

LabOS

**Closed-loop, Protocol-driven,
Data Acquisition & Control
for Biomedical Experiments**

July 2006

**Joel M Miller, PhD
Martin Wiesmair, DI(FH)**

Eidactics
San Francisco

Supported by NIH/NIBIB R41 EB006219 to JMM

Problems



Problems of Data Acquisition & Control (1/2)

● Input-Output Coupling Problem ("control" experimental devices)

- ◆ Reliably sample multiple channels
- ◆ Refresh displays & produce other outputs
- ◆ Coordinate input & output
 - loose coordination – “triggering”, “synchronization”
 - closed-loop control – input determines output, on a sample-by-sample basis

● Experimental Protocol Problem ("control" the course of the experiment)

- ◆ A protocol is the plan for the course of an experiment
- ◆ Complex experiments, particularly those with living subjects, may require multi-phased, data-contingent protocols
 - Simple processing pipelines cannot provide this flexibility
 - State machines provide natural descriptions of multi-phase experiments



Problems of Data Acquisition & Control (2/2)

● Operator Interaction Problem

- ◆ Experimental data must be promptly displayed in useful forms
 - Operator must have adequate control of the experiment, and
 - See experimental data as it is collected
 - Requirements of a rich, interactive operator interface tend to be incompatible with deterministic data acquisition & control

● Flexibility Problem

- ◆ Innovative experiments may require protocols unanticipated by vendor “solutions”
- ◆ Complex protocols must be easily developable by experimenters, without complex programming, in response to experience, and changing needs
- ◆ A wide range of standard hardware should work with the basic system



Input-Output Problem example: Oculomotor Physiology

- Realtime data acquisition

- ◆ eye position ~ 6 channels, 16 bits @ 500 Hz
- ◆ muscle force ~ 4 channels, 16 bits @ 1 KHz
- ◆ motor neuron spike profiles ~ 4 channels, 12 bits @ 50 KHz
- ◆ subject response switches ~ 1 digital input port (8 lines)

- Generation of stimuli & other control signals, *contingent on incoming data*

- ◆ LED visual stimulus array ~ 4 digital output ports (32 lines)
- ◆ Reward pump, tone generator, light intensities ~ 6 analog outputs

➡ A demanding set of tasks for a single lab computer!



“Loose-Coupling” Approach to the I-O Problem

1. Separate input & output computers

- ◆ One computer for sampling
- ◆ A second computer for stimulus generation & other outputs
- ◆ Input & output computers are loosely coupled (eg: ethernet, GPIB)

2. A single computer with dedicated input & output devices streaming input data to, or output data from, memory or disk

- ◆ Boards using buffered input and output
- ◆ Stand-alone devices connected to computer ports

➔ These solutions are only suitable where simple interactions like “triggering” & “synchronization” provide sufficient coordination between inputs & outputs

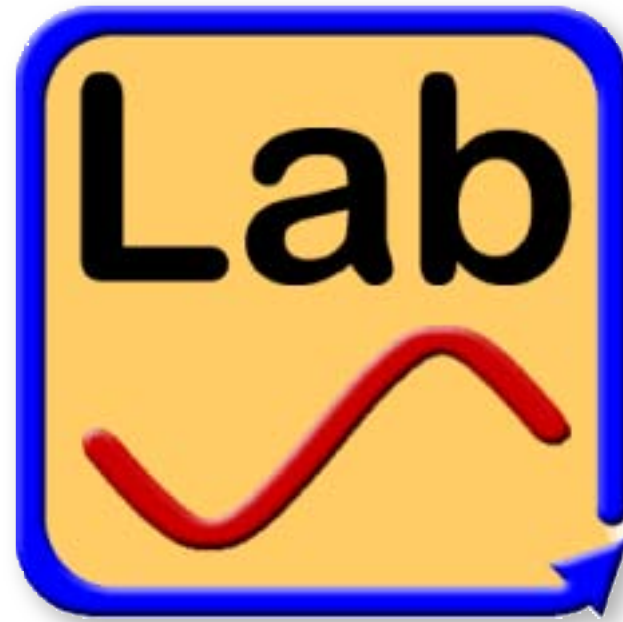
- ◆ Closed-loop control is impossible



Specialized Solutions to the I-O Coupling Problem

- Based on DSP (digital signal processor) chips, rather than general purpose processors
 - High speed sampling and filtering
 - Limited hardware choices
 - Specialized software applications
 - Targeted (limited) programability
- ➔ Such specialized systems
- ◆ May be the best solutions for some applications
 - ◆ But tend to restrict labs to experiments that they were designed to support

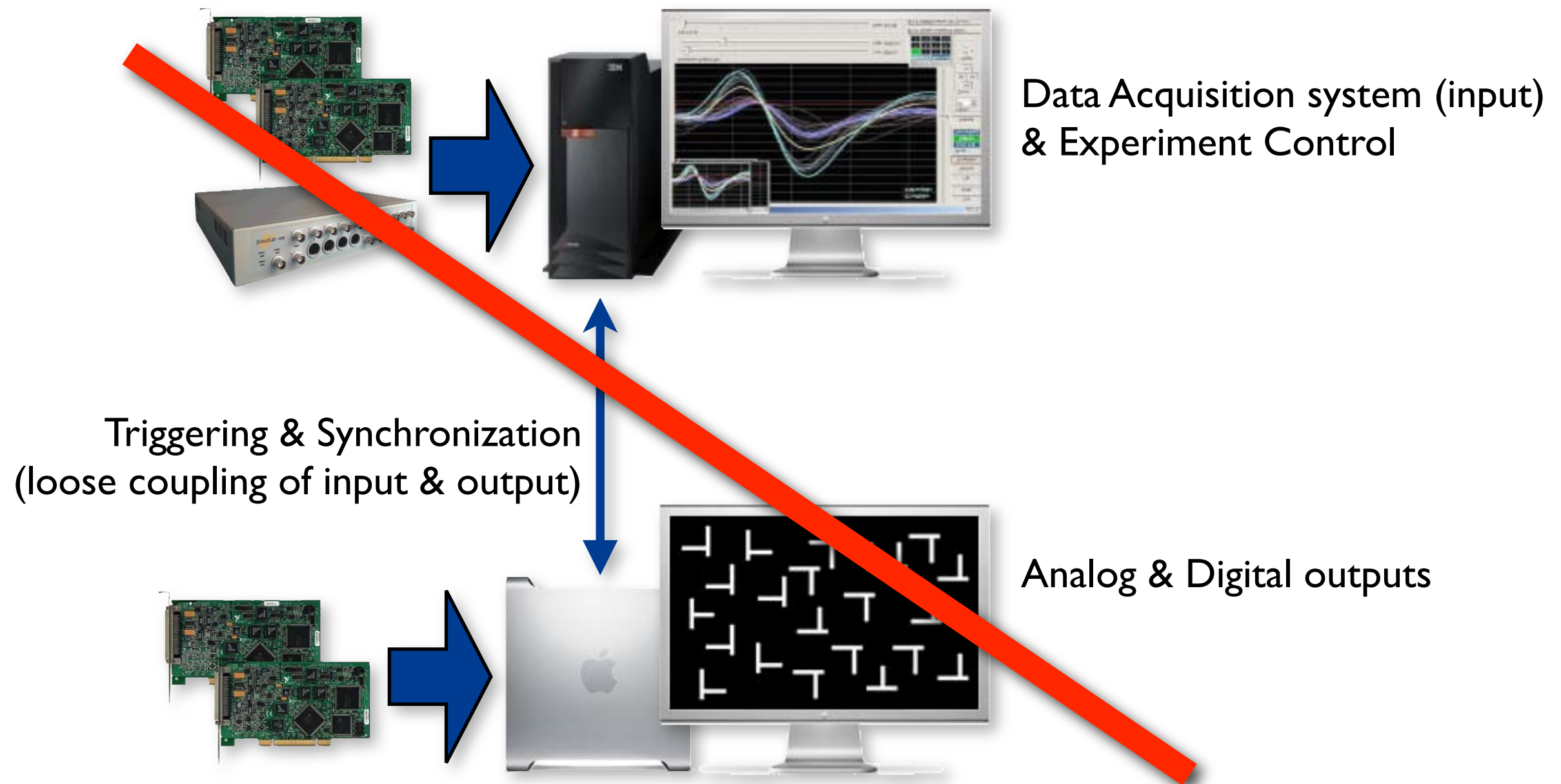




LabOS Solution

LabOS Solution to the I-O Coupling Problem

Two computers configured, not for input & output, respectively ...



LabOS Solution to the Input-Output Problem ^(1/2)

Two computers configured, not for input & output, respectively ...
but for the differing needs of a *human operator* & a *realtime experiment*



Operator's
Console

TCP



Data Acquisition (analog & digital inputs),
Analog & Digital output
& Experiment Control integrated in a
realtime processor



LabOS Solution to the Input-Output Problem (2/2)

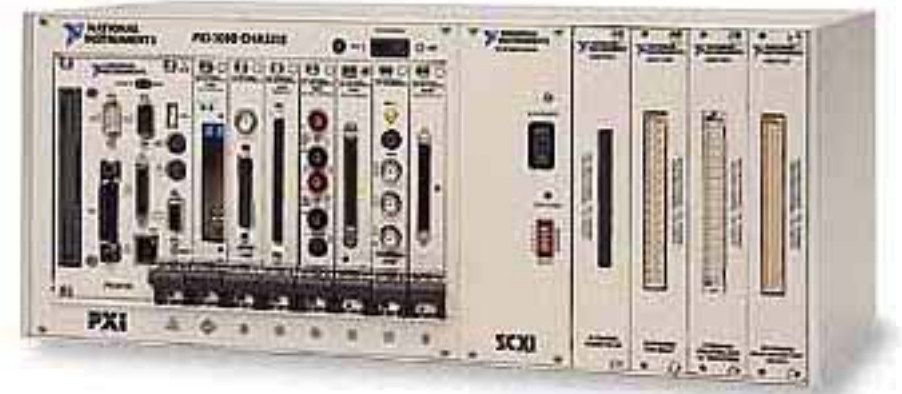
● Operator's Console

- ◆ Provides a rich, interactive interface to the ongoing experiment
- ◆ Responds promptly, as perceived by a human operator



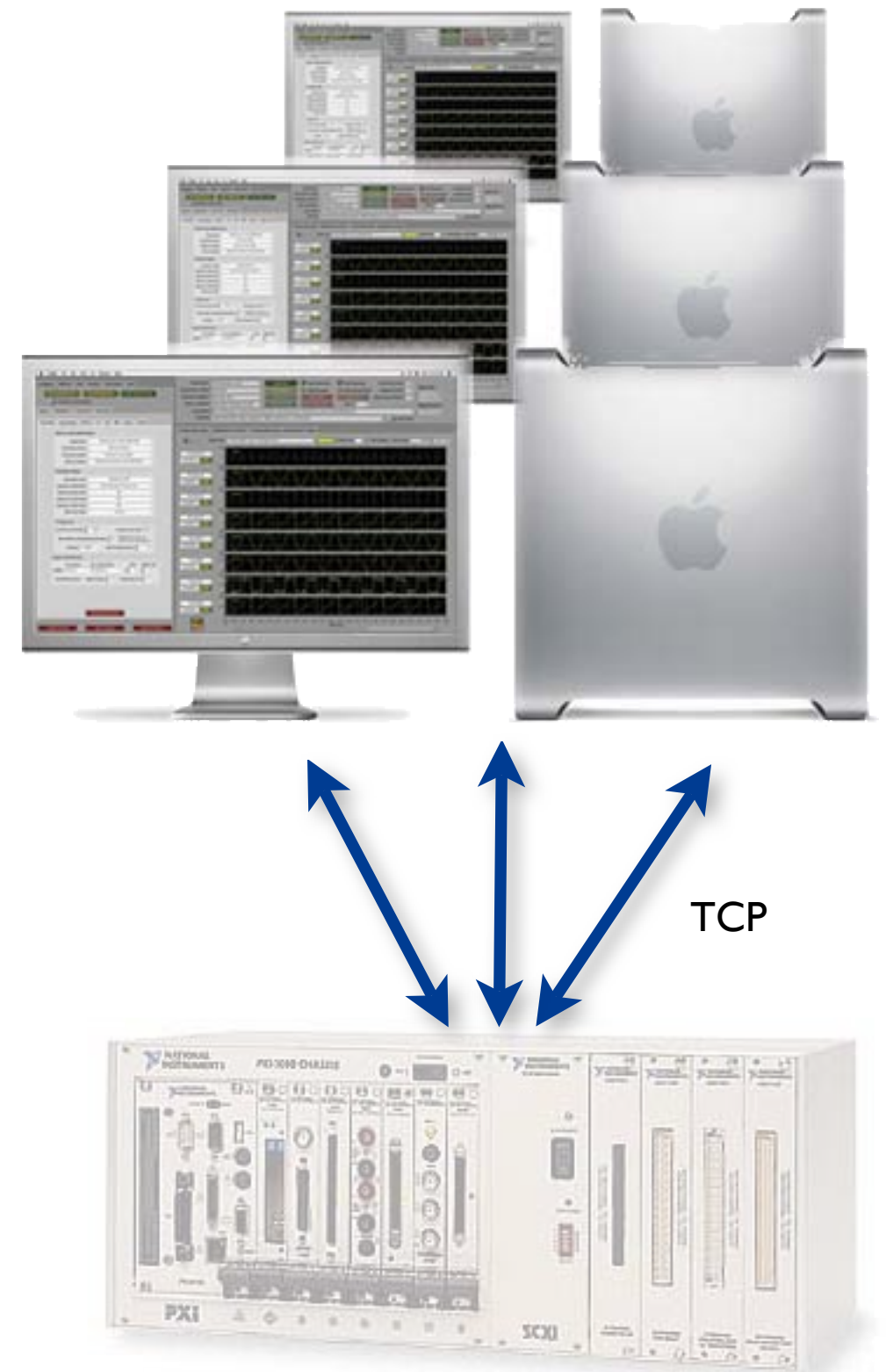
● Experiment Server-Controller

- ◆ Provides deterministic protocol-based control of the experiment
- ◆ Integrates input & output
- ◆ Responds on experiment's time scale



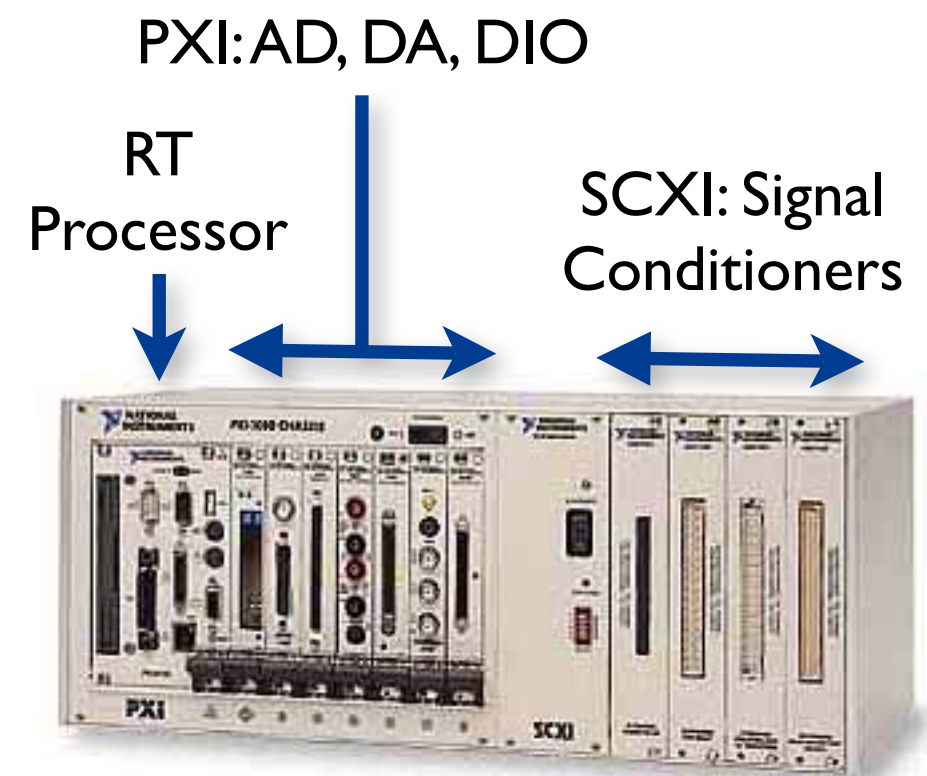
LabOS Operator's Console

- General purpose, multitasking Macintosh, Windows or other computer
 - ◆ Responds on the *Operator's time scale* of ~100 ms to:
 - Implement Operator requests
 - Show experimental data & events
 - ◆ Provides multitasking access to 3rd party applications during experiment
- Multiple consoles can observe ongoing experiment



LabOS Experiment Server-Controller

- A general purpose processor running a realtime operating system (RTOS)
 - ◆ Ardenne *PharLap* ETS (National Instruments LabVIEW RT module) on a Pentium 4
- Deterministic response on the *experiment's* time scale:
 - ◆ Closed-loop control at ~1 KHz
 - All inputs are sampled and calibrated
 - Operator requests are read & responded
 - Protocol machine states are run
 - All outputs are produced
 - ◆ Buffered Analog data sampled at ~50 KHz
 - ◆ Timing resolution = 1 μ s



LabOS Solution to the Protocol Problem

- Complex experimental protocols are naturally expressed as a **state machine**, whose states correspond to meaningful phases of the experiment, eg:
 - ◆ “Present next target”
 - ◆ “Wait for eye to leave fixation window on a saccade”
- States consist of **protocol modules**, which
 - ◆ Perform small, meaningful operations
 - ◆ Are parameterized in calibrated, real-world units, eg:
 - Saccade detector specifies eye velocities in deg/s
 - Fluid rewards are specified in μ l
 - ◆ Are programmed in a general-purpose language
 - ◆ Are refined and reused
 - ◆ Egs: “Detect beginning of saccade”, “Is eye in window?”



LabOS Solution to the Operator Interaction Problem

- Information-rich, task-oriented, Experimenter's Console
 - ◆ Single control Console
 - ◆ Multiple viewing consoles connected to experiment by Internet
 - ◆ Modular control panels and viewers for new experiments
 - ◆ Current and saved data shown in same viewers
 - ◆ Use external applications for off-line data manipulation
- Experimenter-modifiable Protocols run in Experiment Controller
 - ◆ Parameterized, modular experimental protocols
 - ◆ Reusable modules, well-defined in experimenter's terms



LabOS Solution to the Flexibility Problem

● State Machine-based Controller

- ◆ Can express arbitrarily complex, multiphase experimental protocols
- ◆ States consist of clearly parameterized, reusable Protocol Modules
- ◆ Protocol Modules are written in a fully general programming language (National Instruments LabVIEW® or other supported language)

● Powerful Operator's Console

- ◆ Panel is modularized with LabVIEW Tab Containers
- ◆ Diagram (code) has a modular Producer-Consumer architecture

● Well-supported, multi-platform hardware & software

- ◆ Console runs on all computers supported by LabVIEW
- ◆ High-performance Server-Controller hardware is industry standard PXI
- ◆ Additional Server-Controller hardware is compatible with LabVIEW-RT



LabOS Supports Three Classes of User

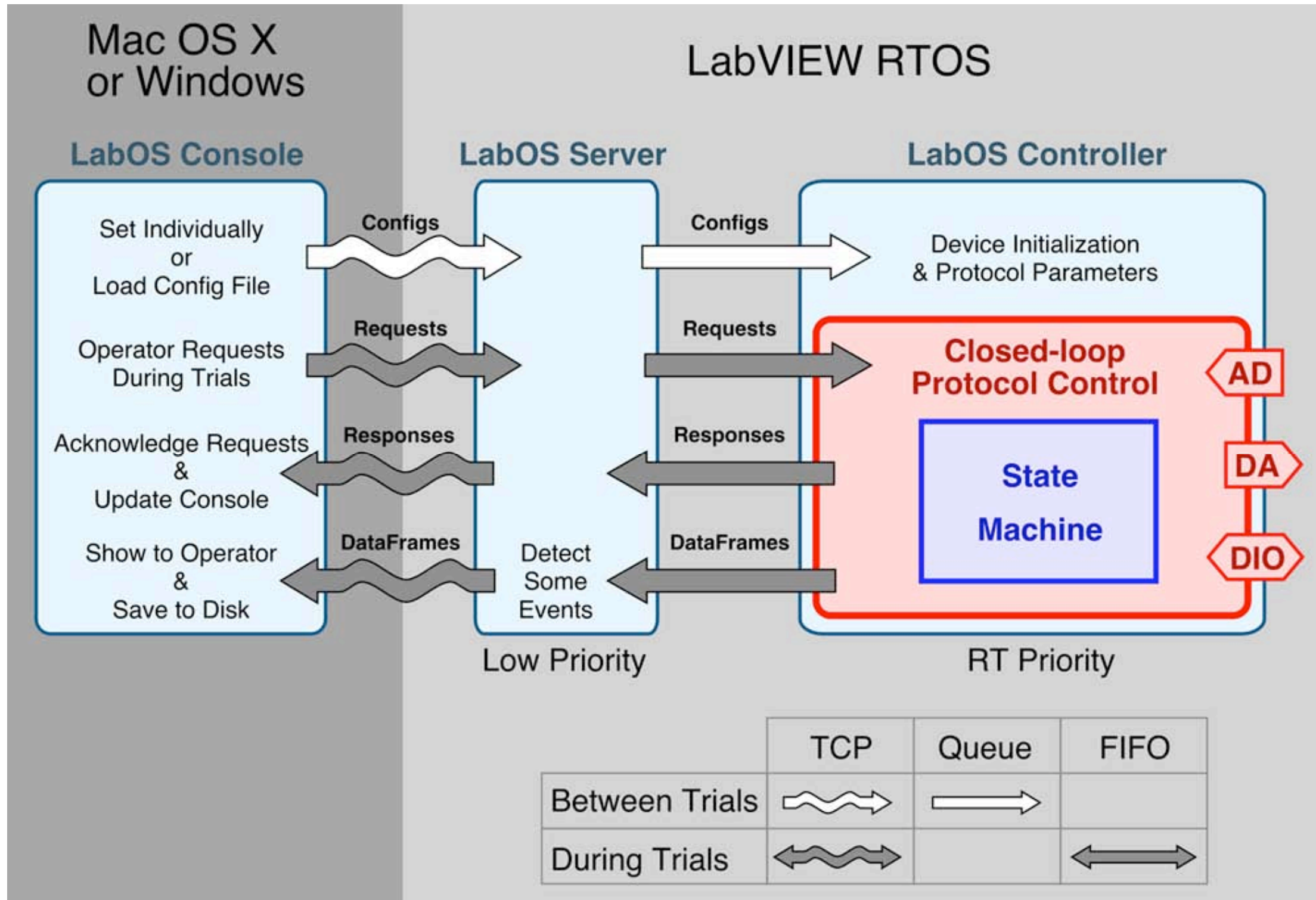
- Operator - runs experiments
 - ◆ Operations are expressed in clear, natural terms; only legal choices (in pop-up lists) & operations (using control dimming) are available
 - ◆ Pre-configured experiments are loaded in a single operation
 - ◆ Data is presented in familiar, natural units
 - ◆ Data files & folders are automatically generated
- Experimenter - designs & configures experiments
 - ◆ Configures system (channels, timings, etc)
 - ◆ Configures experiment by choosing Protocol Modules, setting parameters and sequencing machine states
- Programmer - creates new LabOS modules using LabVIEW or another supported language



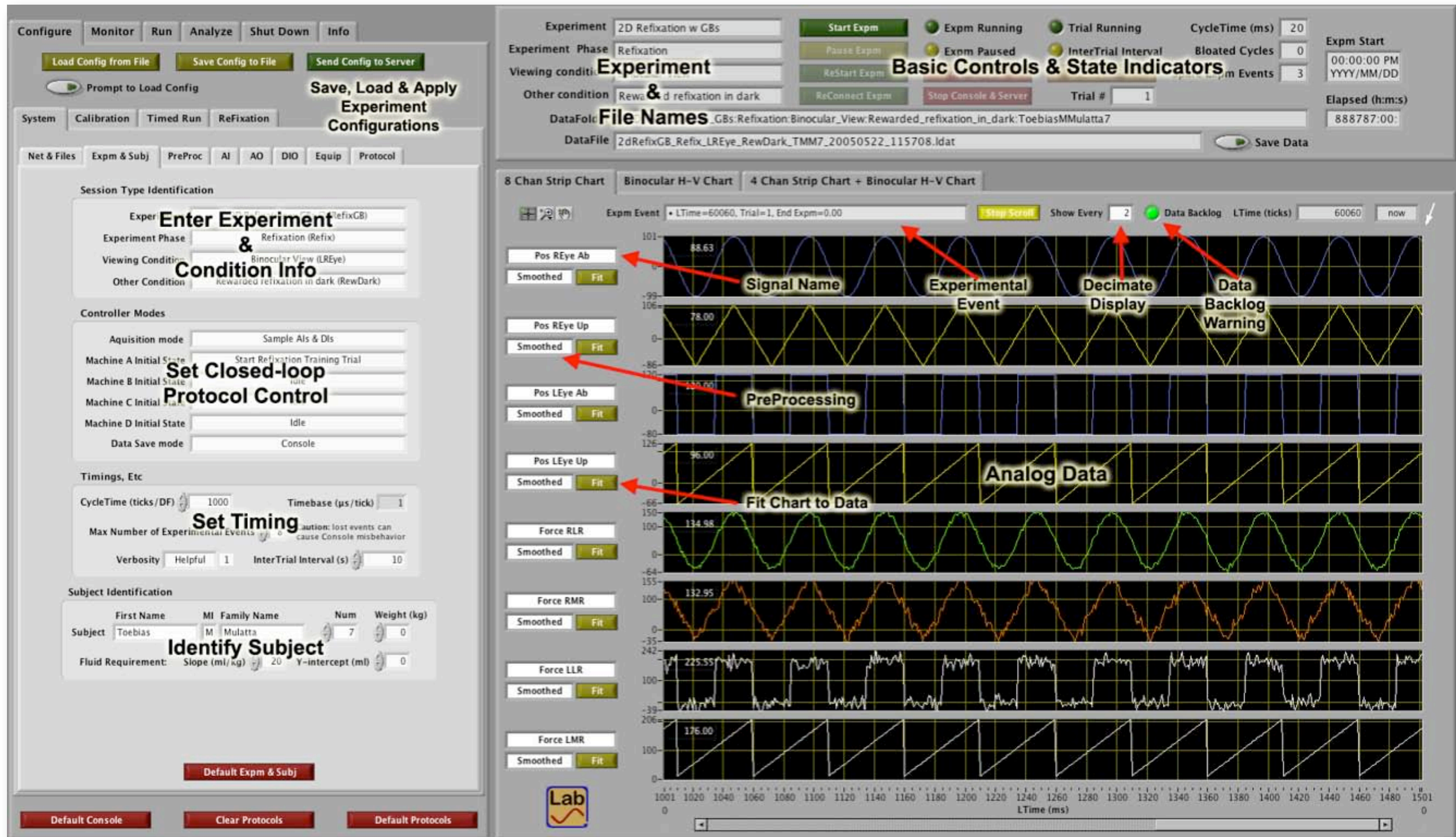


A Look at LabOS

LabOS Architecture – 3 Main Processes



LabOS Console

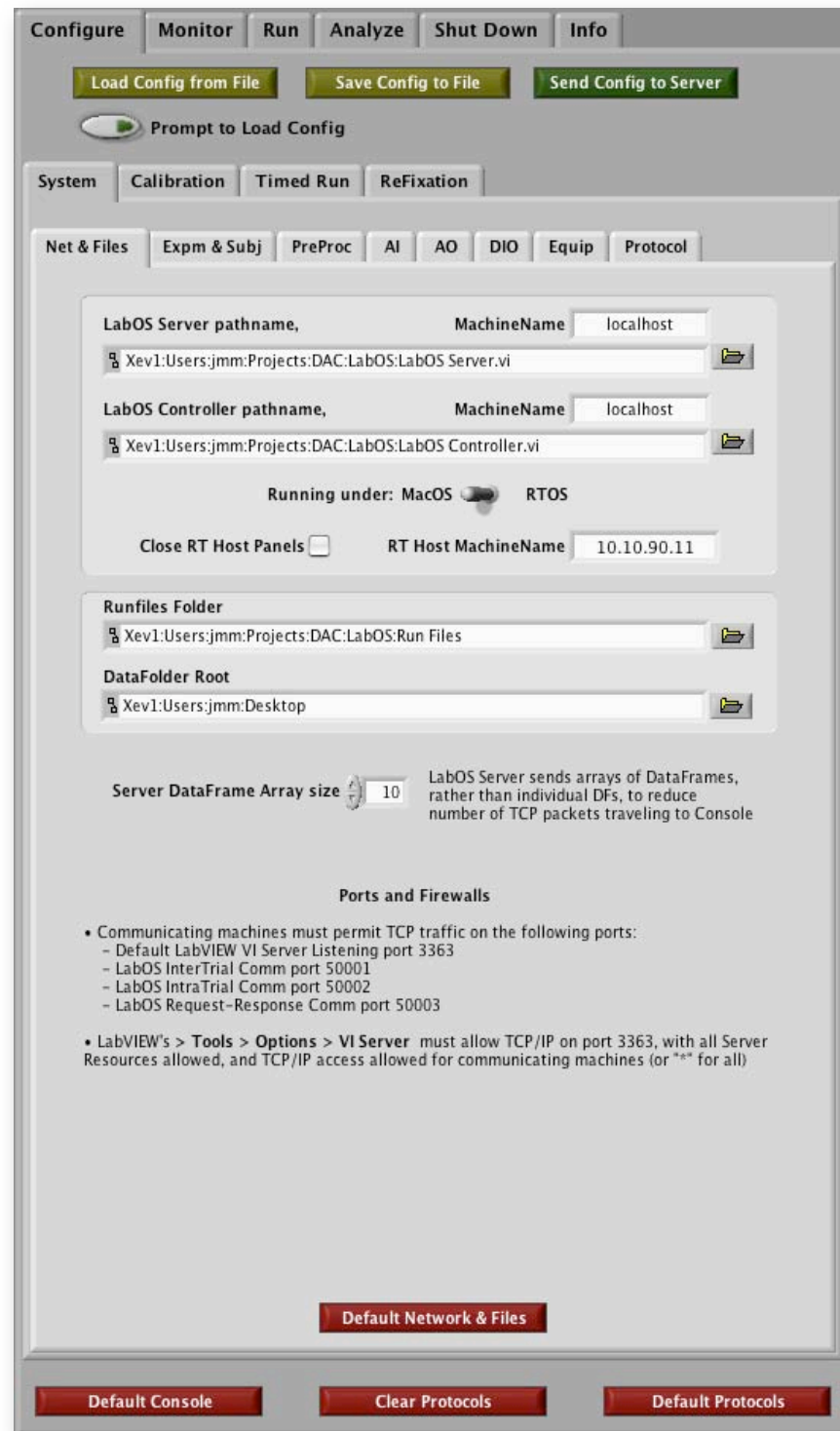


The Operator's Console provides setup and run controls, and a rich and timely view of the ongoing experiment.



Some Other Console Configuration Panels

Choose machines & file locations



This panel is used to configure the LabOS system, including setting the LabOS Server and Controller paths, the Runfiles Folder, DataFolder Root, and the Server DataFrame Array size. It also includes a section for Ports and Firewalls and a Default Network & Files button.

Configure Monitor Run Analyze Shut Down Info

Load Config from File Save Config to File Send Config to Server

Prompt to Load Config

System Calibration Timed Run ReFixation

Net & Files Expm & Subj PreProc AI AO DIO Equip Protocol

LabOS Server pathname, MachineName localhost
Xev1:Users:jmm:Projects:DAC:LabOS:LabOS Server.vi

LabOS Controller pathname, MachineName localhost
Xev1:Users:jmm:Projects:DAC:LabOS:LabOS Controller.vi

Running under: MacOS RTOS

Close RT Host Panels RT Host MachineName 10.10.90.11

Runfiles Folder
Xev1:Users:jmm:Projects:DAC:LabOS:Run Files

DataFolder Root
Xev1:Users:jmm:Desktop

Server DataFrame Array size 10 LabOS Server sends arrays of DataFrames, rather than individual DFs, to reduce number of TCP packets traveling to Console

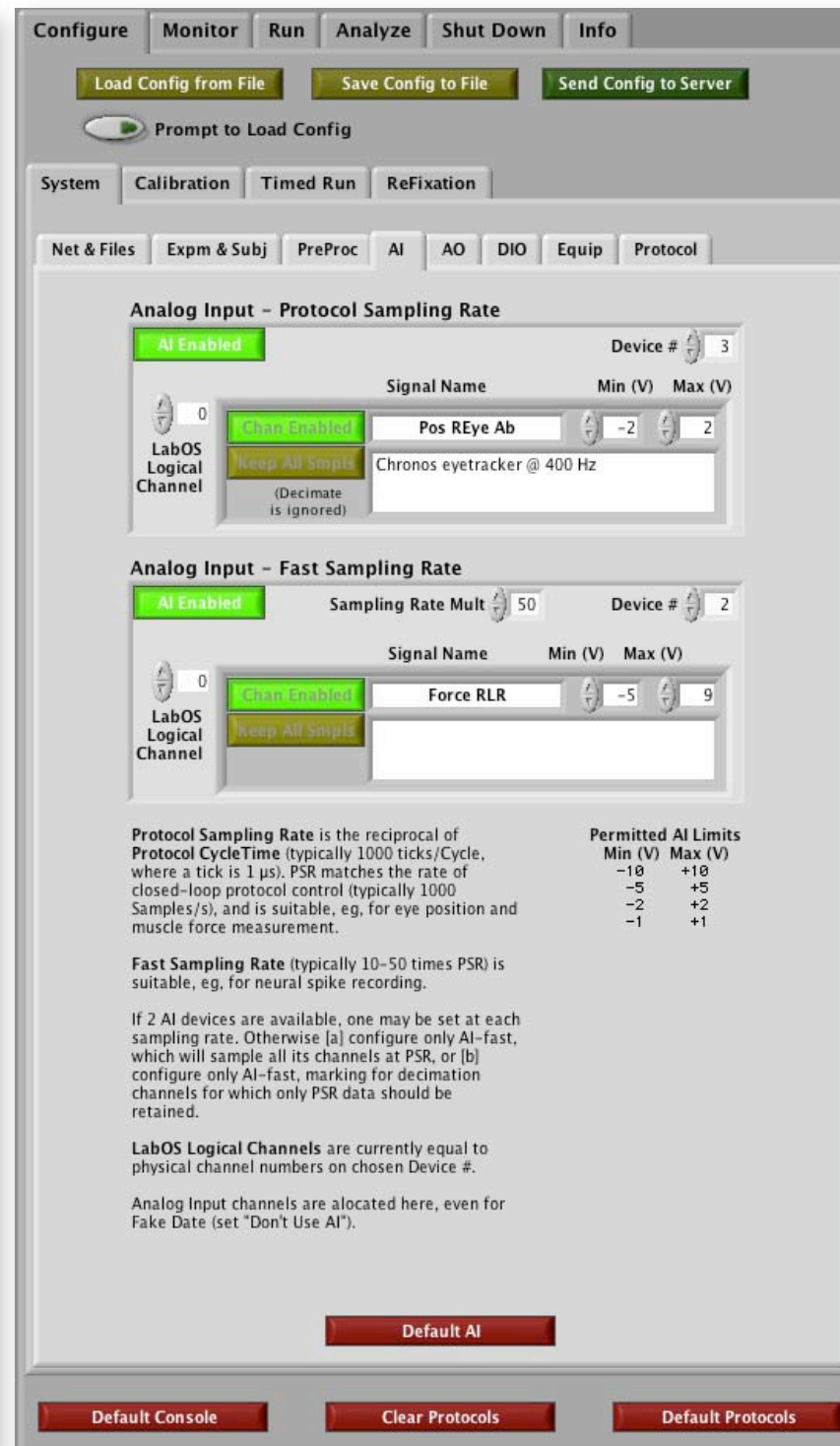
Ports and Firewalls

- Communicating machines must permit TCP traffic on the following ports:
 - Default LabVIEW VI Server Listening port 3363
 - LabOS InterTrial Comm port 50001
 - LabOS IntraTrial Comm port 50002
 - LabOS Request-Response Comm port 50003
- LabVIEW's > Tools > Options > VI Server must allow TCP/IP on port 3363, with all Server Resources allowed, and TCP/IP access allowed for communicating machines (or "*" for all)

Default Network & Files

Default Console Clear Protocols Default Protocols

Setup Analog Inputs



This panel is used to configure analog inputs, including setting the Protocol Sampling Rate, Analog Input - Fast Sampling Rate, and LabOS Logical Channels. It also includes a section for Permitted AI Limits and a Default AI button.

Configure Monitor Run Analyze Shut Down Info

Load Config from File Save Config to File Send Config to Server

Prompt to Load Config

System Calibration Timed Run ReFixation

Net & Files Expm & Subj PreProc AI AO DIO Equip Protocol

Analog Input - Protocol Sampling Rate

AI Enabled Device # 3

Signal Name Min (V) Max (V)

Pos REye Ab -2 2

Chronos eyetracker @ 400 Hz

LabOS Logical Channel (Decimate is ignored)

Analog Input - Fast Sampling Rate

AI Enabled Sampling Rate Mult 50 Device # 2

Signal Name Min (V) Max (V)

Force RLR -5 9

LabOS Logical Channel

Protocol Sampling Rate is the reciprocal of Protocol CycleTime (typically 1000 ticks/Cycle, where a tick is 1 μ s). PSR matches the rate of closed-loop protocol control (typically 1000 Samples/s), and is suitable, eg, for eye position and muscle force measurement.

Fast Sampling Rate (typically 10-50 times PSR) is suitable, eg, for neural spike recording.

If 2 AI devices are available, one may be set at each sampling rate. Otherwise [a] configure only AI-fast, which will sample all its channels at PSR, or [b] configure only AI-fast, marking for decimation channels for which only PSR data should be retained.

LabOS Logical Channels are currently equal to physical channel numbers on chosen Device #.

Analog Input channels are allocated here, even for Fake Date (set "Don't Use AI").

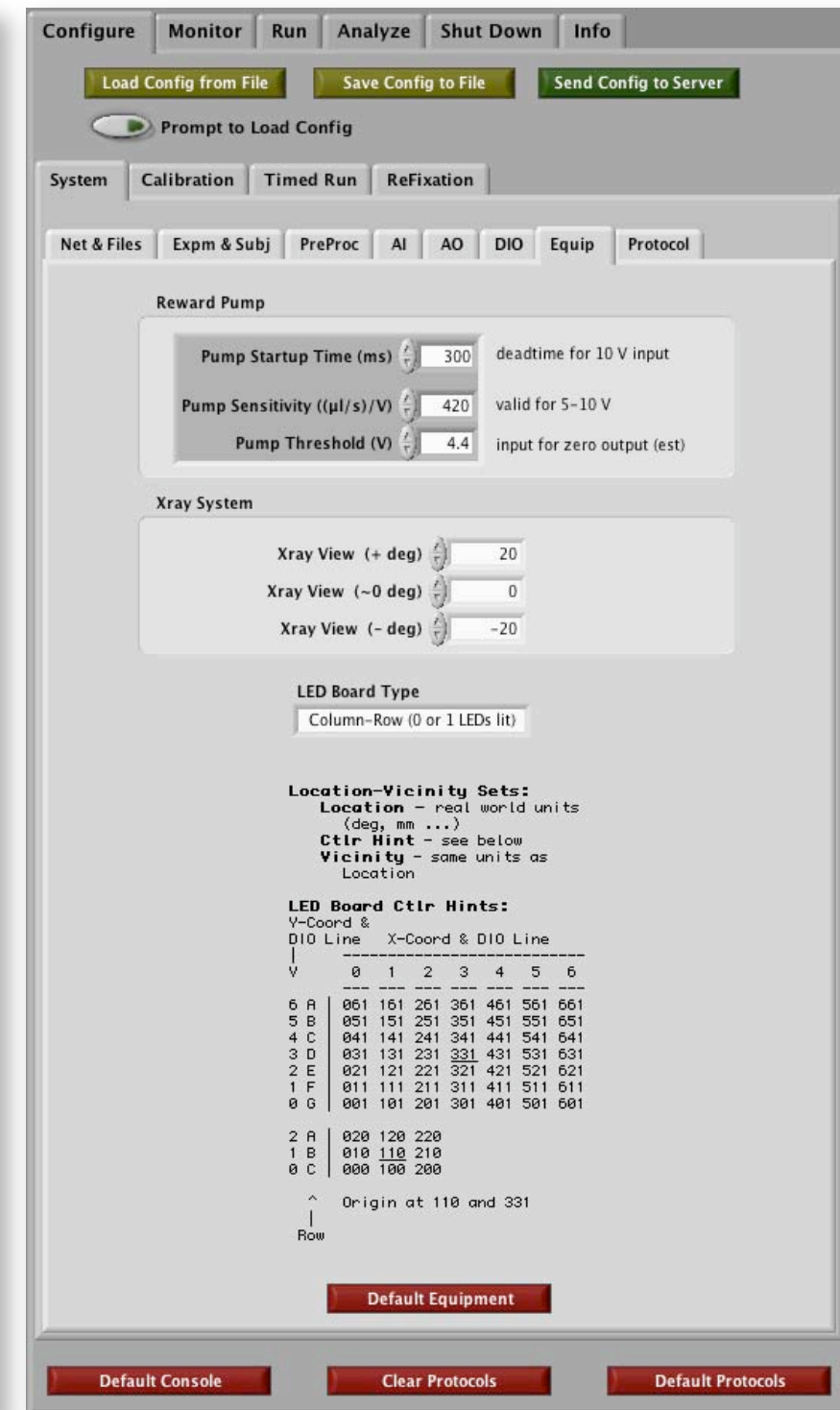
Permitted AI Limits

Min (V)	Max (V)
-10	+10
-5	+5
-2	+2
-1	+1

Default AI

Default Console Clear Protocols Default Protocols

Configure attached equipment



This panel is used to configure attached equipment, including the Reward Pump, Xray System, LED Board Type, Location-Vicinity Sets, and LED Board Ctlr Hints. It also includes a Default Equipment button.

Configure Monitor Run Analyze Shut Down Info

Load Config from File Save Config to File Send Config to Server

Prompt to Load Config

System Calibration Timed Run ReFixation

Net & Files Expm & Subj PreProc AI AO DIO Equip Protocol

Reward Pump

Pump Startup Time (ms) 300 deadtime for 10 V input

Pump Sensitivity ((μ l/s)/V) 420 valid for 5-10 V

Pump Threshold (V) 4.4 input for zero output (est)

Xray System

Xray View (+ deg) 20

Xray View (~0 deg) 0

Xray View (- deg) -20

LED Board Type

Column-Row (0 or 1 LEDs lit)

Location-Vicinity Sets:

Location - real world units (deg, mm ...)

Ctlr Hint - see below

Vicinity - same units as Location

LED Board Ctlr Hints:

V-Coord & DIO Line X-Coord & DIO Line

V	0	1	2	3	4	5	6
6 A	061	161	261	361	461	561	661
5 B	051	151	251	351	451	551	651
4 C	041	141	241	341	441	541	641
3 D	031	131	231	331	431	531	631
2 E	021	121	221	321	421	521	621
1 F	011	111	211	311	411	511	611
0 G	001	101	201	301	401	501	601

2 A 020 120 220

1 B 010 110 210

0 C 000 100 200

Origin at 110 and 331

Row

Default Equipment

Default Console Clear Protocols Default Protocols



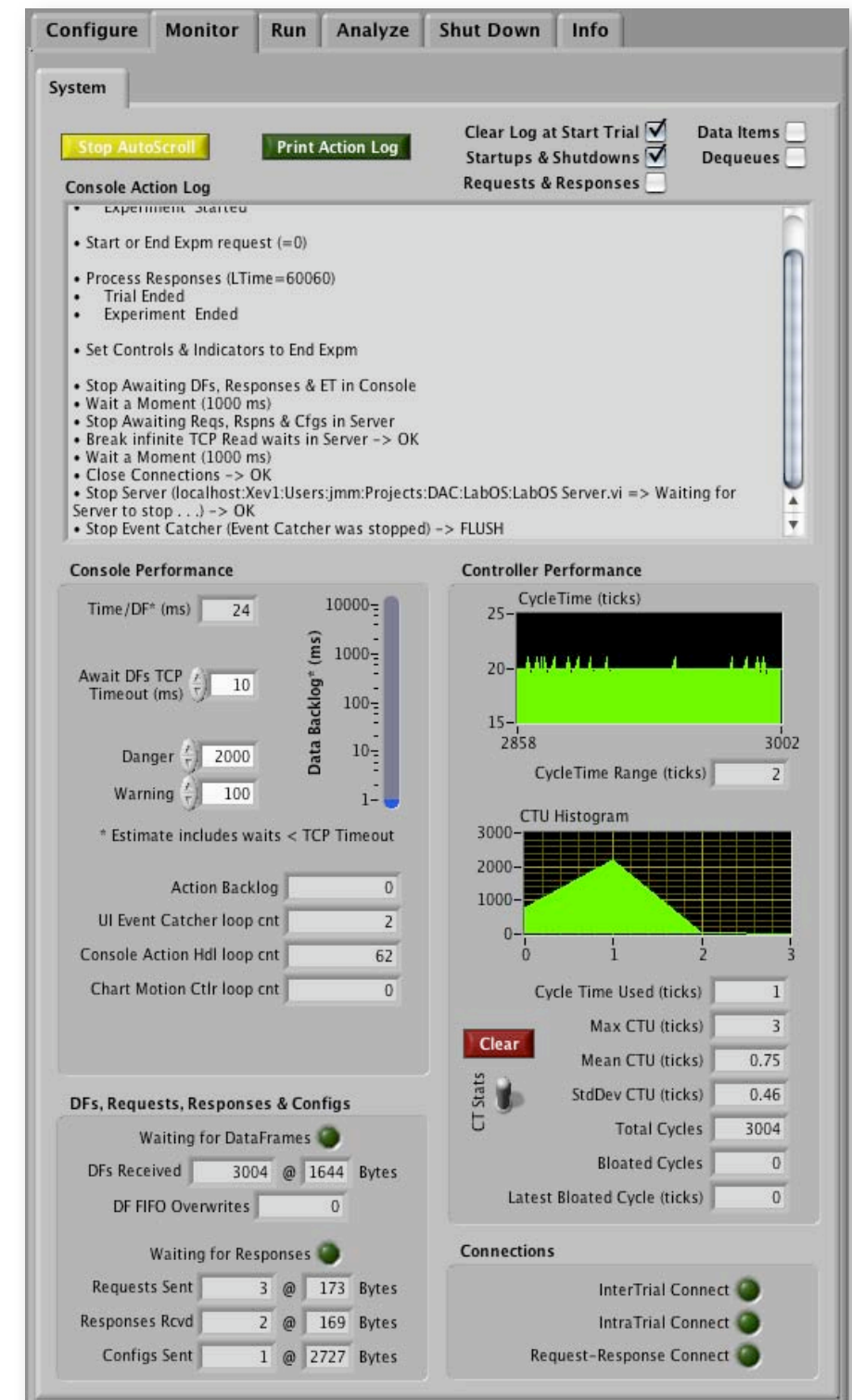
System Monitor Console Panel

● Console activities & performance

- ◆ Monitors TCP connections
- ◆ Logs Console Actions
- ◆ Controls & monitors Console's currency with respect to data flow
 - Charts & indicators may fall behind, but data is never lost

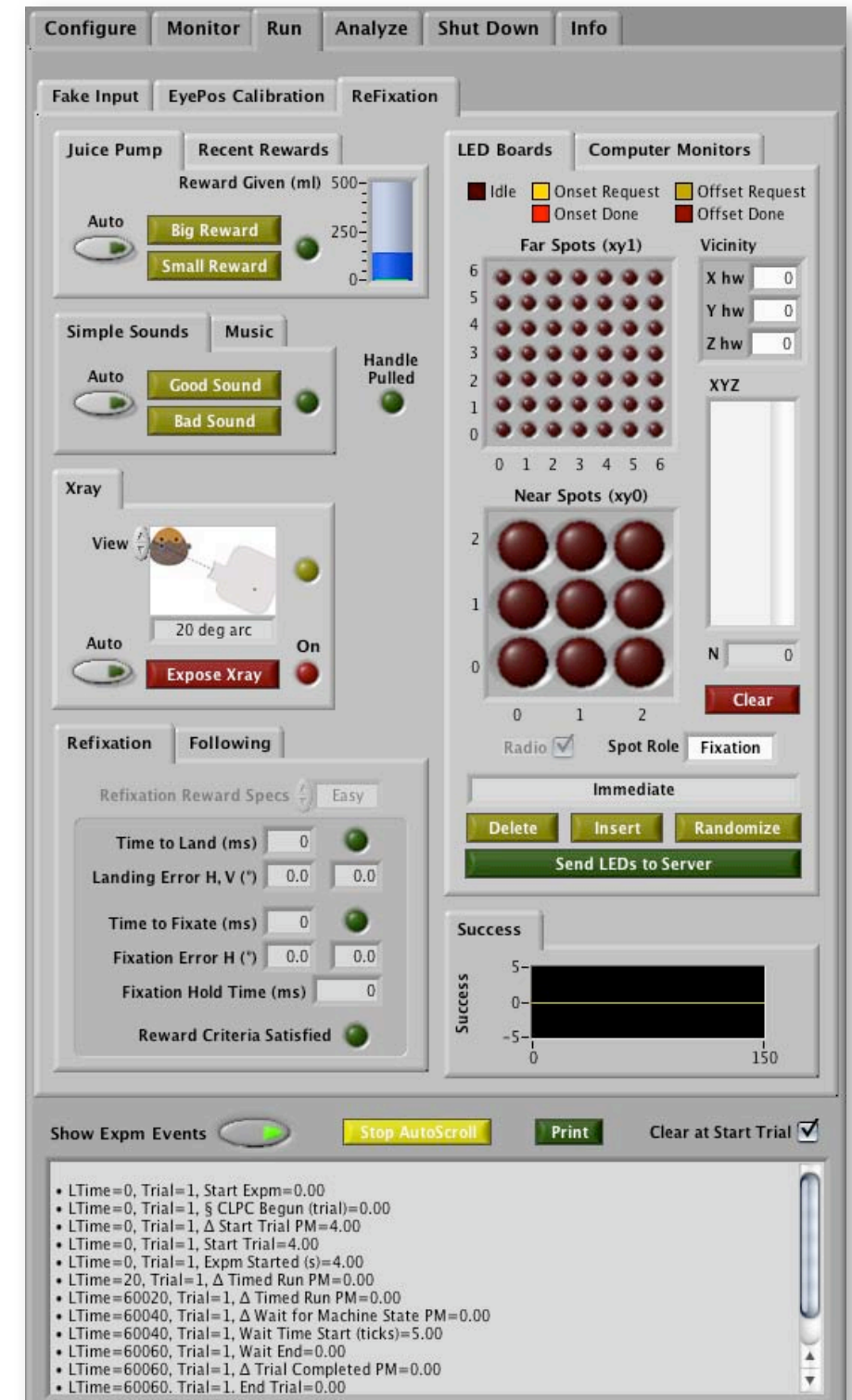
● Controller performance

- ◆ Stability of closed-loop CycleTime
- ◆ Time actually used by Protocol & other code in Protocol Loop (histogram + statistics)



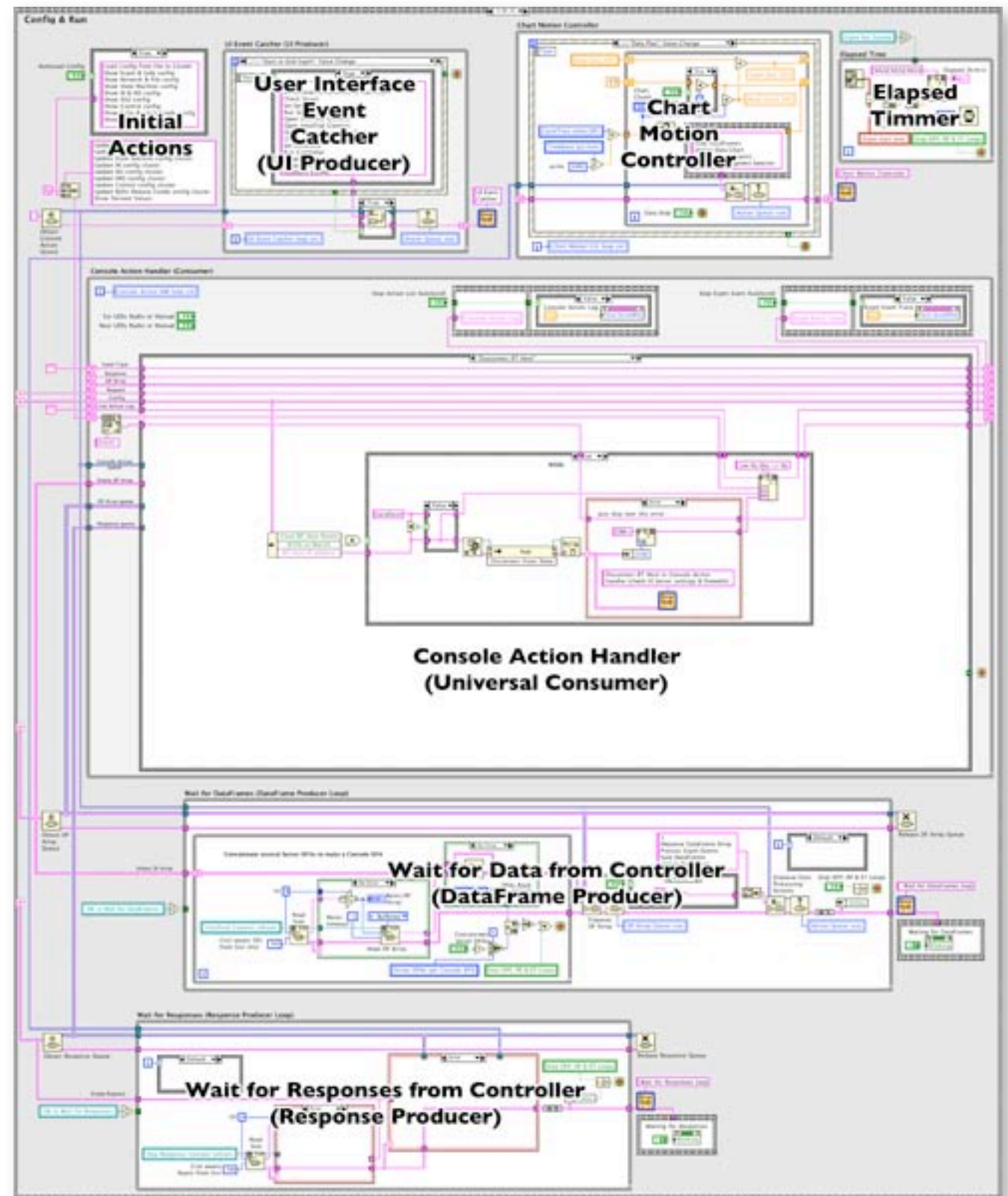
A Runtime Console Panel

- A runtime panel can be built for each experimental paradigm
 - ◆ Modular producer-consumer architecture makes it easy for a programmer to support new experiments
- This panel contains
 - ◆ Reward controls and indicators that show reward history & total fluid given
 - ◆ Controls to position an Xray system and take pictures
 - ◆ Refixation specifications, with indications as each is achieved
 - ◆ Near & far LED board controls & indicators (immediate & batch modes)



LabOS Console Diagram (LabVIEW code)

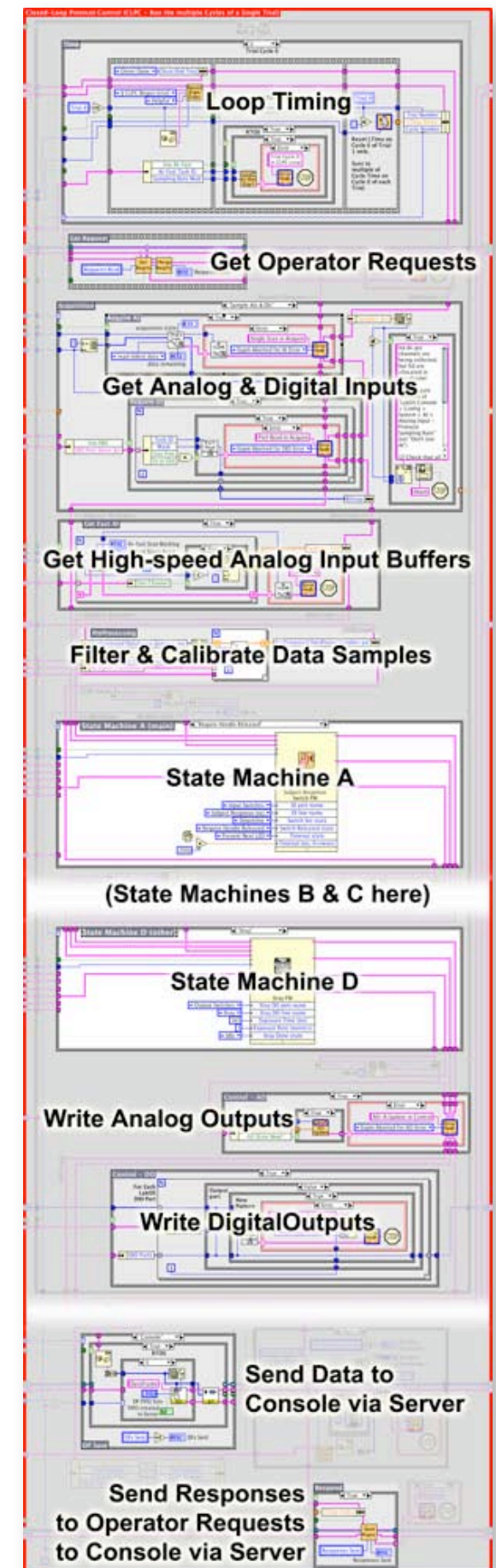
- Producer-Consumer Architecture
- Well-structured & highly modular
 - ◆ Easy for Programmer to add code for controls and viewers to support new experiments



Controller Protocol Control Loop

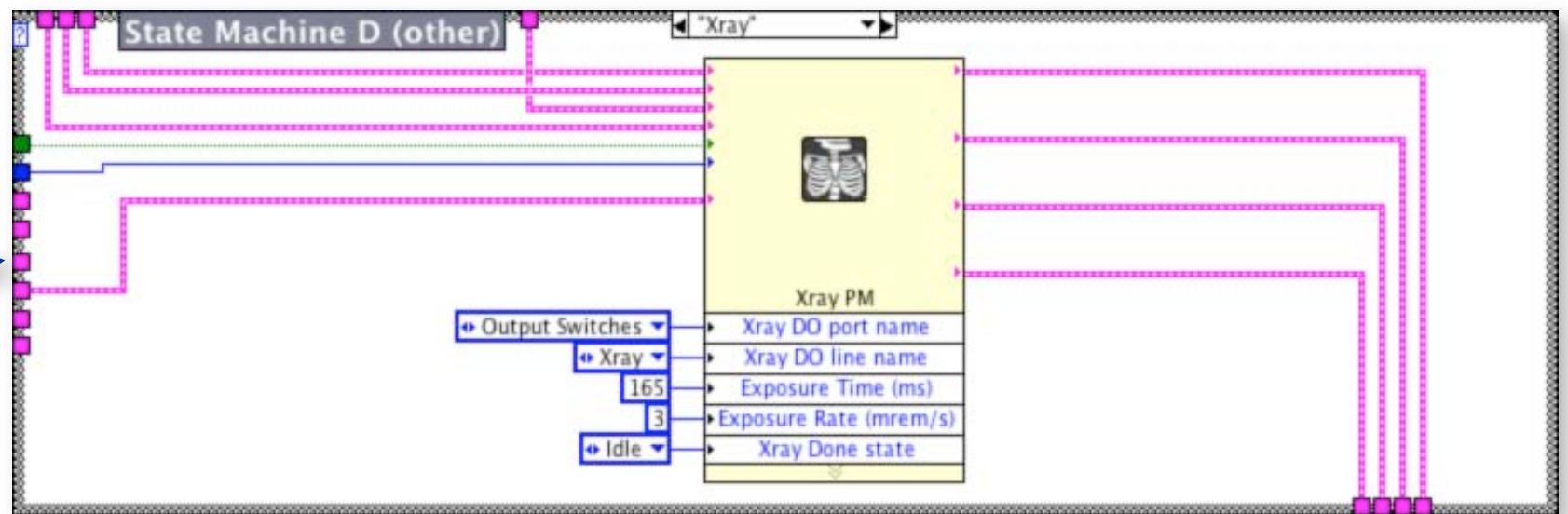
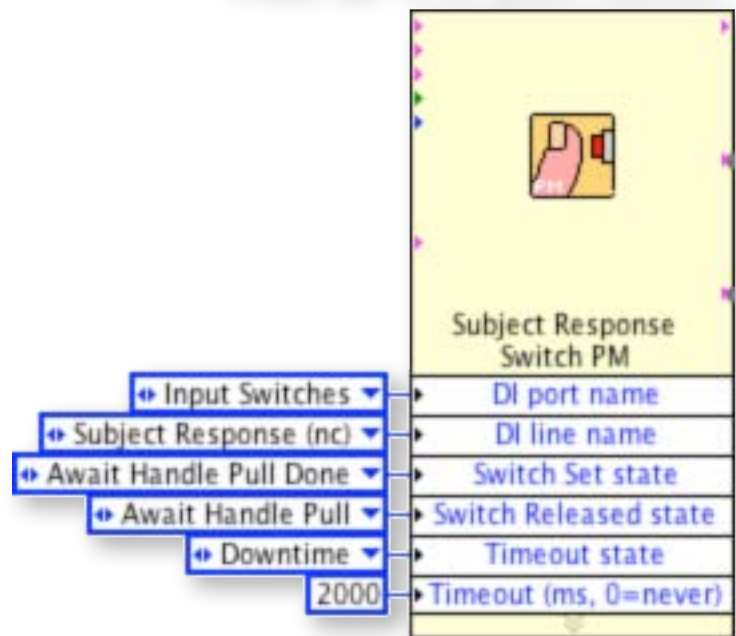
Executes all operations every Cycle (~1 ms)

- ◆ Measures idle time for previous Cycle
- ◆ Gets Operator Requests from Console
- ◆ Read all “protocol sampling rate” analog channels
- ◆ Read all digital inputs
- ◆ Get a buffer of (~50) samples from each “fast sampling rate” channel
- ◆ Optionally filter PSR samples & apply calibrations
 - Protocol modules can accurately evaluate input
- ◆ Run Machine States
- ◆ Write all analog and digital outputs
- ◆ Send a *DataFrame* with all Cycle data to Console
- ◆ Respond to Operator Requests

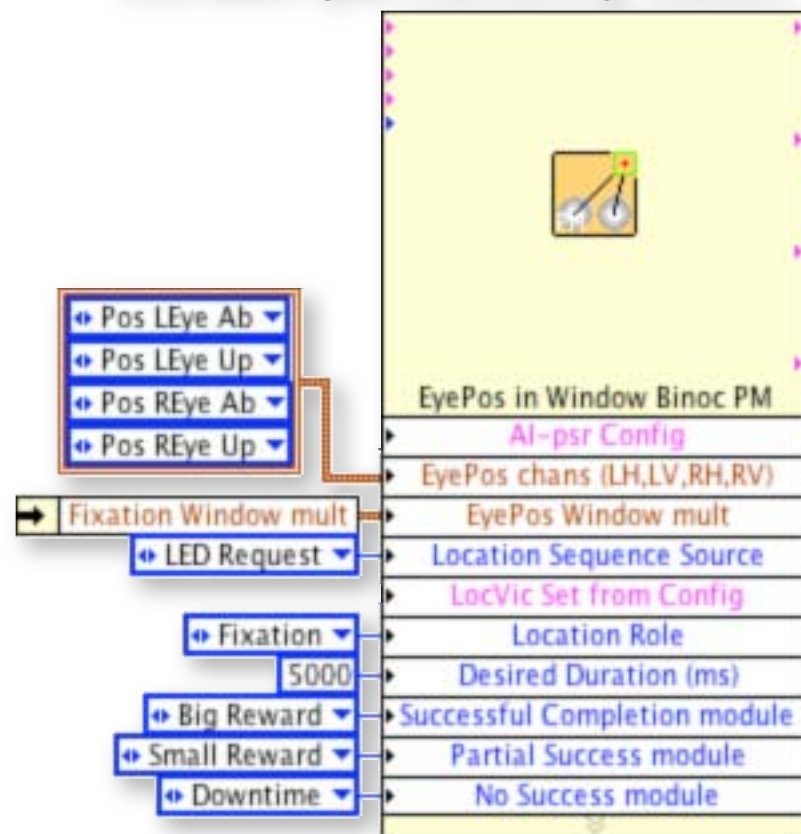


States Consist of Protocol Modules (PMs)

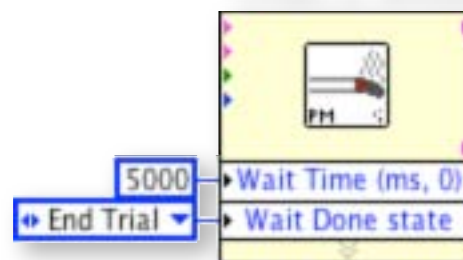
Wait for S to push button



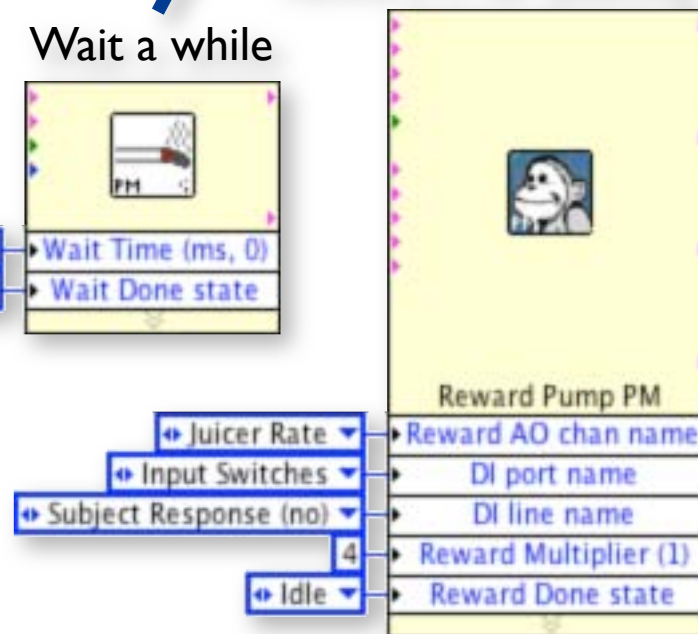
Are both eyes maintaining fixation?



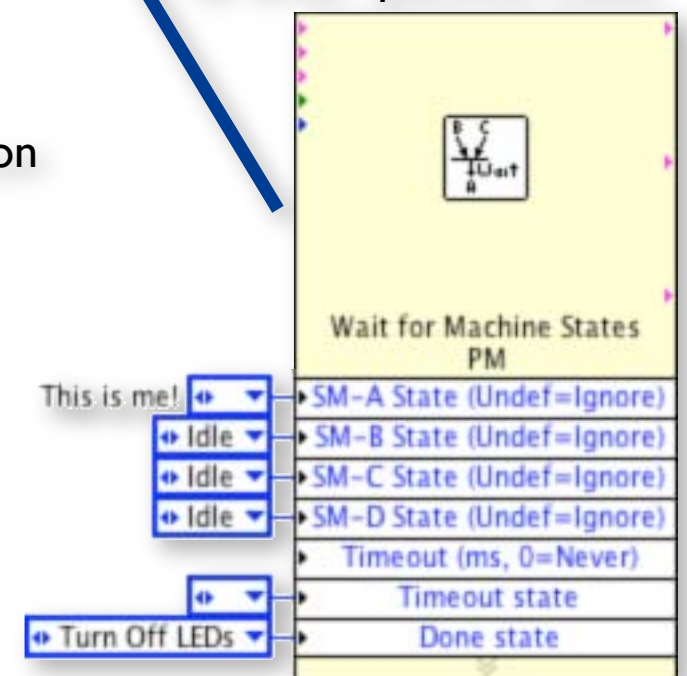
Wait a while



Reward S for pushing button



Wait for other machines to enter specified states



General Design Principles

● Homogeneity

- ◆ Use generalized design patterns whenever possible
- ◆ Avoid special-case optimizations (or describe clearly & encapsulate)

● Understandability

- ◆ Functions must be completely described in terms meaningful to each class of intended users
 - Operator – operates debugged protocols
 - Scientist – creates & parameterizes protocols from modules
 - Programmer – writes new modules
- ◆ Data files must be completely self-describing

● Modularity

- ◆ Keep data formats simple & accessible to external applications
- ◆ Buy, don't build



LabOS Experimental Capabilities

- Experiment control protocols
 - ◆ Support complex experiments that proceed in parallel threads through multiple phases, contingent on incoming data
 - ◆ Manageable by Experimenters
- Closed-loop control. On every *protocol cycle* (~ 1 ms):
 1. Sample all analog & digital inputs channels
 2. Check for Requests from Experimenter
 3. Make sample-dependent protocol decisions
 4. Update all output channels
 5. Make all data available to the Experimenter (as a *dataframe*)
- Designed for Biomedical & Psychophysical Experiments
 - ◆ Modest input channel counts (~ 32 analog & ~ 16 digital)
 - ◆ Modest output channel counts (~ 8 analog channels, ~ 100 digital)
 - ◆ Modest data rates
 - Protocol-controlled, closed-loop cycle time ~ 1 ms
 - Input streaming at ~ 50 KHz (for sampling brief signals like neural spikes)
- Remote monitoring of experiments
 - ◆ Optimized command & data transmission over TCP



DAQ&C System Comparisons (1/2)

- Few DAQ&C systems, provide well-developed software (as opposed to software development tools) for biomedical and similar experimental applications.
- The traditional DAQ&C platform consists of ADC, DAC, DIO & clock cards (“DAQ” cards) in a multitasking PC. The most powerful and elegant of these is *Rex*, which runs under *QNX*, a Unix with realtime enhancements. *Rex* appears capable of reliable, closed-loop control, but cannot be described as deterministic, because there is a possibility of conflict between the high-priority *Rex* process and high-priority system processes. *Rex* requires very specific hardware, cannot do fast sampling, and is strongly programmer-oriented.
- The Tucker-Davis (TDT) systems have a more powerful platform architecture, linking a DSP at the experiment end, to a multitasking PC at the operator’s end. Experimental Protocols, unfortunately, are limited to flowchart-like processing chains, without the control structures central to all modern programming languages.
- Only 2 DAQ&C system run on 2 general-purpose processors, under a time-sharing OS (TSOS) at the operator’s end, and a true realtime OS (RTOS) at the experiment end. *Tempo* is a reasonably capable system, but is specialized for electrophysiology, and yet cannot do fast sampling. It uses a proprietary RTOS, rather than one that is widely supported, and has a restrictive menu-driven and line-oriented user interface.
- LabOS has a superior design and superior capabilities. It supports Protocols as general as those of *Rex*, deterministic closed-loop control with unmatched 1 μ s timing, and fast sampling. Both of its processors support general-purpose programming languages, a level of generality that is reflected in its capabilities. All hardware & software components are well-supported. The modern graphical LabOS user interface is expressive and natural to use.



DAQ&C System Comparisons (2/2)

Experimental protocols should be powerful, and easy to construct and modify. The multi-threaded, multi-phasic nature of complex biological experiments is best expressed as a multi-thread state machine.

Input-output Integration can be loose, as where a DAQ system triggers a display system, or tight, as in closed-loop control.

Realtime Performance: Speed depends on many factors. Determinism - whether realtime scheduling can be reliably maintained - depends on the operating system.

Fast Sampling means ADC rates greater than 20 KHz.

A good **Operator Interface** is highly interactive, richly informative about the ongoing experiment, and allows access to arbitrary data analysis, visualization and other external applications.

Generality - is the system targeted to specific paradigms?

The most capable systems, in each of 3 architectural categories, according to 6 important criteria:

Platform Architecture	Product (& overall rating)	Experimental Protocols	Input-Output Integration	RT Performance	Fast Sampling	Operator Interface	Generality
DAQ cards controlled by TSOS on PC	AD Instr PowerLab	None	Input only	Input only	Yes	Simple graphs, spreadsheets	General
	NIMH Neuro-Cortex	In "C" code, mixed with other functions	?	Only sampling is deterministic	Yes	Simple	Visual neuro-physiology
	NEI LSR Rex	Multithread State Machine; textual lang	Closed-loop control, modest 0.5 ms resolution	Good, but not deterministic	No	Menu driven	Visual neuro-physiology + more
DSP systems linked to TSOS on host PC	TDT PsychRP	Simple templates	Closed-loop control, except functions that execute on PC	Fast & deterministic, except for functions that execute on host PC	Yes	Rich	Psycho-acoustics
	TDT OpenEx & PRvds	Unstructured flowchart-like					Neuro-physiology
RTOS & TSOS on general-purpose processors	Reflective Computing Tempo	Multithread, simple textual lang	Closed-loop control, 1 ms mean response	Proprietary RT kernel	No	Menu driven, Specialized	Electro-physiology
	LabOS	Multithread State Machine; states consist of modules	Closed-loop control on RTOS processor, 1 μ s resolution	Fast & deterministic, widely-used RTOS	Yes	Rich & Interactive	Potentially very general

Weak or very limited, good but with significant limitations, excellent or state of the art



Status as of July 2006

- Eidactics has been awarded an NIH STTR Phase I Grant to support the generalization and commercialization of LabOS
- Beta testing will begin in 2007
- LabOS will subsequently be sold and supported by Eidactics (eidactics.com)

