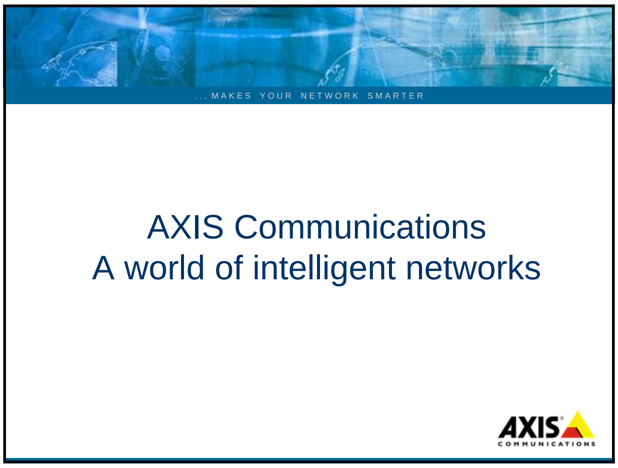


... MAKES YOUR NETWORK SMARTER

AXIS Communications
A world of intelligent networks

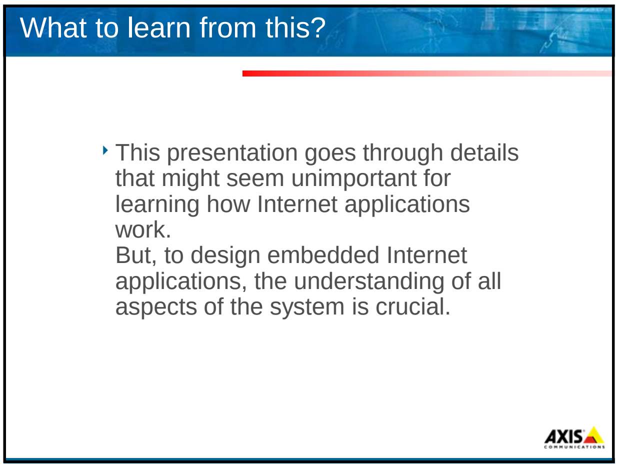
... MAKES YOUR NETWORK SMARTER

ETRAX

Per Zander

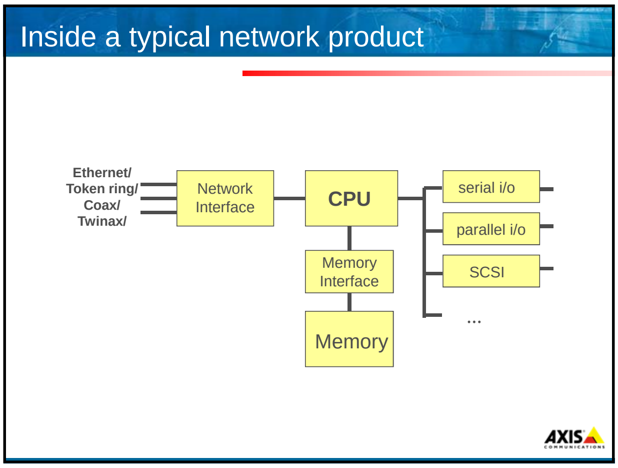
Axis Communications AB

per.zander@axis.com

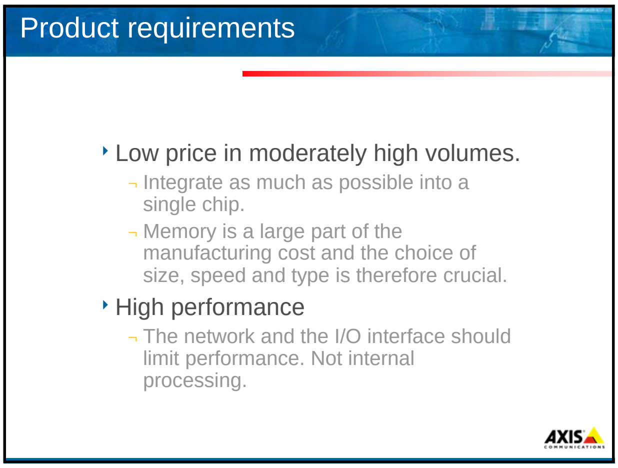


What to learn from this?

- This presentation goes through details that might seem unimportant for learning how Internet applications work.
  But, to design embedded Internet applications, the understanding of all aspects of the system is crucial.

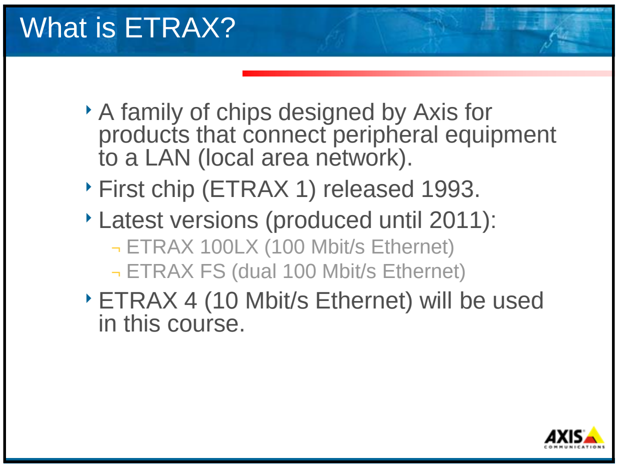


Inside a typical network product



Product requirements

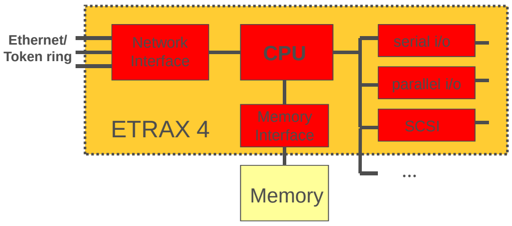
- Low price in moderately high volumes.
  - Integrate as much as possible into a single chip.
  - Memory is a large part of the manufacturing cost and the choice of size, speed and type is therefore crucial.
- High performance
  - The network and the I/O interface should limit performance. Not internal processing.



What is ETRAX?

- A family of chips designed by Axis for products that connect peripheral equipment to a LAN (local area network).
- First chip (ETRAX 1) released 1993.
- Latest versions (produced until 2011):
  - ETRAX 100LX (100 Mbit/s Ethernet)
  - ETRAX FS (dual 100 Mbit/s Ethernet)
- ETRAX 4 (10 Mbit/s Ethernet) will be used in this course.

ETRAX 4 Overview



ETRAX 4 is not a CPU!
It’s a System on a Chip.

ETRAX 4 Overview

- Network controller for Ethernet (802.3) and Token ring (802.5) networks.
- CPU with an architecture (CRIS) optimized for this type of embedded systems.
- Two advanced parallel ports for connecting all types of printers and other parallel devices.
- SCSI-2 interface.
- RS-232 serial port.

ETRAX 4 Overview

- Two DMA channels.
- Timer for generating periodic interrupts.
- Memory interface that can connect standard memory types and speeds without external logic.
- General purpose I/O pins.
- Internal registers (mode registers) for controlling all I/O interfaces.

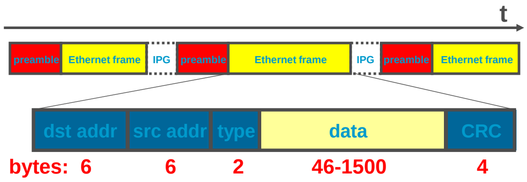
... MAKES YOUR NETWORK SMARTER

Data Communication Repetition

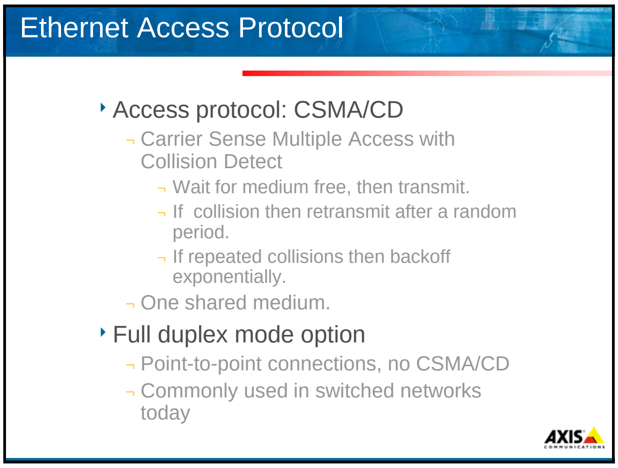
Ethernet

- Ethernet is a communication protocol at OSI layer 1 and 2 (physical and data link).
- It was created in the early 80:s and has then evolved from 10 Mbit/s performance to today's 1 Gbit/s and 10 Gbit/s.
- The original usage was for communication between a small number of scientific computers.

Ethernet Packet Format

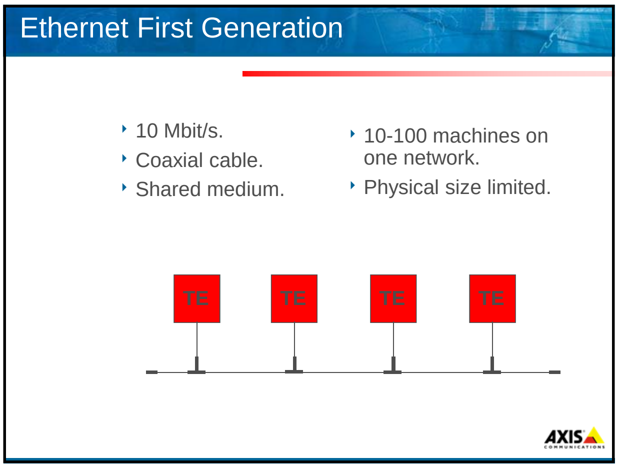


- Data is divided into variable length packets.
- Minimum frame size 64 bytes and maximum 1518 bytes.
- Minimum inter packet gap 96 bits.



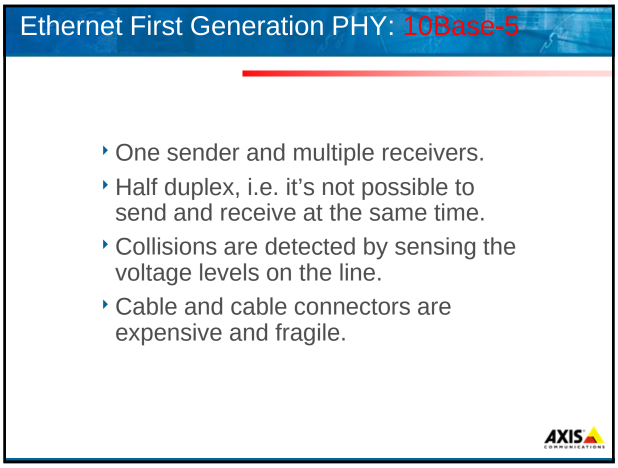
Ethernet Access Protocol

- Access protocol: CSMA/CD
  - Carrier Sense Multiple Access with Collision Detect
    - Wait for medium free, then transmit.
    - If collision then retransmit after a random period.
    - If repeated collisions then backoff exponentially.
  - One shared medium.
- Full duplex mode option
  - Point-to-point connections, no CSMA/CD
  - Commonly used in switched networks today



Ethernet First Generation

- 10 Mbit/s.
- Coaxial cable.
- Shared medium.
- 10-100 machines on one network.
- Physical size limited.

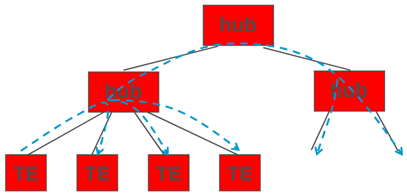


Ethernet First Generation PHY: 10Base-5

- One sender and multiple receivers.
- Half duplex, i.e. it’s not possible to send and receive at the same time.
- Collisions are detected by sensing the voltage levels on the line.
- Cable and cable connectors are expensive and fragile.

Ethernet PHY: 10Base-T

- The shared coaxial cable is replaced with hubs and dedicated cables.
- 10 Mbit/s.
- Unshielded twisted pair cable (UTP).

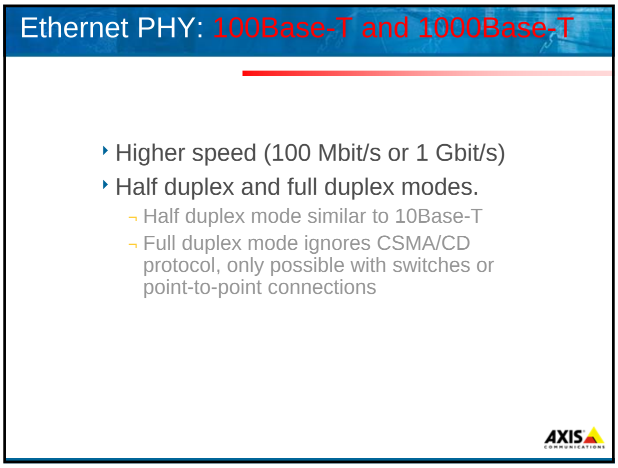


Ethernet PHY: 10Base-T

- Logically but not physically shared medium. Still CSMA/CD.
  - Point-to-point links, i.e. the cable is dedicated to one station.
  - Hubs are active devices that propagate all transmissions and collisions to all stations with no delay or buffering.
- Half duplex (one sender at a time).

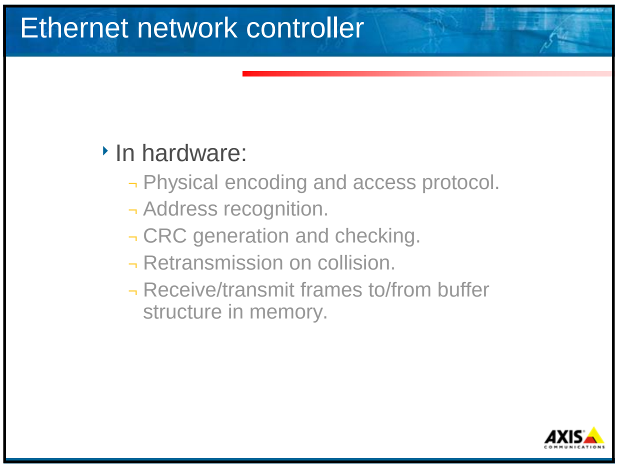
Ethernet PHY: 10Base-T

- The cable is a cheap unshielded twisted pair cable (voice grade telephone cable). One twisted pair for transmission and one pair for receiving.
- Collisions are detected by simultaneous activity on receive and transmit pair.



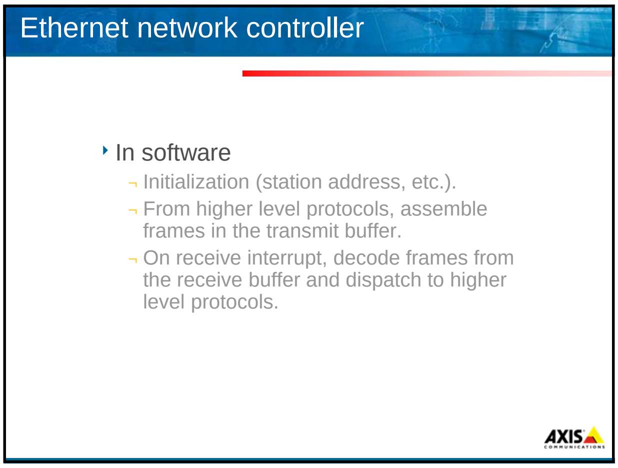
Ethernet PHY: 100Base-T and 1000Base-T

- Higher speed (100 Mbit/s or 1 Gbit/s)
- Half duplex and full duplex modes.
  - Half duplex mode similar to 10Base-T
  - Full duplex mode ignores CSMA/CD protocol, only possible with switches or point-to-point connections



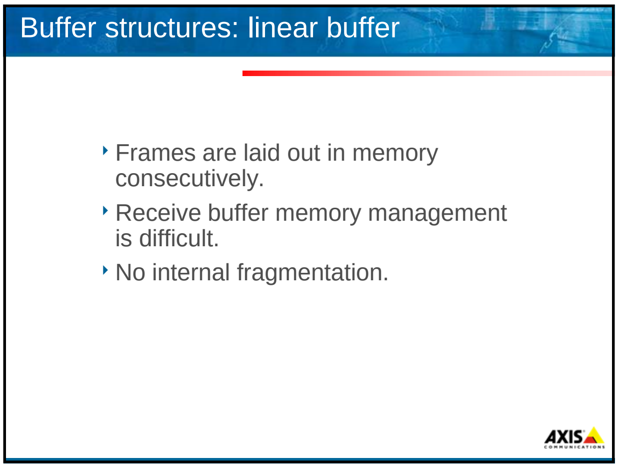
Ethernet network controller

- In hardware:
  - Physical encoding and access protocol.
  - Address recognition.
  - CRC generation and checking.
  - Retransmission on collision.
  - Receive/transmit frames to/from buffer structure in memory.



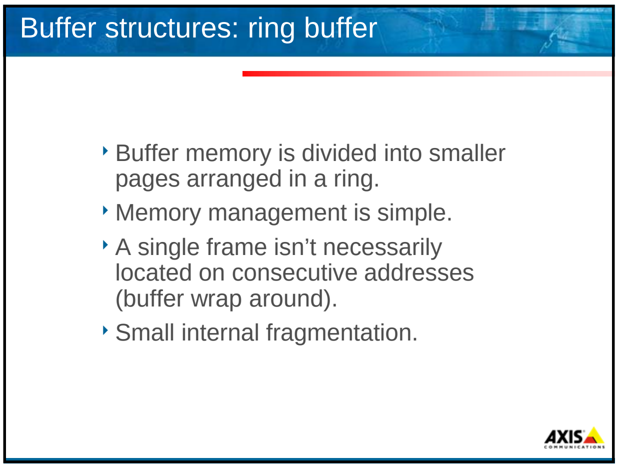
Ethernet network controller

- In software
  - Initialization (station address, etc.).
  - From higher level protocols, assemble frames in the transmit buffer.
  - On receive interrupt, decode frames from the receive buffer and dispatch to higher level protocols.



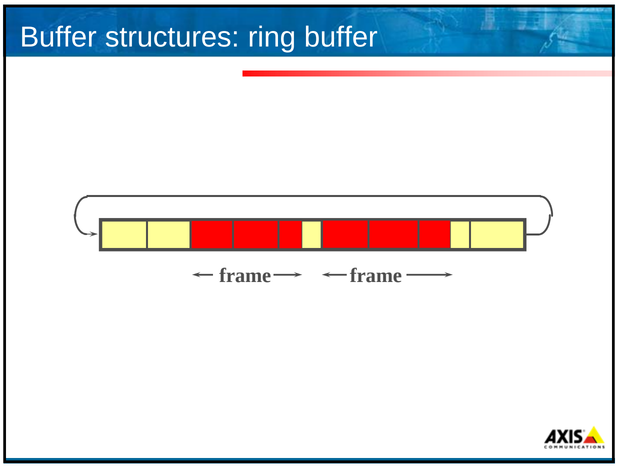
Buffer structures: linear buffer

- Frames are laid out in memory consecutively.
- Receive buffer memory management is difficult.
- No internal fragmentation.

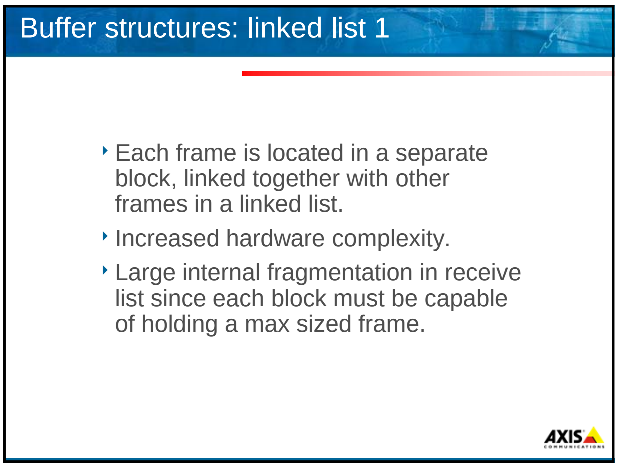


Buffer structures: ring buffer

- Buffer memory is divided into smaller pages arranged in a ring.
- Memory management is simple.
- A single frame isn't necessarily located on consecutive addresses (buffer wrap around).
- Small internal fragmentation.

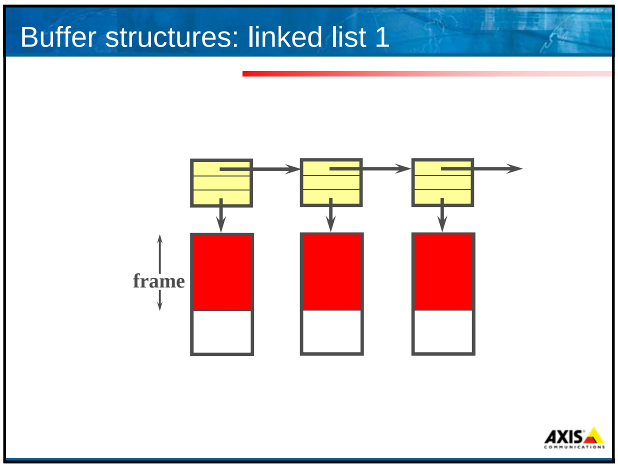


Buffer structures: ring buffer

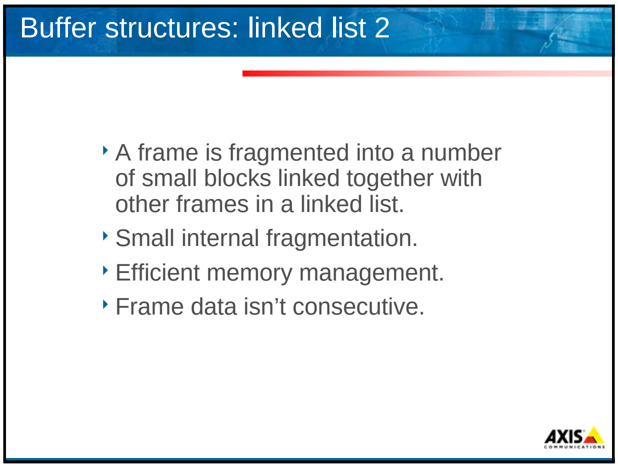


Buffer structures: linked list 1

- Each frame is located in a separate block, linked together with other frames in a linked list.
- Increased hardware complexity.
- Large internal fragmentation in receive list since each block must be capable of holding a max sized frame.



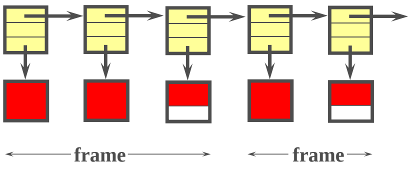
Buffer structures: linked list 1



Buffer structures: linked list 2

- A frame is fragmented into a number of small blocks linked together with other frames in a linked list.
- Small internal fragmentation.
- Efficient memory management.
- Frame data isn’t consecutive.

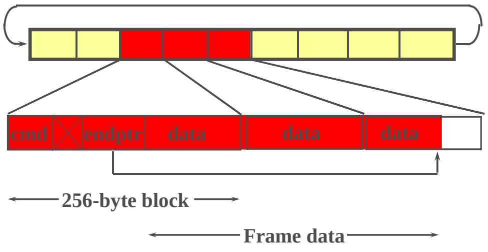
Buffer structures: linked list 2

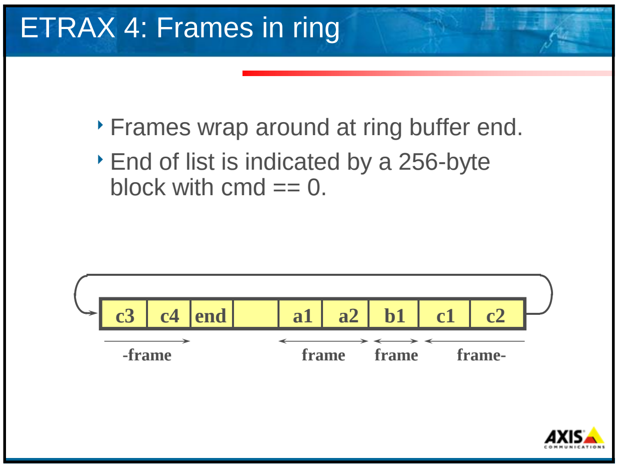


ETRAX 4 buffer structure

- Ring buffer, size: 2k - 64k.
- Divided into 256-byte blocks.
- Protocol headers fit within first block.
- Frames linked together by an end of frame pointer.

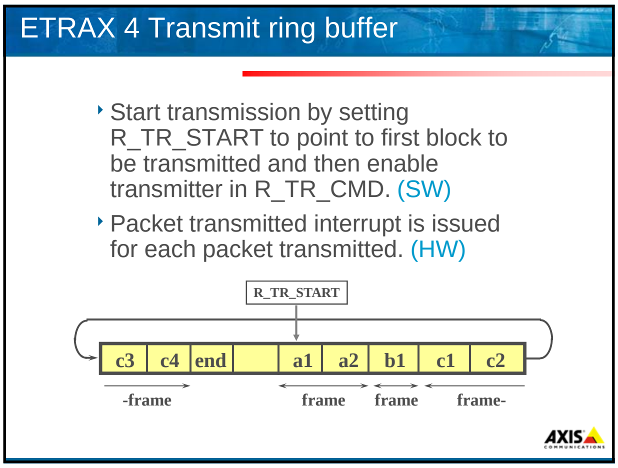
ETRAX 4 Frame structure





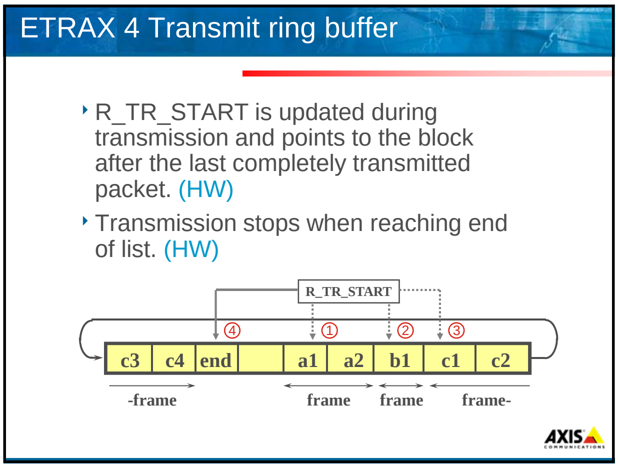
ETRAX 4: Frames in ring

- Frames wrap around at ring buffer end.
- End of list is indicated by a 256-byte block with cmd == 0.



ETRAX 4 Transmit ring buffer

- Start transmission by setting R_TR_START to point to first block to be transmitted and then enable transmitter in R_TR_CMD. (SW)
- Packet transmitted interrupt is issued for each packet transmitted. (HW)

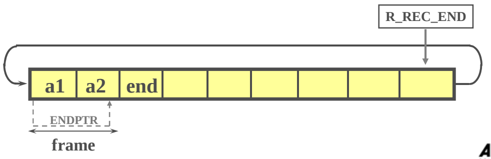


ETRAX 4 Transmit ring buffer

- R_TR_START is updated during transmission and points to the block after the last completely transmitted packet. (HW)
- Transmission stops when reaching end of list. (HW)

ETRAX 4 Receive ring buffer

- Receiver starts at block 0 in ring buffer after reset. (HW)
- When a complete packet is received, CMD and ENDPTR for the packet is updated, and CMD for the block after the packet is set to “end of list” (cmd == 0). (HW)

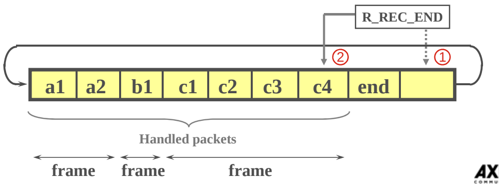


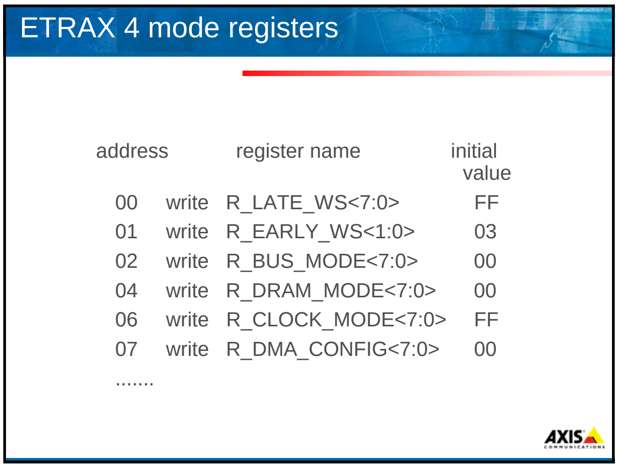
ETRAX 4 Receive ring buffer

- Receive packet interrupt is issued for each packet written into buffer. (HW)
- Buffer full condition occurs when data is written into the block pointed to by R_REC_END. (HW)
- Received packets are handled by traversing the buffer from last previously processed receive packet till end of list (cmd == 0). (SW)

ETRAX 4 Receive ring buffer

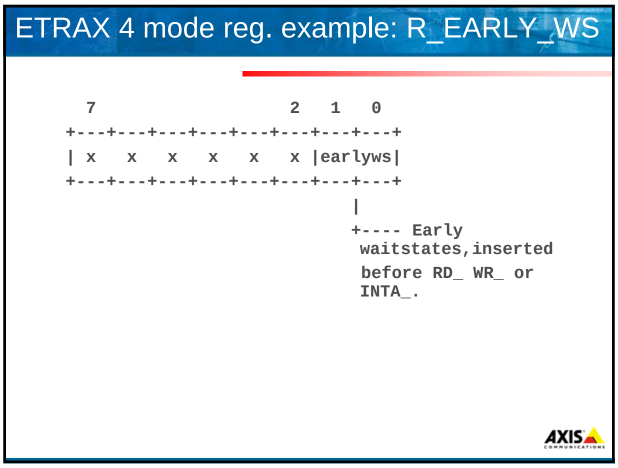
- When a receive packet has been handled, the space it occupies is made available for new packets by moving R_REC_END forward to the end of the handled packet. (SW)





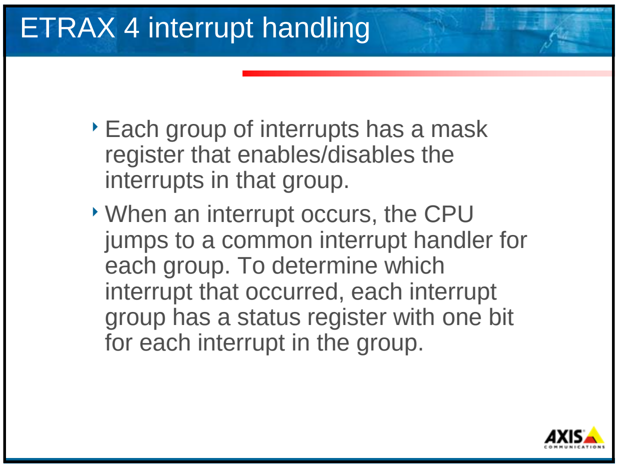
ETRAX 4 mode registers

address		register name	initial value
00	write	R_LATE_WS<7:0>	FF
01	write	R_EARLY_WS<1:0>	03
02	write	R_BUS_MODE<7:0>	00
04	write	R_DRAM_MODE<7:0>	00
06	write	R_CLOCK_MODE<7:0>	FF
07	write	R_DMA_CONFIG<7:0>	00
....			



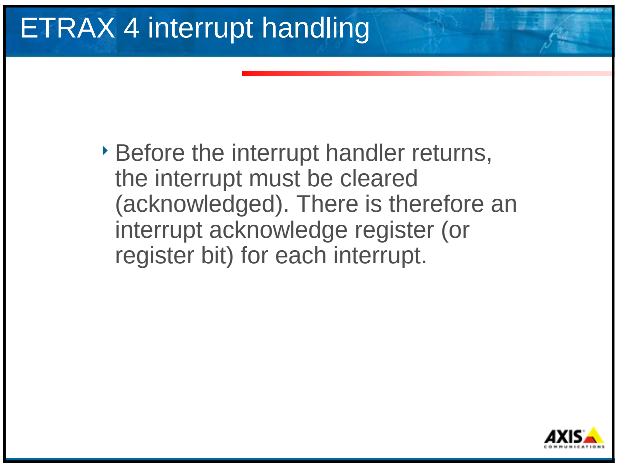
ETRAX 4 mode reg. example: R_EARLY_WS

```
  7                   2   1   0
+---+---+---+---+---+---+---+---+
| x   x   x   x   x   x |earlyws|
+---+---+---+---+---+---+---+---+
                          |
                          +---- Early
                           waitstates,inserted
                           before RD_ WR_ or
                           INTA_.
```



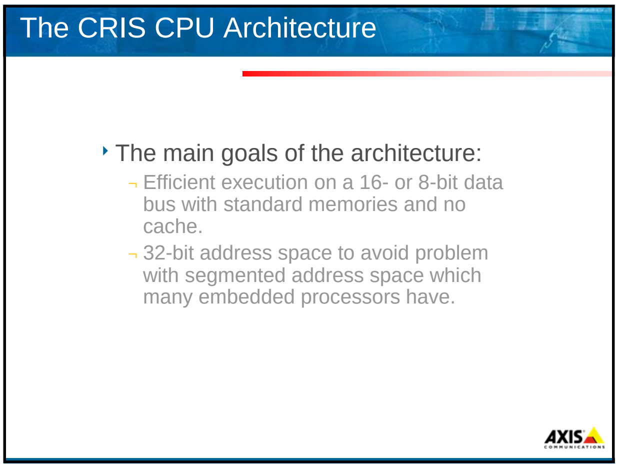
ETRAX 4 interrupt handling

- Each group of interrupts has a mask register that enables/disables the interrupts in that group.
- When an interrupt occurs, the CPU jumps to a common interrupt handler for each group. To determine which interrupt that occurred, each interrupt group has a status register with one bit for each interrupt in the group.



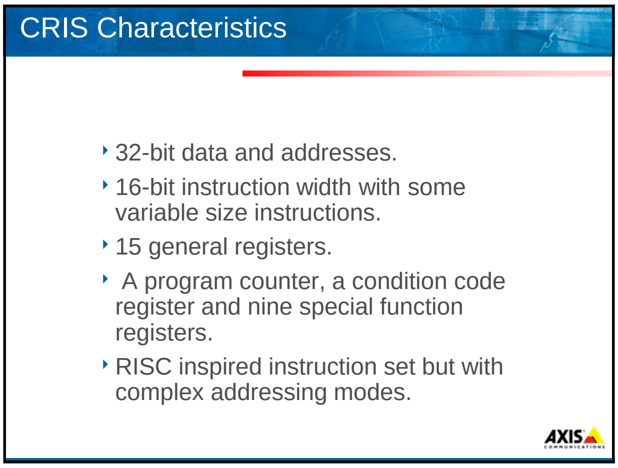
ETRAX 4 interrupt handling

- Before the interrupt handler returns, the interrupt must be cleared (acknowledged). There is therefore an interrupt acknowledge register (or register bit) for each interrupt.



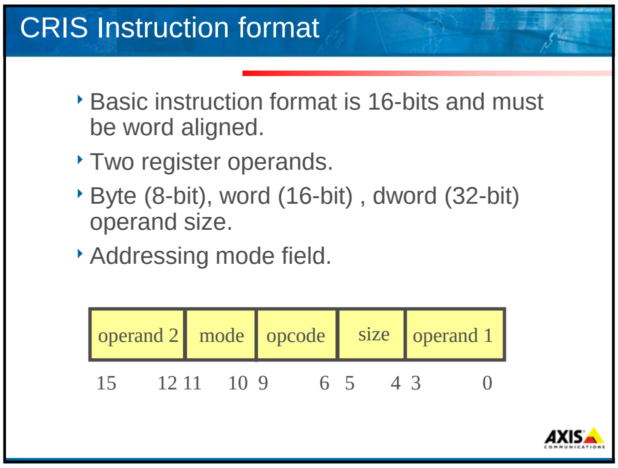
The CRIS CPU Architecture

- The main goals of the architecture:
  - Efficient execution on a 16- or 8-bit data bus with standard memories and no cache.
  - 32-bit address space to avoid problem with segmented address space which many embedded processors have.



CRIS Characteristics

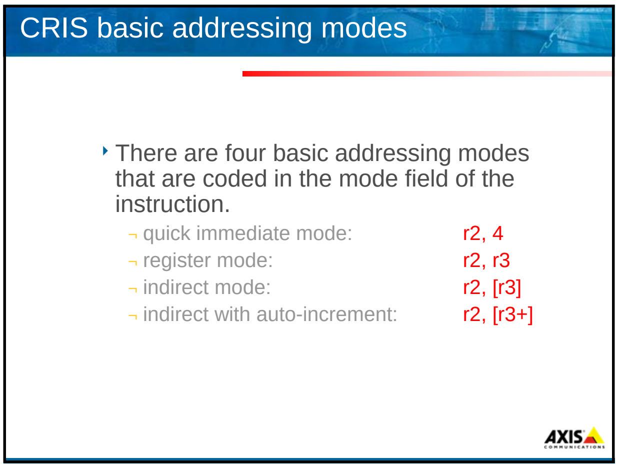
- 32-bit data and addresses.
- 16-bit instruction width with some variable size instructions.
- 15 general registers.
- A program counter, a condition code register and nine special function registers.
- RISC inspired instruction set but with complex addressing modes.



CRIS Instruction format

- Basic instruction format is 16-bits and must be word aligned.
- Two register operands.
- Byte (8-bit), word (16-bit) , dword (32-bit) operand size.
- Addressing mode field.

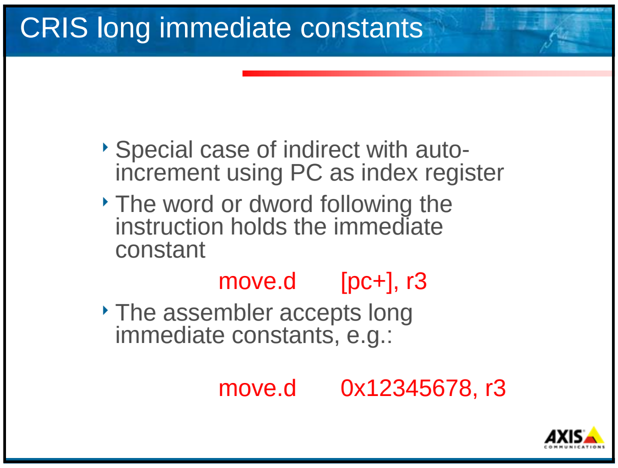
operand 2	mode	opcode	size	operand 1
15 12	11 10	9 6	5 4	3 0



CRIS basic addressing modes

- There are four basic addressing modes that are coded in the mode field of the instruction.

quick immediate mode:	r2, 4
register mode:	r2, r3
indirect mode:	r2, [r3]
indirect with auto-increment:	r2, [r3+]



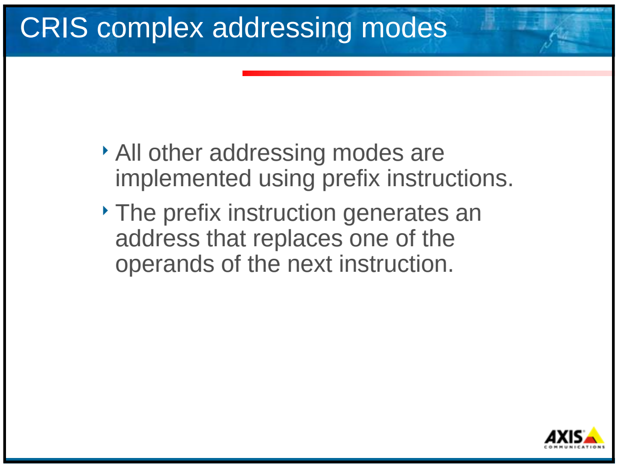
CRIS long immediate constants

- Special case of indirect with auto-increment using PC as index register
- The word or dword following the instruction holds the immediate constant

```
move.d    [pc+], r3
```

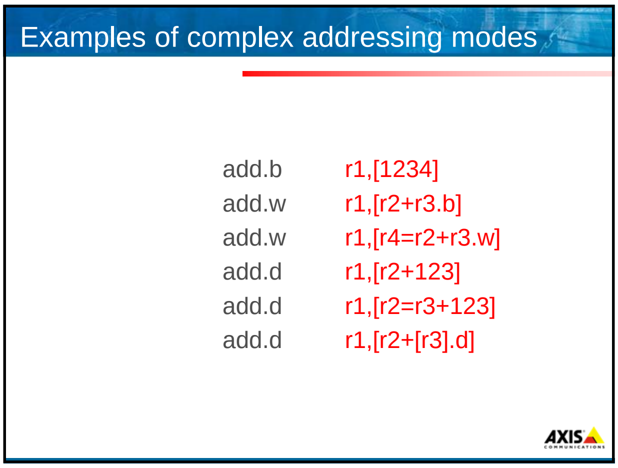
- The assembler accepts long immediate constants, e.g.:

```
move.d    0x12345678, r3
```



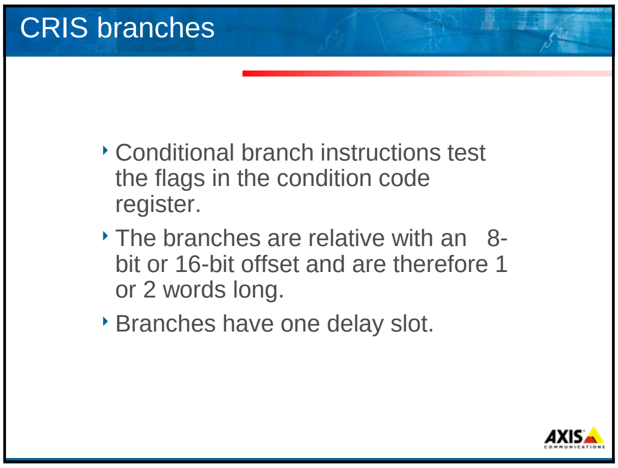
CRIS complex addressing modes

- All other addressing modes are implemented using prefix instructions.
- The prefix instruction generates an address that replaces one of the operands of the next instruction.



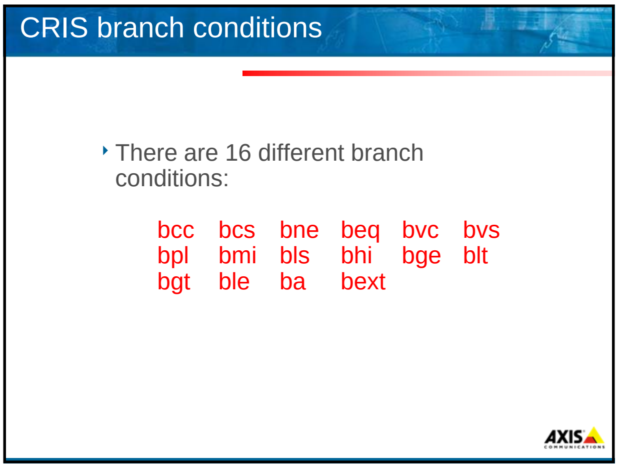
Examples of complex addressing modes

```
add.b    r1,[1234]
add.w    r1,[r2+r3.b]
add.w    r1,[r4=r2+r3.w]
add.d    r1,[r2+123]
add.d    r1,[r2=r3+123]
add.d    r1,[r2+[r3].d]
```



CRIS branches

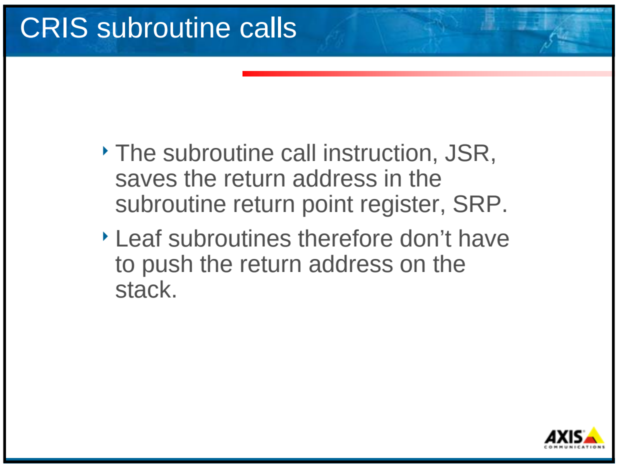
- Conditional branch instructions test the flags in the condition code register.
- The branches are relative with an 8-bit or 16-bit offset and are therefore 1 or 2 words long.
- Branches have one delay slot.



CRIS branch conditions

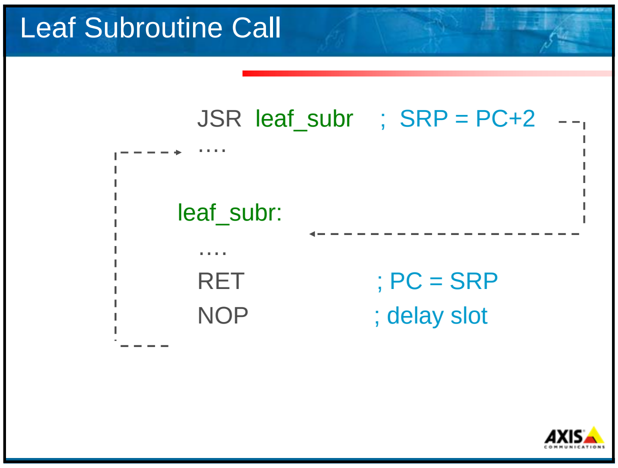
- There are 16 different branch conditions:

bcc bcs bne beq bvc bvs
bpl bmi bls bhi bge blt
bgt ble ba bext



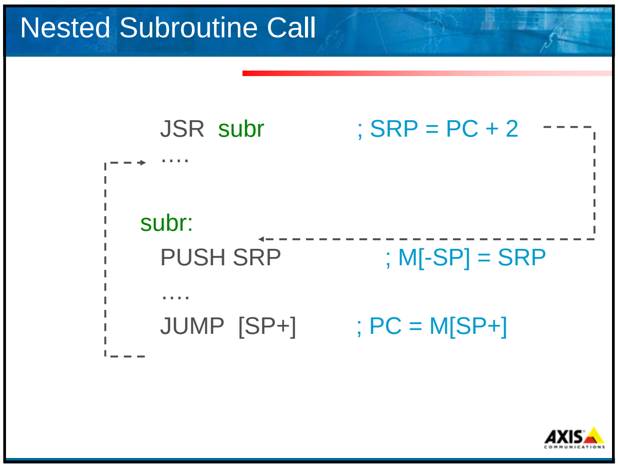
CRIS subroutine calls

- The subroutine call instruction, JSR, saves the return address in the subroutine return point register, SRP.
- Leaf subroutines therefore don’t have to push the return address on the stack.



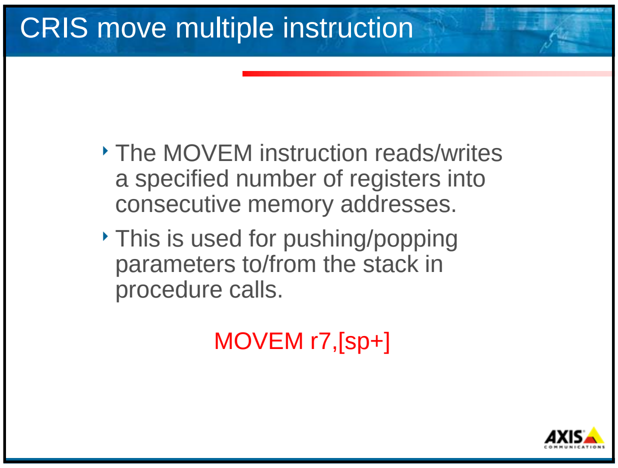
Leaf Subroutine Call

```
   JSR  leaf_subr    ;  SRP = PC+2
   ....

  leaf_subr:
   ....
   RET                ; PC = SRP
   NOP                ; delay slot
```



Nested Subroutine Call

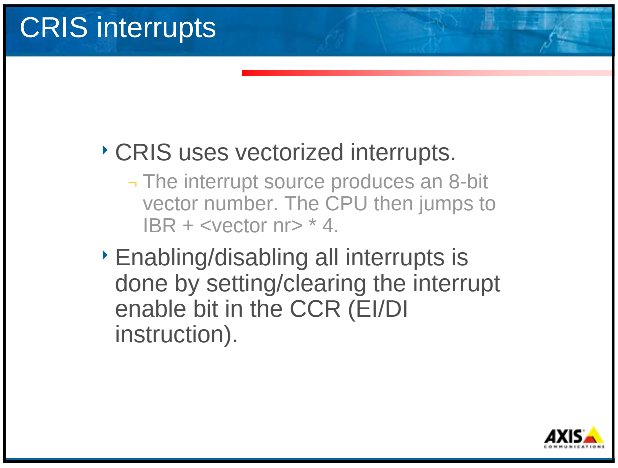
```
  JSR  subr          ; SRP = PC + 2
  ....

subr:
  PUSH SRP              ; M[-SP] = SRP
  ....
  JUMP  [SP+]        ; PC = M[SP+]
```



CRIS move multiple instruction

- The MOVEM instruction reads/writes a specified number of registers into consecutive memory addresses.
- This is used for pushing/popping parameters to/from the stack in procedure calls.

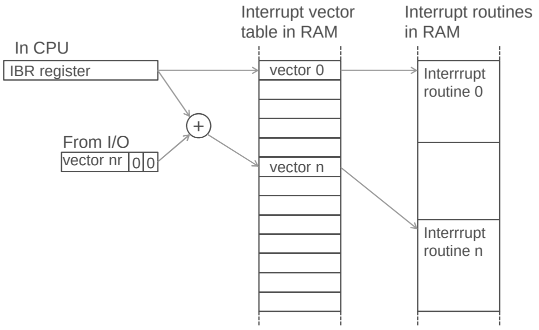
MOVEM r7,[sp+]



CRIS interrupts

- CRIS uses vectorized interrupts.
  - The interrupt source produces an 8-bit vector number. The CPU then jumps to IBR + <vector nr> * 4.
- Enabling/disabling all interrupts is done by setting/clearing the interrupt enable bit in the CCR (EI/DI instruction).

Vectorized interrupts



In CPU

IBR register

From I/O

vector nr | 0 | 0

+

Interrupt vector table in RAM

vector 0

vector n

Interrupt routines in RAM

Interrrupt routine 0

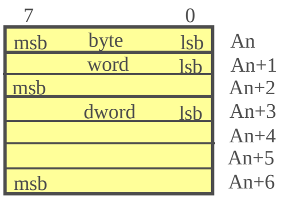
Interrrupt routine n

Data organization in memory

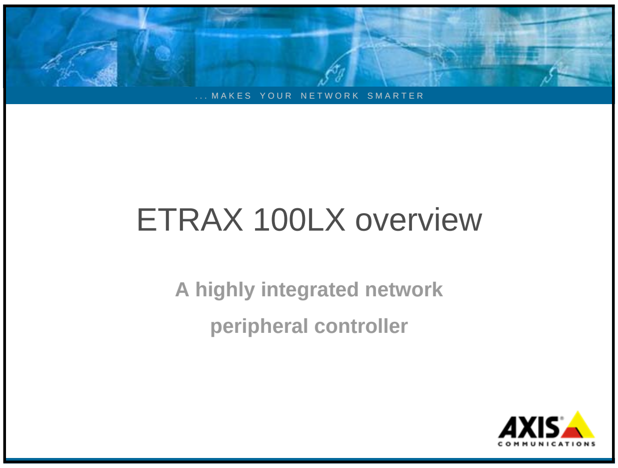
- CRIS is a little endian CPU.
- Data has no alignment restrictions, but there is a performance penalty for unaligned data accesses.
- Instructions must be word aligned.

Little endian data

Least significant bit in a word is placed in the lowest address.



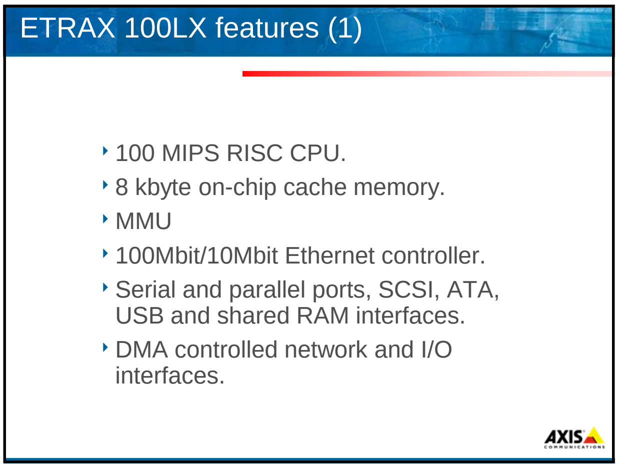
7		0	
msb	byte	lsb	An
	word	lsb	An+1
msb			An+2
	dword	lsb	An+3
			An+4
			An+5
msb			An+6



... MAKES YOUR NETWORK SMARTER

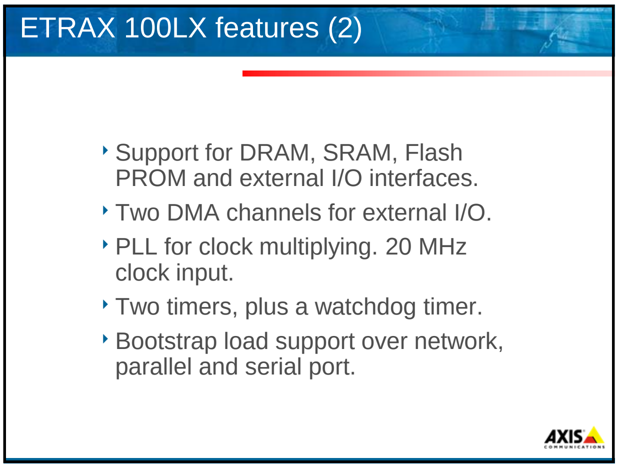
ETRAX 100LX overview

A highly integrated network peripheral controller



ETRAX 100LX features (1)

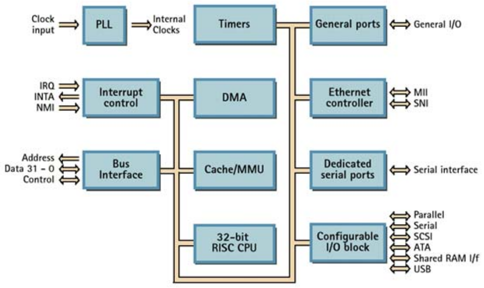
- 100 MIPS RISC CPU.
- 8 kbyte on-chip cache memory.
- MMU
- 100Mbit/10Mbit Ethernet controller.
- Serial and parallel ports, SCSI, ATA, USB and shared RAM interfaces.
- DMA controlled network and I/O interfaces.



ETRAX 100LX features (2)

- Support for DRAM, SRAM, Flash PROM and external I/O interfaces.
- Two DMA channels for external I/O.
- PLL for clock multiplying. 20 MHz clock input.
- Two timers, plus a watchdog timer.
- Bootstrap load support over network, parallel and serial port.

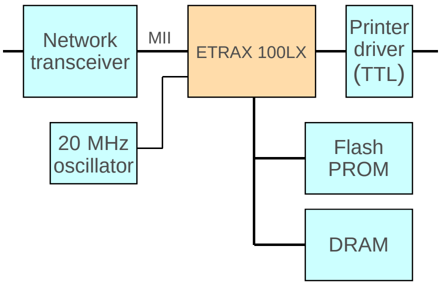
ETRAX 100LX block diagram



ETRAX 100LX characteristics

- 256-pin BGA package, 27x27x2.15 mm.
- 3.3 V, 105 mA (typ).
- 5 V-tolerant I/O.
- 0-70 C ambient temp.
- 0.25 µm process, chip size 5.6x5.6 mm.

Print server block diagram:



Network transceiver

MII

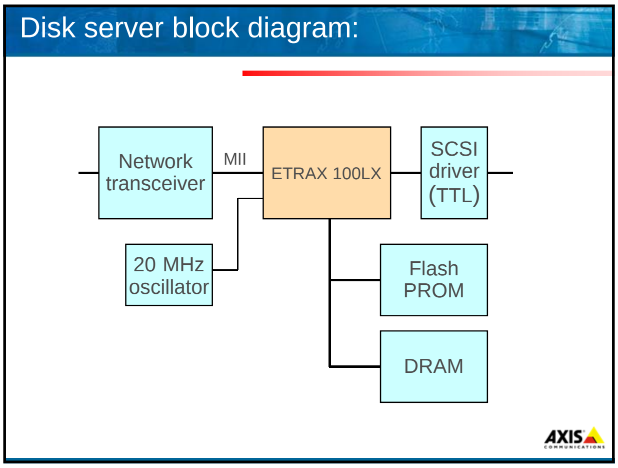
ETRAX 100LX

Printer driver (TTL)

20 MHz oscillator

Flash PROM

DRAM



Disk server block diagram:

Network transceiver

MII

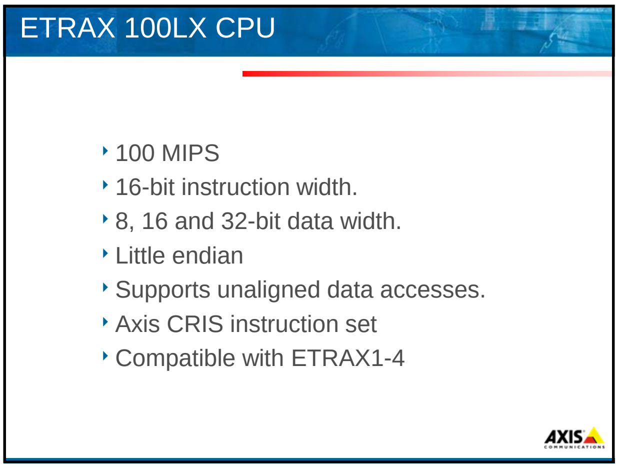
ETRAX 100LX

SCSI driver (TTL)

20 MHz oscillator

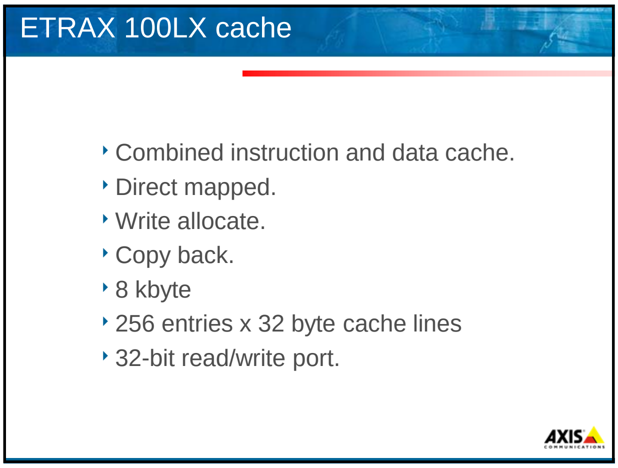
Flash PROM

DRAM



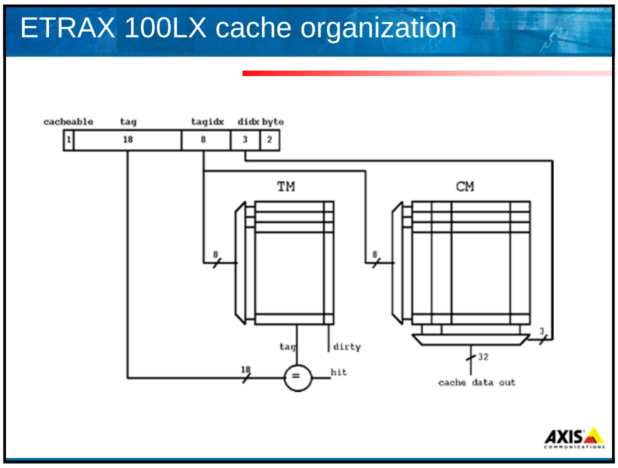
ETRAX 100LX CPU

- 100 MIPS
- 16-bit instruction width.
- 8, 16 and 32-bit data width.
- Little endian
- Supports unaligned data accesses.
- Axis CRIS instruction set
- Compatible with ETRAX1-4

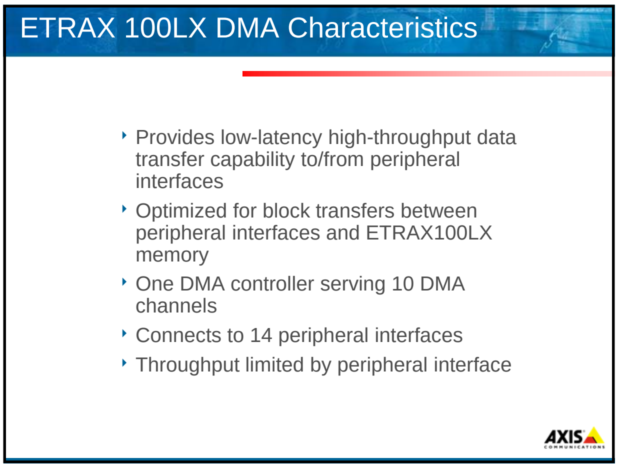


ETRAX 100LX cache

- Combined instruction and data cache.
- Direct mapped.
- Write allocate.
- Copy back.
- 8 kbyte
- 256 entries x 32 byte cache lines
- 32-bit read/write port.

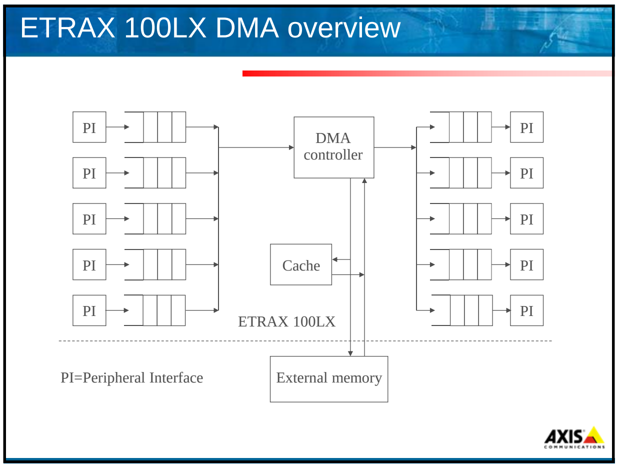


ETRAX 100LX cache organization



ETRAX 100LX DMA Characteristics

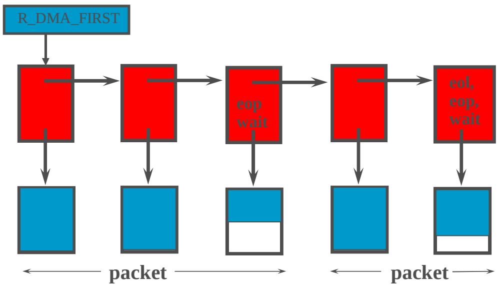
- Provides low-latency high-throughput data transfer capability to/from peripheral interfaces
- Optimized for block transfers between peripheral interfaces and ETRAX100LX memory
- One DMA controller serving 10 DMA channels
- Connects to 14 peripheral interfaces
- Throughput limited by peripheral interface



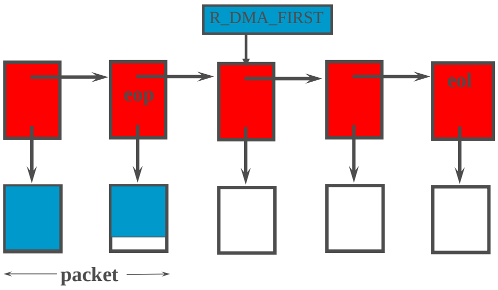
ETRAX 100LX DMA overview

PI=Peripheral Interface

ETRAX 100LX DMA Transmit list

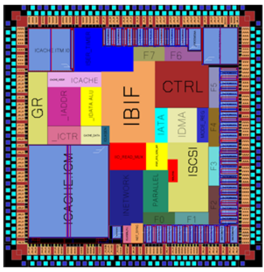


ETRAX 100LX DMA Receive list

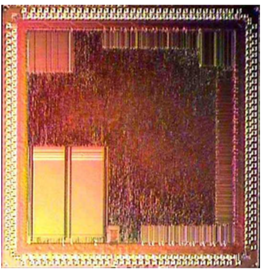


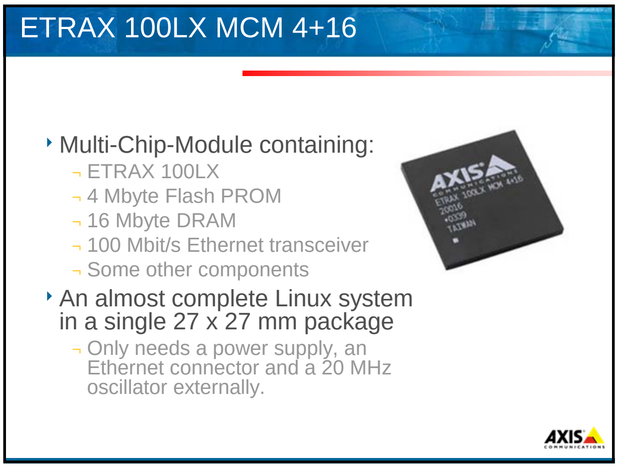
ETRAX 100 (Predecessor to ETRAX 100LX)

Floorplan



Chip photo

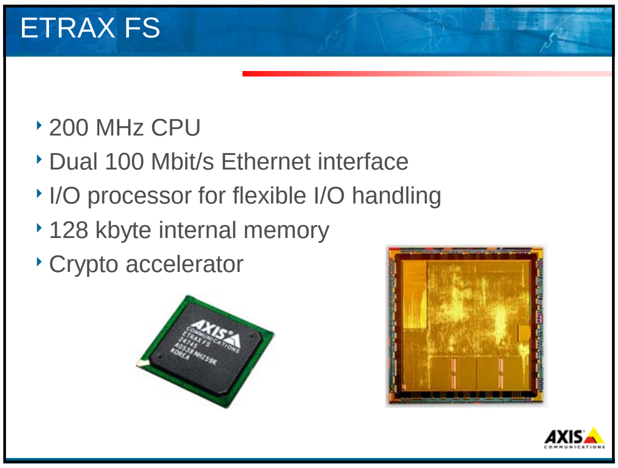




ETRAX 100LX MCM 4+16

- Multi-Chip-Module containing:
  - ETRAX 100LX
  - 4 Mbyte Flash PROM
  - 16 Mbyte DRAM
  - 100 Mbit/s Ethernet transceiver
  - Some other components
- An almost complete Linux system in a single 27 x 27 mm package
  - Only needs a power supply, an Ethernet connector and a 20 MHz oscillator externally.

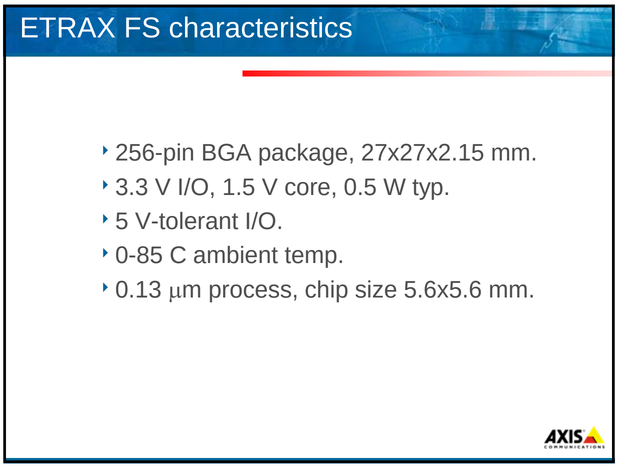
AXIS
ETRAX 100LX MCM 4+16
20016
+0339
TAIWAN



ETRAX FS

- 200 MHz CPU
- Dual 100 Mbit/s Ethernet interface
- I/O processor for flexible I/O handling
- 128 kbyte internal memory
- Crypto accelerator

AXIS
ETRAX FS



ETRAX FS characteristics

- 256-pin BGA package, 27x27x2.15 mm.
- 3.3 V I/O, 1.5 V core, 0.5 W typ.
- 5 V-tolerant I/O.
- 0-85 C ambient temp.
- 0.13 μm process, chip size 5.6x5.6 mm.