Open Source Real-Time Operating Systems for Plasma Control at FTU

C. Centioli, F. Iannone, G. Mazza, M. Panella, L. Pangione, V. Vitale, and L. Zaccarian

***Abstract*—Nowadays, every tokamak has a plasma control system to monitor and drive discharge parameters like position, density, current, and shape. Several different solutions have been adopted to cope with the real-time constraints, ranging from the shared memory to the transputers technologies. At present, a VME/PPC604r embedded controller running a LynxOS operating system is used on Frascati Tokamak Upgrade (FTU) for the plasma position control, with a response time of about 150 μs. In this paper, a report of the tests carried out using the open source Linux real-time operating system RTAI on a VME/INTELx86 embedded controller will be presented, in order to evaluate the replacement of the existing expensive commercial solution for the plasma control system. To this aim, the plasma position control software has been ported on the open source platform. Moreover, the performance of the two real-time operating systems will be compared, both at system level and at application level. The comparison will show that the open source real-time operating system is a good alternative to the commercial product and, because of its hard real-time features, it is a well-guaranteed platform for our plasma control system software.**

***Index Terms*—Open source systems, plasma control, real time systems.**

I. Introduction

Frascati Tokamak Upgrade (FTU) is a compact, high magnetic field tokamak, aimed at the generation of plasmas relevant for fusion experiments [1]. The standing goals of the device are studying plasma transport, plasma–wall interactions, plasma heating, and current drive in the presence of strong additional RF heating, and studying the plasma profile by means of pellet injection. The plasma discharge lasts about 1.5 s, and in standard operating mode an experiment is ran every 20–25 min.

The plasma discharge startup, position, current intensity and particle density are regulated by a real-time Plasma Control System (PCS), guaranteeing feedback actions to deal with perturbations introduced by external events like pellet injections or radiofrequency power heating. The position and current control is performed actuating four sets of coils which regulate the parameters describing the plasma position and elongation, while the density control is enforced through fast valves and real-time density detection. The control algorithm is essentially based on a mathematical model of the interactions between plasma and control coils [2], and on a proportional, integral and derivative (PID) controller [3].

Manuscript received June 9, 2003; revised January 19, 2004.

C. Centioli, F. Iannone, G. Mazza, M. Panella, and V. Vitale are with ENEA-EURATOM Association for Fusion Research, CP 65, 00044 Rome, Italy (e-mail: centioli@frascati.enea.it).

L. Pangione and L. Zaccarian are with Dipartimento di Informatica, Sistemi e Produzione, Universitá di Roma, 1-00133 Rome, Italy.

Digital Object Identifier 10.1109/TNS.2004.828789

As in any other large, and thus, long-lasting experiment, the FTU plasma control system has evolved during its lifetime into the current release, based on a single VME processor running a LynxOS real-time operating system which implements the controller code. The real-time duty cycle is fast (0.5 ms) and the system as a whole offers, beside a satisfactory regulation of the physical quantities, very good performance and a high degree of reliability.

Nevertheless, an alternative solution to the current system has been examined, for the following reasons:

a) the considerably high cost of the system (about 45 000 euro, including the backup system) makes it expensive to deal with software maintenance and spare provisioning; moreover, it prevents it from being considered for further applications, for example in the diagnostics area, where an interest for real-time controls is growing;

b) because of its proprietary operating system, the integration with the FTU open source environment and commodity hardware is difficult and requires a great deal of not portable *ad hoc* coding;

c) new time consuming enhancements of the PCS, such as plasma shaping [4] and radiofrequency coupling, will have to be implemented shortly.

The aim of this paper is to show that even in a complex environment, a control system with hard real-time requirements can be realized with a suitable open source real-time operating system such as RTAI-Linux.

The results obtained have been outstanding in terms of performance and costs, without any major intervention on the existing software architecture.

In the following, a short description of the current system will be provided; then the open source alternatives will be examined as well as the solutions adopted. An account of the porting of the application will be given, together with the results obtained.

II. FTU Plasma Control System: System Architecture and Current Status

The FTU feedback control system performs two real-time activities: the plasma position and current control and the gas density regulation [5].

The actual real-time activity lasts 5 s and may be roughly divided into two phases: in the first phase—from 2 s to 1 ms before the plasma discharge—measurement offsets are calculated and preprogrammed references are emitted to the actuators, while in the second phase—from 1 ms before to 3 s after the plasma discharge—the proper control activity is performed.

The operating context of the FTU real-time system is sketched in Fig. 1. The software application consists of two

TABLE I
FTU PCS CONTROLLERS

	CES RIO2 8062	Eltec Eurocom 138
CPU Type	PPC 604r @ 300 MHz	Celeron Mendocino @ 433 MHz
Global Memory	64 up 128 Mbytes EDO DRAM	128 MByte RAM
L2 Cache	1 MByte fast SSRAM	128 KByte
Ethernet	10BaseT	10BaseT
External HD (SCSI)	4 GBytes	10 GBytes
Operating System	LynxOS 3.0.1	Red Hat 8.0 Linux kernel 2.4.18

The main characteristics of the VME CPUs used in the FTU feedback control system (respectively, in the old system and in the new system) are compared

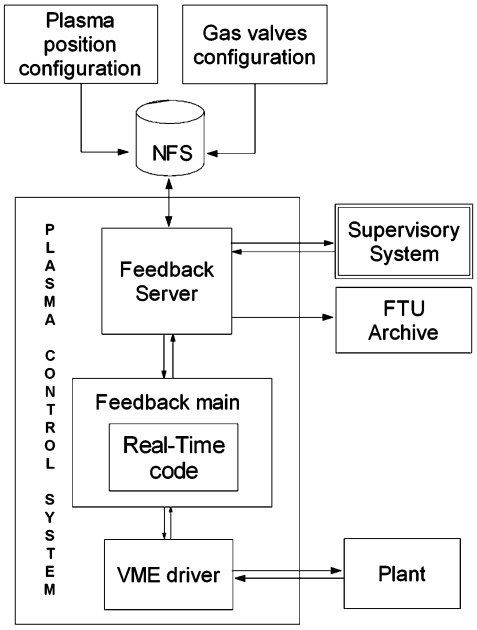

Fig. 1. The FTU PCS block diagram and operative context. Its two components and their interactions with the FTU control and data acquisition environment are shown.

components, running on a single controller. The Feedback Server is devoted to the communication with the FTU supervisory system, to data acquisition, control parameters collection and data storage on the FTU archive. The multithreaded component, Feedback Main, is aimed at hardware control and real-time functions. Inter-process communications are developed under POSIX standard and make use of threads and message queues.

TCP/IP messages are exchanged between the FTU supervisory system and the Feedback Server in order to synchronize the latter during the non real-time phases of the shot evolution (i.e., before and after the plasma discharge); on the other hand, during the plasma discharge, the PCS uses exclusively the Central Timing System (CTS) signals to schedule its activities.

All the code is written in C/C++ language. In the release used so far, the code was optimized so that the complex control algorithm took $\sim 150\ \mu$s to perform both the real-time control calculations (namely, the gas density regulation and the position and plasma current control).

At the end of each experiment ("*shot*"), all the work variables (i.e., inputs, states and outputs) are transferred to the FTU shot archive through the standard remote procedure call (RPC) protocol and they can be analyzed and elaborated by the physicists with the graphical software tools available in the FTU software environment.

The pre-programmed references and other working parameters originating from the FTU control system are transferred to the PCS through the Network File System (NFS), which allows the remote file sharing among different platforms.

A Java GUI Application running on MacOS X is used to set, via NFS, the pre-programmed references and the PID coefficients.

The PCS hardware components consist of the following.

- A VME CPU controller board. Before the system was upgraded, this was a PowerPC based CES RIO2 8062 VME board [6] running a LynxOS™ [7] operating system. Table I gives an account of the RIO2 8062 technical specifications.
- Three custom-designed VME fast ADCs (CAEN V652B [8]) modules that take about 30 μs to acquire 40 input signals.
- Two Acromag [9] VME DACs to emit 11 output signals (four references toward the coils, six on/off signals toward the fast valves, and one signal indicating the plasma presence and used by other plants).
- A VME timing module used, through its two digital inputs, to catch the synchronizing signals coming from CTS and to give the internal working time to the feedback CPU.

The whole system is clocked at 2 kHz.

III. The New Plasma Control System

The cornerstone of the new FTU PCS is the real-time open source Linux RTAI operating system. This choice has been made after a thorough analysis of experiences in other fusion laboratories, particularly those using plasma control systems comparable to the one at FTU, but still taking into account the relative "uniqueness" of controls in fusion experiments.

In the present paper, the key issue has been not to change the software environment and the system architecture, mainly with regard to the hardware and to the interfaces between the PCS and the FTU supervisory system described in Section II. As a consequence, among the possible solutions, the ones which could comply with the existing Unix-like environment have been obviously privileged [10], [11]. Yet, a closer examination has shown

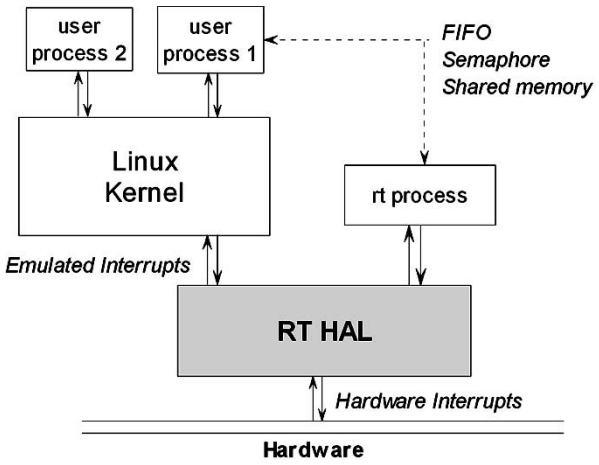

Fig. 2. Relations between user processes, real-time processes, and Linux kernel in the hardware abstraction layer framework.

that these two experiences could not be exported directly to the FTU plasma control system, because the first proposed the use of another expensive commercial real-time operating system, while the second achieved real-time capabilities from Linux by customization. Finally it has been decided to investigate the real-time Linux versions publicly available.

A. Real-Time Operating System Considerations

There are two alternative ways of obtaining real-time services from Linux: preemption improvement and the insertion of an hardware abstraction layer [12]. With the first technique, the kernel activities can be interrupted whenever possible in order to privilege processes with higher priority. Segmentation and the minimum path length are evaluated on a statistical basis. Operating systems using this approach are TimeSys and Love [13]. The second approach seemed more promising, because it leaves the Linux kernel basically unchanged. This is due to the insertion of a layer, called real-time abstraction layer (RTHAL) between the Linux kernel and the hardware (see Fig. 2). The purpose of the layer is to intercept hardware interrupts and system calls which would be not allowed in a real-time environment, and handle them in the correct way. This, in our opinion, makes the system more reliable; moreover, it preserves a high degree of independence between the Linux kernel, the real-time implementation layer and the application level. On the other hand, this approach implies a more difficult process implementation.

In this context, a selection has been made between RTLinux and RTAI [14], [15]. These two solutions are largely similar and both distributed freely, but RTAI has been privileged because it presents several advantages, both from the system level and from the application level perspective. In particular:

- it "officially" supports the Linux kernel 2.4.18, contained in the distribution;
- it is being developed by a large community mainly belonging to public [1] institutions; this guarantees that, for the time being, no commercial versions with different added features will be produced;
- the patch to be installed is less pervasive, and does not undermine the robustness of the operating system;
- it is under LGPL licence;
- it provides more hard real-time features;
- it provides extended POSIX compliance (1003.1c: POSIX queues and 1003.1b: POSIX threads);
- it supports hard real-time for nonroot users (LXRT), thus simplifying the application porting and development;
- it allows dynamic memory allocation.

Moreover, a recent study [11] has shown that RTLinux worst case latency times are too high for plasma control systems requirements.

B. Hardware Selection

In order to preserve the previous investments, only the CPU CES RIO2 board has been replaced, leaving the VME hardware modules unchanged. Due to previous experiences in a data acquisition (and thus non real-time) environment, an Eltec Eurocom®138 CPU board [16] has been selected, equipped with a Celeron processor @433 MHz and 128 MB RAM—see Table I for the technical specifications. The VME interface is implemented by the Tundra Universe II chipset [17]. The underlying operating system is Linux Red Hat 8.0 [18] with Linux kernel 2.4.18.

C. The VME Real-Time Driver

A Linux VME driver is a kernel module devoted to map VME modules onto a set of memory addresses, and to give the user an interface to access it. Normally, the interface is composed of a set of functions accessing device files like /dev/vme_m0. This is surely not desirable in a real-time environment.

After a thorough analysis, it has been found that no proper "real-time" Linux VME driver was available, thus to get real-time compatible behavior from an existing driver, the existing one had to be customized.

As a starting point the nonreal-time VMELinux driver v.1.3.0 has been adopted, because of the fact that:

- it has been developed for a Tundra Universe II VME-PCI bridge;
- it contains only function calls that are fully compatible with the RTAI environment, namely it does not use "forbidden" statements like *(k)malloc*, *calloc* or *sti—cli*;
- it can be customized to support up to eight VME modules, which is particularly desirable for FTU feedback signal processing unit, consisting of six VME boards.

The customized driver includes a set of dedicated functions to access the hardware. These functions (such as, for example, write_DAC and read_ADC) grant access to a given module under predetermined conditions. There is obviously a loss in generality and portability by adopting this solution, but the price to be paid is worth the advantage arising from the fact that the code accessing the driver is optimized to comply with the control time requirements and can be easily written.

The average access time to the VME is $\sim 1\ \mu$s, (0.9 μs with direct pointer access, 1.2 μs through *readw()*) and it has been mea-

[1]The RTAI project is being developed and maintained since 1998 by Paolo Mantegazza and his team at the "Dipartimento di Ingegneria Aerospaziale—Politecnico di Milano" (DIAPM), Italy.

sured via software entirely at module level; in this way the latencies due to context switching have not been taken into account.

D. Application Porting Issues

To suit the FTU feedback system to the new environment, two alternative approaches were possible: porting the code to native RTAI architecture or using an advanced RTAI feature instead: LXRT-extended. With the first approach (see Figs. 1 and 2) the hard real-time section of the feedback main would have been disentangled and loaded as a kernel module. Moreover, shared memory, semaphores, and FIFOs would have been necessary for the information exchange between feedback main and the VME driver; but the huge amount of variables to be shared made this solution not viable.

The second solution seemed more promising, because it provides the RTAI API to the user space, and LXRT-extended allows hard real-time activities from the user space environment. In a few words, the application runs in the user space while the real-time functions are dealt with by the real-time Linux scheduler. The time switching between Linux and the real-time scheduler is negligible even for outdated processors like Pentium II.

The main advantage in using LXRT-Extended is that the hard real-time section of the code does not need to be extracted from the Feedback Main. Therefore complicated structures for interprocess communication are not necessary. Furthermore, the functions of the VME driver, extending the RTAI real-time functions, are used as a real-time library. This solution also grants definite advantages in debugging real-time applications; in fact, the debug can be carried out running the process in user space without the real-time scheduler, that can be reinserted after the validation of the results and the removal of the non real-time I/O calls. Moreover, applications are generally kernel-independent, and may be easily ported in binary form to machines with different kernel versions. This last statement does not apply to the present application, because the VME driver has an obvious kernel dependency. Any user can run real-time code without needing a superuser profile, thus preventing possible damages to the system. On the other hand, since the code can be debugged and ran in user space, typical non real-time functions (e.g., non-real-time I/O activities) could be erroneously inserted or left in the real-time section of the code without any notification at compilation and run time, thus causing a loss of determinism in the process. This requires extra care on the programmer's side.

During the porting, an acquisition of the average jitter time has been made using the DAC access routine included in the feedback application. The result is an average of 20 μs, practically negligible in this application.

On the whole, the porting has been quite straightforward, with the only exception for the unexpected misbehavior of the message library—founded on POSIX message queues—that required a few software interventions on the communication layer between the Feedback Server and the Feedback Main, while the main program and the related functions are basically unchanged.

IV. Results

The new feedback has been first tested in passive mode during FTU experiments. The results obtained during the first runs have shown that the porting did not affect very much the overall performance—even if the real-time cycle duration during the control regime may last 200 μs, thus longer than the corresponding cycle ran by the previous control system but still within the 500 μs limit imposed by the system clock. At the same time, the performances are improved in the noncontrolling phase of

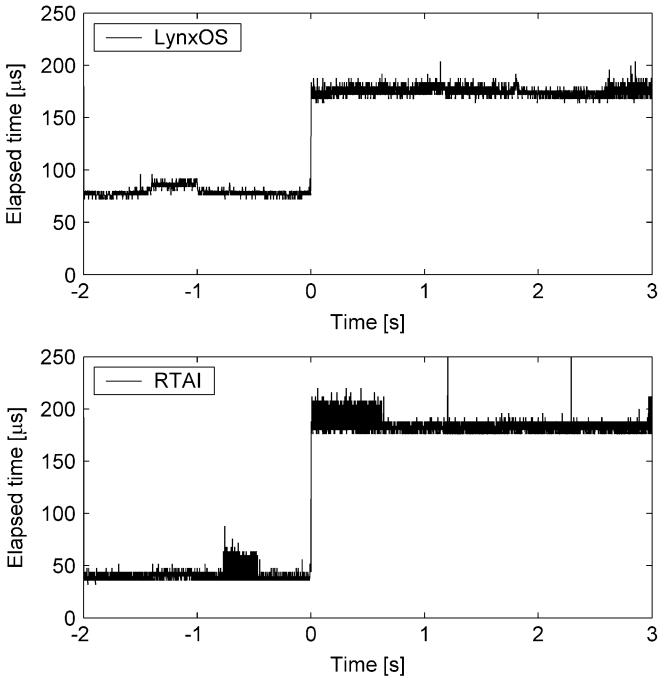

Fig. 3. Shot #23 812. Comparison of the measurements of the acquired real-time duty cycle duration of the new system based on RTAI (bottom) to the previous system (top) before the optimization.

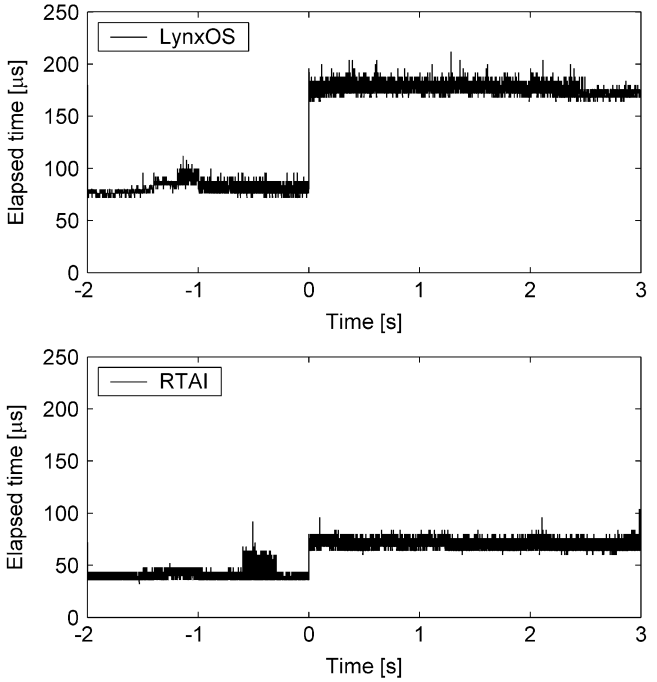

Fig. 4. Shot #24 057. Comparison of the measurements of the acquired real-time duty cycle duration of the new system based on RTAI (bottom) to the previous system (top), after the optimization. The performance is dramatically improved, especially in the control regime.

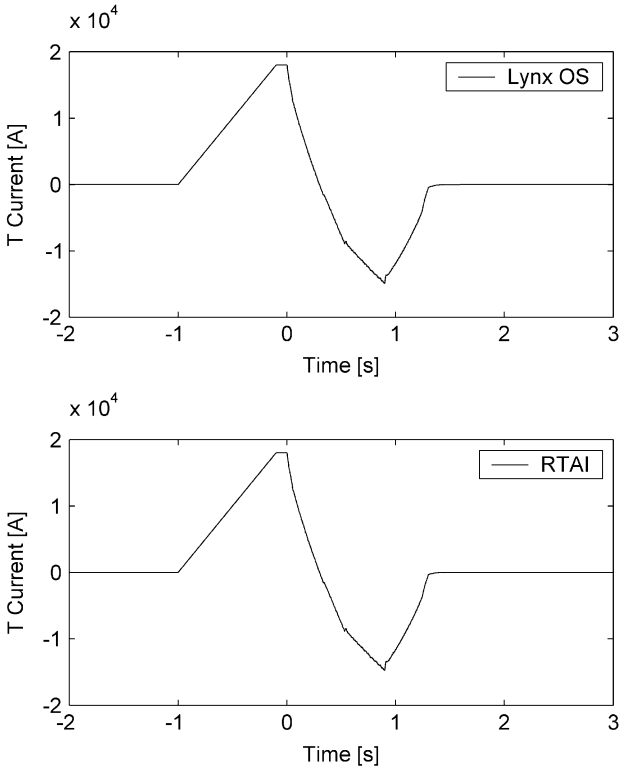

Fig. 5. T-coil current reference signal produced respectively by the previous feedback system (top) and by the new system (bottom). The measurements shown have been taken from shot #24 057.

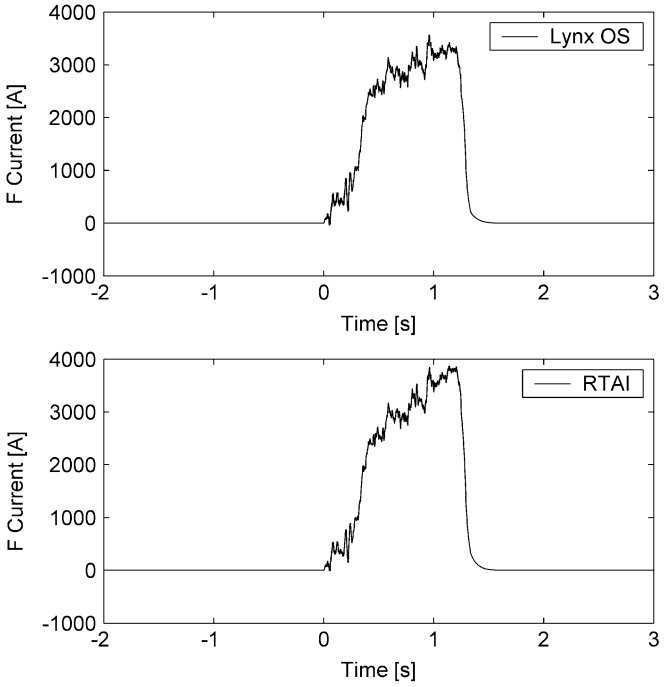

Fig. 6. F-coil current reference signal produced respectively by the previous feedback system (top) and the by new system (bottom). The measurements shown have been taken from shot #24 057.

the real-time activity (Fig. 3). Although this behavior could be perfectly acceptable for the current plasma control system, an optimization has been done on the code, to achieve better performance and, consequently, to comply with the long term objectives of this work. To this aim, a more "real-time" programming technique has been adopted, replacing indexes with pointers in array access in the real-time VME driver. As a result, the performance of the new system has been greatly improved, reducing the real-time control cycle duration to an average value of 75 μs (Fig. 4), to be compared to the average cycle duration of the previous system (150 μs).

Data acquired during experimental sessions show that the model is being ran smoothly and produces results compatible with the existing feedback system (Figs. 5 and 6). As a follow up to the passive mode results aforementioned, active sessions on FTU machine to test active control on plasma position and gas injection have been successfully performed.

V. Conclusion

The new open source plasma control system has proven to be more performant than the previous one, based on the old commercial LynxOS operating system, and at least as reliable. The migration from LynxOS to a real-time Linux solution has been quite harmless, thanks to the abstraction layers provided by RTAI and LXRT-Extended. The experimental tests have shown that to obtain improved performance on the new platform, as compared to the old one, only minor changes were necessary (and easily implemented) on the existing control code. On the whole, the results are excellent and very promising for medium term developments: in fact, this experience tracks the way to new enhancements of the FTU feedback control, like plasma shaping and the experimentation of novel techniques for maximum coupling between plasma and RF power. Furthermore, arrangements have been taken to export this experience to other fusion devices like Joint European Torus (JET), with cost-effective solutions in the framework of open source software environments.

Acknowledgment

The authors wish to thank the session leader S. Podda and the FTU operators team for their invaluable assistance during the experimental sessions and G. Mazzitelli for his encouragement during this work.

References

[1] R. Andreani *et al.*, "The frascati tokamak upgrade: Electrical, mechanical, and thermal features," in *Fusion Technol. 1990*, B. E. Keen, H. Huguet, and R. Hemsworth, Eds. New York: Elsevier, 1991, p. 218.
[2] F. Crisanti and M. Santinelli, "Active plasma position and current feedback in the FTU tokamak machine," in *Proc. 16th Symp. Fusion Engineering*, London, U.K., Sept. 3–7, 1990.
[3] F. Crisanti, C. Neri, and M. Santinelli, "FTU plasma position and current feedback," in *Proc. 16th Symp. Fusion Engineering*, London, U.K., Sept. 3–7, 1990.
[4] V. Vitale *et al.*, "A 10 KHz feedback control system for plasma shaping on FTU," in *Proc. 11th IEEE NPSS Real-Time Conf.*, Santa Fe, NM, June 14–18, 1999.
[5] N. A. Uckan and C. Gormezano, Eds., *Fusion Science and Technology*, May 2004. Special Issue—FTU (to be published).
[6] CES—Creative Electronic System—RIO2 8062 Data Sheet [Online]. Available: http://www.ces.ch
[7] LynxOS [Online]. Available: http://www.lynuxworks.com
[8] CAEN—Costruzioni Apparecchiature Elettroniche Nucleari v652B Data Sheet [Online]. Available: http://www.caen.it
[9] Acromag [Online]. Available: http://www.acromag.com

[10] M. Lenholm *et al.*, "Plasma control at JET," *Fusion Eng. Design*, vol. 48, pp. 37–45, 2000.
[11] B. G. Penaflor, J. R. Ferron, M. L. Walker, D. A. Piglowski, and R. D. Johnson, "Real-time control of DIII-D plasma discharges using a linux alpha computing cluster," *Fusion Eng. Design*, vol. 56–57, pp. 739–742, Oct. 2001.
[12] T. Bird. (2000, Nov.) Two approaches to real time services in Linux. *RTC Mag.* [Online]. Available: http://www.linuxdevices.com/articles/AT7 005 360 270.html
[13] K. Dankwardt, "Comparing real time Linux alternatives," *Linux Devices*, Oct. 2000.
[14] P. Mantegazza *et al.*, "RTAI Programming Guide,", http://www.aero.polimi.it/~rtai/documentation/index.html, Sept. 2000. Available.
[15] J. I. Ripoll. (2002) RTLinux Versus RTAI. [Online]. Available: http://bernia.disca.upv.es/rtportal/comparative/rtl_vs_rtai.html
[16] ELTEC—Elektronik Mainz—Eurocom 130 Data Sheet [Online]. Available: http://www.eltec.de/datasheets/e138_1h.pdf
[17] Universe II VME-to-PCI Bus Bridge Manual [Online]. Available: http://www.tundra.com
[18] Red Hot Linux 8.0 Reference Guide [Online]. Available: http://www.redhat.com