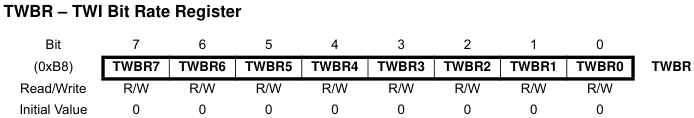
TWI相关的寄存器：

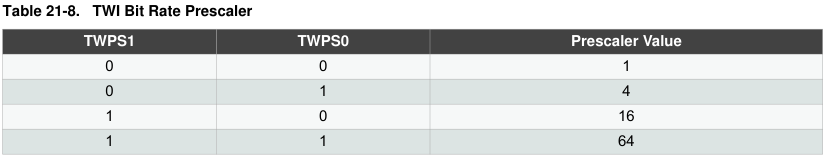
1，TWI比特率寄存器TWBR



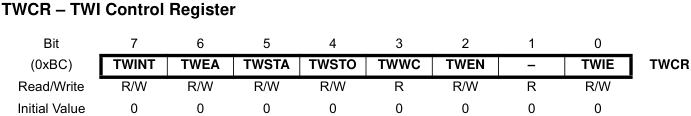
TWBR为比特率发生器分频因子，TWI工作于主机模式时，比特率发生器控制时钟信号SCL的周期，由TWBR寄存器和TWSR寄存器[bit1:bit0]共同决定。TWI工作在主机模式时，TWBR寄存器的值应该不小于10。SCL频率计算公式：



上式中，PrescalerValue即TWSR寄存器[bit1:bit0]的值：



2，TWI控制寄存器TWCR



TWCR寄存器用来控制TWI的操作，该寄存器用来使能TWI、通过施加START到总线上来启动主机访问、产生接收器应答、产生STOP状态、在写入数据到TWDR寄存器时控制总线暂停等。

TWINT(TWI Interrupt Flag)：TWI中断标志。当TWI完成当前工作，希望应用程序介入时，应使TWINT置位。若SREG寄存器的I标志位及TWCR寄存器的TWIE位也置位，则MCU执行TWI中断服务程序。

TWEA(TWI Enable Acknowledge Bit)：使能TWI应答。TWEA位控制应答脉冲的产生，若TWEA置位，则出现如下条件时接口发出ACK脉冲：

①器件的从机地址与主机发出的地址相符合；

②TWAR寄存器的TWGCE位在置位情况下，接收到广播呼叫；

③在主机/从机接收模式下接收到一个字节的数据；

④将TWEA位清零可以使器件暂时脱离总线，置位后器件重新恢复总线识别。

TWSTA(TWI START Condition Bit)：TWI START信号控制位。当MCU希望自己成为总线上的主机时需要置位TWSTA。这样，TWI硬件电路检测总线是否可用，若总线空闲，接口就在总线上产生START信号；若总线忙，接口就一直等待，直到检测到一个STOP信号为止。然后接口产生START信号以声明希望自己成为主机。发送START信号后，必须用软件清零TWSTA位。

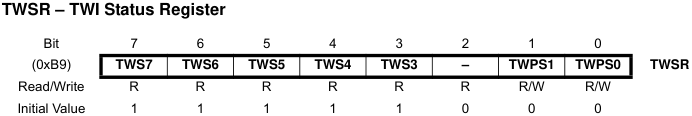
TWSTO(TWI STOP Condition Bit)：TWI STOP信号控制位。在主机模式下，如果置位TWSTO，则TWI接口将在总线上产生STOP信号，然后TWSTO位自动清零；在从机模式下，置位TWSTO可以使接口从错误状态恢复到未被寻找的状态。

TWWC(TWI Write Collision Flag)：TWI写碰撞标志位。当TWINT位为低时，试图写TWI数据寄存器TWDR会使TWWC置位；当TWINT位为高时，写TWDR寄存器会清零该位。

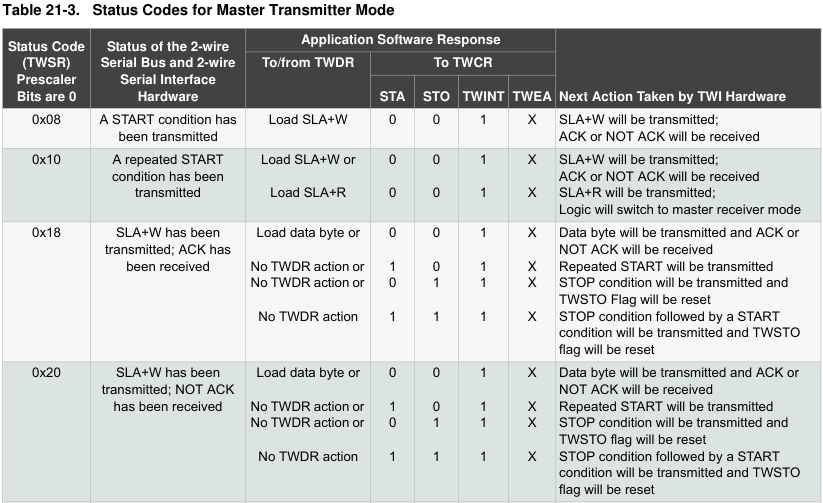
TWEN(TWI Enable Bit)：TWI使能位。TWEN位用于使能TWI操作和激活TWI接口。当TWEN置位时，MCU将相应的IO引脚切换到SCL与SDA引脚。当TWEN清零时，TWI接口模块被关闭，所有TWI传输被终止。

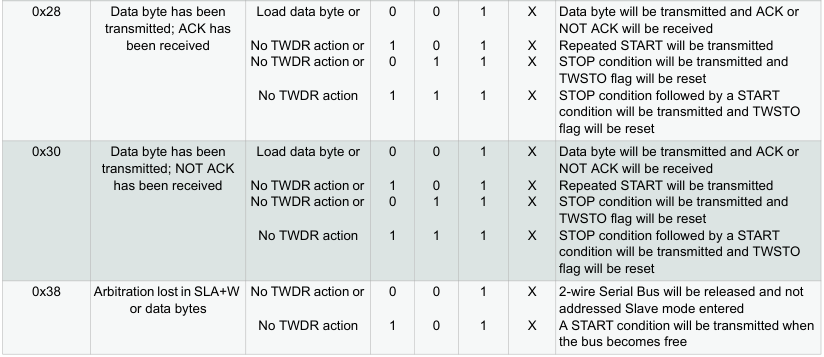
TWIE(TWI Interrupt Enable)：TWI中断使能位。当SREG寄存器的I位及TWIE置位时，只要TWINT置位，则立即执行TWI中断服务函数。

3，TWI状态寄存器TWSR

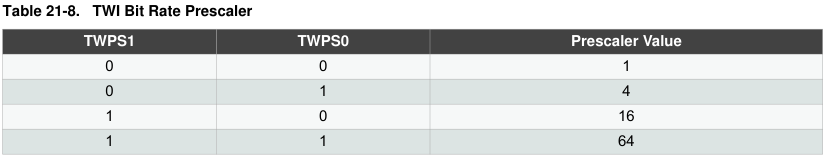


[bit7:bit3]：TWI状态位，这5位用来反映TWI总线的状态，只读。例如主机发送模式的状态码：

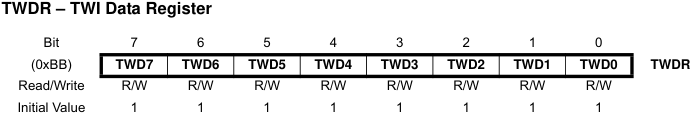




[bit1:bit0]：TWI预分频位，这2位用来设置TWI比特率，可读写。

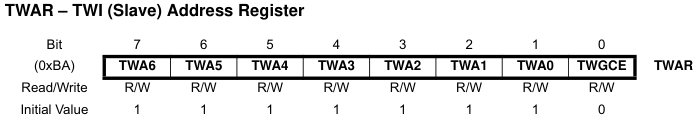


4，TWI数据寄存器TWDR



在发送模式下，TWDR寄存器包含了要发送的字节；在接收模式下，TWDR寄存器包含了接收到的数据。在TWINT置位情况下，可以对该寄存器写操作。

5，TWI地址寄存器TWAR



[bit7:bit1]：在主机模式下，不需要高7位；在从机模式下，高7位用于存放从机地址，TWI将根据这个地址进行响应。

TWGCE(TWI General Call Recognition Enable Bit)：无论主机模式还是从机模式，该位用于配置TWI是读还是写。置位时，读总线；清零时，写总线。

u

根据atmega328p参考手册page183页的例程写TWI总线的驱动代码：

1. //TWI状态定义，MT主机传输，MR主机接收
2. #define START 0x08
3. #define RE\_START 0x10
4. #define MT\_SLA\_ACK 0x18
5. #define MT\_SLA\_NACK 0x20
6. #define MT\_DATA\_ACK 0x28
7. #define MT\_DATA\_NACK 0x30
8. #define MR\_SLA\_ACK 0x40
9. #define MR\_SLA\_NACK 0x48
10. #define MR\_DATA\_ACK 0x50
11. #define MR\_DATA\_NACK 0x58
13. #define RD\_24C02\_ADDR 0xa1 //前4位器件固定，后三位看连线，最后一位1为读，0为写
14. #define WR\_24C02\_ADDR 0xa0
16. //常用TWI操作
17. #define start() (set\_bits(TWCR, \_BV(TWINT) | \_BV(TWSTA) | \_BV(TWEN))) //发送START信号
18. #define stop() (set\_bits(TWCR, \_BV(TWINT) | \_BV(TWSTO) | \_BV(TWEN))) //发送STOP信号
19. #define wait() (loop\_until\_bit\_is\_set(TWCR, TWINT)) //等待中断发生
20. #define test\_ack (TWSR & 0xf8) //观察返回状态
21. #define set\_ack (set\_bits(TWCR, \_BV(TWEA))) //做出ACK应答
22. #define set\_nack (clear\_bits(TWCR, \_BV(TWEA))) //做出NACK应答
23. #define twi() (set\_bits(TWCR, \_BV(TWINT) | \_BV(TWEN))) //使能TWI
24. #define write\_8bits(x) {TWDR = (x); set\_bits(TWCR, \_BV(TWINT) | \_BV(TWEN));} //写8位数据