第 1 章

单片机基础知识

1.1 概述

1.1.1 单片机发展历程

单片机专业名称为 Micro Controller Unit（微控制器件），是由 Intel 公司发明的，最早的系列是 MCS-48，后来有了 MCS-51。常说的 51 系列单片机就是 MCS-51（Micro Controller System），这是一种 8 位的单片机。后来 Intel 公司把它的核心技术转让给世界上很多小公司，所以就有许多公司生产 51 系列兼容单片机，例如，飞利浦的 87LPC 系列、华邦的 W78 系列、达拉斯的 DS87 系列、现代的 GSM97 系列等。目前在我国比较流行的就是美国 ATMEL 公司的 89C51，它是一种带 Flash ROM 的单片机，本书的内容就是基于该型号的单片机来开展相关设计和实验仿真的。讲到这里，也许有的读者会有疑问，为什么平时在各种书上看到全是讲解 8031、8051 等型号的单片机，二者又有什么不同呢？其实 8031、8051 与 89C51 同属于一个系列，只是 89C51 的单片机更新型一些。目前 89C51 已经被 89S51 代替，89S51 兼容 89C51，同时近几年也出现了像宏晶公司研发生产的 STC89C 系列的 51 内核的单片机，该单片机最大的特点就是可以利用串口线在线下载程序，使用非常方便，因此该系列单片机近几年占据了国内 51 单片机的大部分市场。

1.1.2 几种常见的单片机

除了上述介绍的单片机，还经常在各种刊物上看到 AVR 系列和 PIC 系列单片机，这么多的单片机，应该先学哪一种呢？在没有学习单片机之前，这是一个令很多初学者非常困惑的问题。

AVR 系列单片机是 ATMEL 公司生产的一种 8 位单片机，采用的是 RISC 精简指令集单片机结构，所以其技术和 51 系列有所不同，开发设备和 51 系列也是不通用的，AVR 系列一条指令的运行速度可以达到纳秒级，即每秒 1000000000 次，是 8 位单片机中的高端产品，由于其出色性能，目前应用范围越来越广，大有取代 51 系列的趋势，所以学完 51 系列，还有必要学习 AVR 系列。PIC 系列单片机则是美国 MICROCHIP 公司生产的另一种 8

位单片机，采用的也是 RISC 的指令集，其指令系统和开发工具与 51 系列更是不同，但由于其价格低和性能出色，目前用户越来越多，国内也有很多公司在推广此系列，不过 PIC 的影响力远没有 51 系列大，所以作为初学者，51 系列仍然是首选。以上几种只是比较常见的系列，日常应用中还有许多其他公司生产的各种各样的单片机，例如，Motorola 的 MC68H 系列、TI 的 MSP430C 系列、德国的西门子 PLC 等，都有各自的结构体系，并不与 51 系列兼容，这里不再深入介绍，感兴趣的读者可在入门后自己研究，下面介绍 51 系列单片机的结构及组成。

1.1.3 单片机的结构及组成

单片机到底是什么，究竟能做什么呢？其实单片机就是一种能进行数学和逻辑运算，根据不同使用对象完成不同控制任务的面向控制而设计的集成电路，就像在计算机中可以用不同的软件在相同的硬件上实现不同的功能。单片机其实也是如此，同样的芯片可以根据不同的要求做出截然不同的产品，只不过计算机是面向应用的，而单片机是面向控制的，例如，控制一个指示灯的亮和灭，控制一台电机的启动和停止等。那么单片机的内部究竟由哪些部件组成呢？大家都知道计算机中有很多的部件，如 CPU（中央处理器）、RAM（内存条）、ROM（程序存储器）、输入/输出设备（并行口/串行口）等，在单片机中这些部件也都具备，并且全部集成在一块芯片上，这就是单片机名称的由来。如图 1-1 所示为单片机的典型组成结构。单片机提供静态逻辑操作功能，工作频率可以低至 0Hz，支持两种软件编程的节约电源管理模式，一种是空闲模式，一种是电源关闭模式。在空闲模式下，单片机的 CPU 停止工作，而内部数据存储器、定时器、串口和中断系统仍然工作。在电源关闭模式下，单片机保存数据存储器的内容，晶体振荡器工作，其余的部分停止工作，直到有中断或硬件复位时，单片机芯片的其他部分才能工作。如图 1-2 所示是 AT89S51 的内部结构图。

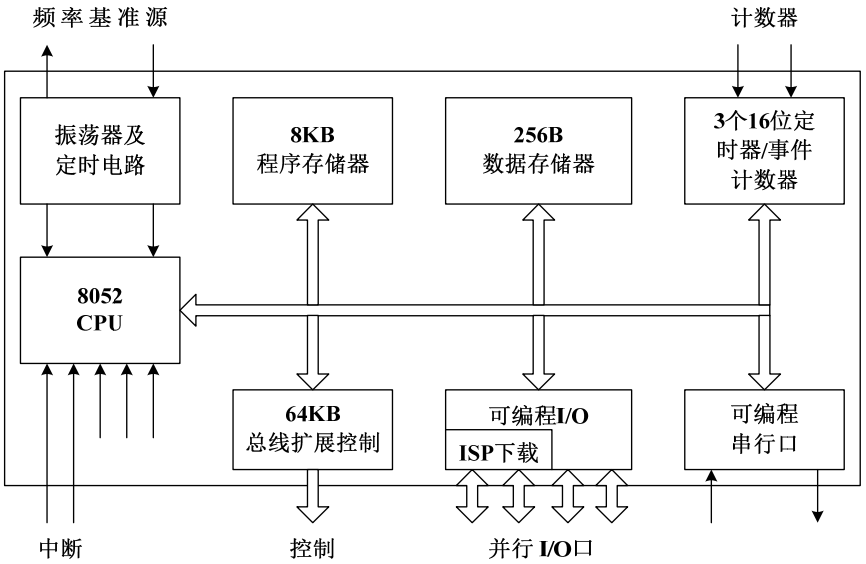

图 1-1 单片机的典型组成结构

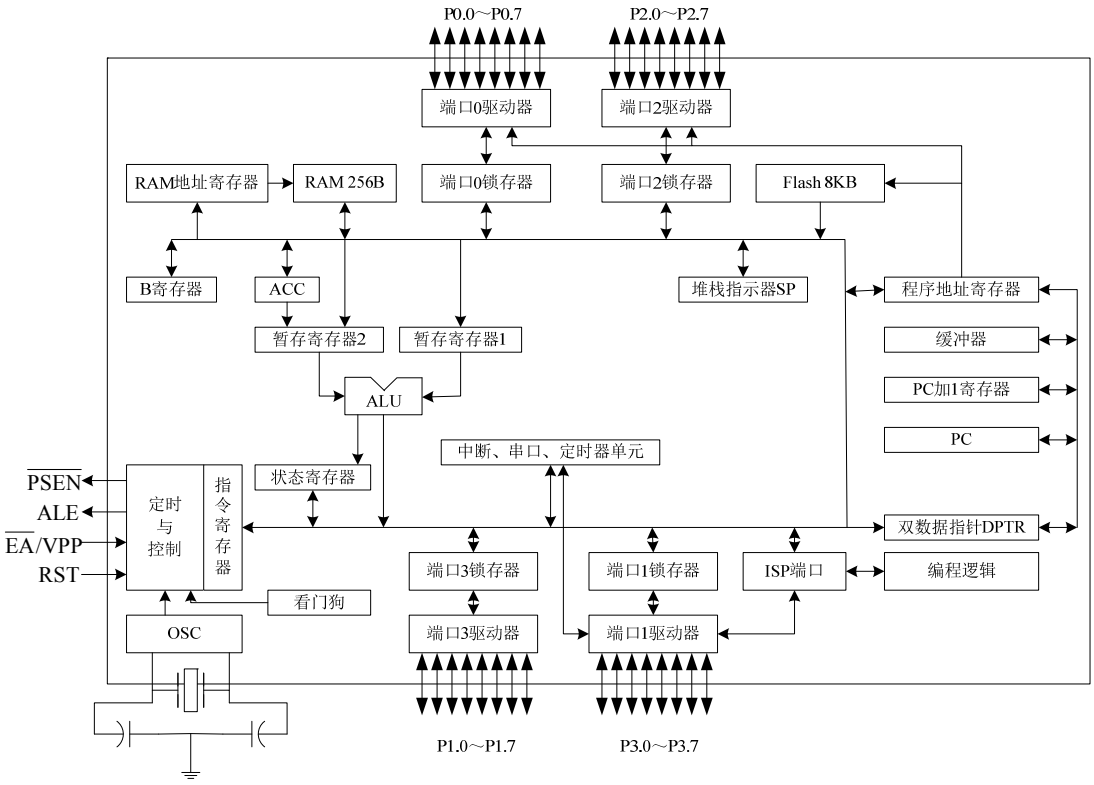

图 1-2　单片机 AT89S51 的内部结构图

1.2　数的进制及位和字节的含义

1.2.1　数制及其转换

1．3 种基本数制

数制指的是数据的表现形式。在计算机中常用的数制有二进制数、十进制数和十六进制数。无论是什么数制，将这个数制所包含的数码个数称为基，将各个数字所具有的数值称为权。

（1）十进制数（Decimal）

十进制数是日常生活中最为常用的数制形式。十进制数的基为 10，其包含 10 个数码，即 0、1、2、3、4、5、6、7、8、9。十进制数的权是以 10 为底的幂，每个数所处位置不同，则其代表值不同。前一个数的权值为其相邻后一个数权值的 10 倍。

例如，一个十进制数为 2345.678D，则其代表的数值为：

$2345.678D=2\times10^{3}+3\times10^{2}+4\times10^{1}+5\times10^{0}+6\times10^{-1}+7\times10^{-2}+8\times10^{-3}$

从左到右各位代表的权值为：

10^{3}　　10^{2}　　10^{1}　　10^{0}　　10^{-1}　　10^{-2}　　10^{-3}

也就是平常所说的千、百、十、个、0.1、0.01、0.001。

（2）二进制数（Binary）

二进制数是计算机中的基本数据形式，所有数据在计算机中都以二进制数形式进行存储和运算。二进制数的基为 2，在二进制数中只包含 0 和 1 两个数码。二进制数的权是以 2 为底的幂，每个数所处位置不同，则其代表值不同。前一个数的权值为其相邻后一个数权值的 2 倍。

例如，一个二进制数为 1011.1001B，则其代表的十进制数值为：

$1001.1011B=1\times 2^3+0\times 2^2+0\times 2^1+1\times 2^0+1\times 2^{-1}+0\times 2^{-2}+1\times 2^{-3}+1\times 2^{-4}$

$=8+0+0+1+0.5+0.25+0.125+0.0625$

$=9.9375$

从左到右各位代表的权值为：

2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
8	4	2	1	1/2	1/4	1/8	1/16

二进制数之所以能够成为计算机工作基本数据形式，这是由计算机中的电路工作原理决定的。计算机中的电路通常只有两种状态，在二进制数中可以用 1 代表电路中的高电平，用 0 代表电路中的低电平；或者用 1 代表电路中的导通，用 0 代表电路中的截止。采用二进制数可以方便地对电路进行计数工作。

（3）十六进制数（Hexadecimal）

十六进制数的基为 16，在十六进制数中包含了 16 个数码，各个数码表示如下：

0　1　2　3　4　5　6　7　8　9　A　B　C　D　E　F

十六进制数的权是以 16 为底的幂，与二进制数相同，每个数所处位置不同，则其代表值不同。前一个数的权值为其相邻后一个数权值的 16 倍。

例如，一个十六进制数为 F72.C4H，则其代表的十进制数值为：

$F72.C4H=15\times 16^2+7\times 16^1+2\times 16^0+12\times 16^{-1}+4\times 16^{-2}$

$=3954.765625$

从左到右各位代表的权值为：

16^2	16^1	16^0	16^{-1}	16^{-2}

相对于二进制数而言，十六进制数能够用较少的位数表现较大的数值，便于书写和记忆。由于二进制数与十六进制数的转换非常简单，所以在计算机中的数值常常采用十六进制数来表示。在汇编语言中，为了将十六进制数中的 A、B、C、D、E、F 与字母区分开来，书写时常常会在其前面加一个 0。例如 F7H 写作 0F7H。

2．数制间的转换

由于不同数制的特性不同，常常需要将一个数转换为十进制数来理解数值大小，将一个数转换为十六进制数来编写程序，将一个数转换为二进制数来理解对应数据在计算机中的含义。下面介绍计算机中不同数制的转换方法。

（1）转换为十进制数

在前面介绍数制的过程中已经介绍了二进制数和十六进制数转换为十进制数的方法。将一个二进制数或十六进制数转换为十进制数只需将该数各个数位的基与权相乘累加即可。例如：

$10110101B=1\times 2^7+0\times 2^6+1\times 2^5+1\times 2^4+0\times 2^3+1\times 2^2+0\times 2^1+1\times 2^0$

$=181$

$B5H=11\times 16^1+5\times 16^0=181$

（2）十六进制数与二进制数的转换

由于十六进制数的基 16 是二进制数基 2 的 4 次方，所以二进制数与十六进制数的转换十分简单。

十六进制数转换为二进制数时，只需从右至左将各个位上的数以二进制数表示出来即可。二进制数转换为十六进制数时，整数部分需要从右至左依次将 4 个二进制数表示成一个十六进制数，左侧不满 4 个时在前面补 0 到 4 个即可。小数部分自左向右每 4 位一组，最后不满 4 位的在后面补 0，写出对应的十六进制数即可。如表 1-1 所示为 0～15 的十进制数、二进制数、十六进制数的转换关系表。

表 1-1　不同数制的数据转换表

十进制数 D	二进制数 B	十六进制数 H	十进制数 D	二进制数 B	十六进制数 H
0	0000	0	8	1000	8
1	0001	1	9	1001	9
2	0010	2	10	1010	A
3	0011	3	11	1011	B
4	0100	4	12	1100	C
5	0101	5	13	1101	D
6	0110	6	14	1110	E
7	0111	7	15	1111	F

例如，一个二进制数 10010110101B 要转换为十六进制数，其过程如下：

10010110101B=0100 1011 0101=4B5H

一个十六进制数 A3FBH 转换为二进制数的过程如下：

A3FBH=1010 0011 1111 1011B

（3）十进制数转换为二进制数或者十六进制数

将一个十进制数转换为二进制数时，对于十进制数整数部分采用“除 2 取整”的方法，小数部分采用“乘 2 取整”的方法。十进制数转换为十六进制数时可以采用类似的“除 16 取整”以及“乘 16 取整”的方法。由于十六进制数与二进制数之间的转换十分简单，在十进制数转换为十六进制数时，也可以先转换为二进制数继而转换为十六进制数。

下面以一个十进制数 78.8125D 为例来介绍“除 2 取整”以及“乘 2 取整”方法的使用。

78.8125D 整数部分为 78，转换为二进制数时将此数不停除 2 直到商为 0 为止，然后将所有余数逆向整合即可得到想要的二进制数。

2∟ 78 ……余数为 0　低位

2∟ 39 ……余数为 1

2∟ 19 ……余数为 1

2∟ 9 ……余数为 1

2∟ 4 ……余数为 0

2∟ 2 ……余数为 0

2∟ 1 ……余数为 1　高位

如上所示将余数逆向整合得到 78D=1001110B。

小数部分 0.8125 转换为二进制数时将该数的纯小数部分不停乘以 2 直至乘积为 1 为止，将所有结果整数部分一次组合即可得到想要的二进制数。

0.8125×2=1.625 ……整数部分为 1 高位
0.625×2=1.25 ……整数部分为 1 ↓
0.25×2=0.5 ……整数部分为 0 ↓
0.5×2=1 ……整数部分为 1 低位

如上所示将整数顺次整合得到 0.8125D=0.1101B。

将小数部分以及整数部分整合得到 78.8125D=1001110.1101B。

1.2.2 数和物理现象的关系

在数字电路中，可以用一盏灯的亮和灭来表示电平的高和低，即用“1”表示高电平（亮），用“0”表示低电平（灭），如果现在有两盏灯，那么会有几种状态呢？如表 1-2 所示为两盏灯的亮灭与二进制数之间的对应关系。

表 1-2 两盏灯的亮灭与二进制数之间的对应关系

第 一 盏 灯	第 二 盏 灯
0	0
0	1
1	0
1	1

两盏灯的组合可以表示 4 种状态，即 00，01，10，11。这样看来，灯的亮和灭这种物理现象同数字确实有着某种联系，如果把各状态按一定的规律排好，那么电平的高或低就可以用数字来表示了，换句话说，不同的数字可以代表不同数量灯的电平高或低，例如，0000，0001，0010，0011，0100，0101，0110，0111，1000，1001，1010，1011，1100，1101，1110，1111，这 16 种组合就可以代表 4 盏灯的不同状态。

1.2.3 位和字节的含义

在单片机中一盏灯（实际上是一根线）可看作一位，有 0 和 1 有两种状态，分别对应电平的低和高，是单片机最基本的数量单位，用 bit 来表示。8 盏灯（8 根线）有 256 种状态，这 8 盏灯（也就是 8 位）可称为一个字节，用 B（即 BYTE）表示。那么单片机是如何来储存这些数字所代表的字节的状态的呢？后文介绍存储器时将做详细说明。

1.3 51 单片机基本硬件结构

1.3.1 硬件结构

前面提到过单片机的内部结构是由 CPU、ROM、RAM 等组成，现在介绍外部引脚。如图 1-3 所示为单片机的引脚图，这就是实验中要用的 89C51 单片机的外部引脚图。如

表 1-3 所示为 89C51 单片机引脚分配表。

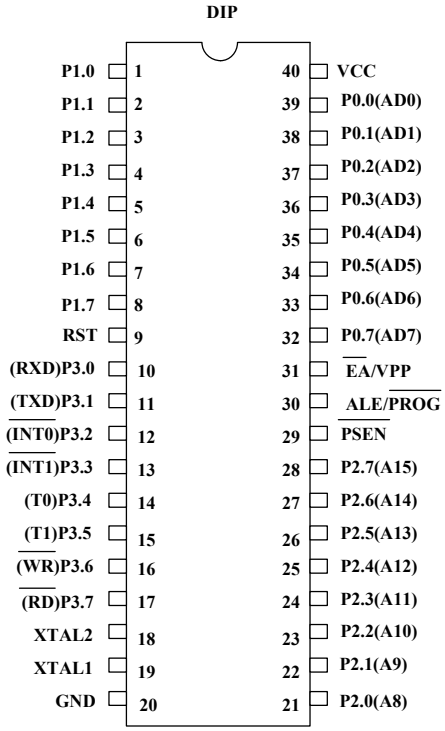

图 1-3　89C51 单片机的引脚图

表 1-3　89C51 单片机引脚分配表

引　脚　号	引 脚 名 称	说　　明
1～8	P1.0～P1.7	端口 P1
9	RST	复位信号输入端
10～17	P3.0～P3.7	端口 P3，该端口具备第二功能，详见 1.3.2 节
18	XTAL2	时钟振荡器输出端，内部振荡器输出端
19	XTAL1	时钟振荡器输入端，内部振荡器输入端
20	GND	电源地
21～28	P2.0～P2.7	端口 P2
29	$\overline{\text{PSEN}}$	外部程序存储器从程序存储器中取指令或读取数据时，该信号有效
30	ALE/ $\overline{\text{PROG}}$	地址锁存信号访问外部存储器时，该信号锁存低 8 位地址；无 RAM 时，此引脚输出晶振的 6 分频信号
31	$\overline{\text{EA}}$/VPP	程序存储器有效地址，EA=1 时从内部开始执行程序；EA=0 时从外部开始执行程序
32～39	P0.7～P0.0	端口 P0
40	VCC	电源正

1.3.2　端口结构分析

从 1.3.1 节的硬件结构中可以看出，89C51 单片机总共有 4 组端口，P0、P1、P2 和 P3，

了解这 4 组端口的结构原理对于日后的编程会有很大的帮助，由于这 4 组端口结构不尽相同，下面分别介绍单片机总的 4 组端口。由于每组端口都是由 8 位组成，故在下面的讲解中，只以每组端口的其中一位来解释。

1．P0 口的结构及工作原理

P0 口字节地址为 80H，位地址 80H～87H。P0 端口 8 位中的一位结构图如图 1-4 所示。

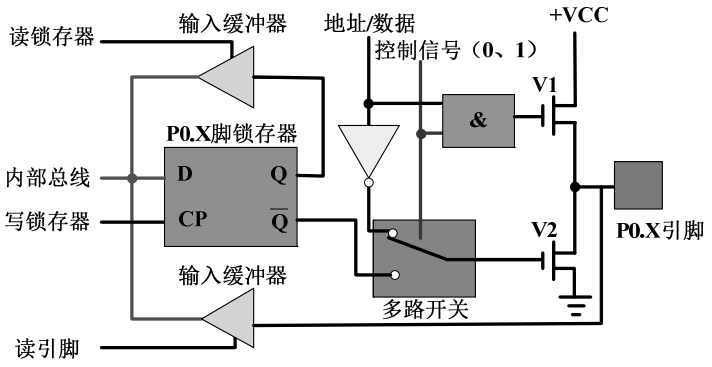

图 1-4　P0 端口位结构图

由图 1-4 可见，P0 端口由锁存器、输入缓冲器、多路开关、一个非门、一个与门及场效应管驱动电路构成。图 1-4 中标号为 P0.X 引脚的图标，表示引脚可以是 P0.0～P0.7 的任何一位，即在 P0 口有 8 个与图 1-4 所示相同的电路组成。下面先介绍组成 P0 口的每个单元部分。

（1）输入缓冲器

在 P0 口中，有两个三态的缓冲器，学过数字电路的读者都知道三态门有 3 个状态，即在其输出端可以是高电平、低电平，同时还有一种高阻状态（或称为禁止状态），图 1-4 中，上面一个是读锁存器的缓冲器，也就是说，要读取 D 锁存器输出端 Q 的数据，需要使读锁存器中这个缓冲器的三态控制端（图 1-4 中标号为“读锁存器”端）有效，下面一个是读引脚的缓冲器，要读取 P0.X 引脚上的数据，也要使标号为“读引脚”的三态缓冲器的控制端有效，引脚上的数据才会传输到单片机的内部数据总线上。

（2）D 锁存器

构成一个锁存器，通常要用一个时序电路（时序的单元电路内容请参考数字电路相关知识），一个触发器可以保存一位二进制数（即具有保持功能），在 51 单片机的 32 根 I/O 口线中，都是用一个 D 触发器来构成锁存器的。图 1-4 中的 D 锁存器，D 端是数据输入端，CP 是控制端（即时序控制信号输入端），Q 是输出端，$\overline{Q}$ 是反向输出端。

对于 D 锁存器来讲，当 D 输入端有一个输入信号，如果这时控制端 CP 没有信号（即时序脉冲没有到来），这时输入端 D 的数据是无法传输到输出端 Q 及反向输出端 $\overline{Q}$ 的。如果时序控制端 CP 的时序脉冲到达，这时 D 端输入的数据就会传输到 Q 及 $\overline{Q}$ 端。数据传送过来后，当 CP 时序控制端的时序信号消失时，输出端还会保持着上次输入端 D 的数据（即把上次的数据锁存起来）。如果下一个时序控制脉冲信号到来，这时 D 端的数据才再次传送到 Q 端，从而改变 Q 端的状态。

（3）多路开关

在 51 单片机中，当内部的存储器够用时（即不需要外扩展存储器时，这里讲的存储器包括数据存储器及程序存储器），P0 口可以作为通用的输入/输出端口（即 I/O）使用，对于 8031（内部没有 ROM）的单片机，或者编写的程序超过了单片机内部的存储器容量需要外扩存储器时，P0 口就作为地址/数据总线使用。那么这个多路选择开关就是用于选择是作为普通 I/O 口使用还是作为地址/数据总线使用的选择开关了。从图 1-4 可知，当多路开关与下端接通时，P0 口作为普通的 I/O 口使用；当多路开关是与上端接通时，P0 口作为地址/数据总线使用。

（4）输出驱动

从图 1-4 中可看出，P0 口的输出是由两个 MOS 管组成的推拉式结构，也就是说，这两个 MOS 管一次只能导通一个，当 Vl 导通时，V2 截止，当 V2 导通时，Vl 截止。

上面已对 P0 口的各单元部件进行了详细的讲解，下面研究一下 P0 口作为 I/O 口及地址/数据总线使用时的具体工作过程。

（1）作为 I/O 端口使用时的工作原理

P0 口作为 I/O 端口使用时，多路开关的控制信号为 0（低电平），如图 1-4 所示，多路开关的控制信号同时和与门的一个输入端相接，与门的逻辑特点是“全 l 出 1，有 0 出 0”，那么控制信号如果是 0，这时与门输出的也是一个 0（低电平），此时 Vl 管就截止，在多路控制开关的控制信号是 0（低电平）时，多路开关是与锁存器的 $\overline{Q}$ 端相接的（即 P0 口作为 I/O 口线使用）。

P0 口用作 I/O 口线，其由数据总线向引脚输出（即输出状态 Output）的工作过程：写锁存器信号 CP 有效，数据总线的信号的输出流程为锁存器的输入端 D→锁存器的反向输出 $\overline{Q}$ 端→多路开关→V2 管的栅极→V2 管的漏极→输出端 P0.X。前面已经介绍过，当多路开关的控制信号为低电平 0 时，与门输出为低电平，Vl 管是截止的，所以作为输出口时，P0 是漏极开路输出状态，类似于 OC 门，当驱动上接电流负载时，需要外接上拉电阻。如图 1-5 所示就是由内部数据总线向 P0 口输出数据的流程图。

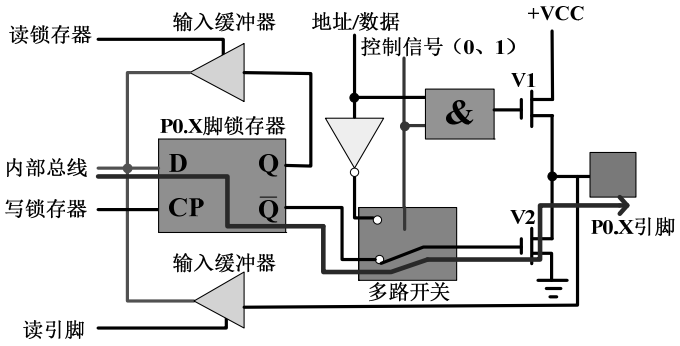

图 1-5　P0 口内部数据总线向引脚输出时的流程图

P0 口用作 I/O 口线，其由一引脚向内部数据总线输入（即输入状态 Input）的工作过程，数据输入时（读 P0 口）有以下两种情况：

第一种情况是读引脚，即读芯片引脚上的数据。读引脚数时，读引脚缓冲器打开（即

三态缓冲器的控制端要有效），通过内部数据总线输入。如图 1-6 所示为 P0 口读引脚时的流程图。

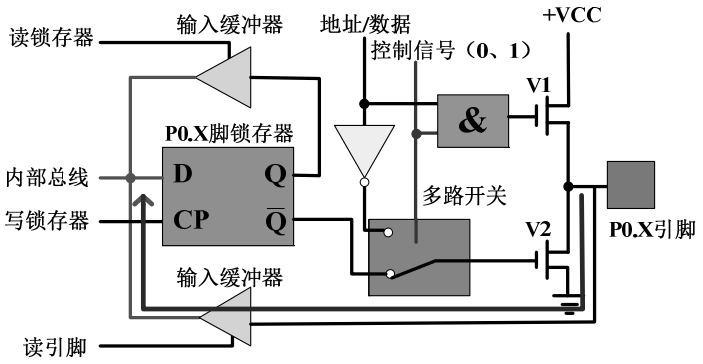

图 1-6　P0 口读引脚时的流程图

第二种情况是读锁存器，通过打开读锁存器三态缓冲器读取锁存器输出端 Q 的状态。如图 1-7 所示为 P0 口读锁存器时的流程图。

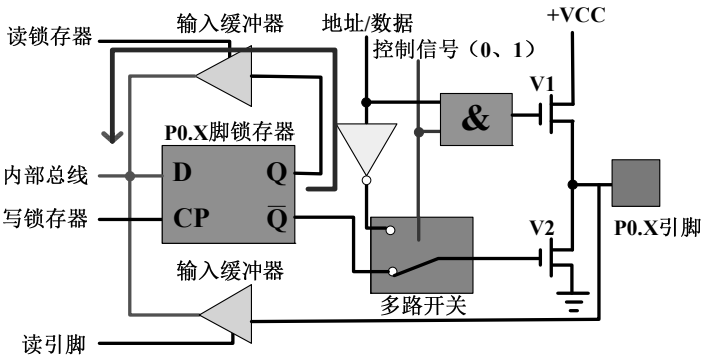

图 1-7　P0 口读锁存器时的流程图

在输入状态下，从锁存器和从引脚上读取的信号一般是一致的，但也有例外。例如，当从内部总线输出低电平后，锁存器 Q=0，$\overline{Q}$=l，场效应管 V2 开通，端口线呈低电平状态，此时无论端口线上外接的信号是低电平还是高电平，从引脚读入单片机的信号都是低电平，因而不能正确地读入端口引脚上的信号。又如，当从内部总线输出高电平后，锁存器 Q=1，$\overline{Q}$=0，场效应管 V2 截止，如果外接引脚信号为低电平，从引脚上读入的信号就与从锁存器读入的信号不同。为此，8031 单片机在对端口 P0～P3 的输入操作有如下约定：凡属于读—改—写方式的指令，从锁存器读入信号，其他指令则从端口引脚线上读入信号。读—改—写指令的特点是，从端口输入（读）信号，在单片机内加以运算（修改）后，再输出（写）到该端口上。下面是几条读—改—写指令的示例。

```
ORL   P0, A P0→AP0
INC   P1 P1+1→P1
DEC   P3 P3-1→P3
CPL   P2 P2→P2
```

这样安排的原因在于读—改—写指令需要得到端口原输出的状态，修改后再输出，读

锁存器而不是读引脚，可以避免因外部电路的原因使原端口的状态被读错。

注意：P0 端口是 8031 单片机的总线口，分时出现数据 D7～D0、低 8 位地址 A7～A0 以及三态，用来连接存储器、外部电路与外部设备。P0 端口是使用最广泛的 I/O 端口。

（2）作为地址/数据复用口使用时的工作原理

在访问外部存储器时，P0 口作为地址/数据复用口使用，这时多路开关控制信号为 l，与门解锁，与门输出信号电平由地址/数据线信号决定；多路开关与反相器的输出端相连，地址信号经地址/数据线→反相器→V2 场效应管栅极→V2 漏极输出。例如，控制信号为 l，地址信号为 0 时，与门输出低电平，Vl 管截止；反相器输出高电平，V2 管导通，输出引脚的地址信号为低电平。如图 1-8 所示为 P0 口作为地址线，控制信号为 1，地址信号为 0 时的工作流程图。

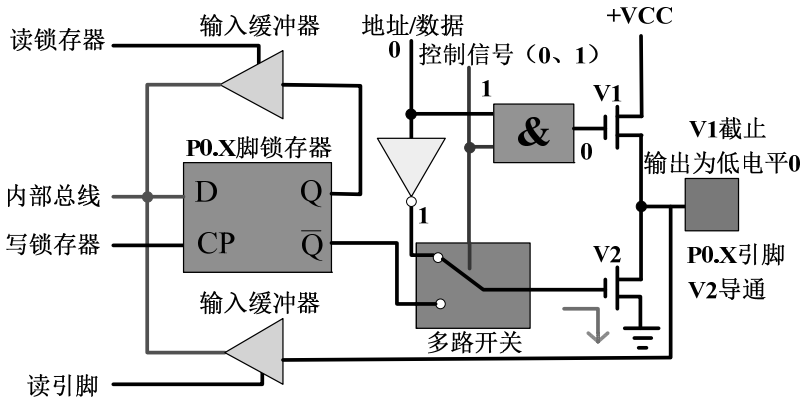

图 1-8　P0 口作为地址线，控制信号为 1，地址信号为 0 时的工作流程图

反之，控制信号为 l、地址信号为 l，与门输出为高电平，Vl 管导通；反相器输出低电平，V2 管截止，输出引脚的地址信号为高电平。如图 1-9 所示为 P0 口作为地址线，控制信号为 1，地址信号为 1 时的工作流程图。

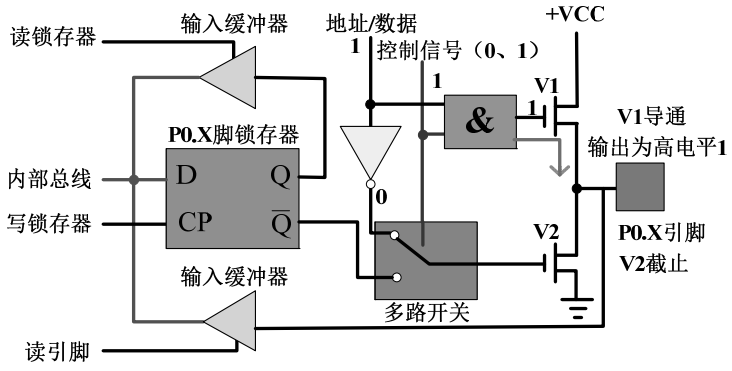

图 1-9　P0 口作为地址线，控制信号为 1，地址信号为 1 时的工作流程图

可见，在输出地址/数据信息时，Vl、V2 管是交替导通的，负载能力很强，可以直接与外设存储器相连，无须增加总线驱动器。P0 口又作为数据总线使用，在访问外部程序存储器时，P0 口输出低 8 位地址信息后，将变为数据总线，以便读指令码（输入）。在存取

指令期间，控制信号为 0，Vl 管截止，多路开关也跟着转向锁存器反相输出端 $\overline{Q}$；CPU 自动将 0FFH（11111111，即向 D 锁存器写入一个高电平 1）写入 P0 口锁存器，使 V2 管截止，在读引脚信号控制下，通过读引脚三态门电路将指令码读到内部总线。如图 1-10 所示为 P0 口作为数据总线，取指期间工作流程图。

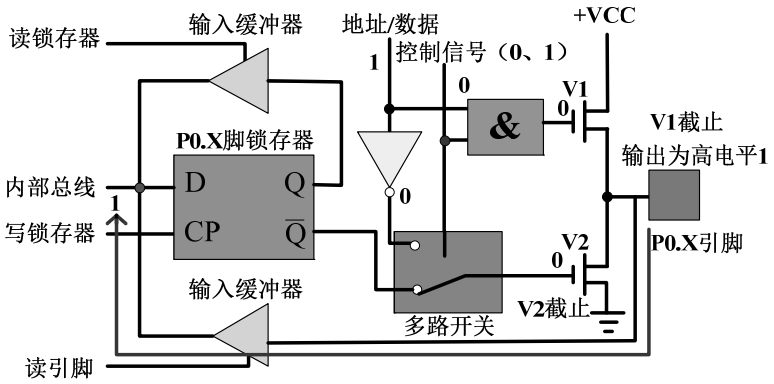

图 1-10　P0 口作为数据总线时取指期间工作流程图

如果该指令是输出数据，如“MOVX@DPTR,A”，该指令将累加器的内容通过 P0 口数据总线传送到外部 RAM 中，则多路开关控制信号为 1，与门解锁，与输出地址信号的工作流程类似，数据由地址/数据线→反相器→V2 场效应管栅极→V2 漏极输出。

如果该指令是输入数据（读外部数据存储器或程序存储器），如“MOVX A,@DPTR”，该指令将外部 RAM 某一存储单元内容通过 P0 口数据总线输入到累加器 A 中，则输入的数据仍通过读引脚三态缓冲器到内部总线，其过程类似于读取指令码流程图。

通过以上分析可以看出，当 P0 作为地址/数据总线使用时，在读指令码或输入数据前，CPU 自动向 P0 口锁存器写入 0FFH，破坏了 P0 口原来的状态。因此，不能再作为通用的 I/O 端口。

注意：系统设计中务必注意，程序中不能再含有以 P0 口作为操作数（包含源操作数和目的操作数）的指令。

当由 P0 口输入数据时，由于外部输入信号既加在缓冲输入端上，又加在驱动电路的漏极上。如果这时 T2 是导通的，则引脚上的电位始终被钳位在 0 电平上，输入数据不可能被正确地读入。因此，在输入数据时，应先把 P0 口置 1，使两个输出 FET 均关断，使引脚“浮置”，成为高阻状态，这样才能正确地插入数据，这就是准双向口。

I/O 口作为输入口时有两种工作方式，即读端口与读引脚，读端口时实际上并不从外部读入数据，而是把端口锁存器的内容读入到内部总线，经过某种运算或变换后再写回到端口锁存器，只有读端口时才真正地把外部的数据读入到内部总线，图 1-10 中的两个三角形表示的就是输入缓冲器，CPU 将根据不同的指令分别发出读端口或读引脚信号以完成不同的操作，这是由硬件自动完成的。读引脚时，就是把端口作为外部输入线时，首先要通过外部指令把端口锁存器置 1，然后再进行读引脚操作，否则就可能读入出错，为什么？看图 1-10 中，如果不对端口置 1，端口锁存器原来的状态有可能为 0，Q 端为 0，$\overline{Q}$ 端为 1，

加到场效应管栅极的信号为 1，该场效应管就导通，对地呈现低阻抗，此时即使引脚上输入的信号为 1，也会因端口的低阻抗而使信号变低，使得外加的 1 信号读入后不一定是 1，若先执行置 1 操作，则可以使场效应管截止，引脚信号直接加到三态缓冲器中，实现正确的读入，由于在输入操作时还必须附加一个准备动作，所以这类 I/O 口被称为准双向口，89C51 的 P0、P1、P2、P3 口作为输入时都是准双向口。接下来再看另一个问题，从图 1-10 中可以看出，这 4 个端口还有一个差别，除了 P1 口外，P0、P2、P3 口都还有其他功能，这些功能又作什么用的呢？下面就来详细讲解这个问题。

每个 I/O 端口都有一个 8 位数据锁存器和两个 8 位数据缓冲器。P0～P3（8 位锁存器）是 SFR，有各自的端口地址，可直接用指令寻址，用于存放需要输出的数据。数据输入时只有缓冲没有锁存，各引脚上输入的数据必须一直保持到 CPU 将其读走为止，如图 1-11 所示为 P0 位结构图。

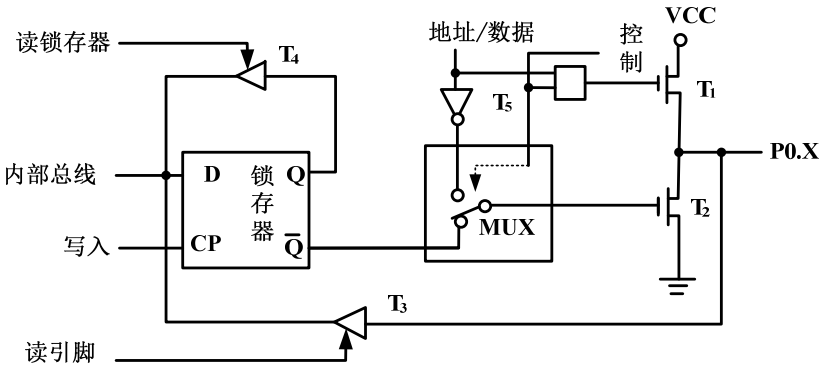

图 1-11　P0 位结构图

从图 1-11 中可以看出，P0 口的内部有一个二选一的选择器，受内部信号的控制，如果在图 1-11 中的位置，则处在 I/O 口工作方式，此时相当于一个准双向口输入，须先将 P0 口置 1，每根口线可以独立定义为输入或输出，但是必须在口线上加上拉电阻，如果将开关拨向另一个方向，则作为地址/数据复用总线用，此时不能逐位定义为输入/输出，有两种用法，当作数据总线用时输入 8 位数据，当作地址总线用时则输出低 8 位地址，注意，当 P0 口作为地址/数据复用总线用之后就不能再作 I/O 口使用了。那么什么叫做地址/数据复用？这其实是当单片机的并行口不够用时需要扩展输入/输出口时的一种用法，具体使用方法会在后续的章节中逐步讲解。

利用 P0 口进行扩展外部存储器和 I/O 时，P0 口将作为地址和数据分时复用，CPU 发控制信号，打开与门，使 MUX 打向上边，形成推拉式结构，数据信号可直接读入或输出到内部总线。利用 P0 作为通用 I/O 时，此时 P0 口是一个准双向口，CPU 发控制信号，封锁与门，使上拉管截止，MUX 打向下边，与 D 触发器 Q 连接。

输入程序举例：

```
MOV    P0, #FFH
```

输出程序举例：

```
MOV    A, P0
```

2．P1 口的结构及工作原理

P1 口字节地址为 90H，位地址为 90H～97H，如图 1-12 所示为 P1 位结构图。

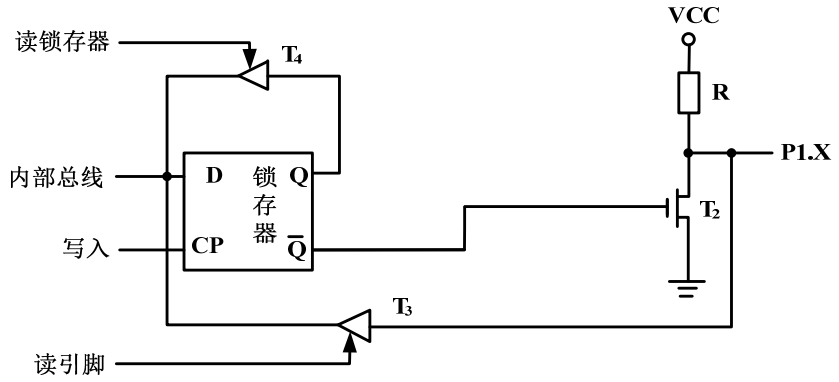

图 1-12　P1 位结构图

与 P0 不同，P1 口只能作为 I/O 口使用，无 MUX，但其内部有一个上拉电阻，所以连接外围负载时不需要外接上拉电阻，这一点 P1、P2、P3 都一样。

输入程序举例：

```
MOV   P1, #FFH
MOV   A, P1
```

输出程序举例：

```
MOV   A, P1
```

3．P2 口的结构及工作原理

P2 口字节地址为 A0H，位地址为 A0H～A7H，如图 1-13 所示为 P2 位结构图。

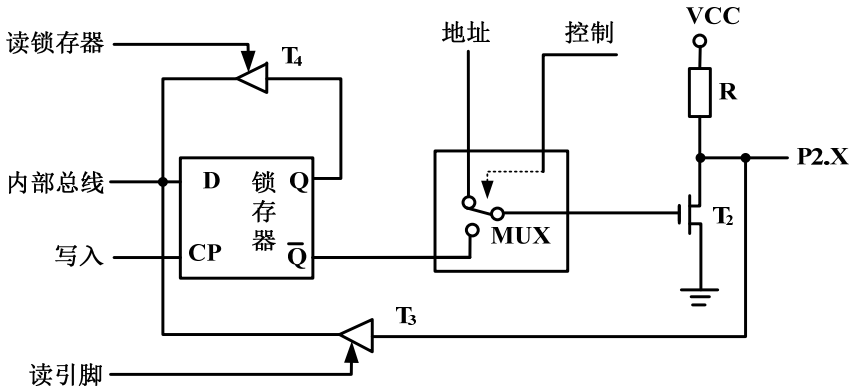

图 1-13　P2 位结构图

P2 口作为 I/O 口线时用法与 P0 口一样，当内部开关拨向另一个方向，即作地址输出时，可以输出程序存储器或外部数据存储器的高 8 位地址，并与 P0 口输出的低地址一起构成 16 位的地址线。

注意：和数据总线的区别，数据总线是 8 位的，很多书上都会提到 51 单片机是 8 位数据总线，16 位地址总线，但都不会解释有什么不同，看到这里读者应该明白二者的区别。

16 位的地址总线可以寻址 64KB 的程序存储器或外部数据存储器，后续章节会讲解，此处要注意的是当 P2 口作为地址总线时，高 8 位地址线是 8 位一起输出的，不能像 I/O 口线那样逐位定义，这与 P0 口是一样的。

当 P2 口用来扩展外存储器和 I/O 时，作为高 8 位地址输出，当进行外部存储器或 I/O 设备读写操作时，CPU 自动发出控制信号，打开与门，使 MUX 拨向上边。当 P2 口当作通用 I/O 时，CPU 自动发出控制信号，MUX 拨向下边，与 D 触发器 Q 连接。

输入程序举例：

```
MOV   P2, #FFH
MOV   A, P2
```

输出程序举例：

```
MOV   A, P2
```

4．P3 口的结构及工作原理

P3 口字节地址为 B0H，位地址为 B0H～B7H。如图 1-14 所示为 P3 位结构图。

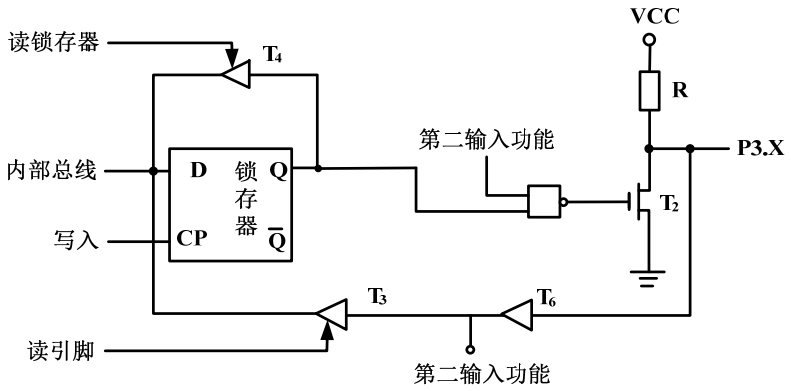

图 1-14　P3 位结构图

P3 口作为 I/O 口线用时同其他的端口相同，也是准双向口，不同的是，P3 口的每一位都有另一种功能，也叫第二功能，具体作用在用到时将详细解释。当 P3 口作为通用 I/O 口时，准双向口第二功能端保持高电平。

输入程序举例：

```
MOV   P3, #FFH
MOV   A, P3
```

输出程序举例：

```
MOV   A, P3
```

当 P3 口作为第二功能时，锁存器输出 Q=1，如表 1-4 所示为 P3 口第二功能列表。

表 1-4　P3 口第二功能列表

端　口　号	第 二 功 能	注　　释
P3.0	RXD	串行输入端
P3.1	TXD	串行输出端
P3.2	INT0	外部中断 0 输入端
P3.3	INT1	外部中断 1 输入端
P3.4	T0	定时/计数器 0 外部事件计数输入端
P3.5	T1	定时/计数器 1 外部事件计数输入端
P3.6	WR	外部数据存储器写选通信号，低电平有效
P3.7	RD	外部数据存储器读选通信号，低电平有效

既然单片机的引脚有第二功能，那么 CPU 是如何识别的呢？这是一个令许多初学者困惑的问题，其实单片机的第二功能是不需要人工干预的，也就是说只要 CPU 执行到相应的指令，就自动转成了第二功能。

思考：输入和输出口简称 I/O 口，是单片机与外部电路接口的唯一途径，4 个并行口的结构是有一定区别的，如何根据系统的设计要求和产品用途来正确灵活地使用是初学者必须掌握的基本功，还需要清楚其功能和用途。

5．应用注意事项

（1）在无片外扩展存储器的系统中，这 4 个端口的每一位都可以作为准双向通用 I/O 端口使用。在具有片外扩展存储器的系统中，P2 口作为高 8 位地址线，P0 口作为双向总线，分时作为低 8 位地址和数据的输入/输出线。

（2）P0 口作为通用双向 I/O 口使用时，必须外接上拉电阻。

（3）P3 口除了作通用 I/O 口使用外，各位还具有第二功能。当 P3 口某一位用于第二功能作输出时，则不能再作通用 I/O 口使用。

（4）当 P0～P4 端口用作输入时，为了避免误读，都必须先向对应的输出锁存器写入 1，使 FET 截止，然后再读端口引脚，例如以下程序：

```
MOV   P1, #0FFH
MOV   A, P1
```

1.4　单片机存储器知识介绍

1.4.1　概述

1．存储器的工作原理

存储器就是用来存放数据的地方，是利用电平的高或低来存放数据的，也就是说，存储器实际上存放的是电平的高或低的状态，而不是习惯上认为的 1、2、3、4 这样的数字。如图 1-15 所示为存储器的内部结构示意图。

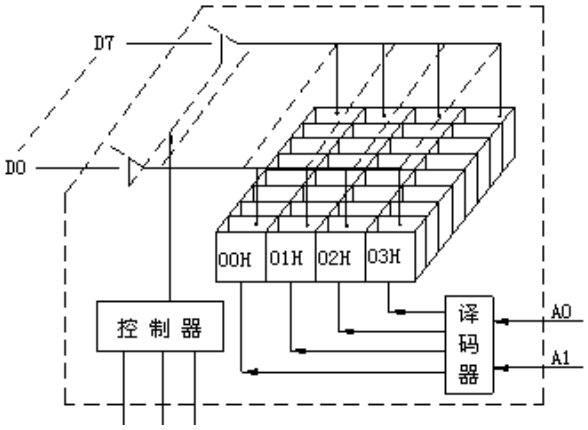

图 1-15　存储器的内部结构示意图

从图 1-15 可以看出，一个存储器就像一个小抽屉，一个小抽屉里有 8 个小空间，也就是单片机的 8 位小盒子，每个小盒子用来存放 1 位电荷，电荷通过与其相连的电线传进来或释放掉，至于电荷在小盒子里是怎样存放的，可以把电线想象成水管，小盒子里的电荷就像是水，这样就好理解了，存储器中的一个小抽屉称为一个单元，相当于 1 个字节，而一个小盒子就相当于 1 位，有了这样一个构造，就可以开始存放数据了。例如，要放进一个数据 00011010，只要把第 2、4、5 号小盒子里存满电荷，而其他小盒子里的电荷放掉即可。可是问题又出来了，一个存储器有好多相同的单元，线是并联的，看 D7～D0 在放入电荷时，会将电荷放入所有的字节单元中，而释放电荷时，会把每个单元中的电荷都放掉，这样，不管存储器有多少个字节单元，都只能放同一个数，这当然不是所希望出现的，因此，要在结构上稍作变化，图 1-15 中，在每个单元上有一根线与译码器相连，想要把数据放进哪个单元，就通过译码器给那个单元发信号，由译码器通过这根线把相应的开关打开，这样电荷就可以自由地进出了，那么这样是不是就能随意地向存储器写入或者读出数据了呢？其实还不能，继续看图 1-15 中与 D7～D0 相连的还有一个控制器，这根连线叫写入/读出控制线，当向存储器写入数据时，必须先把开关切换到写入端，而要读出数据时，需先把开关切换到读出端，片选端则是为了区分不同的存储器设置的。

2．存储器的片选及总线的概念

上面简单了解了存储器的工作原理，下面再来关注另外一个问题，送入每个字节的 8 根线从何而来？其实是从单片机的外部引脚上接过来的，一般这 8 根线除了接一个存储器之外还要接其他器件，这 8 根线既然不是存储器和单片机之间专用的，如果总是将某个单元接在这 8 根线上，就会出现问题，例如，存储器单元中的数值是 11111111，另一个存储器的单元数值是 00000000，那么这根线是处于高电平还是低电平？所以必须让它们分离，方法很简单，当外面的线接到集成电路的引脚上后，不直接接到各单元，而是在中间加一组开关，这组开关就是控制器，看图 1-15，平时让开关打开，如果确实要向这个存储器中写入数据，或要从存储器中读出数据，再让开关切换到相应的位置即可，这组开关由 3 根引线选择读控制端、写控制端和片选端，要将数据写入，先由控制器选中该片，然后发出相应的写信号，开关切换到相应的位置，并将传过来的数据、电荷写入片中，如果要读信

号，先选中该片，然后发出读信号，开关也切换到相应的位置上，数据就被送出去了，另外，读和写信号还同时受到译码器的控制，由于片选端不同，所以虽有读或写信号，但没有片选信号，另一个存储器就不会错误响应而造成冲突。那么会不会同时选中两个存储器呢？设计好的系统不会出现这种问题，如果出现同时选中两个存储器的情况，那就是电路出现故障了。从上面的介绍中可知，用来传递数据的 8 根线并不是专用的，而是由很多器件共用，所将其称为数据总线（BUS）。与之联系紧密的还有地址线。51 单片机共有 16 根地址线，后面章节将详细讲解。

既然单片机存储器内存放的是数据，为什么还要有地址的概念呢？这好比寄信，要寄一封信，就必须写好信的内容，然后在信封上填写详细地址，邮局才能按地址投递。给单片机传送数据也一样，除了要给出立即数（犹如信的内容），还必须知道这个数送达的地址（犹如信的地址或邮政编码），所以必须给每个寄存器（即半导体存储器）规定不同的地址，只不过在单片机中地址的编码也是用数字来表示的，那么单片机中有多少个寄存器，寄存器的地址又是如何规定的呢？下面看一下 51 单片机的存储结构及空间分配。

单片机的存储器结构一般有两种基本形式：一种是在通用微型计算机中广泛采用的将程序存储器和数据存储器合用一个存储空间的结构，称为普林斯顿（Princeton）结构或冯·诺依曼结构；另一种是将程序存储器 ROM 和数据存储器 RAM 截然分开，分别寻址的结构，称为哈佛（Har-Vard）结构。Intel 公司的 MCS-51 和 80C51 系列单片机采用的是哈佛结构。目前的单片机较多采用程序存储器 ROM 和数据存储器 RAM 截然分开的结构。在哈佛结构中，两种存储器单独编址，指令不一样。

内外 ROM 统一编址（MOVC），内外 RAM 分开编址（内 MOV，外 MOVX），外设和外 RAM 要统一编址（MOVX），占用一部分地址单元。在物理层面，有 4 个相互独立的存储空间：片内和片外程序存储器；片内和片外数据存储器。在逻辑上，有 3 个彼此独立的地址空间，即片内外统一编地址的 64KB 程序存储器地址空间、256B 的片内数据存储器地址空间和 64KB 片外数据存储器地址空间。如图 1-16 所示为 MCS-51 系列存储器地址空间分配图。

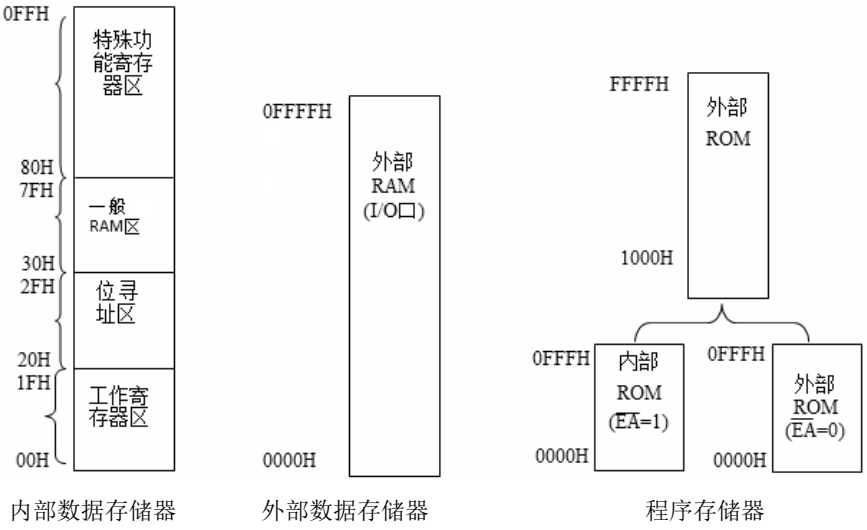

图 1-16　MCS-51 系列存储器地址空间分配图

1.4.2 程序存储器

89C51 单片机最多可寻址 64KB 的程序存储器。89C51 的内部有 4KB 的 Flash ROM 空间，其寻址范围为 0000H～0FFFH，换算为十进制数，即为 0～4095（单位：字节）。

这 4KB 的 ROM 空间用于存放为单片机编写的程序。单片机执行指令时就是一条一条顺序地从 ROM 中寻找指令并执行。如图 1-17 所示为单片机 ROM 空间。

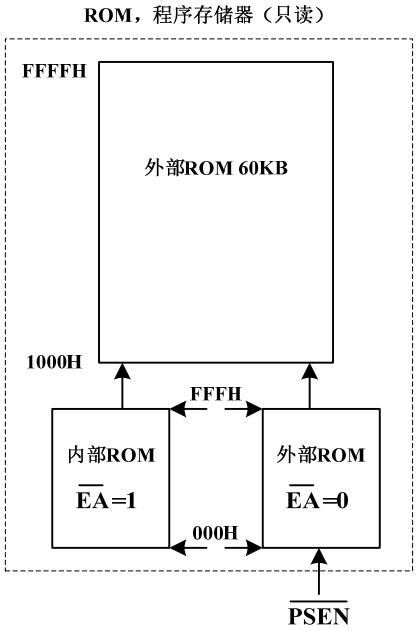

图 1-17　单片机 ROM 空间

$\overline{EA}=0$ 时，全部自动执行片外程序存储器；$\overline{EA}=1$ 时，先执行片内程序存储器，执行完 4KB 后，自动执行片外程序存储器。

1.4.3 数据存储器

89C51 单片机数据存储地址空间由内部和外部数据存储器空间组成，需要注意的是，内部和外部数据存储器空间存在地址重叠，如图 1-18 所示。如图 1-19 所示为 AT89S51 单片机特殊功能寄存器分布图。

1．内部数据存储器

89C51 内部共有 128B 的 RAM 空间，加上内部特殊功能寄存器空间，其寻址范围为 00H～FFH，可以被分成 4 个区域：

（1）区域 00H～1FH，安排了 4 组工作寄存器，每组用 8B，共 32B，分别为 R0～R7。在同一时刻，只能用其中的一组工作寄存器，实现控制要用到程序状态字 PWS 中的 RS0、RS1 两位。

（2）区域 20H～2FH，共 16B，除了可以作为一般的 RAM 单元读写外，还可以对每个字节的每一位，即每一个抽屉中的每一个小盒子进行操作，并且对这些位都规定了固定的位地址，从 20H 单元的第 0 位开始到 2FH 单元的第 7 位结束，共 128 位。

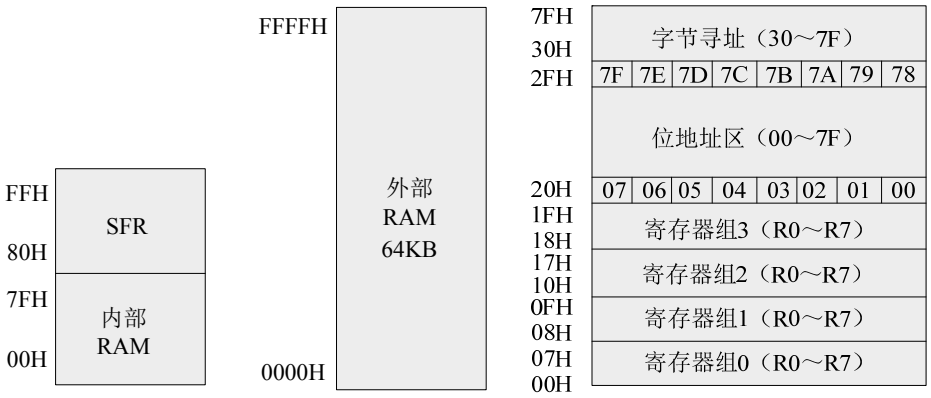

图 1-18　数据存储器示意图

0F8H								0FFH
0F0H	B							0F7H
0E8H								0EFH
0E0H	ACC							0E7H
0D8H								0DFH
0D0H	PSW							0D7H
0C8H								0CFH
0C0H								0C7H
0B8H	IP							0BFH
0B0H	P3							0B7H
0A8H	IE							0AFH
0A0H	P2		AUXR1				WDTRST	0A7H
98H	SCON	SBUF						9FH
90H	P1							97H
88H	TCON	TMOD	TL0	TL1	TH0	TH1	AUXR	8FH
80H	P0	SP	DP0L	DP0H	DP1L	DP1H		PCON 87H

图 1-19　AT89S51 单片机特殊功能寄存器分布图

（3）区域 30H～7FH，是一般的 RAM 单元地址，共 80B。内部数据存储器可用以下关键字说明。

- data：直接寻址区，为内部 RAM 的低 128B，即 00H～7FH。
- idata：间接寻址区，包括整个内部 RAM 区 00H～FFH。
- bdata：可位寻址区，20H～2FH。

（4）区域 80H～FFH，是特殊功能寄存器地址。89C51 提供 128B 的 SFR 寻址区，该区域中可位寻址、字节寻址或字寻址，用以控制定时器、计数器、串口、I/O 及其他部件，可由以下几种关键字说明。

- sfr：字节寻址，如 sfr P0=0x80；指定 P0 口地址为 80H。
- sfr16：字寻址，如 sfr T2=0xCC；指定 Timer2 口地址，T2L=0xCC，T2H=0xCD。
- sbit：位寻址，如 sbit EA=0xAF；指定第 0xAF 位为 EA，即中断允许。

还可以有如下定义方法：

SbitV0=PSW^2；（定义 V0 为 PSW 的第 2 位）

2．外部数据存储器

外部 RAM 视使用情况可由以下关键字标识。

- xdata：可指定多达 64KB 的外部直接寻址区，地址范围为 0000H～0FFFFH。
- pdata：能访问 1 页（256B）的外部 RAM，主要用于紧凑模式（Compact Model）。

3．关于数据存储器

（1）根据地址总线宽度，在片外可扩展的存储器最大容量为 64KB，地址范围为 0000H～FFFFH。

（2）片外数据存储器与程序存储器的操作使用不同的指令和控制信号，允许两者的地址重复，因此，片外要扩展的数据存储器与程序存储器各为 64KB。

（3）片外数据存储器与片内数据存储器的操作指令亦不同（对片外 RAM 用 MOV 指令），所以也允许二者的地址重复，内部数据存储器的地址为 00H～FFH，外部扩展数据存储器的地址可以为 0000H～FFFFH。

注意：采用 R0、Rl 或 DPTR 寄存器间址方式访问片外数据存储器。当采用 R0、Rl 间址时只能访问低 256B，采用 DPTR 间址可访问整个 64KB 空间。

1.4.4 单片机存储模式

存储模式决定了没有明确指定存储类型的变量、函数参数等的默认存储区域，共 3 种。

1．Small 模式

所有默认变量参数均装入内部 RAM，优点是访问速度快，缺点是空间有限，只适用于小程序。

2．Compact 模式

所有默认变量均位于外部 RAM 区的一页（256B），具体页可由 P2 口指定，在 STARTUP.A51 文件中说明，也可用 pdata 指定，优点是空间较 Small 模式宽裕，速度较 Small 模式慢，较 Large 模式快，是一种中间状态。

3．Large 模式

所有默认变量可放在多达 64KB 的外部 RAM 区，优点是空间大，可存变量多，缺点是速度较慢。

1.5 单片机 CPU 的时序

1.5.1 单片机的时序

1．时序的由来

单片机执行指令的过程就是顺序地从 ROM 程序存储器中取出指令，一条一条地顺序执行，然后进行一系列微操作控制，来完成各种指定的动作，在协调内部的各种动作时必须要有一定的顺序，换句话说，就是这一系列微操作控制信号在时间上要有一个严格的先

后次序，这种次序就是单片机的时序，好比学校上课时用的铃声，为了保证课堂秩序，学校必须在铃声的统一协调下安排各个课程和活动，那么单片机的时序是如何规定的呢？

2．时序的周期

计算机每访问一次存储器的时间，称为一个机器周期，它是一个时间基准，就像日常生活中使用的秒一样，计算机中一个机器周期包括 12 个振荡周期。而振荡周期就是振荡源的周期，也就是使用的晶振的时间周期，一个 12M 晶振的时间周期是 T=1/f，也就是 1/12μs，那么 12M 晶振的单片机的一个机器周期就应该等于 12×1/12μs，也就是 1μs。

在 89C51 单片机中，有些指令只要一个机器周期，而有些指令则需要 2～3 个机器周期，另外还有两条指令需要 4 个机器周期，衡量指令执行时间的长短要用到一个新的概念——指令周期，即执行一条指令所需的机器周期。Intel 公司规定了每一条指令执行的机器周期，当然这不需要记住，不过在这里 DJNZ 指令是要记住的，它是双周期指令，执行一次需要两个机器周期，即 2μs，则 12M 晶振的延时时间是 62500×2μs=125000μs，也就是 125ms，即 0.125s，所以 LED 灯闪烁很快。

1.5.2　单片机的时钟电路

大家已经知道，单片机是在一定的时序控制下工作的，那么时序和时钟又有什么关系呢？时钟是时序的基础，单片机本身就如同一个复杂的同步时序电路，为了保证同步工作方式的实现，电路要在唯一的时钟信号控制下按时序工作，那么单片机内的时钟是如何产生的呢？

1．内部时钟电路

在 MCS-51 单片机的内部，有一个高增益的反相放大器，其输入端为引脚 XTAL1（19），输出端为引脚 XTAL2（18），只要在外部接上两个电容和一个晶振，就能构成一个稳定的自激振荡器，如图 1-20 所示。这里主要讲解电容和晶振的选择。晶振的大小与单片机的振荡频率有关，待介绍串行接口时再详细讲解，电容的大小影响振荡器振荡的稳定性和起振的快速性，通常选择 10～30pF 的瓷片电容或校正电容。另外在设计电路时，晶振和电容应尽可能地靠近芯片以减少 PCB 板的分布电容，保证振荡器工作的稳定性，提高系统的抗干扰能力。

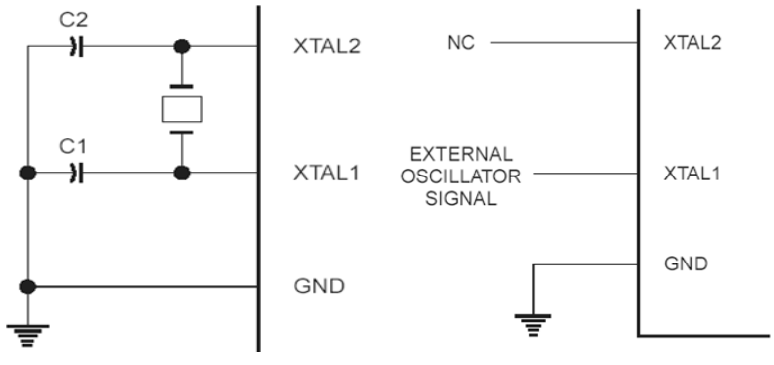

图 1-20　单片机的时钟振荡电路

单片机的时钟振荡电路有两种方式：

（1）由单片机外部的独立振荡电路产生的时钟信号，通过单片机 XTAL1 引脚输入，

再经 12 分频后为单片机提供工作时钟信号。

（2）通过外接晶振和两个 30pF 的起振电容，与单片机内部的反相器构成振荡电路，产生振荡信号，经 12 分频后为单片机提供工作时钟信号。在这里采用了第二种方式获得时钟振荡信号。

2．外部时钟电路

除了内部时钟方式外，单片机还可以采用引入外部时钟的振荡方式。当系统由多片单片机组成时，为了保证各单片机之间时钟信号的同步，就应当引入唯一公用的外部脉冲信号作为各单片机的振荡脉冲，此时应将 XTAL2 悬空不用，外部脉冲信号由 XTAL1 引入，如图 1-20 的右图所示。

- 振荡周期：为单片机提供定时信号的振荡源的周期，若为内部产生方式，为石英晶体的振荡周期。
- 时钟周期：也称为状态周期，用 S 表示。时钟周期是计算机中最基本的时间单位，在一个时钟周期内，CPU 完成一个最基本的动作。MCS-51 单片机中一个时钟周期为振荡周期的 2 倍。
- 机器周期：完成一个基本操作（如取指令、存储器读、存储器写等）所需要的时间称为机器周期。MCS-51 的一个机器周期含有 6 个时钟周期。
- 指令周期：完成一条指令所需要的时间称为指令周期。MCS-5l 的指令周期含 1～4 个机器周期不等，其中多数为单周期指令，还有 2 周期和 4 周期指令。4 周期指令只有乘、除两条指令。

如图 1-21 所示为 MCS-51 单片机各种周期的相互关系。

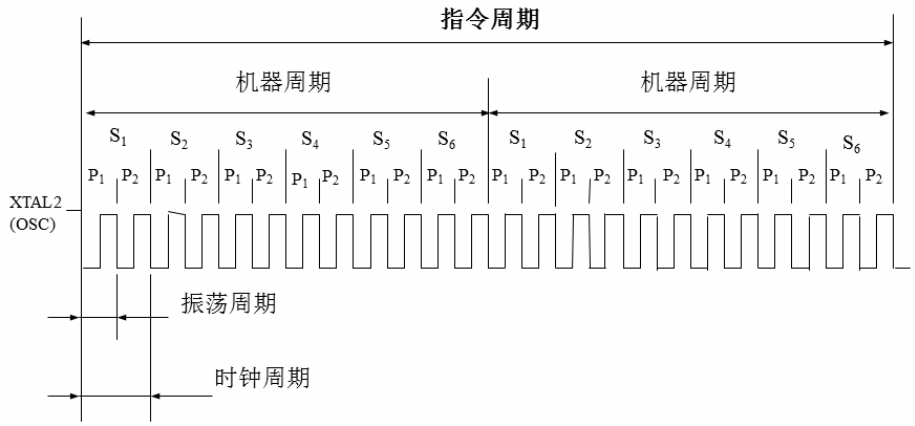

图 1-21　MCS-51 单片机各种周期的相互关系

1.6　单片机的外部接口及其扩展

1.6.1　中断系统

在 CPU 与外设交换信息时，存在一个快速的 CPU 与慢速的外设间的矛盾。为解决这

个问题，发展了中断的概念。当 CPU 正在处理某项事务时，如果外界或内部发生了紧急事件，要求 CPU 暂停正在处理的工作转而去处理这个紧急事件，待处理完以后再回到原来被中断的地方，继续执行原来被中断的程序，这样的过程称为中断。向 CPU 提出中断请求的源称为中断源。

当 CPU 正在处理一个优先级低的中断请求时，如果发生另一个优先级比它高的中断请求，CPU 将暂停正在处理的中断源的处理程序，转去处理优先级高的中断请求，待处理完以后，再回到原来正在处理的低级中断程序，这种高级中断源中断低级中断源的中断处理称为中断嵌套。51 单片机总共有 6 个中断源，包括两个外部中断、3 个定时器中断和一个串行接口中断。6 个中断源可分为两级中断优先级，每个中断源都有一个对应中断请求标志位，设置在特殊功能寄存器 TCON 和 SCON 中。在 51 单片机中断系统中，中断允许或禁止是由片内的中断允许寄存器 IE 控制的，中断寄存器各位的含义如表 1-5 所示。

表 1-5　中断寄存器位功能

符　　号	位　　置	功　　能
EA	IE.7	全局中断使能位。如果 EA=0，中断全部关闭。如果 EA=1，相应位定
-	IE.6	保留
ET2	IE.5	定时器 2 中断使能位
ES	IE.4	串口中断使能位
ET1	IE.3	定时 1 中断使能位
EX1	IE.2	外部中断 1 使能位
ET0	IE.1	定时器 0 中断使能位
EX0	IE.0	外部中断 0 使能位

AT89S51 对每一个中断请求源都可以编程为高优先级中断源或低优先级中断源。中断优先级是由片内的中断优先级寄存器 IP 控制的，IP 寄存器中各位的功能如表 1-6 所示。中断优先级控制寄存器 IP 中的各个控制位都可由编程来置位或复位（用位操作指令或字节操作指令），单片机复位后 IP 中各位均为 0，各个中断源均为低优先级中断源。

表 1-6　中断优先级控制寄存器

符　　号	位　　置	功　　能
-	IP.7	保留
-	IP.6	保留
PT2	IP.5	定时器 2 中断优先级控制位
PS	IP.4	串口中断优先级控制位
PT1	IP.3	定时器 1 优先级控制位
PX1	IP.2	外部中断 1 优先级控制位
PT0	IP.1	定时器 0 优先级控制位
PX0	IP.0	外部中断 0 优先级控制位

1.6.2　定时器/计数器

AT89S51 单片机内部有 3 个 16 位可编程定时器，简称为定时器 T0、T1 和 T2。由于 T0 和 T1 定时器的结构和早期的 51 系列单片机系统相同，这里不再叙述，主要讲解关于

T2 定时器的结构、操作模式和控制寄存器。T2CON 的作用是控制定时器 2 的工作模式、启、停，以及标志定时器 2 的溢出和中断情况。定时器 2 的格式如表 1-7 所示。

表 1-7　定时器 2 控制寄存器 T2CON

位 序 号	位 符 号	位 功 能
7	TF2	定时器 2 溢出标志。当定时器 2 溢出时，由硬件使 TF2 置 1，并且申请中断，必须由软件清零。当 RCLK 或者 TCLK 等于 1 时，TF2 不会被置位
6	EXF2	定时器 2 外部标志位。当 T2EX 引脚上有负跳变，同时 EXEN2 等于 1，处于捕捉模式或再装载模式时，EXF2 被置位
5	RCLK	接收时钟使能位
4	TCLK	发送时钟使能位
3	EXEN2	定时器 2 外部使能位
2	TR2	定时器 2 启动、停止位
1	C/$\overline{T2}$	定时器 2 定时计数选择位
0	CP/$\overline{RL2}$	捕捉模式、再装载模式选择位

定时器 2 的工作模式通过 T2CON 设置，有 3 种工作模式：捕捉模式、自动再加载模式（加或减计数器）和用作波特率发生器。定时器 2 的工作模式如表 1-8 所示。

表 1-8　定时器 2 的工作模式

RCLK+TCLK	CP / $\overline{RL2}$	TR2	MODE
0 （RCLK 和 TCLK 全为 0）	0	1	16 位自动装载模式
0 （RCLK 和 TCLK 全为 0）	1	1	16 位捕捉模式
1 （仅 RCLK 和 TCLK 其中一个为 1）	X	1	波特率发生器
X	X	0	关

1.6.3　串口

AT89S51 单片机内部的串行口是全双工的，即能同时发送和接收数据。发送缓冲器只能写入而不能读出；接收缓冲器只能读出而不能写入。串行接口还有接收缓冲的作用，即从接收寄存器中读出前一个已收到的字节之前就能开始接收第二字节。

两个串行接口数据缓冲器（实际上是两个寄存器）通过特殊功能寄存器 SBUF 访问。写入 SBUF 的数据存储在发送缓冲器中，用于串行发送；从 SBUF 读出的数据来自接收缓冲器。两个缓冲器共用一个地址 99H（特殊功能寄存器 SBUF 的地址），与通用的 51 单片机串行接口相同，不再详细叙述。

1.6.4　特有寄存器

辅助寄存器 AUXR 的作用是设定单片机工作时外部端口的一些时序，各位的功能如表 1-9 所示，在复位以后的值为×××00××0B。

表 1-9　辅助寄存器 AUXR

位 序 号	位 符 号	位 功 能
7	—	—
6	—	—
5	—	—
4	WDIDLE	WDT 在空闲模式下的禁止/允许位 当 WDIDLE＝0 时，WDT 在空闲模式下继续计数 当 WDIDLE＝1 时，WDT 在空闲模式下暂停计数
3	DISRTO	禁止/允许 WDT 溢出时的复位输出 当 DISRTO＝0 时，WDT 定时器溢出时，在 RST 引脚输出一个高电平脉冲 当 DISRTO＝1 时，RST 引脚为输入脚
2	—	—
1	—	—
0	DISALE	ALE 禁止/允许位 当 DISALE＝0 时，ALE 有效，发出恒定频率脉冲 当 DISALE＝1 时，ALE 仅在 CPU 执行 MOVC 和 MOVX 类指令时有效，不访问外寄存器时，ALE 不输出脉冲信号

辅助寄存器 AUXR1 的作用也是设定单片机工作时外部端口的一些时序。寄存器 AUXR1 各位的功能如表 1-10 所示，在复位以后的值为×××××××0B。

表 1-10　辅助寄存器 AUXR1

位 序 号	位 符 号	位 功 能
7	—	—
6	—	—
5	—	—
4	—	—
3	—	—
2	—	—
1	—	—
0	DPS	数据指针寄存器选择位 当 DPS＝0 时，选择数据指针寄存器 DPRT0 当 DPS＝1 时，选择数据指针寄存器 DPRT

AT89S51 有一个看门狗，以前的 80C52 和 AT89C52 是没有的。看门狗可以看作是当 CPU 在执行指令出现混乱时恢复系统的一个方法。AT89S51 的看门狗由一个 14 位计数器和看门狗定时器复位寄存器（WDTRST）组成。为了启动看门狗，用户必须按顺序向看门狗定时器复位寄存器写入 1EH 和 E1H。当看门狗启动以后，用户必须隔一定时间向看门狗定时器复位寄存器写入 1EH 和 E1H 以防止看门狗溢出。在单片机处于电源关闭模式时，不需要向 WDTRST 写入数值。AT89S51 有 3 个锁定位，可以编程也可以不编程，通过 3

个锁定位的设定，AT89S51 可以获得不同的功能，具体的功能如表 1-11 所示。

表 1-11　编程锁定位说明

模　式	LB1	LB2	LB3	功　能
1	U	U	U	没有编程锁定特征
2	P	U	U	来自外部程序存储器的 MOVC 指令被禁止，以防止从内部程序存储器中读取代码。在单片机复位时，引脚 $\overline{EA}$ 被采样和锁定。对内部程序存储器 FLASH 的编程被禁止
3	P	P	U	同模式 2，但校验被禁止
4	P	P	P	同模式 3，但从外部程序存储器执行程序被禁止