Fusion Engineering and Design 85 (2010) 470–473

Contents lists available at ScienceDirect

Fusion Engineering and Design

journal homepage: www.elsevier.com/locate/fusengdes

Real-time software for the COMPASS tokamak plasma control

D.F. Valcárcel[a,*], A.S. Duarte[a], A. Neto[a], I.S. Carvalho[a], B.B. Carvalho[a], H. Fernandes[a], J. Sousa[a], F. Sartori[b], F. Janky[c], P. Cahyna[c], M. Hron[c], R. Pánek[c]

[a] Associação EURATOM/IST, Instituto de Plasmas e Fusão Nuclear - Laboratório Associado, Instituto Superior Técnico, P-1049-001 Lisboa, Portugal
[b] Euratom-UKAEA, Culham Science Centre, Abingdon, OX14 3DB Oxon, United Kingdom
[c] Institute of Plasma Physics AS CR, v.v.i., Association EURATOM/IPP.CR, Za Slovankou 3, 182 00 Prague, Czech Republic

ARTICLE INFO

Article history:
Available online 24 April 2010

Real-time
ATCA
Data acquisition
Plasma control software

ABSTRACT

The COMPASS tokamak has started its operation recently in Prague and to meet the necessary operation parameters its real-time system, for data processing and control, must be designed for both flexibility and performance, allowing the easy integration of code from several developers and to guarantee the desired time cycle. For this purpose an Advanced Telecommunications Computing Architecture based real-time system has been deployed with a solution built on a multi-core x86 processor. It makes use of two software components: the BaseLib2 and the MARTe (Multithreaded Application Real-Time executor) real-time frameworks. The BaseLib2 framework is a generic real-time library with optimized objects for the implementation of real-time algorithms. This allowed to build a library of modules that process the acquired data and execute control algorithms. MARTe executes these modules in kernel space Real-Time Application Interface allowing to attain the required cycle time and a jitter of less than 1.5 μs. MARTe configuration and data storage are accomplished through a Java hardware client that connects to the FireSignal control and data acquisition software.

This article details the implementation of the real-time system for the COMPASS tokamak, in particular the organization of the control code, the design and implementation of the communications with the actuators and how MARTe integrates with the FireSignal software.

1. Introduction

The COMPASS tokamak ($R = 0.56$ m, $a = 0.18$–0.23 m), which has an ITER-like geometry, has started its operation recently in Prague, Czech Republic. The scientific programme will concern on edge plasma physics, namely pedestal studies and resonant magnetic perturbations, and on wave–plasma interaction studies [1]. This offers an important research potential as a small, flexible and low-cost facility with an ITER-relevant programme.

To meet the operation parameters its real-time system for data processing and control must be designed for both flexibility and performance, allowing the easy integration of code from several developers and to guarantee the desired time cycle. In particular, a control cycle of 50 μs [2] is needed for the vertical and horizontal positions control and a 500 μs loop is required for the plasma current, equilibrium, and shaping control.

Traditionally the control and data acquisition systems are based on VME or PCI. For example the DIII-D PCS [3] is based on VME PC components with PCI ethernet links. This system uses a VME rack with several PCs, each with a customization of a standard Linux distribution. The real-time performance is accomplished by turning off interrupts during the shot and turning them back on at the end. This means no timers and no communication with the outside world during real-time operation. Although this approach has been successfully used in several tokamaks controlling on 50 μs cycles, it is not ITER-relevant given the long duration for the pulses and the ability to reconfigure parameters during the discharge is required.

The MARTe framework [4] is currently being used at JET for the plasma vertical stabilization also in a 50 μs control cycle [5]. It is a real-time multi-platform C++ framework that allows the development of real-time systems with a modular design. This modularity, embodied on the Generic Application Module (GAM) concept, enables code reuse and integration into the same system from several developers. MARTe can be executed in several platforms such as the Linux OS, VxWorks and the Linux RTAI OS. The first provides an excellent testing environment for the codes and also has very good performance when the system on which it runs has isolated CPU cores, one for the Linux processes and MARTe can use all the others for itself. The latter operating systems allow

* Corresponding author. Tel.: +351 218417823
E-mail address: danielv@ipfn.ist.utl.pt (D.F. Valcárcel).

0920-3796/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.fusengdes.2010.03.049

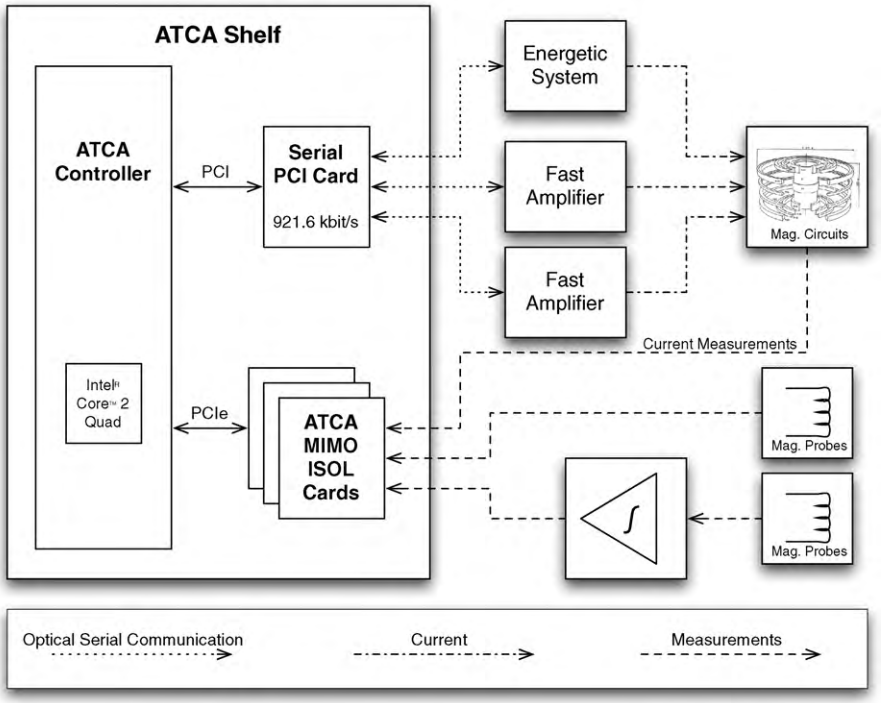

Fig. 1. COMPASS control system assembly: the crate communicates with the power supplies through a fast optical serial link and these are connected to the magnetic field coils. The currents, magnetic fluctuations and the integrated magnetic signals can be measured and processed in real-time.

executing the validated and optimized code in a hard real-time environment.

The COMPASS real-time system is based on the JET approach and the hardware solution for its Control and Data Acquisition and Communications (CODAC) [6] is based on the PICMG 3.0 Advanced Telecommunications Computing Architecture (ATCA) standard and on multi-core x86 processor technology.

Although COMPASS does not have long plasma discharges compared to ITER, having an ATCA system allows to make use of its inherent reliability and robustness for COMPASS operation and to test, even in small scale, ITER-relevant concepts.

This article describes the real-time system presenting the architecture and the design decisions for the magnetic plasma control application.

2. The real-time system

2.1. Introduction

Fig. 1 depicts the connections for the components of the COMPASS control system. It is intended for plasma control and stabilization: toroidal field generation, plasma current, equilibrium and shape control and vertical and horizontal stabilization. The ATCA shelf contains two types of cards. The first is a controller card based on a standard motherboard with an Intel® Core® 2 Quad Q6600 processor @ 2.4 GHz and 4 MB of cache memory. The second type is a data acquisition card, named ATCA MIMO ISOL [7] with 32 galvanically isolated input channels. The maximum number of cards per shelf is 12. This is the platform intended for real-time where the data acquisition is performed at 20 kSamples/s with 18-bit resolution. The system measures 4 flux loop probes to determine the plasma position, a Rogowski probe for the plasma current and 6 current sensors to measure the current on all circuits that generate magnetic fields. This system is connected through a PCI Lindy® Quad Fast Serial card via optical fiber to the energetic system and the fast amplifiers which drive the magnetic actuator coils. The chip that enables the communications is an Oxford Semiconductor® OXmPCI964, supporting 8- or 9-bit protocols at a maximum baud rate of 921.6 kbit/s, and the optical converters were developed onsite.

Regarding MARTe, the GAMs are instantiated on C++ objects belonging to a real-time thread (RTTh), their code gets executed in sequence and they exchange connections to one another through input and output signals, thus sharing calculation results. This implies that a major portion of the effort to use this platform is spent on designing the architecture of the system, apart from the programming of the GAM functionality.

MARTe defines a state machine with some standard states, such as online and offline. These can be used to trigger modifications on the GAMs behaviour. For instance offline might be associated with an idle period when only monitoring is needed, whereas online can be associated with the generation of a plasma pulse.

The communication with the acquisition boards is accomplished through a specialized GAM (ATCAAdc). One of its main responsibilities, apart from transferring acquired data, is the generation of the system timing signal, named *usecTime*. This signal contains important information: it is the internal counter of the master GPIO acquisition board, updated each 50 μs, and marks the occurrence of a hardware trigger through a reset of the counter.

Finally, GAMs are allowed to build an HTML interface available to the outside world. This provides a flexible, licence-free, means of configuration and introspection.

2.2. Code architecture

Fig. 2 shows the organization of the code in RTTh and GAMs and the most important signals that connect them. This design reflects the needs for the timings of the control cycles, one RTTh with 500 μs repetition rate for the control of the plasma parameters and another with 50 μs for the fast stabilization. Use is made of some of the GAMs already developed for JET, such as the DataCollectionGAM and the WebStatisticGAM. As an ongoing project not all the mod-

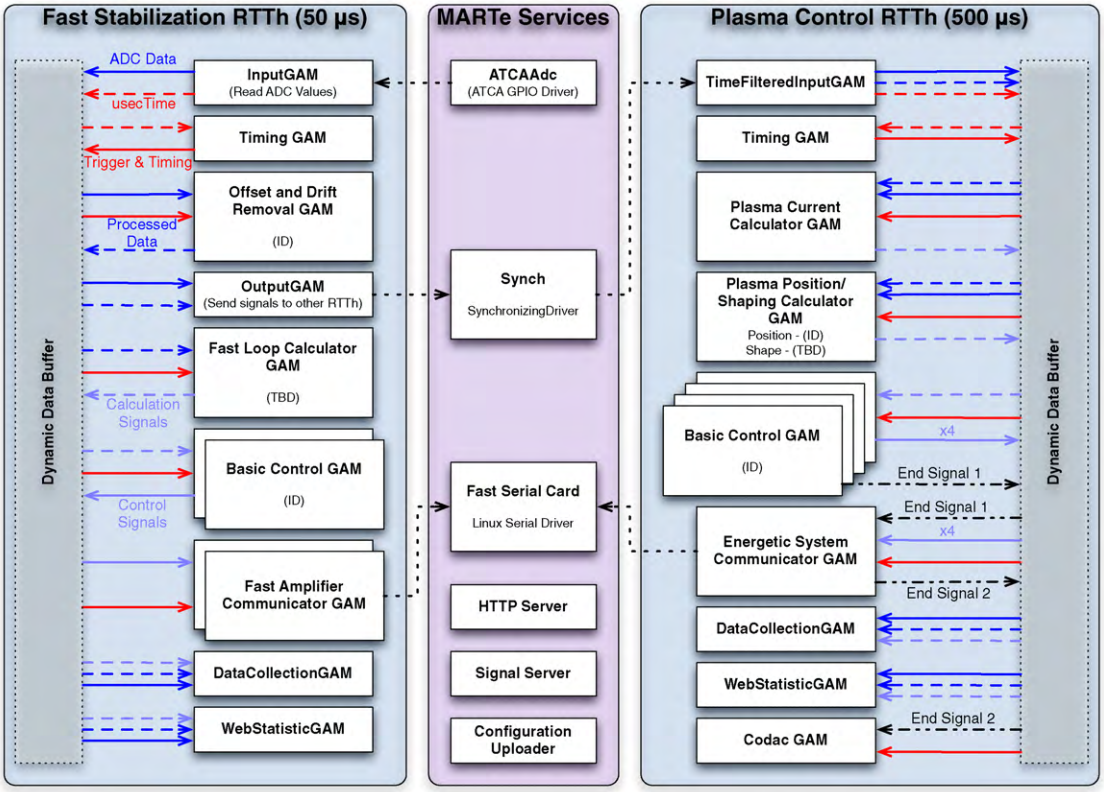

Fig. 2. MARTe real-time threads with the associated GAMs for the proposed control system. The GAM execution inside each thread takes place from top to bottom. ID—in development, TDB—to be developed.

ules are implemented yet or fully developed, the current schematic of the proposed solution for the control of COMPASS is presented.

The suite of modules for this real-time system can be divided into 4 main categories: processing, control, communication and system management. Each of these has different requirements about how they integrate on the MARTe state machine. For instance, the processing modules might be continuously processing data whereas the communication modules have to respond accordingly to the plant state, e.g. monitoring on the offline state or dealing with a plasma discharge on the online state.

The processing modules are the Offset And Drift Removal GAM, the Plasma Position/Shape Calculator GAM, the Plasma Current Calculator GAM and the Fast Loop Calculator GAM. The first is used to remove the offsets from the signals acquired by the ADCs and their drift caused by analog integration, which is assumed to be linear. A set of samples is collected before the shot and a first degree polynomial is fitted to get the drift parameters. Subsequently the drift is removed from all samples. The plasma position calculation developed at Prague IPP measures the magnetic signals from 4 magnetic flux probes, two at the top/bottom and 2 at the equatorial plane high field side/low field side. The signals are analog integrated and the ratios between the two vertical and the two horizontal probes are calculated. The ratio values allow to get the circular plasma position from a simulation generated table. For a shaped plasma the real-time boundary determination is needed, rtEFIT [8] or XLOC [9] will be used for this purpose.

One of the control modules in development is the Basic Controller GAM. It implements a feedback loop (proportional–integral–derivative controller) and a feedforward loop working together as an automatic controller, but also lets the user choose which time intervals during a plasma discharge are to be driven manually, setting the actuator to a pre-defined value.

The communication GAMs are used to communicate with the power supply systems. The Energetic System Communicator GAM uses the Current Control Protocol (CCP) to communicate with the toroidal, magnetising, equilibrium and shaping power supplies. The FA Power Supply Communicator GAM uses the Fast Amplifier Communication Protocol (FACP) to communicate with the fast amplifiers. Both protocols were developed at IPFN, Lisbon. This implementation of the communications, for both GAMs, is accomplished through calls to the Linux kernel functions *ioread8* and *iowrite8* which allow to write directly on the registers of the serial card. This solution has proven to meet the functional requirements, as it executes in a short period of time, and does not require the development of a special driver to handle the communications.

From the system management family there is the Timing GAM which receives the *usecTime* signal from MARTe from where it can generate other timing and trigger signals for internal use of the GAMs, and also the Codac GAM which handles the communication with the CODAC, as described in Section 2.3.

The *usecTime* trigger described is internally converted by the Timing GAM to provide a uniform trigger to all the GAMs. When the transition to zero of the *usecTrigger* is detected, which is related to a real hardware trigger signal received, the Timing GAM generates an output signal which is different from zero for a few RTTh cycles and zero on all others. The GAMs reading this signal detect this change from zero and assert an internal flag that informs about their reaction to the trigger. This information is used when the behaviour of the GAMs is dependent on the state of the trigger.

Preliminary time measurements have been made and are shown in Table 1. It can be seen that for the 50 µs RTTh the current execution time is about 17 µs and the GAMs in development have still 33 µs available to execute. Regarding the 500 µs RTTh, the total time is around 105 µs (395 µs remaining), the main responsible for this value being the Energetic System Communication GAM.

Table 1
Measured execution times for the system. The worst results are from the serial communicators due to the hardware accesses. Almost all the other GAMs execute on sub-μs intervals.

		Time (μs)
50 μ RTTh	Timing GAM	$1.060 \pm 2.28 \times 10^{-5}$
	Fast Amplifier Communicator GAM	$6.221 \pm 1.28 \times 10^{-4}$
	DataCollectionGAM	$0.913 \pm 4.74 \times 10^{-5}$
	WebStatisticGAM	$2.272 \pm 3.70 \times 10^{-5}$
	OutputGAM	$0.561 \pm 4.44 \times 10^{-4}$
	Total	$17.248 \pm 6.79 \times 10^{-4}$
500 μ RTTh	Timing GAM	$1.145 \pm 1.59 \times 10^{-5}$
	Plasma Current Calculator GAM	$0.740 \pm 8.85 \times 10^{-6}$
	Energetic System Communicator GAM	$100.800 \pm 4.51 \times 10^{-4}$
	Codac GAM	$0.440 \pm 2.35 \times 10^{-5}$
	DataCollectionGAM	$0.687 \pm 6.99 \times 10^{-4}$
	WebStatisticGAM	$0.860 \pm 1.26 \times 10^{-5}$
	Total	$104.672 \pm 1.21 \times 10^{-3}$

2.3. Integration with the CODAC system

The FireSignal system [10] is the CODAC for COMPASS. This system requires a hardware client (node) to provide access to the subsystems. The node that accesses MARTe and its GAMs was programmed in the Java language, a choice made by its ease of use and fast development. It accesses the HTTP server to obtain the information it requires from the GAMs, namely all the information provided by the Codac GAM.

The node can interact with MARTe via 3 channels, the Signal Server (SS), the Configuration File Uploader (CFU) and the State Machine (SM), all 3 available through the HTTP server. The SS after each discharge makes the collected data available and the FireSignal node fetches it and sends it to the database. The SM interaction allows the node to change its state, i.e. to online or offline (among others). The CFU allows for the node to upload a new configuration file for MARTe without the need to shutdown and restart MARTe. This file contains information about what RTTh should be created, which GAMs to run and the parameter values to configure the entire system. In order to configure the GAMs from the FireSignal front-end this configuration file must be parsed, the configuration values written in the proper place and then it must be passed to MARTe through the CFU.

Because some GAMs respond according to the MARTe state machine state, before the CODAC can change the state from online to offline it must know if all the GAMs are ready to move to the offline state. This led to the development of the *engine* concept.

In this context, an engine is a set of GAMs that are chained with signals that inform each other in sequence that they are ready to stop. The example in this particular application is represented in Fig. 2. These signals are represented in dashed black: the first signal connects the Basic Control GAM that controls the toroidal field to the Energetic System Communicator GAM (*End Signal* 1) and the second links the latter to the Codac GAM (*End Signal* 2). When the toroidal field current is set to zero at the end of the discharge the Basic Control GAM informs the Energetic System Communicator GAM that it has finished and the energetic system may shutdown. After this is done the Energetic System Communicator GAM informs the Codac GAM that it has also finished. The final step is the Codac GAM changing a flag in its HTML interface to inform the FireSignal node that the MARTe state can be changed to offline. It would be easier to use just one GAM for this purpose but this way keeps the modularity of the system with strict boundaries between different functionalities. These chained GAMs operate together in harmony and thus the name *engine* for this configuration.

3. Conclusions

A flexible system was developed for the COMPASS tokamak on top of a high-performance generic processor and of a software framework tailored for real-time. It allows to integrate code from different developers into the same system, due to the modular nature of the underlying framework. The preliminary time measurements performed on the system indicate that its performance capabilities will allow to execute the proposed algorithms in the specified time constraints of 500 μs for the slow controllers (plasma current, shaping and equilibrium) and 50 μs for the fast vertical instability and equilibrium controllers.

With the fulfilment of the proposed requirements for COMPASS, its scientific programme can be exploited with enough flexibility due to the easiness to make quick changes to the system. In particular a Multiple-Input Multiple-Output (MIMO) scheme taking into account all the controlled variables might replace the approach of 4 separate controllers. The flexibility of the system allows to exchange the controller modules without affecting the remainder of the system.

Acknowledgements

This work has been carried out within the framework of the Contracts of Association between the European Atomic Energy Community and "Instituto Superior Técnico" (IST) and IPP.CR. IST also received financial support from "Fundação para a Cincia e Tecnologia" in the frame of the Contract of Associated Laboratory. The views and opinions expressed herein do not necessarily reflect those of the European Commission.

References

[1] R. Pánek, O. Bilykova, V. Fuchs, M. Hron, et al., Reinstallation of the COMPASS-D tokamak in IPP ASCR, Czechoslovak Journal of Physics 56 (Suppl. B) (2006) B125–B137.
[2] P. Vyas, Plasma vertical position control on COMPASS-D, PhD Dissertation, Dept. Eng. Sci., Oxford Univ., Oxford, U.K., 1996.
[3] B.G. Penaflor, J.R. Ferron, R.D. Johnson, D.A. Piglowski, Current status of the upgraded DIII-D real-time digital plasma control system, Fusion Eng. Des. (71) (2004) 47–52.
[4] A. Neto, et al., MARTe: a multi-platform real-time framework, IEEE Trans. Nucl. Sci., IEEE-NPSS 16th Real Time Conference Special Issue, in press.
[5] F. Sartori, et al., The PCU JET plasma vertical stabilisation control system, Fusion Eng. Des. 85 (3–4) (2010) 438–442.
[6] D.F. Valcárcel, et al., An ATCA embedded data acquisition and control system for the compass tokamak, Fusion Eng. Des. 84 (7–11) (2009) 1901–1904.
[7] A.J.N. Batista, J. Sousa, C.A.F. Varandas, ATCA digital controller hardware for vertical stabilization of plasmas in tokamaks, Rev. Sci. Instrum. 77 (10F527) (2006) 1–3.
[8] J.R. Ferron, et al., Real time equilibrium reconstruction for tokamak discharge control, Nucl. Fusion 38 (7) (1998) 1055–1066.
[9] D.P. O'Brien, et al., Local expansion method for fast plasma boundary identification in JET, Nucl. Fusion 33 (3) (1993) 467–474.
[10] A. Neto, H. Fernandes, A. Duarte, B.B. Carvalho, et al., FireSignal—data acquisition and control system software, Fusion Eng. Des. 82 (5–14) (2007) 1359–1364.