Available online at www.sciencedirect.com
ScienceDirect
Fusion Engineering and Design 83 (2008) 215–219
Fusion Engineering and Design
www.elsevier.com/locate/fusengdes

The integration of the new advanced digital plasma control system in TCV

N. Cruz [a,*], A.P. Rodrigues [a], B. Santos [a], C.A.F. Varandas [a], B.P. Duval [b], J.-M. Moret [b], J. Berrino [b], Y. Martin [b], X. Llobet [b]

[a] *CFN-IST, Centro de Fusão Nuclear, Associação Euratom-IST, Av. Rovisco Pais, 1049-001 Lisboa, Portugal*
[b] *CRPP-EPFL, Centre de Recherches en Physique des Plasmas, Association Euratom-Confédération Suisse, École Polytechnique Fédérale de Lausanne, CH-1015 Lausanne, Switzerland*

Available online 4 January 2008

Abstract

A new advanced digital plasma control system was developed aiming at a larger flexibility and enhanced performance controlling the TCV plasma shape, position, current and density. The system is a complex grid of 32 acquisition processing and control channels (APCCs) that can go up to 25 μs for the slow control cycle and a grid of 4 APCCs that can go up to 5 μs for the fast control cycle. Each APCC is composed of analogue input, digital signal processor (DSP) and analogue output. All APCCs are interconnected broadcasting data in each cycle.

A suitable interface between the system and the TCV plant enabling the complete integration into the existing schema running and controlling the TCV was designed and implemented. The system state-machine is presented. The advantages of using a structured integrated tool such as MDSPlus is evaluated, as well as the results on the final performance, usability and stability of the system.

Keywords: MDSPlus; Plasma control; Digital signal processor; Parallel processing; System integration

1. Introduction

The new advanced digital plasma control system (APCS) is a complex grid of 32 acquisition processing and control channels (APCCs) for the slow control cycle (25 μs) and a grid of 4 APCCs for the fast control cycle (5 μs) aiming at a larger flexibility and enhanced performance controlling the TCV plasma shape, position, current and density.

Each APCC has one analogue input channel (ADC), one analogue output channel (DAC) and one digital signal processor (DSP) connected to three other APCCs, to the VME bus and capable of connecting to APCCs on other hardware modules [1].

Digital signal processors (DSPs) and field programmable gate arrays (FPGAs) have been employed in a variety of systems due to its high processing capability together with low power consumption and low price. To increase the performance and cope with distributed processing systems (like the APCS), complex parallel systems have been developed in several scientific areas [2].

Interface to such complex systems is a key issue on its usability, integration and algorithm development speed. Moreover, in fusion research experiments, an effort has been made to structure and ease the integration of new systems not only for data acquisition [3], but also to control systems [4].

A suitable interface between the APCS system and the TCV plant control software enabling the complete integration into the existing schema running and controlling the TCV was designed and implemented. This interface enables the abstraction from the hardware implementation when building the physics control algorithm.

2. Host software overview

Fig. 1 depicts the host software structure.

The host software is built on top of the Linux VME device driver [5]. The hardware access is made using a shared library built for the module (RTPro library) and a set of commands that can be called from the Linux command line (Coreutils). Then a specific shared library was built to deal with the APCS system (DPCS library) that makes calls to the RTPro library but also uses some system commands from the Coreutils. On top

* Corresponding author. Tel.: +351 239410108; fax: +351 239829158.
E-mail address: nunocruz@lei.fis.uc.pt (N. Cruz).

0920-3796/$ – see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.fusengdes.2007.11.005

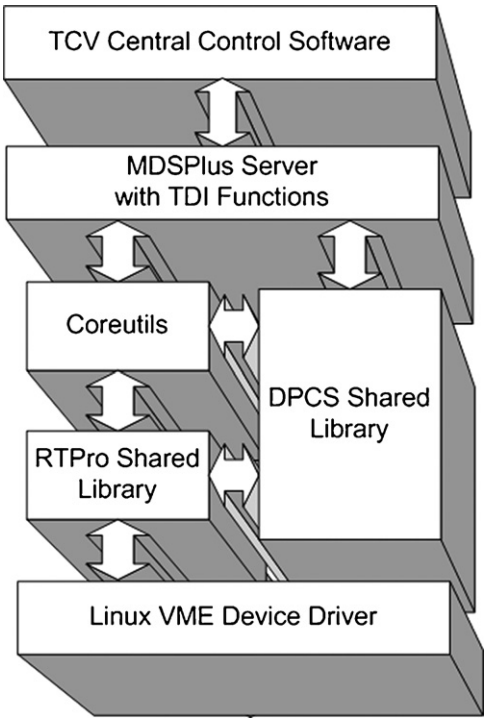

Fig. 1. The APCS software block diagram.

of these libraries a set of tree data interface (TDI) functions [6] were developed to interface the TCV central control software.

3. DSP programming interface

The DSP programming interface appears as a need to develop a simple way of programming the real-time parallel applications with an abstraction layer from the hardware. Without this tool algorithm developers and physicists would have to become experts on the hardware platform before they could take advantage of the system features. Moreover, when applications require timing constraints and real-time interaction with the environment the difficulties are even greater.

The usual approach has been that the system designers provide the user with an API and it is up to the application developers to satisfy the final requirements, checking the time constraints and the quality of service. In this case a different approach is being tried. The user provides the mathematical algorithm while the timing and data transferring requirements are handled by the DSP operating system. With this approach, application programmers can concentrate on improving the algorithms and mathematical specific code, letting the OS guarantee the best possible performance for the hardware.

One of the common difficulties in real-time systems is the scheduling of the several components. When it comes to a synchronous real-time parallel processing system with several CPUs (such as the APCS system) this problem becomes even bigger than in one CPU systems. It is very hard (impossible, most of the times, etc.) for the application programmers to coordinate the use of all hardware resources, providing the desired system performance.

The deeper system designers knowledge when building the low-level software (the OS) is an obvious advantage. Moreover, algorithm developers may be provided with a layer of abstraction from the hardware making it easier to develop the algorithms. In the future, new systems may be designed using the same philosophy for different platforms, providing code portability without spending time learning the new hardware architectures.

3.1. The DATAMOVER

The DSP OS has two main algorithms when in the control cycle:

- *DATAMOVER algorithm*: Responsible for read, format and broadcast the data to all DSPs.
- *TCV control algorithm* (*TCV-ALGO*): Responsible for data processing.

It is the independence between the data transfer (using the DATAMOVER algorithm) and the TCV control algorithm that permits the desired hardware abstraction. The TCV-ALGO developer knows that the data is available for use without worrying with how it was transferred between DSPs. Therefore all his attention may be put into the control algorithm implementation.

3.2. Integrating the TCV-ALGO in the DSP code

To easily interface the DSP OS structure a simple method was found to include the algorithm developed by the physicists in the DSP. The tools available from Texas Instruments are the C and Assembler compiler for the TMS320C3X/4X [7]. Therefore, the solution is to include predefined C and H files into the project where the algorithm can be coded.

This approach permits a higher level programming environment such as a visual language (similar to MatLab/Simulink) to be used. Using a pre-compiler the real-time programming blocks are converted into the C and H code files that are included in the DSP project [8].

The DSP OS interfaces the TCV algorithm by calling three C functions in defined time slots during the control process: (i) `void RT_init(void)`—called when the system is armed, before the first trigger; (ii) `void RT_loop(void)` —called at each real-time control cycle, triggered by the real-time loop clock; (iii) `void RT_post(void)` —called at the end of the shot, after last trigger, when the gate closes or on error as a safety procedure.

3.3. Compiling and loading the DSP software

The TCV-ALGO functions of the previous section are coded in tcvalgo.c and tcvalgo.h in the DSP project. To compile and load the DSPs some problems had to be solved:

(i) Texas Instruments uses the Code Composer Studio as the standard IDE for software development, without Linux version, and with limited scripting capabilities.

(ii) The DSP operating system source code is different for each DSP in a module. There are 36 tcvalgo.c and tcvalgo.h files to be compiled and linked, each one corresponding to a DSP0, DSP1, DSP2 or DSP3 source code that must be linked to the correct one as they are different from each other.
(iii) The upload of the DSPs code must be fast enough so that it can be made at the arm signal of each TCV shot.

The capability to compile and script under Linux OS was achieved using Wine [9], the Texas Instruments win32 shell version of the compiler and linker tools (TMS320C3x/4x Code Generation Tools) and a set of bash shell scripts developed for the system. The 36 C and H source code files are numbered in a working directory and when the `builddsp.sh` script is called they are compiled and linked against the correct DSP OS files according to its number, producing 36 DSP binary code files ready to upload to the DSPs.

The DSPs can boot from the module flash memory, however uploading the code to the module global memory and making the DSPs boot from there is faster and permits that the code stored in the flash memory is used for testing and fail safe booting. A dual-boot mode was implemented.

4. Why using MDSPlus?

In this section will be depicted the reason to use MDSPlus [6] to interface the TCV plant control software [4].

The MDSPlus programming interface contains the basic commands, simplifying data access even when using complex structures. Using the client/server model it is possible to read or write data at remote sites without file transfers.

For the current application we used the built-in expression evaluator that extends the capabilities of MDSPlus. All interfaces to MDSPlus data are based on the evaluation of an expression. These expressions are written in a language called tree data interface (TDI). External routines written in other programming languages can be invoked by calling shareable libraries, providing a huge flexibility for calculus and hardware access [10]. This capability has been widely used in this implementation of the interface.

Another important feature used was the MdsIP, a client–server with remote routine execution and automatic machine-binary data translation [10]. MdsIP permits that a client on any platform make remote routine calls in the server on another platform, receiving a reply in its own binary format. With MdsIP it is possible to run native remote routine calls without any programming effort from the developer. This approach besides making the client relatively simple (because the server performs all the computation), also permits the remote TCV operator to control the hardware in the APCS system. Because the underlying technology is TCP/IP sockets, the security can be dealt with like in any other TCP/IP service, defining user and IP address access permissions.

5. Interfacing the TCV plant: the system state-machine and data tree

To interface the TCV plant control software to the APCS system a control state-machine was developed. In this section the state-machine is presented, as well as its implementation. Fig. 2 depicts the state-machine diagram.

The following list describes the different states of the machine describing its purpose in the system:

- *Not initialized*: Initial state after a power up. This state is meant to detect undesirable system resets and power ups. The system must be initialized by the operator to go into idle state.
- *Idle*: Between pulses and when initialized after power up, the system is in an idle state waiting for a load command.
- *Loading data*: After receiving the load command the system reads the data from tree and prepares the DSPs for pulse. DSPs are reset and data is uploaded.
- *Loading DSP and data*: After receiving the load command the system reads the data and software from tree and prepares the DSPs for pulse. DSPs are reset, software and data is uploaded.
- *Loaded ready*: System is ready for pulse after an arm command.
- *Running pulse*: System has been armed and is pulsing.
- *End shot*: System can detect the end of a pulse if the hardware gate controlling the software is closed or if the number of programmed iterations is obtained. In this case, system automatically goes from running pulse to end shot state.
- *After pulse operations*: When the pulse is over the system starts collecting data from hardware modules, storing it in the local MDS tree.
- *Data ready*: After all data is collected and stored, system goes to data ready. Data can now be retrieved from local tree.
- *Error*: From any state, if a severe system internal error is detected the system goes into error state. It can also go to error if `DPCS_Abort()` is called externally. The error code can be checked using the `DPCS_Status` TDI function.

A set of TDI functions are the state-machine's stimulus. These functions can be called remotely from the central control software using MdsIP by connecting to the MDSPlus server installed in the VME crate. There are two functions (`DPCS_State` and `DPCS_Status`) to check the system state and errors. The complete list is `DPCS_Init()`, `DPCS_Ld_Data()`, `DPCS_Ld_ DSP()`, `DPCS_Arm()`, `DPCS_Abort()`, `DPCS_End_Shot()`, `DPCS_Collect_Dt()`, `DPCS_Dt_Collected()`, `DPCS_State()` and `DPCS_Status()`.

The same principle was also used to build a compilation state-machine. This state-machine reads the source code from the data tree, compiles and links against the DSP OS using the tools described in the previous section and puts the binary code back on the tree.

The system MDSPlus tree has two main branches. Because this is a control system the first branch is used to store the configuration, algorithm code and initial data to be uploaded to the

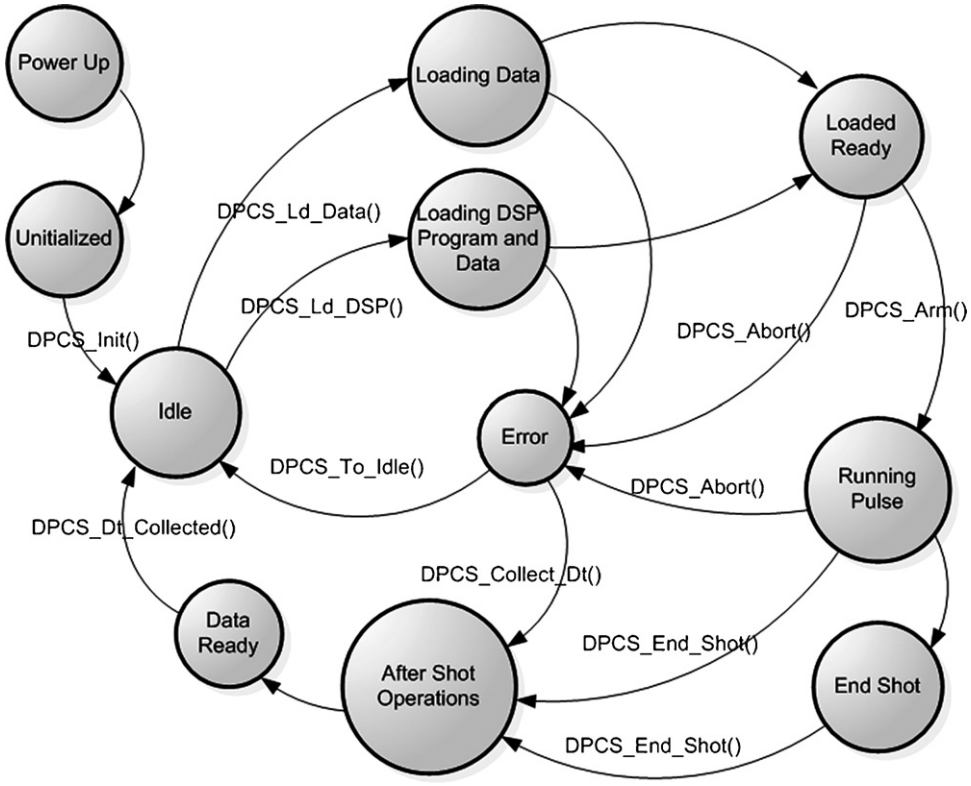

Fig. 2. The APCS interface state-machine.

DSPs before the shot—the control branch. The second branch is the data retrieved from the system after the shot. This data includes the final algorithm DSP memory space and a decimation of iterations of the control cycle's DATAMOVER data (including ADC, DAC and software data) for debugging purposes.

The data is stored in a local tree that can be uploaded with new DSP code and configuration before the shot. After the shot the debug data can be retrieved. Between the shots the tree is copied through the network using standard `rsync` over `ssh` Linux commands for backup.

6. Conclusions

After preliminary tests in Portugal, the system was installed in TCV. The DSP interface was tested using several algorithms: (i) putting some APCC input into another APCC output; (ii) controlling a digital i/o based on an APCC input level; (iii) output an IIR low-pass, high-pass or band-pass filter from the corresponding input or from other APCC input. These tests revealed that the system is appropriate for the application and performs according to the expectations when properly configured. However the lack of some scripting tools capable of automatically create the configuration for the DATAMOVER created some difficulties with a lot of time being necessary to debug small configuration errors. These problems obviate the necessity for the tools presented in this paper as well as some new ones under development.

The APCS integration into the TCV plant control software was successful. The system was controlled remotely from the control room for several shots during January and February 2007. These tests were made using the APCS system in parallel with the PID controllers. The outputs were compared to check the algorithm showing the system behaved correctly. Some problems were found and corrected using this method.

An effective TCV plasma control using the new APCS was achieved in February 2007, in shot 34313. In Ref. [11] some results of the system are presented including a signal comparison between the new APCS and the previous analogue system.

Although some work has to be finished for daily use of the system after the 2008 TCV restart, this is a promising system with the appropriate development and control tools to enhance plasma control in TCV. The same philosophy should be implemented in the new hardware that is under development using the advanced telecommunications computing architecture (ATCA) standard.

Acknowledgements

This work has been carried out in the frame of the Contract of Association between the European Atomic Energy Community and Instituto Superior Técnico (IST) and of the Contract of Associated Laboratory between Fundação para a Ciência e Tecnologia (FCT) and IST. Financial support was also received from the Association EURATOM/Swiss Confederation.

The content of the publication is the sole responsibility of the authors and it does not necessarily represent the views of the Commission of the European Union or FCT or their services.

References

[1] A.P. Rodrigues, C. Correia, C. Varandas, A high performance real-time plasma control and event detection DSP based VME system, Fusion Eng. Design 60 (2002) 435–441.

[2] J. Kohout, A.D. George, A high-performance communication service for parallel computing on distributed DSP systems, Parallel Comput. 29 (2003) 851–878.

[3] C. Hogben, S. Griph, Interfacing to JET plant equipment using the HTTP protocol, JDN/H(02)11, CODAS Documentation Centre, http://www.iop.org/Jet/article?EFDR02004, 2002.

[4] Y.R. Martin, S. Coda, B.P. Duval, X. Llobet, J.-M. Moret, A new plant control software for the TCV tokamak, in: ICALEPCS 2005, Tenth International Conference on Accelerator and Large Experimental Physics Control Systems, Geneva, October, 2005.

[5] http://www.vmelinux.org/.

[6] http://www.mdsplus.org.

[7] http://www.ti.com.

[8] J.-M. Moret, A software package to manipulate space dependencies and geometry in magnetic confinement fusion, Rev. Sci. Instrum. 76 (2005) 073507.

[9] http://www.winehq.org/.

[10] B.P. Duval, X. Llobet, P.F. Isoz, J.B. Lister, B. Marletaz, Ph. Marmillod, J.-M. Moret, Evolution not revolution in the TCV tokamak control and acquisition system, Fusion Eng. Design 5657 (2001) 1023–1028.

[11] A.P. Rodrigues, N. Cruz, B. Santos, C.A.F. Varandas, J.-M. Moret, J. Berrino, B.P. Duval, TCV advanced plasma control system software architecture and results, in: IEEE NPSS, 15th Real Time Conference, Fermilab, Batavia, IL, USA, April 29–May 4, 2007.