The ESS EPICS Environment e3
Its build and deployment

Timo Korhonen
Jeong Han Lee

Integrated Control System Division
ESS, Sweden

https://www.europeanspallationsource.se
June 6, 2019
Why Environment is Needed

**Complexity**

- Each Site or Person follows various ways to develop, maintain, and configure modules and applications.
- Each Site uses a different HW and SW architecture
- Site-wide subsystems to be monitored by EPICS IOCs have their own requirements

**Consistency**

- Consistency for users and developers and even more for ESS Facility is the key to develop, operate, and maintain the control system from its initial conception through its retirement within the ESS’s life cycle.
- ESS (or each site) needs its own Environment. (For FRIB, NSLS-II, Debian Packaging System / For ITER, CODAC / PSI, the original version of e3 / ...)
- e3 was designed to achieve it.
Why e3 is needed: Global Prospective

- Quality management of IOCs
  - EPICS full freedom: good for small groups
  - e3 limited freedom: good for ICS who has to provide a consistent environment to any stakeholders in ICS, Accelerator, Target, and Neutron Science

- Common source code management problems:
  - varying quality of modules (open source): code, documentation, & styles
  - version changes of base, modules, etc.
  - customized patch files, while keeping in sync with the EPICS community
  - platform variability
  - inconsistent version management overall in EPICS community

- Have to consider different EPICS users over ESS life time
  - advanced users: can manage their own IOC details
  - device integration focused (time limited) users: want to avoid low level development, compiling code etc.
  - less experienced users: benefit from pre-selection and prepared modules
  - core development users
Why e3 is needed: Local Prospective

**Users**

- avoid re-building IOCs from scratch
- do not care about internal dependency among EPICS base, and modules
- focus more on the IOC functionality and post-processing of signals for each sub-system
- focus more on the user specific functionality (post-process, data analysis, user interface, and so on)
- transfer some IOC development effort to a team of e3 Architects (currently, only one)
- use the ESS specific version rules consistently on EPICS base, and modules independent upon external sources
- avoid incompatible version combinations
- have the future migration process over EPICS base versions is less likely to cause problems
**Desideratum for e3**

**Source Code Changes**
- site, community, both patch files
- a time interval for possible merging activities
- rejection from community sources
- files, clone, fork, and branch ...

**Release Version Numbers**
- handle various version numbers (e.g., R1-0, v1.1, s7plc_1_4_0, no version)
- define ESS version for all of them (1.0.0, test, ae5d083, han-4am)

**Disk and Network Resources**
- Separate src files from installed files in order to save disk and network resources (e3 source files ~ 2 GiB)
- Isolate system specific applications in different locations

**Users, Users, and Users**
- can run only working IOCs
- can integrate an existent EPICS module into e3
- can develop a module within e3
- can develop a non-existent EPICS module with EPICS, and integrate into e3
- can develop an application with the existent e3 modules
- can deploy any of them in an emergency situation (4am) with limited resources and environment

**Maintain, maintain, and maintain**
- clear structure to understand its dependencies
- easily retire unused base, modules within the ESS life cycle
- duplicate a specific version of the e3 production in any places
- add new base, new modules into the production and into a development
- easily distinguish between e3 at different time domains

**Increase Degree of Freedom for Users**
- Allow to have multiple e3 versions in a single host with the standard EPICS environment.
- Allow to install e3 in any other Linux flavors (CentOS, Debian, Ubuntu, Fedora, Raspbian, ...)
- Allow to setup the standard EPICS env with e3
- Allow to have the entire setup locally

**Quantized Integration / Deployment Continuously**
- Deploy only the quantized version of the combination of all components (base, require, all other modules), since EPICS base is the long-standing one.
- Save resources to resolve potential overheads and to focus IOC functionality.

**Mimicked EPICS Building System**
- Mimic the EPICS building system
  * Makfile, configuration, rules, and so on
- In the future, will design the ESS rules, and configuration for the Standard EPICS building system.
### EPICS IOC vs e3 IOC in the simplest case

<table>
<thead>
<tr>
<th>EPICS IOC</th>
<th>e3 IOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run <code>makeBaseApp</code></td>
<td>Add database and protocol files</td>
</tr>
<tr>
<td>Define Base, Modules in RELEASE</td>
<td></td>
</tr>
<tr>
<td>Add database and protocol files</td>
<td>Update Makefile</td>
</tr>
<tr>
<td>Update Makefile</td>
<td>Build</td>
</tr>
<tr>
<td>Build</td>
<td>Edit <code>st.cmd</code></td>
</tr>
<tr>
<td>Edit <code>st.cmd</code></td>
<td>Write <code>st.cmd</code></td>
</tr>
<tr>
<td>Run</td>
<td>Run <code>iocsh.bash</code></td>
</tr>
</tbody>
</table>

\(^{1}\) define modules
**EPICS IOC**

```bash
#!/usr/bin/linux-x86_64/gconpi

epicsEnvSet(P, "ICS")
epicsEnvSet(R, "E3TRNG")
epicsEnvSet("IOC", "$(P):$(R)"
epicsEnvSet("IOCST", "$(IOC):IocStats")

epicsEnvSet("TOP", "../gconpi")
epicsEnvSet("STREAM_PROTOCOL_PATH", ".:${TOP}/db")

cd "${TOP}"

dbLoadDatabase "dbd/gconpi.dbd"
gconpi_registerRecordDeviceDriver pdbbase
drvAsynIPPortConfigure("CGONPI", "127.0.0.1:9999", ...)
dbLoadRecords("db/gconpi-stream.db", "SYSDEV.....")
dbLoadRecords("db/gconpi-admin.db", "IOC=${IOCST}")

cd "${TOP}/iocBoot/${IOC}"
iocInit
```

**e3 IOC**

```bash
source setE3env.bash

put the asyn dependency within stream dependency

require stream,2.7.7
require iocStats,1856ef5

epicsEnvSet(P, "ICS")
epicsEnvSet(R, "E3TRNG")
epicsEnvSet("IOC", "$(P):$(R)"
epicsEnvSet("IOCST", "$(IOC):IocStats")

epicsEnvSet(TOP, "$(E3_CMD_TOP)"
epicsEnvSet("STREAM_PROTOCOL_PATH", ".:${TOP}/db")

drvAsynIPPortConfigure("CGONPI", "127.0.0.1:9999", 0, 0, 0)
dbLoadRecords("${TOP}/db/gconpi-stream.db", "SYSDEV=...")
iocshLoad("${iocStats_DIR}/iocStats.iocsh","IOCNAME=${IOCST}")

iocInit
```
**e3 Anatomy and Requirements**

### Building

- source codes
- configure
- customize
- patch files
- compile
- install

- versioning
- bi-sync with community's work
- track down changes
- kernel driver installation
- userspace libraries
- vendor libraries
- system libraries
- how to get source codes (git, hg, svn, tar, .... )
- various targets
- global configuration
- special makefile
- useful makefile rules

### Static

- directory structure

- multiple NFS shared paths
- decouple production from unused old directories
- transparency from IOCs
- global environment
- local environment
- emergency environment
- easy to duplicate
- define the dependency among base, modules, and others

### Running

- find dynamic loadable libraries, dbd, db files
- load all sources code dependent modules
- define the absolute path within an IOC
- check running base and module versions
- set multiple e3 in any forms
- select global or local environment in multiple options
- simply run iocsh.bash
e3 : Require from PSI, heavily customized one

- Require is an EPICS module with its own Makefile, and more
- ESS require\(^2\) at https://github.com/icshwi/require-ess
- synced with latest changes of the PSI one as much as we can
- to benefit from 10+ years experience of PSI, and customize it to meet the ESS own requirements

<table>
<thead>
<tr>
<th>e3 Anatomy</th>
<th>PSI</th>
<th>ESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Building &amp; Static</td>
<td></td>
<td>e3 Front End</td>
</tr>
<tr>
<td>Static &amp; Running</td>
<td>require</td>
<td>require-ess(^\dagger)</td>
</tr>
<tr>
<td>Building</td>
<td>driver.Makefile</td>
<td>driver.Makefile(^\dagger)</td>
</tr>
<tr>
<td>Running</td>
<td>iocsh</td>
<td>iocsh.bash(^\ddagger)</td>
</tr>
</tbody>
</table>

\(^\dagger\): ESS customized one
\(^\ddagger\): Redesigned and rewritten

\(^2\)from the PSI require https://github.com/paulscherrerinstitute/require
A Real Example e3 Structure for caPutLog

e3-caPutLog (master)$
  ├── caPutLog.Makefile
  │    └── e3 Module Makefile
  │
  │    └── caPutLog
  │         └── Deployment
  │             (git submodule)
  │
  │    └── caPutLog-dev
  │         └── Development (git clone)
  │
  │    └── configure
  │
  │    └── docs
  │
  │    └── iocsh
  │
  │    └── patch
  │
  │    └── tools
  │
  │    └── README.md
  │
  │    └── Makefile

  └── Build Configuration Files / Makefile Rules

In case, we need patch files

modularized startup scripts
Complicated?

$ bash e3TemplateGenerator.bash -m modules_conf/caputlog.conf

caputlog.conf

EPICS_MODULE_NAME:=caPutLog
EPICS_MODULE_URL:=https://github.com/epics-modules
E3_TARGET_URL:=https://github.com/icshwi
E3_MODULE_SRC_PATH:=caPutLog

e3 Tools

https://github.com/icshwi/e3-tools

- e3 Template Generator
- Linux RT PREEMPT Kernel configuration tool
- e3 Release tools
- Others
Modes for build and installation

<table>
<thead>
<tr>
<th>Type</th>
<th>Deployment</th>
<th>Development</th>
<th>Cell³</th>
</tr>
</thead>
<tbody>
<tr>
<td>Files</td>
<td>RELEASE CONFIG_MODULE</td>
<td>RELEASE_DEV CONFIG_MODULE_DEV</td>
<td></td>
</tr>
<tr>
<td>make</td>
<td>init</td>
<td>devinit</td>
<td>both</td>
</tr>
<tr>
<td>make</td>
<td>build</td>
<td>devbuild</td>
<td>both</td>
</tr>
<tr>
<td>make</td>
<td>install</td>
<td>devinstall</td>
<td>cellinstall</td>
</tr>
<tr>
<td>make</td>
<td>vars</td>
<td>devvars</td>
<td>cellvars</td>
</tr>
<tr>
<td>make</td>
<td>uninstall</td>
<td>devuninstall</td>
<td>celluninstall</td>
</tr>
<tr>
<td>make</td>
<td>rebuild</td>
<td>devrebuild</td>
<td>NA</td>
</tr>
</tbody>
</table>

Modes for running an IOC

- Normal: `iocsh.bash`
- Cell: `iocsh.bash -l ${CELLPATH}`
- Realtime: `iocsh.bash -rt`

³Emergency case, usually 3am on-call service, however, it can be used in any development stage.
jhlee@hadron: e3-mrfioc2 (master)$ tree -L 1
.
  ├── [jhlee  4.0K]  cmds
  │    └── [jhlee  4.0K]  configure
  ├── [jhlee  4.0K]  dkms
  ├── [jhlee  4.0K]  docs
  └── [jhlee  4.0K]  iocsh

  ├── [jhlee  4.0K]  mrfioc2
  │    └── [jhlee  4.0K]  mrfioc2-dev
  ├── [jhlee  9.1K]  mrfioc2.Makefile
  ├── [jhlee  4.0K]  opi
  └── [jhlee  4.0K]  patch
      └── [jhlee  1.7K]  README.md
  └── [jhlee  4.0K]  template
      └── [jhlee  4.0K]  tools

11 directories, 3 files

DKMS (Dynamic Kernel Module Support) configuration

https://github.com/javicereijo/mrfioc2

https://github.com/epics-modules/mrfioc2

e3 specific submakefile for mrfioc2

ESS specific mrfioc2 substitutions files

mrf related tools
E3 Front End Example : e3-mrfioc2

jhlee@hadron: e3-mrfioc2 (master)$ tree  template/  
  template/
      ├── [jhlee  4.3K] cpci-evg230-ess.substitutions
      ├── [jhlee  4.2K] evg-cpci-230-ess.substitutions
      ├── [jhlee  33K] evm-mtca-300-ess.substitutions
      ├── [jhlee  7.0K] evr-cpci-230-ess.substitutions
      ├── [jhlee  291] evr-delayModule-ess.substitutions
      ├── [jhlee  403] evr-event-ess.substitutions
      └── [jhlee  17K] evr-mtca-300-ess.substitutions

jhlee@hadron: e3-mrfioc2 (master)$ bash tools/get_pciaddr.bash
Usage: get_pciaddr.bash possible_devices [cpcievr220|cpcievr230|cpcievr300|pcieevr300|mtcaevr300|cpcievg220|cpcievg230|cpcievg300|mtcaevm300]

jhlee@hadron: e3-mrfioc2 (master)$ cd mrfioc2 && git remote -v && cd ..
origin git@github.com:javicereijo/mrfioc2 (fetch)
origin git@github.com:javicereijo/mrfioc2 (push)

jhlee@hadron: e3-mrfioc2 (master)$ cd mrfioc2-dev && git remote -v && cd ..
origin git@github.com:epics-modules/mrfioc2 (fetch)
origin git@github.com:epics-modules/mrfioc2 (push)
E3 Front End Example : e3-mrfioc2

jhlee@hadron: e3-mrfioc2 (master)$ make dep
  require mrfioc2:2.2.0-rc5
  < configured ...
  DEVLIB2_DEP_VERSION = 2.9.0
  > generated ...
  devlib2 2.9.0

jhlee@hadron: e3-mrfioc2 (master)$ make vers
  EPICS_MODULE_TAG:=tags/2.2.0-ess-rc5
  E3_MODULE_VERSION:=2.2.0-rc5

jhlee@hadron: e3-mrfioc2 (master)$ make existent
  /epics/base-3.15.6/require/3.1.0/siteMods/mrfioc2
    2.2.0-rc5
      configure_sequencer_14Hz.bash
      db
      dbd
      evr-mtca-300.iocsh
      evr-pcie-300dc.iocsh
      evr-standalone-mode.iocsh
      include
      lib
      mtca-evr-FP0-output.iocsh
      univ0-ttl-output.iocsh

5 directories, 6 files
E3 Front End Example : e3-mrfioc2

jhlee@hadron: e3-mrfioc2 (master)$ make devdep
  require mrfioc2, develop
< configured ...
  DEVLIB2_DEP_VERSION = 2.9.0
> generated ...
  devlib2 2.9.0

jhlee@hadron: e3-mrfioc2 (master)$ make devvers
  EPICS_MODULE_TAG:=master
  E3_MODULE_VERSION:=develop

jhlee@hadron: e3-mrfioc2 (master)$ make existent LEVEL=1
  /epics/base-3.15.6/require/3.1.0/siteMods/mrfioc2
      └── 2.2.0-rc5
          └── develop

jhlee@hadron: e3-mrfioc2 (master)$ tree -L 2 /epics/base-3.15.6/require/3.1.0/siteMods/mrfioc2/2.2.0-rc5/lib/
    /epics/base-3.15.6/require/3.1.0/siteMods/mrfioc2/2.2.0-rc5/lib/
        └── [root  4.0K]  linux-corei7-poky
            └── [root  1.1M]  libmrfioc2.so
                └── [root   45]  mrfioc2.dep
        └── [root  4.0K]  linux-ppc64le6500
            └── [root  1.4M]  libmrfioc2.so
                └── [root   45]  mrfioc2.dep
        └── [root  4.0K]  linux-x86_64
            └── [root  13M]  libmrfioc2.so
                └── [root   45]  mrfioc2.dep
**Base Support List**

- Base 3.15.5 / **3.15.6**
- Base 7.0.1.1 / 7.0.2 / 7.0.2.1

**Modules Support List**

- Almost all EPICS modules (iocStats, autosave, caPutLog, asyn, busy, modbus, ipmiComm, sequencer, sscan, std, ip, calc, delaygen, StreamDevice, s7plc, recsync, MCoreUtils, devlib2, mrfioc2, motor, ecmc, ethercatmc, ....)

- Area Detector (ADSupport, ADCore, ADSimDetector, ADCSimDetector, NDDRiverStdArrays, ADAndor, ADAndor3, ADPointGrey, ADProsilica, ...)

- EPICS V4 modules (pvData, pvAccess, pva2pva, pvDatabase, normativeTypes, pvaClient) for Base 3.15.X
Where E3 is

Clone It Today!

```
git clone https://github.com/icshwi/e3
```

```
git clone https://github.com/icshwi/e3training
```

Building and Running Tested on

- ESS Yocto Linux (Normal & PEEMPT RT Kernels and linux-corei7-poky, linux-ppc64e6500, & linux-x86_64)
  `https://gitlab.esss.lu.se/icshwi/yocto-ess`
- CentOS Linux (Normal and CERN PREEMPT RT Kernels)
- Debian Linux (Normal and PREEMPT RT Kernels)
- Raspbian Stretch, Ubuntu, LinuxMint, Fedora, and Arch Linux
Summary

- ESS has its own EPICS environment for ESS and many in-kind collaborations across Europe.
- Tried to cover various scenarios which we usually meet in different space-time domains in order to resolve issues consistently.
- Looks very promising when we work within the local environment and development phase.
Outlook

- We are integrating e3 with NFS, procServ, systemd, conserver, site-wide configuration & life-cycle managements, and so on for final production.
- The early discussion within the system engineering procedure is in progress and is shown in the next ...
Hmm, control, control. You must learn control!

Yoda (The Empire Strikes Back)
Merci!
감사합니다!
Kiitos!
ありがとうございます!
Thank you!
Dankeschön!
Tack!
😊