



0.1.1 数制与转换

1. 常用的数制

➤ 十进制数 (Decimal)

十进制有 0、1、2、3、4、5、6、7、8、9 共 10 个数字符号，其基是 “10”，即“逢 10 进 1”。**十进制数的后缀是 D，但通常可以省略。**

十进制各位的权是以 10 为底的幂。如：368D 按权的展开式为：

$$368D = 3 \times 10^2 + 6 \times 10^1 + 8 \times 10^0$$

➤ 二进制数 (Binary)

二进制只有 0、1 共 2 个数字符号；其基是 “2”，即“逢 2 进 1”。

二进制数的后缀是 B。

二进制各位的权是以 2 为底的幂。如：1101B 按权的展开式为：

$$1101B = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 8 + 4 + 0 + 1 = 13$$



0.1.1 数制与转换

1. 常用的数制

➤ 十六进制数 (Hexadecimal)

十六进制有 0、1、2、3、4、5、6、7、8、9、A、B、C、D、E、F 共 16 个数字符号，其基是 “16”，即“逢 16 进 1”。其中 A-F 相当于十进制数的 10~15。十六进制数的后缀是 H，或者前缀 0X。

$$A3EH = 10 \times 16^2 + 3 \times 16^1 + 14 \times 16^0 = 2560 + 48 + 14 = 2622$$

各数制的作用：

- 十进制是人们最熟悉、方便使用的数制；
- 二进制是计算机使用和方便硬件实现计算的数制；
- 十六进制与二进制的转换十分方便，其作用仅仅是用来简化和方便二进制的书写和阅读。



0.1.2 微机中数的表示

1. 无符号数

无符号数不需要符号位，对于字长是 8 位的微机，8 位二进制 D7~D0 均为数值位。其数值范围为：**00H—FFH (0-255D)**

即 8 位二进制数从全 0 到全 1，有 256 种状态。

2. 带符号数

计算机中，规定数值的**最高位为符号位**。当最高位为 “1” 时，表示该数值为负；当最高位为 “0” 时，表示该数值为正。

对于 8 位二进制数：D7=0 表示正数；D7=1 表示负数。



0.1.2 微机中数的表示

2. 带符号数

计算机中的带符号数有三种表示方法：**原码、反码、补码**。

- 对于带符号数中的**正数**，其**原码、反码、补码**的表示值是相同的。

	原码	反码	补码
+2	00000010	00000010	00000010

- 对于带符号数中的**负数**，其**原码、反码、补码**的表示值是不同的。

	原码	反码	补码
-2	10000010	11111101	11111110

反码是原码取反；补码是反码 +1，或原码取反 +1



0.1.3 二进制编码

由于计算机只能处理二进制数，所以计算机中的数字、字母、字符、汉字等都要用特定的二进制数进行编码。

1. 二—十进制数的表示（BCD 码）

用二进制表示的十进制数（Binary Coded Decimal），简称BCD码。是用4位二进制数给0-9这10个数字编码。

BCD 在微控制器中的两种存放方式：

- **1个字节存放1位BCD码，此时高半字节为0；**

如：6的BCD码为0000 0110；9的BCD码为0000 1001；

- **1个字节存放2位BCD码，称为压缩BCD码数；此时高半字节和低半字节分别存放1位BCD码；**

如：78是一个压缩BCD码，则表示为0111 1000；

59是一个压缩BCD码，则表示为0101 1001；



0.1.3 二进制编码

2. BCD 码与十进制数的转换

1 位十进制数用 4 位二进制的 BCD 码表示；4 位二进制用 1 位十进制数表示。

如：BCD 码 0101 1000 0110 的十进制数为 586

十进制数 865 的 BCD 表示 1000 0110 0101

各数制对照表

十	十六	二	BCD 码	十	十六	二	BCD 码
0	0	0000	0000	8	8	1000	1000
1	1	0001	0001	9	9	1001	1001
2	2	0010	0010	10	A	1010	0001 0000
3	3	0011	0011	11	B	1011	0001 0001
4	4	0100	0100	12	C	1100	0001 0010
5	5	0101	0101	13	D	1101	0001 0011
6	6	0110	0110	14	E	1110	0001 0100
7	7	0111	0111	15	F	1111	0001 0101

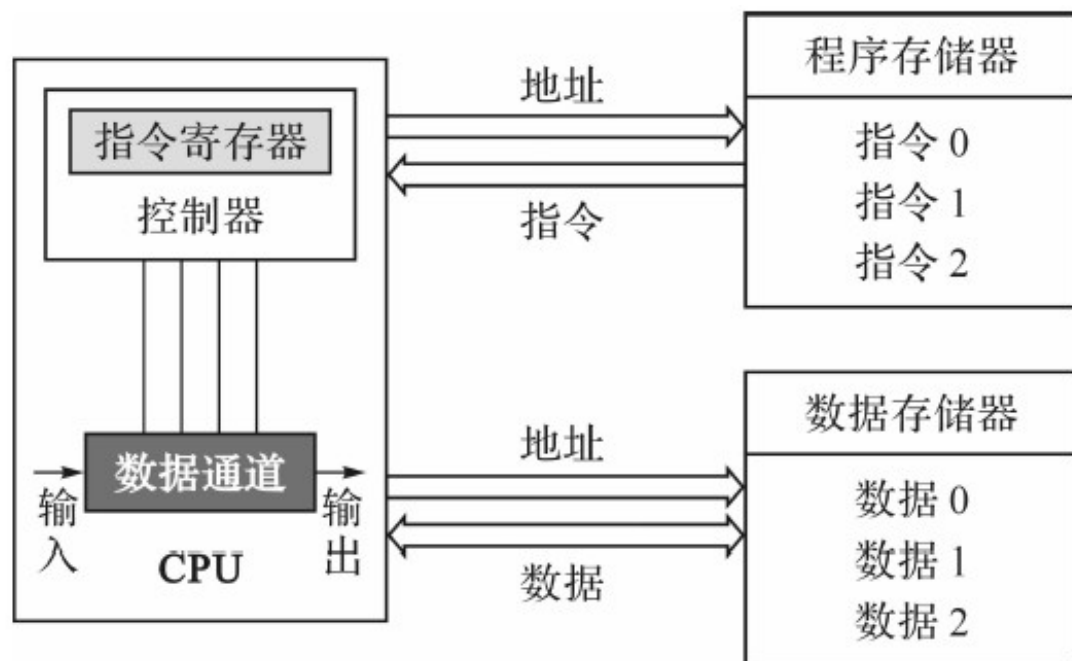


1.2.1 微控制器的存储结构

存储器的用途是用于存放程序和数据。微控制器中的存储器有两种基本结构形式。

1. 哈佛 (Harvard) 结构

程序存储和数据存储分为 2 个寻址空间，指令和数据可采用不同的数据宽度，具有较高的执行效率。**是微控制器常用的存储结构。**

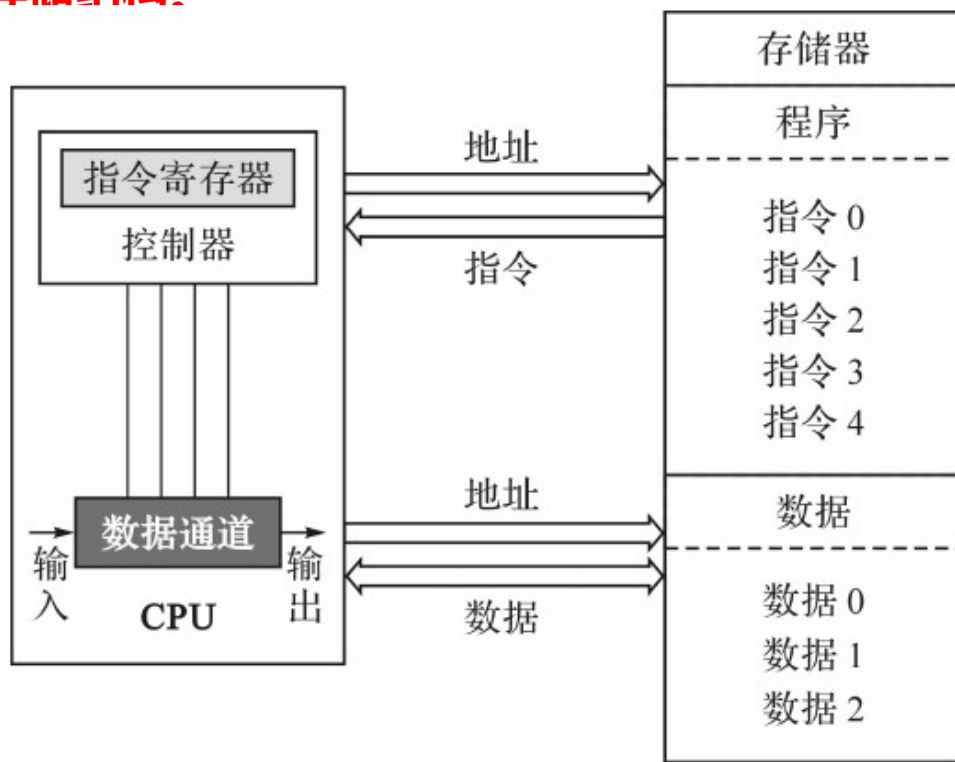




1.2.1 微控制器的存储结构

2. 普林斯顿 (Princeton) 结构

也即冯·诺依曼结构，程序存储器和数据存储器在同一个寻址空间中。ROM 和 RAM 指向同一个存储器的不同物理位置，指令和数据的宽度相同，**是通用微型计算机常用的存储结构。**





1.2.2 CISC 和 RISC 处理器

1. 两种指令集处理器

复杂指令集计算机 (Complex Instruction Set Computer , **CISC**)
和精简指令集计算机 (Reduced Instruction Set Computer , **RISC**)。

- **CISC 指令集**：CISC 的设计理念是要用最少的指令来完成所需的计算、控制任务，即要尽量简化软件。
- **优点**：寻址方式多、指令丰富，一条指令往往可以完成一串动作，且具有专用指令（如各种运算、控制转移等）；因此对于复杂计算与操作的程序设计相对容易，编程效率较高。
- **缺点**：CISC 指令的长度不同，操作进程、代码结构复杂，执行速度低；该体系的 CPU 硬件结构复杂、面积大、功耗大，对工艺要求高。



1.2.2 CISC 和 RISC 处理器

1. 两种指令集处理器

- **RISC 指令集：**RISC 的设计理念是尽可能简化指令系统，提高程序运行速度，以满足微控制器在测量与控制应用中的实时性要求。
- **优点：**单周期、定长代码的指令体系，每条指令操作少，具有归一化的指令操作进程，简单高效；每条指令归一化的取指、译码、操作、回授 4 个进程，则可实现 4 条指令相差一个进程的并行流水操作，大大提高了指令的运行速度。指令代码短、种类少、格式规范，并且 CPU 硬件结构简单、布局紧凑。
- **缺点：**对于特殊或复杂功能的程序，汇编程序设计难度增大，编程效率较低，并且一般需要较大的内存空间。

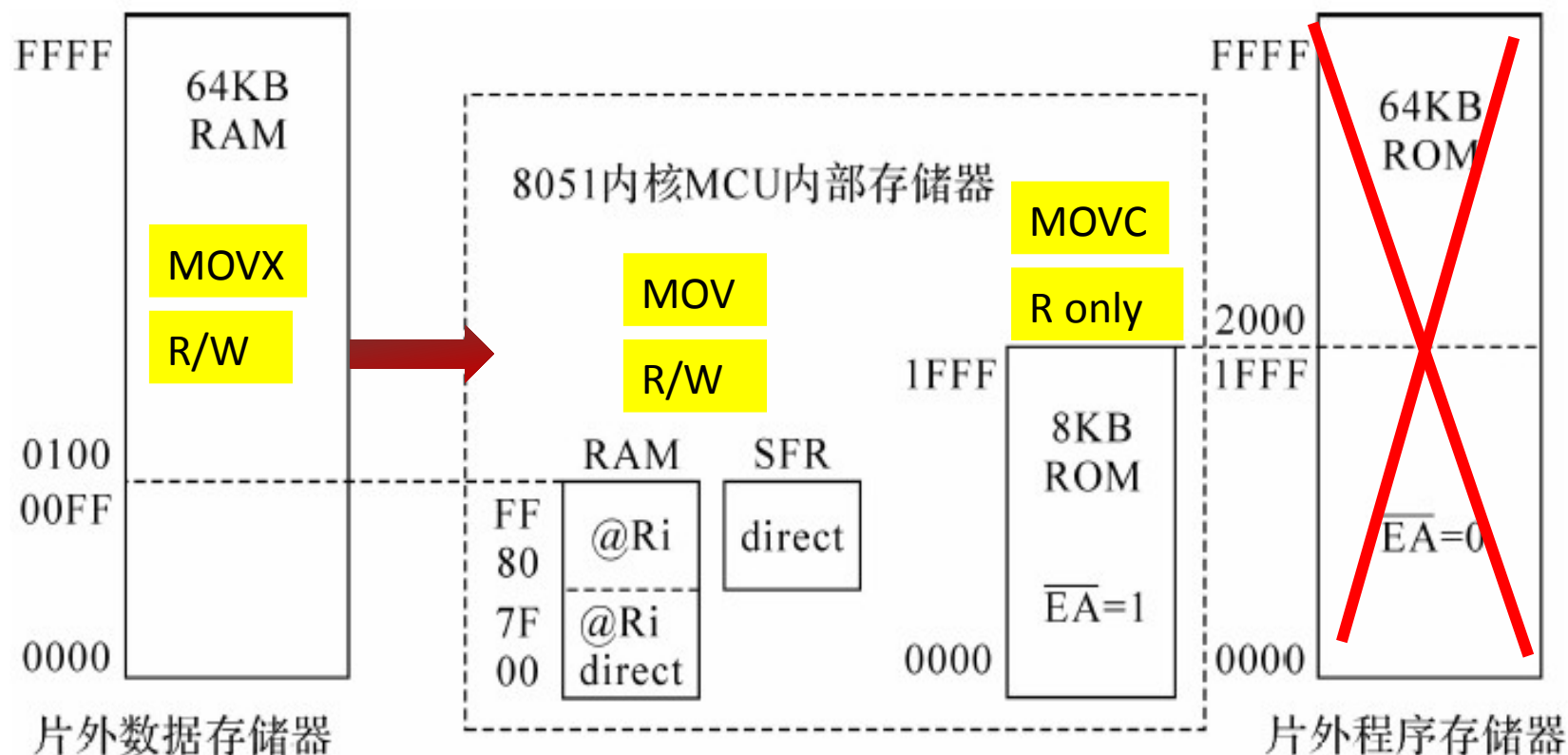
Intel CPU 全都采用 CISC 指令集，ARM 的 CPU 采用 RISC 指令集。



2.3.1 存储器配置

2. 8051 存储器结构图

8051 微控制器的存储器采用哈佛结构，ROM 和 RAM 是分开寻址





2.3.2 程序存储器 ROM

2. ROM 中的 6 个特殊单元

8051 MCU 的 ROM 中有 6 个特殊单元，是设置的特定程序入口地址，**1 个复位入口和 5 个中断入口**。所谓入口，是指一旦满足条件，PC 的值自动变为这些入口地址，则 CPU 将自动转向这些 ROM 地址取指令执行程序。

名称	入口地址	意义
复位	0000H	系统复位后 PC = 0000H
外部中断 0	0003H	外部中断 0 响应时程序转向 0003H
定时器 T0 溢出	000BH	T0 中断响应时程序转向 000BH
外部中断 1	0013H	外部中断 1 响应时程序转向 0013H
定时器 T1 溢出	001BH	T1 中断响应时程序转向 001BH
串行口中断	0023H	串行口中断响应时程序转向 0023H



2.3.3 数据存储器 RAM

2. 内部 RAM 配置

内部 RAM 中，低 128B (00H-7FH) 是基本数据存储器，可采用直接寻址、寄存器间接寻址、位寻址等多种寻址方式；高 128B (80H-FFH) 是扩展数据存储器，只能采用寄存器间接寻址方式。

内部 RAM 可划分为三块空间：

- (1) 工作寄存器区
- (2) 位寻址区
- (3) 用户 RAM 区 (包括堆栈)





2.3.3 数据存储器 RAM

(2) 位寻址区

内部 RAM 中的 20H ~ 2FH ，共 16 个单元是位寻址区。
共有 **128 位**，位地址为 **00H ~ 7FH**。

可位寻址的 16B 内部 RAM ，
既可进行字节寻址，又可进行
位寻址。

可以使用位操作指令，如
CLR ， SETB 等。

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



2.3.4 特殊功能寄存器 SFR

1. SFR 简介

特殊功能寄存器 SFR (Special Function Register) , 也称专用寄存器。

用于管理和控制 MCU 内部硬件功能模块 (如定时器 / 计数器、串行口、中断系统等) 的寄存器 , 用来存放功能模块的控制命令、状态或数据。

8051 微控制器的 SFR :

- 21 个 SFR , 离散分布于 80H ~ FFH 的专用寄存器区 , 未定义的访问无效。
- 除程序计数器 PC 指针和 R0 ~ R7 工作寄存器外 , 其余所有定义的寄存器都属 SFR 。
- 有些 SFR 可以位寻址 , 能位寻址的单元一定能字节寻址。



2.3.4 特殊功能寄存器 SFR

2. SFR 定义与分布

序 号	符 号	地 址	名称和作用		位寻址
1	B	F0H	称为 B 的一个寄存器(乘除指令中用)		✓
2	A	E0H	Accumulator	累加器	✓
3	PSW	D0H	Program Status Word	程序状态字	✓
4	IP	B8H	Interrupt Priority	中断优先级控制寄存器	✓
5	P3	B0H	Port 3	并行口 P3	✓
6	IE	A8H	Interrupt Enable	中断允许控制寄存器	✓
7	P2	A0H	Port 2	并行口 P2	✓
8	SBUF	99H	Serial Data Buffer	串行口数据寄存器	
9	SCON	98H	Serial Control	串行口控制寄存器	✓
10	P1	90H	Port 1	并行口 P1	✓
11	TH1	8DH	Timer 1 High Byte	定时器 1 高 8 位	
12	TH0	8CH	Timer 0 High Byte	定时器 0 高 8 位	
13	TL1	8BH	Timer 1 Low Byte	定时器 1 低 8 位	
14	TL0	8AH	Timer 0 Low Byte	定时器 0 低 8 位	
15	TMOD	89H	Timer Mode	定时器/计数器方式寄存器	
16	TCON	88H	Timer Control	定时器/计数器控制寄存器	✓
17	PCON	87H	Power Control	电源控制寄存器	
18	DPH	83H	Data Pointer High Byte	数据指针 DPTR 高 8 位	
19	DPL	82H	Data Pointer Low Byte	数据指针 DPTR 低 8 位	
20	SP	81H	Stack Pointer	堆栈指针	
21	P0	80H	Port 0	并行口 P0	✓

2.3.4 特殊功能寄存器 SFR

3. SFR 的位寻址空间

字节地址的低位为 0H 或 8H 的 SFR，是可位寻址的 SFR。定义了 83 位。

通用 RAM 中的位寻址区和 SFR 中的位寻址区，构成了 8051 微控制器的位寻址空间。

符 号	寄存器名	位符号地址和物理地址								字节地址
		D7	D6	D5	D4	D3	D2	D1	D0	
B	B 寄存器	F7H	F6H	F5H	F4H	F3H	F2H	F1H	F0H	F0H
		B. 7	B. 6	B. 5	B. 4	B. 3	B. 2	B. 1	B. 0	
ACC	累加器	E7H	E6H	E5H	E4H	E3H	E2H	E1H	E0H	E0H
		ACC. 7	ACC. 6	ACC. 5	ACC. 4	ACC. 3	ACC. 2	ACC. 1	ACC. 0	
PSW	程序状态字	D7H	D6H	D5H	D4H	D3H	D2H	D1H	D0H	D0H
		Cy	AC	F0	RS1	RS0	OV	F1	P	
IP	中断优先级寄存器	BFH	BEH	BDH	BCH	BBH	BAH	B9H	B8H	B8H
		—	—	—	PS	PT1	PX1	PT0	PX0	
P3	P3 口	B7H	B6H	B5H	B4H	B3H	B2H	B1H	B0H	B0H
		P3. 7	P3. 6	P3. 5	P3. 4	P3. 3	P3. 2	P3. 1	P3. 0	
IE	中断允许寄存器	AFH	AEH	ADH	ACH	ABH	AAH	A9H	A8H	A8H
		EA	—	—	ES	ET1	EX1	ET0	EX0	
P2	P2 口	A7H	A6H	A5H	A4H	A3H	A2H	A1H	A0H	A0H
		P2. 7	P2. 6	P2. 5	P2. 4	P2. 3	P2. 2	P2. 1	P2. 0	
SCON	串行口控制寄存器	9FH	9EH	9EH	9CH	9BH	9AH	99H	98H	98H
		SM0	SM1	SM2	REN	TB8	RB8	TI	RI	
P1	P1 口	97H	96H	95H	94H	93H	92H	91H	90H	90H
		P1. 7	P1. 6	P1. 5	P1. 4	P1. 3	P1. 2	P1. 1	P1. 0	
TCON	定时器控制寄存器	8FH	8EH	8DH	8CH	8BH	8AH	89H	88H	88H
		TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	
P0	P0 口	87H	86H	85H	84H	83H	82H	81H	80H	80H
		P0. 7	P0. 6	P0. 5	P0. 4	P0. 3	P0. 2	P0. 1	P0. 0	



2.3.4 特殊功能寄存器 SFR

4. 程序计数器 PC

也称为程序指针或 PC 指针，具有如下特点：

- PC 是一个 16 位的专用寄存器，存放 ROM 的地址，因此称为程序指针，其寻址范围为 0 ~ 64KB。
- PC 存放的是下一条要执行的指令地址。复位后 PC 的内容为 0000H，表示 CPU 将从 ROM 的 0000H 单元取指令执行。也即 PC 指向哪里，CPU 就从其指向的 ROM 单元取指令执行程序。
- PC 不属于特殊功能寄存器。因此不占用 SFR 地址空间，是不可寻址的，在程序中不能直接访问。
- PC 可以通过 LJMP、SJMP 等转移指令来间接修改 PC 的值。



2.3.4 特殊功能寄存器 SFR

5. 特殊功能寄存器介绍

(1) 累加器 A

累加器 A (或 ACC) 是 CPU 中使用最频繁 8 位专用寄存器。在算术、逻辑类操作时, ALU 的一个输入来自 A, 运算结果也大多保存于 A。

A 的字节地址是 E0H, 可位寻址。

位地址	E7	E6	E5	E4	E3	E2	E1	E0
位符号	Acc.7	Acc.6	Acc.5	Acc.4	Acc.3	Acc.2	Acc.1	Acc.0



2.3.4 特殊功能寄存器 SFR

5. 特殊功能寄存器介绍

(2) B 寄存器

B 寄存器是一个 8 位寄存器，一般用于乘除指令中：

MUL AB ; $A*B=BA$;

DIV AB ; $A/B=$ 商 A..... 余数 B

B 的字节地址是 F0H ；可位寻址，位地址为 F0H ~ F7H。

在其它情况下，B 寄存器可以作为内部 RAM 中的一个单元来使用。



2.3.4 特殊功能寄存器 SFR

5. 特殊功能寄存器介绍

(3) 程序状态字 PSW (Program Status Word)

PSW 用来存放程序状态信息，表征指令执行后的状态，供程序查询和判别之用。字节地址： D0H ； 位地址为： D0H-D7H

位地址	D7	D6	D5	D4	D3	D2	D1	D0
位符号	Cy	AC	F0	RS1	RS0	OV	F1	P
注释	Carry	Assistant Carry	Flag 0	Register bank Selector bit 1	Register bank Selector bit0	Overflow	Flag 1	Parity Flag

对于 **C**、**AC**、**OV**、**P**，根据指令执行结果，由硬件置位或清 0，称为状态位。对于 **RS1**、**RS0**、**F1**、**F0**，根据需要使用，由指令设定，称为控制位。
RS1、RS0 为工作寄存器组选择位，F1、F0 由用户自定义使用。



2.3.4 特殊功能寄存器 SFR

(3) 程序状态字 PSW (C、AC、F0、F1、RS1、RS0、OV、P)

- **C (CY)** : 进位标志。在加、减法运算时, 若高位 (D7) 发生进位或借位则被置 1 (即 $C=1$), 否则被清 0 ($C=0$)。
- **AC** : 辅助进位标志。在加、减法运算时, 若低 4 位向高 4 位发生进位或借位则 $AC=1$, 否则 $AC=0$; AC 标志在十进制调整指令 DA A 中要用到。
- **F0、F1** : 软件标志。由软件置位或复位, 由用户定义使用。
- **RS1、RS0** : 工作寄存器组选择位。由软件置位或复位, 用来选择 4 组工作寄存器中的一组。



2.3.4 特殊功能寄存器 SFR

(3) 程序状态字 PSW (C、AC、F0、F1、RS1、RS0、OV、P)

➤ **OV** : 溢出标志。对于带符号数而言，反映运算结果是否溢出。

- **OV=1** : 溢出，表示运算结果超出了 A 所能表示的带符号数的范围 (-128 ~ +127 ，即 8 位带符号数的范围)。
- **OV=0** : 没有溢出。
- 对于乘法 MUL ，当 A、B 两个乘数的积超过 255 时 OV 置位；否则，OV = 0。
- 对于除法 DIV ，若除数为 0 时，OV=1 ；否则，OV=0。



2.3.4 特殊功能寄存器 SFR

(3) 程序状态字 PSW (C、AC、F0、F1、RS1、RS0、OV、
判断 OV 的 2 种方法 :

方法 1 : 对于加法运算 , 当 D6、D7 均向或均不向其高位 (D7、Cy) 进位 , 则 $OV=0$, 否则 $OV=1$; 对于减法运算 , 当 D6、D7 均向或均不向其高位 (D7、Cy) 借位 , 则 $OV=0$, 否则 $OV=1$ 。

也即 , 下述情况 $OV = 1$ (表示带符号数运算结果错误) :

- 当 D6 向 D7 有进位 (借位) , 而 D7 不向 Cy 进位 (借位) 时 ;
- 当 D7 向 Cy 进位 (借位) , 而 D6 不向 D7 进位 (借位) 时。

其它情况下 $OV = 0$ (表示带符号数运算结果正确) 。



2.3.4 特殊功能寄存器 SFR

(3) 程序状态字 PSW (C、AC、F0、F1、RS1、RS0、OV、P)

方法 2 : 当加法或减法的运算结果超出 $-128 \sim +127$ 这个范围时, $OV=1$, 否则 $OV=0$ 。

以下 2 种情况表示发生了溢出 :

- 两个正数相加, 结果变成负数 ;
- 两个负数相加, 结果变成正数。



2.3.4 特殊功能寄存器 SFR

(3) 程序状态字 PSW (C、AC、F0、F1、RS1、RS0、OV、P)

- **P** : 奇偶标志。用以表示指令操作之后，累加器 A 中 1 的个数的奇偶性。
- 若 A 中 “1” 的个数为奇数个，则 $P=1$ ；
 - 若 A 中 “1” 的个数为偶数个，则 $P=0$ ；

只要 A 中的数据变化，P 的状态亦随之变化。如：

MOV A, #35H; 则 $P=0$;

ADD A, #14H; 则 $(A) = 49H$, $P=1$





2.3.4 特殊功能寄存器 SFR

5. 特殊功能寄存器介绍

(4) 堆栈指针 SP (Stack Pointer)

➤ 堆栈的概念：

堆栈是定义为特殊用途的存储区，主要功能是临时存放数据和地址，通常用于保护断点和保护现场。堆栈有二种形式，一是向上（向高地址）生成，二是向下（向低地址）生成。

8051 MCU 的堆栈为**满顶法向上生成**的软件堆栈，其堆栈区必须开辟在内部通用 RAM 中。

堆栈按照“先进后出”即“后进先出”的原则存取数据，最后进栈的数据最先被弹出。压入堆栈的数据总是保存在堆栈的顶部，从堆栈弹出的总是栈顶的数据。



2.3.4 特殊功能寄存器 SFR

(4) 堆栈指针 SP

➤ 堆栈指针 SP (Stack Pointer) :

存放堆栈栈顶地址的一个 8 位寄存器，地址为 81H。8051 MCU 的堆栈是向上生成的：进栈时栈顶向高地址生长，SP 的内容增加；出栈时栈顶向下回落，SP 的内容减少。所以 SP 总是指向堆栈的栈顶。

➤ 堆栈的设置：

8051 微控制器的堆栈必须设在内部通用 RAM 中。MCU 复位后 SP 内容为 07H，即默认堆栈从 08H 单元开始。

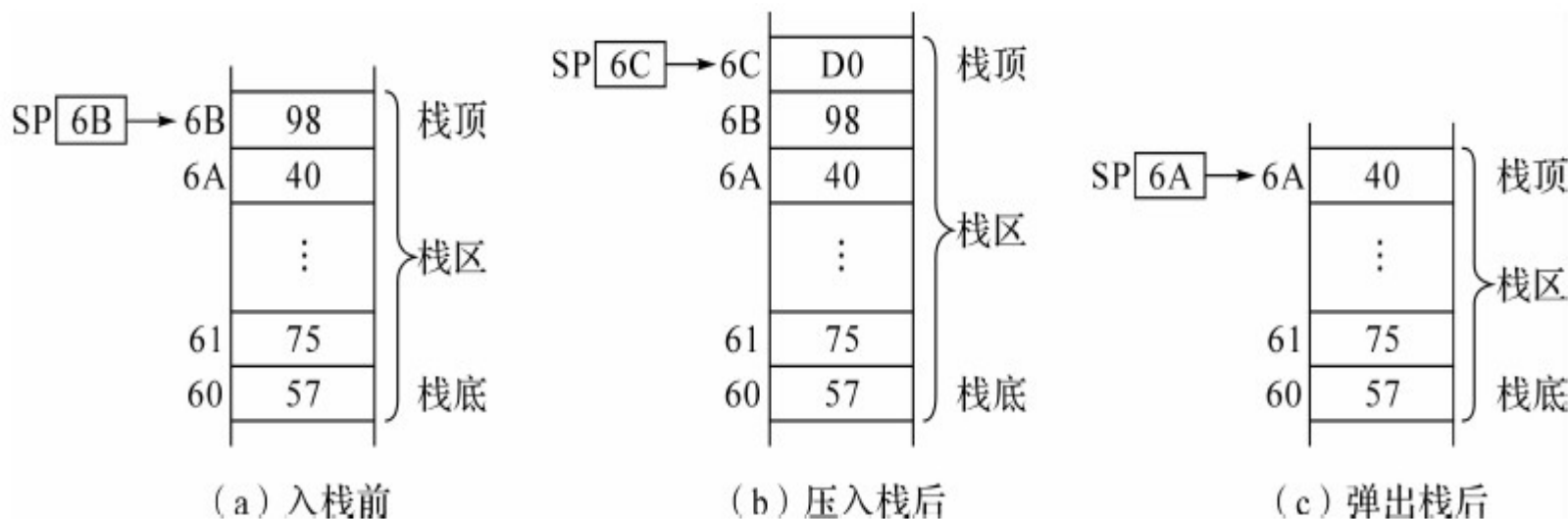
➤ 堆栈的操作方式：

- 指令方式：PUSH, POP
- 自动方式：在调用子程序或发生中断时。

2.3.4 特殊功能寄存器 SFR

(4) 堆栈指针 SP

➤ 堆栈的操作过程：



栈底 = 60H (第一个进栈数据存放单元) , 压入数据 , 堆栈向上生长 , 图示栈顶是 6BH , (SP) = 6BH , 即 SP 指向栈顶。

再压入 1 个数据 D0H , 压入 6CH 单元 , 栈顶变为 6CH , (SP) = 6CH 。

弹出 2 个数据 , 栈顶变为 6AH , (SP) 也变为 6AH 。



2.3.4 特殊功能寄存器 SFR

5. 特殊功能寄存器介绍

(5) 数据指针 DPTR (Data Pointer)

数据指针 DPTR 是一个 16 位的 SFR。其功能是外部 RAM（地址范围 0000H-FFFFH）的地址指针，是存放外部 RAM 地址的 16 位寄存器。

DPTR 由两个 8 位寄存器组成，高 8 位为 DPH 表示，低 8 位用 DPL。

DPH 地址：83H；DPL 地址：82H



2.3.4 特殊功能寄存器 SFR

5. 特殊功能寄存器介绍

(6) P0-P3 端口寄存器

P0、P1、P2、P3：分别是 I/O 端口 P0 ~ P3 的锁存器，地址分别为：80H、90H、A0H、B0H，可以位寻址。

对于端口即引脚的操作实际上是对这些寄存器的操作，其端口引脚与端口寄存器的位具有映射关系。

(7) 其它特殊功能寄存器

SBUF、IP、IE、TMOD、TCON、SCON、PCON 等，将结合相关章节内容进行介绍。

微控制器复位后，除 SP 为 07H，P0~P3 为 FFH 外，其余 SFR

均为 0。



2.3.4 特殊功能寄存器 SFR

6. 内部 RAM 和 SFR 的寻址方式

- **对于 00H-7FH 存储空间**：可运用直接寻址和寄存器间接寻址这两种寻址方式，对其进行访问。
- **对于 80H-FFH 存储空间**：只能采用寄存器间接寻址方式进行访问。
- **对于地址范围同为 80H-FFH 的特殊功能寄存器 (SFR)**：只能采用直接寻址方式。

解决 2 个存储空间的地址重叠问题，避免存储单元访问的冲突，常用的办法是：采用不同的寻址方式。



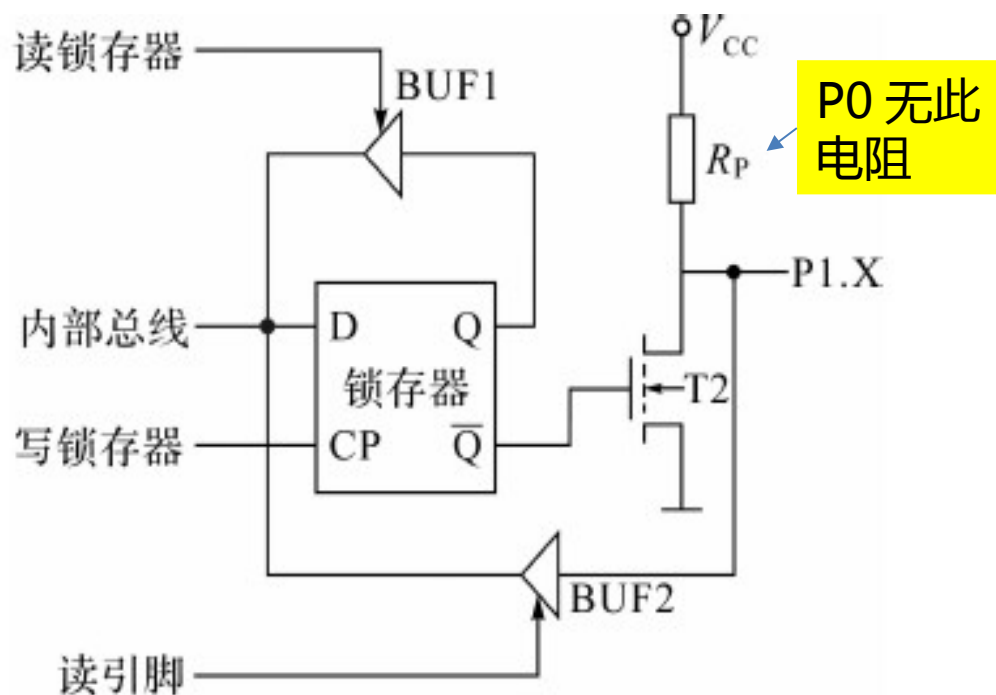
2.4.1 P0-P3 端口内部结构

端口内部结构：准双向 I/O 口结构

P0-P3 端口的每一位，均有一个输出锁存 D 触发器、输出驱动电路组成；以及两个分别用于读锁存器数据和读引脚的三态输入缓冲器 BUF1 和 BUF2。

输出时：当 CPU 通过内部总线向端口锁存器输出 1 或 0 时，通过输出驱动电路，端口相应引脚就会输出高电平或低电平。

输入时：即读引脚时，如果端口锁存器状态为 0，则 T2 导通，引脚被钳位在“0”状态，导致无法得到端口引脚的高电平状态。（**准双向口的特点**）



端口内部结构



2.4.3 P0-P3 端口的结构特点和应用特性

P0-P3 端口既有共同点，也有差异点。使用时，应根据各端口的特点正确使用。

1. 准双向 I/O 口特性

- P0-P3 端口均是准双向口，准双向口的输入操作和输出操作本质不同，输入操作是读引脚状态；输出操作是对口锁存器的写入操作。
- 在输出时，与真正双向口一样，CPU 输出的“0”或“1”，通过输出驱动电路在引脚上表现为低电平或高电平。



2.4.3 P0-P3 端口的结构特点和应用特性

1. 准双向 I/O 口特性

➤ 在输入时，必须先向锁存器输出 1，使得输出驱动电路中的 T2 处于截止状态，即将端口设置为输入方式。只有这样外部引脚的高、低电平状态才能被正确读入；反之若 T2 处于导通状态，则端口的引脚被钳位在 0 电平，而无法得到引脚所接外设的真实状态。

• **准双向口的输入操作**：先向锁存器输出 1，然后再输入引脚状态。

例：要将 P1 口状态读入 A 中，应执行以下两条指令

MOV P1, #0FFH ; P1 口置为输入方式

MOV A, P1 ; 读 P1 口引脚状态



2.5.1 时钟电路与时序

2. 时序与工作周期

微控制器的时序就是 CPU 和功能模块工作时，各控制信号之间的时间顺序关系。微控制器的内部电路在时钟信号控制下，严格按时序执行指令规定的操作。

8051 微控制器规定了几种工作周期：

- **时钟周期（振荡周期）**
- **状态周期**
- **机器周期**
- **指令周期**



2.5.1 时钟电路与时序

3. 工作周期（时钟周期、状态周期、机器周期、指令周期）

➤ 时钟周期 T_0

也称为振荡周期，是外接晶振频率的倒数；是微控制器中最基本、最小的时间单位。若振荡源频率为 f_{osc} ，则振荡周期为 $T_0 = 1/f_{osc}$ ；一个振荡周期定义为一个节拍 P 。

若晶振频率 f_{osc} 为 6MHz，则时钟周期为 $1/f_{osc}$ 即 $1/6 \mu s$ ；若晶振频率为 f_{osc} 为 12MHz，则时钟周期为 $1/f_{osc}$ 即 $1/12 \mu s$ 。

- 外接晶振频率的高低，决定了 CPU 执行速度的快慢。
- 但振荡频率太高，会引入高频辐射。
- 不同型号 MCU 有不同的时钟频率范围要求。



2.5.1 时钟电路与时序

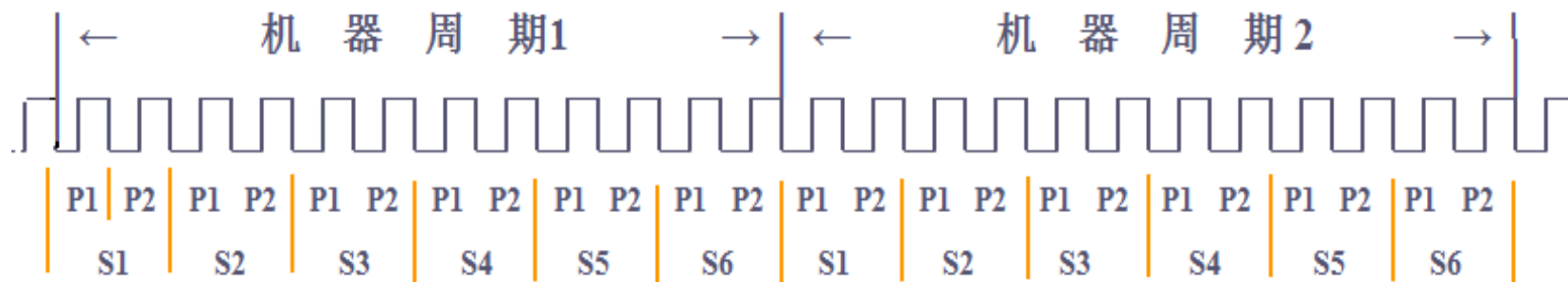
3. 工作周期 (时钟周期、状态周期、机器周期、指令周期)

➤ 状态周期 S

是时钟周期的两倍 $S = 2T = 2 / f_{osc}$, 即由连续的两个节拍 P1 和 P2 组成。

➤ 机器周期 T_M

CPU 执行一个基本操作所需要的时间。一个机器周期由 6 个状态周期 (S1-S6) 即 12 个时钟周期组成。



1 个机器周期 $T_M = 6$ 个状态周期 $S = 12$ 个振荡周期 T_0



2.5.1 时钟电路与时序

3. 工作周期 (时钟周期、状态周期、机器周期、指令周期)

➤ 指令周期

执行一条指令所需要的时间，为指令周期。

通常每条指令的执行可划分为 1-4 个基本操作，完成一个基本操作所需时间为机器周期。所以指令周期由 1-4 个机器周期组成。



2.5.2 复位与复位电路

2. 复位状态

复位后，8051 微控制器的初始化状态为：

- (PC) = 0000H ;
- (SP) = 07H ;
- (PSW) = 00H ;
- P0 ~ P3 端口的锁存器为 FFH，端口引脚全为 1，处于可输入状态；
- 除上述 SFR 外，其余特殊功能寄存器 SFR 均为 0；

RAM 数据在热复位后保持不变，在上电复位后为随机数。





3 .1.1 指令系统概述

微控制器具有的**指令集合**即为该微控制器的指令系统（或指令集），指令系统中的各条指令对应不同的机器代码，是由微控制器内核设计人员确定的。

1. 指令分类

8051 微控制器采用 CISC 结构的指令集，共有 111 条指令。可根据指令的长度（即指令机器码的字节数）、指令的执行速度（即指令的机器周期数）和指令的功能进行分类。



3.1.1 指令系统概述

1. 指令分类

(1) 按指令的长度(字节数)分类：

- **单字节指令 49 条**：1 字节机器码，指令的操作数隐含在操作码中。

如 INC Rn 和 MOV A, Rn 的机器码分别为 00001xxx 和 11001xxx，其中 xxx 可表示的 000-111，分别代表 R0~R7，实际上各有 8 条指令。

- **双字节指令 46 条**：机器码有 2 个字节：第 1 字节为操作码，第 2 字节为操作数。

如 MOV A, #data；**机器码**：74H, data；

- **三字节指令 16 条**：机器码有 3 个字节：第 1 字节为操作码，第 2、3 字节为操作数。

如 MOV DPTR, #data16；**机器码**：90H, #data 的高 8 位, #data 的低 8 位



3.1.1 指令系统概述

1. 指令分类

(2) 根据指令的执行时间(速度)分：

- 单机器周期指令：64 条
- 双机器周期指令：45 条
- 四机器周期指令：2 条

(3) 根据指令功能划分：

- 数据传送类指令：29 条
- 算术运算类指令：24 条
- 逻辑运算类指令：24 条
- 控制转移类指令：17 条
- 位操作类指令：17 条



3.1.1 指令系统概述

2. 指令格式

指令的表示方式称为指令格式，任一条指令均由操作码和操作数两部分组成。操作码用来规定指令所要完成的操作；操作数是指令操作的对象。

指令的典型格式：

标号： **助记符** **目的操作数，源操作数** **；注释**

如： **LOOP：** **ADD** **A, Rn** **；加法**

- **标号：**是该指令的符号地址，根据需要设置。标号的第一个字符必须是字母，其余可以是符号或数字。
- **助记符：**表述指令的功能，规定执行某种操作；助记符用英文名称或缩写表示。
- **操作数：**是指令操作的对象，可以是具体数据、数据保存的地址、寄存器或标号等。对于有两个操作数的指令，左边的为目的操作数，右边的为源操作数。
- **注释：**是对该指令在程序中作用的说明，帮助阅读、理解和使用源程序。

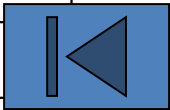


3 .1.2 寻址方式

8. 寻址方式和寻址空间

指令的寻址方式通常是指源操作数的寻址方式。 8051 指令系统源操作数的 7 种寻址方式可使用的变量与可寻址的空间有所不同， 概括如下。

寻址方式	使用的变量	寻址空间
直接寻址	direct	内 部 RAM 低 128 字
寄存器寻址	R0~R7、 A	内 部 RAM R A
寄 存 器 间 接 寻址	@R0~R1	内部 RAM 的 256 字节
	@R0~R1、 @DPTR	外部 RAM
立即寻址	#data、 #data16	程序存储器
变址寻址	基址寄存器： DPTR、 PC ； 变址寄存器： A ；	程序存储器
相对寻址	PC+ 偏移量	程序存储器
位寻址	bit、 C	位寻址空间





3.2.1 数据传送类指令

数据传送类指令是最基本、使用最多的一类指令，共有 29 条。

主要用于数据的传送、保存以及交换等场合。

该类指令中，除给 A 赋值（A 内容发生变化）会影响 P 标志外，其余标志不受影响。可分为以下五组：

- 内部 RAM 数据传送类指令： 16 条（MOV）
- 程序存储器访问类指令： 2 条（MOVC）
- 外部 RAM 访问类指令： 4 条（MOVSX）
- 堆栈操作类指令： 2 条（PUSH POP）
- 数据交换类指令： 5 条



3.2.2 算术运算类指令

算术运算指令是通过**算术逻辑运算单元 ALU** 进行数据运算与处理的指令，主要完成加、减、乘、除四则运算，以及加 1、减 1、BCD 码运算和调整等。除加 1、减 1 运算外，这类指令大多数要影响 PSW 中的标志位。24 条指令可分为 **6 组**。

- | | | |
|--------------------|-----|-----------|
| ➤ 不带进位加法指令： | 4 条 | 影响 PSW ! |
| ➤ 带进位加法指令： | 4 条 | 影响 PSW ! |
| ➤ 带借位减法指令： | 4 条 | 影响 PSW ! |
| ➤ 加 1 指令： | 5 条 | 不影响 PSW ! |
| ➤ 减 1 指令： | 4 条 | 不影响 PSW ! |
| ➤ 乘法、除法、十进制运算调整指令： | 3 条 | 影响 PSW ! |



3.2.3 逻辑操作类指令

逻辑操作类指令包含**逻辑与、逻辑或、逻辑异或、求反、左右移位、清 0**等。该类指令不影响标志位，仅当其目的操作数为 A 时，对奇偶标志位 P 有影响。24 条指令可分为 **5 组**。

- 逻辑与操作指令：6 条
- 逻辑或操作指令：6 条
- 逻辑异或指令：6 条
- 累加器清 0 和取反指令：2 条
- 循环移位指令：4 条



3.2.4 控制转移类指令

程序的顺序执行是由程序计数器（PC）自动增1来实现的，要改变程序的执行顺序，控制程序的流向，必须通过控制转移类指令实现，所控制的范围为程序存储器的64KB空间。8051MCU的控制转移类指令，共17条，可分为4组。

- 无条件转移指令：4条
- 条件转移指令：8条
- 子程序调用和返回指令：4条
- 空操作指令：1条



3.2.5 位操作类指令

8051MCU 具有一个功能齐全的位处理器，进位标志 **Cy** 为位累加位，可以执行多种“位”操作。所有位操作均采用直接寻址方式，寻址位空间为 8051MCU 的位寻址空间。17 条位操作类指令可分为 **4 组**。

- 位数据传送指令：2 条
- 位状态设置指令：6 条
- 位逻辑运算指令：4 条
- 位转移指令：5 条



3.4.2 汇编语言编程风格

完成同一个任务所使用的方法或程序不是唯一的。程序设计的质量将直接影响到计算机系统的工作效率、运行可靠性。良好的编程风格对于实现高质量的程序具有重要意义。

汇编程序设计时，应关注以下几点。

1. 注释

- 注释是程序设计中的重要内容之一，汇编指令的抽象特性更要重视注释的作用。
- 注释内容用 “;” 与助记符指令隔离，注释内容长度不限，换行时，头部仍要标注 “;”。

2. 标号的使用

- 标号由不多于 8 个 ASCII 字符组成，第一个字符必须是字母，不能使用汇编语言已定义的符号，如助记符，寄存器名等。
- 同一个标号在一个独立的程序中只能定义一次。
- 标号通常代表地址，标号名的选取应能够表达所代表的目的地址。



3.4.3 汇编程序中的伪指令

1. 起始汇编 **ORG** (**Origin**) 格式 : **ORG** **nn**

2. 赋值 **EQU** (**Equal**) 格式 : 字符名称 **EQU** 数据或表达式

例 : DATA1 EQU 22H
3. 定义字节 **DB** (**Define Byte**) 格式 : 标号 : **DB** 字节常数或字符串

例 : **ORG** 2000H

TABLE : **DB** 73H,04,100,32,00,-2,"ABC" ;
4. 定义字 **DW** (**Define Word**) 格式 : 标号 : **DW** 字或字串

例 : **ORG** 1000H

TABLE : **DW** 100H , 3456H , 1357H ,



3.4.3 汇编程序中的伪指令

5. 位地址赋值 BIT 格式：字符名称 BIT 位地址

➤ 例：LED1 BIT P3.1 ； LED1 赋值为 P3.1 ，在整个程序中，LED1 即为 P3.1 。

SETB LED1 ； 设置 P3.1 为高电平

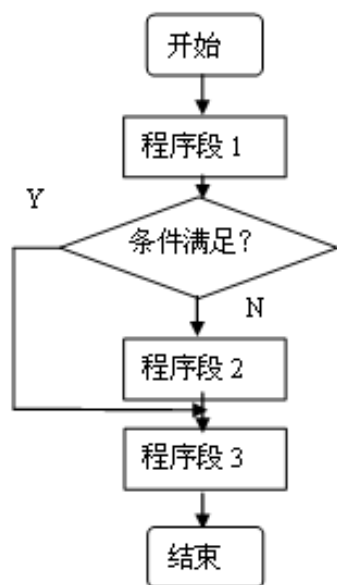
6. 结束汇编 END 格式：END

7. 定义存储器空间 DS (Define Storage) 格式：标号：DS nn

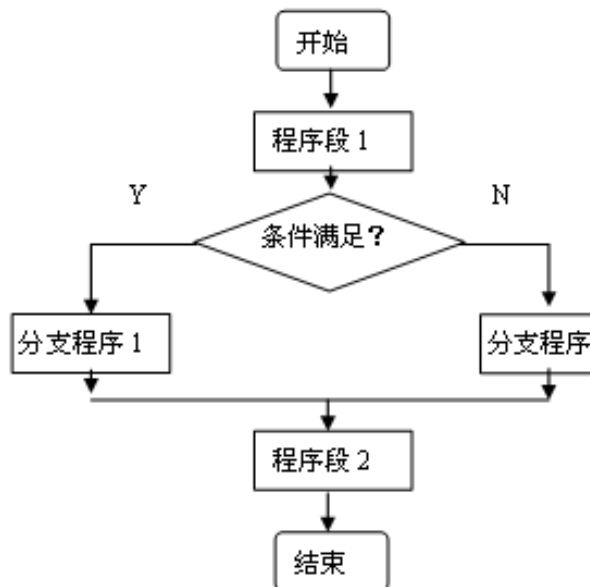
例：BASE: DS 100H

- 对于 8051 微控制器，DB、DW、DS 等伪指令是应用于 ROM，而不能用于 RAM。

3.5.1 程序设计的结构化

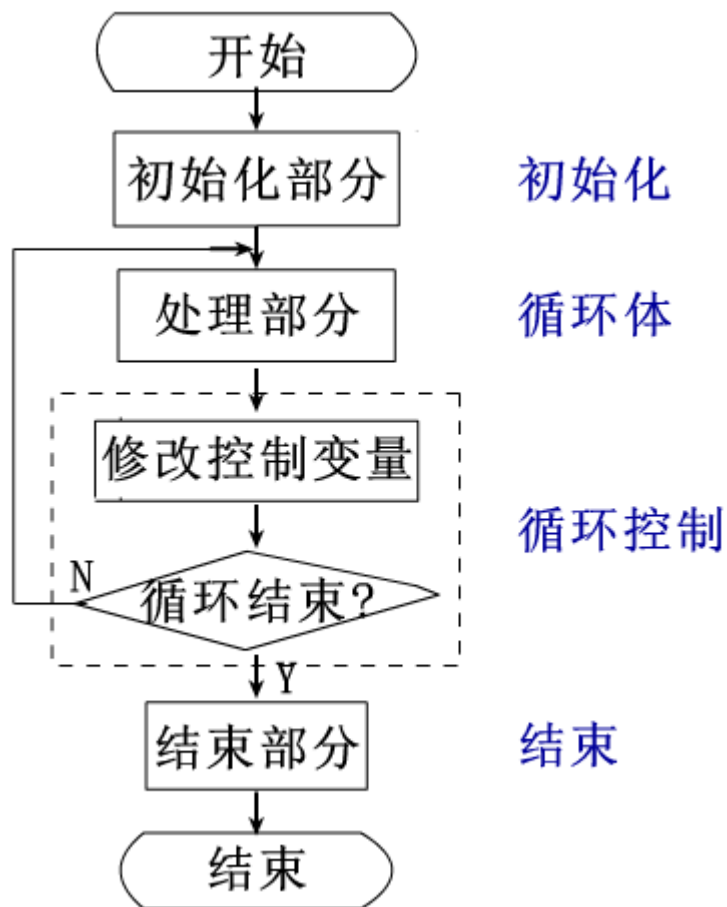


(1)



(2)

分支结构



循环结构



4.2.1 数据类型

1. C51 支持的数据类型

数据类型	位数	字节数	值域
[signed] char	8	1	-128 to +127
unsigned char	8	1	0 to 255
[signed] int	16	2	-32768 to +32767
unsigned int	16	2	0 to 65535
[signed] long	32	4	-2147483648 to +2147483647
unsigned long	32	4	0 to 4294967295
float	32	4	$\pm 1.175494\text{E}-38$ to $\pm 3.402823\text{E}+38$



4.2.1 数据类型

2. C51 扩展的数据类型

数据类型	位数	字节数	值域
bit	1		0 to 1
sbit	1		0 to 1
sfr	8	1	0 to 255
sfr16	16	2	0 to 65535

sbit sfr sfr16

➤ **功能：**其所声明的位变量是位于 SFR 区的可位寻址空间。

例： sbit P10=P1^0; // 定义位变量 P10 为 P1.0

sfr P0=0x80; // 定义 P0 口的地址为 80H

sfr16 DPTR=0x82; // 指定 DPTR 地址，

~~DPL=0x82, DPH=0x83~~
4.2 C51 基础



4.2.2 存储器类型与存储模式

1. C51 数据的存储器类型

是指 C51 变量的存储空间。存储器类型可由关键字直接声明。

存储器类型	关键字	存储空间描述
程序存储器	code	程序存储器 (ROM) 空间 64 KB
内部数据存储器	data	直接寻址的内部 RAM 空间 128B (00H-7FH) , 访问速度最快
	idata	间接访问的内部 RAM 空间 256B (00H-FFH)
	bdata	可位寻址的内部 RAM 空间 16B (20H-2FH)
外部数据存储器	xdata	外部 RAM 空间 64 KB
	pdata	分页的外部 RAM 256 B/ 每页



4.2.2 存储器类型与存储模式

2. 存储模式

➤ **变量定义格式**：**数据类型** [**存储器类型**] **变量名**

(数据类型：指变量的数据类型；存储器类型：明确变量存放的存储器空间，可缺省。
如未声明，则变量的存储器类型由 C51 编译器决定)

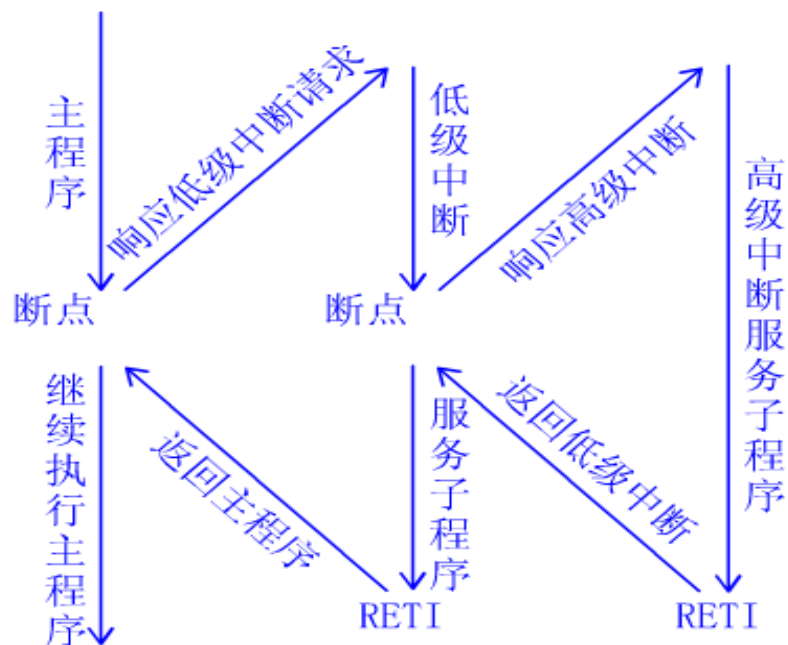
```
char code text[] = "enter password";  
char data var1;  
float idata x,y,z;  
char bdata flags;  
unsigned long xdata array[100];  
unsigned int pdata dimension;  
float x,y,z; // 未包括存储器类型  
unsigned long array[100]; // 未包括存储器类型
```

默认的存储器类型由编译器的存储模式 SMALL、COMPACT 及
LARGE 决定。



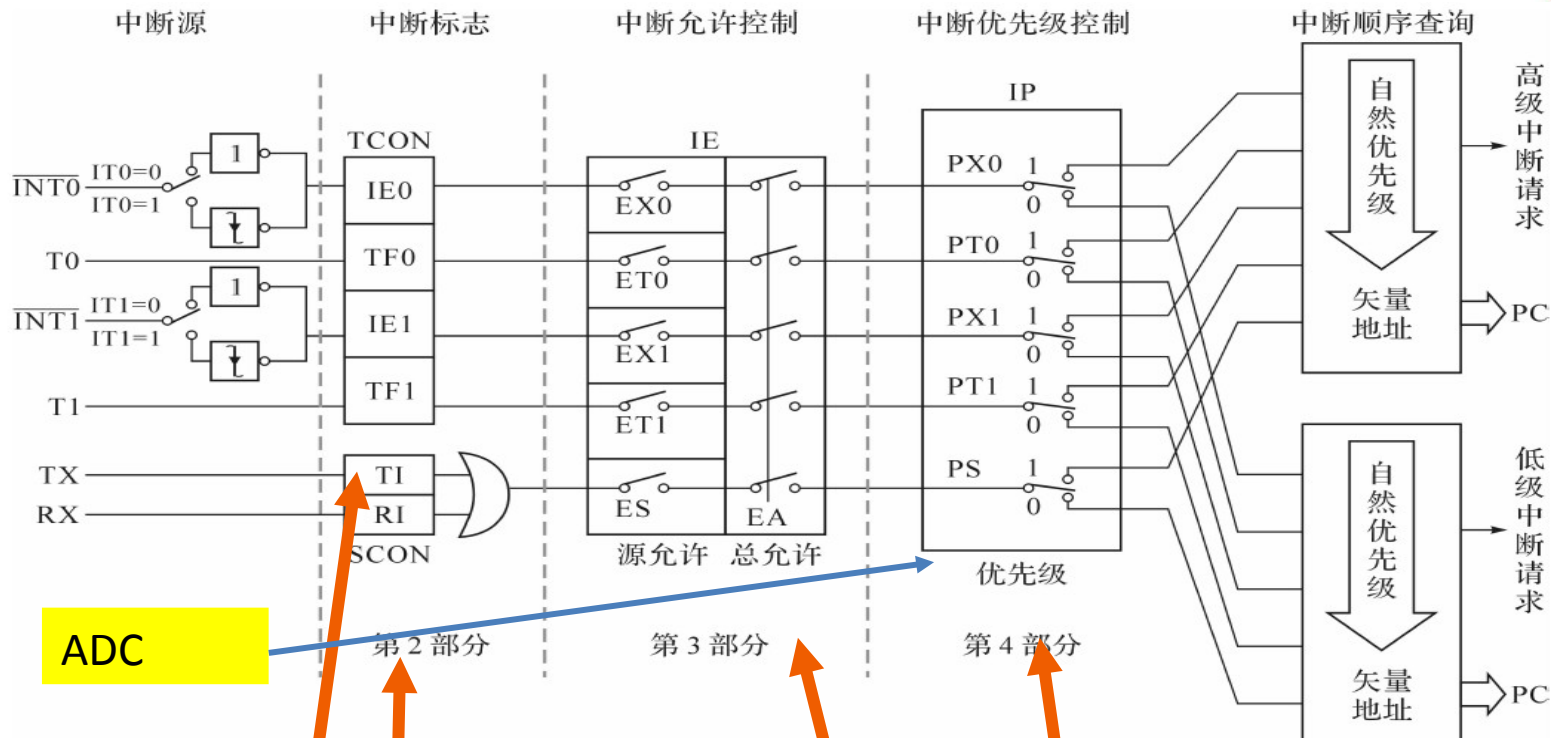
5.1.1 中断的概念

中断：微控制器执行程序过程中，由于内部或者外部的某种原因，要求 MCU 中止正在运行的程序，转去执行另外一段处理程序，待处理结束后，再回来继续执行被中止了的原程序。这种程序在执行过程中，由于外界的原因而被中间打断的情况称为“中断”。





51 中断系统总结



TCON 88H	7	6	5	4	3	2	1	0
位符号	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

SCON 98H	7	6	5	4	3	2	1	0
位符号	SM0	SM1	SM2	REN	TB8	RB8	TI	RI

ADC_CON TR BCH	7	6	5	4	3	2	1	0
位符号	-	-	-	ADC_FLAG	-	-	-	-

IP A8H	7	6	5	4	3	2	1	0
位符号	EA	-	EADC	ES	ET1	EX1	ET0	EX0

IP B8H	7	6	5	4	3	2	1	0
位符号	-	-	PADC	PS	PT1	PX1	PT0	PX0

— TI, RI, ADC_FLAG 不会硬件自动清零



5.3.1 中断响应的自主操作过程

2. 中断响应时的自主操作

- 置位内部相应的“优先级标志”，以标明所响应中断的优先级别；
- 中断源标志清 0（**TI、RI 除外**）；
- 中断断点地址装入堆栈保护（不保护寄存器等）；
- 中断入口地址装入 PC，以便使程序转到中断入口地址处。

3. 中断返回时的自主操作

- 内部“优先级标志”清 0；
- 断点地址送入 PC，以便使程序返回到断点处。



5.3.2 中断响应条件

中断响应：中断源发出中断请求、在满足 CPU 中断响应条件之后，CPU 处理中断请求的过程。

中断响应的基本条件：

- 1) 有中断源发出中断请求；
- 2) CPU 中断允许位（总允许）置位，即 $EA = 1$ ；
- 3) 申请中断的中断源的中断允许位（源允许）置位，即允许该中断源中断。

CPU 在每个机器周期 (S6 状态)，按优先次序查询各中断标志，找到所有有效的中断请求，并对其优先级排队，如果满足：

- 4) 无同级或高级中断正在服务；
- 5) 现行指令已执行完毕；
- 6) 若执行指令为 RETI 或是读 / 写 IE 或 IP 指令时，则该指令的下一条指令也执行完毕。

CPU 便在紧接着的下一个机器周期响应中断，否则将丢弃中断查询结果。



4.2.5 函数

2. 中断函数

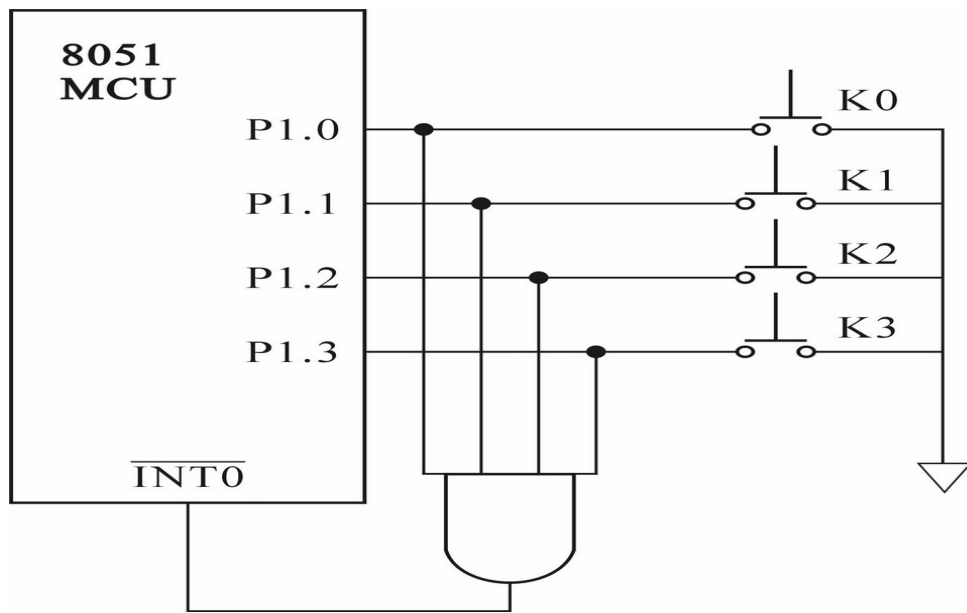
C51 处理中断的方法为在中断发生时，自动调用中断函数，执行完毕后，自动返回主程序。

➤ **定义：** `[void] 中断函数名 () interrupt 中断号 [using n]`

- **Interrupt** : 中断函数的关键字；
- **中断号** : 取值范围为 0 ~ 31，分别对应不同的中断函数。C51 编译器最多可支持 32 个中断。
- **using n** : 用于指定中断函数使用的工作寄存器组，n=0-3 分别表示选择第 0-3 组。可缺省，表示中断函数与调用函数采用相同的工作寄存器组。



5.4.5 利用 I/O 端口扩展外部中断源



将 4 个按键（外部中断源）连接到一个 4 “与门” 的输入端，当其中一个或几个按键按下时，“与门” 输出从高电平变为低电平，该下降沿触发 MCU 的 $\overline{INT0}$ 。在中断服务程序中，按程序设置的顺序查询 4 条 I/O 口线的状态，确定本次是哪个按键按下引起的中断，然后进行相应的按键处理。



6.1.1 定时器 / 计数器的原理

定时器 / 计数器的**核心**是计数器 (Counter)。计数器是能够对输入脉冲信号的跳变沿进行检测并能进行加法或减法计数的电路模块。

1. 加法计数器

2. 减法计数器

1. 定时功能

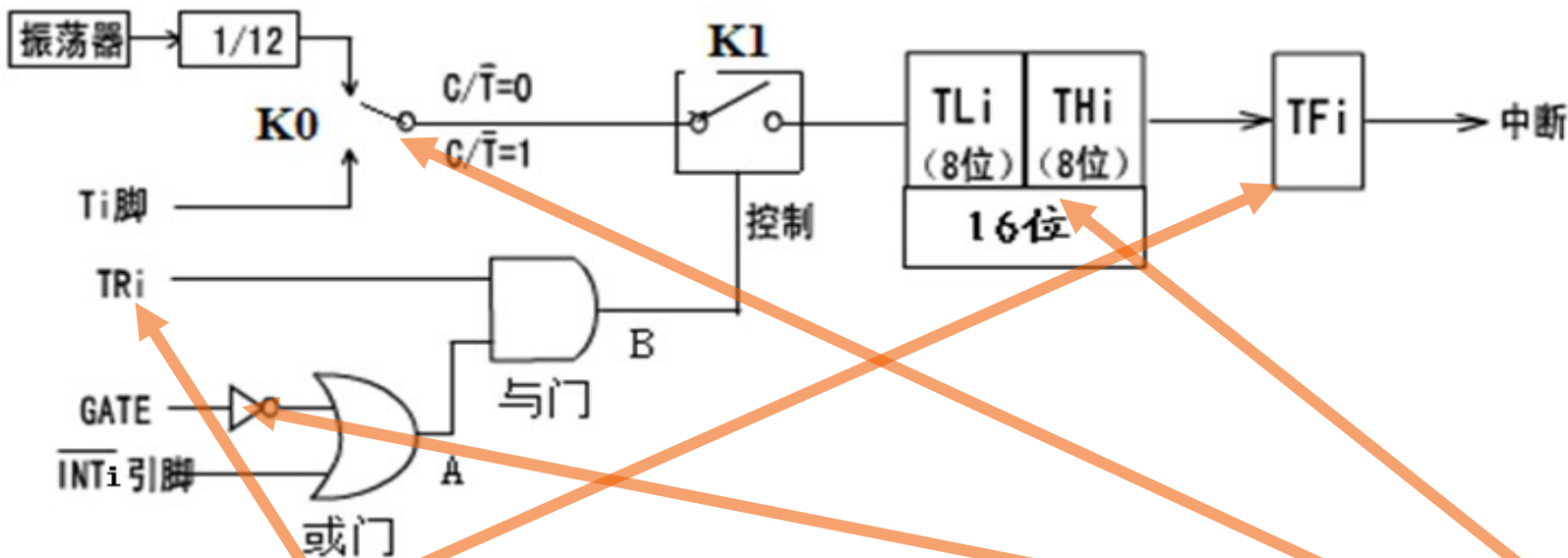
- 软件定时：
- 硬件定时：

2. 计数功能



6.2.3 定时器 / 计数器的工作方式

总结



88H	7	6	5	4	3	2	1	0
位符号	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TCON

89H	7	6	5	4	3	2	1	0
位符号	GATE	C/~T	M1	M0	GATE	C/~T	M1	M0

T1

T0


TMOD




6.2.3 定时器 / 计数器的生活方式

工作方式的选择：有 4 种工作方式，用 TMOD 的 M1、M0 进行选择。

	7	6	5	4	3	2	1	0
位符号	GATE	C/ \overline{T}	M1	M0	GATE	C/ \overline{T}	M1	M0
英文注释	Gate	Counter/ Timer	Mode bit 1	Mode bit 0	Gate	Counter/ Timer	Mode bit 1	Mode bit 0



定时/计数器 T1



定时/计数器 T0

M1 M0	工作方式	功 能
0 0	方式 0	13 位计数器 <small>STC51 : 16 位自重载模式</small>
0 1	方式 1	16 位计数器
1 0	方式 2	计数初值自动重装载 8 位计数器
1 1	方式 3	定时器 0 分成两个 8 位计数器，定时器 1 停止计数。

CT=0 , GATE=0 普通计时功能。

CT=0 , GATE=1 用于测量外部脉冲高电平宽度 (INTi)。

CT=1 , GATE=0 用于对外部脉冲数量计数 (Ti)。



6.2.4 定时器 / 计数器的初始化

2. 定时 / 计数初值的确定

12Mhz 晶振，设定定时初值为 X ，则定时时间 $t = (M - X) \times 1\mu s$ ；

对于晶振频率为 f 的 MCU 系统，定时时间 t 为：

$$t = (M - X) \times \frac{12}{f} = (M - X) \times \text{机器周期} (\mu s)$$

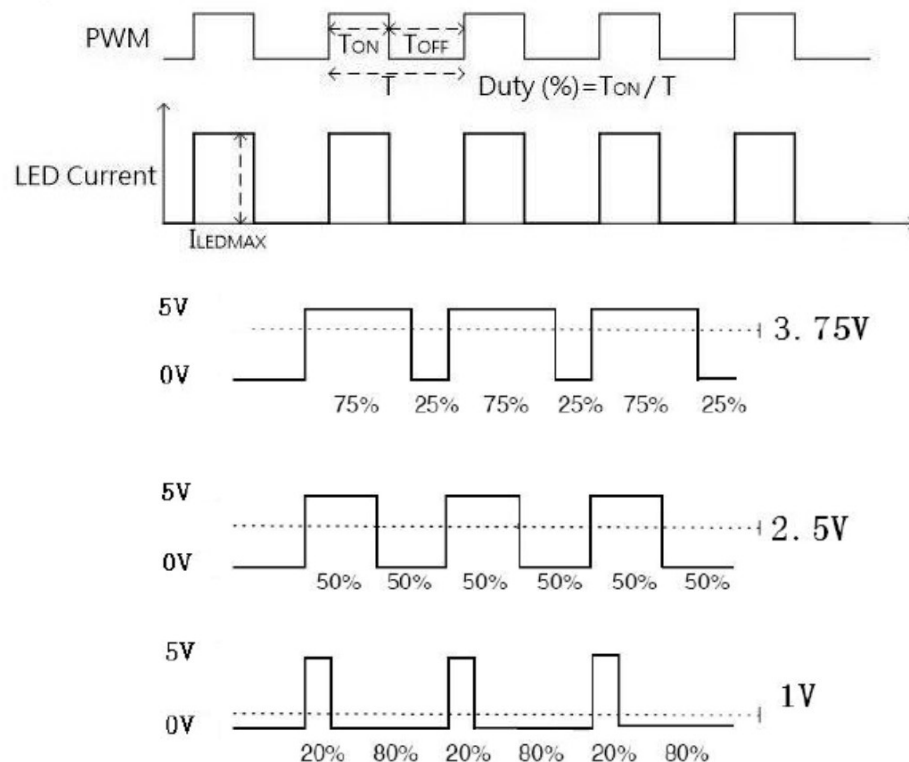
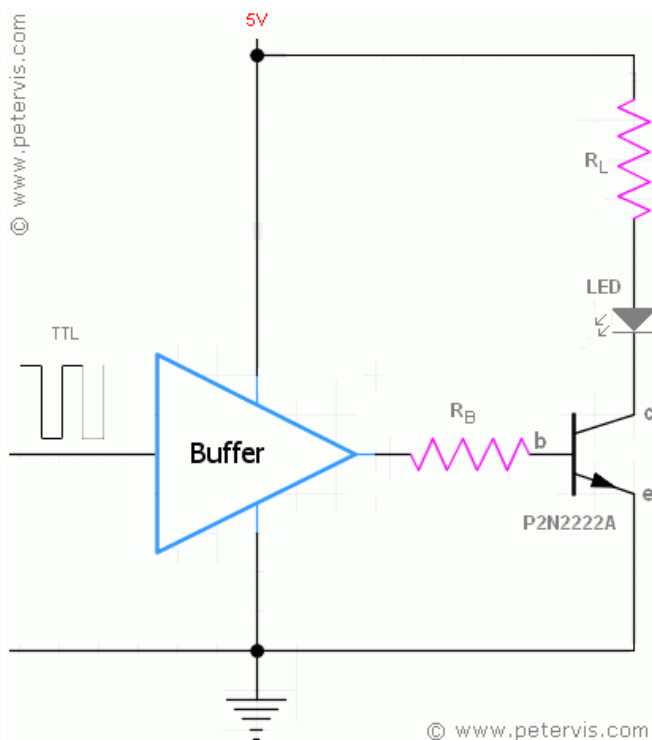
如果要求定时的时间为 t ，则根据上式可以得到定时初值 X ：

$$X = M - \frac{tf}{12}$$



6.3.1 定时方式的应用

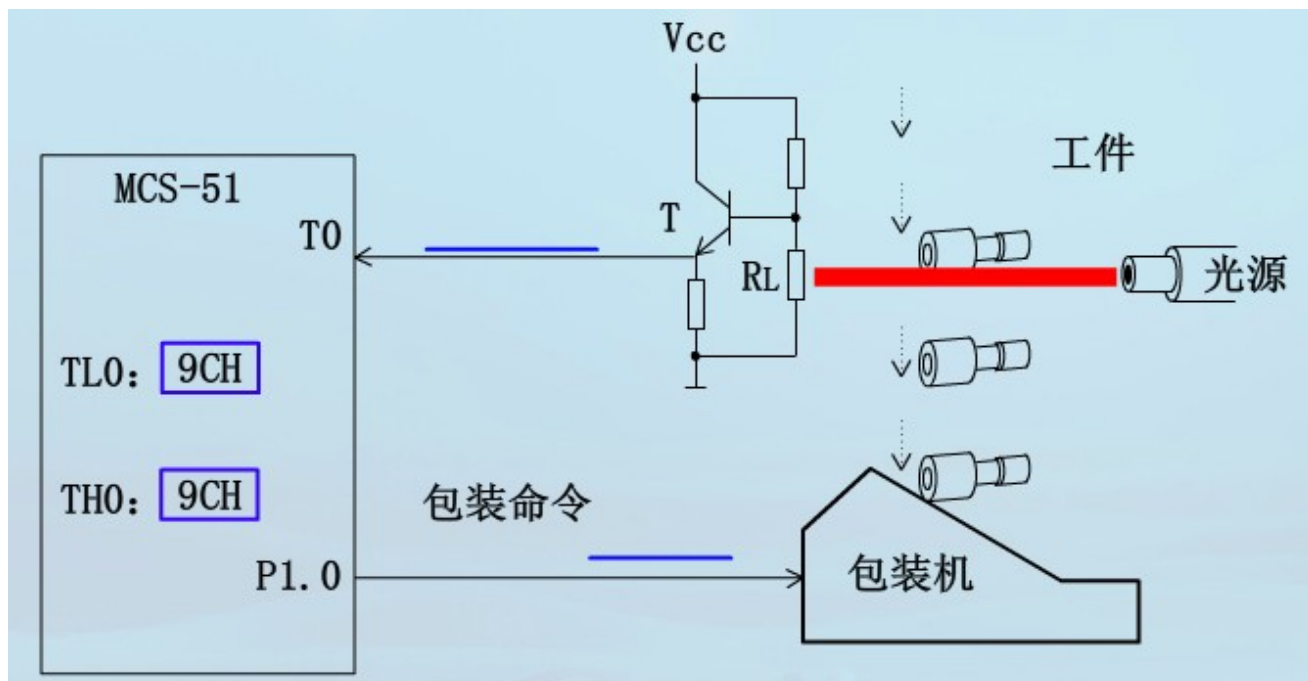
- PWM 调光，即按一个固定的频率来接通和断开 LED 驱动，并需要根据改变一个周期内“接通”和“断开”时间的长短。
- 通过改变 LED 驱动的“占空比”来改变平均电压的大小，实现 LED 光通量的调节。





6.3.2 计数方式的应用

外围电路设计：选用 LED 光源和光敏电阻 RL 作为流水线上工件的检测模块。当有工件通过时，LED 发出的光线受阻挡无法到达光敏电阻 RL，其阻值很大而使三极管 T 导通输出高电平；当没有工件时，光敏电阻接收到 LED 光使 RL 变小，此时 T 截止而输出低电平。因此，每通过一个工件，T0 端就会接收到一个正脉冲信号，由 T0 进行计数。





7.1.3 通信种类

1. 异步通信 与 同步通信

2. 单工，半双工，全双工通信

3. 串行通信 与 并行通信

4. 奇偶校验

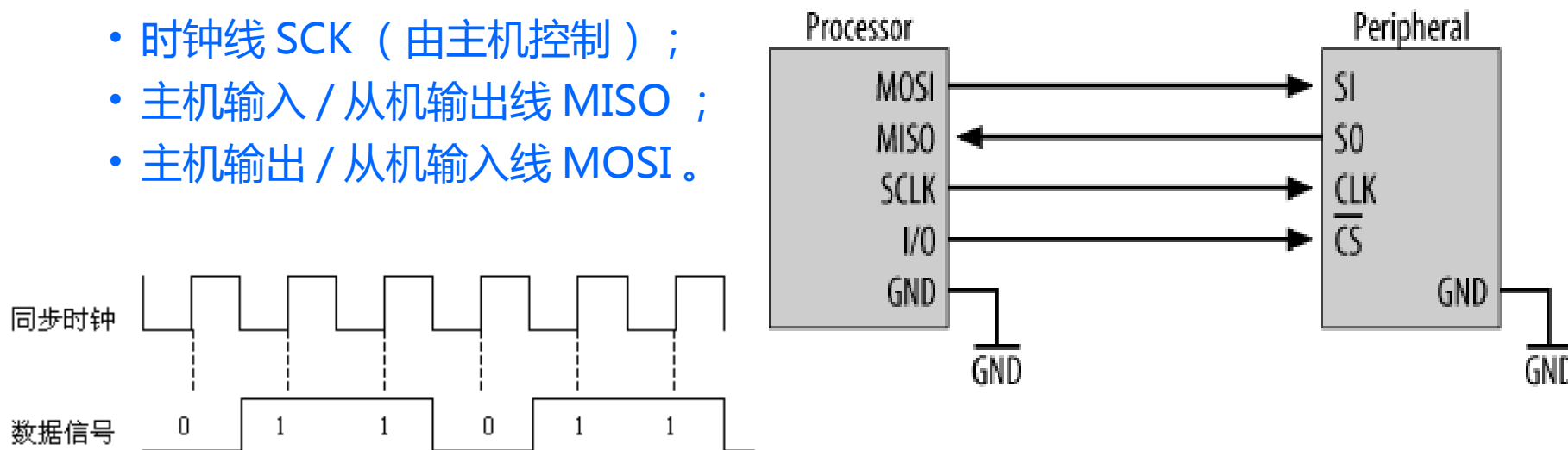


7.3 SPI 串行接口

1. SPI 概述

SPI 接口一般使用 4 条线：串行时钟线（SCLK）、主机输入 / 从机输出数据线 MISO（Master Input/Slave Output）、主机输出 / 从机输入数据线 MOSI（Master Output/Slave Input）和低电平有效的从机选择线 CS。

- 时钟线 SCK（由主机控制）；
- 主机输入 / 从机输出线 MISO；
- 主机输出 / 从机输入线 MOSI。



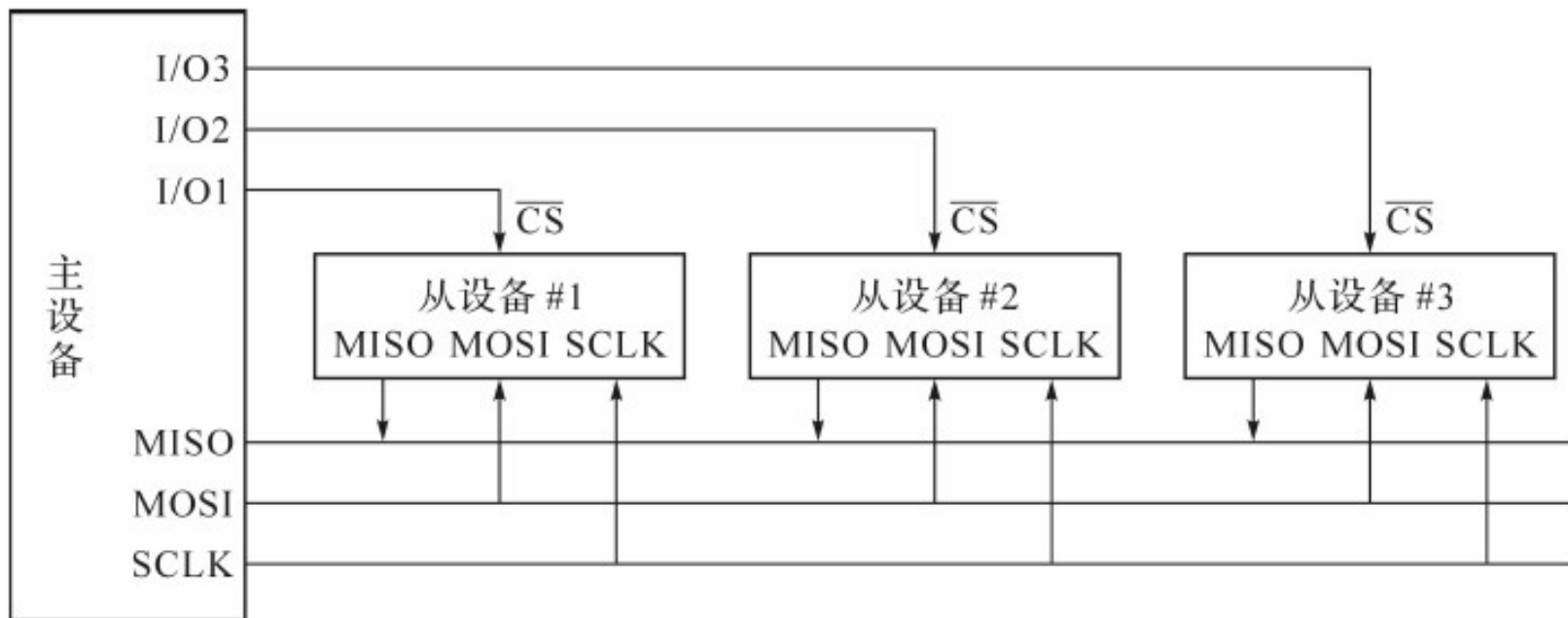
器件的寻址方式？

每个 SPI 芯片还需要片选信号，各芯片的片选信号由主机产生；任何时刻只能一个 SPI 芯片的片选信号有效，此时主机可以对该芯片进行操作。

7.3 SPI 串行接口

2. SPI 接口的扩展

- 常用的是一主多从结构。主机分别通过 I/O 口线来分时选通各从机。
- 有些从机是单向的，如键盘管理芯片只作为输入；显示器管理芯片只作输出，此时与这些芯片的连接就只需一条数据线（MISO 或 MOSI）。



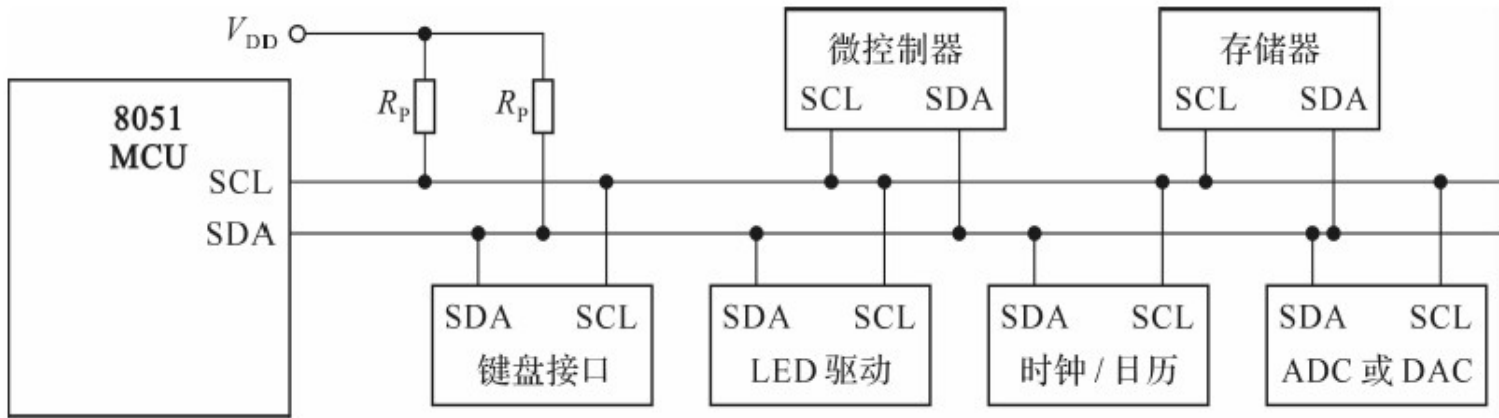


7.4.1 I²C 总线概述

1. I²C 总线系统基本结构 (Inter-Integrated Circuit)

所有器件的数据线均连接到 I²C 总线的 SDA 线，时钟线均连接到 SCL 线。

I²C 串行总线的运行由主器件控制，主器件是指启动和结束数据发送、发出时钟信号的器件，通常是微控制器承担。主器件可以具有 I²C 串行总线接口，也可以不带 I²C 串行总线接口，而用普通 I/O 口线用软件模拟。从器件必须带有 I²C 总线接口。



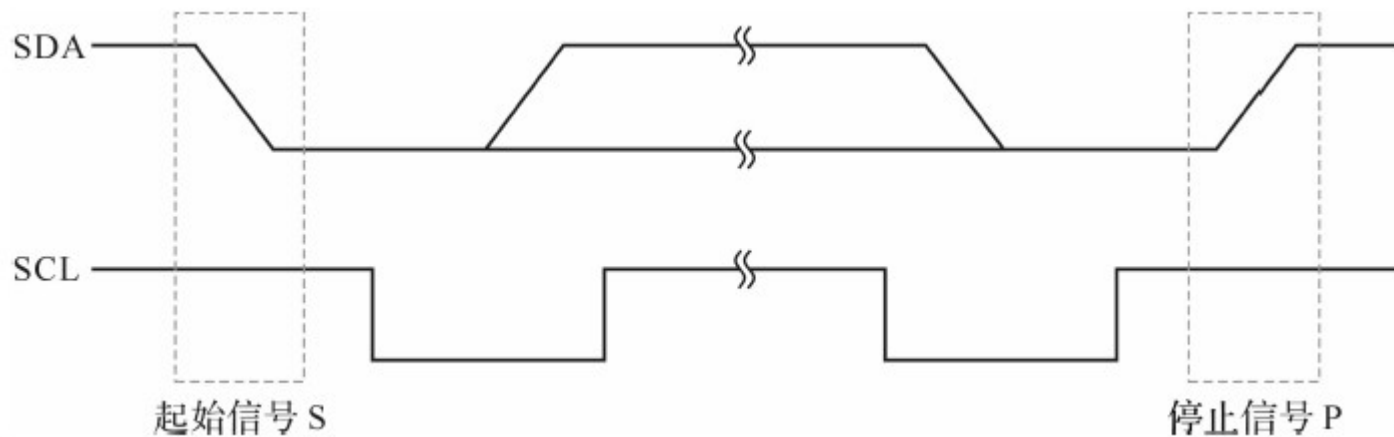
I²C 串行总线系统的结构示意图



7.5 I²C 总线的操作

1. I²C 总线的起始信号与停止信号

- **起始信号 S** : 当 SCL 处于高电平期间, SDA 从高电平向低电平跳变时产生起始信号。总线在起始信号后才开始数据传送。
- **停止信号 P** : 当 SCL 处于高电平期间, SDA 从低电平向高电平跳变时产生停止信号。总线在停止信号产生后处于空闲状态。

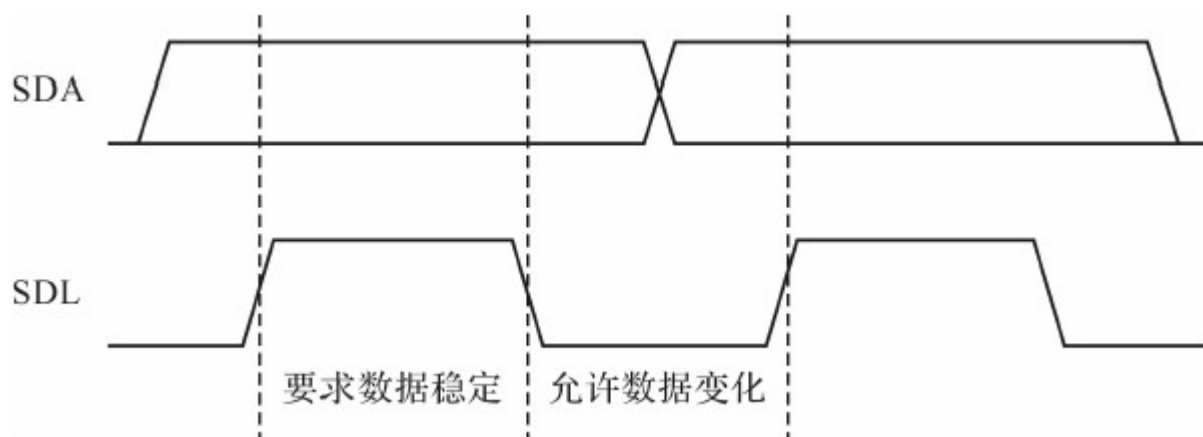




7.5 I²C 总线的操作

2. I²C 总线上数据位的有效性

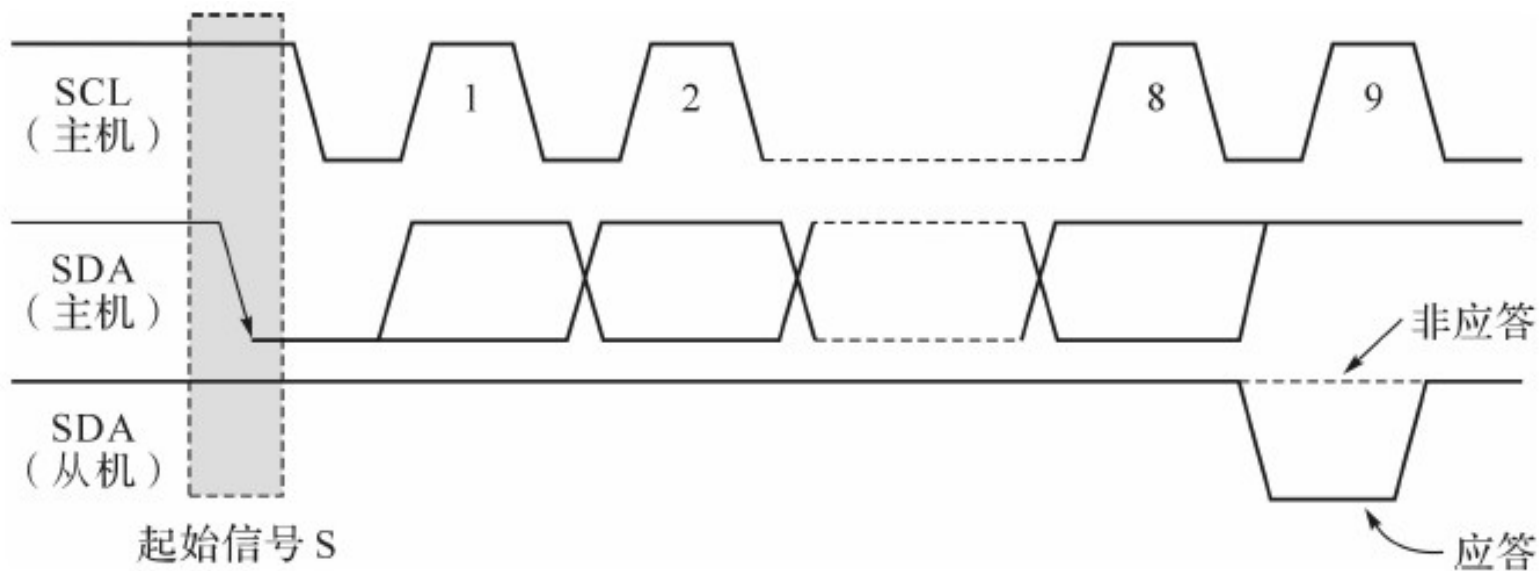
- I²C 总线为同步传输总线，每一数据位的传送都与时钟脉冲 SCL 同步下进行的。在时钟信号 SCL 高电平期间，数据线 SDA 上的数据必须保持不变。
- SDA 的数据只有在 SCL 低电平期间才允许改变。但是在 I²C 总线的起始和结束时例外。





7.5 I²C 总线的操作

4. 应答 (Acknowledge) 与非应答





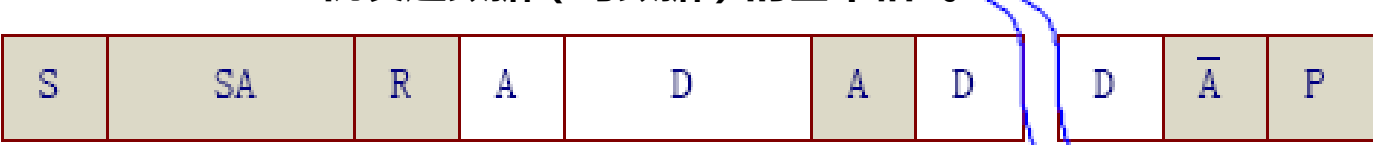
7.5 I²C 总线的操作

6. 数据传输格式

一次完整的数据操作包括起始 (S)、发送从机地址 (SLA R/W) + 应答、发送数据 + 应答、.....、停止 (P)。



主机发送数据 (写数据) 的基本格式



主机接收数据 (读数据) 的基本格式

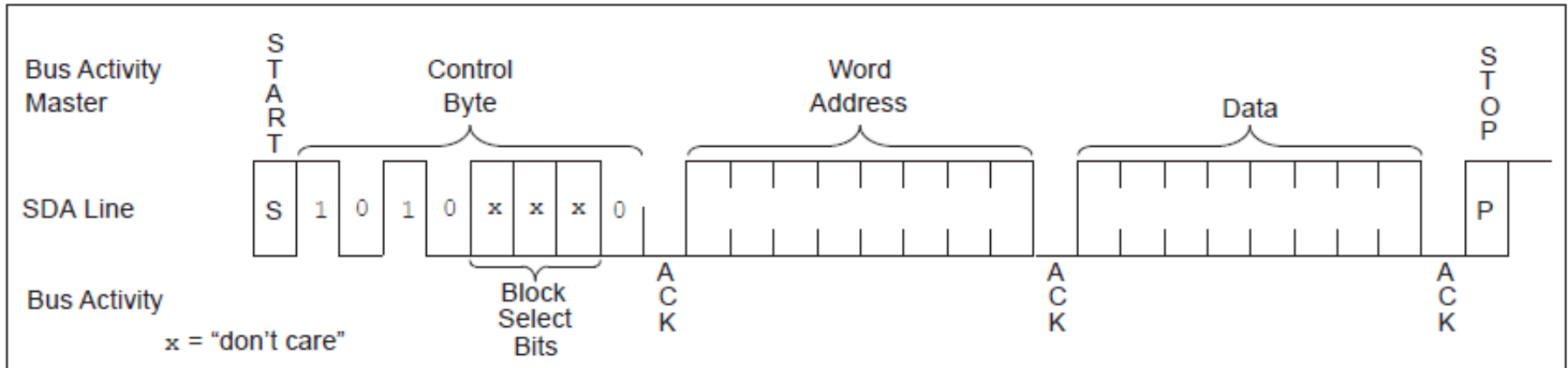
SA : 从机地址高 7 位。
A : 应答信号 ; D : 写入或读出数据。

-  : 主机发送、从机接收。
-  : 主机接收、从机发送。

注意 : 主机向从机发送最后一个字节数据时 , 从机可能应答也可能非应答 , 但不管怎样主机都可以产生停止条件。

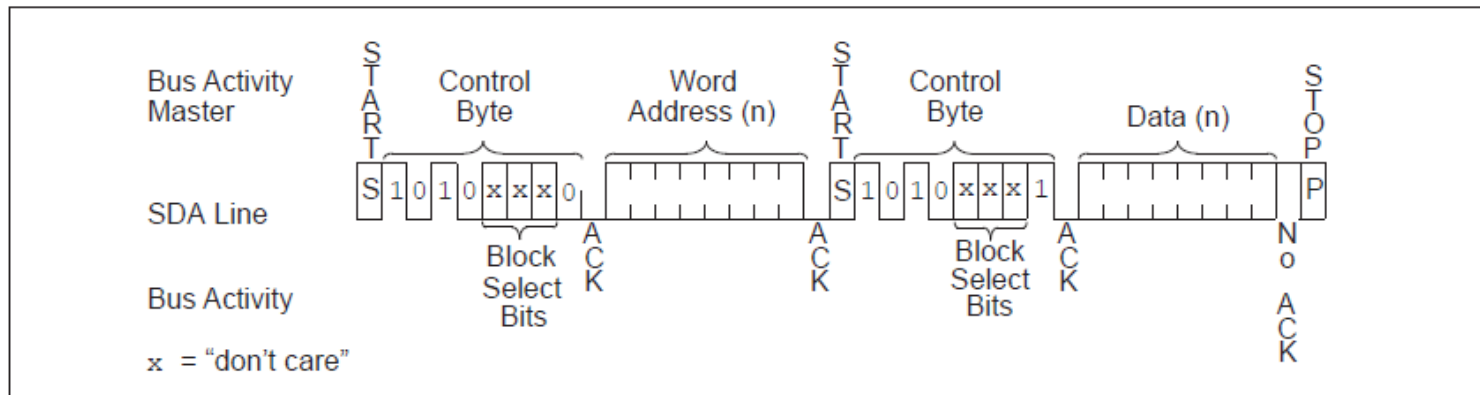


FIGURE 6-1: BYTE WRITE



```
void WrToROM(unsigned char Data_Write,unsigned char Address)
//Data_Write: 要写的字节 , Address: 写入的起始地址
{
    Start();
    Send(0xA2);                // 发送从机地址
    CheckACK();
    Send(Address);            // 发送存储地址
    CheckACK();
    Send(Data_Write);         // 写 8 位
    CheckACK();
    Stop();
}
```

FIGURE 8-2: RANDOM READ

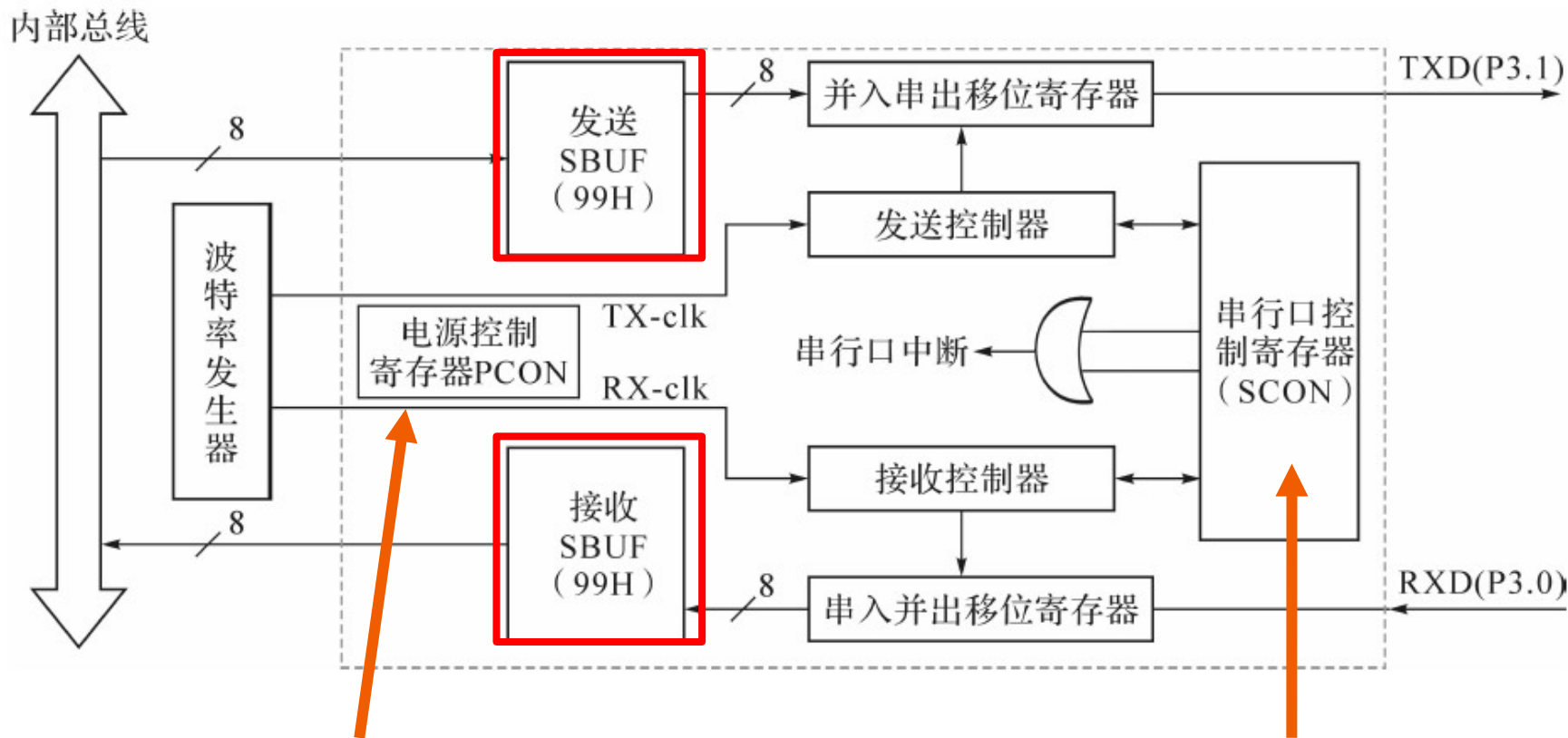


```

unsigned char RdFromROM(unsigned char Address) // 读取数据
{
    unsigned int xdata Temp_Data;
    Start();
    Send(0xA2); // 发送从机写地址
    CheckACK();
    Send(Address); // 发送数据存储地址
    CheckACK();
    Stop();

    Start();
    Send(0xA3); // 发送从机读地址
    CheckACK();
    Temp_Data=Read(); // 读取 8 位
    NoAck();
    Stop();
    return Temp_Data;
}
    
```


7.2.1 UART 的组成结构



87H	7	6	5	4	3	2	1	0
位符号	SMOD	-	-	-	GF1	GF0	PD	IDL

PCON

98H	7	6	5	4	3	2	1	0
位符号	SM0	SM1	SM2	REN	TB8	RB8	TI	RI

SCON



7.2.1 UART 的组成结构

1. 串行口控制寄存器 SCON

字节地址为 98H ，可位寻址。

位	7	6	5	4	3	2	1	0
位符号	SM0	SM1	SM2	REN	TB8	RB8	TI	RI
英文注释	Serial Mode bit 0	Serial Mode bit 1	Serial Mode bit 2	Receive Enable	Transmit bit 8	Receive bit 8	Transmit Interrupt Flag	Receive Interrupt Flag

➤SM1、SM0：
工作方式选择位。

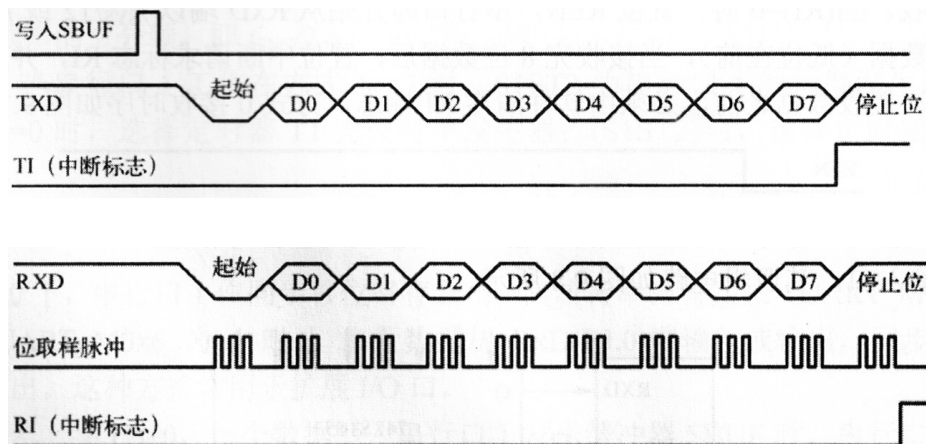
SM0	SM1	工作 方式	特 点	波特率
0	0	方式 0	8 位移位寄存器	$f_{osc}/12$
0	1	方式 1	10 位 UART	可设置
1	0	方式 2	11 位 UART	$f_{osc}/64$ 或 $f_{osc}/32$
1	1	方式 3	11 位 UART	可设置

7.2.2 UART 的工作方式

方式 1

发送时序

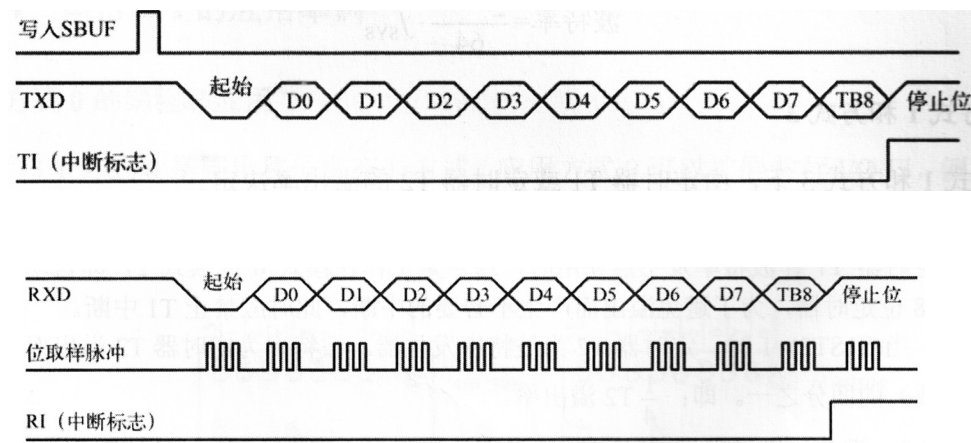
接收时序



方式 2 ,
3

发送时序

接收时序





7.2.3 UART 的波特率

2. 定时器 T1 作为波特率发生器

T1 作波特率发生器时，应工作在定时方式 2，即 8 位自动重装载方式，禁止中断。设初值为 X，则 T1 的定时时间（溢出周期）T 为：

$$T = \frac{12}{f_{\text{osc}}} \times (2^8 - X)$$

溢出率为溢出周期的倒数，
所以

$$\text{波特率} = \frac{2^{\text{SMOD}}}{32} \times T1 \text{溢出率} = \frac{2^{\text{SMOD}}}{32} \times \frac{f_{\text{osc}}}{12 \times (2^8 - X)}$$

则 T1 的定时初值 X 为：

$$X = 2^8 - \frac{f_{\text{osc}} \times (\text{SMOD} + 1)}{384 \times \text{波特率}}$$



7.2.6 RS232/485 通信技术

6. RS232 , RS485 , RS422 的区别

RS-232 使用负逻辑电平。逻辑 “1” : -5V ~ -15V ; 逻辑 “0” : +5V ~ +15V 。

RS485 采用差分信号传输信息：差分电压为 -2500 ~ -200mV 为逻辑 “0” ；差分电压为 + 2500 ~ + 200mV 时为逻辑 “1” ；差分电压信号为 - 200 ~ + 200mV 时为高阻状态。

特性	RS232	RS485/RS422
传输距离	传输距离短，最大距离 50 米左右	最大传输距离实际上可达 1200 米
传输速率	传输速率较低，最大波特率为 200Kbps	最高传输速率为 10Mbps
抗干扰能力	存在共地噪声，不能抑制共模干扰	具有抑制共模干扰的能力
通信方式	点对点通信	可以组网构成分布式系统
信号类型	数字信号	差分信号
连接方式	1 对 1	1 对多



8.2.1 LED 显示原理

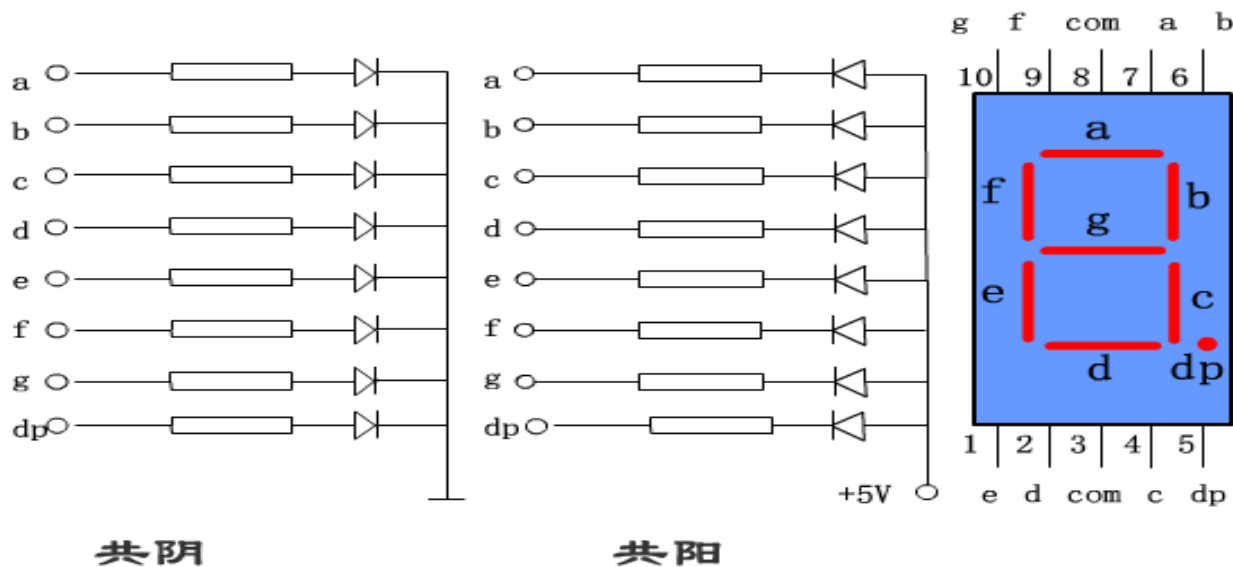
LED 即发光二极管，是微机系统中最常用的显示器。LED 显示器有单个 LED、8 个 LED 组成的数码管和点阵式（ 5×7 、 8×8 ）LED 显示器等几种

类型

1. 段码式 LED 显示器（数码管）

共阴数码管：COM 端接地或具有较大灌电流的输入口线，阳极高电平时点亮。

共阳数码管：共阳极接电源或具有强高电平驱动输出口线，阴极低电平时点亮。

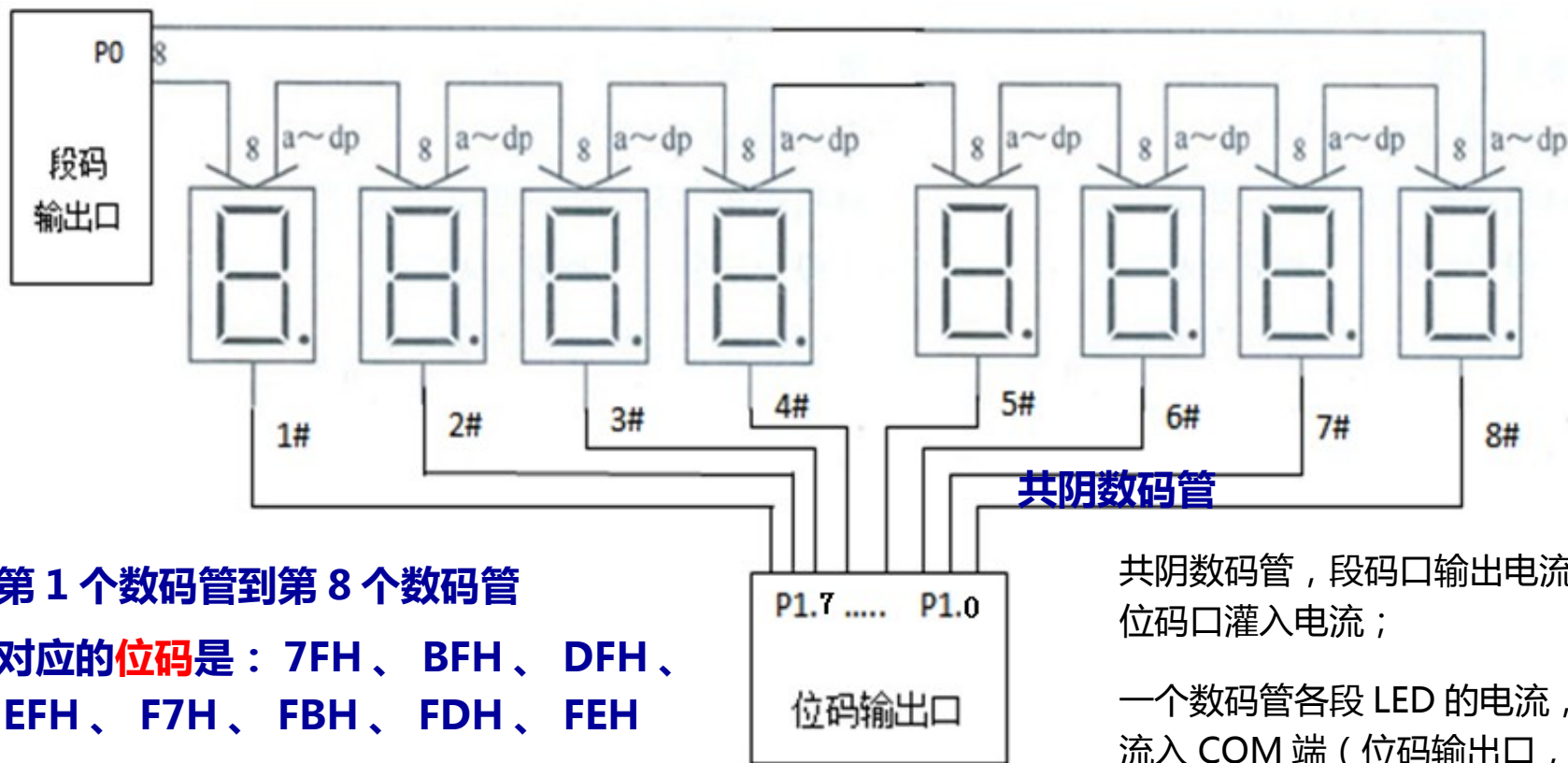




8.2.2 段码式 LED 显示技术

2.LED 动态显示技术

(1) 硬件连接：P0 口作为段码输出口，P1 口作为位码输出口。



第 1 个数码管到第 8 个数码管

对应的位码是：7FH、BFH、DFH、
EFH、F7H、FBH、FDH、FEH

段码：通过查表得到

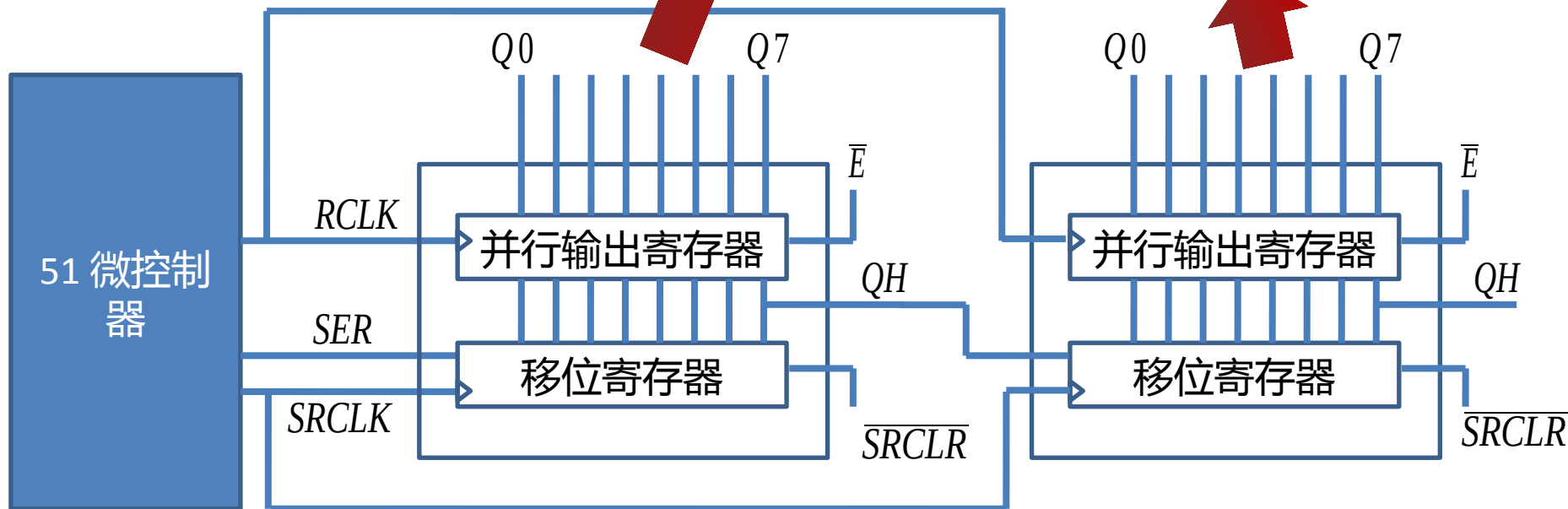
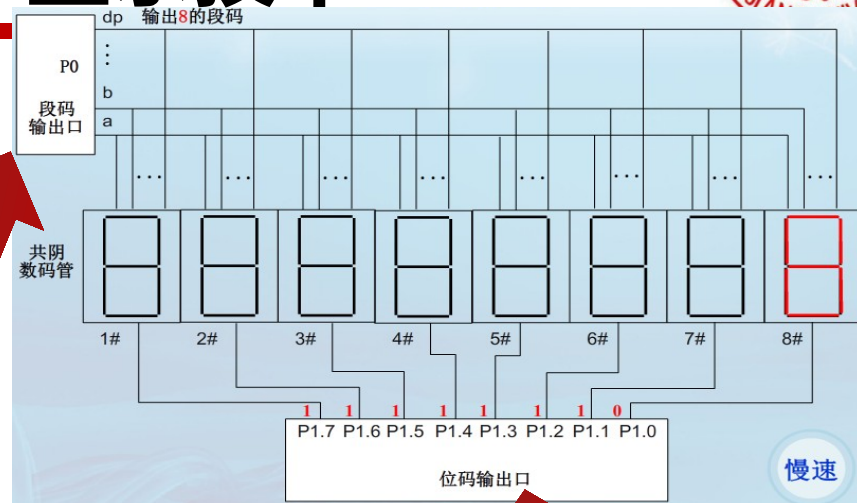
共阴数码管，段码口输出电流，
位码口灌入电流；

一个数码管各段 LED 的电流，均
流入 COM 端（位码输出口，要
考虑驱动（灌电流）能力



8.2.2 段码式 LED 显示技术

用 2 片 74HC595
扩展 2 个输出接
口

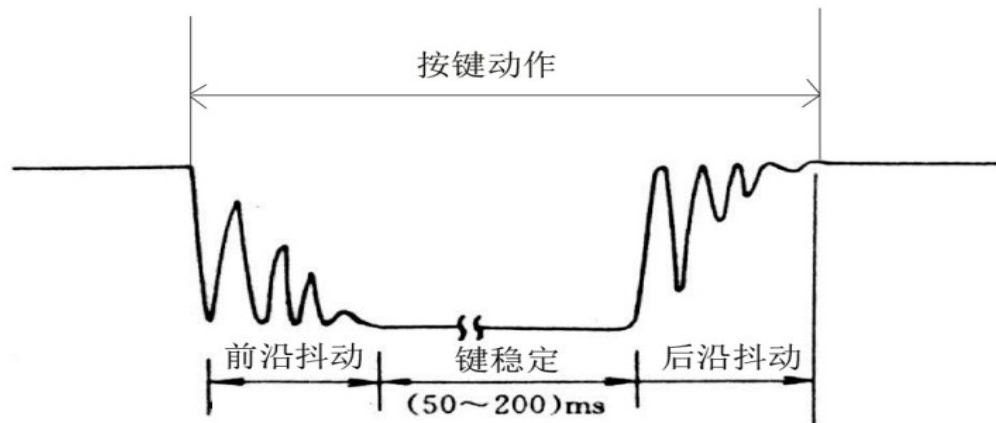




8.4.1 键盘基础知识

2. 按键抖动与消除

触点式按键在闭合和断开瞬间存在抖动过程，即存在抖动现象，前后沿抖动时间一般在 $5\text{ms} \sim 10\text{ms}$ 。按键的稳定时间与按键动作有关，通常大于 50ms 。



按键抖动可能导致微机对一次按键操作作出多次响应，所以要去抖动。

(1) 硬件电路去抖动：需要利用 RS 触发器等构成去抖动电路（很少使用）。

(2) 软件延时法：当检测到有键按下时，用软件延时 $10\text{ms} \sim 20\text{ms}$ ，等待键稳定后重新再判一次，以躲过触点的抖动期。

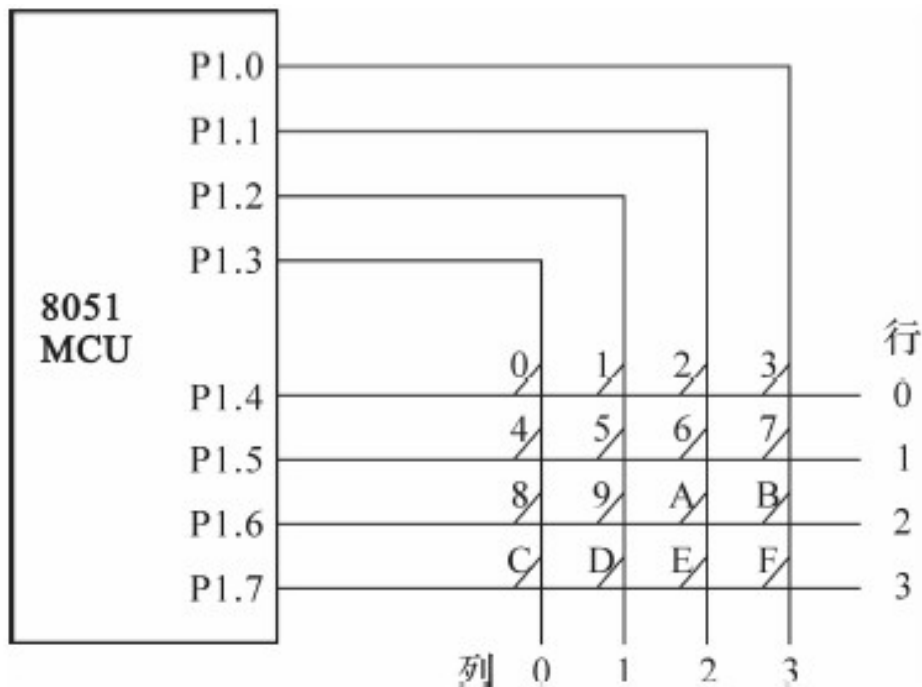


8.6.3 矩阵式键盘接口

矩阵式键盘：需要行线和列线，按键位于行线和列线的交叉点上； $m \times n$ 矩阵键盘只需要 $m + n$ 条口线。

按键数目较多的系统中，矩阵式键盘比独立式按键要节省很多 I/O 口线。

矩阵式键盘判别按键的方法有行扫描法和线反转法。



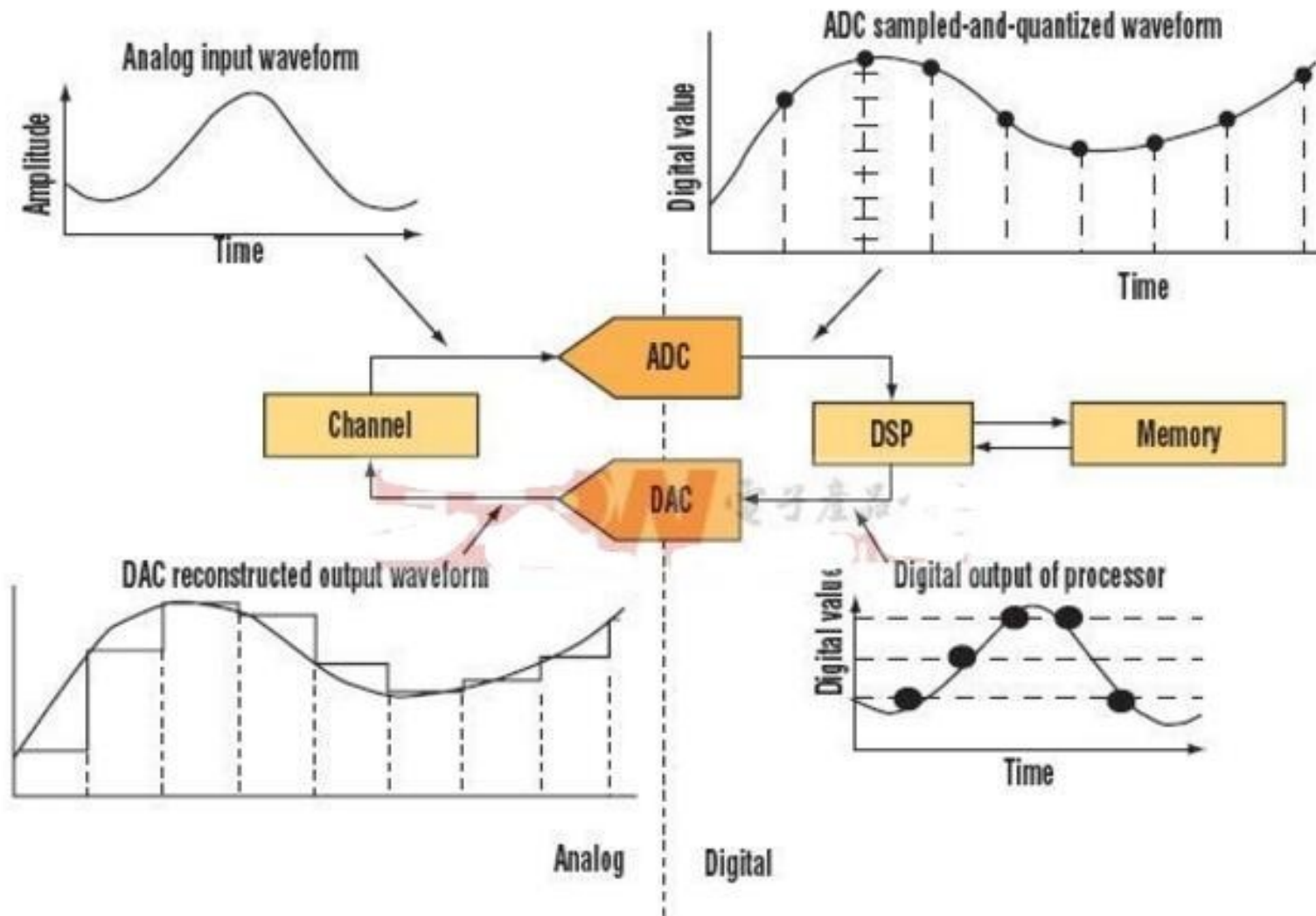
P1.4-P1.7 为行扫描输出线；

P1.0-P1.3 是列输入线。

若将 4 个列信号连接到一个 4 输入的与门，与门输出连接到外部中断引脚，则有键按下时，就会向 CPU 请求中断。



内容提要





9.1.3 AD 转换器及其特性

3. AD 转换器主要指标

(1) 分辨率与量化误差

- **分辨率**：反映 ADC 对输入电压微小变化的响应能力，用 ADC 输出的二进制位数表示，如 8 位、12 位分辨率。

n 位 ADC 表示可用 2^n 个数进行量化，位数越多分辨率越高。

- **电压分辨率**：与 ADC 的满量程电压有关，是输出数字量变化 1 所对应的输入电压，也称最小分辨电压，用 LSB (Least Significant Bit) 最低有效位表示。

$$\text{电压分辨率 } \Delta U = 1/2^n \times \text{满量程} = 1\text{LSB}$$

满量程为 5V 的 8 位和 12 位 ADC，它们的电压分辨率分别为：

$$\Delta U_8 = 19.5\text{mv}, \Delta U_{12} = 1.22\text{mv}$$

- **量化误差**：由于用有限数字对连续模拟量进行离散而引起的误差，其数值为一个

单位分辨率，即 $\pm 1/2\text{LSB}$ 。减少量化误差的方法是提高分辨率，即增加 ADC 位数。



9.1.3 AD 转换器及其特性

3. AD 转换器主要指标

(2) 转换时间和转换速率

- **转换时间**：ADC 完成一次模数转换所需的时间。
- **转换速率**：转换时间的倒数，通常用次数 / 秒表示，也称转换频率。如 10K/s (10KHz)、2M/s (2MHz) 速率的 ADC，它们的转换时间是 100 μ s、0.5 μ s。

(3) 转换精度

转换精度：ADC 的实际输出结果与理论转换结果的偏差，有两种表示方法。

- **绝对精度**：用电压分辨率 (LSB) 的倍数表示，如 $\pm 1/2\text{LSB}$ 、 $\pm 1\text{LSB}$ 等。
- **相对精度**：用绝对精度除以满量程值的百分数表示， $\pm 0.05\%$ 、 $\pm 0.1\%$ 等。

(4) 量程

量程是指输入模拟电压的变化范围。例如某转换器具有 10V 的单极性范围

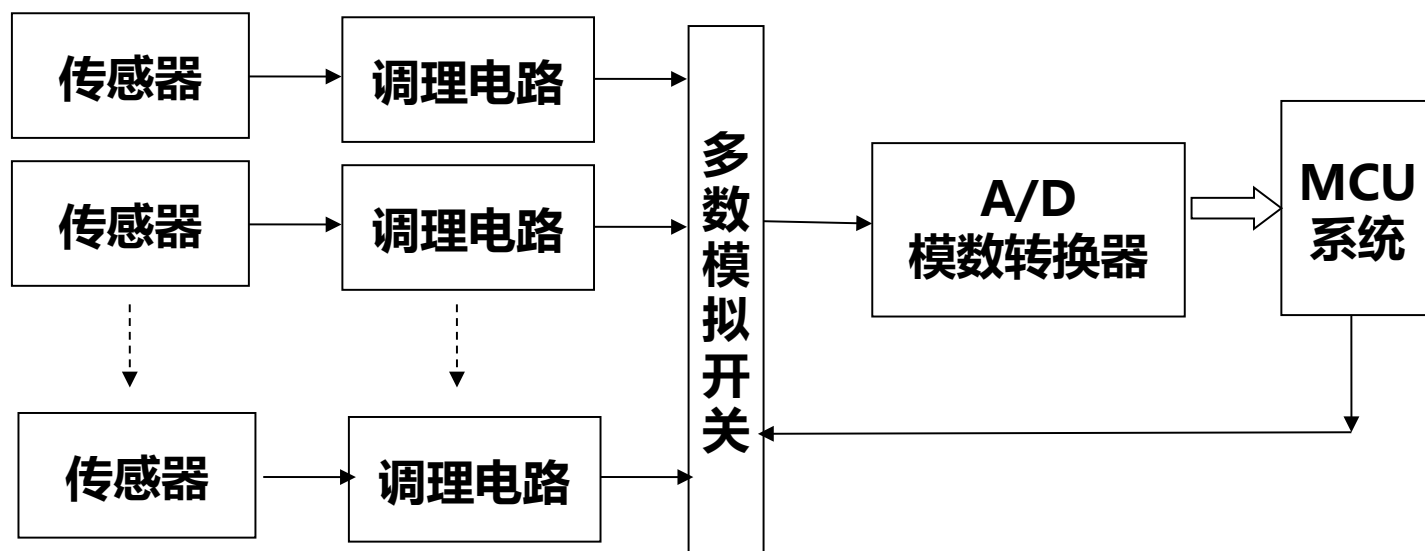
或 $-5 \sim +5\text{V}$ 的双极性范围，则它们的量程都为 10V。



9.1.1 模拟输入通道基本结构

2. 分时采集型

特点：多个监测参数共用一个 A/D 转换器，电路简单，成本低；但速度慢，适用于多路缓变信号的采集。



分时采集型模拟输入通道结构

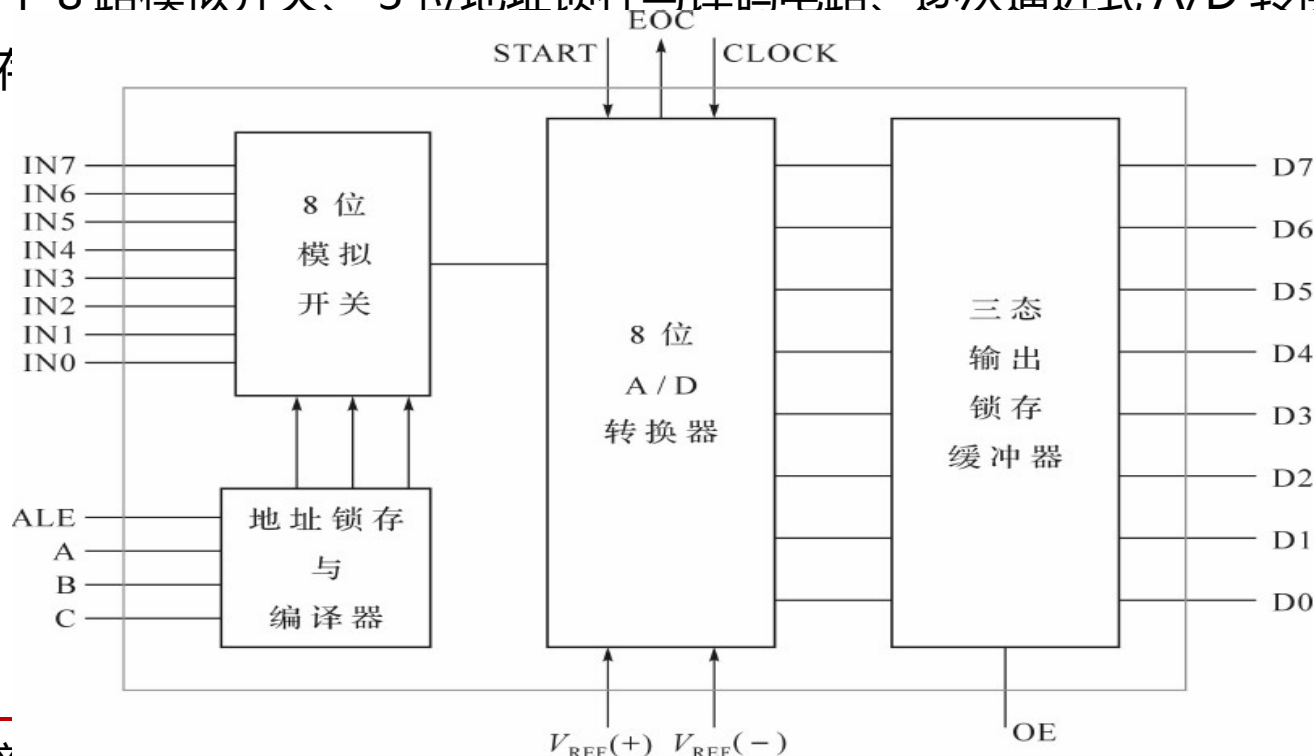


9.2.1 并行 AD 转换器

1. ADC0809 与内部结构

ADC0809 是采用 CMOS 工艺制成的 8 位 8 通道逐次逼近式并行模数转换器，可分时对 8 路模拟信号进行 A/D 转换。典型转换时间 100us；转换精度为 $\pm 1\text{LSB}$ 。

内部结构：1 个 8 路模拟开关、3 位地址锁存与译码电路、逐次逼近式 A/D 转换器和三态输出锁存缓冲器

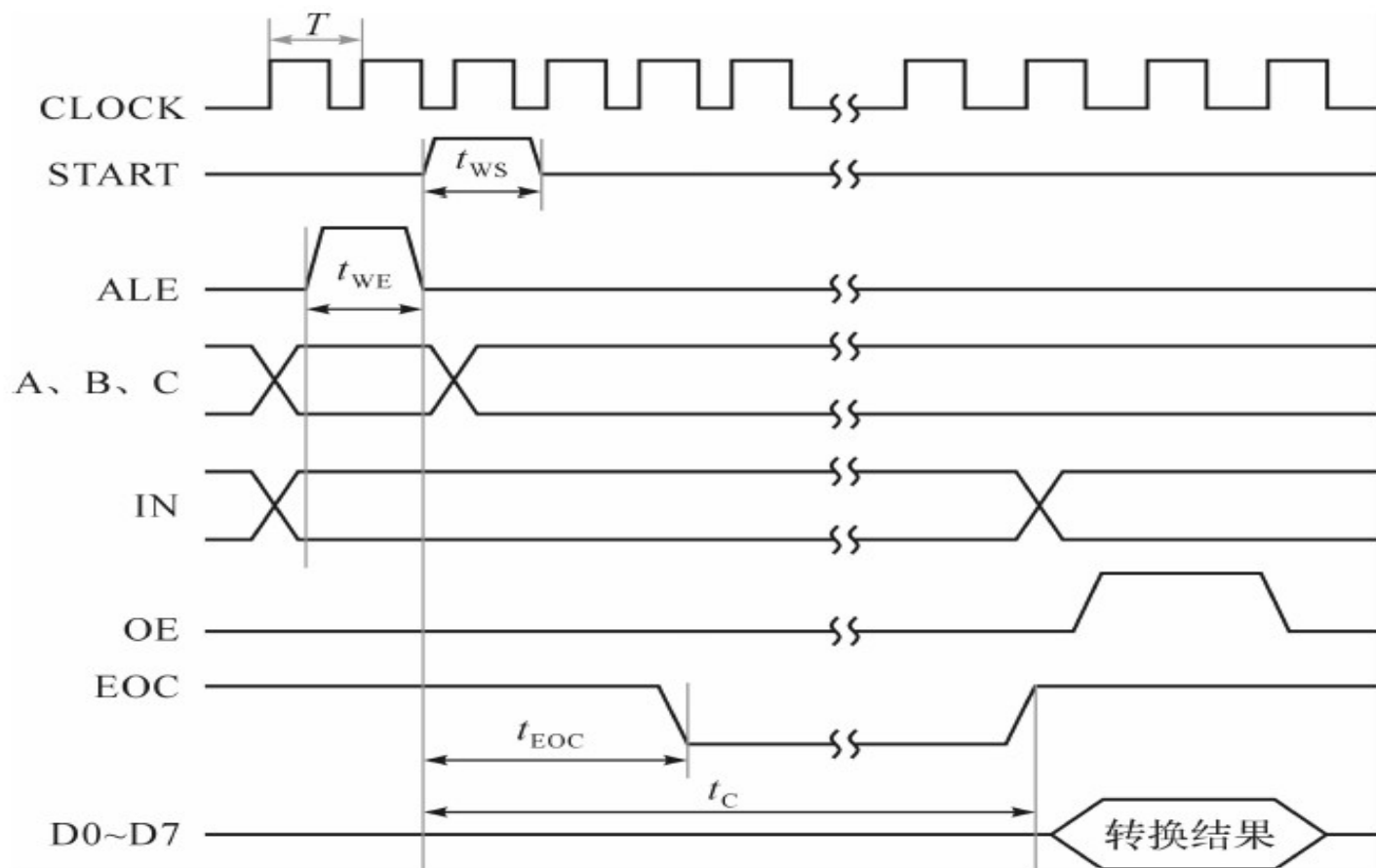




9.2.1 并行 AD 转换器

3. ADC0809 转换时序与过程

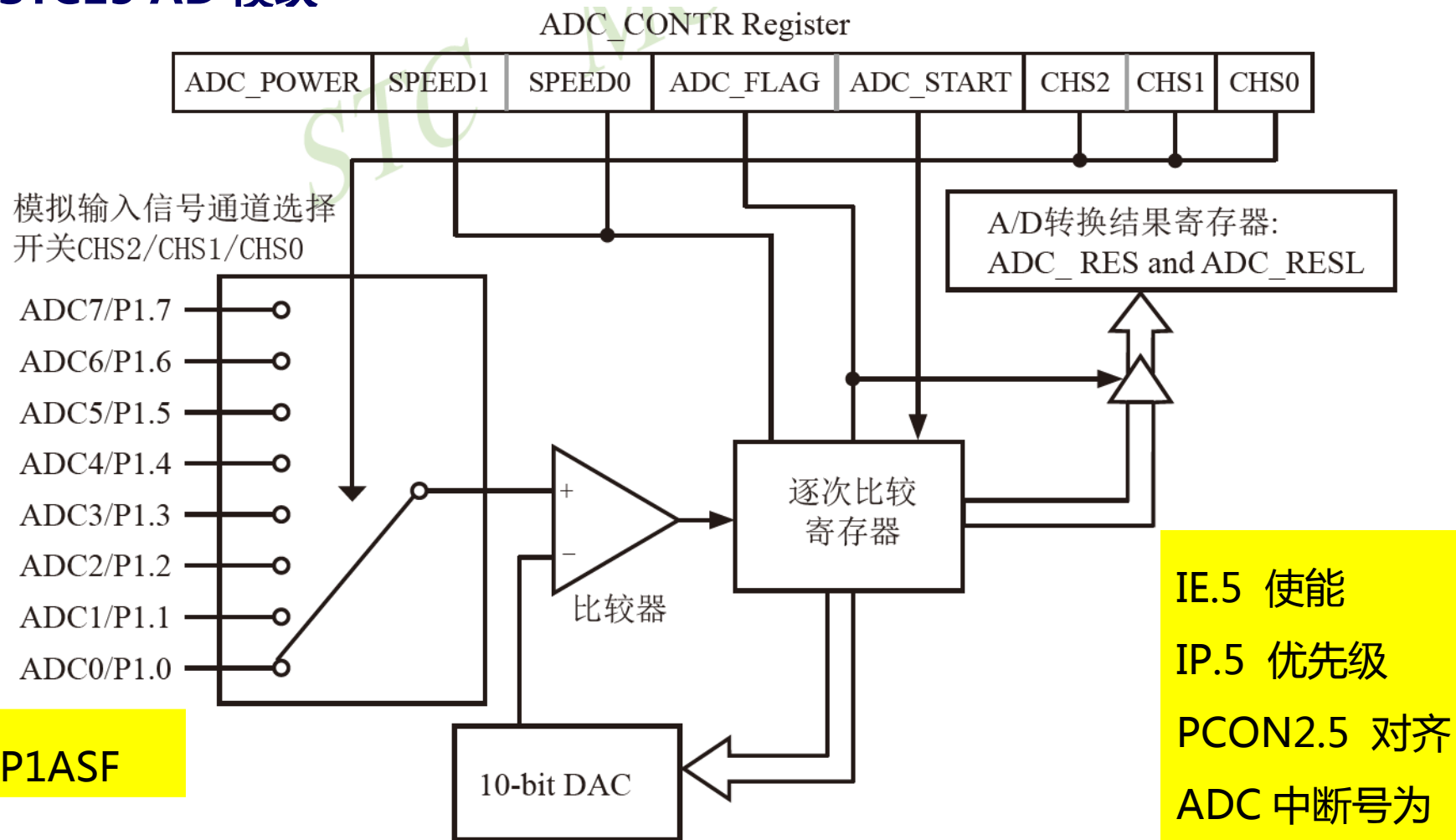
(1) 转换时序





9.2.4 AD 转换器的应用

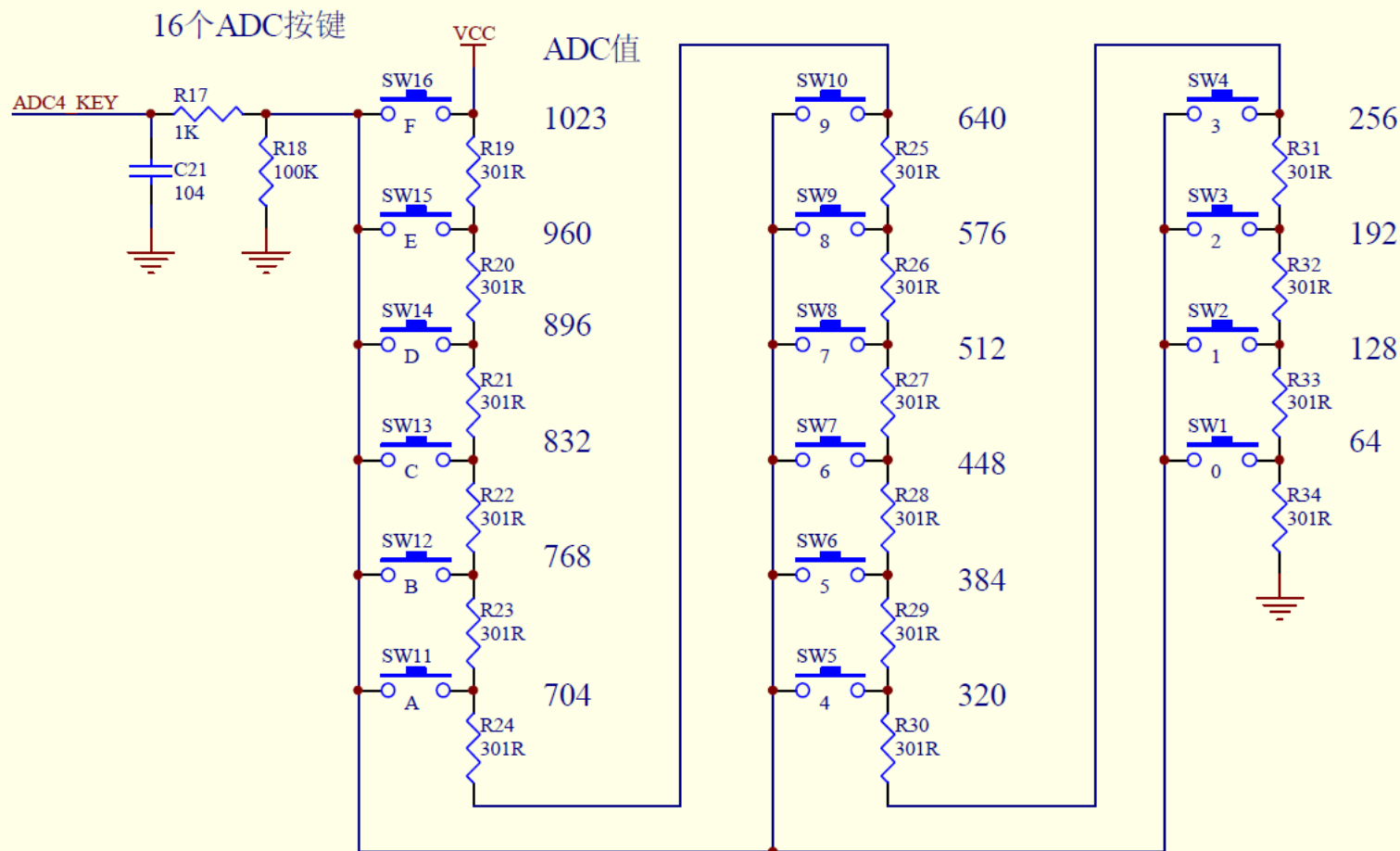
0. STC15 AD 模块





9.2.4 AD 转换器的应用

2. STC15 实验板

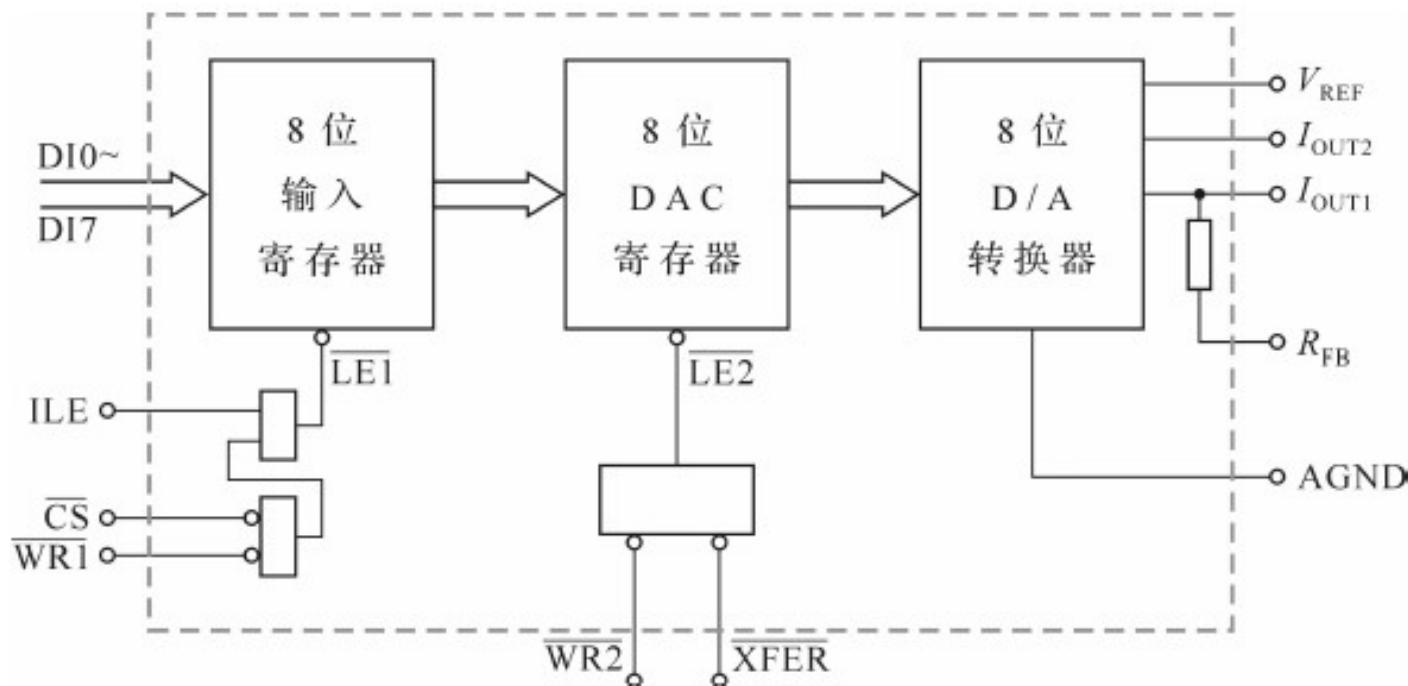




9.3.1 并行 DA 转换器与接口技术

1. DAC0832 与内部结构

由 8 位输入寄存器、8 位 DAC 寄存器、8 位 D/A 转换器、选通逻辑和输出反馈电阻等组成。**输入寄存器和 DAC 寄存器**构成的数字输入的两级缓冲结构，使得 DAC0832 具有单缓冲和双缓冲两种输入方式。



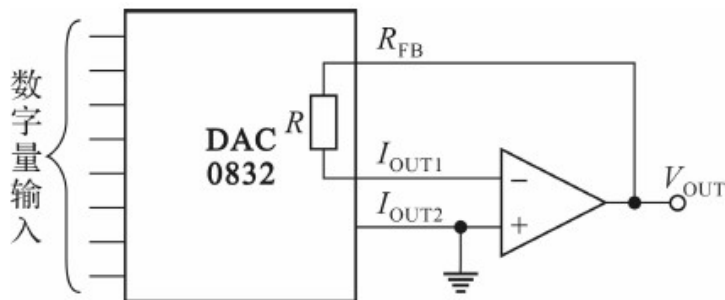


9.3.1 并行 DA 转换器与接口技术

3. DAC0832 的应用特性

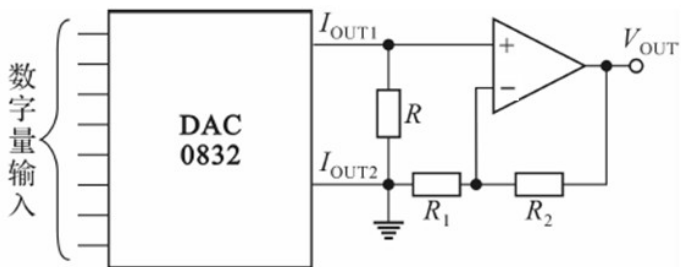
对于电流型 DAC，可通过外接运放，内部电阻作为反馈电阻，通过 I/V 转换得到电压。

(1) 反相电压输出



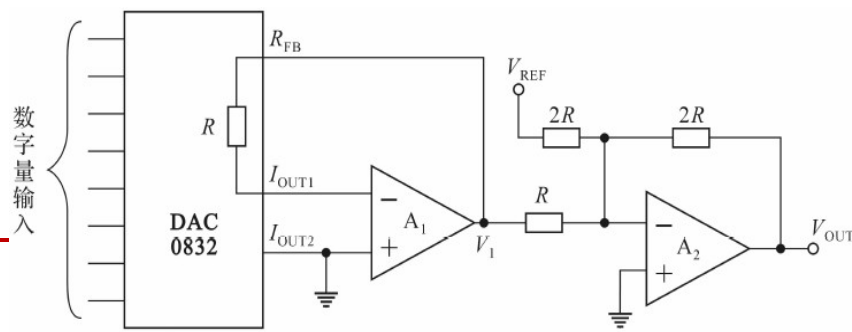
$$0 \sim -\frac{255}{256}V_{REF}$$

(2) 同相电压输出



$$V_{OUT} = \frac{D}{128}V_{REF}$$

(3) 双极电压输出



$$\frac{D}{128}V_{REF} - V_{REF}$$