O/2-4

MARTe2 and MDSplus Integration for a Comprehensive Fast Control and Data Acquisition System

G.Manduchi[1], A. Rigoni[1], T.Fredian[2], J.Stillerman[2], A. Neto[3], F. Sartori[3]

gabriele.manduchi@igi.cnr.it

1) Consorzio RFX, Corso Stati Uniti 4, 35127 Padova, Italy
2) Massachusetts Institute of Technology, 175 Albany Street, Cambridge, MA 02139, United States
3) Fusion for Energy, Barcelona, Spain

MARTe2 and MDSplus Integration for a Comprehensive Fast Control and Data Acquisition System

G.Manduchi[1], A. Rigoni[1], T.Fredian[2], J.Stillerman[2], A. Neto[3], F. Sartori[3]

1) Consorzio RFX, Corso Stati Uniti 4, 35127 Padova, Italy

2) Massachusetts Institute of Technology, 175 Albany Street, Cambridge, MA 02139, United States

3) Fusion for Energy, Barcelona, Spain

ACKNOWLEDGMENT
The work leading to this presentation has been funded partially by Fusion for Energy under work programs WP2017/18/19. This publication reflects the views only of the authors, and Fusion for Energy and ITER Organization cannot be held responsible for any use, which may be made of the information contained therein. The views and opinions expressed herein do not necessarily reflect those of the ITER Organization.

Data Acquisition and Real-Time control in long lasting experiments

- In the past two different HW and SW solutions
  - Fast computation and low latency for real-time control
  - Bulk transfer and high data throughput for data acquisition
- Not anymore valid when streaming data in long lasting experiments
  - Current bus and disk technology allow managing high speed data movement from ADC to disk
  - Availability of different cores on computer allows co-existence of real-time tasks with other system activities
- Same Hardware prescribed in ITER for data acquisition and real-time control
  - The only difference in underlying bus (DAN, SDN)

Uniformity in Hardware means Uniformity in Software?

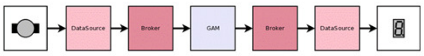
- Separate Frameworks have been traditionally used for real-time control and data acquisition
- MARTe and MDSplus have already been integrated in the past
  - MDSplus used to store data produced by MARTe
  - Configuration data and reference waveforms used in MARTe retrieved from MDSplus pulse files
- Data Plumbing implemented BUT the two systems were mainly independent
  - Two systems to be learnt, maintained and configured in day-per-day operation

MARTe => MARTe2

- MARTe2 is a completely re-written version of MARTe developed under strict quality standards
  - MISRA compliance
  - Full test units for largest code coverage
- MARTe2 improves MARTe platform abstraction
  - From bare-metal microcontrollers to full fledged Linux
  - OS abstraction performed at several layers, with or without threads
- MARTe2 introduces a new and more powerful system abstraction
  - MARTe Generic Application Modules (GAMs) now enriched with two new components: **DataSources** and **Brokers**

MARTe2 Data Sources and Brokers

- In the former MARTe GAMs were associated to real-time threads and exchanged data in shared memory
  - I/O carried out by specialized GAMs
- In MARTe2 a GAM can only exchange data with DataSource components
  - Data Sources can implement memory buffers or I/O devices
- A step further: Broker objects manage data exchange between GAMs and Data Sources
  - Broker not exposed in the configuration, but chosen by the Data



Data Sources and Brokers
A way to complicate one's life?

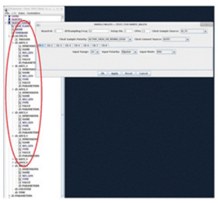
- The answer is definitely: **NO**
- Three logical components each mapping an activity in real-time control
  - GAM => The ***Algorithm***
  - DataSource => The management of ***Data***
  - Broker => The management of ***Data Flow***
- Data flow can be (among others):
  - ***Plain***, i.e. just copy in memory. In this case Data Source implements just the buffer and the broker the copy
  - ***Synchronized***, where the broker triggers some action like ADC sampling, DAC output.
  - ***Decoupled***, to handle non real-time storage of real-time data stream. The broker will handle buffering and the management of a separate thread

MDSplus Data Plumbing in MARTe2

- Carried out by two DataSource implementations: **MDSReader** and **MDSWriter**
  - ***MDSReader*** will load reference waveforms in memory and will return appropriate sample whenever the corresponding broker (*MemoryMapSynchronizedInputBroker*) is executed
  - ***MDSWriter*** receives data decoupled from real-time threads thanks to the associated Data Broker (*MemoryMapAsynchOutputBroker*)
- Nevertheless the two systems are still mostly independent, and two different configurations must be provided
  - MDSplus experiment model for Data Acquisition configuration
  - MARTe2 configuration file for the definition of the real-time components (GAMs, Data Sources, Threads)

MDSplus Devices

- Model the different object instances in the experiment database
- A ***Device*** is the container of set of related data (e.g. to describe a piece pf HW)
  - A subtree in the data hierarchy associated with every device instance
- Devices are similar to classes and bring a data structure (a subtree) and a set of methods
- A constructor method will instantiate the corresponding data set when the experiment database is built
- A Setup Method will be invoked by the graphical browser to show the content of the corresponding instance
- Other methods will carry out device specific functions (INIT, STORE)



Adding a new device in MDSplus

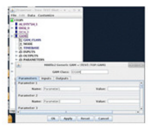
- MDSplus devices are mapped against ***python classes***
- Developing a new device means developing a new python class that inherits fro ***class Device***
- All the required interaction with MDSplus is carried out by the superlass. The new device has to:
  - Declare the structure of the underlying subtree by means of a ***python dictionary***
  - Implement device specific methods. Associated data items are available as instance fields
- **It is therefore natural to import MARTe2 configuration as a set of devices**

Mapping MARTe2 components into MDSplus devices

- A straight mapping is not the best approach.
- It is possible to provide a high level view of the system, including all the required information, and then generating on the fly the corresponding MARTe2 configuration
- The system can be described by the following devices:
  - MARTE2_GAM: describing a computation carried out in the system
  - MARTE2_IN: describing an input device
  - MARTE2_OUT: describing an output device
- Data flow will be specified by input/output node references in the associated device fields
  - Naturally expressed in MDSplus by means of ***Expressions***
- All data handled by the system will be stored in the pulse file, providing a complete picture of the system behavior.

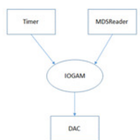
Use case: A simulation program

- Implemented by a (subclass of) MARTE2_GAM device
- It specifies the Parameters, the Inputs and the Outputs
- Input fields refer to stored input signals that are read from the pulse file by means of the generated Data Source instance
- The generated MARTe2 components include a MDSWriter instance to store results in the pulse file



Use case: Waveform generation

- Implemented by a (subclass of) MARTE2_OUT
- It specifies the Parameters and the Inputs.
- Input fields refer to stored input signals that are read from the pulse file by means of the generated Data Source instance
- The generated MARTe2 components include a MDSReader instance to read data from the pulse file, and a specific DAC DataSource for waveform generation.



Use case: Control Loop

- Implemented by a set of MARTE2_IN, MARTE2_GAM and MARTE2_OUT instances
- Input fields of MARTE2_GAM instance will contain a reference to output fields of MARTE2_IN
- Input fields of MARTE2_OUT will refer to output fields of MARTE2_GAM
- The generated MARTe2 components include two MDSWriter instances to write data into the pulse file, one DAC DataSource for waveform generation, and one ADC DataSource for Data Acquisition.

Conclusions

- The proposed approach provides a full integration of MARTe2 and MDSplus
- A subset of the possible MARTe2 configurations can be described in this way
  - However it covers the use cases of interest
- A real world MARTe2 configuration file is composed of many thousands of lines, and editing it manually is impossible
- Users do not need a detailed MARTe2 knowledge for system configuration
- Developers can easily wrap MARTe2 DataSources and GAMs into MDSplus devices by inheriting from the python superclasses MARTE2_GAM, MARTE2_IN and MARTE2_OUT.
- The final target will be the generation of MARTe2 GAM and MDSplus MARTE2_GAM device directly from Simulink.