ITKC11 Mobile application platforms

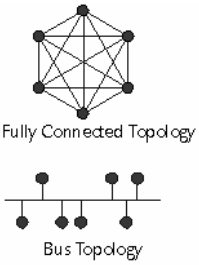
http://tisu.mit.jyu.fi/embedded/ITKC11/ITKC11.htm

Interconnection buses

Ville Pietikäinen and Jarkko Vuori

(Jarkko.Vuori@jyu.fi)

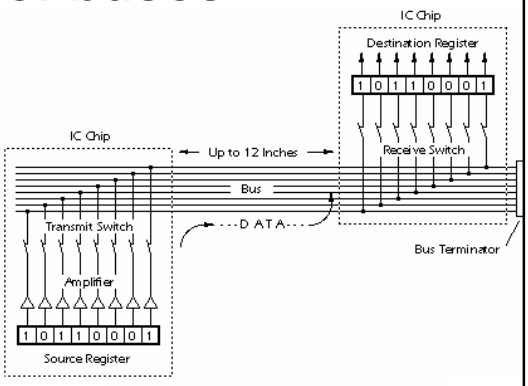
Basics of buses

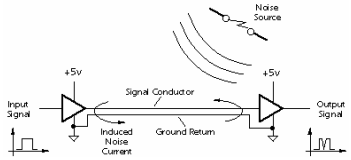


- A bus = a collection of appropriate signals
- Bus signals:
- Control
  - Control signals time the bus transactions is asynchronous buses
- Address
  - Unidirectional output from processor, addresses a memory area, a bus itself
- Data
  - Bidirectional, a bus that transfers the data

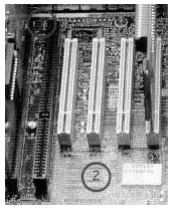
Basics of buses

- Bus contains a group of different signals. In addition to signals, a protocol describing how the bus transactions operate is needed.
- Protocol tells what happens on the bus and when
  - Control signals are part of the protocol
  - In a bus without control signals the protocol is embedded in bus data transactions. The same bus can therefore transfer commands, addresses and data.
- Capacity of a bus is average (sustained) data transfer ability in bytes per second
  - Quite often the peak data transfer rate of the bus is used for capacity indication.

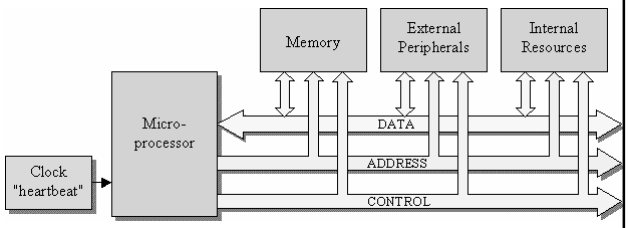




Buses



- Bus examples:
- Every processor, processor core or microcontroller has a bus or multiple buses
- In addition, several different expansion buses exist: PCI (several versions), VME, Futurebus, ISA, etc. Most of the expansion buses are well standardized.
  - First open source standardised bus (by IEEE) was GPIB measurement device interconnection bus (in the middle of 70's)
- Processor and expansion buses can also be proprietary solutions depending on the manufacturer

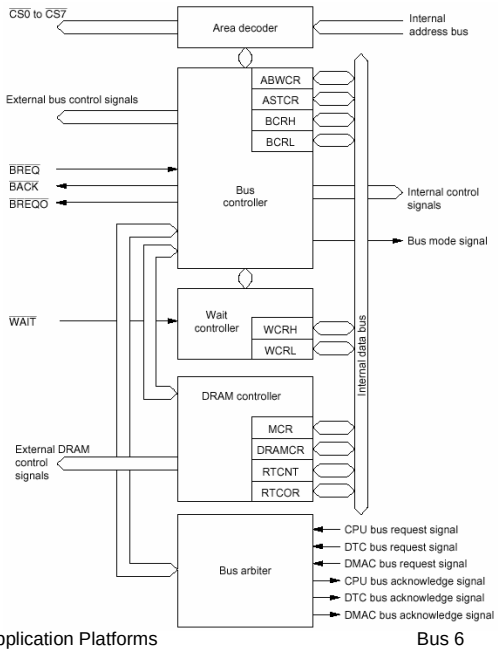


Processor bus

- The immediate interface of a processor to off chip devices, FSB (Front Side Bus)
  - Memories, peripherals
  - Extension bus
- Signals:
  - Typical minimum direct connections are Dx - D0, Ax - A0, RD, WR
  - Can contain signals like AS, CSx, etc
  - DMA- or other bus mastering: DREQ, DACK, BUSRQ, BGRNT
  - Slowing down: WAIT, READY, etc
  - Synchronous: CLK
  - Harvard: PGM_select, DATA_select, IO_select
  - Power supplies: +3,3V, +5V, +12V

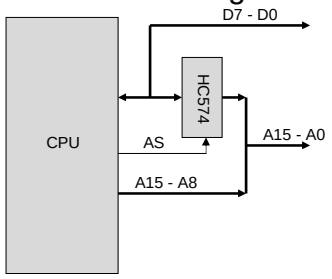
Processor bus

- Bus controller often controls the bus:
  - Bus controller configures the bus after the reset to the needed operating mode
  - Bus controller creates necessary control lines for the address and data busses
  - Bus controller also takes care of bus timing
- In the figure is Hitachi H8S-series processor's bus controller block



Processor bus

- Two types of busses: multiplexed and direct
- Multiplexed bus uses same pins for both the address and data
  - In order to reduce pin counts
- Differentiating the address and data is done with the signal AS (Address Strobe)
- All multiplexed: AD15 - AD0
- Partial multiplexing,
  e.g. AD7 - AD0 and A15 - A8.



Address bus

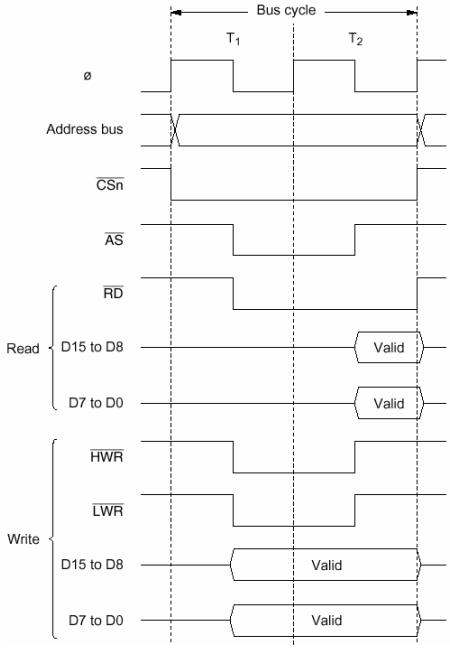
- Signals of the address bus are addresses: A[bit]
- 8-bit data:
  - Axx - A0
- 16-bit data:
  - Axx - A1
- 32-bit data:
  - Axx - A2
- Address busses that use more that 8-bit data have often separate byte addressing signals: BS3, BS2, BS1, BS0
  - In order to write separate bytes
  - It is also possible to use lower address lines
- Signal naming: A[xx:0] etc.

Data bus

- Signals of the data bus: D[bit]
- 4-bit:
  - D3 - D0
- 8-bit:
  - D7 - D0
- 16-bit:
  - D15 - D0
- 32-bit
  - D31 - D0
- Bus: D[xx:0] etc.

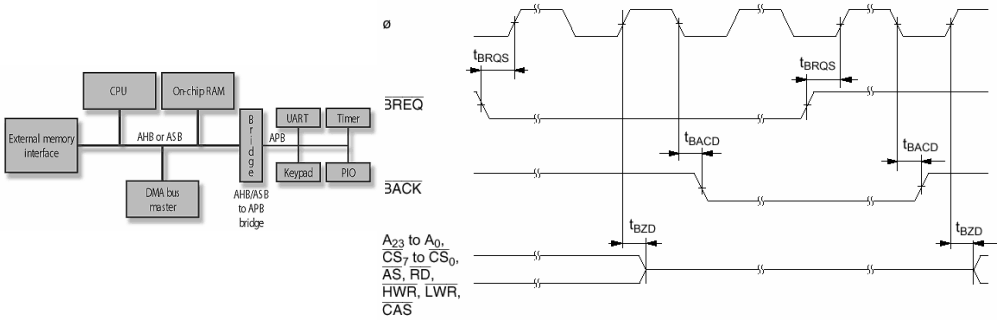
Bus cycle

- One bus cycle consists:
  - Address to the bus (at the same time giving Address Strobe signal)
  - Data to the bus when writing (WR)
  - Reading data from the bus (RD)
- All actions are synchronized to the time reference
  - E.g. machine cycle of the processor (internal clock signal, possibly divided by some value)



Releasing the bus

- Bus can be given to the control of other (than processor) device:
  - BREQ-signal is used to request the bus
  - Processor gives BACK-signal and tri-stating the bus (high-impedance state, that does not interfere the other device using the bus)



Byte order

- Little Endian (Intel):
- Lower byte is in smaller address
  - For example 16-bit value 0x8B5E:
  - For example 32-bit value 0xCC428B5E:
- E.g.. Hitachi and Fujitsu RISC-prosessors use this ordering for keeping instructions in memory.
- Big Endian (Motorola):
- Higher byte in smaller address
  - For example 16-bit value 0x8B5E (at address 2):
  - For example 32-bit value 0xCC428B5E:
- Interesting exception: Intel 80x86 – instructions in Big Endian-format but immediate addressing data is in Little Endian-format.

8-bit data bus

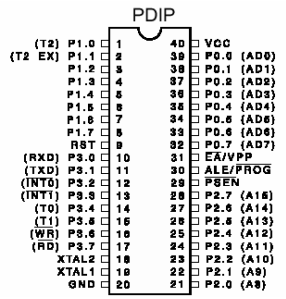
Address 0	0x5E
Address 1	0x8B
Address 2	0x42
Address 3	0xCC

8-bit data bus

Address 0	0xCC
Address 1	0x42
Address 2	0x8B
Address 3	0x5E

Bus examples: 8051

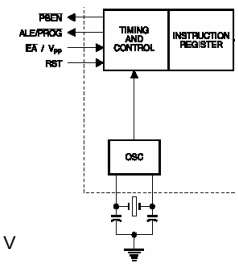
- 8-bit: 8051
- 8051 is a widely used microcontroller for applications where no performance is needed (1 execution / 12 clock cycles)
- External bus is enabled with EA-signal (External Access)
- Address space of the bus is different for address and data (Harvard), width of data bus is 8 bits. Address space is 65536 bytes
- Lower byte of the address bus is multiplexed with data bus (ALE acting as strobe)



8051: timings

- Timing diagram, (waveforms on the next page …)

- 8051 was designed before EMC directives: all external signals are synchronized on internal processor timings

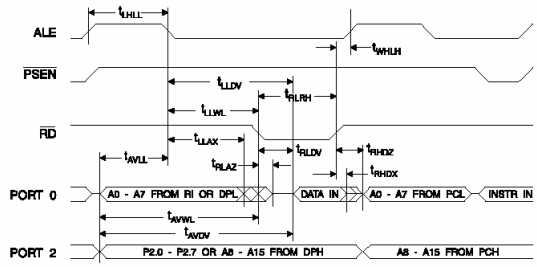


Symbol	Parameter	12 MHz Oscillator		Variable Oscillator		Units
		Min	Max	Min	Max	
$1/t_{CLCL}$	Oscillator Frequency			0	12	MHz
t_{LHLL}	ALE Pulse Width	127		$2t_{CLCL}$-40		ns
t_{AVLL}	Address Valid to ALE Low	43		t_{CLCL}-40		ns
t_{LLAX}	Address Hold After ALE Low	48		t_{CLCL}-35		ns
t_{LLIV}	ALE Low to Valid Instruction In		233		$4t_{CLCL}$-100	ns
t_{LLPL}	ALE Low to $\overline{PSEN}$ Low	43		t_{CLCL}-40		ns
t_{PLPH}	$\overline{PSEN}$ Pulse Width	205		$3t_{CLCL}$-45		ns
t_{PLIV}	$\overline{PSEN}$ Low to Valid Instruction In		145		$3t_{CLCL}$-105	ns
t_{PXIX}	Input Instruction Hold After $\overline{PSEN}$	0		0		ns
t_{PXIZ}	Input Instruction Float After $\overline{PSEN}$		59		t_{CLCL}-25	ns
t_{PXAV}	$\overline{PSEN}$ to Address Valid	75		t_{CLCL}-8		ns
t_{AVIV}	Address to Valid Instruction In		312		$5t_{CLCL}$-105	ns
t_{PLAZ}	$\overline{PSEN}$ Low to Address Float		10		10	ns
t_{RLRH}	$\overline{RD}$ Pulse Width	400		$6t_{CLCL}$-100		ns
t_{WLWH}	$\overline{WR}$ Pulse Width	400		$6t_{CLCL}$-100		ns
t_{RLDV}	$\overline{RD}$ Low to Valid Data In		252		$5t_{CLCL}$-165	ns
t_{RHDX}	Data Hold After $\overline{RD}$	0		0		ns
t_{RHDZ}	Data Float After $\overline{RD}$		97		$2t_{CLCL}$-70	ns
t_{LLDV}	ALE Low to Valid Data In		517		$8t_{CLCL}$-150	ns
t_{AVDV}	Address to Valid Data In		585		$9t_{CLCL}$-165	ns
t_{LLWL}	ALE Low to $\overline{RD}$ or $\overline{WR}$ Low	200	300	$3t_{CLCL}$-50	$3t_{CLCL}$+50	ns
t_{AVWL}	Address to $\overline{RD}$ or $\overline{WR}$ Low	203		$4t_{CLCL}$-130		ns
t_{QVWX}	Data Valid to $\overline{WR}$ Transition	23		t_{CLCL}-60		ns
t_{QVWH}	Data Valid to $\overline{WR}$ High	433		$7t_{CLCL}$-150		ns
t_{WHQX}	Data Hold After $\overline{WR}$	33		t_{CLCL}-50		ns
t_{RLAZ}	$\overline{RD}$ Low to Address Float		0		0	ns
t_{WHLH}	$\overline{RD}$ or $\overline{WR}$ High to ALE High	43	123	t_{CLCL}-40	t_{CLCL}+40	ns

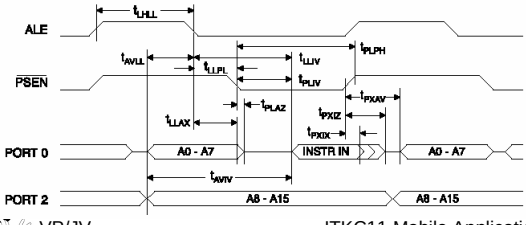
8051: timings

- Progam and data memories are separately adresssable
- PSEN address program memory (Program Store ENable)
- RD and WR address data memory (PSEN not active)

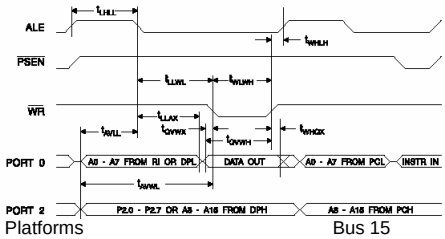
External Data Memory Read Cycle



External Program Memory Read Cycle

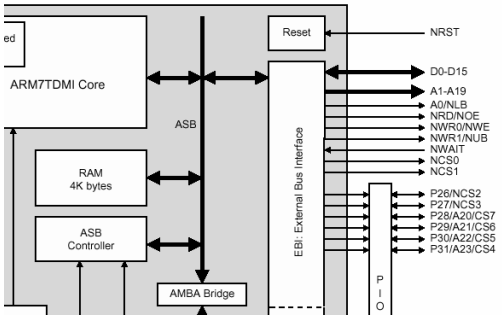


ternal Data Memory Write Cycle



Bus examples: Atmel ARM

- 16-bit: ARM
- Because ARM is licenced prosessor core, bus implementation depends on the implementor. This example is based on Atmel ARM7TDMI
- Data bus is 16-bit, A0 is not used for 16-bit data
- Address bus has 19 pins (A1-A20), max address space is therefore 2^{20} bytes (512k x 16)
- Separate A0 pin for 8-bit data
- Group of CS (Chip Select) signals,
  - These works also as upper address bus bit, therefore addressing space is 24-bit



Atmel ARM

- Atmel ARM-microcontroller (AT91M40400) bus interface (EBI):
  - 8/16-bit bus, part of the signals are multiplexed
    - Not possible to use both configurations at the same time

Name	Description	Type
A0 - A23	Address bus (output)	Output
D0 - D15	Data bus (input/output)	I/O
NCS0 - NCS3	Active low chip selects (output)	Output
CS4 - CS7	Active high chip selects (output)	Output
NRD	Read Enable (output)	Output
NWR0 - NWR1	Lower and upper write enable (output)	Output
NOE	Output enable (output)	Output
NWE	Write enable (output)	Output
NUB, NLB	Upper and lower byte select (output)	Output
NWAIT	Wait request (input)	Input

The following table shows how certain EBI signals are multiplexed:

Multiplexed Signals		Functions
A23 - A20	CS4 - CS7	Allows from 4 to 8 chip select lines to be used.
A0	NLB	8- or 16-bit data bus
NRD	NOE	Byte-write or byte select access
NWR0	NWE	Byte-write or byte select access
NWR1	NUB	Byte-write or byte select access

Atmel ARM

- Bus interconnection possibilities:

Figure 8. Memory Connection for an 8-Bit Data Bus

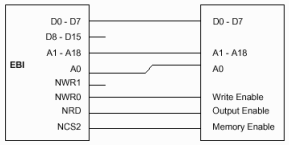


Figure 9. Memory Connection for a 16-Bit Data Bus

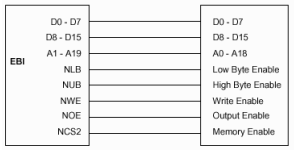


Figure 10. Memory Connection for 2 x 8-Bit Data Busses

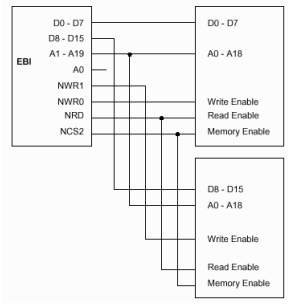


Figure 11. Connection for a 16-Bit Data Bus with byte and half word access

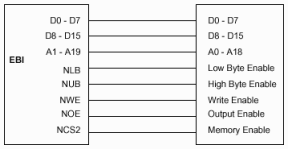
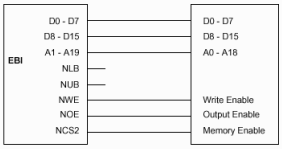


Figure 12. Connection for a 16-Bit Data Bus Without Byte Write Capability.



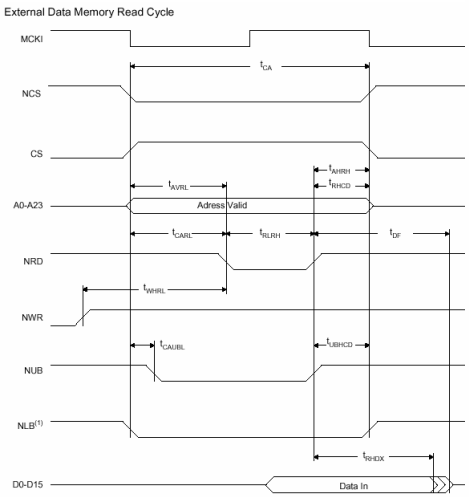
Atmel ARM: timings

- Timing table:
- T_{CP} = time of one processor clock cycle

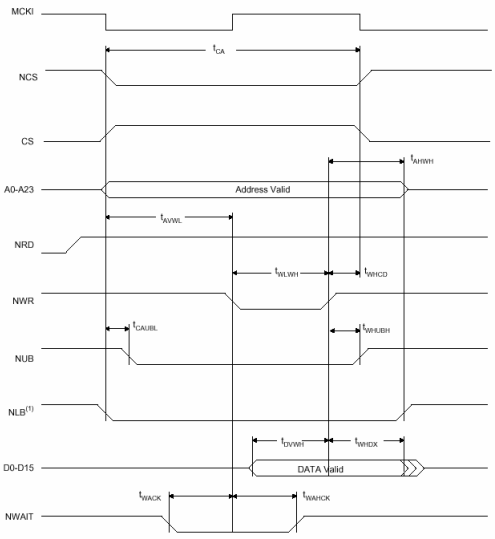
Symbol	Parameter	Minimum	Maximum AT91M40400 -25AC	Maximum AT91M40400 -25AI	Maximum AT91M40400 -33AC	Units
t_{AHRH}	Address Hold After NRD High	0				ns
t_{AHWH}	Address Hold After NWR High (no wait)	2				ns
	Address Hold After NWR High (wait)	$t_{CP/2}$				ns
t_{AVRL}	Address Valid to NRD Low	0				ns
t_{AVWL}	Address Valid to NWR Low	t_{CL} - 6				ns
t_{CA}	Chip Select Active Pulse Width	t_{CP} - 2				ns
t_{CARL}	Chip Select Active to NRD Low	0				ns
t_{CAUBL}	Chip Select Active to NUB Low	0				ns
t_{DF}	Data Float After NRD High		7(t_{CP})			ns
t_{DVWH}	Data Out Valid Before NWR High	t_{CH} - 5				ns
t_{RHCD}	NRD High to Chip Select Disabled	0				ns
t_{RHDX}	Data Hold After NRD High	0				ns
t_{RLRH}	NRD Pulse Width	t_{CP} - 2[1]				ns
		t_{CP} /2 - 2[2]				ns
t_{UBHCD}	NUB High to Chip Select Disabled	0				ns
t_{WACK}	NWAIT Setup Before MCKI High	8				ns
t_{WAHCK}	NWAIT Hold After MCKI High	4				ns
t_{WHCD}	NWR High to Chip Select Disabled (no wait)	2				ns
	NWR High to Chip Select Disabled (wait)	$t_{CP/2}$				ns
t_{WHDX}	Data Out Hold After NWR High[2] (no wait)	2				ns
	Data Out Hold After NWR High[2] (wait)	$t_{CP/2}$				ns
	Data Out Hold After NWR High[1]	$t_{CP/2}$				ns
t_{WHRL}	NWR High to NRD Low	t_{CL}				ns
t_{WHUBH}	NWR High to NUB High	2				ns
t_{WLWH}	NWR Pulse Width	t_{CP} /2 - 2				ns

Atmel ARM: timings

External Data Memory Read Cycle

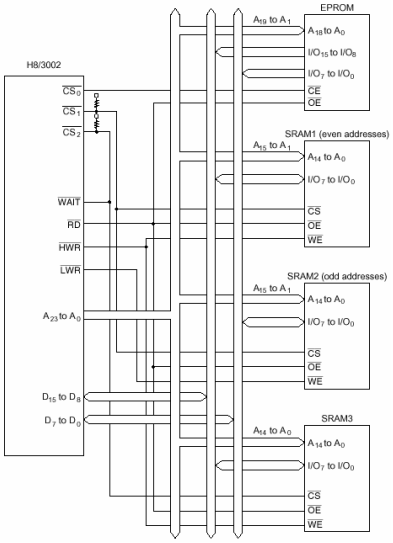


External Data Memory Write Cycle



Bus examples: Hitachi H8(S)

- 8/16-bit: Hitachi H8/3002:
- Databus D15 - D0, upper part D15 - D8 act as a 8-bit data bus (RD, HWR), SRAM3 in the picture
- A0-address is not fed to the16-bit memory (EPROM in the picture) because cycles are 16-bit
- Bus configuration is similar in all H8 and H8S processors

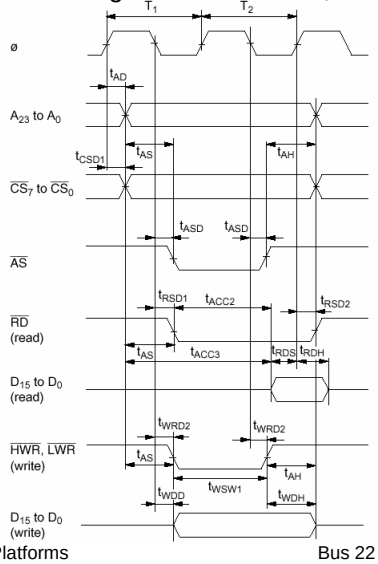


Hitachi: timings

- A: 2.7 - 3.6V, <= 20 MHz
- B: 3.0 - 3.6V, <= 25 MHz

Item	Symbol	Condition A Min	Condition A Max	Condition B Min	Condition B Max	Unit	Test Conditions
WR delay time 1	t_{WRD1}	—	20	—	15	ns	Figures 7.6 to 7.13
WR delay time 2	t_{WRD2}	—	20	—	15	ns	
WR pulse width 1	t_{WSW1}	1.0 × t_{cyc} – 20	—	1.0 × t_{cyc} – 15	—	ns	
WR pulse width 2	t_{WSW2}	1.5 × t_{cyc} – 20	—	1.5 × t_{cyc} – 15	—	ns	
Write data delay time	t_{WDD}	—	30	—	20	ns	
Write data setup time	t_{WDS}	0.5 × t_{cyc} – 20	—	0.5 × t_{cyc} – 15	—	ns	
Write data hold time	t_{WDH}	0.5 × t_{cyc} – 10	—	0.5 × t_{cyc} – 8	—	ns	
WR setup time	t_{WCS}	0.5 × t_{cyc} – 10	—	0.5 × t_{cyc} – 10	—	ns	
WR hold time	t_{WCH}	0.5 × t_{cyc} – 10	—	0.5 × t_{cyc} – 10	—	ns	
CAS setup time	t_{CSR}	0.5 × t_{cyc} – 10	—	0.5 × t_{cyc} – 8	—	ns	Figure 7.10
WAIT setup time	t_{WTS}	30	—	25	—	ns	Figure 7.8
WAIT hold time	t_{WTH}	5	—	5	—	ns	
BREQ setup time	t_{BRQS}	30	—	30	—	ns	Figure 7.14
BACK delay time	t_{BACD}	—	15	—	15	ns	
Bus floating time	t_{BZD}	—	50	—	40	ns	
BREQO delay time	t_{BRQOD}	—	30	—	25	ns	Figure 7.15

Basic timings: 2-state access, H8S

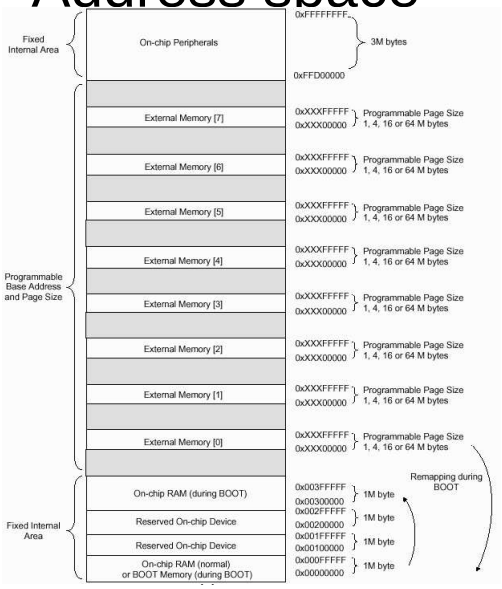


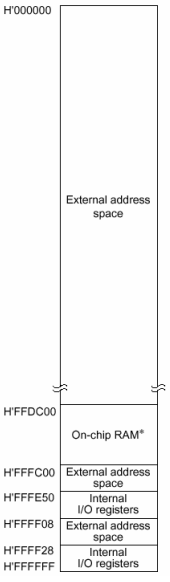
Bus interfacing

- Bus interfaces the processor to the outside world
- There are memories (ROM/RAM/Flash jne…) and peripheral devices connected to the bus
  - There can be multiple numbers of those memories and peripheral devices
- Processor bus and perihperal device may be needed to have “glue logic” when timings and signals does not directly fit together
- When considering timings, marginals must be taken care of
  - Bus cycle must be carried thru on every possible power supply voltages and temperatures where the device should be operated
- Processor bus and peripheral device must be electrically compatible
  - Same signalling leves must be used
- Whatever is interfaced to the bus, memory map is a good aid …

Address space

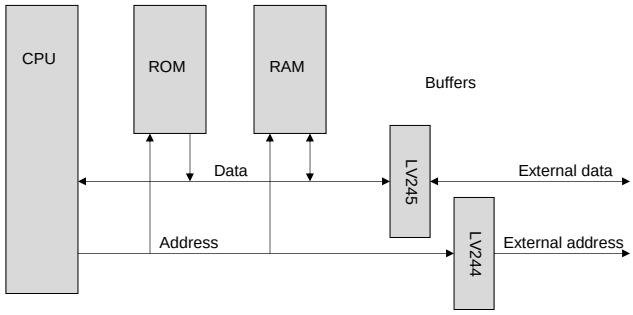
- Memory map:
- Tells what memory spaces are used and how
- Tells how peripheral devices are mapped to the memory of the microcontroller
- Tells where are RESET and interrupt vectors
- Tells possible CS-signals (Chip Select) addressing space
- Tells where the internal ROM/RAM-memory resides
- It is the first drawing, the designer must be draw





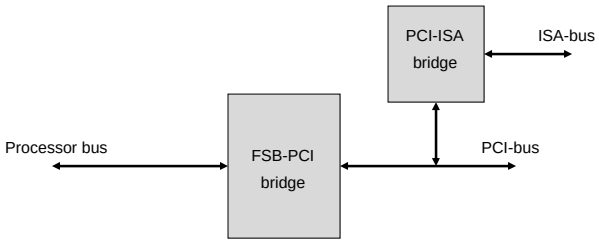
Extending the processor bus

- Extending the processor bus is done with buffers
  - Lessens the current load from the peripherals
  - Decreases the interference from the bus



Extending buses

- Bridge circuit:
  - Converts bus from one format to the other
    - Signalling and protocol are also changed
    - There are also buffers in the bridge circuit
  - Example: PC processor bus (FSB) ↔ PCI-bus ↔ ISA-bus



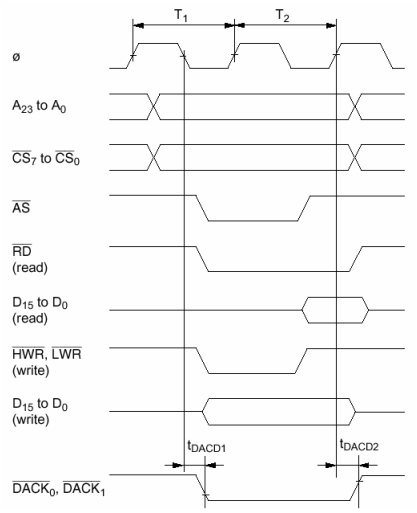
DMA

- DMA i.e. Direct Memory Access
- Enables the external bus control
- DMA is used for transferring relatively large data blocks from peripheral device to the processor's memory space
- Transfer is done by programming the (internal) DMA-controller to use the appropriate memory locations. After that, the peripheral device requests the access to the bus (DREQ) and the processor reply by releasing the bus (DACK). After the transfer, the peripheral device set the ending signal (TEND) after the processor takes the control of the bus and continues the normal operation
- In real-time systems, the length of DMA-transfer can't be long
  - Because the processor cannot use the bus, and therefore its operation is suspended

DMA

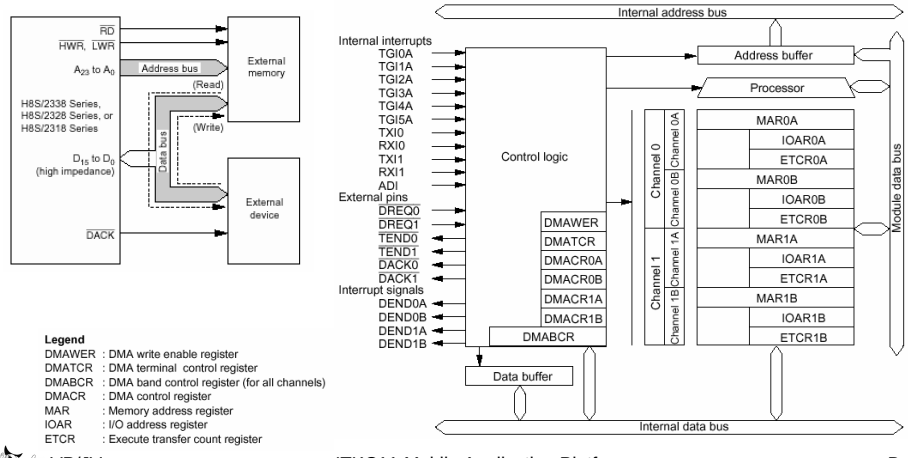
- During the DMA-transfer, processor bus is in tri-state
  - In order to release the bus for the external user
- In the picture is shown one DMA-transfer, normally there are more cycles in the transfer (Hitachi H8S/23xx)

		Condition A		Condition B		
Item	Symbol	Min	Max	Min	Max	Unit
$\overline{DREQ}$ setup time	t_{DRQS}	30	—	25	—	ns
$\overline{DREQ}$ hold time	t_{DRQH}	10	—	10	—	
$\overline{TEND}$ delay time	t_{TED}	—	20	—	18	
$\overline{DACK}$ delay time 1	t_{DACD1}	—	20	—	18	ns
$\overline{DACK}$ delay time 2	t_{DACD2}	—	20	—	18	



DMA

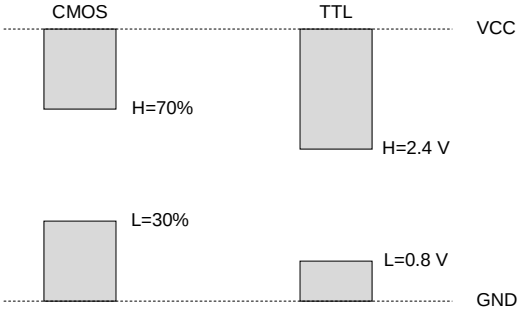
- DMA-controller (Hitachi H8S/23xx):



Legend
DMAWER : DMA write enable register
DMATCR : DMA terminal control register
DMABCR : DMA band control register (for all channels)
DMACR : DMA control register
MAR : Memory address register
IOAR : I/O address register
ETCR : Execute transfer count register

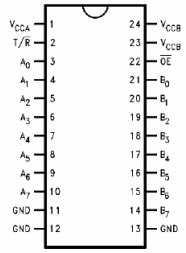
Level conversions

- Is neede when logic with different supply voltages is connected together
- For example 3.3 - 5V and vice versa
- In addition there could be (on the bus) devices with different logic leves that should be connected together, e.g. HC- and HCT -devices



Level conversions

- E.g. 74LVX4245, LVX3245
- Two power supply 245
- Enables bidirectional data bus level conversions to both directions



Pin Descriptions

Pin Names	Description
$\overline{OE}$	Output Enable Input
T/$\overline{R}$	Transmit/Receive Input
A_0–A_7	Side A Inputs or 3-STATE Outputs
B_0–B_7	Side B Inputs or 3-STATE Outputs

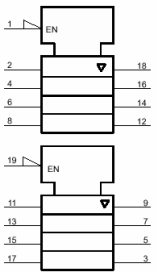
Truth Table

Inputs		Outputs
$\overline{OE}$	T/$\overline{R}$	
L	L	Bus B Data to Bus A
L	H	Bus A Data to Bus B
H	X	HIGH-Z State

H = HIGH Voltage Level
L = LOW Voltage Level
X = Immaterial

Level conversions

- From higher voltage to lower
- Example: LVC244: 5V -> 1.8V
- LVC-series logic stands up to 5 volts at inputs even if power supply is 1.8 V
- Inputs are 5V-tolerants

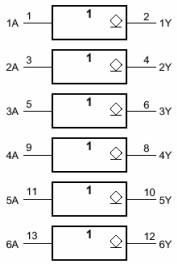


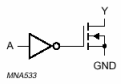
PIN DESCRIPTION

PIN NUMBER	SYMBOL	FUNCTION
1	1$\overline{OE}$	Output enable input (active LOW)
2, 4, 6, 8	$1A_0$ to $1A_3$	Data inputs
3, 5, 7, 9	$2Y_0$ to $2Y_3$	Bus outputs
10	GND	Ground (0V)
17, 15, 13, 11	$2A_0$ to $2A_3$	Bus inputs
18, 16, 14, 12	$1Y_0$ to $1Y_3$	Bus outputs
19	$\overline{2OE}$	Output enable input (active-LOW)
20	V_{CC}	Positive supply voltage

Level conversions

- From lower to higher voltage
- Example LVC07: 1.8V → 5V
- Output of LVC-series logic is also tolerant for 5 volt even if the supply voltage is lower than 5V





PINNING

PIN	SYMBOL	DESCRIPTION
1, 3, 5, 9, 11 and 13	1A to 6A	data inputs
2, 4, 6, 8, 10 and 12	1Y to 6Y	data outputs
7	GND	ground (0 V)
14	V_{CC}	DC supply voltage

Real world schematic

- H8/3002 + 2Mb Flash + 256kb SRAM + some buses