GA–A22160

CONF-950956--2

A FLEXIBLE SOFTWARE ARCHITECTURE FOR TOKAMAK DISCHARGE CONTROL SYSTEMS

by

J.R. FERRON, B. PENTAFLOR, M.L. WALKER,
J. MOLLER, and D. BUTNER

OCTOBER 1996

GENERAL ATOMICS

DISCLAIMER

GA–A22160

A FLEXIBLE SOFTWARE ARCHITECTURE FOR TOKAMAK DISCHARGE CONTROL SYSTEMS

by

J.R. FERRON, B. PENTAFLOR, M.L. WALKER,
J. MOLLER,* and D. BUTNER*

This is a preprint of a paper presented at the 16th IEEE/NPSS Symposium on Fusion Engineering, September 30–October 5, 1995, Champaign, Illinois, and to be printed in the *Proceedings.*

Work supported by
U.S. Department of Energy
Contracts DE–AC03–89ER51114
and W–7405–ENG–48

*Lawrence Livermore National Laboratory

GENERAL ATOMICS PROJECT 3466
OCTOBER 1996

A Flexible Software Architecture for Tokamak Discharge Control Systems*

J.R. Ferron, B. Penaflor, M.L. Walker, J. Moller,[a] D. Butner[a]
General Atomics, San Diego, California 92186-9784
[a]Lawrence Livermore National Laboratory

ABSTRACT

The software structure of the plasma control system in use on the DIII–D tokamak experiment is described. This system implements control functions through software executing in real time on one or more digital computers. The software is organized into a hierarchy that allows new control functions needed to support the DIII–D experimental program to be added easily without affecting previously implemented functions. This also allows the software to be portable in order to create control systems for other applications. The tokamak operator uses an X-windows based interface to specify the time evolution of a tokamak discharge. The interface provides a high level view for the operator that reduces the need for detailed knowledge of the control system operation. There is provision for an asynchronous change to an alternate discharge time evolution in response to an event that is detected in real time. Quality control is enhanced through off-line testing that can make use of software-based tokamak simulators.

INTRODUCTION

Active control of discharge parameters is playing an increasingly important role in present-day tokamak experiments and is expected to be a key design feature in future experiments such as TPX and ITER. For example, precise control of the discharge shape and position is required because of the effect on a wide range of parameters such as impurity influx through wall contact, coupling of auxiliary heating power, magnetohydrodynamic stability, confinement, edge localized mode (ELM) characteristics, and H–mode power threshold. Radiative divertor designs require precise control of the position of the X-point, the divertor strike points and/or the divertor radiation level. Advanced tokamak applications require control of the current density profile.

This paper describes the software architecture of the discharge control system in use on the DIII–D tokamak experiment [1-4]. This system implements control functions through software executing in real time on one or more digital computers. Because control applications are implemented in software there is little restriction on the discharge parameters that can be controlled and the type of control algorithm that can be implemented. The flexibility to make the frequent control application changes required to support the DIII–D experimental program is provided by organizing the software so that new control functions can be added easily without affecting previously implemented functions. The tokamak operator also has flexibility in the specification of the time evolution of the tokamak discharge including the capability to provide for an asynchronous change to an alternate discharge time evolution in response to an event that is detected in real time. The operator uses an X-windows based interface that provides a high level view that reduces the need for detailed knowledge of the control system operation. Software quality control is enhanced through off-line testing that can make use of software-based tokamak simulators.

MODEL FOR THE SYSTEM OPERATION

The task of the plasma control system is to send the commands to the various tokamak systems, or ``actuators" (e.g. magnetic field coil power supplies, gas valves, plasma heating sources etc.), that are required to produce the discharge desired by the operator. Sometimes this involves simply providing to the actuators a predetermined sequence of control commands but, more often, this involves performing some sort of feedback control.

Feedback control is implemented by repeatedly executing a simple cycle through the entire discharge period. This cycle consists of these steps: (i) measurements are made of various tokamak diagnostic signals, (ii) the value of some quantity to be controlled is calculated from the diagnostic data, (iii) the calculated value is compared to a desired value and (iv) the required commands to the actuators to correct any difference between the actual and desired values are calculated and communicated to the actuators. The complexity of the calculations that are performed to execute one control cycle determines how rapidly the cycle can be executed and the frequency bandwidth of the commands sent to the actuators.

The basic model for the control system software is that the device being controlled runs in pulses, with the capability for the pulse length to be essentially indefinite. The software synchronizes with the start of the tokamak pulse and provides the capability for the operator to specify the way in which the various control system parameters should evolve as a function of time after the pulse begins. One or more computers are dedicated to executing the code that performs the control functions. During the tokamak pulse, these real time computers execute without operator intervention. The operator specifies, before the pulse, all of the data that is required to run the complete pulse. The amount and type of data required depend on the sophistication of the control algorithms. These data are loaded into the memory of the real time systems during pre-pulse preparation and are referenced by the control algorithms during the pulse.

HIGH LEVEL OVERVIEW

The control system is organized with distinct tasks distributed among multiple processes. Fig. 1 shows a block diagram of these processes and the communication paths

*Work supported by the U.S. Department of Energy under Contract Nos. DE-AC03-89ER51114 and W-7405-ENG-48.

between them. There can be multiple real time computers, executing control applications in parallel. Each of the real time computers is paired with a process executing on a host computer that handles communication with the remainder of the system, communicating with the real time computer through shared memory. During a discharge, each of the real time computers runs a single program, consuming the entire computation bandwidth, that implements the feedback cycle. Apart from the real time code, the control system processes execute on a host computer. Communication between the processes is by message passing using the standard Berkeley socket mechanism provided by the UNIX operating system. Use of this communication method allows the system processes to be distributed among multiple computers if required. The host computers are not required to perform real time functions during the tokamak pulse.

The "lockout server" provides synchronization with the tokamak discharge cycle. This server detects the moment prior to a new discharge after which no more changes in the discharge specifications can be made and the control system begins its preparation for the discharge. The lockout server coordinates the activities of the various processes during pre-pulse preparation, waits while the discharge executes, and coordinates the activities during post-pulse cleanup.

The waveform server holds the database of specifications for the time evolution of the discharge. This database is divided into "raw" and "processed" data. The raw data are obtained through messages from the user interface and consist of specifications in terms and units that are familiar to the operator. The processed data are computed by combining the raw data from the operator with any necessary additional precomputed data loaded from disk files, in a manner specified by the control algorithm designer, to produce the data the real time computer requires to execute the discharge. This separation of the raw and processed data using a computation that can be unique to a particular control function allows details of the control implementation to be hidden, providing the operator with a more friendly interface.

The functions of the waveform server are split into synchronous and asynchronous portions. Communication with the user interface is asynchronous. Whenever a request for a message exchange arrives from a user interface, this request is honored immediately. One copy of the raw data is updated during each exchange of messages. When an operator wishes to examine the current set of specifications for the discharge, the waveform server can always respond quickly by consulting this copy. A separate record of each message is placed on a "job queue." The messages in this queue are processed sequentially with any necessary work being done to recompute processed data that depends on the raw data that was altered. Because this computation could be intensive it is performed in the background without requiring the operator to wait for its completion during each exchange of messages.

An X-windows application provides the operator with an interface to the control system. Using an X-terminal or workstation the operator has point-and-click access to all functions necessary to specify the discharge parameters. The operator is aware only of the "control panel" presented on the X-windows display which hides the details of the process structure shown in Fig. 1. There can be multiple operators, each executing a user interface process and each accessing the same database of specifications stored by the waveform server. Access to the servers is arbitrated by the socket mechanism, with multiple requests for message exchange being queued by the operating system.

The control system provides a generic construct called a "waveform" that the operator uses to specify the time evolution of a discharge parameter. For instance, the desired total plasma current as a function of time would be specified using a waveform. Typical control algorithms require the operator to change relatively few parameters from one discharge to another. So, most of the operator's activity is in modifying a few waveforms for each algorithm, each of which communicates a single value versus time to the control system. A given algorithm may require the ability for the operator to modify data structures that do not match the generic usage of a waveform. For this purpose, the algorithm designer can design data structures unique to the algorithm and provide acustom user interface window so that the operator can modify this algorithm-specific data.

ORGANIZATION OF CONTROL APPLICATIONS

The number of parameters of the tokamak discharge that can be under feedback control is large. In addition, the time

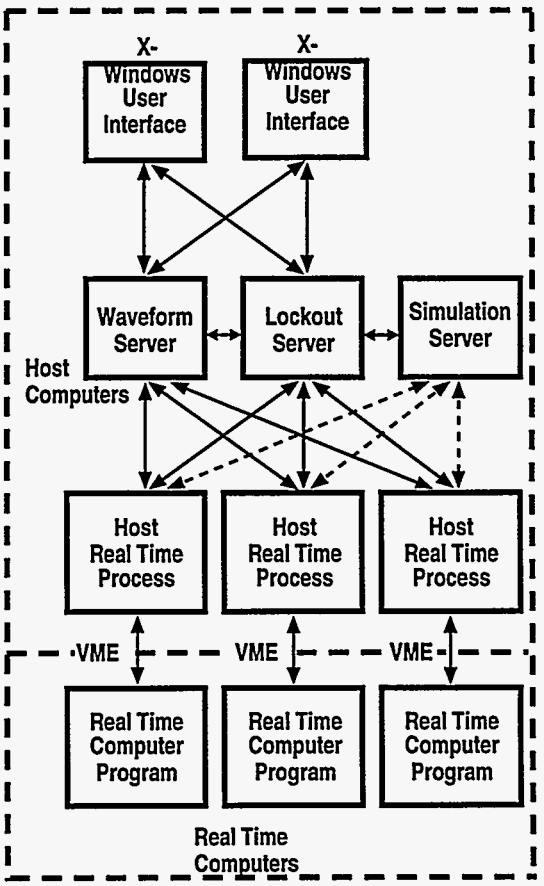

Fig. 1. Block diagram of the computer processes in the control system. The arrows represent communication paths.

evolution of the discharge parameters that is required to support an experiment can be quite complex. The operator needs a systematic way to choose among the large number of options for specifying the discharge parameters, and the control application designer and programmer must be able to coordinate their control functions with many other control applications. Several levels of structure are used to help make these jobs organized and manageable.

The control application designer organizes the controlled parameters into "categories," usually determining the groupings based on the type of actuator used. For instance, for DIII-D, discharge shape control is one category and the actuators for this are the power supplies for the poloidal field coils. These groupings provide an organization method to aid operator understanding of the control system and to use in storing control system data.

The method used to control the discharge parameters in a given category is the "algorithm." The choice of control algorithm determines the data required from the operator, the data provided to the real time computer and the computations performed in real time to perform the control function. For each control category the operator selects, from a list of available choices, a single algorithm to be in use at any given moment during a discharge.

For each control category, the operator can specify any number of "phases" of a discharge. A discharge phase is a segment of time during the tokamak pulse during which a specified control algorithm is in use and during which the discharge parameters controlled by that algorithm should evolve with time in a manner specified by the operator.

The use of the discharge phase is illustrated in Fig. 2. The figure shows, as an example, three discharge phases for the category that controls the ohmic heating coil power supply. In phases #1 and #3 the algorithm chosen controls the power supply to produce a particular plasma current as a function of time (I_p). In phase #2, the algorithm is designed to produce a required loop voltage (V_{loop}). The time evolutions of I_p and V_{loop} are programmed separately relative to time zero for the appropriate phase. Defining time relative to the start of the phase allows the segment of time in a discharge in which a given phase is in use to be easily relocatable. A phase can continue indefinitely because there is no ending point for the definition of the time evolution.

For each category, the operator specifies a sequence of one or more phases to be active during the discharge. For instance, if several separate experiments are being conducted during a single discharge, the operator might create a phase for each experiment. There is a primary sequence of phases that starts at a time that is synchronous with the tokamak pulse. In the example of Fig. 2, phase #1 is chosen to start at t = 0 s during the discharge and phase #2 is selected to start at t = 2 s. One or more alternate sequences of discharge phases can also be included, any of which could become active asynchronously during a discharge in response to some event detected by a control algorithm. The operator would program the time evolution of discharge parameters in an alternate sequence of

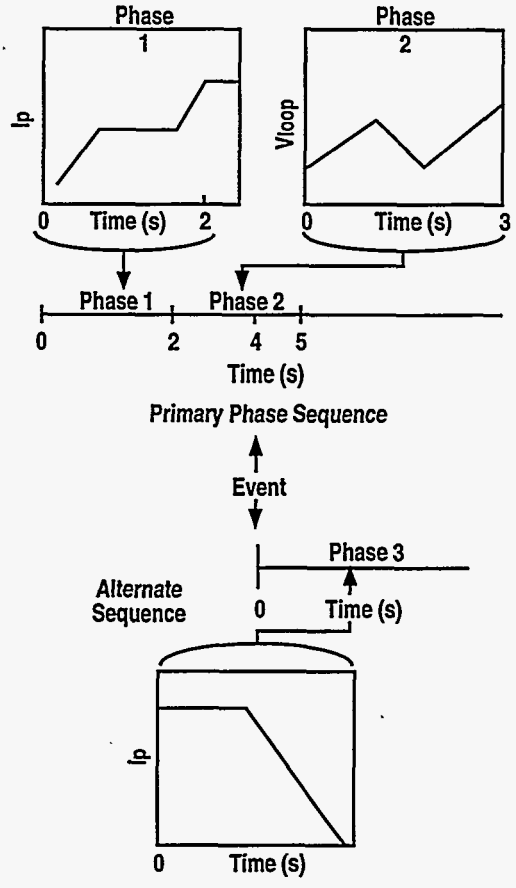

Fig. 2. Illustration of the use of the discharge phase and phase sequence.

phases to include the proper response to the event that was detected. For instance, in Fig. 2 an event was detected at t = 4 s that required the discharge to be ended gently. A switch is made to the alternate phase sequence in which phase #3 uses an algorithm that brings the plasma current slowly to zero.

SOFTWARE FOR THE REAL TIME COMPUTERS

The code on the real time computers executes the feedback cycle to provide all of the tokamak control functions during a discharge. The operator's choices of control algorithm determine the specific functions to be called and the input parameters provided for each discharge phase determine exactly how the feedback algorithms behave.

A set of generic data structures on the real time computer provides the real time code with input data from the waveform server and a place to store output data. It is also possible to make use of data structures that are custom designed for a particular algorithm. The generic structures are vectors (one dimensional arrays), each element of which specifies a single, time varying parameter to the control algorithm or holds an output value computed by a control algorithm. The data structures on the real time computers can vary in size from one discharge to another with memory allocation performed for each discharge as required. A single central

structure contains pointers to locate all other structures. To avoid software failures in real time resulting from lack of memory, all memory on the computer is allocated during the pre-pulse preparation phase. The values in the vectors that provide input parameters are fixed during a feedback cycle, but can change between cycles to provide the mechanism for input parameters that change as a time function.

The "target" vector is the primary source of input values for the real time code that implements a control algorithm. Each element of this vector contains either a floating point value or an integer value that can vary as a function of time. Typically a target vector element is the desired value of a feedback controlled parameter or a switch or some other parameter used to control how the control algorithm behaves. An integer value in the target vector can also be a pointer to an algorithm-specific data structure.

The functions to be executed in real time are specified in a flexible manner rather than being fixed at compile time. The "function" vector contains a list of pointers to the functions that execute the control algorithms chosen by the operator. To execute one feedback cycle, each of the functions in this list is called once. If the operator chooses to change the algorithm as a function of time during the discharge, the content of the function vector will change in time.

In order to provide a way to diagnose the performance of the control system software, the input buffer for the measured diagnostic values and the output vectors are allocated as arrays of vectors. The set of vectors in use during a given feedback cycle is fixed, but at the end of each cycle the pointers to the vectors in use can be changed to new, as yet unused vectors, leaving behind the values written on the previous cycle. This produces snapshots of the complete set of input and output values, all from a single feedback cycle.

APPLICATION PROGRAMMING

The control system can be completely configured for each unique installation by writing the appropriate software. The control system designer combines object code libraries containing the installation independent support facilities with installation specific code written in the C programming language to implement all control application functionality. Emphasis is placed on providing as much generic capability as possible as part of the support facilities in order to minimize the work required to implement a control application. The installation specific code is used to define the control categories, the control algorithms, the real time computer configuration and which categories and algorithms execute code on each of the real time computers.

Each custom control system component is defined in a "master" file which is created to define each real time computer, each control category, and each control algorithm. The master file contains several sections of source code, each of which is automatically included in one or more of the system processes (waveform server, real time host process or real time code) when the code is compiled. Adding a new computer, category, or algorithm involves modifying only the master file for that function without disturbing the files associated with the remainder of the control system code.

To add a control algorithm to the system, a master file is created that defines the way the algorithm uses elements of the real time data vectors, defines the waveforms that appear on the user interface for that algorithm, uses library routines to specify how the operator-provided raw data is converted to the processed data for the real time computer, defines the structure of any algorithm-specific data and includes the code to be executed in real time to implement the algorithm.

TESTING USING THE SIMULATION SERVER

Debugging control algorithm software during operation of the tokamak is expensive, risky for the tokamak hardware and inefficincy. Therefore, it is desireable to have a way to accomplish two primary debugging tasks in an off-line mode: (a) determining whether the control algorithm code implements the algorithm correctly and (b) determining whether the control algorithm can properly control the relevant tokamak parameters. The simulation server is a separate process (Fig. 1) that emulates the tokamak systems to provide this off-line debugging capability. During each feedback cycle the real time computers receive the tokamak diagnostic data from the simulation server and return to the server the computed actuator commands.

The simulation server is customizable to match the control algorithm under test. In the simplest case the server passes the data recorded from a previously executed discharge to the real time computers so that the algorithm computation results can be checked. A more complex server simulates the tokamak's response to actuator commands to test the complete feedback control functionality of the algorithm.

SUMMARY

We have described the software structure that provides the power and flexibility required for a discharge control system to support the rapidly evolving DIII–D tokamak experimental program. The separation of the basic software framework and the application-specific code allows the rapid implementation of new control systems through modification of only the application code that implements the required control algorithms. For instance, at DIII–D the same software and hardware technology have been used to implement the tokamak discharge control system and a control system for the ICRF transmitters. There is limited dependency of the software on the control system's hardware architecture allowing relatively rapid modification to take advantage of future advances in computing hardware. Experience gained with this system during operation of DIII–D advanced tokamak and radiative divertor experiments will result in a control system thoroughly tested on a operating tokamak that can be adopted directly by future experiments, reducing future development time and costs.

REFERENCES

[1] J.R. Ferron, *Rev. Sci. Instrum.* 63, p. 5464, 1992.
[2] J.R. Ferron and E.J. Strait, *Rev. Sci. Instrum.* 63, p. 4799, 1992.
[3] J.R. Ferron, A.G. Kellman, E. McKee, T.H. Osborne, P. Petrach, T.S. Taylor, J. Wight, in Proc. of the 14th IEEE/NPS Symp. on Fusion Engineering, p. 761, 1991.
[4] G.L. Campbell, J.R. Ferron, E. McKee, A. Nerem, T. Smith, E.A. Lazarus, C.M. Greenfield, and R.I. Pinsker, in Proc. of the 17th Symp. on Fusion Technology, p. 1017, 1993.