I2C Registers in AVR

There are five registers in AVR

- TWI Bit rate register (TWBR)
- TWI Status register (TWSR)
- TWI Control register (TWCR)
- TWI Data register (TWDR)
- TWI Address register (TWAR)

TWI Bit rate register (TWBR):

7	6	5	4	3	2	1	0
TWBR7	TWBR6	TWBR5	TWBR4	TWBR3	TWBR2	TWBR2	TWBR0

TWI Status register (TWSR):

7	6	5	4	3	2	1	0

TWI control register (TWCR):

7	6	5	4	3	2	1	0
TWINT	TWEA	TWSTA	TSTO	TWWC	TWEN	-	TWIE

- TWINT :
    - Stands for TWI Intruppt
    - Bit is set by hardware when the TWI module has finished its current job
    - If TWI and general interrupts are enabled , changing TWINT to
        - One will cause the MCU to jump to TWI interrupt vector
        - Zero will start the operation of TWI
        - TWINT must be cleared by software

- TWEA :
    - Stands for TWI enable acknowledgement
    - Making this high will enable the generation of ACK when needed in slave or receiver mode

- TWSTA
    - Stands for TW start condition bit
    - Making this high generates start condition if bus is free:

    - If bus is not free, TWi module waits till bus gets free and then generates a start condition.
    -

- TSTO :
    - Stands for TW stop condition bit
    - Making this high generates stop condition :
    - This bit is cleared by hardware , when stop condition is transmitted

- TWWC :
    - Stands for TW write collision flag
    - This bit gets set when we try to access data register when TWINT is low
    - This bit is cleared by writing to TWDR (data) register when TWINT is high

- TWEN :
    - Stands for TW enable
    - Making this bit high enables the TWI module

- TWIE :
    - Stands for TW interrupt enable
    - Making this bit high enables the TWI interrupt if the general interrupt is enabled.

TWI data register (TWCR):

7	6	5	4	3	2	1	0

- Can be accessed only when TWEI is set to one
- In receiving mode , this register contains last received byte
- In transmit mod , next byte shall be written in this register.

TWI address register (TWCR):

7	6	5	4	3	2	1	0

- TWAR consists of 7 bit slave address to which TWI will respond , when working as slave
- **Eight th bit is call recognition bit** , it controls recognition of general call address
    - If this bit is set to one , receiving of general call address will cause an interrupt request.

MASTER MODE:

Source code: write one byte (0xf0) to a slave having address 1101000 :

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

Void i2c_write(unint8_t data)
{
        TWDR = data;
        TWCR = (1<<TWINT) | (1<< TWEN);
        while( ( TWCR & (1<<TWINT)) == 0);
}

Void i2c_start (void)
{
        TWCR =( (1<<TWINT) | (1<< TWEN) | (1<< TWSTA));
        while( ( TWCR & (1<<TWINT)) == 0);
}

Void i2c_stop (void)
{
        TWCR =( (1<<TWINT) | (1<< TWEN) | (1<< TWST0));

}
Void i2c_init (void)
{
        TWSR = 0x00;
        TWBR = 0x47;
        TWCR = 0x04;
}

Int main( void)
{
i2c_init();
i2c_start();
i2c_write(0b11010000);  // seven bit address + write bit “0”
i2c_write(0xfo);
i2c_stop();
return 0;
}
```

Source code: read a byte in master

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

```
Void i2c_write(unint8_t data)
{
        TWDR = data;
        TWCR = (1<<TWINT) | (1<< TWEN);
        while( ( TWCR & (1<<TWINT)) == 0);
}

Void i2c_start (void)
{
        TWCR =( (1<<TWINT) | (1<< TWEN) | (1<< TWSTA));
        while( ( TWCR & (1<<TWINT)) == 0);
}

Void i2c_stop (void)
{
        TWCR =( (1<<TWINT) | (1<< TWEN) | (1<< TWST0));

}
Void i2c_init (void)
{
        TWSR = 0x00;
        TWBR = 0x47;
        TWCR = 0x04;
}
Uint8_t i2c_read(uint8_t is_last_byte)
{
        If (is_last_byte == 0)
                TWCR =( (1<<TWINT) | (1<< TWEN) | (1<< TWEA));
        else
                TWCR =( (1<<TWINT) | (1<< TWEN) );
        While ( (TWCR & (1<< TWINT ))==0);
        return TWDR;

}

Int main( void)
{
Uint8_t i = 0;
i2c_init();
i2c_start();
i2c_write(0b11010001);  // seven bit address + read bit “0”
i = i2c_read(1);
```

```
        i2c_stop();
        return 0;
        }
```

SLAVE MODE:

Source code : write a byte in slave mode

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

Void i2c_initSlave(uint8_t slave_address)
{
        TWCR = 0x04;
        TWAR = slave_address;
        TWCR =( (1<<TWINT) | (1<< TWEN) | (1<< TWEA));
}

Void i2c_send (uint8_t data)
{
        TWDR = data;
        TWCR = (1<<TWINT) | (1<< TWEN);
        while( ( TWCR & (1<<TWINT)) == 0);
}

Void i2c_listen ()
{
        while( ( TWCR & (1<<TWINT)) == 0);
}

Int main(void)
{
        i2c_initSlave(0x10);
        i2c_listen();
        i2c_send('A');
return 0;
}
```

Source code : write a read in slave mode

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

Void i2c_initSlave(uint8_t slave_address)
{
        TWCR = 0x04;
        TWAR = slave_address;
        TWCR =( (1<<TWINT) | (1<< TWEN) | (1<< TWEA));
}

Void i2c_receive (uint8_t isLastByte)
{
        If (is_last_byte == 0)
                        TWCR =( (1<<TWINT) | (1<< TWEN) | (1<< TWEA));
                else
                        TWCR =( (1<<TWINT) | (1<< TWEN) );
                While ( (TWCR & (1<< TWINT ))==0);
                return TWDR;

}

Void i2c_listen ()
{
        while( ( TWCR & (1<<TWINT)) == 0);
}

Int main(void)
{
Uint8_t i;
        i2c_initSlave(0x10);
        i2c_listen();
        i = i2c_receive(1);
return 0;
}
```