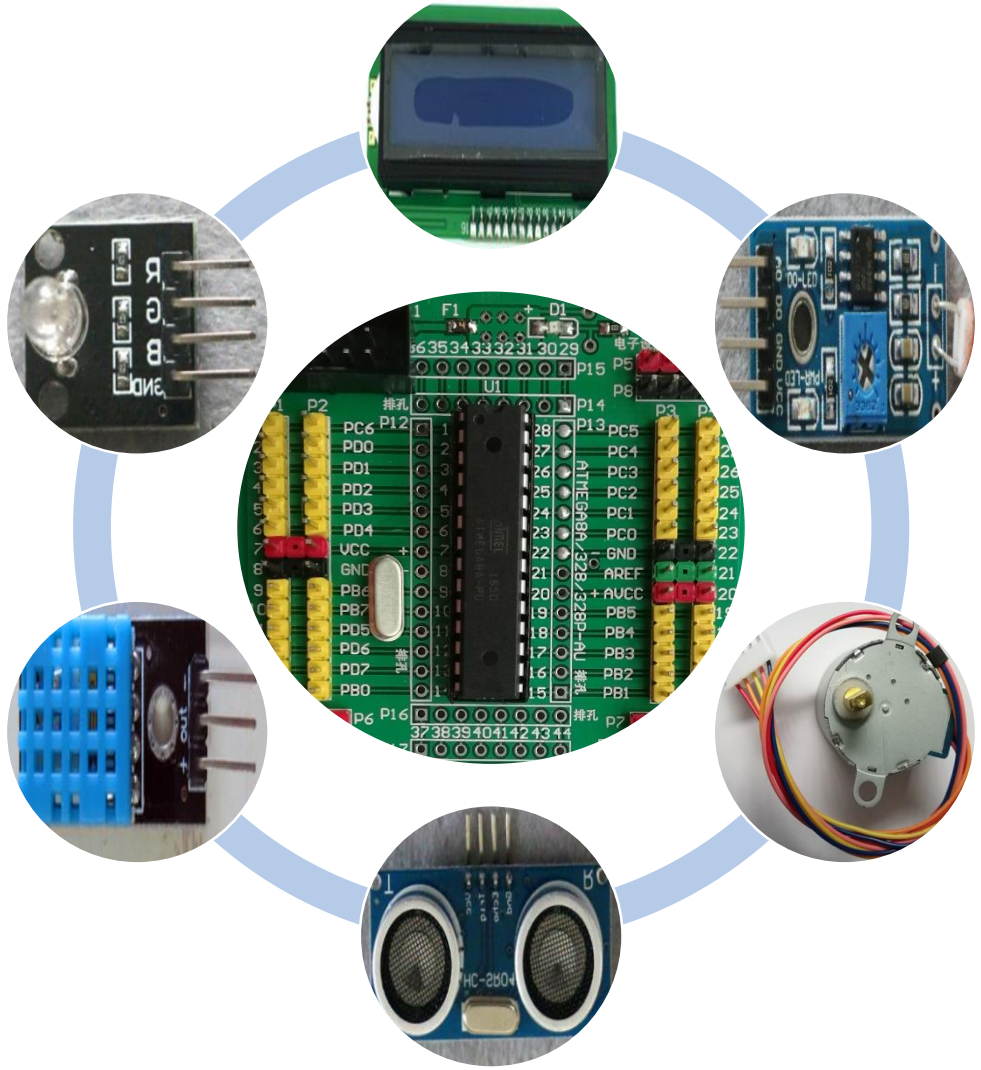
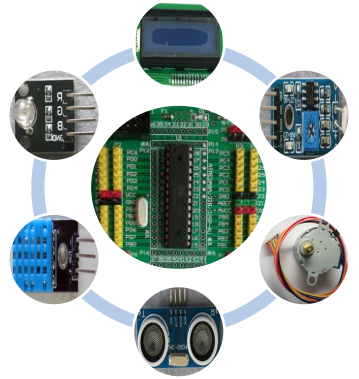
电子设计实践基础

MCU定时器/计数器与PWM、步进电机、直流电机及其编程



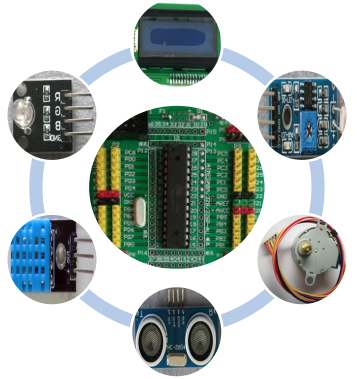
上节课内容回顾

- 温湿度传感器
- 超声波传感器
- MCU ADC与光敏电阻



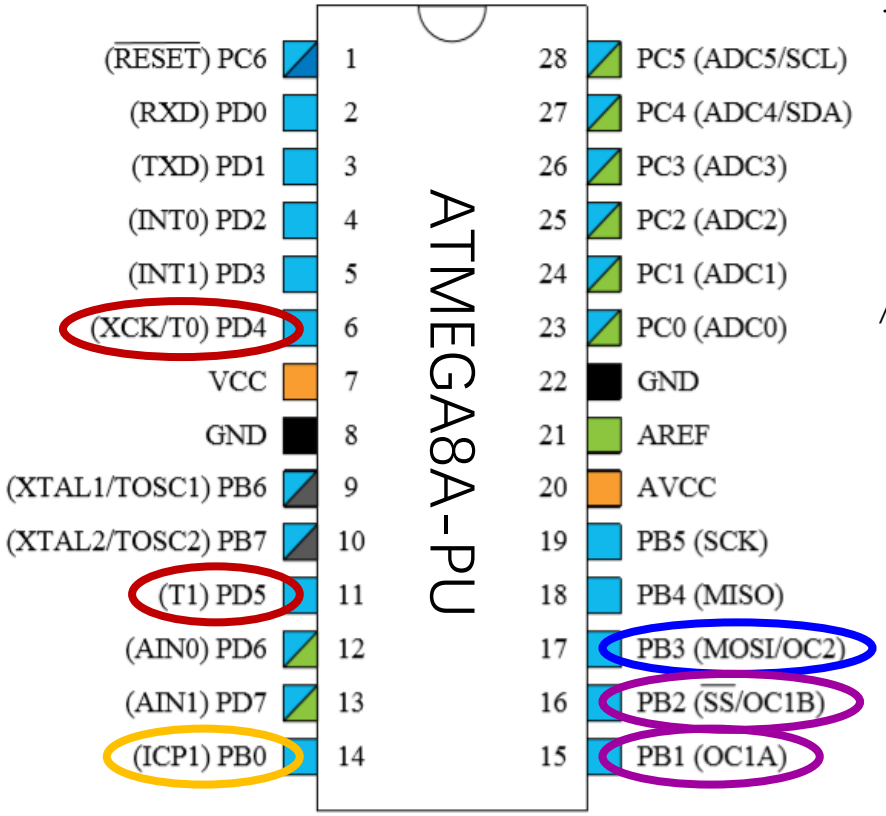
本节课主要内容

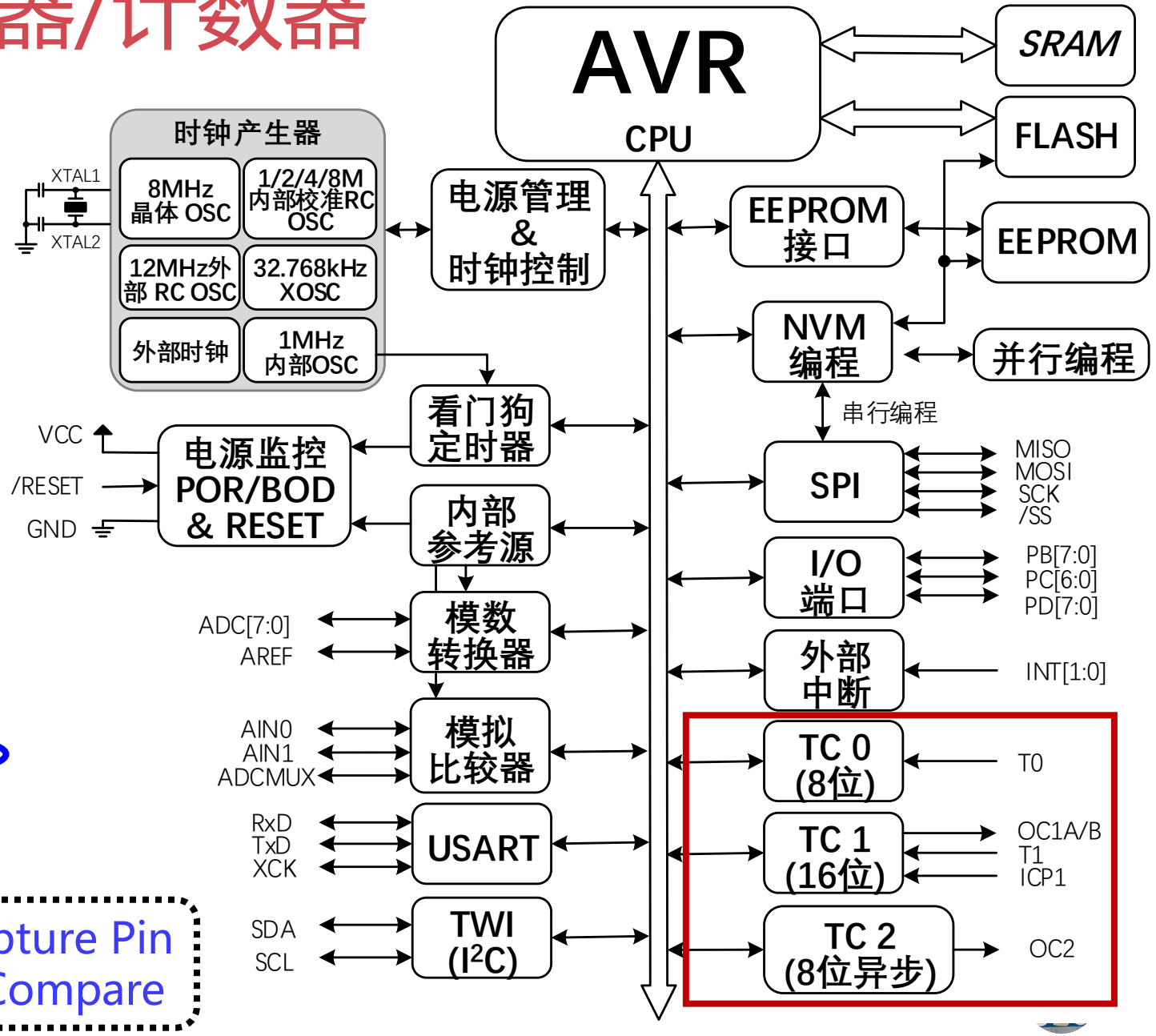
- ATMEGA8A的定时器/计数器
- PWM
- 步进电机
- 直流电机



ATmega8A定时器/计数器

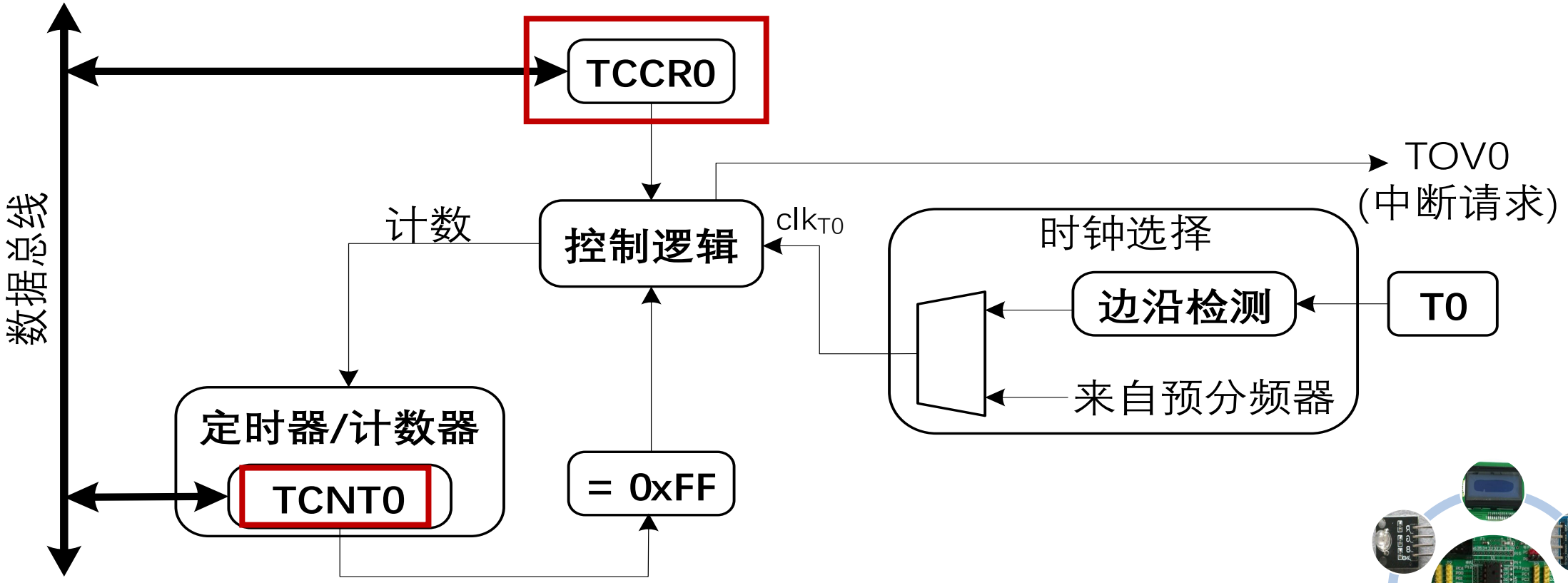
Timer/Counter

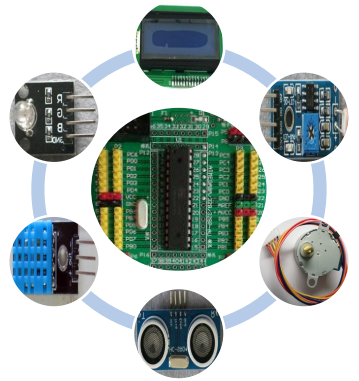




ICP-Input Capture Pin
OC-Output Compare

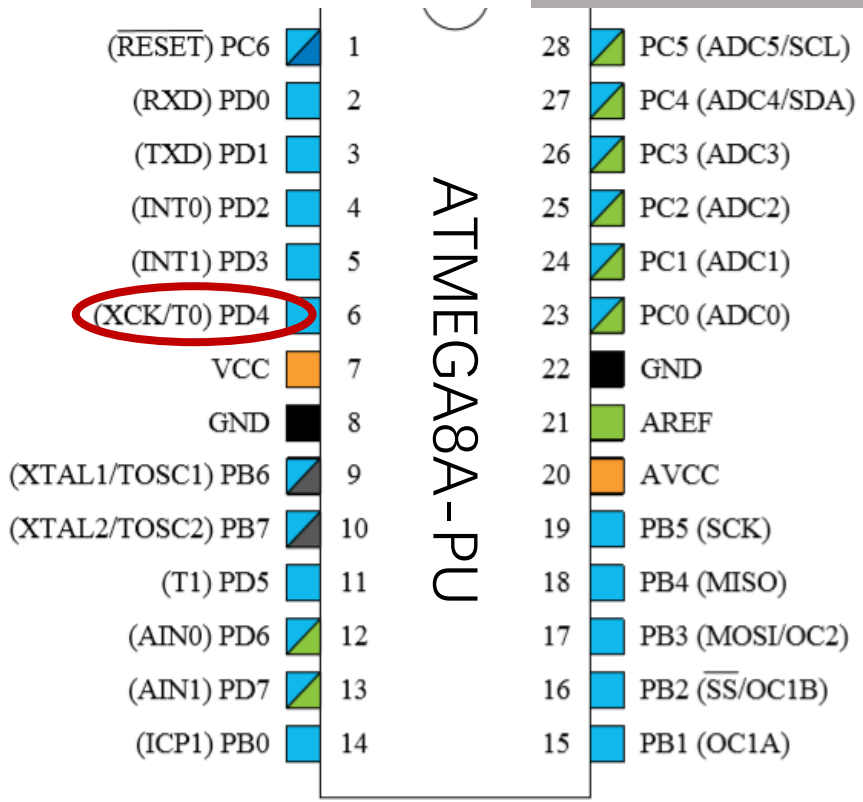
ATmega8A 定时器/计数器0(TC0)的结构



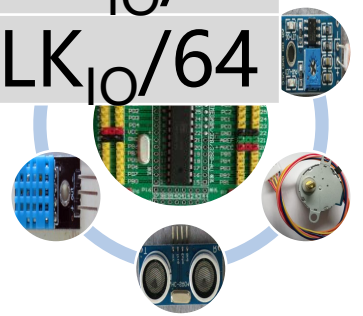


定时器/计数器0(TC0)的控制寄存器

Bit	7	6	5	4	3	2	1	0
TCCR0						CS02	CS01	CS00
Access						R/W	R/W	R/W
Reset						0	0	0



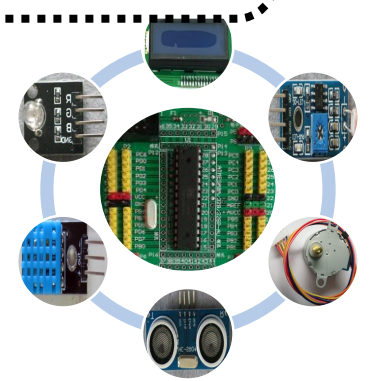
CS0[2:0]	TC0时钟	CS0[2:0]	TC0时钟
100	$CLK_{IO}/256$	000	无，停止
101	$CLK_{IO}/1024$	001	CLK_{IO}
110	T0,下降沿	010	$CLK_{IO}/8$
111	T0,上升沿	011	$CLK_{IO}/64$



定时器/计数器0(TC0)的计数寄存器

Bit	7	6	5	4	3	2	1	0
TCNT0	TCNT0[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

TCNT0：8位的计数寄存器，只能加计数，溢出后（0xFF）从头（0x0）开始加计数；CPU可直接读写，或设置其初值/读取当前值



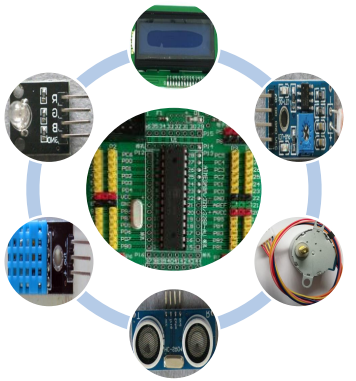
定时器中断屏蔽（TIMSK）寄存器

TIMSK-Timer Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0
TIMSK	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1		TOIE0
Access	R/W	R/W	R/W	R/W	R/W	R/W		R/W
Reset	0	0	0	0	0	0		0

定时器/计数器0溢出中断允许，TOIE0为1、且SREG寄存器的I位也为1时[sei()/cli();]开启TOV0中断

TOIE0-Timer0 Overflow Interrupt Enable



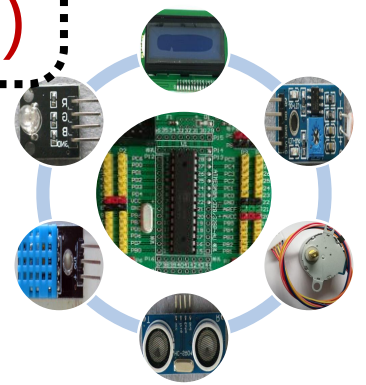
定时器中断标志(TIFR)寄存器

TIFR-Timer Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0
TIFR	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1		TOV0
Access	R/W	R/W	R/W	R/W	R/W	R/W		R/W
Reset	0	0	0	0	0	0		0

定时器/计数器0溢出时TOV0=1，当执行相应的中断处理程序后TOV0自动清零，也可以写1到此位进行清零(手动)

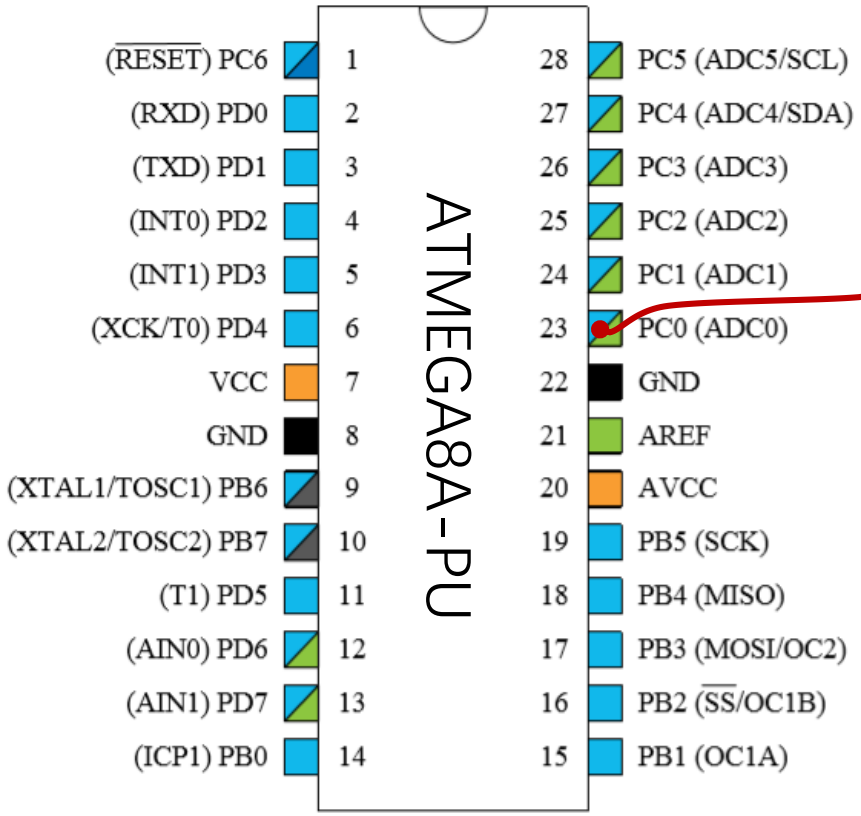
TOV0-Timer0 Overflow Flag

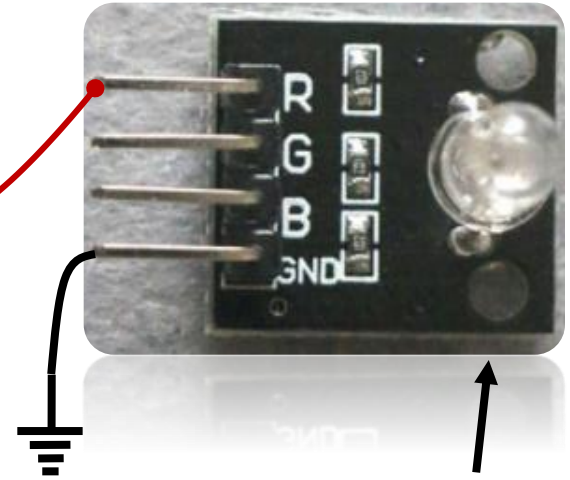


定时器/计数器0(TC0)的示例（1）

用TC0的中断去控制LED灯的闪烁

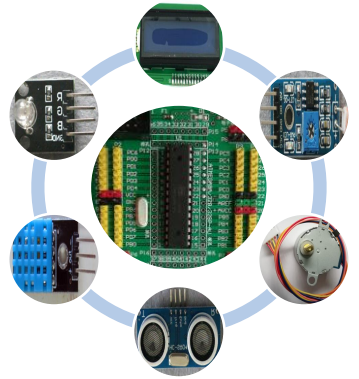
实验内容1





闪烁

1st 内部中断

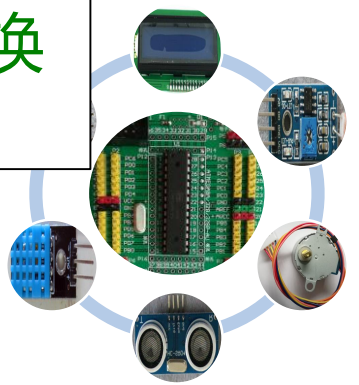


定时器/计数器0(TC0)的示例编程（2）

```
#include <avr/io.h>
#include <avr/interrupt.h>
int main(void)
{    DDRC |=(1<<DDRC0);//PC0为输出到LED正极
     TCCR0 = (1<<CS02)|(1<<CS00);//1024分频
     TCNT0 = 0;//从0开始计数
     TIMSK |=(1<<TOIE0);//允许TOV0中断
     sei();//全局中断开
    while (1)
    {
    }
}
```

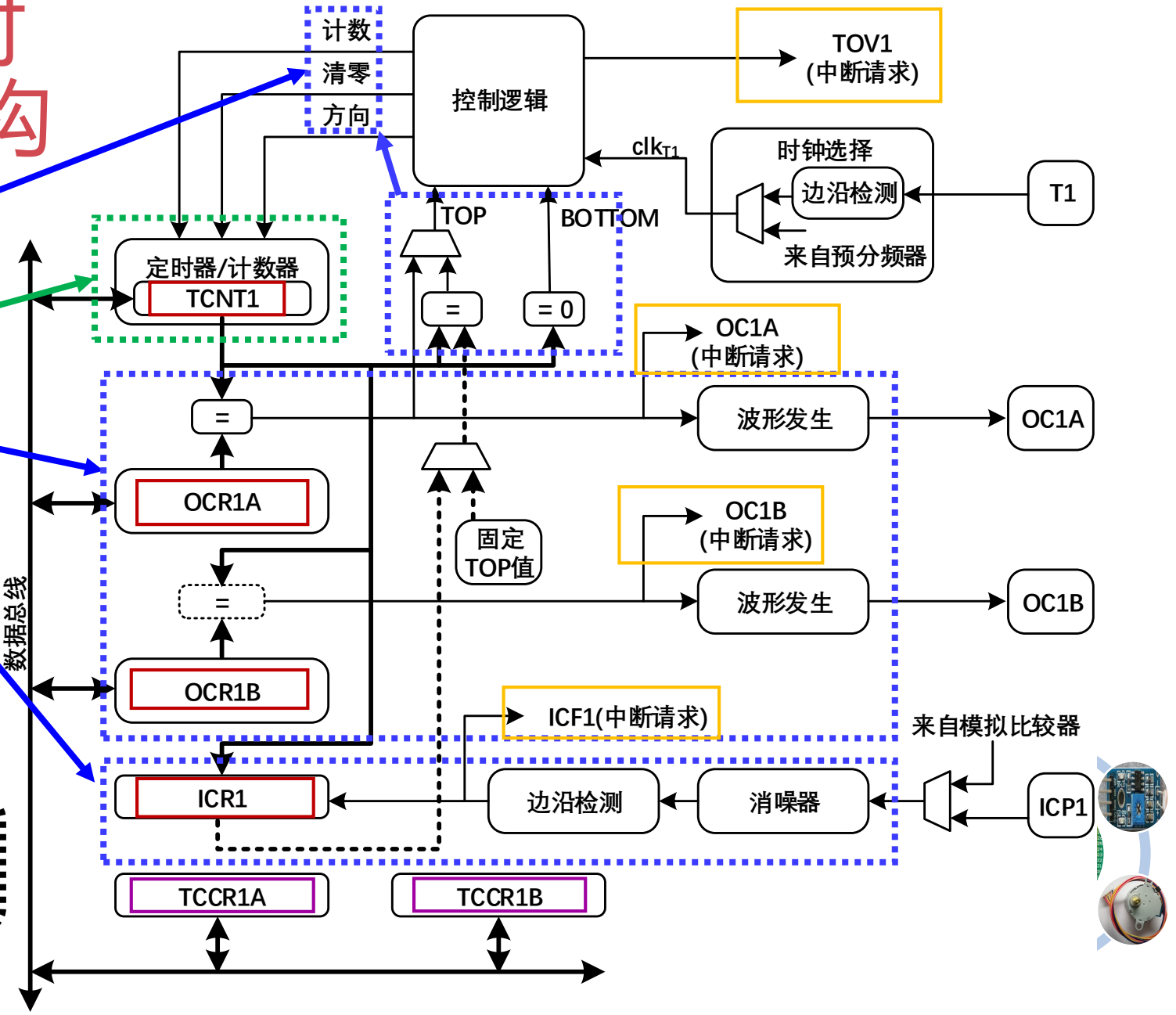
CS0[2:0]	TC0时钟
100	$CLK_{IO}/256$
101	$CLK_{IO}/1024$
110	T0,下降沿
111	T0,上升沿

```
ISR(TIMER0_OVF_vect)
{ PORTC ^=(1<<PORTC0);//输出高低电平切换
}
```



ATmega8A 定时器/计数器1：结构

- 带清零双向计数器
- 16位计数器
- 2路比较输出:16位PWM
- 1路输入捕获:时间戳
- 4个独立的内部中断源
- TCNT1,OCR1A/B,ICR1均为16位寄存器，TCCR1A/B为8位



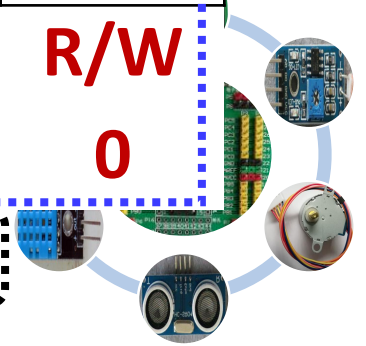
ATmega8A 定时器/计数器1控制寄存器（1）

Bit	7	6	5	4	3	2	1	0
TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

比较输出模式　　强制比较输出　　PWM模式

Bit	7	6	5	4	3	2	1	0
TCCR1B	ICNC1	ICES1		WGM13	WGM12	CS12	CS11	CS10
Access	R/W	R/W		R/W	R/W	R/W	R/W	R/W
Reset	0	0		0	0	0	0	0

输入捕获　　PWM模式　　工作时钟

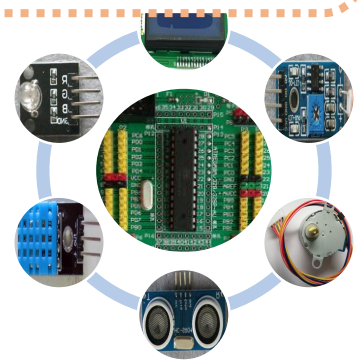


ATmega8A 定时器/计数器1控制寄存器（2）

CS12	CS11	CS10	说明
0	0	0	无时钟源(定时器停止)
0	0	1	$clk_{I/O}$(没用预分频)
0	1	0	$clk_{I/O}$ /8(用预分频)
0	1	1	$clk_{I/O}$ /64(用预分频)
1	0	0	$clk_{I/O}$ /256(用预分频)
1	0	1	$clk_{I/O}$ /1024(用预分频)
1	1	0	外部时钟源T1,下降沿
1	1	1	外部时钟源T1,上升沿

ICNC1:输入捕获噪声衰减，
=1时，过滤ICP1管脚
ICES1:输入捕获边沿选择，
=0时，下降沿(0)触发，
=1时，上升沿(1)触发

FOC1A/B：强制比较输出，
非PWM模式下用。



TC1控制寄存器(3)

模式	WGM[13:10]				操作模式	TOP	OCR1x更新点	TOV1置位点
0	0	0	0	0	普通-Normal	0xFFFF	立即	MAX
1	0	0	0	1	PWM,相位校正,8位	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM,相位校正,9位	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM,相位校正,10位	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC【CTCM】	OCR1A	立即	MAX
5	0	1	0	1	快速PWM,8位	0x00FF	BOTTOM	TOP
6	0	1	1	0	快速PWM,9位	0x01FF	BOTTOM	TOP
7	0	1	1	1	快速PWM,10位	0x03FF	BOTTOM	TOP
8	1	0	0	0	PWM,相位与频率校正	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	PWM,相位与频率校正	OCR1A	BOTTOM	BOTTOM
10	1	0	1	0	PWM,相位校正	ICR1	TOP	BOTTOM
11	1	0	1	1	PWM,相位校正	OCR1A	TOP	BOTTOM
12	1	1	0	0	CTC	ICR1	立即	MAX
13	1	1	0	1	保留	-	-	-
14	1	1	1	0	快速PWM	ICR1	BOTTOM	TOP
15	1	1	1	1	快速PWM	OCR1A	BOTTOM	TOP

ATmega8A 定时器/计数器1控制寄存器（4）

非PWM模式

COM1(A/B)1	COM1(A/B)0	说明
0	0	普通端口操作,OC1A/OC1B断开
0	1	比较匹配时OC1A/OC1B翻转切换
1	0	比较匹配时OC1A/OC1B清零（低电平）
1	1	比较匹配时OC1A/OC1B置位（高电平）

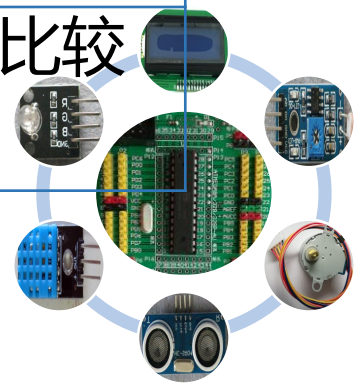
快速PWM模式

COM1(A/B)1	COM1(A/B)0	说明
0	0	普通端口操作,OC1A/OC1B断开
0	1	WGM[13:10]=15：比较匹配时OC1A翻转，OC1B断开。其它WGM1设置，普通操作
1	0	比较匹配时OC1A/OC1B清零，在BOTTOM置位OC1A/OC1B（非反向模式）
1	1	比较匹配时OC1A/OC1B置位(1)，在BOTTOM清零OC1A/OC1B（反向模式）

ATmega8A 定时器/计数器1控制寄存器（5）

相位校正与相位和频率校正PWM模式

COM1A1/ COM1B1	COM1A0/ COM1B0	说明
0	0	普通端口操作,OC1A/OC1B断开
0	1	WGM[13:10]=9/14：比较匹配时OC1A翻转，OC1B断开。其它WGM1，普通操作
1	0	加计数比较匹配时OC1A/OC1B清零，减计数比较匹配时OC1A/OC1B置位(1)。
1	1	加计数比较匹配时OC1A/OC1B置位，减计数比较匹配时OC1A/OC1B清零。

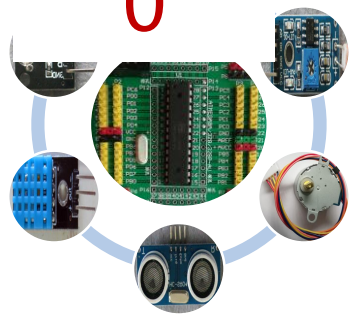


ATmega8A 定时器/计数器1 计数寄存器

Bit	7	6	5	4	3	2	1	0
TCNT1L	TCNT1L[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
TCNT1H	TCNT1H[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

读：先L后H，写先H后L，编程时可直接使用TCNT1进行16位操作，由编译器完成访问顺序

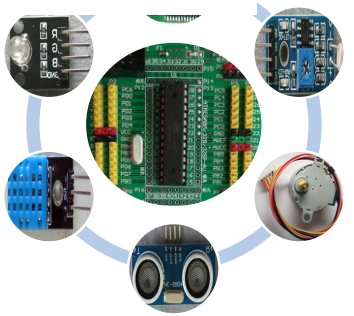


ATmega8A 定时器/计数器1输出比较寄存器A

Bit	7	6	5	4	3	2	1	0
OCR1AL	OCR1AL[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
OCR1AH	OCR1AH[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

读：先L后H，写先H后L，编程时可直接使用OCR1A进行16位操作，由编译器完成访问顺序

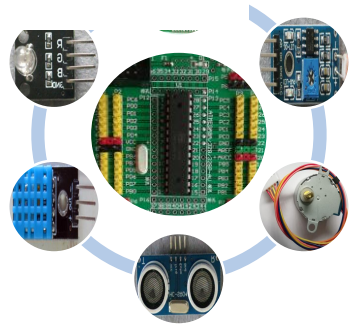


ATmega8A 定时器/计数器1输出比较寄存器B

Bit	7	6	5	4	3	2	1	0
OCR1BL	OCR1BL[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
OCR1BH	OCR1BH[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

读：先L后H，写先H后L，编程时可直接使用OCR1B进行16位操作，由编译器完成访问顺序

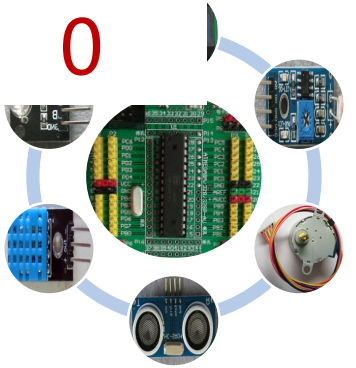


ATmega8A 定时器/计数器1输入捕获寄存器

Bit	7	6	5	4	3	2	1	0
ICR1L	ICR1L[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
ICR1H	ICR1H[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

读：先L后H，写先H后L，编程时可直接使用ICR1进行16位操作，由编译器完成访问顺序

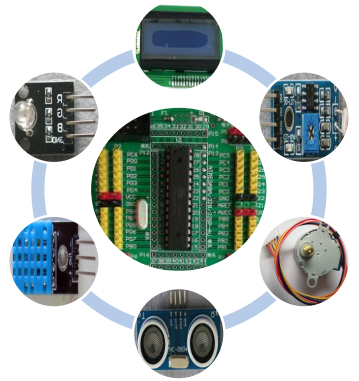


ATmega8A 定时器中断屏蔽寄存器TIMSK

TIMSK-Timer Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0
TIMSK	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1		TOIE0
Access	R/W	R/W	R/W	R/W	R/W	R/W		R/W
Reset	0	0	0	0	0	0		0

开启定时器/计数器1内部的四个独立中断之一

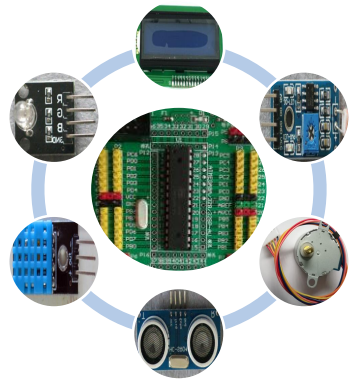


ATmega8A 定时器中断标志寄存器TIFR

TIFR-Timer Interrupt Flag Register

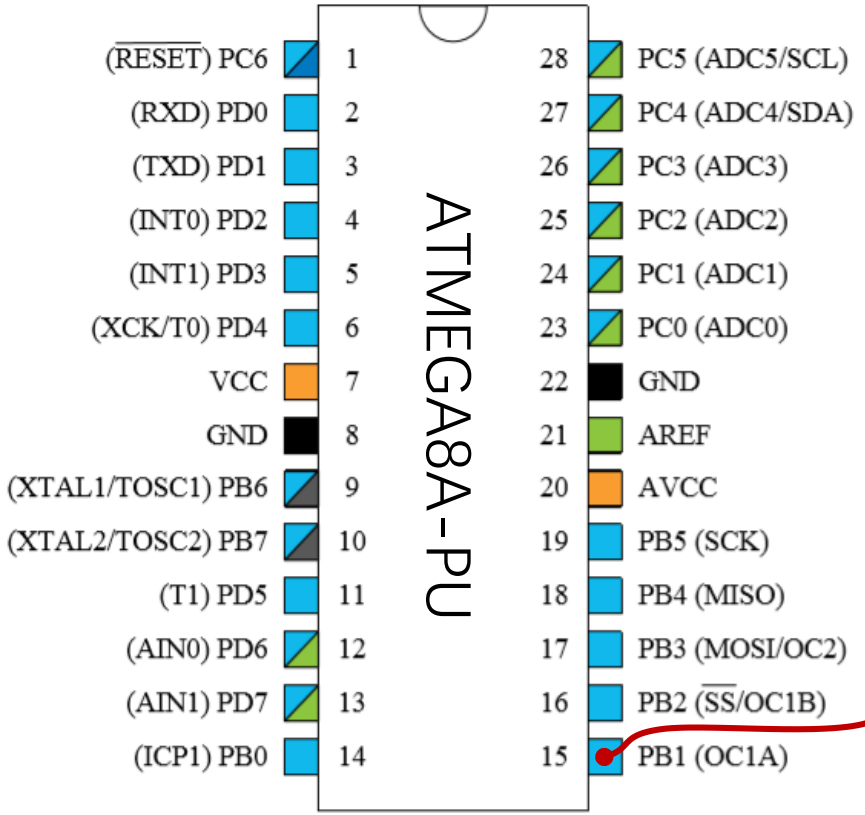
Bit	7	6	5	4	3	2	1	0
TIFR	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1		TOV0
Access	R/W	R/W	R/W	R/W	R/W	R/W		R/W
Reset	0	0	0	0	0	0		0

定时器/计数器1内部四个独立中断的标志位，如为1时，当执行相应的中断处理程序后自动清零，也可以写1到此位进行清零(查询方式)



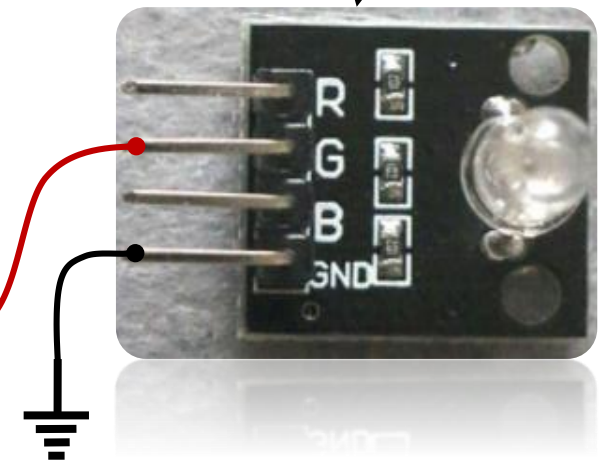
ATmega8A 定时器/计数器1示例

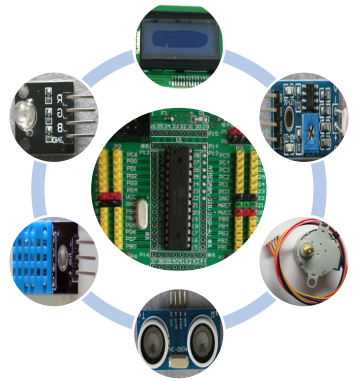
用TC1的比较输出A（OC1A）控制LED灯的闪烁



实验内容2

闪烁





ATmega8A 定时器/计数器1示例编程

```
#include <avr/io.h>
int main(void)
{
    DDRB |=(1<<DDRB1);//PB1(OC1A)为输出
    ICR1 =2048;//设置TOP值
    OCR1A = 1024;//设置A路比较值
    TCCR1A |= (1<<COM1A1);//设置输出比较管脚切换方式等
    TCCR1B |= (1<<WGM13)|(1<<CS12);//设置PWM模式为8，时钟256分频
   while (1)
   {
   }
}
```

定时器/计数器1的计数比较输出功能设置好以后，自动工作，不占用CPU时间

CS1[2:0]	TC1时钟
100	$CLK_{IO}/256$

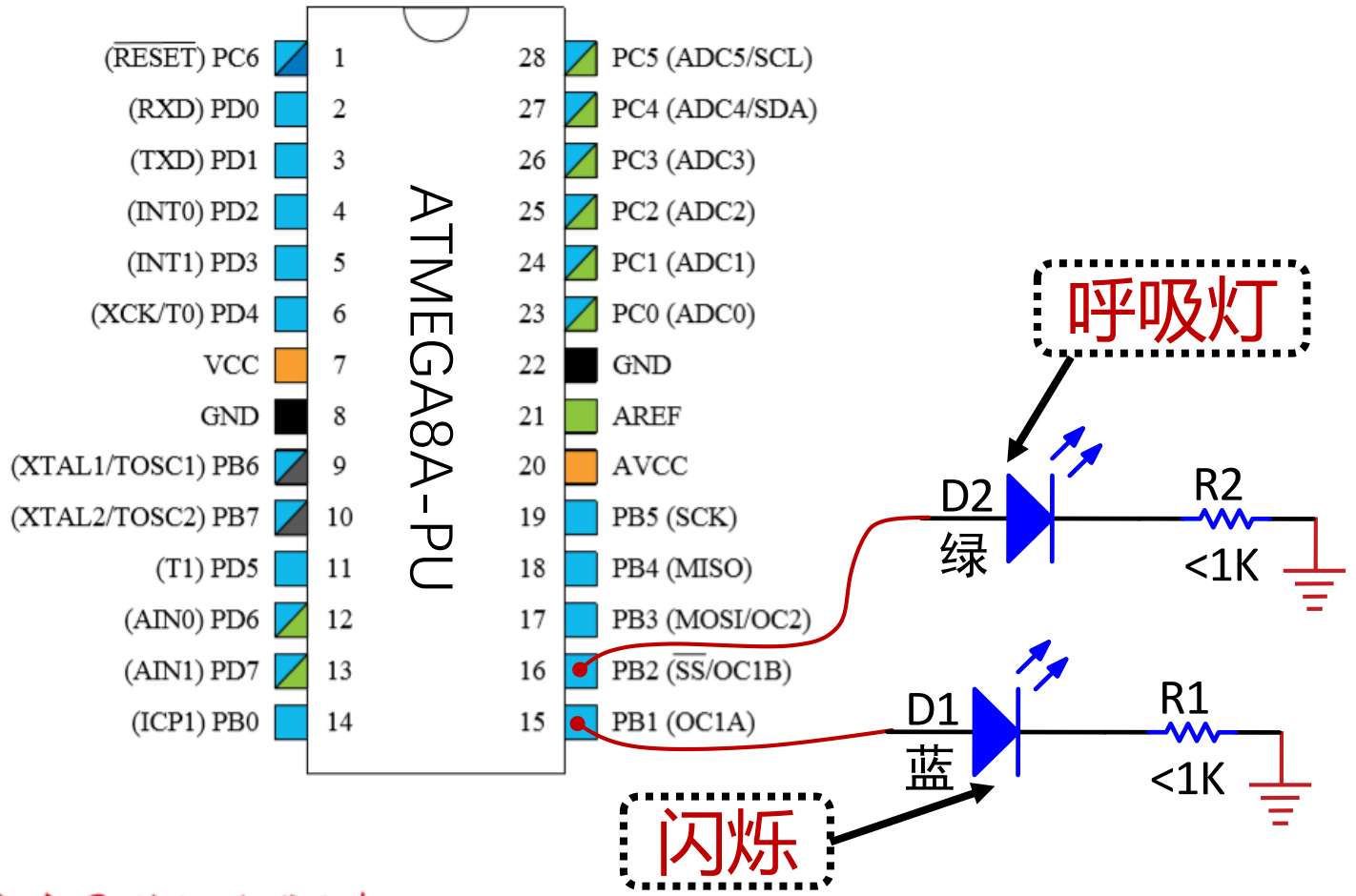
COM1A1/COM1B1	COM1A0/COM1B0	说明
1	0	加计数比较匹配时OC1A/OC1B清零， 减计数比较匹配时OC1A/OC1B置位(1)

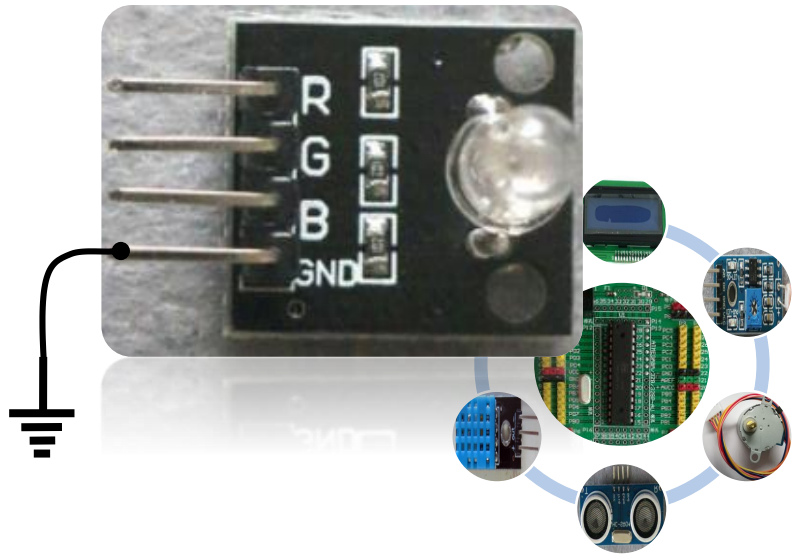
模式	WGM[13:10]				操作模式	TOP	OCR1x更新点	TOV1置位点
8	1	0	0	0	PWM,相位与频率校正	ICR1	BOTTOM	BOTTOM

ATmega8A 定时器/计数器1示例2

用TC1的比较输出A和B(OC1A和OC1B)分别控制两个LED灯的亮灭

实验内容3



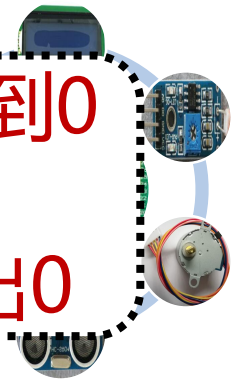


ATmega8A 定时器/计数器1示例2编程（1）

```
int main(void)
{
   DDRB |=(1<<DDRB1)|(1<<DDRB2);//PB1(OC1A),PB2(OC1B)为输出
   ICR1 = 500;//设置TOP值
   OCR1A = 1;//设置A路比较值
   OCR1B = 1;//设置B路比较值
   TCCR1A |= (1<<COM1A1)|(1<<COM1A0)|(1<<COM1B1)|(1<<COM1B0);//
设置’输出比较’管脚切换方式等
   TCCR1B |= (1<<WGM13)|(1<<CS11);//设置PWM模式为8，时钟8分频
   TIMSK = (1<<TOIE1);//开启TC1溢出中断
   sei();//开启全局中断
    while (1)
    {   }
}
```

```
#include <avr/io.h>
#include <avr/interrupt.h>
```

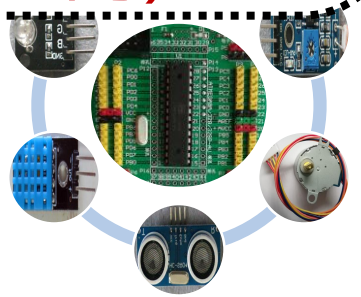
TC1计数：TCNT1从0计到ICR1，再从ICR1计到0
A/B路比较器在加计数时TCNT1=OCR1A/B，
OC1A/B输出1，减计数时相等，OC1A/B输出0



ATmega8A 定时器/计数器1示例2编程（2）

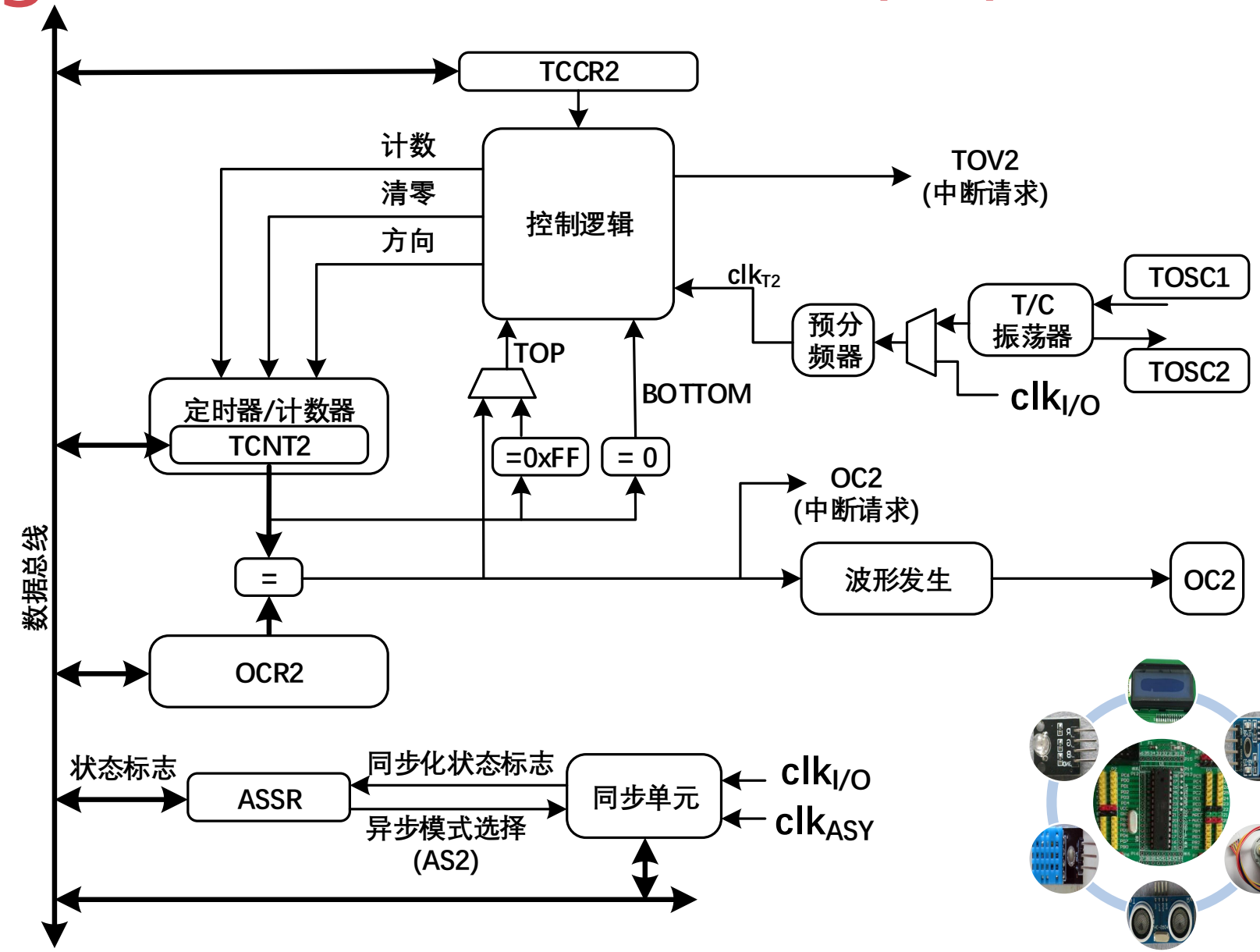
```
unsigned char ui_cnt=0;//统计中断次数，确定OCR1A/B的值
ISR(TIMER1_OVF_vect)
{   ui_cnt++;//统计TC1溢出中断次数
    if(ui_cnt>249)ui_cnt=0;//统计到250次时清零
    if(ui_cnt<125)//前半程
    {   OCR1A = 0;//高电平的持续时间长
        OCR1B +=4;//高电平的持续时间渐短
    }
    else//后半程
    {   OCR1A =500;//低电平的持续时间长
        OCR1B -=4;//高电平的持续时间渐长
    }
}
```

TC1计数溢出时(Bottom)，产生TOV1溢出中断。在TOV1 ISR中改变OCR1A/B的值，从而改变下一次OC1A/B匹配输出时间(高/低电平的时间)



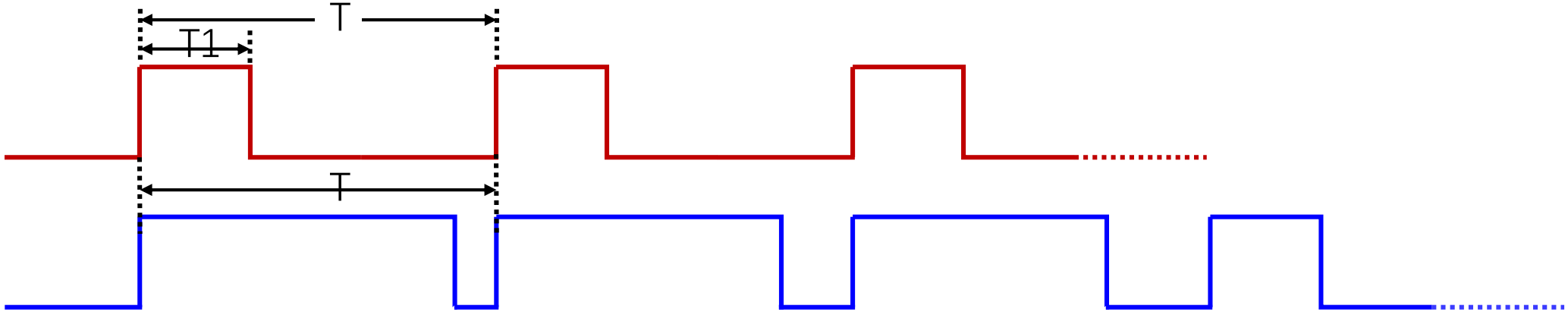
ATmega8A 定时器/计数器2（略）

TC2：8位、1路PWM、可异步操作

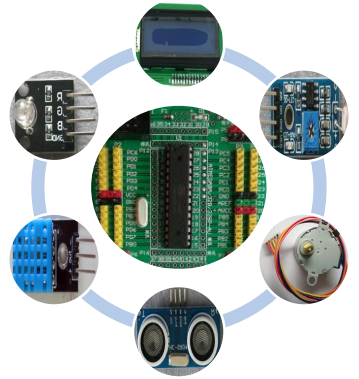


PWM

- PWM：Pulse Width Modulation
- 改变脉冲宽度（占空比：高电平宽度/周期）



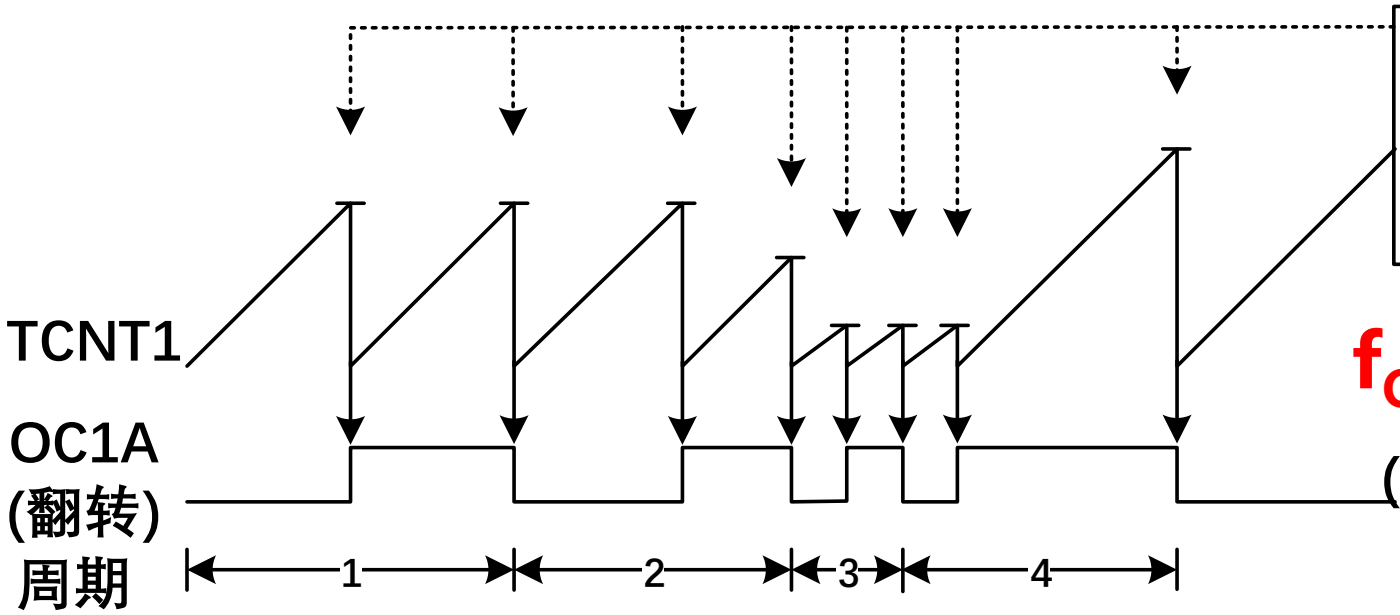
灯光亮度、电机速度、
开关电源。。。



TC1 工作模式:CTC

CTC-Clear Timer on Compare match

模式	WGM[13:10]				操作模式	TOP	OCR1x更新点	TOV1置位点
4	0	1	0	0	CTC	OCR1A	立即	MAX
12	1	1	0	0	CTC	ICR1	立即	MAX



OC1A或ICF1中断标志置位(在TOP上中断)

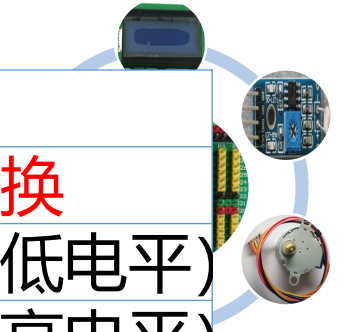
$$f_{OC1A} = \frac{f_{clk\_I/O}}{2 \times N \times (1 + OCR1A)}$$

(COM1A1:0=1)

N-预分频

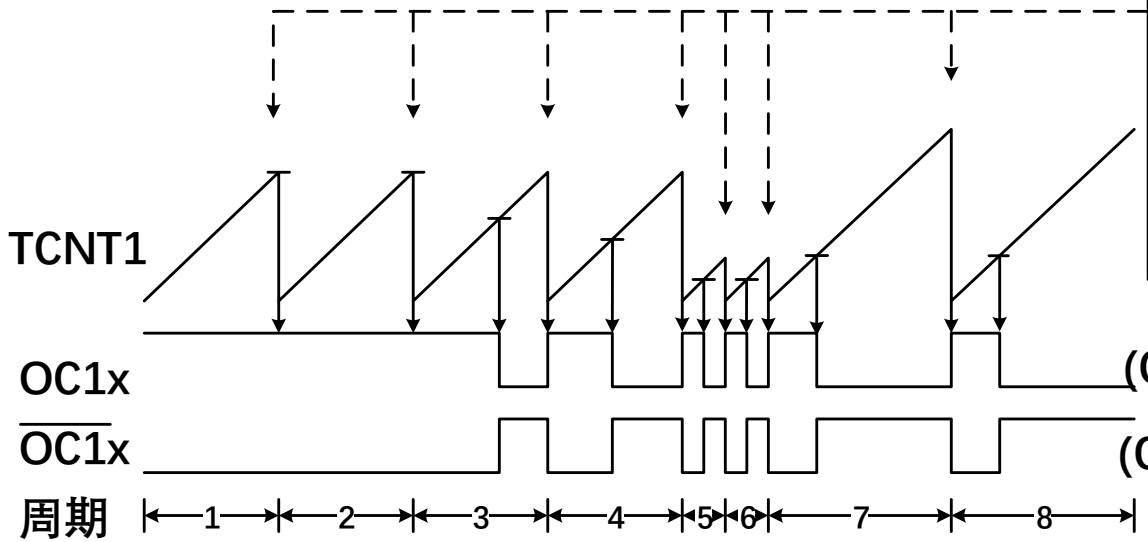
非PWM模式

COM1(A/B)1	COM1(A/B)0	说明
0	1	比较匹配时OC1A/OC1B翻转切换
1	0	比较匹配时OC1A/OC1B清零（低电平）
1	1	比较匹配时OC1A/OC1B置位（高电平）



TC1 工作模式：快速PWM

模式	WGM[13:10]				操作模式	TOP	OCR1x更新点	TOV1置位点
5	0	1	0	1	快速PWM,8位	0x00FF	BOTTOM	TOP
6	0	1	1	0	快速PWM,9位	0x01FF	BOTTOM	TOP
7	0	1	1	1	快速PWM,10位	0x03FF	BOTTOM	TOP
14	1	1	1	0	快速PWM	ICR1	BOTTOM	TOP
15	1	1	1	1	快速PWM	OCR1A	BOTTOM	TOP



OCR1x/TOP更新、TOV1中断标志和OC1A或ICF1中断标志置位
(在TOP上中断)

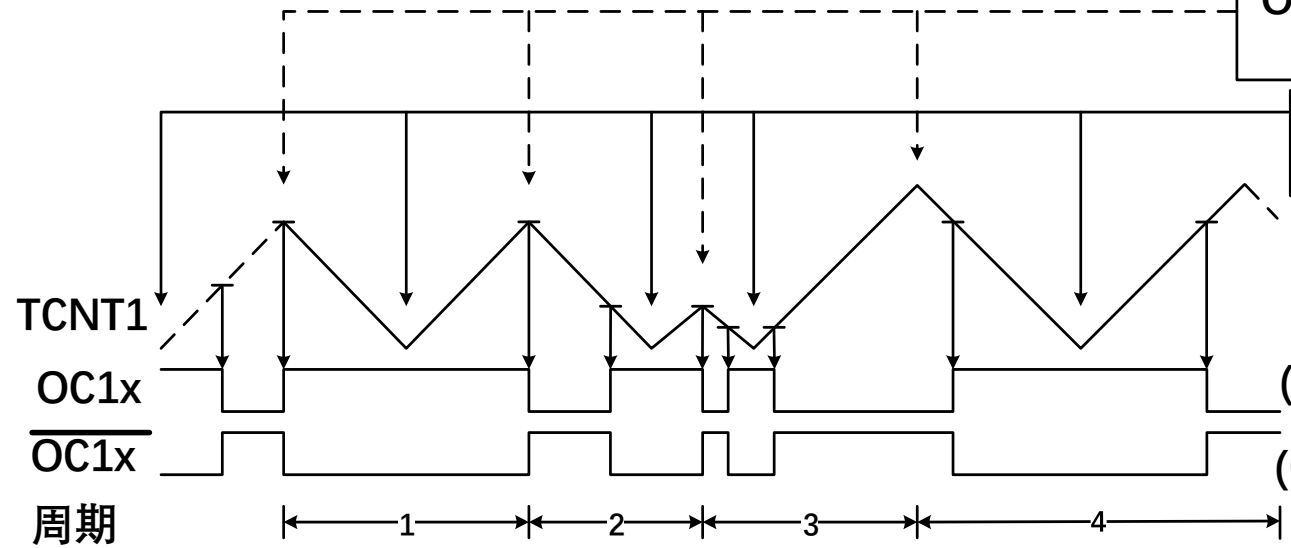
$$f_{OCnxPWM} = \frac{f_{clk\_I/O}}{N\times(1+TOP)}$$

(COM1x1:0 = 2) 比较匹配时OC1A/OC1B清零，在底置位OC1A/OC1B（非反向模式）

(COM1x1:0 = 3) 比较匹配时OC1A/OC1B置位，在底清零OC1A/OC1B（反向模式）

TC1 工作模式：相位校正PWM

模式	WGM[13:10]				操作模式	TOP	OCR1x更新点	TOV1置位点
1	0	0	0	1	PWM,相位校正,8位	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM,相位校正,9位	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM,相位校正,10位	0x03FF	TOP	BOTTOM
10	1	0	1	0	PWM,相位校正	ICR1	TOP	BOTTOM
11	1	0	1	1	PWM,相位校正	OCR1A	TOP	BOTTOM



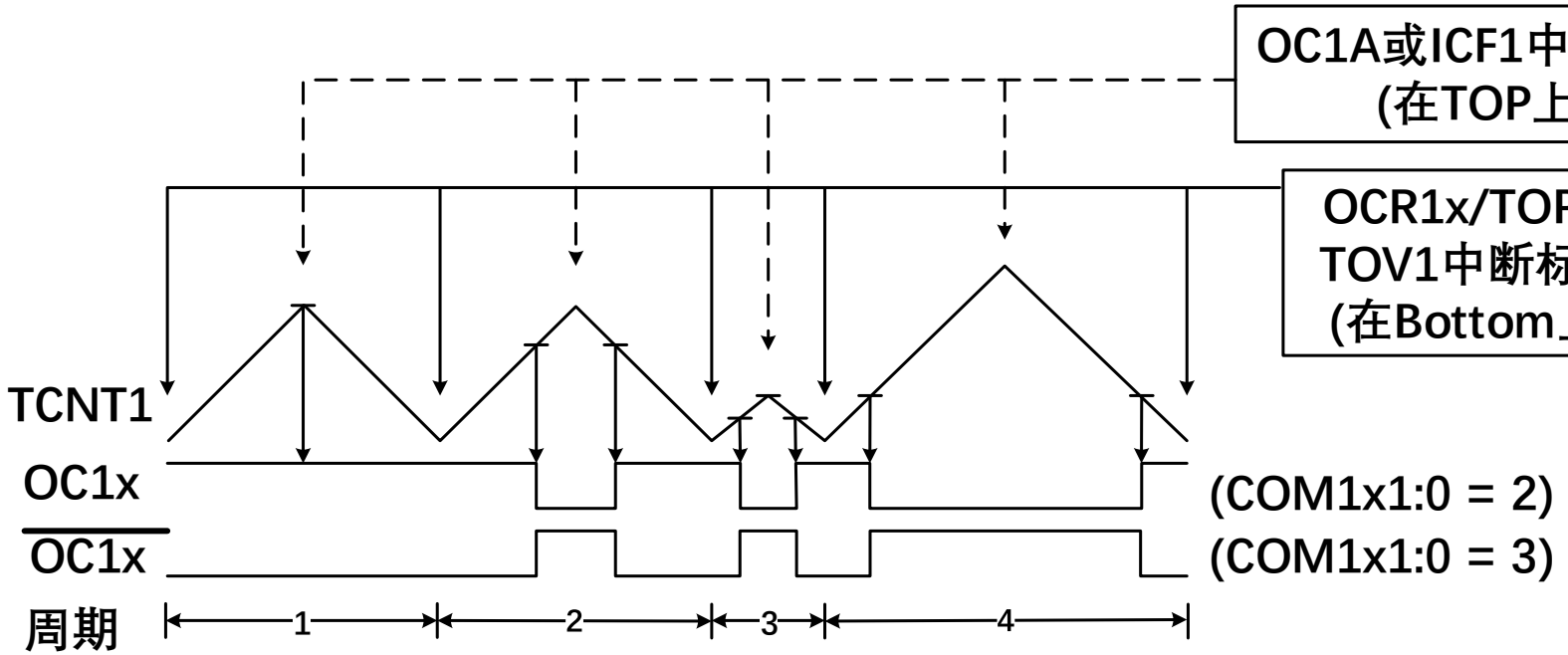
$$f_{OCnxPCPWM} = \frac{f_{clk\_I/O}}{2 \times N \times TOP}$$

(COM1x1:0 = 2) 加计数匹配时OC1A/OC1B清零，减计数匹配时OC1A/OC1B置位。

(COM1x1:0 = 3) 加计数匹配时OC1A/OC1B置位，减计数匹配时OC1A/OC1B清零。

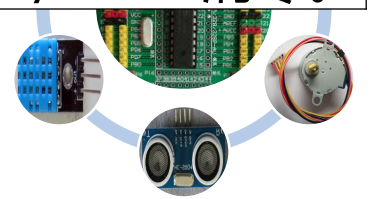
TC1 工作模式：相位和频率校正PWM

模式	WGM[13:10]				操作模式	TOP	OCR1x更新点	TOV1置位点
8	1	0	0	0	PWM,相位与频率校正	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	PWM,相位与频率校正	OCR1A	BOTTOM	BOTTOM



$$f_{OCnxPCPWM} = \frac{f_{clk\_I/O}}{2 \times N \times TOP}$$

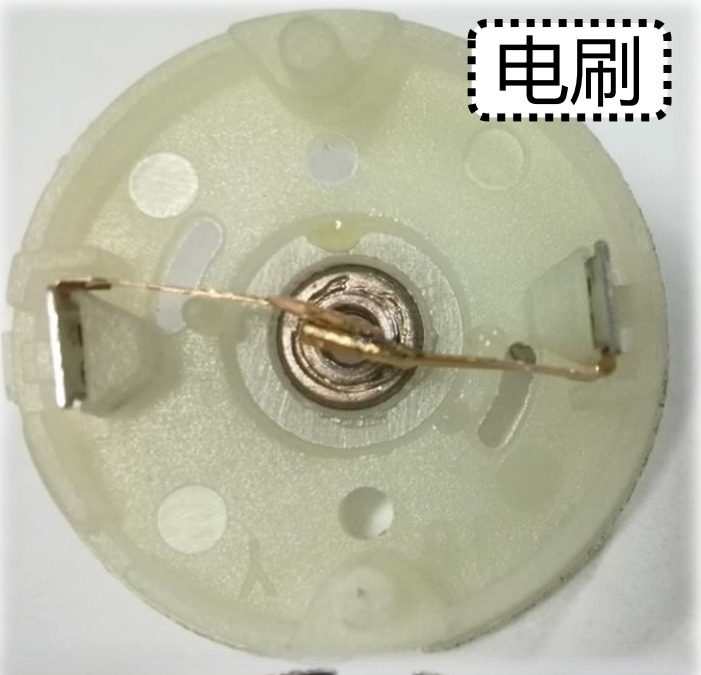
加计数匹配时OC1A/OC1B清零，减计数匹配时OC1A/OC1B置位。
加计数匹配时OC1A/OC1B置位，减计数匹配时OC1A/OC1B清零。

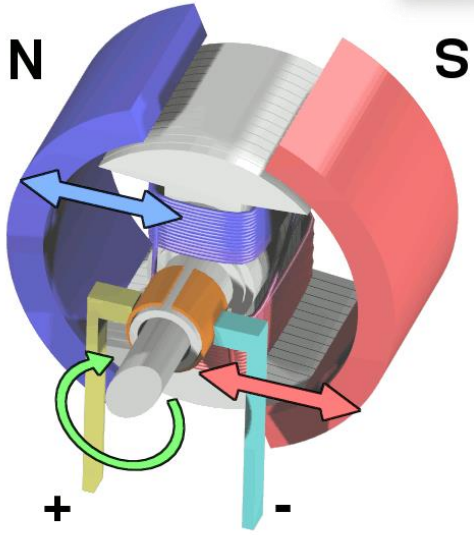


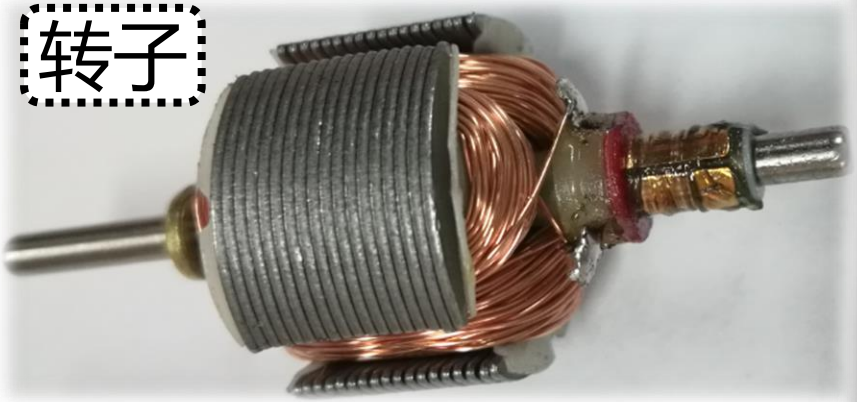
微型有刷直流电机：R260

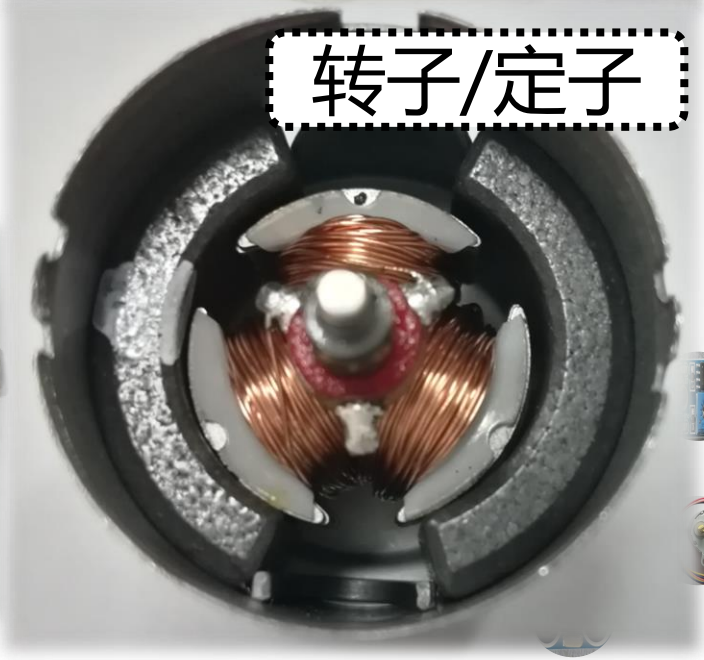
R-圆形
2-转子直径
0-3槽

4.5V：
16000RPM







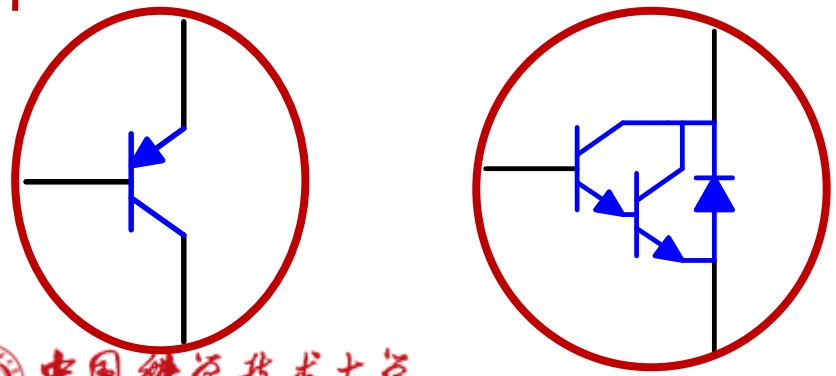


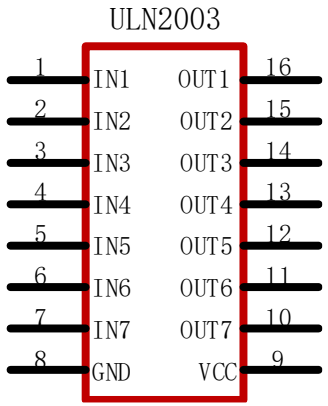
微型有刷直流电机：R260的驱动

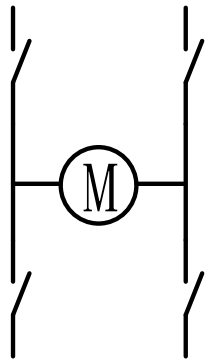
5V工作电压 ＋ 线圈电阻：很小 ＝ 转速>电流最大260mA

MCU最大IO电流40mA(20mA) ✖ 无法驱动

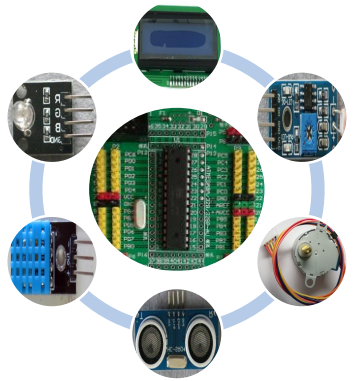
功率放大(电流放大) → 直流电机



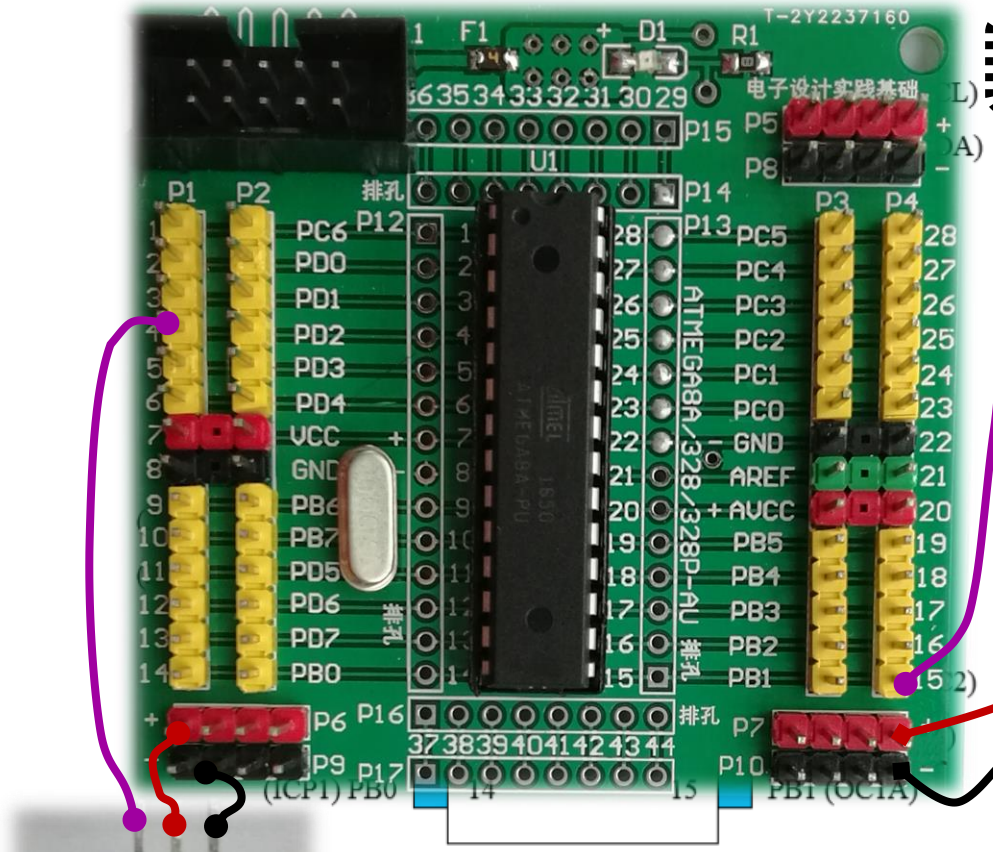




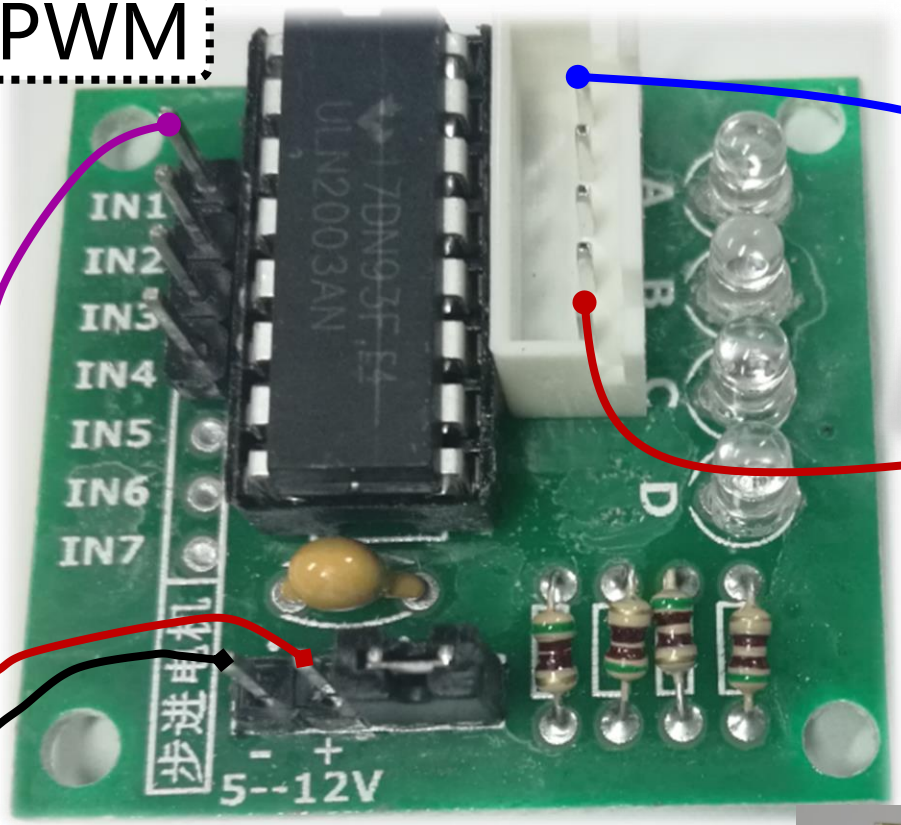
……

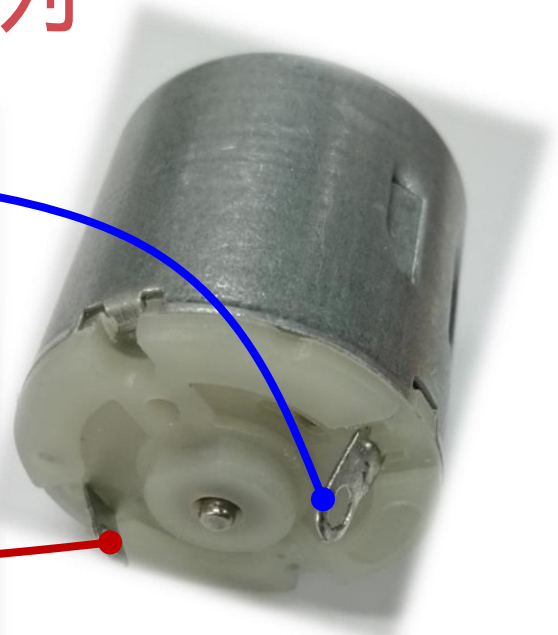


ATMEGA8A控制控制微型直流电机示例



PWM





实验内容4

调速

会拉低供电电压1V

技术大学
Technology of China

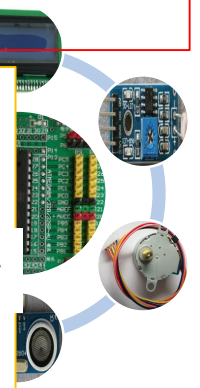
ATMEGA8A控制控制微型直流电机示例编程

```
int main(void)
{  unsigned char uc_tmp;   LCD_Init();
   DDRB |=(1<<DDRB1);//PB1(OC1A)输出
   ICR1 = 200;/*TOP值*/ OCR1A = 10;//A路比较值
   TCCR1A |= (1<<COM1A1)|(1<<COM1A0);//管脚切换方式
   TCCR1B |= (1<<WGM13)|(1<<CS11);//模式为8，时钟8分频
   TIMSK =(1<<TOIE1);//开启TC1溢出中断
   DDRD &= ~(1<<DDRD2);//PD2(int0)为输入<-tp223
   MCUCR |=((1<<ISC01)|(1<<ISC00));//int0 上升沿触发中断
   GICR |= (1<<INT0);/*允许INT0中断*/ sei(); //开全局中断
    while (1) {  uc_tmp = counter;
         LCD_Write_Char(0,6,uc_tmp/10+0x30);
         LCD_Write_NewChar(uc_tmp%10+0x30);
         _delay_ms(2);    }
}
```

```
#include <avr/io.h>
#include "twi_lcd.h"
#include <avr/interrupt.h>
unsigned char counter=1; //
```

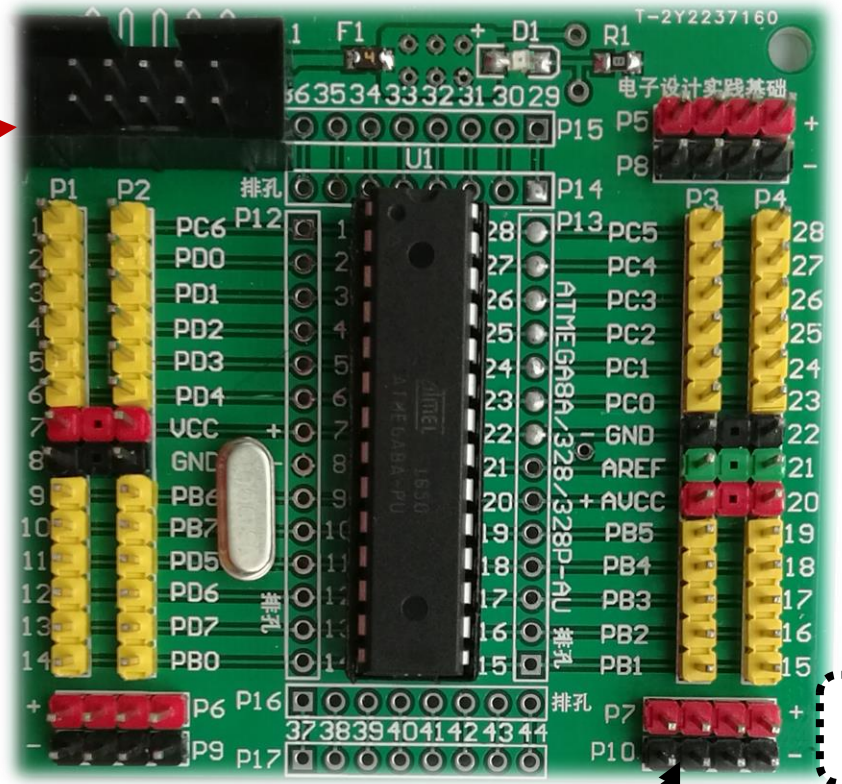
```
ISR(INT0_vect)
{if(counter <20)
  counter++;
else
  counter = 1;
}
```

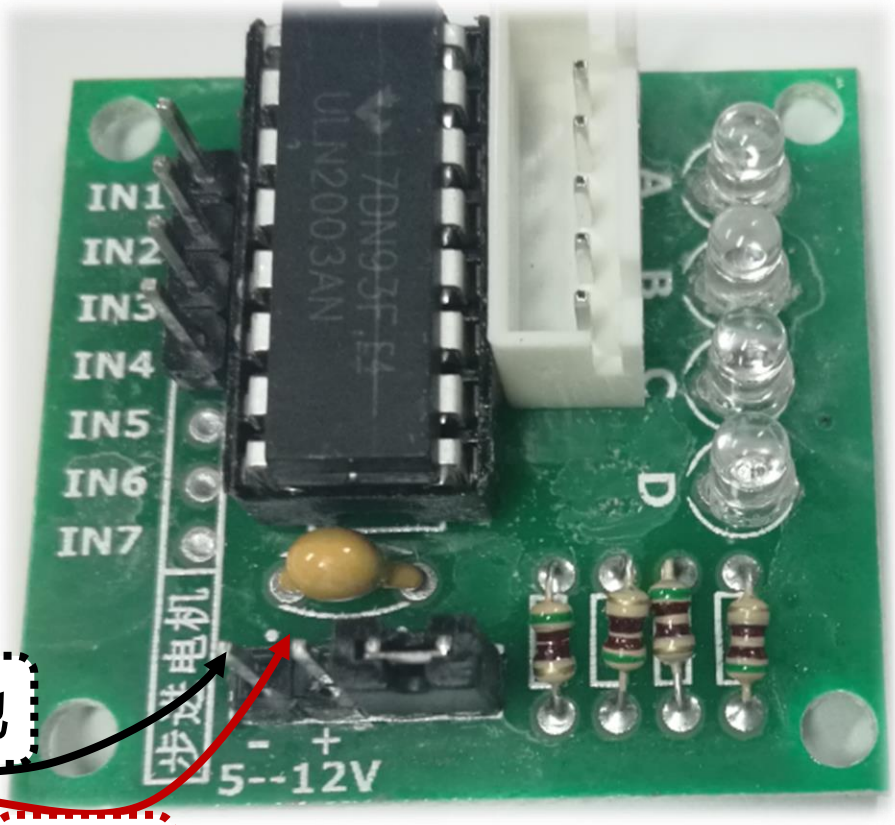
```
ISR(TIMER1_OVF_vect)
{
  OCR1A = counter * 6;
}
```



ULN2003单独供电的参考连接：+5V充电器

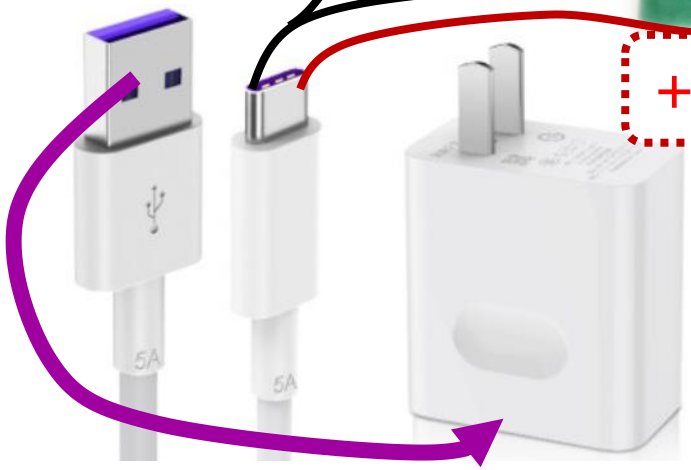
电脑USB

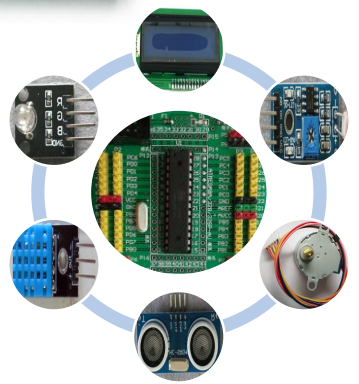




共地

+5V

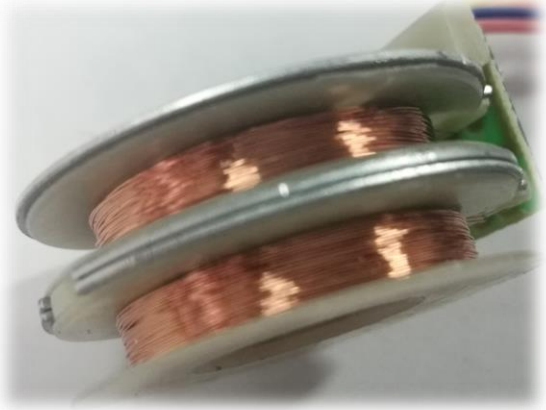




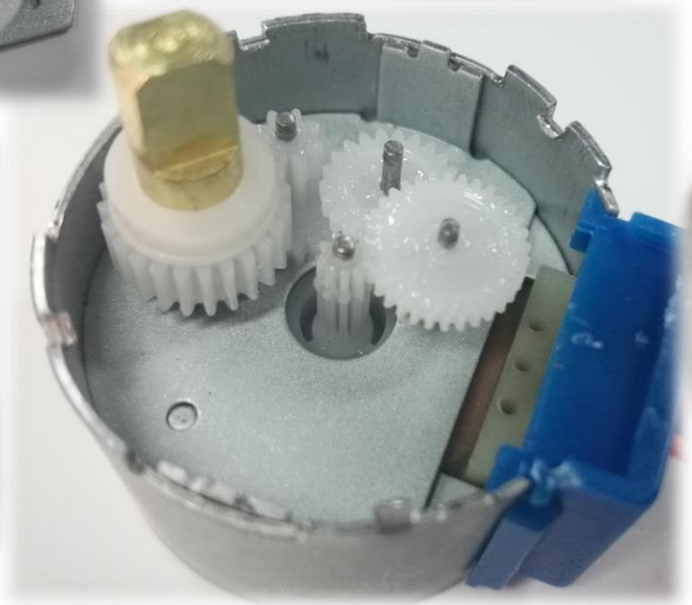
步进电机：28BYJ-48

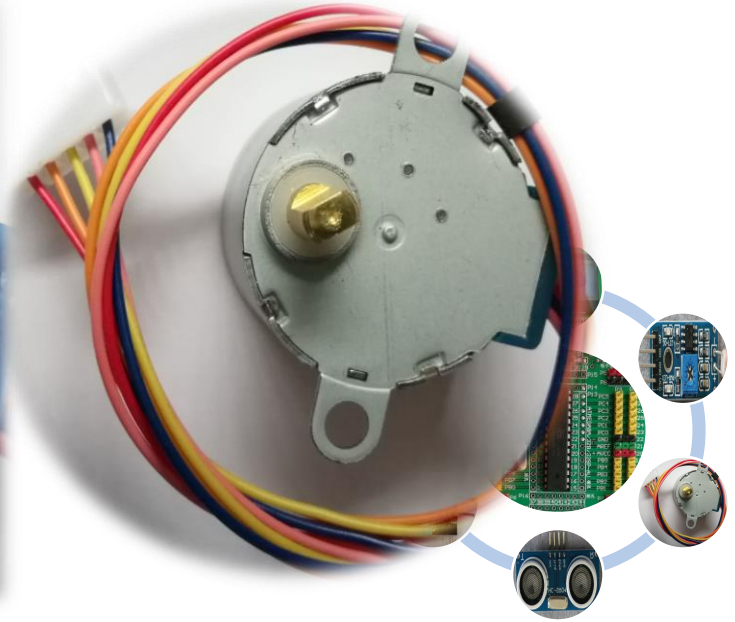
- 28-mm，外径；B-步进；Y-永磁

J-减速；48-四相八拍



减速比1:64





步进电机：28BYJ-48

四相：

单四拍：A<>B<>C<>D<>A

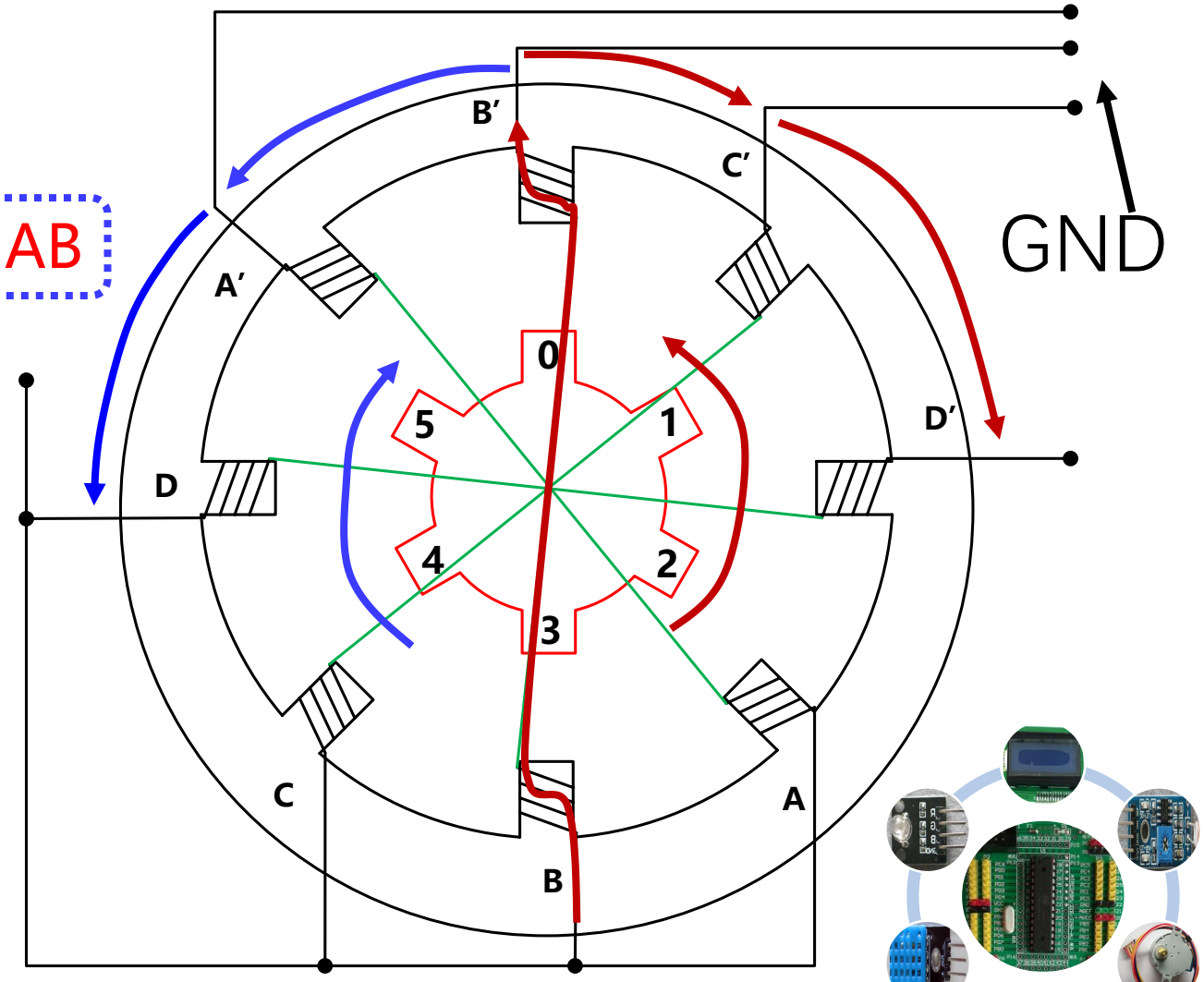
双四拍：AB<>BC<>CD<>DA<>AB

步距角：11.25°

八拍：

A<>AB<>B<>BC<>C<>CD<>D<>DA<>A

步距角：5.625°

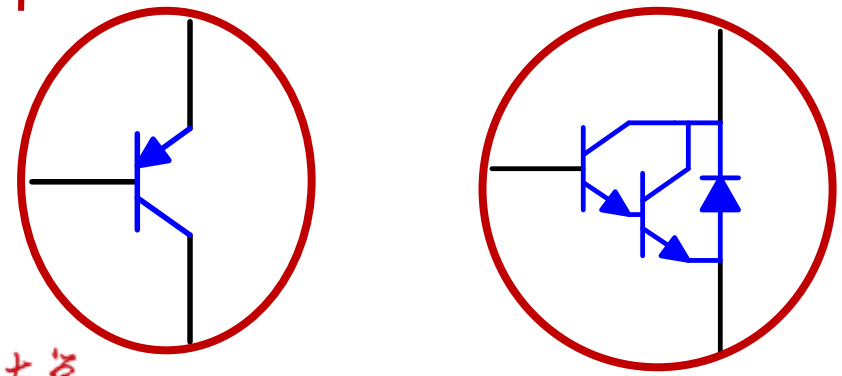


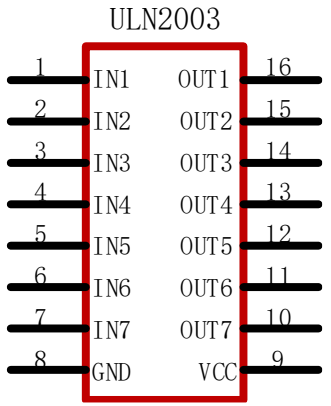
步进电机：28BYJ-48的驱动

5V工作电压 ✚ 线圈电阻：20~40Ω ＝ 相电流250~125mA

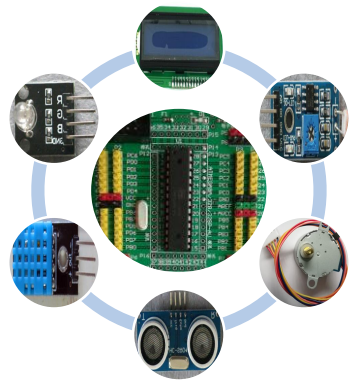
MCU最大IO电流40mA(20mA) ✖ 无法驱动28BYJ

功率放大(电流放大) → 28BYJ

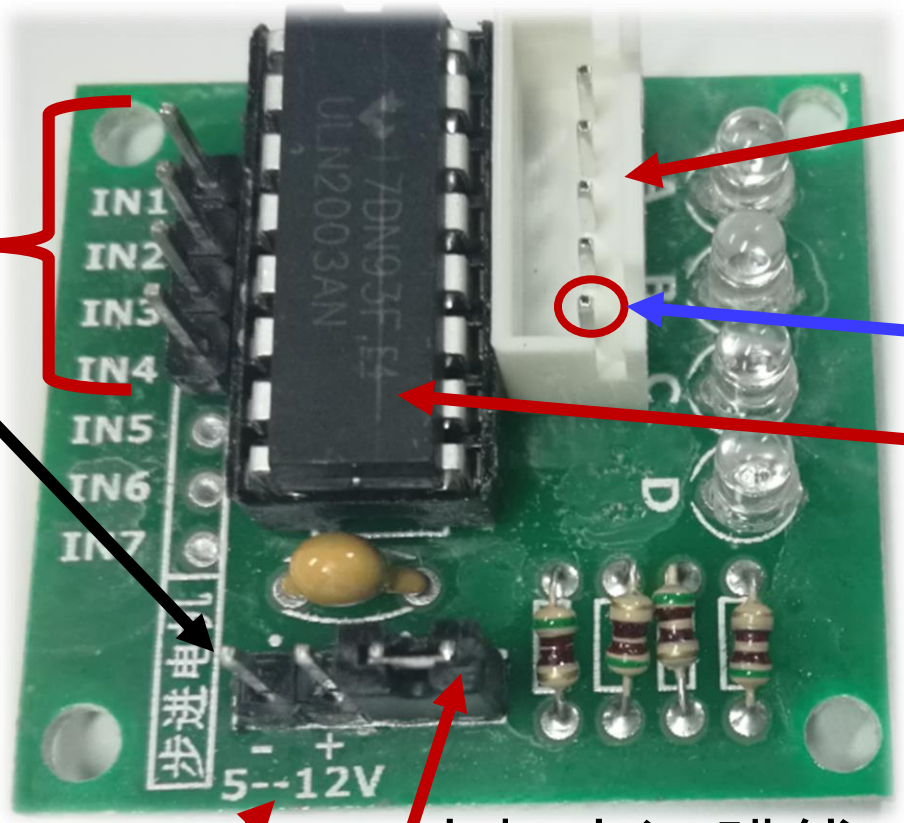




……



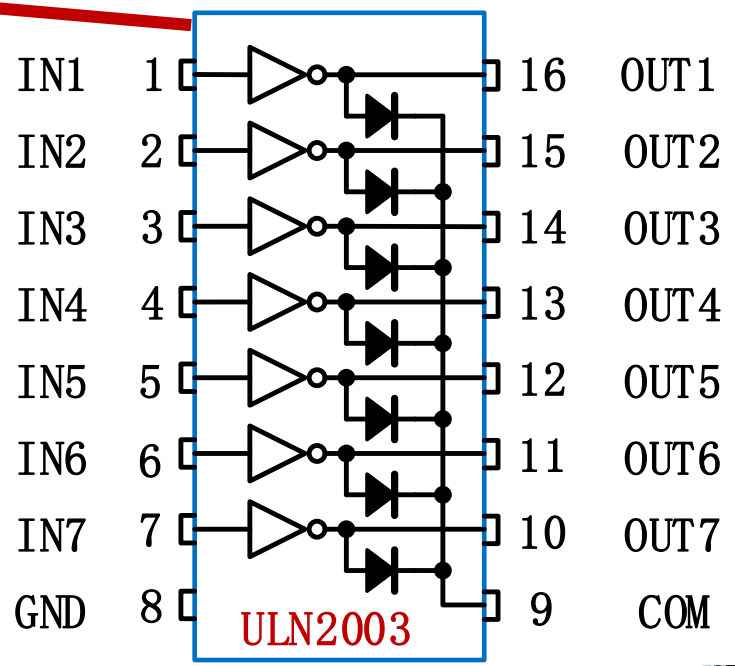
步进电机：28BYJ-48的驱动板



MCU IO

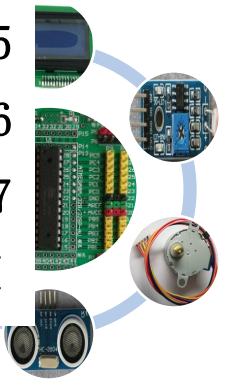
驱动板单独供电时，需与MCU共地(GND)

COM

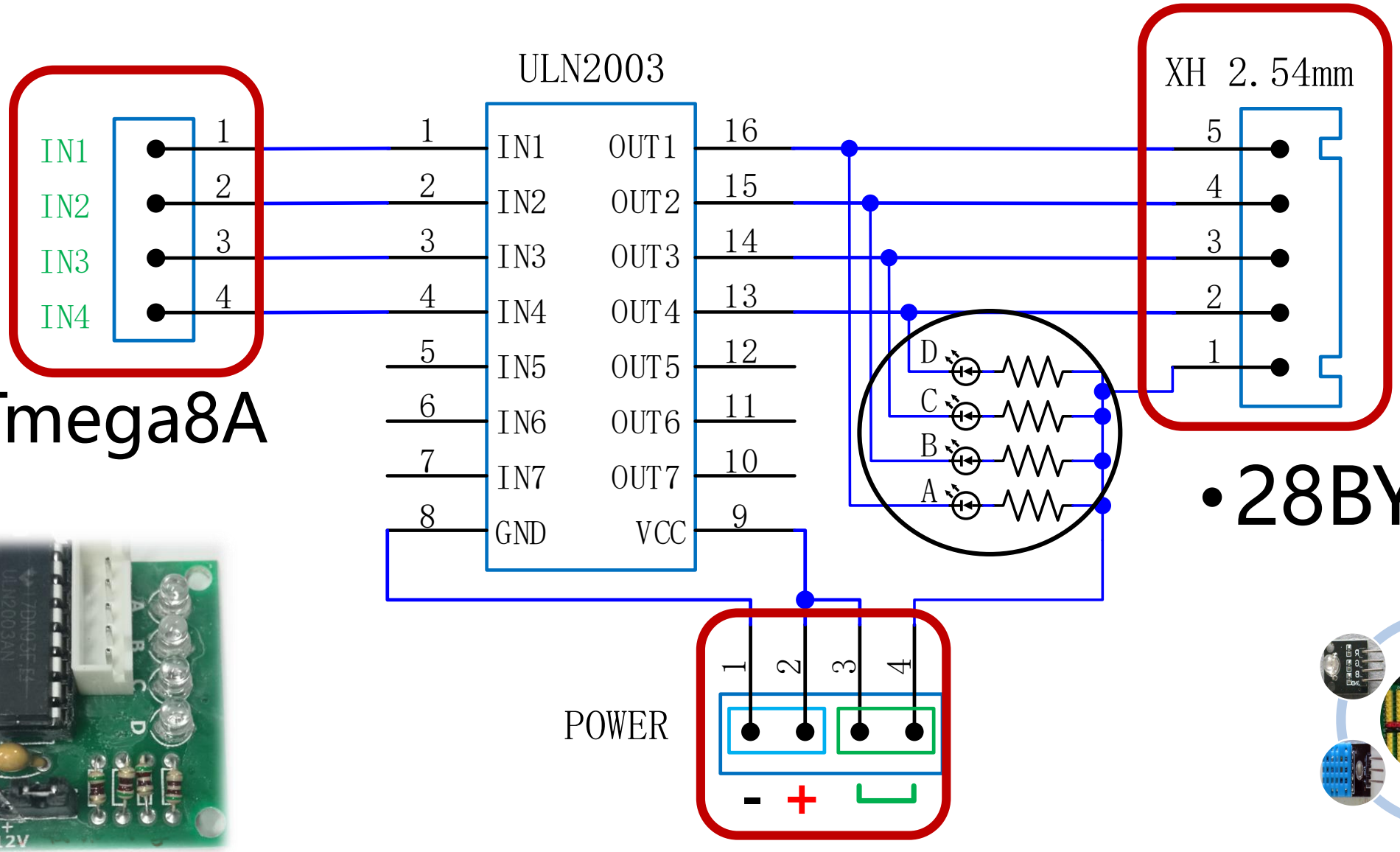


电机电源跳线

可单独供电(+:DC5~12V)、-:GND

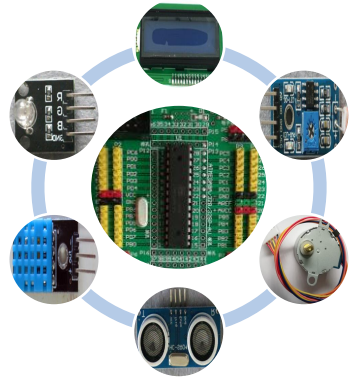


步进电机：28BYJ-48的驱动板



- ATmega8A

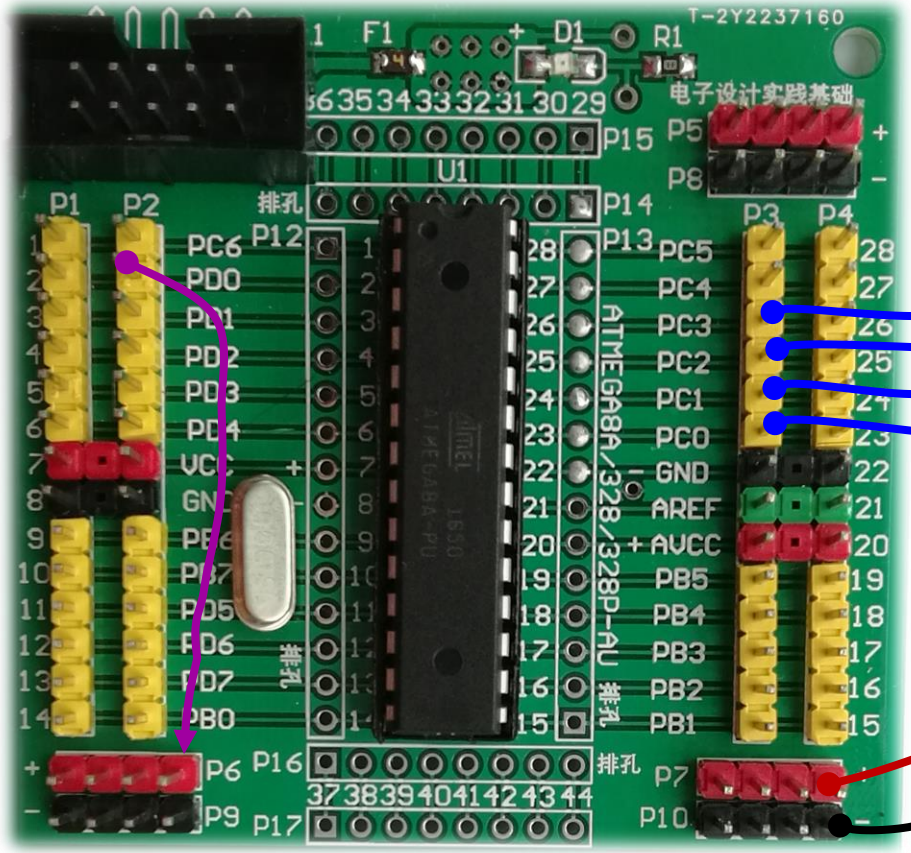
- 28BYJ-48

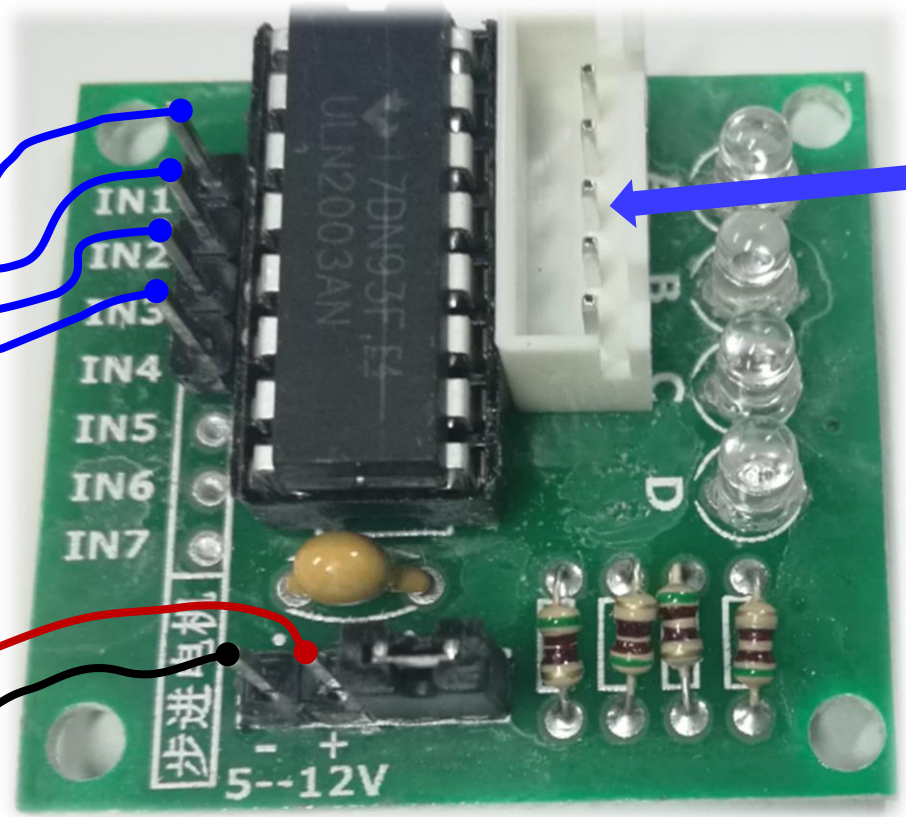


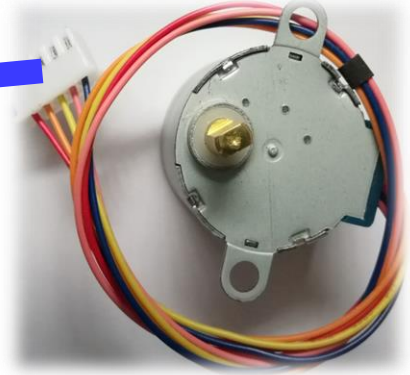
ATMEGA8A控制步进电机示例

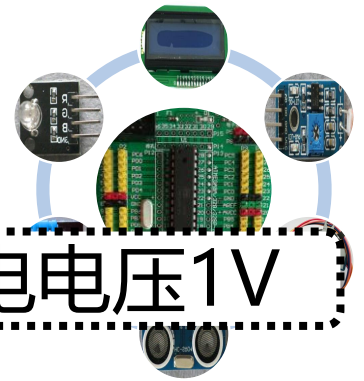
PD0：跳线，接到+或-，换转动方向

实验内容5





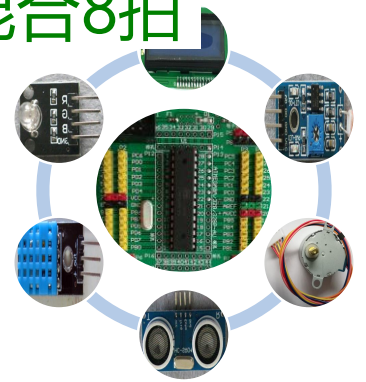




用实验板供电时，会拉低供电电压1V

28BYJ-48的ATMEGA8A控制编程：头文件（1）

```
#ifndef M_28BYJ48_H
#define M_28BYJ48_H_ //头文件 "m_28byj48.h"
#ifndef F_CPU
#define F_CPU 1000000UL/*<util/delay.h>需要定义F_CPU参数*/
#endif
#include <util/delay.h> //包含延时函数_delay_ms()和_delay_us()等
#include <avr/io.h>
/* 3种不同驱动方式下28byj48步进电机的控制信号，脉冲数据
* 低4位中bit3控制28byj48的蓝色线(D),bit2-粉(C),bit1-黄(B),bit0-橙(A)*/
const unsigned char
stepper_ph[3][8]={{0x01,0x03,0x02,0x06,0x04,0x0c,0x08,0x09},//混合8拍
{0x03,0x06,0x0C,0x09},//双4拍
{0x01,0x02,0x04,0x08}};//单4拍
unsigned char stepper_index = 0;//记录并控制步进电机的步伐
```



28BYJ-48的ATMEGA8A控制编程：头文件（2）

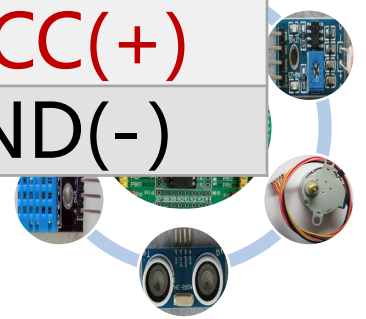
```
/* 28BYJ-48步进电机的控制函数，其参数的作用为：
* phase:选择步进电机的驱动方式，0：8拍，1：双4拍，2：单4拍
* dir:步进电机的转动方向，0：顺时针，1：逆时针
* step:控制步进电机转动多少步（多少个脉冲），0~65535*/
void run  stepper(unsigned char phase,unsigned char dir,unsigned int step)
{    unsigned char ph = phase ? 0x03:0x07;//驱动方式，确定脉冲的选择增量
     unsigned char inc = dir ?0x01:ph; /*逆时针增量为1，顺时针  4拍为3,8拍为7*/
     unsigned int i;//定义循环变量i
     DDRC |=(1<<DDRC3)|(1<<DDRC2)|(1<<DDRC1)|(1<<DDRC0);//PC输出
     for(i=0;i<step;i++)//输出step个脉冲步
     {  stepper  index += inc;//通过增量调整，不断地切换输出脉冲
          stepper  index &= ph; //去掉递增后的无效高位值
          PORTC &=0xf0; //选择相应的控制脉冲通过端口C进行输出
          PORTC |= stepper  ph[phase][stepper  index];
          _delay_us(900);//8拍 _delay_us(1600);//双4拍//_delay_us(2000);//单四拍
     } }
#endif /* M_28BYJ48_H_ */
```

28BYJ-48的ATMEGA8A控制编程：主程序

```
#include <avr/io.h>
#include "m_28byj48.h"
int main(void)
{ DDRD &=~(1<<DDRD0);//输入：通过PD0连接到电源正极或负极进行
正反转选择
  PORTD |=(1<<PORTD0); //内部上拉
  while (1)
  { if(PIND & (1<<PIND0))
      run_stepper(0,1,4096);
      //8拍驱动方式，顺时针，转动一周
     else
      run_stepper(0,0,4096);
      //8拍驱动方式，逆时针，转动一周
     }
}
```

MCU与28BYJ-48驱动板的连接

电机驱动板	MCU
IN1	PC3
IN2	PC2
IN3	PC1
IN4	PC0
+	VCC(+)
-	GND(-)



本周实验内容

1，用TC0的中断功能实现LED灯的闪烁

2，仅用TC1实现LED灯的闪烁，过程不用CPU

3，用TC1的1路比较输出控制1个LED呼吸灯，另1路的控制1路LED灯闪烁

4，用PWM控制直流电机的旋转，能调速

5，用MCU控制步进电机的旋转，能调方向

前三个完成后一起验收，后面的每个完成后验收