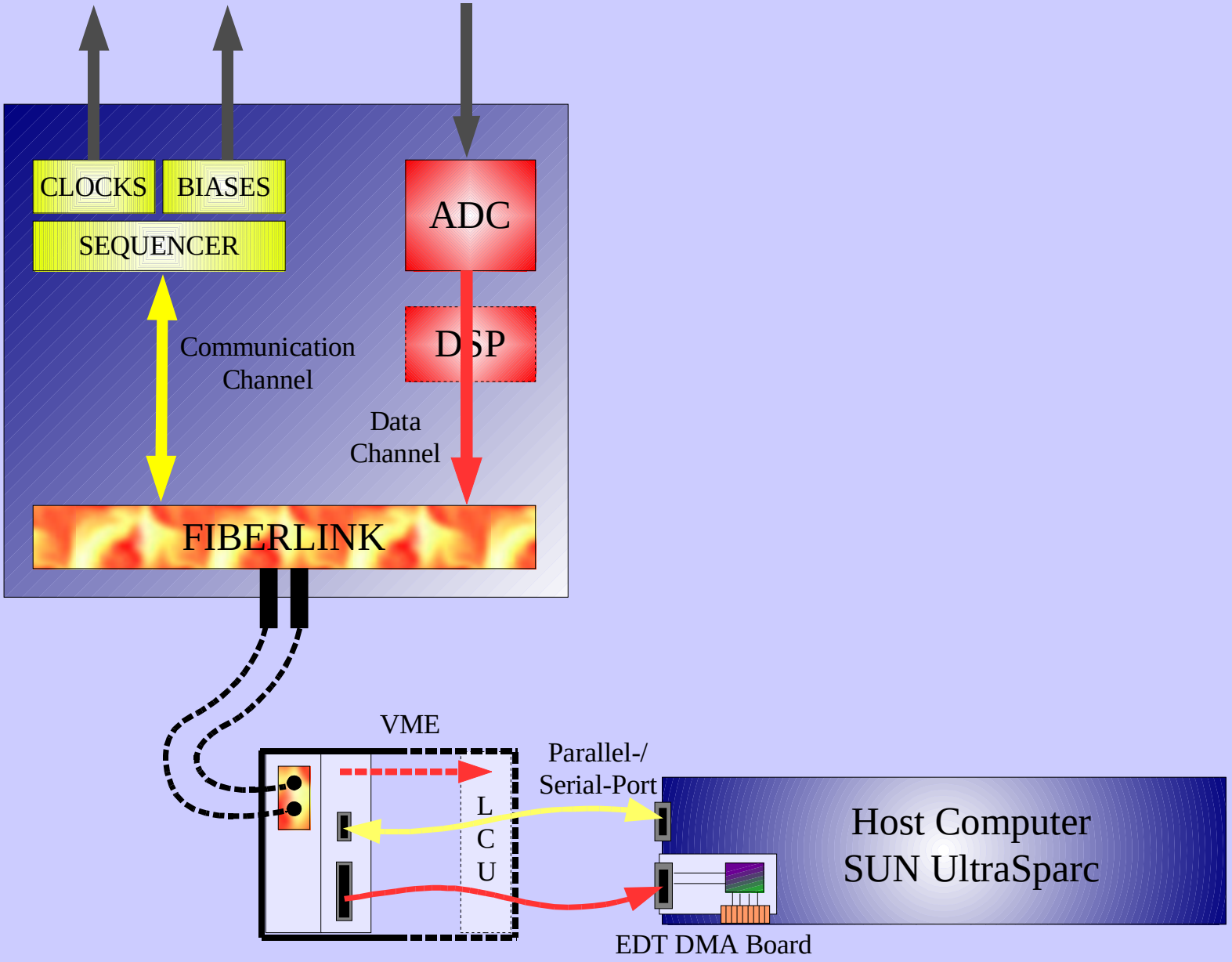
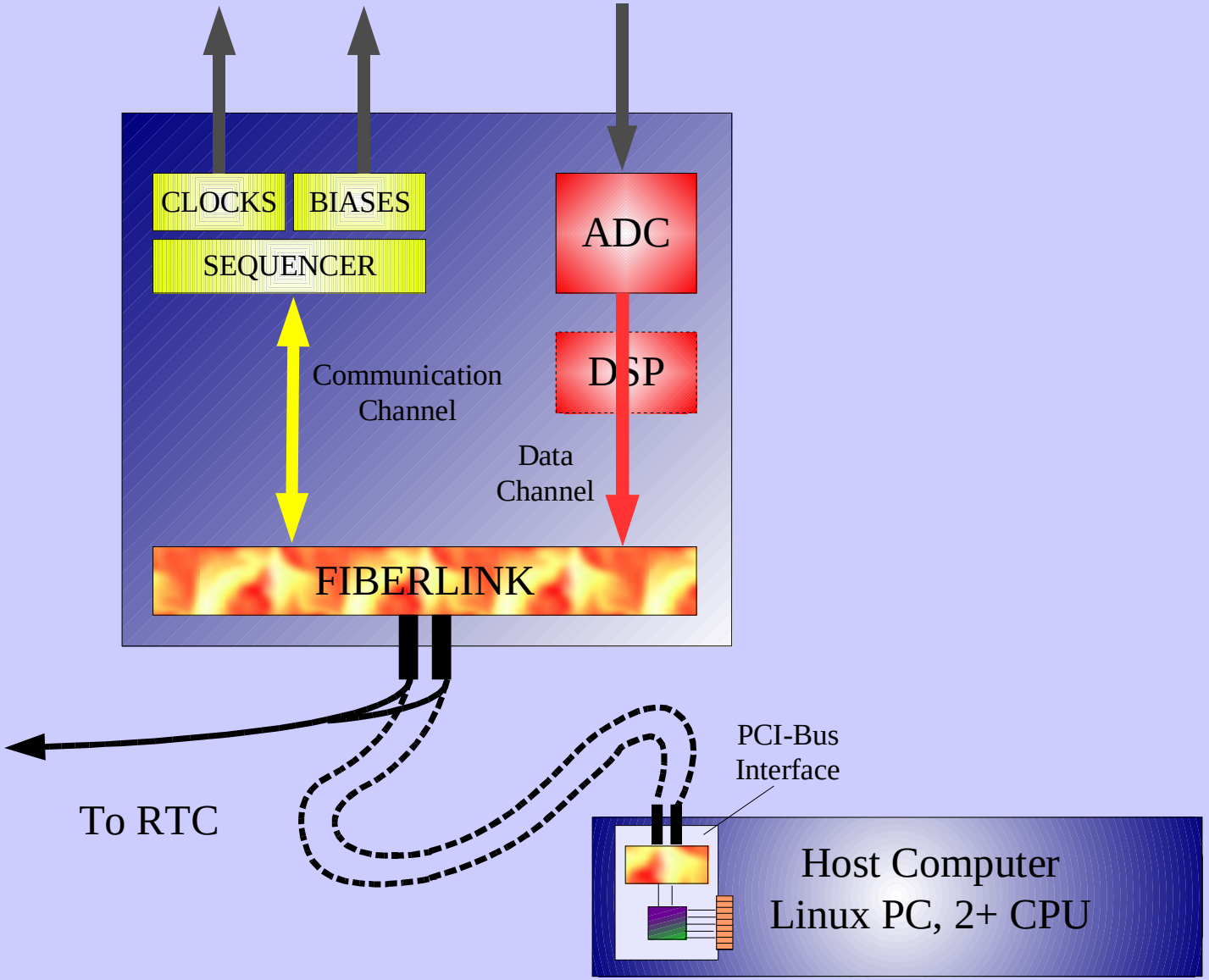
Hardware Structure



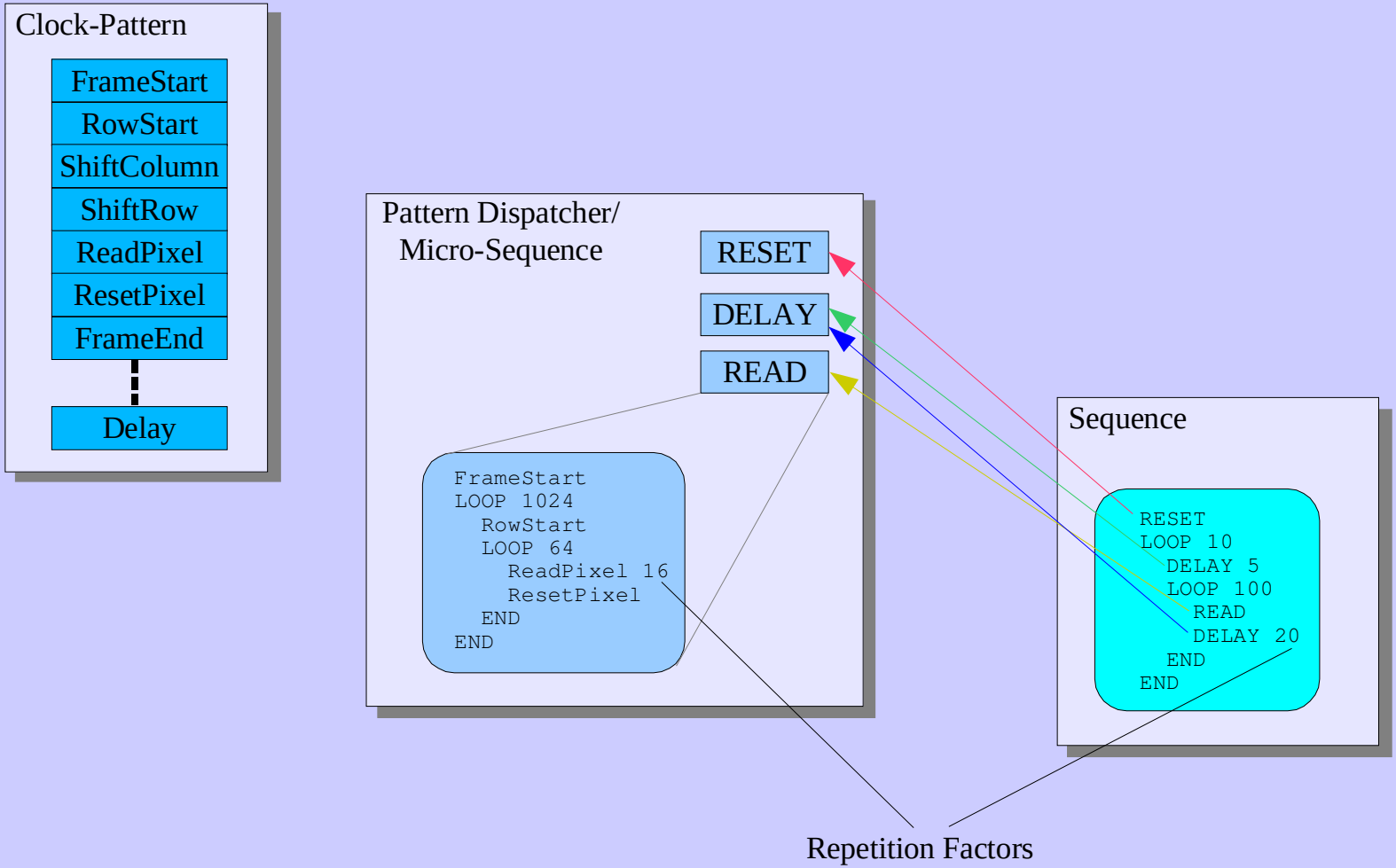
Hardware Structure



Physical Interfaces

- EDT DMA Interface Board plus Serial/Parallel Port
  - Driver for data-channel delivered by EDT (Linux, Solaris)
  - Driver for communication-channel is part of the operating system
  - Back-End and interface cables required
- PCI-Bus Interface
  - Driver development by ESO (Linux, Solaris)
  - Data-channel and communication-channel in one board
  - No Back-End required
  - Use commercial product instead?
    - Saves driver development
    - Protocol might be an overkill
    - Must support continuous ring-buffer DMA (most products do not!)
- VME-Bus Interface
  - Needed for data transfer to LCU (PowerPC / VxWorks)
  - VxWorks driver development by ESO

Sequencer Programming



Sequencer Programming

- Clock-Patterns (smallest unit) are stored in sequencer "memory".
- The loops are downloaded to sequencer as (checked) structural code and not in ASCII format.
- ASCII format is interpreted at higher level.
- Loop parameters and pattern repetition factors must be derived from arbitrary functions depending on DIT, NDIT, NDSAMPLES, NDSKIP, ...
- Special tokens like SYNC ("wait for external trigger").
- Loop structures can be executed in real time by FPGA (no processor required).
- Logical hierarchy in three steps (pattern / micro-sequence / sequence) should be kept.

Digital Signal Processing

- Digital Signal Processing might be required to reduce traffic on Fiber-Link.
- To be placed on front-end.
- Currently not required by Infrared Applications.
- Can be done via DSP or be programmed in FPGA.

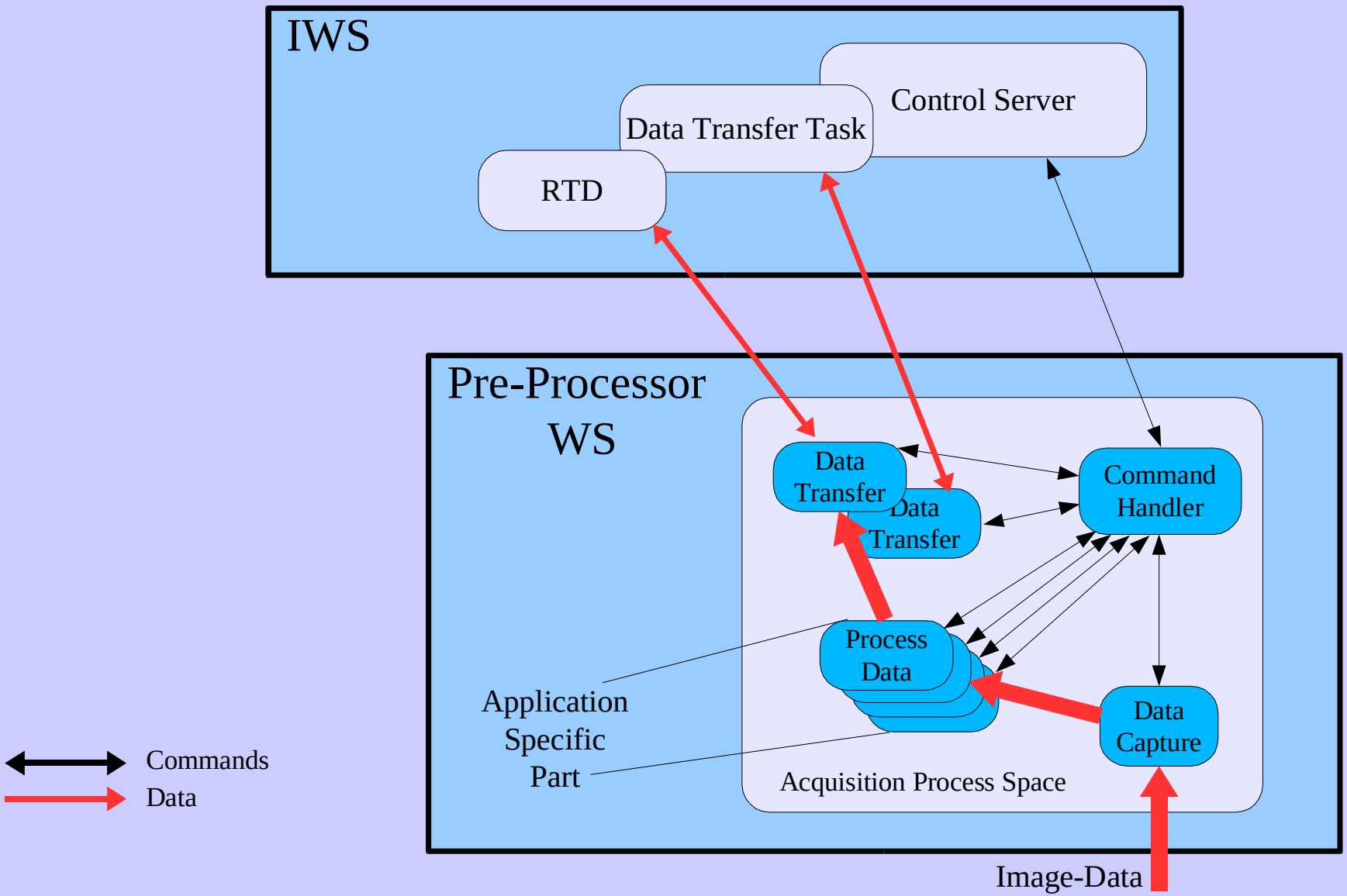
Image Processing – Acquisition Loop

- Image Processing generates arbitrary numbers and types of frames during one exposure.
- Parameter sets are fully application specific.
- Continuous input loop via DMA without memory copies (ring-buffer mode).
- Parallel processing algorithms on the host computer to benefit from multiple CPU architectures (Multi-Threading).
- Burst Mode for large raw data transfer to Disk/Network. Limitations are the amount of memory and the PCI-Bus bandwidth.
- Priority based scheduling to grant maximum CPU usage to the processing algorithm.
- No hard real-time requirements for imaging.
- High memory needs up to several GBytes.
- High processing speed requirements. The Current Limit is: 190 MBytes/s for Double Correlated Averaging on a 3 GHz Dual Processor PC and Linux RH 7.3.
- Maximum Frame-Rate (=Interrupt Rate) is 200 KHz.
- If the system limits (bus bandwidth, memory, CPU power) are reached, the image processing can be distributed across several computers.

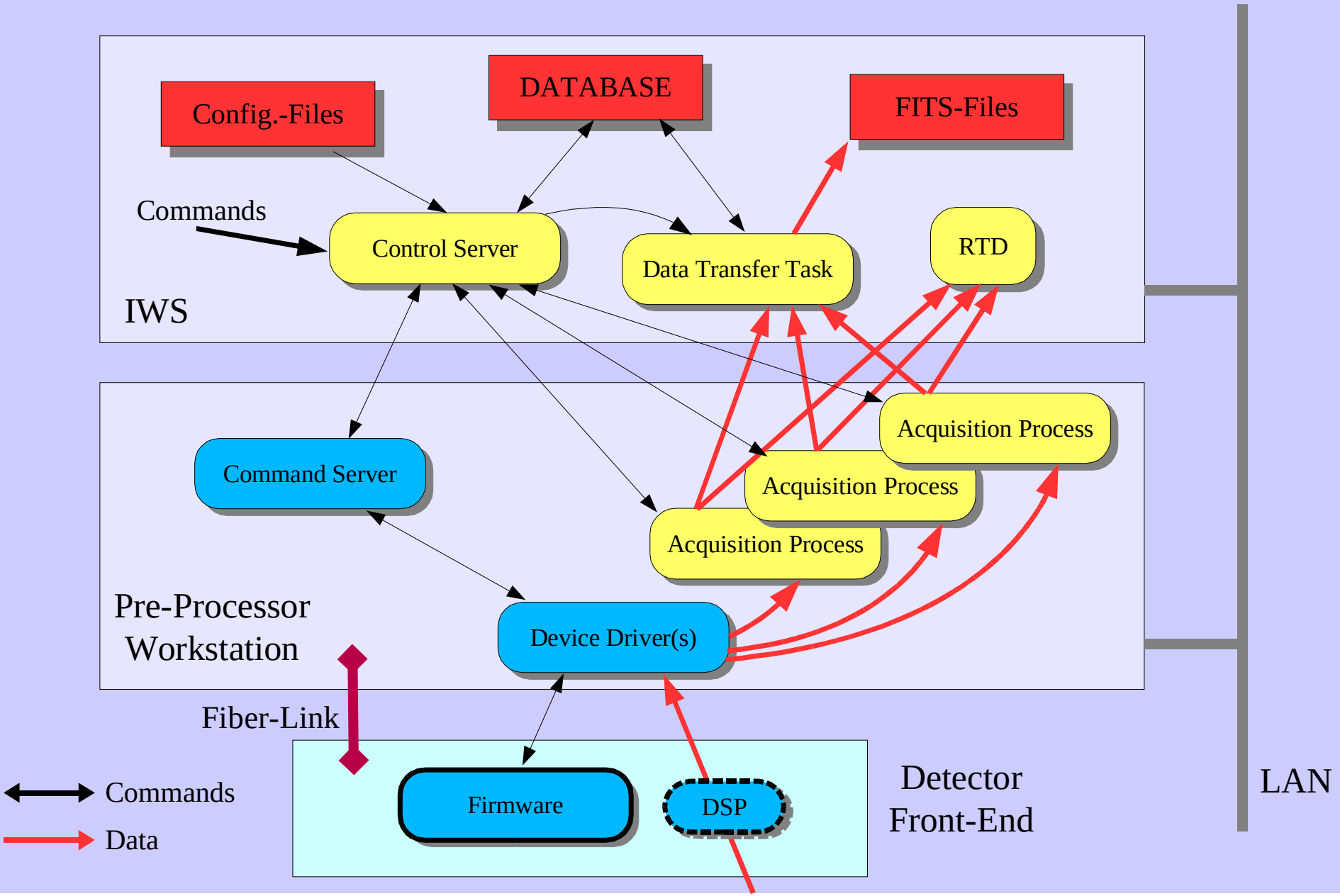
Image Processing – Data Transfer

- Any frame type may be transferred to Instrument Workstation and/or Real-Time-Display (RTD).
- Any data transfer is fully parallel to the processing loop (i.e. while transferring the result, the next result is computed).
- No memory copies required.
- The frames to be transferred are put on an output queue.
- Frames are transferred on request. The request may be FIFO (for science data transfer) or LIFO (for video data transfer). Retransmission should be possible (display several different windows of the same image).
- Only limited by network bandwidth and computing power of the requesting client process.
- Different frame types can be transferred in parallel to different requesting data clients (i.e. RTD shows Raw-Frames while the Averaged-Frame is transferred and stored to a FITS-file).
- Burst Mode (buffer large number of raw data in memory):
  - Parallel output queue structure required to be able to dump data to disk while next data is read.
  - Transfer “overhead” is large, but off-line image-processing can be done on a set of raw-images.

Acquisition Process



DCS Processes



Acquisition Process and CCS

- Data acquisition must be divided into separate tasks to use full Multi-CPU bandwidth. Task architecture is complex and synchronization must be fast (use semaphores, mutexes rather than command messages).
- Priority based “real-time” scheduling is required to grant maximum CPU usage to the data processing.
- Data-flow is continuous and must not imply any memory copy.
- Multi-Threading is the only choice to fulfill these requirements on a UNIX platform (Linux, Solaris) -> there is only one Process Space!
  - One Process Space can only register once to a CCS-environment.
  - It is not possible to run several CCS event handler loops within one Process Space.
  - A central command handling would be needed to distribute CCS-messages internally (commands, data requests, ...). This would only shift the “interface” and would make the process itself much heavier.
- The QSEMU itself does not run with “real-time” priority. In the worst case (full CPU load) it would be impossible to STOP the system.
- With high CPU-load the QSEMU/CCS-Scheduler stacks and message queues would run over.
- At this level we cannot afford to consume resources for just driving a general and complex protocol and inter-process communication. With arrays/mosaics becoming larger/faster/... we are always near to the system limit.
- A dedicated, task-oriented protocol helps to do just the processing job and the data-transfer(s) and nothing else.

Image Processing – Real-Time

- Real-Time OS guarantees response times in the range of microseconds.
- Memory requirements for the real-time control loops are much lower than for pure imaging (might increase in future).
- Current Real-Time OS standard at ESO is VxWorks running on a PowerPC architecture.
  - Maximum Data Rate via 64-Bit VME: 80 MBytes/s.
- Other Real-Time-Computer (RTC) standards are under investigation (to be considered as a black box).
  - Might require special interfaces – to be provided by the Controller or by the RTC?

Current IRACE Implementation

- VxWorks driver for IRACE VME-Interface handles double buffered DMA transfers.
- The driver module provides a higher level task handling (Execute/Kill/Start/Stop/ParameterSetup) to control the application via DCS. This is no requirement for normal operation but saves time for maintenance and development.
- The detector front-end administration is not done via the RTC. An additional host computer (Sun SPARC/Linux PC) is required to provide at least a communication channel.

Control Server

- Handles all commands and configuration files.
- Updates database.
- Starts and controls the Data Transfer Task.
- Starts and controls processes on the pre-processor workstation (acquisition process, command server) .
- Can be customized for specific needs via derived server classes (application specific command callbacks, parameter handling, add-on's for STANDBY/ONLINE commands).
- Standard commands: PING, VERSION, SIMULAT, STOPSIM, STANDBY, ONLINE, OFF, EXIT, SETUP, STATUS, START, PAUSE, CONT, END, STOP, ABORT.
- Additional commands for FRAME selection, Telemetry, Reset of detector front-end, Start/Stop sequencer/acquisition process independently, Enable/Disable CLDC,..., (-> mainly used for maintenance and development).

Data Transfer Task

- Started via Control Server. Mostly transparent to the outside.
- Requests and receives images from the acquisition process(es).
- Assembles images from several acquisition processes.
- Generates FITS-header and stores data to disk (single image file or data cube).
- Data Cubes may contain different frame types (configurable).
- Can be customized for specific needs via derived server classes (application specific command callbacks, database accesses).
- Provide a data-callback for application specific data handling (i.e. store data in different format).
- Provide built-in post-processing features (image rotation, flip, ...).

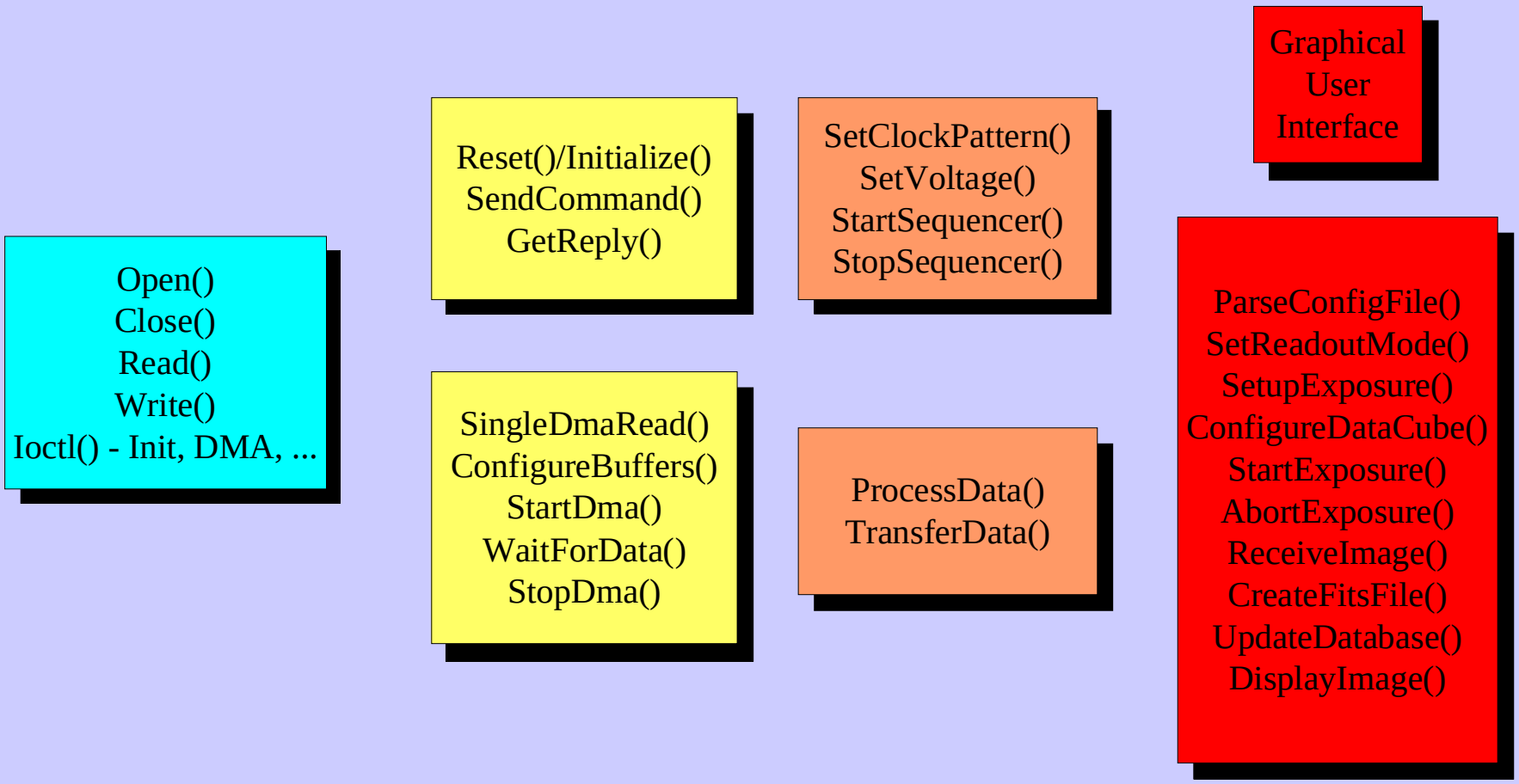
Simulation Mode

- Acquisition-Process simulates DMA data with (configurable) interval timer.
- Command Server simulates detector front-end including firmware.
- Control Server and Data Transfer Task operate in "real" mode.

User Interfaces

- Graphical User Interface (GUI)
  - Keep what the user is used to?
  - Stand-alone operation
  - Provide interface to low level functions
  - Application specific extensions
  - On-line help – should cover main issues of the User Manual
- Configuration Files
  - Backwards compatibility
  - Can be edited with simplest ASCII-Editor
  - Graphical Design Tools on top
- Real Time Display (RTD) for visualization
- Image Processing (IDL, MATLAB, ...) as Plug-In (- at which level?)

Software Layers



What is common?

- Physical interfaces (PCI-Bus board, Fiber-Link)
- Common device driver for standard system calls (open, close, read, write, DMA-ioctl)
- Common protocol on the communication channel
  - Reset/Initialize(), SendCommand(), GetReply(), ...
- Common DMA-interface
  - SingleDmaRead()
  - ConfigureRingBuffers(), StartDma(), WaitForData(), StopDma(), ...
- Next level of abstraction already interferes with application specific features (sequence logic, data handling,...). Backwards compatibility must be considered!
  - SetVoltage(), SetClockPattern()
  - ProcessData(), TransferData()
- At Control-Server Level implementations may diverge...
  - ParseConfigFile(), SetupExposure(), UpdateDataBase(), ConfigureDataCube(), CreateFitsHeader(),...

ASICS

- Use the same physical interface and device driver(s).
  - ASIC specific command-/ data-stream conversion is done in the detector front-end.
  - Acquisition process architecture can be kept – just another Plug-In.
  - Command interface must be adapted for each design:
    - Requires strict modularity.
    - Common parts are assembled in an ASIC interface library / class.
- ASICS are not yet available.
  - Requirements are not yet defined.
  - Requirements may change in future.

User Support

- HW-Development (“run a clock pattern”, ...)
  - May use just parts of the system or just run an acquisition process doing a consistency check.
  - Will not operate the system through BOB/OS just to “start a clock pattern on the sequencer and check it with the oscilloscope”. Startup-overhead is too big in case of frequent system-reboots.
  - Stand-Alone operation on the computer hosting the physical interface(s) is required. Startup from scratch must be fast (< 5 seconds).
  - Interaction to all lowest level functions must be supported and must be fast.
- Detector Development (“read-out a detector”, ...)
  - Always uses the whole system.
  - Operates DCS in stand-alone mode, but with its full functionality.
  - Might use BOB/OS for system integration or maintenance- and test-templates.
- End User (“run a set of exposures”, ...)
  - Operates system only through OS.
  - Only uses a well-defined subset of commands, but still requires flexibility in exposure definitions (frame to be stored/assembled in cubes/...).

IRACE Backwards Compatibility

- More than 400 application files currently operational at instrument level:
  - 88 Acquisition Processes
  - 180 Sequencer Files
  - 86 Default Parameter Setup Files
  - 17 Clock Pattern Files
  - 17 Clock-and Bias-Voltage Files
  - 15 Detector Configuration Files
  - 14 System Configuration Files
- About the same number for detector tests in the IR-Lab.
- Instrument plans relying on the current system go beyond 2006.