Integration of a Real-Time Node for Magnetic Perturbations Signal Analysis in the Distributed Digital Control System of the TCV Tokamak.

C. Galperti, S. Coda, B. P. Duval, X. Llobet, P. Milne, O. Sauter, J. M. Moret, D. Testa and the TCV team

***Abstract*—This paper presents the integration of an acquisition and computing unit capable of acquiring and processing fast magnetic signals in real-time in the control system of the TCV tokamak. All aspects of system integration and testing are reported, leading to testing of the system on plasma discharges. An example of a real-time analysis algorithm designad for detecting and classifying NTM plasma instabilities is also described.**

I. Introduction

THE analysis of fast magnetic signals measured by magnetic probes installed on a tokamak vessel has been widely employed to detect and analyze a number of plasma instabilities, notably rotating tearing modes. The high frequency components of the measured signals contain information directly related to rapid changes within the plasma, for instance those generated by tearing mode magnetic islands rotating in the torus. Off-line analysis of these signals employs powerful mathematical tools such as spectrograms, principal component analysis and amplitude-phase fitting algorithms [1] [2] [3] [4]. These provide copious information on magnetic islands topology, size and evolution.

In this paper we describe the integration of these techniques into the TCV digital real time control system to execute advanced magnetic analysis codes during the tokamak discharge. The final goal is to provide plasma health status information to decision making algorithms that can initiate actuator reactions. Severe design constraints, posed by TCV's real-time environment, had to be addressed. TCV presently employs a distributed real time control system capable of controlling the entire plant during a discharge, consisting of commercial PCs interconnected with a self controlled high speed fiber optic digital link [5] [6] [7]. Each node of the system can house ADC and DAC boards that interact with the plant.

This paper describes the integration of a supplementary real-time magnetic coil analysis node into this system, taking into consideration the restrictions posed by the demanding real-time environment whilst respecting all the pre-existent features of the control system. It is organized as follows: section II introduces the TCV distributed control system, section III describes the integration of the new node and finally section IV presents system tests during TCV operation.

Manuscript received May 30, 2016. 978-1-5090-2014-0/16/$31.00 ©2016 IEEE

C. Galperti, S. Coda, B. P. Duval, X. Llobet, O. Sauter, J. M. Moret and D. Testa are with Swiss Plasma Center (SPC) of the Ecole Polytechnique Fédérale de Lausanne (FPFL). CH-1015 Lausanne, Switzerland. Contact email: cristian.galperti@epfl.ch

P. Milne is with D-TACQ Solutions Ltd. James Watt Building, Scottish Enterprise Technology Park, East Kilbride, G75 0QD, Scotland UK.

II. The TCV distributed digital control system

The TCV tokamak has a fully functional digital real time control system capable of controlling almost all aspects of a plasma discharge. The system is based on a real-time sharing data network (reflective memory) of modular computer nodes, each an embedded or desktop PC which may include local ADC and/or DAC cards. Owing to the restricted resources, particularly in terms of manpower, available to the TCV team, design choices were made favoring simplicity, flexibility and maintainability. The main requirements and resulting design choices are listed in Table I. From the top level, the system is a distributed acquisition and processing real-time system build on top of standard COTS PCs augmented with ADC and DAC boards and high speed dedicated communication links. Hence its French name: Système de Contrôle Distribué (or SCD), which translates into Distributed Control System.

A. Real-time computer nodes

Figure 1 shows the SCD control system layout with the connectivity to the diagnostics and actuators. Some nodes are connected to a compact-PCI (cPCI) crate hosting one or more D-tacq ACQ-196 acquisition cards with 96 ADCs, and output cards housing 16 or 32 DACs. Some nodes are only connected via the reflective memory network and act as computational nodes. At present there are seven nodes, the seventh is the one described in this work.

All the nodes exchange data with all the others exploiting the reflective memory link, (RFM in brief) an industry standard high-speed digital communication system that transparently synchronizes a shared memory to all nodes [12]. The synchronization is performed without user and/or kernel intervention but it has the drawback that no data handshaking is performed so the user is responsible of preventing read-write race conditions. In SCD this is accomplished by using a node as a RFM master which synchronizes all operations on the RFM of all others nodes w.r.t. a common timebase. On each clock of the RFM master node (this is assigned by the SCD operator), to prevent read/write collisions, we alternate a read or write from/to the RFM using DMA (i.e. write on odd cycle and then read on even cycle). All nodes must be synchronized with the write and read cycle, and all the nodes write (or read) at the

TABLE I
REQUIREMENTS AND RESULTING DESIGN CHOICES FOR THE TCV DISTRIBUTED DIGITAL REAL TIME CONTROL SYSTEM

Aspect	Requirement	Design choice
Hardware	Minimise learning curve for new hardware, minimise spares, cost	Base system on same ADC modules (D-tacq ACQ-196, cPCI form factor) already used for many TCV diagnostics. Use COTS Linux PCs.
Sampling rate	1 kHz (shape control) to 250 kHz (MHD mode detection), 100 kHz (vertical stabilisation)	Use different nodes that can run at different sampling rates, vertical stabilisation (1 channel) still handled by the analog system.
Real-time performance	Hard real-time performance on COTS Linux PC	Standard Linux OS with interrupt masking during real-time code execution, progressive migration to multi-core processors and SMP kernels capable of reserving cores for real-time processes.
Diagnostics	$>$ 200 diagnostic channels (100+ magnetics, 64 soft X-ray, 14 interferometer...) plus flexibility to add more in the future.	ACQ-196 ADC front-ends with 96 channels each, maximum of 3 cards per node used so far.
Actuators	Control all TCV actuators including 16 PF coils, 7 EC launchers, 6 EC power supplies, 4 gas valves, with possibility for expansion.	DAC modules with up to 32 channels each, possibility to connect different nodes to different actuators.
Inter-node communication	Transparent, low overhead and kernel-independent communication between different nodes of the control system.	Reflective memory system (from General Electric) with reserved memory space for each node.
Algorithms	Rapid algorithm development, debugging, simulation, sharing, archiving and tuning during operations	Real-time programming entirely with Simulink block-programming language, automatic conversion to C.
Data storage	Algorithms and data must be stored for later analysis, troubleshooting	Real-time input-output data stored to MDSplus database, operative code folder under SVN version control and committed at every shot.

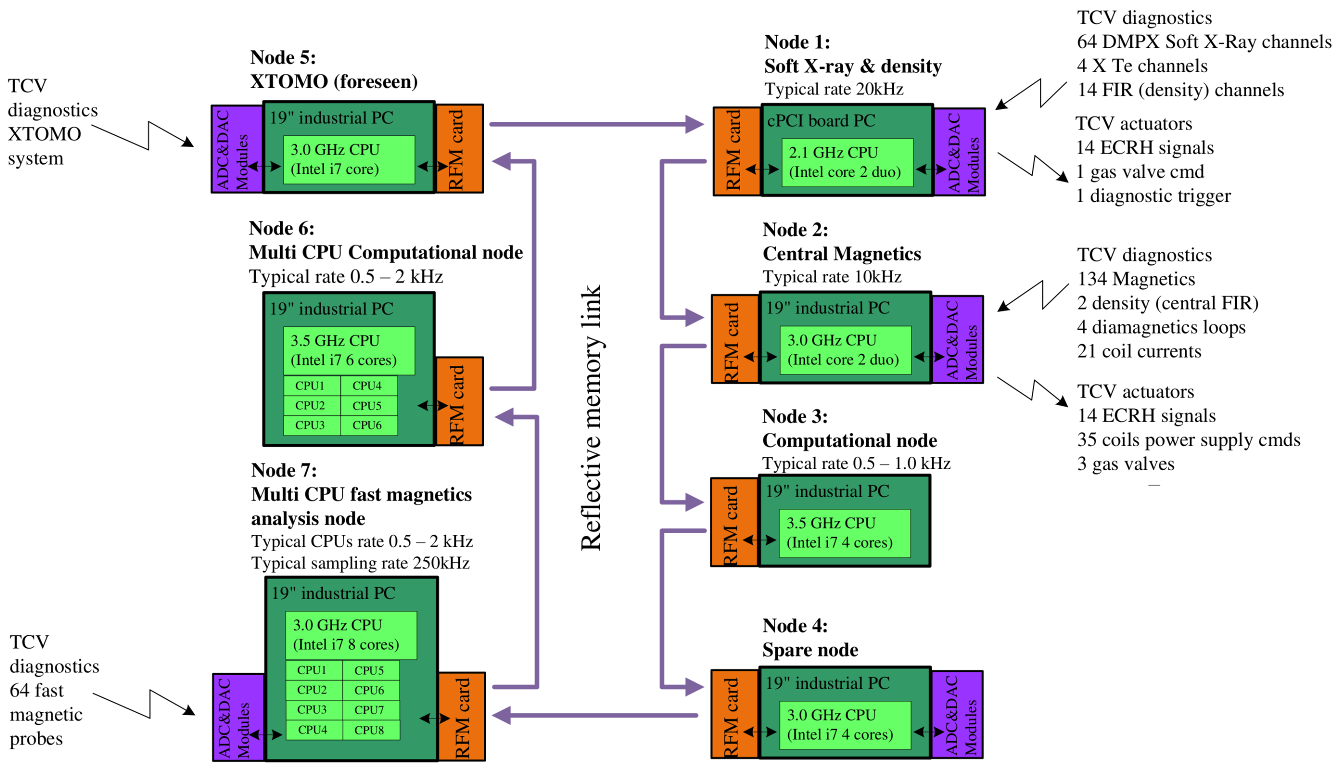
TCV diagnostics XTOMO system
Node 5: XTOMO (foreseen)
ADC&DAC Modules
19" industrial PC
3.0 GHz CPU (Intel i7 core)
RFM card
Node 6: Multi CPU Computational node
Typical rate 0.5 – 2 kHz
19" industrial PC
3.5 GHz CPU (Intel i7 6 cores)
CPU1 CPU2 CPU3 CPU4 CPU5 CPU6
RFM card
Node 7: Multi CPU fast magnetics analysis node
Typical CPUs rate 0.5 – 2 kHz
Typical sampling rate 250kHz
19" industrial PC
3.0 GHz CPU (Intel i7 8 cores)
CPU1 CPU2 CPU3 CPU4 CPU5 CPU6 CPU7 CPU8
TCV diagnostics 64 fast magnetic probes
Reflective memory link
Node 1: Soft X-ray & density
Typical rate 20kHz
cPCI board PC
2.1 GHz CPU (Intel core 2 duo)
TCV diagnostics
64 DMPX Soft X-Ray channels
4 X Te channels
14 FIR (density) channels
TCV actuators
14 ECRH signals
1 gas valve cmd
1 diagnostic trigger
Node 2: Central Magnetics
Typical rate 10kHz
3.0 GHz CPU (Intel core 2 duo)
TCV diagnostics
134 Magnetics
2 density (central FIR)
4 diamagnetics loops
21 coil currents
TCV actuators
14 ECRH signals
35 coils power supply cmds
3 gas valves
Node 3: Computational node
Typical rate 0.5 – 1.0 kHz
3.5 GHz CPU (Intel i7 4 cores)
Node 4: Spare node
3.0 GHz CPU (Intel i7 4 cores)

Fig. 1. Top view of the TCV Système de Contrôle Distribué.

same time, although nodes may skip cycles depending on their control rates [7].

The nodes with ADCs and DACs are interfaced to machine's diagnostics systems and to machine's actuators. Node 1 is interfaced to 2 soft-X diagnostics (DMPX, or Duplex Multiwire Proportional soft X-ray counter, a pinhole type soft-

X camera and X-Te, a four filter soft-X spectrometer that provides central electron temperature using the differential filter method). It is also interfaced to the 14 vertical chords of the Far InfraRed (or FIR) interferometer providing the electron density profile information. Node 2 acquires all magnetics measurements from the tokamak and is thus responsible for plasma shape and position control; it also acquires the central FIR channel for real-time control of the density. This node is used routinely as the main plasma position and density controller and it is almost always the RFM master node. Node 3 is a computational node that computes plasma magnetic equilibrium in real-time. Node 4 is a replacement node for node 2, while node 5 is an acquisition and processing node connected to the 200 channel soft x-ray tomographic system. Node 6 is a very recently installed multicore computational node that has been used to run multicore complex control codes (a faster real time equilibrium reconstruction replica and RAPTOR based advanced plasma performances controllers). Finally, node 7 is devoted to real-time analysis of fast magnetic perturbations in the plasma.

B. Software organisation

Fig. 2 presents the modular, portable software organization of the SCD control system. The SCD code is divided into two main sections:

1) **Hardware interface code** written in C/C++ language by the system developers, it provides the input/output interface to the control algorithm code. Once compiled, the executable is uploaded to real-time computer nodes, where it is considered fixed and unchanging between plasma shots. However, the hardware code can change its functional behaviour depending on external configuration parameters, such as a varying operational mode with or without suspending interrupts, the number of ADC/DAC cards, number of cores to be used (for multicores nodes), the data shared via RFM memory, etc.
2) **Control algorithm code** realised in MathWorks-Simulink [13] block programming language by the control algorithm author(s), using Simulink templates given by the system developers. It performs signal processing and computational actions to provide output signals consisting of new values for the actuators, the reflective memory and other signals (probe signals) used for post-shot analyses. This algorithm is in user-friendly Simulink block format and is automatically converted into target code that is a dynamically linked shared object library by MathWorks Simulink Embedded Coder (SEC) [14].

This modular architecture offers great advantages. The abstraction from the hardware specific code makes the control algorithm code portable, i.e. control algorithms tested on TCV could be readily reused in other fusion facilities with minor adaptations. Another salient advantage of this modularity is its flexibility, i.e. the control algorithm can be developed on any computer equipped with Matlab-Simulink without requiring the Simulink Embedded Coder (SEC) package installed, and then simulated for debugging using the real diagnostic signals of past shots. This feature has already been exploited by external collaborators from several institutions across Europe and beyond to test their control algorithms on TCV. Only once the correctness and the robustness of the control algorithm are verified in simulation using experimental data of the previous shots, is it commissioned into RT system. Another essential advantage is that algorithm authors can use the extensive Simulink block library for standard components such as filters, integrators, matrix multiplications and other advanced signal processing tools. For TCV, standard custom-built blocks are provided for interfacing algorithms with TCV diagnostics and actuators, implementing calibration factors, signal selection, actuator constraints and saturation, etc.

III. Integration of node 7

Having introduced the general architecture of TCV SCD control system, now we explain the integration of the new node 7.

A. Hardware and the hardware interface code

From the point of view of the hardware architecture, node 7 is similar to all others SCD nodes, being constituted of a cPCI acquisition crate fitted with a DTACQ ACQ196 96 channels synchronous digitizer card, a rear transition module (DTACQ RTM type T) and an industrial PC, acting as the host processing unit. The industrial PC is fitted with a Reflective Memory Card of the same kind as all the other nodes of the system to communicate with real time network.

As mentioned, node 7 is devoted to fast magnetic signal acquisition and processing in real time. The main technical requirements to be fulfilled are as follows:

1) **Sampling frequency.** The frequency of the usually observed plasma instabilities of the TCV tokamak lie in the range from some hundreds of Hz to 100kHz. Furthermore, phase coherence based algorithms are often used to infer coherent plasma structures and so simultaneous sampling of signals is mandatory. Simultaneous sampling of at least 32 magnetic signals at frequencies above 200 ksps is required.
2) **CPU process time.** The previous requirement limits the CPU processing time since higher sampling rates lower the available processing time on the CPU (being equal to the sampling period). The aim of this node is to execute complex analysis algorithms on high speed multichannel data streams within a reasonable processing time.
3) **Multi-processing capability.** Processing algorithms may be spread over multiple cores to increase the computational power.
4) **Legacy RFM interface**. Data sharing with other nodes of the control system have to comply with the existing data distribution scheme described in II-A.
5) **Processing algorithms written in Mathworks/Simulink**. The analysis algorithms should be developed in Matlab/Simulink and automatically uploaded to the analysis node. Therefore, node 7 must

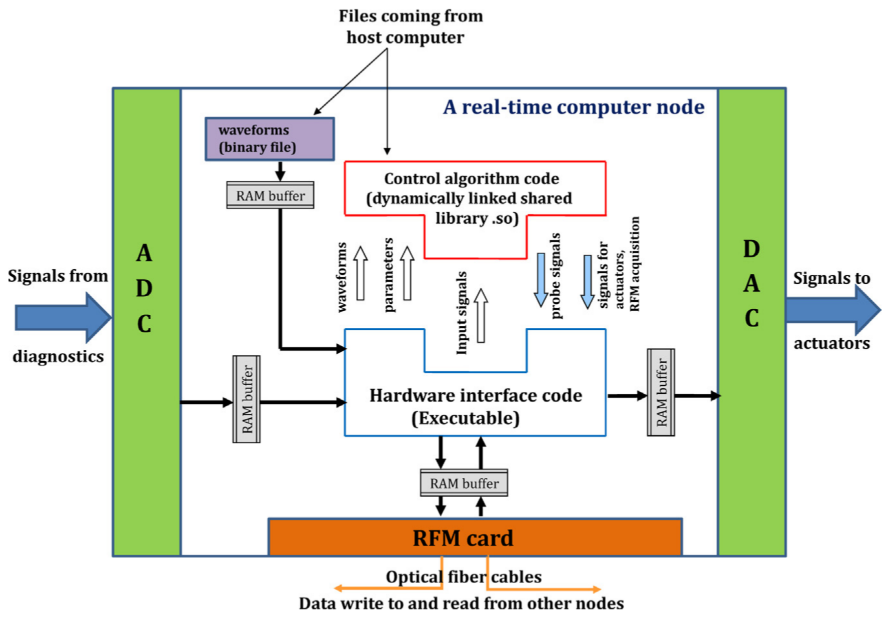
Files coming from host computer
A real-time computer node
waveforms (binary file)
RAM buffer
Control algorithm code (dynamically linked shared library .so)
waveforms
parameters
input signals
probe signals
signals for actuators, RFM acquisition
Signals from diagnostics
A D C
D A C
Signals to actuators
RAM buffer
Hardware interface code (Executable)
RAM buffer
RAM buffer
RFM card
Optical fiber cables
Data write to and read from other nodes

Fig. 2. Software organization in each node.

TABLE II
TASK ASSIGNMENT TO CPU CORES OF SCD NODE 07

Core no.	Task
1	Linux kernel (forced here in the boot command).
2	Data communications with DTACQ196/RTM-T acquisition subsystem, data dispatch to processing cores.
3	Processing core 1.
4	Processing core 2.
5	Processing core 3.
6	Processing core 4.
7	Synchronization handshaking and data exchange on the RFM interface.
8	Not used.

be fully compatible with the software architecture described in figure 2.

To comply with the above requirements, we adopted a packet acquisition-processing scheme where the ADCs have a double buffered data path allowing the CPUs to operate on each data packet as described in [8]. The acquisition hardware employs an ADC subsystem with a D-TACQ ADC196 board augmented with a RTM-T rear transition module [11]. This module provides a high speed PCI-Express 1x link to the host PC, which is equipped with a recent, high-end, motherboard hosting an Intel i7-5960X 8-core 3GHz processor. The adopted kernel is Scientific Linux 6.7 especially configured for user-mode RT capabilities with a dedicated hardware interface code fully compliant with fig. 2. The ad-hoc configured kernel together with hardware interface code exploit the multicore CPU by distributing tasks on the cores as summarized in table II. The kernel is restricted to the first core by Linux init command options flexibility with all main kernel threads and user processes executed on this core leaving the others free for user real-time processes and/or threads; the only kernel threads remaining on these cores are the basic Linux kernel threads required for handling the cores (such as the interrupt handling kthread, the migration kthread and the watchdog kthread).

Once the system enters the real-time state before the plasma discharge, the hardware interface code is launched as an user process on core 2. Node 7's hardware interface code is a multi-thread process that spawns a thread per used core to distribute the computational tasks. Besides hosting the hardware interface code, core 2 receives data-ready interrupts from the acquisition subsystem (which thus also synchronizes the node), distributes fresh ADC data to the processing threads and prepares RFM read/write buffer (without actually moving them on the RFM board). Cores 3 to 6 each can host a processing Simulink algorithm, making this node fully multiprocessing. Finally core 7 is used to synchronize data transmission and reception on the RFM card; we employ a dedicated thread on a dedicated core to run this task since the node must resynchronize itself to the master RFM node's read and write clocks and the fastest way to achieve this is by polling the RFM card. Since a polling thread usually consumes all the core activity rate and can only be preempted by an higher priority thread (on a RT kernel), we chose to employ a separated core and thread to do this, leaving core 2 fully committed to real time ADC data handling.

Figure 3 presents the typical working time chart of node 7. The ADCs on the ACQ196 board are clocked at 256 kHz synchronous with TCV main clock at 1 MHz; the synchronous sampling clock is generated by a high accuracy DDS (Direct Digital Synthesizer) chip on a companion board in the acquisition cPCI crate. ADCs' data stream is directed to a double buffer on the acquisition board, when one buffer is full, a data ready interrupt is sent to the host PC and the buffer is switched, releasing the just filled one to the host CPU. The arrival rate of this interrupt is evaluated as follows:

$$f_{int} = \frac{f_s}{N_{sch}} \tag{1}$$

where f_{int} is the interrupt frequency, f_s the sampling frequency and N_{sch} the number of samples per channel in the buffer. With typical working parameters: $f_s = 256000$ and $N_{sch} = 256$ the interrupt frequency is $f_{int} = 1000$, i.e. we have a 1 ms cycle time. With this hardware, N_{sch} must be a power of two, hence the choice of 256 kHz for the sampling frequency to obtain a 1 ms cycle time.

The hardware interface code exploits Intel TSC (Time Stamp Counter [9]) technology to precisely time stamp its activities. Every core has a 64 bit wide counter clocked at the CPU nominal frequency that can be read very quickly using dedicated assembler code. These counters can be exploited to time stamp key passages of the hardware interface code, providing the timing information for system monitoring and algorithm benchmarking. The initial timing information is the measured system cycle time, i.e. time interval t1 in figure 3. The time trace of this value is the first indicator of good or improper RT behavior of the system; with the working parameters introduced in the previous paragraph, this trace should be a constant value of 1 ms. Once the data ready interrupt has been issued, the CPU must copy fresh ADC data from the acquisition board to main memory, distribute it to the processing cores and update the RFM buffers. This requires time that is stored in time stamp t2 and is depicted as the purple phase in figure 3. After this phase the processing cores have new data to process and thus processing algorithms can be triggered; several processing cores act on ADC data in parallel as depicted by the red phases in figure 3, their processing time are stored in markers t5 and t6. Up to four processing cores can be used: the algorithm presented in this paper uses two. The available computational time to all processing cores in every cycle is equal to t1 - t2. The hardware interface code can allow some processing cores to work at a lower speed w.r.t. the main ADC data ready by triggering at an integer fraction of the rate of data ready interrupt arrival. Following the t2 phase, the hardware interface code exchanges data with the reflective memory board. It must first synchronize itself with the read and write phases dictated by the RFM master node. Time elapsed from the data ready interrupt and the read and write synch time are recorded in time stamps t3 and t4 of figure 3.

All timing information are stored to the SCD MDSplus tree and are used to check the node functionality.

B. Software, the Simulink model and integration of the MHD analysis code

As all the other nodes of the control system, the control code (or diagnostic processing code, in this case) is written as a Simulink block diagram. This enormously facilitates algorithms' management during both experimental activity and later data analysis. Since node 7 is a multicore processing machine, a dedicated Simulink block model was developed. This model is shown in figure 4. Every processing core is modeled as Simulink block with standardized input and output ports; since node 7 has 4 parallel processing cores, four of these blocks are inserted. Each block represents an independent processing thread that can process input data in

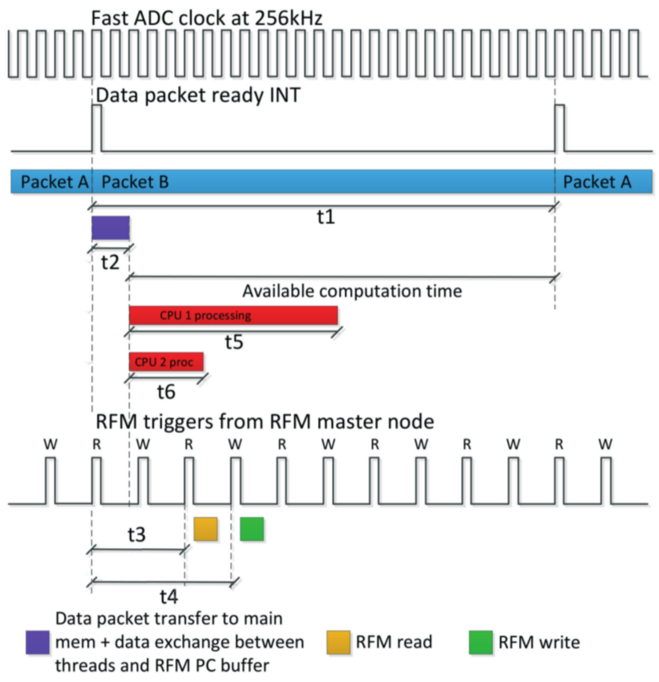
Fast ADC clock at 256kHz
Data packet ready INT
Packet A
Packet B
Packet A
t1
t2
Available computation time
CPU 1 processing
t5
CPU 2 proc
t6
RFM triggers from RFM master node
W R W R W R W R W R W R W
t3
t4
Data packet transfer to main mem + data exchange between threads and RFM PC buffer
RFM read
RFM write

Fig. 3. Time chart of node 7 cycle.

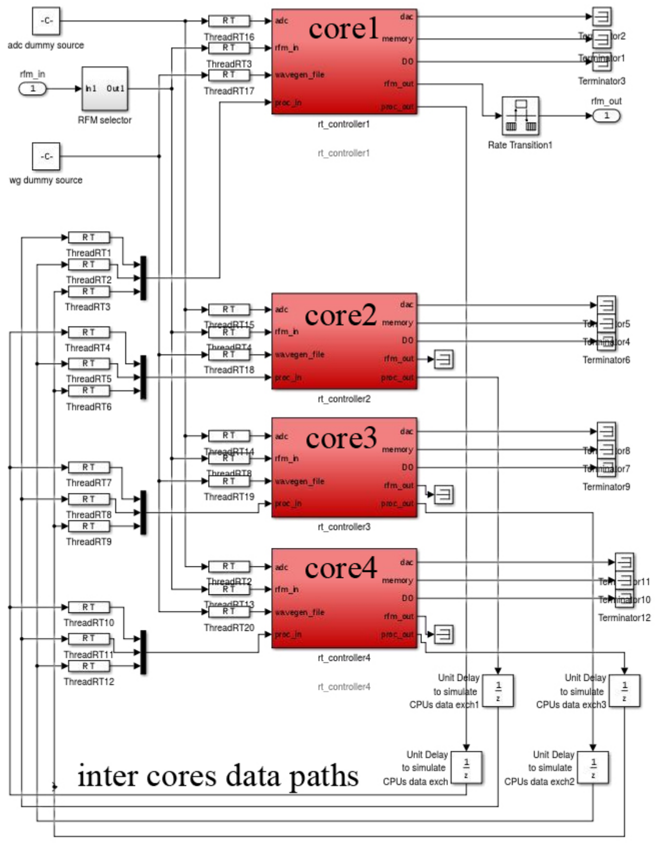

Fig. 4. Top Simulink diagram of node 7.

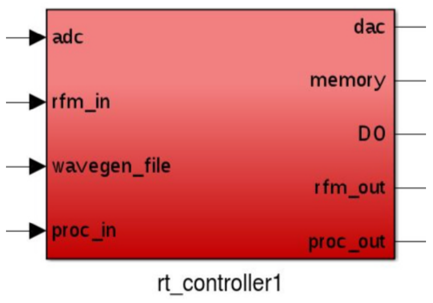

Fig. 5. Simulink diagram of one of node 7 processing cores.

parallel w.r.t. the others.

Once one TCV pulse is initiated, the active blocks' content is opened by the SCD build process, a Linux shared object library is generated for each and these libraries are distributed to node 7. Upon entering the real-time phase of the discharge, the hardware interface code of node 7 is launched and loads these shared objects assigning their processing code to one thread per core. During the discharge, these cores process data from the fast magnetic signals in parallel and return the results via the RFM real time network, as described in III-A.

Figure 5 shows the Simulink model of a single core. This is the Simulink object that is then translated into C code and then compiled into the Linux shared library upon shot preparation. It has four input and five output ports whose meaning and sizes are summarized in table III. This block acts as a wrapper for every processing algorithm, a Simulink model inserted and it also provides a clear and straightforward mean of standardizing the input and output interfaces.

Figure 4 shows the actual interconnections of the core models on the node CPU. Basically every core's proc_out port is looped back to the proc_in port of the other cores. This provides a simple and general modeling of inter-thread data communication. The hardware interface code takes care of these data movements once the algorithms models are uploaded to node 7. Obviously, this is not the most efficient way to handle inter core data communications since a fixed amount of data (figures are in table III) is always exchanged irrespective of what is really present on the proc_out ports. Nevertheless, we chose a general, fixed interface, approach at the expense of communication overhead. This is also true for all the input and output ports.

Input and output ports to and from the CPU are arranged with the twofold aim of mimicking the real system (particularly the hardware interface code) allowing SCD re-simulation on a past data in Simulink. As an example, the ADC input port is connected to a InputFromWorkspace block that is initialized with previous shot ADC data, taken from the SCD MDSplus database if the model is used for re-simulation. It is connected to the real ADCs by the hardware interface code when the core model is uploaded to node 7. The same happens for the other ports, allowing an efficient and easily manageable re-simulation of past shots whilst providing a very flexible processing algorithm development environment.

Node 7 was programmed and tested with the SVD rt MHD analysis code described in [10]. In essence this code employs a

TABLE III
INPUT AND OUTPUT PORT MEANING OF THE EMPLOYED SIMULINK MODEL OF 1 PROCESSING CORE

Port name	Function	Size
adc	data buffer from the ADCs, it contains the last acquired data packet.	16384 words (16 bits)
rfm_in	RFM buffer from other nodes, it contains the last data sent of the RFM network by other nodes.	1024 singles (32 bits)
wave-gen_file	It can be used to provide pre computed signals in alternative of putting them into the Simulink model.	1024 singles (32 bits)
proc_in	Inter-cores data communication input, it provides the mean of receiving data from other cores of the CPU.	1536 singles (32 bits)
dac	DAC output port, it contains data to be sent to node's DACs, if present.	96 words (16 bits)
memory	Real-time storage memories. Data put into this port is stored into dedicated channels during the discharge and stored to MDSplus database, i.e. it is the main way to check and debug the real-time code.	256 doubles (64 bits)
DO	Digital output port, it contains data to be sent to node's digital outputs, if present.	8 int (32 bits)
rfm_out	RFM write buffer, this is the port used to transmit data on the RFM network; data put here by the algorithms are broadcast on the RFM to other nodes.	196 singles (32 bits)
proc_out	Inter-cores data communication output, it provides the mean of transmitting data to other cores of the CPU.	512 singles (32 bits)

Singular Value Decomposition of a matrix whose columns are a bandpass filtered copy of the fast magnetic signals followed by a post-processing phase that compares the experimental principal axes (i.e. the columns of one of the three matrices computed by the SV decomposition) with those computed on a numerically generated set of signals from a theoretical model of rotating modes. This algorithm was ported to the Simulink representation introduced here. It exploits the multiprocessing capability of node 7 operating on to 2 cores: the first computes the SVD decomposition of live signals from the ADCs and the second computes the SVD decomposition of the theoretical model. Not only does this reduce the computational time but the theoretical model can be made dependable on other live plasma signals. In the presented implementation it depends upon the plasma magnetic axis in the machine. Conventional oddN and evenN MHD markers have also been included in the analysis code.

Figure 6 reports the algorithm that runs on core 1. The algorithm that runs on core 2 is omitted for brevity. This Simulink block is contained in the wrapper of figure 5. In the algorithm model, we followed the usual Simulink convention with the signals flow from left to right. So the ADC input port is placed at top-left in the figure and supplies all acquired 64 channels, each having 256 samples. This data buffer is decomposed by the pre-treatment stage whose tasks are to extract 13 channels (12 for the SVD plus 1 for one of the conventional markers), to apply the inverse sensor-adc

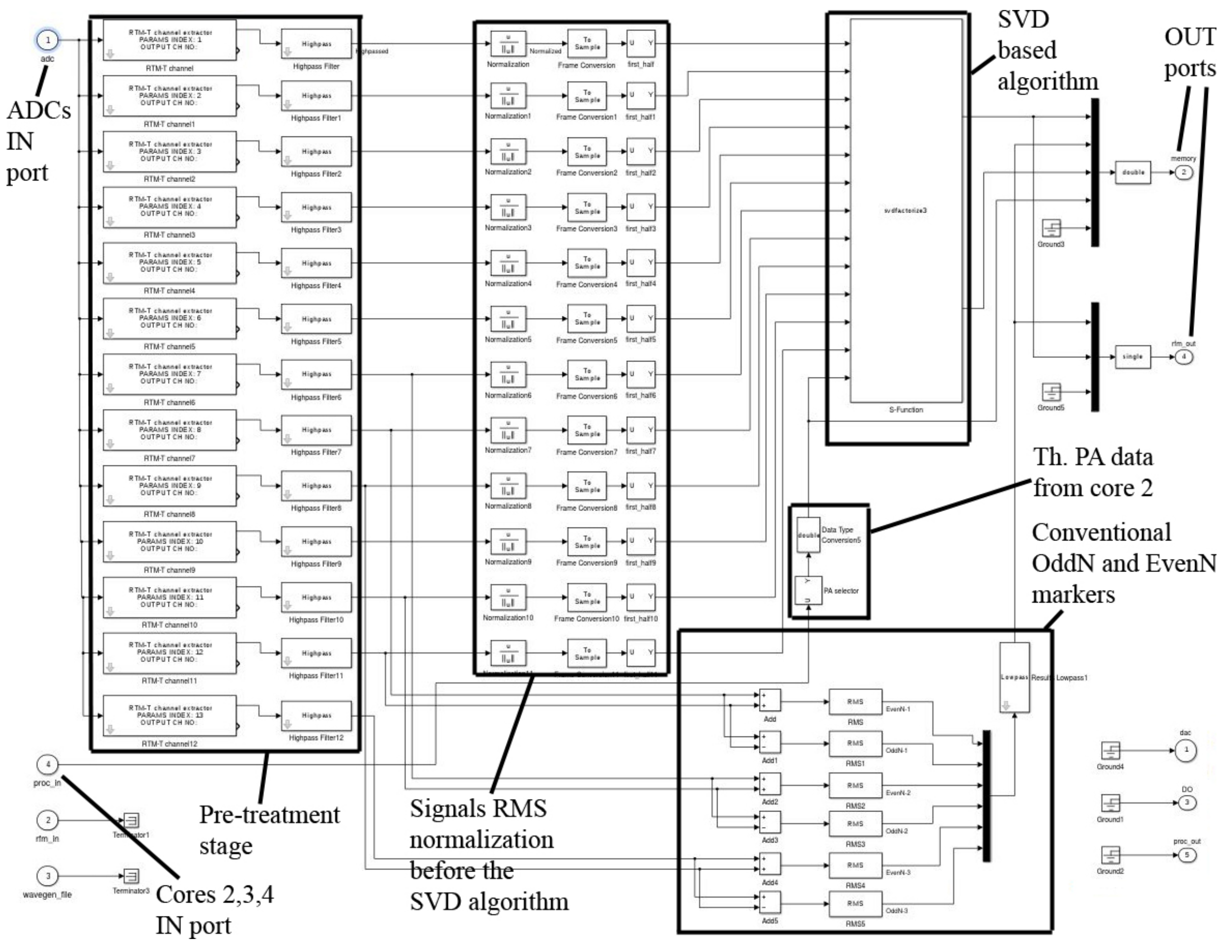

Fig. 6. Simulink diagram MHD analysis algorithm on node 7 core 1.

transfer function to compensate for the analog chain, 2x digital downsample the signals (applying a low pass filter and a 2x decimator) and to apply a high-pass filter. Pre-treated channels are prepared by the RMS normalization stage before being fed to the actual SVD factorization and post-processing algorithm (currently implemented as a Simulink S-function C++ block). The outputs of this block are passed to two output ports of the core: the RFM port to distribute the results to other nodes and the memory port to store the results on the MDSplus database for later analysis. The SVD factorization block needs the theoretical principal axes computed by the companion algorithm that runs on core 2. This data enters the block through the proc_in port at the bottom left of the figure, is extracted by a signal extraction block and connected to the SVD block.

Six pre-treated signals are employed to compute three pairs of conventional oddN and evenN markers. These signals come from three couples of fast magnetic sensors located each 180 degrees apart on the equatorial LFS plane on the machine (each couple at a different absolute toroidal angle). From their location, summing and subtracting their signals and cascading an RMS average stage (and possibly low pass filtering) provides information on the presence and amplitude of modes with even and odd toroidal mode number (N) separately. This algorithm was installed and is depicted at the bottom right of figure 6.

IV. Tests on the TCV plasma

Node 7 has been tested on real TCV plasmas during the most recent campaign with the MHD analysis code described in section III-B. The node performed well both in terms of real time performance and effectiveness in providing on line MHD activity markers. Figure 7 shows real time traces measured by the hardware interface code on node 7 during a discharge. The abscissa axis is shot time, starts at t=-0.5 s and ends at t=2.0 s (breakdown is intended at t=0.0s). Timings are arranged by the MDSplus archiving routines of node 7 in order to facilitate algorithms' computational time compliance check by the system operator. In particular, timings t2, t3, t4 are plotted "'upside-down"' from the cycle time (t1) track whereas cores' computational time are plotted normally. In this way it is simple to check when a core isn't able to keep-up with the RT requirements since the available computational time is quickly identified (see figure 7 on the right).

The node was parametrized with a 1 ms nominal cycle time and the measured one exactly matches with only tens of μs of jitter (t1 in figure 7). With two active processing core data distribution takes approximately 80 μs (t2 interval, i.e. difference between black and green plots in figure 7). The two processing cores execute the two algorithms in parallel

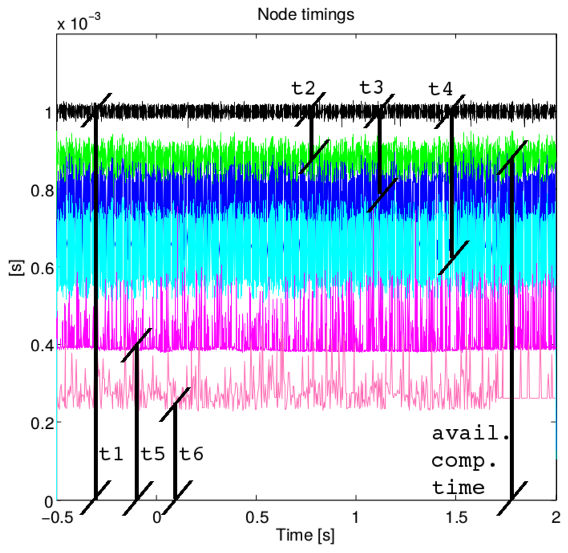

Fig. 7. Timing traces of node 7 during TCV shot no. 51995.

and respect the limit on available computational time, as depicted by the magenta (t5) and pink (t6) tracks in the figure. Finally reflective memory read and write synch timestamps are recorded (t3 and t4 intervals in the figure), proving that the 10kHz RFM strobe synch from RFM master node (node 2 in this shot) was correctly captured and data was correctly distributed to and from the real time network.

Figure 8 shows some MHD analysis results. The abscissa axis is identical to figure 7.

The first plot shows node timings as in figure 7. The second plot shows raw ADC channels of magnetic signals from TCV. The third plot shows SVD based NTM presence markers, as described in [10], the fourth graph the frequency of the rotating mode, if any, the fifth the (2,1), (3,1) and (3,2) modes likelihood markers as described in [10]. The last plot shows the standard MHD markers oddN and evenN. From the signals, a clear mode onset occours at t=1.3s which possibly leads to the disruption at t=1.4s.

Once these signals are distributed on the real time network, they could be used to trigger countermeasures to control the MHD activity (for instance by means of ECR heating) or, more simply, to soft land the discharge without incurring a full high current disruption.

V. Conclusion and Outlook

We have described the integration of a new node into the TCV real time control system. The new node is tailored to performing advanced analysis algorithm on the fast magnetic signals of TCV in real time and to distribute the results to the real time network of the control system. During the development of the system we succeeded in respecting all useful features of the legacy system, namely: a strict separation between the environment in which algorithms are developed (Mathworks/Simulink) and that in which they are executed (custom C/C++ code) and compatibility with the legacy data interface on the RFM network, etc. New features have also been introduced: a real-time packet acquisition and processing approach, multi-core data processing and data transfers on a multi synchronous system (ADC side and RFM side).

We think that the approach followed in this paper can be continued in the future whenever there is the need for the real-time processing of fast diagnostics with complex algorithms and to distribute results to other actors in the control system. One example on our system is the soft-x XTOMO acquisition node no. 5, which could be refurbished with an approach like that described here.

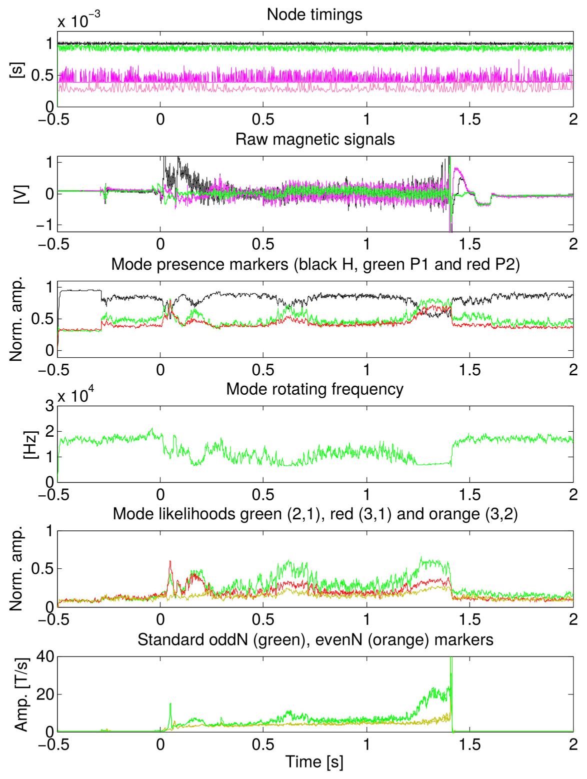

Fig. 8. MHD analysis results during TCV shot no. 52219.

Acknowledgment

The authors are grateful to the TCV team, which provided great help in commissioning the described system.

References

[1] Nardone C, Plasma Phys. Control. Fusion 1992, 34 1447
[2] Dudok de Wit T, Phys. Plasmas 1994, 1 3288
[3] Schittenheim M, Nucl. Fusion 1997, 37 1255-70
[4] Kim J S, Plasma Phys. Control. Fusion 1999, 41 1399-420
[5] Paley, J. I., 17th IEEE-NPSS 2010, pp. 1-6
[6] Felici F, Fusion Eng. Design 2014, 89 165-176
[7] Le H B, Fusion Eng. Design 2014, 89 155-164
[8] Galperti C, Fusion Eng. Design 2014, 89 214-223
[9] https://en.wikipedia.org/wiki/Time_Stamp_Counter
[10] Galperti C, Plasma Phys. Control. Fusion 2014, 56
[11] http://www.d-tacq.com/acq196cpci.shtml
[12] http://www.ge-ip.com/reflectivememorynetworks
[13] http://www.mathworks.com/products/simulink/
[14] http://www.mathworks.com/products/embedded-coder/