Control System Integration

Tom Shea

ORNL
Accelerator Control System

• Conveys all monitor, control, model-based, and computed data from all accelerator, facility, safety, and operations subsystems to accomplish supervisory control, automation, and operational analysis.

• Scope extends from the interface of the equipment being controlled through to the designers and operators of the accelerator facility. Sometimes including experimenters.

• Includes all hardware and software between these bounds: computer systems, networking, hardware interfaces, and programmable logic controllers.

• Not restricted to closed loop control systems (a primary focus in this school). In fact, at some accelerator facilities, these systems are treated as black boxes developed by the technical groups.
Responsiveness

- Configuration
- Archiving
- Analysis...

- Control Room
- Physics Apps...

- Digital Signal
  Processing
  Subsystem

Office PC
Network

Supervisory Control
System (SCADA)

Fly by Wire

Nodes are responsive
Network is responsive

Nodes are real-time
Network is real-time
Out With the Old, In With the New, circa 82
The "Standard Model"
Recent Example

High level physics programs that provide model-based control and analysis.

Channel Access and Data Concentrator Clients

Device Concentrator Server
Data concentrators for collecting, transforming, and correlating data from many subsystems.

Channel Access Client

Device-Oriented Protocol

Channel Access Protocol

Engineering Viewing Tools for Machine Monitoring and Control

Channel Access Client

Global Synchronization

Machine Protection System

Fast Feedback

Channel Access Server
Equipment controllers that provide independent operation of local equipment that is tied to all other subsystems through the global control buses

GPIB, Serial, 1553, Control Logix, Modbus, and other proprietary protocols

PLC or other remote I/O with proprietary point-to-point protocols
Evolution of Distributed Systems

- **mainframe/VAX**
  - limited I/O drops lead to long analog signal runs
  - Centralized computing resources

- **Standard Model**
  - distributed I/O controllers
  - shorter analog runs

- **Refinements to standard model**
  - NAD: finer grained distribution
  - Multi-Tier/Middleware
  - New platforms: ATCA, etc...
Functional Interfaces

Communications

- Backplane (Internal)
- Network
- Real-time data*
- Event*
- Reference Clock**
- Machine protect*
- Utility*
- Beamline Device I/O***
- Power
- Environmental

Typically specific to:
* facility
** machine
*** device
Once upon a time, there were transparencies...

- Network links should become redundant
- Smart boxes should share common platform
Motivation for NAD

- **Shared resources** - tightly coupled
  - Tempting to make efficient use of modular infrastructure: pack with channels or share with multiple system types
  - System integration becomes more complex
- **NAD** - loosely coupled
  - limit unnecessary coupling between systems
  - vertically test individual units
  - integrate single units with focussed, well defined interfaces
  - Rapid SNS integration demonstrated success
- Even a subset of the concept can be helpful:
  - i.e. get the mission critical vacuum interlock boards out of the experimental diagnostic crate, or integrate timing decoder on acquisition board in ATCA module
Middleware

• Glue
• From component/device view to physics view
• Uniform access to multiple information sources (front end computers, online model servers, databases...).
• Could be soft real-time
• Complete solutions used heavily in the financial industry, less so in accelerators (but watch LHC)
• But many purpose-built implementations in use:
  • adapters
  • aggregators of requests (multiple clients) or responses (device collections particularly to support distributed high channel count systems)
  • correlators
RHIC BPM System Overview

VME chasses (20 in RHIC, 2 in ATR)

IFEs (digital+analog)
Tunnel/alcove/eqpt bldg

728 BPM “planes”
Matlab MiddleLayer

High Level Matlab Applications
  (scripts and functions)

Matlab Middle Layer

Matlab to EPICS
  (MCA, LabCA, SCAIII)

Accelerator Toolbox
  (AT - Model)

Accelerator Hardware

AT Server
  (Simulator)
Physical Packaging

• Field replaceable units can be:
  • Modular
  • Self contained - i.e. rackmount, wall mount
  • Integrated - i.e. PS controller
Packaging issues

- Power
- Cooling
- Shielding
- Connections
- Partitioning
- Standardization
- Availability/MTTR
Modular Packaging

- VME/PCI/cPCI
- VXI/PXI
- ATCA (shown)
Self contained

- Rack mount industrial PCs acting as NADs
  - could choose to stock prebuilt units as spares
  - changing I/O boards could lead to higher MTTR
- Chassis containing more integrated electronics
- RHIC BPM (19” rackmount - in house)
- Libera (19” rackmount - vendor supplied)
Communications
Communications Trends at Accelerator Facilities

- End of traditional parallel backplane bus paradigm (Announced every year since ~1989)
  - VME-PCI still there
  - watch PCI Express, RapidIO, ATCA
- Commercial networking products
  - DAQ 94’ Conference: ATM, DS-Link, FibreChannel, SCI
  - Today: Gigabit Ethernet (1, 10, 30 GB/s)
- The ideal processing / memory / IO bandwidth device
  - The past: Transputers, DSPs
  - Today: FPGAs - Integrates receiver links, PPC, links, PPC, DSPs, memory........
- Point-to-point link technology
  - The old style: Parallel Copper –Serial Optical
  - The modern style: Serial Copper –Parallel Optics, >3Gb/s today, 10Gb/s in demonstration
Latency and Throughput

- Gigabit Ethernet
- Fast Ethernet
- USB 1.1
- USB 2.0
- IEEE 1394a
- PCI Express/PCI Express (x4)
- PCI/PCI (32/33)
- VME/VXI
- GPIB (HS 488)
- GPIB (488.1)

Point to point serial
Intra-chassis Standards

- CAMAC
- VME/VXI
- PCI/PXI
- IP, PMC
- PCIe
  - multilane serial
- ATCA
  - serial fabric
  - PCIe, ethernet, etc

<table>
<thead>
<tr>
<th></th>
<th>ATCA</th>
<th>PCI (long)</th>
<th>VME 6U</th>
</tr>
</thead>
<tbody>
<tr>
<td>Board Area</td>
<td>995</td>
<td>316</td>
<td>373</td>
</tr>
<tr>
<td>cm²</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Power Watts</td>
<td>200</td>
<td>10/25</td>
<td>30</td>
</tr>
<tr>
<td>Watts</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bandwidth</td>
<td>20</td>
<td>4.3</td>
<td>2.4</td>
</tr>
<tr>
<td>I/O Gb/s</td>
<td></td>
<td>66MHz</td>
<td>VME</td>
</tr>
<tr>
<td>full duplex</td>
<td></td>
<td>64 bits</td>
<td>2eSST</td>
</tr>
<tr>
<td>Front panel</td>
<td>30 * 2</td>
<td>8 * 1.2</td>
<td>21.5 * 2</td>
</tr>
<tr>
<td>H * W cm</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Component Height</td>
<td>21.33</td>
<td>14.48</td>
<td>13.72</td>
</tr>
<tr>
<td>mm</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
UDP over Ethernet

- TCP: http, EPICS Channel Access
- UDP: deterministic communication, streaming multicast
- Either one: COTS routers, diagnostic tools
- inteded for implementation with processor, but simple UDP demonstrated in hardware
Latency test for user mode process

Measure time difference
UDP Send – Ack Response
At Master using high res. timer

300 usec round trip latency.
one hit at 1.7 msec
Point to point, no switch
2002 Vintage Dell running
Standard Windows 2000

Round Trip with 2005 vintage Tyan, Gbit ethernet switch

Packet size (bytes) vs. micro seconds
Response Time

• What affects response time?
  • Network collisions (use switches/synchronization)
  • Thread Quantum (improved with preemptive interrupt capable OS: RTOS, Solaris, Windows, and new Linux Kernels >=2.6)
  • Context Switching times (10usec-100usec)
  • Packet size (Minor for small packets)
• Example using RTOS:
  • ALS orbit feedback system demonstrates negligible effect of UDP packet jitter @ 100Mb/s. Currently 1.1kHz rate. Expect to achieve 5 kHz rate @ Gb/s
Reflective Memory Example

- Also known as Replicated Memory
- Each participant in the reflective memory network has a reflective memory module.
- Each module can be written and read as simple memory
- Property:
  - Anything written to a location in one reflective memory module appears (after a loop transit time) in the same location in all attached modules.
  - Transfer rate 29.5 Mbytes/second
- All Nodes see the same memory image
- Designed for Real-time Performance – Latency is minimized
- Typical implementation: Commercial PMC card on VME carrier or VME CPU board
- Calculated loop settling time = 21.4 usecs (22 nodes and 1200 meters of fiber)
Timing/Event Systems

- Functions:
  - Coordination of actions
  - Correlation of data

- Other aspects to consider:
  - Multiuser operation
  - Frequency change if beam synchronous

- Scalability: information should reach ALL devices

- Implementations
  - FNAL/AGS/RHIC/SNS separated function
    - Event Link, TClock
    - RTDL, MDat
  - SLS, Diamond: Events/info on Ethernet PHY

- Other options
  - UDP broadcast
  - Direct fiber
  - “tagged” RF reference clock
Typical Event System

The system provides:
- global distribution of events to all systems that have a receiver
- trigger and gate signals to hardware
- synchronized timestamp facility
- software sequencing by triggering channels to process from events
- software can be used to send events

When an event code is received the receiver can:
- output a pulse, of specified delay and width
- trigger a software action (process an EPICS record)

Each event receiver can be programmed to respond in a different way to the same event code.

The stimulus to send an event can be:
- pulse on a hardware input
- software event (write to a register)
- an entry in an event playback RAM.
Event Timeline

Real-Time Data Link (RTDL)

Time Critical Events, (soft events disabled)

Informational Events, non critical timing

RF & High Voltage Events
Cycle Start
Beam On

Snapshot 1Hz, 6Hz etc...

MPS FPL
MPS FPAR
System xxx Trigger Events
End Injection
Extraction Kicker Charge

Beam On Range
Beam On

Allowed Range for Variable Triggers

beam accumulation

Machine

Anytime

Anytime

Line-Synch Reference Clock
+60 Hz Zero Crossing

-60 Hz Zero Crossing

0 1 ms 2 ms 3 ms 4 ms 5 ms 6 ms 7 ms 8 ms
Example receiver

For <<1% utilization of modern FPGA, get events within signal processing system.

This example is in-house standard. Trend is to adopt standard PHY.
Multimode Operation

- AKA: Multimode, Multiuser, Pulse to Pulse Modulation, Flavors, ...
- Different Beam Parameters
- Different feedforward tables
- Different results updated
Machine Protection
Motivation

Livingston type plot:

- LHC injection (12 SPS batches)
- LHC top energy
- SPS fixed target
- SPS batch to LHC
- ISR
- HERA
- TEVATRON
- SPS ppbar
- SNS
- LEP2
- Factor ~200

Energy stored in the beam [MJ]

Momentum [GeV/c]

courtesy R. Assmann
Issues

• Subsystem drives MPS input

• DSP detects loss of control

• Differential current measurement detects beam loss

• Subsystem responds to MPS trip

• Communicated on timing or real-time data broadcast system

• Circular buffers
Interface to MPS

Typical MPS Input

Typical MPS Fault Drive Circuit
Utility

- Temperature monitor
  - for fault monitoring (filter change time)
  - for correction of thermal effects
- Fan Speed
- Power supply current and voltage
- Heartbeat
  - monitored
  - optional watchdog timer for automatic reset
- Remote Power and Reset
  - Personnel access restrictions
- Facility Size
  - MTTR
Safe Reconfiguration

- avoiding the “turn antenna away from earth” command.
- protected boot memory containing communication code
- permanent communication subsytem (tiny IOC on a card, hard core on FPGA,...)
- Out of band communication as an option - but may add interconnect or cable in some cases
Remote Debugging

by example

- RHIC BPM: Overview
  - Most electronics in radiation area
- SNS LLRF: In depth
  - SNS Equipment gallery put under access restrictions during commissioning
Event links and RTDL

Parameter editing Tool (PET)

Instrumentation system

ADO

Snap-ADO

Snap-memory

VME memory

Snap-cdev

manager

app 1

app 2

Engineering application

RHIC Debugging Tools
Interface to LLRF Module

- Register based
  - Random read and write access to ease debugging
    - No "don't write during …" conditions that require tricky timing.
    - No write-only registers that one cannot verify.
- Message based interfaces were avoided
  - Because they are harder to debug.
  - Also harder to implement in an FPGA, requiring parser.
  - One can easily trigger VME/VXI/PCI backplane analyzer on register access. Messages are harder.
Register Documentation

Register † LLRF_DDS_FREQ

The offset frequency for the on-board Direct Digital Synthesizer (DDS). This is a signed 16-bit number that adjusts the frequency of cavity operation, relative to the Master Oscillator, in the range of ±625 kHz. One bit corresponds to 19.073486328125 Hz exactly (assuming the RF sampling clock is a perfect 40 MHz).

Register LLRF_STATUS

Read-only bit map of module status.

<table>
<thead>
<tr>
<th>D15</th>
<th>D14</th>
<th>D13</th>
<th>D12</th>
<th>D11</th>
<th>D10</th>
<th>D9</th>
<th>D8</th>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>CLP</td>
<td>FLT</td>
<td>LRC2</td>
<td>LRC1</td>
<td>PLL</td>
<td>IL</td>
</tr>
</tbody>
</table>

CLP: Latched indicator of occurrence of any premature termination of RF due to output magnitude clipping fault. Cleared at beginning of each pulse.

FLT: Latched indicator of occurrence of any premature termination of RF due to faults within the pulse. Cleared at beginning of each pulse.

LRC2, LRC1: FCM-specific readout of the configuration switches that define its position. 0=Left, 1=Center, 2=Right, 3=Invalid. Non-FCM systems read 0.

PLL: Instantaneous readout of the same signal described in the PLL bit of the
Perl scripts convert Verilog register definitions into C++ include file, EPICS Database config, …
Debugging Registers: Console

- Via serial line or "telnet"
# Debugging Registers:

- **Note:**
  
  Our hardware returns value **0xBAD** for undecoded registers (pullups & downs on data lines in case FPGA doesn't drive them)
End-User Register Access

SCL 01a DDS Control

Frequency Agile DDS Mode
Freq.

To locate the cavity resonance frequency in case of severe detuning:
1) Establish a small, open loop amplitude.
2) Try to maximize the cavity field via the DDS frequency.

That DDS frequency is the current resonance error. Use the tuner or RCCS to correct it.

Amplitudes

(0.00 kHz)

Forward Reflected Cavity
(horizontal axes in microsecs)
Other Debug Tools

• EPICS Sequencer and Database
  – Online, remote access to internal state
  – View the "raw" value behind some displayed data
  – Timestamps
  – Anything can be displayed on "Strip chart", or added to an archive tool for later analysis - good for infrequent events and unanticipated correlations

• Custom C/C++ Code
  – Access to internal data must be specifically added to the code
    • "Report" methods
    • Debug flags
    • Time stamps
Source Code Control
Version Control Benefits

• Serves as repository
  – Develop some bugfix on laptop, merge that into copy on the main development machine
  – Deploy "latest" version onto linac server

• "Roll back to state of January 18, 2006"

• View history of changes, compare different versions
  – "When did we add this behavior?"
  – "How was this handled 2 years ago?"
CVS - Concurrent

• A free, open source version control system
  − Available for every operating system
  − Stable
  − Command-line, Emacs, Eclipse, …

• Handles text very well
  − Plain ASCII or LaTeX documents
  − EPICS Sequencer and Database sources
  − C/C++
  − Verilog/VHDL
  − Front-end computer startup files

• For binary files, only date & comments available, no comparisons possible
  − Images, FPGA bitfiles, LabVIEW, …

• Old
  − "subversion" might be better at handling directories
FPGA Firmware Handling

• Verilog sources are in CVS
  – Full benefit of version comparisons

• Bitfiles also in CVS as 'binary'
  – No insight into changes, but since each "place-and-route" creates different bitfile, it's good to keep a copy of the specific bitfile

• Front-end computer programs FPGA
  – Loads bitfile via network

(For machine protection related FPGA, bitfile is in local EEPROM)
Configuration Control
IOC Application which pulls Configuration file to IOC from RDB.
Uses HTTP Socket Library TCP/IP to connect to Web server.

Web Server uses standard RDB connectivity:
PHP, JSP, ASP

Files pulled from here.

Portion of stored procedure used when IOC Configuration File generation UI creates a new configuration file for a given IOC. These selects confirm active versions which then dictate data to be used to create the file stored for use.
Example: LLRF Multiplicity

• Almost 100 SNS LLRF Systems, handled by ~50 front-ends
  - As different as warm vs. super-conducting cavities

• One source base for all of them
  - Differences handled by configuration settings
  - If possible, startup files and overview displays script-generated from central system info.
Configuration File Strategy

- Track changes to configuration files
  - Who made the change
  - When was the change made
  - Why was the change made
- Restore past configuration files when necessary
- Configuration consists of structure and data
  - Structure (collection of properties that describe the device) is typically common across devices of a specific type
  - Data typically varies for each device and represents the values for a device’s properties
  - Structure is associated with a configuration’s major version number and data is associated with the minor version
Configuration File
Storage/Retrieval

Configuration Type: [RPC]  Device Overview | Configuration Templates | Batch Import | Logout

Class:  DTL_Diag:IOC_BCM622  Warning: This is not the default device for your IP Address!

25 Configurations  Display [ ] more [ ] Page [ ] of [ ]

<table>
<thead>
<tr>
<th>Active</th>
<th>Configuration</th>
<th>Date</th>
<th>Author</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>☐</td>
<td>17.22</td>
<td>May 01, 2007 16:14</td>
<td>900688</td>
<td>Batch upload of configuration files.</td>
</tr>
<tr>
<td>☐</td>
<td>17.21</td>
<td>May 01, 2007 15:48</td>
<td>900688</td>
<td>Demo Batch upload of configuration files.</td>
</tr>
<tr>
<td>☐</td>
<td>17.20</td>
<td>May 01, 2007 11:36</td>
<td>900688</td>
<td>Test batch upload of configuration files.</td>
</tr>
<tr>
<td>☐</td>
<td>17.19</td>
<td>May 01, 2007 11:56</td>
<td>900688</td>
<td>Test batch upload of configuration files.</td>
</tr>
<tr>
<td>☐</td>
<td>17.18</td>
<td>May 01, 2007 10:46</td>
<td>900688</td>
<td>Batch upload of configuration files.</td>
</tr>
<tr>
<td>☐</td>
<td>17.17</td>
<td>May 01, 2007 10:44</td>
<td>900688</td>
<td>Batch upload of configuration files.</td>
</tr>
<tr>
<td>☐</td>
<td>17.16</td>
<td>May 01, 2007 10:43</td>
<td>900688</td>
<td>Batch upload of configuration files.</td>
</tr>
<tr>
<td>☐</td>
<td>17.15</td>
<td>May 01, 2007 10:42</td>
<td>900688</td>
<td>Batch upload of configuration files.</td>
</tr>
<tr>
<td>☐</td>
<td>17.14</td>
<td>May 01, 2007 10:41</td>
<td>900688</td>
<td>Batch upload of configuration files.</td>
</tr>
<tr>
<td>☐</td>
<td>17.13</td>
<td>May 01, 2007 10:40</td>
<td>900688</td>
<td>Batch upload of configuration files.</td>
</tr>
</tbody>
</table>

Activation notes:

BCM Diagnostic Device Active Configurations:

<table>
<thead>
<tr>
<th>Device</th>
<th>Active Configuration</th>
<th>Date</th>
<th>Editor</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCI_Diag:IOC_BCM02</td>
<td>14.22</td>
<td>May 01, 2007 16:14</td>
<td>900688</td>
<td>Batch upload of configuration files.</td>
</tr>
<tr>
<td>DTL_Diag:IOC_BCM622</td>
<td>17.22</td>
<td>May 01, 2007 16:14</td>
<td>900688</td>
<td>Batch upload of configuration files.</td>
</tr>
<tr>
<td>LBBT_Diag:IOC_BCM05</td>
<td>2.1</td>
<td>May 31, 2006 14:28</td>
<td>9P3</td>
<td>Original Configuration taken during initial May 2006</td>
</tr>
<tr>
<td>LBBT_Diag:IOC_BCM14</td>
<td>11.1</td>
<td>May 01, 2007 11:44</td>
<td>900688</td>
<td>Test new activation code.</td>
</tr>
</tbody>
</table>
Implementation
Choices

- Human
- Commercial/Scripted High Level
- High Level Application (Multiuser OS)
- Low Level Application (RTOS target)
- Embedded (DSP, limited OS)
- FPGA
- ASIC
- Analog

“Some folk built like this, some folk built like that
But the way I'm built, you shouldn't call me fat
Because I'm built for comfort, I ain't built for speed...”

- Willie Dixon
Flexibility or Performance?

• Flexibility in the form of
  − Rapid development, independent testing, remote access, online changes, rich set of debug tools

… often differs from …

• Performance
  − Fast startup times, short response times, deterministic "real time" behavior.
SNS LLRF Choices

- Software based a control system **framework** (Experimental Physics and Industrial Control System, EPICS)
- Matlab scripts for test & development of algorithms
- Front-End computer uses EPICS **State Machine Tool** for automation, and "runtime **Database**" for data flow.
- C/C++ driver code
- **Fast Feedback** (~40MHz) and interlocks in **Verilog**, VHDL, AHDL. Several Iterations
- Hardware as simple as possible: Analog filtering, ADCs/DACs, then **FPGA**
Operator Interface: Display Manager

- Drawing package for
  - Placing labels, text-monitors, meters, ... on a screen
  - Connecting them to online Process Variables
  - Display panel Configuration instead of coding
Overall SNS LLRF

• Requirements change, so **Flexibility** is key.

• Resonance Error computation, feedback loop setup, ...
  − If possible, first developed in Matlab
  − Then implemented on Front-End as State Machine or EPICS Database
  − **If necessary**, later moved into custom C++ driver code, or even FPGA
State Machine (EPICS "Sequencer")

• Used for automation whenever possible

• "On Demand" tasks, response times of .1 to 1 sec

• Safest and most flexible tool
  - Runs on host as well as front-end
  - Start/Stop/Update without front-end reboot
EPICS "Database"

• Used for steady-state control, data flow.

• Full remote access to any detail.

• Limited online changes.

• "Records", building blocks
  – Read input, computation, write output, …

• Database Engine handles
  – Periodic or event-driven scanning
  – Time stamps(!)
  – Check of alarm limits
  – Publication of data in "Process Variables"

• Response times of millisecs possible, or more than 10000 records per front-end computer.
Custom Low Level Code

• Custom C/C++
  - When required for higher performance, or interrupt service routines, low-level access to custom hardware
  - Usually no online changes.
  - It's really hard to understand, extend, debug somebody else's custom code
  - Debug tools vary with operating system
    • SNS, using vxWorks5 with Linux hosts has currently no online source-level debugger….

• FPGA
  - Same problems as custom C/C++ code
    • Probably even more so, since HDL is a "code", but often not implemented by software engineers.
  - Good simulation and offline analysis tools but online debugging limited to scope, signal analyzer.
SNS LLRF Changes in early 2007

- Improve performance of tested algorithms by converting Sequencer (State Machine) code and Database Records to C++

Before

- C++ LOC: 2209
- Database Recs: 253
- Sequencer LOC: 2581

Now

- C++ LOC: 3600
- Database Recs: 217
- Sequencer LOC: 1900
Alternative DSP Implementations

- High level environment
- Commercial
- Physics application framework
- Within control system toolkit
- Vertically integrated commercial products
% Get the vertical orbit
Y = getam('BPMy');

% Get the Vertical response matrix from the model
Ry = getrespmat('BPMy', 'VCM'); % 120x70 matrix

% Computes the SVD of the response matrix
lvec = 1:48;
[U, S, V] = svd(Ry, 0);

% Find the corrector changes use 48 singular values
DeltaAmps = -V(:,lvec) * S(lvec,lvec)^-1 * U(:,lvec)' * Y;

% Changes the corrector strengths
stepsp('VCM', DeltaAmps);
LabVIEW FPGA with EPICS

LabVIEW FPGA development environment

LabVIEW runtime

Shared Memory

FPGA based DAQ board (~ 40 MHz limitation)

IOC (database, CA)

Channel Access

DBD and DBD files

config

data

ReadWrite() WriteData() SetInterrupt()

UserData() WaitForInterrupt() GetIndexByName()