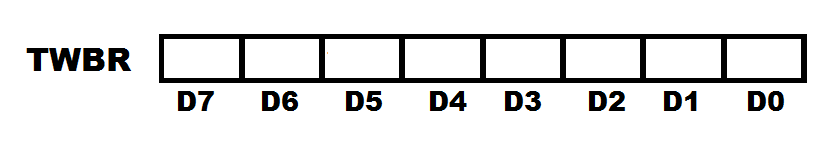
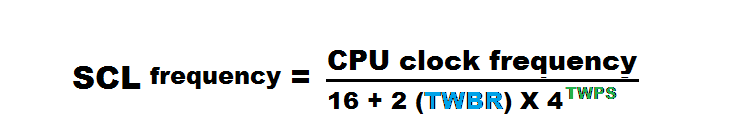
1. **xTWBR(TWI Bit Rate Register)**

****

**TWBR register help us to set the SCL clock frequency**

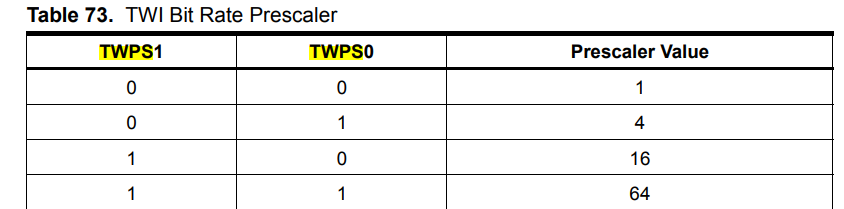
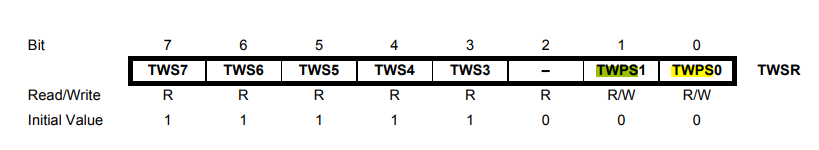


* TWPS is bit of TWSR register which full form is two wire prescale used to prescale the frequency
* For example – calculate the SCL frequency if value of TWPS in TWSR is 1 and TWBR is 38 (00100110).assume frequency of crystal is 8 MHz. (answer is 25 KHz).

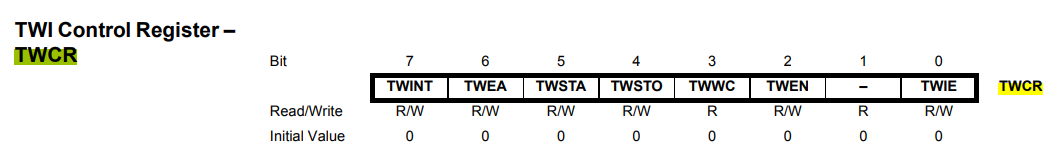
1. **TWSR(TWI Status Register)**

TWI status, these five bits shows the status of TWI control

TWPS – TWI Prescale bits

****

1. **TWCR(TWI Control Register)**

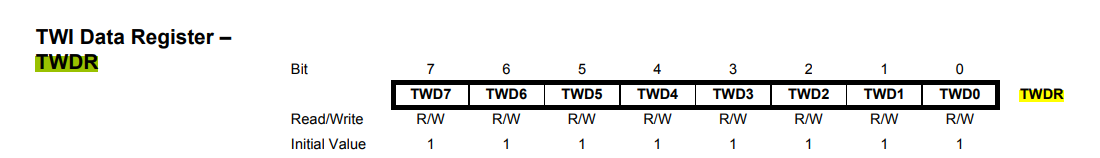
****

**TWEN para habilitar I2C**

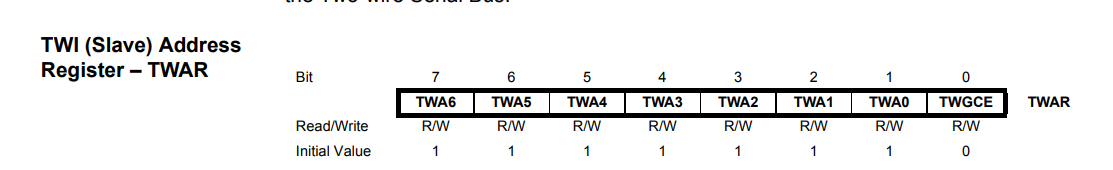
|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Description** | **0** | **1** |
| **TWINT** | **TWI interrupt** | **Transmission continue** | **Transmission complete** |
| **TWEA** | **TWI Enable ACKnowledgement** | **No ACK** | **Generate ACK in slave or receiving** |
| **TWSTA** | **TWI Start Bit** | **Generate start condition if bus is free** | **No start** |
| **TWSTO** | **TWI Stop Bit** | **Generate Stop Condition. Bit clear by hardware set by user** | |
| **TWWC** | **TWI Write Collision Flag** | **NO Collision Occured** | **Collision Occurred** |
| **TWEN** | **TWI Enable** | **Making this bit active TWI module is start** | |
| **TWIE** | **TWI Interrupt Enable** | **1 mean using by interrupt** | |

1. **TWDR(TWI Data Register)**

Used to hold the data to be transmitted

****

1. **TWAR (TWI Address Register)**

****

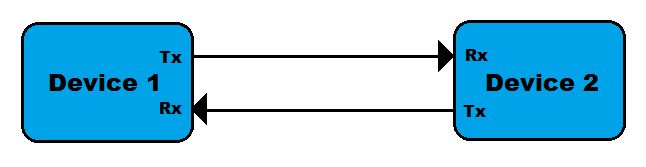
Used to hold address to access device

**I2C Information**

Usually embedded systems are complex designs which include multiple peripherals interconnected like LED, LCD, RTC, other microcontroller, Memory card, etc with main computer or master to expand its capabilities. All these peripheral’s interfacing is done by standard rules which is called protocol. Protocol is a set of rules that defines how communication between systems and devices are done which include bit ordering, bit pattern meanings, creating data frames, error checking, etc. UART, SPI, I2C, USB and Ethernet are some of the protocols widely used in embedded systems for serial data communication.

The Inter-integrated Circuit (I2C) Protocol was originally developed in 1982 by Philips used to connect multiple devices with only two wires, that’s why it is also called two wire interface protocol.

**Comparing I2C with UART**

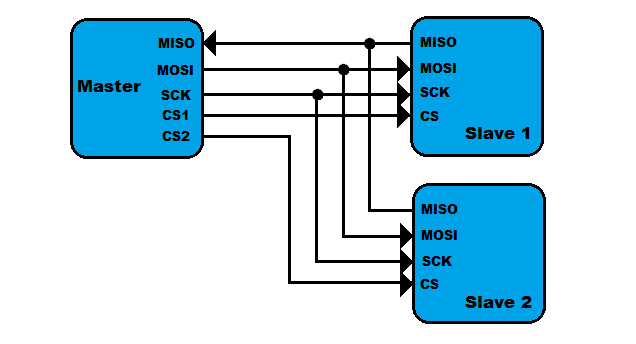
****In UART, separate clock lines are not used with data transmission so both devices should agree on the same Baud-rate. If differences occur between Baud-rates on either end, it will cause garbled data. In I2C a separated clock line is used.

UART uses hardware overheads like start bit, stop bit, parity bit or CRC bit in each frame of transmission which will lead to increase in transmission time while I2C uses only acknowledgement bit (ACK) or no acknowledgement bit.

Another drawback of UART is that they are suited to communications between two, and only two, devices while I2C can used with multiple devices and different modes like multi-master or single master, etc.

Data transfer speed is also an issue for UART, they are limited to some extent and maximum baud-rate is around 230400 bps.

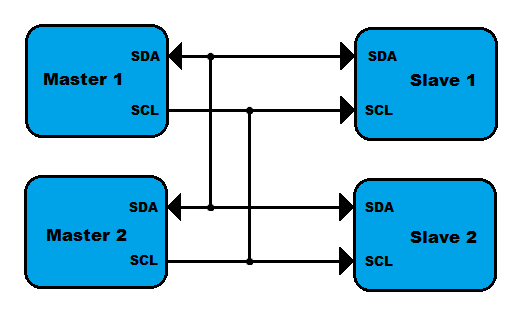
**Comparing I2C with SPI**

****

The first drawback of SPI is the pin used to connect multiple devices. If we connect two devices with SPI standard it will use four pins MISO, MOSI, SCK, CS. In SPI protocol, there is only a single master but it can support an arbitrary number of slaves. While in I2C, multiple masters can be possible.

If we compare data transmission speed of SPI and I2C, then SPI will definitely win. SPI supports speeds up to 10MHz (10 Mbps).

**Advantages:**

****

I2C requires only two lines (two wires), but those two wires can support up to 1008 slave devices. Unlike SPI, I2C can support a multi-master system, allowing more than one master to communicate with all devices on the bus (although the master devices can’t talk to each other over the bus and must take turns using the bus lines). Data rates of I2C devices can communicate at the rate of 100kHz to 400kHz.

**Referencias:**

<http://www.firmcodes.com/microcontrollers/avr/i2c-protocol-of-atmega16/registers-associated-with-i2c/>

<http://www.firmcodes.com/i2c-protocol/>

<http://www.firmcodes.com/microcontrollers/avr/i2c-protocol-of-atmega16/>

Notas de alfred:

Se utiliza dos resistencias de pull up en las dos líneas porque es configuración open-drain

El master inicia y termina la transmisión de datos.

Address/Data packet format: 9 bits(7 address|W(0)/R(1) y se responde con ACK en 0)

Si el ACK no fue recibido , puede transimitir condición de STOP o de REPEATED START

000 0000 es dirección de general call(broad cast) siempre se pone W, no R

direccicónes 1111 xxxx deben reservarse para futuros propositos

**Bit Rate Generator Unit:**

la frecuencia de los slaves tiene que ser al menos 16 veces más grande que el del master.

Proceso general:

1. Enviar condición START(1x10x10x). Escribir en TWCR(TWINT-1, se apaga con 1)
2. Se actualiza solo TWSR
3. Checar TWSR para confirmar condición START.
   1. SLA+W en TWDR, escribir TWCR para indicar que listo para transmitir, TWCR(TWINT-1, se apaga con 1). enviar dirección slave
   2. Subrutina de error
4. TWINT-1 indica paquete recibido, se actualiza solo TWSR
5. Checar TWSR para confirmar transmisión de dirección y ACK.
6. datos en TWDR, escribir TWCR para indicar que listo para transmitir, TWCR(TWINT-1, se apaga con 1). enviar datos
7. Subrutina de errror

6. TWINT-1 indica paquete recibido, se actualiza solo TWSR

7. Checar TWSR para confirmar transmisión de datos y ACK.

1. escribir condición de STOP en TWCR(1x01x10x), , TWCR(TWINT-1, se apaga con 1). enviar condición de STOP
2. Subrutina de errror

After a repeated START condition (state $10) the Two-wire Serial Interface can access the same Slave again, or a new Slave without transmitting a STOP condition. Repeated START enables the Master to switch between Slaves, Master Transmitter mode and Master Receiver mode without losing control of the bus.