



UNIVERSIDAD
DE GRANADA



Sistemas con Microprocesadores

Comunicaciones serie
en microcontroladores

Comunicación serie

Entre microcontroladores o microcontrolador con dispositivo

Larga distancia: RS232

- Fácil de implementar
 - Hardware/software
- Transmisión/recepción de bytes
 - Se pueden agrupar formando paquetes
- Velocidad moderada de transmisión
- Gran cantidad de bibliotecas / API / funciones para TX/RX
- Esta comunicación se puede virtualizar:
 - Sobre USB, Bluetooth

Corta distancia:

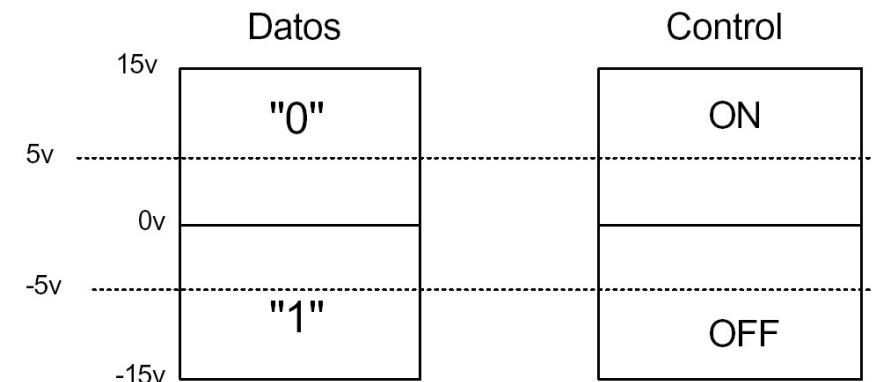
- I2C, SPI, One Wire, (USB) ...



RS-232

Características:

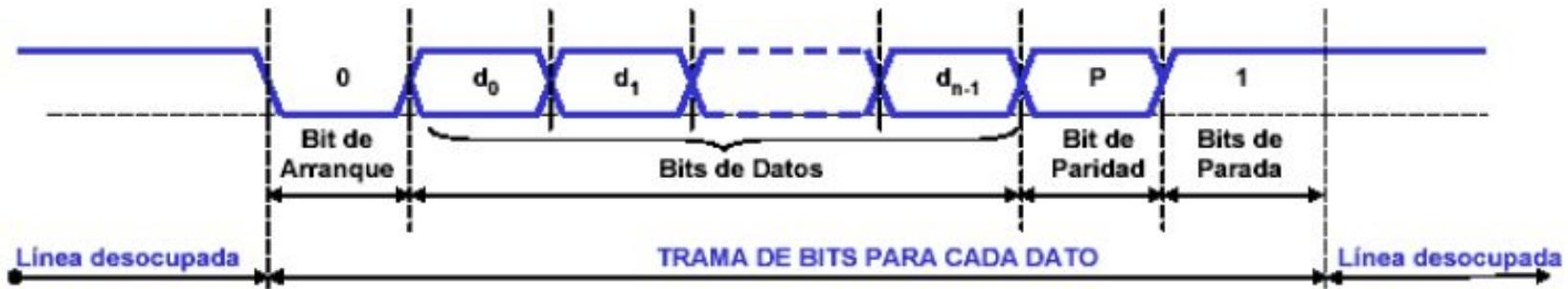
- Interconexión punto a punto
- Velocidades estandarizadas (300, ..., 115200 bps y superiores)
- Transmisión asíncrona (o síncrona)
- Modos *half-duplex* y *full-duplex*
- Tres especificaciones
 - Eléctrica (V.28)
 - Mecánica (ISO 2110)
 - Funcional (V.24)



Especificación eléctrica:

- Transmisión no balanceada
 - Referencias a 0V
- Limitación de corriente a 0.5 A
- Capacidad máxima 2500 pF

Formato de la trama:



RS-232

Especificación funcional:

- Líneas de datos:
 - TxD y RxD
- Líneas de control de flujo:
 - Request to send (RTS)
 - Clear to send (CTS)
 - Data Carrier Detected (DCD)
- Líneas de establecimiento de conexión:
 - Data Terminal Ready (DTR)
 - Data Set Ready (DSR)
 - Ring Indicator (RI)
- Líneas de referencia:
 - Masa (GND)
 - Masa de protección (SGH)



DB25



DB9

Especificación mecánica:

- Conector ISO 2110 (DB25)
 - 25 pines, se usan 21.
 - Numeración ISO (AA, BB, etc..) o CCITT (101, 102, etc.)
- Conector DB9
 - Versión IBM (asíncrona)
- Longitud del cable: Depende de los baudios: 15 m, 1 km...

RS-232: Control de flujo

□ Protocolos software

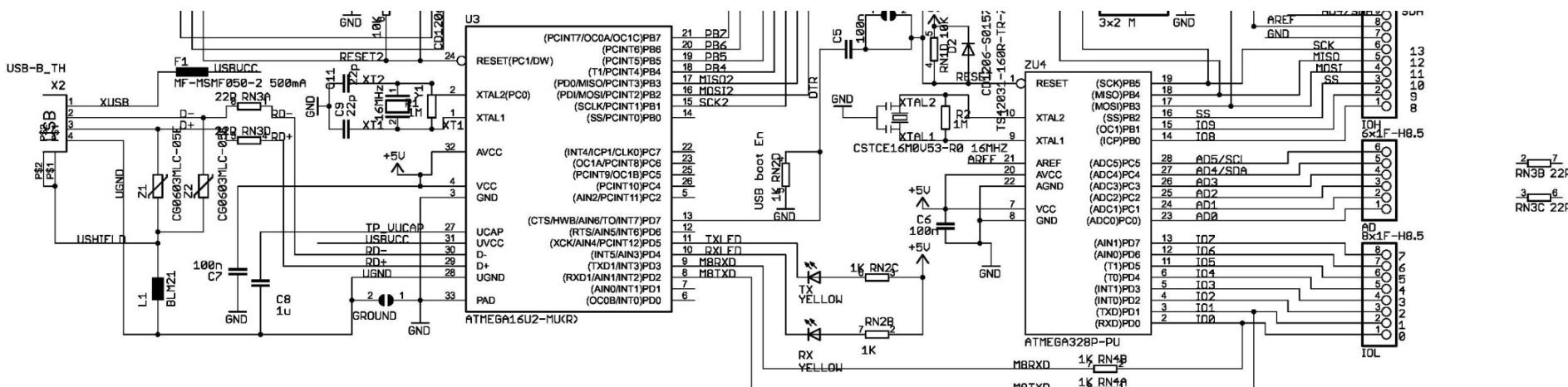
- XON – OFF: Caracteres especiales XON y XOFF. XOFF detiene el envío de datos desde el modem al PC, XON solicita más datos.
- ENQ – ACK (ASCII-BSC): Enquiry: solicitud de información. Acknowledge: confirmación (ASCII) .

□ Protocolos hardware

- RTS – CTS: líneas de control: Request To Send – Clear To Send.
- DTR – DSR: líneas de control: Data Terminal Ready – Data Set Ready.

Esquema Arduino: Comunicación Serie RS232 sin especificaciones eléctricas

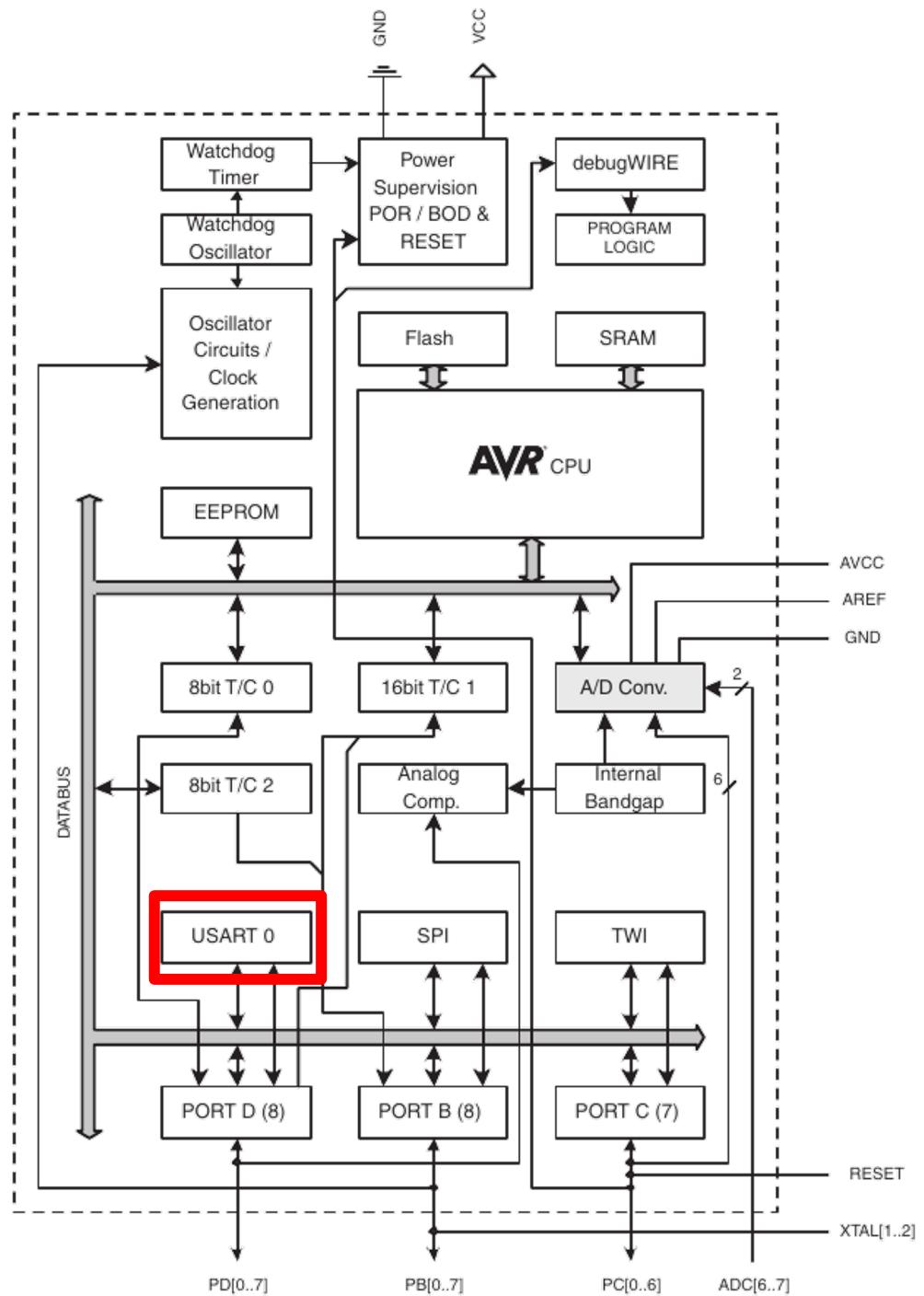
- Sobre pines digitales 0 (RX) y 1 (TX)
 - Arduino Mega tiene 3 puertos serie adicionales
- ATmega16 pasarela USB-serie



Cruce de líneas

- RX-TX
- TX-RX

USART en ATmega328P



Comunicación serie con Arduino

Llamada y respuesta (*handshaking*)

```
void setup() {  
    Serial.begin(9600); //Inicia el puerto a 9600 bps  
    while (!Serial) {} //Espera la conexión (sólo Leonardo)  
    pinMode(2, INPUT); //Entrada digital pin 2 (sensor a monit.)  
    establishContact(); //Envía un byte mientras no hay respuesta  
}  
  
void establishContact() {  
    while (Serial.available() <= 0) {  
        Serial.print('A'); // envía una A  
        delay(300);  
    }  
}
```

Comunicación serie con Arduino

```
void loop() {  
    if (Serial.available() > 0) { // si hay un byte válido  
        inByte = Serial.read(); // lee el byte  
  
        // lee el valor del conversor y /4 -> rango: 0..255  
        firstSensor = analogRead(A0) / 4;  
        delay(10); // espera 10 ms  
        // lee el segundo conversor  
        secondSensor = analogRead(A1) / 4;  
        thirdSensor = map(digitalRead(2), 0, 1, 0, 255);  
  
        Serial.write(firstSensor); // envía los valores  
        Serial.write(secondSensor);  
        Serial.write(thirdSensor);  
    }  
}
```

Puerto serie en Arduino

❑ Serial:

- ❑ <https://www.arduino.cc/reference/en/language/functions/communication/serial/>

- ❑ Serial.begin()
- ❑ Serial.write()
- ❑ Serial.print()
- ❑ if (Serial.available()) {
 salida = Serial.read()
}

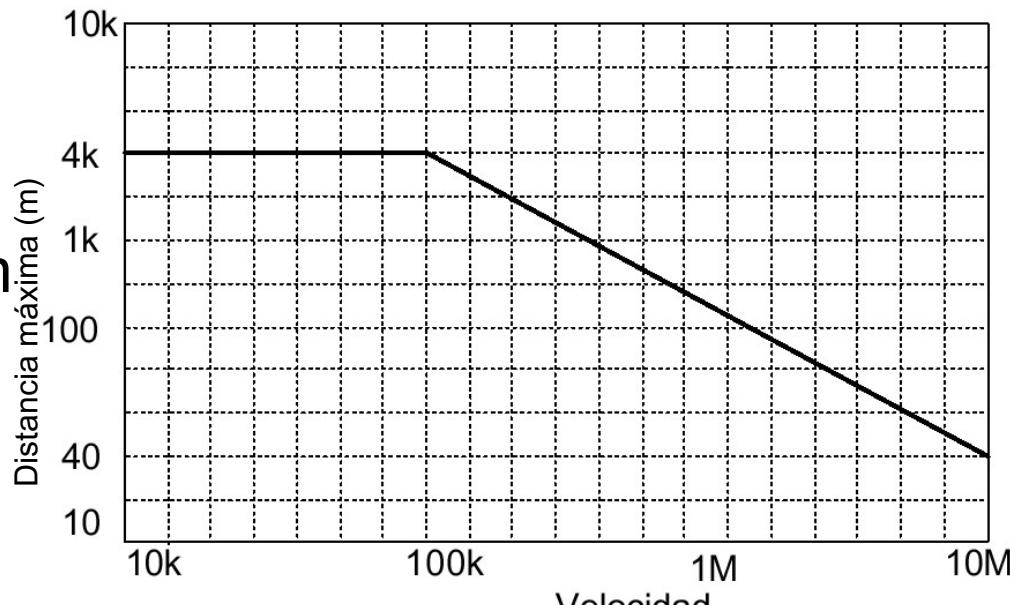
❑ Comunicación serie por Software:

- ❑ <https://www.arduino.cc/en/Reference/SoftwareSerial>

RS-422 y RS-423-A

□ Especificaciones eléctricas

- RS-423 no balanceada, voltaje más reducido y mayor sensibilidad que RS-232.
- RS-422 balanceada e impedancia controlada (100 ohmios).
- Niveles de tensión: -6V a 6V
- Sensibilidad 200 mV.
- 1 emisor, hasta 10 recept.
- Longitud de cable ~1500 m
- Utilizada en ambientes industriales, transmisión de vídeo, etc.



RS-485

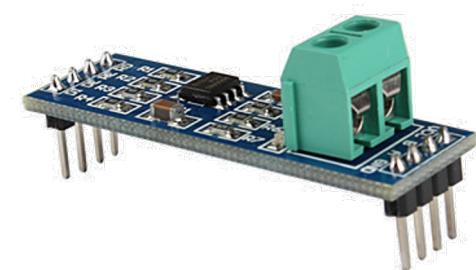
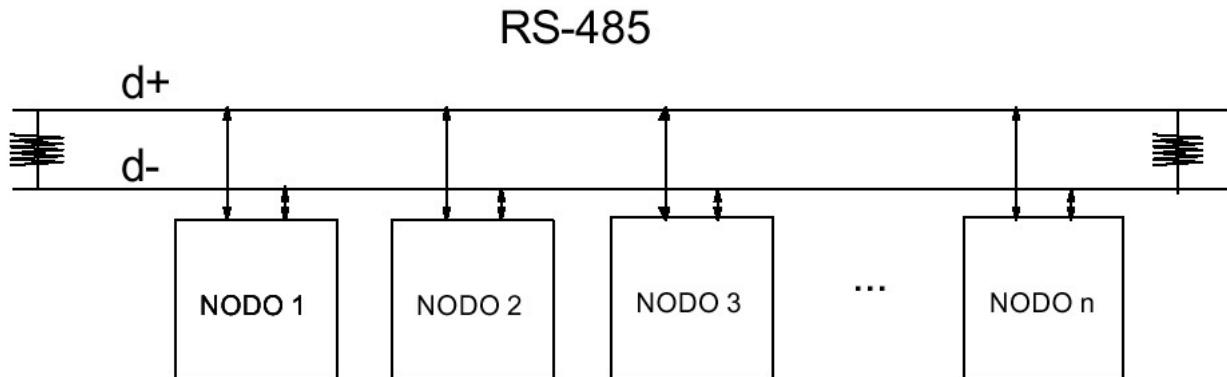
Multipunto

- Hasta 32 emisores y 32 receptores (equivalentes).
- Posibilidad de conexión en alta impedancia: capacidad de *hot plug*

Par trenzado

- 120 ohmios con terminadores.
- Resistencias de pull-up y pull-down.

Comunicación p2p. Maestro – Esclavo. Multimaestro.



Comparación de normas: RS-232, RS-423, RS-422, RS-485

NORMAS	RS-232	RS-423	RS-422	RS-485
Modo	Simple	Simple	Diferencial	Diferencial
Número transmisores	1	1	1	32
Número receptores	1	10	10	32
Longitud máxima	15 m	1200 m	1200 m	1200 m
Velocidad máxima	20 Kbp	100 Kbps	10 Mbps	10 Mbps
Salida transmisor	± 5 V min ± 15 V max	± 3.6 V min ± 6 V max	± 2 V min	± 1.5 V min
Carga al transmisor	3K a 7K	450 mÍn	100 min	60 min
R de entrada al receptor	3K a 7K	4K min	4K min	12 K min
Sensibilidad del receptor	± 3 V	± 200 mV	± 200 mV	± 200 mV

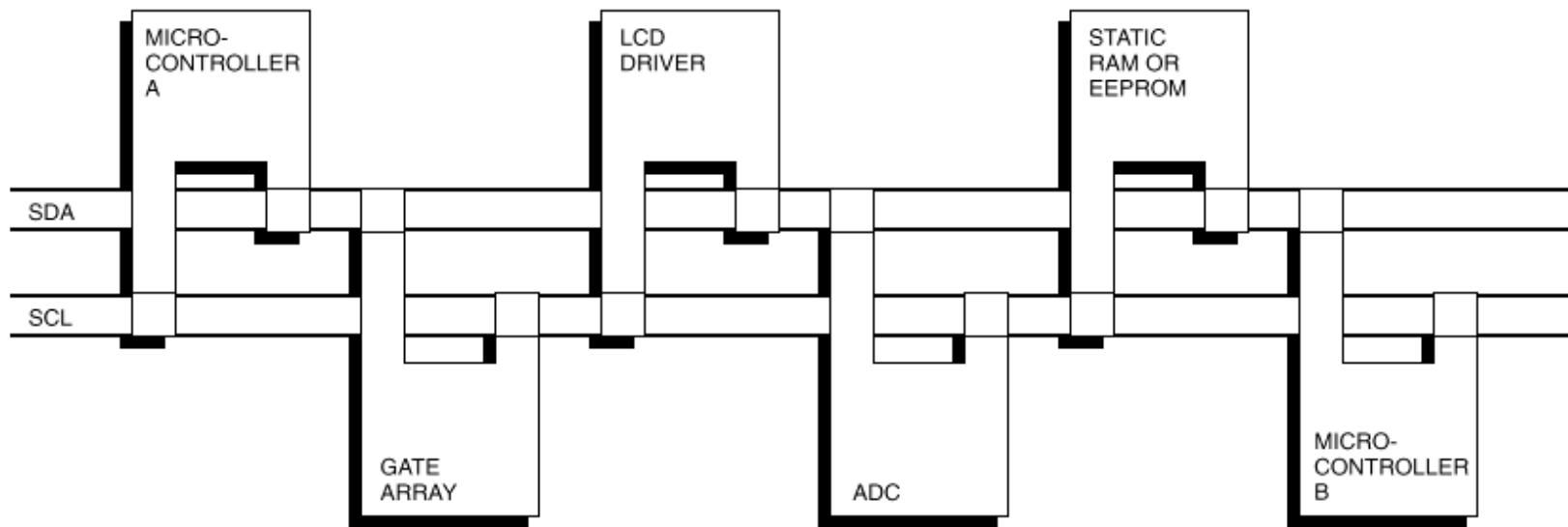
Bus I2C

- Interface síncrono con 2 líneas pensado para la comunicación entre C.I. Desarrollado por Philips en 1980.
- Líneas: reloj (SCL) y datos (SDA), ambas con resistencias de *pull-up*. Semi-dúplex. 100 Kbps, 400 Kbps y actualmente 3.4 Mbps
- Cada dispositivo tiene una dirección única codificada con 7 bits (o 10 bits de manera opcional).
- Al menos un dispositivo Maestro, que genera SCL, y uno o varios Esclavos que la reciben.

Bus I2C

- El Maestro es el único que tiene capacidad de iniciar la transferencia, decidir con quién se realiza, el sentido de la misma (envío o recepción del Maestro) y cuándo se finaliza.
- Se admite la presencia de varios Maestros en el bus (sistema multi-Maestro) con un arbitraje que asegura que en cada instante sólo hay uno que controla el bus.
- Diseño modular y compacto de sistemas con microcontroladores: se añaden o quitan dispositivos sin afectar al funcionamiento del resto del sistema.

Bus I2C: conexión al bus

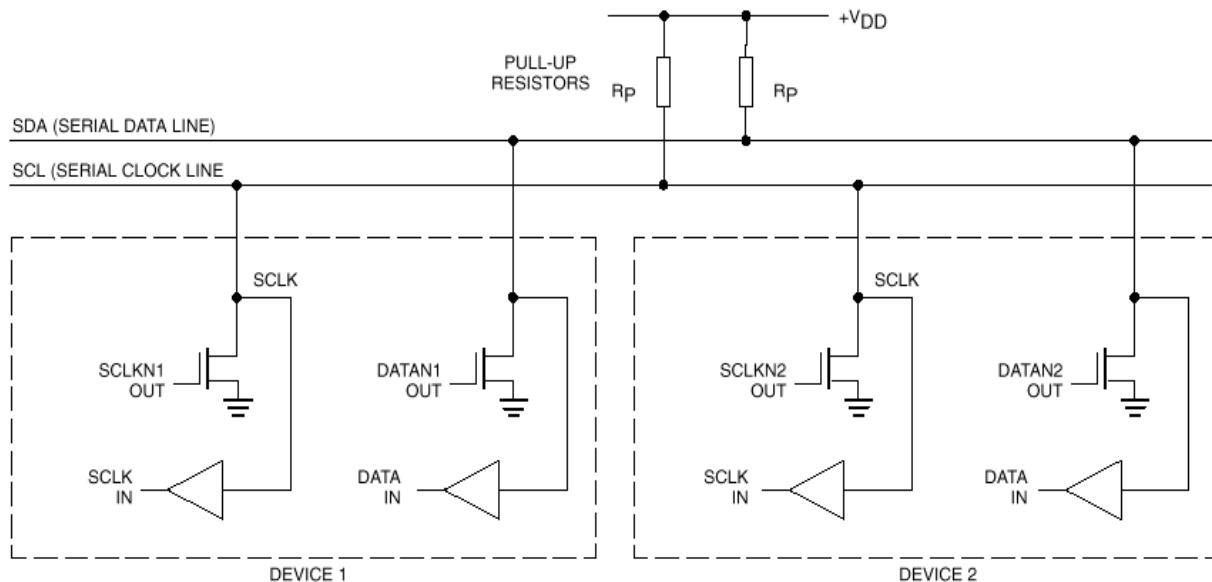


Terminología relacionada:

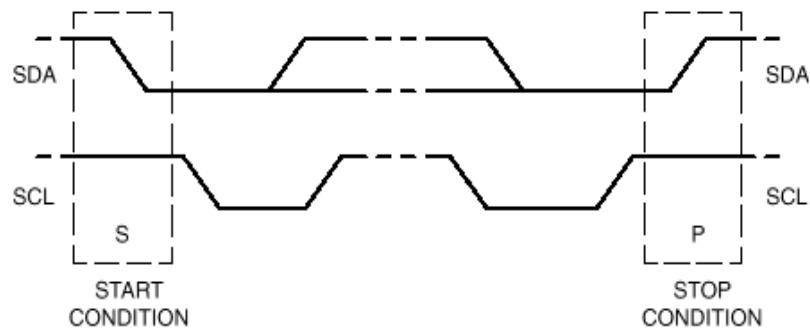
TERM	DESCRIPTION
Transmitter	The device which sends the data to the bus
Receiver	The device which receives the data from the bus
Master	The device which initiates a transfer, generates clock signals and terminates a transfer
Slave	The device addressed by a master
Multi-master	More than one master can attempt to control the bus at the same time without corrupting the message
Arbitration	Procedure to ensure that, if more than one master simultaneously tries to control the bus, only one is allowed to do so and the message is not corrupted
Synchronization	Procedure to synchronize the clock signals of two or more devices

Bus I2C: conexión al bus

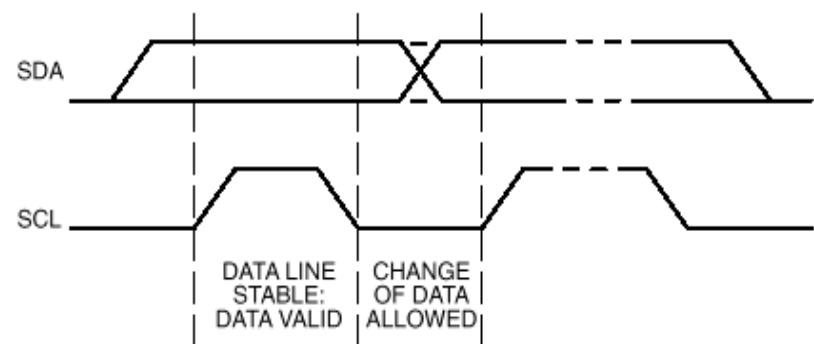
Dispositivos con salidas en colector abierto o en drenador abierto. Bus libre (nivel H) cuando todas las salidas “desconectadas” (NOR cableada).



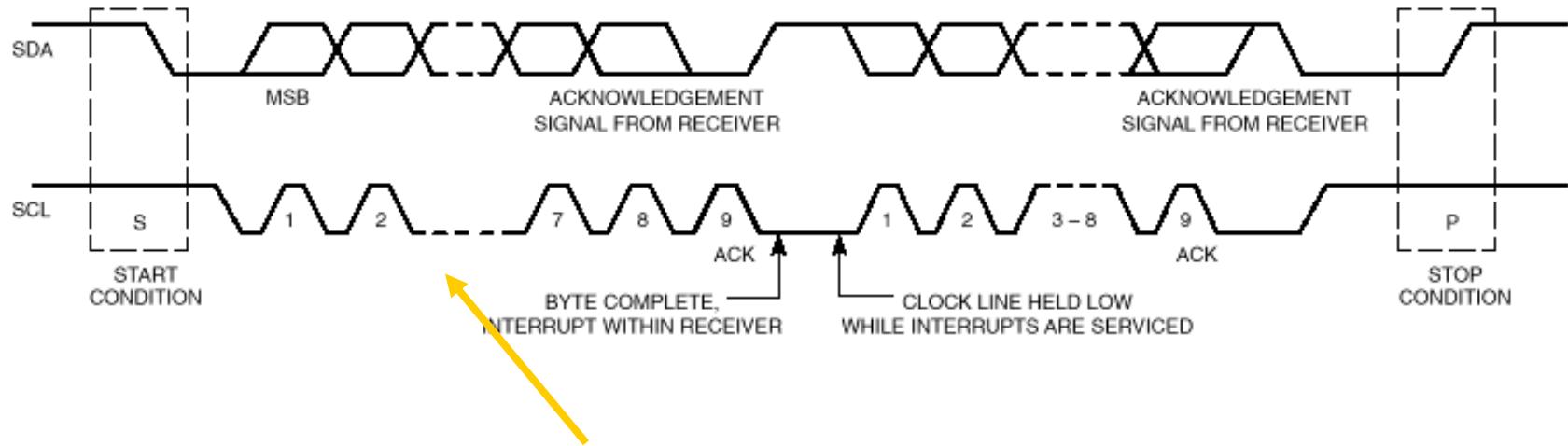
Condiciones de inicio y de parada:



Transferencia de bits en el bus:

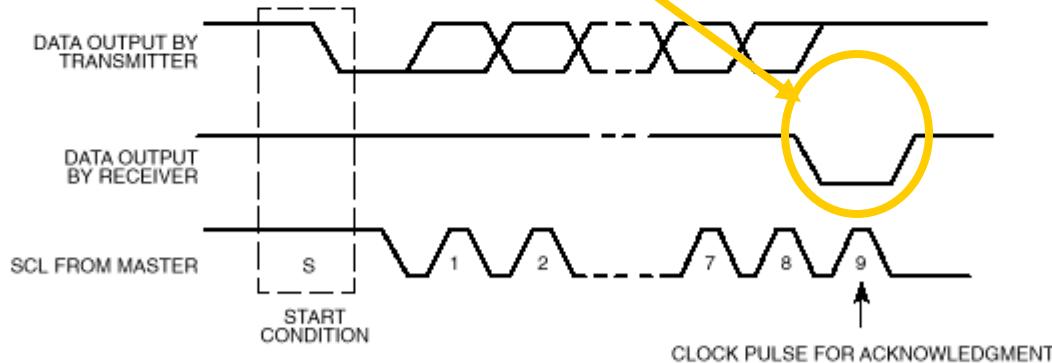


Bus I2C: transferencia de bytes



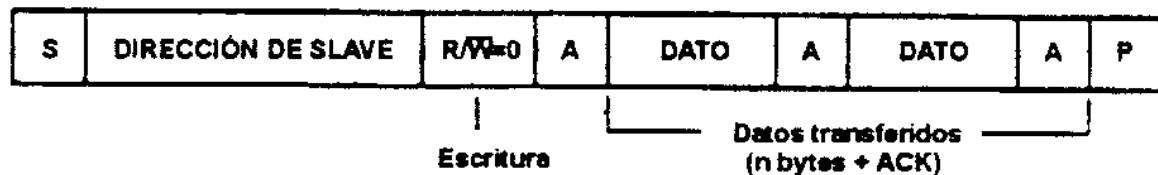
En el primer byte el maestro direcciona al esclavo (direcciones de 7 bits, + 1 bit de R/W). También puede ser una dirección de 10 bits (se da en los 2 primeros bytes).

El receptor debe reconocer (**ACKnowledge**) la recepción de cada byte.

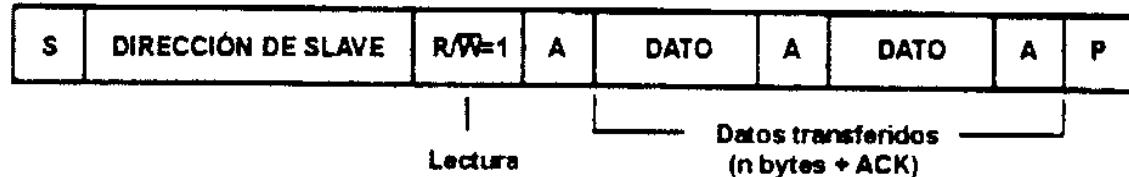


Bus I2C: formatos de trama

- a) El MASTER transmite al SLAVE receptor. No cambia el byte de dirección.



- b) El MASTER recibe desde el SLAVE después de enviarle a éste la dirección



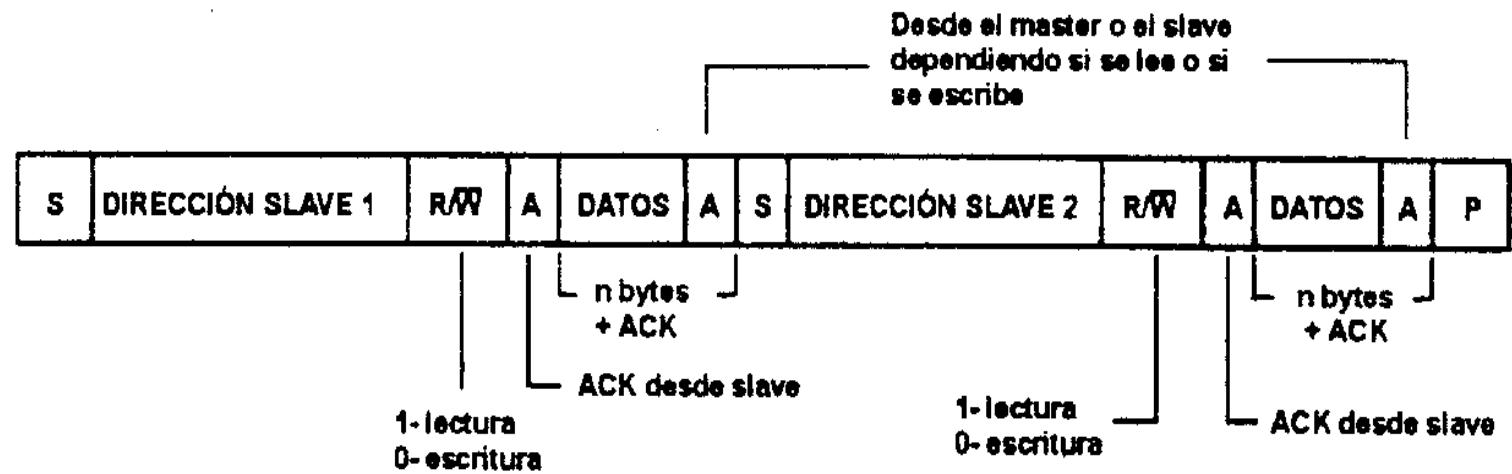
S: secuencia de inicio

P: secuencia de parada

A: ACK

Bus I2C: formatos de trama

- c) El MASTER cambia el byte de dirección y selecciona un SLAVE distinto.



S: secuencia de inicio

P: secuencia de parada

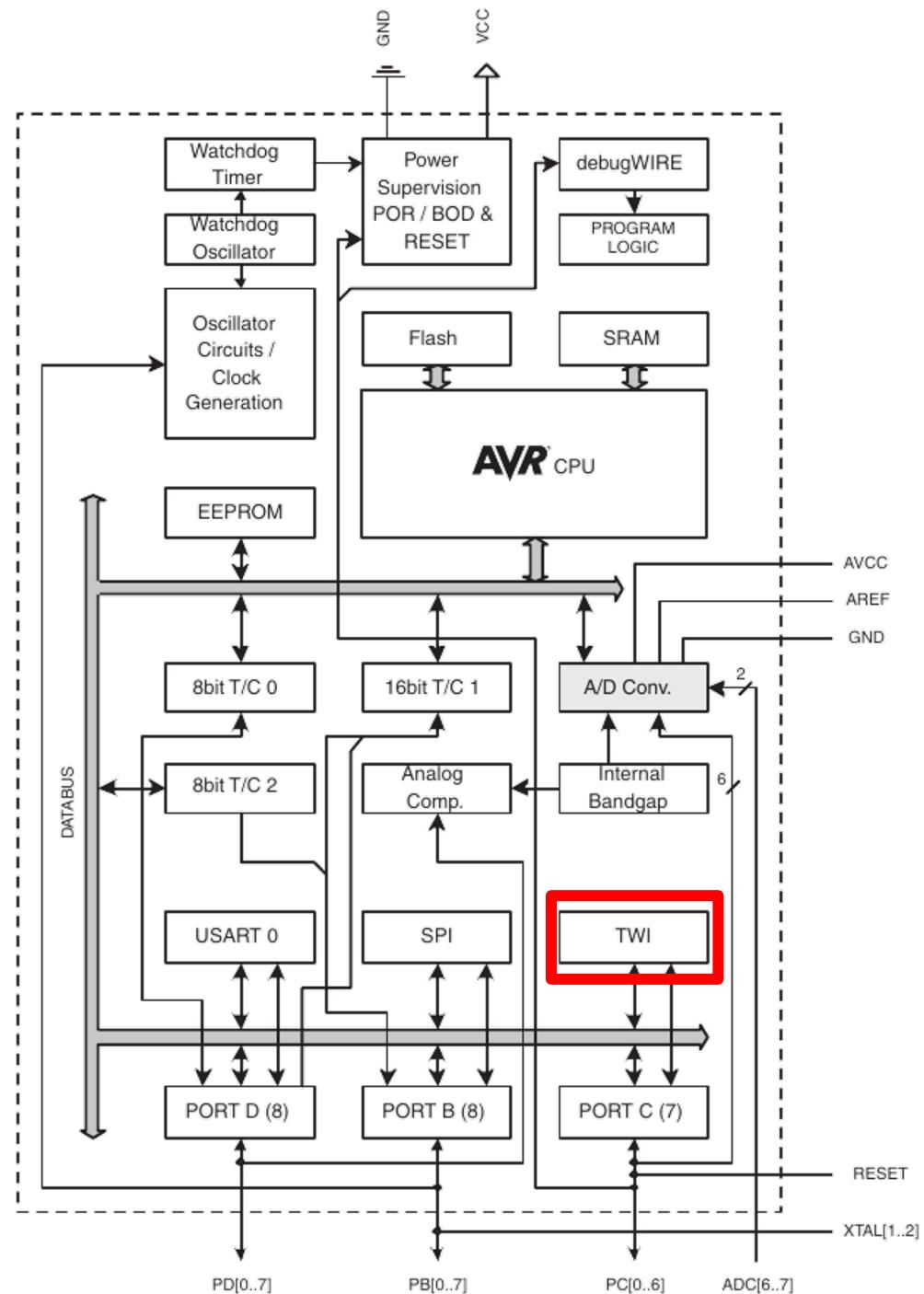
A: ACK

Circuitos típicos I2C

- Extensor de puertos: PCF8574 (8bits), MCP23017 (16 bits)
- Reloj RTC: DS1301, DS3231 (precisión)
- Memorias EEPROM: 24C04, 24C256(32 Kbytes),
- Conversor A/D 16 bits: ADS1115
- Conversor D/A 12 bits: MCP4725
- Sensor temperatura: Si7021, SHT30, SHT31
- Acelerómetros y giróscopo: MPU6050, ADXL345 (sólo acel.)
- Pantalla OLED: SSD1306
- Pantalla TFT Color: MIKROE-2163
- Intensidad luz UV: VEML607
- Sensor Presión y temperatura: BME280
- Control 16 servomotores: PCA9685
- Medidor intensidad: INA219
- Sensor táctil capacitivo: CAP1188

I2C en ATmega328P

I2C \leftrightarrow Two Wire Interface



I2C con Arduino: Wire

□ Wire:

□ <https://www.arduino.cc/en/Reference/Wire>

□ #include <Wire.h>

□ Wire.begin()

Condiciones de inicio y parada desde el maestro:

□ Wire.beginTransmission() Wire.endTransmission()

□ Petición desde el maestro: Wire.requestFrom()

Escritura y lectura:

□ Wire.write()

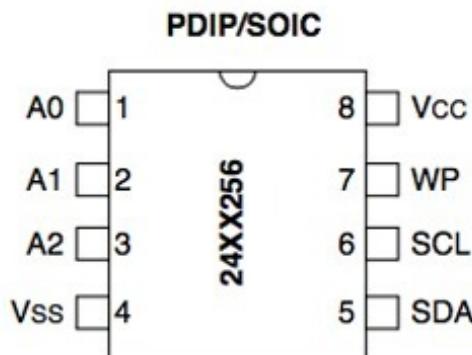
□ if (Wire.available())
 salida = Wire.read()

Interrupciones recepción y petición (en el esclavo):

□ Wire.onReceive() Wire.onRequest()

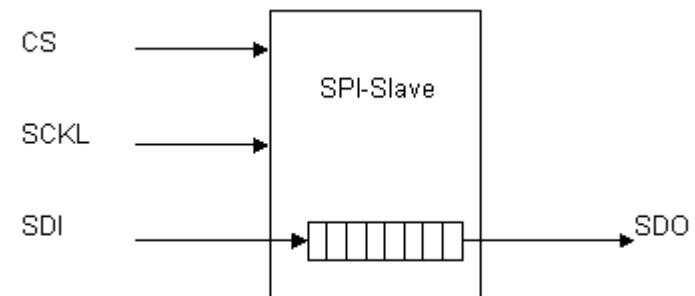
Ejemplo I2C: EEPROM 24C256

- Voltaje: 4,5V...5,5V (24C256), 2,5V...5,5V (24LC256)
 - Consumo máximo en escritura: 3mA a 5.5V
 - Consumo *standby*: 500nA a 5.0V
 - Compatible 100 Kbps y 400 Kbps
 - Circuito de protección “*power on/off*”
 - Protección de escritura hardware (pin WP)
 - 1,000,000 ciclos de borrado/escritura
 - Hasta 64 bytes en modo página
 - Tiempo de escritura típico: 5 ms (byte o página)
 - Hasta 8 dispositivos en el mismo bus
 - Retención de datos > 200 años
-
- Funciones acceso Arduino: <http://playground.arduino.cc/Code/I2CEEPROM>
`i2c_eeprom_write_byte, i2c_eeprom_read_byte, ...`



SPI: Serial Peripheral Interface

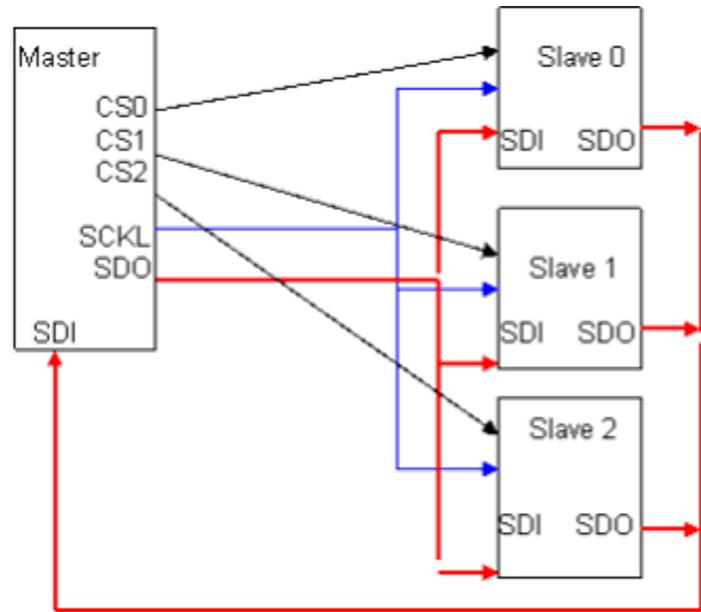
- Desarrollado por Motorola.
- Otra versión reducida: MicroWire (National Semiconductor).
- Se utiliza principalmente para comunicación vía serie entre un procesador (maestro) y los periféricos (esclavos).
- También se puede conectar 2 procesadores a través de SPI.
- Conversores, sensores, memorias, potenciómetros, ...
- SPI funciona en una configuración maestro-esclavo.
- Señales: reloj (SCLK), selección de chip (CS), Serial Data In (SDI) y Serial Data Out (SDO).
- *Full-dúplex*
- ~20 Mbps



SPI con múltiples dispositivos

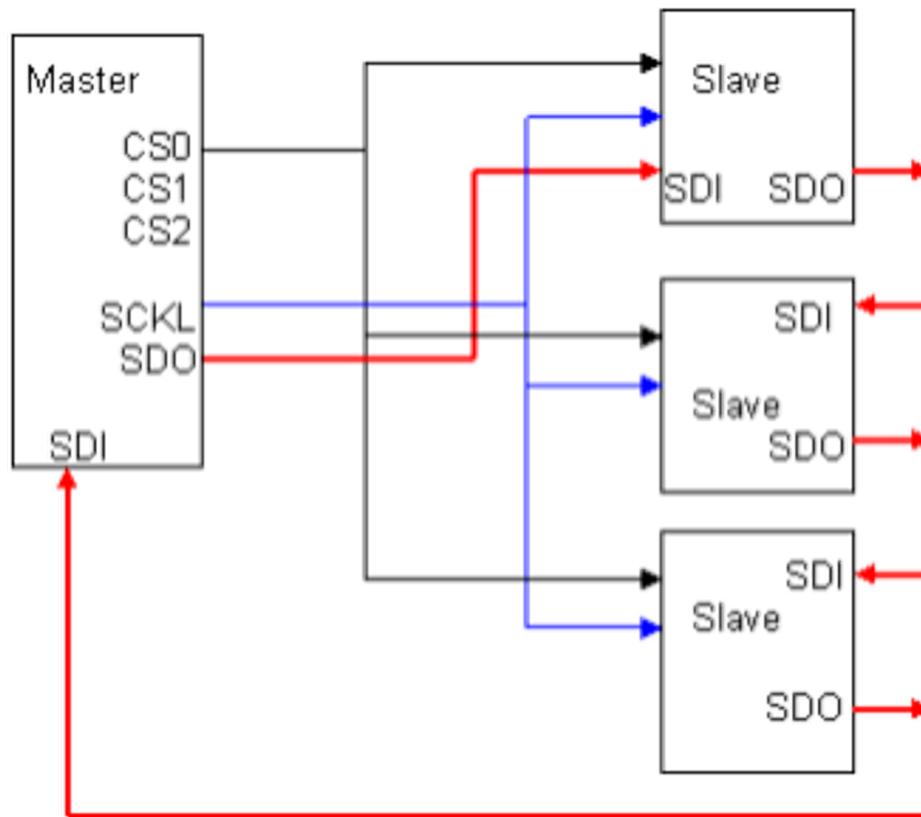
- Sólo hay un maestro, el número de esclavos depende de líneas de selección.
- Maestro envía relojes y selecciona circuitos integrados. Activa los esclavos con quien se quiere comunicar.
- En esta configuración hay 3 dispositivos esclavos. Las líneas SDO están unidas a la línea SDI del maestro.

El maestro determina el C.I. con el que quiere comunicar con las líneas CS. La salida de los esclavos que no están transmitiendo pasa a un estado de alta impedancia.

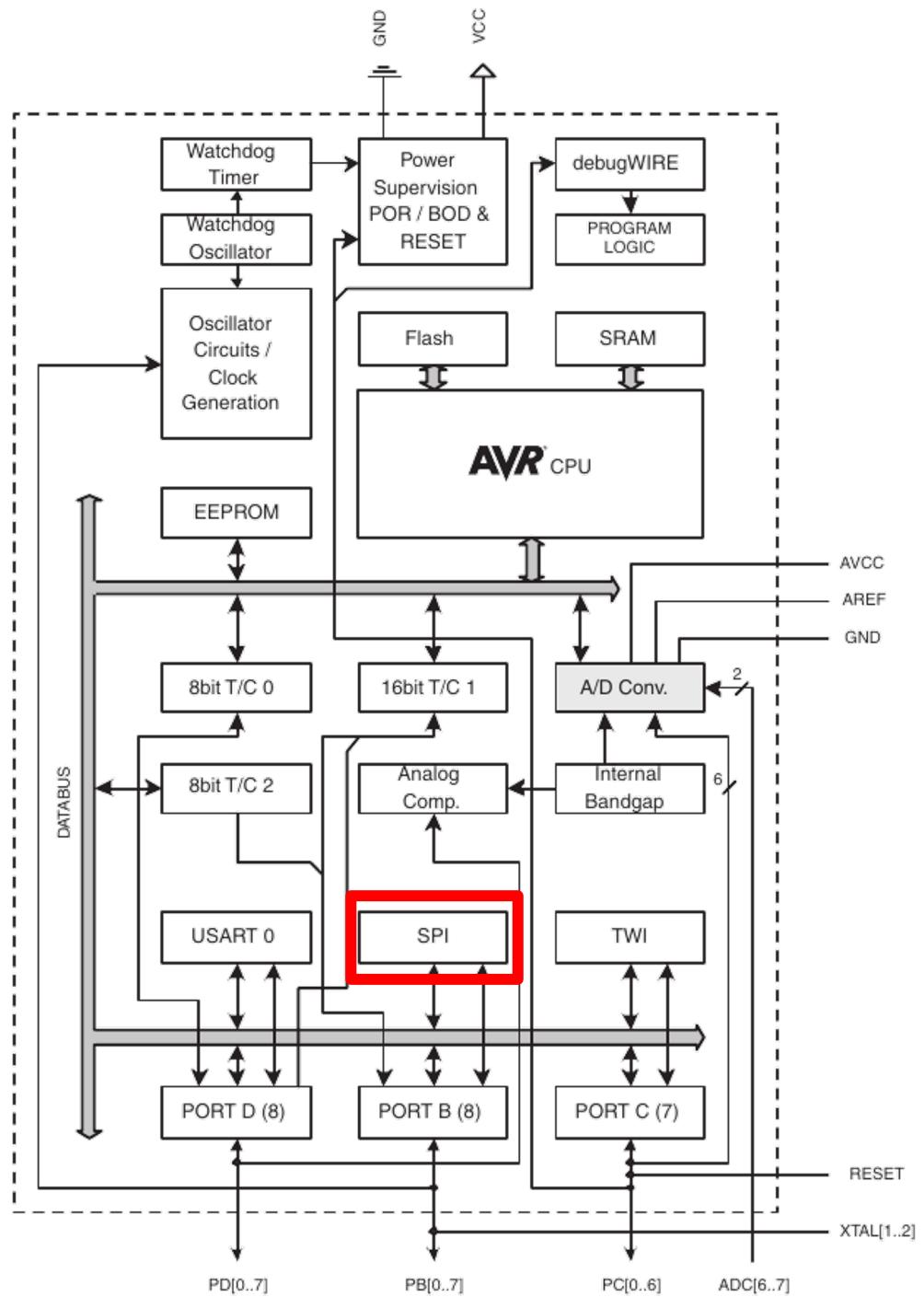


SPI: Configuración en cascada

- Cada esclavo se conecta en cascada de manera que la salida de un esclavo es la entrada de otro. El conjunto se trata como un único esclavo y se conecta a la misma CS.



SPI en ATmega328P



SPI en Arduino

□ SPI:

- <https://www.arduino.cc/en/Reference/SPI>

- #include <SPI.h>

- SPI.begin()

Tanto para lectura como escritura síncrona:

- Serial.transfer()

□ Inicializa el bus definiendo parámetros de funcionamiento:

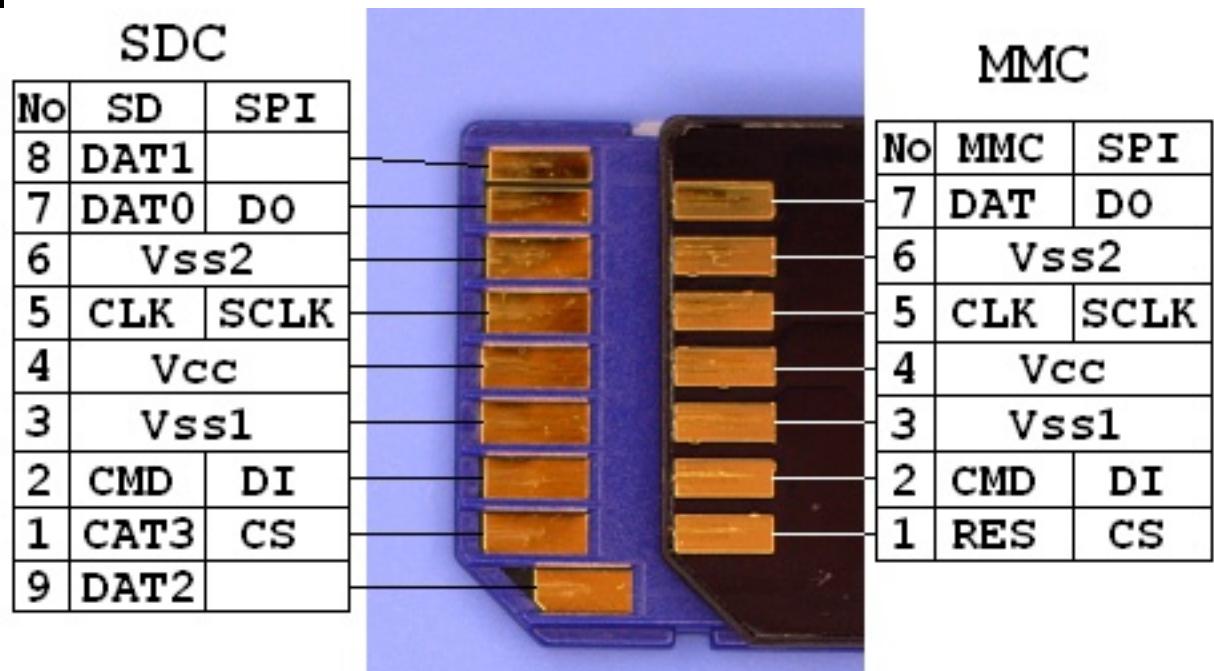
- SPI.beginTransaction(SPISettings(Fclk, MSBFIRST, SPI_MODE<X>));

...

- SPI.endTransaction();

Uso de SD/MMC mediante SPI

- Modo SD nativo (es más rápido)
- Soporta modo SPI
- Voltaje: 3V...3V3

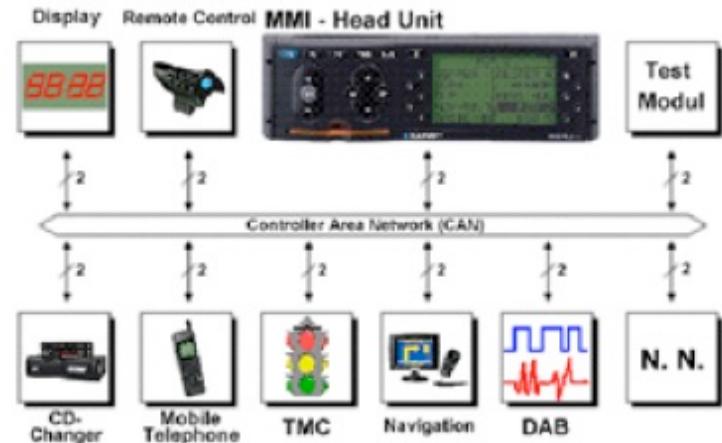
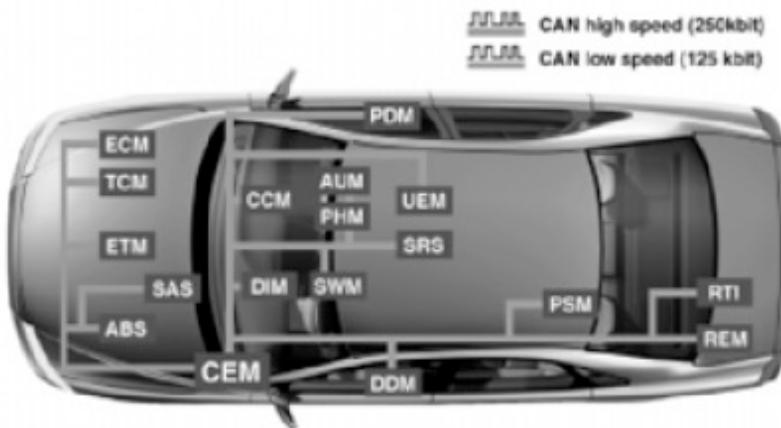


- <http://arduino.cc/en/pmwiki.php?n=Reference/SD>

Bus CAN

Historia:

- ❑ Desarrollado en los años 80 por Bosch. Originalmente para aplicaciones en automoción.
- ❑ Idea inicial: comunicar entre si varios microcontroladores:
 - ❑ control del motor
 - ❑ control de transmisión automática
 - ❑ sistemas de frenos ABS (antibloqueo de ruedas)...
- ❑ Desde 1992 se utiliza en coches, y la mayoría de las compañías de automoción europeas lo utilizan.



Bus CAN

- ❑ Flexibilidad. Se pueden añadir nodos, sin que ello perturbe el funcionamiento de los demás nodos.
- ❑ Gran fiabilidad en la transmisión:
 - ❑ Detección de errores
 - ❑ Señalización y envío de errores
 - ❑ Reenvío automático de mensajes corruptos cuando el bus está activo de nuevo
- ❑ Distinción entre errores temporales y fallos permanentes de nodos.
- ❑ Desconexión automática de nodos defectuosos.
- ❑ Puede operar en ambientes con condiciones extremas de ruido e interferencias.
- ❑ Velocidades de hasta 1Mpbs.

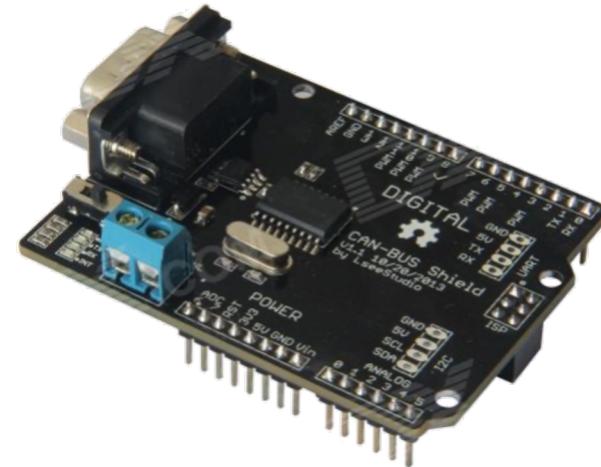
Bus CAN con Arduino

- El bus EOBD es obligatorio para todos los coches de gasolina desde 2001 y diésel desde 2004.

```
uint8_t messageAvailable(void);      // Comprueba si hay mensaje CAN
char getMessage(messageCAN *msje);   // Recibe mensaje CAN
sendMessage(messageCAN *msje);       // Envía mensaje CAN
printMessage(messageCAN *msje);     // Imprime mensaje

// Funciones directas
unsigned int getEngineLoad();
unsigned int getEngineCoolantTemp();
unsigned int getFuelPressure();
unsigned int getIntakeMAPressure();
unsigned int getEngineRPM();
unsigned int getVehicleSpeed();

...
```



Bus One Wire

- Bus basado en 1 hilo + GND. (2,8v...6v)
- Incluye alimentación, bidireccional, half-duplex.
(15,4Kb/s...125Kb/s) (Difícil > 30Kb/s)
- 1 Maestro, múltiples dispositivos
- Se identifican por dirección
- Cada dispositivo tiene un ID único de 64 bits
- Sensores temperatura (DS18B20), iButton, ...
- <http://playground.arduino.cc/Learning/OneWire>

