

7919 | **Sistemas Embebidos**

2º Cuatrimestre de 2017

CLASE 11: INTERFACES DE RED Y DEBUG

Prof: José H. Moyano

Autor original: Sebastián Escarza

Dpto. de Cs. e Ing. de la Computación
Universidad Nacional del Sur
Bahía Blanca, Buenos Aires, Argentina



GrIDSE

The background of the slide is a detailed, high-resolution image of a printed circuit board (PCB). It shows a complex network of copper traces, various electronic components like integrated circuits, capacitors, and resistors. The image is slightly faded and serves as a technical backdrop for the text.

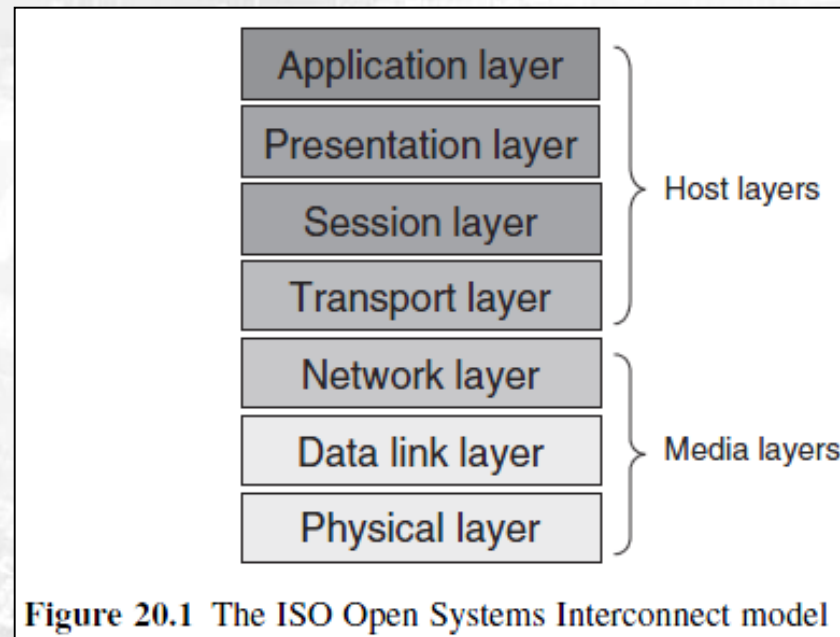
Interfaces de red

Interfaces de red

- Hemos visto un conjunto representativo de interfaces de comunicación.
- Sin embargo, el escenario es más amplio:
 - múltiples medios (cable, radio, IR, fibra óptica).
 - las conexiones no necesariamente son permanentes.
- Un entorno de red tiene una infraestructura mayor en la que se debe:
 - determinar cómo se formatean e interpretan los datos
 - establecer mecanismos de direccionado
 - fijar mecanismos de detección y corrección de errores.

Interfaces de red

- **Protocolos de red:**
 - normas que definen los acuerdos entre los nodos de la red relativas a los diversos aspectos para establecer la comunicación.



The background of the slide is a detailed, high-resolution image of a printed circuit board (PCB). It features a complex network of copper traces, various electronic components like integrated circuits, capacitors, and resistors. The image is slightly desaturated, giving it a technical and professional appearance. A dark blue horizontal band is superimposed over the middle of the image, containing the title text.

Conectividad por radio

Interfaces de red - Radio

- La comunicación puede darse a través de objetos no conductores.
- A baja potencia se pueden lograr conexiones locales.
- Más propenso a interferencias: los protocolos definen mecanismos específicos para evitar el acople de canales de comunicación, etc.
- Analizaremos dos estándares:
 - Bluetooth
 - Zigbee

Interfaces de red - Bluetooth

- Estándar abierto controlado por el Bluetooth Special Interest Group.
- Opera en el rango de 2.402 a 2.480 Ghz (ISM).
- Provee enlaces de radio de baja potencia.
- Alcance teórico de 1, 10 y 100 metros.
- Tasa de transferencia: 1Mbps (3Mbps BT 2.0).
- Hasta 8 dispositivos por red (1 master y 7 slaves).
- Usa spread-spectrum frequency hopping: el transmisor cambia la frecuencia de manera pseudo-aleatoria 1600 veces por segundo.

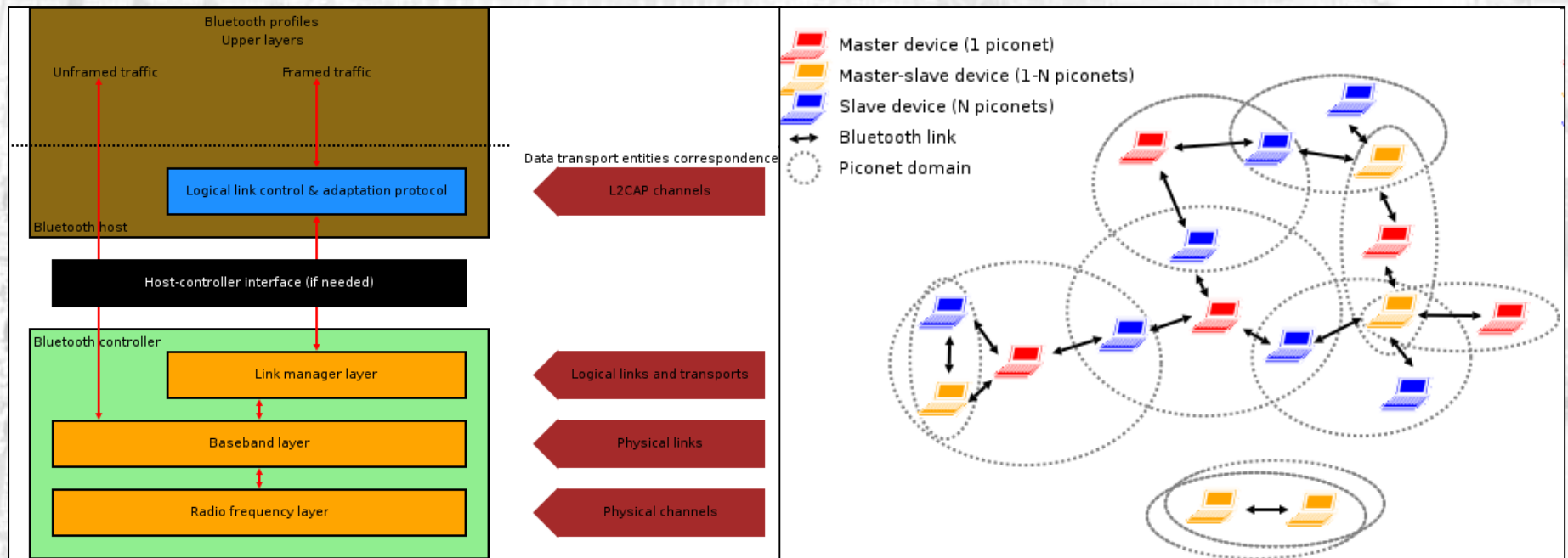
Interfaces de red - Bluetooth

- Cada dispositivo tiene una dirección, una clase y una lista de servicios (permite filtrar dispositivos de interés p/establecer comunicación).
- Los dispositivos se conectan sin intervención del usuario (sincronizan sus cambios de frecuencia) formando mini redes (piconets), las cuales pueden coexistir en un mismo ambiente (cada una cambia según su patrón).
- En caso de colisiones y corrupción de datos, el sistema es capaz de detectar mediante software esta condición y rechazar los datos erróneos.

Interfaces de red - Bluetooth

- **El protocolo Bluetooth:**
 - permite configurar redes de dispositivos de manera autónoma.
 - provee varios mecanismos de corrección de errores.
- **Bluetooth define un stack de protocolos variable:**
 - protocolos obligatorios
 - protocolos para distintos tipos de enlaces
 - protocolos de control de telefonía
 - protocolos adoptados
- **La infraestructura es compleja y el costo de implementación elevado para sistemas simples.**

Interfaces de red - Bluetooth



Interfaces de red - Zigbee

- **Estándar administrado por la Zigbee Alliance.**
- **Es más simple y económico que Bluetooth:**
 - menos requerimientos de software
 - tasas de transferencia menores
 - consumo más bajo
- **Tres tipos de nodos:**
 - **Coordinador Zigbee:** cada red tiene uno. Puede comunicarse con otras redes (el más poderoso).
 - **Full Function Device:** puede pasar datos a otros dispositivos (rutear información dentro de la red).
 - **Reduced Function Device:** puede comunicarse con la red (el más simple).

Interfaces de red - Zigbee

- A menor capacidad en el nodo, menor memoria y poder de cómputo requiere (menor costo).
- Los slaves pueden dormir la mayor cantidad de tiempo posible y activarse cuando sea necesario, dependiendo si la red es con o sin beacon.

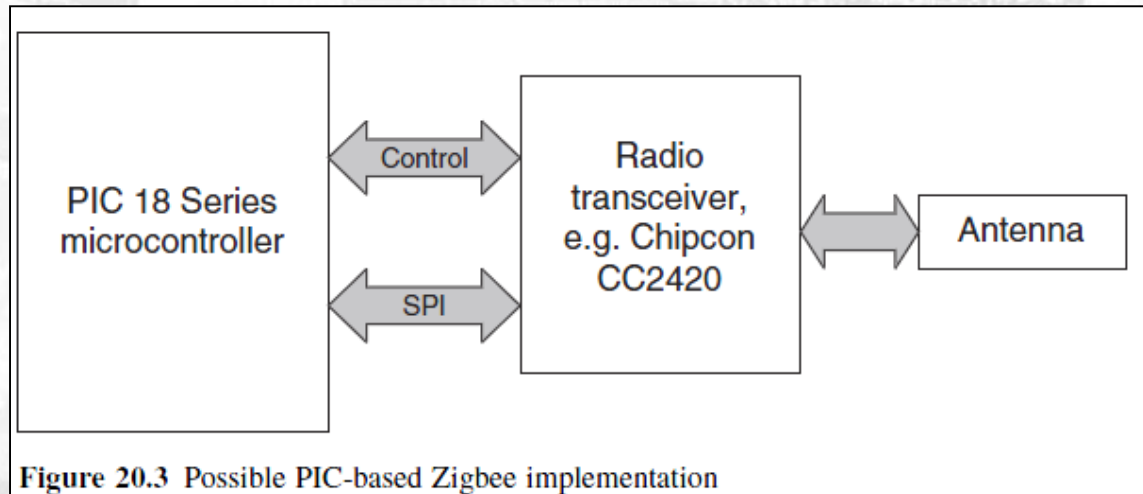


Figure 20.3 Possible PIC-based Zigbee implementation

The background of the slide is a detailed, high-resolution image of a printed circuit board (PCB). It shows various electronic components such as integrated circuits, capacitors, and resistors, along with intricate copper traces. The image is slightly faded and serves as a technical backdrop for the title.

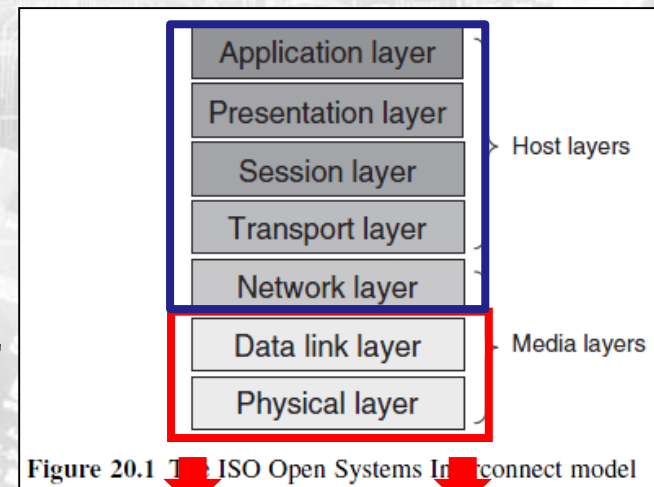
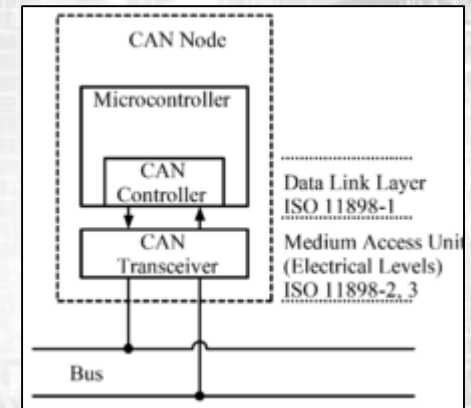
Redes de dispositivos

Interfaces de red – CAN/LIN

- **Respecto a comunicación serie habíamos visto:**
 - SPI
 - I²C
 - Comunicación asincrónica
- **Ninguno de ellos es tolerante a fallas.**
- **Los estándares CAN y LIN se utilizan en escenarios específicos donde se requiere de alta confiabilidad en las comunicaciones.**

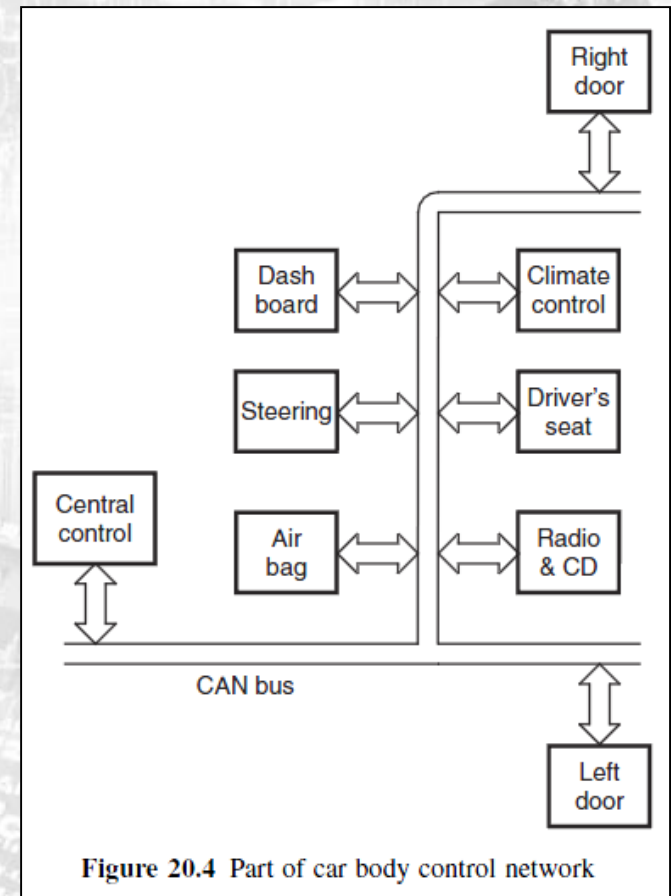
Interfaces de red – CAN

- **Controller Area Network (CAN):**
 - Estándar propuesto por Bosch para subsistemas en automóviles:
 - interferencia electromagnética
 - amplio rango de operación (temperatura, humedad, etc.)
 - Serie asincrónico multimaestro
 - Alta confiabilidad.
 - Protocolo complejo.
 - Aborda sólo las dos capas inferiores del modelo OSI.
 - Soporte en capas superiores mediante estándares derivados.



Interfaces de red – CAN

- **Controller Area Network (CAN):**
 - Cantidad ilimitada de nodos.
 - Los nodos no tienen direcciones. Aplican filtrado de mensajes para saber si el mensaje en el bus es relevante y debe ser leído.
 - Transferencia de datos en frames complejos.
 - Chequeo de errores exhaustivo. Si un nodo detecta que está fallando, puede desconectarse de la red.
 - Protocolo usualmente manejado vía controladores y librerías.



Interfaces de red – CAN

- **Controller Area Network (CAN):**
 - Comunicación asincrónica con tasa fija de transf.
 - Lógica dominante-recesiva en el bus (de mínima).
 - Mecanismo de arbitraje:
 - privilegia al mensaje de mayor prioridad
 - si un nodo detecta que su mensaje ha sido corrompido y tiene menor prioridad, deja de transmitir y espera.
 - no se pierde tiempo, ni se pierde prioridad.

Truth tables for dominant/recessive and logical AND

Bus state with two nodes transmitting

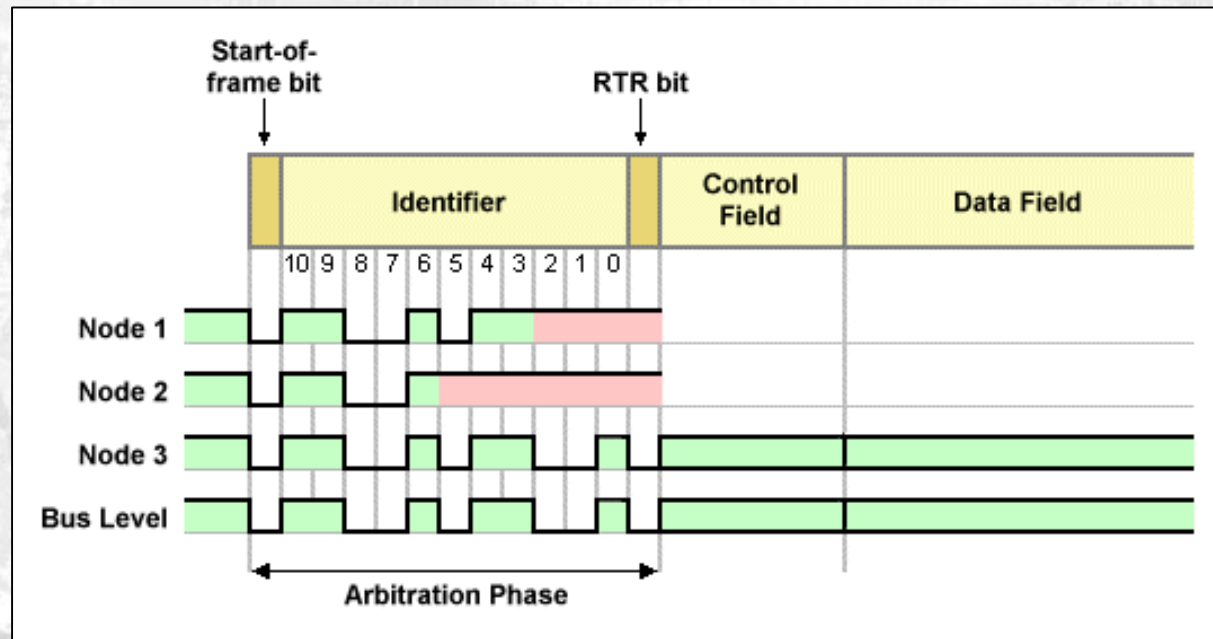
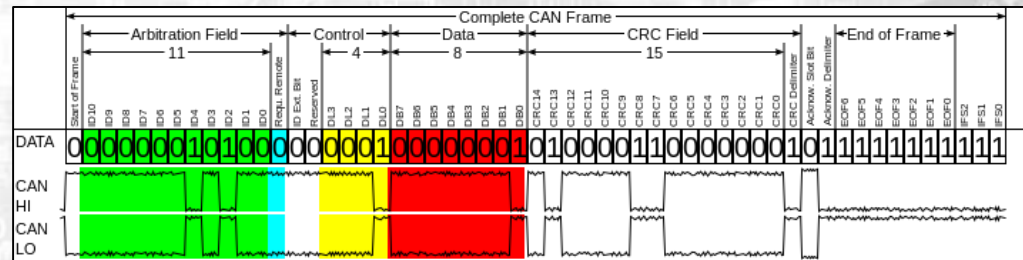
Logical AND

	dominant	recessive
dominant	dominant	dominant
recessive	dominant	recessive

	0	1
0	0	0
1	0	1

Interfaces de red – CAN

- **CAN: Arbitraje de bus**



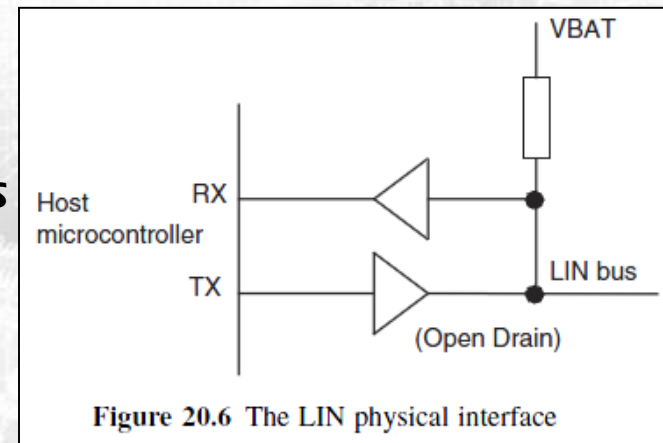
Interfaces de red – LIN

- **CAN es muy confiable, pero complejo y costoso para dispositivos simples.**
- **Local Interconnect Network (LIN):**
 - **Alternativa más económica, pequeña y lenta que CAN. Interopera con esta última.**
 - **Topología de red fija.**
 - **Único nodo master (mayor poder de procesamiento) y los demás nodos son esclavos (nodos simples o hardware dedicado).**
 - **El master inicia las comunicaciones y sólo un slave o el mismo maestro puede contestar. No hay mecanismo de gestión de colisiones.**
 - **Se incluye una API en C como parte del estándar.**

Interfaces de red – LIN

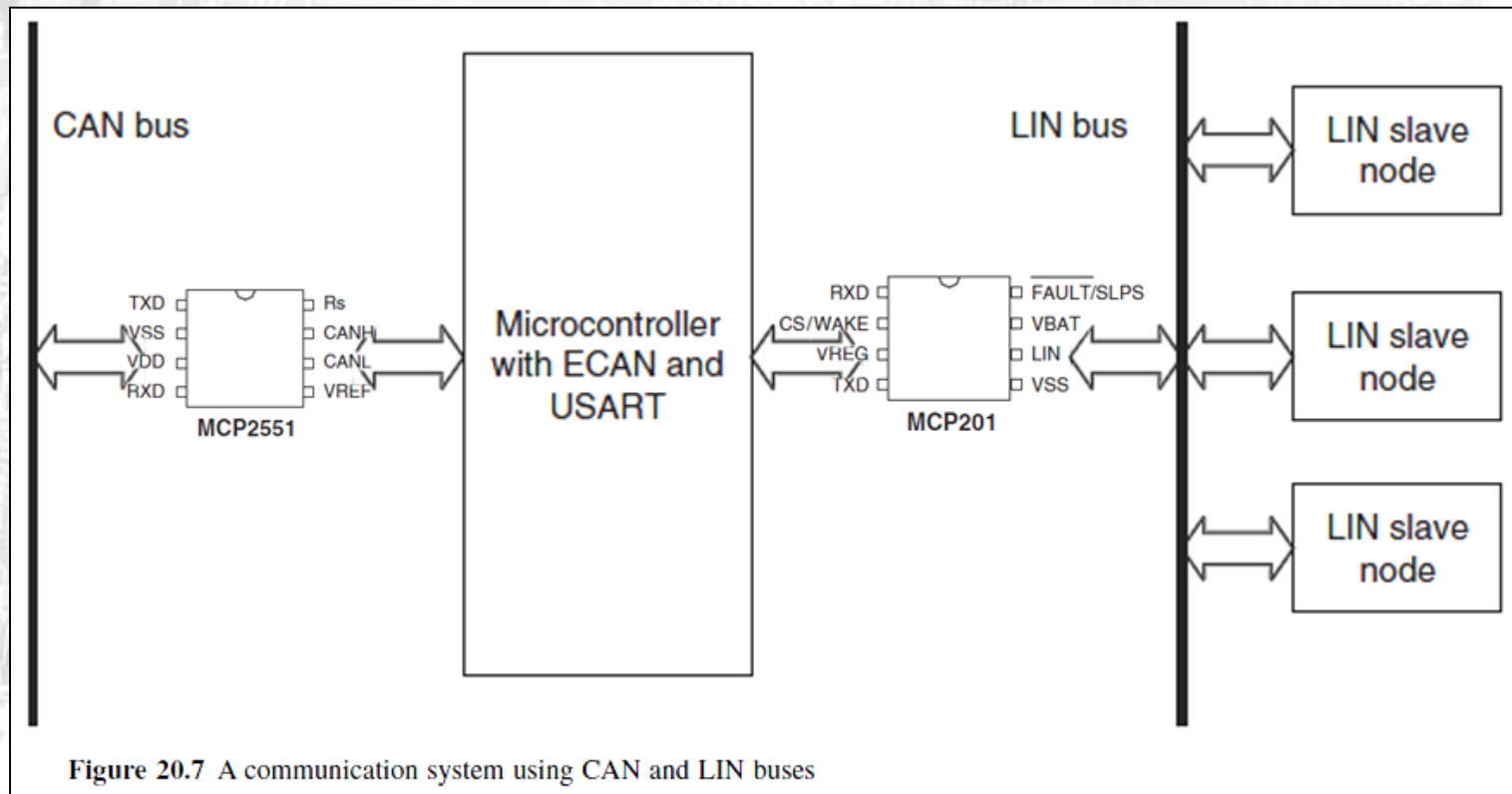
- **Local Interconnect Network (LIN):**

- **Bus:** un único cable (open drain).
- **Frame:** header + response.
- **Header (a cargo del master):**
 - **Break field:** alerta a slaves acerca de un nuevo mensaje
 - **Sync byte:** para que los slaves sincronicen sus relojes
 - **Id:** identifica el slave y la acción a realizar.
- **Response (a cargo del master o slave):**
 - hasta 8 bytes + checksum.



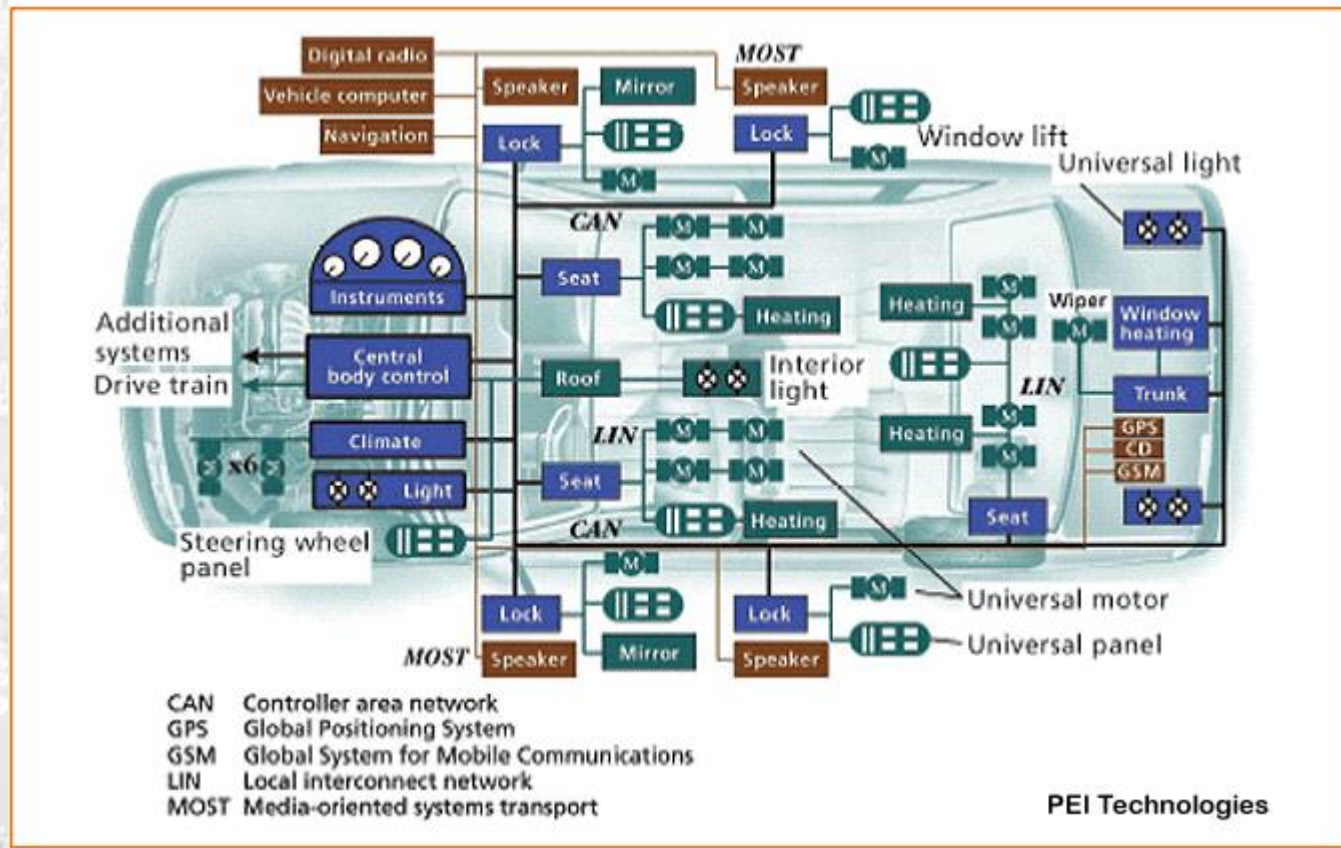
Interfaces de red – CAN/LIN

- Ej: interface CAN/LIN (CAN–ECAN<->USART–LIN).



Interfaces de red – CAN/LIN

- Ej. CAN/LIN

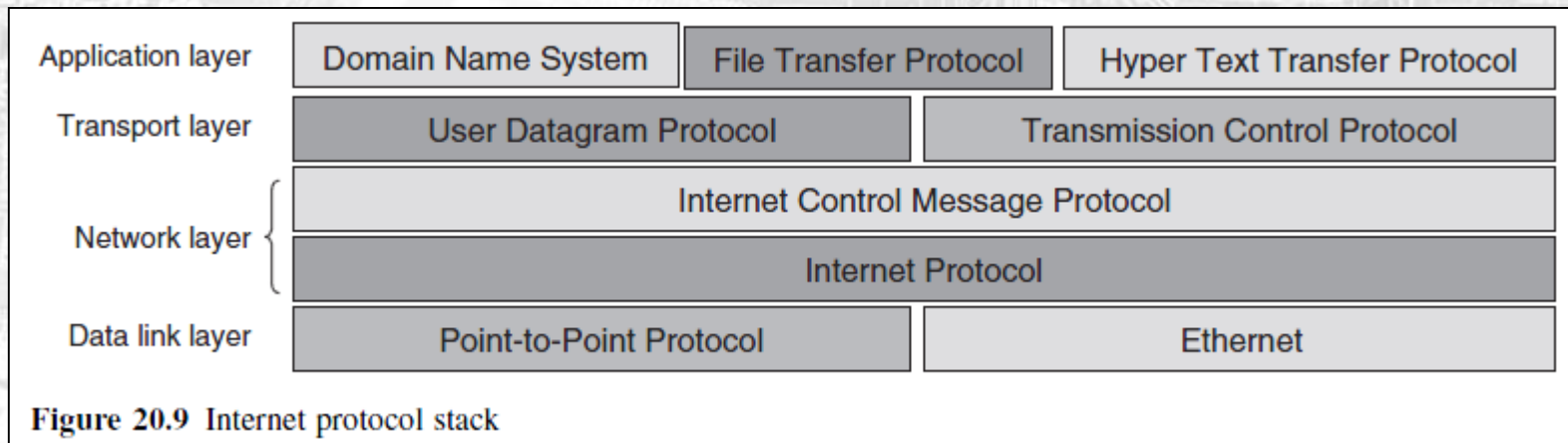




Interfaces de Internet

Interfaces de red – Internet

- La conexión a Internet de sistemas embebidos es cada día más habitual:
 - interfaces de control
 - transferencia de datos
 - IoT (Internet of Things)
- Varios protocolos interactúan para dar soporte:

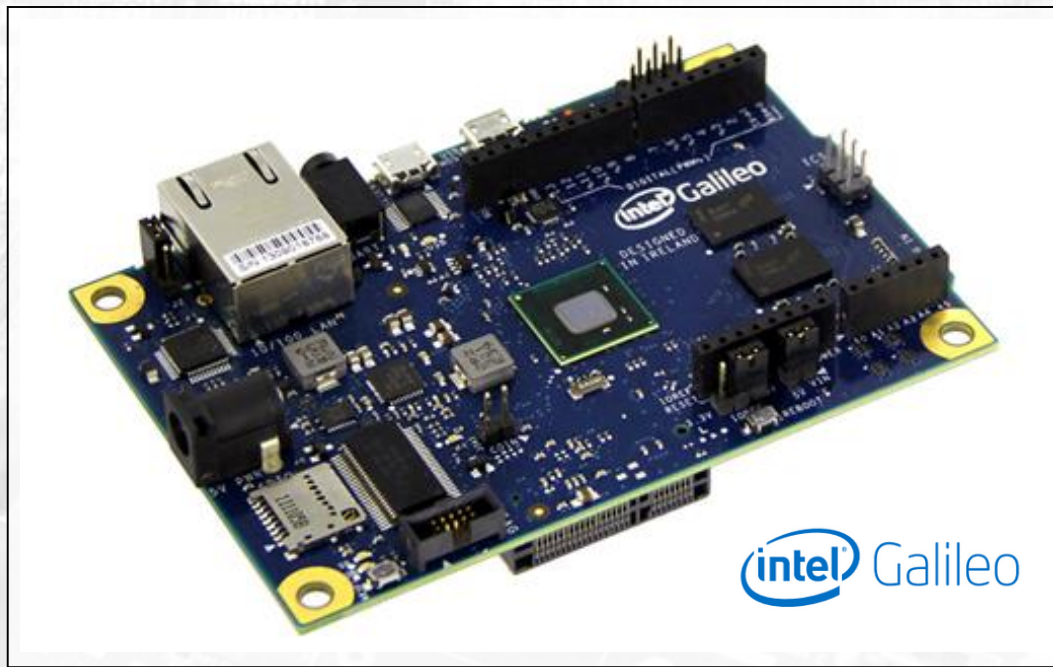


Interfaces de red – Internet

- **Dada la complejidad existente en este tipo de contextos el manejo de protocolos de internet en Sistemas Embebidos se realiza mediante:**
 - los servicios del sistema operativo (si se utiliza uno).
 - un conjunto de librerías que facilitan la utilización de los servicios de cada una de las capas (protocol stack).
 - el uso de microcontroladores o dispositivos que cuentan con la capacidad de procesar las tramas y paquetes en hardware.
- **Se suelen ver implementaciones embebidas de servidores Web, DNS, FTP, etc.**

Interfaces de red – Internet

- Intel Galileo Gen 1:
 - TCP/IP, UDP/IP, SSH, Telnet, etc.
 - Posibilidad de abrir sockets (Embedded Linux)



The background of the slide is a grayscale, high-magnification photograph of a printed circuit board (PCB). It shows a dense array of electronic components, including integrated circuits, capacitors, and various connectors. The board's surface is covered with intricate patterns of copper traces and solder points. A dark blue horizontal band is superimposed across the middle of the image, serving as a backdrop for the title text.

Interfaces de Debug

Interfaces de debug

- **Habíamos visto tres alternativas para depurar un sistema embebido:**
 - host based debugging (simulación)
 - debuggers remotos y kernels de debugging (software de debug en el target)
 - hardware de debugging
 - ICEs: hardware que se añade al target
 - Interfaces de debugging: hardware que forma parte del target
- **A continuación nos centraremos en la última alternativa: las interfaces de debug.**

Interfaces de debug

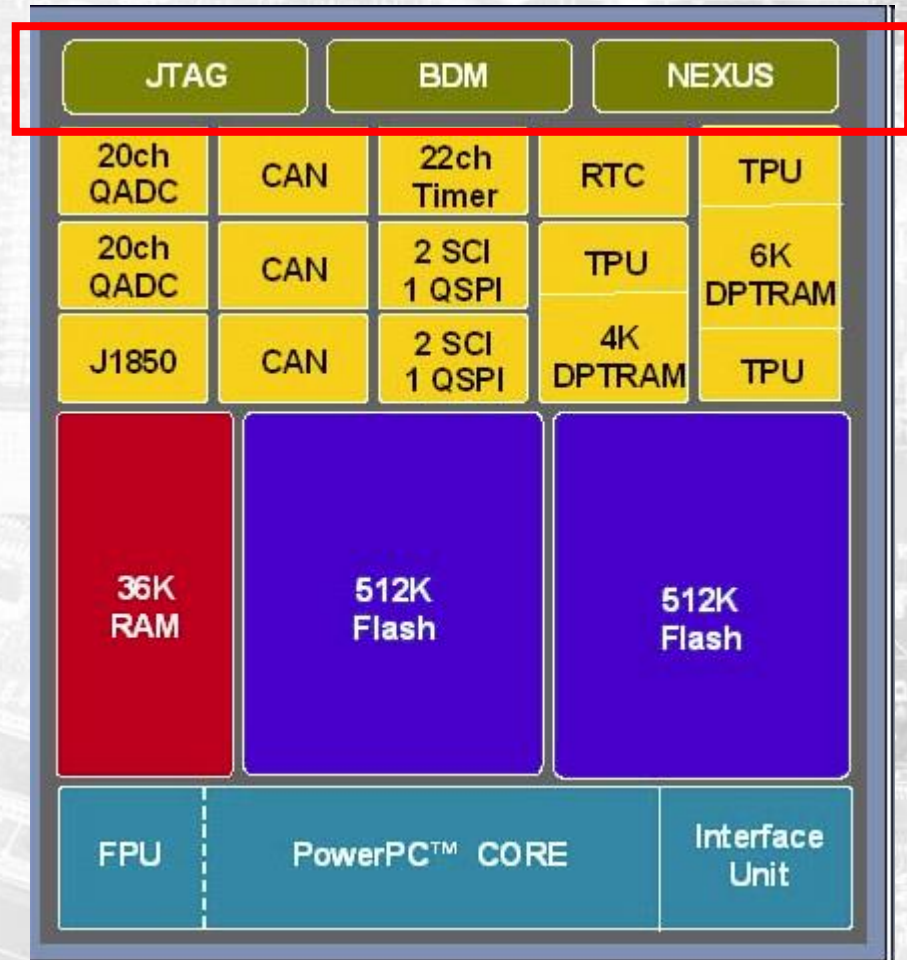
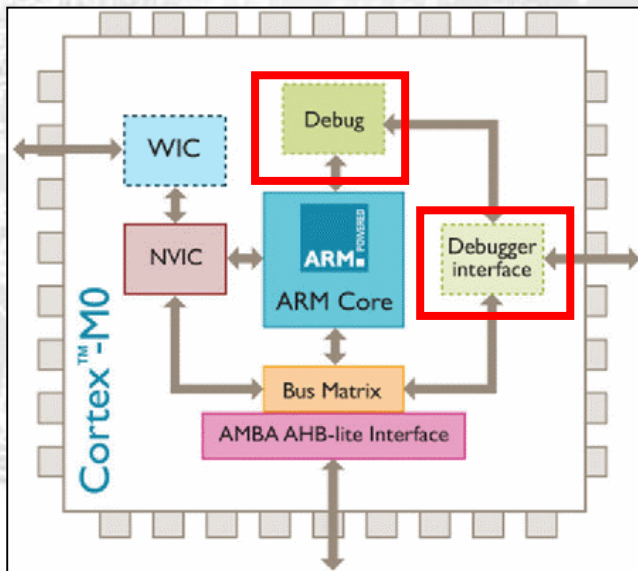
- **Ventajas con respecto a los kernels de debug:**
 - no hay que añadir software adicional en el target
 - no se requiere un funcionamiento mínimo que garantice que el kernel se ejecuta libre de errores para debuggear
 - no existe riesgo de sobrescribir el kernel de debug
- **Ventajas con respecto al hardware de debug:**
 - no hay que añadir hardware (no siempre posible en SoCs)
 - no hay problemas de visibilidad de pipelines y caches (logic analyzers).
- **Existe una tendencia cada vez mayor a incorporar circuitería de debug en los procesadores centrales a partir de cierto rango de prestaciones.**

Interfaces de debug

- Se incorpora circuitería dedicada en el target.
 - implementa una suerte de kernel de debug en hardware.
- Se dispone de un pinout estándar (útil para conectar diversos dispositivos de debug que adopten el estándar utilizado).
- No se modifica el sistema (ni en software, ni en hardware). Depuración en condiciones “release”.
- Veremos 3 estándares:
 - Background Debug Mode (BDM)
 - Joint Test Action Group (JTAG)
 - Nexus

Interfaces de debug

- Ej: MPC565
32 Bit Microcontroller
- Ej: ARM Cortex-M0



The background of the slide is a grayscale, high-contrast image of a printed circuit board (PCB). It shows various electronic components such as integrated circuits, capacitors, and resistors, along with the intricate network of copper traces. The image is slightly blurred and has a grainy texture, giving it a technical and industrial feel. A dark blue horizontal band is superimposed over the middle of the image, containing the title text.

Interfaces de Debug - BDM

Interfaces de debug - BDM

- **Background Debug Mode – BDM**
 - Estándar propietario de Motorola (Freescale).
 - Un conector dedicado (pinout estándar). El hardware provee las señales de debug a dicho conector.
 - Dispositivo n-wire/wiggler se conecta y realiza la interfaz con el host.



Interfaces de debug - BDM

- **Background Debug Mode – BDM**
 - Mucho más económico que un ICE.
 - Muchos ICEs se valen de interfaces estándares para conectarse al Target.

- **BDM pinout (se transmite en tiempo real):**
 - PSTX: Processor Status
 - DDATA: Debug Data

RESERVED	1	2	BREAKPOINT
GROUND	3	4	DSCLK
GROUND	5	6	RESERVED
RESET	7	8	DSI
+5 Volts	9	10	DSO
GROUND	11	12	PST3
PST2	13	14	PST1
PST0	15	16	DDATA3
DDATA2	17	18	DDATA1
DDATA0	19	20	GROUND
RESERVED	21	22	RESERVED
GROUND	23	24	CLK_CPU
VCC_CPU	25	26	TEA

Figure 7.2: Pinout for the Motorola BDM debug interface.

Interfaces de debug - BDM

- BDM Processor status codes (PST3 – PST0):
- Diseñados para:
 - debugger en host que tiene conocimiento de la imagen del programa a debuggear.
- Instrucciones especiales de debug en el I.Set:
 - PULSE: PSTX = 0100.
 - WDDATA: Sacar datos por DDATA0-3.

PST3 - PST0	Definition
0000	Continue execution
0001	Begin execution of an instruction
0010	Reserved
0011	Entry into user mode
0100	Begin execution of PULSE or WDDATA instruction
0101	Begin execution of taken branch
0110	Reserved
0111	Begin execution of RTE instruction
1000	Begin 1-byte transfer on DDATA
1001	Begin 2-byte transfer on DDATA
1010	Begin 3-byte transfer on DDATA
1011	Begin 4-byte transfer on DDATA
1100	Exception processing (asserted for multiple cycles)
1101	Emulator-mode entry exception processing (as above)
1110	Processor is stopped, waiting for interrupt (as above)
1111	Processor is halted (as above)

Figure 7.3: Processor codes output.
Status signals output through the BDM debug core.

Interfaces de debug - BDM

- **Instrucciones especiales de debug en el I.Set:**
 - **WDDATA:** Permite transmitir datos al host (por ej. para incorporar watches).
- **BDM define una serie de comandos para controlar el proceso de debug. Dichos comandos:**
 - se envían y ejecutan directamente en el procesador central del target
 - tienen impacto en la ejecución del programa en el target (por ej. robo de ciclos, detención del procesador)

The background of the slide is a grayscale, high-resolution photograph of a printed circuit board (PCB). The image shows a dense array of electronic components, including integrated circuits, capacitors, and resistors, interconnected by a complex network of fine copper traces. The perspective is slightly angled, providing a three-dimensional feel to the flat board. A dark blue horizontal band is superimposed over the middle of the image, serving as a backdrop for the title text.

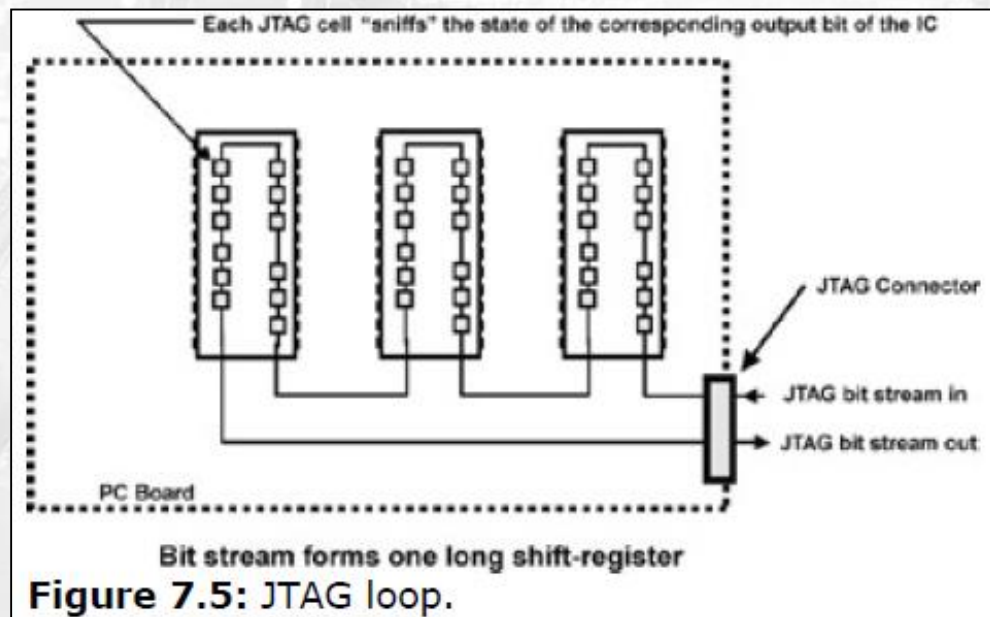
Interfaces de Debug - JTAG

Interfaces de debug - JTAG

- **Joint Test Action Group - JTAG**
 - **Protocolo IEEE 1149.1. Evolucionó a partir del testeo en la industria de placas madres de PC.**
 - **Standard Test Access Port and Boundary-Scan Architecture.**
 - **El testeo de placas se realizaba en máquinas especializadas con arreglos muy densos de puntos de contacto.**
 - **conectan cada nodo de la placa**
 - **detección de cortos, conexiones erróneas entre nodos, circuitos abiertos, etc.**
 - **El diseño de JTAG conecta cada nodo del circuito a un bit de un registro de desplazamiento (serializa el acceso a los nodos de la placa). Cientos o miles de bits por stream serial.**

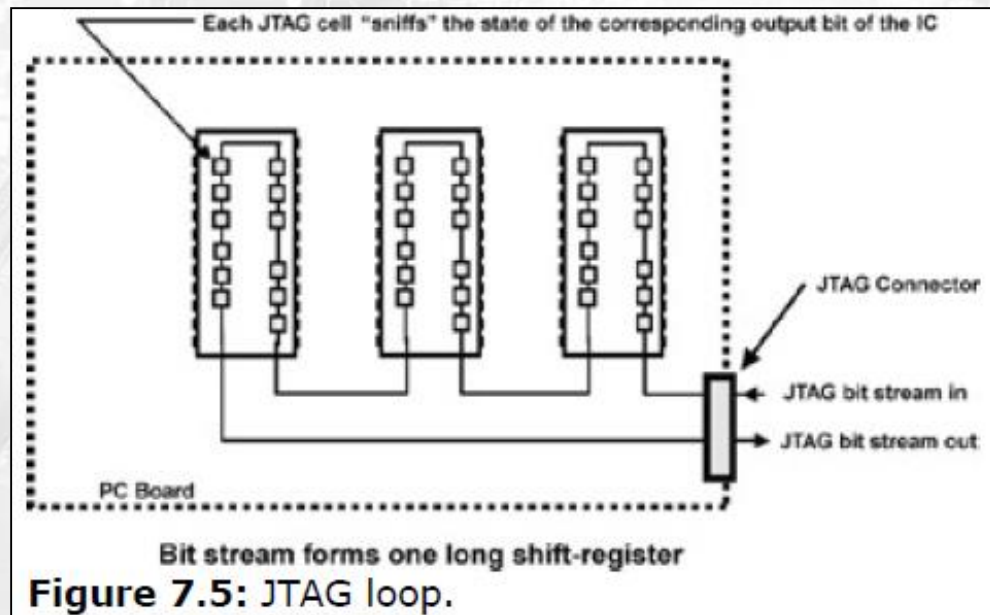
Interfaces de debug - JTAG

- El target añade circuitería específica para JTAG:
 - cada nodo a testear es interfaceado a un bit en el stream.
 - lógica de control para el desplazamiento entre las celdas JTAG de cada nodo/pin.



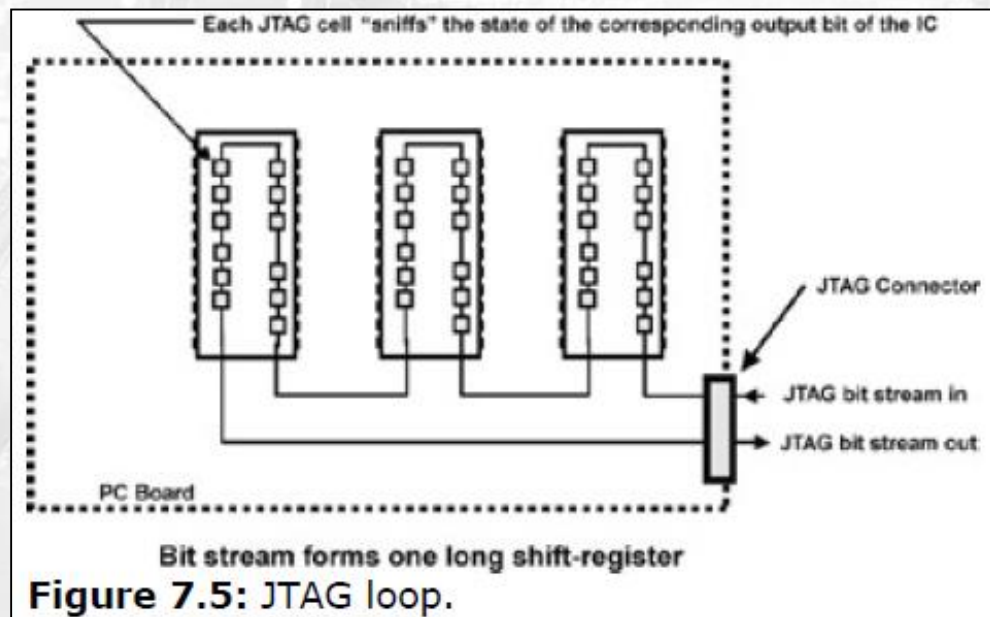
Interfaces de debug - JTAG

- El estado del sistema puede muestrearse por completo en un instante. Luego se desplazan los valores obtenidos para ser recuperados en serie por la interface.



Interfaces de debug - JTAG

- Así como el estado de un pin puede ser leído por la celda JTAG asociada, es posible establecer el valor de un pin a partir de la misma (JTAG stream in).



Interfaces de debug - JTAG

- Pines JTAG (relativamente pocos – interface serial)
 - TCK: Señal de clock
 - TMS: Test mode select (selecciona el estado de una máquina de estados que implementa el controlador).
 - TDI/TDO: La entrada/salida serial del protocolo
 - TRST: Reset.

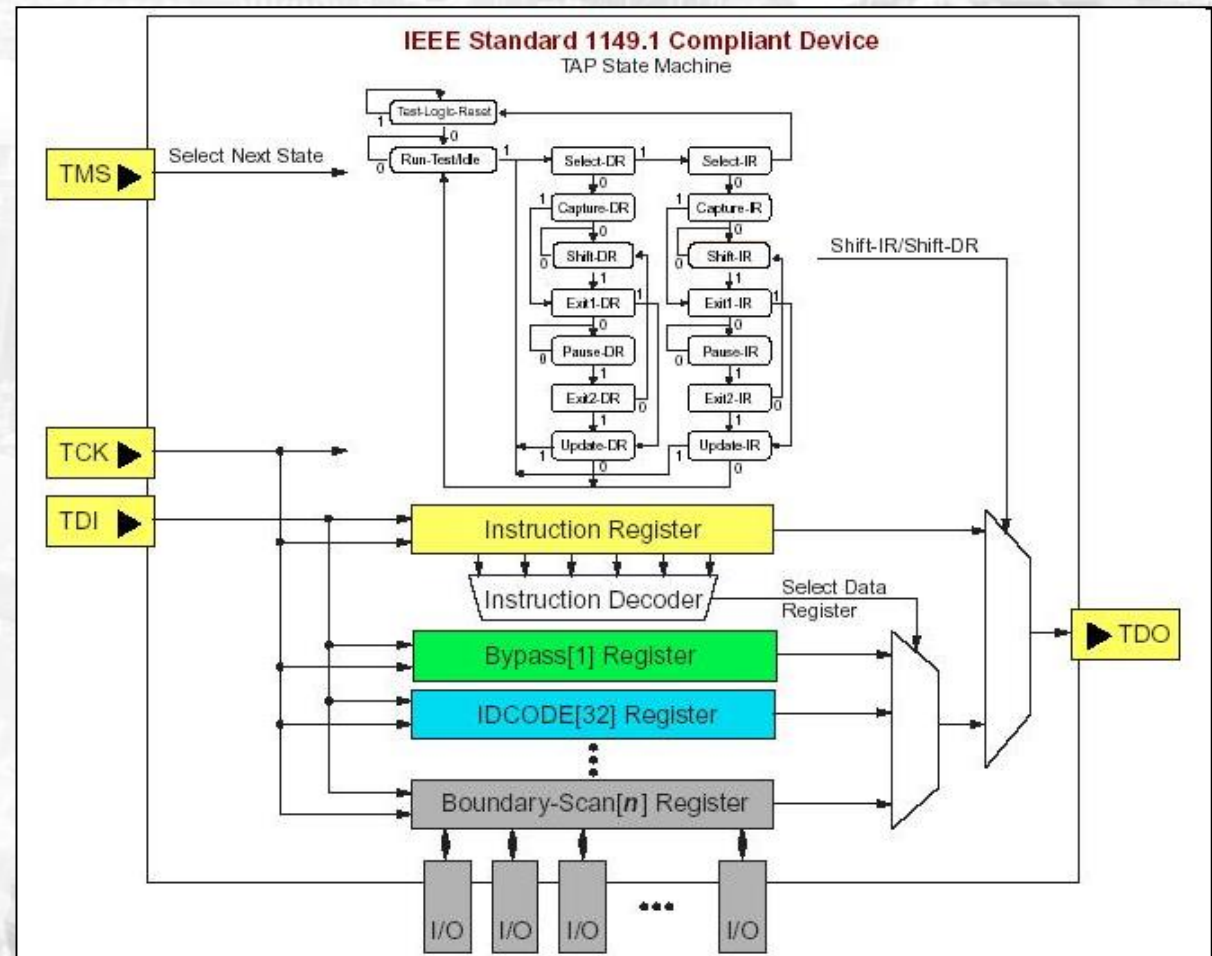
PIN	Description
TCK	A clock input that synchronizes the JTAG port logical operations
TMS	A test mode select input that is sampled on the rising edge of TCK to sequence the internal state machine controller (TAP Controller)
TDI	The input test data stream that is sampled on the rising edge of TCK
TDO	The output test data stream that updates on the falling edge of TCK
TRST	An active low asynchronous reset

Figure 7.7: Pin descriptions.

Pin descriptions for the IEEE 1149.1 (JTAG) interface.

Interfaces de debug - JTAG

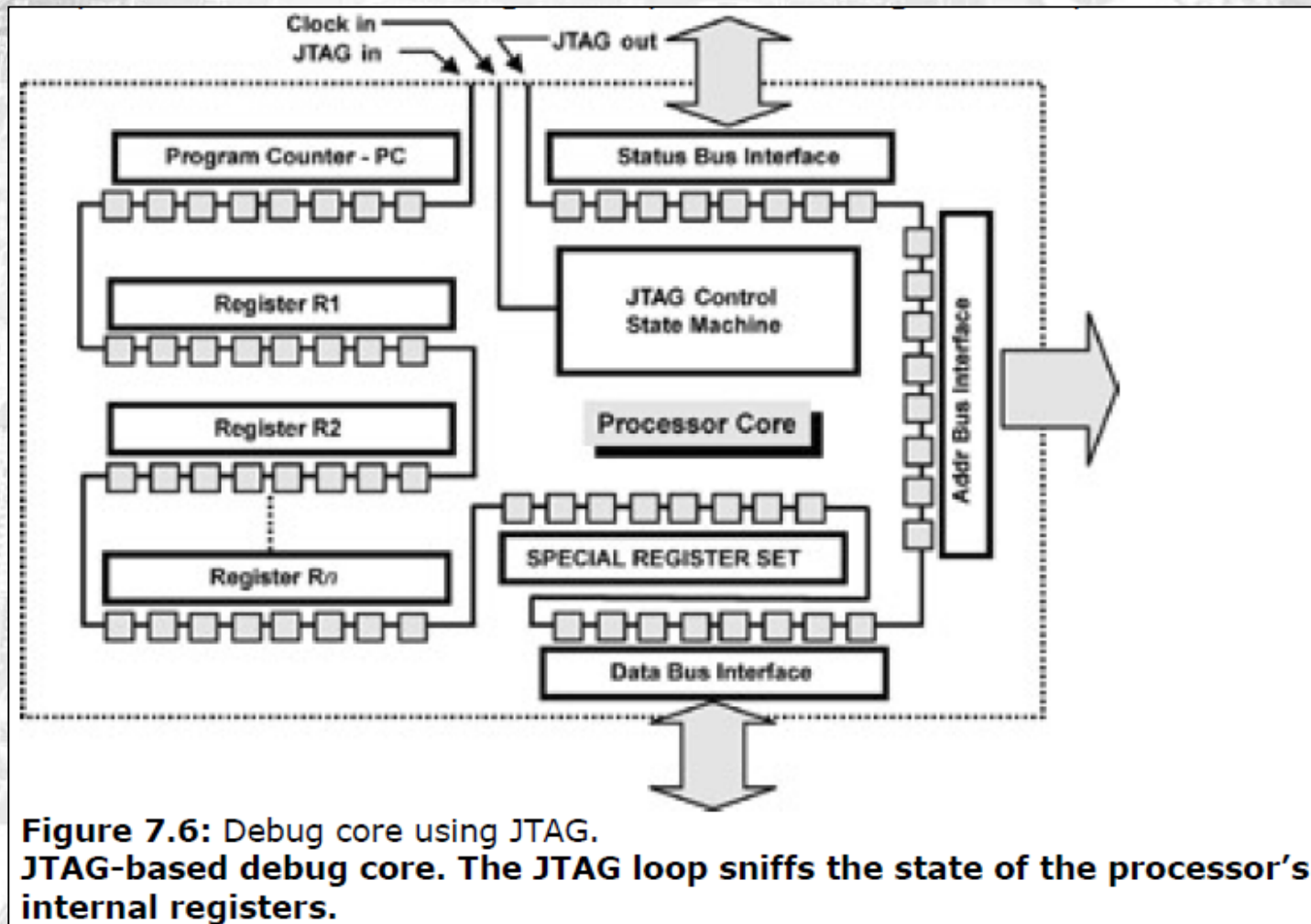
- Pines del protocolo:



Interfaces de debug - JTAG

- **JTAG:**
 - Algoritmos complejos para establecer bits.
 - Puede ser lento el proceso (para cada lectura o escritura hay que hacer N shifts para posicionar cada valor en la celda que corresponde).
 - Más económico que cablear cada uno de los nodos de una placa manualmente (bed of nails testers).
 - El mismo principio de las placas se aplica en SoCs. Deteniendo el clock del CPU, se puede obtener una instantánea del estado del procesador, registros, etc... o modificarlos.

Interfaces de debug - JTAG



Interfaces de debug - JTAG

- **Mejoras surgidas con el tiempo:**
 - **Comandos JTAG:** Se usa el loop de JTAG para introducir comandos de debug convencionales que son interpretados por una lógica interna (streams más cortos). Menos pines que BDM.
 - **Loops direccionables:** Se introducen múltiples loops más pequeños en lugar de uno sólo grande. Con un stream corto, se selecciona el loop a manipular y luego se opera normalmente (pero los streams son más cortos y los tiempos se reducen).
 - mejora la depuración de elementos individuales del SoC (por ej. múltiples cores o dispositivos).
 - requiere una lógica que gestione el direccionado y habilitación de los loops.

Interfaces de debug - JTAG

- La interface JTAG es un estándar abierto, pero la forma de conectar cada celda JTAG internamente es propiedad de cada fabricante.
- Cada fabricante suele proveer de herramientas de debug específicas para sus productos (más allá de implementar interfaces estándar de comunicación con el core de debug en hardware).

The background of the slide is a grayscale, high-magnification photograph of a printed circuit board (PCB). The board is densely packed with various electronic components, including integrated circuits, capacitors, and resistors. A prominent horizontal band of dark blue color is superimposed over the center of the image, serving as a backdrop for the title text.

Interfaces de Debug - Nexus

Interfaces de debug - Nexus

- **Nexus: estándar IEEE-ISTO 5001-2003.**
 - Impulsado por la industria automotriz, pero no limitado a dicha industria.
 - Múltiples niveles (escalabilidad):
 - se provee de funcionalidad básica con pocos pines (para controladores económicos), JTAG solamente para la clase 1.
 - a medida que se aumenta el nivel, se añaden pines adicionales que permiten debugging en tiempo real, etc.
(microcontroladores con mayor poder de procesamiento).
 - Permite definir mensajes privados (sólo para ciertas implementaciones). Son visibles pero no interpretables para terceros (para protocolos propietarios de 3º partes).
 - Es un estándar más completo que BDM y JTAG.

Interfaces de debug - Nexus

- **Nexus:**
Clases de cumplimiento del estándar.


IEEE-ISTO 5001™-1999 Class	Class 1	Class 2	Class 3	Class 4
Trace features	Trace not supported	Adds ownership trace and program trace via Auxiliary ports	Adds data write trace and read/write memory on the fly via Auxiliary ports	Allows tracing to be triggered by a watchpoint via Auxiliary ports
Debug communication method	Half-duplex communication (Limited bandwidth)	Communication may be full duplex using Auxiliary port (higher bandwidth)	Communication may be full duplex using Auxiliary port (higher bandwidth)	Communication may be full duplex using Auxiliary port (higher bandwidth)
Run-time control	Supports run time control features using JTAG interface	Supports run time control features using JTAG interface or Auxiliary port	Supports run time control features using JTAG interface or Auxiliary port	Supports run time control features using JTAG interface or Auxiliary port
Auxiliary Port Implementation	No Auxiliary Port	Allows Port sharing; the Auxiliary port may be shared with slow IO port pins (e.g. static Config pins that are latched at reset time).	Allows Port sharing with high speed I/O ports.	Allows Port sharing with high speed I/O ports.
Data acquisition	Not supported	Not supported	Supports data acquisition	Supports data acquisition
Memory Substitution	Not supported	Not supported	Not supported	Supports memory substitution (fetching or reading data over the IEEE-ISTO 5001™-1999 auxiliary port) Triggering memory substitution on a watchpoint is an optional feature of Class 4 compliance

Interfaces de debug - Nexus

- Se requiere mayor número de pines conforme el CPU emite más información y a tasas más elevadas.
- El pinout está estandarizado según la clase.

Compliance Class and Port Type	Number of Device Data Pins				
	1	2	4	8	16
Class 2 input port	X	X	-	-	-
Class 2 output port	X	X	-	-	-
Class 3 or 4 input port	X	X	X	-	-
Class 3 or 4 output port	-	-	X	X	X

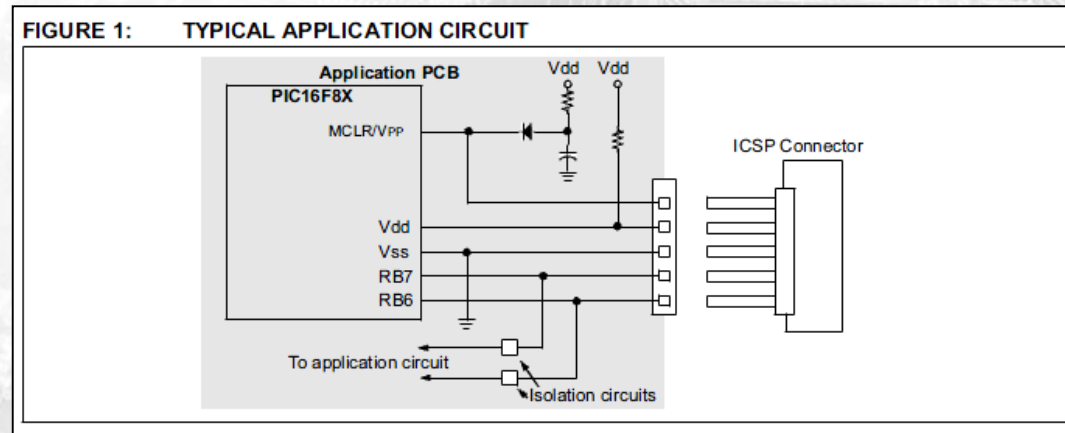
Figure 7.11: I/O pins.

The background of the slide is a grayscale, high-magnification photograph of a printed circuit board (PCB). It shows a dense array of electronic components, including integrated circuits, capacitors, and various connectors. The board's surface is covered with intricate patterns of copper traces and solder points. A dark blue horizontal band is superimposed over the middle of the image, containing the title text in white.

Otras interfaces propietarias de programación y debug

Otras interfaces propietarias

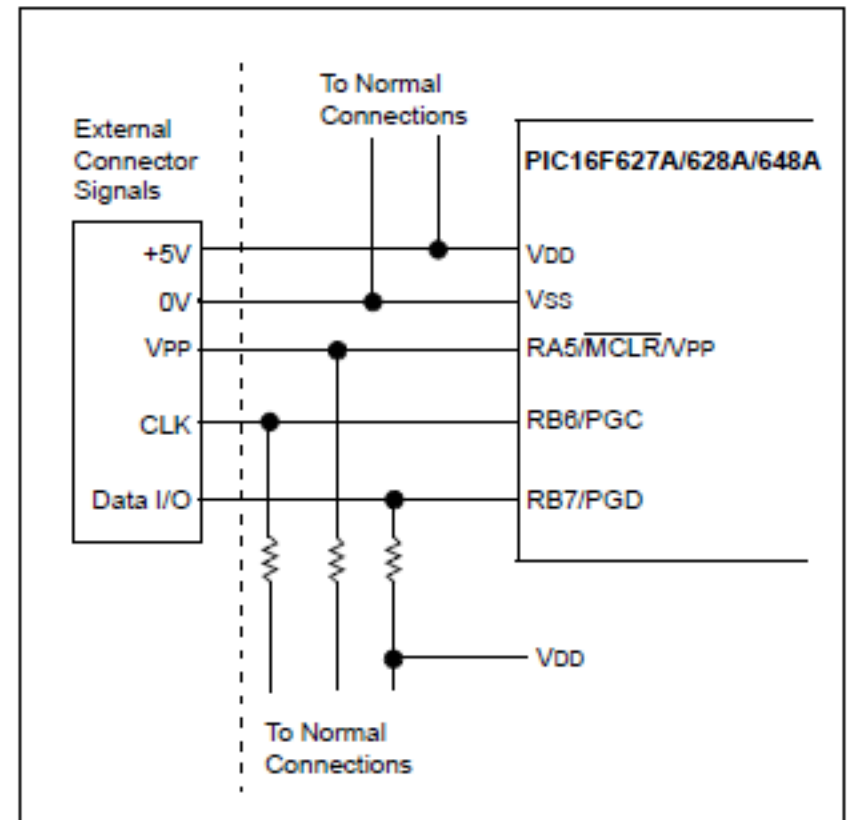
- Las posibilidades no se agotan en las interfaces estándar presentadas.
- Existen interfaces de programación/debug propietarias de ciertos fabricantes:
 - In Circuit Serial Programming (ICSP) para PICs de Microchip
 - In System Programming (ISP) para AVR de Atmel
 - Etc.



Otras interfaces propietarias

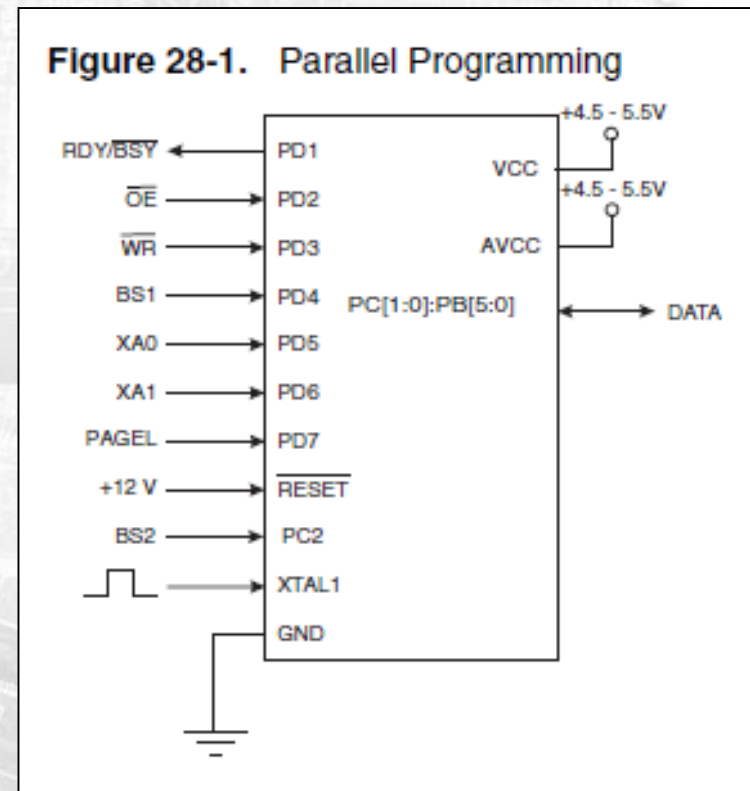
- Ej: In Circuit Serial Programming (ICSP) para PICs de Microchip
 - Para programar
 - Para debuggear
 - Presente en toda la línea de PICs (desde 8 a 32 bits).

FIGURE 14-18: TYPICAL IN-CIRCUIT SERIAL PROGRAMMING CONNECTION



Otras interfaces propietarias

- Ej: In System Programming (ISP) para Atmel AVR's
 - Alternativa al uso de bootloaders (para quemar bootloaders por 1ª vez)
- Programación paralela
 - no tan usado hoy en día.
 - programación más veloz.
 - más número de líneas.



Otras interfaces propietarias

- Ej: In System Programming (ISP) para Atmel AVR^s
 - Alternativa al uso de bootloaders (para quemar bootloaders por 1ª vez)
- Programación en serie
 - usa la unidad SPI del uC.

Figure 28-9. Serial Programming Waveforms

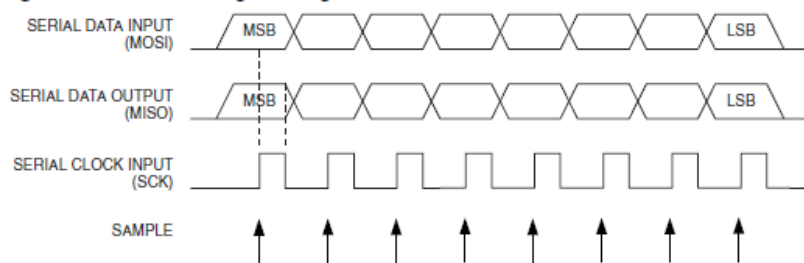
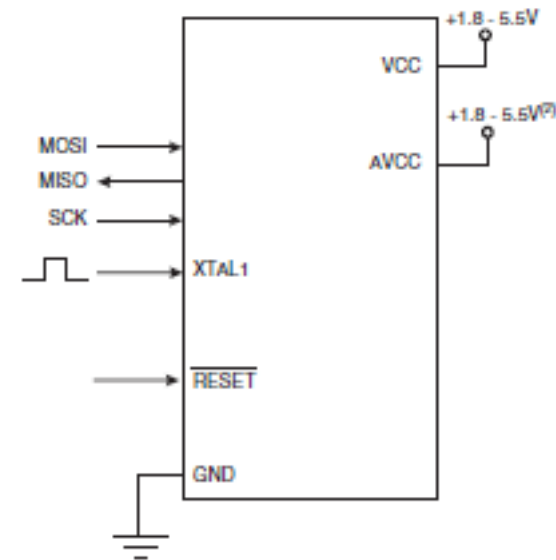
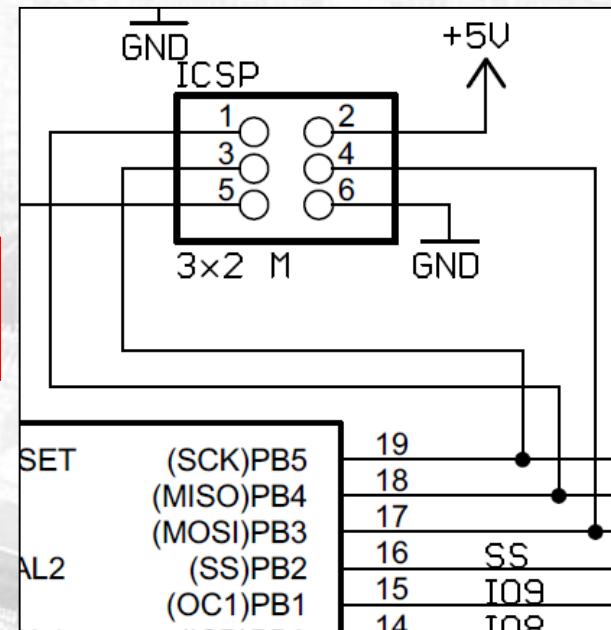
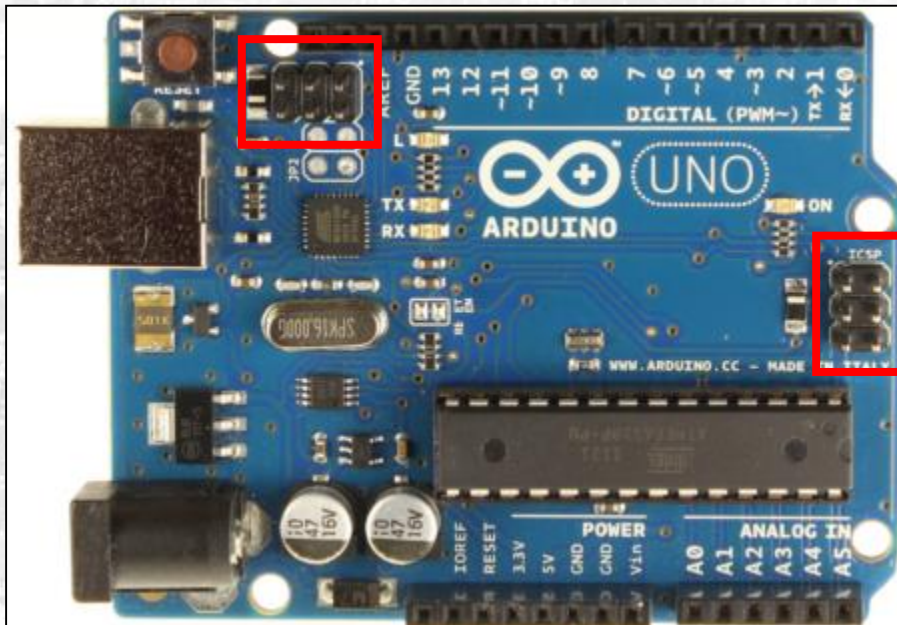


Figure 28-7. Serial Programming and Verify⁽¹⁾



Otras interfaces propietarias

- Ej: In Circuit Serial Programming (ICSP) en Arduino (i.e ISP Serial sobre los ATMegas)



Referencias

- **Atmel AVR ATmega328P Datasheet.**
- **Berger, A. Embedded Systems Design: An Introduction to Processes, Tools & Techniques. CMP Books. 2001. ISBN: 978-1578200733. Capítulo 7.**
- **Wilmshurst, T. Designing Embedded Systems with PIC Microcontrollers: Principles and Applications. Newnes. 2006. ISBN: 978-0750667555. Capítulo 20.**