Tema 4

Buses industriales

Temario

1. Sistemas empotrados: ámbitos de aplicación y flujo de diseño (2 horas)
2. Microprocesadores, microcontroladores y procesadores de señal digital (4+1 horas)
3. Subsistema de memoria en sistemas empotrados. (3+1 horas)
4. **Buses industriales. (2+2 horas)**
5. Periféricos: sensores y actuadores. (3+1 horas)
6. Integración, coste y prestaciones. (8+2 horas)
7. Casos prácticos. (8+3 horas)

Prácticas: 3-4 prácticas con el entorno EDK Xilinx y placas de Spartan 3 (10 horas)

Bibliografía

Bibliografía básica

- Computers as components: principles of embedded computing system design. Capítulo 4. Autor: Marilyn Wolf. San Francisco, Editorial: Morgan Kaufmann Publishers, 4th Edition, 2016, ISBN: 9780128053874

- Embedded Systems Design, Capítulo 5. Autor: Steve Heath, Editoral Elsevier Science & Technology, 2nd Edition, 2002, 978-0750655460

- Embedded hardware., know it all. Capítulos 4, 8 y 9 Autores: Jack Ganssle, Tammy Noergaard, Fred Eady, Lewin Edwards, David J. Katz, Amsterdam. Editorial: Elsevier/Newnes,. 2007

Bibliografía complementaria:

- Embedded Systems Architecture. A Comprehensive Guide for Engineers and Programmers. Tammy Noergaard. Ed Elsevier, Second Edition, 2013

Tema 4

Buses industriales. (2+2 horas)

4.1 Protocolos
4.2 Comunicación serie
4.3 Comunicación paralela
4.4 Rendimiento del bus
4.5 Ejemplos

Presentaciones de los alumnos:

- Bus SPI
- CAN BUS
- JTAG
- Bluetooth
- Zigbee
- Wifi
- RFID
- NFC

4.1 Protocolos

Los buses son los encargados de conectar los componentes de un sistema empotrado

Se trata de una colección de cables que llevan señales de:

1.- Datos

2.- Direcciones

3.- Control: reloj, petición, reconocimiento

En tarjetas muy complejas puede haber varios buses y puentes (bridges) que se encargan de conectar los buses entre sí.

Los puentes proporcionan un mapeo transparente de información cuando los datos se transfieren de un bus a otro

Tipos de buses:

1.- Bus del sistema o principal. Conecta la memoria principal y la cache al master. Suelen ser cortos y rápidos

2.- Backplane: conectan memoria, master y E/S en un único bus. Son rápidos

3.- Bus de E/S o de expansión: resto de componentes. Suelen ser estándares (USB, PCI). Suelen tener posibilidad de interrupciones

4.1 Protocolos

Otra clasificación:

1.- Expandibles: permite que nuevas componentes se conecten on-the-fly
ej. IDE, SCASI, USB

2.- No expandibles: I2C, VME

El protocolo de un bus define:

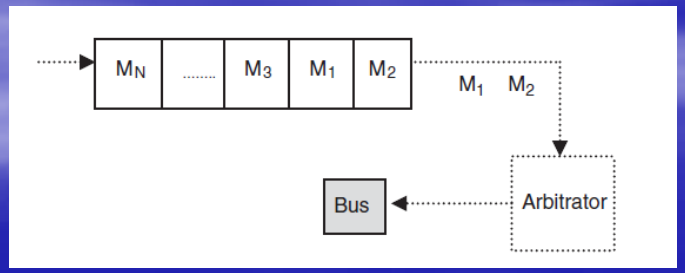
1.- Cómo obtener el acceso al bus: arbitraje

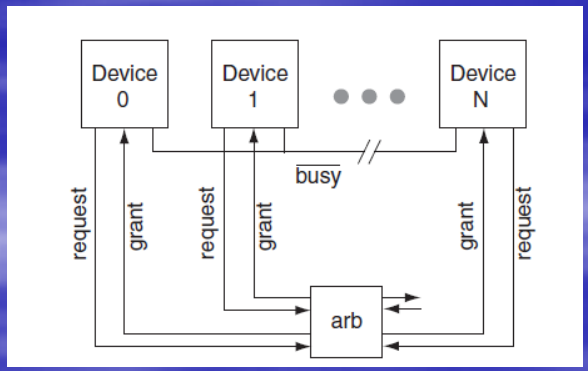
2.- Reglas de comunicación: handshaking

Protocolos de arbitraje

Maestros: son los dispositivos que pueden iniciar una transacción. Si sólo hay un maestro no es necesario ningún protocolo de arbitraje
Esclavos: son los dispositivos que sólo pueden obtener el acceso a bus en respuesta a una petición de un maestro.
Árbitro: hw que determina bajo qué circunstancia un maestro puede conseguir el control del bus
Esquemas de arbitraje:

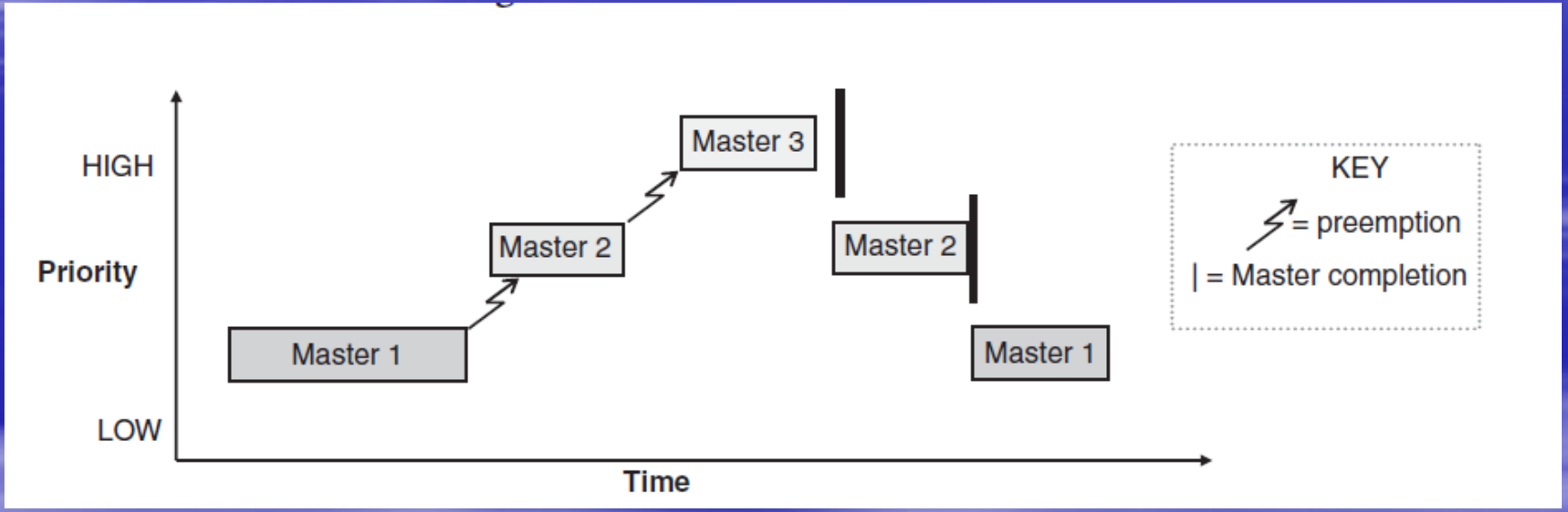
1.- **Arbitraje Paralelo central dinámico**





Arbitraje basado en FIFO

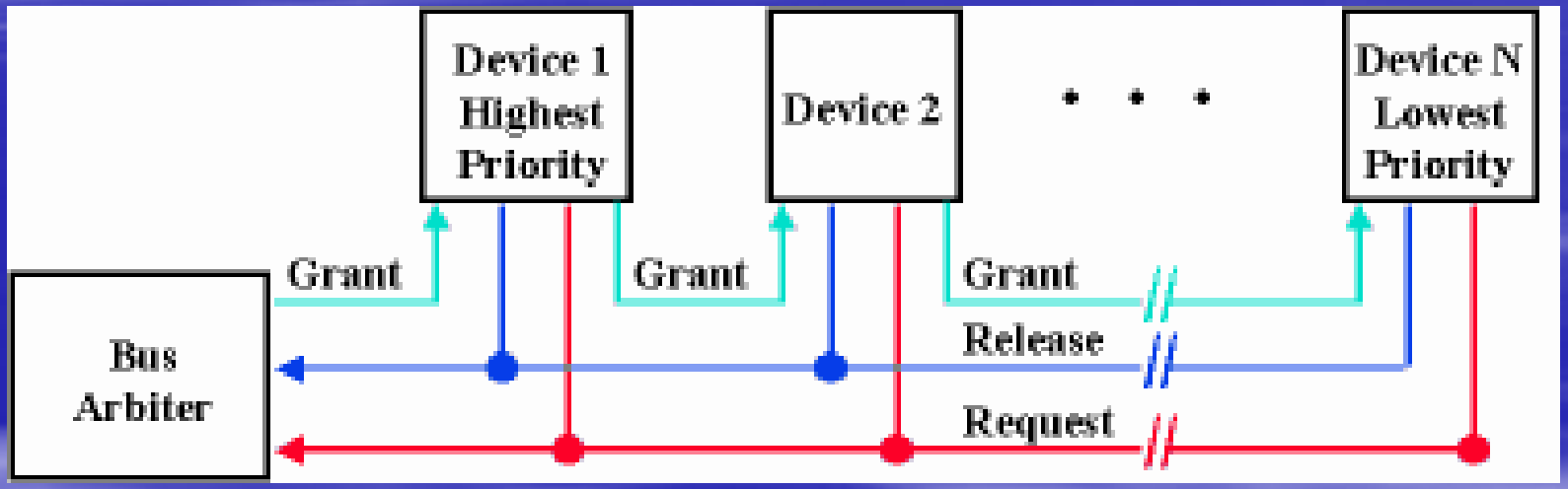
Protocolos de arbitraje



El arbitraje tb puede estar basado en prioridades

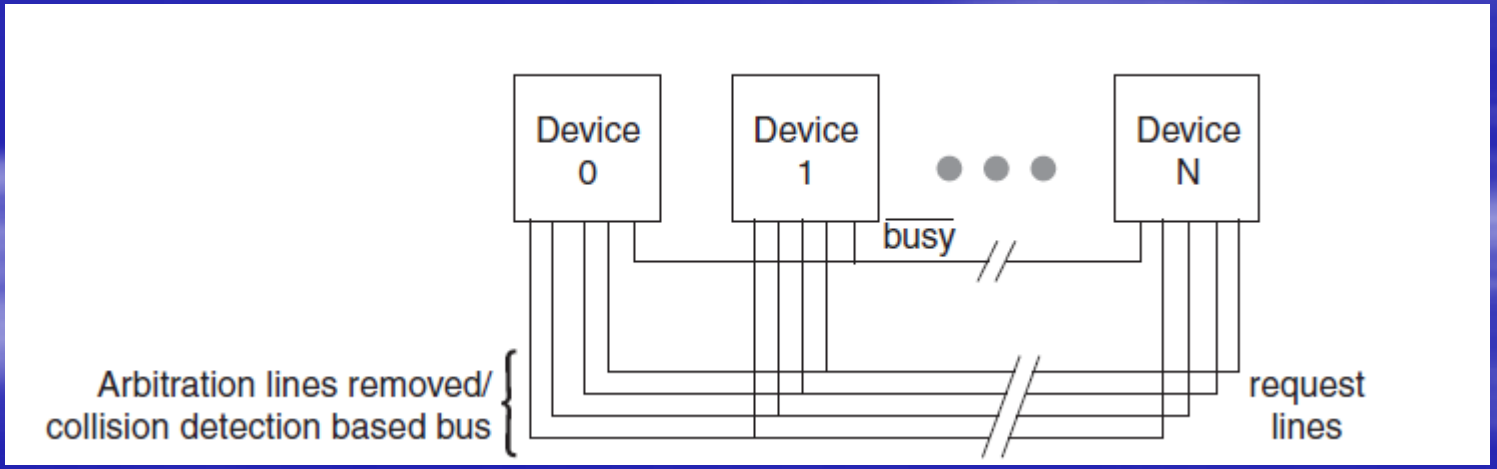
Protocolos de arbitraje

2.- Arbitraje central serie (daisy-chain)



Protocolos de arbitraje

3.- **Arbitraje distribuido** (Self selection): no existe un árbitro central. Existe información de prioridad que permite determinar si un master de mayor prioridad está haciendo una petición



Protocolos de comunicación

Una vez que un maestro tiene el control del bus, la comunicación se realiza entre 1 maestro y uno o varios esclavos.
Hay dos tipos de transacciones, de lectura y de escritura y cada una con sus propias reglas que determinan el protocolo o handshake

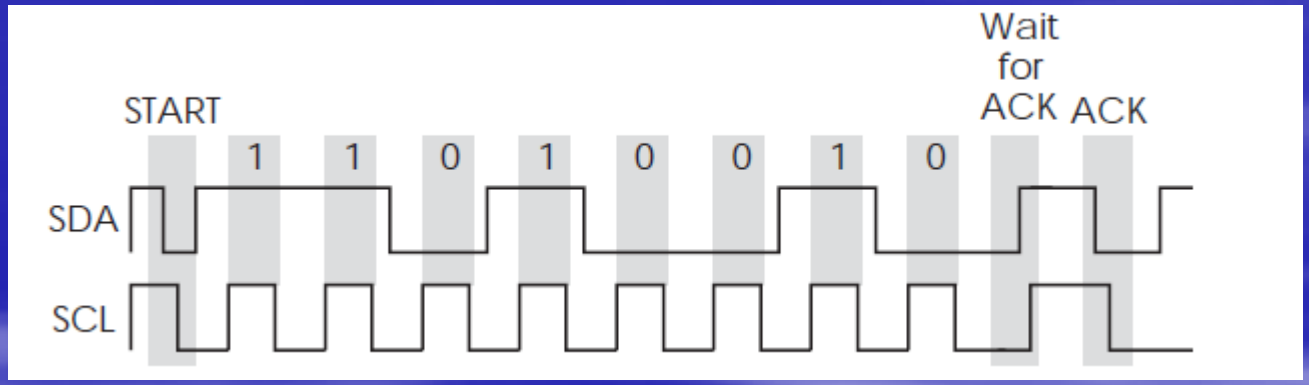
La base de cualquier handshake es el esquema de tiempo, y puede ser síncrono o asíncrono
Bus síncrono: se transmite tb el reloj, y las transacciones se realizan en el flanco de subida o bajada del mismo. Sólo funciona a bajas frecuencias o si las componentes están muy próximas entre sí.
Bus asíncrono: la señal de reloj no se transmite. Existen otras señales para establecer las comunicaciones, como la de request y acknowledge. La longitud del bus puede ser mayor.

Protocolos de comunicación

Los 2 protocolos más básico que inician cualquier comunicación son:

1.- El maestro indica o pide una transacción de lectura o escritura a un determinado esclavo

2.- El esclavo responde a la petición enviando un ACK



Si hay una petición de datos de una posición de memoria o de un controlador de E/S, la dirección correspondiente se envía. Si existen líneas de dirección, se utilizan las mismas. Si la dirección es válida, se envía el dato. Puede usarse el modo ráfaga (burst).

Tema 4

Buses industriales. (2+2 horas)

4.1 Protocolos

4.2 Comunicación serie

4.3 Comunicación paralela

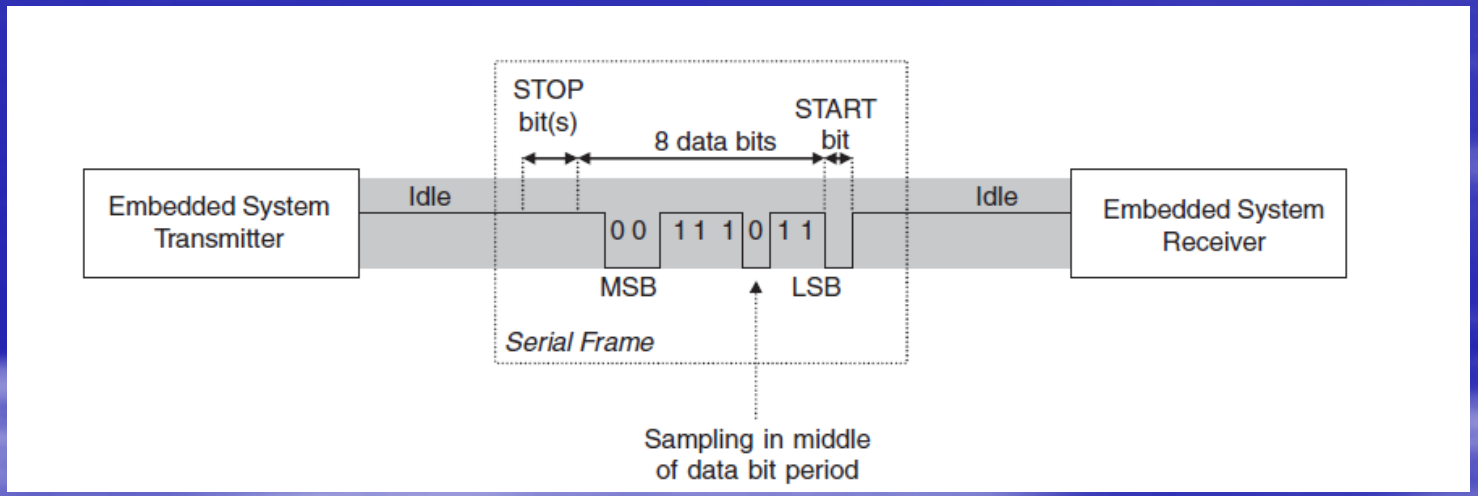
4.4 Rendimiento del bus

4.5 Ejemplos

4.2 Comunicación serie

Las transferencias pueden ser síncronas o asíncronas

En la transferencia síncrona hay un reloj que marca cuando muestrear. Ej SPI (Serial Peripheral Interface)



Trasferencia asíncrona: se necesitan bits de START y STOP

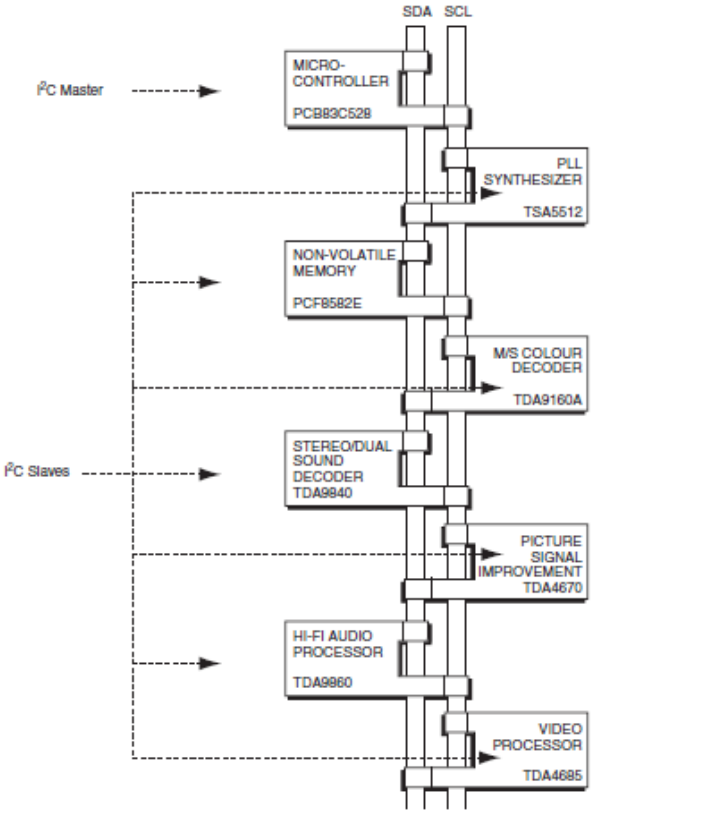
Puede añadirse un bit de paridad par o impar

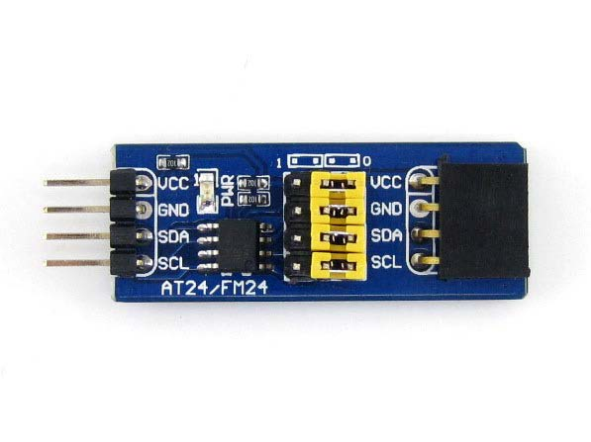
Ej. UART: Universal Asynchronous Receiver-Transmitter

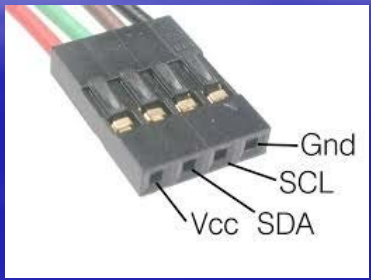
4.2 Comunicación serie: bus I2C

Inter I C (I2C) Se trata de un bus no expandible que se utiliza para conectar procesadores y periféricos con un interfaz I2C. Las transacciones son de tipo maestro/esclavo. Cada dispositivo se direcciona por una dirección única, que es parte de la cadena enviada

SDA: Serial Data Line
SCL: Serial Clock Line

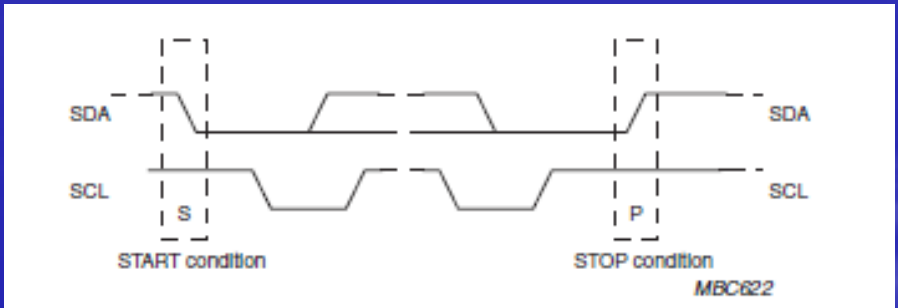






4.2 Comunicación serie: bus I2C

Toda transacción comienza con un START (el maestro pone SDA a L con SCL a H) y termina con un STOP (SDA a H con SCL a HIGH)



En cada transferencia se transmite 1 byte (1 bit por ciclo).
El dato se lee por flanco de bajada de SCL

Message	1st event	2nd event
START	SDA H\L	SCL H\L
STOP	SCL L\H	SDA L\H
ACK	SDA H\L	SCL H\L

4.2 Comunicación serie: bus I2C

La línea SDA lleva información de dirección, datos y reconocimiento (ACK)
Tanto el maestro como el esclavo pueden actuar como emisor o receptor, aunque sólo el primero puede iniciar una transmisión.
Cada dispositivo tiene una única dirección de 7 o 10 bits que se usa para identificarlo.
Una transferencia tiene las siguientes etapas:

1.- Idle SDA=SCL=H

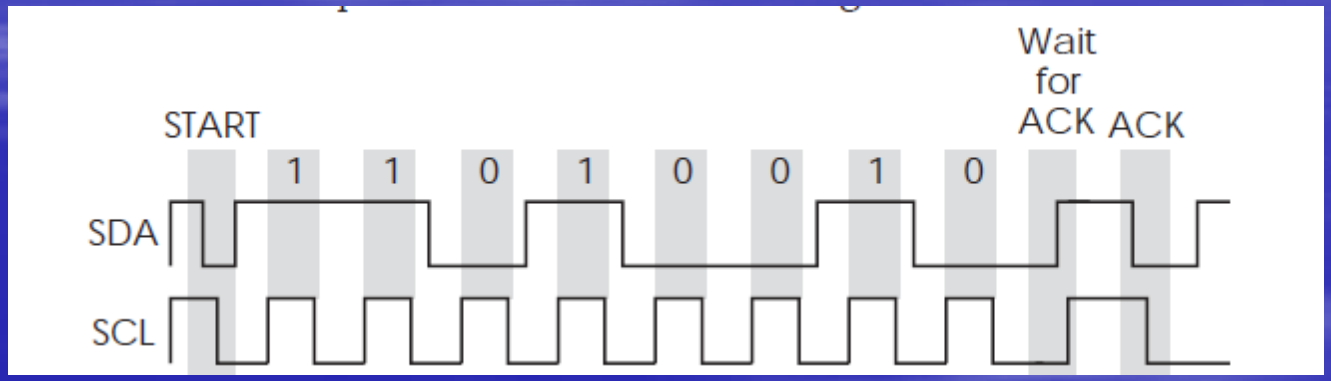
2.- Start SDA H/L y SCL=H indica comienzo

3.- Address: el master envía la dirección y el modo de transferencia (7+1 bit). El esclavo responde con un ACK

4.- Transferencia de datos: se transmite un byte del emisor al receptor + ACK del receptor al transmisor

5.- Stop SDA L/H con SCL =H

4.2 Comunicación serie: bus I2C



Escritura con Acknowledge

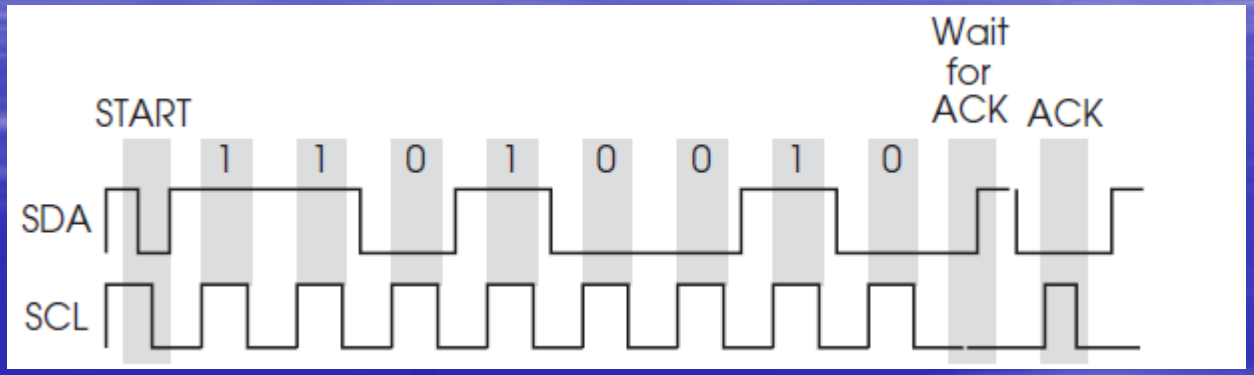
La dirección de escritura se incrementa automáticamente

Master	Slave

Idle	Start	SLAVE ADDRESS	Write mode	AcK	ADDRESS

AcK	Data	AcK	***	Data	AcK	Stop

4.2 Comunicación serie: bus I2C



Lectura con Acknowledge

En el último byte no se envía ACK para evitar nuevos envíos

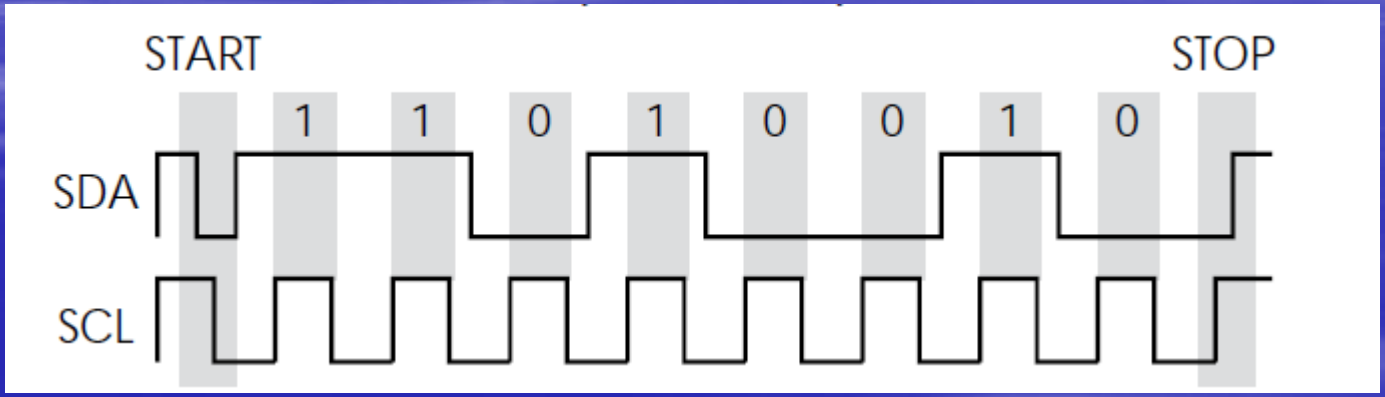
Master	**Slave**

Idle	Start	SLAVE ADDRESS	Read mode	**AcK**	ADDRESS

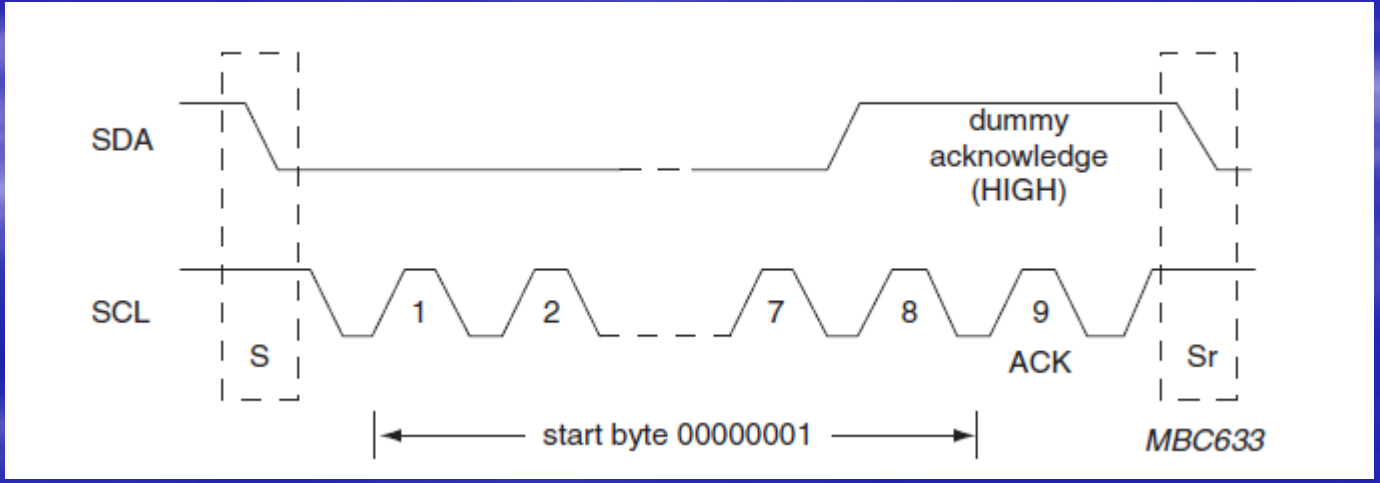
AcK	**Data**	Ack	***	**Data**	Ack	**Data**	Stop

4.2 Comunicación serie: bus I2C

Puede terminarse una transferencia sin ACK, con STOP

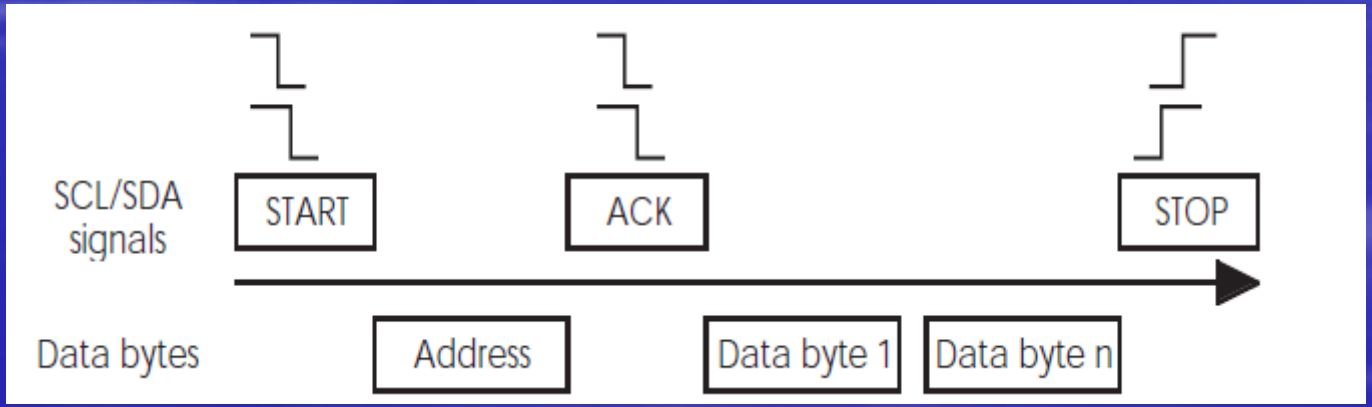


Escritura con Stop



4.2 Comunicación serie: bus I2C

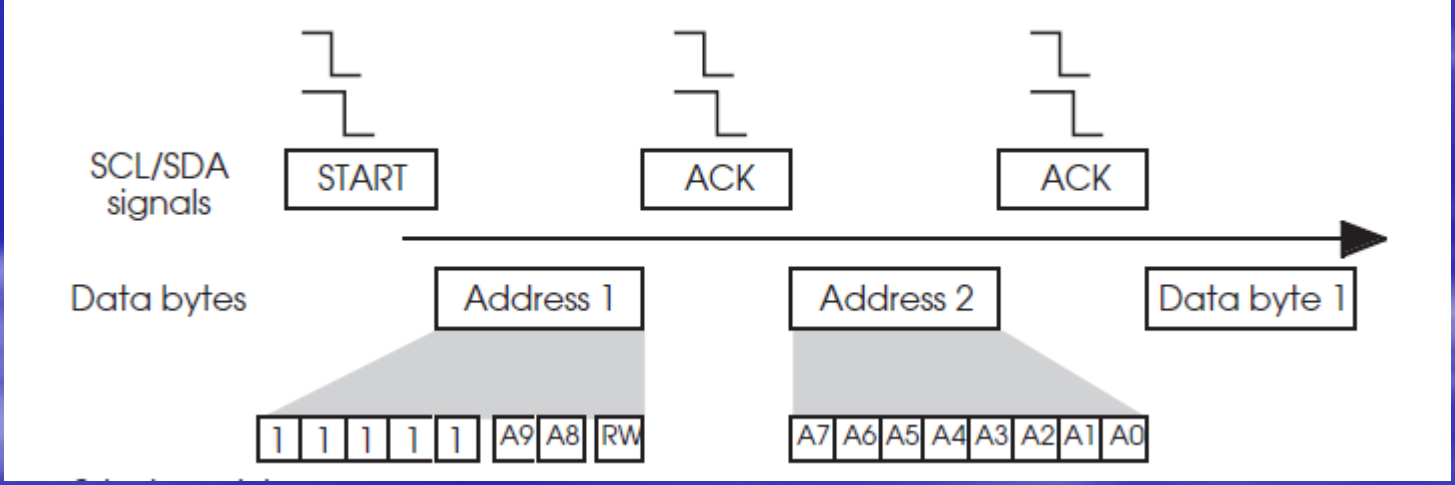
Direccionamiento



7 bits de dirección -128 dispositivos- + 1 modo
La dirección está preprogramada o se programa a través de switches externos

4.2 Comunicación serie: bus I2C

Modo direccionamiento extendido -2 bytes-



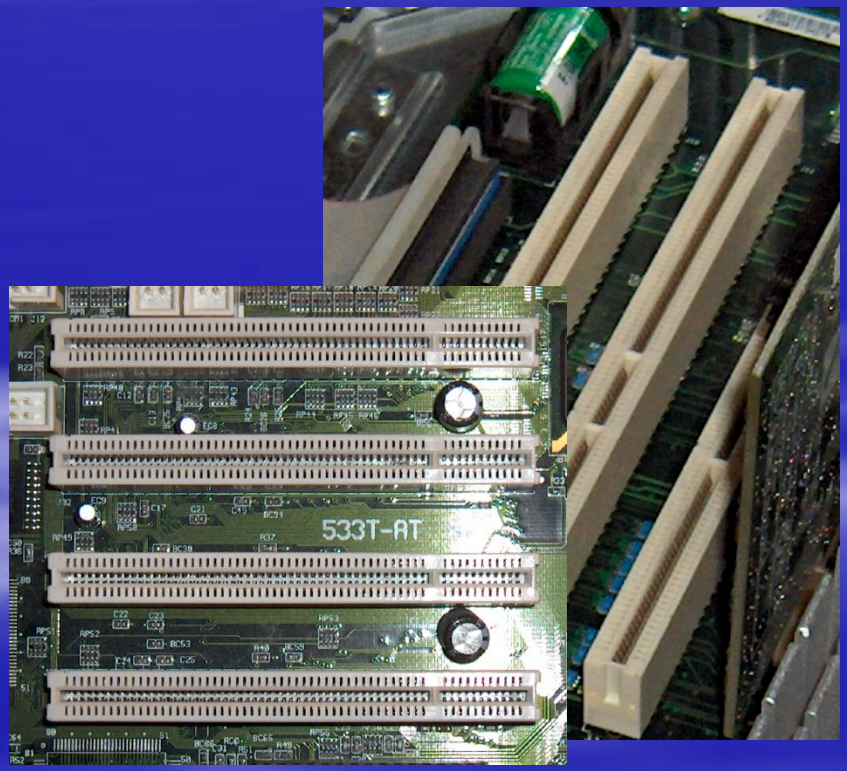
Tema 4

Buses industriales. (2+2 horas)

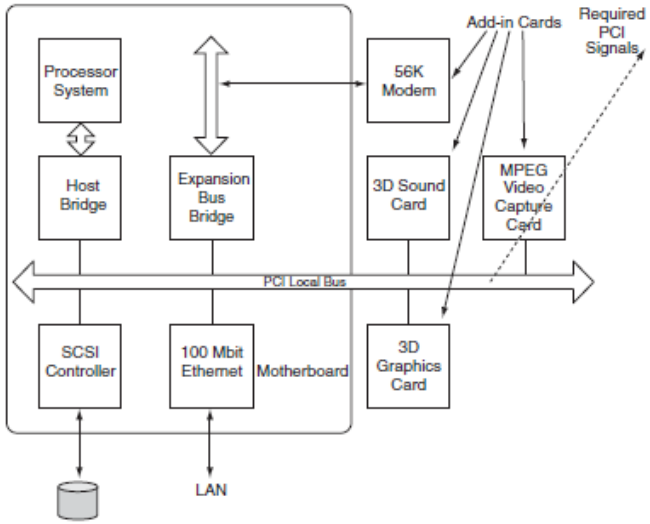
4.1 Protocolos
4.2 Comunicación serie
4.3 Comunicación paralela
4.4 Rendimiento del bus
4.5 Ejemplos

4.3 Comunicación paralela: PCI

El PCI (Peripheral Component Interconnect) es un bus síncrono (33Mhz-66Mhz) con un ancho de bits de 32-64, permitiendo hasta 528Mbytes/s



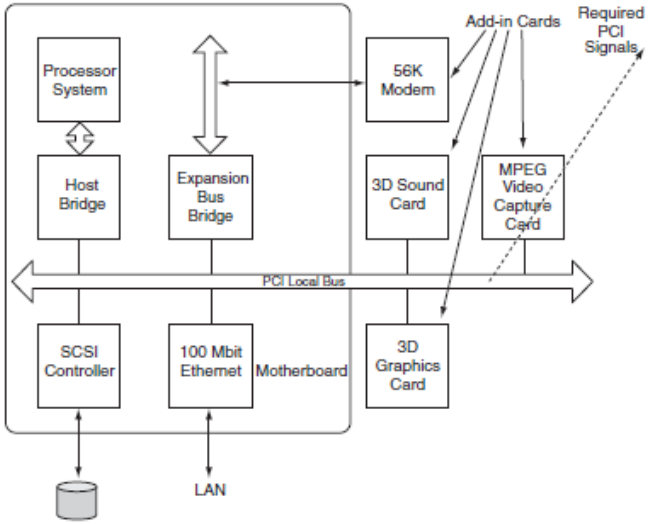
4.3 Comunicación paralela: PCI



Signal Name	Driven by	Description
CLK	Master	Bus Clock (normally 33MHz; DC okay)
FRAME#	Master	Indicates start of a bus cycle
AD[31:0]	Master/Target	Address/Data bus (multiplexed)
C/BE#[3:0]	Master	Bus command (address phase) Byte enables (data phases)
IRDY#	Master	Ready signal from master
TRDY#		Ready signal from target
DEVSEL#	Target	Address recognized
RST#	Master	System Reset
PAR	Master/Target	Parity on AD, C/BE#
STOP#		Request to stop transaction
ILESD	Target	Chip select during initialization transactions
PERR#	Receiver	Parity Error
SERR#	Any	Catastrophic system error
R#QE		Request Bus
G#TN		Bus Grant

Implementación 32 bits 49 líneas multiplexadas de datos y direcciones

4.3 Comunicación paralela: PCI

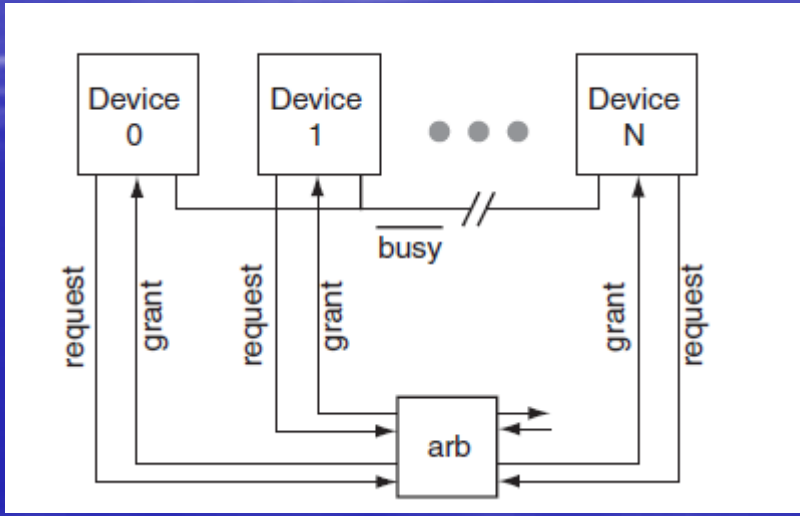


Signal Name	Driven by	Description
CLK	Master	Bus Clock (normally 33MHz; DC okay)
FRAME#	Master	Indicates start of a bus cycle
AD[31:0]	Master/Target	Address/Data bus (multiplexed)
C/BE#[3:0]	Master	Bus command (address phase) Byte enables (data phases)
IRDY#	Master	Ready signal from master
TRDY#		Ready signal from target
DEVSEL#	Target	Address recognized
RST#	Master	System Reset
PAR	Master/Target	Parity on AD, C/BE#
STOP#		Request to stop transaction
ILESD	Target	Chip select during initialization transactions
PERR#	Receiver	Parity Error
SERR#	Any	Catastrophic system error
R#QE		Request Bus
G#TN		Bus Grant

Datos y direcciones AD y C/BE System: CLK, RST#
Errores 2: PERR# SERR#
Arbitraje 2: R#QE G#TN
Control: Frame# IRDY# TRDY# DEVSEL# STOP#

4.3 Comunicación paralela: PCI

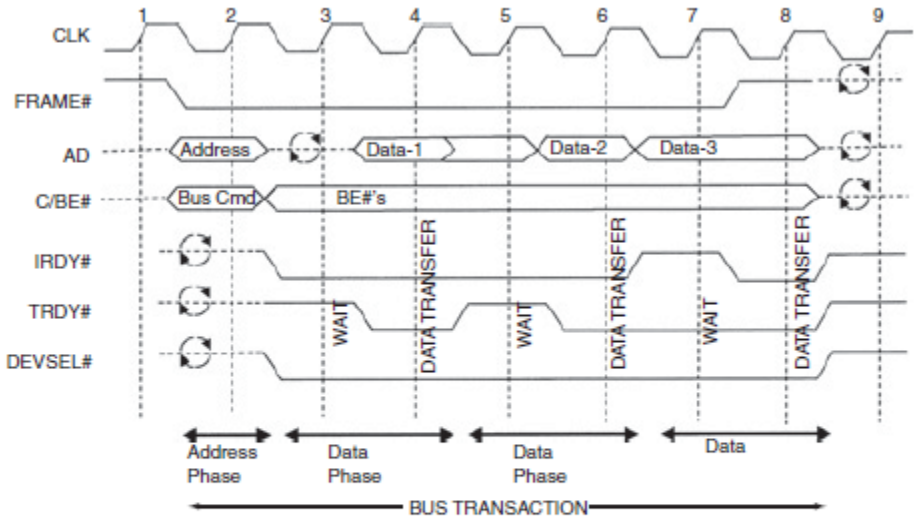
Arbitraje



Toda transacción comienza por un iniciador pidiendo el bus REQ# y el árbitro se lo concede GNT#

4.3 Comunicación paralela: PCI

lectura



CLK Cycle 1 – The bus is idle.

CLK Cycle 2 – The initiator asserts a valid address and places a read command on the C/BE# signals.

** Start of address phase. **

CLK Cycle 3 – The initiator tri-states the address in preparation for the target driving read data. The initiator now drives valid byte enable information on the C/BE# signals. The initiator asserts IRDY# low indicating it is ready to capture read data. The target asserts DEVSEL# low (in this cycle or the next) as an acknowledgment it has positively decoded the address. The target drives TRDY# high indicating it is not yet providing valid read data.

CLK Cycle 4 – The target provides valid data and asserts TRDY# low indicating to the initiator that data is valid. IRDY# and TRDY# are both low during this cycle causing a data transfer to take place.

** Start of first data phase occurs, and the initiator captures the data. **

CLK Cycle 5 – The target deasserts TRDY# high indicating it needs more time to prepare the next data transfer.

CLK Cycle 6 – Both IRDY# and TRDY# are low.

** Start of next data phase occurs, and the initiator captures the data provided by the target. **

CLK Cycle 7 – The target provides valid data for the third data phase, but the initiator indicates it is not ready by deasserting IRDY# high.

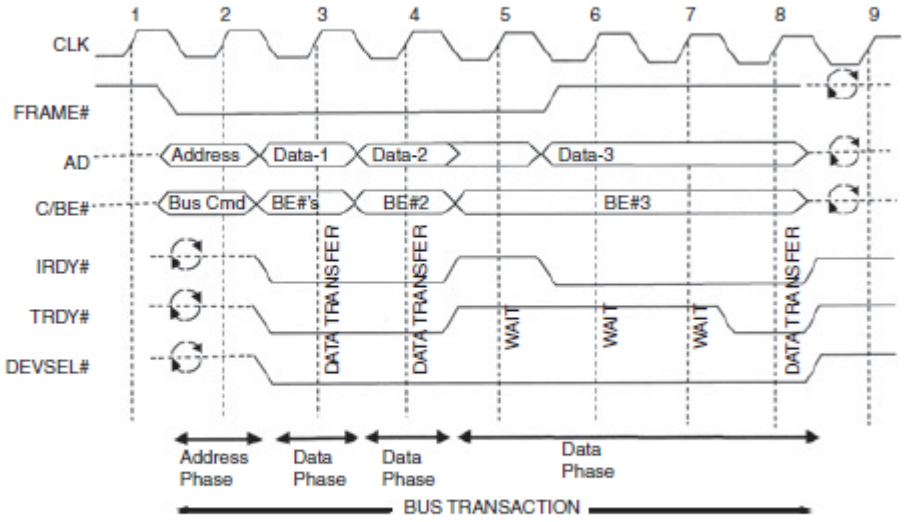
CLK Cycle 8 – The initiator re-asserts IRDY# low to complete the third data phase. The initiator drives FRAME# high indicating this is thelfl data phase (master termination).

** Final data phase occurs, the initiator captures the data provided by the target, and terminates. **

CLK Cycle 9 – FRAME#, AD, and C/BE# are tri-stated, as IRDY#, TRDY#, and DEVSEL# are driven inactive high for one cycle prior to being tri-stated.

4.3 Comunicación paralela: PCI

Escritura



CLK Cycle 1 – The bus is idle.

CLK Cycle 2 – The initiator asserts a valid address and places a write command on the C/BE# signals.

** Start of address phase. **

CLK Cycle 3 – The initiator drives valid write data and byte enable signals. The initiator asserts IRDY# low indicating valid write data is available. The target asserts DEVSEL# low as an acknowledgment it has positively decoded the address (the target may not assert TRDY# before DEVSEL#). The target drives TRDY# low indicating it is ready to capture data. Both IRDY# and TRDY# are low.

** First data phase occurs with target capturing write data. **

CLK Cycle 4 – The initiator provides new data and byte enables. Both IRDY# and TRDY# are low.

** Next data phase occurs with target capturing write data. **

CLK Cycle 5 – The initiator deasserts IRDY# indicating it is not ready to provide the next data. The target deasserts TRDY# indicating it is not ready to capture the next data.

CLK Cycle 6 – The initiator provides the next valid data and asserts IRDY# low. The initiator drives FRAME# high indicating this is the final data phase (master termination). The target is still not ready and keeps TRDY# high.

CLK Cycle 7 – The target is still not ready and keeps TRDY# high.

CLK Cycle 8 – The target becomes ready and asserts TRDY# low. Both IRDY# and TRDY# are low.

** Final data phase occurs with target capturing write data. **

CLK Cycle 9 – FRAME#, AD, and C/BE# are tri-stated, as IRDY#, TRDY#, and DEVSEL# are driven inactive high for one cycle prior to being tri-stated.

Tema 4

Buses industriales. (2+2 horas)

4.1 Protocolos
4.2 Comunicación serie
4.3 Comunicación paralela
4.4 Rendimiento del bus
4.5 Ejemplos

4.4 Rendimiento del bus

Ancho de banda: cantidad de datos que el bus puede transmitir por unidad de tiempo, Depende de:

1.- Diseño físico: nº líneas, longitud, dispositivos soportados

Cuanto más corto, menos dispositivos y más líneas de datos, es más rápido

2.- Protocolo: cuanto más sencillo mayor ancho de banda. Las transferencias modo ráfaga (burst) aumentan el ancho de banda pero añaden latencia

Split transaction: las líneas de datos se dejan durante el handshaking

Tema 4

Buses industriales. (2+2 horas)

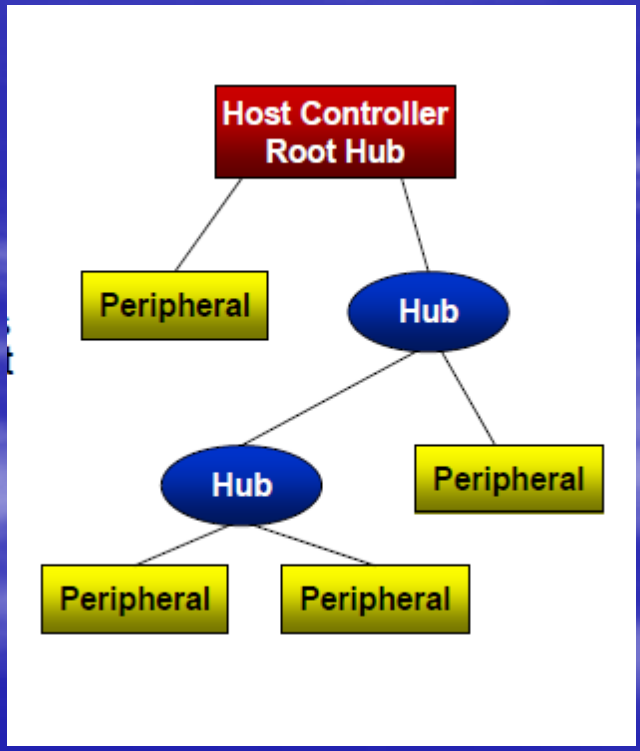
4.1 Protocolos
4.2 Comunicación serie
4.3 Comunicación paralela
4.4 Rendimiento del bus
4.5 Ejemplos: USB, SPI, CANBUS, JTAG

4.6 Ejemplos

Universal Serial Bus (USB):

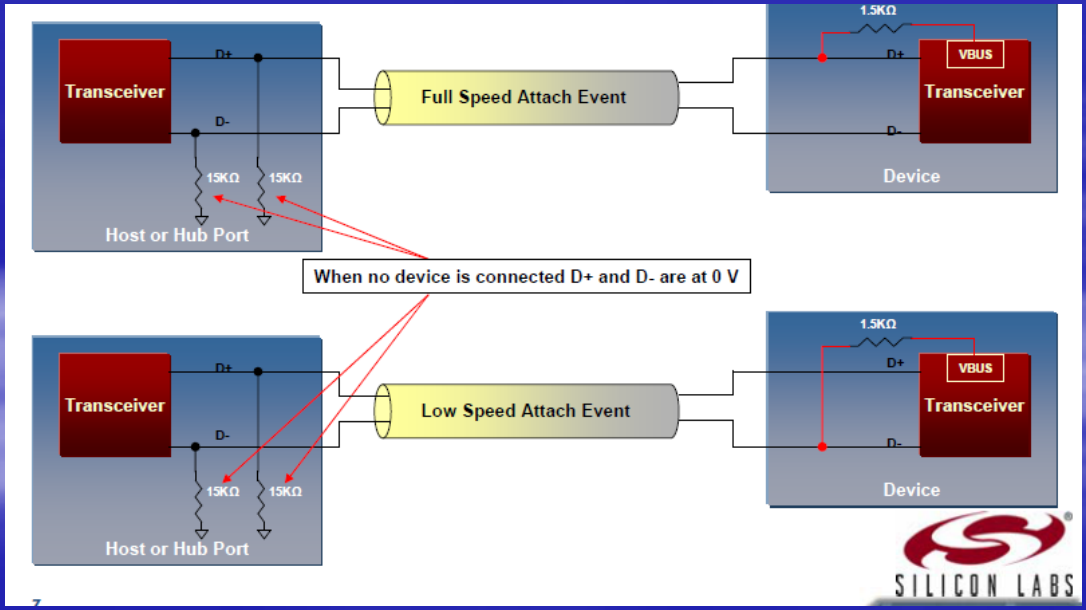
Es un bus de comunicación para conectar el PC o procesador a varios periféricos. El ancho de banda se configura para cada dispositivo (480Mbps)

- La topología es como un árbol con el host USB como raíz.
- La raíz ejecuta un driver de controlador USB responsable de controlar los dispositivos conectados al bus
- Los dispositivos pueden conectarse y desconectarse “on th fly”
- El host es responsable de identificar estos eventos y configurar el bus
- Cada dispositivo tiene un único id para transferir los datos.



4.6 Ejemplos

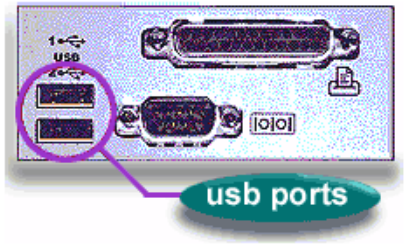
Cuando se conecta el dispositivo se le asigna un nº. La comunicación tiene lugar sobre pipes unidireccionales llamados endpoints (posiciones de memoria).
Uno o varios endpoints forman un interface (teclado, ratón, etc.)
Se usa protocolo half-duplex (full-duplex para 3.0) donde los datos se pasan a través de un interfaz de 2 cables

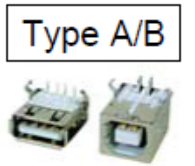


Al conectar un dispositivo se produce un attach event

4.6 Ejemplos

- Ease of Use
  - One interface for many devices
  - Hot pluggable
  - Automatic configuration
  - No power supply required
    - Devices can pull up to 500 mA from the bus
- Reliability
  - Lossless data transfers
- Speed
  - Three transfer speeds
    - Low Speed – 1.5 Mbps (USB 1.1 and 2.0)
    - Full Speed – 12 Mbps (USB 1.1 and 2.0)
    - Hi-Speed – 480 Mbps (USB 2.0 only)
- Low Power Consumption
  - Suspend mode
    - Devices consume 500 uA or less (USB 2.0)
    - Devices consume 2.5 mA or less (USB 3.0)
- Availability
  - Microsoft and Intel's PC 2001 System Design Guide requires that all new PC's have two user-accessible USB ports





SILICON LABS

4.6 Ejemplos

Los dispositivos se clasifican en:

1.- Almacenamiento: discos duros, flash
2.- Interfaz humano: ratón, teclado
3.- Hub: Para extensiones
4.- Comunicaciones: tarjetas de red, modems.

4.6 Ejemplos

La transferencia es el nivel más alto del protocolo.

Tipos de transferencias:

1.- Control: todo dispositivo debe tener un endpoint para configuración (endpoint 0)

2.- Interrupciones: para datos de baja frecuencia

3.- Bulk: grandes cantidades de datos, ej. impresora

4.- Isocronas: periódicas y continuas (video y audio)

Las transferencias se dividen en transacciones, que a su vez se dividen en paquetes. Una transacción suele constar de al menos 3 paquetes

4.6 Ejemplos

Cada paquete lleva un PID (Packet Identifier Structure)

PID Type	PID Name	PID Value <3:0>
Token	OUT	0001b
	IN	1001b
	SOF	0101b
	SETUP	1101b
Data	DATA0	0011b
	DATA1	1011b
Handshake	ACK	0010b
	NAK	1010b
	STALL	1110b

PID Format

PID0	PID1	PID2	PID3	$\overline{PID0}$	$\overline{PID1}$	$\overline{PID2}$	$\overline{PID3}$

IN – data transfers to the host
OUT – data transfers from the host
SOF – Timing marker at 1mS
Setup – Specifies control transfers

Data0 – data transfer with data toggle clear
Data1 – data transfer with data toggle set

ACK – data received without error
NAK – Device busy or no data available
Stall – Unsupported control request, control request failed, or endpoint failed

4.6 Ejemplos

Token Packet format:

Field	PID	Address	Endpoint	CRC
Bits	8	7	4	5

SOF Packet format:

Field	PID	Frame Number	CRC
Bits	8	11	5

Data Packet format:

Field	PID	Data	CRC
Bits	8	0-1023	16

Handshake Packet format:

Field	PID
Bits	8

4.6 Ejemplos

Transfer Type	Stages (Transactions)	Phases (Packets)	Comments
Control	Setup	Token	♦Enables host to read configuration information, set addresses and select configurations ♦Only transfer that is required to be supported by peripherals ♦Has both IN and OUT transfers to a single endpoint
		Data	
		Handshake	
	Data (IN or OUT) (optional)	Token	
		Data	
		Handshake	
	Status (IN or OUT)	Token	
		Data	
		Handshake	
Bulk	Data (IN or OUT)	Token	♦Non-critical data transfers ♦Bandwidth allocated to the host ♦Good for file transfer where time critical data is not required
		Data	
		Handshake	
Interrupt	Data (IN or OUT)	Token	♦Periodic transfers on the time base conveyed during enumeration ♦Host guarantees attention before this elapsed time
		Data	
		Handshake	
Isochronous	Data (IN or OUT)	Token	♦Guaranteed delivery time of packets for data streaming ♦No-retransmitting of data allowed
		Data	