Embedded Systems

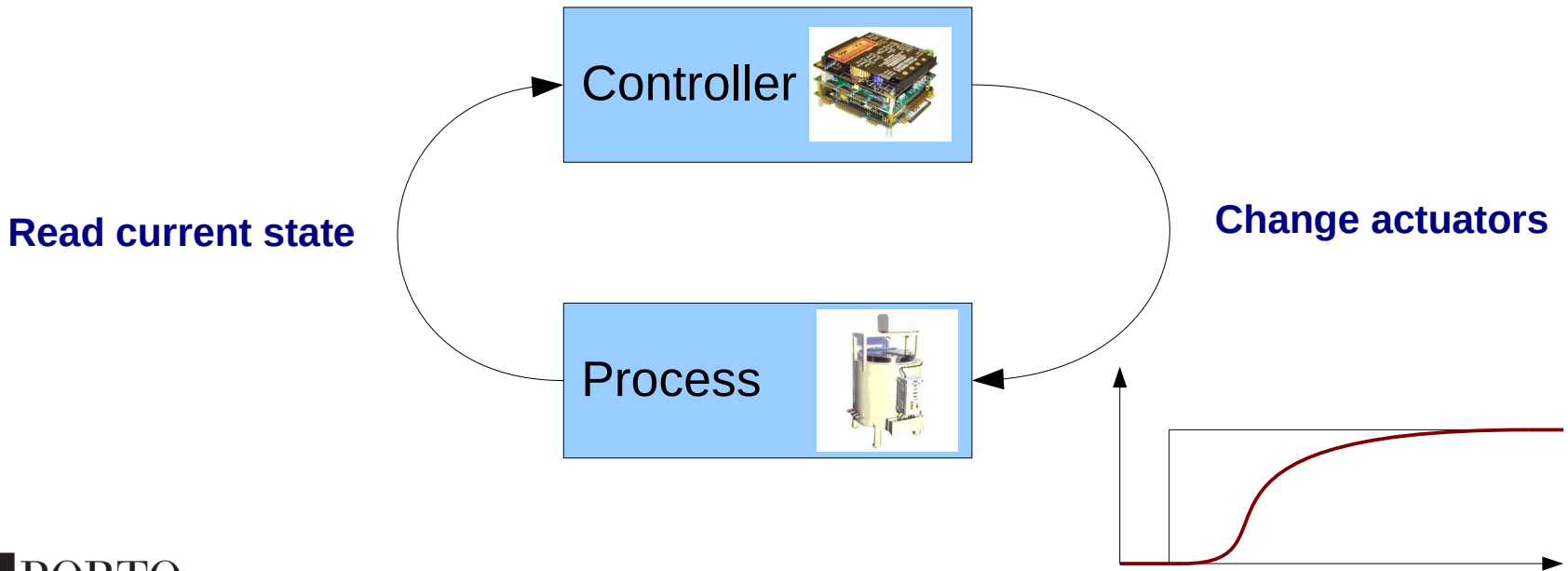
U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Hardware Architectures

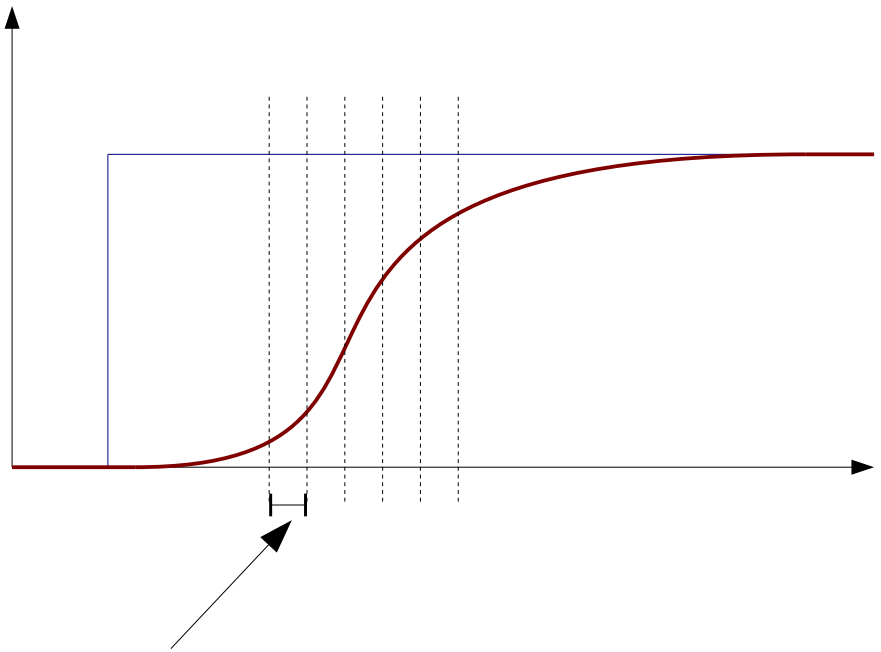
Mário de Sousa
msousa@fe.up.pt

Requirements Analysis

- Functional requirements
  - What actions should the program take?
- Timing Requirements
  - When (or latest time) should each action be taken?



Timing Requirements



Control algorithms may be configured to compensate Delay, but Jitter is difficult to tolerate.

Sampling period rule of thumb: < 10% of rise time
Jitter: changes in sampling period (should be much smaller then sampling period!)
Delay: time between reading of inputs, and actuating

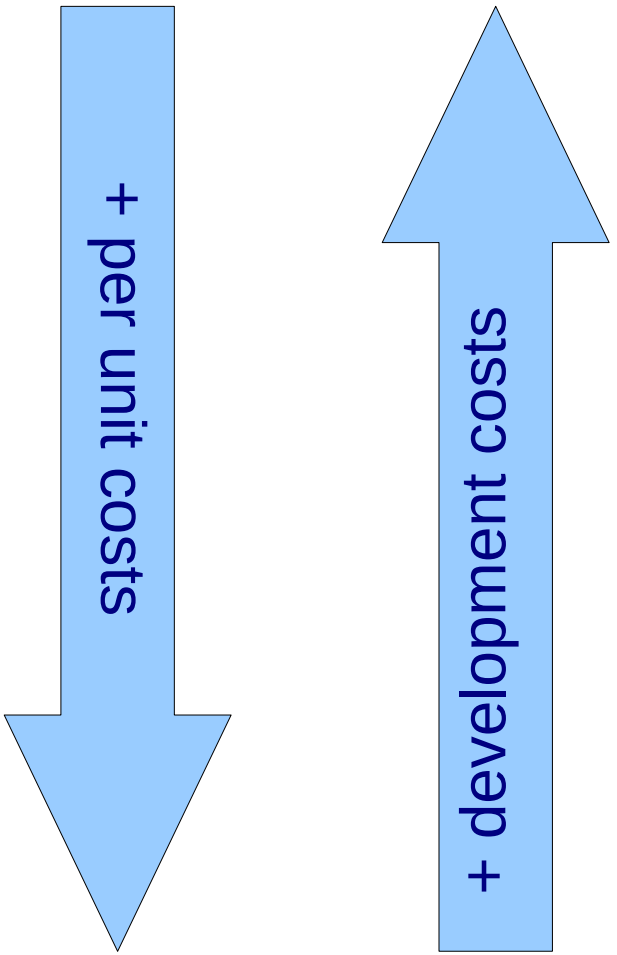
Actual values will depend on how fast the controlled process changes!

Hardware Architectures for Embedded Systems

- Processors / Execution of Control Logic
- Boards / Form Factors
- Memory Technologies
- Interface / Communication Buses

Hardware / Processor Spectrum

- FPGA / ASIC
- Micro Controllers
- Micro Processors
- System on a Chip - SOC
- Single Board Computer - SBC

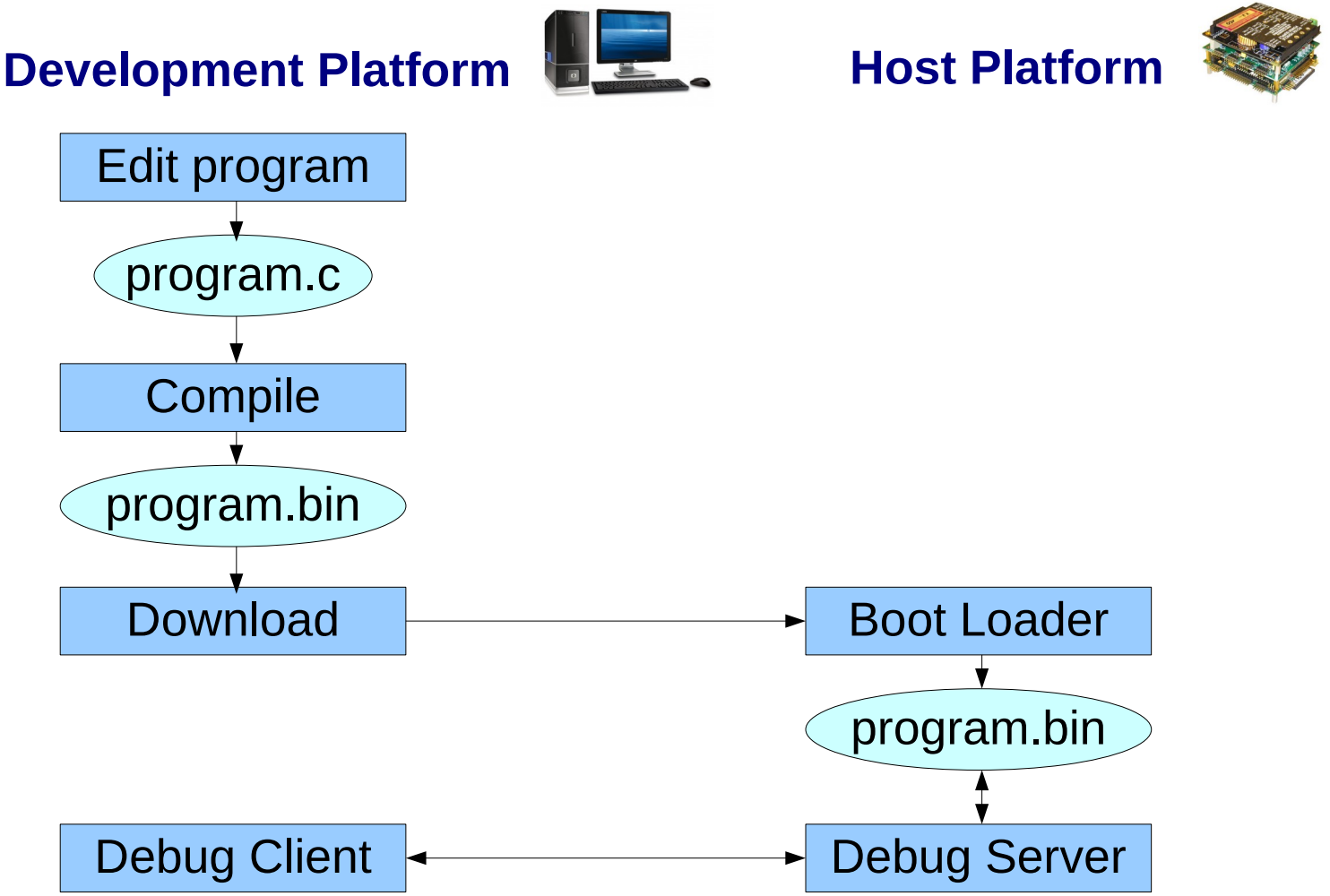


ASIC: Application Specific Integrated Circuit
FPGA: Field Programmable Gate Array

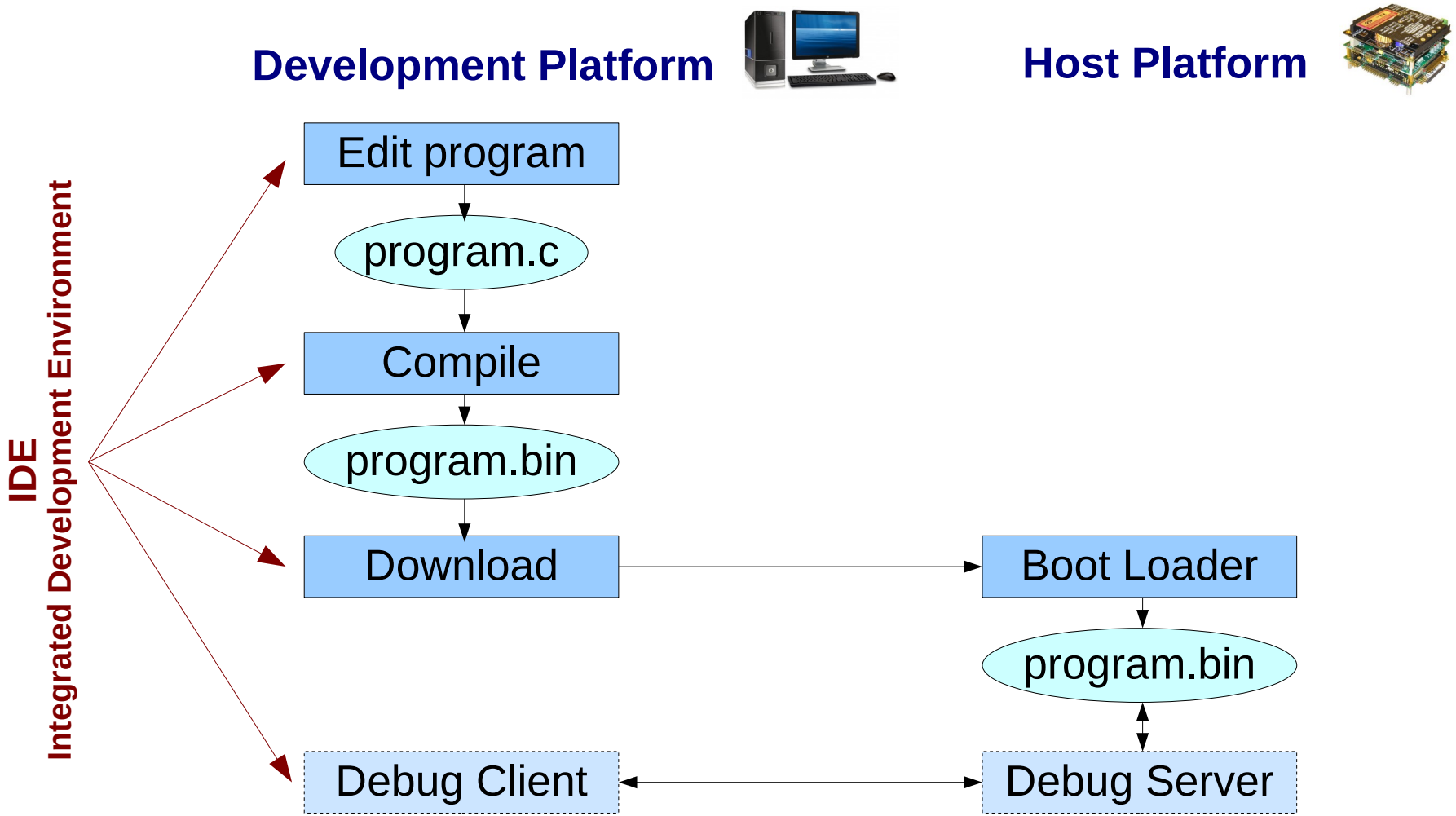
Processors/Execution of Control Logic

- Criteria for choice:
  - Unit Costs
  - Development Costs
  - Functional Requirements:
    - Number and type of I/Os (boolean, network/data bus interfaces, ...)
    - Processing capacity
    - Data/Program storage capacity (memory)
    - Power consumption (energy storage => run time)
    - Size / Weight
    - Physical robustness (vibration, temperature, ...)
  - ...

The development process...



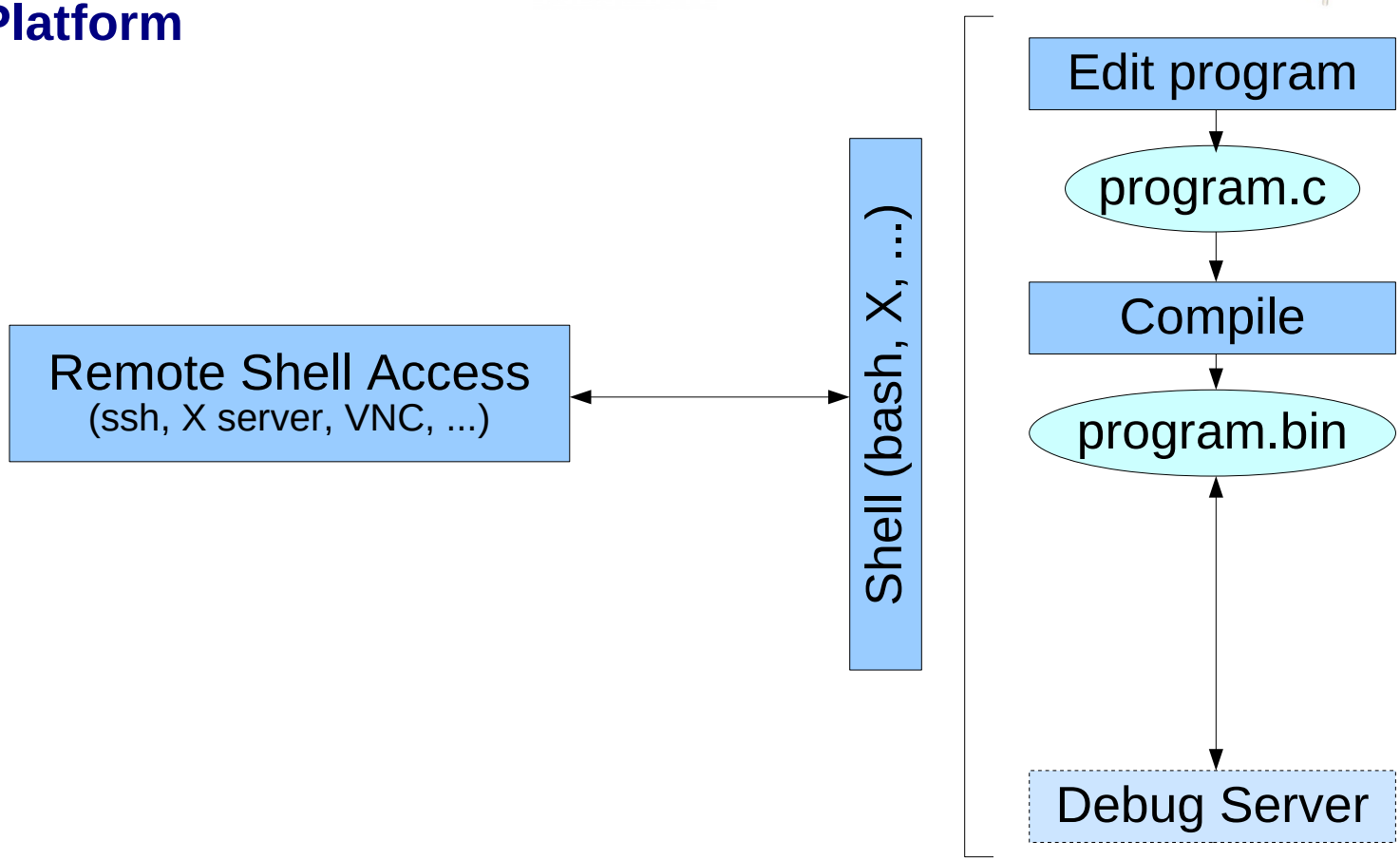
The development process...



The development process...

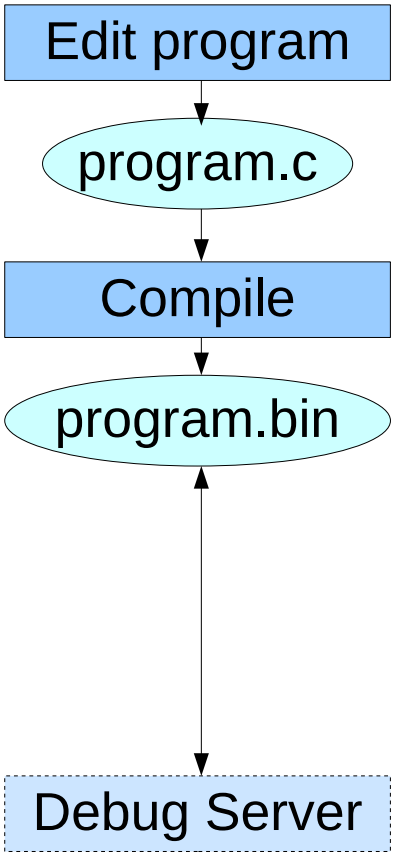
Development Platform

Host Platform



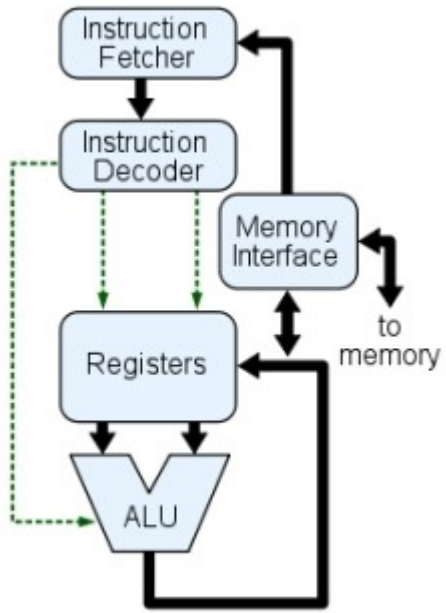
The development process...

Development + Host Platform



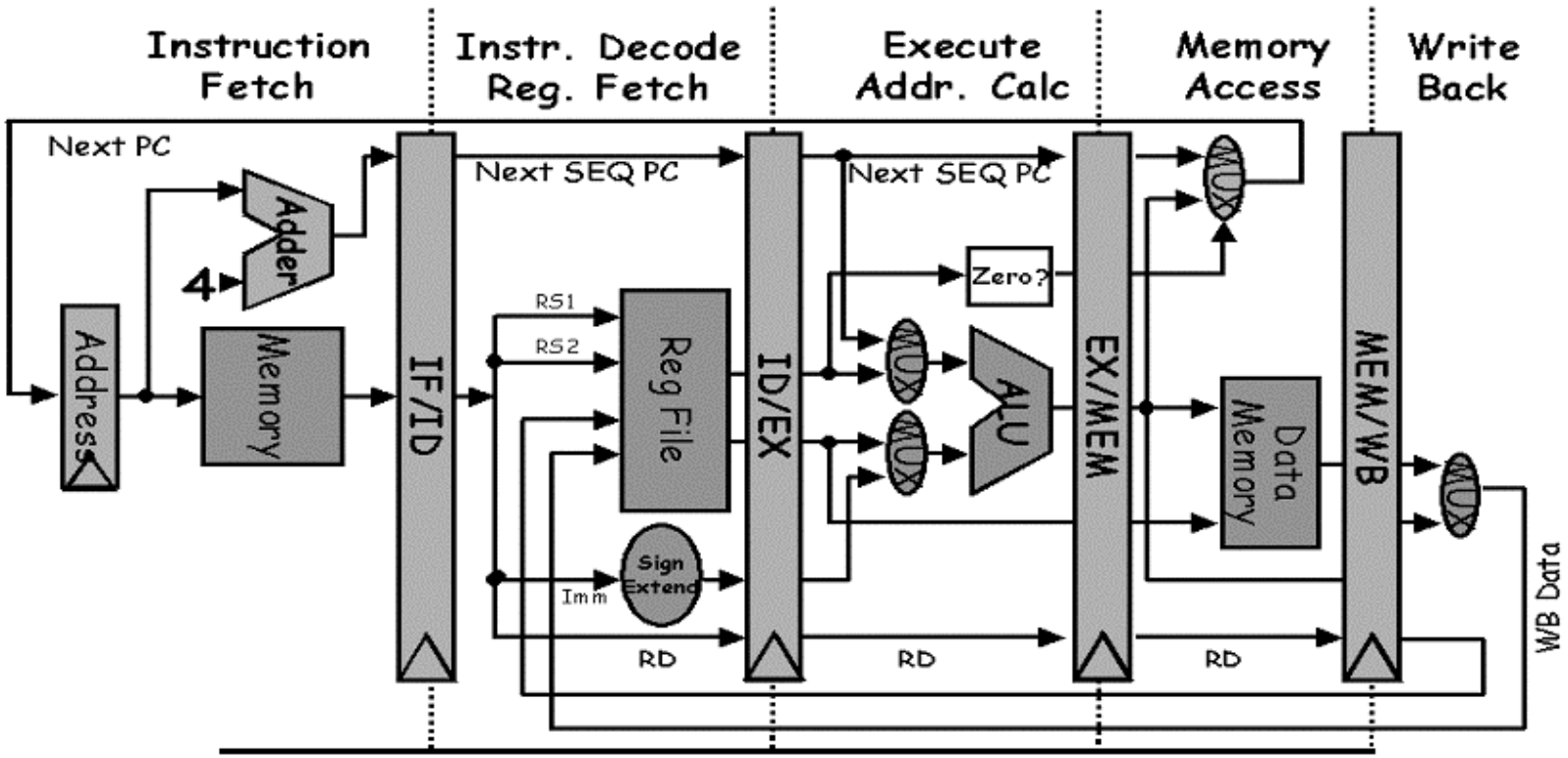
NOTE: More details regarding development process to come in later lectures...

CPU architecture



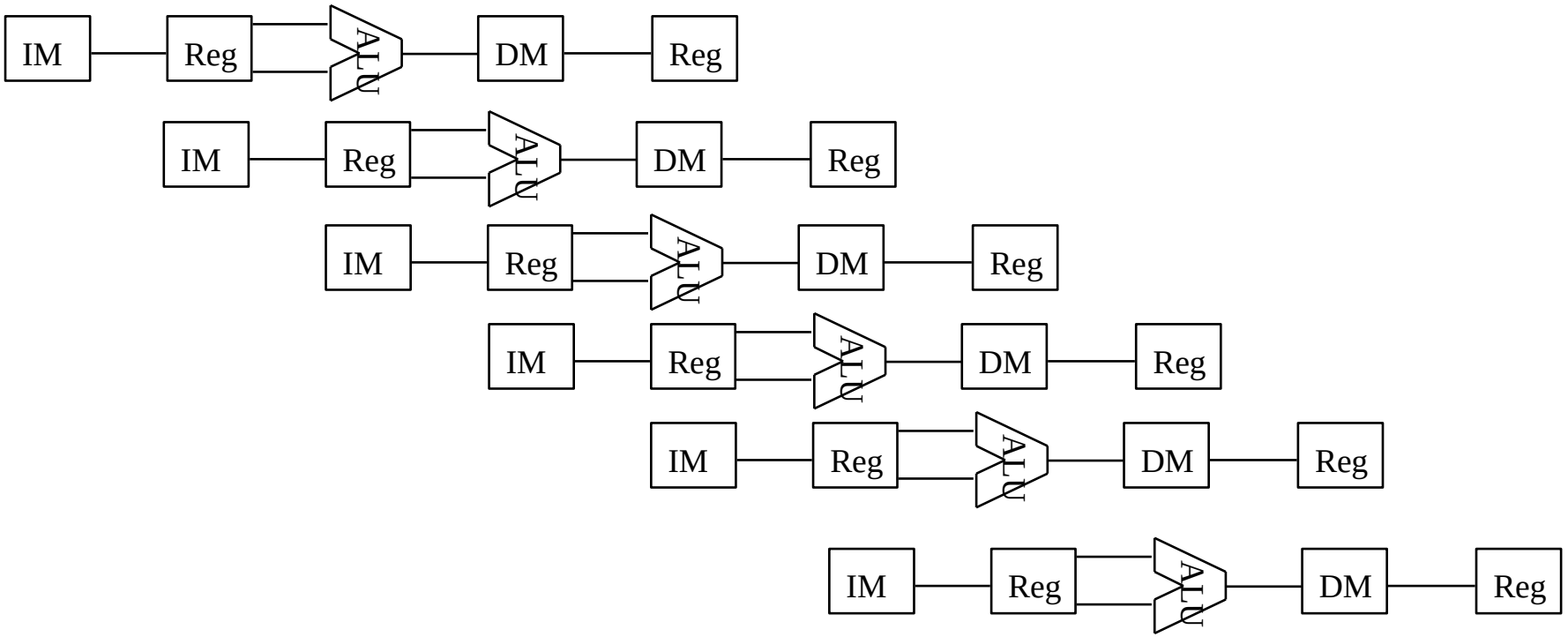
Pipeline Basics

typical 5 stage pipeline



Pipeline Basics

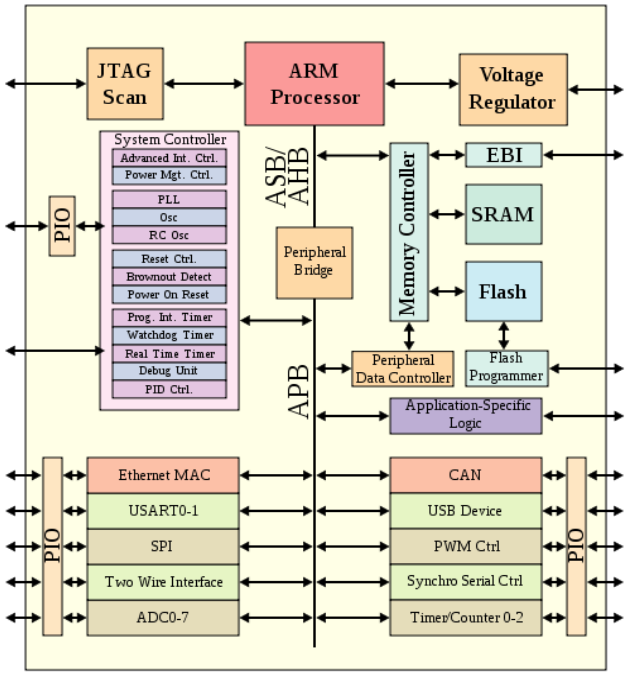
Instruction overlap

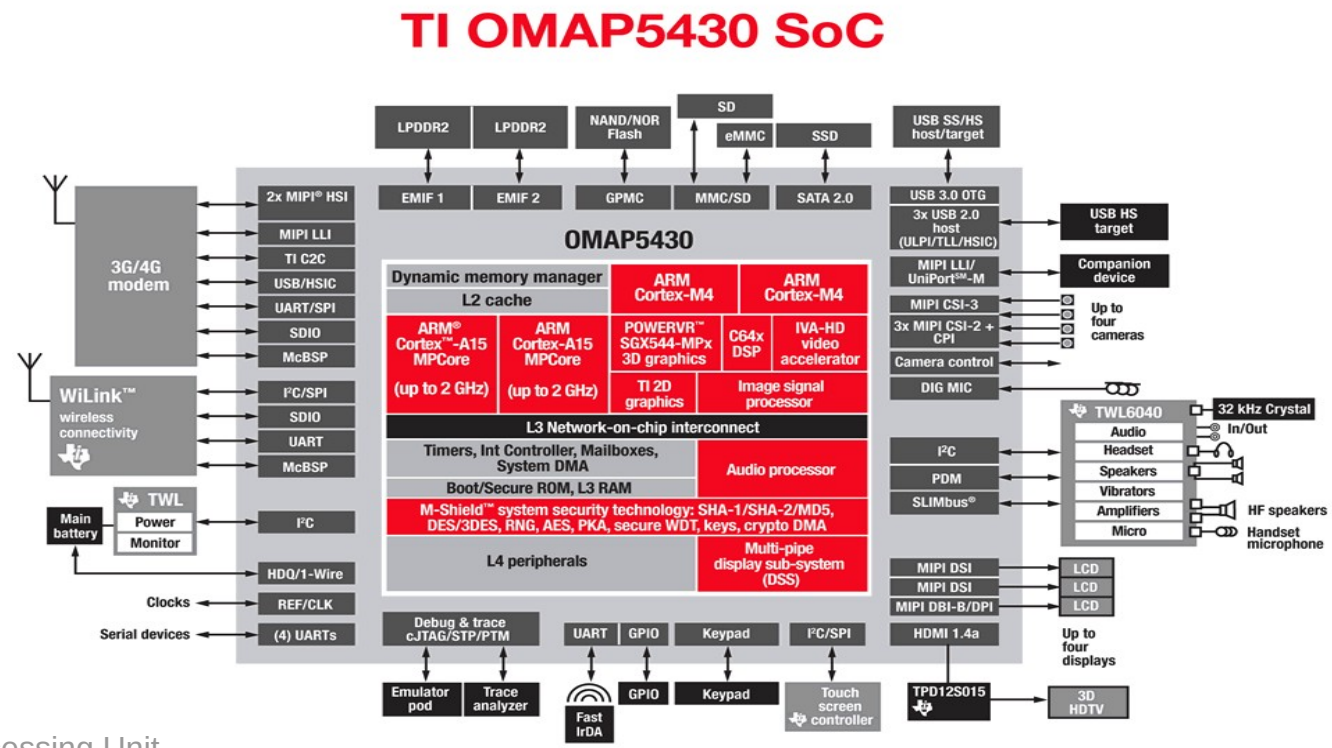


Limitations	- non uniform delay - branching (speculative execution, branch prediction)
Enhancements	- out of order execution

ARM CPUs

- ARM Holdings
  - sells IP core designs, which licensees use to create micro-controllers (MCUs), CPUs, and systems-on-chips based on those cores.
  - Licensees include Texas Instruments, Apple, Atmel, Broadcom, ...

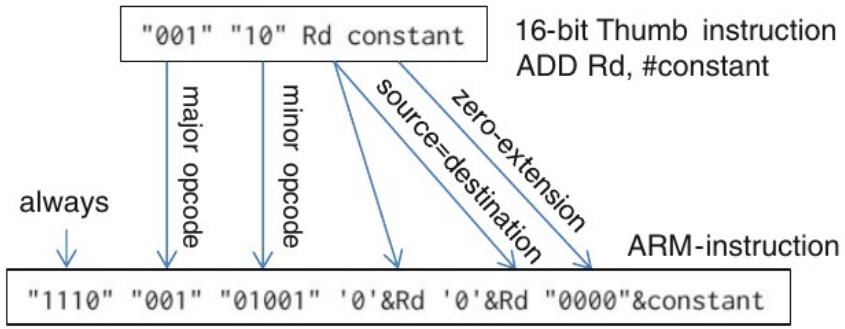




CPU: Central Processing Unit
MCU: MicroController Unit

ARM CPUs – Instruction Sets

- **RISC**
  - Reduced Instruction Set Computer
    - Smaller number of simpler instructions
    - Reduces complexity of CPU and transistor count
    - Requires more work out of compiler



  - **A32, A64**
    - Instructions for 32 and 64 bit wide registers / memory addressing
    - Instructions are 32 bits wide
  - **T32** **(T - thumb)**
    - Reduced number of instructions (some are 16 and others 32 bit width)
    - Allows generation of smaller code (higher code density)
    - provides compilers the opportunity to balance performance and code size trade-offs in a single instruction set.

ARM CPUs – Architectures

- **A Profile (Application)**
  - Armv8-A (64 bits)
    - Supports A64, A32 and T32 instruction sets
    - Ability to use 64-bit (AArch64) and 32-bit (AArch32) Execution states (for backward compatibility with Armv7-A)
  - Armv7-A (32 bits)
    - Supports A32 and T32 instruction sets
- **R Profile (Real-Time)**
    - Supports a Protected Memory System Architecture (PMSA) based on a Memory Protection Unit (MPU, simpler than MMU)
  - Armv8-R (32 bits)
    - Supports A32 and T32 instruction sets
    - Virtualization support for guest operating systems, no overlapping memory regions, ...
  - Armv7-R (32 bits)
    - Supports A32 and T32 instruction sets

ARM CPUs – Architectures

- **M Profile (Microcontroller)**
  - **low-latency, highly deterministic operation for deeply embedded systems.**

  - Armv8-M
    - optionally implements a Memory Protection Unit (MPU)
      based on Protected Memory System Architecture (PMSA).
    - Supports variation of T32 instruction set
    - Backward compatible with Armv6-M
  - Armv7-M
    - low cycle count execution, minimal interrupt latency and cache-less operation,
    - designed for implementations where overall size and deterministic operation are more important than absolute performance.
    - Supports variation of T32 instruction set
  - Armv6-M
    - Support for the T32 instruction set.
    - Upward compatibility with Armv7-M

Armv7A – Processor Modes

Processor mode		Mode number	Description
User	usr	0b10000	Normal program execution mode
FIQ	fiq	0b10001	Supports a high-speed data transfer or channel process
IRQ	irq	0b10010	Used for general-purpose interrupt handling
Supervisor	svc	0b10011	A protected mode for the operating system
Abort	abt	0b10111	Implements virtual memory and/or memory protection
Undefined	und	0b11011	Supports software emulation of hardware coprocessors
System	sys	0b11111	Runs privileged operating system tasks (ARMv4 and above)

Previliged Modes

Exception Modes: entered when specific exceptions occur

Image taken from:
ARM Architecture Reference Manual (pg A2-3)

Armv7A – Registers in AArch32 state

15 General Purpose Registers

SP: Stack Pointer
LR: Link Register
PC: Program Counter
APSR: Application Program Status Register
CPSR: like APSR with additional status bits

Processor Mode

Application level view | System level view

	User	System	Hyp †	Supervisor	Abort	Undefined	Monitor ‡	IRQ	FIQ
R0	R0_usr								
R1	R1_usr								
R2	R2_usr								
R3	R3_usr								
R4	R4_usr								
R5	R5_usr								
R6	R6_usr								
R7	R7_usr								
R8	R8_usr								R8_fiq
R9	R9_usr								R9_fiq
R10	R10_usr								R10_fiq
R11	R11_usr								R11_fiq
R12	R12_usr								R12_fiq
SP	SP_usr		SP_hyp	SP_svc	SP_abt	SP_und	SP_mon	SP_irq	SP_fiq
LR	LR_usr			LR_svc	LR_abt	LR_und	LR_mon	LR_irq	LR_fiq
PC	PC								
APSR	CPSR								
			SPSR_hyp	SPSR_svc	SPSR_abt	SPSR_und	SPSR_mon	SPSR_irq	SPSR_fiq
			ELR_hyp						

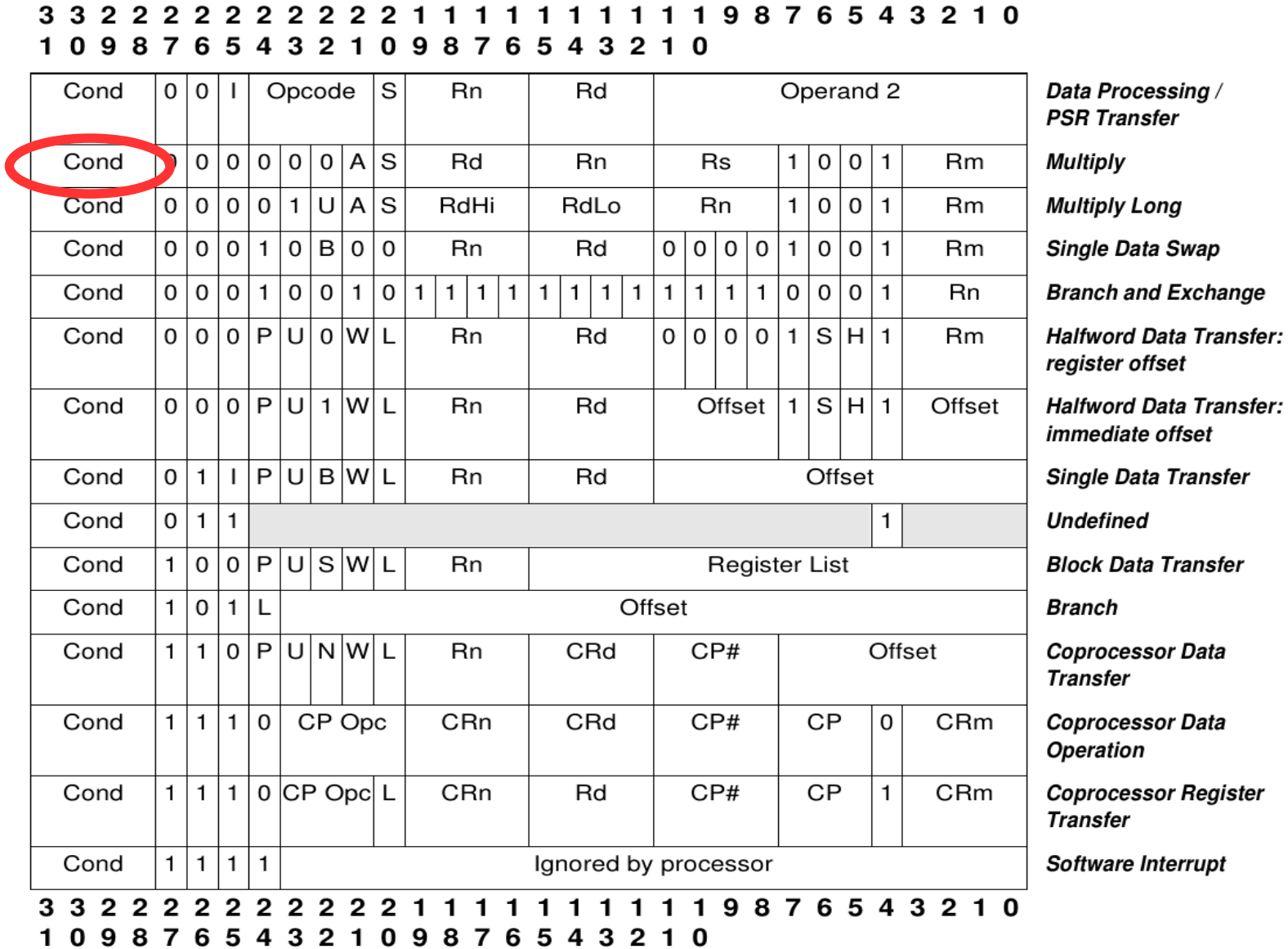
‡ Exists only in Secure state.
† Exists only in Non-secure state.
Cells with no entry indicate that the User mode register is used.

Image taken from:
ARM Instruction Set Reference Guide (pg 58)

A32 Instruction Set

All instructions have conditional execution

Conditional execution depends on status of APSR Register



31 30 29 28	27 26	25	24 23 22 21	20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0	
Cond	0 0	I	Opcode	S	Rn	Rd	Operand 2			*Data Processing / PSR Transfer*
Cond	0 0	0	0 0 0 A	S	Rd	Rn	Rs	1 0 0 1	Rm	*Multiply*
Cond	0 0	0	0 1 U A	S	RdHi	RdLo	Rn	1 0 0 1	Rm	*Multiply Long*
Cond	0 0	0	1 0 B 0	0	Rn	Rd	0 0 0 0	1 0 0 1	Rm	*Single Data Swap*
Cond	0 0	0	1 0 0 1	0	1 1 1 1	1 1 1 1	1 1 1 1	0 0 0 1	Rn	*Branch and Exchange*
Cond	0 0	0	P U 0 W	L	Rn	Rd	0 0 0 0	1 S H 1	Rm	*Halfword Data Transfer: register offset*
Cond	0 0	0	P U 1 W	L	Rn	Rd	Offset	1 S H 1	Offset	*Halfword Data Transfer: immediate offset*
Cond	0 1	I	P U B W	L	Rn	Rd	Offset			*Single Data Transfer*
Cond	0 1	1						1		*Undefined*
Cond	1 0	0	P U S W	L	Rn	Register List				*Block Data Transfer*
Cond	1 0	1	L	Offset						*Branch*
Cond	1 1	0	P U N W	L	Rn	CRd	CP#	Offset		*Coprocessor Data Transfer*
Cond	1 1	1	0	CP Opc	CRn	CRd	CP#	CP	0 CRm	*Coprocessor Data Operation*
Cond	1 1	1	0	CP Opc	L	CRn	Rd	CP#	CP	1 CRm	*Coprocessor Register Transfer*
Cond	1 1	1	1	Ignored by processor						*Software Interrupt*

Image taken from:
ARM7TDMI-S Data Sheet (pg 4-2)

A32 Instruction Set

Conditional execution depends on status of APSR Register

Format of DATA PROCESSING instructions

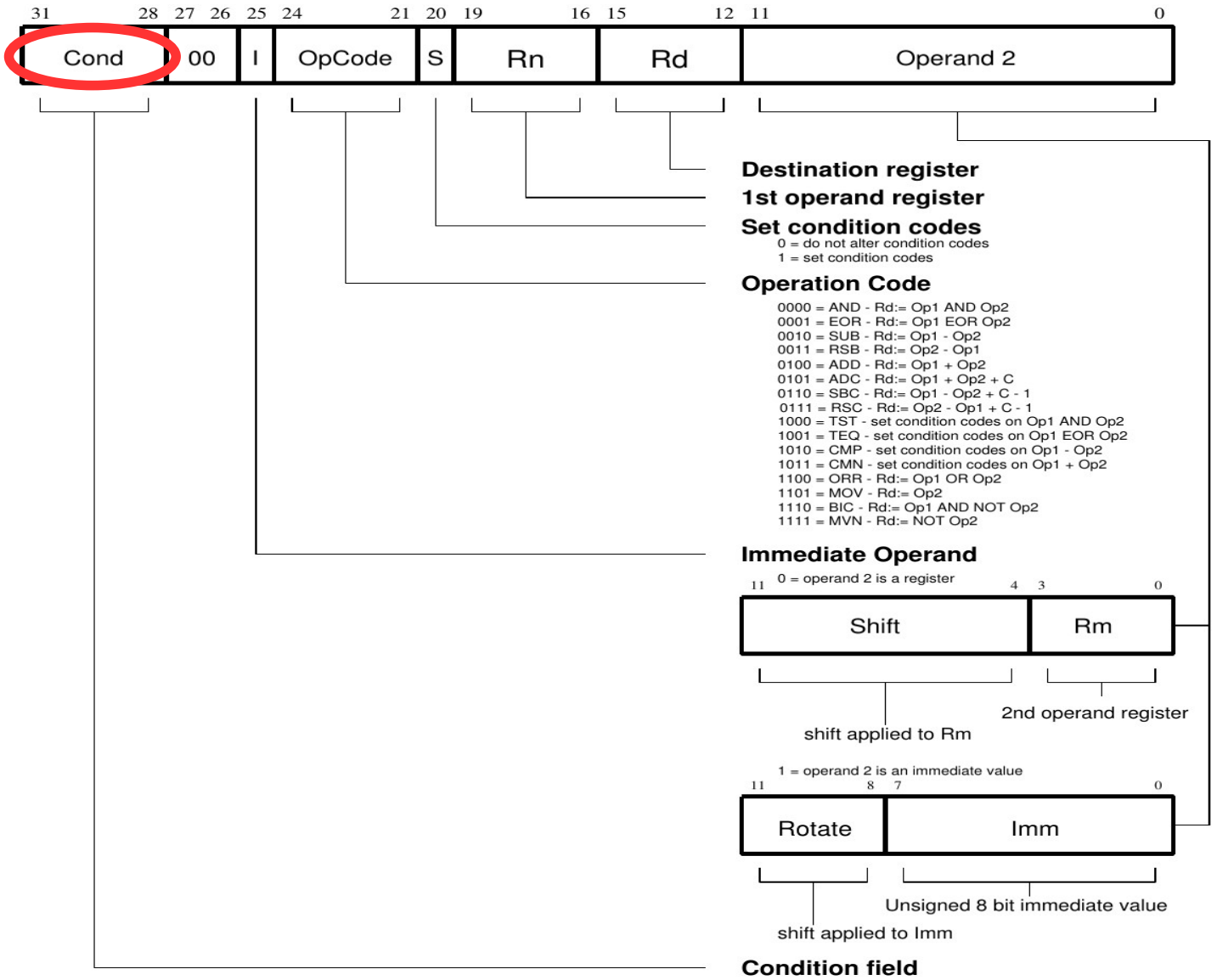
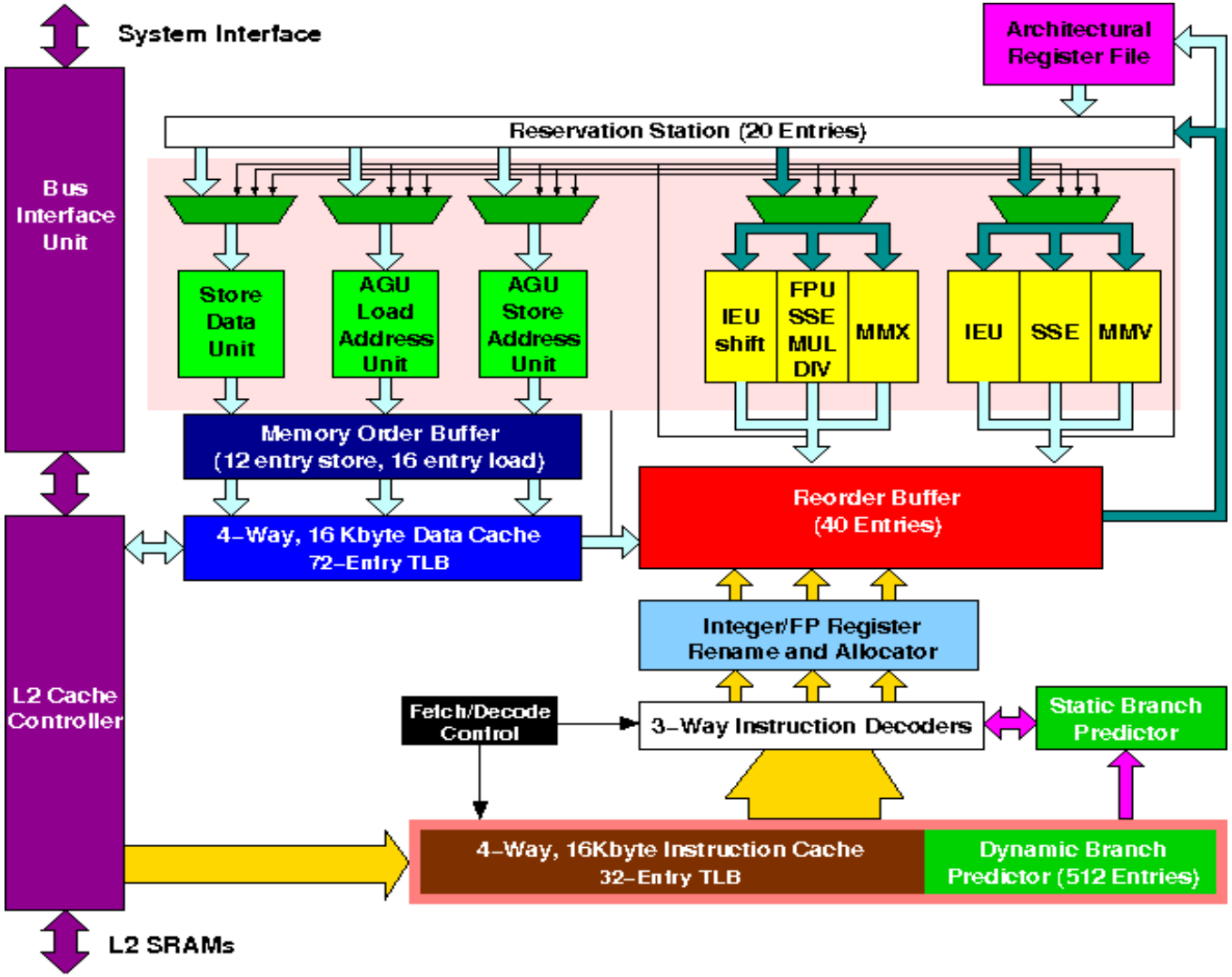
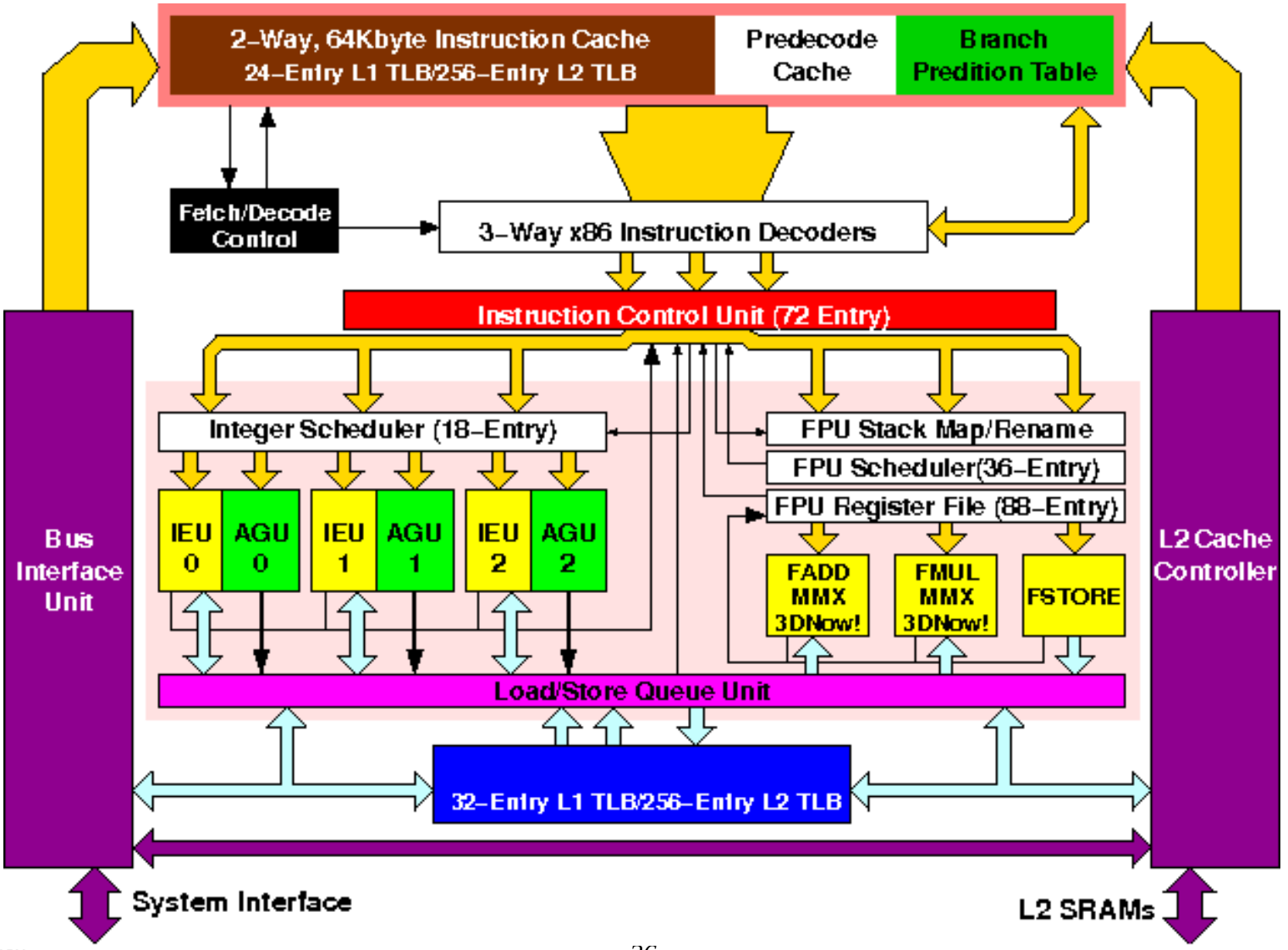


Image taken from:
ARM7TDMI-S Data Sheet (pg 4-2)

Intel P6 Core



AMD Athlon Processor Core



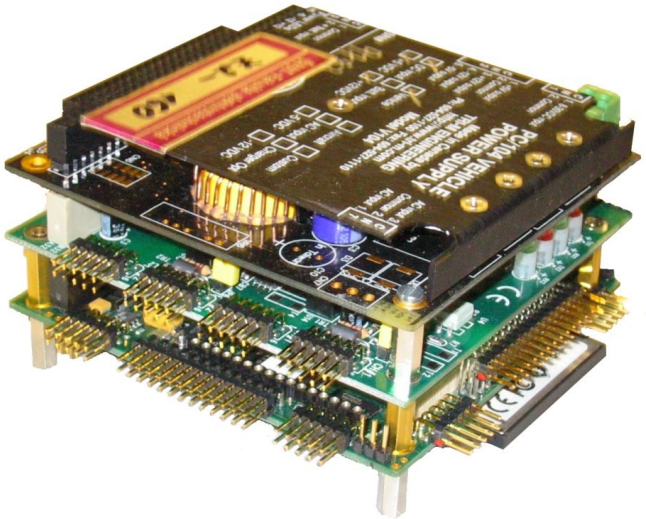
Hardware Architectures for Embedded Systems

- Processors / Execution of Control Logic
- Boards / Form Factors
- Memory Technologies
- Interface / Communication Buses

Single-Board Computer

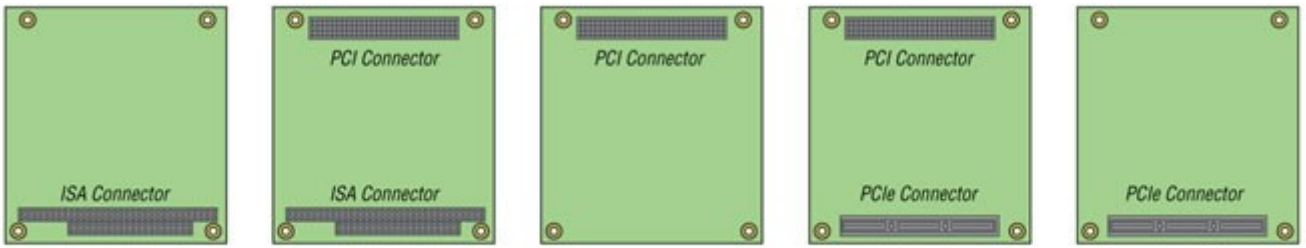
- Form factors
  - AdvancedTCA (Advanced Telecommunications Computing Architecture)
  - CompactPCI
  - PICMG
  - Mini-ITX, Nano-ITX, Pico-ITX, Mobile-ITX
  - PC/104, EBX, EPIC
  - PXI
  - Qseven
  - VMEbus, VPX, VXI (VME eXtension for Instrumentation)
  - 96Boards (CE, EE, EETV and IE)
  - Etc...

Embeddable PCs



- PC/104 → ISA Bus
  - DIP switches to set the Base Address, IRQ, and DMA channel of peripheral boards
- PC104-Plus → ISA + PCI Buses
  - adds 120-pin connector for the PCI bus, on the opposite side of the board from PC/104
    - boards have DIP switch to set PCI slot number 0-3, lowest closest to CPU
- PCI/104 → PCI Bus
- PCI/104-Express → PCI + PCIe Bus
- PCIe/104 → PCIe Bus

Feature	Type 1	Type 2
USB 2.0	2	2
SMB	1	1
PCIe x1	4	4
PCIe x4		2
PCIE x16	1	
USB 3.0		2
SATA		2
LPC		1
RTC Battery		1



PC/104 PC/104-*Plus* PCI-104 PCI/104-Express PCIe/104

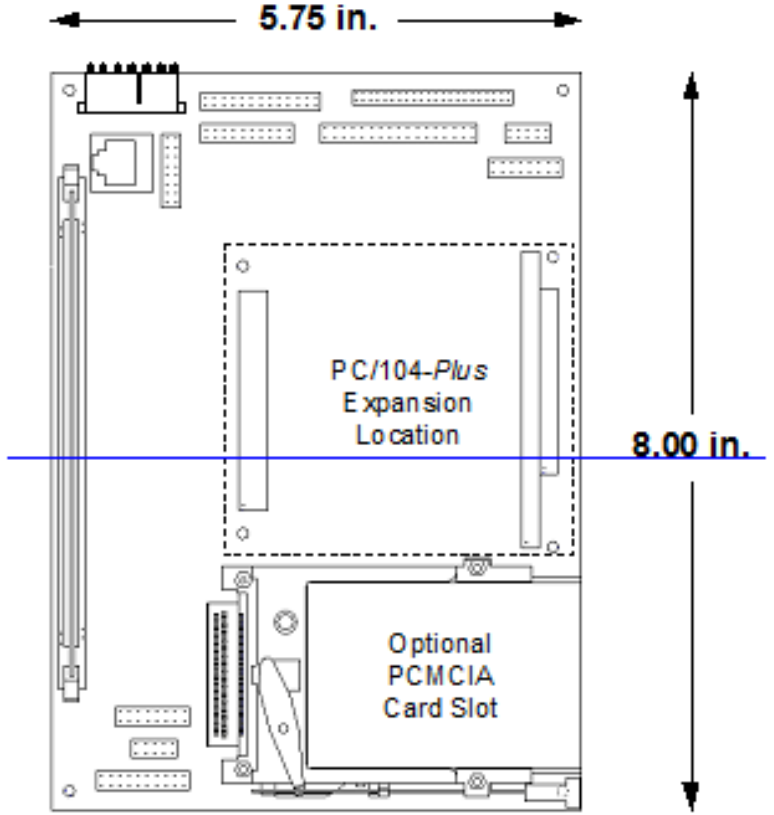
www.pc104.org

Embeddable PCs

- PC/104, PC104-Plus, PCI/104, PCI/104-Express
  - Max number of boards:
    - ISA: limited by electrical signal quality
    - PCI: 4 (PCI bus limit)
  - PCI Boards must be mounted next to the CPU
  - ISA Boards may be mounted at top of stack.
  - Dimensions: 3.550” x 3.775” (90.17 x 95.89 mm)

Embeddable PCs

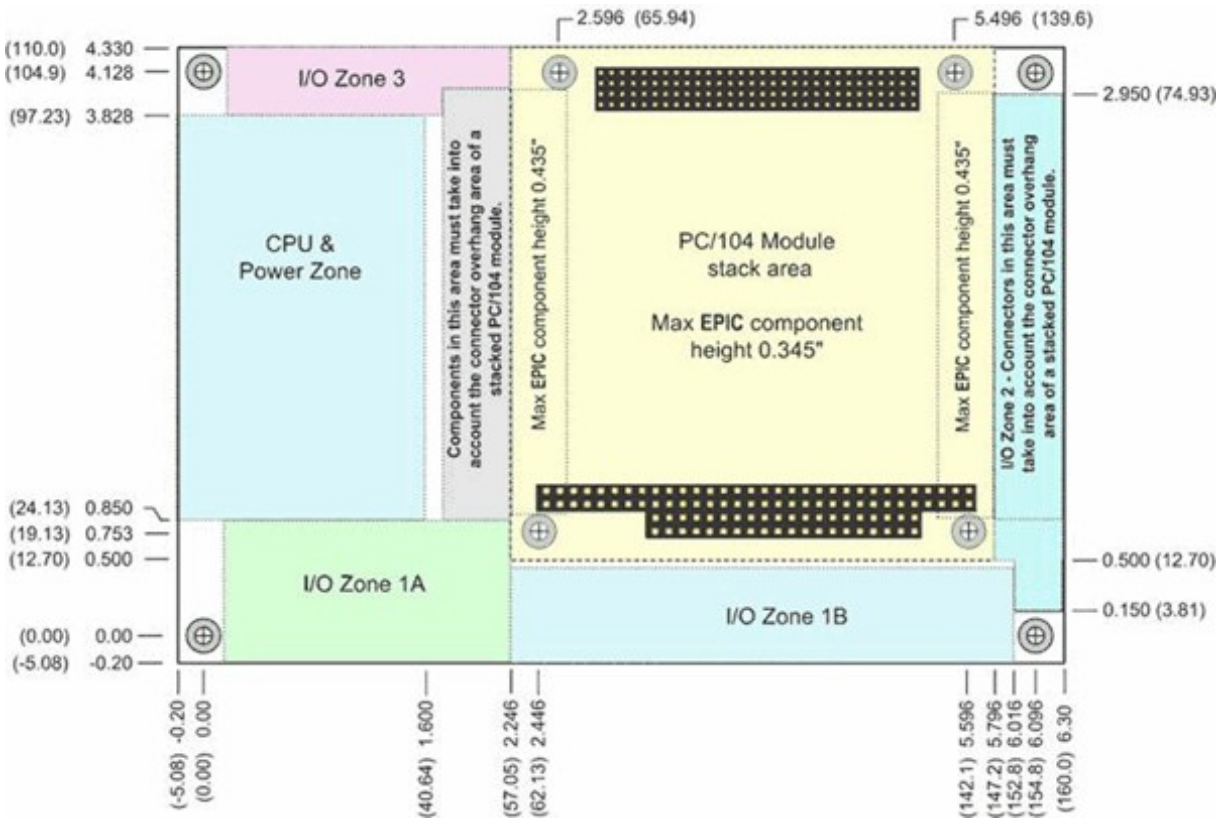
- EBX (Embedded Board, eXpandable)
  - Large enough to include:
    - CPU with heatsink
    - memory
    - mass storage interfaces
    - graphical controller
    - serial/parallel ports
    - ...
  - Dimensions:
    - 5.750 x 8.000 inches (146.05 x 203.20 mm)



EBX Form-factor

Embeddable PCs

- EPIC / EPIC-Express
  - Large enough to include:
    - CPU with heatsink
    - I/O zones
    - ...
  - Dimensions:
    - 115.00 x 165.00 mm (4.528 x 6.496 inches)
  - Express supports PCIe



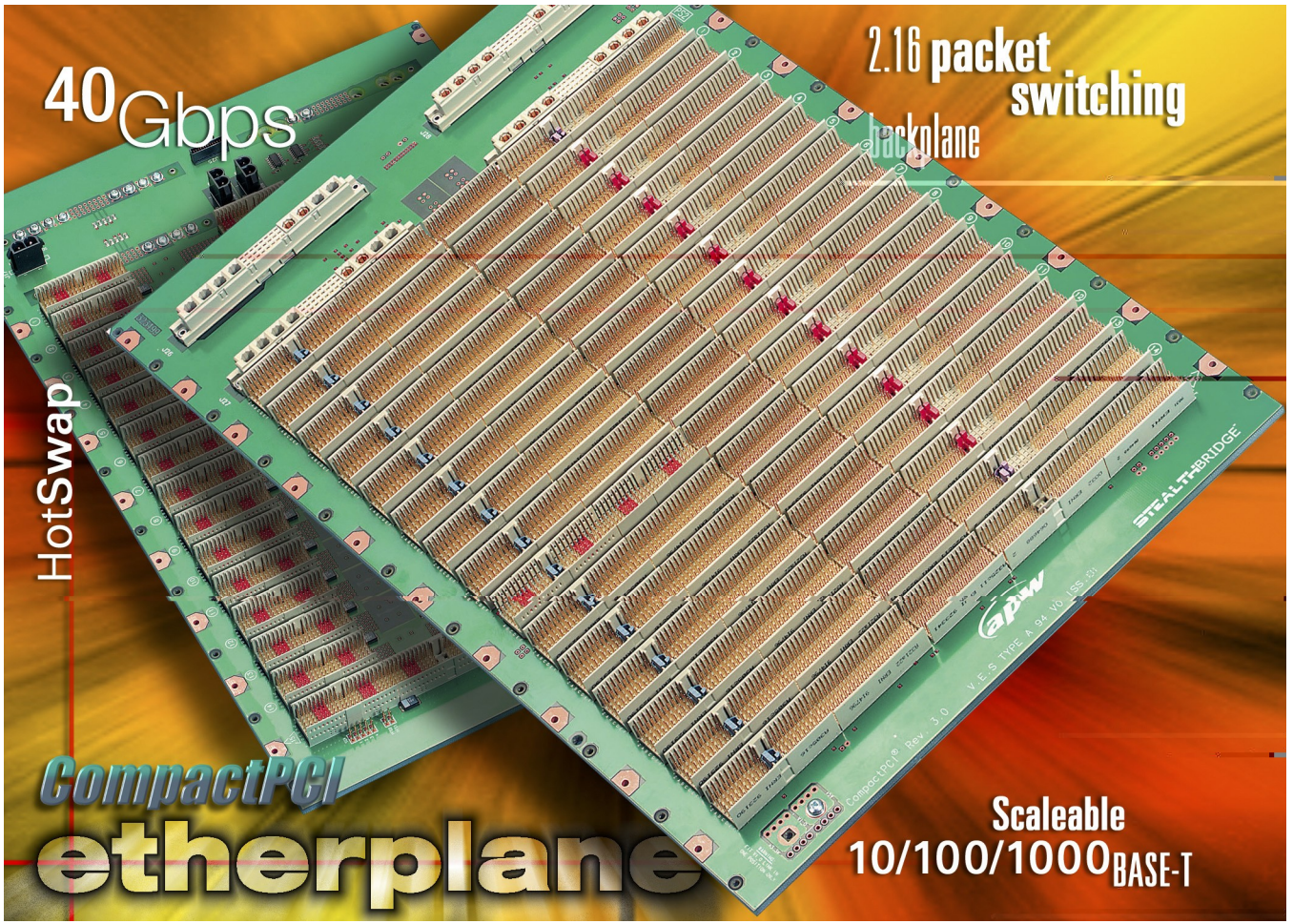
Embeddable PCs

- CompactPCI
  - Uses passive backplane
    - PCI 32 and 64 bits
  - Boards slot into backplane
    - CPU, network, graphical controller, ...
    - Simultaneous use of 32 and 64 bits boards
  - Dimensions:
    - Eurocard Format
    - 3U (100mm by 160 mm)
    - 6U (160mm by 233 mm)
  - 3U Boards
    - 220 pin connector: PCI 64 bit bus
    - 110 pin connector: PCI 32 bit bus
  - 6U Boards
    - 110 / 220 pin connector: PCI 32 / 64 bit bus
    - extra 220 pin connector: non standardized (may be used for other buses: ISA, VME, ...)

More details: www.picmg.org

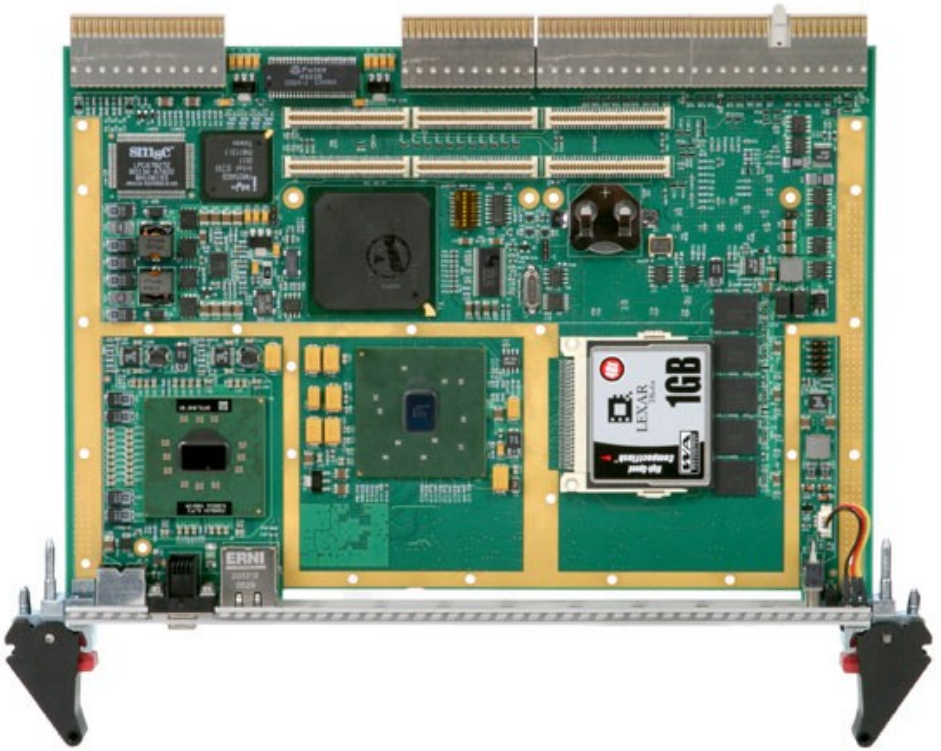
Embeddable PCs

- CompactPCI



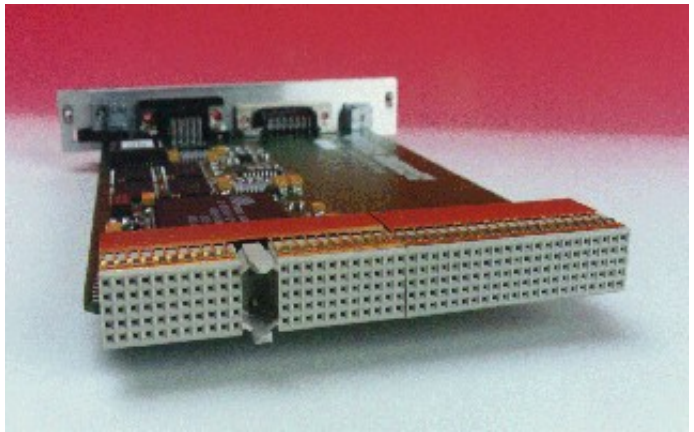
Embeddable PCs

- CompactPCI



Embeddable PCs

- CompactPCI



Embeddable PCs

- CompactPCI
  - Vertical Card Orientation for good cooling
  - Positive Card Retention /
    Excellent Shock and Vibration Characteristics
  - User I/O Connections on Front or Rear of module
  - Uses Standard PCI Silicon Manufactured in Large Volumes
  - Staged Power Pins for Hot Swap Capability
  - Eight Slots in Basic Configuration.
    Easily expanded with Bridge Chips

Embeddable PCs

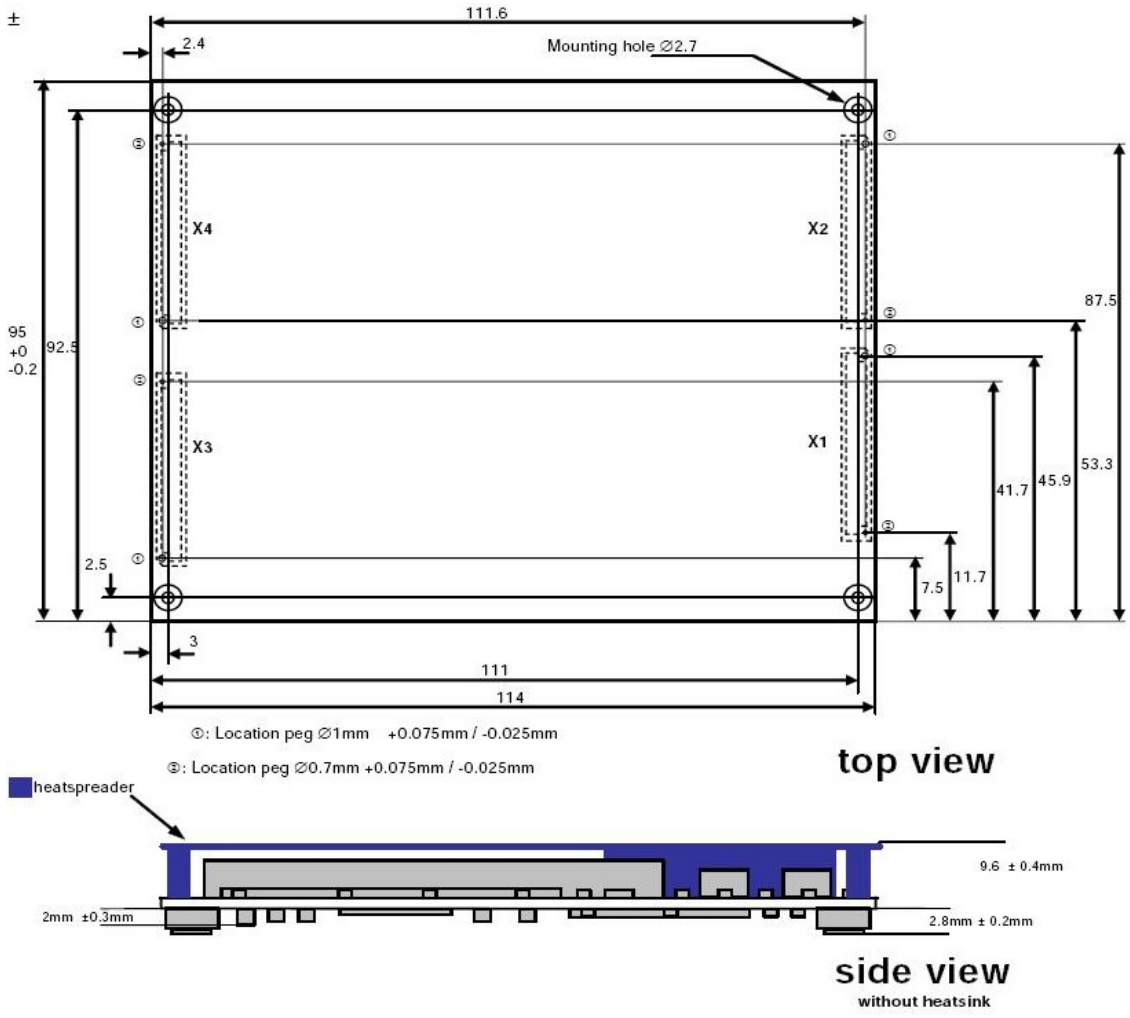
- VMEbus
  - Origin of CompactPCI
  - No longer in wide use

Embeddable PCs

- VMEbus
  - Original VME: IEEE 1014-1987
    - two 3 row P1/P2 connectors,
    - providing 32 bit Data Xfers @ 40MBytes/second.
  - Revision C
    - 64 bit Data Xfers @ 80MBytes/second.
    - the upper 32 data bits being multiplexed onto the address bus.
  - Revision D
    - 64 bit Data Bus, Xfers @ 80MBytes/second.
    - ANSI/VITA 1-1994 termed VME64,
  - VME320
    - increased the bus speed to 320 Mbytes/sec on the back-plane.

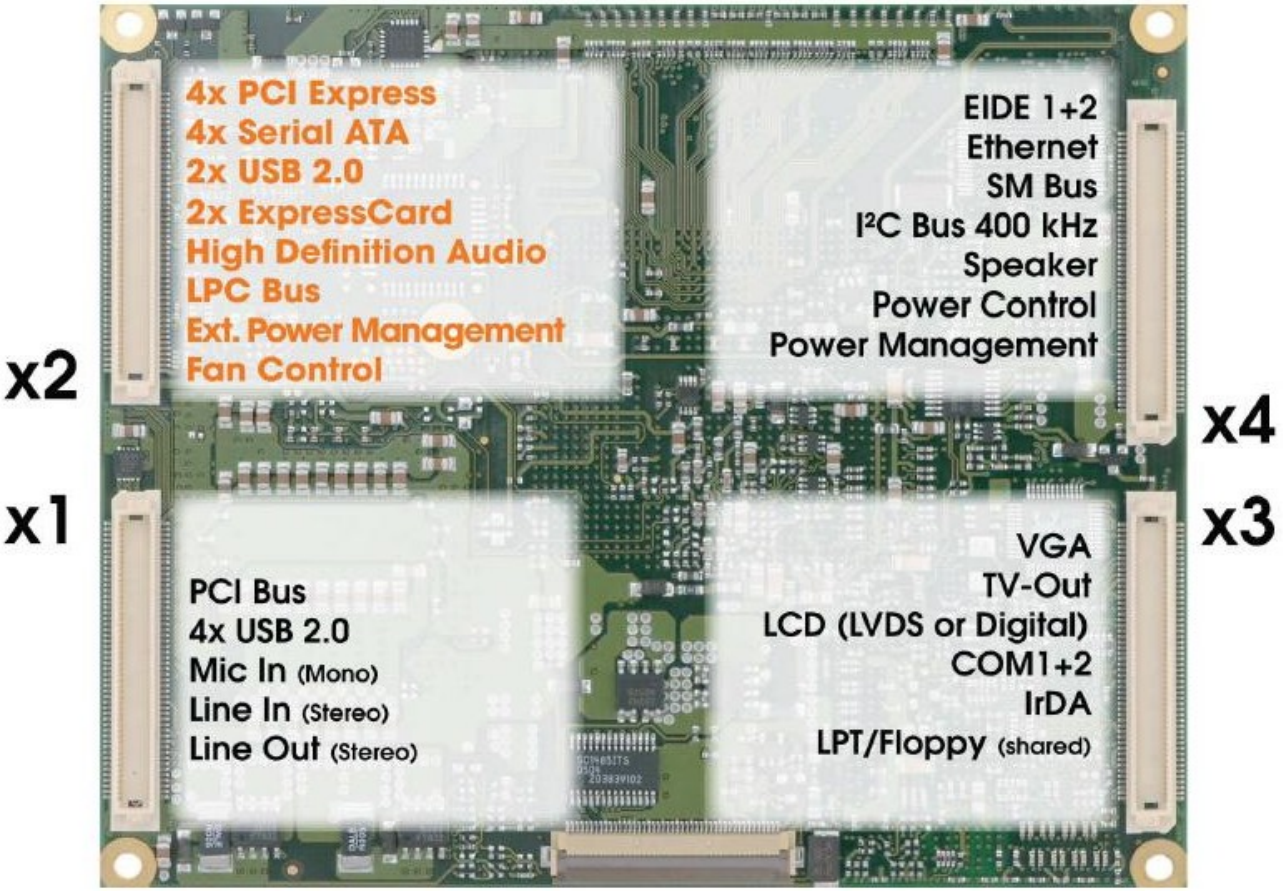
Embeddable PCs

- ETX (Embedded Technology, eXtended)
  - Interfaces:
    - *X1: PCI, USB, Audio*
    - X2: ISA
    - X3: VGA, LCD, Video
      COM1, COM2, IrDA
      LPT/Floppy, PS/2, ...
    - X4: IDE1, IDE2,
      Ethernet
    - optional: SATA
  - Dimensions:
    - 114 x 95 mm



Embeddable PCs

- XTX (eXpress Technology for ETX)
  - Interfaces:
    - *same as ETX subtracts ISA, and adds PCIe, ...*
  - Dimensions:
    - 114 x 95 mm

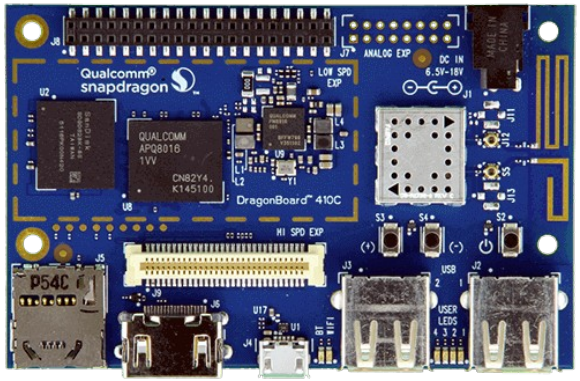


96 Boards

- Defined by Linaro

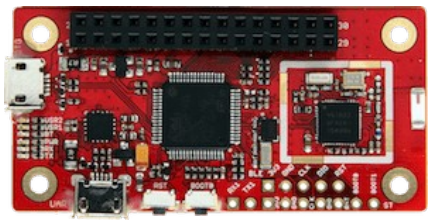
  - Linaro – consortium of companies focused on developing open source software for ARM based CPUs

- Currently defined 3 board form factors
  - 96Boards CE (Consumer Edition) - targets mobile, embedded and digital home segments.
  - 96Boards EE (Enterprise Edition) - targets the networking and server segments.
  - 96Boards IE (IoT Edition) - supports development in the Internet of Things (IoT).



CE

EE

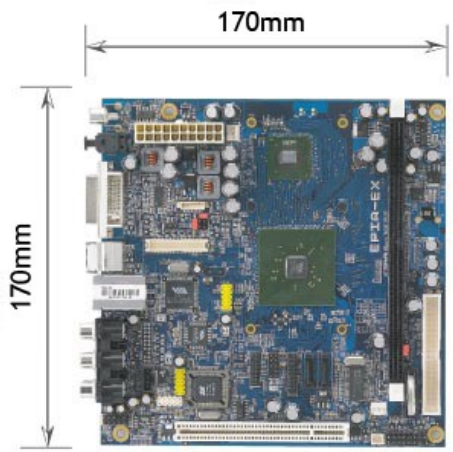


IE

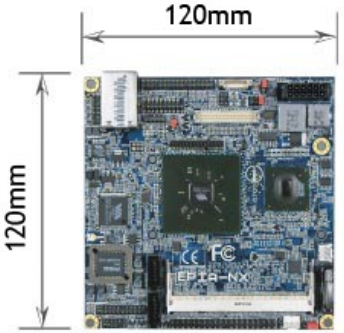
Embeddable PCs

- Traditional formats of PC type computers

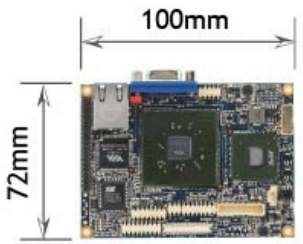
Name	PCB Size (mm)
WTX	356×425
AT	350×305
Baby-AT	330×216
BTX	325×266
ATX	305×244
LPX	330×229
NLX	254×228
microATX	244×244
DTX	244×203
FlexATX	229×191
Mini-DTX	203×170
EBX	203×146
microATX (Min.)	171×171
Mini-ITX	170×170
EPIC (Express)	165×115
Nano-ITX	120×120
COM Express	125×95
ETX / XTX	114×95
Pico-ITX	100×72
PC/104 (-Plus)	96×90
mobile-ITX	75×45



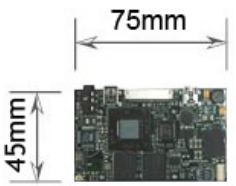
Mini-ITX



Nano-ITX



Pico-ITX



Mobile-ITX

Embeddable PCs

- EOMA-68 (Embedded Open Modular Architecture)
  "bring simple robust mass-produced CPU Cards to end-users"

  - PCMCIA physical form-factor → 68 pin interface (re-purposed interface!)
  - Mandatory interfaces:
    - 24-pin RGB/TTL (for LCD Panels and DVI/VGA/HDMI or other display conversion ICs)
    - I^2C
    - USB (Low Speed, Full Speed, optionally Hi Speed/480 Mbit/s and optionally USB3)
    - 10/100 Ethernet (optionally 1,000 Ethernet)
    - SATA-II (optionally SATA-III)
    - 8 pins of General-purpose Digital I/O (GPIO with multiplexing to SD/MMC and SPI on 6 pins)
    - SD/MMC (multiplexed with 6 of the GPIO pins)
    - TTL-compatible UART (Tx and Rx only)

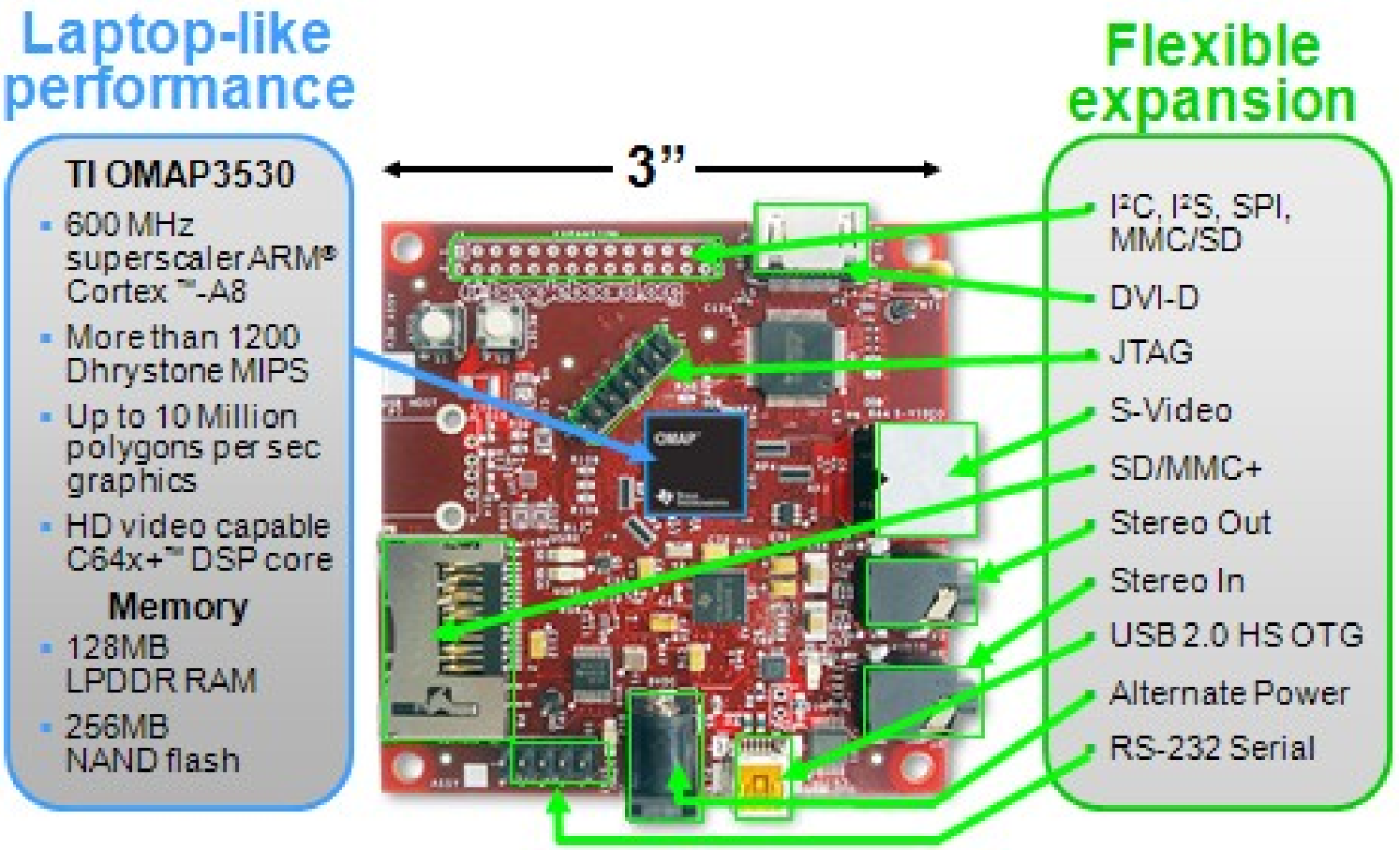
Embeddable PCs

- PicoTux

Processor:	32-bit ARM 7
Processor Clock:	55 MHz
Flash Memory:	2 MB
RAM:	8 MB SDRAM
Ethernet:	10/100 Mbit, HD and FD
Serial (TTL):	Up to 230.400 bps
General Input/Output Pins(TTL):	5, (also for Handshake)
LED for Ethernet:	2; green (programmable) and yellow (Carrier)
Supply Voltage:	3,3 Volt +- 5%
Supply Current:	250 mA
Operating System:	uClinux 2.4.27
Shell:	Busybox 1.0 and others
File Systems:	CRAMFS, JFFS2, NFS
Applications:	Webserver, Telnet
Size of the Linux Systems in Flash:	720 KB and more
Protected Bootloader for Update over Network:	64 KB Code
Dimensions:	19 mm x 19 mm x 36 mm
Weight:	ca.18 g
Ambient Temperature:	-40°C to 85°C

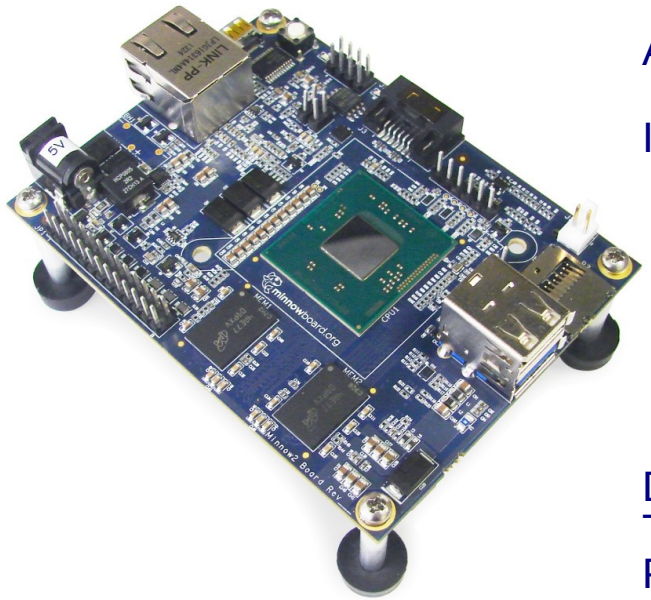
Embeddable PCs

- BeagleBoard



Embeddable PCs

- MinnowBoardMAX



Core Logic:	64-bit Intel® Atom™ E38xx Series SoC
	$99: E3815 (single-core, 1.46 GHz)
	$129: E3825 (dual-core, 1.33 GHz)
Graphics:	Integrated Intel® HD Graphics
	(micro HDMI connector)
Memory:	DDR3 RAM - System Memory
	($99: 1 GB, $129: 2 GB)
	8 MB SPI Flash - System Firmware Memory
Audio	Digital (via HDM)
	Analog (via MinnowBoard MAX Lure - sold separately)
I/O	1 – Micro SD SDIO
	1 – SATA2 3Gb/sec
	1 – USB 3.0 (host)
	1 – USB 2.0 (host)
	1 – Serial debug (via FTDI cable)
	10/100/1000 Ethernet RJ-45 connector
	8 – Buffered GPIO pins (2 pins support PWM)
	I2C & SPI bus
Dimensions	99 x 74mm (2.9 x 3.9in)
Temperature	0 – 70 deg C
Power	5V DC
System Boot	UEFI Firmware
Operating Systems	
	Debian GNU/Linux, Yocto Project Compatible, Android 4.4 System

Computer on a Board

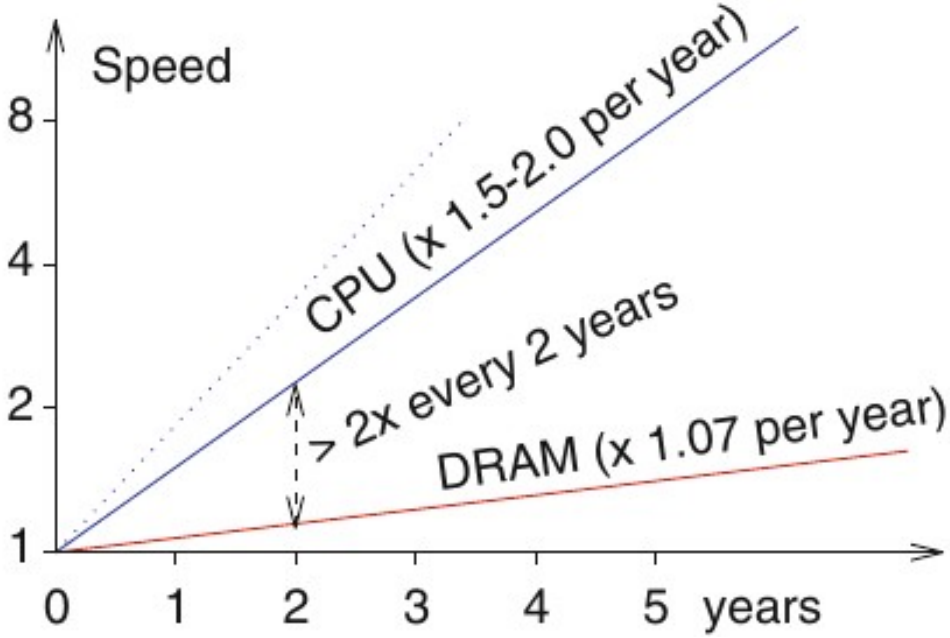
- Others...
  - RaspberryPi (ARM)
  - BananaPi
  - OrangePi
  - Cubieboard
  - ODROID
  - Mbed (ARM)
  - ALIX (AMD x86)
  - ESP32
  - ...

Hardware Architectures for Embedded Systems

- Processors / Execution of Control Logic
- Boards / Form Factors
- Memory Technologies
- Interface / Communication Buses

Memory Speed

- Up to 2003, CPU speeds increased faster than memory speed
- CPU frequencies have saturated since 2003 but speed diference remains
- Recent mutli-core CPUs also increase required memory bandwidth
- Difference in speeds results in a "**memory wall**"



Memory Hierarchy

- Work around the "memory wall" problem
  → use hierarchy of caches

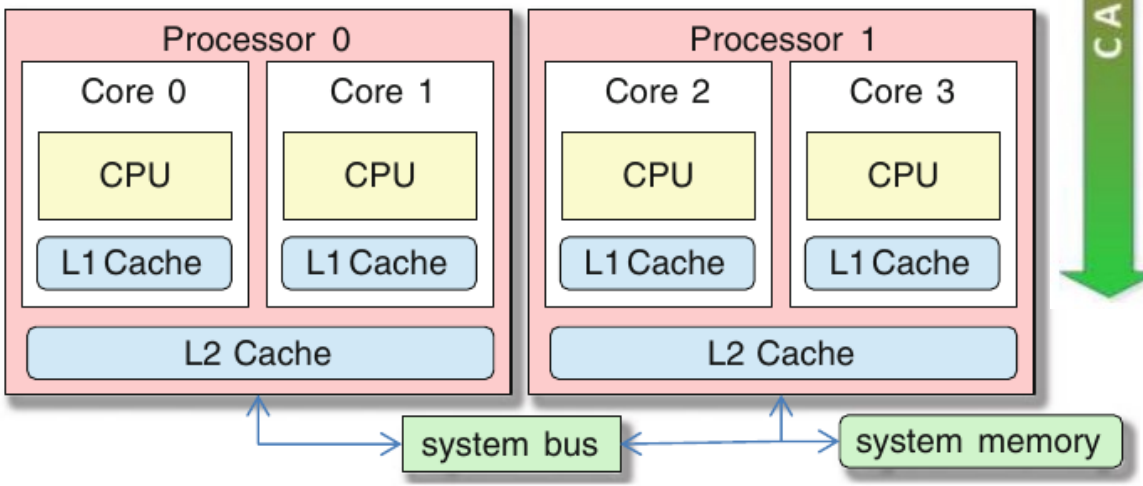


Image: "Embeded systems", Peter Marwedel

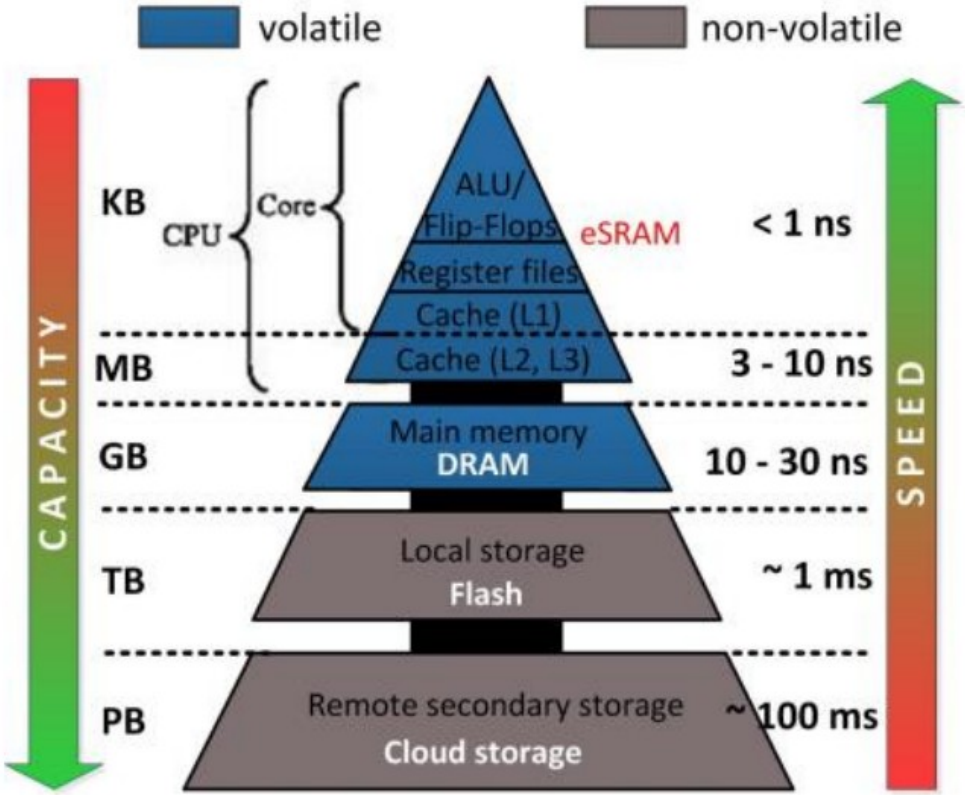
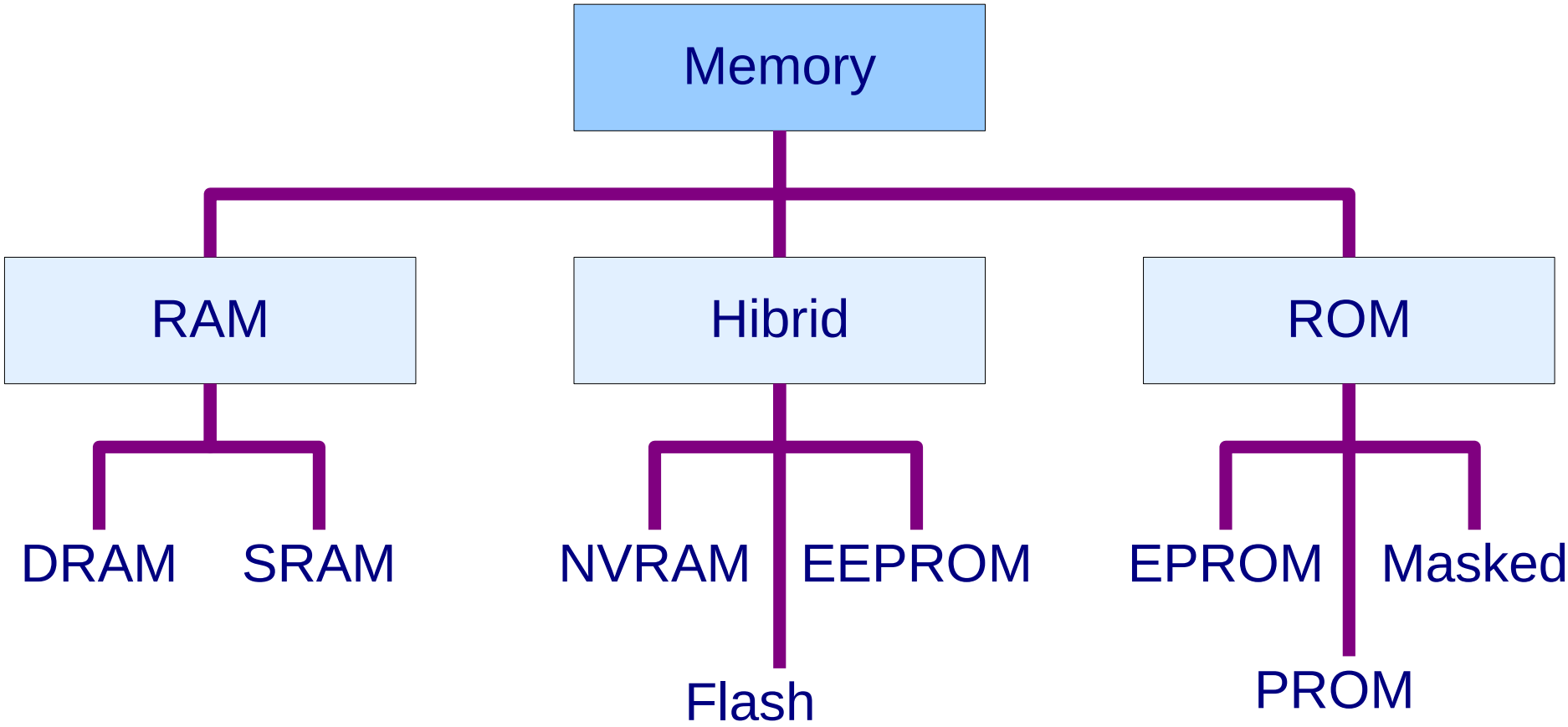


Image: Jovanovic, Bojan et al.. (2015) "MTJ-based hybrid storage cells for "normally-off and instant-on" computing". Facta universitatis - series: Electronics and Energetics. 28. 465-476. 10.2298/FUEE1503465J.

Memory Technologies



Memory Technologies

- RAM
  - loses data when powered off
  - SRAM faster than DRAM
  - SRAM more expensive DRAM
  - DRAM requires refresh (based on charge of capacitor, with leakage current)
- ROM
  - maintains data when powered off
  - Masked ROM: data is programmed during manufacturing process
  - PROM: programmable, once, after manufacture.
  - EPROM: once programmed, may be erased using ultra-violet light
  - EPROM more expensive than PROM

Memory Technologies

- Hibrid
  - do not lose data when powered off
  - EEPROM, Flash:
    - may be electrically erasable
    - EEPROMS – possible to erase individual bytes;
    - Flash – erase by sectors [typically from 256 to 16kBytes]
    - fast read, but writing slower than RAM
    - NAND Flash: writing by sector – may replace hard disks
    - EEPROM more expensive than FLASH
  - NVRAM (Non Volatile RAM):
    - SRAM with backup battery
    - when powered, works like SRAM
      (similar working characteristics as SRAM)

Memory Technologies

Type	*Volatile?*	*Writeable?*	*Erase Size*	*Max Erase Cycles*	*Cost (per Byte)*	*Speed*
SRAM	Yes	Yes	Byte	Unlimited	Expensive	Fast
DRAM	Yes	Yes	Byte	Unlimited	Moderate	Moderate
Masked ROM	No	No	n/a	n/a	Inexpensive	Fast
PROM	No	Once, with a device programmer	n/a	n/a	Moderate	Fast
EPROM	No	Yes, with a device programmer	Entire Chip	Limited (consult datasheet)	Moderate	Fast
EEPROM	No	Yes	Byte	Limited (consult datasheet)	Expensive	Fast to read, slow to erase/write
Flash (NOR)	No	Yes	Sector	Limited (consult datasheet)	Moderate	Fast to read, slow to erase/write
Flash (NAND)	No	Yes – by sector	Sector	Limited (consult datasheet)	Moderate	Fast to read, slow to erase/write
NVRAM	No	Yes	Byte	Unlimited	Expensive (SRAM + battery)	Fast

[Michael Barr] , *adapted...*

Memory Technologies

Test data bus:

- write data with all bits set to 0, except 1 (walking 1)

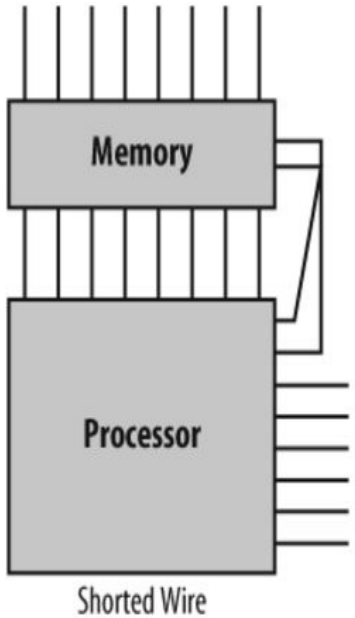
Test address bus:

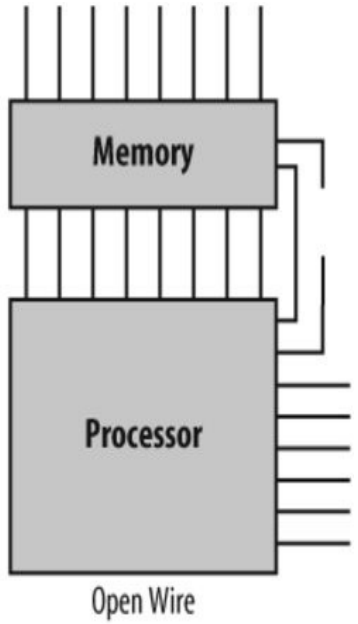
- write same value (ex. 0xFF) to addresses with only 1 bit set to 1.
- change the data in one of those Addresses, and check that no Other value was changed.

Test control lines:

- test each byte with at least 2 distinct (preferably complementary values).

Beware of caches!





Validating Memory

- Non-Parity (8 bits)
- Parity (8 bits + 1 parity bit)
- ECC (8 bits + 1 ecc bit)
  - ECC: Error code correction
  - Group bytes in 32 / 64 bits: use remaining bits for error codes (CRC)
  - Can detect and correct one error bit.
  - Used mostly in servers!
  - NOTE: use of memory modules where more than one bit is stored in the same chip (i.e. With less than 9 chips) increases the odds of errors on more than one bit!

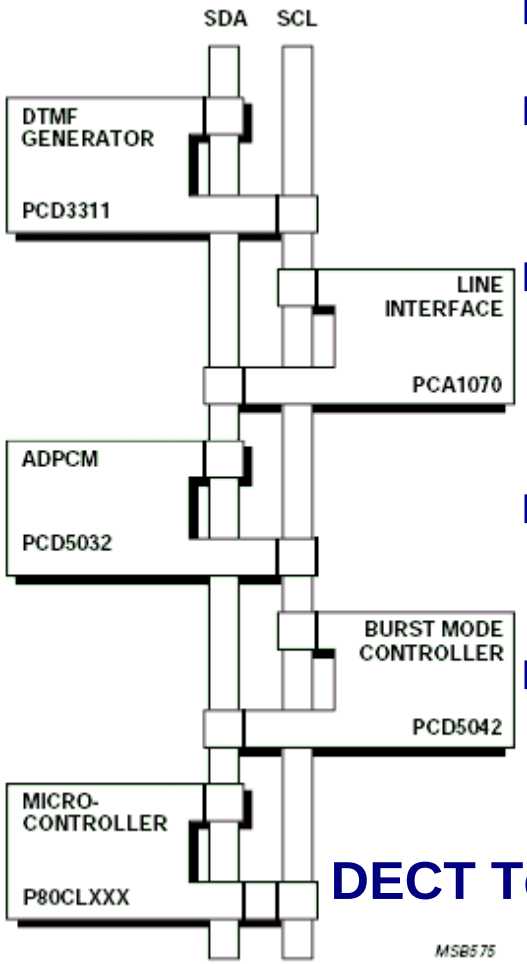
Hardware Architectures for Embedded Systems

- Processors / Execution of Control Logic
- Boards / Form Factors
- Memory Technologies
- Interface / Communication Buses

Interface / Communication Buses

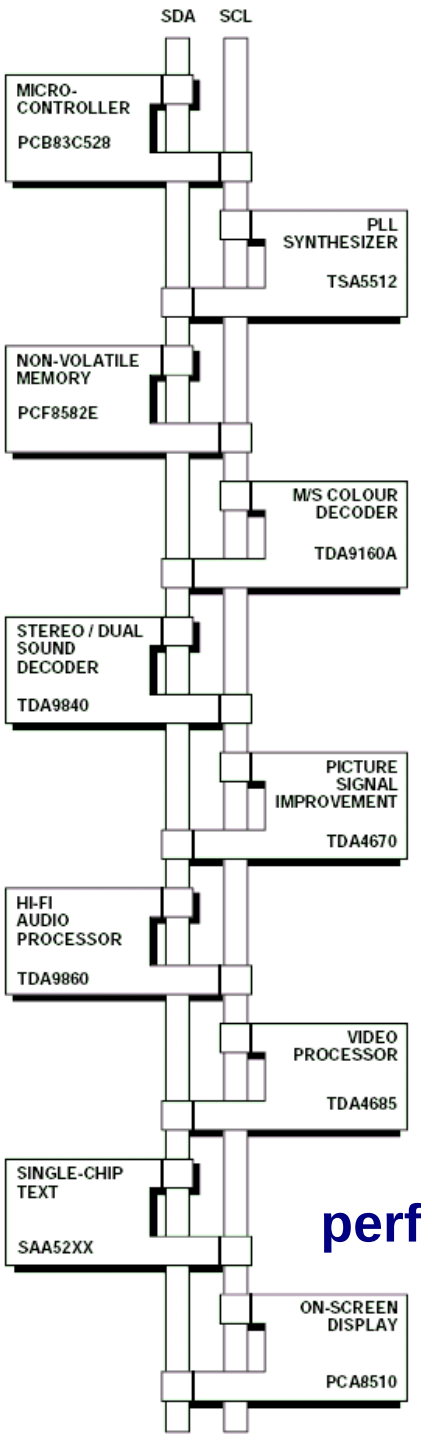
- I^2C – Inter-Integrated Circuit
- SMBus
- CBus
- SPI – Serial Peripheral Interface
- USB – Universal Serial Bus
- ZigBee
- RS232 / RS422 / RS 485
- CAN
- JTAG

I2C - Overview



DECT Telefone

- Uses 2 lines + ground
- Microcontrollers, LCDs, remote I/O, RAM, EEPROM, sensors
- Allows the reconfiguration of a PCB without changing the traces (change to new ICs)
- Can quickly pass from block level design to implementation
- Does not require interface hardware between ICs



high performance TV

[figures: I2C specification ver. 2.1]

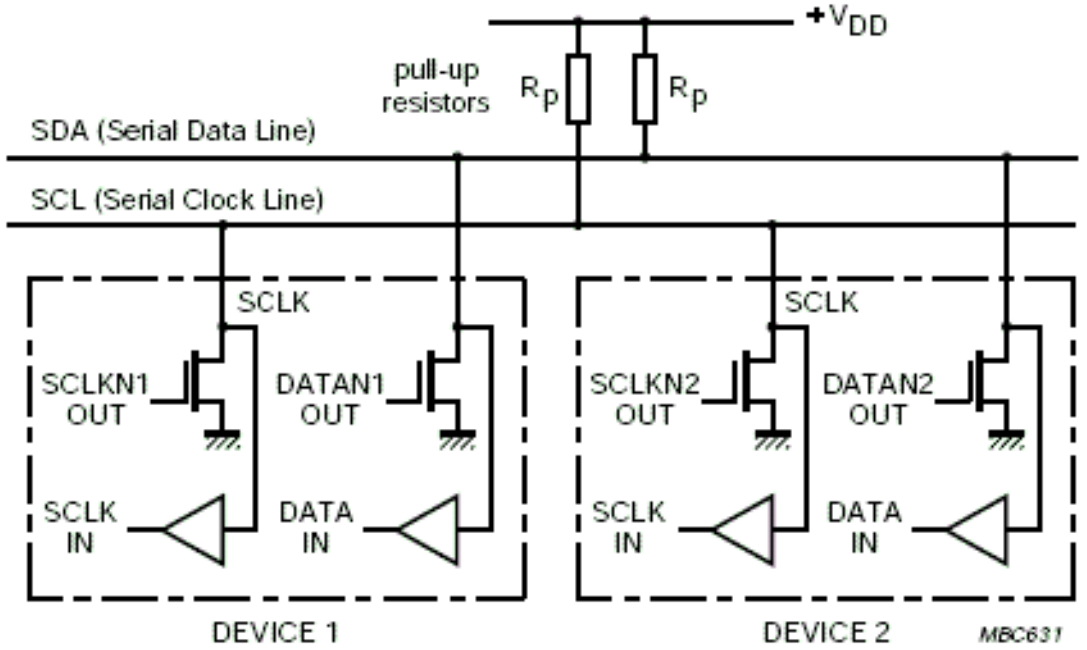
I2C - Overview

TERM	DESCRIPTION
Transmitter	The device which sends data to the bus
Receiver	The device which receives 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 winning message is not corrupted
Synchronization	Procedure to synchronize the clock signals of two or more devices

- Each IC on the bus is directly addressable
- Uses master/slave interaction model
- Supports multi-master
- Master initiates data transfer
- In each transfer sequence, both the master and the slave may be the emitter and receiver.

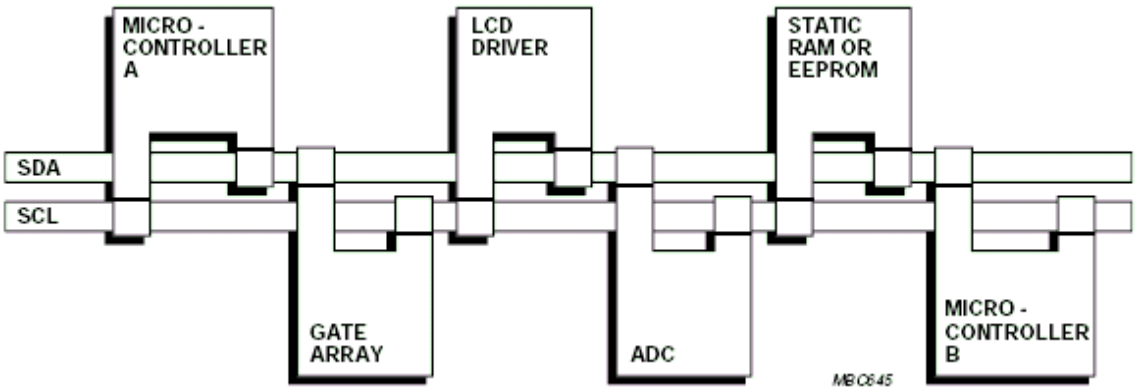
I2C – Physical Layer

- Hardware interface to interface lines uses open/collector outputs
- Lines work as Wired-AND
- SDA: Data line
- SCL: Clock line
- Both lines are bi/directional
- Number of devices on the lines is limited by a maximum of 400pF on the line (each device is 10pF)



I2C - Arbitration

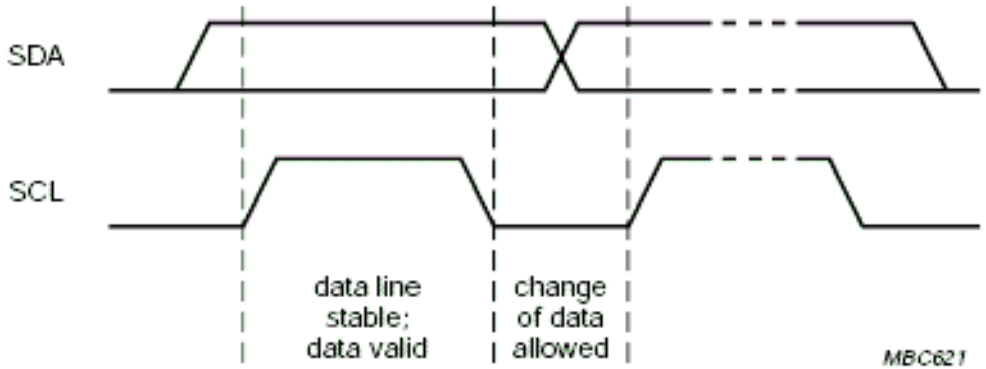
- Medium access control
  - can detect a collision when two or more masters are transmitting simultaneously.
  - collision is non destructive for the master that writes 0 to the bus.
  - The last master gains access to the medium.

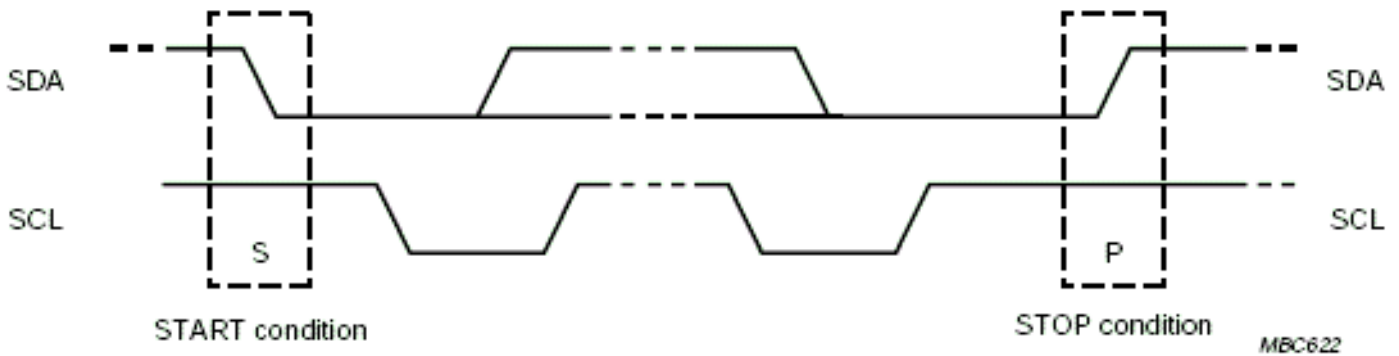


I2C Multi-Master Bus

I2C – Bit Transmission

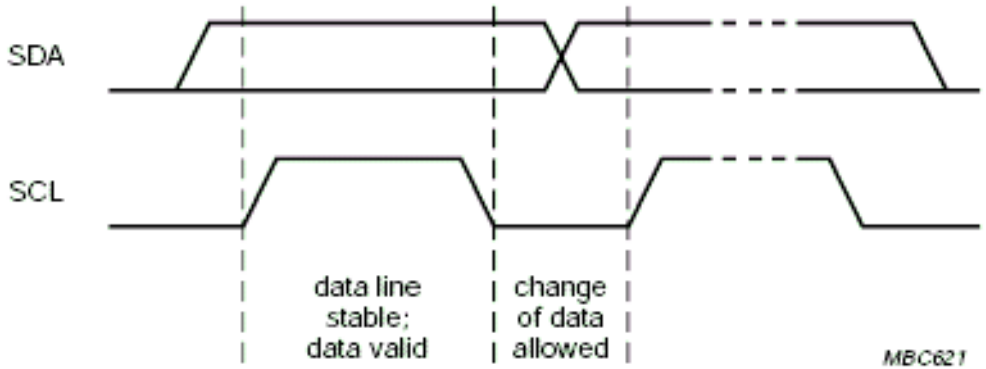
- SCL at 1 → stable data
  SCL at 0 → data is changing
- SDA never changes state when SCL is at 1, except when:
  - Marks the beginning of a frame (START)
  - Marks the end of a frame (STOP)
- START and STOP: only generated by the master

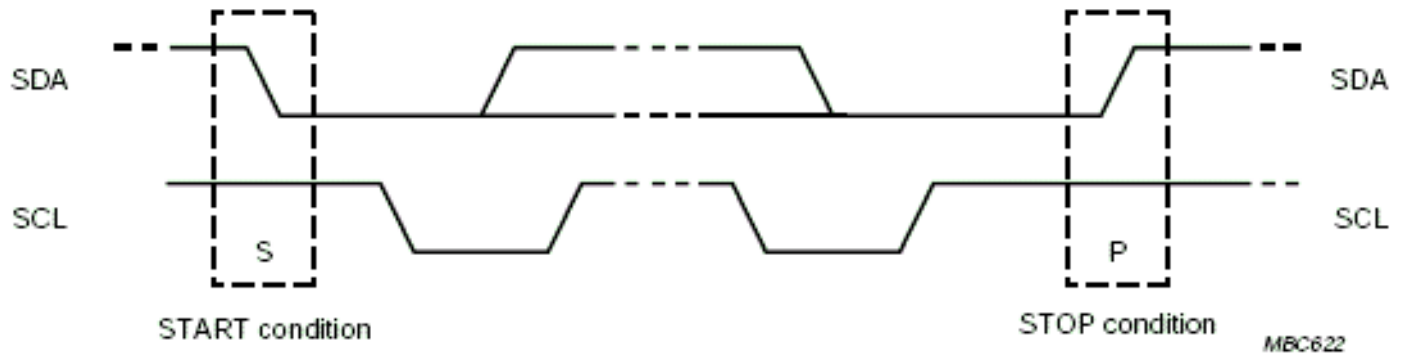




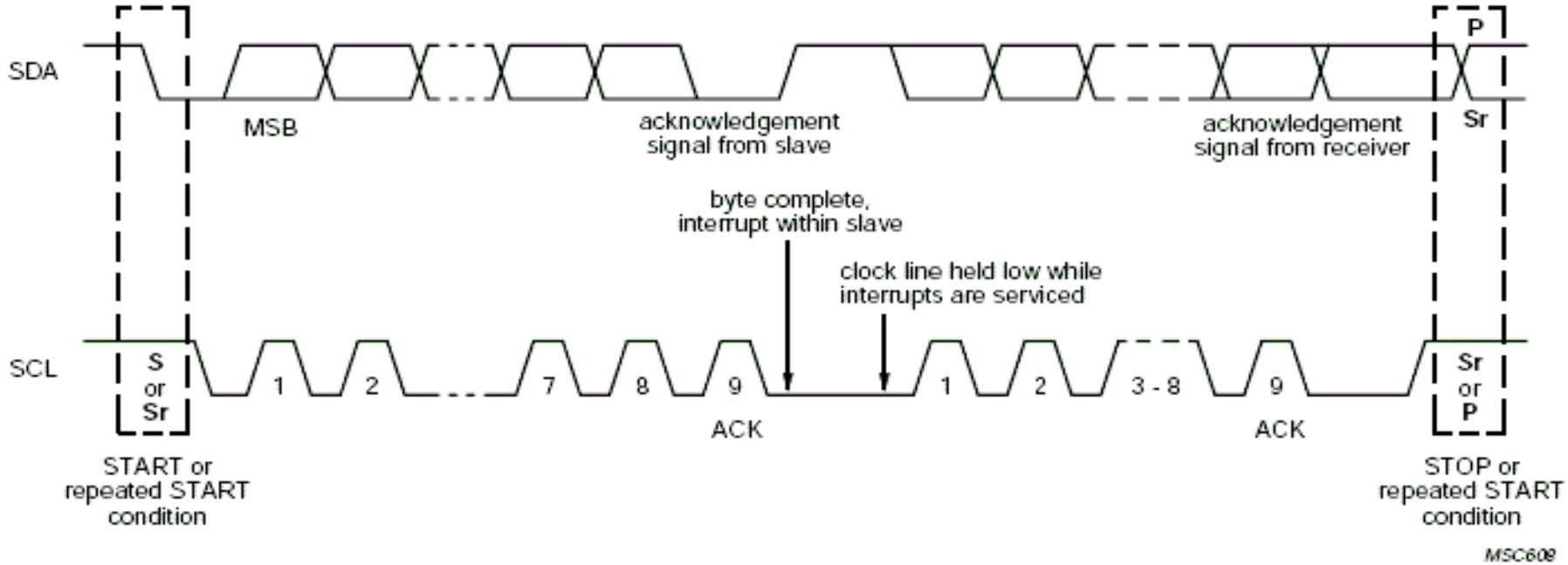
I2C – Bit Transmission

- Master can emit another START at the end of a transfer sequence, to start a second sequence without sending a STOP for the first.
- After a START the bus is considered occupied
- After a STOP the bus is considered free



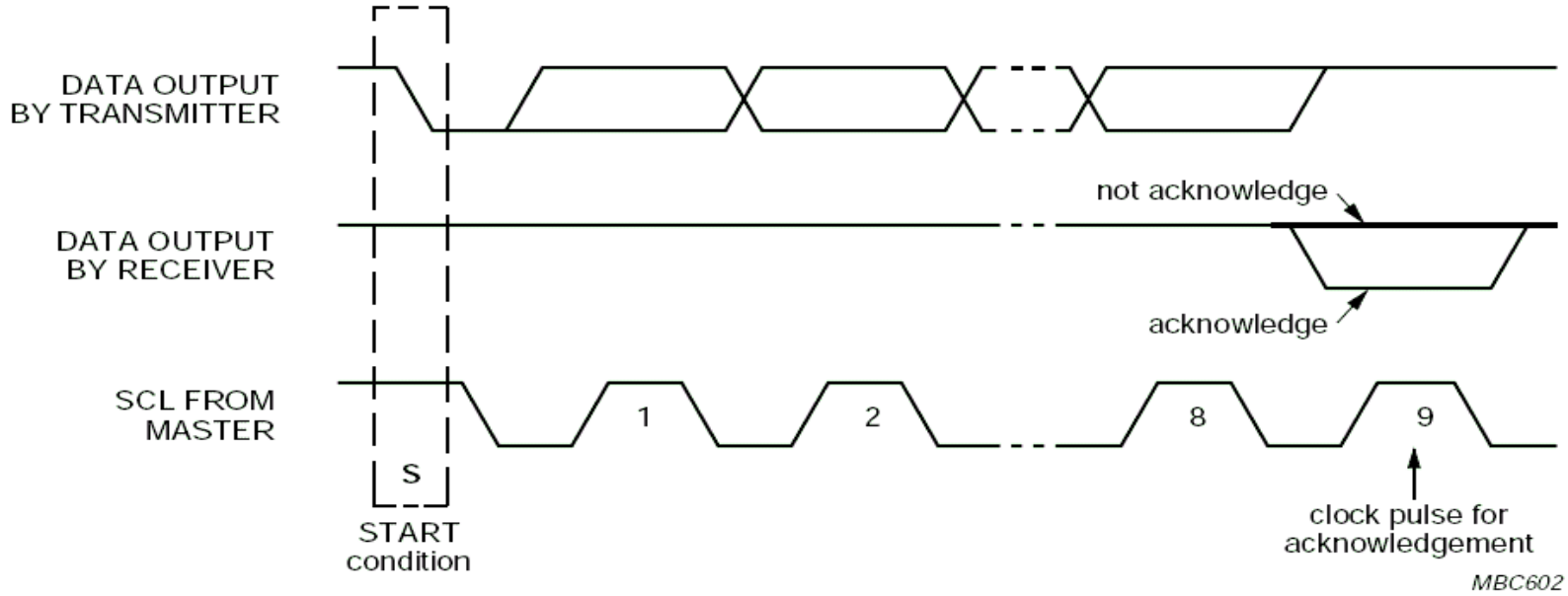


I2C – Data Transmission



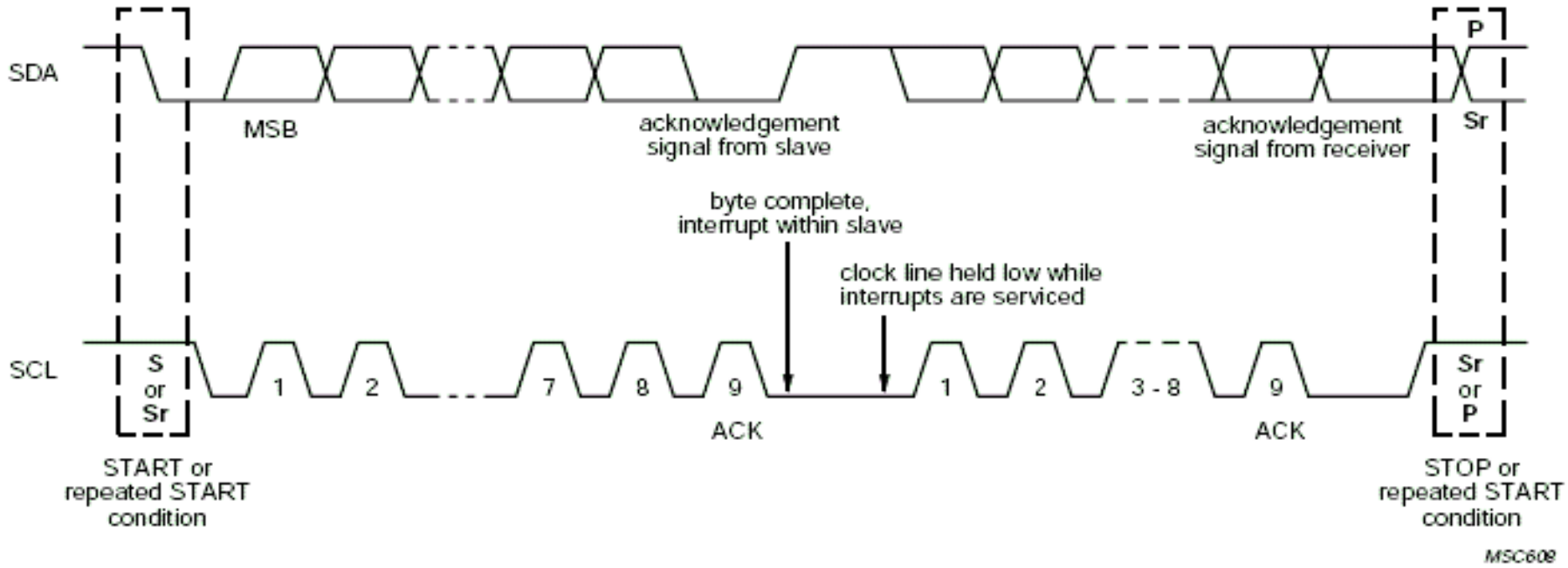
- Data is always transmitted in 8 bit sequences (MSB first)
- Transmission Speed:
  - Standard: 0 to 100 kbit/s
  - Fast: 100 to 400 kbit/s
  - High-Speed: up to 3,4 Mbits/s

I2C – Data Transmission



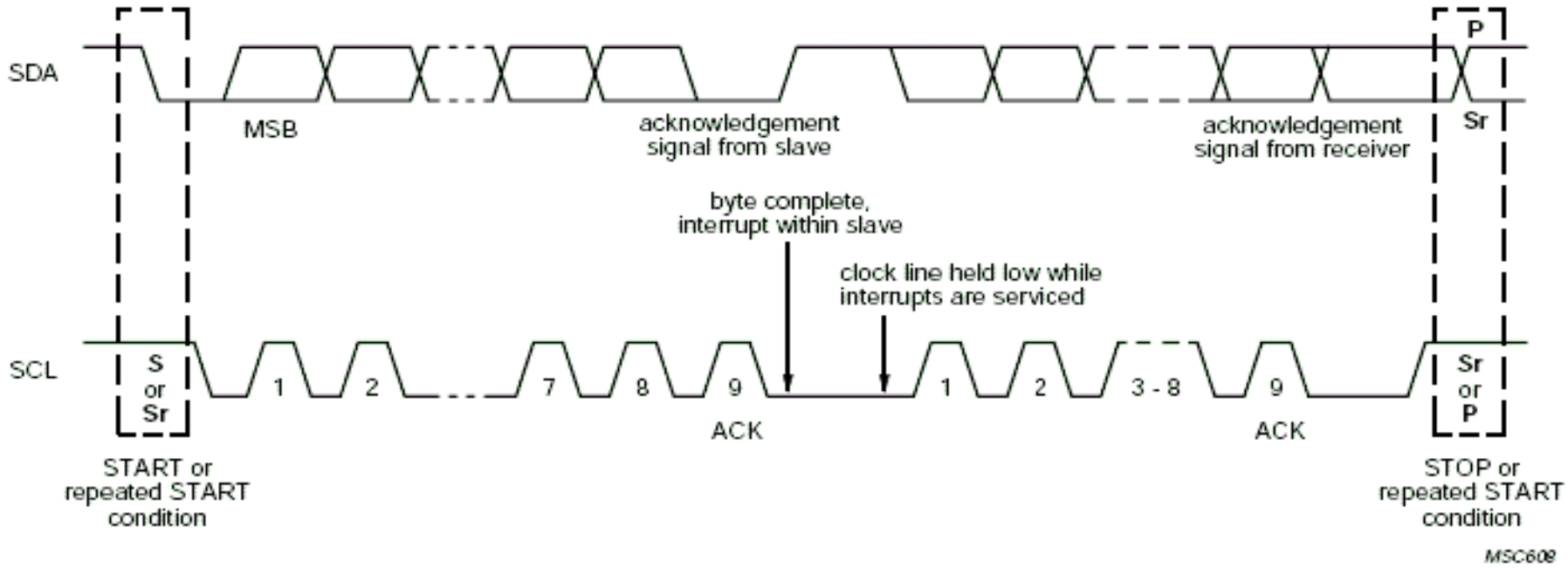
- At the end of each byte (8 bit sequence), the receiver must generate an Acknowledge bit
  - *Acknowledge* is active 0
  - Receiver may be either the master or the slave
  - When the master is the receiver, it must generate a negative acknowledgement at end of sequence/frame

I2C – Data Transmission



- A slave may delay the transmission of the second (or later) bytes:
  - by forcing SCL to 0 !!
  - as the master cannot force SCL to 1 (due to the wired AND), it will be forced to delay its data transmission!!

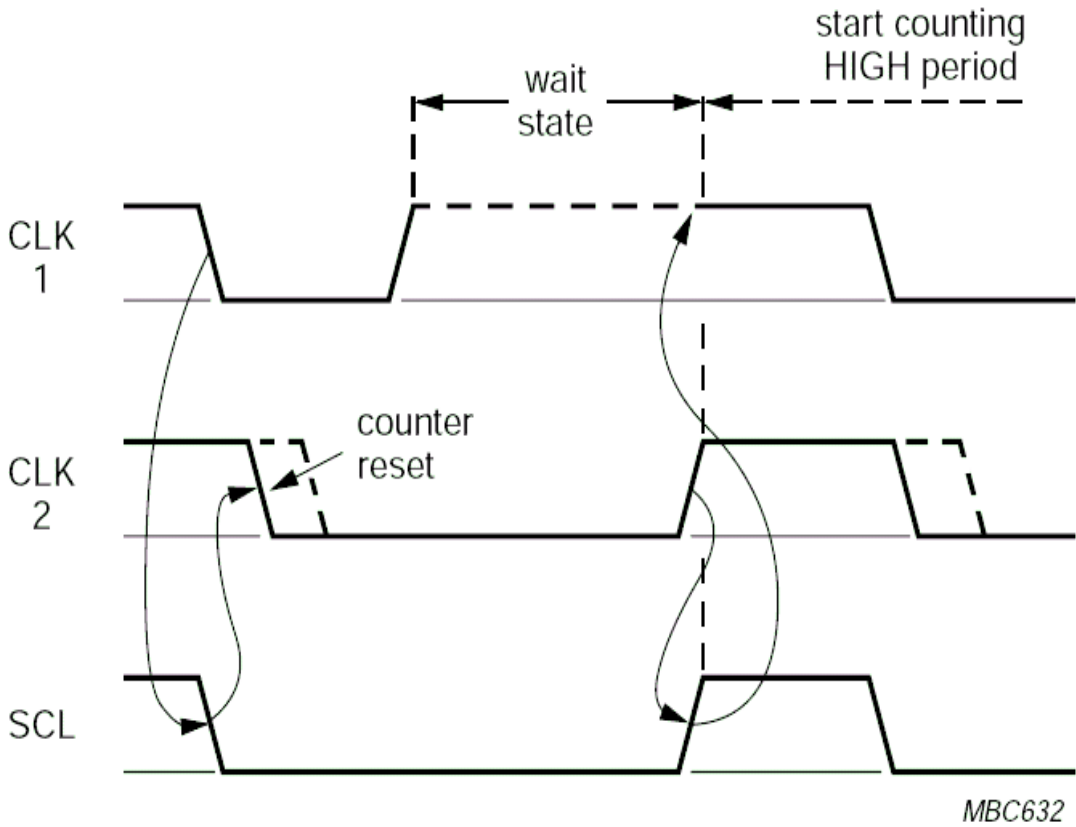
I2C – Data Transmission



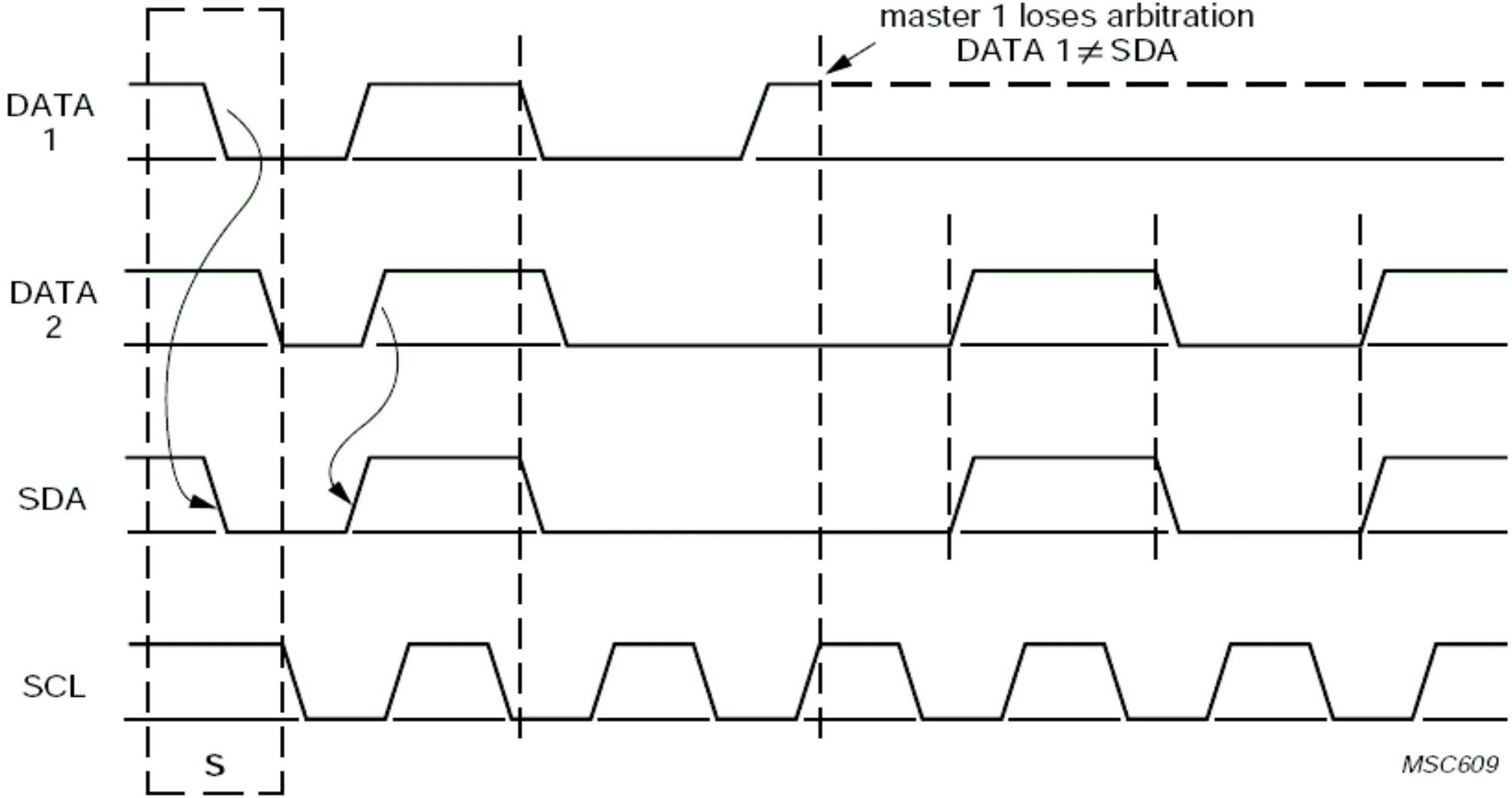
- The clock signal SCL is always generated by the master (independent of whether the master is the emitter or receiver!)
  - In the case of Master/Receiver, the Slave will be the Emitter.

I2C - Arbitration

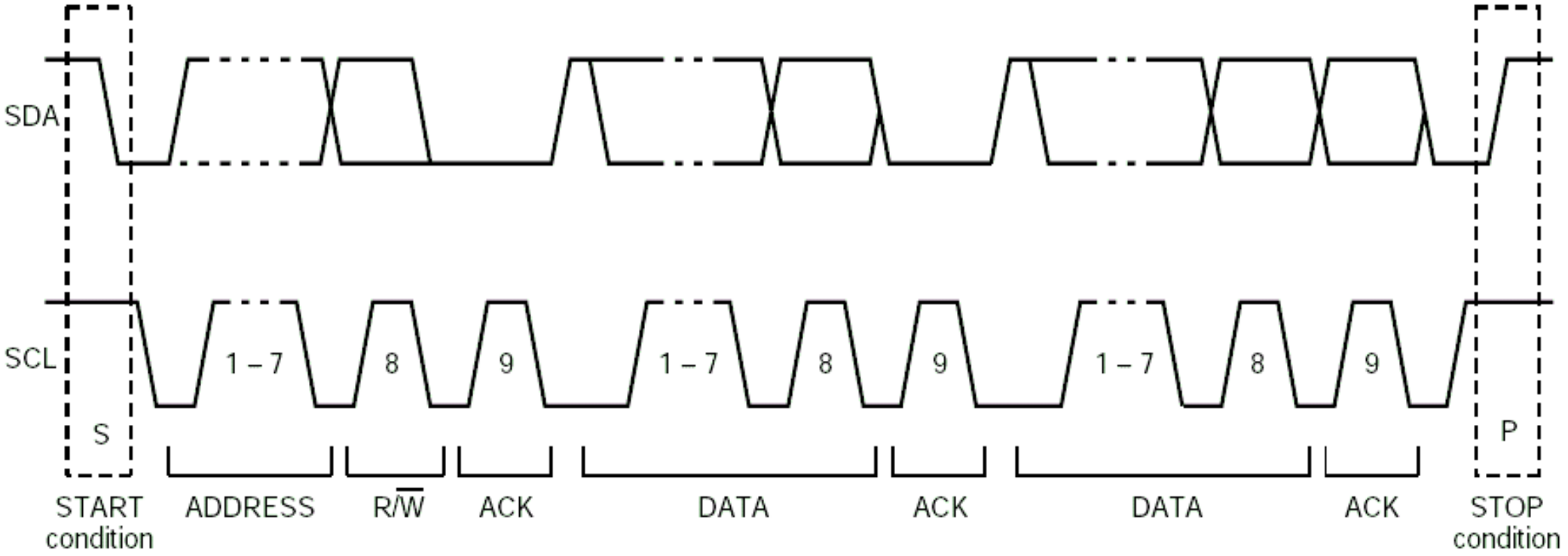
- During arbitration (i.e several masters transmitting simultaneously) these synchronize the clocks generated on the SCL
- Arbitration occurs on the SDA line:
  - Each master that attempts to transmit a 1 but reads a 0, aborts frame transmission
  - When aborting transmission, it immediately passes to receiving mode



I2C - Arbitration



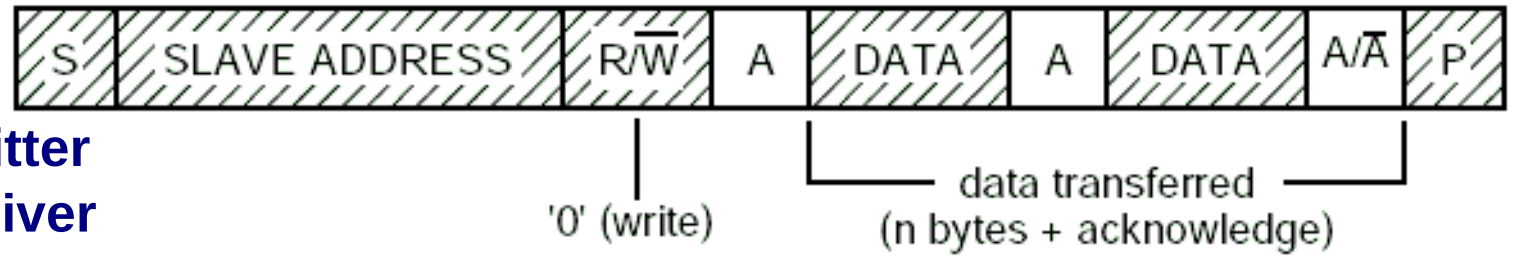
I2C – Frame Format

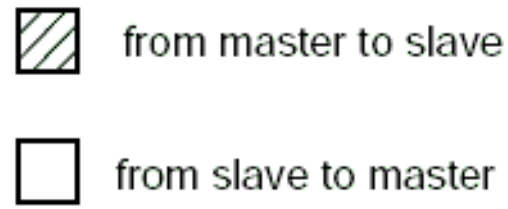


- 7 address bits
- 1 Read / Write bit (Active on Read)
- Variable number of bytes

I2C – Frame Format

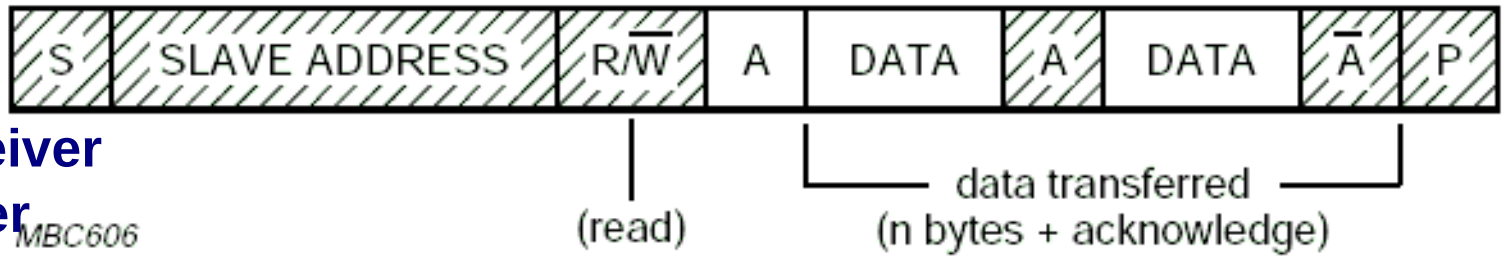
Master → Emitter
Slave → Receiver





A = acknowledge (SDA LOW)
Ā = not acknowledge (SDA HIGH)
S = START condition
P = STOP condition

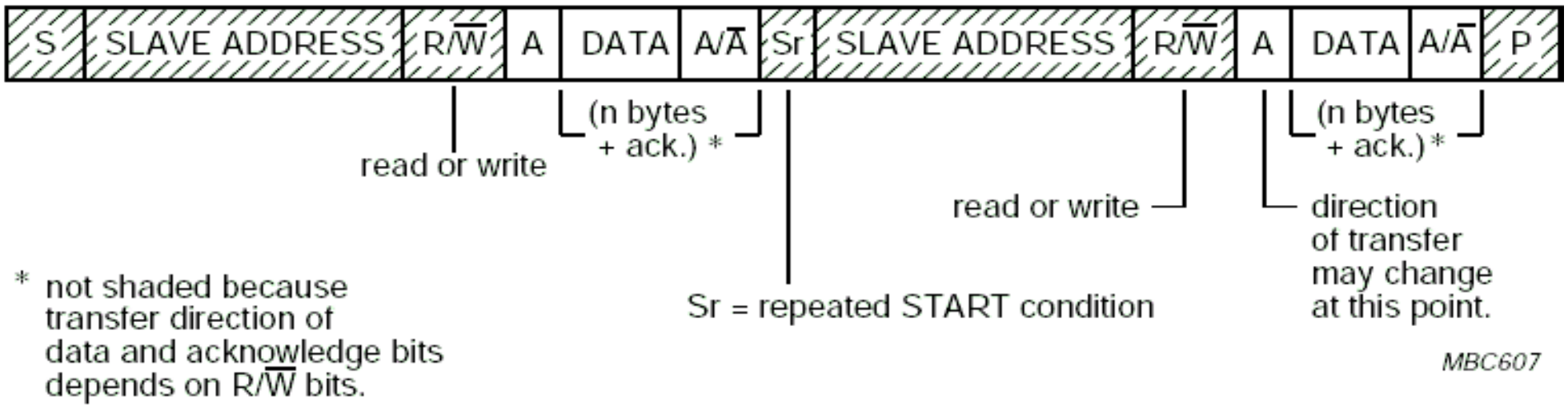
Master → Receiver
Slave → Emitter



MBC606

I2C – Frame Format

- Reading and/or writing to a memory (RAM, EEPROM, FLASH, etc...) generally requires more than one sequence:
  - one sequence for the address
  - another sequence for the data
  - Operating characteristics of memory is not defined by I2C (ex. Is the address auto-incremented?)



I2C – Address Format

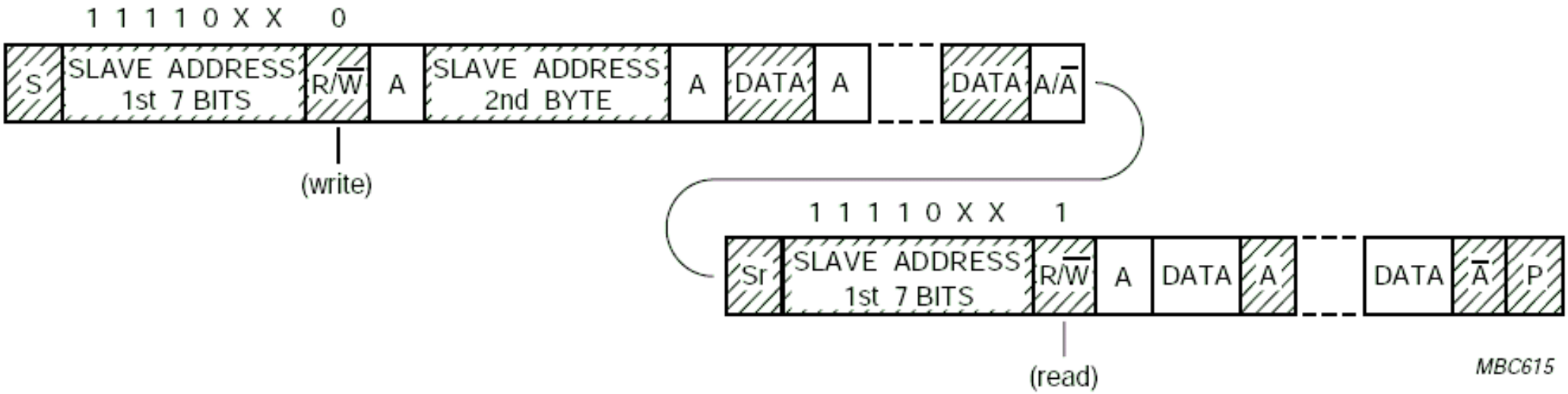
7 bit addresses

Table 2 Definition of bits in the first byte

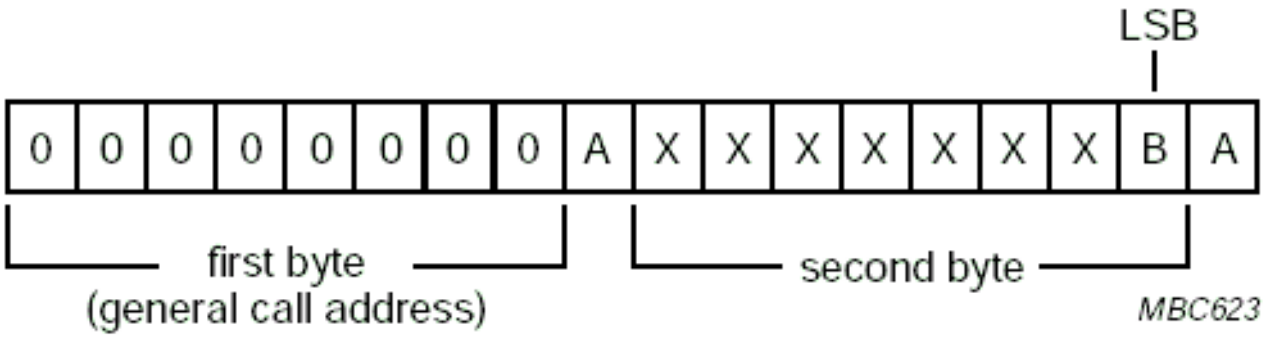
SLAVE ADDRESS	R/$\overline{W}$ BIT	DESCRIPTION
0000 000	0	General call address
0000 000	1	START byte(1)
0000 001	X	CBUS address(2)
0000 010	X	Reserved for different bus format(3)
0000 011	X	Reserved for future purposes
0000 1XX	X	Hs-mode master code
1111 1XX	X	Reserved for future purposes
1111 0XX	X	10-bit slave addressing

I2C – Address Format

Use of 10 bit Addresses

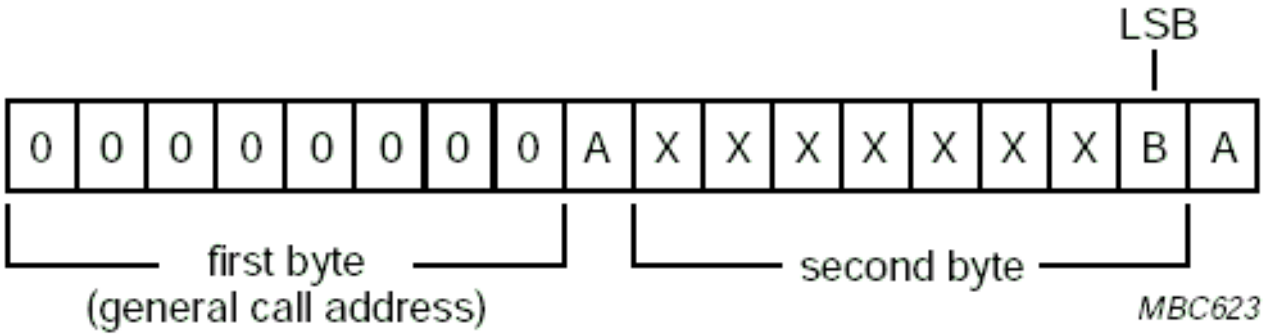


I2C – General Call (Broadcast)

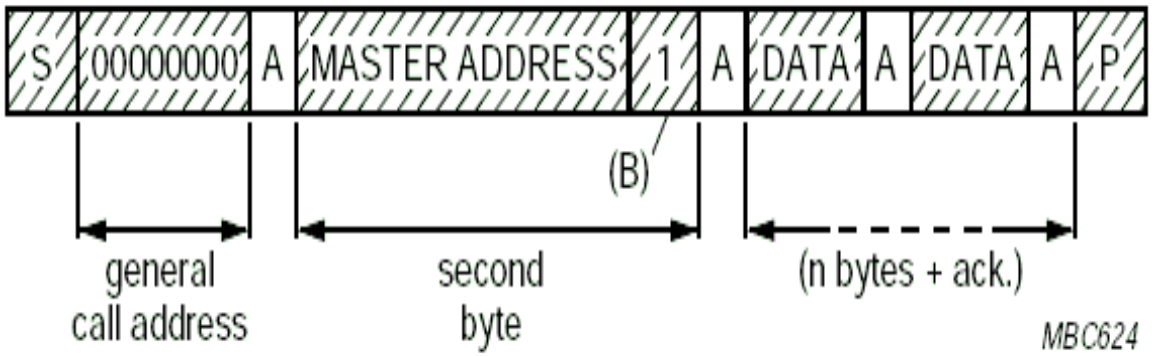


- B = 0
  - Second byte = 0x06
    Reset Device + Reset Address
  - Second byte = 0x04
    Reset Address
  - Second byte = other values
    ILEGAL
- B = 1
  - Used by masters that wish to broadcast information.
    (ex. Keyboard reader)
  - Second Byte
    identifies the sending device

I2C – General Call (Broadcast)



- B = 1
  - Used by masters that wish to broadcast information. (ex. Keyboard reader)
  - Second Byte identifies the sending device

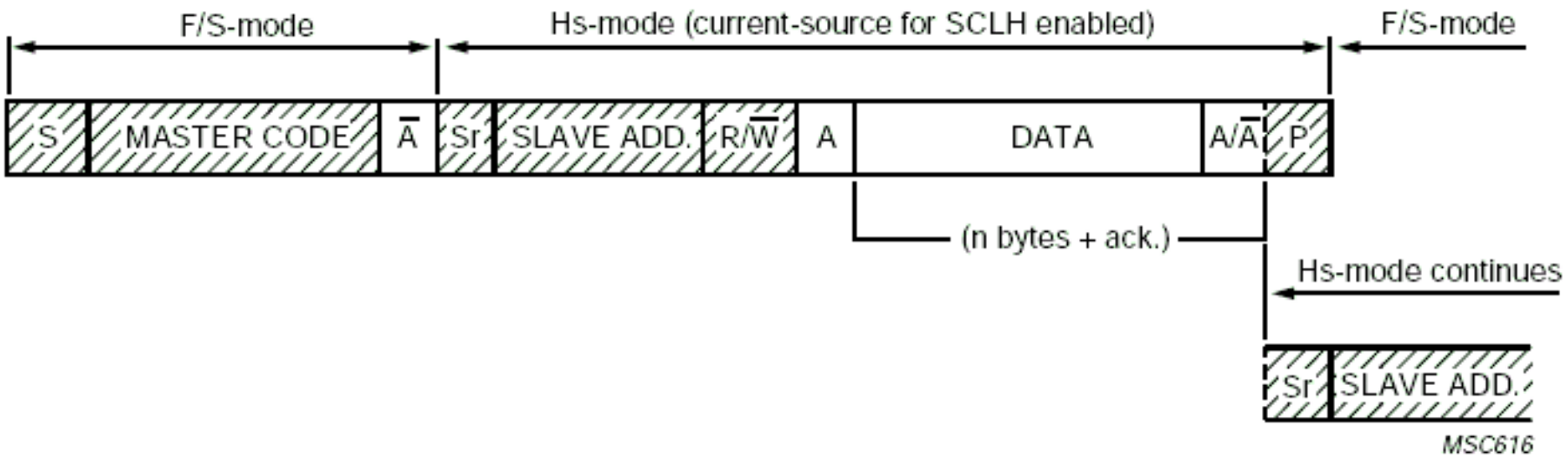


I2C – Transmission Speeds

- Standard
  - 0 to 100 kbits/s
- Fast
  - 0 to 400 kbits/s
  - devices also compatible with Standard Mode
- High Speed
  - up to 3,4 Mbits/s
  - devices also compatible with Standard and Fast Modes

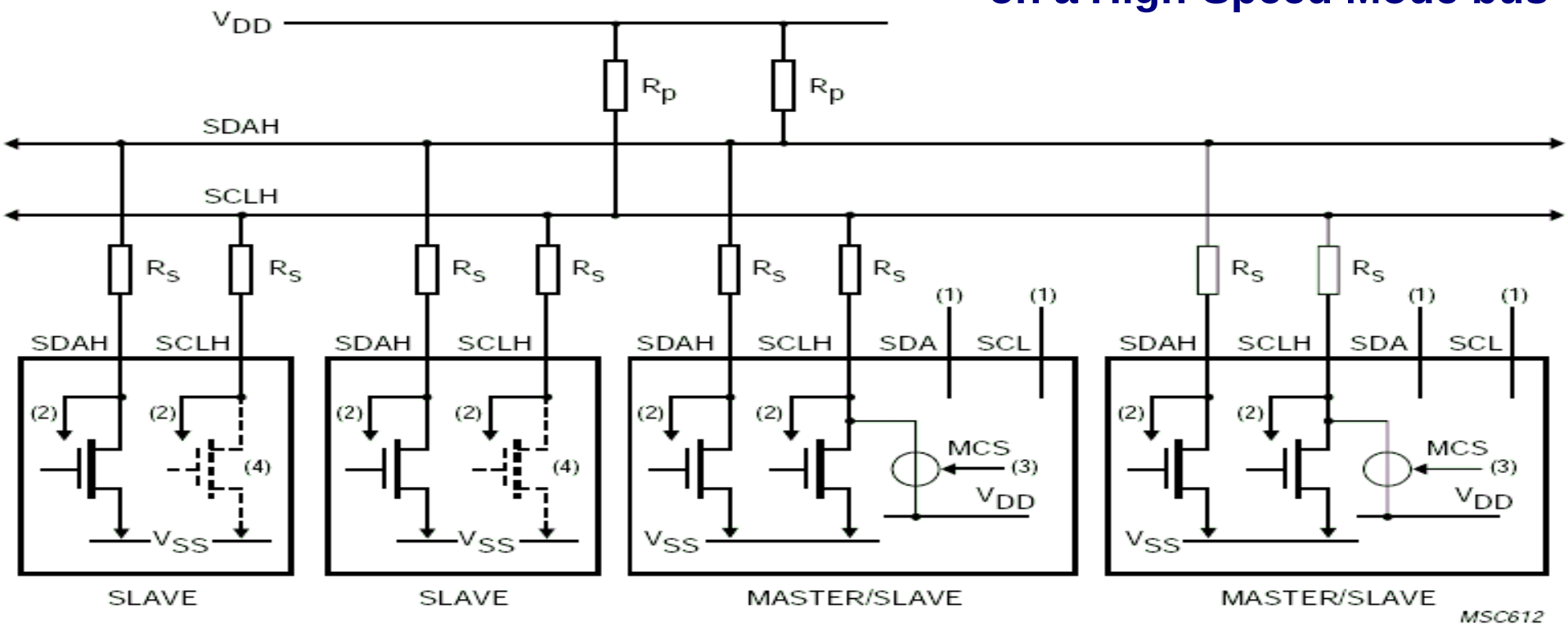
- Transmission in Fast mode is identical to Standard Mode, only quicker
- Transmission in High-Speed mode only starts after arbitration occurs in either Standard or Fast mode!

I2C – High-Speed Transmission



I2C – Multi-Speed Buses

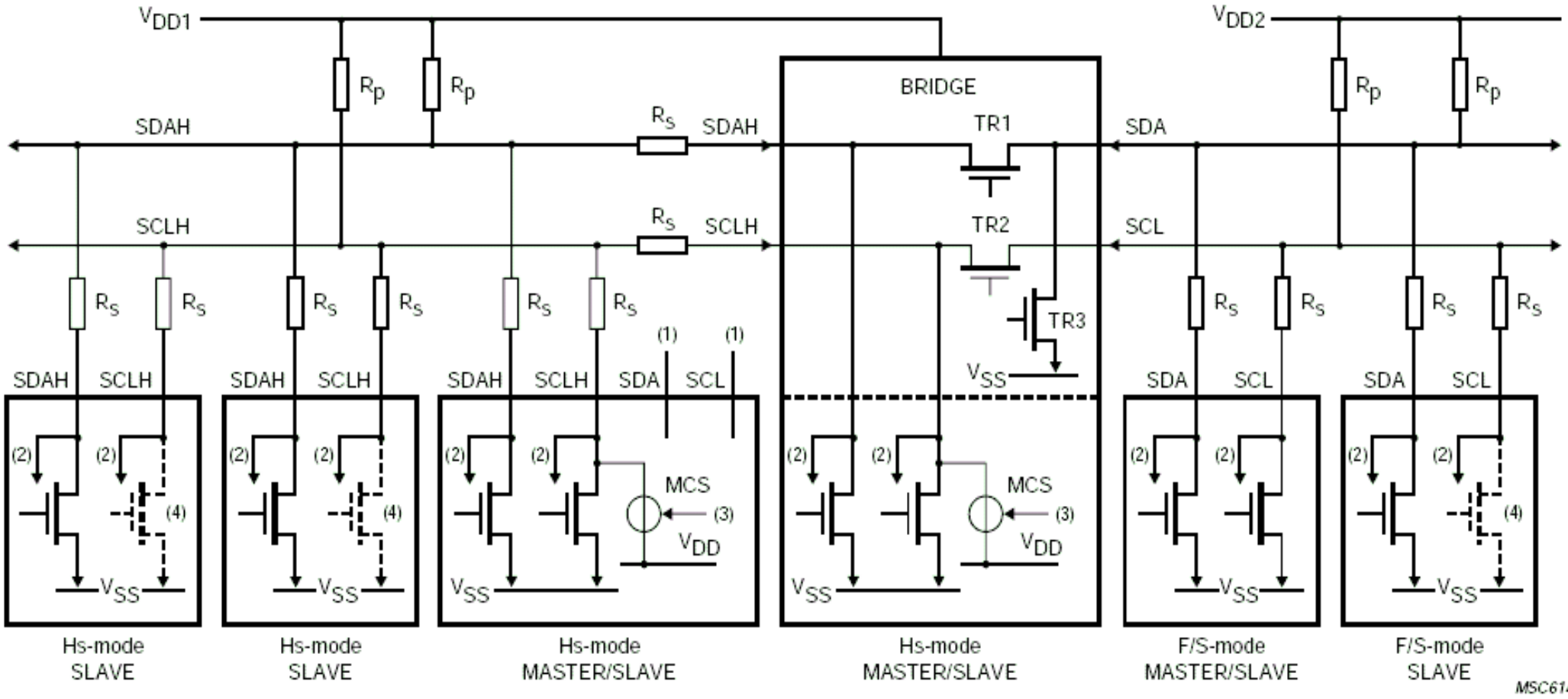
High-Speed devices on a High-Speed Mode bus



(1) SDA and SCL are not used here but may be used for other functions.
(2) To input filter.
(3) Only the active master can enable its current-source pull-up circuit
(4) Dotted transistors are optional open-drain outputs which can stretch the serial clock signal SCLH.

I2C – Multi-Speed Buses

Mixed devices on a mixed speed bus



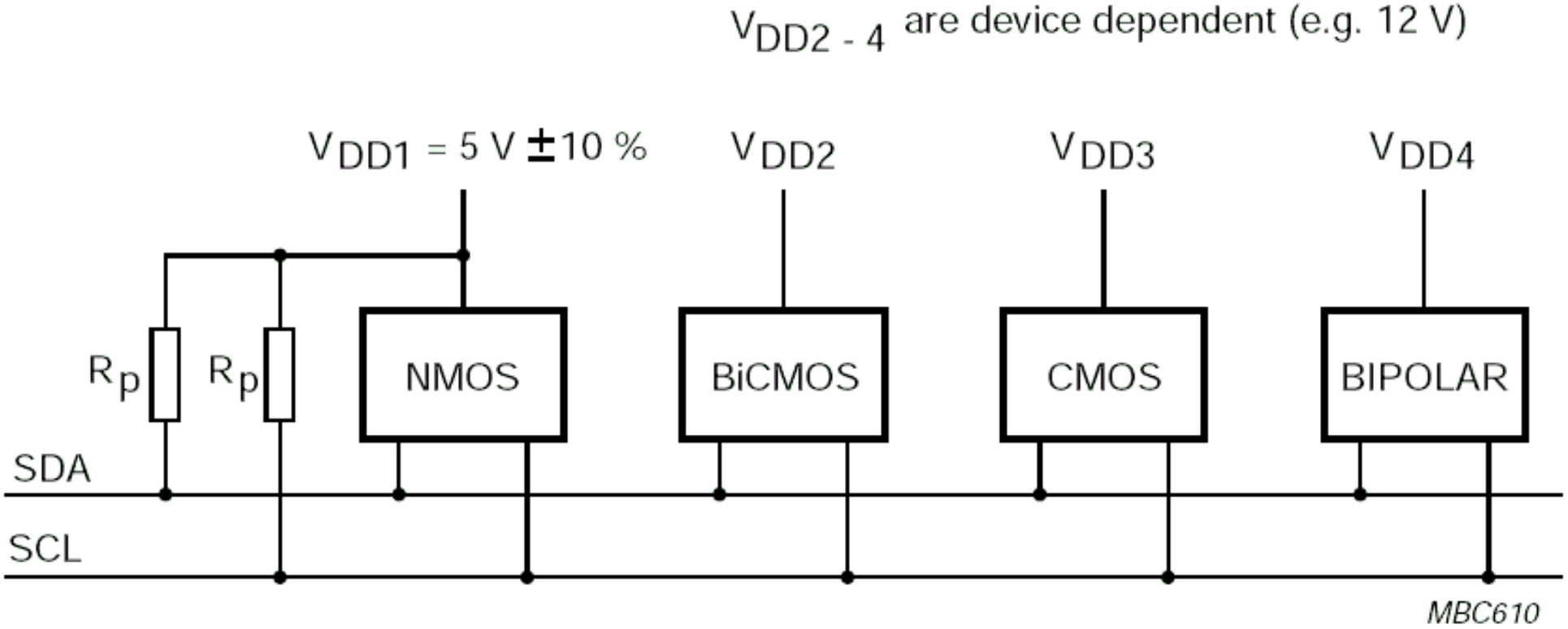
I2C – Multi-Speed Buses

Mixed devices on a mixed speed bus

TRANSFER BETWEEN	SERIAL BUS SYSTEM CONFIGURATION			
	Hs + FAST + STANDARD	Hs + FAST	Hs + STANDARD	FAST + STANDARD
Hs <–> Hs	0 to 3.4 Mbit/s	0 to 3.4 Mbit/s	0 to 3.4 Mbit/s	–
Hs <–> Fast	0 to 100 kbit/s	0 to 400 kbit/s	–	–
Hs <–> Standard	0 to 100 kbit/s	–	0 to 100 kbit/s	–
Fast <–> Standard	0 to 100 kbit/s	–	–	0 to 100 kbit/s
Fast <–> Fast	0 to 100 kbit/s	0 to 400 kbit/s	–	0 to 100 kbit/s
Standard <–> Standard	0 to 100 kbit/s	–	0 to 100 kbit/s	0 to 100 kbit/s

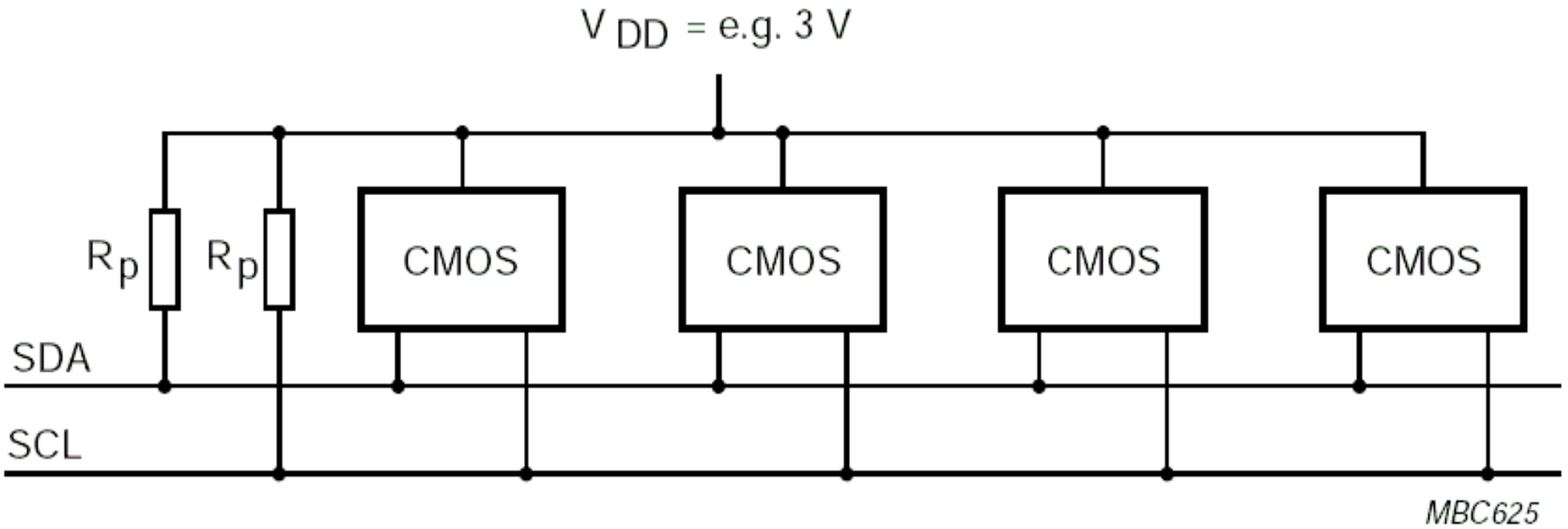
I2C – Bus Power Supply

Devices with fixed bus voltage
(and differing power supply voltages).



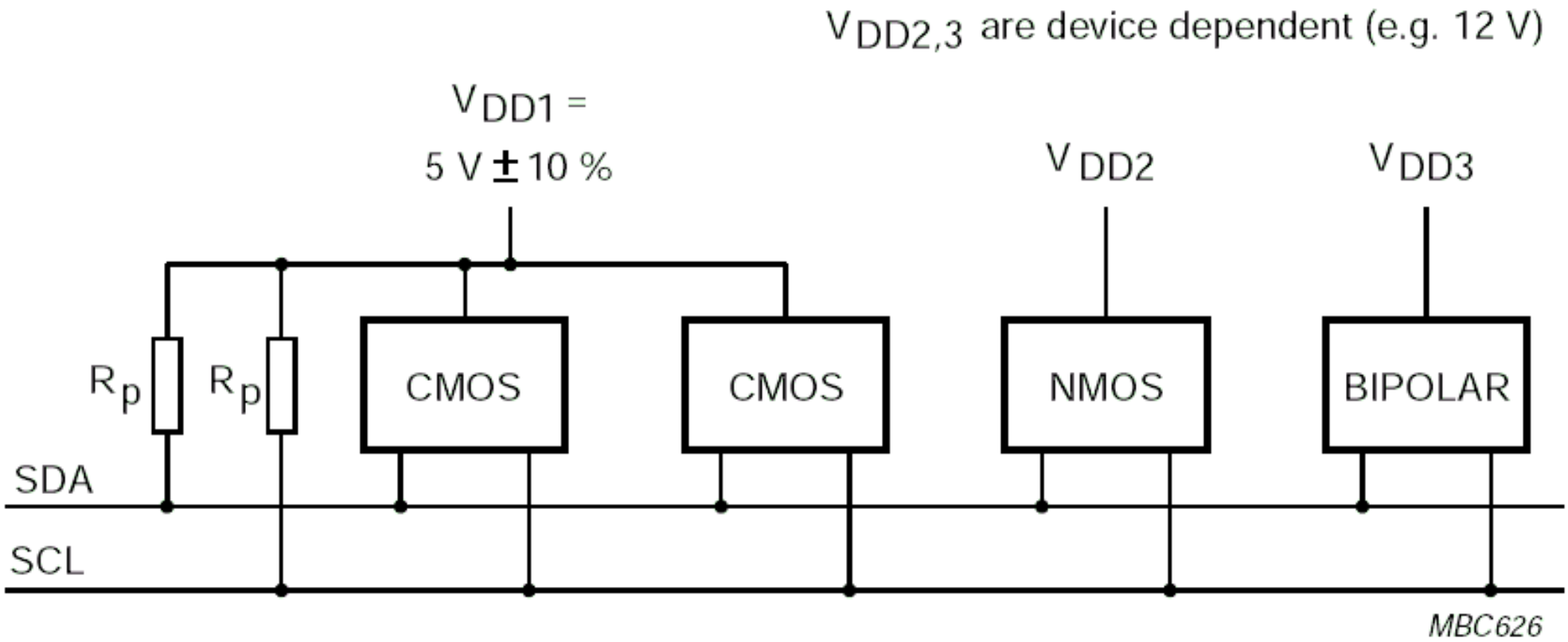
I2C – Bus Power Supply

All devices with the same variable bus voltage (bus voltage dependant on power supply voltage).



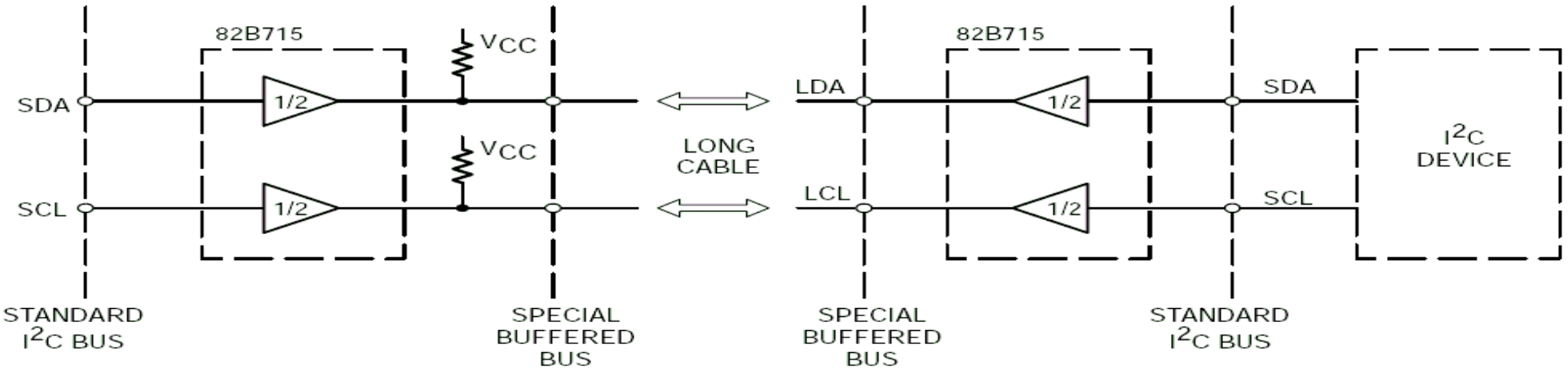
I2C – Bus Power Supply

Some devices with variable bus voltage, sharing bus with other devices using fixed voltage bus.



I2C – Practical details...

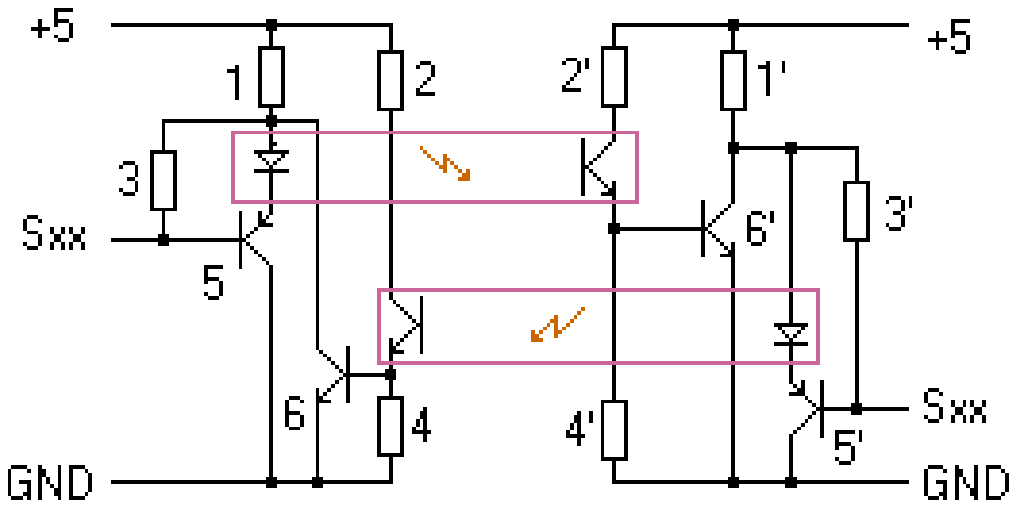
- Maximum bus length
  - typically limited from 2 to 3 meters
  - very low speeds => longer distances (ex. 100m @ 500hz)
  - use of active pull-ups allows longer distances (ex. **LTC1694**)
  - use of 'repeaters'
    (ex. Phillips **P82B715** : transforms voltage signal into current signal, lines with max. 2nF).



[Phillips: P82B715 Datasheet]

I2C – Practical details...

- Galvanic isolation between devices
  - difficult due to bi-directionality of lines
  - A possible opto-coupler based circuit...
    (must use fast opto-couplers for 100 khz bus.)



Component Values:

5 and 5' : PNP like 2n2219 or BC557
6 and 6' : NPN like 2n2222 or BC547
1 and 1' : 270 Ohm
2 and 2' : 3300 Ohm
3 and 3' : 1800 Ohm
4 and 4' : 1000 Ohm
Optocouplers : 6n139 , 4n27 or Til 111

[http://www.esacademy.com/faq/i2c/q_and_a/faq/i2cqa3.htm]

Interface / Communication Buses

- I2C – Inter-Integrated Circuit
- SMBus
- CBus
- SPI – Serial Peripheral Interface
- USB – Universal Serial Bus
- ZigBee
- RS232 / RS422 / RS 485
- CAN
- JTAG

SMBus - Overview

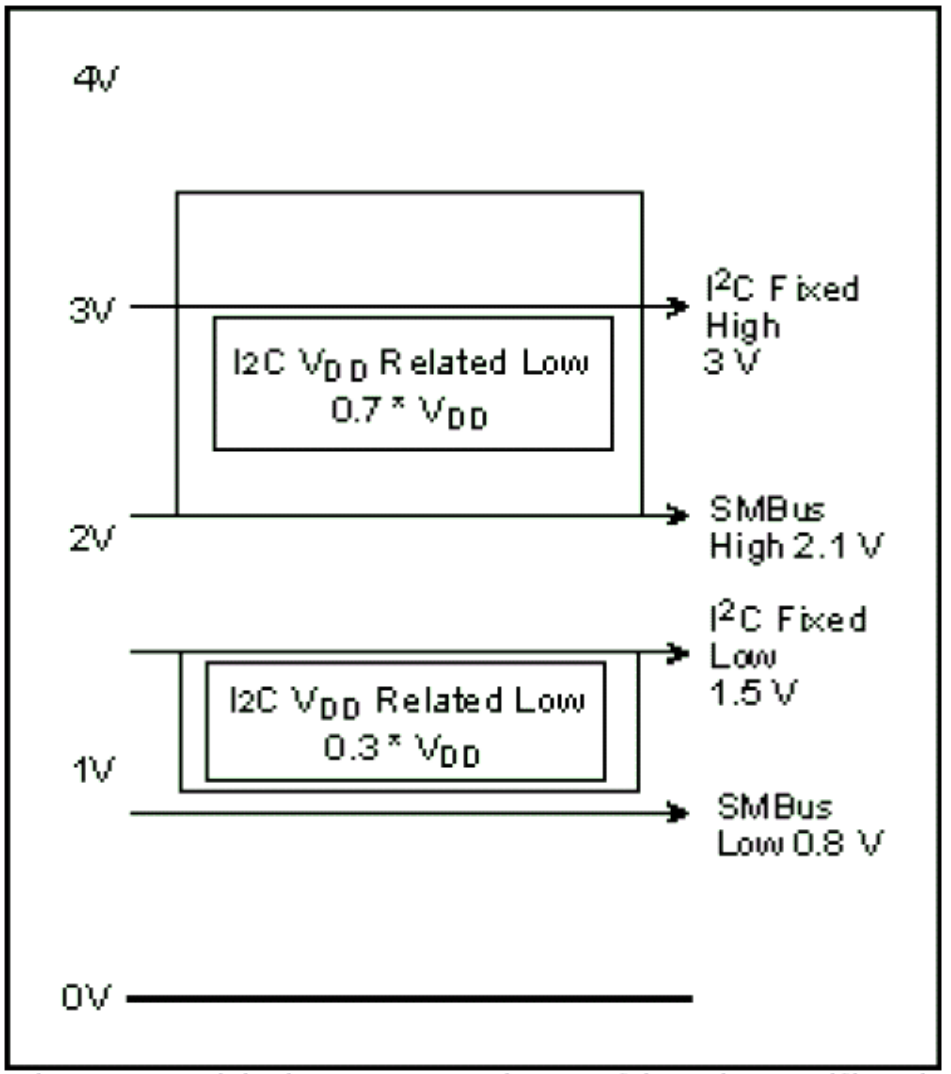
- Mostly compatible with I2C...
- **I2C**
  - Phillips (1992)
  - max. speed:
    100 khz, 400 khz, 3.4 Mhz
  - min. speed: DC
    (no timeout!)
- **SMBus**
  - Intel (1995)
  - max. speed: 100 khz
  - min. speed: 10 khz
    (timeout 35 ms)

	I2C Slave	SMBus Slave
I2C Master		min. 10 khz
SMBus Master	may block	

SMBus - Overview

High	I2C Vdd Dependent	0.7 x Vdd
	I2C Fixed	3.0 V
	SMBus	2.1 V
Low	I2C Vdd Dependent	0.3 x Vdd
	I2C Fixed	1.5 V
	SMBus	0.8 V

- Despite voltage differences, generally the voltages are usually compatible...



[Maxim: I2C vs SMBus]

SMBus - Overview

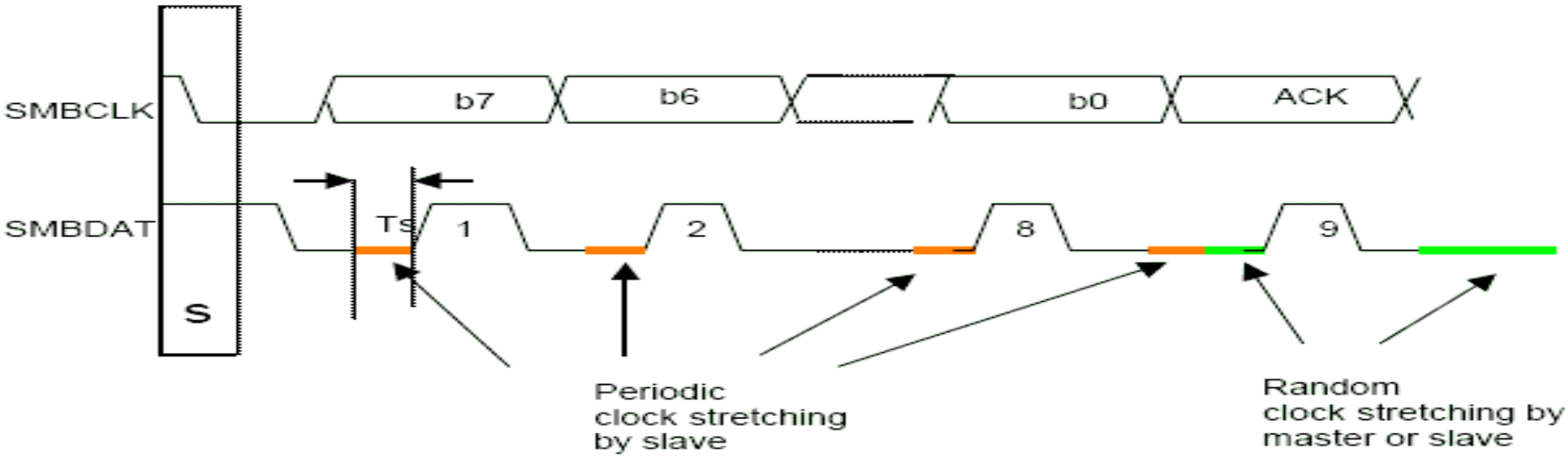
- Differences in maximum bus currents...
  - I2C – Standard: 3 mA < I_{sink}
  - I2C - Fast: 6 mA < I_{sink}
  - SMBus: 100uA < I_{sink} < 350uA

	3 V V_{dd}	5 V V_{dd}
I2C	> 1 kΩ	> 1.6 kΩ
SMBus	> 8.5 kΩ	> 14 kΩ

- pull-up Resistance of 2.4kΩ to 3.9kΩ is common on SMBus

SMBus - Overview

- Protocol:
  - SMBus: slaves must acknowledge reading their address (allows identification of missing devices).
  - SMBus: allows slaves to stretch each bit, as long as min. speed of 10 khz is met...



SMBus - Overview

- Protocol:
  - SMBus may include error checking byte...
  - PEC : Packet Error Checking
  - CRC - 8 bits: $C(x) = x^8 + x^2 + x^1 + 1$

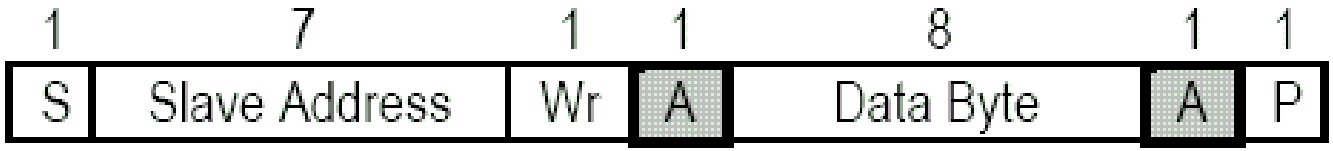


Figure 5-3: Send byte protocol

Figure 5-4: Send byte protocol with PEC

SMBus - Overview

Slave Address Bits 7-1	R/W# bit Bit 0	Comment
0000 000	0	General Call Address
0000 000	1	START byte
0000 001	X	CBUS address
0000 010	X	Address reserved for different bus format
0000 011	X	Reserved for future use
0000 1XX	X	Reserved for future use
0101 000	X	Reserved for ACCESS.bus host
0110 111	X	Reserved for ACCESS.bus default address
1111 0XX	X	10-bit slave addressing
1111 1XX	X	Reserved for future use
0001 000	X	SMBus Host
0001 100	X	SMBus Alert Response Address
1100 001	X	SMBus Device Default Address

SMBus - Overview

Slave Address Bits 7-1	R/W# bit Bit 0	Comment
0000 000	0	General Call Address
0000 000	1	START byte
0000 001	X	CBUS address
0000 010	X	Address reserved for different bus format
0000 011	X	Reserved for future use
0000 1XX	X	Reserved for future use
0101 000	X	Reserved for ACCESS.bus host
0110 111	X	Reserved for ACCESS.bus default address
1111 0XX	X	10-bit slave addressing
1111 1XX	X	Reserved for future use
0001 000	X	SMBus Host
0001 100	X	SMBus Alert Response Address
1100 001	X	SMBus Device Default Address

Table 2 Definition of bits in the first byte

SLAVE ADDRESS	$\overline{R/W}$ BIT	DESCRIPTION
0000 000	0	General call address
0000 000	1	START byte(1)
0000 001	X	CBUS address(2)
0000 010	X	Reserved for different bus format(3)
0000 011	X	Reserved for future purposes
0000 1XX	X	Hs-mode master code
1111 1XX	X	Reserved for future purposes
1111 0XX	X	10-bit slave addressing

SMBus vs I2C - Overview

	I2C	SMBus
Timeout	No	Yes (35 ms)
Min. Clock	DC	10 khz
Max. Clock	100 khz; 400 khz; 3.4 Mhz	100 khz
V_{high}	0.7 x V_{dd} ; 3.0 V Fixed	2.1 V
V_{low}	0.3 x V_{dd} ; 1.5 V Fixed	0.8 V
Max I_{sink}	3 mA (Standard); 6 mA (Fast)	350 uA
Clock Name	SCL	SMBCLK
Data Name	SDA	SMBDAT
General Call	Yes	Yes
Packet CRC	No	Optional
Dynamic Addresses	No (v 2.1)	Yes
Interrupt	No	Optional (ALERT#)
Bus Suspend	No	Optional (SMBSUS#)
Slave Addr. Ack	Yes/No	Always Yes

Interface / Communication Buses

- I2C – Inter-Integrated Circuit
- SMBus
- CBus
- SPI – Serial Peripheral Interface
- USB – Universal Serial Bus
- ZigBee
- RS232 / RS422 / RS 485
- CAN
- JTAG

CBus

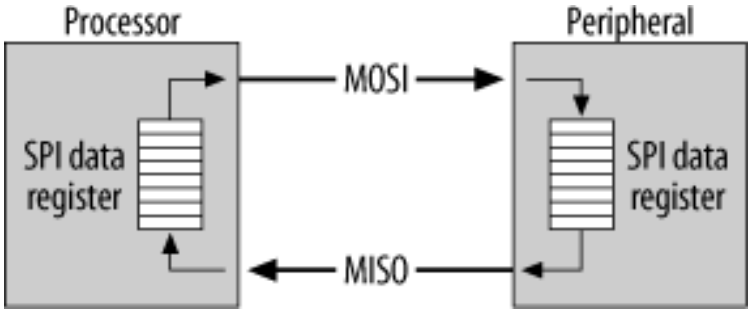
- Obsolete...
  - Cbus devices no longer manufactured
- 3 line bus (similar to I2C)
  - physical layer compatible to I2C
  - network layer (frames) not compatible with I2C
    (special address on I2C allows co-existence of CBUS and I2C devices on the same bus.)

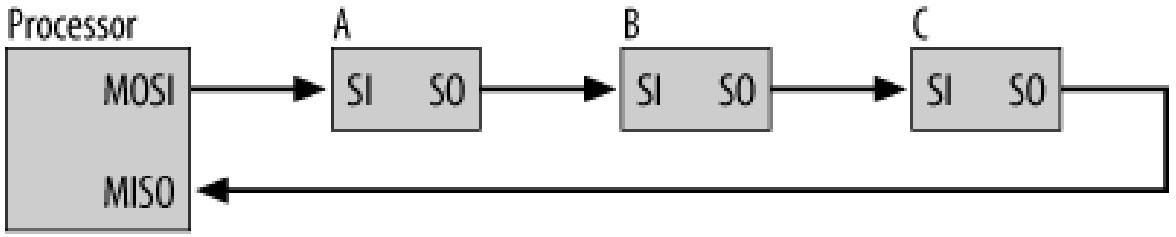
Interface / Communication Buses

- I2C – Inter-Integrated Circuit
- SMBus
- CBus
- SPI – Serial Peripheral Interface
- USB – Universal Serial Bus
- ZigBee
- RS232 / RS422 / RS 485
- CAN
- JTAG

SPI - Overview

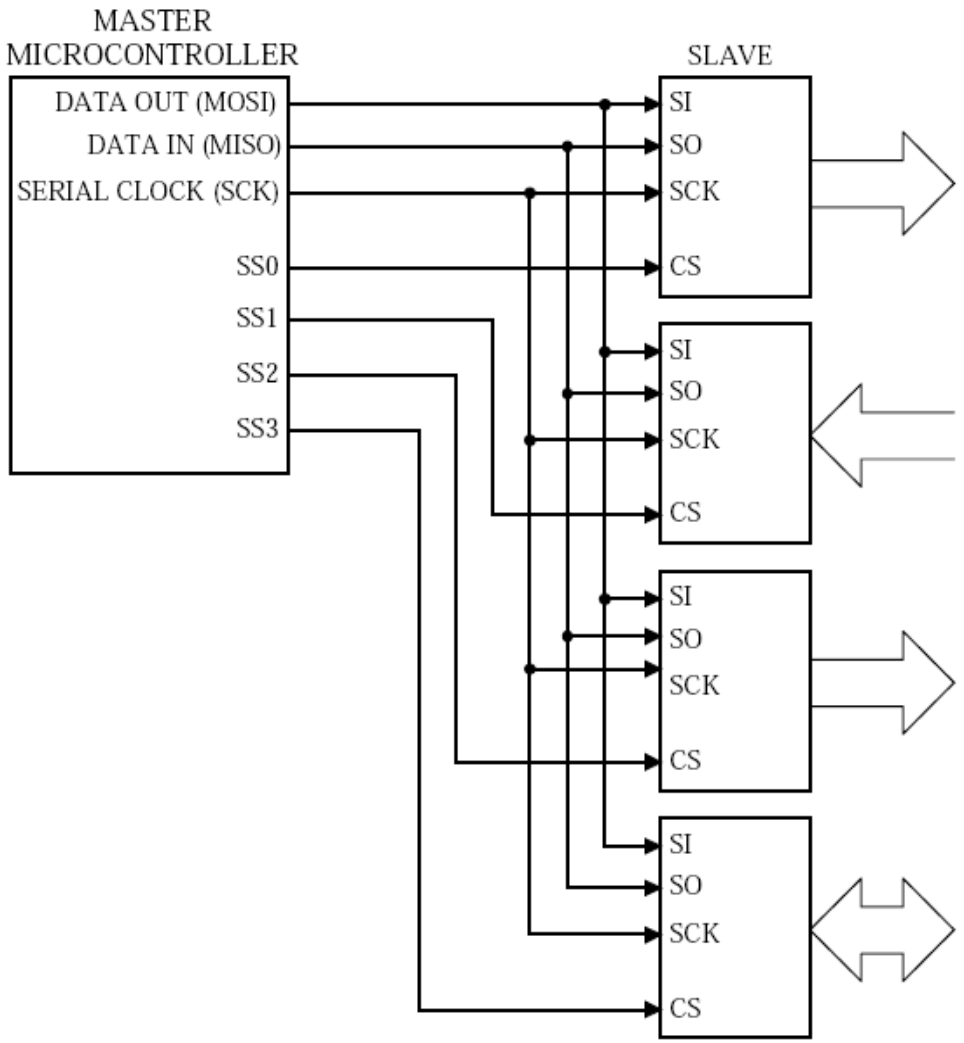
- SPI – Serial Peripheral Interface
  - Serial interface, Synchronous, Full-Duplex





SPI - Overview

- SPI –
  Serial Peripheral Interface
  - Serial Interface
  - Synchronous
  - Master/Slave
  - Full-Duplex
  - 3 lines + GND:
    - Clock,
    - Data Master → Slave,
    - Data Slave → Master
    - +1 Chip Select per slave

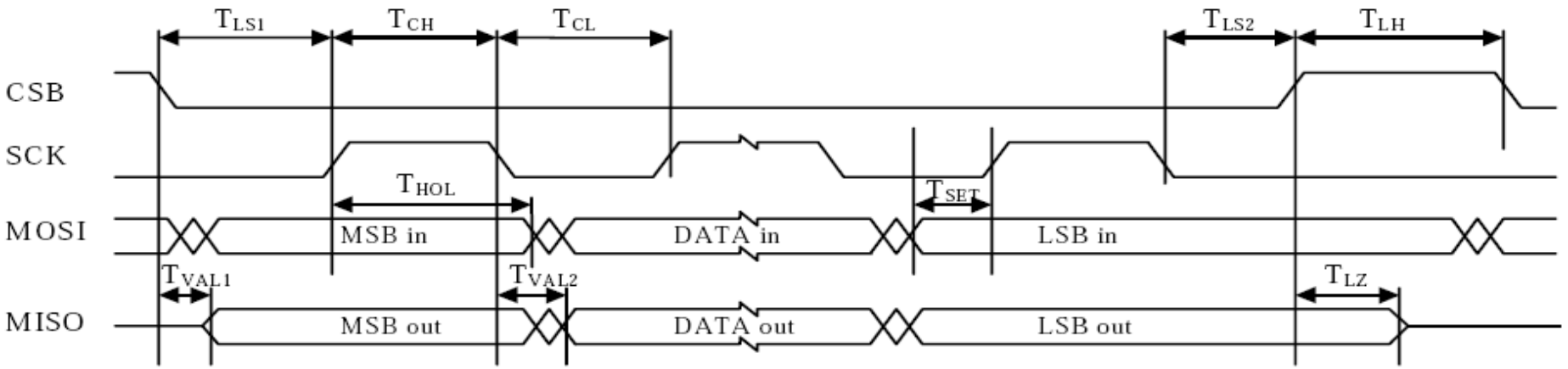


[VTI Technologies: Technical Note 15]

SPI - Overview

- started with Motorola's M68HC11 micro-controller
  (now Freescale)
- SPI is not standardized
  - Timings are not defined
  - Voltages and impedances are not defined!

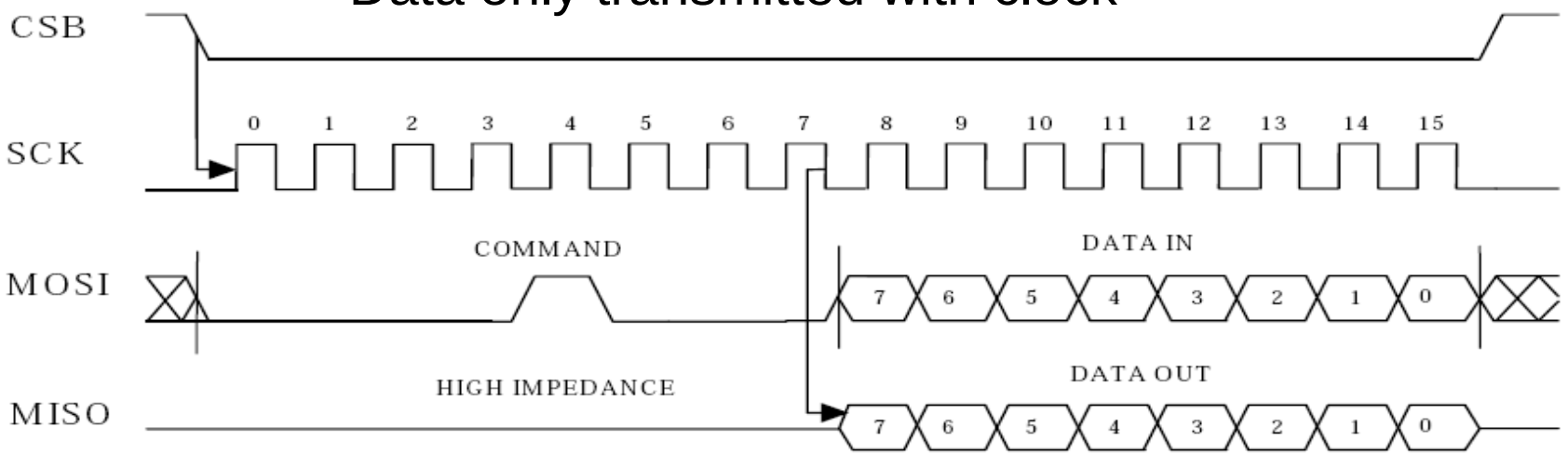
It is up to the engineer to guarantee compatibility between devices on the same bus!



SPI - Overview

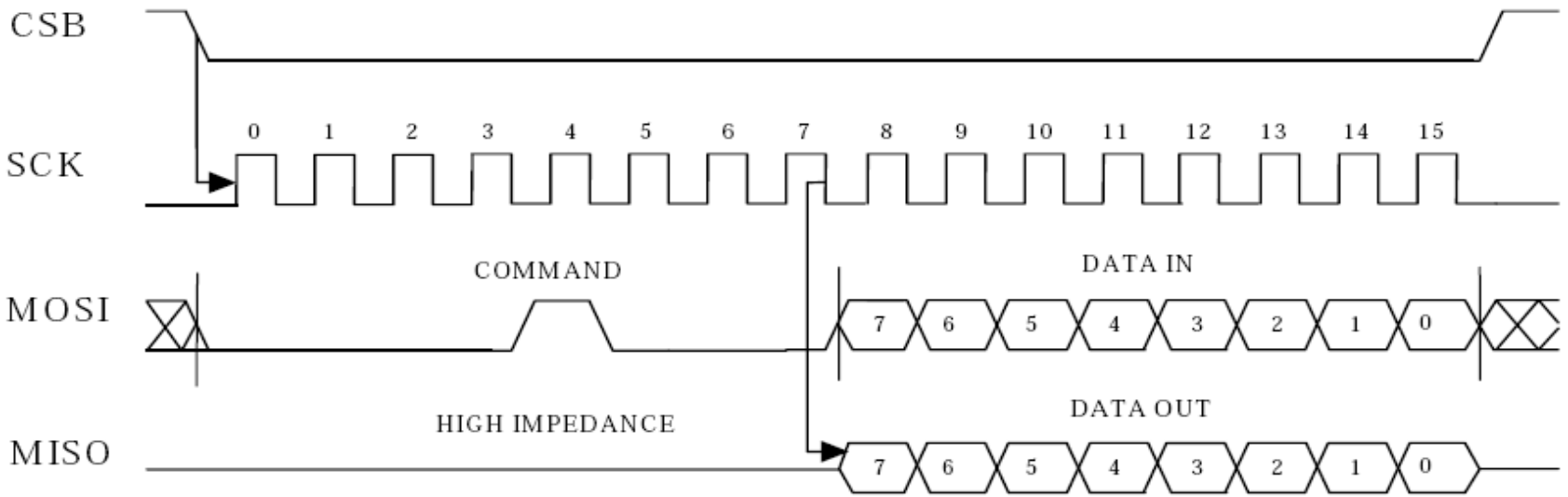
- SS: Slave Select
- MOSI: Master Output, Slave Input
- SCK: Serial Clock
- MISO: Master Input, Slave Output

  - Clock line is controlled by master only
  - Clock frequency may change
  - Data only transmitted with clock



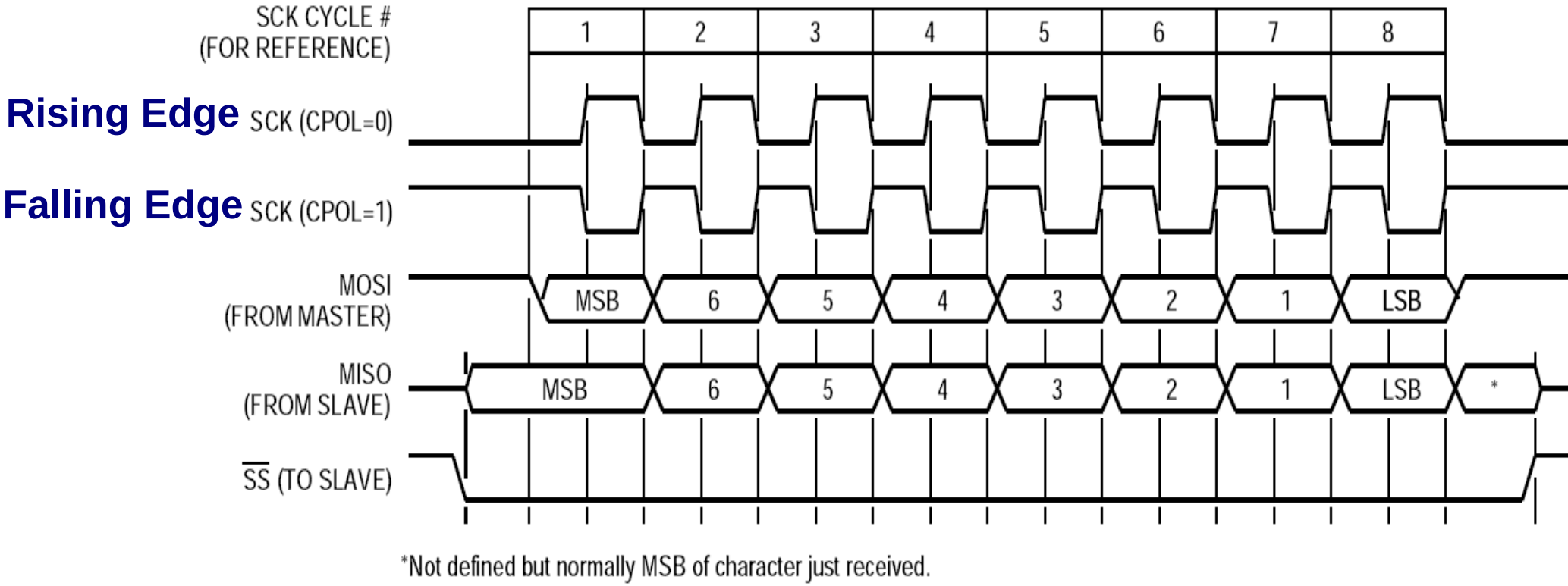
SPI - Overview

- Data usually transferred as bytes, but not always!...
  (ex. **`SCA61T, SCA100T, SCA103T`**, ...)
- 4 main modes, depending on clock phase and edge



SPI

Phase = 0

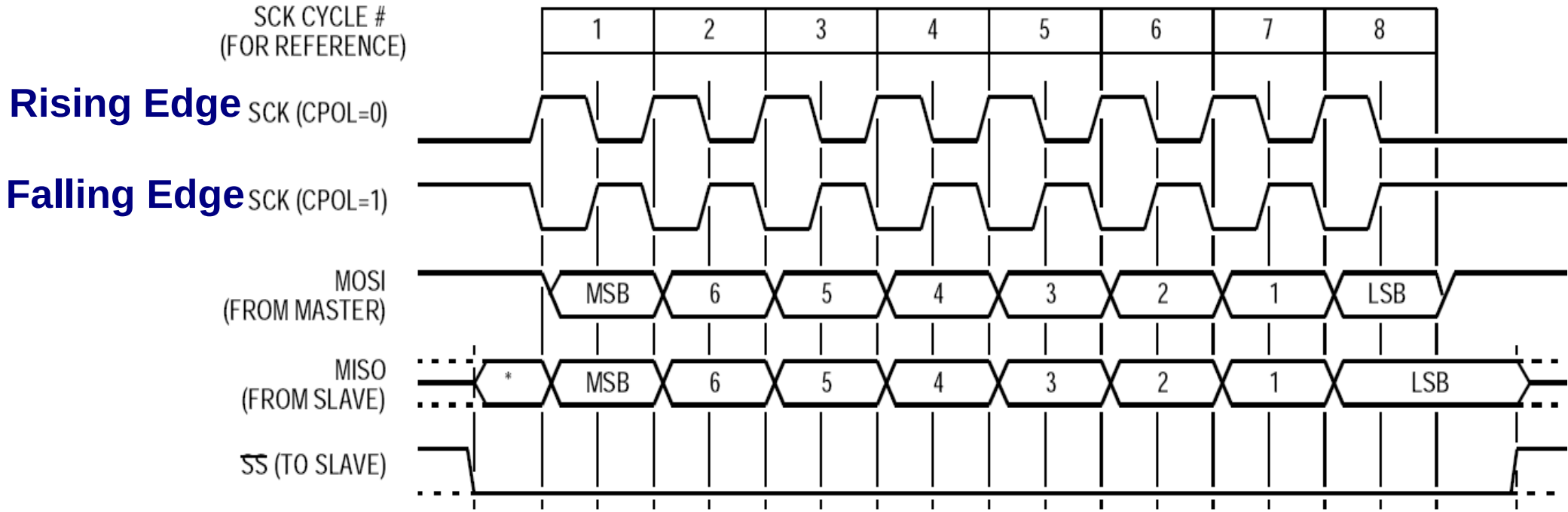


*Not defined but normally MSB of character just received.

- SS (Slave Select)
  must go to 0, then to 1, between transmission of 2 consecutive bytes!

SPI

Phase = 1



*Not defined but normally LSB of previously transmitted character.

- SS (Slave Select)
  may remain active (0) between transmission of 2 consecutive bytes!

Interface / Communication Buses

- I2C – Inter-Integrated Circuit
- SMBus
- CBus
- SPI – Serial Peripheral Interface
- USB – Universal Serial Bus
- ZigBee
- RS232 / RS422 / RS 485
- CAN
- JTAG

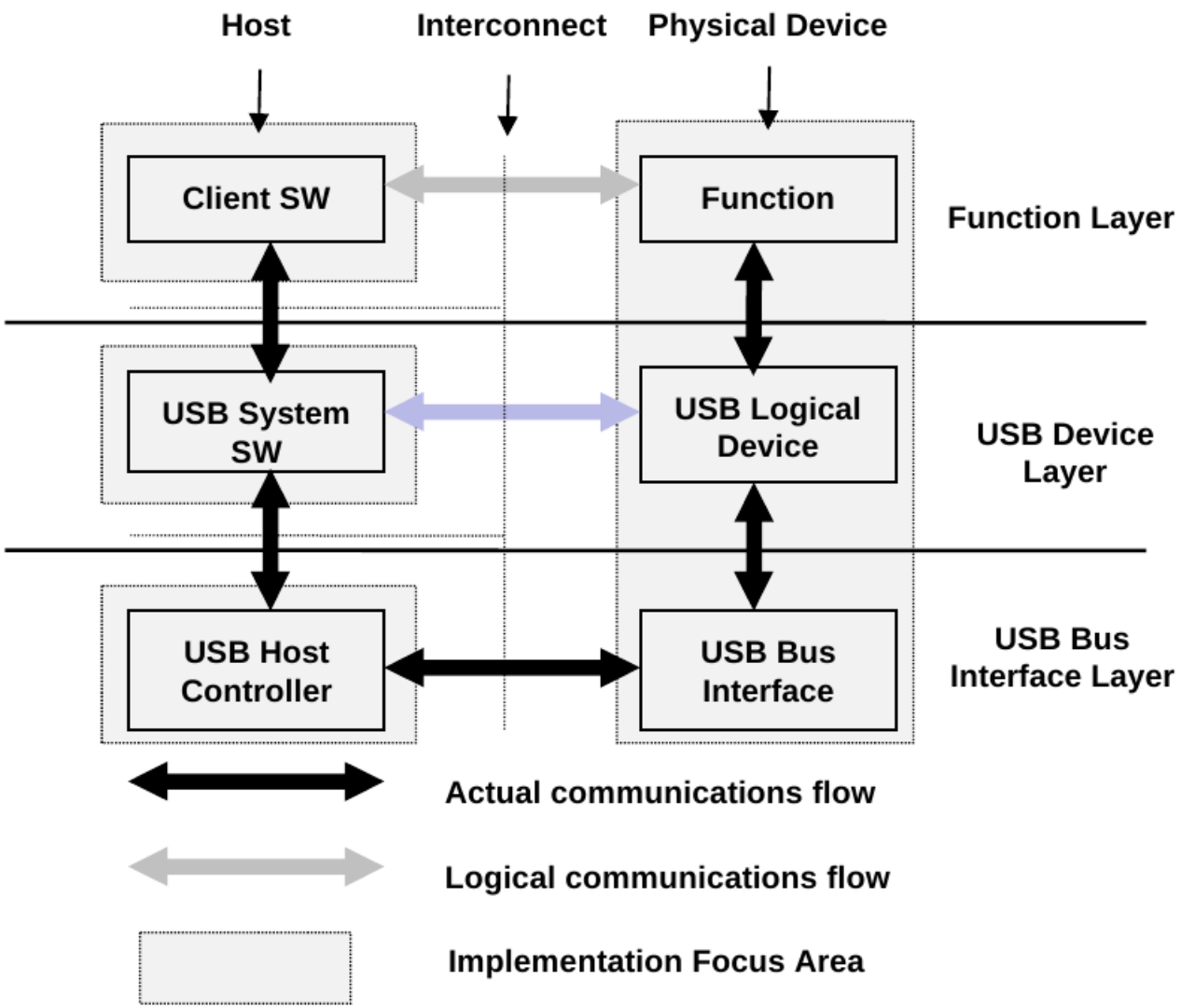
USB - Overview

- USB – Universal Serial Bus

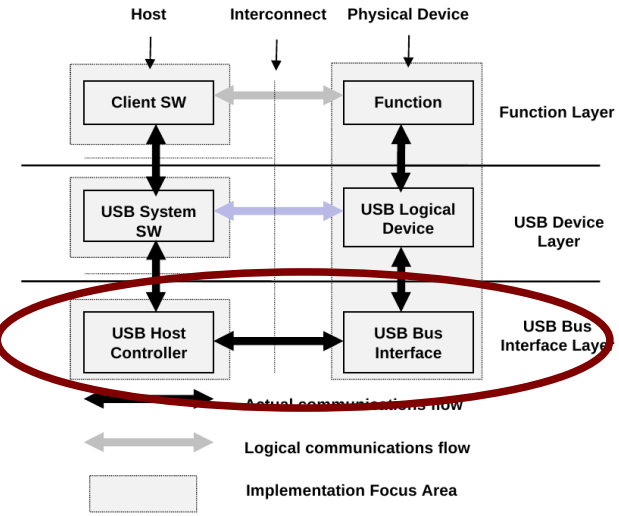
- V 1.0
  - Low-Speed: 1.5 Mbit/s
  - Full-Speed: 12 Mbit/s
- V 2.0
  - Low-Speed: 1.5 Mbit/s
  - Full-Speed: 12 Mbit/s
  - High-Speed: 480 Mbit/s

- 1 Master (Host),
  n Slaves (Device)
- Master initiates all data transfers
- Master periodically polls each slave

USB - Overview



USB – Physical Topology



Interface between Host and RootHub

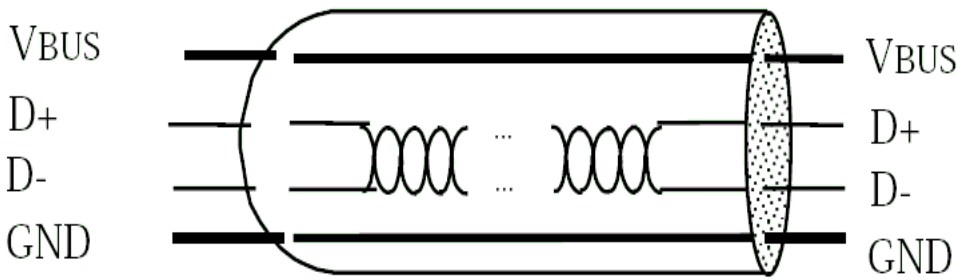
ver1.0: UHCI, OHCI
ver 2.0: EHCI
(Universal / Open / Extended HCI → Host Controller Interface)

USB Device
(up to 5 levels)

USB Device

USB – Physical Layer

- 2 Power Lines
  - V_{BUS} – 5 V
  - 100 mA por device
  - 500 mA
    (after host's authorisation)

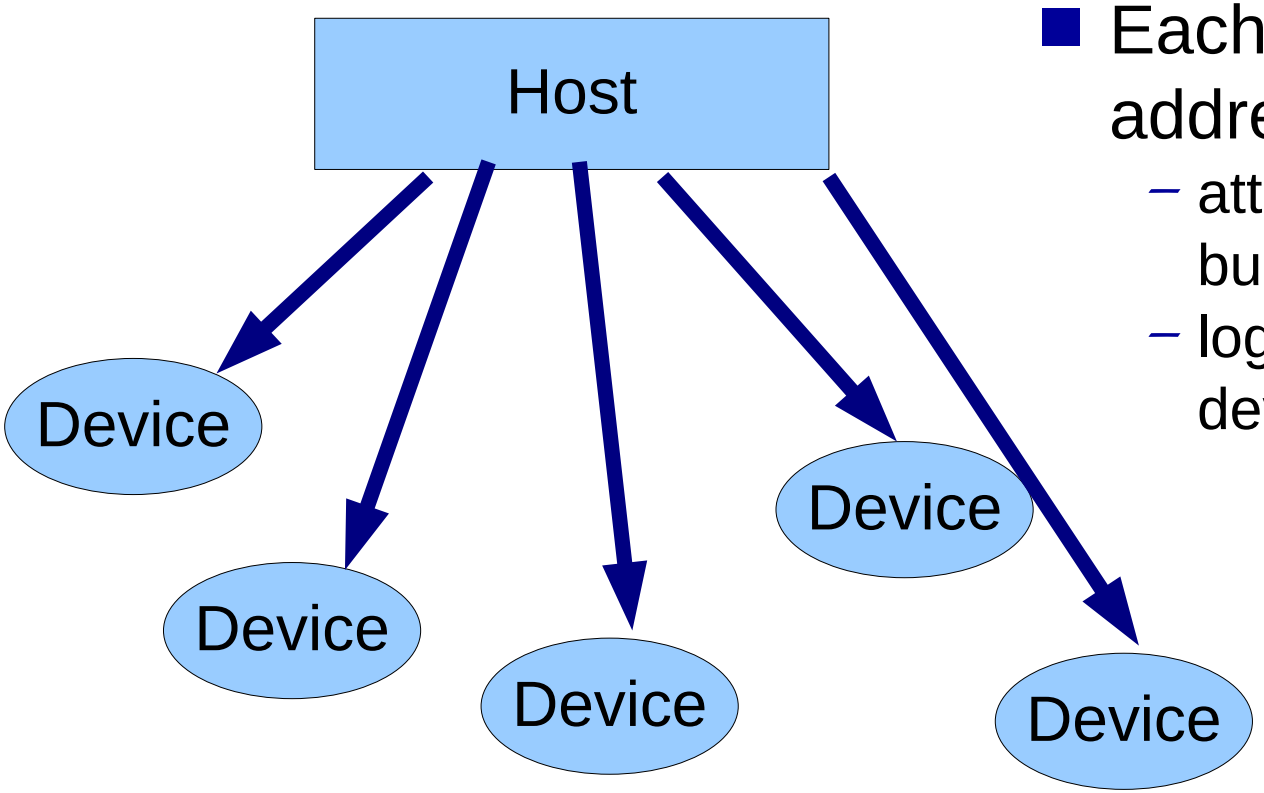


- 2 Data Lines
  - Half-Duplex
  - Differential
- NRZI transmission
  - 0 - edge; 1 – no edge
  - Clock obtained from data
    (if required for clock → Bit stuffing after 6 bits)
- Max. cable length - 5m
  - Guarantees delays < 26ns
    (and therefore slave's device eaches host on time)

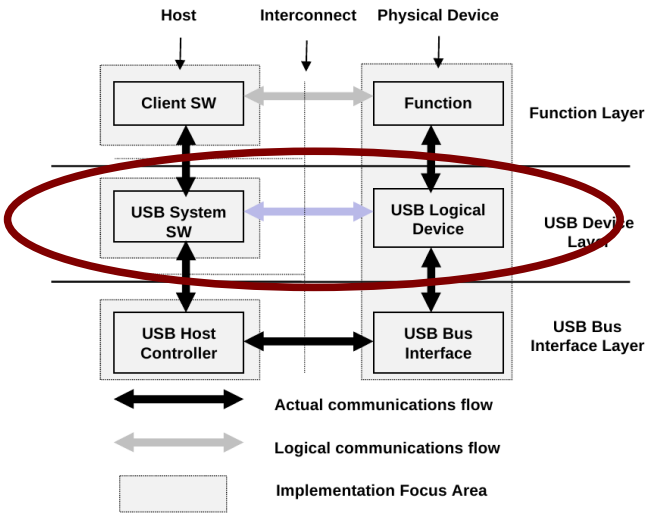
USB – Physical Layer

- No device
  - Data lines (D+, D-) float (i.e. pulled to GND by 15 kΩ)
- Connected device
  - pulls D+ or D- to V+ with 1.5 kΩ
    - Low-Speed: D-
    - Full-Speed: D+

USB – Logical Topology

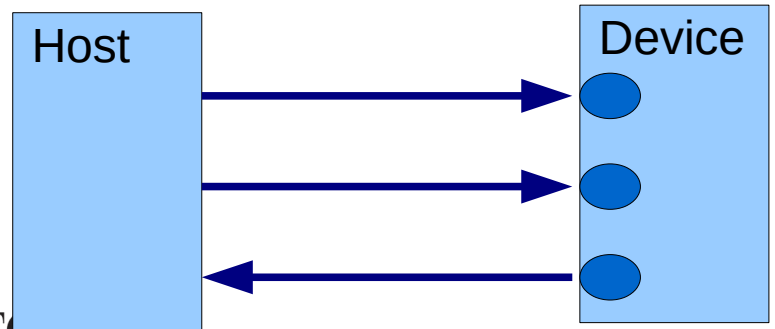


- Each devices has its own address
  - attributed when connected to bus
  - logical access is direct to device



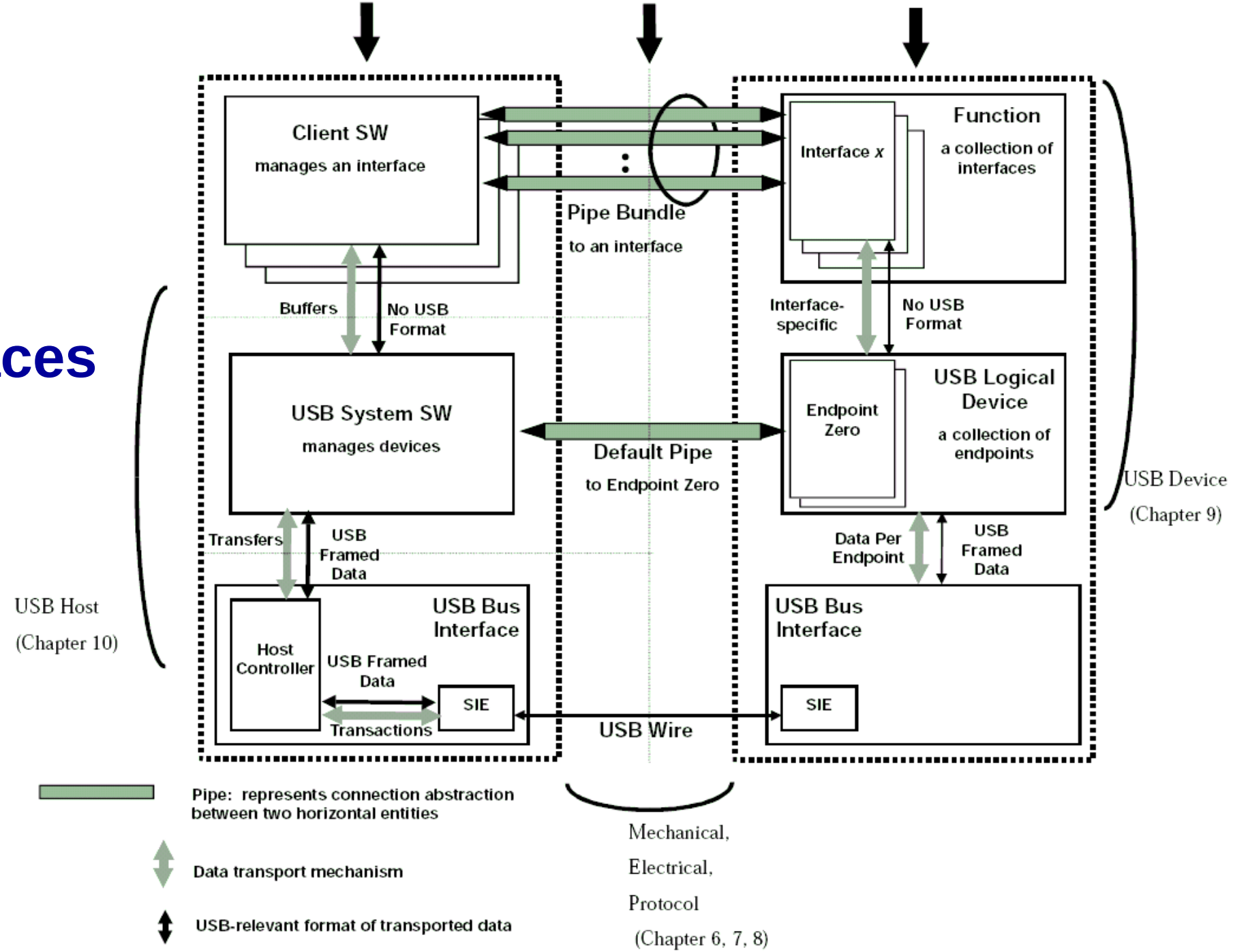
USB

- Communication between host and device over virtual pipes
  - each pipe has an end-point on the device
  - pipe with end-point 0 always exists



- Each device implements a function:
  - HMI: mouse, keyboard, joystick, ...
  - Image: scanner, printer, camera, ...
  - Mass Storage: CD-ROM, hard disk, flash disk, ...
- A physical device may simultaneously implement more than one function.

USB - Interfaces



USB – Device Classes

Class	Usage	Description	Examples
00h	Device	Unspecifiedclass 0	(Device class is unspecified. Interface descriptors are used for determining the required drivers.)
01h	Interface	Audio	Speaker, microphone, sound card
02h	Both	Communications and CDC Control	Ethernet adapter, modem, serial port adapter
03h	Interface	Human Interface Device (HID)	Keyboard, mouse, joystick
05h	Interface	Physical Interface Device (PID)	Force feedback joystick
06h	Interface	Image	Digital camera (Most cameras function as Mass Storage for direct access to storage media).
07h	Interface	Printer	Laser printer, Inkjet printer
08h	Interface	Mass Storage	USB flash drive, memory card reader, digital audio player, external drives
09h	Device	USB hub	Full speed hub, hi-speed hub
0Ah	Interface	CDC-Data	(This class is used together with class 02h - Communications and CDC Control.)
0Bh	Interface	Smart Card	USB smart card reader
0Dh	Interface	Content Security	-
0Eh	Interface	Video	Webcam
0Fh	Interface	Personal Healthcare	-
DCh	Both	Diagnostic Device	USB compliance testing device
E0h	Interface	Wireless Controller	Wi-Fi adapter, Bluetooth adapter
EFh	Both	Miscellaneous	ActiveSync device
FEh	Interface	Application Specific	IrDA Bridge
FFh	Both	Vendor Specific	(This class code indicates that the device needs vendor specific drivers.)

USB – Logical Topology - Function

- Each function is controlled independently

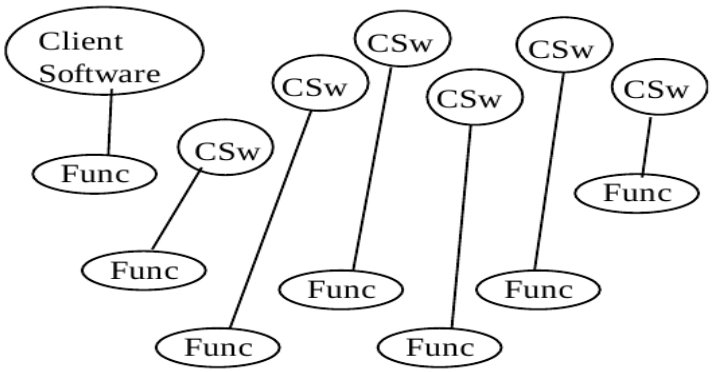
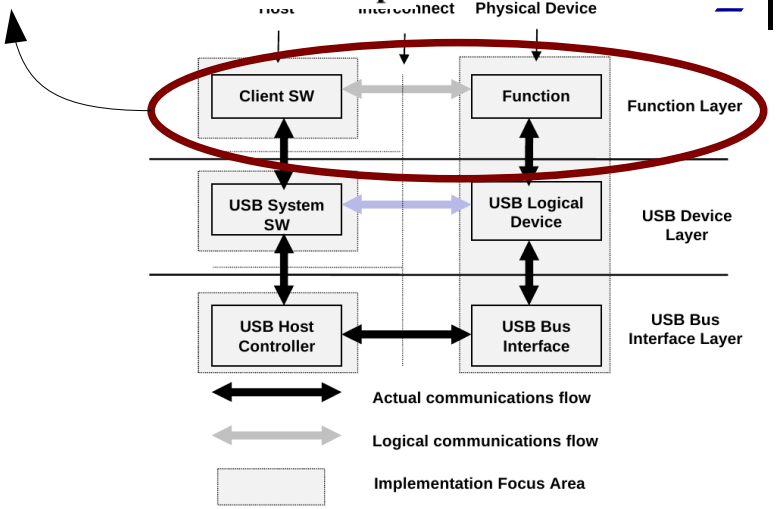
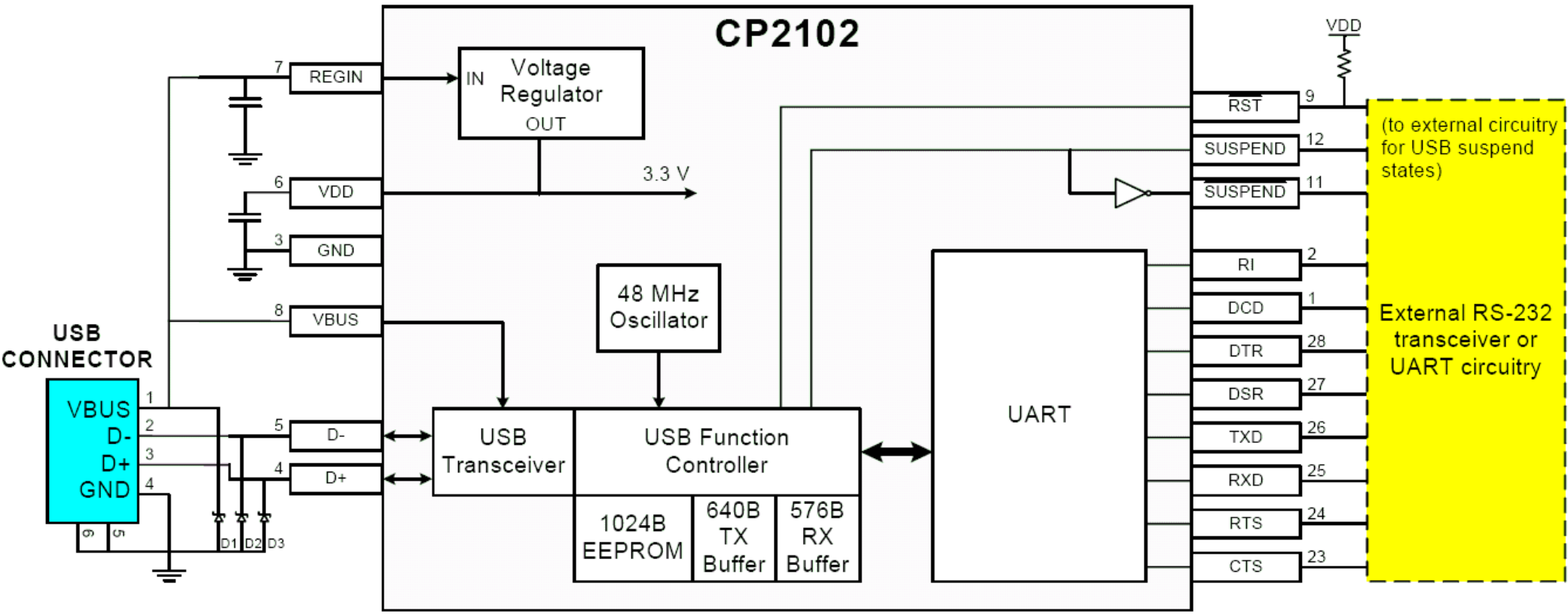


Figure 5-8. Client Software-to-function Relationships



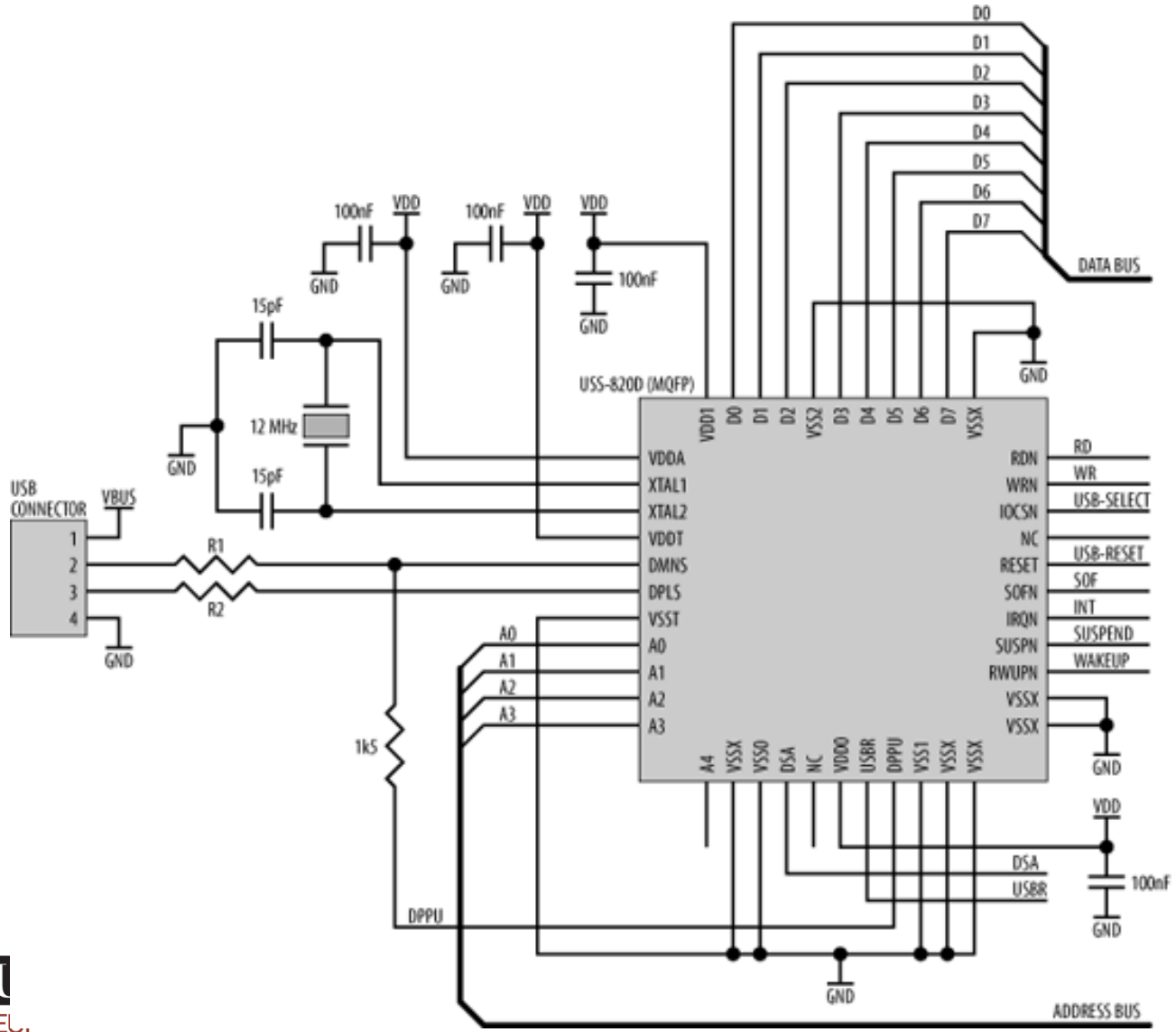
- Data transfer types:
  - Control Transfers
    - (configuration & control)
  - Bulk Data Transfers
    - (large sequential data - disks, scanners, ...)
  - Interrupt Data Transfers
    - (limited negotiated latency transfers – event notification)
  - Isochronous Data Transfers
    - a.k.a. streaming real-time
    - (large) fixed negotiated bandwidth
    - negotiated minimum latency

USB to UART bridge



Implements class 02h USB device
(Communications and CDC Control – ex. modem, serial port adaptor)

USB Device IC



USS-820D (Agere Systems)

- Implements a generic USB device.
- USB function must be implemented by CPU.
- Supports up to 6 endpoints.

Interface / Communication Buses

- I2C – Inter-Integrated Circuit
- SMBus
- CBus
- SPI – Serial Peripheral Interface
- USB – Universal Serial Bus
- JTAG
- ZigBee
- RS232 / RS422 / RS 485
- CAN
- JTAG

ZigBee - Overview

- Name of a specification which includes:
  - several high level protocols
  - radio transmission
    - low power transmission
    - low power consumption => good for battery based systems
    - low data rate
    - reduced dimensions
    - digital transmission
    - secure

ZigBee

- Based on IEEE 802.15.4-2006 for WPANs

  (wireless personal area networks)

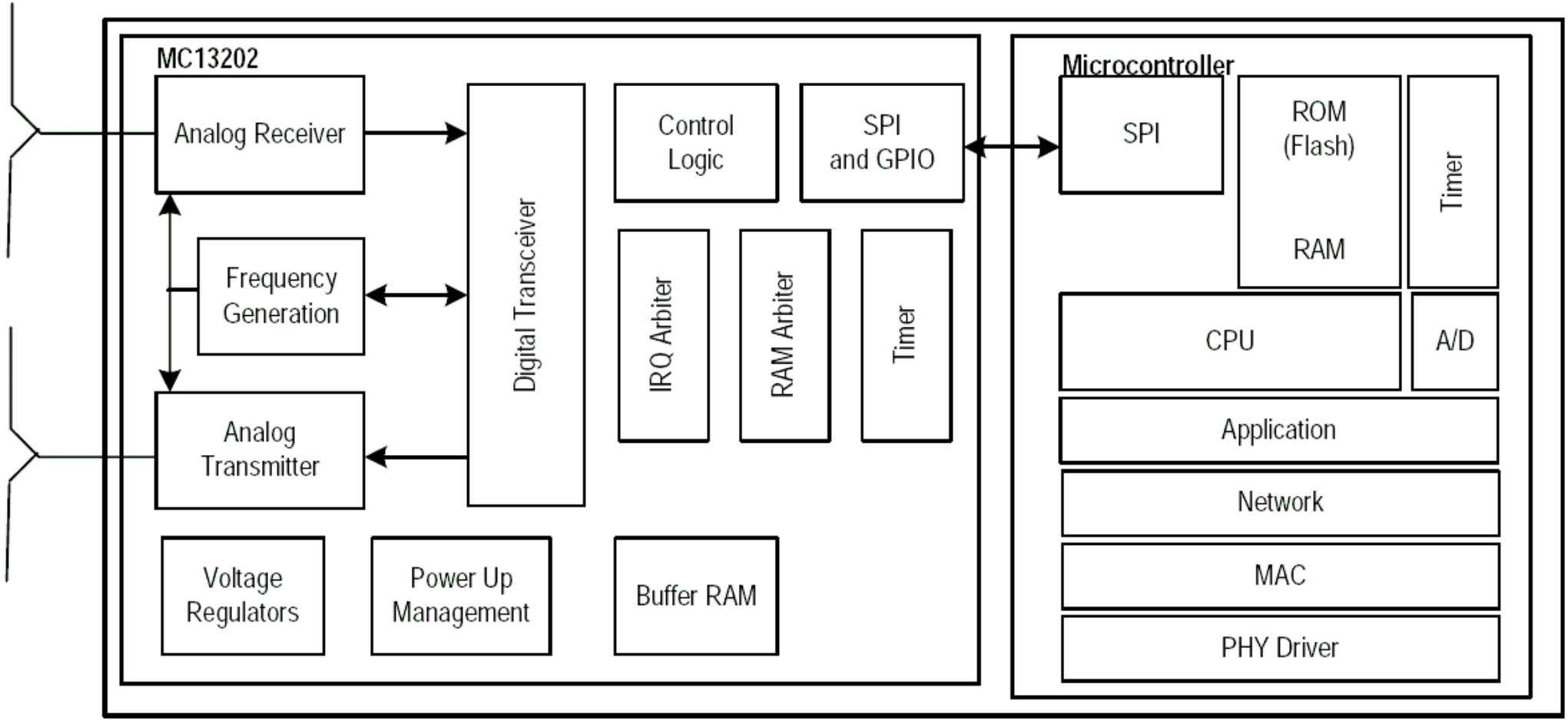
  - IEEE 802.15.4 defines physical & medium access control layers
    (used by ZigBee, WirelessHART, & MiWi)
  - Uses frequencies
    - 868-868.8 MHz: Europe, suports 1 communication channel
    - 902-928 MHz: USA, up to 30 coms. channels
    - 2400-2483.5 MHz: worldwide, up to 16 channels
  - up to 250 kbits/s

ZigBee

- ZigBee Device Types:
  - ZigBee coordinator(ZC):
    - most powerful device.
    - device that creates the network root → 1 coordinator in each network.
  - ZigBee Router (ZR):
    - may execute an application function
    - may route data from one device to another
  - ZigBee End Device (ZED):
    - only communicates with its immediate supervisor (coordinator or router).
    - may sleep most of the time => little energy consumtion => batteries
- ZigBee Profiles
  - Home Automation
  - Telecommunication Applications
  - Hospital Care
  - ZigBee Smart Energy
  - Personal Home

ZigBee

MC13202 – IEEE 802.15.4 Transceiver
Micro-controller: ZigBee Stack (commercially available)



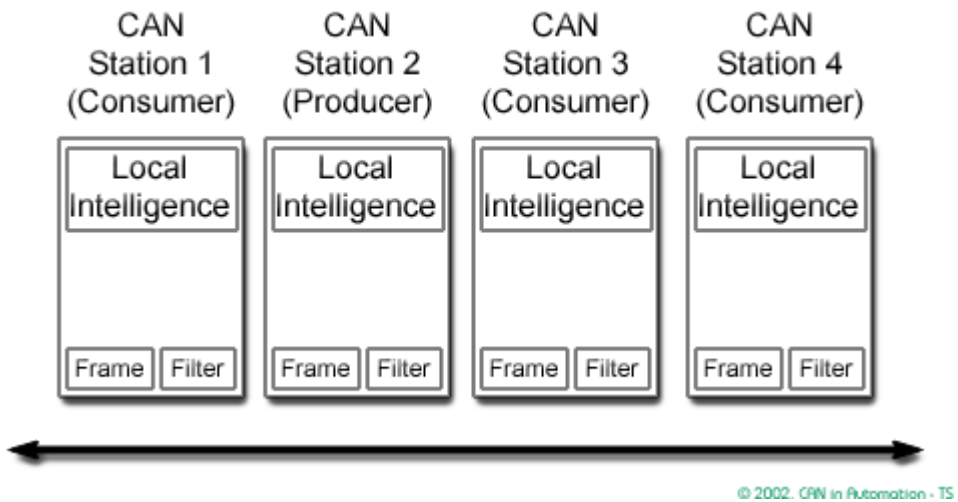
NOTE: ICs with both controller & transceiver are also available

Interface / Communication Buses

- I2C – Inter-Integrated Circuit
- SMBus
- CBus
- SPI – Serial Peripheral Interface
- USB – Universal Serial Bus
- JTAG
- ZigBee
- RS232 / RS422 / RS 485
- CAN
- JTAG

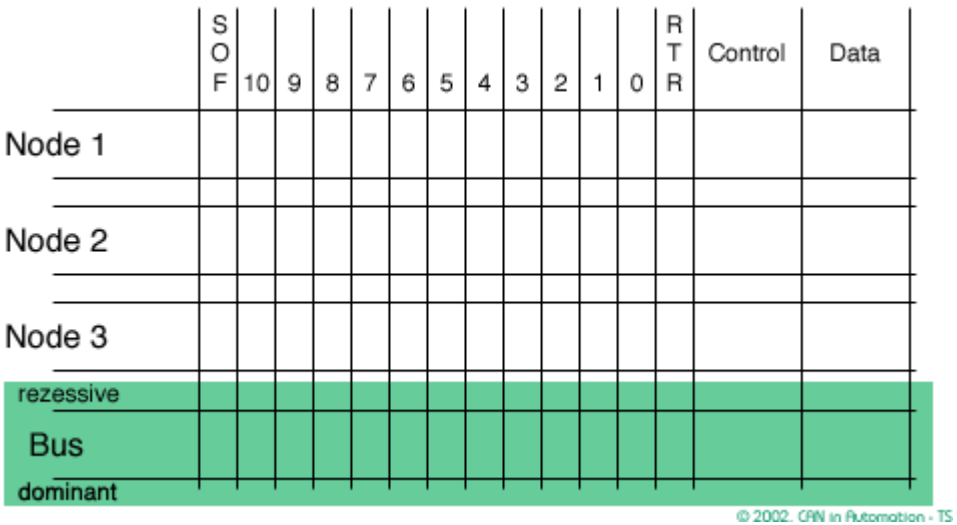
CAN-bus

- Industrial Communication Bus (OSI layers 1 & 2).
  - Used in cars & several fieldbuses (e.g. CANopen & DeviceNET)
- Message oriented
  - Based on message broadcast
  - Stations do not have network address.
  - Each message has a unique identifier.



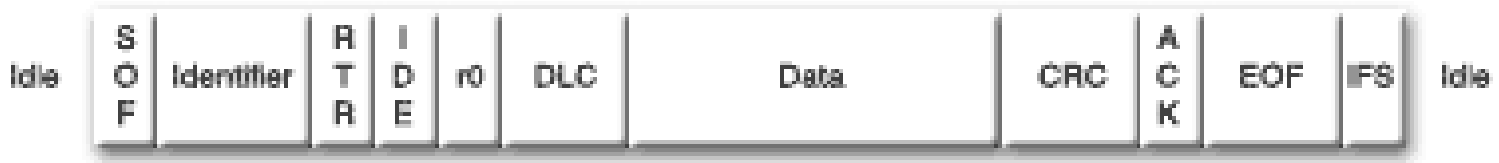
CAN-bus

- Medium Access Control:
  - Bus consists of a 'wired AND'.
  - a transmitting station that detects a bit distinct to the one that it a wanted to transmit, will stop to and listen to the bus.
  - Messages have priority that coincides with the message identity.



CAN-bus

- Frame Format
  - **SOF**: start of frame
  - **Identifier**: 11 bits (version A), 29 bits (version B)
  - **RTR**: Remote Transmission Request (frame requesting data)
  - **IDE**: Identifier Extension (distinguishes version A from B)
  - **DLC**: Data Length Code (data bytes – max of 8 per frame)
  - **CRC**: Cyclic Redundancy Check (for error detection)
  - **ACK**: Acknowledge (emitter sends 1; receiver confirm with 0)
  - **EOF**: End of Frame
  - **IFS**: Intermission Frame Space (silence between frames)



CAN-bus

- Physical Layer
  - Relation speed/bus length is limited by speed of signal propagation in medium (data must stabilize due to wired AND operation)

Bit Rate	Nominal Bit Time	Remarks
1 Mbit/s	1 µs	max. 40 m
800 kBit/s	1.25 µs	
500 kbit/s	2 µs	
250 kbit/s	4 µs	
125 kbit/s	8 µs	
(100 kbit/s)	(10 µs)	(1)
50 kbit/s	20 µs	max. 1 km
20 kbit/s	50 µs	has to be supported by all modules
10 kbit/s	100 µs	min. bit rate

CAN-bus

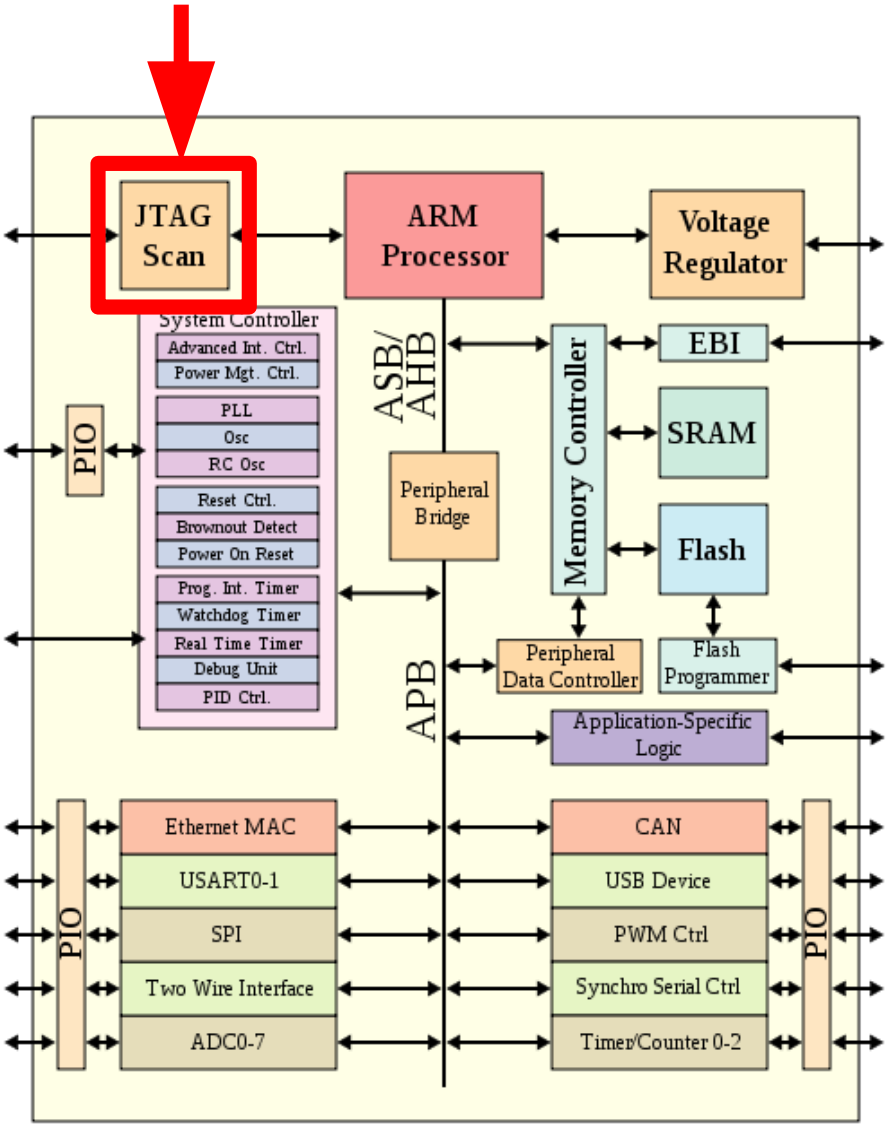
- Physical Layer
  - Uses DB9 connectors (similar to serial ports)

Pin	Signal	Description
1	-	Reserved
2	CAN_L	CAN_L bus line (dominant low)
3	CAN_GND	CAN Ground
4	-	Reserved
5	(CAN_SHLD)	Optional CAN Shield
6	(GND)	Optional CAN Ground
7	CAN_H	CAN_H bus line (dominant high)
8	-	Reserved (error line)
9	(CAN_V+)	Optional CAN external positive supply (dedicated for supply of transceiver and optocouplers, if galvanic isolation of the bus nodes applies)

Interface / Communication Buses

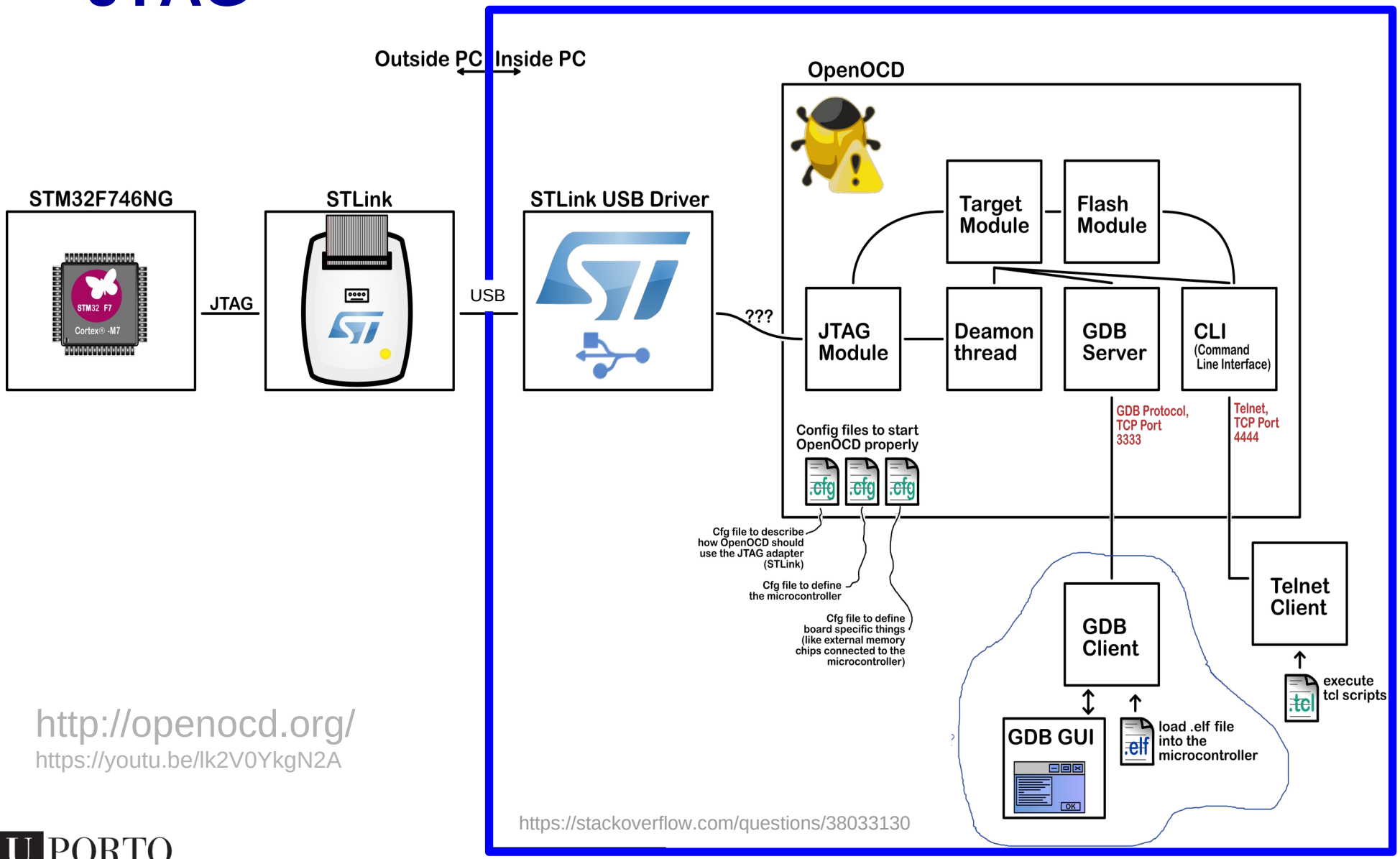
- I2C – Inter-Integrated Circuit
- SMBus
- CBus
- SPI – Serial Peripheral Interface
- USB – Universal Serial Bus
- JTAG
- ZigBee
- RS232 / RS422 / RS 485
- CAN
- JTAG

JTAG



- Communication interface for board-level and chip-level debugging
- Chips on a board connected in daisy-chain or star topology
- Supports boundary scan testing
  - Forcing and reading of I/O pins
- Programming of on-board memory
- Supports CPU debugging
  - Halt CPU
  - Single-step
  - read/write CPU registers

JTAG



http://openocd.org/

https://youtu.be/lk2V0YkgN2A

PCI bus
PCIe bus

Main Bibliography

- “Programming Embedded Systems”, Michael Barr
- “Embedded Systems Memory Types”, Michael Barr, in May 2001 issue of Embedded Systems Programming
- “The I2C Bus Specification, Version 2.1”, Phillips Semiconductors, January 2000
- “M68HC11 Reference Manual”,
  Chapter 8 “Synchronous Serial Peripheral Interface”,
  Freescale Semiconductor, 2007
- “Embedded System Design” Peter Marwedel