第 5 章 CHAPTER 5 AT89C51 的中断系统与定时系统

本章介绍 AT89C51 中断系统的基本概念和基础知识，外部中断系统、定时器/计数器的结构与功能，有关特殊功能寄存器的含义与功能，以及外部中断系统、定时器/计数器的初始化编程方法。通过本章的学习，应该达到以下目标。

(1) 理解中断系统的基本概念和基础知识。

(2) 掌握 AT89C51 外部中断系统的结构、功能、控制与使用方法。

(3) 掌握 AT89C51 定时器/计数器的结构、功能、控制与使用方法。

5.1 中断系统介绍

5.1.1 中断的概念

单片机在执行正常程序时，系统中出现了急需处理的异常情况，请求 CPU 迅速去处理，这就是中断。CPU 暂时中止当前正在执行的程序，转而去处理这个异常情况。处理完这个异常情况后，CPU 再回到原来被中断的程序继续执行。CPU 响应中断与中断处理的过程如图 5.1 所示。

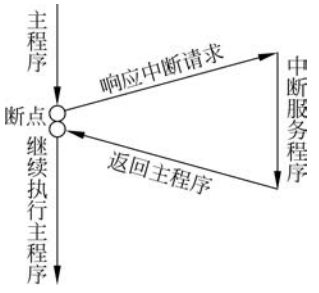

图 5.1 CPU 响应中断与中断处理的过程

单片机中具有中断处理功能的部件，称为中断系统。中断之后所执行的处理程序，称为中断服务子程序，原来执行的程序称为主程序，主程序被中断的位置称为断点，中断申请的来源称为中断源。中断源向 CPU 发出的服务请求，称为中断请求。CPU 对中断请求的处理过程，称为中断处理或中断服务。

单片机的中断系统主要用于实时控制。所谓实时控制，就是要求单片机能够及时响应和处理中断源发出的中断服务请求。

对于中断源发出的中断服务请求，如果没有中断系统，而采用软件查询的方式，那么，CPU 就要定时查询是否有服务请求，不论是否有服务请求，都必须去查询，这将浪费大量的时间。采用中断工作方式，就可以消除 CPU 在查询方式中的等待现象，有助于提高 CPU 的工作效率。

5.1.2 AT89C51 中断系统的结构

AT89C51 有 5 个中断源，分别是两个外部中断、两个定时器/计数器中断、一个串行通信中断。AT89C51 中断系统的结构如图 5.2 所示。

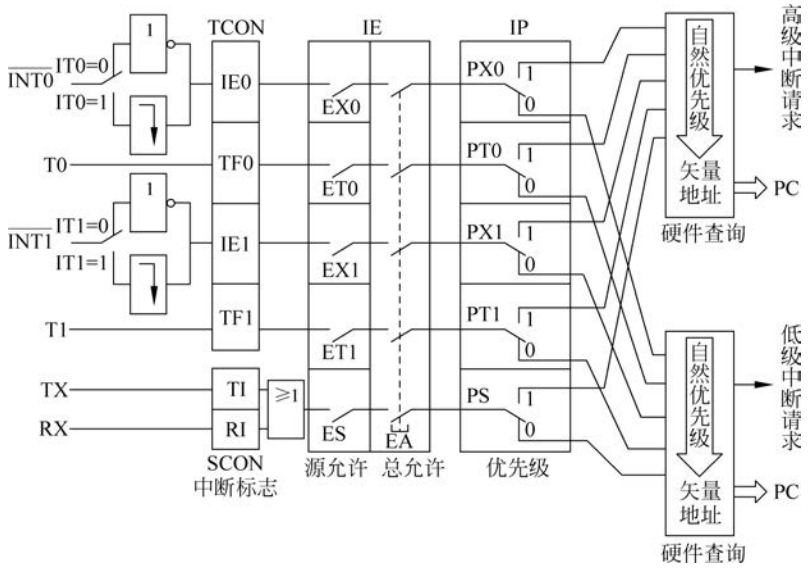

图 5.2 AT89C51 中断系统的结构

中断源的名称、引脚、功能、编号与入口地址如表 5.1 所示。

表 5.1 中断源的名称、引脚、功能、编号与入口地址

名　称	引　脚	功　能	编号	入口地址
外部中断 0	$\overline{\text{INT0}}$(P3.2)	外部中断 0 请求	0	0x0003
T0 中断	T0(P3.4)	T0 溢出中断请求	1	0x000B
外部中断 1	$\overline{\text{INT1}}$(P3.3)	外部中断 1 请求	2	0x0013
T1 中断	T1(P3.5)	T1 溢出中断请求	3	0x001B
串口中断	TX(P3.1)、RX(P3.0)	TX 为发送中断请求，RX 为接收中断请求	4	0x0023

5.1.3 中断控制

AT89C51 通过特殊功能寄存器对中断系统进行控制，所用的特殊功能寄存器分别是定时器/计数器控制寄存器、串口控制寄存器、中断允许寄存器、中断优先级寄存器等。

1. 定时器/计数器控制寄存器(TCON)

TCON 的字节地址为 0x88，可位寻址，位地址为 0x88～0x8F，格式如图 5.3 所示。

位符号	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
位地址	0x8F	0x8E	0x8D	0x8C	0x8B	0x8A	0x89	0x88

图 5.3 TCON 的格式

TCON 中与中断系统有关的各标志位的作用如下。

(1) IT0：外部中断 0 中断请求触发方式选择位。当 IT0＝0 时，采用电平触发方式，$\overline{\text{INT0}}$ 低电平有效；当 IT0＝1 时，采用跳沿触发方式，$\overline{\text{INT0}}$ 负跳变有效。

如果外部中断 0 为电平触发方式，那么，外部中断申请触发器的状态随着 CPU 在每个机器周期采样到的 $\overline{\text{INT0}}$ 引脚电平的变化而变化。在中断服务子程序返回之前，$\overline{\text{INT0}}$ 引脚必须变为高电平，否则，CPU 返回主程序后会再次响应中断。电平触发方式适合于外部中断以电平输入且中断服务子程序能够清除外部中断请求的情况。

如果外部中断 0 为跳沿触发方式，那么，外部中断申请触发器能够锁存 $\overline{\text{INT0}}$ 引脚的负跳变，即使 CPU 暂时不能响应中断，中断申请也不会丢失。当外部中断 0 为跳沿触发方式时，连续两次采样，如果一个机器周期采样到 $\overline{\text{INT0}}$ 引脚为高电平，下一个机器周期采样为低电平，则把中断请求触发器置 1，直到 CPU 响应此中断时，该标志才清 0。此时，输入的负脉冲宽度至少保持一个机器周期，才能保证被 CPU 采样到。跳沿触发方式适合于以负脉冲形式输入的外部中断请求的情况。

(2) IT1：外部中断 1 中断请求触发方式选择位，作用与 IT0 相同。

(3) IE0：外部中断 0 的中断请求标志位。当 CPU 检测到 $\overline{\text{INT0}}$ 的中断请求有效时，硬件自动使 IE0 置 1，向 CPU 申请中断。CPU 响应此中断时，硬件自动使 IE0 清 0，撤销该中断请求。

(4) IE1：外部中断 1 的中断请求标志位，作用与 IE0 相同。

(5) TR0：定时器/计数器 T0 的运行控制位。

(6) TR1：定时器/计数器 T1 的运行控制位。

(7) TF0：T0 溢出标志位。

(8) TF1：T1 溢出标志位。

2. 串口控制寄存器(SCON)

SCON 的字节地址为 0x98，可位寻址，位地址为 0x98～0x9F，格式如图 5.4 所示。

位符号	SM0	SM1	SM2	REN	TB8	RB8	TI	RI
位地址	0x9F	0x9E	0x9D	0x9C	0x9B	0x9A	0x99	0x98

图 5.4　SCON 的格式

SCON 的低 2 位与中断控制有关，作用如下。

(1) RI：串口的接收中断请求标志位。串口接收完一帧串行数据后，硬件自动使 RI 置 1，向 CPU 申请中断。CPU 响应中断时，硬件不会自动对 RI 清 0，必须在中断服务子程序中用软件对 RI 清 0。

(2) TI：串口的发送中断请求标志位。CPU 将 1B 的数据写入发送缓冲器 SBUF 时，就启动一帧串行数据的发送，每发送完一帧数据后，硬件自动使 TI 置 1，向 CPU 申请中断。CPU 响应中断时，不自动对 TI 清 0，必须在中断服务子程序中用软件对 TI 清 0。

SCON 的高 6 位是串行通信控制位，将在第 6 章详细介绍。

3. 中断允许寄存器(IE)

AT89C51 通过中断允许寄存器(IE)对 5 个中断源进行两级控制。所谓两级控制，是指

有一个中断允许总控制位 EA，配合各个中断源的中断允许控制位，共同实现对中断请求的控制。IE 字节地址为 0xA8，可位寻址，位地址为 0xA8～0xAF，格式如图 5.5 所示。

位符号	EA	—	—	ES	ET1	EX1	ET0	EX0
位地址	0xAF	0xAE	0xAD	0xAC	0xAB	0xAA	0xA9	0xA8

图 5.5　IE 的格式

IE 中各位的作用如下。

（1）EX0：外部中断 0 中断允许位。当 EX0＝1 时，允许外部中断 0 中断；当 EX0＝0 时，禁止外部中断 0 中断。

（2）EX1：外部中断 1 中断允许位，作用与 EX0 相同。

（3）ET0：定时器/计数器 T0 的中断允许位。当 ET0＝1 时，允许 T0 中断；当 ET0＝0 时，禁止 T0 中断。

（4）ET1：定时器/计数器 T1 的中断允许位，作用与 ET0 相同。

（5）ES：串口中断允许位。当 ES＝1 时，允许串口的接收和发送中断；当 ES＝0 时，禁止串口中断。

（6）EA：中断允许总控制位。当 EA＝0 时，CPU 关闭所有中断请求；当 EA＝1 时，CPU 开放所有中断源的中断请求，但是，这些中断请求能否被 CPU 响应，还要由 IE 中相应中断源的允许控制位决定。

AT89C51 复位后，IE 清 0，CPU 关闭所有中断请求。因此，在 AT89C51 复位后，必须通过程序中的指令来开放所需的中断。要使某一个中断源被允许中断，除了 IE 相应位置 1 外，还必须使 EA＝1。

例 5.1　初始化 IE，允许片内两个定时器/计数器中断，禁止其他中断源的中断请求。

解　由图 5.5 知，IE 各位的数值为 10001010B，因此，初始化 IE 的语句为“IE＝0x8A;”。

4. 中断优先级寄存器（IP）

AT89C51 中断系统有两个中断优先级，并可实现两级中断的嵌套。所谓两级中断嵌套，是指 AT89C51 在执行低优先级中断服务程序时，可被高优先级中断请求所中断，待高优先级中断处理完毕后，再返回低优先级中断服务程序。两级中断嵌套的过程如图 5.6 所示。

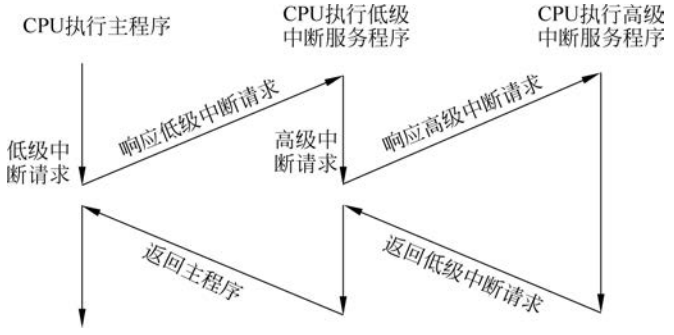

图 5.6　两级中断嵌套的过程

关于中断优先级的关系，有下面两条基本规则。

(1) 低优先级中断服务程序可以被高优先级中断请求所中断，但是，高优先级中断服务程序不能被低优先级中断请求所中断。

(2) 任何一个中断请求，一旦得到响应，就不会被与它同优先级的中断请求所中断。

AT89C51 中断系统有两个不可寻址的“优先级激活触发器”。一个用来指示某高优先级的中断正在执行，所有后来的中断均被阻止。另一个用来指示某低优先级的中断正在执行，所有同级中断都被阻止，但是不阻止高优先级的中断请求。

可以由软件设置每个中断源的中断优先级。中断优先级寄存器(IP)的字节地址为 0xB8，可位寻址，位地址为 0xB8～0xBF，格式如图 5.7 所示。

位符号	—	—	—	PS	PT1	PX1	PT0	PX0
位地址	0xBF	0xBE	0xBD	0xBC	0xBB	0xBA	0xB9	0xB8

图 5.7　IP 的格式

IP 中各位的作用如下。

(1) PX0：外部中断 0 中断优先级控制位。当 PX0＝1 时，$\overline{\text{INT0}}$ 为高优先级中断；当 PX0＝0 时，$\overline{\text{INT0}}$ 为低优先级中断。

(2) PX1：外部中断 1 中断优先级控制位，作用与 PX0 相同。

(3) PT0：定时器/计数器 T0 中断优先级控制位。当 PT0＝1 时，T0 为高优先级中断；当 PT0＝0 时，T0 为低优先级中断。

(4) PT1：定时器/计数器 T1 中断优先级控制位，作用与 PT0 相同。

(5) PS：串口中断优先级控制位。当 PS＝1 时，串口为高优先级中断；当 PS＝0 时，串口为低优先级中断。

AT89C51 复位后，IP 清 0，所有中断源被设置为低优先级中断。

例 5.2　初始化 IP，使两个外部中断请求为高优先级，其他中断请求均为低优先级。

解　由图 5.7 知，IP 各位的值为 00000101B，因此，初始化 IP 的语句为“IP＝0x05；”。

在执行主程序过程中，若只有一个中断源向 CPU 发出中断请求，而这时 CPU 又允许中断，则这个中断请求可以得到响应。当 CPU 同时收到几个不同优先级的中断请求时，先处理高优先级的中断，后处理低优先级的中断。当 CPU 同时收到几个同优先级的中断请求时，CPU 将按照中断的自然优先级顺序确定响应一个中断请求。中断的自然优先级由硬件形成，如表 5.2 所示。

表 5.2　中断的自然优先级

中断源名称	中 断 级 别
外部中断 $\overline{\text{INT0}}$	最高优先级
T0 中断	↓
外部中断 $\overline{\text{INT1}}$	↓
T1 中断	↓
串口中断	最低优先级

例 5.3 假设IP的值被设置为0x06。如果5个中断请求同时发生，试分析CPU响应中断的次序。

解 十六进制数0x06写成二进制数是0000 0110B，由图5.7知，定时器T0和外部中断$\overline{\text{INT1}}$被设置成高优先级中断。

由表5.2知，如果5个中断请求同时发生，那么，CPU响应中断的先后次序为：T0中断→$\overline{\text{INT1}}$中断→$\overline{\text{INT0}}$中断→T1中断→串行中断。

5.2 AT89C51中断处理过程

5.2.1 中断响应的条件

1. CPU查询到中断请求的必要条件

中断响应就是CPU接受并处理中断源提出的中断请求。任何中断源发出的中断请求，只有被CPU查询到，才有可能得到CPU的响应。一个中断请求被CPU查询到的必要条件如下。

(1) 该中断源发出中断请求，即该中断源对应的中断请求标志为1。

(2) 该中断源的中断允许标志位为1。

(3) IE寄存器中的中断总允许位EA=1。

2. CPU丢弃中断请求的情况

一般情况下，当CPU查询到有效的中断请求时，就开始响应中断。需要注意的是，并不是CPU查询到的所有中断请求都能立即得到响应，当遇到下列三种情况之一时，CPU将丢弃中断查询结果，不能对中断进行响应。

(1) CPU正在处理同级或高优先级的中断。

(2) 查询所在的机器周期不是当前正在执行指令的最后一个机器周期。此时，只有在当前指令执行完毕后，才能进行中断响应。

(3) 正在执行的指令是子程序返回指令RET、RETI，或是访问IE、IP的指令。按照AT89C51中断系统的规定，CPU需要再去执行完一条指令，才能响应中断请求。

5.2.2 中断响应后CPU的工作过程

对于汇编语言程序而言，从响应一个中断请求到中断返回，CPU的工作过程如下。

(1) 由硬件自动生成一条长调用指令“LCALL addr16”，这里的addr16是程序存储区中相应中断的入口地址。各中断源服务子程序的入口地址是固定的，如表2.3所示。例如，对于定时器/计数器T0的中断响应，长调用指令为“LCALL 000BH”。

(2) 生成LCALL指令后，紧接着就由CPU执行该指令。首先将PC的内容压入堆栈以保护断点，再将中断入口地址装入PC，使程序转到中断入口地址。由于每个中断源的中断区只有8个单元，一般难以安排一个完整的中断服务子程序，因此，通常在各中断区入口地址处放置一条无条件转移指令，使程序转向存放中断服务子程序的地址。

从中断服务子程序的第一条指令到中断返回指令RETI为止，这个过程称为中断处理或中断服务。中断处理一般包括三部分内容：保护现场、中断源服务和恢复现场。

现场通常包括累加器 A、程序状态字 PSW、工作寄存器 Rn 等。如果在中断服务子程序中要用到这些寄存器，那么，在进入中断源服务之前，应该将它们的内容保护起来。中断源服务结束后，在执行 RETI 指令之前，应该恢复现场。

(3) 中断返回。在汇编语言程序中，中断服务子程序的最后一条指令必须是中断返回指令 RETI。CPU 执行完这条指令后，把响应中断时所保护的断点地址从堆栈中弹出并装入 PC，CPU 就从断点处继续执行原来被中断的程序。

如果使用汇编语言进行单片机应用系统程序设计，程序员需要深刻理解上述过程，并且需要在程序中的中断区入口地址处放置无条件转移指令，保护现场，设计中断服务子程序和恢复现场，难度比较大。如果使用 C 语言进行单片机应用系统程序设计，那么，上述过程中的大部分操作都由编译系统自动完成，程序员只需专注于中断服务子程序设计即可。

5.2.3　中断请求的撤销

1. 外部中断请求的撤销

(1) 跳沿触发方式外部中断请求的撤销。CPU 响应跳沿触发方式外部中断请求后，硬件自动把中断请求标志位 IE0 或 IE1 清 0，而外部中断请求信号当跳沿信号过后也就消失了，因此，跳沿触发方式外部中断请求是自动撤销的。

(2) 电平触发方式外部中断请求的撤销。CPU 响应电平触发方式外部中断请求后，硬件自动把中断请求标志位 IE0 或 IE1 清 0，但是，中断请求信号的低电平可能继续存在，CPU 在以后的机器周期进行采样时，又会把已经清 0 的中断请求标志位 IE0 或 IE1 重新置 1。为了彻底撤销电平触发方式外部中断请求，除了把标志位 IE0 或 IE1 清 0 之外，还要把中断请求信号引脚 $\overline{\text{INT0}}$ 或 $\overline{\text{INT1}}$ 从低电平强制改为高电平。为此，可以在系统中增加电平方式外部中断请求的撤销电路，如图 5.8 所示。

从图 5.8 可见，用 D 触发器锁存外来的中断请求低电平，并通过 D 触发器的输出端 Q 接到 $\overline{\text{INT0}}$ 或 $\overline{\text{INT1}}$，因此，增加的 D 触发器不影响中断请求。中断响应后，为了撤销中断请求，可以利用 D 触发器的直接置 1 端 SD 来实现。例如，把 SD 端接 AT89C51 的 P1.0，只要在 P1.0 输出一个负脉冲，就可以使 D 触发器置 1，撤销低电平的中断请求。

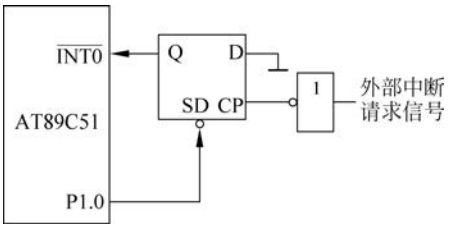

图 5.8　电平方式外部中断请求的撤销电路

所需的负脉冲可在中断服务子程序中通过指令得到。因此，电平触发方式的外部中断请求信号的完全撤销，需要软件、硬件结合来实现。

2. 定时器/计数器中断请求的撤销

CPU 响应定时器/计数器中断请求后，硬件自动把中断请求标志位 TF0 或 TF1 清 0，因此，定时器/计数器中断请求是自动撤销的。

3. 串口中断请求的撤销

CPU 响应串口中断请求后，硬件不对标志位 TI 和 RI 清 0，因为在响应串口中断后，CPU 无法知道是发送中断还是接收中断，还需要测试这两个中断标志位的状态，以判定是

发送操作还是接收操作，然后才能清除。因此，串口中断请求只能用软件撤销。用软件撤销的方法是，在中断服务函数中把串口中断标志位 TI 或 RI 清 0。

5.2.4 采用中断时程序设计的任务

单片机采用中断时，程序设计的基本任务如下。

(1) 设置中断允许寄存器 IE。

(2) 设置中断优先级寄存器 IP。

(3) 对于外部中断，设置中断的触发方式。

(4) 在主函数外，编写中断服务函数，处理中断请求。

前 3 条任务一般放在主函数的初始化程序段中。

例 5.4 假设允许外部中断 0 中断，采用跳沿触发方式，并设定它为高优先级中断，其他中断源为低优先级中断。试编写主函数的初始化程序段。

解 在主函数的初始化部分，编写如下程序段。

```
IE = 0x81;      //IE = 10000001B,中断总允许开,允许 INT0 中断
IT0 = 1;        //INT0 为跳沿触发方式
PX0 = 1;        //INT0 为高优先级中断
```

单片机采用中断时，程序设计的关键是编写中断服务函数。中断服务函数的格式为：

```
void 函数名(void) interrupt n using m
{
  中断服务函数内容
}
```

interrupt n 指出该中断服务函数所对应的中断源的编号，其中，n 的取值如表 5.1 所示。编译器将在对应的中断入口地址处添加跳转指令，跳到本中断服务函数。using m 用于指定本函数所使用的当前工作寄存器，m 的取值为 0～3。该修饰符可以省略，由编译器自动分配。

编写中断服务函数时，应该注意以下几点。

(1) 在中断服务函数的函数名后面，必须加上“interrupt n”，用来指出该中断服务函数所对应的中断源的编号，n 的取值为 0～4。

(2) 中断服务函数不能带参数，否则会导致编译错误。

(3) 中断服务函数没有返回值，函数类型必须为 void。

(4) 中断服务函数因中断源触发而由 CPU 自动调用，不能在程序中直接调用，因此，不必提前声明。

(5) 中断服务函数要简短，避免因执行时间过长而影响 CPU 对其他中断的响应。

(6) CPU 在执行一个低级中断的中断服务函数时，如果有高级中断申请，CPU 应该暂停执行该中断服务函数，而去执行高级中断的中断服务函数，待高级中断的中断服务函数执行完毕后，再回到中断点，接着执行低级中断的中断服务函数。换句话说，中断服务函数可以嵌套执行。但是，这种嵌套执行是由 CPU 决定的，在编写一个中断服务函数时，不能嵌套调用另一个中断服务函数。

5.3　外部中断

5.3.1　外部中断程序设计

单片机的外部中断主要用于解决单片机应用系统正常运行过程中出现的紧急情况，其输入信号来自于单片机的外部。AT89C51 中断系统有两个外部中断 $\overline{\text{INT0}}$ 和 $\overline{\text{INT1}}$，每个中断有两个中断优先级，可以实现两级中断的嵌套。如果一个单片机应用系统只需要一个外部中断，那么，外部中断输入信号可以连接到 $\overline{\text{INT0}}$ 或 $\overline{\text{INT1}}$ 中的一个，并且不需要设置中断优先级。如果一个单片机应用系统需要两个外部中断，那么，两个外部中断输入信号可以分别连接到 $\overline{\text{INT0}}$ 和 $\overline{\text{INT1}}$，并且可以根据实际控制需要设置中断优先级。

一般来说，外部中断编程需要进行下面的设计。

(1) 在主函数的初始化程序段中，通过 TCON 设置外部中断触发方式 IT0、IT1，通过 IP 设置外部中断的优先级，通过 IE 开外部中断 EX0、EX1，开总中断允许 EA。

(2) 在主函数外，编写中断服务函数，实现外部中断的功能。

5.3.2　外部中断应用举例

在 Proteus 仿真平台进行仿真实验时，可以用闸刀开关或按钮开关来模拟外部中断。

闸刀开关具有保持功能。将开关拨动到 ON 时，内部的开关接通，形成通路；若要断开通路，则需要再次拨动开关。比较典型的闸刀开关是指拨开关，如图 5.9 所示。指拨开关可以是一个独立的开关，也可以把几个独立的开关封装起来。一个独立的指拨开关只有两个引脚，在进行电路设计时，把这两个引脚接入电路，即可控制电路的通断。

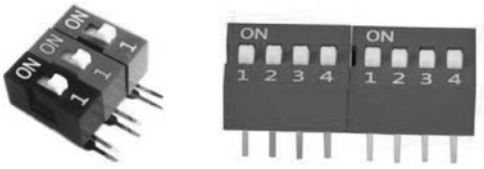

图 5.9　指拨开关

常用的按钮开关是轻触开关，具有自动回弹功能。按下按钮时，开关接通；松开按钮时，开关断开。按钮开关有 4 个引脚，引脚 1 与引脚 4 连通，引脚 2 与引脚 3 连通，如图 5.10 所示。在进行电路设计时，把对角的两个引脚接入电路，即可控制电路的通断。

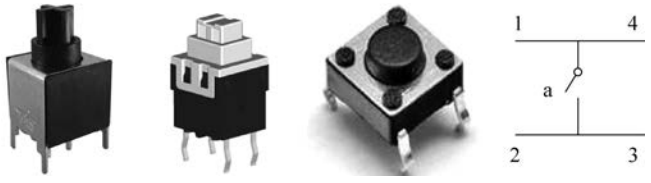

图 5.10　按钮开关

例 5.5　单片机引脚 P1.0 连接一个 LED，引脚 $\overline{\text{INT0}}$(P3.2)连接按键 S1。试编写程序，实现如下功能：每次按键 S1 按下后，改变 LED 的亮灭状态。

分析：从应用系统的功能需求可知，本应用系统用到外部中断 $\overline{\text{INT0}}$。由于只有一个中断源，因此不需要设置中断优先级。

解　(1) 硬件系统设计。外部中断控制 LED 亮灭的电路原理图如图 5.11 所示。

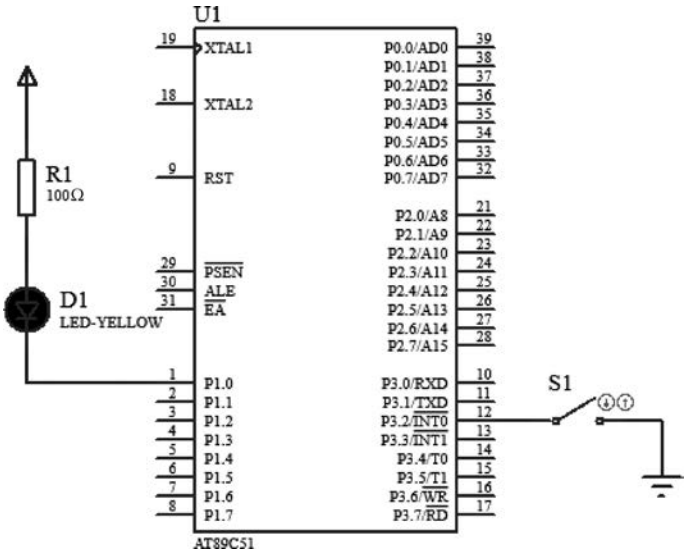

图 5.11　外部中断控制 LED 亮灭的电路原理图

(2) 软件系统设计。控制程序如下。

```
#include <reg51.h>
sbit LED = P1^0;
sbit KEY = P3^2;

/* 外部中断服务函数 */
void int0_ISR(void) interrupt 0
{
  LED = ~LED;
}

main(void)
{
  IT0 = 1;     //跳沿触发方式
  EX0 = 1;     //开 INT0 中断允许
  EA = 1;      //开中断允许总开关
  LED = 0;     //LED 点亮
  while(1);
}
```

说明：

(1) 在主函数的初始化程序段中，设置外部中断 0 为跳沿触发方式，开 $\overline{\text{INT0}}$ 中断允许，开中断允许总开关。这里，通过位操作进行初始化设置。

(2) 程序执行时，在正常情况下，LED 点亮，执行语句“while(1);”，等待外部中断。当按键 S1 按下时，有外部中断请求，程序自动跳到外部中断 0 的外部中断服务函数“void int0_ISR(void)”，改变 LED 的亮灭状态。然后，程序又回到原来被中断的地方，继续执行语句“while(1);”，等待下一次外部中断。

例 5.6　单片机引脚 P1.0、P1.1、P1.2 分别连接一个 LED，名称分别为 D0、D1、D2，引脚 $\overline{\text{INT0}}$(P3.2)上连接按键 S1，引脚 $\overline{\text{INT1}}$(P3.3)上连接按键 S2。试编写程序，实现如下功能：系统正常运行时，D0 点亮；按键 S1 按下时，D0 熄灭，D1 点亮；按键 S2 按下时，D0 熄

灭，D2 点亮；D1 点亮时，按键 S2 按下时，D1 熄灭，D2 点亮；D2 点亮时，按键 S1 按下时，LED 状态不变。

分析：从应用系统的功能需求可知，本应用系统用到两个外部中断 $\overline{\text{INT0}}$ 和 $\overline{\text{INT1}}$，而且 $\overline{\text{INT0}}$ 是低级中断，$\overline{\text{INT1}}$ 是高级中断。

解　（1）硬件系统设计。两个外部中断源的电路原理图如图 5.12 所示。

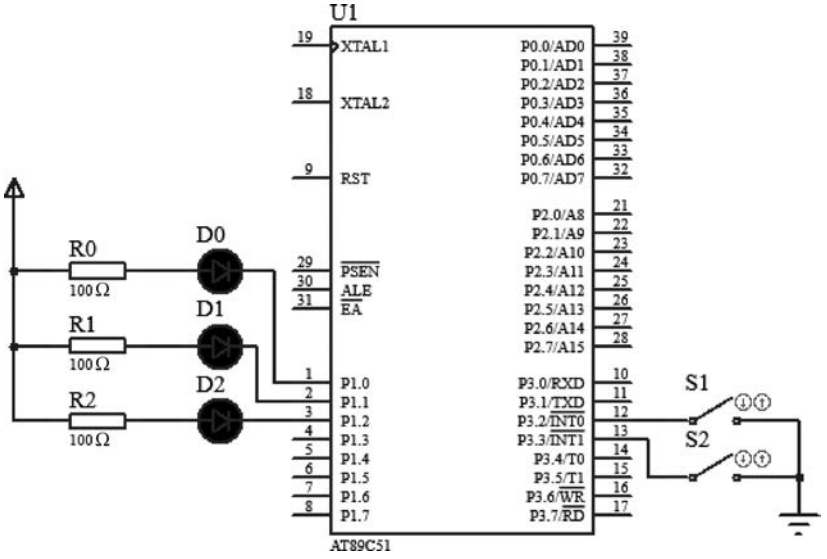

图 5.12　两个外部中断源的电路原理图

（2）软件系统设计。控制程序如下。

```
#include <reg51.h>
#define uint unsigned int      //重新定义关键字,简化程序

sbit LED0 = P1^0;
sbit LED1 = P1^1;
sbit LED2 = P1^2;
sbit KEY1 = P3^2;
sbit KEY2 = P3^3;

/*用二重循环设计的延时函数*/
void delay(uint n)
{
  uint a,b;
  for(a = 0;a < n;a++)
    for(b = 0;b < n;b++);
}

/*外部中断 0 的中断服务函数*/
void int0_ISR (void) interrupt 0
{
  LED0 = 1;                    //D0 熄灭
  LED1 = 0;                    //D1 点亮
  LED2 = 1;                    //D2 熄灭
  delay(500);
```

```
  LED1 = 1;                    //D1 熄灭
}

/* 外部中断 1 的中断服务函数 */
void int1_ISR (void) interrupt 2
{
  LED0 = 1;                    //D0 熄灭
  LED1 = 1;                    //D1 熄灭
  LED2 = 0;                    //D2 点亮
  delay(500);
  LED2 = 1;                    //D2 熄灭
}

int main(void)
{
  TCON = 0x05;                 //INT0、INT1 为跳沿触发方式
  IE = 0x85;                   //开中断允许总开关,开两个外部中断允许开关
  IP = 0x04;                   //INT0 为低级中断,INT1 为高级中断
  while(1)                     //系统正常运行,D0 点亮
  {
    LED0 = 0;
  }
}
```

说明：

(1) 在主函数的初始化程序段中，设置外部中断 $\overline{\text{INT0}}$、$\overline{\text{INT1}}$ 为跳沿触发方式，开 $\overline{\text{INT0}}$、$\overline{\text{INT1}}$ 中断允许，开中断允许总开关，$\overline{\text{INT1}}$ 为高级中断。这里，通过字节操作进行初始化设置，使程序更加简洁。

(2) 程序执行时，在正常情况下，执行语句“while(1)”，D0 点亮，等待外部中断。当按键 S1 按下时，$\overline{\text{INT0}}$ 中断，程序自动跳到外部中断 $\overline{\text{INT0}}$ 的外部中断服务函数 int0_ISR()，D0 熄灭，D1 点亮。当按键 S2 按下时，$\overline{\text{INT1}}$ 中断，程序自动跳到外部中断 $\overline{\text{INT1}}$ 的外部中断服务函数 int1_ISR()，D0 熄灭，D2 点亮。外部中断服务函数执行完成后，程序又回到原来被中断的地方，继续执行语句“while(1)”，系统正常运行时，D0 点亮，等待下一次外部中断。

(3) 当低优先级外部中断 $\overline{\text{INT0}}$ 的中断服务函数 int0_ISR()执行时，高优先级外部中断 $\overline{\text{INT1}}$ 可以中断低优先级外部中断 $\overline{\text{INT0}}$。等到高优先级外部中断 $\overline{\text{INT1}}$ 的中断服务函数 int1_ISR()执行完成后，继续执行低优先级外部中断 $\overline{\text{INT0}}$ 的中断服务函数 int0_ISR()。

(4) 当高优先级外部中断 $\overline{\text{INT1}}$ 的中断服务函数 int1_ISR()执行时，低优先级外部中断 $\overline{\text{INT0}}$ 不能中断高优先级外部中断 $\overline{\text{INT1}}$，需要等高优先级外部中断 $\overline{\text{INT1}}$ 的中断服务函数 int1_ISR()执行完成后，才能执行低优先级外部中断 $\overline{\text{INT0}}$ 的中断服务函数 int0_ISR()。

5.4 定时器/计数器

5.4.1 定时器/计数器的结构

AT89C51 有两个可编程的 16 位定时器/计数器 T0、T1。T0 由特殊功能寄存器 TH0

和 TL0 构成，T1 由特殊功能寄存器 TH1 和 TL1 构成。AT89C51 的定时器/计数器的结构如图 5.13 所示。

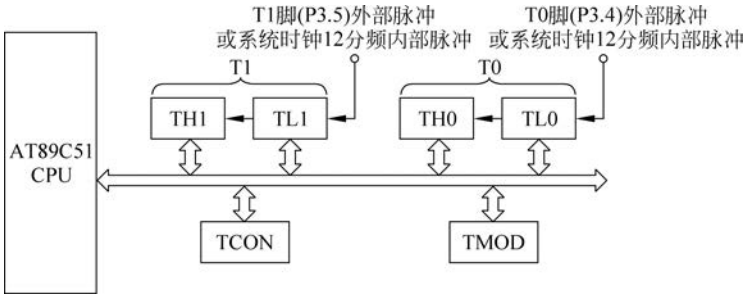

图 5.13　AT89C51 的定时器/计数器的结构

T0、T1 都有定时和计数两种工作模式。定时工作模式对单片机时钟信号经过 12 分频后的脉冲进行计数。由于时钟频率是定值，因此，可以根据计数值计算出时间，从而实现定时的功能。计数工作模式就是对加在 T0 或 T1 引脚上的外部脉冲进行计数。

5.4.2　定时器/计数器的控制

CPU 通过两个特殊功能寄存器 TMOD 和 TCON 对定时器/计数器进行控制。

1. 工作模式寄存器(TMOD)

TMOD 用于选择定时器/计数器的工作模式和工作方式，字节地址为 0x89，不可位寻址。TMOD 的 8 位分为两组，低 4 位控制 T0，高 4 位控制 T1。TMOD 的格式如图 5.14 所示。

位符号	GATE	$C/\overline{T}$	M1	M0	GATE	$C/\overline{T}$	M1	M0

图 5.14　TMOD 的格式

下面以低 4 位为例，说明 TMOD 中各位对 T0 的控制作用。

(1) M1、M0：工作方式选择位。M1、M0 共有四种编码，对应于 T0 的四种工作方式，如表 5.3 所示。

表 5.3　M1、M0 编码与 T0 工作方式的对应关系

M1	M0	T0 的工作方式	说　明
0	0	方式 0	13 位定时器/计数器
0	1	方式 1	16 位定时器/计数器
1	0	方式 2	8 位定时器/计数器，自动重新装载
1	1	方式 3	T0 分成两个 8 位计数器

(2) $C/\overline{T}$：计数模式和定时模式选择位。当 $C/\overline{T}=0$ 时，T0 为定时模式；当 $C/\overline{T}=1$ 时，T0 为计数模式。定时器、计数器的本质都是计数，只是计数对象不同。定时器对内部机器脉冲计数，计数器对外部输入的脉冲计数。

(3) GATE：门控制位。当 GATE=0 时，只要用软件将 TCON 中的 TR0 置 1，就可以启动 T0 工作。当 GATE=1 时，要用软件将 TCON 中的 TR0 置 1，同时，外部中断引脚 $\overline{\text{INT0}}$ 必须为高电平，才能启动 T0 工作。

2. 定时器/计数器控制寄存器(TCON)

TCON 的字节地址为 0x88,可位寻址,位地址为 0x88～0x8F。TCON 的格式如图 5.15 所示。

位符号	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
位地址	0x8F	0x8E	0x8D	0x8C	0x8B	0x8A	0x89	0x88

图 5.15 TCON 的格式

TCON 的低 4 位与外部中断有关,其作用参见 5.1.3 节。TCON 的高 4 位作用如下。

(1) TR0:T0 运行控制位。当 TR0=0 时,T0 停止工作;TR0=1 是启动 T0 的必要条件。

(2) TR1:T1 运行控制位,作用与 TR0 相同。

(3) TF0:T0 溢出标志位。当启动 T0 计数后,T0 从初值开始计数。当计数器 T0 计满溢出时,硬件自动把 TF0 置 1。采用查询方式时,TF0 作为状态位供 CPU 查询。在查询有效后,应该用程序及时将 TF0 清 0。采用中断方式时,TF0 作为中断请求标志位,进入中断服务函数后,由硬件自动将 TF0 清 0,撤销该中断请求。

(4) TF1:T1 溢出标志位,作用与 TF0 相同。

5.4.3 定时器/计数器的工作方式

T0 有 4 种工作方式,分别是工作方式 0、工作方式 1、工作方式 2 和工作方式 3;T1 有 3 种工作方式,分别是工作方式 0、工作方式 1 和工作方式 2。

1. 工作方式 0

下面以 T1 为例加以说明。当 TMOD 中的第 4、5 位 M1M0=00 时,T1 工作在方式 0,为 13 位的定时器/计数器,由 TL1 的低五位与 TH1 构成。T1 工作在方式 0 的逻辑结构如图 5.16 所示。

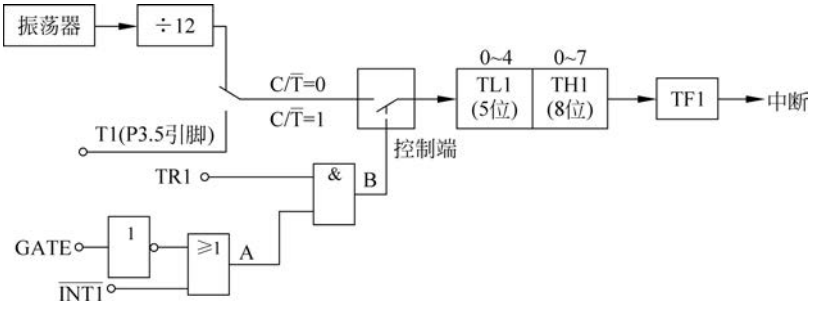

图 5.16 T1 工作在方式 0 的逻辑结构

图 5.16 中,各个控制位的作用如下。

(1) C/$\overline{T}$:工作模式选择位。当 C/$\overline{T}$=0 时,开关打在上面,T1 为定时工作模式,把内部时钟振荡器 12 分频后的脉冲作为计数信号;当 C/$\overline{T}$=1 时,开关打在下面,T1 为计数工作模式,计数脉冲为 P3.5 引脚的外部输入脉冲,当引脚发生负跳变时,计数器加 1。

(2) TR1:T1 运行控制位。

(3) GATE:门控位。当 GATE=0 时,T1 是否计数,仅取决于 TR1 的状态。当

GATE＝1 时，T1 是否计数，取决于 TR1 的状态和 $\overline{INT1}$ 的电平这两个条件。

(4) TF1：计数溢出标志位。当 TL1 低五位计数溢出时，向 TH1 进 1；当 TH1 计数溢出时，溢出标志位 TF1 置 1。

2. 工作方式 1

下面以 T1 为例加以说明。当 TMOD 中的第 4、5 位 M1M0＝01 时，T1 工作在方式 1，为 16 位的定时器/计数器。方式 1 与方式 0 的差别仅在于计数器的位数不同，各个控制位的含义与方式 0 相同。T1 工作在方式 1 的逻辑结构如图 5.17 所示。

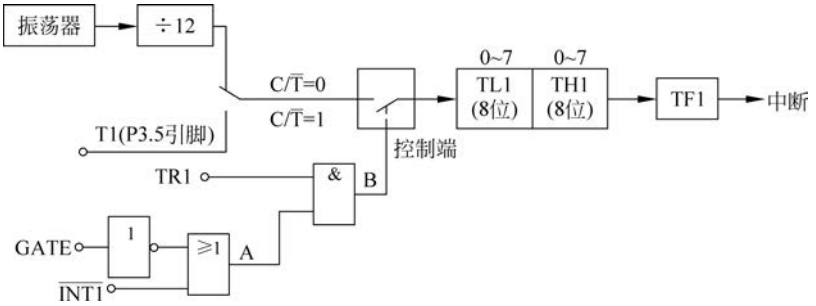

图 5.17　T1 工作在方式 1 的逻辑结构

3. 工作方式 2

下面以 T1 为例加以说明。当 TMOD 中的第 4、5 位 M1M0＝10 时，T1 工作在方式 2，为自动装填初值的 8 位定时器/计数器。T1 工作在方式 2 的逻辑结构如图 5.18 所示。

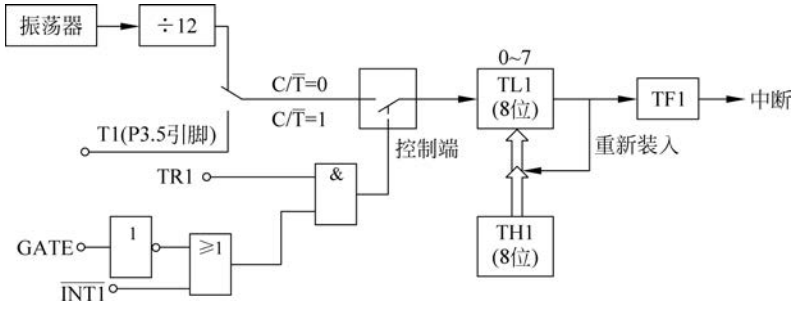

图 5.18　T1 工作在方式 2 的逻辑结构

16 位计数器 T1 被拆成 TL1 和 TH1，TL1 用作 8 位计数器，TH1 用作计数初值寄存器。T1 工作在方式 2 时，编程时必须为 TL1 和 TH1 设置相同的初值。当 T1 启动后，TL1 为 8 位计数器。当 TL1 计数溢出时，硬件把溢出标志 TF1 置 1，同时，自动将 TH1 中的初值送至 TL1，使 TL1 从初值开始重新计数。

定时器/计数器以方式 2 工作，可以省去软件中重装初值的指令，定时精度比较高。但是，工作方式 2 的定时时间短，用作计时器时，最大计数值仅有 $2^8=256$。

4. 工作方式 3

在工作方式 3 下，T0 分为两个独立的 8 位计数器 TL0 和 TH0。TL0 可以作为 8 位定时器或计数器，使用 T0 的所有控制位 GATE、C/$\overline{T}$、M1、M0、TR0、TF0、$\overline{INT0}$(P3.2)、T0(P3.4)等。TL0 计数溢出时，溢出标志 TF0 置 1，TL0 计数初值必须每次由软件设定。TH0 被固定为一个 8 位定时器，不能用作计数器，使用 T1 的状态控制位 TR1 和 TF1。当

TR1＝1 时，允许 TH0 计数，当 TH0 计数溢出时，溢出标志 TF1 置 1。T0 工作在方式 3 的逻辑结构如图 5.19 所示。

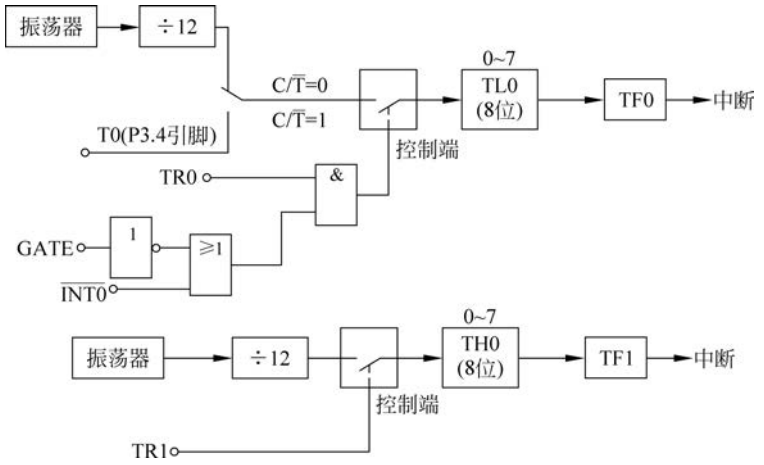

图 5.19　T0 工作在方式 3 的逻辑结构框图

工作方式 3 是为了增加一个 8 位定时器/计数器而设置的，使 AT89C51 具有 3 个定时器/计数器。一般情况下，只有当 T1 用作串口的波特率发生器时，T0 才在需要时选工作方式 3，以增加一个定时器。工作方式 3 只适用于 T0，不适用于 T1。若强行把 T1 设置为方式 3，则 T1 停止计数。

5.4.4　计数器模式下对输入信号的要求

当定时器/计数器工作在计数模式时，计数脉冲来自于外部信号输入引脚 T0 或 T1。当输入信号产生由 1 至 0 的负跳变时，计数器的值加 1。在每个机器周期的 S5P2 期间，CPU 对外部输入引脚进行采样。如果在第一个机器周期中采样值为 1，而在下一个机器周期中采样值为 0，则在再下一个机器周期的 S3P1 期间，计数器加 1。

由于确认一次负跳变需要两个机器周期，即 24 个时钟周期，因此，外部输入的计数脉冲的最高频率只能为系统振荡器频率的 1/24。例如，选用 12MHz 频率的晶体振荡器，则外部输入的计数脉冲的最高频率为 500kHz。

为了确保输入计数脉冲的高电平、低电平在变化之前能被采样一次，两个电平必须分别保持一个机器周期以上。

5.5　定时器/计数器的应用

在定时器/计数器的 4 种工作方式中，方式 0 与方式 1 基本相同，只是计数位数不同。方式 0 是为了兼容 MCS-48 而设计的，计数位数只有 13 位，计时比较短。在实际应用中，一般不用方式 0，而采用方式 1。T0 工作在方式 3 只是为了增加一个定时器/计数器，一般也不需要。因此，下面只介绍定时器/计数器工作方式 1、工作方式 2 的应用。

定时器/计数器开始计数后，从初值开始计数。当定时器/计数器计满溢出时，TF0 或 TF1 被置 1。此时，CPU 可以采用查询方式或中断方式对之进行响应。如果采用查询方

式，那么，在程序运行过程中，CPU 不断地查询 TF0 或 TF1 的值。当 TF0 或 TF1 为 1 时，CPU 进行相应的处理。如果采用中断方式，那么，在程序运行过程中，CPU 不需要查询 TF0 或 TF1 的值。此时，TF0 或 TF1 作为中断请求标志位，当 TF0 或 TF1 为 1 时，CPU 暂停正在运行的程序，转而去运行中断服务函数。相对于查询方式而言，中断方式可以节约 CPU 的时间，从而提高单片机应用系统的工作效率。

5.5.1　定时器/计数器的初始化

1. 初始化的任务

AT89C51 的定时器/计数器是可编程的，在使用定时器/计数器之前必须对它进行初始化。一般情况下，初始化需要做如下工作。

(1) 设置 TMOD，确定定时器/计数器的工作模式与工作方式。

(2) 设置 TH0、TL0 或 TH1、TL1，装填定时器/计数器的计数初值。

(3) 根据需要设置 IE、IP，开启中断，确定中断优先级。

(4) 将 TR0、TR1 置 1 或清 0，启动或禁止定时器/计数器工作。

2. 计数器初值的计算

定时器/计数器工作于计数模式时，必须给计数器一个计数初值。计数器在计数初值的基础上进行加 1 计数，直至溢出。溢出时，T0 或 T1 寄存器被清 0，TF0 或 TF1 被置 1。计数溢出值 M、计数个数 X、计数初值 X_0 三者之间的关系如下：

$$X_0 = M - X \tag{5.1}$$

计数溢出值 M 与计数器的工作方式有关。在方式 0，$M=2^{13}$；在方式 1，$M=2^{16}$；在方式 2 和方式 3，$M=2^8$。

3. 定时器初值的计算

定时器/计数器工作于定时模式时，计数器对单片机晶振频率 f_{osc} 经过 12 分频后的脉冲进行加 1 计数。计数溢出值 M、计数个数 X、计数初值 X_0、定时时间 T、机器周期 T_{cy} 之间的关系如下：

$$T = X \times T_{cy} = (M - X_0) \times T_{cy} \tag{5.2}$$

由此计算出计数初值为：

$$X_0 = M - \frac{T}{T_{cy}} \tag{5.3}$$

5.5.2　定时器/计数器工作方式 1 的应用

例 5.7　单片机外接 12MHz 的晶振，引脚 P1.0 连接一个 LED。试编写程序，实现如下功能：LED 以 2s 为周期闪烁。

分析：LED 以 2s 为周期闪烁，即 LED 点亮 1s，熄灭 1s，循环往复。因此，可以用定时器产生 1s 的定时。这里使用定时器/计数器 T0，工作于定时模式。由于晶振频率为 12MHz，因此，1 个机器周期

$$T_{cy} = 12 \times \frac{1}{f_{osc}}(\mathrm{s}) = \frac{12}{12 \times 10^6}(\mathrm{s}) = 10^{-6}(s) = 1(\mu\mathrm{s})$$

定时器 T0 在工作方式 1 时的最大计数值为 65 536，因此，最长计时为 65 536μs，约为

65ms。如果一次计时取为 50ms,那么,20 次计时就是 1s。

根据公式(5.3),得到定时器 T0 的计数初值为 $X_0=65\,536-50\,000=15\,536$。

解 (1) 硬件系统设计。LED 周期闪烁的电路原理图如图 5.20 所示。

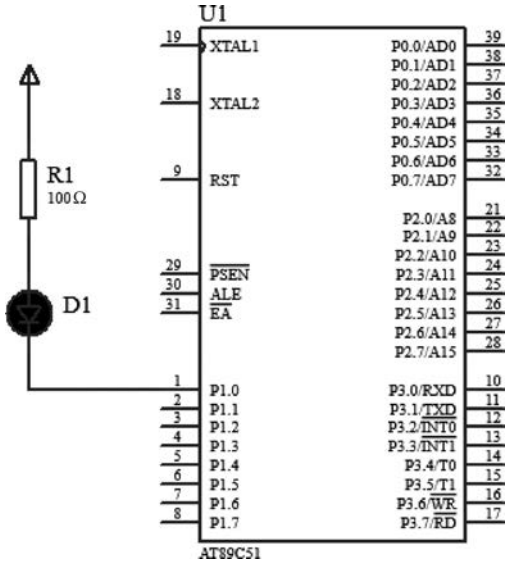

图 5.20 LED 周期闪烁的电路原理图

(2) 采用查询方式,控制程序如下。

```
#include <reg51.h>
sbit LED = P1^0;

main(void )
{
  unsigned char counter;              //用于统计定时器计满溢出的次数
  TMOD = 0x01;                        //T0 工作方式 1,定时模式,由 TR0 控制启停
  TH0 = (65536 - 50000)/256;          //装填 TH0 的初值
  TL0 = (65536 - 50000) % 256;        //装填 TL0 的初值
  TF0 = 0;                            //初始化定时器溢出标志
  LED = 1;                            //关闭 LED
  counter = 0;                        //计时次数从 0 开始
  TR0 = 1;                            //启动定时器 T0
  while(1)
  {
    while(TF0 == 1)                   //查询方式: 定时器溢出
    {
      counter++;                      //计时次数加 1
      if(counter == 20)               //计时次数达到 20,计时达到 1s
      {
        LED = ~LED;                   //LED 取反,使 LED 闪烁
        counter = 0;                  //计时次数重新从 0 开始
      }
      TH0 = (65536 - 50000)/256;      //重新装填 TH0 的初值
      TL0 = (65536 - 50000) % 256;    //重新装填 TL0 的初值
      TF0 = 0;                        //采用查询方式时,需要用程序把 TF0 清 0
    }
```

```
  }
}
```

(3) 采用中断方式,控制程序如下。

```
#include <reg51.h>
sbit LED = P1^0;
unsigned char counter;                  //用于统计定时器中断次数

/* 定时器 T0 中断服务函数 */
void T0_ISR(void) interrupt 1
{
  counter++;                            //中断次数加 1
  TH0 = (65536 - 50000)/256;            //重新装填 TH0 的初值
  TL0 = (65536 - 50000) % 256;          //重新装填 TL0 的初值
}

main(void)
{
  TMOD = 0x01;                          //T0 工作方式 1,定时模式,由 TR0 控制启停
  TH0 = (65536 - 50000)/256;            //装填 TH0 的初值
  TL0 = (65536 - 50000) % 256;          //装填 TL0 的初值
  IE = 0x82;                            //允许 T0 中断
  LED = 1;                              //关闭 LED
  counter = 0;                          //中断次数从 0 开始
  TR0 = 1;                              //启动定时器 T0
  while(1)
  {
    if(counter == 20)                   //中断次数达到 20,计时达到 1s
    {
      LED = ~LED;                       //LED 取反,使 LED 闪烁
      counter = 0;                      //中断次数重新从 0 开始
    }
  }
}
```

说明:

(1) 使用定时器/计数器时,如果采用查询方式,那么,在程序执行过程中,CPU 需要不断查询定时器 T0 溢出标志位 TF0 的值,即在采用查询方式的控制程序中,不断执行循环结构语句“while(TF0==1)”,这将浪费 CPU 的时间,降低 CPU 的效率。如果采用中断方式,那么,在程序执行过程中,CPU 不需要查询 TF0 的值,当 TF0=1 时,系统会自动跳到定时器 T0 中断服务函数,执行完中断服务函数后,再回到中断点继续执行被中断的程序,这样就不会降低 CPU 的效率。

(2) 由于定时器/计数器在工作方式 0 和工作方式 1 时没有自动装载初值功能,因此,定时器/计数器每次溢出后,必须重新装载计数初值。

例 5.8　设计流水灯的电路原理图,编写程序,实现如下功能:使用定时器 T0 中断实现流水灯,流水频率为每 0.5s 更替一次。

解　(1) 硬件系统设计。流水灯的电路原理图如图 5.21 所示,P0 连接 8 个 LED。

(2) 软件系统设计。单片机外接 12MHz 的晶振,定时器/计数器 T0 采用中断方式,控制程序如下。

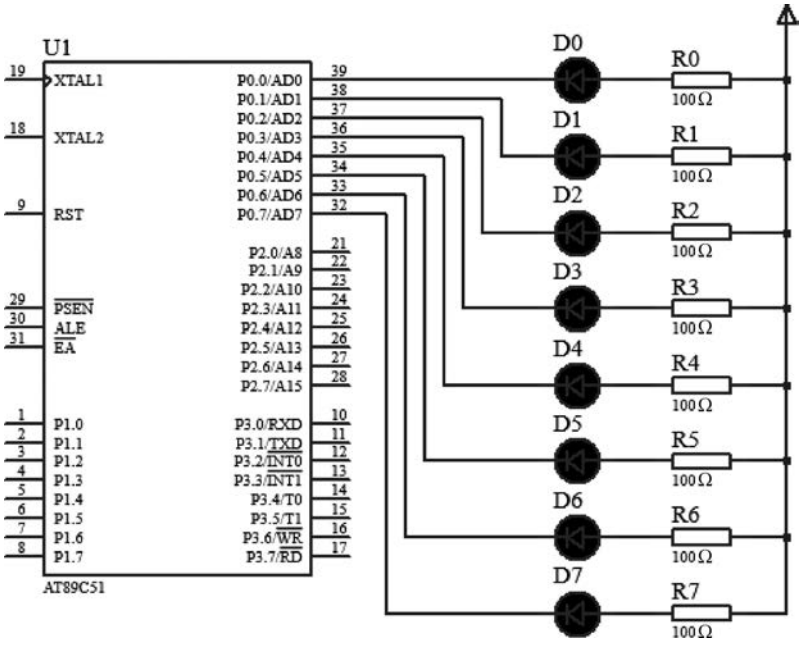

图 5.21　流水灯的电路原理图

```
#include <reg51.h>
unsigned char cnt = 0;                  //用于统计定时器中断次数
unsigned char led  = 0xfe;              //初始化流水灯

int main(void)
{
  TMOD = 0x01;                          //T0 工作方式 1,定时模式,由 TR0 控制启停
  TH0 = (65536 - 50000)/256;            //装填 TH0 的初值
  TL0 = (65536 - 50000) % 256;          //装填 TL0 的初值
  EA = 1;                               //总中断允许开
  ET0 = 1;                              //T0 中断允许开
  TR0 = 1;                              //启动定时器 T0
  while(1);                             //等待中断
}

void T0_ISR(void) interrupt 1
{
  cnt++;
  if(cnt == 10)                         //0.5s 时间到
  {
    cnt = 0;                            //清除中断次数统计
    led = (led << 1)|1;                 //更新流水灯数据
    if(led == 0xff)
    {
      led = 0xfe;
    }
    P0 = led;                           //显示流水灯
  }
  TH0 = (65536 - 50000)/256;            //重新装填 TH0 的初值
  TL0 = (65536 - 50000) % 256;          //重新装填 TL0 的初值
}
```

说明：本例把更新流水灯数据等操作放在定时器 T0 的中断服务函数中，这样处理比较直观，程序容易理解，但是，中断服务函数就需要比较长的时间。如果在执行中断服务函数期间又出现新的低优先级或同优先级的中断，那么，这个新的中断就不能得到及时响应。而在例 5.7 中，定时器 T0 的中断服务函数很简短，因此不会出现这个问题。读者可以参照例 5.7，对本例的程序进行优化。

5.5.3　定时器/计数器工作方式 2 的应用

定时器/计数器工作在方式 0 和方式 1 时，计数溢出后计数器被清 0，当进行循环定时或循环计数时，需要反复装填计数初值，这使得程序设计变得烦琐，并且影响计时精度。而定时器/计数器的工作方式 2 能够自动装填计数初值。

在工作方式 2 下，把 16 位计数器被分为两部分，以 TL0 作为计数器，以 TH0 作为存储器，用于存储计数初值。初始化时，把计数初值分别装填到 TL0 和 TH0 中。当计数溢出时，不需要用程序重新装填计数初值，而是由存储器 TH0 自动给计数器 TL0 装填计数初值。

例 5.9　设计产品包装生产线计数系统，每包有 5 件产品。每个产品经过计数装置时，由机械杆碰合按键 S1 一次。当计满第一包时，指示灯 D0 点亮；当计满第二包时，指示灯 D1 点亮；……；当计满第八包时，指示灯 D0～D7 全亮。重复以上过程。

分析：把单片机的 T1(P3.5)引脚连接一个按键 S1，当每个产品经过计数装置时，由机械杆碰合按键 S1 一次，T1 引脚由高电平变为低电平，产生一个负跳变。因此，可以 T1 作为计数器，对负跳变进行计数。由于一包产品只有 5 个，数量较少，用 8 位二进制数完全可以满足计数需要，因此，使 T1 工作在方式 2。

解　(1) 硬件系统设计。产品包装生产线计数系统如图 5.22 所示，单片机的 T1(P3.5)引脚连接一个按键 S1，P0 连接 8 个 LED，代表指示灯 D0～D7。

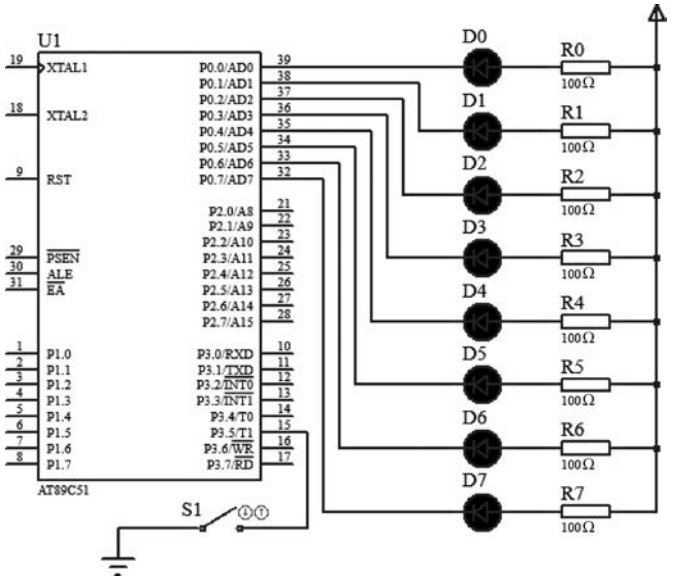

图 5.22　产品包装生产线计数系统

(2) 软件系统设计。定时器/计数器 T1 采用查询方式,控制程序如下。

```
#include <reg51.h>
unsigned char counter;              //统计包的数量

main(void)
{
  TMOD = 0x60;                      //T1 工作方式 2,计数模式,由 TR1 控制启停
  TH1 = 256 - 5;                    //T1 高 8 位赋初值
  TL1 = 256 - 5;                    //T1 低 8 位赋初值
  counter = 0;
  TR1 = 1;                          //启动 T1
  while(1)
  {
    while(TF1 == 1)                 //查询方式: 计满一包
    {
      TF1 = 0;                      //无中断服务程序,需要用程序把 TF1 清 0
      counter++;                    //包的计数加 1
      switch(counter)               //检查包的数量
      {
        case 1: P0 = 0xfe; break;   //1 包满,第 1 个灯亮
        case 2: P0 = 0xfd; break;   //2 包满,第 2 个灯亮
        case 3: P0 = 0xfb; break;   //3 包满,第 3 个灯亮
        case 4: P0 = 0xf7; break;   //4 包满,第 4 个灯亮
        case 5: P0 = 0xef; break;   //5 包满,第 5 个灯亮
        case 6: P0 = 0xdf; break;   //6 包满,第 6 个灯亮
        case 7: P0 = 0xbf; break;   //7 包满,第 7 个灯亮
        case 8: P0 = 0x00; counter = 0; break;  //8 包满,8 个灯全亮,包的数量清 0
      }
    }
  }
}
```

5.5.4 外部中断与定时器/计数器综合应用

例 5.10 设计流水灯的电路原理图,编写控制程序,实现如下功能:通过按键改变流水灯的流动方向,使用定时器控制流水灯的切换时间。

分析:本例综合利用外部中断与定时器/计数器来控制流水灯,以按键作为外部中断信号的输入设备,以定时器/计数器的定时方式进行计时。

解 (1) 硬件系统设计。流水灯的电路原理图,如图 5.23 所示,单片机的 $\overline{\text{INT0}}$(P3.2)引脚连接一个按键 S1,P0 连接 8 个 LED。

(2) 软件系统设计。单片机外接 12MHz 的晶振,定时器/计数器 T0 采用中断方式,控制程序如下。

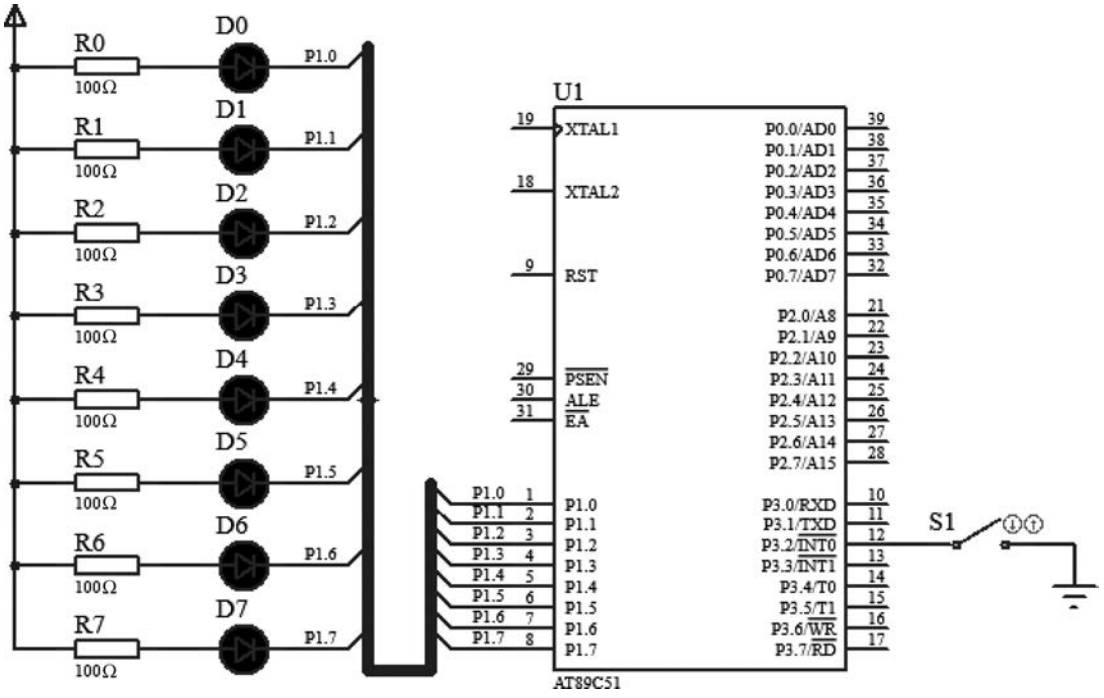

图 5.23　流水灯的电路原理图

```
#include <reg51.h>
unsigned char cnt = 0;                  //用于统计定时器中断次数
bit flag = 0;                           //外部中断标识
unsigned char led = 0xff;               //初始化流水灯

int main(void)
{
  IT0 = 1;                              //外部中断 INT0 跳沿触发方式
  TMOD = 0x01;                          //T0 工作方式 1,定时模式,由 TR0 控制启停
  TH0 = (65536 - 50000)/256;            //装填 TH0 的初值
  TL0 = (65536 - 50000) % 256;          //装填 TL0 的初值
  IE = 0x83;                            //开中断允许
  TR0 = 1;                              //启动 T0 工作
  while(1)
  {
    if(flag == 0)
    {
      if(cnt == 10)                     //0.5s 时间到
      {
        cnt = 0;                        //定时器中断次数清 0
        led = (led << 1)|1;             //更新流水灯数据
        if(led == 0xff)
        {
          led = 0xfe;
```

```
            }
            P1 = led;                    //显示流水灯
          }
        }
        else
        {
          if(cnt == 10)                  //0.5s 时间到
          {
            cnt = 0;                     //定时器中断次数清 0
            led = (led >> 1)|0x80;       //更新流水灯数据
            if(led == 0xff)
            {
              led = 0x7f;
            }
            P1 = led;                    //显示流水灯
          }
        }
      }
    }

    /* INT0 中断服务函数 */
    void int0_ISR(void) interrupt 0
    {
      flag = ~flag;
    }

    /* 定时器 T0 中断服务函数 */
    void T0_ISR(void) interrupt 1
    {
      cnt++;
      TH0 = (65536 - 50000)/256;
      TL0 = (65536 - 50000) % 256;
    }
```

在很多实际应用中，需要各种不同频率、不同波形的数字信号，例如，方波、锯齿波、三角波、脉冲宽度调制（Pulse Width Modulation，PWM）波等。特别是方波，应用最为广泛。例如，在单片机串行通信时，就需要生成一定频率的方波，来控制波特率。可以利用单片机的定时器/计数器生成不同频率的方波。

例 5.11 设计生成不同频率方波的仿真电路，编写控制程序，实现如下功能：使用外部中断方式控制按键，通过按键控制单片机的某个引脚分别输出频率为 100Hz、1kHz 的方波。

分析：所谓方波，就是在一个周期内，高电平和低电平各占一半的波形。因此，只需在单片机的某个引脚输出半个周期的高电平，再输出半个周期的低电平，如此循环往复，就可以得到所需的方波。

解　(1) 硬件系统设计。产生不同频率方波的电路原理图如图 5.24 所示,单片机 $\overline{\text{INT1}}$(P3.3)引脚连接一个按键 S1,P2.0 连接一个示波器。

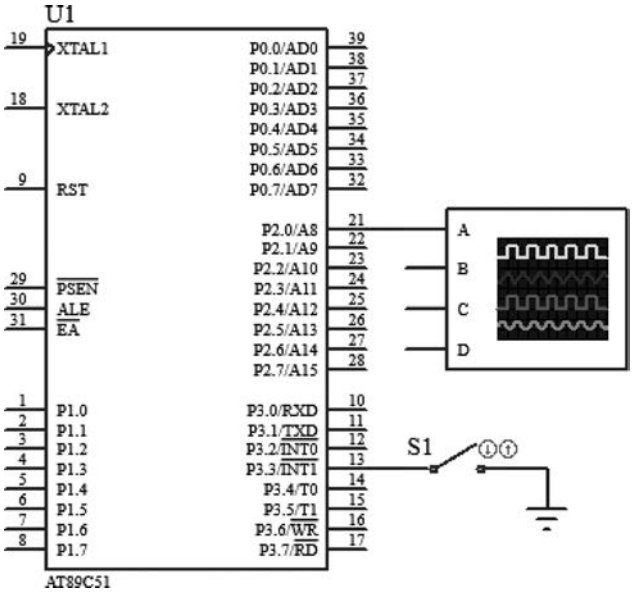

图 5.24　产生不同频率方波的电路原理图

(2) 软件系统设计。单片机外接 12MHz 的晶振,定时器/计数器 T0 采用中断方式,控制程序如下。

```
#include <reg51.h>
sbit P2_0 = P2^0;
unsigned int cnt = 0;                    //用于统计定时器中断次数
bit flag = 0;                            //外部中断标识

int main(void)
{
  IT1 = 1;                               //外部中断 INT1,跳沿触发方式
  TMOD = 0x02;                           //T0 工作方式 2,定时模式,由 TR0 控制启停
  TH0 = 256 - 250;                       //装填 TH0 的初值
  TL0 = 256 - 250;                       //装填 TL0 的初值
  IE = 0x86;                             //开中断允许
  TR0 = 1;                               //启动 T0 工作
  while(1)
  {
    if(flag == 0)                        //产生 100Hz 的方波
    {
      if(cnt == 20)                      //0.005s 时间到
      {
        cnt = 0;                         //定时器中断次数清 0
        P2_0 = ~P2_0;
      }
    }
    else                                 //产生 1kHz 的方波
    {
      if(cnt == 2)                       //0.0005s 时间到
```

```
        {
          cnt = 0;                    //定时器中断次数清 0
          P2_0 = ~P2_0;
        }
      }
    }
}

/* INT1 中断服务函数 */
void int1_ISR(void) interrupt 2
{
  flag = ~flag;
}

/* 定时器 T0 中断服务函数 */
void T0_ISR(void) interrupt 1       //定时器 T0 中断服务函数
{
  cnt++;
}
```

说明：

(1) 一般来说，在使用方波作为控制信号时，要求信号的频率参数尽可能精确，即要求信号的高电平部分和低电平部分的时间尽可能精确。因此，可以使用定时器/计数器的工作方式 2，自动装填初值，提高计时的精确度。本例就是使用定时器/计数器的工作方式 2。

(2) 对于频率比较低的方波，半个周期的时间比较长，而定时器/计数器的工作方式 2 一次计时时间很短，需要定时器中断很多次，而程序需要不断查询定时器中断次数变量，这也会浪费很多时间，影响计时精度。此时，可以考虑使用定时器/计数器的工作方式 1，由于一次中断的时间比较长，定时器只需中断较少的次数就可以达到半个周期的时间。

习题

一、选择题

1. 各中断源发出的中断请求信号，都会标记在 AT89C51 的_____中。

   A. IE 寄存器　　B. TMOD 寄存器

   C. IP 寄存器　　D. TCON 与 SCON 寄存器

2. 在 AT89C51 的中断源中，需要外加电路实现中断撤销的是_____。

   A. 电平触发方式的外部中断　　B. 串行中断

   C. 跳沿触发方式的外部中断　　D. 定时中断

3. 当外部中断 0 发出中断请求后，中断响应的必要条件是_____。

   A. ET0＝1　　B. EX0＝1　　C. IE＝0x81　　D. IE＝0x61

4. AT89C51 的中断系统有两个中断优先级，各中断源的优先级设定是利用特殊功能寄存器_____。

   A. IE　　B. IP　　C. TCON　　D. SCON

5. 下列说法中，错误的是_____。

   A. 同一级别的中断请求按时间的先后顺序响应

B. 同一时间同一级别的多中断请求将形成阻塞，系统无法响应

C. 低优先级中断请求不能中断高优先级中断服务程序

D. 同级中断不能嵌套

6. 单片机默认的最高等级中断源是_____。

A. 定时器 T0　　B. 定时器 T1　　C. 外部中断 $\overline{\text{INT0}}$　D. 外部中断 $\overline{\text{INT1}}$

7. AT89C51 在同一优先级的中断源同时申请中断时，CPU 首先响应_____。

A. $\overline{\text{INT0}}$ 中断　　B. $\overline{\text{INT1}}$ 中断　　C. T0 中断　　D. T1 中断

8. 如果将中断优先级寄在器 IP 设置为 0x0A，则优先级最高的是_____。

A. $\overline{\text{INT1}}$　　B. $\overline{\text{INT0}}$　　C. T1　　D. T0

9. 在 AT89C51 中，与外部中断无关的特殊功能寄存器是_____。

A. TCON　　B. TMOD　　C. IE　　D. IP

10. 当外部中断请求为跳沿触发方式时，要求中断请求信号的高电平状态和低电平状态都应至少维持_____个机器周期。

A. 1　　B. 2　　C. 4　　D. 12

11. AT89C51 的外部中断 $\overline{\text{INT1}}$ 的中断请求标志是_____。

A. ETI　　B. TEI　　C. IT1　　D. IE1

12. 在 AT89C51 中，与定时器/计数器中断无关的特殊功能寄存器是_____。

A. TCON　　B. TMOD　　C. SCON　　D. IP

13. 溢出后不用重装计数初值，定时器/计数器的工作方式为_____。

A. 方式 0　　B. 方式 1　　C. 方式 2　　D. 方式 3

14. 若 TMOD 中的 M1M0 为 11，则设置定时器/计数器工作于_____。

A. 方式 0　　B. 方式 1　　C. 方式 2　　D. 方式 3

15. 使 AT89C51 的定时器 T0 停止计数的语句是_____。

A. TR0＝1；　　B. TR0＝0；　　C. TR1＝1；　　D. TR1＝0；

16. 若要求最大定时时间为 2^{16} 个机器周期，则应使定时器/计数器工作于_____。

A. 方式 0　　B. 方式 1　　C. 方式 2　　D. 方式 3

17. 晶振频率为 12MHz 的单片机，在定时工作模式下，定时器可能实现的最大定时时间为_____。

A. 4096μs　　B. 8192μs　　C. 65 536μs　　D. 32 768μs

18. 设定时器/计数器 T0 的初始化程序段如下：

```
TMOD = 0x06;
TH0 = 0xFF;
TL0 = 0xFF;
EA = 1;
ET0 = 1;
```

则执行该程序段后，把定时器/计数器 T0 的工作状态设置为_____。

A. 工作方式 0，定时应用，定时时间为 2μs，中断禁止

B. 工作方式 1，计数应用，计数值为 255，中断允许

C. 工作方式 2，定时应用，定时时间为 510μs，中断禁止

D. 工作方式 2，计数应用，计数值为 1，中断允许

19. 单片机定时器/计数器根据需要有4种工作方式，其中工作方式1是_____。

A. 16位的定时器/计数器　　B. 13位的定时器/计数器

C. 8位可自动重载的定时器/计数器　　D. 两个独立的8位定时器/计数器

20. 应用单片机定时器/计数器时，控制T0的启动和停止的关键字是_____。

A. TMOD　　B. TR0　　C. ET0　　D. TF0

21. AT89C51外接6MHz的晶振，采用16位定时器计时50ms，则设置定时器的计数初值为_____。

A. 0　　B. 65 536　　C. 50 000　　D. 40 536

22. 若AT89C51的振荡频率为6MHz，设定时器工作在方式0需要定时1ms，则定时器初值应为_____。

A. 500　　B. 1000　　C. 7692　　D. 7192

23. 定时器/计数器T1工作在计数模式时，外加的计数脉冲信号应连接到_____引脚。

A. P3.2　　B. P3.3　　C. P3.4　　D. P3.5

24. AT89C51的定时器/计数器T0用于定时，采用工作方式1，则初始化编程为_____。

A. TMOD=0x01　　B. TMOD=0x50　　C. TMOD=0x10　　D. TCON=0x02

25. AT89C51内部有_____个16位的定时器/计数器，其中，定时器/计数器T1有_____种工作方式。

A. 4,5　　B. 2,4　　C. 5,2　　D. 2,3

26. AT89C51的定时器/计数器T1用作计数时的计数脉冲由_____提供。

A. P3.5引脚　　B. 内部时钟频率

C. P3.4引脚　　D. 内部时钟频率12分频

二、填空题

1. AT89C51有5个中断源，有_____个优先级。控制中断允许的特殊功能寄存器是_____，控制中断优先级的特殊功能寄存器是_____。

2. 外部中断1的中断入口地址为_____，定时器1的中断入口地址为_____。

3. AT89C51外部中断请求信号有电平触发方式和_____。在电平触发方式下，当采集到$\overline{\text{INT0}}$、$\overline{\text{INT1}}$为_____时，激活外部中断。

4. 要将外部中断$\overline{\text{INT0}}$设置为电平触发方式，则应将_____位设置成_____。要将外部中断$\overline{\text{INT1}}$设置成跳沿触发方式，则应将_____位设置为_____。

5. AT89C51有两个定时器/计数器，它们是由_____、_____、_____和_____4个专用的特殊寄存器构成的。

6. 当定时器/计数器T0申请中断时，T0的计满溢出标志位TF0为_____，当中断得到CPU响应后，TF0为_____。

7. 若IP=00010100B，则优先级最高者为_____，最低者为_____。

8. 定时和计数都是对_____进行计数，定时与计数的区别是_____。

9. 当定时器/计数器T0工作在计数模式时，计数脉冲来自于_____引脚。

10. 当_____时，定时器/计数器发出中断请求。

11. 设定时器/计数器 T0 工作在计数模式，工作方式 2，TR0 控制启停；定时器/计数器 T1 工作在计时模式，工作方式 1，TR1 控制启停，那么，TMOD=_____。

12. 定时器/计数器的工作方式 3 是指将_____拆成两个独立的 8 位计数器。另一个定时器/计数器 T1 此时通常只用作_____。

13. 设单片机晶振频率为 12MHz，利用定时器 T0 定时 200μs，那么，采用工作方式 0 时的计数初值为_____，采用工作方式 1 时的计数初值为_____，采用工作方式 2 时的计数初值为_____。

14. 设单片机晶振频率为 6MHz，利用定时器 T0 定时。在工作方式 0 下，最大定时时间是_____μs；在工作方式 1 下，最大定时时间是_____μs；在工作方式 2 下，最大定时时间是_____μs；在工作方式 3 下，最大定时时间是_____μs。

三、判断题

1. 定时器/计数器控制寄存器 TCON 只是控制定时器/计数器的。（　　）

2. 设置中断允许寄存器 IE=0x05，外部中断 $\overline{\text{INT0}}$、$\overline{\text{INT1}}$ 能够正常中断。（　　）

3. 低优先级中断服务程序可以被高优先级中断请求所中断，但是，高优先级中断服务程序不能被低优先级中断请求所中断。（　　）

4. 一个中断请求得到响应后，不会被与它同优先级的中断请求所中断。（　　）

5. 外部中断 $\overline{\text{INT1}}$ 的中断编号是 1。（　　）

6. 通过 TMOD，可以设置定时器/计数器的 4 种工作方式。（　　）

7. 定时器/计数器都是对输入脉冲进行计数的。（　　）

8. 当 TMOD 中的 GATE=1 时，表示由两个信号控制定时器/计数器的启停。（　　）

四、简答题

1. 试叙述中断的全过程。

2. 中断优先级控制的原则是什么？

3. 说明中断服务函数与普通函数的不同之处。

4. 定时器/计数器用作定时器时，其计数脉冲由谁提供？定时时间与哪些因素有关？

5. 定时器/计数器用作计数器模式时，对外界输入脉冲的频率有何限制？

6. 定时器/计数器的工作方式 2 有什么特点？适用于哪些应用场合？

7. 如果采用的晶振的频率为 3MHz，定时器/计数器工作在方式 0、1、2 下，其最大定时时间各为多少？

8. 用定时器/计数器测量某正单脉冲的宽度，采用何种方式可得到最大量程？若时钟频率为 6MHz，求允许测量的最大脉冲宽度是多少？

9. 说明 T0 计满溢出标志位 TF0 是怎么置 1 与清 0 的？

10. AT89C51 在使用定时器/计数器之前必须对它进行初始化，初始化的任务是什么？

五、论述题

1. 一个中断请求被 CPU 响应的条件是什么？

2. 下面几种中断从高到低优先顺序的安排是否可行？如果可行，写出中断优先级寄存器 IP 的初始化值。如果不可行，请说明理由。

(1) T0，T1，$\overline{\text{INT0}}$，$\overline{\text{INT1}}$，串行中断。

(2) 串行中断，$\overline{\text{INT0}}$，T0，$\overline{\text{INT1}}$，T1。

(3) $\overline{\text{INT0}}$,T1,$\overline{\text{INT1}}$,T0,串行中断。

(4) $\overline{\text{INT0}}$,$\overline{\text{INT1}}$,串行中断,T0,T1。

(5) 串行中断,T0,$\overline{\text{INT0}}$,$\overline{\text{INT1}}$,T1。

(6) $\overline{\text{INT0}}$,$\overline{\text{INT1}}$,T0,串行中断,T1。

(7) $\overline{\text{INT0}}$,T1,T0,$\overline{\text{INT1}}$,串行中断。

3. 设单片机晶振频率为6MHz,利用定时器/计数器T0定时。在工作方式1下,最大定时时间约为130ms。现在需要定时1min,可以怎么实现?

4. 工作模式寄存器TMOD的低4位各位的作用是什么?怎么设置?

5. 定时器/计数器控制寄存器TCON在定时应用中起什么作用?怎么设置?

6. AT89C51用于中断允许控制的寄存器是什么?写出中断允许控制寄存器各位的符号及含义。中断允许控制寄存器是怎么控制中断允许的?

六、程序设计题

1. 外部中断$\overline{\text{INT1}}$为跳沿触发、高优先级的中断。试编写$\overline{\text{INT1}}$的中断初始化程序段。

2. 设单片机晶振频率为6MHz,定时器/计数器T1工作在计时模式,工作方式2,TR1控制启停,产生200μs定时,采用中断方式。试编写T1的中断初始化程序段。

3. 设单片机晶振频率为12MHz,单片机$\overline{\text{INT0}}$(P3.2)引脚连接一个按键S1,P0连接8个LED。编写控制程序,实现如下功能:通过按键S1改变流水灯的流动方向;使用软件延时方式控制流水灯的切换时间。

4. 设单片机晶振频率为12MHz,利用定时器/计数器T0中断,采用工作方式1定时,在P2.0输出频率为1Hz的方波。

5. 设单片机晶振频率为12MHz,利用定时器/计数器T0中断,采用工作方式2定时,在P2.0输出周期为500μs,占空比为4∶1的矩形波。