MCS-51 单片机原理

李　　洁　　等编著

北京大学信息科学技术学院

2009 年 2 月

目　　录

第一章 MCS－51 单片机

单片机、单板机以及个人计算机都属于微型计算机范畴，共同点是具有体积小、重量轻、功能强、操作简便和价格低廉等特征；不同点是单片机用于控制，而个人计算机用于数据处理。例如，单片机将运算器、控制器、内存以及内部总线和 I/O 接口集成在 5mm×5mm 贴片式芯片内，因体积小而被广泛用于智能化仪器仪表、数控机床、电梯控制、计程器、收银机等，由此单片机常被称作嵌入式计算机。51 系列单片机内部寄存器和总线是 8 位的、内存较小，系统程序（监控程序）和用户程序都是以目标代码的形式保存在存储器中。大多数单片机不使用高级语言，只能利用 IBM-PC 微机系统通过交叉汇编方法得到二进制的目标码，在教学过程中常把指令系统和汇编语言程序设计作为单片机软件的学习内容。

1.1 MCS－51 单片机的总体结构和引脚

MCS－51 单片机 8051 的引脚如图 1-1 所示，总体结构如图 1-2 所示。

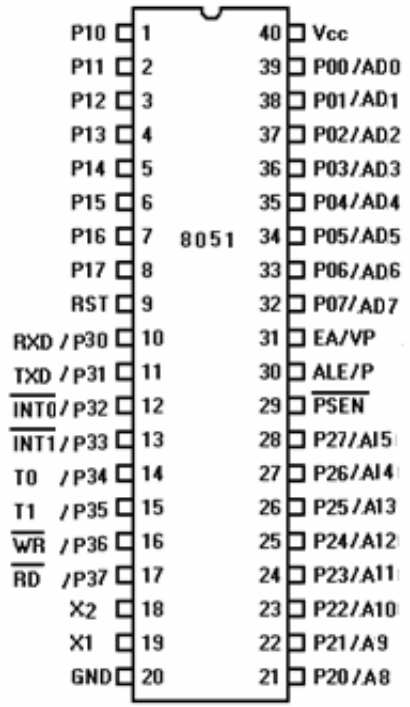
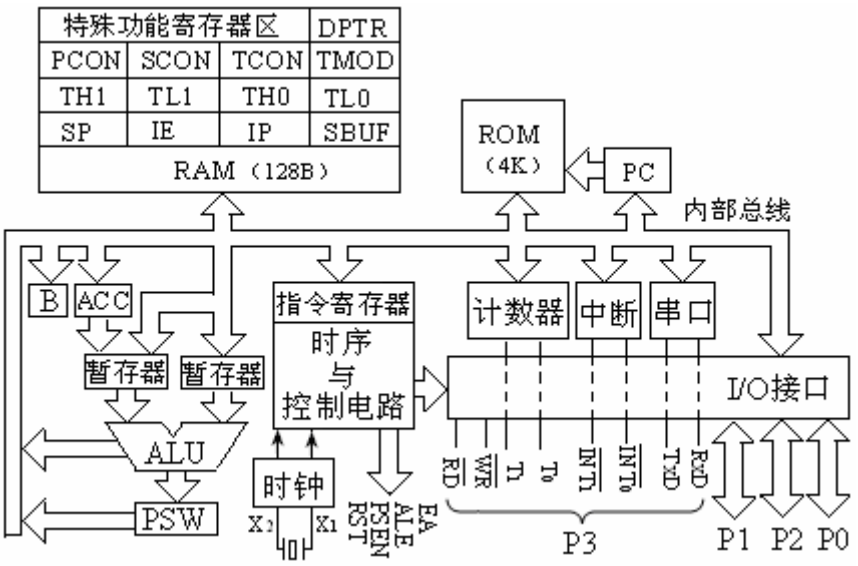


图 1-1 8051 系列单片机的管脚图



CPU 运算器与控制器　　控制总线　　P1 接外设　P2P0 地址/数据

图 1-2　8051 系列单片机的内部结构图

1.1.1 内部结构

1. 运算器与控制器（CPU—Central Processing Unit）

运算器与控制器构成了中央处理器 **CPU**，它是单片机的核心，完成运算和控制功能。控制器包括程序计数器 **PC**、指令寄存器、指令译码器、微指令控制器、时钟、以及时序和控制电路。程序的执行是根据 **PC** 指示的指令地址到内存 **ROM** 中取指令、对指令译码产生控制信号、取数据、运算并存结果。每执行完一条指令，**PC** 就自动加 1 保证了程序的顺序执行。若想转移到某子程序或其它地址，只要改变 **PC** 的内容就改变了程序的执行方向。用于实时控制的程序在执行过程中还要与定时、中断以及外部设备有联系，控制器保证了单片机内外电路协调地工作。

运算器由算术逻辑运算部件 **ALU**、累加器 **ACC**、寄存器 **B**、暂存器和程序状态字 **PSW** 组成。

ALU（Arithmetic logic unit）算术逻辑运算部件。

ACC（Accumulator）累加器是使用最频繁的寄存器，许多指令规定了必须由累加器 **ACC** 提供一个操作数，其助记符是 A。

B 寄存器通常在乘除运算中与累加器 A 一同构成 16 位的数据寄存器。其它时间可作 8 位寄存器使用。

PSW（Program Status Word Register）程序状态字存放程序执行过程的状态特征，为程序的执行提供判断的依据，各位含义如下所述：

D7	D6	D5	D4	D3	D2	D1	D0
Cy	AC	F0	RS1	RS0	OV	—	P

Cy（Carry flag）进位标志位，当 **Cy**＝1 时表示加法时的进位或减法时的借位。**Cy** 可以用软件置 1 或置 0，在位操作中 **Cy** 被简写成 **C**。

AC（Auxiliary Carry flag）辅助进位标志，**AC**＝1 表示在执行加法运算时，低 4 位向高 4 位有进位，该标志只在十进制调整中用到。

F0 这是用户自定义的程序标志位，通过软件置 1 或置 0 。

RS1、**RS0** 通过软件对这两位置 1 或置 0 来选择内部 **RAM** 的四个工作寄存器组。

OV（Overflow flag）溢出标志位，当 8 位有符号数进行算术运算时，其结果超出 －128～＋127 时 OV＝1。

P（Parity flag）奇偶校验位，MCS－51 采用偶校验，当 P＝1 时表示本次运算结果和本标志中“1”的总数必须为偶数。

D1 这一位是保留位。

另外，MCS－51 系列单片机还具有位寻址和位操作功能，这是与单板机以及个人机最大不同之处，这一功能在开关决策、逻辑仿真和实时控制方面可以大大提高运行速度，体现了单片机的独有优势。

2. 存贮器 在 1.2 节专门讨论

3. 内部总线和 I/O 接口

芯片内部总线为片内各个功能部件相互之间的数据、地址和控制信号的传送提供了公共的通路。芯片内部的 4 个 8 位并行的输入/输出接口简称 I/O 接口 **P3～P0**，除了 **P1** 口留给用户使用外，其它口都具有第二功能。如：**P0** 口为分时复用的数据线和低 8 位地址线、**P2** 口提供高 8 位地址、**P3** 口用作控制口传送部分控制线。每一位用 D 触发器作锁存器使用、当锁存信号 cp 有效时锁存来自内部总线上的信号，之后内部总线可以传送其它信号；增强型管 FET1 和 FET2 是输出驱动器；输入有三态缓冲器。各口功能有所差异，内部结构有所不同，如图 1-3 所示。

P0 口有二个功能，通过二选一数据选择器 MUX 的选择端 S 选择，第一功能是作 I/O 接口使用（S＝0）、第二功能是传送地址或数据 A/D（S＝1）， 耗尽型管 FET0 作为上拉电阻使用。二个功能不同之处体现在输出。

⑴输出

P0 口作 I/O 接口使用时 S＝0（FET1 断开），数据选择器 MUX 的输出是数据 Y＝/Q。一旦通过指令向 **P0** 口输出一个逻辑电平，**P0** 口的锁存器以及管脚将维持这个电平不变，直到下一个输出指令为止。

若 D＝0→Y＝/Q＝1→SFET2 导通→P0.X＝0；

若 D＝1→Y＝/Q＝0→SFET2 截止→P0.X＝1。

P0 口作为地址/数据线使用时 S＝1，MUX 输出地址或数据 $Y=\overline{A/D}$。

若 A/D＝0→Y＝1→FET2 导通、FET1 断开→P0.X＝0；

若 A/D＝1→Y＝0→FET2 断开、FET1 导通→P0.X＝1。

⑵输入

输入时必须首先对锁存器置“1”使内部 FET2 断开，输入又分为读引脚和读锁存器的二种方式，例如：

```
MOV   P0，0FFH    ；对锁存器送“1”使内部 FET2 断开
MOV   A，P0       ；MOV 指令读引脚将引脚信号送到内部总线
ANL   A，P0       ；该指令读锁存器将锁存器当前状态送到内部总线
```

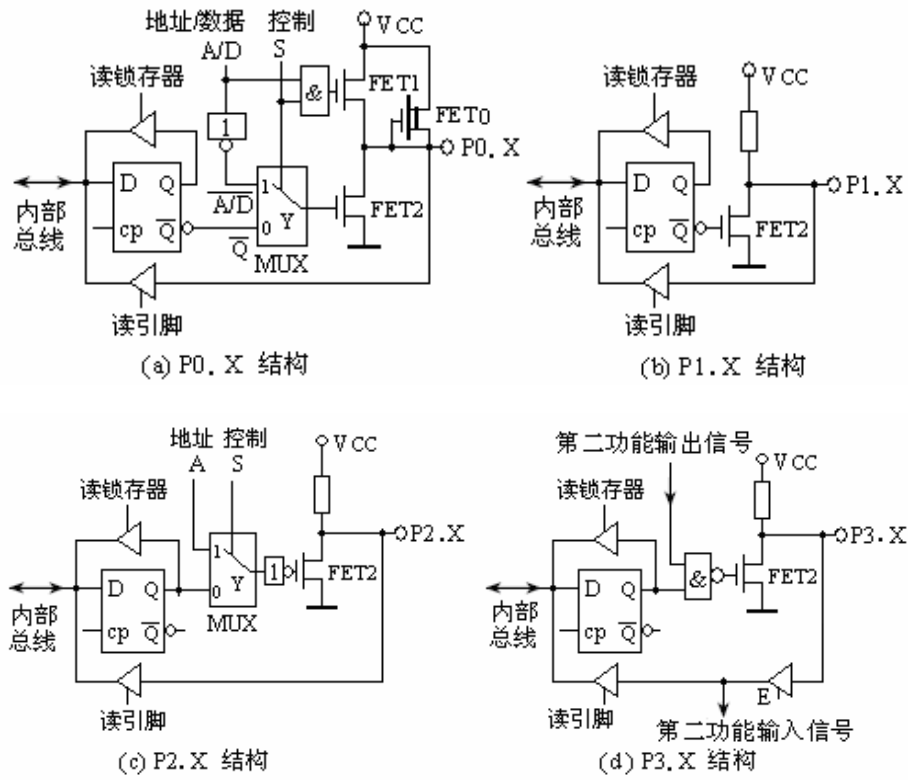


图 1-3　8051 系列单片机的 I/O 接口

P1 口和 **P2** 口比较简单，读者自己分析。

P3 口作为通用 I/O 口输出使用时“第二功能输出信号”为 1，以保证输出信号通路畅通；作为通用 I/O 口输入使用时必须首先向 **P3** 端口送 0FFH，待“读

引脚”及 E 有效时数据送到内部总线。

P3 口作为第二功能输出使用时，应使锁存器 Q 端置“1”以保证“第二功能输出信号”输出畅通；作为第二功能输入使用时输入引脚信号，必须首先通过指令对锁存器送“1”使内部 FET2 断开，输入指令产生“读引脚”无效而 E 有效，把引脚信号送到芯片内相关的中断、定时或串行口。

4. 串行口 8051 系列单片机有一个全双工的串行 I/O 口，其接收数据线 RxD 和发送数据线 TxD 分别占用 **P3** 口的 2 位引脚。

5. 中断系统 8051 系列单片机可以接收处理 5 个中断源，即定时器/计数器中断源 2 个、外部中断源 2 个、串行中断源 1 个。

6. 定时器/计数器 芯片内部有 2 个 16 位的定时器/计数器 **T1** 和 **T0**，通过编程可以将它们组合成 13 位或 8 位的定时器/计数器。

1.1.2 引脚功能

1. P00～P07 / AD0～AD7 分时复用的低 8 位地址/ 数据线

（Address bus / Data bus）三态，双向，未接外部存贮器时是 8 位并行口。接有外部存贮器时分时复用输出低 8 位地址或双向数据线。

三态是指逻辑高电平、低电平和断开三种状态。

2. P10～P17 三态，双向，8 位并行接口可接外设。

3. P20～P27 / A8～A15 三态，未接外部存贮器时是双向 8 位并行口。接有外部存贮器时输出高 8 位地址线。

4. P30～P37 三态，双向，二个功能：第一功能为通用的并行 I/O 口，第二功能在串行通信、外部中断、定时器/计数器使用以及为访问外部 RAM 提供控制信号。

5. 其它控制线或电源复用线

RST (Reset) 复位信号，输入，一般外接 RC 电路和复位按键，利用上电或按动复位按键输入的正脉冲使单片机初始化，使(**PC**)＝0000H 从程序存贮器的 0000H 单元取监控程序第一条指令的操作码、堆栈指针 **SP**＝07H 以及所有特殊功能寄存器 **SFR** 的初值均为 0。

X2，X1 时钟信号，输入，在此外接频率为 4～12 MHz 的石英晶体。

$\overline{\text{EA}}$ **/VPP**（External Access）读外部 **ROM** 允许/ **EPROM** 编程电源

第一功能当 $\overline{\text{EA}}$ ＝1 时访问从内部 **ROM** 开始，若地址范围超出了内部 **ROM** 的最大容量时将自动转向外部 **ROM**。当 $\overline{\text{EA}}$ ＝0 时访问从外部 **ROM** 开始。8051 单片机配置有内部 4KB **ROM**，因此 $\overline{\text{EA}}$ 接 5V；只有在无内部 **ROM** 的 8031 单片机必须接外部 **ROM** 时才令 $\overline{\text{EA}}$ 接地。

采用 8751 单片机时利用该引脚的第二功能 **VPP** 外接 12～25V 直流电压为固化内部 **EEROM** 提供电源。

ALE/ $\overline{\text{PROG}}$ 低 8 位地址锁存允许/ **EEPROM** 编程脉冲输入端

第一功能 **ALE**（Address Latch Enable）当该管脚输出由高到低的下降沿

时将 **P0** 口输出的低 8 位地址锁存到外部地址锁存器中。第二功能 $\overline{\text{PROG}}$（Program）用于固化内部 **EEPROM** 时从该引脚输入编程脉冲信号。

$\overline{\text{PSEN}}$（Peripheral Storage Enable）外部 **ROM** 读选通信号 低电平有效，该引脚连接到外部 **ROM** 芯片的输出允许端 $\overline{\text{OE}}$。信号有效时，外部 **ROM** 被选中的存贮单元的内容出现在外部数据线上，等待 **CPU** 读入。

6. 电源线 **Vcc** 接＋5V， **GND** 接地。

1.2 MCS－51 单片机的存贮器组织

计算机把将要执行的程序和数据存贮在存贮器中，一旦开始工作，将一条指令接一条指令地取出加以执行。存贮器一般采用半导体存贮器，每个单元以字节为单位存放 8 位二进制数据，存贮器的最大容量 M 由地址线位数 n 决定，即 $M=2^n$。MCS－51 系列单片机主要特征如表 1-1 所示。

芯片内部 4K 字节的“只读性”**ROM** 区用以存放程序、原始数据和表格，所以称为程序存贮器；芯片内部 256 字节的“可读写”**RAM** 区只有地址在 00H～7FH 这 128 个字节单元可供用户使用，而高 128 个字节单元中的一部分被“特殊功能寄存器”**SFR** 占用。某些单片机内部无 **ROM** 区或者内部 **ROM**、**RAM** 不够用时常常需要在单片机外部扩展存贮器。使用内存不仅速度快、而且指令也更丰富。

表 1-1 MCS－51 单片机系列

器件名称	无 ROM 型	EPROM 型	ROM 字节	RAM 字节	16 位定时器	电路类型
8051	8031	（8751）	4K	128	2	HMOSⅠ
8051AH	8031AH	8517H	4K	128	2	HMOSⅡ
8052AH	8032AH	8752BH	8K	256	3	HMOSⅡ
80C51BH	80C31BH	87C51	4K	128	2	CHMOS

MCS－51 系列单片机存贮器组织可以分成四个独立的存贮器空间，它们是：

⑴ 256 字节内部 **RAM** 区。

⑵ 4K 字节内部 **ROM** 区。

⑶ 64K 字节的外部扩展 **RAM** 区（地址为 0000H～FFFFH）。

⑷ 64K 字节的外部扩展 **ROM** 区（地址为 0000H～FFFFH）。

上述四个存贮器空间地址都是从 0000H 或 00H 开始的，它们必然存在地址重叠现象，可以通过以下方法加以区别：

⑴ 用 **MOV** 指令访问内部数据存贮器 **RAM**。

⑵ 用 **MOVX** 指令访问外部数据存贮器 **RAM**，该指令产生控制信号 $\overline{\text{RD}}$ 和 $\overline{\text{WR}}$ 读写外部 **RAM** 内容。

⑶ 用 **MOVC** 指令读取内部或外部程序存贮器 **ROM**，访问外部 **ROM** 时该

指令产生$\overline{\text{PSEN}}$信号。外部存贮器的扩展如图 1-4 所示。

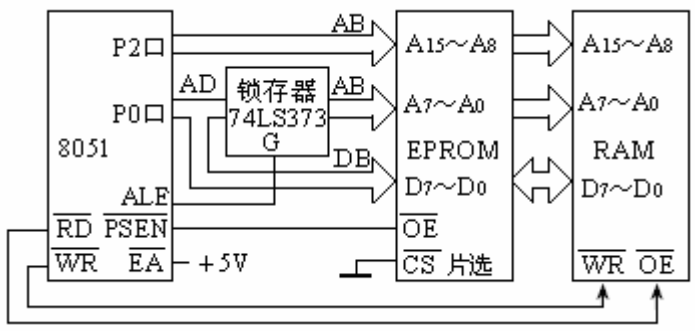
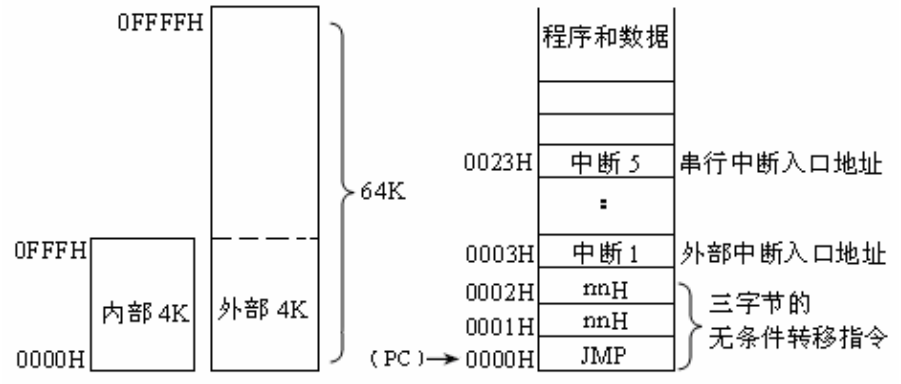


图 1-4　8051 的外部存贮器扩展

1.2.1　程序存贮器

访问程序存贮器以程序计数器 **PC** 为地址指针。8051 的程序存贮器结构如图 1-5 所示，图(a)是内部带有 4K**ROM** 的单片机，超过 4K 字节的地址会使$\overline{\text{PSEN}}$有效并自动转到外部 **ROM**；图(b)表示程序存贮器地址 0003H～0023H 单元作为系统中断向量表存放了 5 个中断处理程序的入口地址，其后存放的是程序和数据。单片机复位后 **PC** 指向 0 地址，该地址存放了一条三字节的无条件转移指令，其目的是跳过中断向量表转到程序和数据区。



(a) 程序存贮器结构　　(b) 程序存贮器地址分配

图 1-5　8051 的程序存贮器结构

1.2.2　内部数据存贮器 RAM 和堆栈

内部 **RAM** 低地址 00H～7FH 这 128 个字节是供用户使用的，分为四部分：工作寄存器区、可位寻址区、堆栈区和数据缓冲区，如图 1-6 所示。最左边一列以及最右边一列分别是用十六进制数和十进制数表示的字节地址，可位寻址区的斜体数字表示位地址。访问内部数据存贮器用 8 位寄存器 **R0**、**R1** 作为地址指针；而访问外部数据存贮器是通过 16 位的 **DPTR**（Data Pointer）作为地

址指针。

	7FH									**127**
	. .		**数**	**堆据**	**栈缓**	**区冲**	**区**			**. .**
	30H									**48**
	2FH	*7F*	*7E*	*7D*	*7C*	*7B*	*7A*	*79*	*78*	**47**
	：	**可**	**位**	**寻**	**址**	**区**		**：**		
	20H	07	*06*	*05*	*04*	*03*	*02*	*01*	*00*	**32**
R7	**1FH**									**31**
	..	**RS_1RS_0=11 工作寄存器区 3**								**..**
R0	**18H**									**24**
R7	**17H**									**23**
	..	**RS_1RS_0=10 工作寄存器区 2**								**..**
R0	**10H**									**16**
R7	**0FH**									**15**
	..	**RS_1RS_0=01 工作寄存器区 1**								**..**
R0	**08H**									**8**
R7	**07H**									**7**
	..	**RS_1RS_0=00 工作寄存器区 0**								**..**
R0	**00H**									**0**

图 1-6　内部 **RAM** 分区图

1. 工作寄存器区 0～1FH 这 32 个字节是工作寄存器区，分成 0、1、2、3 四个区。每个区的 8 个字节按照地址从低到高顺序被称为寄存器 **R0～R7**。编写程序使用寄存器只能按区使用，究竟使用哪一个区，通过位操作指令对程序状态字 **PSW** 中的 **RS1**、**RS0** 的设置进行确定。复位时，**PSW** 的值等于 00H，故复位后自动使用 0 区。

2. 位寻址区　20H～2FH 这 16 字节是位寻址区，或布尔处理器区。每个单元有 8 位，位地址从 00H，01H，…，7FH 共 128 位。位地址也可以表示成某字节的某一位，例如，位地址 0～7 可以写成 20H. 0～20H. 7。用户可以定义位寻址区中的某一位或几位作为程序中的判断标志。

3. 数据缓冲区　数据缓冲区用于存放运算数据和结果。实际上，不使用的工作寄存器区和可以位寻址的字节都可以作为数据缓冲区使用。

4. 堆栈区　用户可以将 30H～7FH 范围的最高若干字节定义成堆栈区，例如指令 MOV SP，#5FH ，定义了从 60H～7FH 单元都是堆栈区。

堆栈是在内存 **RAM** 中开辟的一个连续存贮区，它的一端是固定的，另一端是浮动的。所有数据的存入或取出只能从浮动的一端进行且符合后进先出的

原则。将浮动一端的地址称作栈顶位置用堆栈指针 **SP**（Stack Pointer）表示，由于复位时 **SP** 的值为 07H，此地址与工作寄存器区地址冲突，为了充分利用各寄存器组，最好把堆栈选择在内部 **RAM** 高地址区。

编程时经常遇到某个寄存器要在多处使用，可以将寄存器中暂时不用的数据保存在堆栈中，等该寄存器空闲时再恢复原先的数据。图 1-7 是数据指针 **DPTR** 内容 0123H 进栈和出栈的操作过程。

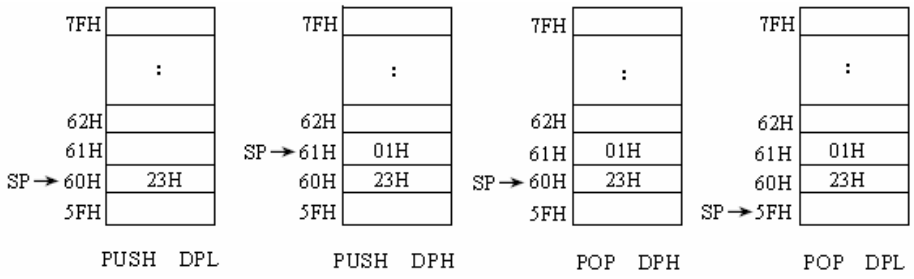


图 1-7　进栈和出栈的操作过程

进栈　　PUSH　DPL　　；SP＝SP＋1 ，（DPL）→SP
出栈　　POP　　DPL　　；（SP）→DPL，SP＝SP－1

上述两条指令的执行过程是：进栈时，首先 **SP**＝**SP**＋1 表示 **SP** 向高地址方向移动一个单元（地址向上增长），然后 **DPL** 中数据放入该单元；出栈时，先把当前 **SP** 所指单元中的数据读出到 **DPL** 中，然后 **SP** 向低地址方向移动一个单元，编程中进栈和出栈操作应成对出现。堆栈还用于中断或子程序调用时保存返回地址和“现场”，有关概念在后面介绍。

1.2.3 内部特殊功能寄存器（**SFR-**Special Function Register）

单片机的四个 I/O 接口 **P3～P0**、串/并口、中断控制器和定时器/计数器各自有多种工作方式，可以通过编写初始化程序的方法确定它们的工作方式并记录在工作方式寄存器中，这样的功能部件被称作是“可编程的”。把四个 I/O 接口 **P3～P0** 以及具有特殊功能的寄存器统一放在内部 **RAM** 高地址 80H～FFH 这 128 个字节中，将寄存器地址与数据缓冲区地址统一编址，就可以用访问数据缓冲区的指令访问寄存器。特殊功能寄存器地址分布如图 1.8 所示，图中右边一列地址能被 8 整除的是可以位寻址的寄存器。

								F8H
							B	F0H
								E8H
							ACC	E0H
								D8H
							PSW	D0H
								C8H
								C0H
							IP	B8H
							P_3	B0H
							IE	A8H
							P_2	A0H
						SBUF	SCON	98H
							P_1	90H
		TH_0	TL_0	TH_1	TL_1	TMOD	TCON	88H
PCON				DPH	DPL	SP	P_0	80H

可位寻址的寄存器 ↑

图 1-8 **SFR** 的地址分布图

从图 1-8 中看到，系统定义的 21 个特殊功能寄存器分布在地址不连续的各处，其中不用的字节也不允许用户使用。特殊寄存器的名称列于表 1-2，它们的工作原理将在后面相关章节中讨论。提示：程序计数器 **PC** 不在特殊寄存器的名单中，只有系统监控程序可以访问 **PC**，一般用户无权访问。

表 1-2 8051 特殊寄存器的符号与名称对应关系

符号	名 称	符 号	名 称
ACC	A 累加器	**PCON**	电源控制寄存器
B	B 寄存器	**SP**	堆栈指针
DPL	数据指针 DPTR 的低字节	**SBUF**	串行数据缓冲器
DPH	数据指针 DPTR 的高字节	**SCON**	串行口控制寄存器
IE	中断允许寄存器	**TCON**	定时器/计数器控制寄存器
IP	中断优先级寄存器	**TMOD**	定时器/计数器方式寄存器
P0	P0 口	**TL0**	定时器/计数器 T0 的低字节
P1	P1 口	**TH0**	定时器/计数器 T0 的高字节
P2	P2 口	**TL1**	定时器/计数器 T1 的低字节
P3	P3 口	**TH1**	定时器/计数器 T1 的高字节
PSW	程序状态字		

1.2.4 布尔处理器

MCS-51 系列单片机不仅可以进行位寻址，它还有位操作的一套指令，如：位置位、位复位、位取反、逢 1 转移、逢 0 转移等，它把 **PSW** 中的进位标志位 **Cy** 作为位累加器 **C** 进行位运算，因此，常把它称作 1 位计算机。通过下面例子的对比了解到，根据指令所用寄存器的位数就可以确定传送数据的位数，从而判断出执行的是位运算还是字节运算。

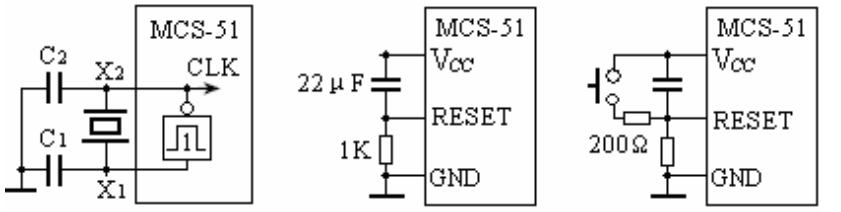
```
MOV C，0E0H  ；C 是位累加器，将位地址 0E0H 中的 ACC. 0 送到 C 中。
MOV B，0E0H  ；B 是 8 位寄存器，把字节地址 0E0H 中 A 内容送到 B。
```

1.3 MCS-51 单片机的时序

单片机的主要任务是做控制器用，其工作过程就是执行指令的过程，对指令采用串行解释方式，取指令、执行指令、再取指令、再执行指令、…，总是一条指令执行完了再取下一条指令，直到整个程序执行完毕，这种工作方式的优点是控制简单。

机器周期——机器操作时重复一系列事件所需的最短时间称作机器周期。单片机在取指、执行指令、存取数据、访问外设等不同事件中处理的时间长短不一，需要长时间处理的事件可以分配多个机器周期。

机器周期由若干个时钟周期组成，时钟信号 **CLK** 的频率（主频）越高、机器处理事件的速度越快。所有 MCS－51 单片机都有一个时钟发生器电路，图 1-9(a)是内部时钟方式，它在管脚 X2X1 外接晶体振荡器和电容构成的并联谐振起到正反馈移相作用，维持时钟发生器的自激振荡，时钟频率与晶振频率相同。图 1-9(b)和(c)是上电复位和手动复位电路，阻容延迟电路使 **RST** 引脚持续 2μs 高电平保证了可靠复位。



(a)内部时钟方式（外接晶体）　　(b)上电复位　　(c)手动按钮复位

图 1-9　时钟信号的产生

1.3.1 指令时序

MCS－51 单片机的一个机器周期由 12 个时钟周期组成，当时钟信号的频率为 12MHz 时，机器周期为 1μs。执行指令可以分解为取指→指令译码→寻址（计算并输出操作数地址）→取操作数→存结果等多个工作状态。单片机把一个机器周期分成了 6 个状态周期 S1～S6，每个状态又由二个时钟周期 P1P2

组成。图 1-10 是执行指令时的时序图，无论是否需要、地址锁存信号 ALE 在每个机器周期有两次有效故两次取指、它是时钟信号的 6 分频、常用作外设接口的时钟信号。如果执行的指令只有一字节，则第二次取指操作无效、程序计数器 **PC** 的值不变。指令功能不同、操作数的位置不同决定了指令码的字节数以及编码不同，它们都会影响执行指令的时间。

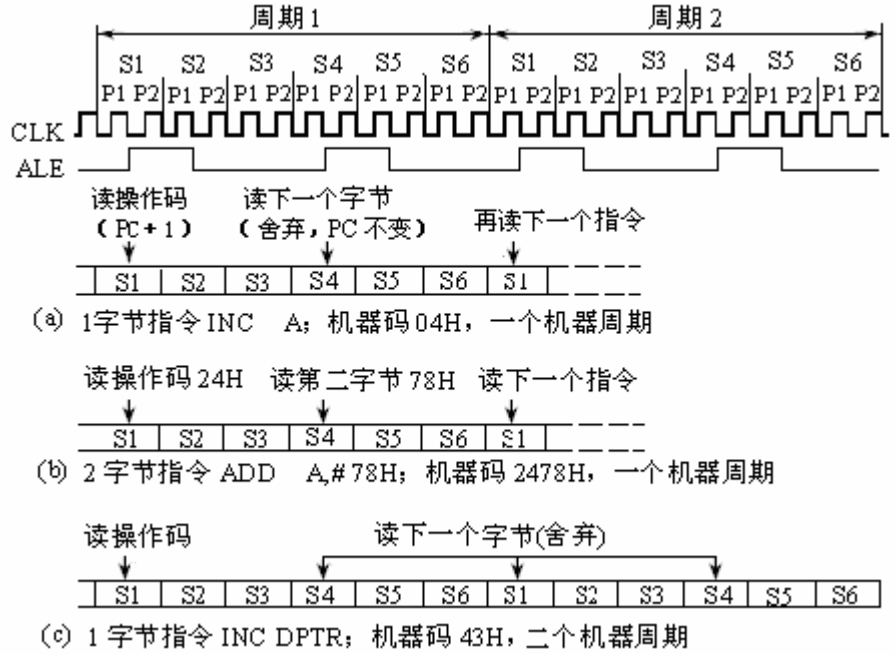


图 1-10　MCS-51 单片机指令时序

图 1-10(a)是执行一字节单周期指令“INC A”的时序图，功能是把寄存器 A 的值加 1 后仍送回到 A。只需在 S1 取操作码 S6 后结束指令的执行。

图 1-10(b)是执行二字节单周期指令“ADD A，#78H”的时序图，功能是把寄存器 A 的值加 78H 后仍送回到 A。不仅在 S1 取操作码，还要在 S4 取第二字节操作数#78H 并做加法，S6 后结束指令的执行。

图 1-10(c)是执行一字节双周期指令“INC DPTR”，功能是把数据指针 DPTR 的值加 1 后送回到 **DPTR**。在 S1 时读一次机器码，执行两次求和，第一次对低字节 **DPL** 加 1，如果有溢出，还应第二次对高字节 **DPH** 加 1，故花费二个机器周期。

1.3.2 总线时序

设指令 “MOVX A，@R1”取自外部程序存贮器，指令功能是读取外部 **RAM** 中的数据送到累加器 **A** 中，数据在外部 **RAM** 中的地址存放在寄存器 **R1** 中，执行该指令需要两个机器周期，出现在 **P0** 口和 **P2** 口的地址/数据线以及控制信号 $\overline{\text{PSEN}}$、$\overline{\text{RD}}$ 和 $\overline{\text{WR}}$ 波形图如 1-11 所示。

第一个机器周期完成取指操作，**ALE** 有效两次，$\overline{\text{PSEN}}$ 有效一次。**ALE** 第一次有效时锁存来自 **P0** 口的指令低地址 **CPL**，**P2** 口的指令高地址 **CPH**；在其后的 S3 和 P1S4 期间 $\overline{\text{PSEN}}$ 有效，指令码经过 **P0** 口送到单片机内部，**P2** 口地址不变。**ALE** 第二次有效时锁存来自 **P0** 口的数据低地址 **DPL**、**P2** 口传送数据高地址 **DPH**。**P0** 口从传送指令地址转换到传送指令码期间，有一个高阻态。

第二个机器周期完成取数操作，**ALE** 和 $\overline{\text{PSEN}}$ 信号有效但所发生的操作是舍弃的。在 S1～S3 期间 $\overline{\text{RD}}$ 或 $\overline{\text{WR}}$ 有效，数据经过 **P0** 口传送。图中“※”表示 $\overline{\text{RD}}$ 有效时 **P0** 口经过高阻态的一段时间的等待才从外部数据区读到数据；$\overline{\text{WR}}$ 有效时无需高阻态，**CPU** 就主动把输出的数据放到 **P0** 口。

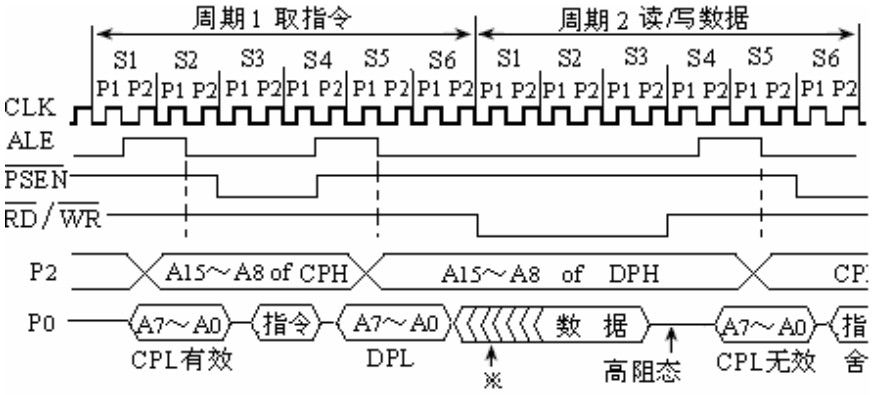


图 1-11　MCS－51 单片机总线时序

1.3.3 复位时序

在 MCS－51 单片机复位信号 **RST** 连接到芯片内部的一个施密特触发器输入端，每个机器周期的 S5P2 采样外部 **RST** 信号，当采样到高电平之后，各口引脚仍维持它们的现行状态长达 19～31 个时钟周期。因此要求 **RST** 高电平应至少维持两个机器周期、在第二个机器周期 S4 时开始内部复位操作，复位之后 **ALE**、$\overline{\text{PSEN}}$ 无效，各口锁存器初始化为 FFH、堆栈指针 **SP** 为 07H、而 **SBUF** 不确定，其它专用寄存器均被写成 00H。上电与复位差不多，复位时序如图 1-12 所示。

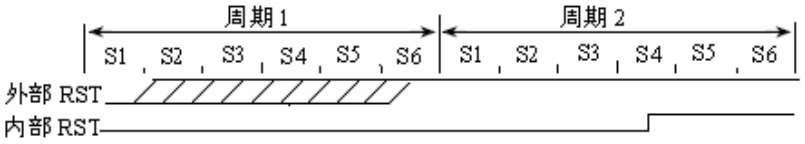


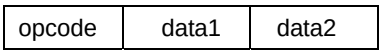
图 1-12　复位时序

第二章　MCS－51 单片机的指令系统以及汇编语言程序设计

指令是规定计算机操作类型以及操作数的一组字符，一种型号的计算机所能识别和执行的全部指令的集合称为该计算机的指令系统。指令与硬件有关，不同系列计算机的指令系统是不相同的、同一系列计算机的指令系统一般向上兼容。

单片机由于体积小、控制简单、内部没有编译程序，因此不能识别高级语言或汇编语言而只能识别以二进制代码形式表示的机器语言。但机器语言可读性差、编程效率低，因此，厂家往往在产品出厂前就把系统管理程序（又被称作监控程序）写入芯片。对于用户程序，人们普遍的做法是通过仿真系统对汇编语言编写的源程序进行调试获得机器语言，然后写入芯片内部或外部的 EEPROM 中供使用。

汇编语言是用具有一定含义的助记符来表示操作命令的低级程序语言，它与机器语言很接近。指令形式由操作码和操作数两部分组成，操作码规定了计算机的操作类型，操作数指出操作数的值或操作数的地址。有些指令不需要操作数，有些则需要一个、二个或三个操作数，常用的二个操作数指令的形式如图 2-1 所示。



操作码　目的操作数　源操作数

图 2-1 二个操作数的指令，功能是 data1＝data1 op data2，data2 不变。

2.1 寻址方式

寻址就是寻找操作数地址。操作数可以放在程序存贮器或数据存贮器中，也可以放在寄存器或外设中。如果把一个 16 位地址放在指令中会使指令变长、占用存贮器字节增多；含有多个操作数的地址可能兼有几种寻址方式、涉及多个寻址空间。一般规定，在多个操作数中至少应该有一个操作数在单片机内。MCS－51 单片机有七种寻址方式，这为程序员编程带来很大方便。

1. 立即寻址（符号为 # ）

指令的源操作数是立即数，目的操作数在寄存器或内部 **RAM** 区中，常用于给寄存器赋初值。立即寻址省去了取操作数时间。例如

```
MOV   A，#10H          ；执行指令后寄存器 A 的内容 (A)＝10H
MOV   DPTR，#8700H ；寄存器 DPTR 的内容 (DPTR)＝8700H
MOV   23H，#10H        ；内部 RAM 地址是 23H 的内容是 10H
```

“# ”表示其后的数是立即数，**不带任何标记的数表示其后是操作数地址。**立即数作为指令的一部分存放在程序存贮器中，数据的高字节放在高地址、低字节放在低地址，如图 2-2 所示。

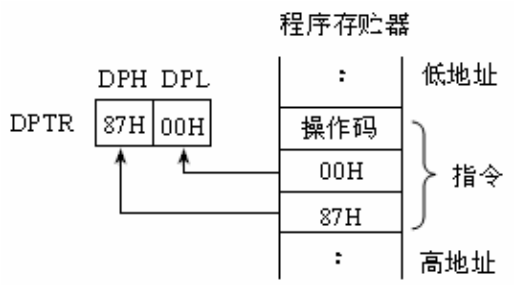


图 2-2　数据存放方式

2. 寄存器寻址 至少要有一个操作数在寄存器中，执行指令速度快。

```
INC    DPTR          ；自加 1（DPTR)＝(DPTR)＋1
MOV    R0，R1        ；传送（R0)＝(R1)，(R1)不变
ADD    A，#23H       ；加法（A）＝(A)＋23H
SWAP   A     ；高低四位互换，执行前(A)＝23H，执行后(A)＝32H
```

3. 直接寻址 指令中直接给出操作数在内部 **RAM** 区的字节地址。

```
MOV    21H，R1       ；(21H)＝(R1)，(R1)不变
ADD    A，23H        ；(A)＝(A)＋(23H)，(23H)不变
ADD    0E0H，23H     ；(A)＝(A)＋(23H)，(23H)不变
```

后二条指令功能相同，操作数用“A”表示属于寄存器寻址；操作数用累加器 A 的地址“0E0H”表示属于直接地址，但后一种表示可读性差。

4. 寄存器间接寻址（符号为 @ ）

寄存器内容指出操作数在存贮器中的地址，符号“@”表示这个寄存器被用作地址指针。内部 **RAM** 的 8 位地址选用寄存器 **R1**、**R0** 或 **SP**；外部数据存贮器的 16 位地址选用数据指针 **DPTR**。

```
MOV    @R0，A        ；((R0))＝(A )，(A )不变
                     (R0)是内存地址，((R0))是该地址中的内容
PUSH   A             ；（SP)＝(SP)＋1，((SP))＝(A)
MOVX   A，@DPTR；(A )＝((DPTR ))，((DPTR )) 不变
```

5. 基址加变址的寄存器间接寻址（符号为 @ ）

用基址寄存器 **DPTR** 或 **PC** 加上变址寄存器 A 的“两寄存器内容之和”指出操作数在程序存贮器的地址，通常简称“基址加变址”。主要用于读出程序存贮器中的检索表，指令形式为

```
MOVC   A，  @A＋DPTR ；(A)＝((A )＋(DPTR))
MOVC   A，  @A＋PC   ；(A)＝((A )＋(PC ))
```

例如，通过下面三条指令读出程序存贮器 8500H 单元中的一字节数据：

```
MOV   DPTR，#8500H      ；赋地址
CLR   A                 ；下一条指令格式要求 A 是 0
MOVC  A，@A+DPTR        ；数据装入 A
```

例 2-1 图 2-3 是汇编语言源程序中定义的一个数据段，表中数据是数码管显示一位十进制数的字形码，伪指令“DB”说明每个字形码是一字节。若寄存器 A 的内容是 5，执行后两条指令后实现了查表并换码，使 A 的内容是 5 的字形码“6DH”。

```
LEDSEG：DB 3FH, 06H, 5BH, 4FH, 66H, 6DH, 7DH, 07H；  ’0,1,2,3,4,5,6,7’
        MOV   A，5
        MOV   DPTR，# LEDSEG   ；赋表基址
        MOVC  A，  @A＋DPTR    ；查表
```

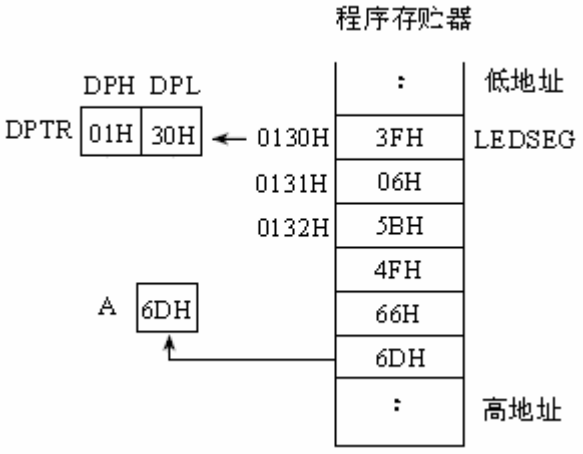


图 2-3　从程序存贮器 LEDSEG 表中读出数据

例 2-2 用 **PC** 作为基址寄存器实现上例换码的另一用法是：将定义的字形码表放在子程序 RET 指令后，设子程序放在 **ROM** 中 0FFFH 开始的地方，执行“MOV　A，@A＋PC”指令时，PC 已指向了下一条指令 RET，即(PC)＝1001H，(A)＝6，故 A＋PC＝1111H 单元中的内容为 6DH。

```
        主程序
          :
MOV   A，＃5          ；赋值将显示的数
LCALL   DISP          ；调用显示子程序
          :
        显示子程序
DISP：
INC   A      ；  A 自加 1，指令代码[04H]
MOVC A，@A＋PC          ；查表[83H]
RET                   ；返回主程序[22H]
DB   3FH              ；以下为字形码表
DB   06H
DB   5BH
DB   4FH
DB   66H
DB   6DH
      :
```

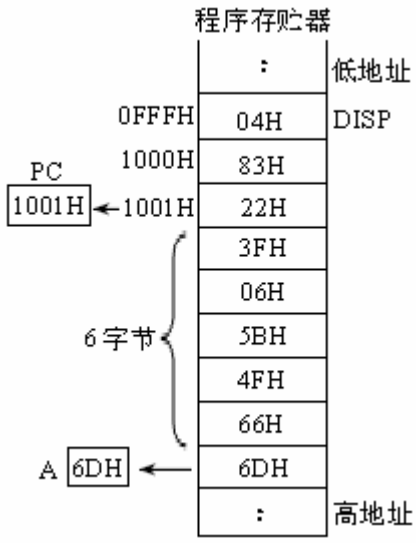


图 2-4 @A＋PC 表示程序存贮器表中数据

6.相对寻址

相对寻址常用于转移指令，下面两条指令中转移的目标地址“LOOP1”和“AGAIN”都是相对于 **PC** 当前地址作转移，相对转移的偏移量 △＝（目标地址－**PC** 当前地址）。若目标地址在转移指令之前则向前转移，偏移量△是负数；目标地址在转移指令之后则应向后转移，偏移量△是正数。

```
SJMP   AGAIN                  ；短转移到 AGAIN
CJNE   @R0，＃00H，LOOP1  ；当((R0))≠0 转移到 LOOP1，
                              否则执行下条指令。
```

7. 位寻址

操作数在内部 **RAM** 可位寻址区或 **SFR** 中可位寻址寄存器的某一位。位地址可以通过图 1-6 查到，通常位地址有多种表示形式。

```
JB   ACC. 7，NEXT   ；当 ACC. 7 位为 1 时转移到 NEXT
 CLR   C              ；清进位标志位，位于 PSW. 7
SETB   TR0            ；置 TR0 为 1，启动计数器 T0 开始计数
SETB   TCON. 4        ；用寄存器名加位数表示 T0 的运行控制位 TR0
SETB   88 H . 4       ；用单元地址加位数表示 T0 的运行控制位 TR0
```

2.2 指令系统

MCS－51 的指令系统有以下五类：①数据传送；②算术运算；③逻辑和移位；④位操作；⑤控制转移。每一类指令表中的“字节数”为指令代码的长度、“周期数”为执行指令所需的机器周期数、“C”、“OV”、“AC”栏表示执行指令是否影响程序状态字的标志位，“√”表示有影响。

下面表中对指令描述约定如下：

Rn	表示可选定寄存器组的工作寄存器 **R0～R7**，地址 00H～1FH
Ri	作为间接寻址的地址指针只能是 **R0、R1**
# data	8 位立即数 00H～FFH
# data16	16 位立即数 0000H～FFFFH
addr16	16 位地址可以在 64K 范围寻址
addr11	11 位地址在 2K 范围寻址
direct	8 位直接地址 00H～FFH，可以是内部 **RAM** 区的某个单元或某一专用工作寄存器的地址
rel	带符号的 8 位偏移量（—128～＋127）用于相对寻址
bit	位寻址区的直接寻址位
（X）	X 地址单元中的内容或 X 作为间接寻址寄存器时所指单元的内容
T	周期

1. 数据传送指令

数据传送指令不影响程序状态字的标志位。寄存器间接寻址只能用 **@R0** 和**@R1**；片外数据传送只能用**@DPTR**、**@R0** 和**@R1** 间接寻址并且有一个操作数在累加器 A 中，数据传送指令如表 2-1。

提示：①下面是一条程序员自己创造的指令，是无效的、汇编将判错

MOVX　@DPTR，direct

②‘PUSH　ACC’用法正确，但‘PUSH　A’用法错误。

表 2-1 数据传送类指令（29 条）

指　　令	功　　能	字节	周期
MOV　A，# data	立即数送到累加器	2	1
MOV　A，Rn	寄存器送到累加器	1	1
MOV　A，direct	直接寻址字节送到累加器	2	1
MOV　A，@Ri，i=0,1	内部 RAM 送到累加器	1	1
MOV　direct，@Ri	内部 RAM 送到直接寻址单元	2	2
MOV　direct，A	累加器送到直接寻址单元	2	1
MOV　direct，Rn	寄存器送到直接寻址单元	2	2
MOV　direct，# data	立即数送到直接寻址单元	3	2
MOV　direct1，direct2	直接寻址 2 送到直接寻址 1	3	2
MOV　Rn，A	累加器送到寄存器	1	1
MOV　Rn，# data	立即数送到寄存器	2	1
MOV　Rn，direct	直接寻址单元送到寄存器	2	2
MOV　@Ri，A	累加器送到内部 RAM 单元	1	1
MOV　@Ri，# data	立即数送到内部 RAM 单元	2	1
MOV　@Ri，direct	直接寻址单元送到内部 RAM 单元	2	2
MOV DPTR，# data16	16 位立即数送到数据指针	3	2
MOVX　@DPTR，A	累加器到外部 RAM 单元（16 位地址）	1	2
MOVX　@Ri，A	累加器到外部 RAM 单元（8 位地址）	1	2
MOVX　A，@DPTR	外部 RAM 单元到累加器（16 位地址）	1	2
MOVX　A，@Ri	外部 RAM 单元到累加器（8 位地址）	1	2
MOVC A，@A＋DPTR	相对 DPTR 的程序代码送到累加器	1	2
MOVC A，@A＋PC	相对 PC 的程序代码送到累加器	1	2
PUSH　direct	直接寻址单元进栈	2	2
POP　direct	出栈到直接寻址单元	2	2
XCH　A，Rn	累加器与寄存器互换	1	1
XCH　A，direct	累加器与直接地址单元互换	1	1
XCH　A，@Ri	累加器与内部 RAM 单元互换	1	1
XCHD A，@Ri	累加器与内部 RAM 单元低 4 位互换	1	1
SWAP　A	累加器高 4 位与低 4 位互换	1	1

2. 算术运算指令

参与算术运算的数都是具有大小、正负、整数小数区别，在进行加、减、乘、除，以及十进制调整时会影响标志位，同时，算术运算指令都是以累加器 A 作为目的操作数地址。由于 INC 和 DEC 指令的操作数常表示循环次数或地

址，因此不影响标志位。所有算术运算只进行 8 位数运算，带进位加“ADDC”和带借位减“SUBB”可以实现多字节数的加减。十进制调整指令只有放在加减指令之后才有效。

例2-3 编写用8421BCD码表示的两个四位十进制数相加的程序。被加数放在程序存贮器以8500H单元开始的地方、加数放在外部**RAM**以0000H单元开始的地方、和存放在内部**RAM**地址为20H、21H、22H的三个字节单元内，所有数据都按照高字节放在高地址、低字节放在低地址顺序存放。

（注：在仿真开发系统中把仿真**RAM**作为用户的程序存储器。）

```
        ORG 0000H              ；ORG是伪指令说明程序的首地址
        AJMP START             ；无条件转移到START
        ORG 0100H
START：MOV R0，#20H            ；设置存放和的间接寻址指针R0
        MOV DPTR，#8500H       ；设置程序存储器数据指针
        CLR A
        MOVC A，@A+DPTR        ；读程序存储器数被加数低二位送R1
        MOV   R1，A
        CLR   A
        INC   DPTR
        MOVC A，@A+DPTR        ；读被加数高二位送R2
        MOV R2，A
        MOV DPTR，#0000H       ；数据指针DPTR指向外部RAM中加数
        MOVX   A，@DPTR        ；读取加数低二位存入A
        ADD A，R1              ；低二位相加
        DA A                   ；十进制数调整
        MOV @R0，A             ；低二位之和写入内部RAM的20H单元
        INC DPTR               ；外部RAM地址加1
        INC R0                 ；内部RAM的地址加1指向21H单元
        MOVX A，@DPTR          ；读取加数高二位送入A
        ADDC A，R2             ；带进位加高二位
        DA A
        MOV @R0，A             ；高二位数之和写入内部RAM的21H单元
        INC R0                 ；内部RAM地址加1指向22H单元
        CLR A
        MOV ACC. 0，C          ；高二位之和的进位写入ACC. 0
        MOV @R0，A             ；进位ACC．0存入内部RAM的22H单元
HERE：SJMP $                   ；原地踏步
        END
```

说明：$ 隐含标号代表所在指令开头地址。$ 也可以组成表达式如$－6 等，但需要对指令字节计算正确，以免转移错或转移到多字节指令的中间字节。

表 2-2 算术运算类指令（24 条）

指　　令	功　　能	字节	T	C	OV	AC
ADD A，# data	累加器“加”立即数	2	1	√	√	√
ADD A， direct	累加器“加”直接寻址字节	2	1	√	√	√
ADD　A，　Rn	累加器“加”寄存器	1	1	√	√	√
ADD　A，@Ri	累加器“加”内部 RAM 单元	1	1	√	√	√
ADDC A，# data	带进位累加器“加”立即数	2	1	√	√	√
ADDC A，direct	带进位累加器“加”直接寻址字节	2	1	√	√	√
ADDC　A，Rn	带进位累加器“加”寄存器	1	1	√	√	√
ADDC　A，@Ri	带进位累加器“加”内部 RAM	1	1	√	√	√
INC　A	累加器“加”1	1	1			
INC　Rn	寄存器“加”1	1	1			
INC　DPTR	数据指针“加”1	1	2			
INC　direct	直接寻址字节“加”1	2	1			
INC　@Ri	内部 RAM 单元“加”1	1	1			
DEC　A	累加器“减”1	1	1			
DEC　Rn	寄存器“减”1	1	1			
DEC　direct	直接寻址字节“减”1	2	1			
DEC　@Ri	内部 RAM 单元“减”1	1	1			
SUBB A，# data	带借位累加器“减”立即数	2	1	√	√	√
SUBB A，direct	带借位累加器“减”直接寻址	2	1	√	√	√
SUBB　A，　Rn	带借位累加器“减”寄存器	1	1	√	√	√
SUBB　A，@Ri	带借位累加器“减”内部 RAM	1	1	√	√	√
DA　A	十进制数调整，紧跟加减指令后用	1	1	√		
MUL　AB	A 乘 B，A 积的高位 B 积的低位	1	4	0	√	
DIV　AB	A 除以 B，A 放商、B 放余数	1	4	0	√	

3. 逻辑运算指令

参与逻辑运算的数用 1 和 0 表示一件事物的两个对立状态，因此没有大小、正负、整数小数区别，逻辑运算是对应位运算不影响标志位；带进位的移位操作往往用来判断某一位是“1”还是“0”才有意地安排把那一位移进“进位标志位 **Cy**”。一位逻辑变量 X 和“1，0”的运算规则如下：

与运算（ANL）：　1・X＝X，　0・X＝0

或运算（ORL）：　1＋X＝1，　0＋X＝X

非运算（CPL）：　/0＝1　，　/1＝0

异或　（XRL）：　1⊕1＝0，　0⊕0＝0，　0⊕1＝1

表 2-3 逻辑运算和移位类指令（24 条）

指　　令	功　　能	字节	周期	C
ANL　A，# data	累加器“与”立即数	2	1	
ANL　A，direct	累加器“与”直接寻址字节	2	1	
ANL　A，Rn	累加器“与”寄存器	1	1	
ANL　A，@Ri	累加器“与”内部 RAM 单元	1	1	
ANL　direct，　A	直接寻址字节“与”累加器	2	1	
ANL　direct，　# data	直接寻址字节“与”立即数	3	2	
ORL　A，# data	累加器“或”立即数	2	1	
ORL　A，direct	累加器“或”直接寻址字节	2	1	
ORL　A，Rn	累加器“或”寄存器	1	1	
ORL　A，@Ri	累加器“或”内部 RAM 单元	1	1	
ORL　direct，　A	直接寻址字节“或”累加器	2	1	
ORL　direct，# data	直接寻址字节“或”立即数	3	2	
XRL　A，# data	累加器“异或”立即数	2	1	
XRL　A，direct	累加器“异或”直接寻址字节	2	1	
XRL　A，Rn	累加器“异或”寄存器	1	1	
XRL　A，@Ri	累加器“异或”内部 RAM 单元	1	1	
XRL　direct，　A	直接寻址字节“异或”累加器	2	1	
XRL　direct，# data	直接寻址字节“异或”立即数	3	2	
RL　A	累加器循环左移	1	1	
RLC　A	累加器连进位标志循环左移	1	1	√
RR　A	累加器循环右移	1	1	
RRC　A	累加器连进位标志循环右移	1	1	√
CPL　A	累加器取反	1	1	
CLR　A	累加器清 0	1	1	

单片机执行部分逻辑运算指令结果如下：

ANL　A，# 0FH　　；执行前(A)＝23H，执行后(A)＝03H 屏蔽高 4 位

ORL　A，direct　　；执行前(A)＝11011001, (direct)＝10101101
　　执行后(A)＝11111101，(direct)不变

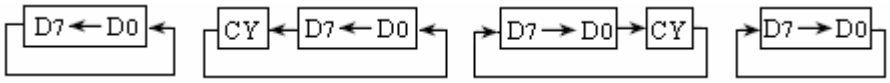
XRL　A，direct　　；执行前(A)＝11011001，(direct)＝10101101
　　执行后(A)＝01110100，(direct)不变

求反　PL　A　　；执行前(A)＝11011001，执行后(A)＝00100110

循环左移 RL　A　　；执行前(A)＝11011001，执行后(A)＝10110011

带进位循环左移 RLC　A ；执行前(A)＝11011001，执行后(A)＝1011001d
　　；(Cy)＝1，d 不确定

提示：使用带进位的循环移位指令之前必须对 Cy 置 1 或置 0，否则存在不确定位。



(a)循环左移　(b) 带进位的循环左移　(c) 带进位的循环右移　(d) 循环右移

图 2-5　循环移位

例 2-4 编写子程序，将保存在内部数据区 21H 和 20H 单元表示时间分和秒的两位压缩 BCD 码转换成分离的 BCD 码并显示在实验箱数码管上。8 位数码管的显示效果是：XX—XX。

```
        :
 DISPPS   EQU   20H          ；定义 DISPPS 为内部数据区保存显示数据的 20H 地址
；————————换码并显示字符的子程序————————
DISP_NUM：  MOV    DPTR，#LEDSEG     ；字符表首地址送到指针 DPTR
            MOVC    A，@A+DPTR       ；查表换码
            MOV     DPTR，#D8279     ；显示控制器 8279 的数据口地址送到 DPTR
            MOVX    @DPTR，A         ；  显示数据
            RET
；————————将压缩 BCD 变成分离的 BCD 并显示的子程序————————
      DISPF：MOV   R0，#DISPPS       ；R0 作为内部数据区显示数据的地址指针
        FOR：MOV    A，@R0           ；取数
             ANL    A，#0FH          ；屏蔽高四位
             LCALL  DISP_NUM         ；显示低位
             MOV    A,@R0            ；取数
             ANL    A,#0F0H          ；屏蔽低四位
             RR     A                ；右移一位
             RR     A
             RR     A
             RR     A
             LCALL  DISP_NUM         ；显示高位
             CJNE   R0，#21h，NXT    ；R0≠21h，转到 NXT
             MOV    A,#12H           ；R0＝21h 字符显示结束，高三位不显示应为 12H
             LCALL  DISP_NUM
             MOV    A,#12H
             LCALL  DISP_NUM
             MOV    A,#12H
             LCALL  DISP_NUM
             RET
```

```
    NXT：INC      R0
         MOV     A,#11H          ；‘—’的字符码是 11H
         LCALL   DISP_NUM
         SJMP    FOR
         RET
；————字符表————
LEDSEG：
    DB 3FH,06H,5BH,4FH,66H,6DH,7DH,07H      ；'0，1，2，3，4，5，6，7'
    DB 7FH,6FH,77H,7CH,39H,5EH,79H,71H      ；'8，9，A，B，C，D，E，F'
    DB 3EH,40H,00H,08H,59H,0FH,76H          ；'U，—，_，  ，I，O，P'
END
```

4. 位操作指令 表 2-4 位操作类指令（17 条）

指　　令	功　　能	字节数	T	C
MOV　C，bit	直接寻址位送 C	2	1	√
MOV　bit，C	C 送直接寻址位	2	1	
ANL　C，bit	C“与”直接寻址位	2	2	√
ANL　C，/ bit	C“与”直接寻址位的反	2	2	√
ORL　C，bit	C“或”直接寻址位	2	2	√
ORL　C，/ bit	C“或”直接寻址位的反	2	2	√
CPL　C	C 取“反”	1	1	√
CPL　bit	直接寻址位取“反”	2	1	
CLR　C	C 清 0	1	1	0
CLR　bit	直接寻址位清 0	2	1	
SETB　C	C 置 1	1	1	1
SETB　bit	直接寻址位置 1	2	1	
JB　bit，rel	直接寻址位为 1 转移	3	2	
JNB　bit，rel	直接寻址位为 0 转移	3	2	
JBC　bit，rel	直接寻址位为 1 转移并清该位	3	2	
JC　rel	进位标志为 1 转移	2	2	
JNC　rel	进位标志为 0 转移	3	2	

5. 控制程序转移类指令

能引起程序转移的指令有无条件转移、条件转移、子程序调用与返回等，MCS－51 单片机控制程序转移的指令如表 2-5 所示。为了提高编程效率，对一些重复的操作可以采取循环方式，根据条件决定继续循环还是退出循环；程序按照模块化结构设计，把不同的操作如初始化、输入数据、处理数据、输出显示等分别用一个子程序模块实现，在需要执行某个子程序时，通过子程序调用

指令“LCALL”，期间单片机不仅保存断点还将子程序的地址装入 **PC**，实现转子。

子程序还可以用中断的方式调用，X86 系列微机为用户准备了大量的以内部中断方式的 DOS 调用和 BIOS 调用。中断方式另一个重要应用是针对在实时控制中外部事件何时发生的不确定性，单片机中的中断触发器随时监视并采集外部事件发生的中断请求信号。例如，在串行通信中，当收到“有接收数据”的中断请求时，单片机将做好准备并转移到“中断处理程序”完成数据的接收工作，有关中断概念将在下章详细介绍。

特别提示：中断调用由系统通过查中断向量表自动转移到中断处理程序而不是通过用户的调用指令“LCALL”实现。

转移和调用可以用图 2-6 和图 2-7 说明：转移是在程序内部的转移、不需要保存断点和现场；子程序调用（包括中断调用）是从主程序转移到子程序，系统在转移之前自动保存好将来返回时的断点、进入子程序首先保存某些关键寄存器中原先的数据(这些数据被称之为现场),子程序结束之前应先恢复现场、最后一条指令 RET 将断点装入 **PC** 实现返回主程序。

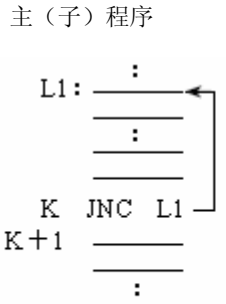


图 2-6 转移示意图

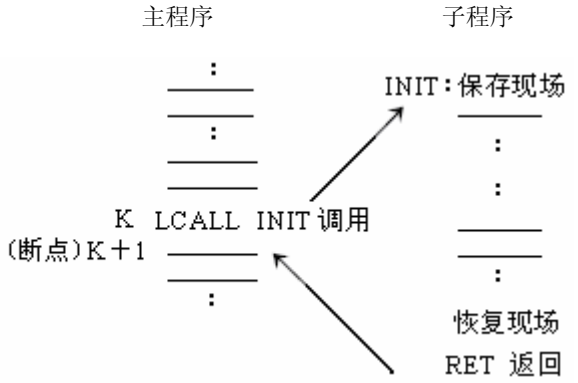


图 2-7 子程序调用示意图

下面就几条指令加以说明：

①绝对调用指令“ACALL addr11”和绝对转移“AJMP addr11”

两条指令的转移地址是 11 位，保证指令长度在 2 字节。该指令主要为了与 48 系列单片机兼容。目前的单片机内存容量很大，可以不用该指令。

②返回指令 RET 的机器码是 22H，中断返回指令 RETI 的机器码是 32H，两者操作相同：

PC15～8 ←（SP），SP←SP－1

PC 7～0 ←（SP）， SP←SP－1

③散转指令 JMP @A＋DPTR

功能：将 **DPTR** 的内容作为基地址，通过修改 A 的内容实现转移，常用于键盘操作时按下不用的按键（键值→A）就转移到不同操作的子程序。

④比较转移指令 CJNE data1，data2，rel

功能：通过减法操作实现比较，结果不修改操作数，只影响标志位 **Cy**。

若 data1≠data2，则 **PC**＝**PC**＋rel，否则，顺序执行；

若 data1＜data2，则 **Cy**＝1，否则 **Cy**＝0。

⑤循环转移指令 DJNE　Rn，rel

功能：**Rn** 放循环次数，**Rn**←**Rn**－1，若 **Rn**≠0，则 **PC**＝**PC**＋rel，否则，顺序执行。

表 2-5 控制程序转移类指令（17 条）

指　　令	功　　能	字节	周期	C
ACALL　addr11	2KB 范围内绝对调用	2	2	
LCALL　addr16	64KB 范围内长调用	3	2	
AJMP　addr11	2KB 范围内绝对转移	2	2	
LJMP　addr16	64KB 范围内长转移	3	2	
SJMP　rel	相对短转移	2	2	
JMP　@A＋DPTR	间接转移的散转指令	1	2	
NOP	空操作	1	1	
RET	从子程序返回	1	2	
RETI	从中断服务程序返回	1	2	
CJNE　A，# data，rel	累加器与立即数不等则转移到 rel	3	2	√
CJNE　A，direct，rel	累加器与直接寻址字节不等转移	3	2	√
CJNE　Rn，# data，rel	寄存器与立即数不等则转移	3	2	√
CJNE　@Ri，# data，rel	内部 RAM 与立即数不等则转移	3	2	√
DJNZ　Rn，rel	寄存器减 1 不为 0 则转移	2	2	
DJNZ　direct，rel	直接寻址单元减 1 不为 0 则转移	3	2	
JZ　rel	累加器为 0 转移	2	2	
JNZ　rel	累加器不为 0 转移	2	2	

2.3 MCS－51 单片机汇编语言程序设计

MCS－51 单片机自身没有开发功能，借助单片机开发系统进行开发。开发系统由开发系统硬件和监控程序软件组成，如本书封面所示。系统硬件有个人计算机 **PC** 机、在线仿真机（仿单片机）和实验箱；软件包括系统监控程序和用户程序，用户程序存入仿真 **RAM** 或用户 **ROM**，在实验箱硬件支持下运行用户程序。本节仅讨论汇编语言程序设计，开发系统以及开发软件的使用等放在后面章节讨论。

图 2-8 汇编程序的功能

1. 汇编语言源程序和汇编程序

为解决某一问题而设计的一系列指令组成汇编语言源程序，扩展名是“.ASM”。“能识别和翻译”汇编语言源程序并生成机器码以及其它文件的专用程序是“汇编程序”。

例 2-5 下面是在 **PC** 机和仿真机上开发软件 LCA51 对源程序 A1. ASM 经过汇编后生成的机器码及列表文件 A1. LST。程序是求两数之和。

```
地址   机内码      标号：助记符   操作数          ；注释
                     ORG    0000H          ；ORG 伪指令定义代码地址
0000   0150          AJMP   START
                     ORG    0050H
0050   7830    START：MOV    R0，#30H
0052   900000        MOV    DPTR，#0000H     ；对指针 DPTR 赋值
0055   E0            MOVX   A，@DPTR         ；取外 RAM 中被加数
0056   F9            MOV    R1，A
0057   9000A0        MOV    DPTR， #00A0H
005A   E4            CLR    A
005B   93            MOVC A，@A+DPTR         ；取 ROM 中加数
005C   29            ADD    A，R1            ；求和
005D   F6            MOV    @R0，A           ；和存到 30H 单元
005E   08            INC    R0
005F   E4            CLR    A
0060   92E0          MOV    ACC.0，C
0062   F6            MOV    @R0，A      ；进位标志 C 保存到 31H 单元
0063   80FE          SJMP   $
                     END
```

2. 汇编语言源程序的格式

汇编语言源程序由若干行组成，每行只能写一条指令，一行分为 4 个区，“[ ]”部分是可省缺的,各区之间用冒号、空格、逗号、分号分开。如：

[标号：] 操作码 [操作数 1,2,3] [； 注释]

① 标号

标号表示其“：”后面的指令所在的地址，它以字母开头、由字母、数字和符号等最多 8 个字符组成的字符串。字符之间不许有空格、标号不能是系统保

留字（如指令助记符、伪指令、特殊功能寄存器名等）。

合法的标号　START　LOOP1　NUM　NEXT　INOUT

不合法的标号　1A　ACC　JMP　EQU

② 操作码　是指令的助记符，也可以是伪指令。

③ 操作数

操作数可以是立即数、寄存器名、地址或标号，多个操作数必须用逗号“,”分开，立即数和地址用十六进制数表示时应加后缀 H、字符 A～F 打头时应前加 0 表示成“0AH”～“0FH”；二进制数表示应加后缀 b；十进制数表示应加后缀 D 或什么也不加。

④ 注释

注释是以“;”开头的简短解释语句，帮助对指令的阅读、备忘和修改。

3. 伪指令

伪指令是对汇编过程进行说明的一组命令，它不被翻译成机器码。

①指令指针定义　**ORG**

格式：　ORG　表达式

功能：把当前的指令地址指针 PC 设置成表达式所表示的值。在程序中多次使用 ORG 必须从小到大，指令代码不能重叠，表达式中的标号应在 ORG 使用之前确切定义。

举例：ORG　0050H

ORG　TLED＋0AFH

②数据字节定义　**DB**

格式：[名字]　DB 操作数

名字可有可无，操作数可以是数据、标号或字符串，若有多个操作数则之间用“,”隔开，操作数取值范围 0～0FFH。

功能：把操作数填入当前指令指针指向的字节单元。每填写一个，指令指针就加 1 。

举例：　LED　DB　3FH，06H，5BH，4FH，66H，’ABCD’

CCC　DB　LED　；LED 地址的低 8 位作为操作数

③数据字定义　**DW**

格式：[字]　DW　操作数

功能：把操作数填入当前指令指针指向的字单元，高八位先填低八位后填。每填写一个，指令指针就加 2。

举例：　WW　DB　0ABCH，06D0H，345BH

④位定义　**BIT**

格式：名字　BIT　表达式

功能：把表达式所表示的值赋给名字。

举例：WIN　BIT　0

DP　BIT　P1. 1

⑤等值定义 **EQU**

格式：名字 EQU 表达式

功能：把表达式所表示的值赋给名字。

举例：Z8279 EQU 0FF82H

LEDMOD EQU 00H

⑥源文件结束 **END** 表示源文件结束，其后字符 ASM51 不予理睬。

4. 程序结构

以图 2-9 为例，放在程序存贮器“0000H”单元的第一条指令是一条无条件转移指令，接下来是放在程序存贮器低地址的“中断服务程序入口地址”，通过该地址中的一条无条件转移指令转移到中断服务程序。主程序一般放在“0050H”开始的地方，“START”和“END”标志着程序段的开始和结束，程序段中按照主程序、子程序、数据表的顺序排开，在主程序和子程序中可以是顺序结构、分支结构、循环结构等形式组成，前面例 2-3 是顺序结构，后面例 2-6 和例 2-7 分别是分支结构和循环结构。

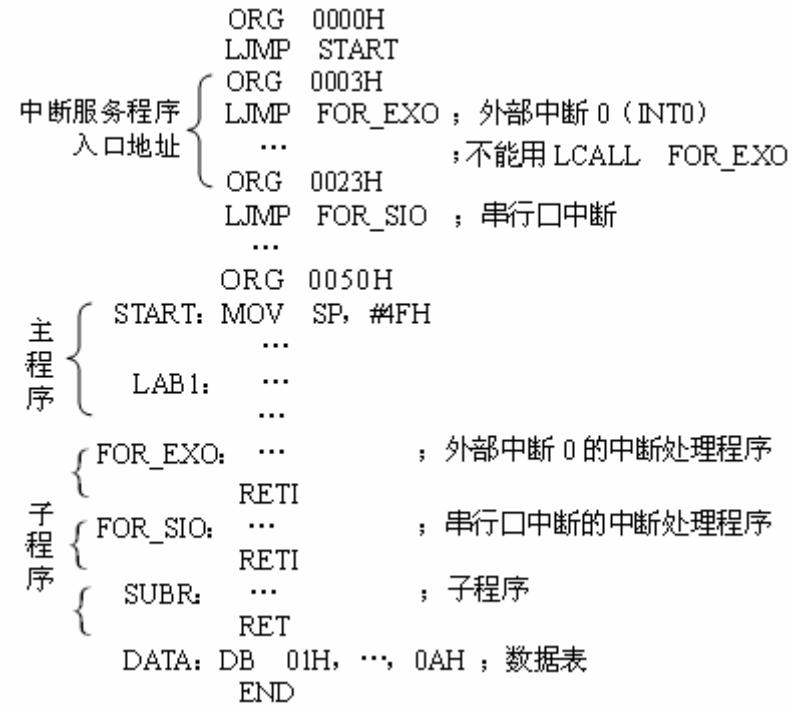


图 2-9 程序基本结构

程序设计的步骤为：

分析题意——→确定算法——→画流程图

↓

执行←——编程、调试和修改←——存贮器资源分配

例 2-6 内部 **RAM** 的 30H 单元放有符号数 X，求符号函数 Y。当 X＜0 时 Y=－1；X＞0 时 Y=1；X=0 时 Y=0。

```
        ORG    0100H
        LJMP   START
        ORG    0040H
START：MOV   R0，#30H          ；送符号数 X 的地址 30H
        MOV   A，@R0
        ANL   A，@R0           ；做与运算出标志 Z
        JZ     ZERO            ；(A)＝0 就转，否则执行下一条指令
        JNB   ACC.7，NEXT      ；ACC. 7≠1 为正数则转到 NEXT
MINUS：MOV   R1，#0FFH     ；ACC. 7＝1 为负数，－1 的补码是 0FFH
        SJMP   EXIT            ；退出
ZERO：MOV   R1，#0；(R1)＝0
       SJMP   EXIT
NEXT：MOV   R1，#1；(R1)＝1
EXIT：SJMP   EXIT
        END
```

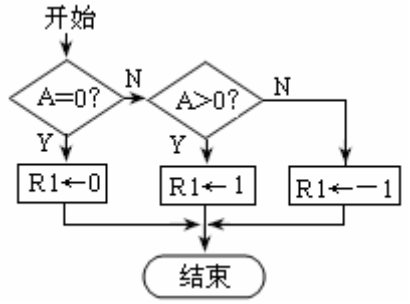


图 2-10　例 2-6 的流程图

例 2-7 求存于内部 **RAM** 的 20H 单元开始的 10 个无符号数的最小值。

解：用比较相邻两数的方法找出最小值，寄存器 A 以及内部 **RAM** 的 40H 单元存放参与比较的前后两个数，执行指令“CJNE A，40H，NEXT”的结果置标志“**C**=1”表示 **A** 中数小。设 **R0** 是数据地址指针初值为 20H、**R7** 存放循环次数 10、，41H 单元存放最终的最小值。

```
        ORG   0000H
        LJMP START
        ORG   0040H
START：MOV R0，#20H
        MOV R7，#10
        MOV A，@R0          ；取第一个数
        DEC R7
    L0：INC R0
        MOV 40H，@R0        ；取下一个数
        CJNE   A，40H，NEXT
 NEXT：JC   L1              ；A 小时 C=1 转
        MOV A，40H
    L1：DJNZ R7，L0
        MOV 41H，A
 HERE：SJMP $
        END
```

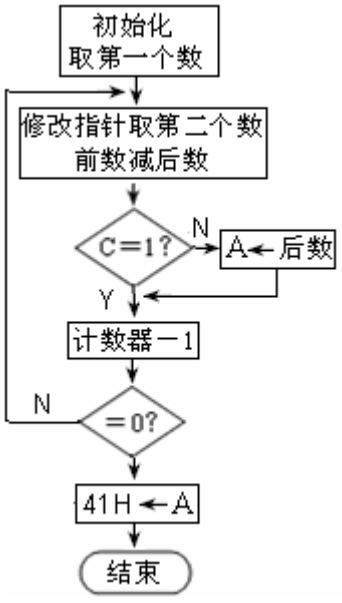


图 2-11　例 2-7 的流程图

例 2-8 子程序调用举例。单片机 **P1** 口接有 8 个发光二极管，编程使 8 个发光二极管中只有一个发光且以左循环移位方式依次使每个管发光 1 秒钟，延时一秒由子程序 DELAY 实现。（连线图略）

```
        ORG     0000H
        LJMP    START
        ORG     0040H
START：MOV SP，#60H    ；定义堆栈区
       MOV   A，#1      ；从 P1. 0 位开始亮
   L0：MOV P1，A
       LCALL   DELAY    ；调用延时 1 秒子程序
       RL   A           ；循环左移一位
       SJMP   L0
DELAY：PUSH   A
       MOV R0，#0H   ；R0 和 R1 做计数器，减 1 计数到 0 表示 1 秒
   L1：MOV   R1，#0H
       DJNZ   R1，$
       DJNZ   R0，L1
       POP   A
       RET
       END
```

第三章 MCS－51 单片机的中断、定时及串行口

中断被计算机用来处理某些不定期发生的事件。而实际生产活动中经常需要“定额包装”、设备的“延时触发”、或者通过“分频”而得到所需频率的时钟信号等等，这些与时间有关的控制通过定时器/计数器实现是普遍的做法。学习 MCS－51 单片机的中断与定时可以与 X86 系列相比较，以助理解。

3.1 中断的概念

1. 什么是中断？

CPU 正在执行某个程序时接收到中断请求，暂停正在执行的程序而转去执行该中断请求的处理程序，处理完之后又返回原先被中断的程序，这样的过程就是中断。

2. 有关中断的几个名词

①中断源——能发出中断请求的部件，一般有多个中断源。

②中断优先级——**CPU** 响应并处理中断的先后顺序，优先级越高、响应并处理越及时。

③中断嵌套——**CPU** 在执行中断处理程序的同时又响应并处理优先级更高的中断请求。8031 单片机仅支持一级中断嵌套，不支持多重中断。

④中断向量——中断处理程序的入口地址，MCS-51 单片机的中断向量位于内部 RAM 的低地址 0003 H～0023H。

⑤开中断与关中断——开中断是允许 **CPU** 响应中断、关中断是不允许 **CPU** 响应中断。在 **CPU** 处理诸如保存断点等一些关键操作时应当关中断以保证此操作的可靠进行。进一步的讨论如下：

在 X86 系列程序状态字中 D9 位“**IF**”是中断标志位，当“**IF**＝1”时开中断允许 **CPU** 响应中断，否则关中断、不允许 **CPU** 响应中断；外部中断控制器 8259 可以屏蔽外部中断请求信号。

MCS－51 单片机将管理 **CPU** 开中断、关中断以及中断请求信号、允许信号放在同一个中断允许寄存器 **IE** 中统一管理。

⑥中断响应——中断响应必须在一条指令执行结束之后，否则没有机会再去执行那未执行完的指令，因为 **PC** 已指向下一条指令的地址。中断响应要做的事情是“保存断点”（或许还有某些寄存器数据）并将“中断处理程序的入口地址”装入 **PC**。以上操作由系统自动完成，程序员涉及不到。

⑦中断处理程序——处理中断请求的子程序，由于该子程序会用到一些寄存器，而这些寄存器中的数据（现场）将来中断返回时还要使用，因此应当先保存好，一般进栈保存；然后才是中断处理程序主体，之后通过出栈恢复现场，中断处理程序的最后一条指令“RETI”将“断点装入 **PC**”实现返回断点。中断响应及处理的流程图如图 3-1 。

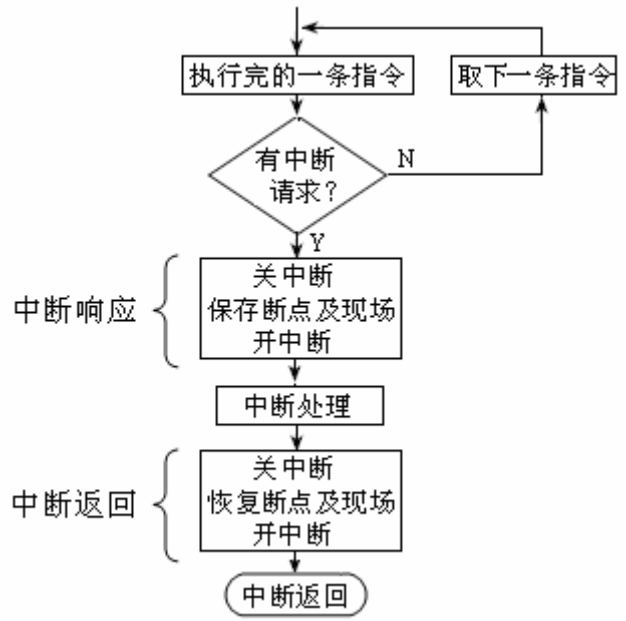


图 3-1　中断响应及处理的流程图

3.2 MCS－51 单片机的中断系统

8051/8031 型单片机提供了 5 个中断源：2 个外部中断源 **INT0** 和 **INT1** 分别由引脚 P3. 2 和 P3. 3 接入；2 个片内定时器/计数器溢出中断 **T0** 和 **T1** 将加 1 计数产生的溢出信号作为中断请求信号；1 个片内串行口的收发中断，每当接收或发送完一帧数据就产生一个中断用以提示控制器准备接收或发送下一帧数据。上述 5 个中断源的中断请求标志位、中断处理程序入口地址以及优先级如表 3-1。

表 3-1 MCS－51 单片机中断处理入口地址和优先级

中　断　源	中断请求标志位	入口地址	优先级
外部中断 **INT0**	IE0（TCON．1）	0003H	高
定时器/计数器溢出中断 **T0**	IF0（TCON．5）	000BH	
外部中断 **INT1**	IE1（TCON．3）	0013H	
定时器/计数器溢出中断 **T1**	IF1（TCON．7）	001BH	
串行口中断	RI （SCON．0） TI （SCON．1）	0023H	低

与中断有关的寄存器有中断允许寄存器 **IE**、中断优先级控制寄存器 **IP**、控制寄存器 **TCON** 和串行口控制寄存器 **SCON** 等，这些寄存器的每一位可以通过指令置 1 或置 0。**SCON** 将在介绍串行口时讨论。

1. 中断允许寄存器 IE　见表 3-2

表 3-2　中断允许寄存器 **IE** 的定义

D7	D6	D5	D4	D3	D2	D1	D0
EA	—	—	ES	ET1	EX1	ET0	EX0
总中断允许位			串行口中断允许	T1 中断允许	INT1 中断允许	T0 中断允许	INT0 中断允许

说明：

①EA＝0 时禁止一切中断，相当于 X86 系列 **PSW** 中的中断标志位 IF=0；

EA＝1 时相当于开中断，各中断源的中断允许位为 1 则开放该中断；为 0 则屏蔽该中断。

②复位后 **IE** 的内容为 0。

③ **IE.5** 留给 8052 做定时器 **T2** 的中断允许位。

2. 中断优先级控制寄存器 IP　见表 3-3

表 3-3　中断优先级寄存器 **IP** 的定义

D7	D6	D5	D4	D3	D2	D1	D0
—	—	—	PS	PT1	PX1	PT0	PX0
			串行口	T1	INT1	T0	INT0

说明：

①单片机将 5 个中断源分成二个优先级别：某一位为 1 时为高优先级；为 0 时为低优先级。仅在没有高优先级的中断请求时才响应低优先级的中断请求。

②若 5 个中断源为同一优先级，系统按照 **INT0→T0→…→**串行口顺序查询中断请求标志，先查询的优先级高、后查询的优先级低。若希望“串行口”优先级最高，可以在 **IP** 中写入 10H，这样在多个中断请求中，串行口中断请求将获得优先响应。

③复位后 **IP** 的内容为 0。

3. 控制寄存器 TCON　见表 3-4

表 3-4　控制寄存器 **TCON** 的定义

与定时器/计数器有关				与外部中断有关			
D7	D6	D5	D4	D3	D2	D1	D0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
T1 溢出标志位	T1 运行控制	T0 溢出标志位	T0 运行控制	INT1 请求标志	INT1 请求触发方式	INT0 请求标志	INT0 请求触发方式

说明：

①IE1 和 IE0——外部中断请求标志位，当 **CPU** 检测到芯片引脚 **INT1** 或 **INT0** 出现外部中断请求信号的下降沿时由硬件置位、请求中断。进入中断处理程序后，被硬件自动清 0。

②IT1 和 IT0——中断请求信号的触发类型，通过软件设置或清除。置 1 时下降沿触发；置 0 时低电平触发。低电平触发容易引入干扰。

③TF1 和 TF0——当定时器/计数器溢出时由硬件置位、请求中断。进入中断处理程序后，被硬件自动清 0。

④TR1 和 TR0——靠软件置位或清除，置位时启动定时器/计数器工作；清除时停止工作。

例 3-1　用中断方式读取 A/D 转换结果并送到 **P1** 口的 LED 显示。

分析：接线图如图 3-2 所示，从电位器 Rw 上获得一个模拟电压并送到模数转换器 **ADC0809** 的通道 IN0，当 ADC 完成一次模拟量到数字量的转换后发出中断请求信号 EOC 到单片机的 **INT1**，请求单片机取走数字量，以便下一次转换。单片机一旦接收到外部的中断请求信号 **INT1** 就保存断点并立即将入口地址“0013H”装入 **PC**，入口地址的长转移指令“LJMP　INT-1”引导到中断处理程序 INT-1。中断处理程序完成两件事：读取转换结果到 **A** 以及修改转换结果标志位为“#00H”表示转换结束，随后返回主程序，主程序将 **A** 中的转换结果送到 **P1** 口指示灯显示。

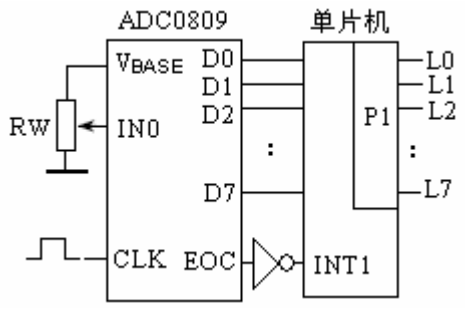


图 3-2　A/D 转换连线图及信号时序波形图

```
        ORG   0000H
        LJMP   A-D
        ORG   0013H          ；外部中断 INT1 的中断处理程序入口地址
        LJMP   INT-1
        ORG   0100H
A-D：SETB   IT1              ；置 IT1=1 为下降沿触发方式
        SETB   EA            ；总中断允许
        SETB   EX1           ；外部中断 INT1 允许
        MOV   DPTR，#8600H   ；ADC 通道 0 的地址是 8600H
```

```
        MOVX   @DPTR，A       ；一条假写命令启动 ADC
        CJNE   R7，#00H，$    ；(R7)≠0,转换未结束应等待
        MOV    P1，A          ；转换结束将结果送到 P1 口
        MOV    R7，#0FFH      ；置转换结束标志为未结束
        SJMP   A-D
INT-1：MOVX    A，@DPTR       ；读取转换结果到 A
        MOV    R7，#00H       ：置转换结束标志为结束
        RETI
        END
```

3.3 MCS－51 单片机的定时器/计数器

8051/8031 型单片机内部有 2 个 16 位的定时器/计数器 **T0** 和 **T1**，每一个定时器/计数器又可以分为 8 位的 **TH** 和 **TL**，结构原理如图 3-3 所示。

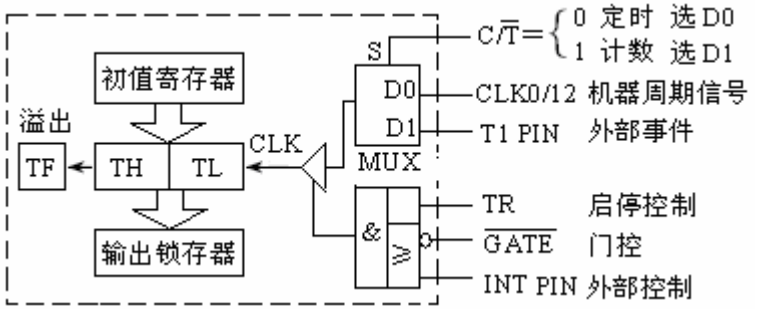


图 3-3 定时器/计数器结构原理图

1. 定时器/计数器工作原理

给定一个计数初值，计数器从初值开始对计数脉冲 CLK 加 1 计数，当计数到 0FFFFH→0000H 产生“溢出”，该“溢出”信号就是一个定时到的信号或者作为计数满的信号。如果希望计数器工作在定时状态就应输入一个周期性的脉冲信号，这样才能根据定时的长短决定计数初值，单片机中计数器在每个机器周期加 1；如果希望计数器工作在对外部事件进行计数状态，如定额包装，则从外部 **T1** 或 **T0** 引脚输入，在输入信号从 1 跳变到 0 时计数器加 1。通过选择控制 $C/\overline{T}$ 确定计数器是工作在定时状态还是工作在计数状态。从图 3-3 看到，计数脉冲 CLK 通过三态门到达计数器，其控制信号的逻辑关系式是

$$TR \bullet (\overline{GATE} + INT)$$

式中，TR 是控制寄存器 **TCON** 中与计数器有关的运行控制位，置位时启动定时器/计数器工作；清除时停止工作。

$\overline{GATE}$ 是定时器/计数器模式控制寄存器 **TMOD** 中门控信号，低电平有效。

当 **GATE**＝0 时，只要 **TR**＝1 就接通三态门与 INT 无关；当 **GATE**＝1 时必须 **TR**＝1 且 **INT**＝1 才接通三态门；

单片机中的每个计数器有 4 种操作模式，其中前三种模式对两者都是一样的，只有模式 3 对两者不同。模式的选择可以通过指令在模式控制寄存器 **TMOD** 中对 **M1M0** 设置，如表 3-5。

表 3-5　计数器 **T1**、**T0** 的四种工作模式

M1 M0	工　作　模　式
0　0	模式 0，$\mathbf{TL_X}$ 的低 5 位与 $\mathbf{TH_X}$ 的 8 位构成 13 位定时器/计数器
0　1	模式 1，$\mathbf{TL_X}$ 与 $\mathbf{TH_X}$ 构成 16 位定时器/计数器
1　0	模式 2，$\mathbf{TL_X}$ 为 8 位自动重装载定时器/计数器，$\mathbf{TH_X}$ 重装载值（分频器）
1　1	模式 3，**T0** 分成两个 8 位计数器，**T1** 停止计数

2. 模式控制寄存器 TMOD　　表 3-6

D7	D6	D5	D4	D3	D2	D1	D0
$\overline{\text{GATE}}$ 1	C/$\overline{\text{T}}$1	M1	M0	$\overline{\text{GATE}}$ 0	C/$\overline{\text{T}}$0	M1	M0

例 3-2　编写程序，使 MCS-51 单片机的内部定时器，通过 **P1** 口控制一个发光二极管每隔一秒钟亮灭一次。

分析：内部计数器用作定时器时是对机器周期信号的计数，首先我们计算一下初值是多少。每个机器周期的长度是 12 个时钟周期，若实验系统的晶振是 12 MHz，则

机器周期＝12÷11.059 MHz≈1μS

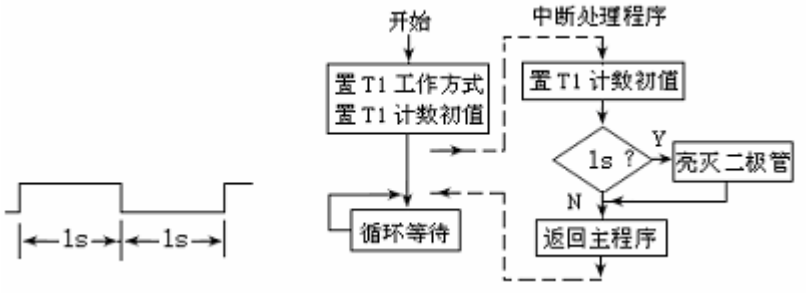
计数初值 X＝1S÷1μS＝1×10^6

由于 10^6 ＞65536 ，超出 16 位计数器范围，故重新考虑通过定时器定时 50ms 和软件循环 20 次的做法实现总定时 1s＝50ms×20 的目的。

定时长度(溢出时间)　50ms＝(65536－计数初值 X)×1μS

故　计数初值　　X＝$(15536)_{10}$＝$(3CB0)_{16}$

选择定时器 1 工作在模式 1，输出信号由芯片引脚 **P1**. 0 连接到发光二极管，由于执行程序还要花费时间，因此输出信号的变化时间比 1 秒时间长。输出波形图以及程序流程图如图 3-4 所示。



(a) 输出信号波形　　(b) 程序流程图

图 3-4 输出信号波形图以及程序流程图

```
        ORG   0000H
        LJMP  START
        ORG   001BH          ；计数器 T1 的溢出中断处理程序入口地址
        LJMP  INT- T1
        ORG   0100H
START：MOV  SP，#60H
        MOV   TMOD，#10H     ；置 T1 为模式 1
        MOV   TL1，#0B0H     ；置 T1 初值
        MOV   TH1，#3CH
        MOV   R0，#00H       ；R0 存放输出信号
        MOV   R1，#20        ；软件循环次数 20
        SETB  TR1
        SETB  EA
        SETB  ET1
        SJMP  $
；————————T1 中断处理子程序————————
INT- T1：CLR  TR1            ；停止计数
        MOV   TL1，#0B0H     ；置计数初值
        MOV   TH1，#3CH
        SETB  TR1            ；启动计数
        DJNZ  R1，EXIT       ；R1 减 1，R1 未减到 0 就转到 EXIT
        MOV   R1，#20H       ；R1 减到 0 就重新赋循环次数 20
        MOV   P1，R0         ；输出
        MOV   A，R0
        CPL   A              ；取反
        MOV   R0，A          ；存入 R0
 EXIT：RETI
        END
```

3.4 MCS－51 单片机的串行口

CPU 与外设之间或计算机与计算机之间的数据传送有并行传送和串行传送两种。

并行传送——是指 N 位数据在 N 条传输线上同时传送。传送效率高，成本也高，一般用在近程传送。主机内部数据都是并行传送。

串行传送——是指数据在一条传输线上一位一位地按顺序传送，传送效率低，成本也低，可以利用现成的电话线。串行传送一般用在远程传送，计算机与远程终端的数据传送就是串行传送。

为了保证可靠传送数据，收发数据双方还应有一些联络信号，如：请求发送、请求接收、准备发送、准备接受、已经接收等同步信号。串行传送时把数据和联络信号放在一条传输线上传送，对传送的数据格式、速度、抗干扰措施等会比并行传送有更严格的限制，下面逐一讨论。

3.4.1 串行传送基础知识

1. 串行传送的数据格式　串行传送又分为：

同步传送——数据成组传送，收发双方采用相同频率的时钟；

异步传送——数据一帧字符接着一帧字符地传送，允许收发双方时钟频率有微弱差别。

⑴异步传送的数据格式

以字符为单位传送，每帧字符以一位起始位开始、然后是 5～8 个数据位、先送低位后送高位，最后是一位校验位和 1、1. 5 或 2 位停止位，字符与字符之间是空闲位，空闲位可省略。数据格式如图 3-5 所示。

图 3-5　串行异步传送数据格式

异步传送数据格式中加入了起始位、校验位、停止位以及空闲位，这些附加位降低了传输效率，但同时也可以消化掉收发双方时钟频率的差异带来的误差，适合于低速通信。

⑵同步传送的数据格式

以几百或上千字节的数据块为单位传送，数据块的开始用 1 或 2 个同步字符表示，数据块的结束有 2 字节校验码。要求收发双方时钟频率必须相同，近程的串行同步传送加一条时钟线、远程的串行同步传送接收端通过锁相技术从

数据位中提取与发送端相同的时钟信号，适合于高速通信。数据格式如图 3-6 所示。

1～2 个同步字符	数　据　块	2 字节校验码

图 3-6　串行同步传送数据格式

2. 波特率与波特率因子

波特率——每秒钟传送二进制数码的位数（又称比特率），单位“位/秒”（bps）。常用的波特率有 600、1200、2400、4800、9600、19200 等。

位周期——每位数据传送的时间(位时间) Td＝1/波特率。

波特率因子(分频系数)——接收方采用比波特率更高频率的时钟来检测起始位，定义：检测时钟频率＝波特率×波特率因子。

通常波特率因子选为 16、64。采用较高频率的检测时钟是为了可靠识别“起始位”，用图 3-7 加以说明。如果选择波特率因子为 16，表示在对接收到的数据采样时，连续 8 个时钟脉冲采集到的都是“0”信号，就可以判断这是“起始位”而不是干扰信号，从此刻开始每经过 16 个时钟脉冲就对数据采样一次。收发双方时钟频率的微小差异有可能造成数据采样不在数据位周期的中心位置，但采样点绝对不应超出整个位周期。

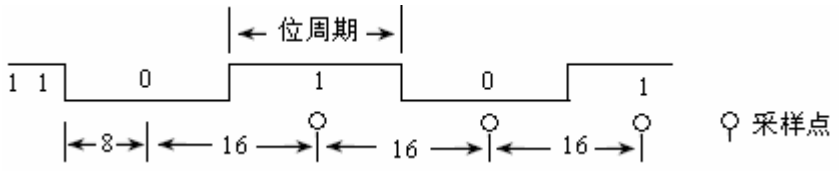


图 3-7　识别起始位

3. 数据通路的制式

不同领域传送电信号采用不同制式，有以下几种：

单工——数据在一条线上始终按照一个方向传送。如：电视接收机。

半双工——通信双方都有发送和接收能力，但只有一条传输线，只能分时地在两个方向传送数据。如：对讲机、收发报机。

全双工——通信双方分别在二条传输线上同时向两个方向传送数据。如电话、互联网。

4. 串行通信的调制与解调

计算机中数据信号电平是 TTL 型的，即电平大于等于 2. 4V 表示逻辑“1”，小于等于 0. 5V 表示逻辑“0”。这种信号在远距离（＞15 米）传输时由于受到线路幅频特性和相频特性的影响以及电磁场的干扰，信号会发生衰减和畸变，误码率会大大上升。解决这个问题的方法通常是用不同频率的载波信号代表数

字信号的“1”态和“0”态。如发送端使用调制器把数码“0”调制成 2400Hz 的正弦信号，把数码“1”调制成 1200Hz 的正弦信号送到传输线路上，在接收端再通过解调器把模拟信号还原成数字信号送到数据处理设备的。调制器和解调器合在一起构成调制解调器 **MODEN**。

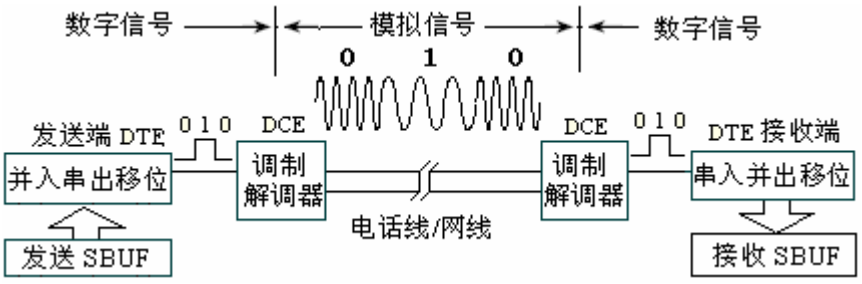


图 3-8　调制与解调

5. RS-232C 串行通信总线标准

串行通信总线标准之一 EIA-RS-232C 是由美国电子工业协会 EIA(Electronic Industry Association)于 1969 年制定的。该标准早期为促进公用电话网络进行数据通信而制定的标准，主要内容包括：

⑴信号电平

控制线/数据线	TTL 电平	RS-232C 电平
接通/ 0 空号 断开/ 1 传号	0 ～ 0.5V 2.4 ～ 5V	＋(5～15)V －(5～15)V (＋5～－5) V 噪声容限

⑵在计算机中我们只用到其中 9 个信号，它们的定义如下表：

引脚	输入/输出	信号	说　　明
1	输入	DCD	载波检测
2	输入	RxD	接收数据
3	输出	TxD	发送数据
4	输出	DTR	数据终端就绪
5		GND	信号地
6	输入	DSR	数据通信设备准备好
7	输出	RTS	请求发送
8	输入	CTS	允许发送
9	输入	RI	振铃指示

RS-232C 获得广泛应用的原因仅在于其推出较早，在现代通信中已暴露出明显缺点：

① 数据传输率低。通常异步通信速率限制在 19.2Kbps 以下，对于某些同步通信系统而言，也难以满足传送速率的要求。

② 传送距离短。RS-232C 接口之间电缆长度一般限于 15 米以内。

③ 接口处信号间容易产生串扰，抗干扰能力差。

为克服 RS-232C 接口的上述缺点，EIA 又相继推出了 RS-499，RS-422A 和 RS-423A 等标准接口。这些标准接口的性能及使用方法可参阅有关资料。

3.4.2 MCS－51 单片机的串行通信接口

MCS－51 单片机内部有一个可编程的全双工串行通信口，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用。该串行口有 4 种工作模式，片内的定时器 / 计数器可用作波特率发生器。接收、发送均可工作在查询方式或中断方式。

1. 串行通信接口结构

MCS－51 系列单片机内部的串行通信口，有二个物理上相互独立的接收、发送缓冲器 **SBUF**，对外也有两条独立的收、发信号线 **RxD**（P3.0）和 **TxD**（P3.1）。可以同时发送、接收数据，实现全双工传送。发送缓冲器和接收缓冲器不能互换，发送缓冲器只能写入不能读出，接收缓冲器只能读出不能写入。两个缓冲器占用同一个端口地址（99H）。具体对哪一个缓冲器进行操作，取决于所用的指令是发送还是接收。

接收是双缓存的，以避免在接收下一帧数据之前 **CPU** 未能及时把上一帧数据取走而产生两帧数据重叠的问题。对于发送器，因为发送时 **CPU** 是主动的，不会产生写重叠的问题，所以不需要双缓存。结构原理如图 3-9。

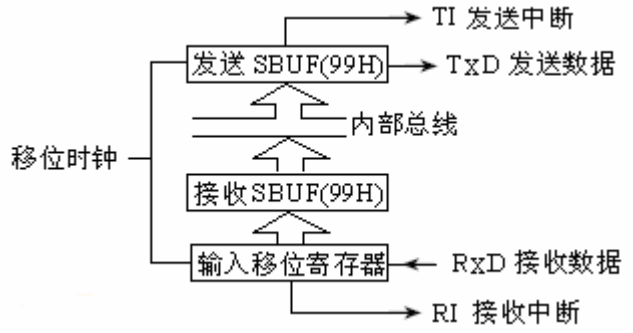


图 3-9 MCS－51 串行通信口结构原理

2. 与串行口有关的控制器

与串行通信口有关的控制寄存器有 **SCON**、**PCON**、**IE** 和定时器/计数器，用校验方式进行通信会用到程序状态字寄存器 **PSW**。

⑴**串行口控制寄存器 SCON**　表 3-7

位地址	9F	9E	9D	9C	9B	9A	99	98
位符号	SM0	SM1	SM2	REN	TB8	RB8	TI	RI

说明：

● **SM0** 和 **SM1**——串行口工作方式选择位，对应四种模式如下

SM1 SM0	功　能	波 特 率
0　0　模式 0	同步移位寄存器	*f*osc /12
0　1　模式 1	8 位 UART	可变
1　0　模式 2	9 位 UART	*f*osc /64 或 *f*osc /32
1　1　模式 3	9 位 UART	可变

模式 3 与模式 2 的操作完全一样，不同之处是模式 2 的波特率只有二种，模式 3 的波特率可由用户设定。

• **SM2**——多机通信控制位，**SM2**＝1 为多机通信，多机通信在方式 2 或方式 3 下进行；**SM2**＝0 为双机通信。

在**方式 0** 时，**SM2** 不用，应设置为 0。

在**方式 1** 时，**SM2** 一般设置为 0。若 **SM2**＝ 1 ，则把“停止位”看作接收到的第 9 位数据，只有停止位有效才会激活 **RI**，**RI** 供查询或作为串行口中断请求信号，若停止位无效，则 **RI** 清零。

在**方式 2 或方式 3** 下包括 9 位数据，必须在启动发送之前将第 9 位数值装入 **SCON** 寄存器的 **TB8** 位；而在接收时会将接收到的第 9 位数值装入 **SCON** 寄存器的 **RB8** 位。第 9 位数值的含义由 **SM2** 决定：

①**SM2**＝0 为双机通信，第 9 位数据用作奇偶位。用以下指令实现发送方将奇偶校验位装入 **TB8**，硬件自动装配数据格式：

```
MOV C，P        ；奇偶校验位送到 C
MOV TB8，C
MOV SBUF，A ；装入发送缓冲器，硬件置位 TI
```

接收方查询到 **RI**＝ 1 时将接收的 8 位数据存入 **SBUF** 并把第 9 位数据装入 **RB8**，对停止位不加理会。接收方可以用发送前的奇偶校验位 **RB8** 与当前接收数据的奇偶校验位 **PSW. 0** 进行比较，当这两位不相同时表明传输出错。

②**SM2**＝ 1 为多机通信，**TB8**＝1 表示发送的是地址帧，**TB8**＝0 表示发送的是数据帧。用指令“SETB TB8”或“CLR TB8” 置 **TB8** 为 1 或 0。

多机通信的操作过程是：主机置 **SM2**＝0、从机都需置 **SM2**＝1，主机先发送从机中靶机的地址帧后再发送数据帧。所有从机将接收到的地址与自己的地址进行比对，若被寻址就先清除 **SM2** 位并准备接收传送来的数据，完成一次通信后需再次置 **SM2**＝1；未被寻址的从机则保留 **SM2** 位并继续进行原先的运行。

● **REN**——允许接收控制位，由软件置位或清除。**REN**＝1 则允许接收，

REN＝0，禁止接收。

● **TB8**——方式 2 和方式 3 中要发送的第 9 位数据。由软件置位或清除。

● **RB8**——方式 2 和方式 3 中已接收的第 9 位数据，可能是奇偶位，或是地址帧/数据帧标识位。在方式 1 中若 **SM2**＝1，**RB8** 是已接收的停止位。在方式 0 中 **RB8** 未用。

● **TI**——发送中断标志。在方式 0 中发送完第 8 位数据时，由硬件置位；在其他方式中，在发送停止位之初由硬件置位。因此，**TI**＝1 表示帧发送结束，其状态可供软件查询、也可请求中断。**CPU** 响应中断后，发送下一帧数据。在任何方式中，都必须由软件清除 **TI**。

● **RI**——接收中断标志。在方式 0 中，接收第 8 位数据结束时由硬件置位；在其他方式中，在接收停止位的半中间时由硬件置位。因此 **RI**＝1 表示帧接收结束，**RI** 可供软件查询或作中断请求信号。但在方式 1 及 **SM2**＝1 时，若未接收到有效的停止位，则不会对 **RI** 置位。在任何模式中，都必须由软件清除 **RI**。

⑵ 电源控制寄存器 PCON 表 3-8

PCON 不可以位寻址，字节地址是 87H。在 HMOS 单片机中，该寄存器除了最高位之外，其余都是虚设的。**SMOD**＝1 时串行口通信波特率加倍。

位　序	D7	D6	D5	D4	D3	D2	D1	D0
位符号	SMOD	—	—	—	GF1	GF0	PD	IDL

⑶ 中断允许寄存器 IE 表 3-9

D7	D6	D5	D4	D3	D2	D1	D0
EA	—	—	ES	ET1	EX1	ET0	EX0
总中断允许位			串行口中断允许	T1 中断允许	INT1 中断允许	T0 中断允许	INT0 中断允许

表中的 **ES** 位为串行口中断控制位，**ES**＝1 且 **EA**＝1 时，允许串行口中断。串行口还涉及在表 3-3 中断优先级寄存器 IP 的定义，请看之前所述。

⑷ 定时器/计数器 1 作波特率发生器（分频器）

串行口工作在方式 1 和方式 3 时，要求定时器/计数器**TL1** 工作于自动重装载方式（计数器/定时器方式 2，作波特率发生器或分频器），f_{osc} 为晶振频率，**TH1** 为定时器 1 的重装载值。定时器 1 中断应禁止。输出波特率计算公式为

$$\text{波特率}=\frac{2^{\text{SMOD}}}{32}\times(\text{定时器/计数器溢出速率})$$

$$=\frac{2^{\text{SMOD}}}{32}\times\frac{f_{\text{OSC}}}{12\times(256-\text{TH1})}$$

3. 串行口四种工作方式及时序

方式 0 用于 I/O 口的扩展, 方式 1, 2, 3 用于通信.

⑴方式 0

方式 0 是将串行口作为同步移位寄存器用于扩展为并口，其波特率固定为 $f_{osc}/12$。以 **RxD** 端输入/输出数据，而 **TxD** 端专用于输出时钟脉冲给外部移位寄存器。发送和接收的数据以 8 位为一帧，不设起始位和停止位。图 3-10 是外接一个 CMOS 的 8 位串入并出移位寄存器 CD4094，**STB**＝1 时打开输出控制门，实现将串口扩展为并行输出口。也可以外接并入串出移位寄存器 CD4014 扩展为并行输入口。

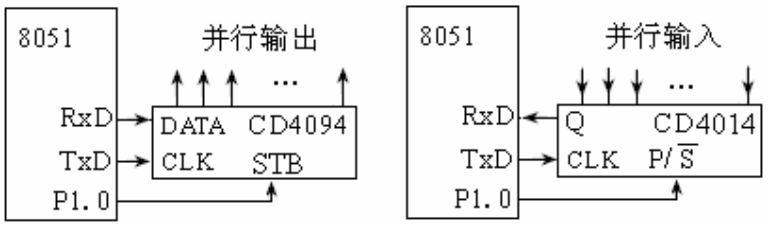


图 3-10　方式 0 将串行口扩展为并行口

⑵方式 1

串行口工作方式 1 为 8 位异步通信口，即一字符帧由 10 位组成：1 位起始位、8 位数据位和 1 位停止位。

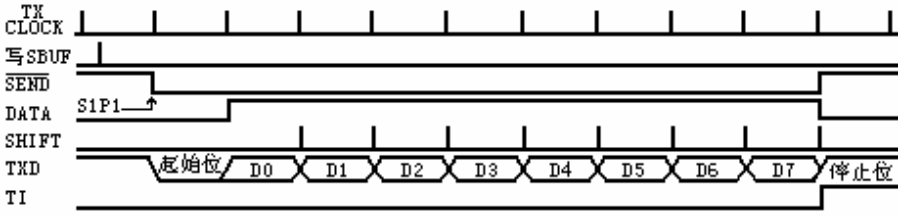


图 3-11 方式 1 发送时序

方式 1 发送如图 3-11 所示，是在发送中断标志 **TI**＝0（发送缓冲器空）时，由一条写 **SBUF** 的指令启动发送控制器的 **SEND** 端，串行口能自动地插入一位起始位 0，在字符结束前插入一位停止位 1，在发送移位脉冲 **SHIFT** 作用下依次由 **TxD** 线上发出。一帧字符发完之后，自动维持 **TxD** 线的信号为 1，在 8 位数据发出之后停止位开始时由硬件使 **TI** 置 1，该信号通知 **CPU** 可以发出下一个字符。

方式 1 发送时的定时信号是发送移位时钟，是由定时器 1 送来的溢出信号经过 16 或 32 分频（取决于 **SMOD** 的值）而取得的。因此，方式 1 的波特率是可变的。

方式 1 接收是在 **SCON** 寄存器中 **REN**＝1（允许接收）时，从检测到 **RxD** 端的负跳变并搜索到有效的起始位开始。在接收移位脉冲的控制下，把收到的

数据逐位移人接收寄存器，直到收齐 8 位数据位和 1 位停止位。

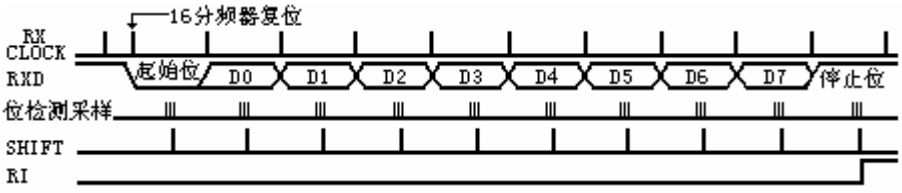


图 3-12 方式 1 接收时序

接收操作有两种定时信号：一种是接收移位脉冲，它的频率和传送波特率相同，也是由定时器 1 的溢出信号经过 16 或 32 分频而得到的。另一种是接收字符的位检测采样脉冲，它的频率是接收移位脉冲的 16 倍。在一位数据期间有 16 个检测脉冲，以其中的第 7、8、9 三个脉冲进行对接收信号的采样，采用三中取二的原则来决定所检测到的值。只要采样信号在接收位的中间位置，就可以避开信号两端的边沿失真，也可以防止由于收发时钟频率不完全一致而带来的接收错误。

在一个字符帧收齐之后，如果以下两个条件同时被满足

①**RI**＝0，即接收的上一帧数据已被 **CPU** 取走，接收缓冲器空；

②**SM2**＝0 或接收到的停止位为 1。

则将接收移位寄存器中的 8 位数据转存入接收 **SBUF**，收到的停止位进入 **RB8**，并使接收标志 **RI** 置 1。上述条件不同时满足，则接收到的数据丢弃，而且不置位 **RI**。

⑶方式 2

方式 2 的一个字符帧由 11 位组成：1 位起始位、8 位数据位、1 位可编程位 **TB8**（第 9 数据位）和 1 位停止位。

方式 2 发送时序如图 3-13 所示。**CPU** 执行一条写 **SBUF** 的指令后，便立即启动发送器发送，送完一帧信息后，**TI** 被置 1，在发送下一帧信息之前，**TI** 必须由中断服务程序（或查询程序）清零。

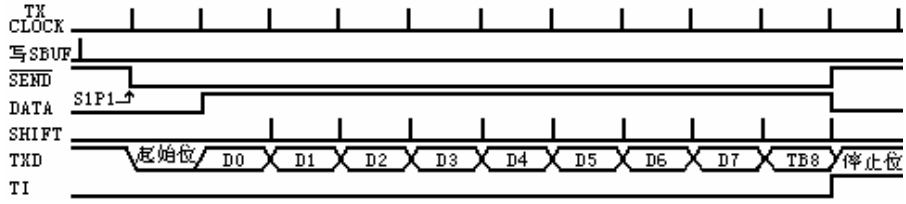


图 3-13 方式 2 发送时序

方式 2 接收与方式 1 基本相似，在模式 1 时把停止位当作第 9 位数据来处理，而在方式 2 中存在着真正的第 9 位数据。方式 2 有效接收数据的二个条件为：

①**RI**＝0，即用户已把 **SBUF** 中上次收到的数据读走；

②**SM2**＝0，为双机通信情况；而 **SM2**=1，为多机通信情况下，收到的第 9 位数据为 1，表示所收到的是地址帧。

若这两个条件成立，接收到的第 9 位数据进入 **RB8**，而前 8 位数据进入 **SBUF** 以准备让 **CPU** 读取，并且置位 **RI**。否则这次接收无效，**RI** 不被置位。图 3-14 表示方式 2 接收时序。

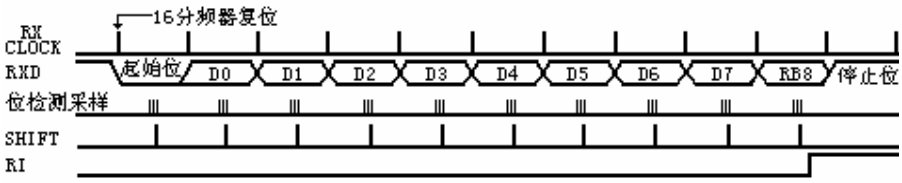


图 3-14 方式 2 接收时序

例 3-3 编写程序用示波器分析串行口工作在方式 1 下输出的信号结构。

分析：设定串行口工作方式 1, 用 1200bps 循环发送一个字节 8AH。用示波器观测 **TxD** 的 TTL 电平波形图和信号的帧结构，标出起始位、数据位和终止位，用示波器测量码元宽度。比较码元宽度的计算值和测量值有无差异。

计算值： 码元宽度＝1/波特率＝1/1200＝833μs

根据波特率的计算公式，设 CPU 的时钟频率 f_{osc}＝11.059 MHz，计算出装载初值 TH1，

因为 $$1200 = \frac{2^0}{32} \times \frac{11.059MHZ}{12 \times (256 - TH1)}$$

所以 TH1=232D=0E8H

参考程序如下：

```
        ORG 0000H
        LJMP   START
        ORG 0050H
START：MOV SP，#50H
        MOV TMOD，#20H   ；T/C1 定时器方式, 模式 2
        MOV SCON，#70H    ；串行口工作模式 1、允许接收
        MOV TH1，#0E8H    ；波特率设置
        MOV TL1，# 0E8H
        SETB TR1          ；启动定时器 1
        MOV A，#8AH
  ST1：MOV SBUF，A        ；将数据送发送缓冲区
        JNB TI，$         ；等待发送结束（查询 TI）
        CLR TI            ；清发送结束标志.
        SJMP ST1
        END
```

例 3-4 编写串行双机通讯程序，发送方每隔 2 秒以中断方式成组发送 40 个字节数据，数据存放在外部数据区 1000H 开始的地方，发送的第一个数据是数据块的长度 40，发送结束后接到来自接收方的反馈信号“0F”表示发送/接收成功。接收方以中断方式成组接收，将收到的数据存放在外部数据区 3000H 开始的地方。

考虑通过 **T0** 定时 50ms 和软件循环 40 次的做法实现总定时 2s，定时器 **T0** 计数初值（之前见 P37 例 3-2 计算过）$X=(15536)_D=(3CB0)_H$

参考程序如下

f43intT.asm　双机通信发送方程序，每隔 2 秒以中断方式发送一次

```
        ORG     0000H
        AJMP START
        ORG     000BH           ；T0 定时器中断服务程序入口地址
        AJMP TF_O
        ORG     0023H           ；串行口中断服务程序入口地址
        AJMP SIO_SUB
        ORG     0050H
；————————主程序　初始化————————
START：MOV   SP，#50H          ；置栈底位置
        MOV   TMOD，#21H        ；T1 模式 2 分频器，T0 定时器
        MOV   SCON，#70H        ；串行口工作方式 1，允许接收
        MOV   TH1，#0E8H        ；T1 分频器的重装载值
        MOV   TL1，#0E8H        ；T1 的初值
        SETB TR1                ；起动 T1
        SETB ES
        SETB EA
        SETB ET0
        MOV   DPTR，#1000H      ；发送的数据放在外部数据区
        MOV   R5，#10H          ；R5、R6 保存地址
        MOV   R6，#00H
        MOV   R3，#40H          ；R3 传送的字节数
        MOV   SBUF，R3          ；发送字符长度
        SJMP $                  ；等待中断

；————————串行口中断服务程序————————
SIO_SUB：JBC   TI，TO_SEND      ；若 TI＝1 就转到 TO_SEND 并清 TI
        JBC   RI，TO_RECE
        RETI
```

```
TO_SEND：
          MOV   A，R3
          JZ   NOTESEND          ；R3＝0 则不再发送
          DEC   R3               ；否则继续发送
          MOVX   A，@DPTR
          INC   DPTR
          MOV   SBUF，A
          RETI
NOTESEND：
          MOV   R4，#28H          ；软件计数器初值 40
          MOV   TH0，#3CH         ；置 T0 定时计数初值
          MOV   TL0，#0B0H
          SETB TR0
          RETI
  TO_RECE：                       ；RI＝1 时接收数据
          MOV   A，SBUF
          CJNE A，#0FH，NOTERECE；反馈信号不是 0FH，应重新发送
          CLR   TR0
          MOV   R5，DPH
          MOV   R6，DPL
          MOV   R3，#40H
          MOV   SBUF，R3
NOTERECE：
          RETI

；————延时 2S 中断服务程序————
  TF_O：MOV   TH0，#3CH          ；置 T0 定时计数初值
          MOV   TL0，#0B0H
          DEC   R4
          CJNE   R4，#00H，NOTE2S ；R4 不为 0 表示不到 2S
          MOV   DPH，R5              ；R4＝0 到 2S 先发送数据长度
          MOV   DPL，R6
          MOV   R3，#40H
          MOV   SBUF，R3
NOTE2S：RETI
          END
```

f43intR.asm　双机通信接收方程序，中断接收

```
        ORG   0000H
        AJMP START
        ORG   000BH
        AJMP TF_O                ；T0 定时器中断服务程序入口地址
        ORG   0023H
        AJMP SIO_SUB             ；串行口中断服务程序入口地址
        ORG   0100H
START：MOV   SP，#50H            ；栈底位置
        MOV TMOD，#21H           ；T1 为 8 位分频器，T0 为 16 位定时器
        MOV   SCON，#70H         ；串行口工作方式 1，允许接收
        MOV   TH1，#0E8H         ；定时器 1 的重装载值
        MOV   TL1，#0E8H
        SETB TR1                 ；起动定时器 1
        SETB ES                  ；置串行中断允许
        SETB EA                  ；置总中断允许
        SETB ET0                 ；置 T0 中断允许
        MOV   DPTR，#3000H       ；接收数据放在外部数据区 3000H
        MOV   R5，#30H           ；R5、R6 保存地址
        MOV   R6，#00H
        MOV   R7，#00H           ；标志 R7=0 表示接收的是数据长度
        MOV   R3，#00H           ；R3 数据长度
        SJMP $
；————串行口中断服务程序————
   SIO_SUB：JBC   TI，TO_SEND
            JBC   RI，TO_RECE
            RETI
  TO_SEND：RETI
  TO_RECE：MOV   A，SBUF          ；接收字符输出到 P1
            MOV   P1，A
            CJNE R7，#00H，DATARECE  ；R7 不为 0 就转去接收数据
            MOV   R3，A          ；R7=0 接收第一个数据是长度放到 R3
            MOV   R7，#01H
            MOV   R4，#14H       ；定时器软件计数初值，延时 1S
            MOV   TH0，#3CH      ；置定时器 T0 初值
            MOV   TL0，#0B0H
            SETB TR0
            RETI
```

```
  DATARECE：MOV   R4，#14H          ；定时器软件计数初值
            MOVX @DPTR，A           ；接收字符保存在外部数据区
            INC   DPTR
            DEC   R3
            MOV   A，R3
            JZ   RECECOMPLETE     ；R3=0 表示接收完毕
            RETI
RECECOMPLET：MOV   A，#0FH          ；接收完反馈 0F
            MOV   SBUF，A
            MOV   R5，DPH
            MOV   R6，DPL
            MOV   R7，#00H
            MOV   R3，#00H
            CLR   TR0
            RETI

；————延时 1S 中断服务程序————
        TF_O：MOV   TH0，#3CH
            MOV   TL0，0B0H
            DEC   R4
            CJNE   R4，#00H，NOTIS
            MOV   DPH，R5
            MOV   DPL，R6
            MOV   R7，#00H
            MOV   R3，#00H
            CLR   TR0
      NOTIS：RETI
            END       (F43intR.ASM)
```

第四章 MCS-51 单片机的系统扩展与接口技术

MCS-51 单片机仅仅是处理数据的主机，数据的输入以及处理结果还需要外部设备提供或消化。连接主机与外设的控制部件是接口，在单片机内部的接口是 **P0**～**P3** 口；在外设一侧的接口有：扩展存贮器的接口、I/O 扩展接口、显示器接口、键盘接口以及模拟接口（A/D 转换器）等。接口内部的寄存器我们称之为端口，为每个端口分配一个地址以便对其访问，接口的内部结构以及外部连线示意图如图 4-1 所示。

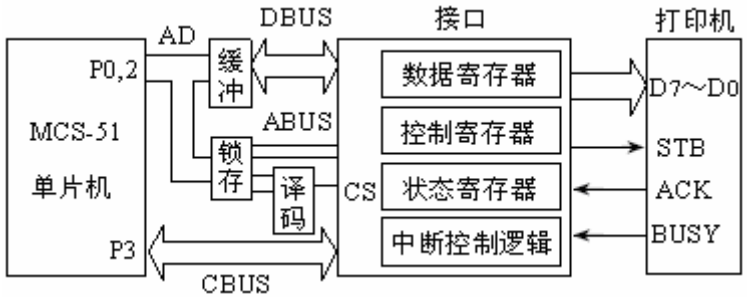


图 4-1　接口的内部结构以及外部连线示意图

4.1　I/O 接口的扩展

1. 接口功能

①数据缓冲——协调主机速度快与外设速度慢在传输速度上的差异。缓冲有二个含义：一是控制与总线隔离或接通，任何时候只允许一个外设与主机通信，以保证逻辑正确和总线负载能力。二是在输出设备暂时不能接收数据时，接口锁存 **CPU** 输出的数据，如打印机接口。

②数据格式转换——并行与串行转换、数字量与模拟量转换、TTL 电平与 RS-232 电平转换等。

③地址译码——通过地址访问接口芯片，芯片地址的高几位由片外译码器选择芯片，芯片地址的低几位由片内译码器选择内部寄存器。

④联络信号——接收来自 **CPU** 的控制字和控制信号、发送到 **CPU** 的设备状态字和状态信号、中断请求与响应信号等等；与外设的联络信号等。

2. I/O 数据传送的控制方式有三种：

①无条件传送——适用于任何时候都处于就绪状态的设备，如电平开关、指示灯等；

②查询（条件传送）——**CPU** 先查询外设的状态，就绪时才传送数据；

③中断方式——外设就绪就发中断，请求 **CPU** 为其传送数据；外设未就绪时，**CPU** 和外设并行地处理各自的事情，提高了 **CPU** 效率。

3. I/O 接口地址

X86 的 I/O 接口与存贮器“分别编址”，用“MOV”指令访问存贮器、用“IN/ OUT”指令访问 I/O 接口。MCS-51 单片机的 I/O 接口地址与外部存贮器地址是“统一编址”，访问 I/O 接口和访问外部存贮器都是使用“MOVX”指令，各接口芯片的片选信号/CS 来自 3～8 译码器 74LS138，如图 4-2 所示。与 AEDK51W 型仿真机配合使用时 EXP51 实验板上 64K 数据空间完全供用户使用，其中部分数据空间已固定分配如下：

1．0000H～7FFFH：32KRAM（62256 芯片）空间地址。
2．8000H、8100H：　扩展地址插孔。
3．8500H：并行扩展芯片 8255 的 PA 口地址；
　8501H：并行扩展芯片 8255 的 PB 口地址；
　8502H：并行扩展芯片 8255 的 PC 口地址；
　8503H：并行扩展芯片 8255 的控制口地址。
4．8600H：A/D 转换芯片 0809 的通道 IN0 地址；
　8602H：A/D 转换芯片 0809 的通道 IN1 地址；
　8604H：A/D 转换芯片 0809 的通道 IN2 地址；
　8606H：A/D 转换芯片 0809 的通道 IN3 地址。
5．8700H：D/A 转换芯片 0832 的片选地址。
6．FF80H：键盘/显示控制器 8279 芯片数据口地址。
　FF82H：键盘/显示控制器 8279 芯片命令/状态口地址。

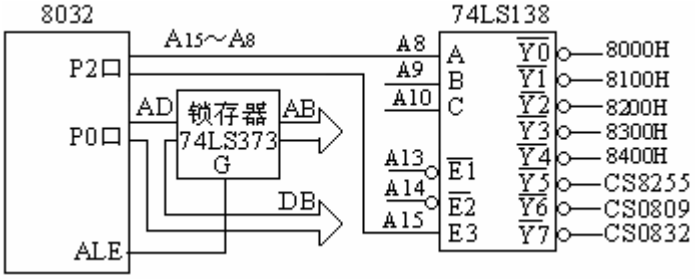


图 4-2　接口的片选信号来自 3～8 译码器 74LS138 的输出
8279 的片选信号由 GAL 产生。

4. I/O 接口　常用的并行接口有：

8 位锁存器 74LS373——为分时复用的 P0 口锁存地址信号；

8 位缓冲器 74LS244——是 8 位的三态门，起到与总线隔离作用；

可编程并行接口 8255——有三个 8 位的并行接口，通过编程选择多种工作方式，内部原理框图如图 4-3 所示。

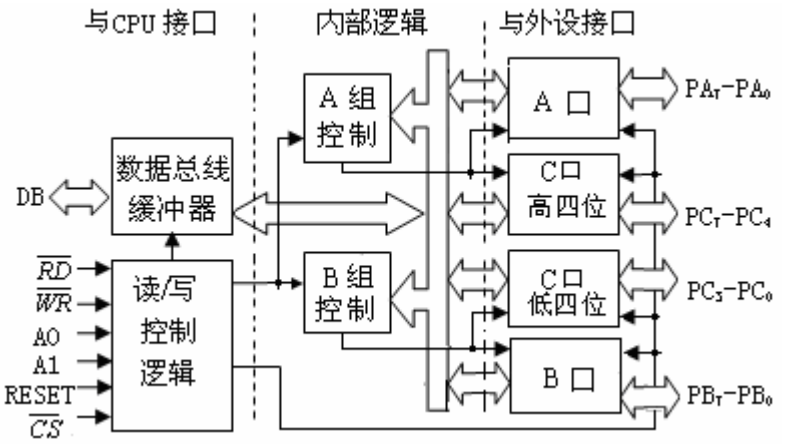


图 4-3　可编程并行接口 8255 内部结构框图

4.2 存贮器扩展

EXP51 实验板上安装的存贮芯片是 64K 的 **EPROM**27512 和 32K 的静态 **RAM**62256。如图 4-4 所示。

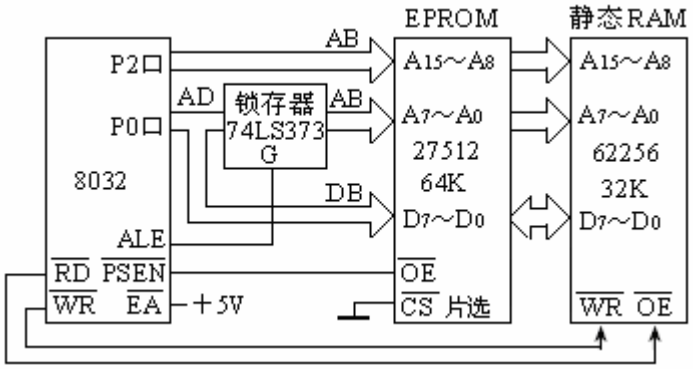


图 4-4 存贮扩展

4.3 模拟通道接口——A/D 转换和 D/A 转换

EXP51 实验板上 A/D 转换器和 D/A 转换器的连线如图 4-5 所示。在第三章图 3-2 讨论了 A/D 转换，下面仅举例 D/A 转换器的应用。

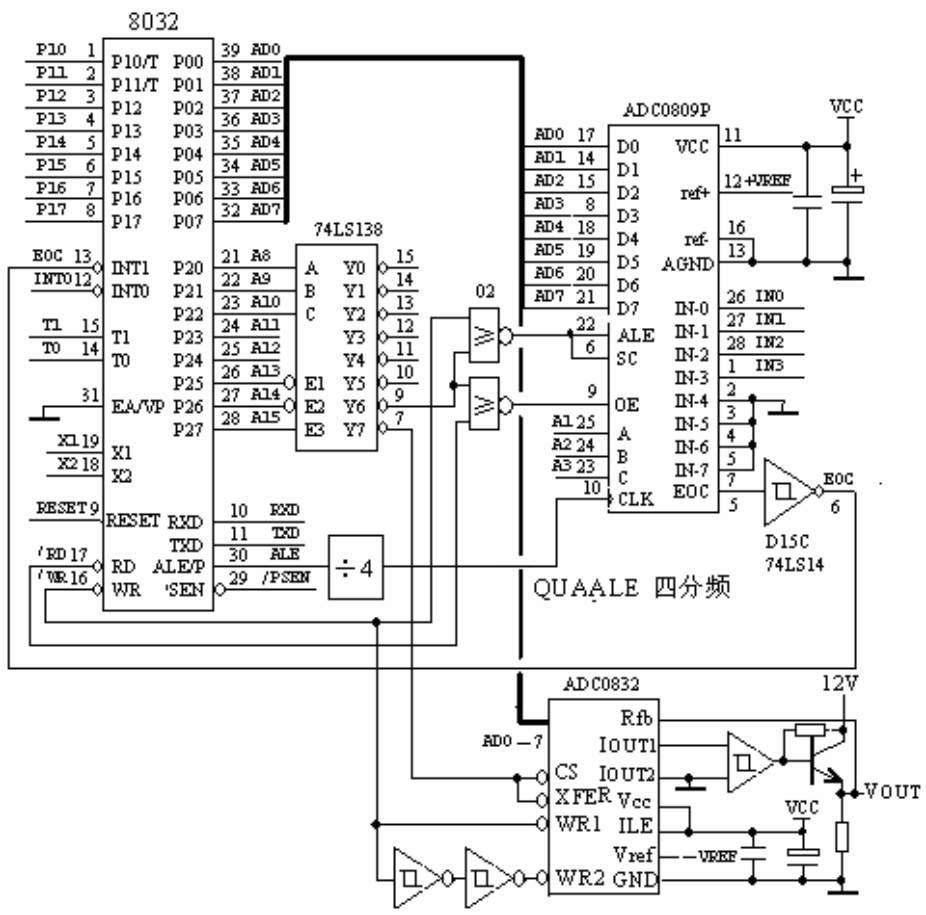


图 4-5　A/D 转换器和 D/A 转换器的连线图

例 4-1 循环地向 DAC0832 的送出数据，使转换后的输出是三角波。参考程序如下：

```
        ORG   0000H
        LJMP   DA0
        ORG   0100H
DA0：MOV   DPTR，#8700H    ；DAC0832 片选地址
        CLR   A
DA1：MOVX   @DPTR，A
        INC   A
        SJMP   DA1
        END
```

4.4 键盘显示接口 8279 的使用

8279 是可编程的键盘、显示接口芯片。它既具有按键处理功能，又具有自动显示功能，在单片机系统中应用很广泛。8279 内部有键盘 FIFO（先进先出堆栈）/传感器，双重功能的 8×8＝64B **RAM**，键盘控制部分可控制 8×8＝64 个按键或 8×8 阵列方式的传感器。该芯片能自动消抖并具有双键锁定保护功能。显示 **RAM** 容量为 16×8，即显示器最大配置可达 16 位 LED 数码显示，EXP－51 实验箱实物照片如图 4-6 所示。在具体介绍 8279 之前，我们先介绍一下键盘与显示接口的工作原理。

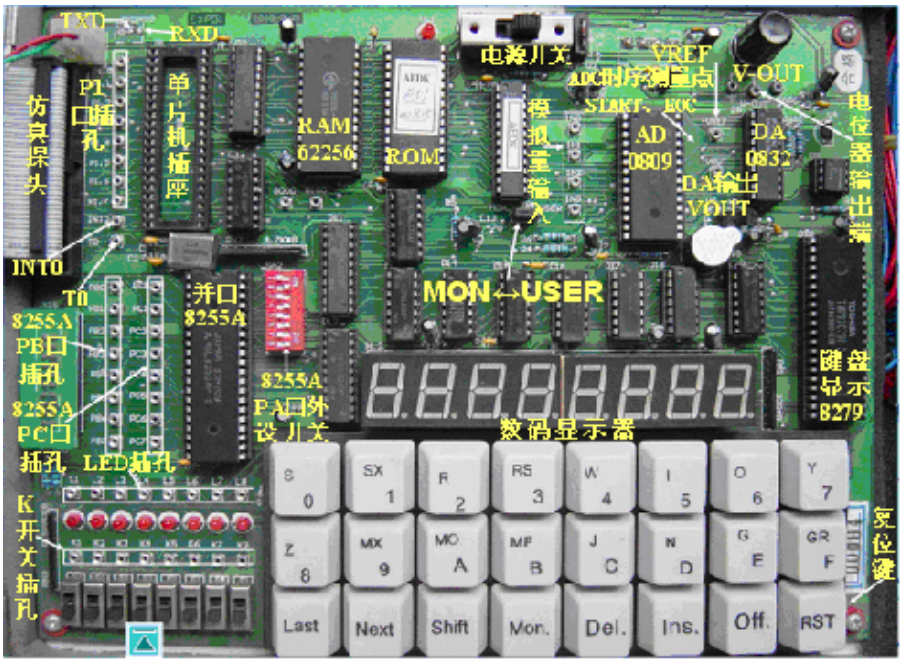


图 4-6　EXP51 实验箱实物照片

4.4.1 键盘与显示接口原理

EXP51 实验箱上 LED 数码管是八个发光二极管以共阳极连接方式通过八反相缓冲器、线驱动器、线接收器 74LS240 与键盘显示接口 8279 连接，数码管和键盘的连线原理图如图 4-7 所示。

1. LED 数码管显示接口原理

从 8279 送到 LED 数码管有两组信号：一组是显示数字的 8 位字符码（低 7 位显示数字，最高位显示小数点 dp）$OUTA_{3\text{-}0}$ 和 $OUTB_{3\text{-}0}$，另一组是扫描信号 **SL1～SL2** 经 3～8 译码器 74LS138 输出各个数码管的位码，在 8 个位码中，任意时刻只有一个有效。用户事先在 8279 内部显示 **RAM** 中存放好了各位数码管的 8 位字符码，在扫描信号 **SL1～SL2** 连续作用下，使 8 个数码管从低位到

高位一位一位地显示相应字符，尽管各位数码管显示实际上不是连续的（只有八分之一周期显示），但由于扫描速度较快，加上人眼睛的视觉暂留现象，感觉显示是连续的。若扫描速度太慢，则显示不连续；若扫描速度太快，则人眼难以分辨显示乱成一片。

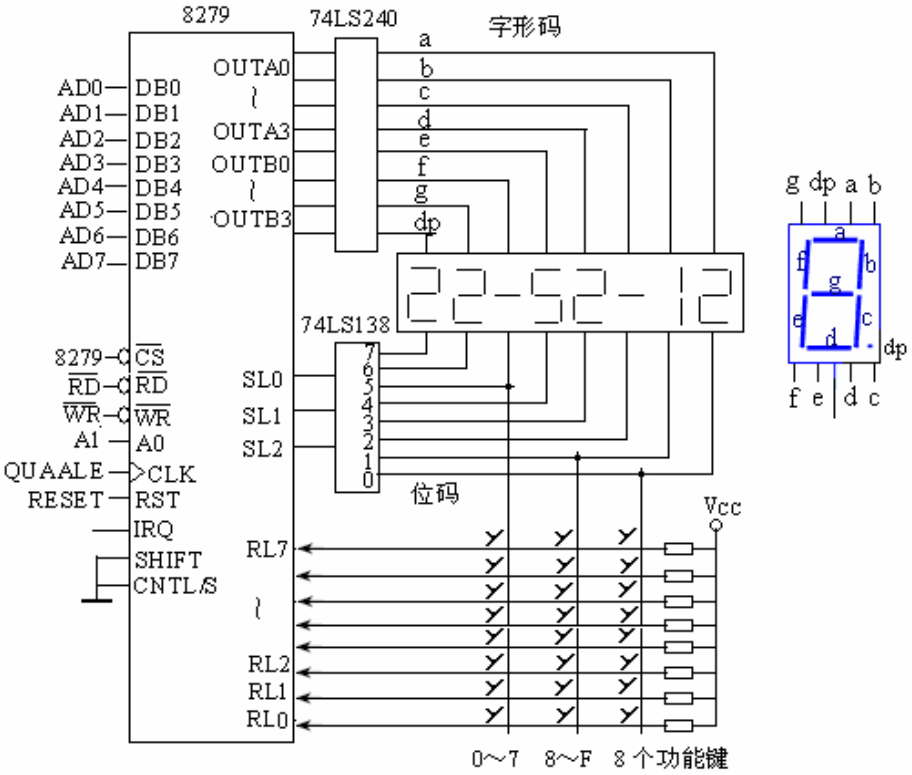


图 4-7　LED 数码管显示接口与键盘接口原理图

扫描信号 **SL1～SL2** 的频率与 8279 的 3# 引脚所接的时钟信号有关，它是单片机时钟频率 11.050MHZ 的 24 分频约 460.8KHZ，这与 8279 要求的工作频率 100KHZ 有一定距离，解决的办法是对 8279 的内部分频器写入分频系数，以获得合适的扫描信号 **SL1～SL2**。

2. 非编码键盘接口原理

单片机所用的键盘有编码键盘和非编码键盘两种。

编码键盘本身除了按键之外，还包括产生键码的硬件电路，只要按下某一按键就能产生这个键的代码，同时还能产生一个脉冲作为中断请求信号通知 **CPU** 读取键码。这种键码的使用比较方便，亦不需要编写很多程序，但使用的硬件较复杂，在微型机系统中使用还不多。

非编码键盘在微型机系统中使用较为普遍，它是由一些按键排列成的一个行列矩阵，按键的作用只是简单的实现接点的接通和断开。非编码键盘几乎不

需要什么硬件电路，但必须有一套相应的软件与之配合才能识别非编码键盘的按键按下、防止抖动以及键码的产生。参见图 4-7，介绍用“行扫描”法识别按键。

① 8279 输出的扫描信号 **SL1～SL2** 经 3～8 译码器 74LS138 产生 8 个输出，同时作为数码管和键盘的扫描码，将 74LS138 输出端 $\overline{m}_5\overline{m}_1\overline{m}_0$ 作为键盘的列线，然后检测键盘的行线，即回复输入线 **RL0～RL7**，通过行扫描并读取回复输入线，可以确定按键的行、列坐标。例如，当列线分别为 110、101、011、行线全都为 1 时表示没有按键按下；当列线为 011 时行线为 11111110 表示 0 行 0 列按键“0”按下，… ，见下表。

$\overline{m}_5\overline{m}_1\overline{m}_0$	110，101 011	011	011	101	011
RL7～RL0	11111111	11111110	10111111	11111011	11111101
按键	无按键	0	6	A	SHIFT

② 一次按下多个键最简单的处理方法是以最后放开的那个键为准。实际上，由于扫描速度很快，遇到两个键同时按下的情况是很少的。

③ 处理按键抖动，按键的簧片在按下时会有轻微的弹跳，需要经过一个短暂的时间才会可靠地接触。若在簧片抖动时进行扫描就可能得出不正确的结果。因此在程序中要考虑防抖动的问题，一般通过调用几次显示子程序达到延迟“行扫描”从而实现消抖目的。8279 芯片能自动消抖。

显示、键盘系统工作流程一般是完成以下功能：

(1)从显示缓冲区中取出数据，送到 8279 数据口供数码管显示；

(2)显示一遍之后，扫描键盘，以确认是否有键按下；

(3)若无键按下，则转到执行显示子程序，若有键按下，则读取键码；

(4)读取键码后分析是什么键，是数字键就按数字键统一处理，是功能键则要执行各自的子程序。

4.4.2 键盘显示接口 8279 的管脚定义

8279 的管脚图如图 4-8 所示，

1．数据线

DB0～DB7 是双向三态数据总线，传送 **CPU** 和 8279 之间的数据和命令。

2．地址线

/CS＝0 选中 8279，当 **A0**＝1 时为命令字及状态字地址；当 **A0**＝0 时为数据地址，故 8279 芯片占用 2 个端口地址。本实验板上管脚 **A0** 接到地址线 **A1**，命令/状态口地址 **FF82H**，数据口地址是 **FF80H**。

3．控制线

CLK：8279 的时钟输入线。

8279 所需频率为 100KHz，在实验板上该引脚接自 QUAALE，即 **ALE** 的四分频或时钟频率 11.059MHz 的 24 分频约 0.5MHz。

IRQ：中断请求输出线，高电平有效。

/RD、/WR：读、写输入控制线。

SL1～SL2：键盘和显示的扫描输出线。

RL0～RL7：键盘或传感器矩阵的回复输入线。

SHIFT：来自外部键盘或传感器矩阵的输入信号，它是 8279 键盘数据的 D6 位的状态，该位状态控制键盘上/下档功能。在传感器方式和选通方式中，该引脚无用。

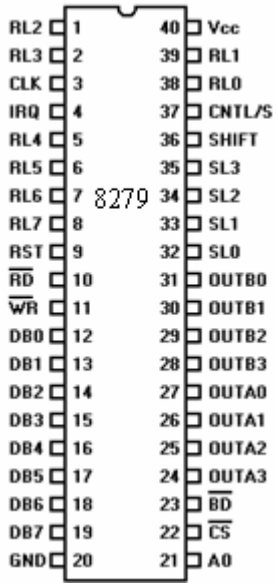


图 4-8　8279 的管脚图

CNTL/S：控制/选通输入线，高电平有效。键盘方式时，键盘数据最高位 D7 的信号输入到该引脚，以扩充键功能；选通方式时，当该引脚信号上升沿到时，把 **RL0～RL7** 的数据存入 **FIFO RAM** 中。

OUTA0～OUTA3：通常作为显示信号（字符码）的高 4 位输出线。

OUTB0～OUTB3：通常作为显示信号（字符码）的低 4 位输出线。

/BD：显示熄灭输出线，低电平有效。当/BD＝0 时将显示全熄灭。

4.4.3　8279 的三种工作方式

1．键盘工作方式　　可设置为双键互锁方式和 N 键循回方式。

双键互锁方式：若有两个或多个键同时按下时，不管按键先后顺序如何，只能识别最后一个被释放的键，并把该键值送入 **FIFO RAM** 中。

N 键循回方式：一次按下多个键均可被识别，按键值按扫描次序被送入 FIFO RAM 中。

2．显示方式

8279 的显示方式又可分为左端入口和右端入口方式。数据写入显示 **RAM** 的顺序，决定了显示的次序。

左端入口方式即显示位置从显示器最左端 1 位（最高位）开始，以后输入的字符逐个向右顺序排列；右端入口方式即显示位置从最右端 1 位（最低位）开始，已显示的字符逐个向左移位。但无论左右入口，后输入的总是显示在最右边。

3．传感器方式

传感器方式是把传感器的开关状态送入传感器 **RAM** 中。当 **CPU** 对传感器

阵列扫描时，一旦发现传感器状态发生变化就发出中断请求（**IRQ** 置 1），中断响应后转入中断处理程序。

4.4.4 8279 的命令及其格式

8279 的各种工作方式是通过写命令寄存器来实现。8279 共有 8 种命令，通过这些命令设置工作寄存器，来选择各种工作方式。命令寄存器共 8 位，高 3 位 D7～D5 决定命令类型，低五位 D4～D0 为命令的具体内容。命令总表见表 4-1。下面详细说明各种命令的设置方法。

1．键盘/显示命令 特征位 D7D6D5＝000

特征位	显示方式		键盘、显示扫描方式		
D7 D6 D5	D4	D3	D2	D1	D0
0 0 0 键盘/显示	0：左端输入 1：右端输入	0：8 字符显示 1：16 字符显示	00 双键锁定 01 N 键轮回 10 扫描传感器矩阵 11 选通输入扫描显示		0：编码扫描键盘 1：译码扫描键盘

8279 的内、外译码由键盘/显示命令字的最低位 D0 选择决定。

D0＝1 选择内部译码，也称为译码方式，**SL0～SL3** 每时刻只能有一位为低电平（4 选 1）。此时 8279 只能接 4 位显示器和 4×8 矩阵式键盘。

D0＝0 选择内部编码，也称为编码方式。所谓编码扫描是指扫描代码经 SL0～SL3 外接 4～16 译码器驱动 16 位显示器，译码器的 16 位输出作为显示扫描输出线（16 选 1），决定第几位显示。显示字段码由 **OUTA0～OUTA3** 和 **OUTB0～OUTB3** 输出。

键盘方式可接 3～8 译码器，构成 8×8 矩阵式键盘。由于键盘最大 8×8＝64 个键，由 **SL0～SL2** 接 3～8 译码器，译码器的 8 位输出作为键盘扫描输出线（列线），**RL0～RL7** 为输入线（行线）。

例 4-2 若希望设置 8279 为键盘译码扫描方式、N 键轮回，显示 8 个字符、右端入口方式，确定其命令字。

分析：据题目要求进行分析，因为具有下列条件：

键盘/显示命令特征位：D7 D6 D5＝000

8 个字符右端入口显示：D4 D3＝10

键盘译码扫描，N 键轮回：D2 D1 D0＝011

所以 8 位命令字为 D7～D0＝00010011B＝13H，送入命令口地址。

例 4-3 若已知命令字为 08H，判断 8279 工作方式。

分析：因为命令字为 08H 即 D7～D0＝00001000B，显然 D7D6D5＝000，该条命令为键盘/显示命令，D4D3＝01 为 16 字符左端入口显示方式，D2D1D0＝000，键盘为编码扫描、双键锁定方式。

2．时钟编程命令　特征位 D7 D6 D5＝001

0 0 1	D4	D3	D2	D1	D0
时钟编程	分频系数取值为：2～31				

D4～D0 用来设置分频系数，分频系数范围在 2～31 之间。

8279 所需工作时钟频率是 100KHz，本实验板上该引脚接自 QUAALE，即 **ALE** 的四分频或时钟频率 11.059MHz 的 24 分频约 0.5MHz。该频率仍然要比 8279 的工作频率高很多，可以通过设置分频系数就可使 8279 得到所需的时钟频率。

例 4-4 若 8279 **CLK** 的输入信号频率为 3.1 MHz，8279 所需工作时钟频率为 100KHz，则分频系数应为 31D＝1FH，于是 D4～D0＝11111，则控制字为：D7～D0＝00111111B＝3FH

3．读 FIFO/传感器 RAM 命令　特征位 D7 D6 D5＝010

0 1 0	D4	D3	D2	D1	D0
读 FIFO/ 传感器 RAM	1：地址+1 0：地址不变	— 不用	确定 FIFO 及传感器 RAM 的首地址 （000～111B 共 8 个单元）		

D4＝1 时 **CPU** 读完一个数据 **RAM** 地址自动加 1，准备读下一个单元数据；
D4＝0 时 **CPU** 读完一个数据，地址不变。

例 4-5 欲编程使单片机连续读 8279 内 FIFO/传感器 **RAM** 中 000～111 单元的数据，设置读命令。

分析：因为要连续读数，地址又连续。所以设置为自动加 1 方式，即 D4＝1，**RAM** 内首地址 000 即 D2～D0＝000，再加上特征位，所以该命令控制字为：D7～D0＝01010000B＝50H， D3 不用设为 0。送入 50H 控制字，在执行读命令时，先从 FIFO/传感器 **RAM** 中 000 单元读数，读完一个数地址自动加 1，又从 001 单元读数，依次类推，直到读完所需数据。

4．读显示 RAM 命令　特征位 D7 D6 D5＝011

0 1 1	D4	D3	D2	D1	D0
读显示器 RAM	1：地址+1 0：地址不变	读显示器 RAM 的地址 （0000～1111B 共 16 个单元）			

D4＝1 **RAM** 地址自动加 1，D4＝0 不加 1。
D3～D0 为显示 **RAM** 中的地址。

例 4-6 欲读显示 **RAM** 中 1000 单元地址，求命令字。

分析：因为只读一个数，地址不需自动加 1，即设置 D4＝0，特征位为 011，地址为 1000，所以其控制命令字为 D7～D0＝01101000B＝68H。

5．写显示 RAM 命令 特征位 D7 D6 D5＝100

1 0 0	D4	D3	D2	D1	D0
写显示器 RAM	1：地址+1 0：地址不变	写显示器 RAM 的地址 （0000～1111B 共 16 个单元）			

D4 是地址自动加 1 控制，D4＝1，地址自动加 1；D4＝0，地址不加 1。D3～D0 是欲写入的 **RAM** 地址，若连续写入则表示 **RAM** 首地址。

6．显示器禁止写入/ 熄灭命令 特征位 D7 D6 D5＝101

1 0 1	D4	D3	D2	D1	D0
显示禁止/熄灭	—	1：禁止写 A 组显示 RAM	1：禁止写 B 组显示 RAM	1：A 组熄灭	1：B 组熄灭

D3：D3＝1 禁止 A 组显示 **RAM** 写入。

D2：D2＝1 禁止 B 组显示 **RAM** 写入。

D1：A 组显示熄灭控制。D1＝1 熄灭；D1＝0 恢复显示。

D0：B 组显示熄灭控制。D0＝1 熄灭；D0＝0 恢复显示。

利用该命令可以控制 A、B 两组显示器，哪组继续显示，哪组被熄灭。

例 4-7 假设 A、B 两组灯均已被点亮，现在希望 A 组灯继续亮，B 组灯熄灭，确定其命令字。

分析：根据命令格式，A 组灯继续亮应禁止 A 组 RAM 再写入其他数据，故 D3＝1；B 组显示熄灭 D0＝1，除特征位外其余位设为“0”。故其控制命令字为 D7～D0＝10101001B＝A9H。

7．清除（显示 RAM 和 FIFO 中的内容）命令　特征位 D7 D6 D5＝110

D7D6D5	D4	D3	D2	D1	D0
110 清除 FIFO/ 显示 RAM	1：允许清除 按 D3D2 方式清除	0×：显示 RAM 全部清 0 10：显示 RAM 置为 20H (A 组 0010B，B 组 0000B) 11：显示 RAM 置为 FFH		1：FIFO 空/ 中断复位/传 感器 RAM 读出地址置 0	1：全部清除 0：不清除
	0：按 D0 方式清除				

D0 为总清除特征位，D0＝1 把显示 RAM 和 FIFO 全部清除。
D1＝1 清除 FIFO 状态，使中断输出线复位，传感器 RAM 的读出地址清 0。
D4～D2：设定清除显示 RAM 的方式。

例 4-8 将全部显示 RAM 清 0，其命令字为：D7～D0＝11010001B＝D1H

8．结束中断/出错方式设置命令　特征位 D7 D6 D5＝111

111	1	D3	D2	D1	D0
结束中断/出 错方式设置	A 在传感器方式，用此命令结束传感器 RAM 的中断请求。 B 在键盘扫描 N 键轮回方式，用此命令设置特定错误方式。				

D4＝1 时（其 D3～D0 位任意）有两种不同作用。

第一：在传感器方式，用此命令结束传感器 RAM 的中断请求。因为在传感器工作方式时，每当传感器状态发生变化，扫描电路自动将传感器状态写入传感器 RAM，同时发出中断申请，即将 IRQ 置高电平，并禁止再写入传感器 RAM。中断响应后，从传感器 RAM 读走数据进行中断处理，但中断标志 IRQ 的撤除分两种情况。若读 RAM 地址自动加 1 标志位为“0”，中断响应后 IRQ 自动变低，撤消中断申请；若读 RAM 地址自动加 1 标志位为“1”，中断响应后 IRQ 不能自动变低，必须通过结束中断命令来撤消中断请求。

第二：在设定为键盘扫描 N 键轮回方式时作为特定错误方式设置命令。在键盘扫描 N 键轮回工作方式，又给 8279 写入结束中断/错误方式命令，则 8279 将以一种特定的错误方式工作，即在 8279 消抖周期内，如果发现多个按键同时按下，则将 FIFO 状态字中错误特征位置“1”，并发出中断请求阻止写入 FIFO RAM。

根据上述 8 种命令可以确定 8279 的工作方式。在 8279 初始化时把各种命令送入命令地址口，根据其特征位可以把命令存入相应的命令寄存器，执行程序时 8279 能自动寻址相应的命令寄存器。8 种命令归纳于表 4-1。

表 4-1　8279　命令功能一澜表

D7D6D5	D4	D3	D2	D1	D0
000 键盘/显示	0： 左端输入 1： 右端输入	0： 8 字符显示 1：16 字符显示	00 双键锁定 01 N 键轮回 10 传感器矩阵 11 选通输入扫描显示		0：内 编码扫描 1：内 译码扫描
001 时钟编程	D4	D3	D2	D1	D0
	分频系数取值为：2～31				
010 读 FIFO/ 传感器 RAM	D4	D3	D2	D1	D0
	1：地址+1 0 地址不变	— （不用）	确定 FIFO 及传感器 RAM 的首址 （000～111B 共 8 个单元）		
011 读显示器 RAM	D4	D3	D2	D1	D0
	1：地址+1 0 地址不变	读显示器 RAM 的地址 (0000～1111B 共 16 个单元)			
100 写显示 RAM	D4	D3	D2	D1	D0
	1：地址+1 0 地址不变	写显示器 RAM 的地址 (0000～1111B 共 16 个单元)			
101 显示禁止/熄灭	—	1：禁止写 A 组显示 RAM	1：禁止写 B 组显示 RAM	1： A 组熄灭	1： B 组熄灭
110 清除 FIFO 或/和 显示 RAM	1：按 D3D2 方式清除 0：按 D0 方式清除	D3 0×：显示 RAM 全部清 0 10：显示 RAM 置为 20H (A 组 0010B，B 组 0000B) 11：显示 RAM 置为 FFH	D2	1：FIFO 空/ 中断复位/ 传感器 RAM 读出地 址置 0	1： 全部清除 0： 不清除
111 结束中断/ 出错方式设置	1	D3	D2	D1	D0
	A 在传感器方式，用此命令结束传感器 RAM 的中断请求。 B 在键盘扫描 N 键轮回方式，用此命令设置特定错误方式。				

4.4.5 8279 的状态字及其格式

状态字显示出 8279 的工作状态。状态字和命令字共用一个地址口。当 A0＝1 时从 8279 命令/状态口地址读出的是状态字。状态字各位意义如下：

D7：D7＝1 表示显示无效，此时不能对显示 RAM 写入。

D6：D6＝1 表示至少有一个键闭合。在特殊方式时有多键同时按下错误。

D5：D5＝1 表示 FIFO RAM 已满，再输入一个字则溢出。

D4：D4＝1 表示 FIFO RAM 中已空，无数据可读。

D3：D3＝1 表示 FIFO RAM 中数据已满。

D2～D0：FIFO RAM 中数据个数。

显然，状态字主要用于键盘和选通工作方式，以指示 FIFO RAM 中的字符数及有无错误发生。

4.4.6　8279 的数据输入/输出格式

对 8279 输入/输出数据不仅要先确定地址口，而且数据存放也要按一定格式，其格式在键盘和传感器方式有所不同。

1．键盘扫描方式数据输入格式

键盘的行号、列号及控制键位置如下：

CNTL	SHIFT	SL2	SL1	SL0	D2～D0 由 RLx 的 x 决定

D7：控制键 “CNTL”状态。

D6：控制键 “SHIFT”状态。

D5 D4 D3：被按键所在行号（由 SL0～SL2）状态确定。

D2 D1 D0：被按键所在列号（由 RL0～RL7）状态确定。

2．传感器方式及选通方式数据输入格式

此种方式 8 位输入数据为 RL0～RL7 的状态。格式如下：

RL7	RL6	RL5	RL4	RL3	RL2	RL1	RL0

图 4-9 和图 4-10 是实验箱实际输入的效果与厂家说明书输入效果的对照图，怀疑实验箱是 3～8 译码器输出端连线有误，用户使用时一定要先试一下，建议采取左端输入或在主程序中加入下面八条指令使显示调整为输入的数字从右向左移动，如

```
MOV  37H，36H
MOV  36H，35H
MOV  35H，34H
MOV  34H，33H
MOV  33H，32H
MOV  32H，31H
MOV  31H，30H
MOV  30H，A
```

写显示 RAM 地址设置	次序	右入口（10） 7 6 5 4 3 2 1 0		左入口（00） 7 6 5 4 3 2 1 0
90	1		1	1
地址加 1	2		1 2	1 2
地址 0 始	3		1 2 3	1 2 3
91	1	1		1
地址加 1	2	2	1	1 2
地址 1 始	3	3	1 2	1 2 3
97	1		1	1
地址加 1	2		1 2	2 1
地址 7 始	3		1 2 3	2 3 1
80	1		1	1
固定地址	2		2	2
0	3		3	3

图 4-9 八位数码管正确的输入方式

写显示 RAM 地址设置	次序	右入口（10） 7 6 5 4 3 2 1 0	左入口（00） 7 6 5 4 3 2 1 0
90	1	1	1
地址加 1	2	2 1	2 1
地址 0 始	3	3 2 1	3 2 1
91	1	1	1
地址加 1	2	1 2	2 1
地址 1 始	3	2 1 3	3 2 1
97	1	1	1
地址加 1	2	2 1	1 2
地址 7 始	3	3 2 1	1 3 2

图 4-10 实验箱连线错误时的实际输入方式

4.5 MCS-51 单片机应用系统的开发及实验设备

开发是指从提出任务到形成产品正式运行的过程。单片机应用系统开发的主要过程有：

① 硬件电路设计、组装和调试；

② 软件编制和调试；

③ 程序固化，单片机脱离仿真机运行。

1. 单片机开发系统的组成

一般的单片机不具有自开发功能，只能借助于单片机开发工具进行软件、硬件调试。单片机开发系统的组成由开发系统硬件和监控程序软件组成，如图 4-11 所示。在调试阶段，先将用户系统的单片机从插座上拆下来，在插座上接上仿真机的电缆，就可以开始对用户系统的硬软件在（与 PC 机及仿真机）连线仿真调试了。调试成功后，将软件写入用户系统的 EPROM 中去并拔下仿真电缆换上单片机，用户系统就能脱离仿真机运行了。

图 4-11　单片机开发系统的组成

2. 单片机开发工具 AEDK

实验室采购的是上海航虹高科技有限公司生产的 AEDK51W 单片机开发系统，其开发工具的主要功能有：

① 电路诊断、检测；

② 程序编辑、汇编与反汇编；

③ 可控制程序（单步、断点、跟踪、全速）运行，运行状态可查询；

④ 不占用用户资源、提供通用子程序库以及程序固化。

调试中，用户程序存入仿真 RAM 或用户 ROM，在用户系统的硬件环境下，运行用户程序。

3. AEDK 使用快速入门

完成一个单片机应用系统的设计，一般要经过软件流程设计，程序编写，程序编译，程序调试，程序修改，程序再编译再调试等反复的过程，直到软件按预期要求符合应用系统的需要。流程图如图 4-12（见“帮助”）。

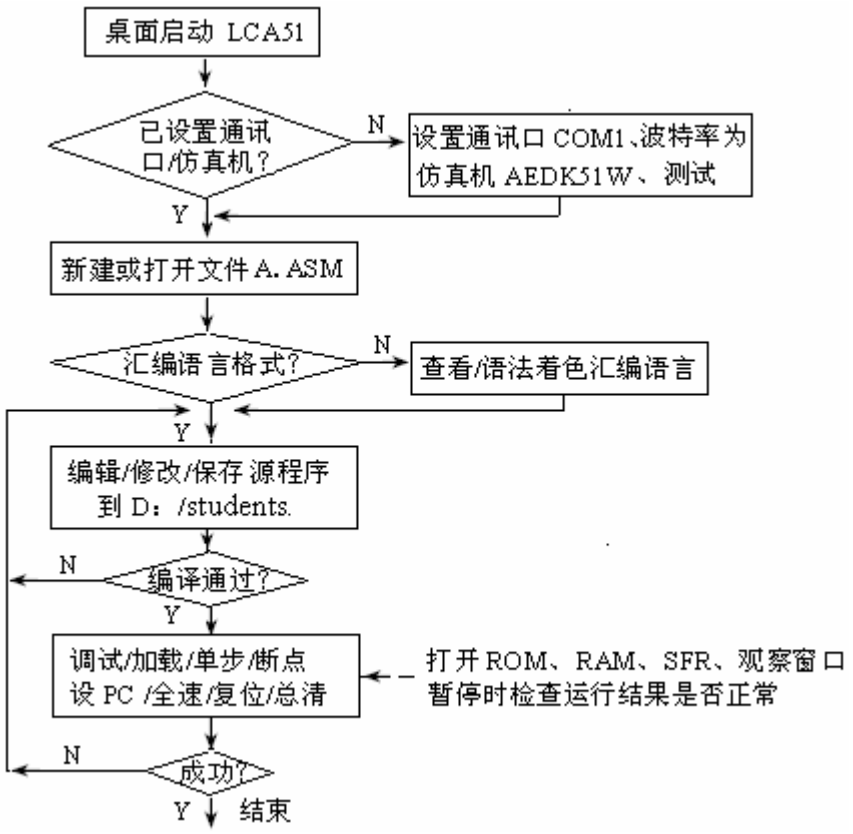


图 4-12　AEDK 使用方法流程图

⑴ 启动 LCA51

用户双击桌面上的 LCA51 软件的图标，即可直接进入本软件。软件的主界面如图 4-13 所示。

第一次启动 LCA51，用户需设置仿真机型号、状态和通讯端口。退出 LCA51 时，会自动保存用户最后一次的设置。用户选择“设置——仿真机”菜单项，在弹出的如图 4-14 对话框中，根据使用的仿真机型号选择相应的类型。在“设置——通讯口”菜单项，选择相应的通讯串口和通讯波特率。最高通讯波特率由仿真机型号决定。点击测试串口按钮，检查通讯口是否可用。

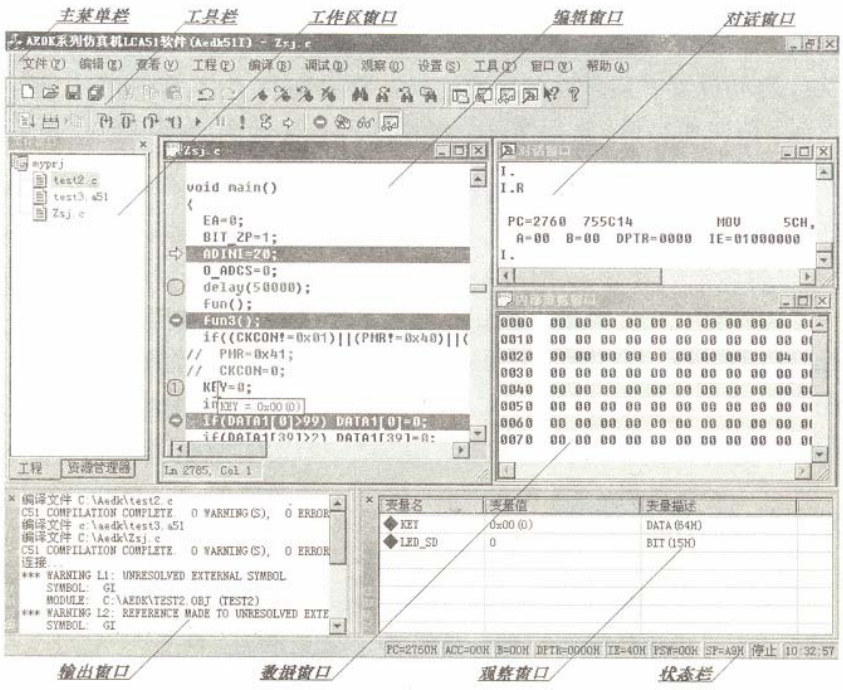

主菜单栏
工具栏
工作区窗口
编辑窗口
对话窗口
输出窗口
数据窗口
观察窗口
状态栏

图 4-13


设置
仿真机
通讯
通讯口
COM1
COM2
COM3
COM4
波特率
2400
4800
9600
19200
57600
115200
延时因子(小->大):
调试串口
确定
取消
帮助

图 4-14

⑵ 编辑程序

如图 4-15，选择“文件——新文件”菜单项，将新建一个空的编辑窗口。用户在编辑窗口中输入程序。输入完毕，选择“文件——另存为”菜单项，把该文件保存为 A1．asm。

编译器不支持长文件名和带有中文的文件名，用户不能把该文件保存在如“我的文档”之类的目录夹中，否则编译器将找不到该文件。

用户可以直接选择“文件——打开”菜单项打开该文件。

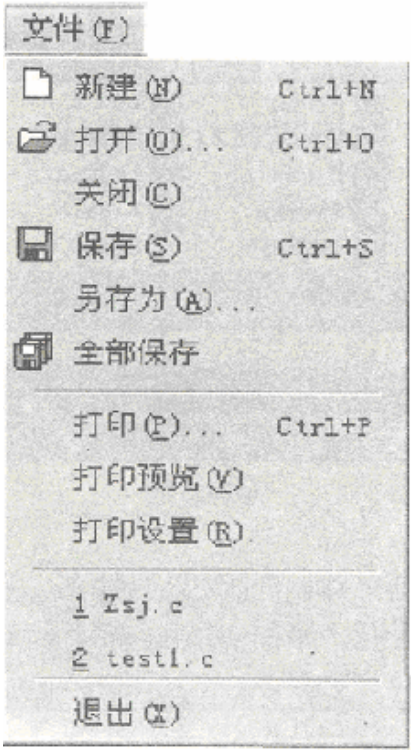


图 4-15

⑶ 程序编译

编译检查源文件的语法错误，如果没有语法错误，编译器将生成源文件的目标代码，对于单汇编文件，编译产生的目标代码可以直接加载调试。对于高级语言的源文件，编译结束还要连接才能产生可加载的目标代码。用户选择“编译——编译当前文件”菜单项，将编译当前活动窗口中的源文件，编译结果的信息显示在输出窗口中。用户可以根据输出窗口中错误信息直接定位到源文件的相应位置进行修改。

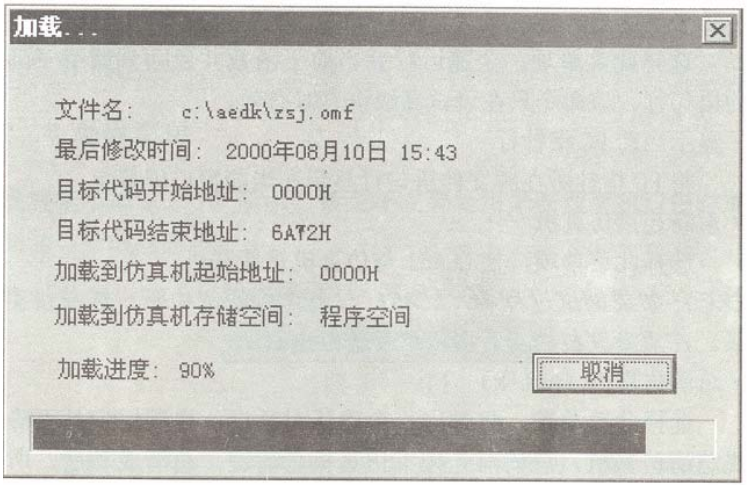


图 4-16

⑷ 程序调试

源程序的语法错误可以通过编译器检查并修改，而逻辑错误必须通过调试，发现程序实际现象与预期设想不同的原因所在。用户可以使用单步，断点，全速，变量察看等调试命令，跟踪程序的执行，直到找到错误原因。

用户选择“调试——加载调试”菜单项，加载当前活动窗口中的源文件所产生的目标文件到仿真机。加载完毕，用户选择“调试——单步、断点、跟踪、全速”等运行方式。

⑸ 修改存贮器的操作

修改存贮器的操作在“数据区窗口” 中进行。打开相关存贮器窗口，利用

鼠标右键菜单设置为“允许直接修改”后，可直接修改存贮器相关区域的数据。

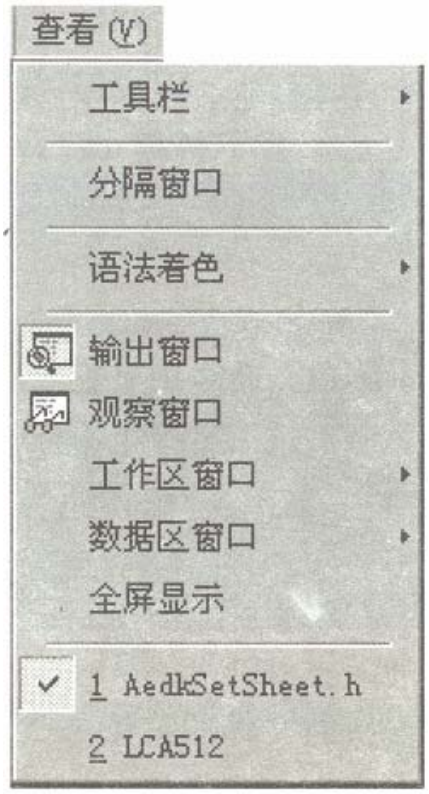
在例2-3中，要求编写用8421BCD码表示的两个四位十进制数相加的程序。

①被加数存放在程序存贮器的8500H和8501H单元；

②加数存放在外部数据存贮器的0000H和0001H单元；

③存放和的内部数据存贮器的20H、21H和22H单元清零。

所有数据都按照高字节放在高地址、低字节放在低地址顺序存放。



(6) 添加观察项

通过“观察窗口”和“数据区窗口”可以检查中间结果和最后结果是否正确。

附录一　习题与思考题

第一章

1-1．MCS-51 单片机由哪几部分组成？有多少 I/O 线？它的地址总线和数据总线各是多少位？

1-2．8051 单片机有多少个特殊功能寄存器？它们可以分为几组、各完成什么主要功能？

1-3.决定程序执行顺序的寄存器是那个？它有几位？它属于特殊功能寄存器吗？

1-4．说明堆栈指针 **SP** 和地址指针 **DPTR** 的用途有什么不同？

1-5．8051 单片机的内部数据存贮器可以分为几部分？各有什么特点？

1-6．什么是时钟周期？机器周期？指令周期？MCS-51 单片机的一个机器周期包括多少时钟周期？

1-7．为什么要了解 **CPU** 的时序？在读外部存贮器时，**P0** 口上一个指令周期中出现的数据序列是什么内容？在读外部数据存贮器时，**P0** 口上出现的数据序列又是什么内容？

1-8．MCS-51 单片机的寻址范围是多少？8051 单片机可以配置的存贮器容量最多是多少？而用户可以使用的程序存贮器和数据存贮器的最大容量又各是多少？

1-9．为什么扩展外部存储器时，**P0** 口要外接锁存器，而 **P2** 口却不接？

1-10．8051 单片机需要扩展 4KB 外部程序存贮器数据，要求地址范围为 1000H～1FFFH，以便和内部程序存贮器地址相衔接。所用芯片除了地址线和数据线外，只有片选控制端/CS。画出扩展系统的连线图。

第二章

2-1．已知 A＝74H，R＝30H，（30H）＝A5H，PSW＝80H，以上面规定的数据参加操作时，将执行以下各条指令后的结果进行填空。

1．XCH　A，R0　　；A＝_____ ，R0＝_____

2．XCH　A，30H　　；A＝_____

3．XCH　A，@R0　　；A＝_____

4．XCHD　A，R0　　；A＝_____

5．SWAP　A　　；A＝_____

6．ADD　A，R0　　；A＝_____，Cy＝_____ ，0V＝_____

7．ADD　A，30H　　；A＝_____，Cy＝_____ ，0V＝_____

8．ADDC A，#30H　　；A＝_____，Cy＝_____ ，0V＝_____

9．SUBB A，30H　　；A＝_____，Cy＝_____ ，0V＝_____

10．SUBB A，#30H　　；A＝_____，Cy＝_____ ，0V＝_____

2-2．思考题：仅交换两个操作数的高 4 位，如何实现？

2-3．已知内部 **RAM** 单元 10H、30H 和 40H 的内容分别是(10H)＝00H，(30H)＝40H，(40H)＝10H，端口 **P1**＝CAH。问执行下面指令后，各寄存器 **R0**、**R1**、端口 **P1**、以及存贮单元 10H、30H 和 40H 的内容分别是什么？

```
MOV   R0，#30H
MOV   A，  @R0
MOV   R1，A
MOV   B，  @R1
MOV   @R1，P1
MOV   P2，P1
MOV   10H，#20H
MOV   30H ，10H
```

2-4．设 A＝83H，R0 ＝17H，(17H) ＝34H，执行下面指令后 A＝_____ ？

```
ANL   A，  #17H
ORL   17H，A
XRL   A，@R0
CPL   A
```

2-5．编写 BCD 减法程序，实现$(91)_{10}-(36)_{10}$，结果存在内部 **RAM** 的 31H 单元。

提示：十进制减法运算可以用加“减数的补码”来进行，两位十进制数是对 100 取补，如 60－30＝30 等同于 60＋（100－30）＝[1]30 ，最高位[1]实际上丢失。因为是 8 位 **CPU**，没有必要同时也无法表示 100，因此可用 8 位二进制数 10011010＝9AH 代替，该数调整后为 100。十进制无符号数减法运算可按以下步骤进行：

⑴ 求减数的补码，9AH－减数；

⑵ 被减数与减数的补码相加；

⑶ 经十进制调整后得到十进制减法运算结果。

编程以及过程如下：

```
CLR   C                          10011010
MOV   A，#9AH                   －00110110
SUBB  A，M2 ；求补               01100100  ＝9AH－36H
ADD   A，M1                     ＋10010001
DA    A       ；调整             11110101  ＝91H＋(－36H)补
MOV   M3，A                     ＋0110
CLR   C                         [1] 01010101  ＝(55)10
```

2-6．编写程序，若累加器内容分别满足以下条件，则程序转至 LABEL 存贮单元。

1．A≥10　　2．A＞10

3．A≤10　　4．A＜10

2-7．已知 **SP**＝25H，**PC**＝2345H，(24H) ＝12H，(25H) ＝34H，(26H) ＝56H，问执行 RET 指令以后，**SP**＝？**PC**＝？

2-8．若 **SP**＝25H，**PC**＝2345H，标号 LABEL 所在的地址为 3456H，问执行长调用指令 LCALL　LABEL 之后，堆栈指针 **SP** 和堆栈内容发生什么变化？**PC**＝？能否将指令 LCALL　LABEL 换成指令 ACALL　LABEL?为什么？

2-9．编程求 20H 和 21H 单元内两数差的绝对值|(20H)－(21H)|存于 A。

2-10．编程查找在内部 **RAM** 的 20H～50H 单元内出现 00H 的次数，并把查找的结果存入 51H 单元。

2-11．编程将内部 **RAM** 的 20H 单元中的 8 位二进制数转换成三位 BCD 码，并存放百位在 RFIRST 中、十位和个位存放在 SECON 中。

2-12．设变量 X 存放在 VAR 单元，函数值 Y 存放在 FUNC 单元。试编程按下式要求给 Y 赋值。

$$Y = \begin{cases} 1 & X > 20 \\ 0 & 20 \geqslant X \geqslant 10 \\ -1 & X < 10 \end{cases}$$

第三章

3-1．用定时器控制，使单片机实验箱 **P1** 口的 8 个发光二极管按照左循环方式每次只有一只发光，发光持续 1 秒钟。

附录二

实验一　单片机在线仿真器的使用

一、实验目的

1. 初步了解MCS-51系列单片机及其存储器构成、对存储器的管理方法；
2. 初步了解MCS-51系列单片机指令系统；
3. 熟悉爱迪克AEDK仿真开发系统及其调试软件LCA51的基本操作方法；
4. 掌握用仿真开发系统调试和运行程序的基本方法。

二、实验器材

微机一台，爱迪克单片机在线仿真开发系统AEDK51W一台，MCS-51实验箱EXP51一套。

三、实验原理及步骤

存储器是单片机系统重要组成部件之一，按照存储内容和存取方式不同，单片机系统的存储器可分为两类。一类是程序存储器，用于存放程序代码和一些常数数据。通常情况，程序代码和常数数据只能被读取，而不能被任意改写，因而程序存储器是只读（**ROM**）的。另一类是数据存储器，用于存放程序运行时的工作变量和数据，如原始数据、运算中间/最终结果、数据暂存/缓冲、标志字节/位等，有时也用于存放待调试的程序。数据存储器中的数据可根据需要写入或读出，因而数据存储器是可读写的（**RAM**）。

用户单片机系统在实验过程中，用户程序编写、修改、调试的过程需反复进行。把需要反复修改的程序直接装入用户自己的程序存储器是不方便和低效的。通常的做法是把需要调试的程序装在单片机仿真开发系统的数据存储器，即所谓仿真**RAM**中，进行程序的编写、修改和调试。

汇编语言源程序的编辑、编译和目标程序的详细运行方法，可参阅《爱迪克单片机开发系统使用说明》和《LCA51（Win9x版本）使用手册》，在LCA51软件操作窗口的“帮助”菜单中也可检索相关内容。下面结合例2—3”参考程序的具体要求掌握爱迪克（AEDK）仿真开发系统及其调试软件LCA51的基本操作方法。主要操作步骤如下：

1. 在整个仿真系统联机上电后运行LCA51软件。

2. 在窗口下编辑、修改汇编语言参考程序F1．ASM，先保存，后编译。编译无误后，执行“加载调试”命令生成扩展名为 ．OBJ的可执行目标程序。

注意：LCA51不支持在WINDOWS下的长文件名和长路径。LCA51编译器提示找不到实际已经保存的源程序，可能与文件名或路径太长有关.

3. 用修改存储器的操作：①被加数存放在程序存储器的8500H 和8501H 单元；②加数存放在外部数据存储器的 0000H 和0001H 单元；③内部数据存

储器的20H、21H 和22H 单元清零.

说明：修改存储器的操作可在“数据区窗口”中进行。打开相关存贮器窗口，利用鼠标右键菜单设置为“允许直接修改”后，即可直接修改该存储器相关区域的数据。其它有关操作方法，可参阅使用手册。

4．按单步、断点、全速运行、设置PC指针等不同方式调试、运行该程序，可用“数据区窗口”、“观察窗口”等方式检查中间结果和最后结果是否正确。记录、分析实验结果。设置断点要领提示：

⑴设置断点可用移动光标的方式。

⑵事先分析程序与数据运算过程的对应关系，以确定在合适的指令处设置断点。

四、实验内容

1．调试、运行P20例2—3参考程序，学习爱迪克（AEDK）仿真开发系统及其调试软件LCA51的基本操作方法和用仿真开发系统调试、运行程序的基本方法。

例2-3 编写用8421BCD码表示的两个四位十进制数相加的程序。被加数放在程序存贮器以8500H单元开始的地方、加数放在外部RAM以0000H单元开始的地方、和存放在内部RAM地址为20H、21H、22H的三个字节单元内。

2．自编一个程序，实现 N（2＜ N ≤10）个四位十进制数相加的算术运算。N 个四位十进制数存放在程序存储器，相加数的个数N存放在外部数据存储器，和数存放在内部存储器。

五、思考题

1．例 2—3 参考程序中” SJMP $” 是必须的吗？若没有” SJMP $”, 程序运行后能否得到正确的结果?

2. 为什么仿真器的电源开关断开后再次连通并运行程序时, 必须重新加载程序?

3．在参考程序中, 将被加数存放在程序存贮器 100H 单元可以吗?为什么?

4．求两个十进制数之和时往往用十六进制数表示 BCD 码，先相加再进行 BCD 调整。若软件 LCA51 不能对“01H＋99H＝0A0H”和的高位调整，你怎么处理？

实验二　模数和数模转换接口

一、实验目的

1. 了解MCS-51系列单片机的中断系统；
2. 了解模数和数模转换电路芯片的性能及相应程序设计方法；
3. 掌握使用示波器、信号源对实验电路进行调试的方法。

二、实验器材

微机一台，爱迪克单片机在线仿真开发系统AEDK51W一台，MCS-51实验箱EXP51一套。示波器、信号源、万用表各一台.

三、实验原理

数字系统是一个数据处理部件，被处理的数据来自于数字系统外部，而数据处理的结果将送到数字系统外部去控制执行机构。

1、DAC 的工作原理和 DAC0832 芯片

DAC 将二进制数每一位的代码按“权”的大小转换成相应的模拟量，然后把各位的模拟量相加，所得之和就是与数字量正比的模拟量。在图实 2-1 所示的 4 位权电阻 DAC 中，每位权电阻支路有一个双向电子开关S，它受这一位的数字量b_i 控制，当$b_i=1$时S_i接通基准电压V_{REF}，支路上就有电流。数据位的“权”越大，它控制的这位权电阻支路的电流就越大。汇集到Σ的总电流是

$$\begin{aligned} I_{\Sigma} &= I_3 + I_2 + I_1 + I_0 \\ &= \frac{V_{REF}}{2^0 R} b_3 + \frac{V_{REF}}{2^1 R} b_2 + \frac{V_{REF}}{2^2 R} b_1 + \frac{V_{REF}}{2^3 R} b_0 \\ &= \frac{V_{REF}}{2^3 R} (2^3 b_3 + 2^2 b_2 + 2^1 b_1 + 2^0 b_0) \end{aligned}$$

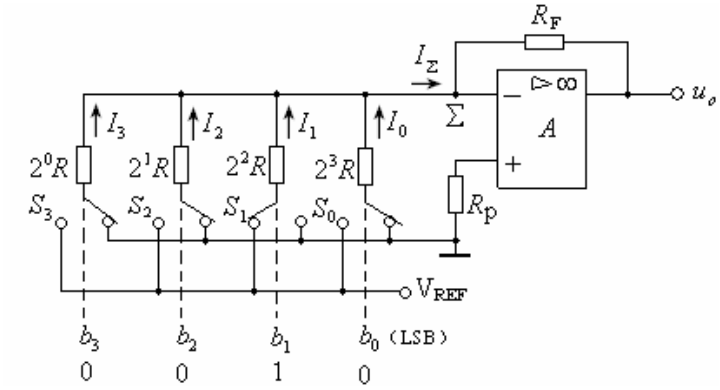
由此类推，n 位权电阻转换器总电流为

$$I_{\Sigma} = \frac{V_{REF}}{2^{n-1} R} (2^{n-1} b_{n-1} + 2^{n-2} b_{n-2} + \ldots + 2^1 b_1 + 2^0 b_0)$$

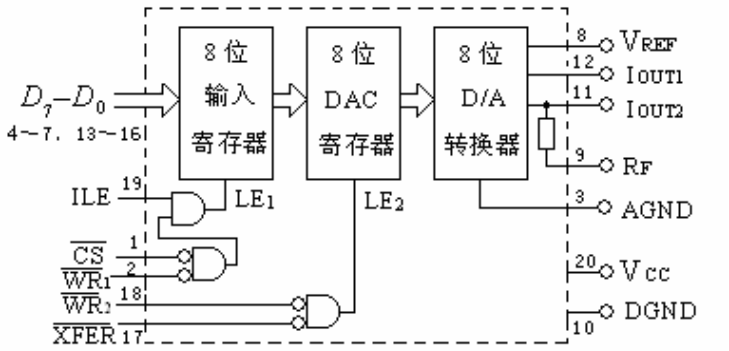
选取 $R_f = \frac{1}{2} R$ 时，运算放大器的输出电压为

$$u_o = -I_{\Sigma} R_f = \frac{V_{REF}}{2^n} (2^{n-1} b_{n-1} + 2^{n-2} b_{n-2} + \ldots + 2^0 b_0) = -\frac{V_{REF}}{2^n} N$$

上式中，$N = 2^{n-1}b_{n-1} + 2^{n-2}b_{n-2} + ... + 2^1 b_1 + 2^0 b_0$ 为数字量按权展开式。



(a) 4 位权电阻 DAC



(b) 内部结构

图实 2-1 4 位权电阻 DAC 和 DAC0832 芯片内部结构

集成 DAC0832 芯片将电阻解码网络和二进制数码控制的开关集成在一块芯片上，从 DAC0832 内部逻辑框图看，它采用二级缓冲输入方式，第一级是输入寄存器、第二级是 DAC 寄存器，二级缓冲可以在转换的同时采集下一个数字量。引脚功能：

ILE：允许输入锁存

$\overline{CS}$：片选信号

$\overline{WR_1}$：写信号 1，在 $\overline{CS}$ 和 ILE 有效时将数据锁存于输入寄存器

$\overline{WR_2}$：写信号 2，在 $\overline{XFER}$ 有效时将输入寄存器的数据传送到 DAC 寄存器

$\overline{XFER}$：传送控制信号

I_{OUT1}：DAC 电流输出 1，是逻辑电平 1 的各位输出电流之和

I_{OUT2}：DAC 电流输出 2，是逻辑电平 0 的各位输出电流之和

R_F：反馈电阻，约 15KΩ

VREF：基准电压，可在±10V 范围内选择

VCC：电源＋5V～＋15V

DAC0832 将输入的数字信号转换为模拟电流的形式输出，为了得到模拟电压，还必须外接运算放大器，具体连线请看 P56 图 4-5。

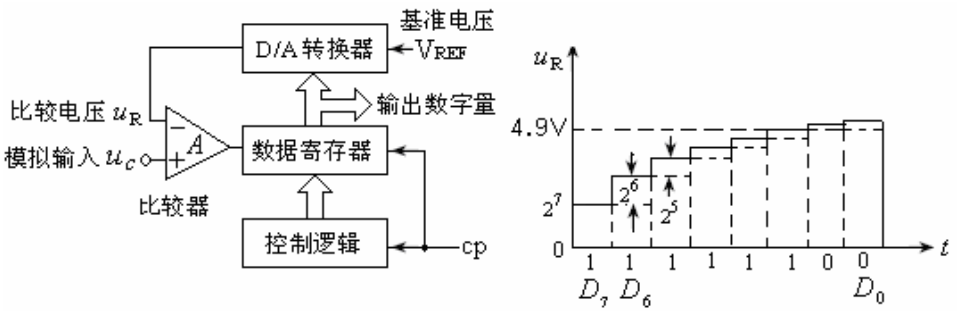
2. ADC 原理和 ADC 0809 芯片

下面以图实 2-2(a) 为例介绍逐渐逼近型模数转换的电路原理，它的操作时序图如图实 2-2 (b)所示。

第一个脉冲到来时，将数据寄存器最高位置“1”，其余位置“0”，该数据送到 DA 转换器可以获得一个比较电压u_{R1}，如果比较器的输出为高电平，则数据寄存器最高位置“1”不变，否则清零；

第二个脉冲到来时，将数据寄存器次高位置“1”，低几位置“0”，该数据送到 DA 转换器可以获得一个新的比较电压u_{R2}，如果比较器的输出为高电平，则数据寄存器的次高位置“1”不变，否则清零；

第三个脉冲到来时，…。以此方法得到 n 位数字量。



(a) 逐渐逼近型电路原理图　　(b) 时序图

图实 2-2

图实 2-3 是 ADC 0809 芯片内部结构的逻辑框图，引脚功能：

IN0～IN7：8 路模拟量输入

ADDA、ADDB、ADDC：三位地址信号选择 8 路输入

ALE：地址锁存信号

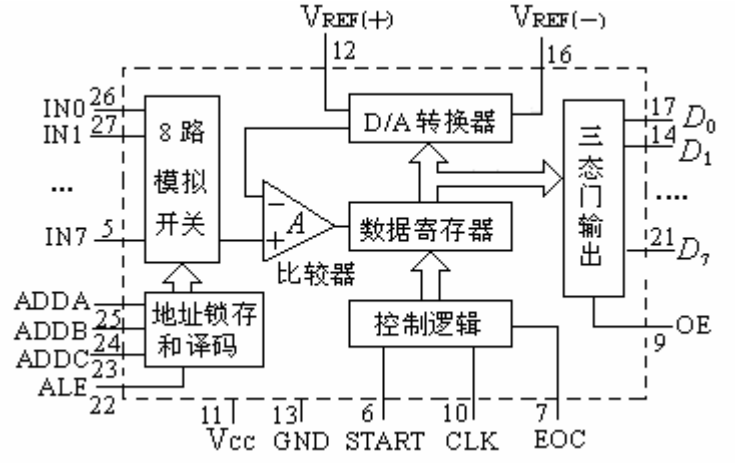
D_0～D_7：8 位数字量输出

START：AD 转换的启动信号

EOC：转换结束信号可用作中断请求信号

OE：输出允许，OE=1 时打开三态门输出数字量

VREF（-），VREF（+）：芯片内部 D/A 转换器的基准电压

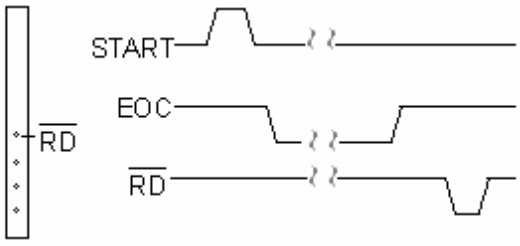


图实 2-3　ADC 0809 芯片内部结构图

四、实验内容

1. 模数转换电路验证, 按照P35例3-1的图示连线，运行其程序。用示波器观察SC、EOC、/RD信号时序, 测量ADC0809模数转换时间，记录用万用表检测输入的模拟电压与转换后的二进制数于下表中。

模拟电压	D7 ～ D0



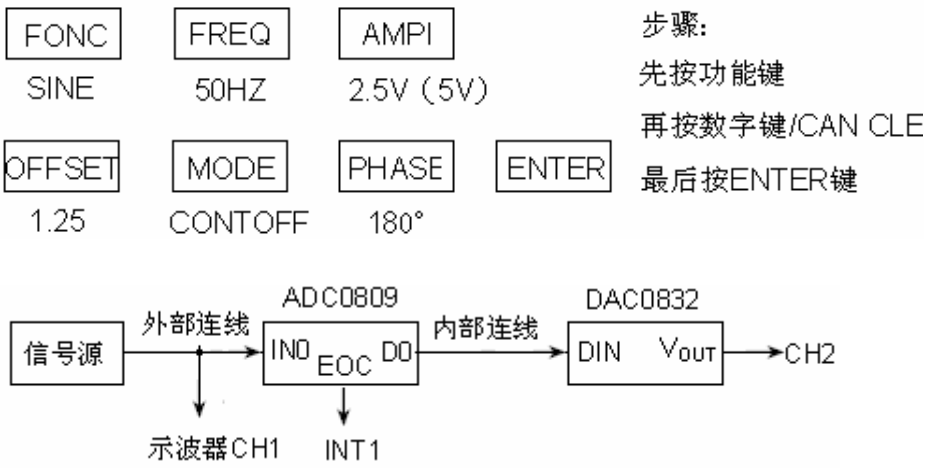
图实 2-4　AD 转换记录及时序图

2．数模转换电路验证, 按照P55例4-1的图示连线, 运行其程序. 用示波器观察并记录转换后的输出波形, 计算其周期和频率. 比较ADC0809 和DAC0832的转换速度。

3. 按图实2-5所示将信号源、示波器和实验箱连线，进行外部数据存贮器、模数转换、数模转换联合调试程序。

编写一个程序将ADC0809的通道0输入的模拟电压转换为一系列的数字量依次存入外部RAM62256 的2048 个存贮单元中，同时将每次转换的数字量立即送到DAC0832作数模转换。调解信号源的频率在100Hz～1000Hz之间变化时，用示波器观察ADC0809的输入端和DAC0832的输出端波形差异，分析造成波形差异的原因。

信号源的设置如下：



图实 2-5　AD 与 DA 的联合调试连线图

4.（选作）试编写一个程序对输入的三角波信号进行简单的数字滤波，使输出的波形比输入波形平滑一些。数字滤波是将连续 4 个或 8 个采样点的输入（4 个点所取的值先除以 4 再相加，即平均值）取平均值后输出。

实验三　键盘显示接口

一、实验目的

1. 了解并掌握MCS-51系列单片机定时器/计数器的工作原理和编程方法。
2. 了解键盘扫描及编码的工作原理；
3. 掌握键盘显示接口芯片8279与单片机的硬件连接及相应编程方法。

二、实验器材

微机一台，爱迪克单片机在线仿真开发系统AEDK51W一台，MCS-51实验箱EXP51一套，示波器。

三、实验原理　如前所述。

四、实验内容

1. 运行 P37 例 3-2 程序，使 MCS-51 单片机的内部定时器工作，通过 **P1** 口控制一个发光二极管每隔一秒钟亮灭一次。

2. 参考下面秒表的初始化程序和主程序以及在 P23 例 2-4 的子程序，利

用 **T0** 或 **T1** 定时器设计功能较完整的电子表（显示格式为 HH-MM-SS）或秒表（显示格式为 XX-XX，精确到 1％秒 ）。功能较完整即具有暂停、继续、清零、置初值等功能。

```
TSH       EQU    04CH      ;计数器初值
TSL       EQU    00H
Z8279     EQU    0FF82H    ;8279 状态/命令口地址
D8279     EQU    0FF80H    ;8279 数据口地址
LEDMOD    EQU    00H       ;左端输入八位字符显示,双键互琐
LEDFEQ    EQU    23H       ;分频系数 20～31
LEDCLS    EQU    0D1H      ;清除显示
LEDWRO    EQU    90H       ;设定写显示 RAM 地址,自动增量
DISPPS    EQU    20H       ;SS-XX 时间保存在内部数据区 21H-20H 中为压缩 BCD
;————————————主程序————————————————
          ORG  0000H
          AJMP   MAIN
          ORG   001BH       ;中断向量
          LJMP   CLKMS
          ORG    0100H
MAIN：    MOV    R2,#00H          ;保存 0.01S 时间的寄存器
          MOV    R3,#00H          ;保存秒时间的寄存器
          MOV    R7,#00H          ;R7 标志，R7=0FFH 等待溢出
          MOV    SP,#60H
          LCALL INIT8279
          MOV    TMOD,#10H        ;设置定时 T0 方式 1
          MOV    TL0,#TSL         ;置 T0 的初值，10mS 中断一次
          MOV    TH0,#TSH
          SETB   EA
          SETB   ET0              ;T0 开中断
          SETB   TR0              ;启动 T0
WAIT：    CJNE   R7,#0H,$         ;R7 不为 0 就原地踏步
          INC    R2               ;R2 作为 0.01s 的计数器
          CJNE   R2,#64H,NEXT     ;软件循环 100 次到 1S
          MOV    R2,#0H           ;到 1S 后 R2 清 0，秒 R3+1
          MOV    A,#01H
          ADD    A,R3             ;ADD 修改标志位 AF 与 C
          DA     A
          MOV    R3,A
```

```
            CJNE    A,#60H,NEXT
            MOV     R3,#0H              ;到 60S，R3 请 0，分 R4+1
 NEXT：MOV      DISPPS,R2
            MOV     DISPPS+1H,R3
            LCALL DISPF                 ;调显示格式整理为分离的 BCD
SCKEY：MOV      DPTR,#Z8279     ;读键盘
            MOVX    A,@DPTR
            ANL     A,#0FH
            JZ      CNTCLK              ;A=0 无键按下，跳转
            MOV     DPTR,#D8279
            MOVX    A,@DPTR             ;读键值
            CJNE    A,#0H,cntclk        ;是 0 键就停止执行
            CLR     EA                  ;关中断
CNTCLK：MOV    R7,#0FFH          ;置 R7=0FFH 等待溢出中断将 R7=0
            AJMP    WAIT                ;主程序结束
 ；————————时钟中断服务程序————————————
 CLKMS:MOV      TL0,#TSL          ;计数初值
            MOV     TH0,#TSH
            SETB    TR0                 ;启动计数
            MOV     R7,#0H              ;置标志 R7=0
            RETI
；————————————8279 初始化————————————————
INIT8279：MOV      DPTR,#Z8279
              MOV       A,#LEDCLS
              MOVX      @DPTR,A
INIT82791：MOVX    A,@DPTR
              JB        ACC.7,INIT82791
              MOV       A,#LEDMOD
              MOVX      @DPTR,A
              MOV       A,#LEDFEQ
              MOVX      @DPTR,A
              MOV       A,#LEDWRO
              MOVX      @DPTR,A
              RET
```

实验四　基于单片机的串行通信

一、实验目的

了解并掌握单片机串行口的应用及相应编程方法。

二、实验器材

微机一台，爱迪克单片机在线仿真开发系统AEDK51W一台，MCS-51实验箱EXP51一套，示波器。

三、实验原理　如前所述。

四、实验内容

1. 参考 P47 例 3-3 程序。用示波器观察串行口工作在方式 1 下信号的帧结构，标出起始位、数据位、停止位，用示波器测量码元宽度。

2, 编写自发自收的自检程序确认串行口工作正常。接收到的字符送到 **P1** 口 LED 发光二极管。

3, 编写实现双机之间的串行口异步通信。下面的具体要求可以逐步实现：

① 中断方式发送和接收；

② 成组发送和接收大于 64K 字节长的数据，数据格式为“字节长度+成组数据”。发送方发送完一组数据后应等待对方反馈信息。若对方确认接收无误，就发送下一组数据。若等待 2 秒钟未收到反馈信息，则重发。接收端将接收到的数据保存在外部数据存贮器并反馈信息,注意不要与发送的数据区重叠。

参考程序如 P49 例 3-4，双机连线为：