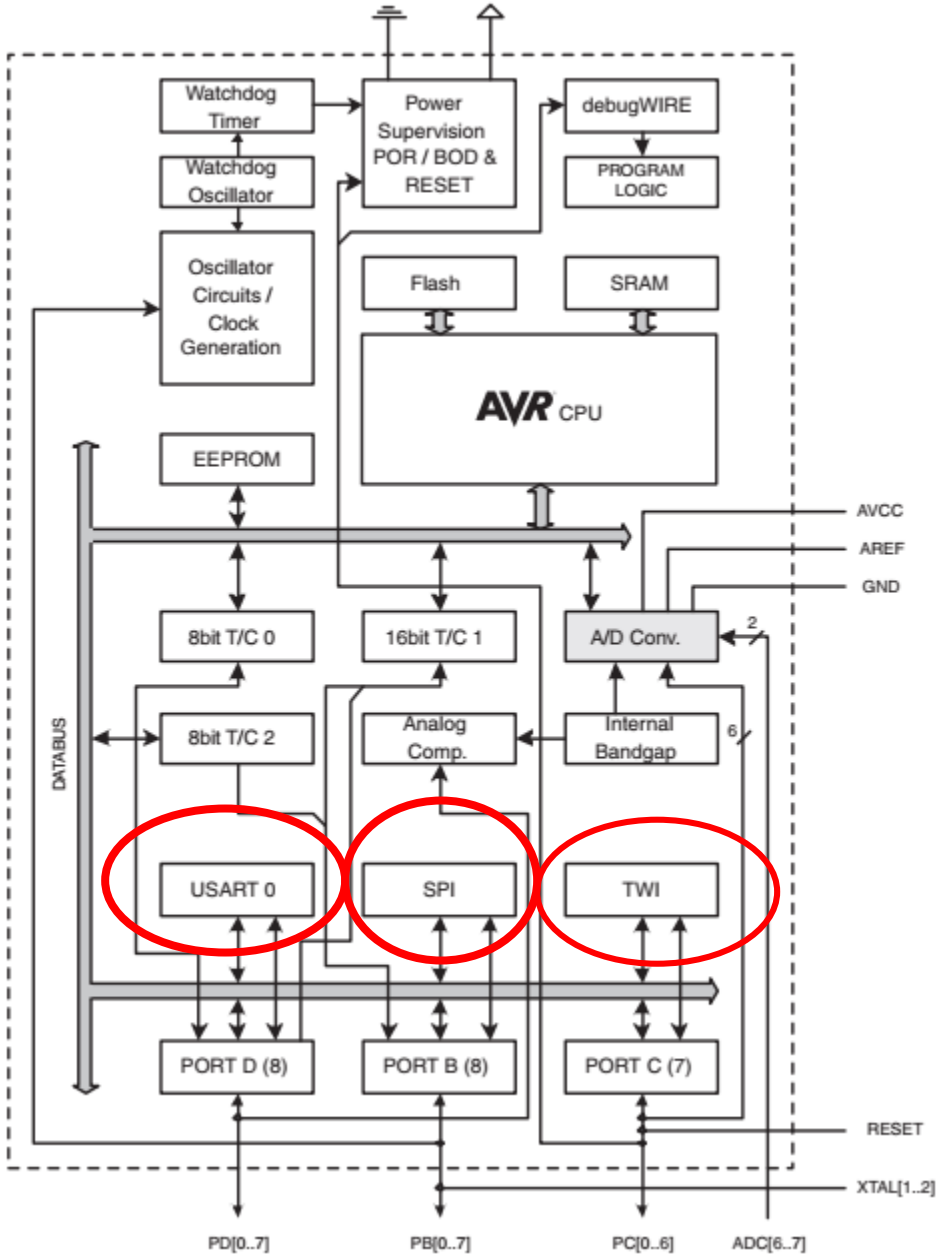
CIRCUITOS DIGITALES Y MICROCONTROLADORES 2025

Facultad de Ingeniería UNLP

Interfaz de comunicación serie I2C (TWI)

Ing. José Juárez

Periféricos de comunicación serie



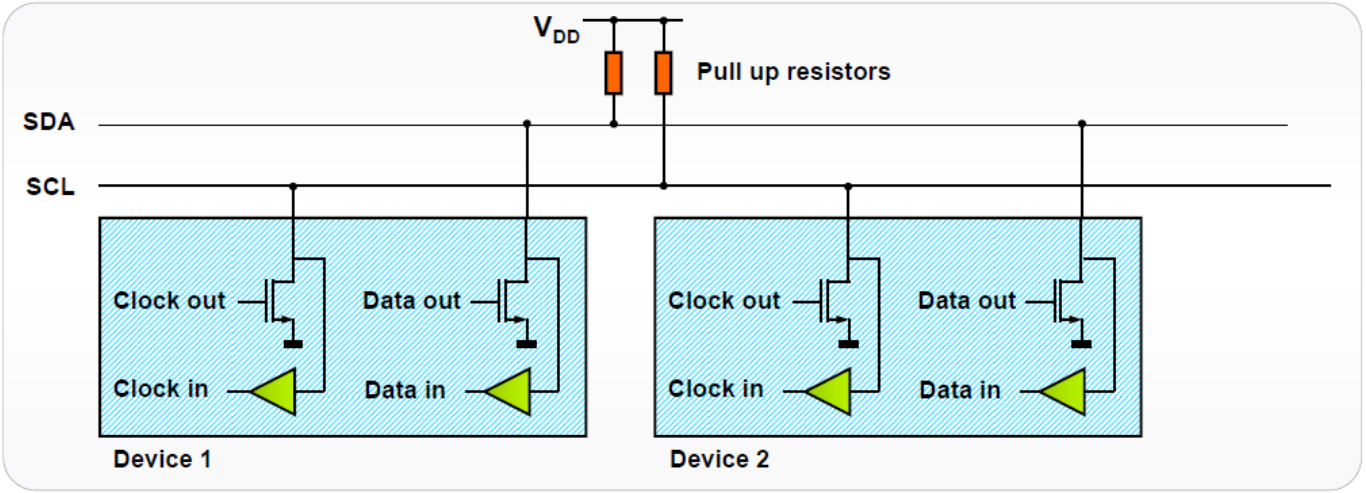
Source: Atmega328p Data Sheet

En los AVR el periférico se llama
TWI: Two Wire Interface

I2C (Inter Integrated Circuit) *

- Propuesto por Philips (hoy NXP) a principios de los 80´s es una interfaz serie sincrónica bidireccional para interconectar dispositivos con solo 2 cables en distancias cortas (por ejemplo dispositivos en un mismo PCB).
- En 1992 la versión 1 del protocolo se convirtió en un estándar para la industria de semiconductores que empezaron a incorporar dispositivos compatibles. En 1998 se pasó a la versión 2.
- En 2006 se liberó la patente y fue masivamente adoptada. Además de los MCU podemos encontrar memorias EEPROM, SRAM, diferentes sensores , conversores AD y DA y displays OLED/LCD con esta interfaz.
- La versión 3 completa (2007) se encuentra en la propia documentación de NXP (UM10204).

Esquema de interconexión

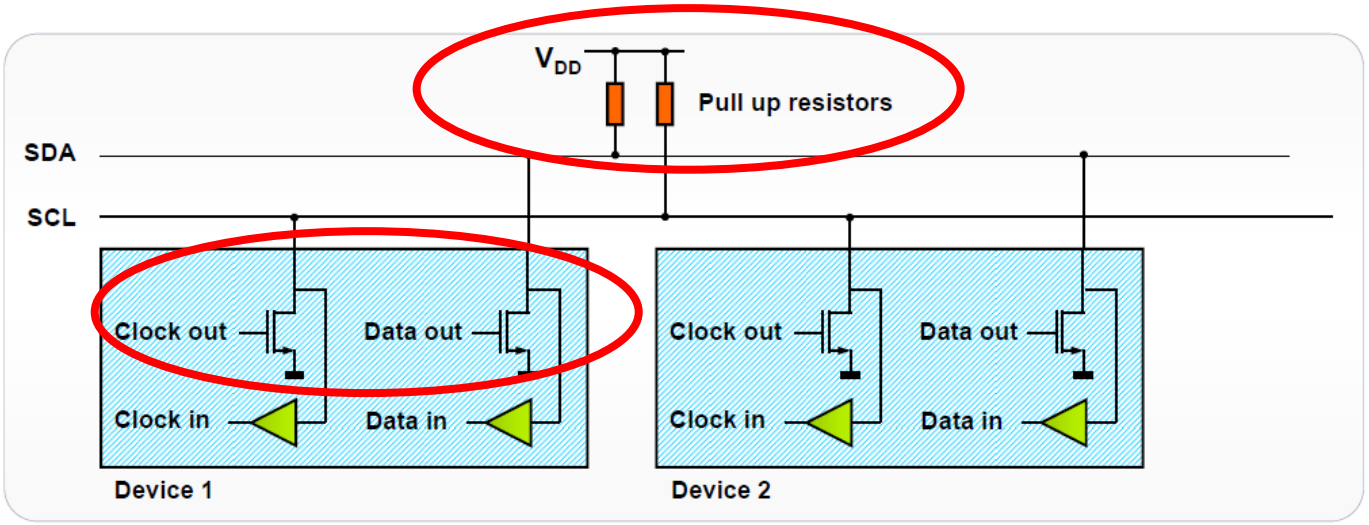


Bus de 2 conductores
SDA: Serial *Data*
SCL : Serial *Clock*

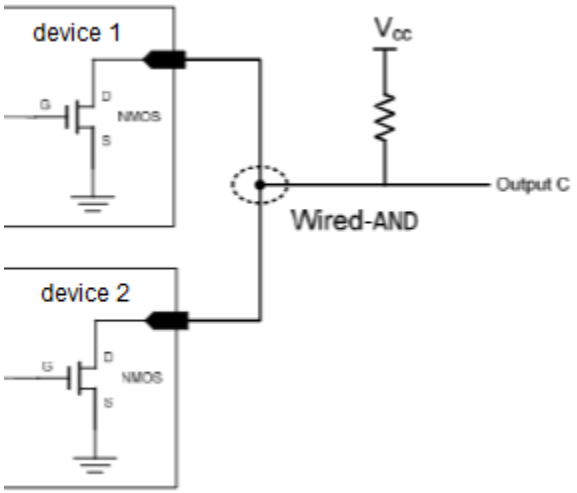
* www.nxp.com

I2C (Inter Integrated Circuit)

Esquema de interconexión



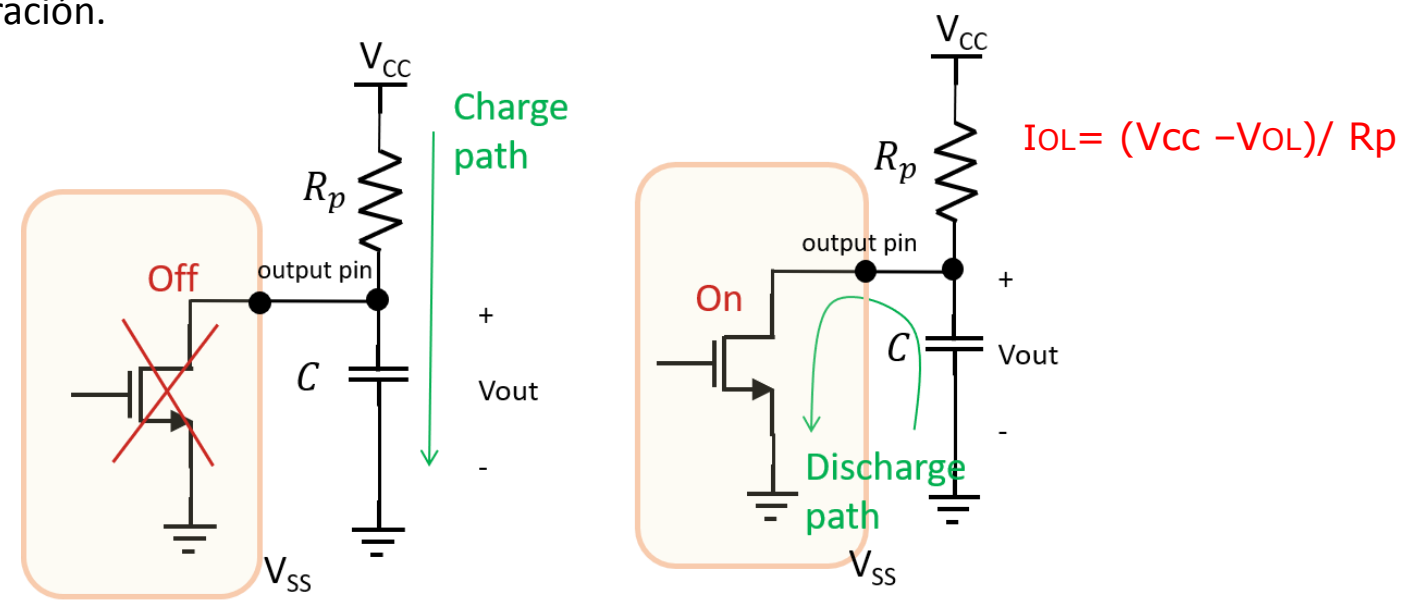
- Salidas Open-Drain/Open- Collector: lógica *AND cableada*
- VDD seleccionable según aplicación (3.3, 5V, etc)
- Resistencias de Pull-up según especificación de corriente máx y de la tasa de transferencia deseada, se recomienda:
  - 4.7K para modo estándar (f_{CLK} =100kHz)
  - 2.2K para modo fast (f_{CLK} =400kHz)
  - 1K para modo High Speed (f_{CLK} max=3.4MHz)



I2C (Inter Integrated Circuit)

Esquema de interconexión

- El número de dispositivos y la máxima distancia están limitados por la máxima capacidad en el bus (fijada en 400pF). La tasa de transferencia esta limitada por tiempo de subida máximo según el modo de operación.



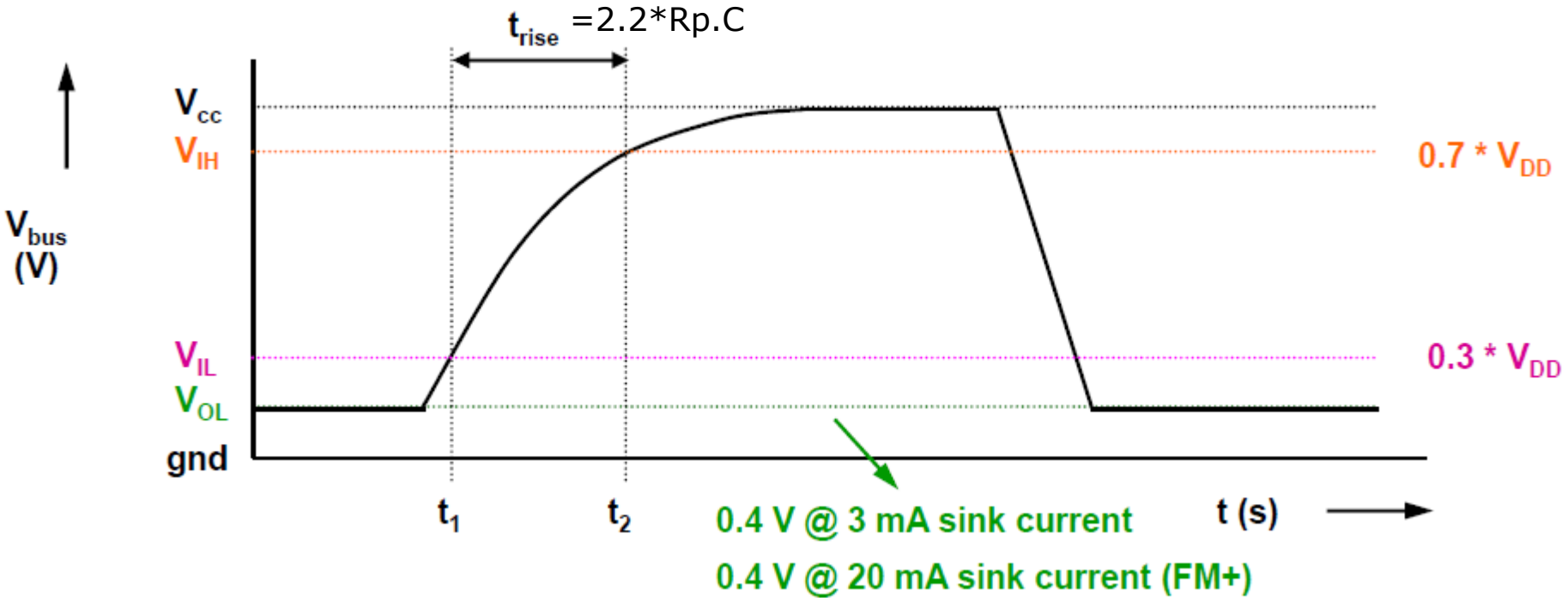
Todas las capacidades de entrada de los dispositivos conectados cargan el bus (modelada como C en la figura). Esta capacidad equivalente se carga a través de la resistencia de pull up Rp y se descarga a través de la resistencia Ron del puerto de salida del dispositivo que se activa.

Las ecuaciones de la carga y descarga de un capacitor son:

$V_{bus}=V_{cc}\,(1-e^{-t/RpC})$ $\qquad$ $V_{bus}=V_{cc}\,e^{-t/RonC}$

I2C (Inter Integrated Circuit)

- La dinámica de las transiciones digitales en cada línea del bus serán como se muestra en la figura:



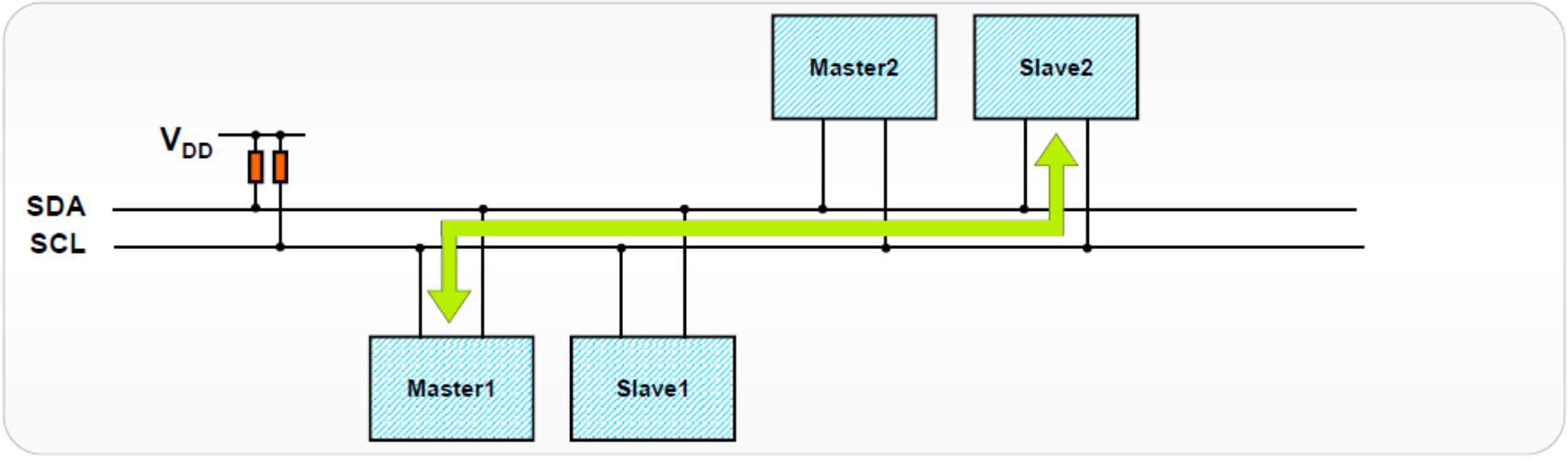
Si f_{CLK} =100kHz => Tclk=10us y por lo tanto: Tclk/2=5us

=> trise=2.2*Rp.C < 5us y Rp > Vcc/3mA

Se verifica para Rp=4.7k, C=400pF y Vcc=5V

I2C (Inter Integrated Circuit)

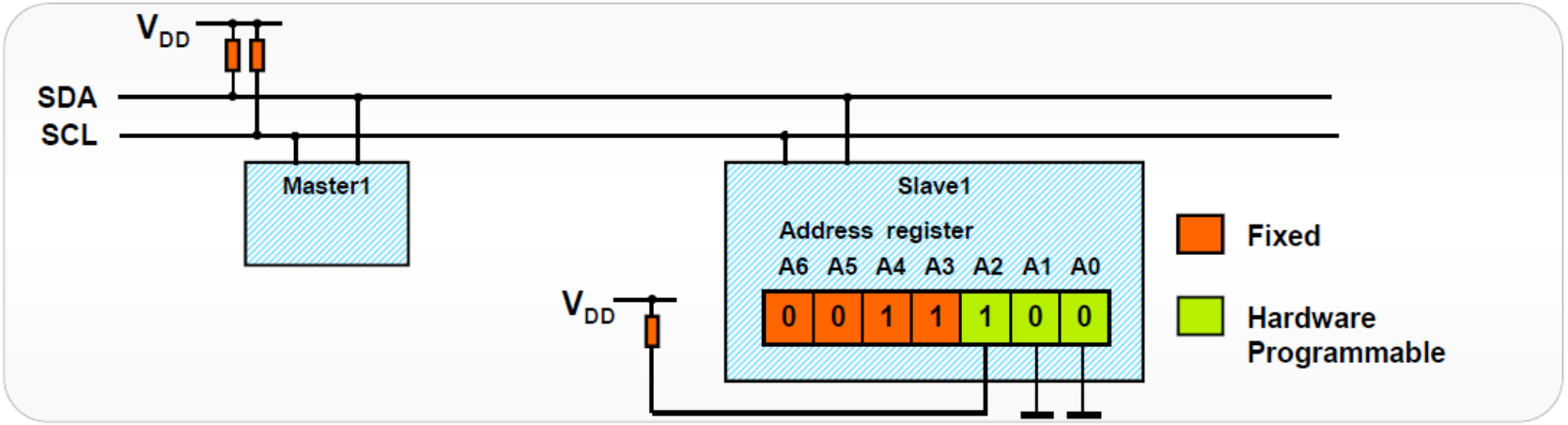
- Características funcionales:



- Multi master: puede haber más de un dispositivo que inicie la comunicación
- Multi slave: pueden conectarse múltiples dispositivos slaves.
- Half-dúplex: la comunicación es bidireccional pero en un sentido a la vez.
- Direccionamiento: Cada esclavo se identifica con una Dirección única de 7bits (o 10bits)
- Prevé la Colisión multi maestro (Arbitración y sincronización mediante la línea SDA)
- Control de flujo por parte del receptor (inserción de wait-states en la línea SCL)

I2C (Inter Integrated Circuit)

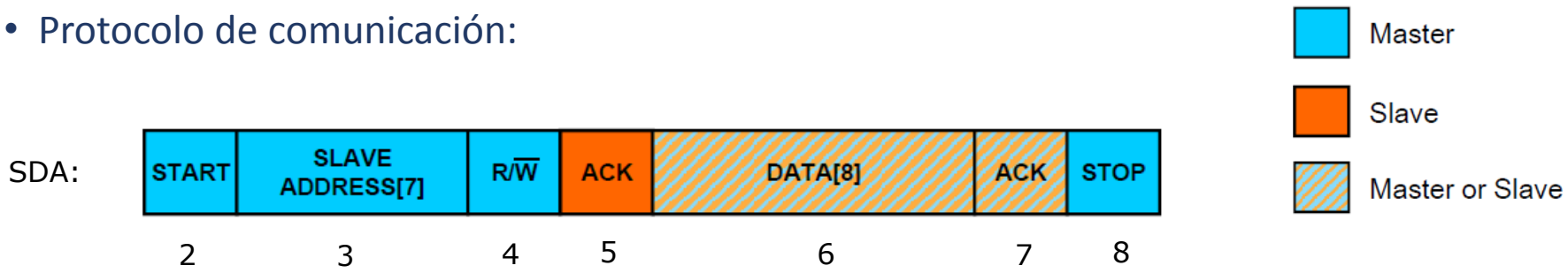
- Direccionamiento y selección del dispositivo:



- Dirección única para cada dispositivo esclavo (posee una parte programable por cableado de terminales)
- Generalmente los pines programables son 3. Esto permite conectar hasta 8 dispositivos del mismo modelo (es una clara limitación)
- Formato de direcciones de 7bits permite hasta 112 nodos (16 direcciones reservadas) y 10bits (1008 nodos)
- Direcciones de 10 bits (introducidas en V2) deben utilizar mensajes de 2 bytes

I2C (Inter Integrated Circuit)

- Protocolo de comunicación:

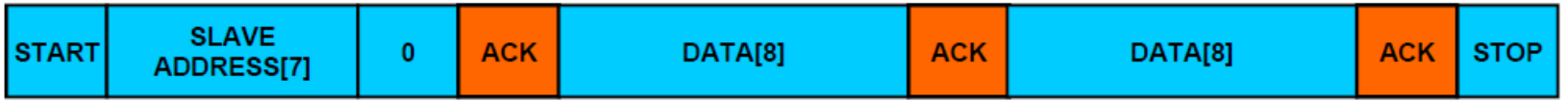


1) El bus se encuentra en estado IDLE cuando ambas líneas SDA y SCL están en ALTO
2) La comunicación la debe comenzar el maestro con una condición de START
3) Luego debe especificar la dirección del dispositivo esclavo (caso 7bits)
4) A continuación debe establecer si la transferencia es de lectura/escritura (1 bit más)
5) El esclavo que corresponda a la dirección presentada debe responder con una condición ACK (bit de Acknowledge)
6) A partir de aquí, el dispositivo que corresponda debe enviar los datos según condición R/W
7) Y el dispositivo que recibe dichos datos (master o slave) debe establecer una confirmación: ACK o NACK
8) El maestro debe finalizar la comunicación con una condición de STOP

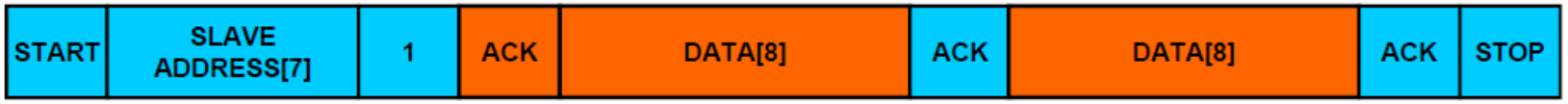
I2C (Inter Integrated Circuit)

- Ejemplos de transacciones entre dispositivos

- Si el master envía datos al esclavo (bit R/W=0)



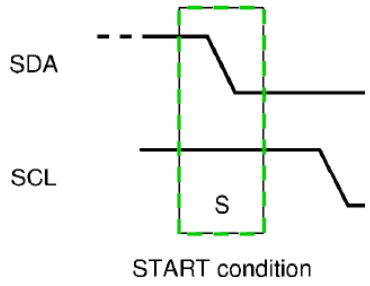
- Si el master recibe datos del esclavo (bit R/W=1)



I2C (Inter Integrated Circuit)

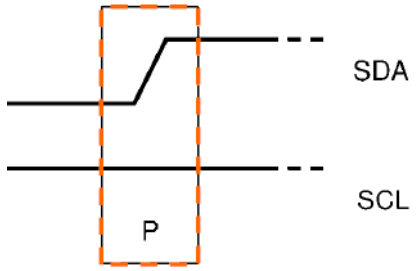
A nivel de señales del bus tenemos:

- Condición de START:



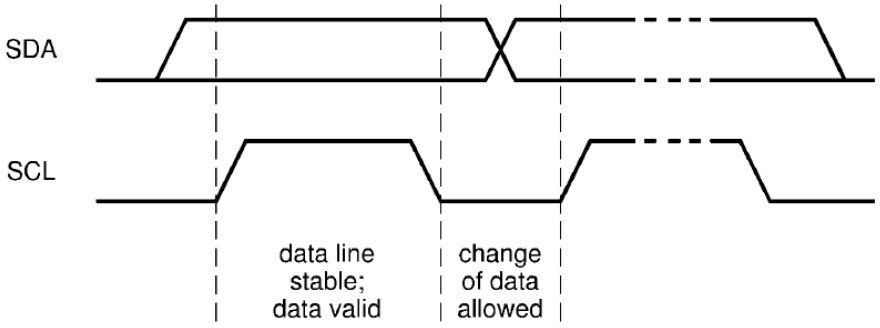
START condition

- Condición de STOP:



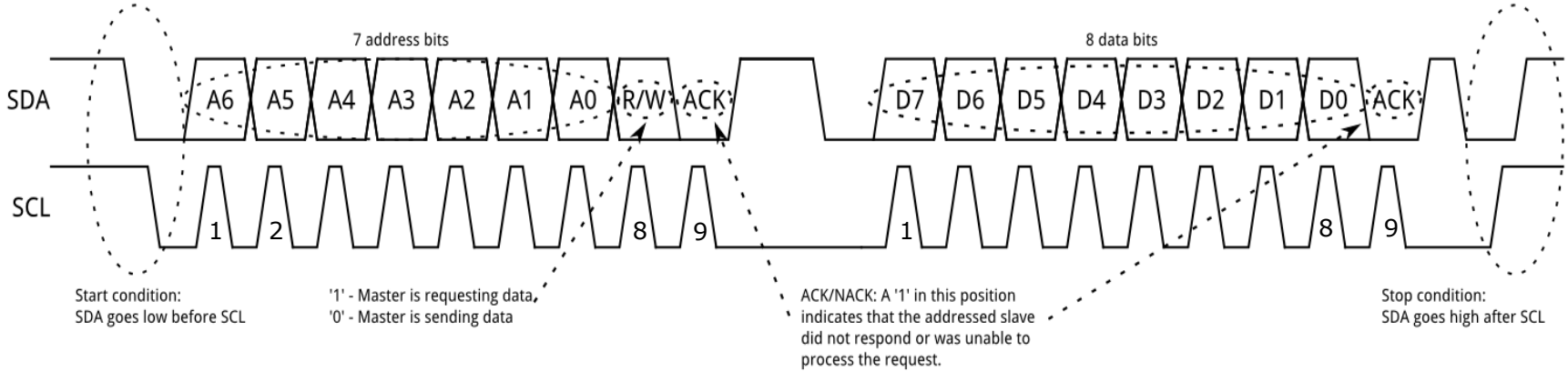
STOP condition

- Transferencia de bits:



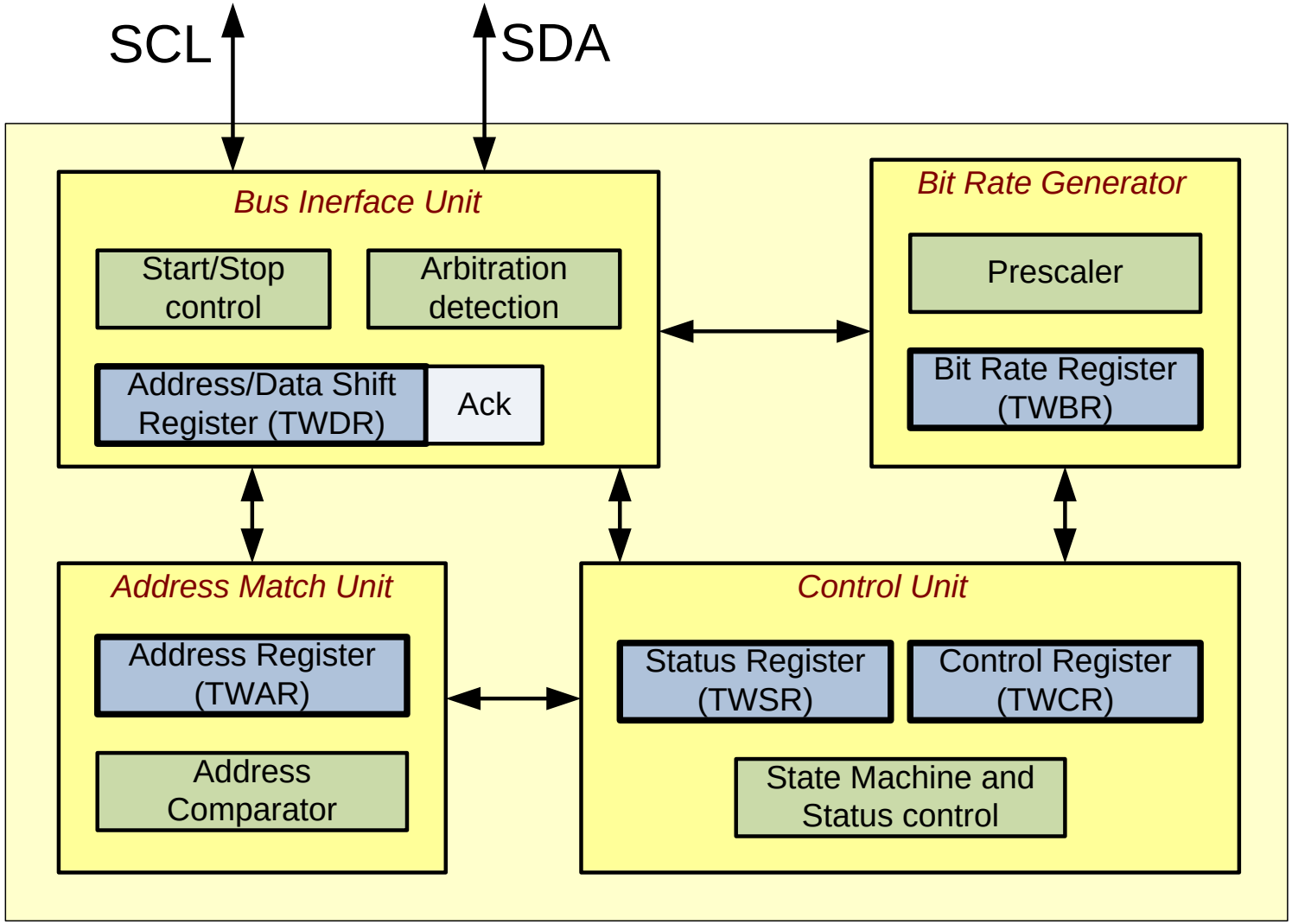
I2C (Inter Integrated Circuit)

- Ejemplo de una Transferencia completa a nivel de señales del bus

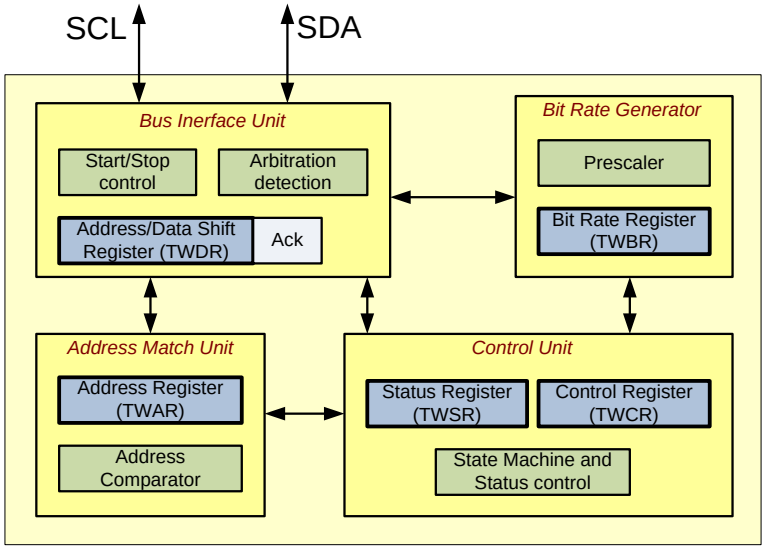


- Cada byte debe tener su correspondiente ACK (9no pulso de reloj)
- La cantidad de paquetes de datos en cada transferencia no esta limitada
- Si el esclavo no puede leer o escribir otro byte (está ocupado por ejemplo) puede forzar al master a esperarlo (control de flujo). Para esto el esclavo debe mantener SCL en bajo (clock stretching)
- Un NACK puede presentarse en el caso que el esclavo no esté conectado o disponible en el bus. En tal caso el master debe terminar la transferencia con un STOP o reintentar con un nuevo START.
- Finalmente, un NACK por parte del master (operación read) significa “fin de la comunicación” y el esclavo libera SDA para que el maestro presente STOP.

I2C (TWI) en AVR



I2C (TWI) en AVR



- **Bus Interface Unit**
  - Detecta y genera START, REPEATED START y STOP.
  - Detecta colisión (arbitración)
  - Controla la recepción y envío del ACK
  - transfiere los datos y direcciones.
- **Bit Rate Generation Unit**
  - Controla la frecuencia de la señal de reloj (SCL) en modo maestro
- **Address Match Unit**
  - Compara la dirección recibida con el TWI address register e informa si hay coincidencia (modo esclavo).
- **Control Unit**
  - Configura y controla la operación del modulo con el TWI control register
  - Actualiza el estado del status register

TWBR (TWI Bit Rate) Register

TWBR:

TWSR:

TWS7	TWS6	TWS5	TWS4	TWS3	-	TWPS1	TWPS0

Status bits — TWPS

$$SCL\ \ frequency = \frac{XTAL}{16 + 2 \times TWBR \times 4^{TWPS}}$$

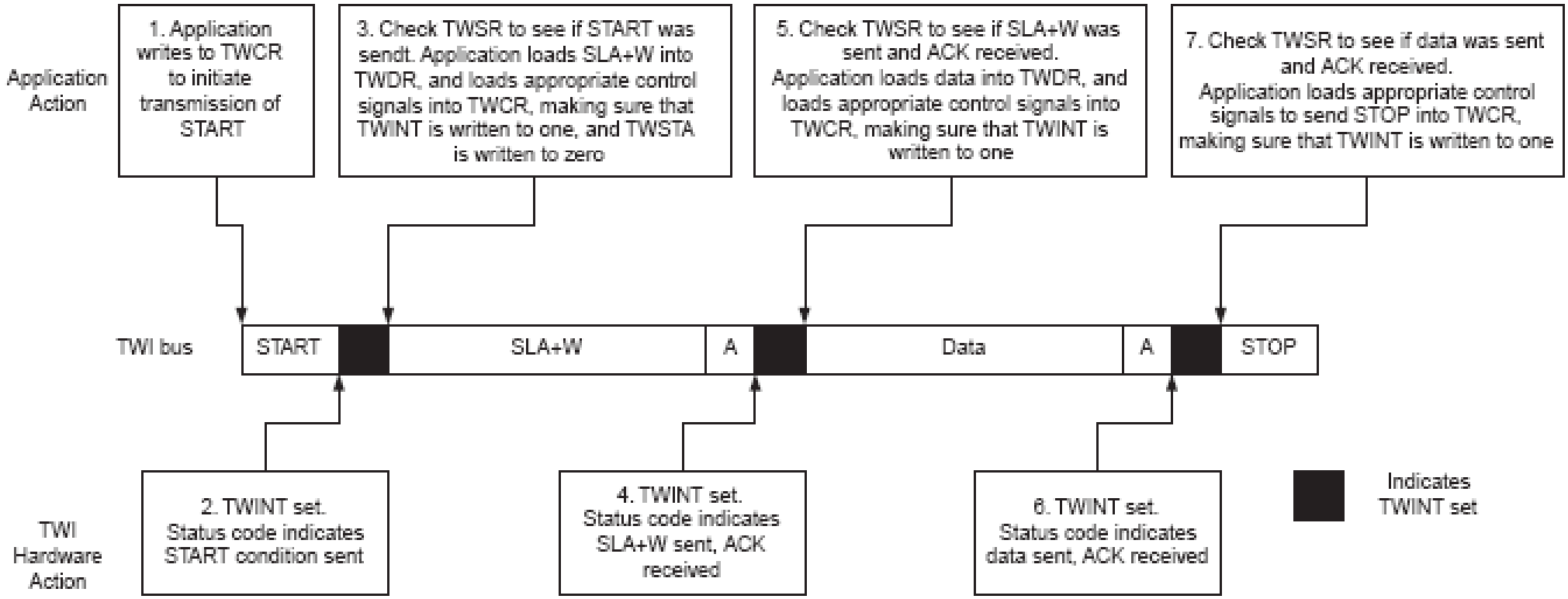
Si fxtal=16MHz ,TWBR=72, TWPS=0 => f_{SCL}=100kHz

TWCR Control Register

TWCR:	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE

- TWINT: TWI Interrupt flag
- TWEA: TWI Enable Acknowledge bit
  - 1:ACK, 0:NACK
- TWSTA: TWI Start condition bit
- TWSTO: TWI Stop condition bit
- TWWC: TWI Write Collision flag
- TWEN: TWI Enable bit
- TWIE: TWI Interrupt Enable

TWI Control Register



Pág. 223, CH22, Atmega328p data sheet

TWAR (TWI Address Register)

TWAR:

TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE

- TWA6-0 (TWI slave Address)
- TWGCE (TWI General Call Recognition Enable bit)
  - 1: Answer to general call

TWI Master Mode

- **Inicialización**
  - Configurar la frecuencia de reloj SCL.
  - Habilitar el periférico con el bit TWEN

```
void i2c_init(void)
{
  TWSR=0x00;     //set prescaler bits  to zero
  TWBR=152;      //SCL frequency is 50K for XTAL = 16M
  TWCR=0x04;     //enable the TWI module
}
```

- **Enviar START y esperar confirmación**

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

TWI Master Mode

- **Enviar datos**
  - Copiar dato a enviar al TWDR
  - TWEN =1 y TWINT =1 para iniciar transmisión
  - Esperar activación del flag TWINT (transmission complete)

```
void i2c_write(unsigned char data)
{
  TWDR = data ;
  TWCR = (1<< TWINT)|(1<<TWEN);
  while ((TWCR & (1 <<TWINT)) == 0);
}
```

- **Recibir datos**
  - TWEN =1 y TWINT =1 para iniciar la recepción
  - Esperar activación del flag TWINT (reception complete) y leer el dato desde el TWDR

```
unsigned char i2c_read(unsigned char isLast)
{
  if (isLast == 0)   //send ACK
    TWCR = (1<< TWINT)|(1<<TWEN)|(1<<TWEA); //send ACK
  else
    TWCR = (1<< TWINT)|(1<<TWEN); //send NACK
  while ((TWCR & (1 <<TWINT)) == 0);
  return TWDR;
}
```

TWI Master Mode

- **Enviar STOP**

```
void i2c_stop()
{
  TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWSTO);
}
```

- Note: no usar el flag TWINT luego de enviar el STOP

Escribir y leer un byte en modo master

```
int main (void)
{
  i2c_init();

  //writing a byte
  i2c_start();       //transmit START condition
  i2c_write(0b11010000);    //transmit SLA + W(0)
  i2c_write(0b11110000);    //transmit data
  i2c_stop();             //transmit STOP condition

  //reading a byte
  i2c_start();       //transmit START condition
  i2c_write(0b11010001);    //transmit SLA + R(1)
  PORTD =i2c_read(1);   //read one byte of data
  i2c_stop();             //transmit STOP condition
  while(1);          //stay here forever
}
```

TWI Slave Mode

- **Inicialización**
  - Configurar la dirección que tendrá el esclavo en el TWAR register.
    - 7 bits for address
    - 8^{th} bit is TWGCE (1 = answer general calls)
  - TWEN =1 para habilitar el periférico
  - TWEN=1, TWINT=1, and TWEA=1 para habilitar TWI y la generación de ACK

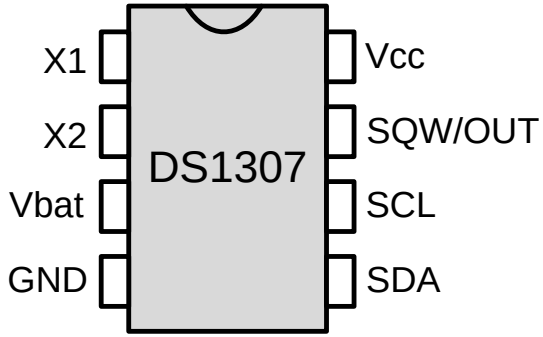
- **Escuchar**
  - sondear flag TWINT o usar la interrupción para detectar cuando un maestro envía la dirección del dispositivo

TWI Slave Mode

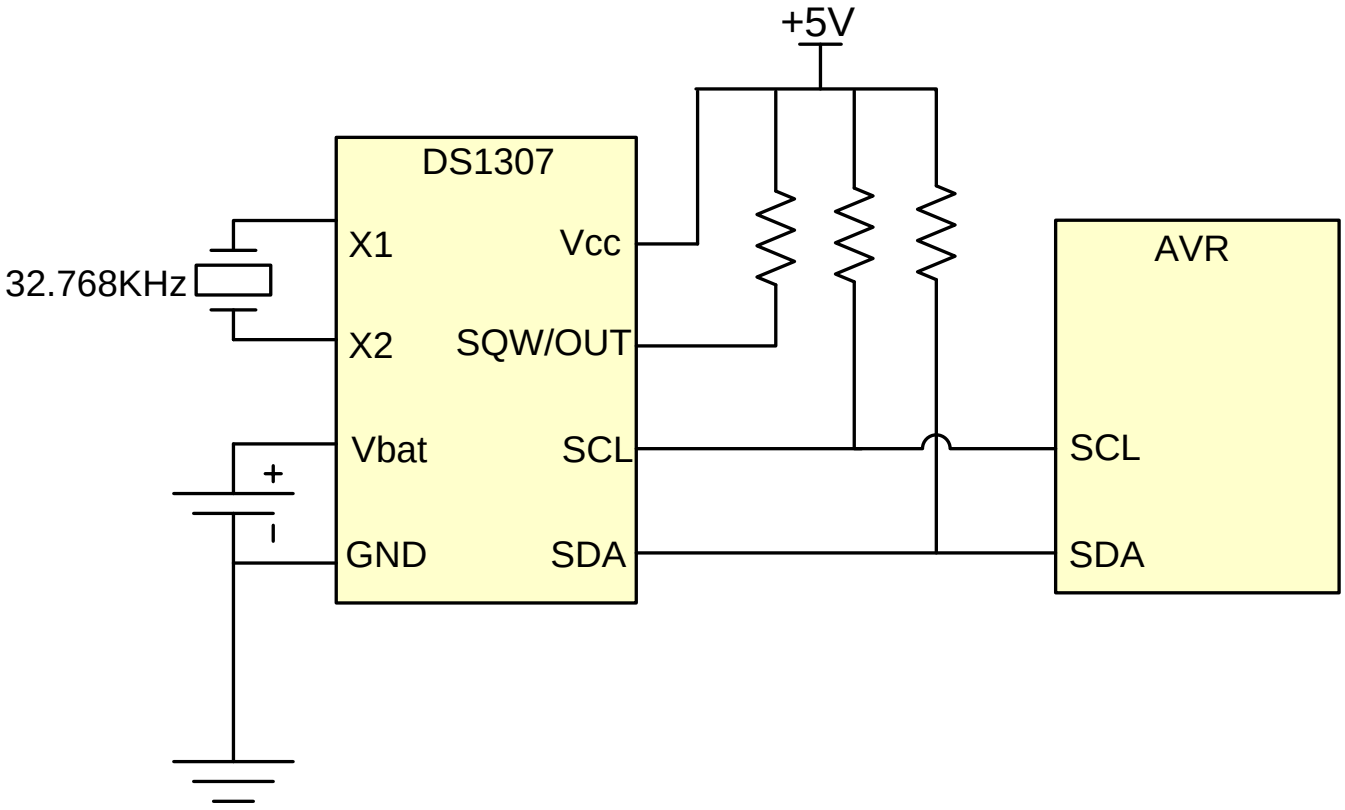
- **Enviar**
  - Copiar dato al TWDR
  - TWEN=1, TWEA=1 y TWINT=1 para iniciar transmisión
  - Esperar activación del flag TWINT (transmitted completely)

- **Recibir**
  - TWEN=1 y TWINT=1 para iniciar recepción
  - Sondear TWINT flag (received completely)
  - Leer el TWDR

Conectando el DS1307 y el AVR



- RTC : reloj de tiempo real con fecha y hora



DS1307 Mapa de memoria

- 64 bytes de RAM (Registros +RAM)
- Usa formato BCD
- Bit7, CH=0 enable the oscillator
- 0x07 es el control register

En el DS1307 hay un puntero a memoria que especifica el byte al que se accederá en el próximo comando de lectura o escritura. Después de cada operación de lectura o escritura, el contenido del puntero de registro se incrementa automáticamente. Esto es útil en la lectura o escritura de varios bytes.

ADDRESS	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	FUNCTION	RANGE
00H	CH	10 Seconds			Seconds				Seconds	00–59
01H	0	10 Minutes			Minutes				Minutes	00–59
02H	0	12	10 Hour	10 Hour	Hours				Hours	1–12 +AM/PM 00–23
		24	PM/AM							
03H	0	0	0	0	0	DAY			Day	01–07
04H	0	0	10 Date		Date				Date	01–31
05H	0	0	0	10 Month	Month				Month	01–12
06H	10 Year				Year				Year	00–99
07H	OUT	0	0	SQWE	0	0	RS1	RS0	Control	—
08H-3FH									RAM 56 x 8	00H–FFH

Escribir el DS1307

- Enviar START
- Enviar la dirección del DS1307 (1001101) + un 0 para indicar operación de escritura
- Enviar la dirección de memoria interna que queremos acceder (valor del Register Pointer)
- Enviar uno o más bytes para configurar la fecha
- Enviar STOP

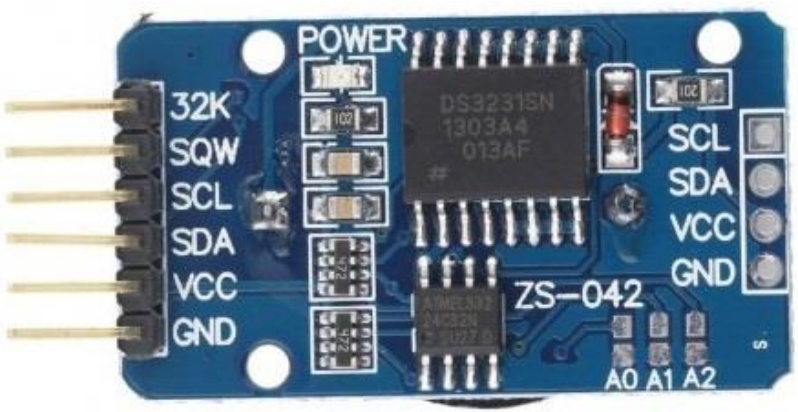
Leer el DS1307

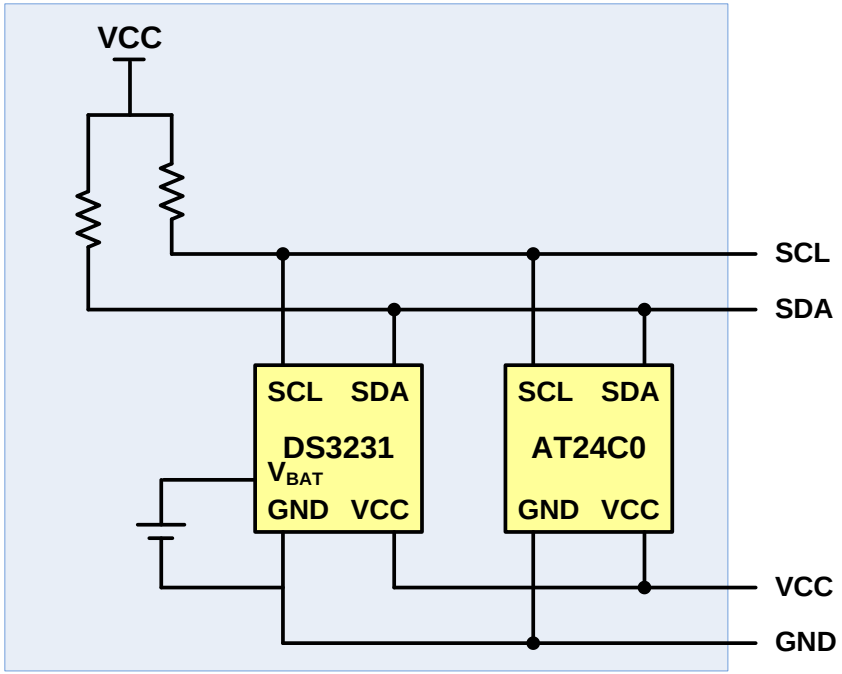
- Enviar START
- Enviar la dirección del DS1307 (1001101) + un 1 para indicar operación de lectura
- Recibir uno o más bytes
- Enviar STOP

  - Nota: el puntero de registro indica qué dirección se leerá (debe configurarlo mediante una operación de escritura)

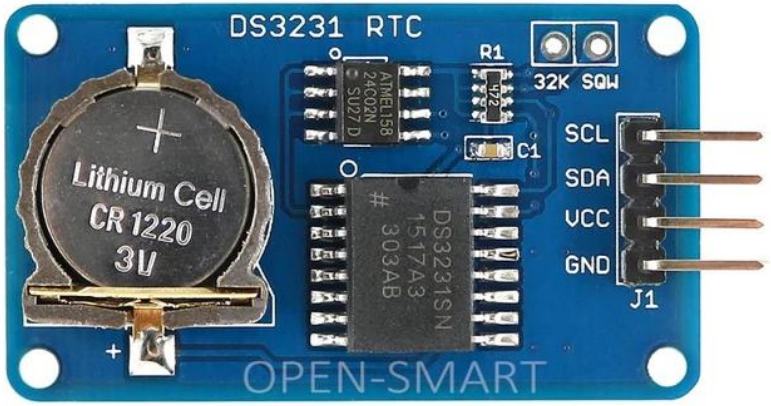
Módulo RTC - DS3231

- Sucesor del DS1307

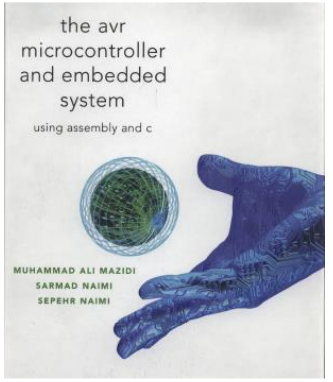




- *El modulo trae además una memoria EEPROM para respaldo de datos y un porta pila para mantener los datos en RAM sin alimentación externa.*



Bibliografía:



- *The AVR microcontroller & Embedded Systems*. Mazidi,Naimis, CH18

- Libro Digital de Felipe Espinosa: CH6.2 y CH6.3

- Hoja de datos: ATMEGA
- Wiki: https://en.wikipedia.org/wiki/I2C
- Videos ARDUINO:
  - RTC I2C
  - LCD I2C