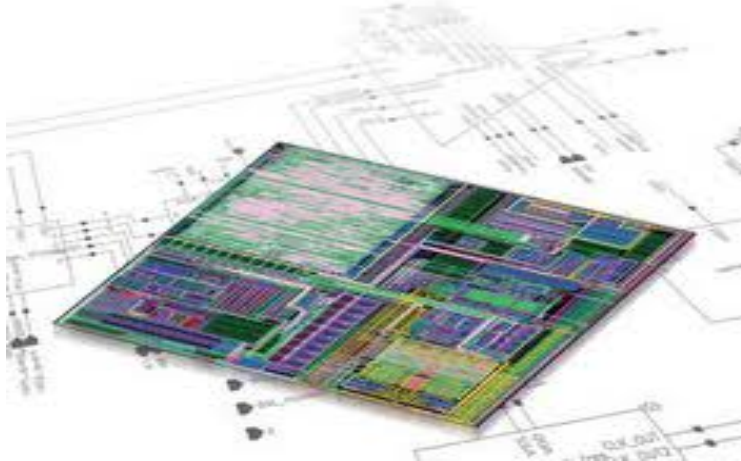
Comunicaciones Digitales: Protocolos seriales (uC)



¿Qué es la comunicación serial?

- La comunicación serial es un protocolo de comunicación entre dispositivos que se incluye de manera estándar en prácticamente cualquier computadora.

- La mayoría de las computadoras incluyen puertos seriales. Actualmente puertos USB, aunque aún se encuentran algunas con puerto serial RS-232.

- La comunicación serial RS232 es un protocolo común utilizado por dispositivos y equipos usados en instrumentación. La comunicación serial puede ser utilizada para adquisición de datos, control, depuración de código, etc.

¿Qué es la comunicación serial?

- El concepto de comunicación serial permite la transmisión-recepción bit a bit de un byte completo, este método de comunicación puede alcanzar mayores distancias.

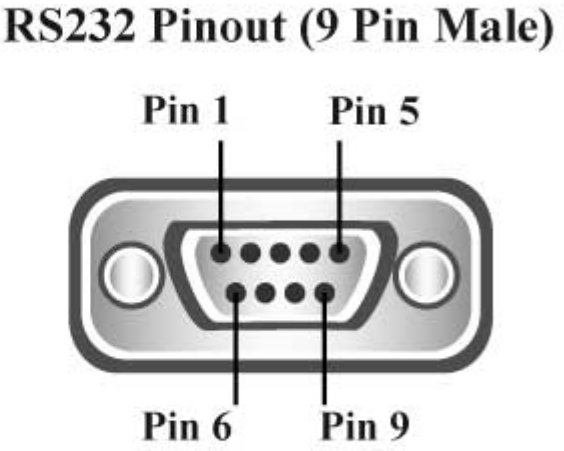
- Por el contrario, la especificación *IEEE 488* (comunicación en paralelo) determina que el largo del cable para el equipo no puede ser mayor a 20 metros, con no más de 2 metros entre cualesquier dos dispositivos; por el contrario, utilizando comunicación serial el largo del cable puede llegar a los 1200 metros.

¿Qué es la comunicación serial?

- Típicamente, la comunicación serial se utiliza para transmitir datos en formato ASCII.
- Para realizar la comunicación se utilizan 3 líneas de transmisión:
  - (1) Tierra (o referencia),
  - (2) Transmitir,
  - (3) Recibir.
- Debido a que la transmisión es asíncrona, es posible enviar datos por un línea mientras se reciben datos por otra.

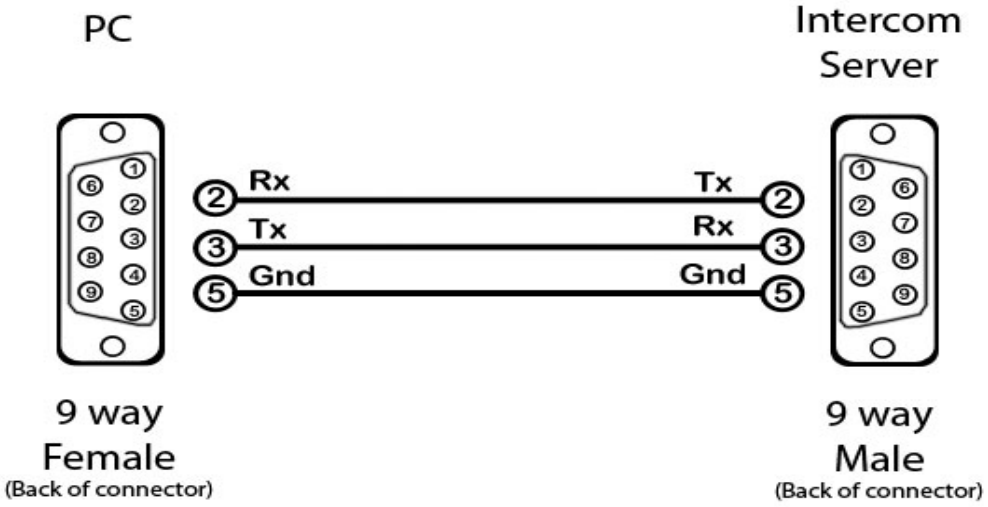
RS232

Pin 1	DCD
Pin 2	RXD
Pin 3	TXD
Pin 4	DTR
Pin 5	GND
Pin 6	DSR
Pin 7	RTS
Pin 8	CTS
Pin 9	RI



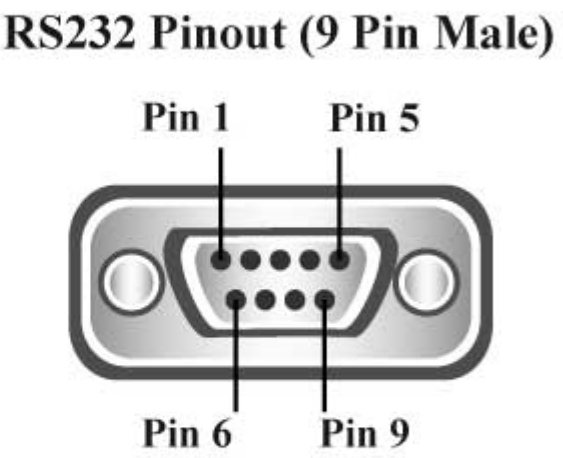
¿Qué es la comunicación serial?

- Existen otras líneas disponibles para realizar *handshaking*, o intercambio de pulsos de sincronización, pero no son forzosamente requeridas.



RS232

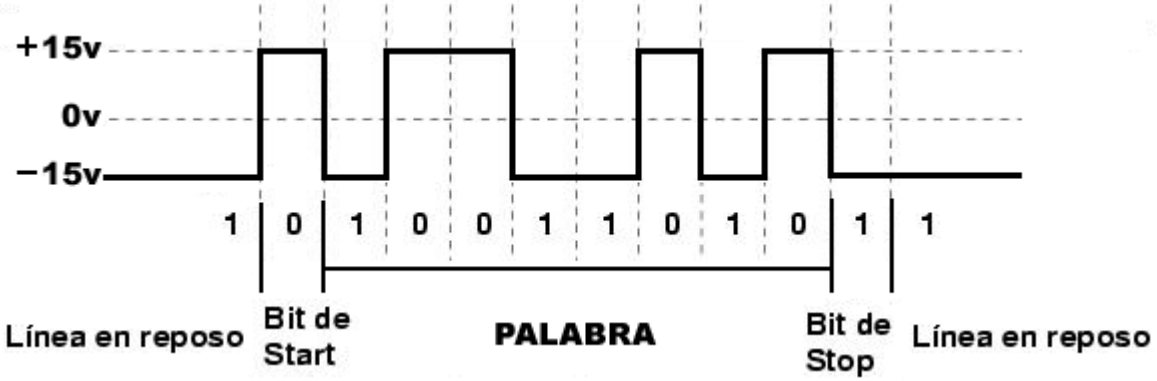
Pin 1	DCD
Pin 2	RXD
Pin 3	TXD
Pin 4	DTR
Pin 5	GND
Pin 6	DSR
Pin 7	RTS
Pin 8	CTS
Pin 9	RI



¿Qué es la comunicación serial?

Las características más importantes de la comunicación serial son:

- la **velocidad de transmisión**
- El **número de bits de datos**
- El número de **bits de paro**
- Y si cuenta con **bit de paridad**



Para que dos puertos se puedan comunicar, es necesario que las características sean iguales.

Velocidad de transmisión *(baud rate)*:

- Indica el número de bits por segundo que se transfieren, y se mide en baudios (*bauds*).
  - Por ejemplo, 300 baudios representa 300 bits por segundo.
- Cuando se hace referencia a los ciclos de reloj se está hablando de la velocidad de transmisión.
  - Por ejemplo, si el protocolo hace una llamada a 4800 ciclos de reloj, entonces el reloj está corriendo a 4800 Hz, lo que significa que el puerto serial está muestreando las líneas de transmisión a 4800 Hz.

Velocidad de transmisión *(baud rate)*:

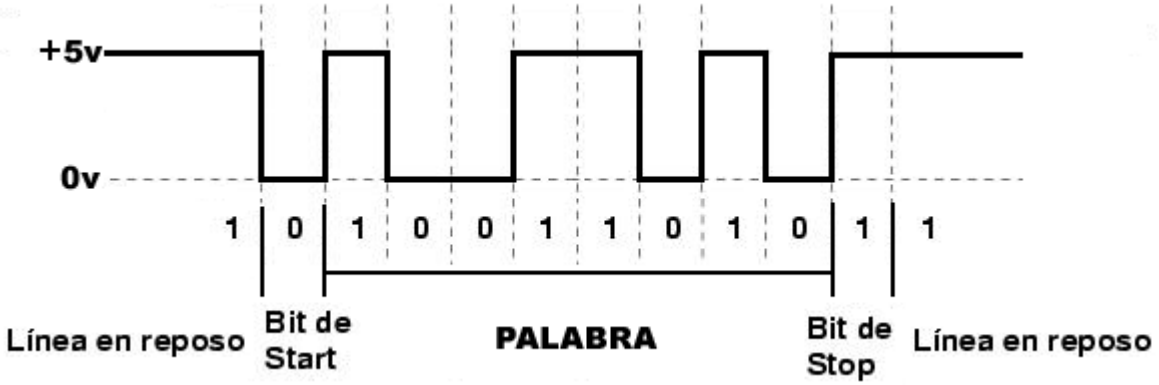
- Las velocidades de transmisión más comunes son de **115200, 9600**, y **4800**.

- Es posible tener velocidades más altas, pero se reduciría la distancia máxima posible entre los dispositivos.

- Las altas velocidades se utilizan en comunicaciones en paralelo cuando los dispositivos se encuentran uno junto al otro, como es el caso de dispositivos GPIB / IEEE488.

Bits de datos:

- Se refiere a la cantidad de bits (palabra) en la transmisión.
- Cuando la computadora envía un paquete de información, el tamaño de ese paquete no necesariamente será de **8 bits**.
- Las cantidades más comunes de bits por paquete son **5, 7 y 8** bits.
- El número de bits que se envía depende en el tipo de información que se transfiere.



Bits de datos:

- Por ejemplo, la representación de caracteres ASCII estándar tiene un intervalo de valores que va de 0 a 127, es decir, utiliza 7 bits.
- Para **ASCII extendido** es de **0 a 255**, lo que utiliza **8 bits**.
- Si el tipo de datos que se está transfiriendo es texto simple (ASCII estándar), entonces es suficiente con utilizar 7 bits por paquete para la comunicación.
- Un **paquete** se refiere a una transferencia de un byte, incluyendo los bits de inicio/paro, bits de datos, y paridad. Debido a que el número actual de bits depende en el protocolo que se seleccione, el término paquete se usar para referirse a todos los casos.

Bits de paro:

- Usado para indicar el fin de la comunicación de un solo paquete.
- Los valores típicos son **1, 1.5 o 2 bits**.
- Debido a la manera como se transfiere la información a través de las líneas de comunicación y que cada dispositivo tiene su **propio reloj**, es posible que los dos dispositivos **no estén sincronizados**. Por lo tanto, los bits de paro no sólo indican el fin de la transmisión sino además dan un margen de tolerancia para esa diferencia de los relojes.
- Mientras **más bits de paro se usen**, mayor será la **tolerancia** a la sincronía de los relojes, sin embargo la transmisión será más lenta.

Paridad:

- Es una forma sencilla de verificar si hay errores en la transmisión serial.
- Existen cuatro tipos de paridad:
  - par,
  - impar,
  - marcada y
  - espaciada.
- La opción de no usar paridad alguna también está disponible.

Start bit	Data 0	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Parity bit(optional)	Stop bit

Figure: Logical frame

Paridad:

En caso de **habilitar** la **paridad par o impar**, el puerto serial fijará el bit de paridad (el último bit después de los bits de datos) a un valor para asegurarse que la transmisión tenga un **número par o impar de bits** en estado lógico alto.

- Por ejemplo, si la información a transmitir es 011 y la paridad es par, el bit de paridad sería 0 para mantener el número de bits en estado alto lógico como par.

- Si la paridad seleccionada fuera impar, entonces el bit de paridad sería 1, para tener 3 bits en estado alto lógico.

Paridad:

- La paridad marcada y espaciada en realidad no verifican el estado de los bits de datos; simplemente fija el bit de paridad en **estado lógico alto para la marcada**, y en **estado lógico bajo para la espaciada**.

- Esto permite al dispositivo receptor **conocer de antemano el estado de un bit**, lo que serviría para determinar si hay ruido que esté afectando de manera negativa la transmisión de los datos, o si los relojes de los dispositivos no están sincronizados.

RS-232 (Estándar ANSI/EIA-232)

Es el conector serial hallado en las PCs IBM y compatibles.

Es utilizado para una gran variedad de propósitos, como conectar un ratón, impresora o modem, así como instrumentación industrial.

Gracias a las mejoras que se han ido desarrollando en las líneas de transmisión y en los cables, existen aplicaciones en las que se aumenta el desempeño de RS-232 en lo que respecta a la distancia y velocidad del estándar.

RS-232 (Estándar ANSI/EIA-232)

RS-232 está limitado a comunicaciones de punto a punto entre los dispositivos y el puerto serial de la computadora. El hardware de RS-232 se puede utilizar para comunicaciones seriales en distancias de hasta 50 pies.

- “1” lógico: -3v..-25v
- “0” lógico: +3v..+25v
- Mark: “1”
- Space: “0”
- Start bit: “0”
- Stop bit: “1”

Terminales del RS-232:

Funciones de los Pines del RS232

DB9	DB25	MISIÓN	DEFINICION
1	8	DCD	Deteccion portadora de datos
2	3	RxD	Recepcion de Datos
3	2	TxD	Transmision de Datos
4	20	DTR	Terminal de Datos Listo
5	7	GND Signal	Circuito Común
6	6	DSR	Dispositivo de Datos Listo
7	4	RTS	Petición de Envio
8	5	CTS	Dispositivo de Datos Listo
9	22	RI	Indicador de llamada (Ring)

¿Qué es RS-422?

RS-422 (Estándar EIA RS-422-A) es el conector serial utilizado en las computadoras Apple de Macintosh.

RS-422 usa señales eléctricas diferenciales, en comparación con señales referenciadas a tierra como en RS-232.

La transmisión diferencial, que utiliza dos líneas para transmitir y recibir, tiene la ventaja que es más inmune al ruido y puede lograr mayores distancias que RS-232.

La inmunidad al ruido y la distancia son dos puntos clave para ambientes y aplicaciones industriales.

¿Qué es RS-485?

RS-485 (Estándar EIA-485) es una mejora sobre RS-422 ya que incrementa el número de dispositivos que se pueden conectar (de 10 a 32) y define las características necesarias para asegurar los valores adecuados de voltaje cuando se tiene la carga máxima.

Gracias a esta capacidad, es posible crear redes de dispositivos conectados a un solo puerto RS-485.

¿Qué es RS-485?

Esta capacidad, y la gran inmunidad al ruido, hacen que este tipo de transmisión serial sea la elección de muchas aplicaciones industriales que necesitan dispositivos distribuidos en red conectados a una PC u otro controlador para la colección de datos, HMI, u otras operaciones.

RS-485 es un conjunto que cubre RS-422, por lo que todos los dispositivos que se comunican usando RS-422 pueden ser controlados por RS-485.

El hardware de RS-485 se puede utilizar en comunicaciones seriales de distancias de hasta 4000 pies de cable.

¿Qué es *handshaking*?

- El método de comunicación usado por RS-232 requiere de una conexión muy simple, utilizando sólo tres líneas: Tx, Rx, y GND.

- El esquema se basa en un intercambio establecido de datos o señales los cuales indican al receptor o transmisor que la contra-parte está preparada para recibir la información

- Aun y cuando este método es más que suficiente para la mayoría de las aplicaciones, es limitado en su respuesta a posibles problemas que puedan surgir durante la comunicación.

¿Qué es *handshaking*?

por ejemplo, si el receptor se comienza a sobrecargar de información.

Es en estos casos cuando el intercambio de pulsos de sincronización, o *handshaking*, es útil.

Las tres formas más populares de *handshaking* con RS-232:

- handshaking for software,
- handshaking por hardware, y
- XModem.

Introducción

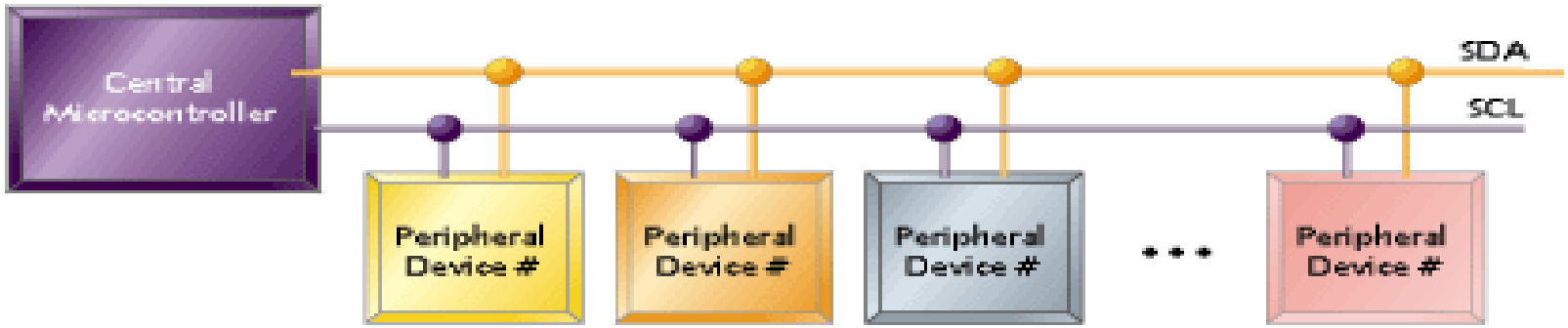
I^2C y SPI

- Protocolos de comunicación síncrona
- Diseñados para alcanzar distancias cortas “inside the box”
- Baja complejidad
- Bajo costo
- Baja velocidad (< algunos cuantos Mbps)

¿Que es el I²C?

- Acrónimo: “Inter-integrated circuit” bus
- Desarrollado por Philips Semiconductor para televisiones en los años 1980’s
- El protocolo I²C se incluye en EEPROMs, sensores y relojes de tiempo real.
- Utilizado como una interface de control para dispositivos que procesan información de diversos tipos y de manera separada e.g. video decodificadores / codificadores, sensores, sintonizadores, receptores IR, etc.).
- El bus I²C puede manejar 3 velocidades :
  - Baja (< 100 Kbps)
  - Media (400 Kbps)
  - Alta (3.4 Mbps) – I²C v.2.0
- Distancia: hasta 3 metros a velocidades moderadas.

Configuración del bus I²C



Configuración de sólo 2 cables

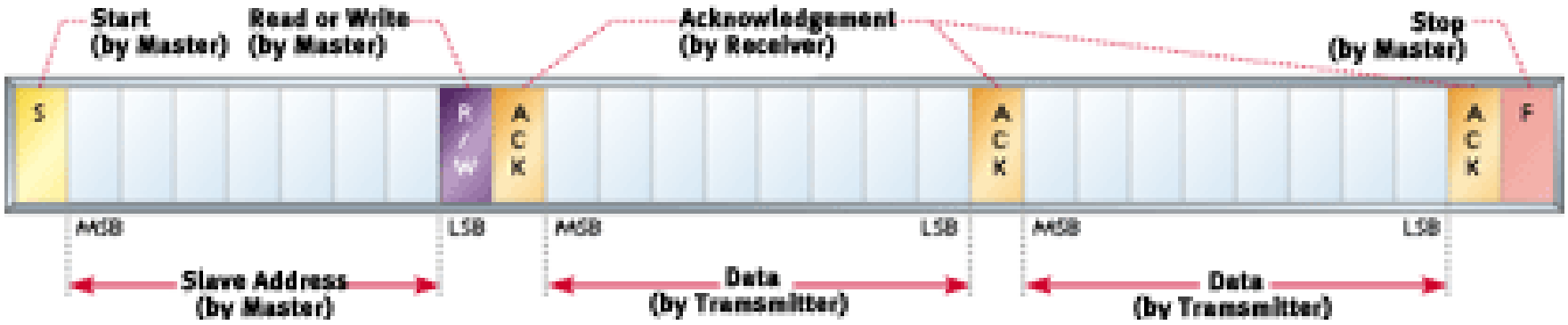
- Serial data (SDA) y Serial clock (SCL)
- Half-duplex, síncrono.

• Capacidad de manejo de múltiples maestros

• No se requiere «chip select» o lógica extra para el direccionamiento de los dispositivos

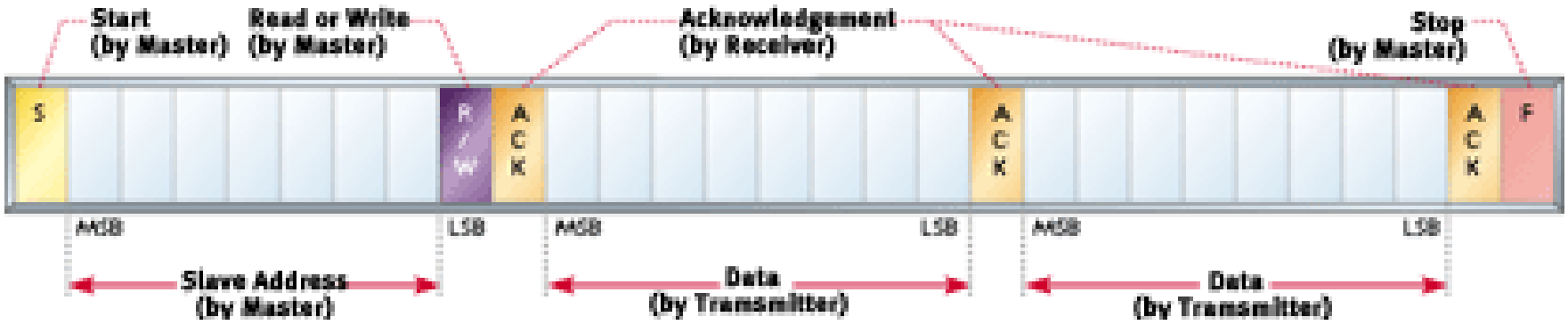
• Las líneas de transmisión son polarizadas mediante resistores de «pull up» y resetadas (0 lógico) mediante transistores de colector abierto (AND-wired)

Protocolo I²C



1. El Maestro manda un bit de inicio (S) y controla la señal de reloj
2. El Maestro envía una dirección única (del esclavo) de 7-bits
3. El Maestro envía un bit read/write (R/W) –
   - 0 : slave receive
   - 1 : slave transmit
4. El Receptor manda a su vez un bit de « acknowledge » (ACK)
5. El Transmisor (esclavo o maestro) transmite 1 byte de datos

Protocolo I²C

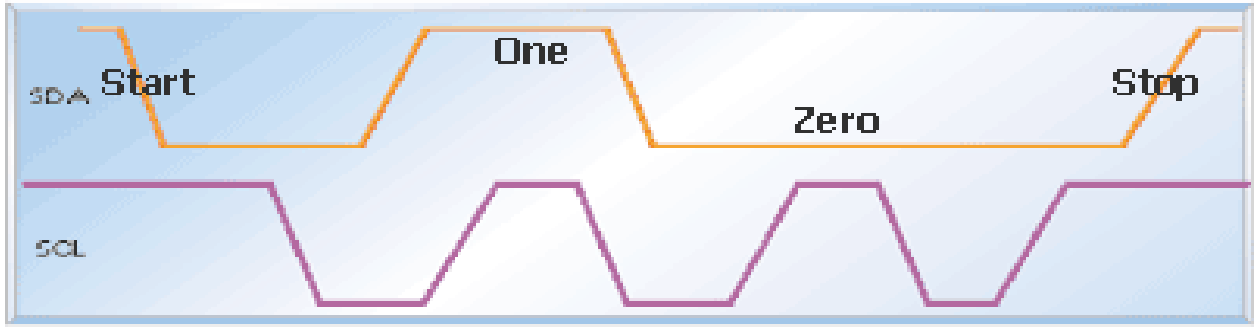


6. El receptor del dato, a su vez manda una señal de ACK (1 bit) por el byte recibido
7. Se repite 5 y 6 si se requiere transmitir más bytes.

8.a) Para la operación de escritura (*master transmitting*), el maestro es el que genera la condición de paro (P) después de transmitir el último byte de datos.

8.b) Para la operación de lectura (*master receiving*), el maestro **NO** genera la señal de ACK del último byte, simplemente genera la condición de paro (P). Avisando al esclavo que se finaliza la transmisión.

Señales del I²C



Sample Signals

- **Start** – Mientras la línea SCL permanece en alto, se genera una transición *high-to-low* de la línea SDA.
- **Stop** – Mientras la línea SCL permanece en alto, se genera una transición *low-to-high* de la línea SDA.
- **Data** – La transmisión de datos ocurre cuando la señal de reloj SCL está en alto (*high*) mientras que las transiciones entre los niveles de datos se permiten cuando la señal SCL está en bajo (*low*).
- **Ack** – Mientras el transmisor permite que flote la línea SDA en alto (*high*) el receptor la jala (pulls) a nivel bajo (*low*).

Particularidades del I²C

- « ***Clock stretching*** »: Cuando el esclavo (receptor) requiere de más tiempo para procesar un bit **puede** mantener baja (***pull-low***) la señal de reloj SCL. El maestro espera en ese caso hasta que el esclavo libere la línea del SCL antes de continuar con el siguiente bit.

- « ***General call*** » transmisión que se recibe por todos los dispositivos conectados al bus.

- Direccionamiento extendido de 10-bits (para diseños nuevos) en caso de que ya no queden disponibles direcciones 7-bits

Ventajas y Desventajas del I²C

Ventajas:

- Excelente protocolo de comunicación para dispositivos en la misma tarjeta de datos con tasas de transmisión bajas o transmisión ocasional de datos entre dispositivos.
- Fácil de conectar múltiples dispositivos debido a su esquema de direccionamiento
- El costo y la complejidad no se incrementan con la adición de nuevos nodos(dispositivos).

Desventajas:

- La complejidad del software para el soporte de los dispositivos es mayor que en otros sistemas de comunicación serial síncrona (e.g.: SPI).

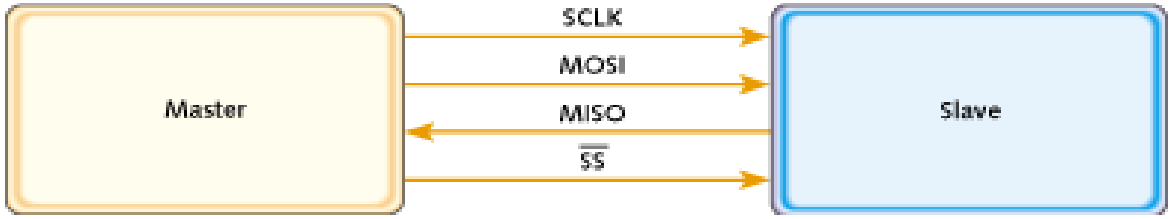
Que es SPI

- Acrónimo para « ***Serial Peripheral Interface*** »
- Definido por Motorola en la familia de micro-controladores MC68HCxx.
- Generalmente más rápido que el I^2C, capaz de transmitir varios Mbps

Aplicaciones:

- Al igual que el I^2C, se usa en EEPROM, Flash, y relojes (*real time clocks*)
- Mejor situado para la transmisión de tramas de datos «data streams» entre componentes, i.e. comunicación de ADC's, DAC's ...
- Capacidad «*Full-duplex*», i.e. Recepción-transmisión de datos simultánea entre el maestro y el esclavo

Configuración del Bus SPI



Enlace de datos serial síncono con operación *full-duplex*

Relación Maestro/Esclavo

2 señales de datos:

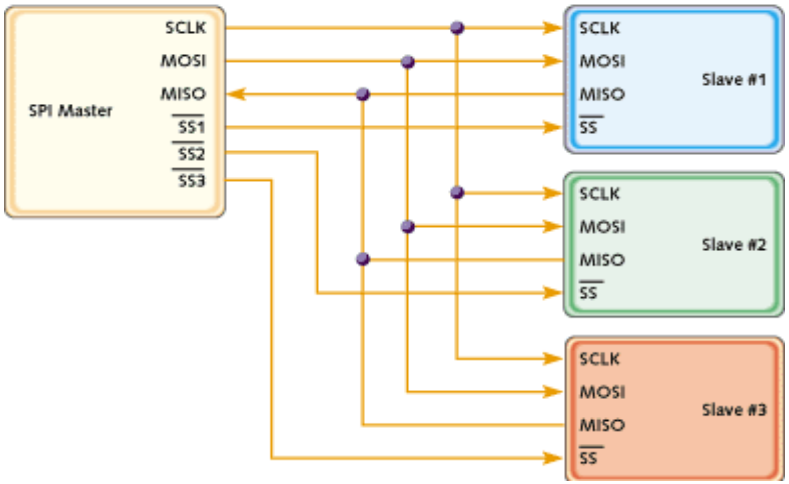
- ***MOSI*** *– master data output, slave data input*
- ***MISO*** *– master data input, slave data output*

2 señales de control :

- ***SCLK*** – reloj
- ! **SS** – selección de esclavo

  (no requiere direccionamiento)

SPI vs. I²C



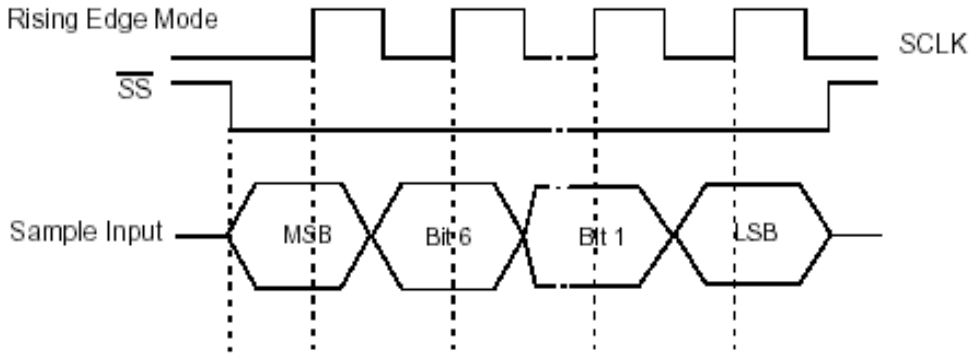
Para comunicación point-to-point, SPI es más simple y eficiente

- Menor ***overhead*** respecto al I²C debido principalmente al direccionamiento directo. Además el SPI opera en modo ***full-duplex.***

Para el caso de múltiples esclavos, cada esclavo requiere de una señal separad de « ***slave select*** »

- Mayor esfuerzo y más hardware en comparación contra el I²C

Protocolo del SPI



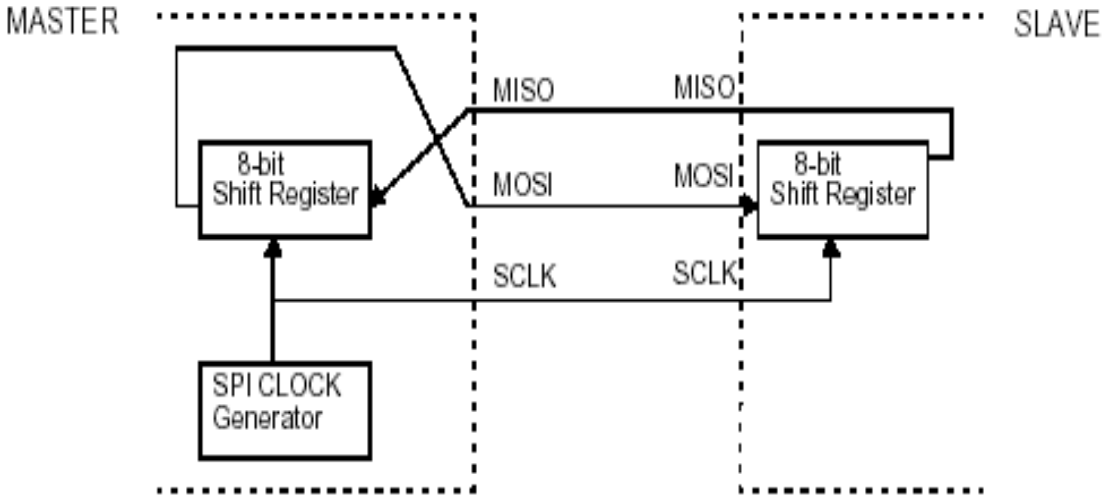
2 Parametros definen el flanco en el que el reloj de la transmisión valida el dato, ***Clock Polarity (CPOL) and Clock Phase (CPHA)***

CPOL	CPHA	Active edge
0	0	Rising
0	1	Falling
1	0	Falling
1	1	Rising

El Maestro y el Esclavo deben concordar en este par de parámetros de lo contrario la comunicación no será posible

Protocolo del SPI

- El SPI define únicamente las lineas y el flanco del reloj
- No existe implementado o definido ninguna especificación de control de flujo (***flow control***).
- No existen mecanismos de « ***acknowledgement*** » para confirmar la recepción de los datos



La implementación de Hardware generalmente se hace empleando un simple registro de corrimiento

Resúmen

I²C y SPI proveen un buen marco para la comunicación entre dispositivos de baja velocidad o que su acceso a los datos es intermitente, principalmente sensores, memorias, EEPROMs y relojes (real-time clocks).

I²C permite direccionar facilmente multiple dispositivos en un bus único.

SPI es más rápido, pero se puede complicar su implementación si se trata de comunicar con más de un dispositivo.