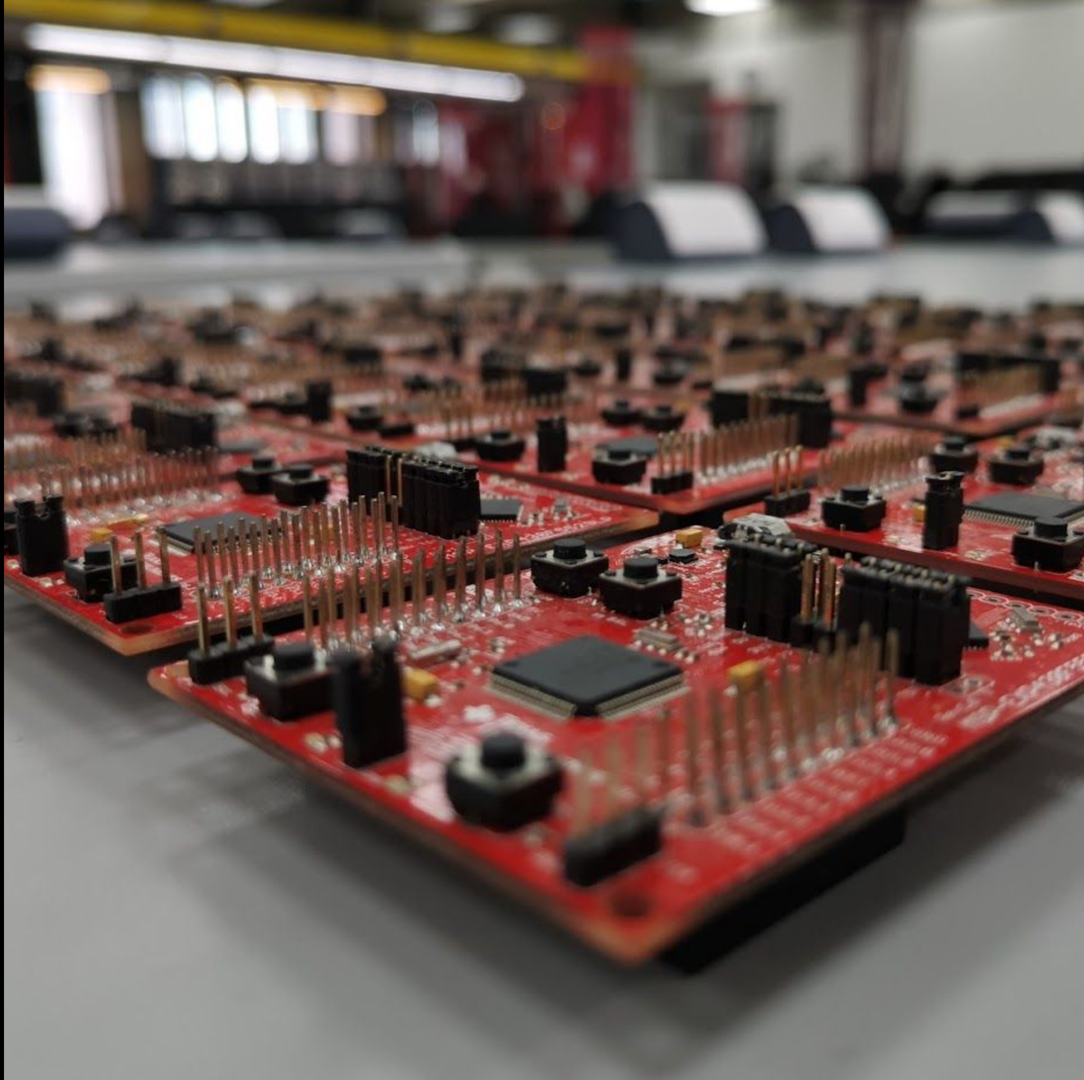
IEE2463 Sistemas Electrónicos Programables

- Felipe Sánchez Varas
- Camila Turrieta González

1er semestre 2020



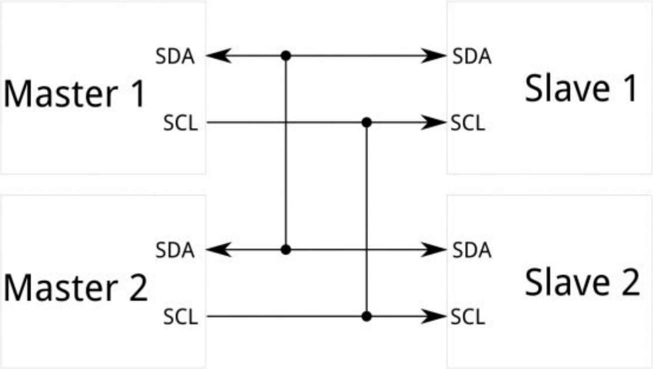
Temas

- Inter-Integrated Circuit (I^2C)
- Serial Peripherical Interface (SPI)
- Resumen protocolos de comunicación

I^2C

Características I^{2}C

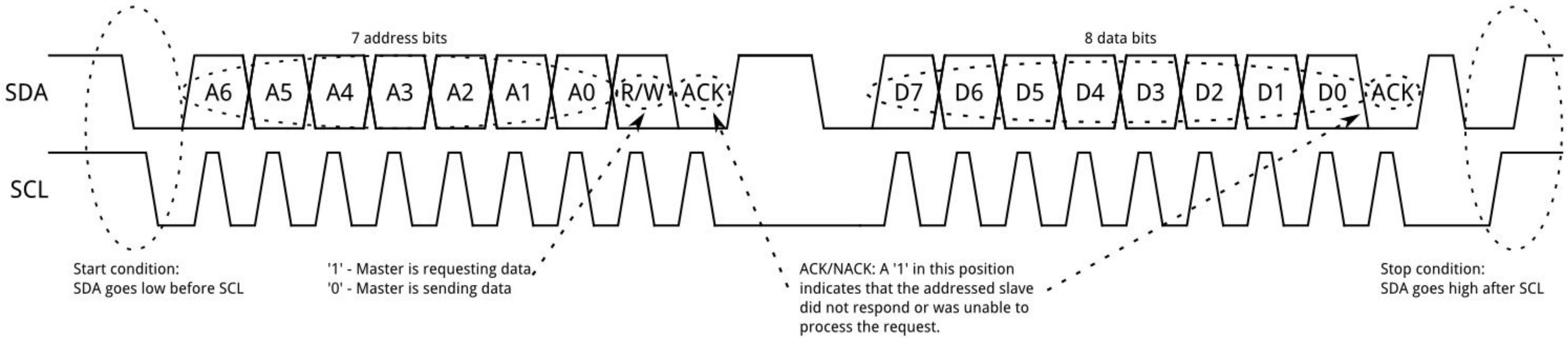
- Comunicación síncrona:
    - Necesitamos señal de clock!
- Conexión con entre múltiples dispositivos:
    - Múltiples Master
    - Múltiples Slave
- Solo requiere de 2 cables
- Originalmente creado por Philips en 1982
    - Ideado para simplificar el control de múltiples chips en televisores
- Atmel introdujo TWI por motivos de licencia
    - TWI = I^{2}C



https://learn.sparkfun.com/tutorials/i2c

Funcionamiento del protocolo

- 7 o 10 bits de direccionamiento
  - En teoría son 128 direcciones
  -
- 8 data bits



Start condition:
SDA goes low before SCL

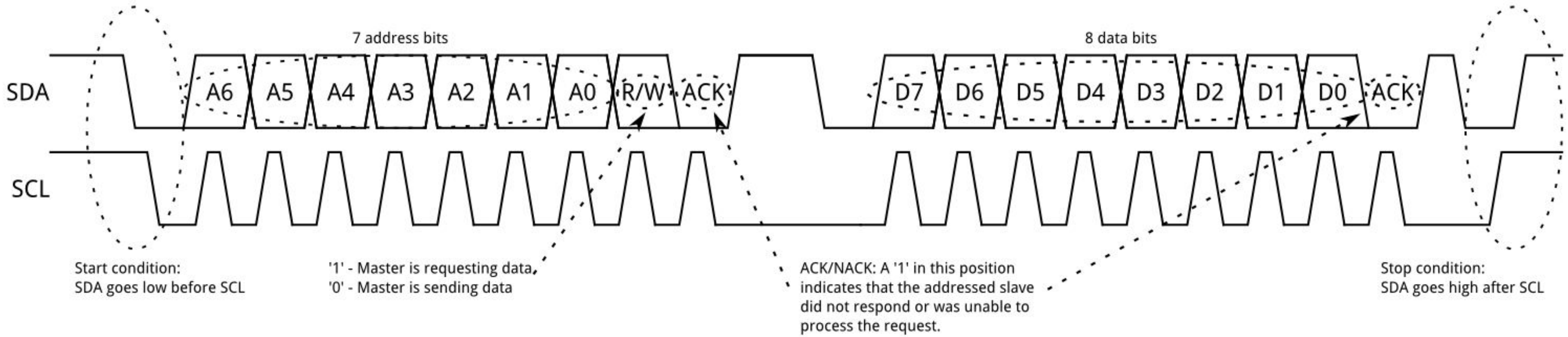
'1' - Master is requesting data,
'0' - Master is sending data

ACK/NACK: A '1' in this position indicates that the addressed slave did not respond or was unable to process the request.

Stop condition:
SDA goes high after SCL

Funcionamiento del protocolo

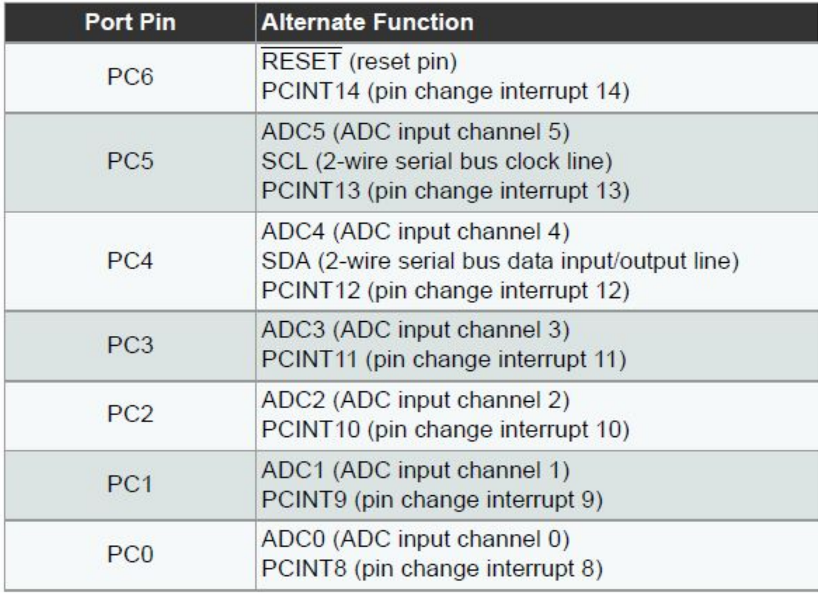
- ¿Qué configurar?
  - Bit Rate
  - Configuración como Master / Slave
  - ¿Cómo coordino varios slave?



Ejemplos de programación

Registros ATmega328P

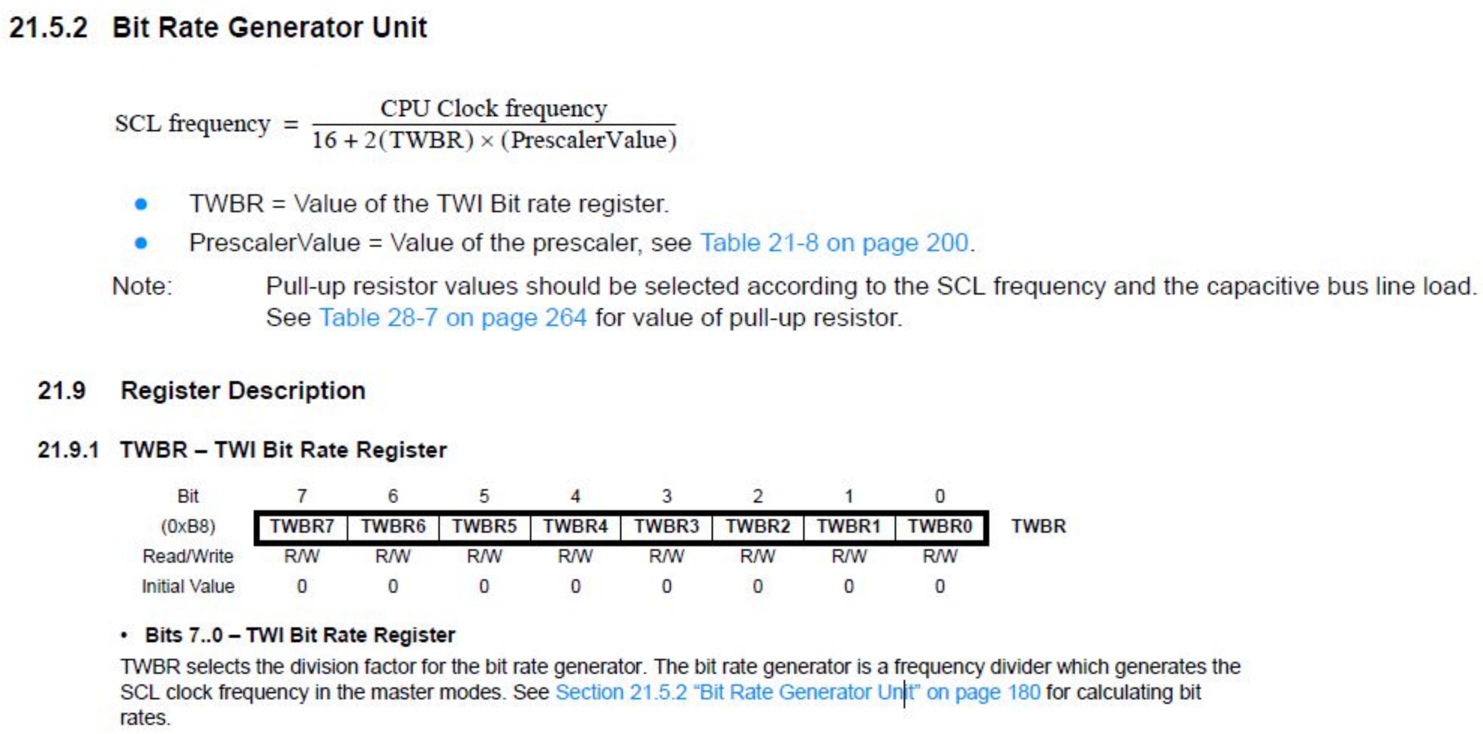
Table 13-6. Port C Pins Alternate Functions



Port Pin	Alternate Function
PC6	$\overline{\text{RESET}}$ (reset pin) PCINT14 (pin change interrupt 14)
PC5	ADC5 (ADC input channel 5) SCL (2-wire serial bus clock line) PCINT13 (pin change interrupt 13)
PC4	ADC4 (ADC input channel 4) SDA (2-wire serial bus data input/output line) PCINT12 (pin change interrupt 12)
PC3	ADC3 (ADC input channel 3) PCINT11 (pin change interrupt 11)
PC2	ADC2 (ADC input channel 2) PCINT10 (pin change interrupt 10)
PC1	ADC1 (ADC input channel 1) PCINT9 (pin change interrupt 9)
PC0	ADC0 (ADC input channel 0) PCINT8 (pin change interrupt 8)

- Pines
  - SCL - PC5
    - Clock
  - SDA - PC4
    - Data input/output

Registros ATmega328P



21.5.2 Bit Rate Generator Unit

$$\text{SCL frequency} = \frac{\text{CPU Clock frequency}}{16 + 2(\text{TWBR}) \times (\text{PrescalerValue})}$$

- TWBR = Value of the TWI Bit rate register.
- PrescalerValue = Value of the prescaler, see Table 21-8 on page 200.

Note: Pull-up resistor values should be selected according to the SCL frequency and the capacitive bus line load. See Table 28-7 on page 264 for value of pull-up resistor.

21.9 Register Description

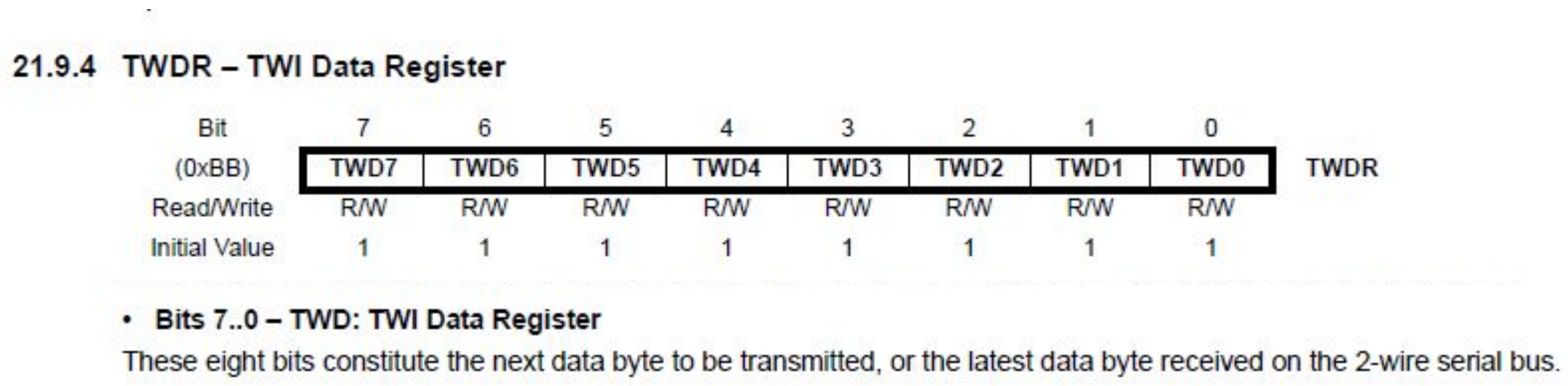
21.9.1 TWBR – TWI Bit Rate Register

Bit	7	6	5	4	3	2	1	0	
(0xB8)	TWBR7	TWBR6	TWBR5	TWBR4	TWBR3	TWBR2	TWBR1	TWBR0	TWBR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7..0 – TWI Bit Rate Register**

TWBR selects the division factor for the bit rate generator. The bit rate generator is a frequency divider which generates the SCL clock frequency in the master modes. See Section 21.5.2 "Bit Rate Generator Unit" on page 180 for calculating bit rates.

Registros ATmega328P



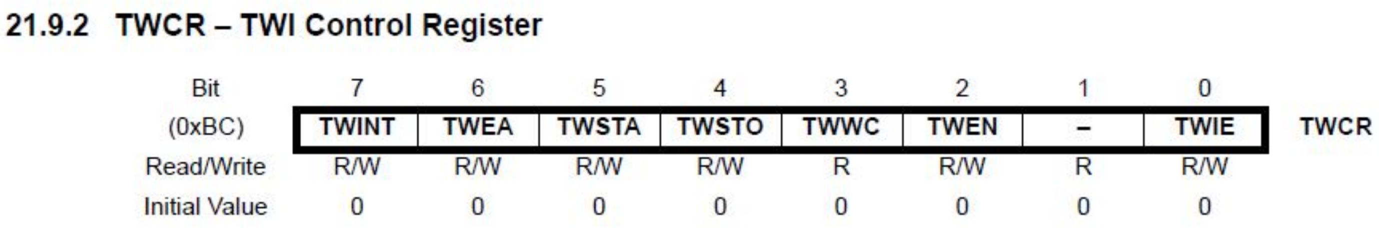
21.9.4 TWDR – TWI Data Register

Bit	7	6	5	4	3	2	1	0	
(0xBB)	TWD7	TWD6	TWD5	TWD4	TWD3	TWD2	TWD1	TWD0	TWDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	1	1	1	1	1	1	1	

- **Bits 7..0 – TWD: TWI Data Register**

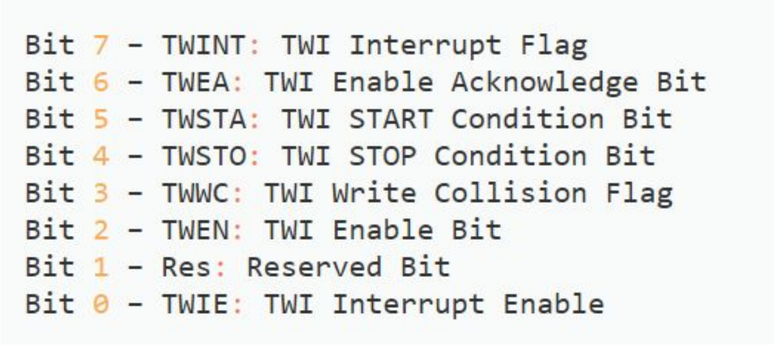
These eight bits constitute the next data byte to be transmitted, or the latest data byte received on the 2-wire serial bus.

Registros ATmega328P



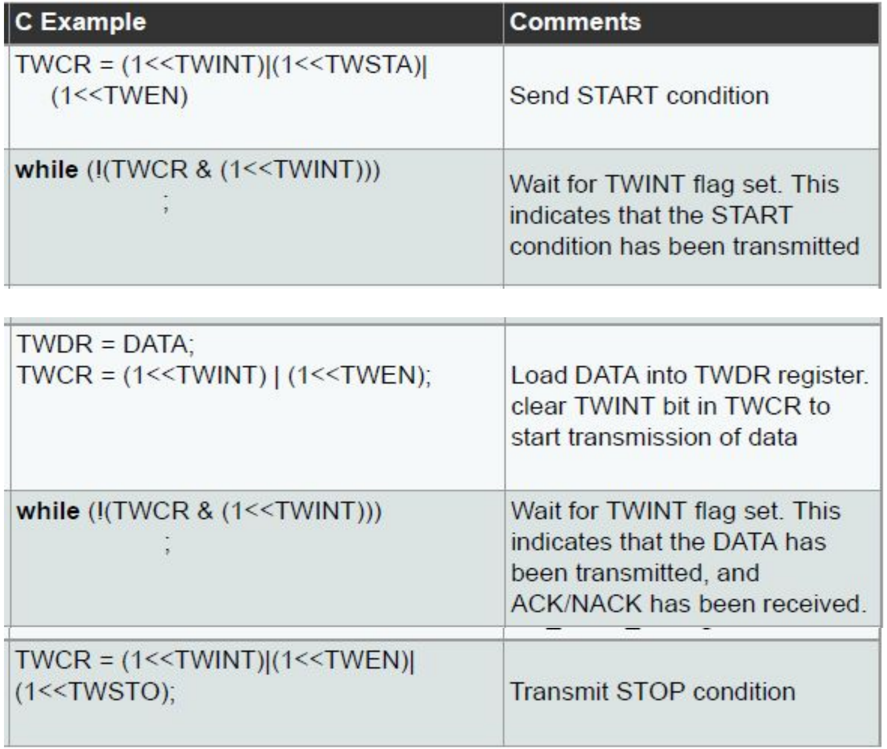
21.9.2 TWCR – TWI Control Register

Bit	7	6	5	4	3	2	1	0	
(0xBC)	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE	TWCR
Read/Write	R/W	R/W	R/W	R/W	R	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	



```
Bit 7 - TWINT: TWI Interrupt Flag
Bit 6 - TWEA: TWI Enable Acknowledge Bit
Bit 5 - TWSTA: TWI START Condition Bit
Bit 4 - TWSTO: TWI STOP Condition Bit
Bit 3 - TWWC: TWI Write Collision Flag
Bit 2 - TWEN: TWI Enable Bit
Bit 1 - Res: Reserved Bit
Bit 0 - TWIE: TWI Interrupt Enable
```

Registros ATmega328P (ejemplos)



C Example	Comments
TWCR = (1<<TWINT)|(1<<TWSTA)| (1<<TWEN)	Send START condition
while (!(TWCR & (1<<TWINT))) ;	Wait for TWINT flag set. This indicates that the START condition has been transmitted

TWDR = DATA;
TWCR = (1<<TWINT) | (1<<TWEN);
	Load DATA into TWDR register. clear TWINT bit in TWCR to start transmission of data

while (!(TWCR & (1<<TWINT))) ;	Wait for TWINT flag set. This indicates that the DATA has been transmitted, and ACK/NACK has been received.

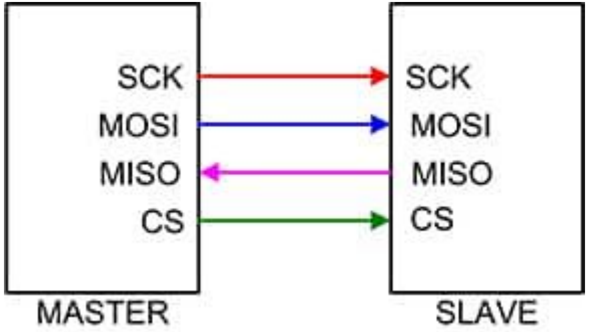
TWCR = (1<<TWINT)|(1<<TWEN)|
(1<<TWSTO);
	Transmit STOP condition

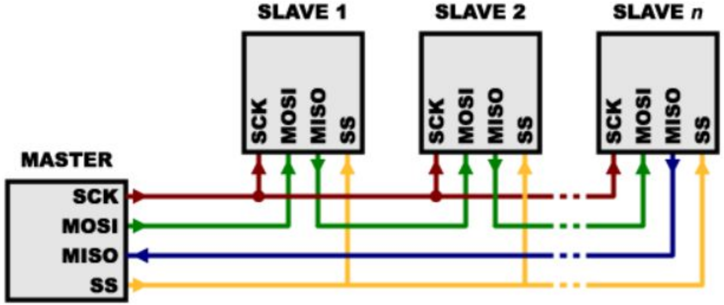
Ejemplos de programación

Serial Peripherical Interface

Características SPI

- Comunicación síncrona:
  - Necesitamos señal de clock!
- Full duplex
- Conexión con entre múltiples dispositivos:
  - 1 Master
  - Múltiples Slave

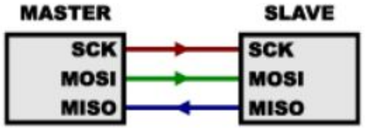


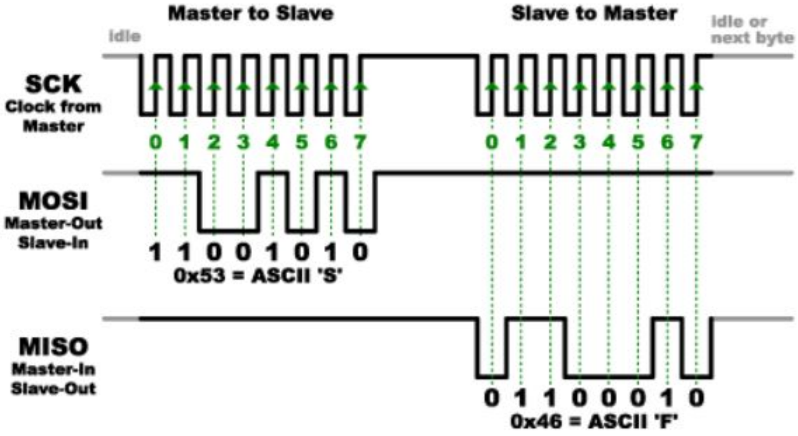


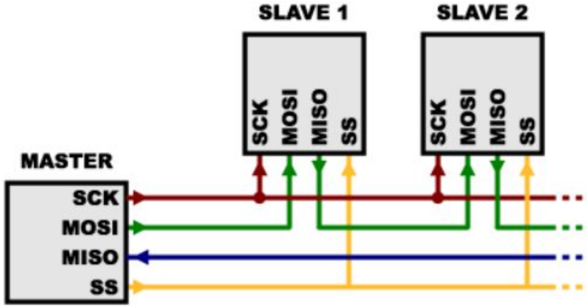
https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi

Funcionamiento del protocolo

- ¿Qué configurar en un uC?
  - Pin SCK, MOSI, MISO
  - Frecuencia del clock
  - ¿Master o slave?
  - ¿Cómo coordino varios slave?





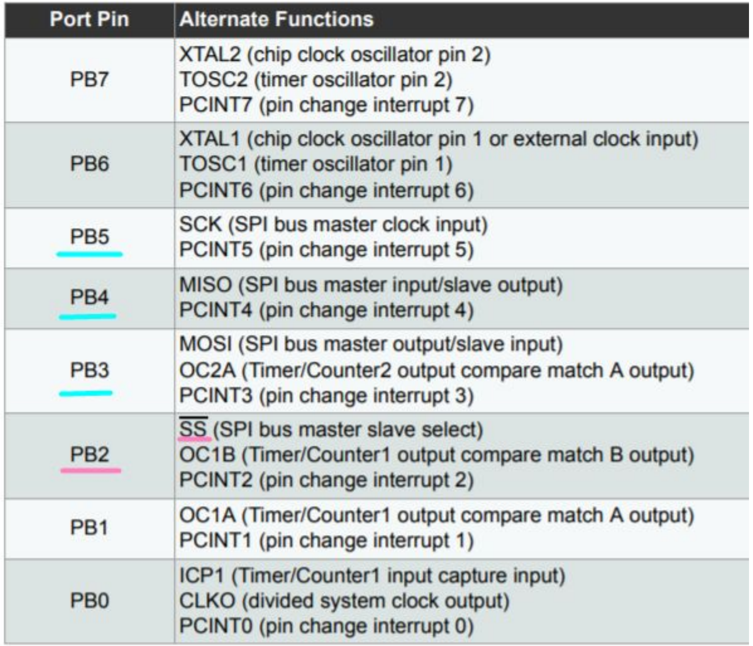


https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi

Ejemplos de programación

Registros ATmega328P

Table 13-3. Port B Pins Alternate Functions



Port Pin	Alternate Functions
PB7	XTAL2 (chip clock oscillator pin 2) TOSC2 (timer oscillator pin 2) PCINT7 (pin change interrupt 7)
PB6	XTAL1 (chip clock oscillator pin 1 or external clock input) TOSC1 (timer oscillator pin 1) PCINT6 (pin change interrupt 6)
PB5	SCK (SPI bus master clock input) PCINT5 (pin change interrupt 5)
PB4	MISO (SPI bus master input/slave output) PCINT4 (pin change interrupt 4)
PB3	MOSI (SPI bus master output/slave input) OC2A (Timer/Counter2 output compare match A output) PCINT3 (pin change interrupt 3)
PB2	$\overline{SS}$ (SPI bus master slave select) OC1B (Timer/Counter1 output compare match B output) PCINT2 (pin change interrupt 2)
PB1	OC1A (Timer/Counter1 output compare match A output) PCINT1 (pin change interrupt 1)
PB0	ICP1 (Timer/Counter1 input capture input) CLKO (divided system clock output) PCINT0 (pin change interrupt 0)

- Pines
  - SCK - PB5
    - Clock
  - MISO - PB4
    - Master input slave output
  - MOSI - PB3
    - Master output slave input
  - SS - PB2
    - Solo si uC como slave
  - Configurar pines para SS
    - GPIO

Registros ATmega328P

- Pines
  - SCK - PB5
    - Clock
  - MISO - PB4
    - Master **input** slave output
  - MOSI - PB3
    - Master **output** slave input
  - Configurar pines para SS
    - GPIO
      - 0: slave seleccionado.
      - 1: no seleccionado.

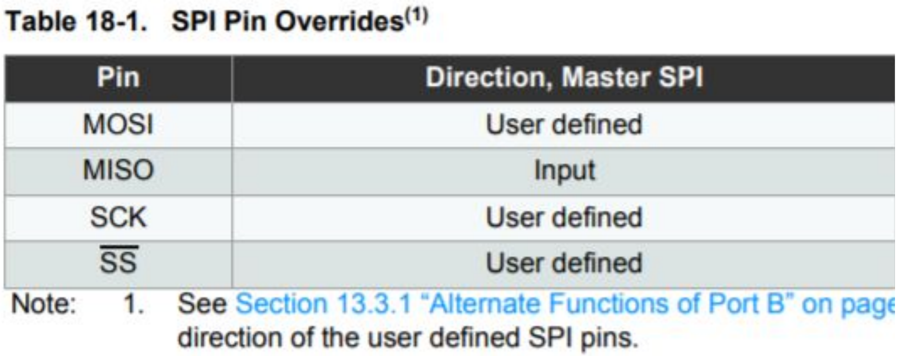
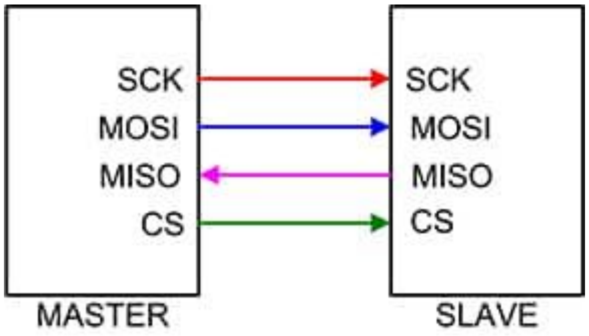


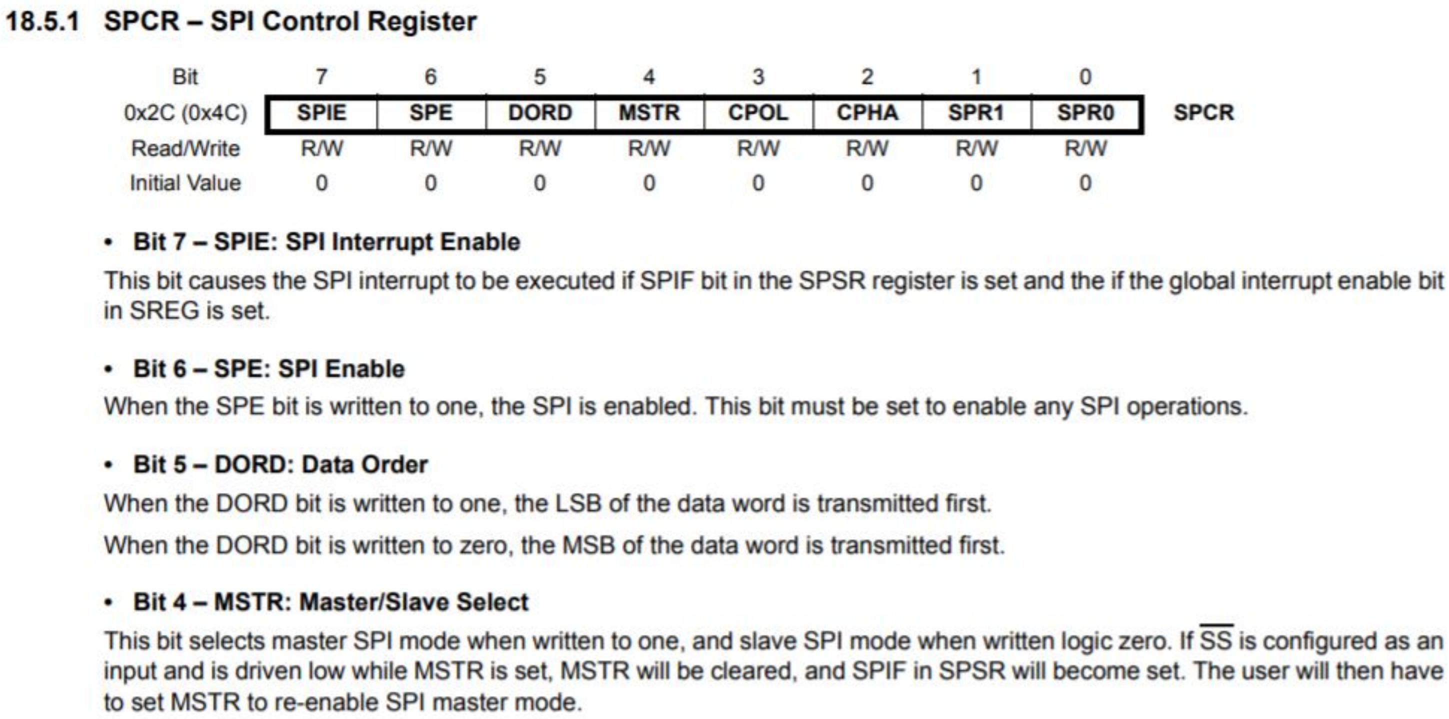
Table 18-1. SPI Pin Overrides[1]

Pin	Direction, Master SPI
MOSI	User defined
MISO	Input
SCK	User defined
$\overline{SS}$	User defined

Note: 1. See Section 13.3.1 "Alternate Functions of Port B" on page
direction of the user defined SPI pins.



Registros ATmega328P



18.5.1 SPCR – SPI Control Register

Bit	7	6	5	4	3	2	1	0	
0x2C (0x4C)	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – SPIE: SPI Interrupt Enable**

This bit causes the SPI interrupt to be executed if SPIF bit in the SPSR register is set and the if the global interrupt enable bit in SREG is set.

- **Bit 6 – SPE: SPI Enable**

When the SPE bit is written to one, the SPI is enabled. This bit must be set to enable any SPI operations.

- **Bit 5 – DORD: Data Order**

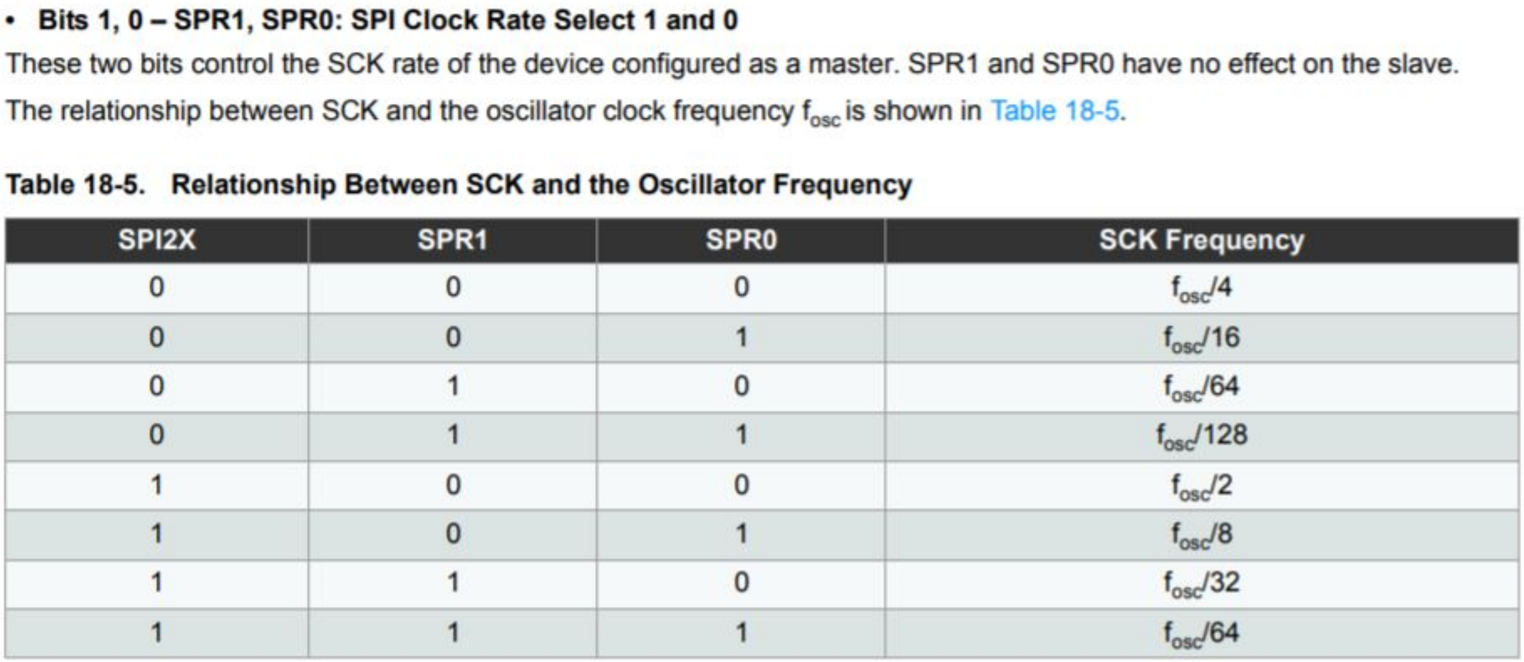
When the DORD bit is written to one, the LSB of the data word is transmitted first.

When the DORD bit is written to zero, the MSB of the data word is transmitted first.

- **Bit 4 – MSTR: Master/Slave Select**

This bit selects master SPI mode when written to one, and slave SPI mode when written logic zero. If $\overline{\text{SS}}$ is configured as an input and is driven low while MSTR is set, MSTR will be cleared, and SPIF in SPSR will become set. The user will then have to set MSTR to re-enable SPI master mode.

Registros ATmega328P



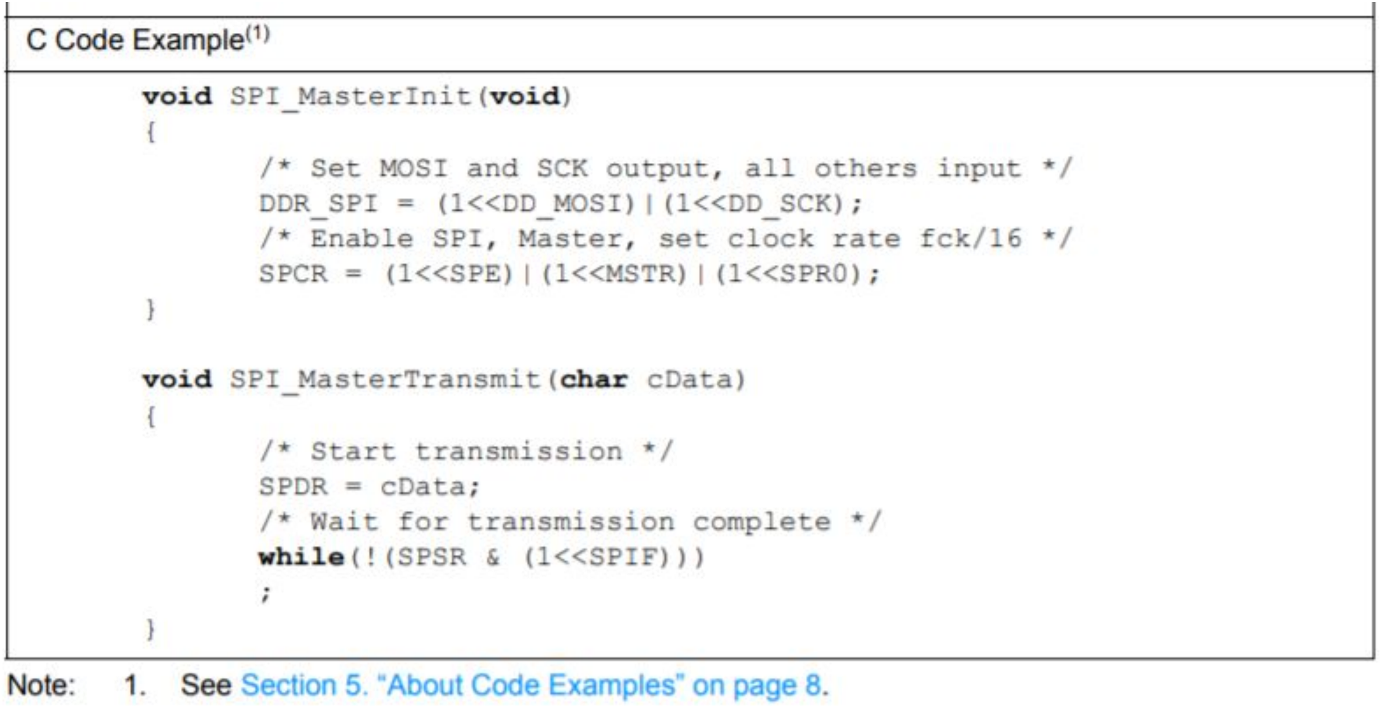
- **Bits 1, 0 – SPR1, SPR0: SPI Clock Rate Select 1 and 0**

These two bits control the SCK rate of the device configured as a master. SPR1 and SPR0 have no effect on the slave. The relationship between SCK and the oscillator clock frequency f_{osc} is shown in Table 18-5.

Table 18-5. Relationship Between SCK and the Oscillator Frequency

SPI2X	SPR1	SPR0	SCK Frequency
0	0	0	$f_{osc}/4$
0	0	1	$f_{osc}/16$
0	1	0	$f_{osc}/64$
0	1	1	$f_{osc}/128$
1	0	0	$f_{osc}/2$
1	0	1	$f_{osc}/8$
1	1	0	$f_{osc}/32$
1	1	1	$f_{osc}/64$

Registros ATmega328P



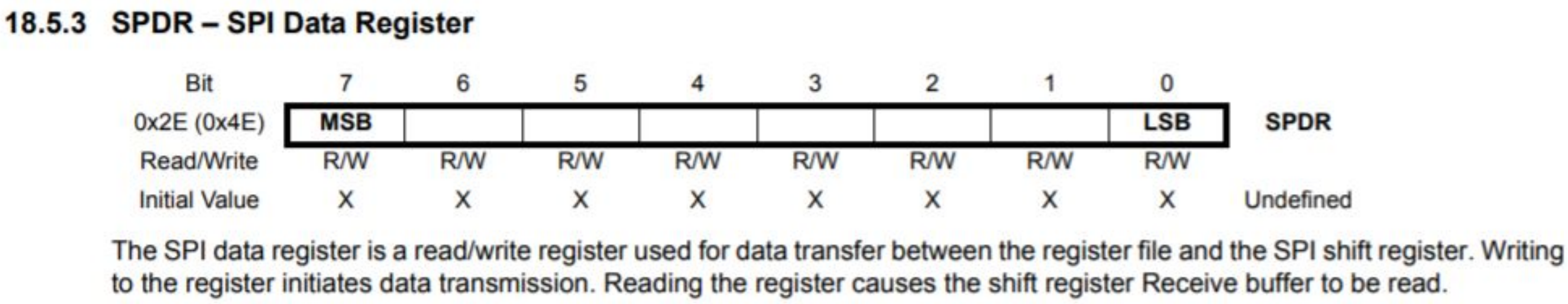
C Code Example[1]

```
void SPI_MasterInit(void)
{
        /* Set MOSI and SCK output, all others input */
        DDR_SPI = (1<<DD_MOSI)|(1<<DD_SCK);
        /* Enable SPI, Master, set clock rate fck/16 */
        SPCR = (1<<SPE)|(1<<MSTR)|(1<<SPR0);
}

void SPI_MasterTransmit(char cData)
{
        /* Start transmission */
        SPDR = cData;
        /* Wait for transmission complete */
        while(!(SPSR & (1<<SPIF)))
        ;
}
```

Note: 1. See Section 5. "About Code Examples" on page 8.

Registros ATmega328P



18.5.3 SPDR – SPI Data Register

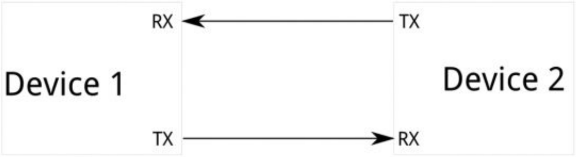
Bit	7	6	5	4	3	2	1	0	
0x2E (0x4E)	**MSB**							**LSB**	**SPDR**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	X	X	X	X	X	X	X	X	Undefined

The SPI data register is a read/write register used for data transfer between the register file and the SPI shift register. Writing to the register initiates data transmission. Reading the register causes the shift register Receive buffer to be read.

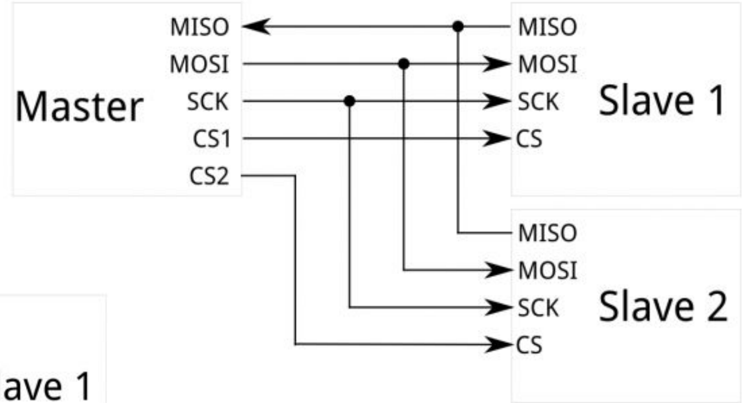
Ejemplos de programación

Protocolos de comunicación (resumen)

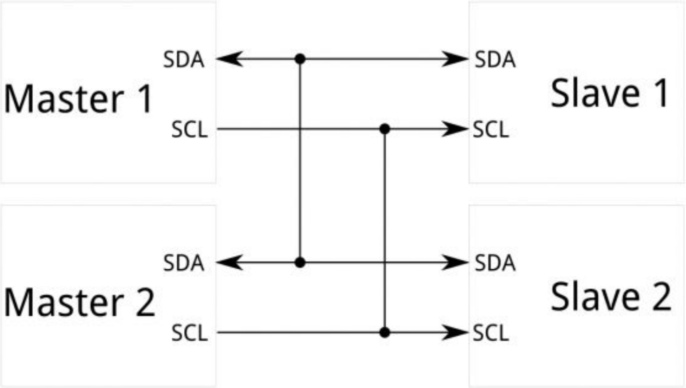
Establecimiento de un estándar



UART



SPI



I2C

¿Hay alguno mejor?

	UART	SPI	I^{2}C
Velocidad (max)	Entre 230 kbps a 460 kbps	Normalmente 10 Mbps a 20 Mbps	High Speed Mode 3.4 Mbps
Master	No	1	Varios
Conexiones mín	2	4	2
Ventajas	Muy simple	Full Duplex, alta velocidad de datos	Sencillo
Desventajas	Solo 2 dispositivos	Conexiones aumentan con número de esclavos	Circuito aumenta su complejidad

Estructura de un paquete de datos