# 第 8 章　单片机系统扩展接口技术

I/O 接口是单片机与外部设备、外部存储器，以及其他单片机/计算机连接的桥梁，是单片机应用系统中非常重要的部分。本章主要介绍接口的基本概念，并行 I/O 口的扩展（包括用简单门电路扩展和使用专用可编程芯片扩展），数据存储器的扩展（包括通过并行口扩展 SRAM 和通过 $I^2C$ 方式扩展 EEPROM）。本章是单片机应用系统接口的基础。

## 8.1　接口的基本概念

### 8.1.1　单片机应用系统构成

一般的单片机应用系统如图 8-1 所示，中间是单片机（包括外部的存储器），最左边是被检测的设备，最右边是被控制的设备，单片机与被检测设备和被控制设备之间是接口设备，就是人们常说的输入、输出设备，接口设备常常简称为接口。我们把接口设备分为三类：人机交互设备（键盘、显示器、打印机等），模拟量设备（传感器、A/D 转换器、D/A 转换器），数字量、开关量设备（并行口、各种串行口、计数器、各种继电器等）。

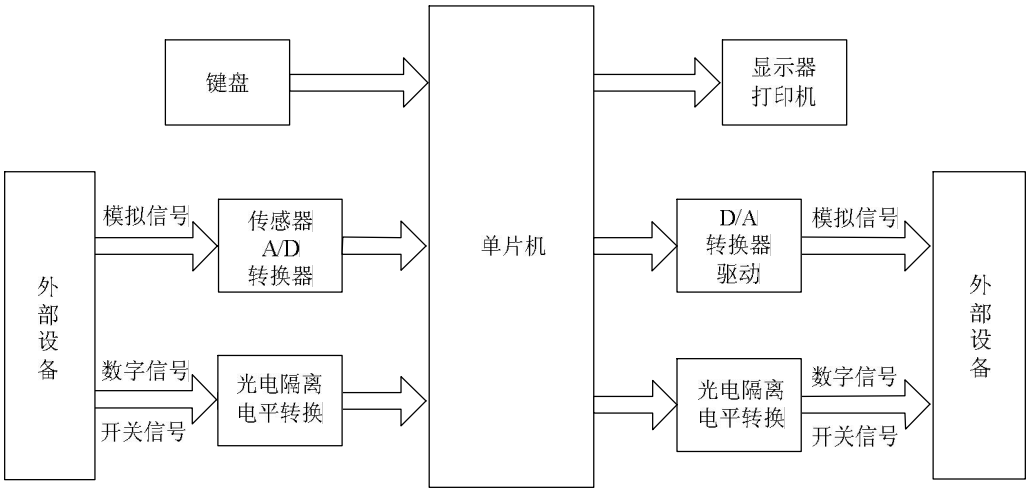


图 8-1　单片机一般应用系统构成

如果单片机应用系统没有右边的被控制设备，则系统成为一测量设备或测量仪器。常见的测量对象有压力、温度、湿度、流量、电压、电流、速度、加速度、脉冲计数、各种设备的状态等。在实际中，有不少应用只需要测量几种信号，如压力、温度等模拟量信号，设备状态、脉冲计数等数字量或开关量信号。

如果单片机应用系统没有左边的被测量设备，则系统成为一控制设备。常见的控制对象有温度、压力、流量、电压、电流、速度、声音、设备开启/关闭、电机运行（通过变频器、脉宽调制、频率等方法）等。在实际中，有不少应用只需要控制几种对象，如压力、温度等模

拟量对象，设备开启/关闭、电机运行、声音等数字量或开关量对象。

如果单片机应用系统右边的被控制设备和左边的被测量设备是同一个设备，则系统成为一闭环控制系统。闭环系统不仅硬件部分复杂，而且控制程序也要复杂得多。但只要掌握了基本的测量/控制设备及应用编程方法，然后再进一步提高编程能力，逐步就能做闭环系统。

从图 8-1 的单片机应用系统和上面的描述可以看出，接口在应用系统中所占的比重很大，并且非常重要。从本章开始，用三章内容讲单片机的接口技术，内容分别是系统扩展接口技术（包含数字量输入、输出）、人机交互接口技术、模拟量和开关量接口技术，通过这些内容的学习，为单片机应用开发打下基础。

### 8.1.2　接口的概念

各种外部设备的信号在电平、信息格式、工作速度、时序上都和 CPU（单片机）有很大的差别，因此不能直接和 CPU（单片机）相连。例如：模拟设备需要把信息转换成数字信号，才能输入单片机；同样，单片机需要把数字格式的控制信号，转换成模拟信号才能控制模拟设备；外部设备的电平也要转换成单片机的工作电平；低速设备需要缓冲，才能和 CPU（单片机）交换数据等。在这种情况下，需要一些中间电路，完成电平转换、格式转换、数据缓冲等功能才能和 CPU（单片机）连接。这些使 CPU 和设备之间能够完成数据交换的中间电路就叫做 I/O 接口，简称接口，也叫做接口适配器。不仅是硬件电路，还包括软件程序。

### 8.1.3　接口的基本功能

接口用于实现 CPU 和外部设备之间的连接，单片机接口的主要功能如下：

1）信号与信息格式转换功能。某些外部设备所提供的信号及信息格式与单片机的内部总线不兼容，因此部分接口需要具备相应的转换功能，如 A/D、D/A 转换、串/并、并/串转换、电平转换等。

2）数据缓冲功能。CPU 与外部设备在工作速度上往往不匹配，因此接口需要有数据缓冲功能，避免因速度不一致而造成数据丢失。

3）接受命令功能。接口要能够接受来自于 CPU 的命令以完成特定的操作。

4）提供状态信息功能。多数接口能够收集外部设备以及接口自身的状态信息并存储，以供 CPU 查询。

5）中断功能。能够以中断方式工作的接口中具有中断电路，可以产生中断请求信号。

### 8.1.4　接口的结构

接口的结构和接口的功能是对应的，一般包括读写控制逻辑、数据缓冲器、数据寄存器、控制寄存器和状态寄存器等部分。

读写控制逻辑连接来自于 CPU 的读写控制信号、地址信号和片选信号。读写控制逻辑控制对接口内部寄存器的选择和读写操作；对控制寄存器存放的控制代码进行译码并执行；控制接口内部各组成部分工作；负责和外部设备进行联络。

数据缓冲器连接系统的数据总线。在读写控制逻辑的控制下接收数据总线上的数据，并转发到对应的寄存器；或把寄存器里的内容发往数据总线。

数据寄存器存放输入/输出的数据。可以分为输入数据寄存器和输出数据寄存器。

控制寄存器用于存放 CPU 发来的控制代码。

状态寄存器用于存放接口和外部设备的状态信息，以供 CPU 查询。

这些寄存器都有地址，CPU 可以像访问存储器一样去访问这些寄存器。对这些寄存器读写就实现了 CPU 对接口的控制。

这些寄存器根据需要可以有不同的配置，可以只有一个，也可以有多个。接口的结构如图 8-2 所示。

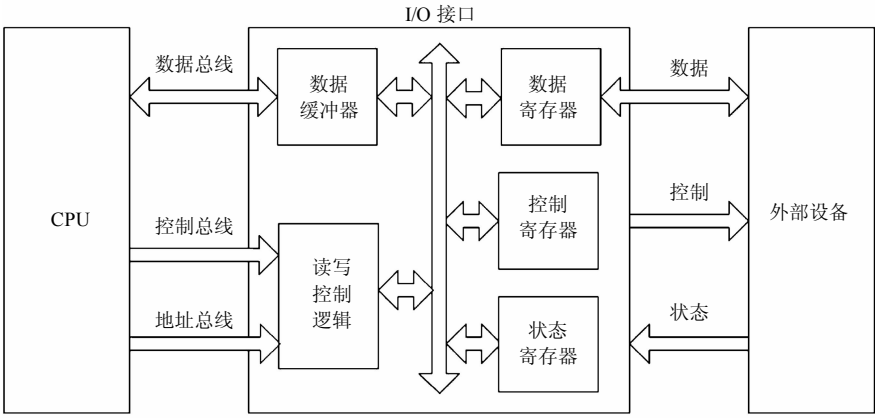


图 8-2　接口的结构

### 8.1.5　端口及编址

1. 端口

除了单片机内部集成的 I/O 接口，接口通常是以芯片的形式出现。有些简单的外部设备会和接口集成在一起。单片机外围可以连接多个接口芯片，每个接口芯片内部通常又会有多个寄存器。如何区分这些芯片和芯片内部的寄存器？方法就是给它们分配地址。每个芯片内部可被访问的寄存器，我们称之为端口。每一个端口都要分配一个地址，以实现对芯片内部不同寄存器的选择。这个地址就叫做端口地址，通常也叫接口地址。

在并行总线扩展系统中，端口地址是通过硬布线完成的，一旦完成了布线，端口地址也就确定了下来。端口地址是由两部分组成的，端口地址的高位是片选，即实现对芯片的选择；端口地址的低位完成对芯片内部寄存器的选择。

2. 片选

芯片的选择有两种方法：线选法和译码法。

1）线选法。所谓线选法，就是直接以系统的地址线作为芯片的片选信号，为此只需把用到的地址线与芯片的片选端直接相连即可。

2）译码法。所谓译码法，就是使用地址译码器对系统的片外地址进行译码，以其译码输出作为存储器芯片的片选信号。

译码法可以有效利用地址总线，生成更多片选信号。但结构复杂，需要连接地址译码器。74LS138 是一种常用的地址译码器芯片，其引脚如图 8-3 所示。

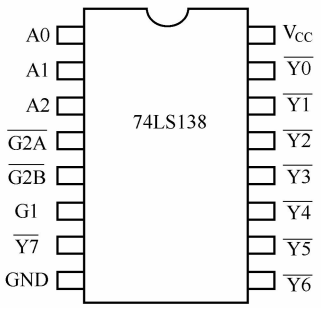


图 8-3　译码器芯片 74LS138

其中，G1、$\overline{G2A}$、$\overline{G2B}$为控制端。只有当 G1 为“1”，且$\overline{G2A}$、$\overline{G2B}$均为“0”时，译码器才能进行译码输出。否则译码器的 8 个输出端全为高阻状态。译码输入端与输出端之间的译码关系如表 8-1 所示。

表 8-1　74LS138 的译码关系

| A2～A0 编码 | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
|---|---|---|---|---|---|---|---|---|
| 输出有效位 | $\overline{Y0}$ | $\overline{Y1}$ | $\overline{Y2}$ | $\overline{Y3}$ | $\overline{Y4}$ | $\overline{Y5}$ | $\overline{Y6}$ | $\overline{Y7}$ |

3. 芯片内部寄存器的选择

地址总线的低位连接到芯片，由芯片内部的地址译码电路和读写控制逻辑共同完成对芯片内部寄存器的选择。

**需要注意的是：**在并行总线扩展系统中，端口地址由系统的地址总线生成。如果没有连接所有的地址线，那么在访问端口时，未用到的地址线置 0 置 1 都可以，即多个地址指向同一端口。

## 8.2　用并行方式扩展数据存储器

### 8.2.1　MCS-51 单片机三总线结构

单片机内部集成了很多资源，但是在实际应用系统中往往不够用，这时候就要进行系统扩展，以连接更多的设备和资源，以满足需求。系统扩展一般是通过扩展系统并行三总线来实现的，如图 8-4 所示。

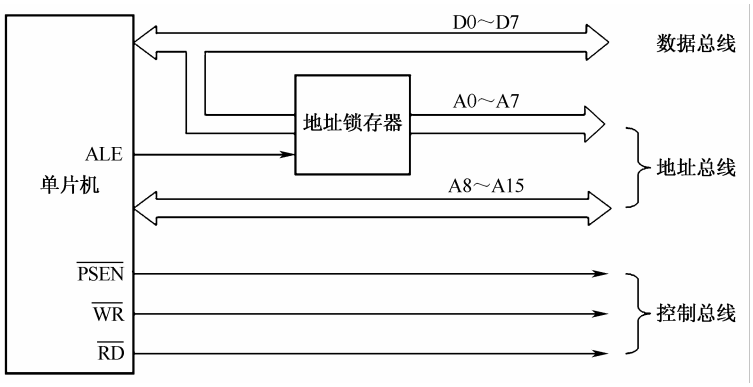


图 8-4　单片机扩展并行三总线

51 单片机由于引脚数量的限制，数据总线和地址总线是复用的，而且由 I/O 端口线兼用。为了能把复用的数据总线和地址总线分离出来以便同外部的芯片正确地连接，需要在单片机的外部增加地址锁存器，将分时输出的低 8 位地址信号锁存。常用地址锁存器 74HC573，其信号及其与单片机 P0 口的连接如图 8-5 所示。

74HC573 是有输出三态门的 8 位锁存器。当 G（使能端）为高电平时，锁存器的数据输出端 Q 的状态与数据输入端 D 相同（透明的）。当 G 端从高电平返回到低电平时（下降沿后），输入端的数据就被锁存在锁存器中，数据输入端 D 的变化不再影响 Q 端输出。

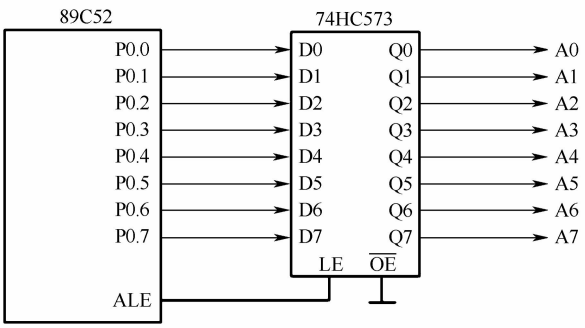


图 8-5　地址总线扩展电路

### 8.2.2　常用的数据存储器

数据存储器即随机存取存储器（Random Access Memory，RAM），用于存放可随时修改的数据信息。与只读存储器（Read Only Memory，ROM）不同，RAM 可以进行读、写两种操作。RAM 为易失性存储器，断电后所存信息立即消失。

按其工作方式，RAM 又分为静态随机存取存储器（SRAM）和动态随机存取存储器（DRAM）两种。静态 RAM 只要电源加上，所存信息就能可靠保存；而动态 RAM 则需要刷新。单片机使用的主要是静态 RAM。

MCS-51 系列单片机的数据存储器与程序存储器的地址空间是互相独立的，其片外数据存储器的空间可达 64KB，而片内数据存储器的空间只有 128B 或 256B。如果片内的数据存储器不够用时，则需进行数据存储器的扩展。

在单片机系统中，扩展数据存储器多用静态 SRAM 芯片。

1. 常用的静态 SRAM 芯片

常见的静态 SRAM 芯片有 6264（8K×8 位）、62256（32K×8 位）、628128（128K×8 位）等。其中静态 SRAM 芯片 6264 如图 8-6 所示。

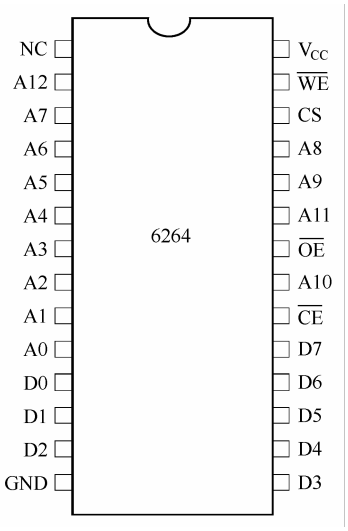


图 8-6　静态 SRAM 芯片 6264

6264 引脚含义如下：

A0～A12：地址信号引脚。

D0～D7：数据信号引脚。

$\overline{CE}$、CS：片选信号引脚，必须同时有效。

$\overline{WE}$：写允许信号引脚。

$\overline{OE}$：读允许信号引脚。

NC：空脚。

2. 扩展存储器所需芯片数目的确定

若所选存储器芯片字长与单片机字长一致，则只需扩展容量。所需芯片数目按下式确定：

芯片数目=系统扩展容量/存储器芯片容量　　（公式 8-1）

若所选存储器芯片字长与单片机字长不一致，则不仅需扩展容量，还需字扩展。所需芯片数目按下式确定：

芯片数目=（系统扩展容量/存储器芯片容量）×（系统字长/存储器芯片字长）（公式 8-2）

### 8.2.3　单片机访问片外 RAM 的操作时序

1．片外 RAM 读操作时序

片外 RAM 读操作时序如图 8-7 所示。

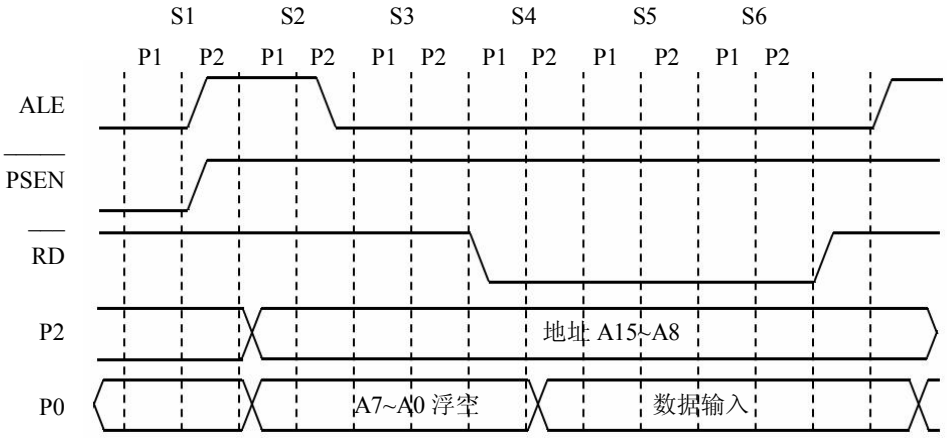


图 8-7　MCS-51 单片机访问片外 RAM 读时序

在 S1 状态 P2 节拍，ALE 信号由低变高，读周期开始。在 S2 状态，单片机把地址的低 8 位从 P0 口输出，地址的高 8 位从 P2 口输出。

在 S2 状态 P2 节拍，ALE 信号由高变低，把低 8 位地址锁存到外部的锁存器里，而高 8 位地址一直由 P2 口输出。

在 S3 状态，P0 口浮空，外部锁存器输出低 8 位地址。

在 S4 状态，$\overline{RD}$ 有效，片外 RAM 经过延迟后，把数据放在总线上，通过 P0 口输入单片机。当 $\overline{RD}$ 返回高电平后，P0 口浮空，读周期结束。

2．片外 RAM 写操作时序

片外 RAM 写操作时序如图 8-8 所示。

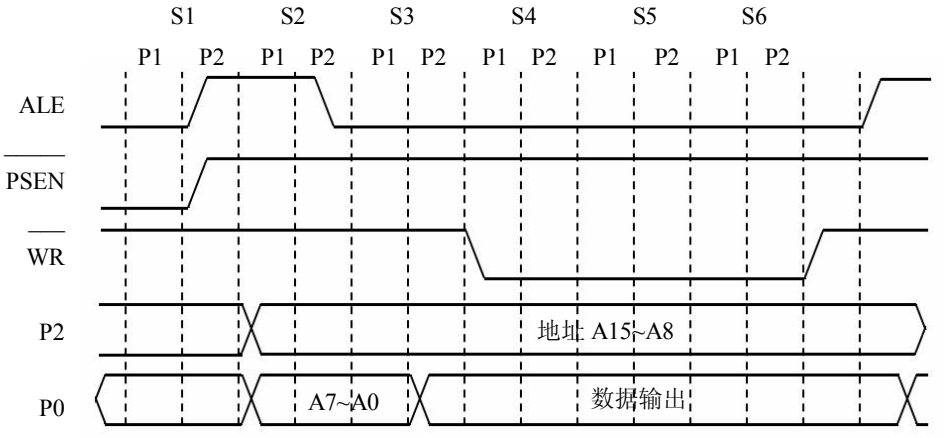


图 8-8　MCS-51 单片机访问片外 RAM 写时序

写操作时序与读操作时序，基本过程类似。不同之处主要在于 S3 状态 P2 节拍，由单片机通过 P0 口输出数据。在 S4 状态，$\overline{WR}$ 有效后，由片外 RAM 从总线上读取数据。

### 8.2.4 扩展数据存储器

存储器扩展往往需要多个存储器芯片，这些存储器芯片直接并联在扩展的系统总线上。其中低位地址线直接和每一个芯片的地址引脚相连，高位地址线作为片选信号线使用。如果空闲的高位地址线较多，可以使用线选法连接；否则需要用译码器进行译码法连接。

下面以静态 SRAM 芯片 6264 扩展 16KB 片外数据存储器为例，介绍一下 89C52 单片机系统的并行存储器扩展，连线如图 8-9 所示。

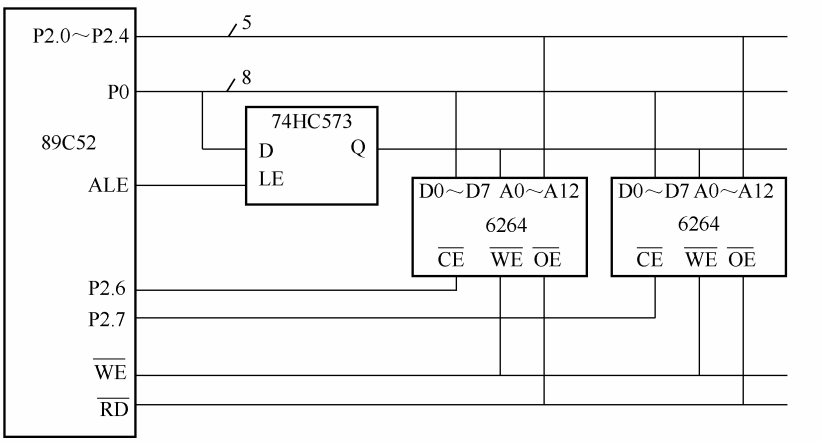


图 8-9　6264 扩展 24KB 数据存储器

根据公式 8-1 可得：芯片数目=16KB/8KB=2 片。6264 需要 13 位地址，由 P0 提供低 8 位地址，P2.0～P2.4 提供高 5 位地址。P2 口空出的口线较多，因此可选用线选法进行片选，两个芯片分别由 P2.6、P2.7 作为片选。

其地址范围分别为 10x0,0000,0000,0000B～10x1,1111,1111,1111B 和 01x0,0000,0000,0000B～01x1,1111,1111,1111B。“x”表示可以取 0 或 1，若取 1，用十六进制数表示为：A000H～BFFFH 和 6000H～7FFFH。

## 8.3　用简单芯片扩展并行 I/O 口

并行总线系统扩展可以方便地连接较多的并行接口芯片，但有些时候，我们仅希望多使用少量并口，在这种情况下，可以采用一些简单的门电路芯片来扩展 I/O 口。

### 8.3.1 扩展 I/O 口常用的门电路芯片

在很多应用系统中常采用 74 系列集成电路芯片进行并行数据输入/输出。

如果需要单向并行数据传输可以选用单向总线驱动器 74HC244，该芯片还带有三态控制，能实现总线缓冲和隔离；如果需要双向并行数据传输可以选用双向总线驱动器 74HC245。74HC245 也是三态的，有一个方向控制端 DIR。DIR=1 时输出（$A_n \rightarrow B_n$），DIR=0 时输入（$A_n \leftarrow B_n$），如图 8-10 所示。

总线驱动器对于单片机的 I/O 口只相当于增加了一个 TTL 负载，因此驱动器除了对后级电路驱动外，还能对负载的波动变化起隔离作用。在对 TTL 负载驱动时，只需考虑驱动电流

的大小；在对 MOS 负载驱动时，MOS 负载的输入电流很小，更多地要考虑对分布电容的电流驱动。

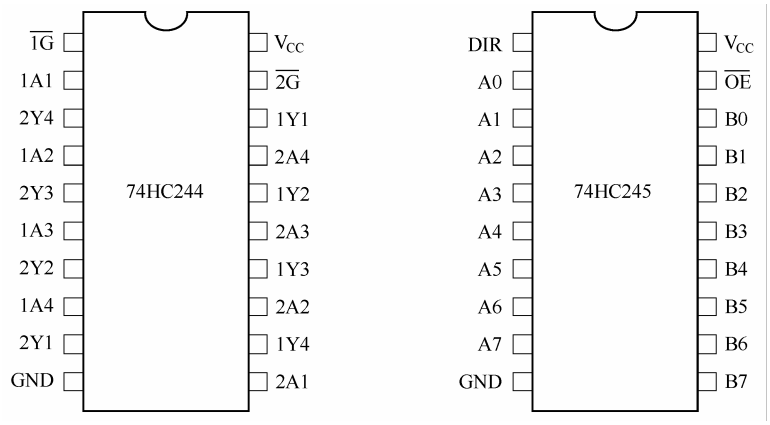


图 8-10　单向缓冲总线驱动器 74HC244 和双向缓冲总线驱动器 74HC245

对于扩展并行输出口，如果需要锁存以实现异步收发可选用锁存器芯片 74HC573，见图 8-5。输出允许引脚 $\overline{OE}$ 接单片机 P2 口的某个高位地址线以实现片选，锁存控制引脚 LE 接单片机取反后的写控制信号 $\overline{WR}$ 控制芯片输出和锁存数据。

### 8.3.2　简单扩展 I/O 口举例

**例 8-1**　利用 74 系列芯片对单片机应用系统做并行 I/O 口扩展，实现从单片机 P0 口读入 8 位开关状态，并将其状态值从 P0 口输出，控制 8 位发光二极管发光显示开关状态。

如图 8-11 所示，可通过 74HC573 和 74HC244 芯片实现单片机一个并口同时输入、输出。当对片外存储区 0x7fff 进行读写操作时，产生的写、读选通信号 $\overline{WR}$ 和 $\overline{RD}$ 分别控制 74HC573 和 74HC244 的输出、输入，P2.7 实现两个芯片的片选。读操作时，$\overline{WR}$ =1，$\overline{RD}$ =0，通过 74HC244 读入开关状态，写操作时，$\overline{WR}$ =0，$\overline{RD}$ =1，把开关状态通过 74HC573 输出驱动发光二极管发光。发光二极管是低电平点亮，刚好与输入状态的开关闭合对应。

C 语言程序如下：

```
unsigned char xdata pt1 _at_ 0x7fff;              //定义设备变量，端口地址是 0x7fff，P2.7 低电平
void main()
{
    unsigned char data tmp1;
    unsigned char data tmp2=0xff;
    while(1)                                      //循环
    {
        tmp1=pt1;                                 //从 P0 口输入数据
        if (tmp1!=tmp2)                           //判断输入数据是否有改变
        {
            tmp2=tmp1;                            //保存新数据
            pt1=tmp1;                             //从 P0 口输出数据
        }
    }
}
```

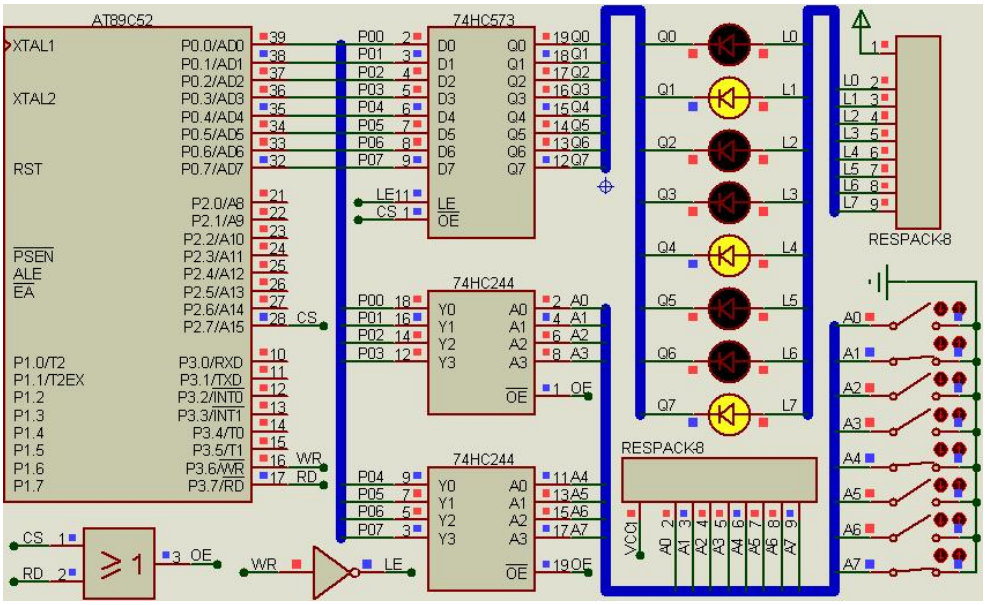


图 8-11　使用 74HC244 和 74HC573 扩展并口

## 8.4　用可编程芯片扩展并行 I/O 口

8255A 是一种通用的可编程并行 I/O 接口芯片，它拥有三个并行 I/O 口，可通过编程设置多种工作方式。通过 8255A 可以方便、快速地扩展并行 I/O 接口。8255A 可用在开关电路、键盘、打印机、A/D 和 D/A 接口等电路中。

### 8.4.1　8255A 的结构

8255A 主要由总线缓冲、读写控制逻辑、A 组与 B 组控制逻辑，以及 A 口、B 口、C 口等部分组成，如图 8-12 所示。

1. 数据总线缓冲器

这是一个双向三态的 8 位数据缓冲器，它是 8255A 与计算机系统数据总线的接口。输入输出的数据、单片机输出的控制字以及单片机输入的状态信息都是通过这个缓冲器传送的。

2. 读/写控制逻辑

用来控制把单片机输出的控制字或数据送至相应端口，也由它来控制把状态信息或输入数据通过相应的端口送到单片机。

3. 数据端口

三个数据端口 A、B、C 分成 A、B 两组。PA 和 PC4～PC7 属于 A 组；PC0～PC3 和 PB 属于 B 组。

端口 A：包含一个 8 位数据输入锁存器、一个 8 位数据输出锁存器和缓冲器。

端口 B：包含一个 8 位数据输入缓冲器、一个 8 位数据输出锁存器和缓冲器。

端口 C：包含一个 8 位数据输入缓冲器、一个 8 位数据输出锁存器和缓冲器。

4. A 组和 B 组控制部件

A 组控制电路控制 A 口和 C 口上半部，B 组控制电路控制 B 口和 C 口下半部。

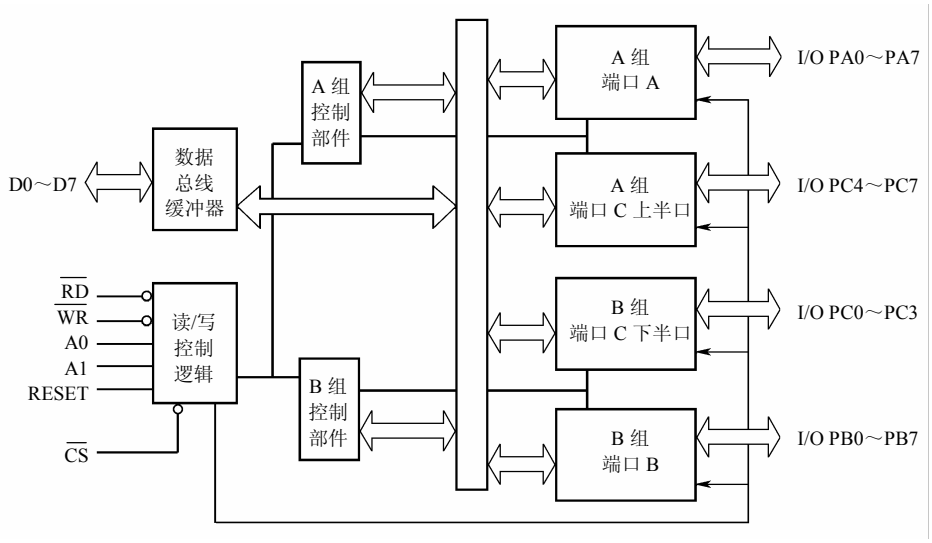


图 8-12　8255A 内部结构

### 8.4.2　8255A 的引脚定义

8255A 的引脚信号如图 8-13 所示，可以分为两组：一组是面向单片机的信号，一组是面向外设的信号。

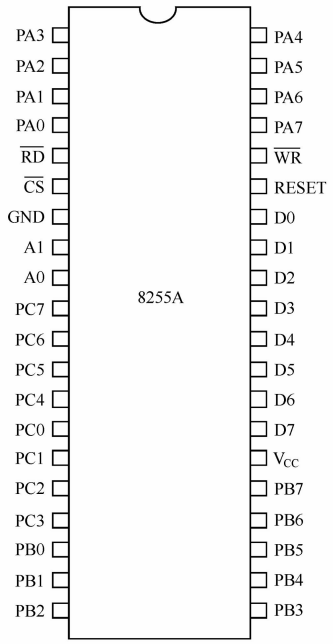


图 8-13　8255A 引脚信号

1. 面向单片机的引脚信号及功能

D0～D7：8 位，双向，三态数据线，用来与系统数据总线相连。

RESET：复位信号，高电平有效，输入，用来清除 8255A 的内部寄存器，并置 A 口、B 口、C 口均为输入方式。

$\overline{CS}$：片选，低电平有效，输入，用来决定芯片是否被选中。

$\overline{RD}$：读信号，低电平有效，输入，控制 8255A 将数据或状态信息送给 CPU。

$\overline{WR}$：写信号，低电平有效，输入，控制 CPU 将数据或控制信息送到 8255A。

A1，A0：端口地址选择信号，输入。

8255A 内部共有 4 个端口：A 口、B 口、C 口和控制口，由$\overline{CS}$、$\overline{RD}$、$\overline{WR}$以及 A1、A0 五个信号的组合来进行端口和读写方式的选择，见表 8-2。

表 8-2　8255A 端口的选择与操作

| A1 | A0 | $\overline{CS}$ | $\overline{RD}$ | $\overline{WR}$ | 操作 |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 写端口 A |
| 0 | 1 | 0 | 1 | 0 | 写端口 B |
| 1 | 0 | 0 | 1 | 0 | 写端口 C |
| 1 | 1 | 0 | 1 | 0 | 写控制寄存器 |
| 0 | 0 | 0 | 0 | 1 | 读端口 A |
| 0 | 1 | 0 | 0 | 1 | 读端口 B |
| 1 | 0 | 0 | 0 | 1 | 读端口 C |
| 1 | 1 | 0 | 0 | 1 | 读控制寄存器 |

2. 面向外设的引脚信号及功能

PA0～PA7：A 组数据信号，用来连接外设。

PB0～PB7：B 组数据信号，用来连接外设。

PC0～PC7：C 组数据信号，用来连接外设或作为控制信号。

### 8.4.3　8255A 的控制字

单片机对 8255A 的控制是通过对控制寄存器写入控制字来实现的。8255A 的控制字有两个，一个是工作方式控制字，另一个是端口 C 置 1/清 0 控制字。两个控制字是通过同一个端口写入的，通过控制字的最高位 D7（特征位）来区分。8255A 控制字端口的地址是由$\overline{CS}$、A1、A0 三个信号决定的。8255A 端口地址见表 8-3。

表 8-3　8255A 端口地址定义

| $\overline{CS}$ | A1 | A0 | 端口 | 端口性质 |
|---|---|---|---|---|
| 0 | 0 | 0 | PA | 数据口 |
| 0 | 0 | 1 | PB | 数据口 |
| 0 | 1 | 0 | PC | 数据口，通信控制/状态口 |
| 0 | 1 | 1 | CW | 控制字端口 |

1. 工作方式控制字

8255A 有三种工作方式，用户可以通过编程来设置。

方式 0：简单输入/输出方式，端口 A、B、C 三个均可。

方式 1：选通输入/输出－中断方式，端口 A、B 两个均可。

方式 2：双向输入输出－中断方式，只有端口 A 才有。

工作方式控制字就是对 8255A 的 3 个数据端口的工作方式及功能进行指定，即进行初始

化，初始工作要在使用 8255A 之前做。8255A 的工作方式控制字各位含义见表 8-4。

表 8-4　8255A 工作方式控制字定义

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| D7=1<br>特征位 | PA 口方式：<br>00=方式 0，01=方式 1<br>1x=方式 2 | | PA 口：<br>0 输出<br>1 输入 | PC4～PC7：<br>0=输出<br>1=输入 | PB 口方式：<br>0=方式 0<br>1=方式 1 | PB 口：<br>0=输出<br>1=输入 | PC0～PC3：<br>0=输出<br>1=输入 |
| | A 组控制 | | | | B 组控制 | | |

工作方式控制字最高位是特征位，一定要写 1，其余各位应根据设计的要求填写 1 或 0。

2. 端口 C 按位输出控制字

8255A 的端口 C 具有按位输出功能。即端口 C 的 8 位中的任一位，都可通过设置控制寄存器输出 1 或输出 0，而端口 C 中其他位的状态不变。注意 8255A 的端口 C 置 1/清 0 控制字的最高位 D7（特征位）应为 0。8255A 的端口 C 置 1/清 0 控制字各位含义见表 8-5。

表 8-5　8255A 端口 C 置 1/清 0 控制字定义

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| D7=0<br>特征位 | 未用 | | | 位选择：000=PC0<br>001=PC1<br>….<br>111=PC7 | | | 1 为置 1<br>0 为清 0 |

8255A 的端口 C 置 1/清 0 控制字不会影响端口的工作方式。

### 8.4.4　8255A 的工作方式

8255A 的方式 0 较为简单，方式 1 和方式 2 较为复杂。8255A 工作在方式 1 和方式 2 时，PA 和 PB 用来传送数据，PC 用来充当它们的联络控制信号和应答信号。实际中方式 1 和方式 2 使用得很少，略去不讲。

方式 0 是一种简单的输入/输出方式，没有应答联络信号，两个 8 位端口（PA、PB）和两个 4 位端口（PC），每一个都可由程序设置作为输入或输出。输出有锁存，输入没有锁存。

3 个 8 位口都设置为输出口，其控制字为：1000 0000B=80H，C 语言值为 0x80。

3 个 8 位口都设置为输入口，其控制字为：1001 1011B=9BH，C 语言值为 0x9b。

PA、PB 为输出，PC 为输入，其控制字为：1000 1001B=89H，C 语言值为 0x89。

PA、PB 为输入，PC 为输出，其控制字为：1001 0010B=92H，C 语言值为 0x92。

### 8.4.5　8255A 的应用举例

**例 8-2**　使用 8255A 的 PA、PC 口接 16 个发光二极管显示流水灯，如图 8-14 所示。其中用单片机的 P2.7 作为 8255A 的片选信号，P2.5、P2.6 作为 8255A 的端口地址选择信号。

低 13 位地址没有使用可取任意值，取为 1，根据表 8-3，可确定 PA、PB、PC 和控制端口 P2.7、P2.6、P2.5 的取值，从而确定 PA、PB、PC 和控制端口的地址分别为 0x1fff、0x3fff、0x5fff、

0x7fff。图 8-14 中的 LED 阴极接 8255A，8255A 输出低电平 LED 点亮。电路中，8255A 输出应该加驱动，可以使用 74HC244、74HC245 等，为了使电路不至于过大而未画出。

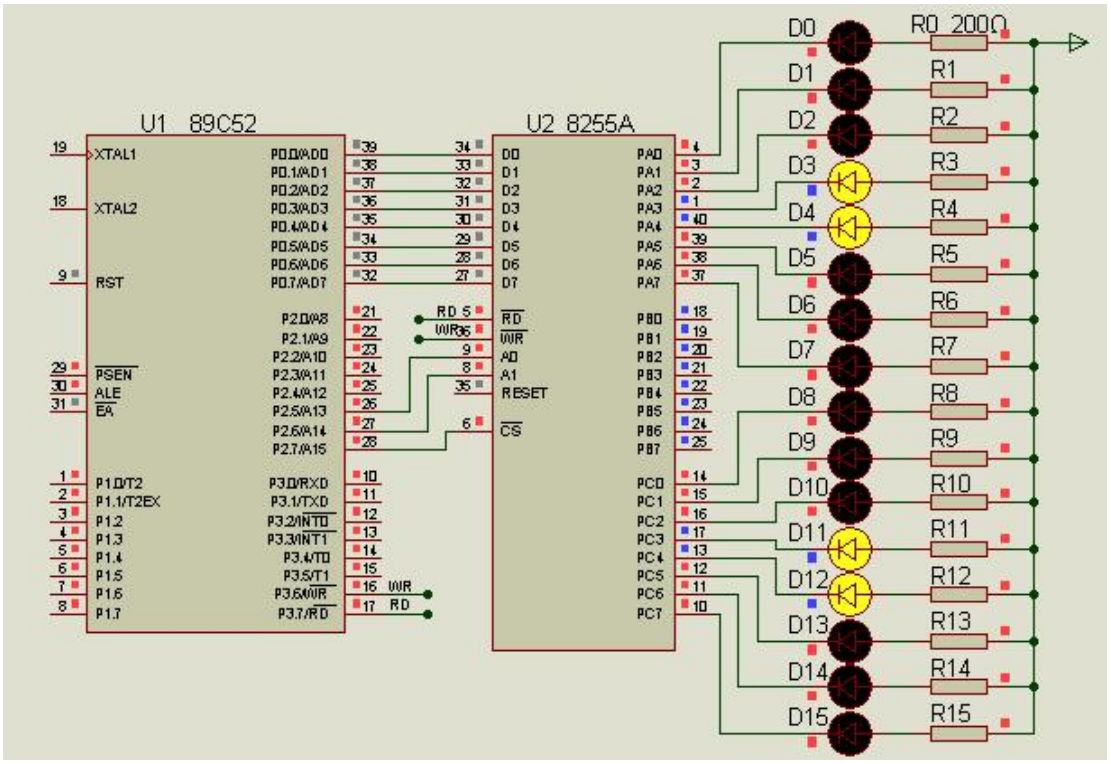


图 8-14　用 8255A 扩展并行口输出显示流水灯

C 语言程序如下：

```
#include <reg52.h>
#include <intrins.h>
unsigned char xdata PA _at_ 0x1fff;              //定义设备变量 PA 口地址为 0x1fff
unsigned char xdata PC _at_ 0x5fff;              //定义设备变量 PC 口地址为 0x5fff
unsigned char xdata CW _at_ 0x7fff;              //定义设备变量 CW 口地址为 0x7fff

void delay10ms(unsigned char x)                  //延时 10ms 函数（设振荡频率为 12MHz）
{
    unsigned int i;
    while(x--)
        for(i=0;i<830;i++);                      //试验得出需要内循环 830 次
}

void main()                                      //主函数
{
    unsigned int data tmp0,tmp1;
    CW =0x80;                                    //写控制字，三个口均设置为输出
    tmp0=0x303;                                  //PA 口和 PC 口都同时点亮 2 个 LED
    while(1)                                     //死循环
    {   tmp1=~tmp0;                              //取反，用于输出显示
```

```
            PA=tmp1&0xff;                    //从 8255A 的 PA 口输出低 8 位
            PC=tmp1/0x100;                   //从 8255A 的 PC 口输出高 8 位
            delay();                         //延时
            tmp0=_irol_(tmp0,1);             //调用循环左移函数，循环左移 1 位
        }
}
```

## 8.5　用串行方式扩展数据存储器

串行总线系统扩展技术是新一代单片机技术发展的一个显著特点。相对于并行总线接口，串行总线接口有着占用 I/O 口线少（一般 2～4 根），编程相对简单，易于实现应用系统软硬件的模块化、标准化。随着串行总线接口技术（SPI、$I^2C$ 等）和各种串行接口芯片的发展，串行总线接口技术越来越受到人们的推崇。

$I^2C$ 是常见的串行通信协议，现在已经广泛地应用于 IC 之间的通信中。并且不少单片机已经整合了 $I^2C$ 的接口。像 89C52 等不支持 $I^2C$ 的单片机，也可以通过编程，用 I/O 引脚模拟 $I^2C$ 通信协议扩展 $I^2C$ 接口。

### 8.5.1　$I^2C$ 总线及操作

$I^2C$（Inter-Integrated Circuit，集成电路间串行传输总线）是一种由 Philips 公司开发的两线式串行总线，用于连接各种集成电路芯片、微控制器及其外围设备。它以 1 根串行数据线（SDA）和 1 根串行时钟线（SCL）实现了双工的同步数据传输。具有接口线少、控制方式简单、器件封装形式小、通信速率较高等优点。

1．$I^2C$ 总线的特点

$I^2C$ 总线最主要的优点是其简单性和有效性。由于接口直接集成在组件之上，因此 $I^2C$ 总线占用的空间非常小，减少了电路板的空间和芯片管脚的数量，降低了互联成本。总线的长度可高达 25 英尺（6.35 米），并且能够以 10Kb/s 的最大传输速率支持 40 个组件。

$I^2C$ 总线的另一个优点是它支持多主控（Multimastering），其中任何能够进行发送和接收的设备（节点）都可以成为主控器。一个主控器能够控制信号的传输和时钟频率。当然，在任何时间点上只能有一个主控器。如图 8-15 所示。

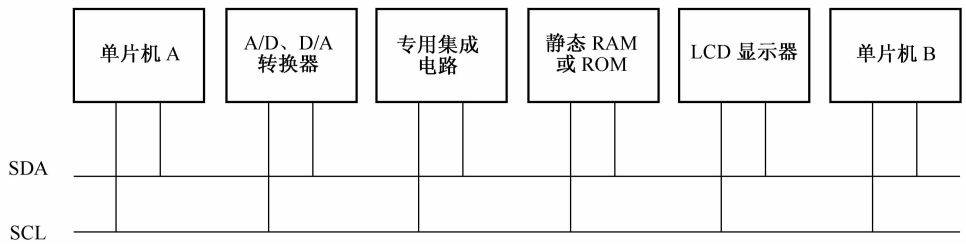


图 8-15　典型 $I^2C$ 总线系统示意图

$I^2C$ 总线是由数据线 SDA 和时钟 SCL 构成的串行总线，可发送和接收数据。在单片机与被控 IC 之间、IC 与 IC 之间进行双向传送，最高传送速率 100kb/s，驱动能力 400pF。每个连接在 $I^2C$ 总线上的电路或模块称之为一个节点，各个节点均并联在这条总线上，每个节点都有唯一的地址，在信息的传输过程中，$I^2C$ 总线上并接的每一个节点既是主控器（或被控器），

又是发送器（或接收器），这取决于它要完成的功能。

各节点供电可以不同，但需共地，SDA 和 SCL 需分别接上拉电阻。

$I^2C$ 总线支持多主和主从两种工作模式。在多主方式中，通过硬件和软件的仲裁，主控器取得总线控制权。在主从方式中，从器件地址包括器件编号地址和引脚地址两部分，器件编号地址由 $I^2C$ 总线委员会分配，引脚地址由外界电平的高低决定。当器件内部有连续的子地址空间时，对这些空间进行连续读写，子地址会自动加 1。

单片机发出的控制信号分为地址码和控制量两部分，地址码用来选址，即接通需要控制的电路，确定控制的种类；控制量决定该调整的类别（如对比度、亮度等）及需要调整的量。这样，各控制电路虽然挂在同一条总线上，却彼此独立，互不相关。

2. $I^2C$ 总线的时序

$I^2C$ 总线在传送数据过程中有 4 种类型信号：开始信号、结束信号、数据信号和应答信号。

1）开始信号：SCL 为高电平时，SDA 由高电平向低电平跳变，开始传送数据。

2）结束信号：SCL 为低电平时，SDA 由低电平向高电平跳变，结束传送数据。

3）数据信号：其格式为每个时钟传送 1 位数据，在时钟的低电平由发送方发出数据电平信号，高电平时接收方读取数据线上的数据。8 位数据信号构成数据字节或地址字节。

4）应答信号：接收数据的 IC 在接收到 8bit 数据（包括命令）后，在第 9 个时钟，向发送数据的 IC 发出低电平脉冲，作应答信号，表示已收到数据，如图 8-16 所示。应答信号与数据信号格式一样。

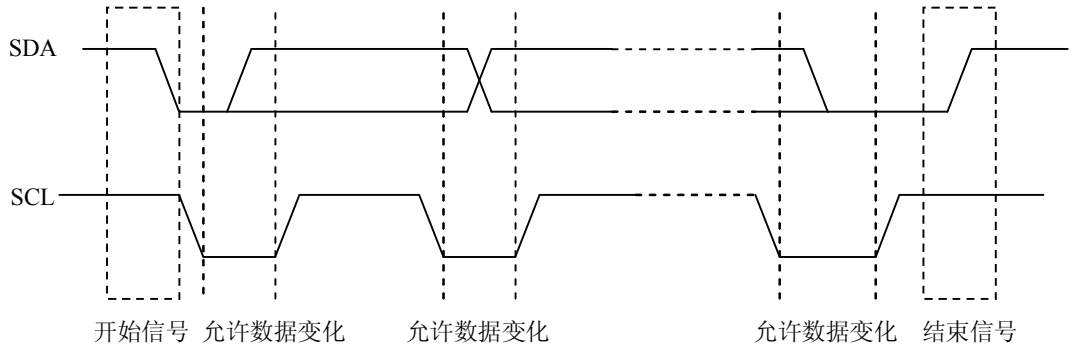


图 8-16　$I^2C$ 总线的时序

单片机向受控单元发出一个信号后，等待受控单元发出一个应答信号，单片机接收到应答信号后，根据实际情况作出是否继续传递信号的判断。若未收到应答信号，就判断为受控单元出现故障。

不论主控器是向被控器发送还是读取信息，开始信号和结束信号都由主控器发出。

3. $I^2C$ 总线的数据传输过程

$I^2C$ 总线以开始信号为启动信号，接着传输的是寻址字节和数据字节，数据字节是没有限制的，但每个字节后必须跟随一个应答位（0），全部数据传输完毕后，以结束信号结尾。

$I^2C$ 总线上传输的数据和地址字节均为 8 位，且高位在前，低位在后。

数据传输时，主机先发送启动信号和时钟信号，随后发送寻址字节来寻址被控器件，并规定数据传送方向。$I^2C$ 总线的寻址字节格式如图 8-17 所示，高 7 位为从器件地址，最低位为数据方向。从器件地址包括器件类型编号和引脚地址两部分，器件类型编号为器件识别码，由

$I^2C$ 总线委员会分配，引脚地址（D3～D1 位），应该与器件的引脚 A2～A0 电平一致。

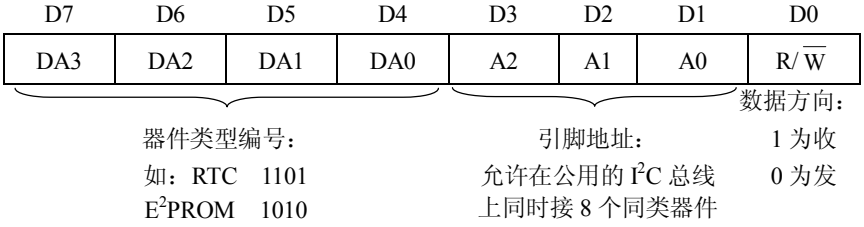


图 8-17　$I^2C$ 总线的寻址字节格式

当主机发送寻址字节时，总线上所有器件都将其中的高 7 位地址与自己的比较，若相同，则该器件根据读/写位确定是从发送器还是从接收器。

若为从接收器，在寻址字节之后，主控发送器通过 SDA 线向从接收器发送信息，信息发送完毕后发送终止信号，以结束传送过程。

若为从发送器，在寻址字节之后，主控接收器通过 SDA 线接收被控发送器的发送信息。

每传输一位数据都有一个时钟脉冲相对应。时钟脉冲不必是相同周期的，它的时钟间隔可以不同。

总线备用时（“非忙”状态），SDA 和 SCL 都为“1”。只有当总线处于“非忙”状态时，数据传输才能被初始化。

关闭 $I^2C$ 总线（等待状态）时，使 SCL 箝位在低电平。SCL 的“线与”特性：SCL 为低电平时，SDA 上数据就被停止传送。

当接收器接收到一个字节后无法立即接收下一个字节时，便向 SCL 线输出低电平而箝住 SCL（SCL=0），迫使 SDA 线处于等待状态，直到接收器准备好接收新的字节时，再释放时钟线 SCL（SCL=1），使 SDA 上的数据传输得以继续进行。

如图 8-18 中的 A 处，当接收器在 A 点接收完主控器发来的一个字节时，需要处理接收中断而无法继续接收，则被控器便可箝住 SCL 线为低电平，使主控发送器处于等待状态，直到被控器处理完接收中断后，再释放 SCL 线。

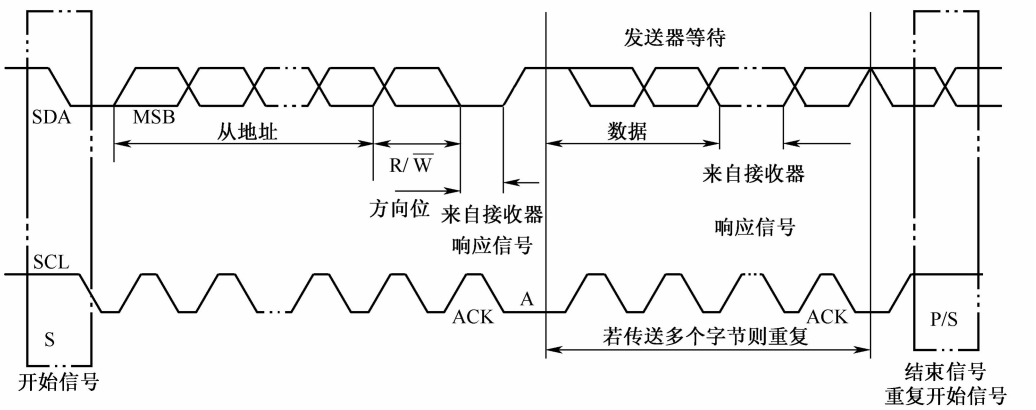


图 8-18　$I^2C$ 总线的数据传送字节格式

数据传输时，发送器每发完一个字节，都要求接收方发回一个应答信号（0）。应答信号的时钟仍由主控器在 SCL 上产生。主控发送器必须在被控接收器发送应答信号前，预先释放

对 SDA 线的控制（SDA＝1），以便主控器对 SDA 线上应答信号的检测。

主控器发送时，被控器接收完每个字节需发回应答信号，主控器据此进行下一字节的发送。如果被控器由于某种原因无法继续接收 SDA 上数据时，可向 SDA 输出一个非应答信号（1），主控器据此便产生一个 Stop 来终止 SDA 线上的数据传输。

主控器接收时，也应给被控器发应答信号。当主控器接收被控器送来的最后一个数据时，必须给被控器发一个非应答信号（1），令被控器释放 SDA 线，以便主控器可以发送 Stop 信号来结束数据的传输，如图 8-19 所示。

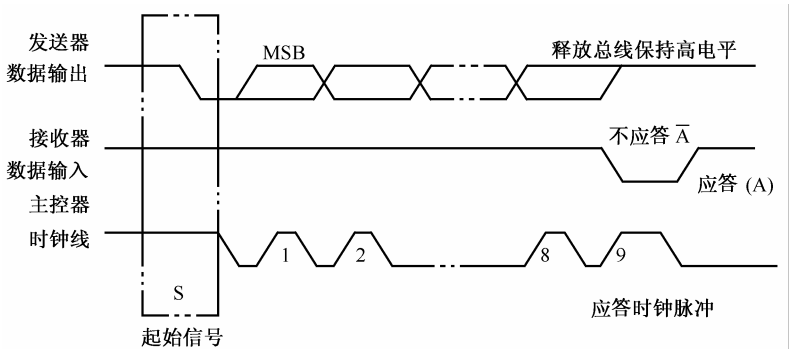


图 8-19　$I^2C$ 总线的应答信号

4. $I^2C$ 的数据格式

1）主控器写数据。主机向被寻址的从机写入 n 个数据字节。整个过程均为主机发送，从机接收，数据的方向位 R/W=0。应答位 ACK 由从机发送，当主机产生结束信号后，数据传输停止。格式如下：

| S | SLA $\overline{W}$ | A | Data 1 | A | Data 2 | A | … | Data n-1 | A | Data n | A/ $\overline{A}$ | P |
|---|---|---|---|---|---|---|---|---|---|---|---|---|

S 为开始信号，P 为结束信号，A 为应答信号，$\overline{A}$ 为非应答信号，SLA $\overline{W}$ 为寻址字节（写），Data 1～Data n 为被传送的 n 个数据。

▭ 为主控器发送，被控器接收。　▭ 为被控器发送，主控器接收。

2）主控器读数据。主机从被寻址的从机读出 n 个数据字节。寻址字节为主机发送、从机接收，方向位 R/W=1，n 个数据字节均为从机发送、主机接收。主机接收完全部数据后发非应答位（1），表明读操作结束。格式如下：

| S | SLA R | A | Data 1 | A | Data 2 | A | … | Data n-1 | A | Data n | $\overline{A}$ | P |
|---|---|---|---|---|---|---|---|---|---|---|---|---|

SLA R 为寻址字节读。

3）主控器读/写数据。主机在一段时间内为读操作，在另一段时间内为写操作。在一次数据传输过程中需要改变数据的传送方向。由于读/写方向有变化，开始信号和寻址字节都会重复一次，但读/写方向（R/W）相反。格式如下：

| S | SLA R | A | Data 1 | A | Data 2 | A | … | Data n | A | Sr | SLA $\overline{W}$ | A |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATA 1 | | A | DATA2 | | A | … | DATA n-1 | | A | DATA n | | A/ $\overline{A}$ | P |

Sr 为重复开始信号，Data 1～Data n 为主控器的读数据，DATA 1～DATA n 为主控器的写数据。

### 8.5.2　I²C 总线扩展存储器

**例 8-3**　对单片机编程，模拟 I²C 操作对 EEPROM 芯片 24C04A（8bit×256×2，512 字节）进行写、读，对前面的 128 字节写入 0～128。

24C04A 的 SDA 引脚是 I²C 数据信号引脚；SCK 引脚是 I²C 时钟信号引脚；A1、A2 引脚是芯片选择引脚，当系统中存在多个 24C04A 时实现片选；WP 引脚是对高 256 字节的写保护。

在 Proteus 中画出的电路如图 8-20 所示。图中左上角为显示的 24C04A 内的数据，可以通过 Proteus 的 Debug 菜单的 I2C Memory Internal Memory 选项查看 24C04A 内的数据。

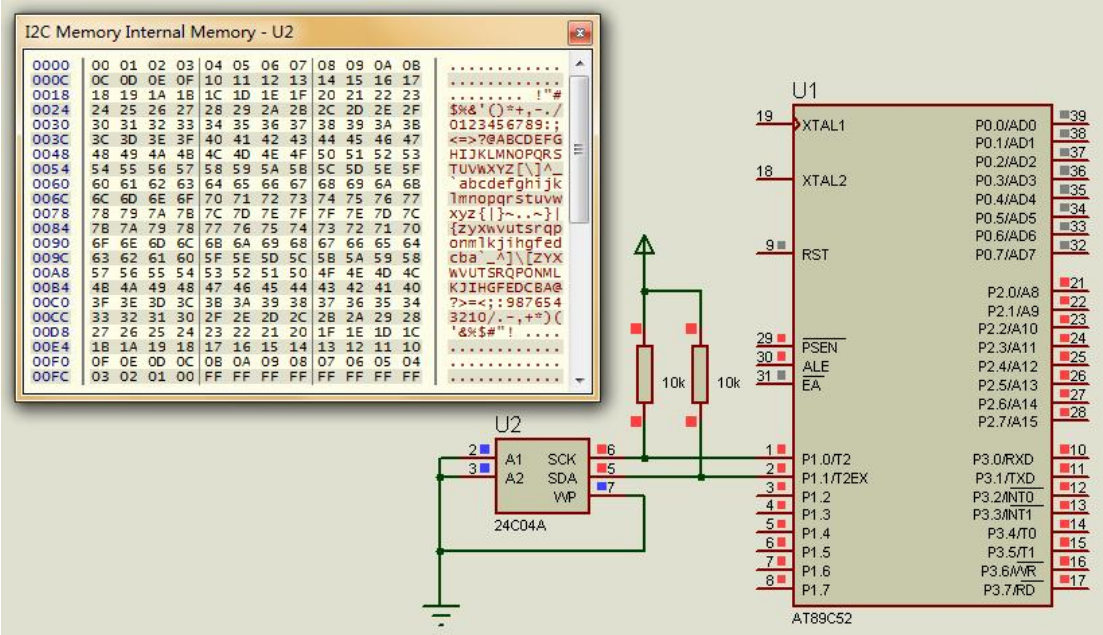


图 8-20　89C52 与 EEPROM 芯片 24C04A 接口

按照 I²C 总线的操作方法，C 语言程序如下。

```
#include<reg52.h>                    //包含 52 系列寄存器定义的头文件
#include<intrins.h>                  //包含空操作函数定义的头文件
sbit SCL=P1^0;                       //I²C 时钟线
sbit SDA=P1^1;                       //I²C 数据线
void start()                         //定义起始信号产生函数
{
    SDA=1;
    SCL=1;
    _nop_();
    _nop_();
    SDA=0;
    _nop_();
    _nop_();
    SCL=0;
```

```
}
void stop()                                          //定义停止信号产生函数
{
    SDA=0;
    SCL=0;
    _nop_();
    _nop_();
    SCL=1;
    _nop_();
    _nop_();
    SDA=1;
}
void rack()                                          //定义应答检测函数
{
    SCL=1;
    _nop_();
    _nop_();
    SCL=0;
}
void nack()                                          //定义不应答信号函数
{
    SDA=1;
    SCL=1;
    _nop_();
    _nop_();
    SCL=0;
    SDA=1;
}
void wbyte(unsigned char tmp)                        //写字节函数
{
    unsigned char i;
    for(i=0;i<8;i++)
    {    tmp<<=1;                                    //左移一位，移出位在 CY
         SDA=CY;                                     //移出位发送
         SCL=1;
         _nop_();
         _nop_();
         SCL=0;
    }
    rack();
}
void wdata(unsigned char addr,unsigned char dat)     //24C04A 写数据函数（格式为先地址，后数据）
{
    unsigned char i;
    start();
    wbyte(0xa0);                                     //器件寻址
    wbyte(addr);                                     //24C04A 内部地址
    wbyte(dat);                                      //写数据
```

```
    stop();
    for(i=0;i<255;i++);                    //每写一个数据，都要延时一段时间
    for(i=0;i<255;i++);
}
unsigned char rbyte()                      //读字节函数
{
    unsigned char i,d;
    for(i=0;i<8;i++)
    {    SCL=1;
         d<<=1;
         d=d|SDA;
         SCL=0;
    }
    return d;
}
unsigned char rdata(unsigned char addr)    //读 24C04A
{
    unsigned char d;
    start();
    wbyte(0xa0);                           //器件寻址
    wbyte(addr);                           //24C04A 内部地址
    start();
    wbyte(0xa1);                           //读命令
    d=rbyte();                             //读数据
    nack();
    stop();
    return d;
}
void main()                                //主函数
{
    unsigned char i,d;
    for(i=0;i<128;i++)                     //向 24C04A 前 128 字节写 0～127
         wdata(i,i);
    for(i=0;i<128;i++)                     //把 24C04A 前 128 字节读出并从 255 地址倒着写入
    {    d=rdata(i);
         wdata(255-i,i);
    }
    while(1);
}
```

## 思考题与习题

1．什么是接口？接口的功能有哪些？

2．简述接口的组成结构。

3．什么是端口？简述端口编址的方法。

4．8255A 的功能是什么？

5．简述 8255A 的内部结构。

6．简述 8255A 的控制字和用法。

7．8255A 有哪些工作方式？这些工作方式有什么不同？

8．简述 $I^2C$ 时序的特点。

9．6 根地址线最多可产生多少个地址？11 根地址线最多可产生多少个地址？

10．假定一个存储器有 4096 个存储单元，其首地址为 0，则末地址为多少？

11．用 2K×4 位的数据存储器芯片扩展 4K×8 位的数据存储器需要多少片？地址总线是多少位？画出连线图。

12．在 Proteus 中绘制单片机应用系统，参考图 8-11，用两片 74HC573 芯片扩展 89C52 的输出端口，实现 16 个发光二极管做流水灯显示（每次亮 1 个、2 个），用 Keil C 编程实现该功能，并把编译好的代码下载到单片机中模拟运行。提示：两个 74HC573 的 $\overline{OE}$ 引脚分别接单片机的 P2.6、P2.7 引脚，两个 74HC573 的端口地址分别是 0xbfff 和 0x7fff。

13．在 Proteus 中绘制单片机应用系统，用 8255A 的 PA 口接 8 个开关，控制 PB 口接的对应的 8 只发光二极管点亮、熄灭。

14．在 Proteus 中绘制单片机应用系统，用 8255A 的 PA、PB、PC 接 24 只发光二极管以 1s 为周期交替闪烁。

15. 在 Proteus 中绘制单片机应用系统，参考例 8-4，编程模拟 $I^2C$ 把 EEPROM 芯片 24C04A 的 20H～40H 写为 0x55。

16. 在 Proteus 中绘制单片机应用系统，参考例 8-4，编程模拟 $I^2C$ 从 24C04A 的 30H～50H 中读数据写入数组 BUF。