GA–A23150

INTERPROCESS COMMUNICATION WITHIN THE DIII–D PLASMA CONTROL SYSTEM

by
D.A. PIGLOWSKI, B.G. PENAFLOR, and J.R. FERRON

JUNE 1999

DISCLAIMER

GA–A23150

INTERPROCESS COMMUNICATION WITHIN THE DIII–D PLASMA CONTROL SYSTEM

by
D.A. PIGLOWSKI, B.G. PENAFLOR, and J.R. FERRON

This is a preprint of a paper to be presented at the 11th IEEE/NPSS Real Time Conference, June 14–18, 1999, Santa Fe, New Mexico and to be published in *Transactions on Nuclear Science*.

Work supported by
the U.S. Department of Energy
under Contract No. DE-AC03-99ER54463

GA PROJECT 30033
JUNE 1999

Interprocess Communication within the DIII–D Plasma Control System*

D.A.Piglowski, B.G. Penaflor, and J.R. Ferron
General Atomics, P.O. Box 85608, San Diego, California 92186-5608

Abstract

The DIII–D tokamak fusion research experiment's real-time digital plasma control system (PCS) is a complex and ever evolving system. During a plasma experiment, it is tasked with some of the most crucial functions at DIII–D. Key responsibilities of the PCS involve sub-system control, data acquisition/storage, and user interface. To accomplish these functions, the PCS is broken down into individual components (both software and hardware), each capable of handling a specific duty set. Constant interaction between these components is necessary prior, during and after a standard plasma cycle. Complicating the matter even more is that some components, mostly those which deal with user interaction, may exist remotely, that is to say they are not part of the immediate hardware which makes up the bulk of the PCS.

The four main objectives of this paper are to 1) present a brief outline of the PCS hardware/software and how they relate to each other; 2) present a brief overview of a standard DIII–D plasma cycle ("a shot"); 3) using three sets of PCS sub-systems, describe in more detail the communication processes; 4) evaluate the benefits and drawbacks of said systems.

I. INTRODUCTION

The DIII–D Tokamak (DIII–D) is a national research facility under the auspices of the United States Department of Energy (DOE). This facility, located in San Diego, California, on the campus of General Atomics (GA), has provided an experimental environment for plasma and fusion research for over twenty years. The DIII–D Plasma Control System (PCS) directly monitors and commands various systems of the DIII–D tokamak to produce the desired discharges. This includes such systems as power supplies, gas injection, neutral beam injection, pellet injection, etc. [1]. In this way, the PCS provides real time control of most DIII–D sub-systems throughout the entire time cycle of the experiment, commonly referred to as a "shot."

II. DESCRIPTION OF PCS

The Plasma Control System is a specialized set of computer hardware and software which enables the users to manipulate the energy, density, shape and position of plasma within the DIII–D vessel [2]. Using the PCS, the user defines target parameters for the various sub-systems or defines directly the plasma characteristics. The user is allowed to select from a finite set of parameters, ranges and algorithms depending upon the desired effect to meet these targets. These control specifications are implemented within the software of the PCS. This gives the user additional ease and flexibility in the experiment design [1]. However, greater flexibility comes at the price of a more complicated control system, especially within the software.

Using a feedback loop, the PCS will take target parameters and attempt to continuously adjust the control settings of all specified devices to meet the intended targets for that time segment of the shot. This feedback control is done in real-time upon an isolated set of digital computational microprocessors, or CPUs. The output settings are then transmitted as a group of set-points across a digital or analog output system to the various DIII–D components needed in the experiment [2]. In a simplistic view, the PCS is dependent upon acquiring a set of data which it uses as a reference in comparison to the expected target to compute output settings for the mechanical devices which then in turn will produce a new set of data. The cycle will continually repeat itself until instructed otherwise at speeds of up to every 60 μs [2]. This is then done for all the requested sub-systems the user has included for the given experiment. As the shot progresses, the PCS will add archiving its own internal data to the final stages of activity.

The main computational resources accessible to the PCS are first its two RISC based workstations. Connected by a local area network, these two computers control most of the software of the PCS. Additionally six single processor real-time computers based upon an Intel i860 RISC designed microprocessor manufactured by CSP Inc. are used by the PCS. These communicate along the VME bus to the primary workstation allowing it to coordinate between the multiple CSP Inc. microprocessors as well as with the outside world. The peripheral hardware of the PCS allows it to communicate to the remaining DIII–D systems as well as an assortment of diagnostic systems [3].

The software that makes up the PCS consists mainly of ten separate subprocesses. Each is tasked with a set of responsibilities which define its individual purpose. These processes are distributed across the PCS computational resources, however a certain set of these PCS processes are restricted in their execution due to the hardware. In particular those processes which are related to the real-time computers. The remaining processes have greater flexibility during execution. For example, the main user interface may be run remotely from what is traditionally consider the PCS hardware. Although this is not normally the case [4].

*Work supported by U.S. Department of Energy under Contract No. DE-AC03-99ER54463.

III. DIII–D AND PCS SHOT CYCLE

A standard shot cycle consists of several progressive steps, under the direction of operators, which ultimately results in the creation of plasma within the DIII–D vessel. A single DIII–D plasma experiment, may take anywhere from a fraction of a second to several seconds. However the preparation and the ramp down/conclusion steps may take several minutes to perform. The normal sequence requires the coordination of the major sub-systems that contribute to the shot. The PCS shot cycle begins far earlier than that of the other independent control or acquisition systems. This early time is dedicated to preparing the layout of the experiment. More importantly, this is where the goals for shape, density, etc. are set-up within the PCS. An operator, usually a physicist, will define the parameter, waveforms and algorithms which will be used throughout the several shot stages. After the shot set-up has been configured to the satisfaction of the operator, the main DIII–D operations system will start in motion a chain of events, which for the PCS, will first lockout users from making changes in the shot set-up. Pertinent information is passed between the PCS sub-processes and timing is synchronized. At this point, the PCS will wait for other DIII–D sub-systems to complete their preparations and for the operations system to commence the shot ("firing the shot"). As the shot is fired, the PCS will begin its real-time processing until the conclusion of the plasma discharge. Later, the PCS will enter into an archiving stage where it will store its various internal information as well as time series it maintains throughout the complete experiment (Fig. 1).

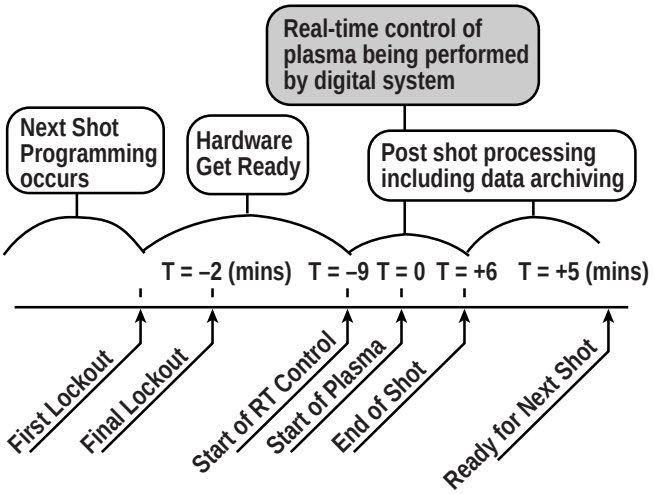

Fig. 1. DIII–D and PCS shot cycle timeline.

IV. PCS INTER-RELATIONS

The inter-relations of PCS software processes are almost entirely dictated by the layout of the PCS hardware. Although the software contains the concepts and goals of the plasma experiment, it is the constraints of the hardware which ultimately limits the control and what can be achieved. Even peripheral devices, i.e. Input/Output (I/O) dictate the purpose and function of specific PCS processes. This is why it is important to first grasp an overview of the PCS hardware before attempting to understand why PCS processes are related to each other.

From the standpoint of the user, the most visible part of the PCS is the user interface. It is through this interface that parameters and target vectors are set for an upcoming shot. This data however is only temporarily stored within the interface (Fig 2). As these values are edited, the information is passed along to the waveform server process. Only one waveform server process maintains the information for the immediate shot. Any number of wave server processes may run at any one time to allow for preparations of future shots but it is the waveform server designated for the next shot that communicates to other PCS processes. When changes are made to the set-up of the next shot (by an operator within the user interface), these changes are passed immediately onto the waveform server via a TCP/IP socket connection. There, the information is stored both in resident memory and in NFS mounted flat files. This information is then referenced, via the same TCP/IP socket by any of the PCS processes that may request it.

As the shot cycle nears and the PCS is triggered to prepare for the pending shot, the lock server will read the trigger from the digital I/O input and start the host processes for the real-time CPUs in their preparation of the shot. The lock server remains in constant communication with each of the host processes in order to monitor the current state of each. This monitoring will persist throughout the shot cycle and even during times in between shots. The lock server will also poll the waveform server for vital information about the shot. Of all the PCS processes, the lock server is the only one to remain in two way communication with the remaining PCS processes. It is arguably the simplest of all the PCS processes except perhaps for the message server but it serves one of the most crucial junctures of the PCS. The real-time host processes will poll the waveform server for the parameter, targets, algorithms, etc. necessary for the shot. Each host server, under the coordination of the lock server, will perform its programmed set of duties.

At anytime before, during, or after the shot, any PCS process may communicate to an auxiliary server, called the "message server." The main purpose of the message server is to gather textual information passed to it directly on its own TCP dedicated socket. This provides PCS processes, which have been programmed to display everything from status messages to error messages, a place to buffer their information. The buffered statements are archived for long term needs and passed to client applications, which are part of the standard PCS user interface. This process gives the programmers a flexible means to convey vital points of interest from most PCS processes. The message server/clients thus complete a circular flow of data back to the user (Fig 3). Like the lock server, the message server is in contact with all

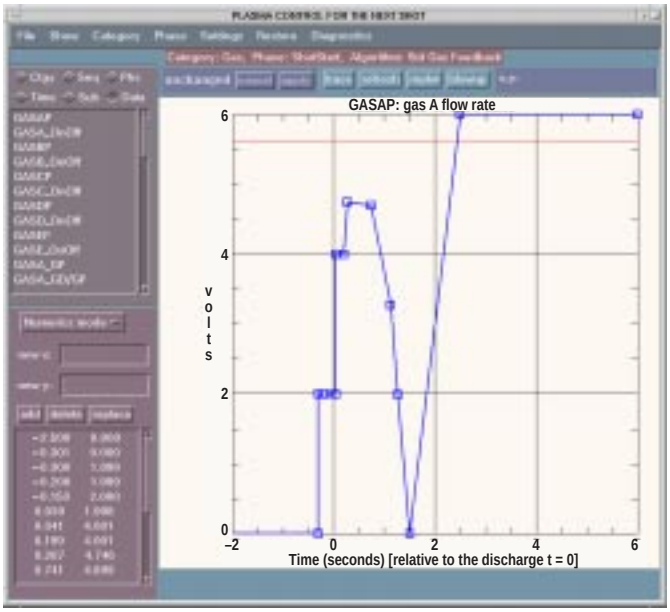

Fig. 2. Typical display from the user interface with information retrieved from the waveform server.

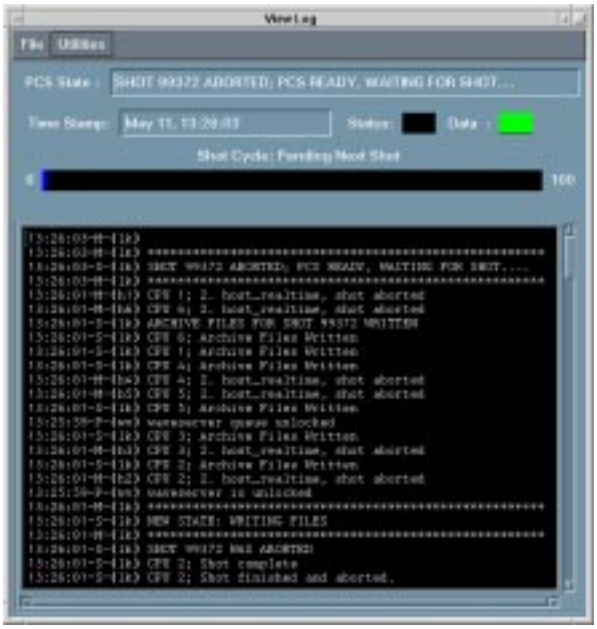

Fig. 3. Display of standard PCS messages.

the remaining PCS processes, however, the message server only receives a one-way communication from those processes and transmits data to only one of the PCS processes, the user interface.

V. METHODS OF COMMUNICATION

All the PCS processes handle data. The flow of data from each of the PCS "server" processes is often two-way. That is to say, each process will gather information and redistribute information (Fig. 4). The responsibilities of the process dictate to whom it needs to communicate with throughout the shot cycle.

The communication method of choice for most PCS processes is the TCP/IP socket. This means of transmitting data is commonplace among UNIX based systems and is quite flexible and reliable. TCP sockets also facilitate remote access from system to system, enabling the PCS to disperse its processes amongst several host machines. Long term storage needs are fulfilled by NFS mounted files. These files, which pass information to and from some of the PCS processes, as well as archive data, are a dependable means to store data between different versions of the PCS and when the PCS is not running. NFS cross mounted file systems allow for data in flat files to be shared remotely, as well. Files can be manipulated, via standard operating system (OS) utilities for backup purposes, viewing (as in the case of the messaging system), copying or duplicating. But as operating system utilities can be used beneficially they also pose a threat. In the past, crucial files have been vulnerable to corruption and deletion by non-savvy users who have accidently or unknowingly affected these files. The other well used method for communication within the PCS is the sharing of information through the use of a shared memory segment. This method is mostly used for the real-time code. It facilitates communication between the CPUs with the least amount of overhead. It is very efficient and allows for almost simultaneous update and retrieval.

Each of the methods mentioned above has drawbacks related to their use. In the case of TCP/IP socket communication, there may be system issues that need to resolved before using sockets. TCP sockets are also susceptible to network traffic. Simultaneous communication between several processes along a single port is done by linearly queueing the correspondences. This may impede the performance of all processes as they wait for their turn. Programming overhead is more complicated during setup and use of sockets than other forms of I/O. Also, the protocol of the messages themselves must be exact. In general, all this makes TCP socket communication trickier. File I/O, which is the second largest means of communication in the PCS, has its own set of drawbacks. Files are best used for large long-term storage of information. However they are inefficient for transmittal of minute amounts of data. They may be limited by available disk space. Data streams may be corrupted by other means, i.e. accidental removal of the file. Two way and instantaneous communication is terribly difficult when using files as a means. Memory sharing, the last method of communication, is restricted by physical access to the residual memory. It is also extremely limited by the amount of memory available.

Because of the inherent problems in each of the methods, it has been best to use a mixture of all methods for the PCS. The content and distribution of information has dictated the means within the PCS.

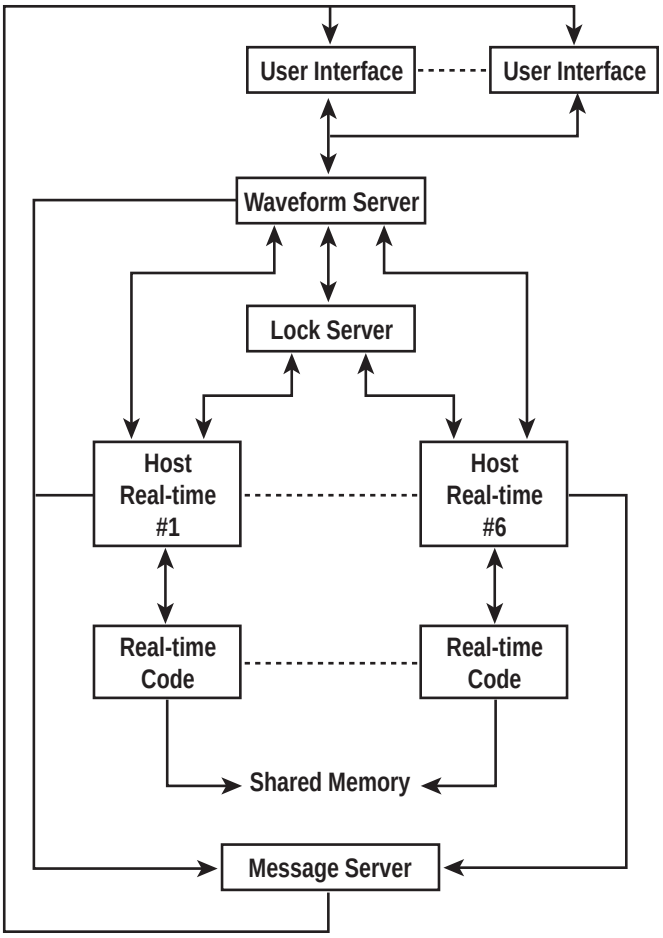

Fig. 4. Communication flow between PCS subprocesses.

VI. CONCLUSIONS

The success of the PCS at control of the DIII–D plasma experiments is due to its flexibility and relative ease of use. Given its complicated nature and wide array of responsibilities during shots, it has performed well over the years that it has been in place [4]. Part of this success is due in part to the distributed design and shared communication between the PCS sub-processes. It continually meets the demands of upcoming experimental needs given its outdated hardware. With projected improvements and upgrades, interprocess communication will become even more critical in order to meet new design criteria and functionality. The basic formulation described here has room to grow and expand with the PCS. Its standard construction should make it easy to port to other systems and platforms.

ACKNOWLEDGMENTS

I would like to thank the personnel at the DIII–D facility who aided me in the preparation of this paper. A special thanks is extended to Al Hyatt and Jim Broesch, Bob Johnson and Mike Walker for their insights to the Plasma Control System as well as their knowledge of the other DIII–D sub-systems, diagnostics and users point of view.

REFERENCES

[1] J.R. Ferron, *et al.*, "A Flexible Software Architecture for Tokamak Discharge Control Systems," Proc. 16th IEEE/NPSS Symp. on Fusion Engineering, September 30–October 5, 1995, Champaign, Illinois.

[2] B.G. Penaflor, *et al.*, "A Structured Architecture for Advanced Plasma Control Experiments," Proc. 19th Sym. on Fusion Technology, September 16–20, 1996, Lisbon, Portugal.

[3] J.R. Ferron, *et al.*, "Application Programmer's Guide to the DIII–D Plasma Control System," \to be published.

[4] B.G. Penaflor, *et al.*, "Current Status of DIII–D Real-Time Digital Plasma Control," these proceedings.