Implementation of the Nominal Device Support in ITER CODAC Core System

Status and Plans

M. Ruiz, M. Astrain, R. Lange, S. Esquembri, V. Costa,
J. Moreno, D. Sanz, D. Rivilla

Universidad Politécnica de Madrid

ITER Organization

GMV Aerospace and Defense

Outline

- Motivation
- Nominal device support (NDS) concept
- Main NDS software layers
- NDS device drivers for PXIe and MTCA devices
- NDS plugins for SDN, DAN and PVXS
- The NDS-SYSTEM
- Other NDS solutions: NDS-IRIO-OpenCL
- Conclusions

Motivations

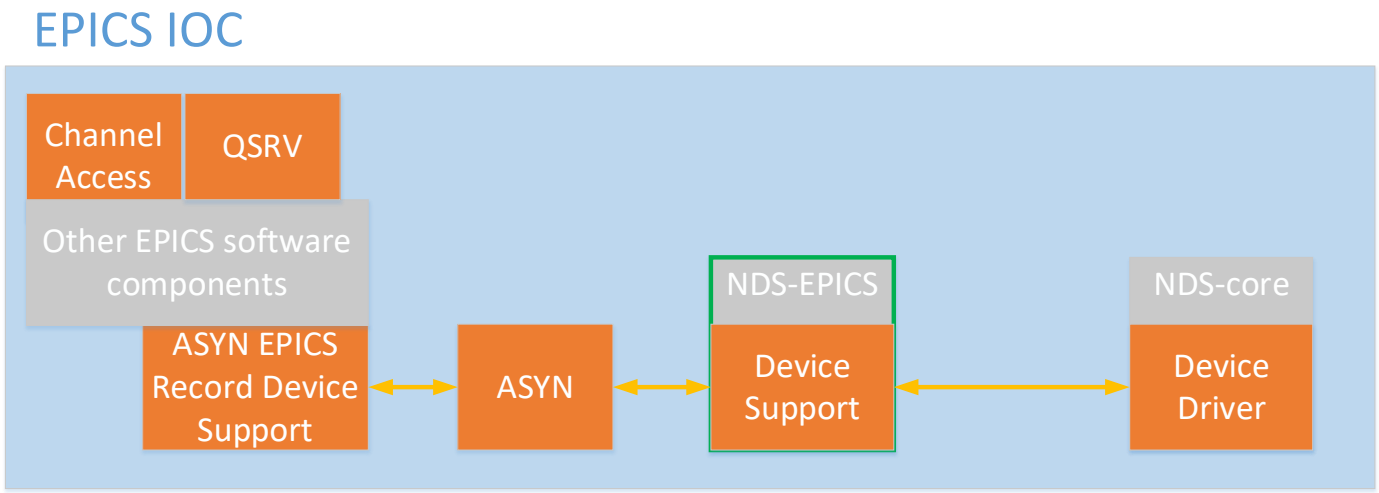
- Project originally developed by Cosylab but improved and extended by ITER, UPM and GMV
- Create a driver development software framework for diagnostics measurement systems (focusing on DAQ)
  - Small functional blocks (nodes) are instantiated and configured to build complex systems
  - Improves code reusability and testability
  - High software quality (automated tests, static code analysis)
  - Good user & developer level documentation
- Provide a generic solution for device driver development in I&C systems using EPICS and other control systems (RTF)

NDS-core

- C++ library (std C++ 11). NDS does not have dependencies of EPICS source code
- Windows/Linux
- Device drivers are organized in a collection of hierarchical nodes that contain:
  - NDS PVs (In, Out, DelegateIn, DelegateOut)
  - State machines
  - Other nodes..
- Predefined complex nodes with already defined NDS-PVs:
  - Analog Input, Waveform generation, Digital I/O
  - Timestamp & Future Time Event Generation
  - Management of trigger and clocks
- These nodes have all the NDS-PVs identified and with a predefined functionality
- Hardware devices with the same functionalities from different manufactures have the same NDS driver organization and NDS-PVs
- Any NDS PV can be replicated and subscribed by other devices
- NDS device drivers are deployed as plug-in libraries –minimizing recompilation of the project
- NDS device drivers can be accessed through a control system API (ie: EPICS, RTF, or other C++ applications)

NDS-EPICS

- Standard software module to connect any NDS device driver to EPICS
- NDS-EPICS uses “asynDriver” to interface with EPICS
- NDS-EPICS layer can be extended to support specific functionalities for your specific device driver when it is used in EPICS (inithooks, etc)
- Templates for records of complex nodes already defined
- NDS-EPICS layer is tested using pyepics



NDS for PXIe devices

- Support for devices in the ITER fast controller catalog
- PXI NDS device drivers use the kernel modules available for CODAC Core System

✓ PXI6683H:
  - PTP timing card
  - timestamping and future time event generation of external signals (PFI) and PXITRIG

✓ X-SERIES: Multifunction devices
  - PXIe6363and PXIe6368: Multiplexed and Simultaneous AI DAQ
  - All functionalities supported (DIO, DAC, Trigger and Clock) with the exception of the Digital Counters

✓ FlexRIO devices with NI5761 adapter module

✓ Teledyne ADQ14 (UKAEA-JET/MAST)

NDS for MTCA devices

- ✓ PTM1588 (product developed by DMCS, Poland)
  - PTP timing card
  - timestamping and future time event generation of external signals (PFI) and “MTCA clocks”
- ✓ MFMC MTCA.4 FMC Carrier Module (DMCS)
  - DAQ device
- ✓ Teledyne ADQ14
- ✓ IOxOS IFC_1410 (ESS)
- ✓ Struck SIS8300 (ESS migration from NDSv2 to NDSv3)

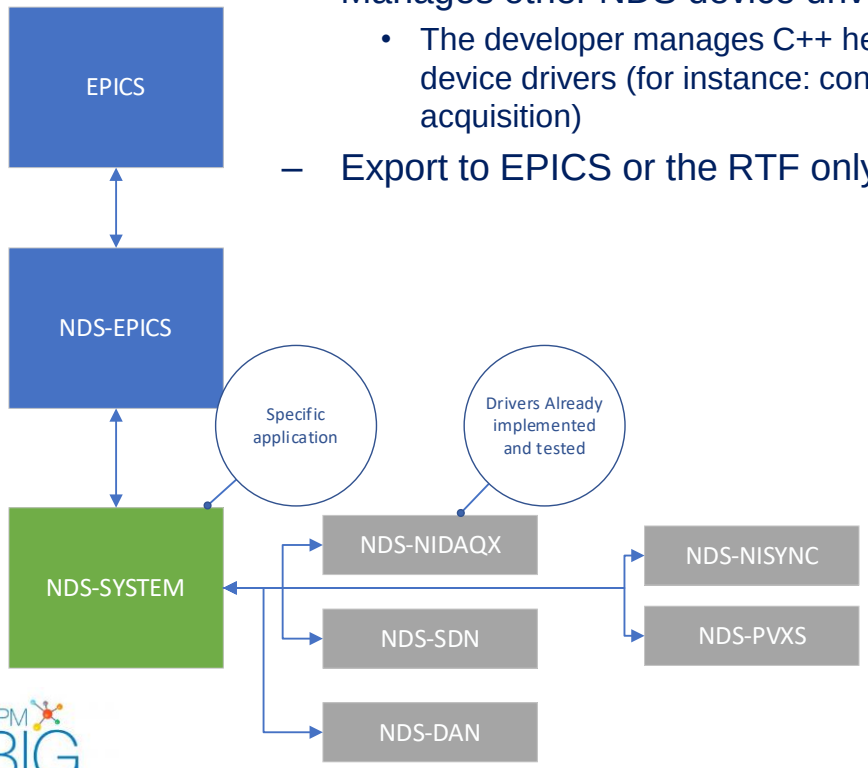
ADQ14DC–4C–MTCA

NDS-plugins: NDS-device drivers that implement a specific functionality

- NDS-SDN (publisher): ITER Synchronous Data Network implemented in SDN-core API
  - The user defines the SDN topic in an xml file and another XML connecting NDS-PVs with the attributes of the SDN topic
  - The SDN topic can be published periodically or when a specific NDS-PV is updated
- NDS-PVXS: Management of PVaccess/Pvdata to put/get/monitor EPICS PVs and for doing pvAccess RPC calls
  - PVXS: next generation user library for pvAccess
  - Generic PVXS node (can be added to NDS system)
  - Supports all access modes: read, write, monitor, RPC
  - Implementation similar to SDN node:
    - XML configuration from file
    - Uses the NDS data flow engine
  - Covers cases where NDS systems need to
    - include EPICS data in e.g. their SDN/DAN output
    - control COTS devices through regular EPICS integration
- NDS-DAN: ITER data archiving
  - The user defines in an XML configuration file the ND-PVs (data source) that connects with DAN streamers
  - The NDS-DAN module publishes automatically the data using the DAN API

NDS-system

- What is an NDS-system?
  - Is an NDS device driver developed to manage other NDS device drivers already implemented
- What is the functionality of the NDS-system?
  - Implement the state machine of an specific “experiment”
  - Manages other NDS device drivers using the subscription/replication utility of NDS-core
    - The developer manages C++ helper classes that encapsulates complex functions of the NDS-device drivers (for instance: configure a trigger, an specific waveform generation or data acquisition)
  - Export to EPICS or the RTF only the PVs that are needed for an specific experiment



- NDS-NISYNC: Timing and triggering, Backplane routing
- PXIe 6368: DAQ, Waveform generation, digital IO
- PVXS: accessing data on external EPICS IOC
- SDN: combining acquired data, timestamp, external data in a published SDN topic
- Hierarchical state machine to control the system
- OPI panel to interact with the NDS-System
- Archiving to DAN (Available Q3/Q4 2021)

NDS for FPGA-based systems

- UPM in-house independent development
- Identified need: development of advanced data acquisition and processing system using SoCs and FPGAs
- NDS-IRIO-OpenCl.
  - Generic NDS device driver supporting the interface with FPGA devices using the OpenCL standard
  - Two platforms:
    - FPGA with PCIe interface in a INTEL host computer (OpenCL SDK for IntelFPGA)
    - SOCs (ARM processor + FPGA): XILINX (VITIS) and IntelFpga (OpenCL SDK)
  - Integration of traditional DAQ applications and HW accelerators for data processing
  - Adding machine learning applications

Results/conclusions

- NDS environment:
  - Already developed and tested. Documentation available
  - Mainly tested in CODAC Core System
- NDS Developers and Users
  - ITER (IO through contractor DMCS as well as Das)
  - European Spallation Soruce
  - UKAEA
  - ASDEX Upgrade
  - Universidad Politécnica de Madrid
- Code available
  - Github (Cosylab's original public development of nds-core and nds-epics)
    - (https://github.com/Cosylab/nds3)
  - ITER GIT (everything, needs ITER account)

Implementation of the Nominal Device Support in ITER CODAC Core System

Status and Plans

Thanks for your attention!!!

mariano.ruiz@upm.es

M. Ruiz, M. Astrain, R. Lange, S. Esquembri, V. Costa,
J. Moreno, D. Sanz, D. Rivilla

Universidad Politécnica de Madrid

ITER Organization

GMV Aerospace and Defense

Backup

Develop a new device driver

- NDS-CORE
  - How to develop a new NDS-driver?
    - Define the hierarchy that better match your hardware device organization
    - Create the basic driver structure using the complex-nodes
    - Define the behaviour of STMs
    - Complete the code to connect NDS-PVs with your specific hardware API
  - Test the driver: Develop specific tests using “google tests”
  - Driver Documentation:
    - NDS uses Doxygen
- NDS-EPICS (How to add support for EPICS?)
  - Customize the available template, basically add the records for the specific PVs of your driver
  - Add initialization in “st.cmd”
  - Test the IOC using PyEpics
- How to add it to ITER SDD-Editor?
  - Customize the SDD templates