Contribution to Nominal Device Support V3 for standardizing device drivers

Moreno, Javier¹ (jmoreno@gmv.com); Ruiz, Mariano²; Melis, Stefano¹; de Gracia, Alberto²; Sanz, Diego¹; Esquembri, Sergio²; Astrain, Miguel²; Lange, Ralph³

¹ GMV
² UPM. Technical University of Madrid, Instrumentation and Applied Acoustic Research Group
³ ITER International Organization
INDEX

- Motivations
- NDS overview
- Contribution to NDS3
- Use cases
- Conclusions and references
CONTRIBUTION TO NOMINAL DEVICE SUPPORT V3 FOR STANDARDIZING DEVICE DRIVERS
The higher the use of EPICS, the more support layers appear to ease IOC development

Standardization claims to minimize IOC development efforts

`asyn` drivers are typically more friendly than just `device` support

`areaDetector` is based on `asyn` to interface with detectors and cameras, as well as devices providing waveforms

`NDS` is a generic solution for device driver development
NOMINAL DEVICE SUPPORT

- NDS introduced the Device Model
- Based on *asyn* to provide common interface to EPICS IOCs
- Device drivers based on:
  - Device
    - State machine
  - ChannelGroup
  - Channel
    - Image
    - Analog/Digital Input/Output
- EPICS Libraries developed in C++
NOMINAL DEVICE SUPPORT V3

- NDS3 drivers do not depend on EPICS*
- Based on its own Control System (Factory) with a common interface to IOCs
- Device drivers based on:
  - Node
  - Port
  - State machine
  - PVs
  - Input
  - Output
- C++11 libraries
The NDS Control System takes care of:

- Registering device drivers
- Creating device drivers instances
- Communicating device drivers with rest of world:
  - Setting values
  - Getting values
  - Publishing values
  - Data sharing

Control System classes inherit from the target system. For instance: `asynPortDriver`
NDS3 drivers do not depend on EPICS* ...

* Unless NDS-EPICS is considered
From a logical point of view:

- Existing elements:
  - nds-core
  - nds-epics
  - state machine

- External elements to interface with:
  - asyn
  - ITER-DAN
  - ITER-NISYNC
  - ITER-SDN

- Added support for:
  - Firmware
  - Routing
  - Triggering and clocking
  - Data Acquisition
  - Waveform Generation
  - Digital IO
Current class diagram:

- **DataAcquisition**: acquiring data
- **DigitalIO**: handling digital signals
- **FTE**: handling Future Time Events
- **Firmware**: providing typical parameters
- **HQMonitor**: providing Health and Quality parameters
- **Port**: providing communication with the control system
- **Routing**: routing signals
- **StateMachine**: handling states
- **Timestamping**: handling timestamps
- **Timing**: providing timing parameters
- **WaveformGeneration**: generation of waveforms
- **TriggerAndClk**: generating trigger and clock signals
CONTRIBUTION TO NDS3

An User may combine NDS PVs and Nodes to develop NDS drivers according to their SRS.

NDS PVs

NDS Nodes: Port, DAQ, IMAQ, State Machine,...

NDS3-Core System

NDS3-Core System

EPICS Record Database Template

EPICS IOC Application

Standard Application

A developer maintains the NDS3-core and provides it new features.

NDS PVs and Nodes are the public elements of the NDS3-core whose pre-defined functionalities must be implemented in their instantiation.
Additionally:

- Documentation: User Manuals, API, Test Plan, Software Architecture and Design Document
- Sample device drivers
- Unit tests with PyEpics

Upcoming features:

- DAN nodes
- SDN nodes
- Data Scheduling nodes
- Removing dependencies from device drivers with data sharing
Developed device drivers – Timing: PTM-1588 (GMV), NI PXI-6683 (UPM)

**USE CASES**

<table>
<thead>
<tr>
<th><strong>PTM-1588</strong></th>
<th><strong>MTCA</strong></th>
<th><strong>NISYNC</strong></th>
<th><strong>PXI</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>Terminal [1..K] Node</td>
<td>State Machine</td>
<td>PVs</td>
</tr>
<tr>
<td>State Machine</td>
<td>PVs</td>
<td>Firmware</td>
<td>FTE</td>
</tr>
<tr>
<td>PVs</td>
<td>Timestamping</td>
<td>MTCA NISYNC Port</td>
<td>State Machine</td>
</tr>
</tbody>
</table>

**Routing**

- Timing

**FTE**

- Firmware

**PVs**

- State Machine

**Timestamping**

- Node
Developed device drivers – DAQ: NI FlexRIO-5761 (UPM)

**USE CASES**

**IRIO Port**

- **State Machine**
- **Firmware**
- **HQ Monitor**
- **irioDMA [1..K] Node**
- **AI Static Group Node**
- **AO Static Group Node**
- **DI Group Node**
- **DO Group Node**
- **WFG Channel [1..L] Node**

**PVs**
- **Data Acquisition**
- **State Machine**
- **AI Static Channel [1..M] Node**
- **State Machine**
- **AO Static Channel [1..N] Node**
- **PVs**
- **Digital IO [1..P]**
- **PVs**
- **Digital IO [1..R]**
Developed device drivers – DAQ: NI PXIe-6363 (UPM)

USE CASES

NI-DAQX Port

State Machine | PVs | Firmware | HQ Monitor | Trig & CLK | Routing | AI Segment Node | DI Segment Node | DIO PFI Node | AO Segment Node | Data Acquisition [1..K]

State Machine | PVs | DAQ Channel [1..L] | Digital IO | Digital IO | State Machine | PVs | WFG Channel [1..M]
CONCLUSIONS AND REFERENCES

Conclusions:
- Standardized development of device drivers
- Independent Control System device drivers
- Easy usability
- High scalability

Suggestions:
- Adding support for image acquisition with areaDetector

References:
- Cosylab’s repositories: NDS3 and NDS3-EPICS
- ITER’s repositories: NDS3-CORE and NDS3-EPICS

Acknowledgments:
- Technical University of Madrid, Instrumentation and Applied Acoustic Research Group
- ITER International Organization
THANK YOU