

Requirements, Software Architecture & Design

...

Contents

Hardware Overview

Underlying Assumptions & Requirements for
MONSOON Software

Design Philosophy & Software System Architecture

Functional Decomposition, Interfaces and Libraries

Top Level Data Flows

PAN Process Architecture and Coordination

Software Management, Test Plan, Resources and
Schedule

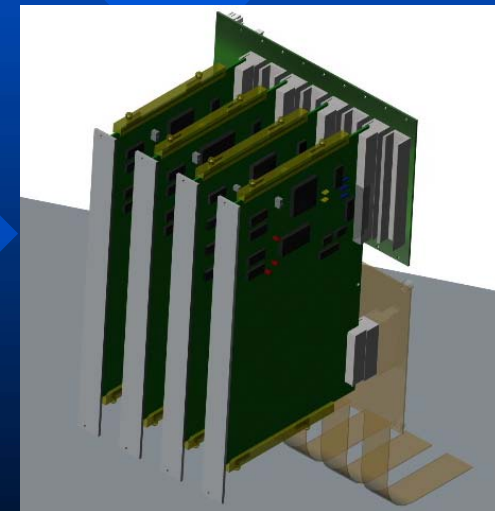
Hardware Overview

MONSOON Hardware Requirements

- Scalable, low-cost, high-performance system.
- Support both IR and OUV devices.
- “Detector-Limited” performance.
- Device independent data acquisition architecture.
- Small modular packaging.
- Low power dissipation.
- Low total cost of ownership.

MONSOON Image Acquisition System

- Scalable multi-channel high-speed Image Acquisition System.
- Scalable at all levels based on cost/performance trade-offs.
- Specifically designed to address the needs of next-generation IR & CCD mosaic systems.
 - ORION (2k x 2k) InSb & HgCdTe development .
 - NEWFIRM (4k x 4k).
 - WYIN QUOTA (8k x 8k) => ODI (32k x 32k).
 - LSST (40k x 40k) and growing....
- Increased performance over existing solutions.
 - With reduced cost, size and power consumption.



MONSOON System Communications

■ 3 Critical Networks.

1. 1 GHz (2.4 GHz) COTS fiber optic network.

- Hi-speed, lo-latency - 50 Mpixel/s SL100, 120 Mpixel/s SL240.
- Handles all communication to DHE.

2. Ethernet.

- Provides power control for DHE for system error recovery.

3. Controller synchronization.

- Key system element, “hard-synchronized” controllers.
- Distributed 40 Mhz master system clock and sync pulse.
- Controlled impedance, skew adjusted LVDS signal distribution.

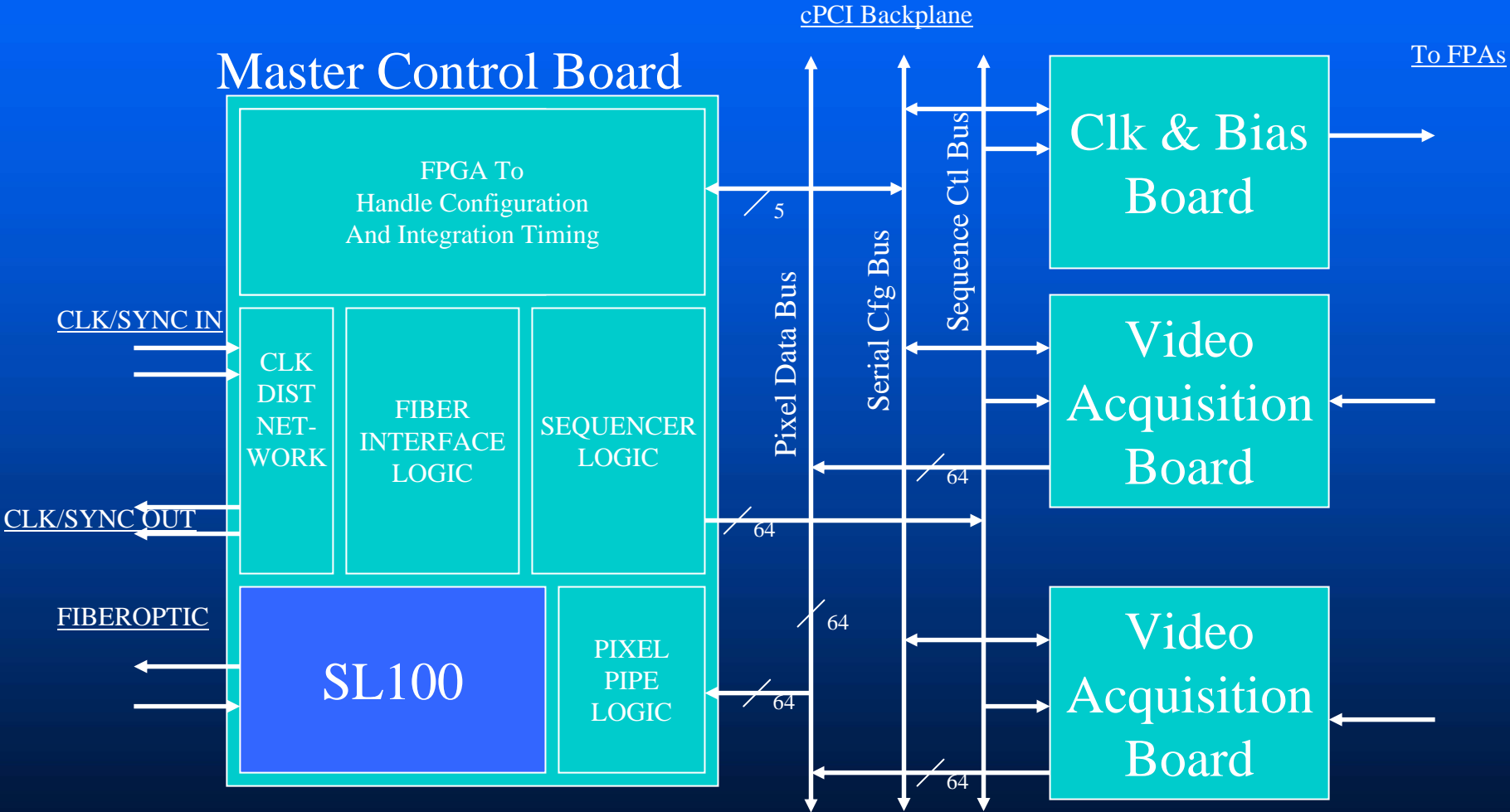
MONSOON ANALOG Performance

- Current dynamic range: $> 60,000:1$
 - 16-bit 1mhz ADC resolution, supporting $S/N > 90\text{db}$
 - Future support for higher resolutions
- Non-linearity: $< 0.1\%$ over entire range
- Read noise: $< 10\%$ contribution to total noise
- Channel to channel cross talk: $< 0.0015\%$ (16-bit resolution)
- Pixel to pixel cross talk: $< 0.01\%$

MONSOON Key Technologies

- **Low-cost “GHz-class” PC’s.**
 - Removes the need for embedded DSPs in system, (PC cost ~ 2.5k).
- **Scalable commercial high-bandwidth fiber optic networks.**
 - Buy not build, use a well-supported commercial product.
 - » Systran FiberExtreme SL100/SL240 .
 - » SL100: 100 Mbyte/s => 50 <Mpix/s , SL240 240 Mbyte/s => 120 Mpix/s.
- **Standard software systems.**
 - Use dependable components with large user base.
 - » Redhat LINUX.
 - » Existing software components or systems or design patterns.
- **State-of-the art analog & mixed signal electronic components.**
 - Increased performance with reduced power, size, and cost .
 - » Allows construction of large channel count systems.

MONSOON DHE Design



MONSOON 3 Board / 3 Bus System

■ 3 Boards

1. Master Control Board (MCB) - Common to all systems
2. Clock & Bias Board (C&B) - Designed to meet FPA needs, 2 or more versions planned (IR & CCD)
3. Acquisition Board - Designed to meet FPA needs, 2 or more versions planned (IR & CCD)

■ 3 Buses (40-66MHz)

1. 64-Bit Pixel Bus - Synchronous transfer of 64-bit pixel data from Acq board to MCB
2. Sequencer Bus - Hi Speed Timing Bus (MCB to Acq & C&B Boards) for all controller timing
3. Serial Configuration Bus - JTAG Serial Configuration Bus to configure & read back Acq/C&B boards

Master Control Board

- Provides all timing & sequencing to system.
 - Provides MONSOON system synchronization.
 - Employs FPGA (Xilinx Virtex) hardware sequencer.
- Provides interface to Systran fiber.
 - Fiber handles all primary cmd/response and pixel data.
- Provides interface to optional embedded Ethernet processor.
 - Ethernet used for “back-door” reset.

Clock & Bias Board Model

- Board may be tailored to FPA & system req's.
- All interface to MONSOON bus through FPGA:
 - » PCI compatible signals - reconfiguration of bus interface signals if needed - room for added functionality & lots of flexibility.
- All clock voltages & bias voltages have read back.
- Bias voltages & clock rails set by Serial Cfg Bus.
 - IR board will support high-speed parallel DACs for some nodes.
- High channel count density on 6U format:
 - Advances in CMOS dacs allow 100's of channels on 6U format.

Acquisition Board Model

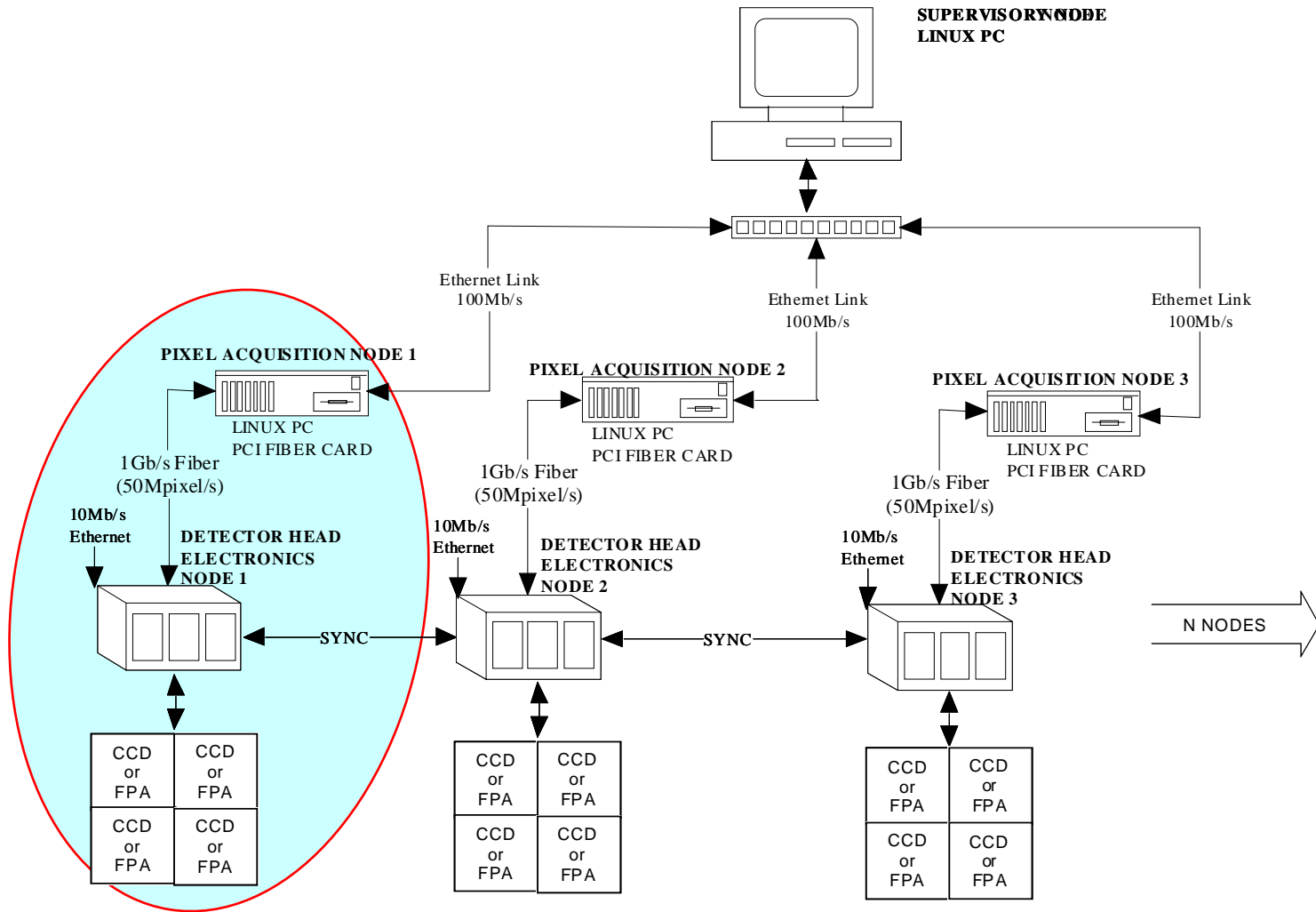
- All interface to MONSOON bus through FPGA:
 - » PCI compatible signals - reconfiguration of bus interface signals if needed - room for added functionality & lots of flexibility.
- High channel counts on 6U format.
 - 36 channel IR board in PCB fabrication.
- Cost & power for 36 1mhz IR channels.
 - 5W – 10W for 36x1MHz IR channels ($< 500\text{mW/MHz/ch}$).
 - $< \$5000$ component cost ($< \$200 / \text{MHz} / \text{ch}$)

PAN-DHE Communications

- ICD 6.1 - MONSOON PAN to DHE Command/Response Interface.
- Four Commands from PAN to DHE.
 - Read memory, Write Memory, AsyncResponse, startExposure.
- DHE echoes every word sent to it...
 - except startExposure command.
- DHE sends a AsyncMsg on hardware reset or power up.
 - This message can have information on cause embedded in it.
- Memory Addresses allow for:
 - 10 board select Bits - expandable to 14.
 - 65536 memory locations on each board.
 - » 0 - 31 are considered 32 bits wide.
 - » 32-65535 are 16 bits wide.

Underlying Assumptions
&
Requirements for MONSOON Software

MONSOON Pixel Server



Underlying Assumptions

- PCI bus system with GIGA-Hz class CPU(s).
- PCI system in console-less operation.
- Will use an Open Source well-known OS. (Linux).
- Hardware and Software Systems to be Open Source.
- Command & Data communication by ethernet.
- DHE Communication over COTS Fiber (3 Km).
- Multi-tiered security policy.
 - Connection location/source (firewalls, name/address).
 - Privileged system (password security).
- Remote (ethernet) power control on DHE.

Requirements for MONSOON Software

Science Generated Requirements
Detector Development Requirements
System Generated Requirements
Software Development Requirements

Science Software Requirements

- 1. Support detector safe operations.**
2. Support detector limited performance.
3. Support both IR and OUV detector systems.
4. Extensible to **Very Large Focal Planes**.
5. Able to treat mosaics as single focal plane.
6. Support science observation by ‘named’ modes.

Science Software Requirements (cont)

7. Provide for efficient science operation support.
8. Support high observing efficiency.
9. Support existing and new observing paradigms.
10. Provide ROI support for readout speed up.
11. Provide ROI support for data compression.
12. Support “technical” imaging. (Guiders, etc.).

Science Software Requirements (cont)

13. Array configuration by ‘standard parameter set’.
14. Parameter sets determined in the Detector Lab.
15. Provide limited tuning of detector performance.
16. Provide limited ‘on-the-fly’ reconfiguration.
17. Allow addition of new processing algorithms.
18. Provide for an instrument calibration mode.

Detector Development Requirements

- Support detector characterization operations.
 - Low-level control of detector parameters.
 - Automated control of characterization process.
 - New data reduction algorithm development.
- Support detector research and development.
 - Incorporation of new detector types.
 - Development of new operation modes.
- Support hardware development & debugging.

System Software Requirements

22. Support efficient boot-up and initialization.
23. Start-up & initialisation without intervention.
24. Support system operations logging.
25. Support connection security.
26. Support convenient error detection/recovery.
27. Support remote diagnosis, debug & operation.
28. System to include simulation capabilities.

System Software Requirements (cont)

29. Help available for commands/parameters.
30. System layered to allow maximum reuse.
31. Pixel data processing chain re-configurable.
32. Support efficient configuration for new systems.
33. Configurable w/o re-compile of base code.
34. Features added without rebuilding system.
35. Software to use well-defined interfaces.

System Software Requirements (cont)

36. Interfaces documented and published.
37. Use GPX Interface to outside world.
38. Documentation maintained with code base.
39. Documentation in standardized format.
40. Support “Package-like” installation.
41. Support verification & removal of versions.
42. Develop source code maintenance manual.

Development Software Requirements

- 43. All Software to be Open Source.
- 44. Use widely available software technologies.
- 45. Use free tools to the greatest extent possible.
- 46. Support multi-site distributed development.
- 47. Use source code version control (CVS).
- 48. Testing and verification built into development.



Design Philosophy
&
Software System Architecture

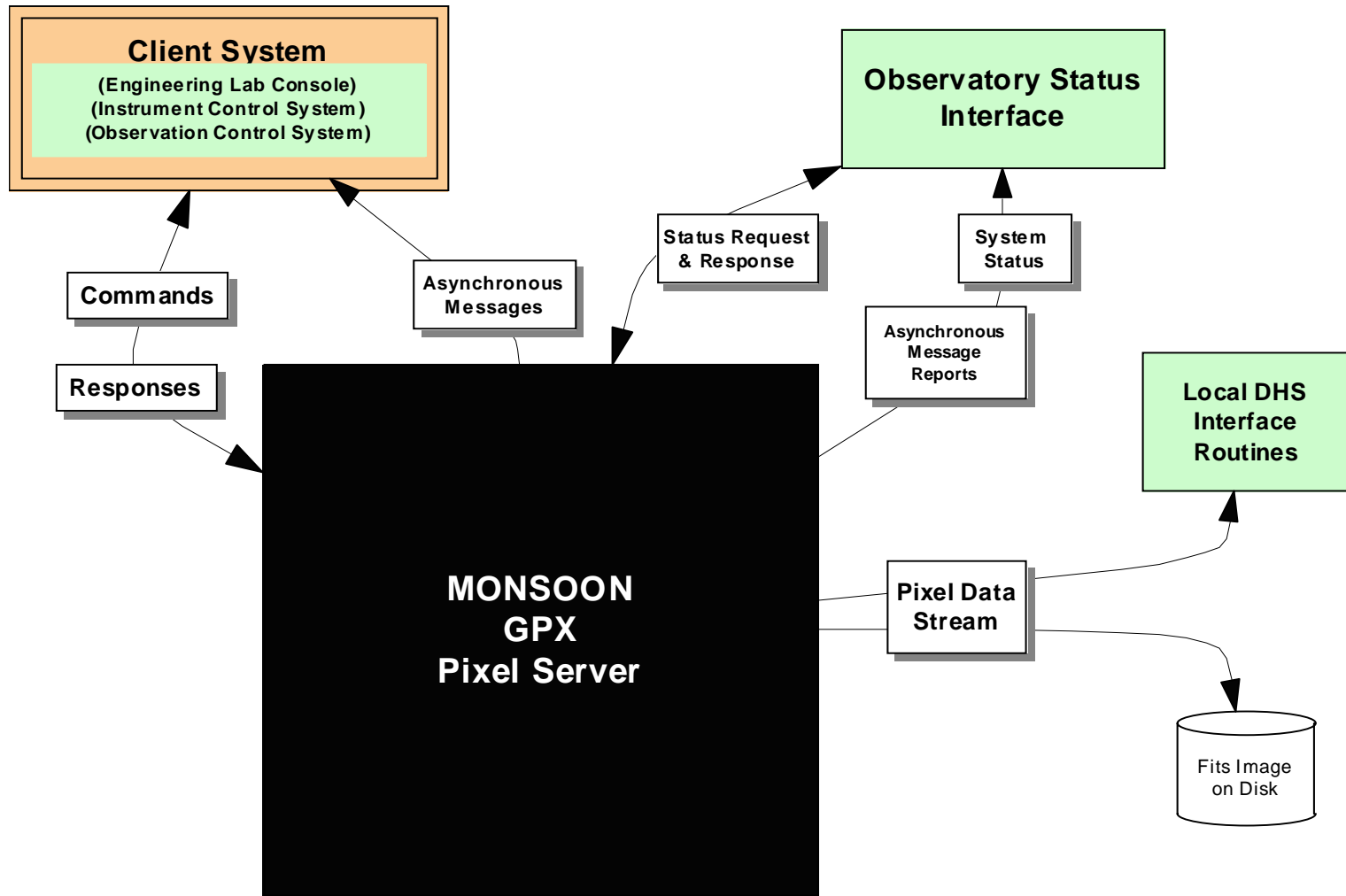
Design Philosophy

- Make maximum use of facilities provided by the OS.
- Isolate & limit scope of software functions.
- Use libraries to provide ease of use & modification.
- Layer functionality in libraries for ease of change.
 - Separate generic and hardware specific layers.
 - Insure isolation between layers.
- Use processes to perform identifiable tasks.

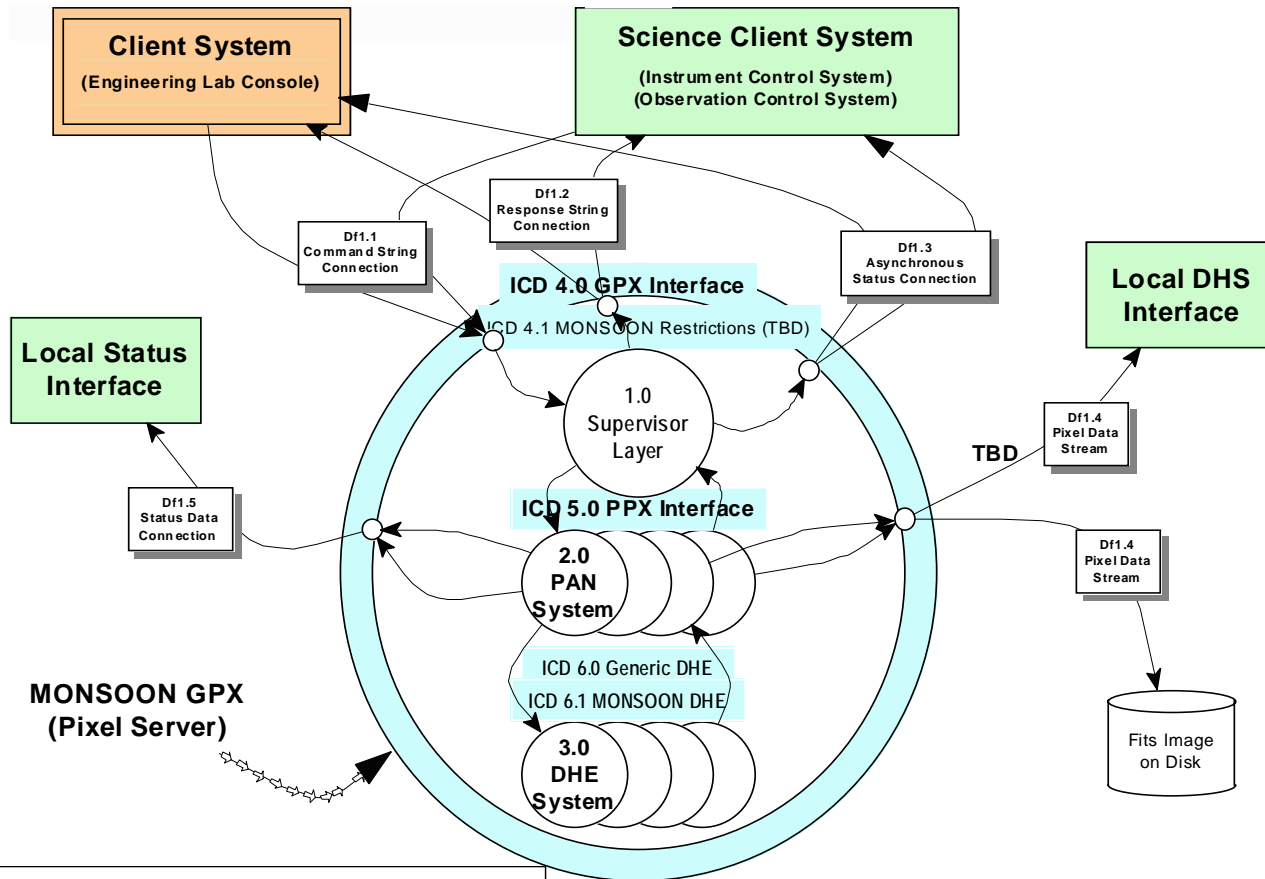
Design Philosophy (cont)

- Build-in monitoring of system processes.
- Allow kill/restart of rogue processes.
- User interface concerns left to client systems.
 - MSL and PAN are software program interfaces.
- Idiosyncrasies left to observatory staffers.
 - Unique DHS requirements.
 - Observatory specific systems (EPICS, WIYN router, etc).
 - Unique GUI tools.

GPX Data Flows



MONSOON Level 0 Context



Three Layer System Architecture

1. MONSOON Supervisor (MSL) Layer.
2. Pixel Acquisition Node (PAN) Layer.
3. Detector Head Electronics (DHE) Layer.

Three Layer System Architecture (cont)

1. MONSOON Supervisor Layer. (MSL).

- A command/control Layer, (no pixel data).
- Provides GPX interface to clients.
- Provide single point contact to system.
- Provides client access security.
- Provides data transfer control.
- Provides multiple client connections.
- Provides error monitoring & recovery.
- Handles Command/response to/from multiple PANs.
- Understands observing 'modes'.
- May run remotely or on a PAN.

Three Layer System Architecture (cont)

2. Pixel Acquisition Node (PAN) Layer.

- No knowledge of other PAN-DHE pairs.
- Provides PPX interface to MSL or users.
- Provides run-time configuration of PAN/DHE.
- Provides first level data archiving.
- Provides multiple image processing ‘modes’.
 - » Fowler Sampling, coadds, MSR techniques, OT imaging.
- Provides parameter verification/control/help.
- Deals with IR/OUV/etc. differences.

Three Layer System Architecture (cont)

2. Pixel Acquisition Node (PAN) Layer (cont).

- Handles single exposure sequencing.
- Handles raw data pre-processing.
- Provides interface to DHE hardware.
- Provides sequence configuration/download.
- Provides PAN error monitoring/reporting.
- Provides PAN process error recovery.
- Provides support for ‘speed ROI’.
- Provides support for ‘compression ROI’s’.

Three Layer System Architecture (cont)

3. Detector Head Electronics (DHE) Layer.

- Handles array hardware control.
 - » Voltage levels, sequencing, monitoring.
- Handles integration timing.
- Handles detector readout sequencing.
- Handles digital averaging.
- Handles shutter control.
- Can handle array temperature control.
- Board Self Identification and Version tracking.

In Current DHE implementation this is all in
FPGA firmware

Functional Decomposition, Interfaces and Libraries

Supervisor Layer Functions

- System start-up and initialization. (multiple PAN's).
- Network/client connection control/security.
- Multiple connection accept & Setup.
- Connection error detection and recovery.
- Primary client communications interface (GPX).
- Command distribution to multiple pans.
- Response gathering from multiple PANs.
- Data transfer control (organizing the transfer).
- Configuration/Mode name->database translation.

PAN (Pixel Acquisition Node) Functions

- Connection Control & Security (MSL & Client).
- Secondary communications interface (PPX).
- PAN/DHE start-up and initialization.
- Command/parameter verification.
- Command/parameter setting security.
- Parameter name/address translation.
- Configuration security/control.

PAN Functions (cont)

- Mid-level exposure control (multiple identical images).
- Pixel data capture.
- Image data pre-processing.
- Immediate image archiving (FITS image on disk).
- DHE interface and control.
- DHS notification of image availability.
- Status Tracking, Notification, Logging.

PAN Functions (cont)

- Operations and error logging.
- Data capture error recovery.
- Diagnosis and debug support.
- Self-test support.
- Operation verification.

DHE (Detector Head Electronics) Functions

- Low-level hardware control.
- Detector timing pattern control.
- Integration timing (if master node).
- Image data conversion and transmission.
- Hardware housekeeping & status response.
- Detector protection.
 - Bias power control, hardware test facilities, ESD Protection.
- Error Detection and Recovery.
 - Power glitches/outages, Electronic component failures.

**In Current DHE implementation this is all in
FPGA firmware**

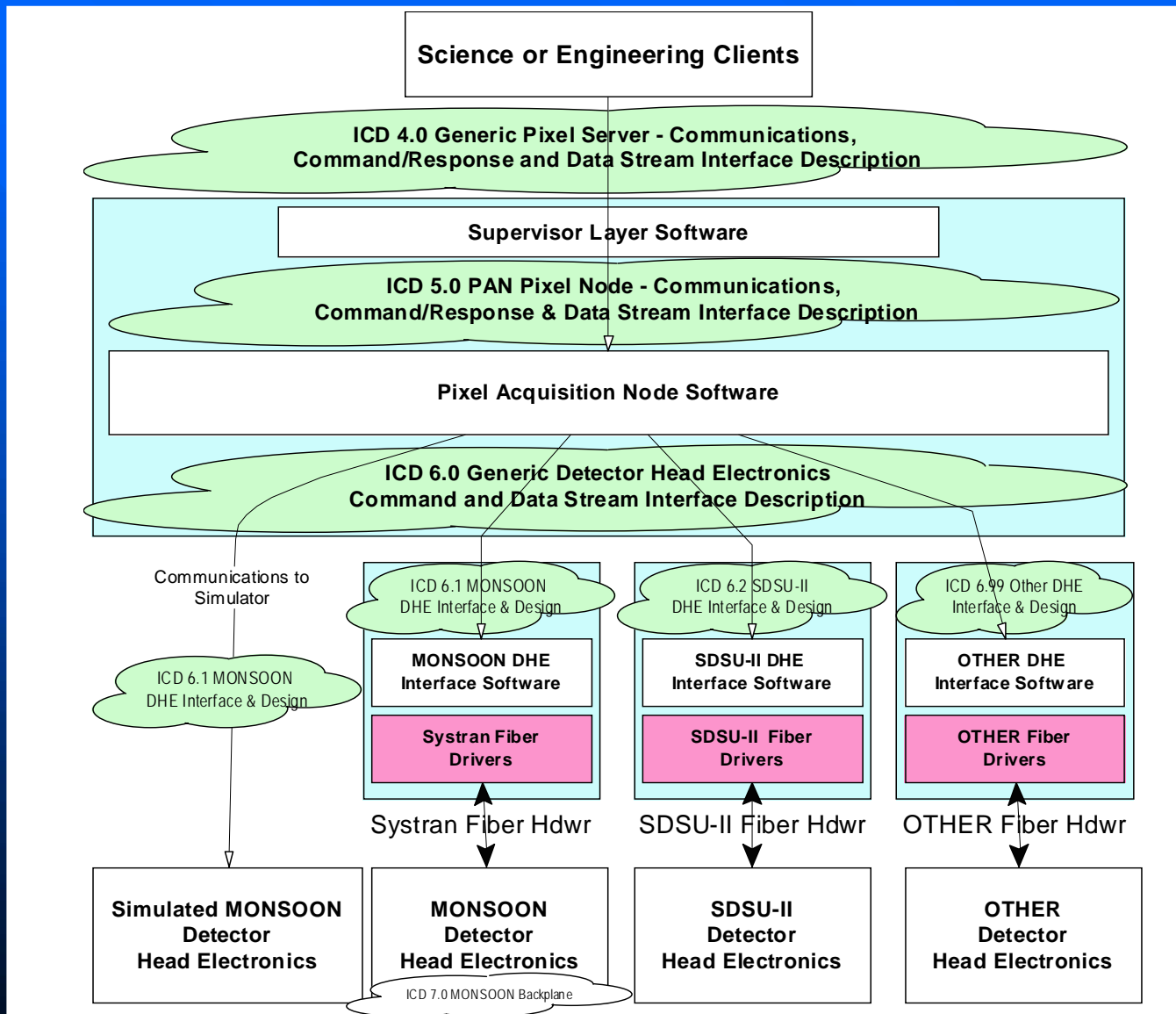
Interface Definitions

- ❖ Client System to Generic Pixel Server.
ICD 4.0 Generic Pixel Server - Communications, Command/Response and Data Stream Interface Description. (*GPX*).
- MONSOON Restrictions on Science Client Access.
ICD 4.1 MONSOON Command and Parameter Restriction Lists.
- ❖ Supervisor Layer to Pixel Acquisition Node.
ICD 5.0 Generic Pixel Acquisition Node – Communications, Command & Response Description. (*PPX*).
- ❖ PAN to Generic DHE (Detector Controller).
ICD 6.0 Generic Detector Head Electronics - Command and Data Stream Interface Description. (*A Command Interface*).
- ❖ MONSOON PAN to MONSOON DHE.
NICD 6.1 MONSOON Detector Head Electronics - Command and Data Stream Interface Description. (*Hardware/Software Interface Details*).

❖ Published interface

■ TBD

Interfaces and Software Layering



MONSOON Software Review

Questions Session

Break

Southern Lunch

Monsoon Layers and ICD's

MONSOON GPX Image Acquisition system ICD and Software Layer relationship

Monsoon Supervisory Layer	(ICD 4.0 GPX)
Pixel Acquisition Node	
Monsoon PAN Interface Layer	(ICD 5.0 PPX)
Monsoon DHE Utility Library (libDheUtil)	(ICD 6.0 Generic DHE)
Monsoon DHE Hardware Library (libmonsoon)	(ICD 6.1 MONSOON DHE)
Monsoon Com Utility Library (libComUtil)	(Com Util Library API)
Monsoon Com Hardware Library (libsystran)	(Com Hdw Library API)
Systran SL240 Drivers (libfxslapi)	(FXSL Library API)
Systran SL240 Hardware	
Monsoon DHE (Detector Head Electronics)	

Not started

Processes in Coding

In coding

Complete (in use)

Complete (in use)

Complete (in use)

Complete (in use)

COTS (Complete)

COTS Hardware

NOAO/CTIO Hdw

MONSOON Libraries

Library Structure & Architecture

General Functionality Libraries

Generic Interface Libraries

Hardware Specific Libraries

Library Structure & Architecture

- Libraries divided into classes of function.
- Call sequence similar for all functions.
- Functions provide inherited status value.
- Functions provide text status string.
- Libraries provide user selectable debug levels.
- Simulation mode included in:
 - Hardware specific libraries.
 - Generic interface libraries.

Library Structure & Architecture

- Individual test program built for each library.
- Library API document provided with libraries.
- API documentation extracted from source code.
- Generic Makefile.
 - Static and shared library versions built.

General Functionality Libraries

- **libavUtil** - Attribute-Value pair search, add & modify routines.
- **libcliUtil**- command line interface; parse, search & help.
- **libqueUtil** - queue functions; new, add, remove, full, empty.
- **libsemUtil** - semaphore functions; new, init, give, take, release...
- **libshmUtil** - shared memory functions; attach, detach...
- **libsockUtil** - socket functions; new, listen, accept, read, write...
- **libmiscUtil** - a set of routines used by many pan processes.

Generic Interface Libraries

- **libppxUtil** - the PAN pixel server interface routines. (ICD 5.0).
 - Pan implementation of simplified ICD 4.0 (GPX).
 - Converts gpxXXX->ppxXXX.
 - Single Attribute-Value pair set/read.
 - Low level Set/Read implemented for engineering consoles.
- **libdheUtil** - generic DHE interface routines (ICD 6.0).
 - Open, Close, Read, Write, IOCTL.
 - readValue, writeValue, readDetector, resetDHE,
 - abortReadout, dhePowerCntl, shutterCntl, biasPwrCntl,
 - startExp, pauseExp, stopExp, abortExp, resumeExp,
 - readValArray, writeValArray, loadWaveform,
 - asyncResponse,
 - testDataLink, testClockDrvr, testDCBiasSup,

Generic Interface Libraries

- **libcomUtil** - generic communications link routines.
 - Modeled on POSIX I/O library facilities.
 - Open, Close, Read, Write, IOCTL ...
- **libpanUtil** - shared memory setup & init used by PAN processes .
 - Shared Memory Allocate, Release,
 - Shared Memory Initialization.
 - Image data buffer setup and Initialization.
 - Get/Set Attribute by name.

Hardware Specific Libraries

- libdetector.
 - Implements routines unique to a detector/instrument.
 - Implements OUV, IR, Guider, etc. differences.
 - Integration Time Calculations, ROI checking & setup,
 - Unique Array Initialization & Setup requirements.
 - OTI details, charge/image shift commands, etc.
 - User Function name to 'C' function translation provided.
 - » Same structure as runtime command configuration.
 - » Translation compiled into library.
 - Shared Library loaded at run time.
 - » One Library for each unique detector/instrument combination.
 - » Library loaded from detector/instrument directory.

Hardware Specific Libraries

■ libsystran.

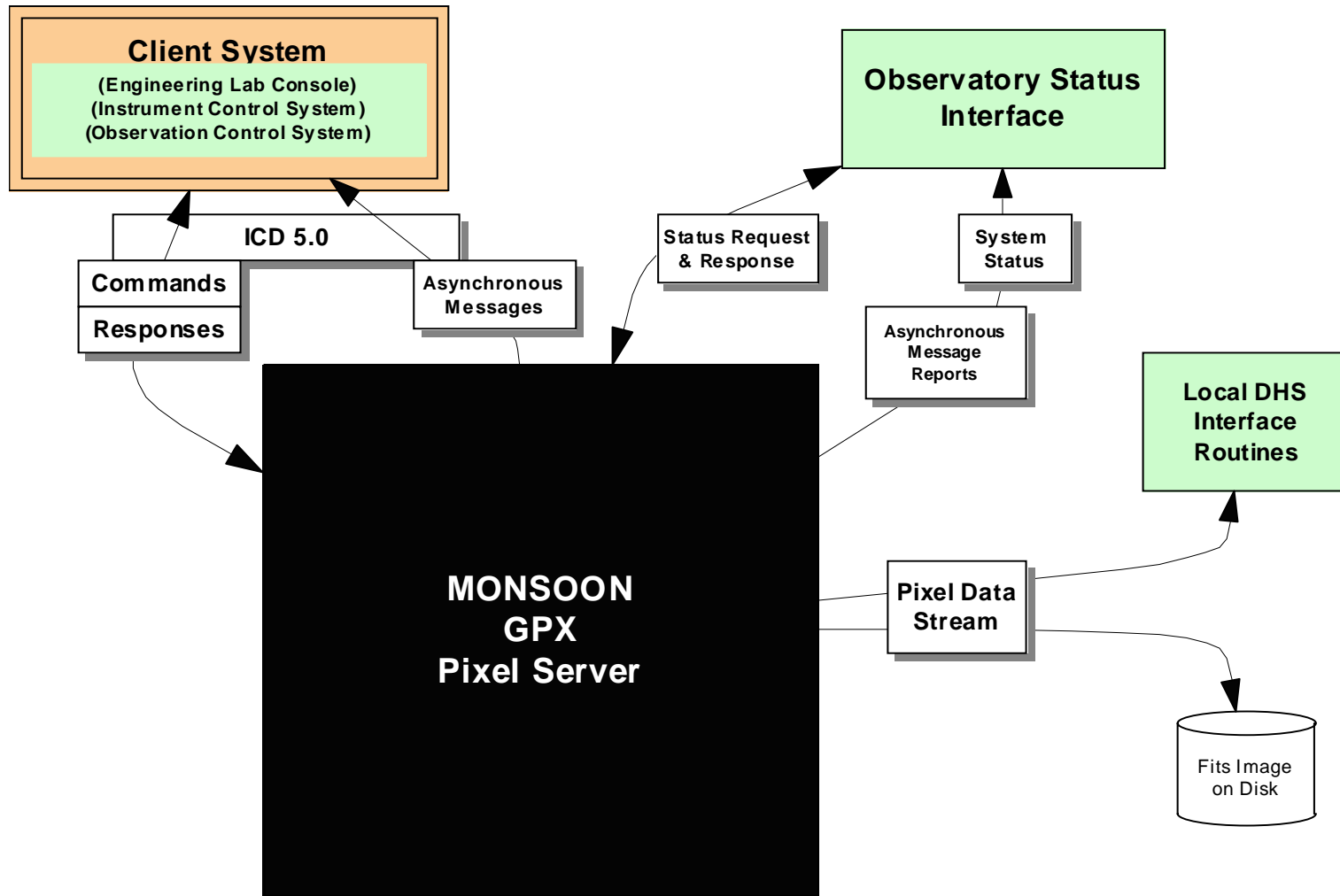
- Implements the SL240 specific interface to the com link.
- Open, Close, Configure hardware link & COTS software.
- Systran hardware functions, read, write, IOCTL functions.

■ libmonsoon.

- Implements the MONSOON DHE access routines (ICD6.1).
- Open, Close, Reset, Read, Write, IOCTL, etc.
- asyncResponse, Test routines, error detection/recovery.
- MONSOON DHE Exposure control commands.
- MONSOON DHE Data Capture Routines.
- Specific command routines implementing ICD 6.0.

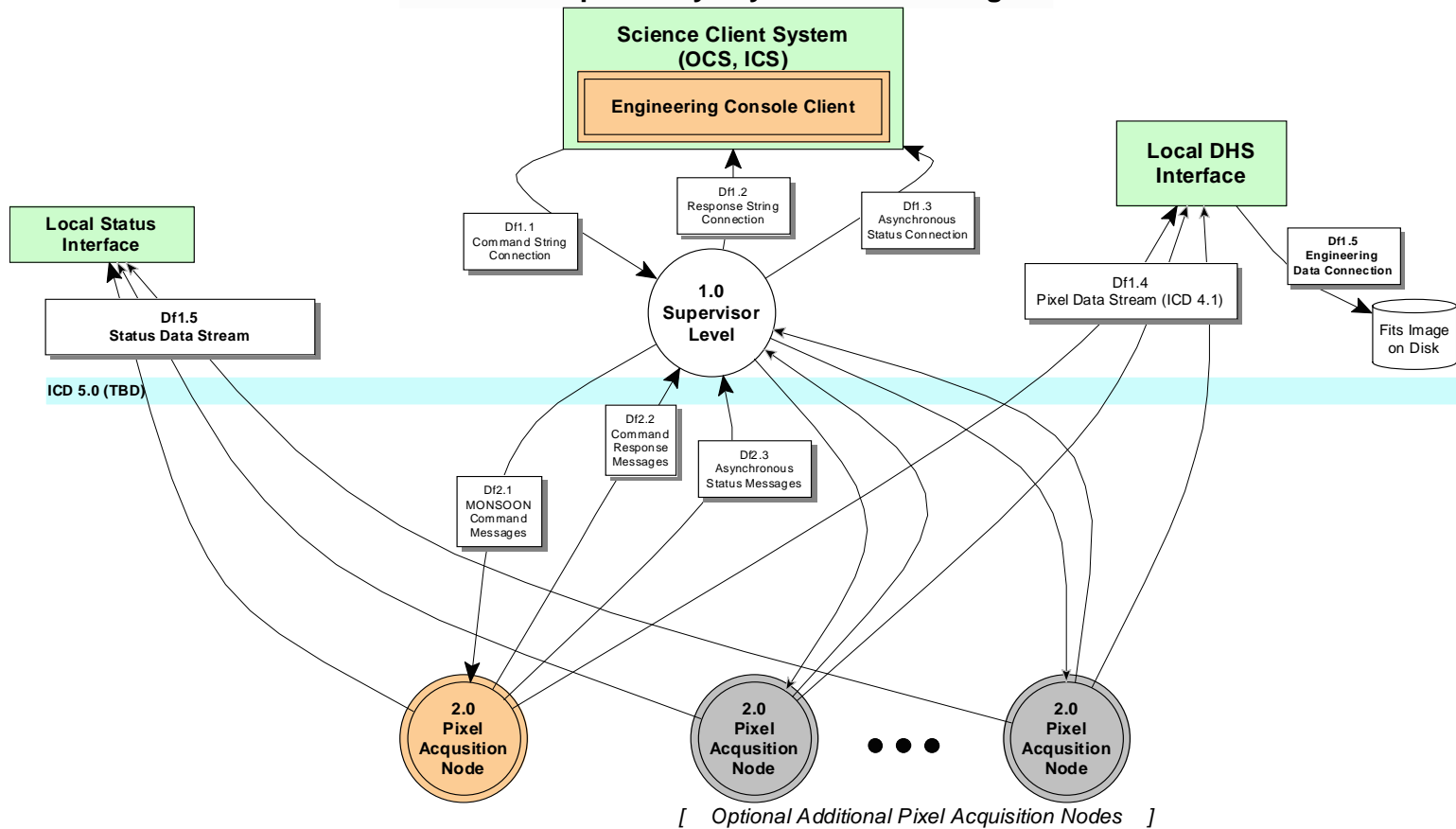
Top Level Data Flows

GPX Data Flows

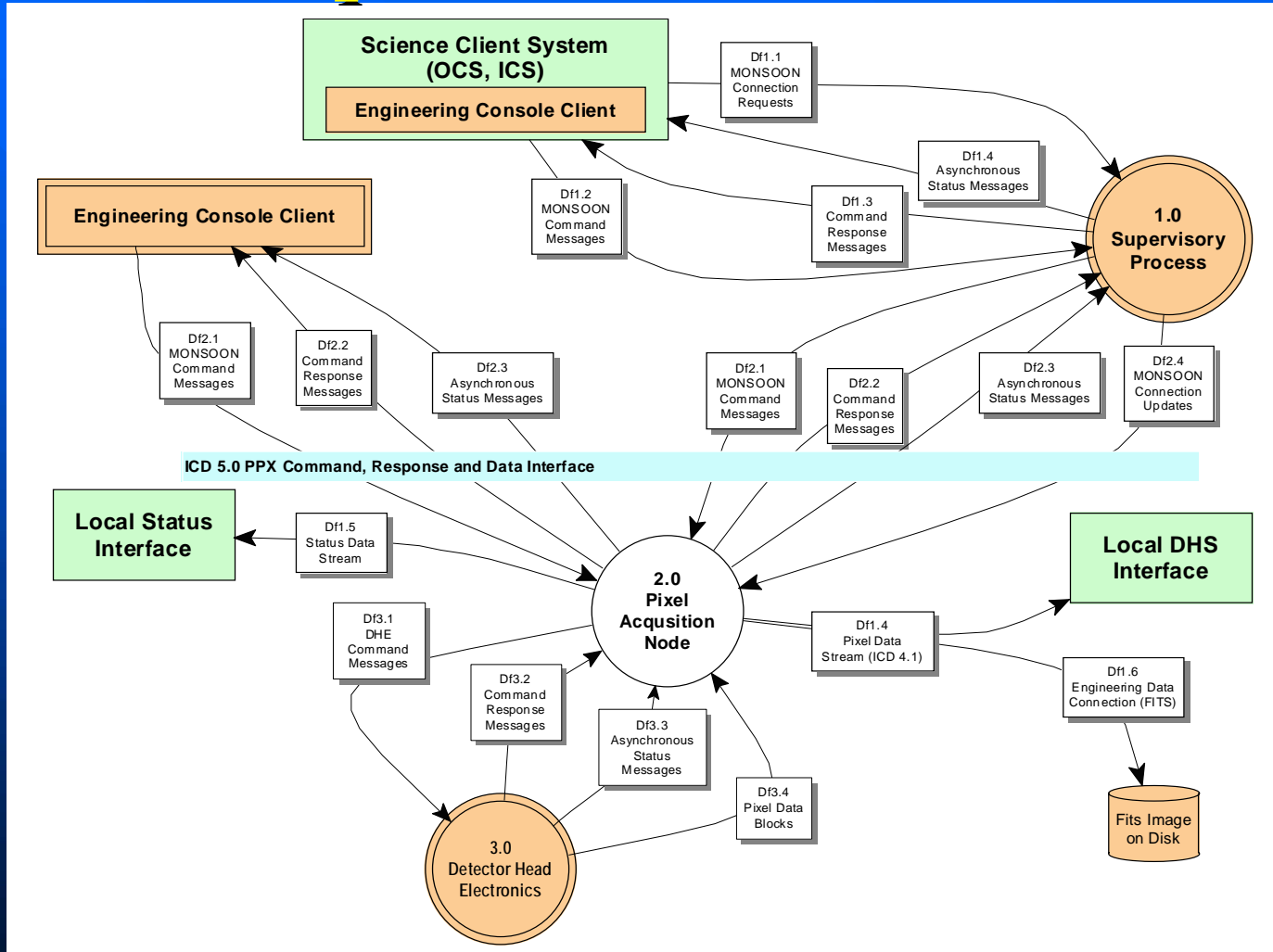


Supervisory Layer DFD (Level 0)

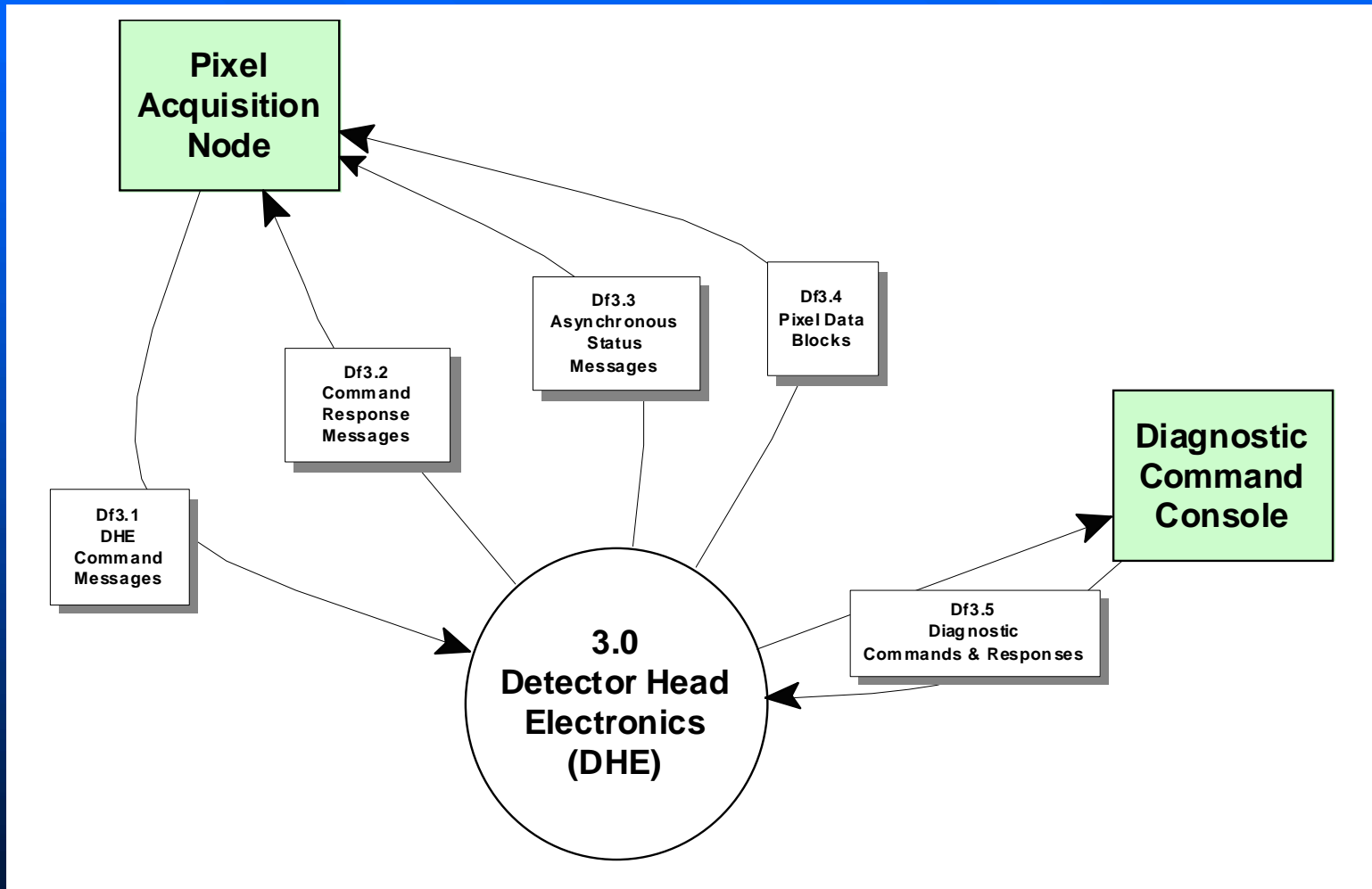
Monsoon Supervisory Layer Data Flow Diagram



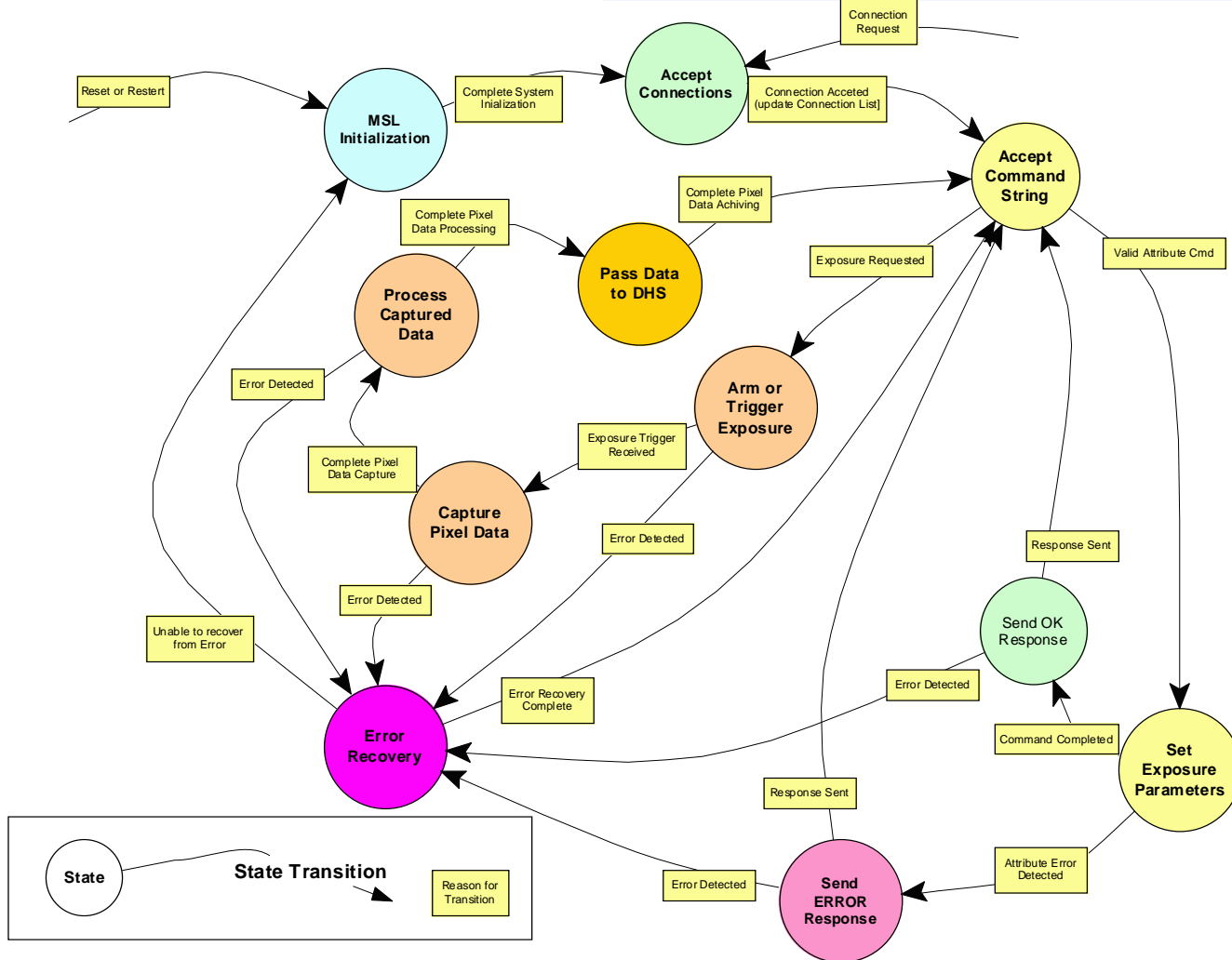
Pixel Acquisition Node DFD (Level 0)



Detector Head Electronics DFD (Level 0)

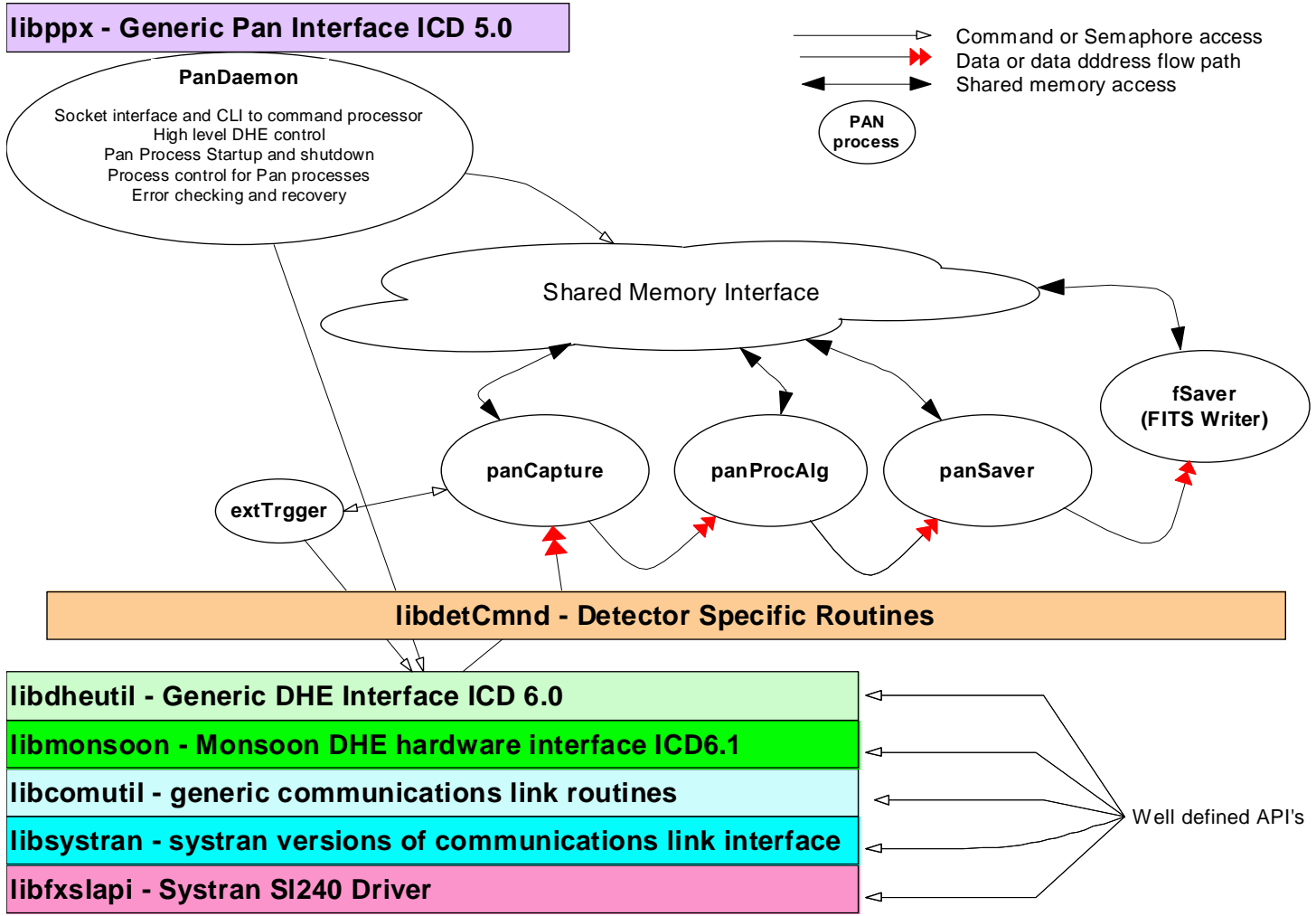


System State Diagram

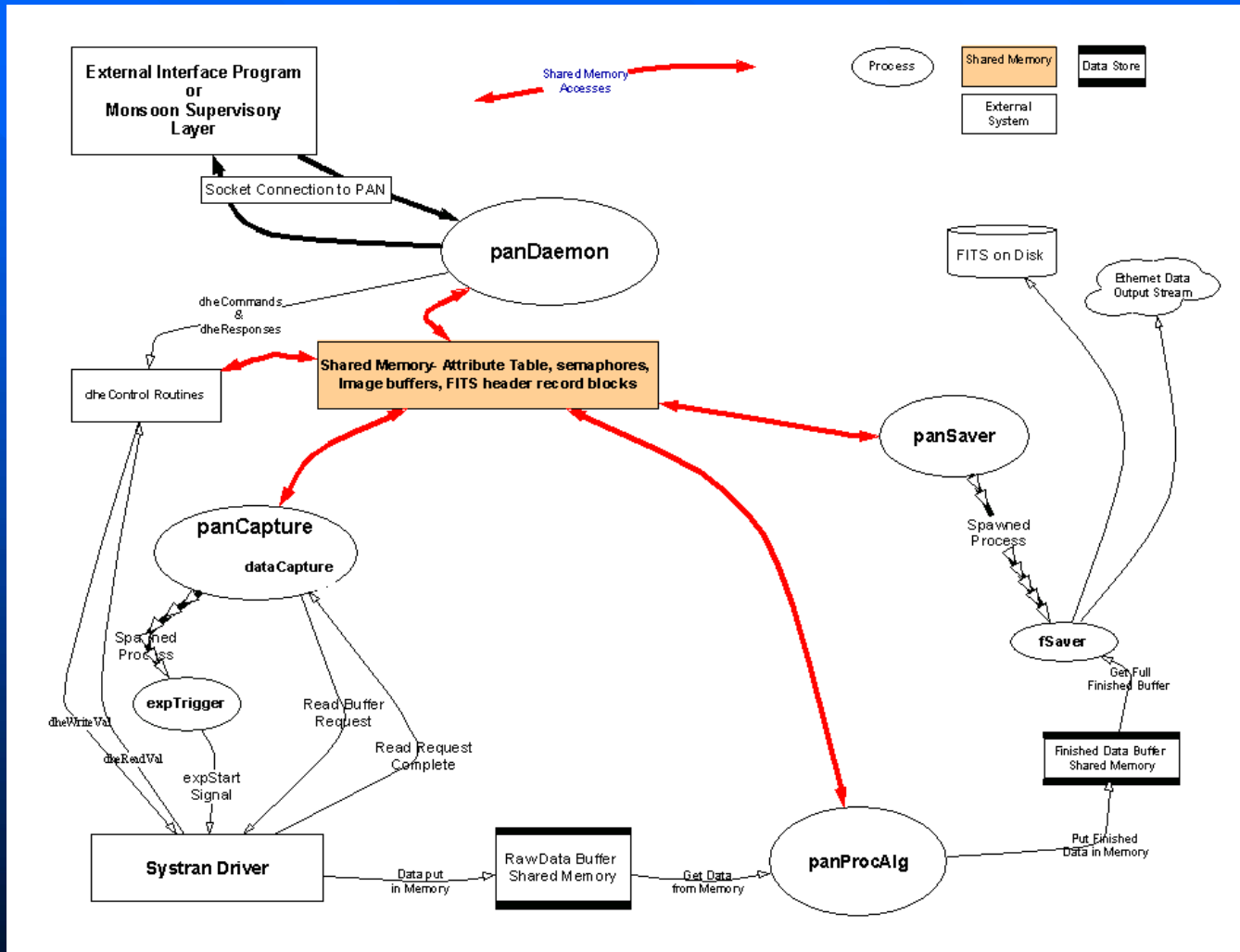


PAN Process Architecture & Coordination

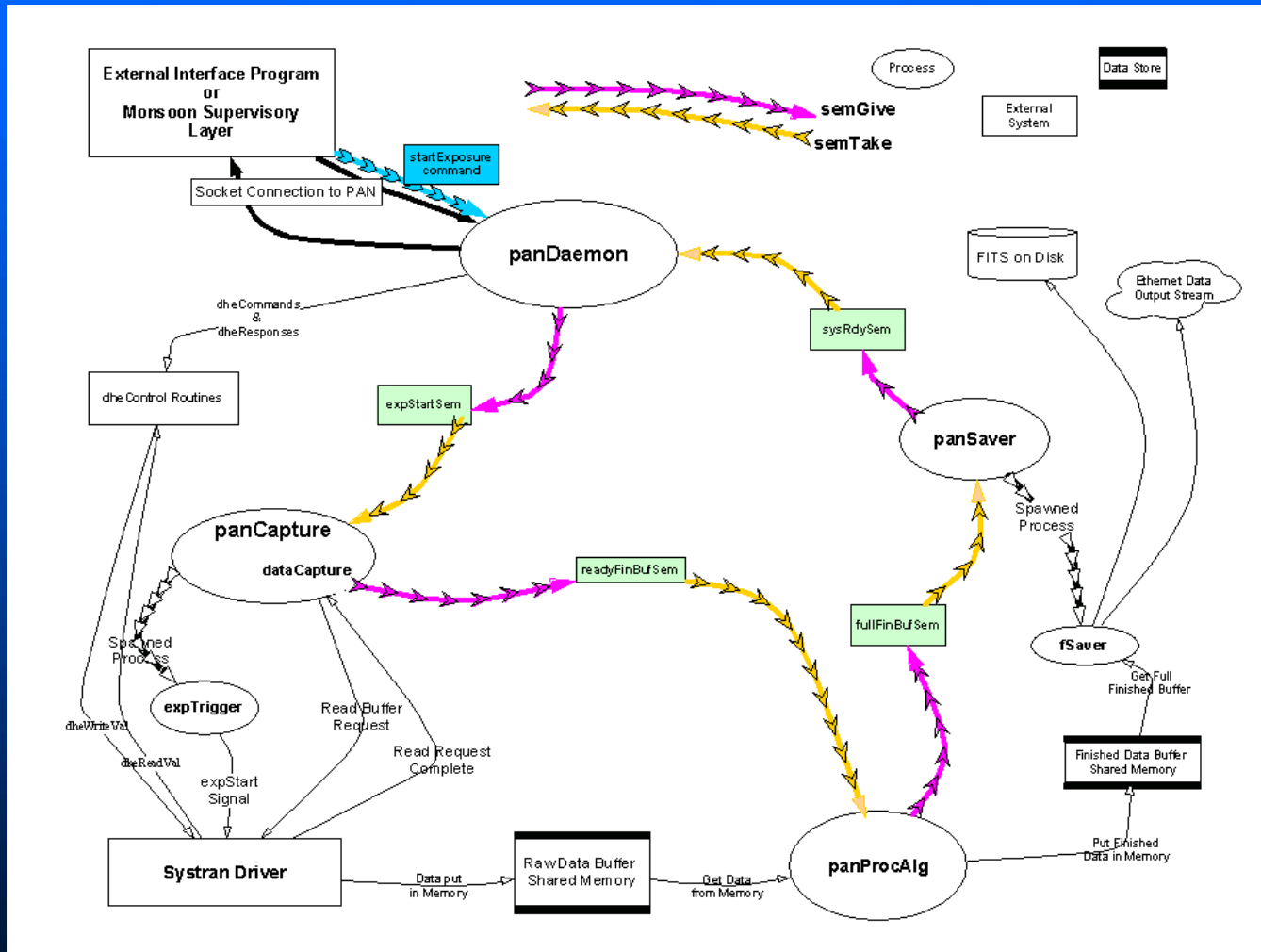
PAN Layered Process Architecture



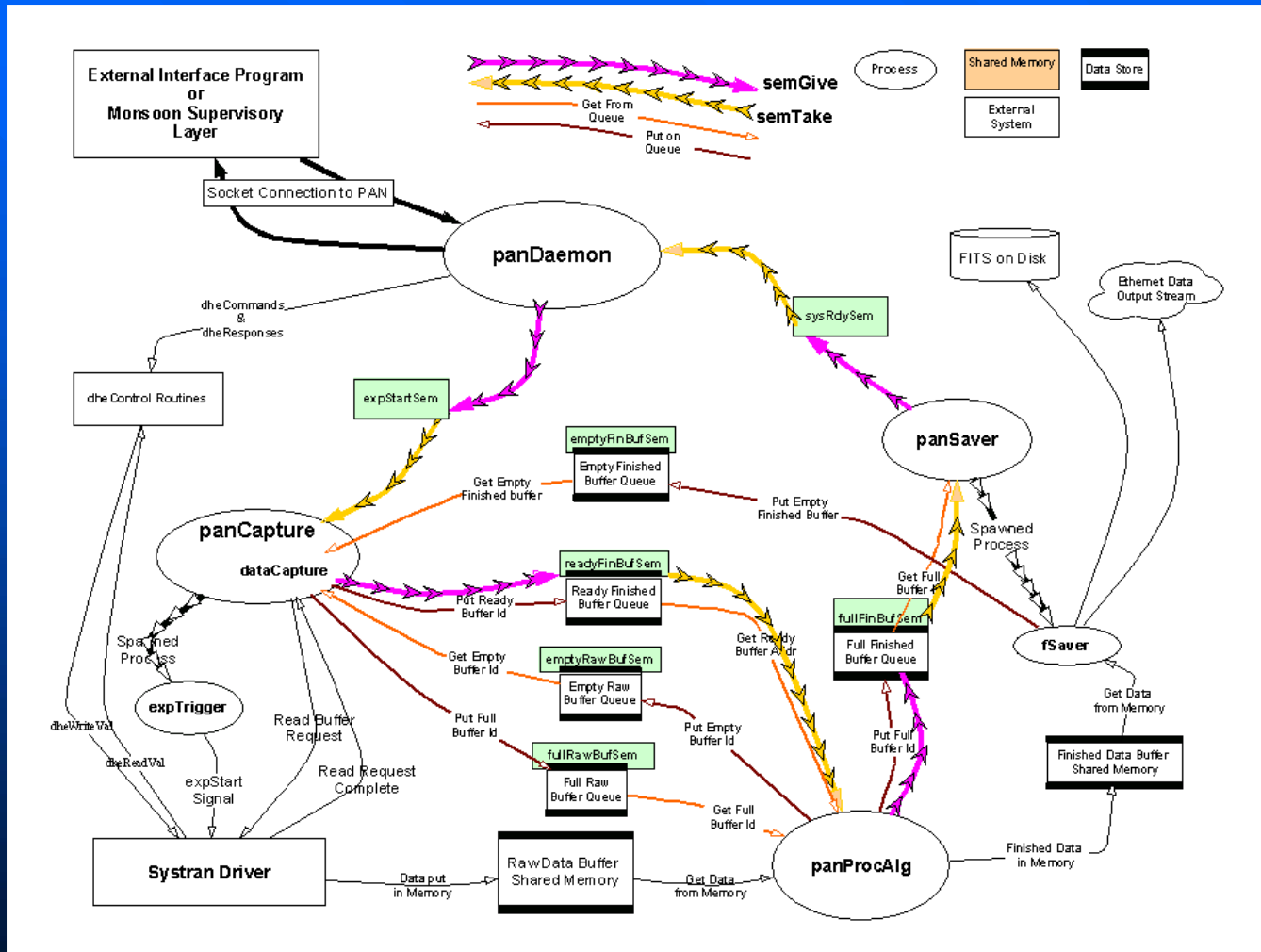
PAN Detailed Process Interconnects



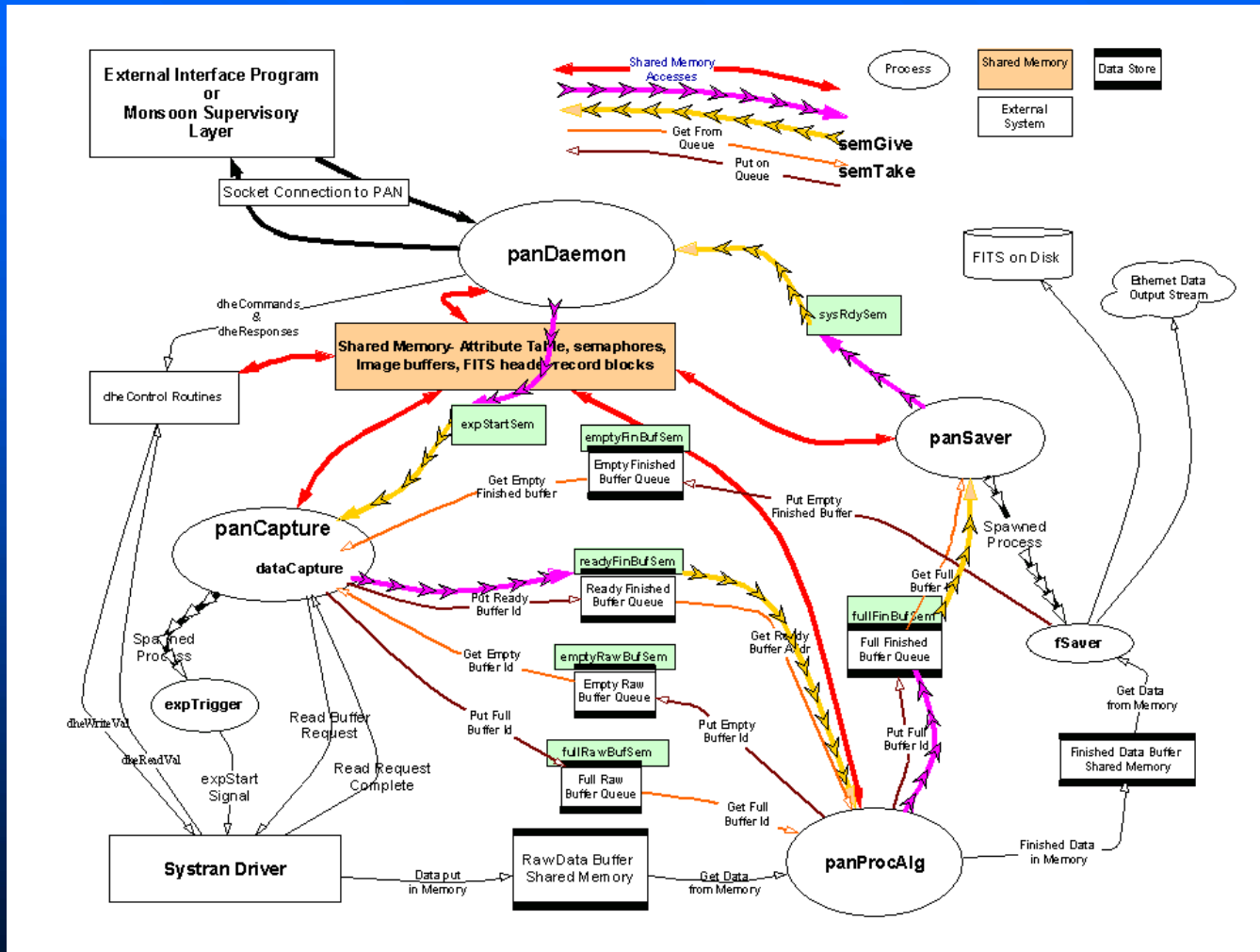
PAN Detailed Process Interconnects



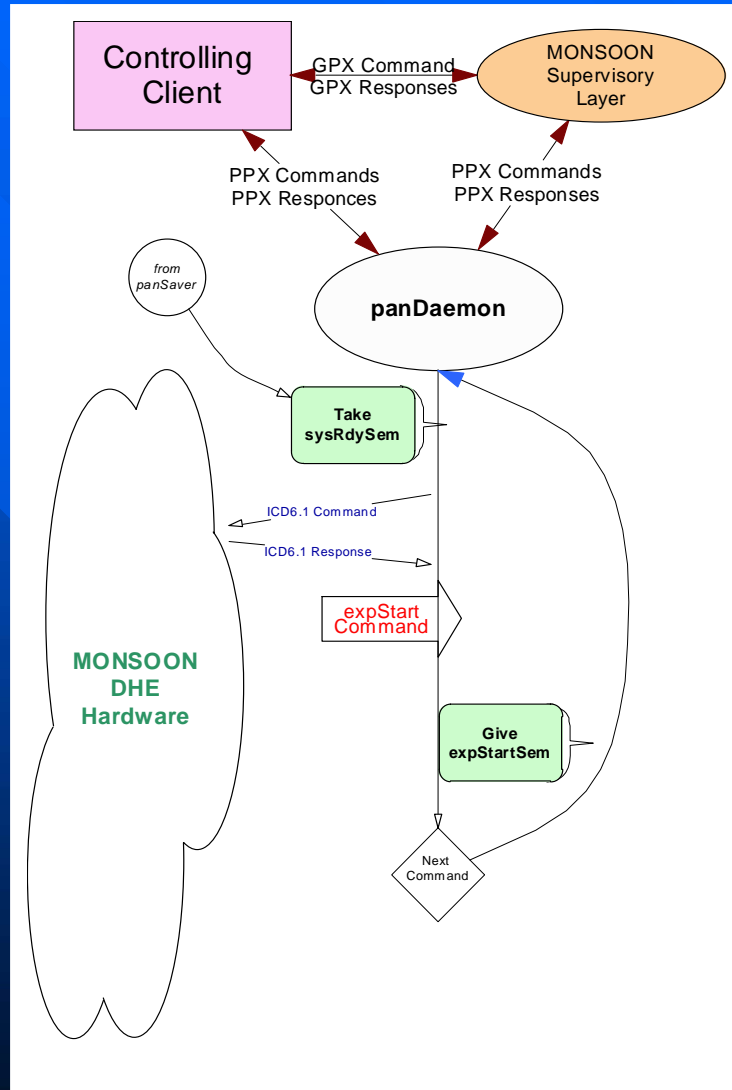
PAN Detailed Process Interconnects



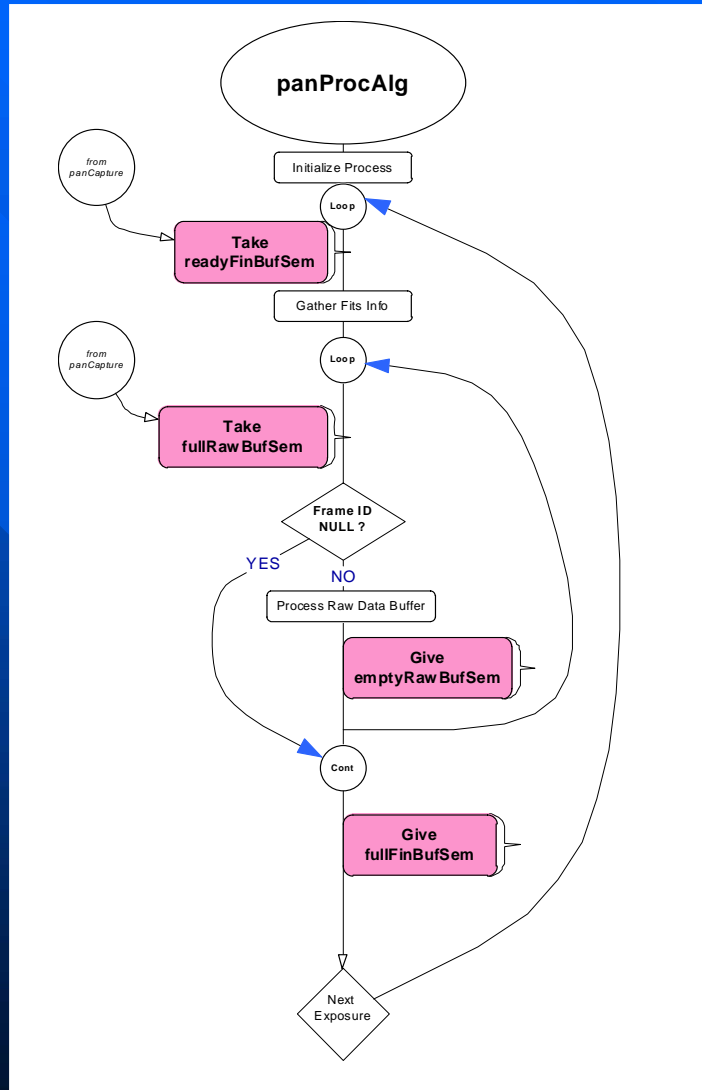
PAN Detailed Process Interconnects



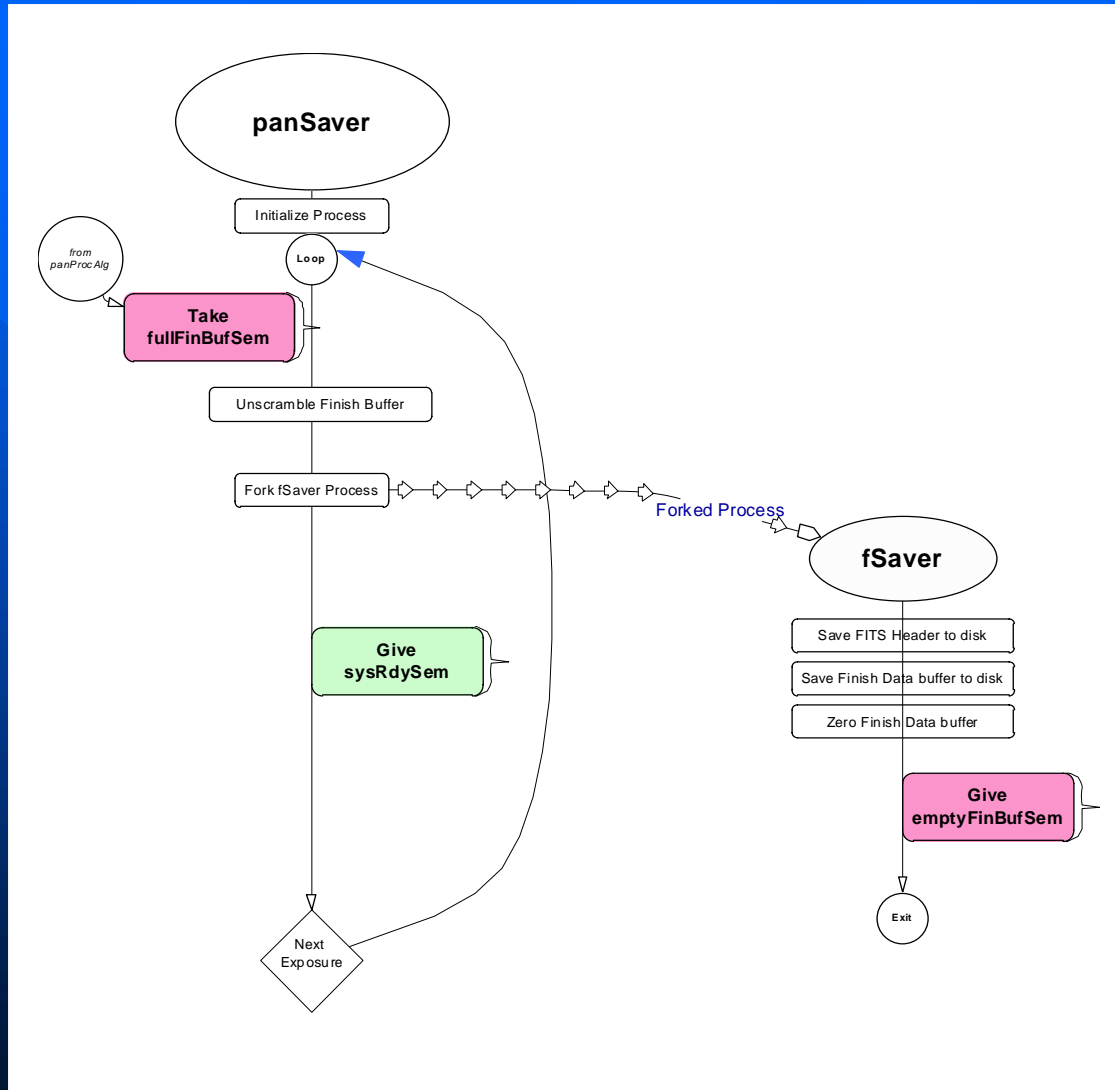
PAN Process Coordination



PAN Process Coordination



PAN Process Coordination



Process State Diagrams

MONSOON Software Review

Questions Session

Break

Northern Lunch

PAN Security Issues

- Preventing unauthorized access.
 - Legal Connection List. Source IP address.
 - Firewall.
- Verifying parameter values.
 - Trusted routines.
 - Internal parameter sets.
 - Password restrictions.
- Restricting Parameter access.
 - Password protection on privileged connection.
 - No Client access to certain parameters.

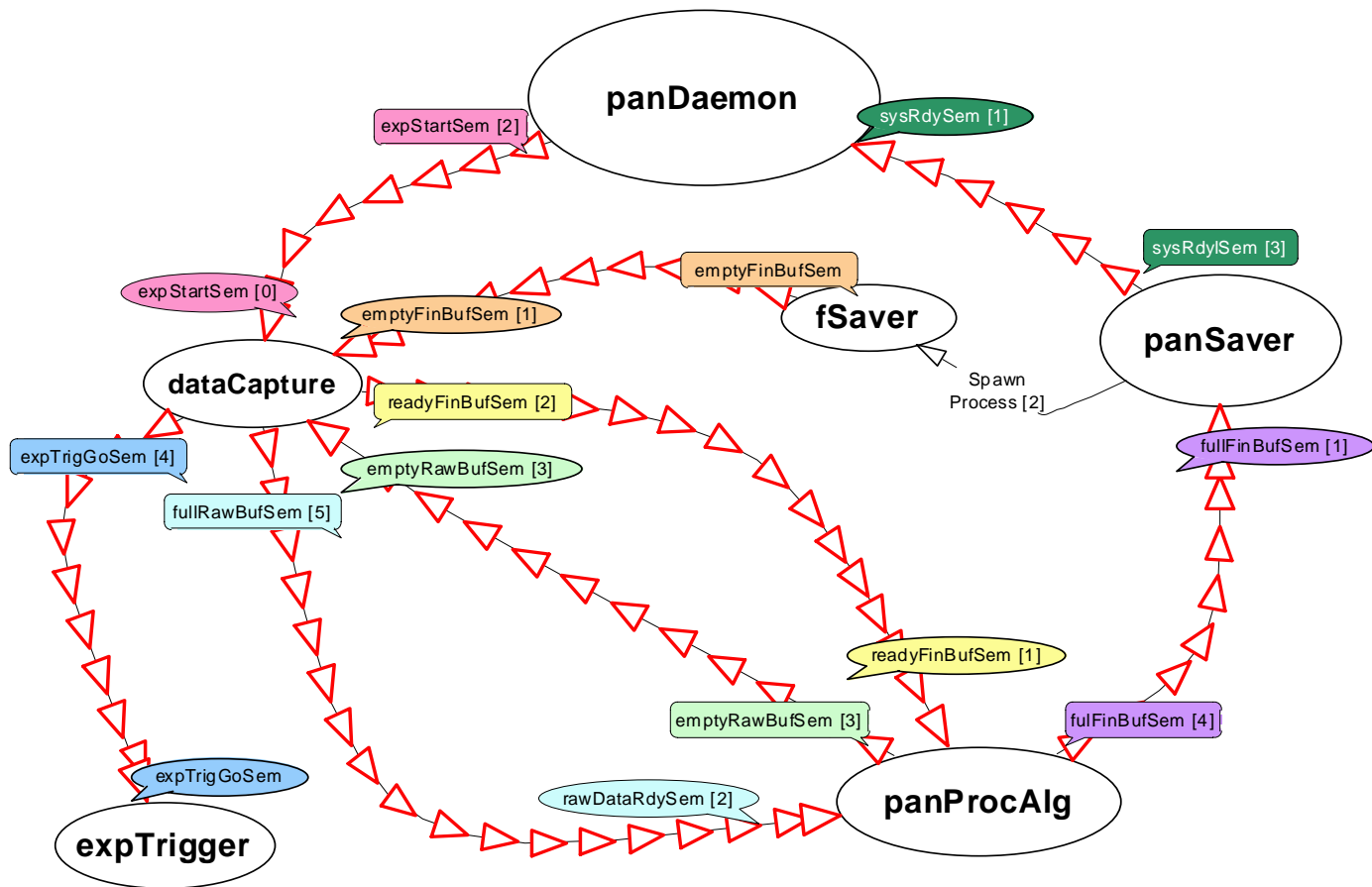
Error Detection/Recovery

- Client-PAN communication link errors.
 - Caught by reliable connection protocol.
 - Command String verification.
- Socket connection errors.
 - Client responsible for reconnection.
 - Responses to pending commands dropped.
- Disk full errors.
 - Spare disk partition for small number of frames.
 - Exposure lockout until Data removed from disk.

Error Detection/Recovery

- PAN process errors.
 - Process verification at key events.
 - Process kill and restart automatic.
- Deadlock Detection/Recovery.
 - High level time outs.
 - Semaphore release on shutdown or failures.
 - Queue content control. (Special buffer ID's).
- Livelock Prevention.
 - Semaphore sequence control.
 - Semaphore count control.
 - Routine invariants.

PAN Semaphore Usage



Error Detection/Recovery

- DHE communication link errors.
 - Timeouts.
 - Echo-back errors.
 - Systran driver errors.
- DHE parameter errors.
 - DHE parameter verification errors.
 - DHE parameter value drift.
- DHE access errors (bad address).
- DHE reset or power glitch.
 - Async Message errors.

Software Management, Test Plan,
Resources
&
Schedule

Management

- Design by Consensus.
- Depend on SWE experience.
- Pseudo-Code by lead designer.
- Final coding by lead or second programmer.
- Verification by second or lead programmer.
- Re-Verification by designer.
- Final test & verification in Use.

Documentation

- OCCD and FPRD published but not complete.
- System Architecture document in work.
- ICD's Published and reviewed.
- Library API Published.
 - Maintained with Source code.
- Process description documentation begun.
 - Using source code Documentation system for processes.
 - Manually produced text used to detail process interactions.
 - Queues and Semaphores described in added text.

Test & Verification

- Much functionality in Libraries.
- Individual Library Test programs.
 - Functions verified against description.
 - Function invariants tested.
 - Library functions tested and verified independently.
 - Test program constructed for each library.
 - Interactions tested w/multiple test program instances.
- In Debug system.
 - Currently libraries are being used in dheHdwDbg.
 - Revealed bugs fixed and re-verified in individual test.
 - New version tested immediately.

Test & Verification (cont)

- In PAN process system.
 - Processes tested against expected behavior using dheSim.
 - Processes verified against functional description.
 - Process invariants tested and verified.
 - Revealed bugs fixed and re-verified in individual test.
 - New version tested immediately.
 - Pan process test bed planned.

Schedule & Resources

- Libraries tested & running in dheHdwDbg (May `03)
 - Generic Interface Libraries
 - Hardware Specific Libraries
 - Some General Utility Libraries
- PAN Interface & processes running DHE (Sept `03)
 - Remaining General Utility Libraries
 - PPX interface Library complete
- TCL/TK Engineering Client (Sept. '03)
- Everything Needed for NEWFIRM (Nov `03)
- Multi-Pan system development starting (Jan '04)

Schedule & Resources (cont)

- 1.0 FTE Sr. Software Engineer NCB
- 0.5 FTE Sr. Software Engineer PND
- 0.1 FTE “black market” Software Engineer II PBR

$$1.1 + .55 + .11 = 1.76 \text{ FTE}$$

NOAO software engineers give 110%

MONSOON Software Review

Questions Session

Panel Deliberation

Panel Verbal Report

MONSOON Software Review

Appendices

- I. Methods to Achieve Requirements.
- II. Process State Diagrams.
- III. Interface Control documents.
 - A. ICD 4.0 GPX Command, Status and Data Interface.
 - B. ICD 5.0 PPX Command, Status and Data Interface.
 - C. ICD 6.0 Generic DHE Command Interface.
 - D. ICD 6.1 MONSOON PAN to DHE Interface.
- IV. Original Project Documents
 - A. MONSOON OCCD
 - B. MONSOON FPRD

Methods for Achieving MONSOON Software Requirements.

Science Generated Requirements
Detector Development Requirements
System Generated Requirements
Software Development Requirements

Science Software Requirements

- Support detector safe operations.
 - Use “Standard Parameter Sets”.
 - Parameter range restrictions in science operations.
 - Only Limited parameter tuning permitted.
 - Bias power off until detector voltages confirmed correct.
 - Detector engineer determines “safe” operation.
- Support detector limited performance.
 - Use high-end giga-Hz CPU’s.
 - No restrictions on speed built into software.
 - Allow overlap of execution to greatest extent possible.

Science Software Requirements (cont)

- Support both IR and OUV detector systems.
 - Per detector runtime command configuration.
 - Runtime pre-processing algorithm selection.
 - Base system code non-specific.
 - User interface left to Observatory/Instrument staff.
- Extensible to “Very Large Focal Planes”.
 - Individual PAN’s know only about their FP section.
 - Final image assembly left to DHS.
 - MSL knows how to deliver commands to FP sections.

Science Software Requirements (cont)

- Able to treat mosaics as single focal plane.
 - User view is of an image server.
 - MSL deals with details of multiple PAN's.
 - Data sent to DHS by individual PAN's for reassembly.
- Support science observation by 'named' modes.
 - Mode database converts names to Parameter settings.
 - MSL knows how to send parameter settings to PAN's.
 - Instrument/detector scientists/engineers determine Modes.
 - User can add features to base modes within restrictions.
 - User modes savable in database.

Science Software Requirements (cont)

- Provide for efficient science operation support.
 - Allow overlap of execution to greatest extent possible.
 - Provide Modes to support Focus, Calibration frames, etc.
 - Provide fast setup and configuration.
- Support high observing efficiency.
 - Allow overlap of execution to greatest extent possible.
 - Provide Modes to support Focus, Calibration frames, etc.
 - Provide fast setup and configuration.
- Support existing and new observing paradigms.
 - Connections permit Remote observing, Queue observing,
 - Published Interfaces allow automation of observations.
 - Facilities for adding functionality allow new modes.

Science Software Requirements (cont)

- Provide ROI support for readout speed up.
 - Single ROI to reduce readout time.
 - ROI spans chips in restricted ways.
 - Allowed alternatives determined by mosaic layout.
- Provide ROI support for data compression.
 - Readout time not reduced all pixels readout.
 - Multiple ROI's allowed.
 - Only data within the ROI's sent to DHS.
- Support “technical” imaging. (Guiders, etc.).
 - Runtime configuration allows customization.
 - Detector library permits unique handling for purpose.

Science Software Requirements (cont)

- Array configuration by ‘standard parameter set’.
 - Detector setup by named Mode.
 - Parameter set database converts names to parameter settings.
 - Provision for expansion of Database by Users.
- Parameter sets determined in the Detector Lab.
 - Initial database created by detector engineer.
 - Instrument scientist can tune modes for instrument.
 - User has limited ability to modify base operating mode.
- Provide limited tuning of detector performance.
 - User can modify restricted range & class of parameters.

Science Software Requirements (cont)

- Provide limited ‘on-the-fly’ reconfiguration.
- Allow addition of new processing algorithms.
 - Processing algorithm can be restarted with new algorithm.
 - New data taking algorithms can be loaded.
 - New detector waveforms can be loaded as needed.
- Provide for an instrument calibration mode.
 - Instrument Client can run MSL/PAN’s automatically.
 - Password restriction allows access to restricted parameters.
 - Engineering console can be used to tune parameters.

Detector Development Requirements

- Support detector characterization operations.
 - ppx interface provides Low-level control of parameters.
 - Interface allows automated control of process.
 - “on the fly” reconfiguration allows new algorithm development.
 - Library & process APIs allow convenient development.
- Support detector research and development.
 - Runtime configuration allows new detector types.
 - “on the fly” reconfiguration allows new operation modes.
- Support hardware development & debugging.
 - Ppx interface allows Low-level hardware command support.

System Software Requirements

- Support efficient boot-up and initialization.
 - Each PAN boots up independently.
 - Boot-up only depends on local information.
 - Runtime configuration sent to all PAN's.
 - Communication mechanism “universal” & easy to use.
- Start-up & initialization without intervention.
 - Startup scripts keyed to Instrument/system.
 - MSL does Startup time checking to insure correct operation.
 - Crashed/down nodes restarted automatically.
 - Start-up monitored by logging mechanism.

System Software Requirements

- Support system operations logging.
 - Error detection and recovery logging.
 - Command sequence playback.
 - Status display of current state saved with exposure.
- Support connection security.
 - Multiple connections allowed.
 - Connection security enforced (firewall, IP restrictions).
 - On-Telescope connection priority observed.
 - Engineering level commands password protected.

System Software Requirements

- Support convenient error detection/recovery.
 - Routines track and inherit status.
 - First error is reported and correction is attempted.
 - Errors reported and corrected automatically if possible.
 - Error help included in report.
- Support remote diagnosis, debug & operation.
 - Multiple connections permitted.
 - Engineering connection password protected.
 - Test facilities built into system at several levels.

System Software Requirements (cont)

- System to include simulation capabilities.
 - Hardware libraries provide simple simulation.
 - Simulation automatic in absence of Hardware.
- Help available for commands/parameters.
 - Help provided by Runtime Configuration process.
 - Command help available for every command.
 - Parameter help available for settable parameters.
- System layered to allow maximum reuse.
 - General Purpose & Generic Interface libraries reusable.
 - Top level process code reusable.
 - Framework of Hardware Specific libraries reusable.

System Software Requirements (cont)

- Pixel data processing chain re-configurable.
 - Multiple processing algorithms supported in panProcAlg.
 - Multiple panProcAlg versions supportable.
 - Process interconnects will be well defined and published.
- Support efficient configuration for new systems.
 - Engineers define hardware with spread sheet.
 - Runtime configuration derived from spread sheet.
- Configurable w/o re-compile of base code.
- Features added without rebuilding system.
 - Hardware specific libraries compiled as shared libraries.
 - New configurations read at runtime.

System Software Requirements (cont)

- Software to use well-defined interfaces.
- Interfaces documented and published.
- Use GPX Interface to outside world.
- Documentation maintained with code base.
- Documentation in standardized format.
 - Library API created by “make” process and published.
 - Process interface being documented and published.
 - GPX or PPX interface used as interfaces to Clients.
 - API’s, ICD’s & Process interface externally available.
 - Web publication planned.

System Software Requirements (cont)

- Support “Package-like” Installation.
 - Planned.
- Support verification & removal of old versions.
 - Planned.
- Develop Source code Maintenance Manual.
 - Will be developed from source code API creator.
 - Manual will include library and Process descriptions.

Development Software Requirements

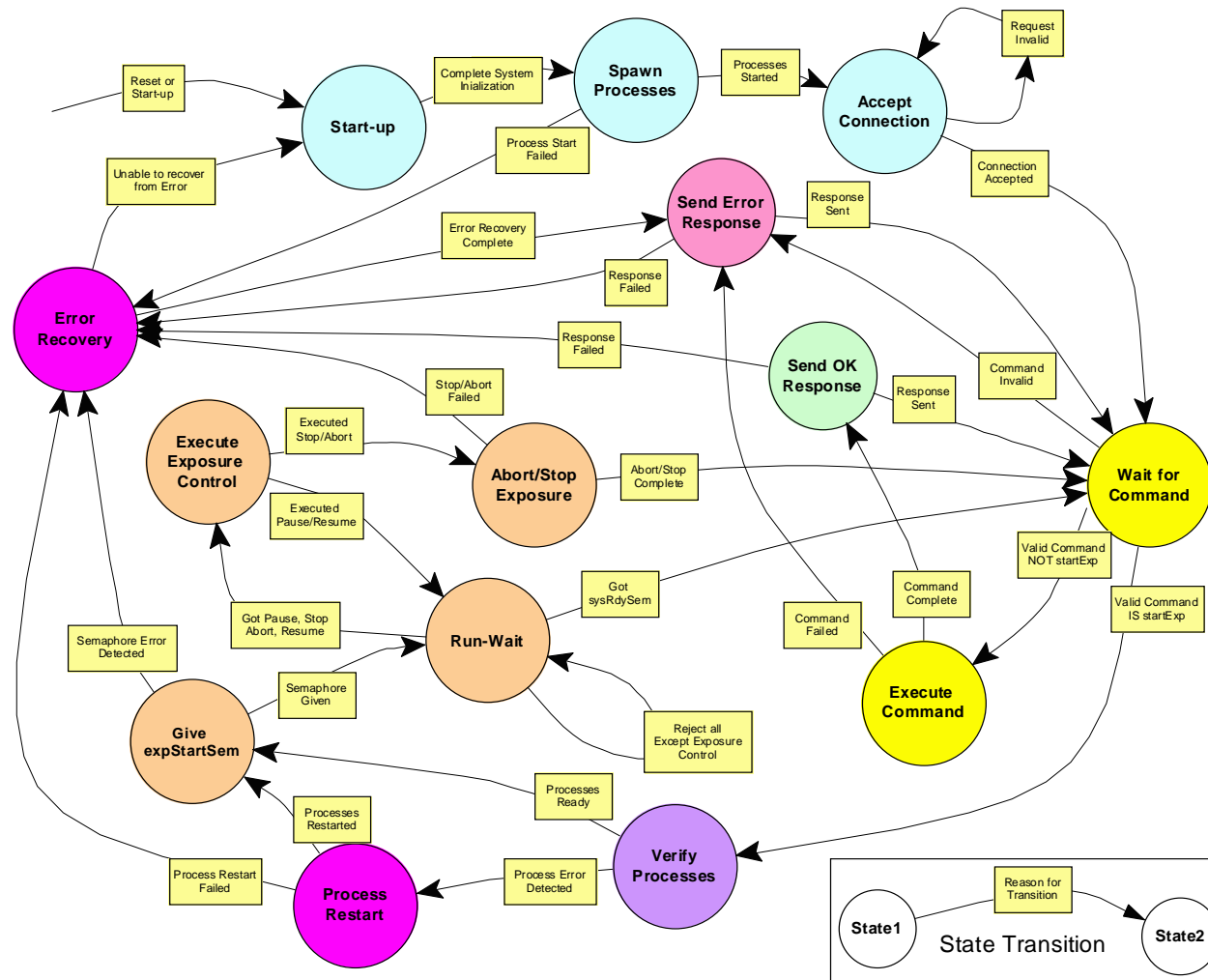
- All Software to be Open Source.
 - Code available on externally visible disk.
 - WEB obtainable source code planned.
- Use widely available software technologies.
 - Sockets, Semaphores, Shared Memory.
 - ANSI standard 'C' compiler.
 - 'POSIX' compliant standard Libraries.
- Use free tools to the greatest extent possible.
 - 'C', LINUX, TCL/TK.
 - Firmware using XLINIX WEBPACK.

Development Software Requirement

- Support multi-site distributed development.
 - CVS Tree on externally visible disk
 - Individual modules obtainable through CVS. (NI).
- Use Source code version control (CVS).
 - CVS Tree built and in use at NOAO.
 - CVS Tree available on externally visible disk.
- Testing and verification built into development.
 - Testing plan for Libraries.
 - PAN process test-bed planned.
 - dheSim complete and operating at rudimentary level.

Process State Diagrams

panDaemon State Diagram



panDaemon States

- Start-up.
 - Read detector configuration, Initialize data Buffers.
 - Attach shared memory, Initialize shared memory.
 - Read DHE configuration.
- Spawn PAN Processes.
 - Start panCapture, panProcAlg, panSaver.
- Accept connection.
 - Accept connection requests.
 - Verify permissions.
- Send Error Response.
- Send OK Response.

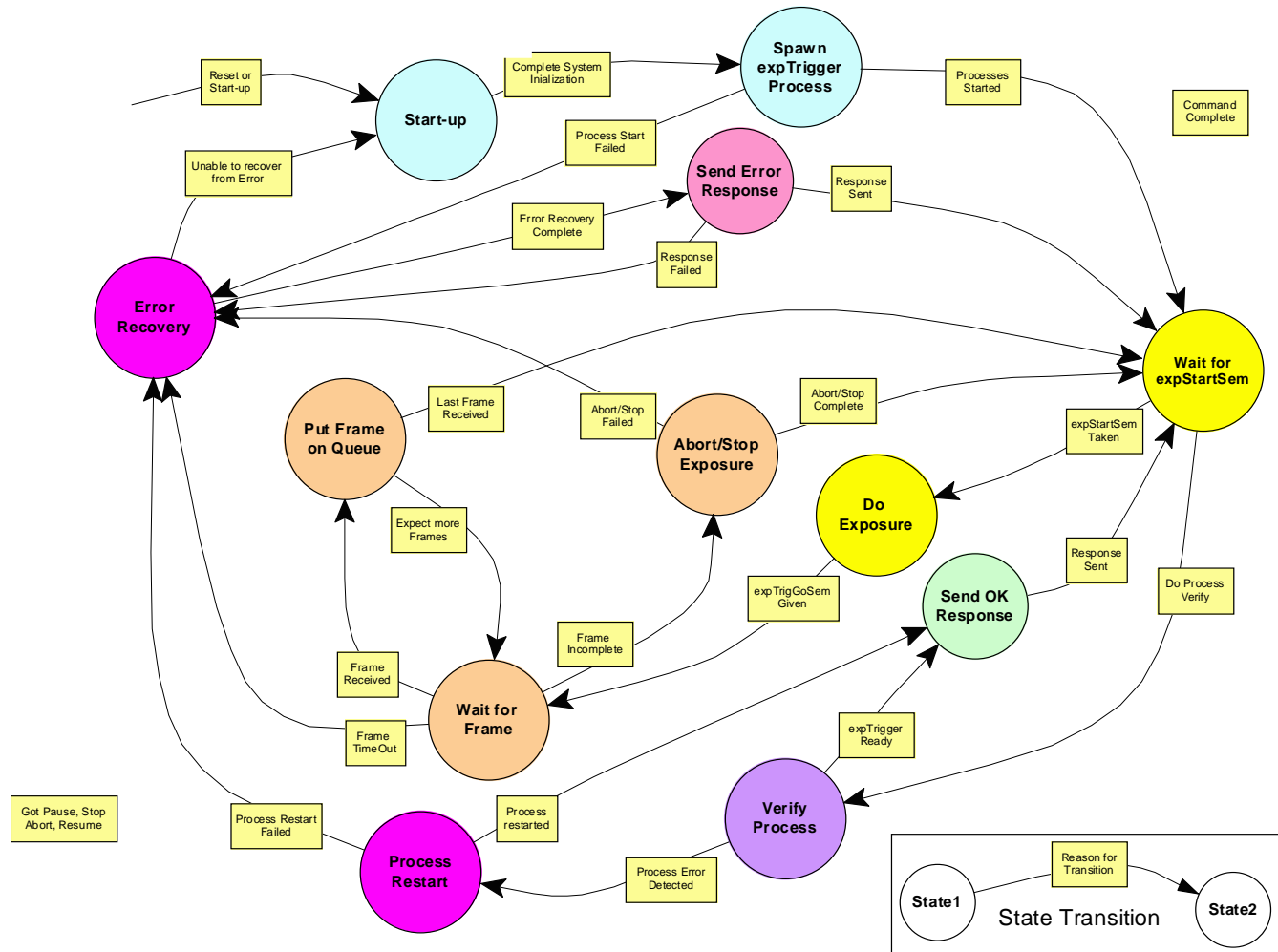
panDaemon States

- Wait for command.
 - Accept, Parse & verify command string from socket or stdin.
- Execute Command.
 - Execute appropriate command function.
- Run-Wait.
 - Accept command string from socket or stdin.
 - Parse command, reject all except Exposure control.
 - Wait for sysRdySem.
- Execute Exposure Control.
- Abort/Stop Exposure.

panDaemon States

- Process Verify.
 - Verify health of other PAN processes.
- Give expStartSem.
- Process Restart.
 - Check on process existence, kill if required.
 - Check on process resource use. Set resources to initial state.
 - Restart process.
- Error Recovery.
 - Determine error.
 - Correct or abdicate responsibility to user.

panCapture State Diagram



panCapture States

- Start-up.
 - Attach shared memory, Initialize shared memory.
- Spawn expTrigger Process.
 - Spawn process check for startup.
- Wait for expStartSem.
 - Do a semTake on expStartSem.
- Error Recovery.
 - Determine error state.
 - Correct or abdicate responsibility to user.
 - Clean out image processing chain.

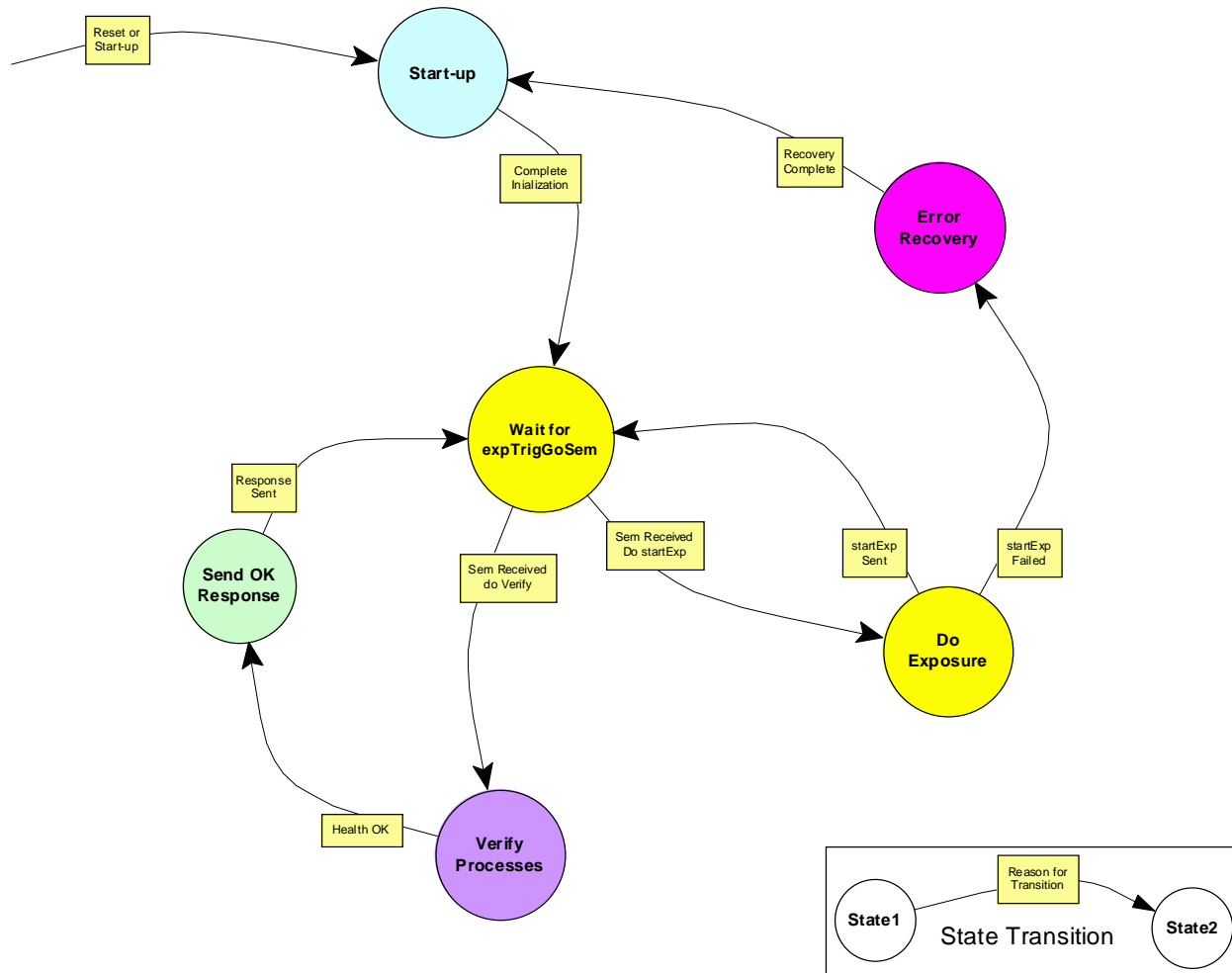
panCapture States

- Do Exposure.
 - Obtain image finish buffer, Prepare meta-data buffers.
 - Pass finish and meta-data to panProcAlg.
 - Obtain empty Raw Data Buffer.
 - Give expTrigGoSem.
- Wait for Frame.
 - Do dheGetData to receive data buffer.
- Frame on Queue.
 - Pass full raw data buffer ID to panProcAlg.
- Abort/Stop Exposure.
 - Clean out processing chain and image buffers.

panCapture States

- Process verify.
 - Check on expTrigger process.
 - Report own health.
- Process Restart.
 - Restart expTrigger if necessary.
- Send OK Response.
- Send ERROR Response.

expTrigger State Diagram



expTrigger States

- Start-up.
 - Attach shared memory, Initialize shared memory.
- Wait for expTrigGoSem.
 - Do a semTake on expStartSem.
 - When taken go to “Do Exposure” or “Process Verify”.
- Do Exposure.
 - Wait pre-determined delay time.
 - Send startExp command to DHE.
- Process verify.
 - Report process health.
- Error Recovery.

panProcAlg States

- Start-up.
 - Attach shared memory, Initialize shared memory.
- Wait for readyFinBufSem.
 - Do a semTake on readyFinBufSem.
 - When taken go to “Do Exposure” or “Process Verify”.
- Error Recovery.
 - Determine error state.
 - Correct or abdicate responsibility to user.
 - Clean out image processing chain.

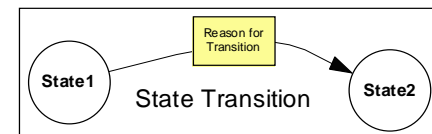
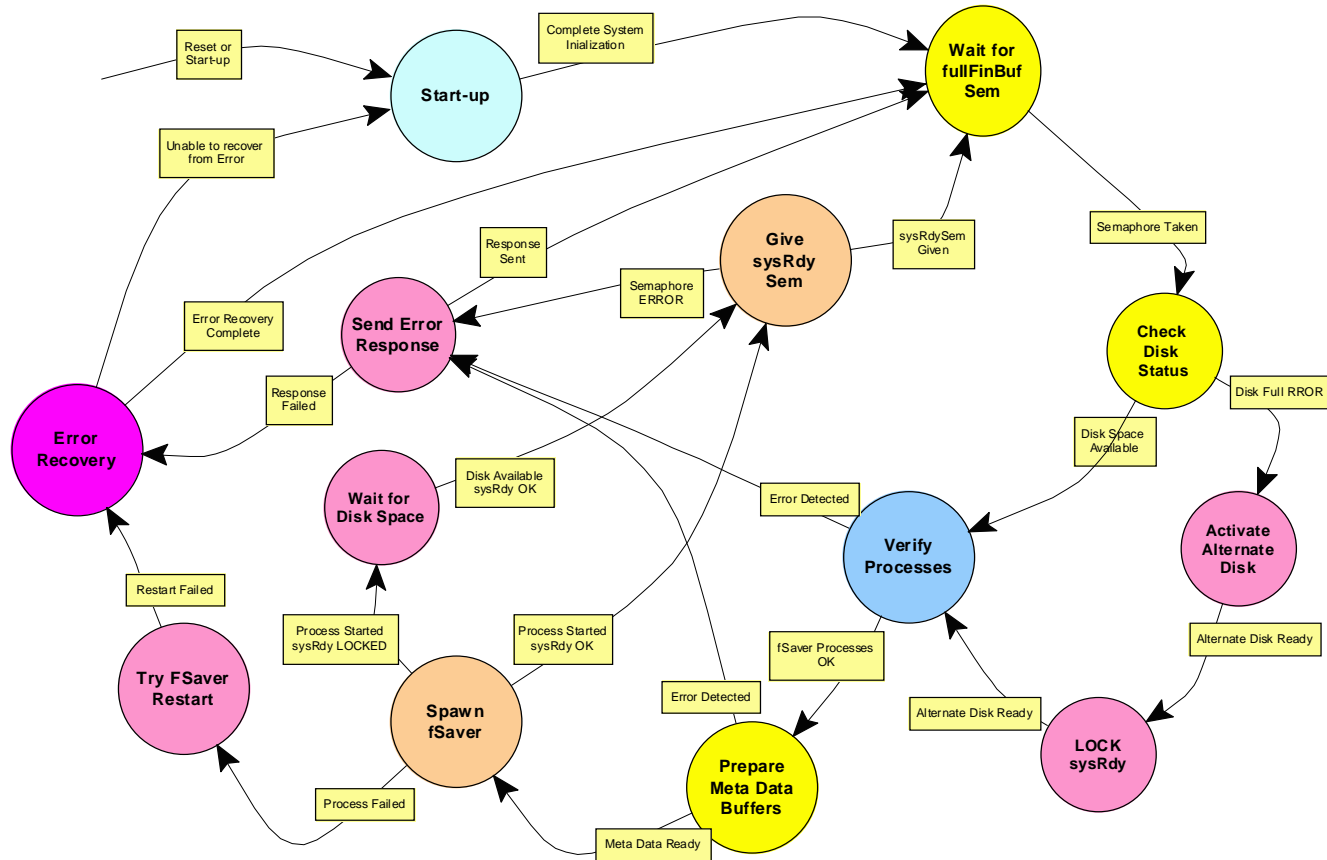
panProcAlg States

- Do Exposure.
 - Obtain finish image buffer, prepare meta-data buffers.
 - Obtain full raw data buffer.
 - If buffer ID valid.
 - » Process Raw data buffer into Finish buffer.
 - » Put raw buffer ID onto emptyRawBuf Queue.
 - » Jump to arrowhead.
 - If buffer ID NULL .
 - » If state is ABORT Got to Abort Exposure.
 - » Otherwise Pass Finish buffer ID to panSaver & go to Wait.

panProcAlg States

- Abort Exposure.
 - dump finish data buffer,
 - put on emptyFinBuf Queue.
 - go to Wait for readyFinBufSem.
- Process verify.
 - Report process health.

panSaver State Diagram



panSaver States

- Start-up.
 - Attach shared memory, Initialize shared memory.
- Wait for fullFinBufSem.
 - Do a semTake on fullFinBufSem.
- Check Disk Status
 - In disk Full go to Activate Alternate Disk.
- Activate Alternate Disk
 - Check Alternate Disk Available.
- Lock sysRdySem.
 - Collect termination data on previous fSaver programs.

panSaver States

- Process verify.
 - Verify fSaver processes health.
 - Report process health.
- Prepare Meta Data.
 - Obtain finished image buffer.
 - Prepare meta-data buffers.
- Spawn fSaver.
 - Provide fSaver with finish Buffer ID.
 - If sysRdySem LOCKED go to Wait for Disk Space.
 - If sysRdySem OK go to Give sysRdySem.

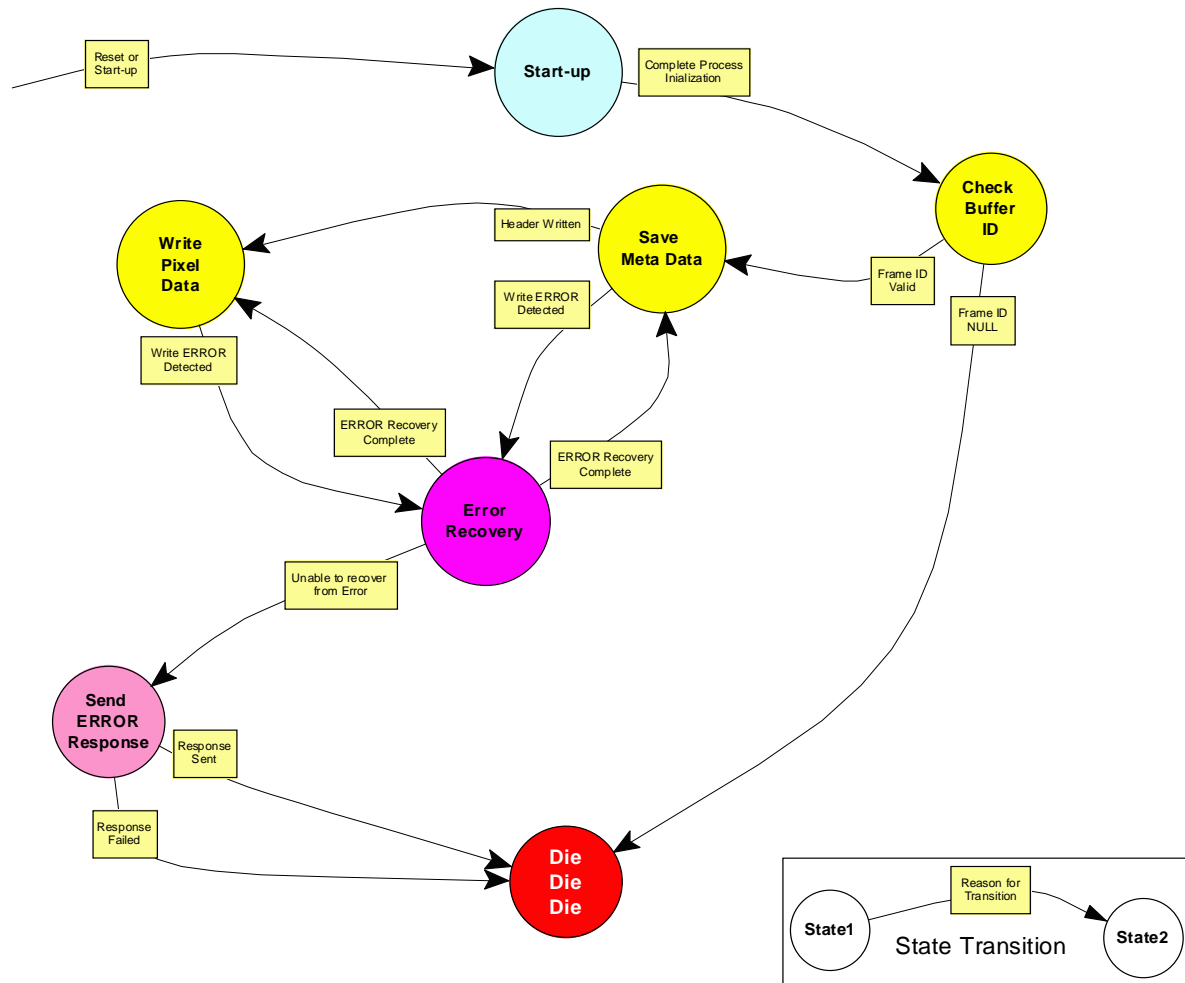
panSaver States

- Give sysRdySem.
 - Give sysRdySem to panDaemon.
- Wait for Disk Space.
 - Wait until disk space is available on Primary Disk.
 - Move alternate Disk to primary disk.
 - Unlock sysRdySem.
 - Go to give sysRdySem.
- Send ERROR Response.

panSaver States

- Try fSaver Restart
 - Try to start fSaver if Fail go to Error Recovery.
- Error Recovery.
 - Determine error state.
 - Correct or abdicate responsibility to user.
 - Clean up file pointers, disk etc.
 - Zero fullFinBuf return to emptyFinBuf Queue.

fSaver State Diagram



fSaver States

- Start-up.
 - Attach shared memory, Initialize shared memory.
- Check buffer ID.
 - If NULL got to Die! Die! Die!
 - If valid go to save Meta Data.
- Save Meta-Data.
 - Save Header data to disk.
 - Zero meta-Data.

fSaver States

- Write Pixel Data.
 - Save pixel data in buffer to Disk.
 - Zero fullFinBuf, Put on emptyFinBuf Queue.
- Error Recovery.
 - Is successful return to previous state.
 - If fail go to send ERROR Response.
- Send ERROR Response.
 - Go to Die, Die, Die!
- Die, Die, Die!
 - Terminate program.

Source Code In-line Documentation


```

/*****
* dheHdwrOpen.c - hardware specific open routine
*
* __doc__ \subsection {dheHdwr.h}
* __doc__ \begin{description}
* __doc__ \item[\sc use:] \emph{void dheHdwrOpen ( long $\star$status, char $\star$response, ulong unitNum,
* __doc__ \item[\sc description:] This function handles the details of opening the dhe device.
* __doc__ it checks that the dhe hardware library has been initialized, opens the communication
* __doc__ device, verifies the dhe hardware is connected and if there is a failure or we are simulating
* __doc__ it returns SIMULATION_OK. When this routine returns the dhe should be ready to go.
* __doc__ \end{description}
*
* History:
* 20020903 - created file for Library - ncb
* 20030226 - added function description documentation - ncb
*
*****/
#include
#define DEFAULT_MONSOON_FILE "monsoon.cfg"
#define DEFAULT_MONSOON_DIR "."

void chk4DheHdwr(long *status, /* status return ulong */
                char *response, /* optional string response to upper level */
                dheHandle dheId /* handle to tell library which DHE to use (usually only one/machine) */
)

```