# **I2C AVR PROGRAMACIÓN**

Para programar la comunicación serial I2C AVR se recomienda mucho leer la hoja de datos del AVR utilizado, es lo mejor que se puede hacer, como referencia se verá para el [**ATMEGA88**](http://www.atmel.com/images/doc2545.pdf), para lo que se utilizan los siguientes registros: el registro**TWBR**, el registro **TWCR**, el registro **TWSR**, el registro **TWDR**, el registro **TWAR**, y el registro **TWAMR**.

## **El** **registro** **TWBR**

Cuando  el microcontrolador AVR es utilizado como maestro, en este registro se cargará el valor con el que se elige la velocidad de transmisión de los datos, esto es los baudios o la cantidad de bits por segundo que se enviarán en la comunicación I2C AVR.

El valor a cargar en el registro **TWBR** se calcula con la siguiente fórmula que se encuentra en la hoja de datos, que dependerán de la velocidad elegida, de la frecuencia del oscilador utilizado y de un prescaler para la frecuencia del oscilador, el prescaler puede ser 1, 4, 16 o 64, se elige mediante las combinaciones de los bits 1 y 0 del registro TWSR.

**Velocidad=Fosc/ (16+2\*(TWBR)\*prescaler)**

Esta fórmula se puede reacomodar para obtener la siguiente fórmula:

**TWBR= ((Fosc/Velocidad)-16)/ (2\*prescaler)**

Si se quiere por ejemplo una velocidad d 100Khz, y el oscilador es de 8Mhz, con un prescaler de 1, si en la fórmula anterior se reemplaza y se despeja se tendrá:

TWBR= ((8000000/400000)-16)/ (2\*1)

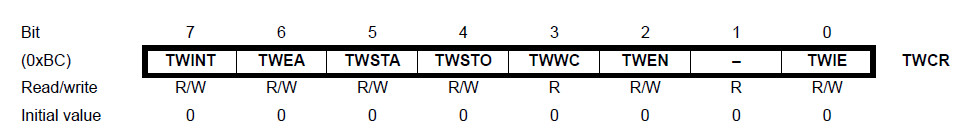
De donde:

TWBR=2

Por lo que para obtener un una velocidad de 400Khz se tendrá que cargar el registro TWBR con 2.

## **El** **registro** **TWCR**

Este es el registro de control del módulo TWI para la comunicación I2C AVR

[](http://microcontroladores-mrelberni.com/ym-romaim/uploads/2015/11/I2C-AVR-TWCR.jpg)

### El bit7 TWINT:

Este es el bit de reconocimiento de que ha ocurrido una interrupción en caso de que esta esté habilitada la interrupción del módulo TWI al poner a uno el bit0 TWIE, en el caso de no habilitar las interrupciones este será el bit más utilizado ya que mediante la verificación del estado de este bit se podrá conocer si se ha iniciado correctamente la comunicación I2C AVR, o también si se ha detenido la comunicación, o también si se han enviado o recibido los datos correctamente.

### El bit6 TWEA:

Al poner este bit a 1 se habilita el uso del bit de reconocimiento ACK

### El bit5 TWSTA:

Al poner este bit a 1 se da inicio a la comunicación I2C AVR

### El bit4 TWSTO:

Al poner este bit a 1 se para o termina la comunicación I2C AVR.

### El bit3 TWWC:

Este bit se pone a 1 automáticamente indicando un error cuando se quiere enviar o recibir un dato mientras no se dan las condiciones apropiadas.

### El bit2 TWEN:

Al poner este bit a 1 se habilita el uso del módulo TWI para la comunicación serial, mientras este bit esté a 0 no se podrá utilizar la comunicación I2C AVR.

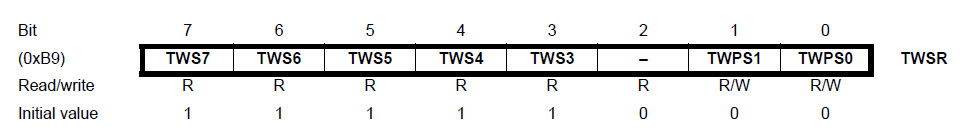
**El bit1**este  bit se pone a 0 ya que no se utiliza.

### El bit0 TWIE:

Habilita el uso de la interrupción del módulo TWI.

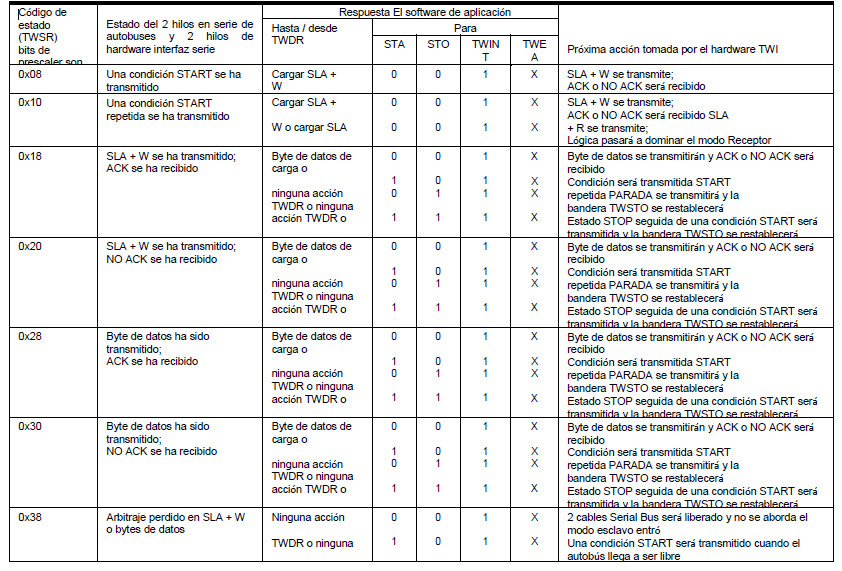
## **El** **registro** **TWSR**

Este registro se utiliza para averiguar en qué estado se encuentra el módulo TWI cuando se realiza la comunicación serial I2C AVR, en este registro también se encuentran los bits mediante los cuales se elige el prescaler para obtener la velocidad de la comunicación I2C AVR.

[](http://microcontroladores-mrelberni.com/ym-romaim/uploads/2015/11/I2C-AVR-TWSR.jpg)

### Bit7 a bit3

Estos 5 bits trabajan en forma automática generando diferentes combinaciones, dependiendo de las combinaciones permite conocer en qué estado se encuentra la comunicación I2C AVR, para ello en la hoja de datos del  AVR utilizado se tienen unas tablas con lo que indican esta combinaciones, la siguiente es por ejemplo para el caso cuando el ATMEGA88 se comporta  como maestro y está enviando un dato al esclavo:

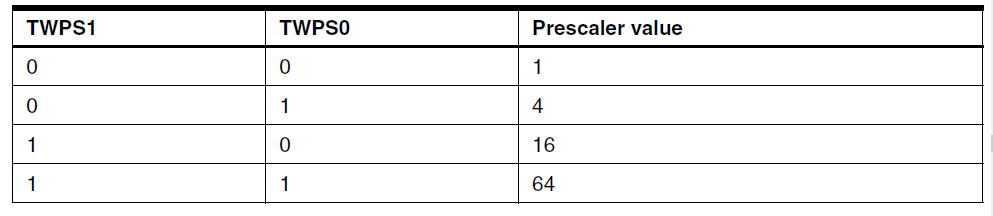
[](http://microcontroladores-mrelberni.com/ym-romaim/uploads/2015/11/I2C-AVR-estado-del-TWI.jpg)

Si las combinaciones de estos bits es por ejemplo 0x08 quiere decir que se ha dado el inicio de la comunicación I2C AVR, las combinaciones de estos bits son utilizadas dentro del programa para verificar que la comunicación  transcurre con normalidad.

**El bit2** no se utiliza se pone a 0.

### Los bits 1 y 0

Con las combinaciones de estos bits se elige el prescaler a utilizar para obtener la frecuencia de transmisión, de donde se obtiene el valor a cargar en el registro **TWBR,**las combinaciones son las siguientes:

[](http://microcontroladores-mrelberni.com/ym-romaim/uploads/2015/11/I2C-AVR-prescaler.jpg)

## **El** **registro** **TWDR**

En la comunicación I2C AVR en este registro se pondrá el dato que se quiera transmitir hacia el esclavo, y también donde se encontrará el dato recibido desde el esclavo, tiene esta doble función.

## **El** **registro** **TWAR**

En los bits del 7 al 1 de este registro se colocará la dirección del microcontrolador AVR cuando este es utilizado como esclavo, el bit0 es para una llamada  general, al poner el bit0 a 0 el microcontrolador responderá solo si el maestro envía su dirección, al poner este bit a 1 el AVR esclavo responde cuando el maestro envíe la dirección 0x00.

# **I2C AVR PROGRAMACIÓN en el ATMEL STUDIO**

Para la programación de la comunicación serial I2C AVR, cuando el AVR sea el maestro se utilizará el ATMEL STUDIO, en este caso se comentará como se puede proceder para utilizar el módulo TWI para la comunicación serial I2C AVR del [**ATMEGA88**](http://www.atmel.com/images/doc2545.pdf), lo aquí comentado es muy similar para otros microcontroladores AVR.

## Para iniciar el I2C AVR, siendo el AVR el maestro.

Los pines del ATMEGA88 para el manejo de la comunicación I2C AVR son el pin PC5 para la señal de reloj por lo que en este caso se le llama pin SCL, y el pin PC4 para la transmisión y recepción de datos, en este caso se le llama pin SDA, los pines serial clock SCL y serial data SDA no es necesario que se configuren como entradas digitales, lo que sí se puede aprovechar es utilizar sus resistencias internas PULL UP, lo que en el ATMEL STUDIO se puede hacer así:

**PORTC|= ((1<<PC4)| (1<<PC5));** //habilitación de resistencias pull up par SCL y SDA

### **Configuración del registro TWBR**

El valor a cargar en el registro **TWBR** se calcula con la siguiente fórmula que se encuentra en la hoja de datos, que dependerán de la velocidad elegida, de la frecuencia del oscilador utilizado y de un prescaler para la frecuencia del oscilador, el prescaler puede ser 1, 4, 16 o 64, se elige mediante las combinaciones de los bits 1 y 0 del registro TWSR.

**Velocidad=Fosc/ (16+2\*(TWBR)\*prescaler)**

Esta fórmula se puede reacomodar para obtener la siguiente fórmula:

**TWBR= ((Fosc/Velocidad)-16)/ (2\*prescaler)**

Se utilizará una velocidad de 400Khz, la frecuencia de trabajo del ATMEGA88 a 8 Mhz, se usará un prescaler de 1, reemplazando se obtendrá:

TWBR= ((8000000/400000)-16)/ (2\*1)

De donde:

TWBR=2

Por lo que para obtener un una velocidad de 400Khz se tendrá que cargar el registro TWBR con 2.

Luego en la inicialización de la comunicación serial I2C AVR el registro **TWBR** en el ATMEGA88 quedará así:

**TWBR=2;** //velocidad 400Khz, Fosc 8Mhz, prescaler de 1

### **Configuración del registro TWCR**

Para iniciar módulo TWI de este registro solo es necesario habilitar el bit TWEN, esto en el ATMEL STUDIO será así.

**TWCR|= (1<<TWEN);** //módulo TWI iniciado

### **Carga del registro TWSR**

De este registro solo serán necesarios configurar  los bits 1 y 0 para el prescaler utilizado, pero como es utiliza el prescaler de 1 las combinaciones de estos bits es 00, y como este registro se inicia con todos sus bits a 0, no será necesario hacer nada en este caso, pero si se utilizará otro prescaler sí que se tendrá que hacer en el inicio.

Entonces se puede iniciar el módulo TWI para la comunicación serial I2C AVR en el ATMEGA STUDIO de la siguiente manera:

1. ///////////////////////////////////////////////////////////////////////////
2. /////////////////////////////////////////////////////////////////////////////
3. //inicialización del módulo TWI I2C AVR en el ATMEL STUDIO////
4. //para el ATMEGA88 como maestro
5. //a 400Khz con un oscilador de 8Mhz
6. PORTC|=((1<<4)|(1<<5)); //activa resistencias pull up para SCL y SDA
7. TWBR=2; //velocidad 400Khz, Fosc 8Mhz, prescaler de 1
8. TWCR|=(1<<TWEN); //módulo TWI iniciado

La inicialización podría hacerse dentro de una función a la que se le puede dar el nombre que se quiera, en este caso se la llamará **i2c\_iniciar()** lo que en el ATMEL STUDIO será así:

1. ///////////////////////////////////////////////////////////////////////////
2. /////////////////////////////////////////////////////////////////////////////
3. //inicialización del módulo TWI I2C AVR en el ATMEL STUDIO en una función////
4. //para el ATMEGA88 como maestro
5. //a 400KHz con un oscilador de 8Mhz
6. void i2c\_iniciar(){
7. PORTC|=((1<<4)|(1<<5)); //activa resistencias pull up para SCL y SDA
8. TWBR=2; //velocidad 400Khz, Fosc 8Mhz, prescaler de 1
9. TWCR|=(1<<TWEN); //módulo TWI iniciado
10. }

### **Antes de empezar una comunicación I2C AVR**

Al realizar una tarea como un inicio, un envío o una recepción de datos por parte del maestro se pondrá el bit TWINT a 1, luego este bit se pondrá automáticamente a 0 y para detectar que ya se realizó la tarea por parte del módulo TWI será necesario esperar que el bit TWINT se ponga a 1, mientras eso no ocurra se tendrá que esperar, esto se realizará con una función a la que se le llamará **i2c\_espera()**, la que en el ATMEL STUDIO puede ser hecha así:

1. ////////////////////////////////////////////////////////////
2. ////////////////////////////////////////////////////////////
3. // Función de espera: mientras el bit7 o bit TWINT del registro
4. // TWCR sea 0, el IC2 AVR se esperará
5. // antes de realizar alguna tarea
6. void i2c\_espera(){
7. while ((TWCR & (1<<TWINT)) == 0);//espera mientras el bit de interrupción sea 0
8. }

### **Para iniciar la comunicación I2C AVR**

Una vez inicializado el módulo TWI I2C AVR se puede iniciar la comunicación poniendo el bit7 TWINT, el bit5 TWSTA y el bit2 TWEN del registro TWCR a 1, luego se esperará a que el bit7 TWINT se vuelva a poner a 1 automáticamente, el inicio de la comunicación se hará mediante una función a la que se le llamará **i2c\_inicia\_com()**, se le puede llamar como se quiera, en el ATMEL STUDIO se puede hacer así:

1. ////////////////////////////////////////////////////////////
2. ////////////////////////////////////////////////////////////
3. // Función de inicio de la comunicación I2C AVR
4. void i2c\_inicia\_com() {
5. TWCR = (1<<TWINT)|(1<<TWSTA)|(1<<TWEN);//bandera de interrupción a 1, start, habilita I2C AVR
6. i2c\_espera(); //espera mientras el bit TWINT sea 0
7. }

### **Para detener la comunicación I2C AVR**

Si ya no se quiere seguir realizando una comunicación I2C AVR se puede detener la comunicación poniendo el bit7 TWINT, el bit5 TWSTO y el bit2 TWEN del registro TWCR a 1, la parada de la comunicación se hará mediante una función a la que se le llamará **i2c\_detener()**, se le puede llamar como se quiera, en el ATMEL STUDIO se puede hacer así:

1. ////////////////////////////////////////////////////////////
2. ////////////////////////////////////////////////////////////
3. // Función de parada de la comunicación I2C AVR
4. void i2c\_detener() {
5. TWCR = (1<<TWINT)|(1<<TWSTO)|(1<<TWEN);//bandera de interrupción a 1, detener, habilita I2C AVR
6. }

### **Para la transmisión de datos del maestro al esclavo**

Para enviar datos al esclavo el bit7 TWINTy el bit2 TWEN del registro TWCR se tendrán que poner a 1, el envío de datos se hará mediante una función a la que se le llamará **i2c\_envia\_dato()**, se le puede llamar como se quiera, en el ATMEL STUDIO se puede hacer así:

1. /////////////////////////////////////////////////////////////////
2. /////////////////////////////////////////////////////////////////
3. //Función de transmisión de datos del maestro al esclavo
4. void i2c\_envia\_dato(unsigned char dato) {
5. TWDR = dato;
6. TWCR = (1<<TWINT)|(1<<TWEN);//para empezar a enviar el dato
7. i2c\_espera();//cuando TWINT se ponga a 1 se habrá terminado de enviar el dato
8. }

### **Para la recepción de datos por el maestro desde el esclavo en forma continua**

Esto es si se quiere recibir más de un dato por parte del maestro se tendrá que habilitar el uso del bit ACK, los datos se leerán desde el registro **TWDR**y serán guardados en una variable, para este tipo de recepción el bit7 TWINT, el bit6 TWEA y el bit2 TWEN del registro TWCR se pondrán a 1, la recepción de datos se hará mediante una función a la que se le llamará **i2c\_recibe\_dato\_ack()**, se le puede llamar como se quiera, esta función devolverá el dato recibido, en el ATMEL STUDIO se puede hacer así:

1. /////////////////////////////////////////////////////////////////
2. /////////////////////////////////////////////////////////////////
3. //Función de recepción de datos enviados por el esclavo al maestro
4. //esta función es para leer los datos que están en el esclavo
5. //en forma continua, esto es tras leer uno se volverá a leer otro
6. unsigned char i2c\_recibe\_dato\_ack(){//maestro envía ack para seguir recibiendo
   * + 1. //más datos
7. TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWEA);
8. i2c\_espera();
9. return TWDR;
10. }

### **Para la recepción de un último dato por el maestro desde el esclavo**

Esto es si ya no se quiere recibir más datos por parte del maestro no se habilitará el uso del bit ACK, los datos se leerán desde el registro **TWDR** y serán guardados en una variable, para este tipo de recepción el bit7 TWINT, el bit6 TWEA y el bit2 TWEN del registro TWCR se pondrán a 1, la recepción de datos se hará mediante una función a la que se le llamará **i2c\_recibe\_dato\_nack()**, se le puede llamar como se quiera, esta función devolverá el dato recibido, en el ATMEL STUDIO se puede hacer así:

1. /////////////////////////////////////////////////////////////////
2. /////////////////////////////////////////////////////////////////
3. //Función de recepción de datos enviados por el esclavo al maestro
4. //esta función es para leer solo un dato desde el esclavo
5. //esto es tras leer uno ya no se volverá a leer otro
6. unsigned char i2c\_recibe\_dato\_nack(){//maestro no envía ack para no seguir recibiendo
   * + 1. //más datos
7. TWCR = (1<<TWINT)|(1<<TWEN);
8. i2c\_espera();
9. return TWDR;
10. }

### **Para conocer el estado de la comunicación I2C AVR**

Se tendrá que obtener el valor que está contenido en los 5 bits de más peso del registro **TWSR** seguido de 3 ceros, el número obtenido indicará el estado de la comunicación I2C AVR, este número tendrá que ser comparado con el valor del estado que le corresponde a la comunicación según tabla, para verificar si todo esta correcto y poder detectar posibles errores de comunicación, las tablas para los estados se encuentran en la hoja de datos del AVR, para el caso del ATMEGA88 se comentó líneas arriba en el registro TWSR.

1. /////////////////////////////////////////////////////////////////
2. //función para averiguar el estado de la comunicación I2C AVR
3. //útil para detectar errores, el valor que retorna esta función
4. //se compara con el estado que deberían indicar los bits del 7 al 3
5. //del registro TWSR según tabla, durante la comunicación I2C AVR,
6. uint8\_t i2c\_estado\_com(){
7. uint8\_t estado; //variable donde se almacena el estado de la comunicación
   * 1. //I2C AVR
8. estado = TWSR & 0xf8; //en la variable estado se guarda el valor de los 5 bits de
   * 1. //más peso del registro TWSR seguidos de 3 ceros,

//el número obtenido indica

//el estado en que se encuentra la comunicación I2C AVR

1. return estado; //la función retorna el estado de la comunicación
2. }