Standardization of software device drivers implementation for data acquisition and timing devices in ITER CODAC Core System: Nominal Device Support

M. Ruiz[1], J. Moreno[2], A. de Gracia[1], S. Melis[2], S. Esquembri[1], D. Sanz[2], M. Astrain[1], R. Lange[3]

[1] Instrumentation and Applied Acoustic Research Group, Technical University of Madrid, Madrid, Spain

[2] GMV Aerospace and Defence S.A.U., Isaac Newton 11, P.T.M. Tres Cantos, 28760, Madrid, Spain

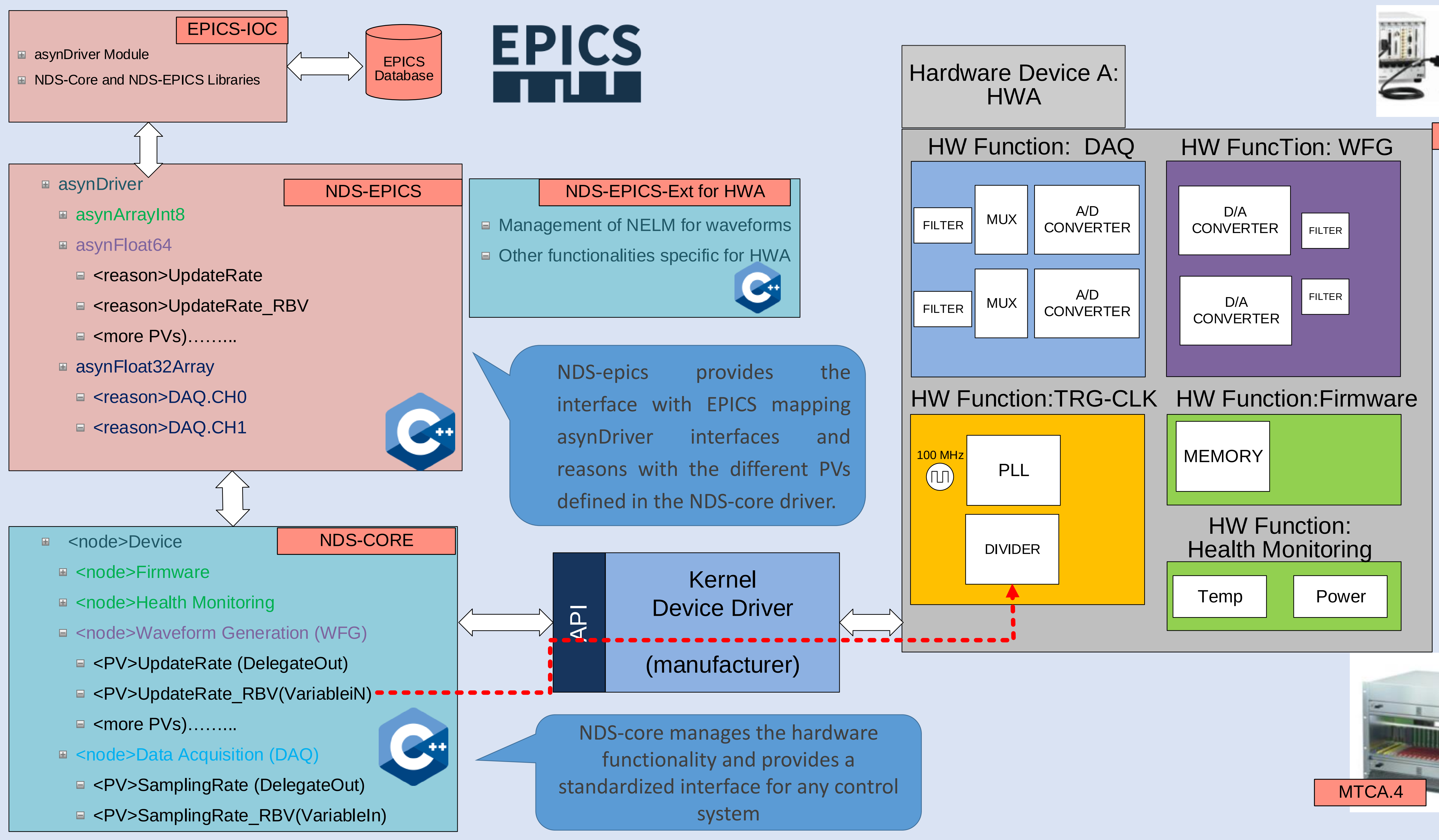
[3] ITER International Organization St. Paul lez Durance, France

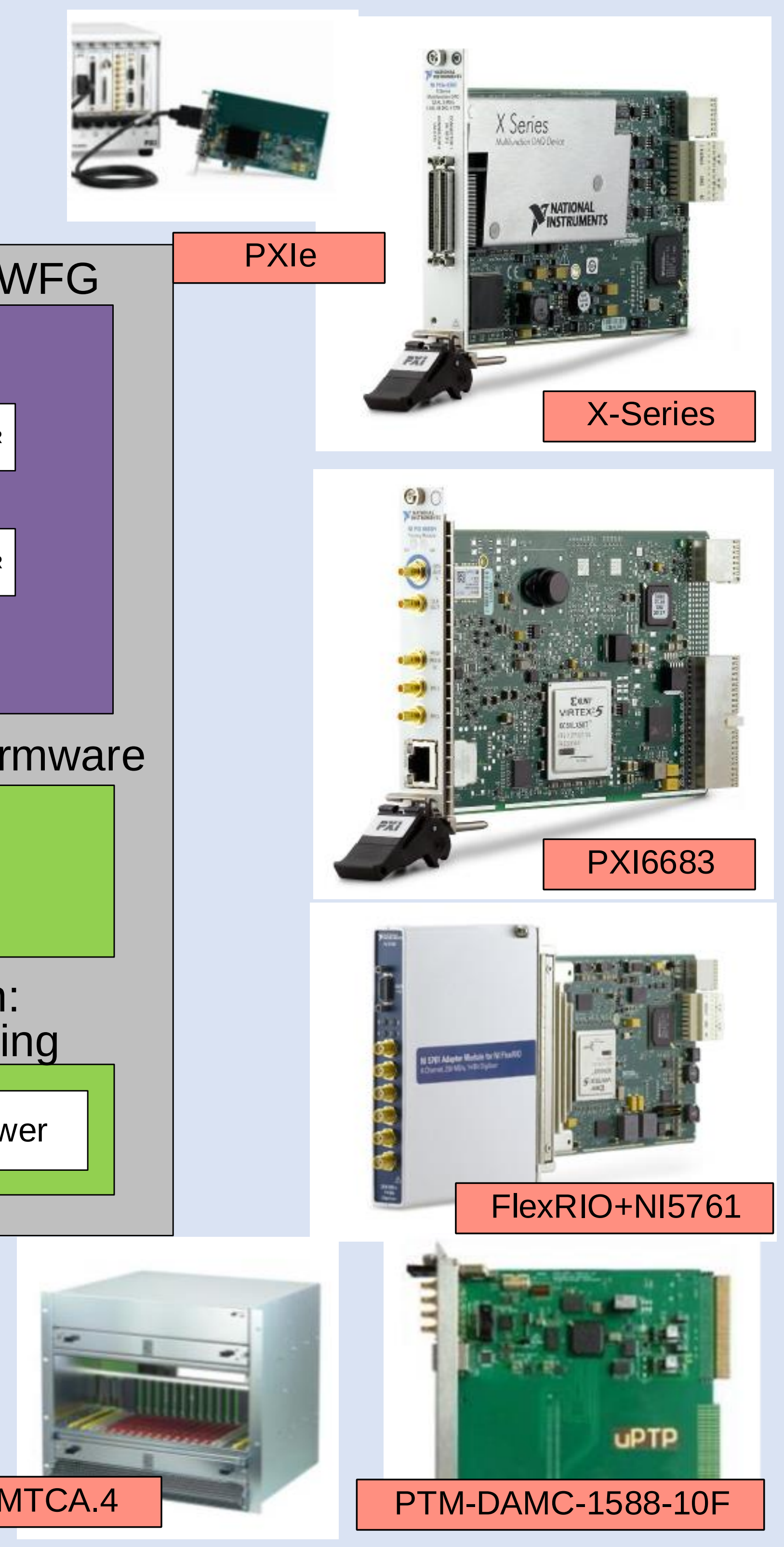
mariano.ruiz@upm.es

ID: 479

SUMMARY

The **standardization of device driver's implementation in distributed control systems** is essential to reduce the development and integration effort. Nowadays, there are multiple distributed control systems and high-level applications used in the scientific community. Due to the lack of standardization in the interfaces with the device drivers, it is required to integrate each hardware device separately in each control system. However, a clear model of how different software applications should manage the device drivers could reduce development efforts and ease the maintainability of such complex systems. The **Nominal Device Support initiative is an open source project**, for the standardization of data acquisition and timing devices. The first implementations of NDS (version 1.x and 2.x) were oriented to simplify the implementation of EPICS device supports. **NDS v3 is more ambitious and is built as an abstraction layer for any control system or software application.**


EPICS-IOC
asynDriver Module
NDS-Core and NDS-EPICS Libraries
EPICS Database
EPICS
asynDriver
NDS-EPICS
asynArrayInt8
asynFloat64
<reason>UpdateRate
<reason>UpdateRate_RBV
<more PVs)……..
asynFloat32Array
<reason>DAQ.CH0
<reason>DAQ.CH1
NDS-EPICS-Ext for HWA
Management of NELM for waveforms
Other functionalities specific for HWA
NDS-epics provides the interface with EPICS mapping asynDriver interfaces and reasons with the different PVs defined in the NDS-core driver.
<node>Device
NDS-CORE
<node>Firmware
<node>Health Monitoring
<node>Waveform Generation (WFG)
<PV>UpdateRate (DelegateOut)
<PV>UpdateRate_RBV(VariableIN)
<more PVs)……..
<node>Data Acquisition (DAQ)
<PV>SamplingRate (DelegateOut)
<PV>SamplingRate_RBV(VariableIn)
API
Kernel Device Driver
(manufacturer)
NDS-core manages the hardware functionality and provides a standardized interface for any control system
Hardware Device A: HWA
HW Function: DAQ
FILTER
MUX
A/D CONVERTER
HW FuncTion: WFG
D/A CONVERTER
FILTER
HW Function:TRG-CLK
100 MHz
PLL
DIVIDER
HW Function:Firmware
MEMORY
HW Function: Health Monitoring
Temp
Power



MAIN FEATURES OF NDS-CORE LAYER

- Software layer implemented in C++ in charge of interfacing with the hardware using the device driver provided by the manufacturer.
- The driver manages the functionality using a hierarchical organization in nodes, called NDS-nodes. Each NDS-node manages an specific hardware block (ie: firmware, ADC, DAC. Timing, clocking, triggering, timestamping, future timing event, etc.)
- An NDS-node provides a list of defined process variables in charge of the configuration and management of the specific hardware functionality.
- The NDS PVs are classified as VariableIn/Out and DelegateIn/Out. NDS PVs implement function to read and write its specific values. Additionally, DelegatePVs trigger the execution of specific methods when are written or read.
- NDS-core provides method to subscribe and replicate PVs. This option allows to share data among different NDS Device Drivers.
- An NDS device driver is implemented using NDS-core and is available for the user as a library (shared or static) to be used in any application.
- NDS-core layer has been tested using google test. A simplified test control system is provided with the test code.

ACKNOWLEDGEMENTS

This work was supported in part by the Spanish Ministry of Economy and Competitiveness, Projects Nº ENE2015-64914-C3-3-R and Madrid regional government (YEI fund), Grant Nº PEJD-2018-PRE/TIC-8571.

MAIN FEATURES OF NDS-EPICS LAYER

- A unique software layer implemented in C++ to interface multiple NDS-core device drivers to EPICS control system. This approach simplifies the code maintainability and testability.
- EPICS PVs are mapped (connected) to NDS-core PVs. Every action to read and write the asynDriver interface triggers the specific read/write function in the NDS-core PV.
- NDS-core Push functions are mapped to asynDriver interrupts to send data to EPICS.
- The user only needs to configure the EPICS database file using templates and substitutions file to manage an specific NDS-Device driver.
- The NDS-EPICS layer can be extended by the user to support additional functionalities if this are need by the NDS-device driver (ie: use of number of elements (NELM) for waveforms).
- NDS-EPICS tests has been implemented using PyEPICS.

RESULTS

- Integrated in ITER CODAC Core System 6.0 (NDSv 3.2.0.0)
- Drivers available for PXIe Devices: PXI6683 (timing card), PXIe6363 (X-Series DAQ Devices) and FlexRIO with analog input module NI5761
- Drivers available for MTCA devices: PTM1588 (timing card)
- Version with new functionalities in Q4 2019 (v3.3.0.0)
- Support for ITER Data Archiving Network and Synchronous Data Network

POLITÉCNICA

UPM BIG SCIENCE

gmv INNOVATING SOLUTIONS

GRUPO DE INVESTIGACIÓN EN INSTRUMENTACIÓN Y ACÚSTICA APLICADA

iter china eu india japan korea russia usa