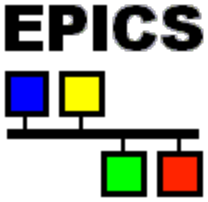
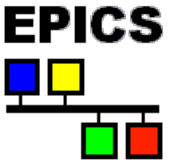


Using COTS Hardware with EPICS Through LabVIEW – A Status Report

EPICS Collaboration Meeting

Fall 2011

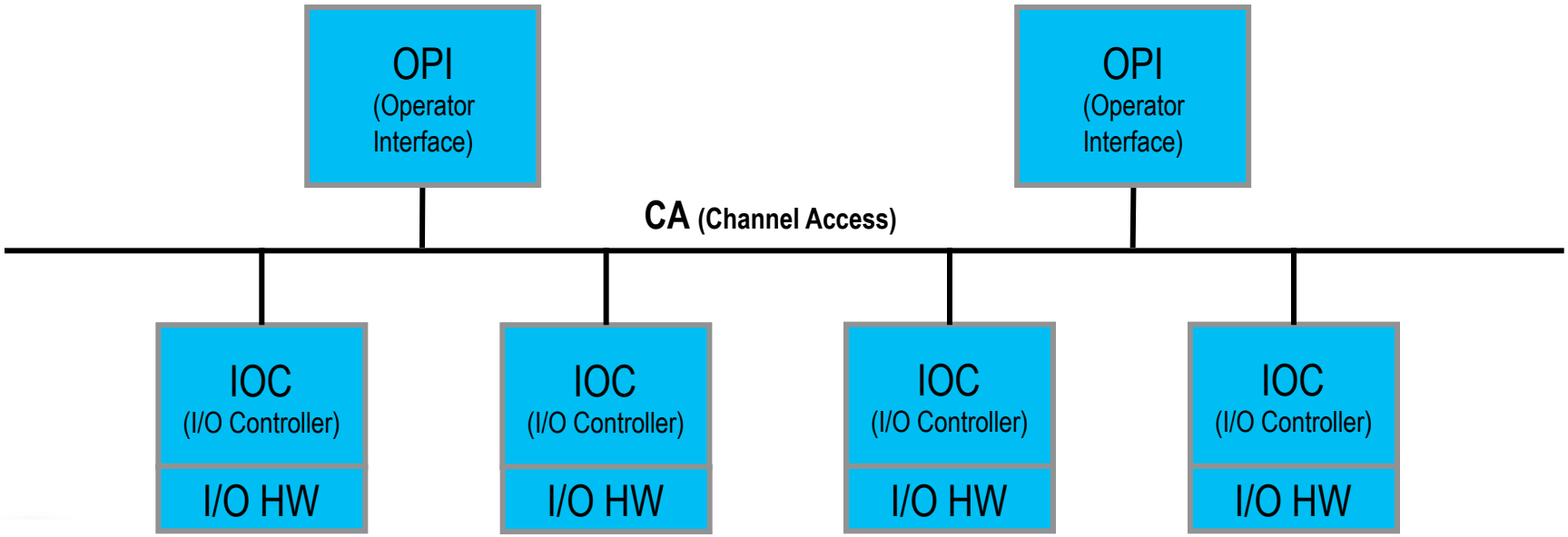
EPICS Overview



- Experimental Physics and Industrial Control System (EPICS)
- Used to develop and implement distributed control systems to operate large experiments
- SCADA architecture that uses client/server and publish/subscribe
- Input/Output Controller (IOC) collect experiment and control data
- Operator Interface (OPI) display data and control experiments
- Channel Access (CA) is a Ethernet protocol used to distribute the data
- Process Variables (PVs) are unique data items on CA protocol

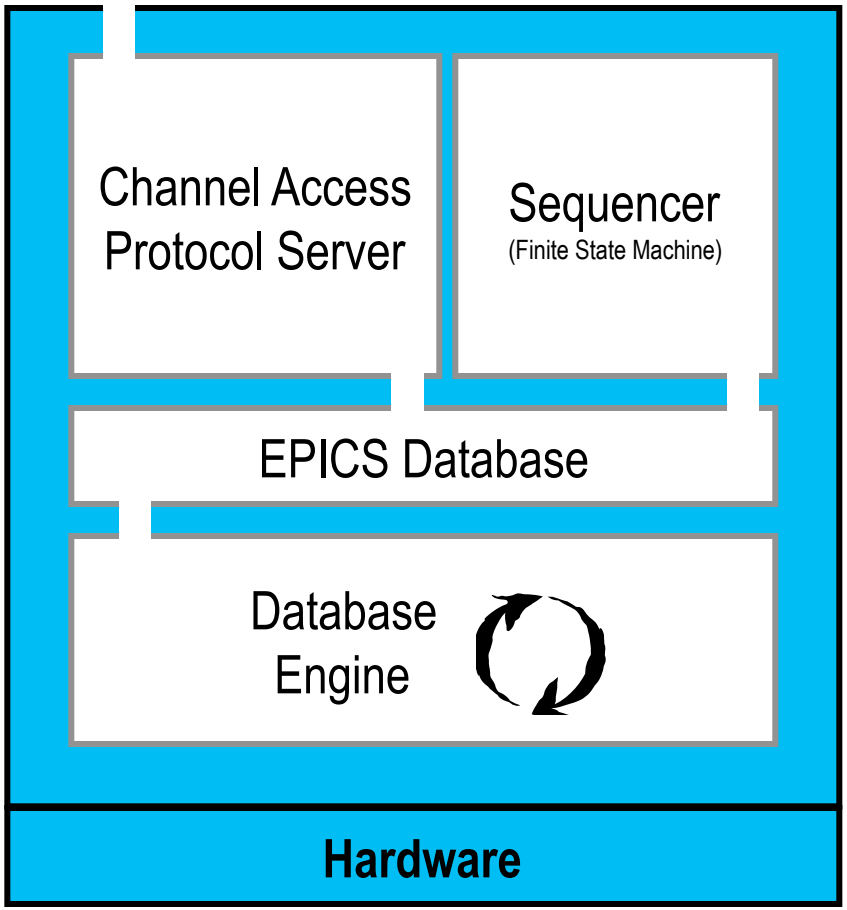
EPICS Software Architecture

- Distributed Clients (OPI – Operator Interface) and Servers (IOC – I/O Controllers)
- Network protocol: Channel Access (CA) with Process Variables (PVs)



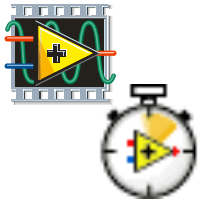
Analog I/O, Digital I/O, Motion Control, Image Acquisition, etc.

IOC Software Architecture



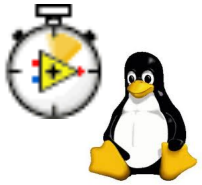
- Channel Access Protocol Server
  Publishes values from the database onto the network using Channel Access protocol
- Sequencer
  Controls timing for when to update values
- EPICS Database
  Contains the record definition and values
- Database Engine
  Writes I/O values to the database

Presentation Scope



LabVIEW	I/O Server	EPICS CA Client or Server

EPICS



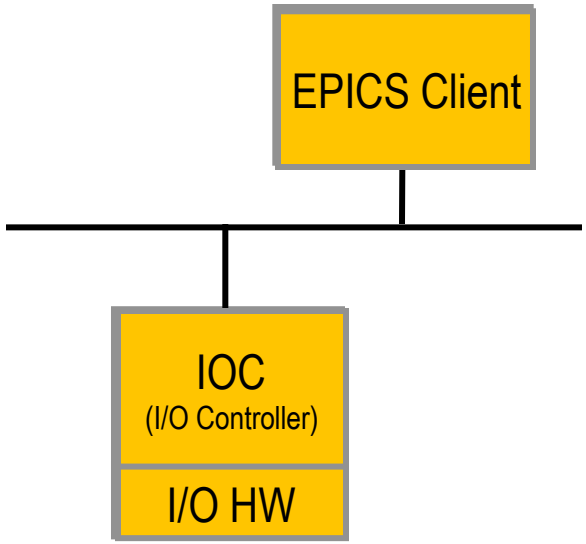
LabVIEW RT on cRIO	Shared Memory	EPICS IOC on VxWorks
LabVIEW RT on PXI	Hypervisor Shared Memory	EPICS IOC on Linux

EPICS

Linux on PXI	Device Support Linux Driver	EPICS IOC on Linux

EPICS

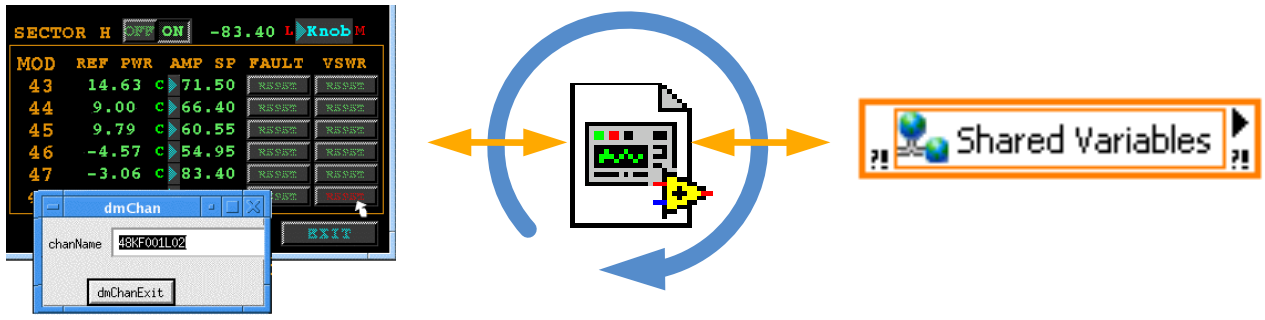
Why Integrate EPICS and LabVIEW ?



- LabVIEW as a Client
  - ✓ Presentation
  - ✓ Analysis
  - ✓ Control
- LabVIEW as a Server
  - ✓ Interface to hardware
  - ✓ Real-time control
  - ✓ Access to FPGA

Concept

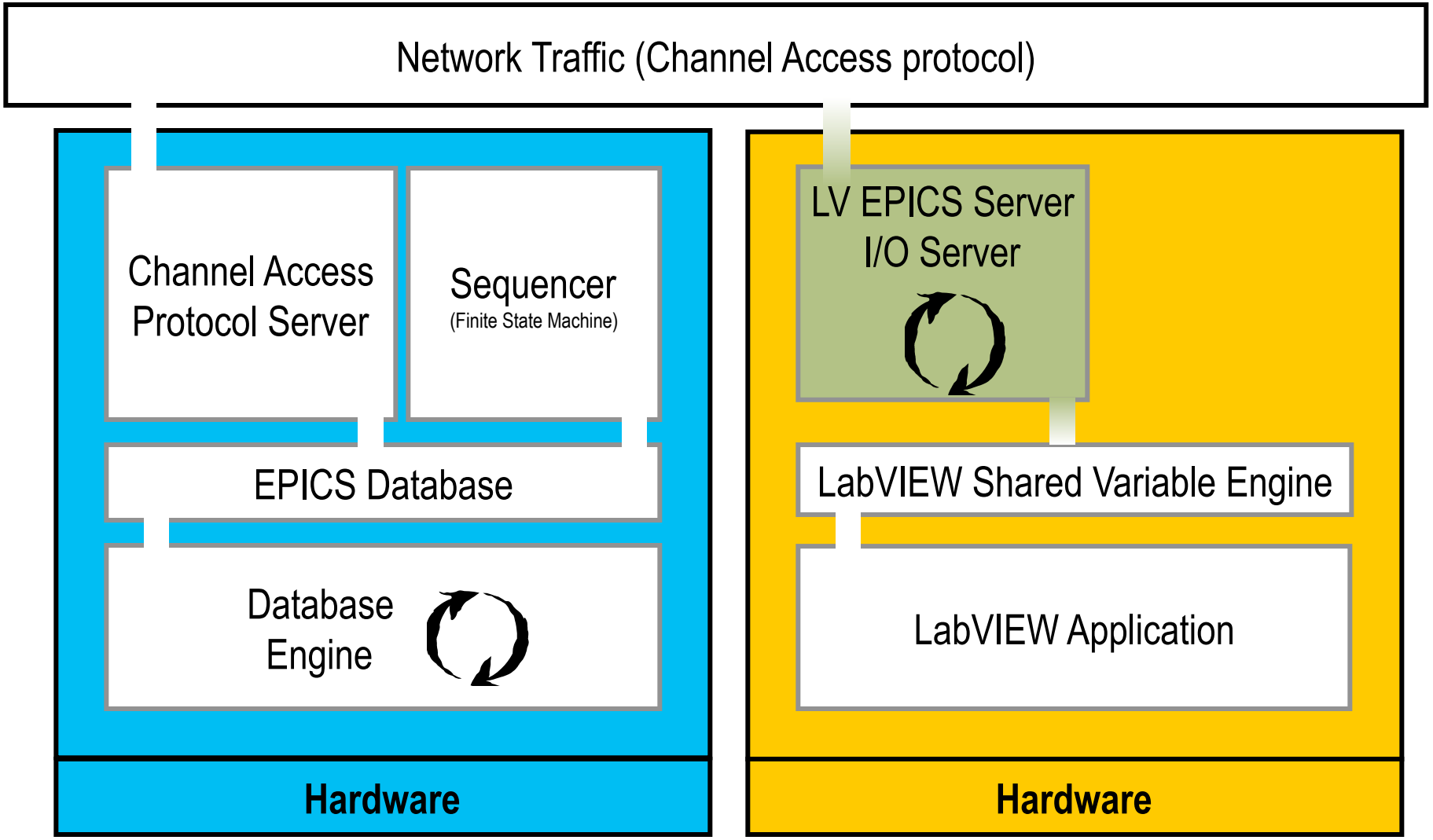
- The EPICS CA Server and EPICS CA Client are implemented as plug-ins to the I/O Server
- In both cases the interface in LabVIEW is implemented via the Shared Variable
- The interface to EPICS is via Channel Access



Process Variables (PVs) **I/O Server** **Shared Variables**

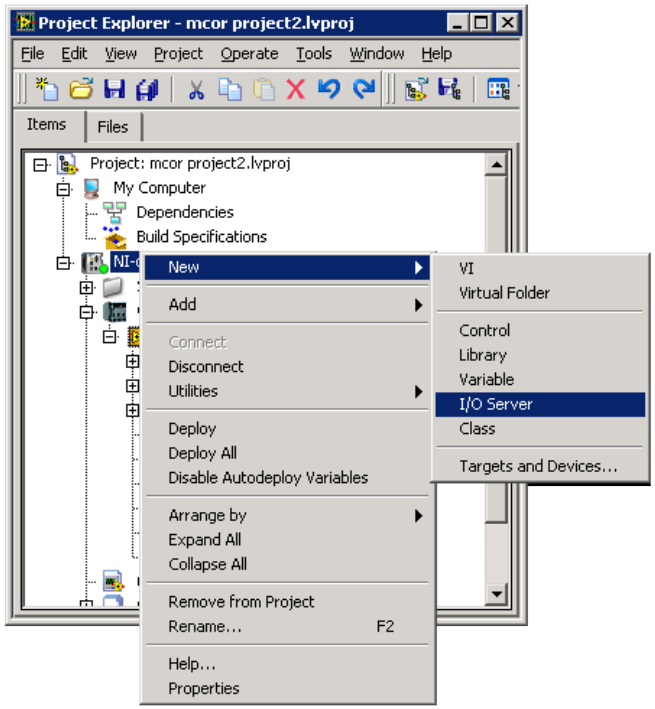
LabVIEW EPICS CA Server

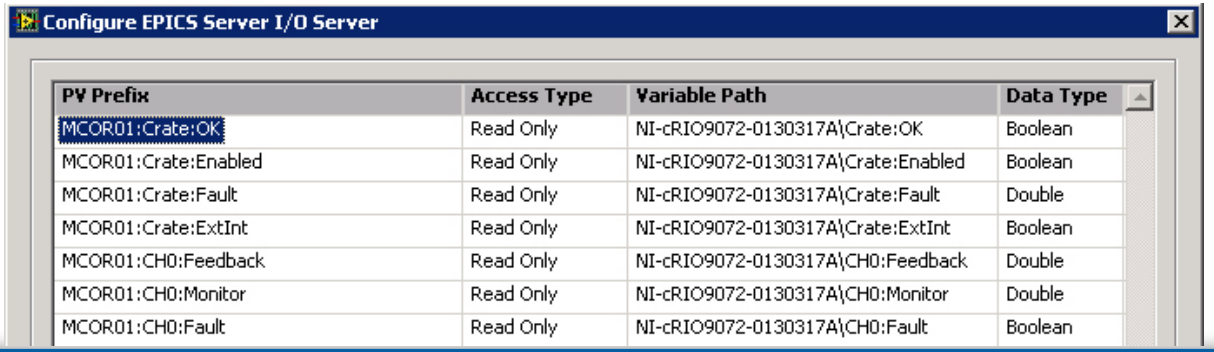
Architecture Comparison



LabVIEW EPICS CA Server

- Runs on LabVIEW for Windows and LabVIEW Real-Time
- Implemented as an I/O Server
- Interfaced via Shared Variable
- Provides Channel Access functionality only

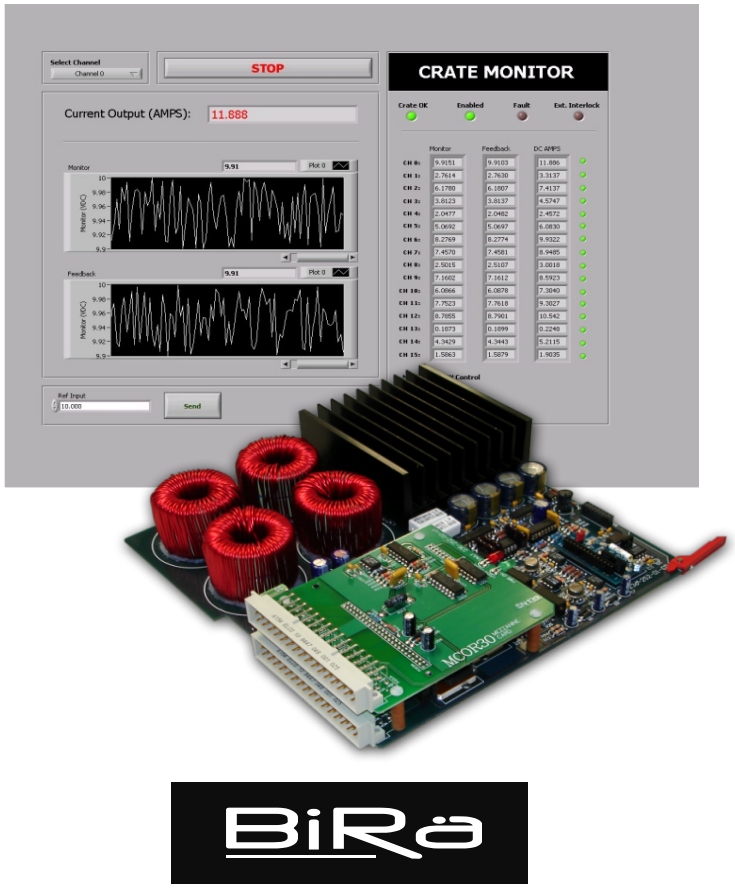




PV Prefix	Access Type	Variable Path	Data Type
MCOR01:Crate:OK	Read Only	NI-cRIO9072-0130317A\Crate:OK	Boolean
MCOR01:Crate:Enabled	Read Only	NI-cRIO9072-0130317A\Crate:Enabled	Boolean
MCOR01:Crate:Fault	Read Only	NI-cRIO9072-0130317A\Crate:Fault	Double
MCOR01:Crate:ExtInt	Read Only	NI-cRIO9072-0130317A\Crate:ExtInt	Boolean
MCOR01:CH0:Feedback	Read Only	NI-cRIO9072-0130317A\CH0:Feedback	Double
MCOR01:CH0:Monitor	Read Only	NI-cRIO9072-0130317A\CH0:Monitor	Double
MCOR01:CH0:Fault	Read Only	NI-cRIO9072-0130317A\CH0:Fault	Boolean

Example – BiRa's Power Supply

- 16 channels of high precision bipolar DC power
- Mainly used for corrector magnets in particle accelerators
- Running LabVIEW EPICS CA Server on an embedded real-time controller



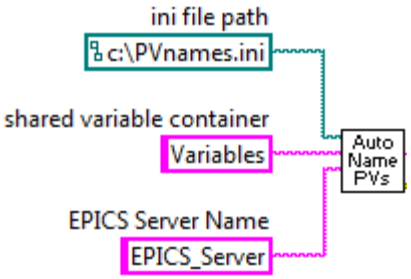
BiRa

Implementation – CA Server

- Start by creating the Shared Variables you want to publish on the EPICS network
- Then create an EPICS CA Server I/O Server
- Define the PV names and associate them with the Shared Variables
- Read or write to the Shared Variables in LabVIEW to access the associated PVs

Programmatic Access to CA Server

- Simple implementation is configuration-based
- New feature since LabVIEW 2010 allows to programmatically:
  - Create an EPICS CA Server
  - Create the Process Variables
  - Bind Process Variables to Shared Variables
- Benefits
  - Easily handles large number of PVs
  - Greatly facilitates deployment



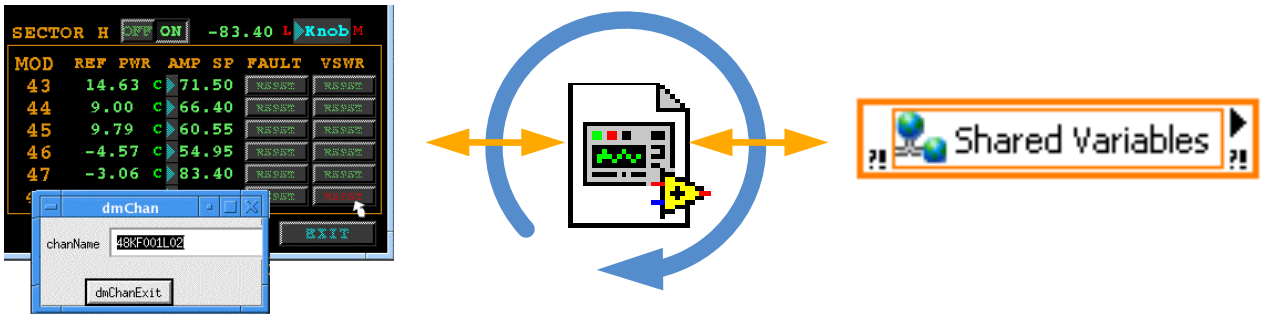
Possible Implementation

- Create PVs programmatically and use an ini file to modify the names of the PVs
- Each unique system can use the same application where the only difference is a text file

```
[Prefix]
Prefix=NI
[RenamePV]
SetPoint=setp
Temperature=temp
```

Concept

- The EPICS CA Server and EPICS CA Client are implemented as plug-ins to the I/O Server
- In both cases the interface in LabVIEW is implemented via the Shared Variable

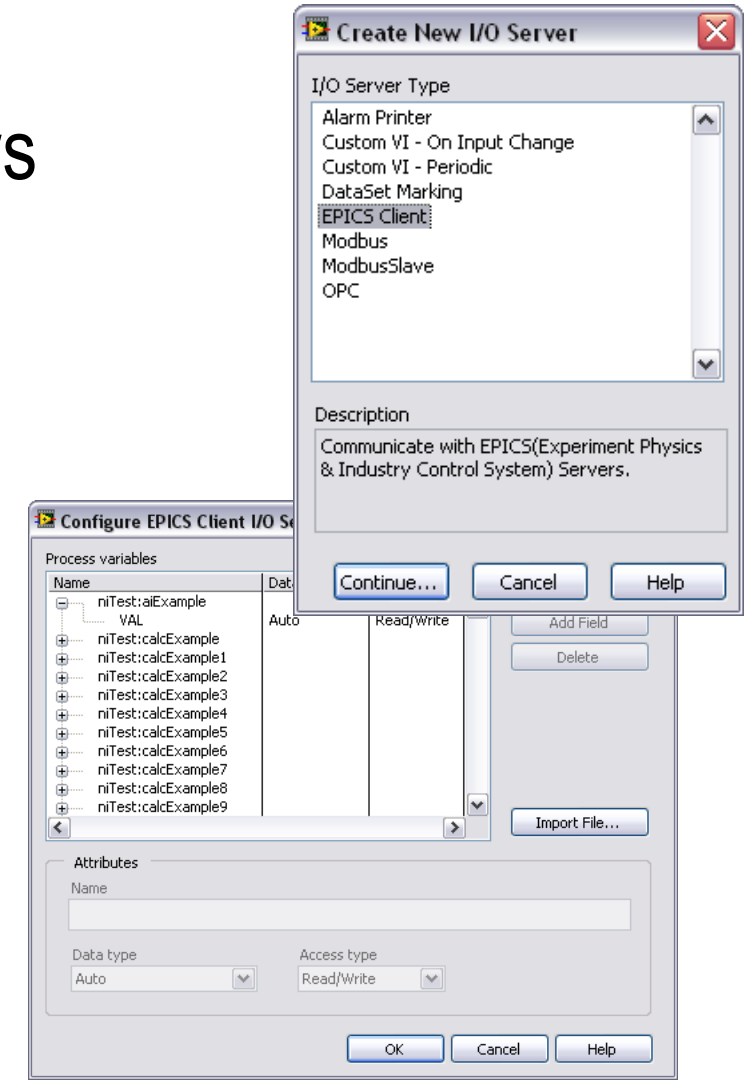


Process Variables (PVs) **I/O Server** **Shared Variables**

LabVIEW EPICS CA Client

LabVIEW EPICS CA Client

- Runs on LabVIEW for Windows and LabVIEW RT
- Implemented as an I/O Server
- Interfaced via Shared Variable



Implementation – CA Client

- Create an EPICS CA Client I/O Server
- Define the PVs you want to monitor, either manually or by importing a *.db* file
- Create the associated Shared Variables and bind them to each PV
- Read or write to the Shared Variables in LabVIEW to access the associated PVs

Programmatic Access to CA Client

- New Feature in LabVIEW 2011
- Allows user to programmatically
  - Create an EPICS CA Client
  - Connect to existing Process Variables
  - Bind Process Variables to Shared Variables
- Benefits
  - Easily handle large numbers of variables
  - Generic UI can dynamically connect to specific PVs

Available Fields

- EPICS CA Server
  - On Windows, when using alarming with LabVIEW DSC, the corresponding fields (**HIHI**, **HHSV**, **SEVR**, etc.) are supported
  - On RT targets, the EPICS CA Server only allows you to set the **VAL** field
- EPICS CA Client
  - Any field can be accessed, but a Shared Variable will have to be created per field

Distribution

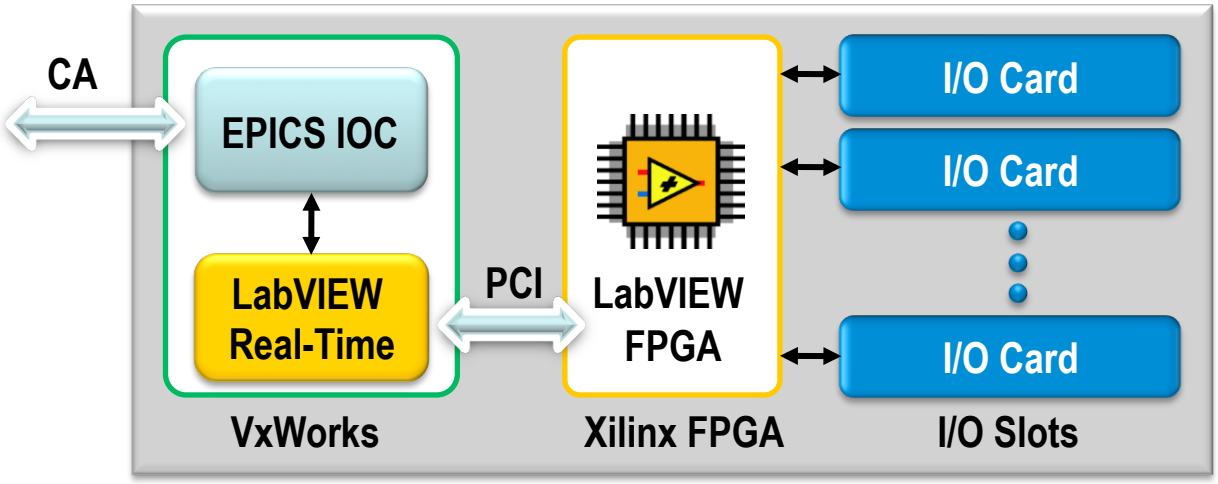
	Windows	Real-Time OS	Linux
CA Server	LabVIEW DSC 	LabVIEW RT 	X
CA Client	LabVIEW DSC  or Free download	LabVIEW RT 	X

- EPICS CA Server and CA Client available in LabVIEW 2011, via the DSC module for Windows and the RT module (PXI and cRIO)
- EPICS CA Client is also available as a free download
- No support for Linux

EPICS IOC on CompactRIO

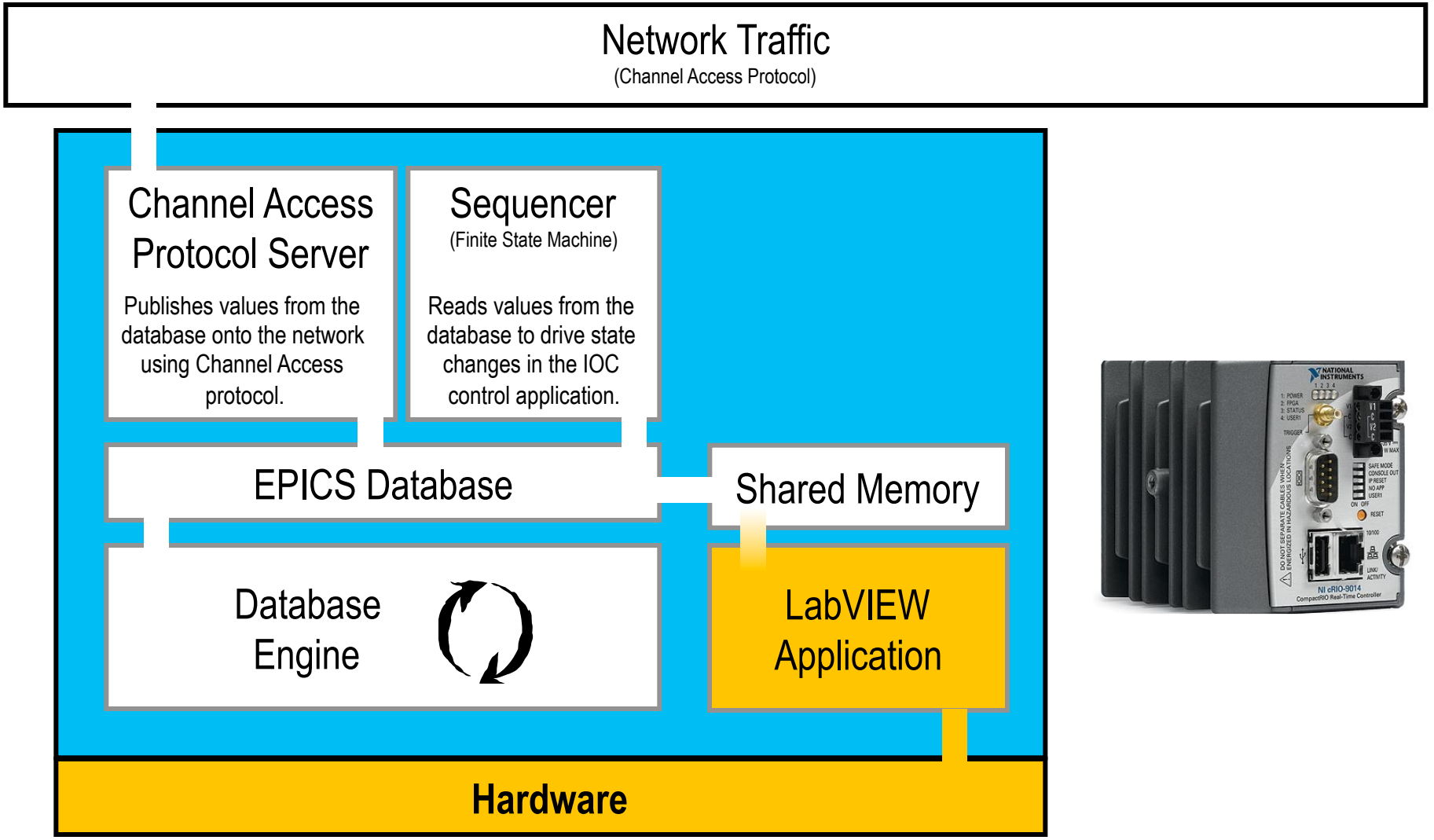
Embedding EPICS IOC on CompactRIO

- EPICS IOC and LabVIEW Real-Time running simultaneously
- Take advantage of FPGA platform with CompactRIO



CompactRIO Architecture

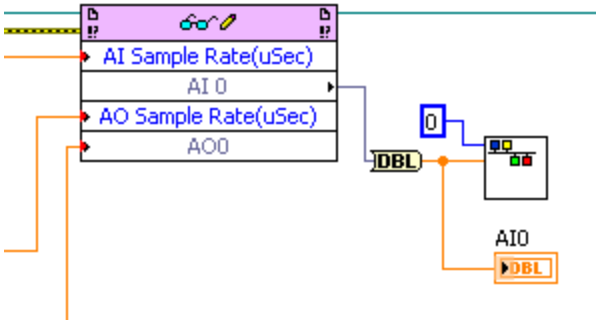
IOC Server on CompactRIO



IOC Server on CompactRIO

- CompactRIO controller runs VxWorks
- Implemented via shared memory
- Interface to hardware via LabVIEW RT and FPGA
- Can be used with Scan Mode
- Requires a custom VxWorks kernel (LV 2010 SP1 and earlier)
- Requires special .out files (LV 2011)
- Limited EPICS Device Support

```
# Read a double from shared memory
record(ai,"$(NAME):AI0") {
# field("Read a double from SM")
  field(DTYP,"SM Device")
  field(INP,"@0")
  field(SCAN,".1 second")
}
```



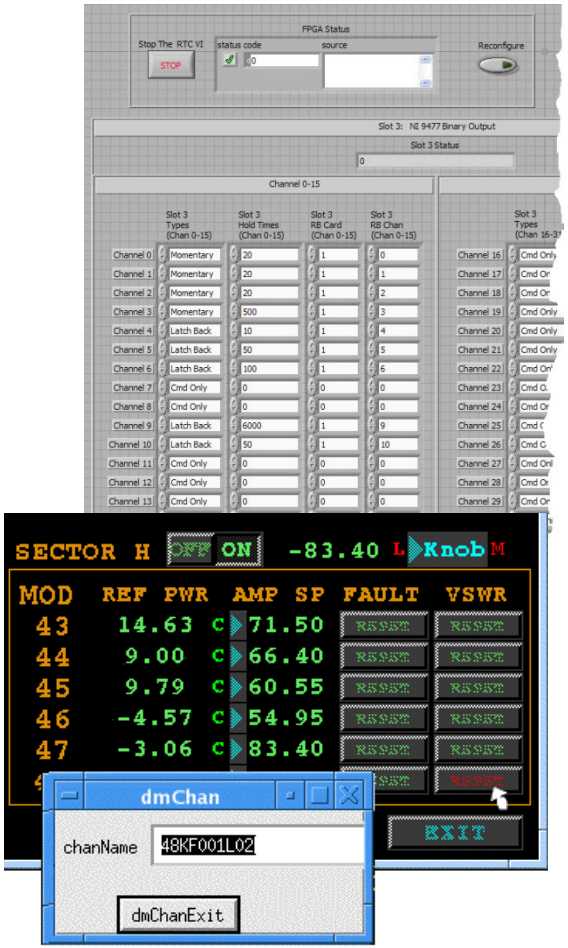
Shared Memory Benchmarks – LV 8.5 – 9014

- Arrays – Maximum transfer of 1 to 5 MB/s
- Integers – Maximum transfer of about 10 to 12 kB/s

Example – Los Alamos LANSCE

Los Alamos NATIONAL LABORATORY EST. 1943

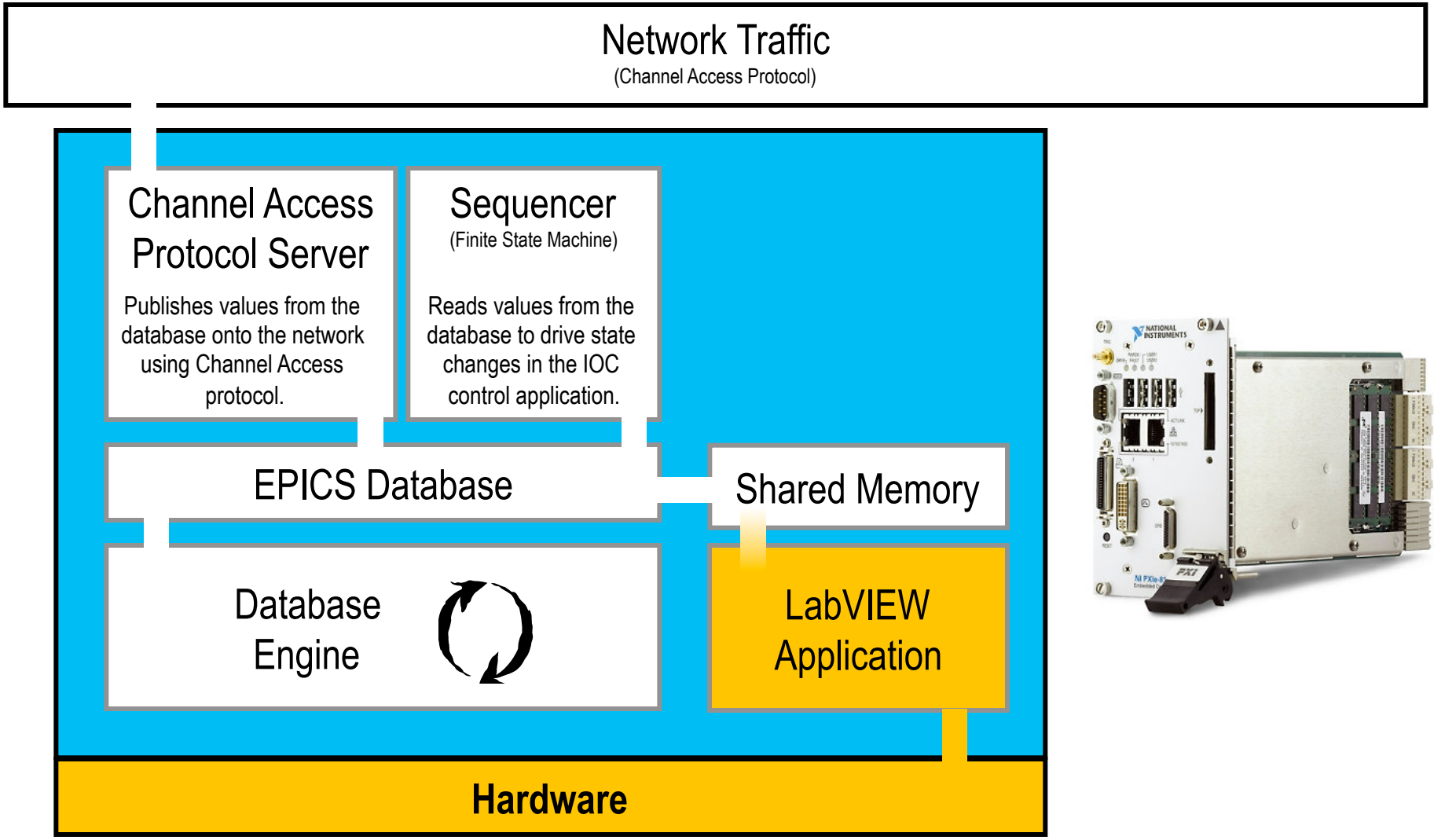
- Migration to a cRIO with embedded EPICS
  - 12 binary outputs
  - 36 binary inputs
  - 12 analog inputs
  - 5 stepper motor channels
- Full IOC functionality allows access to all record fields and EPICS utilities
- Maximum flexibility for partitioning the problem
  - LabVIEW for beam diagnostic
  - EPICS for industrial control



NATIONAL INSTRUMENTS

EPICS IOC on PXI

IOC Server on PXI



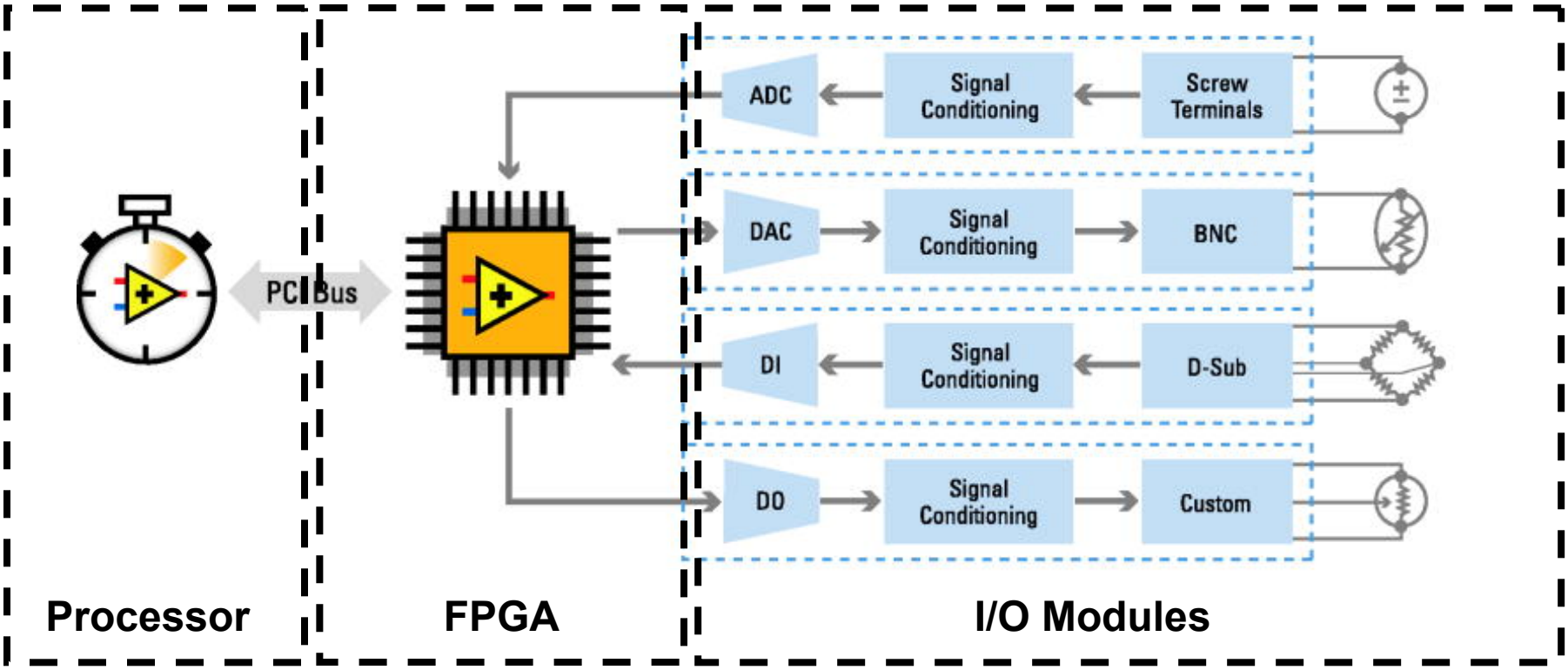
IOC Server on PXI – Shared Memory

- PXI controller runs Linux, hypervisor, and LV RT
- Implemented via hypervisor shared memory
- Interface to hardware via LabVIEW RT and FPGA (FlexRIO)
- EPICS Device Support needs to be developed by customer or integrator

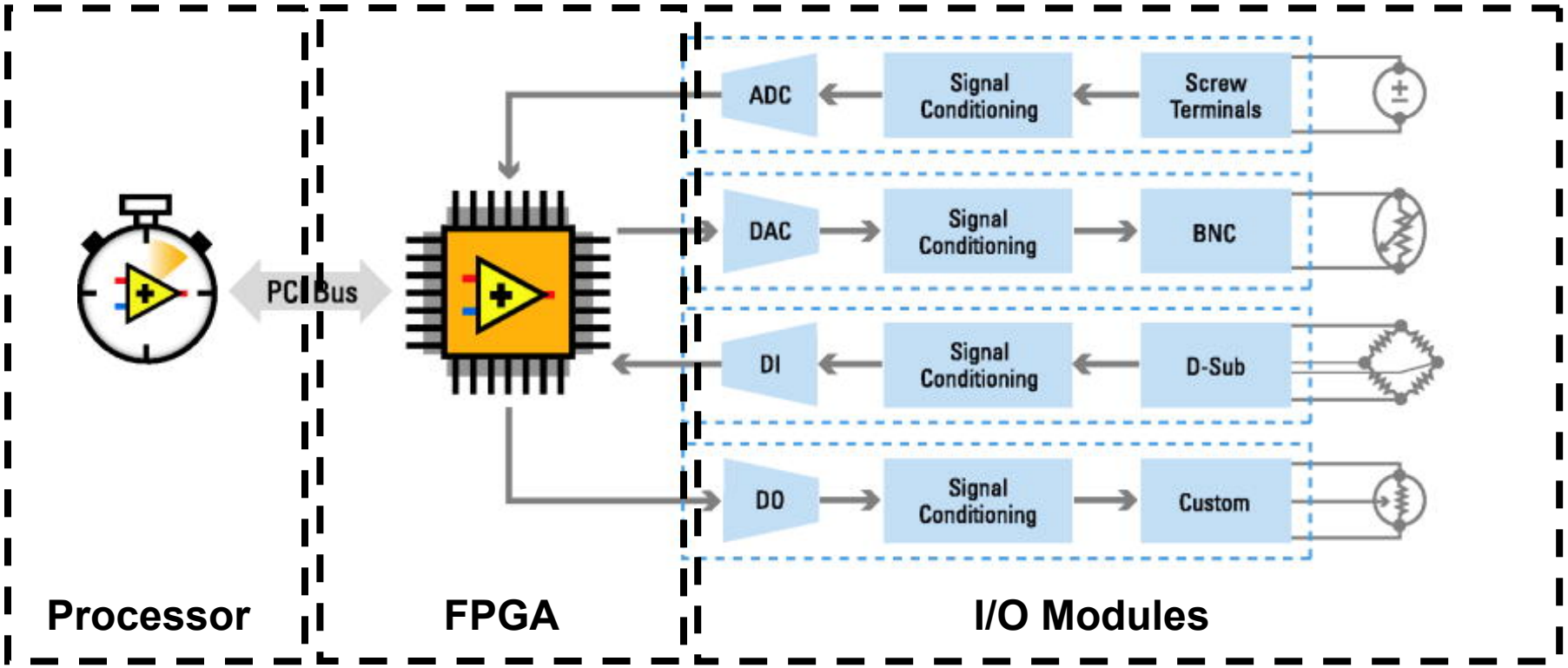
EPICS IOC on PXI (no LabVIEW)

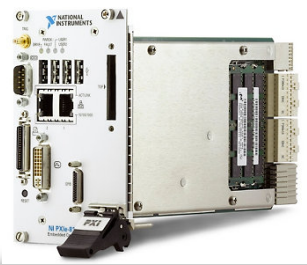
- Support driven by ITER requirements
  - EPICS and Linux (RHEL)
- EPICS device driver
  - Multifunction DAQ cards (M Series and X Series)
  - Timing and Synchronization (PXI-6682)
  - Ongoing efforts to support FPGA-based devices (R Series and FlexRIO) through device support and C API

Standard Embedded RIO Architecture

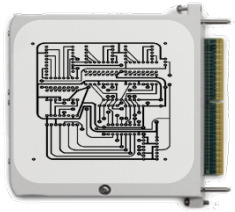


Standard Embedded RIO Architecture





NI



Custom

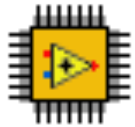
NATIONAL INSTRUMENTS

EPICS and FPGA-based Devices

- LabVIEW FPGA strategy is to empower domain experts to leverage FPGA technology
  - No VHDL / Digital Design training
- For VHDL experts
  - CLIP / IP Integration nodes to integrate IP (minimal LV FPGA required during development)
- EPICS device driver support implemented in C
- Generic FPGA interface, to be customized based on the application and FlexRIO adapter module

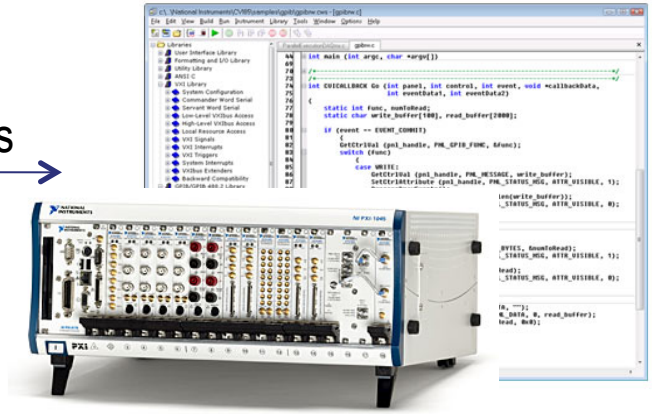
FPGA Interface C API and Linux

- Use RIO devices (e.g. R Series or FlexRIO) from C/C++ applications running on Linux
- Generate C header file from Windows development machine, then use C compiler of choice on Linux
- Development System Requirements for LabVIEW FPGA:
  - Windows XP (or later)
  - NI-RIO 3.5.0 (or later)
  - FPGA Interface C API 1.2 (or later)
  - LabVIEW FPGA 2009 (or later)
- Deployment System Requirements:
  - 32-bit Red Hat Enterprise Linux 5.x or 32-bit Scientific Linux 5.x
  - NI-RIO 3.5.0 for Linux (or later)



FPGA Interface C API on Linux

Generated
Header and
Source Files

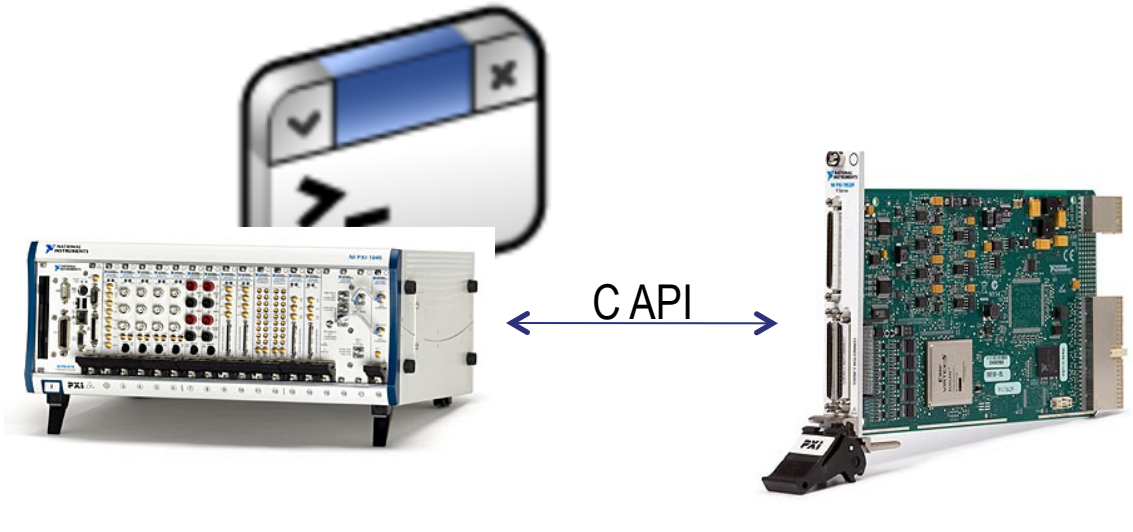


LabVIEW FPGA Development
(Windows)

Linux Development

1. Develop LabVIEW FPGA VI, compile bitfile, and generate C API.
2. Develop and build C application with generated C API.

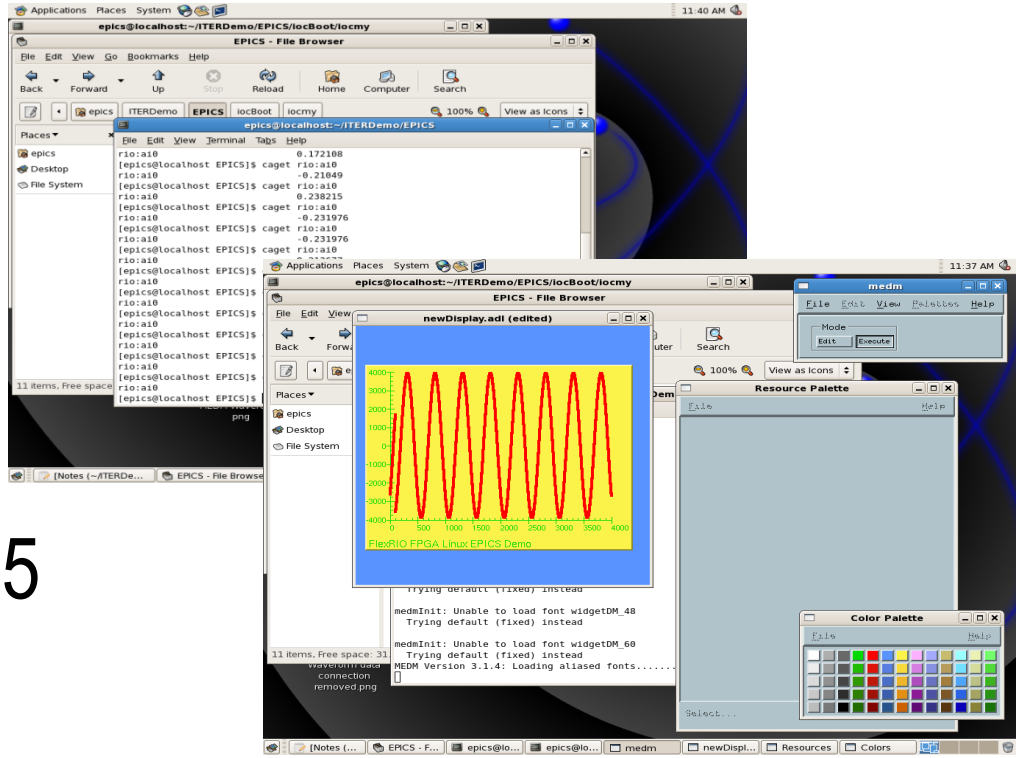
FPGA Interface C API on Linux



1. Develop LabVIEW FPGA VI, compile bitfile, and generate C API.
2. Develop and build C application with generated C API.
3. Deploy built application and bitfile to Linux target and run.

Current Support

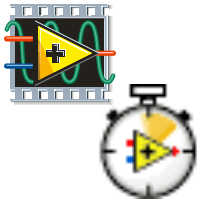
- EPICS Record types supported
  - Binary in
  - Binary out
  - Analog in
  - Analog out
  - Waveforms
- Linux 32bit RHEL 5.5
- EPICS 3.14.11



NATIONAL INSTRUMENTS

Options for EPICS and NI Hardware

- 1



LabVIEW	I/O Server	EPICS CA Client or Server

EPICS

- 2a

LabVIEW RT on cRIO	Shared Memory	EPICS IOC on VxWorks

EPICS

- 2b

LabVIEW RT on PXI	Hypervisor Shared Memory	EPICS IOC on Linux

- 3

PXI (No LabVIEW)	Linux Driver Device Support	EPICS IOC on Linux

EPICS