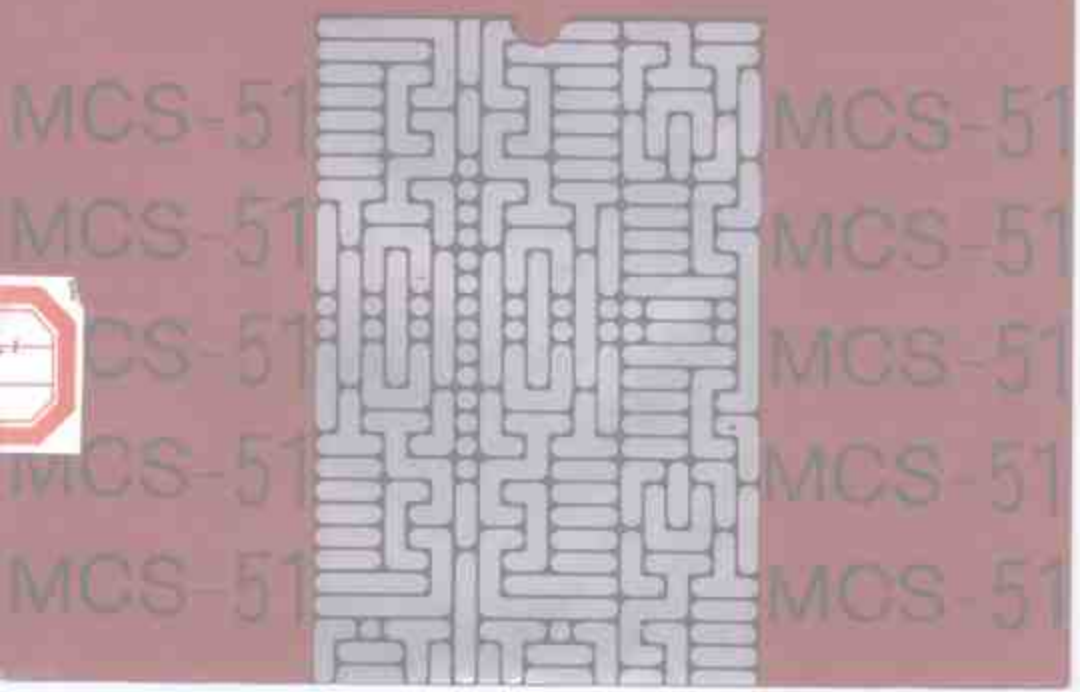
哈尔滨工业大学出版社

MCS-51 单片机原理及接口技术

修订版

马家辰 孙宏德 刘涛 编



修订版

MCS-51单片机原理及接口技术

马家辰　孙玉德　张颖　编

哈尔滨工业大学出版社

内 容 简 介

本书全面系统地介绍MCS-51单片机的结构、原理、接口技术、扩展应用等知识，主要内容包括：计算机运算基础，计算机硬件电路基础，单片微型机的组成原理，MCS-51系列单片机的指令系统，汇编语言程序设计，MCS-51单片机的扩展应用，MCS-51单片机接口技术，最新增强型51系列兼容单片机介绍，单片机指令一览表和常用芯片的引脚图等。

本书可作为高等理工科院校非计算机专业计算机原理和单片机课程的教材，也可供工程技术人员参考。

MCS-51单片机原理及接口技术

MCS-51 Danpianji Yuanli ji Jiekoujishu

（修订版）

马家辰　孙玉德　张颖　编

*

哈尔滨工业大学出版社出版发行

（哈尔滨市南岗区复兴街18号 150001）

肇东粮食印刷厂印刷

*

开本 787×1092　1/16　印张 20　字数 459 千字

1998年9月第2版　1998年9月第2次印刷

印数 6 001—11 000

ISBN 7-5603-1207-1/TP·95　定价 19.80 元

再 版 前 言

单片机具有集成度高、功能强、可靠性高、结构简单、易于掌握、应用灵活和价格低等优点，在工业控制、机电一体化、智能仪表、通信、家用电器等诸多领域中得到了广泛应用。单片机的应用提高了机电设备的技术水平和自动化程度，成为产品更新换代的重要手段。因此，高等理工科院校师生和工程技术人员了解和掌握单片机的原理、结构和应用技术是十分必要的。

我们针对非计算机专业师生和工程技术人员的特点，结合自己长期从事计算机原理和单片机教学与科研工作的体会编写了本书，其主要目的是使读者掌握计算机的基本原理和操作方法。

就单片机本身而言，它具有完整的计算机结构，而且各种机型越来越多，功能越来越强。正是考虑到单片机的这一特点，在内容安排上，我们力求体现从易到难、循序渐进的原则，在书中增加了计算机的基础知识、模型机的结构和工作过程，并以目前广泛使用的MCS-51单片机为典型介绍了一般计算机的原理和汇编语言程序设计，以及MCS-51单片机的结构、原理，从而使读者可以在没有计算机系统知识的情况下阅读本书，并通过本书建立一个完整的计算机系统知识体系，达到举一反三的目的，这也是很多教师在计算机教学中总结出来的经验。

考虑到实际应用及发展的需要，本书着重介绍了单片机汇编语言的程序设计、系统扩展、接口技术、最新增强型51系列单片机等内容，给出了许多具体的电路和应用实例，在存储器方面侧重于较大容量的存储器扩展的介绍，对常用芯片也作了比较详细的说明。本书在附录部分给出了MCS-51单片机指令速查表、常用芯片的引脚图等，为读者查阅提供了方便。

本书第一、二、三章及附录由张颖编写，第四、七、九章由马家辰编写，第五、六、八章由孙玉德编写。

在编写及修订过程中，我们参考了部分有关书刊、资料，在此对作者一并表示感谢。

由于水平有限，书中不妥之处在所难免，恳请读者批评指正。

编 者

1998年6月

目　录

第一章 绪 论

第一节 计算机的分类与发展

世界上第一台可以用程序控制的计算机被称为电子数字积分器与计算器(Electronic Numerical Integrator And Calculator),简称 ENIAC,是 1946 年由美国宾夕法尼亚大学研制的。这台计算机的字长为 12 位,主存储器只有 17K,运算速度为每秒 5 000 次加法运算,但它却是庞然大物。ENIAC 共使用 18 800 个电子管,1 500 个继电器,占地面积为 150m^2,重 30t,耗电 150kW,造价为 100 多万美元。今天看来,这台计算机既贵且重,运算速度低,字长不够长,而且耗电多。但它正是今天大小不一、花样繁多的各种类型电子计算机的先驱,为计算机技术的发展奠定了基础。如果该 ENIAC 称为第一代电子计算机的话,至今已发展至第四代超大规模集成电路计算机。

第一代(1946～1958 年)电子管数字计算机

计算机的逻辑元件采用电子管,主存储器采用磁鼓、磁芯,外存储器已开始采用磁带;软件主要用机器语言编制,后期逐步发展了汇编语言。主要用于科学计算。

第二代(1958～1964 年)晶体管数字计算机

计算机的逻辑元件采用晶体管,主存储器采用磁芯,外存储器已开始使用磁盘;软件已开始有很大的发展,出现了各种高级语言及编译程序。此时,计算机速度明显提高,耗电下降,寿命提高。计算机已发展至用于各种事务处理,并开始用于工业控制。

第三代(1964～1971 年)集成电路计算机

计算机的逻辑元件采用小规模和中规模集成电路,即所谓的 SSI 和 MSI;软件发展更快,已有分时操作系统,应用范围日益扩大。

第四代(1971 年以后)大规模集成电路计算机

计算机的逻辑元件采用大规模集成电路。所谓的大规模集成电路(LSI)是指在单片硅片上可集成 1 000 至 20 000 个晶体管的集成电路。由于 LSI 的体积小,耗能减少,可靠性高,因而促使计算机以极快的速度发展。

目前计算机的发展动向一是向大型、巨型化发展,二是向小型、微型化发展。

1. 大型、巨型计算机

为了适应现代科学技术发展的需要，要求计算机提高运算速度，加大主储容量，为此出现了大型和巨型计算机。如美国的克雷公司生产的 Cray-1、Cray-2、Cray-3 巨型计算机是比较著名的巨型计算机。我国的银河 I 就是每秒 10 亿次并行巨型计算机。目前，只有少数几个国家有能力生产巨型计算机，它象征着一个国家的科技实力。

2. 小型、微型计算机

大型机速度快，容量大，解决了过去无法计算的实时及复杂的数学问题，但是由于设备庞大，价格昂贵，给普及和应用带来了一定困难。另一方面，为了适应宇航、导弹技术及一般应用的要求，体积小、造价低、高可靠性就成了问题的关键，小型机特别是微型机的出现有效地解决了这个问题。

所谓的微型计算机(Microcomputer，简称 MC)是指把计算机的心脏——中央处理器(CPU)集成在一小块硅片上。为了区别于大、中、小型计算机的 CPU，而称微型计算机的 CPU 芯片为微处理器 MPU(Microprocessing Unit 或 Microprocesser)。

微型计算机除有 MPU 作为中央处理器以外，还有以大规模集成电路制成的主存储器和输入输出接口电路，三者之间是采用总线结构联系起来的。如果再配上相应的外围设备如显示器(CRT)、键盘及打印机等，这就成为微型计算机系统(Microcomputer System)。

目前，微型计算机功能已经很强，比如“奔腾”(Pentium)586 CPU 的集成度已达到 300 多万只晶体管，时钟频率高达 200MHz。由于结构简单、通用性强、价格便宜，微型计算机已成为现代计算机领域中的一个极为重要的分支，正在突飞猛进地发展。

第二节　计算机的应用

众所周知，计算机能控制机床自动加工复杂的零件，能使火箭准确地进入轨道，使导弹准确击中目标；可以代替人管理城市交通，实现航空和火车的调度，银行储蓄可以通存通兑；可以编辑稿件，自动排版；可以代替医生诊断疾病，自动开药方和假条；与计算机对弈，连优秀的棋手也会失败，……。现代科学的发展使计算机应用的领域已极其广泛，概括起来，可以归纳为以下几个主要方面：

1. 科学计算

计算机广泛地应用于科学技术方面的计算，这是计算机应用的一个基本方面，也是我们比较熟悉的。如：人造卫星轨迹计算，导弹发射的各项参数的计算，房屋抗震强度的计算，24 小时的天气预报等，通常需要求解几十阶微分方程组，进行大型矩阵运算。

2. 数据处理

用计算机对数据及时地加以记录、整理和计算，加工成人们所要求的形式，称为数据处理。通常，在生产组织、企业管理、市场预测、情报检索等方面，存在着大量的数据需要及时进行搜集、归纳、分类、整理、存储、检索、统计、分析、列表、绘图等。这类问题数据量大，而运算又比较简单，包含大量的逻辑运算与判断，其处理结果往往以表格或文件形式存储或输出。

据统计，目前在计算机应用中，数据处理所占的比重最大。它使人们从大量繁杂的数据统计管理事物中解放出来，大大提高了工作质量、管理水平和效率。

随着计算机的普及，在数据处理方面的应用还将继续扩大与深入。

3. 自动控制

自动控制也是计算机应用的一个重要方面。在生产过程中，采用计算机进行自动控制，可以大大提高产品的数量和质量，提高劳动生产率，改善人们的工作条件，减少原材料的消耗，降低生产成本。如：航天飞行、宇航空间站的发射、对接、测控，代替人类进行有害危险工序的现场操作、控制等。

4. 辅助设计

计算机辅助设计，简称 CAD(Computer Aided Design)。用计算机辅助人们进行设计工作，如设计飞机、汽车、房屋、服装、集成电路等，使设计工作自动化。

由于 CAD 技术的迅速发展，应用范围不断扩大，又派生出许多新的技术分支，如计算机辅助制造(Computer Aided Manufacture，简称 CAM)、计算机辅助教学(Computer Aided Instruction，简称 CAI)等等，有些技术的应用及发展提高了机械、电子等行业的设计水平和自动化水平。

5. 系统仿真

计算机仿真是指应用计算机来模仿实际的系统，这是新兴的计算机应用领域，有着广泛的应用前景。如大型电站仿真、航天飞机的仿真、火箭的仿真、汽车的仿真等等。在计算机仿真系统上进行实验、研究，可以节约大量资金，并且实验安全。目前像飞机、汽车的驾驶培训，已经开始使用飞机、汽车仿真系统，学员可以在仿真系统上进行各种训练。

6. 智能模拟

智能模拟是用计算机软硬件系统模拟人类某些智能行为，如感知、思维、推理、学习、理解等理论和技术。它是在计算机科学、控制论、仿生学和心理学等基础上发展起来的边缘学科。这正是国内外争先研究的人工智能技术，它包括专家系统、模式(声、图、文)识别、问题求解、定理证明、机器翻译、自然语言理解等等。据最新报道，日本富士通公司目前正试图开发一种能识别人脑思维的计算机。这种具有智能的计算机，不再需要敲击键盘，甚至不需对计算机讲话，只要操作人员想一个单词，如“Yes”或“No”，就可以控制计算机。

7. 计算机网络与信息高速公路

(1)计算机网络

计算机网络是计算机技术和数字通讯技术发展并相融合的产物。它是指多个独立的计算机系统之间通过通讯线路、专用电缆、微波卫星、光导纤维等各种通讯介质进行数据、通信、资源共享(软件、硬件、数据库等)，而成为联系在一起的具有多种功能的网络系统。

如 Internet(国际计算机互联网络)是由国际上成千上万个计算机遵循 Internet 的有关网络协议在物理上互相连接，为实现信息交互而组成的联合体。Internet 的服务包括电子邮件、信息查询、购物、健康咨询、电子报刊以及娱乐，还可以刊登广告，通过它进行公司的跨国管理。

(2)信息高速公路

1993 年 9 月 15 日，美国正式推出跨世纪的“国家信息基础设施”工程计划(National

Information Infrastracture，简称 NII 计划），也有人称之为信息高速公路（Information Highway）。

信息高速公路的含义是建设 21 世纪“信息国道”，“建设全国性的信息网络”，将全国的研究机构、学校、办公室、图书馆、家庭等都连在一起，使每个人都能平等地享受信息资源。有人预计，到 21 世纪初将会在世界范围内形成“信息高速公路”联网热潮，由地区或国家扩展至几个国家或几个地区，最终形成“全球高速信息网络”，冲破地理位置和时间限制，从而大幅度提高社会生产力，加快人类各种文化交流与沟通，促进人类社会向更高级阶段发展。

第三节　微型计算机的系统组成

计算机系统通常是由硬件及软件系统两大部分组成，如图 1-1。

硬件（Hardware）是指实际的物理设备，它包括计算机的主机及其外部设备。

软件（Software）是指计算机完成某一工作的程序，包括系统软件、程序设计语言及应用软件等。

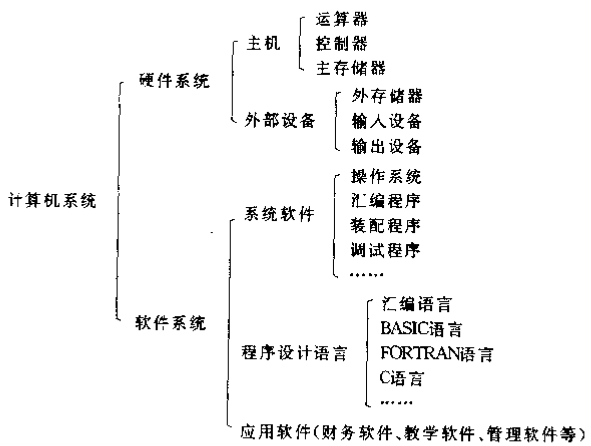


图1-1　计算机系统组成

计算机原理主要介绍计算机的硬件原理。其组成框图如图 1-2。

1. 主机

主机一般包括运算器（Arithmetic Logic Unit）、控制器（Control Unit）和主存储器（Main Memory）。

（1）运算器

运算器是进行算术和逻辑运算的部件，它由完成加法运算的加法器、存放操作数和运算结果的寄存器和累加器等组成。

(2)控制器

它是整个计算机硬件系统的指挥中心，根据不同的指令产生不同的命令，指挥整个机器有条不紊地自动地进行工作。

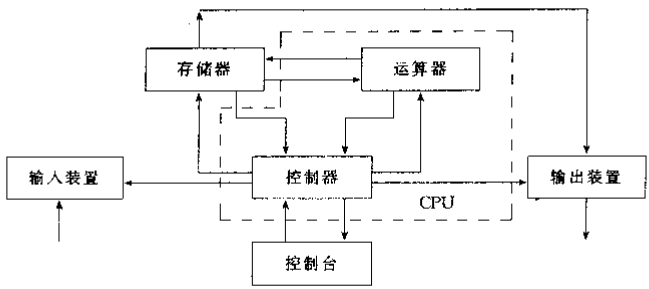


图1-2　电子计算机的组成框图

(3)主存储器

主存储器又称为内存储器，它由大量的存储单元组成，用以存储大量的数据及程序。目前的主存储器一般是由半导体电路组成，称为半导体存储器。

运算器和控制器又称中央处理机 CPU(Central Processing Unit)。

2. 外部设备

(1)输入设备(Input Device)

它将计算程序和原始数据转换为电信号，在控制器的控制下，按地址顺序地存入主存。过去，通常采用纸带输入机，目前常用的有键盘、软驱、磁带机、光驱。

(2)输出设备(Output Device)

它将运算的结果以人们容易识别的形式，在控制器的控制下，由主存通过外部设备呈现给人们。输出设备有显示器、打印机、绘图仪。

(3)外存储器(External Storage)

它是主存的后备存储器，存取速度比主存慢，但容量是海量。它不直接和 CPU 打交道，存储主机工作时暂时不用的那些程序和数据。常用的外存有磁带、磁盘、光盘，其中磁盘又可分为硬盘及软盘。

第四节　单片微型计算机的发展及应用

单片微型计算机简称单片机，它是把组成微型计算机的各部件：中央处理器、存储器、输入输出接口电路、定时器/计数器等，制作在一块集成电路芯片中，构成一个完整的微型计算机。

1971 年，Intel 公司首次推出 4004 的 4 位单片微处理器。1974 年 12 月仙童(Fairchild)公司推出 8 位单片机 F8(需另加一块 3851 芯片)，其后，Mostek 公司和仙童公司一起推出了与 F8 兼容的 3870 单片机系列。Intel 公司 1976 年推出 MCS-48 系列单片

机。GI(Gentral Instrument Crop)公司在1977年10月宣布了PIC1650单片机系列。1978年,Rockwell公司也推出了R6500/1系列(与6502兼容)。有些单片机都有8位CPU,若干个并行I/O,8位定时器/计数器,容量有限的RAM和ROM,以及简单中断处理功能。

Motorola公司和Zilog公司的单片机问世较迟,但是产品性能较高,单片机内有串行I/O,多级中断处理能力,片内的RAM和ROM容量较大,有些还带有A/D转换接口。Motorola公司在1978年下半年宣布了与6800微处理机兼容的6801单片机。Zilog公司在同年10月也推出了Z80单片机系列。Intel公司在原MCS-48基础上,于1980年推出了高性能的MCS-51系列(包括8031/8051/8751)。1982年,Mostek公司和Intel公司先后推出了16位单片机MK68200(与68000微处理器兼容)和MCS-96(8096、8098)系列。1987年Intel公司推出了性能是8096 2.5倍的新型单片机80296。

目前单片机的现状为:

(1)4位单片机

4位单片机的主要产品有:

NEC公司的μPD75xx;

TI公司的TMS1000系列;

松下公司的MN1400系列;

NS公司的COP400;

Rockwell公司的PPS/1系列;

SAMSUNG公司的KS56和KS57系列;

富士通公司的MB88系列。

其中,μPD75xx与COP400在4位机中占有重要地位,年产量已达到数千万片。

4位单片机的特点是价格便宜,如COP400的价格仅为8位单片机8048和6805价格的一半,但是功能并不弱,只是4位CPU,片内的ROM有2K,RAM为128×4位。NEC公司的μPD75xx片内的ROM可达8K字节,RAM为512×4位,I/O引脚为58根,甚至还有6位A/D。近年来,4位单片机的产量仍在增长,但所占比例逐年下降,单片机的主角已让给了8位单片机。4位机与8位机进行竞争,只有进一步降低价格,并增强I/O的功能(特别是专用I/O功能)。4位机主要用于家用电器和电子玩具等方面。

(2)8位单片机

8位单片机的产量占整个单片机的60%以上,并逐年增长。1985年的产量为1.7亿片,1986年的产量为2.1亿片,1992年达7亿片。8位单片机的旧的机种正在被淘汰,新的机型不断涌现。

自1985年以来,各种高性能、大容量、多功能的新型8位单片机不断地推出。如Intel公司的8x552、μPI-452(8051的增强型)、Motorola公司的MC68HC11(6801增强型)、Zilog公司的Super8等,它们将代表单片机发展的方向,将在单片机领域中起越来越大的作用。常用的8位单片机性能如表1-1所示。

表 1-1　主要 8 位单片机的性能表

公　司	系　列	片内存储器		寻址范围	片内 I/O		定时器/计数器	中断	备　注
		ROM	RAM		并行口	串行口			
Intel	MCS-48	1K/4K	64/256B	4KB	3×8 位	/	1×8 位	2	
	MCS-51	4K/8K	128/256B	64K	4×8 位	UART	2×16 位	5/6	
	8XC51FX	8/32K	256B	64K	4×8 位	UART	3×16 位	7	PCA
	8XC51GB	8K	256B	64K	6×8 位	2UART	3×16 位	15	PCA 8×8A/D
Motorola	6801	2K/4K	128/256B	64K	3×8 位 1×5 位	UART	3×16 位	2	
	6805	1K/4K	64B/112B	2K/8K	2×8 位 1×4 位	/	1×8 位	1/4	
	68HC11A	8K	256B	64K	22～ 38 位	1 SCI 1SPI	16 位,3-IC 5-OC, RTI	20	WDOG 8×8A/D
Zilog	Z8	2K/4K	124B	64K	8×1 位 4×4 位 1×8 位	UART	2×8 位	6	
Fairchild	F8	/	64K	4KB	2×8 位	/	/	/	
Mostek	3870	1K/4K	64B	4KB	4×8 位	/	1×8 位	2	
Reckwell	6500/1	256B/3K	64B/192B	64KB	4×8 位	UART	1×16 位	4/8	
NEC	UPD78XX	4K/6K	128/256B	64KB	6×8 位	UART	1×12 位	3	
TI	TMS7000	2K/12K	128B	64KB	4×8 位	UART	1/2×13 位	2/6	微程序
GI	PIC16XX	512×12/ 2K×12 位	32B/64B	512B/ 2KB	8×4 位	/	1×8 位	1/2	
NS	8070	2K/2.5K	64B/128B	64/128K	5×8 位	UART		/	
RCA	CDP1800	2K	64B	64KB	12 位	/	/	3	DMA
Philips	8XC552	8K	256B	64K	6×8 位	UART	3×16 位	15	8×10ADC 2×8PWM

注：IC：输入捕捉；OC：输出比较；PWM：脉宽调制；RTI：实时中断；SPI：串行外围接口；SCI：串行通信接口；PCA：可编程计数器阵列。

8 位单片机由于其功能强、品种多，正广泛应用于各个领域，是单片机的主流机种。随着集成电路工艺的不断改进，8 位单片机的价格也不断降低，甚至比 4 位单片机的价格低，如 Motorola 公司的 MC6804J1 价格已不足 0.5 美元。

(3)16 位单片机

16 位单片机自 1982 年开始推出，已有很大发展，但是它的增长没有人们预计的快。目前，16 位机的产量还不到 8 位机的十分之一。由于 16 位机价格高，应用还不广泛，主要应用于汽车控制、自动控制等方面。常用的 16 位单片机性能如表 1-2 所示。

表 1-2　16 位单片机性能表

公　司		Thomson	Intel	NS	NEC
型　号		68200	MCS-96	BPC16040	783××
片内	ROM	4KB	8KB	4KB	8KB
	RAM	256B	232B	256B	256B
中断源		15	8	8	15
串行口		异/同步	异　步	异　步	异　步
A/D		无	8×10 位	无	4×8 位
PWM 输出		借用通用计数器	有	有	有
Watchdog 定时器			有	有	有
计数器		3×16	2×16	8×16	2×16
高速 I/O		无	HSIO	有	有
DMA		无	无	无	8 个宏通道
备　注		与 68000 指令兼容			7811 升级产品

注：68200 原为 Mostek 公司产品，是第一个 16 位单片机。由于 Mostek 公司经营不景气。1985 年宣布倒闭，由 Thomson 公司接管该公司。

由于单片机超小型化，结构紧凑，可靠性高，价格低廉，在国民经济中得到广泛应用。

①工业方面：电机控制、工业机器人、过程控制、数字控制。

②仪器仪表方面：智能仪器、医疗器械、色谱仪、示波器。

③民用方面：电子玩具、高级电视游戏机、录像机、激光盘驱动。

④电讯方面：调制解调器、智能线路运行控制。

⑤导航与控制方面：导弹控制、鱼雷制导控制、智能武器装置、航天导航系统。

⑥数据处理方面：图形终端、彩色黑白复印机、温氏硬盘驱动器、磁带机、打印机。

⑦汽车方面：点火控制、变速器控制、防滑刹车、排气控制。

单片机的发展趋势是：增加存储器容量，片内 EPROM 开始 EEPROM 化，存储器编程保密化，片内 I/O 多功能化及低功耗 CMOS 化。

思考题与习题

1. 什么是单片机？它与一般微型计算机在结构上有什么区别？
2. 微型计算机由哪几部分组成？

第二章　计算机运算基础

本章主要介绍计数方法、数的表示法和运算方法。

在计数方法中，主要介绍各种进位计数制及不同进位计数制之间的转换。

在数的表示法中，主要介绍真值与机器数、定点与浮点数的表示方法、原码、补码、反码以及数的编码方法。

在运算方法中，主要介绍定点加减法补码运算及逻辑运算。

本章内容是必要的入门知识。对于已学过这些知识的读者，本章将起到复习和系统化的作用。

第一节　数　　制

一、进位计数制

按进位的原则进行计数的数制，称为进位计数制。

1. 十进制(Decimal System)

大约在公元 400 年左右，印度数学家首先发明了用十进制计数。约在公元 800 年，阿拉伯人开始使用，所以又称为阿拉伯数制。以后传到欧洲，才被命名为“十进制”。

十进制计数制中，是根据“逢十进一”的原则进行计数的。一个十进制数，它的数值是由数码 0、1、2、…、8、9 来表示的。数码所处的位置不同，代表数的大小也不同。从右起的第一位是个位，第二位是十位，第三位是百位、……。个、十、百、千等在数学上叫做“权”。十进制数的权是以 10 为底的幂。所使用的数码的个数称为基(如十进制数中为 10)。每一位上的数码与该位“权”的乘积表示了该位数值的大小。如下面的数：

十进制	5	2	3	8	9	4
	10^5	10^4	10^3	10^2	10^1	10^0
	十万	万	千	百	十	个

$$523894=5\times10^5+2\times10^4+3\times10^3+8\times10^2+9\times10^1+4\times10^0$$

“权”和“基”是进位计数制中的两个要素。

2. 二进制(Binary System)

二进制是按“逢二进一”的原则进行计数的。二进制数的基为“2”，即其使用的数码为 0、1，共 2 个。二进制数的权是以 2 为底的幂。如下面这个数：

二进制	1	1	0	1	1	1
	2^5	2^4	2^3	2^2	2^1	2^0
十进制	32	16	8	4	2	1

其各位的权为 1、2、4、8…，即以 2 为底的 0 次幂、1 次幂、2 次幂等。

$$(110111)_2=1\times2^5+1\times2^4+0\times2^3+1\times2^2+1\times2^1+1\times2^0=(55)_{10}$$

3. 八进制(Octave System)

八进制是按"逢八进一"的原则进行计数的。八进制数的基为"8"，它使用的数码有 8 个，即为 0、1、2、3、4、5、6、7。八进制的权是以 8 为底的幂。如下面这个数：

八进制	1	0	3	5	2	4
	8^5	8^4	8^3	8^2	8^1	8^0

$$(103524)_8=1\times8^5+0\times8^4+3\times8^3+5\times8^2+2\times8^1+4\times8^0=(34644)_{10}$$

4. 十六进制(Hexadecimal System)

十六进制是按"逢十六进一"的原则进行计数的。十六进制数的基为 16，即基数码共有 16 个：0、1、2、3、4、5、6、7、8、9、A、B、C、D、E、F。其中 A、B、C、D、E、F 分别代表值为十进制数中的 10、11、12、13、14、15。十六进制的权为以 16 为底的幂。如下面的数：

十六进制	4	A	0	7	F	1
	16^5	16^4	16^3	16^2	16^1	16^0

$$(4A07F1)_{16}=4\times16^5+10\times16^4+0\times16^3+7\times16^2+15\times16^1+1\times16^0=(4851679)_{10}$$

在数字后面加上(2)、(8)、(10)或(16)是指二进制、八进制、十进制数和十六进制数。也有用字母来表示这些数制的，B——二进制，H——十六进制，O——八进制，D——十进制。通常十进制数的 D 或 10 可以省略不写。

常用计数制表示数的方法如表 2-1。

表 2-1　常用计数制表示数的方法

十进制	二进制	八进制	十六进制
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7

续表 2-1

8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10

二、计算机中为什么要用二进制或十六进制计数

在计算机中采用什么数制，考虑因素之一是能用最少的状态表示的数最大，这样能使计算机的硬件结构最简单。

例如：4 位十进制数的最大数值为 9999。每位数可能为十个数码之一，若表示一位十进制数，则需要 10 个状态量，每一状态代表十个数码之一。4 位十进制数则需要 $4\times10=40$ 个状态量表示。若采用八进制数，每位有八个数码，需要 8 个状态量，40 个状态量则能表示八进制数 5 位，其最大数为 $8^5-1=32767$。若采用二进制数，每位有两个数码，40 个状态量能表示 20 位二进制数，其最大数为 $2^{20}-1=1048575$。显然同样用 40 个状态量，用二进制数表示的数值最大。也就是用二进制数表示一个数，所用的状态量最少，硬件设备最少。

那么从理论上，究竟用几进制的数最节省状态数呢？

设采用 x 进制数，位数为 n，$x\cdot n$ 为状态数。假设状态 $x\cdot n=$const(常数)，其表示的最大数为 x^n-1。

设 $f(x)=x^n-1$，显然 $f(x)$为最大时的 x 即为最节省状态的进制数。

若 $f(x)$为最大，则 $p(x)=f(x)+1=x^n$ 亦应为最大。

对 $p(x)=x^n=x^{\frac{c}{x}}$两边取对数，则

$$\ln p(x)=\ln x^{\frac{c}{x}}=\frac{c}{x}\ln x$$

两边对 x 求导，则

$$\frac{p'(x)}{p(x)}=\frac{\mathrm{d}[\frac{c}{x}\ln x]}{\mathrm{d}x} \qquad p'(x)=p(x)\cdot\frac{\mathrm{d}[\frac{c}{x}\ln x]}{\mathrm{d}x}$$

当 $p'(x)=0$ 时，$p(x)$取得最大值。由于 $p(x)\neq0$，则

$$\frac{\mathrm{d}[\frac{c}{x}\ln x]}{\mathrm{d}x}=0 \qquad \frac{c}{x}\cdot\frac{1}{x}-\frac{c}{x^2}\ln x=0$$

求得 $\ln x=1$，则 $x=\mathrm{e}=2.718$。

由于进制数必须为整数，则从理论上说，当用三进制数时所用的状态量最少，其次为二进制数。

电路中一般只用两种稳态：导通与阻塞，饱和与截止，高电位与低电位等，具有两个稳态的电路称为二值电路。用二值电路来计数时，只能代表两个数码：0 和 1，如用 1 代表高电位，则 0 代表低电位。采用二进制，就可以利用现有电路进行计数工作，并且所用的状态量较少。所以，计算机中采用二进制数计数是必然的。

用二进制表示一个数时位数较长，不容易记忆，如$(44421)_{10}$用二进制和十六进制分别表示为

$$(\underline{1010}\ \ \underline{1101}\ \ \underline{1000}\ \ \underline{0101})_2=(AD85)_{16}=(44421)_{10}$$

A D 8 5

显然用十六进制表示与用二进制表示相比，书写简短，便于记忆。这就是应用十六进制数的意义。

三、数制的转换方法

由于我们习惯十进制计数，所以在研究问题或讨论解题的过程时，总是用十进制来考虑和书写的。当考虑成熟后，要把问题变成计算机能够"认识"的形式，即把问题中的所有十进制数转换成二进制代码，因此需要用到"十进制转换成二进制数的方法"。计算机运算完毕得到二进制结果后，又需要用到"二进制数转换为十进制数的方法"，才能把运算结果用十进制形式显示出来。二进制数书写较烦，经常用十六进制表示，这又需要二进制与十六进制数之间的转换。

1. 十进制数转换成二进制数的方法

十进制整数转换成二进制整数，通常采用"除二取余法"。所谓的"除二取余法"，就是用 2 去除该十进制数，得商和余数，此余数为二进制代码的最小有效位(LSB)或最低位的值；再用 2 除该商数，又可得商数和余数，则此余数为 LSB 左邻的二进制代码(次低位)。以此类推，从低位到高位逐次进行，直到商是 0 为止，就可得到该十进制数的二进制代码。

例：将$(53)_{10}$转换成二进制数，其过程如下：

```
2 | 53   ——余数为1   低位
  | 26   ——余数为0     |
  | 13   ——余数为1     |
  |  6   ——余数为0     |
  |  3   ——余数为1     |
  |  1   ——余数为1     ↓ 高位
  |  0
```

所以 $$(53)_{10}=(110101)_2$$

十进制纯小数转换成二进制纯小数，通常采用"乘二取整法"。所谓"乘二取整法"，就是将已知十进制的小数乘以 2 之后，可能有进位，使整数位为 1(当该小数大于 0.5 时)，

也可能没有进位，其整数位仍为零。该整数位的值为二进制小数的最高位，再将乘积的小数部分乘以2，所得整数位的值为二进制小数的次高位。以此类推，直到满足精度要求或乘2后的小数部分为0为止。

例：求十进制数0.625的二进制数。

其进行步骤可以用乘法的竖式算法：

```
    0. 6 2 5
×)        2
-----------
    1. 2 5          整数部为1即二进制小数后第一位为1
    0. 2 5
×)        2
-----------
    0. 5            整数部为0即二进制小数后第二位为0
×)      2
-----------
    1. 0            整数部为1即二进制小数后第三位为1
```

所以 $(0.625)_{10}=(0.101)_2$

如果小数位不是0，则还得继续乘下去，直至变成0为止或者满足精度要求。因此一个十进制小数在转换为二进制小数时，有可能无法准确地转换。如十进制数0.1转换为二进制数时为0.0001100110…，因此只能近似以0.00011001来表示。

我们把既有整数部又有小数部复合而成的小数称为混合小数。十进制混合小数转换为二进制数，只要按上述方法将整数及小数部分分别进行转换，然后将转换结果组合起来即可。

2. 二进制数转换为十进制数的方法

将二进制数转换成十进制数，只要将二进制数各位的权乘以各位的数码(0或1)再相加即可。例如：

$$(11001.1001)_2=1\times2^4+1\times2^3+0\times2^2+0\times2^1+1\times2^0+1\times2^{-1}+0\times2^{-2}+0\times2^{-3}+1\times2^{-4}$$
$$=16+8+1+0.5+0.00625=(25.5625)_{10}$$

3. 二进制与十六进制数之间的转换方法

由于$2^4=16$，可见每4位二进制数与一位十六进制数相对应。这样就使二进制与十六进制数之间的转换较为简单。

(1)二进制数转换成十六进制数

对二进制整数，只要自右向左将每4位二进制数分为一组，不足4位时，在左面添0，补足4位；对二进制小数，只要从小数点后自左向右将每4位二进制数分为一组，不足4位时，在右面添0，补足4位，然后将每组二进制数用相应的十六进制数代替，即可完成转换。

例：把$(101101101.0100101)_2$转换成十六进制数，则

$$(0001\ \ 0110\ \ 1101.0100\ \ 1010)_2$$
$$\downarrow\quad\downarrow\quad\downarrow\quad\downarrow\quad\downarrow$$
$$(1\quad 6\quad D\ .\ 4\quad A)_{16}$$

所以 $(101101101.0100101)_2=(16D.4A)_{16}$

(2)十六进制数转换成二进制数

如将十六进制数转换成二进制数，只要将每1位十六进制数用4位相应的二进制数表示即可完成转换。

例：将$(1863.5B)_{16}$转换成二进制数，则

$$(\ 1\quad 8\quad 6\quad 3\ .\ 5\quad B\ )_{16}$$
$$(0001\ 1000\ 0110\ 0011\ .\ 0101\ \ 1011)_2$$

所以 $(1863.5B)_{16}=(1100001100011.01011011)_2$

第二节　数的表示方法

一、真值与机器数

设 $N_1=+1001010$

$N_2=-1001010$

N_1和N_2在机器中表示为

N_1:01001010

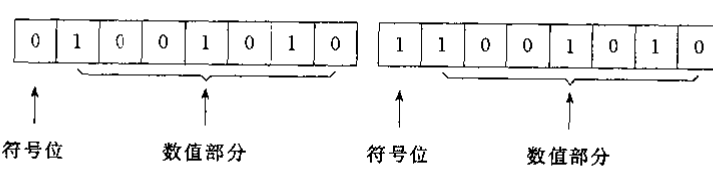
N_2:11001010

这就是说，数的符号在机器中数码化了，正数的符号用0表示，负数的符号用1表示。

一个数在机器中的表示形式，称为机器数；而把这个数本身，即用“+”“-”号表示的数称为真值。

可见$N_1=+1001010$和$N_2=-1001010$为真值，其在机器中的表示形式01001010和11001010为机器数。

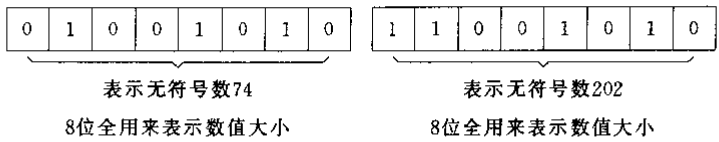
上面提到的机器数表示方法，用0表示正数的符号，用1表示负数的符号。这种表示数的方法，称为带符号数的表示方法。在机器中的表示形式为



前者表示正74，后者表示负74。

在计算机中还有一种数的表示方法，机器的全部有效位均用来表示数的大小，此时无

符号位，这种表示方法称为无符号数的表示方法。无符号数相当于数的绝对值大小。上例机器数的表示方法，若看作无符号数，则为



二、数的定点和浮点表示方法

在计算机中，数有两种表示方法，即定点表示法和浮点表示法。所谓定点表示法，就是小数点在数中的位置是固定不变的；所谓浮点表示法，就是小数点在数中的位置是浮动的。

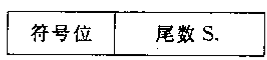
1. 定点表示法

通常，对于任意一个二进制数总可以表示为正整数（或纯小数）和一个 2 的整数次幂的乘积。例如，二进制数 N 可写成

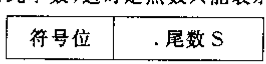
$$N=2^P\times S$$

其中，S 称为数 N 的尾数，P 称为数 N 的阶码。此处 P、S 都是用二进制表示的数。尾数 S 表示了数 N 的全部有效数字，阶码 P 指明了小数点的位置。

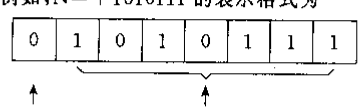
如假定 P=0，且尾数 S 为纯整数，这时定点数只能表示整数，即



如假定 P=0，且尾数 S 为纯小数，这时定点数只能表示小数，即



定点数的这两种表示方法，在计算机中均有采用。但是对一台具体机器而言，采用哪种方法，均是事先约定。例如，N=+1010111 的表示格式为



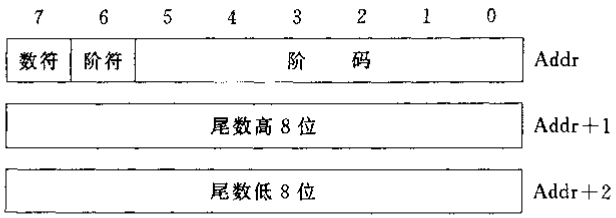
2. 浮点表示法

如果阶码是个可变的数值，称这种表示方法为数的浮点表示法。这样的数称为浮点数。设

$$N=2^P\times S$$

其中，阶码 P 为整数，可为正数，也可为负数；用 P_f 表示阶码的符号位，当 $P_f=0$ 时，表示阶码为正，$P_f=1$ 时表示阶码为负。同样，尾数 S 用 1 位二进制数 S_f 表示尾数的符号。

浮点数在机器中的表示方法如下：



三、原码、补码、反码

原码、补码、反码是带符号数的机器数的表示方法。

1. 模的概念

我们把一个计量器的容量称为模或模数，记为模 M 或 mod M。可见一个 n 位二进制寄存器，它的容量为 2^n。所以，它的模为 $M=2^n$。模具有这样的性质，当模为 2^n 时，2^n 和 0 在机器中表示方法是相同的。例如，当 $n=4$ 时，计数器的模为 $2^4=16$，它可以从 0 到 $2^4-1=(1111)_2=(15)_{10}$ 计数。当已经计到 1111 时，再加 1，则应为 10000，但此时机器中变成了 0000，即当模 $M=2^4$ 时，2^4 和 0 在这个计数器中表示方法相同。

2. 原码表示法

前面介绍的带符号数在机器中的表示方法，实际上就是原码表示法。

原码表示法是最简单的一种机器数表示法，只要把真值的符号部分用 0 或 1 表示即可。例如：

$$N_1=+1001010$$

$$N_2=-1001010$$

其原码记为

$$[N_1]_{原}=[+1001010]_{原}=01001010$$

$$[N_2]_{原}=[-1001010]_{原}=11001010$$

0 的原码有两种表示形式，即 +0 和 −0：

$$[+0]_{原}=00000000$$

$$[-0]_{原}=10000000$$

3. 反码表示法

反码是二进制数的另一种表示形式，正数的反码与原码相同；负数的反码是将其原码除符号位外，按位求反，即原来为 1 变为 0，原来为 0 变为 1。例如：

$$X_1=+1010011 \quad X_2=-1010011$$

则

$$[X_1]_{反}=[+1010011]_{反}=01010011$$

$$[X_2]_{反}=[-1010011]_{反}=10101100$$

从定义可知，0 的反码也有两种形式：

$$[+0]_{反}=00000000$$

$$[-0]_{反}=11111111$$

4. 补码表示法

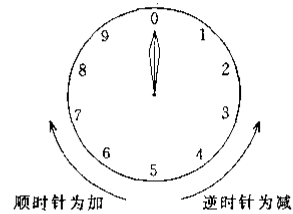


图2-1 数字拨盘

如图 2-1,把 0～9 十个数字刻在一个圆盘上,用来作加、减法。例如 A+B,它是先将指针转到 A 的位置,然后根据 B 的正负将指针在数字拨盘上按顺时针或逆时针转动|B|格,这时在数字拨盘上指针所指的数字就是运算结果。例如:

A=5　B=3　则 A+B=5+3=8

A=5　B=−3　则 A+B=5−3=2

但是,如果在 B=−3 时,不是逆时针转动 3 格而顺时针转动 7 格,即 A+B′=5+7=12,指针指到的数字仍然是 2,与逆时针转动看到的结果完全一样。这是因为数字拨盘的模为 10,进位 10 被丢掉了。我们称 B=−3 与 B′=7 对模 10 同余。B′称为 B 对模 10 的补数或补码。引入补码以后就可以把减运算变成加法运算。在计算机中,减法运算通常用它的补码的加法运算来代替,因为这样可以使计算机的硬件电路简单化。

设 B 为 -负数,其模为 M,则 B 的补码为 M+B。如上面 B=−3,模为 10,则−3 的补码为 10+(−3)=7。

在二进制中,通常以 2^n 为模,这样其补码表达式为

$$[X]_{补}=2^n+X \quad (模\ 2^n——n\ 位二进制数)$$

从上式可以看出,当 X 为正数时,$[X]_{补}$就是 X 本身;当 X 为负数时,从 2^n 中减去|X|,便可得到$[X]_{补}$;当 X 等于 0 时,$[X]_{补}$为 0。显然,0 的补码是唯一的。

例:$X_1=+1010011$,$X_2=-1010011$,求其补码。(n=8)

$$[X_1]_{补}=[+1010011]_{补}=2^8+1010011=01010011$$

$$[X_2]_{补}=[-1010011]_{补}=2^8-1010011=10101101$$

上例中$[X_2]_{补}$也可这样求得

$$[X_2]_{补}=[-1010011]_{补}=2^8-1010011=(11111111+1)-1010011$$

$$=(11111111-1010011)+1=[X_2]_{反}+1$$

即

$$[X]_{补}=[X]_{反}+1$$

补码的求法只要使其符号位不变,将数值部分逐位变反,末位加 1 即可。可见,反码通常是作为补码求法过程的中间形式。通过反码的概念求取补码更为直观、简单。

正数的原码、补码表示方法相同,不存在转换问题。下面讨论负数情况。

(1)已知$[X]_{原}$,求$[X]_{补}$

例如：

$$[X]_{原}=10011010$$

$$\downarrow\downarrow\downarrow\downarrow\downarrow\downarrow\downarrow\downarrow$$

$$11100101$$

$$+) \qquad\qquad 1$$

$$[X]_{补}=11100110$$

注意：求反时符号位不变。

(2)已知$[X]_{补}$，求$[X]_{原}$

我们有 $[[X]_{补}]_{补}=[X]_{原}$

例如：

$$[X]_{补}=1110110$$

$$\downarrow\downarrow\downarrow\downarrow\downarrow\downarrow\downarrow$$

$$1001001$$

$$+) \qquad\qquad 1$$

$$[X]_{原}=1001010$$

(3)求补：已知$[X]_{补}$，求$[-X]_{补}$

所谓求补，就是将$[X]_{补}$连同符号位一起逐位求反，然后在末位加1，即得到$[-X]_{补}$。应注意的是，不管$[X]_{补}$是正数还是负数，都应按上述方法判别。

例如：

$$[X]_{补}=01010110$$

$$\downarrow\downarrow\downarrow\downarrow\downarrow\downarrow\downarrow\downarrow$$

$$10101001$$

$$+) \qquad\qquad 1$$

$$[-X]_{补}=10101010$$

又如：

$$[X]_{补}=10101110$$

$$\downarrow\downarrow\downarrow\downarrow\downarrow\downarrow\downarrow\downarrow$$

$$01010001$$

$$+) \qquad\qquad 1$$

$$[-X]_{补}=01010010$$

求补在进行补码的减法运算时会用到。

表2-2列出了8位二进制码表示与无符号数、原码、补码和反码的相应关系。

由表2-2可见，用8位二进制数码表示无符号数范围是0～255；表示原码为－127～＋127；表示补码为－128～＋127；表示反码为－127～＋127。

5.常用编码

计算机只能识别0和1两种符号，而计算机处理的信息却有多种形式。如：数字、标点符号、运算符号、各种命令及各种文字、图形等。要表示这么多的信息并识别它们，就必须对这些信息进行编码。换句话说，以上这众多的信息只有按特定的规则用二进制编码后才能在机器中表示和识别。比如我们在操作键盘时，敲击的是字母、数字或功能符号，可键入机器内部的却是相应的一组二进制编码。

计算机中根据信息对象不同，编码的方式即码制也不同。常见的码制有二-十进制(BCD)码、ASCII码等等。

表 2-2 数的表示方法小结

八位二进制数码表示	无符号数	原码	补码	反码
00000000	0	+0	+0	+0
00000001	1	+1	+1	+1
00000010	2	+2	+2	+2
⋮	⋮	⋮	⋮	⋮
01111101	125	+125	+125	+125
01111110	126	+126	+126	+126
01111111	127	+127	+127	+127
10000000	128	−0	−128	−127
10000001	129	−1	−127	−126
10000010	130	−2	−126	−125
⋮	⋮	⋮	⋮	⋮
11111101	253	−125	−3	−2
11111110	254	−126	−2	−1
11111111	255	−127	−1	−0

(1)二-十进制(BCD)码

人们通常使用十进制来计数，而计算机则采用二进制，所以必须对十进制的 0～9 这十个数字进行二进制编码。但从二进制数直接看出所对应的十进制数是比较困难的。虽然我们学过二-十进制的转换，但这种转换比较慢，这是一个明显的缺点。为此，人们提出了一个比较适合十进制的二进制码的特殊形式，即二-十进制码，通常用英文字母 BCD 表示，BCD 码具有二进制和十进制两种数制的某些特征。

BCD 编码用 4 位二进制码表示 0～9 的十进制数。它采用了标准的 8421 的纯二进制码的十六个状态，其中只有 0000～1001 十个码有效，其余 1010～1111 没有使用。表 2-3 列出了标准的 8421 BCD 码对应的十进制数。

用 BCD 码表示十进制数，只要把每位十进制数用适当的二进制 4 位码代替即可。例如，十进制数 834 用 BCD 表示，则为 1000 0011 0100。为了避免 BCD 格式的码与纯二进制码混淆，必须在每 4 位之间留一空格。这种表示方法也适用于十进制小数。例如，十进制小数 0.764 用 BCD 码表示为 0.0111 0110 0100。

BCD 码的优点是与十进制数转换方便，容易阅读；缺点是用 BCD 码表示的十进制数的数位要较纯二进制表示的十进制数位更长，使电路复杂性增加，运算速度减慢些。

当希望计算机直接用十进制数进行运算时，应将数用 BCD 码来储存和运算。例如 4+3，应是 0100+0011=0111。但若是改为 4+8，直接运算结果为 0100+1000=1100，但从 BCD 数的运算来说应为 0001 0010，亦即十进制数 12。因此，在这种情况下，就要对二进制运算结果(1100)进行调整，使之符合十进制数的运算和进位规律。这种调整称为十进制调整，其内容有两条：

①若两个 BCD 数相加结果大于 1001，亦即大于十进制数 9，则应作加 0110(即加 6)调整；

②若两个 BCD 数相加结果在本位上并不大于 1001，但却产生了进位，相当于十进制

运算大于等于16，则也要作加0110(加6)调整。

如上面提到的4+8，直接运算结果为0100+1000=1100，作加6调整后所得结果为1100+0110=0001 0010，亦即十进制数12，结果正确。因此，BCD数运算一定要作十进制调整。

表2-3 BCD编码

十进制	8421 BCD	二进制
0	0000	0000
1	0001	0001
2	0010	0010
3	0011	0011
4	0100	0100
5	0101	0101
6	0110	0110
7	0111	0111
8	1000	1000
9	1001	1001
10	0001 0000	1010
11	0001 0001	1011
12	0001 0010	1100
13	0001 0011	1101
14	0001 0100	1110
15	0001 0101	1111

例：用BCD数完成54+48的运算。

```
    0101 0100
+)  0100 1000
-------------
    1001 1100
+)       0110        加6调整
-------------
    1010 0010
+)  0110             高4位加6调整
-------------
0001 0000 0010
```

此时，低4位之和为1100，应作加6调整，调整后高4位又为1010，还应作一次加6调整，故最后结果为0001 0000 0010，即102。

若是两个BCD数相减，则也要进行十进制调整，其规律是：当相减时，若低4位向高4位有借位，在低4位就要作减0110(减6)调整。

(2)字母数字代码——ASCII码及国内通用字符编码

计算机系统中，除了数字0～9以外，还经常用到其它各种字符，如26个英文字母A～Z、各种标点符号、控制符号等等。这些信息都要将它们编成计算机接受的二进制码。

ASCII(American Standard Code for Information Interchange)代码是美国的一个标准,全称为"美国信息交换标准代码",简称为 ASCII 码。美国的这个标准制订于 1963 年,后来国际标准化组织 ISO 和国际电报电话咨询委员会 CCITT 以它为基础制订了相应的国际标准。现在微机的字符编码都采用 ASCII 码。通常称作 ASCII 的美国信息交换标准代码是二进制编码的一种特殊形式,在微处理机外部设备(CRT 显示器、键盘、终端等等)和通讯设备数据表示中广泛使用。

ASCII 码是一种 8 位代码,最高位一般用于奇偶校验,用 7 位代码来对 128 个字符编码,其中 32 个是控制字符,96 个是图形字符。

表 2-4 为 7 位 ASCII 码字符表,表中最高位未列出,一般表示时都以 0 来代替,而暂不考虑其奇偶校验功能。如数字 0～9 的 ASCII 表示为十六进制数 30H～39H,字母 A～Z 的 ASCII 码为 41H～5AH。

表示数字、字母或控制功能的 7 位 ASCII 代码是由 3 位和 4 位一组组成的。图 2-2 表示这两组的安排和号码的顺序。4 位一组在右边,而且位 1 是最低位。要注意这些组在表 2-4 行、列中的排列情况。4 位一组表示行,3 位一组表示列。

表 2-4 美国信息交换标准代码 ASCII(7 位代码)

列	0③	1③	2③	3	4	5	6	7③
行 / 位 765→ / ↓4321	000	001	010	011	100	101	110	111
0	0000	NUL	DLE	SP	0	@	P	`	p
1	0001	SOH	DC1	!	1	A	Q	a	q
2	0010	STX	DC2	"	2	B	R	b	r
3	0011	ETX	DC3	#	3	C	S	c	s
4	0100	EOT	DC4	$	4	D	T	d	t
5	0101	ENQ	NAK	%	5	E	U	e	u
6	0110	ACK	SYN	&	6	F	V	f	v
7	0111	BEL	ETB	'	7	G	W	g	w
8	1000	BS	CAN	(	8	H	X	h	x
9	1001	HT	EM	)	9	I	Y	i	y
10	1010	LF	SUB	*	:	J	Z	j	z
11	1011	VT	ESC	+	;	K	[	k	{
12	1100	FF	FS	,	<	L	\	l	|
13	1101	CR	GS	-	=	M	]	m	}
14	1110	SO	RS	.	>	N	↑①	n	~
15	1111	SI	US	/	?	O	↓②	o	DEL

注:①取决于使用这种代码的机器,它的符号可以是弯曲符号、向上箭头、或者(一)标记。

②取决于使用这种代码的机器，它的符号可以是下划线、向下箭头或心形标记。

③是第 0、1、2 和 7 列特殊控制功能的解释：

NUL	空白	DLE	转意
SOH	序始	DC1	设备控制 1
STX	文始	DC2	设备控制 2
ETX	文终	DC3	设备控制 3
EOT	送毕	DC4	设备控制 4
ENQ	询问	NAK	否认
ACK	承认	SYN	同步
BEL	告警	ETB	组终
BS	退一格	CAN	作废
HT	横表	EM	载终
LF	换行	SUB	取代
VT	纵表	ESC	扩展
FF	换页	FS	卷隙
CR	回车	GS	群隙
SO	移出	RS	录隙
SI	移入	US	元隙
SP	空格	DEL	作废

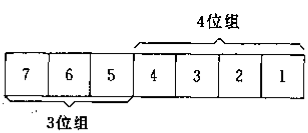


图2-2 ASCII代码格式

要确定某数字、字母或控制操作码的 ASCII 码，在表中可查到，然后根据该项的位置从相应的行和列中找出 3 位和 4 位的码，这就是所需的 ASCII 码。例如，字母 L 的 ASCII 代码是 1001100，它在表的第 4 列第 12 行，最大的 3 位是 100，最低的 4 位是 1100。

在 7 位的 ASCII 码中，第 8 位常用作奇偶校验位，以确定数据传送的是否正确。该位的数值由所要求的奇偶类型确定。偶数奇偶校验位是指每个代码中所有 1 位的和(包括奇偶校验位)总是偶数。例如，传递的字母是 G，则 ASCII 代码是 1000111，因其中有四个 1，所以奇偶校验位是 0，8 位代码将是 01000111。奇数奇偶校验位是指每个代码中所有 1 位的和(包括奇偶校验位)是奇数，若用奇数奇偶校验传送 ASCII 代码中的 G，其二进制的表示应为 11000111。

以下表格为各种 ASCII 字符的代码名称和功能解释等。

表 2-5 传输控制字符

字 符	代 码	名 称	功 能
SOH	01	标题开始	文件标题的开始
STX	02	正文开始	正文的开始,文件标题的结束
ETX	03	正文结束	正文的结束
EOT	04	传输结束	一次传输结束
ENQ	05	询问	向已建立联系的站请求回答
ACK	06	应答	对已建立联系的站作肯定回答
DLE	10	数据链转意	使紧随其后的有限个字符或代码改变含义
NAK	15	否认	对已建立联系的站作否定回答
SYN	16	同步空转	用于同步传输系统的收发同步
ETB	17	组传输结束	一组数据传输的结束

表 2-6 格式控制字符

字 符	代 码	名 称	功 能
BS	08	退格	使打印或显示位置在同一行中退回一格
HT	09	横向制表	使打印或显示位置在同一行内进至下一组预定格位
LF	0A	换行	使打印或显示位置换到下一行同一格位
VT	0B	纵向制表	使打印或显示位置在同一列内进至下一组预定行
FF	0C	换页	使打印或显示位置进至下一页第一行第一格
CR	0D	回车	使打印或显示位置回至同一行的第一格位

表 2-7 信息分隔控制字符

字 符	代 码	名 称	功 能
US	1F	单元分隔符	用于逻辑上分隔数据单元
RS	1E	记录分隔符	用于逻辑上分隔数据记录
GS	1D	群分隔符	用于逻辑上分隔数据群
FS	1C	文件分隔符	用于逻辑上分隔数据文件

表 2-8 设备控制字符

字 符	代 码	名 称	功 能
DC1	11	设备控制符 1	使辅助设备接通或启动
DC2	12	设备控制符 2	使辅助设备接通或启动
DC3	13	设备控制符 3	使辅助设备断开或停止
DC4	14	设备控制符 4	使辅助设备断、停止或中断

表 2-9 其它控制字符

字 符	代 码	名 称	功 能
NUL	00	空白符	在字符串中插入或去掉空白符，字符串含义不变
BEL	07	告警符始	控制警铃
SO	0E	移出符	使此字符后的各字符改变含义
SI	0F	移入符	由 SO 字符开始的字符转意到此结束
CAN	18	作废符	表明数据或字符是错误的或可略去
SP	20	空格符	使打印或显示位置前进一格
EM	19	媒体尽头	用于识别物理媒体的物理末端
SUB	1A	取代符	用于取代无效或错误的字符
ESC	1B	换码符	
DEL	7F	作废符	清除错误的或不要的字符

我国于 1980 年制订了"信息处理交换用的 7 位编码字符集"，即国家标准 GB1988-80。除用人民币符号¥代替美元符号 $ 以外，其余代码与含义均与 ASCII 码相同。

第三节 数的运算方法

一、基本运算

由于二进制数只有两个字符，因此运算规则非常简单。

加法：0+0=0

0+1=1+0=1

1+1=10

例：1101+1001=10110

```
  1101
+)1001
-----
10110
```

乘法：0×0=0

0×1=0

1×1=1

例：1101−0111=0110

```
  1101
-)0100
-----
0110
```

例：1101×110=1001110

```
  1101
×)  110
------
  0000
 1101
1101
-------
1001110
```

例：11011÷101=101 余 10

```
     101
101 )11011
    101
    ---
     111
     101
     ---
      10
```

二、补码的加减法运算

补码加减法运算是带符号数加法运算的一种。其运算特点是：符号位与数值部分一起参加运算，并且自动获得结果(包括符号和数值部分)。

1. 补码的加法运算

在进行补码的加法时，不管两个数是正数还是负数，按补码的和等于和的补码进行。

因为 $[X]_{补}+[Y]_{补}=2^n+X+2^n+Y=2^n+(X+Y)$

$=[X+Y]_{补}$ $\quad(\mathrm{mod}\ 2^n)$

所以 $[X]_{补}+[Y]_{补}=[X+Y]_{补}$

例：X＝＋10010，Y＝－01111，则

$$\begin{array}{rl} [X]_{补} & =0\ 10010 \\ +)[Y]_{补} & =1\ 10001 \\ \hline [X+Y]_{补} & =\boxed{1}\ 00011 \end{array}$$

符号位的进位，丢掉

2. 补码的减法运算

在进行减法时，按补码的差等于差的补码进行。

因为 $[X]_{补}-[Y]_{补}=[X]_{补}+[-Y]_{补}=2^n+X+2^n+(-Y)$

$=2^n+(X-Y)=[X-Y]_{补}$

所以 $[X]_{补}-[Y]_{补}=[X]_{补}+[-Y]_{补}=[X-Y]_{补}$

例：X＝－0111000，Y＝－0010001，则

$[X]_{补}=11001000$

$[Y]_{补}=11101111$ $\qquad [-Y]_{补}=00010001$

$$\begin{array}{rl} & [X]_{补}=11001000 \\ +) & [-Y]_{补}=00010001 \\ \hline & [X]_{补}+[-Y]_{补}=11011001 \end{array}$$

$[X-Y]_{补}=[X]_{补}+[-Y]_{补}=11011001$

三、定点乘法运算

实现定点乘法运算，就是确定乘积的符号和乘积的数值。

两个用原码表示的数相乘时，乘积的符号按正正得正，负负得正，正负得负，负正得负的原则进行。即同号相乘，乘积为正；异号相乘，乘积为负。这正好可用异或运算实现。如两个数 A 和 B，其符号位分别为 A_f 和 B_f，则乘积符号＝$A_f \oplus B_f$。

乘积的数值就是两数尾数之积。

例如，有两个无符号数 A＝1011，B＝1101，其乘积为

```
      1011        被乘数
×)    1101        乘数
-----------
      1011
     0000         部分积
    1011
   1011
-----------
  10001111        乘积
```

可见，两个 n 位的无符号数相乘，乘积的位数为 2n 位；乘积等于各部分积之和。由乘数从低位到高位逐位去乘被乘数，当乘数相应位为 1 时，则该位部分积等于被乘数；当乘数的相应位为 0 时，部分积为 0。

四、逻辑运算(布尔代数)

布尔代数也称逻辑代数。同普通代数一样，可以写成下面这样的表达式：

$$Y=f(A,B,C,D)$$

但它有两个特点：

第一，其中的变量 A、B、C、D 等均只有两种可能的数值：0 或 1。即布尔代数变量的数值并无大小之意，只是代表事物的两个不同性质。

如用于开关上，则 0 代表关或低电位，1 代表开或高电位。

如用于逻辑推理，则 0 代表错误(伪)，1 代表正确(真)。

第二，函数 f 只有三种基本方式："或"运算、"与"运算及"非"运算。由这三种运算可以导出其它的逻辑运算，如异或运算、同或运算、与或非运算等等。这里我们只介绍四种逻辑运算：与运算、或运算、非运算和异或运算。

1. 与运算($Y=A\cdot B$)

与运算也称为逻辑乘法运算，通常用符号"·"或"∧"或"×"表示。它的运算规则为

$$\left.\begin{array}{l}0\cdot 0=0\\1\cdot 0=0\\0\cdot 1=0\end{array}\right\}\ Y=0$$

$$1\cdot 1=1 \quad Y=1$$

这种运算的结果也可以归纳成为两句话：二者皆为真者结果为真，有一伪者结果必伪。同样，这个结论也可推广至多变量：各变量均为真者结果必真，有一伪者结果必伪。

与运算有时称为"逻辑与"。当 A 和 B 为多位二进制数时，则进行"逻辑与"运算时，各对应位分别进行"与"运算。

例：设 A=11001010，B=00001111，则 Y=A·B，写成竖式则为

```
    11001010
×)  00001111
------------
    00001010
```

Y= 00001010

由此可见，用“0”去和一个数位相“与”，就将其“抹掉”而成为“0”，用“1”去和一个数位相“与”，就是将此数位“保存”下来。这种方法在计算机的程序设计中经常会用到，称为“屏蔽”。例中B数(00001111)则称为“屏蔽字”，它将A数的高4位给屏蔽起来而都变成0了。

2. 或运算(Y=A+B)

“或”运算也称逻辑加法，常用符号“+”或“∨”表示。它的运算规则为

$$\left.\begin{array}{l} Y=0+0=0 \\ Y=0+1=1 \\ Y=1+0=1 \\ Y=1+1=1 \end{array}\right\} \begin{array}{l} Y=0 \\ Y=1 \end{array}$$

第四个式子与一般加法不同，这是因为Y也是只能有两种数值0或1。上面四个式子可以归纳成两句话：两者皆伪者结果必伪，有一为真者则结果为真。这个结论也可推广至多变量：各变量全伪者则结果必伪，有一真者结果必真。

或运算有时也称为“逻辑或”。当A或B为多位二进制数时，则进行或运算时，各对应位分别进行“或”运算。

例：设A=10101，B=11011，则写成竖式为

$$\begin{array}{rr} & 10101 \\ +) & 11011 \\ \hline & 11111 \end{array}$$

所以　　Y=A+B=11111

注意：1“或”1等于1，是没有进位的。

3. 反运算($Y=\overline{A}$)

反运算又称非运算，逻辑否定。如果一事物的性质为A，则其经过“反”运算之后，其性质必与A相反，用表达式表示为：$Y=\overline{A}$。其运算规则为

$\overline{0}=1$　读成非0等于1

$\overline{1}=0$　读成非1等于0

当A为多位数时，如$A=A_1A_2A_3\cdots A_n$，则其“逻辑反”为$Y=\overline{A}_1\overline{A}_2\overline{A}_3\cdots\overline{A}_n$。

例：设A=11010000，则

$Y=\overline{A}=00101111$

4. 异或运算($Y=A\oplus B$)

异或运算通常用符号“⊕”表示。它的运算规则为

$$\left.\begin{array}{l} Y=0\oplus 0=0 \\ Y=0\oplus 1=1 \\ Y=1\oplus 0=1 \\ Y=1\oplus 1=0 \end{array}\right\} \begin{array}{l} Y=0 \\ Y=1 \\ Y=0 \end{array}$$

同样异或运算结果也可归纳成两句话：只要两逻辑变量相同，则异或运算的结果就为0；当两个逻辑变量不同时，异或运算的结果才为1。

当A或B为二进制数时，则进行异或运算时，各对应位1分别进行异或运算。

例：设 A＝1010，B＝1101，写成竖式，则

$$\begin{array}{rr} & 1010 \\ \oplus) & 1101 \\ \hline & 0111 \end{array}$$

$$Y=A\oplus B=0111$$

5. 布尔代数的基本运算规律

①恒等式　$A\cdot 0=0$　　$A\cdot 1=A$　　$A\cdot A=A$

$A+0=A$　　$A+1=1$　　$A+A=A$

$A+\overline{A}=1$　　$A\cdot\overline{A}=0$　　$\overline{\overline{A}}=A$

②运算规律

与普通代数一样，布尔代数也有交换律、结合律、分配律，而且它们与普通代数的规律完全相同。

交换律　$A\cdot B=B\cdot A$

$A+B=B+A$

结合律　$(A\cdot B)\cdot C=A\cdot(B\cdot C)=A\cdot B\cdot C$

$(A+B)+C=A+(B+C)=A+B+C$

分配律　$A\cdot(B+C)=A\cdot B+A\cdot C$

$(A+B)\cdot(C+D)=A\cdot C+A\cdot D+B\cdot C+B\cdot D$

利用这些运算规律及恒等式，就可以化简很多逻辑关系式。

例：　$A+AB=A\cdot(1+B)=A$

$A+\overline{A}\cdot B=A+AB+\overline{A}B=A+(A+\overline{A})\cdot B=A+B$

6. 摩根定理

在电路设计时，有时人们手边没有“与”门，而只有“或”门和“非”门，或者只有“与”门和“非”门，没有“或”门。利用摩根定理，可以帮助你解决元件互换问题。

摩根定理式为：　$\overline{(A+B)}=\overline{A}\cdot\overline{B}$

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

推广到多变量：　$\overline{A+B+C+\cdots}=\overline{A}\cdot\overline{B}\cdot\overline{C}\cdot\cdots$

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

这个定理可以用一句话来记忆使用：“头上切一刀，下面变个号。”

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

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

第四节　二进制数加法电路

作为算术的基本运算，众所周知，有四种运算：加、减、乘和除。在微型计算机中，经常只有加法电路，这是为了使硬件结构简单，同时成本较低。利用加法电路，也能完成算术的四则运算。

一、基本门电路

在布尔代数中，函数有三种基本方式：与运算、或运算、反运算，与之相对应，在逻辑电路中也有三种基本门电路（或称判定元素）。图 2-3 是这几个门电路的名称、符号及表达式。

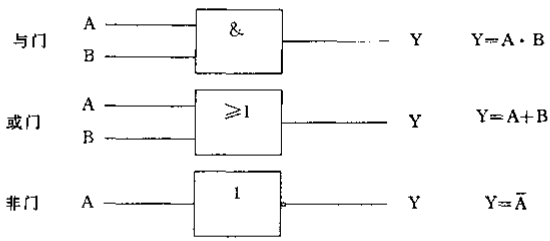


图2-3　三个基本门电路

在三个基本门电路的基础上，还可发展成如图 2-4 所示更为复杂的逻辑电路。最后一个，叫做缓冲器（Buffer），实际是两个非门串联，以达到改变输出电阻的目的。如 A 点左边电路的输出电阻很高，经过这个缓冲门之后，在 Y 点处的输出电阻就可以变得低许多倍。这就能够提高带负载的能力。

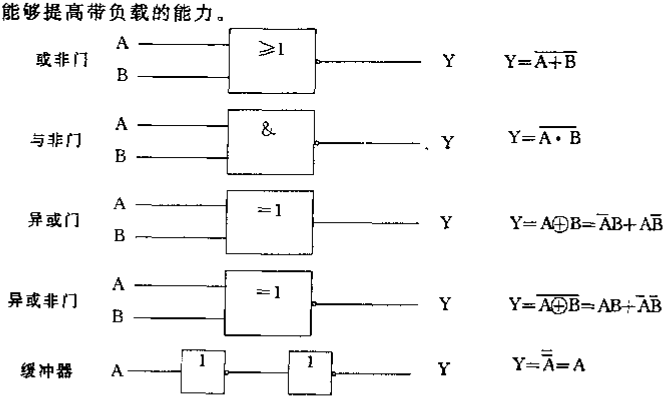


图2-4　其它的门电路

二、二进制数加法

例如两个二进数相加：

$$\begin{array}{rcl} & \boxed{1}\boxed{1} & C \\ & 0\ 1\ 1 & A \\ +) & 0\ 1\ 1 & B \\ \hline & 1\ 1\ 0 & S \end{array}$$

例中第一位(0 权位)上有加数与被加数,不可能有进位与之相加,因此要求运算的只有两个数 A_0 和 B_0,其结果为 S_0。第二位(称 1 权位)就有三个数 A_1、B_1、C_1 参与运算了,其中 C_1 是由于第一位相加的结果产生的进位。此三个数相加的和为 $S_1=1$,同时又产生进位 C_2,送入下一位(第三位)。第三位(或称 2 权位)也是三个数 A_2、B_2 及 C_2 相加运算,由于 A_2 及 B_2 都是 0,所以 C_2 即等于第三位的相加结果 S_2。

从上例的分析可以得到下列结论:

(1)两个二进制数相加,可以逐位相加。如二进制数可写成

$$A=A_3A_2A_1A_0$$

$$B=B_3B_2B_1B_0$$

其结果可以写成 $S=S_3S_2S_1S_0$

其中分别求出各位是 $S_0=A_0+B_0\rightarrow$进位 C_1

$$S_1=A_1+B_1+C_1\rightarrow\text{进位 }C_2$$

$$S_2=A_2+B_2+C_2\rightarrow\text{进位 }C_3$$

$$S_3=A_3+B_3+C_3\rightarrow\text{进位 }C_4$$

而最后所得到的和是 $A+B=C_4S_3S_2S_1S_0$

(2)右边第一位相加的电路要求:输入量为两个,即 A_0 及 B_0;输出量为两个,即 S_0 及 C_1。这样的一个二进制位相加的电路称为半加器(Half Adder)。

(3)右边第二位开始,各位可以对应相加。各位对应相加的电路要求:输入量为三个,即 A_i、B_i、C_i,输出量为两个,即 S_i 及 C_{i+1},其中 $i=1,2,3,\cdots,n$。这样的一个二进制位相加电路称为全加器(Full Adder)。

三、半加器电路

半加器电路设计:有两个输入端,以供两个代表数字(A_0,B_0)的电位输入;有两个输出端,用以输出总和 S_0 及进位 C_1。

这样的电路可能出现的状态用图 2-5 中的表表示。此表在布尔代数中称为真值表(True Table)。

考察一下 S_0 与 A_0 及 B_0 之间的关系,可以看出,当 A_0 与 B_0 取值相同时,其结果为 0;当 A_0 与 B_0 取值不同时,其结果为 1。这正是"异或"关系,即

真 值 表

A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0
		与门	异或门

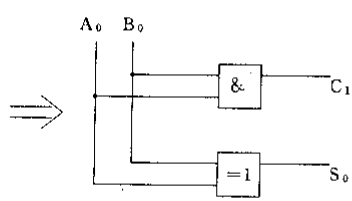


图2-5 半加器的真值表及电路

$$S_0=A_0\oplus B_0=A_0\overline{B_0}+\overline{A_0}B_0$$

再看一下 C_1 与 A_0 及 B_0 的关系，可以看出，当 A_0 及 B_0 有一个量为 0 或全为 0 时，结果 C_1 为 0；当 A_0 及 B_0 都为 1 时，其结果 C_1 为 1。显然这种“有零为零，全一为一”的关系正是“与”的关系，即

$$C_1 = A_0 \cdot B_0$$

因此，可以用“与”门及“异或门”来实现真值表中 C_1、S_0 与 A_0、B_0 的关系。图 2-5 就是这个真值表及半加器的电路图。图 2-6 所示为半加器符号。

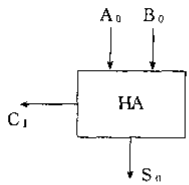


图2-6 半加器的符号

四、全加器电路

全加器电路设计：有三个输入端，以输入 A_i、B_i 和 C_i；有两个输出端，即 S_i 及 C_{i+1}。其真值表可以写成如图 2-7 所示。

真 值 表

A	B	C	C	S
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
			先“与”后“或”	异 或

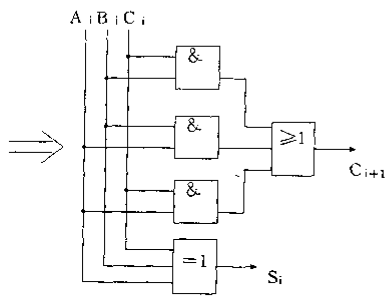


图2-7 全加器的真值表及电路图

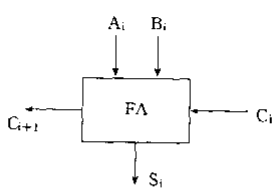


图2-8 全加器的符号

由此表分析其总和 S_i 与三个输入 A_i、B_i、C_i 的关系为：A_i、B_i、C_i 三个输入量中“1”的个数为奇数，则 S_i 为“1”；若 A_i、B_i 及 C_i 三个输入量中“1”的个数为零或偶数，则 S_i 为“0”。这正是三输入异或关系，即

$$S_i = A_i \oplus B_i \oplus C_i$$

我们分析进位 C_{i+1} 与输入 A_i、B_i 及 C_i 的关系，得到

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

这可以用三个“与门”及一个“或门”来实现。全加器的电路图见图 2-7。全加器的表示符号如图 2-8 所示。

五、二进制数的加法电路

设 $A = 1010 = (10)_{10}$，$B = 1011 = (11)_{10}$，A 与 B 相加，写成竖式算式如下：

$$\begin{array}{rll} & 1010 & A \\ +) & 1011 & B \\ \hline & 10101 & S \end{array}$$

即其相加结果为 S=10101。

如上 4 位二进制加法，可安排如下加法电路(图 2-9)。

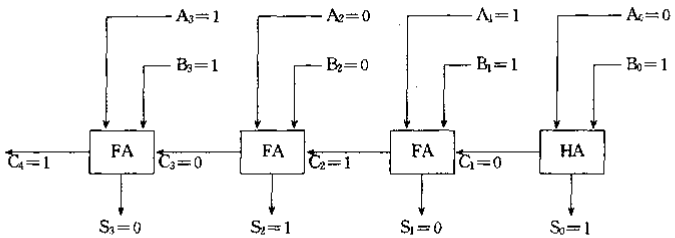


图2-9 4位二进制加法电路

从加法电路，可得到同样的结果 $S=C_4S_3S_2S_1S_0=10101$。

六、二进制减法电路

前面讲过，利用补码可以将二进制数的减法运算变为加法运算。因此首先需要这样一个电路，即能将原码变成补码(反码+1)。

图 2-10 的可控反相器就是为了使原码变为反码而设计的。这是一个异或门，两个输入端的异或门的特点是：两者相同则输出为 0，两者不同则输出为 1。用真值表来表示这个关系，更容易看到其意义(表 2-10)。

表 2-10

SUB	B_0	Y	Y 与 B_0 的关系	
0	0	0	Y 与 B_0 相同	同相
	1	1	Y 与 B_0 相同	
1	0	1	Y 与 B_0 相反	反相
	1	0	Y 与 B_0 相反	

由此真值表可见，如将 SUB 端看作控制端，则当在 SUB 端加上低电平时，Y 端的电平就与 B_0 端的电平相同；在 SUB 端加上高电平，则 Y 端的电平就与 B_0 端的电平相反。

利用这个特点，在图 2-9 的 4 位二进制数加法电路上增加四个可控反相器，并将最低位的半加器改用全加器，就可以得到如图 2-11 所示的 4 位二进制数加法器/减法器电路，这个电路既可以作为加法器电路(当 SUB=0)，又可以作为减法器电路(当 SUB=1)。

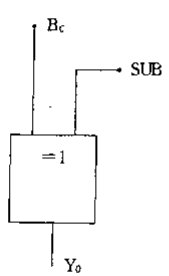


图2-10 可控反相器

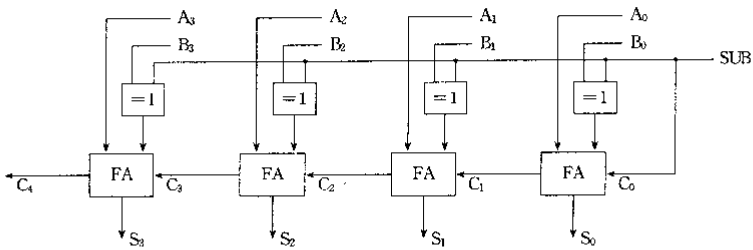


图2-11 二进制补码加法器/减法器

如果有两个二进制数：

$A=A_3A_2A_1A_0$　　$B=B_3B_2B_1B_0$

则可将它们的各位分别送入该电路的对应端，于是

当 SUB=0 时，电路作加法运算：A+B

当 SUB=1 时，电路作减法运算：A−B

图 2-11 电路的原理如下：当 SUB=0 时，各位的可控反相器的输出与 B 的各位同相，所以图 2-11 与图 2-9 的原理完全一样，各位均按位相加，结果 $S=S_3S_2S_1S_0$，而其和为：$C_4S=C_4S_3S_2S_1S_0$。

当 SUB=1 时，各位的反相器的输出与 B 的各位反相。注意，最右边第一位（即 S_0 位）也用全加器，其进位输入与 SUB 端相连，因此 $C_0=SUB=1$。所以此位的和即为

$$A_0+\overline{B_0}+1$$

其它各位是

$$A_1+\overline{B_1}+C_1$$

$$A_2+\overline{B_2}+C_2$$

$$A_3+\overline{B_3}+C_3$$

因此其总和输出 $S=S_3S_2S_1S_0$ 就是

$$\begin{aligned}S&=A+\overline{B}+1\\&=A_3A_2A_1A_0+\overline{B_3}\,\overline{B_2}\,\overline{B_1}\,\overline{B_0}+1\\&=A+B'\\&=A-B\end{aligned}$$

当然，此时 C_4 如不等于 0 的话，则要被舍去。

七、算术逻辑单元

算术逻辑单元（Arithmatic Logica Unit——ALU）是计算机中的重要部件，这个部件既能进行二进制数的四则运算，也能进行布尔代数的逻辑运算。

前面讲过，二进制的运算电路只能进行加法运算，增加可控反相器后，又能进行减法运算。只要利用适当的软件，乘法可以变成加法运算，除法也可以变成减法运算。所以这种二进制补码加法器/减法器就是最简单的算术部件。

如果在这个基础上，增加一些门电路，也可以使简单的 ALU 进行逻辑运算，所谓逻辑运算就是指“与”运算、“或”运算及“反”运算。为了不使初学者陷入复杂的电路分析之中，本书不在逻辑运算问题上展开讨论。

ALU 的符号一般如图 2-12 所示。A 和 B 为两个二进制数，S 为其运算结果，Control为控制信号，如图 2-11 的控制线端 SUB 就是其中的一个。

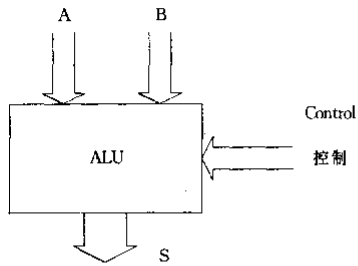


图2-12 ALU的符号

思考题与习题

1. 将下列二进制数转换成十进制数及十六进制数。
   (1) 1101　(2) 101101　(3) 1011101　(4) 10101010
2. 将下列十进制数转换成二进制数。
   (1) 57　(2) 101.42　(3) 0.042　(4) 256
3. 求下列二进制数的补码(8 位)。
   (1) −1011011　(2) +1000011　(3) −0000011　(4) −1011111
4. 什么叫原码、反码及补码？
5. 布尔代数的“或”运算可归纳为哪两句话？“与”运算呢？
6. 半加器与全加器的区别是什么？
7. 用补码法写出下列减法的步骤。
   (1) $1111_{(2)}-1010_{(2)}=?_{(2)}=?_{(10)}$
   (2) $1100_{(2)}-0011_{(2)}=?_{(2)}=?_{(10)}$
8. 做出 $101010_{(2)}+011101_{(2)}$的门电路图，并求其相加的结果。

第三章　计算机的硬件电路基础

任何一个复杂的电路系统都可以划分为若干电路，这些电路大都由一些典型的电路组成。微型计算机就是由若干典型电路通过精心设计而组成的，各个典型电路在整体电路系统中又称为基本电路部件。

微型计算机中最常见的基本电路部件有算术逻辑运算单元(ALU)、寄存器(Register)、存储器(Memory)及总线结构。上一章我们已介绍了算术逻辑运算单元ALU，本章将主要介绍其它三大计算机基本电路部件：寄存器、存储器及总线结构。由于构成寄存器、存储器的基本单元是触发器，本章第一节将介绍触发器。本章的最后介绍最简单的模型计算机原理，以便读者对计算机有一个基本的认识，同时了解数据在这些部件之间的流通过程。

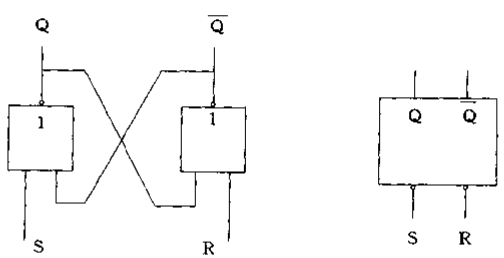
第一节　触发器

触发器(Trigger)是计算机的记忆装置的基本单元，也可以说是记忆细胞。触发器可以组成寄存器，寄存器又可以组成存储器。寄存器和存储器统称计算机的记忆装置。

微型计算机所用触发器一般用晶体管元件而不是磁性元件，这是因为晶体管元件可以制成大规模集成电路，体积可以更小。

一、RS触发器

基本RS触发器的逻辑图和逻辑符号如图3-1所示，电路由两个与非门交叉耦合构成。R、S是信号输入端，低电平有效，Q、$\overline{Q}$既表示触发器状态又是输出端。



(a)逻辑图　　(b)逻辑符号

图3-1　基本RS触发器

根据与非门的逻辑关系，触发器的表达式为

$$Q=\overline{S\overline{Q}} \quad \overline{Q}=\overline{RQ}$$

根据以上两式，触发器的输出与输入之间的关系可有四种情况：

(1)R=1,S=0

由式 $Q=\overline{S\overline{Q}}$ 可知，当 S=0 时，无论 Q 为何种状态，都有 Q=1；再由式 $\overline{Q}=\overline{PQ}$ 可得 $\overline{Q}=0$。

(2)R=0,S=1

由于电路的对称性，这时 Q=0,$\overline{Q}=1$。

当触发器的两个输入端加入不同逻辑电平时，它的两个输入端 Q 和 $\overline{Q}$ 有两种互补的稳定状态。一般规定触发器 Q 端的状态作为触发器的状态。Q=1、$\overline{Q}=0$ 时，触发器处于 1 态；反之，Q=0、$\overline{Q}=1$，触发器处于 0 态。S=0、R=1 使触发器置 1，或称置位。因置位的决定条件是 S=0，故称 S 端为置位端。R=0、S=1，使触发器置 0，或称为复位。同理，R 端为复位端。

(3)R=S=1

根据与非门的逻辑功能，当 R、S 全为 1 时，触发器保持原有状态不变，即原来的状态被触发器存储起来。这体现了触发器具有记忆功能。

(4)R=S=0

在此条件下，两个与非门的输出端 Q 和 $\overline{Q}$ 全为 1，破坏了触发器的逻辑关系。在两输入端的 0 信号同时撤除后，由于与非门延迟时间不可能完全相等，将不能确定触发器是处于 1 态还是 0 态。这种情况应避免。

上述逻辑关系可以用表 3-1 所示真值表来表示。

为了使触发器在整个机器工作中能与其它部件协调工作，RS 触发器经常有外加同步时钟脉冲，如图 3-2 所示，这种触发器叫作同步 RS 触发器。

表 3-1　基本RS触发器真值表

R	S	Q
1	0	1
0	1	0
1	1	不变
0	0	不定

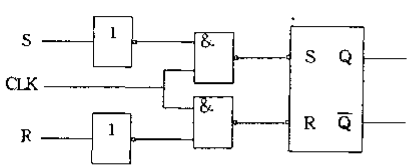


图 3-2　同步RS触发器

图中 CLK 即为同步时钟脉冲。它与置位信号脉冲 S 同时加到一个与非门的两个输入端；而与复位信号脉冲同时加到另一个与非门的两个输入端。这样无论是置位还是复位，都必须在时钟脉冲端为高电位时才能进行。

二、D 触发器

RS 触发器有两个输入端 S 和 R，为了存储一个高电位，就需要一个低电位输入 S 端；为了存储一个低电位，就需要低电位输入 R 端，这在很多应用中很不方便。D 触发器是在 RS 触发器的基础上引伸出来的，它只需一个输入口，图 3-3 就是 D 触发器的原理图。

当 D 端为高电位时，R 端为高电位，而通过非门加到 S 端的就是低电位，所以此时 Q

端为高电位，称为置位。当D端为低电位时，R端为低电位，而通过与非门加到S端变为高电位，所以Q端是低电位，称为复位。

没有同步时钟脉冲的D触发器是不能协调运行的，图3-4为同步D触发器电路。此图与图3-2的

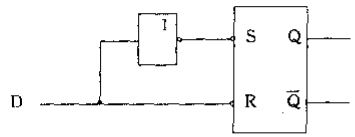


图3-3 D触发器

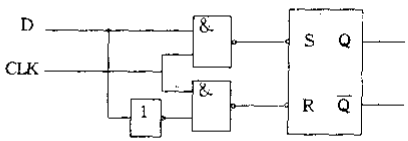


图3-4 同步D触发器

道理一样，也是增加两个与非门就可以接受同步时钟脉冲CLK的控制。时钟脉冲CLK一般都是方波，在CLK处于正半周内的任何瞬间，触发器都有翻转的可能。这样计算机的动作就不可能整齐如一。我们总是希望由同步时钟脉冲CLK来指挥整个机器统一行动。因此采用时钟脉冲边沿触发的方式就可以得到准确如一的动作。图3-5就是边沿触发的D触发器的电路原理图。

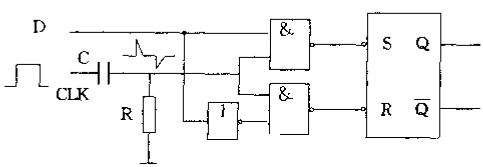


图3-5 边沿触发的D触发器

此图与图3-4的区别仅为增加了一个RC微分电路，它能使方波电压信号的前沿产生正尖峰，后沿产生负尖峰。这样，在D端输入信号建立之后，当时钟脉冲的前沿到达的瞬间，触发器才产生翻转动作。如果D输入信号是在时钟脉冲的前沿到达之后才建立起来的，则虽然仍在时钟脉冲的正半周时间内，也不能影响触发器的状态，而必须留到下一个时钟脉冲的正半周前沿到达时才起作用。这样就可以使整个计算机运行在高度准确的协调节拍之中。

触发器的预置和清除：在一些电路中，有时需要预先给某个触发器置位（即置1）或清除（即置0），而与时钟脉冲以及D输入端信号无关，这就是所谓的预置及清除。这种电路很简单，只要增加两个与门就可以实现，如图3-6所示。当预置端为低电平（有效），输出Q端为高电平，系统置位；当清除端为低电平（有效），输出Q端为低电平，系统复位。

边沿触发的D触发器在计算机电路中常用图3-7的符号表示。

图3-7(a)为正边沿触发的符号，而图3-7(b)为负边沿触发的符号。此二符号之差别在于后者增加了一个所谓汽泡"O"。这实际上是在D触发器的时钟脉冲CLK的微分电路之后再串联一个非门（反相器）的简化符号，表示低电平有效。在图3-7(a)及图3-7(b)中的PRESET及CLEAR端均有这种"汽泡"，表明PRESET及CLEAR均为低电平有效置

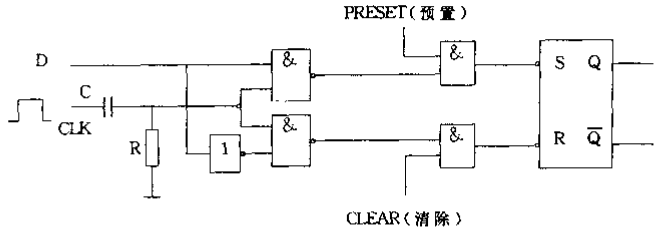
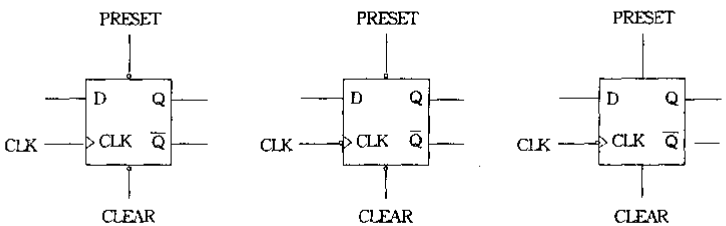


图3-6　边沿触发的D触发器预置及清除电路



(a)CLK正边沿触发的D触发器　(b)CLK负边沿触发的D触发器　(c)高电平预置及清除的D触发器

图3-7　正、负边沿触发的D触发器符号

位及清除。而图 3-7(c)中的 PRESET 及 CLEAR 端均无此"汽泡",表明高电平有效预置及清除。

三、JK 触发器

JK 触发器是组成计数器的理想记忆元件,这里就 JK 触发器的电路原理作一简要介绍。在 RS 触发器前面增加两个与非门,并从输出(Q 和 $\overline{Q}$)到输入(与门的输入端)作交叉反馈,即可得到 JK 触发器,如图 3-8 所示。图中的 CLK 输入端串有 RC 电路,是为获得正边沿触发的工作方式。其输入与输出关系为:

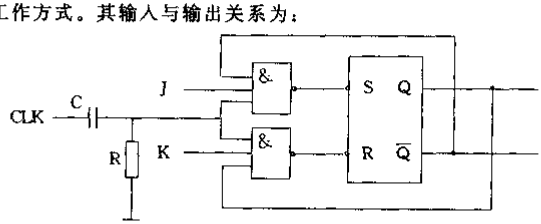


图3-8　JK触发器

(1)J=0,K=0

即 J 和 K 都是低电平时,两个与门都被阻塞。无论此时 Q 和 $\overline{Q}$ 是什么状态,由于 S 和 R 均为高电平(无效),所以不会改变 Q 和 $\overline{Q}$ 的状态,保持原状态不变。

(2)J=0,K=1

此时上面的与非门被阻塞,S为高电平(无效)。如果此时Q=1,则反馈至下面与非门。在下一个CLK的正脉冲边沿到达时,R端为有效低电平,使触发器复位(Q=0,$\overline{Q}$=1)。如果Q原来为低电平(Q=0,$\overline{Q}$=1),则反馈至下面的与非门也被阻塞,所以即使是K=1、R端仍然为高电平(无效),触发器保持原来复位状态(Q=0,$\overline{Q}$=1)不变。故当J=0,K=1时,不管原来状态如何,在CLK正脉冲边沿到达时,触发器处于复位状态(Q=0,$\overline{Q}$=1)。

(3)J=1,K=0

此时下面的与非门被阻塞,R为高电平,如果此时Q=0,而$\overline{Q}$=1,则反馈至上面的与非门。在下一个CLK的正脉冲边沿到达时,S端为低电平,使触发器置位(Q=1,$\overline{Q}$=0)。如果Q原来为高电平(Q=1,$\overline{Q}$=0),则反馈至上面的与非门也被阻塞,所以即使是J=1,S端仍然为高电平(无效),触发器保持原来置位状态不变,故当J=1、K=0时,不管原来状态如何,在CLK正脉冲边沿到达时,触发器处于置位状态(Q=1,$\overline{Q}$=0)。

(4)J=1,K=1

此时上、下面的两与非门均未被阻塞。当原状态(Q=0,$\overline{Q}$=1)时,在CLK正脉冲边沿到达时,上面的与非门输出为低电平,S端有效使触发器置位(Q=1,$\overline{Q}$=0),正好与原状态相反。如果原状态为(Q=1,$\overline{Q}$=0),在CLK正脉冲边沿到达时,下面的与非门输出为低电平,使R端有效,触发器被复位(Q=1,$\overline{Q}$=0),也与原状态相反。这种触发器的状态改变被称为翻转。

归纳上述四种情况,写成JK触发器的真值表见表3-2。JK触发器的符号表示如图3-9所示。

表3-2 JK触发器的真值表

J	K	Q
0	0	保持
0	1	复位
1	0	置位
1	1	翻转

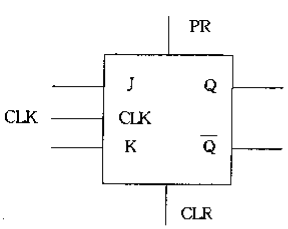


图3-9 JK触发器的符号

第二节 寄存器

寄存器(Register)是由触发器组成的。一个触发器就是一个1位寄存器。由多个触发器可以组成一个多位寄存器。寄存器由于其在计算机中的作用之不同而具有不同的功能,从而被命名为不同的名称。常见的寄存器有:

缓冲寄存器——用于暂存数据;

移位寄存器——能够将其所存的数据一位一位地向左或向右移;

计数器——一个计数脉冲到达时，会按二进制数的规律累计脉冲数。

累加器——用以暂存每次在 ALU 中计算的中间结果。

下面分别介绍这些寄存器的大致工作原理及其电路结构。

一、缓冲寄存器

缓冲寄存器(Buffer)用以暂存某个数据，以便在适当的时间按给定的计算步骤将数据输入或输出到其它记忆元件中去。图 3-10 是一个 4 位缓冲寄存器的电路原理图。

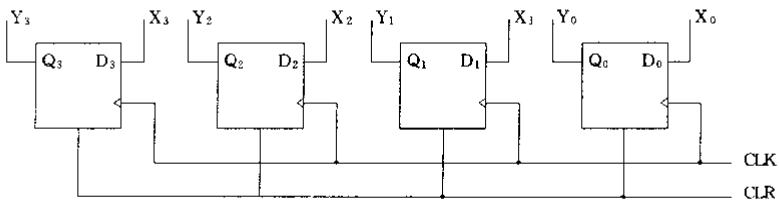


图 3-10　4 位缓冲寄存器电路原理图

此寄存器是由 4 个 D 触发器组成，每个 D 触发器均为正脉冲前沿控制。下面介绍 4 位缓冲寄存器的基本工作原理。

设有一个 4 位二进制数，从高到低 4 位分别用 X_3、X_2、X_1、X_0 表示，则这个 4 位二进制数可以表示为 $X_3X_2X_1X_0$。若要将此数存到缓冲寄存器中，只要将 X_3、X_2、X_1、X_0 分别送到各个 D 触发器的输入端 D_3、D_2、D_1、D_0 端上，当时钟脉冲 CLK 的正前沿还未到来之前，缓冲寄存器的输出 Q_3、Q_2、Q_1、Q_0 并不能接受 X_3、X_2、X_1、X_0 的影响而保持其原有的数据，只有当 CLK 的正前沿来到时，Q_3、Q_2、Q_1、Q_0 才接受 D_3、D_2、D_1、D_0 的影响，而变成

$$Q_3=X_3 \qquad Q_2=X_2$$

$$Q_1=X_1 \qquad Q_0=X_0$$

从而将数据 X 装到寄存器中。如果要将此数据送至其它记忆元件，则可由 Y_3、Y_2、Y_1、Y_0 各条引线引出去。

图 3-10 的缓冲寄存器的数据 X 输入到 Q 只是受 CLK 的脉冲影响，即只要将 X 各位加到寄存器各位 D 输入端，时钟脉冲一到，就会立即送到 Q 去。很多时候，X 数据输入一段时间后即被撤除，在这种情况下，在 CLK 正前沿一到 Q 就会随 X 的改变而变化。也许我们还想让原来缓冲寄存器中的数据多保留一些时间，上述电路显然无法实现。

解决的办法是为这个寄存器增设一个可控“门”。这个“门”的基本原理如图 3-11 所示，它是由两个与门、一个或门及一个与非门组成。

X_0 端送入数据(0 或 1)后，如 LOAD 端(以下简称 L 端)为低电位，则右边的与门被阻塞，X_0 过不去，而原来已存在此位中的数据由 Q_0 送至左边的与门。此与门的另一端输入从非门引来的与 L 端反相的电平，即高电位，所以 Q_0 的数据可以通过左边的与门，再经或门而送达 D_0 端。这就形成自锁，即寄存的数据能够可靠地存在其中而不会丢失。如 L 端为高电位，则左边与门被阻塞而右边与门可让 X_0 通过，这样 Q_0 的寄存数据不再受到

自锁，而 X_0 可以到达 D_0 端。只要 CLK 的正前沿一到达，X_0 即被送到 Q_0 去，这时就叫做装入(LOAD)。一旦装入之后，L 端又降至低电平，则利用左边的与门，X_0 就能自锁而稳定地存在 Q_0 中。

要记住，以后我们一提到"L 门"，大家就要想到图 3-11 的电路结构及其作用：高电平时使数据装入；低电平时，数据自锁在其中。

对于多位的寄存器，每位各自有一套如图 3-11 所示的电路，不过只用一个非门，并且只有一个 LOAD 输入端，如图 3-12 所示。

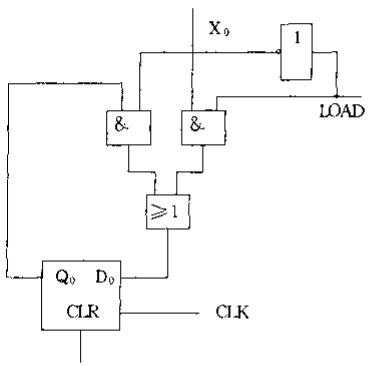


图 3-11　寄存器的装入门LOAD

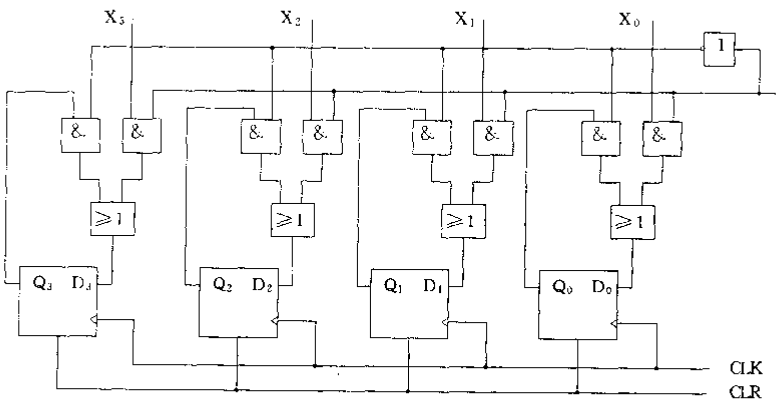


图 3-12　可控缓冲寄存器

可控缓冲寄存器的符号一般画成图 3-13 所示，LOAD 为其控制门，而 CLR 则为高电平时用做以清除，使其各位变 0。

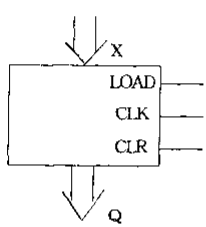


图3-13　可控缓冲寄存器的符号

二、移位寄存器

移位寄存器(Shifting Register)的主要功能是将寄存器中所存的数据，可以在移动脉冲作用下逐次左移或右移。

图 3-14 是用 D 触发器组成的单向左移寄存器，其中每个触发器的输出端 Q 依次接到下一个触发器的 D 端，只有第一个触发器的 D 端接收数据。假设初始状态 Q=

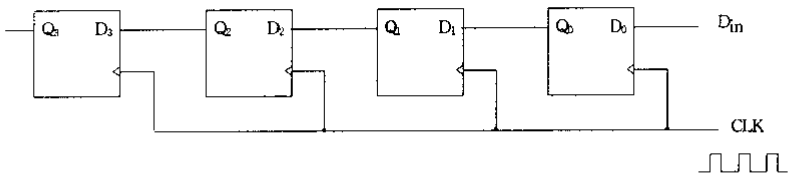


图3-14　左移寄存器

$Q_3Q_2Q_1Q_0=0000$，当 $D_{in}=1$ 送至最右边的第一位时，D_0 即为 1。当时钟脉冲前沿到来时，$D_0=1$ 状态送入第一个 D 触发器，即 $Q_0=D_0=1$，同时每个触发器的状态也送给下一个触发器，即 $D_1=Q_0=1$。当第二个时钟脉冲前沿到来时，$Q_0=1$。整个左移过程如下：

初始　　Q=0000　D=0001
第一个脉冲前沿到来　Q=0001　D=0011
第二个脉冲前沿到来　Q=0011　D=0111
第三个脉冲前沿到来　Q=0111　D=1111
第四个脉冲前沿到来　Q=1111

可见，右端输入一个 $D_{in}=1$，经过四个脉冲左移至左端 Q_3 端。当 Q=1111 时，改变 D_{in}，使 $D_{in}=0$，则结果将把 0 逐位左移。

第一个脉冲前沿到来　Q=1110
第二个脉冲前沿到来　Q=1100
第三个脉冲前沿到来　Q=1000
第四个脉冲前沿到来　Q=0000

如果按图 3-15 组成移位寄存器，则实现了单向右移电路。

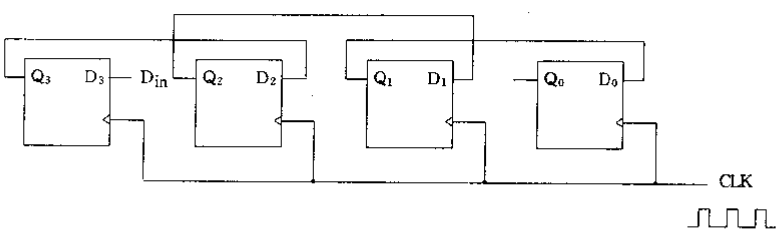


图3-15　右移寄存器

图 3-15 与图 3-14 差别仅在于各位的接法不同，图 3-15 输入数据 D_{in} 是加到左边第一位的输入端 D_3，其输出 Q_3 接右边第二位的输入端 D_2，以此类推，当 $D_{in}=1$ 时，随着时钟脉冲而逐步位移是这样的：

初始　　Q=0000
第一个脉冲前沿到来　Q=1000
第二个脉冲前沿到来　Q=1100
第三个脉冲前沿到来　Q=1110

第四个脉冲前沿到来　　Q=1111

由此可见，在右移寄存器中，每个时钟脉冲都要把存储的各位向右移动一个位置。

和缓冲寄存器一样，在整机运行中，移位寄存器也需要另有控制电路，以保证其在适当时机才参与协调工作。和上面的图 3-12 一样，只要在每一位的电路上增加一个 L 门即可达到控制的目的。这种移位寄存器叫可控移位寄存器，其符号如图 3-16 所示。

图中：SHL——左移(Shift to the Left)；

SHR——右移(Shift to the Right)。

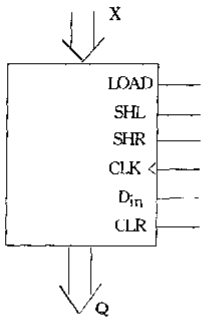


图3-16　可控移位寄存器的符号

三、计数器

计数器(Counter)也是由若干个触发器组成的寄存器，它的特点是能够把储存在其中的数字加 1。

计数器的种类很多，常用的有行波计数器、环形计数器和程序计数器等。

1. 行波计数器

行波计数器(Travelling Wave Counter)的特点是第一个时钟脉冲促使其最低有效位加 1，由 0 变 1。第二个时钟脉冲促使最低有效位由 1 变 0，同时推动第二位，使其由 0 变 1。同理，随着时钟脉冲变化计数，当第二位由 1 变 0 时又去推动第三位，使其由 0 变 1，这样有如水波前进一样逐位进位下去。图 3-17 就是由 JK 触发器组成的行波计数器的工作原理图。

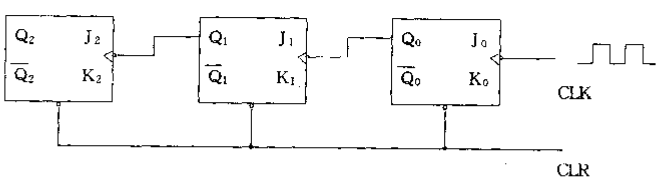


图3-17　行波计数器的工作原理图

图 3-17 中各位的 J、K 输入端都是悬浮的，这相当于 J、K 端都是 1 状态，触发器处于翻转状态。只要时钟脉冲边沿一到，最右边的触发器就会翻转，即 Q 由 0 转为 1 或由 1 转为 0。各位 JK 触发器的时钟脉冲输入端都带有一个“汽泡”，表示时钟脉冲的后沿有效。

计数器的工作过程如图 3-18 所示。

计数器在 CLK 脉冲作用下，计数值 $Q=Q_2Q_1Q_0$。变化为 000→001→010→011→100→101→110→111。当第 8 位时钟脉冲 CLK 后沿到达，计数器的值复位至 000。可见 3 位计数器可计 0～7 的数。如果要计的数更多，就需要更多的位，即由更多个 JK 触发器来组成计数器。如 4 位计数器可计由 0～15 的数，8 位计数器可计从 0～255 的数，16 位计数器可计从 0～65 535 的数。

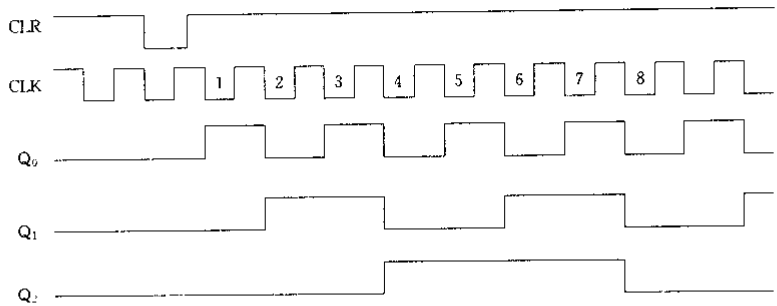


图3-18　3位行波计数器工作时序图

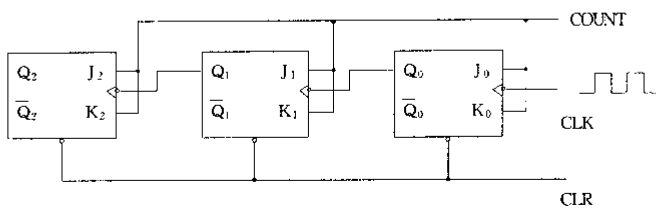


图3-19　3位可控计数器原理

如果把图 3-17 中的各个 J、K 输入端连在一起引出来，由计数器控制端 COUNT 的电位信号来控制，如图 3-19 所示，当 COUNT 为高电位时，JK 触发器才有翻转的可能；当 COUNT 为低电位时，根据 JK 触发器的功能表尽管遇到有效的时钟脉冲边沿，也不能翻转，而保持原有数据不变。这种计数器称为可控行波计数器，图 3-20 是这种计数器的符号。

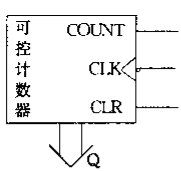


图3-20　可控计数器符号

2. 环形计数器

环形计数器(Ring Counter)与行波计数器不同，环形计数器仅有唯一的一个位为高电位，即只有一位为 1，其它各位为 0。环形计数器也是由若干个触发器组成的。图 3-21 为用 D 触发器组成环形计数器的电路原理图。

当 CLR 端有高电位输入时，除右边第一位外，其它各位的清除电位 CLR 都接至它们的 CLR 端，从而使这些位清零。而右边第一位则由于清除电位 CLR 接至其 PRESET(置位)端，而使该位置 1。就是说开始时，$Q_0=1$，$Q_1=Q_2=Q_3=0$，因此，$D_1=1$，而 $D_0=Q_3=0$。在时钟脉冲前沿来到时，则 $Q_0=0$，$Q_1=1$，$Q_2=Q_3=0$。第二个时钟脉冲前沿来到时，$Q_0=0$，$Q_1=0$，$Q_2=1$，$Q_3=0$。这样，随着时钟脉冲前沿的来到，各位轮流置 1，并且是在最后一

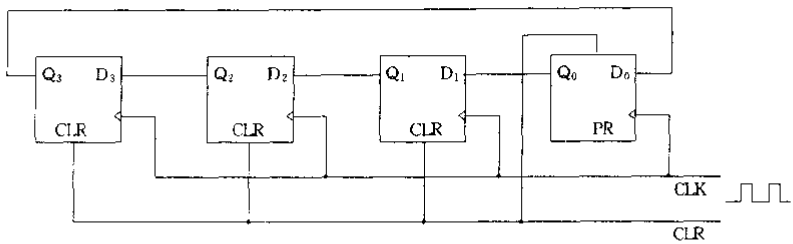


图 3-21　环形计数器的电路原理图

位(左边第一位)置 1 之后又回到右边第一位,这就形成环形置位,所以称为环形计数器。环形计数器的工作时序见图 3-22,其符号见图 3-23 所示。

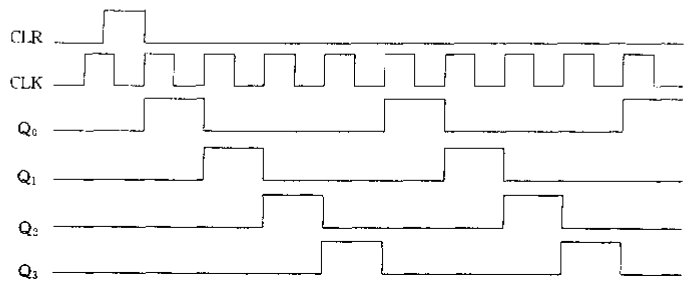


图 3-22　环形计数器工作时序图

环形计数器不是用来计数的,而是用来发出顺序控制信号。它在计算机的控制器中是一个很重要的部件。

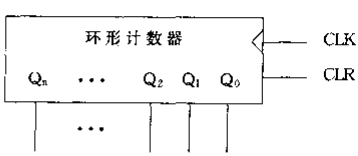


图 3-23　环形计数器的符号

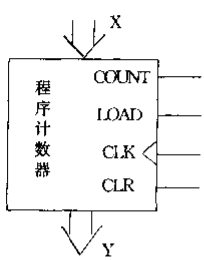


图 3-24　程序计数器符号

3. 程序计数器

程序计数器(Program Counter)也是一个行波计数器,它不但可以从 0 开始计数,也可以将外来的数装入其中,并从此开始计数,因此需要有一个 COUNT 输入控制端及一个 L 门。程序计数器的符号可用图 3-24 表示。

四、累加器

累加器也是一个由多个触发器组成的多位寄存器，累加器原文为 Accumulator，译作累加器。其实累加器并不是进行算术加法运算，而是作为算术/逻辑运算部件 ALU 运算过程的代数和的临时存储寄存器。这种特殊的寄存器在微型计算机的数据处理中担负着重要的任务。累加器除了能装入和输出数据外，还能使存储器中的数据左移或右移，所以它又是一种移位寄存器。累加器的符号如图 3-25 所示。

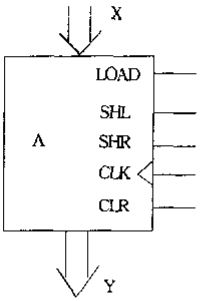


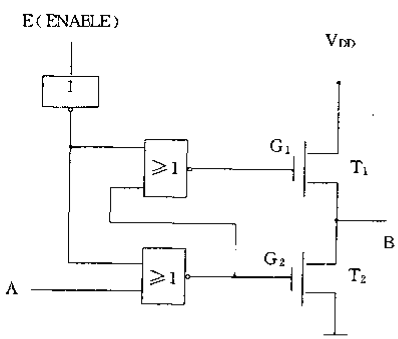
图 3-25　累加器的符号

第三节　总线结构

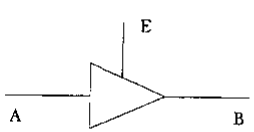
计算机中有许多寄存器，这些寄存器的输入输出端往往都需要连接在一组传输线上，进行相互之间的信息传送。由于寄存器是由触发器组成的，而触发器只有状态 0 和 1，故一般情况下每条信号传输线只能传送一个触发器的信息（0 或 1）。如果一条传输线既能与一个触发器接通，也可以与其断开而与另一个触发器接通，则一条信息传输线就可以分时传送多个触发器的信息。同样，一组传输线就能传输多位寄存器中的信息。我们把这样的一组传输线称为“总线”。为了完成这样的功能，需要在构成寄存器的每个触发器上增加一些电路。三态输出电路（或称三态门）就是为了达到这个目的而设计的。

三态输出电路可以由两个或非门和两个 NMOS 晶体管（T_1、T_2）及一个非门组成，如图 3-26 所示。

当 ENABLE（选通端）为高电位



(a) 电路



(b) 符号

图 3-26　三态输出电路及其符号

时，通过非门而加至两个或非门的将为低电位，则两个或非门的输出状态将取决于 A 端的电位。当 A 为高电位，G_2 就是低电位，而 G_1 为高电位，因而 T_1 导通而 T_2 截止，所以 B 端也呈现高电位（$V_B=V_{DD}$）；当 A 为低电位，G_2 将呈现高电位而 G_1 为低电位，因而 T_1 截止而 T_2 导通，所以 B 也呈现低电位（$V_B=0$）。这就是说，在选通端 E 为高电位时，A 的两种可能电平（0 和 1）都可以顺利地通到 B 输出去，即 E=1 时，B=A。

当选通端 E 为低电位时，通过非门加至两个或非门的将为高电位。此时，无论 A 为高或低电位，两个或非门的输出都是低电位，即 G_1 与 G_2 都是低电位，所以 T_1 和 T_2 同时都是截止状态。这就是说，在选通端 E 为低电位时，A 端和 B 端是不相通的，即它们之间存在着高阻状态。

这种电路的通断状态可以用表 3-3 来表示，三态输出电路的符号如图 3-26 表示。

图 3-26 所示为单向三态输出电路。有时需要双向输出时，一般可以用两个单向三态输出电路来组成，如图 3-27 所示。A 为某个电路装置的输出端，C 为其输入端。当 $E_{OUT}=1$ 时，B=A，即信息由左向右传输；$E_{IN}=1$ 时，C=B，即信息由右向左传输。

表3-3　三态输出电路的逻辑表

E	A	B
0	0	高阻
	1	高阻
1	0	0
	1	1

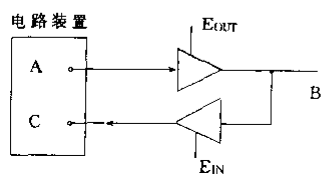


图 3-27　双向三态输出电路

三态门（E 门）和装入门（L 门）一样，都可加到任何寄存器（包括计数器和累加器）电路上去。这样的寄存器称为三态寄存器。L 门只控制对寄存器装入数据，而 E 门只控制由寄存器输出数据。

有了 L 门和 E 门就可以利用总线结构，使计算机的信息传递线路简单化，控制器的设计也更为合理而易于理解了。

设有 A、B、C、D 四个寄存器，它们都有 L 门和 E 门，其符号分别附以 A、B、C、D 下标。设它们的数据位数为 4 位，这样只要有四条数据线即可沟通它们之间的信息来往。图 3-28 就是总线结构的原理图。

如果将各个寄存器的 L 门和 E 门按次序排成一列，则可称其为控制字：

$$CON = L_AE_AL_BE_BL_CE_CL_DE_D$$

为了避免信息在公共总线 W 中乱窜，必须规定在某一时钟节拍（CLK 的正半周），只有一个寄存器 L 门为高电位，和另一寄存器的 E 门为高电位，其余各门则必须为低电位。这样，E 门为高电位的寄存器的数据就可以流入到 L 门为高电位的寄存器中去。详见表 3-4。

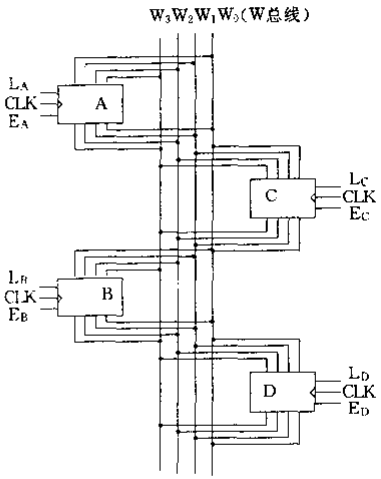


图3-28　总线结构的信息传输

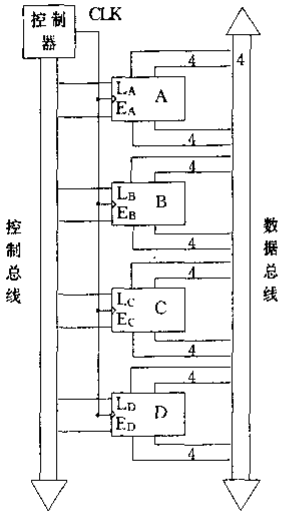


图3-29　总线结构符号图

表 3-4　控制字的意义

控制字 CON								信息流通
L_A	E_A	L_B	E_B	L_C	E_C	L_D	E_D	
1	0	0	1	0	0	0	0	数据由 B→A
0	1	1	0	0	0	0	0	数据由 A→B
0	1	0	0	1	0	0	0	数据由 A→C
0	1	0	0	0	0	1	0	数据由 A→D
0	0	1	0	0	0	0	1	数据由 D→B
1	0	0	0	0	1	0	0	数据由 C→A

控制字中哪些位为高电平，将由控制器发出并送到各个寄存器上去。

为了简化作图，不论总线包含几条导线，都用一条粗线表示。在图 3-29 中，有两条总线，一条称数据总线，专门让信息(数据)在其中流通；另一条称为控制总线，发自控制器，它能将控制字各位分别送至各个寄存器上去。控制器有一个时钟(CLK)，能把 CLK 脉冲送到各个寄存器上去。

第四节　存储器

存储器(Memory)是微型计算机的重要组成部分，有了它计算机才能有记忆功能，才能把要计算和处理的数据以及程序存入计算机，使计算机脱离人的直接干预，自动地工作。

显然，存储器的容量越大，则记忆的信息越多，计算机的功能就越强。同时如前所述，计算机中的操作多数是与存储器交换信息，但是存储器的工作速度相对于CPU中的算术和逻辑运算单元的速度要低，所以，存储器的工作速度又是影响计算机的运算速度的主要因素。

计算机存储器的发展主要是扩大容量，加快速度，缩小体积，降低成本。随着大规模集成电路技术的发展，半导体存储器的集成度大大提高，体积急剧减小，同时存取速度也有了极大的提高。目前，在微型计算机中，内部存储器几乎全部采用半导体存储器，而外部存储器采用磁盘、光盘等。

本节我们只讨论半导体存储器。

一、半导体存储器的分类及性能指标

半导体存储器从使用功能上来分，可分为随机存储器RAM(Random Access Memory)和只读存储器ROM(Read Only Memory)两类。随机存储器是在机器运行期间随时可以写入或读出的存储器，按其基本存储单元电路的类型可分为双极型和MOS型随机存储器。MOS型随机存储器又可分为静态存储器和动态存储器两种。只读存储器中的信息是在机器制造时，或者使用机器之前已经写入，机器运行时只能读出使用，而不能随机存入。只读存储器又可分为三种：固定只读存储器ROM、可编程序的只读存储器PROM和可改写的只读存储器EPROM。固定只读存储器ROM是指其中的信息由厂家在制造时写入，用户买回后只能读出使用，而不能对其进行任何修改。可编程只读存储器PROM是指存储器买回时，所存的信息为全"0"或全"1"，用户可以根据自己的需要，使用专门的电路进行写入。一旦写入，以后只能读出使用，而不能再进行修改。可改写的只读存储器EPROM是指用户在写入以后，可以通过专门的方法将其擦去，重新写入。在联机使用时，其中的内容只能读出，不能随机写入。

半导体存储器的分类如图3-30所示。

半导体存储器种类不同，其性能指标也各不相同，选用时要注意以下几个问题：

(1)存储容量

在存储器中，常以存储单元为其单位。一个存储器单元可以存放若干个二进制数。这若干个二进制数可以是一个字节或一个字长。在微型计算机中，常以8位二进制数为一个字节，16位二进制数为一个字长。存储器容量是指存储字节的数量。在微型计算机中，常用的存储器容量有2K、4K、8K、16K、32K、64K、128K、256K、512K、640K、1M(1KK)等，在一些高档微型计算机中容量可达4M～32M字节单元，甚至更多。存储1位二进制信息的单元称为基本存储单元。对于8K字节的存储器，其内部有8K×8个基本存储单元。

(2)存取周期

存储器从接到存储单元的地址开始，到读出或写入数据为止所需要的时间称为存取周期。

计算机的速度与存储器的存取周期有着直接的关系，因此它是存储器的重要参数。一般情况下，存取周期越短，计算机速度才能越高。半导体存储器的存取周期一般为几十～几百纳秒。现在半导体存储芯片上都有类似于"－6"、"－7"、"－10"、"－15"等标记，分别

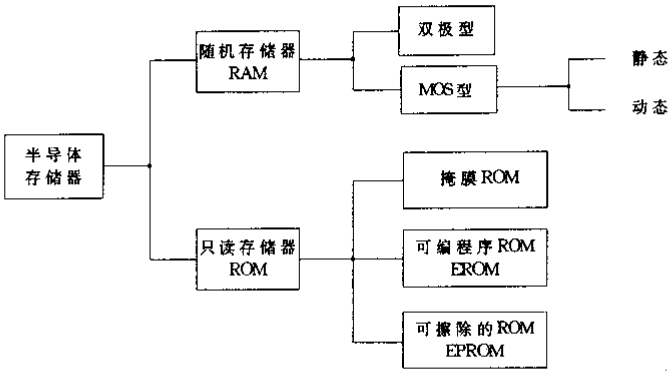


图3-30　半导体存储器的分类

指存取时间为 60ns、70ns、100ns、150ns。

(3)功耗

半导体存储器集成度高，体积小，散热困难，因此在保证存取速度的前提下，应尽量减小功耗。在半导体存储器中，MOS 型存储器的功耗小于双极型存储器的功耗。

(4)可靠性

可靠性是指存储器对磁场、温度变化等因素的抗干扰能力，以及在高速度使用时也能正确存取。半导体存储器采用大规模集成电路，内部电路少，抗干扰能力强。

(5)集成度

半导体存储器的集成度是指一块数平方毫米芯片上所制作的基本存储单元数。常用单元/片或位/片来表示。目前，MOS 型存储器集成度高于双极型存储器，动态存储器的集成度高于静态存储器，如 Intel 2114 为 1K×4 位/片，动态 MOS 存储器的集成度发展到 16M 位/片。

二、随机存储器(RAM)

1. 基本存储单元电路

基本存储单元电路是组成存储器的基础和核心，它用以存储 1 位二进制信息："0"或"1"。基本存储单元分为静态和动态两大类，下面主要介绍 MOS 型静态和动态基本存储单元电路的工作原理。

(1)六管静态存储单元电路

图 3-31 是 N 通道增强型 MOS 六管静态存储元件。它可以存储 1 位二进制信息。

T_3、T_4、T_5、T_6 组成一个双稳态触发器，T_1、T_2 相当于二个门开关，称为门控管。当 T_1、T_2 截止时，存储元件与外界没有联系；T_1、T_2 导通时，则与外界联系，可以读写信息。T_1、T_2 的栅极受字线 Z 控制，漏极分别与位线 W、W′连接，W、W′线也是读出线。

当字线 Z 为低电位时，T_1、T_2 均截止，相当于开关断开，因而这个存储元件与外界无联系，触发器所保存的信息不会传送到位线 W、W′上去，所以触发器维持原状态不变。

读出时，字线 Z 加高电位，使 T_1、T_2 导通，若原存信息为 1 及 T_3 导通、T_4 截止，则使 Q 点的高电位传送到 W′线上，$\overline{Q}$ 点的低电位传送到 W 线上。同理，若原存信息为 0 及 T_3 截止、T_4 导通，则使 Q 点的低电位传送到 W′线上，$\overline{Q}$ 点的高电位传送到 W 线上。

写入时，字线 Z 上加高电位，与此同时，若写入 1，则在 W 线上加低电位，W′线上加高电位，从而使 T_3 导通、T_4 截止，达到写 1 的目的；若写 0，则在 W 线上加高电位，W′线上加低电位，从而使 T_3 截止、T_4 导通，达到写 0 的目的。

(2)四管动态存储单元

静态存储单元由 1 个双稳态触发器组成。在没有外界触发信息时，触发器的状态不会改变，能长时间保持所存信息不变，这也就是静态这个名称的来源。

动态存储单元和静态储存单元不同，即使没有外界信号作用，原有信息也不能长期保存，只能保存很短一段时间(几毫秒～几十毫秒)，超过这一时间，原存信息就会自动消失。为了能长期保存原有信息不变，必须在它消失之前，经常地、周期地使原有信息再生。这一工作称为“刷新”。这种需要刷新的存储单元称为动态存储单元。

四管动态存储单元如图 3-32 所示。

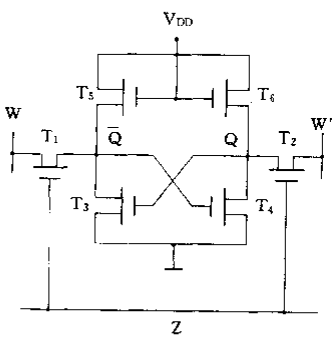


图 3-31　六管静态存储单元

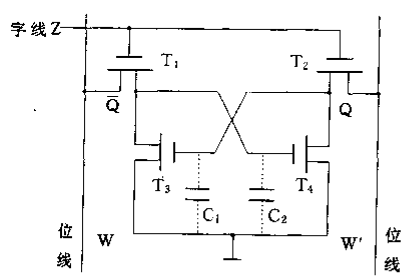


图 3-32　四管动态存储单元

去掉图 3-31 中的 T_5、T_6，就将六管静态存储单元变成图 3-32 所示的四管动态存储单元。这时，二进制信息以电荷形式存储在 MOS 场效应晶体管 T_3、T_4 的栅极电容 C_1 和 C_2 上。若 C_2 被充电到高电位，C_1 为低电位，即 T_4 导通、T_3 截止，则 Q 点为低电位，$\overline{Q}$ 成为高电位，所存信息为 0；而当 C_1 充电到高电位，C_2 为低电位时，T_3 导通，T_4 截止，$\overline{Q}$ 点为低电位，Q 点为高电位，所存信息为 1。

读出时，Z 线为高电位，W、W′也为高电位，若原存信息为 0，即 C_2 充电到高电位，T_4 导通，位线 W′上有电流经 T_2 流入 T_4；若原存信息为 1，即 C_1 充电到高电位，T_3 导通，位线 W 上有电流经 T_1 到 T_3。因此，根据位线 W′上有电流还是 W 上有电流，便可以判断读出信息是 0 还是 1，这种读出称为不破坏读出，读出后存储元件所存内容不变，所以不需重写。

写入时，Z 为高电位，同时使 W′为低电位，W 为高电位，则 T_1、T_2 都导通，位线 W 的高电位通过 T_1 向电容 C_2 充电，使 C_2 充电到高电位，此时位线 W′的低电位通过 T_2 使 C_1 放电到低电位，于是 T_4 导通，T_3 截止，Q 点为低电位，这样就实现了写 0。反之，若使 W′为高电位，W 为低电位，便可实现写 1。

由于信息是以电荷的形式存储在 T_3、T_4 管栅极电容 C_1、C_2 上，当 Z 线为低电位时，存储元件处于维持状态，即不读不写，虽然 MOS 晶体管对于地泄漏电流很小，但总会有一点，因而 C_1、C_2 上的电荷就会慢慢释放掉。由于电荷不断释放，C_1、C_2 上的电位不断降低，到一定时间就由高电位变成低电位，这样原在 C_1、C_2 上的信息（信息以电位高低表示）就会丢失。因而，采用这种存储元件时，每隔一定时间，对栅极电容 C_1、C_2 必须充电一次，以补充泄漏掉的电荷，使 C_1、C_2 上的电位在降到允许值之下以前就恢复到原来的电位值，这就是所谓的刷新。

刷新过程对存储元件本身来讲，和读出过程基本相同，只要使位线 W′、W 为高电位，并使 Z 线也为高电位，则位线 W′、W 就可以分别通过 T_2、T_1 管对栅极电容 C_1、C_2 充电，使 C_1 或 C_2 上由于泄漏电流而降低了的电位恢复到原来的电位。

（3）三管动态存储单元

从四管动态存储单元可以看出，T_3、T_4 的状态总是相反，因此只要用一个管子的状态：导通或截止，也就是用栅极电容充电或放电来代表 1 或 0，就可以将四管电路改为三管电路，如图 3-33 所示。

写入时，写字线为高电位，这时 T_1 管导通。若写 1，则写位线为高电位，C 通过 T_1 充电，这时 C 中保存的信息为 1；若写 0，则写位线为低电位，C 通过 T_1 放电，这时 C 中保存的信息为 0。当写入完毕，写字线为低电位，T_1 截止，写入的 1 或 0 便保存在 C 中。

读出时，读字线上加高电平，使 T_3 导通。若原存信息为 1，则 T_2 导通，由此便有读出电流流过读位线；若原存信息为 0，则 T_2 截止，读位线上无电流流过。所以，根据读位线上是否有电流流过，便可以判断原存信息是 0 还是 1。如果周期性地读出信息（但不往外输出），把它反相后再写入此单元，就可以实现刷新。

（4）单管动态存储单元

单管动态存储单元的工作原理与三管动态存储单元相似，C 是存储电容，如图 3-34 所示。

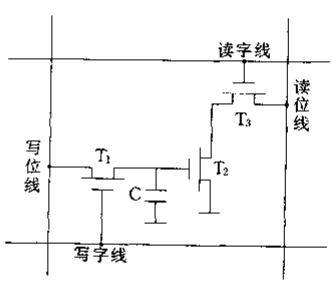


图3-33　三管动态存储元件

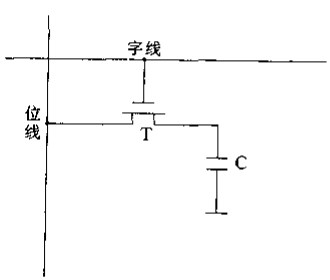


图3-34　单管动态存储元件

写入时，字线为1，T导通，写入信号由位线存入电容C中；读出时，字线为1，存储在电容C上的电荷，通过T管输出到位线上，通过读出放大器即可得到存储信息。

为了节省面积，这种单管存储单元的电容不可能做得很大。一般都比数据线上的分布电容小，因此每次读出后，存储内容就被破坏，要保存存储的信息必须采取恢复措施。

综上所述，动态存储单元由于采用电荷存储方式存储信息，所需管子数比静态存储器要少，速度快，集成度高，价格便宜，但由于需要刷新，故用它构成存储器时，外围控制电路比较复杂；静态存储单元工作稳定，外围控制电路简单。另外，在上面介绍的三种动态存储单元中，也各有优缺点：四管单元的缺点是管子多，占用的芯片面积大，它的优点是外围电路简单，读出过程就是刷新过程，故在刷新时不需要另加外部逻辑；三管单元管子数量较少，但因读写是分开的，存储信息刷新需要通过外部的电路反馈，所以存储电路与外围电路的连接较多；单管单元元件数量最少，但因读"1"和"0"时，数据线上的单元差别很小，需要有相当高鉴别能力的读出放大器配合工作，外围电路更复杂。

2. RAM 的结构

一个基本存储单元能存储一个二进制位，如果一个存储容量为 32K 的存储器，则需要 32K×8 个基本存储单元电路，因而存储器是由大量的存储单元电路组成的。这些存储单元电路必须有规则地组合起来，这就是存储体。

为了区别不同的存储单元，就要给它们各起一个"门牌号"——地址，不同的单元，具有不同的地址。我们就是以地址来选择不同的存储单元的。于是，在电路中就要有地址寄存器和地址译码器用来选择所需要的单元；另外，选择时往往还需要有驱动电路，读出的信息还要有放大电路等等。总之，在存储器中除了存储体外，还要有相应的外围电路。一个典型的 RAM 的示意图如图 3-35 所示。

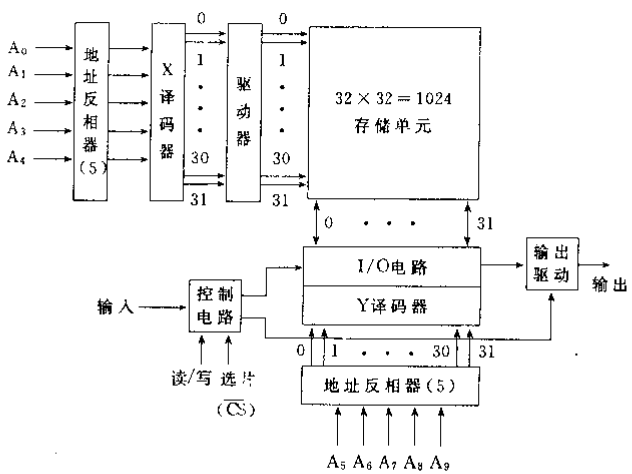


图 3-35　典型的RAM示意图

(1)存储体

在较大容量的存储器中，往往把各个字在同一位组织在一个芯片上，如图 3-35 中的 1024×1，它是 1024 个字的同一位。由这样的 8 个芯片则可组成 1024×8。同一位的这些字通常排列成矩阵的形式，如 32×32=1024。由 X 选择线——行线和 Y 选择线——列线的重叠来选择所需要的单元。这样做可以节省译码和驱动电路。

(2)外围电路

一个存储器除了由基本存储单元构成存储体外，还有许多外围电路。

①地址译码器：存储单元是按地址来选择的，如 64K 存储器，地址信息需要 16 位（2^{16}=64K）。计算机需要选择某一单元，就在地址总线上输出该单元的地址信号给存储器，存储器就必须对地址信号进行译码，用以选择需要访问的单元。

②I/O 电路：它处于数据总线和被选用的单元之间，用以接通被选中的单元读出或写入，并且有放大信息的作用。

③电路控制端$\overline{CS}$(Chip Select)：目前每一芯片的存储容量仍然是有限的，往往一个存储体是由一定数量的芯片组成。在地址选择时，首先要选片，用地址译码器输出和一些控制信号形成的片选信号，只有当片选信号$\overline{CS}$有效（低电平），此片所连接的地址线才有效，才能对这一芯片上的存储单元进行读写操作。

④三态输出缓冲器：为了扩展存储器的字数，常需要将几片 RAM 的数据线与双向的数据总线相连，这就需要用到三态输出缓冲器。

此外，在有些 RAM 中为了降低平均功耗，采用浮动电源控制电路，对未选中的单元降低电源电压，但保证维持信息。在动态 RAM 中，还有刷新方面的控制电路。

(3)地址译码方式

地址译码就是选择某一存储单元，共有两种方式：一种是单译码方式，适用于小容量存储器中；另一种是复合译码方式，常用于大容量存储器中。

①单译码方式：单译码方式如图 3-36 所示，有一条字线一次性选择某一字的所有位。图中的存储器中，有 16 个存储单元，每个单元有 4 位，共有 64 个基本存储单元电路，排列成 16 行×4 列的矩阵。每一条对应一个字，共用一个字选择线。这个字有 4 位，分别有自己的读/写控制电路。基本存储单元可采用六管静态存储器，也可采用四管动态存储器。每位的内部读/写数据线分别通过各位的读/写控制电路与外部数据总线连接。读/写控制电路只有接到读/写命令后才能对存储器进行相应的读写操作。

由于只有 16 个存储单元，故只需有 4 条地址线 A_3、A_2、A_1、A_0，经地址译码器译码后可产生 16 个信号，分别控制 16 个存储单元的地址选择。例如若输入地址 A_3～A_0 为 0000，经译码后字 0 选择线为有效高电平，表示第 0 个存储单元被选中。若输入地址 A_3～A_0=1111，则第 15 个存储单元被选中。当某一存储单元被选中后，其所有位在读/写命令的控制下，同时读出或写入，完成一次读/写操作。

②复合译码方式：复合译码是由纵横交错的 X 选择线和 Y 选择线互相配合来选择某一存储单元，如图 3-37 所示。电路中，1024 个基本存储单元排列成 32×32 的存储阵列，有两个地址译码器，一个是水平方向的 X 地址译码器，它决定选择 32 行中的某一行，另一个是垂直方向的 Y 地址译码器，它决定选择 32 列中的某一列。地址线分为两部分，一

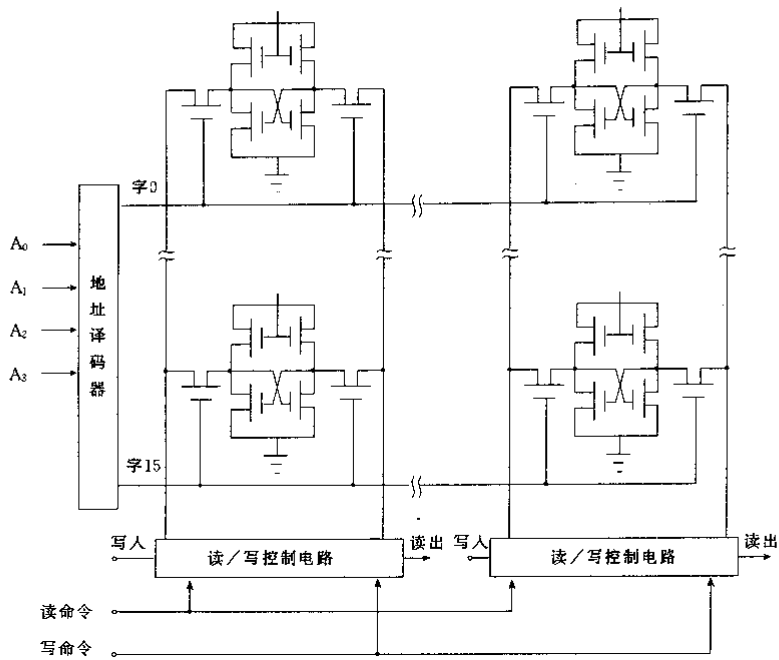


图3-36 单译码方式存储器

部分 $A_4 \sim A_0$ 是 X 地址译码器，另一部分 $A_9 \sim A_5$ 是 Y 地址译码器。工作时，10 位地址分别经 X、Y 地址译码器译码，选择出某一行和某一列交叉处的一个存储单元。

例如，当地址 $A_9 \sim A_0 = 0000000000$ 时，$A_4 \sim A_0$ 位经 X 地址译码器译码，译码选择第一行，则 X_0 为高电平；$A_9 \sim A_5$ 经 Y 地址译码器译码，选中了第一列，Y_0 为高电平。二者共同作用，则选中了(0,0)单元，即可向该单元进行读/写操作。

通过以上分析可以看出，采用复合译码方式可以减少选择线的数量。例如，对于 1024 个单元的存储量，若采用复合译码方式，需要 $2^5+2^5=32+32=64$ 条选择线；若采用单译码方式，需要 $2^{10}=1024$ 条选择线。

图 3-37 所示为所有单元的同一位。对于 8 位存储器，则由 8 个这样的阵列重叠起来组成，每一层为一位。

(4)一个实际的静态 RAM 芯片

Intel 2114 是一个 1K×4 位的静态 RAM 芯片，它的结构如图 3-38 所示，芯片的引脚如图 3-39 所示。Intel 2114 芯片地址共有 1024×4=4096 个静态六管基本存储单元，排列成 64×64 的方阵。2114 共有 10 条地址线 $A_9 \sim A_0$，其中 6 根 $A_3 \sim A_8$ 用于行译码，产生 64 根行选择线，用于选择 64 行中的某一行；另外四根地址线 A_0、A_1、A_2、A_9 用于列译码以产生 16 组列选择线，在每组列选择线上同时接 4 位(一个存储单元为 4 位)。

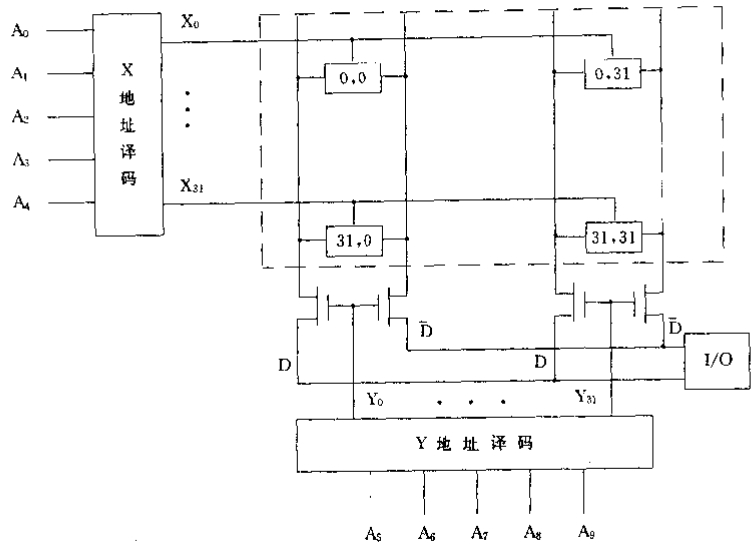


图3-37　复合译码方式存储器

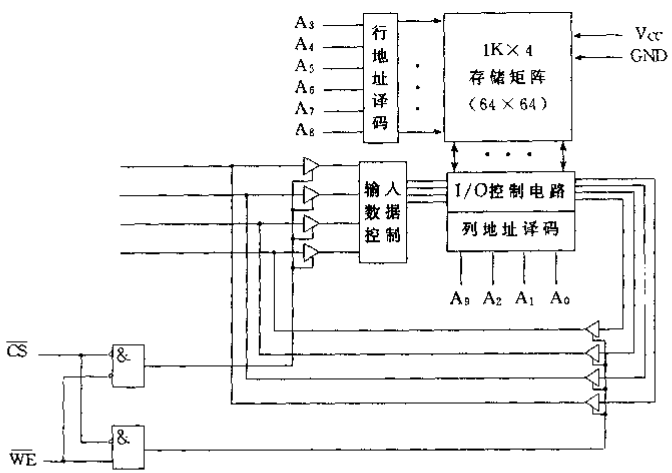
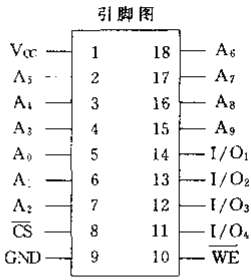


图3-38　2114的结构方框图

存储器的内部数据总线通过I/O电路以及输入输出三态门与外部数据总线连接。由片选信号$\overline{CS}$和写允许信号$\overline{WE}$一起控制这些三态门。当$\overline{WE}$有效(低电平)时,使输入三态



引 脚 名

$A_0 \sim A_9$	地址输入	Vcc	电源(+5V)
$\overline{WE}$	写允许	GND	地
$\overline{CS}$	片 选		
$I/O_1 \sim I/O_4$	数据输入输出		

图 3-39 2114的引脚图

门导通,信号由数据总线写入存储器;当$\overline{WE}$为高电平时,则输出三态门打开,从存储器读出信号,送至数据总线。

若要用 2114 芯片构成 8 位的存储器,则需要把两片重叠起来使用,其中一片作为 8 位存储器的低 4 位,另一片作为高 4 位,两片的片选信号$\overline{CS}$并联在一起表示同时选中。

三、只读存储器(ROM)

1. 固定只读存储器 ROM

固定只读存储器是采用掩膜工艺制成,因此也称为掩膜只读存储器。它是由制造厂制成,用户不能加以修改。这类 ROM 可由二极管、双极型晶体管或 MOS 电路构成,其工作原理类似。

图 3-40 是一简单的 4×4 位的 MOS 管 ROM,采用单译码方式,2 位地址输入,经译码后,四条输出选择线,每一条选中一个字,位线输出即为这个字的各位。

图 3-40 中,若地址信号为 00H,选中第一条字线,则它的输出为高电平。若该字线上某位有 MOS 管(如位线 1),则相应的 MOS 管导电,于是该位线输出为"0",而位线 0、位线 2 和位线 3 没有 MOS 管与字线相连,则输出为"1"。一般情况下,在 MOS 管 ROM 输出线上加一反相器,使输出与存入信息相同。即当某一字线被选中时,连有管子的位线输出为1,而没有管子相连的位线输出为 0。图 3-40 中的存储矩阵的内容如右表。

字 \ 位	位 0	位 1	位 2	位 3
字 0	0	1	0	0
字 1	1	0	1	0
字 2	0	0	1	0
字 3	1	1	1	1

从图中也可看到 ROM 有一个重要特点:它所存储的信息不会丢失,即当电源掉电后又上电时,存储信息是不变的。

图 3-41 为一个 1024×1 位的复合译码结构的 MOS 管 ROM 电路。10 条地址信号线分成两组,分别经 X 和 Y 地址译码,各产生 32 条选译线。X 译码输出选中某一行,Y 译码输出选择某一列。$T_0 \sim T_{31}$为选择控制门。当某一行某一列选中,则交叉处为选中单元,所存信息经反相后输出。

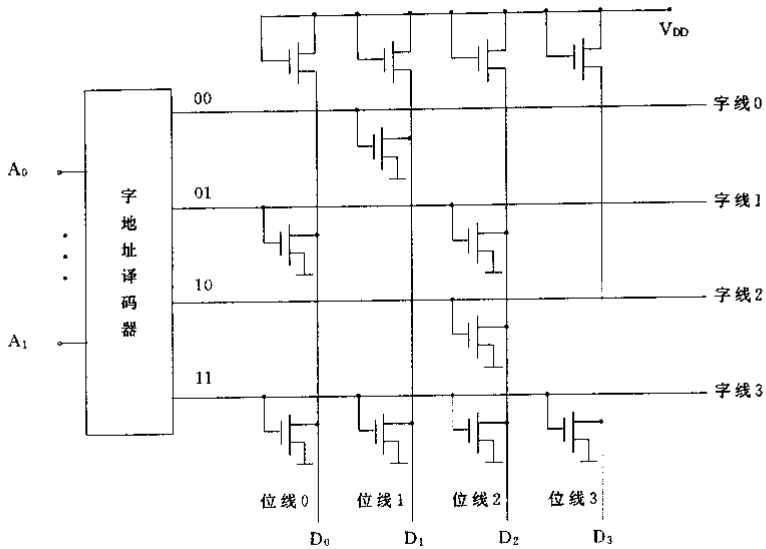


图3-40　4×4位 MOS管RAM图

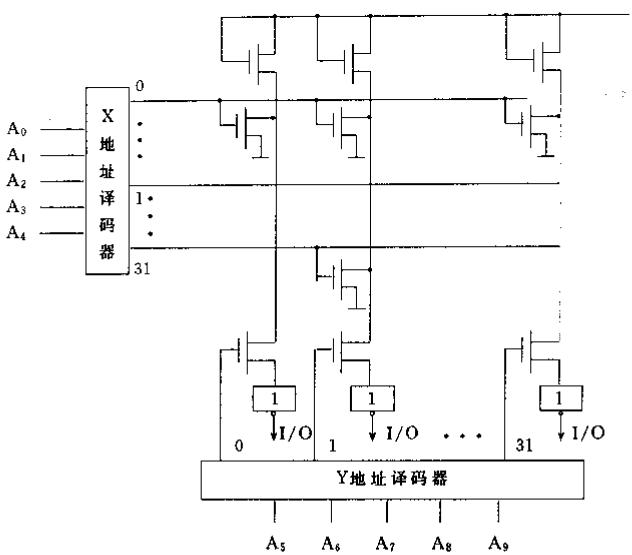


图3-41　复合MOS管ROM电路

2. 可编程只读存储器(PROM)

固定只读存储器ROM中的内容是厂方在生产时已经存入。为了便于用户根据自己的需要确定ROM中的存储内容，生产了一种可编程序的只读存储器(PROM)。图3-42是一种32×8的熔丝式PROM图。该PROM采用了单译码结构，共有32个字，每一个字8位，实际上是一个多发射极(8个)管，每个发射极通过一个熔丝与位线及读写控制电路相连，集电极接电源，基极接字选择线。图中共有32个多发射极晶体管。管子工作在射极输出器状态，当它被选中时，基极为高电位，则射极输出也为高电位，故若有熔丝存在，位线有电流输出，经T_1管反相后输出低电平，即读出"0"；若熔丝烧断，位线没有电流输出，T_1管不导通，输出高电平，即读出"1"。在出厂时，熔丝全部存在，用户可根据自己的需要烧断，以存入"1"来实现编程。

编程时(写入)，V_C接+12V，要写入"1"的位D端断开，要写入"0"的位D端接地。写入"1"时，D为高电平，稳压管DW导通，T_2导通，发射极流过足够大的电流把熔丝烧断；写入"0"时，D为低电平，稳压管DW不导通，T_2管不导通，发射极无电流流过，熔丝不被熔断。

正常读出时，V_C接+5V，稳压管DW不导通，T_2截止，位线上的信号经T_1管反相后输出。显然这种PROM只能写一次。

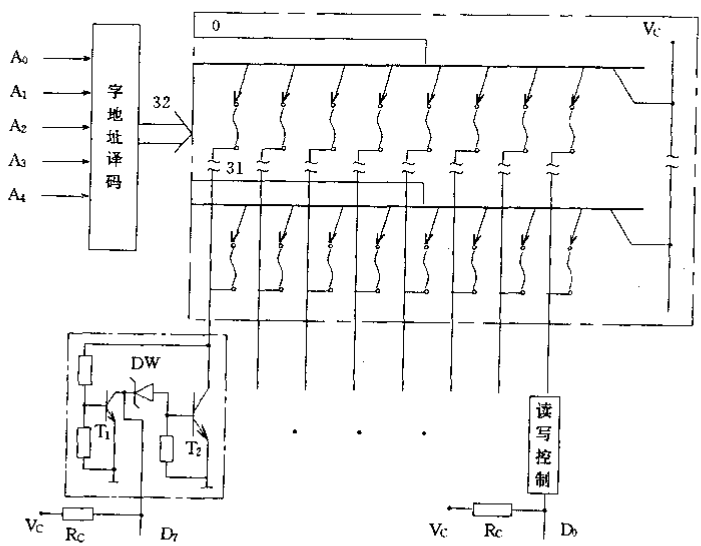


图3-42 熔丝式PROM电路

3. 可改写的只读存储器(EPROM)

上述可编程序只读存储器(PROM)只能进行一次性写入，无法实现对其中的内容进行修改。则为了能对只读存储器的内容进行修改，就发展生产了一种可改写的只读存储器

(EPROM)。

EPROM 的基本电路如图 3-43 所示。它与普通的 P 沟道增强型 MOS 电路相似，在 N 型的基础上生长了两个高浓度的 P 型区，它们通过欧姆接触，分别引出源极(S)和漏极(D)，在 S 和 D 之间有一个由单晶硅做的栅极，但它是浮空的，被绝缘物 SiO_2 所包围。在制造时，硅栅上没有电荷，则 MOS 管内没有导电沟道，D 和 S 之间是不导电的。用 EPROM管作存储矩阵时，一个基本存储电路如图 3-43(b)所示，则这样的电路组成的存储矩阵输出全为 1。

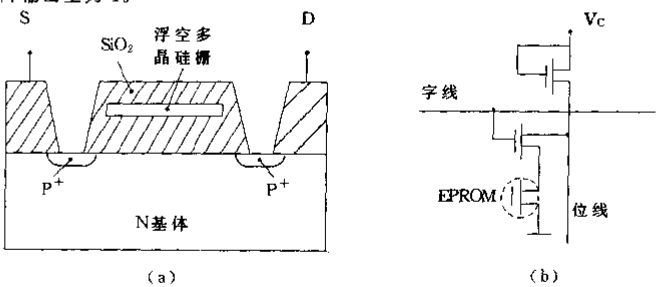


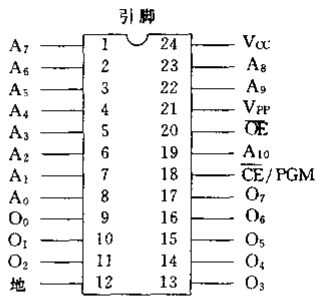
图 3-43　P沟道EPROM结构示意图

写入时，在 D 和 S 之间加 25V 的高电压，选中单元在这个电源作用下，N 和 S 之间被瞬时击穿，发生雪崩效应，有电子通过绝缘层注入到硅栅。当高电压除去后，因为硅栅被绝缘层包围，注入到浮空栅极中的电子被绝缘层包围而长期保存下来，形成了导电沟道，从而 EPROM 使单元导通，输出为“0”。

改写时，首先擦除，然后再重新写入。目前所用的擦除方法是紫外线照射。一般在 EPROM芯片的上方有一个石英玻璃窗口，当紫外线照射时，浮空栅极中的电子形成光电流而泄漏掉，使电路恢复到初始状态。擦除后即可重新写入。为了可靠地写入，一般要求编程脉冲的宽度为 50ms。写入后，为了不使存储信息丢失，芯片玻璃窗口应用不导光薄片盖住，以免受到光照。

例如 Intel 2716 为 2K×8 位的 EPROM，它的引脚及内部方框如图 3-44 所示。正常工作使用单一电源 +5V。

在 2716 中，基本存储单元排列成 128×128 的方阵。在 X 方向有 128 行，由 7 位地址经译码后进行选择；Y 方向也有 128 列，每 8 列为一组，用以形成一个字的 8 位，共有 16 组，由 4 位地址经译码后进行选择。输出端有输出缓冲器。读出时，行地址经译码后选中 X 方向的某一行，列地址经译码后选中 Y 方向的某一组。在 I/O 控制门的控制下，读出信息(8 位)送入输出缓冲器。片选信号$\overline{CE}$用来进行芯片选择，只有当$\overline{CE}$为低电平(有效)，芯片才被选中工作。$\overline{OE}$为输出控制端，低电平时数据输出。若要改写，首先擦除，使之成为全“1”状态。在写入时 V_{PP}接 25V 高电压，$\overline{OE}$为高电平，给出地址和要写入的 8 位数据后，由$\overline{CE}$/PGM 端输入一个宽度为 50ms 的正脉冲，即可完成写入。



引脚名

$A_0 \sim A_{10}$	地址
$\overline{CE}$/PGM	片选／编程
$\overline{OE}$	输出允许
$O_0 \sim O_7$	输出

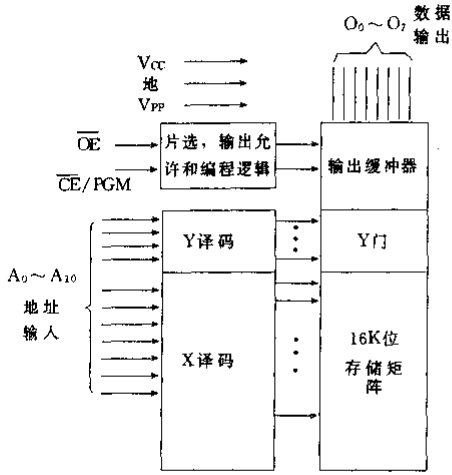


图3-44　2716方框图

第五节　模型计算机的工作原理

在我们开始接触计算机内部结构时，一个实际的微型机结构就显得太复杂了。如果从一个实际的微型计算机出发来讲解其工作原理，会使人们不知所措，很难弄清基本部件、基本概念和基本工作原理。所以，我们先从一个以实际结构为基础，经过简化抽象的模型机，着手来分析基本原理，逐步完善实际微型计算机的全貌。

一、模型计算机结构

图3-45是微型计算机的结构图。它是由微处理器(CPU)、存储器、接口电路组成，通过接口电路再与外部设备相连接。相互之间通过三条总线(BUS)——地址总线(Address Bus)、数据总线(Data Bus)和控制总线(Control Bus)来连接。有时也把数据总线地址总线统称为W总线。

本节我们介绍的模型计算机类同于微型计算机的结构，只是比较简单，如图3-46所示，且具有如下特点：功能简单——只能做两个数的加法；内存容量小——只有一个16×8位的RAM；字长8位。

(1)算术逻辑部件ALU：一个二进制补码加法器，当$E_U=1$时，能将其结果送到W总线上。

(2)累加器A：用以存储计算机运行期间的中间结果。当$L_A=1$时，它能接收W总线送来的数据；当$E_A=1$时，能将数据送到W总线上去。它还有一个数据输出端，将数据送

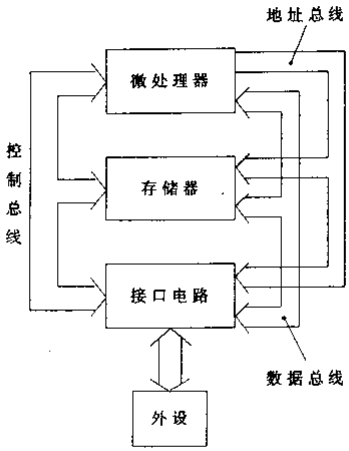


图3-45 微型计算机的结构图

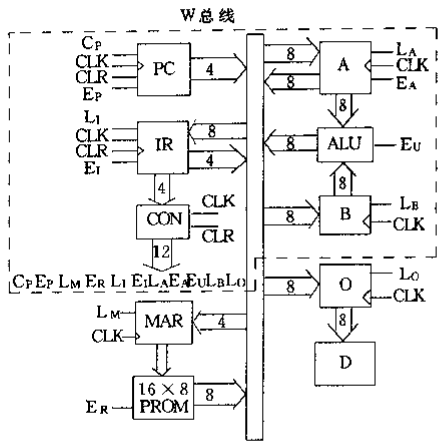


图3-46 模型计算机的结构图

到 ALU 去进行算术运算，这个输出不受 E 门的控制。

(3)寄存器 B：将要与 A 相加减的数据暂存于此寄存器。它到 ALU 的输出也是双态的，不受 E 门的控制。

(4)程序计数器 PC：要执行指令的地址由程序计数器 PC 提供。计数范围为 0000～1111(用十六进制可记作由 0～F)。每次运行之前，先复位至 0000。当取出一条指令后，PC 自动加 1。

(5)指令寄存器 IR：IR 从存放程序的存储器中接收到指令字(当 $L_I=1, E_R=1$)，同时将指令字(8 位)分成高 4 位和低 4 位，分别送到控制部件 CON 和 W 总线上去。

指令字是 8 位的：

X X X X　X X X X

高有效位　低有效位

左 4 位为高有效位，称为指令字段；右 4 位为低有效位，称为地址字段。

(6)控制部件 CON：其功能为：

①每次运行之前，CON 先发出 CLR=1，使有关的部件清 0。此时：

PC=0000　IR=0000 0000

②CON 有一个同步时钟，能发出脉冲 CLK 到各个部件去，使它们同步运行。

③在CON 中有一个控制矩阵 CM，能根据 IR 送来的指令发出 11 位的控制字：

$CON=C_PE_PL_ME_R\ L_IE_IL_AE_A\ E_UL_BL_O$

根据控制字中各位的置 1 或置 0 情况，计算机能自动地按指令程序而有秩序地运行。

(7)输出寄存器 O：计算机运行结束时，累加器 A 中存有结果，如果输出此结果，就得

送入O。此时$E_A=1$,$L_O=1$,则O=A。

(8)存储地址寄存器MAR:接收来自PC的二进制数据,作为地址码送至RAM中去,以选中RAM中存放程序的某单元。

(9)存储器RAM:存储器结构如图3-47所示。它由16个单元组成,为了能区分不同的单元,对这些单元分别编号:用1位十六进制数表示,这就是它们的地址,如0、1、2、…、F等。每一个单元可存放8位二进制信息,这就是它们的内容。每一个存储单元的地址,和这一个地址中存放的内容这两者是完全不同的,千万不要混淆。

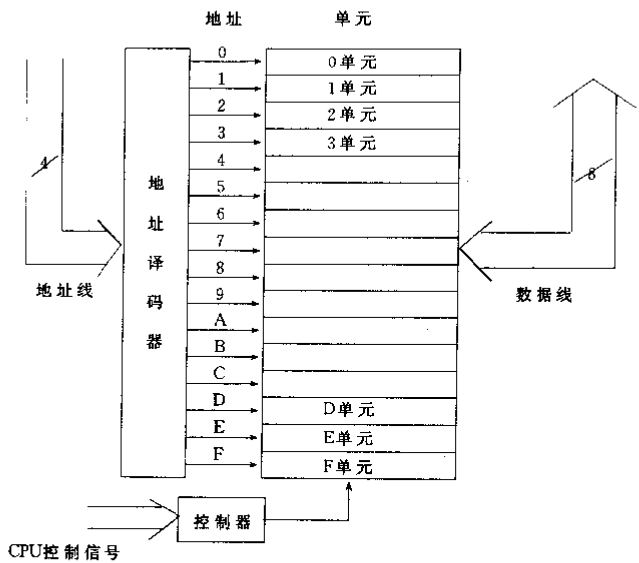


图3-47 模型机存储器示意图

存储器中的不同存储单元,是由地址总线上送来的地址(4位二进制数),经过存储器中的地址译码器,形成16个选择信号中的某一个来选择16个单元中的某一个单元,然后就可以对这个单元的内容进行读或写操作。

1. 读操作

若已知在04号存储单元中,存的内容为1000 0100(即84H),要把它读出至数据总线上,则要求CPU的地址寄存器先给出地址号0100,然后通过地址总线送至存储器,存储器中的地址译码器对它进行译码,找到04号单元;然后要求CPU发出读的控制命令,于是04号单元的内容84H就出现在数据总线上,送至相应的地方,如图3-48所示。

2. 写操作

若要把26H数据写入到08号存储单元,则要求CPU的地址寄存器MAR先给出地

址 1000，通过地址总线送至存储器，经译码后找到 08 号单元；然后把 26H 数据经数据总线送至存储器，且 CPU 发出写的控制命令，于是数据总线上的信息 26H 就可以写入到 08

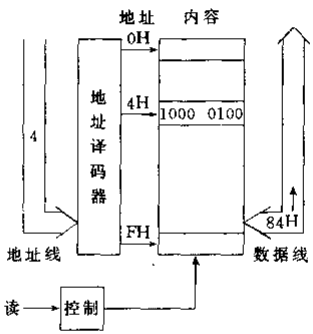


图3-48 存储器读操作示意图

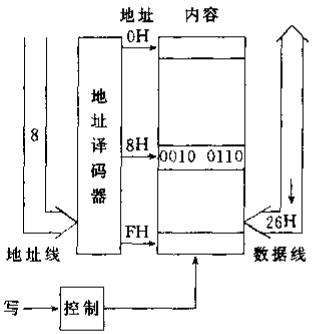


图3-49 存储器写操作示意图

号单元中，如图 3-49 所示。信息写入后，在没有新的信息写入以前是一直保留的。

模型计算机也具有微型计算机的三大部分：

(1)中央处理机 CPU(包括 PC、IR、CON、ALU、A 及 B)；

(2)记忆装置 M(MAR 及 RAM)；

(3)输出装置(O 接口)。

中央处理器是将程序计数功能(PC)、指令寄存功能(IR)、控制功能(CON)、算术逻辑功能(ALU)以及暂存中间数据功能(A 及 B)集成在一块电路芯片上。实际的 CPU 要比这里的模型机 CPU 复杂，但其主要功能是基本一样的。

存储器在此例中只包括存储地址寄存器(MAR)及随机存储器(RAM)，这就是微型计算机的“内存”。实际的“内存”要包括更多的内容(如 ROM、RAM 及 EPROM)和更大的存储容量。

输入/输出接口是计算机实行人-机对话的重要部件。实际微型计算机的输入设备多为键盘，输出设备则为显示器，因而必须有专用的输入/输出接口电路。

二、模型机的指令系统及程序设计

指令系统就是用来编制计算程序的一个指令集合。在未编制出计算程序之前，计算机是一堆无价值的电路硬件。

这台模型机有四条指令，即其控制部件能完成一系列例行程序(由厂家编好的程序)以执行四种命令：

LDA——将数据装入累加器 A；

ADD——进行加法运算；

OUT——输出结果；

HLT——停机。

这四条指令在一起就称为这台计算机的指令系统。

不同型号的微处理机的指令系统是不同的，指令的条数也不相同，如 Z80 型的指令系统可达 158 条，而 Intel 80386 则为 150 条，MCS-51 单片机为 111 种。

例如一个计算程序的格式如下：

```
LDA   R9        ;把 R9 中的数据存入 A
ADD   RA        ;把 RA 中的数据与 A 相加，结果放于 A 中
ADD   RB        ;把 RB 中的数据与 A 相加，结果放于 A 中
OUT             ;输出 A 中的数据，即结果
HLT             ;停机
```

这样的格式称为用汇编语言写的汇编语言程序。最左边的符号称为助记符，中间的符号 R9、RA、RB 等称为操作数，在";"之后的称为注释，每一行就是一条指令。

执行第一条指令的结果：(A)=(R9)；

执行第二条指令的结果：(A)=(R9)+(RA)；

执行第三条指令的结果：(A)=(R9)+(RA)+(RB)；

执行第四条指令的结果：(O)=(A)；

执行第五条指令的结果：CLK 停止发脉冲，结果保持不变。

上面加括号的意义是指被括上的寄存器或存储单元的内容，如(A)是指累加器 A 中的内容。

由于计算机只认识"0"和"1"，而不认识助记符的符号，因此必须将指令清单中每一条指令都翻译成二进制码——机器码。每个助记符与二进制码的相应对照表(操作码表)由计算机制造厂提供。由于我们的模型计算机很简单，只有四个助记符，形成对照表(表 3-5)比较容易记忆。但如果指令系统很庞大，此表也就很大且不容易记忆，因此就必须有特殊的方法才能迅速可靠地使用它。

表 3-5

助记符	操作码
LDA	0001
ADD	0010
OUT	1000
HLT	1111

表 3-6 存储器分配

	指令区	数据区
存储单元	R0～R7	R8～RF
二进制地址	0000～0111	1000～1111
十六进制地址	0H～7H	8H～FH

本模型机中共有 16 个存储单元，这 16 个存储单元分成两个区：程序存放区(指令区)和数据存放区(数据区)，其分配如表 3-6 所示。

根据助记符与二进制的对照表将上例中的每条指令的助记符译成二进制码，并将存储单元符号写成地址码(即 R9→1001、RA→1010、RB→1011)就形成下面的样子。

这个例题在输入这些数据之后，就是要求演算这样一个具体算术题：

存储单元	地址	目的程序(存储单元) 指令字段 地址字段	源程序(或数据)
R0	0000	0 0 0 1 1 0 0 1	(LDA R9)
R1	0001	0 0 1 0 1 0 1 0	(LDA RA)
R2	0010	0 0 1 0 1 0 1 1	(ADD RB)
R3	0011	1 0 0 0 ××××	(OUT)
R4	0100	1 1 1 1 ××××	(HLT)
R5	0101	×××× ××××	
R6	0110	×××× ××××	
R7	0111	×××× ××××	
R8	1000	×××× ××××	
R9	1001	0 0 0 1 0 0 0 0	(16)
RA	1010	0 0 0 1 0 1 0 0	(20)
RB	1011	0 0 0 1 1 0 0 0	(24)
RC	1100	×××× ××××	
RD	1101	×××× ××××	
RE	1110	×××× ××××	
RF	1111	×××× ××××	
		$D_7D_6D_5D_4$ $D_3D_2D_1D_0$	

O=16+20+24=?

在前面例题的计算程序设计好并输入至 RAM 中，就可以开始执行程序了。程序执行的第一步必须先使计算机复位，此时控制器先发出一个 CLR 为高电位的脉冲，同时时钟脉冲开始工作，在控制部件指挥下，一条一条执行上例指令，得到结果(O)=16+20+24=60。

三、模型机指令的执行过程

在程序和数据装入之后，启动按钮将启动信号传给控制部件 CON，然后控制部件产生控制字，以便取出和执行每一条指令。

执行一条指令的时间为一个机器周期，机器周期又可分为取指周期和执行周期。取指周期和执行周期都得通过不同的机器节拍。

1. 机器节拍产生

机器节拍的产生是通过环形计数器来完成的，如图 3-50 所示。

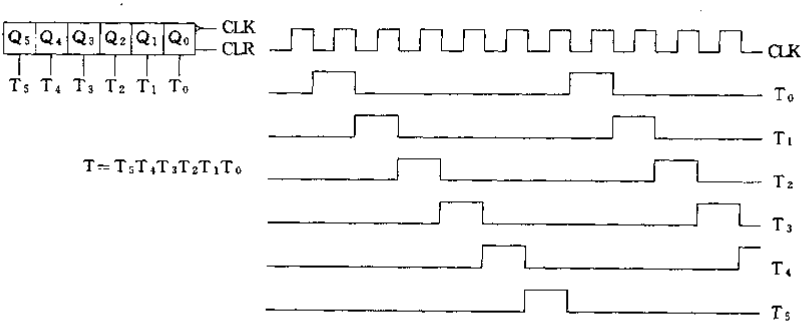


图3-50 机器节拍的产生

环形计数器的各位输出端 $Q_0 \sim Q_5$ 的电位就是机器节拍 $T_0T_1T_2T_3T_4T_5$ 的电位，它们依次轮流为高电位，形成环形字，用以控制六条电路。

2. 取指周期及执行周期

取指周期的过程需要三个机器节拍。

(1)地址节拍：在 $T_0=1$ 时，应将 PC 的内容(即第一个地址码)送入 MAR，从而选择 RAM 指定单元。此时有

$E_P=1$，即 PC 准备放出数据

$L_M=1$，即 MAR 准备接收数据

因此，控制部件应发出的控制字为

$$CON=C_PE_PL_ME_RL_IE_IL_AE_AE_UL_BI_O$$
$$=0110\ 0000\ 000$$

(2)储存节拍：在 $T_1=1$ 时，应将 RAM 中由 PC 送来的地址码指定单元中的内容送到指令寄存器 IR 中，同时 IR 立即将其高 4 位送至控制部件。因此有

$E_R=1$，即 RAM 准备放出数据

$L_I=1$，即 IR 准备接收数据

$CON=0001\ 1000\ 000$

(3)增量节拍：在 $T_2=1$ 时，应使 PC 加 1，做好下一条指令取指准备。因此，$C_P=1$，即命令 PC 计数。所以

$CON=1000\ 0000\ 000$

这三节拍称为取指周期。这样三节拍的取指周期，对任何一条指令都是一样的。因此任何一条指令都是沿着这个程式而将指令取出来，再将其高 4 位送入控制部件去进行分析，决定下面应如何执行。

执行指令过程也需要三拍(T_3、T_4、T_5)。以 LDA 为例，考察一下在此三拍中，各个寄存器的内容应有何变化。

(4)在 $T_3=1$ 时，IR 已将从 RAM 中取来的指令码的高 4 位送至控制部件进行分析，此高 4 位是与 LDA 相应的二进制码"0001"，控制部件经过分析后就发出命令：

$E_I=1$，将 IR 的低 4 位送至 W 总线

$L_M=1$，MAR 接收此低 4 位数作为地址并立即送至 RAM

所以 $CON=0010\ 0100\ 000$

如 LDA R9，送至 RAM 的地址为 R9 的二进制码地址(1001)。也就是说，第一次访问 RAM 的是其指令区，第二次的是其数据区。

(5)在 $T_4=1$ 时，应将 RAM 的数据区的存储单元(如 R9 1001)的内容送入累加器 A，即

$E_R=1$，即 RAM 准备放出数据

$L_A=1$，即 A 准备接收数据

即 $CON=0001\ 0010\ 000$

(6)在 $T_5=1$ 时，由于 $T_4=1$ 时已将数据存放于 A 中，所以 LDA 的例行程序已完成，此节拍变成空拍，即

CON=0000 0000 000

为什么需要这个空拍呢？这是因为虽然 LDA 的例行程序用不着这个节拍，但其它例行程序（如 ADD）需要，为了使每条指令的机器周期都是一样长，即六个节拍，所以，在不需六个节拍的指令语言中都加上空拍。

执行 ADD R9 指令其取指周期与前面一样，只是现在存于 PC 中的内容已不是0000，而是(PC)+1，即 0001 了。执行周期中各有不同，T_3 节拍同样要求从 IR 将低 4 位的数据作为地址码送到 MAR 上，其控制字为 $L_M=1$ 及 $E_I=1$。但 T_4 节拍就不同了，从 RAM 中来的数据不用送入累加器 A，而是送入寄存器 B，这样 A 和 B 的数据就能直接被送入 ALU 中相加。在 T_5 节拍，不再是空拍，因为还需要将 A 和 B 的内容相加的结果送回到 A 去，所以要求 $L_A=1$，$E_U=1$。

执行 OUT 指令要求将累加器的内容送入输出寄存器，而与存储器 RAM 无关，所以只在 T_3 节拍要求 $L_O=1$，$E_A=1$，而 T_4 及 T_5 节拍为空拍。

执行 HLT 指令，在 T_3 节拍发出 HLT 后，T_4 及 T_5 节拍也是空拍。

四、控制部件

控制部件是使计算机能够成为自动机的关键部件。控制部件包括下列主要环节：环形计数器(PC)，指令译码器(ID)，控制矩阵(CM)——称为控制器，其它控制电路。

环形计数器用以发出环形字，从而产生机器节拍，其原理前面已经讲过。

1. 指令译码器(ID)

指令寄存器 IR 中的数据的高 4 位立即送入控制部件，这高 4 位就是各种控制动作的代码。比如：

0001——代表 LDA 的控制动作；

0010——代表 ADD 的控制动作；

1000——代表 OUT 的控制动作；

1111——代表 HLT 的控制动作。

一个动作相当于一条控制线，要该动作实现，就必须使该控制线为高电位。将 IR 中的高 4 位 $I_7I_6I_5I_4$ 组成编码，4 位最多可以组成 16 种编码，我们按照上述要求，取其中四个即可，如图 3-51 所示。

2. 控制矩阵(CM)

这里介绍的模型机具有四条指令，每条指令的每一节拍大都要求两个控制字为高电位，虽然也有 1 位为高电位的（如 $C_P=1$），但只要不是 1 位的就要求 2 位为高电位。而控制字为 11 位，每一指令要执行六拍，每拍均有不同的位为高电位，这又如何实现呢？

控制矩阵就是为了解决这个问题的，所以控制矩阵是控制部件的核心，图 3-52 是这样一个控制矩阵(CM)的功能图。

开机前总是先使 CLR 为高电位，则此时环形计数器复位至 $T_0=1$，其它各位为 0，就是说每一节拍都是从 T_0 开始的。

3. 其它控制电路

为了实现控制动作，还需要下述几个电路（如图 3-53）：

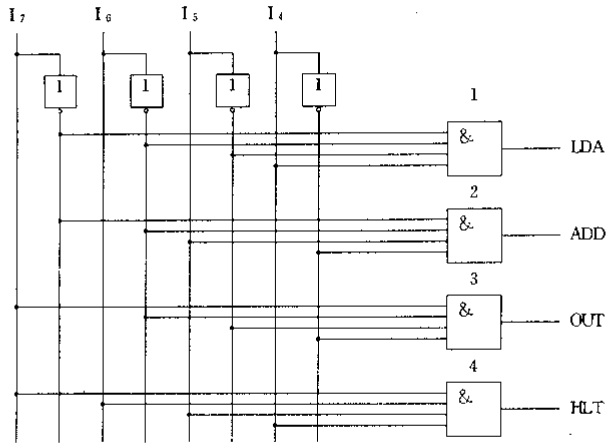


图 3-51　4 位译码器电路

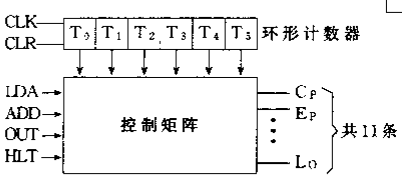


图 3-52　控制矩阵(CM)

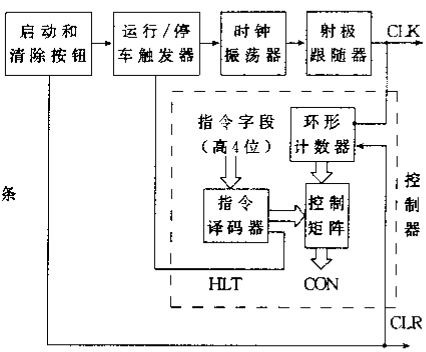


图 3-53　控制部件的结构图

(1)时钟脉冲发生器:由时钟振荡器及射极跟随器组成,射极跟随器主要用来降低输出电阻,以便有更大的电流,同时推动很多的电路。

(2)运行/停止触发器:这个电路既接收来自按钮的"运行"脉冲信号,也接收来自指令译码器的"HLT"停止信号,而其输出去启动时钟振荡器。

(3)"起动"和"清除"按钮:这是由人直接操作的主令信号,命令都是由此开始的。

思考题与习题

1. ALU 是什么部件,它能完成什么运算功能? 试画出其符号。
2. 触发器、寄存器、存储器之间有什么关系,请画出这几种器件的符号。
3. 试叙述 JK 触发器、环形计数器、程序计数器的功能,并画出它们的符号。

4. 三态输出电路有何意义？

5. 何为L门和E门，在总线结构中有什么用处？

6. ROM 和 RAM 的特点及用处？

7. 除地线公用外，5 根地址线和 11 根地址线各可选多少个地址？

第四章　单片微型计算机的组成原理

在第三章我们介绍了模型式计算机的结构、指令系统，本章我们将介绍实际应用的微处理器、微型计算机的结构，主要介绍 MCS-51 单片机结构、组成原理及功能。

第一节　微型计算机的结构及指令执行过程

一、微型计算机结构

微型计算机是计算机的微型化，它的基本结构同样也是由运算器、控制器、存储器及输入输出五部分组成，只是大规模集成电路工艺的发展，使人们能把过去用电子管、晶体管组成的庞大繁杂的线路压缩到 $5\times5mm^2$ 大小的芯片上，从而使制造成本大大下降，价格大大降低，运行可靠性大大提高。

图 3-45 已经给出了微型计算机的基本结构，其中微处理器 CPU 是原计算机中的运算器和控制器的集成，它是计算机系统的核心，也称之为中央处理单元(Center Pressing Unit)。这种把运算器和控制器集成在一个芯片上的 CPU，不同厂家有不同的结构和不同的型号，例如 Intel 公司的 8080、8086/8088、80286、80386、80486、Pentium 586；Zilog 公司的 Z80、Z8000；Motorola 公司的 6800、68000。

在微处理器内部，运算器、控制器及一个包括若干寄存器的寄存器组通过内部总线互相连接。在微处理器外部，微处理器通过外部总线与存储器、输入输出接口电路及必要的输入输出设备连接，构成一台微型计算机。

由于微型计算机采用总线结构，故可以减少各部件之间信息传送线的条数，从而使信息传送规格整齐，提高了整机的可靠性。采用总线结构以后，存储器、输入输出等外部设备都通过专门的接口电路独立地挂在总线上，因而使得微型计算机可扩充能力强，使用方便灵活。另外，可根据不同的需要增加存储器的容量或增添外部设备，也可以根据不同的需要组成各种专用微型机，例如专用教学机、专用控制机等。

微型计算机采用大规模集成电路组成，各组成部件可以方便地装配在一块印刷电路板上，构成单板机。由于结构简单、体积小、携带方便、抗干扰能力强、工作可靠、价格低廉、易于扩充使用，单板机在科研、国防、工业、农业、医疗卫生以及教学等各个方面得到了广泛应用。

二、单片微型计算机结构

上面所述的微型计算机是由中央处理器 CPU、存储器、输入输出接口电路等多块集成电路芯片组成，因此称为多片微型计算机，简称多片机。所谓的单片机是把中央处理器

CPU、存储器 ROM/RAM、输入输出接口电路以及定时器/计数器等部件制作在一块集成电路芯片中，构成一个完整的微型计算机——单片微型计算机，其组成框图如图 4-1 所示。由于单片机把各功能部件集成在一块芯片上，因此它结构紧凑、超小型化、可靠性高、价格低廉、易于开发应用。

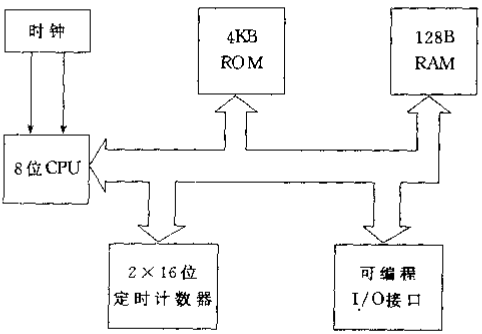


图4-1　单片微型计算机组成原理框图

下面着重分析单片机的性能特点：

(1)集成度高。在单片机芯片中，除中央处理器 CPU 之外，还有存储器 ROM/RAM，I/O接口电路、定时器/计数器等部件，因此集成度高，在几至几十平方毫米的芯片上可制作上万个晶体管电路。

(2)结构紧凑、可靠性高。单片机把各功能部件集成在一块芯片上，采用内部总线结构，减少了多片机中各芯片之间的连线，大大提高了单片机的抗干扰能力。另外，单片机超小型化、结构紧凑、体积小，对于强磁场环境易于采取屏蔽措施，因而抗干扰能力强，可靠性高，适合于在一些恶劣环境中工作。

(3)数据处理能力强、速度快。单片机除具有一般微处理器的数据处理能力外，在一系列产品(如 MCS-51)的指令系统中，增加了乘除法指令及布尔(二进制)处理机功能，提高了数据处理能力。同时，由于中央处理器与存储器在同一芯片上，因而减少了多片之间数据传送所需时间，提高了数据处理速度。例如 MCS-51 的 CPU，采用 12MHz 时钟时，单字节乘除法仅需 6.5μs。

(4)功耗小、成本低。单片机结构紧凑，数据传送路径短，所需要功耗小；内部采用准静态 RAM，结构与动态 RAM 类似，但不需要刷新，可使功耗下降。

单片机内部电路虽然比相应微处理器芯片复杂，但是一旦设计好后，进入批量生产，成本不会提高。单片机内部设置一定容量的只读存储器 ROM/EPROM，用于存储用户的专用程序，这些程序称之为内部程序。内部程序可由厂方在制作芯片时代为烧制，也可由用户自己写入，这样可使单片机成为具有不同特殊功能的专用机，易于形成产品。

由于单片机具有以上优点，若配以适当的外围设备，构成单片单板机，则功能更强，应用范围更广。目前，单片机发展很快，大有取代当年单板机之势，成为计算机发展的一个重

要方面。

三、指令执行过程

对于大多数计算机用户来说，并不需要十分详细地了解 CPU、存储器、输入输出接口等芯片的具体线路。为了能比较清楚地理解微型计算机的工作原理，即执行指令的过程，这里给出一个较详细的微型计算机框图，见图 4-2。设 CPU 为 8 位中央处理器。

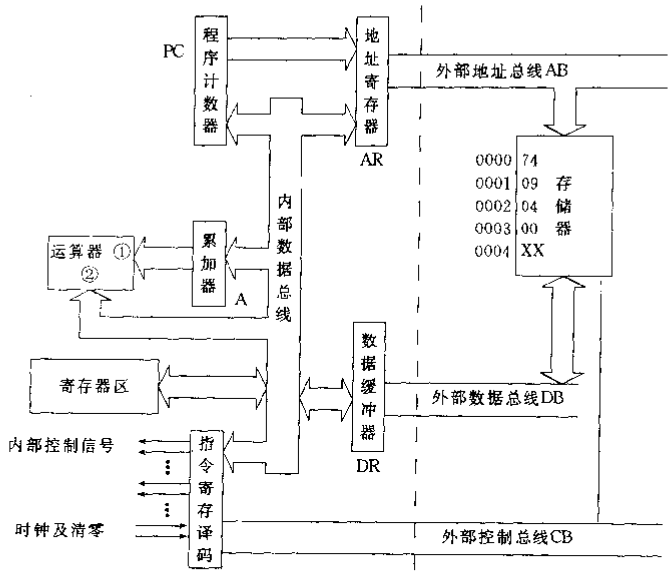


图4-2　指令执行经过示意图

图 4-2 中虚线左边部分是中央处理器 CPU 的框图，其中运算器是执行数据运算的部件，基本电路就是第一章中介绍的加法器。它有两组输入，一组(8 位)是 A 累加器的内容，另一组是数据总线上的内容，它的输出又送至内部数据总线上。

(1)程序计数器 PC(16 位)：PC 是一个自动加 1 计数器，它提供要执行指令的地址。它的内容也可以通过内部数据总线得到修改。

(2)地址寄存器 AR(16 位)：AR 的输入可以是程序计数器的内容，也可以是内部数据总线的内容。它的输出送到 CPU 外边，作为存储器或输入输出接口的某地址值。

(3)数据缓冲器 DR(8 位)：DR 起 CPU 内外数据传送缓冲的作用。

(4)指令寄存译码器：其基本电路是译码器。它的输入是内部总线上的内容(一般在取指周期得到的指令代码)和外部时钟脉冲清零信号等，输出则是各种分时的“0”、“1”电平和脉冲。它根据输入指令代码的不同，发出一系列相应的“0 ”、“1”电平和脉冲，控制各部

分协调工作，完成该指令的执行任务。

(5)寄存器组：寄存器区是为暂存数据或其它特殊用途而设置的，将在后续章节中介绍。

图 4-2 虚线右边部分是一个存储器。存储器芯片的基本电路是存储体和译码器。在大规模集成电路的存储芯片中，存储体的数量是惊人的。现在常用的只读存储器(ROM)或随机存储器(RAM)都是每片数千个存储单元(8 位)，甚至更多。另外，对某一单元的读写，只要地址线给出该单元地址，控制线给出读写信号，即可由存储器内部译码器电路找到该单元，并且写入或读出该单元中的内容。

我们假设图 4-2 中的存储器已经同 CPU 接好了数据总线、地址总线、控制总线，并且已经通过输入输出设备在地址从 0000H 开始的单元里装好了一些指令代码：74 09 04 00。这些代码代表了某种命令，这将在指令系统中学习，这里为了说明，暂由我们给出：

74 09　把 09 这个数送到累加器 A 中

04　把 A 中内容加 1，送回 A 中

00　等待，空操作

现在我们来看看机器指令执行的经过。

开机时，我们先通过 CPU 外部清零线使程序计数器 PC 变成 0000H(也可以通过键盘来改变 PC 的值)，然后微型计算机在外部时钟脉冲作用下自动进入执行过程。

执行过程实际上就是微型计算机取指(取出存储器中事先存好的指令)阶段和分析执行指令阶段这两个阶段的循环过程。

像图 4-2 系统中，当机器进入运行时，首先是进入取指阶段，其顺序是：

①程序计数器的内容(这时是 0000H)送到地址寄存器。

②程序计数器的内容自动加 1(变为 0001H)。

③地址寄存器中内容(0000H)通过 CPU 外部地址总线送到存储器，经存储器中地址译码器，使地址为 0000H 的单元被选中。

④CPU 控制总线中读控制线等有效(产生一个低电平)。

⑤存储器中被选中单元内容(此时应为 74)送到数据总线上。

⑥通过数据总线送该内容到数据缓冲器。

⑦该内容送到内部数据总线。因为是取指阶段，所以一定是送到指令寄存译码器。

至此，取指阶段完成，进入执行阶段。

当机器进入执行阶段后，由于进入指令寄存译码器中的内容是 74H(指令代码)，机器就会知道该指令是要将一个数送到累加器 A 中，而该数是在此指令代码的下一个存储单元，所以执行该指令还必须把数(09H)从存储器中取到 CPU，即还要到存储器中取第二个数字(或称字节)，其过程与取指阶段①～⑦步类同，只是第①步中 PC 已为 0001H，从而第⑤步取出的数是 09H，而且第⑦步因为指令是要求把取得的数送到累加器 A，所以取出的内容由内部数据总线进入累加器 A，而不是进入指令寄存译码器。

至此一条指令执行完毕，CPU 中 PC=0002H(PC 在 CPU 每次向存储器取指或取数

时都自动加 1)，机器又进入了取指阶段。

取第二条指令的取指阶段过程与取第一条指令的取指阶段过程完全相同。第⑦步一定把从存储器取来的内容送到指令寄存译码器，只是此时的指令代码是 04H。

经指令分析知道，这是一个单字节指令，它要求把 A 的内容加 1 以后，结果取代原来 A 中的内容。所以，当机器进入执行阶段时，控制器(即指令寄存译码器)发出各种内部控制信号。执行过程是：

①把累加器 A 内容送入运算器。

②加 1 以后送到内部数据总线。

③内部数据总线上的数又送回累加器 A。

这样第二条指令又执行完毕，CPU 中 PC=0003H，机器又进入了第三条指令的取指阶段。按①～⑦取指过程从存储器中取得 00H，送到控制器。经分析译码，知道该代码是让 CPU 空操作。

至此，程序取指执行完毕。累加器 A 中数为 0AH(09H+01H=0AH)。

从上面介绍可以看出，机器执行程序时其过程如同交通管理，有各种通道、门岗。人们要传送一个数，对一个数进行运算，要让机器执行某一单元的指令代码，要机器暂停工作，都必须制定好各种指令的次序，然后机器在一定节拍中发出种种信号(如同交通路中的红绿灯)，有条不紊地进行控制执行。

第二节　MCS-51 单片计算机的组成原理

MCS-51 系列单片微型计算机包括 8031、8051、8751 等型号，其代表型号是 8051。8051 内部组成方框图如图 4-3 所示，内部总体结构框图如图 4-4 所示。

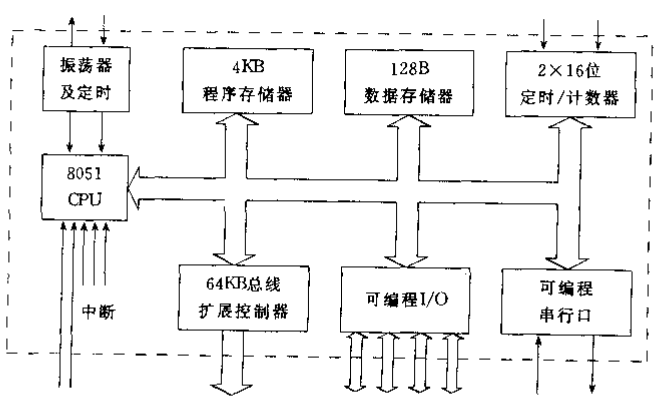


图4-3　8051单片机组成方框图

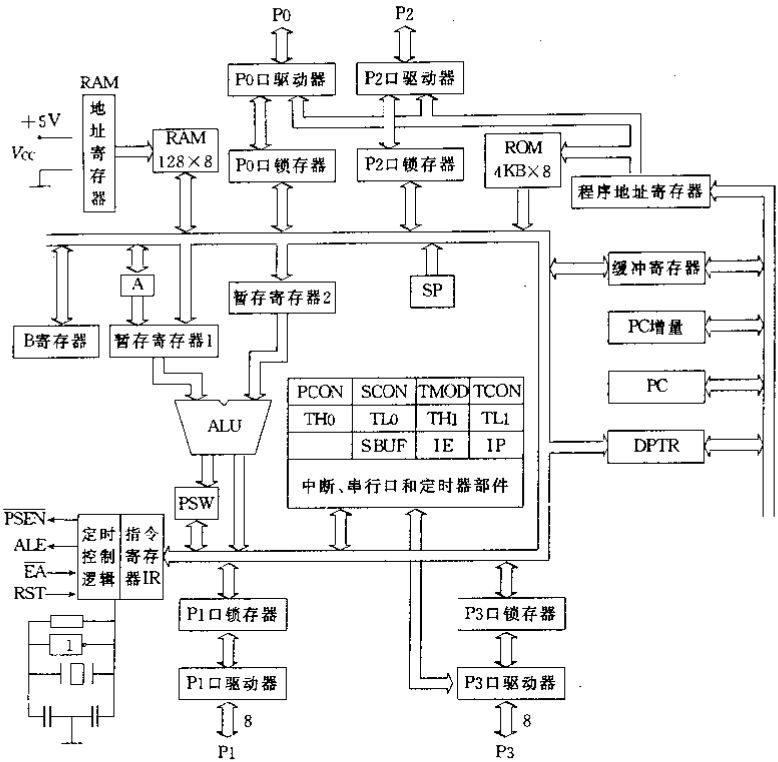


图4-4　8051总体结构框图

8051 主要包括算术/逻辑部件 ALU、累加器 A(有时也称 ACC)、只读存储器 ROM、随机存储器 RAM、指令寄存器 IR、程序计数器 PC、定时器/计数器、I/O 接口电路、程序状态寄存器 PSW、寄存器组，此外，还有堆栈寄存器 SP、数据指针寄存器 DPTR 等部件。这些部件集成在一块芯片上，通过内部总线连接，构成完整的微型计算机。下面按其部件功能分类予以介绍。

一、寄存器

微处理器中的寄存器是使用者应该十分重视的部分，因为读者在将来学习指令系统和程序设计中常会接触它们。

寄存器在第三章已经讲述过，它是由触发器组成的，8 位寄存器由 8 个触发器组成，16 位寄存器由 16 个触发器组成。MCS-51 中的寄存器较多，大体可分为通用寄存器和专用寄存器两类。

1. 通用寄存器

MCS-51 共有 32 个通用寄存器，均为 8 位寄存器，分为 4 个区，分别称为 0 区、1 区、2 区和 3 区，每个区内有 8 个寄存器，分别为 R0、R1、R2、R3、R4、R5、R6、R7。MCS-51 单片机每次只能选择一个寄存器区工作，因此尽管各寄存器区内的寄存器名称对应相同，也不会发生混乱。4 个寄存器区的选择是由程序状态字中 RS1、RS0 来控制的，其对应关系如下表。

RS1	RS0	寄存器区
0	0	0 区
0	1	1 区
1	0	2 区
1	1	3 区

2. 专用寄存器

MCS-51 系列有 20 个专用寄存器，其中 4 个 16 位寄存器。

(1)程序计数器 PC

程序计数器 PC 用于存放下一条要执行的指令地址(程序存储器地址)，是一个 16 位专用寄存器，它有自动加 1 的功能，微处理器用它来指出应该轮到执行的指令地址。当一条指令按照 PC 所指的地址从存储器中取出之后，PC 就会自动加 1。这意味着加 1 以后的 PC 的内容是下一条将要轮到执行的指令地址。所以，PC 是维持一个机器有秩序地执行程序的关键性寄存器。应该注意，有些指令代码并不只占一个存储器单元(有 2 字节、3 字节)，这时 PC 在 CPU 每次去存储器取一个数(可以是指令第一字节，也可以是该指令的第二字节、第三字节)时就自动加 1。当需要改变程序的执行顺序时，要使用转移指令，由转移指令给出转移后指令的起始地址，将此地址送到程序计数器 PC，即从新的转移后起始地址执行，然后程序计数器 PC 的内容自动加 1。

(2)累加器 A

累加器 A 是一个 8 位寄存器，是 CPU 中工作最频繁的寄存器。因为在进行算术逻辑类操作时，大部分单操作数指令的操作数取自累加器 A，很多双操作数指令的一个操作数取自累加器 A。加、减、乘、除算术运算结果都存于累加器 A 或 AB 寄存器对中。

(3)B 寄存器

B 寄存器是一个 8 位寄存器，用于存取乘除指令中的操作数。乘法指令的两个操作数分别取自 A 和 B，其结果存放于 AB 寄存器对中。除法指令中，被除数取自 A，除数取自 B，商数存于 A 中，余数存于 B 中。在其它指令中，B 寄存器可作为 RAM 中的一个单元来使用。

(4)程序状态字

程序状态字(PSW)是一个 8 位寄存器，它包含了程序状态信息。此寄存器各位的含义见图 4-5，其中 PSW1 是保留位，未用。

PSW7 PSW6 PSW5 PSW4 PSW3 PSW2 PSW1 PSW0

CY	AC	F0	RS1	RS0	OV	—	P

图 4-5 程序状态字

①CY(PSW7)：进位标志。在执行某些算术和逻辑指令时，可以被硬件或软件置位或清除(详见指令系统一章)。在布尔处理机中，它被认为是位累加器。例如进行二进制加法

运算(8 位)：

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

运算结果超出 8 位，产生进位，此时置位 CY(即 CY=1)，以表示二进制加法运算产生了进位。

②AC(PSW6)：辅助进位标志。当进行加法或减法操作时，如果产生低 4 位数向高 4 位数进位或借位时，AC 被硬件置 1，否则被清 0。AC 被用于十进制调整(详见 DA 指令)。

③F0(PSW5)：标志 0。是用户定义的一个状态标记。可以用软件来使它置位或清除，也可以靠软件测试 F0 以控制程序的流向。

④RS1、RS0(PSW4、PSW3)：寄存器区选择控制位 1 和 0。可以靠软件来置位或清除，以确定工作寄存器区。

⑤OV(PSW2)：溢出标志。当执行算术指令时，由硬件置位或清除，以指示溢出状态。

溢出与进位的概念不同。进位用于表示 8 位二进制加减法是否产生进位与借位，而溢出则主要用于表示有符号二进制数加减法的正确性，OV=1 表示加减法运算的结果超出了目的寄存器 A 所能表示的带符号数(−128～+127)。例如二进制加法运算+1011011 与+0111011 相加：

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

↑ 符号位

显然，两有符号正数相加，第 6 位向第 7 位进位，而第 7 位不向进位位进位。结果为负数，不能正确表示原两正数相加的结果，即产生溢出(91+59=150 超过+127)，置 1 标志位 OV。

又如二进制数−0001010 与−0011000 相加：

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

↑ ↑ 进位位 符号位

可以看出，该运算第 6 位向第 7 位进位，第 7 位又向进位位进位，但结果正确(−10−24=−34)。

再如−1100000 与−1111011 相加：

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

↑ ↑ 进位位 符号位

运算中，第 6 位没有向第 7 位进位，而第 7 位向进位位进位，运算结果为正数，显然是错误的(−96−123=−219 超出−128)。

综上所述，当第 6 位与第 7 位同时不向前进位或同时向前进位时，没有溢出，OV=0；当第 6 位与第 7 位只有一个发生进位，结果溢出，OV=1。

若以 C_i 表示位 i 向 i+1 有进位，则

$$OV=C_6\oplus C_7$$

在 MCS-51 中，无符号数乘法指令 MUL 的执行结果也会影响溢出标志。当累加器 A 和寄存器 B 的两个乘数的积超过 255 时，OV=1，否则 OV=0。由于此积的高 8 位放在 B 中，低 8 位放在 A 中，因此 OV=0 意味着只要从 A 中取得乘积即可，否则要从 AB 寄存器对中取得乘积。

除法指令 DIV 也会影响溢出标志。当除数为 0 时，OV=1，否则 OV=0。

⑧P(PSW0)：奇偶标志。每个指令周期都由硬件来置位或清除，以表示累加器 A 中值为 1 的位数的奇偶数。若值为 1 的位数为奇数，则 P 置位，否则清除。

此标志位对串行通信中的数据传输有重要的意义。

(5)栈指针寄存器 SP

它是一个 8 位的专用寄存器。所谓堆栈是在 CPU 外部存储器中一个按先进后出原则组织的存储区域。堆栈指针寄存器中的 8 位二进制数始终等于堆栈的顶部地址值。MCS-51 提供了一个向上升的堆栈，即每向堆栈中推入(可用 PUSH 指令)一个数据(8 位)，堆栈指针的内容加 1，指向刚推入数据存放的地址(最后推入的数据存于栈顶)，即栈顶地址；再推入(PUSH)一个数据，堆栈指针的内容再加 1，指向新的栈顶地址。同样可用弹出(POP)指令从堆栈中弹出一个数据，堆栈指针相应减 1，指向下一数据(栈顶)地址。利用压入和弹出指令，可以把 CPU 内部寄存器的内容推入堆栈或把堆栈中某单元的内容弹到内部寄存器中。利用堆栈指令可以简化寄存器与存储器之间数据传送的程序。

系统复位后，SP 初始化为 07H，使得堆栈事实上由 08H 单元开始。MCS-51 中栈指针寄存器的值可以由软件改变，因此堆栈在存储器中的位置比较灵活。

(6)其它专用寄存器

除上述的 5 个专用寄存器外，MCS-51 还有数据指针寄存器(DPTR)，端口寄存器(P0、P1、P2、P3)，串行数据缓冲器(SBUF)，定时/计数器(T0、T1)，及 IP、IE、TMOD、TCON、SCON、PCON 控制寄存器，这些寄存器将在后续章节中叙述。

二、运算器

运算器包括算术/逻辑部件(ALU)、累加器 A、暂存寄存器、程序状态寄存器(PSW)。运算器主要用来实现对操作数的算术/逻辑运算和位操作，其功能如下：

带进位和不带进位的加法、减法运算；

逻辑与、逻辑或和逻辑异或；

加 1、减 1 和位操作；

左移位、右移位和半字节移位；

BCD 码调整等。

三、控制器

控制器包括定时控制逻辑电路、指令寄存器 IR、指令译码器 ID 等，是微处理器 CPU 的大脑中枢。

CPU 从存储器中取来的指令，首先被送入指令寄存器寄存，使整个分析执行过程一直在该指令控制下。然后，指令寄存器中的指令代码被译码分析成一种或几种电平信号，这些电平信号与外部时钟脉冲在 CPU 定时与控制电路中组合，形成各种按一定时间节拍变化的电平和脉冲，即是控制信息，在 CPU 内部协调像寄存器之间的数据传送、数据运算等操作，对外部发出像对存储器读写、输入输出等控制信息。

这部分电路的定时由外部振荡器的脉冲所控制，脉冲周期称为时钟周期。每执行某一个指令需要的时间常常是几个甚至几十个时钟周期时间。

第三节　MCS-51 存储器配置

微型计算机必须配置一定数量的存储器，但不同的微型计算机存储器的配置不同。

一种是程序与数据共用一个存储器，如图 4-6(a)所示。一般的通用计算机都采用此种形式。另一种是将程序与数据分别放在两个存储器内，一个称程序存储器，另一个称数据存储器，如图 4-6(b)所示。MCS-51 单片机属于此类。这是由单片机的应用特点所决定的，因为单片机往往是为某个特定对象服务的，这是与通用计算机不同的一个显著特点。它的程序设计调试成功后，一般是固定不变的，因而程序(包括常数表)可以而且也应该一次性地永久放到单片机内。这样不仅省去了每次开机后的程序重新装入步骤，还可以有效地防止因掉电和其它干扰而引起的程序丢失的错误。

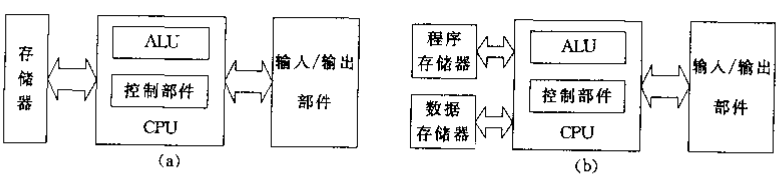


图4-6　微处理器存储器结构

MCS-51 片内集成有一定容量的程序存储器(8031/80c31/8032 除外)和数据存储器，并具有较大的外部存储器扩展能力。

物理上，MCS-51 有 4 个存储器空间：片内程序存储器、片外程序存储器，片内数据存储器、片外数据存储器。

从用户使用角度，即逻辑上，MCS-51 有 3 个存储器地址空间：片内外统一的 64KB 的程序存储器地址空间，256B 内部数据存储器地址空间及 64KB 外部数据存储器地址空间。这 3 个空间的地址可能重叠，因此，在访问这 3 个不同的逻辑空间时应采用不同形式的指令。其配置图如图 4-7(以 8051 为例)所示。

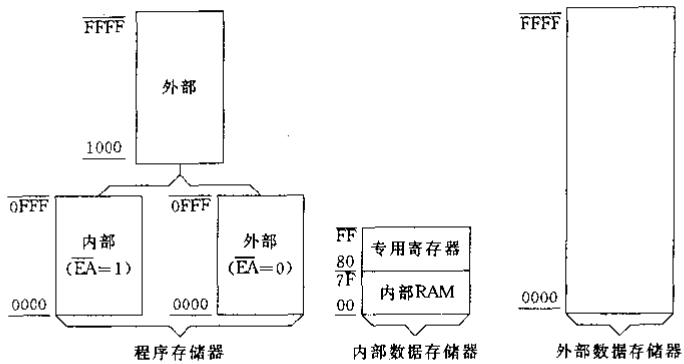


图4-7　8051存储器的配置图

一、程序存储器

8051单片机内部设置有4KB的ROM，8751单片机内部设置有4KB的EPROM作为内部程序存储器，而8031内部没有程序存储器，必须外接程序存储器。由于MCS-51单片机设置有16位的程序计数器，因此可以寻址64KB的程序存储器。程序存储器的作用是用于存放编好的程序和表格常数，程序存储器可通过MOVC指令访问。单片机一般作为专用计算机使用，因此程序存储器通常选用ROM或EPROM来固化用户程序。

二、内部数据存储器

数据存储器在物理上和逻辑上都分为二个地址空间：一个内部和一个外部数据存储器空间。访问内部数据存储器，用MOV指令，访问外部数据存储器用MOVX指令。

8051单片机内部设置有128B的内部数据存储器和128B的特殊功能寄存器寻址空间，在特殊功能寄存器寻址空间离散地分布着19个特殊功能寄存器。

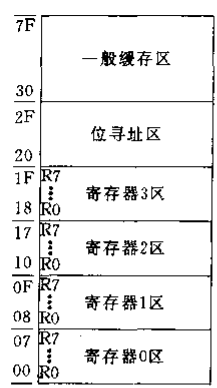


图4-8　内部数据存储器

1. 内部数据存储器

内部数据存储器共有128个字节单元，其分布如图4-8所示。

(1)寄存器区

内部数据存储器的00H～1FH(共32个单元)为4个寄存器工作区，每区8个寄存

器，表示为R0～R7。由于每个寄存器区8个寄存器都记为R0～R7，因此每次只能选择一个寄存器区工作。寄存器工作区的选择是通过状态标志寄存器PSW的第3、4位也即RS1、RS0进行，如表4-1。

表4-1 寄存器工作区选择与地址分配

RS1	RS2	寄存器工作区	R0～R7占用地址
0	0	0区(BANK0)	00H～07H
0	1	1区(BANK1)	08H～0FH
1	0	2区(BANK2)	10H～17H
1	1	3区(BANK3)	18H～1FH

由于MCS-51单片机对寄存器操作最灵活且响应速度最快，一般情况下，总是首先选择应用寄存器，因此有效地设置4个寄存器工作区可以提高现场保护能力和CPU实时响应的速度。

(2)位寻址区

内部数据存储器20H～2FH(16个单元)既可按字节寻址，作为一般工作单元，又可以按位由CPU直接寻址，进行位操作。这16个字节每字节8位，共有16×8=128位，占用地址为00H～7FH(位地址)，如表4-2所示。应用这些位单元，可以方便地进行各种布尔逻辑操作。

表4-2 位地址单元地址分配表

字节地址	D7～D0							
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	549	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

位寻址空间00H～7FH

(3)一般缓冲存储区

内部RAM块中的30H～7FH构成一般缓冲存储区，可用于存放数据，也可作为堆栈存储区域。

2. 专用寄存器

8051 内部有 19 个专用寄存器，其中有 4 个双字节寄存器，PC 寄存器在物理上是独立的，其余 18 个寄存器都属于内部数据存储器的专用寄存器(SFR)块。表 4-3 列出了 18 个专用寄存器的名称、地址分配以及复位后的初始状态。

表 4-3 MCS-51 专用寄存器

特殊功能寄存器	功能名称	地 址	复位后初态
B	通用寄存器	F0H	00H
A	累加器	E0H	00H
PSW	程序状态寄存器	D0H	00H
IP	中断优先级控制寄存器	B8H	×××00000B
P3	P3 口数据寄存器	B0H	FFH
IE	中断允许控制寄存器	A8H	0××00000B
P2	P2 口数据寄存器	A0H	FFH
SBUF	串行口发送/接收缓冲器	99H	不定
SCON	串行口控制寄存器	98H	00H
P1	P1 口数据寄存器	90H	FFH
TH1	T1 计数器高 8 位	8DH	00H
TH0	T0 计数器高 8 位	8CH	00H
TL1	T1 计数器低 8 位	8BH	00H
TL0	T0 计数器低 8 位	8AH	00H
TMOD	定时器/计数器方式控制寄存器	89H	00H
TCON	定时器控制寄存器	88H	00H
PCON	电源控制寄存器	87H	00H
DPH	地址寄存器高 8 位	83H	00H
DPL	地址寄存器低 8 位	82H	00H
SP	堆栈指针寄存器	81H	07H
P0	P0 口数据寄存器	80H	FFH

在专用寄存器中有 11 个寄存器(单元)有专门的位地址，可以按位寻址，其位地址安排如图 4-9 所示。

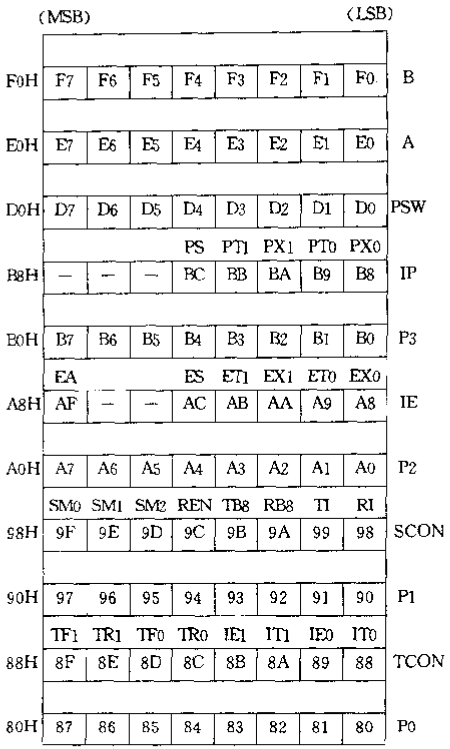


图4-9 SFR块中专用位的地址

三、外部数据存储器

MCS-51 外部数据存储器寻址空间为 64KB，这对多数应用领域已足够使用。对外部数据存储器可用 R0、R1 及 DPTR 间接寻址寄存器。R0、R1 为 8 位寄存器，寻址范围为 256B，DPTR 为 16 位的数据指针寄存器，寻址范围为 64KB。

第四节 时钟电路及时序

一、时钟电路

MCS-51 内部有一个用于构成振荡器的高增益反相放大器，引脚 XTAL1 和 XTAL2 分别是此放大器的输入端和输出端。

MCS-51 的时钟可由内部方式或外部方式产生。

内部方式时钟电路如图 4-10 所示。外接晶体以及电容 C_1、C_2 构成并联谐振电路，接在放大器的反馈回路中，内部振荡器产生自激振荡，一般晶振可在 2～12MHz 之间任选。对外接电容值虽然没有严格的要求，但电容的大小多少会影响振荡频率的高低、振荡器的稳定性、起振的快速性和温度的稳定性。外接晶体时，C_1 和 C_2 通常选 30pF 左右；外接陶瓷谐振器时，C_1 和 C_2 的典型值为 47pF。在设计印刷线路板时，晶体和电容应尽可能安装得与单片机芯片靠近，以保证稳定可靠。

当采用外部方式时钟电路时，外部信号接至 XTAL2（内部时钟电路输入端），而 XTAL1 接地。由于 XTAL2 端的逻辑电平不是 TTL 的，故建议外接一个上拉电阻。通常对外部振荡信号无特殊要求，但需要保证最小高电平及低电平脉宽，一般为频率低于 12MHz 的方波。如图 4-11 所示。

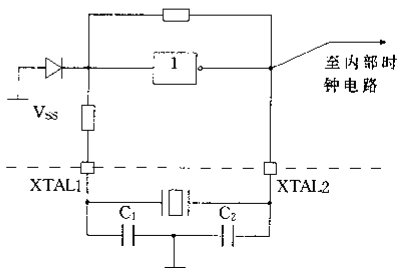


图 4-10　MCS-51 内部方式时钟电路

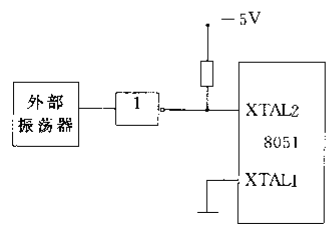


图 4-11　MCS-51 外部方式时钟电路

二、时序

CPU 执行一条指令的时间称为指令周期，它是以机器周期为单位的，MCS-51 典型的指令周期为一个机器周期。MCS-51 的 CPU 取指令和执行指令的时序如图 4-12 所示。

一个机器周期由 6 个状态（12 个振荡脉冲）组成，每一个状态分为两个节拍 P1 和 P2，所以一个机器周期可依次表示为 S1P1、S1P2、…、S6P1、S6P2。一般情况下，算术逻辑操作发生在节拍 P1 时间，而内部寄存器到寄存器的传送发生在节拍 P2 期间。图 4-12 用状态及节拍表明 CPU 指令取出的执行的时序，这些内部信号不能从外部观察到，所以用 XTAL2 振荡信号作参考。

对于单周期指令，在把指令码读入指令寄存器时，从 S1P2 开始执行指令。如果它为双字节指令，则在同一机器周期的 S4 读入第二字节；如果它为单字节指令，则在 S4 仍旧进行读操作，但读入的字节（它应是下一个指令码）被忽略，而且程序计数器不加 1。在任何情况下，在 S6P2 结束指令操作。图 4-12(a)、(b)分别为单字节单周期和双字节单周期指令的时序。

图 4-12(c)所示是单字节双周期指令的时序，在二个机器周期内发生 4 次读操作码的操作，由于是单字节指令，后 3 次读操作都是无效的。图 4-12(d)是访问外部数据存储器

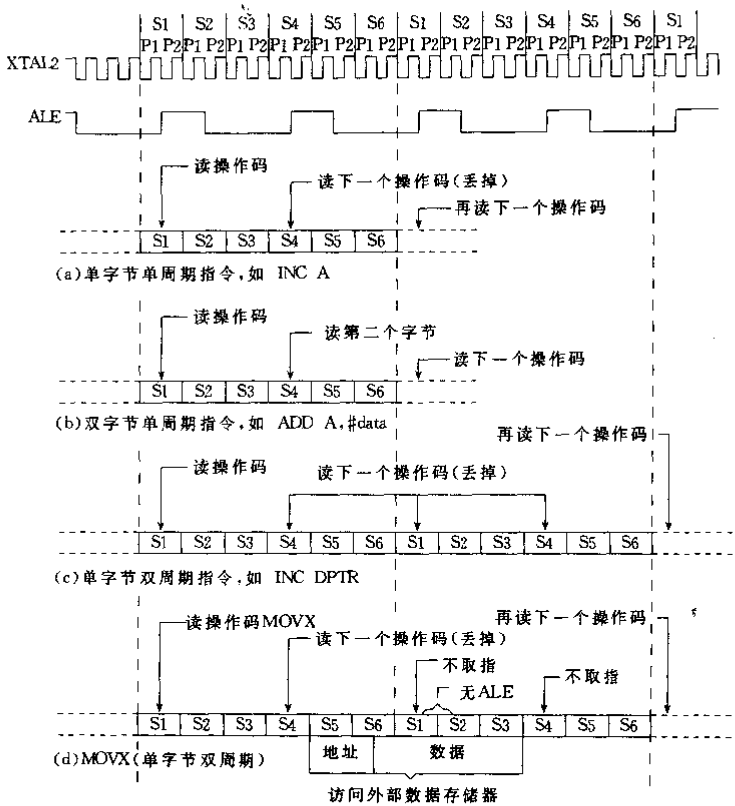


图4-12 8051的取指/执行时序

的指令 MOVX 的时序,它是一条单字节双周期指令。在第一机器周期 S5 开始时,送出外部数据存储器的地址,随后读或写数据,读写期间在 ALE 端不输出有效信号;在第二机器周期,即外部数据存储器已被寻址和选通后,也不产生取指操作。ALE 信号为 MCS-51 扩展系统的外部存储器地址低 8 位的锁存信号,在访问程序存储器的机器周期内,ALE 信号二次有效(S1P2～S2P1 产生正脉冲),因此可以用作为时钟输出信号,但要注意,在执行访问外部数据存储器指令 MOVX 时,要跳过一个 ALE 信号,所以 ALE 的频率可能是不稳定的。

大多数 8051 指令执行时间为一个机器周期,MUL(乘法)和 DIV(除法)是仅有的需要二个以上机器周期的指令,它们需要 4 个机器周期。

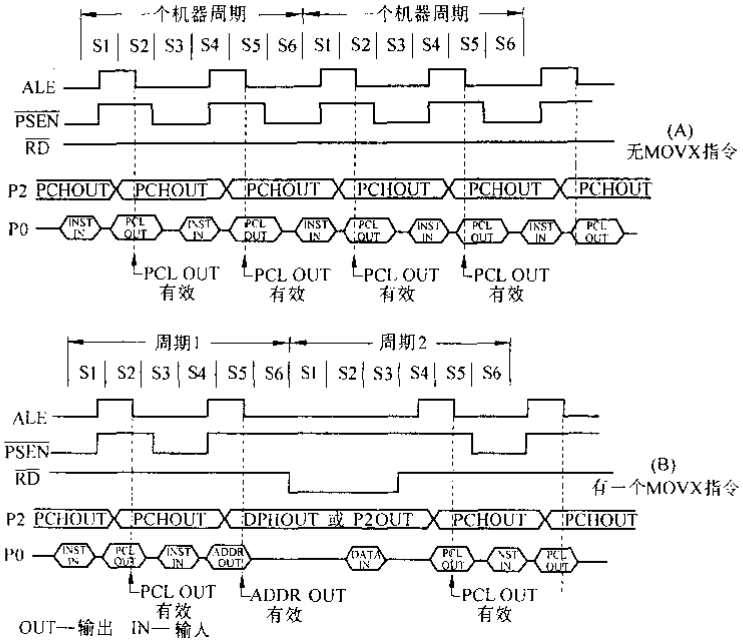


图 4-13 MCS-51 执引外部程序存储器中指令码时的总线周期

第五节 输入输出端口

MCS-51 单片机有 4 个口，共 32 根 I/O 线。所有 4 个端口都是双向口，每口都包含一个锁存器，即专用寄存器 P0～P3，一个输出驱动器和输入缓冲器。为了方便起见，我们把 4 个端口和其中的锁存器(即专用寄存器)都笼统地表示为 P0～P3。

MCS-51 在访问外部存储器时，地址由 P0、P2 口送出，数据则通过 P0 口传送，这时 P0 口是分时多路转换的双向总线。无外部存储器的系统中，所有 4 个端口都可以作为准双向口使用。

图 4-13 给出了访问程序存储器时，程序取指所涉及到的信号和时序。如果程序存储器是外部的，则程序存储器读选通 PSEN 一般是每个机器周期两次有效，如图 4-13(a)所示，如果是访问外部数据存储器，如图 4-13(b)所示，则要跳过两个 PSEN，因为地址和数据总线正在用于访问数据存储器。

应该注意的是，数据存储器总线周期为程序存储器总线周期的 2 倍，图 4-13 给出了端口 0 和端口 2 所发送的地址 ALE 和 PSEN 的相对时序。ALE 用于将 P0 的低位地址字节锁存到地址锁存器中。

1. P0 口

P0 口是 8 位双向三态输入/输出接口，如图 4-14(a)所示。P0 口既可作地址/数据总线使用，又可作通用 I/O 口用。连接外部存储器时，P0 口一方面作为 8 位数据输入/输出口，另一方面用来输出外部存储器的低 8 位地址。作输出口时，输出漏极开路，驱动 NMOS 电路时应外接上拉电阻；作输入口之前，应先向锁存器写 1，使输出的两个场效应管均关断，引脚处于"浮空"状态，这样才能做到高阻输入，以保证输入数据的正确。正是由于该端口用作 I/O 口，输入时应先写 1，故称为准双向口。当 P0 口作地址/数据总线使用时，就不能再把它当通用 I/O 口使用。

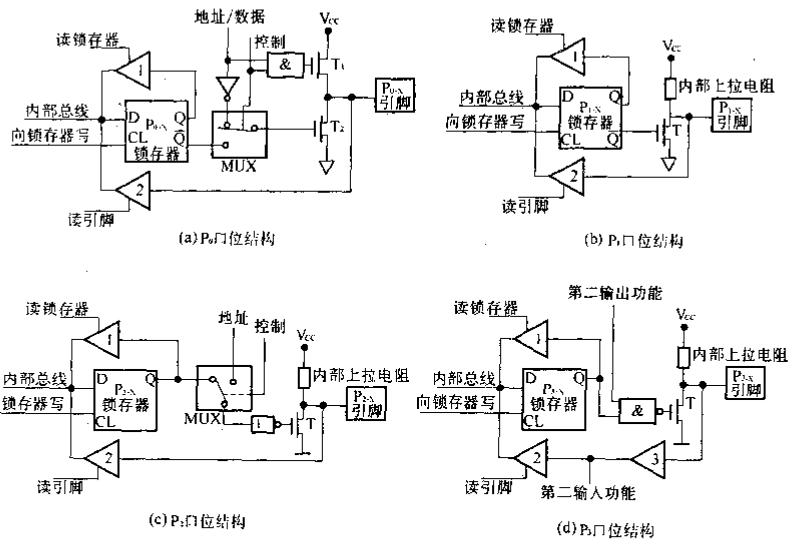


图 4-14 I/O 一位锁存器和缓冲器结构

2. P1 口

P1 口是 8 位准双向口，作通用输入/输出口使用，如图 4-14(b)所示。在输出驱动器部分，P1 口有别于 P0 口，它接有内部上拉电阻。P1 口的每一位可以独立地定义为输入或者输出，因此，P1 口既可以作为 8 位并行输入/输出口，又可作为 8 位输入/输出端。CPU 既可以对 P1 口进行字节操作，又可以进行位操作。当作输入方式时，该位的锁存器必须预写 1。

3. P2 口

P2 口是 8 位准双向输入/输出接口，如图 4-14(c)所示。P2 口可作通用 I/O 口使用，与 P1 口相同。当外接程序存储器时，P2 口给出地址的高 8 位，此时不能用作通用 I/O 口。当外接数据存储器时，若 RAM 小于 256B，用 R0、R1 作间址寄存器，只需 P0 口送出地址低 8 位，P2 口可以用作通用 I/O 口；若 RAM 大于 256B，必须用 16 位寄存器 DPTR 作间址寄存器，则 P2 口只能在一定限度内作一般 I/O 口使用。

4. P3 口

P3 口也是一个 8 位的准双向输入/输出接口，如图 4-14(d)所示。它具有多种功能。一方面与 P1 口一样作为一般准双向输入/输出接口，具有字节操作和位操作二种工作方式；另一方面 8 条输入/输出线可以独立地作为串行输入/输出口和其它控制信号线(详见 MCS-51 引脚功能一节)。

5. P0～P3 端口的负载能力及接口要求

P0 口的输出级与 P1～P3 口的输出级在结构上是不同的，因此它们的负载能力和接口要求也各不相同。

P0 口的每一位输出可驱动 8 个 LS TTL 输入，但把它当通用口使用时，输出级是开漏电路，故用它驱动 NMOS 输入时需外接上拉电阻；把它当地址/数据总线时，则无需接外部上拉电阻。

P1～P3 口的输出级接有内部上拉电阻，它们的每一位输出可驱动 4 个 LS TTL 输入。

CHMOS 端口只能提供几毫安的输出电流，故当作为输出口去驱动一个普通晶体管的基极时，应在端口与晶体管基极间串联一个电阻，以限制高电平输出时的电流。

6. I/O 口的读—修改—写特性

由图 4-14 可见，每个 I/O 端口均有两种读入方法，即读锁存器和读引脚，并有相应的指令，那么如何区分读端口的指令是读锁存器还是读引脚呢？

读锁存器指令是从锁存器中读取数据，进行处理，并把处理以后的数据重新写入锁存器中，这类指令称为“读—修改—写”批令。当目的操作数是一个 I/O 端口或 I/O 端口的某一位时，这些指令是读锁存器而不是读引脚，即为“读—修改—写”指令，下面列出的是一些“读—修改—写”指令。

```
ANL          (逻辑与，例如 ANL  P1,A)
ORL          (逻辑或，例如 ORL  P2,A)
XRL          (逻辑异或，例如 XRL  P3,A)
JBC          (若位=1，则转移并清零，例如 JBC  P1.1, LABEL)
CPL          (取反位，例如 CPL  P3.0)
INC          (递增，例如 INC  P2)
DEC          (递减，例如 DEC  P2)
DJNZ         (递减，若不等于 0 则转移，例如 DJNZ  P3,LABEL)
MOV  P1.7  C (进位位送到端口 P1的位7)
CLR  P1.4    (清零端口 P1的位4)
SETB P1.2    (置位端口 P1的位2)
```

读引脚指令一般都是以 I/O 端口为原操作数的指令，执行读引脚指令时，打开三态门，输入口状态。例如，读 P1口的输入状态时，读引脚指令为：MOV A,P1。

读—修改—写指令指向锁存器而不是引脚，其理由是为了避免可能误解引脚上的电平。例如，端口位可能用于驱动晶体管的基极，在写1至该位时，晶体管导通，若 CPU 随后在引脚处而不是在锁存器处读端口位，则它将读回晶体管的基极电压，将其解释为逻辑0。读该锁存器而不是引脚将返回正确值逻辑1。

第六节　复位电路

复位即回到初始状态，是单片机经常进入的工作状态。在设计单片机应用系统时，必须了解单片机的复位状态。

MCS-51单片机有 HMOS 型(如8051)及 CHMOS 型(如80c51)两种。HMOS 型8051的复位结构如图4-15所示。复位引脚 RST/VPD(它还是掉电方式下内部 RAM 的供电端 VPD)通过一个斯密特触发器与复位电路相连。斯密特触发器用来抑制噪声，它的输出在每个机器周期的 S5P2由复位电路采样一次。

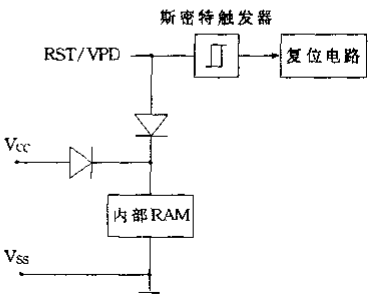


图4-15　HMOS复位结构

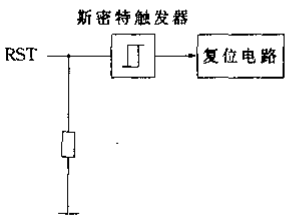


图4-16　CHMOS复位结构

CHMOS 型80C51的复位结构见图4-16，此处复位引脚只是单纯地称为 RST，而不是 RST/VPD，因此 CHMOS 单片机的备用电源也是由 V_{CC} 引脚提供的。

单片机的复位是靠外部电路实现的。无论是 HMOS 还是 CHMOS 型，在振荡器正在运行的情况下，RST 引脚保持二个机器周期以上时间的高电平，系统复位。在 RST 端出现高电平的第二个周期，执行内部复位，以后每个周期重复一次，直至 RST 端变低，复位后，各内部寄存器的状态如表4-3所示。

复位时，ALE 和$\overline{PSEN}$配置为输入状态，即 ALE=1，$\overline{PSEN}$=1。内部 RAM 不受复位的影响。

MCS-51常见的复位电路有以下四种：

1. 上电复位电路

上电复位电路如图4-17所示。上电瞬间，RST 端的电位与 V_{CC} 相同，随着电容的逐步充电，充电电流减小，RST 电位逐渐下降。上电复位所需的最短时间是振荡器建立时间加上二个机器周期，在这段时间内，RST 端口的电平应维持高于斯密特触发器的下阈值。一般 V_{CC} 的上升时间不超过1ms，振荡器建立时间不超过10ms。复位电路的典型值为：C 取10μF，R 取8.2kΩ，放时间常数 $\tau=RC=10\times10^{-6}\times8.2\times10^{3}=82$ms，足以满足要求。

2. 外部复位电路

外部复位电路如图4-18所示，按下按钮时，电源对外接电容充电，使 RST 端为高电

平，复位按钮松开后，电容通过内部下拉电阻放电，逐渐使 RST 端恢复低电平。

3. 上电外部复位电路

典型的复位电路是既具有上电复位又具有外部复位的电路，如图4-19所示。上电瞬间，C 与 R_x 构成充电电路，RST 引脚端出现正脉冲，只要 RST 保持足够的高电平，就能使单片机复位。

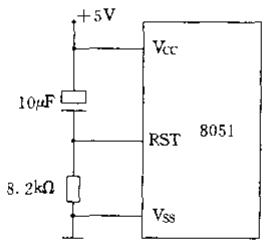


图4-17 上电复位电路

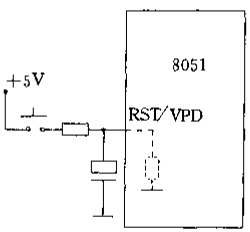


图4-18 外部复位电路

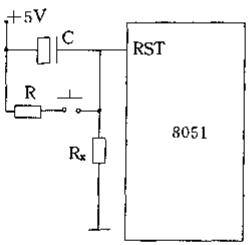
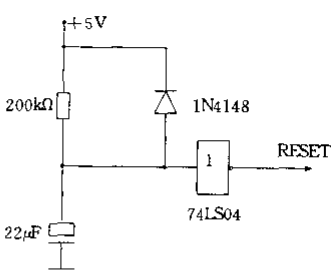


图4-19 上电外部复位电路

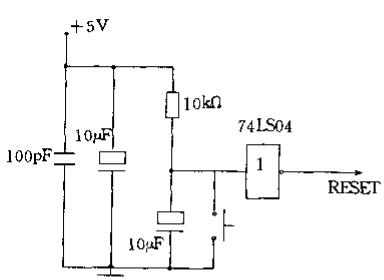
一般取 $C=22\mu F$，$R=200\Omega$，$R_x=1k\Omega$，此时 $\tau=22\times10^{-6}\times1\times10^{3}=22ms$。当按下按钮，RST 出现 $\frac{1000}{1200}\times5=4.2V$，使单片机复位。

4. 抗干扰复位电路

上面介绍的几种复位电路中，干扰易串入复位端，虽然在大多数情况下不会造成单片机的错误复位，但有可能引起内部某些寄存错误复位。在应用系统中，为了保证复位电路可靠地工作，常将 RC 电路在接斯密特电路后再接入单片机复位端及外围电路复位端。图4-20给出了两种实用电路。



(a)实用的上电复位电路



(b)实用的上电及外部复位电路

图4-20 两种实用复位电路

第七节　MCS-51单片机的引脚功能

MCS-51单片机采用40引脚的双列直插封装(DIP)方式。图4-21为其引脚图，逻辑符号如图4-22所示。

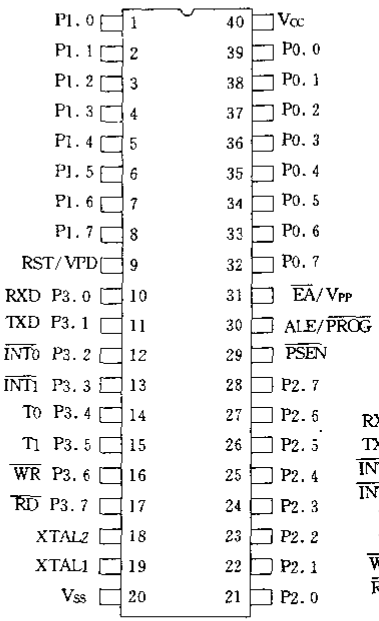


图4-21　MCS-51引脚图

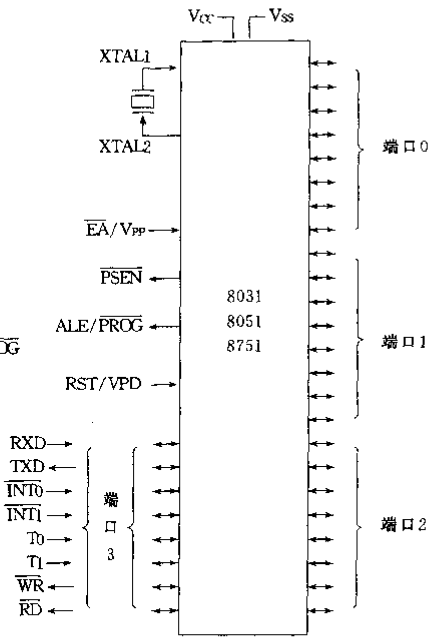


图4-22　MCS-51单片机逻辑符号图

在40条引脚中，有2条专用于主电源的引脚，2条外接晶体的引脚，4条控制引脚，3条I/O引脚。下面分别叙述各引脚的功能。

1. 主电源引脚 V_{SS} 和 V_{CC}

V_{SS}(20)：接地；V_{CC}(40)：正常操作时接+5V电源。

2. 外接晶体引脚 XTAL1和XTAL2

当外接晶体振荡器时，XTAL1和XTAL2分别接在外接晶体两端。当采用外部时钟方式时，XTAL1接地，XTAL2接外来振荡信号。

3. 控制引脚 RST/VPD、ALE/$\overline{PROG}$、$\overline{PSEN}$、$\overline{EA}$/V_{PP}

RST/V_{PP}(9)：当振荡器正常运行时，在此引脚上出现二个机器周期以上的高电平使

单片机复位。

V_{CC}掉电期间，此引脚可接备用电源，以保持内部RAM的数据。当V_{CC}下降掉到低于规定的水平，而VPD在其规定的电压范围内，VPD就向内部RAM提供备用电源。

ALE/$\overline{PROG}$(30)：当访问外部存储器时，由单片机的P2口送出地址的高8位，P0口送出地址的低8位，数据也是通过P0口传送。作为P0口某时送出的信息到底是低8位地址还是传送的数据，需要有一信号同步地进行分别。当ALE信号（允许地址锁存）为高电平（有效），P0口送出低8位地址，通过ALE信号锁存低8位地址。即使不访问外部存储器，ALE端仍以不变的频率周期性地出现正脉冲信号，此频率为振荡器频率的1/6，因此可用作对外输出的时钟。但需注意：当访问外部数据存储器（执行MOVX指令）时，将跳过一个ALE脉冲。ALE端可驱动8个LS TTL输入。

对于8751EPROM型单片机，在EPROM编程期间，此引脚用于输入编程脉冲（$\overline{PROG}$）。

$\overline{PSEN}$(29)：程序存储器读选通信号，低电平有效。

MCS-51单片机可以外接程序存储器及数据存储器，它们的地址可以是重合的。MCS-51单片机是通过相应的控制信号来区别到底P2口和P0口送出的是程序存储器还是数据存储器地址。从外部程序存储器取指令（或常数）期间，每个机器周期两次$\overline{PSEN}$有效，此时地址总线上送出地址为程序存储器地址；如果访问外部数据存储器，这两次有效的$\overline{PSEN}$信号将不出现。外部数据存储器是靠$\overline{RD}$（读）及$\overline{WR}$（写）信号控制的。$\overline{PSEN}$同样可以驱动8个LSTTL输入。

$\overline{EA}$/V_{PP}(31)：当$\overline{EA}$端保持高电平时，访问内部程序存储器（4KB），但当PC（程序计数器）值超过0FFFH时，将自动转向执行外部程序存储器内的程序。当$\overline{EA}$保持低电平时，则只访问外部程序存储器（从0000H地址开始），不管单片机内部是否有程序存储器。

对于EPROM型单片机，在EPROM编程期间，此引脚用于施加21V的编程电源（V_{PP}）。

4. 输入输出引脚

P0.0～P0.7(39～32)：P0口是一个漏极开路型准双向I/O口。在访问外部存储器时，它是分时多路转换的地址（低8位）和数据总线，在访问期间激活了内部的上拉电阻。在EPROM编程时，它接收指令字节，而在验证程序时，则输出指令字节。验证时，要求外接上拉电阻。

P1.0～P1.7(1～8)：P1口是带内部上拉电阻的8位双向I/O口。在EPROM编程和程序验证时，它接收低8位地址。

P2.0～P2.7(21～28)：P2口是一个带内部上拉电阻的8位双向I/O口。在访问外部存储器时，它送出高8位地址。在对EPROM编程和程序验证期间，它接收高8位地址。

P3.0～P3.7(10～17)：P3口是一个带内部上拉电阻的8位双向I/O口。在MCS-51中，这8个引脚还兼有专用功能，这些功能见表4-4。

表4-4 P3各口线与专用功能

口线	替代的专用功能
P3.0	RXD(串行输入口)
P3.1	TXD(串行输出口)
P3.2	$\overline{\text{INT0}}$(外部中断0)
P3.3	$\overline{\text{INT1}}$(外部中断1)
P3.4	T0(定时器0的外部输入)
P3.5	T1(定时器1的外部输入)
P3.6	$\overline{\text{WR}}$(外部数据存储器写选通)
P3.7	$\overline{\text{RD}}$(外部数据存储器读选通)

这些专用功能的口线,在与外部设备接口、外接数据存储器等方面具有非常重要的作用。

思考题与习题

1. MCS-51系列单片机内部有哪些主要的逻辑部件?

2. MCS-51的程序存储器和数据存储器各有什么用处?

3. MCS-51内部RAM区功能结构如何分配?4组工作寄存器使用时如何选用?位寻址区域的字节地址范围是多少?

4. 简述程序状态字PSW中各位的含义?

5. MCS-51设有4个8位并行端口,实际应用中8位数据信息由哪个端口传送?16位地址如何形成?

6. 试分析MCS-51端口的两种读操作(读引脚和读锁存器),读一修改一写操作是哪一种操作?结构上这种安排有何功用?

第五章　指令系统

计算机可以笼统地分为硬件和软件两大部分。上一章我们介绍了 MCS-51 系列单片机的各个组成部分，它们构成了单片机的硬件。但计算机要发挥它应有的效能，软件的支持是必不可少的。软件是各种程序的总称，而程序又是由一条条指令组成的。从本章开始，我们介绍 MCS-51 系列单片机的指令系统及汇编语言程序设计。

第一节　指令系统概述

计算机的所有指令的集合称为该计算机的指令系统。不同型号的计算机其指令系统是不同的。指令系统也是衡量计算机性能的一个重要方面。

一、MCS-51 系列单片机指令的分类

MCS-51 系列单片机共有 111 种指令。这些指令可分为几类。

1. 按指令所占的字节数分类

(1)单字节指令(49 条)；

(2)双字节指令(46 条)；

(3)三字节指令(16 条)。

2. 按指令执行时间的长短分类

(1)单周期指令(65 条)；

(2)双周期指令(44 条)；

(3)四周期指令(2 条)。

3. 按指令的功能分类

(1)数据传送指令(29 条)；

(2)算术运算指令(24 条)；

(3)逻辑运算指令(24 条)；

(4)控制转移指令(17 条)；

(5)位操作指令(17 条)。

MCS-51 系列单片机的指令系统设有位操作指令集，与位处理器和可位寻址位构成了一个完整的位处理机，对于设计那些处理位变量的程序十分有效，特别适用于逻辑控制应用。

二、寻址方式

寻址方式就是计算机确定操作数或下一条要执行指令的地址的方法。由于寻址方式基本反映了计算机的操作过程，因此对于我们理解指令的功能是非常重要的。

1. 立即寻址

立即寻址是指指令中直接给出操作数的寻址方式。指令中的操作数也称为立即数，其标志为前面加“#”。例如指令：MOV A，#30H 中，30H 为立即数，该指令的功能为将立即数 30H 送入累加器 A 中，其执行过程如图 5-1 所示。

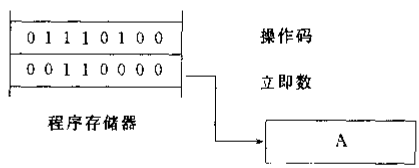


图5-1　立即寻址示意图

在 MCS-51 系列单片机指令系统中，立即数多为 8 位，只有一条指令的立即数为 16 位，如 MOV DPTR，#2000H。

2. 直接寻址

直接寻址是指令中给出操作数地址的寻址方式。这种寻址方式可以访问的地址空间有：内部数据存储器、特殊功能寄存器和可位寻址位。需要指出的是，直接寻址是访问特殊功能寄存器的唯一寻址方式。例如指令 MOV A，30H，其功能为将内部数据存储器的 30H 单元中的内容传送到累加器 A 中。指令执行的示意图如图 5-2 所示。

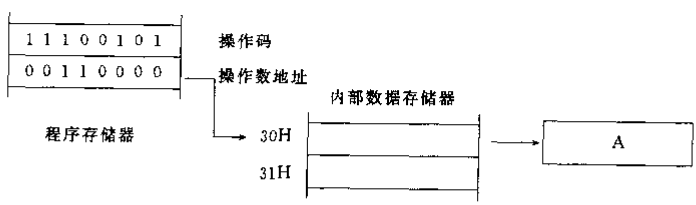


图5-2　直接寻址示意图

3. 寄存器寻址

寄存器寻址是对选定的工作寄存器 R0～R7、累加器 A、通用寄存器 B、地址寄存器 DPTR 和位累加器 C 中的数进行操作的寻址方式。这种寻址方式的特点是工作寄存器 R0～R7 与指令码构成一个字节，A、B、DPTR 和 C 隐含在指令码中。如指令 MOV A，R0，其功能为将工作寄存器 R0 中的内容传送到 A 中。指令执行的示意图如图 5-3 所示。

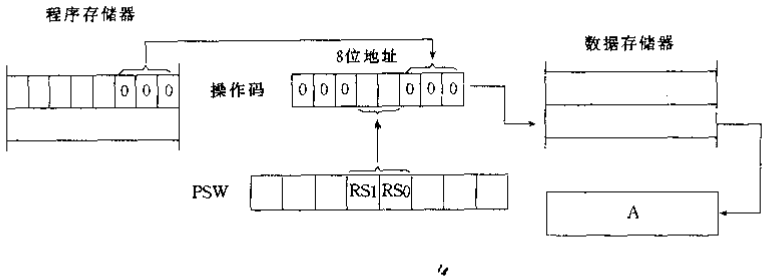


图5-3 寄存器寻址示意图

4. 寄存器间接寻址

寄存器间接寻址是将指令指定的寄存器内容为地址，对该地址单元中的内容进行操作的寻址方式。如指令 MOV A,@R0，其功能为将 R0 中的内容为地址的单元中的内容传送到 A 中。若 R0 中的内容为 30H，则上面指令就是将 30H 中的内容传送到 A 中。指令的执行过程如图 5-4 所示。

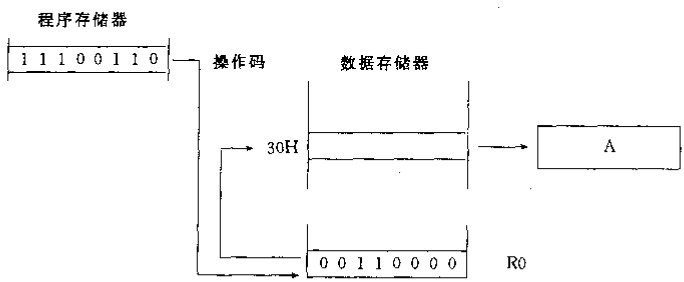


图5-4 寄存器间接寻址示意图

在 MCS-51 单片机指令系统中规定，R0、R1 和 DPTR 作为间接寻址寄存器。这种寻址方法可以访问内部数据存储器的 128 个字节和外部数据存储器的 64K 空间，但不能访问特殊功能寄存器。

5. 相对寻址

相对寻址是以当前的 PC 值加上指令中给出的相对偏移量形成程序转移的目的地址的寻址方式。相对偏移量是有符号的 8 位二进制数，用补码表示。例如指令 JC rel 所在地址为 2100H，当 rel＝75H，Cy＝1 时，执行该指令，由于当前的 PC 值为指令所在地址 2100H＋2 即 2102H，程序将转移到 2177H 单元去执行。其执行的示意图如图 5-5 所示。

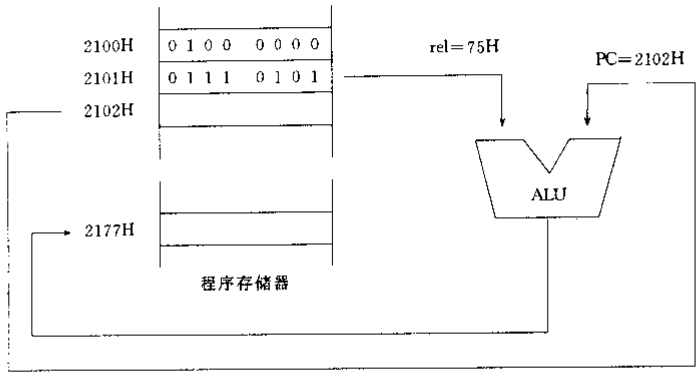


图5-5 相对寻址示意图

6. 变址寻址

变址寻址是指令中指定的变址寄存器和基址寄存器的内容相加形成操作数地址的寻址方式。在这种寻址方式中，累加器 A 作为变址寄存器，程序计数器 PC 或地址寄存器 DPTR 作为基址寄存器。变址寻址在所谓的查表指令中采用，用于访问程序存储器。例如指令 MOVC A,@A+DPTR，设累加器中的内容为 30H，DPTR 中的内容为 2100H，执行这条指令，就是把程序存储器 2100H+30H=2130H 单元中的内容传送到累加器中。执行过程如图 5-6 所示。

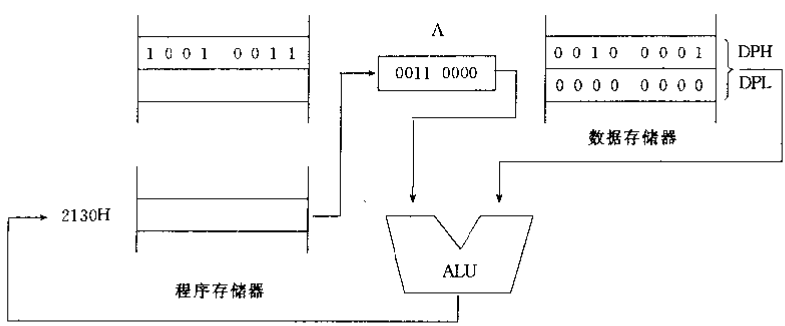


图5-6 变址寻址示意图

7. 位寻址

位寻址是对内部 RAM 和特殊功能寄存器的可位寻址位的内容进行操作的寻址方式。这种寻址方式与直接寻址方式的形式和执行过程基本相同，但参与操作的数据是 1 位

而不是 8 位，这里不再举例。

三、指令中的常用符号

在介绍指令之前，先说明指令中常用的符号：

Rn——当前寄存器区的 8 个工作寄存器 R0～R7(n=0～7)；

Ri——当前寄存器区可作地址寄存器的 2 个工作寄存器 R0 和 R1(i=0,1)；

direct——8 位内部数据存储器单元的地址及特殊功能寄存器的地址；

#data——表示 8 位常数；

#data16——表示 16 位常数；

addr16——表示 16 位地址；

addr11——表示 11 位地址；

rel——8 位带符号的地址偏移量，取值范围为－128～＋127；

bit——内部 RAM 和特殊功能寄存器中的可直接寻址位；

@——间接寻址寄存器或基址寄存器的前缀；

( )——表示括号中单元的内容；

(())——表示间接寻址的内容；

←——表示数据的传送方向；

⇌——表示数据交换。

第二节　MCS-51 单片机指令系统

指令分类方法很多，为方便起见，这里按功能分类介绍 MCS-51 单片机的指令系统。

一、数据传送指令

数据传送指令是应用最频繁的指令。MCS-51 单片机提供了丰富的数据传送指令，是数量最多的一类指令。

数据传送指令的助记符为 MOV，其汇编语言指令格式为

MOV　[目的字节],[源字节]

指令功能是将源字节的内容传送到目的字节，源字节的内容不变。这类指令不影响标志位。

1. 内部 8 位数据传送指令

内部 8 位数据传送指令共有 15 条，用于单片机内部的数据存储器和寄存器之间的数据传送。所采用的寻址方式有立即寻址、直接寻址、寄存器寻址和寄存器间接寻址。

汇编语言指令			机器语言指令		功　能
MOV	A,Rn	;	1110 1rrr	,	A←(Rn)
MOV	A,direct	;	1110 0101 direct	,	A←(direct)
MOV	A,@Ri	;	1110 011i	,	A←((Ri))

MOV A,#data ; 0111 0100 , A←data
data

这 4 条指令的目的字节单元都是累加器 A，源字节单元分别采用四种寻址方式。

指令中 Rn 表示工作寄存器 R0～R7，机器码指令中的低 3 位是工作寄存器的地址：rrr=000～111，对应工作寄存器 R0～R7。Ri 表示间接寻址寄存器，i=0，1，即 Ri 为 R0 和 R1。

MOV Rn,A ; 1111 1rrr , Rn←(A)

MOV Rn,direct ; 1010 1rrr , Rn←(direct)
direct

MOV Rn,#data ; 0111 1rrr , Rn←data
data

这 3 条指令都是以工作寄存器为目的字节单元，源字节单元的寻址方式有寄存器寻址、直接寻址和立即寻址。

MOV direct,A ; 1111 0101 , direct←(A)
direct

MOV direct,Rn ; 1000 1rrr , direct←(Rn)
direct

MOV direct1,direct2 ; 1000 0101 , direc1←(direct2)
direct2
direct1

MOV direct,@Ri ; 1000 011i , direct←((Ri))
direct

MOV direct,#data ; 0111 0101 , direct←data
direct
data

这组指令的目的字节都是直接寻址单元，源字节单元采用的寻址方式有寄存器寻址、直接寻址、寄存器间接寻址和立即寻址。

MOV @Ri,A ; 1111 011i , (Ri)←(A)

MOV @Ri,direct ; 1010 011i , (Ri)←direct
direct

MOV @Ri,#data ; 0111 011i , (Ri)←data
data

上述 3 条指令的目的字节是寄存器间接寻址单元，源字节单元的寻址方式为寄存器寻址、直接寻址和立即寻址。

例 5-1 分析程序的执行结果。

解 设内部 RAM 中 30H 单元的内容为 50H，试分析执行下面程序后各有关单元的内容。

```
MOV    60H,#30H      ;立即数 30H 送 60H 单元，即 60H←30H
```

```
MOV     R0,#60H         ;立即数 60H 送入 R0
MOV     A,@R0           ;间接寻址,即 60H 单元内容送入 A,A←30H
MOV     R1,A            ;将 A 中的内容送入 R1,即 R1←30H
MOV     40H,@R1         ;间接寻址,将 30H 中的内容送 40H 单元,即 40H←50H
MOV     60H,30H         ;30H 单元的内容送入 60H,即 60H←50H
```

程序执行结果是(A)=30H,(R0)=60H,(R1)=30H,(60H)=50H,(40H)=50H,(30H)=50H 内容未变。

2. 16 位数据传送指令

MOV DPTR,#data16 ; 1001 0000 / data $_{15\sim8}$ / data $_{7\sim0}$, DPTR←data16

MCS-51 单片机指令系统中仅此一条 16 位数据传送指令,功能是将 16 位数据送入地址寄存器 DPTR 中,其中数据的高 8 位送入 DPH 中,低 8 位送入 DPL 中。

3. 外部数据传送指令

外部数据传送指令用于 CPU 与外部数据存储器之间的数据传送。对外部数据存储器的访问均采用间接寻址方式。间接寻址寄存器有两类,即 8 位间址寄存器 R0、R1,寻址范围为 256 个字节;16 位间址寄存器 DPTR,寻址范围为 64K 地址空间。

```
MOVX    A,@Ri           ;  1110 001i  ,  A←((Ri))
MOVX    A,@DPTR         ;  1110 0000  ,  A←((DPTR))
MOVX    @Ri,A           ;  1111 001i  ,  (Ri)←(A)
MOVX    @DPTR,A         ;  1111 0000  ,  (DPTR)←(A)
```

前 2 条指令为外部数据存储器读指令,另外 2 条指令则为外部数据存储器写指令。它们的特点是数据都通过累加器 A 进行传送。

例 5-2 将累加器 A 中的内容送入外部数据存储器的 60H 单元。

解 根据题意编程如下:

```
MOV     R0,#60H         ;地址送间址寄存器
MOVX    @R0,A           ;(R0)←(A),A 中内容送外部数据存储器的 60H 单元
```

例 5-3 将外部数据存储器的 1F00H 单元的内容传送到内部数据存储器的 60H 单元。

解 程序如下:

```
MOV     DPTR,#1F00H     ;外部数据存储器的地址送地址寄存器
MOV     R0,#60H         ;内部数据存储器地址送间址寄存器
MOVX    A,@DPTR         ;读外部数据存储器
MOV     @R0,A           ;(R0)←A
```

例 5-4 将外部数据存储器的 2000H 单元的内容送入 2100H 单元。

解 程序如下:

```
MOV    DPTR,#2000H;DPTR←2000H,准备外部数据存储器单元的地址
MOVX   A,@DPTR     ;A←((DPTR))
MOVX   DPTR,#2100H;DPTR←2100H
MOVX   @DPTR,A     ;(DPTR)←(A)
```

4. 查表指令

MCS-51 单片机的程序存储器除了存放程序外，还可以存放一些常数，称为表格。MCS-51 单片机指令系统提供了 2 条访问程序存储器的指令，我们称为查表指令。

```
MOVC   A,@A+DPTR   ;  1001 0011  ,  A←((A)+(DPTR))
MOVC   A,@A+PC     ;  1000 0011  ,  PC←(PC)+1
                                    A←((A)+(PC))
```

查表指令的源字节单元都采用变址寻址方式，第一条指令的基址寄存器为 DPTR，因此其寻址范围为整个程序存储器的 64K 空间，表格可以放在程序存储器的任何位置。第二条指令的基址寄存器为 PC，该指令中访问程序存储器的地址为(A)+(PC)，其中(PC)为程序计数器的当前内容，即查表指令的地址加 1。因此，当基址寄存器为 PC 时，查表范围实际为查表指令后 256 个字节的地址空间。例如：

```
1232H:MOV A,#30H
1234H:MOVC A,@A+PC
1235H:MOV 60H,A
      ⋮
1265H:3FH
      ⋮
```

当执行查表指令时，PC 的当前值为 1235H，所以，查表指令访问的程序存储器单元的地址为(A)+(PC)=30H+1235H=1265H。

5. 交换指令

累加器低 4 位与高 4 位互换指令：

```
SWAP   A           ;  1100 0100  ,  (A)3~0⇌(A)7~4
```

例如：设(A)=ABH，执行指令 SWAP A 后，(A)=BAH。

内部 RAM 单元低 4 位内容与累加器低 4 位内容交换指令：

```
XCHD   A,@Ri       ;  1101 011i  ,  (A)3~0⇌((Ri))3~0
```

这条指令也属于半字节交换指令，例如：设(A)=ABH，(R0)=30H，(30H)=12H，执行指令 XCHD A,@R0 后，(A)=A2H，(30H)=1BH。

字节交换指令：

```
XCH    A,Rn        ;  1100 1rrr  ,  (A)⇌(Rn)
XCH    A,direct    ;  1100 0101  ,  (A)⇌(direct)
                      direct
XCH    A,@Ri       ;  1100 011i  ,  (A)⇌((Ri))
```

这类指令是将源字节单元和目的字节单元的内容互换。例如：设(A)=ABH，(R1)=12H，执行指令 XCH A,R1，则结果为(A)=12H，(R1)=ABH。

例 5-5 设内部数据存储器的60H、61H单元中连续存放着4位BCD码。试编写一段程序将这4位BCD码倒序排列。

解 根据题意设4位BCD码为$a_3a_2a_1a_0$，在程序执行前后，它们在数据存储器中的排列如下图：

程序如下：

```
MOV   R0,#60H      ;R0←60H
MOV   R1,#61H      ;R1←61H
MOV   A,@R0        ;A←((R0))
SWAP  A            ;A 中内容的高、低 4 位互换
XCH   A,@R1        ;(A)⇌((R1))
SWAP  A
MOV   @R0,A        ; (R1)←(A)
```

6. 堆栈操作指令

入栈指令：

PUSH direct ; 1100 0000 / direct , SP←(SP)+1 / (SP)←(direct)

入栈指令是将其指定的直接寻址单元的内容压入堆栈。具体操作是：先将堆栈指针寄存器的内容加1，指向堆栈顶的一个空单元，然后将指令指定的直接寻址单元内容传送到这个空单元中。

出栈指令：

POP direct ; 1101 0000 / direct , (direct)←((SP)) / SP←(SP)−1

出栈指令是将当前堆栈指针寄存器SP所指示的单元的内容传送到该指令指定的直接寻址单元中去，然后SP中的内容减1。

由入栈和出栈指令的操作过程可以看出，堆栈中的数据的压入和弹出应遵循"先进后出"的原则。

表 5-1 数据传送类指令一览表

助记符	功能说明	字节数	振荡器周期
MOV A,Rn	寄存器内容传送到累加器	1	12
MOV A,direct	直接寻址字节传送到累加器	2	12
MOV A,@Ri	间接 RAM 传送到累加器	1	12
MOV A,#data	立即数传送到累加器	2	12

续表 5-1

助记符	功能说明	字节数	振荡器周期
MOV Rr,A	累加器内容传送到工作寄存器	1	12
MOV Rn,direct	直接寻址字节传送到工作寄存器	2	24
MOV Rn,#data	立即数传送到工作寄存器	2	12
MOV direct,A	累加器内容传送到直接寻址字节	2	12
MOV direct,Rn	工作寄存器内容传送到直接寻址字节	2	24
MOV direct,direct	直接寻址字节传送到直接寻址字节	3	24
MOV direct, @Ri	间接 RAM 传送到直接寻址字节	2	24
MOV direct, #data	立即数传送到直接寻址字节	3	24
MOV @Ri,A	累加器传送到间接寻址 RAM	1	12
MOV @Ri,direct	直接寻址字节传送到间接寻址 RAM	2	24
MOV @Ri,#data	立即数传送到间接寻址 RAM	2	12
MOV DPTR,#data16	16 位常数传送到地址寄存器	3	24
MOVX A,@Ri	外部 RAM(8 位地址)传送到累加器	1	24
MOVX A, @DPTR	外部 RAM(16 位地址)传送到累加器	1	24
MOVX @Ri,A	累加器传送到外部 RAM(8 位地址)	1	24
MOVX @DPTR,A	累加器传送到外部 RAM(16 位地址)	1	24
MOVC A, @A+DPTR	程序存储器字节传送到累加器	1	24
MOVC A, @A+PC	程序存储器字节传送到累加器	1	24
SWAP A	累加器内半字节交换	1	12
XCHD A ,@Ri	间接寻址 RAM 和累加器低半字节交换	1	12
XCH A,Rn	寄存器和累加器交换	1	12
XCH A,direct	直接寻址字节和累加器交换	2	12
XCH A, @Ri	间接寻址 RAM 和累加器交换	1	12
PUSH direct	直接寻址字节压入栈顶	2	24
POP direct	栈顶弹到直接寻址字节	2	24

二、算术运算指令

MCS-51 单片机的算术运算指令包括加、减、乘、除、加 1、减 1 等指令。这类指令大都影响标志位。加法和减法的执行结果将影响程序状态标志寄存器 PSW 的进位位 C、溢出位 OV、辅助进位位 AC 和奇偶校验位 P；乘、除指令的执行结果将影响 PSW 的溢出位 OV、进位位 C 和奇偶校验位 P；加 1、减 1 指令的执行结果只影响 PSW 的奇偶校验位 P。

1. 加法指令

ADD　A,Rn　　；　0010 1rrr　，　A←(A)+(Rn)

ADD	A,direct	;	0010 0101 direct	, A←(A)+(direct)
ADD	A,@Ri	;	0010 011i	, A←(A)+((Ri))
ADD	A,#data	;	0010 0100 data	, A←(A)+data

加法指令是将源字节单元的内容与累加器 A 中的内容相加，结果存入累加器 A 中。

2. 带进位的加法指令

ADDC	A,Rn	;	0011 1rrr	, A←(A)+(Rn)+(C)
ADDC	A,direct	;	0011 0101 direct	, A←(A)+(direct)+(C)
ADDC	A,@Ri	;	0011 011i	, A←(A)+((Ri))+(C)
ADDC	A,#data	;	0011 0100 data	, A←(A)+data+(C)

这 4 条指令与上述加法指令一一对应。这些指令是将源字节单元的内容与累加器 A 中的内容相加，再加上进位位 C 的内容，结果放入累加器 A 中。带进位的加法指令主要用于多字节数的加法运算。

3. 带借位的减法指令

SUBB	A,Rn	;	1001 1rrr	, A←(A)−(Rn)−(C)
SUBB	A,direct	;	1001 0101 direct	, A←(A)−(direct)−(C)
SUBB	A,@Ri	;	1001 011i	, A←(A)−((Ri))−(C)
SUBB	A,#data	;	1001 0100 data	, A←(A)−data−(C)

带借位的减法指令的功能是将累加器 A 中的内容减去源字节单元的内容，再减去进位位 C 的内容，结果放入累加器 A 中。这组指令主要用于多字节数的减法，这时 C 中保存着低位字节向高位字节的借位。如果要进行单字节或多字节数的最低 8 位数的减法运算，应先清除进位位 C。

加、减法运算对 PSW 中的状态标志位的影响在前面已经介绍，这里再总结如下：当加法运算结果的最高位有进位，或减法运算的最高位有借位时，进位位 C 置位，否则 C 清零；当加法运算时低 4 位向高 4 位有进位，或减法运算时低 4 位向高 4 位有借位时，辅助进位位 AC 置位，否则 AC 清零；在加、减运算过程中，位 6 和位 7 不同时产生进位或借位时，辅助溢出标志位 OV 置位，否则清零；当累加器 A 中的 8 位数据的各位有奇数个 1 时，奇偶校验位 P 置位，否则清零。

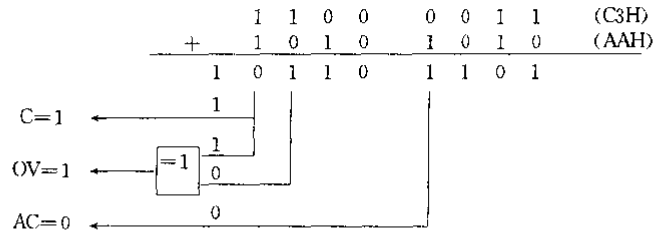
例 5-6 设(A)=0C3H,(R0)=0AAH,执行指令：

```
ADD  A,R0
```

结果为 (A)=6DH

标志位为 C=1 OV=1 AC=0

运算过程为



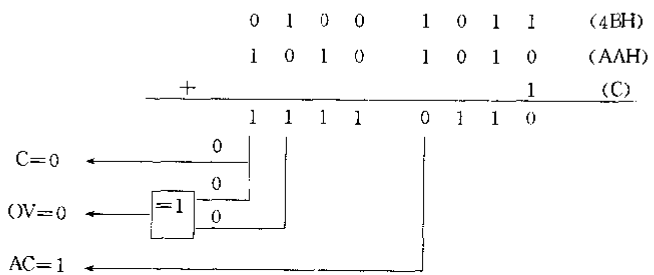
例 5-7 设(A)=4BH,(C)=1,执行指令:

ADDC A,#0AAH

结果为 (A)=F6H

标志位为 C=0 OV=0 AC=1

运算过程为



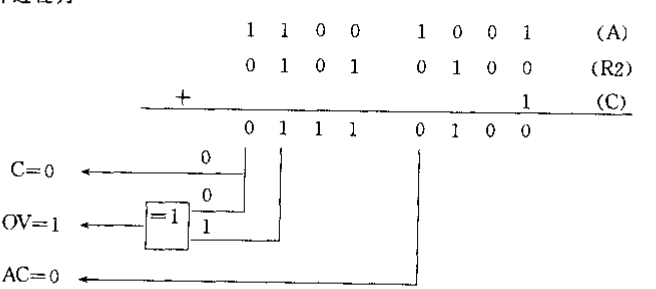
例 5-8 设(A)=0C9H,(R2)=54H,(C)=1,执行指令:

SUBB A,R2

结果为 (A)=74H

标志位为 C=0 AC=0 OV=1

运算过程为



值得一提的是,状态标志位也应看做是运算结果的一部分,在不同情况下其意义也不同。例如:对于无符号二进制加法,进位位 Cy 置 1,表示运算结果大于 255,产生了溢出;对

于有符号二进制加法，溢出标志位 OV 置 1，表示运算结果小于－128 或大于 127，也产生了溢出。在例 5-6 中，如果将两个操作数都看成有符号数，两个负数相加结果为正，显然是错误的，溢出标志位置 1。

例 5-9 试编写计算 6655H＋11FFH 的程序。

解 加数和被加数是 16 位数，需二步完成运算：低 8 位数相加，若有进位保存在 C 中；高 8 位采用带进位的加法，结果放入 50H、51H 中。

```
MOV    A,#55H        ;A←55H
ADD    A,#0FFH       ;A←(A)+FFH
MOV    50H,A         ;50H←(A)
MOV    A,#66H        ;A←66H
ADDC   A,#11H        ;A←(A)+11H+(C)
MOV    51H,A         ;51H←(A)
```

例 5-10 试编写计算 EE33H－A0E0H 的程序。

解 16 位减法运算也需分二步完成：在进行低 8 位运算前应将进位位清零，低 8 位运算后由 C 保存借位；在进行高 8 位运算时，借位位也一起参与运算。程序如下：

```
CLR    C             ;C←0
MOV    A,#33H        ;A←33H
SUBB   A,#0E0H       ;A←(A)－E0
MOV    50H,A         ;50H←(A)
MOV    A,#0EEH
SUBB   A,#0A0H
MOV    51H,A
```

4. BCD 码修正指令

DA　A　；[1101 0100]，若〔$(A)_{3\sim0}>9$〕或〔(AC)＝1〕，则 $A_{3\sim0}\leftarrow(A)_{3\sim0}+6$；同时，若〔$(A)_{7\sim4}>9$〕或〔(C)＝1〕，则 $A_{7\sim4}\leftarrow(A)_{7\sim4}+6$

BCD 码调整指令用来对 BCD 码的加法运算结果进行修正。

在计算机中，表示 0～9 之间的十进制数也是用二进制数表示的，即 BCD 码。在运算过程中，计算机按二进制规则进行，即每位相加大于 16 时进位，十进制数在大于 10 时进位，因此 BCD 码运算时，结果大于 9 时得到的结果是不正确的，必须进行修正。

例如：设 A＝34，B＝53，求 A＋B＝？

34 的 BCD 码为 00110100，53 的 BCD 码为 01010011。运算过程为

```
    0 0 1 1   0 1 0 0
+   0 1 0 1   0 0 1 1
-------------------------
    1 0 0 0   0 1 1 1
```

结果为 87，显然运算结果仍然为 BCD 码，并且运算过程没有产生进位，这一结果是正确的，可以不必修正。

设 A＝37，B＝46，求 A＋B＝？

这两个 BCD 码的运算过程为：

$$\begin{array}{rcccccccc} & 0 & 0 & 1 & 1 & & 0 & 1 & 1 & 1 \\ + & 0 & 1 & 0 & 0 & & 0 & 1 & 1 & 0 \\ \hline & 0 & 1 & 1 & 1 & & 1 & 1 & 0 & 1 \end{array}$$

结果为 7DH。运算结果的低位大于 9，已不是 BCD 码。这种情况下就需要进行修正，以得正确的 BCD 码。

设 A=39，B=48，求 A+B=？

这两个 BCD 码的运算过程为：

$$\begin{array}{rcccccccc} & 0 & 0 & 1 & 1 & & 1 & 0 & 0 & 1 \\ + & 0 & 1 & 0 & 0 & & 1 & 0 & 0 & 0 \\ \hline & 1 & 0 & 0 & 0 & & 0 & 0 & 0 & 1 \end{array}$$

运算结果为 81。显然这一结果是错误的，在运算过程中低位产生了进位，必须进行修正。

在进行减法运算时，低位可能向高位产生借位。按照十进制运算规则，借 1 当 10。在计算机中是按照二进制规则进行运算，即借 1 当 16，这样在进行 BCD 码的减法运算时，如果产生借位也必须进行修正。

在计算机中，BCD 码加法运算时，中间结果的修正是由 ALU 硬件中的十进制修正电路自动进行的。使用时只要在上述加法运算指令的后面紧跟一条 BCD 码修正指令即可。

必须注意的是：在 MCS-51 单片机中，DA　A 指令不能直接用在减法指令后面，即不能直接用来对 BCD 码的减法运算进行修正。

例 5-11　设有两个 4 位 BCD 码，分别存放在内部数据存储器的 50H～51H 单元和 60H～61H 单元中，试编写求这两数之和的程序，结果存放到 40H～41H 单元中。

解　求两个 BCD 码之和的程序如下：

```
MOV     A,50H       ;A←(50H)
ADD     A,60H       ;A←(A)+(60H)
DA      A           ; BCD 码修正
MOV     40H,A
MOV     A,51H
ADDC    A,61H
DA      A
MOV     41H,A
```

5. 加 1 指令

```
INC     A           ;  0000 0100 ,  A←(A)+1
```

INC	Rn	;	0000 1rrr	, Rn←(Rn)+1
INC	direct	;	0000 0101 direct	, direct←(direct)+1
INC	@Ri	;	0000 011i	, (Ri)←((Ri))+1
INC	DPTR	;	1010 0011	, DPTR←(DPTR)+1

加1指令的功能是将指定单元的内容加1，结果再放回到该单元中去。这类指令不影响标志位。

6. 减1指令

DEC	A	;	0001 0100	, A←(A)−1
DEC	Rn	;	0001 1rrr	, Rn←(Rn)−1
DEC	direct	;	0001 0101 direct	, direct←(direct)−1
DEC	@Ri	;	0001 011i	, (Ri)←((Ri))−1

减1指令的功能是将指定单元的内容减1，结果再放回到该单元中去。这类指令不影响标志位。

7. 乘、除指令

乘、除指令是MCS-51单片机新增加的指令，它们的指令周期为4个机器周期。乘、除指令的设置大大增强了MCS-51单片机的运算功能。

乘法指令：

MUL AB ; 1010 0100 , $\left.\begin{matrix}(A)_{7\sim0}\\(B)_{15\sim8}\end{matrix}\right\}\leftarrow(A)\times(B)$

乘法指令的功能是将累加器A和寄存器B中的8位无符号整数进行相乘，16位积的低8位存于A中，高8位存于B中。如果积大于255(FFH)，则溢出标志位OV置1，否则清0。进位标志位Cy总是清0。

例如：设(A)=4EH，(B)=5DH，执行指令：

MUL AB

结果为 (A)=56H (B)=1CH OV=1

除法指令：

DIV AB ; 1000 0100 , (A)/(B)

该指令的功能是累加器A中8位无符号整数除以B寄存器中8位无符号整数，所得到的商的整数部分存于A中，余数部分存于B中。标志位Cy和OV清0(当除数为0时OV置1)。

例如：设(A)=B6H，(B)=17H，执行指令：

DIV AB

结果为 (A)=07H (B)=15H Cy=0 OV=0

表 5-2 算术运算类指令一览表

助记符	功能说明	字节数	振荡器周期
ADD A,Rn	寄存器内容加到累加器	1	12
ADD A,direct	直接寻址字节内容加到累加器	2	12
ADD A,@Ri	间接寻址 RAM 内容加到累加器	1	12
ADD A,#data	立即数加到累加器	2	12
ADDC A,Rn	寄存器加到累加器(带进位)	1	12
ADDC A,direct	直接寻址字节加到累加器(带进位)	2	12
ADDC A,@Ri	间接寻址 RAM 加到累加器(带进位)	1	12
ADDC A,#data	立即数加到累加器(带进位)	2	12
SUBB A,Rn	累加器内容减去寄存器内容(带借位)	1	12
SUBB A,direct	累加器内容减去直接寻址字节(带借位)	2	12
SUBB A,@Ri	累加器为容减去间接寻址 RAM(带借位)	1	12
SUBB A,#data	累加器减去立即数(带借位)	2	12
DA A	累加器十进制调整	1	12
INC A	累加器加 1	1	12
INC Rn	寄存器加 1	1	12
INC direct	直接寻址字节加 1	2	12
INC @Ri	间接寻址 RAM 加 1	1	12
INC DPTR	地址寄存器加 1	1	24
DEC A	累加器减 1	1	12
DEC Rn	寄存器减 1	1	12
DEC direct	直接寻址地址字节减 1	2	12
DEC @Ri	间接寻址 RAM 减 1	1	12
MUL AB	累加器 A 和寄存器 B 相乘	1	48
DIV AB	累加器 A 除以寄存器 B	1	48

三、逻辑运算指令

逻辑运算指令包括与、或、异或,循环、累加器清零与求反指令。这些指令中的操作数都是 8 位,它们在执行时,不影响标志位。

1. 逻辑与指令

```
ANL   A,Rn          ;  [0101 1rrr]                    ,  A←(A)∧(Rn)
ANL   A, direct     ;  [0101 0101][direct]            ,  A←(A)∧(direct)
ANL   A,@Ri         ;  [0101 011i]                    ,  A←(A)∧((Ri))
ANL   A,#data       ;  [0101 0100][data]              ,  A←(A)∧data
ANL   direct,A      ;  [0101 0010][direct]            ,  direct←(direct)∧(A)
ANL   direct,#data  ;  [0101 0011][direct][data]      ,  direct←(direct)∧data
```

例 5-12 将累加器 A 中的压缩 BCD 码分为二个字节,形成非压缩 BCD 码,放入

30H 和 31H 单元中。

解 由题意，将累加器 A 中的低 4 位保留，高 4 位清零放入 30H；高 4 位保留，低 4 位清零，半字节交换后存入 31H 单元中。程序为：

```
MOV   40H,A          ;保存 A 中的内容
ANL   A,#00001111B   ;清高 4 位，保留低 4 位
MOV   30H,A
MOV   A,40H          ;取原数据
ANL   A,#11110000B   ; 保留高 4 位，清低 4 位
SWAP  A
MOV   31H,A
```

2. 逻辑或指令

ORL	A,Rn	;	0100 1rrr	, A←(A)∨(Rn)
ORL	A, direct	;	0100 0101 direct	, A←(A)∨(direct)
ORL	A,@Ri	;	0100 011i	, A←(A)∨((Ri))
ORL	A,#data	;	0100 0100 data	, A←(A)∨data
ORL	direct,A	;	0100 0010 direct	, direct←(direct)∨(A)
ORL	direct,#data	;	0100 0011 direct data	, direct←(direct)∨data

例 5-13 将累加器 A 中的低 4 位由 P1 口的低 4 位输出，P1 口高 4 位不变。

解 据题意程序如下：

```
ANL   A,#00001111B
MOV   30H,A
MOV   A,P1
ANL   A,#11110000B
ORL   A,30H
MOV   P1,A
```

3. 逻辑异或指令

XRL	A,Rn	;	0110 1rrr	, A←(A)∀(Rn)
XRL	A, direct	;	0110 0101 direct	, A←(A)∀(direct)
XRL	A,@Ri	;	0110 011i	, A←(A)∀((Ri))
XRL	A,#data	;	0110 0100 data	, A←(A)∀data
XRL	direct,A	;	0110 0010	, A←(direct)∀(A)

			direct		
XRL	direct,#data	;	0110 0011	,	direct←(direct)∀data
			direct		
			data		

例如：累加器 A 的内容为 6BH，执行指令：

XRL A,#10101010B

累加器 A 中的内容为 C1H。

4. 移位指令

累加器 A 循环左移指令：

RL	A	;	0010 0011	,	a_{n+1}←(a_n) (n=0～6) a_0←(a_7)

累加器 A 带进位标志位的循环左移指令：

RLC	A	;	0011 0011	,	a_n←(a_{n+1}) (n=0～6) a_0←(Cy),Cy←(a_7)

累加器 A 循环右移指令：

RR	A	;	0000 0011	,	a_n←(a_{n+1}) (n=0～6) a_7←a_0

累加器 A 带进位标志位的循环右移指令：

RRC	A	;	0001 0011	,	a_n←(a_{n+1}) (n=0～6) a_7←(Cy),Cy←(a_0)

例如：累加器 A 中的内容为 01100111，执行指令

RL A

结果为 (A)=11001110

又如：(A)=01100110,(Cy)=1，执行指令：

RRC A

结果为 (A)=10110011(Cy)=0

5. 累加器 A 清零与取反指令

累加器清零指令：

CLR	A	;	1110 0100	,	A←00H

累加器取反指令：

CPL	A	;	1111 0100	,	A←$(\overline{A})$

四、控制和转移指令

控制和转移指令包括无条件转移指令、条件转移指令、比较转移指令、循环转移指令及调用与返回指令。这类指令通过修改 PC 的内容来控制程序的执行过程，可极大地提高程序的效率，实现复杂的功能。

1. 无条件转移指令

16 位地址的无条件转移指令：

表 5-3　逻辑运算类指令一览表

助记符	功能说明	字节数	振荡器周期
ANL A,Rn	寄存器"与"到累加器	1	12
ANL A,direct	直接寻址字节"与"到累加器	2	12
ANL A,@Ri	间接寻址 RAM"与"到累加器	1	12
ANL A,#data	立即数"与"到累加器	2	12
ANL direct,A	累加器"与"到直接寻址字节	2	12
ANL direct,#data	立即数"与"到直接寻址字节	3	24
ORL A,Rn	寄存器"或"到累加器	1	12
ORL A,direct	直接寻址字节"或"到累加器	2	12
ORL A,@Ri	间接寻址 RAM"或"到累加器	1	12
ORL A,#data	立即数"或"到累加器	2	12
ORL direct,A	累加器"或"到直接寻址字节	2	12
ORL direct,#data	立即数"或"到直接寻址字节	3	24
XRL A,Rn	寄存器"异或"到累加器	1	12
XRL A,direct	直接寻址字节"异或"到累加器	2	12
XRL A,@Ri	间接寻址 RAM"异或"到累加器	1	12
XRL A,#data	立即数"异或"到累加器	2	12
XRL direct,A	累加器"异或"到直接寻址字节	2	24
XRL direct,#data	立即数"异或"到直接寻址字节	3	12
RL A	累加器循环左移	1	12
RLC A	经过进位位的累加器循环左移	1	12
RR A	累加器循环右移	1	12
RRC A	经过进位位的累加器循环右移	1	12
CLR A	累加器清零	1	12
CPL A	累加器求反	1	12

LJMP　addr16　　；

0000 0010
$addr_{15\sim8}$
$addr_{7\sim0}$

，$PC \leftarrow addr_{15\sim0}$

在这条指令中，地址是 16 位的，因此目标地址的选择范围为 64K 空间的任意单元。该指令不影响标志位。

11 位地址的无条件转移指令：

AJMP　addr11　　；

a_{10} a_9 a_8 0 0 0 0 1
a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0

，$PC \leftarrow (PC)+2$

$PC_{10\sim0} \leftarrow addr_{11}$

指令中的地址为 11 位，其目标地址在指令的 2K 地址空间内。必须注意的是在形成目标地址之前 PC 的内容先加 2。

例如：在程序存储器的 0123H 单元中放入一条指令：AJMP addr11。如图 5-7 所示，从指令中可以得到 11 位地址为 01101111000，由它来取代该指令后的地址 0125H 的低 11 位，形成目标地址：0378H，即程序将跳转到 0378H 单元继续执行。

相对转移指令：

SJMP　rel　　；

1000 0000

，$PC \leftarrow (PC)+2$

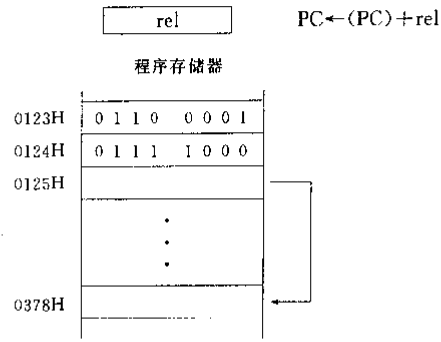


图5-7 跳转指令的执行过程

指令中 rel 是相对地址，是一个有符号的偏移量，其范围为－128～＋127，以补码的形式存在。正数表示程序向后跳转，负数表示向前跳转。

如在 0123H 单元存放着指令 AJMP 45H，则目标地址为 0123H＋45H＝0168H。若指令为 SJMP F2H，则目标地址为 0123H－0EH＝0114H。

间接转移指令：

JMP @A＋DPTR ；[0111 0011] ，PC←(A)＋(DPTR)

该指令的目标地址由地址寄存器 DPTR 和累加器 A 的内容相加形成。该指令不改变 DPTR 和 A 中的值，也不影响标志位。这条指令可以根据累加器 A 中的值控制程序转向不同的程序段，因此有时也称为散转指令。

例如下面一段程序：

```
        CLR     C               ;清进位位
        RLC     A               ;累加器内容乘 2
        MOV     DPTR,#TABLH
        JMP     @A+DPTR
TABLH:  AJMP    KEY0
        AJMP    KEY1
        AJMP    KEY2
         ⋮
```

其功能是：当(A)＝00H 时，程序散转到 KEY0；当(A)＝01H 时，散转到 KEY1；等等。

2. 条件转移指令

累加器为零转移指令：

JZ rel ；[0110 0000] ，若(A)＝0，则 PC←(PC)＋2＋rel
[rel] 若(A)≠0，则 PC←(PC)＋2

累加器非零转移指令：

JNZ rel ；

0111 0000
rel

，若(A)≠0，则 PC←(PC)+2+rel
若(A)=0，则 PC←(PC)+2

执行这两条指令时，首先对累加器内容进行判断，满足条件则转移，否则程序顺序执行。rel 是相对地址，为补码形式。

3. 比较转移指令

累加器内容与直接地址单元中的内容不等转移指令：

CJNE A,direct,rel ；

1011 0101
direct
rel

，若(direct)<(A)，
则 PC←(PC)+3+rel，且 Cy←0
若(direct)>(A)，
则 PC←(PC)+3+rel，且 Cy←1
若(direct)=(A)，则 PC←(PC)+3

累加器内容与立即数不等转移指令：

CJNE A，#data,rel ；

1011 0100
data
rel

，若 data<(A)，
则 PC←(PC)+3+rel，且 Cy←0
若 data>(A)，
则 PC←(PC)+3+rel，且 Cy←1
若 data=(A)，则 PC←(PC)+3

寄存器内容与立即数的内容不等转移指令：

CJNE Rn，#data,rel ；

1011 1rrr
data
rel

，若 data<(Rn)，
则 PC←(PC)+3+rel，且 Cy←0
若 data>(Rn)，
则 PC←(PC)+3+rel，且 Cy←1
若 data=(Rn)，则 PC←(PC)+3

间接寻址单元内容与立即数不等转移指令：

CJNE @Ri，#data,rel ；

1011 011i
data
rel

，若 data<((Ri))，
则 PC←(PC)+3+rel，且 Cy←0
若 data>((Ri))，
则 PC←(PC)+3+rel，且 Cy←1
若 data=((Ri))，则 PC←(PC)+3

这组指令在执行过程中，首先对两个 8 位无符号数进行比较，不等时，按偏移地址转移，并根据两数大小确定 Cy 的值，相等时，程序顺序执行。

4. 循环转移指令

DJNZ Rn,rel ；

1101 1rrr
rel

，Rn←(Rn)−1
若 Rn=0，则 PC←(PC)+2
若 Rn≠0，则 PC←(PC)+2+rel

DJNZ direct,rel ；

1101 0101
direct
rel

，direct ←(direct)−1
若(direct)=0，则 PC←(PC)+2
若(direct)≠0，
则 PC←(PC)+2+rel

在执行这两条指令时，首先将第一个操作数减 1，判断结果是否为 0，若为 0，程序顺序执行，若不为 0，程序按偏移地址转移。这两条指令可以用于构成循环程序，循环的次数

就是第一个操作数的值。

5. 调用与返回指令

绝对调用指令：

ACALL addr11 ；

a_{10} a_9 a_8 1 0 0 0 1	,PC←(PC)+2,SP←(SP)+1
a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	(SP)←(PC)$_{7\sim0}$,SP←(SP)+1
	(SP)←(PC)$_{15\sim8}$,PC$_{10\sim0}$←addr$_{11}$

该指令的目标地址的形成与 11 位地址的无条件转移指令类似，增加断点压栈过程。指令的执行不影响标志位。

LCALL addr16 ；

0001 0010	,PC←(PC)+2,SP←(SP)+1
addr$_{15\sim8}$	(SP)←(PC)$_{7\sim0}$,SP←(SP)+1
addr$_{7\sim0}$	(SP)←(PC)$_{15\sim8}$,PC$_{15\sim0}$←addr$_{16}$

该指令除了压栈操作外与 16 位地址的无条件转移指令的执行过程相同。指令的执行不影响标志位。上述两条指令用于调用一段子程序。

子程序返回指令：

RET ；

0010 0010	,PC$_{15\sim8}$←((SP)),SP←(SP)−1
	PC$_{7\sim0}$←((SP)),SP←(SP)−1

这条指令将堆栈顶的 2 个字节的内容送到 PC 中。用于子程序的末尾，使程序返回到调用处。

中断返回指令：

RETI ；

0011 0010	,PC$_{15\sim8}$←((SP)),SP←(SP)−1
	PC$_{7\sim0}$←((SP)),SP←(SP)−1

该指令将堆栈顶的 2 个字节的内容送到 PC 中，其操作与 RET 指令相同，不同的是它同时释放中断逻辑，使同级中断可以被接受。它只能用于中断服务子程序的末尾。

6. 空操作指令

NOP ；

0000 0000	,PC←(PC)+1

这条指令除了使 PC 内容加 1 外，不进行任何操作。常用于产生一个机器周期的延时。

五、位操作指令

位操作指令包括位数据传送指令、位逻辑操作指令、位条件转移指令。指令中的操作数都是 1 位的。

1. 位数据传送指令

MOV C,bit ；

1010 0010	,C←(bit)
bit	

MOV bit,C ；

1001 0010	,bit←(C)
bit	

指令中 C 即进位标志位 Cy，bit 为内部 RAM 中 20H～2FH 单元中 128 个可寻址位和特殊功能寄存器中的可寻址位的地址。

表 5-4 控制转移类指令一览表

助记符	功能说明	字节数	振荡器周期
LJMP addr16	长转移	3	24
AJMP addr11	绝对转移	2	24
SJMP rel	短转移(相对偏移)	2	24
JMP @A+DPTR	相对 DPTR 的间接转移	1	24
JZ rel	累加器为零则转移	2	24
JNZ rel	累加器为非零则转移	2	24
CJNE A,direct,rel	比较直接寻址字节和 A 不相等则转移	3	24
CJNE A,#data,rel	比较立即数和 A 不相等则转移	3	24
CJNE Rn,#data,rel	比较立即数和寄存器不相等则转移	3	24
CJNE @Ri,#data,rel	比较立即数和间接寻址 RAM 不相等则转移	3	24
DJNZ Rn,rel	寄存器减 1 不为零则转移	2	24
DJNZ direct,rel	直接寻址字节减 1 不为零则转移	3	24
ACALL addr11	绝对调用子程序	2	24
LCALL addr16	长调用子程序	3	24
RET	从子程序返回	1	24
RETI	从中断返回	1	24
NOP	空操作	1	12

2. 位逻辑操作指令

位逻辑与指令：

ANL C,bit ; 1000 0010 / bit ,C←(C) ∧ (bit)

ANL C,/bit ; 1011 0000 / bit ,C←(C) ∧ ($\overline{bit}$)

位逻辑或指令：

ORL C,bit ; 0111 0010 / bit ,C←(C) ∨ (bit)

ORL C,/bit ; 1010 0000 / bit ,C←(C) ∨ ($\overline{bit}$)

位清 0 指令：

CLR C ; 1100 0011 ,C←0

CLR bit ; 1100 0010 / bit ,bit←0

位置 1 指令：

SETB C ; 1101 0011 ,C←1

```
SETB    bit          ;  1101 0010   bit←1
                        bit
```

位取反指令：

```
CPL     C            ;  1011 0011   ,C←(C̄)
CPL     bit          ;  1011 0010   ,bit←(b̄it)
                        bit
```

3. 位条件转移指令

进位标志位为 1 转移指令：

```
JC      rel          ;  0100 0000   ,若(C)=1,则 PC←(PC)+2+rel
                        rel         若(C)=0,则 PC←(PC)+2
```

进位标志位为 0 转移指令：

```
JNC     rel          ;  0101 0000   ,若(C)=0,则 PC←(PC)+2+rel
                        rel         若(C)=1,则 PC←(PC)+2
```

直接寻址位为 1 转移指令：

```
JB      bit,rel      ;  0010 0000   ,若(bit)=1,则 PC←(PC)+3+rel
                        bit         若(bit)=0,则 PC←(PC)+3
                        rel
```

直接寻址位内容为 0 转移指令：

```
JNB     bit,rel      ;  0011 0000   ,若(bit)=0,则 PC←(PC)+3+rel
                        bit         若(bit)=1,则 PC←(PC)+3
                        rel
```

直接寻址位内容为 1 转移并清 0 该位指令：

```
JBC     bit,rel      ;  0001 0000   ,若(bit)=0,则 PC←(PC)+3
                        bit         若 bit=1,则 PC←(PC)+3+rel,
                        rel         且 bit←0
```

这类指令与位累加器 C 和可位寻址位构成了一个完整的位处理机，大大增强了 MCS-51 单片机的位处理功能。

例 5-14 用单片机来实现下列电路的逻辑功能。

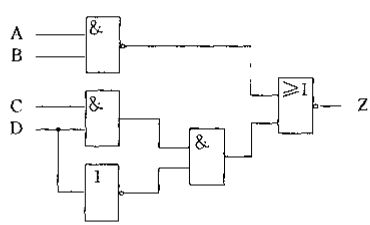


图5-8 逻辑电路

解 为了使逻辑问题适合单片机来处理，先来选择一些端口位作为输入逻辑变量和输出逻辑变量。

设 P1.0=A,P1.3=D,P1.1=B,P1.4=Z,P1.2=C,程序为:

```
MOV   C,P1.0        ;读入变量 A
ANL   C,P1.1
CPL   C
MOV   30H,C         ;保存中间运算结果
MOV   C,P1.2
ANL   C,P1.3
CPL   C
ANL   C,/P1.3
ORL   C,30H
CPL   C
MOV   P1.4,C        ;输出运算结果
```

思考题与习题

1. 什么是指令系统？MCS-51 单片机共有多少种指令？
2. 什么是寻址方式？MCS-51 单片机有哪几种寻址方式？
3. 指出下列指令中画线的操作数的寻址方式。

```
MOV   R0,#60H
MOV   A,30H
MOV   A,@Ri
MOV   @R1,A
ADD   A,B
SUBB  A,R7
```

4. 指出下列指令中画线的操作数的寻址方式。

```
MOVX A,@DPTR
MOV  DPTR,#0123H
MOVC A,@A+DPTR
MUL  A B
INC  DPTR
```

5. 指出下列指令中画线的操作数的寻址方式。

```
SJMP  NEXT
JZ    AB
CJNE  A,#00H,ONE
CPL   C
MOV   C,30H
```

6. 说明下列指令的作用。

```
MOV  30H,@R0
SWAP A
XCH  A,@R1
```

7. 已知内部数据存储器 30H 和 40H 单元的内容分别为 70H 和 71H，执行下列一段程序后，试分析有关单元内容。

```
MOV  R0,#30H
MOV  A,@R0
MOV  @R0,40H
MOV  40H,A
MOV  R0,#60H
```

8. 试编写一段程序，将内部数据存储器 30H、31H 单元内容传送到外部数据存储器 1000H、1001H 单元中去。

9. 试编写一段程序，将外部数据存储器 40H 单元中的内容传送到 0100H 单元。

10. 将 A 中的 2 位压缩 BCD 码分开后送入 60H 和 61H 单元的低位。

11. 试编写一段程序，将 R3R2 中的 4 位 BCD 码倒序排列。

12. 加法和减法指令影响那些标志位？怎么影响的？

13. 试编写计算下列算式的程序。

(1)1F0AH+B2ECH

(2)8ED0H−324FH

14. 试编写计算下列算式的程序。

(1)8250+6632

(2)2411+6897

15. 试分析执行下列指令后，标志位的内容。

(1)已知(A)=C3H，ADD A，#0AAH

(2)已知(A)=C9H，SUBB A，#54H

16. 试编写一段程序，将累加器 A 的高 4 位由 P1 口的高 4 位输出，P1 口低 4 位保持不变。

17. 试编写一段程序，将 P1 口的高 5 位置位，低 3 位不变。

18. 试编写一段程序，将累加器 A 中的负数转换为其补码。

19. 试编写一段程序，将 R3R2 中的双字节负数转换成补码。

20. 试编写一段程序，将 R2 中的各位倒序排列后送入 R3 中。

21. 试编写一段程序，将 R2 中的数乘 4(用移位指令)。

22. 已知指令 AJMP addr11 的机器码为 41H 和 FFH，指令所在的地址为 0810H，求转移指令的目的地址。

23. 已知指令 SJMP 70H 所在的地址为 0100H，求其转移的目的地址。

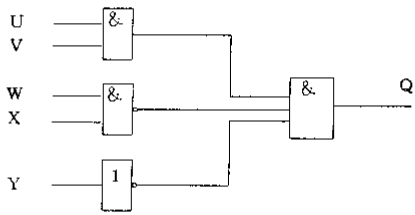
24. 已知指令 SJMP A0H 所在的地址为 0100H，求其转移的目的地址。

25. 求指令 H:SJMP H 中的地址偏移量。

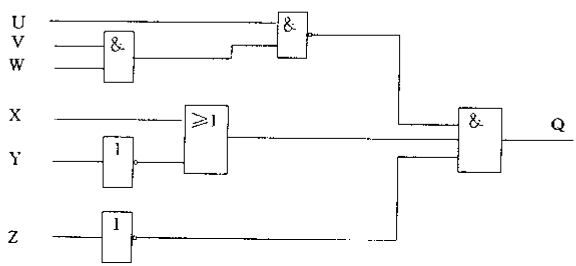
26. 试比较指令 AJMP addr11 和 ACALL addr11 的不同。

27. 试用单片机来实现下列逻辑电路的功能。

(1)



(2)



注:图中逻辑变量可设为单片机 P1 口的各位。

28. 试编写一段程序,完成下列逻辑运算。

(1) $Y=X_0 \cdot X_1 \cdot \overline{X}_2 \cdot \overline{X}_3+X_4 \cdot X_5+X_6$

(2) $Y=\overline{X_0 \cdot X_1+\overline{X}_2 \cdot \overline{X}_3 \cdot X_4+\overline{X}_4 \cdot \overline{X}_5 \cdot X_6+\overline{X}_6 \cdot \overline{X}_7}$

第六章　汇编语言程序设计

计算机软件可以分为系统软件和用户软件两部分。系统软件中的操作系统是由计算机制造厂商提供的，是必不可少的。与其它微型计算机不同的是，单片机没有像监控系统或操作系统那样的系统软件，所有的单片机程序均需用户设计完成。因此，程序设计就成为单片机应用不可缺少的内容。虽然许多单片机开发系统提供了一些高级语言，但目前被广泛采用的仍然是汇编语言。本章通过例举大量的程序设计实例，来说明汇编语言的设计过程。

第一节　汇编语言的基本知识

一、汇编语言及其语句结构

计算机能识别的是用二进制表示的指令，它们被称为代码指令或机器码。机器码虽然能被计算机直接识别，但在书写、阅读、记忆上都很困难，用它来编写程序也很不便。为了解决这一问题，人们用英文字母来代替机器码，这些英文字母称为助记符。汇编语言是用助记符来表示指令的一种计算机语言。它由汇编语句组成，这些语句在书写上有一定的要求，这就是汇编语句的结构。

一条汇编语言的语句最多包括四部分：标号、操作码、操作数和注释，其结构为：

标号：　操作码　操作数；　注释

①标号位于语句的开始，由字母和数字组成，它代表该语句的地址。标号必须由字母打头，冒号结束，字母和数字的总数不应超过一定数量，一般标号不能为助记符。标号不是语句必要的组成部分。

②操作码在标号之后，是指令的助记符，表示语句的性质，是语句的核心。没有标号时，它作为语句的开始。

③操作数在操作码之后，二者用空格分开。操作数既可以是数据，也可以是地址，且必须满足寻址方式的规定。有多个操作数时，操作数之间用“,”分开。

指令中的常数可以是十进制、十六进制、二进制或八进制数，具体格式如下：

二进制常数以 B 结尾，如：10100011B；

十六进制常数以 H 结尾，如：65H、0F1H；

十进制常数以 D(可以省略)结尾，如：65D 或 65；

八进制常数以 Q 结尾，如 175Q。

字符串常数用‘ ’表示，如‘A’表示 A 的 ASCII 码。

④注释在语句的最后，以“;”开始，是说明语句的功能和性质的文字。

例如：

START:MOV A,#30H;A←30H

START 为标号，它以“:”结束，表示该指令的地址；MOV 为用助记符表示的操作码，表示指令的功能为数据传送；A 和 #30H 为操作数；A←30H 则为注释，它以“;”开始，说明这条语句的功能。

二、伪指令

所谓伪指令是指由汇编程序提供的，在汇编时起作用，在执行时不起作用的一类指令。其特点是没有对应的机器码。伪指令提供了像规定程序地址、建立数据表格一类的功能，为汇编语言程序的编写提供了方便。

1. 定位伪指令

格式：ORG n

n 为十进制或十六进制常数，代表地址。该伪指令规定了后面指令的地址。例如：

```
ORG  0100H
AJMP PRG1
```

AJMP 为双字节指令，其首字节放在 0100H 单元，第二个字节放在 0101H 单元。

2. 汇编结束伪指令

格式：END

当汇编程序遇到该指令后，结束汇编过程，其后的指令将不加处理。

3. 定义字节伪指令

格式： DB X1,X2,…,Xn

其中 Xi 为 8 位数据或 ASCII 码。

例如：

```
        ORG  1000H
        DB 01H,02H
则      (1000H)=01H
        (1001H)=02H
```

又如：

```
        ORG  1100H
        DB  '01'
则      (1100H)=30H    0 的 ASCII 码
        (1101H)=31H    1 的 ASCII 码
```

4. 定义双字节伪指令

格式： DW X1,X2,…,Xn

其中 Xi 为双字节数据。

例如：

```
        ORG   2000H
        DW    2546H,0178H
```

则

```
(2000H)=25H
(2001H)=46H
(2002H)=01H
(2003H)=78H
```

5. 单字节数据赋值伪指令

格式： x EQU n

x 为用户定义的标号，n 为常数、工作寄存器或特殊功能寄存器，为单字节数。

该伪指令将 n 的值赋给标号 x。

x 可用于指令中，作为单字节操作数；包括立即数和直接地址 direct。

6. 双字节数据赋值伪指令

格式： y EQU n

y 为用户定义的标号，n 为双字节常数。

该伪指令将 n 的值赋给标号 y。

y 可作为指令中的双字节操作数，或转移指令中的目的地址。

需要注意的是，不同的汇编程序提供的伪指令可能不同，使用时请参考相应的说明。

三、汇编语言程序设计过程

程序是指令的有序集合。一个好的程序不仅应完成规定的功能，而且还应该占内存最少、执行时间最短。一般情况下，程序设计过程可分为以下几个步骤：

1. 分析课题，确定算法或解题思路

实际问题是多种多样的，不可能有统一的模式，必须具体问题具体分析。对于同一个问题，也存在多种不同的解决方案，应通过认真比较从中挑选最佳方案。这是程序设计的基础。

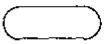
2. 画流程图

流程图又称程序框图。它可以直观地表示出程序的执行过程或解题步骤和方法。同时，它给出程序的结构，体现整体与部分之间的关系，将复杂的程序分成若干简单的部分，给编程工作带来方便。流程图还充分地表达了程序的设计思路，将问题与程序联系起来，便于我们阅读、理解程序，查找错误。因此，画流程图是程序设计的一种简单、易行、有效的方法。

3. 编写程序

根据流程图中各部分的功能，写出具体程序，再由流程图给出的各部分之间的关系，整理出全部程序。

流程图中经常使用的几种符号：

端点符号  表示程序的开始或结束

处理符号  表示处理过程

判断符号  表示判断

第二节　简单程序设计

简单程序是指那些按顺序依次执行的程序，也称为顺序程序或直线程序。

例 6-1　求多项式 $y=a^2-b$。

解　设 a 存放在 R2 中，b 在 R3 中，结果放入 R6 和 R7 中。

流程图如图 6-1 所示。

程序为：

```
MOV    A,R2    ;A←a
MOV    B,A
MUL    AB
CLR    C
SUBB   A,R3
MOV    R7,A    ;R7←结果的低 8 位
MOV    A,B
SUBB   A,#00H;高 8 位减进位位
MOV    R6,A
END
```

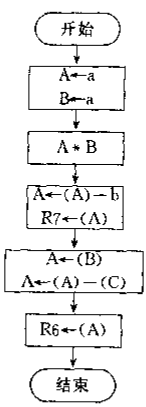


图6-1　例6-1流程图

例 6-2　单字节十六进制数转换为 BCD 码。

解　单字节十六进制数在 0～255 之间，将其除 100 后，商即百位数，余数除以 10，商为十位数，余数即个位数。

设单字节数在累加器 A 中，转换结果的百位数放在 R3 中，十位和个位数则放入 A 中。

流程图如图 6-2 所示。

程序为：

```
HBCD:  MOV    B,#100 ;分离出百位数
       DIV    AB
       MOV    R3,A    ;R3←百位数
       MOV    A,#10  ;分离十位和个位数
       XCH    A,B
```

```
        DIV     AB
        SWAP    A
        ADD     A,B
        RET
```

该程序写成子程序形式。

例 6-3 无符号双字节乘法。

解 设乘数和被乘数分别放在R2、R3 和 R6、R7 中，计算结果放入R4、R5、R6、R7 中。

双字节无符号数的乘法可以采用重复加法的方法进行。考虑到 MCS-51 单片机设置了乘法指令，并根据乘法的运算过程：

$$
\begin{array}{rcccc}
 & & (R2 & & R3) \\
 & \times & (R6 & & R7) \\
\hline
 & & (R3*R7)_H & (R3*R7)_L & \\
 & (R2*R7)_H & (R2*R7)_L & & \\
 & (R3*R6)_H & (R3*R6)_L & & \\
+ & (R2*R6)_H & (R2*R6)_L & & \\
\hline
 & R4 & R5 & R6 & R7
\end{array}
$$

本程序的流程图如图 6-3 所示。

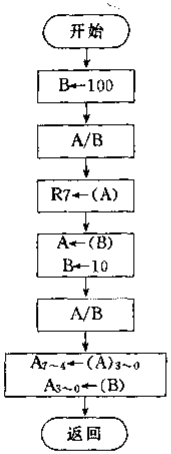


图6-2 例6-2流程图

程序为：

```
MUL:    MOV   A,R3
        MOV   B,R7
        MUL   AB
        XCH   A,R7
        MOV   R5,B
        MOV   B,R2
        MUL   AB
        ADD   A,R5
        MOV   R4,A
        CLR   A
        ADDC  A,B
        MOV   R5,A
        MOV   A,R6
```

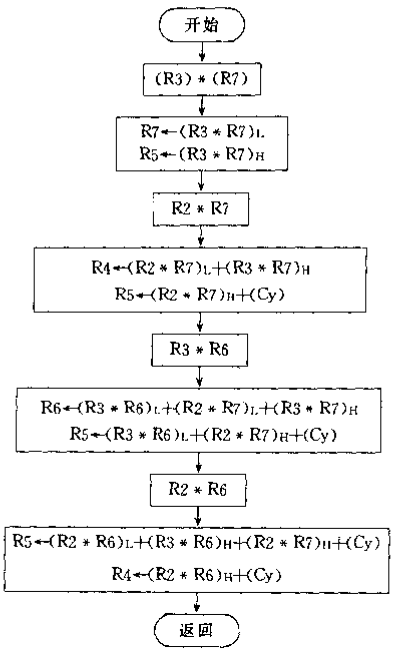


图6-3　例6-3流程图

```
MOV   B,R3
MUL   AB
ADD   A,R4
XCH   A,R6
XCH   A,B
ADDC  A,R5
MOV   R5,A
MOV   F0,C
MOV   A,R2
MUL   AB
ADD   A,R5
MOV   R5,A
CLR   A
MOV   ACC.0,C
MOV   C,F0
```

```
        ADDC   A,B
        MOV    R4,A
        RET
```

我们看到，此程序除使用用户标志外，相关单元外只使用了标志位。为做到这一点，程序中多次使用字节交换指令，中间运算结果只使用 R2～R7 中的工作寄存器，使改变内容的单元达到最少。

该方法很容易推广到多字节无符号数的乘法运算中。

第三节　分支程序设计

在许多情况下，需要根据不同的条件转向不同的处理程序，这种结构的程序称为分支程序。MCS-51 单片机中设置的条件转移指令、比较转移指令和位转移指令可以实现程序的分支。

例 6-4　求单字节有符号二进制数的补码。

解　正数补码是其本身，负数的补码是其反码加 1。因此，程序首先判断被转换数符号，负数进行转换，正数即为补码。

设二进制数放在累加器 A 中，其补码放回到 A 中。其程序框图如图 6-4 所示。

程序为：

```
CMPT:  JNB    ACC.7,NCH   ;(A)>0,不需转换
       MOV    C,ACC.7     ;保存符号
       MOV    00H,C       ;保存符号
       CPL    A
       ADD    A,#1
       MOV    C,00H
       MOV    ACC.7,C     ;恢复符号
NCH:   RET
```

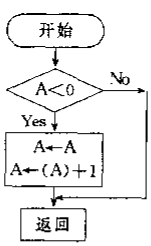


图6-4　例6-4流程图

例 6-5　求符号函数。

$$Y=\begin{cases}1 & \text{当 } X>0\\ 0 & \text{当 } X=0\\ -1 & \text{当 } X<0\end{cases}$$

解　X 存放在 30H 单元，Y 存放在 31H 单元。

流程图如图 6-5 所示。

程序为：

```
START: MOV    A,30H
       CJNE   A,#00H,NZ
       AJMP   LL
NZ:    JB     ACC.7,MM
```

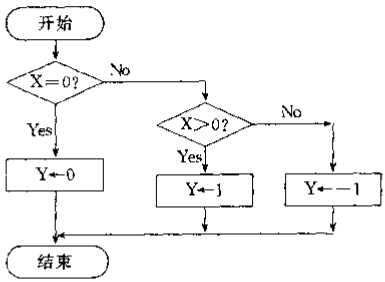


图6-5　例6-5流程图

```
        MOV   A,#01H
        AJMP  LL
MM:     MOV   A,#81H
LL:     MOV   31H,A
        END
```

例6-6　将ASCII码转换为十六进制数。如果不是十六进制数的ASCII码，用户标志位置1。

解　由ASCII码表可知，30H～39H为0～9的ASCII码，41H～46H为A～F的ASCII码。在这一范围内的ASCII码减30H或37H就可以获得对应的十六进制数。

设ASCII码放在累加器A中，转换结果放回A中。

程序流程图如图6-6所示。

程序为：

```
START:  CLR   C
        SUBB  A,#30H
        JC    NASC          ;(A)<0,不是十六进制数
        CJNE  A,#0AH,MM
MM:     JC    ASC           ;0≤(A)<0AH,是十六进制数
        SUBB  A,#07H
        CJNE  A,#0AH,NN
NN:     JC    NASC
        CJNE  A,#10H,LL
LL:     JC    ASC
NASC:   SETB  F0
ASC:    RET
```

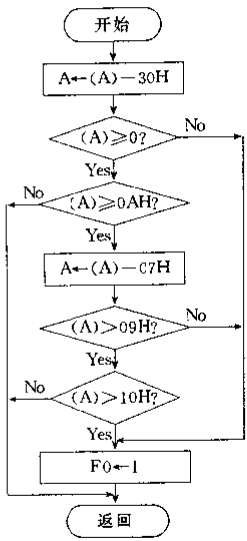


图6-6　例6-6程序流程图

第四节　循环程序设计

循环程序是指一段反复执行的程序。在许多问题中，需多次执行一段完全相同的程序，只是参加运算（或其它处理过程）的操作数不同。这时就可以采用循环程序结构。循环程序可以缩短程序，减少程序所占的内存空间。一般情况下，循环程序包括下面几部分：

（1）循环体：需要多次执行的程序的主体。

（2）循环控制：对循环次数进行计数，判断循环结束的条件。

（3）循环初值：包括循环次数、循环体中工作单元的初值等。

循环程序的一般结构如图 6-7 所示。

在许多情况下，循环程序中还可能包含着循环程序，这种现象称为循环嵌套。具有循环嵌套的程序称为多重循环程序。

例 6-7　将内部数据存储器 30H～7FH 单元的内容传送到外部数据存储器以 1000H 开始的连续单元中去。

解　30H～7FH 共计 80 个单元，需传送 80 次数据。将 R7 作为循环计数寄存器，流程图如图 6-8 所示。

程序为：

```
        MOV    R0,#30H
        MOV    DPTR,#1000H
```

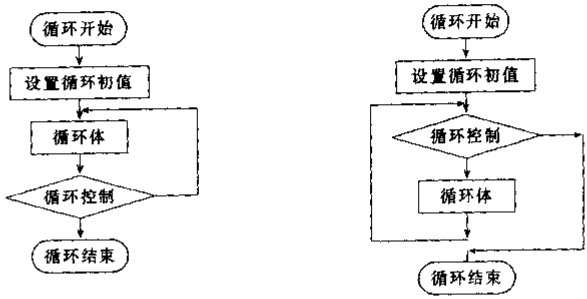


图6-7 循环程序结构

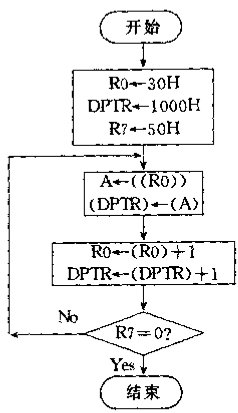


图6-8 例6-7程序流程图

```
        MOV   R7,#50H
LOOP:   MOV   A,@R0
        MOVX  @DPTR,A
        INC   R0            ;指向下一个数据
        INC   DPTR
        DJNZ  R7,LOOP       ;未完继续
        END
```

例 6-8 多字节 BCD 码加法。

解 设 R7 存放着 BCD 码的字节数(一个字节中存 2 位 BCD 码),R0 为被加数和结果的低位地址寄存器,R1 为加数的低位地址寄存器,Cy 保存最高字节的进位位。

流程图如图 6-9 所示。

程序为：

```
MADD:  MOV   A,R7
       MOV   R2,A          ;保存字节数
       MOV   A,R0          ;保存结果的地址指针
       MOV   R6,A
       CLR   C
LOOP:  MOV   A,@R0
       ADDC  A,@R1
       DA    A
       MOV   @R0,A
       INC   R0
       INC   R1
       DJNZ  R2,LOOP
       MOV   A,R6          ;恢复地址
       MOV   R0,A
       RET
```

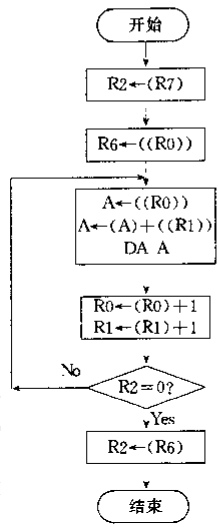


图6-9　例6-8程序流程图

例 6-9　内部数据存储器中连续存放着若干 8 位二进制数据，其中一个数的值为 X，试求出该数的地址。

解　设 R0 为数据区的首地址寄存器，R7 存放数据的个数，R6 存放 X 的地址。

将数据存储器中的内容取出，分别与 X 相比较，若相等，当前 R0 中的值即为所求的地址。程序的循环次数是不确定的，一旦找到 X，循环立即结束。程序的流程图如图 6-10 所示。

程序为：

```
START: MOV   A,R7          ;保存字节数
       MOV   R2,A
LOOP:  MOV   A,@R0
       CJNE  A,#X,LL
       SJMP  OUTL
LL:    INC   R0
       DJNZ  R2,LOOP
       SJMP  MM
OUTL:  MOV   A,R0
       MOV   R6,A
MM:    RET
```

例 6-10　试编写排序程序，将内部数据存储器中连续存放的若干数据由小到大排列起来。

解 将两个相邻的数据相比较，如果前数大于后数，两个数的位置互换，否则位置不变。所有数据比较完后，找出最大的数，并存在最后一个单元中。第二次比较在剩余的数据中进行，找到剩余数据中最大的数。以此类推，即可完成数据的排序。设数据的个数为n，第一次需比较的次数为n－1，第二次需n－2次，等等。在理论上，需要进行n－1次循环比较才能完成排序过程。事实上，有可能提前完成排序过程。如果在某次循环比较过程中没有发生位置互换，说明数据的排序已完成。为此在程序中设置一个数据换位标志位，当其发生变化时，排序即告完成。

设R0为数据区的首地址寄存器，R7为数据的字节数。

程序流程图如图6-11所示。

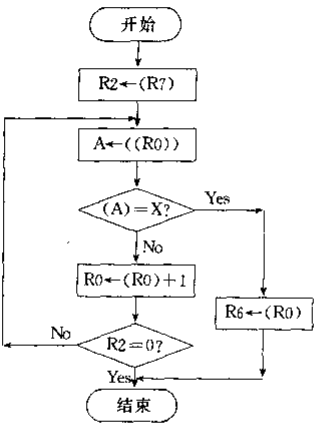


图6-10 例6-9程序流程图

程序为：

```
SS:   MOV   A,R7
      MOV   R2,A        ;保存字节数
      MOV   60H,R0      ;保存地址指针
NN:   DEC   R2
      MOV   A,R2
      MOV   R3,A        ;初始化循环次数
      MOV   R0,60H      ;恢复地址指针
L1:   CLR   F0
      MOV   A,@R0       ;取前数
      INC   R0
      CLR   C
      SUBB  A,@R0       ;减后数
      JC    MM
      MOV   A,@R0       ;位置互换
      DEC   R0
      XCH   A,@R0
      INC   R0
      MOV   @R0,A
      SETB  F0
MM:   DJNZ  R3,L1
      JB    F0,NN       ;排序未完转NN
      MOV   R0,60H      ;恢复地址指针
      RET
```

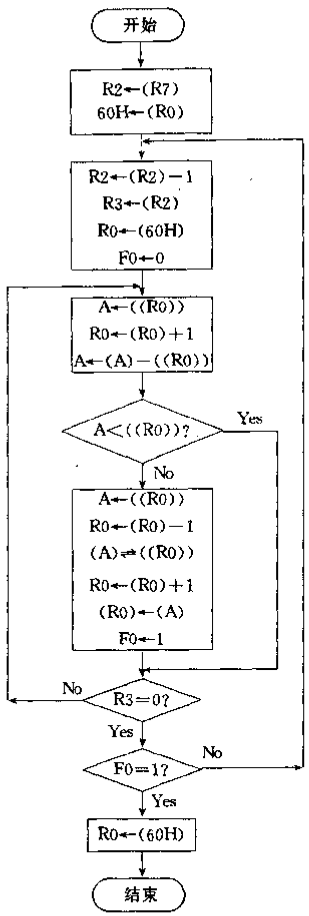


图6-11　例6-10程序流程图

本程序实际是一个双重循环程序，内循环的循环次数是变化的，外层循环没有采用循环指令，是一个循环次数不定的循环。

例 6-11　试编写延时程序。

计算机反复执行一段程序以达到延时的目的称为软件延时。通过控制执行指令的数量可以延时不同的时间。要实现较长时间的延时，一般需采用多重循环。下面是延时 50ms 的延时程序，设单片机的晶振为 12MHz。

```
DEL:    MOV    R7,#200
DEL1:   MOV    R6,#125
```

```
DEL2:   DJNZ   R6,DEL2
        DJNZ   R7,DEL1
        RET
```

该程序的延时时间实际为(125×2+1+2)×200+2=50.602ms。

第五节　查表程序设计

在单片机应用系统中，查表程序使用频繁。利用它能避免进行复杂的运算或转换过程，故它广泛应用于显示、打印字符的转换以及数据补偿、计算、转换等程序中。

查表就是根据自变量 x 的值，在表中查找 y，使 y=f(x)。x 和 y 可以是各种类型的数据。表的结构也是多种多样的。表格可以放在程序存储器中，也可以存放在数据存储器中。一般情况下，对自变量 x 是有变化规律的数据，可以根据这一规律形成地址，对应的 y 则存放于该地址单元中；对 x 是没有变化规律的数据，在表中存放 x 及其对应的 y 值。前者形成的表格是有序的，后者形成的表格可以是无序的。

例 6-12　将 1 位十六进制数转换为 ASCII 码。

解　在前面的例子中，我们介绍了将 ASCII 码转换为十六进数的程序，本例是其逆变换。这里采用查表的方法完成十六进制数到 ASCII 码的转换。

建立一个表格，首先确定表格的首地址，在相对于表首的地址单元中存放 ASCII 码。

设十六进制存放在 R0 中，转换结果存放在 R1 中。

程序为：

```
        ORG    0300H
        MOV    A,R0
        ANL    A,#0FH        ;屏蔽高位
        MOV    DPTR,#TAB
        MOVC   A,@A+DPTR
        MOV    R1,A
        ORG    0380H
TAB:    DB     '01234567'
        DB     '89ABCDEF'
        END
```

例 6-13　在一个单片机测温装置中，已知电压和温度之间是非线性关系。在校正过程中，电压值取连续的 10 位二进制数，在这些电压值下，测量对应温度最多可达 1024 个)。用这些校正数据建立一个表格，以电压为相对地址，这样就可以根据测得的不同电压值求出被测温度。

解　设电压测量值 X 放在 R2、R3 中(10 位二进制数占两个字节)，求出的温度仍放 R2、R3 中(也是双精度数)。与 X 值的对应的温度放在地址为 2X 加表格的首地址的单元中。

程序为：

```
        MOV   DPTR,#TAB
        MOV   A,R3
        CLR   C
        RLC   A              ;X*2
        MOV   R3,A
        XCH   A,R2
        RLC   A
        XCH   A,R2
        ADD   A,DPL          ;加表首地址
        MOV   DPL,A
        MOV   A,DPH
        ADDC  A,R2
        MOV   DPH,A
        CLR   A
        MOVC  A,@A+DPTR
        MOV   R2,A
        CLR   A
        INC   DPTR
        MOVC  A,@A+DPTR
        MOV   R3,A
        RET
TAB:    DW
        ⋮
```

例 6-14 通过一个键盘输入一组命令，完成不同的功能。

解 设从键盘输入的命令字符为'A'、'D'、'E'、'L'、'M'、'X'、'Z'七种，当键入不同的命令字符时转向不同的处理程序，对应的处理程序的入口标号为 XA、XD、XE、XL、XM、XX、XZ。由于输入的字符之间很难找到什么规律，建立表格时将字符和其对应的处理程序的地址一同存入。查表时先查找字符，其后就是处理程序的入口地址。表格以 0 为结束标志。

入口条件为：命令字符放在 A 中。

程序为：

```
LTB:    MOV   DPTR,#TAB
        MOV   B,A
LOOP:   CLR   A
        MOVC  A,@A+DPTR
        JZ    LEND
        INC   DPTR
        CJNE  A,B,LNF
```

```
        CLR   A
        MOVC  A,@A+DPTR   ;取出入口地址第一个字节
        MOV   B,A
        INC   DPTR
        CLR   A
        MOVC  A,@A+DPTR
        MOV   DPL,A
        MOV   DPH,B
        CLR   A
        JMP   @A+DPTR     ;转向相应处理程序
LNF:    INC   DPTR
        INC   DPTR        ;继续查
        SJMP  LOOP
LEND:   (查不到处理程序)
TAB:    DB    'A'
        DW    XA
        DB    'D'
        DW    XD
        DB    'E'
        DW    XE
        DB    'L'
        DW    XL
        DB    'M'
        DW    XM
        DB    'X'
        DW    XX
        DB    'Z'
        DW    XZ
        DB    0           ;表格结束标志
```

第六节　散转程序设计

散转程序是一种并行多分支程序。它根据系统的某种输入或运算结果，分别转向各个处理程序。与分支程序不同的是，散转程序多采用指令：JMP @A+DPTR，根据输入或运算结果，确定A或DPTR的内容，直接跳转到相应的分支程序中去。而分支程序一般是采用条件转移指令或比较转移指令实现程序的跳转。散转程序的基本结构如图6-12所示。在本节中，先给出一个散转程序的具体例子，然后再介绍几种常见的散转程序的实现方法。

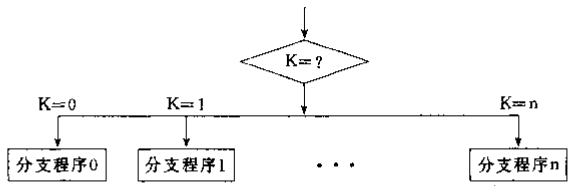


图6-12 散转程序的结构

例 6-15 单片机四则运算系统。

解 在单片机系统中设置+、-、×、÷、四个运算命令键，它们的键号分别为 0、1、2、3。当其中一个键按下时，进行相应的运算。操作数由 P1 口和 P3 口输入，结果再由 P1 口和 P3 口输出。具体如下：P1 口输入被加数、被减数、被乘数和被除数，输出结果的低 8 位或商；P3 口输入加数、减数、乘数和除数，输出进位(借位)、结果的高 8 位以及余数。键号放在 A 中。

程序为：

```
FOUR:  MOV   P1,#0FFH
       MOV   P3,#0FFH
       MOV   DPTR,#TBJ4
       RL    A              ;键号 * 2
       JMP   @A+DPTR
TBJ4:  AJMP  PRG0
       AJMP  PRG1
       AJMP  PRG2
       AJMP  PRG3
PRG0:  MOV   A,P1
       ADD   A,P3
       MOV   P1,A
       CLR   A
       ADDC  A,#00H          ;进位位放入 A 中
       MOV   P3,A
       RET
PRG1:  MOV   A,P1
       CLR   C
       SUBB  A,P3
       MOV   P1,A
       CLR   A
       RLC   A               ;借位位放入 A 中
       MOV   P3,A
```

```
        RET
PRG2:   MOV     A,P1
        MOV     B,P3
        MUL     AB
        MOV     P1,A
        MOV     P3,B
        RET
PRG3:   MOV     A,P1
        MOV     B,P3
        DIV     AB
        MOV     P1,A
        MOV     P3,B
        RET
```

这个例子中，由于 AJMP rel 只占二个字节，因此，键号先需乘 2，与 DPTR 中的内容形成正确的目的地址。由于 A 中的内容乘 2 后不能大于 255，所以本例中最多可扩展到 128 个分支程序。另外，由于散转过程采用了 AJMP rel 指令，故分支程序必须设置在 2K 范围的地址空间内。对于分支较多(大于 128 个)或分支程序较长的情况，本例就不适用了。下面的例子可以使分支程序的数量大大增加，并且分支程序可以放在 64K 程序存储器的任意位置。

例 6-16 根据(R3R2)的内容转向不同的处理程序。

解 程序为：

```
JMPN:   MOV     DPTR,#TABN
        MOV     A,R3
        MOV     B,#3
        MUL     AB
        ADD     A,DPH
        MOV     DPH,A
        MOV     A,R2
        MOV     B,#3
        MUL     AB
        XCH     A,B
        ADD     A,DPH
        XCH     A,B
        JMP     @A+DPTR
TABN:   LJMP    PRG0
        LJMP    PRG1
          ⋮
        LJMP    PRGN
```

下面的例子是利用查表的方法来实现程序散转的。

例 6-17 根据 R2 的内容转向不同的处理程序。

解 程序为：

```
JMPN:  MOV    A,R2
       MOV    DPTR,#TBJN
       MOVC   A,@A-DPTR
       JMP    @A+DPTR
JBJN:  DB     PRG0
       DB     PRG1
        ⋮
       DB     PRGN
PRG0:  分支程序 0
PRG1:  分支程序 1
        ⋮
PRGN:  分支程序 n
```

这类程序中，地址表和所有的分支程序必须设在 256 个字节中。

例 6-18 根据 R2 的内容转向不同的处理程序。

解 程序为：

```
JMPN:  MOV    DPTR,#TABN
       MOV    A,R2
       MOV    B,#2
       MUL    AB
       ADD    A,DPL
       MOV    DPL,A
       MOV    A,B
       ADDC   A,DPH
       MOV    DPH,A
       CLR    A
       MOVC   A,@A+DPTR
       MOV    R7,A
       CLR    A
       INC    DPTR
       MOVC   A,@A+DPTR
       MOV    DPH,A
       MOV    DPL,R7
       CLR    A
       JMP    @A+DPTR
TABN:  DW     PRG0
```

```
        DW      PRG1
        ⋮
        DW      PRGn
```

该散转程序最多可以实现 256 个分支。

第七节　子程序设计

在程序设计过程中，经常遇到在程序的不同位置需使用完全相同的一段程序的情况。为了避免多次出现同一段程序，节省程序存储器空间，可将这段程序写成独立的程序段，在任何需要的地方即可调用这段程序，运行完毕后再从这段程序返回，程序继续运行，这样的独立程序段就称为子程序，调用子程序的程序称为主程序。子程序的引入大大简化了主程序的结构，增加了程序的可读性，避免了重复性工作，缩短了整个程序。子程序还增加了程序的可移植性，一些常用的运算程序写成子程序形式，可以被随时引用、参考，为广大单片机用户提供了方便。

调用子程序过程由调用指令完成，MCS-51 单片机有二条调用指令：ACALL addr11 和 ACALL addr16。指令中的地址为子程序的入口地址，在汇编语言中通常用标号来代表。在执行这二条指令时，单片机将当前的 PC 值压入堆栈。子程序的最后是返回指令 RET，这条指令将堆栈的内容传入 PC 中，保证程序返回调用的地方继续运行。

在子程序运行时，可能要改变一些寄存器或数据存储器的内容，有时这些内容是主程序正确运行所必不可少的。因此，在子程序调用时，先将这些内容保存起来，子程序返回前再恢复原来的内容，这一过程称为保护现场。保护现场通常由堆栈来完成。需要保护现场时，在子程序开始时安排压栈指令，将需要保护的内容压入堆栈，在子程序的最后，则设置出栈指令，将这些内容弹出堆栈，送回原来的单元。

在子程序调用时，还有一个需要注意的问题，即参数传递问题。在子程序调用时，主程序应先把有关参数放到某些约定的位置，子程序运行时，可以从约定位置得到这些参数。同样，子程序结束前也应把运算结果送到约定位置，返回主程序后，主程序从约定位置上获得这些结果。参数传递可以采用工作寄存器或累加器 A、地址寄存器和堆栈等来完成。

在子程序执行过程中调用其它子程序，这种现象称为子程序嵌套。MCS-51 单片机允许多重嵌套。

子程序的结构与一般程序没有什么差别，最重要的标志是在其最后有一条返回指令。另外，在需要保护现场时，子程序的开始和结束都有入栈和出栈指令。前面的例子中，很多都写成了子程序的形式。这里的例子主要是说明子程序的调用过程和参数传递的方法。

例 6-19　单字节有符号数的加、减法程序。

解　该程序的功能为(R2)±(R3)→R7。R2 和 R3 中为有符号数的原码，R7 中存放计算结果的原码。运算结果溢出时 OV 置位。

程序为：

```
SUB1:   MOV     A,R3
        CPL     ACC.7           ;符号位求反
```

```
        MOV     R3,A
ADD1:   MOV     A,R3
        ACALL   CMPT        ;加数或减数求补码
        MOV     R3,A
        MOV     A,R2
        ACALL   CMPT        ;被加数或被减数求补码
        ADD     A,R3
        JB      OV,OVER
        ACALL   CMPT        ;将结果转换成原码
        MOV     R7,A
OVER:   RET
```

SUB1 为减法程序入口，ADD1 为加法程序入口。CMPT 为单字节有符号数求补码程序，参看例 6-4。本例中，参数传递是通过累加器 A 完成的，主程序将被转换的数送到 A 中，子程序将 A 中的有符号数求补后，存于 A 中。主程序可从 A 中得到运算结果。

例 6-20 4 位 BCD 码的减法程序。

解 BCD 码调整指令不能用于减法指令后面进行 BCD 码调整，也就是说，不能用直接采用 BCD 码调整指令来完成 BCD 码的减法运算。本例是 BCD 码减法运算的一种方法。

设被减数放在 50H 和 51H 单元中，减数放在 60H 和 61H 中，差放入 40H 和 41H 中。

主程序为：

```
        MOV     R1,#50H
        MOV     R0,#60H
        CLR     C
        ACALL   BSUB
        MOV     40H,A
        ACALL   BSUB
        MOV     41H,A
          ⋮
```

子程序为：

```
BSUB:   MOV     A,#9AH
        SUBB    A,@R0
        ADD     A,@R1
        DA      A
        INC     R0
        INC     R1
        CPL     C
        RET
```

在这个例子中，主程序通过地址寄存器 R0 和 R1 将参加运算的 BCD 码的地址传递给子程序，子程序则通过累加器将差传递给主程序。

第八节　浮点数及其程序设计

在单片机应用系统的数据处理过程中，经常会遇到小数的运算问题，如求解 PID 的增量算式、线性化处理等。因此，需要用二进制数来表示小数。表示小数的方法一般有两种，定点数和浮点数。定点数结构简单，与整数的运算过程相同，运算速度快。但随着所表示数的范围的扩大，其位数成倍增加，给运算和存储带来不便，而且也不能保证相对精度不变。浮点数的结构相对复杂，但它能够以固定的字节长度保持相对精度不变，用较少的字节表示很大的数的范围，便于存储和运算，在处理的数据范围较大和要求精度较高时，浮点数被广泛采用。

一、浮点数的概念

我们经常用科学计数法来表示一个十进制数，如

$$1234.75=1.23475E3=1.23475\times10^3$$

在数据很大或很小时，采用科学计数避免了在有效数字前加 0 来确定小数点的位置，突出了数据的有效数字的位数，简化了数据的表示。可以认为，科学计数法就是十进制数的浮点数表示方法。

在二进制数中，也可以用类似的方法来表示一个数，如

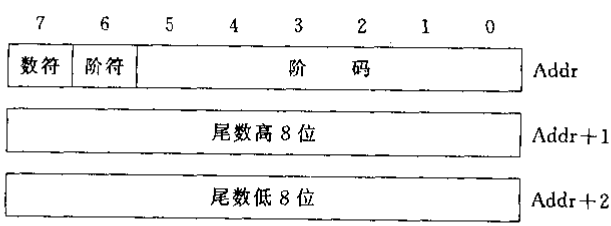
$$1234.75=0.1001\ 10100101\times10^{1011}(\text{二进制})$$
$$=0.1001\ 10100101\times2^{11}$$

一般表达式为

$$N=S\times2^P$$

在这种表示方法中，数据由四个部分组成，即尾数 S 及符号，阶码 P 及符号。

在二进制中，通过定义相应字节或位来表示这四部分，就形成了二进制浮点数。二进制浮点数可以有多种不同的表示方法，下面是一种常见的三字节浮点数的格式：



其中尾数占 16 位，阶码占 6 位，阶符占 1 位，数符占 1 位。阶码通常用补码来表示。

在这种表示方法中，小数点的实际位置要由阶码来确定，而阶码又是可变的，因此称为浮点数。

1234.75 用这种格式的浮点数表示就是

0000 1011 1001 1010 0101 1000

用十六进制表示为

$$1234.75=0B9A58H$$

$$-1234.75=4B9A58H$$

$$0.171875=043B00H$$

$$-0.171875=443B00H$$

三字节浮点数所能表示的最大值为

$$1\times2^{63}=9.22\times10^{18}$$

能表示的最小数的绝对值为

$$0.5\times2^{-63}=5.42\times10^{-20}$$

其所表示的数的绝对值范围$=(5.42\times10^{-20}\sim9.22\times10^{18})$，由此可以看到，三字节浮点数比三字节定点数表示的数的范围大得多。

按同样方法可以定义一个四字节的浮点数，以满足更高精度的需要。

二、规格化浮点数

同一个数用浮点数表示可以是不同的，如

$$1234.75=0B9A58H=0C4D2CH=0D2696H$$

虽然这几种表示其数值是相同的，但其尾数的有效数字的位数不同，分别为16位、15位和14位。在运算过程中，为了最大限度地保持运算精度，应尽量增加尾数的有效位数。这就需要对浮点数进行规格化处理。

在只考虑用二进制原码表示尾数时，尾数的最高位为1，则该浮点数为规格化浮点数。在规格化浮点数中，用尾数为0和最小阶码表示0，三字节规格化浮点数的0表示为410000H。

浮点数在运算之前和运算之后都要进行规格化，规格化过程包括以下步骤：

(1) 首先判断尾是否为0，如果为0，规格化结果为410000H；

(2) 如果尾数不为0，判断尾数的最高位是否为1，如果不为1，尾数左移，阶码减1；

(3) 再判断尾数的最高位是否为1，如果不为1，继续进行规格化操作，如果为1，则规格化结束。

三、浮点数运算

浮点数运算包括加、减、乘、除四则运算，比较运算，开方运算，多项式运算和函数运算。其它运算都可用这些基本运算的组合来完成。本节主要介绍浮点数四则运算及其子程序。

1. 浮点数的加、减运算

浮点数的运算就是求结果的尾数、数符、阶码包括阶符的过程。在加、减运算中，参加运算的浮点数的阶码可能是不同的，其尾数所代表的值也是不同的。在这种情况下，尾数不能直接相加或相减，必须首先使两个数的阶相同，这一过程称为对阶。一般是让小阶向大阶对齐，尾数相应右移。对阶相当于算术中的小数点对齐或代数中的通分。尾数相加或相减得到了结果的尾数，数符由尾数的运算结果的符号确定，阶码就是两个数中较大的阶

码。

例 1 计算 132.25+69.75。

132.25+69.75=088440H+078B80H=088444H+0845C0H=08CA00H=202

由于两个浮点数的阶码分别为 8 和 7，先将加数的阶码变为 8，其尾数右移 1 位。两个数的阶码相同后，尾数直接相加即为和的尾数，和的尾数的最高位为 1，为规格化浮点数。

例 2 计算 12.39−93.1。

12.39−93.1=04C651H−07BA33H=87A169H=−80.71

本例中被减数小于减数，差为负数，结果的数符为 1。差的阶码为两个数中较大的阶码。

2. 浮点数乘法运算

如果设参加运算的两个操作数分别表示为

$$Na=(-1)^{SSa}\times Sa\times 2^{Pa}$$

$$Nb=(-1)^{SSb}\times Sb\times 2^{Pb}$$

它们的积为

$$N=Na\times Nb=(-1)^{SSa+SSb}\times(Sa\times Sb)\times 2^{Pa+Pb}$$

式中 SSa 和 SSb 为两个数的数符。

乘法运算可总结为：

(1) 积的数符为乘数和被乘数的符号位按模 2 求和的结果，即符号位的异或为结果的符号位；

(2) 积的阶为乘数和被乘数的阶的和；

(3) 积的尾数为被乘数和乘数的尾数的积；

参加运算的浮点数一般都是规格化的浮点数，尾数的积小于 1，不需进行右规格化处理。但有可能小于 0.5，所以需进行左规格化处理，使积为规格化浮点数。如果乘数或被乘数的尾为 0，则积为 410000H。由于在尾数相乘时，积的低 16 位不能反映在结果中，因此，积可能会产生一定的误差。

例 3 计算 22.41×4.23。

22.41×4.23=05B349H×03875EH=07BD9AH=94.8

积的阶为乘数和被乘数的和，即 8，尾数相乘时，积小于 0.5，进行左规格化处理，阶码变为 7。

例 4 计算 2586.5×(−6.91)。

2586.5×(−6.91)=0CA1B0H×83DD13H=8F8BA0H=−17872

被乘数为正数，数符为 0，乘数为负数，数符为 1，积的数符为 0+1=1，所以积为负数。

3. 浮点数的除法运算

除法运算可以表示为

$$N=Na/Nb=[(-1)^{SSa}\times Sa\times 2^{Pa}]/[(-1)^{SSb}\times Sb\times 2^{Pb}]$$

$$=(-1)^{SSa-SSb}\times(Sa/Sb)\times 2^{Pa-Pb}$$

浮点数的除法运算可以总结为：

(1) 商的数符为被除数与除数的符号位的差；

(2) 商的阶码为被除数和除数的阶码的差；

(3) 商的尾数为被除数和除数的尾数的商。

规格化的浮点数进行除法运算时，尾数相除，商不会小于 0.5，不需进行左规格化处理。但有可能大于 1，有时需进右规格化处理。

例 5 计算 390.67÷14.31。

390.67÷14.31＝09C357H÷04E511H＝05DA4EH＝27.3

商的阶码为被除数与除数的阶码的差。尾数相除时，结果的最高位为 1，商为规格化浮点数。

例 6 计算－6.02÷16.157。

－6.02÷16.157＝83C0AAH÷058143H＝FFBEC8H＝－0.373

异号相除时，商为负数。由于被除数的尾数大于除数的尾数，所以，被除数先进行右规格化，阶码变为 4，商的阶码为－1，用补码来表示。

四、浮点数运算子程序

通过前面的分析我们看到，浮点运算比较复杂，有其特有的方法和规格规律。这里介绍几种常用的三字节浮点数运算子程序，通过分析、设计这些程序，可以进一步了解浮点数的运算过程和特点，熟悉复杂程序的设计方法。

1. 浮点数通用规格化子程序

在浮点数运算过程中，有时需要左规格化，有时需要右规格化。通用规格化子程序既可实现左规格化，又可实现右规格化，其具体功能如下：

当 C_Y＝0 时，进行右规格化：F0＝0 时，对 R6(阶)R2R3(尾数)右规格化 1 位；F0＝1 时，对 R7(阶)R4R5(尾数)右规格化 1 位。

当 C_Y＝1 时，对 R6(阶)R2R3(尾数)执行左规格化。

程序开始时，判断是执行左规格化还是右规格化。如果是右规格化，还要判断是对 R6(阶)R2R3(尾数)还是对 R7(阶)R4R5(尾数)进行规格化。如果是左规格化，直至将操作数变为规格化浮点数。其程序框图如图 6-13 所示。

程序为：

```
FSDT:   JC   LNORMS
        MOV  C,39H          ;进行右规格化
        JB   F0,NR7
        MOV  A,R2           ;R2R3 右移 1 位
        RRC  A              ;(CY)移入尾数最高位
        MOV  R2,A
        MOV  A,R3
        RRC  A
        MOV  R3,A
        INC  R6             ;阶码加 1
        RET
```

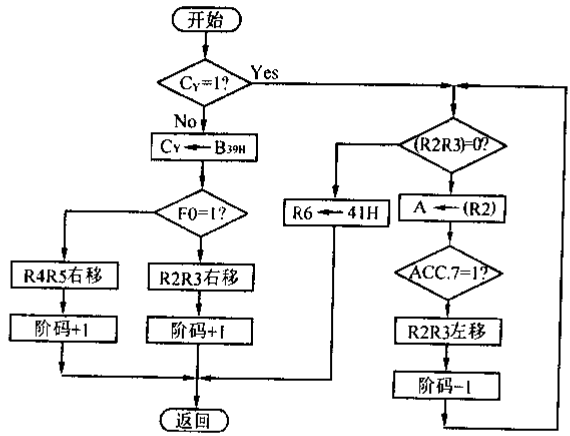


图 6-13　通用规格化子程序框图

```
NR7:    MOV   A,R4
        RRC   A
        MOV   R4,A
        MOV   A,R5
        RRC   A
        MOV   R5,A
        INC   R7
        RET
LNORMS:MOV   A,R7
        JNZ   LSHIFT
        CJNE  R3,#00H,LSBIT8  ;尾数为 0,阶码为 41H
        MOV   R6,#41H
LSEND:  RET
LSHIFT: JB    ACC.7,LSEND
LSBIT8: MOV   C,F0
        MOV   A,R3
        RLC   A
        MOV   R3,A
        MOV   A,R2
        RLC   A
        MOV   R2,A
        CLR   F0
```

```
        DEC    R6
        SJMP   LNORMS
```

2. 浮点数加减运算子程序

参加运算的浮点数可能是正数，也可能是负数。对于加法运算，当加数和被加数的数符相同时，尾数相加，不同时尾数相减；对于减法运算，当减数和被减数的数符相同时，尾数相减，不同时尾数相加。当两个浮点数的阶码不同时，要进行对阶，使小阶与大阶相等，因此，结果的阶码与其较大的阶码相同。

在执行加法运算时，尾数有可能大于 1，因此要进行右规格化处理；执行减法运算时，尾数有可能小于 0.5，因此，要进行左规格化处理。

下面是三字节浮点数加、减法处理子程序，具体功能为：

R6(阶)R2R3(尾)±R7(阶)R4R5(尾)——→R4(阶)R2R3(尾)；

当位 3AH＝0 时，执行加法；

当位 3AH＝1 时，执行减法。

程序框图如图 6-14 所示。

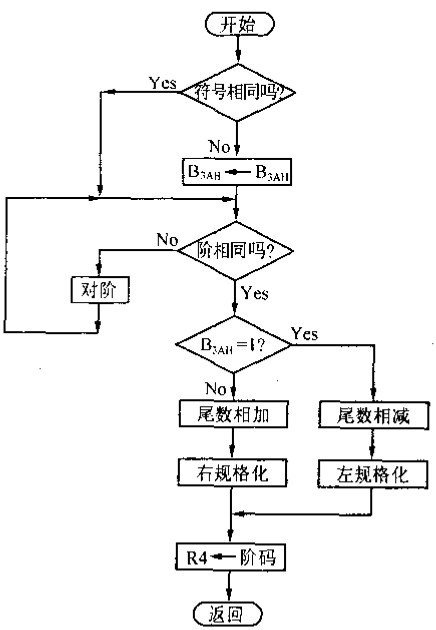


图 6-14　浮点数加、减操作程序框图

程序如下：

```
FABP:  MOV    A,R6
```

```
        MOV   C,ACC.7
        MOV   38H,C              ;存被加数数符
        XRL   A,R7
        JNB   ACC.7,FAB1         ;数符相同则转
        CPL   3AH                ;数符不等,求反运算标志
FAB1:   MOV   A,R6
        MOV   C,ACC.6            ;扩展阶码符号位
        MOV   ACC.7,C
        MOV   R6,A
        MOV   A,R7
        MOV   C,ACC.6
        MOV   ACC.7,C
        MOV   R7,A
        CLR   C
        MOV   A,R6
        SUBB  A,R7
        JZ    FAB2               ;阶码相同则转
        CLR   F0
        CLR   39H
        JB    ACC.7,FAB6
        CJNE  R4,#00H,FAB7
        CJNE  R5,#00H,FAB7
FAB2:   JB    3AH,FAB9           ;转向执行尾数减法
        MOV   A,R3               ;执行尾数加法
        ADD   A,R5
        MOV   R3,A
        ADD   A,R2
        ADDC  A,R4
        MOV   R2,A
        JNC   FAB4
        SETB  39H
        CLR   C
FAB3:   CLR   F0
        LCALL FSDT
FAB4:   CJNE  R2,#00H,FAB5
        CJNE  R3,#00H,FAB5
        MOV   R4,#41H            ;结果为0,规格化
        RET
```

```
FAB5:  MOV   A,R6
       MOV   C,38H
       MOV   ACC.7,C
       XCH   A,R4
       MOV   R6,A
       RET
FAB6:  CJNE  R2,#00H,FAB8
       CJNE  R3,#00H,FAB8
       MOV   A,R7
       MOV   R6,A
       SJMP  FAB2
FAB7:  CPL   F0
FAB8:  CLR   C
       LCALL FSDT
       SJMP  FAB1
FAB9:  MOV   A,R3                ;尾数相减
       CLR   C
       SUBB  A,R5
       MOV   R3,A
       MOV   A,R2
       SUBB  A,R4
       MOV   R2,A
       JNC   FAB10
       CLR   A
       CLR   C
       SUBB  A,R3
       MOV   R3,A
       CLR   A
       SUBB  A,R2
       MOV   R2,A
       CPL   38H
FAB10: SETB  C
       SJMP  FAB3
```

3. 浮点数乘法运算子程序

浮点数相乘时，阶码直接相加即获得积的阶码，尾数相乘时，结果可能小于 0.5，需进行左规格化处理。下面是三字节浮点数乘法运算子程序，具体功能为：

(R0)指向的三字节浮点数×(R1)指向的三字节浮点数——→R4(阶)R2R3(尾数)。

图 6-15 为三字节浮点数乘法的程序框图。

程序为：

```
FMUL:LCALL FMLD      ;传送浮点数
     MOV   A,R6
     XRL   A,R7      ;求积的数符
     MOV   C,ACC.7
     MOV   38H,C
     LCALL DMUL      ;调用双字节无符号数
                      乘法子程序
     MOV   A,R7
     MOV   C,ACC.7
     MOV   F0,C
     MOV   A,@R0
     ADD   A,@R1
     MOV   R6,A
     SETB  C
     LCALL FSDT      ;进行规格化操作
     MOV   A,R6
     MOV   C,38H
     MOV   ACC.7,C   ;置积的数符
     MOV   R4,A
     RET
```

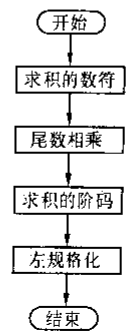


图 6-15 三字节浮点数乘法子程序框图

注：(1)FMLD 为浮点数取数子程序，功能为：将(R0)指向的三字节浮点数送入 R6(阶)R2R3(尾数)中，将(R1)指向的三字节浮点数送入 R7(阶)R4R5(尾数)中。

(2)DMUL 为双字节无符号数乘法子程序。

4. 浮点数除法运算子程序

在进行除法运算时，被除数的尾数可能比除数的尾数大很多，使结果大于 1。为避免这种情况，如果被除数尾数大于除数的尾数，先将被除数的尾数右移，使其小于除数的尾数，阶码也相应增加，保持其数值不变。下面是三字节浮点数除法运算程序，其功能为：

(R0)指向的三字节浮点数除以(R1)指向的三字节浮点数——→

R4(阶)R2R3(尾数)中。

程序框图如图 6-16 所示。程序为：

```
FDIV:  LCALL FMLD
       MOV   A,R6
       XRL   A,R7          ;求商的数符
       MOV   C,ACC.7
       MOV   38H,C
       CLR   A
```

```
        MOV   R6,A
        MOV   R7,A
        CJNE  R4,#00H,FDIV1
        CJNE  R5,#00H,FDIV1
        SETB  C
        RET                    ;除数为 0,返回
FDIV1:  MOV   A,R3             ;比较被除数与除数的尾数
        SUBB  A,R5
        MOV   A,R2
        SUBB  A,R4
        JC    FDIV2
        CLR   F0
        CLR   39H
        LCALL FSDT
        MOV   A,R7
        RRC   A
        MOV   R7,A
        CLR   C
        SJMP  FDIV1
FDIV2:  CLR   A
        XCH   A,R6
        PUSH  ACC
        LCALL DDIV             ;调用双字节除法子程序
        POP   ACC
        ADD   A,@R0
        CLR   C
        SUBB  A,@R1
        MOV   C,38H
        MOV   ACC.7,C
        MOV   R4,A
        CLR   C
        RET
```

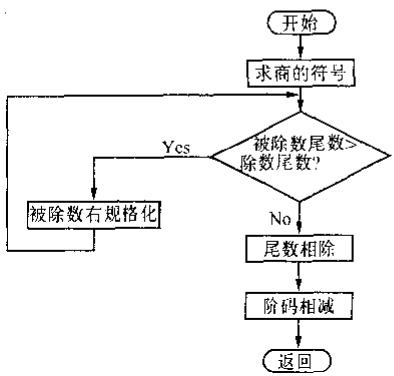


图 6-16　浮点数除法程序框图

注:DDIV 为双字节无符号数除法子程序。

思考题与习题

1. 试编写双字节无符号数的加法程序。具体功能为 R2R3+R6R7→R4R5。
2. 试编写双字节无符号数的减法程序。具体功能为 R2R3−R6R7→R4R5。
3. 试编写将 16 位二进制数右移 1 位的程序。

4. 试编写将 16 位二进制数左移 1 位的程序。

5. 试编写将 2 位 BCD 码转换为二进制数的程序。

6. 已知 A 中为十六进制数的 ASCII 码，试编写一段程序将其转换为对应的十六进制数。

7. 已知 A 中为 1 位十六进制数，试编写将其转换为其对应的 ASCII 码的程序。

8. 试编写求符号函数

$$f(x)=\begin{cases}1 & (x>0)\\ -1 & (x\leqslant 0)\end{cases}$$

的程序。

9. 编写求下列函数值的程序。

(1) $y=\begin{cases}a\times b & (a\leqslant b)\\ a\div b & (a>b)\end{cases}$（设 a 和 b 放在 30H 和 31H 中）

(2) $y=\begin{cases}1 & |x|\leqslant 2\\ |x| & |x|>2\end{cases}$（设 x 放在 R7 中）

10. 试编写将以 R0 中的内容为首地址的内部 RAM 中 n 个连续单元清零的程序。

11. 试编写一段程序，将内部数据存储器以 30H 开始的 16 个数搬到以 40H 开始的 16 个单元中。

12. 试编写统计数据区长度的程序，该数据区以 0 结束。

13. 设在数据存储器中连续存放着 n 个数，试编写求这 n 个数和的程序。

14. 试编写多字节加法程序。

15. 试编写多字节 BCD 码加法程序。

16. 试编写多字节减法程序。

17. 试编写一无符号数乘法程序，功能为

(1)R2R3 * R7→R4R5R6

(2)R2R2 * R6R7→R4R5R0R6

18. 试编写一无符号数除法程序，功能为

(1)R2R3÷R7　商→R6R7，余数→R5

(2)R2R3÷R6R7　商→R6R7，余数→R5

19. 试编写将内部数据存储器中 60H 开始的 32 个数倒序排列起来的程序。

20. 试编一段程序，找出一连续数据区中最大的数，并放入 R2 中。

21. 试编写一段程序，将一连续数据区中的数据由小到大排列起来。

22. 试编写一段程序，将若干个十六进制数转换为 ASCII 码。

23. 试编写延时 2ms 的程序。

24. 用查表程序求 0～8 之间整数的立方。

25. 编写有 6 个命令键的散转程序。

(1)键号为 0、1、2、3、4、5；

(2)6 个键 A、B、C、D、1、2 的 ASCII 码放在累加器 A 中。

26. 试编写双字节有符号原码的加、减法程序。

27. 试编写一段程序，将 R2 中的 2 位十六进制数转换为 ASCII 码。

28. 试用子程序求下列多项式。

(1) $y = a^2 + b^2 + c^2$

(2) $y = (a + b)^2 + (b + c)^2 + (c + a)^2$

(a、b、c 均为二进制无符号单字节数)

29. 试编写一段程序，将内部数据存储器中连续存放的若干单字节十六进制数转换为 ASCII 码，放入外部数据存储器的连续单元中。

30. 浮点数由哪几部分组成？有什么特点？

31. 下面浮点数中，哪些是正数？哪些是负数？哪些绝对值大于 1？哪些绝对值小于 1？

07E033H　3088EAH　92B3D0H　E3C131H

B08D59H　FEA000H　43FA57H　410000H

32. 浮点数能否精确地表示任意数(在其所能表示的范围)？

33. 求下列浮点数的十进制表示。

05DE11H　E38143H　83D14AH

34. 什么是规格化浮点数？为什么要进行规格化？

35. 已知 a、b、c、x 为三字节浮点数，试编程，求 $N=1+bx+cx^2$。

(用本节介绍的子程序，a、b、c、x 存放于内部数据存储中)

第七章　MCS-51 单片机的扩展应用

前面已经介绍了 MCS-51 单片机基本原理及程序设计，这一章主要介绍 MCS-51 存储器的扩展方法及中断、定时器/计数器、串行口的原理及其应用。通过本章的讲述，读者将进一步掌握 MCS-51 单片机的工作原理，并加深其使用方法的了解。

第一节　程序存储器的扩展

MCS-51 单片机程序存储器的寻址空间为 64KB，对于 8051/8751 片内程序存储器为 4KB 的 ROM 或 EPROM，在单片机的应用系统中，片内的存储容量往往不够，特别是 8031，片内没有程序存储器，必须外扩程序存储器。MCS-51 外扩程序存储器结构图如图 7-1 所示。

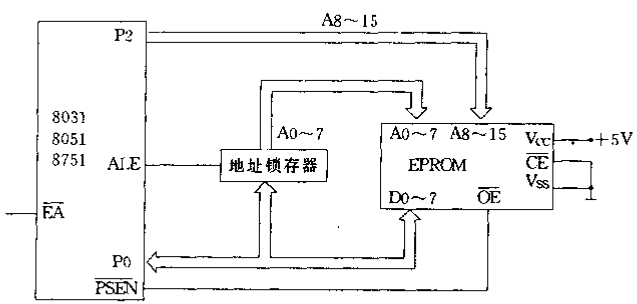


图7-1 MCS-51外扩程序存储器结构图

对于 8051/8751，由于内部有 4KB 的 ROM/EPROM，在外扩程序存储器时，一般情况下$\overline{EA}$接高电平。此时 8051/8751 内部 4KB ROM/EPROM 程序存储器地址为 0000H～0FFFH，外部程序存储器的地址为 1000H～FFFFH。计数器 PC 值在 0000H～0FFFH 时，指向片内程序存储器；当 PC 值大于 0FFFH 时，则指向片外程序存储器。当$\overline{EA}$接低电平时，8051/8751 内部程序存储器无效，系统只有外部程序存储器，地址从 0000H 开始，此时 8051/8751 相当于 8031。

MCS-51 单片机在访问外部程序存储器时，由 P2 口送出地址的高 8 位，P2 口有输出锁存功能，可直接接至外部存储器的地址端，无需再加地址锁存器。P0 口则作为分时数据/地址双向总线，分别用于输出地址的低 8 位和输入指令。在这种情况下，每一个机器周期中，允许地址锁存信号 ALE 两次有效，在 ALE 由高变低时，有效地址高 8 位出现在 P2

口，有效地址低 8 位出现在 P0 口上，低 8 位地址应通过地址锁存器把地址锁存起来。同时，$\overline{PSEN}$也是每个机器周期两次有效，用于选通外部程序存储器，使指令送到 P0 总线上，由 CPU 取入。可见，在外接程序存储器时，P0 口既作为地址低 8 位输出口，又作为指令的输入口，当 ALE 有效时，P0 口上的数据为有效地址，当$\overline{PSEN}$有效时，P0 口上的数据为指令代码。其时序图如图 7-2 所示。

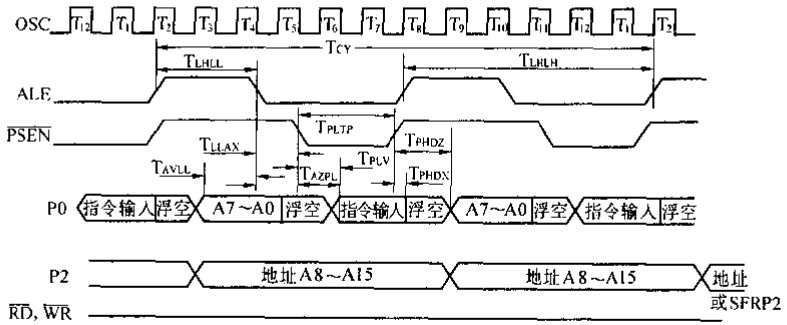


图 7-2　程序存储器的读周期时序

一、地址锁存器 8282 或 74LS373

地址锁存器通常使用 Intel 公司的 8282 或 TTL 芯片 74LS373。它们都是双列直插 20 引脚的塑封芯片，其引脚图如图 7-3 所示。

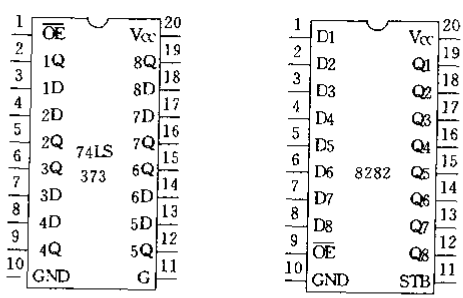


图7-3 74LS373 Intel8282引脚图

8282 和 74LS373 都是透明的带有三态门的 8 口锁存器。图 7-4 所示为 74LS373 结构图，共有 8 个输入端 D1～D8 及 8 个输出端 Q1～Q8。

74LS373 的 G 端(或 8282 的 STB 端)为输入选通端，当 G＝1 时，锁存器处于透明工作状态，即锁存器的输出状态随数据端的变化而变化，即 Qi＝Di(i＝1,2,…,8)。当 G 端由 1 变 0 时，数据被锁存起来，此时输出端 Qi 不再随输入端的变化而变化，而一直保持锁存前的值不变。G 端(或 STB 端)可直接与单片机的锁存控制信号端 ALE 相连，在 ALE

的下降沿进行地址锁存。

二、常用 EPROM 程序存储器芯片介绍

MCS-51 单片机应用系统中使用得最多的 EPROM 程序存储器是 Intel 公司的典型系列芯片 2716(2K×8)、2732A(4K×8)、2764(8K×8)、27128(16K×8)、27256(32K×8)和 27512(64K×8)。

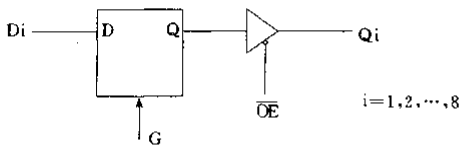
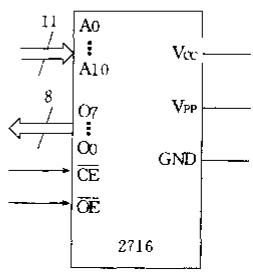


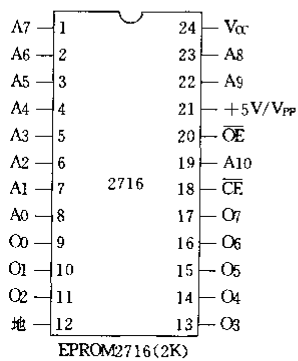
图7-4 74LS373 8282原理结构

1. 2716 及 2732 引脚及功能

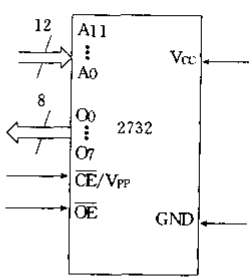
2716 及 2732 均为双列直插 24 引脚芯片,其引脚图如图 7-5 所



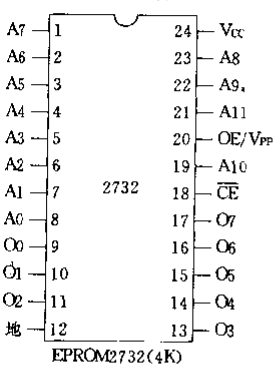
(a)2716逻辑图



(b)2716引脚图



(c)2732逻辑图



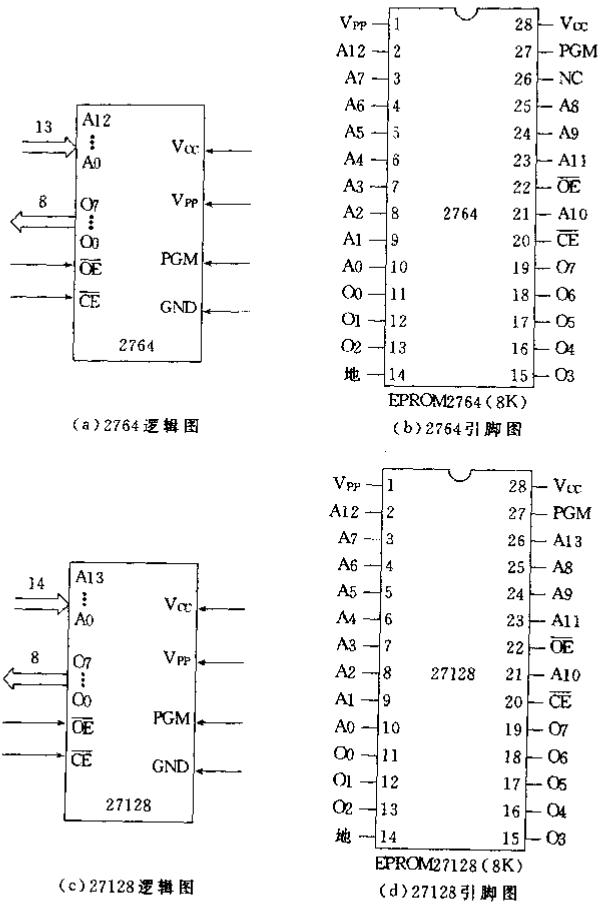
(d)2732引脚图

图7-5 2716、2732引脚与逻辑图

示。图中，A0～A10(2732 为 A0～A11)为地址线，O0～O7 为数据线；$\overline{CE}$为片选信号，低电平有效，$\overline{OE}$为允许数据输出选通线(低电平有效)，Vcc 为主电源(+5V)，Vpp 为编程电压(通常为+25V)。2716 及 2732 读出最大时间为 450ns，具有读、编程、校验等等工作方式。2716 与 2732 具有兼容性，将 2732A 插入 2716 电路中也可正常工作，但只能当作 2716，即只有 2KB 有效。

2. 2764、27128 引脚及功能

2764 及 27128 均为双列直插 28 引脚的芯片，其引脚图如图 7-6 所示。



(a)2764 逻辑图　(b)2764 引脚图

(c)27128 逻辑图　(d)27128 引脚图

图 7-6　2764、27128 引脚与逻辑图

图中 A0～A12(27128 为 A0～A13)为地址线；O0～O7 为 8 位数据线；$\overline{CE}$为选片信号线，低电平有效；$\overline{OE}$为允许数据输出选通信号线，低电平有效；PGM 为编程脉冲输入线；Vcc 为主电源线(+5V)；Vpp 为编程电源线(典型值有 21V，12.5V)。2764 和 27128 的读出时间为 250ns，同样具有读出、编程、校验等方式。2764 与 27128 也具有相互兼容性。

3. 27256 引脚及功能

27256 为 32K×8 的双列直插 28 引脚芯片，引脚及逻辑图如图 7-7 所示。其中 A0～A14 为 15 条地址线；O0～O7 为 8 位数据线；$\overline{CE}$为低电平有效的片选信号；$\overline{OE}$为低电平有效的数据输出允许信号，Vcc 为主电源(+5V)；V_{PP}为编程电压(典型值 12.5V)。

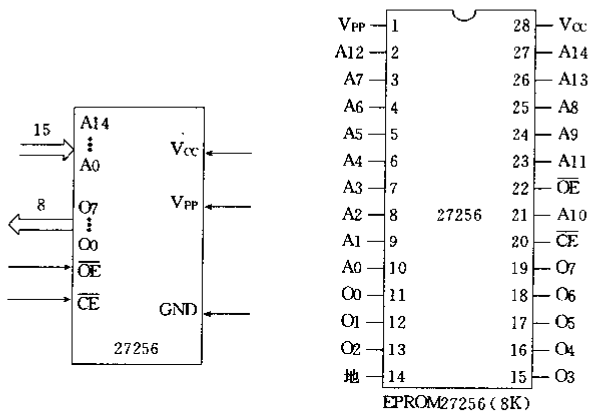


图 7-7 27256 引脚与逻辑图

4. 实际扩展 EPROM 程序存储器的注意事项

(1)根据应用系统容量要求选择 EPROM 芯片时，应使应用系统电路尽量简化，在满足容量要求时尽可能选择大容量芯片，以减少芯片组合数量。目前大容量芯片价格日趋便宜，故采用较大容量芯片从长远的经济效益看也有好处。

(2)选择好 EPROM 容量后，要选择好能满足应用系统应用环境要求的芯片型号。例如在确定选择 8K EPROM 芯片后，根据不同的应用参数在 2764 中选择相应的型号规格芯片。这些应用参数主要有最大读取时间、电源容差、工作温度以及老化时间等。如果所选择的型号不能满足使用环境要求，会造成工作不可靠，甚至不能工作。

(3)选用的锁存器不同，电路连接可能不同。目前使用最多的几种锁存器管脚不一定兼容。

(4)Intel 公司的通用 EPROM 芯片管脚有一定的兼容性，在电路设计时应充分考虑其兼容特点。例如，为了保证 2764、27128、27256 在电路中的兼容，可将第 26、27 管脚的印刷电路连线做成易于改接的形式。

三、几种典型 EPROM 扩展电路

1. EPROM2764 扩展电路

图 7-8 所示为 8031 经过地址锁存器 74LS373 与 2764 的连接图，用于扩展产生 8K 字节的外部程序存储器。8031 的 P0 口作为地址/数据输入/输出接口，既与 74LS373 地址输入端连接，又与 2764 的数据输出端连接。2764 的高位地址由 P2 口的低位来提供，由于 2764 共有 13 条地址线，P2 口其使用 P2.0～P2.4 共 5 条地址线，分别与 2764 的 A8～

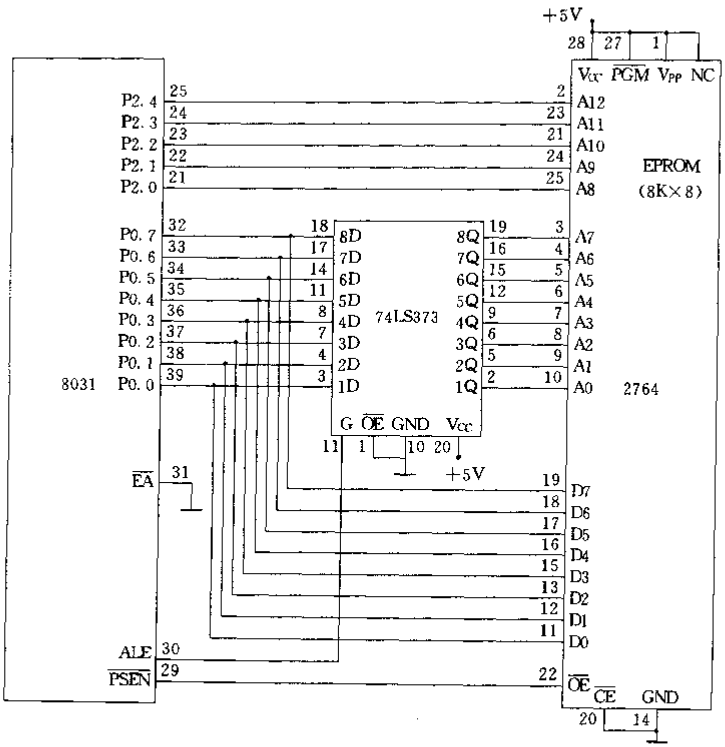


图7-8 8031与2764的连接图

A12 相连。8031 的$\overline{PSEN}$与 2764 的$\overline{OE}$相连，用于从 EPROM 读入数据控制。8031 的 ALE 与 74LS373 的 G 端相连，用于锁存 P0 口送出的低 8 位地址。

2. EPROM27256 扩展电路

图 7-9 所示为 8031 经过地址译码器 8282 与 27256 的连接图，用于扩展产生 32K 字节的外部程序存储器。8031 的 P0 口作为数据口，与 27256 数据输出端连接，同时 P0 口与地址锁存器 8282 输入端连接，8282 输出端与 27256 低 8 位地址线相连，用于提供

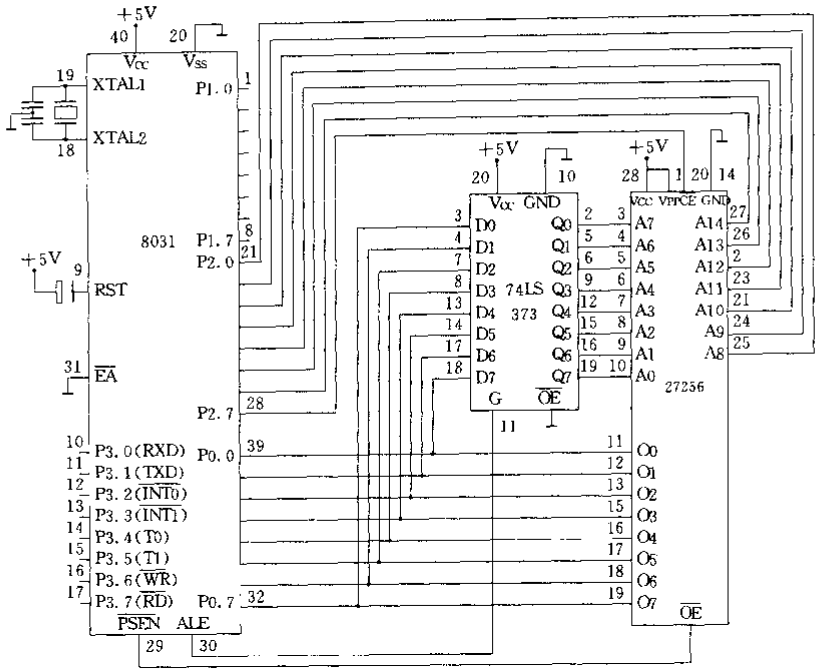


图7-9　8031与27256的连接图

EPROM的低8位地址；8031的P2口(共用7条引线P2.0～P2.6)与27256的A0～A14相连。

3. 使用多片EPROM的扩展电路

在扩展多片EPROM时，所有EPROM芯片的选片端$\overline{CE}$都必须按照地址线进行选择，以使不同的EPROM芯片具有不同的地址区间。片选线使用8031剩余的地址线进行选片或地址译码选择。例如扩展3片2764EPROM时，每片地址有8K，每片地址线为A0～A12，剩余3条地址线。采用线选法就可用这3条地址线分别选通3片EPROM2764的$\overline{CE}$端，如图7-10所示。显然采用线选法扩展2764EPROM，只能扩展3片。经扩展后3片EPROM的地址范围分别为：

EPROM　#1　0000H～1FFFH

EPROM　#2　4000H～5FFFH

EPROM　#3　8000H～9FFFH

由上可见，采用线选办法扩展多片EPROM时，一是扩展容量有限(如扩展2764最多3片)，二是地址往往不能连续，使用起来不方便。为解决这一问题，可以采用译码选择的方法。常用的译码器有74LS138(3线8线译码器)，其引脚及功能表如图7-11(a)所示(见163页)。采用译码方式扩展3片2764如图7-11(b)所示。P2.5～P2.7接74LS138的

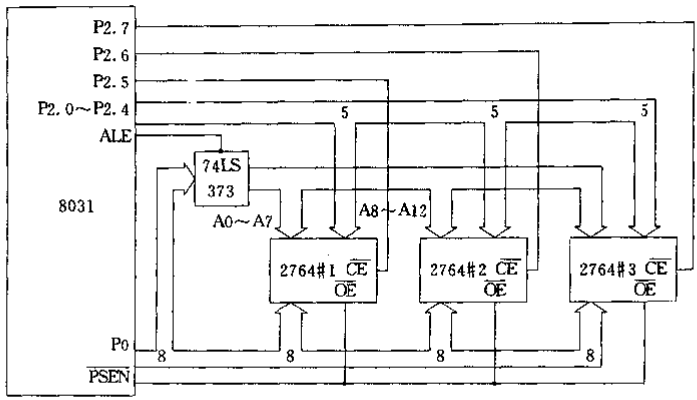


图 7-10 线选 EPROM 扩展图

输入端 A、B、C，输出 $\overline{Y}_0$、$\overline{Y}_1$、$\overline{Y}_2$ 分别接于 3 个 2764 的$\overline{CE}$端。这样 3 个 EPROM 芯片工作地址范围为：

EPROM #1 0000H～1FFFH

EPROM #2 2000H～3FFFH

EPROM #3 4000H～5FFFH

显然，采用译码方式扩展 2764，最多可以扩展 8 片 2764。

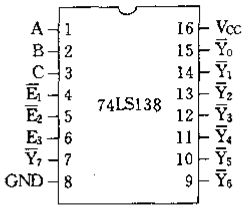
第二节 外部数据存储器的扩展

MCS-51 芯片内部具有 128 个字节 RAM 存储器，它们可以作为寄存器、堆栈、数据缓冲器。CPU 对其内部 RAM 有丰富的操作指令，因此这个 RAM 是十分珍贵的资源。在许多系统中，仅仅片内的 RAM 存储器往往不够，在这种情况下，可以扩展外部数据存储器。在扩展时，外接电路除了随机存储器 RAM 之外，还有地址锁存器、地址译码器等电路，外接最大容量不超过 64K 字节。

图 7-12 给出了单片机扩展 RAM 的电路结构。图中 P0 口分时传送 RAM 的低 8 位地址和数据，P2 口为高 8 位地址线，用于对 RAM 进行页寻址。在外部 RAM 读/写周期，CPU 产生$\overline{RD}$/$\overline{WR}$信号。

MCS-51 单片机与外部 RAM 单元之间数据传送的时序波形如图 7-13 所示。

在图 7-13(a)的外部数据存储器读周期中，P2 口输出外部 RAM 单元的高 8 位地址，P0 口分时传送低 8 位地址及数据。当地址锁存允许信号 ALE 为高电平时，P0 口输出的地址信息有效，ALE 的下降沿将此地址打入外部地址锁存器，接着 P0 口变为输入方式，读信号$\overline{RD}$有效，选通外部 RAM，相应存储单元的内容出现在 P0 口，由 CPU 读入累加器。



74LS138真值表

输入						输出							
使能			选择										
E_3	$\overline{E_2}$	$\overline{E_1}$	C	B	A	$\overline{Y_0}$	$\overline{Y_1}$	$\overline{Y_2}$	$\overline{Y_3}$	$\overline{Y_4}$	$\overline{Y_5}$	$\overline{Y_6}$	$\overline{Y_7}$
1	0	0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	0	0	1	1	0	1	1	1	1	1	1
1	0	0	0	1	0	1	1	0	1	1	1	1	1
1	0	0	0	1	1	1	1	1	0	1	1	1	1
1	0	0	1	0	0	1	1	1	1	0	1	1	1
1	0	0	1	0	1	1	1	1	1	1	0	1	1
1	0	0	1	1	0	1	1	1	1	1	1	0	1
1	0	0	1	1	1	1	1	1	1	1	1	1	0
0	×	×	×	×	×	1	1	1	1	1	1	1	1
×	1	×	×	×	×	1	1	1	1	1	1	1	1
×	×	1	×	×	×	1	1	1	1	1	1	1	1

图7-11 (a)74LS138 引脚图及功能表

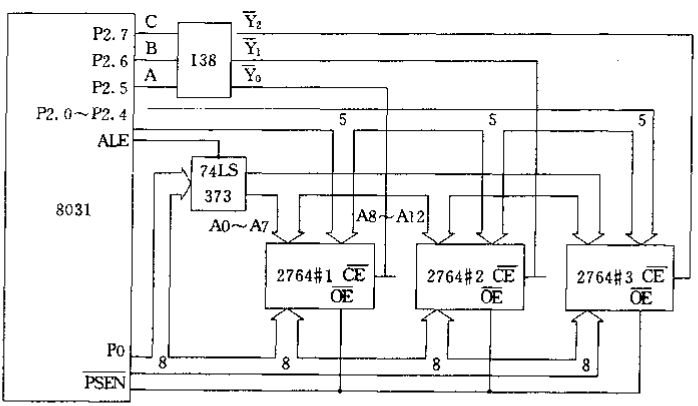


图7-11 (b)译码方式EPROM扩展图

外部数据存储器写周期波形，如图 7-13(b)所示，其操作过程与读周期类似。写操作时，在 ALE 下降为低电平以后，$\overline{WR}$信号才有效，P0 口上出现的数据写入相应的 RAM 单元。

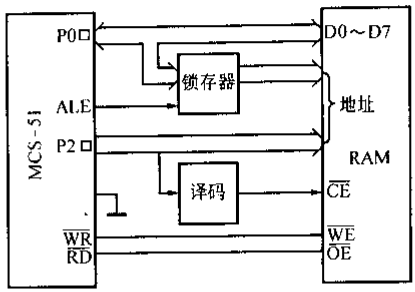
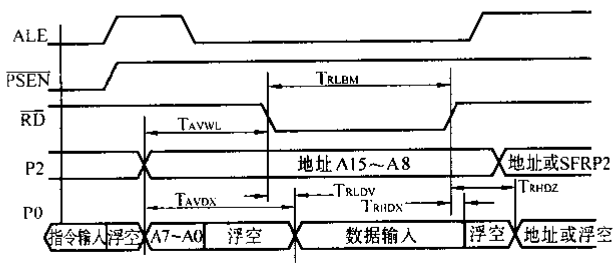
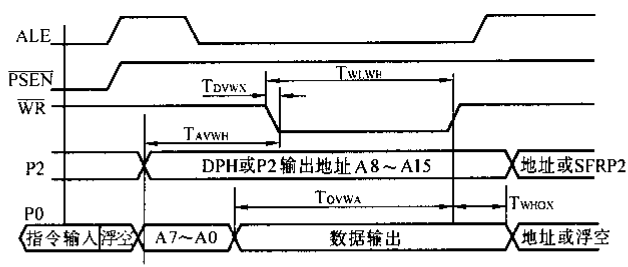


图 7-12　外部数据存储器扩展



(a) 数据存储器读周期



(b) 数据存储器写周期

图 7-13　访问外部 RAM 的时序波形

一、常用 RAM 芯片

在 8031 应用系统中，最常用的静态随机存取存储器 RAM 电路有 6116(2K×8)和 6264(8K×8)。

1. 6116 引脚及功能

6116 是一种 16384 位(2K×8)的静态随机存储电路，24 线的双列直插式器件。逻辑符号及引脚图如图 7-14 所示。A0～A10 为 11 位地址线；O0～O7 为 8 位数据线；$\overline{CE}$片选

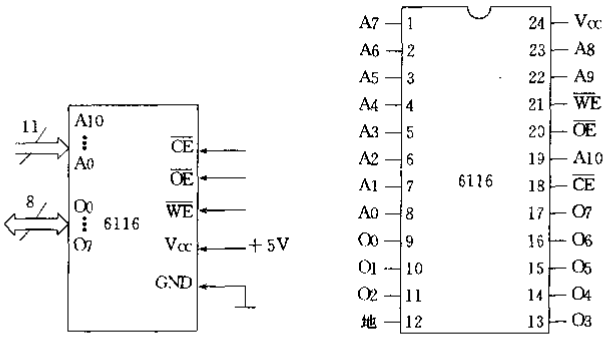


图7-14 6116逻辑及引脚图

信号线；$\overline{OE}$、$\overline{WE}$为读/写信号线。6116的操作方式控制如表7-1所列。

表7-1 6116操作方式选择

$\overline{CE}$	$\overline{WE}$	$\overline{OE}$	方式	功能
0	0	1	写	O0～O7上内容写入A0～A10对应单元
0	1	0	读	A0～A10对应单元内容输出到O0～O7
1	×	×	非选	O0～O7呈高阻抗

2. 6264引脚及功能

6264是一种65536位(8K×8)的存储器电路，28引脚。其逻辑符号及引脚如图7-15所示。A0～A12为13位地址线，输入地址和内部字节的单元对应。O0～O7为8位数据线；$\overline{CE}$

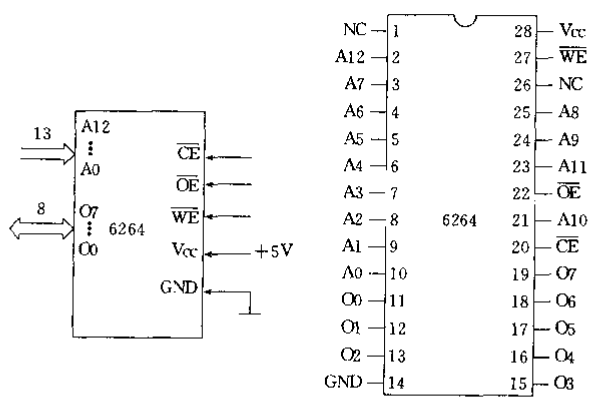


图7-15 6264逻辑及引脚图

为选片信号线；$\overline{OE}$、$\overline{WE}$为读/写信号线，都是低电平有效。6264的操作方式控制如表7-2所列。

表 7-2　6264 操作方式选择

$\overline{CE}$	$\overline{WE}$	$\overline{OE}$	方　式	说　明
0	0	1	写	O0～O7 上信息写入 A0～A12 上地址对应单元
0	1	0	读	A0～A12 上地址对应单元内容输出到 O0～O7
1	×	×	非选	O0～O7 呈高阻抗

二、典型外部数据存储器的扩展方法

随着 RAM 芯片的发展，目前作为 MCS-51 外扩数据存储器的典型芯片采用容量为 8K×8 的 6264。其连接图如图 7-16 所示。

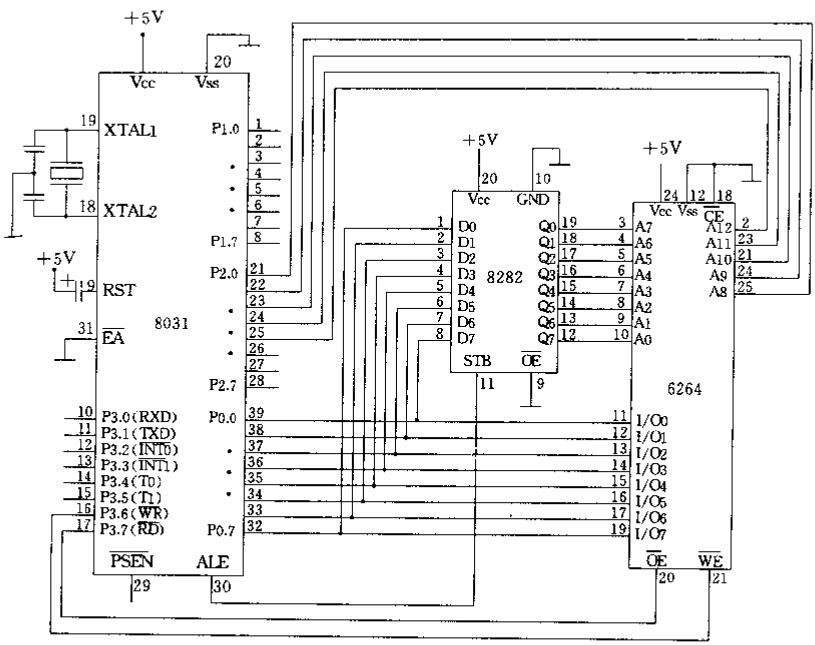


图7-16　8031外部8K ROM连接图

8031 的 P0 口一方面与 6264 的 8 条 I/O 数据线相连，同时经锁存器 74LS373 与 6264 的低 8 位地址线相连。P2.0～P2.4 与 6264 的高 5 位地址线相连。6264 的写使能位与单片机的$\overline{WR}$端相连；输出使能位$\overline{OE}$与单片机的$\overline{RD}$端相连；$\overline{CS}$片选信号在只有一片 6264 的情况下，可接低电平。

对于图 7-16 所示线路，6264 的地址范围为 0000H～3FFFH。访问此外部 RAM，可用 MOVX　@DPTR 指令。

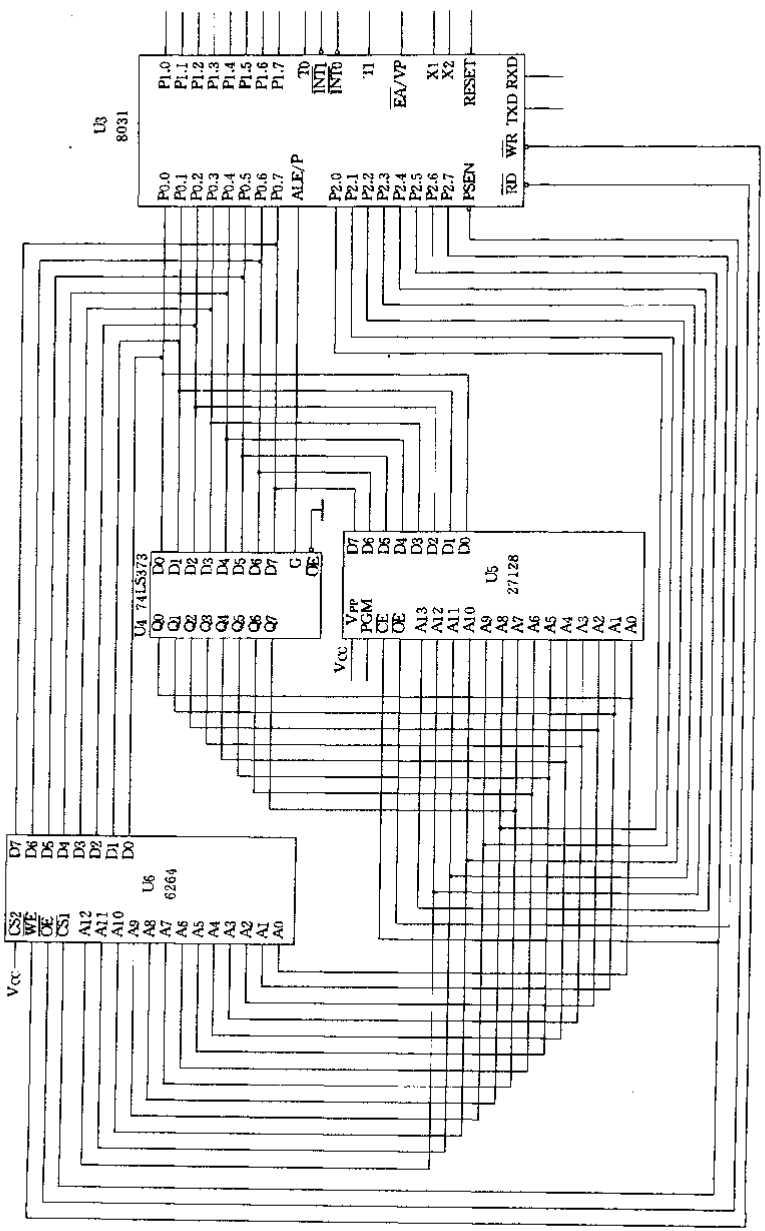


图7-17 8031外部存储器的典型结构

三、8031 外扩程序存储器与数据存储器的典型结构

随着大容量的芯片价格的进一步下降，在目前的扩展存储器中可选用一片 16K 字节的程序存储器 27128 和一片 8K 字节的数据存储器 6264，连接如图 7-17 所示。

如图连接，27128 的地址为 0000H～3FFFH；6264 地址范围为 0000H～1FFFH。

第三节　输入/输出与中断

一、输入/输出信息的传送方式

在微型计算机中，通常把主机(CPU 与存储器)以外的设备统称为外部设备或输入/输出设备，简称为 I/O 设备。常用的 I/O 设备有 CRT 显示器、键盘、打印机、磁盘驱动器、A/D、D/A 转换器等。

1. 输入输出设备与 CPU 的连接

外部设备通常是通过接口电路与主机相连接的。输入数据是输入设备把数据给接口，然后 CPU 从接口把数据取走；输出数据是 CPU 把数据送给接口，然后输出设备从接口获得数据输出。因此，CPU 与 I/O 设备交换信息，实际上是与这些接口交换信息。

采用接口电路来作为两者之间进行联系的桥梁，是由于外部设备本身的复杂性所决定的。一是外部设备的种类繁多，可以是机械式、电动式、电子式或其它形式。二是外部设备与 CPU 交换的信息是多种多样的，交换的信息有：(1)数据信息：可以是数字量、模拟量、开关量，而传送方式可以是串行传送或并行传送；(2)状态信息：可以有输入装置的是否“准备就绪”信号以及输出装置的“忙”和“闲”信号；(3)控制信息：可以有启动或停止外部设备的命令等。另外，输入信息的速率也相差很大，可以是手动的键盘输入，也可以是高速的硬盘输入等。

所以，采用接口电路，其作用就在于解决外设与 CPU 之间的信息传送的匹配问题。具体说，表现在以下几个方面：

(1)锁存作用

例如，计算机的内容需要输出打印时，输出数据在写周期($\overline{WR}$为低电平)时传送，而写周期一般约 1μs 左右。因此，被传送的数据仅在短暂的时间内呈现在数据总线上，同时传递给外部设备。这样短的时间用于驱动打字机的字头是不可能的，因为电磁机构根本来不及动作，而数据总线上的信息便消失了。为此，就要采用接口电路，在其中设置一个数据锁存器，将 CPU 输出的数据先放置在数据锁存器中锁存，然后再由外设慢慢地进行处理。这样，就可解决双方的速度匹配问题。

(2)隔离作用

CPU 与外设之间的信息交换和 CPU 与存储器之间的信息交换一样，也是通过 CPU 的数据总线来完成的。因此，外部设备和存储器都不允许长期占用数据总线，而仅允许被选中的设备在读写周期($\overline{RD}$和$\overline{WR}$为低电平)时享用数据总线。通过接口电路，可使每个设备的信息仅在以 CPU 发来的“允许信号”有效期间将数据 ID7～ID0 与总线 D7～D0 接

通，在其它时间该接口与总线相连接的线处于高阻浮空状态，起到与总线隔离的作用。这样各个设备就互不干扰，从而可使 CPU 高速利用数据总线。

(3)变换作用

例如，外部设备输入电流量通过接口电路可以变为电压量；或外设的电平幅度不符合计算机的要求，通过接口电路可以进行电平匹配。此外，有些外设(如磁带机、磁盘机以及通讯设备)以串行方式接收或发送数据，通过接口电路可将串行数据转换成并行数据，再经数据总线与 CPU 打交道。

(4)联络作用

通过接口电路，可使计算机和外部设备事先进行联系，当收、发双方都处于"就绪状态"时再进行交换数据，这样可以避免出错，并且提高效率。

目前，不少接口电路已采用大规模集成电路制成各种接口芯片，并已系列化、标准化，而且具有可编程的功能，因而使用十分方便灵活，下章将详细讲述。

CPU 只有找到唯一的接口(即唯一的 I/O 设备)，才能实现与指定的 I/O 设备交换信息。在 CPU 找到接口的条件下，讨论 I/O 数据传送才有意义。

2. I/O 编址方式

我们知道，CPU 与主存储器交换信息是按地址进行的。显然，CPU 与接口交换信息也必须采用同样的方法，即给每一个接口赋给一个唯一的地址，称为口地址，这样，CPU 通过地址总线发送口地址就可以找到指定的接口。口地址的安排有两种方法：一是把每一个接口都看成一个主存单元进行统一编址；二是对接口单独编址。

(1)接口与主存储器单元统一编址

把每一个接口当做一个存储器单元，即把主存的存储空间的一部分分给接口，进行统一编址。这样，对接口中的信息进行处理，就像对主存单元中的信息一样，可以使用全部访问主存的指令，而不必专设输入/输出指令。访问主存的指令相当多，功能也比较强，不仅可以实现数据输入/输出操作，而且还可对接口中所存信息进行算术、逻辑、移位等操作。但是，统一编址也有缺点，因为每一个接口都得给一定数量的缓冲区，仅软磁盘和打印机控制器、键盘和显示器缓冲区就要占上千个单元，所以，减少了主存的容量(由 16 位地址总线规定的)。苹果微型机与 MCS-51 单片机就采用此种方式编址。

(2)接口单独编址

为了不减少主存的存储空间，可以对接口单独编址。若对接口采用 8 位二进制编址，最多可有 256 个口地址。单独编址需要设置专用的 I/O 指令，例如，IBM-PC 及 Z80 微型机系统的 I/O 接口就是采用这种编址方式。由于 Z80 是单总线的系统结构，CPU 和主存或 CPU 和接口之间交换信息，都是通过同一的地址总线，那么究竟是访问主存储器还是访问 I/O 接口，是靠发不同的控制信号$\overline{\text{MREQ}}$或$\overline{\text{IORQ}}$等区别的。当$\overline{\text{MREQ}}$无效，而$\overline{\text{IORQ}}$有效时，表示访问 I/O 接口；反之，$\overline{\text{MREQ}}$有效，$\overline{\text{IORQ}}$无效，表示访问主存储器。这样，尽管口地址和主存地址有重叠区，也不会使 CPU 选错。

3. 输入/输出传送方式

单片机的运行与其它微机系统一样，CPU 不断地与外部输入/输出设备交换信息，CPU 与外部设备交换信息通常有如下几种方式：

(1)无条件传送方式

这种数据传送方式有点类似于CPU和存储器之间的数据传送,即CPU总是认为外设在任何时刻都是处于"准备好"的状态,因此,采用这种传送方式不需要交换状态信息,只需在程序中加入访问外设的指令,数据传送便可以实现。此种方法很少使用。因为无条件传送任何时候都不考虑外设是否准备好,经常造成错误。

(2)查询传送方式

为了解决CPU与外部设备匹配的问题,可以采用查询传送方式。查询传送方式也称条件传送。在这种传送方式中,不论是输入还是输出,都是以计算机为主动的一方。为了保证数据传送的正确性,计算机在传送数据之前,要首先查询外部设备是否处于准备好的状态。对于输入操作,需要知道外设是否已把要输入的数据准备好了;对于输出操作,则要知道外设是否把上一次计算机输出的数据处理完毕。只有通过查询,确信外设确实已处于"准备好"的状态,计算机才能发出访问的指令,实现数据的交换。

状态信息一般只需要一位二进制码,所以,在接口中只用一个D触发器就可用来保存和产生状态信息。例如,"准备好"用D触发器Q=1表示;"未准备好"用Q=0表示。图7-18(a)为查询方式程序的一般流程图。

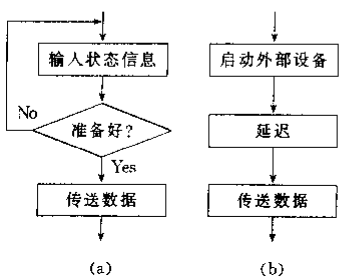


图7-18 查询方式流程图

查询方式的过程为:查询—等待—数据传送,待到下一次数据传送时则重复以上过程。等待也可以不采用循环等待,而用软件插入固定延时的方法来完成,如图7-18(b)所示。

查询方式的优点是通用性好,可以用于各类外部设备和CPU间的数据传送。缺点是需要有一个等待过程,特别是在连续进行数据传送时,由于外设工作比CPU慢得多,所以CPU在完成一次数据传送后要等待很长的时间(与数据传送相比)才能进行下一次的传送。在等待过程中,CPU不能进行其它操作,所以效率比较低。提高CPU效率的一个有效途径是采用中断方式。

(3)直接存储器存取(DMA)方式

DMA(Direct Memory Access)方式是CPU让出数据总线(悬浮状态),使外部设备和存储器之间直接传送(不通过CPU)数据的方式。下面两种情况时可考虑使用DMA方式:

①外设和存储器直接有大量的数据需要传送,如磁盘驱动器中的大量数据快速传送到微机系统的RAM中。

②外部设备的工作速度很快的情况。

(4)中断传送方式

为了提高CPU的利用率,避免查询等待时间,可以采用中断方式来传送数据。采用中断方式传送数据时,CPU不进行查询,而是依旧执行原来(与传送无关)的程序(主程序),当外设需要与CPU进行数据传送时,告知CPU,让CPU暂时停止当前的工作,而与外设传送信息,待与外设信息传送结束后,再回到原来的程序执行。

二、中断的概念

早期的计算机没有中断功能，主机和外设交换信息（数据）只能采用程序控制传送方式。如前所述，查询传送方式交换信息时，由于是CPU主动要求传送数据，而它又不能控制外设的工作速度，因此只能用等待的方式来解决速度的匹配问题，即CPU不能再做别的事，而大部分时间处于等待I/O接口准备好（就绪）状态。

现代的计算机都具有实时处理功能，能对外界异常发生的事件作出及时的处理，这是靠中断技术来实现的。

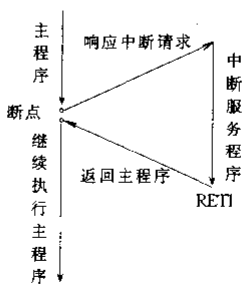


图7-19　中断流程

所谓中断是指中央处理器CPU正在处理某件事情的时候，外部又发生了某一事件（如一个电平的变化，一个脉冲沿的发生，或定时器计数溢出等），请求CPU迅速处理，于是，CPU暂时中断当前的工作，转入处理所发生的事件；中断服务处理完以后，再回到原来被中断的地方，继续原来的工作，这样的过程称为中断，如图7-19所示。实现这种功能的部件称为中断系统（中断机构），产生中断的请求源称为中断源。

对于数据传送，是外设主动提出要求，CPU在收到这个要求以前，执行本身的程序（主程序），只是在收到外设希望进行数据交换的请求之后，才中断原有主程序的执行，暂时地去进行与外设的数据交换。由于CPU工作速度很快，传送数据所花费的时间很短。对于主程序来讲，虽然中断了一个瞬间，由于时间很短，对计算机的运行也不会有什么影响。

中断方式完全消除了CPU在查询方式中的等待现象，大大提高了CPU的工作效率。没有中断技术，CPU的大量时间可能浪费在原地踏步的操作上。

一般计算机系统允许有多个中断源，8051单片机就有五个中断源。当几个中断源同时向CPU请求中断，要求CPU提供服务的时候，就存在CPU优先响应哪一个中断请求源的问题。一般根据中断源（所发生的实时事件）的轻重缓急排队，优先处理最紧急事件的中断请求，于是一些微处理器和单片机规定了每个中断源的优先级。当CPU正在处理一个中断请求，又发生了另一个优先级比它高的中断请求，CPU暂时中止对前一中断的处理，转而去处理优先级更高的中断请求，待处理完以后，再继续执行原来的中断处理程序。这样的过程称为中断嵌套，这样的中断系统称为多级中断系统。没有中断嵌套功能的中断系统称为单级中断系统。8051五个中断源有两个优先级，可实现二级中断嵌套，二级中断嵌套的中断过程如图7-20所示。

中断方式的另一个应用领域是实时控制。将从现场采集到的数据通过中断方式及时地传送给CPU，经过计算后就可立即作出响应，实现现场控制。而采用查询方式就很难做到及时采集，及时控制。

由于外界随机事件中断CPU正在执行的程序（只要允许的话）是随机的，CPU转向去执行中断服务程序时，除了硬件会自动把断点地址（16位PC程序计数器的值）压入堆

栈之外，用户还得注意保护有关工作寄存器、累加器、标志位等信息(通常称为保护现场)，以便在完成中断服务程序后，恢复原工作寄存器、累加器、标志位等的内容(称恢复现场)；最后执行中断返回指令，自动弹出断点到 PC，返回主程序，继续执行被中断的程序。

由于中断传送方式的优点极为明显，因此，在现代计算机系统中采用极为广泛。应该说，没有中断技术，就没有目前计算机的广泛应用。

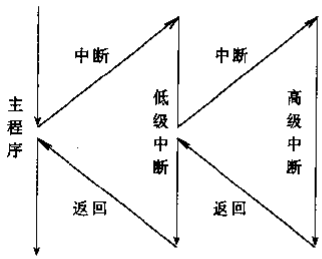


图7-20 中断嵌套流程

三、8051 中断系统结构及中断控制

8051 单片机中断系统的结构如图 7-21 所示。

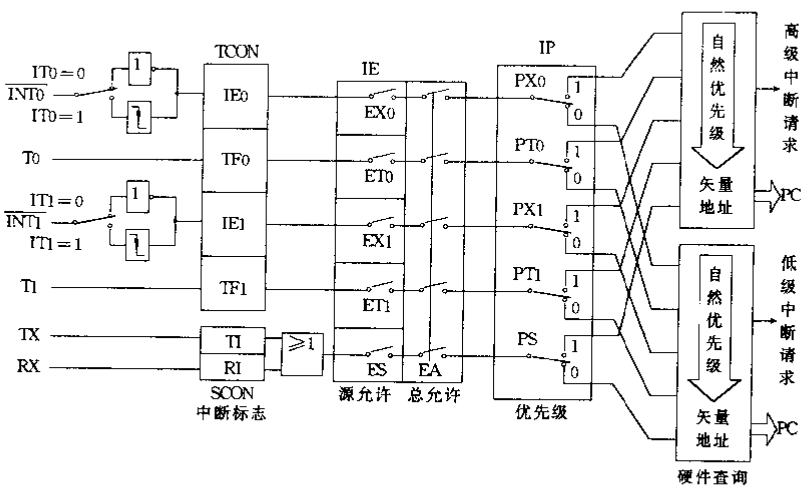


图7-21 中断字位结构

由图 7-21 结构可知，51 单片机有五个中断请求源，四个用于中断控制的寄存器 IE、IP、TCON(用 6 位)和 SCON(用 2 位)——用于控制中断的类型、中断的开/关和各种中断源的优先级别。五个中断源有二个中断优先级，每个中断源可以编程为高优先级或低优先级中断，可以实现二级中断服务程序的嵌套。

1. 中断源

8051 单片机的五个中断源包括：$\overline{INT0}$、$\overline{INT1}$引脚输入的外部中断源，三个内部中断源(定时器 T0、T1 的溢出中断源和串行口的发送/接收中断源)。

(1)外部中断源

从$\overline{INT0}$、$\overline{INT1}$输入的二个外部中断源和它们的触发方式控制位锁存在特殊功能寄存器 TCON 的低 4 位，其格式如下：

TCON (88H)	TF1		TF0		IE1	IT1	IE0	IT0

IE1(TCON.3)：外部中断$\overline{INT1}$(P3.3)请求标志位。当 CPU 检测到在$\overline{INT1}$引脚上出现的外部中断信号(低电平或脉冲下降沿)时，由硬件置位 IE=1，请求中断。CPU 响应中断进入中断服务程序后，IE1 位被硬件自动清 0(指脉冲边沿触发方式，电平触发方式时 IE1 不能由硬件清 0)。

IT1(TCON.2)：外部中断$\overline{INT1}$请求类型(触发方式)控制位。有软件来置 1 或清 0，以控制外部中断 1 的触发类型。

IT1=0 时，外部中断 1 程控为电平触发方式，当$\overline{INT1}$(P3.3)输入低电平时，置位 IE=1，申请中断。CPU 在每个机器周期的 S5P2 期间采样$\overline{INT1}$(P3.3)的输入电平，当采样到低电平时，置 IE1=1。采用电平触发方式时，外部中断源(输入到$\overline{INT1}$)必须保持低电平有效，直到该中断被 CPU 响应。同时，在该中断服务程序执行完之前，外部中断源有效电平必须被撤消，否则将产生另一次中断。

IT1=1，外部中断 1 程序控制为边沿触发方式，CPU 在每个机器周期的 S5P2 采样$\overline{INT1}$(P3.3)的输入电平。如果相继的两次采样，一个周期中采样到$\overline{INT1}$为高电平，接着下个周期中采样到$\overline{INT1}$为低电平，则置 IE1=1，表示外部中断 1 正在向 CPU 申请中断，直到该中断被 CPU 响应时，IE1 由硬件自动清 0。因为每个机器周期采样一次外部中断输入电平，因此，采用边沿触发方式时，外部中断源输入的高电平和低电平时间必须保持 12 个振荡周期以上，才能保证 CPU 检测到由高到低的负跳变。

IE0(TCON.1)：外部中断 0($\overline{INT0}$)请求标志位。IE0=1，外部中断 0 向 CPU 请求中断，当 CPU 响应外部中断时，IE0 由硬件清 0(指边沿触发方式)。

IT0(TCON.0)：外部中断 0($\overline{INT0}$)触发方式控制位。IT0=0，外部中断 0 程控为电平触发方式；IT0=1，外部中断 0 为边沿触发方式，其功能和 IT1 类似。

(2)内部中断源

TF0(TCON.5)：定时器 T0 的溢出中断申请位。TF0 实际上是 T0 中断触发器的一个输出端。T0 被允许计数以后，从初值开始加 1 计数，当产生溢出时置 TF0=1，向 CPU 请求中断，直到 CPU 响应该中断时才由硬件清 0(也可由查询程序清 0)。

TF1(TCON.7)：定时器 T1 的溢出中断申请位。定时器 T1 被允许计数以后，从初值开始加 1 计数，当产生溢出时置 TF1=1，向 CPU 请求中断，直到 CPU 响应该中断时才由硬件清 0(也可由查询程序清 0)。

SCON(98H)为串行口控制寄存器，SCON 的低两位锁存串行口接收中断和发送中断标志 RI 和 TI，其格式如下：

SCON		TI	RI

RI(SCON.0)和TI(SCON.1)：串行口内部中断申请标志位。串行口的接收中断RI和发送中断TI逻辑"或"以后作为内部的一个中断源。当串行口发送完一个字符后，由内部硬件置位发送中断标志TI；接收到一个字符后也由内部硬件置位接收中断标志RI。应该注意，CPU响应串行口的中断时，并不清零TI和RI中断标志，TI和RI必须由软件清0(中断服务程序中必须有清TI、RI的指令)。

2. 中断控制

除特殊功能寄存器TCON和SCON中某些位与中断有关外，还有二个特殊功能寄存器IE和IP专门用于中断控制。

(1)中断允许寄存器IE(A8H)

8051单片机中，特殊功能寄存器IE为中断允许寄存器，控制CPU对中断源的开放或屏蔽(禁止)，以及每个中断源是否允许中断。其格式为：

IE (A8H)	EA	—	—	ES	ET1	EX1	ET0	EX0

EA(IE.7)：CPU中断允许位。EA=1，CPU开放中断；EA=0，CPU屏蔽所有的中断请求。

ES(IE.4)：串行中断允许位。ES=1，允许串行口中断；ES=0，禁止串行口中断。

ET1(IE.3)：T1溢出中断允许位。ET=1，允许T1中断；ET1=0，禁止T1中断。

EX1(IE.2)：外部中断1($\overline{INT1}$)允许位。EX1=1，允许外部中断1中断；EX1=0，禁止外部中断1中断。

ET0(IE.1)：T0溢出中断允许位。ET0=1，允许T0中断；ET0=1，禁止T0中断。

EX0(IE.0)：外部中断0($\overline{INT0}$)允许位。EX0=1，允许外部中断0中断；EX0=0，禁止外部中断0中断。

8051系统复位后，IE中各位均被清0，即禁止所有中断。

(2)中断源优先级设定寄存器IP(B8H)

8051单片机具有二个中断优先级，每个中断源可编程为高优先级中断或低优先级中断，并可实现二级中断嵌套。

高优先级中断源可中断正在执行的低优先级中断服务程序，除非正在执行低优先级中断服务程序时设置了CPU关中断或禁止某些高优先级中断；同级或低优先级的中断源不能中断正在执行的中断程序。为此，在8051中断系统中，内部有两个(用户不能访问的)优先级状态触发器，它们分别指出CPU是否在执行高优先级或低优先级中断服务程序，从而分别屏蔽所有的低级中断申请和同一级的其它中断源申请。

特殊功能寄存器IP为中断优先级寄存器，锁存各种中断源优先级的控制位，用户可用软件设定，其格式如下：

IP (B8H)	—	—	—	PS	PT1	PX1	PT0	PX0

PS(IP.4)：串行口中断优先级控制位。PS=1，设定串行口为高优先级中断；PS=0，为低优先级中断。

PT1(IP.3)：T1 中断优先级控制位。PT1=1，设定定时器 T1 为高优先级中断；PT1=0，为低优先级中断。

PX1(IP.2)：外部中断 1 中断优先级控制位。PX1=1，设定外部中断 1 为高优先级中断；PX1=0，为低优先级中断。

PT0(IP.1)：T0 中断优先级控制位。PT0=1，设定定时器 T0 为高优先级中断；PT0=0，为低优先级中断。

PX0(IP.0)：外部中断 0 中断优先级控制位。PX0=1，设定外部中断 1 为高优先级中断；PX0=0，为低优先级中断。

8051 复位后，IP 低 5 位全部清 0，将所有中断源设置为低优先级中断。

如果几个同优先级的中断源同时向 CPU 申请中断，哪一个申请得到服务，取决于它们在 CPU 内部登记排队的序号。CPU 通过内部硬件查询登记序号，按自然优先级确定优先响应哪个中断请求。其内部登记序号是由硬件形成，排列如表 7-3 所示。这种顺序排列给实际应用提供了很大方便。

表 7-3　同优先级中断源登记序号

中断源	同级内部优先级
外部中断 0	最高级
定时器 T0 中断	↓
外部中断 1	↓
定时器 T1 中断	↓
串行口中断	最低级

四、中断响应过程及响应时间

1. 中断响应存在过程

8051 的 CPU 在每个机器周期的 S5P2 期间，顺序采样每个中断源，CPU 在下一个机器周期 S6 期间按优先级顺序查询中断标志，如查询到某个中断标志为 1，将在再下一个机器周期 S1 期间按优先级进行中断处理。中断系统通过硬件自动将相应的中断矢量地址装入 PC，以便进入相应的中断服务程序。

在下列任何一种情况存在时，中断申请将被封锁。

(1)CPU 正在执行一个同级或高一级的中断服务程序。

(2)当前周期(即查询周期)不是执行当前指令的最后一个周期，即要保证把当前的一条指令执行完才会响应。

(3)当前正在执行的指令是返回(RETI)指令或对 IE、IP 寄存器进行读/写指令，执行指令后至少再执行一条指令才会响应中断。

CPU 响应中断，由硬件自动将响应的中断矢量地址装入程序计数器 PC，转入该中断服务程序进行处理。对于有些中断源，CPU 在响应中断后会自动清除中断标志，如定时器

溢出标志 TF0、TF1，以及边沿触发方式下的外部中断标志 IE0、IE1；而有些中断标志不会自动清除，只能由用户用软件清除，如串行口的接收发送中断标志 RI、TI；在电平触发方式下的外部中断标志 IE0 和 IE1 则是根据引脚$\overline{INT0}$和$\overline{INT1}$的电平变化的，CPU 无法直接干预，需在引脚外加硬件（如 D 触发器）使其自动撤消外部中断请求。

CPU 执行中断服务程序之前，自动将程序计数器 PC 内容（断点地址）压入堆栈保护（但不保护状态寄存器 PSW 的内容，更不保护累加器 A 和其它寄存器的内容），然后将对应的中断矢量装入程序计数器 PC，使程序转向该中断矢量地址单元中，以执行中断服务程序。各中断源及与之对应的矢量地址见表 7-4。

表 7-4　中断源及其对应的矢量地址

中断源	中断矢量地址
外部中断 0($\overline{INT0}$)	0003H
定时器 T0 中断	000BH
外部中断 1($\overline{INT1}$)	0013H
定时器 T1 中断	001BH
串行口中断	0023H

中断服务程序从矢量地址开始执行，一直到返回指令“RETI”为止。“RETI”指令的操作，一方面告诉中断系统该中断服务程序已经执行完毕，另一方面把原来压入堆栈保护的断点地址从栈顶弹出，装入程序计数器 PC，使程序返回到被中断的程序断点处，以便继续执行。

我们在编写中断服务程序时应注意：

（1）在中断矢量地址单元处放一条无条件转移指令（如 JMP xxxxH），使中断服务程序可灵活地安排在 64K 字节程序存储器的任何空间。

（2）在中断服务程序中，用户应注意用软件保护现场，以免中断返回后，丢失原寄存器、累加器中的信息。

（3）若要在执行当前中断程序时禁止更高优先级中断，可以先用软件关闭 CPU 中断（CLR EA），或禁止某中断源中断，在中断返回前再开放中断。

2. 外部中断响应时间

外部中断$\overline{INT0}$和$\overline{INT1}$的电平在每个机器周期的 S5P2 期间，经反相锁存到 IE0 和 IE1 标志位，CPU 在下一机器周期才会查询到新置入的 IE0 和 IE1，这时如果满足响应条件，CPU 响应中断时要用二个机器周期执行一条硬件长调用指令“LCALL”，由硬件完成将中断矢量地址装入 PC，使程序转入中断矢量入口。所以，从产生外部中断到开始执行中断程序至少需要三个完整的机器周期。

如果在中断申请时，CPU 正在处理最长指令（如乘法和除法指令均为四个周期），则额外等待时间增加三个周期；若正在执行“RETI”或访问 IE、IP 指令，这时单片机规定，必须在 RETI 或读写 IE、IP 指令后，再执行一条指令，才允许进入响应状态，则额外时间又增加二个周期。

综合估计，在单一中断源系统里，外部中断响应时间约在 3～8 个机器周期之间。

五、中断举例

本节通过一个实例说明中断程序的设计方法并分析单级外部中断的过程。

实验线路如图 7-22 所示。单片机读 P1.0 的状态，把这个状态送到 P1.7 的指示灯去，当 P1.0 为高电平，指示灯亮；P1.0 为低电平时，指示灯不亮。要求用中断控制这一输入/输出过程，每请求中断一次，完成一个读/写过程。

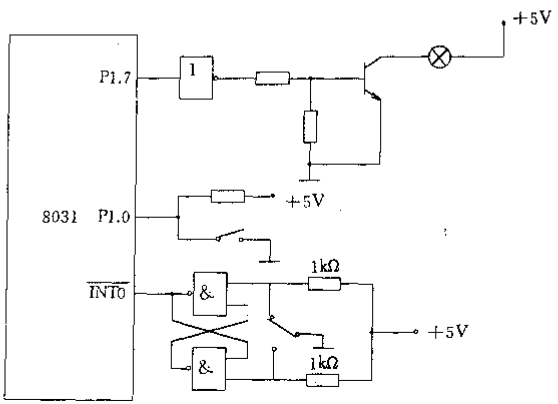


图7-22　外部中断实验图

1. 中断方式的选择

8051 提供了两根 I/O 口线作为请求中断信号的输入端($\overline{INT0}$和$\overline{INT1}$)，如果中断请求信号接在$\overline{INT0}$上，选择的是外部中断 0。

外部中断可用边沿触发，也可选择电平触发。如果选择边沿触发，在程序中应设一条 SETB IT0 指令。在硬件上，应在$\overline{INT0}$脚接一个负脉冲产生电路，发出中断请求信号。为了保证中断的唯一性，应该加上去抖电路，每次只产生单脉冲，因此可构成图 7-19 那样的实验线路。

边沿触发的最大优点在于不会丢失中断。只要中断请求负跳变的宽度大于 1 个机器周期，单片机就能采样到中断请求信号。单片机将采样到的信号锁存到中断请求标志寄存器中，硬件自动置 IE0 为 1。即使单片机暂时不响应，这个标志也不丢失，只有在外部中断响应之后，硬件才将 IE0 清除。

2. 逐级开放中断

有了中断请求不一定能产生中断，还必须利用程序将各个“允许/禁止中断”开关设置成开放状态。8051 的中断控制有两级，一级是针对不同中断源的，开放外部中断 0 只要用指令 SETB EX0 即可。另一级是管总的，必须用指令 SETB EA，这些也应放在中断程序初始化中进行。也可用指令 CLR EA 随时关闭中断。

3. 中断服务子程序的位置

根据约定，外部中断 0 的入口地址在 0003H 单元，而中断服务子程序则可放在程序存储器的任何地方，但必须在 0003H 单元放一条跳转指令，指向中断服务子程序的起始地址。中断服务子程序最后一条指令必须是 RETI 指令，以便在结束时能返回到被中断的主程序。

根据以上讨论，可编写实验程序如下：

```
        ORG     0000H
        AJMP    MAIN        ;上电自动转向主程序
        ORG     0003H       ;外部中断 0 入口地址
        AJMP    WINT        ;指向中断服务子程序
        ORG     0100H       ;主程序
MAIN:   SETB    IT0         ;选择边沿触发方式
        SETB    EX0         ;允许外部中断 0
        SETB    EA          ;CPU 允许中断
HERE:   AJMP    HERE        ;主程序踏步
```

以下是中断服务子程序：

```
        ORG     0200H
WINT:   MOV     A,#0FFH
        MOV     P1,A        ;设输入态
        MOV     A,P1        ;取开关数
        RR      A           ;P1.0 送 P1.7
        MOV     P1,A        ;输出驱动灯泡发光
        RETI                ;中断返回
        END
```

4. 中断响应的过程

上电之后，由 0000H 单元自动跳到主程序执行，主程序完成中断初始化程序之后，立即进入到指令：

HERE: AJMP HERE

这是一条跳转指令，每执行一次，仍然跳回到原处，因此是一个踏步动作，它相当于一个很长的主程序，一直执行下去，等待中断的到来。

当外设数据准备好以后（重设开关一次），按下单脉冲发生键，向 8031 的$\overline{\text{INT0}}$脚输送一个负脉冲——请求中断的信号。

单片机在每个机器周期的 S5P2 期间对$\overline{\text{INT0}}$采样（注意此处选择为边沿触发），如果连续采样到一个周期为高电平，下一个周期紧接着为低电平，则硬件自动将 TCON 寄存器的中断请求标志位 IE0 置位，由 IE0 标志请求中断（保存中断请求）。

单片机对中断源的查询实际并不限于$\overline{\text{INT0}}$，在每个机器周期内，对所有中断源都进行顺序检查，由于这个实验只考虑单级中断，因此，在单片机检查到外部中断 0 有中断请求时，在当前指令执行完毕之后，下一个机器周期的 S1 期间开始响应。在响应期间单片机

自动完成一系列复杂的动作：

(1)将相应的优先级有效触发器置位，清除中断请求标志(此处，令 IE0=0)。

(2)执行一个硬件子程序，把程序计数器的内容(主程序被中断处的地址)压入堆栈。

(3)把请求中断的相应中断入口地址(此处为 0003H)装入 PC。

(4)由 0003H 再跳到中断服务子程序——这就是响应中断。

(5)当中断服务子程序的指令全部执行完毕，最后执行 RETI，单片机又自动将优先级有效触发器复位，把保存在堆栈的主程序返回地址重新装入 PC，使主程序继续执行下去。

以上就是一个中断的全过程。

第四节　定时器/计数器

一、定时器概述

8051 单片机内有二个 16 位定时器/计数器：定时器 0(T0)和定时器 1(T1)，它们都有定时或对外部事件计数的功能，可用于定时控制、延时、对外部事件检测和计数等场合。

定时器 T0 和 T1 二个 16 位定时器实际上都是 16 位加 1 计数器。T0 实际是由两个 8 位特殊功能寄存器 TH0(8CH)和 TL0(8AH)组成，T1 是由 TH1(8DH)和 TL1(8BH)组成。每个定时器都可由软件设置为定时工作方式或计数工作方式及其它灵活多样的可控功能方式。这些都是由特殊功能寄存器 TMOD 设置和 TCON 控制。

设置为定时工作方式时，定时器计数 8051 片内振荡器输出经 12 分频后的脉冲(机器周期信号)。即每个机器周期使定时器(T0 或 T1)的数值增加 1 直至计满溢出。当 8051 采用 12MHz 晶体时，一个机器周期为 1μs，计数频率为 1MHz。

设置为计数工作方式时，通过引脚 T0(P3.4)和 T1(P3.5)对外部脉冲信号计数。当输入脉冲信号产生由 1 至 0 的下降沿时，定时器的值增加 1。在每个机器周期的 S5P2 期间采样 T0 和 T1 引脚的输入电平，若前一个机器周期采样值为 1，下一个机器周期采样值为 0，则计数器加 1。此后的机器周期 S3P1 期间，新的数值装入计数器。所以，检测一个 1 至 0 的跳变需要二个机器周期，故最高计数频率为振荡频率的 1/24。虽然对输入信号的占空比无特殊要求，但为了确保某个电平在变化之前至少被采样一次，要求电平保持时间至少是一个完整的机器周期。对输入脉冲信号的基本要求如图 7-23 所示。

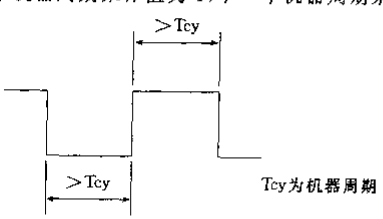


图7-23　对输入脉冲宽度的要求

除了可以选择定时器或计数器工作方式外，每个定时器/计数器还有四种工作方式，也就是每个定时器可构成四种电路结构方式。其中，0～2 方式对 T0 和 T1 都是一样的，方式 3 对两者是不同的。

二、定时器的控制字

定时器共有二个控制字，由软件写入 TMOD 和 TCON 二个 8 位寄存器，用来设置 T0 或 T1 的操作方式和控制功能。当 8051 系统复位时，二个寄存器所有位都被清 0。

1. 工作方式寄存器 TMOD(89H)

TMOD 用于控制 T0 和 T1 的操作方式，其各位的定义格式如下：

	D7	D6	D5	D4	D3	D2	D1	D0
TMOD (89H)	GATE	C/$\overline{T}$	M1	M0	GATE	C/$\overline{T}$	M1	M0
	← 定时器 T1 →				← 定时器 T0 →			

其中，低 4 位用于 T0，高 4 位用于 T1。各位功能如下：

(1)M1 和 M0：操作方式控制位。2 位可形成 4 种编码，对应 4 种工作方式，见表 7-5。

表 7-5 M1、M0 控制的四种操作方式

M1 M0	工作方式	功能描述
0 0	方式 0	13 位计数器
0 1	方式 1	16 位计数器
1 0	方式 2	自动再装入 8 位计数器
1 1	方式 3	定时器 0：分成二个 8 位计数器 定时器 1：对外部停止计数

(2)C/$\overline{T}$：计数器方式/定时器方式选择位。C/$\overline{T}$=0，设置为定时方式，定时器计数 8051 片内脉冲，即对机器周期(时钟周期的 12 倍)计数；C/$\overline{T}$=1，设置为计数方式，计数器的输入是来自 T0(P3.4)或 T1(P3.5)端的外部脉冲。

(3)GATE：门控位。GATE=0 时，只要用软件使 TR0(或 TR1)置 1 就启动了定时器，而不管$\overline{INT0}$(或$\overline{INT1}$)的电平是高还是低；GATE=1 时，只有$\overline{INT0}$(或$\overline{INT1}$)引脚为高电平且由软件使 TR0(或 TR1)置 1 时，才能启动定时器工作。

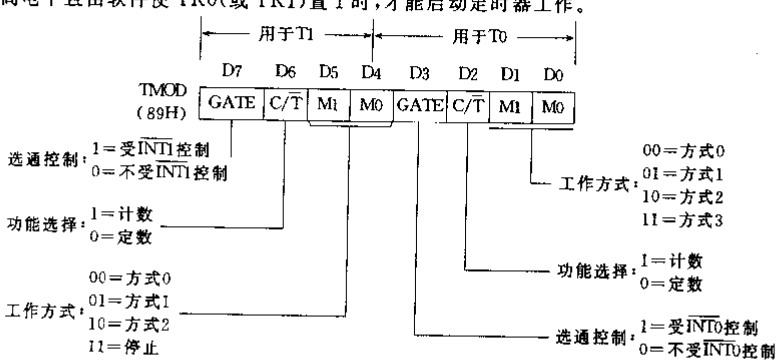


图7-24 TMOD各位定义

TMOD 不能位寻址，只能按字节设置定时器工作方式，低半字节设定 T0，高半字节设定 T1。归纳如图 7-24 所示。

2. 控制寄存器 TCON(88H)

定时器控制寄存器 TCON 除可字节寻址外，各位还可位寻址，各位定义及格式如下：

	8FH	8EH	8DH	8CH	8BH	8AH	89H	88H
TCON (88H)	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TCON 各位的作用如下：

TF1(TCON.7)：T1 溢出标志位。当 T1 溢出时，由硬件自动使振荡触发器 TF1 置 1，并向 CPU 申请中断。当 CPU 响应中断服务程序后，TF1 又被硬件自动清 0。TF1 也可以用软件清 0。

TF0(TCON.5)：T0 溢出标志位。其功能和操作情况同 TF1。

TR1(TCON.6)：T1 运行控制位。可由软件置 1 或清 0 来启动或关闭 T1。在程序中用一条指令(SETB TR1)使 TR1 位置 1，定时器 T1 便开始计数。

TR0(TCON.4)：T0 运行控制位。其功能和操作情况同 TR1。

以上 4 位控制 T1 和 T0 以定时器方式运行或中断。

IE1、IT1、IE0 和 IT0(TCON.3～TCON.0)：外部中断$\overline{INT1}$、$\overline{INT0}$请求及请求方式控制位。这 4 位的功能已在中断一节讲述过。

8051 复位时，TCON 的所有位被清 0。各位功能归纳如图 7-25。

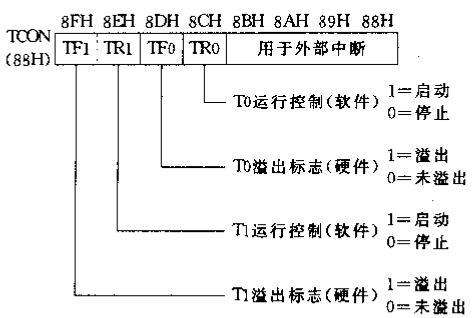


图7-25 TCON各位定义

三、定时器的 4 种工作方式及应用举例

8051 单片机的定时器/计数器可由软件对 TMOD 中控制位 C/$\overline{T}$的设置，以选择定时功能或计数功能。无论哪种功能，均可再由 M1 和 M0 位的设置，选择为 4 种工作方式之一：即方式 0、方式 1、方式 2 和方式 3。对于方式 0、1、2，T0 与 T1 的工作模式相同；对于方式 3，只有 T0 才能设置。

1. 方式 0

方式 0 是选择定时器(T0 或 T1)高 8 位加低 5 位组成一个 13 位的定时器/计数器，其逻辑电路结构如图 7-26 所示(以 T0 为例)。

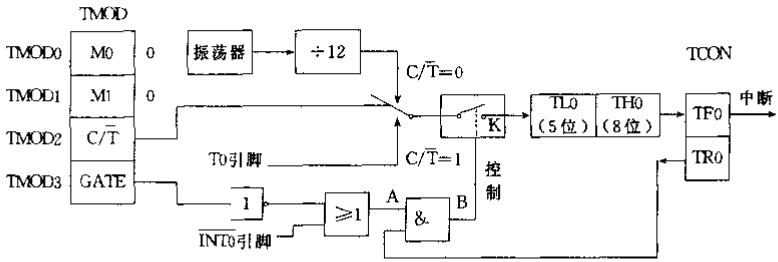


图7-26 定时器0方式0——13位计数器

在这种方式下，16位寄存器(TH0和TL0)只用13位，其中TL0的高3位未用而其余位占整个13位的低5位，TH0占整个13位高8位。当TL0的低5位溢出时，向TH0进位，而TH0溢出时向中断标志位TF0进位(硬件置位TF0)，并申请中断。T0溢出否可查询TF0是否置位。

在图7-26中，$C/\overline{T}=0$时，控制开关接通振荡器十二分频输出端，T0对机器周期计数，这就是定时器工作方式。其定时时间为

$$t=(2^{13}-\text{T0初值})\times\text{机器周期}$$

当$C/\overline{T}=1$时，控制开关使引脚T0(P3.4)与13位计数器相连，外部计数脉冲由引脚T0(P3.4)输入，当外部信号电平发生"1"到"0"跳变时，计数器加1，这时，T0成为外部事件计数器。这就是计数工作方式。

GATE=0时，使或门输出A点电位为常"1"，或门被封锁，于是，引脚$\overline{INT0}$输入信号无效。这时或门输出的常"1"打开与门，B点电位取决于TR0状态，于是由TR0一位就可控制计数开关K开启或关断T0。若软件使TR0置1，便接通计数开关K，启动T0在原值上加1计数，直至溢出。溢出时，13位寄存器清0，TF0置位，并申请中断，T0仍从0重新开始计数。若TR0=0，则关断计数开关K，停止计数。

当GATE=1时，A点电位取决于$\overline{INT0}$(P3.4)引脚的输入电平。仅当$\overline{INT0}$输入高电平且TR0=1时，B点才是高电平，计数开关K闭合T0开始计数，当$\overline{INT0}$由1变0时，T0停止计数。这一特性可以用来测量在$\overline{INT0}$端出现的正脉冲的宽度。

例：利用定时器/计数器每隔1ms控制产生宽度为二个机器周期的负脉冲，由P1.0送出。设时钟频率为12MHz。

为了提高CPU的效率，采用中断工作方式。

首先求定时器初值，设定时器初值为X，则定时1ms时，应有

$$(2^{13}-X)\times10^{-6}=1\times10^{-3}$$

式中机器周期为1ms，可求得X=7096=11011101 11000B，其中高8位DDH赋给TH0，低5位18H赋给TL0。由于系统复位后，TMOD被清，正好处于定时器方式0状态，且GATE=0，也可不设置TMOD。程序如下：

```
        ORG     0000H
        AJMP    MAIN
```

```
        ORG     000BH
        AJMP    TOINT
        ORG     100H
MAIN:   MOV     TH0,#0DDH
        MOV     TL0,#18H        ;送定时初值
        MOV     IE,#82H         ;允许 T0 中断 EA=1,ET0=1
        SETB    TR0             ;启动定时器 0
LOOP:   SJMP    LOOP
        ORG     200H
TOINT:  CLR     P1.0
        SETB    P1.0            ;送 2μs 负脉冲,CLR P1.0 指令需二个机器周期时间
        MOV     TH0,#0DDH       ;用软件重新装载 TH0 和 TL0
        MOV     TL0,#18H
        RETI
```

上述过程只是完整的软件中的一小部分，CPU 还要完成大量其它任务，(用许多程序代替 LOOP:SJMP LOOP 语句)，而 1ms 产生一个脉冲，其间 CPU 可以进行大量的操作，故采用定时器定时，且采用中断方式，提高 CPU 的工作效率。

2. 方式 1

在方式 1 中定时器 0 与定时器 1 的应用相同，我们以定时器 0 为例。该方式是一个 16 位定时器/计数器，见图 7-27。在方式 1 中，寄存器 TH0 和 TL0 是以全 16 位参与操作，用于定时工作方式时，定时时间为

$$t=(2^{16}-\text{T0 初值})\times\text{机器周期}$$

用于计数工作方式时，计数长度为 $2^{16}=65536$(个外部脉冲)。

方式 1 与方式 0 基本相同，只是方式 1 改用了 16 位计数器。要求定时周期较长时，13 位计数器不够用，可改用 16 位计数器。

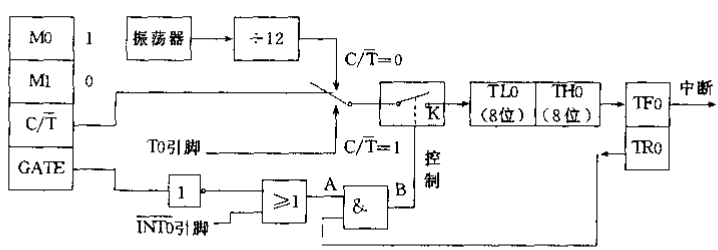
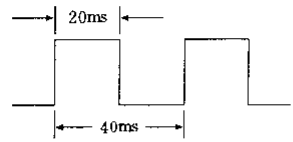


图7-27 定时器0方式1——16位计数器

例如，利用定时器 0 产生 25Hz 的方波，由 CPU 输出。假定 CPU 不做其它工作，则可采用查询方式进行控制，设晶振频率为 12MHz。

25Hz 方波，周期为 1/25=40ms，可以采用定时器定时 20ms 时间，每隔 20ms 变化一

下 P1.0 的高低电平，即可得到 25Hz 的方波信号。若采用定时器 0 方式 0，则最大定时时间为 $t = 2^{13} \times$ 机器周期 $= 2^{13} \times 1 \times 10^{-6} = 8.192$ms。显然一次定时不能满足要求，我们可采用定时器 0 方式 1 工作。



设初值为 X，则有

$$t = (2^{16} - X) \times 1 \times 10^{-6} = 20 \times 10^{-3}$$

求得 X＝45536＝B1E0H，设计程序如下：

```
        ORG     100H
        MOV     TMOD,#01H
        SETB    TR0
LOOP:   MOV     TH0,#0B1H
        MOV     TL0,#0E0H
        JNB     TF0,$       ;$为当前指令指针地址
        CLR     TF0
        CPL     P1.0
        SJMP    LOOP
        END
```

应该注意，TMOD 中的各位不是可直接寻址位，因此不能用 SETB TMOD.0 指令，否则汇编时出错。

3. 方式 2

方式 2 把 TL0(TL1)配置成一个可以自动重装载的 8 位定时器/计数器，以 T0 为例，如图 7-28 所示。

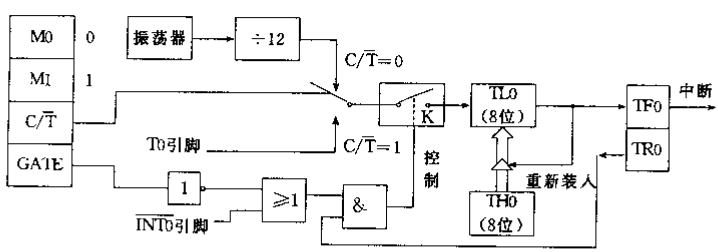


图7-28 T0方式2结构

TL0 计算溢出时，不仅使溢出中断标志位 TF0 置 1，而且还自动把 TH0 中的内容重装载到 TL0 中。这里 16 位的计数器被拆成二个，TL0 用作 8 位计数器，TH0 用以保持初值。

在程序初始化时，TL0 和 TH0 由软件赋于相同的初值。一旦 TL0 计数溢出，置位 TF0，并将 TH0 中的初值再自动装入 TL0，继续计数，循环重复。用于定时器工作方式时，其定时时间(TF0 溢出周期)为

$$t = (2^8 - \text{TH0 初值}) \times \text{机器周期}$$

用于计数器工作方式时，最大计数长度(TH0 初值为 0)为 $2^8=256$(个外部脉冲)。

这种工作方式可省去用户软件中重装常数的程序，并可产生相当精度的定时时间，特别适于作串行口波特率发生器(详见下一节)。

4. 方式 3

操作方式 3 对 T0 和 T1 是大不相同的。

若将 T0 设置为方式 3，TL0 和 TH0 被分成为两个互相独立的 8 位计数器，如图 7-29 所示。

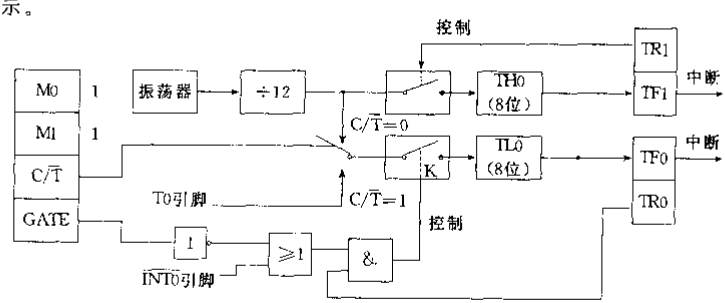


图7-29 T0方式3结构

其中 TL0 用原 T0 的各控制位、引脚和中断源，即 C/T、GATE、TR0、TF0 和 T0(P3.4)引脚、INT0(P3.2)引脚。TL0 除仅用 8 位寄存器外，其功能和操作与方式 0(13 位计数器)、方式 1(16 位计数器)完全相同。TL0 也可设置为定时器方式或计数器方式。

TH0 只有简单的内部定时功能(见图 7-29 上半部分)，它占用了定时器 T1 的控制位 TR1 和 T1 的中断标志位 TF1，其启动和关闭仅受 TR1 的控制。

定时器 T1 无操作方式 3 状态，若将 T1 设置为方式 3，就会使 T1 立即停止计数，也就是保持原有的计数值，其作用相当于使 TR1=0，封锁与门，断开计数开关 K。

在定时器 T0 用作方式 3 时，T1 仍可设置为方式 0～2。由于 TR1 和 TF1 被定时器 T0(TH0)占用，计数器开关 K 已被接通，此时仅用 T1 控制位 C/T 切换其定时器或计数器工作方式就可使 T1 运行。寄存器(8 位、13 位或 16 位)溢出时，只能将输出送入串行口或用于不需要中断的场合。在一般情况下，当定时器 T1 用作串行口波特率发生器时，定时器 T0 才设置为工作方式 3。此时，常把定时器 T1 设置为方式 2，用作波特率发生器，见图 7-30 所示。

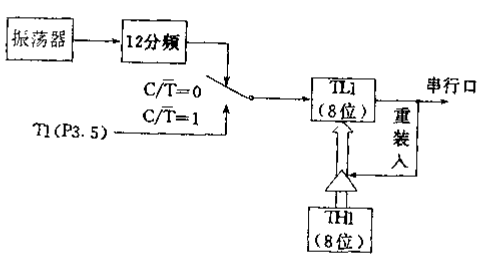


图 7-30 T1 方式 2 结构

定时器0操作方式3是二个8位定时器/计数器方式，且TH0借用了定时器1的溢出中断标志TF1和运行控制位TR1。

假设有一个用户系统中已使用了二个外部中断源，并置定时器1于方式2，作串行口波特率发生器用，现要求再增加一个外部中断源并由P1.0口输出一个5kHz的方波。

为了不增加其它硬件开销，可把定时器/计数器置于计数器工作方式3，利用T0端作附加的外部中断输入端，把TL0预置为0FFH，这样在输出端出现由1至0的负跳变时，TL0立即溢出，申请中断，相当于边沿激活的外部中断源。在方式3下，TH0总是作为8位定时器用，可以靠它来控制由P1.0输出的方波频率。

由P1.0输出5kHz的方波，即每隔100μs使P1.0的电平变化一次。若采用12MHz的晶体，把TH0预置为156，则TH0溢出周期为100μs。

有关程序如下：

```
        ORG     0000H
        AJMP    MAIN
        ORG     0003H
        AJMP    INT0
        ORG     000BH
        AJMP    TL0INT
        ORG     0013H
        AJMP    INT1
        ORG     001AH
        AJMP    TH0INT
        ORG     0023H
        AJMP    SI0
        ORG     100H
MAIN:   MOV     TL0,#0FFH
        MOV     TH0,#156
        MOV     TL1,#BAUD   ;BAUD是根据波特率要求设置的常数
        MOV     TH1,#BAUD
        MOV     TMOD,#27H   ;置定时器0为方式3，使TL0工作于
                            计数器方式
        MOV     TCON,#55H   ;置外部中断0和1于边沿激活方式，
                            启动定时器0和1
        MOV     IP,#05H     ;外部中断高优先级
        MOV     IE,#9FH
INT0:    ⋮
        RETI
INT1:    ⋮
```

```
        RETI
TLOINT: MOV     TL0,#0FFH
        ⋮
        RETI
THOINT: MOV     TH0,#156
        CPL     P1.0
        RETI
SIO:    ⋮
        RETI
```

第五节　串行通信

8051单片机除具有四个8位并行口外，还具有串行接口。此串行接口是一个全双工串行通信接口，即能同时进行串行发送和接收。它可以作UART(通用异步接收和发送器)用，也可以作同步位移寄存器用。应用串行接口可以实现8051单片机系统之间点对点的单机通信、多机通信和8051与系统机(如IBM-PC机等)的单机或多机通信。

一、串行通信及基础知识

1. 数据通信的概念

在实际工作中，计算机的CPU与外部设备之间常常要进行信息交换，一台计算机与其它计算机之间也往往要交换信息，所有这些信息交换均可称为通信。

通信方式有两种，即并行通信和串行通信。通常根据信息传送的距离决定采用哪种通信方式。例如，在IBM-PC机与外部设备(如打印机等)通信时，如果距离小于30m，可采用并行通信方式；当距离大于30m时，则要采用串行通信方式。8051单片机具有并行和串行二种基本通信方式。

并行通信是指数据的各位同时进行传送(发送或接收)的通信方式。其优点是传递速度快；缺点是数据有多少位，就需要多少根传送线。例如8051单片机与打印机之间的数据传送就属于并行通信(8位数据并行通信)。并行通信在位数多、传送距离又远时就不太适宜。

串行通信指数据是一位一位按顺序传送的通信方式，它的突出优点是只需一对传送线(利用电话线就可作为传送线)，这样就大大降低了传送成本，特别适用于远距离通信；其缺点是传送速度较低。

2. 串行通信的传送方向

串行通信的传送方向通常有三种：一种为单工(或单向)配置只允许数据向一个方向进行传送；另一种是半双工(或半双向)配置，允许数据向两个方向中的任何一方向传送，但一次只能有一个发送，一个接收；第三种传送方式是全双工(或全双向)配置，允许同时双向传送数据，因此，全双工配置是一对单工配置，它要求两端的通信设备都具有完整和独立的发送和接收能力。

3. 异步通信和同步通信

串行通信有两种基本通信方式，即异步通信和同步通信。

(1)异步通信

在异步通信中，数据是一帧一帧(包含一个字符代码或一字节数据)传送的，每一串行帧的数据格式如图 7-31 所示。

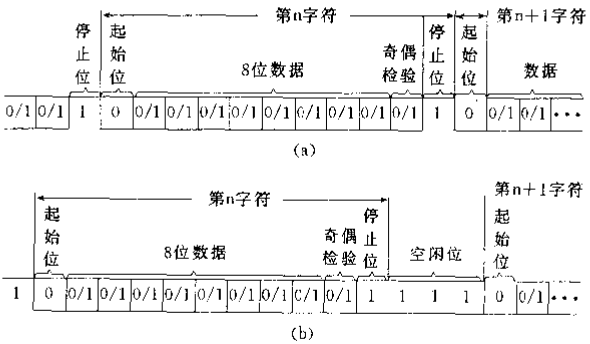


图7-31 异步通信的一帧数据格式

在帧格式中，一个字符由四个部分组成：起始位、数据位、奇偶校验位和停止位。首先是一个起始位“0”，然后是 5～8 个数据位(规定低位在前，高位在后)，接下来是奇偶校验位(可省略)，最后是停止位“1”。起始位“0”信号只占用一位，用来通知接收设备一个待接收的字符开始到来。线路上在不传送字符时应保持为“1”。接收端不断检测线路的状态，若连续为“1”以后测到一个“0”，则可知发来一个新字符，应马上准备接收。字符的起始位还被用作同步接收端的时钟，以保证以后的接收能正确进行。

起始位后面紧接着是数据位，它可以是 5 位(D0～D4)、6 位、7 位或 8 位(D0～D7)。

奇偶校验位(D8)只占一位，但在字符中也可以规定不同奇偶校验位，则这一位就可省去。也可用这一位(1/0)来确定这一帧中的字符所代表信息的性质(地址/数据等)。

停止位用来表征字符的结束，它一定是高电位(逻辑“1”)。停止位可以是 1 位、1.5 位或 2 位。接收端接收到停止位后，知道上一字符已传送完毕，同时，也为接收下一字符作好准备——只要再收到“0”就是新的字符的起始位置。若停止位以后不是紧接着传送下一字符，则让线路上保持为“1”。图 7-31(a)表示一个字符紧接着一个字符传送的情况，上一个字符的停止位和下一个字符的起始位是紧相邻的；图 7-31(b)则是两个字符间有空闲的情况，空闲为“1”，线路处于等待状态。存在空闲位正是异步通信的特征之一。

例如，规定用 ASCII 编码，字符为 7 位加一个奇偶校验位、一个起始位、一个停止位，则一帧共 10 位。

(2)同步通信

同步通信中，在数据开始传送前用同步字符来指示(常约定 1～2 个)，并由时钟来实现发送端和接收端同步，即检测到规定的同步字符后，下面就连续按顺序传送数据，直到通信告一段落。同步传送时，字符与字符之间没有间隙，也不用起始位和停止位，仅在数据

块开始时用同步字符 SYNC 来指示，其数据格式如图 7-32 所示。

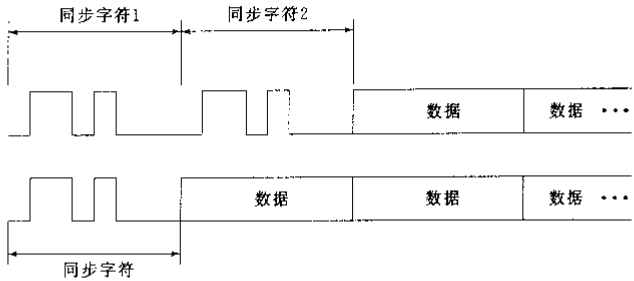


图7-32 同步通信的数据格式

同步字符的插入可以是单同步字符方式或双同步字符方式，如图 7-32。然后是连续的数据块。同步字符可以由用户约定，当然也可以采用 ASCII 码中规定的 SYN 代码，即 16H。按同步方式通信时，先发送同步字符，接收方检测到同步字符后，即准备接收数据。

在同步传送时，要求用时钟来实现发送端与接收端之间的同步。为了保证接收正确无误，发送方除了传送数据外，同时还要传送时钟信号。

同步传送的优点是可以提高传送速率（达 56kbps 或更高），但硬件比较复杂。

(3)波特率(Band Rate)

波特率，即数据传送速率，表示每秒钟传送二进制代码的位数，它的单位是波特(bps，位/s)。波特率对于 CPU 与外界的通信是很重要的。假设数据传送速率是 120 字符/s，而每个字符格式包含 10 个代码位（一个起始位、一个终止位、8 个数据位），这时传送的波特率为

$$10\times 120\ 位/s=1200bps$$

每一位代码的传送时间 T_d 为波特率的倒数

$$T_d=1/1200=0.833ms$$

波特率是衡量传送通道频宽的指标，它和传送数据的速率并不一致。如上例中，因为除掉起始位和终止位，每一个数据实际只占 8 位。所以数位的传送速率为

$$8\times 120=960\ 位/s$$

异步通讯的传送速度在 50 到 64 000bps 之间。常用于计算机到终端和打印机之间的通信、直通电报以及无线电通讯的数据发送等。

二、串行接口结构与控制寄存器

8051 有一个可编程的双工串行通信接口，它可作为通用异步接收和发送器 UART，也可作为同步移位寄存器。其帧格式可有 8 位、10 位和 11 位，并能设置各种波特率，给使用者带来很大的灵活性。

1. 结构

8051 通过引脚 RXD(P3.0 串行数据接收端)和引脚 TXD(P3.1 串行数据发送端)与

外界进行通信。8051 串行口内有一个可直接寻址的专用寄存器——串行口缓冲寄存器 SBUF。在物理上，SBUF 由二个 8 位寄存器组成：一个是发送寄存器，一个是接收寄存器，两者共用一个地址 99H，可同时发送、接收数据。CPU 写 SBUF，就是修改发送寄存器；读 SBUF，就是读接收寄存器。接收寄存器是双缓冲的，以避免在接收下一帧数据之前，CPU 未能及时响应接收寄存器的中断，没有把上一帧数据读走，而产生两帧数据重叠的问题。对于发送寄存器，为了保持最大的传输速率，一般不需要双缓冲，因为发送时 CPU 是主动的，不会产生写重叠的问题。

2. 串行口控制寄存器 SCON(98H)

8051 串行通信的方式选择，接收和发送控制以及串行口的状态标志等均由特殊功能寄存器 SCON 控制和指示，其控制字格式如图 7-33 所示。

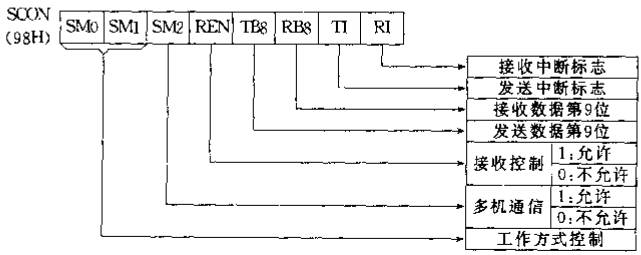


图7-33 SCON：串行口控制寄存器

(1)SM0 和 SM1(SCON. 7、SCON. 6)

串行口工作方式选择位。二个选择位对应四种通信方式(见表 7-6)，其中 f_{osc}是振荡频率。

表 7-6 串行口的工作方式

SM0	SM1	工作方式	说明	波特率
0	0	方式 0	同步移位寄存器	$f_{osc}/12$
0	1	方式 1	10 位移位收发	由定时器控制
1	0	方式 2	11 位移位收发	$f_{osc}/32$ 或 $f_{osc}/64$
1	1	方式 3	11 位移位收发	由定时器控制

(2)SM2(SCON. 5)

多机通信控制位，主要用于方式 2 和方式 3。若置 SM2=1，则允许多机通信。多机通信协议规定，第 9 位数据(D8)为 1，说明本帧数据为地址帧；若第 9 位为 0，则本帧为数据帧。当一个 8051(主机)与多个 8051(从机)通信时，所有从机的 SM2 位都置 1。主机首先发送的一帧数据为地址，即某从机机号，其中第 9 位为 1，被寻地址的某个从机接收到数据后，将其中第 9 位装入 RB8。从机依据收到的第 9 位数据(RB8 中)的值来决定从机可否再接收主机的信息，若(RB8)=0，说明是数据帧，则使接收中断标志位 RI=0，信息丢失；

若(RB8)=1,说明是地址帧,数据装入 SBUF 并置 RI=1,中断所有从机,被寻址的目标从机清除 SM2 以接收主机发来的一帧数据。其它从机仍然保持 SM2=1。

若 SM2=0,即不属于多机通信情况,则接收一帧数据后,不管第 9 位数据是 0 还是 1,都置 RI=1,接收到的数据装入 SBUF 中。

根据 SM2 这个功能,可实现多个 8051 应用系统的串行通信。在方式 1 时,若 SM2=1,则只有接收到有效停止位时,RI 才置 1,以便接收下一帧数据。在方式 0 时,SM2 必须是零。

(3)REN(SCON.4)

允许接收控制位。由软件置 1 或清 0,只有当 REN=1 时才允许接收,相当于串行接收的开关;若 REN=0,则禁止接收。

在串行通信接收控制程序中,如果满足 RI=0,置位 REN=1(允许接收)的条件,就会启动一次接收过程,一帧数据就装入 SBUF 中。

(4)TB8(SCON.3)

发送数据的第 9 位(D8)装入 TB8 中。在方式 2 或方式 3 中,根据发送数据的需要由软件置位或复位。在许多通信协议中可作奇偶校验位,也可在多机通信中作为发送地址帧或数据帧的标志。对于后者,TB8=1,说明发送该帧数据为地址;TB8=0,说明发送该帧数据为数据字节。在方式 0 和方式 1 中,该位未用。

(5)RB8(SCON.2)

接收数据的第 9 位。在方式 2 或方式 3 中,接收到的第 9 位数据放在 RB8 位。它或是约定的奇/偶校验位,或是约定的地址/数据标志位。在方式 2 和 3 多机通信中,若 SM2=1,如果 RB8=1,说明收到的数据为地址帧。

在方式 1 中,若 SM2=0(即不是多机通信情况),RB8 中存放的是已接收到的停止位。在方式 0 中,该位未用。

(6)TI(SCON.1)

发送中断标志。在一帧数据发送完时被置位。在方式 0 串行发送第 8 位结束时,或其它方式串行发送到停止位的开始时由硬件置位,可用软件查询。它同时也申请中断,TI 置位意味着向 CPU 提供"发送缓冲器 SBUF 已空"的信息,CPU 可以准备发送下一帧数据。串行口发送中断被响应后,TI 不会自动清 0,必须由软件清 0。

(7)RI(SCON.0)

接收中断标志。在接收到一帧有效数据后由硬件置位。在方式 0 中,第 8 位数据发送结束时,由硬件置位;在其它三种方式中,则在接收到停止位中间时由硬件置位。RI=1,申请中断,表示一帧数据接收结束,并已装入接收 SBUF 中,要求 CPU 响应中断,取走数据。RI 也必须由软件清 0,解除中断申请,并准备接收下一帧数据。

串行发送中断标志 TI 和接收中断标志 RI 是同一个中断源,CPU 事先不知道是发送中断 TI 还是接收中断 RI 产生的中断请求,所以在全双工通信时,必须由软件来判别。

复位时,SCON 所有位均清 0。

3. 串行波特率倍增位

电源控制寄存器 PCON 中有一位 SMOD 与串行口工作有关,如图 7-34 所示。

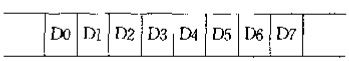
图7-34 PCON电源控制寄存器

SMOD(PCON.7)：波特率倍增位。在串行口方式1、方式2和方式3时，波特率与2^{SMOD}成正比，亦即当SMOD=1时，波特率提高一倍。复位时，SMOD=0。

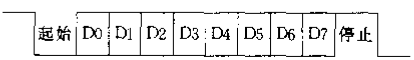
三、串行通信工作方式

根据实际需要，8051串行口可设置四种工作方式，可有8位、10位和11位帧格式。

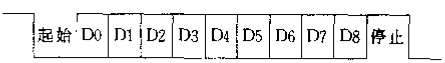
方式0以8位数据为一帧，不设起始位和停止位，先发送或接收最低位。其帧格式如下：



方式1以10位为一帧传输，设有1个起始位"0"，8个数据位和1个停止位"1"。其帧格式为：



方式2和3以11位为一帧传输，设有1个起始位"0"，8个数据位，一个附加第9位和1个停止位"1"。其帧格式为：



附加第9位(D8)由软件设置1或清0。发送时在TB8中，接收时送RB8中。

1. 串行口方式0

方式0为同步移位寄存器输入/输出方式，常用于扩展I/O口。串行数据通过RXD输入或输出，而TXD用于输出移位时钟，作为外接部件的同步信号，如图7-35(a)为发送电路及时序，图7-36为接收电路及时序。这种方式不适用于两个8051之间的数据通信，但可以通过外接移位寄存器来实现单片机的接口扩展。例如，采用74LS164可用于扩展并行输出口，74LS165可用于扩展输入口。在这种方式下，收/发的数据为8位，低位在前，无起始位、奇偶位及停止位，波特率固定为振荡频率f_{osc}的1/12，即

$$\text{方式0波特率}=f_{osc}/12$$

例如，当晶体振荡频率为12MHz时，则波特率为1Mbps。发送、接收时序如图7-35(b)和图7-36(b)所示。

发送过程中，当执行一个数据写入发送缓冲器SBUF(99H)的指令时，串行口把SBUF中8位数据以$f_{osc}/12$的波特率从RXD(P3.0)端输出，发送完毕置中断标志TI=1，方式0发送时序如图7-35(b)所示。写SBUF指令在S6P1处产生一个正脉冲，在下一个机器周期的S6P2处数据的最低位输出到RXD(P3.0)脚上，在再下一个机器周期的

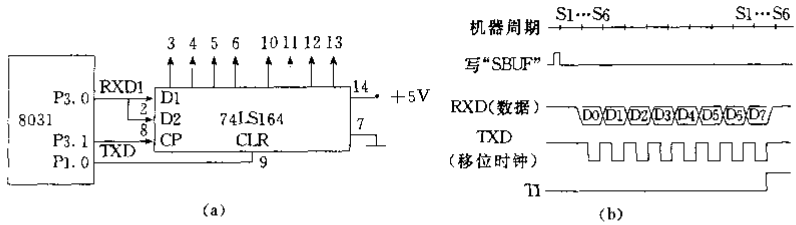


图7-35 方式0发送电路及时序

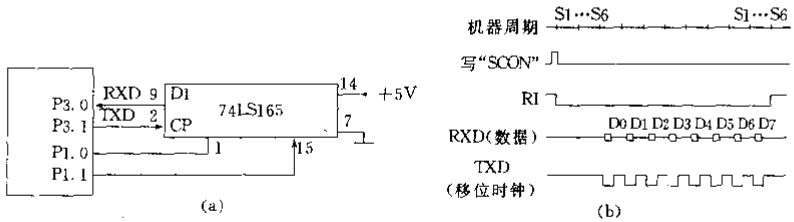


图7-36 方式0接收电路与时序

S3、S4、S5输出移位时钟为低电平，而在S6及下一个机器周期的S1、S2为高电平，就这样将8位数据由低位至高位一位一位顺序通过RXD线输出，并在TXD脚上输出$f_{OSC}/12$的位移时钟。在写"SBUF"有效后的第10个机器周期的S1P1将发送中断标志TI置位。图中74LS164是TTL"串入并出"移位寄存器。

接收时，用软件置REN=1(同时RI=0)，即开始接收。如图7-36所示。当使SCON中的REN=1(RI=0)时，产生一个正脉冲，在下一个机器周期的S3P1～S5P2从TXD(P3.1)脚上输出低电平的移位时钟，在此机器周期的S5P2对P3.0脚采样，并在本机器周期的S6P2通过串行口内的输入移位寄存器将采样值移位接收；在同一个机器周期的S6P1到下一个机器周期的S2P5，输出移位时钟为高电平。于是，将数据字节从低位至高位一位一位地接收下来并装入SBUF中，在启动接收过程(即写SCON，清RI位)将SCON中的RI清0之后的第10个机器周期的S1P1，RI被置位。这一帧数据接收完毕，可进行下一帧接收。图中74LS165是TTL"并入串出"移位寄存器。

2. 串行口方式1

方式1真正用于串行发送或接收，为10位通用异步接口。TXD与RXD分别用于发送与接收数据，收发一帧数据的格式为：1位起始位、8位数据位(低位在前)、1位停止位，共10位。在接收时，停止位进入SCON的RB8，此方式的传送波特率可调。串行口方式1的发送和接收时序如图7-37所示。

方式1发送时，数据从引脚TXD(P3.1)端输出，当执行数据写入发送缓冲器SBUF的命令时，就启动了发送器开始发送。发送时的定时信号，也就是发送移位时钟(TX时钟)，是由定时器T1送来的溢出信号经过16分频或32分频(取决SMOD的值)而取得的，TX时钟就是内部产生的发送波特率，可见方式1波特率是可变的。发送开始的同时，内部控制信号$\overline{SEND}$变为有效，将起始位向TXD输出，此后没过一个TX时钟周期(16/32分频计数器溢出一次为一个TX时钟周期，因此，TX时钟频率由波特率决定)产生一

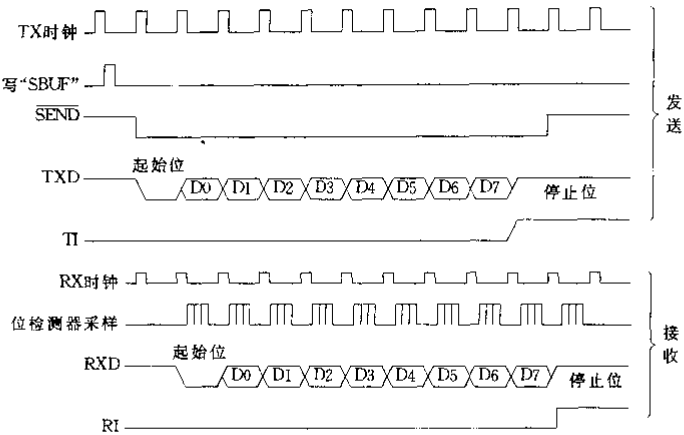


图7-37 方式1发送接收时序

个移位脉冲，并由 TXD 输出一个数据位，8 位数据位全部发送完后，置位 TI，并申请中断置 TXD 为 1 作为停止位，再经一个时钟周期$\overline{SEND}$失效。

方式 1 接收时，数据从引脚 RXD(P3.0)端输入。接收是在 SCON 寄存器中 REN 位置 1 的前提下，并检测到起始位(RXD 上检测到"1"→"0"的跳变，即起始位)而开始的。接收时，定时信号有两种：一种是接收移位时钟(RX 时钟)，它的频率和传送波特率相同，也是由定时器 T1 的溢出信号经过 16 或 32 分频而得到的；另一种是位检测器采样脉冲，它的频率是 RX 时钟的 16 倍，即在一位数据的期间有 16 位检测器采样脉冲，为完成检测，以 16 倍于波特率的速率对 RXD 进行采样。为了接收准确无误，在正式接收数据之前，还必须判定这个"1"→"0"跳变是否由干扰引起。为此，在这位中间(即 1 位时间分成 16 等份，在第 7、8 及 9 等份)连续对 RXD 采样三次，取其中两次相同的值进行判断所检测的值。这样能较好地消除干扰的影响。当确认是真正的起始位"0"后，就开始接收一帧数据。当一帧数据接收完毕后，必须同时满足以下两个条件，这次接收才真正有效。

(1)RI=0，即上一帧数据接收完成时，RI=1 发出的中断请求已被响应，SBUF 中数据已被取走。由软件使 RI=0，以便提供"接收 SBUF 已空"的信息。

(2)SM2=0 或收到的停止位为 1(方式 1 时停止位进入 RB8)，则将接收到的数据装入串行口的 SBUF 和 RB8(方式 1 RB8 装入停止位)，并置位 RI；如果不满足，接收到的数据不能装入 SBUF，这意味着该帧信息将会丢失。

值得注意的是：在整个接收过程中，保证 REN=1 是一个先决条件，只有当 REN=1，才能对 RXD 进行检测。

3. 串行口方式 2 和方式 3

串行口工作在方式 2 和方式 3 均为每帧 11 位异步通信格式，由 TXD 和 RXD 发送与接收(两种方式操作是完全一样的，所不同的只是波特率)。每帧 11 位：1 位起始位，8 位数据位(低位在前)，1 位可编程的第 9 数据位和 1 位停止位。发送时，第 9 数据位(TB8)可以

设置为 1 或 0，也可将奇偶位装入 TB8，从而进行奇偶校验；接收时，第 9 数据位进入 SCON 的 RB8。

方式 2 和方式 3 的发送、接收时序如图 7-38 所示。其操作与方式 1 类似。

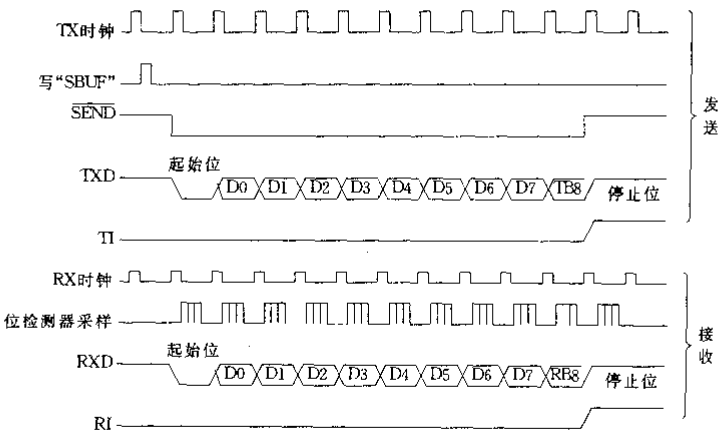


图7-38 方式2、3发送接收时序

发送前，先根据通信协议由软件设置 TB8（如作奇偶校验位或地址/数据标志位），然后将要发送的数据写入 SBUF，即能启动发送过程。串行口能自动把 TB8 取出，并装入到第 9 位数据的位置，再逐一发送出去。发送完毕，使 TI=1。

接收时，使 SCON 中的 REN=1，允许接收。当检测到 RXD(P3.0)端有"1"到"0"的跳变（起始位）时，开始接收 9 位数据，送入移位寄存器（9 位）。当满足 RI=0 且 SM2=0 或接收到的第 9 位为 1 时，前 8 位数据送入 SBUF，附加的第 9 位数据送入 SCON 中的 RB8，置 RI 为 1；否则，这次接收无效，也不置位 RI。

四、波特率设计

在串行通信中，收发双方或接收的数据速率要有一定的约定，我们通过软件对 8051 串行口编程可约定 4 种工作方式。其中，方式 0 和方式 2 的波特率是固定的，而方式 1 和方式 3 的波特率是可变的，由定时器 T1 的溢出率来决定。

串行口的 4 种工作方式对应着 3 种波特率。由于输入的移位时钟的来源不同，所以，各种方式的波特率计算公式也不同。

1. 方式 0 的波特率

方式 0 时，移位时钟脉冲由 S6（即第 6 个状态周期，第 12 个节拍）给出，即每个机器周期产生一个移位时钟，发送或接收一位数据。因此，波特率固定为振荡频率的 1/12，即

$$\text{方式 0 波特率} = f_{osc}/12$$

且不受 PCON 寄存器中 SMOD 位的影响。

2. 方式 2 的波特率

串行口方式 2 波特率的产生与方式 0 不同，即输入的时钟源不同。控制接收与发送的移位时钟由振荡频率 f_{OSC} 的第二节拍 P2 时钟（即 $f_{OSC}/2$）给出，所以，方式 2 波特率取决于 PCON 中 SMOD 位的值：当 SMOD=0 时，波特率为 f_{OSC} 的 1/64；若 SMOD=1，则波特率为 f_{OSC} 的 1/32，即

$$\text{方式 2 波特率}=(2^{SMOD}/64)\times f_{OSC}$$

3. 方式 1 和方式 3 的波特率

方式 1 和方式 3 的移位时钟脉冲由定时器 T1 的溢出速率决定。因此，8051 串行口方式 1 和方式 3 的波特率由定时器 T1 的溢出率与 SMOD 值同时决定。即

$$\text{方式 1、3 波特率}=\text{T1 溢出速率}/n$$

当 SMOD=0 时，n=32；当 SMOD=1 时，n=16。所以可用下式确定方式 1 和 3 的波特率：

$$\text{方式 1、3 波特率}=(2^{SMOD}/32)\times(\text{T1 的溢出速率})$$

其中，T1 溢出速率取决于 T1 的计数速率（计数速率=机器周期=$f_{OSC}/12$）和 T1 预置的初值。附录 Ⅵ 中列出了串行口方式 1、3 常用波特率及其初值。

定时器 T1 作波特率发生器使用时，通常选用定时器方式 2（自动重装初值定时器）比较实用。要设置定时器 T1 为定时器方式（使 $C/\overline{T}=0$），让 T1 计数内部振荡脉冲，即计数速率为 $f_{OSC}/12$（注意应禁止 T1 中断，以免溢出而产生不必要的中断）。先设定 TH1 和 TL1 定时计数初值为 X，那么每过“2^8-X”个机器周期，定时器 T1 就会产生一次溢出。因此 T1 溢出速率为

$$\text{T1 溢出速率}=(f_{OSC}/12)/(2^8-X)$$

$$\text{串行口方式 1、3 波特率}=\frac{2^{SMOD}}{32}\times\frac{f_{OSC}}{12\times(256-X)}$$

我们可得出定时器 T1 方式 2 的初始值

$$X=256-\frac{f_{OSC}\times(SMOD+1)}{384\times\text{波特率}}$$

例 8051 单片机时钟振荡频率为 11.0592MHz，选用定时器 T1 工作方式 2 作波特率发生器，波特率为 2400bps，求初值。

解 设置波特率控制位(SMOD)=0，则

$$X=256-\frac{11.0592\times10^6\times(0+1)}{384\times2400}=244=F4H$$

所以

$$(TH1)=(TL1)=F4H$$

系统晶体振荡频率选为 11.0592MHz 就是为了使初值为整数，从而产生精确的波特率。

如果串行通信选用很低的波特率，可将定时器 T1 置于方式 0 或方式 1，即 13 位或 16 位定时方式。但在这种情况下，T1 溢出时，需用中断服务程序重装初值。中断响应和执行命令时间会使波特率产生一定的误差，可用改变初值的办法加以调整。

五、串行口应用举例

设计一程序，将 8031(1)片内 RAM50H～5FH 中的数据串行发送到 8031(2)中，并存储于 8031(2)片内 RAM40H～4FH 单元中。

假设两单片机晶振均为 11.0592MHz，并且按图 7-39 已经连接完毕。

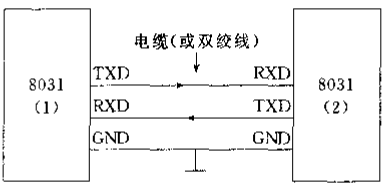


图7-39 点对点通信连接

根据题目要求，我们选择串行口方式 3 通信，接收/发送 11 位信息，开始为 1 位起始位(0)，中间 8 位数据位，数据位后为奇偶校验位，最后 1 位为停止位(1)。

奇偶校验的过程是这样实现的：在发送端，TB8 作奇偶校验位。在数据写入缓冲发送器之前，先将数据的奇偶位写入 TB8，作为第 9 位数据传送，这个奇偶性数据传送到接收一方 RB8 位上。在接收一方，接收到一个字符(8 位二进制信息与奇偶校验位)后，从 SBUF 转移到 A 中时，状态标志寄存器会产生已接收到数据的奇偶值，将此奇偶值与 RB8 中的奇偶值相比较，两者应该相符，否则接收字符有错。发现错误要通知对方重发。

如果选择波特率为 9600bps，且选择定时器 1 方式 2 定时，则发送与接收程序如下：

1. 单片机(1)发送程序：

```
TTTT:   MOV     TMOD,#20H  ;定时器 1 方式 2 定时
        MOV     TL1,#0FDH
        MOV     TH1,#0FDH  ;置定时器初值，选定 9600bps
        MOV     SCON,#0C0H ;选择通信方式 3
        SETB    TR1        ;启动定时器 1
        MOV     R0,#50H    ;首地址 50H→R0
        MOV     R7,#10H    ;数据长度 10H→R7
LOOP:   MOV     A,@R0      ;取数据→A
        MOV     C,PSW.0
        MOV     TB8,C      ;P→TB8
        MOV     SUBF,A     ;数据→SBUF 启动发送
WAIT:   JBC     TI,CONT    ;判断发送中断标志
        SJMP    WAIT
CONT:   INC     R0
        DJNZ    R7,LOOP
```

2. 单片机(2)接收程序

```
RRRR:   MOV     TMOD,#20H  ;定时器 1 方式 2 定时
```

```
        MOV     TL1,#0FDH
        MOV     TH1,#0FDH   ;置定时器初值,选定 9600bps
        MOV     SCON,#0D0H  ;选方式 3,允许接收(REN=1)
        SETB    TR1         ;启动定时器 1
        MOV     R0,#40H     ;首地址 40H→R0
        MOV     R7,#10H     ;数据长度 10H→R7
LOOP:   JBC     RI,RECE     ;等待接收
        SJMP    LOOP
RECE:   MOV     A,SBUF
        JB      PSW.0,ONEE  ;判断接收数据奇偶性
        JB      RB8,ERRR    ;判断发送端奇偶性
        SJMP    RIGT
ONEE:   JNB     RB8,ERRR
RIGT:   MOV     @R0,A       ;接收正确
        INC     R0
        DJNZ    R7,LOOP
ERRR:   ⋮                   ;接收错误
```

思考题与习题

1. 8051 单片机内设有几个可编程的定时器/计数器?它们可以有 4 种工作方式,如何选择和设定?

2. 简述 8051 单片机内部串行接口的 4 种工作方式。

3. 简述中断、中断源、中断优先级及中断嵌套的含义。

4. MCS-51 单片机能提供几个中断源? 几个中断优先级? 在同一优先级中各中断源优先顺序如何确定?

5. MCS-51 的外部中断有哪两种触发方式? 如何选择?

6. 简述 MCS-51 单片机中断响应过程。

7. 8051 单片机如何访问外部 ROM 及外部 RAM?

8. 试用 Intel 2764、6116 为 8031 单片机设计一个存储系统,它具有 8KEPROM(地址为 0000H～1FFFH)和 16K 的程序、数据兼用的 RAM 存储器(地址为 2000H～5FFFH)。具体要求:画出硬件连接图,并指出每片芯片的地址空间。

第八章　MCS-51 单片机接口技术

通过前面的学习我们知道，MCS-51 系列单片机共有四个 I/O 端口。当单片机外部扩展了程序存储器、数据存储器时，可专供外部输入/输出设备使用的只有 P1 口。如果单片机需要与多个外设进行数据传输，I/O 口的数量显然是不够的，这就需要对 I/O 口进行扩展。除此以外，外设的种类是多种多样的，如：键盘、显示器、打印机、A/D 和 D/A 转换器等，它们的工作速度相差很大，并且与单片机的运行速度也不同，因此，I/O 接口电路还应解决外设与单片机之间速度匹配问题。I/O 接口电路应具备的基本功能为：

(1)数据的缓冲和锁存功能，用于暂存数据；

(2)提供 I/O 接口电路的工作状态和控制信号，CPU 通过这些信号了解 I/O 接口电路的工作状态，控制 I/O 口读写过程。

MCS-51 单片机 I/O 口的扩展方法主要有：

(1)总线扩展方法。数据通过单片机的 P0 口输入/输出，为了区分不同的外设，每个 I/O 口都应有自己的地址。I/O 口的地址占用外部数据存储器的地址空间，因此在进行 I/O 口扩展时，应全面考虑 I/O 口和外部数据存储器的地址分配，避免地址重叠。

(2)串行扩展方法。利用 MCS-51 单片机串行口的同步移位寄存器工作方式进行 I/O 口扩展。由于数据是通过串行口传输的，因此传输速度较慢，并且必须进行串行—并行数据的转换。

Intel 公司生产了种类繁多的与计算机配套的外围接口芯片，如：8255A 和 8155 等，这些芯片大多比 MCS-51 单片机接口电路逻辑简单，因此，广泛应用于 51 系列单片机的接口电路中。下面将介绍几种常用的接口芯片及与单片机的接口电路。

第一节　MCS-51 单片机的并行接口电路

一、扩展并行接口芯片 8255A

8255A 为 Intel 公司生产的可编程输入/输出接口芯片。它具有 3 个 8 位 I/O 口，即 A 口、B 口、C 口和 3 种工作方式，并且 C 口具有按位操作功能，是一种功能很强的接口芯片。

1. 8255A 的引脚

8255A 具有 40 个引脚，采用双列直插封装形式，其引脚图如图 8-1 所示。各引脚的功能如下：

D7～D0：三态双向数据引脚，与单片机的数据总线相连，用于 CPU 与各 I/O 口之间的数据传输；

PA7～PA0：A 口输入/输出引脚；

PB7～PB0：B 口输入/输出引脚；

PC7～PC0：C 口输入/输出引脚；

$\overline{CS}$：片选信号线，低电平有效，表示芯片被选中；

$\overline{RD}$：读信号线，低电平有效，控制数据读出；

$\overline{WR}$：写信号线，低电平有效，控制数据写入；

RESET：复位信号线，高电平有效；

A1、A0：地址线，输入三个端口和控制寄存器的地址。

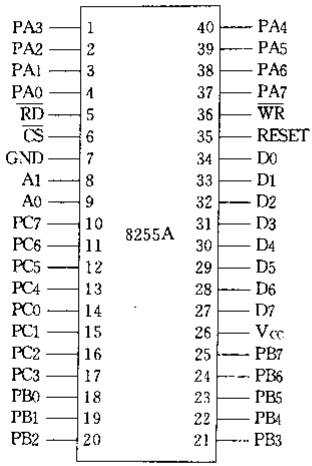


图8-1　8255A引脚图

2. 8255A 的内部结构

8255A 由 3 个数据输入/输出端口、二个工作方式控制电路、一个读/写控制逻辑电路和一个总线数据缓冲器组成，其内部结构如图 8-2 所示。

(1)端口 A、B、C

A 口：具有 8 位数据输出锁存/缓冲器和一个 8 位数据输入锁存器；

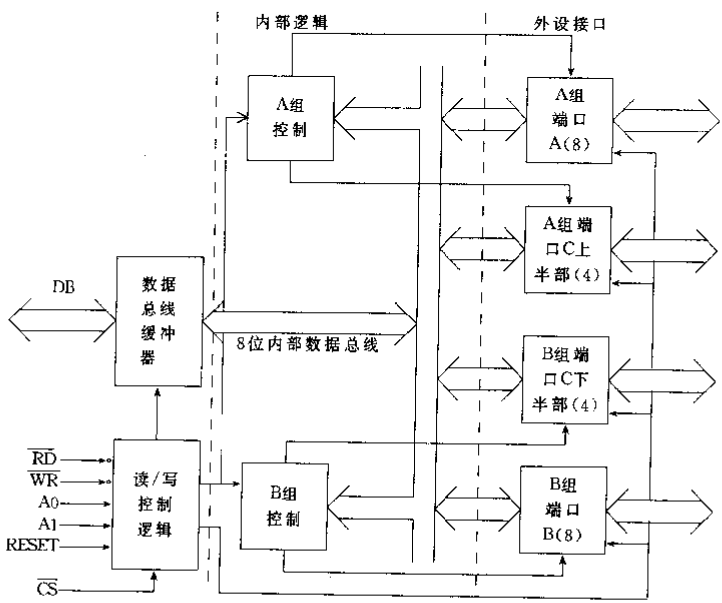


图8-2　8255的内部结构

B口：具有一个8位数据输入/输出锁存/缓冲器和一个8位数据输入缓冲器；

C口：具有一个8位数据输出锁存/缓冲器和一个8位数据输入缓冲器(不锁存)。

三个端口中A口和B口总是作为数据输入/输出端口，C口有时作为控制信号和状态信号的输入/输出端口。

(2)工作方式控制电路

工作方式控制电路有两个，即A组控制电路和B组控制电路，分别控制A口和C口上半部、B口和C口下半部。

工作方式控制电路根据控制字寄存器的内容控制A组和B组的工作方式，也可根据控制字寄存器的内容对C口按位进行操作。

(3)总线数据缓冲器

总线数据缓冲器是一个三态双向8位缓冲器。它的一端作为8255A与单片机的数据总线的接口，另一端与A口、B口、C口和控制字寄存器相连，作为单片机与I/O口和控制字寄存器之间的数据缓冲器。

(4)读/写控制逻辑电路

读/写控制逻辑电路输入的控制信号有$\overline{RD}$、$\overline{WR}$、RESET和A1、A0。它根据这些信号控制I/O口及控制寄存器的读/写操作。其中地址线A1、A0用来选择I/O口和控制字寄存器，与读/写控制信号$\overline{RD}$和$\overline{WR}$构成各种工作状态，如表8-1所示。

表8-1 8255A的I/O口工作状态

A1	A0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	工作状态
0	0	0	1	0	A口数据→数据总线
0	1	0	1	0	B口数据→数据总线
1	0	0	1	0	C口数据→数据总线
0	0	1	0	0	总线数据→A口
0	1	1	0	0	总线数据→B口
1	0	1	0	0	总线数据→C口
1	1	1	0	0	总线数据→控制字寄存器
×	×	×	×	1	数据总线→三态
1	1	0	1	0	非法状态
×	×	1	1	0	数据总线→三态

3. 8255A的工作方式

8255A有3种工作方式，即工作方式0、工作方式1和工作方式2。

(1)工作方式0

工作方式0为基本输入/输出方式，其功能概括如下：

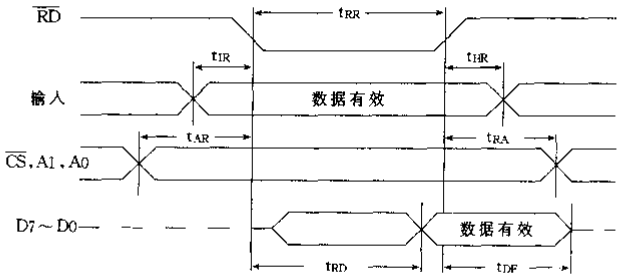
①具有二个8位端口(A口、B口)和二个4位端口(C口的上、下半部分)；

②任意一个端口都可以设定为输入或输出，各端口的输入/输出状态可构成16种组合；

③数据输出可锁存，输入不锁存。

在工作方式0状态下，A口、B口和C口都作为I/O端口，没有设置控制/状态信号。单片机可以通过访问外部数据存储器指令，对任一端口进行读/写操作。

图8-3和图8-4为方式0输入和输出的时序。



符　号	参　　数	8255A		单　位
		MIN	MAX	
t_{RR}	读脉冲宽度	300		ns
t_{IR}	输入领先于$\overline{RD}$的时间	0		ns
t_{HR}	输入滞后$\overline{RD}$的时间	0		ns
t_{AR}	地址稳定领先读信号的时间	0		ns
t_{RA}	读信号无效后地址保持时间	0		ns
t_{RD}	从读信号有效到数据稳定		250	ns
t_{DF}	读信号去除后到数据浮空	10	150	ns
t_{RY}	在两次读(或写)之间的时间间隔	850		ns

图8-3　方式0输入时序

(2)工作方式1

工作方式1为选通工作方式。其功能概括如下：

①三个端口分为两组——A组和B组，A组由A口和C口上半部分组成，B组由B口和C口下半部分组成；

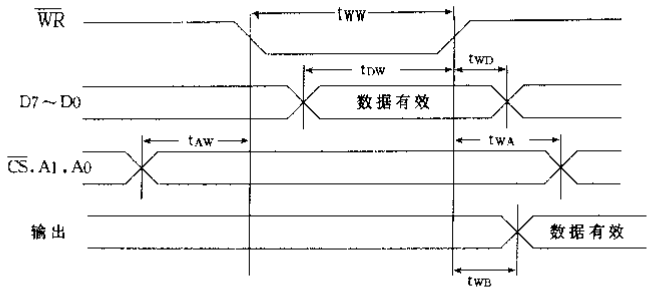
②每组包括一个8位数据端口和一个4位控制/状态端口；

③每个8位数据端口均可设置为输入或者输出，输入、输出均可锁存；

④C口没有用作控制/状态信号的位仍可作为I/O口。

图8-5为A组和B组方式1输入时的控制/状态信号的示意图，图8-6为方式1输出时的控制/状态信号示意图。

下面简单介绍一下控制/状态信号的功能：



符　号	参　　数	8255A		单　位
		MIN	MAX	
t_{AW}	地址稳定领先写信号的时间	0		ns
t_{WA}	写信号后地址保持时间	20		ns
t_{WW}	写脉冲宽度	400		ns
t_{DW}	写脉冲结束之前数据有效时间	100		ns
t_{WD}	数据保持时间	30		ns
t_{WB}	从写信号结束到输出		350	ns

图8-4　方式0输出时序

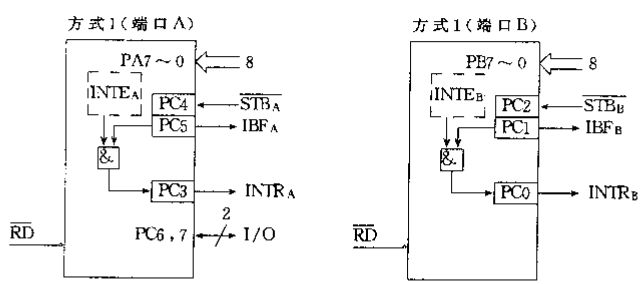


图8-5　方式1输入控制/状态信号

①方式1输入

$\overline{STB}$：选通输入控制信号，低电平有效。该信号由外设提供，用来将外设送来的输入数据送入输入锁存器。

IBF：输入缓冲器满信号，高电平有效。它是由8255A输出的状态信号，其有效时，表

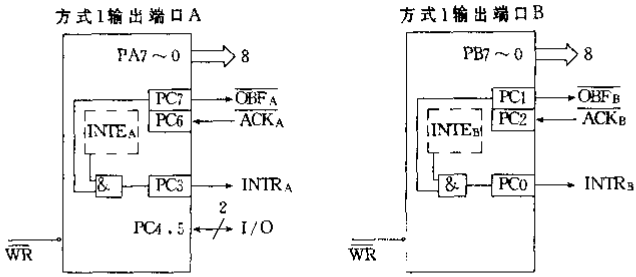
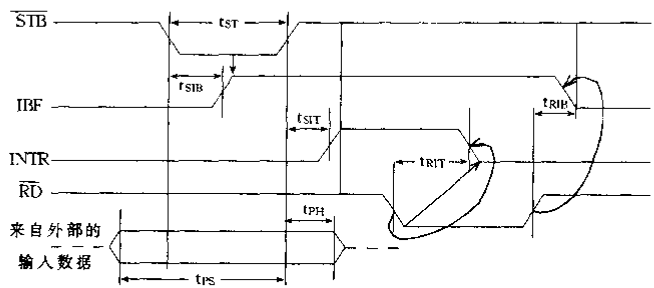


图8-6　方式1输出控制/状态信号

示数据已输入至输入锁存器。

INTR：中断请求信号，高电平有效。它由8255A输出，用于向CPU申请中断。

INTE：中断允许控制位，该位为1时，允许中断请求。$INTE_A$由PC4来置位/复位，$INTE_B$由PC2来置位/复位。



符　号	参　　数	8255A		单　位
		MIN	MAX	
t_{ST}	STB脉冲宽度	500		ns
t_{SIB}	STB=0到IBF=1		300	ns
t_{SIT}	STB=1到INTR=1		300	ns
t_{RIB}	RD=1到IBF=0		300	ns
t_{RIT}	RD=0到INTR=0		400	ns
t_{PS}	数据提前STB无效时间	0		ns
t_{PH}	数据保持时间	180		ns

图8-7　方式1输入时序

方式1输入的时序如图8-7所示。从图中可以看到$\overline{STB}$有效信号使IBF置位，表示数据已输入到输入锁存器。$\overline{STB}$、IBF和INTE为高电平时，INTR置位，向CPU申请中断。$\overline{RD}$的下降沿将INTR复位，上升沿将IBF复位。

②方式1输出

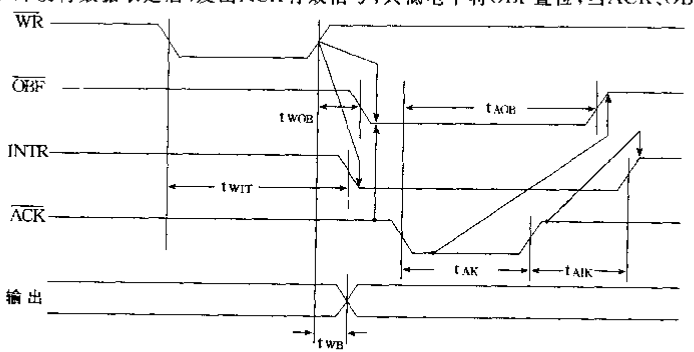
$\overline{OBF}$：输出缓冲器满信号，低电平有效。该信号为8255A发出的状态信号，表示CPU将数据输出到指定的端口。

$\overline{ACK}$：外设响应信号，低电平有效。该信号由外设提供，表示输出到8255A端口的数据已被取走。

INTR：中断请求信号，高电平有效。它由8255A发出，用于向CPU申请中断。

INTE：中断允许控制位，该位为1时，允许中断请求。$INTE_A$由PC6控制置位/复位，$INTE_B$由PC2控制置位/复位。

方式1输出时序如图8-8所示。数据输出过程从CPU向端口输出数据开始，$\overline{WR}$有效信号的上升沿将$\overline{OBF}$信号清0，表示数据已输出到了输出锁存器，同时INTR信号也被清除。外设将数据取走后，发出$\overline{ACK}$有效信号，其低电平将$\overline{OBF}$置位，当$\overline{ACK}$、$\overline{OBF}$和INTE



符　号	参　　数	8255A		单　位
		MIN	MAX	
t_{WOB}	WR＝1到OBF＝0		650	ns
t_{WIB}	MR＝0到INTR＝0		850	ns
t_{AOB}	ACK＝0到OBF＝1		350	ns
t_{AK}	ACK脉冲宽度	300		ns
t_{AIT}	ACK＝1到INTR＝1		350	ns
t_{WB}	WR＝1到输出		350	ns

图8-8　方式1输出时序

为高电平时，INTR 置位，向 CPU 发出中断请求。

(3)工作方式 2

工作方式 2 为双向数据传送方式。其功能概括如下：

①有一个 8 位双向数据端口(A)和一个 5 位控制/状态信号端口(C)；

②输入、输出均锁存；

③工作方式仅适用于 A 口；

④C 口没有用作控制/状态信号的位仍可用作 I/O 口。图 8-9 为方式 2 控制/状态信号的示意图。

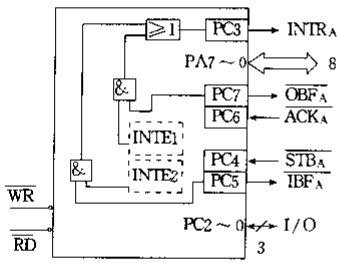


图 8-9　方式 2 控制/状态信号

图中各信号的功能如下：

INTR：中断请求信号，高电平有效。输入、输出时都由它来请求中断。

$\overline{OBF}$：输出缓冲器满信号，低电平有效。

$\overline{ACK}$：外设响应信号，低电平有效。该信号由外设提供，通常用来启动 A 口三态输出缓冲器输出数据。

INTE1：8255A 内部与输出缓冲器有关的中断允许触发器，输出为 1 时，允许输出中断请求。由 PC6 控制置位/复位。

$\overline{STB}$：选通输入控制信号，低电平有效。

IBF：输入缓冲满信号，高电平有效。

INTE2：8255A 内部与输入缓冲器有关的中断允许触发器，输出为 1 时，允许输入中断请求。由 PC4 控制置位/复位。

方式 2 的时序如图 8-10 所示。方式 2 是方式 1 输入、输出的结合，时序图中的时间参

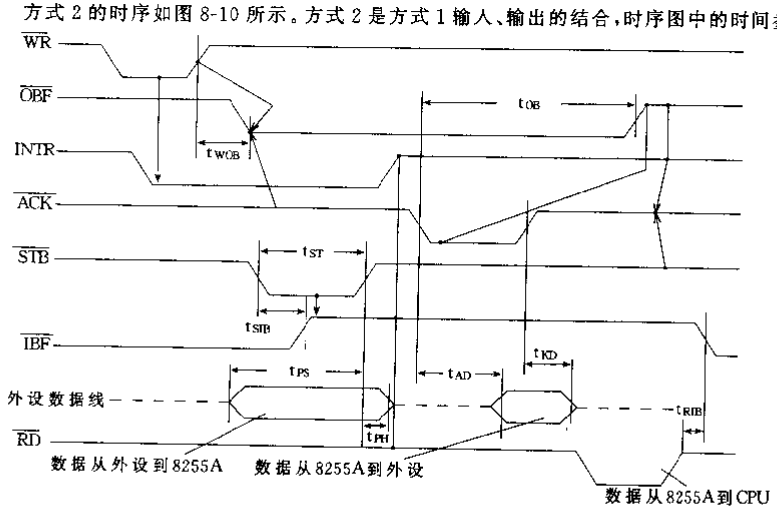


图 8-10　方式 2 的时序

数与方式 1 相同，这里不再赘述。

工作方式是由 8255A 内的工作方式控制字寄存器的内容决定的。单片机可以通过编程来改变其内容。方式控制字寄存器的格式如图 8-11 所示。

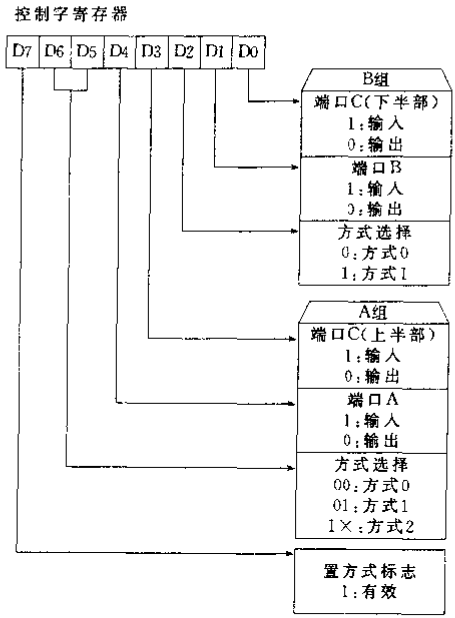


图8-11　8255A的控制字

例如：将 A 口、B 口和 C 口设置成基本输入输出方式，A 口为输入，B 口和 C 口为输出，则控制字寄存器的内容为 90H。

C 口具有位操作功能，通过控制位操作控制字将其某一位置位或复位，工作方式控制字与按位操作控制字的地址相同，如图8-12所示。

例如：若将 PC7 置位，向工作方式控制字写入 0FH 即可。

4. 8031 与 8255A 的接口

8031 和 8255A 的接口电路同单片机 CPU 与 I/O 口之间的数据传送方式有关，传送方式通常可分为无条件传送方式、查询传送方式和中断传送方式。在无条件传送方式中，8255A 与单片机之间无需状态/控制信号相连，CPU 可以随时对 I/O 进行访问。后两种传送方式中 8255A 与单片机之间要有状态信号或中断请求信号线相连。图 8-13 是 8031 扩展一片 8255A 的接口电路。在 CPU 访问 I/O 口之前，需要先设置 8255A 的工作方式和各 I/O 口的输入/输出状态，即对 8255A 初始化。图 8-13 中，8255A 的各 I/O 口和控制字寄存器的地址为：

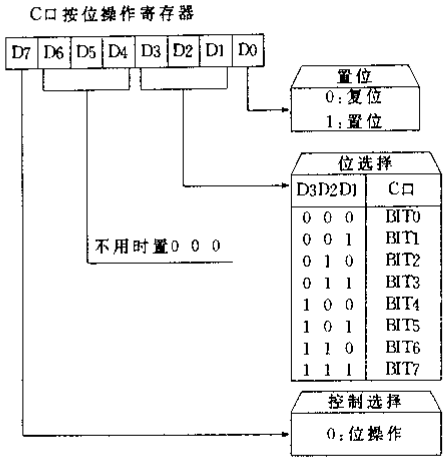


图8-12　C口按位操作控制字

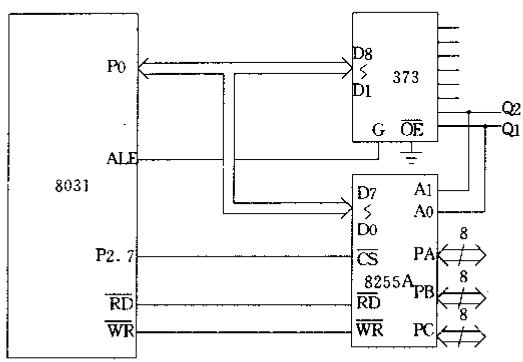


图8-13　8031与8255A的接口电路

A 口：7FFCH

B 口：7FFDH

C 口：7FFEH

控制字寄存器：7FFFH

设 A 口、B 口和 C 口为基本输入输出工作方式，A 口为输入，B 口和 C 口为输出，初始化程序为：

```
MOV    DPTR,#7FFFH  ;置控制字的地址
MOV    A,#90H       ;置控制字的内容
MOVX   @DPTR,A
```

图 8-13 中，8255A 中 8031 之间没有状态信号线或中断请求信号线相连，因此，只能选择 8255A 的工作方式 0。这种 CPU 与 I/O 口之间的数据为无条件传送方式，只相当于单片机系统扩展了 I/O 口的数量。图 8-14 是 8031 采用中断方式从 8255A 的 A 口输入数据的接口电路。

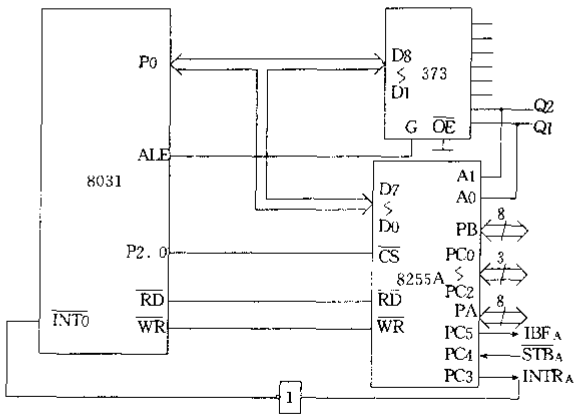


图 8-14　8031 与 8255A 中断方式接口电路

设 8255A 的 A 口为工作方式 1 输入，B 口和 C 口下半部分为工作方式 0 输出，数据存入以 R0 为地址寄存器的单元中。初始化程序和中断服务程序如下：

初始化程序：

```
    MOV     DPTR,#0FEFFH    ;置控制字地址
    MOV     A,#0D0H         ;置控制字内容
    MOVX    @DPTR,A         ;A 口方式 1 输入,B 口、C 口下半部方式 0 输入
    MOV     A,#09H          ;置 C 口位操作控制内容
    MOVX    @DPTR,A         ;允许 A 口中断请求
    SETB    EA
    SETB    EX0
    SETB    IT0
     ⋮
```

中断服务程序：

```
AI: PUSH    A
    MOV     DPTR,#0FEFCH    ;指向 A 口
    MOVX    A,@DPTR
    MOV     @R0,A
    INC     R0
    POP     A
    RETI
```

二、扩展并行接口芯片 8155

8155 为 Intel 公司的另一种可编程并行 I/O 接口芯片。它具有二个 8 位和一个 6 位 I/O 口，以及 256 个字节 RAM、一个 14 位计数器。它与单片机的接口简单，在单片机系统中应用广泛。

1. 8155 的引脚

8155 为 40 引脚芯片，采用双列直插封装形式，其引脚如图 8-15 所示。

各引脚功能如下：

AD7～AD0：三态地址/数据引线；

PA7～PA0：A 口输入/输出引线；

PB7～PB0：B 口输入/输出引线；

PC5～PC0：C 口输入/输出引线或控制信号线。

当 C 口作为控制信号线时，功能如下：

PC0：$INTR_A$——A 口中断申请信号线；

PC1：BF_A——A 口缓冲器满信号线；

PC2：$\overline{STB}_A$——A 口选通信号线；

PC3：$INTR_B$——B 口中断申请信号线；

PC4：BF_B——B 口缓冲器满信号线；

PC5：$\overline{STB}_B$——B 口选通信号线；

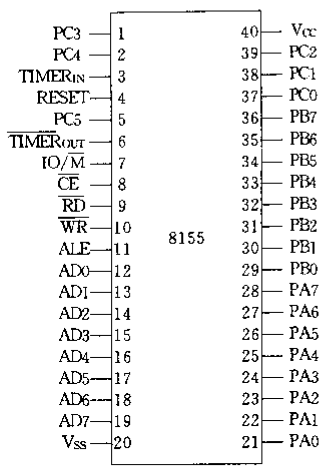


图 8-15　8155 的引脚

$\overline{CE}$：片选信号线，低电平有效；

$\overline{RD}$：存储器读控制信号线，低电平有效；

$\overline{WR}$：存储器写控制信号线，低电平有效；

ALE：地址及片选信号锁存信号线，高电平有效；

IO/$\overline{M}$：I/O 口与存储器选择信号线，IO/$\overline{M}$=1 时，选择 I/O 口，IO/$\overline{M}$=0 时，选择存储器；

$\overline{TIMER}_{IN}$：定时/计数器输入端；

$\overline{TIMER}_{OUT}$：定时/计数器输出端；

RESET：复位信号线，高电平有效。

2. 8155 的内部结构

8155 的内部结构如图 8-16 所示，它包括二个 8 位并行输入/输出端口，一个 6 位并行输入/输出端口，256 个字节的静态 RAM，一个地址锁存器，一个 14 位的定时/计数器和控制逻辑电路。

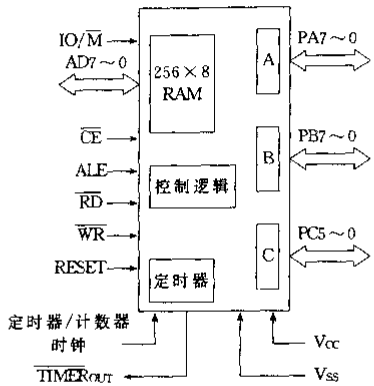


图8-16　8155的结构框图

在控制信号中，IO/$\overline{M}$ 为 IO 口和存储器选择信号，当 IO/$\overline{M}$=1 时，CPU 选择对 I/O 口和 8155 片内的寄存器进行读/写操作；当 IO/$\overline{M}$=0 时，CPU 选择对存储器进行读/写操作。256 个字节的存储器的地址范围为 00H～FFH，I/O 口和寄存器的地址如表 8-2 所示。

表 8-2　8155 I/O 的地址

AD7～AD0								I/O 口与寄存器
A7	A6	A5	A4	A3	A2	A1	A0	
×	×	×	×	×	0	0	0	命令/状态寄存器
×	×	×	×	×	0	0	1	A 口
×	×	×	×	×	0	1	0	B 口
×	×	×	×	×	0	1	1	C 口
×	×	×	×	×	1	0	0	定时器的低 8 位
×	×	×	×	×	1	0	1	定时器的高 6 位与 2 位计数器方式位

3. 8155 的工作方式

8155 内的控制逻辑电路中，设置了一个命令/状态寄存器，该寄存器分为两部分：一个是控制命令寄存器，它只能写入，不能读出，用于选择 I/O 口的工作方式，其格式如图 8-17 所示；另一个是状态标志寄存器，它只能读出，不能写入，用于存放 A 口和 B 口的工作状态，其格式如图 8-18 所示。

从图 8-17 中我们可以看到，8155 I/O 口的工作方式可以分为基本输入输出工作方式和选通工作方式。这两种工作方式与 8255A 的工作方式 0 和工作方式 1 具有相同的时序。与 8255A 不同的是，在选通工作方式时，8155 的输入/输出共用一组控制/状态信号。

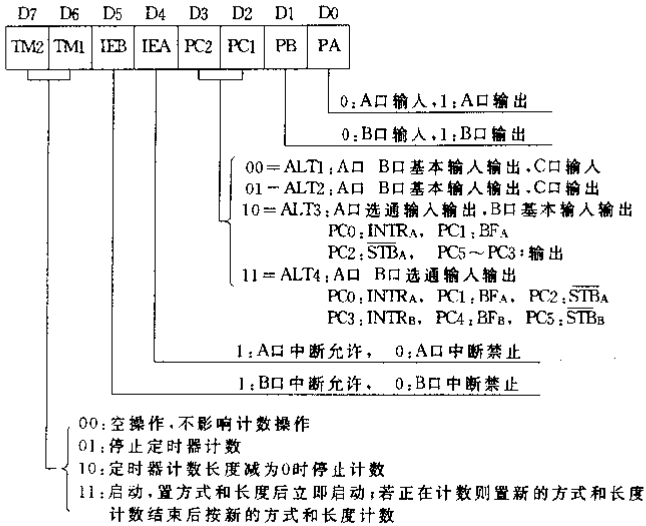


图8-17 8155命令寄存器格式

D7	D6	D5	D4	D3	D2	D1	D0
×	TIMER	$INTE_B$	BF_B	$INTR_B$	$INTE_A$	BF_A	$INTR_A$

INTR:中断请求 INTE:端口中断允许
BF:缓冲器满标志 TIMER:定时中断

图8-18 8155状态标志寄存器

选通输入时,$\overline{STB}$为选通输入信号,BF为输入缓冲器满信号;选通输出时,$\overline{STB}$为外设响应信号,BF为输出缓冲器满信号。

4. 8155的定时器

8155片内设置了一个14位的减法计数器,用于对外部输入的脉冲信号进行计数或定时。脉冲信号由$TIMER_{IN}$引脚输入,定时器的输出引脚为$\overline{TIMER}_{OUT}$。8155定时器的格式如图8-19所示,其中T13~T0为计数器的长度,其范围为2H~3FFFH,M2、M1用于设置定时器的输出方式,如图8-20所示。

5. 8031与8155的接口电路

由于8155内部设有地址锁存器,因此,它与8031的接口电路非常简单,不需任何附加电路。图8-21是8031与8155的一种接口电路,其RAM和I/O口的地址分配如下:

数据存贮器的地址:7E00 H~7EFFH

I/O的地址:

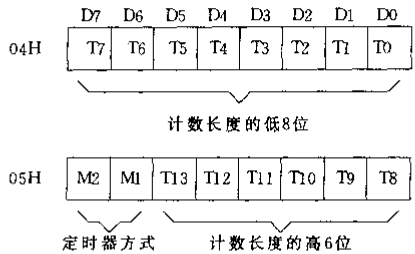


图8-19 8155定时器的格式

M1	M2	方式	定时器输出波形
0	0	单方波	
0	1	连续方波	
1	0	单脉冲	
1	1	连续脉冲	

图8-20 8155定时器输出方式

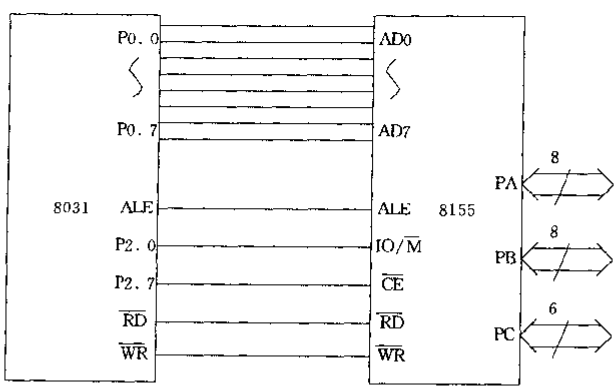


图8-21 8031与8155接口电路

命令/状态寄存器 7FF8H
PA口 7FF9H
PB口 7FFAH

PC 口	7FFBH
定时器低 8 位	7FFCH
定时器高 8 位	7FFDH

设 8155 的 A 口、B 口为基本输入输出方式，A 口为输入，B 口为输出，定时器输出连续方波，输入脉冲 24 分频，则 8155 的初始化程序为：

```
MOV     DPTR,#7FFCH
MOV     A,#18H
MOVX    @DPTR,A
INC     DPTR
MOV     A,#40H          ;定时器为连续方波输出
MOVX    @DPTR,A
MOV     DPTR,#7FF8H
MOV     A,#0C2H         ;置命令控制字:A 口基本输入,B 口基本
MOVX    @DPTR,A         ;输出,启动定时器
```

第二节　键盘与数码管显示器接口电路

键盘和显示器是计算机常用的输入输出设备，用于输入数据和命令，显示计算机的运行状态、命令和计算结果。考虑到简化结构，降低成本，单片机系统中经常采用简单键盘和数码管显示器，本节介绍它们与单片机的接口电路。

一、键盘接口电路

键盘由一组常开的按键开关组成，每个按键都被赋予一个代码，称为键码。键盘系统的主要工作包括及时发现有键闭合，求闭合键的键码。根据这一过程的不同，键盘可以分为二种，即编码键盘和非编码键盘。编码键盘是通过一个编码电路来识别闭合键的键码，非编码键盘是通过软件来识别键码。由于非编码键盘的硬件电路简单，用户可以方便地增减键的数量，因此在单片机系统中应用广泛。这里着重介绍非编码键盘的接口电路。

1. 按键电路和消除抖动

键盘中按键的开关状态，通过一定的电路转换为高、低电平状态，如图 8-22 所示。按键闭合过程在相应的 I/O 端口形成一个负脉冲，如图 8-23 所示。闭合和释放过程都要经过一定的过程才能达到稳定，这一过程是处于高、低电平之间的一种不稳定状态，称为抖动。抖动持续时间的长短与开关的机械特性有关，一般在 5～10ms 之间。为了避免 CPU 多次处理按键的一次闭合，应采取措施消除抖动。

消除抖动的方法有两种，一种是采用硬件电路来实现，如用滤波电路、双稳态电路等。另一种是利用软件来实现，即当发现有键按下时，延时 10～20ms 再查询是否有键按下，若没有键按下，说明上次查询结果为干扰或抖动；若仍有键按下，则说明闭合键已稳定，即可判断其键码。

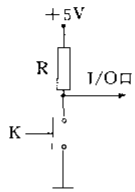


图8-22　按键电路

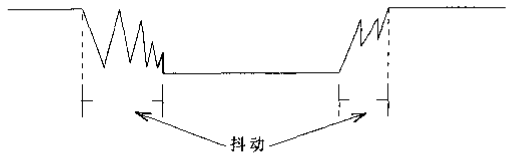


图8-23　键抖动的波形

2、非编码键盘的结构

非编码键盘可以分为二种结构形式：独立式按键和行列式键盘。

(1)独立式按键

独立式按键是指直接用I/O口线构成单个按键电路，每个按键占用一条I/O口线，每个按键的工作状态不会产生相互影响。图8-24是一种独立式按键电路，当图中的某一个键闭合时，相应的I/O口线变为低电平。当程序查询到为低电平的I/O口线时，就可以确定处于闭合状态的键。

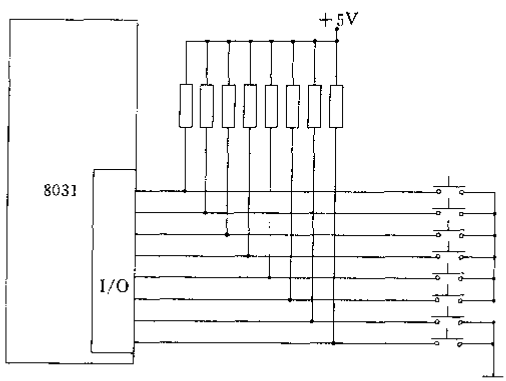


图8-24　独立式按键电路

独立式按键电路的结构和处理程序简单，扩展方便，但其占用的I/O口线相对较多，不适合在按键数量较多的场合下采用。

(2)行列式键盘

将I/O口线的一部分作为行线，另一部分作为列线，按键设置在行线和列线的交叉点上，这就构成了行列式键盘。行列式键盘中按键的数量可达行线数n乘以列线数m，如4行、4列行列式键盘的按键数可以达到4×4=16个。由此可以看到行列式键盘在按键较多时，可以节省I/O口线。图8-25为4×4行列式键盘的电路原理图。8条I/O口线分为4条行线和4条列线，按键设置在行线和列线交点上，即按键开关的两端分别接在行线和列线上。行线通过一个电阻接到+5V电源上，在没有键按下时，行线处于高电平状态。

判断是否有键按下的方法是：向所有的列线I/O口输出低电平，然后将行线的电平

状态读入累加器 A 中，若无键按下，行线仍保持高电平状态，若有键按下，行线至少应有一条为低电平。

当确定有键按下后，即可进行求键码的过程。其方法是：依次从一条列线上输出低电平，然后检查各行线的状态，若全为高电平，说明闭合键不在该列（输出低电平），若不全为 1，则说明闭合键在该列，且在变为低电平的行的交点上。

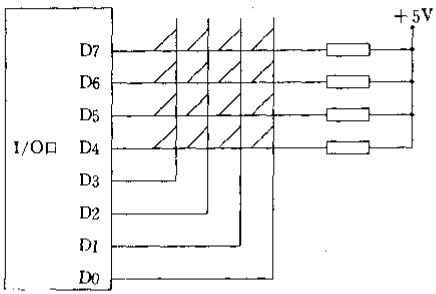


图8-25　行列式键盘的电路原理

在键盘处理程序中，每个键都被赋予了一个键号，由从列线 I/O 口输出的数据和从行线 I/O 口读入的数据可以求出闭合键的键号。

3. 非编码键盘的工作方式

在单片机应用系统中，非编码键盘由 CPU 通过键盘处理程序完成整个工作过程。相对 CPU 来言，按键闭合是随机发生的，键盘处理程序必须能够及时捕捉到闭合的键，并求出其键码。按照这一过程的不同，非编码键盘的工作方式可分为程序扫描方式和中断扫描方式。

(1)程序扫描方式

一般情况下，在单片机应用系统中，键盘处理只是CUP 工作的一部分。为了能及时发现有键按下，CPU 必须不断调用键盘处理程序，对键盘进行扫描，因此称为程序扫描方式。

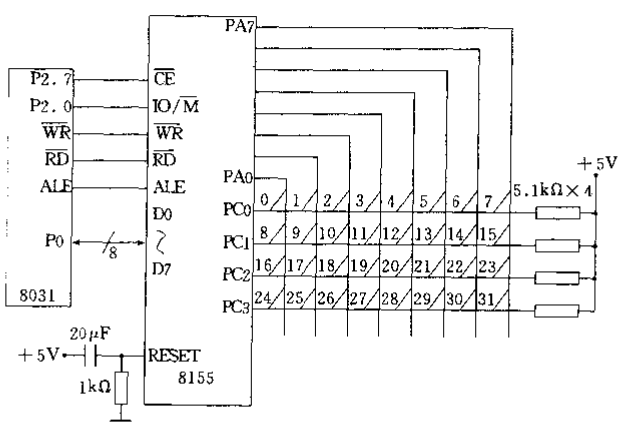


图8-26　8031与非编码键盘的接口电路

图 8-26 是一种非编码键盘与 8031 的接口电路。通过 8155 扩展的I/O 口作为行线和

列线，构成具有32键的4×8的行列式键盘。行线与8155的PC0～PC3相连，列线与PA口的8条线相连，键码如图8-26所示。

键盘处理程序的功能包括：

①判断键盘中有无键按下。由PA口输出00H，再将PC口的状态读入，若PC0～PC3全为1，说明无键按下，若不全为1则有键按下。

②消除抖动。当发现有键按下时，延时一段时间后再判断键盘的状态，若仍有键保持按下状态，则可断定有键按下，否则认为是抖动。

③求键号。从PA口依次输出下列扫描信号：

	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
第一次	1	1	1	1	1	1	1	0
第二次	1	1	1	1	1	1	0	1
第三次	1	1	1	1	1	0	1	1
第四次	1	1	1	1	0	1	1	1
第五次	1	1	1	0	1	1	1	1
第六次	1	1	0	1	1	1	1	1
第七次	1	0	1	1	1	1	1	1
第八次	0	1	1	1	1	1	1	1

每次输入扫描信号后，检查PC口的状态，若某一位为0，说明闭合的键在该行，即可确定闭合键的行和列。例如：当由PA口输出11111011，从PC口读入的状态为1101，说明闭合键位于一行二列。

④等待闭合键的释放。为了避免一次闭合多次求其键码，等待闭合键释放后再将键号送入A。

根据以上功能，键盘处理程序的框图如图8-27所示。

键盘处理程序如下：其中KS1为判断有无键闭合子程序，TIM为延时6ms子程序。

```
KEY1:  ACALL  KS1              ;判断有无键闭合
       JNZ    LK1              ;(A)≠0,转去抖动
       ACALL  TIM              ;延时6ms
LK1:   AJMP   KEY1             ;无键闭合返回
LK1:   ACALL  TIM
       ACALL  TIM              ;延时6ms
       ACALL  KS1              ;判断有无键闭合
       JNZ    LK2              ;(A)≠0,有键闭合,转求键码
       ACALL  TIM              ;延时6ms
       AJMP   KEY1             ;非键闭合,返回
LK2:   MOV    R2,#0FEH         ;R2←第一次扫描输出信号
       MOV    R4,#00H          ;R4←列号
LK3:   MOV    DPTR,#7F01H      ;DPTR←A口
       MOV    A,R2
       MOVX   @DPTR,A          ;输出扫描信号
```

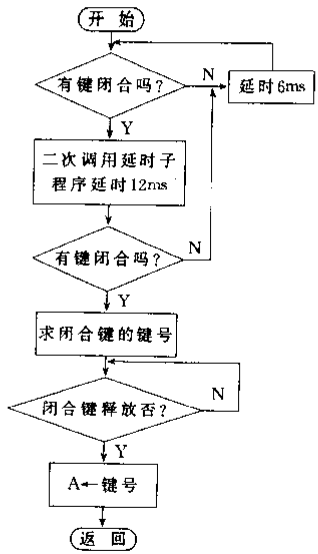


图8-27 键盘处理程序框图

```
        INC     DPTR
        INC     DPTR            ;指向C口
        MOVX    A,@DPTR         ;读C口状态
        JB      ACC.0,L1        ;PC0=1,转移
        MOV     A,#00H          ;A←0行0列键号,准备求键号
        AJMP    LK
L1:     JB      ACC.1,L2        ;PC1=1转移
        MOV     A,#08H          ;A←1行0列键号
        AJMP    LK
L2:     JB      ACC.2,L3        ;PC2=1,转移
        MOV     A,#10H          ;A←2行0列键号
        AJMP    LK
L3:     JB      ACC.3,NEXT      ;PC3=1,转移
        MOV     A,#18H          ;A←3行0列键号
LK:     ADD     A,R1            ;形成键码
        PUSH    ACC             ;暂存键码
LK4:    ACALL   TIM
        ACALL   KS1
        JNZ     LK4             ;等待键释放
```

```
        POP     ACC             ;键号送回 A
NS:     RET
NEXT:   INC     R4              ;列号加 1
        MOV     A,R2
        JNB     ACC.7,NS        ;8 列扫描完毕,返回主程序
        RL      A               ;形成下次扫描输出信号
        MOV     R2,A
        AJMP    LK3
KS1:    MOV     DPTR,#7F01H
        MOV     A,#00H
        MOVX    @DPTR,A         ;输出扫描信号 00H
        INC     DPTR
        INC     DPTR
        MOVX    A,@DPTR         ;读入 C 口状态
        CPL     A               ;求返
        ANL     A,#0FH          ;屏敞高 4 位
        RET
```

(2)中断扫描方式

在程序扫描工作方式中,为了能及时响应键盘输入,需要不停地对键盘进行扫描,即使没有键操作时,也不能中断。这就浪费了大量 CPU 保贵的时间。为了提高 CPU 的效率,在电路中增加适当的电路,当有键闭合时,产生中断请求信号。在中断服务子程序中进行去抖动、求键码和处理重键等工作。

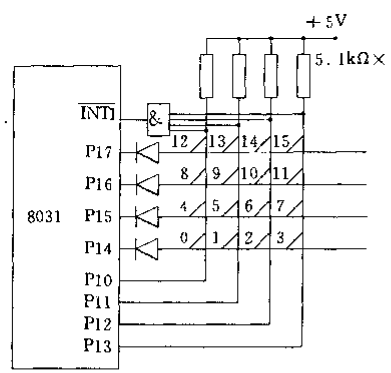


图8-28 中断扫描方式键盘接口电路

图 8-28 所示为中断扫描方式的接口电路图。行列式键盘与 8031 单片机的 P1 口直接相连,其中 P1.7～P1.4 经二极管与行线连接,P1.3～P1.0 与列线连接,另一端经电阻与+5V 电源相连。列线与一个与门的输入端相连,与门输出端接 8031 的 $\overline{INT0}$。当 P1.7～P1.4 全为 0 状态时,若无键闭合,$\overline{INT0}$保持高电平,若有键闭合时,$\overline{INT0}$变为低电平,CPU 开中断时,就会响应中断,转向中断服务程序。由于键盘处理过程与程序扫描方式大致相同,这里不再详细介绍。

二、数码管显示器接口电路

数码管显示器由于其成本低，配置灵活，与单片机接口简单，广泛应用于单片机应用系统中。下面介绍其工作原理及与单片机的接口电路。

1. 数码管的工作原理

数码管是由8个发光二级管构成的显示器件，其外形如图8-29(a)所示。a～g和h为8个发光二极管。在数码管中，若将二级管的阳极连在一起，称为共阳极数码管；若将二极管的阴极连在一起，称为共阴极数码管，如图8-29(b)。当发光二极管导通时，它就会发

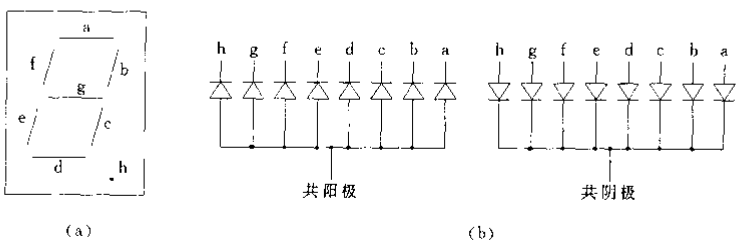


图8-29 数码管的结构

光。每个二极管就是一个笔划，若干个二极管发光时，就构成了一个显示字符。若将单片机的I/O口与数码管的a～g及h相连，高电平的位对应的发光二极管亮，这样，由I/O口输出不同的代码，就可以控制数码管显示不同的字符。例如：当I/O口输出的代码为0011 1111时，数码管显示的字符为0。这样形成的显示字符的代码称为显示代码或段选码。表8-3为十六进制数字的显示代码。

2. 数码管显示器与单片机的接口电路

数码管显示器有二种工作方式，即静态显示方式和动态显示方式。

在静态显示方式下，每位数码管的a～g和h端与一个8位的I/O口相连。要在某一位数码管上显示字符时，只要从对应的I/O口输出并锁存其显示代码即可。其特点为：数码管中的发光二极管恒定地导通或截止，直到显示字符改变为止。图8-30为采用BCD或十六进制-七段锁存译码驱动器MC14495构成的多位数码管静态显示器与8031的接口电路。MC14495的输入锁存选通信号$\overline{LE}=0$时，允许数据输入到其内部锁存器中；当$\overline{LE}=1$时，输入数据被锁存。

表8-3 十六进制数字的显示代码

十六进制数	h	g	f	e	d	c	b	a	显示代码
0	0	0	1	1	1	1	1	1	3FH
1	0	0	0	0	0	1	1	0	06H
2	0	1	0	1	1	0	1	1	5BH

续表 8-3

十六进制数	h	g	f	e	d	c	b	a	显示代码
3	0	1	0	0	1	1	1	1	4FH
4	0	1	1	0	0	1	1	0	66H
5	0	1	1	0	1	1	0	1	6DH
6	0	1	1	1	1	1	0	1	7DH
7	0	0	0	0	0	1	1	1	07H
8	0	1	1	1	1	1	1	1	7FH
9	0	1	1	0	1	1	1	1	6FH
A	0	1	1	1	0	1	1	1	77H
B	0	1	1	1	1	1	0	0	7CH
C	0	0	1	1	1	0	0	1	39H
D	0	1	0	1	1	1	1	0	5EH
E	0	1	1	1	1	0	0	1	79H
F	0	1	1	1	0	0	0	1	71H
·	1	0	0	0	0	0	0	0	80H

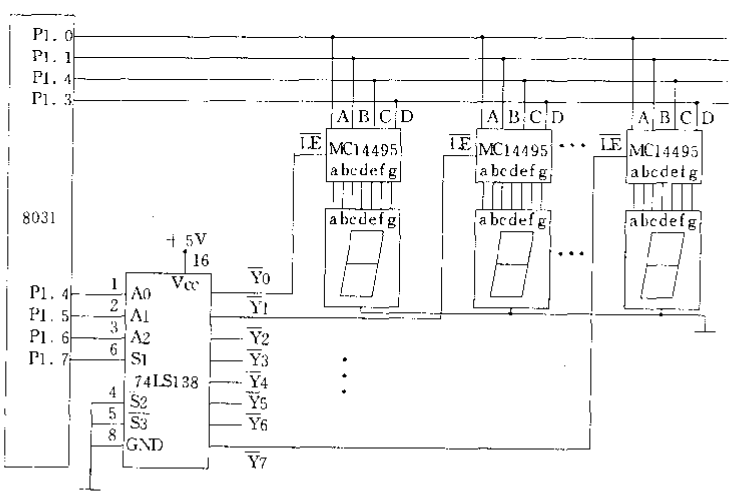


图 8-30　数码管静态显示接口电路

被显示的数据直接由 P1 口的低 4 位输出，P1.4～P1.6 用来选择数码管，经译码后产生输入锁存选通信号，P1.7 用来控制多位显示器数据字符的改写和锁存。当 P1.7 为高电平时，允许改写各位的显示字符；当 P1.7 输出低电平时，$\overline{Y}0\sim\overline{Y}7=1$，各位显示字符不

变。下面是将显示缓冲区 78H～7FH 中的 BCD 码送数码管显示器的程序。

```
DIR:  MOV   R0,#78H          ;置首地址
      MOV   R2,#00H          ;初始化位计数器
      MOV   R7,#08H          ;置循环次数
      SETB  P1.7
LOOP: MOV   A,R2
      SWAP  A
      ADD   A,@R0            ;显示数据送低 4 位
      MOV   P1,A             ;输出显示数据和位地址
      INC   R0               ;指向下一个数
      INC   R2               ;指向下一位
      DJNZ  R7,LOOP
      CLR   P1.7
      RET
```

动态显示方式的每位数码管都需要一个数据锁存器，因此，其硬件电路较为复杂。但它的显示程序非常简单。

在动态显示方式中，各位数码管的 a～h 端并连在一起，与单片机系统的一个 I/O 口相连，从该 I/O 口输出显示代码。每只数码管的共阳极或共阴极则与另一 I/O 口相连，控制被点亮的位。动态显示的特点是：每一时刻只能有 1 位数码管被点亮，各位依次轮流被点亮；对于每一位来说，每隔一段时间点亮一次。为了每位数码管能够充分被点亮，二极管应持续发光一段时间。利用发光二极管的余辉和人眼的驻留效应，通过适当地调整每位数码管被点亮的时间间隔，可以观察到稳定的显示输出。

图 8-31 是通过 8155 扩展的 6 位动态数码管显示器的接口电路。由 PB 口输出显示代码，PA 口输出位选码。设显示数据的缓冲区为 79H～7EH，由前面的分析，显示程序的流程框图如图 8-32 所示。

程序如下(其中 DLT 为延时 2ms 子程序)：

```
DIR:  MOV   R0,#79H          ;置显示缓冲区首地址
      MOV   R3,#01H          ;置位选码初值
      MOV   A,R3
LOOP: MOV   DPTR,#7F01H      ;DPTR←PA 口地址
      MOVX  @DPTR,A          ;输出位选码
      INC   DPTR             ;指向 PB 口
      MOV   A,@R0            ;取被显示的数据
      ADD   A,#12H           ;形成查表的偏移地址
      MOVC  A,@A+PC          ;求出显示代码
      JNB   PSW.5,DIR1       ;判断是否显示小数点
      SETB  ACC.7            ;显示小数点
```

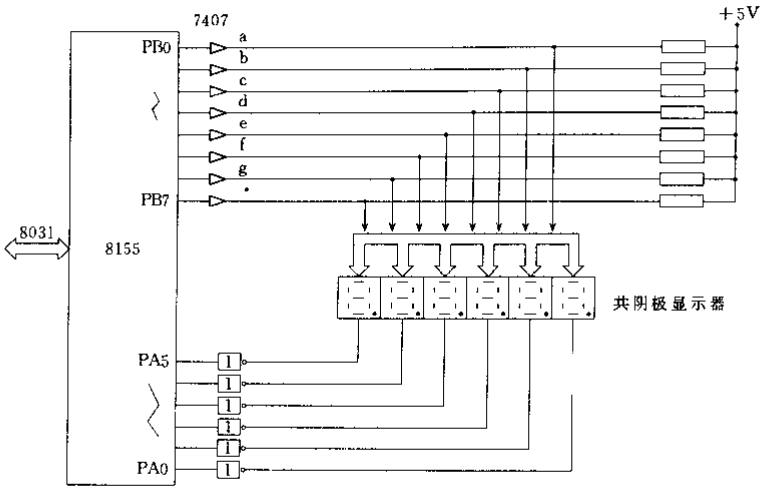


图8-31　6位动态显示器的接口电路

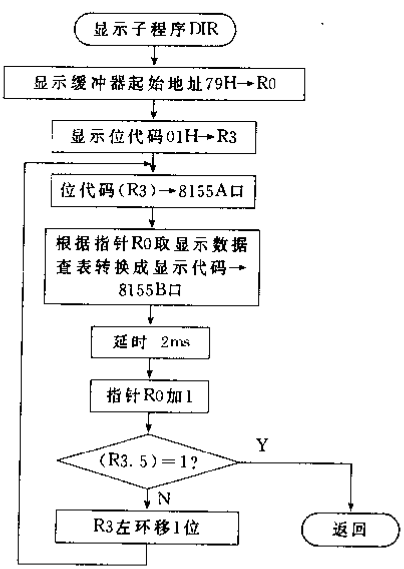


图8-32　显示程序框图

```
DIR1:   MOVX   @DPTR,A          ;输出显示代码
        ACALL  DLT              ;延时
```

```
        INC     R0                  ;指向下一个显示数据
        MOV     A,R3
        JB      ACC.5,LOOP1         ;判断6位是否显示完毕
        RL      A                   ;形成下一个位代码
        MOV     R3,A
        AJMP    LOOP
LOOP1:  RET
        DB      3FH,06H,5BH,4FH,66H,6DH
        DB      7DH,07H,7FH,6FH,77H,7CH
        DB      39H,5EH,79H,71H
DLF:    MOV     R7,#04H
DLT1:   MOV     R6,#0FFH
DLT2:   DJNZ    R6,DLT2
        DJNZ    R7,DLT1
        RET
```

上面我们分别介绍了键盘接口电路和数码管显示器接口电路。在单片机应用系统中，往往同时需要扩展键盘和显示器，图 8-33 为 8031 通过 8155 扩展 32 键键盘和 6 位动态数码管显示器接口电路，供参考。

第三节　专用键盘显示器接口芯片 8279 与单片机的接口

8279 是 Intel 公司生产的通用可编程键盘和显示器接口电路芯片。8279 可以实现对键盘和显示器的自动扫描，识别闭合键的键号，完成显示器动态显示。从而大大节省了 CPU 处理键盘和显示器的时间，提高了 CPU 的工作效率。另外，8279 与单片机的接口简单，显示稳定，工作可靠，应用愈来愈广泛。

一、8279 的引脚及结构

8279 为 40 引脚芯片，其引脚如图 8-34 所示，逻辑符号如图 8-35 所示，内部结构如图 8-36 所示。

8279 的各组成部分介绍如下。

1. I/O 控制及数据缓冲器

数据缓冲器为双向、三态缓冲器，与内部和外部数据总线相连，用于传送 CPU 与 8279 之间的命令/状态和数据。

I/O 控制处理由 CPU 送来的控制信号。其中：

$\overline{CS}$为片选信号，当$\overline{CS}=0$时，8279 被选中。

$\overline{RD}$和$\overline{WR}$为读/写控制信号。

A0 用于区分信息的特征。当 A0=1 时，CPU 从 8279 读出和写入的是状态和命令，当

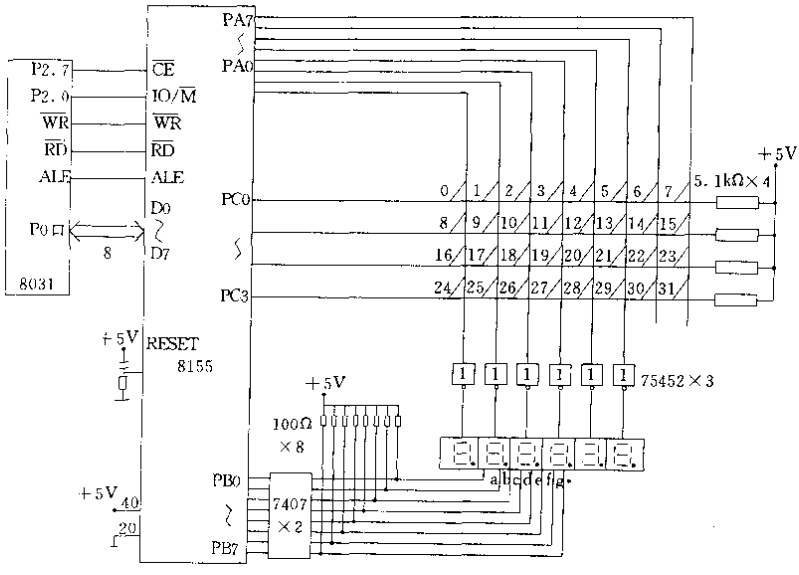


图8-33 键盘/显示器的接口电路

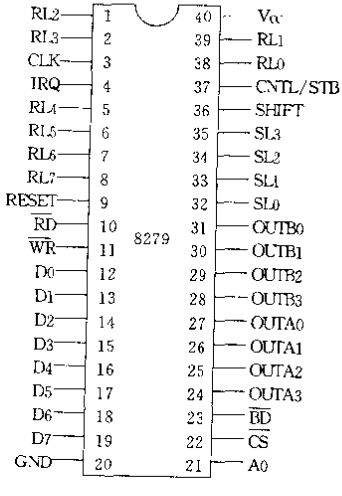


图8-34 8279的引脚

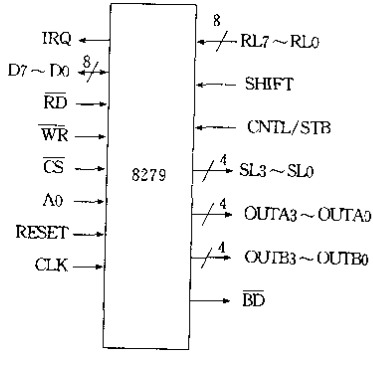


图8-35 8279的逻辑符号

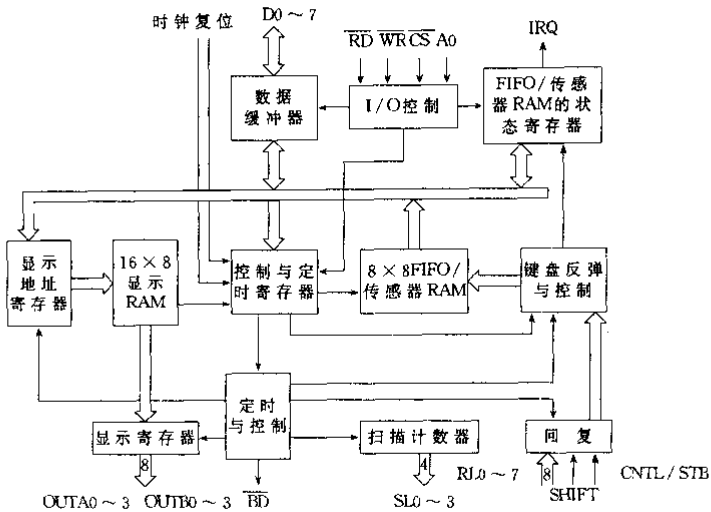


图 8-36　8279 的内部结构

A0＝0 时，读出和写入的是数据。

2. 控制与定时寄存器

控制与定时寄存器用来存放键盘及显示器的工作方式和其它操作控制命令。这些寄存器一接收并锁存控制命令，就通过译码产生相应的信号，完成相应的控制功能。

定时控制包含一个可编程的 5 位计数器，可选择分频系数，对外部输入的时钟信号进行分频，产生 100kHz 的内部定时信号。

3. 扫描计数器

扫描计数器有二种工作方式。一种为外部译码方式(编码方式)，计数器以二进制方式计数，4 位计数状态从扫描线 SL0～SL3 输出，经外部译码器译码后，为键盘和显示器提供扫描线；另一种为内部译码方式(译码方式)，扫描计数器的低 2 位经内部译码器译码后由 SL0～SL3 输出。

4. 回复缓冲器、键盘去抖及控制

RL0～RL7 8 条回复线的状态信号输入到回复缓冲器并锁存。

在键盘工作方式中，回复线作为行列式键盘的列线。回复线搜寻闭合键，当有键闭合时，经去抖动，形成键盘数据送入 8279 的 FIFO 存储器。

在传感器开关状态矩阵方式中，回复线上的内容直接送入 FIFO 存储器。在选通输入方式中，回复线的内容在 CNTL/ STB 端的脉冲上升沿被送入 FIFO 存储器。

5. FIFO RAM 及其状态寄存器

8279 具有 8 个 8 位的先进先出 FIFO 键输入缓冲器。当有键闭合时，键盘数据送入 FIFO 存储器，其输入或读出遵循先入先出原则。FIFO 状态寄存器存放 FIFO 的工作状

态，当 FIFO 不空时，使 IRQ 置位，向 CPU 申请中断。

在传感器矩阵方式中，这个存储器为传感器存储器，用于存放传感器的状态。

6. 显示 RAM 和显示地址寄存器

8279 设有 16 个 8 位显示数据缓冲区(RAM)，存放显示数据的段选码。缓冲区的数据轮流从显示寄存器输出，在显示扫描配合下，送显示器的各位，实现动态显示。

显示地址寄存器用于存放 CPU 读/写显示 RAM 的地址，它可在命令的设置下，实现读出或写入时的自动递增。

8279 的引脚功能：

D0～D7：三态、双向数据线。用于 CPU 与 8279 之间的数据/命令传送。

CLK：时钟信号输入端。

RESET：复位信号输入端，高电平有效。8279 的复位状态为：

16 个字符显示；

外部译码扫描键盘——双键锁定；

程序时钟编程为 31。

$\overline{CS}$：片选信号输入端，低电平有效，表示被选中。

A0：数据/命令选择信号输入端。当 A0＝1 时，CPU 写入的数据为命令字，读出的数据为状态字；当 A0＝1 时，CPU 读、写的均为数据。

$\overline{RD}$、$\overline{WR}$：读、写控制信号，低电平有效。

IRQ：中断请求输出信号，高电平有效。在键盘工作方式中，当 FIFO RAM 中存有键盘数据时，IRQ 变为高电平，CPU 每次从 RAM 中读取数据时，IRQ 变为低电平。若 FIFO RAM 中仍有数据，IRQ 再次恢复高电平。在传感器工作方式中，当检测到传感器状态变化时，IRQ 就出现高电平。

SL0～SL3：扫描信号输出端。用于对键盘和显示器进行扫描。可通过键盘/显示方式命令字选择为外部译码方式和内部译码方式。

RL0～RL7：回复输入端。作为行列式键盘或传感器矩阵的列(或行)信号的输入端，自动检测信号的变化，并输入到 FIFO RAM 中。

SHIFT：移位信号输入端，高电平有效。该信号自动进入键盘数据的位 6(D6)，用于扩充键的功能，作为按键的上、下挡功能转换键。在传感器和选通工作方式中，该信号无效。

CNTL/STB：控制/选通信号输入端。高电平有效。在键盘工作方式中，该信号自动进入键盘数据的最高位(D7)，一般用来扩充按键的功能，作为控制功能键。在选通输入方式时，该信号的上升沿将 RL0～RL7 端的数据送入 FIFO RAM 中。在传感器工作方式中，该信号无效。

OUTA0～OUTA3：A 组显示信号输出端。

OUTB0～OUTB3：B 组显示信号输出端。

这两组端口都作为显示数据的输出端口，与扫描信号 SL0～SL3 同步。两组可以独立使用，也可以合并使用，OUTB0 为最低位，OUTA3 为最高位。

$\overline{BD}$：显示消隐信号输出位，低电平有效。该信号在显示数据切换或使用消隐命令时，将显示消隐。

二、8279 的命令字、状态字和数据格式

8279 与 CPU 之间的信息交换可分为 3 种类型，即命令字、状态字和数据。它们的作

用和格式如下：

1. 命令字的格式

当 8279 的 A0 端输入高电平时，CPU 向 8279 写入的数据为命令字，读出的数据为状态字。

(1)键盘/显示方式设置命令字

命令格式：

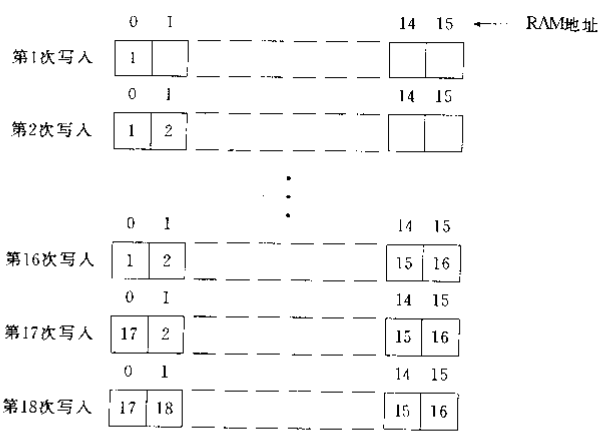
D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	D	D	K	K	K

D7D6D5＝000 为键盘/显示方式设置命令的特征位。

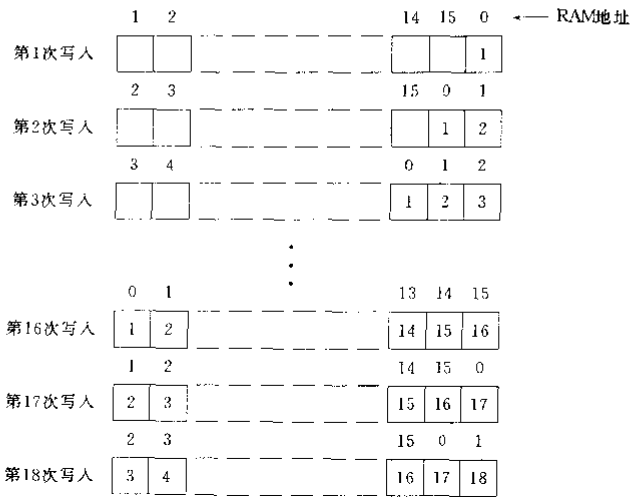
DD(D4D3)为显示方式控制位，其定义如下：

D4	D3	显示方式
0	0	8 个字符显示——左边输入
0	1	16 个字符显示——左边输入
1	0	8 个字符显示——右边输入
1	1	16 个字符显示——右边输入

8279 最多可以控制 16 位数码管显示器，当超过 8 位时，也需选择 16 位显示。显示器的每一位与显示数据缓冲区的一个单元对应。显示数据输入缓冲区可以选择二种方式，左边输入和右边输入。在左边输入方式中，地址为 0～15 的 16 个显示数据 RAM 单元分别对应显示器的 0(左)～15(右)位。CPU 依次从 0 地址或某一地址开始将显示数据的段选码写入显示缓冲区，当地址超过 15 时，再从 0 地址开始写入。其过程如下(从 0 地址开始写入)：



右端输入是移位输入方式，输入的数据总是写入右边的显示 RAM 单元，数据写入后，原来各单元的内容左移 1 位字节，原来最左端的显示 RAM 单元的内容被移出。其过程如下：



在右端输入方式中，显示器的位置和显示 RAM 单元的地址并不是对应的。

在内部译码方式中，显示器只能接 4 位。

KKK(D2D1D0)为键盘工作方式控制位，具体如下：

D2	D1	D0	工 作 方 式
0	0	0	外部译码扫描键盘，双键互锁
0	0	1	内部译码扫描键盘，双键互锁
0	1	0	外部译码扫描键盘，N 键依次读出
0	1	1	内部译码扫描键盘，N 键依次读出
1	0	0	外部译码扫描传感器矩阵
1	0	1	内部译码扫描传感器矩阵
1	1	0	选通输入，外部译码扫描显示
1	1	1	选通输入，内部译码扫描显示

双键互锁是指当有两个键同时闭合时，任一个键的编码信息也不能进入 FIFO RAM 中，直至仅剩一个键闭合时，该键的编码信息才进入 FIFO RAM 中。

在 N 键依次读出工作方式中，任何一个键闭合时，片内去抖电路等待两个键盘扫描周期，如果该键仍然闭合，其键码信息送入 FIFO RAM 中。如果是 N 键同时闭合，键盘扫描根据发现它们的顺序依次读入 FIFO RAM 中。

(2)时钟编程命令字

命令格式：

D7	D6	D5	D4	D3	D2	D1	D0
0	0	1	P	P	P	P	P

D7D6D5＝001 为时钟编程命令字的特征位。

PPPPP(D4D3D2D1D0)用来设定对外部时钟信号的分频系数 N，N 取 2～31。通过对外部输入信号的 N 分频获得 8279 所需的 100kHz 的内部时钟信号。

(3)读 FIFO/传感器 RAM 命令字

命令格式：

D7	D6	D5	D4	D3	D2	D1	D0
0	1	0	AI	×	A	A	A

D7D6D5＝010 为读 FIFO/传感器命令字的特征位。在 CPU 读传感器 RAM 之前，必须用该命令设置传感器 RAM 中要读单元的地址。

AAA(D2D1D0)为传感器 RAM 的地址。

AI(D4)为自动增量标志位。AI＝1 时，每次读出后地址自动加 1，指向下一个 RAM 单元；AI＝0 时，CPU 仅读出一个单元的内容。

该命令只在传感器工作方式中使用，在键盘工作方式中，读出操作按先进先出的顺序进行，不需此命令。

(4)读显示 RAM 命令字

命令格式：

D7	D6	D5	D4	D3	D2	D1	D0
0	1	1	AI	A	A	A	A

D7D6D5＝011 为读显示 RAM 命令字特征位。该命令字用来设定要读出的显示 RAM 的地址。

AAAA(D3D2D1D0)用来设定显示 RAM 单元的地址。

AI(D4)为自动增量标志位。当 AI＝1 时，CPU 每次读出后，地址自动加 1。

(5)写显示 RAM 命令字

命令格式：

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	AI	A	A	A	A

D7D6D5＝100 为写显示 RAM 命令字特征位。在写显示 RAM 之前用该命令字来设定要写入 RAM 单元的地址。

AAAA(D3D2D1D0)为要写入显示 RAM 单元的地址。

AI(D4)为自动增量标志位。当 AI=1 时,每次写入后地址自动加 1。

(6)显示禁止写入、消隐命令字

命令格式:

D7	D6	D5	D4	D3	D2	D1	D0
1	0	1	×	IWA	IWB	BLA	BLB

D7D6D5=101 为显示禁止写入和消隐命令字特征位。

IWA、IWB(D3、D2)用来分别屏蔽 A 组和 B 组显示 RAM。当 OUTA0~OUTA3 和 OUTB0~OUTB3 分别作为两个独立的显示器输出端口时,若 IWA=1,可改写显示 RAM 的高 4 位,而低 4 位不变;若 IWA=1,可改写显示 RAM 的低 4 位,而高 4 位不变。

BLA、BLB(D1、D0)为消隐设置位。分别用于两组显示的消隐设置。当 BL=1 时,对应的显示输出被消隐;当 BL=0 时,恢复显示。

(7)清除命令字

命令格式:

D7	D6	D5	D4	D3	D2	D1	D0
1	1	0	C_D	C_D	C_D	C_F	C_A

D7D6D5=110 为清除命令的特征位。该命令用来清除 FIFO RAM 和显示 RAM。

CDCDCD(D4D3D2)用来设定清除显示 RAM 的方式,定义如下:

D4	D3	D2	清除显示 RAM 的方式
1	0	×	将显示 RAM 清 0
1	1	0	将显示 RAM 清成 20H
1	1	1	将显示 RAM 全部置 1
0	×	×	若 CA=0,不清除;若 CA=1,则 D3、D2 仍有效

CF(D1)用来清除 FIFO RAM。当 CF=1 时,执行清除命令时,FIFO RAM 被置空,中断请求信号被复位,同时传感器 RAM 的读出地址也被置位。

CA(D0)为总清除标志位,具有 CD 和 CF 的功能。当 CA=1 时,对显示 RAM 的清除方式由 D3、D2 二位确定。

清除显示 RAM 大约需 100μs 时间,在此期间 CPU 不能向显示 RAM 中写入数据。

(8)结束中断/错误方式设置命令字

命令格式:

1	1	1	E	×	×	×	×

在传感器工作方式中,若传感器 RAM 读出地址的自动增量标志位 AI=0,则中断请求信号 IRQ 在 CPU 读出数据后即被清除;若 AI=0,CPU 读出数据后,并不清除 IRQ,必须通过写入结束中断/错误方式设置命令字(E=1)来清除 IRQ。

在键盘工作方式中,若 8279 设置为 N 键依次读出方式后,若 CPU 给 8279 又写入结

束中断/错误方式设置命令(E=1),则 8279 将以特定的错误方式工作。如果 8279 发现多个键同时闭合,则 FIFO 状态字的错误特征位 S/E 置 1,并产生中断请求信号和阻止写入 FIFO RAM。

(9)状态字

在键输入和选通输入方式中,状态字给出了 FIFO RAM 中的数据个数、是否出错等信息,其格式为:

D7	D6	D5	D4	D3	D2	D1	D0
DU	S/E	O	U	F	N	N	N

NNN(D2D1D0)为 FIFO RAM 中数据的个数。

F(D3)=1 时,表示 FIFO RAM 已满。

U(D4):当 FIFO RAM 已空,CPU 读时,U 置位。

O(D5):当 FIFO 已满,又输入一个数据时,O 置位,表示溢出错误。

S/E(D6)用于传感器矩阵方式,几个传感器同时闭合时,该位置 1。

DU(D7)在执行清除命令期间置 1,此时对显示 RAM 操作无效。

(10)输入数据的格式

当 A0=0 时,CPU 对 8279 读、写的均为数据。写入的为要显示数符的段选码,读出的数据为键盘数据或传感器矩阵数据。

在键盘工作方式中,当有键闭合时,其行号和列号(分别由 RL0~RL7 和 SL0~SL3 确定)输入 FIFO RAM 中。其格式为:

D7	D6	D5	D4	D3	D2	D1	D0
CNTL	SHIFT	SCAN			RETURN		

RETURN(D2D1D0)为闭合键的列号,由 RL0~RL7 状态的编码确定。

SCAN(D5D4D3)为闭合键的行号,是扫描计数器的值,与 SL0~SL3 的状态对应。

SHIFT(D6)为控制键 SHIFT 的状态位。

CNTL(D7)为控制键 CNTL 的状态位。

在传感器方式或选通方式中,当传感器的状态变化,或选通信号有效时,RL0~RL7 的状态输入到 FIFO RAM 中,该数据的格式为:

D7	D6	D5	D4	D3	D2	D1	D0
RL7	RL6	RL5	RL4	RL3	RL2	RL1	RL0

三、8279 键盘、显示器接口

8279 最多可以接 16 位显示器,一个 8×8 的行列式键盘。当有键按下时,键号自动进入 FIFO RAM,并置中断请求信号有效,向 CPU 请求中断。要显的数据的段选码送到显示 RAM 中,8279 自动完成扫描显示。CPU 所要做的工作是对 8279 进行初始化,输出显示数据的段选码,有键按下时,读入键号。因此,在 8279 键盘、显示系统中,CPU 用于处理键盘和显示器的时间明显减少,提高了 CPU 的工作效率。

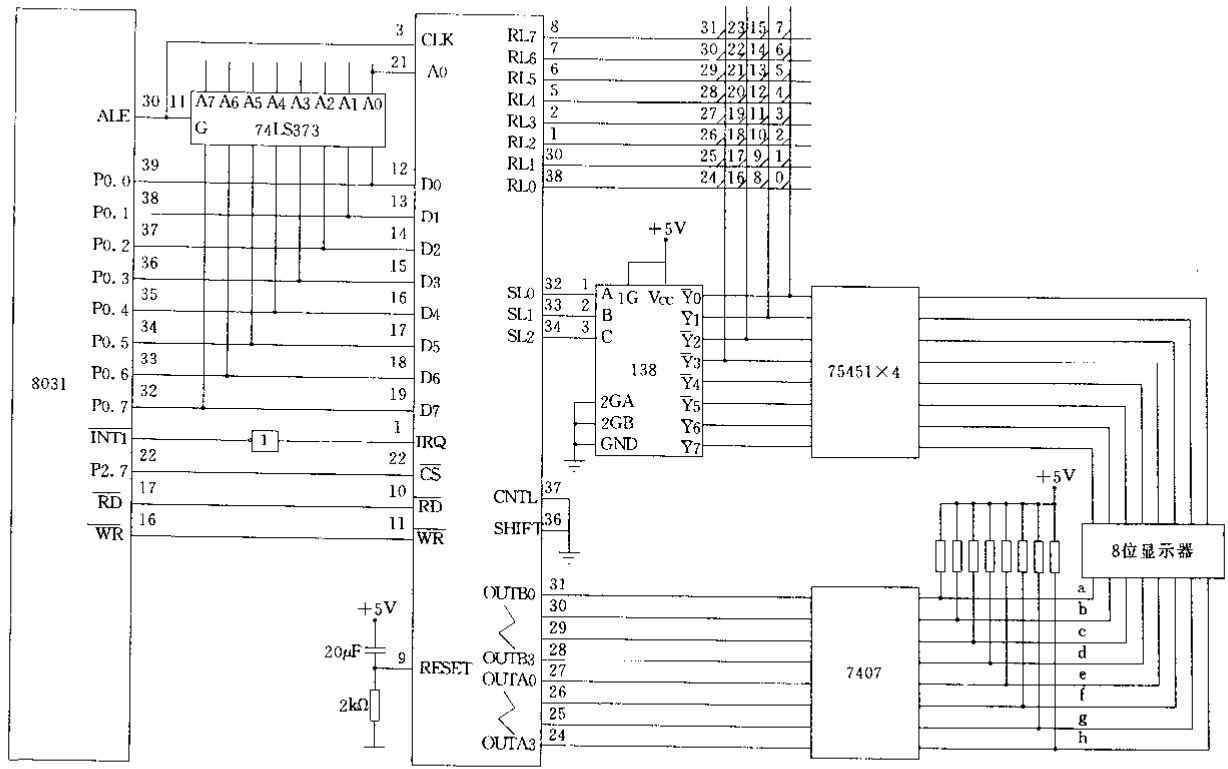


图8-37　8279键盘、显示器接口电路

图 8-37 为 8 位显示器、8×4 的行列式键盘的接口电路。电路中 8279 采用外部译码方式，SL0～SL2 经译码器 74LS138 译码产生扫描信号 $\overline{Y}0$～$\overline{Y}7$，作为键盘的列线和显示器的扫描信号。OUTB0～OUTB3 和 OUTA0～OUTA3 作为 8 位段选码的输出端口。IRQ 经反向后接 8031 单片机的$\overline{INT}1$ 端，当有键按下时，申请中断，在中断服务子程序中将键号读入 CPU。按图 8-37，8279 的命令/状态口的地址为 7FFFH，数据口的地址为 7FFEH。键盘数据输入中断服务程序的框图如图 8-38 所示，显示子程序框图如图 8-39 所示。

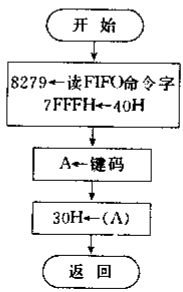


图8-38 键盘输入程序框图

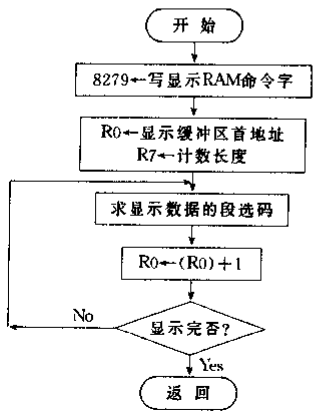


图8-39 显示程序框图

8279 键盘、显示器程序如下：

8279 初始化程序：

```
MAIN:CLR   EA                    ;关中断
     MOV   DPTR,#7FFFH
     MOV   A,#0D1H
     MOVX  @DPTR,A
LP:  MOVX  A,@DPTR
     JB    ACC.7,LP              ;等待清除结束
     MOV   A,#00H                ;置键盘/显示方式设置命令字
     MOVX  @DPTR,A
     MOV   A,#2AH                ;置时钟编程命令 ALE 10 分频
     MOV   @DPTR,A
     SETB  IT1                   ;外部中断 1 边沿触发
     SETB  EA
     SETB  EX1                   ;开外部中断 1
     ⋮
```

键码输入中断服务程序(键码送 30H 单元)：

```
KEYI:  PUSH   PSW
       PUSH   DPH
       PUSH   DPL
       PUSH   A
       MOV    DPTR,#7FFFH
       MOV    A,#40H
       MOVX   @DPTR,A
       MOV    DPTR,#7FFEH      ;指 8279 的数据口
       MOVX   A,@DPTR          ;读入键码
       MOV    30H,A
       POP    A
       POP    DPL
       POP    DPH
       POP    PSW
       RETI
```

显示子程序(设显示数据缓冲区为 70H～77H):

```
DIR:   MOV    DPTR,#7FFFH
       MOV    A,#90H           ;置写显示 RAM 命令
       MOV    @DPTR,A
       MOV    R0,#70H
       MOV    R7,#08H
       MOV    DPTR,#7FFEH
LOOP:  MOV    A,@R0
       ADD    A,#05H
       MOVC   A,@A+PC
       MOVX   @DPTR,A
       INC    R0
       DJNZ   R7,LOOP
       RET
TAB:   DB     3FH,06H,5BH,4FH,66H,6DH
       DB     7DH,07H,7FH,6FH,77H,7CH
       DB     39H,5EH,79H,71H
```

第四节　MCS-51 单片机串行口扩展

MCS-51 单片机的串行口有 4 种工作方式,在第七章中,介绍了异步串行通讯的应用。这里介绍利用串行口的同步串行通讯方式(方式 0),进行并行数据的输入/输出接口

的扩展。这种扩展方法不占用外部数据存储器地址空间，占用硬件资源少，并且同时可以扩展多个 8 位接口；但其操作速度较慢，且扩展的接口数量越多，速度越慢。

一、串行口扩展的并行输入接口

74LS165 是 8 位并行输入串行输出移位寄存器，其引脚如图 8-40 所示。D7～D0 为 8 位并行数据输入端，D_{SR} 为串行数据输入端，CP_A 和 CP_B 为时钟脉冲输入端，SH/LD 为串行移位/并行输入控制信号输入端，Q7 为串行数据输出端。

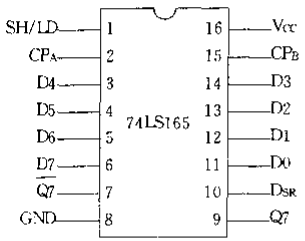


图8-40　74LS165的引脚图

图 8-41 为采用 74LS165 扩展的二个 8

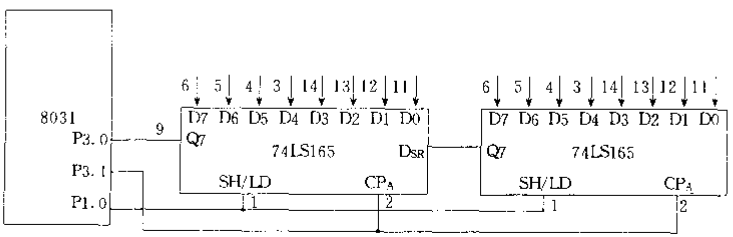


图8-41　串行口扩展的并行输入接口电路

位并行输入口电路。8031 的串行输入端 RXD(P3.0)与 74LS165 的串行数据输入端 Q7 相连，移位脉冲输出端 TXD(P3.1)与两片 74LS165 的时钟脉冲输入端 CP_A 相连，P1.0 与 74LS165 的 SH/LD 端相连用于控制串行移位和并行数据的输入，后一片 74LS165 的串行数据输出端与前一片的串行数据输入端相连，实现多位并行数据端口的扩展。

下面是根据图 8-41 中两个并行的 8 位数据端口中输入 10H 个字节的数据，并存入内部数据存储器以 30H 单元开始的缓冲区中的程序：

```
        MOV   R7,#10H          ;设置循环次数
        MOV   R0,#30H          ;设置缓冲区的首地址
        SETB  F0               ;读入字节的奇偶标志
RCV0:   CLR   P1.0             ;将并行数据锁存到 74LS165
        SETB0 P1.0             ;允许串行移位
RCV1:   MOV   SCON,#00010000B  ;串行口工作方式设置为方式 0,允许接收
STP:    JNB   RI,STP           ;等待接收一个字节的数据
        CLR   RI
        MOV   A,SBUF           ;接收的数据送 A
        MOV   @R0,A            ;数据送数据缓冲器
        INC   R0               ;指向下一个字节单元
        CPL   F0               ;奇偶标志转换
```

```
        JB      F0,RCV2          ;若读入了偶数个数据,转去读下一个数据
        SJMP    RCV1             ;若读入了奇数个数据,转去读下一个数据
RCV2:   DJNZ    R7,RCV0
        ⋮
```

二、串行口扩展的行输出接口

74LS164是8位串入并出移位寄存器,其引脚如图8-42所示。各引脚功能如下:

D_{SA}、D_{SB}:串行数据输入端;

Q7～Q0:并行数据输出端口;

CP:时钟脉冲信号输入端;

$\overline{Cr}$:清除信号输入端。

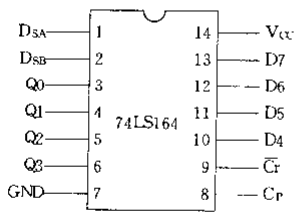


图8-42　74LS164的引脚图

图8-43为采用74LS164扩展的二个8位并行输出接口电路。8031串行数据输出端RXD(P3.0)与74LS164的串行数据的输入端相连,移位脉冲输出端TXD(P3.1)与74LS164的时钟脉冲输入端CP相连,P1.0

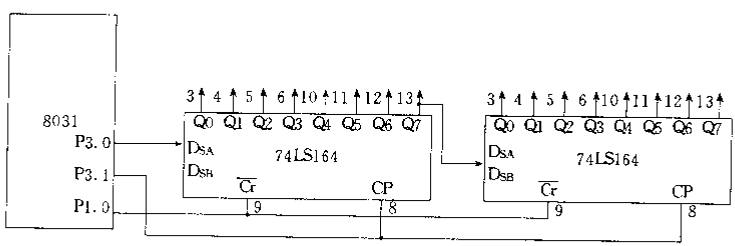


图8-43　串行口扩展的并行输出接口电路

口与74LS164的清除信号端相连,控制清除并行端口的数据。图8-43为串行口扩展的并行输出接口的应用——8031串行口扩展的键盘、显示器接口。

图8-44中,用8片74LS164扩展了8位数码管显示器,用1片74LS164扩展一个2×8的行列式键盘。P3.4、P3.5与键盘的行相连,扫描码与74LS164的并行数据输出端口相连。P3.3用来区别键盘扫描码和段选码。键盘扫描子程序显示子程序如下:

显示子程序:

```
DIR:    SETB    P3.3             ;允许数据送入显示器
        CLR     TI
        MOV     R7,#08H
        MOV     R0,#78H          ;置显示数据缓冲区首地址
DL0:    MOV     A,@R0            ;取显示数据
        ADD     A,#0DH
        MOVC    A,@A+PC          ;查表求出段选码
```

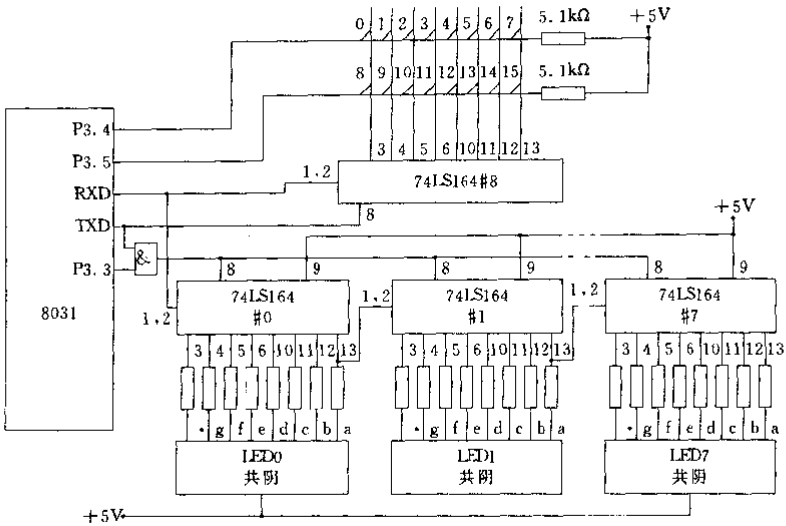


图8-44　串行口扩展的键盘、显示器接口电路

```
        CPL   A
        MOV   SBUF,A          ;段选码送串行口
DL1:    JNB   TI,DL1          ;等待串行发送一帧数据结束
        CLR   TI
        INC   R0
        DJNZ  R7,DL0
        CLR   P3.3            ;禁止数据送入显示器
        RET
TAB:    DB    3FH,06H,5BH,4FH,66H,6DH
        DB    7DH,07H,7FH,6FH,77H,7CH
        DB    39H,5EH,79H,71H
键盘扫描子程序:
KEY:    MOV   A,#00H
        MOV   SBUF,A          ;由 74LS164(8)输出 00H
KL0:    JNB   TI,KL0          ;等待串行数输出结束
        CLR   TI
KL1:    JNB   P3.4,PK1        ;判断第一排是否有键按下
        JB    P3.5,KL1        ;判断第二排是否有键闭合
PK1:    ACALL DL10
        JNB   P3.4,PK2        ;去抖动
```

```
        JB      P3.5,KL1
PK2:    MOV     R7,#08H         ;确定有键闭合
        MOV     R6,#0FEH        ;R6←键盘扫描代码
        MOV     R3,#00H
        MOV     A,R6
KL5:    MOV     SBUF,A
KL2:    JNB     TI,KL2
        CLR     TI
        JNB     P3.4,PKONE      ;第一排有键闭合转
        JB      P3.5,NEXT       ;本列无键闭合转下一列
        MOV     R4,#08H         ;第二排有键按下
        AJMP    PK3
PKONE:  MOV     R4,#00H
PK3:    MOV     SBUF,#00H
KL3:    JNB     TI,KL3
        CLR     TI
KL4:    JNB     P3.4,KL4
        JNB     P3.5,KL4        ;等待闭合键释放
        MOV     A,R4
        ADD     A,R3            ;求键码
NC:     RET
NEXT:   MOV     A,R6
        RL      A               ;键盘扫描码左移
        MOV     R6,A
        INC     R3
        DJNZ    R7,KL5
        AJMP    NC
DL10:   MOV     R7,#0AH
DL:     MOV     R6,#0FFH
DL6:    DJNZ    R6,DL6
        DJNZ    R7,DL
        RET
```

第五节　单片机与 D/A 和 A/D 转换器的接口

在测控系统中，除了数字量之外，还存在着大量的模拟量，如：温度、压力、流量、速度、电压、电流等。而计算机只能处理数字量，要实现对模拟量的测量和控制，首先必须将模拟量转换成数字量(A/D 转换)。相反，计算机输出时，有时也需要将数字量转换成模拟量

(D/A 转换)。目前,A/D 转换和 D/A 转换电路都已集成化,它们具有体积小、功能强、可靠性高、误差小、功耗低、与计算机接口简单等特点。

一、单片机与 D/A 转换器的接口

D/A 转换器(DAC)输入的是数字量,经转换输出的是模拟量。DAC 的技术指标很多,如:分辨率、满刻度误差、线性度、绝对精度、相对精度、建立时间、输入/输出特性等。这里只介绍几种主要的技术性能指标。

1. DAC 的主要技术指标

(1)分辨率

DAC 的分辨率反映了它的输出模拟电压的最小变化量。其定义为输出满刻度电压与 2^n 的比值,其中 n 为 DAC 的位数。如:8 位 DAC 的满刻度输出电压为 5V,则其分辨率为 $5/2^8=5/255(V)$;10 位 DAC 的分辨率为 $5/2^{10}=5/1024(V)$。可见,DAC 的位数越高,分辨率越小。

(2)建立时间

建立时间是描述 DAC 转换速度快慢的参数。其定义为从输入数字量变化到输出达到终值误差 $\pm\frac{1}{2}$LSB(最低有效位)所需的时间。高速 DAC 的建立时间可达 1μs。

(3)接口形式

接口形式是 DAC 输入/输出特性之一。包括输入数字量的形式:十六进制或 BCD,输入是否带有锁存器等。

2. DAC0832

DAC0832 为 8 位 D/A 转换器。单电源供电,范围为+5V～+15V,基准电压范围为±10V。电流的建立时间为 1μs。CMOS 工艺功耗 20mw。输入设有两级缓冲锁存器。

DAC0832 为 20 引脚,采用双列直插封装,其引脚如图 8-45 所示。

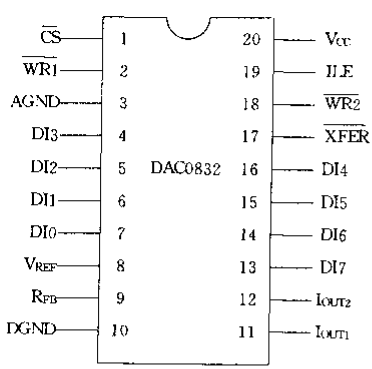


图8-45 DAC0832的引脚图

各引脚的功能如下:

DI7～DI0:数字量输入端;

$\overline{CS}$:片选信号,低电平有效;

ILE:数据锁存允许信号,高电平有效;

$\overline{WR1}$:第 1 写信号,低电平有效;

$\overline{WR2}$:第 2 写信号,低电平有效;

$\overline{XFER}$:数据传送控制信号,低电平有效;

I_{OUT1}:电流输出端 1;

I_{OUT2}:电流输出端 2;

R_{FB}:反馈电阻端;

V_{REF}:基准电压,基电压范围为−10～+10V;

DGND：数字地；

AGND：模拟地。

DAC0832 的内部结构如图 8-46 所示，它主要包括输入寄存器，DAC 寄存器，D/A 转换器和控制逻辑电路。

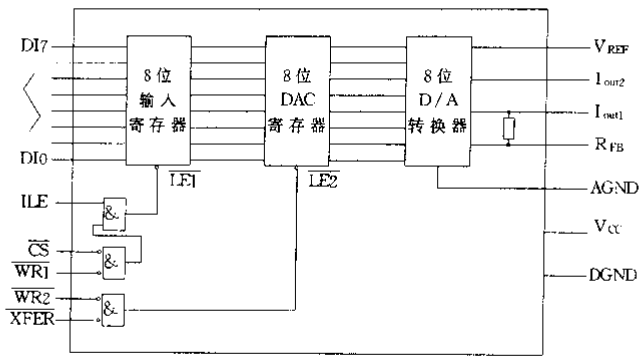


图 8-46　DAC0832 的内部结构

图中，$\overline{LE1}$是输入寄存器锁存选通信号，其逻辑表达式为$\overline{LE1}=\overline{\overline{WR1}\cdot\overline{CS}\cdot ILE}$。$\overline{LE2}=\overline{WR2}\cdot\overline{XFER}$为 DAC 寄存器的锁存选通信号。

DAC0832 是电流输出型 D/A 转换器，要得到电压信号，输出端需接运算放大器进行转换，其电路如图 8-47 所示。

根据数据输入的过程，DAC0832 有三种联接方式：二级缓冲器联接方式，单级缓冲器联接方式和直通联接方式，如图 8-48 所示。

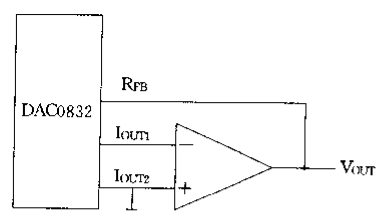


图 8-47　DAC0832 的输出电路

3. 单片机与 DAC0832 的接口

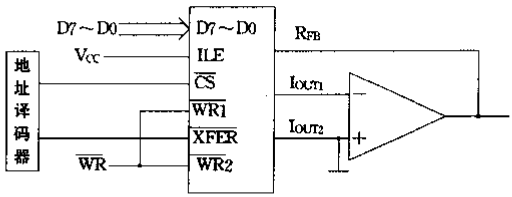
根据需要，单片机与 DAC0832 的接口可按二级缓冲器方式、单缓冲器方式和直通方式联接。

(1)单缓冲器联接方式接口电路

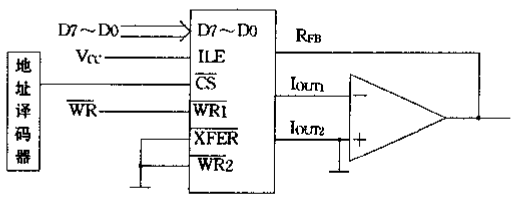
图 8-49 是 DAC0832 的单缓冲器连接方式与 8031 的接口电路。从图中得到 DAC0832 的输入寄存器和 DAC 寄存器的端口地址为 FEFFH。由于二个寄存器共用一个地址，当数据写入输入寄存器后，同时也写入了 DAC 寄存器，因此称为单缓冲联接方式。

下面是产生一个锯齿波信号的程序：

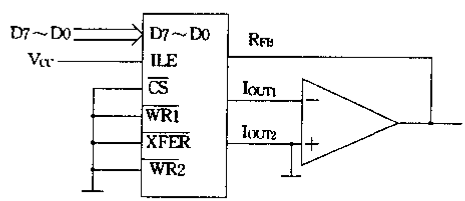
```
DIRE:  MOV   DPTR,#0FEFFH    ;DAC 寄存器的地址
       MOV   R0,#00H         ;输出数字量的初值
NEXT:  MOV   @DPTR,A
```



(a) 二级缓冲器联接方式



(b) 单级缓冲器联接方式



(c)直通联接方式

图8-48 DAC0832的联接方式

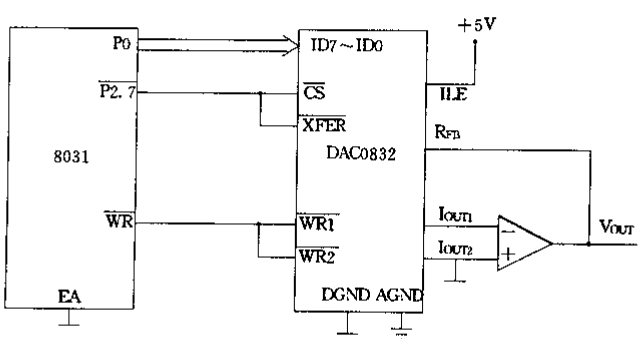


图8-49 8031与DAC0832单缓冲器联接方式的接口电路

```
INC     R0                  ;下一个数字量
NOP                         ;延时
NOP
NOP
AJMP    NEXT
```

(2)双缓冲器连接方式接口电路

图 8-50 是 DAC0832 双缓冲器连接方式与 8031 的接口电路，利用此电路可以输出一对同步信号，如从 X、Y 输出一组同步的锯齿波和正弦波信号。

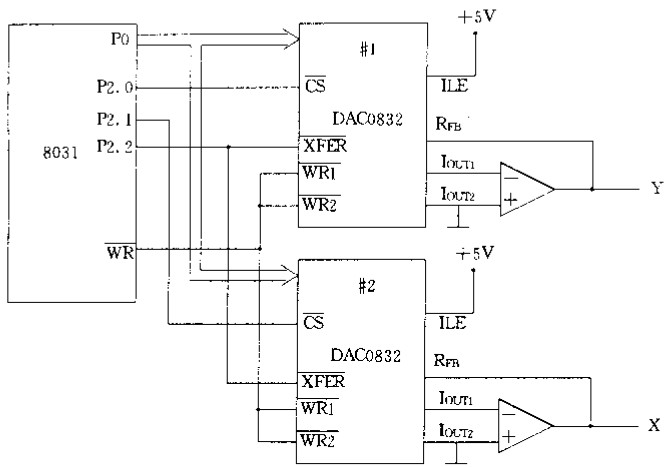


图8-50　8031与DAC0832双缓冲器联接方式的接口电路

由图 8-50，#1DAC0832 的输入寄存器的地址为 FEFFH，DAC 寄存器的地址为 FBFFH。#2DAC0832 的输入寄存器的地址为 FDFFH，DAC 寄存器的地址为 FBFFH。二个 DAC0832 的输入寄存地址不同，而 DAC 寄存器的地址相同，因此，可以将不同的数据分别写入二个输入寄存器，再同时将它们输入到 DAC 寄存器中，同时进行转换。

下面是从 X 和 Y 同时输出 0V 和 2.5V 电压的程序。

```
MOV     DPTR,#0FEFFH    ;置#1 输入寄存器地址
MOV     A,#80H          ;A←数字量
MOV     @DPTR,A         ;数字量送#1 输入寄存器
MOV     DPTR,#0FDFFH    ;置#2 输入寄存器地址
MOV     A,#00H
MOV     @DPTR,A
MOV     DPTR,#0FBFFH    ;置#1、#2DAC 寄存器地址
MOV     @DPTR,A         ;数据送 DAC 寄存器
```

二、单片机与A/D转换器的接口

A/D转换是把模拟量转换为数字量的过程。A/D转换的方法很多，如：频率法、双斜积分法、逐次逼近法等。其性能指标也很多，如：分辨率、转换时间、转换精度、电源、输出特性等。

1. A/D转换器(ADC)0809

ADC0809是一种典型的A/D转换器。它是采用逐次逼近方法的8位8通道A/D转换器。+5V单电源供电。转换时间在100μs左右。

ADC0809为28引脚，双列直插芯片，其引脚如图8-51所示。

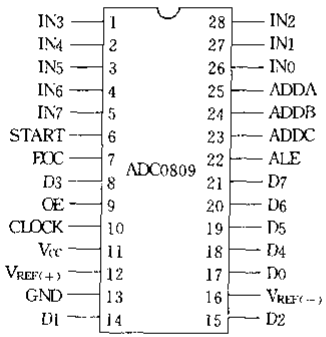


图8-51 ADC0809的引脚图

各引脚功能如下：

IN7～IN0 ：8路模拟量输入端；

D7～D0：8位数字量输出端口；

START：A/D转换启动信号输入端；

ALE：地址锁存允许信号，高电平有效；

EOC：转换结束信号，高电平有效；

OE：输出允许控制信号，高电平有效；

CLK：时钟信号输入端；

A、B、C：转换通道的地址；

$V_{REF(+)}$：参考电源的正端；

$V_{REF(-)}$：参考电源的负端；

V_{CC}：电源正端；

GND：地。

ADC0809由一个8位A/D转换器、一个8路模拟量开关、8路模拟量地址锁存译码器和一个三态数据输出锁存器组成，如图8-52所示。各通道的地址如表8-4所示。

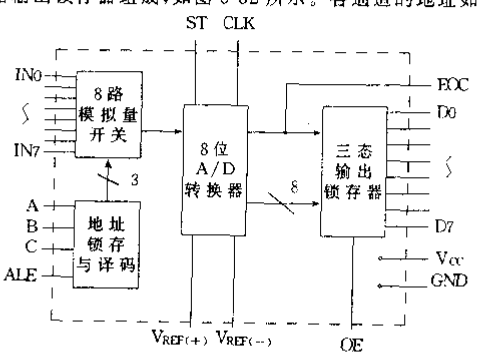


图8-52 ADC0809的内部结构

当ALE为高电平时，通道地址输入到地址锁存器中，下降沿将地址锁存，并译码。在START上跳沿时，所有的内部寄存器清0，在下降沿时，开始进行A/D转换，此期间

START 应保持低电平。需要注意的是，在 START 下降沿后 10μs 左右，转换结束信号 EOC 变为低电平，EOC 低电平时，表示正在转换，变为高电平时，表示转换结束。OE 为输出允许信号，控制三态输出锁存器输出数据，OE=0 时，输出数据端口线为高阻状态，OE=1，允许转换结果输出。

ADC0809 内部没有时钟电路，需外部提供时钟信号，CLK 为时钟信号输入端。

2. ADC0809 与 8031 的接口

图 8-53 为 ADC0809 与 8031 之间的接口电路。时钟信号由单片机的 ALE 信号 4 分频获得。由于 ADC0809 内部设有地址锁存器，所以通道地址由 P0 口的低 3 位直接与 ADC0809 的 A、B、C 相连。

表 8-4 通道地址

C	B	A	通道
0	0	0	IN0
0	0	1	IN1
0	1	0	IN2
0	1	1	IN3
1	0	0	IN4
1	0	1	IN5
1	1	0	IN6
1	1	1	IN7

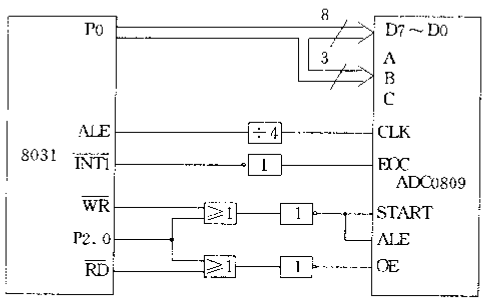


图 8-53 8031 与 ADC0809 的接口电路

START 信号和 OE 信号的逻辑表达式为

$$START = ALE = \overline{\overline{WR} + P2.0}$$

$$OE = \overline{\overline{RD} + P2.0}$$

启动一次转换过的控制信号之间的关系如图 8-54 所示。

读转换结果时，控制信号之间的关系如图 8-55 所示。

下面是启动一次转换，转换结束后，将结果读入单片机内部数据存储器的程序。

查询方式：

```
ADT:  MOV   A,#00H          ;设置通道地址
      MOV   DPTR,#0FEFFH    ;设置 ADC0809 的口地址
      MOVX  @DPTR,A         ;启动转换
      ACALL DLT             ;延时 10μs
WAIT: JB    P3.3,WAIT       ;等待转换结束
      MOVX  A,@DPTR         ;读入转换结果
      MOV   30H,A
      RET
DLT:  (延时 10μs 子程序)
```

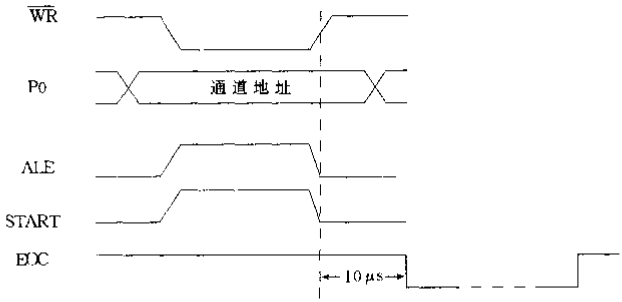


图 8-54　启动时控制信号的关系

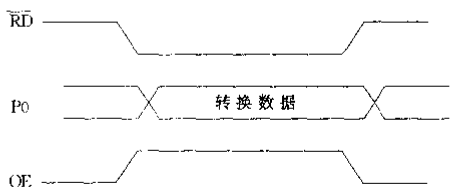


图8-55　读结果时控制信号的关系

中断方式：

主程序：

```
MAIN: SETB  IT1                    ;INT1下降沿触发
      SETB  EA                     ;开总中断
      SETB  EX1                    ;开外部中断 1
      MOV   DPTR,#0FEFFH
      MOV   A,#00H
      MOVX  @DPTR,A
        ⋮
```

中断服务程序：

```
ADINT: PUSH  DPL
       PUSH  DPH
       MOV   DPTR,#0FEFH
       MOVX  A,@DPTR
       MOV   30H,A
       MOV   A,#00H
       MOVX  @DPTR,A               ;启动下一次转换
       POP   DPH
       POP   DPL
       RET
```

思考题与习题

1. 简述 I/O 接口电路的基本功能，MCS-51 单片机 I/O 接口扩展的主要方法。

2. 8255A 有几种工作方式？它们的功能各是什么？

3. 当 8255A 的 A 口工作在方式 2 时，C 口各位的功能是什么？

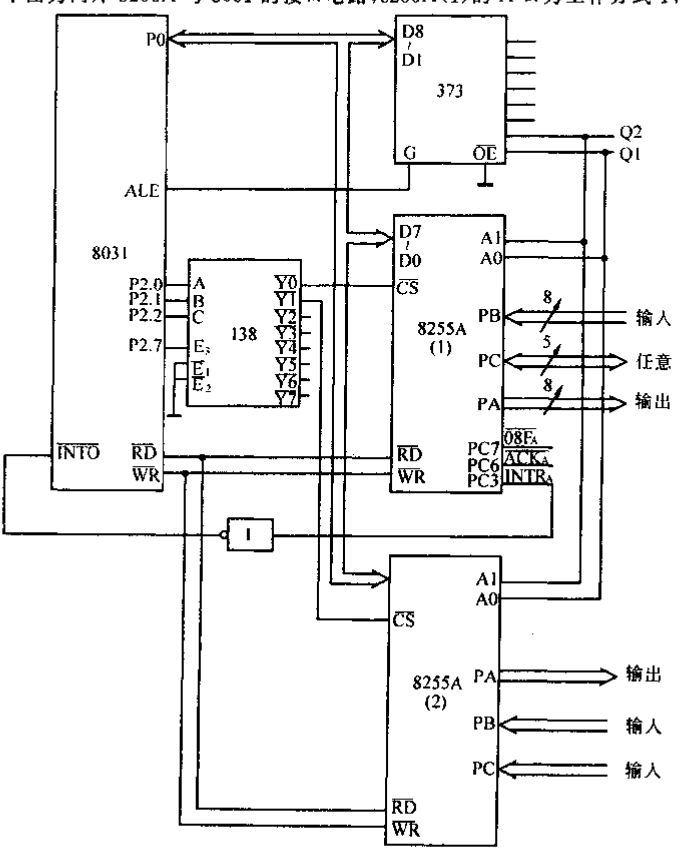
4. 若设 8255A 的 A 口为工作方式 2，B 口为工作方式 0 输入，C 口剩余口线为方式 0 输出，试确定工作方式控制字的内容。

5. 若将 8255A 的 PC.5 置位，写入按位操作寄存器的内容应是什么？

6. 设 8255A 与 8031 的接口电路如图 8-13 所示，试编写一段程序（包括 8255A 的初始化）从 PC.0 输出一个 100μs 的正脉冲。

7. 8255A 与 8031 之间的接口电路如图 8-14 所示，采用查询方式，当外设数据准备好时，从 A 口读入数据，存放在 8031 的内部数据存贮器中，试编写有关程序。

8. 下图为两片 8255A 与 8031 的接口电路，8255A(1)的 A 口为工作方式 1，试编写



下列程序：

① 按图中的要求，编写两片 8255A 的初始化程度；

② 采用中断方式从 8255A(1)的 A 口输出一个数据，同时从 B 口输出一个数据；

③ 采用查询方式，将从 8255A(2)的 PB 口和 PC 口输入的数据，由 8255A(1)的 A 口输出。

9. 8155 有几种工作方式？如何选择其工作方式？

10. 8155 定时器有几种工作方式？如何选择其工作方式？

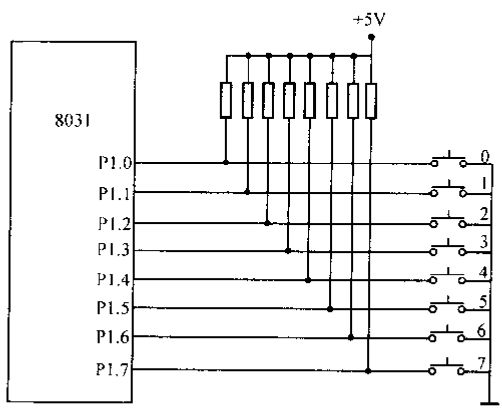
11. 设 8155A 口为选择输入，B 口为基本输出工作方式，允许 A 口中断，启动定时器计数，试确定 8155 命令寄存器的内容。

12. 8155 与 8031 之间的接口电路如图 8-21 所示。将从 8155A 口和 B 口输入的数据存放在其数据存贮器中，试编写程序。

13. 什么是抖动？为什么要消除抖动？消除抖动的方法有几种？

14. 在非编码键盘的程序扫描方式中，包括哪几部分工作？

15. 如下图所示，独立式按键的电路中的键号分别为 0～7，试编写程序扫描方式的键盘处理程序，键号放入 A 中。



16. 试设计一个 8031 通过 8155 扩展一个具有 16 键和 4 位数码管的键盘显示系统，并编写键盘处理子程序和显示子程序。

17. 简述 8279 的主要功能。

18. 8279 有几个命令字，它们的具体格式是什么？

19. 如何区分 8279 的命令字、状态字和数据？

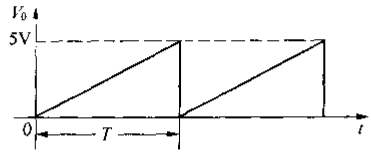
20. 设 8279 的工作状态为：8 个字符显示，右边输入、内部译码扫描键盘、双键互锁，对输入时钟信号 16 分频，试确定有关命令字的内容。

21. 用串行口扩展键盘显示器时，串行口应工作在什么状态？它有什么优、缺点？

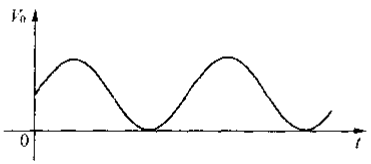
22. 什么是 A/D、D/A 转换，它们各有哪些技术指标，它们的分辨率是由什么决定的？

23. 8031 与 DAC0832 的接口电路如图 8-49 所示，试编写输出下列信号的程序：

①



②



24. 试编写由图 8-50 中的 X 和 Y 端输出一对同步信号，锯齿波和正弦波，如题 23 中的①和②。

25. 设 8031 与 ADC0809 的接口电路如图 8-53 所示，试编写从通道 IN0～IN7，循环采集数据的程序。

第九章　增强 51 单片机

前面几章我们主要介绍了 51 系列单片机的 51 子系列，随着单片机的研制发展，现在已经出现了许多功能更强的单片机，这些单片机都与 51 子系列硬件兼容，软件指令通用。本章主要讲述 8XC52/54/58，8XC51FX 及 87C51GB 单片机的硬件及增强功能。

第一节　8XC52/54/58 系列单片机硬件说明

前面我们介绍了 51 单片机片内硬件特性，它包括：

- 0K/4KROM(EPROM)；
- 128BRAM，128B 特殊功能寄存器(SFR)空间；
- 2 个 16 位定时器/计数器；
- 5 个中断源，两级中断优先级；
- 4 个并行 I/O 接口；
- 可编程全双工串行通信接口。

一、8XC52/54/58 系列新增功能介绍

8XC52/54/58 是基于 MCS-51 结构的高集成度 8 位微控制器，其主要特性是具有一个用于多处理器通信的增强型串行口和一个增/减定时器/计数器。因为该产品采用了 CHMOS 工艺，因而它具有两个软件可选择的节能方式：空闲方式和掉电方式。作为 MCS-51 系列的一员，8XC52/54/58 经过专门的优化，适用于控制应用。

因为它们不同于 8051，所以下面给出了 8XC52/54/58 硬件的新增功能说明：

- 256 字节片内 RAM
- 特殊功能寄存器(SFR)
- 定时器 2
  - ——捕捉定时器/计数器；
  - ——增/减定时器/计数器；
  - ——波特率发生器。
- 全双工可编程串行接口
  - ——成帧错误检测；
  - ——自动地址识别。
- 6 个中断源
- 增强型掉电方式

- 电源切断标志
- 在线仿真(ONCE)方式

8XC52/54/58 采用标准的 8051 指令系统，并与现有的 MCS-51 系列产品对应引脚兼容。表 9-1 给出了目前使用的 8XC52/54/58 器件名及各器件之间的存储器差异。

表 9-1 8XC52/54/58 器件名及存储器差异

有 ROM 的器件型号	有 EPROM 的器件型号	无 ROM 的器件型号	ROM/EPROM 字节数	RAM 字节数
80C52	87C52	80C32	8K	256
80C54	87C54	80C38	16K	256
80C58	87C58	80C32	32K	256

二、内部数据存储器

8XC52/54/58 系列内部数据存储器为 256B，地址范围为 00H ~ FFH，比 51 系列多 128 字节。同时内部还有 128B 空间的专用寄存器区(SFR)，地址范围为 80H ~ FFH，可见高 128B 单元的内部 RAM 地址与专用寄存器区的地址重叠，如图 9-1 所示。这些地址重叠的区域是靠不同的寻址方式来区别的。

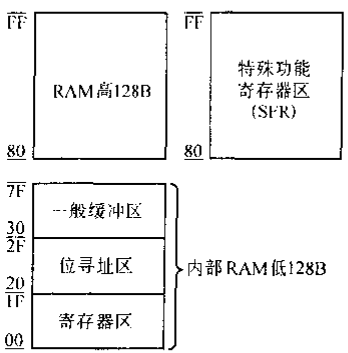


图 9-1 内部存储器地址空间分布

所有的内部 RAM 低 128 字节都可以用直接寻址或间接寻址来访问，高 128 字节 RAM 只能通过间接寻址访问，而专用寄存器区 SFR 只能通过直接寻址方式访问。

当一条指令访问内部单元时，若地址高于 7FH，则 CPU 会知道是访问数据 RAM 的高 128 字节还是访问 SFR 空间。例如：

MOV 0A0H, # DATA

因为是使用直接寻址的指令，故访问 SFR 空间的 0A0H 单元(P2)。又如：设 R0 的内容为 0A0H，则

MOV @R0, # DATA

是采用间接寻址方式，故访问数据 RAM 高端 0A0H 地址单元，而不是 SFR 中的 P2。

应该注意，堆栈操作也是间接寻址，因此，数据 RAM 的高 128 字节也可用作堆栈空间。

8XC52/54/58 系列的 SFR 中除了含有 51 系列的所有 19 个专用寄存器外，还增加了 6 个专用寄存器，用于定时器 2 及串行口的控制，如表 9-2 所示。

表 9-2

特殊功能寄存器	功能名称	地　址	复位后状态
T2CON	定时/计数器 2 控制寄存器	C8H	00H
T2MOD	定时/计数器方式控制寄存器	C9H	00H
RCAP2L	重装寄存器低 8 位	CAH	00H
RCAP2H	重装寄存器高 8 位	CBH	00H
TL2	T2 计数器低 8 位	CCH	00H
TH2	T2 计数器高 8 位	CDH	00H
SADDR	个别地址寄存器	A9H	00H
SADEN	个别地址屏蔽寄存器	B9H	00H

三、8XC52/54/58 系列定时器 2 原理

定时器 2 是一个 16 位定时器/计数器，与定时器 0 和 1 一样，它可以作为一个定时器或事件计数器，这可通过特殊功能寄存器 T2CON 中位 $C/\overline{T}$ 选择。它有三种工作方式："捕捉"、"自动重装"和"波特率发生器"，这是由 T2CON 中的控制位来选择。T2CON 是可以位寻址的寄存器。各位功能如表 9-3 所示。三种工作方式的选择见表 9-4。

表 9-3　T2CON：定时器 2 控制寄存器

T2CON 地址 = 0C8H　　复位值 = 0000 0000B

位可寻址

TF2	EXF2	RCLK	TCLK	EXEN2	TR2	CP/$\overline{T2}$	CP/$\overline{RL2}$
7	6	5	4	3	2	1	0

符号	功　能
TF2	定时器 2 溢出标志，由定时器 2 溢出置位，并必须由软件清零。当 RCLK = 1 或 TCLK = 1 时 TF2 将不被置位
EXF2	当 EXEN2 = 1 和 T2EX 负跳变引起捕捉或重装时，定时器 2 外部标志置位。当定时器 2 中断处于允许状态时，EXF2 = 1 将使 CPU 指向定时器 2 的中断程序。EXF2 必须由软件清零。在增/减计数方式(DCEN = 1)中 EXF2 并不引起中断
RCLK	接收时钟控制。置位时，在串行口方式 1 和 3 中使串行口用定时器 2 的溢出脉冲作为其接收时钟。RCLK = 0 用定时器 1 的脉冲作接收时钟
TCLK	发送时钟控制。置位时，在串行口方式 1 和 3 中使串行口用定时器 2 的溢出脉冲作为其发送时钟。TCLK = 0 用定时器 1 的溢出脉冲作发送时钟
EXEN2	定时器 2 外部控制。置位时，若定时器 2 不被用作同步串行口的时钟，则作为 T2EX 上负跳变的结果允许捕捉或重装发生。EXEN2 = 0 使得定时器 2 忽略 T2EX 上的事件
TR2	定时器 2 启动/停止控制。TR2 = 1 启动定时器
CP/$\overline{T2}$	定时器 2 的定时或计数功能选择。CP/$\overline{T2}$ = 0 时用作定时功能。CP/$\overline{T2}$ = 1 时用作外部事件计数器(下降沿触发)
CP/$\overline{RL2}$	捕捉/重装选择。如 EXEN2 = 1，则 CP/$\overline{RL2}$ = 1 会使捕捉发生在 T2EX 为负跳变处。当 EXEN2 = 1，定时器 2 溢出，或在 T2EX 发生负跳变时，CP/$\overline{RL2}$ = 0 导致自动重装发生。当 RCLK = 1 或 TCLK = 1 时，该位被忽略，在定时器 2 溢出时使它自动重装

表 9-4 定时器 2 工作方式

RCLK + TCLK	CP/$\overline{RL2}$	TR2	方式
0	0	1	16 位自动重装
0	1	1	16 位捕捉
1	×	1	波特率发生器
×	×	0	(关闭)

在 8XC52/54/58 系列单片机中,除了 P3 口以外,P1 口的两条线也具有第二种功能:

P1.0 T2 (定时器/计数器 2 外部输入);

P1.1 T2EX(定时器/计数器 2 捕捉/重装触发引脚)。

1.捕捉方式

在捕捉方式下,有两种选择,它由 T2CON 中的位 EXEN2 决定。若 EXEN2 = 0,则定时器 2 是一个 16 位定时器/计数器,当它产生溢出时就使 TF2(定时器 2 的溢出位)置位,这就可用来产生一个中断。若 EXEN2 = 1,则定时器 2 仍如上工作,但有一个附加功能,即外部输入端 T2EX 的一个 1 到 0 的跳变能把定时器 2 寄存器对 TL2 和 TH2 当前值捕捉到寄存器对 RCAP2L 和 RCAP2H 中去。此外,T2EX 上的跳变还使 T2CON 中的位 EXF2 置位,且 EXF2 像 TF2 一样能产生中断。捕捉方式如图 9-2。

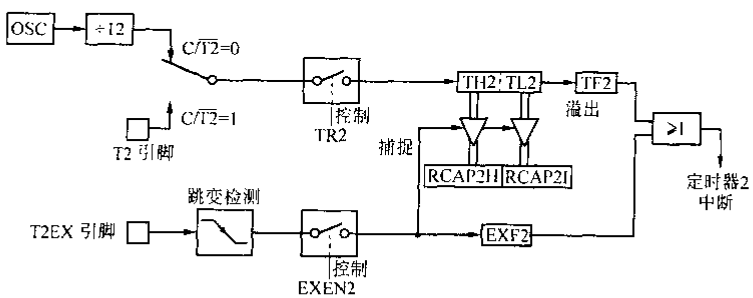


图 9-2 捕捉方式下的定时器 2

2.自动重装方式

在 16 位自动重装方式中,定时器 2 可以编程为增或减计数。该特性受 T2MOD 中的 DCEN 位(减计数器控制位)控制,见表 9-5 所示。复位后,DCEN 为 0,这时定时器 2 将缺省为增计数;当 DCEN 置位时,定时器 2 可以增计数也可以减计数,这取决于 T2EX 引脚的值。

图 9-3 所示为 DCEN = 0 时定时器 2 自动增计数。在这种方式下,有两种选择方式,它由 T2CON 中的位 EXEN2 决定。若 EXEN2 = 0,则当定时器 2 溢出时不仅置位 TF2,还使得定时器 2 寄存器(TL2 和 TH2)重装寄存器 RCAP2L 和 RCAP2H 中由软件预置的值。若 EXEN2 = 1,则定时器 2 仍做上述工作,但还有一个附加功能,即外部输入端 T2EX 上的一个 1 到 0 的跳变也将触发该定时器 2 重新装载并使 EXF2 置位。

表 9-5　T2MOD:定时器 2 控制寄存器

T2MOD 地址=0C9H　　复位值=XXXX XX00B
位不可寻址

位	—	—	—	—	—	—	T2OE	DCEN
	7	6	5	4	3	2	1	0

符　号	功　能
—	未实现，保留供以后用。
T2OE	定时器2输出允许位。
DCEN	置位时，该位允许定时器2构成一个增/减计数器。

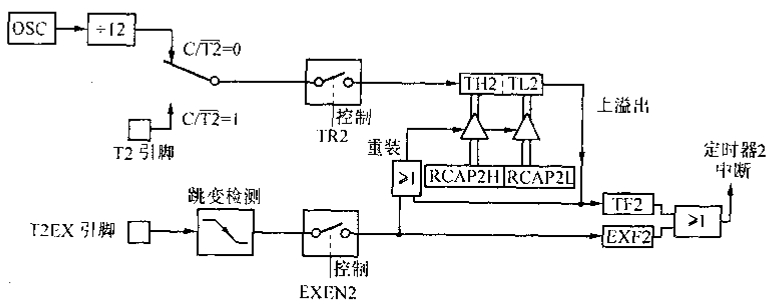


图 9-3　自动装载方式下的定时器 2(DCEN = 0)

置位 DCEN 使定时器 2 能够增计数或减计数，如图 9-4 所示。在这种方式下，T2EX 引脚控制计数的方向。T2EX = 1 使定时器 2 按增量方式计数，则当定时器 2 向溢出出时置

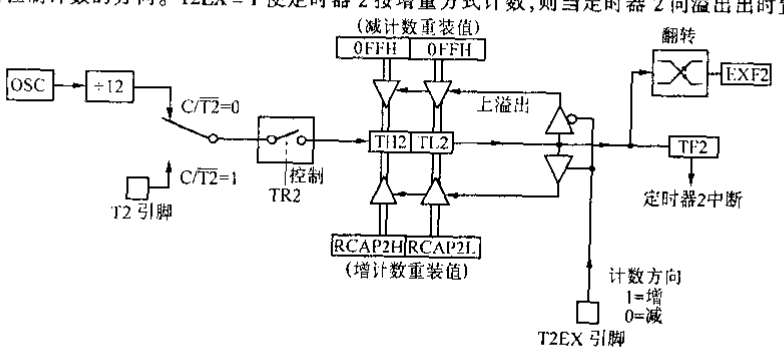


图 9-4　自动装载方式下的定时器 2(DCEN = 1)

位 TF2,同时使得定时器 2 寄存器(TL2 和 TH2)重装寄存器 RCAP2L 和 RCAP2H 中由软件预置的值;T2EX = 0 使定时器 2 按减量方式计数,当 TH2 和 TL2 与存储在 RCAP2H 和 RCAP2L 中的值相等,定时器向下溢出,置位 TF2,并使 0FFFFH 重装入定时器 2 寄存器中。

EXF2 位在定时器 2 向溢出出或向下溢出时翻转。如果需要的话,该位可用作定时器分辨率的第 17 位。在这种方式下,EXF2 并不产生中断。

3.波特率发生器

在 8XC52/54/58 系列单片机中,置位 T2CON 中的 TCLK 和/或 RCLK 选择定时器 2 工作为波特率发生器。注意,同时进行的发送和接收波特率可以不一样。其实现可采用这样的方法:定时器 2 用于发送器或接收器,定时器 1 用于另一功能。

波特率发生器方式类似于自动重装方式,T2 溢出使得定时器 2 寄存器重装 RCAP2H 和 RCAP2L 寄存器中由软件预置的 16 位值。

串行口工作于方式 1 和方式 3 由定时器的溢出速率决定串行通信的波特率,若使用定时器 2 作波特率发生器,则

$$\text{波特率} = \frac{\text{定时器 2 溢出速率}}{16}$$

定时器可设置成"定时器"或"计数器"工作。在大多数典型应用中,它可设置成"定时器"工作($C/\overline{T2}=0$)。"定时器"工作和用作波特率发生器的定时器 2 稍有不同。一般来说,定时器每一个机器周期(振荡频率的 1/12)加 1,而波特率发生器每个状态时间(1/2 的振荡器频率)加 1。这时,波特率由如下公式给出:

$$\text{波特率} = \frac{\text{振荡器频率}}{32 \times [65536 - (\text{RCAP2H,RCAP2L})]}$$

这里,(RCAP2H,RCAP2L)为 RCAP2H 和 RCAP2L 中的 16 位无符号整数。

定时器 2 用作波特率发生器如图 9-5 所示。该图仅在 T2CON 的 TCLK + RCLK = 1 时有效。注意,TH2 翻转并不置位 TF2,且不产生中断。因此,在定时器 2 处于波特率发生器方式时不必禁止定时器 2 中断。还必须注意,若 EXEN2 置位,则 T2EX 上 1 到 0 的跳变将置位于 EXF2,但并不导致(RCAP2H,RCAP2L)到(TH2,TL2)的重装。因此,在定时器 2 用波特率发生器时,如需要的话,T2EX 可用作一个额外的外部中断。

应该注意的是,定时器 2 在波段特率发生器方式下正运行(TR2 = 1)"定时器"功能时,不要试图去读或写 TH2 或 TL2。在这些条件下,定时器每个状态时间加 1,读或写所得结果可能会不准确。RCAP2 寄存器可读,但不可以写,因为写可能会覆盖重装并导致写和/或重装出错。在这种情况下,应在访问定时器 2 或 RCAP2 寄存器之前关闭定时器(清零 TR2)。

4.可编程时钟输出

在 P1.0 引脚可以编程产生占空比为 50% 的时钟。该引脚除了用作正常的 I/O 外,它还有另外的两个功能,通过编程:

(1)可以为定时器/计数器 2 输入外部时钟信号。

(2)作为输出引脚,输出占空比为 50% 的时钟信号。在 16MHz 工作频率下,输出信号频率从 61Hz 到 4MHz。

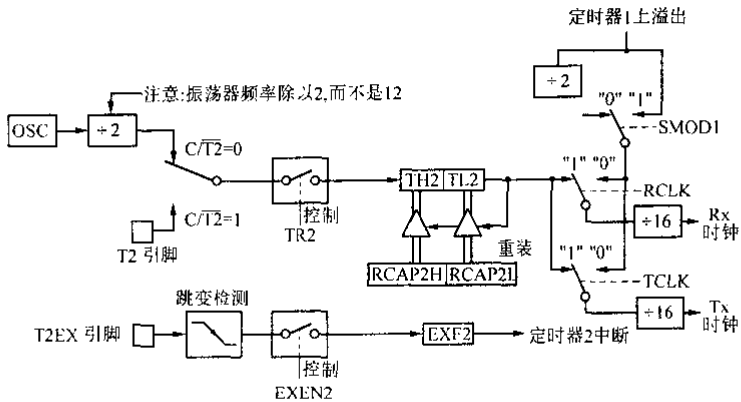


图 9-5　波特率发生器方式下的定时器 2

为了将定时器/计数器 2 构成一个时钟发生器，C/$\overline{T2}$(T2CON.1)位必须清零，位 T2OE (T2MOD.1)则必须置位。位 TR2(T2CON.2)则用来启动或停止定时器。时钟输出频率取决于振荡器频率和定时器 2 捕捉寄存器(RCAP2H,RCP2L)的重装值,其计算公式如下：

$$\text{时钟输出频率} = \frac{\text{振荡器频率}}{4 \times [65536 - (\text{RCAP2H,RCAP2L})]}$$

在时钟输出方式,定时器 2 溢出并不产生一个中断。这点类似于定时器 2 用作波特率发生器时的情况。这就使得将定时器同时用作波特率发生器和时钟发生器成为可能。但应注意,波特率和时钟输出频率并不能相互独立地确定,因为它们均使用 RCAP2H 和 RCAP2L。

四、UART(通用异步收发器)

8XC5X 中的 UART 和 8051 中的 UART 的工作情况完全相同,但有如下改进：

成帧错误检测——成帧错误检测允许串行口在方式 1、2 或 3 下检验有效停止位。举例来说,串行线上的噪声或两个 CPU 同时发送会引起停止位丢失。

如果停止位丢失,则成帧错误(FE)置位。每次接收后,可用软件校验 FE 位,以便检测通信出错。一旦置位,则必须用软件给 FE 清零。有效停止位并不清 FE。

FE 位在 SCON 中,和 SM0 位共享相同的位地址。PCON 寄存器(于 PCON.6)中的控制位 SMOD0 决定是访问 SM0 位还是访问 FE 位。若 SMOD0 = 0,则访问 SCON.7 是针对 SM0 的;若 SMOD0 = 1,则访问 SCON.7 是针对 FE 的。

自动地址识别——自动地址识别减少了为串行口服务所需的 CPU 时间,因为 CPU 仅在接收到其自身的地址时中断,所以避免了比较地址所需的软件开销。若在 9 位方式之一中允许了这样的特性,则当接收的字节相同于给定地址或广播地址时,接收中断(RI)标志才置位。

在多处理器系统中使用这个特性的方法如下：

当一个主处理器要发送一个数据块到几个从处理器之一时，它首先发送一个标识目标从处理器的地址字节。记住，一个地址字节的第 9 位置为 1，而数据字的第 9 位置为 0。所有的从处理器应该将它们的 SM2 位置 1，以便它们仅被一个地址字节中断。自动地址识别特性仅允许被寻址的从处理器被中断。在这种情况下，地址比较由硬件完成，而非由软件完成(在 8051 串行口中，一个地址字节中断所有从处理器来进行地址比较)。

然后，被寻址的从处理器将其 SM2 位清零，并且准备接收将要到来的数据字节，其它从处理器不受这些数据字节的影响，因为它们仍处在等待接收地址字节的状态。

在 8 位方式(方式 1)和在 9 位方式下的主要工作情况相同，不同之处是停止位占了第九个数据位的位置。如果 SM2 被置位，仅在所接收到的字节与给定字节或广播字节匹配，并由一个有效停止位所终止时，RI 标志才置位。设置 SM2 位对方式 0 并无影响。

利用给定地址主处理器可以有选择地和一组从处理器通信。在广播地址的情况下，一次可以寻址所有的从处理器。利用两个特殊功能寄存器：SADDR 和 SADEN 为每一个从处理器定义这些地址。

从处理器的个别地址在 SADDR 中被指定。SADEN 则是一个屏蔽字节，它定义无关位，从而构成给定地址。这些无关位给用户规定同一时刻访问一个或多个从处理器的协议带来一定的灵活性。以下是一个用户如何定义给定地址，从而有选择地访问不同从处理器的例子。

从处理器 1：

```
SADDR     = 1111 0001
SADEN     = 1111 1010
给定地址  = 1111 0x0x
```

从处理器 2：

```
SADDR     = 1111 0001
SADEN     = 1111 1001
给定地址  = 1111 0xx1
```

SADEN 位经过选择而定，这样可以分别访问每个从处理器。注意，位 0(LSB)对于从处理器 1 给定地址来说它是无关的位，但对从处理器 2 的给定地址而言，位 0 等于 1。因此，为了有选择地只和从处理器 1 通信，主处理器则必须发送一个位 0 等于 0 的地址(如 1111 0000)。

类似地，位 1 等于 0 是对从处理器 1 的给定地址而言，但它对从处理器 2 来说是无关的位。因此，如果现在跟从处理器 2 通信，则必须使用位 1 等于 1 的地址 (如 1111 0111)。

最后，对于一个主处理器与两个从处理器同时通信的情况，地址必须有位 0 等于 1 和位 1 等于 0。但应注意，对于两个从处理器的给定地址来说，位 2 是无关。从而允许用两个不同的地址来选择两个从处理器(1111 0001 或 1111 0101)。如果加入第三个从处理器，需要其位 2 等于 0，则后一个地址可用来与从处理器 1 和 2 但不与从处理器 3 通信。

主处理器还可以同时与所有从处理器用广播地址通信。它是这样构成的，由 SADDR 和 SADEN 寄存器做逻辑或运算，0 定义为无关位。无关位为广播地址位提供一定的灵活

性,但在大多数应用中,广播地址为0FFH。

SADDR 和 SADEN 分别位于0A9H 和0B9H 处。复位时,SADDR 和 SADEN 寄存器被初始化为 00H,它定义给定地址和广播地址为 × × × × × × × ×(所有无关位),这样确保 8XC5X 串行口向下与其它并未实现自动地址识别的 MCS-51 产品兼容。

五、8XC5X 系列中断结构

8XC5X 系列提供 6 个中断源,比 51 系列增加了一个中断,这个中断是由 TF2 和 EXF2 的逻辑或所产生。在 CPU 转向中断服务程序时硬件并不对这两个标志清零。事实上,服务程序也许必须确定是 TF2 还是 EXF2 产生的中断,然后由软件清零。

6 个中断源的允许位由 IE 寄存器控制。IE 的各位功能如下:

EA	—	ET2	ES	ET1	EX1	ET0	EX0

ET2(IE.5)定时器 2 中断允许控制位,当 ET2 = 1,允许定时器 2 中断,ET2 = 0,禁止定时器 2 中断。

6 个中断源中断优先级由 IP 决定,IP 各位功能如下:

—	—	PT2	PS	PT1	PX1	PT0	PX0

PT2(IP.5)定时器 2 中断优先级位。PT2 = 1 为高优先级;PT2 = 0 为低优先级。8XC5X 系列内部优先级查询顺序如表 9-6。

表 9-6 8XC5X 系列内部优先级查询顺序及矢量地址

中断源	优先级	矢量地址
外部中断 0	最高	0003H
定时器 0	↓	000BH
外部中断 1	↓	0013H
定时器 1	↓	001BH
串行口	↓	0023H
定时器 2	最低	002BH

6 个中断源的中断矢量地址也如表 9-6 所示。

第二节 8XC51FX 硬件说明

一、引言

8XC51FX 是基于 MCS-51 结构的高集成度 8 位微控制器。作为 MCS-51 系列的一员,8XC51FX 经过优化,适合于控制应用场合。其主要特征是具有可编程计数器阵列(PCA),它可以在 5 个 I/O 引脚上测量和产生脉冲信息。另外,它还包括用于多处理器通信的增强型串行口、增/减定时器/计数器和一个片内程序储存器的程序保密方案。由于 8XC51FX 产品采用了 CHMOS 工艺,因而具有两个软件可选择的节电方式:空闲方式和掉

电方式。

8XC51FX 采用了标准的 8051 指令系统，且和已有的 MCS-51 系列产品对应引脚完全兼容。表 9-7 总结了当前常用的 8XC51FX 产品名称和存储器差异。在本文中，产品一般指 C51FX。

表 9-7 C51FX 系列单片机

有 ROM 的器件	有 EPROM 的器件	无 ROM 的器件	ROM/EPROM 字节	RAM 字节
83C51FA	87C51FA	80C51FA	8K	256B
83C51FB	87C51FB	80C51FB	16K	256B
83C51FC	87C51FC	80C51FC	32K	256B

请注意，8XC51FX 并不包括 80C51FA 和 83C51FA。因为这两种器件并不具有 8XC51FX 上所具有的某些特征：可编程时钟输出、四级中断的优先级结构、改进的程序保密方案和异步端口复位。

8XC51FX 系列单片机主要特性如下：

- 4 个 8 位双向并行端口
- 3 个 16 位定时器/计数器
  ——一个增/减定时器/计数器；
  ——时钟输出。
- 可编程计数器阵列
  ——比较/捕捉；
  ——软件定时器；
  ——高速输出；
  ——脉宽调制器；
  ——监视定时器。
- 全双工可编程串行口
  ——成帧错误检测；
  ——自动地址识别。
- 中断结构
  ——7 个中断源；
  ——4 个优先级。
- 节电方式
  ——空闲方式；
  ——掉电方式。

二、特殊功能寄存器

片内存储器中 SFR（特殊功能寄存器）见表 9-8 所示。

- 可编程计数器阵列(PCA)寄存器

16 位的 PCA 定时器由寄存器 CH 和 CL 组成，寄存器 CCON 和 CMOD 含有 PCA 的控制位和状态位，CCAPMn(n=0，1,2,3,4)寄存器控制 5 个 PCA 模块的工作方式，寄存器对(CCAPnH，CCAPnL)是每一个 PCA 模块的 16 位比较/捕捉寄存器。

● 中断寄存器

IE 寄存器中包含了各中断的控制位，7 个中断源中的每一个可由 IP 及 IPH 组合成四个优先级之一。

● 电源控制寄存器

PCON 控制节电的方式：空闲方式和掉电方式。

表 9-8 C51FX 系列 SFR 映象和复位值

F8		CH 00000000	CCAP0H xxxxxxxx	CCAP1H xxxxxxxx	CCAP2H xxxxxxxx	CCAP3H xxxxxxxx	CCAP4H xxxxxxxx		FF
F0	* B 00000000								F7
E8		CL 00000000	CCAP0L xxxxxxxx	CCAP1L xxxxxxxx	CCAP2L xxxxxxxx	CCAP3L xxxxxxxx	CCAP4L xxxxxxxx		EF
E0	* ACC 00000000								E7
D8	CCON 00x00000	CMOD 00xxx000	CCAPM0 x0000000	CCAPM1 x0000000	CCAPM2 x0000000	CCAPM3 x0000000	CCAPM4 x0000000		DF
D0	* PSW 00000000								D7
C8	T2CON 00000000	T2MOD xxxxxx00	RCAP2L 00000000	PCAP2H 00000000	TL2 00000000	TH2 00000000			CF
C0									C7
B8	* IP x0000000	SADEN 00000000							BF
B0	* P3 11111111							IPH x0000000	B7
A8	* IE 00000000	SADDR 00000000							AF
A0	* P2 11111111								A7
98	* SCON 00000000	* SBUF xxxxxxxx							9F
90	* P1 11111111								97
88	* TCON 00000000	* TMOD 00000000	* TL0 00000000	* TL1 00000000	* TH0 00000000	* TH1 00000000			8F
80	* P0 11111111	* SP 00000111	* DPL 00000000	* DPH 00000000				* PCON 00xx0000	87

* = 可以在 8051 芯片中找到的 SFR(见 8051 硬件说明中的 SFR 解释)。x = 未定义。

三、端口结构和操作

C51FX 所有四个端口均是双向的，第一个端口由一个锁存器(特殊功能寄存器 P0-P3)、一个输出驱动器和一个输入缓冲器组成。

端口 0 和 2 的输出驱动器与端口 0 的输入缓冲器用于访问外部存储器。在这种应用中，端口 0 输出外部存储器地址的低字节，又读写外部存储器的数据字节，完成这些操作是靠分时使用端口 P0。当地址长度为 16 位时，端口 2 输出外部存储器地址的高位字节。其它情况下，端口 2 仍发送 P2 口 SFR 中 P2 的内容。

端口 1 和端口 3 的所有引脚都是多功能的，它们不仅仅是端口的引脚，它们还用于各种各样的特殊功能，如表 9-9 所示。

表 9-9 C51FX 端口的第二功能

端口管脚	第二功能
P0.0/AD0 ~ P0.7/AD7	外部存储器的地址/数据复用字节
P1.0/T2	定时器 2 的外部时钟输入/时钟输出
P1.1/T2EX	定时器 2 的重装/捕捉/计数方向控制
P1.2/ECI	PCA 外部时钟输入
P1.3/CEX0	PCA 模块 0 捕捉输入/比较/PWM 输出
P1.4/CEX1	PCA 模块 1 捕捉输入/比较/PWM 输出
P1.5/CEX2	PCA 模块 2 捕捉输入/比较/PWM 输出
P1.6/CEX3	PCA 模块 3 捕捉输入/比较/PWM 输出
P1.7/CEX4	PCA 模块 4 捕捉输入/比较/PWM 输出
P2.0/A8 ~ P2.7/A15	外部存储器地址的高位字节
P3.0/RXD	串行口输入
P3.1/TXD	串行口输出
P3.2/$\overline{INT0}$	外部中断 0
P3.3/$\overline{INT1}$	外部中断 1
P3.4/T0	定时器 0 外部时钟输入
P3.5/T1	定时器 1 外部时钟输入
P3.6/$\overline{WR}$	外部存储器写选通
P3.7/$\overline{RD}$	外部存储器读选通

仅当锁存在端口特殊功能寄存器的相应位内容为 1 时，其第二功能才有效。否则，该端口引脚锁定为 0。

四、可编程计数器阵列

可编程计数器阵列(PCA)由一个 16 位定时器/计数器和 5 个 16 位比较/捕捉模块组成，如图 9-6 所示。PCA 定时器/计数器用作 5 个模块的公共时基，也是唯一服务于 PCA 的定时器。其时钟输入信号可以编程为计数下列信号的任意一个：

- 振荡器频率 ÷ 12
- 振荡器频率 ÷ 4
- 定时器 0 溢出
- ECI 上外部输入信号(P1.2)

每一个比较/捕捉模块可以编程为下列方式的任意一种：

- 上升和/或下降沿捕捉
- 软件定时器
- 高速输出
- 脉冲宽度调制器

模块 4 还可以编程为监视定时器。

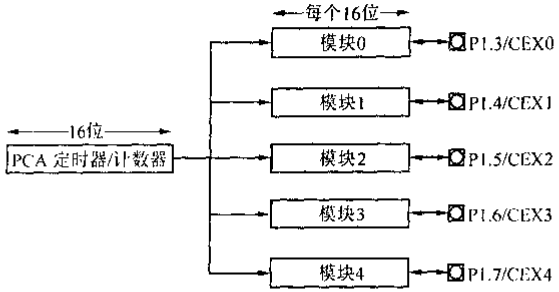


图 9-6　可编程计数器阵列(PCA)

当比较/捕捉模块被编程为捕捉方式、软件定时器或高速输出方式，在模块执行它的功能时将产生一个中断。所有 5 个模块加上 PCA 计数器溢出共用一个中断向量(更详细的说明见 PCA 中断部分)。

PCA 定时器/计数器和比较/捕捉模块共用端口 1 作为外部引脚。

1. PCA　16 位定时器/计数器

PCA 有一个独立运行的 16 位定时器/计数器，它由 CH 和 CL(计数值的高位字节和低位字节)组成。在任何时候，这两个寄存器均能读或写。图 9-7 为该定时器的方框图。时钟输入信号可以从以下 4 种方式中选取。

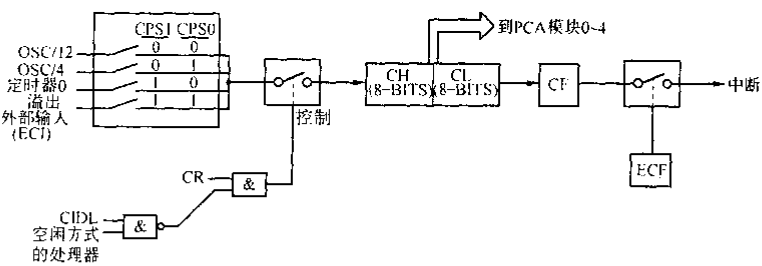


图 9-7　PCA 定时器/计数器

● 振荡器频率 ÷ 12

在每个机器周期的 S5P2，CL 寄存器递增一次。若采用 16MHz 晶振，则每 750ns 定时器递增一次。

● 振荡器频率 ÷ 4

在每个机器周期的 S1P2、S3P2 和 S5P2，CL 寄存器递增一次。若采用 16MHz 晶振，则每 250ns 定时器递增一次。

● 定时器 0 溢出

当定时器 0 向上溢出时，则在该机器周期的 S5P2，CL 寄存器递增一次。该方式允许一个可编程输入频率到 PCA。

● 外部输入

在 ECI 引脚(P1.2)上检测到一个 1 到 0 跳变之后的第一个 S1P2、S3P2 和 S5P2,CL 寄存器递增一次。在每个机器周期的 S1P2、S3P2 和 S5P2 对 P1.2 进行采样。这种方式下的最大输入频率为振荡器频率的八分之一。

当 CL 溢出时,两个振荡器周期后 CH 递增一次。

方式寄存器 CMOD 包含计数脉冲选择位(CPS1)和(CPS0),用来指定时钟输入。CMOD 如表 9-10 所示。该寄存器还包含 ECF 位,它允许 PCA 计数器溢出以产生 PCA 中断。此外,在空闲方式期间用户可以通过置位计数器空闲位(CIDL)来选择是否关闭 PCA 定时器。监视器定时器控制位(WDTE)在以后部分讨论。

CCON 寄存器,如表 9-11 所示,含有另外两位和 PCA 计数器/计数器有关的位。位 CF 在计数器溢出时由硬件置位,对位 CR 置位或清零则会打开或关闭计数器。该寄存器的其它 5 位是比较/捕捉模块的事件标志,这将在下面部分讨论。

表 9-10 CMOD:PCA 计数器方式寄存器

CMOD 地址 = 0D9H　　复位值 = 00× × × ×000B

位不可寻址

	CIDL	WDTE	—	—	—	CPS1	CPS0	ECF
位	7	6	5	4	3	2	1	0

符号	功　能
CIDL	计数器空闲控制:CIDL = 0 使 PCA 计数器编程为在空闲方式下继续计数,CIDL = 1 使在空闲方式下计数器停止计数
WDTE	监视器定时器控制,WDTE = 0 禁止 PCA 模块 4 的监视器定时器功能,WDTE = 1 允许该功能
—	未实现,保留供以后使用
CPS1	PCA 计数脉冲选择位 1
CPS0	PCA 计数脉冲选择位 0

CPS1	CPS0	选择 PCA 输入
0	0	内部时钟,$F_{OSC} \div 12$
0	1	内部时钟,$F_{OSC} \div 4$
1	0	定时器 0 溢出出
1	1	ECI/P1.2 管脚上外部时钟(最大频率 = $F_{OSC} \div 8$)

ECF	PCA 控制计数器溢出中断,EFC = 1 允许 CCON 中的 CF 位产生一个中断;ECF = 0 禁止 CF 的该功能

F_{OSC} = 振荡器频率

表 9-11 CCON:PCA 计数器控制寄存器

CCON 地址 = 0D8H　　复位值 = 00×0 0000B

位可寻址

	CF	CR	—	CCF4	CCF3	CCF2	CCF1	CCF0
位	7	6	5	4	3	2	1	0

符号	功　能
CF	PCA 计数器溢出标志。计数器溢出时由硬件置位，若 CMOD 中 ECF 位置位，则 CF 标志着一个中断，CF 位可以由硬件或软件置位，但仅由软件清零
CR	PCA 计数器运行控制位。软件置位时打开 PCA 计数器，必须由软件清零才能关闭 PCA 计数器
—	未实现，保留供以后使用
CCF4	PCA 模块 4 中断标志。当匹配或捕捉发生时由硬件置位，但必须由软件清零
CCF3	PCA 模块 3 中断标志。当匹配或捕捉发生时由硬件置位，但必须由软件清零
CCF2	PCA 模块 2 中断标志。当匹配或捕捉发生时由硬件置位，但必须由软件清零
CCF1	PCA 模块 1 中断标志。当匹配或捕捉发生时由硬件置位，但必须由软件清零
CCF0	PCA 模块 0 中断标志。当匹配或捕捉发生时由硬件置位，但必须由软件清零

2. 捕捉/比较模块

5 个比较/捕捉均有 6 种可以执行的功能：

- 16 位捕捉，正边沿触发；
- 16 位捕捉，负边沿触发；
- 16 位捕捉，正边沿和负边沿触发；
- 16 位软件定时器；
- 16 位高速输出；
- 8 位脉冲宽度调制器。

此外，模块 4 还可以用作监视定时器。这些模块可以编程为任意不同方式的组合。

每一个模块都有一个方式寄存器 CCAPMn(n = 0、1、2、3 或 4)，用来选择模块执行的功能。CCAPMn 寄存器如表 9-12 所示。注意，ECCFn 位是在模块事件标志位置位时允许 PCA 的 CCFn 产生中断。事件标志(CCFn)位于 CCON 寄存器中，当一个给定模块在捕捉到一个事件、软件定时器或高速输出事件发生时 CCFn 模块置位。

表 9-12 CCAPMn:PCA 模块比较/捕捉寄存器

CCAPMn 地址 = 0D8H (n = 0 ~ 4)

CCAPM0 0DAH
CCAPM1 0DBH
CCAPM2 0DCH
CCAPM3 0DDH
CCAPM4 0DEH

复位值 = 00 × 0 0000B

位不可寻址

—	ECOMn	CAPPn	CAPNn	MATn	TOGn	PWMn	ECCFn
位 7	6	5	4	3	2	1	0

符号	功 能
—	未实现,保留供以后使用
ECOMn	比较器控制。ECOMn = 1 控制比较器功能
CAPPn	捕捉正边沿。CAPPn = 1 时控制正边沿捕捉
CAPNn	捕捉负边沿。CAPNn = 1 时控制负边沿捕捉
MATn	匹配。当 MATn = 1 时,PCA 计数器同该模块的比较/捕捉寄存器匹配,使得 CCON 中的 CCFn 位置位,标志一个中断
TOGn	触发。当 TOGn = 1 时,PCA 计数器同该模块的比较/捕捉寄存器匹配,使得 CEXn 管脚翻转
PWMn	脉宽调制方式。PWMn = 1 控制 CEXn 管脚用作脉宽调制输出
ECCFn	使 CCF 中断。控制 CCON 寄存器中比较/捕捉标志 CCFn 以产生一个中断

表 9-13 中说明了 CCAPMn 寄存器中有效的位的组合和所具有的规定功能,无效组合将产生不确定的结果。

每个模块都有一对 8 位比较/捕捉寄存器(CCAPMnH,CCAPMnL)与之关联。这些寄存器存放捕捉事件发生的时间或比较事件应该发生的时间。对于 PWM 方式,高位字节寄存器 CCAPMnH 控制波形的空比。

表 9-13 PCA 模块方式(CCAPMn 寄存器)

—	ECOMn	CAPPn	CAPNn	MATn	TOGn	PWMn	ECCFn	模块功能
×	0	0	0	0	0	0	0	无操作
×	×	1	0	0	0	0	×	由 CEXn 上正边沿触发 16 位捕捉
×	×	0	1	0	0	0	×	由 CEXn 上负边沿触发 16 位捕捉
×	×	1	1	0	0	0	×	由 CEXn 上跳变触发 16 位捕捉
×	1	0	0	1	0	0	×	16 位软件定时器
×	1	0	0	1	1	0	×	16 位高速输出
×	1	0	0	0	0	1	0	8 位脉冲宽度调制器
×	1	0	0	1	×	0	×	监视器定时器

下面五部分将详细讨论每个比较/捕捉方式。

3. 16 位捕捉方式

正跳变和负跳变都能触发 PCA 的捕捉，这给 PCA 用来测量多达五个独立输入的脉冲周期、脉冲宽度、脉冲占空比以及脉冲相位差带来了很大的灵活性。置位 CCAPMn 方式寄存器中的 CAPPn 和/或 CAPNn 来选择模块 n 的输入触发(正跳变触发和/或负跳变触发)，见图 9-8。

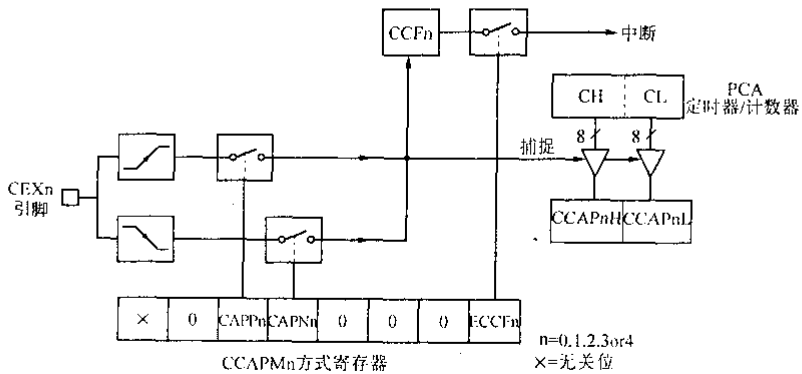


图 9-8 PCA16 位捕捉方式

采样外部输入引脚 CEX0～CEX4 上的脉冲跳变，当检测到一个有效的跳变时(上升沿和/或下降沿)，由硬件将 16 位 PCA 定时器(CH，CL)值装到模块的捕捉寄存器(CCAPnH，CCAPnL)中，捕捉寄存器中的值反映了检测到 CEXn 引脚上跳变时 PCA 定时器的计数值。

在一次捕捉后，置位 CCON 寄存器中模块事件标志 CCFn。如果方式寄存器 CCAPMn 中 ECCFn 置位，就产生一个中断标志。如果中断允许的话，则将产生 PCA 中断。因为在响应中断时硬件并不清除事件标志，事件标志必须由软件清零。在中断服务程序中，在下一个捕捉事件发生之前必须把 16 位的捕捉值保存到 RAM 中。在同一个 CEXn 引脚上随后捕捉到一个跳变时，会把于 CCAPnH 和 CCAPnL 的第一次捕捉值覆盖。

4. 16 位软件定时器方式

在比较方式下，PCA 定时器的 16 位计数值同模块比较寄存器(CCAPnH，CCAPnL)中预装 16 位的值比较。为了能分辨最快的时钟输入(振荡器频率的 1/4)，每个机器周期进行三次比较。置位方式寄存器中的 ECOMn 位控制比较器功能，如图 9-9 所示。

对软件定时器方式，MATn 位也需置位。当 PCA 定时器和比较寄存器之间匹配时，产生一个匹配信号，模块事件标志(CCFn)置位。若 ECCFn 置位，则申请中断。只有在 PCA 中断已经正确地被允许时，才产生一个 PCA 中断。在申请下一次中断之前，软件必须对事件标志清零。

在中断服务程序中，一个新的 16 位比较值可以写入比较寄存器(CCAPnH 和 CCAPnL)。但必须注意的是，在对 CCAPnL 写操作时对 ECOMn 位清零，使比较寄存器更新时暂禁止比较器功能，这样避免了发生无效的匹配。对 CCAPnH 写操作置位 ECOMn，即恢

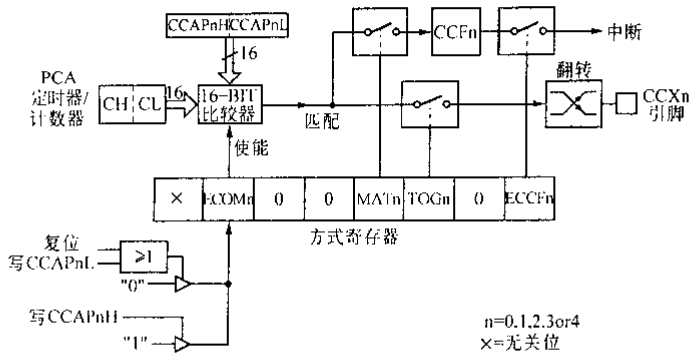


图 9-9　PCA16 位比较器方式

复比较器功能。出于这样的原因，用户软件应首先写 CCAPnL 然后再写 CCAPnH。

5.高速输出方式

在 PCA 定时器和一个模块比较寄存器的预装入值匹配时，高速输出(HSO)方式翻转 CEXn 引脚。对于这种方式，除了 ECOMn 和 MATn 位外还需对 TOGn 置位，正如图 9-9 所示。通过软件对引脚 CEXn 置位或清零，用户可以选择引脚从逻辑 0 变到逻辑 1，或相反方向变化。当匹配事件发生时，通过对 ECCFn 置位，用户可以选择是否要产生中断标志。

高速输出方式比用软件翻转端口的引脚更精确，因为翻转发生在程序转移到中断之前。这就是说中断有等待不影响输出的精度。如果用户在中断服务程序中不改变比较寄存器中的内容，则下次翻转发生在 PCA 定时器计数溢出且与最后的比较值匹配的时候。

6.监视定时器方式

监视定时器是一个引起自动复位电路，除非被监视的系统将有规律地访问监视器。这种电路应用于具有电噪声、电源毛刺、静电放电等应用领域，或者是要求可靠性高的领域。

监视定时器功能只在 PCA 模块 4 中才有。在这种方式下，PCA 定时器的每次计数值和存放在模块 4 的比较寄存器中的值匹配，则产生内部复位(见图 9-10)。选择该方式的位是 CMOD 寄存器中的 WDTE。模块 4 必须置为软件定时器或高速输出的其中一种比较方式。

当 PCA 监视定时器定时已到，它对芯片复位，就像硬位复位一样，不同之处仅在于它不驱动复位引脚到高电平。

为了避免复位，用户有三种选择：

(1)周期性地改变比较值，使它和 PCA 定时器一直不匹配。

(2)周期性地改变 PCA 定时器的值，使它和比较值一直不匹配。

(3)在匹配发生之前对 WDTE 位清零禁止监视定时器作用，然后延迟一段时间再恢复监视定时器功能。

前两种选择比较可靠。如果其它 PCA 模块正被使用，不提倡使用第二种选择，因为该定时器为所有五个模块的时基。因此，在绝大多数情况下，采用第一种解决方案。

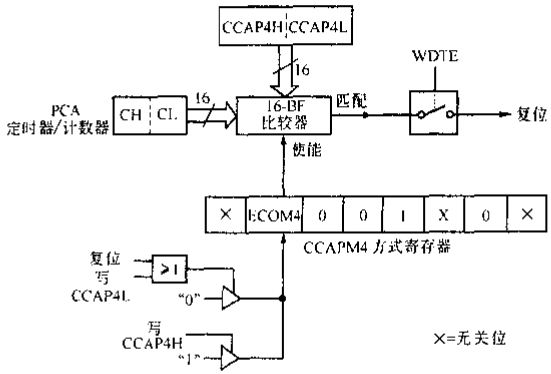


图 9-10 监视定时器方式

如果无需监视定时器,则模块 4 仍可用于其它方式。

7. 脉宽调制解调器 PWM 方式

五个 PCA 模块可以部分或全部编成为 PWM 方式。通过简单的外部电路,PWM 输出可用于将数字量转换为模拟信号。PWM 输出的脉冲频率取决于 PCA 定时器的时钟源。对于 16MHz 晶振,波形的最高频率为 15.6kHz。

通过比较 PCA 定时器的低位字节(CL)和模块比较寄存器(CCAPnL)的低位字节,产生 8 位 PWM 波,参见图 9-11。当 CL< CCAPnL 时,输出为低电平;当 CL≥CCAPnL 时,输出为高电平。CCAPnL 中的值控制波形的占空比。为改变 CCAPnL 中的值而又不至于有输出毛刺,用户必须写入高位字节寄存器(CCAPnH)。当 CL 计数从 0FFH 到 00H 溢出时,由硬件将该值送到 CCAPnL,这相当于下个输出脉冲周期。

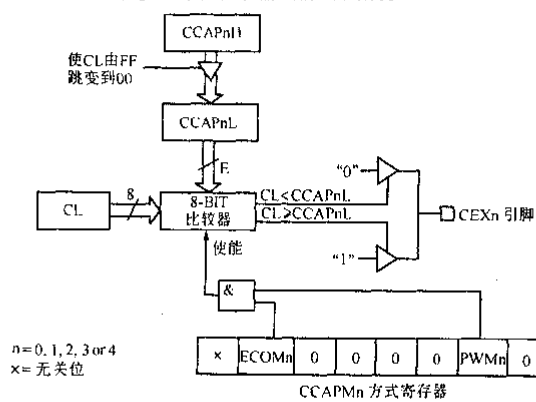


图 9-11 PCA8 位 PWM 方式

CCAPnH 可装有 0~255 的任何整数,使占空比在 100%~0.4%之间变化,见图 9-12。

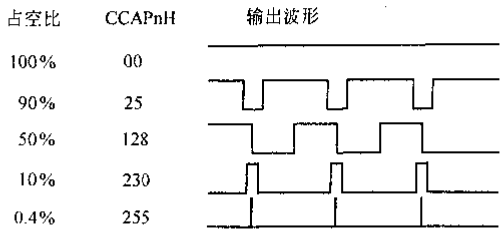


图 9-12　占空比随 CCAPnH 值的变化

五、中断结构

C51FX 共有 7 个中断源：两个外部中断源（$\overline{INT0}$和$\overline{INT1}$），三个定时器中断（定时器 0，1 和 2），串行口中断和 PCA 中断。全部这些中断如图 9-13 所示。

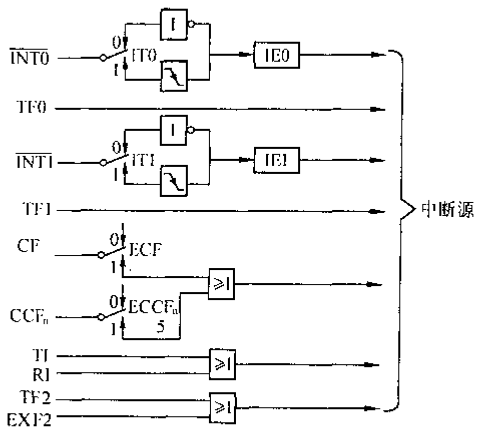


图 9-13　C51FX 中断源

在 7 个中断源中 2 个外部中断，三个定时器中断及串行口中断前面都已介绍过，这里主要介绍 PCA 中断。

1. PCA 中断

PCA 中断由 CCON 寄存器中的 CF、CCF0、CCF1、CCF2、CCF3 和 CCF4 逻辑或产生。在 CPU 转向服务程序时，硬件不对这些标志中的任何一位清零。正常情况下，服务程序要确定是哪一位标志申请这个中断，并在软件中对该位清零。通过中断允许寄存器中的 EC 位（IE.6）控制 PCA 中断允许或中断禁止。此外，还必须通过寄存器 CMOD 和 CCAPMn 中的位 ECF 和 ECCFn，分别控制 CF 标志和每个 CCFn 标志，以便让该标志能申请中断。

2. 中断允许寄存器 IE

C51FX 中的中断允许寄存器共有 8 位，可以位寻址，各位的定义如下：

IE	EA	EC	ET2	ES	ET1	EX1	ET0	EX0

IE 中每一位为 1，允许中断；为 0，禁止中断。

EA 为全局禁止位，若 EA = 0，所有中断都被禁止。

EC 为 PCA 中断允许控制位。EC = 1，允许 PCA 中断；EC = 0，禁止 PCA 中断。

3. 中断优先级结构

在 8XC51FX 中，通过置位或清零中断优先级寄存器 IP 及 IPH 中的位，来控制每一中断源的优先级。

IP	—	PPC	PT2	PS	PT1	PX1	PT0	PX0

IP 中的每一位，构成 C51FX 单片机中断优先级的低位，在 IP 中每位为 1，选择高优先级；位为 0，选择低优先级。IP 中的 7 位使用后 6 位分别控制 6 个中断源，其中 PPC 为 PCA 中断优先级低位，其它位前面章节已经介绍过了。

IPH	—	PPCH	PT2H	PSH	PT1H	PX1H	PT0H	PX0H

IPH 中的每一位，构成 C51FX 单片机中断优先级的高位，各位与中断源的对应关系与 IP 相同。同样在 IPH 中每位为 1，选择高优先级；每位为 0，选择低优先级。

对于每个中断源分别由 2 位（IP 中一位，IPH 中一位）中断优先级控制位来控制其中断优先级。2 位共有 4 种译码状态，分别选择 4 个优先级。其组合关系如表 9-14。

在 C51FX 中低先优先级的中断可被较高优先级中断所中断，但不能被另一个更低优先级或同优先级的中断所中断。如果同时接收到两个相同优先级的中断，单片机服务于哪一个中断请求，由各中断源的内部查询顺序决定。内部查询顺序如表 9-15 所示。

表 9-14 优先级位值

优先组合		中断优先级
IPH.×	IP.×	
0	0	优先级 0 （最低）
0	1	优先级 1
1	0	优先级 2
1	1	优先级 3 （最高）

表 9-15 中断优先级内部查询顺序

中断源	顺序	
$\overline{INT0}$	1	（最高）
定时器 0	2	
$\overline{INT1}$	3	
定时器 1	4	
PCA	5	
串行口	6	
定时器 2	7	（最低）

第三节 87C51GB 单片机

一、引言

8XC51GB 是基于 MCS-51 结构的高集成度 8 位微控制器，共有 68 个引脚（引脚功能图见附录）。作为 MCS-51 的成员，8XC51GB 经过优化，适用于控制应用场合。其主要特征是具有一个 A/D 转换器和两个能用于在 10 个 I/O 引脚上测量及产生脉冲信息的可编程计数器阵列（PCA 及 PCA1）。它还包括一个用于多处理器通信的增强型串行口，一个串行扩展口，硬件监视定时器，振荡器失效检测，一个增/减定时器/计数器和一个片内程序存

储器的程序保密方案。由于8XC51GB采用了CHMOS工艺,因此它有两个软件可选择的节点方式:空闲方式和掉电方式。

8XC51GB采用了标准的8051指令系统,功能上与已有的MCS-51系列产品兼容。

本文将对8XC51GB片内新增硬件功能给出比较全面的说明。其主要性能有:

- 6个8位双向并行口
- 3个16位定时器/计数器
  ——一个增/减定时器/计数器;
  ——可编程时钟输出。
- A/D转换器
  ——8条通道;
  ——8位分辨率;
  ——比较方式。
- 两个可编程计数器阵列
  ——比较/捕捉;
  ——软件定时器;
  ——高速输出;
  ——脉宽调制器;
  ——监视定时器(仅PCA)。
- 全双工可编程串行口
  ——成帧错误检测;
  ——自动地址识别。
- 串行口扩展
  ——4个可编程方式;
  ——4个可选频率。
- 硬件监视定时器
- 复位
  ——异步;
  ——低电平有效。
- 振荡器失效检测
- 中断结构
  ——15个中断源;
  ——4个优先级。
- 节电方式
  ——空闲方式;
  ——掉电方式。

当前使用的8XC51GB产品名称如下:

ROM器件	OTP型号	无ROM的器件型号	ROM/OTP字节	ROM字节
8751GB	8751GB	80C51GB	8KB	256B

二、存储器组织

同所有的 MCS－51 器件程序存储器和数据存储器有各自单独的地址空间。程序存储器和数据存储器逻辑分开，允许通过 8 位地址方式访问数据存储器（特别是内部数据存储器），这样可以用 8 位 CPU 更快地进行存储和操作。当然也可以通过 DPTR 产生的 16 位数据存储器地址。程序存储器和数据存储器的最大寻址空间都是 64KB。

三、特殊功能寄存器（SFR）

片内特殊功能寄存器 SFR 空间的映象如表 9-16 所示。特殊功能寄存器（SFR）包括端口锁存器、定时器、外围控制器等等。这些寄存器只能通过直接寻址访问。SFR 空间有 16 个地址，既可以按字节寻址，又可按位寻址。位寻址 SFR 其地址以 000B 结尾，该区域的位地址为 80H～0FFH。

表 9-16　8XC51GB 特殊功能寄存器（SFR）

F8	P5 00000000	CH 00000000	CCAP0H xxxxxxxx	CCAP1H xxxxxxxx	CCAP2H xxxxxxxx	CCAP3H xxxxxxxx	CCAP4H xxxxxxxx		FF
F0	* B 00000000				AD7 00000000			SEPSTAT xxxxx000	F7
E8	C1CON 00000000	CL 00000000	CCAP0L xxxxxxxx	CCAP1L xxxxxxxx	CCAP2L xxxxxxxx	CCAP3L xxxxxxxx	CCAP4L xxxxxxxx		EF
E0	* ACC 00000000				AD6 00000000			SEPSDAT xxxxxxxx	E7
D8	CCON 00x00000	CMOD 00xxx000	CCAPM0 x0000000	CCAPM1 x0000000	CCAPM2 x0000000	CCAPM3 x0000000	CCAPM4 x0000000		DF
D0	* PSW 00000000				AD5 00000000			SEPCON xx000000	D7
C8	T2CON 00000000	T2MOD xxxxxx00	RCAP2L 00000000	PCAP2H 00000000	TL2 00000000	TH2 00000000			CF
C0	P4 00000000				AD4 00000000		EXICON x0000000	ACMP 00000000	C7
B8	* IP x0000000	SADEN 00000000	C1CAP0H xxxxxxxx	C1CAP1H xxxxxxxx	C1CAP2H xxxxxxxx	C1CAP3H xxxxxxxx	C1CAP4H xxxxxxxx	CH1 00000000	BF
B0	* P3 11111111				AD3 00000000	IPAH 00000000	IPA 00000000	IPH x0000000	B7
A8	* IE 00000000	SADDR 00000000	C1CAP0L xxxxxxxx	C1CAP1L xxxxxxxx	C1CAP2L xxxxxxxx	C1CAP3L xxxxxxxx	C1CAP4L xxxxxxxx	CL1 00000000	AF
A0	* P2 00000000				AD2 00000000	OSCR xxxxxxx0	WDTRST xxxxxxxx	IEA 00000000	A7
98	* SCON 00000000	* SBUF xxxxxxxx	C1CAPM0 x0000000	C1CAPM1 x0000000	C1CAPM2 x0000000	C1CAPM3 x0000000	C1CAPM4 x0000000	C1MOD xxxx0000	9F
90	* P1 00000000				AD1 00000000			ACON xx000000	97
88	* TCON 00000000	* TMOD 00000000	* TL0 00000000	* TL1 00000000	* TH0 00000000	* TH1 00000000			8F
80	* P0 11111111	* SP 00000111	* DPL 00000000	* DPH 00000000	AD0 00000000			* PCON 00x0000	87

* = 可以在 8051 芯片中找到的 SFR（见 8051 硬件注意说明中的 SFR 解释）。x = 未定义。

并不是 SFR 中 256B 空间所有地址都被占据了。未被占据的地址在片内未实现。对这些地址的读访问一般会返回一个随机数,而写访问则没有效果。

- 端口 0~5 寄存器

P0、P1、P2、P3、P4 和 P5 分别为端口 0 到端口 5 的端口锁存器。

- 可编程计数器阵列(PCA 和 PCA1)寄存器

16 位 PCA 和 PCA1 定时器/计数器由寄存器 CH(CH1)和 CL(CL1)组成。寄存器 CCON(CCON1)和 CMOD(CMOD1)包含了 PCA(和 PCA1)的控制位和状态位。CCAPMn(n=0,1,2,3 或 4)和 C1CAPMn 寄存器控制五个 PCA 和五个 PCA1 模块中每一个模块的工作方式。寄存器对(CCAPnH,CCAPnL;C1CAPnH,C1CAPnL)是每个 PCA 和 PCA1 模块的 16 位比较/捕捉寄存器。

- 串行口寄存器

串行数据缓冲器 SBUF 实际上是两个单独的寄存器:一个发送缓冲寄存器和一个接收缓冲寄存器。当数据传送到 SBUF 时,它写接收缓冲器;当从串行口发送数据,它从发送缓冲器取数据。寄存器 SCON 包含了串行口的控制位和状态位。寄存器 SADDR 和 SADEN 用来定义自动地址识别功能的给定地址和广播地址。

- 串行扩展口寄存器

串行扩展口通过寄存器 SERCON 控制。SEPDAT 包含串行扩展口的数据,SEPSTAT 用于监视其状态。

- 中断寄存器

中断控制位在 IE 和 IEA 寄存器中。使用 IP、IPH、IPA 和 IPAH 寄存器可为每个中断选择 4 个优先级中的一个。EXICON 寄存器控制外部中断 2 和 3 触发方向的选择。

- A/D 转换器寄存器

模拟通道 0~7 的 A/D 转换结果分别放在寄存器 AD0、AD1、AD2、AD3、AD4、AD5、AD6 和 AD7 中。寄存器 ACMP 包含 A/D 比较功能的结果。ACON 是 A/D 转换的控制寄存器。

- 电源控制寄存器

PCON 控制节电方式:空闲方式和掉电方式。

- 振荡器失效检测寄存器

OSCR 寄存器既用于监视 OFD 电路的状态,又用于禁止该功能。

- 监视定时器寄存器

监视定时器复位(WDTRST)寄存器用于控制监视定时器对该器件周期性的复位。

四、I/O 端口

8XC51GB 中的全部 6 个端口均是双向的。每个端口由一个锁存器(特殊功能寄存器 P0~P5)、输出驱动器和一个输入缓冲器组成。全部端口除端口 0 之外均为施密特触发器输入。

端口 0 和 2 的输出驱动器与端口 0 的输入缓冲器用于访问外部存储器。在这种应用中,端口 0 输出外部存储器地址的低位字节,并分时复用读写 8 位数据的字节。当地址是 16 位长时,端口 2 输出外部存储器地址的高位字节,否则,端口 2 引脚继续发送 P2 SFR 的

内容。

端口 1、2、3 和 4 的所有引脚和端口 5 的绝大多数引脚是多功能的，如表 9-17 所示。

表 9-17 端口的第二功能

端口管脚	第二功能
P0.0/AD0 ~ P0.7/AD7	外部存储器地址/数据多路复用字节
P1.0/T2	定时器 2 外部时钟输入/时钟输出
P1.1/T2EX	定时器 2 重装/捕捉/指向控制
P1.2/ECI	PCA 外部时钟输入
P1.3/CEX0	PCA 模块 0 捕捉输入，比较/PWM 输出
P1.4/CEX1	PCA 模块 1 捕捉输入，比较/PWM 输出
P1.5/CEX2	PCA 模块 2 捕捉输入，比较/PWM 输出
P1.6/CEX3	PCA 模块 3 捕捉输入，比较/PWM 输出
P1.7/CEX4	PCA 模块 4 捕捉输入，比较/PWM 输出
P2.0/A8 ~ P2.7/A15	外部存储器地址的高位字节
P3.0/RXD	串行口输入
P3.1/TXD	串行口输出
P3.2/$\overline{INT0}$	外部中断 0
P3.3/$\overline{INT1}$	外部中断 1
P3.4/T0	定时器 0 外部时钟输入
P3.5/T1	定时器 1 外部时钟输入
P3.6/$\overline{WR}$	外部存储器写选通
P3.7/$\overline{RD}$	外部存储器读选通
P4.0/SEPCLK	SEP 时钟源
P4.1/SEPDAT	SEP 数据 I/O
P4.2/ECI1	PCA1 外部时钟输入
P4.3/C1EX0	PCA1 模块 0，捕捉输入，比较/PWM 输出
P4.4/C1EX1	PCA1 模块 1，捕捉输入，比较/PWM 输出
P4.5/C1EX2	PCA1 模块 2，捕捉输入，比较/PWM 输出
P4.6/C1EX3	PCA1 模块 3，捕捉输入，比较/PWM 输出
P4.7/C1EX4	PCA1 模块 4，捕捉输入，比较/PWM 输出
P5.2/INT2	外部中断 2
P5.3/INT3	外部中断 3
P5.4/INT4	外部中断 4
P5.5/INT5	外部中断 5
P5.6/INT6	外部中断 6

注：这里所说的第二功能仅在端口 SFR 中相应位锁存内容为 1 时才能触发。否则，端口引脚不会变为高电平。

五、A/D 转换器

8XC51GB 的 A/D 转换器其组成是这样的：8 个模拟输入(ACH0 ~ ACH7)、一个外部触发输入(TRIGIN)、分开的模拟电压源(AV_{SS}和AV_{REF})、一个比较参考输入(COMPREF)和内部电路。内部电路包括：一个 8 通道多路选择器、一个由 256 个电阻构成的梯形网络、一

个比较器、采样保持电容、逐次逼近寄存器、A/D 触发器控制、一个比较结果寄存器和 8 个 A/D 结果寄存器，如图 9-14 中 A/D 框图所示。

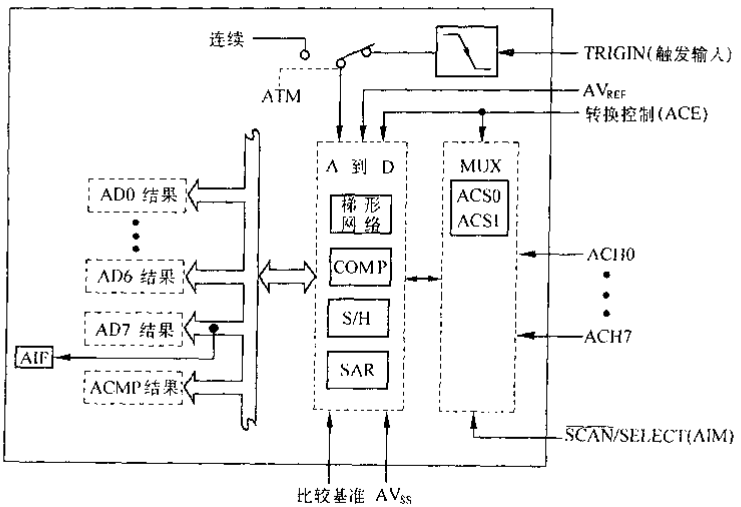


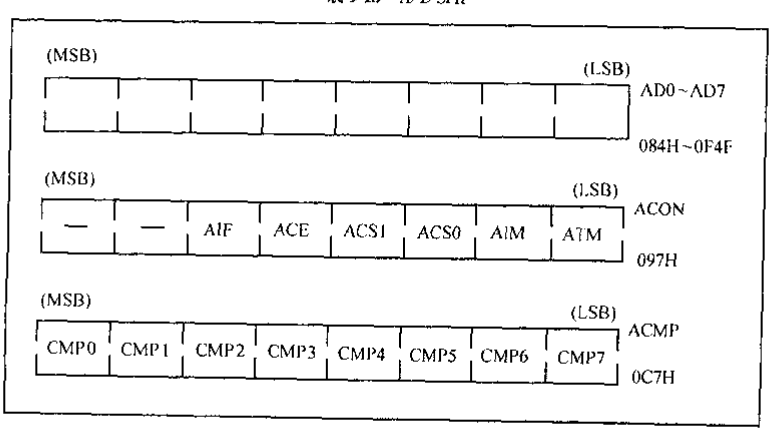
图 9-14 A/D 框图

AV_{REF}必须保持在 8XC51GB 数据资料所规定的容差范围之内。例如，将 AV_{REF}固定到 Vcc 电压的 1/2 处并不能改善 A/D 的准确度。

1. A/D 特殊功能寄存器

A/D 有 10 个 SER 与之关联，如表 9-18。

表 9-18 A/D SFR



AD0～AD7 包含有 8 个模拟量的转换结果。每个转换完成后修改对应的 SFR，从最低通道开始，到通道 7 结束。

ACMP 是比较结果寄存器。ACMP 在组织上和所有其它 SFR 不同，ACMP 中 CMP0 占据 MSB，而 CMP7 则占据 LSB。CMP0～CMP7 对应于模拟输入 0～7。如果模拟输入大于 $COMP_{REF}$，则 CMPn 置 1；如果模拟输入小于或等于 $COMP_{REF}$，则 CMPn 清零。

ACON 是 A/D 控制寄存器，并包含了 A/D 中断标志(AIF)、A/D 转换开始控制(ACE)、A/D 通道选择(ACS0 和 ACS1)、A/D 输入方式(AIM)和 A/D 触发方式(ATM)。

2．A/D 比较方式

当开启 A/D 转换器，A/D 比较方式总是有效。比较方式将每个模拟输入与加至 $COMP_{REF}$上的外部基准电压进行比较。每当 A/D 转换器被触发时，一旦每个模块转换完成，从通道 0 开始到通道 7 结束，ACMP 中的每一位将被更新，不管调用的是选择方式还是扫描方式。比较方式可以提供软件完成得更快的"大于或小于"判决功能，且代码的效率更高。它还可用于将模拟输入转换为带有可变阈值的数字化输入。如果比较方式不同，则 $COMP_{REF}$应接至 Vcc 或 Vss。

3．A/D 触发方式

模拟转换器可以内部触发也可以外部触发。为了允许内部触发方式，ATM 应清零。

当处于内部触发方式时，在 ACE 置位的下一个机器周期，A/D 转换开始。转换首先从最低的通道开始(见"A/D 输入方式")，然后按顺序进行所有其它通道。通道 7 转换完成之后，AIF 标志置位。如果 A/D 中断处于允许状态，则 AIF 将产生一个中断。一旦一个转换周期结束，则新一轮转换周期又从最低通道开始。如果用户需要，每个通道仅转换一次，则 ACE 位应清零。对 ACE 清零将停止 A/D 的所有转换活动。如果一个新的 A/D 周期开始，则前面的转换结果将被覆盖。

在外部方式下，当在 TRIGIN 引脚上检测到一个下降沿信号时，A/D 转换开始。TRIGIN引脚上没有边沿检测器，因此，该引脚每个机器周期被采样一次。

当 TRIGIN 在一个机器周期内为高电平，在下一个机器周期为低电平时，这时就识别了一个负边沿信号。因为这个原因，TRIGIN 高电平应至少保持一个机器周期，低电平也应至少保持一个机器周期。一旦检测到一个下降沿信号，A/D 转换应在下一个机器周期开始，在通道 7 完成转换时结束。通道 7 完成转换之后，AIF 置位，并在 ACE＝1 时该转换停止，直至检测到另一个触发信号。在转换周期进行时，忽略外部触发。

4．A/D 输入方式

8XC51GB 有两个输入方式：扫描方式和选择方式。AIM 清零，将 8XC51GB 置于扫描方式。在扫描方式下，各模拟信号的转换按照 ACH0、ACH1、ACH2、ACH3、ACH4、ACH5、ACH6 和 ACH7 的顺序进行。每个模拟转换的结果分别放入相应的模拟结果寄存器：AD0、AD1、AD2、AD3、AD4、AD5、AD6 和 AD7。

置位 AIM，激活选择方式。在选择方式下，较低 4 个模拟输入(ACH0～ACH3)之一将转换四次。在 4 个转换完成之后，该周期继续转换 ACH4～ACH7。首 4 个转换的结果放在较低 4 个结果寄存器(AD0～AD3)中，其余的转换结果放在其相匹配的结果寄存器中。ACS0 和 ACS1 决定使用哪一个模拟输入，如表 9-19 所示。

表 9-19 A/D 通道选择

ACS1	ACS0	被选择的通道
0	0	ACH0
0	1	ACH1
1	0	ACH2
1	1	ACH3

六、扩展串行口

串行扩展口(SEP)允许较多的串行主机外设接到 8XC51GB 上。SEP 有 4 种可编程方式和 4 种时钟选择。这里有一个双向数据引脚(P4.1)和一个时钟输出引脚(P4.0)。数据传输由带有 8 位接收或发送数据的 8 个时钟脉冲组成。没有发送或接收时,数据引脚和时钟引脚不起作用。有 3 个特殊功能寄存器与 SEP 有关,如图 9-15 所示。

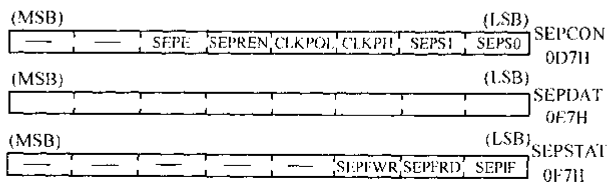


图 9-15 SEP 中的特殊功能寄存器

SEP 所用的特殊功能寄存器没有一个是可以按位寻址。然而,SEPSTAT 和 SEPCON 的个别位比较重要,有专门的符号名与之对应,这些位是:

SEPE ——SEP 控制位;

SEPREN ——SEP 接收允许位;

CLKPOL ——时钟极性位;

CLKPH ——时钟相位位;

SEPS1 ——SEP 速率选择位 1;

SEPS0 ——SEP 速率选择位 0;

SEPFWR——写期间 SEP 故障位;

SEPRD ——读期间 SEP 故障位;

SEPIF ——SEP 中断标志位。

1. 可编程方式和时钟选择

4 种可编程方式决定时钟引脚的无效电平和时钟的哪一个边沿用于发送或接收。这 4 种方式如图 9-16 所示。表 9-20 给出了如何确定这 4 种方式。

4 种时钟选择决定数据移出或移入 SEP 的速率。所有 4 种速率均为振荡器频率的分数。表 9-21 给出了可供 SEP 选择的各种速率。

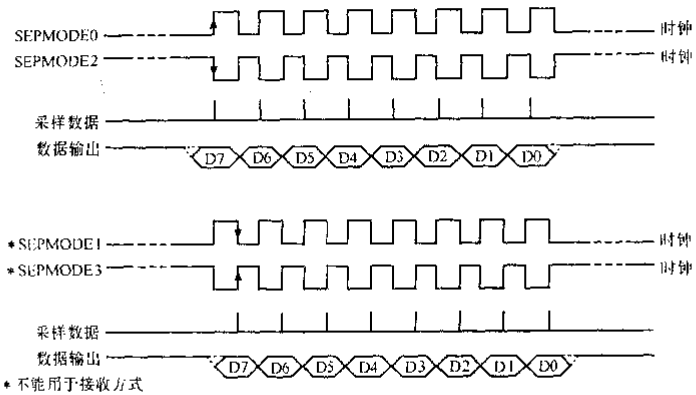


图 9-16 SEP 工作方式图形

表 9-20 SEP 工作方工的确定

CLKPOL	CLKPH	SEP 模 式
0	0	SEP 方式 0
0	1	SEP 方式 1*
1	0	SEP 方式 2
1	1	SEP 方式 3*

表 9-21 SEP 数据速率

SEPS1	SEPS0	数 据 速 率
0	0	振荡器频率/12
0	1	振荡器频率/24
1	0	振荡器频率/48
1	1	振荡器频率/96

2.SEP 发送与接收

为了发送或接收一个字节,用户应初始化 SEP 方式(CLKPOL 和 CLKPH)、时钟频率(SEPS1 和 SEPS0),并置位 SEPE 位开启 SEP。如果用户将数据装入 SEPDATA,则发送发生。当 SEPDATA 为空且不在发送时,如用户置位 SEPREN,则接收发生。当已接收到 8 位时,硬件对 SEPREN 清零。一旦发送或接收完成,SEPIF 就被置位。SEPIF 保持置位结果直到软件清零。SEPIF 也是 SEP 的中断源。发送或接收数据以 MSB 为首。

在 SEP 正在发送或正在接收时,如果用户试图对 SEPDATA 寄存器读或写,或者写 SEPCON 寄存器,则错误位被置位。在 SEP 正在发送时如果进行上述操作,则 SEPFWR 置位;在 SEP 正在接收时如果进行上述操作,则 SEPFRD 置位。没有中断与这些错误位关联。这些位保持置位值直至软件清零。

七、硬件监视定时器

硬件监视寄存器(WDT)在它溢出时复位 8XC51GB。当系统由于干扰导致软件运行紊乱时,WDT 能够作为一种恢复方法。WDT 由一个 14 位计数器和监视定时器 ReSeT(WDTRST)组成。WDT 总是处于有效状态,且振荡器工作时 WDT 递增。没有办法可以禁止 WDT 工作,也就是说,用户在测试或调试一个应用程序时,WDT 一直有效。8XC51GB

退出复位状态时，WDT 装入 0。本节所描述的 WDT 并不是与 PCA 模块 4 关联的那个监视定时器。这个 WDT 不驱动复位引脚。

1.监视定时器的使用

由于处理器运行时 WDT 自动有效，故用户仅需关心对它的服务。当它达到 16383(3FFFH)时 14 位计数器溢出。WDT 每个机器周期递增，也就是说，用户必须每 16383 个机器周期至少复位 WDT 一次。如果用户不希望在应用中使用 WDT 功能，则可用定时器中断复位 WDT。为了复位 WDT，用户必须写 1EH 和 0E1H 两个立即数字节到 WDTRST。WDTRST 为只写寄存器。WDT 计数不能被读或写。在使用 WDT 的应用中不提倡使用定时器中断，因为即使软件紊乱之后，中断仍可被处理。为了更好地发挥 WDT 的作用，WDT 应该服务于那些在防止 WDT 复位所需时间内周期性执行的程序中。

2.掉电方式和空闲方式期间的监视定时器

在掉电方式下，振荡器停止工作，这意味着 WDT 也停止工作。在掉电方式下，用户无须考虑 WDT。这里有两种方法可以退出掉电方式：一是通过复位，另一个则是通过在进入掉电方式之前允许的电平触发外部中断。若用复位方法退出掉电方式，则 WDT 服务就像正常情况下每当 8XC51GB 复位时所做的一样。采用外部中断退出掉电方式与采用复位方法大为不同。中断应保持低电平使器件退出掉电方式并启动振荡器。用户必须维持中断低电平足够长的时间直至振荡器稳定下来。在 Vcc 恢复正常以后，才对中断进行处理。在中断引脚保持低电平时，为了防止 WDT 复位器件，WDT 在中断引脚变为高电平时才启动。建议在处理退出掉电方式的中断期间复位 WDT。

为了确保 WDT 在退出掉电方式的几个状态之内不发生溢出，最好在进入掉电方式之前复位 WDT。

在空闲方式下，振荡器继续工作。为了防止 WDT 在空闲方式期间复位 8XC51GB，用户应该始终设置一个定时器，它周期性地退出空闲方式，处理 WDT，并重新进入空闲方式。

八、中断

8XC51GB 总共有 15 个中断矢量：7 个外部中断($\overline{INT0}$, $\overline{INT1}$, INT2, INT3, INT4, INT5 和 INT6)，3 个定时器中断(定时器 0,1 和 2)，2 个 PCA 中断(PCA 和 PCA1)，A/D 中断，SEP 中断和串行口中断。图 9-17 给出了所有的中断源。

产生中断的所有位可以由软件置位或清零，其效果和由硬件置位或清零效果一样。这就是说，可以在软件中产生中断或将悬挂的中断取消。

1.外部中断

外部中断 $\overline{INT0}$ 和 $\overline{INT1}$ 各自可以是电平触发或负边沿触发，取决于寄存器 TCON 中的 IT0 和 IT1 位。若 ITx = 0，则外部中断 X 由 INTx 引脚上所检测的低电平触发；若 ITx = 1，则外部中断 X 为负边沿触发。

INT2 和 INT3 各自可以是正边沿或负边沿触发，取决于寄存器 EXICON 中的 IT2 和 IT3 位。若 ITx = 0，则外部中断 X 为负边沿触发；若 ITx = 1，则外部中断 X 是正边沿触发。

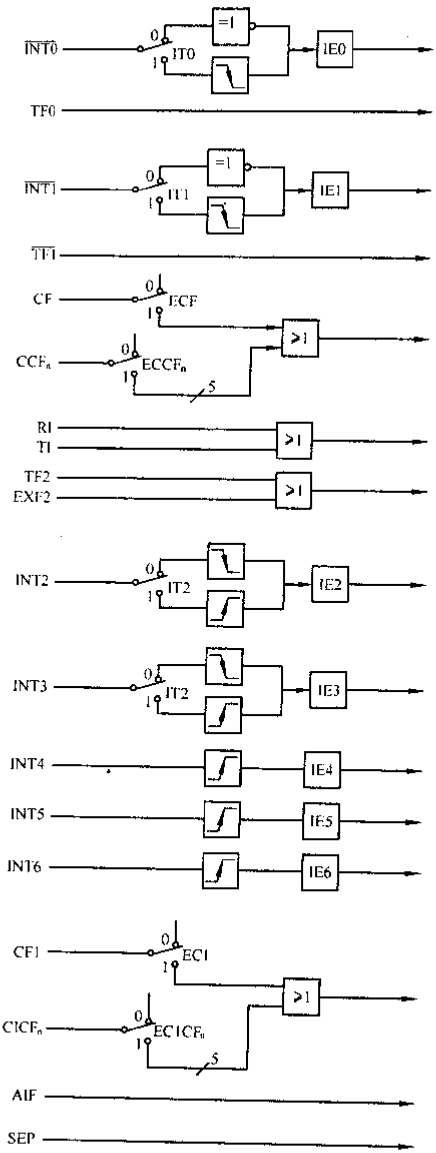

INT0
IT0
=1
IE0
TF0
INT1
IT1
=1
IE1
TF1
CF
ECF
CCF$_n$
ECCF$_n$
5
>1
RI
TI
>1
TF2
EXF2
>1
INT2
IT2
IE2
INT3
IT2
IE3
INT4
IE4
INT5
IE5
INT6
IE6
CF1
EC1
CCF$_n$
EC1CF$_n$
5
>1
AIF
SEP

图 9-17　中断源

INT4、INT5 和 INT6 都只是正边沿触发。

7 个外部中断源产生的中断标志分别在 TCON 中的位 IE0 和 IE1，EXICON 中的位 IE2、IE3、IE4、IE5 和 IE6。如表 9-22。如果中断是由边沿触发，则这些标志在 CPU 转向服务程序时由硬件清零；如果中断是由电平触发的，则是外部中断请求源控制着请求标志，而不是片内硬件控制的，即必须在中断服务程序中应用软件清零。外部中断通过 IE 寄存器中的 EX0 和 EX1 与 IEA 寄存器中的 EX2、EX3、EX4、EX5 和 EX6 控制是否允许中断。

表 9-22　EXICON：外部中断控制寄存器

EXICON 地址 = 0C6H　　复位值 = x 000 0000B

位不可寻址

—	IE6	IE5	IE4	IE3	IE2	IT3	IT2
位 7	6	5	4	3	2	1	0

符号	功　能
—	未实现，保留供以后用
IE6	中断 6 边沿标志。检测到外部中断边沿信号时由硬件置位
IE5	中断 5 边沿标志。检测到外部中断边沿信号时由硬件置位
IE4	中断 4 边沿标志。检测到外部中断边沿信号时由硬件置位
IE3	中断 3 边沿标志。检测到外部中断边沿信号时由硬件置位
IE2	中断 2 边沿标志。检测到外部中断边沿信号时由硬件置位
IT3	中断 3 类型控制位。该位由软件置位或清零，从而控制 INT3 是正跳变触发还是负跳变触发。当 IT3 为高电平时，IE3 由管脚 INT3 上的正跳变置位；当 IT3 为低电平时，IE3 由管脚 INT3 上的负跳变置位
IT2	中断 2 类型控制位。该位由软件置位或清零，从而控制 INT2 是正跳变触发还是负跳变触发。当 IT2 为高电平时，IE2 由管脚 INT2 上的正跳变置位；当 IT2 为低电平时，IE2 由管脚 INT2 上的负跳变置位

2. 中断允许控制

通过置位或清零中断允许寄存器(EA 和 IEA)中的某一位，可以单独地允许或禁止这些中断源的每一个，如表 9-23 所示。应当注意，IE 还包含了一个全局禁止位 EA。若 EA 置位，则通过 IE 和 IEA 中的相应位可以单独地允许或禁止对应的中断；若 EA 清零，则所有中断都被禁止。

3. 中断优先级

通过对中断优先级寄存器(IP 和 IPA)和中断优先级高字节寄存器(IPH 和 IPAH)有关

位置位或清零，可将 8XC51GB 的每一个中断源单独编程为 4 个优先级中的一个，见表 9-24。IPH 寄存器位映象和 IP 寄存器的位映象相同，只是对应的位名后加上字符“H”。这就使每个中断源有 2 位来设置优先级。2 位优先级选择位的低位在 IP 和 IPA 寄存器中，高位在 IPH 和 IPAH 寄存器中，用于选择 4 个优先级中的一个。

表 9-23　中断允许控制寄存器

IE 地址 = 0A8H　　复位值 = 0000 0000B

位可寻址

	EA	EC	ET2	ES	ET1	EX1	ET0	EX0
位	7	6	5	4	3	2	1	0

IEA 地址 = 0A7H　　复位值 = 0000 0000B

位不可寻址

	EAD	EX6	EX5	EX4	EX3	EX2	EX1	EXEP
位	7	6	5	4	3	2	1	0

位 = 1，允许中断

位 = 0，禁止中断

符号	功　能
EA	全部禁止位，若 EA = 0，则所有中断都禁止；若 EA = 1，则通过对每个中断允许位置位或清零可单独地允许或禁止该中断
EC	PCA 中断允许位
ET2	定时器 2 中断允许位
ES	串行口中断允许位
ET1	定时器 1 中断允许位
ET0	定时器 0 中断允许位
EX0	外部中断 0 允许位
EAD	A/D 转换器中断允许位
EX6	外部中断 6 允许位
EX5	外部中断 5 允许位
EX4	外部中断 4 允许位
EX3	外部中断 3 允许位
EX2	外部中断 2 允许位
EX1	外部中断 1 允许位
ESEP	串行扩展口中断允许位

表 9-24 中断优先级寄存器

IP 地址 = 0B8H　　复位值 = ×000 0000B

位可寻址

	—	PPC	PT2	PS	PT1	PX1	PT0	PX0
位	7	6	5	4	3	2	1	0

IPA 地址 = 0B6H　　复位值 = 0000 0000B

位不可寻址

	PAD	PX6	PX5	PX4	PX3	PX2	PC1	PSEP
位	7	6	5	4	3	2	1	0

IPH 地址 = 0B7H　　复位值 = ×000 0000B

位不可寻址

	—	PPPC	PT2H	PSH	PT1H	PX1H	PT0H	PX0H
位	7	6	5	4	3	2	1	0

IPAH 地址 = 0B5H　　复位值 = 0000 0000B

位不可寻址

	PADH	PX6H	PX5H	PX4H	PX3H	PX2H	PC1H	PSEPH
位	7	6	5	4	3	2	1	0

优先级位	优先级位 H	优先级
0	0	最低
0	1	↓
1	0	↓
1	1	最高

符号	功　能
—	未实现，保留供以后用
PPC，PPCH	PCA 中断优先级位
PT2，PT2H	定时器 2 中断优先级位
PS，PSH	串行口中断优先级位
PT1，PT1H	定时器 1 中断优先级位
PX1，PX1H	外部中断 1 中断优先级位
PT0，PT0H	定时器 0 中断优先级位
PX0，PX0H	外部中断 0 中断优先级位
PAD，PADH	A/D 转换器中断优先级位
PX6，PX6H	外部中断 6 中断优先级位
PX5，PX5H	外部中断 5 中断优先级位
PX4，PX4H	外部中断 4 中断优先级位
PX3，PX3H	外部中断 3 中断优先级位
PX2，PX2H	外部中断 2 中断优先级位
PC1，PC1H	PCA1 中断优先级位
PSEP，PSEPH	串行扩展口中断优先级位

低优先级中断本身可被较高优先级的中断所中断,但不能被另一个同等优先级的中断所中断。最高优先级的中断不能被其它任何中断源所中断。

如果同时接收到两个或多个不同优先级的中断请求,则服务较高优先级的中断。如果同时接收到同优先级的中断请求,则由内部查询顺序决定哪一个请求被服务。因此,在同等优先级内还有一个由查询顺序决定的二次优先级结构,如表 9-25 所示。

表 9-25 中断查询顺序

中断源	查询顺序
$\overline{INT0}$	1 (最高)
SEP	2
INT2	3
定时器 0	4
PCA1	5
INT3	6
$\overline{INT1}$	7
A/D	8
INT4	9
定时器 1	10
PCA	11
INT5	12
串行口	13
定时器 2	14
INT6	15 (最低)

4.中断矢量地址

8XC51GB 15 个中断源的矢量地址(中断服务程序入口地址)如表 9-26 所示。

服务程序执行从矢量地址开始,直至遇到 RETI 指令结束。相邻的中断服务程序的起始地址间隔仅为 8 个字节。这也就是说,如果使用相邻中断,而且第一个中断程序比 7 个字节长,必须使用跳转指令到其它存储地址,这样可以完成服务程序而不至于覆盖下一个中断程序的起始地址。一般情况下,中断服务程序的起始地址处都安放一条跳转指令。

表 9-26 中断矢量地址

中断源	中断请求位	硬件清零	矢量地址
$\overline{INT0}$	IE0	否(电平触发) 是(边沿触发)	0003H
定时器 0	TF0	是	000BH
$\overline{INT1}$	IE1	否(电平触发) 是(边沿触发)	0013H
定时器 1	TF1	是	001BH
串行口	RI,TI	否	0023H
定时器 2	TF2,EXF2	否	002BH
PCA	CF,CCFn(n = 0 ~ 4)	否	0033H
A/D	AIF	否	003BH
PCA1	CF1,C1CFn(n = 0 ~ 4)	否	0043H
SEP	SEPIF	否	004BH
INT2	IE2	是	0053H
INT3	IE3	是	005BH
INT4	IE4	是	0063H
INT5	IE5	是	006BH
INT6	IE6	是	0073H

九、节电方式

在有些应用中功耗是很关键的,8XC51GB 提供两种节电方式:空闲方式和掉电方式。在这些方式中,后备电源输入为 Vcc。空闲方式和掉电方式分别由置位 PCON 中的位 IDL

和 PD 来触发，见表 9-27。图 9-19 给出了空闲方式和掉电方式的电路。

表 9-27　PCON：功率控制寄存器

PCON 地址 = 87H　　复位值 = 00× × 0000B

位不可寻址

	SMOD1	SMOD0	—	POF	GF1	GF0	PD	IDL
位	7	6	5	4	3	2	1	0

符号	功　能
SMOD1	双波特率位，置位且定器 1 用作波特率发生器和串行口用在方式 1、2 和 3 时
SMOD0	置位时，读/写访问 SCON.7 是针对 FE 位；清零时，读/写访问 SCON.7 是针对 SM0 位的
—	未实现，保留供以后用
POF	切断电源标志。在 V_{CC} 的上升沿由硬件置位，也可由软件置位或清零。该标志允许检测电源失效引起的复位。V_{CC} 必须保持在 3V 以上，从而保持该位的值
GF1	通用标志位
GF0	通用标志位
PD	掉电方式位。置位触发掉电方式工作
IDL	空闲方式位。置位触发空闲方式工作。若同时写 1 至 PD 和 IDL，则优先执行 PD

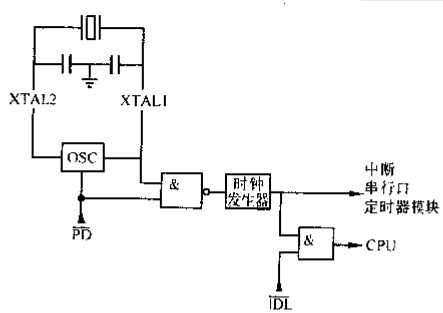


图 9-18　空闲方式和掉电方式硬件电路

在空闲方式(IDL = 1)，振荡器继续运行，中断、串行、PCA 和定时器模块也继续记录，但 CPU 无时钟信号。在掉电方式(PD = 1)，振荡器停止工作。

1.空闲方式

对 IDL 置位的指令是进入空闲方式之前执行的最后一条指令。在空闲方式下，CPU 无内部时钟信号，但中断、定时器和串行口仍有时钟信号。PCA 和 PCA1 定时器可编程为

暂停或继续工作,在空闲期间这由 CMOD(C1MOD)中 CIDL(C1IDL)的值决定。CPU 的状态完全保存着,堆栈指针、程序计数器、程序状态字、累加器和所有其它寄存器的内容都保持它们在空闲期间的数据。端口引脚保持在触发空闲方式时的逻辑状态。ALE 和 PSEN 保持为逻辑高电平。参见表 9-28。

表 9-28 空闲方式期间外部引脚状态

程序存储器	ALE	$\overline{PSEN}$	端口 0	端口 1	端口 2	端口 3,4 和 5
内部	1	1	数据	数据	数据	数据
外部	1	1	悬空	数据	地址	数据

有两种方法可以结束空闲方式。第一是触发任何被允许的中断会使得硬件清零 IDL 位,从而结束掉电模式工作。执行中断服务,在 RETI 后执行的指令将是使器件进入空闲方式时的指令。标志位(PCON 中的 GF0 和 GF1)可用来指示正常方式或空闲期间是否发生中断。例如,触发空闲方式的指令还可对其中的一个标志位或两个标志位置位。中断结束空闲方式时,中断服务程序检查该标志位。

第二是硬件复位。由于时钟振荡器仍在运行,硬件复位仅需保持有效状态 2 个机器周期(24 个振荡器周期)即可完成复位。RESET 引脚上的信号直接而又异步地对 IDL 位清零。这时 CPU 从它停止处开始重新执行程序,也就是说,从触发空闲方式的指令的下一条指令处开始。

2.掉电方式

对 PD 置位的指令是进入掉电方式之前执行的最后一条指令。在该方式下,片内振荡器停止工作。由于时钟停止,所有功能也停止,但片上 RAM 和特殊功能寄存器的内容保持。端口引脚输出值由它们各处的 SFR 保持,ALE 和$\overline{PSEN}$输出低电平。在掉电方式下,Vcc 可降低到 2V。然而,必须注意,确保在进入掉电方式之前 Vcc 不降低。如果在进入掉电方式之前不禁止振荡器失效检测电路,该部分将复位它自身。表 9-29 给出了掉电方式期间外部引脚的状态。

表 9-29 掉电方式期间外部引脚状态

程序存储器	ALE	$\overline{PSEN}$	端口 0	端口 1	端口 2	端口 3,4 和 5
内部	0	0	数据	数据	数据	数据
外部	0	0	悬空	数据	数据	数据

8XC51GB 可使用硬件复位或外部中断退出掉电方式。复位重新定义大部分 SFR 的内容,但不会改变片内 RAM 的值。外部中断允许 SFR 和片内 RAM 均保持原来的内容。

为了正确地结束掉电方式,复位或外部中断在 Vcc 恢复到它的正常工作电平之前不应执行,而且它们还必须保持在触发状态足够长的时间,从而使振荡器重新启动并稳定(一般小于 10ms)。

采用外部中断,INT0 或 INT1 必须被允许,并被设置成电平触发。维持该引脚低电

平,重新启动振荡器,然后将该引脚拉回到高电平完成退出任务。在中断服务程序中执行完 RETI 指令后,下一条指令将是使器件进入掉电方式指令的后面的那条指令。

3.电源切断标志

电源切断标志(POF)位于 PCON.4,Vcc 从 0V 上升到 5V 时由硬件置位。POF 也可由软件置位或清零。这允许用户区分"冷启动"和"热启动"复位。

冷启动复位对应着关掉电源之后再打开 Vcc 并加到器件上的复位。热启动复位则对应着 Vcc 一直加在器件上,而复位是由诸如监视定时器或退出掉电方式所产生的。

复位之后,用户软件可随即检查 POF 位的状态。POF = 1,表示一个冷启动。软件随后对 POF 清零,并开始其任务。复位之后随即检查 POF = 0,表示一次热启动。

为了 POF 保持为 0,Vcc 必须保持在 3V 以上。

思考题与习题

1.8XC52/54/58 系列单片机内部数据存储器与特殊功能寄存器区地址怎样安排?如何访问?

2. 定时器 2 具有几种工作方式?怎样设置?

3.8XC51FX 可编程计数器阵列(PCA)可以有哪几种时钟输入信号?PCA 每个比较/捕捉模块可以编程为哪几种方式?

4.8XC51FX 系列单片机有哪几个中断源?可以选择几个中断优先级?如何选择中断优先级?

5. 简述 8XC51GB 单片机的主要性能。

6. 简述硬件监视定时器的作用。

7.8XC51GB 共有几个中断源?它们是什么?如何选择 INT2 及 INT3 的触发边沿?

8. 有哪两种方法可以结束空闲方式?

附录 I　MCS-51 系列单片机指令一览表

一、按字母顺序排列

助记符		机器码
ACALL	addr 11	* 1 $addr_{7\sim0}$
ADD	A,Rn	28～2F
ADD	A,direct	25 direct
ADD	A,@Ri	26～27
ADD	A,#data	24 data
ADDC	A,Rn	38～3F
ADDC	A,direct	35 direct
ADDC	A,@Ri	36～37
ADDC	A,#data	34 data
AJMP	addr 11	△1 $addr_{7\sim0}$
ANL	A,Rn	58～5F
ANL	A,direct	55 direct
ANL	A,@Ri	56～57
ANL	A,#data	54 data
ANL	direct,A	52 direct
ANL	direct,#data	53 direct data
ANL	C,bit	82 bit
ANL	C,/bit	B0 bit
CJNE	A,direct,rel	B5 direct,rel
CJNE	A,#data,rel	B4 data,rel
CJNE	Rn,#data,rel	B8～BF data,rel
CJNE	@Ri,#data,rel	B6～B7 data,rel
CLR	A	E4
CLR	C	C3
CLR	bit	C2 bit
CPL	A	F4
CPL	C	B3

助记符		机器码
CPL	bit	B2 bit
DA	A	D4
DEC	A	14
DEC	Rn	18～1F
DEC	direct	15 direct
DEC	@Ri	16～17
DIV	AB	84
DJNZ	Rn,rel	D8～DF rel
DJNZ	direct,rel	D5 direct rel
INC	A	04
INC	Rn	08～0F
INC	direct	05 direct
INC	@Ri	06～07
INC	DPTR	A3
JB	bit,rel	20 bit rel
JBC	bit,rel	10 bit rel
JC	rel	40 rel
JMP	@A+DPTR	73
JNB	bit,rel	30 bit rel
JNC	rel	50 rel
JNZ	rel	70 rel
JZ	rel	60 rel
LCALL	addr 16	12 $addr_{15\sim8}$ $addr_{7\sim0}$
LJMP	addr 16	02 $addr_{15\sim8}$ $addr_{7\sim0}$
MOV	A,Rn	E8～EF
MOV	A,direct	E5 direct
MOV	A,@Ri	E6～E7
MOV	A,#data	74 data
MOV	Rn,A	F8～FF
MOV	Rn,direct	A8～FF direct
MOV	Rn,#data	78～7F data

助 记 符		机 器 码
MOV	direct,A	F5 direct
MOV	direct,Rn	88～8F direct
MOV	direct,direct	85 direct direct
MOV	direct,@Ri	86～87 direct
MOV	direct,#data	75 direct data
MOV	@Ri,A	F6～F7
MOV	@Ri,direct	A6～A7 direct
MOV	@Ri,#data	76～77 data
MOV	C,bit	A2 bit
MOV	bit,C	92 bit
MOV	DPTR,#data16	90 $data_{15\sim8}$ $data_{7\sim0}$
MOVC	A,@A+DPTR	93
MOVC	A,@A+PC	83
MOVX	A,@Ri	E2～E3
MOVX	A,@DPTR	E0
MOVX	@Ri,A	F2～F3
MOVX	@DPTR,A	F0
MUL	AB	A4
NOP		00
ORL	A,Rn	48～4F
ORL	A,direct	45 direct
ORL	A,@Ri	46～47
ORL	A,#data	44 data
ORL	direct,A	42 direct
ORL	direct,#data	43 direct,data
ORL	C,bit	72 bit
ORL	C,/bit	A0 bit
POP	direct	D0 direct
PUSH	direct	C0 direct
RET		22
RETI		32

助记符		机器码
RL	A	23
RLC	A	33
RR	A	03
RRC	A	13
SETB	C	D3
SETB	bit	D2 bit
SJMP	rel	80 rel
SUBB	A,Rn	98～9F
SUBB	A,direct	95 direct
SUBB	A,@Ri	96～97
SUBB	A,#data	94 data
SWAP	A	C4
XCH	A,Rn	C8～CF
XCH	A,direct	C5 direct
XCH	A,@Ri	C6～C7
XCHD	A,@Ri	C6～D7
XRL	A,Rn	68～6F
XRL	A,direct	65 direct
XRL	A,@Ri	66～67
XRL	A,#data	64 data
XRL	direct,A	62 direct
XRL	direct,#data	63 direct data

* $=a_{15}a_5a_81$； △$=a_{16}a_5a_80$。

二、按指令功能排列

1. 数据传送指令

助记符		机器码
MOV	A,Rn	E8～EF
MOV	A,direct	E5 direct
MOV	A,@Ri	E6～E7
MOV	A,#data	74 data

助记符		机器码
MOV	Rn,A	F8～FF
MOV	Rn,direct	A8～AF direct
MOV	Rn,#data	78～7F data
MOV	direct,A	F5 direct
MOV	direct,Rn	88～8F direct
MOV	direct,direct	85 direct direct
MOV	direct,@Ri	86～87
MOV	direct,#data	75 direct data
MOV	@Ri,A	F6～F7
MOV	@Ri,direct	A6～A7 direct
MOV	@Ri,#data	76～77 data
MOV	DPTR,#data 16	90 $data_{15\sim8}$ $data_{7\sim0}$
MOVC	A,@A+DPTR	93
MOVC	A,@A+PC	83
MOVX	A,@Ri	E2～E3
MOVX	A,@DPTR	E0
MOVX	@Ri,A	F2～F3
MOVX	@DPTR,A	F0
PUSH	direct	C0 direct
POP	direct	D0 direct
XCH	A,Rn	C8～CF
XCH	A,direct	C5 direct
XCH	A,@Ri	C6～C7
XCHD	A,@Ri	D6～D7
SWAP	A	C4

2. 算术操作指令

助记符		机器码
ADD	A,Rn	28～2F
ADD	A,direct	25 direct
ADD	A,@Ri	26～27
ADD	A,#data	24 data

助记符		机器码
ADDC	A,Rn	38～3F
ADDC	A,direct	35 direct
ADDC	A,@Ri	36～37
ADDC	A,#data	34 data
SUBB	A,Rn	98～9F
SUBB	A,direct	95 direct
SUBB	A,@Ri	96～97
SUBB	A,#data	94 data
INC	A	04
INC	Rn	08～0F
INC	direct	05 direct
INC	@Ri	06～07
DEC	A	14
DEC	Rn	18～1F
DEC	direct	15 direct
DEC	@Ri	16～17
INC	DPTR	A3
MUL	AB	A4
DIV	AB	84
DA	A	D4

3. 逻辑操作指令

助记符		机器码
ANL	A,Rn	58～5F
ANL	A,direct	55 direct
ANL	A,@Ri	56～57
ANL	A,#data	54 data
ANL	direct,A	52 direct
ANL	direct,#data	53 direct data
ORL	A,Rn	48～4F
ORL	A,direct	45 direct
ORL	A,@Ri	46～47

助 记 符		机 器 码
ORL	A,#data	44 data
ORL	direct,A	42 direct
ORL	direct,#data	43 direct data
XRL	A,Rn	68~6F
XRL	A,direct	65 direct
XRL	A,@Ri	66~67
XRL	A,#data	64 data
XRL	direct,A	62 direct
XRL	direct,#data	63 direct data
CLR	A	E4
CPL	A	F4
RL	A	23
RLC	A	33
RR	A	03
RRC	A	13

4. 控制程序转移指令

助 记 符		机 器 码
ACALL	addr11	* 1 $addr_{7\sim0}$
LCALL	addr16	12 $addr_{15\sim8}$$addr_{7\sim0}$
RET		22
RETI		32
AJMP	addr11	△1 $addr_{7\sim0}$
LJMP	addr16	02 $addr_{15\sim8}$$addr_{7\sim0}$
SJMP	rel	80 rel
JMP	@A+DPTR	73
JZ	rel	60 rel
JNZ	rel	70 rel
CJNE	A,direct,rel	B5 direct rel
CJNE	A,#data,rel	B4 data rel
CJNE	Rn,#data,rel	B8~BF data rel
CJNE	@Ri,#data,rel	B6~B7 data rel

助记符		机器码
DJNZ	Rn,rel	D8~DF rel
DJNZ	direct,rel	B5 direct rel
NOP		00

* $=a_{10}a_9a_81$；　△$=a_{10}a_9a_80$。

5. 布尔变量操作指令

助记符		机器码
CLR	C	C3
CLR	bit	C2
SETB	C	D3
SETB	bit	D2
CPL	C	B3
CPL	bit	B2
ANL	C,bit	82 bit
ANL	C,/bit	B0 bit
ORL	C,bit	72 bit
ORL	C,/bit	A0 bit
MOV	C,bit	A2 bit
MOV	bit,C	92 bit
JC	rel	40 rel
JNC	rel	50 rel
JB	bit rel	20 bit rel
JNB	bit rel	30 bit rel
JBC	bit rel	10 bit rel

三、按代码顺序排列

机器码	助记符	
00	NOP	
01 ( )	AJMP	addr
02 ( ) ( )	LJMP	addr 16
03	RR	A

机 器 码	助 记 符	
04	INC	A
05 ( )	INC	direct
06	INC	@R0
07	INC	@R1
08	INC	R0
09	INC	R1
0A	INC	R2
0B	INC	R3
0C	INC	R4
0D	INC	R5
0E	INC	R6
0F	INC	R7
10 ( ) ( )	JBC	bit,rel
11 ( )	ACALL	addr
12 ( ) ( )	LCALL	addr 16
13	RRC	A
14	DEC	A
15 ( )	DEC	direct
16	DEC	@R0
17	DEC	@R1
18	DEC	R0
19	DEC	R1
1A	DEC	R2
1B	DEC	R3
1C	DEC	R4
1D	DEC	R5
1E	DEC	R6
1F	DEC	R7
20 ( ) ( )	JB	bit,rel
21 ( )	AJMP	addr
22	RET	
23	RL	A
24 ( )	ADD	A,#data
25 ( )	ADD	A,direct
26	ADD	A,@R0
27	ADD	A,@R1

机 器 码	助 记 符	
28	ADD	A,R0
29	ADD	A,R1
2A	ADD	A,R2
2B	ADD	A,R3
2C	ADD	A,R4
2D	ADD	A,R5
2E	ADD	A,R6
2F	ADD	A,R7
30 ( )( )	JNB	bit,rel
31 ( )	ACALL	addr
32	RETI	
33	RLC	A
34 ( )	ADDC	A,#data
35 ( )	ADDC	A,direct
36	ADDC	A,@R0
37	ADDC	A,@R1
38	ADDC	A,R0
39	ADDC	A,R1
3A	ADDC	A,R2
3B	ADDC	A,R3
3C	ADDC	A,R4
3D	ADDC	A,R5
3E	ADDC	A,R6
3F	ADDC	A,R7
40 ( )	JC	rel
41 ( )	AJMP	addr
42 ( )	ORL	direct,A
43 ( )( )	ORL	direct,#data
44 ( )	ORL	A,#data
45 ( )	ORL	A,direct
46	ORL	A,@R0
47	ORL	A,@R1
48	ORL	A,R0
49	ORL	A,R1
4A	ORL	A,R2
4B	ORL	A,R3

机 器 码	助 记 符	
4C	ORL	A,R4
4D	ORL	A,R5
4E	ORL	A,R6
4F	ORL	A,R7
50 ( )	JNC	rel
51 ( )	ACALL	addr
52 ( )	ANL	direct,A
53 ( ) ( )	ANL	direct,#data
54 ( )	ANL	A,#data
55 ( )	ANL	A,direct
56	ANL	A,@R0
57	ANL	A,@R1
58	ANL	A,R0
59	ANL	A,R1
5A	ANL	A,R2
5B	ANL	A,R3
5C	ANL	A,R4
5D	ANL	A,R5
5E	ANL	A,R6
5F	ANL	A,R7
60 ( )	JZ	rel
61 ( )	AJMP	addr
62 ( )	XRL	direct,A
63 ( ) ( )	XRL	direct,#data
64 ( )	XRL	A,#data
65 ( )	XRL	A,direct
66	XRL	A,@R0
67	XRL	A,@R1
68	XRL	A,R0
69	XRL	A,R1
6A	XRL	A,R2
6B	XRL	A,R3
6C	XRL	A,R4
6D	XRL	A,R5
6E	XRL	A,R6
6F	XRL	A,R7

机 器 码	助 记 符	
70 ( )	JNZ	rel
71 ( )	ACALL	addr
72 ( )	ORL	C,bit
73	JMP	@A+DPTR
74	MOV	A,#data
75	MOV	direct,#data
76	MOV	@R0,#data
77	MOV	@R1,#data
78 ( )	MOV	R0,#data
79 ( )	MOV	R1,#data
7A ( )	MOV	R2,#data
7B ( )	MOV	R3,#data
7C ( )	MOV	R4,#data
7D ( )	MOV	R5,#data
7E ( )	MOV	R6,#data
7F ( )	MOV	R7,#data
80 ( )	SJMP	rel
81 ( )	AJMP	addr
82 ( )	ANL	C,bit
83	MOVC	A,@A+PC
84	DIV	AB
85 ( ) ( )	MOV	direct,direct
86 ( )	MOV	direct,@R0
87 ( )	MOV	direct,@R1
88 ( )	MOV	direct,R0
89 ( )	MOV	direct,R1
8A ( )	MOV	direct,R2
8B ( )	MOV	direct,R3
8C ( )	MOV	direct,R4
8D ( )	MOV	direct,R5
8E ( )	MOV	direct,R6
8F ( )	MOV	direct,R7
90 ( ) ( )	MOV	DPTR+#data16
91 ( )	ACALL	addr
92 ( )	MOV	bit,C
93	MOVC	A,@A+DPTR

机 器 码	助 记 符	
94 ( )	SUBB	A,#data
95 ( )	SUBB	A,direct
96	SUBB	A,@R0
97	SUBB	A,@R1
98	SUBB	A,R0
99	SUBB	A,R1
9A	SUBB	A,R2
9B	SUBB	A,R3
9C	SUBB	A,R4
9D	SUBB	A,R5
9E	SUBB	A,R6
9F	SUBB	A,R7
A0 ( )	ORL	C,bit
A1 ( )	AJMP	addr
A2 ( )	MOV	C,bit
A3	INC	DPTR
A4	MUL	AB
A5	保留	
A6 ( )	MOV	@R0,direct
A7 ( )	MOV	@R1,direct
A8 ( )	MOV	R0,direct
A9 ( )	MOV	R1,direct
AA	MOV	R2,direct
AB	MOV	R3,direct
AC	MOV	R4,direct
AD	MOV	R5,direct
AE	MOV	R6,direct
AF ( )	MOV	R7,direct
B0 ( )	ANL	C,bit
B1 ( )	ACALL	addr
B2 ( )	CPL	bit
B3	CPL	C
B4 ( ) ( )	CJNE	A,#data,rel
B5 ( ) ( )	CJNE	A,#data,rel
B6 ( ) ( )	CJNE	@R0,#data,rel
B7 ( ) ( )	CJNE	@R1,#data,rel

机 器 码	助 记 符	
B8 ( ) ( )	CJNE	R0,#data,rel
B9 ( ) ( )	CJNE	R1,#data,rel
BA ( ) ( )	CJNE	R2,#data,rel
BB ( ) ( )	CJNE	R3,#data,rel
BC ( ) ( )	CJNE	R4,#data,rel
BD ( ) ( )	CJNE	R5,#data,rel
BE ( ) ( )	CJNE	R6,#data,rel
BF ( ) ( )	CJNE	R7,#data,rel
C0 ( )	PUSH	direct
C1 ( )	AJMP	addr
C2 ( )	CLR	bit
C3	CLR	C
C4	SWAP	A
C5 ( )	XCH	A,direct
C6	XCH	A,@R0
C7	XCH	A,@R1
C8	XCH	A,R0
C9	XCH	A,R1
CA	XCH	A,R2
CB	XCH	A,R3
CC	XCH	A,R4
CD	XCH	A,R5
CE	XCH	A,R6
CF	XCH	A,R7
D0 ( )	POP	direct
D1 ( )	ACALL	addr
D2 ( )	SETB	bit
D3	SETB	C
D4	DA	A
D5 ( ) ( )	DJNZ	direct,rel
D6	XCHD	A,@R0
D7	XCHD	A,@R1
D8	DJNZ	R0,rel
D9	DJNZ	R1,rel
DA	DJNZ	R2,rel
DB	DJNZ	R3,rel

机 器 码	助 记 符	
DC	DJNZ	R4,rel
DD	DJNZ	R5,rel
DE	DJNZ	R6,rel
DF	DJNZ	R7,rel
E0	MOVX	A,@DPTR
E1 ( )	AJMP	addr
E2	MOVX	A,@R0
E3	MOVX	A,@R1
E4	CLR	A
E5 ( )	MOV	A,direct
E6	MOV	A,@R1
E7	MOV	A,@R0
E8	MOV	A,R0
E9	MOV	A,R1
EA	MOV	A,R2
EB	MOV	A,R3
EC	MOV	A,R4
ED	MOV	A,R5
EE	MOV	A,R6
EF	MOV	A,R7
F0	MOVX	@DPTR,A
F1 ( )	ACALL	addr
F2	MOVX	@R0,A
F3	MOVX	@R1,A
F4	CPL	A
F5 ( )	MOV	direct,A
F6	MOV	@R0,A
F7	MOV	@R1,A
F8	MOV	R0,A
F9	MOV	R1,A
FA	MOV	R2,A
FB	MOV	R3,A
FC	MOV	R4,A
FD	MOV	R5,A
FE	MOV	R6,A
FF	MOV	R7,A

附录Ⅰ　MCS-51特殊功能寄存器一览表

寄存器符号	名　　称	地　　址
*ACC	累加器	0E0H
*B	B寄存器	0F0H
*PSW	程序状态字	0D0H
SP	堆栈指针	81H
DPTR	数据指针	
DPL	数据指针低位	82H
DPH	数据指针高位	83H
*P0	P0口锁存器	80H
*P1	P1口锁存器	90H
*P2	P2口锁存器	0A0H
*P3	P3口锁存器	0B0H
*IP	中断优先级控制寄存器	0B8H
*IE	中断允许控制寄存器	0A8H
TMOD	定时器/计数器方式控制寄存器	89H
*TCON	定时器/计数器控制寄存器	88H
TH0	定时器/计数器0(高字节)	8CH
TL0	定时器/计数器0(低字节)	8AH
TH1	定时器/计数器1(高字节)	8DH
TL1	定时器/计数器1(低字节)	8BH
*SCON	串行控制寄存器	98H
SBUF	串行数据缓冲器	99H
PCON	电源控制寄存器	87H

* =可位寻址。

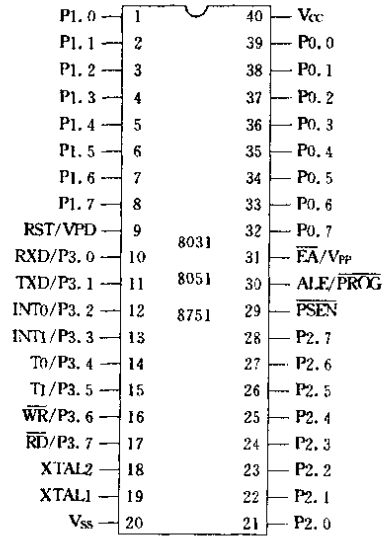
附录Ⅲ　MCS-51特殊功能寄存器位地址分布

直接字节地址	位地址（高位）							位地址（低位）	硬件寄存器符号
F0H	F7	F6	F5	F4	F3	F2	F1	F0	B
E0H	E7	E6	E5	E4	E3	E2	E1	E0	ACC
D0H	D7	D6	D5	D4	D3	D2	D1	D0	PSW
B8H	—	—	—	BC	BB	BA	B9	B8	IP
B0H	B7	B6	B5	B4	B3	B2	B1	B0	P3
A8H	AF	—	—	AC	AB	AA	A9	A8	IE
A0H	A7	A6	A5	A4	A3	A2	A1	A0	P2
98H	9F	9E	9D	9C	9B	9A	99	98	SCON
90H	97	96	95	94	93	92	91	90	P1
88H	8F	8E	8D	8C	8B	8A	89	88	TCON
80H	87	86	85	84	83	82	81	80	P0

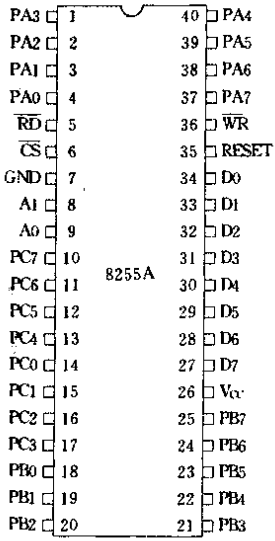
附录Ⅳ　MCS-51 内部 RAM 的位地址分布

字节地址	位地址							
	D7	D6	D5	D4	D3	D2	D1	D0
FH	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

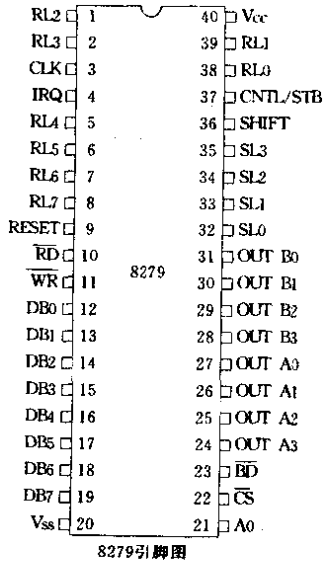
附录Ⅴ　本书选取的芯片的引脚图



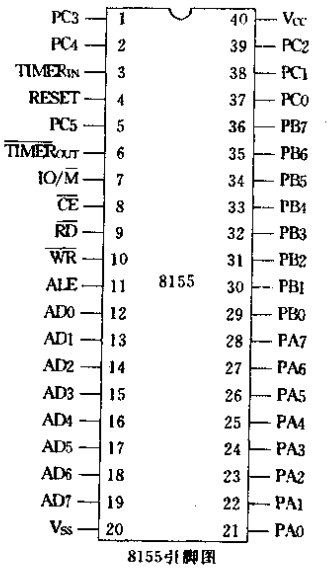
MCS-51引脚图



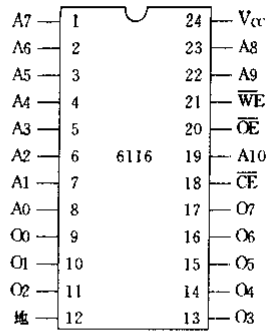
8255引脚图



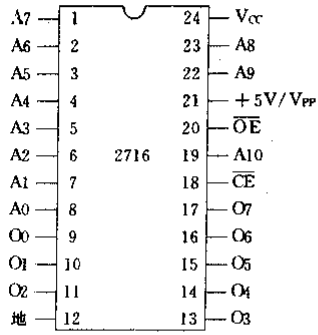
8279引脚图



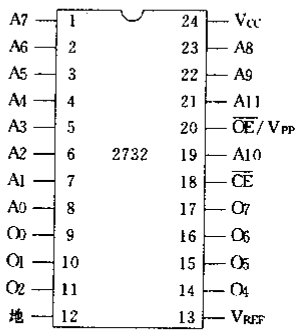
8155引脚图



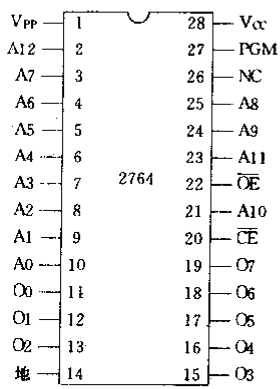
RAM6116(2K×8)



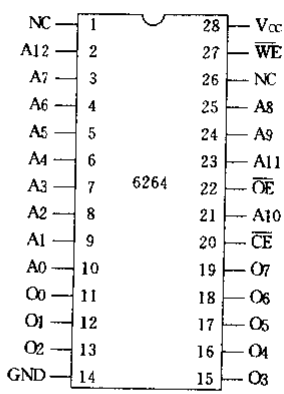
EPROM2716(2K)



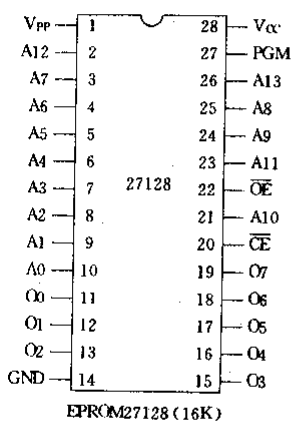
EPROM2732(4K)



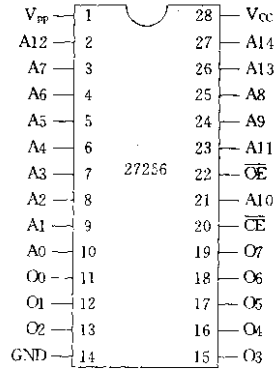
EPROM2764(8K)



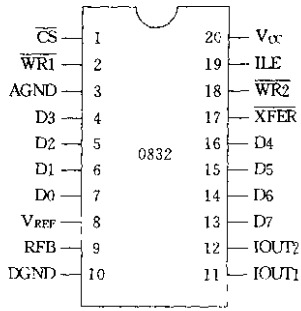
RAM6264(8K×8)



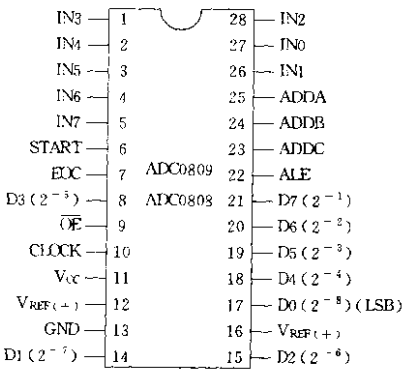
EPROM27128(16K)



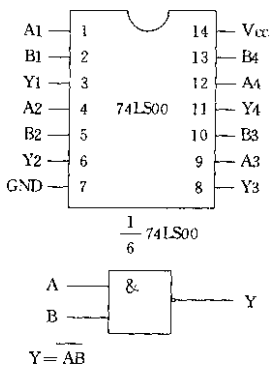
EPROM27256(32K)



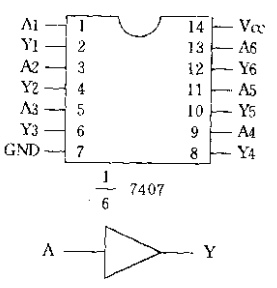
DAC0832(8位D/A)引脚图



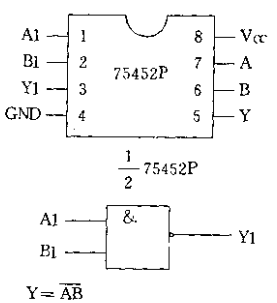
ADC0809引脚图



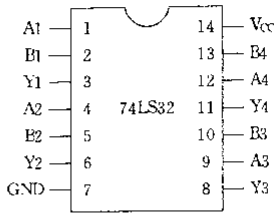
74LS00四-2输入端与非门引脚图



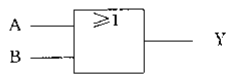
7407六开集电极高电压输出缓冲器引脚图



75452P双驱动器引脚图

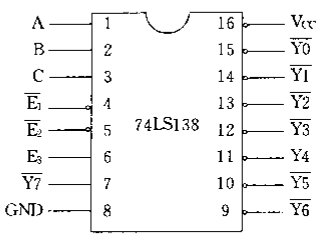


$\frac{1}{4}$ 74LS32



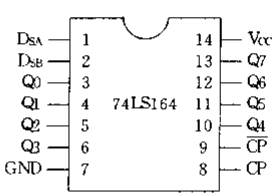
Y=A+B

74LS32四-2输入端或门的引脚图

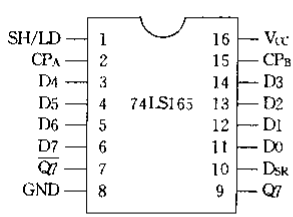


输入						输出
译码控制			译码选择			
E_3	$\overline{E_2}$	$\overline{E_1}$	C	B	A	
1	0	0	0	0	0	$\overline{Y0}=0$ 其余为1
1	0	0	0	0	1	$\overline{Y1}=0$ 其余为1
1	0	0	0	1	0	$\overline{Y2}=0$ 其余为1
1	0	0	0	1	1	$\overline{Y3}=0$ 其余为1
1	0	0	1	0	0	$\overline{Y4}=0$ 其余为1
1	0	0	1	0	1	$\overline{Y5}=0$ 其余为1
1	0	0	1	1	0	$\overline{Y6}=0$ 其余为1
1	0	0	1	1	1	$\overline{Y7}=0$ 其余为1
不是上述情况			×	×	×	全部输出为1

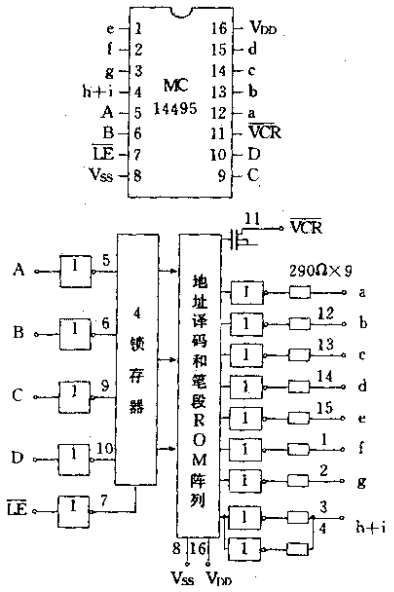
74LS138 3-8译码器引脚图、真值表



74LS164引脚图

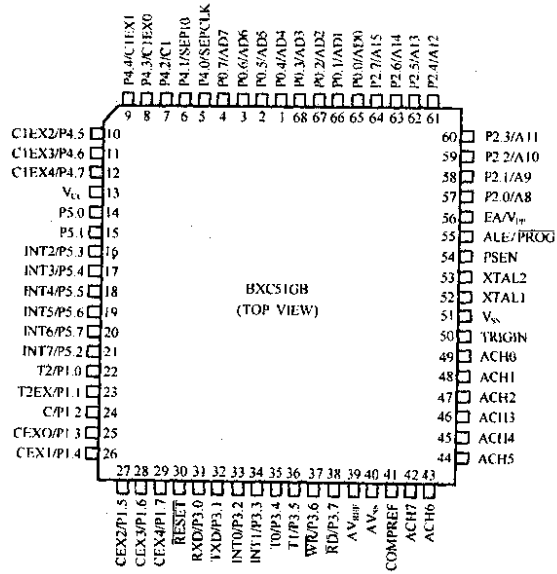


74LS165引脚图



输入				输出							显示
D	C	B	A	a	b	c	d	e	f	g	h+i	
0	0	0	0	1	1	1	1	1	1	0	0	0
0	0	0	1	0	1	1	0	0	0	0	0	1
0	0	1	0	1	1	0	1	1	0	1	0	2
0	0	1	1	1	1	1	1	0	0	1	0	3
0	1	0	0	0	1	1	0	0	1	1	0	4
0	1	0	1	1	0	1	1	0	1	1	0	5
0	1	1	0	1	0	1	1	1	1	1	0	6
0	1	1	1	1	1	1	0	0	0	0	0	7
1	0	0	0	1	1	1	1	1	1	1	0	8
1	0	0	1	1	1	1	0	1	1	1	0	9
1	0	1	0	1	1	1	0	1	1	1	1	A
1	0	1	1	0	0	1	1	1	1	1	1	b
1	1	0	0	1	0	0	1	1	1	0	1	C
1	1	0	1	0	1	1	1	1	0	1	1	d
1	1	1	0	1	0	0	1	1	1	1	1	E
1	1	1	1	1	0	0	0	1	1	1	1	F

MC14495引脚图



附录Ⅵ　常用波特率与其它参数选取关系

				定时器 T1		
串行口工作方式	波特率	*fosc*	SMOD	C/T	模式	定时器初值
方式 0	1M	12MHz	×	×	×	×
方式 2	375k	12MHz	1	×	×	×
	187.5k	12MHz	0	×	×	×
方式 1 或方式 3	62.5k	12MHz	1	0	2	FFH
	19.2k	11.059MHz	1	0	2	FDE
	9.6k	11.059MHz	0	0	2	FDH
	4.8k	11.059MHz	0	0	2	FAH
	2.4k	11.059MHz	0	0	2	FAH
	1.2k	11.059MHz	0	0	2	F8H
	137.5k	11.059MHz	0	0	2	1DH
	110	12MHz	0	0	1	FEEBH
方式 0	0.5M	6MHz	×	×	×	×
方式 2	187.5k	6MHz	1	×	×	×
方式 1、3	19.2k	6MHz	1	0	2	FEH
	9.6k	6MHz	1	0	2	FDH
	4.8k	6MHz	0	0	2	FDH
	2.4k	6MHz	0	0	2	FAH
	1.2k	6MHz	0	0	2	F4H
	0.6k	6MHz	0	0	2	E8H
	110	6MHz	0	0	2	72H
	55	6MHz	0	0	1	FEEBH