中断控制应用

【知识目标】

- 了解单片机系统中断系统的概念
- 学会与外部中断系统有关的寄存器的功能

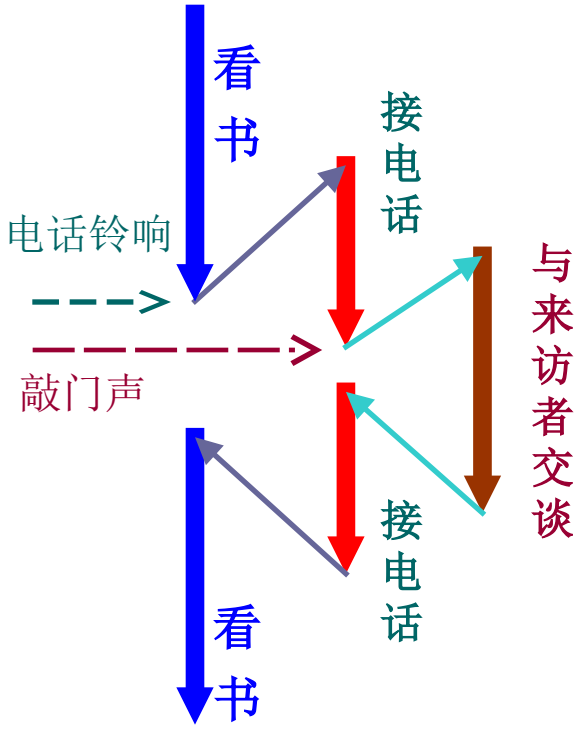
【能力目标】

- 掌握与外部中断系统有关的寄存器的设置方法
- 掌握中断服务子程序的结构及基本编程方法
- 掌握简单中断应用系统的程序编写、调试方法

任务一 项目知识点学习

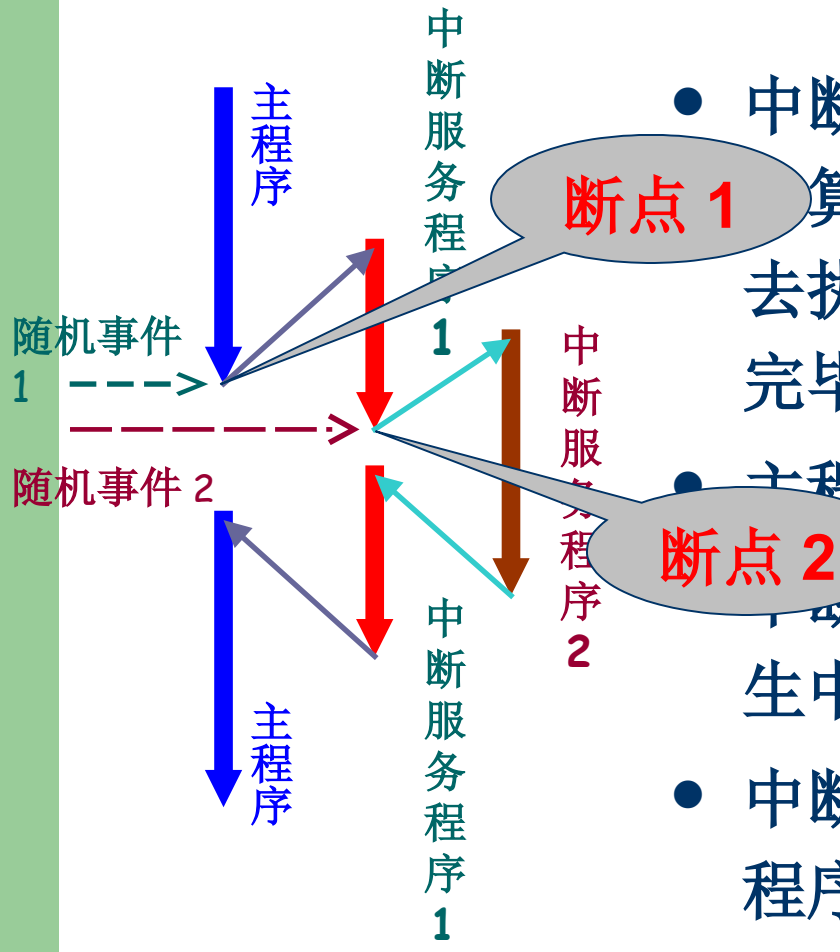
一、中断的基本概念

日常生活中的中断



你在看书，电话铃响，于是你在书上做上记号，去接电话，与对方通话；门铃响了，有人敲门，你让打电话的对方稍等一下，你去开门，并在门旁与来访者交谈，谈话结束，关好门；回到电话机旁，继续通话，接完电话后再回来从做记号的地方接着看书。

计算机中的中断概念

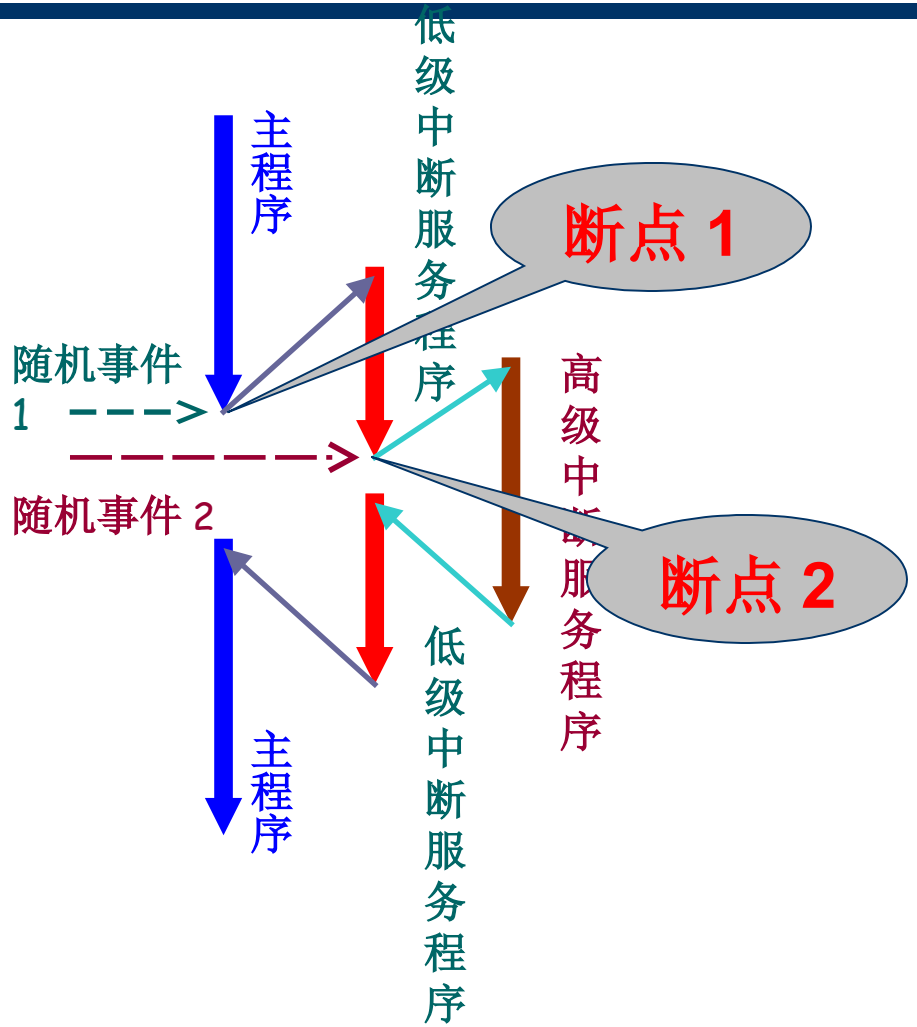


- 中断——由于某一随机事件的发生，计算机 CPU 暂停原程序的运行，转去执行另一程序（随机事件），处理完毕后又自动返回原程序继续运行。
- 主程序——计算机现行运行程序。
- 中断源——引起中断的原因，或能发生中断申请的来源。
- 中断服务子程序——处理随机事件的程序。

采用中断的优点

- 分时操作。
- 实时控制。
- 故障处理。

中断的优先级及嵌套



二、中断源和中断向量

- 1、中断源

  中断源是指能够向单片机发出中断请求信号的部件和设备。AVR 单片机具有丰富的中断源，ATmega16 单片机有 21 个中断源，如表 4-1 所示。

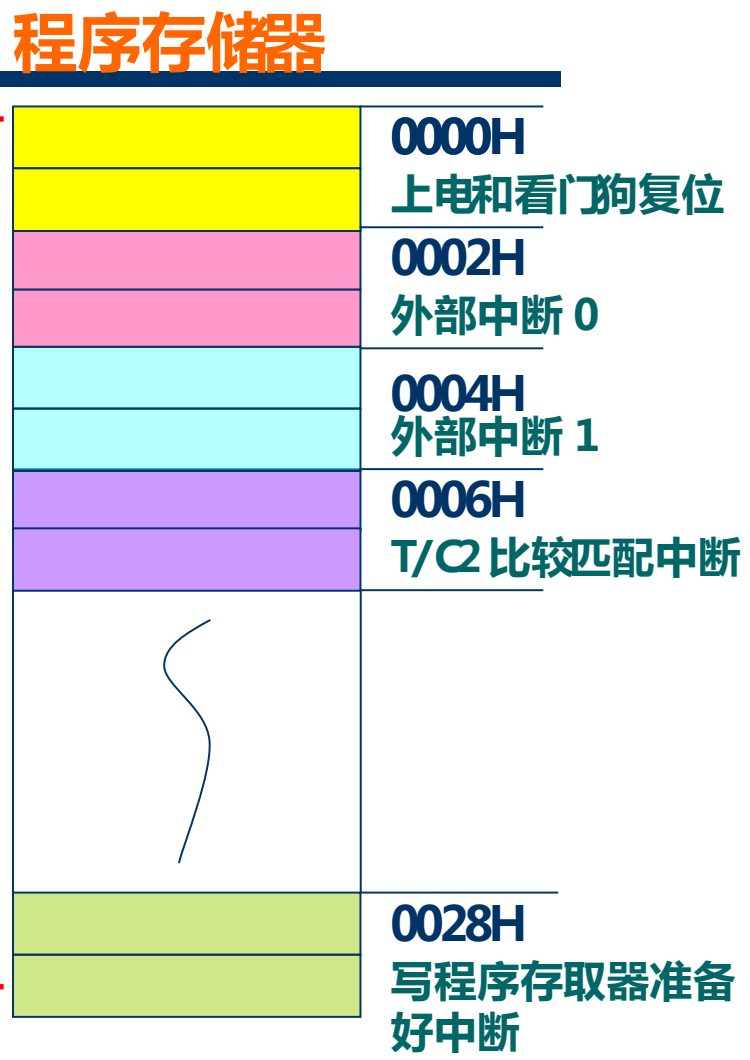
- 2、中断向量

  中断源发出的请求信号被 CPU 检测到之后，如果单片机的中断控制系统允许响应中断，则 CPU 会自动转移，执行一个固定的程序空间地址中的指令。这个固定的地址称为中断入口地址，也称中断向量。中断入口地址通常是由单片机内部硬件决定的。ATmega16 单片机的中断向量如 4-1 所示。

中断服务程序的转入

- 中断服务子程序入口地址也称为**中断向量**或**中断矢量**。
- 单片机中的中断入口地址是**固定**的，不能改动。
- 单片机中的中断源不同中断服务程序的入口地址也不同。
- RESET 是系统复位中断，为**非**

使用时，通常在这些入口地址处存放一条跳转指令，使程序跳转到用户安排的中断服务程序起始地址上去！



三、中断控制与响应过程

- 1、中断的优先级

  AVR 单片机中，一个中断在中断向量区中的位置决定了其优先级，位于低地址的中断优先级高于高地址的中断。对于 ATmega16 单片机，复位中断 **RESET 具有最高优先级**，外部中断 INT0 其次， **SPM_RDY(保存程序存储器内容就绪) 的中断优先级最低**。

三、中断控制与响应过程

- 2、中断管理及中断标志

AVR 有两种不同的中断：带**有中断标志位的中断**和**不带中断标志的位的中断**。（低电平触发的外部中断）。

AVR 对中断采用两级控制方式，有一个**总的中断允许控制位**（SREG 中的 I 标志位 SREG..7），同时每一个中断源都设置了**独立的中断允许控制位**（在各中断源所属模块的控制寄存器中）。

三、中断控制与响应过程

- 3、中断嵌套

由于 AVR 在响应一个中断的过程中，通过硬件自动将 **I 标志位清 0**，这样就阻止了 MCU 响应其它的中断，因此，在通常情况下 AVR 是不能实现中断嵌套的，如果系统中必须要有中断嵌套的应用，可以在**中断服务程序中使用指令将全局中断允许位打开**，以间接的方式实现中断的嵌套。

三、中断控制与响应过程

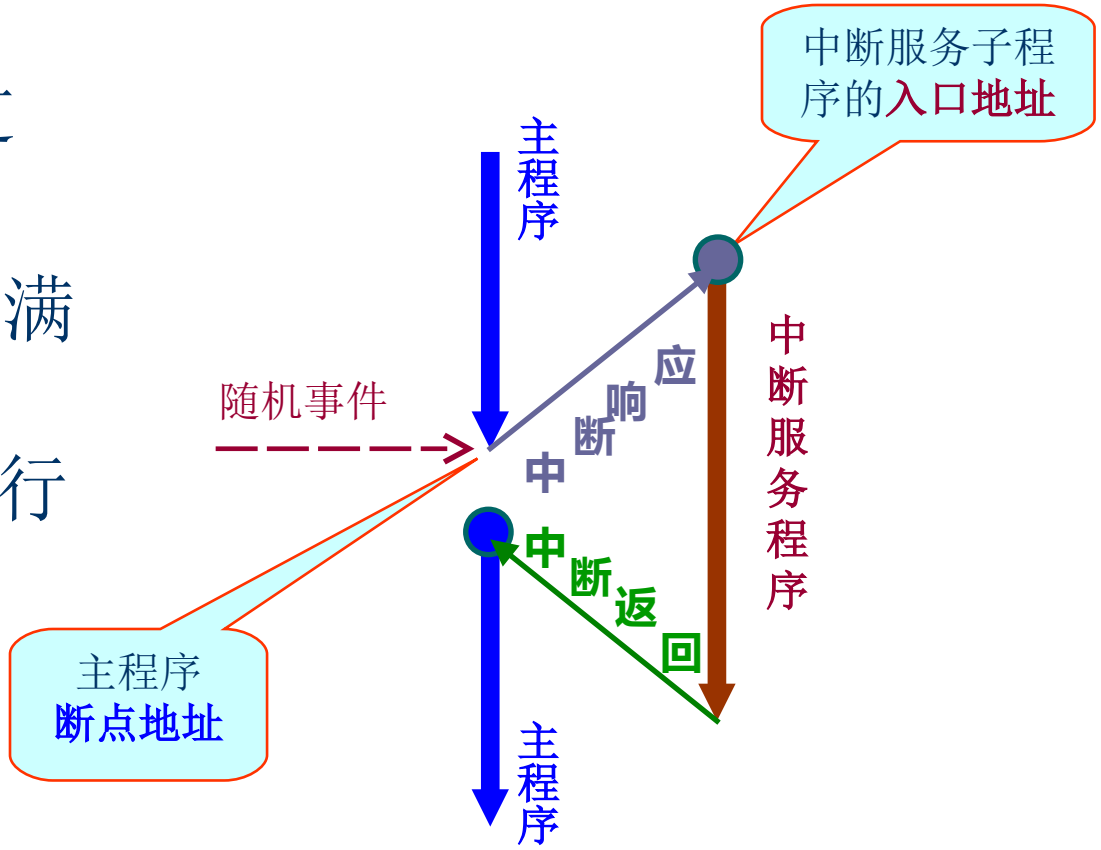
- 3、中断嵌套

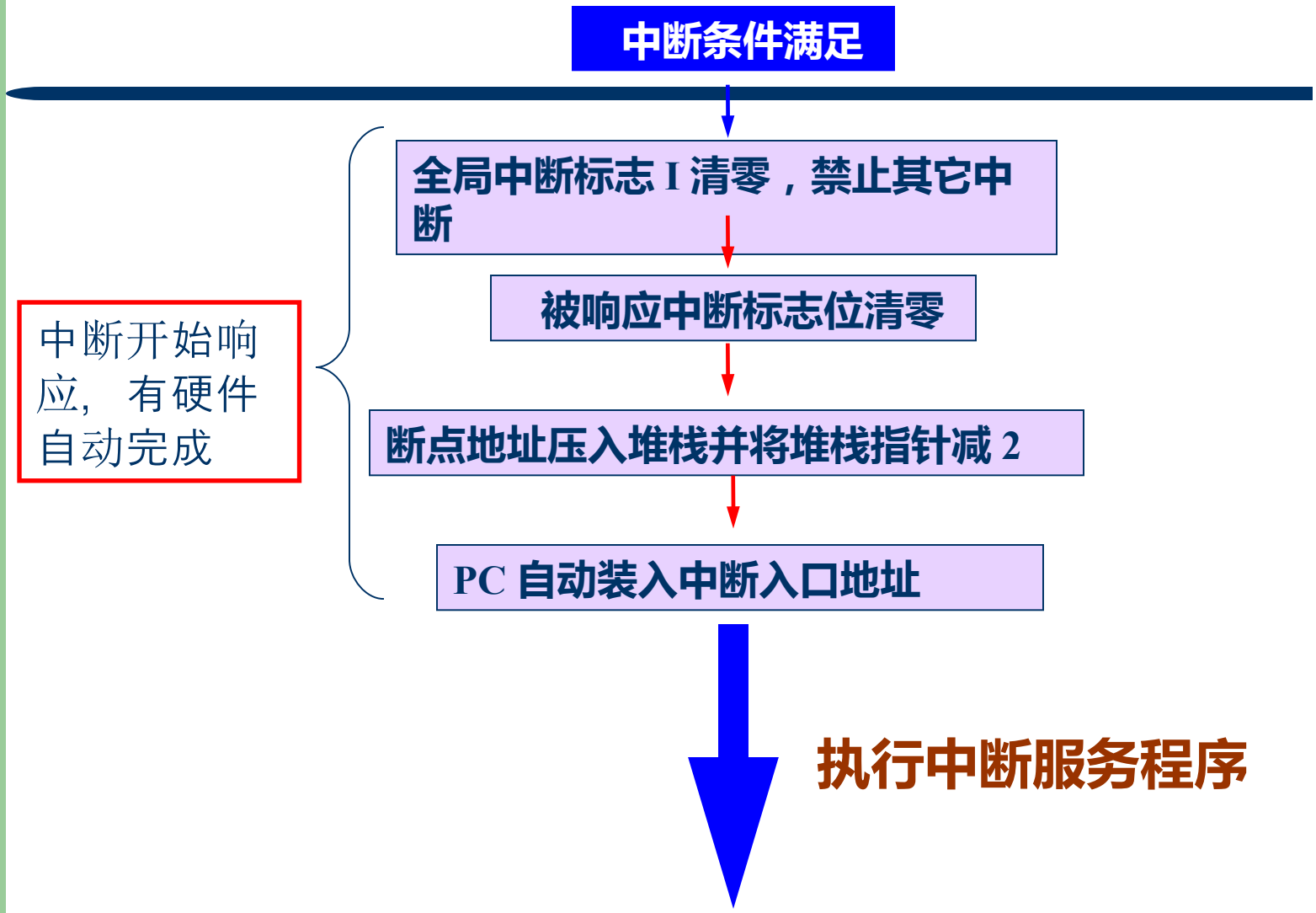
由于 AVR 在响应一个中断的过程中，通过硬件自动将 **I 标志位清 0**，这样就阻止了 MCU 响应其它的中断，因此，在通常情况下 AVR 是不能实现中断嵌套的，如果系统中必须要有中断嵌套的应用，可以在**中断服务程序中使用指令将全局中断允许位打开**，以间接的方式实现中断的嵌套。

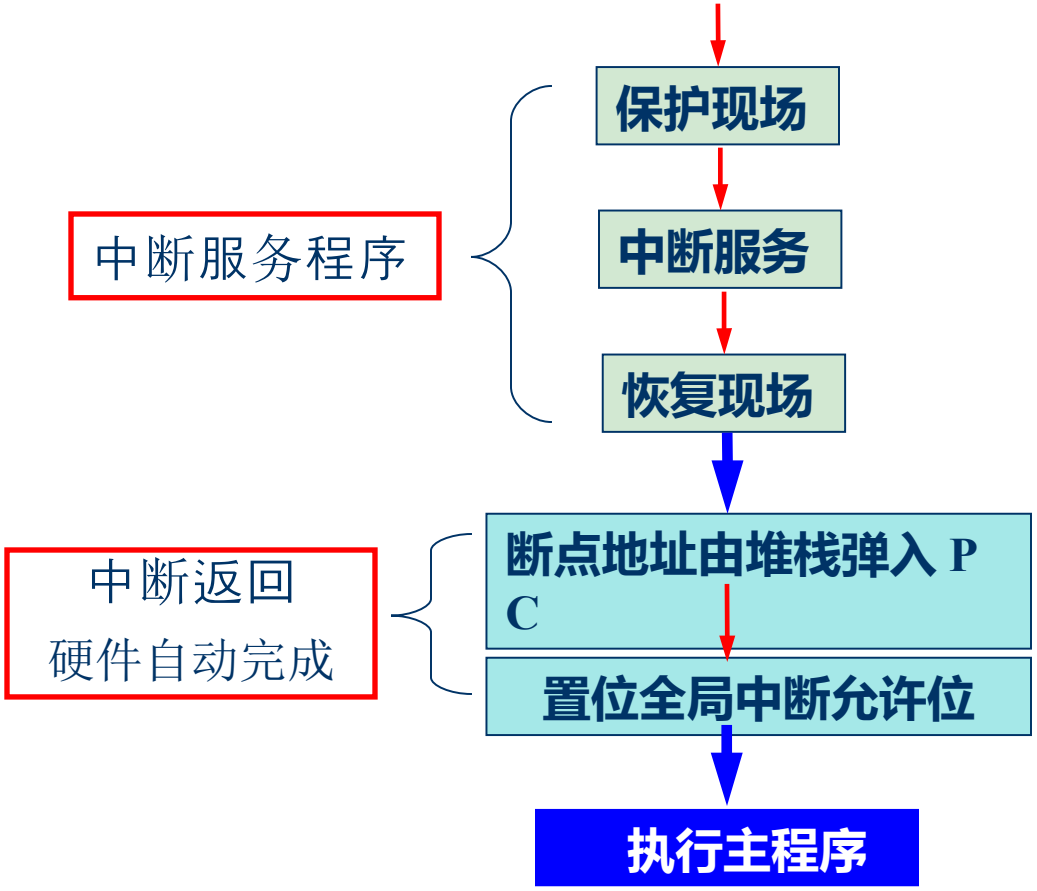
三、中断控制与响应过程

- 4、中断响应过程

  当一个中断满足响应条件后，MCU 便可以执行中断响应。




中断条件满足
全局中断标志 I 清零，禁止其它中断
被响应中断标志位清零
断点地址压入堆栈并将堆栈指针减 2
PC 自动装入中断入口地址
中断开始响应，有硬件自动完成
执行中断服务程序


保护现场
中断服务程序
中断服务
恢复现场
断点地址由堆栈弹入 PC
中断返回
硬件自动完成
置位全局中断允许位
执行主程序

四、 GCCAVR 高级语言下中断服务程序编写

- 在高级语言的开发环境中，都扩展和提供了相应的编写中断服务程序的方法，通常不必考虑中断现场保护和回复的处理，因为编译器在编译中断服务程序代码时，会在生成的目标代码中自动加入相应的中断现场保护和回复的指令。

在本书使用的 WinAVR 版本下，中断服务例程格式为：

```
ISR( 中断向量名称 )
{
// 中断发生后要执行的语句
}
```

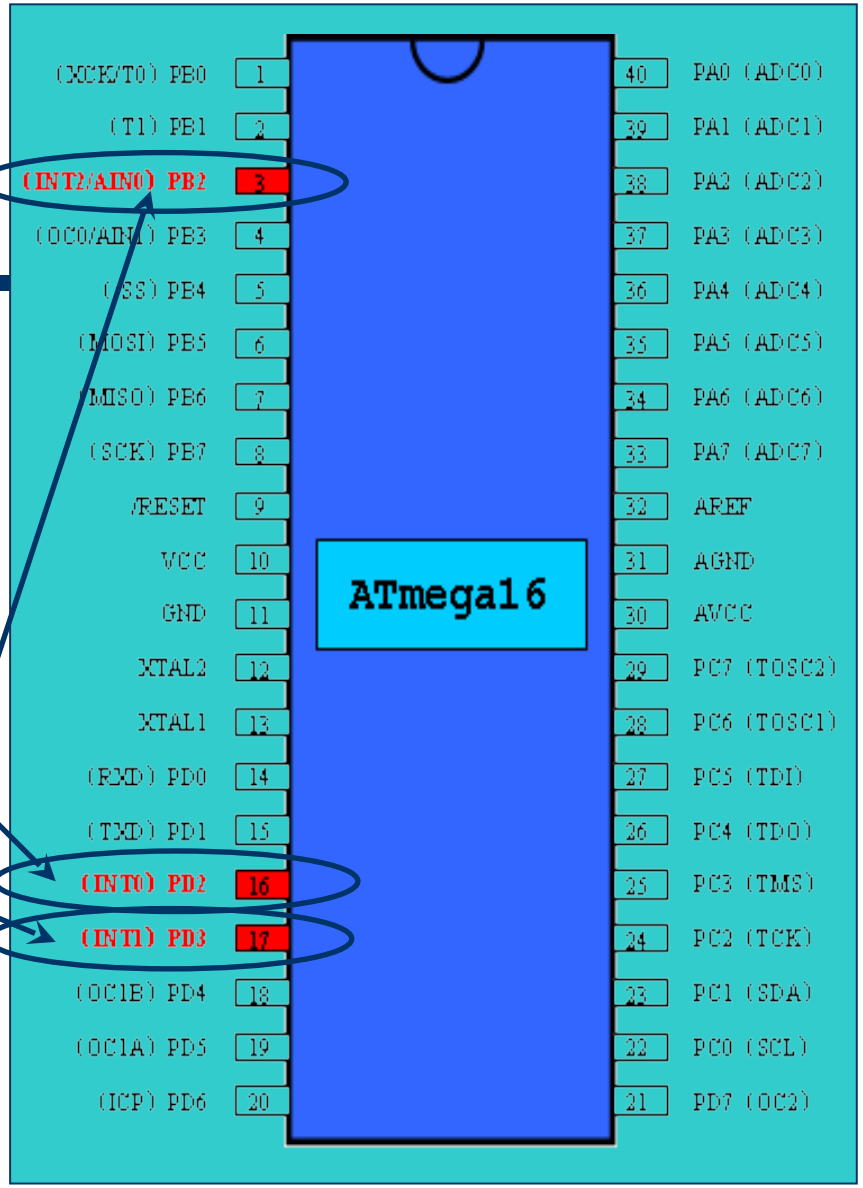
五、 ATmega16 的外部中断

- 外部中断源
- 外部中断的控制

外部中断源

ATmega16 有 3 个外部中断源：

管脚	外部中断
PD2	INT0 （外部中断 0 输入）
PD3	INT1 （外部中断 1 输入）
PB2	INT2 （外部中断 2 输入）



外中断的 4 种中断触发方式

同步边沿触发中断类型

异步方式方式检测

触发方式	INT0	INT1	INT2	说明
上升沿触发	√	√	√	
下降沿触发	√	√	√	
任意电平触发	√	√	—	
低电平触发	√	√	—	

外部中断的控制

ATmega16 的外部中断用户可以控制：

- MCUCR——MCU 控制寄存器
- MCUCSR——MCU 控制和状态寄存器
- GICR—— 通用中断控制寄存器
- GIFR—— 通用中断标志寄存器
- SREG—— 状态寄存器

MCU 控制寄存器 —— MCUCR

7	6	5	4	3	2	1	0
SM2	SE	SM1	SM0	ISC11	ISC10	ISC01	ISC00
复位值： 0	0	0	0	0	0	0	0

位 3~0 ：外部中断 1 、 0 中断请求信号有效方式控制位

外部中断 1 、 0 中断请求信号方式：

ISCx1	ISCx0	中断请求信号有效方式	ISCx1	ISCx0	中断请求信号有效方式
0	0	低电平	1	0	下降沿
0	1	上升沿或下降沿	1	1	上升沿

位 7~4 ：与外部中断的设置无关。

MCU 控制和状态寄存器 —— MCUC

SR	7	6	5	4	3	2	1	0
	JTD	**ISC2**	—	JTRF	WDRF	BORF	EXTRF	PORF
复位值：	0	0	0	0	0	0	0	0

位 6 ：外部中断 2 中断请求信号有效方式控制位。

当 ISC2 清 “ **0**” 时， INT2 引脚上的**下降沿信号异步**触发中断请求；

当 ISC2 置 “ **1**” 时， INT2 引脚上的**上升沿信号异步**触发中断请求。

通用中断控制寄存器 —— GICR

	7	6	5	4	3	2	1	0
	INT1	**INT0**	**INT2**	—	—	—	IVSEL	IVCE
复位值：	0	0	0	0	0	0	0	0

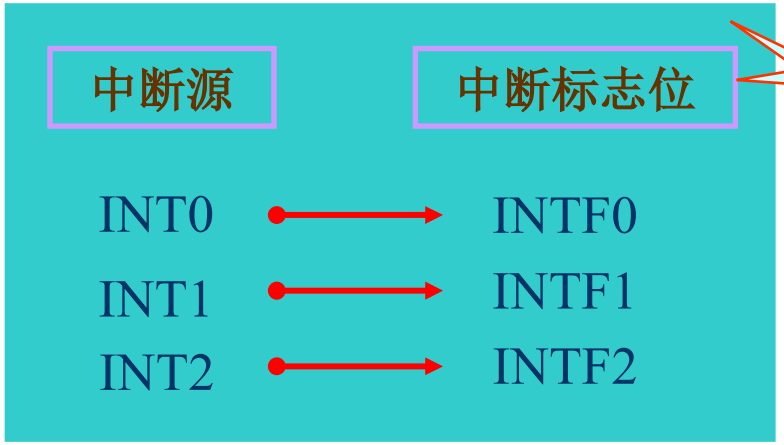
- BIT7： INT1 为外部中断 1 的中断使能位。
- BIT6： INT0 为外部中断 0 的中断使能位。
- BIT5： INT2 为外部中断 2 的中断使能位。

当 **SREG** 寄存器中的**全局中断 I 位为“ 1”**， 且 GICR 寄存器中相应的中断允许位置 1， ‘那么当外部中断触发时， MCU 会响应相应的中断请求。

通用中断标志寄存器 —— GIFR

7	6	5	4	3	2	1	0
INTF1	INTF0	INTF2	—	—	—	—	—
复位值：0	0	0	0	0	0	0	0

- 每一个外部中断源都有相应的中断标志位；
- 某一个外部中断源申请中断，相应中断标志位置1。



中断标志位由CPU自动置“1”

当CPU响应中断后，则中断标志位**自动**清“0”；也可以通过对其写“**1**”来清

状态寄存器 —— SREG

7	6	5	4	3	2	1	0
I	T	H	S	V	N	Z	C
复位值：0	0	0	0	0	0	0	0

位 7：全

局中断，单独的

中断使能由

。

一旦 CPU 响应中断，I 标志位由硬件自动清“0”；当执行中断返回时，I 标志位由硬件自动置“1”。

位 6~0：与中断无关，在 C 语言编程时由系统管理。

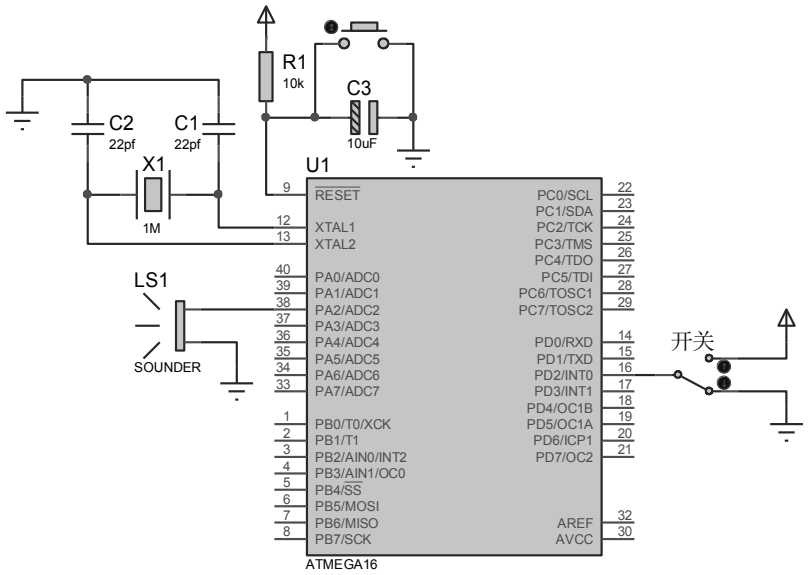
在 GCC AVR C 开发系统中，用 **SEI()**：设置全局中断使能，对应的 **C 语言语句**为 **SREG|=0x80**；用 **CLI()** 禁止中断，对应 **SREG&=~0x80;**

任务二中断报警控制

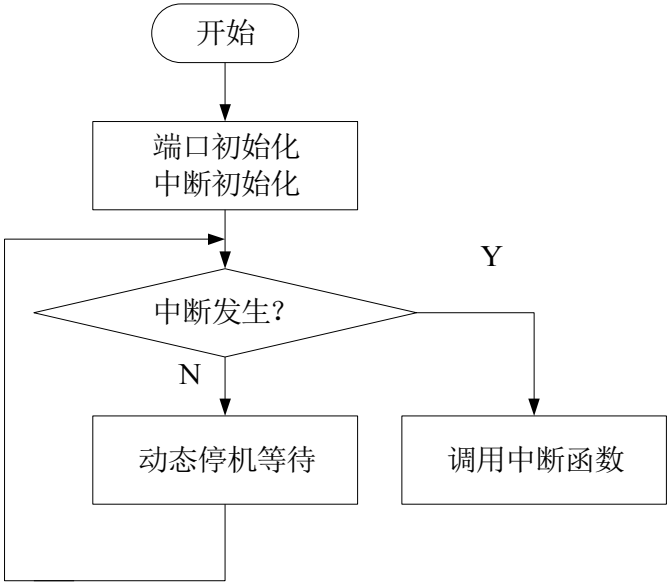
一、任务要求：

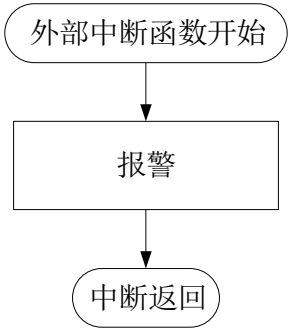
- 设计一个报警系统，利用 ATmega16 单片机的外部中断源，用开关模拟报警信号，当触发报警时，有蜂鸣器报警。

二、硬件电路


R1
10k
C3
10uF
C2
22pf
C1
22pf
X1
1M
U1
RESET
XTAL1
XTAL2
PA0/ADC0
PA1/ADC1
PA2/ADC2
PA3/ADC3
PA4/ADC4
PA5/ADC5
PA6/ADC6
PA7/ADC7
PB0/T0/XCK
PB1/T1
PB2/AIN0/INT2
PB3/AIN1/OC0
PB4/SS
PB5/MOSI
PB6/MISO
PB7/SCK
PC0/SCL
PC1/SDA
PC2/TCK
PC3/TMS
PC4/TDO
PC5/TDI
PC6/TOSC1
PC7/TOSC2
PD0/RXD
PD1/TXD
PD2/INT0
PD3/INT1
PD4/OC1B
PD5/OC1A
PD6/ICP1
PD7/OC2
AREF
AVCC
ATMEGA16
LS1
SOUNDER
开关

三、程序设计



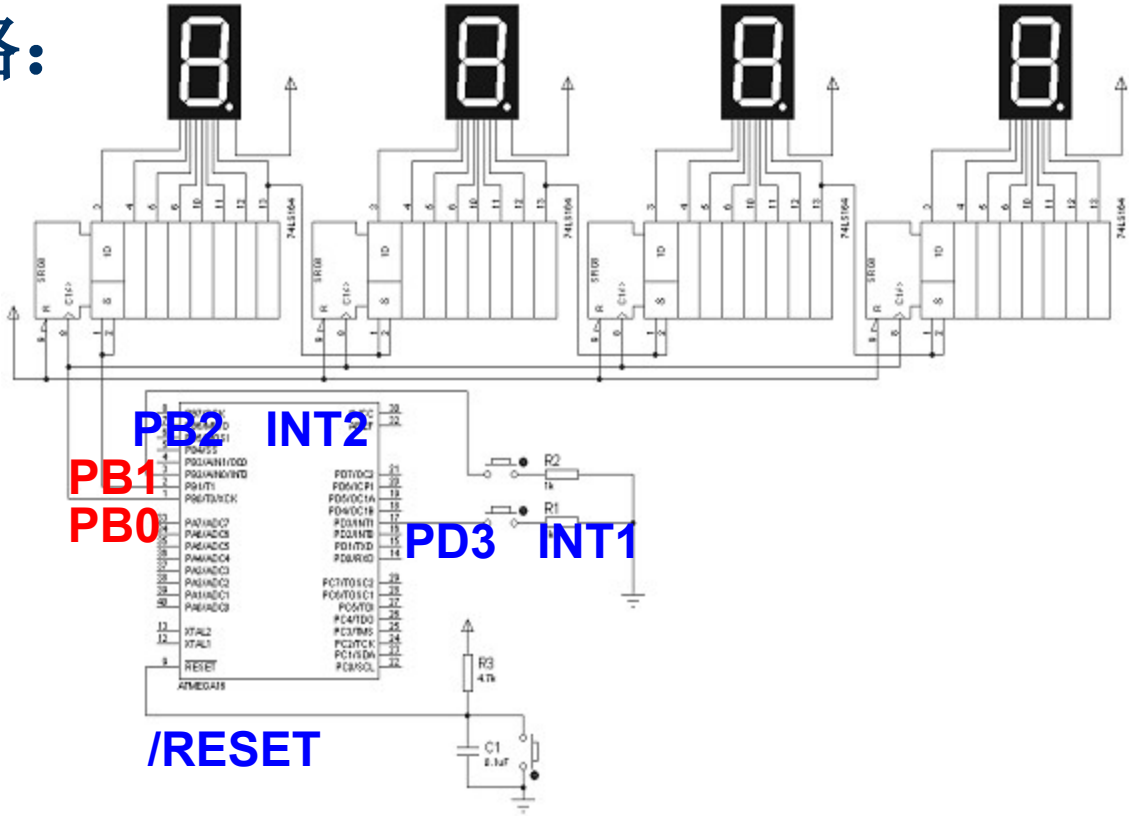


任务 3　加减计数器设计

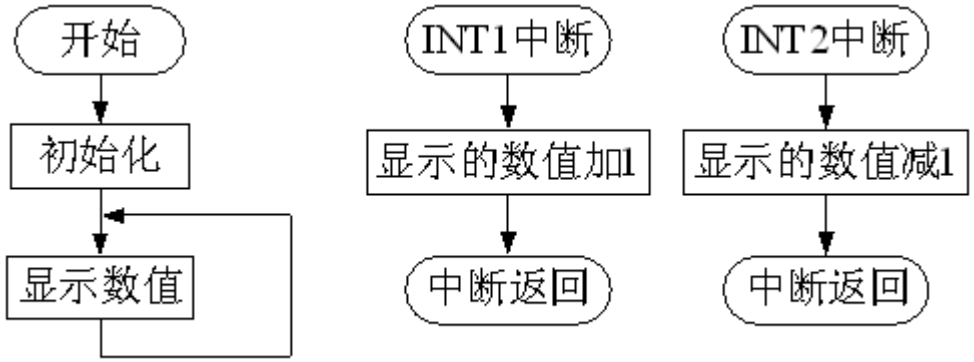
一、任务要求：

- 设计一个加减计数器，利用 ATmega16 单片机的两个外部中断源，分别实现加减功能。触发外部中断 1 时，数码管显示的数据加 1，触发外部中断 2 时，数码管显示的数据减 1。

二、硬件电路：



三、软件设计



【参考程序】

端口初始化：

```
    DDRB|=0x03;          // 设置 PB0 和 PB1 设为输出
    PORTB|=0x04;         //PB2 上拉电阻使能
    DDRD&=0xF7;          // 设置 PD3 输入
    PORTD|=0x08;         // PD3 上拉电阻使能
    CLI();                        // 关全局中断
    GICR|=0xA0;                   //INT1 和 INT2 使能
    MCUCR|=0x08;         // 外部中断 1 下降沿产生中断
    MCUSR|=0x40;         // 外部中断 2 上降沿产生中断
```

任务 4　看门狗报警

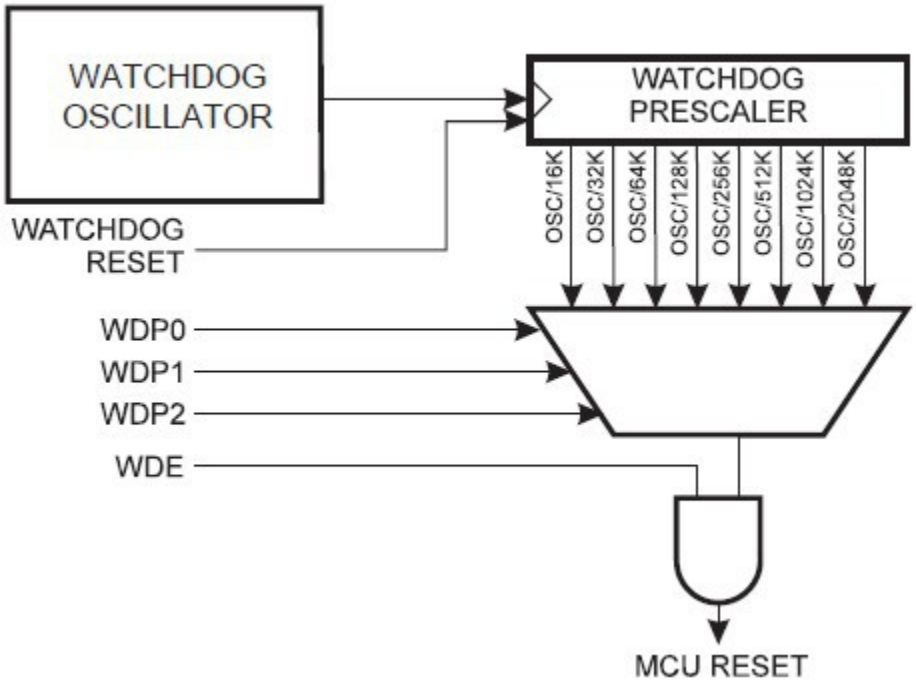
- AVR 单片机的多数型号都有芯片内置的看门狗（ watch dog ）电路，看门狗电路实际上是一个定时器电路，该定时器采用独立的内部 1M 的 RC 振荡器驱动。

- 根据设置的看门狗定时时间，当程序运行时间超过定时时间后，如果没有及时复位看门狗（就是俗称的“喂狗”），看门狗定时器就会发生溢出 (饿死)，这个溢出将导致程序的复位，从而保证在程序跑飞的情况下，不会长时间没有响应。

基于 ATmega16 内部看门狗操作实验

- 复位看门狗（喂狗）
- 使能看门狗定时器
- 关闭看门狗定时器
- 定义看门狗定时器溢出时间

基于 ATmega16 内部看门狗操作实验



基于 ATmega16 内部看门狗操作实验

WINAVR 中自带了看门狗操作函数，利用这些函数可以很轻松的实现对 AVR 单片机内部的看门狗进行控制。

如果要使用 WINAVR 中自带的看门狗操作函数，首先要在程序中包含看门狗操作函数的头文件，使用如下语句即 #include <avr/wdt.h>

基于 ATmega16 内部看门狗操作实验

1、复位看门狗定时器。程序允许在使能看门狗定时器后，在溢出时间到达之前，调用该函数将看门狗复位。如果在规定时间内不调用此函数，则会发生看门狗溢出，导致程序复位。

#define wdt_reset()

基于 ATmega16 内部看门狗操作实验

2、使能看门狗定时器，同时设置看门狗溢出时间

```
#define  wdt_enable(timeout)
```

基于 ATmega16 内部看门狗操作实验

3、关闭看门狗定时器

```
#define  wdt_disable()
```

基于 ATmega16 内部看门狗操作实验

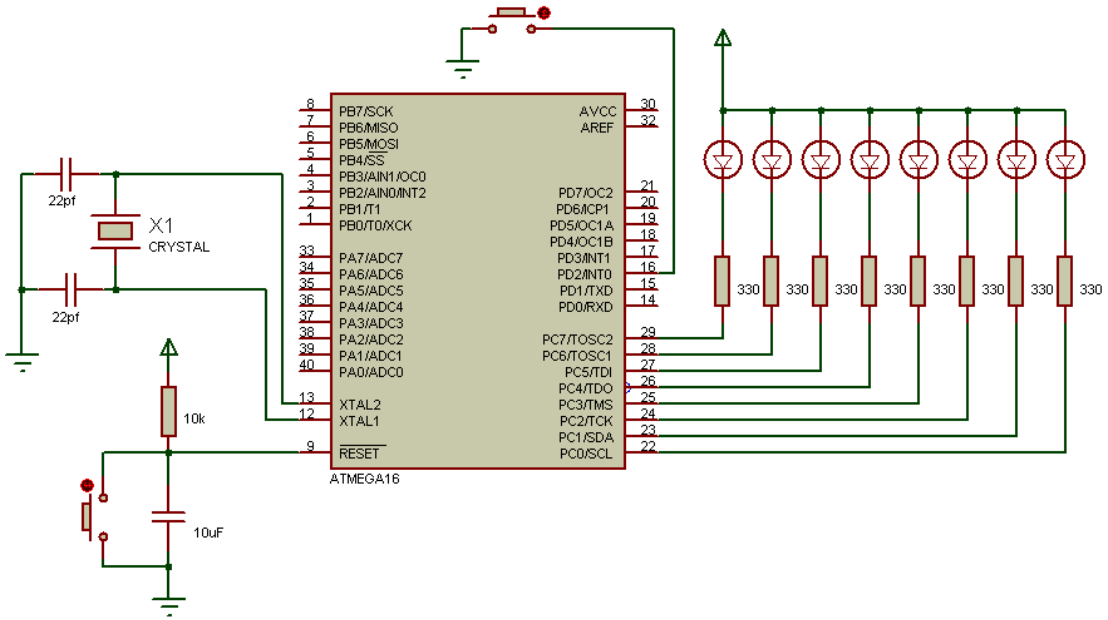
4、定义看门狗定时器溢出时间

```
#define WDTO_15MS   0
#define WDTO_30MS   1
#define WDTO_60MS   2
#define WDTO_120MS  3
#define  WDTO_250MS 4
#define  WDTO_500MS 5
#define  WDTO_1S    6
#define  WDTO_2S    7
```

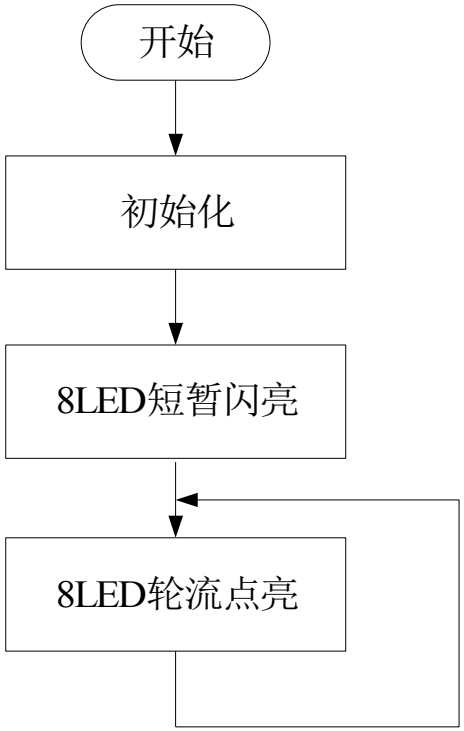
一、任务要求：

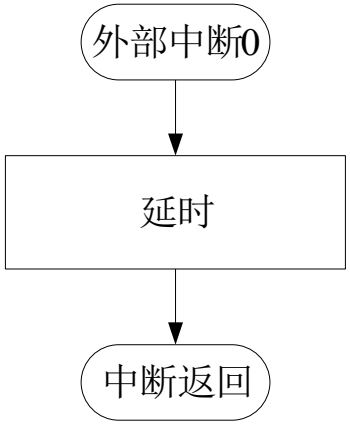
- 利用 ATmega16 单片机内部的看门狗定时器，设计一个单片机抗干扰应用系统。当霓虹灯显示系统启动时， 8 个 LED 等短暂闪烁，正常运行后 8 个 LED 循环点亮，由外部按键触发模拟干扰源（停止喂狗， LED 不正常点亮），系统自动重新启动进入正常运行状态。

二、硬件电路


22pf
X1
CRYSTAL
22pf
10k
10uF
RESET
XTAL2
XTAL1
PB7/SCK
PB6/MISO
PB5/MOSI
PB4/SS
PB3/AIN1/OC0
PB2/AIN0/INT2
PB1/T1
PB0/T0/XCK
PA7/ADC7
PA6/ADC6
PA5/ADC5
PA4/ADC4
PA3/ADC3
PA2/ADC2
PA1/ADC1
PA0/ADC0
AVCC
AREF
PD7/OC2
PD6/ICP1
PD5/OC1A
PD4/OC1B
PD3/INT1
PD2/INT0
PD1/TXD
PD0/RXD
PC7/TOSC2
PC6/TOSC1
PC5/TDI
PC4/TDO
PC3/TMS
PC2/TCK
PC1/SDA
PC0/SCL
ATMEGA16
330
330
330
330
330
330
330
330

三、程序设计





【项目实施】

1. 根据元器件清单选择合适的元器件。
2. 根据硬件设计原理图，在万能电路板进行元器件布局，并进行焊接工作。
3. 焊接完成后，重复进行线路检查，防止短路、虚接现象。
4. 在 AVR Studio 软件中创建项目，输入源代码并生成 *.hex 文件。
5. 在确认硬件电路正确的前提下，通过 JTAG 仿真器进行程序的下载与硬件在线调试。