任务 1

ATmega16 单片机 I/O 端口应用

1.1 任务要求

1. 闪烁灯

设计制作一个 8 位 LED 闪烁灯，要求如下：

- PA0、PA1 口为输入口，分别接一个开关。
- PC 口为输出口，接 8 只 LED。
- PA0 开关断开且 PA1 开关闭合时，PC 口所接的 D2、D4、D6、D8 点亮。
- PA0 开关闭合且 PA1 开关断开时，PC 口所接的 D1、D3、D5、D7 点亮。
- LED 阳极接 I/O 口，阴极接地。

2. 流水灯

设计制作一个 8 位 LED 流水灯，要求如下：

- PC 口为输出口，接 8 只 LED。
- 运行程序后，8 只 LED 依次从上到下点亮，时间间隔 1 秒，形成流水效果。
- LED 阳极接 I/O 口，阴极接地。

1.2 相关知识

1.2.1 I/O 端口介绍

作为通用数字 I/O 使用时，所有 AVRI/O 端口都具有真正的读－修改－写功能。这意味着用 SBI 或 CBI 指令改变某些管脚的方向（或者是端口电平、禁止/使能上拉电阻）时不会无意地改变其他管脚的方向（或者是端口电平、禁止/使能上拉电阻）。输出缓冲器具有对称的驱动能力，可以输出或吸收大电流，直接驱动 LED。所有的端口引脚都具有与电压无关的上拉电阻，并有保护二极管与 V_{CC} 和地相连。

这里所有的寄存器和位以通用格式表示：小写的“x”表示端口的序号，小写的“n”代表位的

序号。但是在程序里要写完整。例如，PORTB3 表示端口 B 的第 3 位，而本节的通用格式为 PORTxn。每个端口都有三个 I/O 存储器地址：数据寄存器（PORTx）、数据方向寄存器（DDRx）和端口输入引脚（PINx）。数据寄存器和数据方向寄存器为读/写寄存器，而端口输入引脚为只读寄存器。但是需要特别注意的是，对 PINx 寄存器某一位写入逻辑“1”将造成数据寄存器相应位的数据发生“0”与“1”的交替变化。当寄存器 MCUCR 的上拉禁止位 PUD 置位时，所有端口引脚的上拉电阻都被禁止。多数端口引脚是与第二功能复用的，使能某些引脚的第二功能不会影响其他属于同一端口的引脚用于通用数字 I/O 的目的。

1.2.2 作为通用数字 I/O 的端口

端口为具有可选上拉电阻的双向 I/O 端口。

1. 配置引脚

每个端口引脚都具有三个寄存器位：DDxn、PORTxn 和 PINxn，DDxn 位于 DDRx 寄存器，PORTxn 位于 PORTx 寄存器，PINxn 位于 PINx 寄存器。

DDxn 用来选择引脚的方向。DDxn 为“1”时，Pxn 配置为输出，否则配置为输入。

引脚配置为输入时，若 PORTxn 为“1”，上拉电阻将使能。如果需要关闭这个上拉电阻，可以将 PORTxn 清零，或者将这个引脚配置为输出。

端口引脚配置如表 2-1-1 所示。

表 2-1-1 端口的引脚配置

DDxn	PORTxn	PUD(in SFIOR)	I/O	上拉电阻	说明
0	0	x	Input	No	高阻态（Hi-Z）
0	1	0	Input	Yes	被外部电路拉低时将输出电流
0	1	1	Input	No	高阻态（Hi-Z）
1	0	x	Output	No	输出低电平（吸收电流）
1	1	x	Output	No	输出高电平（输出电流）

复位时各引脚为高阻态，即使此时并没有时钟在运行。

当引脚配置为输出时，若 PORTxn 为“1”，引脚输出高电平（1），否则输出低电平（0）。在（高阻态）三态（{DDxn，PORTxn} = 0b00）输出高电平（{DDxn，PORTxn} = 0b11）两种状态之间进行切换时，上拉电阻使能（{DDxn，PORTxn} = 0b01）或输出低电平（{DDxn，PORTxn} = 0b10），这两种模式必然会有一个发生。通常，上拉电阻使能是完全可以接受的，因为高阻环境不在意是强高电平输出还是上拉输出。如果使用情况不是这样，可以通过置位 SFIOR 寄存器的 PUD 来禁止所有端口的上拉电阻。在上拉输入和输出低电平之间切换也有同样的问题。用户必须选择高阻态（{DDxn，PORTxn} = 0b00）或输出高电平（{DDxn，PORTxn} = 0b10）作为中间步骤。

2. 读取引脚上的数据

不论如何配置 DDxn，都可以通过读取 PINxn 寄存器来获得引脚电平。PINxn 寄存器的各个位与其前面的锁存器组成了一个同步器。这样就可以避免在内部时钟状态发生改变的短时间范围内由于引脚电平变化而造成的信号不稳定。其缺点是引入了延迟。

3. 数字输入使能和休眠模式

数字输入信号（施密特触发器的输入）可以钳位到地。SLEEP 信号由 MCU 休眠控制器在各种掉电模式、省电模式、Standby 模式下设置，以防止在输入悬空或模拟输入电平接近 $V_{CC}/2$ 时消耗太多的电流。引脚作为外部中断输入时 SLEEP 信号无效。但若外部中断没有使能，SLEEP 信号仍然有效。引脚的第二功能使能时 SLEEP 也让位于第二功能。如果逻辑高电平（1）出现在一个被设置为“上升沿、下降沿或任何逻辑电平变化都引起中断”的外部异步中断引脚上，即使该外部中断未被使能，但从上述休眠模式唤醒时，相应的外部中断标志位仍会被置“1”。这是因为引脚电平在休眠模式下被钳位到“0”电平，唤醒过程造成了引脚电平从“0”到“1”的变化。

4. 未连接引脚的处理

如果有引脚未被使用，建议给这些引脚赋予一个确定电平。虽然如上文所述，在深层休眠模式下大多数数字输入被禁用，但还是需要避免因引脚没有确定的电平而造成悬空引脚在其他数字输入使能模式（复位、工作模式、空闲模式）消耗电流。最简单的保证未用引脚具有确定电平的方法是使能内部上拉电阻。但要注意的是复位时上拉电阻将被禁用。如果复位时的功耗也有严格要求，则建议使用外部上拉或下拉电阻。不推荐直接将未用引脚与 V_{CC} 或 GND 连接，因为这样可能会在引脚偶然作为输出时出现冲击电流。

1.2.3 端口的第二功能

AVR 单片机除了通用数字 I/O 功能之外，大多数端口引脚都具有第二功能。

1. 特殊功能 I/O 寄存器 SFIOR

定义如下：

Bit	7	6	5	4	3	2	1	0	
	ADTS2	ADTS1	ADTS0	—	ACME	PUD	PSR2	PSR10	SFIOR
读/写	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
初始值	0	0	0	0	0	0	0	0	

Bit 2 – PUD：禁用上拉电阻。

置位时，即使将寄存器 DDxn 和 PORTxn 配置为使能上拉电阻 ({DDxn, PORTxn} =0b01)，I/O 端口的上拉电阻也被禁止。

2. 端口 A 的第二功能

端口 A 作为 ADC 模拟输入的第二功能如表 2-1-2 所示。如果端口 A 的部分引脚置为输出，当转换时不能切换，否则会影响转换结果。

表 2-1-2 端口 A 的第二功能

端口引脚	第二功能
PA7	ADC7
PA6	ADC6
PA5	ADC5
PA4	ADC4

续表

端口引脚	第二功能
PA3	ADC3
PA2	ADC2
PA1	ADC1
PA0	ADC0

3. 端口 B 的第二功能

端口 B 的第二功能如表 2-1-3 所示。

表 2-1-3 端口 B 的第二功能

端口引脚	第二功能
PB7	SCK（SPI 总线的串行时钟）
PB6	MISO（SPI 总线的主机输入/从机输出信号）
PB5	MOSI（SPI 总线的主机输出/从机输入信号）
PB4	SS（SPI 从机选择引脚）
PB3	AIN1（模拟比较负输入） OC0（T/C0 输出比较匹配输出）
PB2	AIN0（模拟比较正输入） INT2（外部中断 2 输入）
PB1	T1（T/C1 外部计数器输入）
PB0	T0（T/C0 外部计数器输入） XCK（USART 外部时钟输入/输出）

- SCK–端口 B，Bit 7

SCK：SPI 通道的主机时钟输出、从机时钟输入端口。工作于从机模式时，不论 DDB7 设置如何，这个引脚都将设置为输入。工作于主机模式时，这个引脚的数据方向由 DDB7 控制。设置为输入后，上拉电阻由 PORTB7 控制。

- MISO–端口 B，Bit 6

MISO：SPI 通道的主机数据输入、从机数据输出端口。工作于主机模式时，不论 DDB6 设置如何，这个引脚都将设置为输入。工作于从机模式时，这个引脚的数据方向由 DDB6 控制。设置为输入后，上拉电阻由 PORTB6 控制。

- MOSI–端口 B，Bit 5

MOSI：SPI 通道的主机数据输出、从机数据输入端口。工作于从机模式时，不论 DDB5 设置如何，这个引脚都将设置为输入。当工作于主机模式时，这个引脚的数据方向由 DDB5 控制。设置为输入后，上拉电阻由 PORTB5 控制。

- SS–端口 B，Bit 4

SS：从机选择输入。工作于从机模式时，不论 DDB4 设置如何，这个引脚都将设置为输入。当此引脚为低时 SPI 被激活。工作于主机模式时，这个引脚的数据方向由 DDB4 控制。设置为输入后，上拉电阻由 PORTB4 控制。

- AIN1/OC0–端口 B，Bit 3

AIN1：模拟比较负输入。配置该引脚为输入时，切断内部上拉电阻，防止数字端口功能与模拟比较器功能相冲突。

OC0：输出比较匹配输出。PB3 引脚可作为 T/C0 比较匹配的外部输出。实现该功能时，PB3 引脚必须配置为输出（设 DDB3 为 1）。在 PWM 模式的定时功能中，OC0 引脚作为输出。

- AIN0/INT2–端口 B，Bit 2

AIN0：模拟比较正输入。配置该引脚为输入时，切断内部上拉电阻，防止数字端口功能与模拟比较器功能相冲突。

INT2：外部中断源 2。PB2 引脚作为 MCU 的外部中断源。

- T1–端口 B，Bit 1

T1：T/C1 计数器源。

- T0/XCK–端口 B，Bit 0

T0：T/C0 计数器源。

XCK：USART 外部时钟。数据方向寄存器（DDB0）控制时钟为输出（DDB0 置位）还是输入（DDB0 清零）。只有当 USART 工作在同步模式时，XCK 引脚才被激活。

4. *端口 C 的第二功能*

端口 C 的第二功能如表 2-1-4 所示。若 JTAG 接口使能，即使出现复位，引脚 PC5（TDI）、PC3（TMS）与 PC2（TCK）的上拉电阻也会被激活。

表 2-1-4　端口 C 的第二功能

端口引脚	第二功能
PC7	TOSC2（定时振荡器引脚 2）
PC6	TOSC1（定时振荡器引脚 1）
PC5	TDI（JTAG 测试数据输入）
PC4	TDO（JTAG 测试数据输出）
PC3	TMS（JTAG 测试模式选择）
PC2	TCK（JTAG 测试时钟）
PC1	SDA（两线串行总线数据输入/输出线）
PC0	SCL（两线串行总线时钟线）

- TOSC2–端口 C，Bit 7

TOSC2：定时振荡器引脚 2。当寄存器 ASSR 的 AS2 位置 1，使能 T/C2 的异步时钟，引脚 PC7 与端口断开，成为振荡器放大器的反向输出。在这种模式下，晶体振荡器与该引脚相连，该引脚不能作为 I/O 引脚。

- TOSC1–端口 C，Bit 6

TOSC1：定时振荡器引脚 1。当寄存器 ASSR 的 AS2 位置 1，使能 T/C2 的异步时钟，引脚 PC6 与端口断开，成为振荡器放大器的反向输出。在这种模式下，晶体振荡器与该引脚相连，该引脚不能作为 I/O 引脚。

- TDI–端口 C，Bit 5

TDI：JTAG 测试数据输入。串行输入数据移入指令寄存器或数据寄存器（扫描链）。当 JTAG

接口使能，该引脚不能作为 I/O 引脚。

- TDO–端口 C，Bit 4

TDO：JTAG 测试数据输入。串行输入数据移入指令寄存器或数据寄存器（扫描链）。当 JTAG 接口使能，该引脚不能作为 I/O 引脚。TD0 引脚在除 TAP 状态情况外为三态，进入移出数据状态。

- TMS–端口 C，Bit 3

TMS：JTAG 测试模式选择。该引脚作为 TAP 控制器状态工具的定位。当 JTAG 接口使能，该引脚不能作为 I/O 引脚。

- TCK–端口 C，Bit 2

TCK：JTAG 测试时钟。JTAG 工作在同步模式下。当 JTAG 接口使能，该引脚不能作为 I/O 引脚。

- SDA–端口 C，Bit 1

SDA：两线串行接口数据。当寄存器 TWCR 的 TWEN 位置 1，使能两线串行接口，引脚 PC1 不与端口相连，且成为两线串行接口的串行数据 I/O 引脚。在该模式下，在引脚处使用窄带滤波器抑制低于 50ns 的输入信号，且该引脚由斜率限制的开漏驱动器驱动。当该引脚使用两线串行接口，仍可由 PORTC1 位控制上拉。

- SCL–端口 C，Bit 0

SCL：两线串行接口时钟。当 TWCR 寄存器的 TWEN 位置 1，使能两线串行接口，引脚 PC0 未与端口连接，成为两线串行接口的串行时钟 I/O 引脚。在该模式下，在引脚处使用窄带滤波器抑制低于 50ns 的输入信号，且该引脚由斜率限制的开漏驱动器驱动。当该引脚使用两线串行接口时，仍可由 PORTC0 位控制上拉。

5. 端口 D 的第二功能

端口 D 的第二功能如表 2-1-5 所示。

表 2-1-5 端口 C 的第二功能

端口引脚	第二功能
PD7	OC2（T/C2 输出比较匹配输出）
PD6	ICP1（T/C1 输入捕捉引脚）
PD5	OC1A（T/C1 输出比较 A 匹配输出）
PD4	OC1B（T/C1 输出比较 B 匹配输出）
PD3	INT1（外部中断 1 的输入）
PD2	INT0（外部中断 0 的输入）
PD1	TXD（USART 输出引脚）
PD0	RXD（USART 输入引脚）

- OC2–端口 D，Bit 7

OC2：T/C2 输出比较匹配输出。PD7 引脚作为 T/C2 输出比较外部输入。在该功能下引脚作为输出（DDD7 置 1）。在 PWM 模式的定时器功能中，OC2 引脚作为输出。

- ICP1–端口 D，Bit 6

ICP1–输入捕捉引脚。PD6 作为 T/C1 的输入捕捉引脚。

- OC1A–端口 D，Bit 5

OC1A：T/C1 输出比较匹配 A 输出。PD5 引脚作为 T/C1 输出比较 A 外部输入。在该功能下引脚作为输出（DDD5 置 1）。在 PWM 模式的定时器功能中，OC1A 引脚作为输出。

- OC1B–端口 D，Bit 4

OC1B：T/C1 输出比较匹配 B 输出。PD4 引脚作为 T/C1 输出比较 B 外部输入。在该功能下引脚作为输出（DDD4 置 1）。在 PWM 模式的定时器功能中，OC1B 引脚作为输出。

- INT1–端口 D，Bit 3

INT1：外部中断 1。PD3 引脚作为 MCU 的外部中断源。

- INT0–端口 D，Bit 2

INT0：外部中断 0。PD2 引脚作为 MCU 的外部中断源。

- TXD–端口 D，Bit 1

TXD：USART 的数据发送引脚。当使能了 USART 的发送器后，这个引脚被强制设置为输出，此时 DDD1 不起作用。

- RXD–端口 D，Bit 0

RXD：USART 的数据接收引脚。当使能了 USART 的接收器后，这个引脚被强制设置为输出，此时 DDD0 不起作用。但是 PORTD0 仍然控制上拉电阻。

1.2.4 I/O 端口寄存器的说明

1. 端口 A 数据寄存器－PORTA

定义如下：

Bit	7	6	5	4	3	2	1	0	
	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	PORTA
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
初始值	0	0	0	0	0	0	0	0	

2. 端口 A 数据方向寄存器－DDRA

定义如下：

Bit	7	6	5	4	3	2	1	0	
	DDRA7	DDRA6	DDRA5	DDRA4	DDRA3	DDRA2	DDRA1	DDRA0	DDRA
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
初始值	0	0	0	0	0	0	0	0	

3. 端口 A 输入引脚地址－PINA

定义如下：

Bit	7	6	5	4	3	2	1	0	
	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	PINA
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
初始值	0	0	0	0	0	0	0	0	

4. 端口 B 数据寄存器 - PORTB

定义如下：

Bit	7	6	5	4	3	2	1	0	
	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
初始值	0	0	0	0	0	0	0	0	

5. 端口 B 数据方向寄存器 - DDRB

定义如下：

Bit	7	6	5	4	3	2	1	0	
	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0	DDRB
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
初始值	0	0	0	0	0	0	0	0	

6. 端口 B 输入引脚地址 - PINB

定义如下：

Bit	7	6	5	4	3	2	1	0	
	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	PINB
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
初始值	0	0	0	0	0	0	0	0	

7. 端口 C 数据寄存器 - PORTC

定义如下：

Bit	7	6	5	4	3	2	1	0	
	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	PORTC
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
初始值	0	0	0	0	0	0	0	0	

8. 端口 C 数据方向寄存器 - DDRC

定义如下：

Bit	7	6	5	4	3	2	1	0	
	DDRC7	DDRC6	DDRC5	DDRC4	DDRC3	DDRC2	DDRC1	DDRC0	DDRC
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
初始值	0	0	0	0	0	0	0	0	

9. 端口 C 输入引脚地址 - PINC

定义如下：

Bit	7	6	5	4	3	2	1	0	
	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	PINC
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
初始值	0	0	0	0	0	0	0	0	

10. 端口 D 数据寄存器－PORTD

定义如下：

Bit	7	6	5	4	3	2	1	0	
	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	PORTD
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
初始值	0	0	0	0	0	0	0	0	

11. 端口 D 数据方向寄存器－DDRD

定义如下：

Bit	7	6	5	4	3	2	1	0	
	DDRD7	DDRD6	DDRD5	DDRD4	DDRD3	DDRD2	DDRD1	DDRD0	DDRD
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
初始值	0	0	0	0	0	0	0	0	

12. 端口 D 输入引脚地址－PIND

定义如下：

Bit	7	6	5	4	3	2	1	0	
	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	PIND
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
初始值	0	0	0	0	0	0	0	0	

1.3 任务分析与实施

1.3.1 闪烁灯

1. 任务构思

根据任务要求，将 PA0、PA1 设置为输入口，故将 DDRA 的 D0、D1 位设置为 0，其他位设置为 1，PC 口设置为输出。

若开关闭合为低电平 0，断开为高电平 1，PC 口输出高电平，LED 点亮，输出低电平，LED 熄灭。

2. 任务设计

读入 PA 口寄存器 PINA 的值，并将 PA0、PA1 值保留，其他位清零，将结果保存在变量 KEY 中，这样可能的结果只有 00，01，10，11 四个取值。使用程序进行判断，当 KEY=0 或者 KEY=3

时，PC 口输出 0，发光二极管全部熄灭；当 KEY=1 时，PC 口输出 0xAA，控制奇数发光管亮，当 KEY=2 时，PC 口输出 0x55，控制偶数发光管亮。

任务设计流程如图 2-1-1 所示。

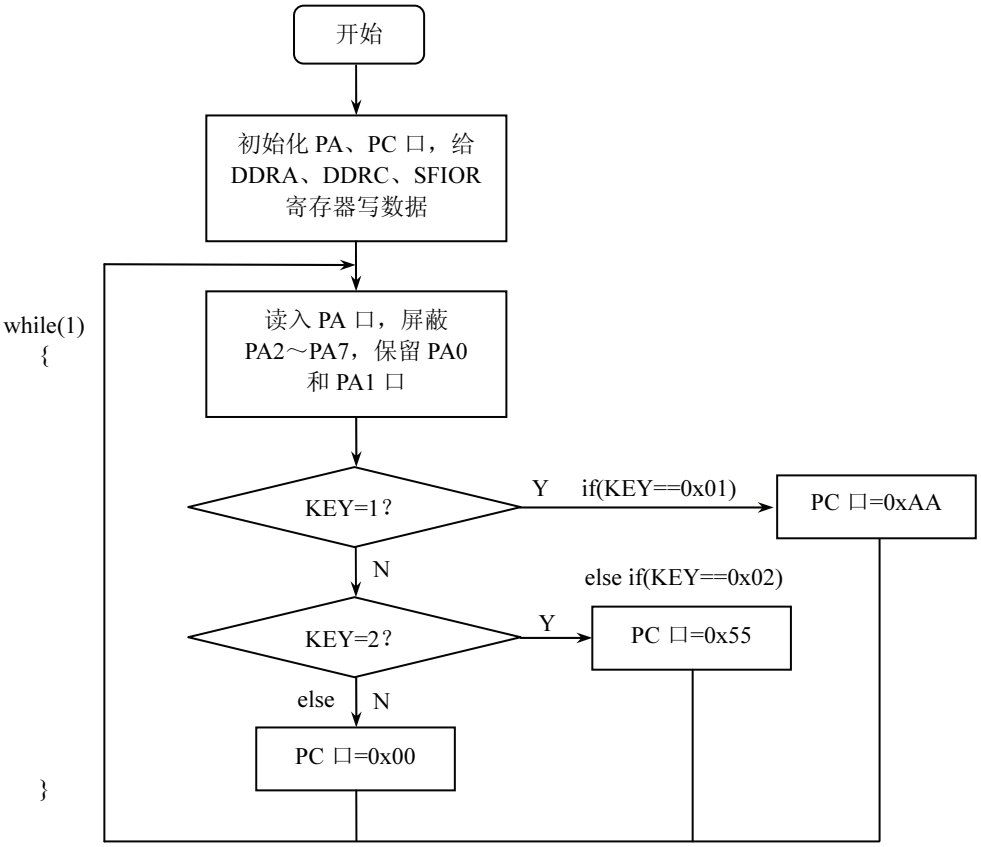

图 2-1-1　任务程序流程图

编写程序如下：

```
/****************************************************
 File name:          闪烁灯.c
 Chip type:          ATmega16
 Clock frequency:    8.0MHz
****************************************************/
#include <iom16v.h>
void main(void)
{
    unsigned char KEY;
    DDRA=0x00;
    DDRC=0xff;
    SFIOR&=0xfb;
    PORTA=0xff;
    while(1)
    {
     KEY=PINA&0x03;
     if(KEY==0x01)
```

```
            PORTC=0xaa;
          else if(KEY==0x02)
            PORTC=0x55;
          else PORTC=0x00;
      }
}
```

3．任务实现

（1）新建设计模板。

打开 ISIS 7 Professional 窗口，选择 File→New Design 命令，弹出模板选择窗口，如图 2-1-2 所示。图中纵向图纸为 Portrait，横向图纸为 Landscape，DEFAULT 为默认。选中 DEFAULT，单击 OK 按钮，即新建好一个 DEFAULT 模板。

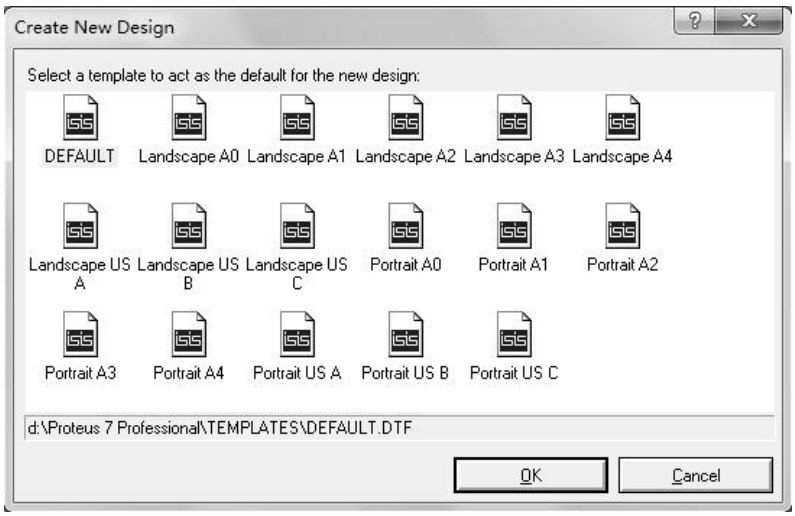

图 2-1-2　模板选择

（2）设定图纸大小。

选择 System→Set Sheet Sizes 命令，弹出对话框，在其中选中 A4 复选框，单击 OK 按钮，完成图纸设定，如图 2-1-3 所示。

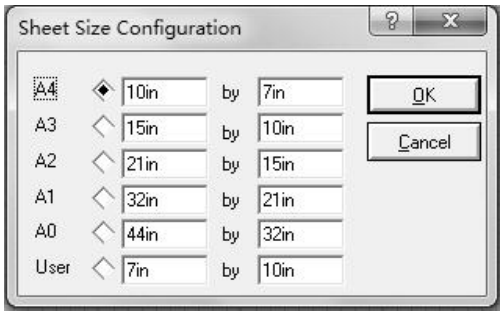

图 2-1-3　图纸选择

（3）元器件的添加。

本次设计图纸所使用的元器件如表 2-1-6 所示。

表 2-1-6 元器件列表

符号	中文	符号	中文	符号	中文
ATmega16	单片机	CAP22pF	瓷片电容	CRYSTAL 8MHz	晶振
CAP-ELEC 10μF	电解电容	LED-RED	发光二极管	LED-BLUE	发光二极管
LED-YELLOW	发光二极管	RES	电阻	SWITCH	开关

选择 Library→Pick Device/Symbol 命令，或者在器件选择按钮栏 P L DEVICES 中单击 P 按钮，弹出如图 2-1-4 所示的对话框。

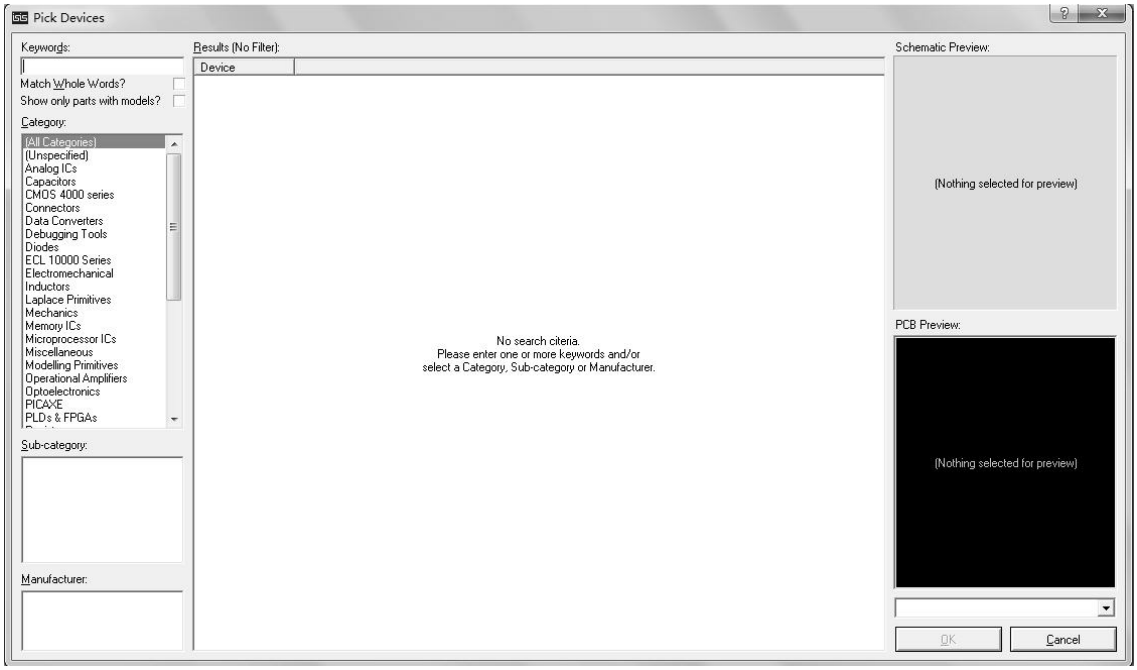

图 2-1-4 元器件选择

在关键字中输入元器件名称，如 ATmega16，则会出现与其相匹配的元器件列表，如图 2-1-5 所示。选中后双击 ATmega16 所在行，再单击 OK 按钮将 ATmega16 元器件加入到 ISIS 对象选择器中。按照上述方法，将其他元器件分别添加到 ISIS 对象选择器中。

（4）原理图绘制。

根据样图将所需元器件放置在图纸上，通过移动、旋转、布线等操作完成整个原理图，如图 2-1-6 所示。

（5）生成网络表并进行电气检测。

选择 Tools→Netlist Compiler 命令，弹出如图 2-1-7 所示的对话框，在其中可以设置网络表的输出形式、模式等，此处不进行修改，单击 OK 按钮以默认方式输出如图 2-1-8 所示的内容。

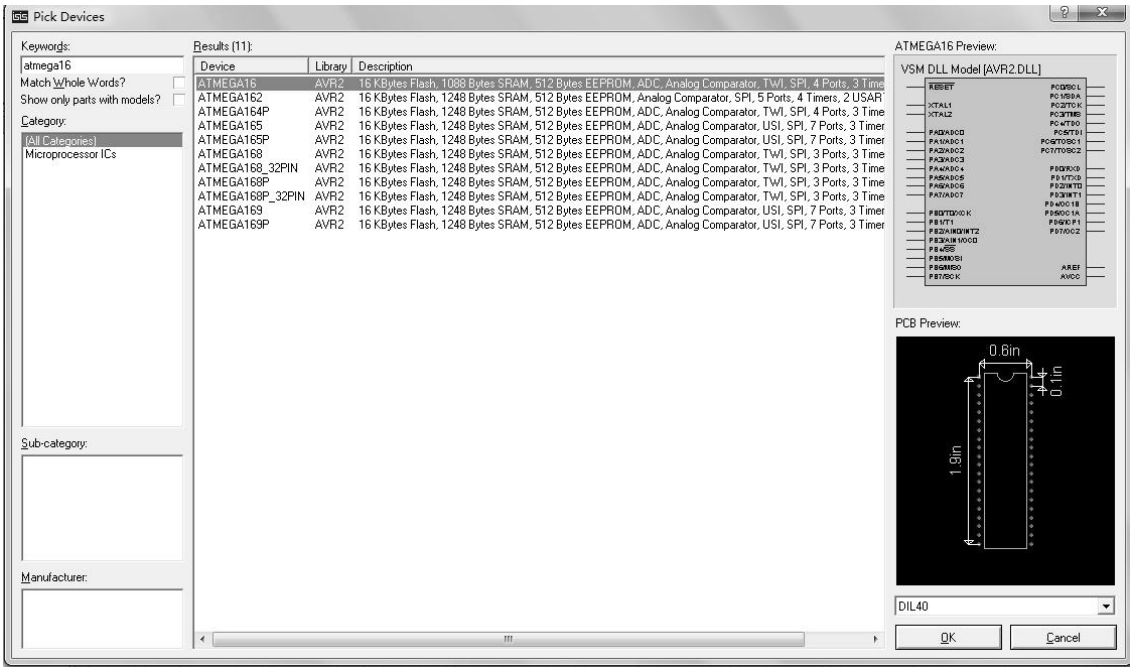

图 2-1-5　输入元器件名称

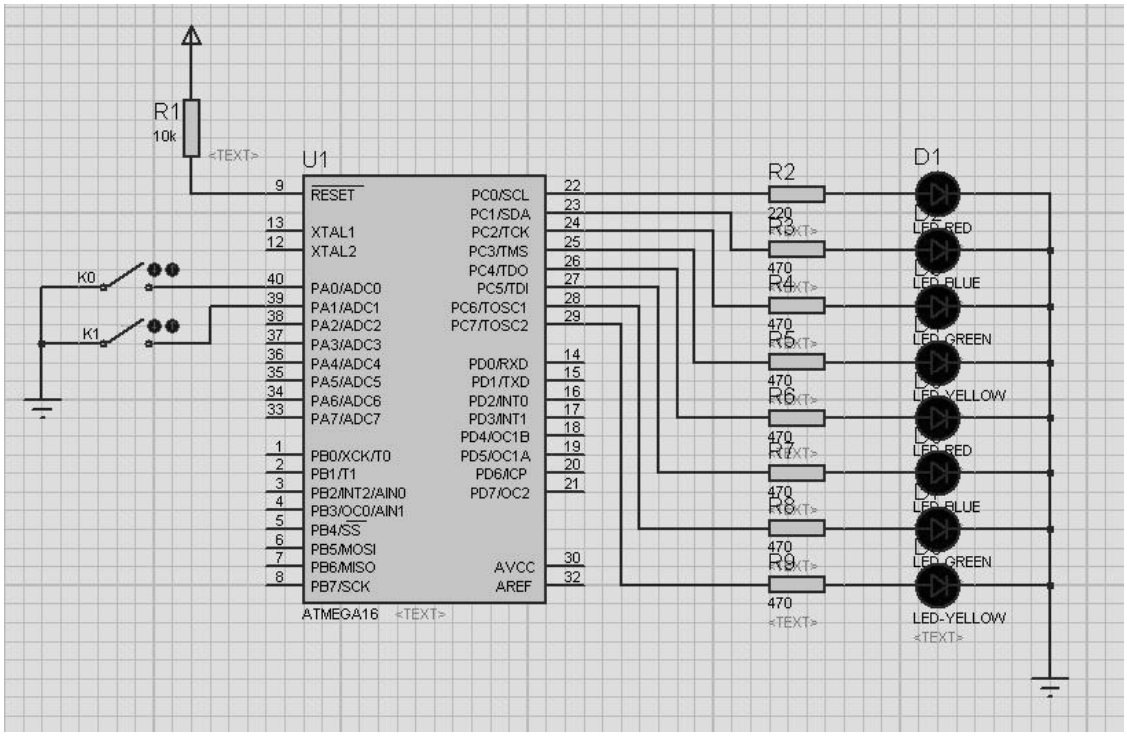

图 2-1-6　原理图

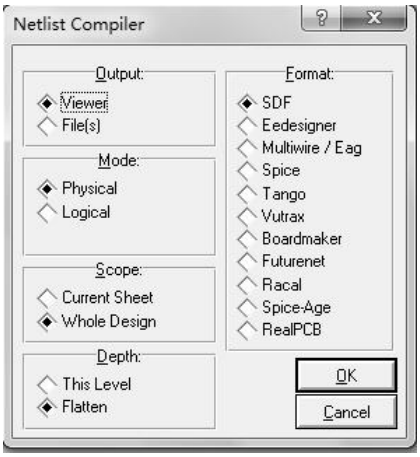

图 2-1-7　网络表设置

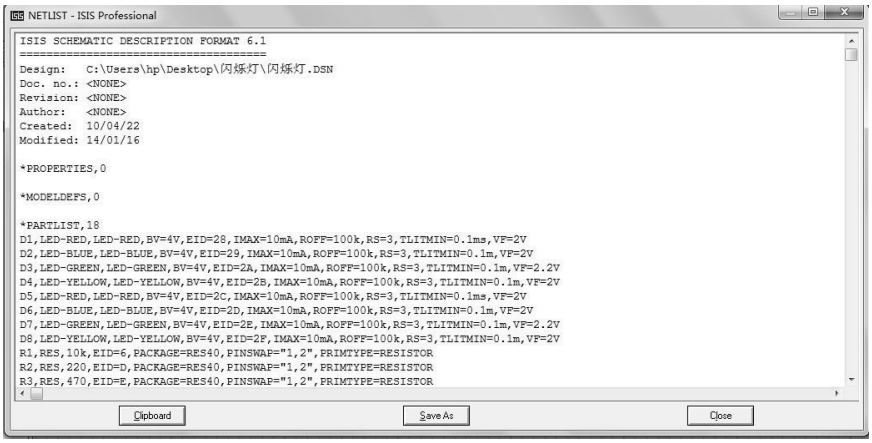

图 2-1-8　输出网络表

电路画完并生成网络表后，可以进行电气检测。选择 Tools→Electrical Rule Check 命令，弹出如图 2-1-9 所示的电气检测窗口，从中可以看到无电气错误。

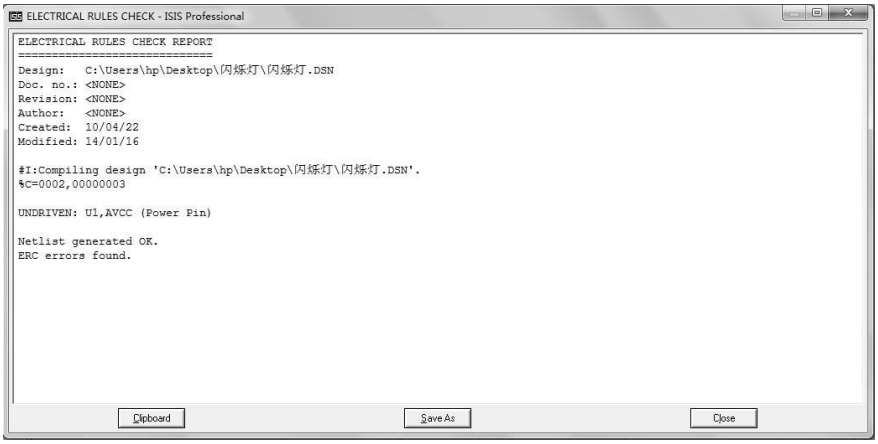

图 2-1-9　电气检测

4. 任务运行

（1）载入。

打开 ATmega16 单片机的属性设置对话框，找到 Program File 选项，如图 2-1-10 所示。载入 ICCAVR 或 CodeVisionAVR 生成的 CHENGXU1.cof 文件或 CHENGXU1.hex 文件，如图 2-1-11 所示。

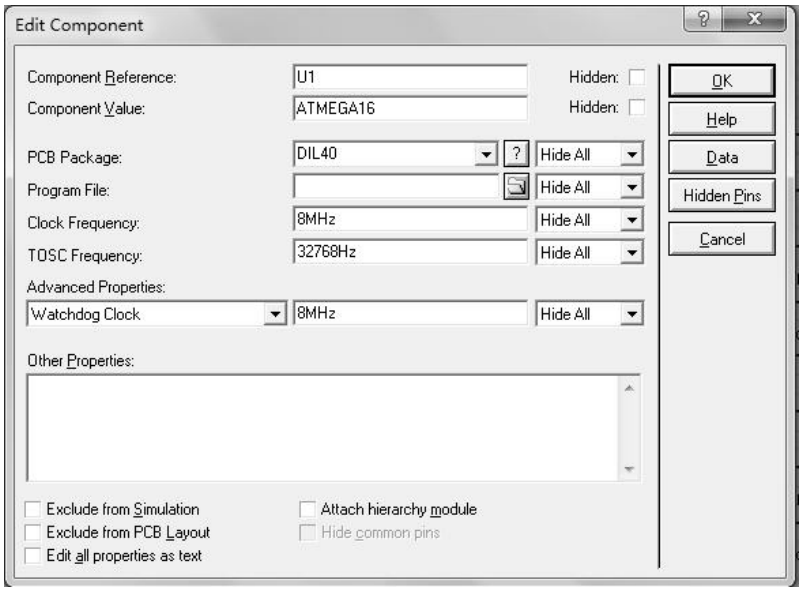

图 2-1-10　单片机属性设置

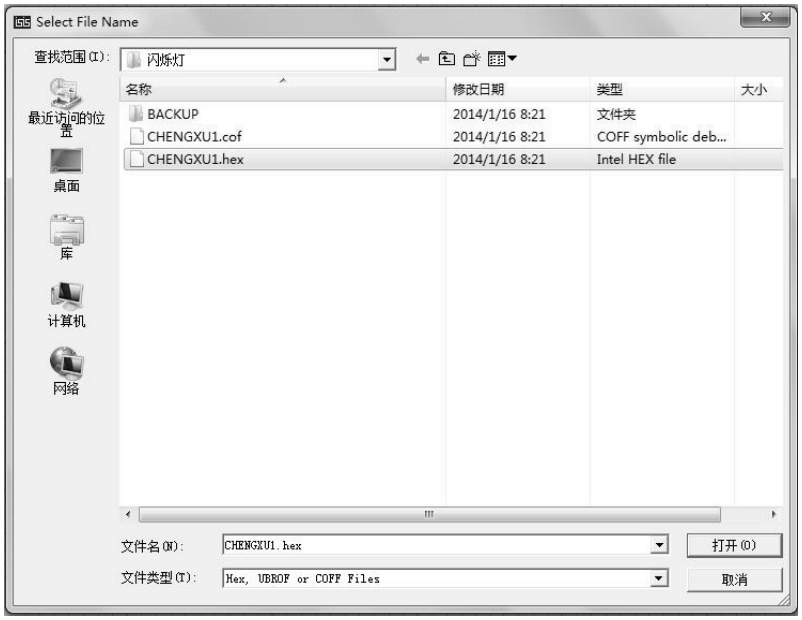

图 2-1-11　载入文件

（2）仿真。

单击 Proteus 的运行按钮，观察仿真现象，如图 2-1-12 和图 1-13 所示。

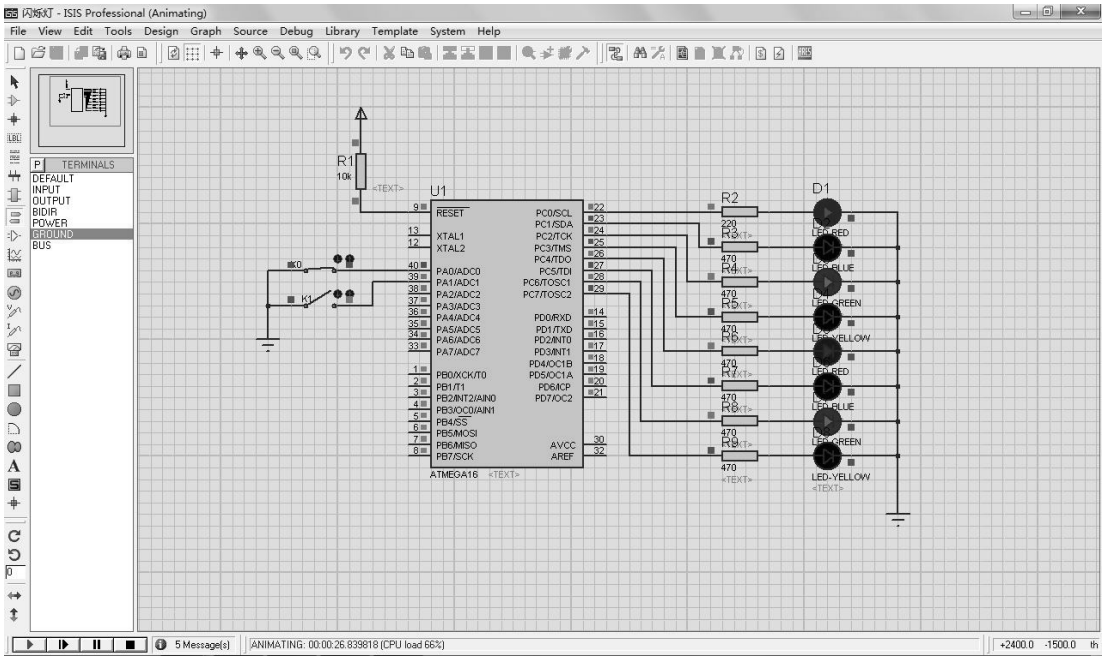

图 2-1-12　运行后按下开关 K0

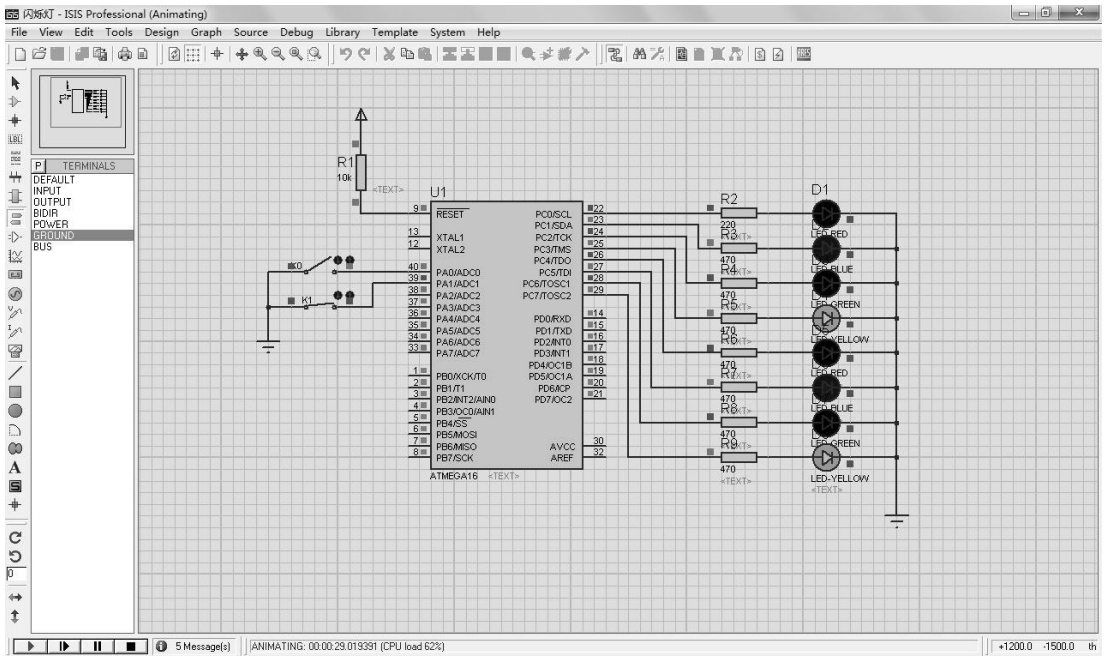

图 2-1-13　运行后按下开关 K1

1.3.2　流水灯

1. 任务构思

根据任务要求，LED 的驱动方式为 I/O 口输出高电平 1 时点亮，输出低电平 0 时熄灭。从上到

下依次使得 PC 口出现高电平 1，控制 8 只 LED 点亮，通过设定延时函数来控制 PC 口输出高电平的频率改变流水灯的流动快慢。

当 PC 口输出 0x01 时，D1 点亮，调用延时函数延时 1 秒后，PC 口输出 0x02，此时 D1 熄灭，D2 点亮……直到 D7 熄灭，D8 点亮，控制 8 只 LED 从上到下点亮一遍，此过程加在 while(1)死循环中，便可以满足任务要求。

2. 任务设计

根据任务要求，控制 D1～D8 点亮的控制字段分别为：0x01，0x02，0x04，0x08，0x10，0x20，0x40，0x80。定义一个数组 tab，将 8 个控制字段放在数组中，定义变量 k，使用 tab[k]独处数组中数据。

任务设计流程如图 2-1-14 所示。

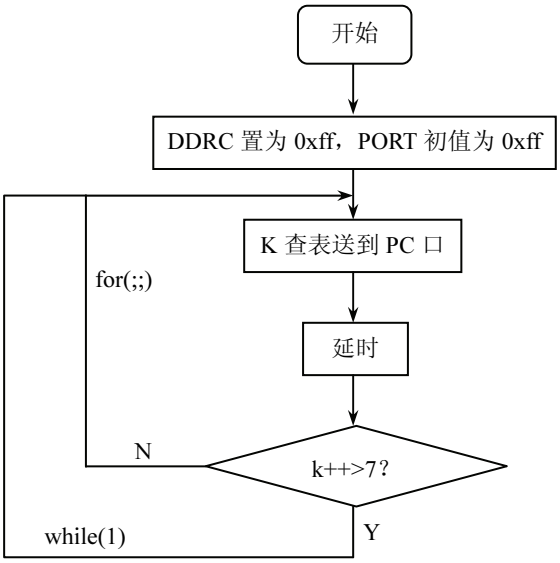

图 2-1-14　任务程序流程图

编写程序如下：

```
/****************************************************
 File name:              流水灯.c
 Chip type:              ATmega16
 Clock frequency:    8.0MHz
****************************************************/
#include <iom16v.h>
unsigned char tab[]={0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80};
void main(void)
{
    unsigned char k;
    DDRC=0xff;
    PORTC=0xff;
while(1)
{
     for(k=0;k<8;k++)
     {
       PORTC=tab[k];
```

```
            delay(1000);
        }
}
}
void delay(unsigned int ms)
{
            unsigned int i,j;
    for(i=0;i<ms;i++)
        {
        for(j=0;j<1140;j++);
        }
}
```

3．任务实现

（1）新建设计模板。

打开 ISIS 7 Professional 窗口，选择 File→New Design 命令，弹出模板选择对话框，如图 2-1-15 所示。图中纵向图纸为 Portrait，横向图纸为 Landscape，DEFAULT 为默认项。选中 DEFAULT，单击 OK 按钮，即新建好一个 DEFAULT 模板。

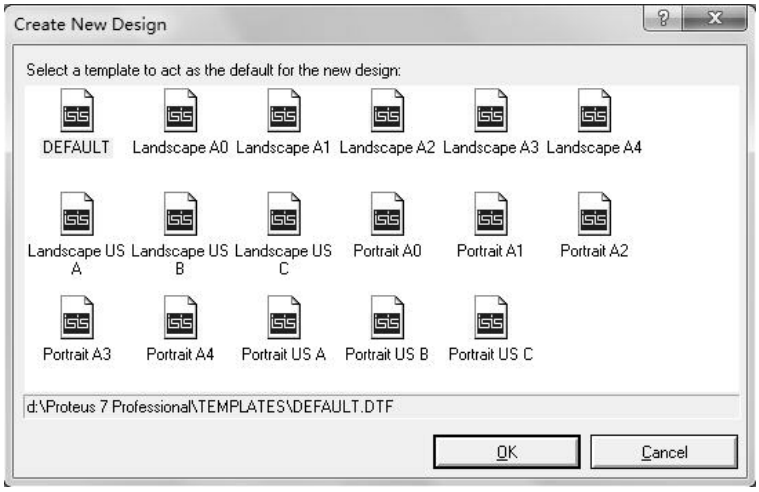

图 2-1-15　模板选择

（2）设定图纸大小。

选择 System→Set Sheet Sizes 命令，弹出对话框，在其中选中 A4 复选框，单击 OK 按钮，完成图纸设定，如图 2-1-16 所示。

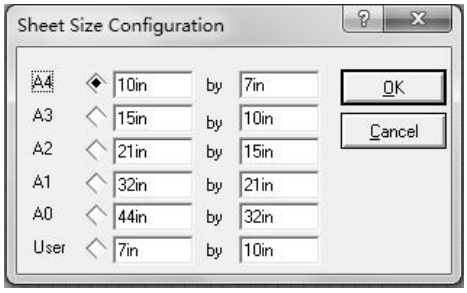

图 2-1-16　图纸选择

（3）元器件的添加。

本次设计图纸所使用的元器件如表 2-1-7 所示。

表 2-1-7　元器件列表

符号	中文	符号	中文	符号	中文
ATmega16	单片机	CAP22pF	瓷片电容	CRYSTAL 8MHz	晶振
CAP-ELEC 10μF	电解电容	LED-RED	发光二极管	LED-BLUE	发光二极管
LED-YELLOW	发光二极管	RES	电阻		

选择 Library→Pick Device/Symbol 命令，或者在器件选择按钮栏 P L DEVICES 中单击 P 按钮，弹出如图 2-1-17 所示的对话框。

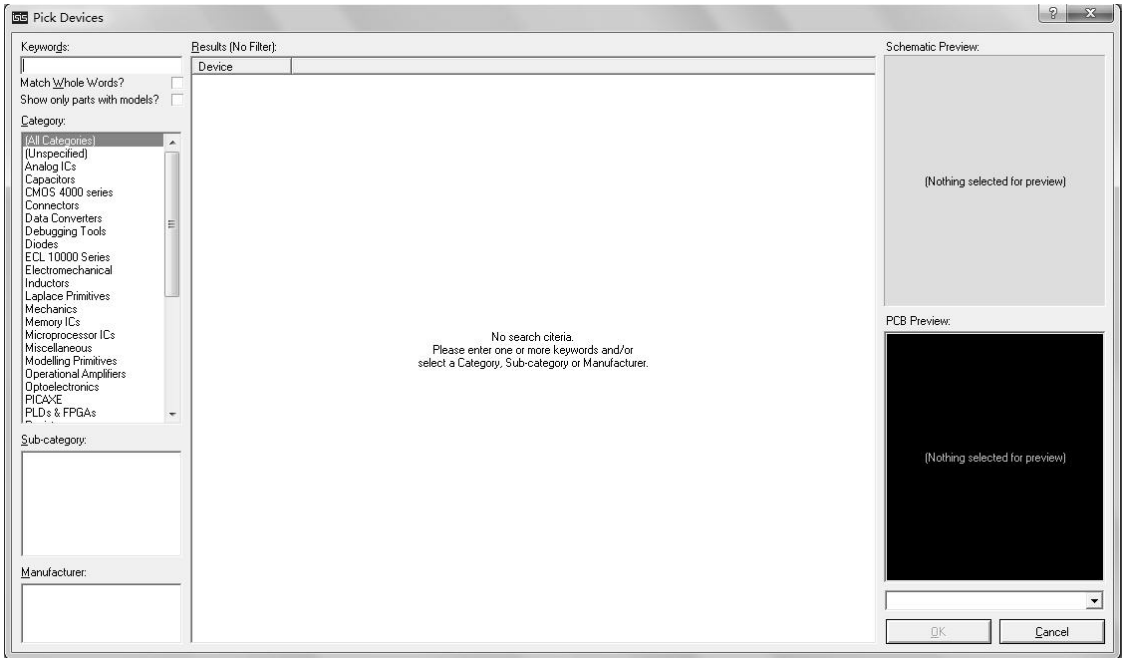

图 2-1-17　元器件选择

在关键字中输入元器件名称，如 ATmega16，则会出现与其相匹配的元器件列表，如图 2-1-18 所示。选中后双击 ATmega16 所在行，再单击 OK 按钮将 ATmega16 元器件加入到 ISIS 对象选择器中。按照上述方法，将其他元器件分别添加到 ISIS 对象选择器中。

（4）原理图绘制。

根据样图将所需元器件放置在图纸上，通过移动、旋转、布线等操作完成整个原理图，如图 2-1-19 所示。

（5）生成网络表并进行电气检测。

选择 Tools→Netlist Compiler 命令，弹出如图 2-1-20 所示的对话框，在其中可以设置网络表的输出形式、模式等，此处不进行修改，单击 OK 按钮以默认方式输出如图 2-1-21 所示的内容。

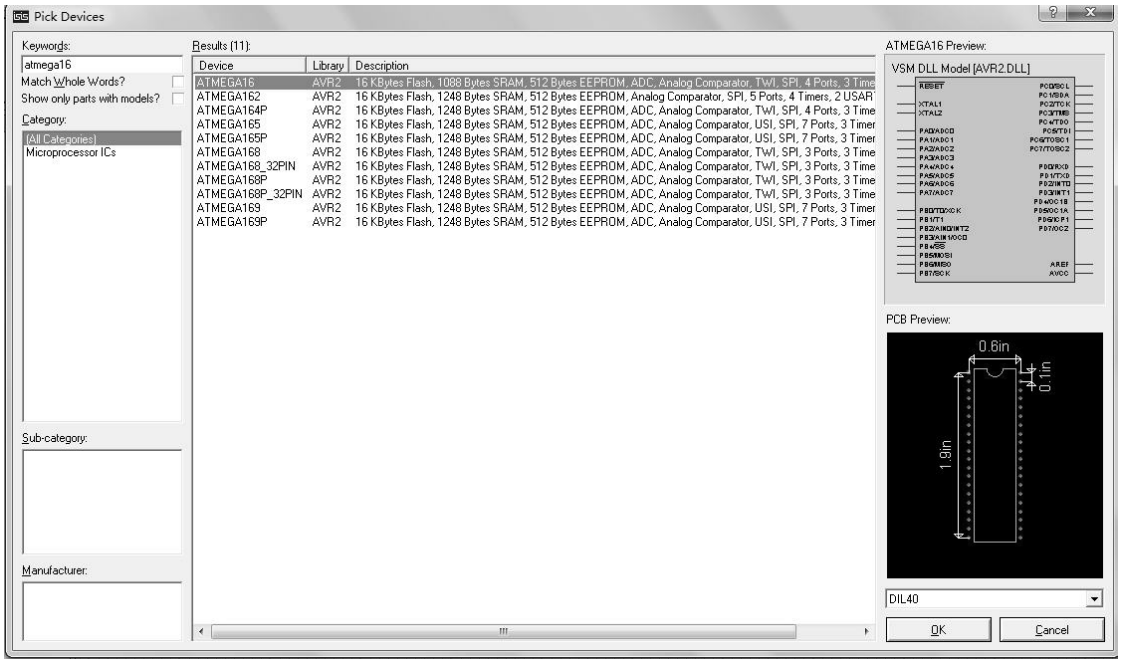

图 2-1-18　输入元器件名称

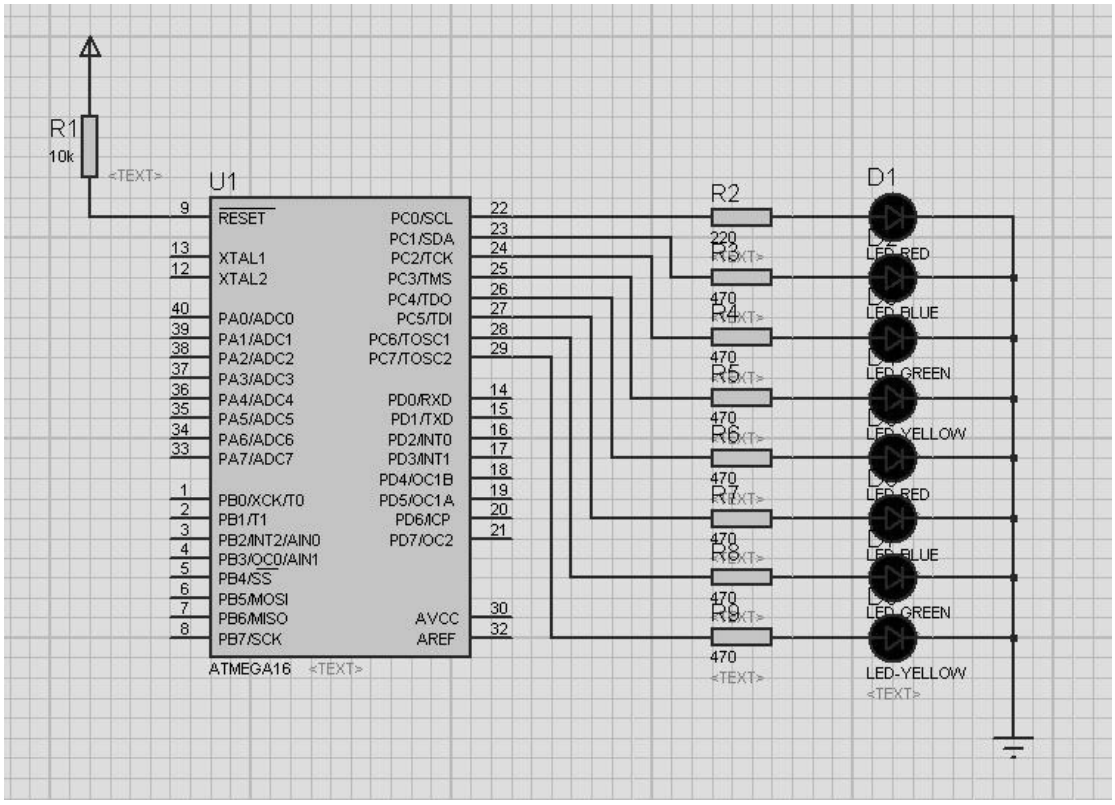

图 2-1-19　原理图

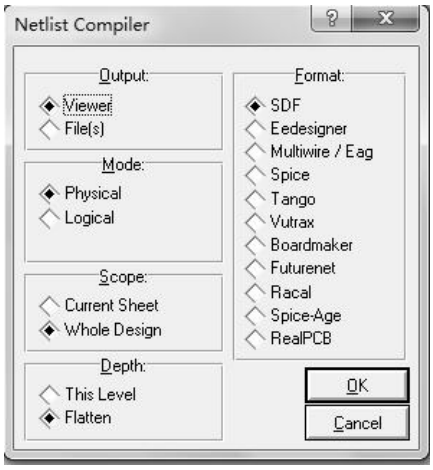

图 2-1-20　网络表设置

图 2-1-21　输出网络表

电路图画完并生成网络表后，可以进行电气检测，选择 Tools→Electrical Rule Check 命令，弹出如图 2-1-22 所示的电气检测窗口，从中可以看到无电气错误。

图 2-1-22　电气检测

4. 任务运行

（1）载入。

打开 ATmega16 单片机的属性设置对话框，找到 Program File 选项，如图 2-1-23 所示。载入 ICCAVR 或 CodeVisionAVR 生成的 CHENGXU2.cof 文件或 CHENGXU2.hex 文件，如图 2-1-24 所示。

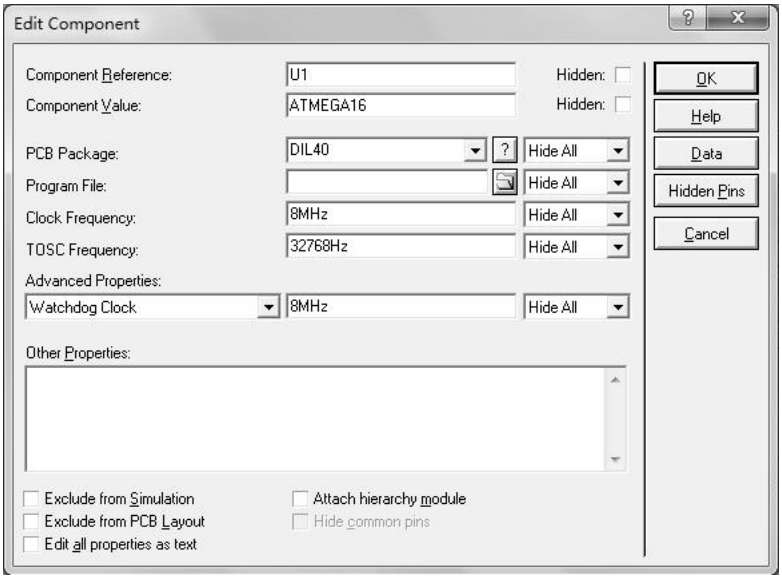

图 2-1-23　单片机属性设置

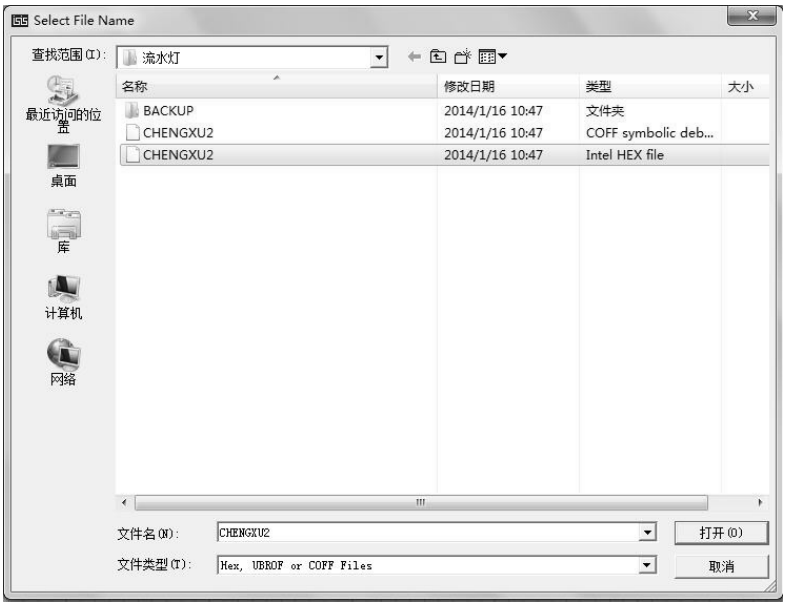

图 2-1-24　载入文件

（2）仿真。

单击 Proteus 的运行按钮，观察仿真现象，如图 2-1-25 和图 2-1-26 所示。

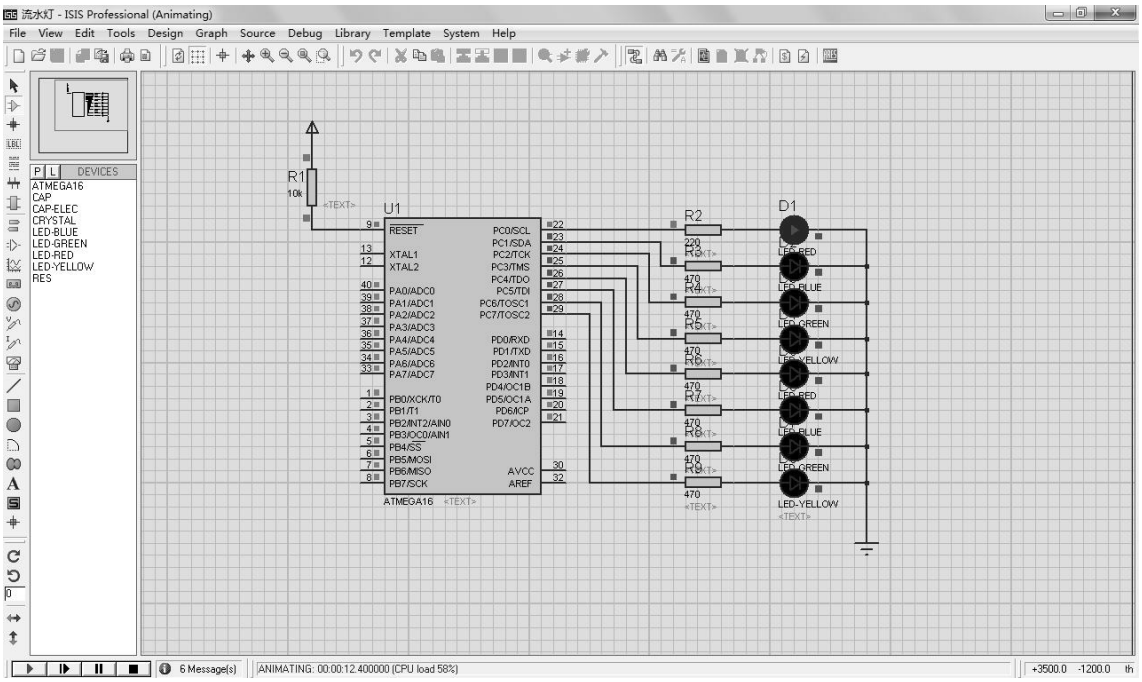

图 2-1-25　运行后按下开关 D1 亮

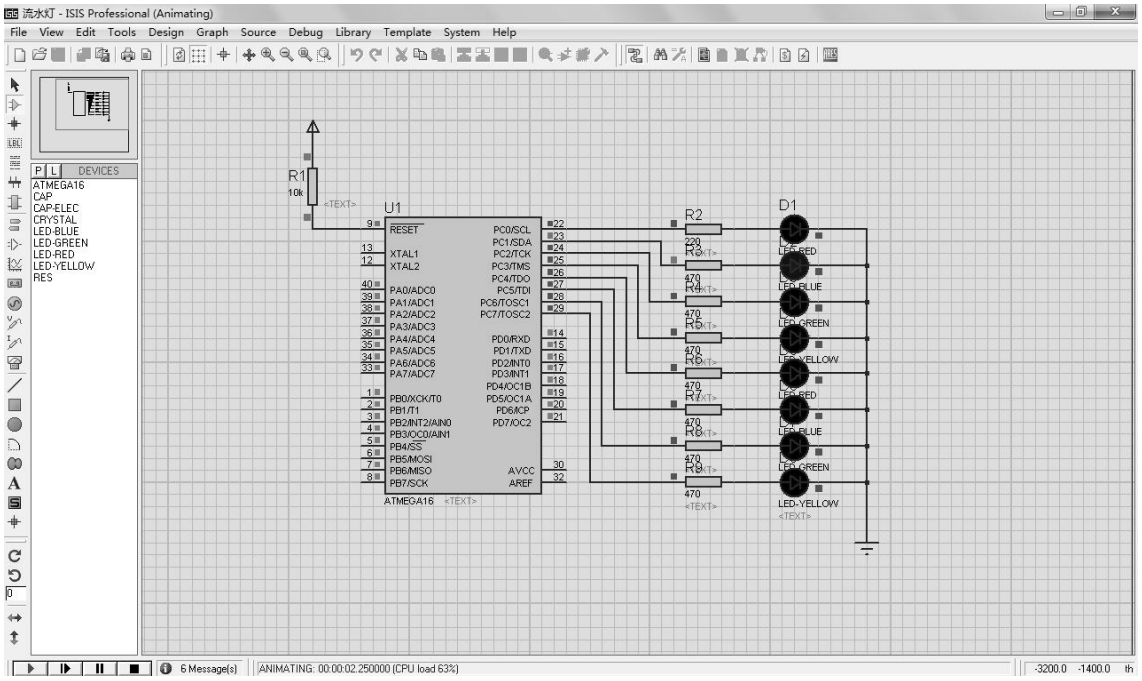

图 2-1-26　D1 熄灭后 D2 点亮

1.4 任务总结

通过闪烁灯和流水灯这两个任务的学习我们要明确以下四点：

- C 语言是一款优秀的单片机程序开发高级语言，有着灵活、方便、快捷等特点。
- 注意延时函数、数组的使用方法。实际应用中可以用不同的编程方法完成同一个任务。
- ATmega16 四组 I/O 口使用前要先确定输入还是输出，然后确定高低电平。
- Proteus 软件在整个操作过程中有着至关重要的作用，虚拟仿真在单片机开发过程中有很好的效果，可以缩短程序开发周期。