Brief Overview of EPICS.

Pierrick Hanlet
18 February 2020
Outline

I  Introduction
II Simple View of EPICS
III Standard Deployment of EPICS
IV EPICS Features and Tools
V  Present Work
VI Merging ACNET/EPICS
VII Conclusions
Outline

I  Introduction
II  Simple View of EPICS
III Standard Deployment of EPICS
IV  EPICS Features and Tools
V   Present Work
VI  Merging ACNET/EPICS
VII Conclusions
Introduction

• Some refer to EPICS as a “toolkit”
  − Toolkit (to me) implies user interfaces and applications only
  − I prefer the word “framework”

• EPICS is:
  − a framework from which to build a control system
  − open source software
  − an international collaboration
  − an end-to-end set of tools and applications

• EPICS history – *see Andrew Johnson's talk following this*
Introduction

- Fermilab control system, based on acnet protocol, is driven by the client – see Beau’s talk Thursday

- EPICS has a different paradigm than our control system in that the front end can be built to exploit all of the functionality of a field device, e.g. in the attributes:
  - Alarms: severity and limits
  - Calibration
  - Some aspects of archiving
  - Scanning
  - Sequencing
  - Timestamp
  - Control limits
  - Operating limits
Introduction

- There is little incentive for new employees to learn ACNET if they don’t imagine staying at Fermilab; however, those who learn EPICS can bring this skill set elsewhere. Hence, the long term maintainability of ACNET is in question – and we’re not getting younger!
- Most use of controls is from the client side, namely with “ACNET” consoles and applications.
- The use of EPICS within AD controls does not necessitate the abandoning of familiar applications.
- We have developed front ends to talk to the DPM via DRF3
  - well underway – see talk on Thursday
- Thus can connect FE information to familiar ACNET applications
- EPICS GUIs would also be developed with the front ends so that users can use either/both ACNET/EPICS GUIs
Outline

I  Introduction

II  Simple View of EPICS

III  Standard Deployment of EPICS

IV  EPICS Features and Tools

V  Present Work

VI  Merging ACNET/EPICS

VII  Conclusions
A Simple View of EPICS

**Experimental Physics & Industrial Control Systems**

- HW+Drivers connect to LAN via IOCs (Input/Output Controllers)
- IOC is controller (VME, PC, ...) and its software
- IOCs create PVs (process variables) to represent HW parameters
- PVs further described with native fields
- PVs available on LAN to other IOCs or clients
- Client Side are different GUIs, alarms, archives, displays, ...

---

![Diagram showing the relationship between HW, PVs, and various software components]

- User Access: CA Ctrl, CA Mon, CA ALH, CA Arch
- Client Side: PVs
- EPICS IOCs: IOC (VME), IOC (PC)
- LAN: PVs
- Drivers: Hardware Devices: Cryo, pump, chiller, ...
- Magnet PSU, I, V, ramp, ...
- HV PSU, I, V, ramp, ...

---

19/02/20  Pierrick M. Hanlet  8 of 38
A Simple View of EPICS

• EPICS “base”: the main core of EPICS, comprising the build system and tools, common and OS-specific interface libraries, Channel Access and PV Access client and server libraries, static and run-time database access routines, the database processing code, and standard record, device and driver support.

• EPICS versions:
  − 3.x.y channel access, most commonly used
  − 4.x.y PVaccess, newer and not commonly used
  − 7.x.y = v3 + v4 combines the two

• For modernization and PIP-II, writing EPICS-based front ends means that we can start now with the software because it will be transferable to any new platform that we will choose and use in the future – only requires rebuilding for new architecture
A Simple View of EPICS

• Features
  – Interfaces: serial, gpib, ethernet, modbus, snmp, ...
  – Platforms: linux, windows, mac, VME, android, raspberry pi, ...
  – International support: experts around the world – rarely waited >1hr for help
  – Many, many device drivers already written

• EPICS version 7: – see Andrew Johnson’s talk Thursday
  – combination of v3 (channel access) and v4 (PV access)
  – can use v3 IOCs and it still works
  – base has ~175k lines of code from 57 contributors
  – Code is mostly C (~60%) and C++ (~25%)
  – v7 has structured data – added from Channel Access
  – PVaccess allows for high bandwidth data (not just slow controls)
  – already stable but remaining bugs cleared and should be as robust as v3-14-xx by PIP-II era
Many “record” types: e.g.
- data types (binary, float, multi-bit, waveform, structures)
- calculations
- functional (heartbeat, computer status, busy, ...)
- sequences
- aSub: C/C++ function inputs PVs, processes, and outputs PVs

Record types have native “fields” to further describe record: e.g.
- alarm
- units
- operating limits
- scanning rates: time based, I/O interrupt, passive
- fanouts and forward links
- description
- dozens of fields; these are defined in xxx.dbd files

Records are used to create PVs
PVs are used to create IOCs
IOCs then can completely describe a hardware device
# A Simple View of EPICS

## Fields common to all record types:

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCAN</td>
<td>GBLCHOICE</td>
</tr>
<tr>
<td>PINI</td>
<td>GBLCHOICE</td>
</tr>
<tr>
<td>PHAS</td>
<td>SHORT</td>
</tr>
<tr>
<td>EVNT</td>
<td>SHORT</td>
</tr>
<tr>
<td>PRIO</td>
<td>GBLCHOICE</td>
</tr>
<tr>
<td>DISV</td>
<td>SHORT</td>
</tr>
<tr>
<td>DISA</td>
<td>SHORT</td>
</tr>
<tr>
<td>SDIS</td>
<td>INLINK</td>
</tr>
<tr>
<td>PROC</td>
<td>UCHAR</td>
</tr>
<tr>
<td>DISS</td>
<td>GBLCHOICE</td>
</tr>
<tr>
<td>LSET</td>
<td>SHORT</td>
</tr>
<tr>
<td>LCNT</td>
<td>UCHAR</td>
</tr>
<tr>
<td>PACT</td>
<td>UCHAR</td>
</tr>
<tr>
<td>FLNK</td>
<td>FWDLINK</td>
</tr>
</tbody>
</table>

## Fields common to most record types:

<table>
<thead>
<tr>
<th>Name</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>INP</td>
<td>Input Link</td>
</tr>
<tr>
<td>DTYP</td>
<td>Device Type</td>
</tr>
<tr>
<td>RVAL</td>
<td>Raw Value</td>
</tr>
<tr>
<td>VAL</td>
<td>Value</td>
</tr>
<tr>
<td>SIMM</td>
<td>Simulation Mode</td>
</tr>
<tr>
<td>SIML</td>
<td>Simulation Mode Location</td>
</tr>
<tr>
<td>SVAL</td>
<td>Simulation Value</td>
</tr>
<tr>
<td>SIOL</td>
<td>Simulation Value Location</td>
</tr>
<tr>
<td>SIMS</td>
<td>Simulation Mode Alarm Severity</td>
</tr>
</tbody>
</table>

- For example an ai (analog input) record has ~50 fields which fall into:
  - scan parameters
  - read and convert parameters
  - operator display parameters
  - alarm parameters
  - monitor parameters
  - run-time parameters
A Simple View of EPICS

• A PV (Process Variable) is a representation of a hardware parameter
  - e.g. temperature, voltage, current, flow, ...
• Fields (attributes) further define a PV
  - dozens of possible fields
• Examples:

```
record(ai, 'MyPVName') {
    field(DESC, 'This is my favorite PV')
    field(SCAN, '.1 second')
    field(EGU, 'l/s')
    field(HHSV, 'Major')
    field(HSV, 'Minor')
    field(HIHI, '7.2')
    field(HIGH, '5.6')
    field(FLNK, 'MyNextFavPV.PROC')
}

record(calc, "MICE-BL-ISIS-01:DBMLS") {
    field(DESC,"ISIS Beam Loss difference")
    field(INPA,"MICE-BL-ISIS-01:IBMLS")
    field(INPB,"MICE-BL-ISIS-01:ABMLS")
    field(CALC, "A-B")
    field(HOPR, "3.0")
    field(LOPR, "-3.0")
    field(SCAN, "Passive")
    field(FLNK,"MICE-BL-ISIS-02:ABMLS.PROC")
}
```
A Simple View of EPICS

- Support modules are analogous to modules one might add to a kernel; i.e. they aren’t required for a stable OS, but are required for a server’s specific function.
- Many support modules for HW already written
- Some support modules are effectively drivers for specific hardware
  - user written
  - found on EPICS website
  - ask EPICS user community
- One of the larger packages which contains many support modules is synApps
A Simple View of EPICS

- Airedale
- Bit8
- CAEN
- FINS
- FINStcp
- FINSudp
- GeaDenco
- HVSy127
- HVSy4527
- HVSy527
- Hy8401ip
- Hy8402ip
- Hy8515
- LeyboldTD20
- LuminOx
- NIKHEF_hall_probe
- OmronE5AC
- OxiInstCryojet-2-18-3
- Riello
- SecurityProbe
- StreamDevice-2-7
- StreamDevice-2-8
- TCW181B
- Tracker
- WTI
- wienerPL508
- adam4017
- alive-1-0-1
- areaDetector-R3-4
- asyn4-18
- asyn4-19
- asyn4-34
- asyn4-35
- autosave-5-10
- busy-1-4
- busy-1-6
- calLab_1320
- calc-2-8
- calc-R3-5
- calc-R3-7-1
- camac-2-7
- cdb
- cdb-python
- cmslon
- commitrevision.bash
- cp2800
- devlocStats-3-1-9
- gpib
- iocStats-3-1-15
- ip-2-10
- ip-2-20
- ipac-2.11
- ipac-2.15
- itc503
- keithley2700
- leyboldC2
- leyboldTD20
- makeall.bash
- mks937
- modbusR2-11
- modbusR2-3
- modbusR2-4
- motor-6-11
- seq-2.0.12
- seq-2.1.11
- seq-2.2.4
- seq-2.2.5
- snmp
- sscan-2-11-2
- sscan-2-6-6
- std-2-8
- std-3-5
Outline

I Introduction
II Simple View of EPICS
III Standard Deployment of EPICS
IV EPICS Features and Tools
V Present Work
VI Merging ACNET/EPICS
VII Conclusions
Standard Deployment of EPICS

• An EPICS IOC is created from 3 layers:
  – base
  – support modules – built against base
  – code specific to the IOC and built from base and selected support modules

Presently, I have 3 versions of base: v3.14.12.8, v3.16, v7.0.3.1
• I also have several versions of Support modules
• I use soft links to point to the version that I want
  – e.g. ~/epicsDEV/base = ~/epicsPRO/base-7.0.3.1
  – this model may change

To set up our standard deployment:
https://cdcvs.fnal.gov/redmine/projects/ad-controls-dept/wiki/Building_an_EPICS_IOC_on_a_clx_Machine
Standard Deployment of EPICS

- Software directory structure:
  - Development
    /home/epics/epicsDEV/base (build tools and core)
    /Support (modules: asyn, seq, ipac...)
    /iocTops (IOCs – built from base+Support)
    /Extensions (user applications)
    /Config (configuration files)
    /Software (external libraries)

- Production
  /usr/local/epics/base
  /usr/local/epics/Support
  /usr/local/epics/Config
  /home/epics/epicsPRO/iocTops
Standard Deployment of EPICS

My directory structure:
/home/epics/epicsDEV/base (build tools and core code)
  /Support (modules: asyn, seq, ipac...)
  /iocTops (IOCs – built from base+Support)
  /Extensions (user applications)
  /Config (configuration files)
  /Software (external libraries)

/home/epics/epicsPRO

/home/epics/epicsDEV/Extensions
  /Config
  /iocTops
  /Support
Standard Deployment of EPICS
Outline

I  Introduction
II Simple View of EPICS
III Standard Deployment of EPICS
IV EPICS Features and Tools
V  Present Work
VI Merging ACNET/EPICS
VII Conclusions
EPICS Features and Tools

EPICS Base

- Robust and well supported
- Build tools – nested Makefiles
- Configuration usually only requires modification to 1 (or 2) files
- Supported platforms (VME, linux, MAC, windows, R-pi, android, …)
- I/O and interrupts under the hood (only need configuration)

Graphical User Interfaces

- Historically, I’ve used edm for GUIs
- Simple to learn
- ascii configuration files
- But, it’s not sexy
- Avoided Control System Studio (CSS) because of eclipse, now has new java version called Phoebus (work in progress)
- Browser based, very configurable, on the fly GUIs
Examples
Examples

Synoptic Displays

- MICE Run Control
- Detector Summary
- CDB Read Status
- Open HV Controls
- High Voltage Display
- CKOV-A and CKOV-B Controls
Examples

Synoptic Displays
Examples

PRY Movement

Magnetic Fields

PRY Movement Summary

MICE B-Field Monitoring

SSU
- Upstream 30: Conversion OKAY 1
- Middle 30: Conversion OKAY 3
- Middle 270: Conversion OKAY 3

SSD
- Downstream 30: Conversion OKAY 3
- Middle 30: Conversion OKAY 3
- Upstream 150: Conversion OKAY 1
- Middle 270: Conversion OKAY 3

External
- South Mezz
- Corner South Mezz
- Under Cooling Channel
- North Outside PRY

Positive
- Solenoid
- Positive
- Positive

Exit
Examples
EPICS Features and Tools

- **Archiver**
  - Historically, I used Channel Archiver, but little support for it now
  - Testing Archive Appliance – SLAC
  - Millions of PVs
  - Clusterable
  - Staged storage

- **StripTool**

- **Probe**
EPICS Features and Tools

• Alarms
  - Limits are HIHI, HIGH, LOW, LOLO
  - Severity is MAJOR, MINOR, NO_ALARM
  - Assuming that hardware is fail-safe, I set HIGH & LOW limits to alarm when tolerance affects desired result, and HIHI & LOLO limits are set to alarm prior to a hardware fail-safe event
  - ForcePVs allow dynamic enabling/disabling of PVs
  - Using the State Machines, limits are set dynamically with state transitions
EPICS Features and Tools

• Gateway
  - Makes copies of PVs on different subnets to ensure control access to PVs
  - Filters: by IP address, by user, by PV
  - Filter none, read, or read/write
  - Filter by calculation
  - Ex. at Diamond: strict limited access to controls network, another gateway to allow operator access, another for general users

• Security
  - IOC access
  - Security levels can be changed dynamically
  - Filters: by IP address, by user, by PV
EPICS Features and Tools

• There are many newer features and tools than I have not shown here. We are actively evaluating many of them. Examples are:
  - areaDetector (n-dimensional data) – see Mark River’s talk Thursday
  - Phoebus – (purely java based CSS without eclipse) integrated tools on a browser like platform
Outline

I  Introduction
II  Simple View of EPICS
III  Standard Deployment of EPICS
IV  EPICS Features and Tools
V  Present Work
VI  Merging ACNET/EPICS
VII  Conclusions
**Present Work**

- All development work on VCLX4
- Rebuilt my code to run with EPICS-7
- Learn motor and areaDetector modules
- Five EPICS IOCs – front ends
  - Environment IOC (resurrected and modified for Fermilab)
  - Tomco 7kW RF Amplifier (2 days to write)
  - TSV-3000 RF Amplifier
  - Converting a VME 162 to a linux based 7805 and building a motor controller
  - areaDetector – cameras
  - SoC
- Archiver Appliance – modern archiver (works)
- Phoebus evaluation

Examples: archiver (3), striptool, edm/phoebus GUIs
Outline

I  Introduction
II  Simple View of EPICS
III Standard Deployment of EPICS
IV  EPICS Features and Tools
V   Present Work
VI  Merging ACNET/EPICS
VII Conclusions
Merging EPICS with ACNET

• So, should we completely switch to EPICS?
  − ACNET is a mature system
  − Change is too dramatic for those embedded in ACNET

• However, most use of controls is client-side
• Keep ACNET consoles and applications
• Feed EPICS IOCs to ACNET
  − Doing this with DRF3 and the DPMs
  − Operators still see ACNET consoles
  − Can also have EPICS GUIs (edm and/or Phoebus)

• New front ends can come online as EPICS IOCs
  − When hardware is upgraded to linux-based system, IOC will only need to be recompiled for new platform, but not re-written
Merging EPICS with ACNET

The tough part is the mapping: – see talk on Thursday

**IOC**
- PV
  - field
  - field
  - field
- PV
  - field
  - field
  - field

**ACNET**
- Device
  - read
  - set
  - control
  - Device
    - read
    - set
    - digital alarms
  - Device
    - read
    - set

DPM
DRF3
Outline

I Introduction
II Simple View of EPICS
III Standard Deployment of EPICS
IV EPICS Features and Tools
V Present Work
VI Merging ACNET/EPICS
VII Conclusions
Conclusions

- Many thanks to Denise for helping to set up new tools
- Need to design configuration: servers, archiver, IOC nodes
- Several tests in progress
  - Historically, I’ve used edm for GUIs, now learning CSS/Phoebus
  - Linux based VME controller and motor driver
  - areaDetector for cameras
  - Archive Appliance
  - EPICS-7 development
  - Template for SoC EPICS deployment
- EPICS IOCs can “easily” be used in conjunction with ACNET applications – well underway
- Standard EPICS setup being deployed
State Machines – Problem

- Different sub-systems have different needs
- Each sub-system has $10^1 - 10^3$ PVs
- Many PVs have up to 4 alarm limits
- Each PV has different archiving needs
- For different operational states:
  - the PVs of interest change
  - the alarm limits change
  - the archiving needs change
  - the list of critical PVs change

Too much room for human error!

e.g. Powering a superconducting magnet
State Machines – Problem, no Problem

EPICS state notation language employed:

- define equipment operational states
- for each state:
  - define transitions out of state
  - set alarm limits
  - set archiving features
  - define critical variables
- check for software interlocks; e.g. quench
- check for errors
- check for transition

All parameters come from configuration database (CDB) – ensures correct settings
Subsystem Owners must enumerate the states and provide:

1) Description of state
2) Transition into state
3) PVs of interest
4) Alarm limits for PVs
5) Archiving features for PVs
6) AutoSMS (auto dialer) flag
7) Hardware interlocks
8) Software “interlocks” (enables)

- Required for each state
- Loaded into the CDB
State Machine Algorithm

For each subsystem & state, the algorithm:

- Transitions:
  - manual
  - automatic
State Machine: SS Example

Spectrometer Solenoid Magnets:
1) Offline
2) Pumping: establish insulating vacuum
3) Pumped: insulating vacuum established
4) PreCooling: $N_2$ pre-cooling (down to T~100K)
5) Cooling: cryo-coolers lower shield/cold mass T
6) LHeFilling: add liquid He
7) Cold: cold and stable
8) Ramping: applying current
9) Powered: stable operation
10) Quenched: quench detected
11) Error: error requires operator intervention
12) Testing: interlocks disabled for manual testing
13) Warming
State Machine: Target Example

Target Example:
1) Offline
2) Parked_Powered
3) Raised_Holding
4) Raised_Actuating
5) Moving_Holding
6) Lowered_Holding
7) Lowered_Actuating
8) Error
9) Unknown

ISIS
MICE
target