Fusion Engineering and Design 89 (2014) 195–203

Contents lists available at ScienceDirect

Fusion Engineering and Design

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

ISTTOK real-time architecture

Ivo S. Carvalho*, Paulo Duarte, Horácio Fernandes, Daniel F. Valcárcel, Pedro J. Carvalho, Carlos Silva, André S. Duarte, André Neto, Jorge Sousa, António J.N. Batista, Tiago Hekkert, Bernardo B. Carvalho

Associação EURATOM/IST, Instituto de Plasmas e Fusão Nuclear, Instituto Superior Técnico, Universidade Técnica de Lisboa, P-1049-001 Lisboa, Portugal

HIGHLIGHTS

- All real-time diagnostics and actuators were integrated in the same control platform.
- A 100 μs control cycle was achieved under the MARTe framework.
- Time-windows based control with several event-driven control strategies implemented.
- AC discharges with exception handling on iron core flux saturation.
- An HTML discharge configuration was developed for configuring the MARTe system.

ARTICLE INFO

Article history:
Received 2 May 2013
Received in revised form
15 November 2013
Accepted 16 December 2013
Available online 24 January 2014

Keywords:
Nuclear fusion
Control and data acquisition
ATCA systems
MARTe framework
Real-time

ABSTRACT

The ISTTOK tokamak was upgraded with a plasma control system based on the Advanced Telecommunications Computing Architecture (ATCA) standard. This control system was designed to improve the discharge stability and to extend the operational space to the alternate plasma current (AC) discharges as part of the ISTTOK scientific program. In order to accomplish these objectives all ISTTOK diagnostics and actuators relevant for real-time operation were integrated in the control system.

The control system was programmed in C++ over the Multi-threaded Application Real-Time executor (MARTe) which provides, among other features, a real-time scheduler, an interrupt handler, an intercommunications interface between code blocks and a clearly bounded interface with the external devices. As a complement to the MARTe framework, the BaseLib2 library provides the foundations for the data, code introspection and also a Hypertext Transfer Protocol (HTTP) server service.

Taking advantage of the modular nature of MARTe, the algorithms of each diagnostic data processing, discharge timing, context switch, control and actuators output reference generation, run on well-defined blocks of code named Generic Application Module (GAM). This approach allows reusability of the code, simplified simulation, replacement or editing without changing the remaining GAMs.

The ISTTOK control system GAMs run sequentially each 100 μs cycle on an Intel® Q8200 4-core processor running at 2.33 GHz located in the ATCA crate. Two boards (inside the ATCA crate) with 32 analog-to-digital converters (ADCs) were used for acquiring the diagnostics data. Each ADC operates at 2 Msample/s but (for real-time operation) the acquired data is decimated in real-time on the board's Field-programmable gate array (FPGA) to a frequency defined by the control cycle time.

This paper presents the ISTTOK real-time architecture and the human–machine Interface (HMI) for simplified AC discharge programming.

1. Introduction

The ISTTOK tokamak (Ip = 7 kA, BT = 0.6 T, R = 0.46 m, a = 0.085 m) has a large aspect ratio and is one of the few tokamaks with a regular Alternate plasma Current (AC) discharges scientific program [1,2]. After a recent update [3], the real-time architecture is now based on the Advanced Telecommunications Computing Architecture (ATCA) standard while the old architecture was based on the Peripheral Component Interconnect (PCI) standard. The main objective of this upgrade was to improve the discharge stability and to increase the number of AC discharge cycles while maintaining stable plasma parameters throughout the discharge.

To increase the discharge duration several systems were updated [3] including the power supplies (PS). The horizontal field power supply [4] had minor changes for compatibility with the new ATCA control system, the vertical field was updated in the power

* Corresponding author. Tel.: +351 218417823; fax: +351 218417819.
E-mail address: ivoc@ipfn.ist.utl.pt (I.S. Carvalho).

0920-3796/$ – see front matter © 2014 Instituto de Plasmas e Fusão Nuclear. Published by Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.fusengdes.2013.12.025

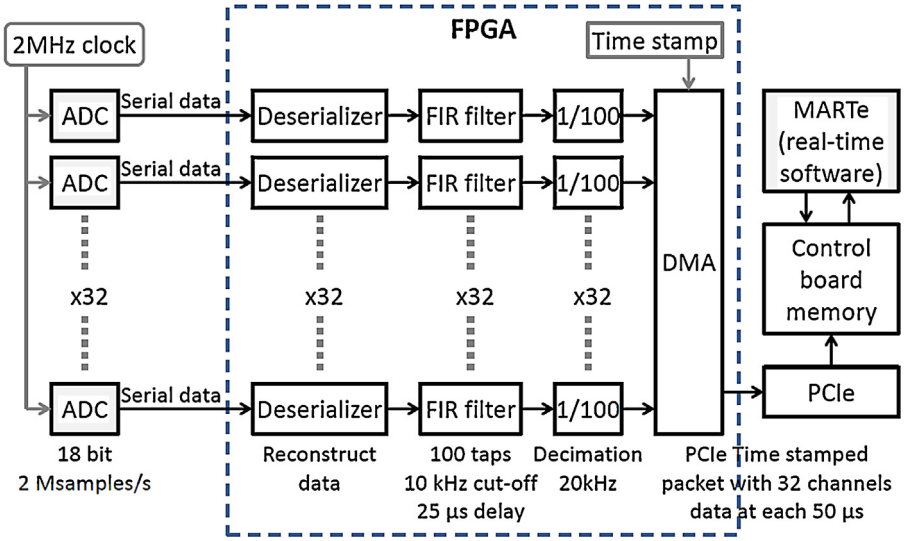

Fig. 1. Data path from acquisition to MARTe DDB. The FPGA firmware transforms the 2 Msamples/s acaquisition into 20 kHz data using an FIR filter before decimation.

rating and the magnetizing field power supply was completely redesigned.

The control software was completely redesigned to accommodate all the real-time diagnostics and all the real-time actuators in a Multiple Input Multiple Output (MIMO) fashion. In addition, the control concept was updated, featuring a new event-driven approach. This mode of operation is described in Section 3.3.

To simplify the role of the machine operator, two C++ objects using BaseLib2 (collection of libraries that support MARTe) were created. These objects implement a HTTP server and the user programs the discharge using a standard web browser such as Google chrome[tm], Microsoft Internet explorer[tm] or Mozilla firefox[tm].

2. Hardware

The hardware used in the ISTTOK real-time system is based on the Advanced Telecommunications Computing Architecture (ATCA) standard, similar to the hardware used in JET vertical stabilization [5]. Recently an improved version of this hardware was used successfully on the "ITER prototype fast plant system controller" project [6].

The ISTTOK control hardware [7] includes a carrier board, two acquisition boards, Rear Transition Module (RTM), a quad Universal asynchronous receiver/transmitter (UART) PCI port connected to the carrier motherboard and the serial to optical converters for the UART ports. The communication between carrier and acquisition boards is based on the PCI-express (PCIe) standard.

The carrier board contains a commercial of the shelf (COTS) PC motherboard with a quad core Intel[tm] Q8200 processor running at 2.33 GHz. The quad UART port is connected to this motherboard.

The ATCA acquisition board is composed by 32 analog-to-digital converter (ADC) modules connected to a Virtex-4 Field-programmable gate array (FPGA) that manages the data path from the ADC to the PCIe bus as depicted in Fig. 1. Since tokamak operation has a noisy environment and the selected ADCs were able to acquire data at 2 Msample/s, it was decided to implement an additional digital filter in the FPGA to filter each ADC sample with a 100 tab finite impulse response (FIR) filter. This filter has a cut-off frequency of 10 kHz and introduces an additional 25 μs delay, not affecting significantly the ISTTOK real-time control. After filtering, the data is decimated to 1/100 and then each data packet travels through the PCIe bus with the respective time stamp.

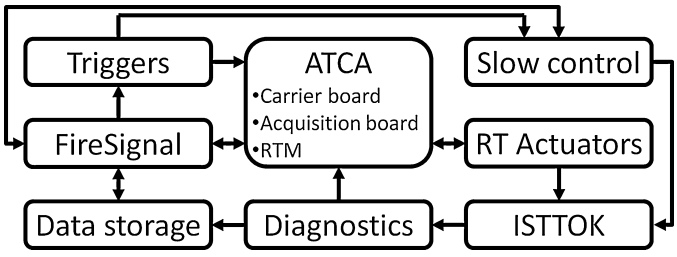

Fig. 2. Simplified ISTTOK connections diagram.

Each ADC module has 1000 V galvanic isolation and can work as a single ended ADC or as differential input. This ADC samples data at 2 Msample/s with 18 bit resolution, the effective number of bits (ENOB) of this module is 13.7 bits at 2 Msamples/s and 15.7 bits after filtering and decimation to 20 ksamples/s.

The RTM module contains an optical or electrical small form–factor pluggable transceiver (SFP) interface (full duplex) and 8 galvanic isolated (up to 700 V) 16-bit digital-to-analog converter (DAC), one of which is used to control the gas puff piezoelectric valve. The RTM also receives the trigger from ISTTOK trigger system and a 2 MHz clock for synchronization.

The quad UART board, together with the serial to optical converters, is used for optical communication with the ISTTOK power supplies at 921.6 kbit/s.

The connection of the hardware contained in the ATCA crate with the rest of ISTTOK systems is presented in Fig. 2. FireSignal [8,9] is used for programming the triggers and the slow tokamak control.

3. Software

Several machines have implemented real-time control systems using the Multi-threaded Application Real-Time executor (MARTe) framework [10,11]. Each real-time system has different cycle time characteristics and is implemented on different operating systems and architectures. These implementations are summarized in Table 1. This table evidences the flexibility of the MARTe real-time framework with control cycles as low as 50 μs up to control cycles of 10 ms. The ISTTOK control system was also implemented on top of the MARTe framework and has a 100 μs control cycle.

Table 1
List of working systems using MARTe framework.

Working systems				
Device	Cycle time	System/sub-system	Operating system	Architecture
JET	50 μs–10 ms	Several sub-systems	Linux/RTAI/VxWorks	ATCA/VME/PC
COMPASS	50 μs/500 μs	Fast and slow PS	Linux	ATCA
FTU	250 μs	Lower hybrid power	RTAI	VME
RFX	200–250 μs	Magnetic control system	Linux(PreemptRT)	PXI
ISTTOK	100 μs	Tokamak RT control	Linux	ATCA

The MARTe framework is based on the C++ programming language and is aimed at providing a standard development base for programming real-time systems across several architectures. The main driving concept for developing the MARTe framework was to standardize the development of real-time control systems by creating a clear separation between the applications, the hardware and the operating system. The underlining BaseLib2 is responsible for providing the abstraction layer that enables MARTe to be deployed in several architectures and operating systems without changing the application code. MARTe provides the synchronization base for the control system, acting as micro-scheduler for the sequential execution of real-time code, scheduling the code to be executed with a fixed control cycle. The BaseLib2 provides the base for the services provided by the MARTe framework such as the logger service and HTTP server for inspecting GAM status information. If correctly configured, these services run on a different CPU core, thus not compromising the system real-time performance. This subject is further developed in Section 3.1.

MARTe is a modular system that synchronizes the execution of well-defined blocks of code called Generic Application Module (GAM) [12]. These GAMs are executed in sequence in one real-time thread and exchange data using a dynamic data buffer (DDB). Each real-time thread can be assigned to a specific CPU core and several real-time threads can be produced [14], although in ISTTOK, only one thread is used. A simplified scheme for MARTe is depicted in Fig. 3.

3.1. MARTe implementation

The MARTe framework was installed on a Linux Gentoo [13] distribution. Since ISTTOK control system included a quad core CPU, it was possible to select the CPU core for running each individual process. The Linux operating system services were constrained to the first CPU core by using the kernel boot option "isolcpus = 1,2,3",

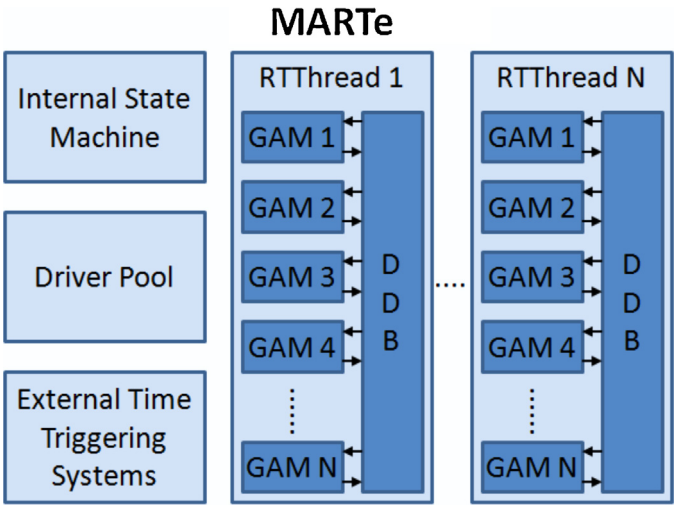

Fig. 3. Simplified MARTe description.

Table 2
CPU load distribution per CPU core in a quad core CPU.

CPU	Process
Core 0	Linux services
Core 1	Drivers for acquisition boards
Core 2	MARTe services
Core 3	ISTTOK real-time thread

which prevented Linux services from running in the remaining cores. After Linux operating system boot, each ISTTOK specific process is distributed to a specific CPU core using the "taskset" command with the corresponding CPU core. The result of the process distribution is in Table 2. To further optimize the real-time code execution, all the ISTTOK real-time code was compiled with the option "-o3" in the C++ compiler (gcc).

For configuration purposes, two objects that interact with MARTe were produced (see Fig. 4). These objects implement a separated HTTP server in C++ using the BaseLib2 libraries. These objects are external software tools for editing the MARTe configuration file which contains the initialization values of the ISTTOK real-time control system. The "Discharge configurator" object handles the most used configuration parameters such as plasma current/position waveforms and the "Advanced configuration" object is used to configure the less changeable parameters such as the serial port address of the UART that communicates with a specific power supply. These objects are a part of the human–machine interface (HMI) and are further described in Section 3.2 of this paper.

By default, the MARTe framework provides the following services: (i) HTTP server, (ii) logger server, (iii) configuration provider and (iv) state machine. The MARTe service defines a HTTP server for inspecting GAM status information, the logger server is used to broadcast via UDP the system status/debugging information. The configuration provider service defines the system configuration and the state machine service is used for implementing a system-wide status definition (for example: online and offline status for non-continuous systems).

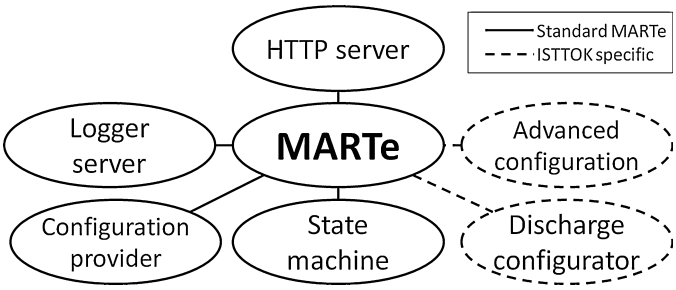

Fig. 4. MARTe connections. The MARTe framework includes the following services: HTTP server, logger server, configuration provider and state machine. In addition two more objects were created to configure the system, these two objects are the "discharge configurator" and the "advanced configurator", which are used to change the control configuration via two HTML webpages.

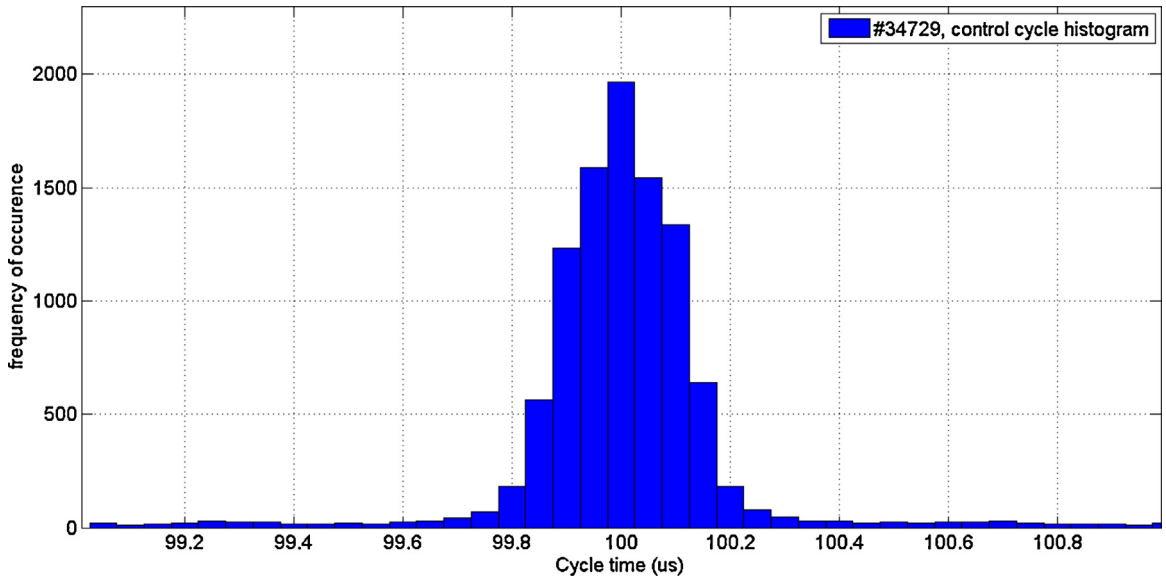

Fig. 5. Control cycle histogram for pulse #34729. The control cycle is centered in 100 μs and the jitter is generally lower than 0.4 μs.

After commissioning the control system, the control cycle was measured using a control cycle variable produced automatically by the MARTe framework. This introspection ability is very useful for debugging the real-time performance of the control system code. MARTe stores the execution time for each individual GAM and stores the overall control cycle as well. The frequency of occurrence of a certain control cycle is depicted in Fig. 5. As it can be observed, the control cycle is clearly centered in 100 μs and has a jitter generally inferior to 0.4 μs. Although not depicted in Fig. 5, the maximum observed jitter is around 6 μs.

Since ISTTOK real-time code normally uses about 50 μs to execute all GAMs it was decided to use a 100 μs control cycle to avoid exceeding the defined control cycle that would cause a non-constant control cycle. The GAMs that contribute the most for the code total execution time are the GAM responsible for the tomographic reconstruction [16] that uses about 12 μs of computation time to provide plasma center of mass position and the three power supplies communication GAM instances (one for each PS), each using about 6 μs to be executed. Apart from the system control cycle, the main latency sources are the digital filtering of the ADC data described in Section 2 (about 25 μs) and the delay in the power supplies actuation. There are three main sources of delays in the PS actuation; (i) the communication with the quad UART board driver (lower than 50 μs), (ii) data transmission time (lower than 25 μs) and (iii) PS actuation after receiving the set-point (lower than 35 μs).

3.2. Human–machine interface

The machine configuration for the real-time control is provided by two objects named "Discharge Configuration" and "Advanced Configuration" that interact with MARTe and use the BaseLib2 libraries as depicted in Fig. 4. These objects are not GAMs and therefore they are isolated from the real-time thread. Since both have a HTTP server the natural way to interact with these objects is to use an internet browser. The web pages are available through the ISTTOK internal network.

These two objects are written in C++ code and implement a web page with HyperText Markup Language (HTML) and Javascript™ code. These configuration objects also include an automatic graphical waveform representation that scales the waveform graph automatically after inserting or deleting a waveform point generating the correspondent scalable vector graphics (SVG) extensible markup language (XML) type code which automatically updates the waveform representation.

The "discharge configurator" (Fig. 6) is divided in 3 sections; (i) Diagnostics settings, (ii) controller settings and (iii) time-windows configuration.

In the diagnostic settings, the operator can specify each diagnostic contribution for the calculated quantity since plasma parameters can be obtained from several diagnostics.

In the control settings, the operator configures the discharge duration, number AC cycles and selects the auto-breakdown feature. The user also configures the control constants for scenario control (see Section 3.4).

In the "time windows" section, the user can select the type of control for each power supply during a specific time interval; the user can add up to seven different time windows per AC semi-cycle time (one for positive plasma current and another for negative plasma current). This subject is further discussed in Section 3.3.

The "Advanced configuration" (Fig. 7) is similar to the "discharge configurator" but contains the less likely to be changed parameters of a discharge. It contains (i) the power supplies limits, (ii) the plasma position limits, (iii) the serial port address for communication with the PS, (iv) the real-time thread priority, (v) the CPU core were the real-time thread is running and (vi) the predictor constants for the iron core saturation.

This configuration editor contains also the waveforms for breakdown, current inversion and gas feedback.

As a matter of fact, these two objects are a frontend for editing the MARTe configuration file (long text file with human readable information similar to XML). This file is where MARTe and all the GAMs get the configuration from. When the user clicks on the save button, the configuration file is overwritten with all the parameters stored on these objects. After all changes are made to the real-time configuration, the user can save and commit it. This action will trigger MARTe to destroy all the objects (including the real-time thread and GAMs) and reload the new configuration file. Using this strategy, MARTe restarts hereafter with the new configuration parameters. After each pulse this configuration file is stored with the corresponding pulse number so that it can be retrieved when the user wants to repeat a pulse or needs only small changes to a

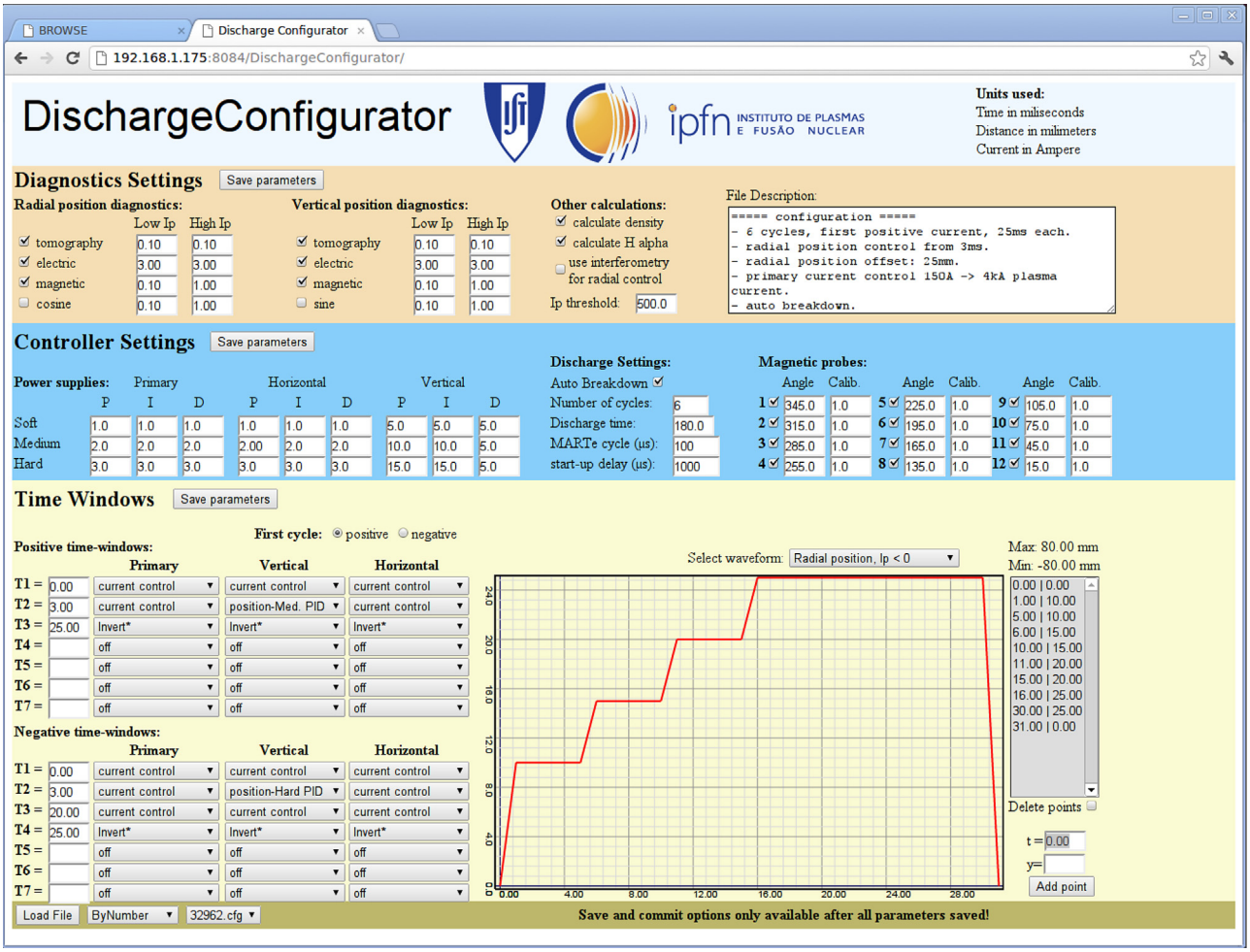

Fig. 6. HTML page used for programming the essential discharge parameters. In this page the user programs the control methods during each part of the discharge using the time-windows paradigm.

known pulse. These two objects also have a specific load button for this operation.

3.3. Event-driven control

The GAM named "TimeWindowsGAM" is the main real-time state machine for the discharge. This GAM changes the control type based on external events such iron core saturation, current ramped up after auto break-down or inversion or machine protection events. This GAM also changes the control sequentially according to the programmed time-windows. There are several discharge modes that can be programmed by the user corresponding to user-defined waveforms (a set of waveforms for each plasma direction) such as breakdown, inversion, current control, scenario control (calculated plasma properties such as plasma position, plasma current, etc.), synchronized and unsynchronized gas puffing and density feedback. The list of inputs and outputs of the TimewindowsGAM is depicted in Fig. 8.

The TimeWindowsGAM controls which waveforms are in use and the waveforms timing, for example, if the auto-breakdown is selected by the user, the TimeWindowsGAM starts applying the programmed breakdown waveforms from time = 0 and when the plasma current is above a certain threshold the TimeWindowsGAM changes the waveform mode to the one defined for the first time-window (starting again by the time = 0 for those waveforms). This has the advantage of executing a certain set of tasks without pre-programming exactly their timing (event driven format). This means that with exactly the same discharge pre-programming the discharge timing can be different for distinct machine conditions. This behavior is depicted in Fig. 9.

Another type of waveform switching occurs when a time-window ends and another time-window starts. This transition is also assured by the TimeWindowsGAM. In this case the TimeWindowsGAM continues to increase the discharge time at the normal pace and changes the waveform mode (current control or scenario control depending on the discharge configuration), so that the waveform has the correct timing and the output from the selected waveform (direct PS current or scenario waveforms). The transition in control is done afterwards by the ControllerGAM.

When there is a time-window named "inversion" (meaning that the plasma current is to be inverted at a certain time) or there is an event of iron core saturation (when flux driven through the iron core approaches the maximum value the plasma current starts to decay even if the primary current is largely increased) the TimeWindowsGAM ends the current plasma cycle (referred to either positive or negative current cycles) and if additional cycles were programmed, the GAM switches the actuators control mode to what is programmed in the first time-window of the opposite plasma current direction (the modified discharge time is set to time = 0). If the user selects a large number of AC semi-cycles the odd semi-cycles have similar programming than the first and

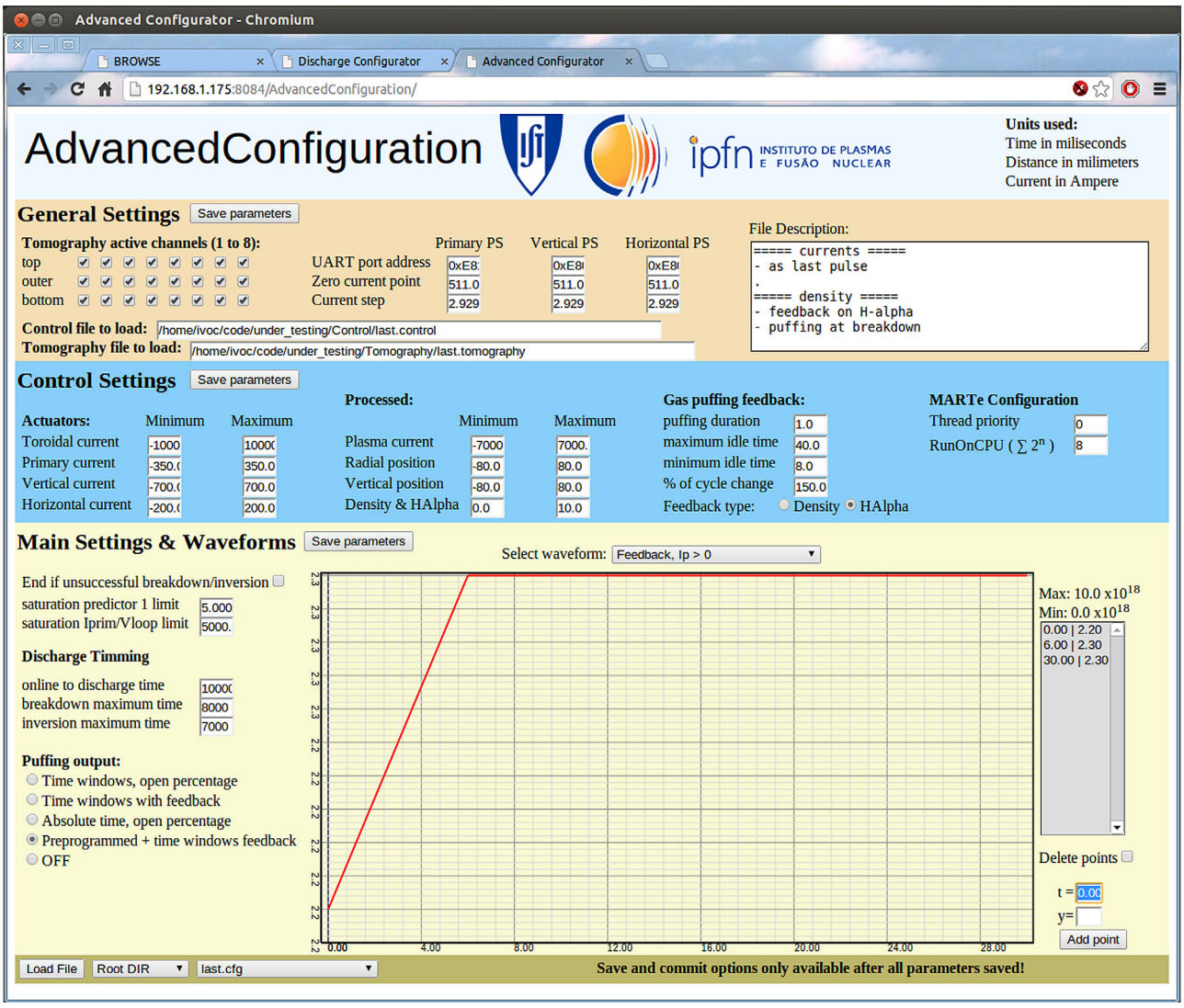

Fig. 7. "Advanced configuration". HTML page containing the plasma parameters that are less likely to be changed from discharge to discharge.

the even semi-cycles have similar programming than the second semi-cycles and this is repeated until the end of the discharge alternating between the positive time-windows set and the negative time-windows set.

This GAM also controls the end of the discharge, ending the discharge if the discharge time has passed the limit defined by the user. This GAM also stops the discharge if the number of alternate discharge cycles have reached the limit value programmed by the user. There are event driven interrupts that can also stop the discharge. If a soft or hard stop alarm is received from the MachineProtectionGAM it can decide how to better shutdown the pulse.

In a standard discharge, three sets of pre-programmed waveforms (each set has one waveform per PS) are normally used, the first is the pre-programming the plasma breakdown currents and the other two are the plasma inversion from positive plasma current to negative plasma current and vice versa. These waveforms are used to program directly the current in the ISTTOK power supplies and when the objective is achieved the control jumps to the next objective, for example: when in breakdown, the control system monitors the plasma current, and when it is greater than a defined value, it means that the breakdown was successful.

In addition to the previously explained event-driven actions, there is another event that can interfere with the normal discharge

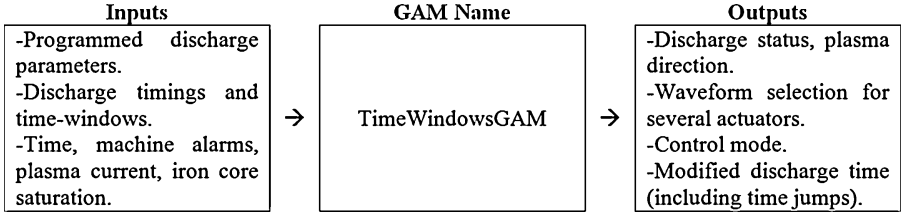

Fig. 8. TimeWindowsGAM inputs and outputs.

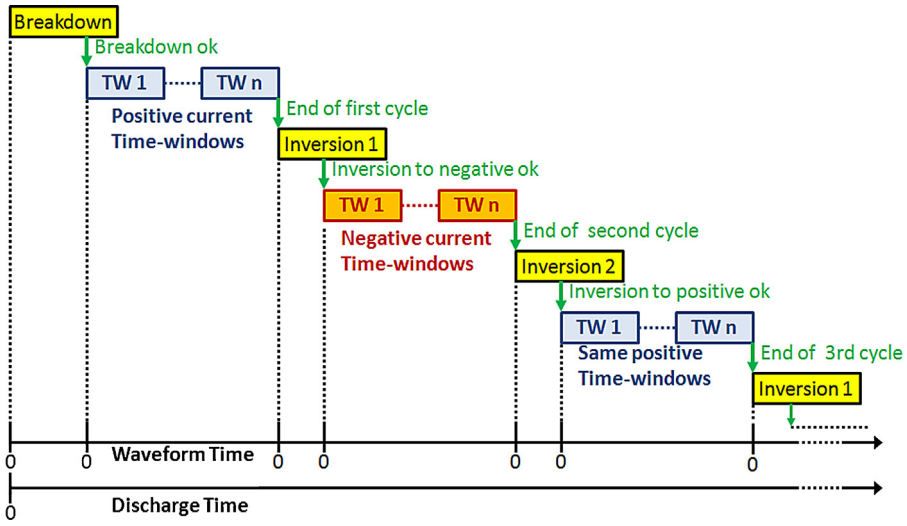

Fig. 9. Discharge time and waveform time of a discharge with auto-breakdown and pre-programmed inversion waveforms. This figure shows the effect of the TimeWindowsGAM on the time that is sent to the user defined waveforms. This feature allows the repetition of the alternate current semi-cycles without programming more than two semi-cycles.

sequence, the iron core saturation exception handling. Since ISTTOK has an iron core, the tokamak can be seen as a transformer, when the iron core is saturated, it is not possible to drive more flux trough the iron core. When this happens the loop voltage drops dramatically for the same magnetizing field current, making the plasma current inductive drive impossible to continue in the same direction. When this event happens, depending on the user discharge configuration, the discharge can either be shutdown or the plasma current must be inverted without any further delay. If the discharge was programmed not to shutdown on this event, the time-window sequence is interrupted and the controller skips this execution and jumps to the next inversion phase, allowing the AC discharge to continue normally.

3.4. Feedback control

Under the new control system the ISTTOK power supplies have two modes of operation; (i) current control and (ii) scenario control. In the first type of control the power supplies receive the current set-point (SP) and their internal controller regulates the current output to match the requested SP. In the latter form of control each power supply is used to control the respective scenario. The magnetizing field power supply is used to control the plasma current, the horizontal and vertical field power supply control the vertical and the radial position respectively.

The ISTTOK power supplies always receive their set-points in current values. The plasma parameters are controlled using PID algorithms for feedback. The process variable is the plasma parameter to be controlled, the plasma parameter waveform is programmed in the main control system prior to the discharge and the result of this PID algorithm is the current reference that is sent to the respective actuator.

For scenario control the operator can program the K_p, K_i, K_d for a standard proportional-integral-derivative (PID) controller defined by Eq. (1). The PID controller can be implemented in differential form that is obtained from differentiating Eq. (1), but in ISTTOK the implemented PID corresponds to Eq. (4). This format of PID controller was selected because it is more robust and anti-windup. It is obtained by simply limiting the output with the operational value of the corresponding output. For example the output of the PID controller for the magnetizing field PS is bounded between −300 A and +300 A corresponding to the PS operational limits.

$$\mathrm{Out} = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{d}{dt} e(t) \tag{1}$$

$$e = SP - PV \tag{2}$$

$$\mathrm{Out}_k = \mathrm{Out}_{K-1} - K_P(e_k - e_{k-1}) + K_i T e_k - \frac{K_d}{T}(e_k - 2e_{k-1} + e_{k-2}) \tag{3}$$

$$\mathrm{Out}_k = \mathrm{Out}_{K-1} - K_P(PV_k - PV_{k-1}) + K_i T e_k - \frac{K_d}{T}(PV_K - 2PV_{k-1} + PV_{k-2}) \tag{4}$$

In Eq. (4) the term error (e) defined by Eq. (2), was replaced by the process variable (PV) in some parts of the equation, this prevents a jump in the output request when the reference changes dramatically from one control cycle to the next one.

When there is a transition from current control to scenario control with a PID controller the ControllerGAM uses the respective power supply last current measure and loads this value on the variable $\mathrm{Out}_{(k-1)}$ of Eq. (4), this prevents a jump in the current request to the power supply and consequently "smoothes" the convergence to the required set-point.

The typical behavior on the scenario control is depicted in Fig. 10. In this figure it is possible to observe the effect of the PID control on the ISTTOK magnetic equilibrium process variables; (i) plasma current, (ii) radial position and (iii) the vertical position, which are controlled by the primary field PS, the vertical field PS and the horizontal field PS respectively. These circuits have a very low coupling between them, but since ISTTOK produces circular plasmas, the last flux surface is defined by the limiter. This means that the plasma position strongly affects the plasma current circuit when the plasma is pushed to the limiters since the plasma is reduced in size undermining the plasma capacity to conduct a large portion of current.

The gas feedback is based on a piezoelectric valve that is either open or closed. Depending on the reference waveform and the associated PV (plasma density or $H\alpha$ radiation), the gas is puffed more or less frequently with a time opening (normally 1 ms or less) defined by the machine operator.

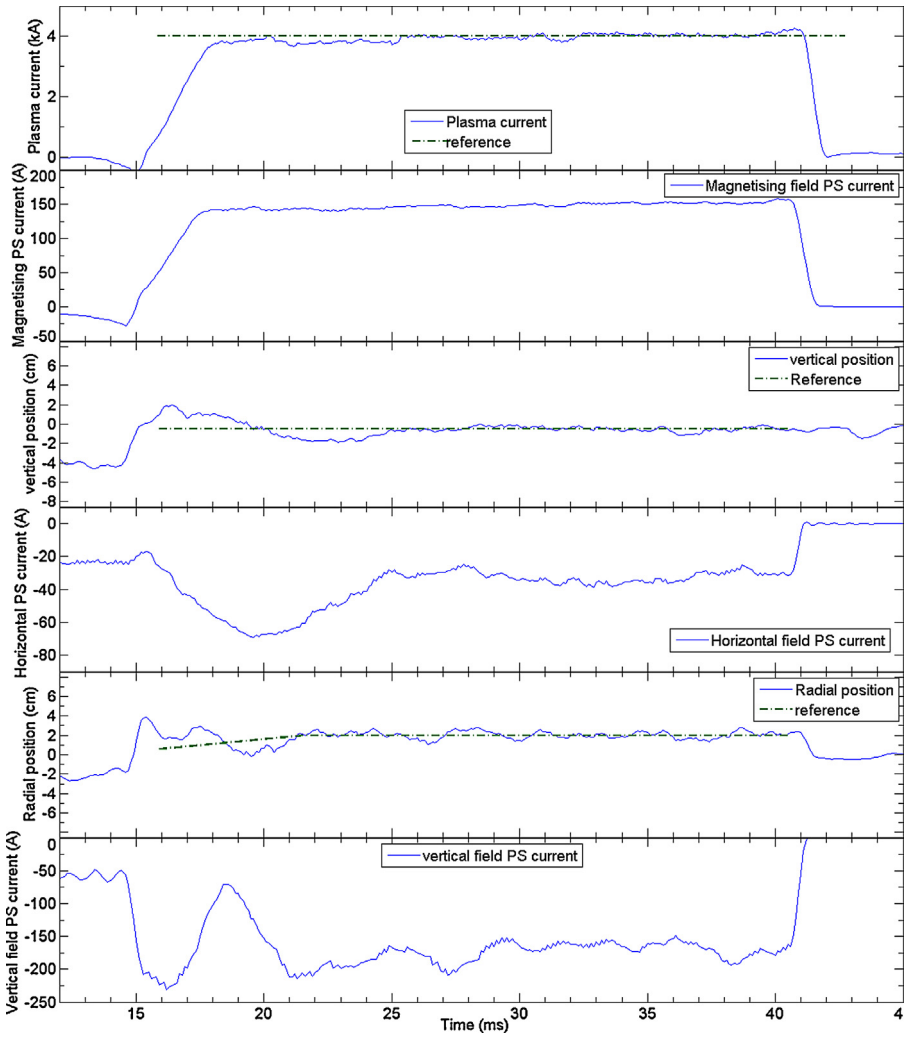

Fig. 10. Power supplies behavior during scenario control mode.

4. Operational results

Fig. 11 shows an ISTTOK AC discharge with 24 semi-cycles corresponding to 0.64 s of plasma. This discharge (pulse number 34327) was obtained using the new control system and was programmed using the new HMI. The machine operator programmed two sets of time-windows, one for the positive cycles and another for the negative cycles. This discharge also used the auto-breakdown and pre-programmed plasma current inversion features. Density/radiation control was used throughout the discharge. A comparison between an old AC discharge and a recent discharge is available in [3].

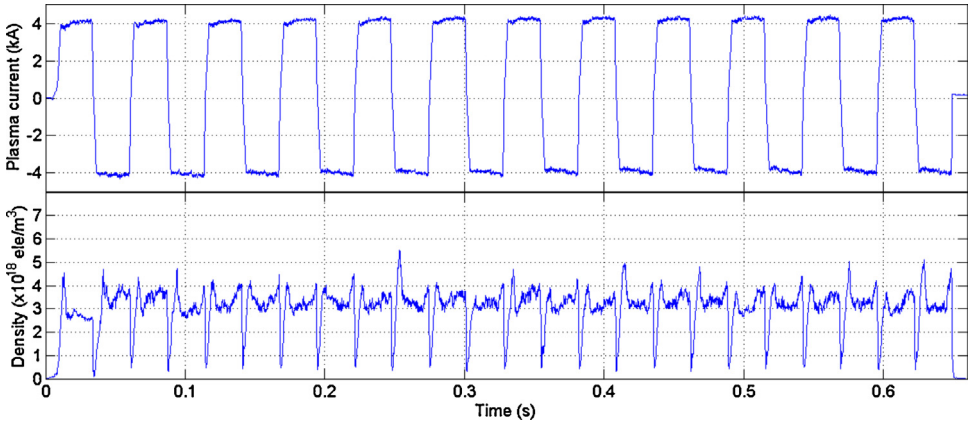

Fig. 11. Pulse #34327. This pulse was programmed with the auto-breakdown and pre-programmed inversion features. In the plasma flat-top plasma is in position control. The gas valve aperture was in feedback control with the H-alpha radiation.

5. Conclusions

The MARTe framework provided the adequate tools for developing the ISTTOK tokamak real-time control and a 100 μs control cycle was achieved with a jitter generally lower than 0.4 μs.

Two objects were produced to interact and configure the control system. These two configuration objects are isolated from the real-time thread that runs the ISTTOK control code. When the pulse ends, the real-time configuration file is stored and the machine operator can now load old configurations and repeat the pulse including event-driven configurations.

The ISTTOK control system demonstrated reliable operation for more than 4000 discharges up to date and enabled ISTTOK to extend discharge duration per AC pulse opening new opportunities for testing materials exposure to plasma environment.

Time windows operation with different control strategies was successfully implemented and together with the graphical HMI software allows an easy path to program a discharge capable of handling complex events. Several synchronization strategies can be implemented using the event driven paradigm largely improving the number stable AC cycles.

Although the system can still be configured for time deterministic operation, the even-driven discharge programming is now the standard discharge operation. The event-driven paradigm can be very useful when programming long discharges with several objectives per discharge such as the envisaged for ITER [15].

Acknowledgments

This work was supported by EURATOM and carried out within the framework of the European Fusion Development Agreement. IST activities also received financial support from “Fundação para a Ciência e Tecnologia” through project Pest-OE/SADG/LA0010/2011. The views and opinions expressed herein do not necessarily reflect those of the European Commission.

References

[1] H. Fernandes, C.A.F. Varandas, J.A.C. Cabral, H. Figueiredo, R. Galvao, A.J. Batista, Engineering aspects of the ISTTOK operation in a multicycle alternating flat-top plasma current regime, Fusion Engineering (1997) 576–580, http://dx.doi.org/10.1109/fusion.1997.687106, 17th IEEE/NPSS Symposium, vol. 1, 6–10 October 1997.

[2] J.A.C. Cabral, H. Fernandes, H. Figueiredo, C.A.F. Varandas, Operation of the tokamak ISTTOK in a multicycle alternating flat-top plasma current regime, Nuclear Fusion 37 (11) (1997) 1575–1581.

[3] I.S. Carvalho, P. Duarte, H. Fernandes, D.F. Valcárcel, P.J. Carvalho, C. Silva, et al., ISTTOK control system upgrade, Fusion Engineering and Design 88 (October (6–8)) (2013) 1122–1126.

[4] I.S. Carvalho, D.F. Valcarcel, H. Fernandes, B.B. Carvalho, J. Sousa, A. Pironti, et al., Fast digital link for a tokamak current source control, Power Electronics, Electrical Drives, Automation and Motion (2008), SPEEDAM 2008.

[5] F.G. Rimini, F. Crisanti, R. Albanese, G. Ambrosino, M. Ariola, G. Artaserse, et al., First plasma operation of the enhanced JET vertical stabilisation system, Fusion Engineering and Design 86 (October (6–8)) (2011) 539–543, http://dx.doi.org/10.1016/j.fusengdes.2011.03.122, ISSN 0920-3796.

[6] B. Gonçalves, J. Sousa, B.B. Carvalho, A. Batista, A. Neto, B. Santos, et al., ITER fast plant system controller prototype based on ATCA platform, Fusion Engineering and Design 87 (12) (December 2012) 2024–2029.

[7] A.J.N. Batista, A. Neto, M. Correia, A.M. Fernandes, B.B. Carvalho, J.C. Fortunato, et al., ATCA control system hardware for the plasma vertical stabilization in the jet tokamak, IEEE Transactions on Nuclear Science 57 (April (2)) (2010) 583–588.

[8] A. Neto, H. Fernandes, A. Duarte, B.B. Carvalho, J. Sousa, D.F. Valcárcel, et al., FireSignal-data acquisition and control system software, Fusion Engineering and Design 82 (5–14) (October 2007) 1359–1364, http://www.sciencedirect.com/science/article/pii/S0920379607000713.

[9] http://atca.ipfn.ist.utl.pt/other-developments/firesignal.html

[10] D. Alves, A.C. Neto, D.F. Valcálrcel, R. Felton, J.M. López, A. Barbalace, et al., A new generation of real-time systems in the JET tokamak, in: 18th IEEE NPSS Real Time Conference (NPSS-RT2012), Berkeley, CA, June 2012, 2012.

[11] A.C. Neto, D. Alves, L. Boncagni, P.J. Carvalho, D.F. Valcárcel, A. Barbalace, et al., A survey of recent MARTe based systems, IEEE Transactions on Nuclear Science 58 (4) (2011) 1482–1489.

[12] A.C. Neto, F. Sartori, F. Piccolo, R. Vitelli, G. De Tommasi, L. Zabeo, et al., MARTe: a multi-platform real-time framework, IEEE Transactions on Nuclear Science 57 (April (2)) (2010) 479–486.

[13] http://www.gentoo.org

[14] D.F. Valcárcel, A. Neto, J. Sousa, B.B. Carvalho, H. Fernandes, J.C. Fortunato, et al., An ATCA embedded data acquisition and control system for the compass tokamak, Fusion Engineering and Design 84 (7–11) (2009) 1901–1904.

[15] A. Winter, D.J. Campbell, T. Casper, Y. Gribov, W-D. Klotz, L. Scibile, et al., Status of the ITER plasma control system conceptual design, in: Proceedings of ICALEPCS2009, Kobe, Japan, 2009.

[16] P.J. Carvalho, H. Thomsen, R. Coelho, P. Duarte, C. Silva, H. Fernandes, ISTTOK plasma control with the tomography diagnostic, Fusion Engineering and Design 85 (April (2)) (2010) 266–271.