LA-UR -83-943

CONF-830406--12

Los Alamos National Laboratory is operated by the University of California for the United States Department of Energy under contract W-7405-ENG-36

LA-UR--83-943

DE83 009902

TITLE SOFTWARE DESIGN FOR THE TRITIUM SYSTEM TEST ASSEMBLY

AUTHOR(S) Garnett W. Claborn, E-8
Robert T. Heaphy, Consultant
Paul S. Lewis, E-8
Lawry W. Mann, CTR-10
Clair W. Nielson, CTR-10

MASTER

SUBMITTED TO 5th Topical Meeting on the Technology of Fusion Energy, Knoxville, TN (April 26-28, 1983)

DISCLAIMER

Los Alamos National Laboratory
Los Alamos, New Mexico 87545

EHB

SOFTWARE DESIGN FOR THE TRITIUM SYSTEMS TEST ASSEMBLY*

G.W.Claborn, R.T.Heaphy, P.S.Lewis, L.W.Mann and C.W.Nielson
Los Alamos National Laboratory
Los Alamos, NM, 87545
505-667-1153

ABSTRACT

The control system for the Tritium Systems Test Assembly (TSTA) must execute complicated algorithms for the control of several sophisticated subsystems. It must implement this control with requirements for easy modifiability, for high availability, and provide stringent protection for personnel and the environment. Software techniques used to deal with these requirements are described, including modularization based on the structure of the physical systems, a two-level hierarchy of concurrency, a dynamically modifiable man-machine interface, and a specification and documentation language based on a computerized form of structured flowcharts.

INTRODUCTION

The purpose of the Tritium Systems Test Assembly is to design and operate a facility to demonstrate the safe handling of tritium in a manner similar to that required for a future fusion reactor. Many of the components and systems required for this purpose are novel and it is expected they will be undergoing change as experience with the facility evolves. The integrated hardware, software, and instrumentation must provide a broad range of functions, including data acquisition, routine and emergency control, operator interaction, and data archiving. The facility operates under automatic control with a minimum of operator interaction.

Because the feasibility of a fusion reactor depends on the availability of the tritium-deuterium fuel system, high availability is one of the major design objectives for TSTA. In addition, the radioactivity present in the tritium mandates a number of safety precautions above and beyond those expected for other systems of this size and complexity. For both of these reasons, a variety of redundant components are included, both in the tritium handling systems and in the control system. A combination of the redundancy necessary for reliability and the desire to divide the processing load in a beneficial way has led to a configuration of two minicomputers and several microcomputers as described in the first part of this paper. The need for effective human interaction with the system, both from the operators and the designers, and for the need to modify that interaction as experience develops has led to the form of Man Machine Interface described in a later section. Finally, the fact that the specification and verification of control algorithms must be done by non-programmers has put a great deal of emphasis on modularity and on structured design. The character of this modularity and the structured flowcharts used to specify the design are described in the last part of the paper.

OVERALL CONFIGURATION

The hardware configuration is split into a process system and a safety system. Each is an independent computer-based system with its own interface to the physical processes. Both systems must handle several thousand TSTA measurements and controls. The only interconnections between the two systems are four optically-isolated serial lines.

During normal operation the process system controls the facility. It has access to all TSTA parameters and controls all TSTA subsystems. The process system is the first line of defense against an accidental malfunction. The safety system adds further protection with independent routes for sensing and control of critical parameters. During normal operation the safety system is solely a monitor. However if it detects either a failure of the process system or a potentially dangerous situation within the facility it shuts down the facility and runs the Emergency Tritium Cleanup system (ETC) if necessary.

PROCESS SYSTEM

The process system is built around two Data General C-330 Eclipse minicomputers. One is used for system control and the second

*Work performed under the auspices of the USDOE.

serves as backup. The backup machine is also used for software development. Both are connected to the TSTA transducers and controls through a standard CAMAC (Computer Automated Measurement And Control) interface.

The CAMAC hardware is configured into two branches of CAMAC crates. Within each branch are two Digital Equipment LSI-11 microcomputers that act as front-end controllers (FEC). Each FEC controls a distinct set of the CAMAC crates. The data acquisition and control components for each TSTA subsystem are grouped together into one of these sets. This results in each FEC having complete responsibility for the interface to a number of subsystems with no subsystem overlap.

The process computers use the Data General Advanced Operating System (AOS), a multiprocess operating system which allows many relatively autonomous computer programs, called processes, to execute concurrently, with the operating system time-slicing their execution. By means of explicitly programmed system calls, one of these processes may send data to another, assuming the other has also programmed a request to receive such data. This highly structured interchange of data contributes to program clarity and minimizes the possibility of unintended interactions between the different processes.

The software for the process computer consists of several of these independent processes. These processes may be viewed as being of two kinds. The first (resource managers) manages one of the general components of the software system while the second (control processes) manages a specific TSTA physical subsystem. Among the first kind are a process to manage the data coming from and going to the FEC microcomputers (CAMAC), a process to manage data passed between control processes (IPD), and a process to control the man-machine interface (MMI). Other processes handle data archiving, and logging. The responsibility for these general purpose processes lies primarily with the software staff. The second kind of process handles the control of a specific physical subsystem such as the isotope separation system (ISS) or the fuel cleanup system (FCU). There are about ten of these control processes. The major responsibility for these algorithms lies with the subsystem designers; however, it is the responsibility of the software staff to maintain a precise specification of the algorithm and to insure its accurate implementation. One master control process coordinates the interaction of all of the subsystems. However, as will be described below, the amount of interaction between the subsystems is very limited and every attempt is made to isolate these interactions to the single master control process.

The control process algorithms are implemented using a few primitives corresponding closely to simple physical operations, i.e., closing and opening valves, setting setpoints, and sensing physical parameters. The details of how these operations are transmitted though the computer systems and CAMAC hardware are handled by the resource managers and are hidden from the process code.

The front-end controllers utilize the RT-11 operating system. Their principal function is the acquisition of data from the CAMAC interfaces and the transmittal of commands to those interfaces. During normal operations these two functions are performed as off-loading for the process computers. To this end, a constantly running foreground program reads all of the data modules associated with a particular FEC and tabulates the data values to the CAMAC memory mailbox where it may be accessed by the process computers. A second part of this foreground program receives control requests placed in the memory mailbox by the process computers and executes them through the CAMAC hardware.

The FEC software is written in a table driven manner. It maintains its CAMAC hardware configuration and variable assignments in a set of dynamically alterable tables. On initialization each front-end controller reads in a set of data files defining its particular configuration. This permits the use of the same foreground code in all four of the FECs. It also allows the dynamic reassignment of CAMAC modules via process computer command in the case of component failure.

CAMAC INTERFACE MANAGER

The purpose of the TSTA CAMAC interface manager is to provide the subsystem processes with the ability to sense and set the values of TSTA physical measurement and control variables in a straightforward manner. For analog measurements and controls this means dealing with the variables in appropriate engineering units. For digital measurements and controls this means providing a limited mnemonic set of defined states for each.

There are two levels to this problem. The first is the actual control of the CAMAC hardware to read or write digital values for physical measurements and controls. The second is that of converting between the raw digital data used by the CAMAC hardware and the engineering units or defined states used by the subsystem processes. The CAMAC interface manager is divided into these two levels. A large subsystem contains about 200 measurement and control devices. A naming convention for these devices had been specified early in the TSTA design process. It is therefore of great value that any algorithm specification use these notations, and that any compilation procedure use them automatically to avoid errors due to human translation between the device designation and the numerical address specification in the computer. A preprocessing symbol definition facility makes this matter straightforward; a central definition file used by all TSTA programs makes the translation from the standard device designation to the appropriate computer addresses. All interaction with TSTA measurement and control variables done by processes consists of subroutine or function calls utilizing the defined value of the variable along with the analog or digital data to be read or written.

Data tables in FEC memories keep track of all of the CAMAC hardware, the TSTA variable assignments to that hardware and the current values for the TSTA variables. These tables are initialized on startup from special files. All configuration dependent information is contained in these tables. This allows the same update and interface software to be used for any TSTA variable-CAMAC hardware configuration. The input data is transferred to the process computer on a cyclic basis. The control information is transmitted to an FEC queue and processed in sequence.

The function of the high level CAMAC software in the process computer is to convert between the TSTA variable values used by the high level processes and the digital values sent to or obtained from the CAMAC hardware. Each TSTA variable has a conversion type and subtype associated with it. Example of types are; PRESSURE, TEMPERATURE, SET_TEMPERATURE, VALVE, SET_VALVE. Subtypes are consecutive numbers from 1 to n that indicate the exact conversion formula to be used within the type for that variable.

INTERPROCESS DATA

All communication between the different control processes (as contrasted to communication between a control process and the physical equipment, or the control processes and the operator interface) is through the Interprocess Data Manager (IPD). IPD stores data in logical mailboxes. These mailboxes have access control such that any process may read them but only the owning process may change them. The data they contain are called "boundary conditions", a terminology which is used to denote both physical parameters which may affect the interaction between subsystems, and setpoint values and mode switch settings which may be either operator or algorithmically determined. For example, a request from MSP (Master Sequencing Process) to go to the isolate safety mode is sent to the Interprocess Data Manager (IPD) which then provides it to each of the subsystem control processes, which of necessity, must interrogate this parameter on a frequent cyclical basis. The subsystem control processes can then direct their subsystems in a manner consistent with the dynamically changing physical boundary data from the IPD and with the subsystem's internal data as obtained from CAMAC (CAMAC Interface Manager).

The passive quality of interprocess data, together with the relative autonomy of each control process, make the complex interactions between subsystems easier to understand. The algorithm for each subsystem is designed and analyzed in the limited context of the relatively small number of boundary conditions which may impact it.

THE MAN-MACHINE INTERFACE

The operator interface to the process system is made through the Man Machine Interface (MMI). The MMI is implemented on an eight-color high resolution, 19-inch, character graphic Aydin terminal. The terminal has 48 lines of 72 columns, a standard ASCII keyboard plus 45 special function keys. Features such as character intensities, blink, reverse video, protect, and color are selectable on a character by character basis.

The screen of this terminal is divided into three distinct functional areas. The center and largest part of the screen is devoted to pictures (diagrams and menus) selected by the operator to best assist him in the control or monitoring function of present interest. Each of these pictures is identified by a name through which it is called up. The

top three lines of the screen are devoted to alarms and messages sent by the various subsystem processes, and queued in a manner described later. The bottom four lines of the screen are devoted to operator interaction, such as answers to specific questions from a sub-system process, and the insertion of commands when appropriate.

The operator interaction can occur on several levels. The lowest level consists of typing individual commands to control components, sense variable values, change displays, view alarms, and provide harduopy output. Examples of commands are SET, SENSE, SHOW, OPEN, CLOSE, and PLOT. Any of these commands may be typed in in the command area. At this level of interaction, sensors and controls may be exercised individually, (although this mode of operation is limited to testing and check out of the components). Any TSTA component is identified in these commands by the same name used in the engineering drawings and specifications. After initial checkout of the original or modified system, operator interaction with a subsystem is through that subsystem's control process. At this level, only modes and options may be set, the control of individual components being algorithmically determined.

Either level of interaction may be simplified through the use of macros. A macro is a file containing a list of commands to be executed. The macro may contain elements of a macro language that allows testing, looping and argument passing to the macro. A macro can be generated while MMI is running and used without requiring a restart of the MMI. Macros can be used to provide more convenient commands, and to avoid long sequences of commands.

Frequently used commands are assigned to function keys in an easy to use arrangement. Those function keys which do not effect control are executed upon actuation. Those which effect control are echoed for operator confirmation. Function keys are provided to move between recently accessed displays. A circular queue of pictures allows easy access to a related selection of pictures.

In order to reduce the amount of typing and to provide the operator with the most relevant selection at the appropriate time, cursor selection from menus is provided. A cursor selected action is echoed on a command line and then confirmed by the operator to insure that no error in selection has been made. Menus may be freely added at any time without recompiling or relinking of the software through the addition of defining text files.

Subsystem pictures are defined in a special graphics language. A file containing the graphic language is preprocessed into a form usable by MMI. The preprocessing can occur while MMI is running, and the new or modified pictures or menus can be made available without restarting MMI. Subsystem displays normally contain a block diagram showing subsystem parameters. The parameters are updated in a cyclic fashion, and should a parameter go outside a predefined range, the parameter is made to blink in order to draw the operator's attention.

EVENT LOGGING

The purpose of event logging is three fold. The first is to alert the operator to potentially dangerous conditions through the use of alarms. The second purpose is to provide a mechanism by which a process can communicate with the operator. The third purpose is to provide a hardcopy history of system events. An event message is the message sent from a process to the LOG process. The LOG process uses the top three lines of the Aydin terminal for the displaying of event messages. All event messages are written to the hardcopy device.

The LOG process places event messages in a queue according to priority. The operator may acknowledge single messages, or all messages from a given process, or all messages of a given priority. If a message is displaced from the Aydin screen by one of higher priority, it is not removed from the queue until it is acknowledged and it will be displayed again when it becomes one of the top three messages on the queue.

DATA ARCHIVING

The purpose of data archiving is three fold. The first is to allow offline data reduction and analysis. The second is to provide a mechanism whereby the system history can be analyzed by reviewing the archives on the back up computer. The third is to provide a restart capability. An archive consists of the CAMAC shared pages containing the physical variables and the IPD shared pages containing the logical variables. Thus an archive provides a snapshot of the system state at a given point in time. Each archive is written to an archive file. A particular archive file contains a certain number of archives. After the file has been filled with archives it is no

longer used for archiving and a new archive file is started. The old and new archive files are chained together in chronological order.

It is desirable for the subsystem designers to be able to review the past history of subsystem operation in order to determine the performance of the subsystem. An offline version of MMI is used to display pictures and to sense variable values captured in the archive. During a review, the user is able to change which archive is currently being used. During this review, all set commands are disabled so that the user is not able to alter the archive in any way. Finally, a library set of subroutines is provided for accessing the data in the archives so that scientists and engineers can write FORTRAN programs to analyze historical information in an arbitrarily complex way.

TSTA SUBSYSTEM PROCESSES

The Tritium Systems Test Assembly is comprised of several subsystems, each with a well defined function in the overall operation. This modularization of the the physical system provides a natural modularization of the related software, and every effort has been made to reflect the physical organization of TSTA in the associated software. Each significant physical subsystem is implemented as a separate process under the Data General AOS operating system. Isolation of a subsystem's software has several advantages. It provides a much needed modularization to keep a given program of comprehensible size. Since the physical devices associated with the subsystem may be controlled only from the associated process, access control to physical devices is readily implemented. Verification and debugging of the process associated with a particular subsystem can be done independently of other subsystems, since each subsystem has modes of operation wherein it can be run alone, and the software's isolation in a single process makes it possible to operate it separately from the software associated with other TSTA subsystems.

Most of the subsystems are sufficiently complex that their control programs benefit from further division. The larger subsystems, in particular, are composed of several hundred components. In most cases, the components can be grouped into functional clusters. These clusters are called devices. The algorithm for the control of each of these devices is implemented in a task, a programming construct available within the AOS operating system which allows an independent execution path through a set of subroutines within a single process. Different tasks within the same process share memory space (and therefore may communicate through common variables) but are time-sliced independently. The independent time slicing is advantageous because it provides a convenient mechanism for the cyclic checking of the state of a single device within a subsystem with the clarity of treating that device in an isolated sequence of code. This two level hierarchy of concurrency, first the process level for separate subsystems, and second the task level for separate devices within subsystems, is very effective in the present application.

Each subsystem, then, consists of a main task, which coordinates the various devices, and several device tasks, which communicate with the main task through mailboxes. Although the tasks within a single subsystem could communicate more directly, for example through common variables, the mailbox communication used for interprocess communication is utilized for intertask communication for two reasons. First, most of the data needed for intertask communication is also essential or at least desirable as data for operator displays and for archiving. Second, the uniform handling of data makes the algorithms more easily comprehensible. The multi-tasking mechanism provides the necessary tools for synchronizing different device clusters when necessary through standard operating system procedures.

SUBSYSTEM ALGORITHM SPECIFICATION LANGUAGE

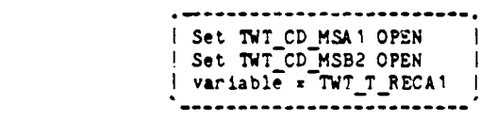
The computer programs for control of TSTA are written in the RATFOR language and preprocessed to FORTRAN on the respective host computer systems. This gives a uniform interface to the programmer for very different computer systems. RATFOR provides excellent control logic facilities and provides the preprocessing of symbols referred to above.

Although the RATFOR language has excellent control structures, it has been found that a graphical presentation of the logic is superior for specification and verification. While this is important to the software staff, it has been found to be of even greater importance to the non-programming scientists and engineers who have the ultimate responsibility for the functioning of TSTA. Traditional flowcharts are not satisfactory. To most viewers, a traditional flowchart is harder to understand than a clearly indented RATFOR program. On the other hand, certain forms of structured flowcharts have been found to be very effective. A structured flowchart has unique graphical constructs for the standard

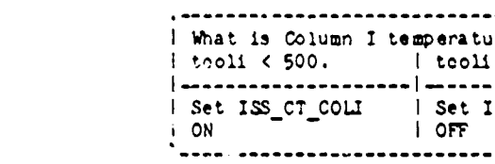
structured programming control structures. More important, it expands not by connection with lines, but by magnification through blocks.

The three different kinds of action chosen for TSTA algorithm specification are (1) the simple step, (2) the multiway branch, and (3) the iterative loop. A simple step executes a single action such as opening a valve or reading a temperature. A multiway test determines which of several alternatives to execute based on the value of some physical measurement or some operator or algorithm determined option. Finally, an iterative loop repeats a sequence of steps until some condition is true or false, or for a fixed number of repetitions.

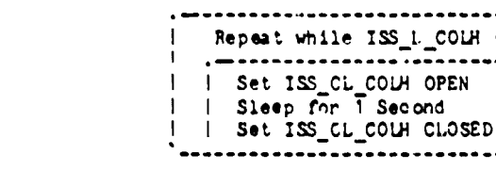
The simple step is represented in our structured flowcharts as a rectangle, with one or more explicit actions stated in a terse manner inside the rectangle, e.g.,



The multiway test is represented by a rectangle with internal vertical lines which do not quite intersect the top of the rectangle, e.g.,



The final structure is the iterative loop. It is drawn as an inverted "L" encompassing the range of steps to be repeated, e.g.,



Structures are nested to create arbitrarily complex structured flowcharts. For example, an iterative loop may be one of the columns of a multi-way branch.

In order to be able to create and modify structured flowcharts conveniently on a computer, an input language to specify them has been designed and implemented. While entering them exactly as they are to appear is possible using various special character keys, such an input scheme is both inconvenient initially and makes modification a difficult job. The input language, on the other hand, is easier to input and modify. It looks very similar to the RATFOR code which will ultimately implement the flowchart in an executable program.

SAFETY SYSTEM

The safety system is based on two Digital Equipment LSI 11/23 microcomputers. One is used as the active safety computer and the other as backup. The backup machine is also utilized for software development.

The safety computers run the RT-11 operating system. They use the RT-11 XM (extended memory) monitor which allows operation in a foreground/background mode. During operation the active safety computer's foreground program tabulates data from the safety system's CAMAC interface and executes CAMAC output commands issued by the background program.

The background program of the active safety computer monitors all critical safety parameters. It provides displays of the safety parameters on both a request basis from the operator and on an alarm basis for abnormal values. It also continually evaluates the safety of each subsystem as revealed by the values of certain critical parameters. If these exceed initial limits, warnings are given to the operator. If they exceed further limits, the safety computer undertakes to shut down the system using its override capability on critical control variables. The background also carries on a status dialog with the process system over the four serial links. If it determines that the process system is not functional it will take over control of TSTA and shut the system down. In the event of a process system failure during a tritium emergency the safety system will additionally take over and run the ETC subsystem in order to reduce the tritium in the building to a safe level.