EFDA–JET–CP(11)06/05

A. Neto, D. Alves, A. Barbalace, L. Boncagni, B.B. Carvalho, P.J. Carvalho,
G. De Tommasi, H. Fernandes, G. Manduchi, P. McCullen, F. Sartori,
A. Stephen, D. Valcárcel, R. Vitelli, L. Zabeo,
and JET EFDA contributors

MARTe Framework: A Middleware for Real-Time Applications Development

MARTe Framework: A Middleware for Real-Time Applications Development

A. Neto[1], D. Alves[1], A. Barbalace[2], L. Boncagni[3], B.B. Carvalho[1], P.J. Carvalho[1], G. De Tommasi[4], H. Fernandes[1], G. Manduchi[2], P. McCullen[5], F. Sartori[6], A. Stephen[5], D. Valcárcel[1], R. Vitelli[7], L. Zabeo[8], and JET EFDA contributors*

JET-EFDA, Culham Science Centre, OX14 3DB, Abingdon, UK

[1]Associação EURATOM/IST, Instituto de Plasmas e Fusão Nuclear Laboratório Associado, 1049-001, Lisboa, Portugal
[2]EURATOM-ENEA Association, Consorzio RFX, 35127 Padova, Italy
[3]EURATOM-ENEA sulla Fusione, C.R. ENEA Frascati, I-00044 Frascati-Rome, Italy
[4]Associazione EURATOM-ENEA-CREATE, Via Claudio 21, 80125, Napoli, Italy
[5]EURATOM-CCFE Fusion Association, Culham Science Centre, OX14 3DB, Abingdon, OXON, UK
[6]Fusion for Energy, Barcelona, Spain
[7]Universit`a di Roma, Tor Vergata, Via del Politecnico 1-00133, Roma, Italy
[8]ITER Organisation, Cadarache, France
** See annex of F. Romanelli et al, "Overview of JET Results", (23rd IAEA Fusion Energy Conference, Daejon, Republic of Korea (2010)).*

Preprint of Paper to be submitted for publication in Proceedings of the 13th International Conference on Accelerator and Large Experimental Physics Control Systems, Grenoble, France
10th October 2011 - 14th October 2011

ABSTRACT
The Multi-threaded Application Real-Time executor (MARTe) is a C++ framework that provides a development environment for the design and deployment of real-time applications, e.g. control systems. The kernel of MARTe comprises a set of data-driven independent blocks, connected using a shared bus. This modular design enforces a clear boundary between algorithms, hardware interaction and system configuration.

The architecture, being multi-platform, facilitates the test and commissioning of new systems, enabling the execution of plant models in offline environments and with the hardware-in-the-loop, whilst also providing a set of nonintrusive introspection and logging facilities. Furthermore, applications can be developed in non real-time environments and deployed in a real-time operating system, using exactly the same code and configuration data.

The framework is already being used in several fusion experiments, with control cycles ranging from 50 microseconds to 10 milliseconds exhibiting jitters of less than 2%, using VxWorks, RTAI or Linux. Codes can also be developed and executed in Microsoft Windows, Solaris and Mac OS X.

This paper discusses the main design concepts of MARTe, in particular the architectural choices which enabled the combination of real-time accuracy, performance and robustness with complex and modular data driven applications.

1. INTRODUCTION

MARTe [1] is a framework tailored at the design and development of real-time control systems. The kernel of MARTe uses a C++ multi-platform library named BaseLib2 [2]. The main ideas behind the original design of the framework were the modularity and portability of its applications. In particular, a strong effort was made in order to allow a robust simulation environment that allows both the models to be simulated with the hardware in the loop and the control algorithms to be validated offline.

This is achieved by providing a clear development boundary between the algorithms, hardware and system configuration. Being multi-platform it also allows to debug and develop in non-real-time environments, where better developing tools are usually available. Currently the framework runs in Linux, Linux with RTAI [3], VxWorks® forPowerPC®, Solaris® and Microsoft® Windows™.

The framework components are configured using a common language, designed to be as simple as possible, but complete enough to provide a clear way of describing the problem. The structure is similar to XML, where the syntax rules are validated by a BaseLib2 parser, whereas the actual validity of the arguments is performed by the component (i.e. no validation schema is available). An example of a configuration is shown in Listing 1.

The configuration file is translated into a database of named objects that can be browsed using the object addresses in the database. By parsing the configuration file, the framework automatically creates and configures instances of all the declared objects. A messaging mechanism uses the database to provide a standard interface for communication between objects. This is the preferred mechanism to interfaceMARTe objects with other systems and protocols.

2. MAIN COMPONENTS

A MARTe application is designed by configuring and connecting a series of blocks named Generic Application Modules (GAM). These modules contain an entry point to receive data driven configuration and a set of optional input and output channels to interface with other GAMs. Each of these channels has a unique name and is connected to a generic memory data pipeline, named Dynamic Data Buffer (DDB). Before starting the execution of the application, MARTe guarantees the coherency of the DDB by checking that all GAMs have the requested inputs being produced by another GAM. This scheme enables to design interchangeable and generic modules, that can be used in different projects without knowing any details or imposing any restrictions in the data producer. Moreover, this is also the key design concept which enables to replace a part of the system by a set of simulation GAMs, without changing the other modules, a very important feature when testing and designing a new control system, before introducing the hardware in the loop, and later in the commissioning of new hardware if good models of the plant are available. An example is shown in Fig.1. A large set of generic GAMs is available to be used in any MARTe application (e.g. PID, waveform generation, live data view, data collection, data statistics).

A special GAM, named IOGAM, enables the connection of any hardware to the DDB, as long as a MARTe high level driver is developed to provide the connection between the IOGAM and the hardware interface (usually through an operating system low level driver). GAMs are sequentially executed, at a given frequency, by a real-time thread. Several real-time threads can be connected, both synchronously and asynchronously, and executed in parallel.

A real-time thread cycle is triggered by an entity named External Time Triggering Service (ETTS). The ETTS is connected to a time provider and checks that the current time is a multiple of the configuredMARTe cycle time. When this condition holds true a new cycle is signalled. Two types of ETTS are available: one based in hardware interrupts and another based in the polling of a resource (e.g. shared memory entry). The synchronisation scheme will issue an alarm if a timeout occurs, either due to a slow execution of the control cycle (e.g. a GAM is consuming too much time and the cycle finishes after its period has already elapsed) or if the jitter in the timing source is very high. MARTe provides some ready to be used synchronisation schemes based in CPU timers and in network inputs.

Taking advantage of the new multi-core processors, the framework enables all of its tasks to be allocated to a specific subset of cores. In Linux, using the special isolcpus kernel parameter, it enables to have an isolated real-time execution environment for the real-time threads with jitters in the order of the microseconds [4].

2. INTERFACINGWITH MARTE

All MARTe objects are setup using the configuration mechanism described above. The framework makes no assumptions on how these files are produced and does not impose any communication protocol on how these should be transmitted. A standard C++ interface, with all the infrastructure from the MARTe side implemented, is available for the development of new communication mechanisms. Currently, for systems outside JET, the only ready to be used configuration mechanism is based in the HTTP protocol, although the preferred way of interfacing with MARTe components is to use the message mechanism introduced before. In order to integrate the framework in different environments, the required protocols are translated into messages and broadcast to the destination objects.

The framework also provides its own HTTP server, capable of browsing and introspecting any of the installed objects. The server was designed to minimise any impact with the real-time activities, by carefully executing all of its activities in low-priority tasks and, in the case of multicore environments, in cores not allocated for the real-time threading. This scheme enables GAMs to publish run-time execution information about the internal state of algorithms and data.

3. MARTE SYSTEMS

As shown in table 1, a large number of control systems is already using MARTe to solve different control problems [5]. These systems are key to the operation of large experiments, and some have an active role providing a first line of defence for the protection of the machine itself. The framework is installed in different experiments, with different configuration and data retrieving protocols.

The most frequent operating systems are Linux, running in Intel® and AMD® processors with isolated cores, and VxWorks® running in PowerPC® . The frequency of execution spans from 100Hz to 20kHz. Some modules, like the waveform generation, data collection and data statistics are shared by all projects.

The COMPASS control system [7], depicted in Fig.2, was the first to exploit the capability of running multiple real-time threads at different frequencies. The fastest thread executes at 20kHz and starts by reading the data from the analogue input channels, followed by an offset removal (due to the use of integrators). Data is then filtered and sent to the second, slower, thread running at 500kHz.

In parallel, the fastest thread control the horizontal and vertical magnetic fields, while the slower thread is responsible for the control of the plasma current and position. Since the hardware is the same used by the JET vertical stabilisation, the only GAMs that had to be developed concerned the plasma control.

CONCLUSIONS

MARTe is C++ real-time framework designed for the development and deployment of control-systems. It is already being used in several experiments to solve different problems that require different operational frequencies and hardware interfaces. Currently it is only being used in magnetic confinement nuclear fusion experiments, but there is not reason why it cannot be deployed in any context where a real-time control system is required.

REFERENCES

[1]. A. Neto et al., “MARTe: A Multiplatform Real-Time Framework”, IEEE Transactions on Nuclear Science, Vol. **57**, pp. 479-486, 2010

[2]. G. De Tommasi et al., “A flexible software for real-time control in nuclear fusion experiments”, Control Engineering Practice, Vol. **14**, pp. 1387-1393, 2006

[3]. A. Neto et al., “Linux real-time framework for fusion devices”, Fusion Engineering and Design, Vol. **84**, pp. 1408-1411, 2009

[4]. D.F. Valcarcel et al., “The COMPASS Tokamak Plasma Control Software Performance”, IEEE Transactions on Nuclear Science, Vol. **58**, pp. pp. 1490-1496, 2011

[5]. A. Neto et al., “A Survey of Recent MARTe Based Systems”, IEEE Transactions on Nuclear Science, Vol. **58**, pp. 1482-1489, 2011

[6]. D. Alves et al., “The new Error Field Correction Coil controller system in the Joint European Torus tokamak”, accepted for publication in Fusion Engineering and Design

[7]. D.F. Valcarcel et al., “Real-time software for the COMPASS tokamak plasma control”, Fusion Engineering and Design, Vol. **85**, pp. 470-473, 2010

[8]. P. J. Carvalho et al.,“ISTTOK plasma control with the tomography diagnostic”, Fusion Engineering and Design, Vol. **85**, pp. 266-271, 2010

[9]. L. Boncagni et al., “First Steps in the FTUMigration Towards a Modular and Distributed Real-Time Control Architecture Based on MARTe”, IEEE Transactions on Nuclear Science, Vol. **58**, pp. 1778-1783, 2011

[10]. A. Stephen et al., “Centralised Coordinated Control To Protect The JET ITER-likeWall”, this conference

[11]. D. Alves et al., “The Software and Hardware Architectural Design of the Vessel Thermal Map Real-Time System in JET”, this conference

Name	Cycle time	O.S.
JET Vertical Stabilisation [1]	50 µs	Linux-RTAI
JET Error Field Correction Coils [6]	200 µs	VxWorks®
COMPASS [7] Shape Controller	500 µs	Linux
COMPASS [7] Vertical Stabilisation	50 µs	Linux
ISTTOK [8] Tomography	100 µs	Linux
FTU [9] Plasma Control	500 µs	Linux-RTAI
JET Real-time Protection Sequencer [10]	2 ms	VxWorks®
JET Vessel Thermal Map [11]	10 ms	Linux
JET Plasma Wall Load System	10 ms	Linux

Table 1: Systems using MARTe

```
+HttpServer = {
  Class = HttpService
  Port  = 8084
}
+MARTe = {
  Class = MARTeContainer
  +RTThread1 = {
    Class = RealTimeThread
    +SurfaceTemperature_WOPL_14 = {
      Class = SurfaceTemperature1DCalculation
      SpecificHeat = 1.925000e+03
      ThermalConductivity = 1.900000e+02
      DeltaT = 0.01
      Nslices = 40
      InputSignalNames = {
        0 = {Q_WOPL_14}
      }
      OutputSignalNames = {
        0 = {SurfaceTemperature_WOPL_14}
      }
    }
    +DivertorThermalCalculations = {
      ...
      WallsPowerPartitionClass = {
        Type = WallsPowerPartitionClass
        PartitionTable = {
          0 = {
            0 = {0.3 0.3 0.3 0.3}
            1 = {0.7 0.7 0.7 0.7}
          }
          ..
        }
        ...
    }
  }
...
```

Listing 1: A small fraction of a configuration file from the real-system responsible for the JET plasma wall load protection system (WALLS).

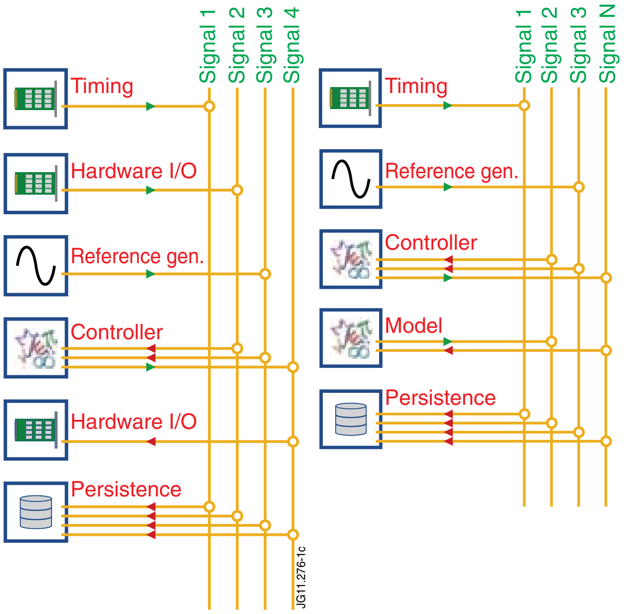

Figure 1: Example of a set of GAMs connected to the DDB. A timing and an hardware GAM provide the I/O interface to the outside world, whereas a generic waveform GAM inputs the reference for a PID controller. Finally, the output is sent to a DAC and the data is stored for analysis by a collection GAM. The picture in the right shows how the same system can be developed using a model of the plant. It should be noticed that the reference generation and the controller GAM are not aware of the changes in the data providers and data consumers.

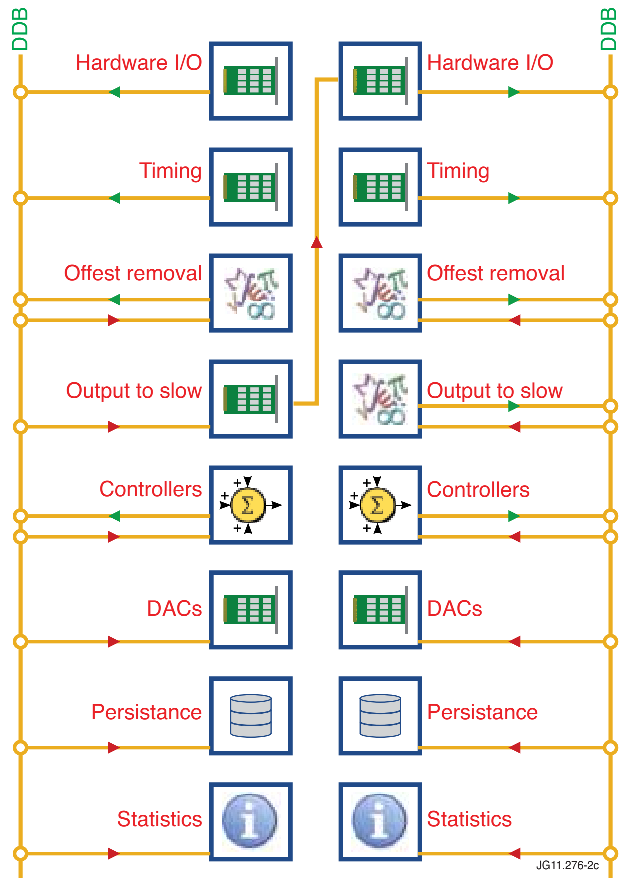

Figure 2: The COMPASS plasma control system was the first system to exploit the multi real-time threading capabilities of MARTe. A fast thread (20kHz) provides the ADC data to a slower thread (2kHz).