# TP801

# Z80单板计算机基础知识

北京工业大学

微型机应用与自动化研究室编

# 前　言

自从一九七一年第一个微处理器 Intel4004 问世以来，微处理器/微型计算机得到了飞速的发展。近年来，每年有一亿多只微处理器投放市场，这种新型器件几乎渗透到各个技术领域。

我国广大技术人员迫切需要掌握和推广应用这种技术。为此，我们曾先后在校内外面向教师和技术人员举办了几次讲座，并为我校工业自动化系七九届研究生开设了《微型计算机化设计》课程。在此基础上，我们配合 TP801—Z80 单板计算机的研制、生产，编写了这本《微处理器实用教材——基础知识和 Z80 器件的使用》，作为《TP801—Z80 单板计算机使用手册》的补充，供初学者学习和广大 TP801 的用户参考使用。

本书第一章概要介绍了数字电路的有关知识。第二章介绍了两种为教学而设计的模型计算机，它们分别只具有五条和十六条指令。通过这一章的学习，读者可以大致了解微型计算机总体结构的全貌以及其中各主要部件的工作原理。在第一种模型计算机中还引入了子程序的调用过程以及栈的概念。第三至第六章介绍 Z80 系列中的三种常用器件（CPU，PIO 和 CTC）的主要特性和操作说明。

本书和《TP801—Z80 单板计算机使用手册》配合，将作为 TP801 用户的培训教材，或供初学者自学选用。讲授这两本书约需 40～60 学时，再配合进行适量的实验课程，可使学员获得使用微型计算机的初步知识和技能。

本书第一、二、五、六章由徐家栋编写，第三、四章由颜超编写。侯伯文参加了本书的修改定稿工作。由于我们的水平有限，书中错误在所难免，恳切希望读者指正。

北京工业大学工业自动化系
微型机应用与自动化研究室
一九八一年四月

# 目　录

# 第一章 基础知识

## 1.1 数制及不同数制間的換算

### 一、十进制数

在日常生活中，人们最熟悉的是十进制数。它有十个不同的数字：0，1，2，3，4，5，6，7，8，9。在表示数时，这些处于不同的位置(或数位)的数字代表的意义是不同的。例如1001，表示一千零一。我们称这是一个四位（十进制）数。

一般地讲，任何十进制数

D3 D2 D1 D0

都可以写成基数十的各次幂的和式，即

$$D3\ D2\ D1\ D0 = D3\times10^3 + D2\times10^2 + D1\times10^1 + D0\times10^0$$

可见同样一个数字，放在最高位与最低位的含义是不同的，D3有表示$10^3$的权，D0有表示$10^0$的权。上式我们又称为按权展开式。

### 二、二进制数

在电子计算机中通常并不采用十进制数，而是采用二进制数。因为电子计算机中使用高电平和低电平来表示两个不同的数码：0，1。一个二进制数的按权展开式如下：

$$B3\ B2\ B1\ B0 = B3\times2^3 + B2\times2^2 + B1\times2^1 + B0\times2^0$$

在这种数制中，1001不是表示一千零一，而是表示九。

### 三、十六进制数

这种数制中有十六个不同的数字：0，1，2，3，4，5，6，7，8，9，A（相应于十进制数中的10），B（11），C（12），D（13）E（14），F（15）。

它的按权展开式如下：

$$H3\ H2\ H1\ H0 = H3\times16^3 + H2\times16^2 + H1\times16^1 + H0\times16^0$$

在微型计算机中经常使用这种十六进制数制，其理由如下：

1．它与二进制数之间的转换比较方便。例如二进制数

1001 1100B*

---

* B（Binary）表示二进制数。同样D（Decimal）和H（Hexcadecimal）分别表示十进制数和十六进制数。

$$= (1\times2^3+1\times2^0)\times16^1+(1\times2^3+1\times2^2)\times16^0$$
$$=9CH$$

即每四个二进制数对应于一个十六进制数。微型机中的二进制数通常是8位或8位的整倍数（16，24，32位），则相应的十六进制数为2，4，6，8位。

2．使用二进制，书写太长，不易记忆，且念起来不易懂。而使用十六进制可以弥补上述缺点。

**四、八进制数**

它的特性与16进制数相似，并且在近代微型机中较少使用，故不予介绍。

**五、二—十进制数**（BCD码）

在这种数制中，用二进制数来表示十进制数字。例如

$97D = 10010111BCD$

$10010111 = (1\times2^3+1\times2^0)\times10^1+(1\times2^2+1\times2^1+1\times2^0)\times10^0$

但须注意这种数制的优点是既照顾了人们使用十进制数的习惯，又考虑到计算机的特点。缺点是不便于运算。由于微型计算机中的运算比较简单，因此经常采用这种数制，特别是微型机化仪器的输入和输出。

**六、不同数制间的换算**

1．二翻十。即把二进制数换算成十进制数。这种方法比较简单，利用二进制数的按权展开式，将二进制数按权相加，就得到等值的十进制数。

例：　$1101B = 1\times2^3+1\times2^2+0\times2^1+1\times2^0 = 13D$

2．十翻二。即把十进制数换算成二进制数。现举例说明其方法：

例：　　求13D的二进制数

| | | |
|---|---|---|
| $13\div2=6$ | 余数1 | （最低位） |
| $6\div2=3$ | 余数0 | |
| $3\div2=1$ | 余数1 | |
| $1\div2=0$ | 余数1 | （最高位） |

结果：1101B

在微型计算机中常有一些实用的子程顺用于进行这类运算。

## 1.2 邏輯电路

通常一个逻辑电路有一或两个以上的输入，一个输出。不同性质的逻辑电路，输出与输入之间有不同的关系，这种关系称为逻辑函数。输入或输出的状态只有两种：高电平和低电平。通常，我们用逻辑1（即有效）来表示高机平，逻辑0（即无效）表示低电平，这称为正逻辑，大多数微型机均使用正逻辑。反之，用逻辑1表示低电平，逻辑

0 表示高电平，称为负逻辑，少数微型机如 Intel 4004/4040 使用负逻辑。本书中使用正逻辑。

下面介绍经常用的逻辑电路。

## 一、与门 (AND gate)

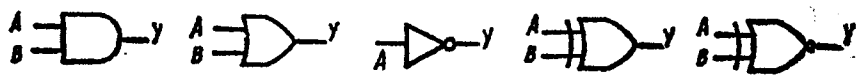
1. 符号：如图 1.1 (a) 所示。
2. 含义：只有当所有输入 (A 和 B) 都为 1 状态时，输出 y 才为 1。
3. 逻辑式：$y = A \times B$ 或 $y = AB$。
4. 真值表 (逻辑函数的另一种表示法)：如表1.1所示。

## 二、或门 (OR gate)

1. 符号：如图1.1 (b) 表示。
2. 含义：只要输入中有一个为逻辑 *1*，输出即为逻辑 1。
3. 逻辑式：$y = A + B$
4. 真值表：如表1.1所示。

**表 1.1 常用逻辑电路真值表**

| 輸 入 | | 輸 出 | | | | |
|---|---|---|---|---|---|---|
| A | B | 与門 | 或門 | 非門 ($Y=\overline{A}$) | 异 門 | 同 門 |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 1 |




(a) 与門　(b) 或門　(c) 非門　(d) 异門　(e) 同門

图 1.1 常用邏輯电路

## 三、非门 (反相器—inverter)

1. 符号：如图1.1 (c) 所示。
2. 含义：输出与输入状态相反。
3. 逻辑式：$y = A$
4. 真值表：如表1.1所示。

## 四、异门、 (cXclusiyc—OR gate)

1. 符号：如图1.1 (d) 所示。

2．含义：输入信号中有奇数个1，输出为逻辑1。反之，为逻辑0。

3．逻辑式：$y = A \oplus B$

4．真值表：如表1.1所示。

### 五、同门 (eXclusive—NOR gate)

1．符号：如图1.1 (e) 所示。

2．含义：输入信号中有偶数个1，输出为逻辑1。反之，为逻辑0。

3．逻辑式：$y = A \overline{\oplus} B$ 或 $y = \overline{A \oplus B}$

4．真值表：如表1.1所示。

### 六、摩根定理

摩根定理是逻辑运算中最常用的定理。即

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

$$\overline{A+B} = \overline{A} \cdot \overline{B}$$

读者可以用真值表来证明这个定理。此定理表明，可以把逻辑“与”运算转换成逻辑“或”运算。反之亦然。

## 1.3 运算电路 (Arithmeetic circuit)

### 一、一位（二进制）数加法与半加法器 (Half addet)

两个一位的二进制数A与B相加，有四种情况：

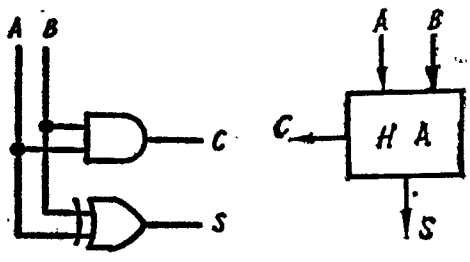
| | (1) | (2) | (3) | (4) |
|---|---|---|---|---|
| A | 0 | 0 | 1 | 1 |
| + B | + 0 | + 1 | + 0 | + 1 |
| C S | 0 0 | 0 1 | 0 1 | 1 0 |
| | ↓ ↓ | ↓ ↓ | ↓ ↓ | ↓ ↓ |
| | C S | C S | C S | C S |

其结果如上，其中S是本位和，C是(对下一位的）进位。由上节可知：

$$S = A \oplus B$$

$$C = A \times B$$

由此可得逻辑图，如图1.2(a) 所示。其方框图如图1.2 (b) 所示，通常叫收半加法器。




(a) 电路　　(b) 框图

图 1 2　半加法器

## 二、多位（二进制）数加法与全加法器（Full addet）

先介绍多位数相加的过程。设有两个 4 位的二进制数A和 B，A = 1011，B = 1001，试求A + B = ?

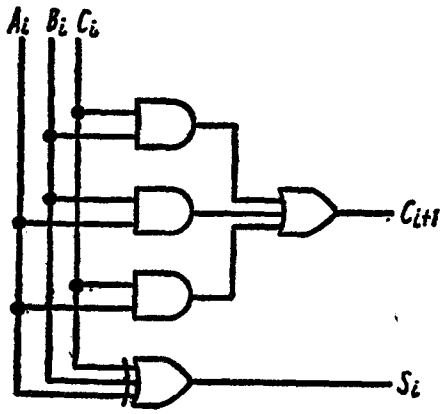
下面按照小学生的习惯进行演算，即逐位进行运算。先从最低位开始，$A_0 + B_0 = C_1S_0$，$S_0$ 是本位和，$C_1$是对下一位的进位。其次$C_1 + A_1 + B_1 = C_2S_1$，依此类推。

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $C_1$ | 1← | | 0 ← | | 1 ← | | 1 ← | |
| $A_1$ | | | 1 | | 0 | | 1 | 1 |
| $+B_1$ | | | 1 | | 0 | | 0 | 1 |
| | | 1 | | 0 | | 1 | | 1 |
| $S_1$ | | ↑ | 0 | ↑ | 1 | ↑ | 0 | ↑ 0 |
| | | $C_4$ | ↑ | $C_3$ | ↑ | $C_2$ | ↑ | $C_1$ ↑ |
| | | | $S_3$ | | $S_2$ | | $S_1$ | $S_0$ |

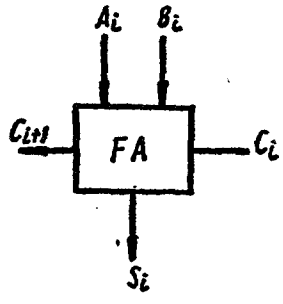
可见，多位数相加与一位数相加的差别在于：前者为三个一位数相加，后者只有两个一位数相加。三个一位数相加有八种情况，其结果如表 1.2 所示。其逻辑式如下：

$$S = A_i \oplus B_i \oplus C_i$$

$$C_{i+1} = A_iB_i + B_iC_i + C_iA_i$$




(a) 电路




(b) 框图

图 1.3 全加法器

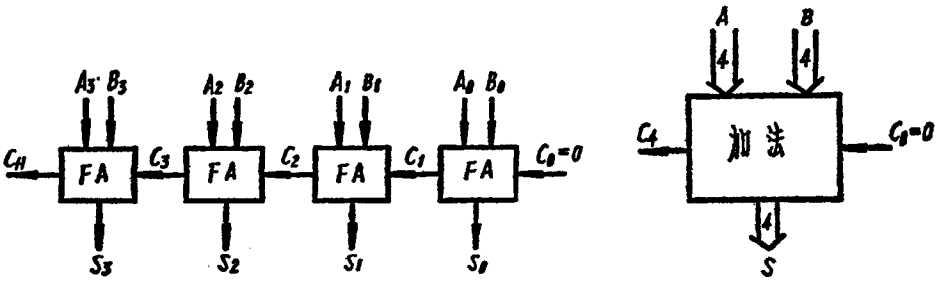
表 1.2 全加法器真值表

| 輸入 | | | 輸出 | |
|---|---|---|---|---|
| $A_i$ | $B_i$ | $C_i$ | $C_{i+1}$ | $S_i$ |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |

图 1.3 (a) 示出其逻辑图，图 1.3 (b) 示出其方框图，通常叫做全加法器。由此可以得到两个 4 位数相加的二进制加法器，如图1.4 (a) 所示。将此图简化可得图 1.4 (b) 所示的方框图。

## 三、二进制数减法与补码运算

按照小学生的演算法则进行二进制数减法并不困难，但要用电子路线来实现减法要比加法麻烦得多。因此人们提出一种新的表示数的形式——补码。




(a) 电路　　(b) 框图

图 1.4 二进制加法器

如前所述，四个二进制数字可以表示 16 个十进制数—0 到 15。现在重新规定其含义，用来表示另外 16 个数，如表 1.3 所示。0 到 7 八个数的表示法与以前相同；所不同的是它能表示负数。以-1为例，表中用 1111 来表示。数 1111 加 1 得10000，最高（第五）位 1 将被丢失而不起作用。所以，数 1111 补上一个 1 得 0000，即数 1111 对 0000 来说缺 1，因而可将 1111 看作-1，其他类推。在补码表示法中，最高位可以看作符号位，0 表示正，1 表示负。由此，利用补码进行减法运算可以将减法转换为加法，例如

5 − 3 = 5 (-3) =？其运算过程如下：

$$\begin{array}{r} 5 \\ +(-3) \\ \hline 2 \end{array} \qquad \begin{array}{r} 0101 \\ +1101 \\ \hline 10010 \end{array}$$

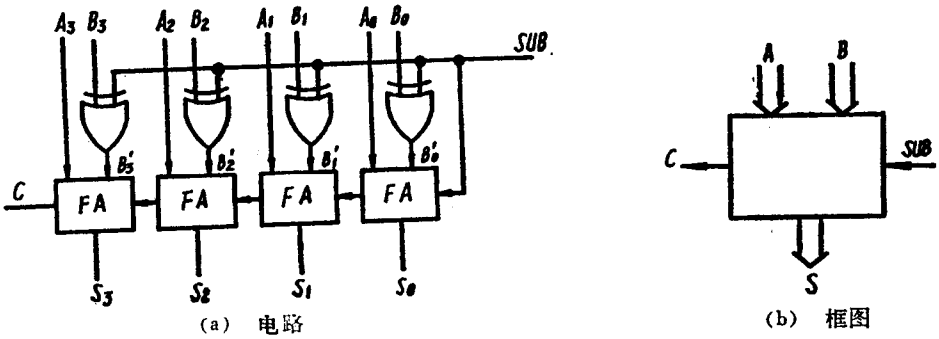
↑ 丢失

但是如何将 3 (0011) 转换成-3 (1101)？简单地说，就是将 0011 进行“求反加 1”运算：

$$\begin{array}{rl} \text{求反} & \begin{array}{cccc} 0 & 0 & 1 & 1 \\ \downarrow & \downarrow & \downarrow & \downarrow \\ 1 & 1 & 0 & 0 \end{array} \end{array}$$

**表 1.3 数的补码表示**

| 数 | 补 码 |
|---|---|
| -8 | 1000 |
| -7 | 1001 |
| -6 | 1010 |
| -5 | 1011 |
| -4 | 1100 |
| -3 | 1101 |
| -2 | 1110 |
| -1 | 1111 |
| 0 | 0000 |
| 1 | 0001 |
| 2 | 0010 |
| 3 | 0011 |
| 4 | 0100 |
| 5 | 0101 |
| 6 | 0110 |
| 7 | 0111 |




(a) 电路　　(b) 框图

图 1.5 加法/减法器

加1

$$\begin{array}{r} 1\ 1\ 0\ 0 \\ +\quad\ \ 1 \\ \hline 1\ 1\ 0\ 1 \end{array}$$

在电路上又如何实现？图 1.5 (a) 示出了利用补码运算的加法/减法器。当进行加法运算时，令 SUB = 0，则 $C_0 = 0$，且 $B_i' = Bi$ (i = 0—3)，它与图 1.4 的二进制加法器相同。当进行减法运算时，令 SUB = 1，则 $C_0 = 1$，且 $Bi = \overline{B_i'}$ (i = 0 - 3)。即通过异门将减数 B 求反，利用 $C_0 = 1$，得到加 1 操作。图1.5 (b) 示出其方框图。

## 1.4 触发器与寄存器 (Flip-flop and Register)

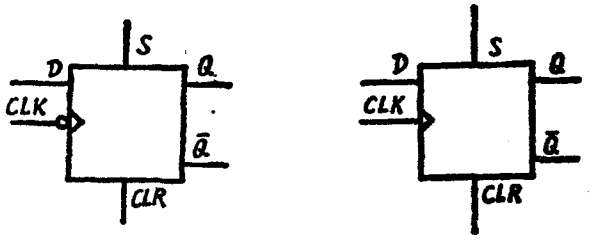
### 一、触发器

触发器是一种常用的数字电路。它可以作为一种无触点开关，也可以存放一个（二进制）位 (bit) 的信息。常用的触发器有：

1. D 触发器

1) 符号：如图 1.6 (a) 所示。图中输入信号有 D—数据，和 CLK—时钟，CLR—复位（清零）信号，S—置位（置 1）信号；输出信号有 Q 和 $\overline{Q}$。

2) 操作：如表 1.4 所示。当 CLK 正跳变时，Q = D；当 CLK 不是正跳变时，Q 保持原状。当 CLR = 1 时，Q = 0；当 S = 1 时，Q = 1。




(a) CLK正跳变有效　　(b) CLK负跳变有效

图 1.6　D触发器

表 1.4

| 输入 | | 输出 |
|---|---|---|
| CLK | D | Q |
| ⌐ | 1 | 1 |
| ↑ | 0 | 0 |
| 其它 | × | 原状 |

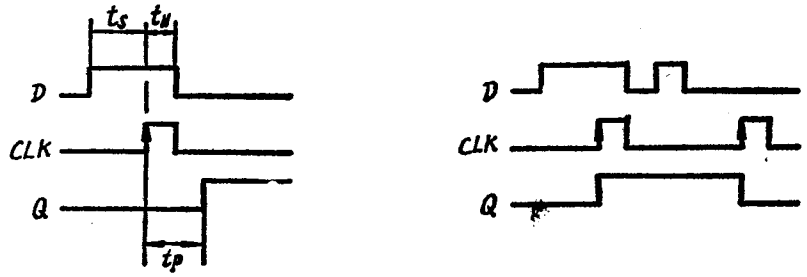
图 1.7 示出 D 触发器操作的时间图。图 1.7 (a) 表明，数据线D的建立必须领先于 CLK 正跳变，这段时间称为建立 (Setup) 时间 tso数据线D 的撤除必须滞后于 CLK 正跳变，这段时间称为保持 (Hold) 时间 $t_H$ 如果不满足上述条件，D 触发器就不能正确动作。输出信号 Q的变化滞后于 CLK 正跳变，这段时间称为传播延迟 (Propagation delay) tpo在以后的叙述中，将忽略这些特性。图1.7(b)示出了不考虑 tp 的操作时间图。

图 1.6 (b) 示出另一种 D 触发器，与前者的差别仅在于 CLK 端上多一个小圆圈。在数字电路中，小圆圈表示反相，因而这种 D 触发器的 CLK 信号在负跳变时为有效。同理，若 CLK，S 端上也加上小圆圈，则这些信号在 0 电平为有效。

2. JK 触发器

1）符号：如图 1.8 所示。J 和 K 为控制输入端。

2）操作：表 1.5 示出了当 CLK 发生正跳变时输出与输入的关系。




(a) 考虑 $t_p$　　(b)不考虑 $t_p$

图 1.7　D触发器的操作时間图

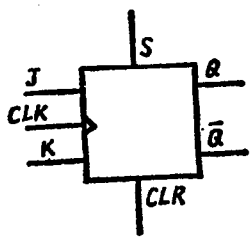



图 1.8　JK触发器

表 1.5

| 輸　入 | | 輸　出 |
|---|---|---|
| J | K | Q |
| 0 | 0 | 不　变 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 翻　轉 |

## 二、寄存器

寄存器由若干个（通常是 4，8，16个等）触发器构成。它可以存放一个 4 位、8 位、或 16 位的字，此外还能执行加 1 、减 1 、移位和其他的操作。

1．计数器

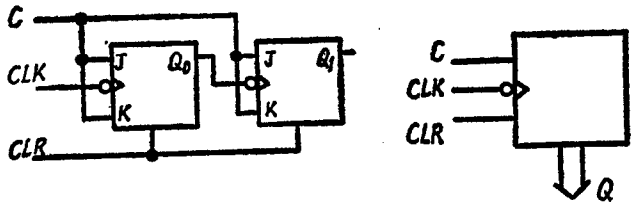
1）电路图及方框图：如图 1.9 (a)　(b) 所示。

2）操作：当 CLR = 1 时，所有触发器被复位，即 $Q_1 = 0$ 、$Q_0 = 0$ 。当 C = 0 时，所有 J、K 端都为 0，施加时钟脉冲 CLK 并不能改变 Q 的状态，即 Q保持原状。当C = 1 时，所有 J、K 端都为 1，因而每一个时钟脉冲都使计数器加 1，其时间图 如 图 1.9 (c) 所示。可知 C 是控制输入端，控制计数器是否对 CLK 进行计数。

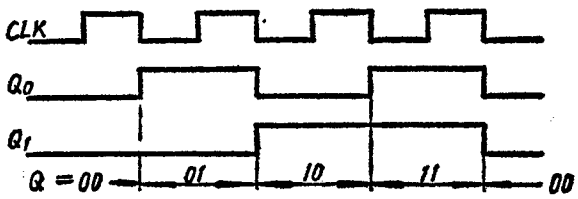
2．缓冲与移位寄存器

1）电路图：如图 1.10 所示。

2）操作：当 CLR = 1 时，所有触发器 被 复位，Q = 0 。当 LOAD = 1 时，在 CLK 正跳瞬间，将 X 装入 Q，即Q = X。当SHL = 1 时，在 CLK 正跳变瞬间，Q 向左移位，即 $Din \rightarrow Q_0$，$Q_0 \rightarrow Q_1$，$Q_1 \rightarrow Q_2$，$Q_2 \rightarrow Q_3$，$Q_3$ 丢失。当 LOAD = 0 和 SHL = 0 时，在 CLK 正跳变瞬间，Q 保持不变。

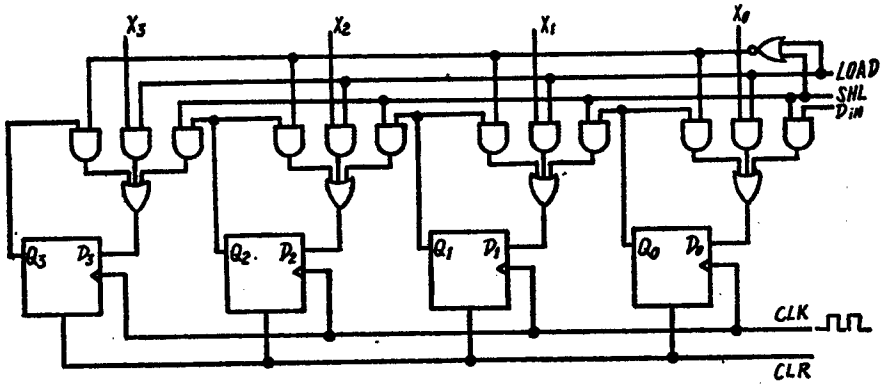



(a) 电路　　(b) 框图

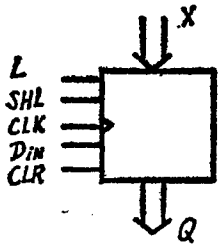



(c) 时間图

图 1.9 計数器




(a) 电路




(b) 框图

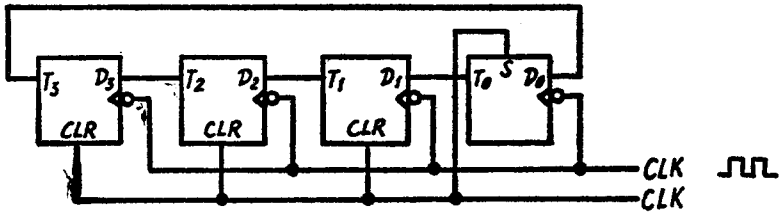
图 1.10 緩冲与移位寄存器

3. 环形计数器

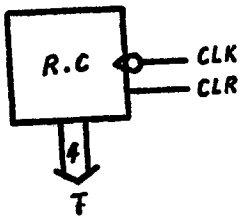
1) 电路图：如图 1.11 (a) (b) 所示。

2) 操 作：当CLR = 1时，$T_3 = 0$，$T_2 = 0$，$T_1 = 0$，$T_0 = 1$，即 T = 0001。当送

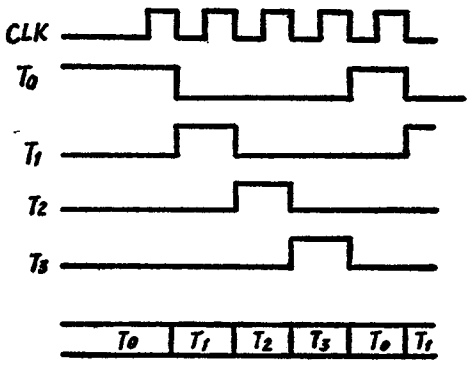
入时钟脉冲 CLK 时，每出现一次正跳变，T 依次为 0010→0100→1000→0001 ……。如图 1.11 (c) 所示。




(a) 电路




(b) 框图




(c) 时間图

图 1.11 环形計数器

### 三、三态输出寄存器

由于一般的数字器件只有两个状态 1 和 0，所以每条信号线只能由一个器件来驱动，从而使信息传输线的数目大为增多。为了减少信息传输线的数目，简化控制系统，将属于不同来源的信息或数据在一组统一的传输线上分时 (Multiplexed) 传送到不同的目的地，这组传输线 (通常是 8 条或 16 条) 称之为总线 (Bus)。在某一时刻，只能有一个器件驱动总线，这个器件的输出可以呈 1 或 0。其他器件不能呈此二状态，它们必须呈第三种状态—高阻抗，即浮动状态。也就是说，好像它们的输出被开关所断开，对总线状态不起作用。

为了将普通器件的二态输出更改为三态输出，通常使用图 1.12 所示的三态开关。当 E = 1 时，$D_{out} = D_{in}$，此时 $D_{out}$ 线由该器件来驱动，当 E = 0 时，$D_{out}$ 呈高阻抗状态，该器件对它不起作用，而是由其他器件驱动此线。

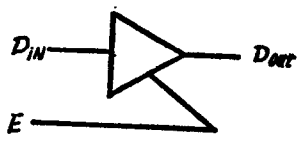
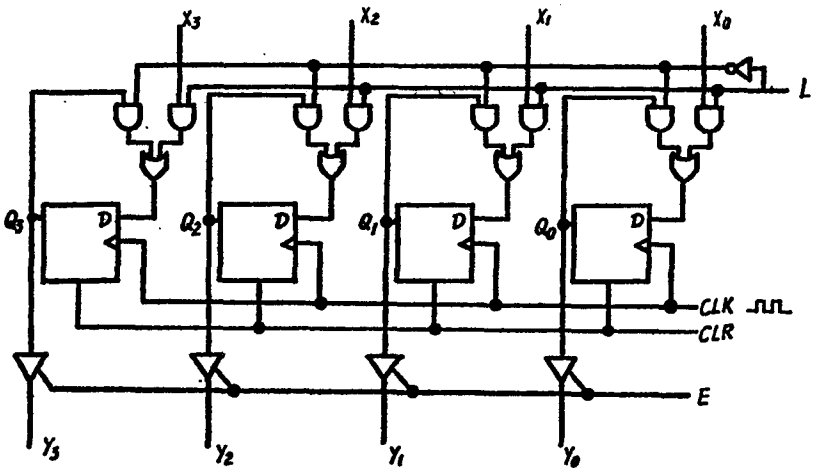


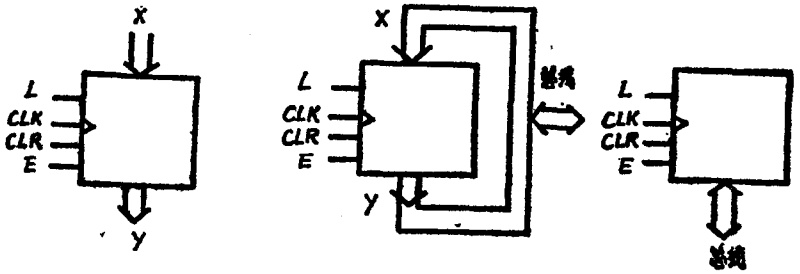

图 1.12 三态开关

图 1.13 示出具有三态输出的缓冲寄存器。当 E = 1 时，输出 Y = Q；当 E = 0时，输出 Y = 高阻抗，与各个触发器的状态 Q 无关。对于具有

三态输出的缓冲寄存器，可以将其数据输出线和数据输入线合并在一起，以便挂到总线上去，如图 1.13（c）所示。




（a） 电路




（b） 框图　　（c）具有数据总线的寄存器

图 1.13 具有三态输出的缓冲寄存器

图 1.14 示出了四个寄存器挂在一条总线 W 上。如果控制信号中只有 $E_A$、$L_B$ 等于 1，其他均为 0，则在 CLK 正跳变时，寄存器 A 的内容将装入寄存器 B。因为只有 $E_A = 1$，其他器件的 E 信号为 0，总线的状态由 A 来决定。换句话说，A 驱动总线。因为只有 $L_B = 1$，总线上内容只装入 B。通俗地说，E 是“放出”控制信号，L 是“装入”信号。

如果只有 $E_B = 1$，$L_C = 1$，$L_D = 1$，其他控制信号均为 0，则在 CLK 正跳变时，寄存器 B 的内容将装入寄存器 C 和 D。

对于其一时刻，只能有一个器件的 E 信号有效，否则将有多个器件“争夺”总线而发生误操作。

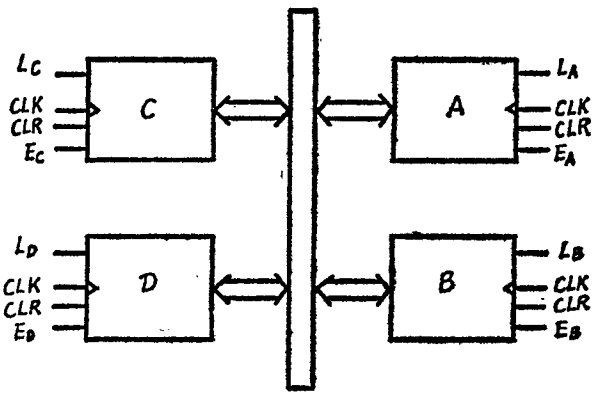



图 1.14 挂到总线上的寄存器组

## 1.5 存储器 Memory

存储器是由若干个寄存器组成，每个寄存器存放一个二进制字（在微型计算机中，通常是8位称做一个字节）存储器用来存放程序和数据。有多种介质可以用来制作存储器，如半导体、磁芯、磁盘、磁带等等。微型计算机的内存储器（简称内存）主大使用半导体存储器，本节内容仅限于这种半导体存储器。

存储器可以分为下列两大类：

1．只读存储器—ROM　它用来存放程序、表和常数。它的内容只能被微处理器读出，而不能被微处理改变，也不会因断开电源而丢失。ROM 通常又分为下列三 种：

1）ROM　它的内容由芯片制造厂使用掩模编程而被永久性地固定下来。由于其工作可靠，在大量生产时价格低廉，在产品已被定型而大量生产时，常常使用ROM。

2）PROM（可编程序只读存储器）　它的内容由用户利用 PROM 写 入 器（或称编程器）而被固定下来，一旦被编定，就不能再被改变，只允许对未曾编 程 的 位 进行编程。在小批量生产时，常使用 PROM。

3）EPROM（可擦除的可编程序只读存储器）　它的内容被用户编定后，可以用紫外线擦掉而再次被编程。虽然 EPROM 的可靠性不如前两者，但是由于它的灵活性，常被使用于产品的研制阶段。

近年来，又出现一些其他类型的 ROM，如 EAROM，EEPROM，此处不作介绍。

2．读写存储器—RAM　它的内容可以由微处理器写入和读出。RAM可分为两种：

1）静态 RAM　每位信息存放在一个触发器中，只要电源有电，其信息能一直被保持。

2）动态 RAM　它利用门—基片电容上的电荷来存放信息。这种电荷将 在 几毫秒内耗散，因而每隔两毫秒需对动态 RAM 的内容刷新一次。

动态 RAM 的优点是器件密度高，价格低，待机功耗低。它的缺点是必须有刷新电

路，每个动态 RAM 芯片的字长仅为一位，因而 8 位微型计算机（即使内存容量很小）要求最少使用八个芯片。

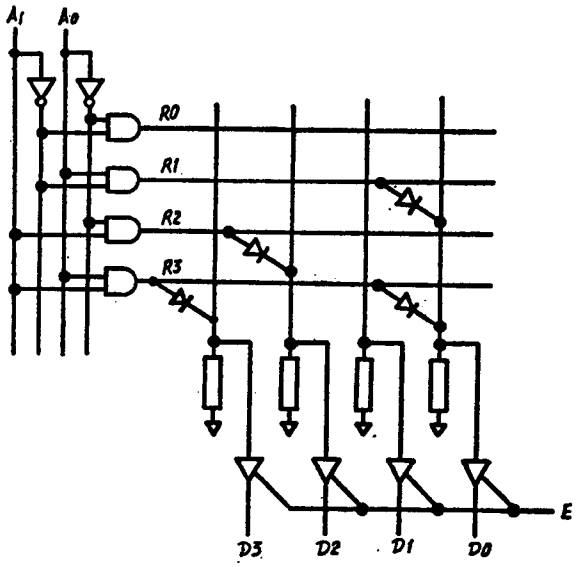
## 一、只读存储器 ROM

只读存储器的原理图如图 1.15（a）所示。每一行可以看作一个存储单元。用来存放一个二进制字。图中共有四行。即这一存储器有四个单元，可以存放四个二进制字。图中共有四列。表示每个字的字长为 4 位。

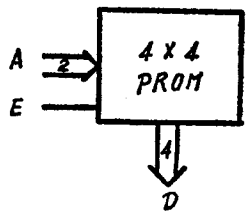
如果图中的地址线A1 = 0 、A0 = 1、E = 1，则第二条线 R1 为高电平，从而通过二极管而使 D0 为高电平，D1、D2、D3 相应的列中没有插入二极管，故为低电平，即

$$D = D3D2D1D0 = 0001$$

其他线 R0，R2，R3 都为低电平，故对输出 D 没有影响。同理，A1，A0 呈不同值时选中不同的水平线呈高电平，从而使数据线上呈现不同单元的内容。有二级管的位，相应于逻辑 1；反之，没有二级管的位，相应于逻辑 0。




（a）电路




（b）框图

| A | D |
|---|---|
| 0 0 | 0 0 0 0 |
| 0 1 | 0 0 0 1 |
| 1 0 | 0 1 0 0 |
| 1 1 | 1 0 0 1 |

（c）存储单元的内容表

图 1.15 只讀儲存器的原理图

如果图中矩阵中二级管的有无可由用户来决定，而且可以再次被改变，则可以认为这是一种 EPROM 的模型。

图 1.15（b）示出它的方框图。数据输出线 D 的位数表示字长，通常是 8 位；地址输入线 A 的位数 N 决定存储字数，字数 = $2^N$。

图 1.15（c）用表格来表示 ROM。在该图中，ROM 的内容 R = $(A1A0)^2$。可见，这样的 ROM 可以用作平方的表格。推而广之，ROM 还可以存放各种函数（包括逻辑运算函数）以及其他的常数等。

图 1.16 示出了目前常用的 2716 型 EPROM（2 K × 8）的引脚图。在 28 根引脚中，有 11 根地址线，8 根数据输出线，以及 $V_{CC}$ 和地线。这些引脚易于理解，不再进一步解释。现介绍其余三根引脚的作用，如表 1.6 所示。当 $V_{PP}$ = + 5 V 时，为读数方式；当 $V_{PP}$ = +25V 时，为编程（即写入—Programming）方式。

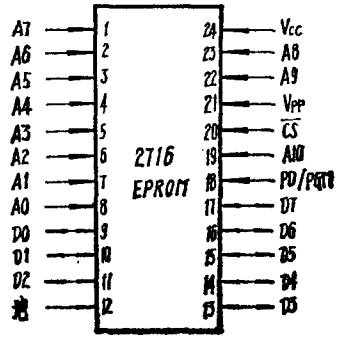



图 1.16　2716 型 EPROM（2 K×8）的引脚图

**表 1.6　工作方式选择**

| 方式 \ 引脚 | PD/PGM (18) | CS (20) | $V_{PP}$ (21) | 数据线状态 |
|---|---|---|---|---|
| 讀 | 0 | 0 | + 5V | $D_{OUT}$ |
| 未选中 | × | 1 | + 5V | 高阻抗 |
| 待机 | 1 | × | + 5V | 高阻抗 |
| 編程 | ⎍ | 1 | +25V | $D_{IN}$ |
| 校驗編程內容 | 0 | 0 | +25V | $D_{OUT}$ |
| 禁止編程 | 0 | 1 | +25V | 高阻抗 |

下面扼要介绍这几种工作方式：

1．读——此时可将其中某一个单元的内容读至数据线上。

2．未选中——因为 CS = 1，数据输出线呈高阻抗，即该芯片不起任何作用。

3．待机（Power Down）——当PD/PGM = 1 时，芯片处于待机方式。这种方式与不选中相似，唯一的差别在于：待机方式的功耗仅为最大运行功耗的四分之一。

4．编程——若要对 EPROM 某个单元进行写入，则应对 PD/PGM 引脚输入一个正脉冲，脉冲宽度为 50 毫秒左右，对 2K 个单元的写入编程总共需要 100 秒左右。由于施加的脉冲电平与 TTL 兼容，不需要施加高电压脉冲，因而不必使用专门装置，而用单板机这样的简单系统即能编程。

5．校验编程内容——在编程完毕后，将其中的内容读出并进行比较，以决定编程的内容有否出错。此方式与读方式相似。

6．禁止编程——此时禁止将数据线上内容写入 EPROM。