

接收：REN=1，允许接收。

接收完一帧，若 RI=0 且第 9 位为 1 (或 SM2=0)，将接收数据装入接收 SBUF，第 9 位装入 RB8，使 RI=1；否则丢弃接收数据，不置位 RI。

时序图：



(b) 接收时序图

(4) 计算波特率：

方式 0 为固定波特率：  $B = fosc/12$

方式 2 可选两种波特率：  $B = (2SMOD/64) \times fosc$

方式 1、3 为可变波特率，用 T1 作波特率发生器。

$$B = (2SMOD/32) \times T1 \text{ 溢出率}$$

T1 为方式 2 的时间常数：  $X = 28 - t/T$  （请见 P153）

溢出时间：  $t = (28 - X)T = (28 - X) \times 12/fosc$

T1 溢出率 =  $1/t = fosc/[12 \times (28 - X)]$

$$\therefore \text{波特率 } B = (2SMOD/32) \times fosc/[12 \times (28 - X)]$$

串行口方式 1、3，根据波特率选择 T1 工作方式，计算时间常数。

T1 选方式 2：  $TH1 = X = 28 - fosc/12 \times 2SMOD/(32 \times B)$

T1 选方式 1 用于低波特率，需考虑 T1 重装时间常数时间。

也可选工作方式 3（请见 P156）。

4 种方式比较:

方 式	波 特 率	传 送 位 数	发 送 端	接 收 端	用 途
0	$1/12 \text{ fosc}$ (固定不变)	8 (数据)	RXD	RXD	接移位寄存器, 扩充并口
1	$2^{\text{SMOD}}/32 T_1$ 溢出率	10 (起始位、8位数据位、停止位)	TXD	RXD	单机通讯
2	$2^{\text{SMOD}}/64 \text{ fosc}$	11 (第9位为1: 地址; 为0:)	TXD	RXD	多机通讯
3	$2^{\text{SMOD}}/32 T_1$ 溢出率	11位 (同方式2)	TXD	RXD	多机通讯

## 8-4 串行口的应用

串行口初始化编程格式:

```

SIO:  MOV  SCON, #控制状态字      ; 写方式字且 TI=RI=0
      ( MOV  PCON, #80H )          ; 波特率加倍
      ( MOV  TMOD, #20H )          ; T1 作波特率发生器
      ( MOV  TH1, #X )              ; 选定波特率
      ( MOV  TL1, #X )
      ( SETB  TR1)
      ( SETB  EA)                  ; 开串行口中断
      ( SETB  ES)

```

发送程序:

先发送一个字符, 等待 TI=1 后再发送下一个字符。

1、查询方式:

```

TRAM:  MOV  A, @R0    ; 取数据
        MOV  SBUF, A    ; 发送一个字符
WAIT:  JBCTI, NEXT    ; 等待发送结束

```

```
SJMP  WAIT

NEXT:  CLR  TI
      INC  R0          ; 准备下一次发送

SJMP  TRAM
```

2、中断方式：

```
ORG  0023H          ; 串行口中断入口

AJMP SINT

MAIN:  ...          ; 初始化编程

TRAM:  MOV  A, @ R0    ; 取数据
      MOV  SBUF, A      ; 发送第一个字符

H:    SJMP  H          ; 其它工作

SINT: CLR  TI          ; 中断服务程序
      INC  R0
      MOV  A, @ R0      ; 取数据
      MOV  SBUF, A      ; 发送下一个字符

      RETI
```

接收程序：

REN=1、RI=0 等待接收，当 RI=1，从 SBUF 读取数据。

1.查询方式：

```
WAIT: JBC  RI, NEXT    ; 查询等待

SJMP  WAIT

NEXT: MOV  A, SBUF      ; 读取接收数据
      MOV  @R0, A       ; 保存数据

      CLR  RI
```

```
INC      R0          ; 准备下一次接收  
SJMP    WAIT
```

## 2. 中断方式：

ORG 0023H

AJMP RINT

MAIN: ... ; 初始化编程

H: SJMP H ; 其它任务

RINT: CLR RI ; 清中断标志

MOV A, SBUF ; 读取接收数据

MOV @R0, A ; 保存数据

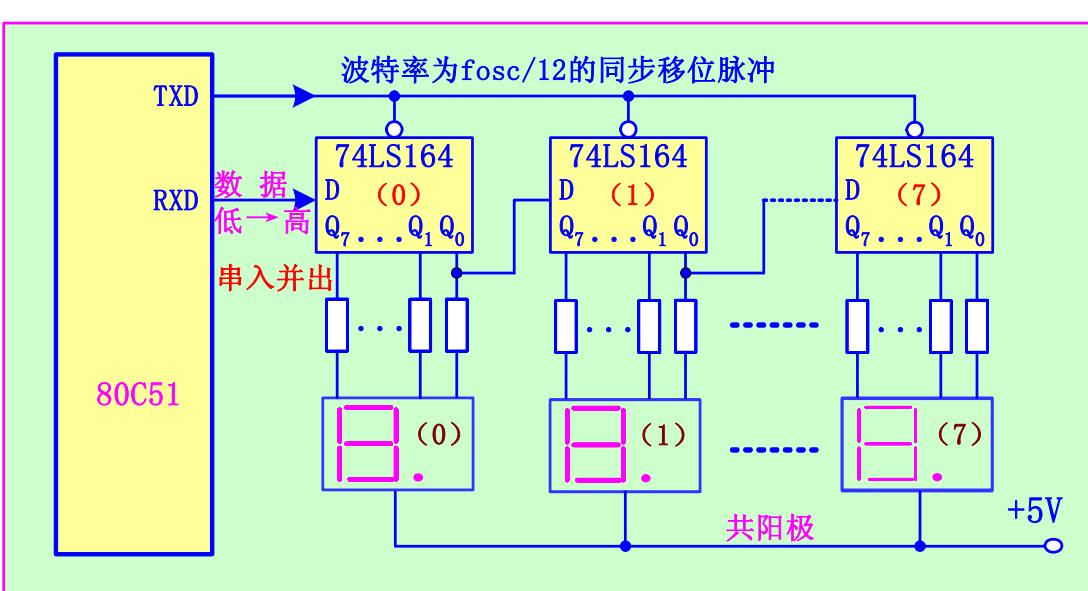
INC R0

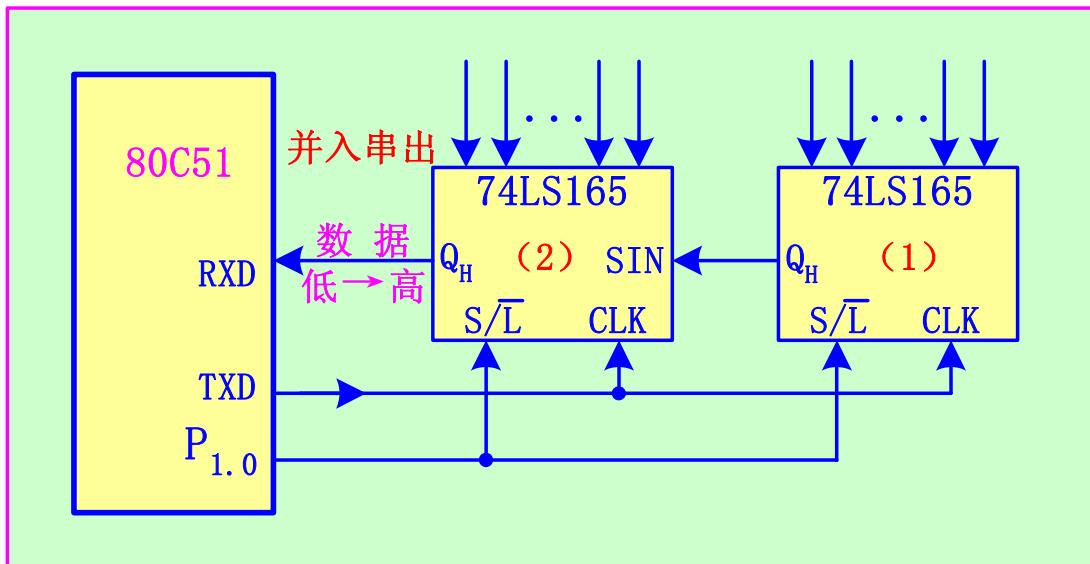
RETI

(一) 串行口方式 0: 用于接移位寄存器扩充并口。

串行口通过接口 74LS164 实现：串行→并行的数据转换(显示器接口)；

通过接口 74LS165 实现：并行 → 串行的数据转换。





程序：

```

MOV R7, #20      ; 送入 20 个字节
MOV R0, #20H     ; 送首地址为 20H
SETB F0          ; 置 1, F0=1 (设置读入字节奇偶数标志)
RCV0: CLR P1.0   ; P1.0=0 (并行置入数据)
SETB P1.0        ; P1.0=1 (允许串行移位)
RCV1: MOV SCON, #10H ; 允许方式 0 接收
JNB RI, $        ; 等待 RI=1, 顺序执行
CLR RI           ; RI=0 为下一帧数据的接收准备
MOV A, SBUF      ; 取数
MOV @R0, A
INC R0
CPL F0          ; 取反, F0=0
JB F0, RCV2      ; F0=1 则转移, F0=0 顺序执行
DEC R7           ; 判是否接收完偶数帧, 接收完则重新并行置入
SJMP RCV1        ; 否则再接收一帧

```

RCV2: DJNZ R7, RCV0 ; R7-1=0?  $\neq 0$  跳 (判是否已读入预定字节数)  
..... ; 对读入数据进行处理

## (二) 异步通讯程序举例:

1.发送程序: 将片内 RAM 50H 起始单元的 16 个数由串行口发送。要求发送波特率为系统时钟的 32 分频, 并进行奇偶校验。

```
MAINT: MOV SCON, #80H      ; 串行口初始化
        MOV PCON, #80H      ; 波特率
        SETB EA
        SETB ES      ; 开串行口中断
        MOV R0, #50H      ; 设数据指针
        MOV R7, #10H      ; 数据长度
LOOP:  MOV A, @R0      ; 取一个字符
        MOV C, P      ; 加奇偶校验
        MOV TB8, C
        MOV SBUF, A      ; 启动一次发送
HERE: SJMP HERE      ; CPU 执行其它任务
        ORG 0023H      ; 串行口中断入口
        AJMP TRANI
TRANI: PUSH A      ; 保护现场
        PUSH PSW
        CLR TI      ; 清发送结束标志
        DJNZ R7, NEXT      ; 是否发送完?
        CLR ES      ; 发送完, 关闭串行口中断
        SJMP TEND
```

```

NEXT: INC    R0          ; 未发送完, 修改指针
      MOV    A, @R0        ; 取下一个字符
      MOV    C, P          ; 加奇偶校验
      MOV    TB8, C
      MOV    SBUF, A        ; 发送一个字符
      POP    PSW          ; 恢复现场
      POP    A
TEND: RETI          ; 中断返回

```

## 2. 接收程序:

串行输入 16 个字符, 存入片内 RAM 的 50H 起始单元, 串行口波特率为 2400(设晶振为 11.0592MHz)。

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

```

RECS: MOV    SCON, #50H          ; 串行口方式 1 允许接收
      MOV    TMOD, #20H        ; T1 方式 2 定时
      MOV    TL1, #0F4H        ; 写入 T1 时间常数
      MOV    TH1, #0F4H
      SETB   TR1          ; 启动 T1
      MOV    R0, #50H        ; 设数据指针
      MOV    R7, #10H        ; 接收数据长度
WAIT: JBC    RI, NEXT          ; 等待串行口接收
      SJMP   WAIT
NEXT: MOV    A, SBUF          ; 读取接收字符
      MOV    @R0, A          ; 保存一个字符
      CLR    RI

```

```

INC    R0          ; 修改指针
DJNZ   R7, WAIT    ; 全部字符接收完?
RET

```

### 3. 接收程序:

串行输入 16 个字符, 进行奇偶校验。

```

RECS: MOV  SCON, #0D0H      ; 串行口方式 3 允许接收
      MOV  TMOD, #20H      ; T1 方式 2 定时
      MOV  TL1, #0F4H      ; 写入 T1 时间常数
      MOV  TH1, #0F4H
      SETB TR1            ; 启动 T1
      MOV  R0, #50H          ; 设数据指针
      MOV  R7, #10H          ; 接收数据长度
WAIT: JBC  RI, NEXT        ; 等待串行口接收
      SJMP WAIT
NEXT: MOV  A, SBUF        ; 取一个接收字符
      JNB  P, COMP          ; 奇偶校验
      JNB  RB8, ERR          ; P≠RB8, 数据出错
      SJMP RIGHT            ; P=RB8, 数据正确
COMP: JB   RB8, ERR
RIGHT: MOV  @ R0, A        ; 保存一个字符
      CLR  RI
      INC  R0          ; 修改指针
      DJNZ R7, WAIT    ; 全部字符接收完?
      CLR  F0          ; F0 =0, 接收数据全部正确

```

REVERR: SETB F0 ; F0=1, 接收数据出错

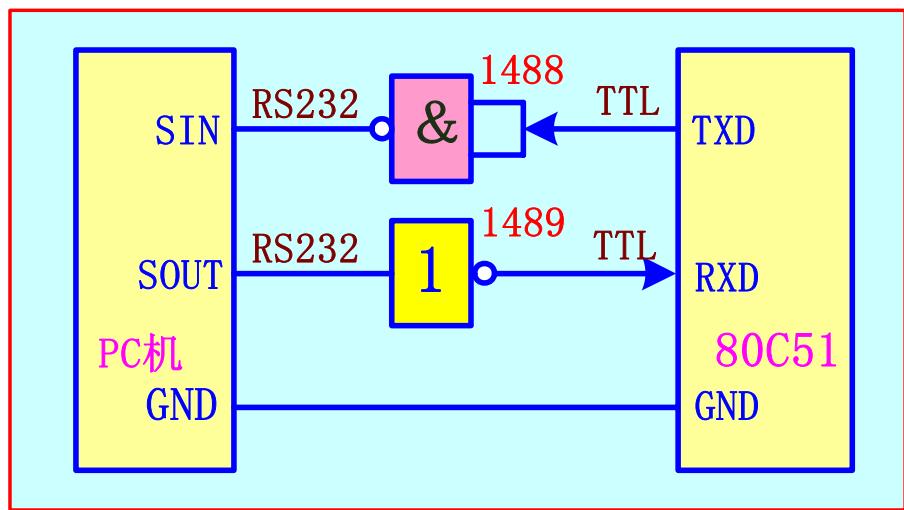
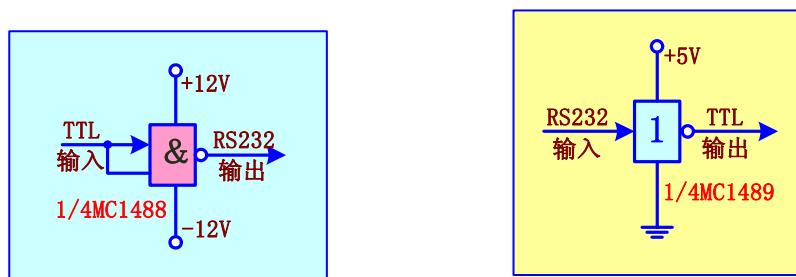
RET

#### 4. 主从分布式微机系统:

也叫集散控制系统: 从机(单片机)作数据采集或实时控制, 主机(PC机)作数据处理、中央管理等。

应用: 过程控制、仪器仪表、生产自动化和企业管理等方面。

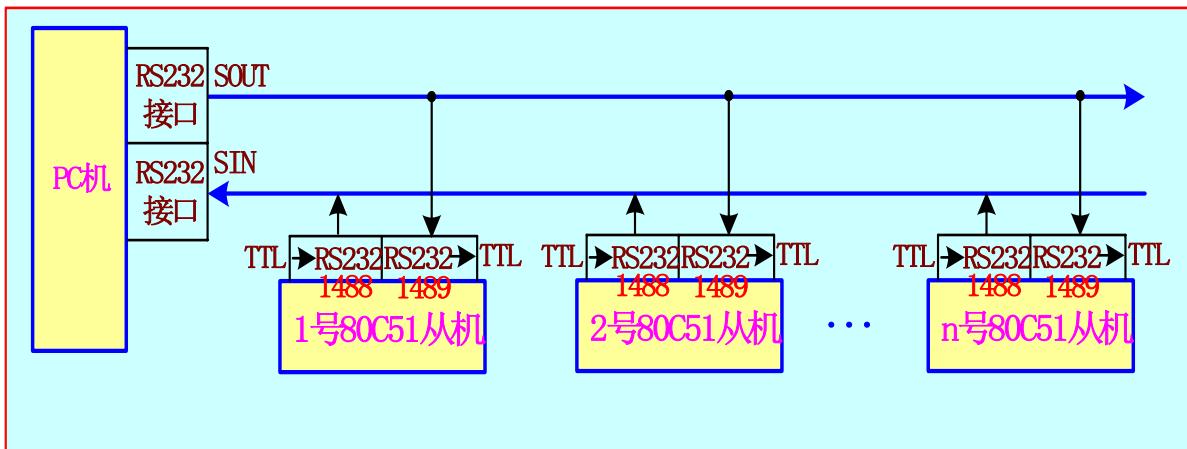
##### ①单机通信:



直接传送串行通信接法

PC机调用的中断指令为: INT 14H

##### ②多机通信系统:



PC 机要对某一指定了地址编号的单片机通讯，就必须作好联络。

- ①PC 机处于发送状态，各单片机的串行口均处于接收状态并使其 SM2=1，作好接收地址信息的准备。
- ②PC 机发出要通讯的那台单片机的地址编号，然后发送通讯数据，发地址时必须使第 9 位信息为 1，发数据时必须使第 9 位数据为 0。
- ③各单片机收到 PC 机发来的地址信息后，因此此时各 SM2=1，所以将引起各单片机的中断。在中断服务程序中，判断 PC 机发来的地址是否是自身的地址编号，仅有符合地址编号的那台才使其 SM2=0，其它不符合者仍是 SM2=1。
- ④随着 PC 机信息的发出（第 9 位信息为 0），因为符合地址编号的那台单片机此时已是 SM2=0，所以这台单片机将再次进入中断，并在中断服务程序中接收 PC 机发来的数据。那些地址不符者，不能进入中断（因 SM2=1），也就不能接收串行来的数据。

接收机的中断服务程序：

已知该机的地址编号为 05H 号，在主程序初始化中已设置了波特率，打开了串行中断，并使 SM2=1。

```

ORG 0023H ; 串行中断入口
JNB RB8, NEXT ; 判断是地址还是数据
MOV A, SBUF ; 读入地址

```

```
XRL    A, #05H      ; 判断地址是否相符
JNZ    EXIT         ; 不符则出中断
CLR    SM2          ; 地址相符则清 SM2
SJMP   EXIT
NEXT: MOV   A, SBUF      ; 读入数据
      MOV   @R0, A       ; 数据存入片内 RAM
      INC   R0          ; 增地址
      CLR   RI          ; 清接收中断标志
EXIT: RETI
```

## 小结

- 1、串行通信的基本原理和基本方式（同步和异步）。
- 2、串行数据的传送方向（单工、半双工和全双工）及信号形式。
- 3、串行口控制/状态控制字 SCON 和节电控制字 PCON。
- 4、串行通信的 4 种工作方式及其对应的波特率、传送位数、时序和应用。
- 5、串行通信的应用：  
发送和接收程序（查询方式和中断方式）。

## 练习题

- (一) 问答题
- (二) 填空题
- (三) 选择题

## 第9章 数/模及模/数转换器接口

### 一、教学要求:

掌握: A/D 和 D/A 转换接口电路及其使用方法。学会单片机与 DAC0832 和 ADC0809 的接口电路与程序。

### 二、教学内容:

9.1 单片机与 D/A 转换器的接口和应用

9.2 单片机与 A/D 转换器的接口和应用

三、教学重点: 单片机与 DAC0832 和 ADC0809 的接口电路与程序。

四、教学难点: A/D 和 D/A 转换接口电路及其使用方法。

五、建议学时: 3 学时。

### 六、教学内容:

#### 9-1 D/A 转换器接口及应用

##### 9-1-1 D/A 转换概述

一、D/A (Digit to Analog) 转换器:

为把数字量转换成模拟量, 在 D/A 转换芯片中要有解码网络:

①权电阻网络;

②倒 T 型电阻网络。

n 位数字量与模拟量的关系式:

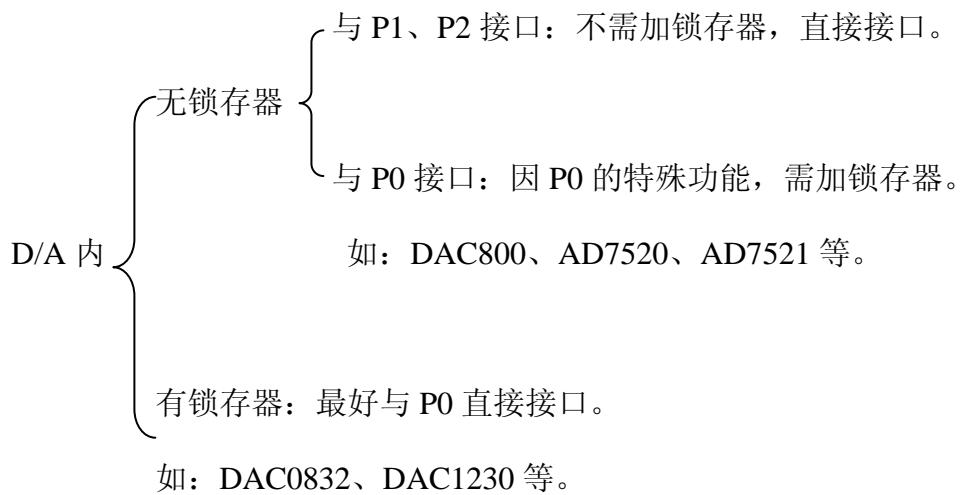
$$V_O = -V_{REF} \times (\text{数字码} / 2^n) \quad (V_{REF} \text{——参考电压})$$

二、D/A 输出形式:

①电压;

②电流  电压。

三、注意区分 D/A 内部是否带有锁存器:



#### 四、主要技术指标:

##### 1、分辨率:

对 D/A 转换器输入量变化敏感程度进行描述, 与输入数字量的位数有关。

- 若数字量的位数为  $n$ , 则分辨率为  $2^{-n}$ 。
- 数字量位数越多, 分辨率就越高。
- 应用时, 应根据分辨率的需要选定转换器的位数。

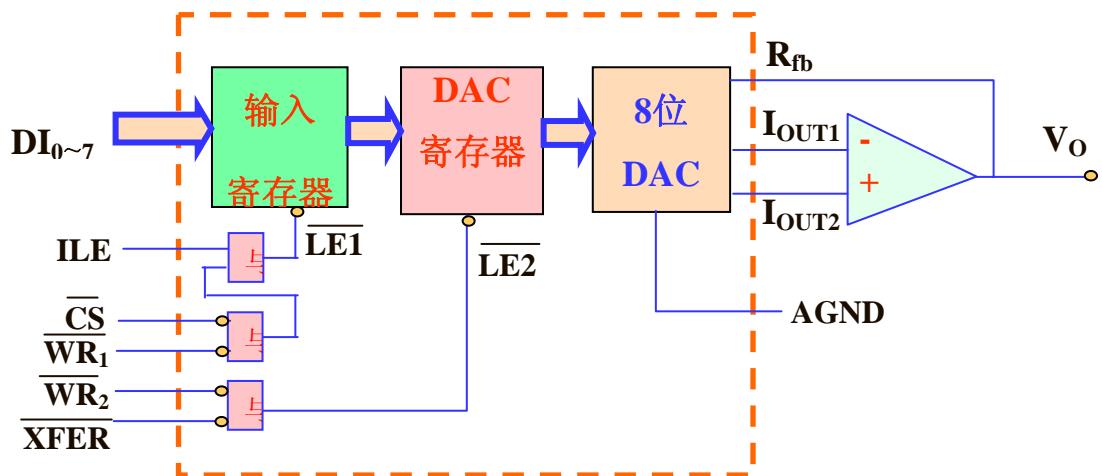
##### 2、建立时间:

描述 D/A 转换速度的快慢。

- 输出形式为电流的转换器比电压的建立时间短。
- D/A 转换速度远高于 A/D 转换。

#### 9-1-2 D/A 转换芯片 DAC0832

##### 一、内部结构: DAC 0832: 8 位双缓冲器结构的 D/A 转换器。



DAC 0832 内部结构框图（请见 P242 图 9.3）

DI0~7: 转换数据输入 (8 位);

CS: 片选信号 (输入);

ILE: 数据锁存允许信号 (输入);

XFER: 数据传送控制信号 (输入);

WR1: 第 1 写信号 (输入), 与 ILE 共同控制输入寄存器是数据直通方式还是数据锁存方式;

WR2: 第 2 写信号 (输入), 与 XFER 共同控制 DAC 寄存器是数据直通方式还是数据锁存方式;

## 二、DAC 0832 与单片机的接口:

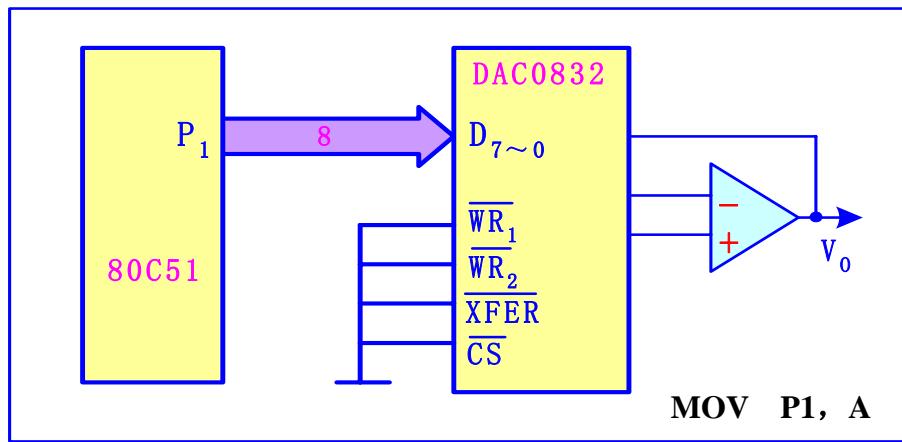
有 3 种工作方法:

1、直通方式:

输入寄存器和 DAC 寄存器共用一个地址, 同时选通输出;

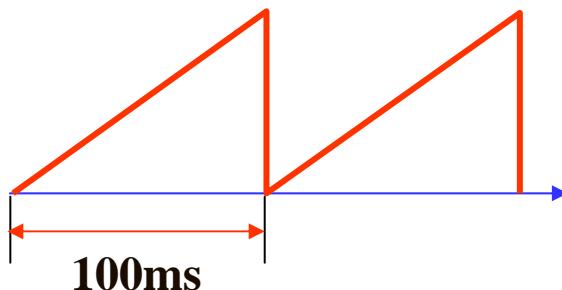
WR1 和 WR2 同时进行, 并且不与 CPU 相接。

特点: 转换速度快。



举例：

例：D/A 转换程序，用 DAC0832 输出 0~+5V 锯齿波，电路为直通方式。设 VREF=-5V，DAC 0832 地址为 00FEH，脉冲周期要求为 100ms。



```

DACS: MOV      DPTR, #00FEH      ; 0832 I/O 地址
      MOV      A, #00H          ; 开始输出 0V
      DACL: MOVX    @DPTR, A      ; D/A 转换
      INC      A                  ; 升压
      ACALL   DELAY            ; 延时 100ms/256: 决定锯齿波的周期
      AJMP    DACL            ; 连续输出
      DELAY: ...          ; 延时子程序

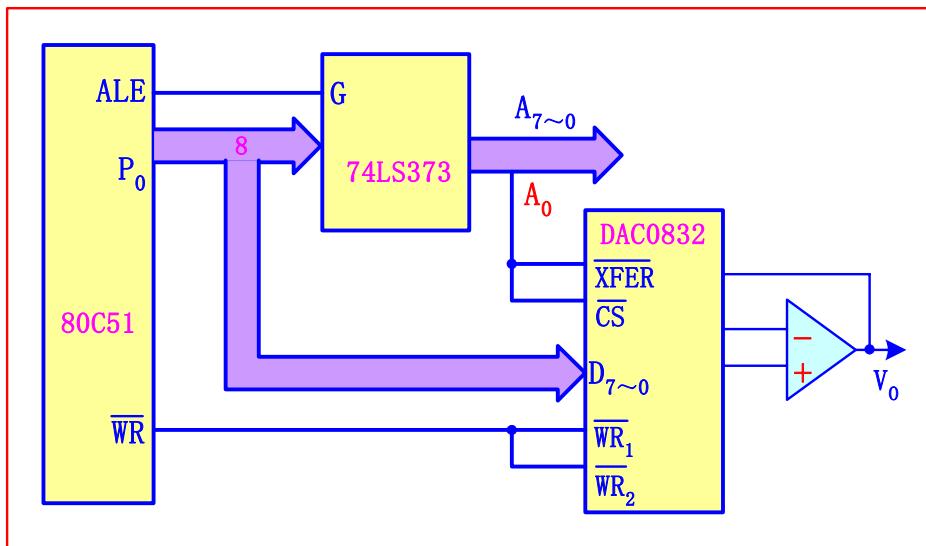
```

2、单缓冲方式：

输入寄存器和 DAC 寄存器共用一个地址，同时选通输出，输入数据在控制信号作用下，直接进入 DAC 寄存器中；

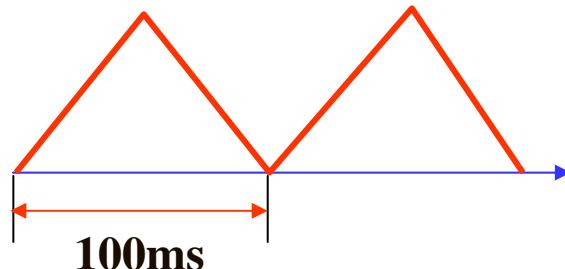
WR1 和 WR2 同时进行，并且与 CPU 的 WR 相连，CPU 对 0832 执行一次写操作，将数据直接写入 DAC 寄存器中。

适用：只有一路模拟信号输出或几路模拟信号非同步输出。



举例：

例：D/A 转换程序，用 DAC0832 输出  $0 \sim +5V$  三角波，电路为单缓冲方式。设  $V_{REF} = -5V$ ，DAC 0832 地址为 00FEH，脉冲周期要求为 (100ms)。



ORG 2000H

```

STAR:  MOV      DPTR, #00FEH      ; 0832 I/O 地址
        MOV      A, #00H          ; 开始输出 0V
UP:    MOVX    @DPTR, A          ; D/A 转换
        INC      A              ; 产生上升段电压
        JNZ      UP             ; 上升到 A 中为 FFH (A≠0 跳)

```

```

DOWN: DEC      A          ; 产生下降段电压
      MOVX @DPTR, A
      JNZ DOWN      ; 下降到 A 中为 00H
      SJMP UP       ; 重复

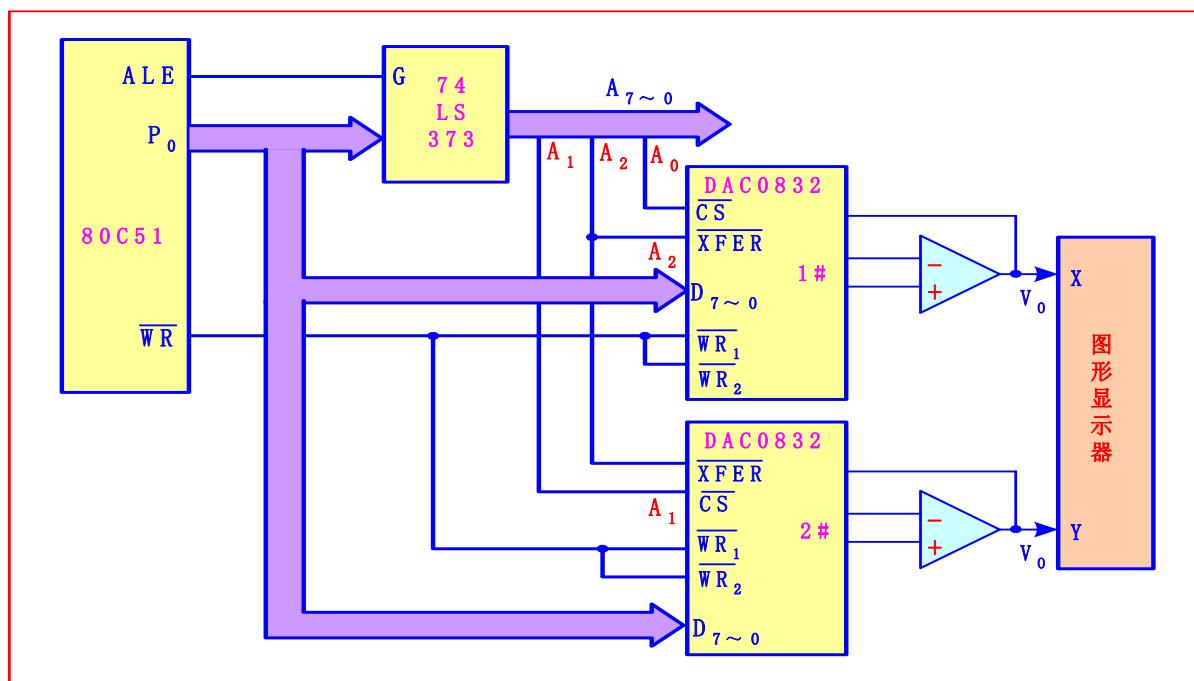
```

注：若想改变波形的周期（频率），只需在 SJMP UP 前插入延时程序即可。

### 3、双缓冲器方式：

输入寄存器和 DAC 寄存器分配有各自的地址，可分别选通用同时输出多路模拟信号。

适用：同时输出几路模拟信号的场合，可构成多个 0832 同步输出电路。



举例：

例：用 DAC0832 实现驱动绘图仪，电路为双缓冲方式。

1#和 2#DAC 0832 地址分别为 00FEH 和 00FDH。

则绘图仪的驱动程序为：

```

ORG 2000H

MOV DPTR, #00FEH ; 选中 1# 0832 (的输入寄存器): A0=0
MOV A, #Datax
MOVX @DPTR, A ; Datax 写入 1# 0832 输入寄存器
MOV DPTR, #00FDH ; 选中 2# 0832 (的输入寄存器): A1=0
MOV A, #Datay
MOVX @DPTR, A ; Datay 写入 2# 0832 输入寄存器
MOV DPTR, #00FBH ; 选中 1#和 2# 0832 的 DAC 寄存器: A2=0
MOVX @DPTR, A ; 1#和 2#输入寄存器的内容同时传送到
                           DAC 寄存器中

```

## 9-2 A/D 转换器接口及应用

一、转换原理:

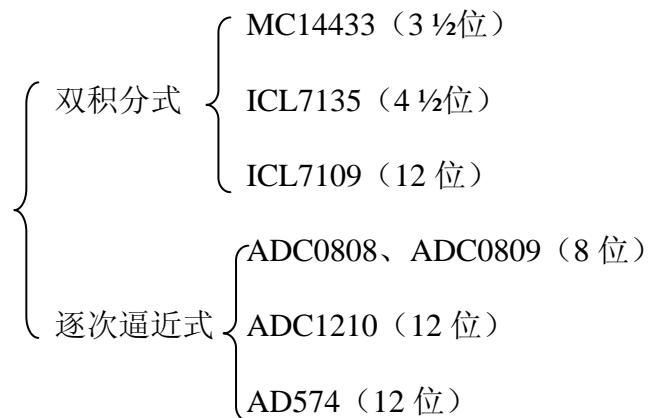
A/D 转换是把模拟量信号转化成与其大小成比例的数字信号。

A/D 转换电路主要分成:

逐次逼近式 (速度较快, 精度较高: 常用);

双积分式 (速度慢, 精度高: 用于速度要求不高的场合)。

常用芯片:

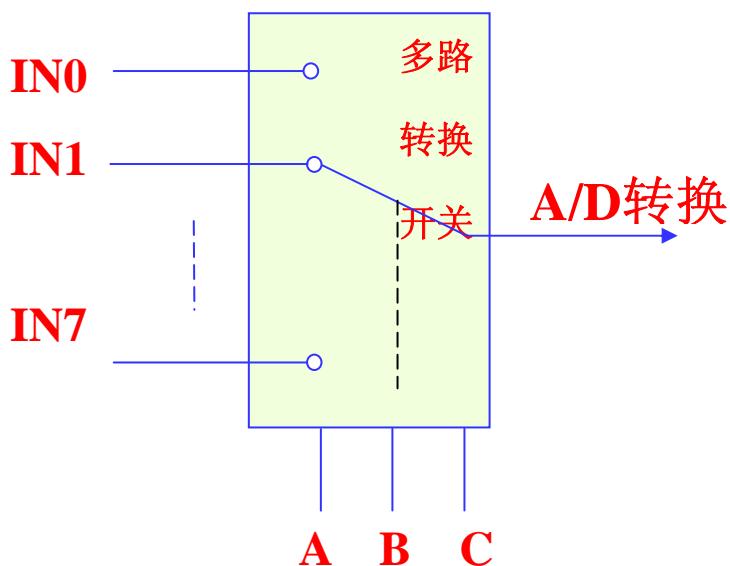


## 二、ADC0809 与单片机的接口：

ADC 0809/0808 为 8 路输入通道、8 位逐次逼近式 A/D 转换器，可分时转换 8 路模拟信号。

### 1、结构：

一个 8 位逐次逼近式 A/D 转换器、8 路模拟转换开关、3-8 地址锁存译码器和三态输出数据锁存器（详见 P249）。



### 2、引脚：

- (1) 8 路模拟量分时输入信号端： IN0～IN7；
- (2) 8 位数字量输出信号端： D0～D7；
- (3) 通道选择地址信号输入端： ADDA、ADDB、ADDC；
- (4) 基准参考电压为 VR (+) 和 VR (-)；

决定输入模拟量的范围。

典型值分别为 +5V 和 0V。

- (5) 转换结束信号 EOC：

1：正在进行转换；

0：一次转换完成。

(6) 时钟信号输入端: CLK (其内部无时钟电路)。

3、ADC 0809 与单片机连接:

涉及 2 个问题:

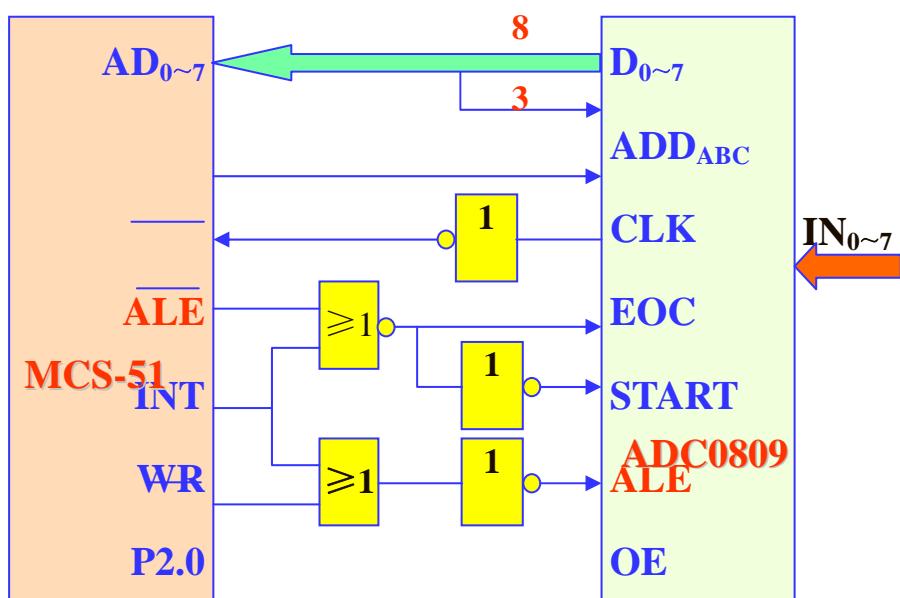
- (1) 8 路模拟信号通道选择;
- (2) A/D 转换完成后转换数据的传送。

转换数据的传送:

- ①定时传送方式; (不需接 EOC 脚)
- ②查询方式; (测试 EOC 脚的状态)
- ③中断方式。(EOC 脚接 INT 脚)

注: (1) 不能用无条件方式;

(2) 2 个 ALE 不能相接。



ADC0809 的口地址: FFFFH;

8 路模拟通道的地址: FEF8H~FEFFH。

A/D 转换程序: (延时等待方法)

```
MOV      DPTR, #0FEFFH      ; ADC0809 地址
```

```

MOV      A, #00H          ; 选中 IN0
MOVX    @DPTR, A          ; 启动 A/D 转换
LCALL   DELAY            ; 等待转换结束
MOVX    A, @DPTR          ; 读转换结果
RET

```

不用接 EOC 脚，采用定时传送方式。

例：P252 应用举例

设有一个 8 路模拟量输入的巡回检测系统，采样数据依次存放在外部 RAM 0A0H～0A7H 单元中，ADC0809 的 8 个通道地址为 0FEF8H～0FEFFH。

初始化程序：（中断方式）

```

MOV  R0, #0A0H          ; 数据存储区首地址
MOV  R2, #08H            ; 8 路计数器
SETB IT1                ; 边沿触发方式
SETB EA                 ; 中断允许
SETB EX1                ; 允许外部中断 1 中断
MOV  DPTR, #0FEF8H       ; 指向 ADC0809 首地址
LOOP: MOVX  @DPTR, A      ; 启动 A/D 转换
HERE: SJMP  HERE          ; 等待中断
DJNZ  R2, LOOP            ; 巡回，未完继续
CLR   EA                 ; 结束，关中断
SJMP  $                  ; 结束停止

```

中断服务程序：

```

MOVX  A, @DPTR          ; 读数
MOVX  @R0, A              ; 存数

```

INC DPTR ; 指向下一模拟通道

INC R0 ; 指向数据存储区下一单元

RETI

## 小 结

- 1、D/A 转换原理、内部结构、信号输出形式和主要技术指标。
- 2、DAC0832 内部结构、管脚、3 种工作方法及其对应接口的特点、电路和应用程序。
- 3、A/D 转换原理和常用 ADC 芯片。
- 4、ADC0809 的内部结构及管脚、转换数据传送方式及其对应接口图和程序。

## 练习题

(一) 问答题

(二) 填空题

(三) 选择题

# 第10章 单片机应用及开发技术

## 一、教学要求：

掌握：单片机应用系统的设计过程、开发工具和方法，以及提高可靠性的方法。

## 二、教学内容：

10.1 单片机应用举例

10.2 单片机系统可靠性接地

## 三、教学重点：单片机应用系统的设计过程、开发工具和方法。

## 四、教学难点：单片机应用系统的具体设计。

五、建议学时：5学时。

## 六、教学内容：

### 10-1 作息时间控制钟

时钟产生  $\left\{ \begin{array}{l} \text{硬件：时钟电路片} \\ \text{软件：片内定时器} \end{array} \right.$

在单片机计时的过程中，每一次秒加1，都与规定的作息时间比较，如比较相等就进行电铃或扩音设备的开关控制。

本系统共有4项控制内容：接通电铃和断开电铃；

接通和断开扩音设备。

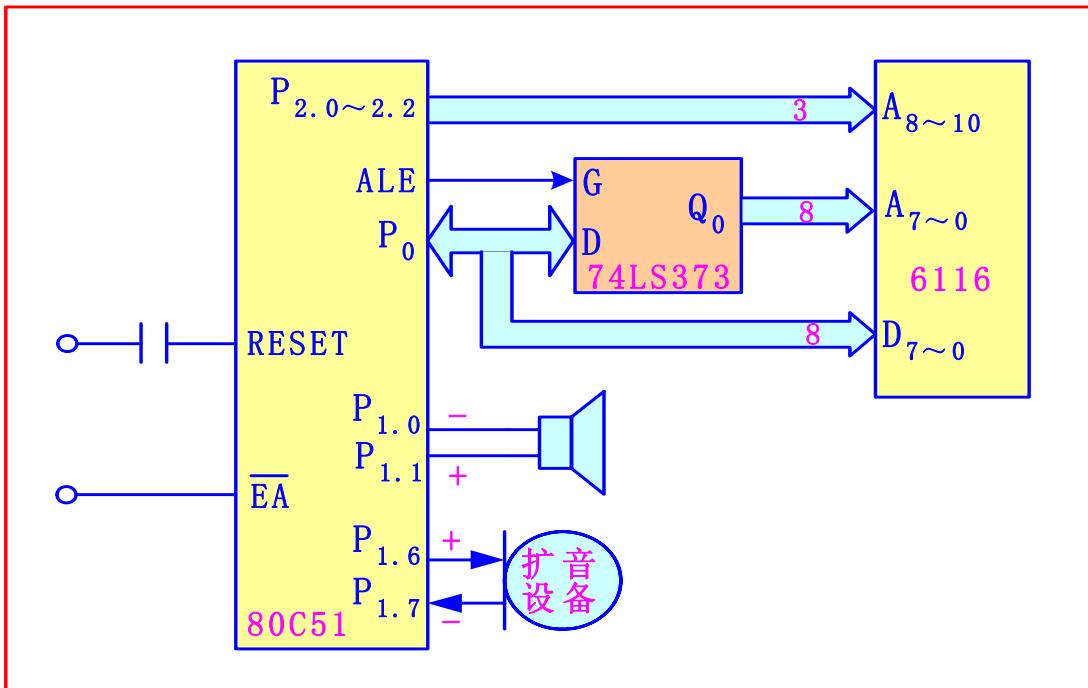
由P1口输出控制码进行控制，其控制码定义为：

接通电铃：0FEH

断开电铃：0FDH

接通扩音设备：7FH

断开扩音设备：0BFH



由 P1 口输出控制码进行控制：

接通电铃：0FEH

断开电铃：0FDH

接通扩音设备：7FH

断开扩音设备：0BFH

构造 4 个字节的存储字，放在外部 RAM50H 开始的存储区中：

格式：

×	×	×	×	×	×	×	×
---	---	---	---	---	---	---	---

开关设备控制码字节

时字节

分字节

秒字节

存储单元	开控制码	时	分	秒	关控制码	时	分	秒
50H~57H	0FEH	08	00	00	0FDH	08	00	10
58H~5FH	0FEH	08	50	00	0FDH	08	50	10
60H~67H	0FEH	09	00	00	0FDH	09	00	10
68H~6FH	0FEH	09	50	00	0FDH	09	50	10
70H~77H	7FH	09	52	00	0BFH	10	05	00
78H~7FH	0FEH	10	10	00	0FDH	10	10	10
80H~87H	0FEH	11	00	00	0FDH	11	00	10
88H~8FH	0FEH	11	10	00	0FDH	11	10	10
90H~97H	0FEH	12	00	00	0FDH	12	00	10
98H~9FH	0FEH	13	30	00	0FDH	13	30	10
0A0H~0A7H	0FEH	14	20	00	0FDH	14	20	10
0A8H~0AFH	0FEH	14	30	00	0FDH	14	30	10
0B0H~0B7H	0FEH	15	20	00	0FDH	15	20	10
0B8H~0BFH	7FH	15	21	00	0BFH	15	50	00
0C0H~0C7H	00H	×	×	×				

程序：

1、主程序：为时钟记时程序，使用内部 RAM 单元：

20H 秒单元

21H 分单元

22H 时单元

每运行一次秒加 1 操作时（参见 P158~162），都调用时间比较子程序。

2、时间比较子程序：记时时间与存储字中的预置时间进行比较：

相等：作息时间已到，发出开关控制码，控制电铃或扩音设备的开或断；

不等：子程序返回。

50H——存储区首地址；

R0——存储区地址指针；

2EH——存储区地址指针暂存单元；

6AH——存开关控制码；

6BH~6DH——依次存放存储字的小时值、分值和秒值。

```

LOOP1:  MOV  R0, #4CH      ; 存储字存储区首地址减 4
        MOV  2EH, R0      ; 送暂存单元

LOOP2:  MOV  R0, 2EH
        MOV  R3, #04H      ; 循环 4 次
        MOV  R1, #23H

LOOP3:  INC  R0      ; 地址指针加 4, 得开关控制码地址
        DJNZ R3, LOOP3
        MOV  2EH, R0      ; 暂存开关控制码地址
        MOV  R3, #03H      ; 循环 3 次
        MOVX A, @R0      ; 读取控制码
        JZ   A, LOOP5      ; 控制码为“0”（结束）则返回
        MOV  6AH, A      ; 存控制码

LOOP4:  INC  R0
        ; 地址指针增量: 50H→51H（时）→52H（分）→53H（秒）
        DEC  R1
        ; 记时单元地址减量: 23H→22H（时）→21H（分）→20H（秒）

        MOVX A, @R0      ; 读取作息时间（时、分、秒）
        MOV  6BH, A      ; 存作息时间
        MOV  A, @R1      ; 读取记时时间
        CJNE A, 6BH, LOOP2
        ; 记时时间（A）与预置作息时间（6BH）比较:
        ; 不等则转, 继续读下面的控制码（时、分、秒）

        DJNZ R3, LOOP4      ; 共读取 3 次
        MOV  A, 6AH      ; 开关控制码送 A

```

```

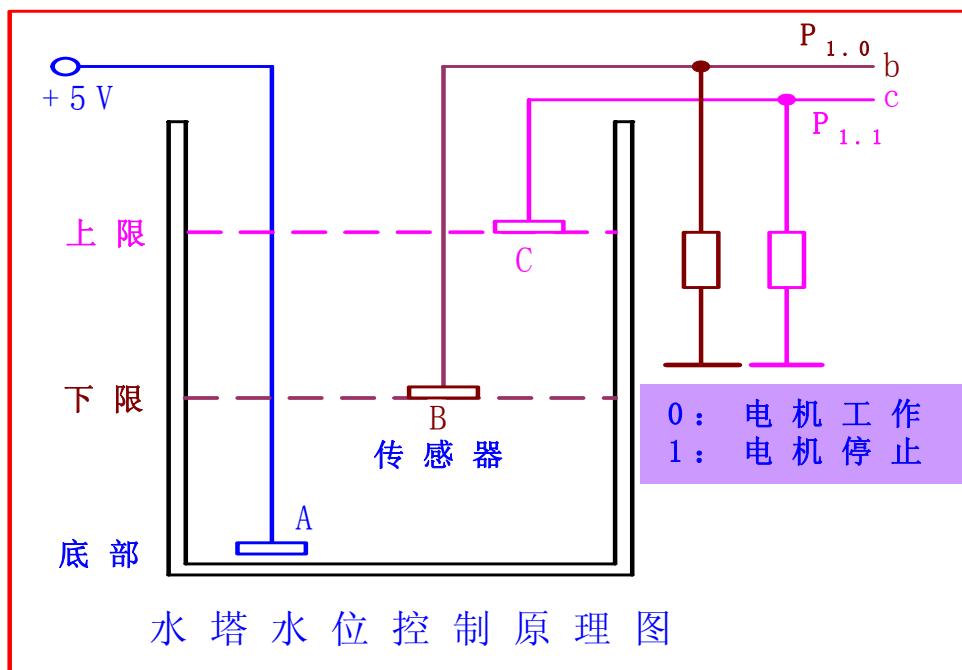
CPL    A          ; 取反 (增大驱动能力)

MOV    P1, A        ; 开关控制码输出

LOOP5:  RET          ; 返回

```

## 10-2 水塔水位控制



### 一、控制原理：

虚线表示允许水位变化的上下限。

水塔由电机带动水泵供水，单片机控制电机转动以达到对水位控制的目的。

- ①当水位上升，达到上限时，因水导电，B、C 棒连通+5V。b、c 均为“1”，应停止电机和水泵的工作，不再供水；
- ②当水位降到下限时，B、C 棒都不能与 A 棒导电。b、c 均为“0”，应启动电机，带动水泵工作，给水塔供水；
- ③当水位处于上下限之间时，B 与 A 棒导通。b 为“1”，c 为“0”，无论怎样都应维持原有的工作状态。

## 二、控制电路：

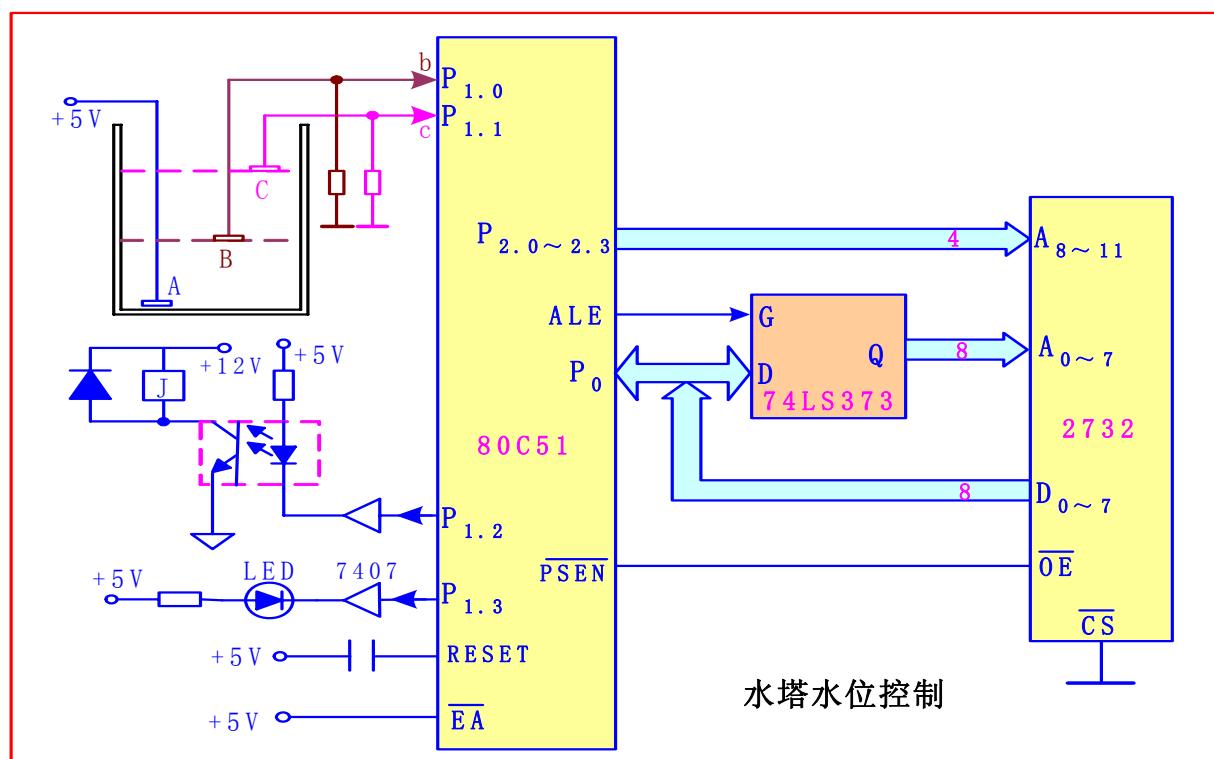
上下限水位信号由 P1.0 和 P1.1 输入，这 2 个信号共有 4 种组合状态：

输入		输出		操作
C (P1.1)	B (P1.0)	P1.2	P1.3	
0	0	0	1	电机运转
0	1	×	1	维持原状
1	0	×	0	故障报警
1	1	1	1	电机停转

控制信号由 P1.2 端输出，去控制电机。

为了提高控制的可靠性，使用了光电耦合；

由 P1.3 输出报警信号，驱动一支发光二极管进行光报警。



程序：

ORG 8000H

AJMP LOOP

```

LOOP: ORL P1, #03H      ; P1.0=P1.1=1, 为检查水位状态做准备
      MOV A, P1
      JNB ACC.0, ONE      ; P1.0=0 则跳转
      JB ACC.1, TWO       ; P1.1=1 则跳转
BACK: ACALL D10S        ; P1.0=1、P1.1=0 维持原状时就延时 10S
      AJMP LOOP
ONE:  JNB ACC.1, THREE   ; P1.1=0 则跳转
      CLR 93H (P1.3)      ; P1.0=0、P1.1=1 时, P1.3=0 启动报警装置
      SETB 92H (P1.2)      ; P1.2=1 停止电机工作
FOUR: SJMP FOUR
THREE: CLR 92H          ; P1.0=P1.1=0 时启动电机
      AJMP BACK
TWO:  SETB 92H          ; 停止电机工作
      AJMP BACK

```

延时子程序 D10S (延时 10S):

```

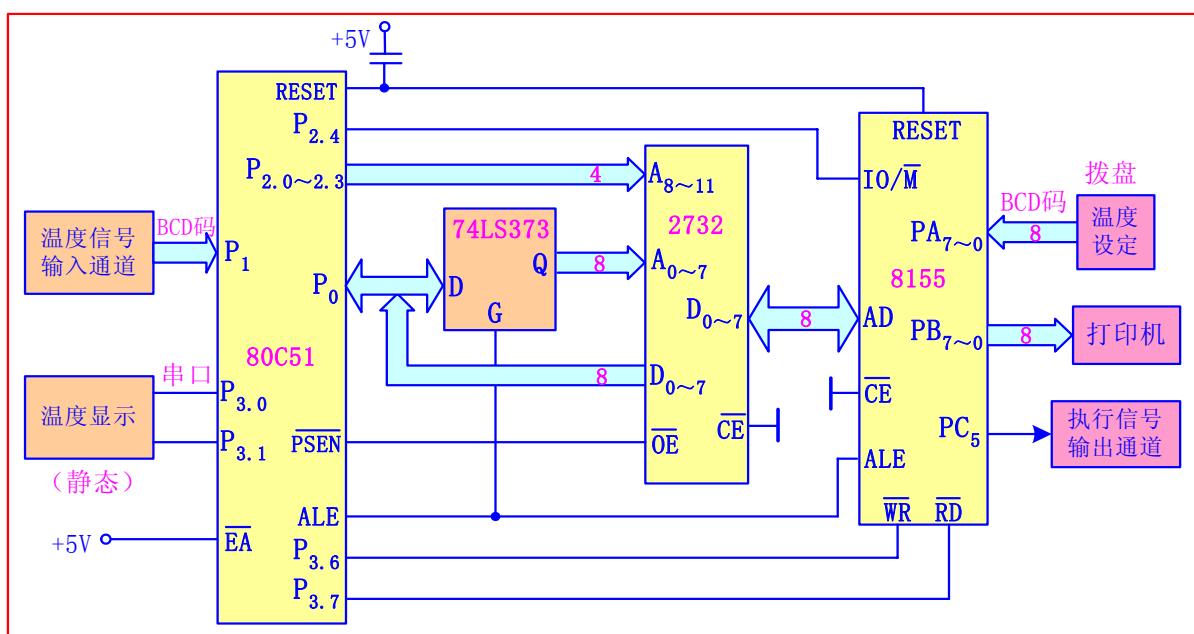
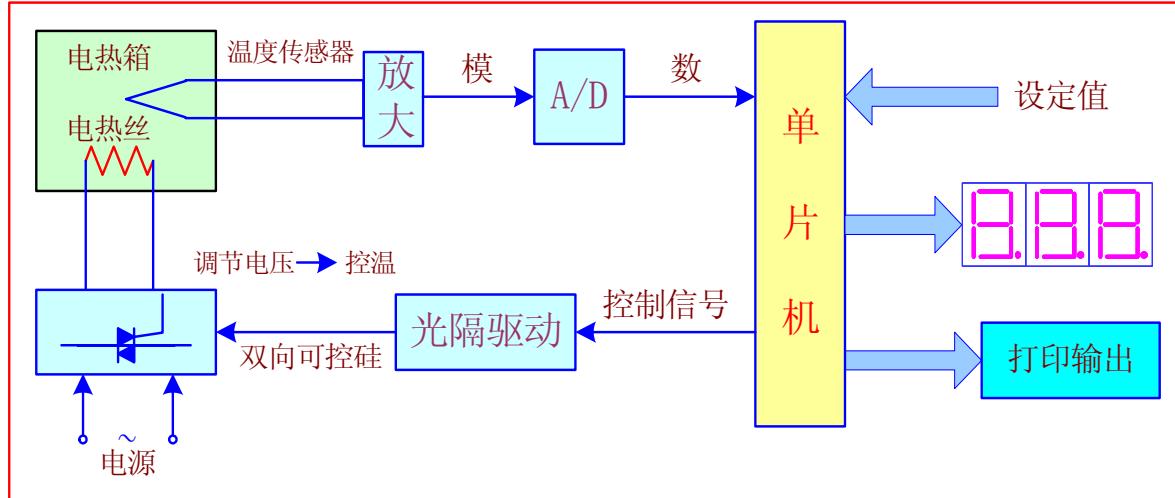
ORG 8030H
      MOV R3, #19H      ; R3=25D
LOOP3: MOV R1, #85H      ; R1=133D
LOOP1: MOV R2, #0FAH      ; R2=250D
LOOP2: DJNZ R2, LOOP2
      DJNZ R1, LOOP1
      DJNZ R3, LOOP3
      RET

```

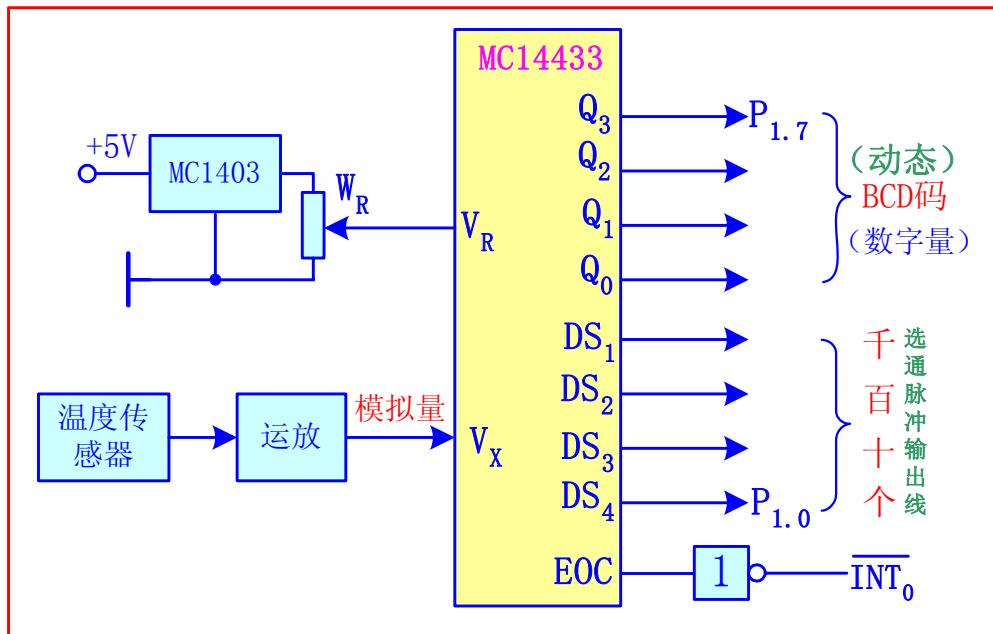
计算延时时间: (若 fosc=6MHz, 则 T=2  $\mu$  S)

$$t = \{ [(2 \times 2 \mu \text{S} \times 250) + 3 \times 2 \mu \text{S}] \times 133 + 3 \times 2 \mu \text{S} \} \times 25 + 3 \times 2 \mu \text{S} = 3345106 \mu \text{S} \approx 3.3 \text{S}$$

### 10-3 电热箱单片机温控系统



单片机具有温度设定、显示、给出采样温度反馈值、输出温度控制量以及作调节器运算等功能。



MC14433 是双积分  $3\frac{1}{2}$  位的 A/D 转换器：采用扫描的方法，输出  $3\frac{1}{2}$  位的 BCD 码，从 0000~1999 共 2000 个数码。内部有时钟源（振荡器）。

$V_R$ ：基准电压输入线，其值为 200mV 或 2V；

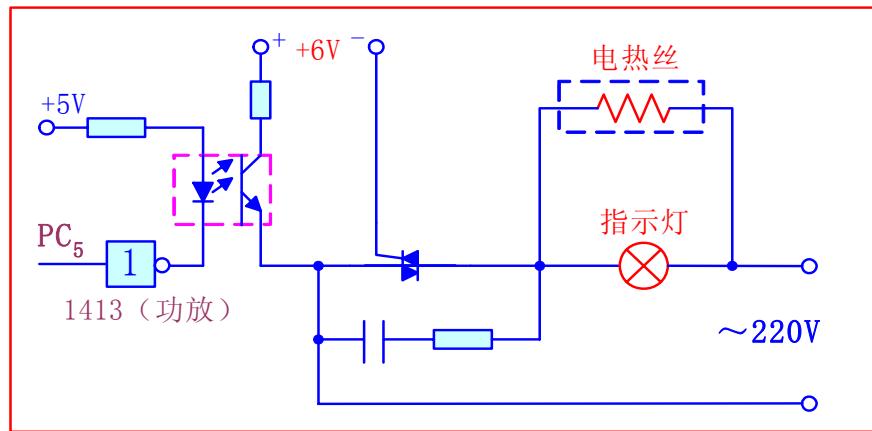
$V_X$ ：被测电压输入线，最大为 199.9mV 或 1.999V。

DS4~DS1：分别是个、十、百、千位的选通脉冲输出线；

Q3~Q0：BCD 码数据输出线，动态地输出千位、百位、个位值。

即 DS4 有效时，Q3~Q0 表示的是个位值 (0~9)；依次类推。

EOC 与 INT0 相接使得 MC14433 每次 A/D 结束后，同时启动下一次转换，使其处于连续的 A/D 转换中，并使得单片机在中断服务程序中读入该次转换结果。



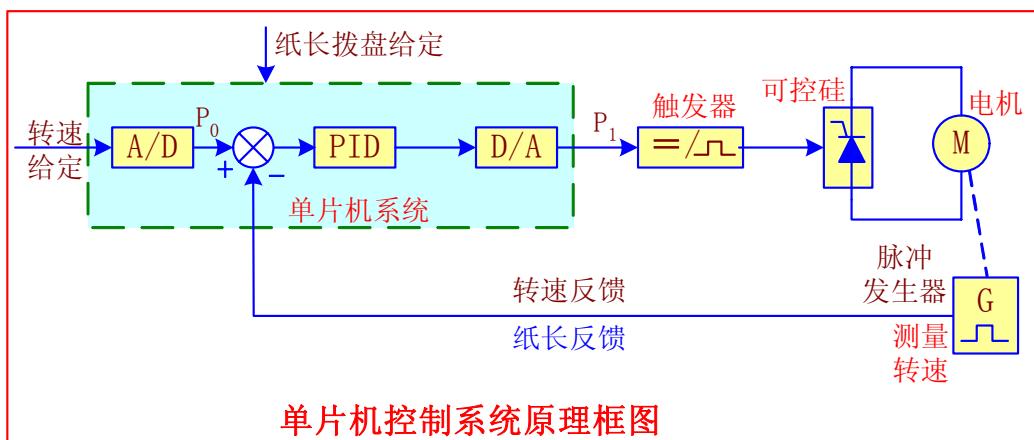
输出高电平：双向可控硅导通，电热丝通电；

输出低电平：双向可控硅截止，电热丝断电。

8155 端口的负载能力不足以驱动光电耦合器的发光二极管，用 1413 作为功放。

控制算法：对于温度控制系统，系统具有大热惯性，可用 PID 算法、Smith 算法、Dalin 算法等。

#### 10-4 纸机转速、纸长的单片机控制

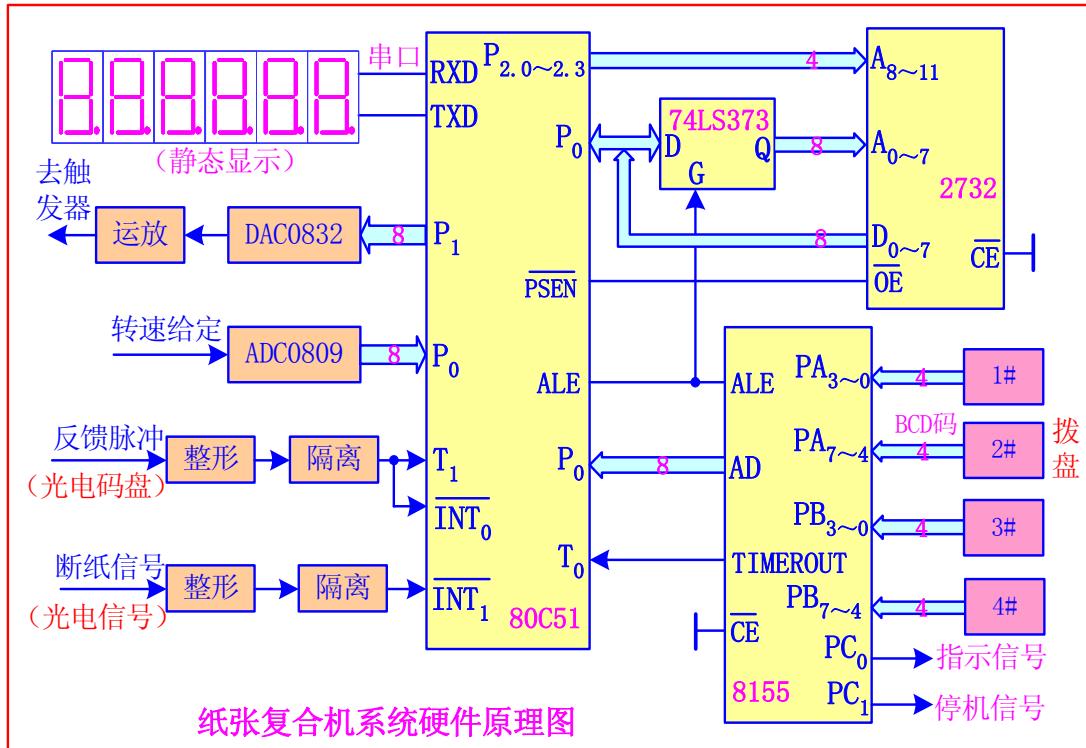


转速控制采用带转速单闭环的直流电动机调速系统；

驱动电路由晶闸管-直流电动机构成；

控制电路主要包括转速给定、转速反馈、PID 调节器、晶闸管脉冲触发电路；

走纸的长度控制由纸长设定、纸长脉冲反馈构成。



T1 计量走纸长度 (1cm/脉冲);

T0 计量信号个数 (与 INT0 配合);

INT0 检测纸机转速 ( $V=D/nTc$ );

INT1 检测断纸 (通过延时);

晶闸管的控制信号 (由 D/A0832 提供)。

走纸长度由拨盘设置;

转速的给定由模拟电压经 ADC0809 转换后设置;

6 位数码管分别用于显示转速 (前 2 位) 和走纸长度 (后 4 位)。

系统控制功能:

1、纸长的设定: (拨盘  $\rightarrow$  8155  $\rightarrow$  80C51)

设置 8155 的 PA 口和 PB 口为基本输入方式, PC 口为基本输出方式,

则 8155 的控制字为 11001100=0CCH;

计数器取分频系数为 1000D=03E8H, 并输出方波信号,

则计数器初值应设置成 0100 0011 1110 1000B = 43E8H

程序：

```
MOV R0, #00H           ; 写控制字 (8155 设定)
MOV A, #0CCH
MOVX @R0, A
MOV R1, #04H           ; 写计数器 (T0) 初值与工作方式
MOV A, #0E8H           ; 置 TL
MOVX @R1, A
INC R1
MOV A, #43H           ; 置 TH
MOVX @R1, A
MOV R0, #01H           ; 把 PA 口内容读入单片机 RAM7FH
MOVX A, @R0           ; 读纸长给定
MOV 7FH, A
INC R0                 ; 把 PB 口内容读入单片机 RAM7EH
MOVX A, @R0
MOV 7EH, A
INC R0                 ; 把 #01H 由 PC 口输出 (PC.0=1)
MOV A, #01H           ; 接信号灯：走纸到否？
MOVX @R0, A
```

2、纸长检测与控制：

走纸长度的设置由 4 位拨盘设定，走纸的检测信号来自线速度不变的码盘脉冲。两脉冲间的距离表示一定的纸长（脉冲当量），当反馈脉冲的引入量达到一定数量后，可使设定值不断做减 1 记数，直至为 0 后停车。

采用脉冲当量为 1cm/脉冲, 当走纸长度单位为 10m 时, 1000 个反馈脉冲可使纸长设定值减 1。由 T1 来实现。由于  $1000D=03E8H$ , T1 的记数初值应为  $(03E8H)$  补  $=FC18H$ , 工作于方式 1。

注:  $10m/1cm=1000D$

每 10 米减 1

T1 的中断服务程序:

```
T1INT: PUSH A          ; 保护
      PUSH PSW
      MOV TH1, #0FCH ; 重置初值
      MOV TL1, #18H
      DEC 7FH          ; 纸长减 1
      MOV A, 7FH
      ANL A, #0FH      ; 取低位
      CJNE A, #0FH, ED ; 判断是否在 BCD 码范围
      DEC 7FH          ; BCD 码调整 (7FH) =#0FH
      DEC 7FH          ; =15
      DEC 7FH          ; (15-6=9)
      DEC 7FH
      DEC 7FH
      DEC 7FH
      ED: POP PSW      ; 恢复
      POP A
      RETI             ; 中断返回
```

### 3、保护：

纸机运行过程中，是否出现断纸现象的检测来自光电信号，经 INT1 引入单片机内。为了区别是过纸出现空洞还是确实出现断纸现象，单片机根据无纸信号出现的时间长短加以判断。

如：当无纸信号持续 1S 后消失，说明无断纸现象，则微机系统仍然正常运行；如超过此时间后，无纸信号依然存在，则判断为出现了断纸现象，单片机立即停机。

程序如下：

```
PIP:    JNB    P3.3, NEXT5      ; 无断纸信号则返回
        MOV    R5, #0AH        ; 延时 (0AH=10D)
LOOP2:  MOV    R7, #32H        ; 32H=50D
LOOP1:  MOV    R6, #00H
        DJNZ   R6, $
        DJNZ   R7, LOOP1
        DJNZ   R5, LOOP2
        JNB    P3.3, NEXT5      ; 再判断有无断纸信号
        AJMP   ED              ; 有断纸信号则停机
NEXT5:  RET
```

### 小结

单片机应用系统是以单片机为核心，扩展外围芯片和电路，能完成一定任务的微机系统。

单片机具有体积小、成本低、抗干扰强、使用方便灵活等优点，已广泛应用于生产和科技等各个领域。

设计步骤：

1、确定系统控制方案，彻底了解控制对象和控制要求。

高速对象（电机调速、图像语音识别等）还是慢速对象（温度、流量等过程控制）；  
开环控制还是闭环控制；等等。

2、确定控制算法，根据系统数学模型和控制要求，选择单片机的控制规律。

直流电机传动系统多用 PID 控制，

交流传动则除 PID 外，还用矢量变换控制；

温度调节等滞后系统多采用达林算法与施密斯预估算法等。

3、微机选型：综合考虑控制要求、经济条件等多种因素进行。

MCS-51 系列单片机是 8 位高档机；

在要求更高的场合，应考虑采用 MCS-96 系列单片机；

对运算速度要求更高的场合，可考虑采用数字信号处理器（DSP）。

4、硬件设计：在系统控制方案的基础上，根据单片机本身的硬件资源，确定出整个系统的控制电路。

单片机片内的资源应充分予以利用，只有在不能满足要求时，才需要扩展。

5、软件设计：依据控制算法和控制电路。

通常硬件少则软件多，反之亦然。

程序编制法则：

画框图→确定软件功能模块→确定流程图→调试子程序→程序总调。

6、系统总调：根据制成的硬件电路和调试过的程序做系统总调。

常用单片机仿真完成，然后固化软件，脱开仿真器，插回单片机与固化了的程序存储器。

——完——