第1章 单片机应用最小系统

单片机作为一个微控制器从应用角度来说已经是相当普遍了。那么，该如何实现对单片机工作原理从不了解到熟悉；从不会使用到得心应手的应用单片机；进一步实现用单片机构架各种大小应用系统，或者为以后嵌入式系统的学习与应用奠定良好的基础。紧扣单片机应用特征，理解单片机概念，熟悉单片机的基本组成，掌握单片机最小系统的硬件组成，是学习单片机的良好起步，单片机应用最小系统构成如图1-1所示。

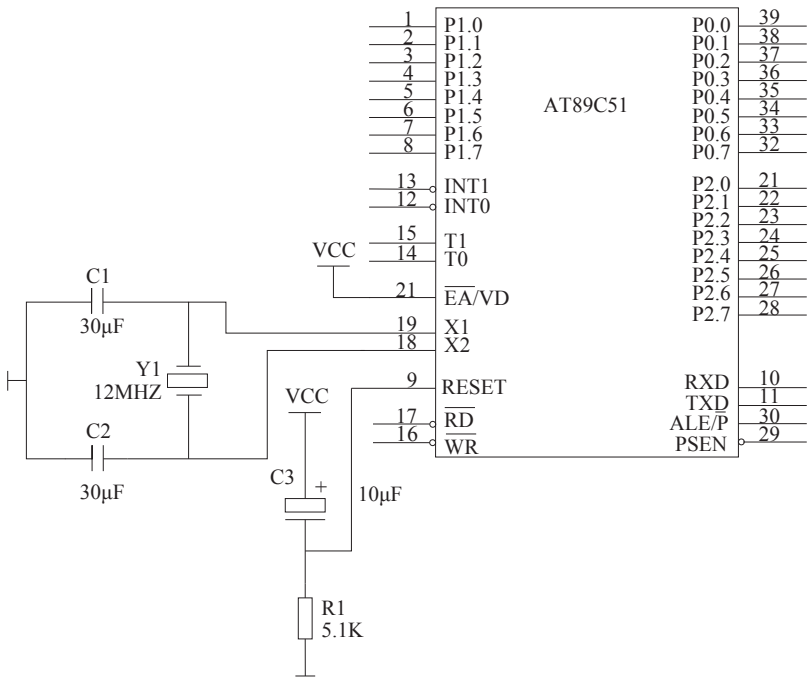


图 1-1 单片机应用最小系统电路图

1.1 单片机概述

学习目标

1. 了解单片机的概念与单片机系列。
2. 了解单片机的应用对象 。
3. 熟悉单片机的基本组成 。

导　入

单片机在家用电器中的应用是十分普通的，例如：洗衣机是不可缺少的家用电器之一，以前洗衣过程需要人的干预，现在基本上都是全自动的洗衣过程。显而易见洗衣机内必定有一个代替人工操作的控制模块，而控制模块的核心就是微控制器。那么，什么是微控制器呢?

1.1.1 单片机的概念

所谓单片机指的就是在一块硅片（单片）上集成了中央处理单元（简称CPU）、随机存储器（RAM）、只读存储器（ROM）、定时器/计数器、串行通信接口与输入/输出接口。早期对于单片机的概述是单片微型计算机（Single Chip Microcomputer），其直接解释为单片机，且沿用至今。单片机的外型封装如图1-2。

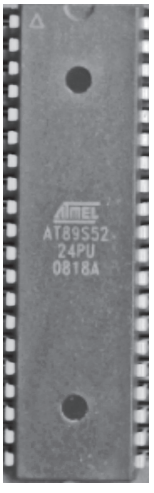


图 1-2 单片机封装

单片机从20世纪70年代早期问世以来，经历了三个大阶段的发展。单片机以体积小，开发比较容易，环境适应能力比较强，可靠性强，低电压/低功耗，高性价比等特点而得到广泛的应用。

单片机的运用遍及家用电器、智能化仪表、工业过程控制、机电一体化等多个领域，归纳其运用本质主要是在测控系统中的应用。可以这样认为，单片机技术就是伴随着控制领域的发展而诞生和不断成长的。随着单片机技术的发展，虽然单片机保持了单片集成的外观形态，但是内部资源在原有的基础上集成了ADC、DAC、PWM等模块，使得其应用在对象之中的特性更加突出，所以单片机被定义为微控制器（Micro Controller Unit）。单片机由于具备了高速性、专用性、高可靠性和抗干扰性，更多的是与应用对象绑定在一起的，用一个形象化的比喻是嵌入在应用系统的对象中的。例如：公交车辆中站名显示牌、语音报站等就是嵌入了单片机的专用控制系统，所以也把单片机归类于“嵌入式微控制器”的大范畴中。从

嵌入式系统角度来认识，单片机应用系统仅仅属于嵌入式的低层次应用系统。

随着超大规模集成电路的快速发展，微计算机还形成了另外一个分支，就是微处理器（Micro Processor Unit），微处理器是微型计算机的核心部件，微处理器的性能决定了微型计算机的主要性能。

当前单片机世界真可谓是眼花缭乱，世界上几大著名的8位单片机制造商，例如：Intel（英特尔公司）MCS-51系列；Atmel公司的兼容MCS-51系列；Microchip公司的PIC165X系列；Motorola公司的6805系列；Philips（飞利浦公司）89C66X系列等。

不同型号的单片机既有功能相同部分，也具备各自的特点。但如何掌握单片机的基本工作原理，驾驭单片机的各种应用，本书选择通用的MCS-51系列作为学习对象，按单片机内部数据传输通道宽度进行划分，有8位、16位、32位，早期还有4位的。我们选择通用8位单片机的工作原理及其应用进行阐述。

MCS-51系列8位单片机主要包括三个基本型：8031、8051和8071，例如：目前用的比较多的AT89C51/AT89C52就是基于8051（内核）系列。AT89C51型号的意义是指由Atmel公司以CHMOS工艺制造的与基本型8051指令系统兼容的8位数据宽度的单片机，其中9表示Flash存储器（0表示Rom存储器，7表示Eprom存储器）；1表示内部程序存储器容量为4K字节（2表示8K字节）；C表示CHMOS工艺，若为S表示采用的是ISP编程方式。

1.1.2 单片机基本组成

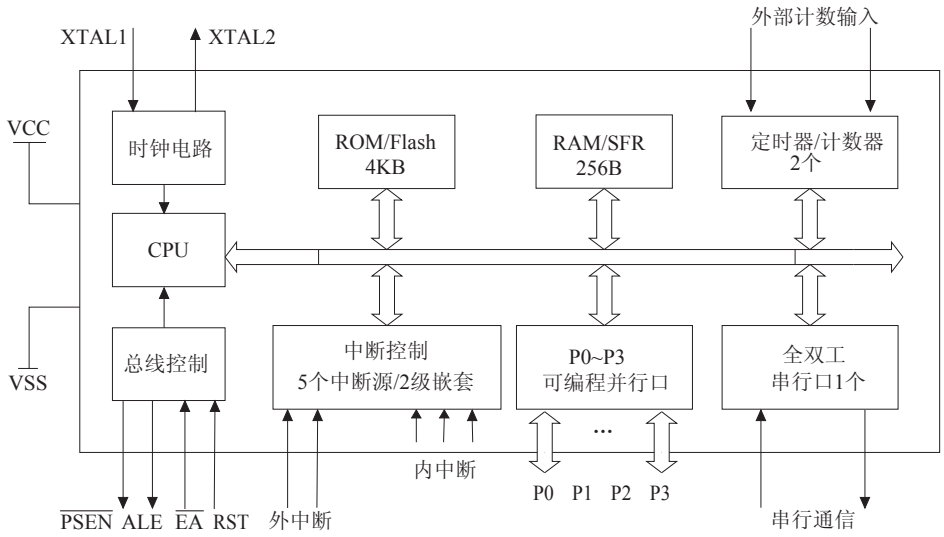


图 1-3 单片机结构框图

单片机基本组成是指在一块IC单片封装内通过片内总线连接的各功能模块。虽然制造商对于不同的应用领域开发了不同型号的单片机系列，但是基本组成是相同的或类似的，单片机的内部结构如图1-3所示。其主要模块有：

1. 一个8位中央处理单元（简称CPU）

CPU主要由运算器、控制器和位处理器等组成。CPU是单片机的核心部件，主要完成控制信号产生、数据传输和数据的算术、逻辑运算、位操作等，8位是指数据宽度为八个二进制位（bit）。

① 运算器：运算器包括算术、逻辑部件ALU，累加器ACC，程序状态寄存器PSW、寄

存器B等。算术、逻辑部件ALU完成数据的算术、逻辑运算以及循环、移位等操作。ACC是送入ALU部件进行运算的一个数据源，当与另一个数据源在ALU中完成运算后结果馈送至ACC。所以累加器ACC是工作最繁忙的一个寄存器，在指令中用字符A表示。

PSW反映CPU的运算结果的状态，是了解CPU工作状态的窗口。B是一个8位寄存器，常用来进行乘法、除法运算以及传送数据。

② 控制器：主要由程序计数器PC，数据指针DPTR，指令寄存器IR、指令译码器ID与控制电路组成。

程序计数器PC是一个16位程序字节地址计数器，PC中的内容就是下一条被取指令的地址，执行完一条指令PC自动加1，PC中内容的变化反映程序运行的流向，PC表示的范围是0~2^{16}，2^{16} = 65536（64KB）。PC是不可访问的。指令寄存器IR是一个专用寄存器，存放指令操作码。指令经指令寄存器IR送指令译码器ID，由ID译码结果形成操作的控制逻辑信号，执行一条指令规定的操作。

2. 存储器

由存储字节单元（八个bit）组成、每个存储单元都有一个唯一编号，称之为地址，存储器在单片机中存放数据与指令（可以统称为代码），存储器主要有以下两类：

① 芯片内含数据存储器（RAM）：容量128字节，可以进行读出和写入。用于存放指令执行过程中数据的暂时保存。另外还有多个特殊功能寄存器，统一编址在后128字节中，其功能后面叙述。片内RAM若容量不够可以进行芯片外部扩展。

② 片内程序存储器（Flash或ROM）：容量4KB，提供用户存放为完成一个任务（如实现一个控制）所编写的程序。程序执行过程之中，只能进行读出，PC依次从中取出一条指令（或者是数据表中一个数）。片内ROM若容量不够也可以进行片外扩展。不同系列不同型号容量有所不同。在应用中一般是扩展片外RAM。

3. 片内其他硬件资源

除了CPU与存储器外还有其他硬件部件，这里统称为硬件资源。

① 两个16位定时器/计数器，配备了四种运行模式供选择。实现单片机对于时间的定时和外部事件的计数应用。

② 具有五个中断源的中断控制系统，可以实现二级中断嵌套。提供单片机对于一些突发的、随机的事件的处理。

③ 四个8位并行的输入/输出端口（简称I/O口），定义为P0、P1、P2和P3端口。这些端口线既可以单个端口使用，也可以组合成总线形式使用。

④ 一个全双工UART（通用异步接收发送器）串行I/O口，可实现按位（Bit）进行数据传输，主要用在单片机之间、单片机与PC机之间的通信。

⑤ 片内振荡器和时钟发生电路。通过外接晶体振荡器，使单片机作为一个复杂的同步时序电路，严格同步在时钟信号的时序下工作。

上述三部分是MCS-51系列单片机，例如：AT89C51型号单片机在40脚的封装内所具备的基本电路组成，这些电路模块主要是理解其基本工作原理（内特性）和掌握具体的应用（外特性），对于片内不存在的电路如AT89C51型号中没有A/D转换电路，则需要外加电路，然后利用单片机的外特性进行控制。

1.2 单片机引脚与功能

学习目标

1. 熟悉MCS-51系列单片机的40个引脚名称。
2. 掌握40个引脚的功能。

导　入

二极管有阳极、阴极2个引脚；三极管有基极、集电极和发射极3个引脚；专用IC模块有多个引脚，每个引脚都有不同的功能，不能接错，同样单片机要正常工作必须正确应用40个引脚。

单片机应用最小系统组成以及在此基础上组成更加复杂的单片机应用系统，对于引脚功能的理解是十分必要的。引脚是连接片内和片外的通道，片外电路的信号通过引脚传入CPU；CPU执行指令后按一定时序送出控制信号对外围电路进行操作，这些都需要通过引脚。89C51单片机共有40个引脚，具体序号分布如图1-4 所示，这是以双列直插封装（DIP）形式为对象，对于系列中各种型号单片机引脚从功能角度理解是可以互相兼容。对于40个引脚，如果是从单个引脚去熟悉相对比较慢，一般方法是对于引脚按功能进行归类，然后细化到每个引脚，这样比较容易把握。

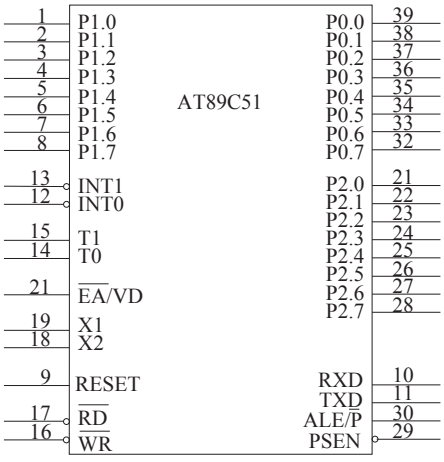


图 1-4 引脚分布图

1.2.1 电源与时钟引脚

1. 电源引脚VCC和VSS

VCC（40脚）：电源端，+5V。

VSS（20脚）：接地端。

2. 时钟电路引脚XTAL1和XTAL2

XTAL1（19脚）：片内反相放大器输入端，外部接晶体与微调电容的一端。

XTAL2（18脚）：片内反相放大器输出端，外部接晶体与微调电容的另一端。

片内反相放大器构成内部振荡电路，通过外接石英晶体、两个微调电容构成自激振荡器。也可以采用外部时钟电路。

1.2.2 控制信号引脚

1. RST/Vpd（9脚）

复位（Reset）信号输入端，高电平有效，引脚上只要出现超过两个机器周期以上的高电平就可以使单片机复位。复位后电路状态后面叙述。引脚的第二功能是备用电源输入端Vpd，主要功能是主电源产生故障时，此引脚接备用电源，保持片内RAM的数据不丢失。

2. ALE/$\overline{\text{PROG}}$（30脚）

ALE（Address Latch Enable/Programming）：地址锁存允许信号端，作用是当访问片外存储器时，ALE信号用于低8位地址的锁存使能信号。若不需要访问片外存储器时同样对外输出脉冲，输出的频率为外加晶体振荡器频率f_{osc}的1/6。ALE引脚驱动负载的能力为八个LS型TTL(低功耗甚高速TTL)负载。

$\overline{\text{PROG}}$为引脚的第二功能，对于片内EPROM的单片机在进行程序写入时（称为固化程序），引脚用作编程脉冲输入端。

3. $\overline{\text{PSEN}}$（29脚）

$\overline{\text{PSEN}}$（Program Store Enable）：程序存储器允许输出信号端，此引脚输出的负脉冲作为读取片外存储器的选通使能信号，在电路上连接外接程序存储器的$\overline{\text{OE}}$（输出允许）脚。$\overline{\text{PSEN}}$引脚驱动负载的能力为八个LS型TTL负载。

4. $\overline{\text{EA}}$/Vpp（31脚）

$\overline{\text{EA}}$（Enable Address/Voltage Pulse Of Programming）：片外程序存储器地址选用端/固化程序电压输入端，当$\overline{\text{EA}}$引脚为低电平时，只选用片外程序存储器；当$\overline{\text{EA}}$为高电平时，复位后先访问片内程序存储器，执行片内程序存储器中的指令，由于片内只有4KB容量的ROM，当超过4KB（地址为0FFFH）后自动转向片外程序存储器地址。

Vpp为第二功能引脚，对于Eprom型单片机在写入程序时的电源脚，这一编程电压高于工作电压（例如：+12V）。

1.2.3 输入/输出端口（I/O口）P0，P1，P2和P3引脚

1. P0口（P0.0~P0.7，39~32脚）

P0口有8个引脚，对于片内是属于漏级开路的8位准双向I/O口。P0口具备两种功能：一个是普通的I/O口；另外一个是作为地址/数据的总线端口(总线是指同类性质的口线组合)。当用作地址/数据总线功能时，就不能作为I/O口了，地址与数据总线之间也是分时使用，是属于分时复用。

① P0口作为地址/数据总线口：当系统需要进行外部存储器扩展时，通过片内控制，使P0口作为系统低8位的地址/数据线分时使用，地址线传输是单方向的，从CPU到引脚，通过引脚

连接片外电路；而数据线双向的，可以从引脚读入或者是CPU向引脚输出。当作为地址/数据总线时，片内输出级驱动电路工作在推拉式状态目的是增大负载能力，内部上拉电阻也有效。

② P0口作为普通的I/O口：当P0口内部控制为一般I/O口应用时，不能作为地址数据线使用，其中的每一位的驱动输出是一对场效应管工作于漏级开路的电路形态。

当作为I/O口的输入：按照内在电路结构分为读引脚和读锁存器口二种情况。读引脚是从引脚状态直接读入到内部总线；读锁存器口是指读端口内的锁存器。由于内部电路的特点，当需要进行读操作功能时，必须先向端口写入“1”使引脚处于高阻抗输入。这也是称之为准双向口的原因。

为什么需要有读锁存器口（或称为读端口）呢？主要是针对有部分指令如：ANL、XRL、ORL等执行完成的是“读—修改—写”操作，因为直接去读引脚，当外接电路原因有可能导致读入引脚电平有误，所以直接设计成读I/O口的锁存器，读完后进行CPU执行相关的逻辑操作，然后直接写端口。

当作为I/O口的输出：通过内部总线传送信号至引脚。由于这时输出级为漏极开路所以必须外接上拉电阻（一般选10K左右电阻）。用作地址/数据总线时不需要外接上拉电阻。

P0口的每一位带负载能力是可以驱动八个LS型TTL负载。

2. P1口（P1.0~P1.7，1~8脚）

P1口是内部具备上拉电阻的8位准双向I/O端口。P1口的负载能力，可驱动四个LS型TTL负载。这是一个不具备其它功能的真正的输入/输出端口，端口的每一位在输出高电平时有内部上拉电阻提供拉电流负载。当P1口用作输入时必须向端口先写入“1”（准双向口）。

3. P2口（P2.0~P2.7，21~28脚）

P2口的8位内部结构类似P1口，但是P2口功能却类似于P0口。

① 当P2口作为一般I/O口时，与P1口相同。

② 当需要扩展片外存储器时，P2口作为高8位地址与P0口（低8位）合在一起形成最大16位地址线可寻址空间为2^{16}。若在扩展过程中，当不需要全部用完8根口线，比如多余P2.7这根线，则P2.7口线可以作为I/O口线用。同样用作地址线后就不能作为I/O口线。P2口的负载能力同P1口。

4. P3口（P3.0~P3.7，10~17脚）

P3口同样是内部有上拉电阻的8位准双向I/O端口。P3口的每一位与P1口内部电路相比较就是多了一个对于第二功能的控制电路，P3口对应第二功能参见表1-1。在实际应用时应该优先考虑第二功能设计，当不需要考虑第二功能应用时就可以作普通I/O口用。P3口作为输入时也必须向端口先置1，P3口的负载能力同P1口。

注意：P口除了I/O口以外的各种功能，如P0、P2口作为地址线低、高8位线，P3口的第二功能等都具备自动识别能力。

表 1-1　P3口的第二功能对照表

引　脚	名　称	引脚功能
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输入口
P3.3	$\overline{\text{INT1}}$	外部中断1输入口
P3.4	T0	定时器/计数器0外部输入口
P3.5	T1	定时器/计数器1外部输入口

（续表）

引 脚	名 称	引脚功能
P3.6	$\overline{WR}$	片外数据存储器的“写选通”输出端
P3.7	$\overline{RD}$	片外数据存储器的“读选通”输出端

1.3 存储器结构

学习目标

1. 了解单片机片内ROM的容量与分布。
2. 了解单片机片内RAM的容量与分布。
3. 熟悉特殊功能寄存器名称、地址与应用功能。
4. 熟悉位寻址区域。

导 入

外出旅游，离不开住宿，不同的旅馆有着不同的结构。可入住的全部客房称之为容量，每个客房都有一个号码，客房有单、双人之分。这与单片机的存储器系统有着惊人的相似之处。存储器是数据的仓库，数据在存储器中的读出与写入，必须对应着每一个存储单元的地址，总的存储单元的集合，称之为存储容量。

MCS-51系列单片机的存储器结构是采用哈佛（Harvard）结构，即程序存储器与数据存储器分开编写地址（内部各寄存器与片内RAM统一编址），有各自的地址空间。完整的存储空间分布如图1-5所示。

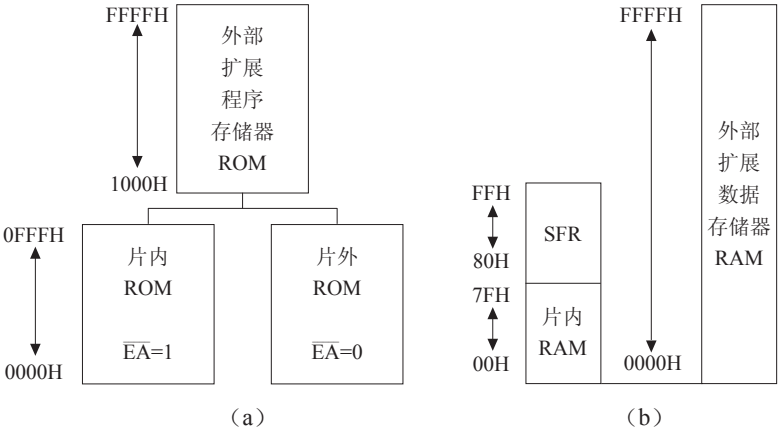


图 1-5 存储空间分布图

（a）程序存储器分布 （b）数据存储器分布

从图1-4的存储器分布图来看，可以归纳如下：

① 片内、片外程序存储器。

② 片内、片外数据存储器。

③ 每一个存储单元标识一个唯一序号——地址。

④ 程序存储器、片内数据存储器、片外数据存储器编址相互有重叠部分。

当CPU复位以后，存储单元都取得一个唯一的序号——地址。对于片内与片外的重叠地址部分，MCS-51是通过不同的指令加以区别。对于片内、片外程序存储器是用MOVC指令；对于片外数据存储器是用MOVX指令、对于片内数据存储器是用MOV指令。

1.3.1 程序存储器

程序存储器（ROM）是存放经过编译以后的程序代码和表格化的数据。从图1-4中可以观察到其分为片内与片外程序存储器。通过16位程序计数器PC寻址访问，最大访问（寻址）空间是64KB。

1. 片内、片外程序存储器

① 片内程序存储器　地址分布0000H~0FFFH，当存放的代码不超过存储器容量时，可以不进行片外扩展，若不够则需要扩展。

② 片外程序存储器　地址分布1000H~FFFFH，总的容量约60KB。片内、外存储器的编址是连续的、序号是唯一的，总的容量为64KB。系统需要扩展存储器时，除了正确的电路连接外，还需要有一个硬件引脚的配合，就是$\overline{EA}$（片外程序存储器选用端）需要接高电平（接VCC），确保上电复位以后从片内0000H开始执行，到了0FFFH后自动转向1000H开始的片外存储器。若$\overline{EA}$接低电平（接地），则使用的全部是片外程序存储器，也就是从0000H~FFFFH。

2. 程序存储器中的系统单元

当单片机上电复位以后，程序计数器PC的内容是0000H，也就是说系统总是从0000H起始地址开始运行指令。除了0000H以外，系统对于表1-2所示的单元有固定的使用，这些固定单元的地址称之为中断矢量地址。

表 1-2　系统固定单元

中断源	中断矢量地址
外部中断0	0003H
定时/计数器0	000BH
外部中断1	0013H
定时/计数器1	001BH
串行口	0023H

仔细观察表1-2可以发现，从上电复位后PC自动指向0000H单元，而0000H与0003H单元，或者是0003H单元与000BH单元，其单元间隔长度都非常小，只有在0023H单元后面系统没有固定，所以采取的方法是在这些固定单元后面存放一条无条件转移指令转向程序代码存放的入口地址，由此可见一般应用程序至少放在0023H单元以后。

1.3.2 数据存储器

数据存储器主要存放中间运算结果。从图1-5中观察数据存储器（RAM），同样也可分为片内、片外数据存储器，片外数据存储器如同程序存储器仅仅是一个容量大小的概念，而片内数据存储器内容就比较丰富了。

图 1-6 片内RAM地址空间

1. 片内数据存储器

片内数据存储器，地址分布00H~7FH，依据图1-6数据存储器可以分为三个区域。

① 通用工作寄存器区：通用工作寄存器区地址分布为00H~1FH，有32个内存单元，共分为四组。从第0组到第3组，每组为八个8位工作寄存器，寄存器序号依次是R0~R7，每个寄存器序号对应一个内存单元的地址，如：第0组R0~R7，对应地址00H~07H；第3组R0~R7，对应地址18H~1FH。

单片机复位后，自动选择第0组为当前运行的寄存器组，在编写程序的过程中可以通过设置PSW中的RS1、RS0位来选择当前工作寄存器组。

② 位寻址区：位寻址是指在操作过程中可以按一个字节中的某一位进行读写操作，这是单片机的特殊点。当可以寻址的位有多个时就构成了位寻址区。在RAM区域中地址为20H~2FH的16字节单元即可以进行字节寻址，也可以按位进行寻址，对应关系参见表1-3。

表 1-3 RAM位寻址区位地址表

字节地址	MSB			位地址				LSB
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	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

③ 通用RAM区：其地址分布为30H~7FH，通常是在编写程序时用于存放运行过程中的临时数据存储，也可以作为堆栈区域。为什么可以作为堆栈区（堆栈概念后续解释）呢？由于初始化后SP自动指向07H单元，则入栈的内容从08H单元开始，这一区间是与工作寄存

区重叠，若在程序中重新设置SP的指向，一般就是移至本区域。

2. 特殊功能寄存器

特殊功能寄存器（简称SFR）是位于RAM区高128B单元，地址是从80H~FFH，不连续地址分布着二十个特殊功能寄存器，SFR的符号、名称和对应地址。见表1-4。

表 1-4　89C51系列单片机的特殊功能寄存器表

符　号	名　称	地　址
*ACC	累加器ACC	E0H
*B	寄存器B	F0H
*PSW	程序状态寄存器	D0H
SP	栈指针	81H
DPTR	数据指针（分DPH，DPL）	83H，82H
*P0	P0口锁存寄存器	80H
*P1	P1口锁存寄存器	90H
*P2	P2口锁存寄存器	A0H
*P3	P3口锁存寄存器	B0H
*IP	中断优先级控制寄存器	B8H
*IE	中断允许控制寄存器	A8H
TMOD	定时器/计数器工作方式寄存器	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）	BDH
TL1	定时器/计数器1（低字节）	8BH
*SCON	串行口控制寄存器	98H
SBUF	串行数据缓冲器	99H
PCON	电源控制及波特率选择寄存器	87H

注：*表示的SFR可以按位寻址，也可以按字节寻址功能。

下面概述部分寄存器，另有部分寄存器在后续章节中介绍。

① 累加器ACC：是一个8位的寄存器，指令中简写为A，是属于传送数据最繁忙的一个寄存器。

② 寄存器B：8位寄存器用于运算乘、除法，也可以作为普通寄存器使用。

③ 程序状态寄存器PSW：PSW是8位的寄存器，是反映CPU运行状态的窗口寄存器，它的各位包含了程序执行后的状态信息，供编程序时查询或判断用。各位的含义见表1-5。

表 1-5　PSW寄存器各位的含义

PSW	D7H	D6H	D5H	D4H	D3H	D2H	D1H	D0H
D0H	CY	AC	F0	RS1	RS0	OV	—	P

CY：进位/借位标志位，当算术运算中的加法/减法产生最高位的进位/借位时，由硬件自动产生置“1”信号，否则清“0”，当进行位操作时，CY充当位累加器，功能相当于累

加器ACC。

AC：辅助进位标志位，当算术运算中的加法/减法产生低4位向高4位进位/借位时，由硬件自动产生置“1”信号，否则清“0”，对于BCD码运算时可用作调整位。

F0：用户标志位

RS1和RS0：工作寄存器组选择控制位。若RS1 RS0=00，则选择第0组寄存器；若RS1 RS0=11，则选择第三组寄存器。

OV：溢出标志位，当进行算术运算时，数值超出－128~＋127时，即数据产生溢出，有溢出该位为“1”，否则为“0”。

P：奇偶校验位。P=1 ACC中有奇数个1；P=0 ACC中有偶数个1。

④ 栈指针SP：对于栈指针的理解，首先需要理解的是栈的概念，我们可以通过枪械中用于射击的子弹盒来比喻，假设子弹盒可以存放20粒子弹，原先是空的，我们把子弹一粒一粒压入，装满后进行射击，射击时最后压入的第20粒子弹第一个出枪膛，最早压入的第1粒子弹最后一个打出。这里子弹盒相当于一个堆栈区，子弹的进出可以理解为压栈（或入栈）和出栈，其机理是先进入后弹出，而堆栈指针SP就相当于是顶出子弹的撞针，当子弹盒是空的，则SP所在位置在底部，当子弹盒是满的，则SP在顶部，顶和底是自动管理的，这同样符合在单片机中的应用，栈指针SP指示栈顶与栈底。

单片机的堆栈区实质就是RAM中一块“后进先出”的对于数据进行保护的临时区域，该临时区域通过栈指针寄存器SP进行管理。在单片机工作过程中需要用栈进行断点地址、现场数据的保存。通常哪些数据需要用到堆栈区域呢？比较典型的运用如下：

- 主程序调用子程序时断点地址的保护，断点地址意思是当主程序运行完第n条指令转向子程序，而当子程序运行结束以后，返回到n+1指令继续执行，否则单片机运行到n+2条指令则出现错误。
- 现场数据的保护。由于单片机资源比较紧凑，有时在调用子程序时，对于累加器A，或一些寄存器或者是有关的标志位需要用栈进行保护。
- 临时数据暂时存放。

其次是栈的管理方式：栈的工作方式有自动方式和指令方式，对于调用子程序和中断的系统是自动把断点压进栈内（简称入栈或压栈），一个数据进栈时先由SP加1，后数据进栈，恢复数据时自动弹出（简称出栈），出栈时先弹出数据，然后SP减1，运行的是先进后出方式。指令方式是运行入栈指令PUSH、出栈指令POP。

初始化后SP自动指向07H单元，栈的深度在理论上是小于RAM区域任何一片位置。

⑤ 数据指针DPTR：DPTR是一个专用的由两个8位DPH（地址是83H）、DPL（地址是84H）合成为一个16位用于存放地址的寄存器，在DPTR寄存器中存放的地址一般作为基地址去访问片外ROM存储器和用间接寻址方式寻址片外RAM。既可以16位应用，也可以分为两个8位寄存器使用。DPTR与PC在应用上有一个显著的区别就是DPTR寄存器是可以访问的，PC则是不可访问的。

1.4 单片机应用最小系统

学习目标

1. 理解单片机最小系统的概念与硬件组成。
2. 掌握复位电路与单片机复位状态。
3. 掌握晶体振荡电路与时序含义。
4. 了解CPU运行过程与工作方式。

导　入

自行车是常用交通工具，有两个轮子、一个掌握方向的龙头、座椅和一套链条组成的传动系统。我们称它为最简单的自行车，换言之就是自行车的最小系统。如果在后面配上一个架子可以带上一个人或者携带一定量的物品；如果在传动系统上配备一个小功率电机可作为一辆电动车，这称之为功能扩展。单片机的应用平台就是单片机应用最小系统的扩展，能使最小系统运行就能初步体验单片机是如何运行工作的，且在此基础上可以不断深化；反之感觉单片机学习、应用比较困难，从硬件角度来观察就是单片机的最小应用基础没有掌握好。

所谓最小系统，就是指单片机能正常运行指令的最少硬件组合且在此基础上可以进行扩展的硬件系统。从前面的图1-1中可以看出，单片机最小系统除了核心单片机芯片以外，还需要配备晶体振荡器电路、复位电路和5V电源。

1.4.1 时钟电路与时序

时钟电路产生的是操作节拍，犹如一个乐队演奏一个乐曲、一个领操员引领大家做队列操。从数字电路角度认识，单片机是一个复杂的同步系统，有条不紊地工作是要在时钟信号产生的时序控制下进行，换言之单片机各功能模块都是以时钟脉冲为基准。时钟脉冲频率高，单片机运行速度快，反之则慢。

1. 时钟电路

89C51典型振荡电路如图1-7所示。在单片机内部设计了一个反相放大器，通过芯片引脚XTAL1、XTAL2跨接一个石英晶体振荡器和两只微调电容构成一个自激振荡器，产生脉冲直接提供给内部时钟电路。石英晶体振荡器一般可以选择0~24MHz，比较典型的应用是选配11.0592MHz，电容选30pF左右。

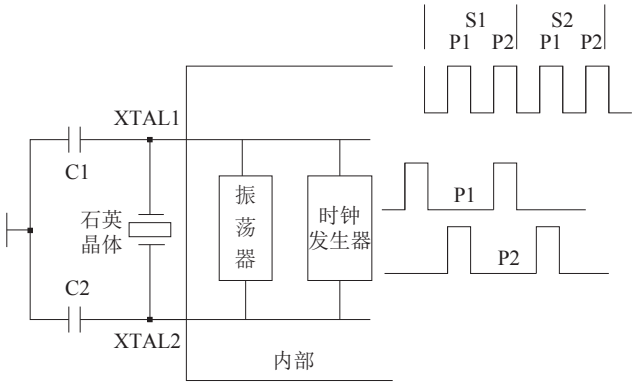


图 1-7 振荡电路

单片机也可以选择外接振荡器来产生时钟。

2. 时序

时序就是以振荡周期为计时单位的一串脉冲序列，在89C51中分别被定义为节拍、状态、机器周期与指令周期。

① 外接晶体振荡器的频率f_{osc}：实际用于计算时是用晶体振荡器其振荡频率f_{osc}的倒数一周期，作为时序中的基本时间单位。

② 状态周期：单片机内在的时钟发生器是一个二分频电路，f_{osc}信号经两分频后，形成两相时钟信号，分别定义为节拍P1和节拍P2（分2路作为CPU同步信号），时钟信号的周期定义为一个状态S（State）周期，是振荡周期的2倍，则一个状态S有两个节拍P1和P2，如图1-7所示。

③ 机器周期：89C51定义一个机器周期由六个状态S1~S6组成，每个状态有两拍，所以一个机器周期有六个状态，十二个振荡周期，分别为S1P1，S1P2，S2P1，S2P2，S3P1，S3P2，……S6P1，S6P2。由于每条指令的字节数不同（占据内存单元数不同）与片内、外存储器的不同，所需要的机器周期也就不同了。

④ 指令周期：运行一条指令所需要的时间。89C51的指令有单字节、双字节和三字节等。所以单片机运行一条指令有1~4个机器周期。

单片机的各种周期的相互之间的关系见图1-8。

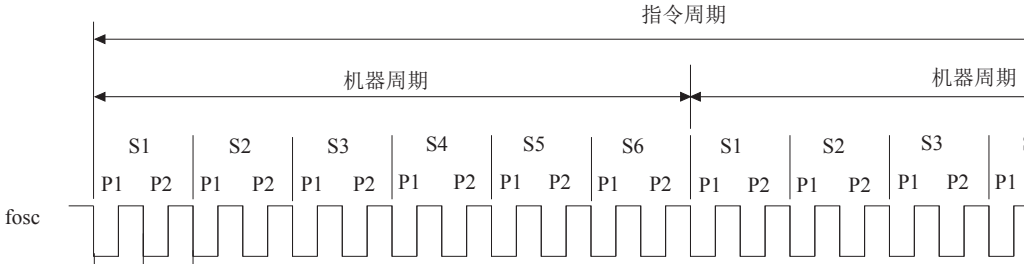


图 1-8 单片机各周期的关系

【例1-1】 若选用f_{osc} =12MHz晶体振荡器，参照图1-8。则

一个振荡周期=1/ f_{osc}=1/12MHz=0.0833μs

一个机器周期是（$12/f_{osc}$）=（12/12MHz）=1μs。

指令周期=（1~4）机器周期= 1~4μs。

由于单片机组成的系统是一个实时系统，无论是定时器/计数器还是串行通信、中断系统，时间的概念和应用都非常重要。通过学习指令系统后，就指令的运用来说，需要多用功能相同而机器周期数少的指令，以节省系统总的运行时间。

1.4.2 复位电路与复位

单片机通电运行以后，若遇异常发生，系统必须重新开启运行。在调试系统的过程中也要经常进行复位操作，这里提及的复位就是使系统各部件都统一回归到初始化状态，MCS-51单片机采用外接复位电路，按照系统应用的功能，可以采用上电自动复位与手动复位两种方法。

1. 复位电路

① 自动复位电路：如图1-9（a）所示。上电自动复位电路通过一个阻容电路，连接到RST引脚，只要电源VCC的上升时间不超过1ms，利用上电的自动充电过程完成复位。对于复位电路的电阻、电容参数值，要求满足当电路稳定后只要在RST引脚上保持高电平大于两个机器周期就可以完成系统复位过程。

② 手动复位电路：如图1-9（b）所示。手动复位电路由于可以人工进行干预复位，所以必须外加一个按键，构成一个按键复位电路。对于RST引脚来说，只要满足自动复位的要求就可以了。

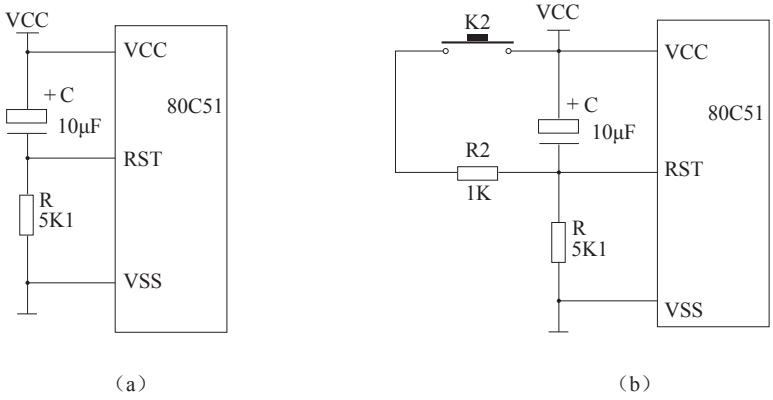


图 1-9 复位电路

（a）自动复位电路　（b）手动复位电路

值得说明的是：在单片机内部有一个施密特触发器与RST引脚相连，能自动进行噪声抑制。

2. 复位状态

电路复位其实质是完成一个拉回到初始化状态的动作。那么，从单片机角度理解通过复位后是处在一个什么样的状态呢？表1-6反映各寄存器复位以后的状态。

如：PC复位以后内容为0000H，表示从0000H开始执行指令；又如P0~P3在初始化以后其状态为FFH表示在实际使用中要考虑的一个状态，ACC=00H，表示累加器已清0等。

表 1-6 特殊功能寄存器复位状态

寄存器	复位状态	寄存器	复位状态
PC	0000H	TCON	00H
ACC	00H	TL0	00H
PSW	00H	TH0	00H
SP	07H	TL1	00H
DPTR	0000H	TH1	00H
P0~P3	FFH	SCON	00H
IP	xxx00000B	PCON	0xxx0000B
IE	0xx00000B	SBUF	不定
TMOD	00H		

注：表中x表示随机状态。

3. 单片机应用最小系统

至此已经基本明确了单片机应用最小系统的组成，它包括：一块单片机芯片（片内有存储器，由于存在片内存储器，所以$\overline{EA}$要连接正确），外围需要加晶体振荡器且配有微调电容、复位电路，连接电源/接地，如图1-1所示。在此硬件基础上写入程序（指令），单片机就可以运行了。同时通过图1-1我们还可以观察到大量的I/O口可以进行扩展使用，比如可以用I/O口驱动发光二极管、可以外接电路驱动电动机、可以接一个温度传感器，可以配键盘、LED显示器构成一个测温仪等。

4. 单片机工作过程和运行方式

有了最小系统，只是一个硬件基础，在写入程序后单片机才能运行，那单片机又是如何工作的呢？

① 工作过程概述：有了硬件的最小系统，程序经过编译软件编译以后下载到ROM（或Flash）中，单片机就可以运行了。所谓工作过程指的就是逐条运行指令过程，单片机执行一条指令历经读取指令→翻译指令→执行指令的过程。如：执行MOV A，#00H指令，设该指令被编译以后存放在地址0100H和0101H单元中，0100H存放内容是代码74H，0101H单元中存放的代码是00H，代码是经过编译软件翻译后得到的机器码。

读取指令是把PC的当前值（0100H）送到地址寄存器中，一方面是PC内容自动加1为0101H即指向下一个单元，另一方面是地址寄存器0100H内容经过片内地址总线选中0100H单元，此时CPU控制读有效，74H经片内数据总线送指令寄存器。

译码与执行阶段是完成对于74H的翻译，通过译码电路得出指令功能是送数据00H到累加器ACC，且数00H存放在0101H单元，由于此时PC已经指向0101H，同样按取指过程取出立即数00H，CPU控制读有效信号，通过数据总线传送立即数到累加器ACC，至此一条指令执行完毕。PC自动指向0102H单元，执行一条指令用时一个机器周期（即按照理论计算耗时1μs）。CPU 整个工作过程就是逐条执行指令直至程序结束。

② 单片机低功耗工作方式：单片机运行必须是处在较低的电压与低功耗状态下。目前，有的型号单片机可以在1.8V电压下工作，可以配用纽扣电池，功耗降至μA级。89C51单片机是属于CHMOS工艺的单片机，不但耗电少，而且还提供了待机模式与掉电保护模式两种节电的工作方式，在其特殊功能寄存器SFR中有一个电源控制寄存器PCON，各位的分

布见表1-7，通过设定相关的位就能实现待机模式与掉电保护模式。

表 1-7 电源控制特殊功能寄存器PCON

PCON	D7H	D6H	D5H	D4H	D3H	D2H	D1H	D0H
（87H）	SMOD	—	—	—	CF1	CF0	PD	IDL

➢ 待机模式

所谓待机模式（空闲工作方式）是指振荡器继续工作，但是内部时钟不向CPU提供，只提供给中断系统、定时器/计数器、串口等电路模块、特殊功能寄存器保持原来状态、端口状态也不变。ALE、$\overline{\text{PSEN}}$保持高电平。

执行指令将PCON寄存器中IDL置1（PCON.0=1），单片机进入待机状态。

退出待机方式可以通过中断响应方式或者是硬件复位方式。退出方式的最终结果都是使IDL清0（PCON.0=0）。但在恢复运行后RAM中数据保持不变。待机方式下虽然工作电压不变，但是电流值下降。

➢ 掉电保护模式

掉电保护模式是停止内部振荡器工作，效果是一切部件都停止工作。但是RAM区数据和SFR内容被保存，ALE、$\overline{\text{PSEN}}$为低电平。

进入掉电保护方式只要执行指令置位PD（PCON.1=1）。退出掉电保护方式只有一种方式就是硬件系统复位，复位以后SFR中内容被初始化和RAM区内容不变化。

1.5 数制、编码与符号数

学习目标

1. 熟练单片机的数制以及数制间的转换。
2. 掌握编码与编码方式。
3. 了解符号数的大小与表示。

导　入

单片机能够识别的代码只能是二进制代码，我们常用的是十进制，而且有诸如‘a’、‘A’等一连串的字符以及回车符等符号，这只能通过编码等一系列方式使单片机能识别。对于数制、编码的掌握将有助于程序的编写。

1.5.1 常用数制

在单片机中常用的数制主要是二进制、十六进制与十进制。其中二进制是CPU能识别的，另外两个是人们编写程序经常使用的，十六进制与十进制可以通过编译软件转换成CPU识别的二进制。

1. 十进制数

这是我们在日常生活中习惯使用的数制，对于十进制数中任何一个数都可以用十个不同的数码0~9按规则排列，这是一种以10为基数的数制，其计数规律为“逢十进一”。十进制数（Decimal）用D表示。

【例1-2】 123.456 该数可以写成：

$$123.456 = 1\times10^{2} + 2\times10^{1} + 3\times10^{0} + 4\times10^{-1} + 5\times10^{-2} + 6\times10^{-3}$$

式中 10为计数制的基数，10^{2}、10^{-3}等为该位的权，分别是百位和$1/10^{3}$位。1~9中的任何一个数作为权数的系数。

2. 二进制数

单片机是超大规模集成电路的产物，所以在CPU中运转的机器代码就是二进制码。二进制数是以0、1两个数码按规则排列。计数规律为“逢二进一”，是以2为基数的计数方法。二进制数（Binary）用B表示。

对于任何一个二进制数都可以用类似于十进制数方法：

【例1-3】 $10101.010B=1\times2^{4}+0\times2^{3}+1\times2^{2}+0\times2^{1}+1\times2^{0}+0\times2^{-1}+1\times2^{-2}+0\times2^{-3}$

式中 2为计数制的基数，2^{2}、2^{-2}等为该位的权，分别是4和1/4。0，1中为权数的系数。

用二进制代码计数简单、可靠，反映了电路状态的特征，而且加减乘除方便。不足之处是人们对于二进制的阅读交流，编写程序时会带来不小的麻烦，所以往往是采用另外一种数制——十六进制数，因为4位二进制可以表示为1位十六进制数。

3. 十六进制数

十六进制数（Hexadecimal）用H表示，每位数码有16个，除0~9外，添加A~F(A~F对应十进制数中11~15)。计数规律为“逢十六进一”，以16为基数的计数方法。

【例1-4】 $AB.4H=10\times16^{1}+11\times16^{0}+4\times16^{-1}$

数制多随之而来的就是识别的困难，为了区别不同数制的书写，通常用两种标识方法：一是数加括号并标注数制代号在数的右下脚，如$(1001)_{10}$、$(1001)_{2}$、$(1001)_{16}$分别表示十进制数、二进制数、十六进制；第二是用数制的英文单词的第一大写字母表示，如1001B、1001D、1001H分别表示二进制数、十进制数和十六进制数，其中十进制数的下脚标10或D可省略。

1.5.2 数制之间转换

1. 十进制数与二进制、十六进制数的相互转换

① 十进制与二进制数互换

➢ 十进制转换成二进制数

【例1-5】 十进制数$(123.625)_{10}$转换为二进制数。

对于整数部分采取的方法是：“除2取余数”。把得到的第一个余数放在二进制数最末位，最后得到一个余数排在最高位。$(123)_{10}=(1111011)_{2}$

对于小数部分采取的方法是：“乘2取整数”。用小数部分乘2，取出乘积的整数部分，第一个整数为小数点后的第一位；后面依次排列，直至小数部分为0或者是按实际需要的精度确定小数后面的尾数。

$(0.625)_{10}=(0.101)_2$

所以 $(123.625)_{10}=(1111011.101)_2$

➢ 二进制数转换成十进制数

方法是把整个二进制数进行按权展开，然后相加。

【例1-6】把二进制数$(10010.01)_2$转换成十进制数为

$$(10010.01)_2=1\times2^4+0\times2^3+0\times2^2+1\times2^1+0\times2^0+0\times2^{-1}+1\times2^{-2}$$
$$=16+2+0.25=(18.25)_{10}$$

② 十进制与十六进制数互换

➢ 十进制转换成十六进制数

【例1-7】十进制数$(123.625)_{10}$转换为十六进制数。

同样，整数部分采取的方法是：“除16取余数”。即连续除以16，余数部分同样先出现余数排在最末位，后出现的余数排在最高位，直至商为0。

对于小数部分采取的方法是：“乘16取整数”。用小数部分乘以16，取出整数部分，第一个整数为小数点后的第一位；后面依次排列，直至小数部分为0或者是按实际需要的精度确定小数后面的尾数。所以$(123.625)_{10}=(7B.A)_{16}$。

➢ 十六进制数转换成十进制数

把整个十六进制数按权展开，然后进行相加，得到十进制数。

【例1-8】把十六进制数$(3A.5)_{16}$转换成十进制数为

$$(3A.5)_{16}=3\times16^1+10\times16^0+5\times16^{-1}=48+10+0.3125=(58.3125)_{10}$$

对于一个包含小数的十进制数，无论是转换成二进制或者十六进制都是整数与小数分开处理，然后连接成一个二进制或者是十六进制数。而二进制、十六进制数转换成十进制数均采用“按权展开后相加”的方法。要注意的是在十六进制数中A为10，以此类推F为15。

2. 二进制与十六进制数的互换

① 二进制转换成十六进制数：因为$2^4=16$，4位二进制数就可以表示1位十六进制数。所以采取方法是以小数点为中心往左、往右取4位表示一位十六进制数。

【例1-9】$(110.011)_2=(0110.0110)_2=(6.6)_{16}$

② 十六进制转换成二进制数：这是上述转换的逆转换，也就是把1位十六进制数用4位二进制数表示。

【例1-10】$(A1.B2)_{16}=(1010\ 0001.10110010)_2$

值得一提的是为了比较完整地说明数制之间的转换，所以一个数有整数部分也有小数部分，而实际在单片机中用的比较多的是整数或者用另外一个名称“定点数”，所谓定点数就是用8位（bit）表示一个数，小数点在最低位后面。

1.5.3 数的表示方法

在数学运算中对于大于0的正数在数值前面用“＋”（或缺省）表示；对于小于0的负数是在数值前面添加一个“－”表示。

由于单片机中CPU是不能识别“＋”、“－”号的，那如何区分正、负数呢？包含现

有的微型计算机在内都作了这样的约定：对于一个8位二进制数如果8位全是数值位，则表示的是无符号数；若要表示正、负数则用8位中最高位用作符号位，后7位是数值位，则称其为有符号数，有符号数在表示正数时最高位是0；负数则是1。这种对符号数字化处理，使整个8位二进制数连同符号位都能被CPU所识别，这样的二进制数（代码）我们称其为机器数（机器码），而机器码所表示的数值被称之为真值。

【例1-11】 0 1011000B=88

由于最高位为0，若表示无符号数或者是有符号数其值比较明确。

1 1011000B，其值是多少呢？若是无符号数则结果很明确为216，如果是有符号数其值是负多少呢？实际单片机中对于有符号数可以有三种表示：

1. *原码*

就是最高为1表示“－”号；用0表示“＋”号，其余为数值位。8位机器数表示的数值范围－127~＋127。

【例1-12】 X0=＋88; 则原码表示$[X0]_{原}$=0 1011000B；

X1=（－88）;则原码表示$[X1]_{原}$=1 1011000B。

2. *反码*

对于正数若用反码表示则同原码相同；负数的反码是保持符号位不变，数值按位取反，即0变1；1变0。

【例1-13】 $[+24]_{反}$= 0 0011000B

$[-88]_{反}$= 1 0100111B

反码特征：

① 8位二进制数用反码表示其数值范围－127~＋127。

② “0”分别可以表示 $[+0]_{反}$=00000000; $[-0]_{反}$= 11111111。

③ 应用时注意对于负数符号位后面的数不是实际数值，而是要取反后才是实际数值部分。

3. *补码*

一个正数的补码与原码相同，0表示正数，后7位为数值位；负数的补码求取方法是：最高符号位为1，7位数值位按位取反后再加1。

【例1-14】 $[+127]_{补}$=0 1111111B

$[-88]_{补}$=1 0101000B

补码特征：

① 8位二进制数用补码表示其数值范围－128~＋127。

② $[+0]_{补}$= $[-0]_{补}$=00000000。

③ 应用时注意对于负数符号位后面的数不是数值而是要把该数减1以后再取反才是其数值大小。

但是，要注意在单片机中给定一个二进制代码就需要事先约定是无符号数还是有符号数，如果是有符号数均是以补码形式出现，那么就有一个还原成真值的问题。另外对于补码表示的负数，可以转化算术运算中的减法为加法（CPU不能做减法），由于一个8位二进制数的最大值是256（称为模），类似时钟的最大数是12，从12点拨到3点钟，正拨3个时钟与倒拨9个时钟都可以得到；这样CPU对减法处理就可以通过转换成补码后做加法可以得到相同的结果。当然注意会产生前面提及的溢出问题，即运算结果超出－128~＋127数值范围。

1.5.4 编码

单片机中运行的机器代码除了数值以外，还需要不同程度处理一些控制符号、字母等，需要用二进制数对控制符号、字母等进行编码。由于处理对象的不同，编码的形式和码制都有所不同。

1. ASCII编码

ASCII（American Standard Code For Information Interchange）码是美国标准信息交换代码的简称，已成为一种通用标准代码，在微机系统中广泛应用。基本的ASCII代码由7位二进制代码组成，最高位是0，所以共有128个字符。其中32个是控制代码，控制代码只有控制作用不能显示或输出，其他字符可以显示与打印。若最高为1，则形成一种扩充ASCII码。对于基本的ASCII代码中部分代码在单片机编程中会经常使用见表1-8，全部代码见附录。

【例1-15】ASCII代码07H只能是“嘟”一声响。

‘B’代码是0100 0010B或42H或66D。

表 1-8 常用字符的ASCII代码（已转换成16进制形式）

	ASCII		ASCII		ASCII		ASCII
BEL（响铃）	07H	0	30H	a	61H	A	41H
BS（退格）	08H	1	31H	b	62H	B	42H
LF（换行）	0AH	2	32H	……	……	……	……
CR（回车）	0DH	……	……	y	79H	Y	59H
SP（空格）	20H	9	39H	z	7AH	Z	5AH

2. 二——十进制编码（BCD码）

BCD码就是用4位二进制数来编码1位十进制数。

【例1-16】35.47D其BCD码0011 0101 . 0100 0111 B

这种码制方便编制程序，因为人们熟悉十进制数，CPU只能运行二进制数，所以对于十进制数只要写成4位二进制就行了。使用时要注意的基本代码为0000~1001这十个有效代码，由于是十进制也应该是“逢十进一”，例如：12D用0001 0010B表示。BCD码的不足会导致机器运行速度下降。

3. 校验码

这是一个在CPU之间进行通信时，由于传输距离、现场状况的诸多因素影响，为使接收方接收的信息可靠无误，产生了对于计算机传输信息过程中的附加校验码发送和接收方接收信息后的解码问题。

校验的编码有：奇偶校验码、循环冗余码校验（CRC）、校验和等。单片机中最简捷的奇偶校验码，就是指在传送字符之外，添加一位奇偶校验码，可以用奇校验，要求编码后的校验码中有奇数个“1”；也可以用偶校验，要求编码后的校验码中有偶数个“1”。运用这种校验方法一旦有错码，只能请求对方重发，无纠错能力。

循环冗余码校验CRC是数据通信领域中最常用的一种差错校验码。其方法是依据发送的二进制代码依据规则产生校验用的监督码，排列在发送代码后面而形成一个新的代码序列，接收方对收到数据重新计算CRC并且与收到CRC相比较，若CRC不一致，表示通信过程有误。

1.6 Keil μVision2集成开发环境的应用入门

学习目标

1. 认识Keil μVision2。
2. 掌握Keil μVision2的基本使用方法。
3. 通过汇编程序掌握单片机的基本应用过程。

导　入

人与环境之间的关系是人们生活中时常讨论的话题，同样在单片机的程序设计完成以后，也需要有的这样的环境，当然这样的环境是在PC机上，来帮助程序员识别语法错误，然后转换成二进制代码。

在PC计算机提供的平台上，有许多工程设计的CAD软件，Keil μVision2是一个基于Windows的集成开发环境（IDE Integrated Development Environment），它包括C编译器、宏汇编、连接器、库管理和一个功能强大的仿真调试器。

单片机的应用除了具备硬件系统外还需配有程序让单片机运行，在前面我们已经了解了单片机（或CPU）能够识别的是机器码（即二进制代码），那就必须要把源程序翻译成机器码。目前用的比较多的就是Keil μVision2 IDE。

Keil公司开发的μVision2 IDE集编辑器、工程管理器和MAKE工具、内嵌DScope51调试器的集成开发环境。Keil μVision2 IDE支持8051MCU的开发，可以用来编译汇编语言源程序、C源程序、连接和重定位目标代码与库文件，生成机器码（HEX文件），调试目标程序等。下面简要介绍该软件的应用，详细可以参考有关书籍。

1.6.1 μVision2 IDE概述

1. 工程管理器

工程是由源文件、编程说明以及开发工具组成。一个工程能够产生一个或者多个目标程序。μVision2包含一个器件数据库（Device Database），器件数据库包含片内存储器与外围设备的信息，可以为片外存储器确定起始地址与规模。

2. 编译器和调试器

μVision2 IDE是基于Windows平台的编译软件，源代码编辑器提供一种便于识别且能快速、方便修改源程序的编辑窗口，为用户提供程序断点设置，可以编写函数语言进行调试，可以观察变量的取值与不同区域的存储区。

C51编译器遵循ANSI标准而且为8051内核进行特殊设计。例如：通过sfr和sbit两个关键字实现对于SFR的读写，大、中、小的存储模式决定变量的存储类型。又如C51可以用来编写中断服务程序。再有C51提供了高效灵活的指针，特殊指针在声明的同时指定了存储类

型，指向某一个特定的存储范围。

1.6.2　μVision2的入门应用

1. 启动μVision2

双击图标，启动μVision2，进入的完整工作界面，如图1-10所示。

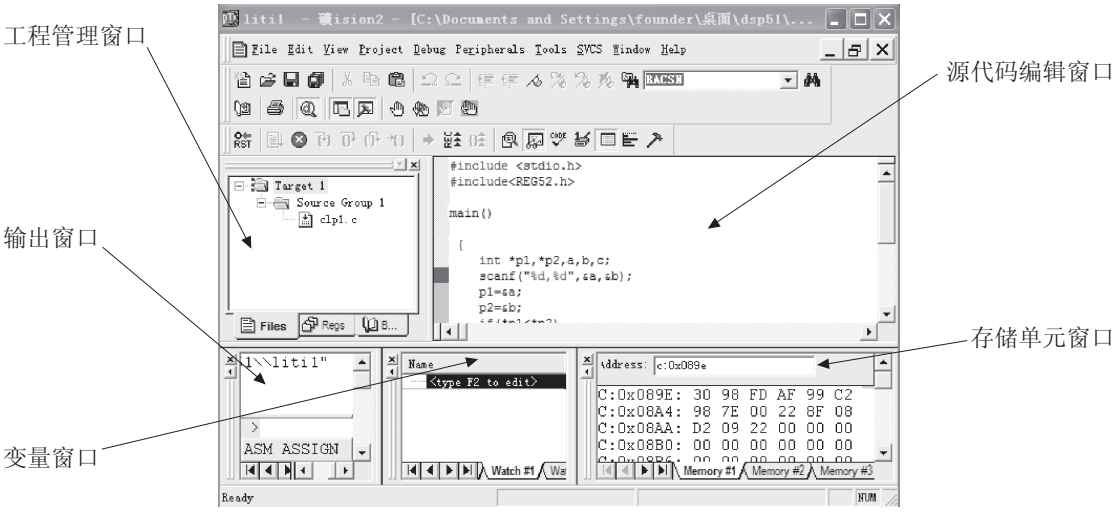


图 1-10　keil μVision2工作界面

2. 创建工程（project）

在界面中单击project/New project，选择存放工程文件的目录。默认的工程文件扩展名为*.uv2。

保存工程文件名后自动弹出单片机型号选择对话框，如图1-11所示，对话框左边是型号选择（依据制造商）这里选择Atmel的AT89C51型，右边显示内容是所选型号的硬件概述。单击确定以后完成工程建立。

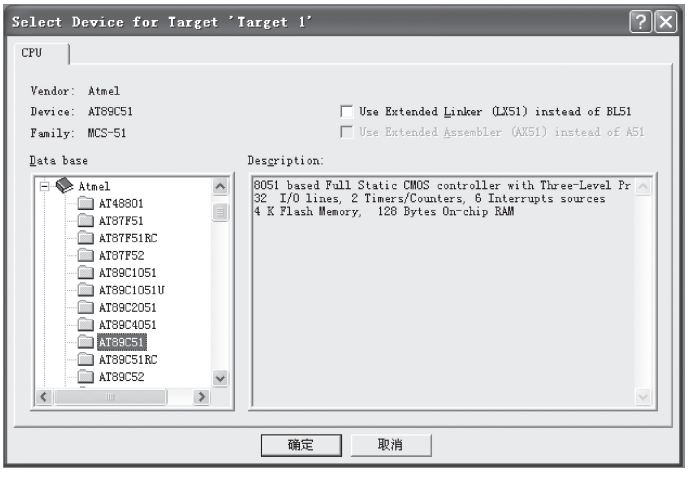


图 1-11　单片机型号选择对话框

3. 源程序文件的创建

单击File/New在弹出的源程序输入框中进行程序输入，完成后进行保存。保存时要注

意若是汇编语言程序扩展名为*.asm（或是*.a51），若为C51写的则保存时扩展名为*.c，一般保存在与工程文件同目录下。

然后在文件栏中右单击Target1展开后的Source Group1项，在弹出的快捷菜单命令中选择Add Files to Group‘Source Group 1’命令，在出现的对话框中按照文件类型选择源程序文件进行工程下的添加，接着双击源程序文件打开源程序文件如图1-12所示。

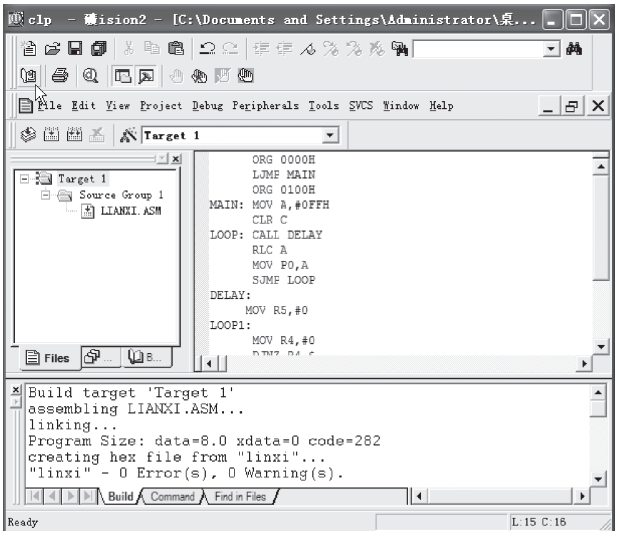


图 1-12 源程序打开以后的对话框

4. 编译与调试

① 生成*.Hex文件设置：右单击文件栏中的Target1，选中块捷菜单中Options for Target‘Target1’命令，在弹出的对话框选择Output选项卡。选中Creat HEX Fi；在Name of Executable：输入生成的十六进制文件名称，如图1-13所示。

② 编译：执行Project/Build Target命令是编译修改过的文件且生成应用；执行Project/ReBuild Target命令重新进行编译且生成应用；也可以单击对应的工具按钮。这是实现对源程序的编译过程，如果有错误则对于源程序进行修改，直到最后编译结果应该是0 Error，对于Warning(S) 则影响不大，当然改完以后应进行保存。如图1-14所示。

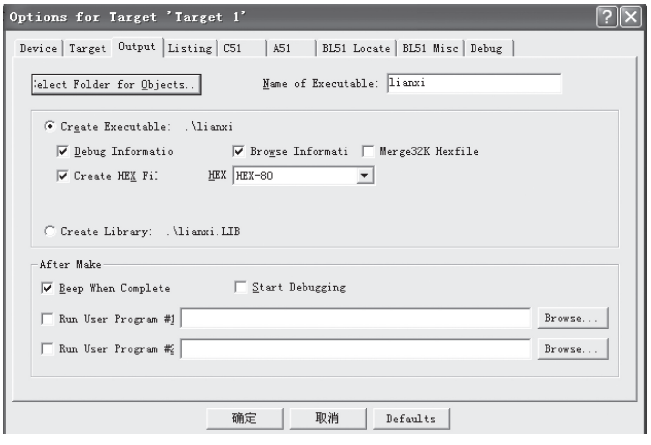


图 1-13 Options for Target‘Target1’设置对话框

调试：执行Debug/ Start/Stop Debug Session 命令，启动调试模式。

① 可以执行菜单命令或快捷键操作

菜单命令：

Go 运行程序，直至结束（若设置断点则到达断点）。

Step 单步依次执行指令，遇子程序则进入子程序。

Step over 单步依次执行指令，遇子程序则跳过子程序。

Step out of Current function 执行到当前函数结束。

Stop running 程序停止运行。

② 调试过程断点设置

Insert/Remove Breakpoint 设置/删除当前断点。

Enable/Disable Breakpoint 使能/取消当前断点。

Kill All Breakpoint 禁止所有断点。

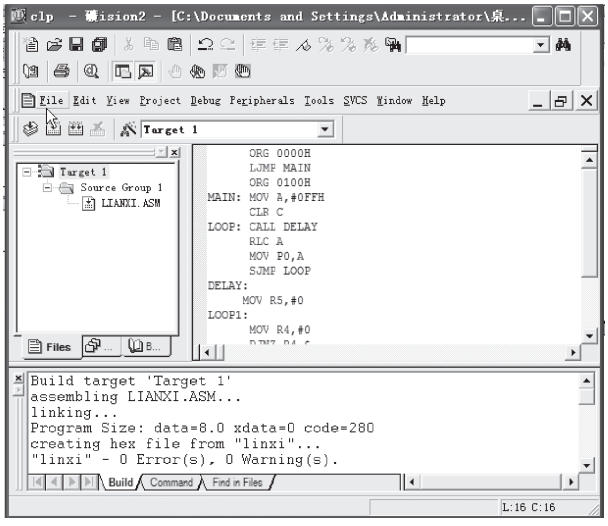


图 1-14 编译结束对话框

③ 调试过程可以打开的窗口

在View栏下的有关命令：

Diassembly Window　显示反汇编窗口

Memory Window　　显示存储器窗口

Serial Window1#　　显示串口窗口

项目窗口中的Register窗口（regs页）

④ 外围部件

在Peripherals栏下的有关命令

Reset CPU　　复位CPU

Interrupt　　中断设置

I/O-Port　　P口设置

Serial　　串口设置

Timer　　定时器设置

⑤ 配置选项

通过右单击Target1，在弹出的快捷菜单中选择Options for Target命令弹出如图1-13对话框所示，除了前面关于选中Hex文件的设置以外还可以进行工作频率、片上ROM、外接ROM、变量存储空间或代码存储空间的设定。

知识拓展

要求通过查阅AT80C2051单片机的资料、并且自己设计一个基于AT80C2051的最小系统。

本章小结

本章以单片机应用的硬件——最小系统为主题，叙述了核心部件单片机的基本定义、型号、基本结构。单片机40个引脚以及每个引脚的功能，单片机的存储器系统结构包括片内RAM、ROM与扩展存储器的最大容量，片内RAM包括工作寄存器、用户RAM区、特殊功能寄存器SFR、位寻址区，端口P0~P3的功能与特点，单片机的运行方式。对于最小系统除了核心部件外，还需要配置时钟电路、复位电路以及电源，$\overline{EA}$引脚的正确连接。最后介绍了单片机中中常用的二进制、十六进制、十进制数以及数制的相互转换，单片机中常用的ASCII、BCD、校验码等代码以及编译软件Keil μVision2 IDE的特点与基本应用。

思考与练习题

一、单项选择题

1. MCS-51单片机的四个并行I/O中，其驱动能力最强的是（　　）。

A. P0口　　B. P1口　　C. P2口　　D. P3口

2. 当晶振频率选用12MHz时，89C51单片机的机器周期是（　　）。

A. 1μs　　B. 1ms　　C. 2μs　　D. 2ms

3. 一字节补码表示的符号数范围是（　　）。

A.－128~＋127　　B.－127~＋128　　C. －128~＋128　　D. －127~＋127

4. MCS-51单片机复位后，堆栈从片内RAM哪单元开始（　　）。

A. 06H　　B. 07H　　C. 08H　　D.30H

5. MCS-51单片机RESET时，PC的内容为（　　）。

A. 0003H　　B. 000BH　　C. 0000H　　D. 0100H

二、判断题

1. 89C51片内RAM中任何一个字节都可以进行位操作。（　　）
2. MCS-51单片机是依靠低电平复位的。（　　）
3. P0口在作输出口使用时，必须外接下拉电阻。（　　）
4. MCS-51单片机的数据存储器与程序存储器是统一编址的。（　　）
5. 程序状态寄存器PSW用于存放运算结果。（　　）

三、计算题

1. 转换下列十进制数为二进制数和十六进制数

0.825、145、12.12、

2. 把下列十六进制无符号数转换成二进制数和十进制数

3DH、0FFH、123H

3. 将下列十进制数用BCD码表示

246　128.79

4. 求取下列机器数的真值

[X1]= 11100001; [X2]= 01100001

5. 写出下列字符的ASCII码

‘Z’、‘z’、CR（回车）、LF（换行）

四、问答题

1. 什么是单片机？
2. 单片机的最小系统由哪些部分组成？
3. 试写出四组工作寄存器对应的RAM地址。如何选择当前工作寄存器组？
4. 试概括DPTR与PC的应用特点。
5. 什么是堆栈，堆栈指针SP的作用是什么？

实验项目1　单片机控制LED灯

一、实验目的

1. 掌握单片机最小系统的电路设计与应用
2. 学会并行I/O对LED灯的开关控制
3. 了解单片机的硬件与程序结合过程和应用

二、实验设备

1. PC计算机一台
2. Keil μVision2集成开发环境
3. 单片机实验装置一套（教材配套）
4. STC_ISP_V480在系统编程软件

三、实验内容

在单片机最小系统基础上对P0口的八条口线，通过八个限流电阻分别连接八个发光二极管，通过指令实现当口线输出为“1”时，发光二极管不亮；当口线输出为“0”时，发光二极管被点亮，利用延时依次电亮LED灯。

实验程序

```
ORG 0000H
LJMP MAIN
ORG 0100H
MAIN:
        MOV A,#0FFH
        CLR C
LOOP: CALL DELAY
        RLC A
        MOV P0,A
        SJMP LOOP
        DELAY:
        MOV R5,#0
LOOP1:
        MOV R4,#0
        DJNZ R4,$
        DJNZ R4,$
        DJNZ R4,$
```

```
DJNZ R4,$
DJNZ R5,LOOP1
RET
END
```

四、实验操作

按照1-6节内容，在指定的文件夹中建立工程文件、建立汇编语言源程序文件，建立完成以后进行编译，修改错误，直至没有错误，编译时同步生成*.Hex文件，然后可以进行设置断点、单步/全速等调试操作。最后应用下载软件STC_ISP_V480下载Hex文件，实际观察运行效果。

第2章 指令系统与程序设计初步

单片机系统的应用是在硬件平台基础上配备程序才能使单片机按照人的设计思想完成针对性的任务，例如：进行温度检测、控制LED等，才能进一步构成所谓的智能化仪器。程序设计的基础是指令，尽管现在比较复杂的系统中使用的是C语言编程，但是对于掌握单片机的工作原理、对于掌握单片机的运行特征，对于更好地用C语言编写程序，学习一套CPU的汇编指令系统还是大有益处的。

2.1 指令系统概述与寻址方式

学习目标

1. 理解指令、程序、编程的基本概念。
2. 熟练掌握89C51的寻址方式。
3. 熟悉指令功能、分类与指令运用。

导　入

了解指令中的基本概念对于指令的理解与编写程序有很大的帮助。寻址方式概括了指令系统中全部指令的操作数读写、通过地址寻找操作数的基本方法，理解了寻址方式对于指令的掌握就顺理成章了。

2.1.1 指令系统概述

指令系统是程序设计的基础，任何一个型号的CPU都有自身能识别、运行的指令系统，不同型号的CPU其指令系统也不同，MCS-51也有自己的一套指令系统。若想掌握单片机的应用，除了理解硬件系统外，还必须掌握指令系统。只有灵活使用指令编写出相关的程序，才算是真正掌握了单片机的应用。

对于学习一套指令系统，首先需要了解几个基本概念和术语。

1. 基本术语

指令：让CPU执行某种操作的命令。

指令系统：CPU所能执行的全部指令的集合，CPU不同则指令系统也不同。

机器语言：用二进制编码表示的CPU能直接识别、执行的机器指令的集合。

汇编指令：为了便于人书写、记忆、应用，使用英文单词及其缩略形式的助记符、数字等组合来表达的指令。

汇编程序：用汇编指令编写的程序，汇编指令与汇编程序统称为汇编语言。

程序：指为完成一个任务、一个算法的指令的有序集合。

程序设计：针对某一个对象、任务而编写程序的过程。

机器语言与汇编语言都归属于低级语言，完全依赖于硬件，是面向机器的语言。低级语言的学习比较枯燥，但是学习与应用一套指令系统对于掌握CPU的工作原理，转向以C语言（高级语言）的应用以及为嵌入式微控制器系统的应用奠定一个良好的基础。

2. 指令格式

89C51指令格式如下：

[标号：] 操作码 [目的操作数][，源操作数] [；注释]

① 标号是指令地址符号，由l～8个ASCII字符组成，以字母开头的字母、数字串。与操作码之间用冒号分隔，是指令的机器代码在存储单元中的首地址。

② 操作码用英文字母表示，定义了指令执行功能，这些英文字母的组合称之为助记符，用空格与目的操作数分割。

③ 操作数指参加操作的数据或者是数据的地址。操作数可以有三个、两个、一个或者无操作数，操作数分左边目的操作数（表示操作结果存放的单元地址）和右边的源操作数（表示数据来源），操作码与操作数之间必须用空格分隔，操作数之间用逗号分割。指令中的数据一般用十进制、十六进制、二进制和字符串。

④ 注释是为指令或程序段作说明，用分号开始，可以换行书写，但换行时应注意必须以分号开头。注释是为了方便阅读、交流。

⑤ 方括号项称为可选项。操作码是主要内容，所以无方括号。

3. 指令中有关符号说明

在89C51汇编指令系统中约定了一些常用符号，见表2-1。

表 2-1 常用符号说明表

符　号	意　　义
Rn（n=0~7）	当前选中的工作寄存器组中的八个寄存器R0~R7。
Ri（i=0，1）	当前选中的寄存器区中的允许作为地址指针的寄存器R0或R1。
#data	#标识立即数，表示8位立即数。

（续表）

符 号	意 义
#data16	表示16位立即数。
direct	表示8位片内RAM单元（包含SFR）的直接地址。
addr16	表示16位目的地址。
addr11	表示11位目的地址。
rel	用补码形式的8位地址偏移量。
bit	片内RAM或SFR的直接寻址的位地址。
@	表示寄存器的间址寻址符号。
/	表示对该位先取反再操作，不影响该位的原值。
(X)	X中的内容
（（X））	X为地址单元中的内容
←	将箭头右边的内容送入箭头左边的单元内
§	当前指令所在地址

4. 指令系统分类

89C51 共有指令111条。若按照指令的字节数分类，可分为：

① 单字节指令49条，所谓单字节指令，8位二进制代码中包含操作码和操作数。

例如：INC DPTR; 机器代码A3H。

② 双字节指令45条，一个字节表示操作码，另一字节表示操作数。

例如：MOV A, #12H 机器代码 74H12H。

③ 3字节指令17条，一个字节操作码，两个字节操作数。

其中：操作数可以是数据也可以是操作数所在的地址。

若按照功能分类，可分为：

① 数据传送指令28条；

② 算术运算指令24条；

③ 逻辑运算与移位指令25条；

④ 控制转移指令17条；

⑤ 位操作指令与布尔操作17条。

2.1.2 89C51寻址方式

单片机中存放数据的存储空间共有三类：片内数据RAM（与特殊功能寄存器SFR统一编址），外部数据RAM和程序存储器均为独立编址。所谓寻址方式，就是对于不同存储器中的数据提供不完全相同的寻找操作数或者是给出操作数所在地址。

【例2-1】 这仅仅是用来进行传送数据的程序段。

```
MAIN:  MOV    R0, #2AH
       MOV    A, R0
       MOV    R1, A
       MOV    A, 2AH
       MOV    A, PSW
```

```
        MOV     DPTR, #1234H
        MOV     A, @R0
        MOV     R2, A
        MOVX    A, @DPTR
        MOV     C, PSW.2
LOOP:   SJMP    LOOP
```

阅读【例2-1】，通过该程序段指令的分析就基本可以浏览89C51的几种寻址方式。

1. 立即数寻址方式

例如：MOV R0, #2AH ;　　R0　　←——2AH

　　MOV DPTR, #1234H ; DPTR ←——1234H

源操作数以#开始，称其为立即数。立即数寻址方式的特征就是1字节或者2字节的源操作数在指令操作码后面，通过分号后面的注释可以明确运行结果是R0中的内容为2AH；DPTR中的内容为1234H。由于操作数是存放在程序存储器ROM中，寻址空间为程序存储器。立即数寻址方式见图2-1（a）。

2. 直接寻址方式

例如：MOV A, 2AH;

因为在指令中直接给出操作数所在存储单元的地址2AH，所以称为直接寻址方式。由于该地址只能是8位，所以寻址空间限制在片内RAM和SFR之中（SFR可以直接用名称）。设在2AH中的内容是0FFH，那么运行结果是（A）=0FFH。直接寻址方式与立即数寻址表示方式的区别在于2A前的#号。直接寻址方式见图2-1（b）。

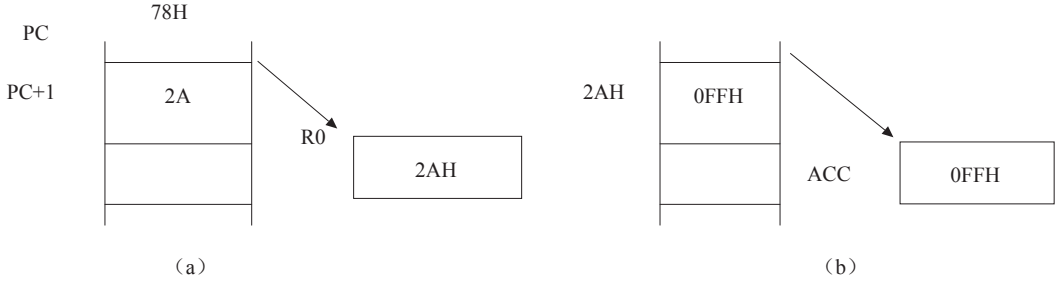


图 2-1　立即数寻址方式与直接寻址方式对照图

（a）直接寻址方式　（b）立即数寻址方式

3. 寄存器寻址

例如：MOV A, R0 ; A←（R0）

　　MOV R1, A ; R1←（A）

寄存器寻址方式就是指源操作数就放在寄存器中，寄存器可以是R0~R7或者是A，B，DPTR。若（R0）=0AAH，则在R1中观察到的也是0AAH。

由此可见，寻址范围是当前工作寄存器组中8个寄存器，或者部分特殊功能寄存器。

4. 寄存器间接寻址

例如：MOV A, @R0 ; A←（（R0））

对比寄存器寻址，本条指令在R0前面多了一个符号@，按照表2-1说明这时R0里面存放的是操作数地址，通过地址间接地找到操作数，所以称为寄存器间接寻址。寄存器间接寻址的范围有多少以及如何正确寻址呢？

① 内部RAM低128个单元，规定用@Ri（i=0，1）作为间接寻址寄存器。

② 片外数据存储器的64KB寻址空间，用@DPTR作间接寻址寄存器。适用的指令MOVX A, @DPTR。

③ 片外数据存储器低256字节，除可使用DPTR作为间址寄存器外，也可使用Ri作寄存器，MOVX A，@Ri，此时P2口为00H。

④ 堆栈区：堆栈操作指令PUSH（压栈）和POP（出栈），使用堆栈指针(SP)作间址寄存器对堆栈区的间接寻址。

5. 变址寻址

变址寻址方式的指令只有三条：

```
MOVC    A, @A+ DPTR;        读程序存储器指令
MOVC    A, @A+PC;           读程序存储器指令
JMP     @A+DPTR;            无条件转移指令
```

针对MCS-51寄存器中没有设计变址寄存器，而采用了一种“基址+变址间接寻址=变址寻址”的方式，简称为变址寻址。在变址寻址方式中，确定以DPTR或PC作基（准）地址寄存器，由于DPTR或PC是16位的，所以作为基地址寄存器可以访问64K存储空间；累加器A前面冠以@作为一个间接寻址寄存器，效果相当于基址寄存器的偏移量，两者内容相加形成的16位地址作为所取的操作数地址，达到变址的效果。变址寻址方式只能对程序存储器，寻址范围可达到64KB。该寻址方式通常用于访问程序存储器中的数据表格。

6. 相对寻址

例如：SJMP LOOP ;

相对寻址是指在一个基准点上下偏移（rel），rel经过编译转换成一个确定的LOOP地址，基准点是用PC执行完本条指令的当前值，本条指令是2字节，所以PC的当前值是本条指令地址＋2为当前值。由于rel值是用二进制补码表示的值，范围为－128~＋127，而形成实际的转移地址是PC＋2＋rel。所以相加后转移的目标地址可以向前小于PC值（－128），也可以向后大于PC值（＋127）。寻址空间为程序存储器。

7. 位寻址

例如：MOV C, 00H ;

由于在片内存储器中有128个位和11个SFR中的位存在，访问位区域的操作就需配有位寻址方式。位寻址方式中用C充当布尔累加器。寻址空间为片内RAM中的位区域和SFR中的部分位。SFR中的部分位的表示方式可以用对应位地址，也可以用类似PSW.2形式。

2.2 指令系统

学习目标

1. 熟悉指令的几种分类。
2. 通过阅读掌握指令的功能。
3. 在编程过程中正确选用指令。

导　入

指令就是命令，指令系统就是全部指令的集合。数据在寄存器与存储器之间流动，要进行加减乘除运算与逻辑判断；程序运行的流向控制，相关的位操作等MCS-51提供有一整套的指令。运用好指令系统便能使单片机的功能十分强大。

2.2.1 数据传送类指令

数据传送指的是把数据从源地址中传送到目的地址中去，而源地址中的数据保持不变。从寻址方式中已经基本了解源操作数可以在寄存器中、存储器中，也可以是立即数；目的地址可以是寄存器、存储单元。存储单元的寻址可以用直接方式或是间接寻址方式包括变址寻址。数据传送类指令运行结果的一个很重要的特点是不影响PSW中的AC、CY、OV，但是有时对于P位可能会有影响。

指令助记符有MOV、MOVC、MOVX、XCH、XCHD、SWAP等，共有29条指令。指令与操作、指令字节数、机器周期参见表2-2。由于存储器的编址方式导致传送指令的分类如下：

1. 内部RAM 的数据传送指令

① 以累加器A为目的操作数的指令

② 以寄存器Rn为目的操作数的指令

③ 以直接地址为目的操作数的指令

④ 以寄存器间接地址为目的操作数的指令

⑤ 16位数据传送指令

⑥ 交换指令（前3条为字节交换指令，后2条为高4位与低4位互换）

⑦ 栈操作指令

2. 片外RAM与累加器A进行数据传送指令

3. 查表指令（程序存储器数据传送）

表 2-2 数据传送类指令

分 类	指令助记符	操 作	字节数	周期
以累加器A为目的操作数的指令	MOV A,Rn	A←(Rn)	1	1
	MOV A,direct	A←(direct)	2	1
	MOV A,@Ri	A←((Ri))	1	1
	MOV A,#data	A←#data	2	1
以寄存器Rn为目的操作数的指令	MOV Rn,A	Rn←(A)	2	1
	MOV Rn,direct	Rn←(direct)	2	2
	MOV Rn,#data	Rn←#data	2	1
以直接地址为目的操作数的指令	MOV direct,A	direct←(A)	2	1
	MOV direct,Rn	direct←(Rn)	2	2
	MOV direct,direct	direct←(direct)	3	2
	MOV direct,@Ri	direct←((Ri))	2	2
	MOVdirect,#data	direct←#data	3	2
以寄存器间接地址为目的操作数的指令	MOV @Ri,A	(Ri)←(A)	1	1
	MOV @Ri,direct	(Ri)←(direct)	2	2
	MOV @Ri,#data	(Ri)←#data	2	1
16位数据传送指令	MOV DPTR,#data	DPTR←#data16	3	2
片外RAM与累加器A进行数据传送指令	MOVX A,@Ri	A←((Ri))	1	2
	MOVX A,@DPTR	A←((DPTR))	1	2
	MOVX @Ri,A	(Ri)←(A)	1	2
	MOVX @DPTR,A	(DPTR)← (A)	1	2
查表指令	MOVC A,@A+PC	PC←(PC)+1 A←((A)+(PC))	1	2
	MOVC A,@A+DPTR	PC←(PC)+1 A←((A)+(DPTR))	1	2
交换指令（前3条为字节交换，后2条为高4位与低4位互换）	XCH A,Rn	(A)与(Rn)互换	1	1
	XCH A,direct	(A)与(direct)互换	2	1
	XCH A,@Ri	(A)与((Ri))互换	1	1
	XCHD A,@Ri	$(A_{0\sim3})$互换$((Ri)_{0\sim3})$	1	1
	SWAP A	$(A_{0\sim3})$与$(A_{4\sim7})$互换	1	1
栈操作指令	PUSH direct	SP←(SP)+1， (SP)←(direct)	2	2
	POP direct	((SP))→(direct) SP←(SP)-1	2	2

【例2-2】 有关查表、外部存储器访问指令的编程

① 在外部ROM的1000H单元开始存有八个采样数据依次读出八个数据的程序段

```
START:    MOV   DPTR, #1000H
```

```
        MOV   R4, #08H
        MOV   A,  #0
LOOP:   MOVC A,@A+DPTR
        ......
        INC  A
        DJNZ  R4, LOOP; 循环控制
```

② 已知（50H）=55H，（60H）=66H，利用堆栈功能实现数据交换

```
MOV  SP，#40H   ; SP初值设定
PUSH  50H       ;（SP）=（SP）+1; 41H← 55H
PUSH  60H       ;（SP）=（SP）+1; 42H← 66H
POP  50H        ; 50H←66H;（SP）=（SP）-1;
POP  60H        ; 60H←55H;（SP）=（SP）-1;
```

说明：上电复位时（SP）=07H，可以在程序开始重新设定初值，中间不宜操作。89C51属于向上生长堆栈，入栈时，先指针操作，后数据操作；出栈时相反。

2.2.2 算术运算类指令

算术运算指令是用于加法、减法、乘法和除法的运算以及十进制调整。指令助记符有ADD、ADDC、SUBB、INC、DEC、DA、MUL、DIV等，具体见表2-3。

对于基本四则运算，如加法有加数与被加数；乘法有乘数与被乘数，需要有两个数据才能完成一个操作。对于加1和减1，十进制调整只需要一个操作数。

对于加法、减法（含带进位/借位）一个操作数放在累加器A中，第二个操作数可以是立即数、直接地址、寄存器及寄存器间接寻址方式；对于乘法、除法运算一个操作数放在累加器A中，第二个操作数只借助于B寄存器进行。

算术运算指令运行后，除了INC、DEC指令，会影响PSW中的进位位CY、半进位位AC或者溢出位OV。算术运算指令共有24条，其中可以划分为：

① 无进位的加法指令
② 有进位的加法指令
③ 加1指令
④ 带借位的减法指令
⑤ 减1指令
⑥ 乘法指令
⑦ 除法指令
⑧ 十进制调整指令

需要特别说明：由于CPU的算术/逻辑运算部件只能执行无符号数的运算，实际编程过程中，借助于OV溢出标记，可以进行二进制补码运算；借助于PSW中CY位可以扩展多字节的加法、减法运算。对于乘法、除法运算的两个数，一个数在累加器A中，第二个数则放在寄存器B中；一个单字节数乘以另外一个单字节数其积为两个字节数，所以运算完成以后乘法注意高位字节在寄存器B中，低位字节在A中；对于除法完成后整数的商在A中，余数在寄存器B中。

表 2-3 算术运算类指令

分　类	指令助记符	操　作	字节数	周期
无进位的加法指令	ADD A, Rn	A←(A)+(Rn)	1	1
	ADD A,direct	A←(A)+(direct)	2	1
	ADD A,@Ri	A←(A)+((Ri))	1	1
	ADD A,#data	A←(A)+#data	2	1
有进位的加法指令	ADDC A,Rn	A←(A)+CY+(Rn)	1	1
	ADDC A,direct	A←(A)+CY+(direct)	2	1
	ADDC A,@Ri	A←(A)+CY+((Ri))	1	1
	ADDC A,#data	A←(A)+CY+#data	2	1
带借位的减法指令	SUBB A,Rn	A←(A)-CY-(Rn)	1	1
	SUBB A, @Ri	A←(A)-CY-((Ri))	1	1
	SUBB A, direct	A←(A)-CY-(direct)	2	1
	SUBB A,#data	A←(A)-CY-#data	2	1
加1指令	INC A	A←(A)+1	1	1
	INC Rn	Rn←(Rn)+1	1	1
	INC @Ri	(Ri)←((Ri))+1	1	1
	INC direct	direct←(direct)+1	2	1
	INC DPTR	DPTR←(DPTR)+1	1	2
减1指令	DEC A	A←(A)-1	1	1
	DEC Rn	Rn←(Rn)-1	1	1
	DEC @Ri	(Ri)←((Ri))-1	1	1
	DEC direct	Direct←(direct)-1	2	1
乘法指令	MUL AB	$A_{0\sim7}$和$B_{8\sim15}$←(A)＊(B)	1	4
除法指令	DIV AB	A(商)和B(余数)←(A)／(B)	1	4
十进制调整指令	DA A	把A内容调整为BCD数	1	1

【例2-3】（1）设（A）= 11011100，（R3）=01010100 执行两数相加指令后，分析对PSW有关位的影响。

```
    1101 1100
  + 0101 0100
  ------------
  1 0011 0000
```

说明：两数执行相加后，最高位有进位则CY=1；位3向位4有进位则辅助进位位AC=1；结果中含有偶数个1则P=0；溢出位OV判断依据为OV=D_{6CY} D_{7CY}，其中D_{6CY}是位6向位7进位，D_{7CY}是位7向CY的进位，按照两位都产生进位或者两位都不产生进位则OV=0；一个有一个没有进位则OV=1。现位6向位7有进位，位7向CY有进位则OV=0。

若加法执行前是无符号数（A）=220D（R3）=84D，两数执行相加以后和是（304）D；最高位（CY）=1是有数值意义的。

若加法执行前是有符号数（A）=－36D（R2）=84D，两数执行相加以后和是48D；OV=0表示结果是正确的，因为补码一字节数值范围是－128~＋127。（CY）=1是无意义的。

两个数相加是视作有符号还是无符号，应依据实际情况在相加之前确定，而OV只有在带符号数运算时才有意义。

（2） 对于上述例子写出执行语句：

```
MOV  R3,  #220;
MOV  A,   #84;
ADD  A,   R3;
```

执行结果除了上述分析的PSW标志位外，（A）= 30H。

（3） 若是进行多字节加法，接着需要考虑是否用到ADDC指令；因为本题运算结果是CY=1，对于高字节就要考虑低字节向高字节的进位CY了。

本题讨论形式可以用在进行减法的操作，但是需要注意的是在89C51中只存在一条减法指令SUBB；即带借位的减法指令，这时就要考虑是连带CY的相减。在实际编程时若不考虑CY位可在两数减法前先对CY清0。

例如：
```
CLR   C       ;  CY←0
SUBB  A, R2   ;  （A）←（A）－（R2）－（CY）
```

【例2-4】 十进制调整用法

执行语句：
```
ADDC  A, R3; 运算结果存于A
DA    A     ; 对A内容进行十进制数调整
```

说明：本例说明了DA指令在应用中的位置。用4位二进制数表示1位十进制数0～9，称为BCD码，一个字节的二进制数表示两个十进制数，称为压缩BCD码。但是指令系统只给出二进制数加法指令，对于和数中超过9的必须要调整掉，例如：对于数码管显示就要十进制数，这种调整称为十进制数调整，其方法就是在加法指令后面紧跟DA指令。

调整机理：若A结果中的低4位大于9或者是3位向4位有进位（AC=1），则低4位加6调整。若A结果中的高4位大于9或者是有进位（CY=1），则高4位加6调整。

这种调整只要书写了DA指令，CPU会依据PSW与数值自动进行上述修正，产生正确的BCD码。

例如：设：（A）=0110 1000B，（R3）= 0101 1000B

```
   0110 1000    BCD 68
 + 0101 1000    BCD 58
-------------
   1100 0000
 + 0110 0110
-------------
 1 0010 0110    BCD 1 26
```

说明：上式中虽然低4位之和小于9，但是AC=1则加6进行调整；高4位大于9同样加6进行调整。调整后CY =1，则表示百位数为1所以最后值是126。

【例2-5】 两个无符号数相乘，无符号数分别存放在40H和41H单元，积高8位存放在43H，低8位存放在42H单元的程序段。

```
MOV R2, #40H ; 第一个乘数地址
MOV  A, @R2  ; 第一个乘数送A
INC R2       ; 修改地址值
MOV  B, @R2  ; 第一个乘数送B
MUL  AB      ; 乘法运算
```

```
INC R2
MOV @R2,A  ; 积的低8位送42H
INC R2
MOV @R2,B  ; 积的高8位送43H
```

2.2.3 逻辑运算类指令

逻辑操作指令包括与、或、异或、清除、求反和移位等操作，具体见表2-4。逻辑运算指令主要是按位进行与、或、异或、移位等操作。运算结果一般不影响PSW，只有当目的操作数为A累加器时，对于奇偶标志P位形成影响和带进位位CY移位时对于CY的影响。有一种情况需要说明的是在若目的操作数是I/O口，则完成的是“读—修改—写”操作。助记符有ANL、ORL、XRL、CLR、CPL、RL、RLC、RR、RRC。逻辑运算指令共有24条，可以划分为：

① 逻辑“与”指令
② 逻辑“或”指令
③ 逻辑“异或”指令
④ 累加器清零与取反指令
⑤ 累加器移位指令

表 2-4 逻辑运算类指令

分 类	指令助记符	操 作	字节数	周期
逻辑“与”指令	ANL A,Rn	A←(A)^(Rn)	1	1
	ANL A,direct	A←(A)^(direct)	2	1
	ANL A,@Ri	A←(A)^((Ri))	1	1
	ANL A,#data	A←(A)^#data	2	1
	ANL direct,A	direct←(direct)^(A)	2	1
	ANL direct,#data	direct←(direct)^#data	3	2
逻辑“或”指令	ORL A,Rn	A←(A)∨(Rn)	1	1
	ORL A,direct	A←(A)∨(direct)	2	1
	ORL A,@Ri	A←(A)∨((Ri))	1	1
	ORL A,#data	A←(A)∨#data	2	1
	ORL direct,A	direct←(direct)∨(A)	2	1
	ORL direct,#data	direct←(direct)∨#data	3	2
逻辑“异或”指令	XRL A,Rn	A←(A) ⊕ (Rn)	1	1
	XRL A,direct	A←(A) ⊕ (direct)	2	1
	XRL A,@Ri	A←(A) ⊕ ((Ri))	1	1
	XRL A,#data	A←(A) ⊕ #d ata	2	1
	XRL direct,A	direct←(direct) ⊕ (A)	2	1
	XRL direct,#data	direct←(direct) ⊕ #data	3	2

（续表）

分　类	指令助记符	操　作	字节数	周期
累加器清零与取反指令	CLR A	A←0	1	1
	CPL A	A←($\overline{A}$)	1	1
累加器移位指令	RL A	A的内容按位循环左移	1	1
	RLC A	A的内容按位带进进位CY循环左移	1	1
	RR A	A的内容按位循环右移	1	1
	RRC A	A的内容按位带进进位CY循环右移	1	1

【例2-5】设（30H）=2CH，若需要保留低4位不变，高4位清0；第7位置1；最后全部清0的操作，则执行操作如下：

```
MOV  A, 30H ;
ANL  A, #0FH ;        （A）的内容与立即数0FH相与
ORL  A, #80H ;        （A）的内容与立即数80H相或
MOV  R2, A   ;
XRL  A, R2   ;        （A）的内容与（R2）相异或
```

89C51系统中的逻辑操作指令在实际运用中常用作对于某些位的置位和清0（屏蔽）等操作。

【例2-6】移位指令应用程序段

```
MOV  A,#03H ;
CLR  C      ;
RL   A      ;        （A）×2；低位补0
MOV  R2, A  ;
CLR  C
RL   A      ;        （A）×4；低位补0
ADD  A, R2  ; 总的实现×6
```

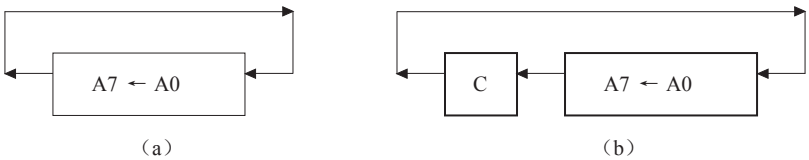


图 2-2 移位示意图

（a）循环左移 （b）带进位位循环左移

说明：图2-2是循环左移与带进位位循环左移图，若是循环右移与带进位位循环右移则移位方向均相反。在例2-6中执行了循环左移指令由于在低位补的0，左移就相当于实现了乘以2功能。

2.2.4 控制转移类指令

通过控制转移指令的执行，改变程序计数器PC的当前值，使程序运行改变方向，所谓程序运行改变方向即原先是按顺序先后依次执行，当执行控制转移指令后不再沿原先方向

执行而是使程序按照需要或者是条件进行转向。控制程序转移指令共有十七条。指令的助记符有AJMP、LJMP、SJMP、JMP、JZ、JNZ、CJNE、DJNZ、ACALL、LCALL、RET、RETI和NOP，其指令见表2-5。

1. 无条件转移指令

所谓无条件转移指令是当程序运行到该条指令时，不需要任何附加条件就转移到指令提供的地址处继续运行。无条件转移指令有：

① 绝对转移指令

② 长转移指令

③ 相对转移指令

④ 变址寻址转移指令

转移目标地址范围在64K程序存储器地址空间内。

例如：ORG　0100H

　　AJMP　MAIN

　　……

MAIN：……

当程序执行到AJMP　MAIN指令时，先（PC）+2→PC表示这时PC已经读取本条指令，AJMP是两字节，然后把MAIN这个标号地址，送入PC的低11位（PC是16位，高5位不变），形成一个新的16位的转移目标地址。11位地址所覆盖的地址范围为2K，就是MAIN不能远离0100H地址2K范围。

从本条指令执行功能理解出发，结合寻址方式，则对于LJMP指令、SJMP指令以及JMP指令执行功能都可以得到正确的理解。

【例2-7】 通过A值的变化，分别转向不同的程序处理，即当A=0，转向K0；A=1，转向K1；A=7转向K7。称为程序散转。

```
        MOV     A, #DATA ;#DATA ;      分别是分支值0~7。
        MOV     R5, A
        RL      A ;                    分支值乘2操作
        ADD     A , R5 ;               分支值乘3操作
        MOV     DPTR, #TABLE
        JMP     @A+DPTR ;              变址寻址转移指令, #TABLE为基准地址
        ……
TABLE:  LJMP  K0 ; 3字节长转移指令, 转向标号（地址）为K0处理程序
        LJMP  K1 ; 3字节长转移指令, 转向标号（地址）为K1处理程序
        ……
        LJMP  K7 ; 3字节长转移指令, 转向标号（地址）为K7处理程序
```

2. 条件转移指令

表示转移时需要满足某种前提条件，分别有：

① 累加器为0条件转移指令

② 比较条件转移指令

③ 循环控制转移指令

【例2-8】 INC　　R0　　;

```
        CJNE    R0, #80H,  NEXT3 ;
```

当执行CJNE指令时，把R0（目的操作数）的内容与立即数80H（源操作数）进行比较，结果往往有两种情况：相等或者是不相等。若相等则继续本条指令以后的指令，若比较后不相等则进行转移，转移的目标地址是（PC）＋3＋NEXT3→PC。

当比较结果是不相等时，是否能进一步比较目的操作数和源操作数的大小呢，借助于CY是可以的，由于CPU在运行本条指令时是把目标操作数内容与源操作数进行的是差运算，但是与真正的减法指令不同的是执行CJNE指令的减法操作不会改变原来的值，只是影响CY位（两数比较是做减法）。若（R0）>#80H，则CY←0；若（R0）<#80H，则CY←1，这样运行本条指令结果依据条件形成了三个分支，即（R0）分别可以是#80H、小于#80H或者是大于#80H，转移指令在程序设计过程中会经常使用。

【例2-9】循环控制转移指令的程序结构

```
        MOV R2, #10H
DLY1：  ……
        ……
        DJNZ R2, DLY1
        ……
```

当执行DJNZ R2，DLY1指令时，先（R2）←（R2）－1；若（R3）≠0则PC←（PC）＋2＋DLY1转移到标号地址DLY1；若（R3）=0则PC←（PC）＋2顺序执行。

3. 子程序调用指令和返回指令

① 短调用指令

② 长调用指令

③ 从子程序返回指令

④ 从中断返回指令

在实际任务处理中，对于在不同的程序中或者是在一个程序的不同位置上都需要反复进行同一个运算或者是控制、转换操作，那就需要把这个运算或者是控制、转换操作单独编写成一个程序段，该程序段称之为子程序。子程序的运行是需要被调用的，调用过程称为子程序调用。当然子程序还可以调用子程序，调用子程序的可以视作为主调者，主调者可以是主程序，也可以是一个子程序，子程序调用与返回过程见图2-3。

【例2-10】调用子程序结构

```
MAIN：  ……
        LCALL   DISP ;    调用子程序DISP
        ……
DISP：  MOV DPTR, #TAB; 子程序DISP
        ……
        RET ;             子程序结束
```

子程序标号通常与子程序内容有关，这样便于阅读。

进入子程序前要考虑一旦进入子程序需要有一个断点保护，表2-5是CPU自动进行的控制转移类指令。但是，除了断点保护外具体在调用过程中还涉及对传入的数据和子程序返回的数据；以及子程序外用到的包括工作寄存器、PSW数据等都需要考虑进行保护，不能因为进入子程序后遭到破坏而导致返回后不能恢复原来的运行环境。所以要充分利用堆栈进行保护，在运行子程序结束后弹出保护内容。调用子程序与中断子程序的返回指令的主

要区别在于RETI有一个自动开放中断逻辑的工作。

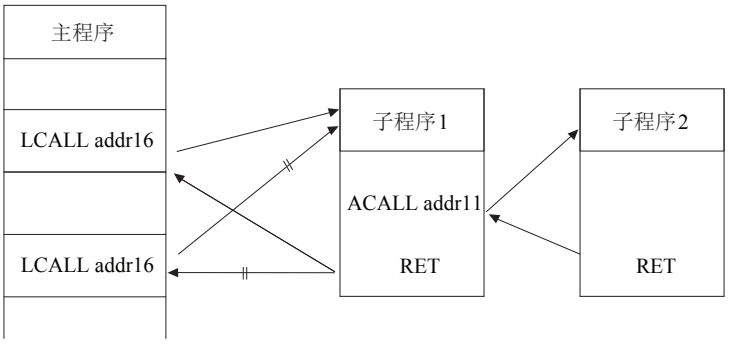


图 2-3 子程序调用与返回过程

4. 空操作指令

指CPU不进行操作，就是消耗一个机器周期时间。

表 2-5 控制转移类指令

分　类		指令助记符	操　作	字节数	周期
无条件转移指令	绝对转移指令	AJMP addr11	PC←(PC)+2 $PC_{10\text{-}0}$←addr11	2	2
	长转移指令	LJMP addr16	PC ← addr16	3	2
	相对转移指令	SJMP rel	PC ←(PC)+2+rel	2	2
	变址寻址转移指令	JMP @A+DPTR	PC←(A)+(DPTR)	1	2
条件转移指令	累加器为0条件转移指令	JZ rel	A=0,PC ←(PC)+2+rel	2	2
		JNZ rel	A≠0,PC←(PC)+2+rel	2	2
	比较条件转移指令	*CJNE A, #data,rel	(A)≠data, (PC)←(PC)+3+rel	3	2
		CJNE A, direct,rel	(A)≠(direct), (PC)←(PC)+3+rel	3	2
		CJNE Rn,#data,rel	(Rn)≠data, (PC)←(PC)+3+rel	3	2
		CJNE@Ri,#data,rel	((Ri))≠data, (PC)←(PC)+3+rel	3	2
	循环控制转移指令	DJNZ Rn,rel	PC←(PC)+2,Rn←(Rn)-1 (Rn) ≠0, PC←(PC) +rel; (Rn) =0则结束	2	2
		DJNZ direct,rel	PC←(PC)+2,direct←(direct)-1 (direct)≠0, PC←(PC) +rel (direct) =0则结束	3	2

（续表）

分　类		指令助记符	操　作	字节数	周期
子程序调用及返回指令	短调用指令	ACALL addr11	PC←(PC)+2；SP←(SP)+1 (SP)←($PC_{7\sim0}$)， SP←(SP)+1 (SP)←($PC_{15\sim8}$)， $PC_{10\sim0}$←add10~0	2	2
	长调用指令	LCALL addr16	PC←(PC)+3；SP←(SP)+1 (SP)←($PC_{7\sim0}$)， SP←(SP)+1 (SP)←($PC_{15\sim8}$)， $PC_{16\sim0}$←add16～0	3	2
子程序调用及返回指令	子程序返回指令	RET	$PC_{15\sim8}$←((SP))， SP←(SP)-1， $PC_{7\sim0}$←((SP)) SP←(SP)-1	1	2
	中断子程序返回指令	RETI	$PC_{15\sim8}$←((SP))， SP←(SP)-1， $PC_{7\sim0}$←((SP)) SP←(SP)-1 开放中断逻辑	1	2
空操作指令		NOP	空操作	1	1

*：对于 CJNE A, #data,rel 指令，完整的操作过程是(PC)←(PC)+3 ；若(A)< data， (PC)←(PC) +rel，且0送CY；若(A)> data，(PC)←(PC) +rel，且1送CY；若(A)= data顺序执行，且0送CY。其他几条指令均如此，只是把源操作数改变一下。

2.2.5 位操作类指令

89C51有一个位处理器，就是PSW中的CY作为位处理器的累加器C。内存中也有位区域和SFR中也有可寻址的位。位操作指令就是对位的传送、位逻辑运算、位控制转移等操作的指令。助记符有 MOV、CLR、CPL、SETB、ANL、ORL、JNC、JC、JNB、JB、JBC。位操作指令、字节数与周期数见表2-6，位操作类共有指令十七条。

对于理解位操作指令完全可以借鉴对字节的操作，字节（8位）操作应用的累加器是A，而位（1位）操作的位累加器为C。

1. 位数据传送指令

把源操作数指定位变量传送到目的操作数指定的变量中，其中一个是累加器C，另一个可以是直接寻址位，结果不影响标志位。

2. 位置1和清0、取反指令

直接对于C位与直接寻址位进行清0、置位、取反操作，结果不影响标志位。

3. 位逻辑运算指令

位逻辑运算指令是把C位与直接寻址位进行逻辑与、或操作结果存入C中，有的是先取

反后操作，但是不改变原先内容。

4. 位条件转移指令

用C内容和直接寻址位内容作为转移的判别条件，一般rel给出的是一个标号地址。

【例2-11】 判断P1口的P1.0位，若为1把60H单元内容送P2口；否则把P1口内容送70H单元。

```
JB   P1.0, ab1  ; 若P1.0位=1, 则转至ab1
MOV 70H, P1  ; 若P1.0位=0, （70H）←（P1）
……
ab1: MOV  P2, 60H  ; （P2）←（60H）
```

表 2-6 位操作类指令表

分　类	指　令	操　作	字节数	周 期
位数据传送指令	MOV C,bit	C←bit	2	1
	MOV bit,C	bit←C	2	2
位置1和清0、取反指令	CLR C	C←0	1	1
	CLR bit	bit←0	2	1
	SETB C	C←1	1	1
	SETB bit	bit←1	2	1
	CPL C	C←($\overline{C}$)	1	1
	CPL bit	bit←($\overline{bit}$)	2	1
位逻辑运算指令	ANL C,bit	C←(C)∧(bit)	2	2
	ANL C, $\overline{bit}$	C←(C)∧($\overline{bit}$)	2	2
	ORL C, bit	CY←(C)∨(bit)	2	2
	ORL C, $\overline{bit}$	CY←(C)∧($\overline{bit}$)	2	2
位条件转移指令	JC rel	(C)=1转，(PC)←(PC)+2+rel (C)=0，(PC)←(PC）+2	2	2
	JNC rel	(C)=0转，(PC)←(PC)+2+rel (C)=1，(PC)←(PC)+2	2	2
	JB bit,rel	(bit)=1转，(PC)←(PC)+3+rel (bit)=0，(PC)←(PC)+3	3	2
	JNB bit,rel	(bit)=0转，(PC)←(PC)+3+rel (bit)=1，(PC)←(PC)+3	3	2
	JBC bit,rel	(bit)=1，则bit←0转， (PC)←(PC)+3+rel (bit)=0，(PC)←(PC)+3	3	2

2.3 伪指令

学习目标

1. 理解伪指令的意义。
2. 理解 MCS-51 伪指令的功能。
3. 在程序编写中正确使用伪指令。

导　入

在指令系统外配有伪指令，使程序员、编译软件与应用程序联系起来，当在程序中加入了伪指令后编译软件通过伪指令的编译对目标代码进行定位，也明确了所定义的存储空间，当然也应清楚认识伪指令与指令的根本区别。

所谓伪指令是相对MCS-51指令系统而言，指令写成程序以后需要经过编译产生机器码（或者称为目标代码），然后把机器码（目标代码）下载到单片机后可以使CPU运行，伪指令在编译过程中是不形成机器码的。伪指令主要功能是用编译软件（或者称为汇编程序）在编译（汇编）源程序生成机器码的过程中需要明确源程序的开始地址、何处结束、数据定位、分配存储空间等。可以这样理解，伪指令就是在编译程序（汇编程序）实施编译过程中的一个约定和对编译软件编译源程序代码的控制。

2.3.1 源程序起始与结束、赋值伪指令

1. 起始伪指令ORG

格式：ORG 16位地址

ORG命令用来规定汇编语言源程序或数据块存放的绝对起始位置。可以在一个源程序中多处应用，第一个绝对地址值应是最小，后面的依次增大，但是不能重叠，若一个也不用则默认为0000H。16位地址范围是0000H~FFFFH。

2. 结束伪指令END

END命令是汇编语言源程序的结束标志，用于终止汇编工作。

3. 赋值伪指令

格式：字符名称 EQU 常数或汇编符号

命令的功能是将一个常数或汇编符号赋予指定的字符名称，一个字符名称只能被赋值一次。被赋值的字符名称在后面指令中可以作为地址或者是一个立即数。

4. 数据地址赋值伪指令

格式：字符名称 DATA 表达式

命令功能与EQU相同，但使用时有所区别：EQU先定义，后使用；DATA则可先用后定义；DATA只能赋值（表达式可求出值）给字符名称，EQU可以把汇编符号赋给字符名称。DATA伪指令常在程序中用作定义数据地址。

2.3.2 存储空间定义伪指令

1. [标号]：DB 字节数据表

命令功能是在程序存储器中定义由标号开始的存储单元，并存储字节数据，若是数据表则连续存储，一个数据占一个存储单元。

字节数据表可以是一个字节数据或者用逗号分割的多个字节数据，字节数据可以是二进制、十进制、十六进制、ASCII码字符串等。

2. [标号]：DW 字数据表

命令功能是在程序存储器中定义用逗号隔开字（16位2字节）数据表。

存放方式为高8位字节数据先存放在地址值小的单元；低8位字节数据后存入地址值大的单元。

3. 字符名称BIT位地址

命令能是将位地址赋予所规定的字符名称。

【例2-12】 伪指令的综合示例

```
            BUZZ   EQU  P3.0
            FLAG1  BIT   00H
            ORG 1000H
START:      MOV A, #01H;
            ……
            CPL BUZZ;
TEMP__TAB1: DB 01H, 12, ‘A’
            ORG 2000H
TEMP__TAB2: DW 3456H, 5H
            END
```

上述例子一方面说明了伪指令的运用，另一个方面表示了在源程序中指令与伪指令配套运用的一般格式。

2.4 汇编语言程序设计初步

学习目标

1. 掌握汇编语言程序设计基本步骤。
2. 熟悉程序设计中流程图的应用。
3. 掌握程序设计的基本结构与方法。

导　入

单片机系统为什么能够显示数字、字符？为什么能够发出美妙的乐曲？为什么能让洗衣机代替人的动作清洗衣物？……

理解掌握了单片机指令系统接下去的任务就是对不同的单片机硬件应用系统编写程序，让单片机“动起来”。编写程序同样也要有一个基本目标与基本方法，比如需要考虑程序结构，是否写成子程序形式等，然后选择有关的指令，完成程序的编写工作。

2.4.1 汇编语言程序设计的基本方法

采用汇编语句编制程序的过程称为汇编语言程序设计。通常需要掌握一些基本思路与方法。

1. 建立思路、确定算法

从实战要求，对于一个实际项目需要有一个收集资料、调研，拟订设计任务书的过程，任务书大致概括程序功能、技术指标、实施技术路径等。但是初步进行程序设计，怎样来掌握程序设计的基本方法呢？主要是进行一些针对性工作，首先分析单片机操作对象的实时过程和逻辑关系，其次拟制出具体的算法和实施步骤，当然对于同一问题，往往有几种不同的路径实现，需要进行比较从中找出一种比较切合实际的方法。若是基本明确（相对较小）的对象可以直接进行下面工作。

2. 设计程序流程图

程序流程图对于初学者是非常必要的，有句俗话“磨刀不误砍柴功”，设计流程图属于构思程序结构阶段，用规范的图形符号，描述程序设计实现的思路，然后进行程序设计。对于规模小的可以一次完成，规模比较大的往往采用模块化程序设计方法，划分若干子模块（子程序），有了子模块就要考虑是先主程序设计后子程序设计，还是先子程序设计后主程序设计，并分别绘制出相应的程序流程图。程序流程图能充分体现程序设计的思想，使复杂问题简单化，有事半功倍之效。但是流程图的表达的线条可以粗也可以比较细腻，一般对于初学者适宜细腻些。

主程序就是任务实现的主框架，一般来说是一种无限循环结构，其中主要是完成一些

初始化参数设置、对于一些用的标志不断顺序查询，依据标志去调用相关的子程序或者进行中断服务子程序。

3. 汇编语言源程序编写

根据程序流程图可以进行汇编语言源程序的编写，当然程序设计者在基本掌握指令系统的基础上还应该掌握程序设计的基本方法。这样编写的程序对于功能的实现以及程序的可读性都是比较好的。

4. 上机调试

对于编写完成的程序除了找出一些书写问题以外，必须通过在编译环境中的调试（统称上机调试），严格检验程序的正确性，以及功能的可实现性，有时往往是程序语法没有问题，而整个程序指挥硬件功能实现却有瑕疵，需要反复修改完善。最终编译成机器码下载到单片机中。

2.4.2 程序设计基本结构

1. 流程图元素

下面是程序流程图约定的几种符号：

① 椭圆框（或者桶型框）表示程序开始和结束，见图2-4（a）。

② 矩型框表示工作内容，特点是一进一出，见图2-4（b）。

③ 菱形框表示判断，框内为判断表达式，特点是一进二出，见图2-4（c）。

④ 尖头表示程序流向，见图2-4（d）。

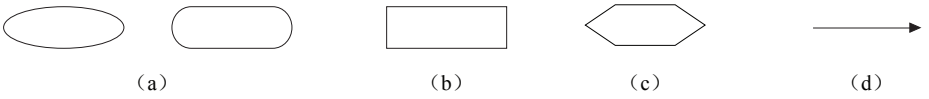


图 2-4 流程图元素

2. 程序设计基本结构

程序设计的基本结构主要有三个，即顺序结构、分支结构和循环结构。为什么是基本结构呢？这主要是从学习角度去理解，把复杂的问题（应用程序）简单化归纳出用汇编指令写程序的基本方法与指令的应用，而实际的应用即所谓复杂问题往往是几种结构套用在一起，没有一个绝对的区分。

① 顺序结构：顺序结构是一种符合指令运行特点的结构，即依次运行每一条指令，完成每一件工作，直至结束，中间没有任何分支或者循环。顺序结构如图2-5（a）所示。

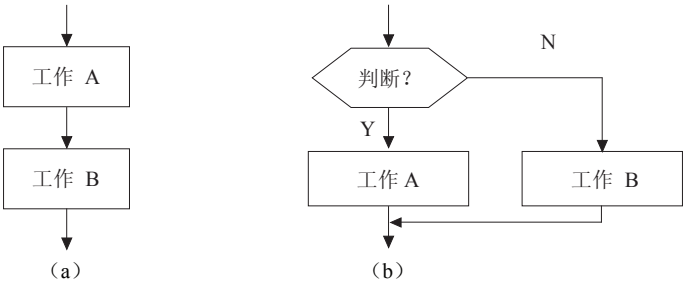


图 2-5 程序的顺序结构和分支结构

（a）顺序结构　（b）分支结构

② 分支结构：顺序执行是一种单一的编程结构，有时依据实际情况，完成一项任务或者是一个算法需要按照特定的条件进行判断，判断后再选择运行方向。同理在指令系统中也分别给出了条件转移指令和无条件转移指令，因此形成了具有分支结构的程序设计。针对客观情况与选用合适的条件（无条件）转移指令可以编制适合更大范围的应用程序。分支结构按条件的多少可以化分为单分支与多分支结构。分支结构如图2-5（b）所示。

【例2-13】 16位加数与16位被加数分别存放在31H~30H和41H~40H单元。把求出的和存放在51H～50H单元。程序流程如图2-6。

```
START：  MOV R1,30H
         MOV A, 40H
         ADD  A,  R1    ；  低字节相加
         MOV  50H, A  ；  和的低字节
         MOV  R1, 31H
         MOV  A, 41H
         ADDC A,R1    ；  高字节相加，带低字节的进位位
         MOV 51H,A    ；  和的高字节
```

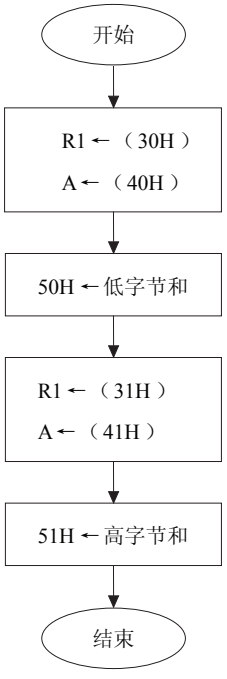


图 2-6 两字节加法实现流程图

【例2-14】 设计一个简单的物体称重显示程序，程序流程如图2-7。

要求：当重量为5KG时，P1.0显示（P1.0←‘0’）；当重量小于5KG时，P1.1显示；当重量大于5KG时，P1.2显示。重量值存放在30H单元。

```
MOV A,30H;              取出存放的重量值
CJNE A,#05H,NEQU1;      与5KG比较，不等则转移至NEQU1
MOV 90H,#0FEH;          相等P1.0显示
SJMP STORE
```

```
NEQU1: JC NEQU2
       MOV 90H,#0FBH ;      大于则P1.2显示
       SJMP STORE
NEQU2: MOV 90H,#0FDH ;      小于则P1.1显示
STORE: SJMP $        ;      $表示PC的当前值，本条指令供调试使用
```

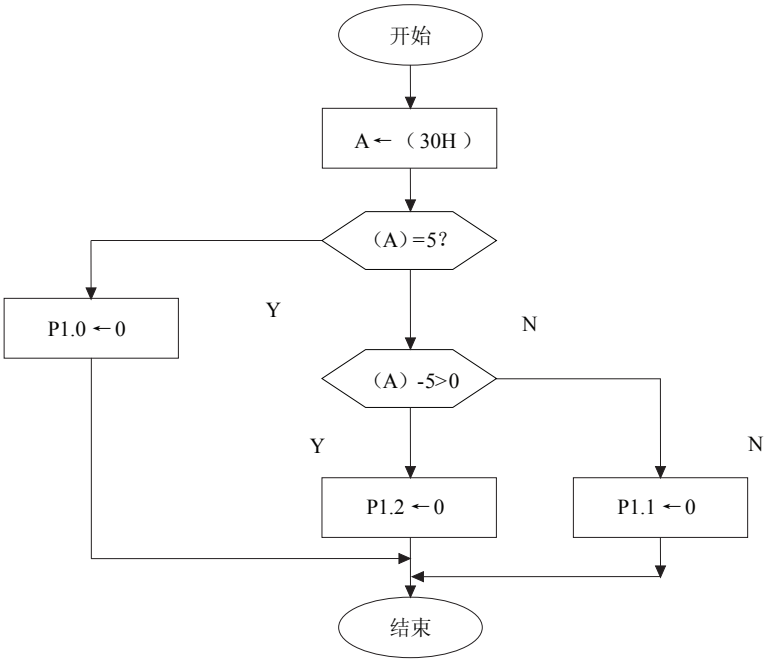


图 2-7 例题2-14程序流程图

【例2-15】以【例2-7】内容为例，设计多分支流程图，见图2-8。

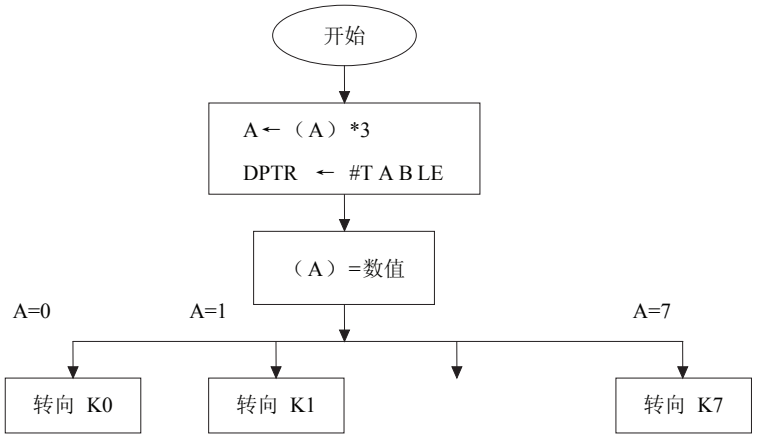


图 2-8 例题2-15多分支程序流程图

③ 循环结构：除了顺序结构与分支结构以外，同样还会遇到第三种情况，即在一个大程序中往往容易产生对于有些操作需要反复运行。例如：假设在一个程序中需要计算0~50累加之和，如果采用顺序结构的方法实现，则程序烦琐，内存资源浪费，这时可以采用循环结构方式实现，使程序结构简单、易读、方便调试和节约内存资源。

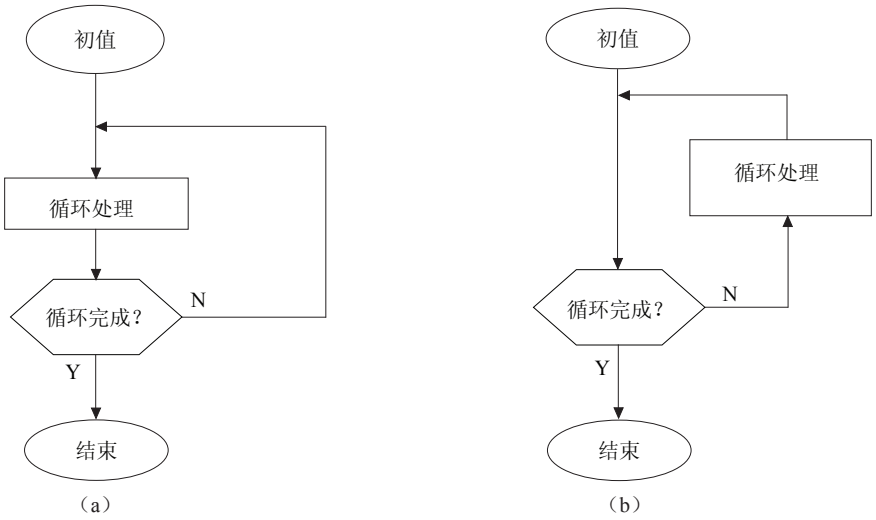


图 2-9 循环程序基本结构

（a）先处理后判断循环　（b）先判断后处理循环

怎样实现循环结构呢？首先对于循环程序要进行一些准备工作，首先是初始化设置，包括循环变量设置、初值等；其次，循环操作内容即循环体；第三，循环控制变量修改和控制用来判断循环是继续，还是结束。

循环结构的组织如图2-9所示的先处理后判断与先判断后处理两种。在这两种循环结构的基础上还可以采取小循环外层套上大循环，构成循环嵌套的多重循环，但是需要注意的是不能进行交叉循环嵌套。

【例2-16】 把片外RAM从2000H~200FH16个单元清0，程序流程如图2-10。

```
        MOV R7,#10H;          设循环初值
        MOV A,#00H
        MOV  DPTR,#2000H;     片外RAM首地址送2000H
LOOP:   MOVX @DPTR, A;        循环体
        INC  DPTR
        DJNZ  R7,LOOP;        循环变量修改且判断是否执行完毕。
        SJMP  $
```

【例2-17】 多重循环实现1S延时程序段。

```
DELAY:  MOV  R3,#10              ; 执行1次
DELAY1: MOV  R4,#200             ; 执行10次
DELAY2: MOV  R5,#250             ; 执行10×200=2000次
        DJNZ  R5,$               ; 执行10×200×250=500000次
        DJNZ  R4,DELAY2          ; 执行10×200=2000次
        DJNZ  R3,DELAY1          ; 执行10次
        RET
```

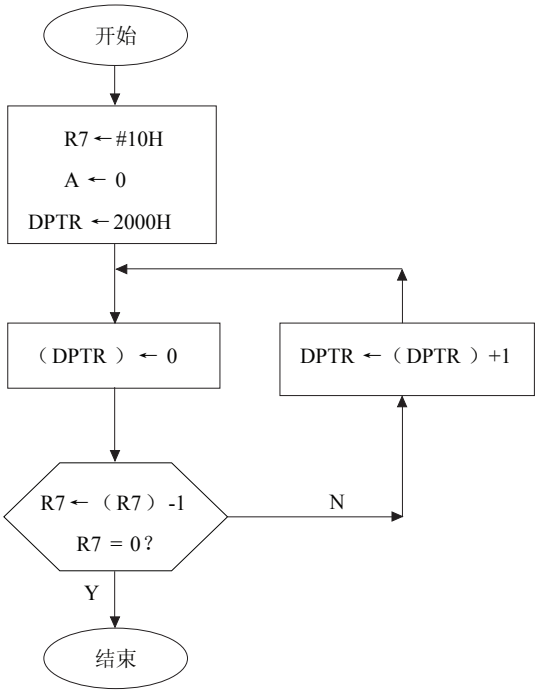


图 2-10 例题2-16流程图

说明：执行延时时间计算：对于选用12MHz晶振时执行一条MOV R3，#10指令是1个机器周期（1μs）；执行一条DJNZ R5，$指令是2个机器周期（2μs）；RET指令为2个机器周期（2μs）。总的延时时间就是循环执行指令总的次数。

总的机器周期为程序右边之执行次数总和：504022。

则总的时间：（1+10+2000）×1μs=2011μs

（500000+2000+10+1）×2μs=502011×2=1004022μs

2011μs+1004044μs=1006033μs≃1s

④ 子程序调用：在前一节中介绍了有关子程序调用和返回指令的应用，子程序的适合范围为

- 多次需要重复运行的程序段；
- 具有通用意义的程序段；
- 中断服务子程序。

而对于调用子程序的程序称为主调程序，简称主程序。那么，主程序与被调用的子程序之间需要进行有关参数的传递，例如主程序把参与子程序运算的数据（入口参数）提供给子程序（可以存放在一个约定的位置），当子程序取得参数且运行完成以后在子程序结束前把结果（出口参数）返回给主程序（送回约定之位置），让主程序继续运行。参数传递主要有：

- 用累加器A或者是工作寄存器R0~R7传递参数；
- 用堆栈传递；
- 用存储单元地址。

知识拓展

通过查阅资料编写程序：

1. 实现十进制BCD数减法程序。

2. 设表中有512个数据，每个数据是16位。设定16位数据存放在R2，R3中，然后在512中查找匹配数据，可以分一个和多个数据匹配。

要求：有题目分析、所用的硬件资源、源程序、调试结果和小结。

本章小结

硬件是运用单片机的实物基础，而指令则是单片机运用的程序基础。本章分别叙述了有关指令、程序的概念，89C51汇编指令的格式，相关符号，涵盖指令系统的7种寻址方式，对于111条指令按照功能分为数据传送、算术运算、逻辑运算、控制转移、位操作等几个大类以及汇编源程序编译过程中用的伪指令。

程序设计初步一节中分别介绍了设计程序的基本步骤，流程图的应用以及在设计程序过程中的三种结构：顺序结构、分支结构中的单分支、多分支和循环结构及其子程序调用的应用。为避免学习指令系统的枯燥和不能从应用角度出发去掌握指令，在介绍过程中采取从系统角度、应用角度分类说明指令的功能，尽量从简单、常用、基础出发选择例题，起到在理解指令的过程中的画龙点睛的效果。

思考与练习题

一、单项选择题

1. MCS-51的无条件转移指令中，其转移范围最小的是（　　）。

A. SJMP　　B. AJMP　　C. LJMP　　D. ACALL

2. 以下指令中，哪条指令执行后使标志位CY清0（　　）。

A. MOV A，#00H　B. CLR A　C. ADD A，#00H　D. MOV 21H. #0

3. 下面那一段程序能准确地读取P1口引脚信号（　　）。

A. MOV A，#00H
　MOV P1，A
　MOV A，P1

B. MOV A，#0FFH
　MOV P1，A
　MOV A，P1

C. MOV A，#0FFH
　MOV A，P1

D. MOV A，#00H
　MOV A，P1

4. 对片外数据RAM单元读写数据须用（　　）。

A. MOV指令　B. MOVX指令　C. MOVC指令　D. LCALL

5. 将片内RAM中的数据送至ACC，执行指令为（　　）。

A. MOVC A, @A+DPTR　　B. MOV A, @R0

C. MOVX A, @DPTR　　D. MOVC A, @A+PC

二、判断题

1. 执行CLR 30H指令后，30H字节单元被清0。（　　）
2. 中断服务程序执行的最后一条指令必须是RETI。（　　）
3. 执行LCALL指令时，栈指针SP的内容不会发生变化。（　　）
4. 主程序与子程序可以互为调用。（　　）
5. MOV A, #45H指令中源操作数寻址方式为直接寻址。（　　）

三、问答题

1. 通过列表表示寻址方式与对应存储空间及寄存器之间关系。
2. 什么是“伪指令”，伪指令与指令有何区别？
3. 用三种方法把累加器A中无符号数乘以2。
4. 如何实现减法的十进制数的调整？
5. 已知：（30H）=A7A6A5A4A3A2A1A0B，（31H）=B7B6B5B4B3B2B1B0B，请给出下列每条指令执行后注释中的结果。

```
MOV   32H, 30H;     （32H）=
ANL   32H, #0FH;    （32H）=
MOV   A, 31H;         （A）=
SWAP  A;              （A）=
RL    A;              （A）=
ANL   A, #0F0H;       （A）=
```

```
ORL    32H, A;          （32H）=
```

6. 执行下列程序后，累加器A和栈指针SP各为何结果？

```
MOV   SP, #60H;
MOV   A, #0ABH
LCALL   SUBRT
INC   A
HERE: SJMP  HERE
SUBRT: PUSH  A
XRL   A, #0F0H
POP   A
RET
```

7. 已知（30H）=95H，（31H）=8FH，问执行下列程序后32H和33H单元内容是什么？该程序的功能是什么？

```
MOV   R0, #30H
MOV  A, @R0
INC  R0
ADD  A, @R0
INC  R0
MOV  @R0, A
CLR  A
ADDC  A, #00H
INC  R0
MOV  @R0,
```

8. 设f_{osc}=12MHz，试编写10ms、500ms的延时子程序。

9. 设有一个字符串是以‘$’结尾的，存放在30H开始的内部RAM之中，试通过编程计算字符串的长度。

10. 一个字节内存有两个BCD码十进制数，通过编程实现把两个数转换成相应的ASCII码存放在30H和31H单元中。

实验项目2　汇编程序阅读与源程序调试

一、 实验目的

1. 进一步对汇编指令功能的理解。
2. 通过阅读程序掌握程序设计的方法。
3. 巩固Keil μVision2 集成开发环境操作步骤。
4. 掌握程序调试的基本方法。

二、 实验设备

1. PC计算机一台
2. Keil μVision2集成开发环境

三、 实验内容

1. 完成对主要语句的阅读、写出注释以及程序流程图。
2. 分析程序功能。
3. 通过调试验证分析的正确性。
4. 实验程序

程序一：

```
        ORG  0000H
        LJMP  MAIN
        ORG  0100H
MAIN:  MOV  A, R1;   R1内存放待转换二进制数
        MOV  B, #64H
        DIV  AB
        MOV  R3, A;   R3存放BCD码百位
        MOV  A, #0AH
        XCH  A, B
        DIV  AB
        SWAP  A
        ADD  A, B
        MOV  R2, A;   R2存放BCD码十位，个位。
        SJMP  $
        END
```

程序二：

```
        ORG  0000H
        LJMP  MAIN
```

```
        ORG  0100H
MAIN:  MOV  A,  #n; 0≤#n≤9
        ADD  A,  #rel; 通过字节数计算rel。
        MOVC  A, @A+PC
        MOV  40H, A
        SJMP $
TABEL: DB  00, 01, 04, 09
        DB  16, 25, 36, 49
        DB  64, 81
        END
```

程序三：

```
        ORG  0000H
        LJMP  MAIN
        ORG  0100H
MAIN:  MOV  90H, #0FFH
        MOV  R2, #00H
        MOV  R1, #0AH
        MOV  R0, #30H
        MOV  X @DPTR, A
EQ1: CJNE  @R0, #0AAH, LOOP
        INC  R2
        ACALL  LEDL
LOOP:  ACALL  DELAY
        INC  R0
        DJNZ  R1, EQ1
        SJMP $

LEDL:  MOV  90H,#0FEH
        ACALL  DELAY
        MOV  90H, #0FEH
        RET
DELAY: MOV  R3, #10
DELAY1: MOV  R4, #200
DELAY2: MOV  R5, #250
          DJNZ  R5, $
          DJNZ  R4, DELAY2
          DJNZ  R3, DELAY1
          RET
          END
```

四、 实验操作

在指定的文件夹中建立工程文件、建立汇编语言源程序文件，建立完成以后进行编译，修改错误，直至编译无错误。

分别进行调试运行，其中可采用设置断点、单步/全速等运行操作，包括可以打开相关的窗口进行验证与调试。

第3章 中断系统

任何计算机都包含中断系统，单片机也是如此。中断技术很好地化解了快速CPU和慢速外部设备间的矛盾，使得CPU和外部设备可以并行工作，从而大大提高了CPU的工作效率。本章着重讲述中断的基本概念，并围绕89C51单片机中断系统的结构和工作原理以及中断应用编程方法等展开详细介绍。

3.1 微机的输入/输出方式

学习目标

1. 掌握无条件传送、查询传送和中断传送方式的基本工作原理及特点。
2. 理解引入中断传送方式的根本原因。

导　入

输入/输出设备是附加到计算机系统中用来加强计算机功能的设备。譬如，办公室里常用的打印机就属于比较典型的在计算机管理和控制下的输出设备。但计算机在控制打印机的同时，为什么还可以处理其它各种事情，即计算机的并行作业。

在微机系统中，大量数据在CPU、存储器和外设之间进行传送，而传送中的关键问题是如何实施数据传送的控制方式，归纳起来通常有两大类控制方式，即程序控制传输方式和直接存储器存取方式。

3.1.1 程序控制传输方式

程序控制传输方式的特点是以CPU为中心，由CPU控制，也就是让CPU执行预先编制好的输入或输出程序来实现数据的传送。此方式又可分为如下三种情况。

1. 无条件传送方式

无条件传送方式下，CPU总是认为外设在任何时刻都处于“准备好”的状态。因此，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，便可以实现数据传送。如按钮开关、发光二极管等，类似这种外设都可以处于CPU控制之下。这里要指出的是，并非所有的外设都可以采用无条件传送方式实施控制，如一些工作速度较慢的外设。

2. 查询传送方式

查询传送也称为条件传送，这种传送方式用以解决CPU与外设之间的速度匹配问题。当CPU与外设工作不同步时，很难确保CPU在执行输入操作时，外设一定是“准备好”的状态；而在执行输出操作时，外设一定是“空闲”的状态。为保证数据传送的可靠进行，可采用此方式。

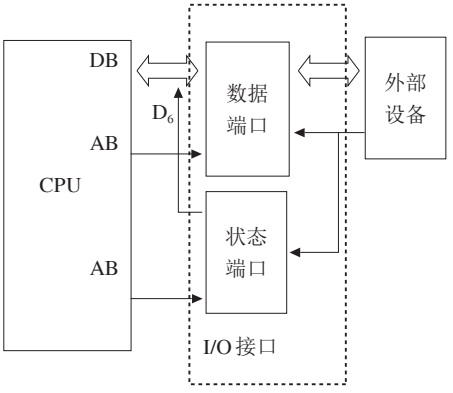


图 3-1 查询传送的I/O接口电路

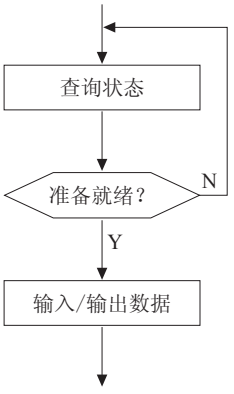


图 3-2 查询传送的程序控制

查询传送方式的I/O接口电路设计如图3-1所示。在I/O接口电路中至少要含两个端口电路，一个是数据端口，用以进行数据传输，另一个是状态端口，用来查询外设状态。在传输数据之前，CPU要先通过状态端口查询外设当前的工作状态，只有确认外设已经具备了传输条件后，才能通过数据端口完成数据传输。

查询传送方式的流程如图3-2所示。程序控制过程中，先查询I/O设备的当前状态，若准备就绪，则交换数据，否则循环查询状态。

查询传送方式的优点是通用性好，可以用于各类外设和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，由于外设工作速度比CPU慢得多，因此，CPU在完成一次数据传送后要等待很长时间（与数据传送相比），才能进行下一次的传送。在等待过程中，CPU不能进行其他操作，所以效率比较低。提高CPU效率的一条有效途径是采用中断方式传送。

3. 中断传送方式

中断传送方式不存在查询传送方式中CPU等待外设的问题。当外设需要CPU为自己服务时，可主动向CPU发出中断请求信号，使CPU中断原来执行的程序（主程序），转去执行为外设服务的输入或输出操作，服务完毕，CPU再继续执行原来的程序。中断传送方式

的基本工作原理如图3-3和图3-4所示。

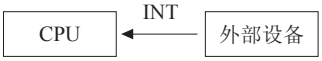


图 3-3 外设中断请求

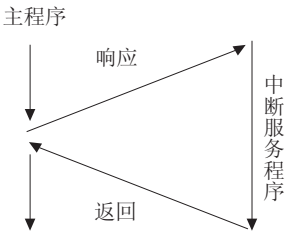


图 3-4 中断过程

采用中断传送方式，CPU和外设可并行工作， 从而大大提高了CPU的工作效率和控制程序执行的实时性。

3.1.2 直接存储器存取方式（DMA传输方式）

中断传送方式可大大提高CPU效率，但仍由CPU来传送，要用不少指令，有些情况下依然太慢，如磁盘与内存间交换数据、高速采集、高速数据块传送等，通常采用DMA(Direct Memory Access)传送，进行DMA传送要用到DMA控制器（DMAC），即为DMA传送而设计的专门接口。DMA方式是CPU让出数据总线（悬浮状态），不通过CPU，使外部设备和存诸器之间直接传送数据的方式。DMA可实现：

1. 存储器←→外设（彼此间的传送）

2. 存储器←→存储器（彼此间的传送）

值得一提的是在MCS-51单片机中几乎不会采用DMA传输方式。

3.2 中断的基本概念

学习目标

1. 理解和掌握中断的基本概念与术语。
2. 掌握中断响应及处理过程。

导　入

人们的日常生活中也有中断，微机系统中有中断，两者的概念基本是一样的。譬如：你正在家中看书，突然电话铃响了，你放下书本，去接电话，和来电话的人交谈，然后挂断电话，回来继续看你的书。这就是生活中的“中断”的现象，中断实际上就是指正常的工作过程被外部的事件打断了。仔细研究一下生活中的中断，对于我们理解微机系统的中断大有帮助。

在微机的输入／输出方式中，我们已经谈到了当CPU用查询方式与外设交换信息时，CPU就要浪费很多时间去等待外设。这样就引出一个快速的CPU与慢速的外设之间数据传输的矛盾，这也是计算机在发展过程中遇到的一个严重问题。为解决这个问题，一方面要提高外设的工作速度，另一方面发展了中断概念。可以这样讲，在了解和掌握微机知识的过程中，中断技术属于最为突出的内容，也是较为复杂的部分。本节将讨论一些最基本的中断概念和中断响应及处理过程。理解和掌握这些概念，对于进一步学习MCS-51单片机的中断系统大有裨益。

3.2.1 中断的常用术语

1. 中断

为了帮助读者对计算机的中断有更深入地理解，图3-5中给出了计算机的中断和日常生活中的电话中断两者之间的对比。

执行主程序	=	某人看书	→	日常事务
中断信号INT	=	电话铃响	→	中断请求
暂停执行主程序	=	暂停看书	→	中断响应
当前PC入栈	=	书中作记号	→	保护断点
执行I/O程序	=	电话谈话	→	中断服务
返回主程序	=	继续看书	→	中断返回

图 3-5 计算机中断和电话中断对比

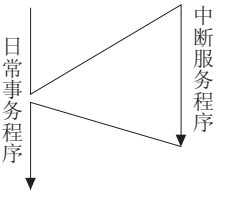


图 3-6 中断过程

从对比中可以看出，两者的概念非常相似。其中，CPU执行主程序类似于某人看书，可以说成是一种日常事务；外设向CPU发出中断信号INT类似于电话铃响，称之为中断请求，也就是表明外部有事件需要服务，这里需要说明的是外设产生中断请求信号的方式很多，可以是一个电平的变化、一个脉冲沿的发生或定时器计数溢出等；CPU暂停执行主程序类似于暂停看书，称之为中断响应，因为外部有事件需要服务，此时只能暂时中止自身的日常事务；当前PC入栈类似于书中作记号，称之为保护断点，此时之所以要保护断点，是因为外部事件服务完毕后，还要回来接着做日常事务，这里需要说明的是对于MCS-51单片机而言，断点指的就是CPU暂停执行主程序时刻所对应的程序计数器PC的值（下一条将要执行的指令地址），一旦进入中断响应硬件会自动把PC值压入堆栈保护起来；执行I/O程序类似于电话谈话，称之为中断服务，也就是为外部事件进行工作，这里需要说明的是计算机为外设服务，实际上就是CPU执行事先编制好的服务于外设的程序，又称中断服务程序；返回主程序类似于继续看书，称之为中断返回，也就是返回到断点处，接着做日常事务，这里需要说明的是从中断服务程序返回到主程序，CPU是通过执行中断服务程序中安排的中断返回指令RETI来实现的。

计算机的中断用规范的语言来描述，就是指计算机在执行某一程序过程中，由于计算机系统内、外的某种原因，而必须中止原程序的执行， 转去执行相应的处理程序（中断服务程序），待处理结束之后，再回来继续执行被中止的原程序的过程。其示意图如图3-6。

计算机的中断功能是很有用的，有时甚至是必不可少的。如PC机的键盘管理就是使用中断功能的一个比较好的例子。键盘在工作时，每次击键就对CPU产生一次中断，因为CPU正在执行其他程序，所以不可能同时处于等待用户击键的循环之中。如果真的等待用

户击键，则其他程序永远得不到执行，因为CPU的全部注意力都集中在寻找下一次击键事件。一个简单的解决办法是使正在执行的程序暂停一下，并查看键盘接口是否已有键被击过。当然，应用程序必须知道什么时候去看、看多少次，否则许多处理时间将白白浪费在对键盘接口的查询之中。PC的中断功能就是在击键事件刚发生时自动暂停程序，并使CPU去执行与击键事件相对应的键盘服务程序。在完成接收输入信息的程序之后，CPU自动回到原程序去执行下一条指令。

计算机采用了中断技术后能实现以下功能：

① 分时操作：计算机的中断系统可以使CPU与外设并行工作。CPU在启动外设后，便继续执行主程序；而外设被启动后，开始进行准备工作。当外设准备就绪时，就向CPU发出中断请求，CPU响应该中断请求并为其服务完毕后，再返回到原来的断点处继续运行主程序。外设在得到服务后，也继续进行自己的工作。因此，CPU可以使多个外设同时工作，并分时为各外设提供服务，从而大大提高了CPU的利用率和输入 / 输出的速度。

② 实时处理：当计算机用于实时控制时，请求CPU提供服务是随机发生的。有了中断系统CPU就可以立即响应并加以处理。

③ 故障处理：计算机在运行时可能会出现一些故障，如电源断电、存储器奇偶校验出错、运算溢出等。有了中断系统，当出现上述情况时CPU可及时转去执行故障处理程序，自行处理故障而不必停机。

2. 中断源

中断源是指在计算机系统中向CPU发出中断请求的来源。中断可以人为设定, 也可以为响应突发性随机事件而设置。中断源通常有I/O设备、定时时钟、实时控制系统中的随机参数和信息故障源等。

中断源又可分为可屏蔽中断源和非屏蔽中断源。

可屏蔽中断源指的是可被程控“开中断/关中断”的中断源，也就是说可以通过软件设置允许/禁止CPU响应其中断请求。如果软件设置成允许CPU响应中断，那么可屏蔽中断源发出的中断请求CPU会作出响应；反之软件设置成禁止CPU响应中断，即便中断源有中断请求，CPU也不会作出中断响应。MCS-51单片机所有的中断源都是属于此类中断源。

非屏蔽中断源指的是不可被程控“关中断”的中断源，也就是说不可以通过软件的设置来禁止CPU响应中断，这类中断源有中断请求时，CPU必须作出响应。MCS-51单片机不存在该类中断源。

3. 中断优先级和中断嵌套

中断优先级指的是有多个中断源同时发出中断请求信号时，CPU响应其中断的先后顺序。道理很简单，因为CPU只有一个，中断服务程序只能一个一个的执行，必然有先后顺序。先响应的优先级别高，后响应的优先级别低。通常情况下，中断源的中断优先级可以通过软件设置，同时硬件设计上也会作出自然安排。

中断嵌套指的是高优先级中断源可以中断低优先级中断源的中断服务。

3.2.2 中断响应及处理过程

通常情况下，计算机形成中断先要满足中断响应条件，然后再进入中断处理过程。需要指出的是不同型号的CPU其内部的中断系统硬件结构设计是不一样的，因此相应的中断

处理过程也不尽相同。MCS-51单片机的中断响应及处理过程概述如下。

1. 中断响应条件

① 中断源发中断请求信号。

② 系统处于开中断状态。

2. 中断处理过程

① 保护断点：将断点地址压入堆栈保存，即当前PC值入栈。

② 寻找中断源：中断服务程序入口地址→PC，转入中断服务。

③ 关中断：屏蔽其它中断请求信号。

④ 保护现场：将中断服务程序使用的所有寄存器内容入栈。

⑤ 开中断：允许接受其它中断请求信号。

⑥ 中断处理：执行中断源所要求的程序段。

⑦ 关中断：屏蔽其它中断请求信号。

⑧ 恢复现场：恢复被使用寄存器的原有内容。

⑨ 开中断：允许接受其它中断请求信号。

⑩ 中断返回：执行RETI指令，栈顶内容→PC，程序跳转回断点处。

MCS-51单片机的中断响应及处理过程的流程如图3-7。

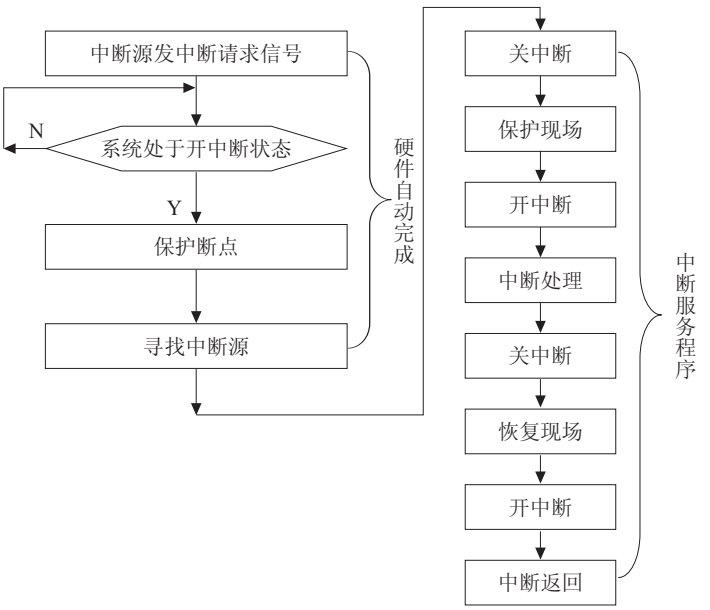


图 3-7 中断响应及处理过程

3.3 89C51中断系统的结构和工作原理

学习目标

1. 理解和掌握89C51中断系统的结构和工作原理。
2. 掌握89C51内部与中断有关的4个寄存器的功能和使用的方法。
3. 理解89C51的中断响应条件及中断处理过程。

导　入

在单片机中，“中断”是一个很重要的概念。中断技术的进步使单片机的发展和应用大大地推进了一步。所以，中断功能的强弱已成为衡量单片机功能完善与否的重要指标。能够实现中断处理功能的部件称为中断系统。具体地讲，在了解计算机的中断系统过程中，关键是要研究清楚从中断源提出中断申请到CPU响应中断这一过程中整套的处理方法和实现手段。

3.3.1 89C51的中断系统结构

MCS-51系列单片机内部的中断系统结构大同小异，只是基本型51子系列的单片机有五个中断请求源，而增强型52子系列的单片机有六个中断请求源。图3-8给出的是51子系列中型号为89C51单片机的中断系统内部结构。

如图3-8中所示，89C51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和SCON，用来控制中断的类型、中断的开／关和各种中断源的优先级别。五个中断源有两个中断优先级，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嵌套。

1. 五个中断源

① $\overline{\text{INT0}}$：外部中断0（P3.2脚输入），由IT0位（TCON.0）来决定是低电平有效还是下跳变有效。一旦输入信号有效，就向CPU申请中断，并建立IE0标志。

② $\overline{\text{INT1}}$：外部中断1（P3.3脚输入），由IT1位（TCON.2）来决定是低电平有效还是下跳变有效。一旦输入信号有效，就向CPU申请中断，并建立IE1标志。

③ 定时器T0：定时器T0溢出中断，当T0产生溢出时，T0中断请求标志位TF0（TCON.5）置位，请求中断处理。

④ 定时器T1：定时器T1溢出中断，当T1产生溢出时，T1中断请求标志位TF1（TCON.7）置位，请求中断处理。

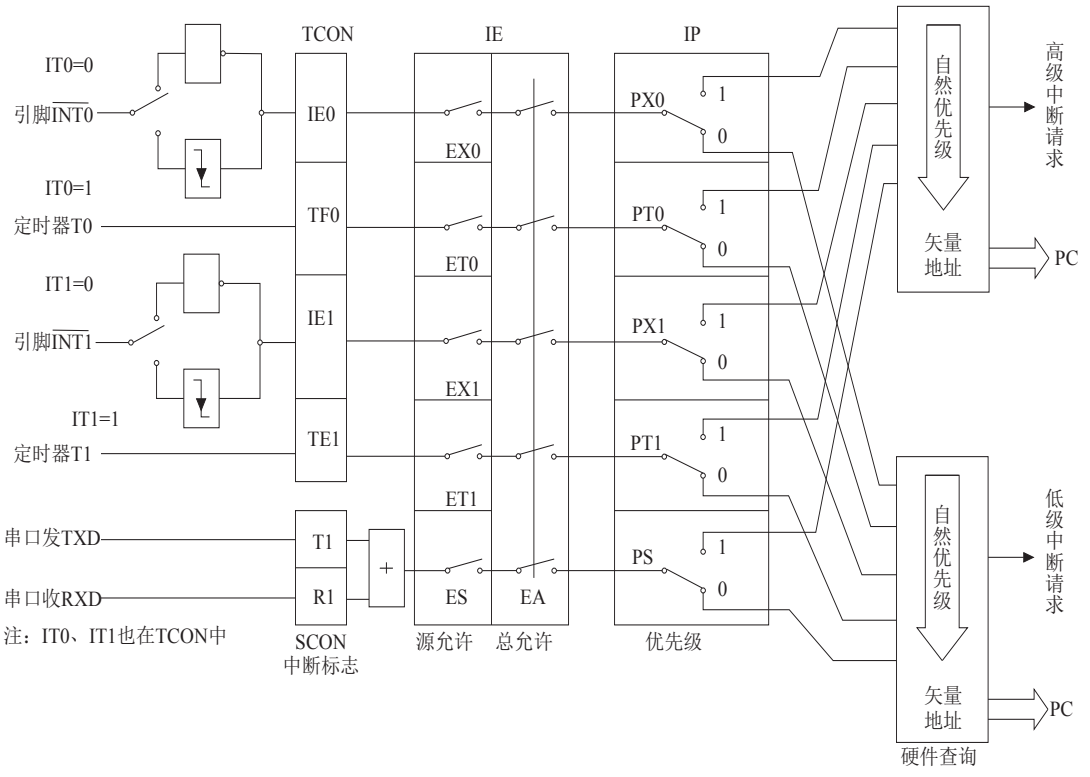


图 3-8 89C51中断系统结构

⑤ TXD/RXD：串行口中断请求。当接收或发送完一帧数据时，内部串行口中断请求标志位RI（SCON.0）或TI（SCON.1）置位，请求中断。

2. 中断允许控制

- CPU总允许开关：EA。
- 各中断源允许开关：ES、ET1、EX1、ET0、EX0。

3. 2级中断优先级控制

- 各中断源优先级选择开关：PS、PT1、PX1、PT0、PX0。

3.3.2 89C51与中断有关的寄存器

表3-1给出的是89C51与中断有关的四个特殊功能寄存器，通过对它们的各位进行置位或复位等操作，可实现各种中断控制功能。使用说明如下。

表 3-1 与中断有关的寄存器

寄存器名称		D7	D6	D5	D4	D3	D2	D1	D0
定时器控制寄存器	TCON（88H）	TF1		TF0		IE1	IT1	IE0	IT0
串行口控制寄存器	SCON（98H）							TI	RI
中断允许寄存器	IE（A8H）	EA			ES	ET1	EX1	ET0	EX0
中断优先级寄存器	IP（B8H）				PS	PT1	PX1	PT0	PX0

1. 中断请求标志位

TF1、TF0、IE1、IE0、RI、TI（其中TF1、TF0、IE1、IE0位于定时器控制寄存器TCON中，RI、TI位于串行口控制寄存器SCON中）。

以上6位分别用于登记各中断源请求信号：=1有中断请求，=0无中断请求。CPU响应中断后，TF1、TF0、IE1和IE0硬件自动清0。TI、RI必须软件清0。详细说明如下。

- TF1：定时器／计数器T1的溢出中断请求标志位。当启动T1计数以后，TI从初值开始加1计数，计数器最高位产生溢出时，由硬件使TFI置1，并向CPU发出中断请求。当CPU响应中断时，硬件将自动对TF1清0。
- TF0：定时器／计数器T0的溢出中断请求标志位。含义与TF1相同。
- IE1：外部中断$\overline{\text{INT1}}$的中断请求标志。当检测到外部中断引脚1上存在有效的中断请求信号时，由硬件使IEl置1。当CPU响应该中断请求时，由硬件使IE1清0。
- IE0：外部中断$\overline{\text{INT0}}$的中断请求标志。其含义与IEl相同。
- TI：串行口发送中断请求标志。CPU将一个数据写入发送缓冲器SBUF时，就启动发送。每发送完一帧串行数据后，硬件置位TI。但CPU响应中断时，并不清除TI，必须在中断服务程序中由软件对TI清0。
- RI：串行口接收中断请求标志。在串行口允许接收时，每接收完一帧串行数据后，硬件置位RI。同样，CPU响应中断时不会清除RI，必须用软件对其清0。

2. 外部中断触发方式选择位

IT1、IT0（位于TCON中）。

以上2位分别用于设置两个外部中断源电信号的触发方式：=1下降沿触发中断请求；= 0低电平触发中断请求。详细说明如下。

- IT1=0时，外部中断$\overline{\text{INT1}}$程控为电平触发方式。CPU在每一个机器周期S5P2期间采样$\overline{\text{INT1}}$请求引脚的输人电平。若$\overline{\text{INT1}}$请求为低电平，则使IE1置1；若$\overline{\text{INT1}}$请求为高电平，则使IEl清0。ITl = 1时，$\overline{\text{INT1}}$程控为边沿触发方式。CPU在每一个机器周期S5P2期间采样$\overline{\text{INT1}}$请求引脚的输人电平。如果在相继的两个机器周期采样过程中，一个机器周期采样到$\overline{\text{INT1}}$请求为高电平，下一个机器周期采样到$\overline{\text{INT1}}$请求为低电平，则使IE1置1。直到CPU响应该中断时，才由硬件使IE1清0。
- IT0：外部中断$\overline{\text{INT0}}$的中断触发方式控制位。其含义与IT1类同。

3. 中断允许控制位

EA、ES、ET1、EX1、ET0、EX0（位于中断允许寄存器IE中）。

以上6位是用于中断源的 “开中断/关中断” 设置：=1开中断；= 0关中断。详细说明如下。

89C51对中断的开放和关闭实现两级控制。所谓两级控制．就是有一个总的开关中断控制位EA，当EA= 0时，屏蔽所有的中断申请，即任何中断申请都不接受；当EA=1时，CPU开放中断，但五个中断源还要由IE中各对应控制位的状态进行中断允许控制。

- EA：中断允许总控制位。EA= 0，屏蔽所有中断请求；EA=1，CPU开放中断。对各中断源的中断请求是否允许，还要取决于各中断源的中断允许控制位的状态。
- ES：串行口中断允许位。ES=0，禁止串行口中断；ES=1，允许串行口中断。
- ET1：定时器T1的溢出中断允许位。ET1 = 0，禁止T1中断；ET1=1，允许T1中断。
- EX1：$\overline{\text{INT1}}$中断允许位。EX1=0，禁止$\overline{\text{INT1}}$中断；EX1=1，允许$\overline{\text{INT1}}$中断。

- ET0：定时器T0的溢出中断允许位。ET0=0，禁止T0中断；ET0 =1，允许T0中断。
- EX0：$\overline{\text{INT0}}$中断允许位。EX0=0，禁止$\overline{\text{INT0}}$中断；EX0=1，允许$\overline{\text{INT0}}$中断。

【例3-1】允许CPU响应$\overline{\text{INT0}}$的中断请求。

```
SETB EX0          ；INT0开中
SETB EA           ；CPU开中
```

4. 中断优先级控制位

PS、PT1、PX1、PT0、PX0（位于中断优先级寄存器IP中）。

以上5位是分别用于五个中断源的优先级设置：=1为高优先级，= 0为低优先级。详细说明如下。

89C51有两个中断优先级。每一个中断请求源均可编程为高优先级中断或低优先级中断。中断系统中有两个不可寻址的“优先级生效”触发器，一个指出CPU是否正在执行高优先级的中断服务程序，另一个指出CPU是否正在执行低优先级中断服务程序。这两个触发器为1时，则分别屏蔽所有的中断请求。另外，对各中断源优先级的安排，可通过IP中各对应位用软件来设定。

- PS：串行口中断优先级控制位。PS=0，低优先级；PS=1，高优先级。
- PT1：定时器 / 计数器T1中断优先级控制位。PT1=0，低优先级；PT1=1，高优先级。
- PXl：外部中断1中断优先级控制位。PX1=0，低优先级；PX1=1，高优先级。
- PT0：定时器 / 计数器TO中断优先级控制位。PT0=0，低优先级；PT0=1，高优先级。
- PX0：外部中断。中断优先级控制位。PX0=0，低优先级；PX0=1，高优先级。

若某几个控制位为1，则相应的中断源就规定为高级中断；反之，若某几个控制位为0，则相对应的中断源就规定为低级中断。

当同时接收到几个同一优先级的中断请求时，响应哪个中断源则取决于内部硬件查询顺序。其优先级的顺序排列如图3-9所示。

图 3-9 同级中断源优先级排列

通常，系统中有多个中断源，因此就会出现数个中断源同时提出中断请求的情况。这样，就必须由设计者事先根据它们的轻重缓急，为每个中断源确定一个CPU为其服务的顺序号。当数个中断源同时向CPU发出中断请求时，CPU根据中断源顺序号的次序依次响应其中断请求。

当CPU正在处理一个中断请求时，又出现了另一个优先级比它高的中断请求，这时，CPU就暂时中止执行对原来优先级较低的中断源的服务程序，保护当前断点，转去响应优先级更高的中断请求，并为其服务。待服务结束，再继续执行原来较低级的中断服务程序，这就是如前所述的中断嵌套。图3-10给出的是二级中断嵌套的中断过程。值得指出的是：89C51的同级中断源不能进行中断嵌套，因此最多只能实现二级中断嵌套。

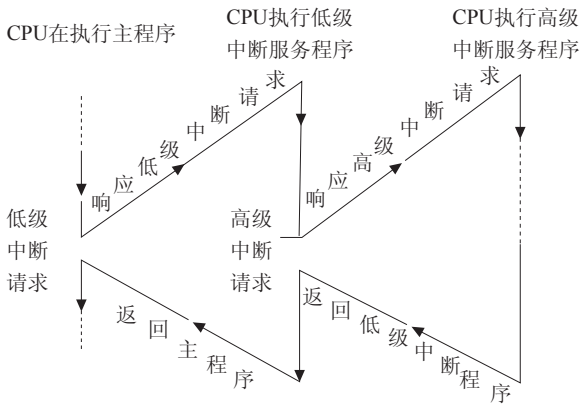


图 3-10　二级中断嵌套

【例3-2】设89C51的片外中断为高优先级，片内中断为低优先级。

```
SETB PX0        ；INT0高优先级
SETB PX1        ；INT0高优先级
CLR PS          ；串行口低优先级
CLR PT0         ；定时器T0低优先级
CLR PT1         ；定时器T1低优先级
```

3.3.3　89C51的中断响应及处理过程

1. 中断响应条件

89C51单片机响应中断的条件是：中断源有请求，中断源开中断（IE相应位置1），CPU开中断（EA＝1）。这样，在每个机器周期内，89C51单片机对所有中断源都进行顺序检测，并可在任一个周期的S6期间找到所有有效的中断请求，对其优先级排队。89C51单片机在紧接着的下一个机器周期S1期间响应中断，条件是：

① 无同级或高级中断正在服务；

② 现行指令执行到最后一个机器周期且已结束；

③ 现行指令为RETI或需访问特殊功能寄存器IE或IP的指令时，执行完该指令且紧随其后的另一条指令也已执行完。否则，将丢弃中断查询的结果。

2. 中断处理过程

89C51单片机一旦响应中断，首先置位相应的优先级有效触发器，然后把断点地址压入堆栈保护，并将对应的中断入口地址值装入程序计数器PC，使程序转向该中断入口地址，以执行中断服务程序。

89C51单片机响应中断后，只保护断点而不保护现场（如累加器A、程序状态寄存器PSW的内容），且不能清除串行口中断标志TI和RI，也无法清除外部中断请求信号$\overline{\text{INT0}}$和$\overline{\text{INT1}}$，故用户在编制程序时应予以考虑。89C51单片机各中断源对应的中断入口地址见表3-2。

表 3-2 中断源对应的中断入口地址

中断源	中断入口地址
外部中断$\overline{\text{INT0}}$	0003H
定时器T0中断	000BH
外部中断$\overline{\text{INT1}}$	0013H
定时器T0中断	001BH
串行口中断	0023H

如前节所述，89C51单片机从相应的入口地址开始执行中断服务程序，直至遇到一条RETI指令为止。若用户在中断服务程序开始时安排了保护现场指令（相应寄存器内容压入堆栈），则在RETI指令前应有恢复现场指令（相应寄存器内容弹出堆栈）。

3. 中断响应时间

由上述可知，CPU不是在任何情况下都对中断请求立即响应，而且，不同情况对应的中断响应时间也不同。下面以外部中断为例，说明中断响应时间。

外部中断请求信号的电平在每个机器周期的S5P2期间经反相后锁存到IE0或IE1标志位，CPU在下一个机器周期才会查询到这些值，这时，如果满足响应条件，CPU响应中断时需执行一条两个机器周期的调用指令，以转到相应的中断服务程序入口。这样，从外部中断请求有效到开始执行中断服务程序的第一条指令，至少需要三个机器周期。

如果在申请中断时CPU正在处理最长指令（如乘、除法指令），则额外等待时间增加三个机器周期；若正在执行RETI或访问IE、IP指令，则额外等待时间又增加两个机器周期。

可以估算，若系统中只有一个中断源，则响应时间为3~8个机器周期。

3.4 89C51中断的应用编程

学习目标

1. 理解89C51的中断程序组织架构。
2. 掌握89C51的中断初始化内容设置。
3. 掌握89C51的中断服务程序设计要点。

导　入

89C51有五个中断源，任何一个中断源在今后的应用中，都离不开中断程序设计。这里给出一个最简单的中断源应用例子，譬如用89C51的外中断方式读按键，做到每按一次按键来改变一次LED灯由亮到灭或由灭到亮的状态，请读者不妨运用前面所学的知识先思考一

下相应的中断程序如何设计。这个程序设计虽说简单，但从设计中可以折射出自己对89C51中断技术应用的全面理解和掌握程度。

3.4.1 入口地址的使用技巧

89C51单片机复位后，主程序的入口地址是0000H（PC＝0000H），而五个中断源的入口地址分布在0003H~0023H区域中。由于各入口地址彼此靠得很近，通常情况下，难以直接容纳各自的程序，最好的办法就是在各自的入口地址单元中存放一条无条件转移指令（一般为长跳转指令LJMP），然后从跳转指令的目标地址处开始存放各自的程序，从而避免入口地址被其它程序占用。如图3-11所示为主程序和$\overline{\text{INT0}}$中断服务程序在存储器中的存放举例。

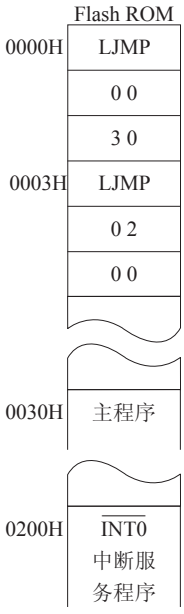


图 3-11 程序存放举例

3.4.2 中断的初始化内容

中断的初始化就是对与中断有关的四个寄存器做必要的设置。89C51单片机复位后，特殊功能寄存器IE和IP的内容均为00H，所有的中断源都处于关中状态及低优先级中断，因此要根据中断源的实际应用情况，对IE和IP进行初始化编程，以开放CPU中断，允许某些中断源中断和设置中断优先级；又如外部中断在使用过程中还要考虑中断触发方式选择位的设置等。

89C51单片机复位后首先执行的是主程序，中断的初始化必须在主程序中完成。

3.4.3 中断服务程序的设计要点

中断服务程序设计的相关知识在前面的章节中有比较详细的介绍，这里再强调一下设计上的几个要点。

- 确定是否需要保护现场。
- 及时清除那些不能被硬件自动清除的中断请求标志，以免产生错误的中断。
- 中断服务程序中的压栈（PUSH）与弹栈（POP）指令必须成对使用，以确保中断服务程序的正确返回。
- 主程序和中断服务程序之间的参数传递与主程序和子程序之间的参数传递方式相同。

【例3-3】 如图3-12所示，用外部中断$\overline{\text{INT0}}$引脚输入按键信号，P1.0接发光二极管LED。要求每次按动按键，外接LED改变一次亮灭状态。采用$\overline{\text{INT0}}$中断方式编制程序。

程序设计1：跳变触发：每次跳变引起一次中断请求。

```
        ORG     0000H          ；复位入口
        AJMP    MAIN
        ORG     0003H          ；中断入口
        AJMP    PINT0
        ORG     0030H          ；主程序
```

```
MAIN：  MOV     SP，#40H        ；设栈底
        SETB    EA              ；开总允许开关
        SETB    EX0             ；开INT0中断
        SETB    IT0             ；负跳变触发中断
H：     SJMP    H               ；执行其它任务
        ORG     0200H           ；中断服务程序
PINT0：  CPL     P1.0            ；改变LED
        RETI                    ；返回主程序
        END
```

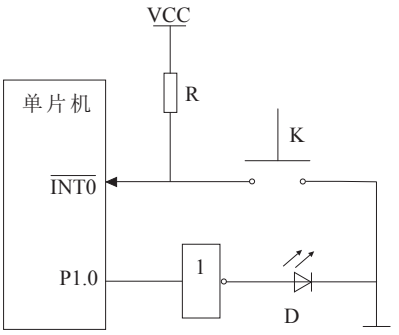


图 3-12 硬件线路连接

程序设计2：电平触发：为避免一次按键引起多次中断响应，软件要等待按键释放。

```
        ORG     0000H           ；复位入口
        AJMP    MAIN
        ORG     0003H           ；中断入口
        AJMP    PINT0
        ORG     0030H           ；主程序
MAIN：  MOV     SP，#40H        ；设栈底
        SETB    EA              ；开总允许开关
        SETB    EX0             ；开INT0中断
        CLR     IT0             ；低电平触发中断
H：     SJMP    H               ；执行其它任务
        ORG     0200H           ；中断服务程序
PINT0：  CPL     P1.0            ；改变LED
WAIT：  JNB     P3.2，WAIT      ；等按键释放
        RETI                    ；返回主程序
        END
```

知识拓展

在实际应用中，如果遇到89C51单片机两个外部中断源（$\overline{\text{INT0}}$和$\overline{\text{INT1}}$）不够使用的情况，可以想办法实施扩展，请给出扩展硬件线路和扩展后的外部中断源的识别程序，并说明扩展后的外部中断源的中断优先级是如何确定的。

本章小结

引入查询传送的根本原因是CPU与外设工作不同步。引入中断传送的根本原因是查询传送时CPU的工作效率低。正是因为有了“中断”，才使计算机的工作更加灵活，效率更高。

89C51单片机有五个中断源，其中三个属内部中断（定时器T0、定时器T1和串行口），另两个属外部中断（$\overline{\text{INT0}}$和$\overline{\text{INT1}}$）。所谓外部中断，就是在外部引脚上施加产生中断所需要的请求信号而引起的中断。

五个中断源各自有固定的中断服务程序的入口地址。当CPU响应任何一个中断源的中断后，单片机内部硬件会自动寻找到相应的入口地址，并送入PC中，使CPU执行中断服务程序。在汇编程序中，中断服务程序应存放在正确的入口地址内（或存放一条转移指令）。

89C51单片机的中断是靠内部四个寄存器（TCON、SCON、IE和IP）管理和控制的。要注意的是由于五个中断源全部属于可屏蔽中断源，只有在CPU开中（EA=1)和中断源自身的中断开关置“1”的情况下，CPU才能响应该中断源的中断请求，也就是所谓的“两级开通”控制。

从程序表面看来，主程序和中断服务程序好像是没有关联的，只有掌握中断响应的过程，才能理解中断的发生和返回，看得懂中断程序并能编写高质量中断程序。

思考与练习题

一、单项选择题

1. 89C51单片机用来设置中断优先级的控制寄存器是（　）。

A. IP　　B. TCON　　C. IE　　D. SCON

2. 外部中断INT0的触发方式控制位IT0置0后，其有效的中断触发信号是（　）。

A. 高电平　　B. 低电平　　C. 上升沿　　D. 下降沿

3. 外部中断INT0开中的初始化执行指令是（　）。

A. SETB ET0　　B. SETB EX0

C. SETB ET0 和SETB EA　　D. SETB EX0 和SETB EA

4. 外部中断INT1的中断服务程序入口地址是（　）。

A. 0003H　　B. 000BH　　C. 0013H　　D. 001BH

5. 89C51单片机在响应中断后，需要用软件来清除的中断标志是（　）。

A. TF0、TF1　　B. RI、TI　　C. IE0、IE1　　D. EX0、EX1

二、判断题

1. 在微机的输入/输出方式中，中断请求方式比查询方式效率高。（　）
2. 多个中断源不可以同时申请中断。（　）
3. 只要EA=1，那么中断请求就一定能够得到响应。（　）
4. 同级中断可以嵌套。（　）
5. 中断服务程序执行的最后一条指令必须是RETI。（　）

三、问答题

1. 什么是中断？引入中断的根本原因是什么？中断能实现哪些功能？
2. 89C51有几个中断源？各中断标志是怎样产生的？
3. 中断响应过程中，为什么通常要保护现场？如何保护？
4. 当89C51正在执行某一中断源的中断服务程序时，如果有新的中断请求出现，试问在什么情况下可响应新的中断请求？在什么情况下不能响应新的中断请求？
5. 试编写一段对中断系统初始化的程序，使其允许$\overline{\text{INT0}}$、$\overline{\text{INT1}}$、T0和串行口中断，且使串行口中断为高优先级中断。

实验项目3　外部中断的使用及编程方法

一、实验目的

1. 进一步理解中断的基本概念，加深对89C51中断原理的认识。
2. 通过外部中断的基本应用实验，牢固掌握89C51中断应用的编程架构及程序设计要点。
3. 进一步掌握Keil μVision2 集成开发环境的使用。
4. 熟悉STC_ISP_V480在系统编程软件的操作。
5. 在实践中，切实提高单片机应用的操作技能和动手能力，独立思考问题、分析问题和解决问题的能力。

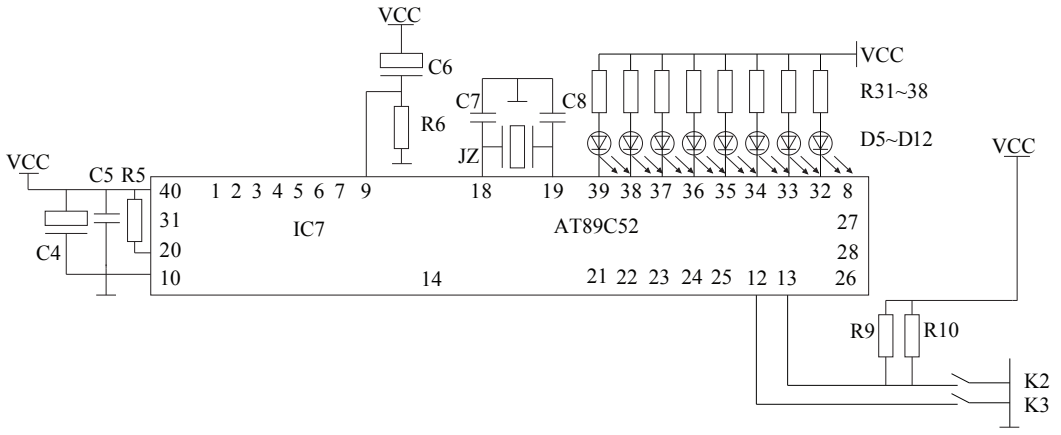
二、实验设备

1. PC计算机一台
2. 单片机实验装置一套（教材配套）
3. Keil μVision2 集成开发环境
4. STC_ISP_V480在系统编程软件

三、实验内容

用外部中断$\overline{\text{INT0}}$引脚输入K3按键信号，P0.0~ P0.7接八个发光二极管。要求每次按动K3按键，使P0口外接八个发光二极管改变一次亮灭状态。采用$\overline{\text{INT0}}$中断方式编制程序。

1. 实验线路



2. 实验程序

程序设计1：跳变触发：每次跳变引起一次中断请求。

```
        ORG     0000H
        AJMP    MAIN
```

```
        ORG     0003H
        AJMP    PINT0
        ORG     0030H
MAIN:   MOV     SP，#40H
        MOV     P0，#0FFH
        SETB    EA
        SETB    EX0
        SETB    IT0
H:      SJMP    H
        ORG     0200H
PINT0:  CPL  P0.0
        CPL  P0.1
        CPL  P0.2
        CPL  P0.3
        CPL  P0.4
        CPL  P0.5
        CPL  P0.6
        CPL  P0.7
        RETI
        END
```

程序设计2：电平触发：为避免一次按键引起多次中断响应，软件要等待按键释放。

```
        ORG     0000H
        AJMP    MAIN
        ORG     0003H
        AJMP    PINT0
        ORG     0030H
MAIN:   MOV     SP，#40H
        SETB    EA
        SETB    EX0
        CLR     IT0
H:      SJMP    H
        ORG     0200H
PINT0:  CPL  P1.0
        CPL  P0.1
        CPL  P0.2
        CPL  P0.3
        CPL  P0.4
        CPL  P0.5
        CPL  P0.6
```

```
        CPL  P0.7
WAIT：  JNB      P3.2，WAIT
        RETI
        END
```

四、实验操作

1. 熟悉实验线路的基本工作原理。
2. 在理解源程序的基础上，依次完成两个实验程序的运行。
3. 打开Keil μVision2 集成开发环境，对源程序进行编辑，然后汇编直至通过，最终生存HEX文件。
4. 完成PC机和单片机实验装置的连接，打开STC_ISP_V480在系统编程软件，将HEX文件下载到单片机中。
5. 运行程序，然后操作K3按键并观察LED显示结果。若符合程序功能的设计要求，表明实验成功。
6. 试用外部中断$\overline{\text{INT1}}$引脚输入K2按键信号，要求在原实验程序基础上完成必要的修改后，然后运行程序，并操作K2按键观察LED显示结果。

第4章 定时器及应用

单片机在实际应用中，经常会遇到定时和计数方面的要求，若用软件来实现，必然会降低CPU的工作效率。89C51单片机内部集成了两个16位的定时器，用来实现定时或计数，使定时和计数工作不会影响CPU的使用效率，从而简化系统的设计。本章重点介绍89C51单片机定时器的结构、原理、控制和工作方式的设置，以及定时器的基本编程应用方法。

4.1 89C51定时器的结构和工作原理

学习目标

1. 熟悉可编程计数器/定时器的基本组成和工作原理。
2. 掌握89C51定时器的组成结构和工作原理。
3. 掌握89C51定时器的四种工作方式控制和应用特点。

导　入

在日常生活中，经常会遇到许多计数和定时上的应用要求。从计数方面来讲，如生产线上的产品计数、汽车上的里程表计数、家用电度表计数等等；从定时方面来讲，如定时采样、实时控制、电视机定时关机、空调定时开关等等。有关计数和定时的应用例子不胜枚举，正是为了满足这方面广泛的应用需求，在单片机内部设置了计数器/定时器部件。

4.1.1 实现计数与定时的基本方法

实现计数与定时的基本方法有三种：

1. 完全硬件方式

过去，许多仪器仪表或设备需要进行延时、定时或计数经常使用数字逻辑电路实现，即完全用硬件电路实现计数/定时功能，若要改变计数/定时的要求，必须改变电路参数，但其通用性和灵活性差。微机出现以后，特别是随着单片机的发展和普及，这种完全硬件方式实现定时与计数的方法已较少使用。

2. 完全软件方式

在计算机中，通过编程，利用计算机执行指令的时间实现定时，称为完全软件方式，简称软件方式。在这种方式中，一般是根据所需要的时间常数来设计一个延时子程序，延时子程序中包含一定的指令，设计者要对这些指令的执行时间进行严密的计算或者精确的测试，以便确定延时时间是否符合要求。当时间常数比较大时，常常将延时子程序设计为一个循环程序，通过循环常数和循环体内的指令来确定延时时间。这样，每当延时子程序结束以后，可以直接转入下面的操作，也可以用输出指令产生一个信号作为定时输出。这种方法的优点是节省硬件，主要缺点是执行延时程序期间，CPU一直被占用，降低了CPU的效率，也不容易提供多作业环境；另外，设计延时子程序时，要用指令执行时间来拼凑延时时间，显得比较麻烦。不过，这种方法在实际应用中还是经常使用的，如在已有系统上作软件开发，且延时时间较小而重复次数又较少时，常用软件方式定时。在计算机控制软件开发过程中，作为粗略的延时，经常使用这种方法。

3. 可编程计数器/定时器

利用专门的可编程计数器/定时器实现计数与定时，克服了完全硬件方式和完全软件方式的缺点，综合了它们的优点，其计数/定时功能可由程序灵活地设置，设定之后与CPU并行工作，不占用CPU的工作时间。应用可编程计数器/定时器，在简单的软件控制下，可以产生准确的延时时间。这种方法是根据需要的定时时间，用指令对计数器/定时器设置定时常数，并用指令启动计数器/定时器。计数器/定时器开始计数，当计到确定值时，便自动产生一个定时输出。在计数器/定时器开始工作以后，CPU可以做其他工作而不必去管它。这种方法最突出的优点是计数时不占用CPU的时间，并且，如果利用计数器/定时器产生的溢出信号作为中断请求信号，就可以建立多作业的环境，所以，可大大提高CPU的利用率。加上计数器/定时器本身的开销并不很大，因此，这种方法在单片机应用系统中被广泛使用。

先介绍一下可编程计数器/定时器的基本组成和工作原理，为进一步学习89C51定时器原理和应用做铺垫。

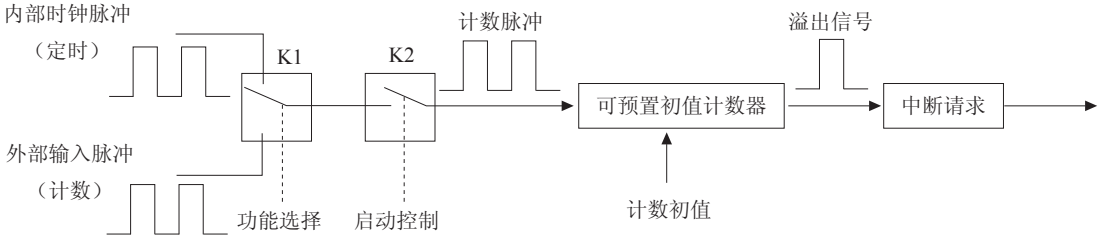


图 4-1 可编程计数器/定时器的逻辑结构

可编程计数器/定时器的逻辑结构如图4-1所示。其核心部件是可预置初值的计数器，当计数器被预置初值后，并使启动控制开关K2闭合，计数器就在所预置的初值基础上开始对输入计数脉冲进行计数，直至计数器产生计数溢出信号。该信号可用来向CPU申请中断，要求CPU进行中断功能处理。另外，可编程计数器/定时器可以通过功能选择开关K1的切换实现两种工作方式，一种叫计数器方式，另一种叫定时器方式。值得一提的是两种工作方式的本质是相同的，它们都是对一个输入脉冲进行计数。

- 计数器工作方式　将图4-1中的开关K1切换到下方时，选择的是计数器工作方式。此时计数器对外部事件进行计数，如对外部随机输入脉冲的个数进行计数，最后根据计数器内容的变化，可以知道外部脉冲的数量。
- 定时器工作方式　将图4-1中的开关K1切换到上方时，选择的是定时器工作方式。此时计数器对内部时钟脉冲进行计数，由于输入脉冲的频率是一定的，则记录一定个数的脉冲，其所需的时间是一定的，从而达到定时的目的。例如，输入脉冲的频率为2MHz，则计满2×10^6个脉冲对应的定时时间为1s。

【例4-1】 设可编程计数器/定时器中的可预置初值的计数器为8位加1计数器。按要求选择功能和初值。

① 要求检测到100个脉冲，发出中断请求（计数器产生溢出），通知CPU。

答：选计数功能，计数初值$X=2^8-100=156$。

② 要求每隔200μs时间，发一次中断请求。设内部时钟周期1μs。

答：选定时功能，计数初值为$X=2^8-(200\ \mu s / 1\mu s)=56$。

4.1.2 89C51定时器的结构

89C51单片机内部有两个16位的可编程计数器/定时器（简称定时器），称为定时器0（T0）和定时器1（T1），通过软件设置可选择其作为计数器用或作为定时器用。此外，工作方式、定时时间、计数值、启动、中断请求等都可以由程序设定，其逻辑结构如图4-2所示。

由图4-2可知，89C51定时器由定时器0、定时器1、定时器方式寄存器TMOD和定时器控制寄存器TCON组成。

定时器0、定时器1都采用16位加法计数器，分别由两个8位专用寄存器组成，定时器0由TH0和TL0组成，定时器1由TH1和TL1组成。TL0、TL1、TH0、TH1均可单独访问。定时器0或定时器1用作计数器时，对芯片引脚T0（P3.4）或T1（P3.5）上输入的脉冲计数，每输入一个脉冲，加法计数器加1；其用作定时器时，对内部机器周期脉冲计数，由于机器周期是定值，因此当计数值确定时，时间也随之确定。

TMOD、TCON与定时器0、定时器1之间通过内部总线及逻辑电路连接，TMOD用于设置定时器的工作方式，TCON用于控制定时器的启动与停止。

当定时器设置为计数工作方式时，计数器对来自输入引脚T0（P3.4）和T1（P3.5）的外部信号计数，外部脉冲的下降沿将触发计数。在每个机器周期的S5P2期间采样引脚输入电平，若前一个机器周期采样值为1，后一个机器周期采样值为0，则计数器加1。新的计数值是在检测到输入引脚电平发生1到0的负跳变后，在下一个机器周期的S3P1期间装入计数器中的，可见，检测一个由1到0的负跳变需要两个机器周期，所以，最高检测频率为振荡

频率的1/24。计数器对外部输入信号的占空比没有特别的限制，但必须保证输入信号的高电平与低电平的持续时间在一个机器周期以上。

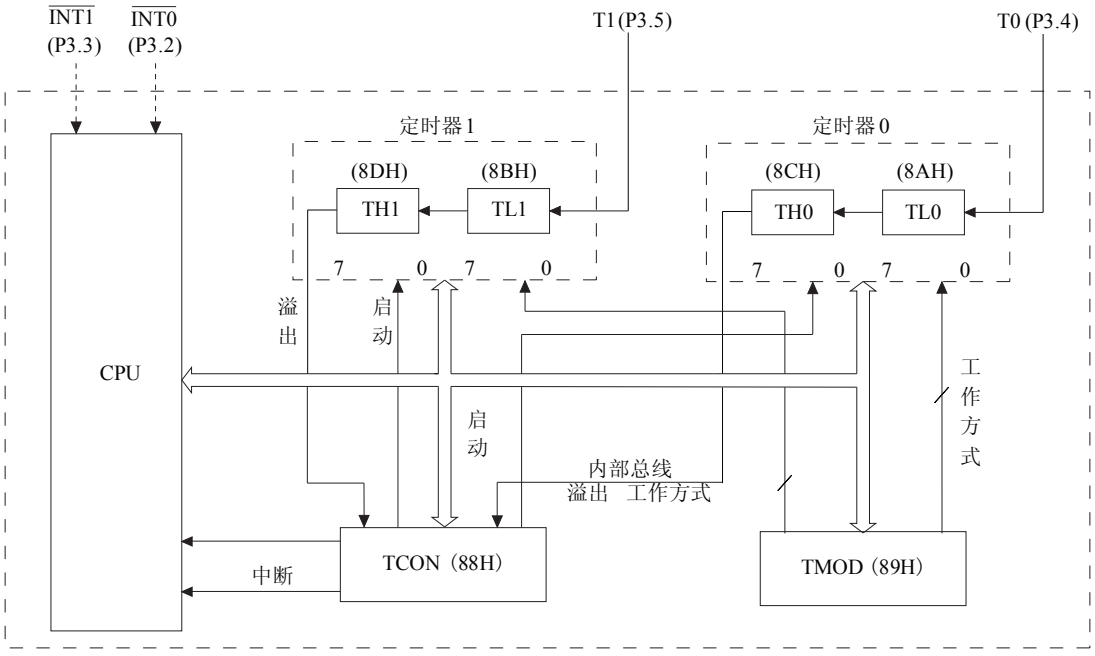


图 4-2 89C51定时器逻辑结构

当定时器设置为定时工作方式时，计数器对内部机器周期计数，每过一个机器周期，计数器增1，直至计满溢出。定时器的定时时间与系统的振荡频率紧密相关，由于89C51单片机的一个机器周期由十二个振荡脉冲组成，所以它的计数速率是振荡频率的1/12。如果单片机系统采用12 MHz晶振，则计数周期为1μs，这是最短的定时周期，适当选择定时器的初值可获取各种定时时间。

当设置了定时器的工作方式并启动定时器工作后，定时器就按被设定的工作方式独立工作，不再占用CPU的操作时间，只有在计数器计满溢出时才可能中断CPU当前的操作。

4.1.3 89C51定时器的控制

在启动定时器工作之前，必须将一些命令（称为控制字）写入定时器中，这个过程称为定时器的初始化。定时器的初始化是通过定时器方式寄存器TMOD和控制寄存器TCON的设置来完成的。

1. 定时器方式寄存器TMOD

TMOD为定时器0、定时器1的工作方式寄存器，其各位的定义格式如图4-3所示。

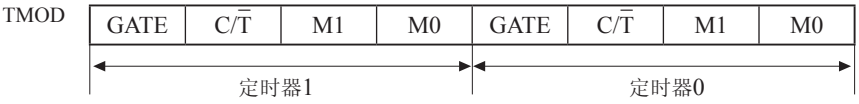


图 4-3 TMOD的位定义

TMOD的低4位为T0的方式字段，高4位为T1的方式字段，它们的含义完全相同。

- GATE：门控位。

GATE =0为非门控方式（又称内部启动），只要软件控制位TR0或TR1置1即可启动定时器。GATE=1为门控方式（又称外部启动），不但软件控制位TR0或TR1须置1，同时还须（P3.2引脚）或（P3.3引脚）为高电平方可启动定时器。

- C/$\overline{T}$：功能选择位。

  C/$\overline{T}$ =0：选择定时功能，计数内部机器周期脉冲。

  C/$\overline{T}$ =1：选择计数功能，计数T0（P3.4引脚）或T1（P3.5引脚）引脚输入的外部脉冲。
- M1和M0：方式选择位。2位有四种编码，对应于四种工作方式。见表4-1。

表 4-1 M1和M0控制的4种工作方式

M1	M0	方 式	功 能 说 明
0	0	0	13 位计数器（TH的8位和TL的低5位）
0	1	1	16 计数器 （TH的8位和TL的8位）
1	0	2	自动重装入初值的8位计数器
1	1	3	T0 分成两个独立的8位计数器，T1 在方式3时停止工作

TMOD不能位寻址，只能用字节指令设置定时器工作方式，高4位定义定时器T1，低4位定义定时器T0。复位时，TMOD所有位均置0。

确定定时器工作方式指令：MOV TMOD，#控制字。

【例4-2】 设T0用方式2非门控定时，T1用方式1门控计数，按此要求设置TMOD。

答：MOV TMOD，#11010010B

2. 定时器控制寄存器**TCON**

TCON的作用是控制定时器的启动、停止，标志定时器的溢出和中断情况。其各位的定义格式如图4-4所示。

TCON	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

图 4-4 TCON的位定义

各位含义如下：

- TR0和TR1：定时器运行控制位。

  TR0 =0，停止T0工作； TR0 =1，启动T0工作。

  TR1 =0，停止T1工作； TR1 =1，启动T1工作。

【例4-3】 启动定时器T0计数。

答：SETB TR0

注意：当GATE=0时，TR0/TR1置1即可启动定时器；当GATE=1时，TR0/TR1置1且引脚P3.2/P3.3为高电平时启动定时器。

- TF0和TF1：溢出中断标志位

  定时器T0计满数产生溢出时，由硬件自动置TF0=1。在中断允许时，向CPU发出定时器T0的中断请求，进入中断服务程序后，由硬件自动清0。在中断屏蔽时，TF0可作查询测 试用，此时只能由软件清0。TF1定时器T1的溢出标志位。其功能及操作情况同TF0。
- TCON中的低4位用于控制外部中断（前一章中已做过介绍），与定时器无关。

  TCON可以位寻址，清溢出标志位或启动定时器都可以用位操作指令。89C51复位

时，TCON的所有位均清0。

4.1.4 89C51定时器的四种工作方式

由前述内容可知，通过对TMOD寄存器中M0、M1位进行设置有四种工作方式，下面逐一进行论述。

1. *方式0*

方式0构成一个13位定时器。图4-5是定时器0在方式0时的逻辑电路结构，定时器1的结构和操作与定时器0完全相同。

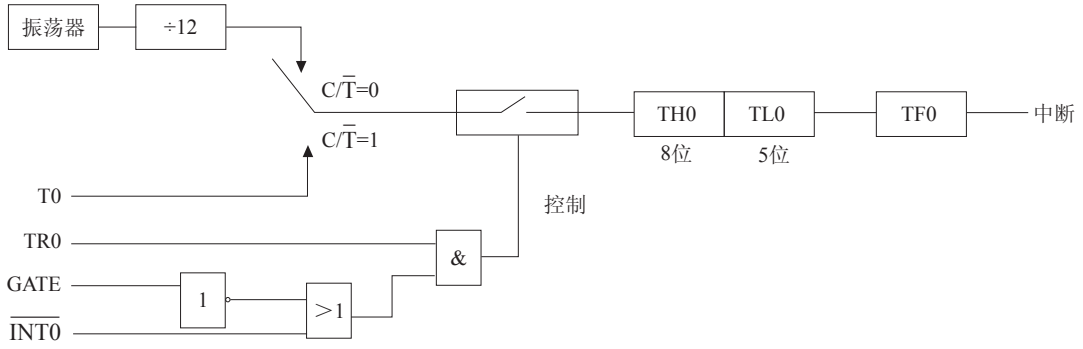


图 4-5　定时器0（或定时器1）在方式0时的逻辑电路结构

由图4-5可知：16位加法计数器（TH0和TL0）只用了13位。其中，TH0占高8位，TL0占低5位（只用低5位，高3位未用）。当TL0低5位溢出时自动向TH0进位，而TH0溢出时向中断标志位TF0进位（硬件自动置位），并申请中断。

当C/$\overline{T}$=0时，多路开关连接12分频器输出，定时器T0对机器周期计数，此时，定时器0为定时器功能。

当C/$\overline{T}$=1时，多路开关与T0（P3.4）相连，外部计数脉冲由T0脚输入，当外部信号电平发生由1到0的负跳变时，计数器加1，此时，定时器0为计数器功能。

当GATE=0时，或门被封锁，信号无效。或门输出常1，打开与门，TR0直接控制定时器0的启动和关闭。TR0=1，接通控制开关，定时器0从初值开始计数直至溢出。溢出时16位加计数器为0，TF0置位，并申请中断。TR0=0，则与门被封锁，控制开关被关断，停止计数。

当GATE=1时，与门的输出由$\overline{INT0}$（P3.2）的输入电平和TR0位两者的状态来确定。若TR0 = 1则与门打开，通过引脚$\overline{INT0}$（P3.2）电平直接开启或关断定时器T0，当引脚$\overline{INT0}$为高电平时，允许计数，否则停止计数；若TR0 = 0，则与门被封锁，控制开关被关断，停止计数。

方式0的计数器长度是13位，其最大计数值： $M=2^{13}=8192$。

2. *方式1*

定时器工作于方式1时，其逻辑结构图如图4-6所示。由图可知，方式1构成一个16位定时器，其结构与操作几乎完全与方式0相同，惟一差别是二者计数位数不同。

方式1的计数器长度是16位，其最大计数值：$M=2^{16}=65536$。

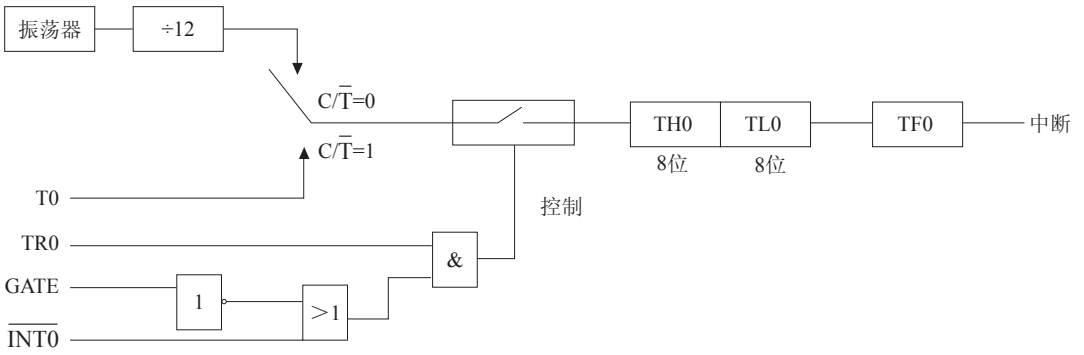


图 4-6 定时器0（或定时器1）在方式1时的逻辑电路结构

3. 方式2

定时器工作于方式2时，其逻辑结构图如图4-7所示。由图可知，方式2中，16位加法计数器的TH0和TL0具有不同功能，其中，TL0是8位计数器，TH0是重置初值的8位缓冲器。

方式0和方式1用于循环计数，在每次计满溢出后，计数器都复0，要进行新一轮计数还须重置计数初值。这不仅导致编程麻烦，而且影响定时时间精度。方式2具有初值自动装入功能，避免了上述缺陷，适合用作较精确的定时脉冲信号发生器。

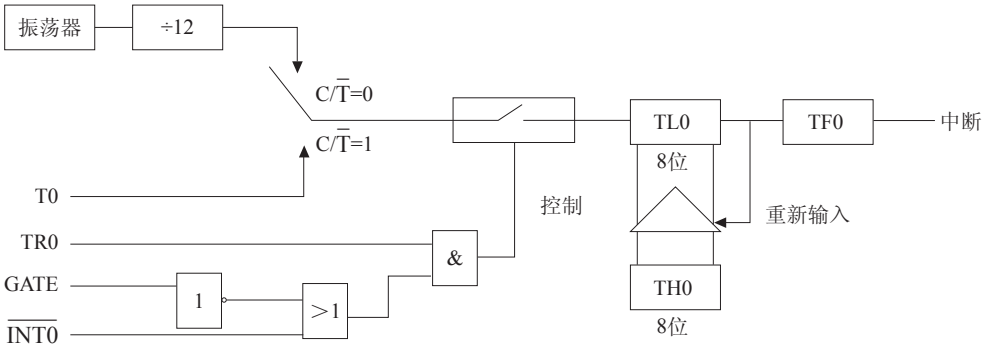


图 4-7 定时器0（或定时器1）在方式2时的逻辑结构

方式2中的16位加法计数器被分割为两个，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由软件赋予相同的初值。一旦TL0计数溢出，TF0将被置位，同时，TH0中的初值会自动装入TL0，从而进入新一轮计数，如此循环不止。

方式2的计数器长度是8位，其最大计数值：$M=2^8=256$ 。

4. 方式3 （仅T0有）

定时器0有方式3，而定时器T1在方式3时停止工作。定时器0工作于方式3时，其逻辑结构图如图4-8所示。

由图4-8可知，方式3时，定时器0被分解成两个独立的8位计数器TL0和TH0。其中，TL0占用原定时器0的控制位、引脚和中断源，即GATE、TR0、TF0和T0（P3.4）引脚、（P3.2）引脚。

除计数位数不同于方式0、方式1外，其功能和操作与方式0、方式1完全相同，可定时亦可计数。

TH0占用原定时器1的控制位TF1和TR1，同时还占用了定时器1的中断源，其启动和关闭仅受TR1置1或清0控制。TH0只能对机器周期进行计数，因此，TH0只能用作简单的内部

定时，不能用作对外部脉冲进行计数，它是定时器T0附加的一个8位定时器。

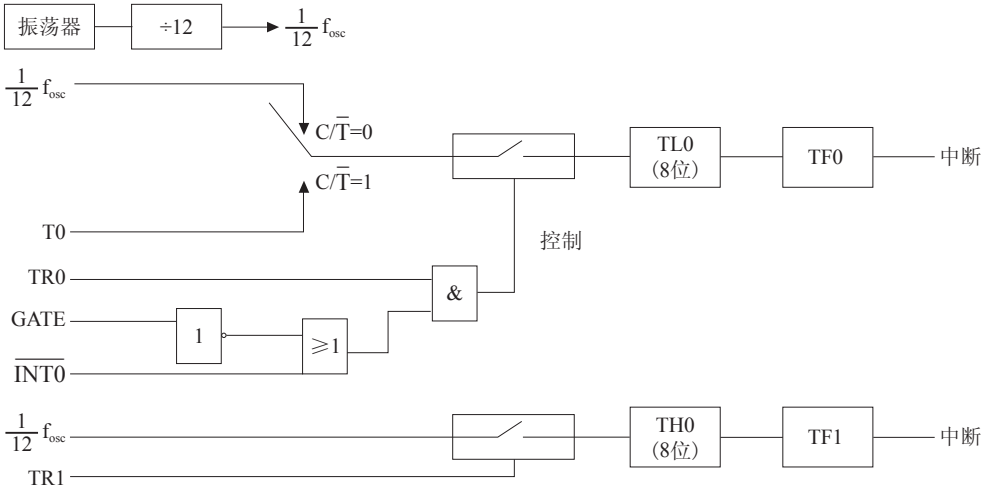


图 4-8　定时器0在方式3时的逻辑结构

方式3时，定时器1仍可设置为方式0、方式1或方式2。但由于TR1、TF1及T1的中断源已被定时器0占用，此时，定时器1仅由控制位切换其定时或计数功能，当计数器计满溢出时，只能将输出送往串行口。在这种情况下，定时器1一般用作串行口波特率发生器或不需要中断的场合。由于定时器1的TR1被占用，因此其启动和关闭较为特殊，当设置好工作方式时，定时器1可自动开始运行。若要停止操作，只需送入一个设置定时器1为方式3的指令即可。

4.2 89C51定时器的应用编程

学习目标

1. 掌握89C51定时器的初始化内容设置。
2. 掌握89C51定时器的初始化编程格式及溢出处理编程格式。
3. 掌握89C51定时器各种简单的应用及其程序设计。

导　入

89C51的定时器是一个可编程的定时器。因此，编程时要结合具体的应用要求来考虑如何选择其功能以及工作方式。除此之外，还要考虑初值设置、采用查询编程方式还是中断编程方式等等。这里给出一个最简单的定时器应用例子，譬如用89C51的定时器去定时控制LED灯由亮到灭或由灭到亮的状态。请学生不妨运用前面所学的知识试着对其控制程序进

行设计，先初步检验一下自己对89C51定时器应用的理解和掌握程度，并带着问题和思考去学习本节内容。

4.2.1 89C51定时器的初始化

由于定时器的功能是由软件编程确定的，所以，在使用前一般都要对其初始化。初始化步骤如下：

1. 确定控制字（对TMOD寄存器设置）

- 按实际应用需要来选择功能（定时功能或计数功能）。
- 按时间或计数长度需要来选择方式。
- 采用内部启动还是外部启动。

2. 预置定时或计数初值（直接将初值写入TH0、TL0或TH1、TL1）

① 计数初值计算

- 计数功能的计数初值：$X= 2^n$ －计数值。
- 定时功能的计数初值：$X= 2^n$ －t/MC。

式中 n为计数器长度（8/13/16）；t为定时时间；MC为机器周期（MC=12/fosc）。

② 预置初值指令

MOV TH0（TH1），#X_H

MOV TL0（TL1），#X_L

式中 X_H为计数初值的高8位；X_L为计数初值的低8位。

3. 根据需要开启定时器中断（对IE寄存器设置）

4. 启动定时器工作（将TR0或TR1置1）

GATE=0时，直接由软件置位TR0或TR1启动。

GATE=1时，除软件置位TR0或TR1外，还必须在外中断引脚$\overline{\text{INT0}}$或$\overline{\text{INT1}}$处加上高电平才能启动。

4.2.2 89C51定时器的编程格式

1. 初始化编程格式

```
MOV     TMOD，# 控制字      ；选择功能和方式
MOV     TH0（TH1），#XH     ；预置初值高8位
MOV     TL0（TL1），#XL     ；预置初值低8位
SETB    EA                  ；CPU开中断（若不需要开中断，该指令就去掉）
SETB    ET0（ET1）          ；定时器开中断（若不需要开中断，该指令就去掉）
SETB    TR0（TR1）          ；启动定时器
```

注意：方式0为13位，TL0（TL1）低5位有效；方式2为8位，TL0（TL1）=TH0（TH1）=X。

2. 溢出处理编程格式

① 查询方式：先查询定时器溢出标志，再进行溢出处理。

```
        ---                          ；定时器初始化
WAIT：  JBC  TF0（TF1），PT          ；检测溢出标志
        SJMP  WAIT
PT：    MOV  TH0（TH1），#XH         ；重装初值
        MOV  TL0（TL1），#XL
        ---                          ；溢出处理
        SJMP  WAIT
```

② 中断方式：初始化后执行其他任务，中断服务程序处理溢出。

```
        ORG   0000H
        LJMP  MAIN
        ORG   000BH（001BH）         ；T0（T1）中断入口
        LJMP  PTS
MAIN：  ---                          ；初始化后执行其他程序
PTS：   MOV   TH0（TH1），#XH        ；重装初值（中断服务程序）
        MOV   TL0（TL1），#XL
        ---                          ；溢出处理
        RETI
```

注意：上述方式2时，不必重装初值。

4.2.3 89C51定时器的应用编程

【例4-4】 利用定时器T0通过P1.0引脚输出周期为2ms的方波，设晶振频率为f_{osc}=12MHz。编制相应程序。

解：方波指的是每个周期内其高电平和低电平宽度相等，若要产生周期为2ms的方波，只要每隔1ms改变一次P1.0的输出状态，即将信号的幅值由0变到1或由1变到0，可采用取反指令CPL来实现。

确定TMOD控制字：对于定时器T0来说，$C/\overline{T}$=0（定时功能）、M1M0=01（方式1）、GATE=0（非门控，即内部启动）。定时器T1不用，取为全0。于是TMOD=00000001B=01H。

计算定时初值：$X = 2^n - t/MC$。其中，计数器长度n=16（方式1），定时时间t=1ms，机器周期$MC=12/f_{osc}$=12/12MHz=1μs。于是$X = 2^n - t/MC = 2^{16} - 1000\mu s / 1\mu s = 64536$=FC18H。

1. 查询方式编程

查询的对象是定时器T0的溢出标志TF0，在计数过程中TF0为0，当定时时间到，计数器溢出使TF0置1。由于未采用中断，TF0置1后不会自动清0，故需用指令使TF0清0。

程序如下：

```
        ORG   0000H
        LJMP  MAIN
        ORG   0030H
MAIN：  MOV   TMOD，#01H             ；置定时器T0为定时功能、方式1、非门控
        MOV   TH0，#0FCH             ；设置定时初值
```

```
        MOV   TL0，#18H
        SETB  TR0                ；启动T0定时
LOOP：  JBC   TF0，LOOP1         ；查询计数溢出
        SJMP  LOOP               ；TF0=0，则反复查询
LOOP1： CPL   P1.0               ；输出方波
        MOV   TH0，#0FCH         ；重新装入计数初值
        MOV   TL0，#18H
        SJMP  LOOP               ；重复循环
        END
```

2. 中断方式编程

为了提高CPU的效率，可采用定时中断的方式，每1ms产生一次中断，在中断服务程序中将输出信号取反即可。

程序如下：

```
        ORG   0000H
        LJMP  MAIN
        ORG   000BH              ；T0中断服务程序入口
        LJMP  INT                ；转至INT处
        ORG   0030H              ；主程序
MAIN：  MOV   TMOD，#01H         ；置定时器T0为定时功能、方式1、非门控
        MOV   TH0，#0FCH         ；设置定时初值
        MOV   TL0，#18H
        SETB  EA                 ；CPU开中断
        SETB  ET0                ；允许T0中断
        SETB  TR0                ；启动T0
        SJMP  $                  ；踏步等待中断
INT：   CPL   P1.0               ；输出方波
        MOV   TH0，#0FCH         ；重新装入计数初值
        MOV   TL0，#18H
        RETI                     ；中断返回
        END
```

注意：当采用方式0、1、3时，只要不关闭定时器，那么，每当计数器溢出时都会归0，都需要重新装入计数初值，以保证定时计数初值不变。

还有一点要说明：采用中断方式编程，主程序在定时器初始化之后，踏步等待中断一般只用于例题程序，而在实际系统中则很少采用，因为在这种方式下CPU效率很低。

【例4-5】 在P3.4（T0）引脚上输入脉冲信号，要求每发生一次负跳变时，从P1.0引脚上输出一个500μs的同步脉冲，设晶振频率为6 MHz。编制相应程序。

解：初始将定时器T0设定为计数功能、方式2及计数初值为FFH。当P3.4引脚上的电平发生负跳变时，定时器T0计数器加1，立刻产生溢出使TF0标志置1。然后把定时器T0改变成定时功能、方式2及定时时间为500μs，并使P1.0输出由1变为0，定时器T0定时到产生溢出使P1.0引脚恢复输出高电平，再重新将定时器T0恢复初始的计数功能，如此循环下去，

使输出波形如图4-9所示。

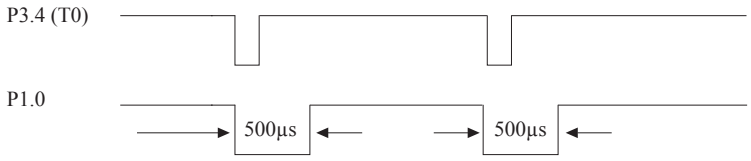


图 4-9 波形示意图

按上述设计方法，定时器T0初值计算如下：

计数初值X = 2^n－计数值=2^8－1=255=FFH。

定时初值X = 2^n－t/MC=2^8－500μs/2μs=6=06H。

采用查询方式编程，程序如下：

```
        ORG   0000H
        LJMP  MAIN
        ORG   0030H
MAIN:   MOV   TMOD，06H   ；设置T0为计数功能，方式2
        MOV   TL0，#0FFH  ；设置T0计数初值FFH
        MOV   TH0，#0FFH
        SETB  TR0         ；启动T0计数
LOOP1:  JBC   TF0，RESP1  ；查询溢出标志，TF0=1（P3.4有负跳变）转RESP1
        SJMP  LOOP1       ；TF0=0，则反复查询
RESP1:  CLR   TR0         ；停止计数
        MOV   TMOD，#02H  ；设置T0为定时功能，方式2
        MOV   TL0，#06H   ；设置T0 定时初值06H（500μs）
        MOV   TH0，#06H
        CLR   P1.0        ；P1.0清0
        SFTB  TR0         ；启动定时500μs
LOOP2:  JBC   TF0，RESP2  ；查询溢出标志，TF0=1（500μs时间到）转RESP2
        SJMP  LOOP2       ；TF0=0，则反复查询
RESP2:  SETB  Pl.0        ；Pl.0置1
        CLR   TR0         ；停止计数
        SJMP  MAIN
        END
```

【例4-6】 将定时器T0工作于方式3，此时TL0和TH0作为两个独立的8位定时器使用。要求TL0使P1.0产生400μs的方波，TH0使P1.1产生800μs的方波。设晶振频率为6MHz。编制相应程序。

解：当采用方式3时，对于TH0来说，需要借用定时器T1的控制信号及中断资源。

计算初值：

TL0的定时初值X = 2^n－t/MC=2^8－200μs /2μs=156=9CH。

TH0的定时初值X = 2^n－t/MC=2^8－400μs/2μs=56=38H。

采用中断方式编程，程序如下：

```
        ORG    0000H
        LJMP   MAIN
        ORG    000BH                ；TL0中断服务程序入口
        LJMP   INTTL0
        ORG    001BH                ；TH0中断服务程序入口
        LJMP   INTTH0
        ORG    0030H                ；主程序
MAIN:   MOV    TMOD，#03H           ；设置T0定时功能、方式3
        MOV    TL0，#9CH            ；置TL0定时初值（200μs）
        MOV    TH0，#38H            ；置TH0定时初值（400μs）
        SETB   EA                   ；CPU开中断
        SETB   ET0                  ；允许T0中断（用于TL0）
        SETB   ET1                  ；允许T1中断（用于TH0）
        SETB   TR0                  ；启动TL0
        SETB   TR1                  ；启动TH0
HALT:   SJMP   HALT                 ；踏步等待中断
INTTL0: CPL    P1.0                 ；P1.0取反
        MOV    TL0，#9CH            ；重新装入定时初值
        RETI                        ；中断返回
INTTH0: CPL    P1.1                 ；P1.1取反
        MOV    TH0，#38H            ；重新装入定时初值
        RETI                        ；中断返回
        END
```

【例4-7】 在P1.7引脚上接一个发光二极管LED，要求利用定时器控制，使LED周而复始地亮一秒灭一秒，设晶振频率为f_{osc}=6MHz。编制相应程序。

解：在定时器的四种工作方式中，方式1的定时时间最长，按晶振频率为f_{osc}=6MHz计算，方式1的最大定时时间为$2^{16}\times 2\mu s$ =131. 072ms，显然不能满足定时1s要求。这种情况称之为长定时，长定时可以通过增加一个硬件计数器或一个软件计数器两种方法解决。

方法**1**：由T0负责在P1.0引脚上产生100ms的方波，T1（作硬件计数器使用）计数P1.0引脚上的负跳边次数，计满10个跳边恰好为1s，每隔1s对P1.7引脚取反一次。硬件线路连接如图4-10所示。

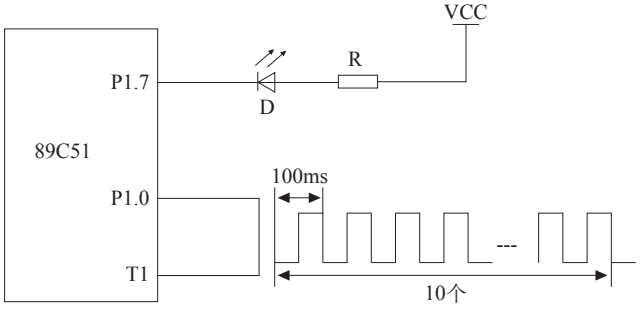


图 4-10 方法1

定时器T0采用定时功能及方式1，定时初值X＝2^n－t/MC＝2^{16}－50000μs/2μs ＝40536=9E58H。
定时器T1采用计数功能及方式2，计数初值X= 2^n－计数值=2^8－10=246=F6H。
采用查询方式编程，程序如下：

```
        ORG   0000H
        LJMP  MAIN
        ORG   0030H
MAIN:   MOV   TMOD，#61H  ；设置T1计数功能、方式2，T0定时功能、方式1
        MOV   TL1，#0F6H  ；设置T1计数初值
        MOV   TH1，#0F6H
        CLR   P1.0        ；P1.0清0
        SETB  TR1         ；启动T1
LOOP1:  CPL   P1.7        ；1s到，P1.7输出取反
LOOP2:  MOV   TL0，#58H   ；设置T0定时初值（50ms）
        MOV   TH0，#9EH
        SETB  TR0         ；启动T0
LOOP3:  JBC   TF0，LOOP4  ；查询溢出标志，TF0=1（50ms时间到）转LOOP4
        SJMP  LOOP3       ；TF0=0，则反复查询
LOOP4:  CPL   P1.0        ；P1.0清0
        JBC   TF1，LOOP1  ；T1计满10个脉冲就溢出（用时1s），转LOOP1
        SJMP  LOOP2       ；T1未计满10个脉冲，转LOOP2
        END
```

方法**2**：定时器T0定时l00ms，每溢出一次计数单元R0（作软件计数器使用）增1，计满10个（恰好为1s）对P1.7脚取反一次。硬件线路连接如图4-11所示。

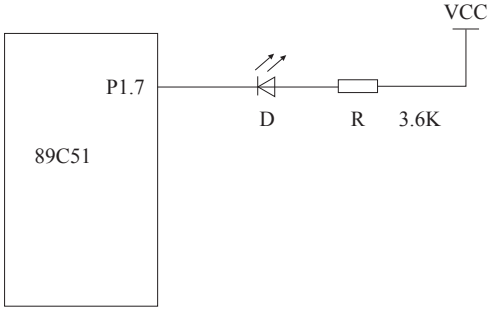


图 4-11 方法2

定时器T0采用定时功能及方式1，定时初值X＝2^n－t/MC=2^{16}－100000μs/2μs= 15536=3CB0H。
采用中断方式编程，程序如下：

```
        ORG   0000H
        LJMP  MAIN
        ORG   000BH
        LJMP  INTT0
        ORG   0030H
MAIN:   MOV   TMOD，#01H  ；设置T0定时功能、方式1
```

```
        MOV   TH0，# 3CH          ；设置T0定时初值（100ms）
        MOV   TL0，#0B0H
        MOV   R0，#0              ；软件计数器R0清0
        SETB  EA                  ；CPU开中断
        SETB  ET0                 ；开T0中断
        SETB  TR0                 ；启动T0
        SJMP  $
INTT0：INC    R0                  ；每中断一次计数单元R0增1
        CJNE  R0，#10，LOOP       ；R0未计满10，转LOOP
        MOV   R0，#0              ；R0清0
        CPL   P1.7                ；R0计满10（1s到），P1.7取反
LOOP：  MOV   TH0，#3CH           ；重装T0定时初值（100ms）
        MOV   TL0，#0B0H
        RETI                      ；中断返回
        END
```

【例4-8】 利用定时器T1的门控位GATE测试$\overline{\text{INT1}}$引脚上出现的正脉冲宽度，设晶振频率为f_{osc}=12MHz。编制相应程序。

解：被测脉冲信号由$\overline{\text{INT1}}$（P3.3）引脚输人，对于定时器T1来讲，C/T=0（定时功能）、M1M0=01（方式1）、GATE=1（门控，即外部启动），定时器T0不用，取为全0。于是TMOD=10010000B=90H。

测试时，应在$\overline{\text{INT1}}$低电平时，设置TR1为1；当$\overline{\text{INT1}}$变为高电平时，就启动计数，$\overline{\text{INT1}}$再次变低时，停止计数。此时的计数值与机器周期的乘积即为被测正脉冲的宽度。测试过程如图4-12所示。

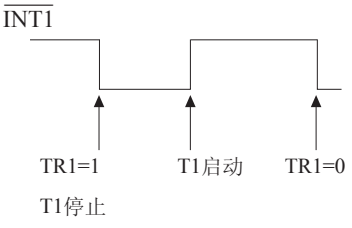


图 4-12 测试过程示意图

由于晶振频率为f_{osc}=12MHz时，定时器T1工作于方式1的最长定时时间为65.536ms。因此分两种情况考虑程序设计。程序如下：

1. 设被测脉宽小于65.536ms

采用查询方式编程，等待查询$\overline{\text{INT1}}$，正脉冲过后，读出TH1和TL1。

```
        ORG   0000H
        LJMP  MAIN
        ORG   0030H
MAIN：  MOV   TMOD，#90H  ；设置T1定时功能、方式1、门控启动
        MOV   TL1，#0     ；初值为0（最大计时时间65.536ms）
        MOV   TH1，#0     ；自动重装初值为0
WAIT1： JB    P3.3，WAIT1 ；等待低电平到来
        SETB  TR1         ；启动T1
WAIT2： JNB   P3.3，WAIT2 ；等待正脉冲到来（正脉冲一到，结束等待并开始计时）
WAIT3： JB    P3.3，WAIT3 ；等待正脉冲结束
        CLR   TR1         ；关闭T1
```

```
        MOV   R2，TL1        ；读出T1
        MOV   R3，TH1
        SJMP  $              ；踏步
        END
```

脉宽算式如下：

脉宽t＝（R3 R2）×MC（机器周期）＝（R3 R2）×1μs

2. 设被测脉宽大于65.536ms

采用中断方式编程，用软件计数器R4记录定时器T1的TH1TL1溢出中断次数。

```
        ORG   0000H
        LJMP  MAIN
        ORG   001BH
        LJMP  PRIC
        ORG   0030H
MAIN：  MOV   TMOD，#90H  ；设置T1定时功能、方式1、门控启动
        MOV   TL1，#0     ；初值为0（最大计时时间65.536ms）
        MOV   TH1，#0     ；自动重装初值为0
        MOV   R4，#0      ；软件计数器R4清0
        SETB  EA          ；CPU开中断
        SETB  ET1         ；开T1中断
        SETB  TR1         ；启动T1
WAIT2：JNB    P3.3，WAIT2 ；等待正脉冲到来（正脉冲一到，结束等待并开始计时）
WAIT3：JB     P3.3，WAIT3 ；等待正脉冲结束
        CLR   TR1         ；关闭T1（计时停止）
        MOV   IE，#00H    ；关闭中断
        MOV   R2，TL1     ；读出T1
        MOV   R3，TH1
        LCALL PPS         ；计算脉宽
        SJMP  $           ；踏步
PRIC：  INC   R4          ；记录溢出次数
        RETI
PPS：   ---               ；计算脉宽子程序
        RET
        END
```

计算脉宽的子程序的算式如下：

脉宽t＝（R4 R3 R2）×MC（机器周期）＝（R4 R3 R2）×1μs

知识拓展

简易电子琴设计。在了解计算机发声原理的基础上，利用定时器产生七个标准音阶信号，用按键K1~K7的操作来模拟电子琴的七个琴键，并通过蜂鸣器发出相应的音调。请读者运用所学知识，试着完成该简易电子琴的硬件线路设计及其应用程序的设计。

本章小结

89C51内部有两个16位的可编程计数器/定时器，简称定时器0（或T0）和定时器1（或T1）。可编程是指其功能选择、工作方式、启动方式及计数初值等均可由指令来设置完成。

89C51定时器有两个功能，一是定时功能（使C/T=0），实现定时和延时控制；二是计数功能（使C/T=1），实现对外界时间进行计数。定时和计数实质都是对脉冲的计数，只是被计数脉冲的来源不同。定时功能的计数初值和被计脉冲的周期有关，而计数功能的计数初值只和被计脉冲的个数有关，在计算计数初值时应予以区分。

89C51定时器有四种工作方式（通过M1M0两位选择），其主要区别是计数器长度不同，分8位、13位和16位三种，另外计数初值装入方式不同，分软件重装和硬件自动重装两种。

方式**0**：M1M0=00，计数器长度13位，最大计数值M=2^{13}=8192，计数初值软件重装。

方式**1**：M1M0=01，计数器长度16位，最大计数值M=2^{16}=65536，计数初值软件重装。

方式**2**：M1M0=10，计数器长度8位，最大计数值M=2^{8}=256，计数初值硬件自动重装。

方式**3**：M1M0=11，计数器长度8位，最大计数值M=2^{8}=256，计数初值软件重装。

89C51定时器的启动方式有两种：一种使GATE=0，称之为为非门控启动（内部启动），只要软件控制位TR0或TR1置1即可启动定时器；另一种使GATE=1，称之为门控启动（外部启动），不但软件控制位TR0或TR1须置1，同时还须（P3.2引脚）或（P3.3引脚）为高电平方可启动定时器。

89C51定时器在应用时必须要预置计数初值，根据使用功能不同初值计算的公式不同。

计数功能的计数初值：X= 2^{n}－计数值。

定时功能的计数初值：X= 2^{n}－t/MC。

式中 n为计数器长度（8/13/16），t为定时时间，MC为机器周期（12/f_{osc}）。

89C51定时器无论计数还是定时，当计满规定的脉冲个数，即计数初值回零时，会自动置位TF0或TF1，可以通过查询方式监视，查询后要注意清TF0或TF1。在允许中断的情况下，定时器自动进人中断，中断后会自动清TF0或TF1。若采用查询方式，CPU不能执行别的任务，如果用中断方式可提高CPU的工作效率。

由于89C51定时器的功能是由软件编程确定的，所以，一般在使用前都要对其初始化。初始化骤如下：

① 确定控制字（对TMOD寄存器设置）。

② 预置定时或计数初值（直接将初值写入TH0、TL0或TH1、TL1）。

③ 根据需要开启定时器中断（对IE寄存器设置）。

④ 启动定时器工作（将TR0或TR1置1）。

思考与练习题

一、单项选择题

1. 89C51定时器设置成计数功能时，外部引脚计数脉冲的最高频率应是晶振频率的（　）。

A. 1/12　　B. 1/24　　C. 1/48　　D. 与晶振频率相同

2. T0的中断服务程序入口地址是（　）。

A. 0003H　　B. 000BH　　C. 0013H　　D. 001BH

3. TMOD=00H，则T1工作方式是（　）。

A. 13位计数器　　B. 16位计数器　　C. 13位定时器　　D. 16位定时器

4. 定时器/计数器工作于模式2，在计数溢出时（　）。

A. 计数从零重新开始　　B. 计数停止

C. 计数从初值重新开始　　D. 软件重装初值重新计数

5. 已知晶振频率为6MHZ，定时器T0工作在模式2时，其初值计算公式是（　）。

A. $t=(2^8-X)\times1\mu s$　　B. $t=(2^8-X)\times2\mu s$

C. $t=(2^{16}-X)\times2\mu s$　　D. $t=(2^{13}-X)\times2\mu s$

二、判断题

1. 可编程计数器/定时器工作在计数功能或定时功能时，本质都是计数。（　）
2. 89C51定时器用作计数功能时，是对内部机器周期脉冲计数。（　）
3. 89C51两个定时器都有四种工作方式。（　）
4. 只要TR0置1，定时器0立刻被启动。（　）
5. 89C51两个定时器只能用中断方式工作。（　）

三、问答题

1. 89C51定时器的门控位GATE设置为1时，定时器如何启动？
2. 89C51的定时器作定时用时，其定时时间与哪些因素有关？作计数用时，对输入信号频率有何限制？
3. 设单片机的晶振频率f_{osc}=6MHz，定时器处于不同工作方式时，最大定时范围分别是多少？
4. 用89C51定时器，如何通过软、硬件结合的方法，实现较长时间的定时？
5. 当定时器T0设为工作方式3时，TR1、TFl已为TH0占用的情况下，如何控制定时器T1的开启和关闭？

四、程序设计题

1. 用定时器T0计数，每计10个脉冲，P1.0输出变反一次，用查询和中断两种方式编程。
2. 用定时器T1的工作方式1，编程产生一个50Hz的方波，由P1.0输出。设晶振频率f_{osc}=12MHz。
3. 在89C51单片机中，已知晶振频率f_{osc}=6MHz，试编程使 P1.0和P1.1分别输出周期为

2ms和500ms的方波。

4. 已知89C51单片机的f_{osc} =6 MHz，请利用定时器T0和P1.0输出矩形波。矩形波高电平宽为50μs，低电平宽为300μs。
5. 设晶振频率f_{osc} =6MHz，用定时器T0作为外部计数器，试编程实现每当计到1000个脉冲使
6. 定时器Tl开始2ms定时，定时时间到后，定时器T0又开始计数，这样反复循环下去。

实验项目4　定时器的使用及编程方法

一、实验目的

1. 进一步加深理解89C51定时器的基本工作原理。
2. 通过定时器的基本应用实验，牢固掌握89C51定时器应用的程序设计要点和编程技巧。
3. 进一步掌握STC_ISP_V480在系统编程软件的操作。
4. 在实践中，切切实实地提高单片机应用的操作技能和动手能力，独立思考问题、分析问题和解决问题的能力。

二、实验设备

1. PC计算机一台
2. 单片机实验装置一套（教材配套）
3. Keil μVision2 集成开发环境
4. STC_ISP_V480在系统编程软件

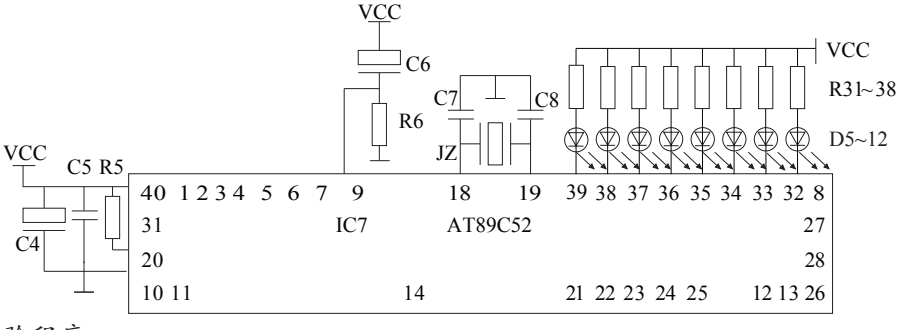
三、实验内容

将P0口接上八个发光二极管，使八个发光二极管依次轮流点亮，每个发光二极管点亮时间为1s。设单片机的晶振为12MHz，用定时器T0模式1溢出中断方式编程实现。

定时分析：12MHz晶振时，T0在模式1情况下的最长定时时间$t = 2^{16} \times 1\mu s = 65.536ms$，显然不能达到1s时间。如果将T0定时50 ms，使其经过20次中断后控制发光二极管的点亮，那么每个发光二极管点亮时间恰好是1s。

定时初值：$t = (2^{16} - X) \times 1\mu s$　　$X = 65536 - 50000 = 15536 = 3CB0H$

1. 实验线路



2. 实验程序

```
ORG    0000H
LJMP   MAIN
ORG    000BH
```

```
        LJMP   CTC0
        ORG    0030H
MAIN:   MOV    SP，#60H
        MOV    TMOD，#01H
        MOV    TL0，#0B0H
        MOV    TH0，#3CH
        MOV    R0，#00H
        MOV    R1，#20
        SETB   TR0
        SETB   ET0
        SETB   EA
        SJMP   $
CTC0:   CLR    TR0
        MOV    TL0，#0B0H
        MOV    TH0，#3CH
        SETB   TR0
        DJNZ   R1，EXIT
        MOV    R1，#20
        MOV    DPTR，#DATA1
        MOV    A，R0
        MOVC A，@A+DPTR
        MOV    P0，A
        INC    R0
        CJNE   R0，#08H，EXIT
        MOV    R0，#00H
EXIT:   RETI
DATA1:  DB     0FEH，0FDH，0FBH，0F7H
        DB     0EFH，0DFH，0BFH，7FH
        END
```

四、实验操作

1. 熟悉实验线路的基本工作原理。
2. 理解源程序，尤其是要弄清定时器溢出中断方式编程结构、程序中DJNZ R1，EXIT指令的作用以及八个发光二极管依次轮流点亮的编程技巧。
3. 打开Keil μVision2 集成开发环境，对源程进行编辑，然后汇编直至通过，最终生存HEX文件。
4. 完成PC机和单片机实验装置的连接，打开STC_ISP_V480在系统编程软件，将HEX文件下载到单片机中。
5. 运行程序，观察LED显示结果。若符合程序功能的设计要求，表明实验成功。
6. 修改程序，使每个发光二极管点亮时间为0.5s，并用定时器T1的模式1的溢出中断方式编程 实现。运行程序，观察LED显示结果。

第5章 串行口及串行通信技术

单片机在实际应用中，经常有通信上的要求。利用89C51单片机自带的串行口可以很方便的实现单片机与单片机之间的串行通信以及单片机与PC机之间的串行通信。本章突出串行通信基本概念的掌握，主要围绕89C51单片机点对点的串行通信原理和方法以及单片机和PC的RS-232接口的通信原理和方法等展开详细介绍。

5.1 串行通信的基本概念

学习目标

1. 熟悉并行通信和串行通信的区别及各自特点。
2. 掌握奇偶校验、波特率、传输方式、异步通信及串行接口组成等基本概念。

导　入

在通信领域内，有两种数据通信方式：并行通信和串行通信。通俗地讲，并行通信的各位数据是同时进行传输的，好比多辆汽车在一条多车道的大路上齐头并进；而串行通信各位数据是一位一位地依次传输的，好比多辆汽车在一条单车道的小路上排队前行。从表面上看，并行通信因其数据传输效率高而优于串行通信，但实际上是各有千秋。串行通信尽管速度较慢，但因其只需要少数几条线就可以在系统间交换信息，并且适合于远距离传输，因此被广泛应用于计算机与计算机、计算机与外设之间的通信。实现串行通信要涉及到方方面面的技术问题，这些问题是本节讨论的重点内容。

5.1.1 并行通信与串行通信

在实际应用中，计算机与外部设备之间常常要进行信息交换，计算机与计算机之间也需要交换信息，所有这些信息的交换均称为“通信”。通信的基本方式分为并行通信和串行通信两种，图5-1为这两种通信方式的示意图。

图 5-1 两种通信方式的示意图

（a）并行通信 （b）串行通信

并行通信是构成1组数据的各位同时进行传送，例如8位数据或16位数据并行传送。其特点是传输速度快，但当距离较远、位数多时会导致通信线路复杂且成本高。

串行通信是数据一位接一位地按顺序传送，其特点是通信线路简单，只要一对传输线就可以实现通信，从而大大地降低了成本，特别适用于远距离通信，缺点是传送速度慢。

需要指出的是：“串行”是指外设与接口电路之间的信息传送方式，CPU与接口之间仍按并行方式工作。

5.1.2 信息传输的检错和纠错

串行数据在传输过程中，由于干扰可能引起信息出错，如何发现传输中的错误叫检错，发现错误后，如何消除错误叫纠错。

最简单的检错方法是奇偶校验，即在传送字符的各位之外，再传送1位奇/偶校验位。可采用奇校验或偶校验。

奇校验：使所有传送的数位（含字符的各数位和校验位）中，含1的个数为奇数个。

偶校验：使所有传送的数位（含字符的各数位和校验位）中，含1的个数为偶数个。

需要注意的是：奇偶校验能够检测出1位误码，但是不能纠错。

5.1.3 波特率（Baudrate）

在串行通信中，用波特率来描述数据的传输速率。波特率即每秒钟传送的二进制位数，单位为b/s，即位/秒。波特率用于表征数据传输的速度，波特率越高，数据传输速度越快。但波特率和字符的实际传输速率不同，字符的实际传输速率是每秒内所传字符帧的帧数，和字符帧格式有关。

【例5-1】 假设字符传送的速率为120字符/秒，而每一个字符为10位，问传送的波特率为多少？每位传送时间为多少？

答：波特率=10位/字符×120字符/秒=1200位/秒=1200b/s。

每1位二进制位的传送时间T_d就是波特率的倒数，T_d=1/1200=0.833ms。

需要注意的是：在串行通信时，发送端与接收端的波特率必须一致。

5.1.4 串行通信的传输方式

在串行通信中数据是在两个站之间进行传送的，按照数据传送方向，串行通信可分为三种方式：单工方式、半双工方式和全双工方式。图5-2为三种方式的示意图。

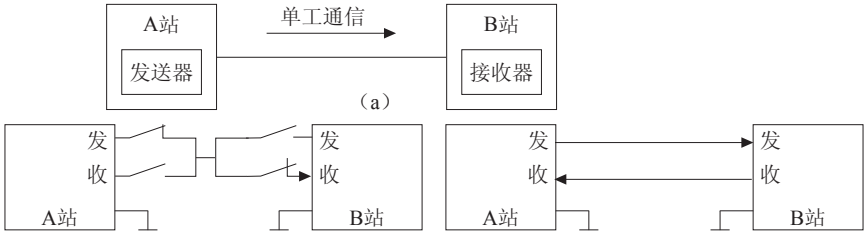


图 5-2 串行通信传输方式

（a）单工通信　（b）半双工通信　（c）全双工通信

在单工方式下，通信线的一端接发送器，一端接接收器，数据只能按照一个固定的方向传送，如图5-2（a）所示。

在半双工方式下，系统的每个通信设备都由一个发送器和一个接收器组成，如图5-2（b）所示。在这种方式下，数据能从A站传送到B站，也可以从B站传送到A站，但是不能同时在两个方向上传送，即只能一端发送，一端接收。其收/发开关一般是由软件控制的电子开关。

在全双工方式下，通信双方都有发送器和接收器，可以同时发送和接收，即数据可以在两个方向上同时传送，如图5-2（c）所示。

5.1.5 异步通信和同步通信

按照串行数据的时钟控制方式，串行通信可分为异步通信和同步通信两类。

1. 异步通信

在异步通信中，数据通常是以字符为单位组成字符帧传送的。字符帧由发送端一帧一帧地发送，每一帧数据均是低位在前，高位在后，通过传输线被接收端一帧一帧地接收。发送端和接收端可以由各自独立的时钟来控制数据的发送和接收，这两个时钟彼此独立，互不同步。

在异步通信中，接收端是依靠字符帧格式来判断发送端是何时开始发送、何时结束发送的。字符帧格式是异步通信的一个重要指标。字符帧也叫数据帧，由起始位、数据位、奇偶校验位和停止位四部分组成，如图5-3所示。

① 起始位：位于字符帧开头，仅占一位，为逻辑0低电平，用于向接收设备表示发送端开始发送一帧信息。

② 数据位：紧跟起始位之后，用户根据情况可取5位、6位、7位或8位，低位在前，高位在后。

③ 奇偶校验位：位于数据位之后，仅占一位，用来表示串行通信中采用奇校验还是偶校验，由用户决定。

④ 停止位：位于字符帧最后，为逻辑1高电平。通常可取1位、1.5位或2位，用于向接收端表示一帧字符信息已经发送完，也为发送下一帧作准备。

⑤ 在异步通信中，两个相邻字符帧之间可以没有空闲位，也可以有若干空闲位，这由

用户来决定。图5-3（a）表示没有空闲位的字符帧格式。图5-3（b）表示有三个空闲位的字符帧格式。

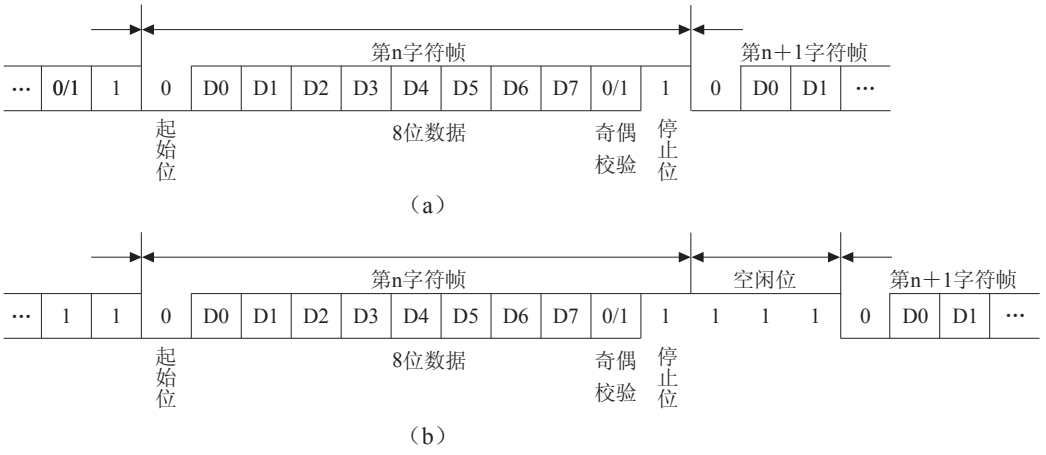


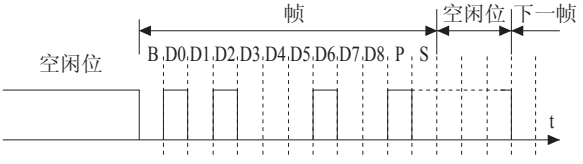
图 5-3 异步通信字符帧格式

（a）无空闲位字符帧 （b）有空闲位字符帧

异步通信的优点是不需要传送同步时钟，字符帧长度不受限制，故设备简单。缺点是字符帧中因包含起始位和停止位而降低了有效数据的传输速率。

【例5-2】传送8位数据45H（01000101B），奇校验，画出信号线上异步通信的波形图。

答：按上所述，其信号线上波形图如下。



2. 同步通信

同步通信是一种连续串行传送数据的通信方式，一次通信只传输一帧信息。这里的信息帧和异步通信的字符帧不同，通常有若干个数据字符，如图5-4所示。图5-4（a）为单同步字符帧结构，图5-4（b）为双同步字符帧结构，但它们均由同步字符、数据字符和校验字符CRC三部分组成。

在同步通信中，同步字符可以采用统一的标准格式，也可以由用户约定。

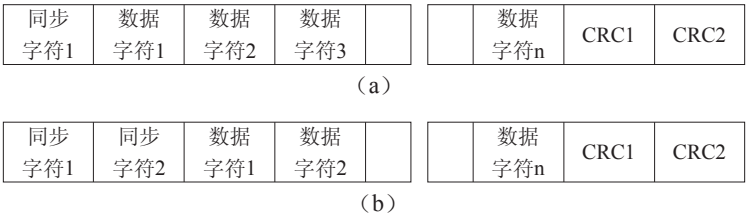


图 5-4 同步通信字符帧格式

（a）单同步字符帧格式 （b）双同步字符帧格式

5.1.6 串行接口必须具备的功能

串行接口电路的种类和型号很多。能够完成异步通信的硬件电路称为UART（Universal

Asychronous Receiver/Transmitter），即通用异步接收器/发送器；能够完成同步通信的硬件电路称为USRT（Universal Sychronous Receiver/Transmitter）；既能够完成异步又能完成同步通信的硬件电路称为USART（Universal Sychronous z Asychronous Receiver/Transmitter）。

从本质上说，所有的串行接口电路都是以并行数据形式与CPU接口，以串行数据形式与外部逻辑接口。它们的基本功能都是从外部逻辑接收串行数据，转换成并行数据后传送给CPU，或从CPU接收并行数据，转换成串行数据后输出到外部逻辑。所有这些串行接口电路都必须具备如下基本功能。

① 发送器：并→串数据转换，添加标识位和校验位，设置发送结束标志，申请中断。

② 接收器：串→并数据转换，检查错误，去掉标识位，保存有效数据，设置接收结束标志，申请中断。

③ 控制器：接收编程命令和控制参数，设置工作方式：同步/异步、字符格式、波特率、校验方式、数据位与同步时钟比例等。

5.2 89C51串行口的结构和工作原理

学习目标

1. 掌握89C51串行口的组成结构和工作原理。
2. 掌握89C51串行口四种工作方式的控制和应用特点。
3. 掌握波特率的计算。

导　入

串行数据通信归根到底是要解决两个关键技术问题，一个是数据传送方式，另一个是数据格式转换。89C51具有一个能完成异步通信的UART接口电路，很好地解决了上述两个关键技术问题。学生对89C51串口的了解应该牢牢把握住它的发送器、接收器和控制器三个核心部件各自所起的作用及其基本工作原理，为串口应用编程打下基础。

5.2.1 89C51串行口的结构

89C51具有一个可编程的全双工串行通信接口，通过软件设置，它既可作为通用异步接收器/发送器UART使用，也可作为同步移位寄存器使用。其帧格式有8位、10位和11位三

种，可设置各种波特率，使用起来灵活方便。其内部结构如图5-5所示。

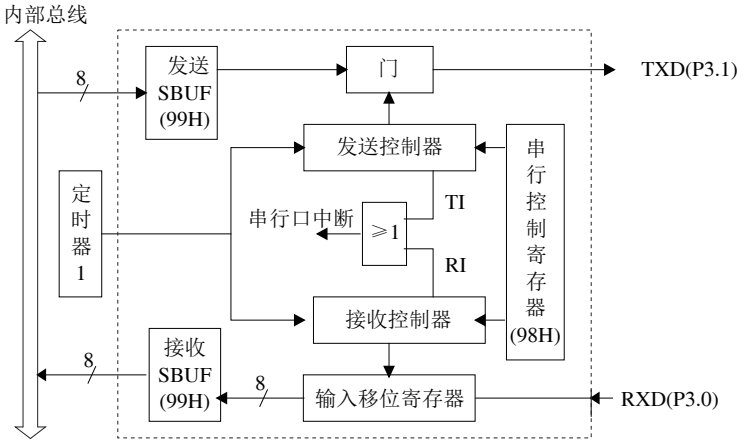


图 5-5 串行口内部结构框图

89C51串行口的组成和功能总体介绍如下：

（1）89C51的串行口是一个通用异步全双工串口，可以同时进行数据的接收和发送。

（2）通过对外的两条独立收、发信号线RXD（P3.0）、TXD(P3.1)与外界进行通信。

（3）按不同工作方式，一帧位数有8位、10位和11位三种，格式如图5-6所示。

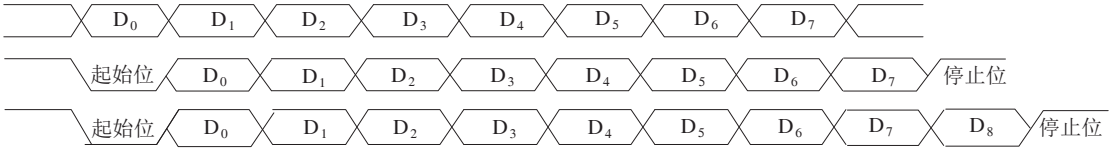


图 5-6 8位/10位/11位帧格式

（4）一帧字符发送或接收完，置位标志位TI或RI，并申请中断。

（5）中断允许控制位ES，中断入口地址为0023H。

（6）有两个独立的接收、发送缓冲器SBUF。SBUF属于特殊功能寄存器。发送缓冲器SBUF只能写入不能读出，接收缓冲器SBUF只能读出不能写入，二者共用一个字节地址（99H）。

- 发送SBUF放待发的8位数据，写入SBUF便启动发送。

发送指令：MOV SBUF，A

- 接收SBUF放已成功接收的8位数据，供CPU读取。

读取指令：MOV A，SBUF

（7）串行口控制寄存器SCON。

SCON用来控制串行口的工作方式和状态，可以位寻址。单片机复位时，所有位全为0。其各位的定义格式如图5-7所示。

SCON	SM0	SM1	SM2	REN	TB8	RB8	TI	RI

图 5-7 SCON的位定义

各位含义如下：

① SM0，SM1：串口四种工作方式选择位，其定义如表5-1所示。

表 5-1 串行口的工作方式

SM0 SM1	工作方式	功 能	波特率
0 0	方式0	8位同步移位寄存器	$f_{osc}/12$
0 1	方式1	10位UART	可变
1 0	方式2	11位UART	$f_{osc}/64$或$f_{osc}/32$
1 1	方式3	11位UART	可变

② SM2：多机通讯控制位，用于方式2和方式3中。

- 在方式2和方式3处于接收方式时，若SM2=1，且接收到的第9位数据RB8为0时，不激活RI；若SM2=1，且RB8=1时，则置RI=1。
- 在方式2和方式3处于接收或发送方式时，若SM2=0，不论接收到的第9位RB8是0还是1，TI、RI都以正常方式被激活。
- 在方式1处于接收方式时，若SM2=1，则只有收到有效的停止位后，RI置1。
- 在方式0中，SM2应为0。

③ REN：允许接收控制位，它由软件置位或清0。REN=1时，允许接收；REN=0时，禁止接收。

④ TB8：在方式2和方式3中，是发送机要发送的第9位数据。

- 可做奇偶校验位，由软件置位或复位。
- 在多机通信中，可作为区别地址帧或数据帧的标识位，一般约定地址帧时，TB8为1，数据帧时，TB8为0。

⑤ RB8：在方式2，3中，接收来自发送机的第9位数据。

⑥ TI：发送中断标志，发送一帧结束，TI=1，必须软件清0。

⑦ RI：接收中断标志，接收一帧结束，RI=1，必须软件清0。

（8）电源控制寄存器PCON（串口框图中未画出）。

特殊功能寄存器PCON中各位的定义格式如图5-8所示。只有一位（最高位）SMOD与串行口的工作有关，该位是串行口波特率系数的控制位：SMOD=1时，波特率加倍，否则不加倍。

PCON不可位寻址，因此初始化时需要字节传送。

PCON	SMOD	×	×	×	GF1	GF0	PD	IDL

图 5-8 PCON的位定义

5.2.2 89C51串行口的工作方式

根据实际需要，89C51的串行口有四种工作方式可供选择，通过软件设置SCON中的SM0、SM1位来决定。见表5-1。

1. *方式0*

在方式0下，串行口作同步移位寄存器用，这种方式常用于扩展I/O口。下面分发送和接收两种情况讨论。

① 发送

- 以8位数据为一帧，无起始位和停止位。其帧格式如下：

- RXD端用作数据输出，先发送最低位。
- TXD端用作同步脉冲输出，每输出一个脉冲，在RXD端就移出一位数据。
- 波特率=f_{osc}/12，是固定的。
- 发送过程：当一个8位数据写入发送缓冲器SBUF时，便启动发送。一帧发送完，中断标志TI为1，请求中断。在再次发送数据之前，必须由软件清TI为0。

② 接收

- 以8位数据为一帧，无起始位和停止位。其帧格式如下：

- RXD端用作数据输入，先接收最低位。
- TXD端用作同步脉冲输出，每输出一个脉冲，在RXD端就移进一位数据。
- 波特率=f_{osc}/12，是固定的。
- 接收过程：在满足REN=1和RI=0的条件下，从RXD端接收数据，当接收完8位数据后，置中断标志RI为1，请求中断。在再次接收数据之前，必须由软件清RI为0。

图5-9给出的是用串行口方式0扩展8位并行I/O口的接线图。（a）图中的74HC164为串入并出 移位寄存器，用作8位输出I/O扩展；（b）图中的74HC165为并入串出移位寄存器，用作8位输入I/O扩展。

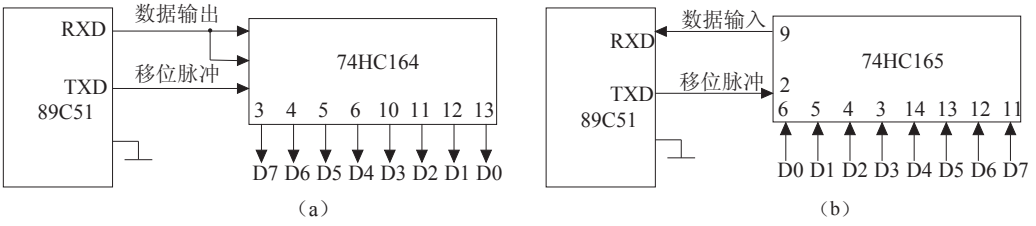


图 5-9 串行口方式0扩展8位并行I/O口

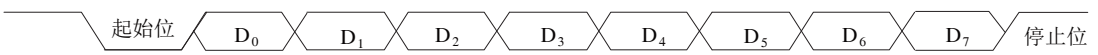
（a）串口方式0的8位输出I/O扩展 （b）串口方式0的8位输入I/O扩展

值得一提的是，方式0并非是作串行通信使用的，通常是用于扩展I/O的。在方式0时，串行控制寄存器SCON中的TB8和RB8未使用，并且SM2必须要设置为0。

2. 方式1

在方式1时，串行口被设置为10位通用异步通信接口UART。下面一并讨论发送和接收。

- 一帧10位：起始位+8位数据位+停止位。其帧格式如下：



- 波特率：用定时器T1作波特率发生器。波特率=$(2^{SMOD}/32)\times$T1溢出率，是可变的。
- 发送：数据从TXD端输出，当数据写入发送缓冲器SBUF后，启动发送器发送。当发送完一帧数据后，置中断标志TI为1。
- 接收：REN置1，允许接收，串行口采样RXD，当采样由1到0跳变时，确认是起始

位“0”，开始接收一帧数据。当RI＝0，且停止位为1或SM2＝0时，停止位进入RB8位，同时置中断标志RI，否则信息将丢失。所以，采用方式1接收时，应先用软件清除RI或SM2标志。

值得一提的是，在整个接收过程中，保证REN＝1是一个先决条件，只有当REN＝1时，才能对RXD进行检测。

3. *方式2和方式3*

在方式2和方式3时，串行口被设置为11位通用异步通信接口UART。方式2和方式3除了波特率不同以外，其余完全相同，所以在此一并讨论。

- 一帧11位：起始位+9位数据位+停止位。其帧格式如下：

- 其中第9位数据位（D_8）在TB8或RB8中，常作奇偶校验位用或作多机通讯标识位（地址/数据）用。
- 波特率：方式2：波特率=$(2^{SMOD}/64)\times f_{osc}$，两种选择（由SMOD=0或1决定）。
  方式3：波特率=$(2^{SMOD}/32)\times$T1溢出率，是可变的。
- 发送：先根据通信协议由软件设置TB8（奇偶校验位或作多机通讯标识位），然后将要发送的数据写入SBUF，即可启动发送过程。串行口能自动把TB8取出，并装入到第9位数据位（D_8）的位置，再逐一发送出去。在送完一帧信息后，TI被自动置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0。
- 接收：当REN=1时，允许串行口接收数据。数据由RXD端输入，接收11位的信息。当接收器采样到RXD端的负跳变，并判断起始位有效后，开始接收一帧信息。当接收器接收到第9位数据后，若同时满足以下两个条件：RI=0和SM2=0或接收到的第9位数据为1，则接收数据有效，8位数据送入SBUF，第9位送入RB8，并置RI=1。若不满足上述两个条件，则信息丢失。

5.2.3 关于波特率的计算

89C51串行口的四种工作方式对应着三种波特率的计算公式，下面加以分析。

1. *方式0*

波特率=$f_{osc}/12$。波特率是固定的，为单片机晶振频率的十二分之一。

2. *方式2*

波特率=$f_{osc}\times 2^{SMOD}/64$。波特率有两种可供选择，即SMOD＝1时为$f_{osc}/32$，SMOD＝0时为$f_{osc}/64$。

3. *方式1和方式3*

波特率=$(2^{SMOD}/32)\times$T1溢出率。波特率由定时器T1的溢出率和SMOD位共同决定，为可变波特率。实际上，当定时器T1做波特率发生器使用时，被设定为定时功能，并且通常是工作在方式2的，即自动重装载的8位定时器，此时TL1作计数用，自动重装载的值在TH1内。设定时计数初值为X，那么每过256－X个机器周期，定时器溢出一次。为了避免因溢出而产生不必要的中断，此时应禁止T1中断。溢出周期为

$$\frac{12}{f_{osc}}\cdot(256-X)$$

而溢出率为溢出周期的倒数，所以方式1和方式3的波特率为

$$\text{波特率}=\frac{2^{SMOD}}{32}\cdot\frac{f_{osc}}{12(256-X)}$$

于是，可得出定时器T1方式2的初值X

$$X=256-\frac{2^{SMOD}\times f_{osc}}{384\times\text{波特率}}$$

表5-2列出了串口方式1和方式3的常用波特率及其初值。

表 5-2 定时器1产生的常用波特率

串口工作方式	波特率(b/s)	f_{osc}(MHz)	定时器T1			
			SMOD	C/T̄	模式	定时器初值
方式0	1 000 K	12	×	×	×	×
方式2	375 K	12	1	×	×	×
	187.5 K	12	0	×	×	×
方式1和方式3	62.5 K	12	1	0	2	FFH
	19.2 K	11.059	1	0	2	FDH
	9.6 K	11.059	0	0	2	FDH
	4.8 K	11.059	0	0	2	FAH
	2.4 K	11.059	0	0	2	F4H
	1.2 K	11.059	0	0	2	E8H
	137.5 K	11.059	0	0	2	1DH
	110 K	12	0	0	1	FEEBH

5.3 89C51串行口的应用编程

学习目标

1. 掌握89C51串行口的初始化内容设置。
2. 掌握89C51串行口的初始化编程格式及异步通信编程格式。
3. 掌握89C51串行口异步通信的程序设计。

导　入

89C51的串口是一个可编程的串口，有四种工作方式可供选择。因此，编程时要结合具体应用要求来决定如何选择合适的工作方式。除此之外还要考虑波特率设置，采用查询方式还是中断方式编程等等。这里给出一个最简单的串口应用例子，譬如用89C51的串口进行自发自收。学生不妨运用前面所学的知识试着编制相应的通信程序，先初步检验一下自己对89C51串口应用的理解和掌握程度，并带着问题和思考去学习本节内容。

5.3.1 89C51串行口的初始化

89C51的串行口和定时器一样，其功能也是由软件编程确定的。因此，在使用前先要对其初始化。初始化步骤如下：

① 设置控制字（对SCON寄存器设置）

根据应用要求，确定串行口的工作方式和状态控制字，并写入串行口控制寄存器SCON中。

② 设置波特率（对SMOD和定时器T1的设置）

- 方式0无需设置，其波特率只与晶振频率f_{osc}有关，而与SMOD和T1无关。
- 方式1、方式2和方式3需考虑SMOD位的设置。SMOD置1，波特率加倍；SMOD清0，波特率不加倍。
- 方式1和方式3需考虑定时器T1的设置。定时器T1设置主要包括控制字、初值X及启动的设置。

③ 根据需要开启串行口中断（对IE寄存器设置）

采用中断方式通信，需将CPU开中断（EA=1）及串口开中断（ES=1）。若采用查询方式通信则串口不必开启中断。

5.3.2 89C51串行口的编程格式

1. 初始化编程格式

```
MOV   SCON，#控制字    ；根据应用要求，给出控制字，设置串口的工作方式和状态
MOV   PCON，#80H       ；SMOD置1波特率加倍，若不需要加倍，该指令就去掉
                         （注意：方式0时SMOD无需设置）
MOV   TMOD，#20H       ；波特率发生器T1设置，T1设置成定时功能及方式2
MOV   H1，#X           ；装入与波特率对应的定时初值X
MOV   TL1，#X
SETB  TR1              ；启动T1（注意：方式0和方式2定时器T1无需设置）
SETB  EA               ；CPU开中断（若不需要开中断，该指令就去掉）
SETB  ES               ；串口开中断（若不需要开中断，该指令就去掉）
```

2. 异步通讯发送程序编程格式

如下给出的程序是连续发送10个数据，这些数据存放在片内RAM中30H开始的单元中。

对于发送，一定要先发送一个字符，等待TI=1后再发送下一个字符。

① 查询方式

```
        ORG     0000H
        LJMP    MAIN
        ORG     0030H
MAIN:   ---                 ；串口初始化程序
        MOV     R0，#30H    ；R0指向30H单元
        MOV     R1，#10     ；R1用作循环控制计数器，循环次数10次
TRAM:   MOV     A，@R0      ；取一个数据
        MOV     SBUF，A     ；发送一个字符
WAIT:   JBC     TI，NEXT    ；发送不结束循环等待，发送结束清TI转NEXT
        SJMP    WAIT
NEXT:   INC     R0          ；R0指向下一个单元，准备下一次发送
        DJNZ    R1，TRAM    ；循环控制，10次未到继续循环
        SJMP    $           ；10个数据发送完，执行踏步指令
        END
```

② 中断方式

```
        ORG     0000H
        LJMP    MAIN
        ORG     0023H       ；串行口中断入口
        LJMP    SINT
        ORG     0030H
MAIN:   ---                 ；串口初始化程序
        MOV     R0，#30H    ；R0指向30H单元
        MOV     R1，#10     ；R1用作循环控制计数器，循环次数10次
        MOV     A，@R0      ；取数据
        MOV     SBUF，A     ；先发送第一个字符
        SJMP    $           ；踏步等待中断
SINT:   CLR     TI          ；中断服务程序开始，发送中断标志TI清0
        DJNZ    R1，TRAM    ；10个数据未发送完，转TRAM继续发送
        CLR     ES          ；10个数据发送完，关串口中断
        RETI                ；中断返回
TRAM:   INC     R0          ；R0指向下一个单元
        MOV     A，@R0      ；取数据
        MOV     SBUF，A     ；发送下一个字符
        RETI                ；中断返回
        END
```

3. 异步通讯接收程序编程格式

如下给出的程序是连续接收数据，并用R0间址将数据存放在片内RAM中。

对于接收，一定要满足REN=1且RI=0，才能等待接收，当RI=1，从接收缓冲器SBUF读取数据。

① 查询方式

```
        ORG     0000H
        LJMP    MAIN
        ORG     0030H
MAIN：  ---                     ；串口初始化程序
WAIT：  JBC     RI，NEXT        ；没接收到循环等待，接收到清RI转NEXT
        SJMP    WAIT
NEXT：  MOV     A，SBUF         ；读取接收数据
        MOV     @R0，A          ；保存数据
        INC     R0              ；准备下一次接收
        SJMP    WAIT
        END
```

② 中断方式

```
        ORG     0000H
        LJMP    MAIN
        ORG     0023H           ；串行口中断入口
        LJMP    RINT
        ORG     0030H
MAIN：  ---                     ；串口初始化程序
        SJMP    $               ；踏步等待中断
RINT：  CLR     RI              ；中断服务程序开始，接收中断标志RI清0
        MOV     A，SBUF         ；读取接收数据
        MOV     @R0，A          ；保存数据
        INC     R0              ；准备下一次接收
        RETI                    ；中断返回
        END
```

5.3.3　89C51串行口异步通信应用编程

【例5-3】用串行口方式1收、发字符，传送波特率为2 400b/s。假设发送缓冲区首址为20H，接收缓冲区首址为40H，晶振频率f_{osc}=11.059MHz，试用中断方式编写通信程序。

解：串口初始化设置考虑如下：

对于SCON：SM0、SM1=01时为方式1，在SM2=0和REN=1条件下，允许接收数据，其余各位均取0。则 SCON=01010000B=50H。

对于PCON：方式1的波特率与SMOD位相关，SMOD置1，波特率加倍；SMOD清0，波特率不加倍。设SMOD=0，所以PCON=00H（系统复位以后的状态PCON=00H，可不赋值）。

对于TMOD：方式1的波特率与定时器T1相关，通常T1要求设置成定时功能并且为方式2，所以，TMOD=00100000B=20H。

定时器T1的定时初值X计算：

$$X=256-\frac{2^{SMOD}\times f_{osc}}{384\times 波特率}=256-\frac{2^{0}\times 11.059MHz}{384\times 2400b/s}=244=F4H$$

另一种方法可以通过查表5-2确定，查表得X=F4H。

题目要求采用中断方式编程，则初始化时要考虑开中断 。

程序如下：

```
        ORG   0000H
        LJMP  MAIN
        ORG   0023H          ；串行中断入口
        LJMP  SBR1           ；转至中断服务程序
        ORG   0030H
MAIN:   MOV   SCON，#50H     ；串行口设为方式1，允许接收
        MOV   TMOD，#20H     ；定时器T1设为方式2
        MOV   TL1，#F4H      ；装入定时器初值
        MOV   TH1，#F4H      ；8位重装值
        SETB  TR1            ；启动定时器T1
        SETB  EA             ；CPU开中断
        SETB  ES             ；允许串行口中断
        MOV   R0，#20H       ；设发送缓冲区指针
        MOV   R1，#40H       ；设接收缓冲区指针
        MOV   A，@R0         ；先发送1个字符，取发送数据到A
        MOV   SBUF，A        ；发送数据
        SJMP  $              ；等待中断
SBR1:   JNB   RI，SEND       ；RI=0不是接收则转SEND
        CLR   RI             ；清接收中断标志
        MOV   A，SUBF        ；读入接收缓冲区内容
        MOV   @R1，A         ；存入接收缓冲区
        INC   R1             ；修改接收缓冲区指针
        SJMP  NEXT           ；转至统一出口NEXT
SEND:   CLR   TI             ；清发送中断标志
        INC   R0             ；修改发送数据指针
        MOV   A，@R0         ；取发送数据到A
        MOV   SBUF，A        ；发送数据
NEXT:   RETI                 ；中断返回
        END
```

【例5-4】用串行口方式2，将片内RAM 50H起始单元的16个字节数据往外发送。波特率为系统时钟的32分频，并进行偶校验。采用中断方式发送。

解：串口初始化设置考虑如下。

SCON：SM0、SM1=10时为方式2，其余各位均取0。则 SCON=10000000B=80H。

PCON：方式2的波特率=$f_{osc}\times 2^{SMOD}/64$。当SMOD=1时，波特率为系统时钟的32分频，所以PCON=80H。

方式2的波特率与定时器T1无关，因此TMOD无需设置，初值X也不用计算。

题目要求采用中断方式发送，所以初始化时要考虑开中断。

程序如下：

```
        ORG   0000H
        LJMP  MAIN
        ORG   0023H              ；串行口中断入口
        LJMP  TRANI              ；转至中断服务程序
        ORG   0030H
MAIN:   MOV   SCON，#80H         ；串行口设为方式2，不接收
        MOV   PCON，#80H         ；SMOD置1
        SETB  EA                 ；CPU开中断
        SETB  ES                 ；允许串行口中断
        MOV   R0，#50H           ；设数据指针
        MOV   R7，#10H           ；数据长度
        MOV   A，@R0             ；取一个字符
        MOV   C，P               ；添加偶校验于TB8
        MOV   TB8，C
        MOV   SBUF，A            ；启动一次发送
HERE:   SJMP  HERE               ；等待中断
TRANI:  CLR   TI                 ；清发送结束标志
        DJNZ  R7，NEXT           ；未发送完转NEXT继续发送
        CLR   ES                 ；发送完，关闭串行口中断
        SJMP  TEND
NEXT:   INC   R0                 ；未发送完，修改指针
        MOV   A，@R0             ；取下一个字符
        MOV   C，P               ；添加偶校验于TB8
        MOV   TB8，C
        MOV   SBUF，A            ；发送一个字符
TEND:   RETI                     ；中断返回
        END
```

【例5-5】用串行口方式3接收16个字符，存入片内RAM的50H起始单元，要求波特率为9 600b/s，进行偶校验。设晶振频率为11.0592MHz，试采用查询方式编程。

解：串口初始化设置考虑如下。

SCON：SM0、SM1=11时为方式3，在SM2=0和REN=1条件下，允许接收数据，其余各位均取0，则 SCON=11010000B=D0H。

PCON：方式3的波特率与SMOD位相关，设SMOD=0，所以PCON=00。

TMOD：方式3的波特率与定时器T1相关，设TMOD=00100000B=20H。

定时器T1的定时初值通过查表5-2得X=FDH。

题目要求采用查询方式接收，所以初始化时不需开中断。

程序如下：

```
        ORG   0000H
        LJMP  MAIN
        ORG   0030H
MAIN:   MOV   SCON，#0D0H      ；串行口方式3允许接收
        MOV   TMOD，#20H       ；T1定时方式2
        MOV   TL1，#0FDH       ；写入定时初值X
        MOV   TH1，#0FDH
        SETB  TR1              ；启动T1
        MOV   R0，#50H         ；设数据指针
        MOV   R7，#10H         ；设接收数据长度16
WAIT:   JBC   RI，NEXT         ；等待串行口接收，收到清RI转NEXT
        SJMP  WAIT
NEXT:   MOV   A，SBUF          ；取一个接收字符
        JNB   P，COMP          ；偶校验
        JNB   RB8，ERR         ；P≠RB8，数据出错转ERR
        SJMP  RIGHT            ；P=RB8，数据正确转RIGHT
COMP:   JB    RB8，ERR         ；P≠RB8，数据出错转ERR
RIGHT:  MOV   @R0，A           ；保存一个字符
        INC   R0               ；修改指针
        DJNZ  R7，WAIT         ；全部字符接收完
        CLR   F0               ；F0 =0，接收数据全部正确
        SJMP  $                ；踏步等待
ERR:    SETB  F0               ；F0 =1，接收数据出错
        SJMP  $                ；踏步等待
        END
```

【例5-6】图5-10所示是89C51双机串行通信的连接线路。要求甲、乙两台单片机实现全双工串行通信，以方式2工作、每帧为11位、可程控的第9位数据位用于奇偶校验的补偶位。编出能实现如下功能的程序。

甲机：每发送1帧信息，乙机对接收的数据进行奇偶校验，若补偶正确，则乙机向甲机发出“数据发送正确”的信息（例中以00H作为回答信号），甲机接收到该回答信号后再发送下1字节；若奇偶校验错，则乙机发出“数据发送不正确”的信息（例中以AAH作为回答信号）给甲机，要求甲机再次发送原数据，直至发送正确。甲机发送128个字节后就停止发送。

乙机：接收甲机发送来的数据并进行奇偶校验，与此同时发出相应的回答信息（即00H或AAH），直到接收完128个字节为止。

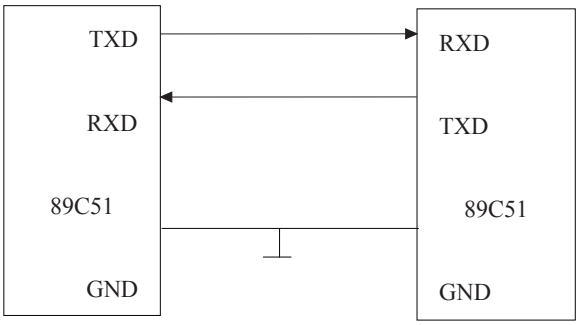


图 5-10　89C51双机串行通信连接

解：实现上述通信要求的甲、乙两机的程序设计如下。其中，图5-11是甲机的程序流程图，图5-12是乙机的程序流程图。

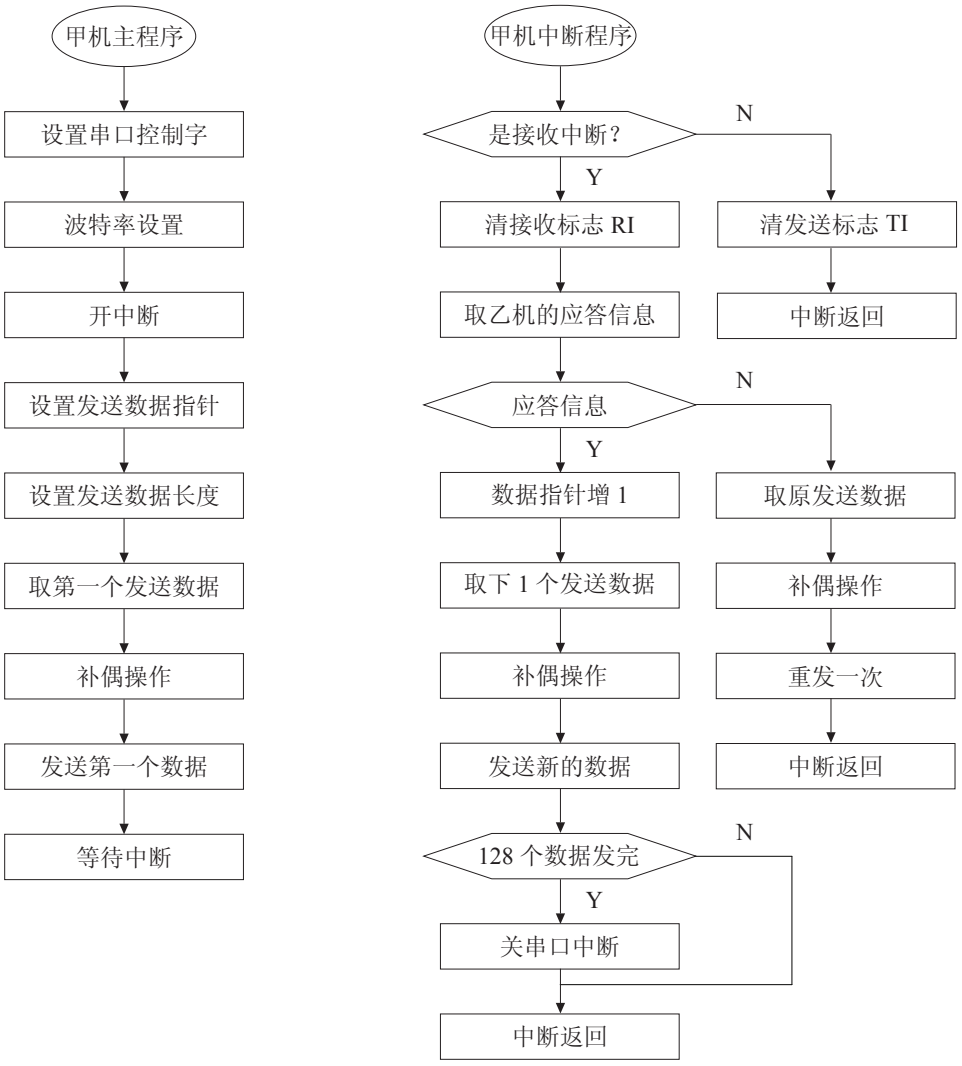


图 5-11　甲机程序流程图

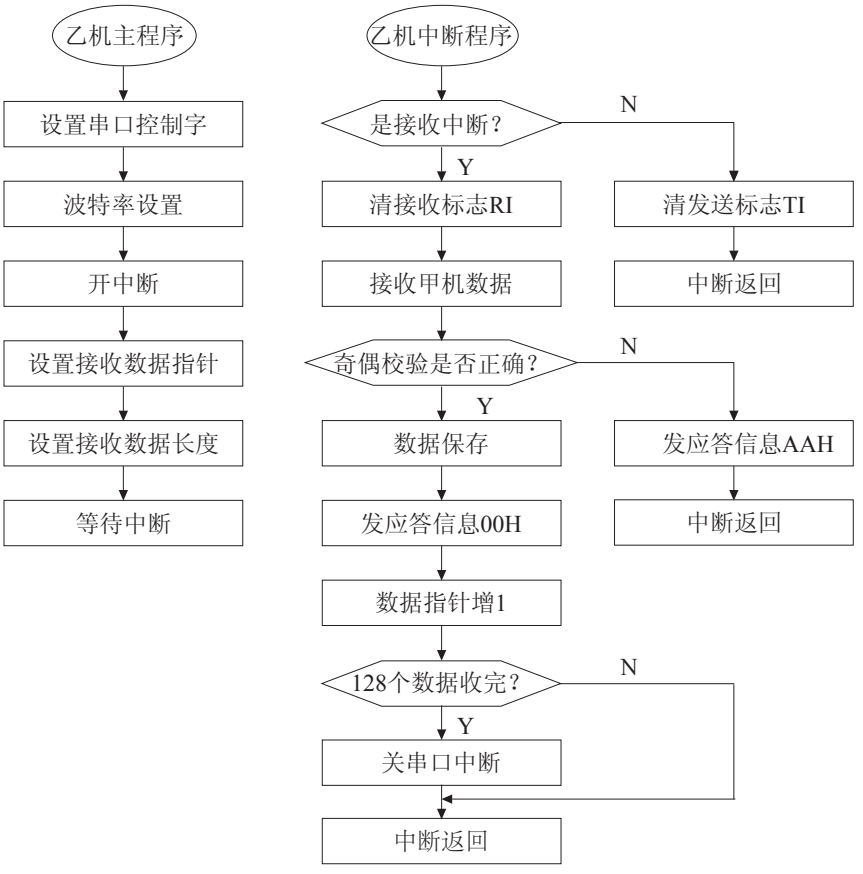


图 5-12 乙机程序流程图

甲机程序如下：

```
        ORG   0000H
        LJMP  MAIN          ；转至主程序入口处
        ORG   0023H         ；串行中断入口
        LJMP  INTSE1        ；转至中断服务程序
        ORG   0030H
MAIN:   MOV   SCON，#90H    ；置串口方式2、允许接收
        MOV   PCON，#80H    ；波特率加倍（波特率为系统时钟的32分频）
        SETB  EA            ；CPU开中断
        SETB  ES            ；允许串行口中断
        MOV   DPTR，#ADDR1  ；设置片外RAM数据块指针
        MOV   R0，#80H      ；设置发送字节数据长度（128字节）
        MOVX  A，@DPTR      ；取第1个发送数据
        MOV   C，P          ；奇偶标志位P送C_Y
        MOV   TB8，C        ；C_Y送TB8，补偶（采用偶校验）
        MOV   SUBF，A       ；发送第1个数据
        SJMP  $             ；等待中断
INTSE1: JB    RI，LOOP1     ；是否接收中断，是则转LOOP1
```

```
        CLR   TI            ；是发送中断，则清TI标志
        RETI                ；中断返回
LOOP1： CLR   RI            ；接收中断，清RI标志
        MOV   A，SBUF       ；取乙机的应答信息
        CJNE A，#00H，LOOP2；若A=00H发送正确则不转，若A=FFH发送不正
                              确，转到LOOP2
        INC   DPTR          ；DPTR地址指针增1
        MOVX A，@DPTR       ；取下1个发送数据
        MOV   C，P
        MOV   TB8，C        ；TB8补偶
        MOV   SUBF，A       ；发送新的数据
        DJNZ  R0，ENDT1     ；未发送完128个数据，转ENDT1
        CLR   ES            ；128个数据发送完毕，则禁止串行口中断
ENDT1： RETI                ；中断返回
LOOP2： MOVX A，@DPTR       ；准备重发1次数据
        MOV   C，P
        MOV   TB8，C        ；TB8补偶
        MOV   SBUF，A       ；重发1次数据
        RETI                ；中断返回
        END
```

乙机程序如下：

```
        ORG   0000H
        LJMP  MAIN          ；转至主程序入口处
        ORG   0023H         ；串行中断入口
        LJMP  INTSE2        ；转至中断服务程序
        ORG   0030H
MAIN：  MOV   SCON，#90H    ；置串口工作方式2、允许接收
        MOV   PCON，#80H    ；波特率加倍（波特率为系统时钟的32分频）
        SETB  EA            ；CPU开中断
        SETB  ES            ；允许串行口中断
        MOV   DPTR，#ADDR2  ；设置片外RAM数据块指针
        MOV   R0，#80H      ；设置接收字节数据长度（128字节）
        SJMP  $             ；等待中断
INTSE2： JNB  RI，LOOP6     ；若不是接收中断，则转LOOP6
        CLR   RI            ；接收中断，则先清RI
        MOV   A，SBUF       ；接收数据
        MOV   C，P          ；奇偶标志位P送CY
        JC    LOOP4         ；若CY=1，转LOOP4（此时接收数据含奇数个1）
        ORL   C，RB8        ；若CY=0，则RB8肯定为0
```

```
        JC      LOOP5          ；若RB8=1说明偶校验出错，此时转LOOP5
LOOP3：MOVX @DPTR，A           ；接收数据正确，存入片外RAM
        MOV     A，00H
        MOV     SBUF，A        ；发送接收正确应答信息00H
        INC     DPTR           ；DPTR地址指针增1
        DJN     ZR0，ENDT2     ；未接收完128个数据，转ENDT2
        CLR     ES             ；接收完128个数据，则关串行口中断
ENDT2：RETI                    ；中断返回
LOOP4：ANL      C，RB8         ；若CY=1则RB8肯定为1
        JC      LOOP3          ；若RB8=1说明偶校验正确，此时转LOOP3
LOOP5：MOV      A，#0AAH       ；接收数据出错，则发送出错应答信息AAH
        MOV     SBUF，A
        RETI                   ；中断返回
LOOP6：CLR      TI             ；是发送中断，则清TI标志
        RETI                   ；中断返回
        END
```

要确保两机的通信成功，至少要满足两个条件：一是要达成通信协议，如本例中甲机和乙机双方通信的约定；二是双方的波特率要一致。

5.3.4 89C51串行口方式0应用编程

89C51单片机的串行口工作在方式0状态下，使用移位寄存器芯片可以扩展一个或多个8位并行I/O口。所以，若串行口别无它用，就可用来扩展并行I/O口，这种方式不占用片外RAM地址，而且还能简化单片机系统的硬件结构。但缺点是操作速度较慢。因为扩展芯片越多，速率越慢。

1. 用74HC164扩展并行输出口

图5-13所示是利用两片74HC164扩展两个8位并行输出口的实用电路。74HC164是8位串入并出移位寄存器，由于其无输出控制端，故在串行输入过程中，输出端会不断地变化，所以，一般应在74HC164和输出装置之间加接输出控制门，以保证串行输入结束后再输出数据。

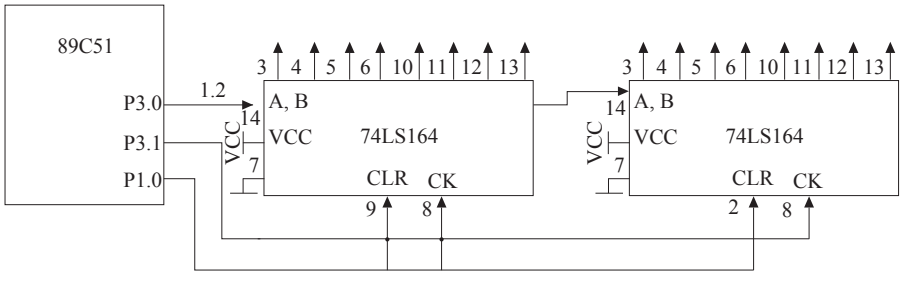


图 5-13 利用两片74HC164扩展两个8位并行输出口

下面的程序将内部RAM区中20H和21H单元的内容经串行口由74HC164并行送出。采用查询方式编程。

```
        ORG     0000H
```

```
        LJMP   START
        ORG    0030H
START:  MOV    SCON，#00H        ；设置串行口方式0
        MOV    R6，#02H          ；设置发送字节数
        MOV    R0，#20H          ；设置片内RAM指针
SEND:   MOV    A，@R0            ；取发送数据
        MOV    SBUF，A           ；启动串行口发送
WAIT:   JNB    TI，WAIT          ；未发送完一帧，等待
        CLR    TI                ；清发送中断标志
        INC    R0                ；指针加R0 1
        DJNZ   R6，SEND          ；未发送完转SEND
        SJMP   $                 ；发送完踏步等待。
        END
```

2. 用74HC165扩展并行输入口

图5-14所示是利用两片74HC165扩展两个8位并行输入口的实用电路。74HC165是可并行置入的8位移位寄存器。当移位 / 置入端S/$\overline{L}$由“1”变为“0”时，并行输入端的数据被置入各寄存器。当S/$\overline{L}$=“1”时，在时钟脉冲作用下，数据向Q_H方向移动，图中SIN为串行输人端。

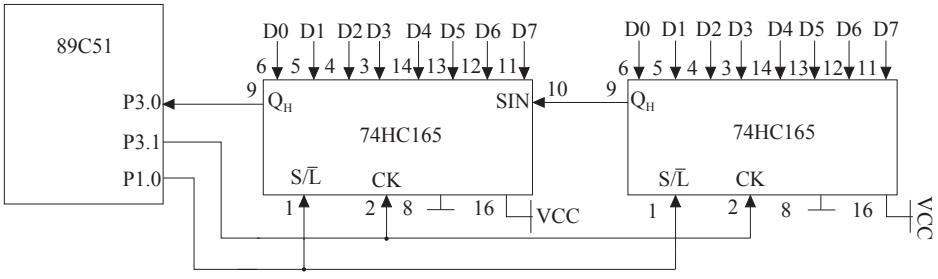


图 5-14 利用两片74HC165扩展两个8位并行输入口

下面的程序是从16位扩展口读入10组数据（每组两个字节），并把它们转存到内部RAM 40H开始的单元。采用查询方式编程。

```
        ORG    0000H
        LJMP   START
        ORG    0030H
START:  MOV    R6，#0AH          ；设置读入组数
        MOV    Rl，#40H          ；设片内RAM指针
RCV0:   CLR    Pl.0              ；并行置入数据
        SETB   Pl.0              ；允许串行移位
        MOV    R0，#02H          ；设置每组字节数
RCV1:   MOV    SCON，#10H        ；设为工作方式0，并启动接收
WAIT:   JNB    RI，WAIT          ；未接收完一帧数据，等待
        CLR    RI                ；清接收中断标志，准备下次接收
        MOV    A，SBUF           ；读入数据
```

```
MOV   @R1，A          ；送内部RAM区
INC   Rl              ；指向下一个地址
DJNZ  R0，RCVl        ；若未读完一组则继续
DJNZ  R6，RCV0        ；10组数据读完转RCV0
SJMP  $               ；10组数据读完踏步等待
END
```

5.4 89C51与PC机之间的通信

学习目标

1. 熟悉89C51与PC机通信实现电平转换的原因。
2. 熟悉MAX232芯片的应用及其与89C51串口的线路连接。
3. 了解89C51和PC机通信的基本方法和手段。

导　入

单片机特别适用于工业现场，但是单片机存在着功能简单且难于管理的缺点，因而在许多场合中，将PC机和单片机通过两者串口实现连接组成上、下位机系统，单片机负责对象的控制，PC机负责对单片机进行集中监控管理以及数据处理，这种应用的核心就是数据通信。本节讨论的主要内容是单片机和PC机之间实现串行通信的RS-232接口及其硬件线路的连接，以及简单的通信程序设计。

5.4.1 89C51与PC机通信的接口电路

在PC机系统内都装有异步通信适配器，该适配器的核心元件是可编程的串行通信接口芯片8250，对外采用的是RS-232C标准串行接口，利用该接口可以与89C51单片机实现异步串行通信。

由于89C51单片机输入/输出为TTL电平，而PC机的RS -232C接口采用的是非TTL电平，二者的电气规范不一致，因此，要完成PC机与89C51单片机之间的数据通信，89C51单片机的串口必须要配以电平转换电路，将TTL电平转换成RS -232C的电平 。

1. RS-232C接口

RS-232C是使用最早、应用最多的一种异步串行通信总线标准。它是美国电子工业协

会（EIA）1962年公布，1969年最后修定而成的。其中，RS表示Recommended Standard，232是该标准的标识号，C表示最后一次修定。

RS-232C主要用来定义计算机系统的一些数据终端设备（DTE）和数据电路终接设备（DCE）之间的电气性能。

RS-232C串行接口总线适用条件：设备之间的通信距离不大于15m，传输速率最大为20kb/s。

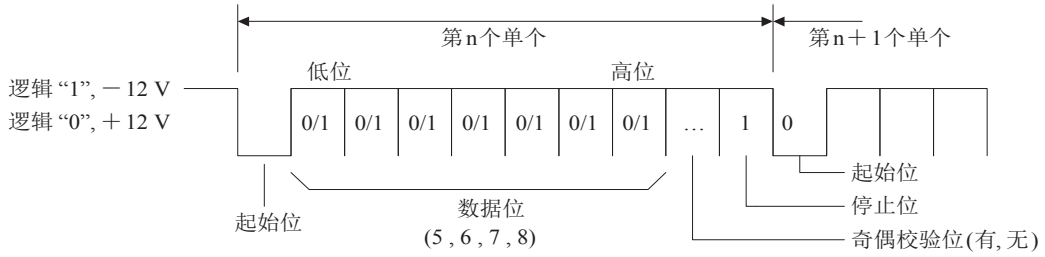


图 5-15 RS-232C信息格式

① RS-232C信息格式标准

RS-232C采用串行格式，如图5-15所示。该标准规定：信息的开始为起始位，信息的结束为停止位；信息本身可以是5、6、7、8位再加一位奇偶校验位。如果两个信息之间无信息，则写“1”，表示空。

② RS-232C电平转换器

RS-232C规定了自己的电气标准，由于它是在TTL电路之前研制的，所以它的电平不是＋5V和地，而是采用负逻辑，即逻辑“0”：＋5V~＋15V；逻辑“1”：－5V~－15V。因此，RS-232C不能和TTL电平直接相连，使用时必须进行电平转换，否则将使TTL电路烧坏，实际应用时必须注意！目前被广泛采用的电平转换电路是MAX232芯片。

MAXIM公司生产的，包含两路接收器和驱动器的IC芯片，适用于各种EIA-232C和V.28/V.24的通信。

MAX232芯片是通信接口。MAX232芯片内部有一个电源电压变换器，可以把输入的＋5V电源电压变换成为RS-232C输出电平所需的±10V电压。所以，采用此芯片接口的串行通信系统只需单一的＋5V电源就可以了。对于没有±12V电源的场合，其适应性更强。加之其价格适中，硬件接口简单，所以被广泛采用。

MAX232芯片的引脚结构如图5-16所示。其典型工作电路如图5-17所示。

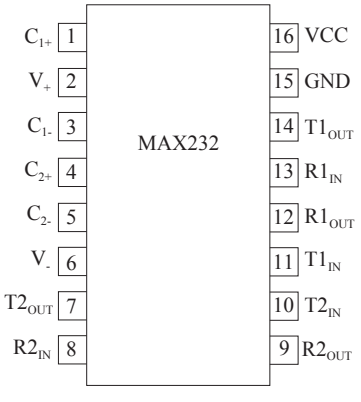


图 5-16 MAX232芯片引脚图

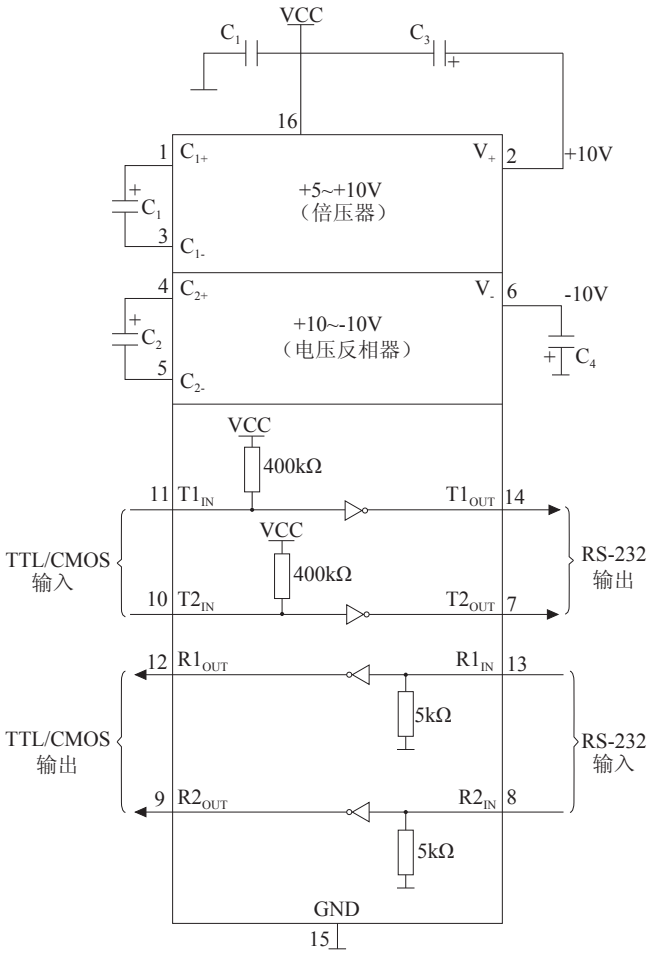


图 5-17 MAX232典型工作电路图

图5-17中，上半部分电容C_1、C_2、C_3、C_4及V+、V-是电源变换电路部分。在实际应用中，器件对电源噪声很敏感。因此，VCC必须要对地加去藕电容C_5，其值为0.1μF。电容C_1、C_2、和C_3取同样数值的钽电解电容1.0μF/16V，用以提高抗干扰能力，在连接时必须尽量靠近器件。下半部分为发送和接收部分。实际应用中，$T1_{IN}$和$T2_{IN}$可直接接TTL/CMOS电平的89C51单片机的串行发送端TXD；$R1_{OUT}$和$R2_{OUT}$可直接接TTL/CMOS电平的89C51单片机的串行接收端RXD；$T1_{OUT}$和$T2_{OUT}$可直接接PC机的RS-232串口的接收端RXD；$R1_{IN}$和$R2_{IN}$可直接接PC机的RS-232串口的发送端TXD。

2. 89C51与PC机串行通信的接口设计

图5-18中，89C51通过配置MAX232接口芯片完成电平转换，然后与PC的RS-232接口实现连接。这种连接称之为最简单的零调制3线经济型连接，也是进行全双工通信所必须的最少连线数目。

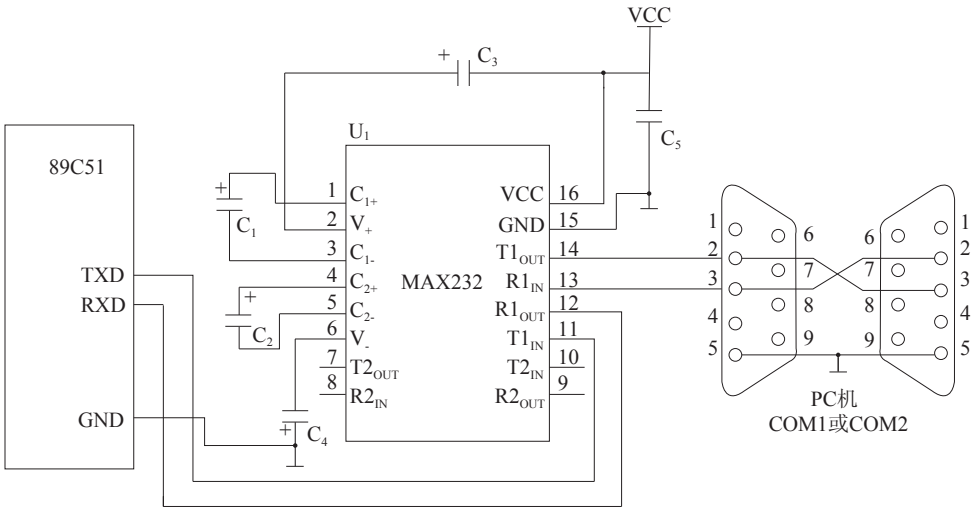


图 5-18　89C51与PC机的串行通信接口图

图5-18中，89C51与MAX232芯片中两路发送接收中任选一路作为接口。应注意其发送、接收的引脚要对应。如果使$T1_{IN}$接单片机的发送端TXD，则PC机的RS-232的接收端RXD一定要对应接$T1_{OUT}$引脚。同时，$R1_{OUT}$接单片机的RXD引脚，PC机的RS-232的发送端TXD对应接$R1_{IN}$引脚。

5.4.2　89C51与PC机通信编程

此处列举一个实用的通信测试软件，其功能为：将PC机键盘的输入发送给89C51单片机，89C51单片机收到PC机发来的数据后，回送同一数据给PC机，并在屏幕上显示出来。只要屏幕上显示的字符与所键入的字符相同，说明二者之间的通信正常。

通信双方约定：波特率为2400b/s；信息格式为8个数据位，无奇偶校验位。

1. 89C51单片机通信程序

通过中断方式接收PC机发送的数据，并回送。89C51单片机串行口工作在方式1，晶振为6MHz，波特率2400b/s，定时器T1按方式2工作，经计算，定时器预置值为X=F3H，SMOD=1。

程序如下：

```
        ORG   0000H
        LJMP  MAIN
        ORG   0023H
        LJMP  INTS              ；转串行口中断程序
        ORG   0030H
MAIN:   MOV   SCON，#50H        ；置串口工作方式1，允许接收
        MOV   PCON，#80H        ；SMOD=1
        MOV   TMOD，#20H        ；设置定时器T1为方式2
        MOV   TL1，#0F3H        ；设置T1初值
        MOV   TH1，#0F3H        ；8位重装初值
```

```
        SETB   TR1              ；启动定时器T1
        SETB   EA               ；允许串行口中断
        SETB   ES
        SJMP   $                ；等待中断
INTS：  JNB    RI，LOOP1        ；若不是接收中断，则转LOOP1
        CLR    RI               ；是接收中断，则先清RI
        MOV    A，SBUF          ；接收PC机发送的数据
        MOV    SBUF，A          ；将数据回送给PC机
        RETI
LOOP1： CLR    TI
        RETI
        END
```

2. PC机通信程序

PC机方面的通信程序可以用汇编语言编写，也可以用例如VC或VB高级语言来编写。这里只介绍用汇编语言编写的程序 。

程序如下：

```
Stack   Segment para stack 'code'   ；堆栈段定义
        db      256 dup(0)
Stack   ends
Code    Segment para public 'code'  ；代码段定义
Start   proc  far
        Assume cs:code,ss:stack
        PUSH   DS
        MOV    AX,0
        PUSH   AX
        CLI
INPUT:  MOV    AL,80H               ；置DLAB=1
        MOV    DX,3FBH              ；写入通信线控制寄存器
        OUT    DX,AL
        MOV    AL,30H               ；置产生2400 b/s波特率除数低位
        MOV    DX,3F8H
        OUT    DX,AL                ；写入除数锁存器低位
        MOV    AL,00H               ；置产生2400 b/s波特率除数高位
        MOV    DX,3F9H
        OUT    DX,AL                ；写入除数锁存器高位
        MOV    AL,03H               ；设置数据格式
        MOV    DX,3FBH              ；写入通信线路控制寄存器
        OUT   DX,AL
        MOV    AL,00H               ；禁止所有中断
        MOV    DX,3F9H
        OUT    DX,AL
```

```
WAIT1:  MOV   DX,3FDH          ；发送保持寄存器不空则循环等待
        IN    AL,DX
        TEST  AL,20H
        JZ    WAIT1
WAIT2:  MOV   AH,1             ；检查键盘缓冲区，无字符则循环等待
        INT   16H
        JZ    WAIT2
        MOV   AH,0             ；若有，则取键盘字符
        INT   16H
SEND:   MOV   DX,3F8H          ；发送键入的字符
        OUT   DX,AL
RECE:   MOV   DX,3FDH          ；检查接收数据是否准备好
        IN    AL,DX
        TEST  AL,01H
        JZ    RECE
        TEST  AL,1AH           ；判断接收到的数据是否出错
        JNZ   ERROR
        MOV   DX,3F8H
        IN    AL,DX            ；读取数据
        AND   AL,7EH           ；去掉无效位
        PUSH  AX
        MOV   BX,0             ；显示接收字符
        MOV   AH,14
        INT   10H
        POP   AXCMP AL,0DH     ；接到的字符若不是回车则返回
        JNZ   WAIT1
        MOV   AL,0AH           ；是回车则回车换行
        MOV   BX,0
        MOV   AH,14H
        INT   10H
        JMP   WAIT1
ERROR:  MOV   DX,3F8H          ；读接收寄存器，清除错误字符
        IN    AL,DX
        MOV   AL, '?'          ；显示‘？’号
        MOV   BX,0
        MOV   AH,14H
        INT   10H
        JMP   WAIT1            ；继续循环
        Start ends
        Code  ends
        End   start
```

知识拓展

89C51单片机的UART异步串行通信接口可以用普通的I/O线来模拟，而I/O模拟串口的关键在于程序设计。在串口资源缺乏时，采用此方法不失为一种有效的解决方法。若选择P1.0和P1.1两根I/O线作为模拟的TXD和RXD引脚，请思考编程方法，并编写发送程序和接收程序。

本章小结

89C51的串行口是一个通用异步全双工串口，可以同时进行数据的接收和发送。89C51的串行口通过对外的两条独立收、发信号线RXD（P3.0）、TXD（P3.1）与外界进行通信的。

89C51的串行口按不同工作方式，一帧位数有8位、10位和11位三种。一帧字符发送或接收完，置位标志位TI或RI，并申请中断。中断允许控制位ES，中断入口地址为0023H。

89C51的串行口含两个独立的接收、发送缓冲器SBUF。SBUF属于特殊功能寄存器。发送缓冲器SBUF只能写入不能读出，放待发的8位数据，写入SBUF便启动发送；接收缓冲器SBUF只能读出不能写入，放已成功接收的8位数据，供CPU读取。二者共用一个字节地址（99H）。

89C51的串行口控制寄存器SCON用来控制串行口的工作方式和状态。通过对SCON的SM0SM1两位的设置可选择四种工作方式。

① 方式0：SM0SM1=00，作同步移位寄存器用，这种方式常用于扩展I/O口。一帧8位，无启停位。数据从RXD端串行输入/输出，同步信号从TXD端输出。
波特率=$f_{osc}/12$。

② 方式1：SM0SM1=01，串行口被设置为10位通用异步通信接口UART。一帧10位：起始位＋8位数据位＋停止位。波特率=$(2^{SMOD}/32)\times$T1溢出率。

③ 方式2和方式3：SM0SM1=10/11，串行口被设置为11位通用异步通信接口UART。一帧11位：起始位＋9位数据位＋停止位。其中第9位数据位（D_8）在TB8或RB8中，常作奇偶校验位用或作多机通讯标识位（地址/数据）用。
方式2波特率=$f_{osc}\times 2^{SMOD}/64$，方式3波特率=$(2^{SMOD}/32)\times$T1溢出率。方式2和方式3除了波特率不同以外，其余完全相同。

串行口的波特率表明了串行口接收或发送二进制数的速率。它除了与振荡频率、PCON的SMOD位有关外，主要与定时器T1的初值设定有关。

由于89C51串行口的功能是由软件编程确定的，所以，一般在使用前都要对其初始化。初始化步骤如下：

① 设置控制字（对SCON寄存器设置）。

② 设置波特率（对SMOD和定时器T1的设置）。

③ 根据需要开启串行口中断（对IE寄存器设置）。

在串行通信的编程中，发送程序应注意先发送，再检查状态TI，再发送；而接收程序应注意先检查状态RI再接收，即发送过程是先发后查，而接收过程是先查后收。无论发送前或接收前，都应该先清状态TI或RI。无论是查询方式还是中断方式，发送或接收后都不会自动清状态标志，必须用程序清零TI和RI。

思考与练习题

一、单项选择题

1. 89C51串行口的中断服务程序入口地址是（　）。

A. 0003H　　B. 000BH　　C. 0013H　　D. 001BH

2. 89C51用来设置串行通信方式的控制寄存器是（　）。

A. IP　　B. TCON　　C. IE　　D. SCON

3. 某异步通讯接口的波特率为2 400，则该接口传送（　）。

A. 2 400位/秒　　B. 2 400字节/秒

C. 2 400位/分　　D. 2 400字节/分

4. 89C51串口方式1的波特率是（　）。

A. $f_{osc}/12$　　B. $f_{osc}/64$或$f_{osc}/32$

C. 由T1决定　　D. 由T1和SMOD共同决定

5. 串口发送器需要实现的数据转换是（　）。

A. 串→并数据转换　　B. 并→串数据转换

C. 串→串数据转换　　D. 并→并数据转换

二、判断题

1. 异步通信和同步通信的数据传输格式完全相同。（　）
2. 波特率反映了串行通讯的速率。（　）
3. 89C51串口方式0可以用来作异步通信。（　）
4. 89C51的串行口的发送中断与接收中断各自有自己的中断入口地址（　）
5. 能够完成异步通信的硬件电路称为UART。（　）

三、问答题

1. 并行通信与串行通信的区别和特点是什么？
2. 什么是串行异步通信？其帧格式是怎样的？
3. 什么是单工方式、半双工方式和全双工方式？
4. 89C51的串行口有几种工作方式？每种工作方式如何确定？每种工作方式的帧格式如何？
5. SCON中的SM2、TB8和RB8有何作用？

四、程序设计题

1. 若晶振为11.0592MHz，串行口工作于方式1，波特率为4 800b/s。写出用Tl作为波特率发生器的方式字和计数初值。
2. 设甲、乙两机采用方式1通信，波特率为4 800b/s。甲机发送10个字符，乙机接收存放在内部RAM以20H为首址的单元，试用查询方式编写甲、乙两机的程序。设两机的晶振f_{osc}=6MHz。
3. 若晶振为11.0592MHz，串行口工作于方式3，波特率为2 400b/s，第9位数据为奇校验

位。试编制一个程序，对串行口初始化，并用查询方式接收串行口上输入的16个字符存于片内RAM从30H单元开始的区域中。若对RB8校验出错，则停止接收，并使P1.0的输出为0；若正确地接收到16个字符，则停止接收，并使P1.1的输出为0。

4. 试设计一个89C51单片机的双机通信系统，并编制程序将甲机片内RAM 20H~3FH单元的数据块通过串行口传送到乙机的片内RAM 40H~5FH单元中去。
5. 利用89C51串行口设计4位静态数码管显示器，画出电路并编写程序，要求4位显示器上每隔1s交替地显示“0123”和“4567”。

实验项目5　串行口的使用及编程方法

一、实验目的

1. 进一步加深理解89C51串行口的基本工作原理以及RS-232芯片的作用。
2. 通过89C51串行口与PC机的基本通信实验，牢固掌握89C51串行口的通信程序设计要点和编程技巧。
3. 掌握STC_ISP_V480在系统编程软件内嵌串口调试器的操作。
4. 在实践中，切实地提高单片机应用的操作技能和动手能力，独立思考问题、分析问题和解决问题的能力。

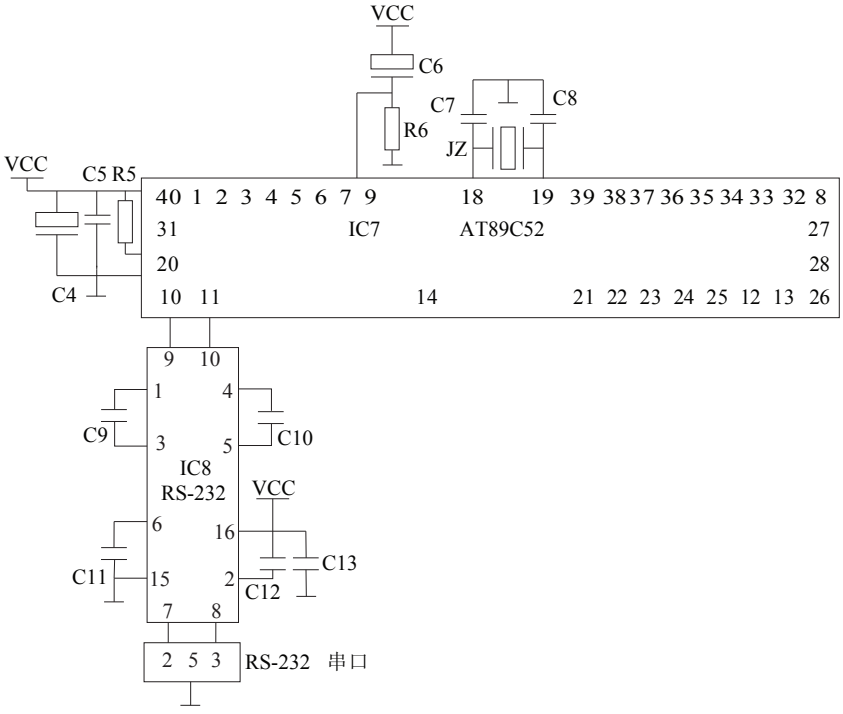
二、实验设备

1. PC计算机一台
2. 单片机实验装置一套（教材配套）
3. Keil μVision2 集成开发环境
4. STC_ISP_V480在系统编程软件

三、实验内容

通过RS-232接口，实现单片机和PC机之间的基本通信。具体要求是：PC机串行口先发送一个大写字母U(ASCII码为55H)，在单片机接收到55H时返回一个41H（ASCII码为字母A）。在PC机一端，接收到41H表示完成。双方的通信波特率为9 600b/s。

1. 实验线路



2. 实验程序

```
        ORG   0000H
        LJMP  MAIN
        ORG   0030H
MAIN:   MOV   SP，#60H
        MOV   TMOD，#20H
        MOV   TH1，#0FDH
        MOV   TL1，#0FDH
        MOV   SCON，#50H
        MOV   PCON，#00H
        SETB  TR1
REC:    JBC   RI，SEWT
        AJMP  REC
SEWT:   MOV   A，SBUF
        CLR   RI
        CJNE  A，#55H，REC
        MOV   SBUF，#41H
        SJMP  $
        END
```

四、实验操作

1. 熟悉实验线路的基本工作原理，尤其是RS-232芯片的作用。
2. 在理解源程序的基础上，完成实验程序的运行。
3. 打开Keil μVision2集成开发环境，对源程进行编辑，然后汇编直至通过，最终生存HEX文件。
4. 完成PC机和单片机实验装置的连接，打开STC_ISP_V480在系统编程软件，将HEX文件下载到单片机中。
5. 运行单片机程序。
6. 打开STC_ISP_V480在系统编程软件内嵌串口助手调试器并完成必要的设置（包括COM端选择、波特率=9 600b/s、校验位=N、数据位=8、停止位=1、并选择字符格式发送）。
7. 在串口助手窗口的单字符串发送区输入一个大写字母的U（ASCII码为55H），然后点击发送字符/数据按钮，此时PC机先向单片机发送一个U，这时可以在串口助手的接收/键盘发送缓冲区窗口看见由单片机向PC机回送一个A（ASCII码为41H）。
8. 修改程序，使用中断方法编程，并完成调试和运行。

第6章 单片机人机通信接口

输入和输出是单片机人机通信的重要接口。在实际应用中，几乎每个单片机系统都会有输入和输出部分。单片机系统使用比较普遍的输入设备有按键或者键盘等，而使用比较普遍的输出设备 有LED数码管和液晶显示器等。本章主要围绕这些常用的输入和输出设备，从原理、器件、电路和编程方面展开详细介绍。

6.1 键盘接口

学习目标

1. 理解按键去抖动的重要性。
2. 掌握独立式键盘和行列式键盘的结构形式及判键原理。
3. 掌握键盘控制程序的设计要点。

导 入

输入/输出设备是计算机系统的重要组成部分，作为输入设备的键盘在计算机应用中是不可或缺的。键盘在单片机应用系统中，实现输入数据、传送命令的功能，是人工干预的主要手段。合理的键盘设计，不仅可以节省系统的设计成本，还可以使仪器设备的操作变得更为简单、方便，从而在很大程度上提高了系统的综合性能。

6.1.1 按键去抖动

单片机中使用的按键开关通常是机械构件开关。在操作时，由于机械触点的弹性作用，按键开关在闭合时不会马上稳定地接通，在断开时也不会一下子断开，总会出现“抖动”现象。抖动时间的长短由按键的机械特性决定，一般在5~l0ms。由于按键的抖动会造成一次按键被CPU误以为按了多次，为了确保CPU对按键的一次操作仅做出一次处理，必须要去除按键抖动。

按键去抖通常有两种方法：一种是硬件去抖，另一种是软件去抖。

1. 硬件去抖

硬件去抖通常采用基本RS触发器来实现。电路原理图如图6-1所示，假设开关S处于A和B之间，即既不与A接触，也不与B接触时的状态为C。由基本RS触发器的特性可知，开关仅与A或B接触时才会改变触发器的状态，处于C时将维持RS触发器的状态。而开关的抖动仅发生在A与C（或B与C）之间，不影响触发器的输出，从而消除了抖动的影响。

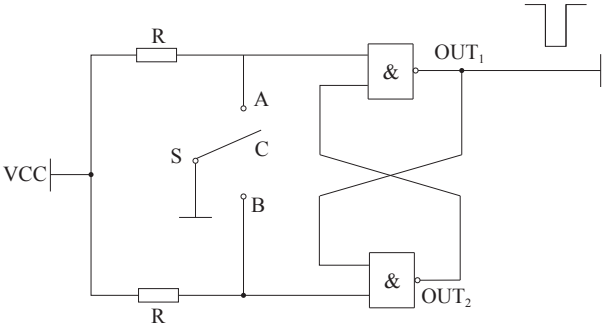


图 6-1 硬件去抖电路

虽然硬件去抖从根本上避免了电压抖动的产生，但在实际应用中较少采用，原因是在按键较多的情况下，每个按键都要配置一个RS触发器，一方面成本高，另一方面印板线路复杂。

2. 软件去抖

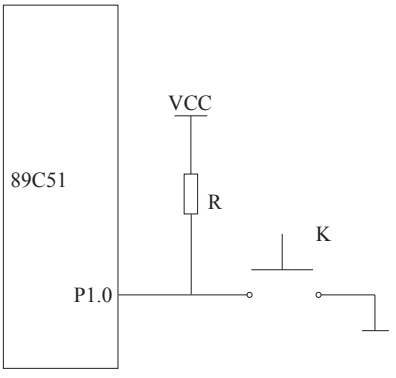


图 6-2 按键电路

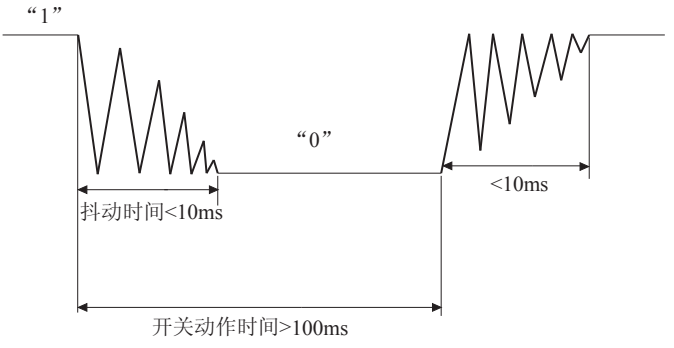


图 6-3 按键抖动

图6-2给出是在89C51单片机的P1.0引脚上连接一个常开开关的按键电路。当按键K未被按下时，P1.0输入为高电平；而按下后，P1.0输入为低电平。由于按键操作会有抖动，实际造成P1.0输入端的电平在按键闭合和断开瞬间会有反复抖动，其波形图如图6-3所示。按键的这种抖动对于人来说是感觉不到的，但对CPU来说，则完全能感应得到。因为CPU处理的速度是在微秒级，而机械抖动的时间至少是毫秒级，对CPU而言，这已是一个“漫长”的时间了。为了使CPU能正确地读出P1.0引脚的状态，对每一次按键只作一次响应，可采用软件去除抖动的办法。

软件去抖就是在CPU获得P1.0引脚为低电平信息后，不是立即认定按键K已被按下，而

是延时10ms或更长一些时间后再次检测P1.0引脚，如果仍为低电平，说明按键K的确按下了，这实际上是避开了按键按下时的抖动时间。而在检测到按键释放后（P1.0为高电平），也要给出10ms延时，消除后沿的抖动。至于CPU对按键的响应处理，可以放在按键释放前处理，也可以放在按键释放后处理。这样就能确保每次按键只作一次响应处理。图6-4给出的是软件去抖的按键处理程序流程图。

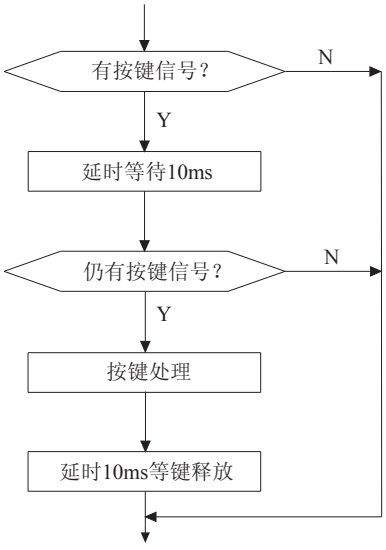


图 6-4　软件去抖键处理流程图

由于操作人员的按键动作一般最快也要用时100ms以上，即便去掉前后各10ms延时时间，按键闭合稳定时间还剩80ms左右，只要CPU每次扫描按键的时间间隔控制在100ms以内，绝对不会造成对按键操作的漏判。如果出现漏判按键，一定是CPU查询按键动作的时间间隔过长。

软件去抖的方法简单可行，而且不需要增加硬件电路，成本低，因而被广泛采用。

6.1.2　键盘结构及编程

键盘是一组按键的集合，它是单片机系统中最常用的人机通信输入设备。用户通过键盘可以向CPU输入数据和命令。

键盘按其结构形式可分为编码式键盘和非编码式键盘两大类。编码式键盘是由其内部硬件逻辑电路自动产生按键的编码。这种键盘使用方便，但价格较贵；非编码键盘是由软件来实现键盘的定义与识别，由于其经济实用，较多地应用于单片机系统中。

键盘按组成结构又可分两种：独立式键盘和行列式（矩阵式）键盘。

1. 独立式键盘

独立式键盘是指键盘中的每个按键都独立地占用一根I/O线，彼此互不相关。如图6-5所示。这种键盘结构的优点是电路简单、按键配置灵活、编程容易；缺点是如果按键多时，单片机的I/O线占用也较多。因此在实际应用中，在按键数量较少的情况下，采用这种键盘结构是一种不错的选择。

独立式键盘的判键原理十分简单。对于图6-5所示的键盘接口电路而言，要判别键盘中是否有键按下，CPU要把整个P1口的8根I/O线的状态全部读进来作判断分析。如果P1=FFH，肯定没有任何一个键按下；如果P1≠FFH，肯定有键按下。当有键按下并要判断具体是哪个键按下时，要用89C51的位处理指令对每一根I/O线依次作判别，判到哪根I/O线是低电平，则对应的按键一定是按下的；反之，就没有按下。

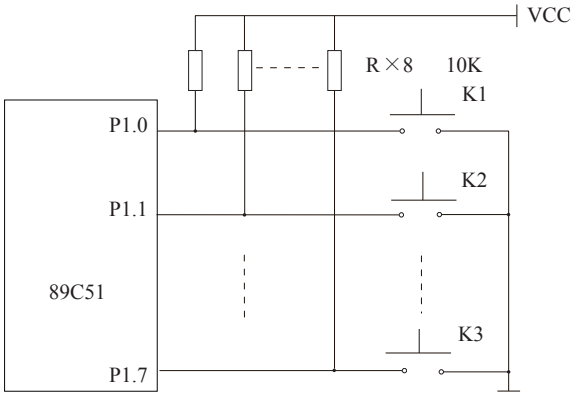


图 6-5　独立式键盘

图6-5所示为独立式键盘的键处理程序，程序中包括键查询、去键抖动、键功能程序转移。

程序清单：

```
        ORG    0000H
        LJMP   MAIN
        ORG    0030H
MAIN:   MOV    P1，#0FFH          ；P1口设置输入用
        MOV    A，Pl              ；键盘状态输人
        CJNE   A，#0FFH，MAIN     ；无键转MAIN
        LCALL DELAY10ms           ；有键调用10ms延时子程序
        MOV    A，Pl              ；键键状态输人
        CJNE   A，#0FFH，MAIN     ；确有键按下往下执行，无键转MAIN
PLl:    JNB    ACC.0，P0F         ；0号键按下转P0F
        JNB    ACC.1，P1F         ；1号键按下转P1F
        JNB    ACC.2，P2F         ；2号键按下转P2F
        JNB    ACC.3，P3F         ；3号键按下转P3F
        JNB    ACC.4，P4F         ；4号键按下转P4F
        JNB    ACC.5，P5F         ；5号键按下转P5F
        JNB    ACC.6，P6F         ；6号键按下转P6F
        JNB    ACC.7，P7F         ；7号键按下转P7F
        SJMP   MAIN               ；无键转MAIN
P0F:    LCALL FUNC0               ；调用0号键功能处理子程序
        SJMP   EXIT               ；转EXIT
P1F:    LCALL FUNC1               ；调用1号键功能处理子程序
        SJMP   EXIT               ；转EXIT
        ⋮
        ⋮
P7F:    LCALL FUNC7               ；调用7号键功能处理子程序
EXIT:   LCALL  DELAY10ms          ；延时10ms等键释放
        SJMP   MAIN               ；返回MAIN
FUNC0:  ...                       ；0号键功能处理子程序
        RET
FUNC1:  ...                       ；1号键功能处理子程序
        RET
        ⋮
        ⋮
FUNC7:  ...                       ；7号键功能处理子程序
        RET
DELAY10ms:...                     ；10ms延时子程序
        RET
        END
```

注意：程序中先判的键优先级高。

2. 行列式（矩阵式）键盘

行列式键盘又叫矩阵式键盘，用I/O口线组成行、列结构，按键设置在行列的交叉点上。在行列式键盘中，每条水平线和垂直线在交叉处不直接连通，而是通过一个按键加以连接。如图6-6所示，一个8位I/O口就能组成4×4=16个按键，与同样使用8根I/O线的独立式键盘结构相比按键数量多出了一倍，而且I/O线数越多，区别就越明显，比如再多加一条I/O线就能组成20键的键盘。由此可见，在需要的按键数比较多时，为了节省I/O线，采用行列式键盘结构是一种合理的选择。

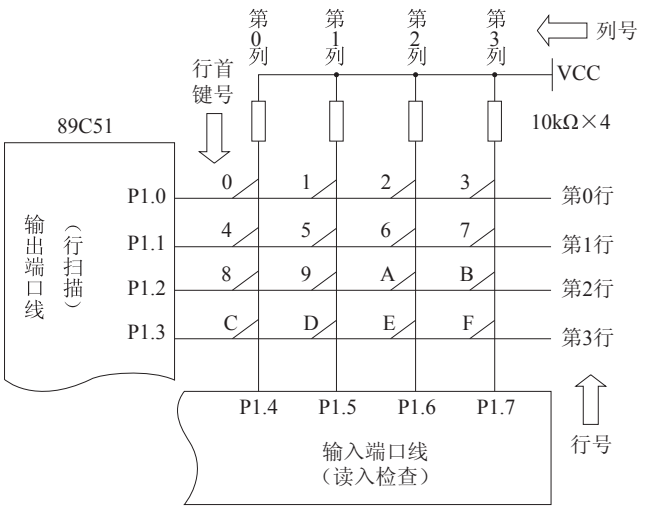


图 6-6　行列式键盘

行列式键盘的判键原理要复杂一些，因此编程也较为复杂。硬件设计规定每一根列线都要通过上拉电阻接正电源。编程时，将行线所接的单片机的I/O线作为输出用，而列线所接的I/O线则作为输入用。对于键的识别通常采用“行扫描”法。

行扫描法的判键通常分两步走，具体方法如下。

第一步：首先判断键盘中有无键按下。单片机将全部行线输出低电平，然后读列线状态，若所有列线均为高电平，则键盘中肯定无键按下；只要有其中的一根列线的电平为低电平，则表示键盘中肯定有键被按下。

第二步：在确认有键按下后，判断具体是哪个键按下。其方法是：单片机依次将行线置为低电平，然后检查所有列线的状态。若列线全是高电平，则所按下的键不在此行；若列线中有低电平，则所按下的键必在此行，而且一定是在行、列线同为0的相交点上。

在行列式键盘的接口电路，每个键闭合时都可以根据行、列数据组合得到一个对应的值，该值称为位置码，而且每个键的位置码都是唯一的，互不重复。图6-6中16个键的位置码依次是：11101110、11011110、10111110、01111110、……、01110111。在编程时，位置码可以用来区分每个按键。

用户人为地对每个键所编的位置序号称为键号。图6-6中16个键的键号依此是：0、1、2、3、……、F，键号与位置码在使用上并不矛盾，彼此对应，程序可以把位置码转换成键号，使用键号为程序散转提供了方便。

下面给出图6-6所示行列式键盘的键盘扫描子程序。该子程序的出口参数放入寄存器B中：若有键按下，B=位置码；若没键按下，B=FFH。

由于程序设计较为复杂，先给出程序流程图的设计，然后按流程图再设计程序。程序流程图设计如图6-7所示。

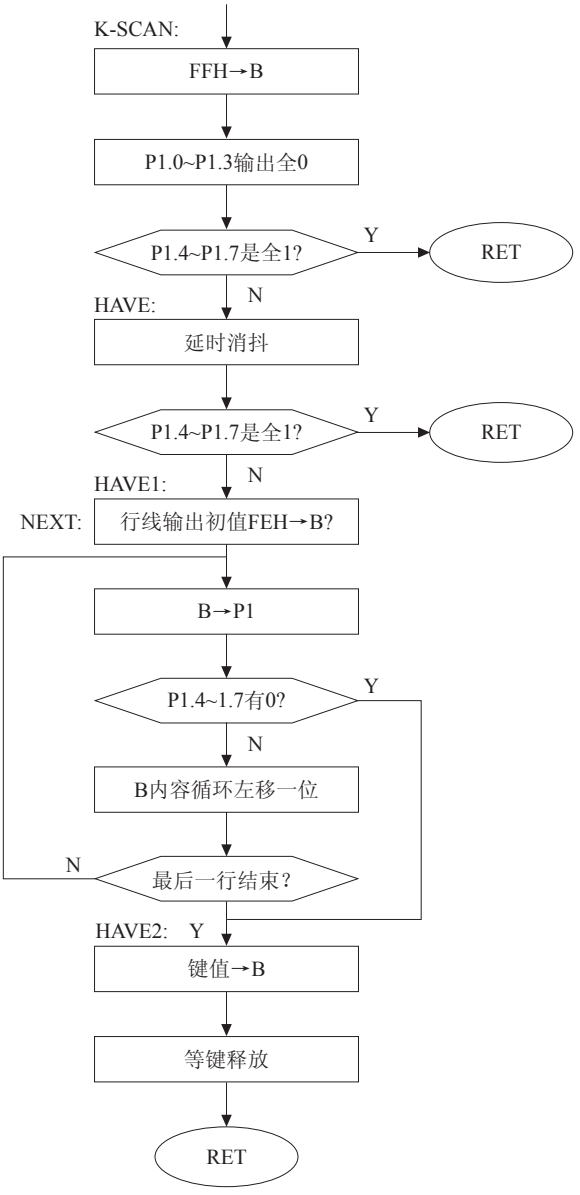


图 6-7 程序流程图

程序设计如下：

```
K-SCAN：MOV   B，#0FFH
        MOV  P1，#0F0H
        MOV   A，P1
        ANL   A，#0F0H
        CJNE  A，#0F0H，HAVE
        RET
HAVE：  LCALL DLY10MS
        MOV   A，P1
        ANL   A，#0F0H
```

```
            CJNE   A，#0F0H，HAVE1
            RET
HAVE1：     MOV    B，#0FEH
NEXT：      MOV   P1，B
            MOV    A，P1
            ANL    A，#0F0H
            CJNE   A，#0F0H，HAVE2
            MOV    A，B
            RL A
            MOV    B，A
            CJNE   A，#0EFH，NEXT
HAVE2：     MOV    A，B
            ANL    A，#0FH
            MOV    B，A
            MOV    A，P1
            ANL    A，#0F0H
            ADD    A，B
            MOV    B，A
            MOV   P1，#0F0H
NEXT1：     MOV    A，P1
            ANL    A，#0F0H
            CJNE   A，#0F0H，NEXT1
            RET
DLY10MS：                          ；10ms延时子程序
            ⋮
            RET
```

6.1.3 键盘控制程序设计要点

键盘中所定义的键一般分两类：数字键和功能键。

数字键操作是要求向CPU输入一个具体数字。当CPU扫描到数字键键码后，将其转换成相应的数字作数字信息用或显示。

功能键操作是要求CPU去完成功能键所指定的任务。当CPU扫描到功能键键码后，应准确无误的找到该功能键的功能处理程序加以执行。

如图6-8给出的是键盘控制程序设计的流程图。

一个完善的键盘控制程序应具备以下功能：

① 检测有无按键按下，并采取硬件或软件措施，消除键盘按键机械触点抖动的影响。

② 有可靠的逻辑处理办法。每次只处理一个按键，其间任何按键的操作对系统不产生影响，且无论一次按键时间有多长，系统仅执行一次按键功能程序。

③ 准确输出键码（或键号），以满足跳转指令要求。

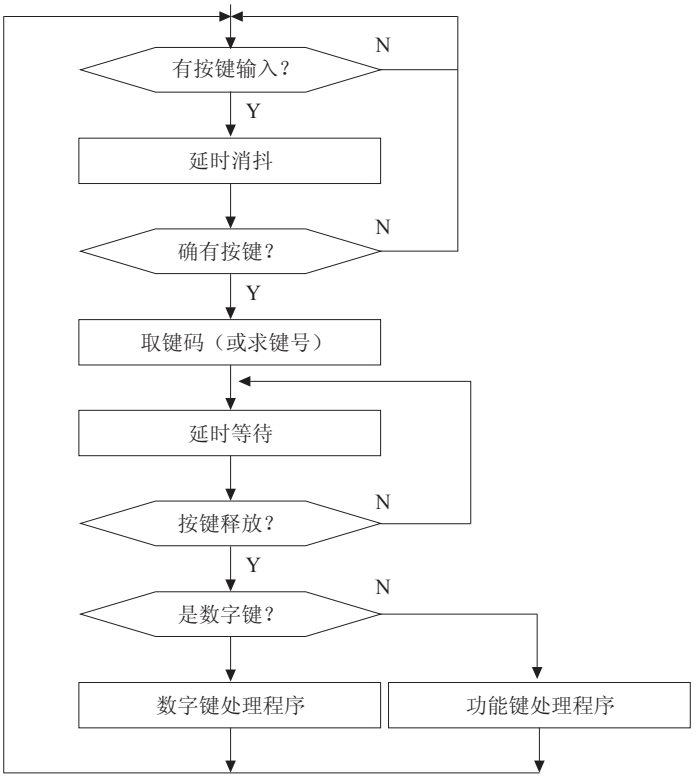


图 6-8 键盘控制程序流程图

6.1.4 键盘的工作方式

CPU对键盘的响应取决于键盘的工作方式，键盘的工作方式应根据实际应用系统中CPU的工作状况而定，其选取的原则是既要保证CPU能及时响应按键操作，又不要过多占用CPU的工作时间。通常键盘的工作方式有三种：编程扫描、定时扫描和中断扫描。

1. *编程扫描方式*

编程扫描方式是CPU对键盘的扫描采取程序控制方式。一旦进入键扫描状态，则反复地扫描键盘，等待用户从键盘输入命令和数据。而在执行键入命令或处理键入数据过程中，CPU不再响应键入要求，直到CPU返回重新扫描键盘为止。采用键盘扫描方式，CPU经常会处于空扫描状态。

2. *定时扫描方式*

定时扫描工作方式是利用单片机内部定时器产生定时中断（如20ms），CPU响应中断后对键盘进行扫描的一种工作方式。在有键按下时识别出该键并执行该键对应的功能程序。定时扫描工作方式的键盘硬件电路与编程扫描工作方式相同。

3. *中断扫描方式*

中断扫描方式是：当无键按下时，CPU不主动扫描键盘；有键按下时，键盘向CPU发一个中断请求信号，CPU响应中断后，转去执行键盘扫描中断服务程序。中断工作方式与前两种工作方式相比大大提高了CPU的工作效率。

中断扫描方式的键盘电路接口设计要使用到外部中断资源，如图6-9所示。编程时，要求四根行线（P1.0~P1.3）平时处于低电平状态。当无键按下时，与门各输入端均为高电

平，保持与门输出端为高电平；当有键按下时，$\overline{\text{INT0}}$端为低电平，向CPU申请中断，CPU转去执行键盘扫描中断服务程序。

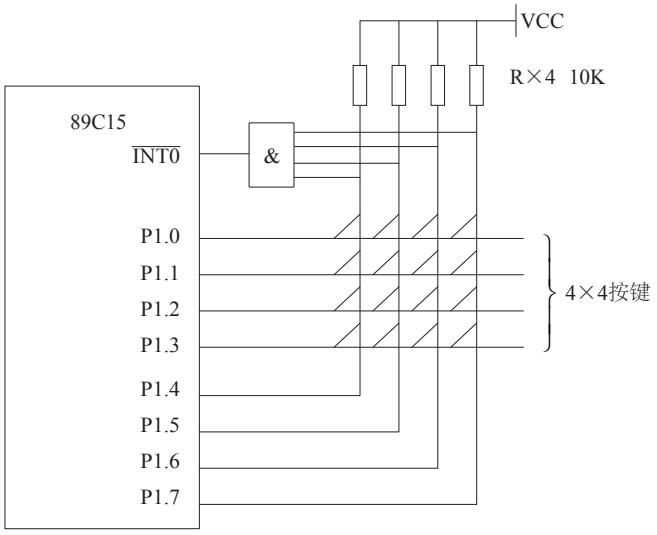


图 6-9 中断方式键盘接口

6.2 LED数码显示器接口

学习目标

1. 理解LED数码管的结构和原理以及使用要点 。
2. 理解LED数码管静态显示和动态显示的区别和优缺点。
3. 掌握LED数码管显示程序的设计要点。

导　入

LED数码管在基于单片机的仪器仪表中有着广泛的用途，主要是显示单片机的输出数据、状态等，因而，作为外围典型器件，LED数码管显示是反应系统输出和操纵输入的有效器件。LED数码管具备数字接口，可以很方便地和单片机系统连接；LED数码管具有价格低廉，性能稳定，显示清晰，亮度高，使用电压低，寿命长等特点。

6.2.1 LED数码显示器的结构及原理

LED数码显示器是一种由LED发光二极管组合成的显示字符的显示器件。它使用了八个LED发光二极管，其中七个发光二极管以段（a~g段）的形式构成“8”字轮廓，用于显示字符；另一个发光二极管以点（dp）的形式出现，用于显示小数点。通常LED数码显示器称之为7段（也有称作8段）发光二极管数码显示器。其外形和管脚定义如图6-10所示。

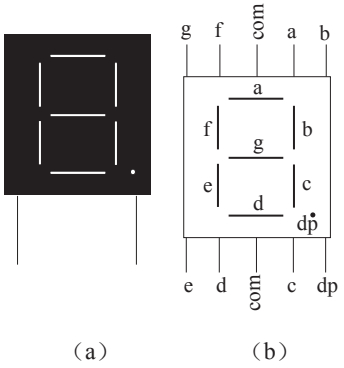


图 6-10 LED外形和管脚

（a）外形图 （b）管脚图

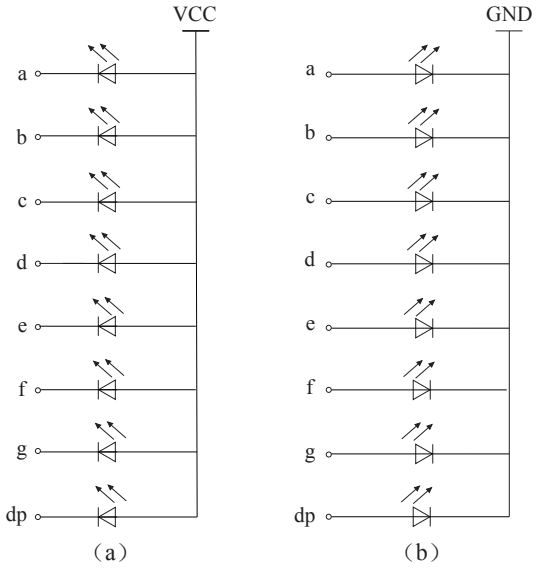


图 6-11 LED内部电路

（a）共阳极 （b）共阴极

LED数码显示器内部有两种连接方法：一种是共阳极接法，另一种是共阴极接法。其内部电路如图6-11所示。

共阳极接法是在内部把发光二极管的阳极连在一起构成公共阳极（com端）。使用时公共阳极接+5V，每个发光二极管的阴极向外引出，通过限流电阻与I/O线相连。

共阴极接法是把发光二极管的阴极连在一起构成公共阴极（com端）。使用时公共阴极接地，每个发光二极管的阳极向外引出，通过限流电阻与I/O线相连。

LED数码显示器可以用来显示数字0~9、带小数点的数字0.~9.、少数几个英文字母或符号“—” 等。为了显示字符，要为LED显示器提供显示段码（或称字形码），组成一个“8”字形字符的7段，再加上一个小数点位，共计8段。因此提供给LED显示器的显示段码为一个字节。假设各显示段（a~dp） 在一字节中的位置如图6-12所示，据此可以分析出如表6-1中给出的各显示字符的显示段码。

段码位	D7	D6	D5	D4	D3	D2	D1	D0
显示段	dp	g	f	e	d	c	b	a

图 6-12 a~dp显示段在一字节中的位置

表 6-1 LED显示段码

显示字符	共阴极段过码	共阳极段选码	显示字符	共阴极段选码	共阳极段法玛
U	3FH	00H	B	7CH	83H
1	06H	F9H	C	39H	C6H
2	5BH	A4H	D	5EH	A1E
3	4FH	B0H	E	79H	86H
4	66H	99H	F	71H	84H
5	6DH	92H	P	73H	82H
6	7DH	82H	U	3EH	C1H
7	07H	F8H	H	76H	89E
8	7FH	80H	•	80H	7FH
9	6FH	90H	8 •	FFH	00H
A	77H	88H	暗	00H	FFH

注意：段码是相对的，它由各字段在字节中所处位决定。在实际应用中，显示段码与显示器和I/O线的具体连接有关。因此，显示段码的确定只有在硬件线路设计定稿后才能进行。

6.2.2 LED数码显示器的接口及显示方式

对于N位LED显示器来讲，有N根位选线（COM脚）和8×N根段选线（a~dp脚）。位选线用来控制显示位的亮暗，段选线用来控制字符选择。依据位选线和段选线连接方式的不同，LED显示器有静态显示和动态显示两种方式。

1. 静态显示方式

静态显示方式的各位数码管相互独立，公共端恒定接地（共阴极）或接正电源（共阳极）。每个数码管的八个段选线（a~dp）分别与一个8位I/O口地址相连，如图6-13所示。该图显示了一个4位静态LED显示器电路，显示器的每一位可独立显示，只要在该位的段选线上保持段选码电平，该位就能保持相应的显示字符。由于每一位由一个8位输出口控制段选码，故在同一时刻各位可以显示不同的字符。

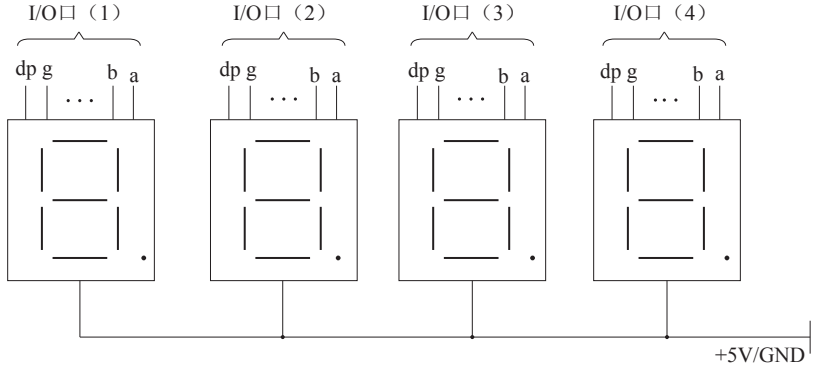


图 6-13 静态显示电路

采用静态显示方式，占用CPU时间少，编程简单，显示便于监测和控制。这是因为CPU将所要显示的数据送出后就不再控制LED显示器，直到下一次显示时再传送一次新的

显示数据。静态显示的缺点是占用I/O口线资源太多。对于N位静态显示器要求有N×8根I/O口线，硬件电路复杂，成本高，故在位数较多时往往采用动态显示方式。

2. *动态显示方式*

动态显示是一位一位地轮流点亮各位数码管，这种逐位点亮显示器的方式称为位扫描。通常，各位数码管相应的段选线并联在一起，由一个8位的I/O口控制；各位的位选线（公共阴极或阳极）由另外的I/O口线控制，如图6-14所示，其显示了一个4位动态LED显示器电路。

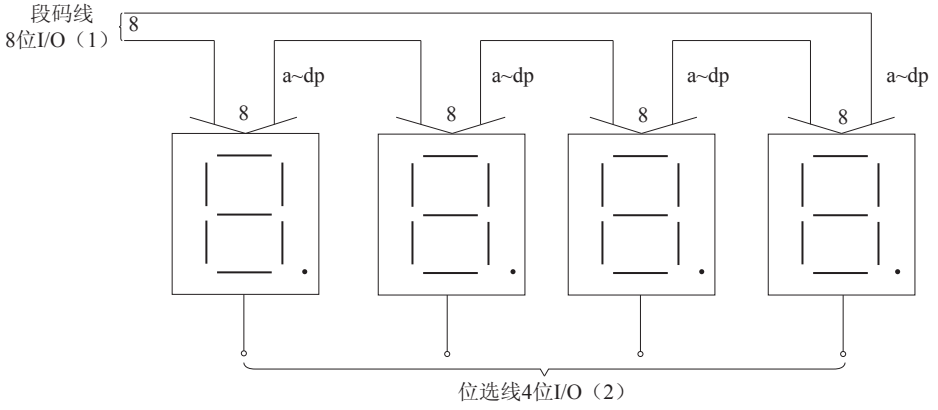


图 6-14 动态显示电路

动态方式显示时，各数码管分时轮流选通，要使其稳定显示，必须采用扫描方式，即在某一时刻只选通一位数码管，并送出相应的段码，在另一时刻选通另一位数码管，并送出相应的段码。依此规律循环，即可使各位数码管显示将要显示的字符。虽然这些字符是在不同的时刻分别显示，但由于人眼存在视觉暂留效应，只要每位显示的时间间隔足够短就可以给人以同时显示的感觉。

动态显示方式在使用时需要注意如下三个方面的问题。

① 显示扫描的刷新频率。每位轮流显示一遍称为扫描（刷新）一次，只有当扫描频率足够高时，人眼才不会觉得闪烁。对应的临界频率称为临界闪烁频率。临界闪烁频率跟多种因素相关，一般认为大于24Hz即可。

② 显示器的亮度问题。通常显示器件从导通到发光有一定的延时，导通时间太短，发光太弱。而这一延时参数决定了动态显示时所能接数码管的极限数目。通常，位线信号为一脉冲信号，该位数码管的亮度是与位线脉冲占空比相关的。

③ LED显示器的驱动问题。LED驱动器驱动能力的高低是直接影响显示器亮度的又一重要因素。驱动能力越强，通过发光二极管的电流就越大，显示亮度也就越高。通常一定规格的发光二极管有相应的额定电流要求，这就决定了段驱动器的驱动能力，而位驱动电流则应为各段驱动电流之和。从理论上看，对于同样的驱动器而言，N位动态显示的亮度不到静态显示亮度的1/N。

采用动态显示方式比较节省I/O口线。对于N位动态显示器要求有N+8根I/O口线，硬件电路也比静态显示方式简单，但由于每隔一定时间必须要扫描一次，因此占用CPU时间较多，管理复杂，编程较难。

6.2.3 LED数码显示器的显示程序设计

无论采用静态显示方式还是采用动态显示方式，LED数码显示器的显示程序设计，一般可以归纳出如下几个设计要点：

① 根据所设计的显示接口电路，分析和确定各显示字符的显示段码，在程序存储器ROM中建立段码表。段码表通常用数据定义伪指令DB来建立，并放在显示程序之后。

② 为每个显示位设置一个显缓单元。显缓单元是用来存放待显示字符在段码表中的段码位置号（偏移量）。

③ 对于某显示位的字符显示，先取出其显缓单元中的位置号，再用查表指令MOVC查段码表找出该显示字符的段码。

④ 将找出的段码输出到连接段选线的端口。

【例6-1】用P1口设计一个1位共阴LED的静态显示电路并显示子程序。

本例的硬件接口电路设计比较简单，如图6-15所示。图中将LED的8个段引脚通过限流电阻与P1口8根I/O线作连接，共阴LED的COM引脚必须接地才能点亮。

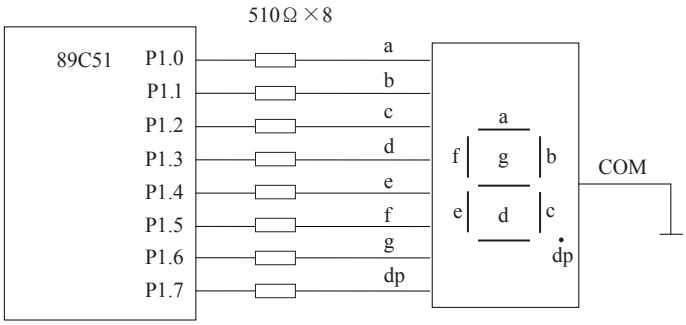


图 6-15　1位共阴LED的静态显示电路

静态显示子程序设计如下：

```
DIS:    MOV   DPTR，#WTAB      ；指向段码表首地址
        MOV   A，DIS_BUFFER    ；取显缓单元中位置号
        MOVC  A，@A+DPTR       ；查表找段码
        MOV   P1，A            ；段码输出，显示字符
        RET                    ；返回
WTAB:   DB    3FH              ；段码表‘0’（显示字符）——0（位置号）
        DB    06H              ；‘1’（显示字符）——1（位置号）
        DB    5BH              ；‘2’（显示字符）——2（位置号）
        ⋮
```

如果要在图6-15的LED显示器上显示字符‘1’，先对DIS_BUFFER显缓单元设置显示字符‘1’在段码表中的段码位置号（=1），然后对上面的子程序进行调用，程序如下。

```
        ⋮
        MOV  DIS_BUFFER，#1
        LCALL        DIS
        ⋮
```

【例6-2】用P1和P2口设计一个6位共阳LED的动态显示电路并显示子程序。

本例的硬件接口电路设计如图6-16所示。其中P2口的8根I/O线作为段选线的接口使用，用来控制字符选择；P1口的6根I/O线作为位选线的接口使用，用来控制各显示位的亮暗。这里要注意的是：由于使用的是共阳LED，因此，当P1口的I/O线输出0时，经反向驱动电路变为1，此时LED点亮，反之LED则变暗。图中的八个电阻用于LED的限流。

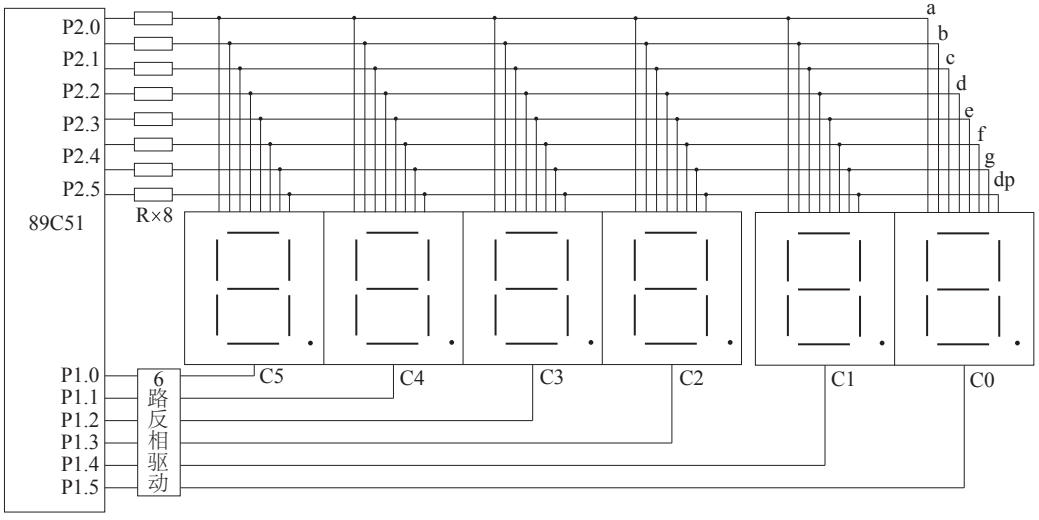


图 6-16 6位共阳LED的动态显示电路

由于动态显示方式接口电路不能在同一时刻显示不同的字符。只能采用动态扫描显示方法，轮流点亮各位数码管，利用人眼的视觉残留效应，使其看起来像是同时显示不同的字符一样。

显示程序设计要考虑为每个显示位设置一个显缓单元。6位LED共开设六个显缓单元，组成一个显示缓冲区dis_buffer0~dis_buffer5，如图6-17所示。

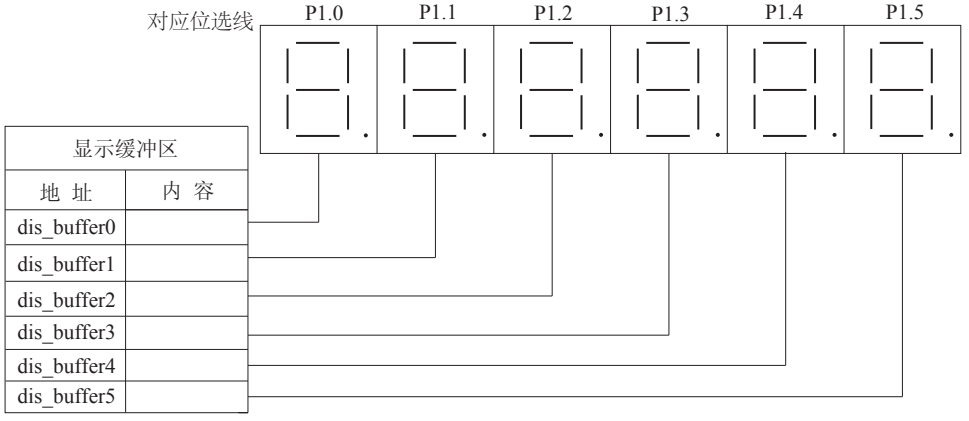


图 6-17 显示缓冲区与各位LED对应关系

本显示程序设计的关键是在轮流点亮各位LED时，一方面位选线控制输出和显缓单元使用的配合要对应好；另一方面要确保每一位有一定的点亮时间，一般每位显示在1ms左右。

由于动态显示子程序设计比较复杂，此处先给出其流程图的设计。如图6-18所示。

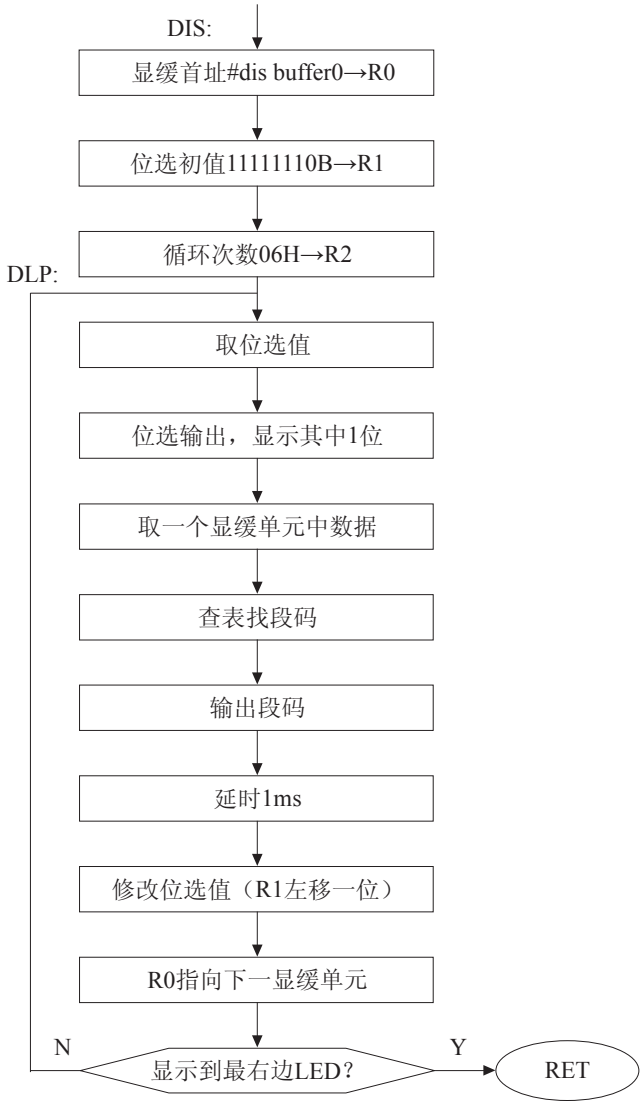


图 6-18　动态显示子程序流程图

动态显示子程序清单如下：

```
DIS：   MOV   R0，#dis_buffer0      ；设显缓首址
        MOV   R1，#11111110B        ；设位选初值，先点亮最高位（最左边一位）
        MOV   R2，#06H              ；设循环次数（6位点亮一遍）
DLP：   MOV   A，R1                 ；取位选值
        MOV   P1，A                 ；位选输出，显示其中1位
        MOV   A，@R0                ；取一个显缓单元中数据
        MOV   DPTR，#DTAB
        MOVC  A，@A+DPTR            ；查表找段码
        MOV   P2，A                 ；输出段码
        ACALL DLY1MS                ；调用延时1ms子程序
        MOV   A，R1                 ；修改位选值（R1左移一位）
```

```
        RL      A
        MOV     R1，A
        INC     R0                  ；R0指向下一显缓单元
        DJNZ    R2，DLP             ；没显示到最右边LED，转DLP继续
        RET                         ；显示完，返回
                                    ；段码表
DTAB:   DB  0C0H，0F9H，0A4H，0B0H，99H，92H，82H，0F8H
        DB  80H，90H，88H，83H，0C6H，0A1H，86H，84H
                                    ；延时1ms子程序
DLY1MS：MOV     R7，#02H
DL：    MOV     R6，#0FFH
DL1：   DJNZ    R6，DL1
        DJNZ    R7，DL
        RET
```

注意：由于是动态显示，该显示子程序DIS必须每隔一定的时间被调用一次，并且要控制好显示扫描的刷新频率，这样才能确保LED显示器的显示不闪烁。

6.3 LCD液晶显示器接口

学习目标

1. 了解LCD显示器的特点和分类。
2. 掌握字符型液晶显示模块的接口电路设计。
3. 掌握字符型液晶显示模块的程序设计要点。

导　入

液晶显示器（LCD）是一种功耗极低的显示器件，它广泛应用于便携式电子产品中。它不仅省电，而且能够显示大量的信息，如文字、曲线、图形等，其显示界面较之数码管有了质的提高。近年来，液晶显示技术发展很快，LCD显示器已经成为仅次于显像管的第二大显示产业。液晶显示器在单片机应用系统中得到越来越普遍的使用。

6.3.1 LCD液晶显示器简介

LCD（Liquid Crystal Display）液晶显示器由于类型、用途不同，其性能、结构也不相同，但其基本形态和结构却大同小异。

1. LCD显示器的结构

液晶显示器的结构如图6-19所示。不同类型的液晶显示器件，其组成可能会有不同，但是所有液晶显示器件都可以认为是由两片光刻有透明导电电极的基板，夹持一个液晶层，封接成一个偏平盒（有时在外表面还可能贴装偏振片等）而构成的。

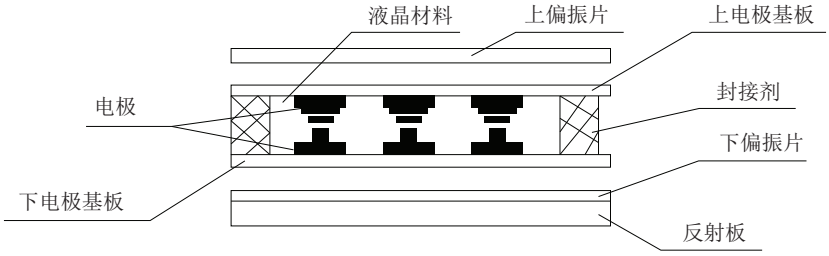


图 6-19 液晶显示器结构

现将构成液晶显示器件的三大基本部件介绍如下：

① 玻璃基板：这是一种表面极其平整的浮法生产的薄玻璃片。其表面蒸镀有一层In_2O_3或S_nO_2透明导电层，即ITO膜层，经光刻加工制成透明导电图形。这些图形由像素图形和外引线图形组成，因此，外引线不能进行传统的锡焊，只能通过导电橡胶条或导电胶带等进行连接。如果划伤、割断或腐蚀，则会造成器件报废。

② 液晶：液晶材料是液晶显示器件的主体。不同器件所用液晶材料不同，液晶材料大都是由几种乃至十几种单体液晶材料混合而成。每种液晶材料都有自己固定的清亮点TL和结晶点TS，因此也要求每种液晶显示器件必须使用和保存在TS～TL的温度范围内。如果使用或保存温度过低，结晶会破坏液晶显示器件的定向层；如果温度过高，液晶会失去液晶态，也就失去了液晶显示器件的功能。

③ 偏振片：偏振片又称偏光片，由塑料膜材料制成，涂有一层光学压敏胶，可以贴在液晶盒的表面。前偏振片表面还有一保护膜，使用时应揭去，偏振片怕高温、高湿，在高温高湿条件下会使其退偏振或起泡。

2. LCD显示器的特点

液晶显示器有以下显著特点：

① 低电压微功耗：工作电压只有3~5V，工作电流只有几毫安。因此它普遍用作便携式和手持仪器仪表的显示屏幕。

② 平板型结构：LCD显示器内由两片平行玻璃组成的夹层盒，面积可大可小，且适合于大批量生产，安装时占用体积小，减小了设备体积。

③ 被动显示：液晶本身不发光，而是靠调制外界光进行显示。因此适合人的视觉习惯，不会使人眼睛疲劳。

④ 显示信息量大：LCD显示器，其像素可以做得很小，相同面积上可容纳更多信息。

⑤ 易于彩色化。

⑥ 没有电磁辐射：在其显示期间不会产生电磁辐射，对环境无污染，有利于人体健康。

⑦ 寿命长：LCD器件本身无老化问题，寿命极长。

3. LCD显示器的分类

通常可将LCD分为笔段型、字符型和点阵图形型。

① 笔段型：笔段型是以长条状显示像素组成一位显示。该类型主要用于数字显示，也可用于显示西文字母或某些字符。这种段型显示通常有六段、七段、八段、九段、十四段和十六段等，在形状上总是围绕数字“8”的结构变化，其中以七段显示最常用。它广泛用于电子表、数字仪表、笔记本计算机中。

② 字符型：字符型液晶显示模块是专门用来显示字母、数字、符号等的点阵型液晶显示模块。在电极图形设计上它是由若干个5×8或5×11点阵组成，每一个点阵显示一个字符。这类模块广泛应用于寻呼机、大哥大电话、电子笔记本等电子设备中。

③ 点阵图形型：点阵图形型是在一平板上排列多行和多列，形成矩阵形式的晶格点，点的大小可根据显示的清晰度来设计。这类液晶显示器可广泛用于图形显示，如游戏机、笔记本电脑和彩色电视等设备中。

LCD还有一些其它的分类方法。

- 按采光方式可分为自然采光和背光源采光LCD。
- 按LCD的显示驱动方式可分为静态驱动、动态驱动和双频驱动LCD。
- 按控制器的安装方式可分为含有控制器和不含控制器LCD两类。含有控制器的LCD又称为内置式LCD。内置式LCD把控制器和驱动器用厚膜电路做在液晶显示模块印制底板上，只需通过控制器接口外接数字信号或模拟信号即可驱动LCD显示。因内置式LCD使用方便简洁，因此，在字符型LCD和点阵图形型LCD中得到了广泛应用。不含控制器的LCD还需另外选配相应的控制器和驱动器才能工作。

6.3.2 字符型液晶显示模块接口电路

1. 字符型液晶显示模块概述

字符型液晶显示模块是一类专用于显示字母、数字和符号等的点阵型液晶显示模块。字符型液晶显示模块是由若干个5×8或5×11点阵块组成的字符块集。每一个字符块是一个字符位，每一位都可以显示一个字符，字符位之间空有一个点距的间隔，起着字符间距和行距的作用。这类模块使用的是专用于字符显示控制与驱动的IC芯片，因此，这类模块的应用范围仅局限于字符而显示不了图形，所以称其为字符型液晶显示模块。

字符型液晶显示驱动控制器广泛应用于字符型液晶显示模块上。目前最常用的字符型液晶显示驱动控制器是HD44780U，最常用的液晶显示驱动器为HD44100及其替代品。

字符型液晶显示模块在世界上是比较通用的，而且接口格式也是比较统一的，其主要原因是各制造商所采用的模块控制器都是HD44780U及其兼容品。不管它的显示屏尺寸如何，它的操作指令及其形成的模块接口信号定义都是兼容的。

HD44780U由控制部、驱动部和接口部三部分组成。

控制部是HD44780U的核心，它产生HD44780U内部的工作时钟，控制着各功能电路的工作。控制部控制全部功能逻辑电路的工作状态，管理字符发生器CGROM和CGRAM，显示存储器DDRAM。HD44780U的控制部由时序发生器电路、地址指针计数器AC、光标闪烁控制电路、字符发生器、显示存储器和复位电路组成。

HD44780U的驱动部具有液晶显示驱动能力和扩展驱动能力，由并/串数据转换电路、16路行驱动器和16位移位寄存器、40路列驱动器、40位锁存器、40位移位寄存器、液晶显示驱动信号输出和液晶显示驱动偏压等组成。

HD44780U的接口部是HD44780U与计算机的接口，由I/O缓冲器，指令寄存器和译码器，数据寄存器，“忙”标志BF触发器等组成。

HD44780U的指令系统共有8条指令，限于篇幅，这里不再列出。

2. 字符型液晶显示模块接口电路

HD44780U可与单片机接口，由单片机输出直接控制HD44780U及其时序。HD44780U与液晶显示器连接框图如图6-20所示。

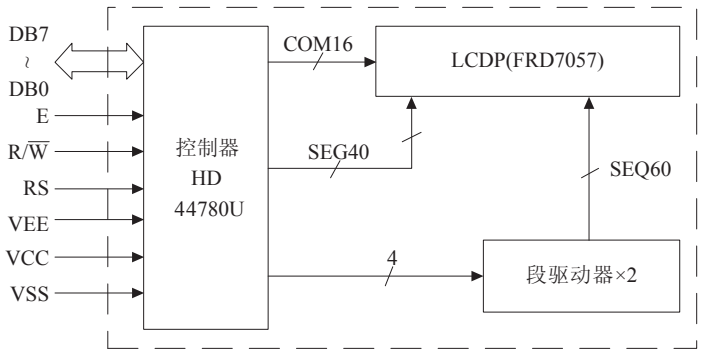


图 6-20 HD44780U与液晶连接框图

单片机与字符型LCD显示模块的连接方法可分为直接访问和间接访问两种，数据传输的形式可分为8位和4位两种。

① 直接访问方式

直接访问方式是把字符型液晶显示模块作为存储器或I/O接口设备直接连到单片机总线上，采用8位数据传输形式时，数据端DB0~DB7直接与单片机的数据总线相连，寄存器选择端RS信号和读/写选择端R/W信号利用单片机的地址线控制。使能端E信号则由单片机的地址线和读/写线信号共同控制，以实现HD44780U所需的接口时序。如图6-21给出了以存储器访问方式对液晶显示驱动的控制电路。

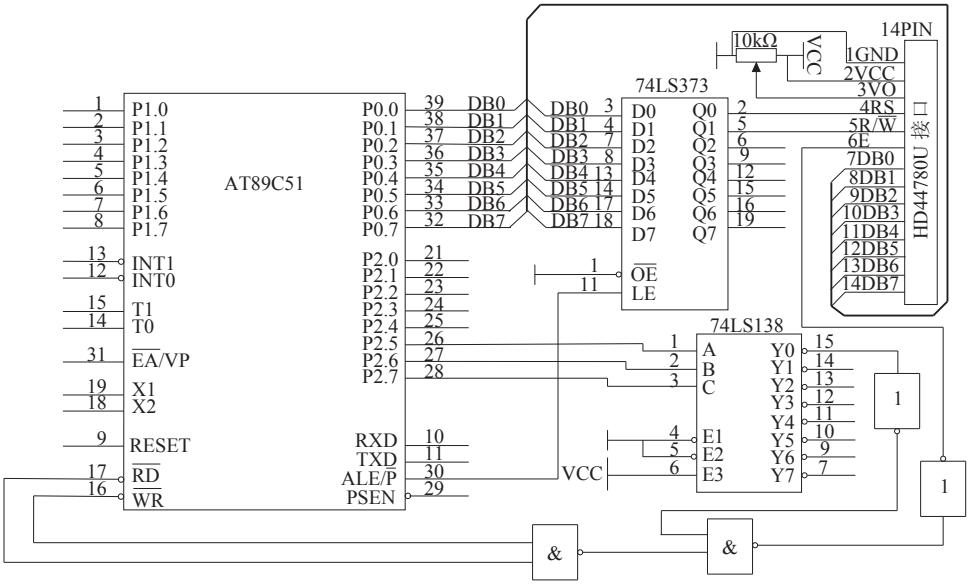


图 6-21 直接访问方式下89C51与字符型液晶显示模块的接口

在图6-21中，HD44780U的8位数据总线与89C51的数据总线直接相连，P0口产生的地址信号被锁存在74LS373内，其输出Q0、Q1给出了RS和R/$\overline{W}$的控制信号。E信号由$\overline{RD}$和$\overline{WR}$信号逻辑与非后，然后与高位地址线组成的“片选”信号实施选通控制。高3位地址线经译码输出打开了E信号的控制门，接着在$\overline{RD}$或$\overline{WR}$信号控制下，经P0口进行数据传输，实现对字符型LCD显示模块的访问。

在写操作过程中，HD 44780U要求E信号结束后，数据线上的数据要保持10μs以上的时间，而89C51单片机的P0接口在信号失效后将有58μs（以12MHz晶振计算）的数据保持时间，足以满足该项控制时间的要求。

在读操作过程中，HD 44780U在E信号为高电平时就将所需数据送到数据线上，E信号结束后，数据可保持20μs，这满足了89C51对该时序的要求。

单片机对字符型LCD显示模块的操作是通过软件实现的。编程时要求单片机每一次访问都要先对忙标志BF进行识别，当BF为0时，即HD 44780U允许单片机访问时，再进行下一步操作。

在图6-21的电路下，字符型液晶显示模块的各驱动子程序如下：

```
COM         EQU   20H          ；指令寄存器
DAT         EQU   21H          ；数据寄存器
CW_Add      EQU   0F000H       ；指令口写地址
CR_Add      EQU   0F002H       ；指令口读地址
DW_Add      EQU   0F001H       ；数据口写地址
DR_Add      EQU   0F003H       ；数据口读地址
```

读BF和AC值子程序：

```
PRO：PUSH  DPH
     PUSH  DPL
     PUSH  ACC
     MOV   DPTR，#CR_Add      ；设置指令口读地址
     MOVX  A，@DPTR           ；读BF和AC值
     MOV   COM，A             ；存入COM单元
     POP   ACC
     POP   DPL
     POP   DPH
     RET
```

写指令代码子程序：

```
PR1：PUSH  DPH
     PUSH  DPL
     PUSH  ACC
     MOV   DPTR，#CR_Add      ；设置指令口读地址
PR11：MOVX A，@DPTR           ；读BF和AC值
     JB    ACC.7，PRll        ；判BF=0否，若是则继续
```

```
        MOV   A，COM              ；取指令代码
        MOV   DPTR，#CW_Add       ；设置指令口写地址
        MOVX @DPTR，A             ；写指令代码
        POP    ACC
        POP    DPL
        POP    DPH
        RET
```

写显示数据子程序：

```
    PR2：PUSH   DPH
        PUSH   DPL
        PUSH   ACC
        MOV   DPTR，#CR_Add       ；设置指令口读地址
   PR21：MOVX A，@DPTR            ；读BF和AC值
        JB      ACC.7，PR21       ；判BF=0否，若是则继续
        MOV   A，DAT              ；取数据
        MOV DPTR，#DW_Add         ；设置数据口写地址
        MOVX  @DPTR，A            ；写数据
        POP    ACC
        POP    DPL
        POP    DPH
        RET
```

读显示数据子程序：

```
PR3：   PUSH  DPH
        PUSH  DPL
        PUSH  ACC
        MOV   DPTR，#CR_Add       ；设置指令口读地址
PR31：  MOVX A，@DPTR             ；读BF和AC值
        JB      ACC.7，PR31       ；判BF=0否，若是则继续
        MOV   DPTR，#DR_Add       ；设置数据口读地址
        MOVX A，@DPTR             ；读数据
        MOV   DAT，A              ；存入 DAT单元
        POP    ACC
        POP    DPL
        POP    DPH
        RET
```

初始化子程序：

```
INT：   MOV   A，#30H             ；工作方式设置指令代码
```

```
        MOV   DPTR，#CW_Add      ；指令口地址设置
        MOV   R2，#03H           ；循环量=3
INT1：  MOVX @DPTR，A            ；写指令代码
        LCALL DELAY              ；调延时子程序
        DJNZ  R2，INT1
        MOV   A，#38H            ；设置工作方式（8位总线）
        MOV  A，#28H             ；设置工作方式（4位总线）
        MOVX @DPTR，A
        MOV  COM，#28H           ；以4位总线形式设置
        LCALL        PR1
        MOV  COM，#01H           ；清屏
        LCALL PR1
        MOV  COM，#06H           ；设置输入方式
        LCAL LPR1
        MOV  COM，#0FH           ；设置显示方式
        LCALL        PR1
        RET
DELAY： ⋮                        ；延时子程序
        RET
```

以上给出了8位数据总线形式的接口电路及驱动程序。4位数据总线形式应用于4位计算机的接口。在89C51上应用4位数据线是将数据总线高4位认为是字符型液晶显示模块的数据总线，数据总线的低4位无用。这样，不用改变图6-21的电路就可以仿真出4位计算机对字符型液晶显示模块的接口。

② 间接访问方式

间接访问方式是计算机把字符型液晶显示模块作为终端与计算机的并行接口连接，计算机通过对该并行接口的操作间接实现对字符型液晶显示模块的控制。

图6-22以89C51的P1和P3接口作为并行接口与字符型液晶显示模块连接的实用接口电路。图中，电位器为VO口提供可调的驱动电压，用以实现显示对比度的调节。在写操作时，使能信号E的下降沿有效。在软件设置顺序上，先设置RS和R/$\overline{W}$状态，

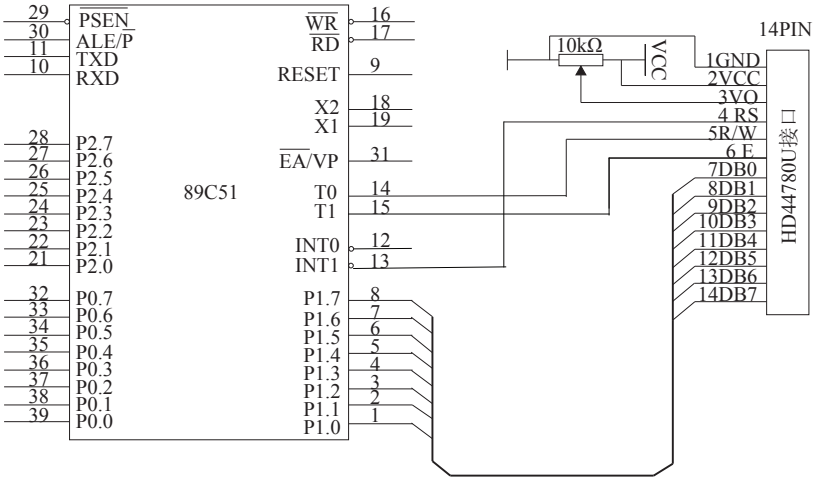


图 6-22 间接访问方式下89C51与字符型液晶显示模块的接口

再设置数据，然后产生E信号的脉冲，最后复位RS和R/$\overline{W}$状态。在读操作时，使能信号E的高电平有效，所以在软件设置顺序上，先设置RS和R/$\overline{W}$状态，再设置E信号为高，这时从数据口读取数据，然后将E信号置低，最后复位RS和R/$\overline{W}$状态。间接访问方式通过软件执行产生操作时序，在时间上足够满足要求，因此，间接访问方式能够实现高速计算机与字符型液晶显示模块的连接。

6.3.3 图形液晶显示接口

图形液晶显示器可显示汉字与复杂图形，广泛应用于游戏机、笔记本电脑和彩色电视等设备中。图形液晶显示一般都需与专用液晶显示控制器配套使用，属于内置式LCD。常用的图形液晶显示控制器有SED1520、HD61202、T6963C、HD61830A/B、SED1330/1335/1336/E1330、MSM6255、CL-GD6245等。各类液晶显示控制器的结构各异，指令系统也不同，但其控制过程基本相同。

知识拓展

完成一个基于89C51单片机的LED数码管时钟电路设计，并编制相应的程序。具体要求是：时、分、秒信号利用单片机的定时器产生，并用6位数码管显示，采用24h计时方式。如果有潜能的话，在此基础上再增加一个按键校时功能，按键数量和功能定义自行拟定。

本章小结

键盘分两种：一种是编码式键盘，另一种是非编码式键盘。单片机应用系统通常使用非编码式键盘。任何机械结构的按键在操作时，都不可避免的存在机械抖动，抖动时间一般在5~10ms。为了避免CPU对按键操作的误判，必须要去抖动，去抖动有两种方法：硬件去抖动和软件去抖动。实际应用中，大多采用软件去抖动。

键盘结构分两种：独立式键盘和行列式键盘。独立式键盘的特点是电路简单、按键配置灵活、编程容易，按键多时I/O线占用多；行列式键盘的特点是判键原理复杂，因此编程也较为复杂，按键多时I/O线占用少。

一个完善的键盘控制程序应具备以下功能：

① 检测有无按键按下，并采取硬件或软件措施，消除键盘按键机械触点抖动的影响。

② 有可靠的逻辑处理办法。每次只处理一个按键，其间任何按键的操作对系统不产生影响，且无论一次按键时间有多长，系统仅执行一次按键功能程序。

③ 准确输出键码（或键号），以满足跳转指令要求。

LED数码显示器是一种由LED发光二极管组合显示字符的显示器件。它使用了8个LED发光二极管，内部有两种连接方法。一种是共阳极接法，另一种是共阴极接法。LED数码显示器的显示方式分两种：一种是静态显示，另一种是动态显示。静态显示的特点是各位数码管相互独立，公共端恒定接地（共阴极）或接正电源（共阳极）。每个数码管的8个段选线（a~dp）分别与一个8位I/O口地址相连；动态显示的特点是各位数码管相应的段选线并联在一起，由一个8位的I/O口控制，各位的位选线（公共阴极或阳极）由另外的I/O口线控制，各位数码管分时轮流点亮。

LED数码显示器的显示程序设计的要点是：

① 在ROM中建立字形码表。

② 设置显缓单元，存放待显字符的字形码位置号。

③ 查表找出对应字符的字形码。

④ 输出字形码到显示端口。

LCD显示器的特点是低压微功耗、显示信息量大、没有电磁辐射、寿命长。LCD显示器分笔段型、字符型和点阵图形型。

思考与练习题

一、单项选择题

1. 按键的机械抖动时间参数通常是（　）。

A. 0　　B. 5~10μs　　C. 5~10ms　　D. 1s以上

2. N位LED显示器采用动态显示方式时，需要提供的I/O线总数是（　）。

A. 8+N　　B. 8×N　　C. N　　D. 8

3. 在键盘的工作方式中，有键操作CPU会立刻执行键盘扫描程序的是（　）。

A. 编程扫描方式　B. 定时扫描方式　　C. 中断扫描方式　　D. 不可能

4. LED显示器的动态显示比起静态显示，以下哪种说法是正确的（　）。

A. 更耗电　　B. 编程简单

C. 刷新一次就不会熄灭　　D. 要不断地扫描

5. 字符型液晶显示模块不能用来显示（　）。

A. 数字　　B. 英文字母　　C. 汉字　　D. 标点符号

二、判断题

1. 单片机键盘通常采用非编码键盘。（　）
2. 在按键数量较多的情况下，独立式键盘占用的I/O线比行列式键盘要多。（　）
3. 点亮共阴数码管时，其COM脚必须提供高电平。（　）
4. 数码管采用动态显示方式比起静态显示方式更省I/O线。（　）
5. 数码管的显示内容比LCD显示器要丰富得多。（　）

三、问答题

1. 为什么要消除按键的机械抖动？有哪些方法？
2. 在行列式键盘中，每个键的位置码是如何确定的？位置码有何意义？
3. 什么叫LED数码管的段码？段码有什么用处？又是如何确定的？
4. LED数码显示器的显示程序设计要点是什么？
5. 字符型液晶显示模块接口电路两种设计方法的各自特点是什么？

实验项目6　键盘和显示的基本应用

一、实验目的

1. 了解和熟悉数码管接口的多种设计方法和技巧。

2. 通过本实验，进一步掌握键盘和显示程序的设计要点和设计方法。

3. 在实践中，切实提高单片机应用的操作技能和动手能力，独立思考问题、分析问题和解决问题的能力。

二、实验设备

1. PC计算机一台

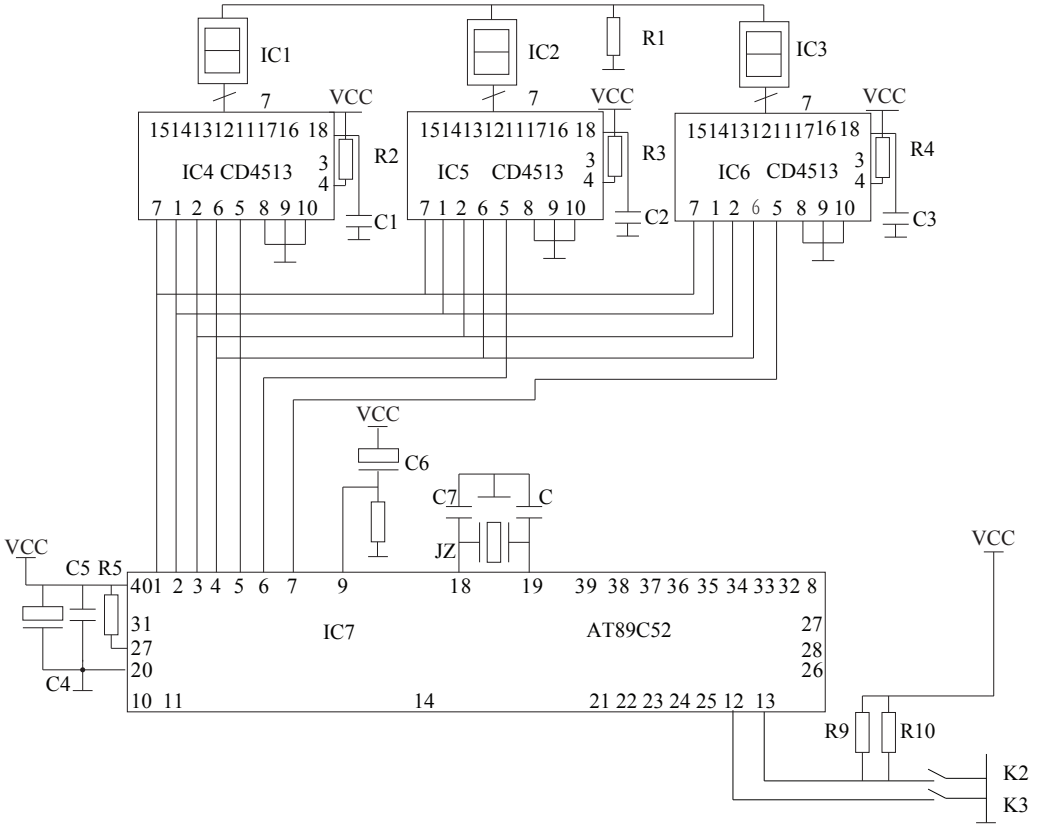
2. 单片机实验装置一套（教材配套）

3. Keil μVision2集成开发环境

4. STC_ISP_V480在系统编程软件

三、实验内容

1. 实验线路



提示：CD4513—BCD锁存7段译码器驱动器

可逆计数器的设计。具体要求是：在单片机内部设置一个软件计数器（初值=100），计数范围0~255；计数信号来自两个按键K2和K3，其中K2为增1键（按一次软件计数器加1），K3为减1键（按一次软件计数器减1）；软件计数器的计数值以十进制形式在3位数码管上显示。

2. 实验程序

```
        BCD10 EQU   20H
        BCD2  EQU   21H
        COUNT EQU   22H
        A0    BIT   00H
        A1    BIT   01H
        A2    BIT   02H
        A3    BIT   03H
        B0    BIT   04H
        B1    BIT   05H
        B2    BIT   06H
        B3    BIT   07H
        C0    BIT   08H
        C1    BIT   09H
        C2    BIT   0AH
        C3    BIT   0BH
        QA    BIT   P1.0
        QB    BIT   P1.1
        QC    BIT   P1.2
        QD    BIT   P1.3
        EL1   BIT   P1.4
        EL2   BIT   P1.5
        EL3   BIT   P1.6
        K2    BIT   P3.3
        K3    BIT   P3.2

        ORG   0000H
        LJMP  MAIN
        ORG   0030H
MAIN:   MOV   SP，#30H
        MOV   P1，#0FFH
        MOV   COUNT，#100
        LCALL BCDCONV
        LCALL DISPLAY
K2SCAN: JB    K2，K3SCAN
        LCALL DELAY10MS
```

```
        JNB     K2，$
        INC     COUNT
        LCALL BCDCONV
        LCALL DISPLAY
K3SCAN：JB      K3，K2SCAN
        LCALL DELAY10MS
        JNB     K3，$
        DEC     COUNT
        LCALL BCDCONV
        LCALL DISPLAY
        LJMP    K2SCAN

BCDCONV：MOV    A，COUNT
        MOV     B，#64H
        DIV     AB
        MOV     BCD2，A
        MOV     A，#0AH
        XCH     A，B
        DIV     AB
        SWAP    A
        ADD     A，B
        MOV     BCD10，A
        RET

DISPLAY：SETB   EL1
        SETB    EL2
        SETB    EL3
        MOV     C，C0
        MOV     QA，C
        MOV     C，C1
        MOV     QB,     C
        MOV     C，C2
        MOV     QC，C
        MOV     C，C3
        MOV     QD，C
        CLR     EL1
        NOP
        SETB    EL1
        MOV     C，B0
        MOV     QA，C
```

```
        MOV   C，B1
        MOV   QB,     C
        MOV   C，B2
        MOV   QC，C
        MOV   C，B3
        MOV   QD，C
        CLR   EL2
        NOP
        SETB  EL2
        MOV   C，A0
        MOV   QA，C
        MOV   C，A1
        MOV   QB,     C
        MOV   C，A2
        MOV   QC，C
        MOV   C，A3
        MOV   QD，C
        CLR   EL3
        NOP
        SETB  EL3
        RET

DELAY10MS：MOV R6，#20
        I1：    MOV R7，#255
        I2：    DJNZ R7，I2
        DJNZ R6，I1
        RET
        END
```

四、实验操作

1. 熟悉实验线路的基本工作原理，尤其是CD4513芯片的功能。
2. 在理解源程序的基础上，完成实验程序的运行。
3. 打开Keil μVision2 集成开发环境，对源程序进行编辑，然后汇编直至通过，最终生存HEX文件。
4. 完成PC机和单片机实验装置的连接，打开STC_ISP_V480在系统编程软件，将HEX文件下载到单片机中。
5. 运行单片机程序。
6. 操作K2和K3键，观察数码管的计数值。
7. 修改程序，将K2和K3键分别定义为增10键和减10键，并完成调试和运行。

第7章 单片机C51程序设计基础

C语言是源于编写UNIX操作系统的语言，属于一种高级语言，至今应用十分广泛。由于单片机应用对象的日趋复杂，汇编语言在编写程序时的不足之处也不断显现，如开发周期比较长，可读性比较差、调试比较麻烦等等。而C语言在解决了生成代码长，运行速度比较慢等缺陷，尤其是在德国Keil公司开发了Keil C51用做单片机C语言的编译器以后，C语言在兼容ANSI C的同时与8051硬件偶合程度十分紧密的编译特性，使得单片机在应用程序开发过程加入了C语言而更为快速和方便。

7.1 C51语言概述

学习目标

1. 了解单片机的C51语言。
2. 认识C51的程序结构。

导　入

单片机技术学到这里你一定会感觉其应用的广度了吧，同时也一定熟悉“事半功倍”的成语吧？单片机的程序除了用汇编语言写以外还可以用C51语言来编写。

7.1.1 单片机的C语言——C51语言

1. C51语言在单片机上应用的优势

通过前面章节学习，已经具备了一定的单片机硬件基础与汇编语言的编程方法，对了解在单片机中使用C51语言编写应用程序与用汇编语言编写程序的不同点是有所裨益的。首先，C51语言是面向对象的高级语言，而汇编语言是面向机器的低级语言，所以用C51语言编写程序更为便捷；其次，C51语言的特点是方便程序的模块化开发、方便阅读、开发周期比较短；再者，用C51语言编写的程序在不同系列的硬件系统上的移植性比较好，调试比较灵活，对程序维护也比较容易；用C51语言编写的程序还能把汇编语言嵌套在其中，在对时间要求相对严格的系统上运行效果也比较理想。

2. Keil C51

不同的单片机系统（或嵌入式系统）的编译器都有各自的侧重点。Keil C51是一款开发单片机C51语言的编译器，即在第二章中用的Keil μVision2集成开发环境。Keil C51是一款基于Windows界面的集成开发环境，由于内部既有汇编语言的汇编器用于对汇编指令写的源程序进行汇编，又有C51编译器完成对用C51语言编写程序进行编译，能对C51语言的结构化源程序经过编译以后的浮动目标地址的目标代码经过链接/定位，产生具有绝对地址的目标代码的功能；在Keil C51中还有功能十分强大的C51语言库函数。Keil C51生成的目标代码效率高，大部分语句生成的汇编代码紧凑、方便理解，库函数的提供在很大程度上提高了编程效率。

3. C51与ANSI C的不同点

C51具有标准ANSI C语言的绝大部分特性，而且基本语法也相同。但由于针对不同的硬件系统，与标准ANSI C也有不同之处。

譬如库函数的不同，C51中既有原有的标准库函数，也有经过重新编写的对应硬件系统的函数；数据类型添加了与8051位操作匹配的位数据类型；C51的头文件反映不同硬件系统的不同功能。在C51中能够方便的指定变量的存储类型，提高程序的执行效率。

7.1.2 C51程序的基本构架

【例7-1】 说明C51程序结构的基本例子。

```
#include "reg51.h"
sbit LED0=P0^0;
......
bit FLAG2;
main()
{
  unsigned int i;
    FLAG1=0;
......
      while(1)
      {
```

```
        if( (KEY1==1)&&(FLAG1==0)) FLAG1=1;
              else  if((KEY1==0)&&(FLAG1==1))
        {
              for (i=0;i<5000;i++);
                 if(KEY1==0)
              {
                   LED0=1;
                   ......
                   LED7=0;
                   FLAG1=0;
              }
        }
        else        if((KEY2==1)&&(FLAG2==0))FLAG2=1;
              else  if((KEY2==0)&&(FLAG2==1))
        {
                   for (i=0;i<5000;i++);
                   if(KEY2==0)
        {
              LED0=0;
              ......
              LED7=1;
              FLAG2=0;
             }
         }
     }
}
```

程序结构大体可以分为这样几部分：用预处理命令#include读入头文件reg51.h，主要是对于8051单片机的特殊功能寄存器说明；其次是一个主函数main()函数，当然在main()函数里面可以调用其它若干函数，本例没有，在主函数中，if语句用法与ANSI C语言一致，而bit FLAG2语句与ANSI C语言有所不同。当然依据需要还可以编写其他函数。

7.2 C51标识符与关键字

学习目标

1. 掌握C51语言对象命名的法则。
2. 熟悉C51语言的关键字。

导　入

任何一个人、任何一个事物、植物都需要给出一个名称来进行标识。有的用来标识特征，有的揭示内涵，有的含特殊定义，总之所给的名称能正确识别某一类中的某一个，C51中也有标识符与关键字，并对标识符应用与关键字做了规定。

7.2.1 标识符

标识符是用来表示源程序中的一个对象的名称，对象可以是变量、函数标号、数组名及语句等。标识符可以由字母、数字（0~9）和下划线“_”组成，而第一个字符必须是小写字母(a~z)、大写字母(A~Z)或下划线，标识符区分大小写。

例如：“timerl”、“A_2”等，是正确的。而“1num”则是错误的标识符。

“ch1”和“CH1”代表两个不同的标识符。

一般不要以下画线开头来命名标识符。C51的标识符长度可达32个字符，在命名C51标识符时，只要是清楚表达含义，有助于阅读和理解源程序就行，不必太长。C51的标识符定义不能使用关键字，也不能和已使用的函数名或C51库函数同名。

例如：“char”是不正确的标识符，因为“char”是关键字。

7.2.2 关键字

关键字是被C51编译器定义的特殊标识符，如for、if、case等，这类关键字有固定的名称并具有一定含义。C语言中一共有32个关键字，全部用小写字母表示。根据应用不同，通常把关键字分为四类，具体参见附录2。

针对8051单片机的特殊性，在Keil C51中增加了一些关键字，新增关键字见表7-1，其中不包括与实时操作系统有关的关键字。

表 7-1 Keil C51新增的关键字

关键字	意义与用法
code	指定程序存储器
bdata	指定RAM中的位寻址区（20H~2FH）
data	定义变量为RAM前128字节（00H~7FH）
idata	定义变量为RAM全部256字节（00H~FFH）
sfr	定义特殊功能寄存器
sbit	定义位寻址
Sfr16	定义16位的特殊功能寄存器
xdata	指定外部RAM存储器，多达64KB
pdata	指定一页（256Bytes）外部RAM存储器
bit	定义位变量
small	指定存储器的使用模式为小模式，缺省则变量为内部RAM
compact	指定存储器的使用模式为紧凑模式，缺省则变量为外部RAM，最多256B（一页）
large	指定存储器的使用模式为大模式，缺省则变量为外部64KBRAM
using	函数中指定使用某一组工作寄存器
interrupt	指定中断程序

注：在表7-1主要关键字中，前9个与存储区域有关；bit与数据类型相关，11~13与存储模式有关；最后两个与中断有关。

7.3 数据类型、运算符与表达式

学习目标

1. 掌握C51语言的数据类型。
2. 掌握C51语言中用各种运算符组成的表达式。

导　入

数据是单片机操作的对象，数据类型揭示了一个数值、字符的不同格式，根据不同数据类型的组合、构架形成数据的结构，用各种运算符号连接数据就形成了各类表达式。

7.3.1 C51数据类型

在存储器中存放的数据有长短、有符号和无符号之区分，在C51中称之为数据类

型。一个变量在使用前必须定义其数据类型。C51除继承ANSI C语言中基本的数据类型char、int、short、long、void 、float和double等外，还提供了几种组合类型，组合包括数组、指针、结构、联合（共用体）、枚举和位域。C51能处理的数据类型如表7-2所示。

表 7-2 Keil C51的数据类型

数据类型	长度	取值范围
signed char	单字节	有符号字符变量，取值：－128~＋127
unsigned char	单字节	无符号字符变量，取值：0~＋255
signed int	双字节	有符号整型数，取值：－32768~＋32767
unsigned int	双字节	无符号整型数，取值：0~＋65535
signed long int	四字节	有符号长整型数，取值：－2147483648~＋2147483647
unsigned long int	四字节	无符号长整型数，范围：0~4294967295
float	四字节	浮点数，范围：-3.4×10^{-38}~$+3.4\times10^{38}$
bit	位	取值：0或1
sfr	单字节	取值：0~255
sfr16	双字节	取值：0~65535
sbit	位	取值：0或1

1. char字符型

char类型的长度是一个字节，通常定义字符数据的常量或变量。分有符号字符型signed char和无符号字符型unsigned char，默认值为signed char型。unsigned char型用字节中所有的位来表示数值，数值范围是0~255。

unsigned char常用于处理ASCII字符、小于或等于255的整型数。signed char型字节中最高位表示数据的正负符号，“0”正数，“1”负数，负数用补码表示，数值范围是－128~＋127。

2. int整型

int类型的数据长度为两字节。同样分有符号整型数signed int与无符号整型数unsigned int，默认值为signed int型。signed int表示的数值范围见表7-2，字节中最高位表示数据的符号，负数用补码表示。unsigned int表示的数值范围为0~+65535。

3. long int长整型

long int类型数据长度为四字节。分有符号长整型signed long int和无符号长整型unsigned long int，默认值为signed long int型。signed long int表示的数值范围见表7-2，字节中最高位表示数据的符号，负数用补码表示。unsigned long表示的数值范围见表7-2。

4. float浮点型

float浮点型是符合IEEE-754标准的单精度浮点型数据，长度为四字节，在十进制中具有7位有效数字。

5. bit位变量

bit位类型是C51编译器的一种扩展数据类型，用于定义一个位变量，但不能定义位指针和位数组，bit取值是一个位二进制数。

6. sfr特殊功能寄存器

sfr作为一种扩展数据类型，占一个内存单元，值域为0~255。使用sfr可以访问

MCS-51单片机内部的所有特殊功能寄存器。

格式：sfr 特殊功能寄存器名 = 绝对地址。

例如：sfr P1 = 0x90；

定义P1端口在片内寄存器绝对地址为0x90，需要注意的是绝对地址是单片机硬件所规定的。

按照格式“=”后的地址必须是常数，不能是有运算符的表达式，而常数值的范围必须在特殊功能寄存器地址范围内，位于0x80~0xFF。

7. sfr16 16位特殊功能寄存器

sfr16作为16位特殊功能寄存器（连续2字节的特殊功能寄存器名）。格式与sfr一致，占用两个字节内存单元，其值域为0~65535。sfr16和sfr一样用于操作特殊功能寄存器，但sfr16类型主要适用8052。

8. sbit 可寻址位

sbit可寻址位也是一种扩展数据类型，sbit可访问片内RAM中的可寻址位或特殊功能寄存器中的可寻址位。用于sbit的三种变量声明形式如下。

- sbit 位名=特殊功能寄存器名^ 整型常量

其中特殊功能寄存器必须在此之前被sfr定义过，且是可以按位寻址的。整型常量必须是0～7的一个数字。

例如： sfr IE=0xA8;//声明IE为特殊功能寄存器，地址为0xA8

sbit EA=IE^7;//指定IE的第7位为EA，即中断允许

- sbit 位名=绝对地址^ 整型常量

例如: sbit CY=0xD0^7;//0xD0是PSW的绝对地址, 第7位是CY。

- sbit 位名=位绝对地址

例如： sbit OV=0xD2;

特殊功能位代表一个独立的声明类，不能和其他位声明或位域互换。

9. *指针型

指针型是一个变量，在这个变量中存放的指向另外一个数据存放的地址。这个指针变量要占据一定的内存单元，在C51中它的长度一般为1~3个字节。

7.3.2 运算符和表达式

运算符就是让编译器进行某种特定运算的符号，丰富的运算符是C语言的特点之一。运算符按其在表达式中所起的作用，可分为算术、关系与逻辑运算符、赋值与位操作运算符、增量与减量运算符等。

运算符按其在表达式中与运算对象的关系，又可分为单目运算符、双目运算符和三目运算符等。单目运算符只有一个运算对象，例如：-a；双目运算符有两个运算对象，例如：a-b；三目运算符用在条件表达式中则有三个运算对象。

表达式是由运算符与运算对象组成的具有一定含义的式子。

1. 赋值运算符与表达式

赋值运算符构成赋值语句，其格式是：变量 = 表达式。

“=”号就是赋值运算符，表示将右边表达式的值赋予左边的变量。利用赋值运算符将

一个变量与一个表达式连起来的式子称为赋值表达式。

注意：“=”与数学运算中的等号不同，也不能与关系运算符“==”相混淆。

赋值语句的意义是先计算出右边表达式的值，然后将该值赋予左边的变量。表达式可以是下面介绍的各种运算或者函数，也可以是一个赋值表达式，即允许进行多重赋值。下面几条语句都是合法的赋值语句：

```
x=6;                //将常数6赋予变量x
a=b=3;              //将常数3同时赋予变量a和b
c=c+5;              //将变量c的值加5后赋予c变量
```

2. 算术运算符组成的表达式

C语言中的算术运算符有以下几种:

- +：加或取正运算符。
- -：减或取负运算符。
- *：乘法运算符。
- /：除法运算符。
- %：取余运算符。

加、减、乘、除法要求有两个运算对象是属于双目运算符。加减和乘法运算与普通的算术运算没有什么不同，而除法运算“/”与取余运算“%”则有所不同，两个整数相除运算时，结果为整数，其小数部分将会舍去，例如：10/3的结果为3；取余运算只能得到两个整数相除的余数，例如：10%3的结果为1。

用算术运算符将运算对象连接起来的式子即为算术表达式。算术表达式的一般格式是：表达式1 算术运算符 表达式2

例如：x+y*(a+b);

(x+2)/(y-a)。

在计算一个表达式的值时，C语言规定了运算符的优先级和结合性，即算术运算中取负值（-）的优先级最高，其次是乘法运算符（*）、除法运算符（/）、取余运算符（%），加法运算符（+）和减法运算符（-）的优先级最低。在需要时，可以在算术表达式中采用圆括号来改变运算符的优先级，例如在计算a+b*(x+y)时，首先计算(x+y)，然后计算出b*(x+y)，最后计算出a+b*(x+y)。如果在一个表达式中各个运算符的优先级别相同，则计算时按从左到右的结合方向计算。C语言中运算符的优先级和结合性详见附录2。

除了基本的加、减、乘、除运算外，C语言还提供了一种与汇编语言类似的增量和减量运算符。

- ++：加1运算符。
- --：减1运算符。

加1或减1运算符用在变量的左边与用在变量的右边其含义是不同的。

例如：++i （或--i）是先执行i+1（或i-1）操作，再用i的值，而i++ （或i--）则是先使用i 的值，再执行i+1（或i-1）操作。

例如：令 m=10,

```
a=m++;    //运算后a为10，m为11，表示m使用后再加1
a=++m;    //运算后a和m都是11，表示m使用前先加1
```

加1或减1操作在变量计数时经常会用到。

3. 关系运算符与表达式

关系运算符通常用来判别某个条件是否满足，C语言中有以下几种关系运算符。

- ＞ ：大于。
- ＜ ：小于。
- ＞= ：大于等于。
- ＜= ：小于等于。
- == ：等于。
- ！=：不等于。

关系表达式一般形式是：<表达式1> <关系运算符> <表达式2>

关系运算符都是双目运算符，关系运算符的优先级高于赋值运算符而低于算术运算符。在6个关系运算符中，<、<=、>、>=的优先级相同，高于==和!=， == 和！=的优先级相同。关系运算符结合性均为左结合。由关系运算符组成的表达式为关系表达式，关系表达式的值为逻辑真或假（“1”为逻辑真；“0”为逻辑假）。

例如： 20>5+4 结果为True，但要注意算数运算符的优先级高于关系运算符。

4. 逻辑运算符与逻辑表达式

由逻辑运算符构成的式子称为逻辑表达式，根据逻辑表达式的值可以进行条件判断。逻辑运算符共有3种。

- ||：逻辑或。
- &&：逻辑与。
- ！：逻辑非。

逻辑表达式的一般形式类似于关系运算表达式，其值也一致。通常在使用时，关系表达式与逻辑表达式以及控制流语句一起使用。

例如：3>5||10>6&&!(10<8) 按优先级与结合性，结果Flase。

5. 位操作运算符与表达式

位操作运算符的作用是对字节中的位进行操作，可以方便进行位测试、移位、清0与置位等操作。位操作是C51语言的一个特点，使C语言能对单片机的硬件直接进行控制操作。

C51语言共有六种位运算符，若a为二进制数 0110 0101，b为二进制数0000 1111，执行这六种位运算后，情况如下：

- «：左移。例如：a«4；a左移4位结果a=0101 0000。
- »：右移。例如：a»4；a右移4位结果a=0000 0110。

注意：数据执行位操作的移位一端移出，另一端移入0。

- &：逻辑与。例如：a=a&b；a和b按位逻辑与后再赋值给a，结果a=0000 0101。
- |：逻辑或。例如：a=a|b；a和b按位逻辑或后再赋值给a，结果a=0110 1111。
- ^：逻辑异或。例如：a=a^b；a和b按位逻辑异或后再赋值给a，结果a=0110 1010。
- ~：逻辑非。例如：a=~b；b逻辑非后再赋值给a，结果a=1111 0000。

C语言还允许使用一些简写来代替各项操作，构成复合运算，其规定是：

<变量> = <变量> <操作> <表达式>;可以缩写成 <变量> <操作> = <表达式>;

例如: a+=b; 表示 a=a+b;

a*=b; 代表 a=a*b;

a%=b; 代表 a=a%b;

a<<=b;　　代表　a=a<<b;
a&=b;　　代表　a=a&b;
a^=b;　　代表　a=a^b;

除了上面这些运算操作外，C语言还提供了诸如逗号运算、条件运算、类型转换等操作运算。可以参考有关书籍。

7.4 C51常量、变量与存储类型

学习目标

1. 熟悉C51语言常量和变量。
2. 正确理解和应用存储种类和存储器类型。

导　入

在数学、物理知识中经常用到常量与变量，在C语言中同样也有常量与变量，问题是对于单片机而言，这些量需要经过Keil C51编译器的编译以及存储器单元存储，在实际应用中该如何把这些事情联系起来呢?

7.4.1 常量与变量

数值在运行过程中不变化的量称为常量。在C51中常量的数据类型有整型、浮点型、字符型、字符串型等。

1. 整型常量

整型常量即整型常数，一般数字写成十进制，如123、0、－123，也可写成十六进制数，但必须以0x开头，如0x123、0xFF。长整形就在数值后面加L，例如：114L。

2. 浮点型常量

浮点型常量可以写成十进制定点表示形式，即由数字和小数点组成，如0.123、－5.8、122.4等，也可以写成指数形式，即：[±]数字[.数字]e[±]数字。

[ ]为可选项，但其余部分必须有，例如：125e2、－123e－2等。

3. 字符型常量

字符型常量是指单引号括起来的字符，如′a′、′C′，′9′等，一个字符在内存中占有一个字节。对于控制字符（非显示字符），在字符前面加一个反斜杠“\”组成转义

字符。用转义字符可以完成一些特殊功能和输出时格式控制。常用转义字符表如表7-3所示。

表 7-3 常用转义字符表

转义字符	含　义	ASCII码(16/10进制)
\o	空字符(NULL)	00/0
\n	换行符(LF)	0xA/10
\r	回车符(CR)	0xD/13
\t	水平制表符(HT)	0x9/9
\b	退格符(BS)	0x8/8
\f	换页符(FF)	0xC/12
\′	单引号	0x27/39
\″	双引号	0x22/34
\\	反斜杠	0x5C/92

4. 字符串型常量

字符串型常量用双引号括起来的字符，如“NO”、“is”等。也可以引号内为空，称空字符串。当需要表示双引号字符串时，可以使用转义符号“\”表示。

在C语言中，字符串常量是作为字符型数组来处理的，所以在存储字符串时，系统自动在字符串尾部加上“\0”转义字符作为该字符串的结束符。字符常量′X′与字符串常量″X″是不同的，前者在存储时用一个字节的单元，后者要多一个结束符字节。

通常常量可用在不需要改变其值的场合，如固定的数据表、字库等。由于数据表、字库都保存在程序存储器中，而程序存储器在运行中是不允许修改的，一旦出现修改则编译时将会出错。

7.4.2 变量

在程序运行过程中其值会发生变化的量称为变量。每个变量都必须有一个标识符作为它的变量名。变量与常量有着对应关系，变量也存在整形变量、字符型变量等。

例如： char a1，b1；

a1=99，b1=100；（类似于a1=‘c’，b1=‘d’）

变量使用前必须先进行定义，指出其数值类型和存储模式，编译系统会为它分配相应的存储单元。C51对变量定义的格式为：

[存储种类] 数据类型 [存储器类型] 变量名表

例如：static unsigned char data j;//在内部数据存储器中定义一个静态无符号字符型变量j

int k;//定义一个整型变量k，它的存储器类型由编译模式确定

变量定义后有了数值类型并分配了存储单元就需要赋值，即先定义后使用。如前面的a1=99，就是一句给a变量赋值的语句，“=”是运算符表示赋值。需要明确的是变量名与变量的值不是一个概念。

变量在使用过程中还分为全局变量和局部变量。全局变量是在函数之外说明，可被任意函数使用，在整个程序执行期间都有效的变量。局部变量用于函数内部说明，只在本函数或功能块以内有效，在该函数功能块以外则不能使用。由于使用范围不同局部变量可以

与全局变量取相同的名字，此时，局部变量的优先级将高于全局变量，即同名的全局变量在局部变量使用的函数内部将被暂时屏蔽。需要提出的是全局变量一般较少使用。

7.4.3 存储种类与存储器类型

变量的存储种类和存储器类型都是可选项。存储种类有自动（auto）、外部（extern）、静态（static）和寄存器（register）四种。如果在定义变量时省略了存储种类这个选项，那么该变量将默认为自动（auto）变量。全局变量可以定义为外部（extern）与静态（static），二者的共同点是在作用的范围之内可以给多个函数调用；不同之处在于当有多个C源程序文件同时被编译时外部（extern）可以被其他源程序引用，而用静态（static）定义的则不能。局部变量可定义为自动（auto）、静态（static）和寄存器（register），其中寄存器主要是指变量存放在CPU的寄存器中，使运算速度提高，但是变量的数量是有限的。

存储器类型的说明见表7-4所示，指定变量在80C51硬件系统中所对应的存储区域且在编译时能够准确地定位。但是需要注意的是在80C51中，RAM为0x00到0x7F的低128字节；高128字节RAM的0x80到0xFF，则在8052芯片或其他扩展的8051后的芯片才有，且与特殊功能寄存器地址重叠。需要提出的是变量的存储种类与存储器类型完全无关。如前面例子的第一句所定义的字符型变量j，其存储种类为静态，即静态变量，它的存储器类型为data，也就是说，这个变量位于8051芯片内部RAM中的低128字节，即地址为0x00~0x7F。后一条语句定义的是一个整型变量，由于没有定义存储种类和存储器类型，于是系统默认存储种类为自动，存储器类型由存储器编译模式确定。

表 7-4 Keil C51编译器能识别的存储器类型

存储器类型	说　明
data	直接访问片内数据存储器（128Byte），访问速度最快
bdata	直接访问片内数据存储器的位寻址（16Byte）区，允许位与字节混合访问
idata	间接访问内部数据存储器（256Byte），允许访问全部片内地址
pdata	分页访问外部数据存储器（256Byte），汇编用MOVX@iR，指令访问 A
xdata	间接访问外部数据存储器（64KByte），汇编用MOVX @DPTR，A指令访问
code	访问程序存储器（64KByte），汇编用MOVC A，@A+DPTR指令访问

7.4.4 编译模式

在定义变量时若缺省存储器类型，Keil C51会按编译模式small、compact和large所规定的默认存储器类型去指定变量的存储区域，如表7-5所示。

表 7-5 C51编译器支持的存储类型

编译模式	默认存储器类型	说　明
small	data	小模式，默认变量在内部数据存储区。变量数小，但访问速度快。
compact	pdata	紧凑模式，默认变量在外部数据存储区的页（256B，高8位地址由P2口确定）。
large	xdata	大模式，默认变量在外部数据存储区（64KB），空间大，速度慢。

small选项的存储类型把变量存放在80C51系统的内部数据存储区，因此访问数据最快。但是该模式的地址空间有限，适合写一些小型的应用程序，否则data区域就很容易溢出。

compact选项把变量定位在80C51系统的外部数据存储器中。外部数据存储段可有最多256字节（一页），对变量的访问通过寄存器间接寻址进行。采用这种编译模式时，变量的高8位地址由P2口确定。

large选项变量被定位在8051系统的外部数据存储器中，外部数据存储器最多可达64KByte，这要求用DPTR数据指针来访问数据，访问方式效率不高。

7.5 程序的控制语句

学习目标

1. 了解C语言编写程序的结构。
2. 掌握条件控制的语句应用。
3. 掌握循环控制的语句应用。
4. 掌握转向控制语句的应用。

导　入

对于城市的建筑来说，有砖混结构的普通住宅、多层建筑的框架结构。用C51语言编写应用程序也存有几种结构，有了结构的不同组合，编写的应用程序就能应对不同的要求，解决不同的问题。

无论是用汇编语言编写的程序还是用C51编写的程序，CPU运行时都是依次执行的，即从第一条到最后一条，这种程序运行的方向称为顺序结构。但有时在总体顺序运行过程中需要给出一定的条件（或者是中间状态的值），有条件的选择不同语句（任务）执行，这是一种条件选择结构。有时也需要重复循环执行一个算法（例如求累加和）到某个条件为止，这就构成了循环结构。一般为了完成一个任务或者是一个算法所编写的应用程序存在总体顺序结构中含盖条件结构和循环结构的语句。C51语言也给出了构成条件选择结构与循环结构的相关语句。

7.5.1 条件控制语句

C51语言提供构架条件选择结构的条件控制语句有if、switch、case等语句。

1. if条件语句

if语句的几种形式如下：

➢ if（条件表达式）语句

含义是：若条件表达式值为真（非0值），则执行语句，否则不执行语句。

例如：计数a=0的次数：

```
if (a==0) b++;
```

➢ if（条件表达式）语句1

```
else 语句2;
```

含义是：若条件表达式值为真，则执行语句1；反之，若条件表达式的结果为假（0值），则执行语句2。语句1和语句2均可以是复合语句，即可用一个大括号{ }将若干条语句组合在一起而形成一个语句块。

【例7-2】 对a取绝对值并赋值给b。

```
if (a>=0) b= a;
    else b= -a;
```

➢ if (条件表达式1) 语句1;

```
else if (条件表达式2) 语句2;
else if (条件表达式3) 语句3;
    ......
else if (条件表达式m) 语句m;
else 语句n;
```

含义是：依次对条件表达式进行判断，其值为真时，则执行对应的语句，随后跳出整个if语句之外执行，若条件表达式均是假则执行语句n。

在编程过程中还可能用到if语句中的执行语句又是if语句时，则构成了if语句嵌套，一般形式可表示如下：

```
if (表达式)
if 语句;
```

或者为：

```
if (表达式)
    if语句;
else
    if语句;
```

在嵌套内的if语句可能又是if-else型的，这将会出现多个if和多个else重叠的情况，这时要特别注意if和else的配对问题。为了避免这种二义性，C语言规定，else总是与它前面最近的if配对。若if与else数目不匹配时可以加{}处理。

2. switch语句

switch语句是多分支选择的语句，又称开关语句。虽然if语句是一种选择语句，如果用多个if语句嵌套构成复合的条件语句也能实现多分支选择跳转，但这样会使程序可读性下降，而用switch语句实现多分支选择，程序就相对简单明了。

switch语句的一般形式：

```
switch(表达式)
{
    case 常量表达式1：    语句1；break；
    case 常量表达式2：    语句2；break；
    ......
    case 常量表达式n：    语句n；break；
    default:              语句 n+1；
}
```

switch语句执行时，会将switch后面表达式的值与case后面各个常量表达式的值逐个比较，当两个值相等，则执行对应的case后语句，语句也可以是多条，然后执行break语句中止当前语句的执行，使程序跳出switch语句。若都不相等，则只执行default指向的语句。

在单片机程序设计中，常用switch语句作为按键输入判别，并根据输入的按键键值跳转至各自的处理分支程序。

【例7-3】

```
input:   key_word=keyinput();
                switch(key_word)
        {  case 1:    key1();break;
                case  2:      key2();break;
                case  3:      key3();break;
                case  4:      key4();break;
           default: goto input;
        }
```

在这个例子中，keyinput()是对一个键盘扫描输入函数，若有按键按下，就对这个按键译码，将代表这个按键的键值赋予变量key_word。switch语句对键值进行比对，如果键值为1，则执行key1()函数，然后跳出switch语句。如果键值等于2，那么执行key2()函数后返回，并跳出switch语句，依此类推，使单片机能达到根据用户按下不同按键进行不同处理的目的。

7.5.2 循环控制语句

C51语言提供构架循环结构的循环控制类语句有while、do-while、for等语句。

1. while语句

while语句构成的循环结构一般形式如下：

```
while (条件表达式)
```

{ 语句 }

其含义是当条件表达式的结果为真（非0）时，则重复执行内嵌的语句（循环体），一直到条件表达式的结果变为假（为0）为止。该语句的执行特点是先判别条件表达式所给出的条件，再依据结果决定是否执行后面语句，因此，若条件表达式的值为假，则后面的语句可以一次也不执行。在单片机应用程序中时常会用到while（1）{ }；或while（1）；表示循环体为空。

【例7-4】 计算自然数0~50的累加和：

```
{
    int sum,i;
    sum = 0;
    i = 0;
    while(i<=50)
   {
        sum = sum +i;
          i++;
   }
}
```

2. do-while*语句*

do-while语句构成的循环结构一般形式为： do { 语句 } while (条件表达式)。

do-while循环与while循环语句的不同之处是先执行循环体语句一次，然后判断表达式是否为真，若真则继续执行循环体语句，若假则终止循环体语句而继续执行后面的语句。所以，do-while语句构成的循环结构，循环体内的语句至少会被执行一次。

【例7-5】 同样是计算自然数0~50的累加和：

```
{
    int sum,i;
    sum = 0;i = 0;
    do
   {
        sum = sum +i;
        i++;
   }
    while(i<=50);
}
```

3. for*循环*

for语句构成循环结构的一般形式如下：

for (表达式1；表达式2；表达式3；) 语句；

表达式1用于循环变量赋初值；表达式2用于判断循环是否结束；表达式3是每一次循环后对循环变量值进行增值，或减值。

【例7-6】 同样计算自然数0~50的和，用for循环可以写成如下语句：

```
{
    int sum,i;
    sum =0;
    for(i=1;i<=50;i++)  sum = sum +i;
}
```

首先循环语句先对i进行赋初值，i=1，接着判断循环是否结束（值为真），没有结束则计算s=s+i；若值为假则结束循环，最后i+1，然后再去判断循环是否结束，这样重复循环，直到i为51，才结束循环。

for循环中的表达式1、表达式2和表达式3都可以是相互独立的，并不要求三个表达式之间必须有相互依赖关系。另外，for语句中的这三个表达式都是可选项，即可以省略。但是分号“；”不能缺省。

使用for语句构成循环时，要特别注意表达式2（循环条件）应设置合理，使循环变量能在一定的循环后达到表达式2的循环条件，否则将可能使程序陷入死循环。

7.5.3 转向控制语句

C51语言中的转向控制语句主要包含转向语句goto、中止语句break、结束循环语句continue、返回语句return等。

1. goto*语句*

goto语句的一般形式：goto 标号。

其中，语句标号是一个带冒号“：”的标识符，用来表示程序的某个位置。同样计算自然数0～50的累加和，可以用goto语句写成如下程序：

```
{
    int sum,i;
    sum = 0;
    i = 0;
loop:  if(i<=50)
           {
               sum = sum +i;
               i++;
               goto loop;
           }
}
```

例子把goto语句和if语句配合使用，也可以构成循环结构，一般常见的是用goto语句来跳出多重循环。需要注意的是，goto语句是无条件转移语句，只能用来从内层循环跳到外层循环，而不能从外层循环跳进内层循环。

使用过多的goto语句容易使程序层次不清，且不易阅读，因此应尽量避免使用。但在多层嵌套退出时，用goto语句则比较合理。

2. break*语句*

break语句的一般格式：break;

前面例子中，我们已经用到了break语句。break语句只能用在switch语句和循环语句之中，功能是终止后面语句的执行或使循环立即结束，实际上它是一种具有特殊功能的无条件转移语句。但是，break语句只能跳出它所在的那一层循环，不像goto语句，可以从多重循环的内层循环中跳出。

3. continue语句

continue语句一般格式：continue ;

continue语句一般用在循环结构中，其功能是结束本次循环，即跳出本次循环体中continue语句后面尚未执行的语句，把程序流程转移到当前循环语句的下一个循环周期。

break语句与continue语句的区别是：break语句终止整个循环语句的执行，转到循环语句后的语句去执行；而continue语句只是结束本次循环，而不是终止整个循环语句的执行。

4. return语句

返回语句return用于结束函数的执行，并控制程序回到调用该函数时所处的位置。它有以下两种形式：

return（表达式）;

return;

对于第一种形式，return后面带有表达式，函数在返回前，将先计算出表达式的值，并将表达式的值作为该函数向主调用函数得返回值。若使用不带表达式的形式，被调用的函数返回主调用函数时，函数值不定。

一个函数内部也可以没有return语句，此时，当程序执行到最后一个界限符“}”处时，就如同插入一个return语句一样，自动返回主调用函数。

7.6 C51数组与指针

学习目标

1. 掌握数组的概念、定义与应用。
2. 理解指针变量的概念。
3. 熟练掌握指针在程序中的应用。

导　入

用单片机进行显示器控制应用时经常会用一些固定数据的循环执行，这些相同数据的集合就是C51中数组的应用；当应用程序比较大时涉及数据类型比较多时就会用简化的手段——指针。

7.6.1 数组

数组是一种由基本数据类型构造而成的数据类型，是同一数据类型的数据的有序集合。用一个数组名与下标可以唯一确定数组中的元素。数组对应在内存的是一段连续的存储空间，最低位地址序号对应数组第一个元素。数组可以有一维、二维、多维的数组。对于n维数组的定义形式如下：

类型说明符 数组名 [常量表达式1] [常量表达式2]……[常量表达式n];

当n=1则为一维数组

当n=2则为二维数组

例如：定义一个具有6个元素的一维整型数组x和一个具有3×6个元素的字符型二维数组y，代码如下：

```
int x[6];
char y[3][6];
```

数组元素从0开始，而不是从1开始。所以x[6]是定义了x[0]～x[5]的6个元素；y[3][6]定义了3行6列共18个元素。

在定义数组的同时给数组中各元素赋初值的一般形式为：

[存储器类型] 数据类型 数组名 [常量表达式] = {值，值，值…值}；

初值的个数必须小于或等于数组中元素的个数。若初值的个数小于数组中的元素个数，则后面的元素都用0来填补。在赋初值时，也可以不定数组的长度，编译器会根据初值的个数自动计算出该数组的长度。下面是几个数组赋值的例子：

```
例如：int x[ ]={1,2,3,4,5};
      int y[5]={1,2,3,4,5};
      char z[7]={'h','e','l','l','o','\0'}
```

数组名可以作为函数的参数。用数组名作函数参数时，参数传递方式采用的是地址传递。用数组作为函数的参数，必须在主调函数和被调函数中分别进行数组定义，而且在两个函数中定义的数组类型必须一致，但实参数组和形参数组的长度可以一致也可以不一致。

【例7-7】用数组作为函数的参数，计算两个不同长度的数组中所有元素的和：

```
#include <reg51.h>
int sum1,sum2;
int sum(int array[],char n)                    //数组元素求和函数
{
    char i;
    int temp=array[0];
    for(i=1;i<n;i++)temp=temp+array[i];
    return(temp);
}
main()
{
    int x[5]={12,23,34,45,56};
```

```
    int y[8]={96,123,75,105,36,48,95,84};     //x、y数组定义并赋值
    sum1=sum(x,5);
    sum2=sum(y,8);                            //分别调用数组元素求和函数
}
```

7.6.2 指针

指针是C语言中的一个重要应用，编程时正确运用指针，可有效地表示复杂数据结构，有效地使用数组，方便使用字符串以及调用函数，且能直接与内存打交道。应用指针能使编写的程序紧凑、效率高，但是对于指针概念的掌握却比较困难。

1．指针变量

在8051汇编指令的寻址方式中有一种寻址方式称为“间址寻址”。

例如：MOV　A, @R0

其含义是表示在工作寄存器R0中存放的是操作数的地址，通过地址才能访问存储器中的操作数，然后取出操作数送累加器A。在C语言规定所有变量在使用前必须指定其数据类型，并按此分配存储单元，而存储单元都是有地址的，同样，对于数组等均是如此。在C语言中取操作数的方法与汇编指令中取操作数的方法相同，都是直接或间接地取得。

在C语言中把内存的地址称为指针，是常量，若把指针存放在一个变量中则称为指针变量，一般简称为指针。指针类型也属于一种数据类型。

例如：int　*cp1;

```
char *ptr1,*ptr2;
```

第一行定义了指针变量cpl，它们是指向整型变量的指针；第二行定义了两个指针变量ptr1和ptr2，它们是指向字符型变量的指针。定义指针变量的一般格式是：

基类型 *指针变量名；

定义指针变量时需注意：

- 指针变量前面的“*”，表示该变量的类型为指针型变量。如：*ptr1、*ptr2。
- 定义指针变量时必须指定基类型。

不同类型的数据在内存中占用的字节数不同。对于C51而言，unsigned char型变量在内存中占用1个字节；unsigned int型变量在内存中占用2个字节，有的为4个字节。在指针的操作中，常用的一种操作是指针变量自增。如ptr++，其意义是将指针指向这个数据的下一个数据，若1个字节对应一个数据，则每次指针自增时，只要将地址值增加1；如果2个字节表示一个数据，每次指针自增时，就必须将该值增加2，这才是指向下一个变量地址，否则结果就不正确。对于不同数据类型，必须明确定义指针变量时指定的基类型。为了表示指针变量与它所指向的变量地址之间的关系，C语言为指针运算专门设置了两个运算符：

◆ &：取地址运算符，取得其后所跟操作数的地址，如：

```
int i，*ip;
ip = &i;
```

将变量i的地址赋予ip。如果i的地址是0x40，那么赋值后ip的值就是0x40。

◆ *：指针运算符（间接访问运算符），通过地址访问变量。

例如：i = 10;　　ip = &i; j = *ip;

j中存放的也是10，因为ip指向i的地址，*ip通过ip中存放的地址，然后取这个地址（即i的地址）中的值，最后赋值给j，也就是说，*ip也表示变量i，但要注意与指针变量定义的含义区别。

C51支持“指定存储区”和“通用”两种指针类型。

2. 通用指针

C51提供一个3字节的通用指针，用以说明指针的存储类型，通用指针的用法与标准C语言相同。例如：

➢ long * big；

big是指向long型整数的指针，而big本身则存放在不同的RAM区。

➢ char * xdata ptr；

ptr是指向char数据的指针，而ptr本身存放于外部RAM区。long、char等指针指向的数据可存放于不同的存储器中。

通用指针由3个字节构成。指针的第一字节表明指针的存储区类型，另外两个字节为16位偏移量，如表7-6所示。

表 7-6 通用指针构成

地址	+0	+1	+2
内容	存储区类型	偏移量高位	偏移量低位

存储区类型编码值，如表7-7所示。

表 7-7 通用指针存储区类型编码

存储区类型	idata	xdata	pdata	data	code
值	1	2	3	4	5

例如，一个通用指针指向地址为0x1234的xdata类型数据时，其指针值如表7- 8所示。

表 7-8 指向xdata类型数据的通用指针值

地址	+0	+1	+2
内容	0x02	0x12	0x34

由于存储区在运行前是未知的，编译器不能优化存储区访问，必须产生可以访问任何存储区的通用代码，所以通用指针产生的代码比指定存储区指针代码的执行速度要慢。若要优化执行速度，应该尽可能地用指定存储类型的指针，而不用通用指针。而在C51库函数中通常使用这种指针类型。

3. 指定存储区专用指针

Keil C5l允许使用者规定指针指向的存储段，这种指针叫指定存储区指针。例如：

```
char data *str;                //str指向data区中char型数据
int xdata *ptr;                //ptr指向外部RAM的int型整数
```

从上述语句可以看出，这种指定类型的存储类型在编译时是确定的，通用指针所需的存储类型字节在指定存储器的指针中是不需要的，指定存储区指针只需用1字节(idata、data、bdata和pdata指针)或2字节(code和xdata指针)。

使用指定存储区指针的好处是节约存储空间，编译器不为存储器选择和决定正确的存储器操作指令产生代码，使代码更加简短，但使用时必须保证指针不指向所声明的存储区

以外的地方，否则会产生错误。

4. 指针与数组

变量在内存中是具有地址的，数组同样如此。对于数组而言数组在内存存放的首地址就是数组的指针，数组和数组中元素的引用可以使用指针变量。

例如：int　a[6]，*ptr；

ptr=&a[0];(或者ptr=a;)

通过上述语句执行，则ptr得到了a数组的首地址。而*（ptr+2）与*（a+2）均表示为数组元素；ptr+3与a+3表示了a[3]这一元素的地址。这是说明了指针与一维数组之间的关系。

对于二维数组a[5][6]共有30个元素，&a[0][0]与a都是表示二维数组的首地址。a[0]表示的是数组a[5][6]的第0行的首地址；同样也是a数组的首地址；a[5]就是5行首地址。

若数组元素是地址，这类数组既具备了数组的特征，且由几个指向数据类型的指针元素组成，其定义为指针数组。

定义形式为： 数据类型 *指针数组名 [元素个数]= {地址表}；

例如：int *ptr[5]；

5. 指针运用

在用C51编程过程中，若需要访问单片机存储空间的地址，时常会运用以下指针类型的变换形式。

- 把一个放在0x2000中的数据转换成code存储区域中的char类型指针。

  i_p=*（（char code*）0x2000）
- 把一个放在0xF5中的值转换成idata存储区域中的char类型指针。

  i_p=*（（char idata*）0xF5）
- 把一个变量j（其类型为int data*）的地址转化为一个idata中存放的char指针。

  i_p=（char idata*）&j

在C语言中这是一种强制类型转换方法，强制类型转换也可以用在表达式中。例如：（int）Y/5；其含义是结果是int型。

7.7 函数

学习目标

1. 理解函数的定义，形式参数与实际参数的概念。
2. 掌握函数的应用。
3. 掌握中断函数的定义与应用。

导　入

C51语言中编写程序的主要工作就是函数的编写，通过多个函数的有机组合并恰当运用好参数的传递所起到的效果，就相当于把大程序转化成小程序（函数）来处理。

对于用C语言编写程序，化大事情（大任务）为小事情的方法（单一任务）就是通过编写函数来实现。C语言结构化程序的集中体现，就是函数的应用，而这种函数不是一个内容很多的大函数而是一种描述单一任务的小函数，由小函数组成大程序，可以认为函数是C51语言的基本单位。

在一个C语言应用程序中定义一个主函数main（），程序的执行就是从主函数main（）开始，在主函数运行过程中可以有若干个函数被主函数调用或是函数之间相互调用，而在每个函数调用结束后进行返回，最终调用后再返回到main函数，由main函数结束整个程序。函数也可以自己调用自己，称之为递归调用。

7.7.1　函数定义

函数可分为标准库函数和自定义函数两种。Keil C51编译器中，提供了许多库函数，这些库函数是最常用的，而且是经过反复验证的，所以应尽量直接调用，以减少程序编写的工作量并降低出错的概率。使用时不必作类型说明，只须在程序前包含有该函数原型的头文件就可以在程序中直接调用。但是一些单片机特定控制功能方面程序，只能根据需要自行定义编写函数。函数必须先定义，然后才能使用。

函数定义的一般形式为：

```
存储类型 函数类型  函数名 （形式参数表）
{
    局部变量定义
    函数体语句
}
```

其中，存储类型指出的是函数作用范围，static在所在的文件中有效，属于内部函数，extern可以被其他文件调用是外部函数。函数类型说明了定义函数返回值的数据类型，可以是指针类型，对于无返回值的函数，函数类型可以写为void。形式参数表中列出了在主调函数与被调用函数之间传递的指定数据类型的变量，若是多个形式参数可以用逗号分隔。也可以是无参数函数，只用一个空括号，圆括号后面也不能用分号。函数也可以什么执行也不做，也无返回结果，仅仅是一个占位符作用，即空函数。

【例7-8】 求取两数中大的数的函数：

```
int  max(int x,int y)
{
    if(x>y) return x;
    else  return y;
}
```

例题定义了一个函数名为max的求两个变量之间大的值的函数，函数有两个整型的形式参数x和y，{ }内为函数体，描述函数功能部分，由于函数体内没有其它变量，所以无须定义变量类型，return语句用来返回变量x或者y的值给主调用函数，属于int型。max函数可以称为被调用函数。若函数不需要返回值，则可以将函数定义为void类型，且可以不用return语句。

7.7.2 函数调用

函数定义后，可以在主调函数内调用被调用函数，与变量相同的是函数也要先说明后调用，说明函数的形式为：函数类型 函数名（）；其中函数类型是数据存储类型。

例如：int put(intx,inty,char * p);

而调用函数的一般形式为： 函数名（实际参数表）

其中，函数名指被调用的函数名，实际参数表中可以包含多个实际参数，实际参数可以是常数、表达式（有确定的值）等并用逗号分隔，其作用是将值传递给被调用函数中的形式参数。如果被调用的函数无形式参数，则调用时不需要实际参数，但圆括号仍要保留。

例如：total=max(5.0, 8.0);

total=max(max(5.0,7.0), 8.0);

调用函数一般有以下三种方式：

- 函数语句。在主调用函数中将函数作为一条语句。
- 函数表达式把函数调用作为一个运算对象直接出现在主调用函数的表达式中。
- 函数值作为参数，在主调用函数中将函数调用作为另一个函数调用的实际参数。

如果调用的是标准库函数，一般应在程序的开始处用预处理命令#include <> 将有关函数说明的头文件包含进来，这样就不用再另行说明了。以下是Keil C51中常用的一些库函数。

- STDIO.H：一般I/O函数
- CTYPE.H：字符函数
- STRING.H：字符串函数
- MATH.H：数学函数
- STDLIB.H：标准函数
- ABSACC.H：绝对地址访问函数
- INTRINS.H：内部函数

形式参数和实际参数的功能是数据传递。在主调函数调用被调用函数时，主调函数把实际参数的值传送给被调函数的形式参数，从而实现主调函数向被调函数的数据传送。函数的形式参数和实际参数具有以下特点：

实际参数可以是常量、变量、表达式、函数等，在对被调函数进行调用时，实际参数必须要有确定的值，因为传送给形式参数的是确定的值。

形式参数变量只在被调用时才分配内存单元。一旦调用结束，立刻释放所分配的内存单元，所以形式参数只在函数内部生效。函数调用结束返回主调函数后不能再用该形式参数变量。

实际参数和形式参数在数量、类型和顺序上应严格一致，否则在编译时会发生“类型不匹配”的差错。

函数调用中所发生的数据传送是单方向的，即把实际参数的值传送给形式参数，而不能逆向，因此在函数调用过程中，形式参数的值会发生改变，而实际参数中的值是不变化。

7.7.3 中断服务函数

中断控制方式可以使CPU的运行与中断事件并行，当外界有异常情况时可以充分利用单片机的中断功能，从而提高CPU的工作效率，尤其是在编制实时性要求较高的控制程序时。

Keil C5l支持在C语言源程序中直接编写MCS-51单片机的中断服务函数程序，为此Keil C51对函数的定义进行了扩展，增加了一个interrupt关键字。若在函数定义时加上interrupt选项，就可以将一个函数定义成中断服务程序。

定义中断服务函数程序的一般形式是：

函数类型 函数名（）[interrupt m][using n]

其中：关键字interrupt后面的m是中断号，m的取值范围为0~4。MCS-51单片机的中断号、中断源和中断向量如表7-9所示。Keil C51编译器从8m+3处产生中断向量，即当响应中断申请时，程序会根据中断号自动转入地址为8m+3处，执行相对应的中断服务函数。

表 7-9 MCS-51单片机中断号、中断源和中断向量表

m	中断源	中断向量8m+3
0	外部中断0	0x0003
1	定时器/计数器0溢出	0x000B
2	外部中断l	0x0013
3	定时器/计数器1溢出	0x001B
4	串行口中断	0x0023

当用汇编指令编写程序时，通常在中断入口处放一条转向指令。同理Keil C51编译时会自动在对应的中断入口地址单元中插入一条转移指令，以便转入中断服务程序。Keil C51编译器会自动将PC寄存器压入堆栈，在中断服务程序的最后自动安排RETI指令，以便清0响应中断时所置位的优先级状态触发器，然后从堆栈中弹出程序计数器PC，从产生原来中断的断点处继续执行被中断的程序。

中断服务函数可以被放置在程序的任意位置。因为Keil C51在最后进行代码连接时会自动将中断服务函数定位到中断入口处，实现中断服务响应。

using n可选项用来指定80C51中不同的工作寄存器组，MCS-51系列单片机可以在内部RAM中使用4个不同的工作寄存器组。n的取值为0~3的常整数，分别代表4个不同的工作寄存器组。

CPU在响应中断进入中断服务函数时，特殊功能寄存器ACC、B、DPH、DPL、PSW都将被保存至堆栈。如果不使用寄存器组切换，则中断函数中所用到的全部工作寄存器也都会入栈。函数返回前，再依次弹出。但如果在中断函数定义时用using指定了工作寄存器

组，则当中断发生时，默认的工作寄存器组就不会被压栈，也就是说系统将直接切换寄存器组而不必进行大量的PUSH和POP操作，这将节省32个处理周期，因为每个寄存器入栈和出栈都需要2个处理周期。由此可以节省RAM空间，加速CPU执行时间。

但缺点是所有调用中断的过程都必须使用指定的同一个寄存器组，否则，参数传递会发生错误。所以对于using的使用应当有所选择。中断函数一般不进行参数传递，同时中断函数没有返回值，否则将会得到错误的结果。中断服务函数不能被其他函数调用，而只能由硬件产生中断后自动调用。依据硬件中断系统的工作原理，由于在中断函数程序执行过程中，对可能在此时产生的另一个中断不响应，因而为了系统能够及时地响应各种中断，加强系统的实时性能，中断函数的执行时间不宜过长，应尽量简洁。

7.7.4 函数与指针

函数名与数组名类似代表着函数的首地址，所以也可以把函数名赋给指针变量，指向函数的指针简称函数指针。

函数指针的定义如下：数据类型 （*函数指针名）（）；

例如：int (*max)( )

其调用格式为：（*函数指针名）（）；

运用时其中的刮号是不能省略的。

当一个函数的返回值是地址时，该函数是指针函数。指针函数的说明语句形式为：

存储类型 数据类型 *函数名（）

指针函数的定义形式为：

存储类型 数据类型 *函数名 (形式参数表) { 语句 }

例如：int (*max)(a, b)

表示用return（）语句返回的值是指针，指向的是整型类型。指针函数的调用形式为：指针函数名(实形参数表);

需要指出的是指针变量、指针数组元素等还同样可以当作函数的参数进行传递，这时传递的值均是地址值。

7.8 C51结构、联合、枚举与编译预处理

学习目标

1. 熟悉结构体的格式与应用。
2. 熟悉共用体的格式与应用。
3. 熟悉枚举类型的格式与应用。
4. 熟悉预处理格式与方法。

导　入

对于不同类型的数据运用，前面的基本数据类型以及构造类型数组是不能解决问题的，C51语言允许自行定义一种结构体、共用体或者枚举类型来处理不同数据类型的集合。对于编译器也提供了对于源程序的预处理。

7.8.1　结构体

结构体是C语言构造类型数据结构之一，结构体是由几个不同数据类型变量有序地组合在一起而形成的一种数据的集合。组成该集合体的各个数据变量称为结构成员，整个结构体用一个标识符来命名。

一般结构中的各个变量之间相互关联，例如日期数据中的年、月、日，人员数据中的姓名、性别、年龄等。结构体将一组相关联的数据变量作为一个整体进行处理，这有利于对一些复杂而又具有内在联系的数据进行有效的管理。

1. 结构体类型的定义

结构体类型的一般定义格式为：

```
struct［结构名]
    {类型标识符    成员名;
     类型标识符 成员名;
    ...
    }[变量列表];
```

例如：定义一个结构名为date的由三个结构体成员year、month、day组成结构类型：

```
struct date
    {
      int  year;
      char  month,day;
    }
```

定义结构类型后，可以用以下几种方式定义结构变量：

例如： struct date d0,d1;

这是先定义结构类型，再定义结构的变量名。

例如：struct date

```
    { int year;
      char month,day;
    }d2,d3;
```

这属于在定义结构类型的同时定义变量名，也可以采用不定义结构体类型名而直接定义结构变量，与变量类似，结构体变量也需要赋初值。

2. 结构体变量的引用

在定义了结构体变量后，就可以对变量进行赋值、运算。由于这是对变量的运算而不

是对结构体的操作，所以只能分别引用结构变量的各成员。引用的格式为：结构体变量名.成员名；这里的点（.）是成员运算符，具有最高的优先权。

例如：d1.year=2012;

sum=d0.day+d1.day;

上述例子包含初始化。

结构体类型变量在解决实际问题时作用不大，往往采用的是结构体数组形式，或者是结构体指针运用。

例如：
```
struct stu
{   char name[10];
    int age;
    float score
};
struct stu student[30]
```

或者 struct stu *p1，*p2；p1，p2分别是指向结构体类型变量，指针的引用方式为：结构指针名->成员。

例如：
```
student[30]. name[2];
p1-> age=20;
```

或者（*p1）.age=20;

结构体类型的变量与结构的指针变量，在使用时同样需要注意是属于局部还是全局变量。

7.8.2　共用体

结构体数据的特点是其成员的不同数据类型组合在一起，它们各自占有一定的内存空间，而共用体可以使各种类型的数据共同使用同一块内存空间，在应用时是根据需要使用其中的一个成员，而且是最新的一个成员，以提高内存的效率，共用体也可称之为联合体。

1. 共用体定义

共用体定义的格式如下：

```
union [共用体名]
{   成员列表
  }[变量列表];
```

例如：
```
union data
  {
  float x;
  int y;
  char z;
  }obj;
```

上例定义了一个共用体类型data，其中包含一个float型变量、一个int型变量、一个char型变量，这三个变量放在分时共享同一个地址开始的内存单元，obj为data类型共用体变量， 也可以把类型定义与变量定义分开，即先定义一个union data类型，再定义obj。

共用体类型与结构体类型的定义方法基本类似，只是将关键字struct改成union，但是在内存的分配上两者却有本质的区别。结构体变量所占用的内存长度是其中各个元素所占用内存长度的总和，而共用体变量所占用的内存长度是其中最长数据类型成员所占有的长度。共用体与结构体可以互为嵌套定义。

例子中，float类型数据需要四个内存存储单元，多于int型和char型数据，因此，系统将按照四个字节长度为obj分配内存存储空间。若把上述例题改为结构体则一共要占用七个内存存储空间。由此可见，合理地运用共用体类型数据将是很有益的。

2. 共用体引用

与结构体变量类似，共用体也是通过对其共用体元素的引用来实现共用体变量的引用。引用共用体成员的一般格式是：

共用体变量名.共用体成员 或 共用体指针名→共用体成员。

例如： obj.x; obj.y; obj.z ;

union data *p; p->y=12;

但是需要注意的是同一个时刻只能引用一个成员。

7.8.3 枚举类型

在实际问题中，变量的取值被限定在一个有限的范围内。例如，一年只有十二个月，一个班级只有三十个人等。为此，C语言提供了一种称为“枚举”的类型，它是一个有名字的一些整型常量的集合，这些整型常量是该类型变量可取的所有合法值。例如表示星期的Sunday，Monday，Tuesday，Wednesday，Thursday，Friday，Saturday就是一个枚举类型。枚举数据类型可以更清楚表达某类问题。

枚举类型的定义形式为：

```
enum 枚举名
        {
        标识符[=整形常数],
        …
        标识符[=整形常数],
}[枚举变量];
```

枚举类型的定义和说明也可以分为分开完成：

```
enum   枚举名   {枚举变量列表};
enum   枚举名   枚举列表;
例如：  enum weekday{sun,mon,tue,wed,thu,fri,sat}d1,d2;
或       enum weekday{sun,mon,tue,wed,thu,fri,sat};
         enum weekday d1,d2;
```

在枚举值表中应罗列出所有可用值，即枚举元素。枚举变量只能取枚举说明结构中的某个标识符常量。如上例子中，该枚举名为weekday，枚举值共有七个，表示一周中的七天。被说明为weekday类型变量的取值只能是七天中的一天。

枚举列表中，每一项符号代表一个数值。如果枚举没有初始化，即省掉“整型常数”时，则从第一个标识符开始，顺次赋给标识符0，1，2……也就是第一项符号取值为0。同

样当枚举中的某个成员赋值后，其后的成员也按依次加1的规则确定其值。在枚举类型应用中需注意的是：枚举中每个成员（标识符）结束符是“，”，而非“；”，最后一个成员可省略逗号“，”；初始化时可以赋负数，以后的标识符仍依次加1；枚举值是常量，而非变量。只能把枚举值赋予枚举类型变量，不能把成员的数值直接赋予直接赋予枚举变量，如一定要把数值赋予枚举类型变量，则必须用强制类型转换。

7.8.4　编译预处理

由C语言编写的源程序的开始部分通常是预处理命令，其作用是在编译时告知编译器在对程序进行编译时，将所需要的“头文件”读入后再一起编译，最终生成扩展名为exe的可执行文件。

C51提供三个预处理命令：宏定义、文件包含、条件编译。

1. 宏定义

“宏定义”是指作一些符号的替换，宏定义可以分为不带参数宏定义格式和带参数宏定义格式。不带参数宏定义格式为：

#define　标识符　字符串

例如：#define　PI　3.1415926

使用时要注意：

① 语句#define一般放在函数外面，作用的范围是从语句开始到文件结束，但是可以使用#undefine进行终止作用。

② #define语句是非正式语句，书写时后面不用分号。

③ 标识符使用时尽量使用大写，与变量名有一种明显的区分，使用宏定义可以减少许多重复输入，可移植性提高。

带参数宏定义格式为：#define　宏名（参数表）　字符串。

2. 文件包含

应用文件包含命令可以把外面的源文件添加到当前文件中来。一般格式为：

#include　“文件名”

例如：　#include “REG52.H”。

#include < REG52.H >。

在REG52.H（扩展名为H的头文件）中定义了特殊功能寄存器的字节地址和位地址。把REG52.H文件包含进C语言程序后，在C语言程序中就可以直接使用这些特殊功能寄存器以及特殊功能寄存器的特定位。

使用时要注意：#include命令行放在第一行；一个#include命令只能包含一个文件，有几个命令则写几个命令行；当使用“”时，包含的文件是在当前目录中查找被包含的文件名；当使用< >包含头文件时，编译器先进入到软件安装文件夹处开始搜索这个头文件，也就是Keil \ C51 \ INC这个文件夹下，如果这个文件夹下没有引用的头文件，编译器将会报错。

3. 条件编译

一般情况下编译器对所有的程序行都要进行编译，而有时会遇到某种条件满足就进行编译，不满足则跳过不进行编译。

条件编译格式：

```
#ifdef  标识符
    程序段1
#else
    程序段2
#endif
```

其意义是：当定义了标识符则对程序段1进行编译；否则对程序段2进行编译。另外还有两种条件编译的格式：

```
#ifndef  标识符
    程序段1
#else
    程序段2
#endif
```

由于#ifndef是指没有定义过标识符，则运行程序段1，否则运行程序段2。

```
#if 标识符
    程序段1
#else
    程序段2
#endif
```

当指定表达式的真为真，则编译程序段1，否则编译程序段2。

知识拓展

要求通过查阅资料设计一个硬件系统：除了单片机最小系统外，扩展16个按键与4只LED显示器。然后用C51编写键盘程序，要求当按下第1个按键，显示器显示字符0；按第二个按键显示器对应显示字符1；依次类推；当按下第16个按键时显示字符F。

本章小结

本章围绕C51语言在单片机程序开发过程中的应用，紧扣C51语言的基本点，着重于C51语言与单片机硬件的结合，先后概述了C51语言的特点，标识符，关键字、基本与扩展数据类型、运算符组成的各类表达式，常量与变量的特点与应用，数据在单片机存储区域中存储模式应用；条件选择语句、循环控制语句的基本语法与基本应用；指针与数组的基本概念与基本的应用；函数的定义、函数的调用以及中断函数，结构体的定义与引用，共同体的定义与引用，枚举类型以及编译预处理。通过本章学习可以独立编写和调试C51源程序。

思考与练习题

一、单项选择题

1. C51语言编写的程序的基本单位是（　）。

A. 字符　　B. 语句　　C. 程序行　　D. 函数

2. 以下不正确的标识符是（　）。

A. CBA　　B. C. BA　　C. c_ba　　D. cba

3. bit可以定义成（　）。

A. 位变量　　B. 位指针　　C. 位数组　　D.SP中的位

4. 关键字interrupt表示的函数是（　）。

A. main函数　　B. 一般的函数　C. 中断函数　　D. 被调函数

5. 若设int i,j,k，则下列语句中符合语法的语句是（　）。

A. if（i=j）++k　　B. if（i=>j）++k

C. if（i=<j）++k　　D. if（i<>j）++k

二、判断题

1. C51语言必须有且只能有一个main函数。（　）

2. sfr定义的数据类型是针对MCS-51单片机硬件的且是ANSI C的一部分。（　）

3. 语句char data *p定义p是指向外部RAM区中的char型数据。（　）

4. C51语言虽然属于高级语言但是可以内嵌汇编程序。（　）

5. 应用C51语言的优点之一是寄存器的分配由编译器管理。（　）

实验项目7　C51语言的调试与应用

一、实验目的

1. 掌握C51语言基本应用。
2. 掌握Keil C51环境下调试程序的基本方法。
3. 掌握中断函数的应用。
4. 掌握C51编程与AT89C51硬件系统的对应关系。

二、实验设备

1. PC计算机一台
2. 单片机实验装置一套（教材配套）
3. Keil μVision2 集成开发环境
4. STC_ISP_V480在系统编程软件

三、实验内容

应用C51语言编写单片机对LED的控制程序。

四、实验程序

1. 采用定时器0中断的方式，实现用AT89C51的P0口控制的 LED 发光二极管，每一秒闪烁一次（亮暗各半秒）的程序实例。

```
#include <reg51.h>
sbit CS=P2^7;
unsigned char data count1,count2;
main()
{
        CS=0;
        count2=0xAA;
         TMOD=0x01;      //定时器0为方式0
           TH0=0x3C;     //12MHz晶振，50ms定时，计数初值=0x3CB0
           TL0=0xB0;
             IP=0x02;    //定时器0为高优先级中断
             IE=0x82;    //允许定时器0中断
TR0=1;                   //启动定时器0
             count1=0;
             while(1);
}
```

```
void time0_int(void)interrupt 1 using 1
{
        TH0=0x3C;
        TL0=0xB0;
        count1++;
        if(count1==10)
        {
          count1=0;
          P0=count2;        //每隔500ms，取反P0
        count2=~count2;
          }
}
```

2. 把例7-1程序补充完整，实现用键盘控制LED发光二极管的C51程序。

五、实验操作

搞清楚实验线路的基本工作原理，在阅读并理解源程序的基础上，完成实验程序的调试运行。

在Keil μVision2 集成开发环境下，源程序进行编辑，然后汇编直至通过，形成*.HEX文件。（注意：源程序扩展名为*.c）。

C51语言程序结构同C语言结构并无差异，但是C51语言要结合硬件、编译程序所以对于标准ANSIC在数据类型、存储器类型、存储器模型、函数、指针等几个方面的扩展。要有所侧重这是快速掌握C51的有效方法之一。单片机的C51程序语言同标准的C语言一样，也包含特定的标识符和关键字

第8章 单片机的扩展技术

MCS-51单片机有运算器、程序存储器、数据存储器、寄存器、定时器、三总线和输入/输出口等。但是它的输入/输出端口数量有限，程序存储器和数据存储器的容量不够大，特别是数据存储器的容量太小，在很多情况下满足不了实际需要。因此就要根据实际情况适当地进行单片机外部功能的扩展，而且MCS-51单片机是属于通用型和可扩展型的单片机。掌握单片机外部器件的扩展设计，对单片机应用系统的设计非常重要，本章将主要介绍并行I/O口、串行数据存储器、A/D、D/A转换的接口扩展应用。

8.1 并行I/O口的扩展技术

学习目标

1. 了解单片机的I/O口结构。
2. 熟悉开关量输入输出接口芯片的性能与应用。
3. 掌握开关量输入输出接口芯片扩展硬件设计方法。
4. 掌握开关量输入输出接口芯片扩展软件设计方法。

导　入

一般计算机的USB接口数量有限，常常因为同时要使用多个U盘、照相机等设备而不够用。会使用U盘扩展器，使计算机的USB口增加，大大方便了应用。同样，单片机的I/O口也是有限的，常常会因为接口不够而无法使用，通过对单片机的I/O口扩展，增加单片机的I/O口，就可以满足各种应用的需要。

8.1.1 74LS377的性能与应用

在单片机应用系统中，利用TTL芯片，CMOS锁存器和三态门等接口芯片将P0接口扩展。如图8-1所示是一个74LS377集成电路，74LS377是8D锁存器，可以直接接到P0口线上。74LS377输出能够驱动8个LED灯。

引脚说明：

$\overline{CS}$　　允许控制端（低电平有效）

1D~8D　数据输入端

1Q~8Q　数据输出端

CLK　　时钟输入端（上升沿有效）

当允许控制端$\overline{CS}$为低电平时，时钟端（CLK）脉冲上升沿作用下，输出端Q与数据端D相一致，当CLK为高电平或者低电平时，D对Q没有影响。74LS377与单片机的扩展连接如图8-2所示。

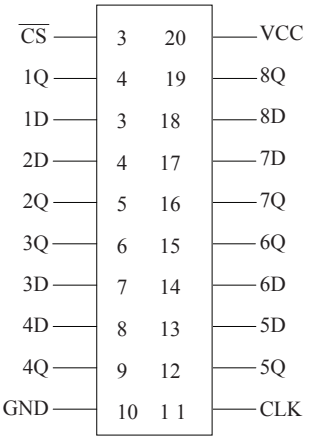


图 8-1　74LS377引脚排列与封装图

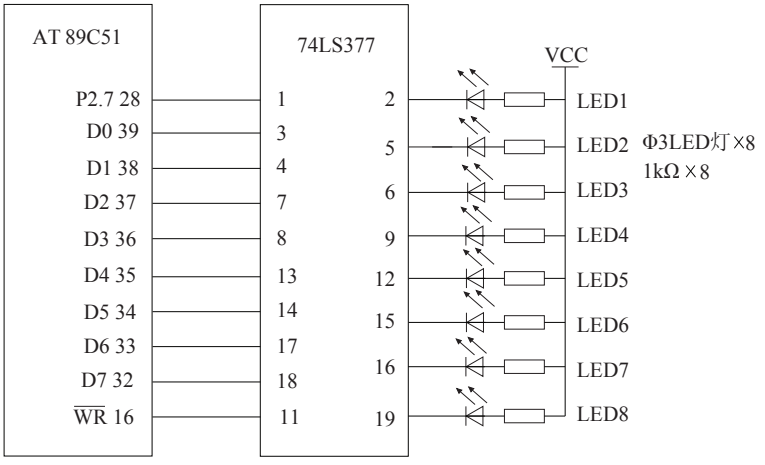


图 8-2　74LS377的扩展

如图8-2，单片机的P27（28）端作为74LS377的片选地址，$\overline{WR}$（16）作为写信号，当单片机向74LS377写入数据时，74LS377的某端口为“0”，则相应的LED灯亮，74LS377的某端口为“1”，则相应的LED灯暗。单片机以总线形式连接74LS377锁存器可以扩展数字量输出口。

8.1.2 74LS244的性能与应用

74LS244是三态总线转换器件，8输入3态缓冲电路，把八个输入分成两组，四个一组，也叫做线驱动或者总线驱动门电路。它有八个输入端，八个输出端。一般用于单片机的输入接口的扩展，也可以用于总线的电平匹配转换。图8-3是74LS244集成电路的引脚排列。

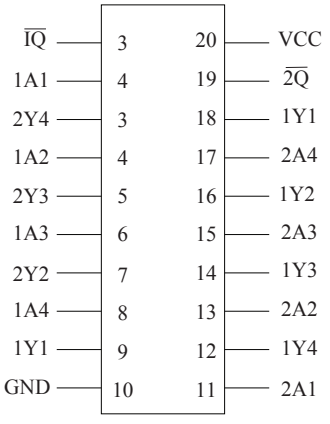


图 8-3　74LS244引脚排列

引脚说明：

$\overline{1Q}$：1Y1-1Y4输出控制，低电平有效，高电平高阻态

$\overline{2Q}$：2Y1-2Y4输出控制，低电平有效，高电平高阻态

1A1：输入端，对应的输出为1Y1

2A1：输入端，对应的输出为2Y1

当输出控制端$\overline{1Q}$、$\overline{2Q}$为低电平时，数据由输入端到输出端。输出控制端$\overline{1Q}$、$\overline{2Q}$为高电平时，总线为高阻态。74LS244与单片机的扩展连接如图8-4所示。

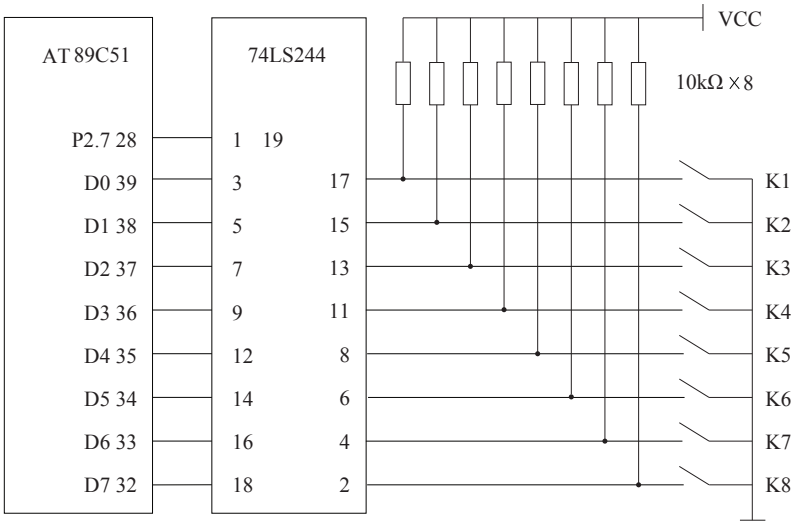


图 8-4 74LS244的扩展

如图8-4，单片机的P2.7（28）端作为74LS244的片选地址，在P2.7（28）端为低电平时，单片机可以读入的数据。数据输入端的开关K在断开状态，单片机读到的是“1”，数据输入端的开关K在合上状态，单片机读到的是“0”。同样单片机以总线形式连接74LS244缓冲器实现数字量输入口的扩展。注意本图中省略了$\overline{RD}$读入引脚。

8.2 单片机存储器扩展技术

学习目标

1. 了解单片机存储器的结构。
2. 熟悉存储器性能与应用。
3. 掌握单片机存储器扩展硬件设计方法。
4. 掌握单片机存储器扩展软件设计方法。

导　入

我们在日常生活中会使用各种各样的信息存储设备，例如交通卡、银行卡等。那么这些存储器或存储卡的工作原理是什么？如何在单片机应用中设计一个存储器来存放数据或信息？应该选择什么样的存储器，如何设计硬件电路和编写程序？如果要设计一个温度测量仪表，并且要记录大量的温度信号，就必须在单片机之外增加一些数据存储器才能够实现，这就是单片机需要扩展存储器的原因。由于技术的进步，单片机内部的程序存储器容量增加了，只要选择适当的单片机型号，一般不需要对单片机的程序存储器进行扩展，主要是数据存储器的扩展。

8.2.1 存储器的分类

一般存储器有磁芯存储器和半导体存储器两种，目前存储器都采用半导体存储器。半导体存储器从使用功能上分，有随机存储器（Random Access Memory，简称RAM），又称读写存储器和只读存储器（Read Only Memory，简称为ROM）。随机存储器可以读出，也可以写入。读出时并不损坏原来存储的内容，只有写入时才修改原来所存储的内容，断电后，存储内容立即消失，即具有易失性，主要作为数据存储器。RAM可分为动态（Dynamic RAM）和静态（Static RAM）两大类。DRAM的特点是集成度高，主要用于大容量内存储器；SRAM的特点是存取速度快，主要用于高速缓冲存储器。

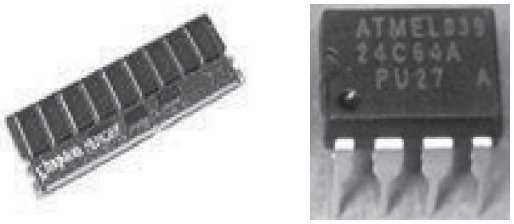
只读存储器的特点是只能读出原有的内容，不能由用户再写入新内容。原来存储的内容是采用掩膜技术由厂家一次性写入的，并永久保存下来。它一般用来存放专用的固定的程序和数据，不会因断电而丢失。另外还有一种CMOS存储器（Complementary Metal Oxide Semiconductor Memory），COMS内存是一种只需要极少电量就能存放数据的芯片。由于耗能极低，CMOS内存可以由集成到主板上的一个小电池供电，这种电池在计算机通电时还能自动充电。因为CMOS芯片可以持续获得电量，所以即使在关机后，也能保存有关计算机系统配置的重要数据。

还有一种叫FLASH闪存（Flash Memory），简称为“Flash”，它属于内存器件的一种，是一种不挥发性（Non-Volatile）内存。闪存在没有电流供应的条件下也能够长久地保持数据，其存储特性相当于硬盘，这项特性正是闪存得以成为各类便携型数字设备的存储介质的基础，已经成为单片机扩展存储器的首要选择。图8-5是各种存储器的实物外观。

存储器技术的成熟使得RAM和ROM之间的界限变得很模糊，如今有一些类型的存储器（如EEPROM和闪存器）组合了两者的特性。这些器件像RAM一样进行读写，并像ROM一样在断电时保持数据，都可电擦除且可编程，但各自有优缺点。从软件角度看，独立的EEPROM和闪存器件类似，主要差别是EEPROM器件可以逐字节地修改，而闪存器件只支持扇区擦除以及对被擦除单元的字、页或扇区进行编程。对闪存的重新编程还需要使用SRAM，因此它要求更长的时间内有更多的器件在工作，从而需要消耗更多的电池能量。存储器密度是决定选择串行EEPROM或者闪存的另一个因素。市场上目前可用的独立串行EEPROM器件的容量在128KB或以下，独立闪存器件的容量在32KB或以上。如果把多个器件级联在一起，可以用串行EEPROM实现高于128KB的容量。闪存器一般可擦除/写入1万

次，但FRAM的可读写次数非常高且写入速度较快。

闪存卡（Flash Card）就是利用闪存（Flash Memory）技术达到存储电子信息的存储器，一般应用在数码相机，掌上电脑，MP3等小型数码产品中作为存储介质，样子很小巧，犹如一张卡片，所以称之为闪存卡。根据不同的生产厂商和不同的应用，闪存卡有SmartMedia（SM卡）、Compact lash（CF卡）、MultiMediaCard（MMC卡）、Secure Digital（SD卡）、Memory Stick（记忆棒）、XD-Picture Card（XD卡）和微硬盘（MICRODRIVE）。这些闪存卡虽然外观、规格不同，但是技术原理都是相同的。



（a）　　（b）

图 8-5 各种存储器

（a）PC机 使用的存储器　（b）FLASH闪存器

8.2.2 AT24C02闪存器

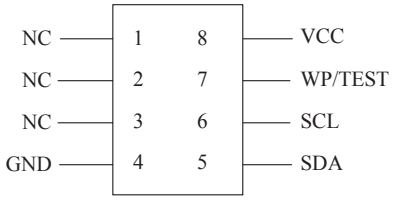


图 8-6 AT24C02闪存器引脚图

AT24C02闪存器的引脚排列如图8-6所示，引脚定义见表8-1。AT24C02是美国ATMEL公司的低功耗COMS串行EEPROM，它是内含256 × 8位存储空间，具有工作电压宽（2.5~5.5V）、擦写次数多（大于10 000次）、写入速度快（小于10ms）等特点。AT24C02是一种I^2C总线结构的闪存存储器，它通过SDA（串行数据线）及SCL（串行时钟线)两根线在连到总线上的器件之间传送信息。在I^2C总线上每次传送的数据字节数不限，但每一个字节必须为8位，而且每个传送的字节后面必须跟一个认可位（第9位），也叫应答位（ACK）。每次都是先传最高位，通常从器件在接收到每个字节后都会作出响应，即释放SCL线返回高电平，准备接收下一个数据字节，主器件可继续传送。

表 8-1 AT24C02 的引脚定义

NC——1脚	片选地址
NC——2脚	片选地址
NC——3脚	片选地址
GND——4脚	接地脚
SDA——5脚	串行数据输入/输出
SCL——6脚	串行时钟，用于对输入和输出数据的同步
WP/TEST——7脚	WP—写保护，该脚接地时，可以对整个存储器进行读、写。该脚接5V时，对存储器的数据只可以读，不能写。TEST—存储器检测
VCC——8脚	电源电压，5V

8.2.3 AT24C02闪存器在单片机中的应用

图8-7为AT24C02闪存器与单片机的连接。其中，AT24C02的1、2、3脚是三条地址线，用于确定芯片的硬件地址，它们都接地，第8脚和第4脚分别为正、负电源。第5脚SDA为串行数据输入/输出，数据通过这条双向I^2C总线串行传送，单片机的P2.1连接。第6脚SCL为串行时钟输入线，和单片机的P2.0连接。SDA、SCL需要通过上拉电阻来稳定端口的电平。

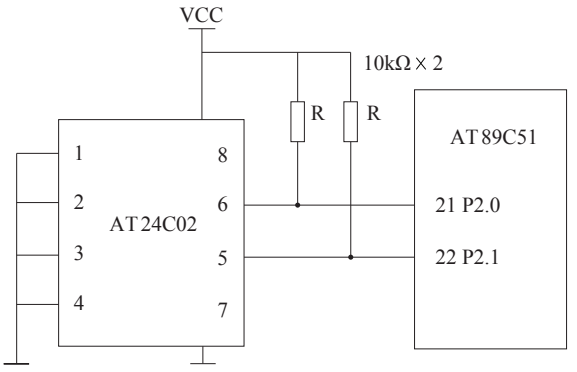


图 8-7 AT24C02闪存器与单片机的连接

8.3 A/D转换接口技术

学习目标

1. 熟悉A/D转换的概念。
2. 掌握A/D转换器的应用。

导　入

在平时，我们会接触到两种类型的数，一种是依靠人的主观来决定的数字，例如测量物体重量的机械式弹簧称，是看弹簧的伸长的位置或指针旋转的角度，重量的多少是由人自己读出的。另外一种是数字称，用它测量得到的重量由数字电子称直接显示出来。在前一种情况下，如果重量发生变化，例如从1公斤慢慢增加到2公斤，弹簧的长度变化或指针旋转的角度变化也是慢慢地连续变化的，依靠人的判断读出数字，这些数字也是连续变化的。但在后一种情况下，如果重量发生变化，数字称显示的数字是一个一个不连续的数字，它要么是1.12公斤，要么是1.13公斤。我们把连续发生变化的量，称为模拟量，而不是连续发生变化的量称数字量。

8.3.1 A/D转换器概述

随着数字技术的飞速发展与普及，在现代控制、通信及检测等领域，为了提高系统的性能指标，对信号的处理广泛采用了数字计算机技术。但系统的实际对象往往都是一些模

拟量（如温度、压力、位移、图像等），要使计算机或数字仪表能识别并处理这些信号，首先必须将这些模拟信号转换成数字信号，然后经计算机分析处理后输出的数字量也往往需要将其转换为相应模拟信号才能为执行机构所接受。这样，就需要一种能在模拟信号与数字信号之间起桥梁作用的电路——模数和数模转换器 。

将模拟信号转换成数字信号的电路，叫模数转换器，简称A/D转换器（Analog/Digital Converter）。A/D转换器已成为计算机应用中不可缺少的器件，在使用中为确保系统处理结果的精确度，A/D转换器必须具有足够的转换精度，有时候还需要具有较高的转换速度。转换精度与转换速度是衡量A/D转换器的重要技术指标。

A/D转换器的功能是把模拟量变换成数字量。由于实现这种转换的工作原理和采用的工艺技术不同，因此生产出种类繁多的A/D转换芯片。A/D转换器按分辨率分为4位、6位、8位、10位、14位、16位和bcd码的$3\frac{1}{2}$位，$5\frac{1}{2}$位等。按照转换速度可分为超高速（转换时间≤330ns），次超高速（330~3.3μs），高速（转换时间3.3~333μs），低速（转换时间＞330μs）等。A/D转换器按照转换原理可分为直接A/D转换器和间接A/D转换器。直接A/D转换器是把模拟信号直接转换成数字信号，如逐次逼近型、并联比较型等。其中逐次逼近型A/D转换器，易于用集成工艺实现，且能达到较高的分辨率和速度，故目前集成化A/D芯片采用逐次逼近型者多。间接A/D转换器是先把模拟量转换成中间量，然后再转换成数字量，如电压/时间转换型（积分型）、电压/频率转换型、电压/脉宽转换型等。其中积分型A/D转换器电路简单，抗干扰能力强，且分辨率高，但转换速度较慢。有些转换器还将多路开关、基准电压源、时钟电路、译码器和转换电路集成在一个芯片内，已超出了单纯A/D转换功能，但使用十分方便。

8.3.2 A/D转换器ADC0832

模数转换器ADC0832是美国国家半导体公司生产的一种8位分辨率、双通道A/D转换芯片。它体积小，兼容性强，性价比高。具有转换快、稳定性好、与微处理器接口简捷、价格低等优点，

1. ADC0832的引脚及功能

ADC0832是8位开关电容逐次逼近型模数转换器，有8脚、20脚两个封装形式，其8脚封装见图8-8，引脚定义见表8-2。

表 8-2 ADC0832引脚定义

引脚名称	引脚功能
$\overline{CS}$	片选端，低电平有效
CH0	模拟量输入通道0
CH1	模拟量输入通道1
GND	接地端
DI	数据信号输入，选择通道控制
DO	数据信号输出，转换数据输出
CLK	芯片时钟输入
VCC	正电源端，一般接+5V电源

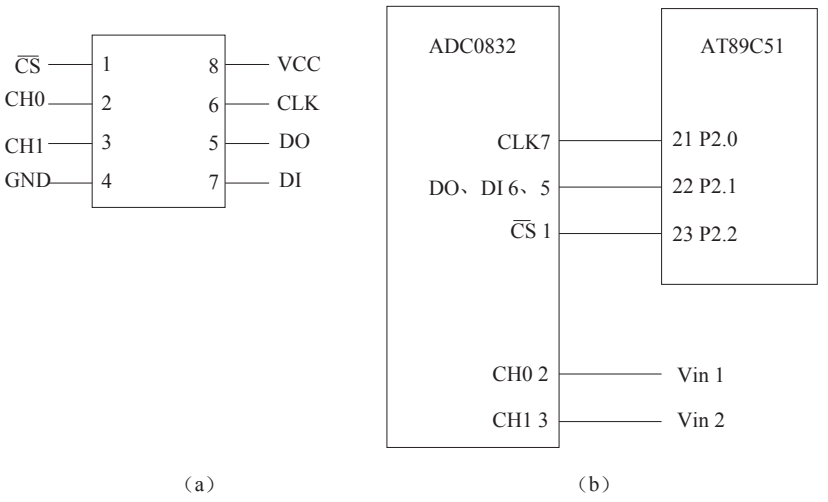


图 8-8 ADC0832管脚排列及与单片机的连接

（a）ADC0832管脚排列 （b）ADC0832与单片机的连接

2. ADC0832的编程要点

ADC0832为8位分辨率A/D转换芯片，其最高分辨可达256级，可以适应一般的模拟量转换要求。其内部电源输入与参考电压的复用使得芯片的模拟电压输入在0~5V。芯片转换时间仅为32μs，有双数据输出可作为数据校验，以减少数据误差，转换速度快且稳定性能强。独立的芯片使处理器控制与多个器件的连接十分方便，通过DI数据输入，可以实现通道功能的选择。

通常情况下，单片机与ADC0832的接口为4根数据线，分别是CS、CLK、DO、DI。但由于DO端与DI端在通信时并未同时有效且与单片机的接口是双向的，所以电路设计时可以将DO和DI并联在一根数据线上。

当ADC0832未工作时其CS输入端应为高电平，CLK和DO/DI的电平可任意。当要进行A/D转换时，须先将CS端置低电平且保持低电平直到转换完全结束。芯片开始转换工作的同时由单片机向芯片时钟输入端CLK输入时钟脉冲，DO/DI端则使用DI端输入通道功能选择的数据信号。

在第1个时钟脉冲下降之前DI端必须是高电平，表示启始信号。在第2、3个脉冲下降之前DI端应输入2数据用于选择通道功能，其功能项见表8-3。

表 8-3 功能项选择

输入形式	电平信号		选择通道	
	CH0	CH1	CH0	CH1
差分	0	0	+	−
	0	1	−	+
单端	1	0	+	
	1	1		+

注：选择通道中的“+”表示正极性，“－”表示负极性。

在表8-3中，电平信号为“1”“0”时，只对CH0通道转换；电平信号为“1”“1”时，只对CH1通道转换，为单端形式。电平信号为“0”“0”时，CH0为正输入端，CH1

为负输入端；电平信号为“0”“1”时，CH0为负输入端，CH1为正输入端，其输入形式为差分。在第3个脉冲下降沿后DI端的输入电平就失去输入作用，此后DO/DI端则开始利用数据输出 DO 进行转换数据的读取。从第4个脉冲下降沿开始由DO端输出转换数据最高位DATA7，随后每一个脉冲下降沿DO端输出下一位数据。直到第11脉冲时发出最低位数据DATA0，一个字节的数据输出完成。也正是从此位开始输出下一个相反字节的数据，即从第11个字节的下降沿输出DATD0。随后输出8位数据，到第19个脉冲时数据输出完成，也标志着一次A/D转换的结束。最后将/CS置高电平禁用芯片，直接将转换后的数据进行处理就可以了。具体的时序见图8-9。

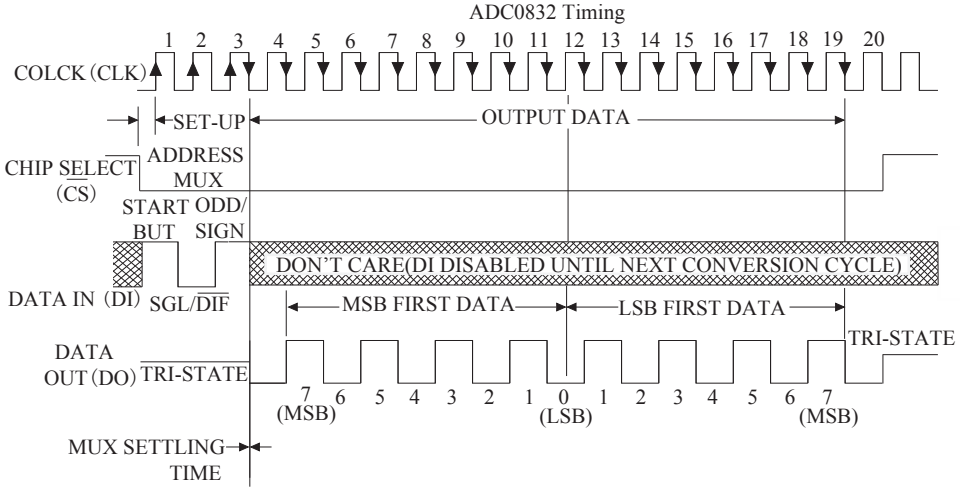


图 8-9 ADC0832的时序图

8.4 D/A转换接口技术

学习目标

1. 熟悉D/A转换的概念。
2. 掌握D/A转换器的应用。

导　入

对于单片机的CPU其运行是由0、1组成的数字量，那么如何与我们接触到的两种类型数联系起来呢？前面一节介绍了把外界的模拟量通过A/D转换器件转换成数字量传送给CPU，同理也必需要有一个转换电路需要把CPU的数字量转换成模拟量--这就是D/A转换器件。

8.4.1 D/A转换器概述

经过计算机分析、处理后只能形成数字量，由于大部分外部执行机构只接受模拟量的信号，所以必须把计算机输出的数字量信号转换成相应的模拟信号才能为执行机构所接受。这样，就需要一种能在数字信号与模拟信号之间起桥梁作用的电路-数模转换器。

将数字信号转换成模拟信号的电路叫数模转换器，简称D/A转换器。D/A转换器已成为计算机应用中不可缺少的器件，在使用中为确保系统处理结果的精确度，D/A转换器必须具有足够的转换精度，有时候还需要具有较高的转换速度。转换精度与转换速度是衡量D/A转换器的重要技术指标。

衡量D/A转换器的性能的主要参数有：

① 分辨率：指D/A转换器能够转换的二进制数的位数，位数越多分辨率也就越高。

② 转换时间：指数字量从输入到完成转换、输出最终值并达到稳定为止所需的时间。电流型D/A转换较快，一般在几ns到几百ns之间。电压型D/A转换较慢，取决于运算放大器的响应时间。

③ 精度：指D/A转换器实际输出电压与理论值之间存在误差，一般采用数字量的最低有效位作为衡量单位。

④ 线性度：当数字量变化时，D/A转换器输出的模拟量按比例关系变化的程度。理想的D/A转换器是线性的，但是实际上是有误差的，模拟输出偏离理想输出的最大值称为线性误差。

8.4.2 D/A转换器TLC5615

1. TLC5615的引脚及功能

TLC5615为美国德州仪器公司1999年推出的产品，是具有串行接口的数模转换器，其输出为电压型，最大输出电压是基准电压值的两倍，带有上电复位功能，即把DAC寄存器复位至全零，性能比早期电流型输出的DAC要好，只需要通过三根串行总线就可以完成10位数据的串行输入，易于和工业标准的微处理器或微控制器（单片机）接口，适用于电池供电的测试仪表、移动电话，也适用于数字失调与增益调整以及工业控制场合。图8-10是TLC5615管脚排列及与单片机的连接图。表8-4是TLC5615引脚说明。

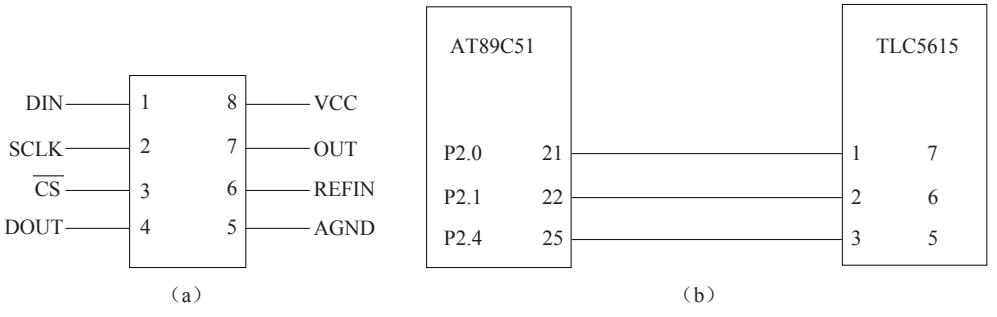


图 8-10 TLC5615管脚排列及与单片机的连接

（a） TLC5615管脚排列 （b） TLC5615与单片机的连接

表 8-4 TLC5615引脚定义

引脚名称	引脚功能
DIN	串行数据输入端
SCLK	串行时钟输入端
$\overline{CS}$	芯片选用通端，低电平有效
DOUT	用于级联时的串行数据输出端
AGND	模拟地
REFIN	基准电压输入端，2V~(VDD-2)
OUT	模拟电压输出端
VDD	正电源

2. TLC5615的内部功能框图

TLC5615的内部功能框图，如图8-11。它主要由以下几部分组成：

- 10位DAC电路
- 一个16位移位寄存器接受串行移入的二进制数，并且有一个级联的数据输出端DOUT
- 并行输入输出的10位DAC寄存器，为10位DAC电路提供待转换的二进制数据
- 电压跟随器为参考电压端REFIN提供很高的输入阻抗，大约10MΩ
- 电路提供最大值为2倍于REFIN的输出
- 上电复位电路和控制电路

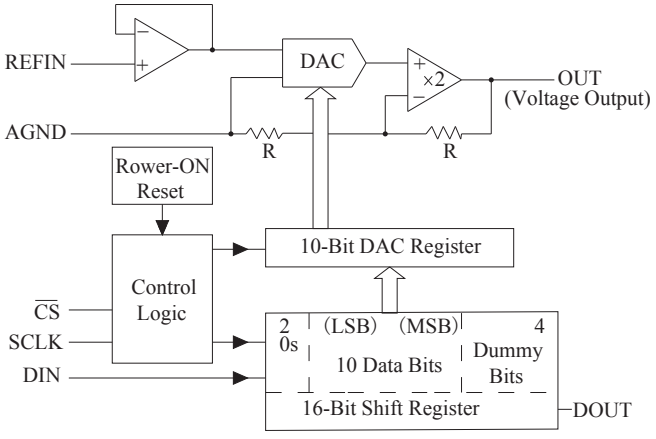


图 8-11 TLC5615功能框图

从图8-11可以看出，16位移位寄存器分为高4位虚拟位、低2位填充位以及10位有效位。在单片机TLC5615工作时，只需要向16位移位寄存器按先后输入10位有效位和低2填充位，2位填充位数据任意，这是第一种方式，即12位数据序列。另外，16位数据列可以将本片的DOU T接到下一片的DIN，需要向16位移位寄存器按先后输入高4位虚拟位、10位有效位和低2位填充位，由于增加了高4位虚拟位，所以需要16个时钟脉冲。

3. TLC5615 工作时序

TLC5615工作时序如图8-12所示。可以看出，只有当片选CS为低电平时，串行输入数据才能被移入16位移位寄存器。当CS为低电平时，在每一个SCLK时钟的上升沿将DIN的一位数据移入16位移寄存器。注意二进制最高有效位被导前移入，接着CS的上升沿将16位移

位寄存器的10位有效数据锁存于10位DAC寄存器，供DAC电路进行转换，当片选CS为高电平时，串行输入数据不能被移入16位移位寄存器。注意CS的上升和下降都必须发生在SCLK为低电平期间。

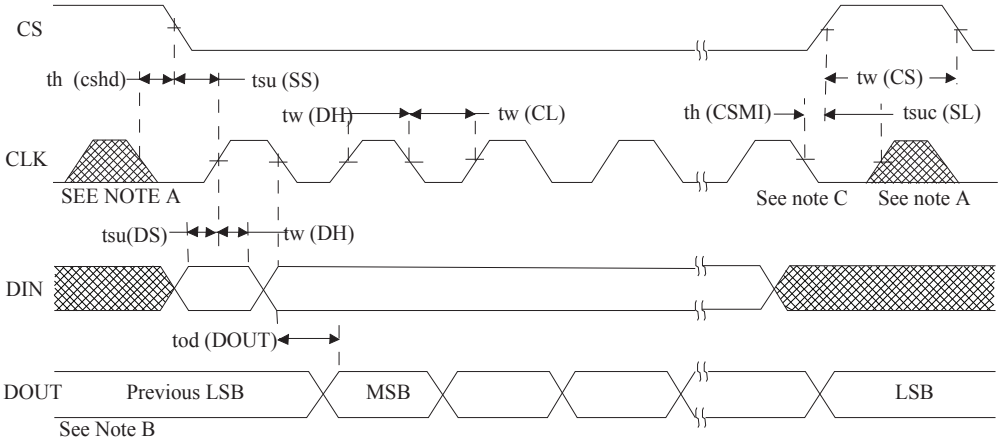


图 8-12 TLC5615工作时序图

知识拓展

1. 可以把继电器、电动机、加热器等设备接到74LS377的输出端进行控制，也可以把灯的开关状态，电机的开关状态或者是某物体的移动的开关量输入到74LS244的输入端上，如图8-13所示。

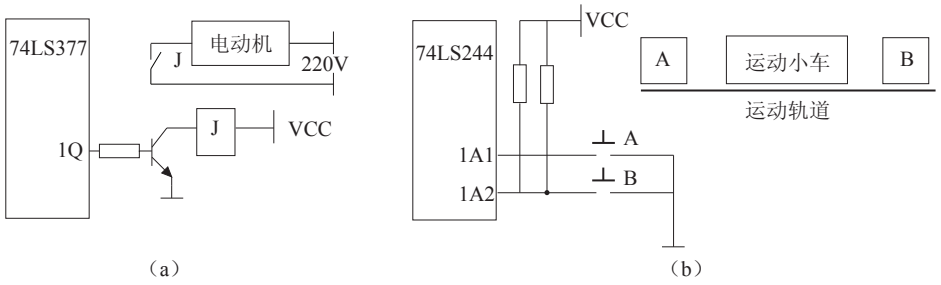


图 8-13　74LS377、74LS244的扩展应用

（a）74LS377的电动机驱动控制　（b）74LS244的运动小车位置检测

在图8-13（a）中，单片机通过扩展接口74LS377输出一个“1”电平，使三极管饱和导通，继电器J得电，继电器的触点J闭合，电动机得到220V电源开始运转，实现了单片机控制电动机的目的。在这里，可以把电动机换成加热器或其他用电设备。

在图8-13（b）中，有一个运动小车，它的两端有A、B两个限位开关，把这两个限位开关接到74LS244的输入端，通过对位置开关A、B的“0”、“1”检测，就可以知道运动小车的运动状态，以决定对运动小车的控制。同样，可以把这两个限位开关安装在电梯门上，可以检测电梯门的开与关，还可以应用在更多的其他位置状态的检测。

2. AT24C系列还有其他更多型号的芯片：通过查阅资料了解AT24C01、AT24C01A、AT24C02、AT24C04、AT24C08、AT24C16、AT24C164、AT24C32、AT24C64、AT24C256、AT24C512等芯片的性能，与单片机的连接方式、存储容量、数据的写入与读出的方式等。

3. A/D转换器的早期产品有AD0809等并行口的产品，有8位和10位的。随着技术的进步，目前比较多的产品是串行口的A/D转换器。除了ADC0832，还有12位串口的MAX187、16位串口的MAX195。

查阅MAX187、MAX195的有关位数、转换时间、相关的引脚功能等技术参数。讨论与单片机的硬件连接方法。完成测量0～100℃的温度方案。

4. D/A转换器的早期产品有DA0832等并行口的产品，目前比较多的产品是串行口D/A转换器。除了串行口TLC5615的8位D/A转换器，还有串行口8位MAX521，12位的MAX5154等各种型号的芯片。查阅MAX521相关的技术参数，讨论与单片机连接方法。

本章小结

本章主要介绍了单片机应用系统的接口技术。主要是常用的单片机数字量扩展芯片，74LS244是数字量输入扩展芯片，一次可以扩展八个数字量输入口，可以输入各种开关量。74LS377是数字量输出扩展芯片，一次可以扩展八个数字量输出口，可以驱动各种开关量控制的设备。具有相同的输入、输出端口，应用不同的程序，可以实现不同的应用。单片机本身的RAM、ROM存储器的存储容量有限，在很多情况下需要增加RAM和ROM的容量。Flash存储器可以在线方便地把信息存储进去，主要是存放需要反复使用的常数或者需要在断电仍然能够保存的数据，特别是具有串口通信功能的Flash存储器，由于与单片机连接简单，占用单片机的端口少，目前被广泛使用。AT2402等Flash存储器存储容量不同，但都是八个引脚的器件，只要掌握了基本的AT2402的使用方法，就可以方便地使用其他型号的器件了。

MCS-51单片机只有数字量输入的端口，如果要测量外部的模拟量值，必须把模拟量通过A/D转换器，把模拟量转换成数字量才能输入单片机。A/D转换器的性能有分辨率和转换速度等几个方面。一般有4位、8位、10位等，位数越高分辨率越高。A/D转换器有并行口和串行口之分。串行口的A/D转换器与单片机连接使用较少的单片机端口，使单片机应用系统的硬件电路更简单。

D/A转换器的性能有分辨率和转换速度等几个方面。一般有8位、10、12位等，位数越高分辨率越高，对外部设备的控制越精细。D/A转换器有并行口和串行口之分。串行口的D/A转换器与单片机连接使用较少的单片机端口，使单片机应用系统的硬件电路更简单。

思考与练习题

问答题

1. 74LS377与74LS244的主要功能是什么？
2. AT2402闪存器总线是I^2C，I^2C总线的显著特点是什么？
3. 串行总线形式与并行总线的技术特点是什么？
4. A/D与D/A转换器的主要功能与应用场合是什么？
5. 若有一个12位D/A转换器满量程输出电压是5V，则其输出电压的分辨率是多少？

实验项目8　接口与转换电路调试

一、实验目的

1. 掌握74LS377扩展接口的应用。
2. 掌握74LS244扩展接口的应用。
3. 掌握AT24C02的数据读写的应用。
4. 掌握ADC0832数据采集的应用。
5. 掌握TLC5615转换输出的应用。

二、实验设备

1. PC计算机一台
2. Keil μVision2集成开发环境
3. 单片机实验装置一套（教材配套）
4. STC_ISP_V480在系统编程软件

三、实验内容

1. 阅读实验源程序，概述程序工作过程。
2. 调试、验证实验现象。
3. 通过实验进一步理解器件的工作原理。

四、实验程序

1. 74LS377扩展接口的应用

依据图8-3，实现使LED灯亮或暗的程序C51源程序如下：

```
#include <reg52.h>
#define uint unsigned int
#define uchar unsigned char

sbit ccs=P2^7;
sbit wr=P3^6;

uchar code light[8]={0x80,0x20,0x08,0x02,0x01,0x04,0x10,0x40} ; //依次是1,3,5,7,8,6,4,2的顺序

void delay_1ms(uint t)
{
   uint i;
   uchar j;
```

```
    for(i=0;i<t;i++)
    {
      for(j=0;j<200;j++);
      for(j=0;j<102;j++);
    }
}

void main()
   {
      uint i;
      wr=0;                          //不允许写入
      ccs=0;                         //片选，芯片开始正常工作
      while(1)
      {
          for(i=0;i<8;i++) //8次
          {
              P0=~light[i];          //依次显示1,3,5,7,8,6,4,2
              wr=1;                  //允许写入，P0接收到数据
              wr=0;                  //不允许写入
                delay_1ms(2000);     //延时2秒
           }
      }
          wr=0;                      //不允许写入
          ccs=1;                     //允许控制端关闭，芯片不工作
}
```

2. 74LS244扩展接口的应用

见图8-5，编写程序，使某个开关合上，则74LS244集成电路中对应的LED灯亮，参考程序如下：

```
#include <reg52.h>
#define uint unsigned int
#define uchar unsigned char

sbit q1=P2^5;
sbit q2=P2^6;
sbit ccs=P2^7;
sbit wr=P3^6;

uchar code light[8]={0x80,0x20,0x08,0x02,0x01,0x04,0x10,0x40} ;
```

```
void delay_1ms(uint t)
{
    uint i;
    uchar j;
    for(i=0;i<t;i++)
     {
       for(j=0;j<200;j++);
       for(j=0;j<102;j++);
      }
    }
    void main()
    {
       uint i;

       wr=0;            //不允许写入
       ccs=0;           //片选，芯片正常工作
       while(1)
       {
         q1=0;          //允许1Y1-1Y4输出
         q2=0;          //允许2Y1-2Y4输出
         P0=P0;         //P0口得到对应8个开关的二进制数
         wr=1;          //允许写入，P0口读取数据
         wr=0;          //不允许写入
         delay_1ms(1);
         q1=1;          //不允许1Y1-1Y4输出
         q2=1;          //不允许2Y1-2Y4输出
       }
         wr=0;          //不允许写入
         ccs=1;         //芯片停止正常工作
    }
```

阅读上述参考程序，叙述程序工作过程，并适当修改程序，观察运行效果。

3. AT24C02的数据读写操作

在单片机内部RAM中从20H单元开始，存储00~19共20个数据，将这20个数据写入AT24C02存储器中，再依次读出它，并在LED数码管上依次显示，每次显示保持1s的时间。该任务的C51参考程序如下：

```
#include<reg52.h>
#include<intrins.h>
#define uchar unsigned char
#define uint unsigned int
sbit SDA=P2^1;
```

```
sbit SCL=P2^2;
sbit QA=P1^0;
sbit QB=P1^1;
sbit QC=P1^2;
sbit QD=P1^3;
sbit EL1=P1^4;
sbit EL2=P1^5;
sbit EL3=P1^6;

    void delay_1ms(uint t)                    //延时1*t毫秒
    {
    uint i;
    uchar j ;
    for(i=0;i<t;i++)
    {
       for(j=0;j<200;j++);
       for(j=0;j<102;j++);
    }
    }

    void delay()
    {
       _nop_();
       _nop_();
       _nop_();
       _nop_();
       _nop_();
       _nop_();
       _nop_();
       _nop_();
    }

    void sendbyte(uint a)                     //十进制转二进制送译码器
    {
    int b,c,d,e;
    b=a%2;
    c=((a-b)/2)%2;
    d=((((a-b)/2)-c)/2)%2;
    e=((((((a-b)/2)-c)/2)-d)/2)%2;
    QA=b;
```

```
QB=c;
QC=d;
QD=e;
}

void display(uint dat)                    //控制译码器显示相应数字
{
EL1=1;
EL2=1;
EL3=1;
sendbyte(dat/100);
EL1=0;
EL1=1;
delay();
sendbyte(dat%100/10);
EL2=0;
EL2=1;
delay();
sendbyte(dat%10);
EL3=0;
EL3=1;
delay();
}

void start() //启动I²C总线
{
SDA=1;
delay();
SCL=1;
delay();
SDA=0;
delay();
}

void stop() //停止I²C总线
{
SDA=0;
delay();
SCL=1;
delay();
```

```
SDA=1;
delay();
}

void ack()                          //I²C总线时钟
{
uchar i;
SCL=1;
delay();
while((SDA==1)&&(i<200))i++;
SCL=0;
delay();
}

void  noack()
{
SDA=1;
delay();
SCL=1;
delay();
SCL=0;
delay();
}

void init()                         //初始化
{
SCL=1;
SDA=1;
}

void iicwr_byte(uchar dat)          //从地址dat写入一个字节数据
{
uchar i;
SCL=0;
        for(i=0;i<8;i++)
        {
             if(dat&0x80)
             {
              SDA=1;
             }
```

```
            else
            {
              SDA=0;
            }
dat=dat<<1;
        delay();
        SCL=1;
        delay();
        SCL=0;
        delay();
        }
SDA=0;
delay();
}

uchar iicre_byte()    //读取一个字节数据
{
uchar i;
uchar dat;
SCL=0;
delay();
SDA=1;
delay();
for(i=0;i<8;i++)
{
SCL=1;
delay();
dat=dat<<1;
if(SDA)
{
dat++;
}
SCL=0;
delay();
}
return dat;
}

void delay1()
{
```

```
uint a;
a=30000;
while(a--);
}

void main()
{
uint num,i,j;
num=0x00;
for(i=0;i<20;i++)                    //存20个字节数据
{
init();
start();
iicwr_byte(0x20);                    //选择20H单元
ack();
iicwr_byte(j);
ack();
iicwr_byte(num);                     //在地址0x20 ROM j 写入一个字节数据
ack();
stop();
delay1();
num=num+0x01;
j++;
}
j=0;
for(i=0;i<20;i++)                    //读取20个字节数据
{
init();
start();
iicwr_byte(0x20);                    //地址选择
ack();
iicwr_byte(j);                       //rom选择
ack();
start();
iicwr_byte(0x21);
ack();
display(iicre_byte()+30);            //读取的数据加三十并显示
noack();
stop();
j++;
```

```
delay_1ms(1000);                    //延时1s
}
}
```

注：本实验内容涉及I^2C原理，对于工作原理的理解需结合具体情况而定。

4. ADC0832数据采集

编写ADC0832的CH0端的数据采样C51程序，在调节电位器的时候，LED数码管显示相应的值。C51参考源程序如下：

```
#include <reg52.h>
#define uint unsigned int           //定义uint为unsigned int
#define uchar unsigned char         //定义uchar为unsigned char

sbit QA=P1^0;
sbit QB=P1^1;
sbit QC=P1^2;
sbit QD=P1^3;
sbit EL1=P1^4;
sbit EL2=P1^5;
sbit EL3=P1^6;
sbit dout=P2^0;
sbit din=P2^1;
sbit clk=P2^3;
sbit cs=P2^4;

void delay_1ms(uint t)              //延时函数
{
uint i;
uchar j;
for(i=0;i<t;i++)
{
for(j=0;j<200;j++);
for(j=0;j<102;j++);
}
}

uint adc0832()                      //AD转换函数
{
uchar i;
uint dat;
cs=0;                               //片选
```

```
clk=1;
din=1;
clk=0;                          //第一次脉冲下降沿
delay_1ms(1);
clk=1;
din=1;
clk=0;                          //第二次脉冲下降沿
delay_1ms(1);
clk=1;
din=0;                          //打开输入
dout=0;                         //打开输出
clk=0;                          //第三次脉冲下降沿
delay_1ms(1);
 for(i=0;i<8;i++)
 {
   clk=1;
   clk=0;                       //时钟脉冲下降沿
   dat=dat<<1|dout;             //由 DO端输出转换数据最高位，随后每一个脉冲下沉 DO
                                  端输出下一位数据。
   delay_1ms(1);
  }
cs=1;                           //禁止芯片工作
return dat;                     //返回转换得到的数据
}

void sendbyte(long a)
{
char b,c,d,e;
b=a%2;
c=((a-b)/2)%2;
d=((((a-b)/2)-c)/2)%2;
e=((((((a-b)/2)-c)/2)-d)/2)%2;
QA=b;
QB=c;
QC=d;
QD=e;
}

void display(long dat)          //0~999循环显示
{
```

```
EL1=1;。
EL2=1;
EL3=1;
sendbyte(dat/100);                    //显示百位
EL1=0;
EL1=1;
sendbyte(dat%100/10);                 //显示十位
EL2=0;
EL2=1;
sendbyte(dat%10);                     //显示个位
EL3=0;
EL3=1;
delay_1ms(1);
}

void main()
{
while(1)
{
display(adc0832());                   //显示转换得到的数据
delay_1ms(100);
}
}
```

5. TLC5615转换输出

设TLC5615的参考电压为4V（第6脚），在CPU的第12脚（P3.2）输入一个脉冲信号，TLC5615的输出端（第7脚）的输出电压就增加4000mV/1024的电压值。在输入脉冲信号的同时，测量TLC5615的电压输出值，分析程序的功能和D/A转换的原理。C51语言参考程序如下：

```
#include<reg52.h>
sbit CS=P2^2;                         //dac5615的片选端3脚
sbit SCLK=P2^1;                       //ac5615的片选端2脚
sbit DIN=P2^0;                        //ac5615的片选端1脚
unsigned int ddata=0x01f;             //设置一个整数
char plus;                            //脉冲信号

void DA(unsigned int x);

main()
{
```

```
  EX0=1;
  IT0=1;
  EX1=1;
  IT1=1;
  EA=1;
  SCLK=0;
  DA(ddata);                    //调用DA转换子程序
for( ;  ; )
{
   PCON=1;
  }
}

void DA(  unsigned int x  )   //DA转换子程序
{
  char i;
  x=x<<2;
  CS=0;
for(i=12;i!=0;i--)
{
    DIN=(x>>(i-1))&0x0001;
    SCLK=1;
    SCLK=0;
  }
  CS=1;
}

void EXIT0( ) interrupt 0    //外中断1，中断服务程序，增加电压4 000mV/1024的电压值
{
    ddata=ddata+plus;
    if(ddata>0x3ff)
{
ddata=0;
}
    DA(ddata);
}

    void EXIT1()interrupt 2 //外中断2，中断服务程序，增加脉冲
{
    char i;
```

```
    if(plus==1)
{
   plus=5;
}
else
{
     plus=1;
   }
   for(i=255;i!=0;i--);
}
```

第9章 单片机综合实训

目前，各行各业都有单片机的应用踪迹，导弹的导航装置，飞机上各种仪表的控制，计算机的网络通讯与数据传输，工业自动化过程的实时控制和数据处理，广泛使用的各种智能IC卡，民用豪华轿车的安全保障系统，录像机、摄像机、全自动洗衣机的控制以及程控玩具、电子宠物等等，这些都离不开单片机。更不用说自动控制领域的机器人、智能仪表、医疗器械以及各种智能机械了。

图9-1是单片机的综合应用例子，图9-1（a）是医疗行业使用的试管旋转混合仪，该仪器可以根据需要设置旋转的方向、频率和幅度。图9-1（b）是工业领域使用的自整定PID控制仪表，可以应用于需要进行温度、湿度、流量、速度的控制。图9-1（c）是智能电导率变送器，能够在线测量液体的电导率，并能自动进行温度补偿。

（a）

（b）

（c）

图 9-1 单片机的应用

（a）试管旋转混合仪 （b）自整定PID控制仪表 （c）工业电导率变送器

学习了单片机的硬件和软件知识，再结合其他方面的知识，就可以自己设计各种单片机的应用项目了，下面介绍四个单片机综合应用的实训项目。

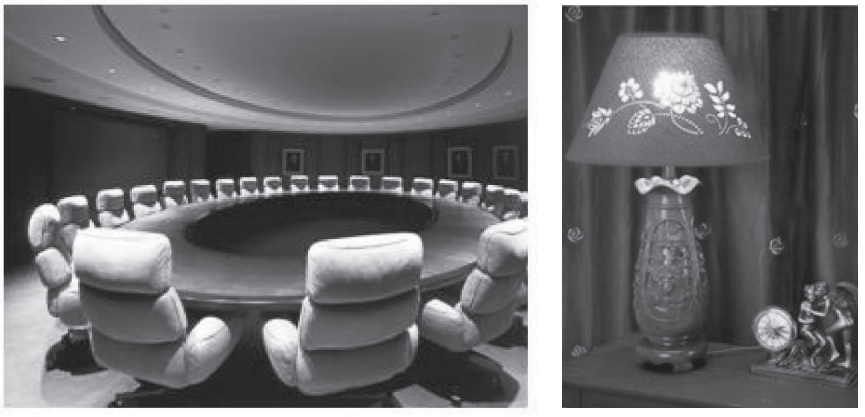
9.1 自动调光灯的设计

学习目标

1. 熟悉应用单片机测量光的强度和控制技术。
2. 了解光敏电阻的特性和应用。
3. 掌握光线强度的测量方法。
4. 掌握灯光的亮度控制。
5. 学会应用单片机实现模拟量输入、输出技术。

在我们的周围环境中，调光技术广泛应用于各种场合，如图9-2所示。一般电影院、剧场等娱乐场所为了营造一些氛围，常常需要调节灯光亮度和色彩。在某些会议室为了放映电子文档或图片也需要进行灯光调节。人们为了保护视力设计制造了各种各样的落地调光灯、调光台灯、调光壁灯等。

在许多家庭，都有一盏调光台灯，当自然光不足的时候，调节调光台灯，让台灯发出更亮的光线以弥补自然光线的不足。目前大部分调光台灯都是手动调节，如果应用单片机技术对灯光亮度进行数字显示，并根据需要自动调节灯的亮度，实现灯光亮度的设置和自动调光。那么，它的工作原理是什么？要实现光线的自动控制需要些什么元器件，如何进行控制呢？



（a）　　（b）

图 9-2 调光灯的应用

（a）会议室的调光应用　（b）调光台灯

任务实施

一、实训器材

PC机、单片机综合应用实验板、光敏电阻、调光灯、数字万用表、频率表、220V/5V电源、220V/30W电烙铁、松香、焊锡丝、导线、螺丝刀、220V电源插座等。

二、知识准备

1. 自动调光灯的工作原理

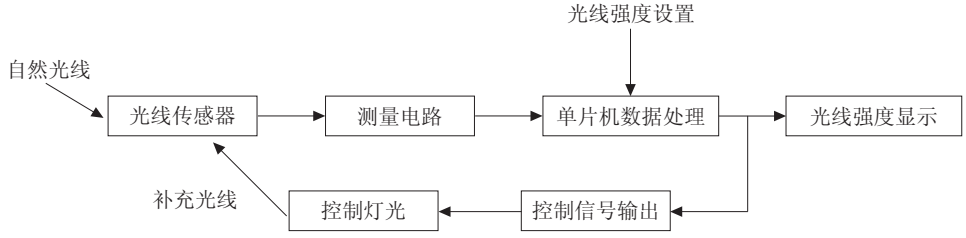


图 9-3　自动调光灯原理框图

如图9-3所示，自然光线照射到光线传感器，光线传感器通过测量电路产生的光线强度信号传送到单片机，由单片机进行运算，得到光线强度的数字信号由显示器显示。同时，单片机根据光线强度设置值和测量值进行比较。如果测量值比光线强度设置值小，说明自然光线比较弱，单片机输出一个控制信号，使被控制的灯发出更强的光。如果测量值比光线强度的设置值大，说明自然光线比较强，单片机输出一个控制信号，使被控制的灯发出比较弱的光。

2. 光线传感器

图9-4所示的是光敏电阻的外形图。在这里使用光敏电阻作为光线传感器。光敏电阻器是利用半导体光电效应制成的一种电阻值随入射光的强弱而改变的电阻器。入射光强，电阻减小，入射光弱，电阻增大。光敏电阻器一般用于可见光的测量、控制和光电转换。在黑暗条件下，它的阻值（暗阻）可达1~10MΩ，在强光条件（100LX）下，它阻值（亮阻）仅有几百至数千欧姆。光敏电阻器对光的敏感性（即光谱特性）与人眼对可见光0.4~0.76μm的响应很接近，只要人眼可感受的光，都会引起它的阻值变化。

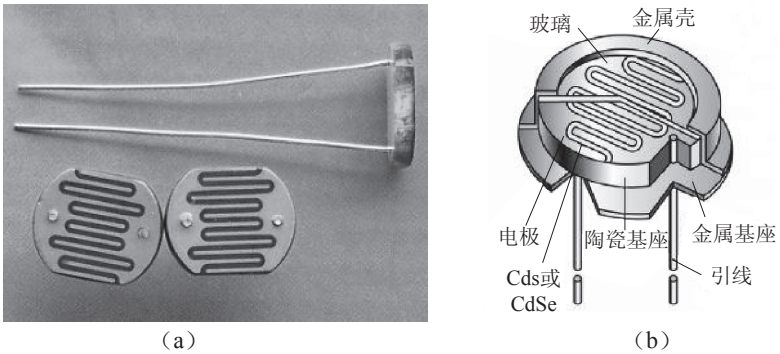


图 9-4　光敏电阻

（a） 光敏电阻外观　（b） 光敏电阻内部结构

光敏电阻的工作原理是基于内光电效应。在半导体光敏材料两端装上电极引线，将其封装在带有透明窗的管壳里就构成光敏电阻，为了增加灵敏度，两电极常做成梳状，如图9-4所示。用于制造光敏电阻的材料主要是金属硫化物、硒化物和碲化物等半导体。

3. 光线测量电路

光线的具体测量如图9-5所示。5V电源通过电阻R24、稳压管T3（LM385-25）产生一个2.5V的稳定电压作为光敏电阻测量电路的电源。光的电阻RG与电阻R23组成光线测量电路。

光线的测量信号：V0 = 2.5 * RG / (RG + R23)　　（9-1）

如果取光敏电阻的暗电阻RG1=2MΩ，亮电阻RG2=600Ω，R23=510Ω

那么：光线暗条件下，V0= 2.5*2MΩ/(2MΩ+510Ω)=2.5V

光线亮条件下，V0= 2.5*600Ω/(600Ω+510Ω)=1.351V

光线的测量信号V0通过LM358运算放大器的电压跟随电路，得到Vi输入到LM331 V/F转换电路，把输入电压Vi转换成频率信号F，转换关系参考图9-5。

F= (R16+W1) * Vi / (2.09 * R17 * R15 * C7)

当R16+W1=14.212kΩ R17=100kΩ R15=6.8kΩ C7=103时，转换关系为：1000Hz/1V。

光线暗条件下，V01= 2.5V，转换为F1=2500Hz；

光线亮条件下，V01= 1.351V，转换为F2=1351Hz；ΔF=F1－F2=2500－1351=1149Hz。

这样，我们就把光线从全暗到全亮，分成1149个等级，这是一个对光线变化分辨率非常高的测量。

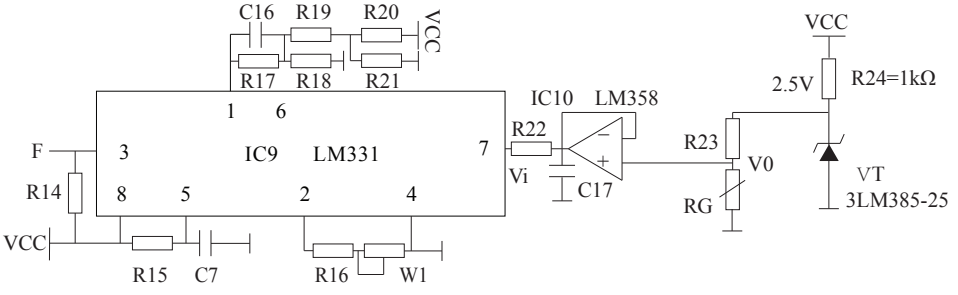
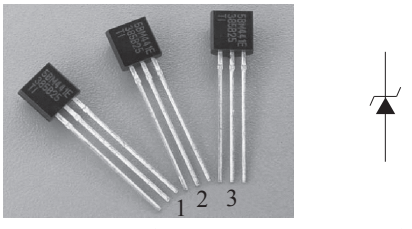


图 9-5 光线的测量

图中VT为精密稳压管，型号是LM385-25，稳定电压为2.5V。LM385-25通过5V、R24=1kΩ，在2脚得到精密稳定2.5V，工作电流=(5－2.5)/1 = 2.5mA ，它的外观图与符号见图9-6。



（a）　　（b）

图 9-6 稳压管LM385-25外形

（a）稳压管LM385-25外形和引脚排列　（b）元件符号

LM358内部有两个独立的、高增益、内部频率补偿的双运算放大器，适合于电源电压范围很宽的单电源使用，也适用于双电源工作模式，在推荐的工作条件下，电源电流与电源电压无关。它的使用范围包括传感放大器、直流增益模组，音频放大器、工业控制、DC增益部件和其他所有可用单电源供电的使用运算放大器的场合。LM358的封装形式有塑封8引线双列直插式和贴片式，管脚排列如图9-7所示。

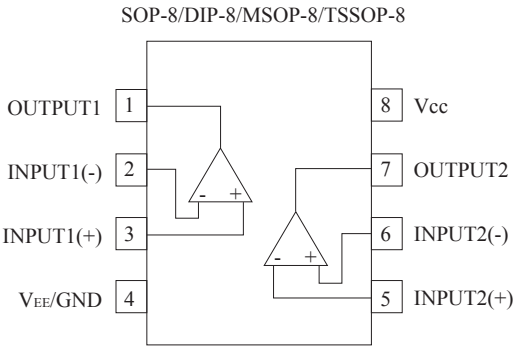


图 9-7 LM358管脚排列

LM331是美国NS公司生产的性价比高、外围电路简单、可单电源供电、低功耗的精密电压/频率转换器集成电路。LM331动态范围宽达100dB，工作频率低至0.1Hz时仍有较好的线性度，数字分辨率达12位。LM331的输出驱动器采用集电极开路形式，因此可通过选择逻辑电流和外接电阻来灵活改变输出脉冲的逻辑电平，以适配TTL、DTL和CMOS等不同逻辑电路。LM331可工作在4.0V~40V，输出可高达40V，而且可以防止VCC短路。外接电路简单，只需接入几个外部元件就可方便构成V/F或F/V等变换电路，并且容易保证转换精度。关于LM331引脚功能见表9-1，元件管脚排列见图9-8。

表 9-1 LM331引脚功能

引脚号	引脚名称	引脚功能
1	Current Output	电流输出
2	Ref Current	基准电流
3	Frequency Output	频率输出
4	GND	接地
5	R/C	接RC电路
6	Thresholod	阈值
7	Comparator Input	比较输入
8	VS	电源

4. 单片机数据处理

光线测量采用V/F方法，单片机AT89C52的14脚作为外部脉冲的输入端，记录LM331的频率值。把光线全暗到全亮平均分为100个亮度等级，用LD表示。当测量到的频率F=2500Hz，表示为亮度LD=0，测量到的频率F=1351Hz，表示为亮度LD=100，那么其余的亮度等级LD= (2500－F)*100/1149

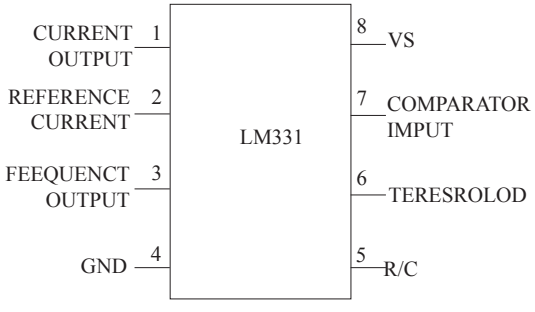


图 9-8 LM331管脚排列

例如：　F=2000Hz　则 LD= (2500－2000)*100/1149 = 43.5

F=1500Hz　则 LD= (2500－1500)*100/1149 = 87.0

F=1351Hz　则 LD= (2500－1351)*100/1149 = 100

单片机把计算得到的亮度值，通过三位LED数码管进行显示。

假设光线强度的设置值用SZ表示，它的值为0~100的整数，那么单片机将把测量到的光线强度LD与设置值SZ进行比较。

如果SZ－LD > 0，表示光线太弱，单片机通过LTC5615 D/A转换器，增加它的输出电压，使控制灯光增加亮度。

如果SZ－LD < 0，表示光线太强，单片机通过LTC5615 D/A转换器，减少它的输出电压，使控制灯光降低亮度。

如果SZ－LD = 0，表示光线正好，单片机保持LTC5615 D/A转换器的输出电压。这样就可以实现自动地进行光线强度的控制了。

5. 控制信号输出

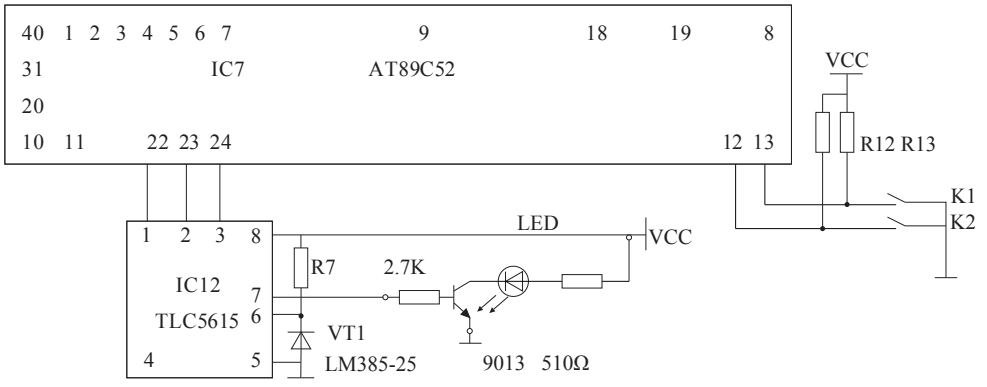


图 9-9 灯光自动控制原理图

灯光的自动控制原理见图9-9。电阻R7和稳压管T1（LM385-25）产生2.5V的稳定电压，使D/A转换器TLC5615输出电压在0~5V。TLC5615的7脚输出0~5V的模拟量电压，该电压由限流电阻2.7K加到三极管（9013）的基极，电阻510Ω为发光二极管LED的限流电阻。当TLC5615的7脚输出电压变化时，三极管的集电极与发射极之间的导通也跟随变化，三极管9013的输出电流驱动发光二极管LED，发光亮度也跟着变化。

单片机AT89C52通过P2.1（22脚）、P2.2（23脚）、P2.3（24脚），控制D/A转换器TLC5615输出0~5V的模拟量。开关K1、K2用于光线强度的设置，在开关断开时，两个上拉电阻R12、R13使单片机AT89C52的12、13脚为高电平。当开关K1合上，单片机AT89C52的1228脚为低电平，表示进入光线强度的设置。在K1合上同时，K2也合上，单片机通过三位LED数码管显示需要的设置值，显示值从50开始，以加1的方式自动显示到100，再从1到100如此循环。在此过程中，如果使K2断开，则K2断开时显示的值就作为光线强度的设置，同时使K1断开，设置工作完成。

灯光自动控制的完整原理见图9-10。IC1~IC3是0.5"共阴LED数码管，用于数字显示，IC4~IC6为CD4513，是LED数码管的译码驱动集成电路，它们由IC7 AT89C52单片机CPU的1~7脚处理。单片机CPU的18、19是时钟端，第31脚接高电平，单片机读取内部ROM内容，第9脚是复位脚。IC8 MAX-232芯片用于单片机与上位机通信。

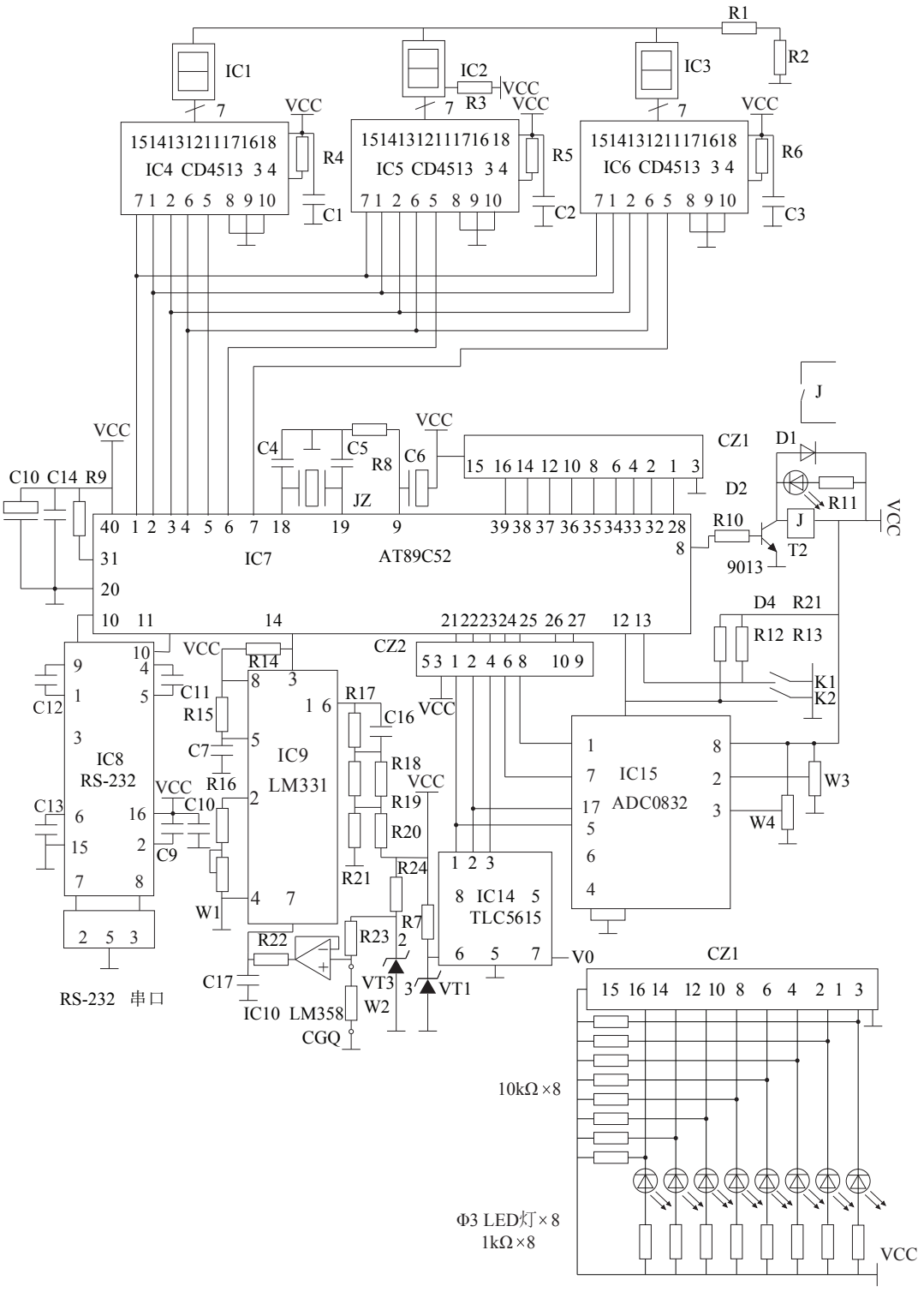


图 9-10 单片机灯光自动控制原理图

图9-11是单片机灯光自动控制原理图的电路板图。

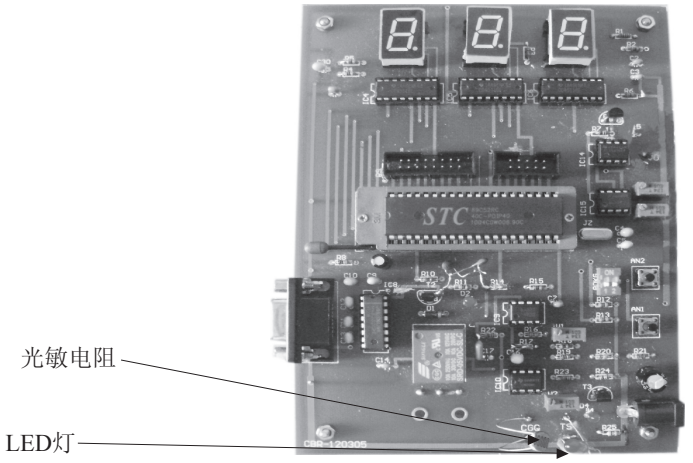


图 9-11 单片机灯光自动控制电路板

三、实训步骤

1. 读电路图

操作：针对图9-10单片机灯光自动控制原理图，结合前面的知识，了解灯光自动控制的原理，了解每个元器件的作用。

2. 看电路板

操作：结合图9-10单片机灯光自动控制原理图，在电路板上找到对应的器件，并了解原理图的连接线与电路板实际连线之间的区别。

3. 认识光敏电阻

操作：观察光敏电阻外观，用万用表的电阻档，测量光敏电阻有光线与没有光线下的电阻变化情况。

4. 电路板通电检查

操作：检查电路板上元器件，保证接触正常，220V/5V电源插到220V电源插座上，用万用表电压档测量是否为5V±0.2V ？如不正常检查原因或更换一个，将正常的电源插入电路板的5V电源接口，检查电路板上的C4电容旁边的＋、－端之间的电压是否为5V±0.2V？，如果不正常，则需检查或更换一台，直到正常为止。然后断开5V电源，插上自检用的单片机芯片，再接通5V电源，三位LED数码管显示器自动显示000～999并循环；用万用表的电压档测量数模转换电路IC12-TCL5615的第六脚是否为5V，如不正常，检查原因直到正常为止。用万用表的电压档测量数模转换电路IC12-TCL5615的第七脚，电压应该在0~5V循环变化；合上开关K3，调节频率输入电位器W1，三位LED数码管显示器的显示数字会发生变化，说明电路板正常，可以使用。

5. 光线测量与控制

操作：将220V/5V电源从220V电源插座上取下，使电路板断电，并使电位器W1与电路断开，将光敏电阻安装在电路板上，使光敏电阻接入电路。将调光灯LED安装在电路板上。

- 将自检用的单片机芯片取下，放上STC89C52RC单片机芯片。
- 将RS-232通信线的九针插头插入电路板上的RS-232插座，另一端USB插头插入计

算机上的USB插座。

- 打开计算机中的Keil软件，编写自动调光灯的C语言程序。
- 使电路板接通5V电源。
- 打开计算机中的ISP单片机在线烧写软件，将编写好的自动调光灯C语言固化在单片机芯片中。
- 运行程序，光敏电阻没有光线时，测量数模转换电路IC12-TCL5615第七脚的电压。光敏电阻有强烈光线再测量数模转换电路IC12-TCL5615第七脚的电压。改变光敏电阻的光线强度时，测量数模转换电路IC12-TCL5615第七脚的电压在变化，调光灯发光亮度也在变化。如果没有以上效果，对程序进行调试，直到达到要求为止。

四、自动调光灯的C51源程序

```
#include <reg52.h>              //输入单片机头文件
#define uchar unsigned char

float f0;                       //设置变量
float f1;                       //设置变量z
unsigned int dy;                //DA5615电压控制变量
unsigned int PL;                //当前测量的频率，是计算的依据
unsigned int PL0;               //中间变量
unsigned int PL1;               //十位显示数
unsigned int PL2;               //个位显示数
unsigned int PL3;               //小数位显示数
unsigned int PL4;               //中间变量
unsigned int xs;                //数码管显示地址和值

sbit P10=P1^0;
sbit P11=P1^1;
sbit P12=P1^2;
sbit P13=P1^3;
sbit P14=P1^4;
sbit P15=P1^5;
sbit P16=P1^6;
sbit P17=P1^7;
sbit P32=P3^2;
sbit P33=P3^3;

sbit DIN=P2^0;
sbit SCLK=P2^1;
```

```
    sbit CS=P2^2;                    //DA5615-片选地址

void DA5615(unsigned int x)          //DA5615应用
    {
      char i;
      x=x<<2;
      CS=0;
      for(i=12;i!=0;i--)
        {
       DIN=(x>>(i-1))&0x0001;
       SCLK=1;
        SCLK=0;
       }
      CS=1;
     }

    void init_t0()                   //定时器0初始化，设置为计数方式，对外部频率信号进行计数
    {
     TMOD=0x25;
     TH0=0x00;
     TL0=0x00;
     TR0=1;
    }

    void delay(unsigned int i)       //延时程序 10ms，
    {
     unsigned int j,k;
     for(k=0;k<i;k++)
     for (j=0;j<1820;j++);
     }

    void send_data()                 //每次采样1s
    {
      init_t0();
      delay(100);                    //调用延时程序，10ms*100=1s定时
        TR0=0;
        PL0=TL0+TH0*256;
        TR0=1;                       //计数器重置
     }
```

```
void main()
{
    dy=150; //设置LED灯初值
    while(1)
     {
      init_t0();                    //定时器0初始化，设置为计数方式，对外部频率信号进行计数

      DA5615(dy);
      send_data();
      if (P32==1)                   //P32=1，使光线控制值PL4=800
       {
                PL4=800;
       }
       if (P32==0)                  //P32=0，使光线控制值PL4=400
      {
              PL4=500;
       }
        f0=PL4;
              PL=PL0/2.5;     //使光线测量值在100~0
      PL=1000-PL;
      f1=PL;
     if ((f0-f1)>10)                //如果光线设置值比测量值大，表示光线弱，使LED灯变亮
       {
                dy=dy+2;
       }

      if ((f0-f1)<10)               //如果光线设置值比测量值小，表示光线强，使LED灯变暗
       {
              dy=dy-2;
       }

    PL1=PL/100;                     //频率的百位数
    PL2=(PL-PL1*100)/10;   //频率的十位数
    PL3=PL-PL1*100-PL2*10; //频率的个位数

    xs=224+PL1;                     //16-频率百位显示地址，后面为百位显示数据
    P1=xs;                          //由P1口输出显示
    delay(5);                       //延时时间
    xs=240+PL1;
    P1=xs;                          //显示地址为F，使显示的值保持稳定
```

```
    delay(5);
     xs=208+PL2;    //32-频率十位显示地址，后面为十位显示数据
     P1=xs;         //由P1口输出显示
     delay(5);      //延时时间
     xs=240+PL2;
     P1=xs;         //显示地址位F，使显示的值保持稳定
     delay(5);

      xs=176+PL3;   //48-频率个数位显示地址，后面为个位显示数据
      P1=xs;        //由P1口输出显示
       delay(5);    //延时时间
       xs=240+PL3;
        P1=xs;      //显示地址位F，使显示的值保持稳定
        delay(5);
        delay(10);
    }
}
```

任务评价

序号	操作内容	配分	得分	评价说明
1	读电路图	25分	≥22.5	能够清晰分析电路中各个元器件的作用与电路工作原理
			≥22.0	能够分析电路中70%元器件的作用与电路工作原理
			≥17.5	对电路中50%的元器件的作用与电路工作原理进行分析
			≥15.0	对电路中30%的元器件的作用与电路工作原理进行分析
			＜15.0	达不到上一条款的
2	看电路板	5分	≥4.5	原理图与电路板之间能够清晰地一一对应
			≥4.0	原理图与电路板之间的70%元器件能够一一对应
			≥3.5	原理图与电路板之间的60%元器件能够一一对应
			≥3.0	原理图与电路板之间的40%元器件能够一一对应
			＜3.0	达不到上一条款的
3	认识光敏电阻	5分	≥4.5	能够正确使用万用表并测量光敏电阻的光敏特性
			≥4.0	能够正确使用万用表但测量方法有误
			≥3.5	不能熟练使用万用表且测量方法有误
			≥3.0	在提示下完成工作
			＜3.0	在提示下也不能完成工作
4	电路板通电检查	5分	≥4.5	独立完成电路板通电检查各项内容
			≥4.0	能够完成电路板通电检查各项内容，但有1-2个错误
			≥3.5	不能按照要求的顺序进行，但主要内容能完成
			≥3.0	在提示下完成工作
			＜3.0	在提示下也不能完成工作
5	光线测量与控制	60分	≥54.0	能将参考程序编译、烧写、运行，并可修改部分程序
			≥48.0	能将参考程序编译、烧写、运行，能修改个别语句
			≥42.	仅能使程序编译、烧写、运行，不能修改个别语句
			≥36.0	过程有多次反复，在提示下完成工作
			＜36.0	在提示下也不能完成工作
总成绩		100分		在教学中，评价内容由教师自己掌握

实训报告

一、实训名称

自动调光灯的设计

二、实训目的与要求

1. 熟悉自动调光灯的硬件电路与C语言程序。
2. 掌握光敏电阻特性和光线强度测量方法。
3. 掌握LM331、TLC5615电路的性能和应用。

三、实训设备

PC机一台（附带软件：Keil software uVision 2、ISP、RS232/USB转换软件）单片机综合应用实验板一块、光敏电阻一只、调光灯一只、数字万用表一台 频率表一台、220V/5V电源一台、RS232/USB转换信号线、焊接器材若干。

四、实训内容及调试

1. 实训内容

- 单片机综合应用实验板自检测试
- LM331 V/F应用
- TLC5615 数/模转换应用
- 光敏电阻的应用
- 光线自动控制

2. 调试内容

- 测量和记录单片机综合应用实验板电源电压
- 测量和记录LM385-25的稳定电压值
- 应用参考程序进行调试直到成功
- 测量和记录开关K1、K2在断开和合上时单片机的12、13 脚电压值
- 测量和记录LM331芯片的输入电压与输出频率值
- 测量和记录单片机输出的数字量和TLC5615的输出电压
- 测量和记录光敏电阻A端的电压与LED数码管的显示值
- 观察光敏电阻上光线变化与调光灯亮度变化的关系

3. 程序分析

- 分析上面的测量和记录的内容，并说明原理
- 分析开关K1、K2关于数字设置的程序，并修改程序改变设置方式
- 分析单片机输出的字量和TLC5615输出电压的关系，并修改程序改变它们的相互关系

五、实训小结

调试过程中所遇到的问题。

1. 问题解决的思路和的方法。
2. 修改程序的效果。
3. 实训收获。

9.2 微型直流电动机的速度控制

学习目标

1. 熟悉应用单片机测量电机的转速和控制技术。
2. 了解微型直流电机的性能和应用。
3. 掌握红外管的性能和应用。
4. 掌握转速的测量方法。
5. 掌握转速的控制。
6. 学会应用单片机实现数字量的输入和模拟量的输出技术。

微型直流电动机的应用无所不在的，常常作为运动机械负载的主要动力源。在盒式录像机中，微型电机是重要元件，也是主导轴驱动以及控制磁带的张力的重要元件。其次，微型电机在办公自动化的应用中也十分广泛，例如打印机、磁盘驱动器、复印机等。在保健用品、玩具、电控门、切割设备、印刷机械、电动工具、机器人、医疗器械等领域都有微型直流电动机的应用。常用直流电机实物图见图9-12。

那么，如何应用单片机对微型直流电机进行控制、驱动，使其按照我们的需要进行工作是本节需要解决的问题。

图 9-12 微型直流电机

任务实施

一、实训器材

PC机、单片机综合应用实验板、微型直流电机、红外发射接收管、数字万用表、频率表、220V/5V电源、220V/30W电烙铁、松香、焊锡丝、导线、螺丝刀、220V电源插座等。

二、知识准备

1. 微型直流电机的工作原理

直流电机的原理结构如图9-13所示，包括转子、整流子、电刷、轴、磁场铁。磁场铁是永久磁铁。导体受力的方向用左手定则确定。这一对电磁力形成了作用于电枢一个力矩，这个力矩在旋转电机里称为电磁转矩，转矩的方向是逆时针方向，企图使电枢逆时针方向转动。如果此电磁转矩能够克服电枢上的阻转矩（例如由摩擦引起的阻转矩以及其它负载转矩），电枢就能按逆时针方向旋转起来。当电枢转了180°后，导体cd转到N极下，导体ab转到S极下时，由于直流电源供给的电流方向不变，仍从电刷A流入，经导体cd、ab后，从电刷B流出。这时导体cd受力方向变为从右向左，导体ab受力方向是从左向右，产生的电磁转矩的方向仍为逆时针方向。

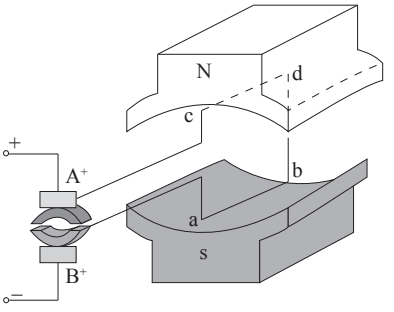


图 9-13 微型直流电机原理图

因此，电枢一经转动，由于换向器配合电刷对电流的换向作用，直流电流交替地由导体ab和cd流入，使线圈边只要处于N极下，其中通过电流的方向总是由电刷A流入的方向，而在S极下时，总是从电刷B流出的方向。这就保证了每个极下线圈边中的电流始终是一个方向，从而形成一种方向不变的转矩，使电动机能连续地旋转。这就是直流电动机的工作原理。

2. 微型直流电机的速度控制

因为载流导体在磁场中会受到力的作用，若磁场与载流导体互相垂直，作用在导体上的电磁力大小为

$$F = B \cdot L \cdot I \qquad (9\text{-}2)$$

式中　B为磁场强度；L为导线有效长度；I为电流。

所以，当磁场强度、导线有效长度不变的情况下，改变电机的负载电压，就改变了通过电机线圈的电流，也就改变了电机的力矩，从而改变电机的旋转速度。在用单片机控制的时候，单片机通过D/A转换芯片输出变化的直流电压，就可以使电机的旋转速度改变。

3. 电机旋转的速度测量

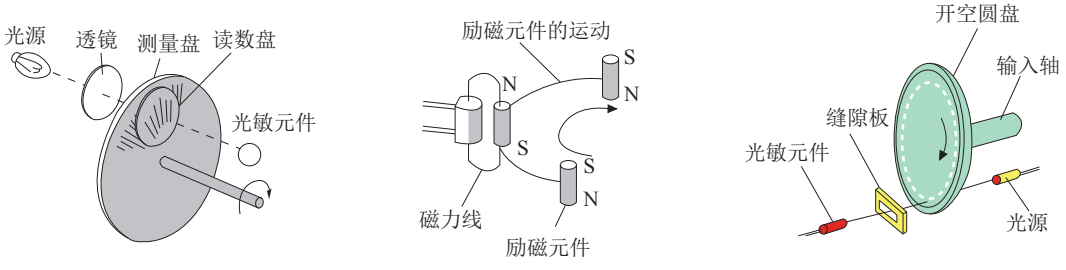
很多场合都需要对旋转设备的转速进行测量，如汽车、轿车、发电机等，如图9-14所示为各种转速表。

（a）　　（b）

图 9-14　各种转速表

（a）汽车转速表　（b）数字转速表

转速测量的方法有很多种，图9-15（a）是利用一般光源发出的光通过光学透镜聚焦成窄的光束穿过与电机轴为一体的转盘上的孔，照射到光敏器件上产生脉冲信号，计算单位时间内的脉冲数就可以测量电机转速了。图9-15（b）是将小的永磁铁放置在与电机轴为一体的转盘上，在转盘的边缘放置霍尔传感器，转盘在旋转过程中，永磁铁依次经过霍尔传感器而产生脉冲信号，同样，计算单位时间内的脉冲数就可以测量电机转速了。图9-15（c）是利用激光光源发出的光穿过与电机轴为一体的转盘上的孔，照射到光敏器件上产生脉冲信号，根据单位时间内的脉冲数就可以计算出电机的转速。

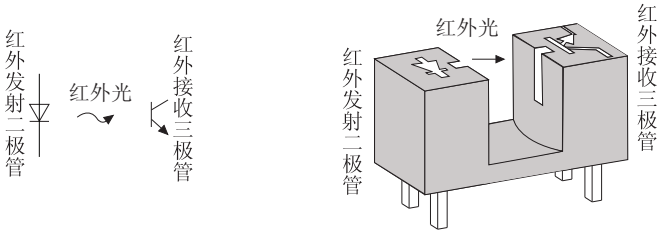


（a）　　（b）　　（c）

图 9-15　转速的测量

（a）转速测量方法1　（b）转速测量方法2　（c）转速测量方法3

图9-16所示为开关型红外光电传感器和转盘来实现电机的转速测量。



（a）　　（b）

图 9-16　开关型红外光电传感器

（a）开关型红外光电传感器符号　（b）开关型红外光电传感器外形

开关型红外光电传感器中的红外发射二极管发出红外线，由红外三极管接收。当有物体通过传感器中间的空隙时就遮挡住红外线，红外三极管接收不到红外线，利用这种方法就可以测量电机的旋转速度了。如图9-17，微型电机旋转时带动边缘有孔的圆盘，圆盘在旋转时，红外发射二极管发出的红外线时而被遮挡，时而通过，圆盘上的小孔为红外三极管接收，在红外三极管的集电极上产生脉冲信号，计算单位时间内的脉冲数就可以测量电机的转速了。

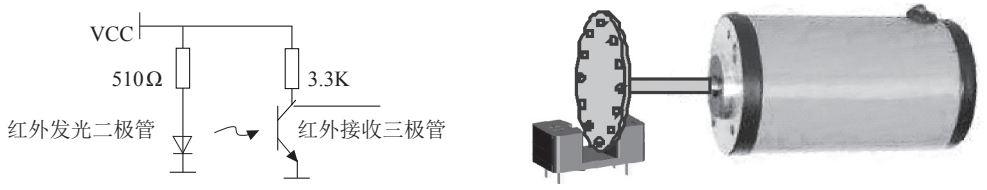


图 9-17 微型电机转速的测量

图中，510Ω是红外发光二极管的限流电阻，3.3K是红外接收三极管的负载电阻，V为红外接收三极管的输出信号电压。当红外线没有被遮挡，红外接收三极管处于饱和状态，输出电压V=0V。当红外线被遮挡，红外接收三极管处于截止状态，输出电压V=5V。

4. 微型电机转速的控制

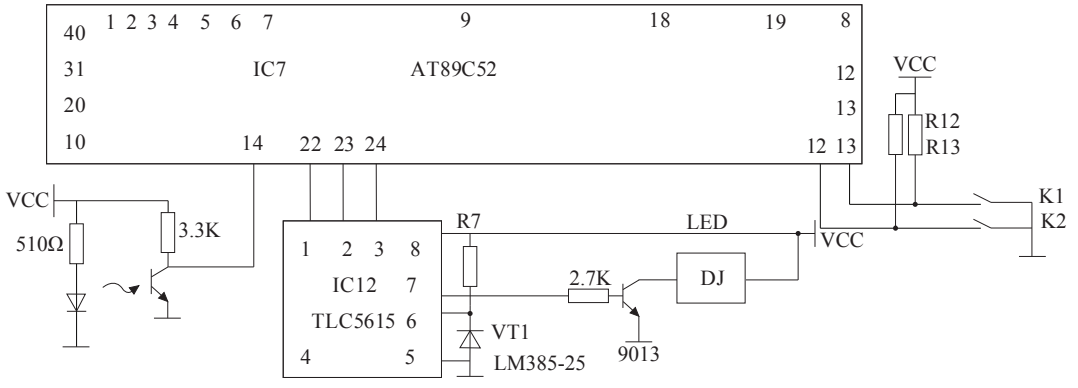


图 9-18 微型电机转速控制原理

见图9-18，红外接收三极管的集电极上产生的脉冲电压信号V，输入到单片机的14脚，将单片机的定时器/计数器0作为外部脉冲计数方式，记录单位时间内测量到的脉冲数，通过计算得到转速。如果圆盘上孔的数目为N，1s内测量到M个脉冲，那么转速RPM（转/分）为：

$$RPM=（M/N）*60 \qquad （9\text{-}3）$$

在此例中，采用直流5V永磁直流减速电机，转速为1200转/分的微型电机，圆盘上孔的数目为20个。

单片机将计算好的电机的转速由三位LED数码管显示。通过开关K1、K2进行转速的设置，当K1合上，表示进入转速设置状态，三位LED数码管显示500转/分并闪烁，同时按下K2，三位LED数码管显示从500加1到999再到1，再到500如此循环，在需要设置的数字显示的时候，断开K2、K1，设置完成。设置值存储在单片机存储器中，设为S0。单片机在工作过程中，不断测量电机当前的速度为S。若：

S0－S > 0 表示电机当前的速度慢，单片机增加TLC5615的电压，使电机的电压增加，

转速加快。

S0－S < 0 表示电机当前的速度快，单片机减少TLC5615的电压，使电机的电压减少，转速变慢。

S0－S = 0 表示电机当前的速度正好，单片机保持TLC5615的电压。

这样，就完成了电机的速度控制。

三、实训步骤

1. 读电路图

操作：针对图9-18单片机转速控制原理图，结合前面的知识，了解转速控制的原理，了解每个元器件的作用。

2. 看电路板

操作：结合图9-18单片机转速控制原理图，在电路板上找到对应的器件，并了解原理图的连接线与电路板实际连线之间的区别。

3. 认识开关型红外光电阻传感器、5V微型直流电机

操作：观察开关型红外光电阻传感器外观，用万用表电阻档的红笔、黑笔，正向和反向测量红外发光二极管与红外三极管光线下的电阻变化情况。观察5V微型直流电机，并调节电压使在0~5V变化，看电机的旋转状况。

4. 电路板通电检查

操作：检查电路板上元器件，保证接触正常，220V/5V电源插到220V电源插座上，用万用表电压档测量是否为5V±0.2V？如不正常，检查原因或更换一个，将正常的电源插入电路板的5V电源接口，检查电路板上的C4电容旁边的＋、－端之间的电压是否为5V±0.2V。如果不正常，则需检查或更换一台，直到正常为止。然后断开5V电源，插上自检用的单片机芯片，再接通5V电源，三位LED数码管显示器自动显示000~999并循环；用万用表的电压档测量数模转换电路IC12-TCL5615的第六脚是否为5V，如不正常，检查原因直到正常为止。用万用表的电压档测量数模转换电路IC12-TCL5615的第七脚，电压应该在0~5V循环变化；合上开关K1，调节频率输入电位器W1，三位LED数码管显示器的显示数字会发生变化，说明电路板正常，可以使用。

5. 5V微型直流电机转速的测量与控制

操作：

- 将5V微型直流电机的应用板接入电路，并检查线路是否正确
- 将带有单片机转速控制的单片机芯片插入40芯插座
- 接通5V电源，电机将开始从慢到快再到慢循环工作，三位LED数码管不断显示电机当前的转速
- 按前面的要求对电机的转速进行设置，并观察电机转速变化的情况
- 用频率表测量单片机14脚的输入频率，并计算电机的转速
- 将单片机与计算机连接，自己修改程序，观察控制的效果

四、微型直流电动机速度控制的C51源程序

```
#include <reg52.h>   //输入单片机头文件
#define uchar unsigned char
```

```
float f0;                    //设置变量
float f1;                    //设置变量
unsigned int dy;             //DA5615电压控制变量
unsigned int PL;             //当前测量的频率，是计算的依据
unsigned int PL0;            //中间变量
unsigned int PL1;            //十位显示数
unsigned int PL2;            //个位显示数
unsigned int PL3;            //小数位显示数
unsigned int PL4;            //中间变量
unsigned int xs;             //数码管显示地址和值

sbit P10=P1^0;
sbit P11=P1^1;
sbit P12=P1^2;
sbit P13=P1^3;
sbit P14=P1^4;
sbit P15=P1^5;
sbit P16=P1^6;
sbit P17=P1^7;
sbit P32=P3^2;
sbit P33=P3^3;

sbit DIN=P2^0;
sbit SCLK=P2^1;
sbit CS=P2^2;                //DA5615-片选地址


void DA5615(unsigned int x)  //DA5615应用
{
  char i;
  x=x<<2;
  CS=0;
  for(i=12;i!=0;i--)
    {
     DIN=(x>>(i-1))&0x0001;
     SCLK=1;
      SCLK=0;
  }
  CS=1;
}
```

```
void init_t0()                  //定时器0初始化，设置为计数方式，对外部频率信号进行计数
{
  TMOD=0x25;
  TH0=0x00;
  TL0=0x00;
  TR0=1;
}

void delay(unsigned int i)    // 延时程序, 10ms，
{
  unsigned int j,k;
  for(k=0;k<i;k++)
  for (j=0;j<1820;j++);
 }

void send_data()                //每次采样1s
{
  init_t0();
  delay(100);                   //调用延时程序，10ms*100=1s定时
    TR0=0;
    PL0=TL0+TH0*256;
    TR0=1;                      //计数器重置
 }

void main()
{
  dy=150;                       //设置电机旋转初值
  while(1)
    {
     init_t0();                 //定时器0初始化，设置为计数方式，对外部频率信号进行计数
     DA5615(dy);
     send_data();
      if (P32==1)               //P32=1，使电机转速控制值PL4=800
     {
              PL4=800;
      }

    if (P32==0)                 //P32=0，使电机转速控制值PL4=400
     {
              PL4=400;
       }
```

```
        f0=PL4;

            PL=PL0;                 //记录电机转速值
    f1=PL;

    if ((f0-f1)>10)                 //如果电机转速设置值比测量值大，表示太慢，使电机加速
     {
                dy=dy+2;
      }

     if ((f0-f1)<10)                //如果电机转速设置值比测量值小，表示太快，使电机减速
      {
                  dy=dy-2;
      }

     PL1=PL/100;                    //频率的百位数
     PL2=(PL-PL1*100)/10;           //频率的十位数
     PL3=PL-PL1*100-PL2*10;         //频率的个位数

     xs=224+PL1;                    //16-频率百位显示地址，后面为百位显示数据
     P1=xs;                         //由P1口输出显示
     delay(5);                      //延时时间
      xs=240+PL1;
     P1=xs;                         //显示地址为F，使显示的值保持稳定
     delay(5);

     xs=208+PL2;                    //32-频率十位显示地址，后面为十位显示数据
     P1=xs;                         //由P1口输出显示
     delay(5);                      //延时时间
     xs=240+PL2;
     P1=xs;                         //显示地址位F，使显示的值保持稳定
     delay(5);

      xs=176+PL3;                   //48-频率个数位显示地址，后面为个位显示数据
      P1=xs;                        //由P1口输出显示
      delay(5);                     //延时时间
      xs=240+PL3;
      P1=xs;                        //显示地址位F，使显示的值保持稳定
      delay(5);
      delay(10);
   }
}
```

任务评价

序号	操作内容	配 分	得 分	评 价 说 明
1	读电路图	25分	≥22.5	能够清晰分析电路中各个元器件的作用与电路工作原理
			≥22.0	能够分析电路中70%元器件的作用与电路工作原理
			≥17.5	对电路中50%的元器件的作用与电路工作原理进行分析
			≥15.0	对电路中30%的元器件的作用与电路工作原理进行分析
			<15.0	达不到上一条款的
2	看电路板	5分	≥4.5	原理图与电路板之间能够清晰地一一对应
			≥4.0	原理图与电路板之间的70%元器件能够一一对应
			≥3.5	原理图与电路板之间的60%元器件能够一一对应
			≥3.0	原理图与电路板之间的40%元器件能够一一对应
			<3.0	达不到上一条款的
3	认识开关型红外光电传感器	5分	≥4.5	能够正确使用万用表并测量红外传感器的开关特性
			≥4.0	能够正确使用万用表但测量方法有误
			≥3.5	不能熟练使用万用表且测量方法有误
			≥3.0	在提示下完成工作
			<3.0	在提示下也不能完成工作
4	电路板通电检查	5分	≥4.5	独立完成电路板通电检查各项内容
			≥4.0	能够完成电路板通电检查各项内容，但有1-2个错误
			≥3.5	不能按照要求的顺序进行，但主要内容能完成
			≥3.0	在提示下完成工作
			<3.0	在提示下也不能完成工作
5	转速测量与控制	60分	≥54.0	能将参考程序编译、烧写、运行，并可修改部分程序
			≥48.0	能将参考程序编译、烧写、运行，能修改个别语句
			≥42.	仅能使程序编译、烧写、运行，不能修改个别语句
			≥36.0	过程有多次反复，在提示下完成工作
			<36.0	在提示下也不能完成工作
总成绩		100分		在教学中，评价内容由教师自己掌握

实训报告

一、实训名称

微型直流电机转速控制的设计

二、实训目的与要求

1. 熟悉直流电机转速控制的硬件电路与C语言程序。
2. 掌握开关型红外传感器的转速测量方法。
3. 掌握微型直流电机转速控制方法和应用。

三、实训设备

PC机一台（附带软件：Keil software uVision 2、ISP、RS232/USB转换软件）单片机综合应用实验板一块、5V微型直流电机的应用板一块（包括开关型红外传感器、微型直流电机）、数字万用表一台、频率表一台、220V/5V电源一台、RS232/USB转换信号线、焊接器材若干。

四、实训内容及调试

1. 实训内容

- 单片机综合应用实验板自检测试
- LM331 V/F应用
- TLC5615 数/模转换应用
- 开关型红外传感器的应用
- 微型直流电机的转速控制应用
- 测量和记录单片机综合应用实验板电源电压
- 测量和记录LM385-25的稳定电压值
- 应用参考程序进行调试直到成功
- 测量和记录LM331芯片的输出频率和电机的转速
- 测量LC5615的输出电压和电机转速的关系

2. 程序分析

- 分析上面的测量和记录的内容，并说明原理。
- 分析开关K1、K2关于数字设置的程序，并修改程序改变设置方式。
- 分析TLC5615输出电压和电机的转速关系，并修改程序改变它们的相互关系。

五、实训小结

1. 调试过程中所遇到的问题。
2. 问题解决的思路和的方法。
3. 修改程序的效果。
4. 实训收获。

9.3 智能温度测量仪

学习目标

1. 熟悉应用单片机测量温度的技术。
2. 熟悉关于温度的概念和温度的重要性。
3. 熟悉各种温度传感器的性能和应用。
4. 掌握温度的测量方法。
5. 学会应用单片机实现模拟量输入技术。

任务实施

一、实训器材

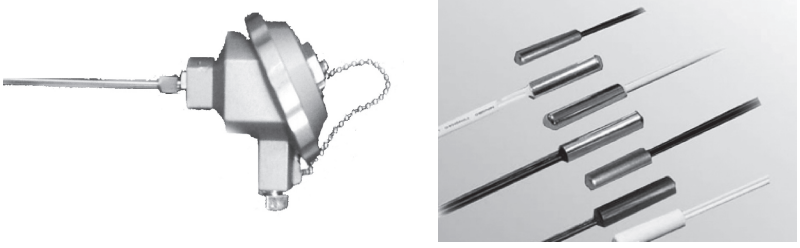
PC机、单片机综合应用实验板、热敏电阻、数字万用表、频率表、220V/5V电源、220V/30W电烙铁、松香、焊锡丝、导线、螺丝刀、220V电源插座等。

二、知识准备

图 9-19 温度的应用(食品保温柜)

温度是一个基本的物理量，自然界中一切过程无不与温度密切相关，适宜的温度也是人类生存的条件。温度的高低会影响植物的生长，为了使家里或办公室的温度更适宜，需要安置空调器来调节室内的温度。加工好的食品为了可以长时间保持食品的新鲜度，需要放置保温箱中，如图9-19所示。许多观赏鱼都需要一定的温度范围才能正常生长。这些都说明温度的重要性，需要对温度进行测量和控制。

1. 温度测量的工作原理



（a）　　（b）

图 9-20 温度传感器

（a）热电偶温度传感器　（b）热敏电阻温度传感器

测量温度的装置称为温度传感器。温度传感器是一种能够将温度变化的物理量，转换成其他的物理量来表示的装置，用于温度测量的传感器种类很多，这里简单介绍两种常用的传感器。

如图9-20（a）所示的是热电偶温度传感器。热电偶测温的基本原理是将两种不同材料的导体或半导体A和B焊接起来，构成一个闭合回路。当导体A和B的两个连接点1和2之间存在温差时，两者之间便产生电动势，因而在回路中形成一定的电流，这种现象称为热电效应。热电偶就是利用这一效应来工作的，如图9-21所示。

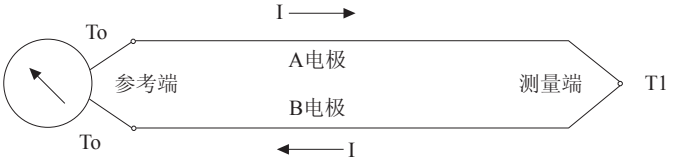
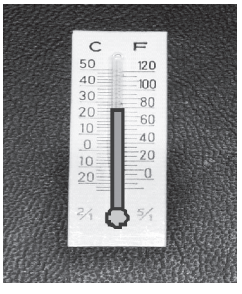


图 9-21 热电偶温度传感器工作原理

图9-20（b）所示的是热敏电阻温度传感器。热敏电阻被广泛应用于电饭煲、电烤箱、电取暖器上，在工业、医疗、环保、气象、食品加工方向的温度测量和控制。它是以锰、钴、镍和铜等金属氧化物为主要材料，采用陶瓷工艺制造而成的。温度低时，其电阻值较高；温度高，电阻值降低。NTC热敏电阻器在室温下的变化范围在100Ω~1000kΩ。它的测量范围一般为－40℃~+150℃，负温度系数热敏电阻器温度计的精度可以达到0.1℃，感温时间在10s以下。所以热敏电阻的特点是体积小、反应灵敏、测量方便。体积可以小到针头大小，便于测量微小体积的对象，由于它将温度的变化转换成电阻值的变化，而且电阻很大，一般采用电阻串联分压电路就可以获得测量信号。

2. 常用温度表的测量原理



（a）

（b）

（c）

图 9-22 各种温度计

（a）水银温度计 （b）膨胀式指针温度计 （c）数字式温度计

图9-22（a）是水银温度计，水银被封装在玻璃管下端的玻璃球内，由于热胀冷缩的原因，温度上升时水银的体积增加，增加部分的水银沿玻璃管向上升，温度变化越大，增加的体积越多，上升的高度越高。这样通过水银体积的变化，把温度变化转换成水银在玻璃管内高度的变化，从而测量玻璃管内水银的高度可以知道温度的高低。图9-22（b）是膨胀式指针温度计 它由两片膨胀系数不同的金属牢固地粘合在一起，其一端固定，另一端通过传动机构和指针相连。当温度变化时，由于膨胀系数不同，双金属片产生位移，带动指针指示相应的温度。这便是双金属温度计的工作原理。图9-22（c）是液晶显示的数字温度表，以热敏电阻作为温度传感器。

3. 热敏电阻的温度测量原理

在这里选用型号为MF-5E 10kΩ/25℃ B=3935的高精度热敏电阻。

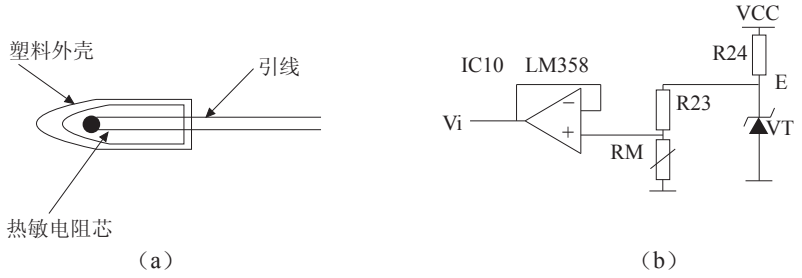


图 9-23 热敏电阻的温度测量

（a）热敏电阻结构　（b）测量电路

如图9-23所示，电阻R24、稳压管T3组成稳压电路，产生稳定的E=2.5V电压，作为热敏电阻温度测量电路的电源。电阻R23、热敏电阻RM组成电阻串联电路，在热敏电阻RM上产生的分压值V就是温度的测量值。测量值V由LM358构成的电压跟随器，使测量部分电路与后面的电路进行隔离。

$$Vi = V = E * RM /（RM + R23）\qquad (9\text{-}4)$$

在该测量电路中，热敏电阻RM，把温度变化转换成电阻值的变化，再由电阻分压得到Vi电压值，把电压V0输入到单片机，经过计算就可以得到温度值。

表9-2是热敏电阻在不同温度下的电阻值，作为测量温度用的热敏电阻，它的阻值随温度上升而减少。

表 9-2 热敏电阻分度值表

温度	电阻值/kΩ	频率/Hz	频率差	温度	电阻值/kΩ	频率/Hz	频率差
-10℃	55.328	2117		26℃	9.57232	1223	-27
-9℃	52.4065	2099	-18	27℃	9.16529	1196	-27
-8℃	49.656	2081	-18	28℃	8.77779	1169	-27
-7℃	47.0659	2062	-19	29℃	8.40868	1142	-27
-6℃	44.6261	2040	-20	30℃	8.05731	1116	-26
-5℃	42.3268	2022	-20	31℃	7.72243	1089	-27
-4℃	40.1594	2002	-20	32℃	7.40329	1063	-26
-3℃	38.1155	1980	-22	33℃	7.09842	1038	-25
-2℃	36.1875	1959	-21	34℃	6.80836	1013	-25
-1℃	34.368	1937	-22	35℃	6.53131	988	-25
0℃	32.6505	1914	-23	36℃	6.26575	963	-25
1℃	31.0321	1891	-23	37℃	6.01647	939	-24
2℃	29.4999	1843	-24	38℃	5.77605	915	-24
3℃	28.0524	1843	-24	39℃	5.54653	892	-23

（续表）

温度	电阻值/kΩ	频率/Hz	频率差	温度	电阻值/kΩ	频率/Hz	频率差
4℃	26.6846	1819	-24	40℃	5.32734	869	-23
5℃	25.3912	1794	-25	41℃	5.11797	846	-23
6℃	24.1682	1768	-26	42℃	4.91794	824	-22
7℃	23.0112	1743	-25	43℃	4.72677	802	-22
8℃	21.9163	1717	-26	44℃	4.54391	781	-21
9℃	20.8798	1690	-27	45℃	4.36933	760	-21
10℃	19.8983	1664	-26	46℃	4.20084	740	-20
11℃	18.9686	1637	-27	47℃	4.04081	720	-20
12℃	18.0876	1610	-27	48℃	3.88951	700	-20
13℃	17.2626	1583	-27	49℃	3.74317	681	-19
14℃	16.4609	1555	-28	50℃	3.6031	662	-19
15℃	15.71	1528	-27	51℃	3.46769	644	-18
16℃	14.9977	1500	-28	52℃	3.34055	626	-18
17℃	14.3216	1472	-28	53℃	3.21754	609	-17
18℃	13.6798	1444	-28	54℃	3.09962	592	-17
19℃	13.0704	1416	-28	55℃	2.9866	575	-17
20℃	12.4916	1388	-28	56℃	2.87849	559	-16
21℃	11.9416	1361	-27	57℃	2.77455	543	-16
22℃	11.4189	1333	-28	58℃	2.6752	528	-15
23℃	10.922	1305	-28	59℃	2.57952	513	-15
24℃	10.922	1278	-27	60℃	2.4882	498	-15
25℃	10	1250	-28				

4. 智能温度测量仪的设计

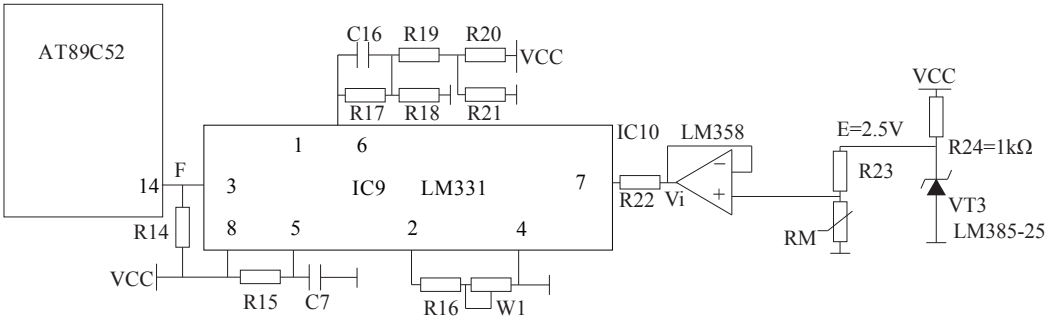


图 9-24 智能温度测量的原理

如图9-24所示是智能温度测量仪原理的测量部分。如上所述，温度的变化通过测量电路转换为Vi电压，Vi电压由LM331V/F转换芯片，把温度变化转换成频率信号，输入单片机的14脚。

设LM331的V/F转换为1000Hz/V　R21=10kΩ　E=2.5V

那么：

-10℃：RM (-10)= 55.3284kΩ　Vi (-10) =2.5×55.3284/(55.3284+10)=2.1173V————2117Hz

0℃：RM (0) = 32.6505kΩ　Vi (0) =2.5×32.6505/(32.6505+10)=1.9138V————1914Hz

20℃：RM (20) = 12.492kΩ　Vi (20) =2.5×12.492/(12.492+10)=1.3885V————1388Hz

25℃：RM (25) = 10kΩ　Vi (25) =2.5×10/(10+10)=1.250V————1250Hz

30℃：RM (30) = 8.057kΩ　Vi (30) =2.5×8.057/(8.057+10)=1.1155V————1116Hz

40℃：RM (40) = 5.327kΩ　Vi (40) =2.5×5.327/(5.327+10)=0.8688V————869Hz

分析上面的计算可知，-10℃~40℃的50℃温差，频率变化为2117 – 869 = 1248Hz，每1℃温度变化能够产生近24Hz的变化，是一个分辨率很高的测量电路。

频率信号输入到单片机AT89C52的14脚作为外部脉冲的输入端，每秒的脉冲数作为温度测量的计算基数。由于热敏电阻的阻值与温度的关系为非线性的，所以温度与频率的关系也是非线性的。对非线性的测量，使用常规的电路进行线性化非常困难或者是无法实现的。在本例中，可以充分发挥单片机的智能功能来实现高精度的测量。

在-10~40℃的常用温度范围内，首先计算出每5℃对应的频率值，例如：20℃—1389Hz，25℃—1250Hz，30℃—1116Hz，它们之间的频率差分别为139Hz和134Hz，

在20~25℃：139/5=27.8Hz/℃　139/50=2.78Hz/0.1℃ 得到：

20℃—1389Hz　21℃—1361.2Hz　22℃—1333.4Hz　23℃—1305.6Hz　24℃—1277.8Hz

在25~30℃：134/5=26.8Hz/℃　134/50=2.68Hz/0.1℃ 得到：

25℃—1250Hz　26℃—1223.2Hz　27℃—1196.4Hz　28℃—1169.6Hz　29℃—1142.8Hz

所以，单片机数据处理过程中，当频率在1389~1250Hz，表示温度在20~25℃。

设：当前频率值P=1350Hz，温度T为：T=20 + (1389 – 1350) / 27.8Hz = 20 + 1.4 = 21.4℃

同样当前频率值P=1150Hz 温度T为：T=25 + (1250 – 1150) / 26.8Hz = 25 + 3.7 = 28.7℃

因此，单片机首先测量频率值，然后判别该频率在哪个温度范围，再运用计算公式直接计算出相应的温度。由于有了单片机，非线性的问题轻松得到了解决，这就是单片机的智能功能的优点。

三、智能温度测量仪的C51源程序

```
#include <reg52.h>
unsigned int i;
float f;

unsigned int PL;     //当前测量的频率，是计算的依据
unsigned int PL0;
unsigned int PL1;    //十位显示数
unsigned int PL2;    //个位显示数
unsigned int PL3;    //小数位显示数
```

```
unsigned int PL4;
unsigned int xs;              //数码管显示地址和值
sbit P10=P1^0;
sbit P11=P1^1;
sbit P12=P1^2;
sbit P13=P1^3;
sbit P14=P1^4;
sbit P15=P1^5;
sbit P16=P1^6;
sbit P17=P1^7;
sbit P32=P3^2;
sbit P33=P3^3;

void init_t0()                //定时器0初始化，设置为计数方式，对外部频率信号进行计数
{
  TMOD=0x25;
  TH0=0x00;
  TL0=0x00;
  TR0=1;
}

void delay(unsigned int i) //  延时程序, 10ms,
{
    unsigned int j,k;
    for(k=0;k<i;k++)
    for (j=0;j<1820;j++);
 }

void send_data()              //每次采样1s
{
  init_t0();
   delay(100);                //调用延时程序，10ms*100=1s定时
     TR0=0;
     PL0=TL0+TH0*256;
     TR0=1;                   //计数器重置
 }

void main()
{
   unsigned int temp;
```

```
//init_t1();                    //定时器1初始化，作为波特率发生器
init_t0();                      //定时器0初始化，设置为计数方式，对外部频率信
                                  号进行计数

 while(1)
 {
   send_data();
   PL=PL0;

    P17=0;
 if (PL>=1914)                  //-10 ~ 0℃
   { PL=(PL-1914)/2.2;
           }
 if (PL<1914 && PL>=1528)       // 0 ~15℃
   { PL=150-(PL-1528)/2.6;
           }
  if (PL<1528 && PL>=869)       //15 ~ 40℃
   { PL=400-(PL-869)/2.6;
           }
 if (PL<869 && PL>=662)         //40 ~ 50℃
   { PL=500-(PL-662)/2.1;
           }

 if (PL<6629 && PL>=498)        //50 ~ 60℃
   { PL=600-(PL-498)/2.1;
           }

PL1=PL/100;                     //百位数
PL2=(PL-PL1*100)/10;            //十位数
PL3=PL-PL1*100-PL2*10;          //个位数

if (PL0>1915)
  { PL1=9;
  }

xs=224+PL1;                     //224-百位显示地址，后面为百位显示数据
P1=xs;                          //由P1口输出显示
P17=0;
delay(5);                       //延时时间
xs=240+PL1;
```

```
    P1=xs;                          //显示地址为F，使显示的值保持稳定
    P17=0;
    delay(5);

    xs=208+PL2;                     //208十位显示地址，后面为十位显示数据
    P1=xs;                          //由P1口输出显示
    P17=0;
    delay(5);                       //延时时间
    xs=240+PL2;
    P1=xs;                          //显示地址位F，使显示的值保持稳定
    P17=0;
    delay(5);

    xs=176+PL3;                     //176个数位显示地址，后面为懈鳊位显示数据
    P1=xs;                          //由P1口输出显示
    P17=0;
    delay(5);                       //延时时间
    xs=240+PL3;
    P1=xs;                          //显示地址位F，使显示的值保持稳定
    P17=0;
    delay(5);

   if (PL0>1915)
      { PL1=0;
      }

    xs=224+PL1;                     //224-百位显示地址，后面为百位显示数据
    P1=xs;                          //由P1口输出显示
    P17=0;
    delay(5);                       //延时时间
    xs=240+PL1;
    P1=xs;                          //显示地址为F，使显示的值保持稳定
    P17=0;
    delay(5);

      }
}
```

任务评价

序号	操作内容	配 分	得 分	评价说明
1	读电路图	25分	≥22.5	能够清晰分析电路中各个元器件的作用与电路工作原理
			≥22.0	能够分析电路中70%元器件的作用与电路工作原理
			≥17.5	对电路中50%的元器件的作用与电路工作原理进行分析
			≥15.0	对电路中30%的元器件的作用与电路工作原理进行分析
			＜15.0	达不到上一条款的
2	看电路板	5分	≥4.5	原理图与电路板之间能够清晰地一一对应
			≥4.0	原理图与电路板之间的70%元器件能够一一对应
			≥3.5	原理图与电路板之间的60%元器件能够一一对应
			≥3.0	原理图与电路板之间的40%元器件能够一一对应
			＜3.0	达不到上一条款的
3	认识热敏电阻温度传感器	5分	≥4.5	能够正确使用万用表并测量热敏电阻的温度特性
			≥4.0	能够正确使用万用表但测量方法有误
			≥3.5	不能熟练使用万用表且测量方法有误
			≥3.0	在提示下完成工作
			＜3.0	在提示下也不能完成工作
4	电路板通电检查	5分	≥4.5	独立完成电路板通电检查各项内容
			≥4.0	能够完成电路板通电检查各项内容，但有1-2个错误
			≥3.5	不能按照要求的顺序进行，但主要内容能完成
			≥3.0	在提示下完成工作
			＜3.0	在提示下也不能完成工作
5	温度测量与控制	60分	≥54.0	能将参考程序编译、烧写、运行，并可修改部分程序
			≥48.0	能将参考程序编译、烧写、运行，能修改个别语句
			≥42.	仅能使程序编译、烧写、运行，不能修改个别语句
			≥36.0	过程有多次反复，在提示下完成工作
			＜36.0	在提示下也不能完成工作
总成绩		100分		在教学中，评价内容由教师自己掌握

实训报告

一、实训名称

智能温度测量仪的设计

二、实训目的与要求

1. 熟悉温度测量的硬件电路与C51程序。
2. 掌握热敏电阻温度传感器的温度测量方法。
3. 掌握智能温度测量仪的设计方法和应用。

三、实训设备

PC机一台（附带软件：Keil software uVision 2、ISP、RS232/USB转换软件）单片机综合应用实验板一块、热敏电阻、数字万用表一台、频率表一台、220V/5V电源一台、RS232/USB转换信号线、焊接器材若干。

四、实训内容及调试

1. 实训内容

- 单片机综合应用实验板自检测试
- LM331 V/F应用
- 热敏电阻温度传感器的应用
- 温度的测量和显示
- 测量和记录单片机综合应用实验板电源电压
- 测量和记录LM385-25的稳定电压值
- 应用参考程序进行调试直到成功
- 测量和记录LM331芯片的输出频率和热敏电阻的电阻值

2. 程序分析

- 分析上面的测量和记录的内容，并说明原理
- 分析热敏电阻测量温度的原理
- 分析应用软件解决热敏电阻非线性温度特性的方法

五、实训小结

1. 调试过程中所遇到的问题。
2. 问题解决的思路和的方法。
3. 修改程序的效果。
4. 实训收获。

9.4 基于VB的湿度虚拟记录仪设计

学习目标

1. 熟悉应用单片机测量空气湿度的技术、掌握虚拟仪器的设计技术。
2. 了解空气湿度的概念和测量技术。
3. 掌握湿度传感器的应用。
4. 了解应用VB语言设计虚拟仪器的技术。
5. 掌握单片机与计算机的通信技术。

任务实施

一、实训器材

PC机、单片机综合应用实验板、HM1500湿度传感器、数字万用表、频率表、220V/5V电源、220V/30W电烙铁、松香、焊锡丝、导线、罗丝刀、220V电源插座等。

二、知识准备

人类的生存环境除了需要适宜的温度，还需要适宜湿度，即空气的相对湿度。根据有关研究报道，保持室内适宜的相对湿度可防感冒，环境相对湿度过低可使人的呼吸系统抵抗力下降，诱发和加重呼吸系统疾病。一般冬季，居室小气候的最佳组合为：温度18~25°C，相对湿度45~65％RH。这时，人的身体和思维皆处于良好状态，无论工作、休息都可得到较好的效果。

同时，工业生产的自然环境重要指标之一即是湿度，很多工业产品必须在规定湿度下才能生产出来。合适的湿度对提高产品质量、降低废品率、防静电、消除粉尘、净化空气和改善环境均具有决定性的作用。例如：烟草、纺织、仓储、制药、电子产品加工、汽车喷涂线、印刷厂、热力发电厂、炼钢厂、农业、花卉栽植温室等，都需要进行相对湿度的控制，以保证能够正常生产。所以，湿度的测量与控制非常重要，通过本节的学习，能够掌握湿度的测量方法。图9-25所示的是工业生产中的各种除湿设备。

1. 空气的相对湿度与测量

通俗来讲，空气的相对湿度是指空气的潮湿程度，用RH表示。相对湿度的定义是单位体积空气内实际所含的水气密度（用d1 表示）和同温度下饱和水气密度（用d2 表示）的百分比，即RH（%）= d1/ d2×100%。

常用的相对湿度测量方法有四种：

①毛发法

利用脱脂人发（或牛的肠衣）具有空气潮湿时伸长，干燥时缩短的特性，制成毛发湿

度表或湿度自记仪器，它的测湿精度较差。

图 9-25 制药厂与电器设备厂的湿度控制

②干湿球法

用一对并列装置的、形状完全相同的温度表，一支测气温，称干球温度表，另一支包有保持浸透蒸馏水的脱脂纱布，称湿球温度表。湿球因表面蒸发需要消耗热量，从而使湿球温度下降。与此同时，湿球又从流经湿球的空气中不断取得热量补给。当湿球因蒸发而消耗的热量和从周围空气中获得的热量相平衡时，湿球温度就不再继续下降，从而出现一个干湿球温度差。干湿球温度差值的大小，主要与当时的空气湿度有关。空气湿度越小，湿球表面的水分蒸发越快，湿球温度降得越多，干湿球的温差就越大；反之，空气湿度越大，湿球表面的水分蒸发越慢，湿球温度降得越少，干湿球的温差就越小。

③高分子湿度法

高分子湿度法是以高分子聚合物为介质的电容器，因吸收（或释放）水汽而改变电容值。它制作精巧，性能优良，常用在探空仪和遥测中。

④露点法

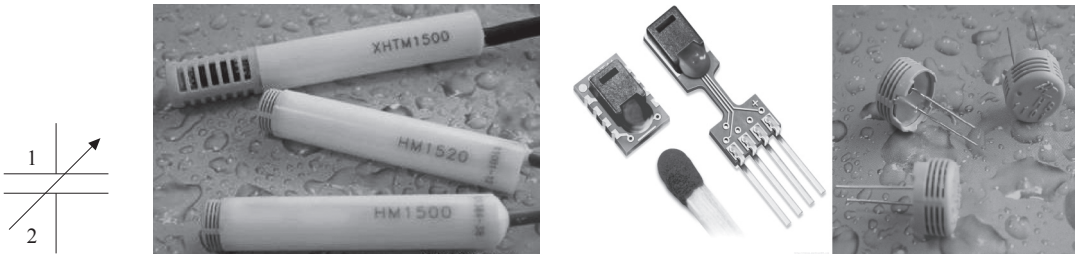
指空气在水汽含量和气压都不改变的条件下，冷却到饱和时的温度。形象地说，就是空气中的水蒸气变为露珠时候的温度叫露点温度。当空气中水汽已达到饱和时，气温与露点温度相同；当水汽未达到饱和时，气温一定高于露点温度。所以露点与气温的差值可以表示空气中的水汽距离饱和的程度。图9-26为各种温湿度仪表。

（a） （b） （c）

图 9-26 各种温湿度仪表

（a）毛发湿度计 （b）干湿球温湿度表 （c）露点温湿度仪

2. 高分子湿度传感器



（a）　　（b）　　（c）

图 9-27 各种高分子湿度传感器

（a）高分子湿度传感器符号　（b）HM1500湿度传感器　（c）其他型号湿度传感器

在前面介绍的四种相对湿度测量方法中，毛发式传感器测量精度差，使用不方便，目前很少使用。干湿球温湿度法和露点温湿度的测量原理复杂，只在特殊需要的场合使用。现在使用比较多的高分子湿度传感器，如图9-27所示。在本例中选择型号为HM1500高分子湿度传感器。HM1500是美国umirel公司于2002年新推出的两种电压输出式集成湿度传感器，其引脚图如图9-28所示。它们的共同特点是先将侧面接触式湿敏电容与湿度信号调理器集成在一个模块中，再封装而成。由于集成度高，因此不需要外围元件，使用非常方便。在它的内部包含由湿敏电容构成的桥式振荡器、低通滤波器和放大器，能输出与相对湿度成线性关系的直流电压信号，输出阻抗为70Ω，适配带ADC的单片机使用。产品的互换性好，抗腐蚀性强。不受水凝结的影响，长期稳定性指标为0.5RH/年。防灰尘，可有效抵抗各种腐蚀性气体物质，在5VDC供电时，0～100％RH对应输出1～4VDC线性电压，温度依赖性非常低。

它的特性有：

①采用专利电容HS1101／HS1101LF设计制造，带防护棒式封装。

②5VDC恒压供电，1-4VDC放大线形电压输出，便于客户使用。

③宽量程：0～100％RH，工作温度范围–30～60℃。

④精度±3％RH（10～95％RH范围）。

⑤抗静电，防灰尘，有效抵抗各种腐蚀性气体物质。

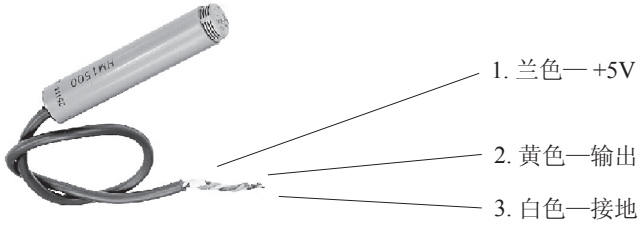


图 9-28 HM1500湿度传感器引脚

3. 湿度的测量

湿度测量原理图如图9-29所示，在图中，IC10、IC9等与HM1500湿度传感器组成湿度测量电路。HM1500把空气的相对湿度转换成0~4V的电压，由LM331转换成0~4000Hz的频率值输入到单片机的14脚，此时单片机的T0作为计数器，T1作为定时器，由单片机计算完成湿度的测量工作。

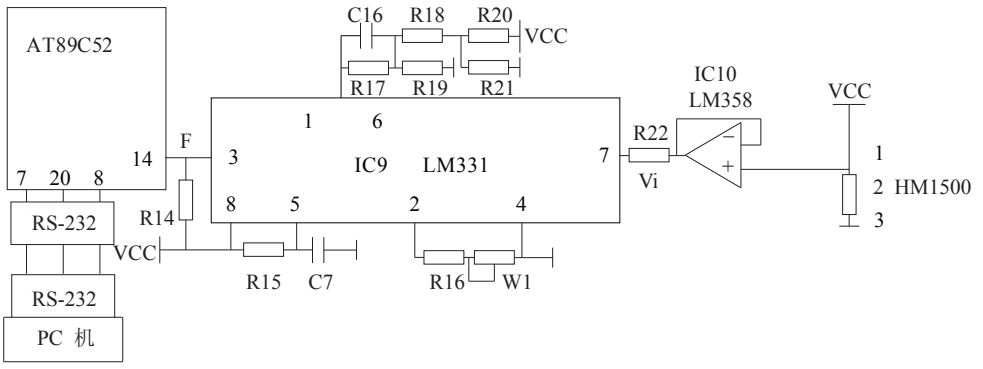


图 9-29 湿度测量原理

在这里，湿度的测量与温度的测量采用了相同的V/F转换的方法。关于湿度的计算，根据图9-30（a）中相对湿度与输出电压之间的关系，用计算公式可以知道相对湿度的值。目前相对湿度的测量精度在±3%左右，一般空气的相对湿度大部分时间在30~90%，相对湿度与对应的输出电压接近线性，如图9-30（b）所示，所以可以用平均法来计算：

设：需要测量的相对湿度为Φ，当前测量的电压为V，频率为P

Φ0=30%　V0= 1860mV　P0=1860Hz

Φ1=90%　V1= 3405mV　P1=3405Hz

Δ% =60　ΔP=3405－1860= 1545Hz　系数为 1545/60=25.75

观察相对湿度与输出电压的关系，实际取系数为24.5。

当：P=2480Hz

则：Φ= (2480-1860) / 24.7 + 30=25 + 30=55%

又　P=2360　Φ = (2360－1860) /24.7 + 30= 20 + 30 = 50%

P=2990　Φ = 50+ (2360－2190) /25.7 = 50－6.6 = 43.4%

测量的电压为V，频率为P，则相对湿度为

$$\Phi=30+(P-1860)/24.7$$

所以在单片机湿度的测量中，编写这样的计算，就方便地得到了空气的相对湿度。

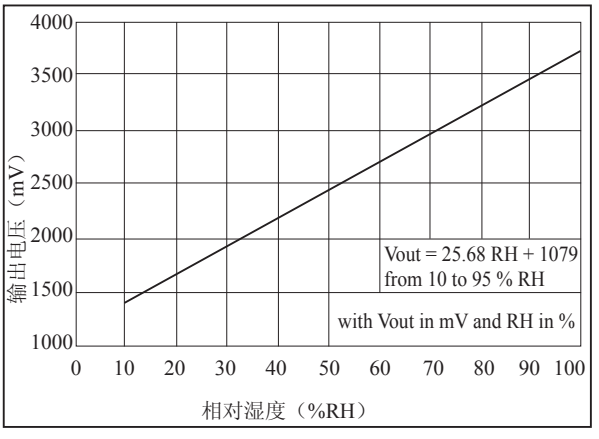
输出值（供参考）

RH(%)	V_{OUT}(mV)	RH(%)	V_{OUT}(mV)
10	1325	55	2480
15	1465	60	2605
20	1600	65	2730
25	1735	70	2860
30	1860	75	2990
25	1990	80	3125
40	2110	85	3260
45	2235	90	3405
50	2360	95	3555

拟合多项式：

$V_{OUT}=9E^{-4}RH^3-1.3E^{-1}RH^2+30.815RH=1030$

（a）



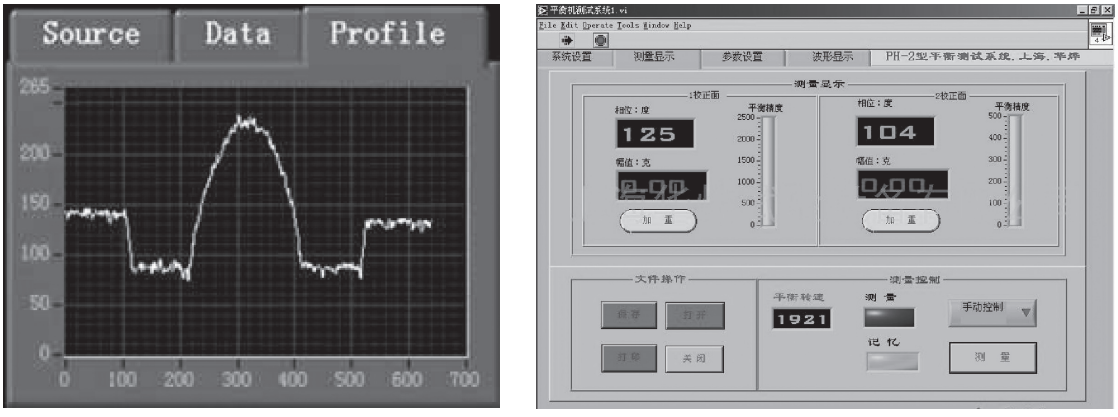
（b）

图 9-30 HM1500高分子湿度传感器特性

（a）相对湿度与输出电压关系　（b）相对湿度与输出电压特性图

4. 虚拟仪器的设计

虚拟仪器（Virtual Instrument）在各种不同的工程应用和行业的测量及控制的用户中广受欢迎，这都归功于其直观化的图形编程语言。虚拟仪器的图形化数据流语言和程序框图能自然地显示数据，同时图形化的用户界面直观地显示数据，使我们能够轻松地查看、修改数据或控制输入。“软件即是仪器”这是美国NI公司提出的虚拟仪器理念的核心思想。从这一思想出发，基于电脑或工作站、软件和I/O部件来构建虚拟仪器。虚拟仪器技术是在PC技术的基础上发展起来的，所以完全“继承”了以现成即用的PC技术为主导的最新商业技术的优点，包括功能超卓的处理器和文件I/O，使您在数据高速导入磁盘的同时就能实时地进行复杂的分析。虚拟仪器见图9-31。



（a）　　（b）

图 9-31　虚拟仪器

（a）显示波形的虚拟仪器　（b）面板式虚拟仪器

虚拟仪器由硬件设备与接口、设备驱动软件和虚拟仪器面板组成。其中，硬件设备与接口可以是各种以PC为基础的内置功能插卡、通用接口总线接口卡、串行口、VXI总线仪器接口等设备，或者是其它各种可程控的外置测试设备，设备驱动软件是直接控制各种硬件接口的驱动程序，虚拟仪器通过底层设备驱动软件与真实的仪器系统进行通讯，并以虚拟仪器面板的形式在计算机屏幕上显示与真实仪器面板操作元素相对应的各种控件，用户用鼠标操作虚拟仪器的面板就如同操作真实仪器一样真实与方便。

虚拟仪器的硬件系统一般分为计算机硬件平台和测控功能硬件。硬件平台可以是各种类型的计算机，如台式计算机、便携式计算机、工作站、嵌入式计算机等。它管理着虚拟仪器的软件资源，是虚拟仪器的硬件基础。因此，计算机技术在显示、存储能力、处理器性能、网络、总线标准等方面的发展促使虚拟仪器系统的快速发展。

虚拟仪器的软件可以是Labview、Visual C++、Visual Basic构成，通过计算机的RS232接口与测量设备的RS232接口连接，不但大大缩短了应用程序的开发周期，还彻底改变了测试软件开发的方式和手段。

5. 关于VB的应用

VB（Visual Basic）语言是一种常用的工具软件，它不需要其他的接口设备就可以直接与单片机系统通信，而且VB的计算功能强大，应用VB设计虚拟仪器的开发成本低，速度快。如图9-30湿度测量原理所示，单片机完成湿度的测量，并将测量的数据通过RS-232接口传送到计算机，计算机接收到数据后，由VB编写的程序以图形、数字、曲线等方式显示

湿度的值。在这里，计算机的屏幕被设计成虚拟的仪器形式，应用计算机的资源，将测量的数据进行存储、处理和打印。

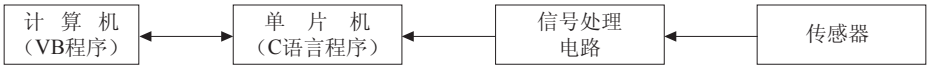


图 9-32 虚拟仪器的结构

虚拟仪器的结构如图9-32所示，传感器将需要测量的信号转换成电压，信号处理电路将传感器的信号转换成单片机能够接收的电压或频率值，并进行计算得到需要的数据传送给计算机，数据的显示、存储等工作由计算机完成。也可以将单片机和信号处理电路集成为一个模块电路，因为在这里单片机只起到信号转换与通信的功能。

三、湿度虚拟记录仪的单片机C51源程序

```
#include <reg52.h>
#define uchar unsigned char
float f;
unsigned char counter,flag;
unsigned char recv_data[2];
unsigned char  wendu_data;
unsigned char flag;
unsigned char i;

unsigned int PL;              //当前测量的频率，是计算的依据
unsigned int PL0;
unsigned int PL1;             //十位显示数
unsigned int PL2;             //个位显示数
unsigned int PL3;             //小数位显示数
unsigned int PL4;
unsigned int xs;              //数码管显示地址和值

sbit P10=P1^0;
sbit P11=P1^1;
sbit P12=P1^2;
sbit P13=P1^3;
sbit P14=P1^4;
sbit P15=P1^5;
sbit P16=P1^6;
sbit P17=P1^7;
sbit P32=P3^2;
sbit P33=P3^3;
```

```
void init_t1(void)              //定时器1初始化，作为波特率发生器
{
  TMOD=0x21;
  TH1=0xfd;
  TL1=0xfd;
  PCON=0x00;
  SCON=0x50;
  TR1=1;
 }
void init_t0()                  //定时器0初始化，设置为计数方式，对外部频率信号进行计数
{
  TMOD=0x25;
  TH0=0x00;
  TL0=0x00;
  TR0=1;
}

void delay(unsigned int i)    // 延时程序，10ms，
{
  unsigned int j,k;
  for(k=0;k<i;k++)
  for (j=0;j<1820;j++);
 }

void send_data(unsigned char temp,unsigned char i)   //每次采样1s
{
  init_t0();
  delay(100);                   //调用延时程序，10ms*100=1s定时
  TR0=0;

  PL0=TL0+TH0*256;
  recv_data[0]=TL0;
  recv_data[1]=TH0;             //读出频率值，存放在recvdata数组中
  TH0=0x00;
  TL0=0x00;
  TR0=1;                        //计数器重置
  SBUF=temp;                    //发送数据的首字节（首字节a/b/c/d由变量temp传递）
  while(TI==0);
  TI=0;
```

```
    SBUF=recv_data[0];              //发送数据的第2个字节
    while(TI==0);
    TI=0;
    SBUF=recv_data[1];              //发送数据的第3个字节
    while(TI==0);
    TI=0;
}

 void main()
{
    unsigned int temp;
    init_t1();                      //定时器1初始化，作为波特率发生器
    init_t0();                      //定时器0初始化，设置为计数方式，对外部频率信号
                                      进行计数

     while(1)
     {
        send_data();
        PL=PL0;

        P17=0;
        PL=10+(PL-1320)/26;
        PL=PL*10;

        PL1=PL/100;                 //百位数
        PL2=(PL-PL1*100)/10;        //十位数
        PL3=PL-PL1*100-PL2*10;      //个位数

        xs=224+PL1;                 //224-百位显示地址，后面为百位显示数据
        P1=xs;                      //由P1口输出显示
        P17=0;
        delay(5);                   //延时时间
        xs=240+PL1;
        P1=xs;                      //显示地址为F，使显示的值保持稳定
        P17=0;
        delay(5);

        xs=208+PL2;                 //208十位显示地址，后面为十位显示数据
        P1=xs;                      //由P1口输出显示
        P17=0;
```

```
        delay(5);                //延时时间
    xs=240+PL2;
    P1=xs;                       //显示地址位F，使显示的值保持稳定
    P17=0;
    delay(5);

    xs=176+PL3;                  //176个数位显示地址，后面为懈鳊位显示数据
    P1=xs;                       //由P1口输出显示
    P17=0;
    delay(5);                    //延时时间
    xs=240+PL3;
    P1=xs;                       //显示地址位F，使显示的值保持稳定
    P17=0;
    delay(5);

    xs=224+PL1;                  //224-百位显示地址，后面为百位显示数据
    P1=xs;                       //由P1口输出显示
    P17=0;
    delay(5);                    //延时时间
    xs=240+PL1;
    P1=xs;                       //显示地址为F，使显示的值保持稳定
    P17=0;
    delay(5);

  }
}
```

四、湿度虚拟记录仪的VB语言程序

编写VB程序首先要进行界面设计，如图9-33所示。

图 9-33　主界面

在主界面上，设置了两个操作按钮，一个是开始测量，它的控件名为commmand1，另一个是测量结束，它的控件名为commmand2。并针对这两个按钮编写相应的程序。

程序如下：

```
Private Sub Command1_Click()
    Form1.Hide      关闭屏幕1
    Form2.Show      开启屏幕2
End Sub
Private Sub Command2_Click()
        End         结束程序运行
 End Sub
```

在第二个界面设计了湿度的显示和湿度变化的曲线，如图9-34所示。

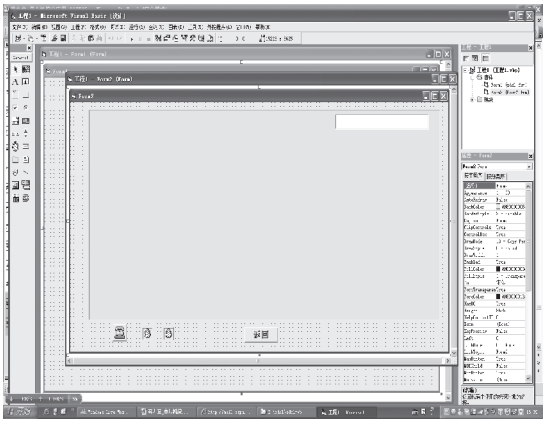


图 9-34 湿度测量显示界面

在第二个界面设计了一个文本框控件text1，用以显示湿度的当前值。设置了一个图形框picture1，用以显示湿度的测量变化曲线，该曲线每6分钟后，重新开始显示。为了控制时间，需要设置一个时钟控件Timer1。更主要的是设置了一个MCScomm控件，是VB实现通讯的控件，MCScomm控件的建立如下：

选择“工程”菜单中的“部件”命令，然后选择Microsoft Comm Contre1 6.0，在工具栏内会添加一个MSComm1控件，如图9-35所示。

图 9-35 MSComm1控件的添加

相应的程序如下：

```
Private Sub Form_Load()
    Call SCIinit(MSComm1, 1, "9600,n,8,1")   '串口控件MSComm1的初始化
    MSComm1.RThreshold = 3          '设置每接收到三个字节，产生oncomm事件
    Timer1.Enabled = True          '时钟2启动，定时1s
    Timer1.Interval = 1000
    Form2.Picture1.Scale (-10, 120)-(300, -5)     '温度曲线
    Form2.Picture1.Cls                  '图形框清0
    Form2.Picture1.DrawStyle = 0            '打印实线
    Form2.Picture1.Line (-5, 0)-(290, 0)          '画X轴
    Form2.Picture1.Line (0, 120)-(0, -10)         '画Y轴
    Form2.Picture1.CurrentX = -6: Form2.Picture1.CurrentY = 100: Form2.Picture1.Print 100
        '打印温度上限值
    Form2.Picture1.CurrentX = -5: Form2.Picture1.CurrentY = 50: Form2.Picture1.Print 50
        '打印温度中心值
    Form2.Picture1.CurrentX = -4: Form2.Picture1.CurrentY = 5: Form2.Picture1.Print 0
        '打印温度中心值
    Form2.Picture1.DrawStyle = 1                        '打印虚线
    Form2.Picture1.Line (0, 100)-(290, 100), QBColor(9)
        '打印温度上限值60的虚直线，有颜色
    Form2.Picture1.DrawStyle = 0                        '打印实线
    Form2.Picture1.Line (0, 50)-(290, 50), QBColor(12)
        '打印温度中心值30的实直线，有颜色
    Form2.Picture1.DrawStyle = 1                        '打印虚线
    Form2.Picture1.DrawStyle = 0                        '打印实线
End Sub

Private Sub MSComm1_OnComm()
    Dim temp As String
    Static aa1, aa2, aa3
    Select Case MSComm1.CommEvent
    Case comEvReceive                   '响应OnComm数据接收事件
    recvdata = MSComm1.Input
            '将接收缓冲区内的数据读出，存放在recvdata数组中
    MSComm1.InBufferCount = 0               '清除接收缓冲区中数据
    MSComm1.RThreshold = 0                  '清除oncomm事件标志位
    temp = Chr$(Trim$(recvdata(0)))
    T2 = Trim$(recvdata(2)) * 256 + Trim$(recvdata(1))
    P = 10 + (T2 - 1320) / 26
    P = Int(P + 0.5)
```

```
        Text1.Text = "当前相对湿度为：" + Str$(P) + " %"
        For i = 1 To 1000
          i = i + 1
          i = i - 1
        Next i
       MSComm1.RThreshold = 3
            '设置每接收到3个字节，产生oncomm事件，为下次接受数据作准备
 s0 = P
t00 = t00 + 1
Form2.Picture1.Line (t00, s0)-(T11, s1)
Form2.Picture1.DrawStyle = 0                          '打印实线
T11 = t00: s1 = s0

If t00 > 280 Then
  t00 = 0: T11 = 0
Form2.Picture1.Cls
 Form2.Picture1.DrawStyle = 0                         '打印实线

 Form2.Picture1.Line (-5, 0)-(300, 0)       '画X轴
 Form2.Picture1.Line (0, 120)-(0, -10)      '画Y轴
 Form2.Picture1.CurrentX = -6: Form2.Picture1.CurrentY = 100: Form2.Picture1.Print 100
              '打印温度上限值
 Form2.Picture1.CurrentX = -5: Form2.Picture1.CurrentY = 50: Form2.Picture1.Print 50
            '打印温度中心值
 Form2.Picture1.CurrentX = -4: Form2.Picture1.CurrentY = 5: Form2.Picture1.Print 0
            '打印温度中心值
 Form2.Picture1.DrawStyle = 1                         '打印虚线
 Form2.Picture1.Line (0, 100)-(290, 100), QBColor(9)
          '打印温度上限值60的虚直线，有颜色
 Form2.Picture1.DrawStyle = 0                         '打印实线
 Form2.Picture1.Line (0, 50)-(300, 50), QBColor(12)
        '打印温度中心值30的实直线，有颜色
 Form2.Picture1.DrawStyle = 1                         '打印虚线
 Form2.Picture1.DrawStyle = 0                         '打印实线

  End If

End Select
End Sub
```

另外还需要编写一个模块：

```
‘Rem 串行口初始化
Public Sub SCIinit(MSComm1 As MSComm, ByVal COMx As Byte, ByVal SCIFormat As String)
  If MSComm1.PortOpen = True Then
    MSComm1.PortOpen = False
  End If
  MSComm1.CommPort = COMx                '串行口号
  MSComm1.Settings = SCIFormat           '定义传输格式
  MSComm1.InputMode = comInputModeBinary     '二进制数据格式
  MSComm1.RThreshold = 0                 '关闭串行中断
  MSComm1.InputLen = 0                   '一次读取缓冲区全部数据
  MSComm1.PortOpen = True                '打开串行口
  MSComm1.InBufferCount = 0

End Sub
```

程序运行的界面如图9-36所示。

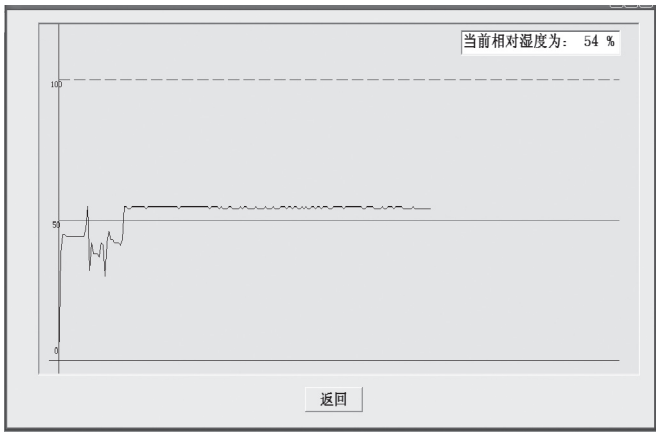


图 9-36 相对湿度测量显示界面

任务评价

序号	操作内容	配分	得 分	评价说明
1	读电路图	25分	≥22.5	能够清晰分析电路中各个元器件的作用与电路工作原理
			≥22.0	能够分析电路中70%元器件的作用与电路工作原理
			≥17.5	对电路中50%的元器件的作用与电路工作原理进行分析
			≥15.0	对电路中30%的元器件的作用与电路工作原理进行分析
			<15.0	达不到上一条款的
2	看电路板	5分	≥4.5	原理图与电路板之间能够清晰地一一对应
			≥4.0	原理图与电路板之间的70%元器件能够一一对应
			≥3.5	原理图与电路板之间的60%元器件能够一一对应
			≥3.0	原理图与电路板之间的40%元器件能够一一对应
			<3.0	达不到上一条款的
3	认识湿度传感器	5分	≥4.5	能够正确使用万用表并测量湿度传感器的湿敏特性
			≥4.0	能够正确使用万用表但测量方法有误
			≥3.5	不能熟练使用万用表且测量方法有误
			≥3.0	在提示下完成工作
			<3.0	在提示下也不能完成工作
4	电路板通电检查	5分	≥4.5	独立完成电路板通电检查各项内容
			≥4.0	能够完成电路板通电检查各项内容，但有1～2个错误
			≥3.5	不能按照要求的顺序进行，但主要内容能完成
			≥3.0	在提示下完成工作
			<3.0	在提示下也不能完成工作
5	湿度的测量	60分	≥54.0	能将参考程序编译、烧写、运行，并可修改部分程序
			≥48.0	能将参考程序编译、烧写、运行，能修改个别语句
			≥42.0	仅能使程序编译、烧写、运行，不能修改个别语句
			≥36.0	过程有多次反复，在提示下完成工作
			<36.0	在提示下也不能完成工作
总成绩		100分		在教学中，评价内容由教师自己掌握

实训报告

一、实训名称

基于VB的湿度虚拟记录仪的设计

二、实训目的与要求

1. 熟悉湿度测量的硬件电路与C语言程序。
2. 掌握湿度传感器的湿度测量方法。
3. 了解VB的性能与应用。
4. 掌握虚拟仪器仪的设计方法和应用。

三、实训设备

PC机一台（附带软件：Keil software uVision 2、ISP、RS232/USB转换软件、VB软件）单片机综合应用实验板一块、湿度传感器、数字万用表一台、频率表一台、220V/5V电源一台、RS232/USB转换信号线、焊接器材若干。

四、实训内容及调试

1. 实验内容

- 单片机综合应用实验板自检测试
- LM331 V/F应用
- 湿度传感器的应用
- 湿度的测量和显示
- 测量和记录单片机综合应用实验板电源电压
- 测量和记录LM385-25的稳定电压值
- 应用参考程序进行调试直到成功
- 测量和记录LM331芯片的输出频率和湿度传感器的电压值
- VB程序的设计

2. 程序分析

- 分析上面的测量和记录的内容，并说明原理
- 分析湿度传感器测量湿度的原理
- 分析应用VB语言设计虚拟仪器的方法

五、实训小结

1. 调试过程中所遇到的问题。
2. 问题解决的思路和的方法。
3. 修改程序的效果。
4. 实训收获。

附 录

附录1 ASCII 字符表

高位 低位	000	001	010	011	100	101	110	111
0000	NUL	DEL	SP	0	@	P	‘	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	“	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	S	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	‘	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LE	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS		<	L	\	l	|
1101	CR	GS		=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	-	o	DeL

附录2　C语言中的关键字

类　型	关键字	意义与用法
数据类型	int	整型变量
	char	字符型变量
	float	实型变量
	double	双精度实型变量
	short	短整型变量
	long	长整型变量
	unsigned	无符号类型变量
	signed	有符号类型变量
	struct	结构体
	union	共用体
	enum	枚举类型
	void	无值型
	volatile	变量可以被改变
	const	常量类型
存储类型	extern	外部变量
	static	静态变量
	register	寄存器变量
	auto	自动变量
	typedef	定义新的数据类型
控制语句	if	if语句
	else	else语句
	for	for语句
	while	while语句
	do	do语句
	goto	无条件转移语句
	switch	switch语句
	case	分支语句
	default	default语句
	return	函数返回语句
	break	break语句
	continue	退出本次循环语句
运算符	sizeof	计算数据长度

附录3　C语言中运算符的优先级和结合性

优先级	运算符	含 义	使用形式	结合方向	说 明
1	( )	圆括号	(表达式)/函数名(形参表)	左到右	
	[ ]	下标运算符	数组名[常量表达式]		
	->	指向结构体成员运算符	对象指针->成员名		
	.	结构体成员运算符	对象.成员名		
2	!	逻辑非运算符	! 表达式	右到左	单目运算符
	～	按位取反运算	~表达式		
	++	增1运算符	++变量名/变量名++		
	－－	减1运算符	－－变量名/变量名－－		
	－	负号运算符	－表达式		
	(类型)	类型转换运算符	(数据类型)表达式		
	*	取值运算符	指针变量		
	&	取地址运算符	&变量名		
	sizeof	长度运算符	sizeof（表达式）		
3	*	乘法运算符	表达式*表达式	左到右	双目运算符
	/	除法运算符	表达式/表达式		
	%	余数（取模）	整型表达式/整型表达式		
4	+	加法运算符	表达式+表达式	左到右	双目运算符
	－	减法运算符	表达式－表达式		
5	«	左移运算符	变量«表达式	左到右	双目运算符
	»	右移运算符	变量»表达式		
6	>	大于运算符	表达式>表达式	左到右	双目运算符
	>=	大于等于运算符	表达式>=表达式		
	<	小于运算符	表达式<表达式		
	<=	小于等于运算符	表达式<=表达式		
7	==	等于运算符	表达式 == 表达式		
	!=	不等于运算符	表达式 != 表达式	左到右	双目运算符
8	&	按位与运算符	表达式 & 表达式	左到右	双目运算符
9	^	按位异或运算符	表达式 ^ 表达式	左到右	双目运算符
10	|	按位或运算符	表达式 | 表达式	左到右	双目运算符
11	&&	逻辑与运算符	表达式 && 表达式	左到右	双目运算符
12	||	逻辑或运算符	表达式 || 表达式	左到右	双目运算符
13	? :	条件运算符	表达式1?表达式2:表达式3	左到右	三目运算符

（续表）

优先级	运算符	含 义	使用形式	结合方向	说 明
14	=	赋值运算符	变量=表达式	右到左	双目运算符
	/=	除后赋值运算符	变量/=表达式		
	*=	乘后赋值运算符	变量*=表达式		
	%=	取模后赋值运算符	变量%=表达式		
	+=	加后赋值运算符	变量+=表达式		
	—=	减后赋值运算符	变量—=表达式		
	<<=	左移后赋值运算符	变量<<=表达式		
	>>=	右移后赋值运算符	变量>>=表达式		
	&=	按位与后赋值运算符	变量&=表达式		
	^=	按位异或后赋值运算符	变量^=表达式		
	|=	按位或后赋值运算符	变量|=表达式		
15	,	逗号运算符	表达式, 表达式, …	左到右	

注：同一优先级的运算符，运算次序由结合方向决定。

参 考 文 献

[1] 李朝青. 单片机原理及接口技术（第3版）[M]. 北京：北京航空航天大学出版社，2005.
[2] 张宏润. 单片机原理及应用（第3版）[M]. -北京：科学出版社，2002.
[3] 求是科技. 单片机典型模块设计实例导航（第2版）[M]. -北京:人民邮电出版社，2008.
[4] 胡伟，季晓衡. 单片机C程序设计及应用实例. 北京：人民邮电出版社，2003.
[5] 谭浩强. C语言程序设计（第二版）[M]. 北京：清华大学出版社，1999.