27
6/12/79
2470 UTIS

MASTER

UCID- 13186

Lawrence Livermore Laboratory

Controller Routines for the DECsystem-10 With Application to a
Tandem-Mirror Plasma Code

Donald R. Faul, R. S. Devoto

April 12, 1979

This is an informal report intended primarily for internal or limited external distribution. The opinions and conclusions stated are those of the author and may or may not be those of the laboratory. Prepared for U. S. Department of Energy under contract No. W-7405-Eng-48.

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

TABLE OF CONTENTS

TABLE OF CONTENTS	PAGE NO.
Abstract	1
1. Introduction	2
1.1 Program Requirements	3
1.2 Process Communication	6
1.2.1 Pseudo Teletypes (PTY)	6
1.2.2 XCNTL Conventions	8
1.3 Parameter Searching	12
1.3.1 Input of Control Information	12
1.3.2 Computation of New Parameters	13
1.4 Use of XCNTL Routines in XTCTMP	17
2. Description of Control Routines	21
2.1 Program Chart for the XTCTMR Controller	21
2.2 XCNTL Routines	22
2.3 Availability of XCNTL Routines	28
3. Running XTCTMP	30
3.1 Loading and Execution	30
3.2 Format of Data in XDATAI	32
3.3 Example LOG File	33
3.4 Description of LOG Messages	34
3.5 Sample Results	35
Acknowledgments	36
Figure 1	37
Figure 2	38
Figure 3	39
Appendix A - References	40
Appendix B - Source Listings	41

NOTICE

This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Department of Energy, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights.

269

ABSTRACT

FORTRAN-callable subroutines have been written to enable controller-controller interaction on the LLL DECSys-10. These subroutines have been used to construct a controller (XTCTMR) for a tandem-mirror physics code (CTCTMR). A description of the subroutines and their use is presented. Also, sample results are given.

"Work performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore Laboratory under contract number W-7405-ENG-48."

Controller Routines for the DECsystem-10

With Application To

A Tandem-Mirror Plasma Code

CHAPTER 1

INTRODUCTION

This document discusses the design and implementation of a general set of program control routines for the DECsystem-10 computer. These control routines (XCNTL) have been used to create a controller program (XTCTMP) for the two-component, tandem-mirror, physics model (CTCTMR). This document is broken into three chapters. In the remainder of Chapter 1, we present an overview of the control routines and how they are used in XTCTMR. Specifically, we explain the design considerations in XCNTL for process communication and parameter searching. Chapter 2 gives a detailed description of the implementation of the control routines, including a description of XCNTL variables and subroutines. Chapter 3 concerns the use of the XTCTMP program. In this chapter, we outline 1) how to compile, load, and execute the controller, 2) how to set up the necessary control inputs, and 3) how to interpret XTCTMR output. Example results will be discussed also for actual runs.

1.1 PROGRAM REQUIREMENTS

The CTCTMR program[1] has been written to study the plasma parameters of various tandem-mirror devices. Input to this FORTRAN program is a FAMELIST that contains various device parameters. With this data, CTCTMR simulates in time certain state variables (e.g. ion temperature and density, total fusion power) within the tandem-mirror model. The simulation continues until the state variables reach steady-state conditions or for a specified maximum time. The steady-state values of the state variables or quantities computed from them (e.g. fusion power) are the output that determine the success or failure of the particular set of design parameters. Unfortunately, the process of choosing the right set of input parameters for the desired set of steady-state conditions can be a very tedious chore, especially if three or more conditions are to be met. For this reason, it is desirable to automate the searching process, allowing the computer to calculate the input to the CTCTMR program. Due to the present size and complexity of CTCTMR, a second program is desired to control the physics model. This controller should communicate with the physics model, interpret the results of the CTCTMR run, calculate a new set of input parameters, and give these parameters to CTCTMR so it can compute a new set of results.

Consider an example. We wish to adjust the gas-feed current density in the central cell of a tandem mirror to meet the criterion that the flux for stabilizing the plugs against

the drift-cyclotron loss-cone mode equals the total loss out each end. The input file to our CTCTMR controller would include 1) an indication that gas-feed current density (variable SPCURJ) is to be the independent variable, 2) an indication that the difference of the two fluxes (variable CURDIF) is to be the dependent variable, 3) the value desired for CURDIF (0.), and 4) an acceptable error in CURDIF. The controller would interpret this input file, generating a series of inputs for execution by its controllee, CTCTMR. The results of each run would be used by the controller to generate a new value of the gas-feed current density to be used in the next CTCTMR run. Eventually the controller should converge to the value of the current density which produces the desired, steady-state condition for the dependent variable. At this point the controller would terminate. In this example, we varied only one independent variable to obtain a value for one dependent variable. However, it is conceivable that we would like to vary several independent variables simultaneously to obtain specific values for corresponding dependent variables.

On the MFE7600, FORTLIB or ORDERLIB support FORTRAN-callable routines for controller/controllee interaction. However, on the DECsystem-10 similar facilities were non-existent. For this reason a general-purpose set of FORTRAN-callable "controller/controllee" communication subprograms (XCNTL) was created. This library was then used to construct the controller (XTCTMR) for the CTCTMR program.

Several requirements were placed on the proposed XCNTL library. The controller would be placed in some sort of "inactive" state while waiting for the controllee to respond. For those routines related to parameter calculation, a symbolic and flexible mechanism for the input and processing of control information was required. Since the controlled program might take several minutes to process one set of input, the parameter calculation routine(s) required a rapidly convergent algorithm. Moreover, the algorithm had to be general enough to permit systems of two or more independent variables that could be used to simultaneously manipulate a corresponding set of dependent variables. Speed of the algorithm was not a design goal, but a short implementation time required that the method be simple. Also, these routines should be able to handle the case where convergence to a solution is not possible (an extrema point).

Roughly then, the design of the control routines can be broken into two distinct areas. The first involves the design of those routines related strictly to controller/controllee communication. The second part of the design concerns those routines related to input and processing of control parameters in the controller. It should be noted that the XCNTL routines are intended for control of programs requiring extensive communication with their controlling jobs. Simple job control is better handled by BATCH system available on the DECsystem-10.

1.2 PROCESS COMMUNICATION

1.2.1 Pseudo Teletypes (PTY)

The mechanism used in the YCRTL routines for inter-process communication is a device known as a Pseudo-teletype (PTY)[2]. The use of the system's Inter-Process Communication Facility (IPCF) was rejected because of the amount of extensive, low-level programming that would be involved to create the necessary general-purpose routines. In contrast, the PTY device is available for use in DECsystem-10 FORTRAN, and is accessed similarly to read-write disc files. With respect to the PTY it is possible to define what is meant by the terms, "controller" and "controllee." A controller is a program that has equipped a PTY device and a controllee is what is currently active on the PTY (a user program or the monitor).

By equipping a PTY device, a program in essence has its own teletype. Suppose KLUN is the logical unit number that the user has provided in a ASSIGN statement with a PTY as the specified device. Writing to the PTY is achieved by statements of the form: WRITE(KLUN,<format>) etc.. Reading from the PTY is performed by statements of the form: READ(KLUN,<format>) etc.. (The controllee communicates with the controller by using the READ and WRITE statements with the logical unit=5.) A crude controller, then, might take the form of a program loop, with the controller program continuously reading from its controllees and branching to various

processing routines depending on the content of the input. However, it is wasteful of system resources to have the controller constantly polling its PTY. Instead, the controller should become "inactive." To make the program in control of the PTY become inactive, a monitor call to the HIBER routine[2] is used. Using the HIBER call, the controller can be put in a suspended (SLEEP) state indefinitely or for a specified amount of milliseconds less than about a minute. The controlling job can be set to be awakened on any PTY activity that occurs during the sleep period or before the last call to HIBER.

Unfortunately, there are several problems to overcome when using a PTY via FORTRAN. If the controller READs from a PTY, but the PTY doesn't have a message to send, then the controller will "hang." Because any teletype activity (e.g. use of FORDDT) or system message can wake the controller, hanging the controller is always a danger. A second phenomenon is the "infinite sleep," where the controller misses a message from the controllee and the controller has been set to sleep indefinitely until it gets a message from the controllee. This can occur if the controller gets too many messages at once from the controllee. While it is true that the HIBER operation can detect PTY activity since the last call, it unfortunately does not record how many lines of PTY activity has occurred. Third, the communication to the PTY cannot be of the form: WRITE(KLUN,XNAME), where XNAME is a NAMELIST.

Also, a couple of problems appear because of the queueing of messages to/from the controllee. If the controllee, for example, is waiting in monitor mode but has messages outstanding to the controller, a command from the controller, say, to start the CTCTMP program on the PTY will not take effect on the PTY until the controller reads all of the outstanding messages from the PTY. Once again, the PTY will not accept the command until the PTY's messages have been read out of the message queue by the controller. Likewise, a user program running on a PTY will not be affected by a double control-C message from the controller unless the PTY program attempts to read from the controller. Finally, a PTY is not truly closed until every message from the PTY has been read by the controller from the message queue. Clearly then, there is always the danger of either hanging the controller or having a detached job (the PTY that "got away") or both. These problems are solved for the user by the utility routines provided in XCNTL, provided the user follows a specified set of conventions to be discussed next.

1.2.2 XCNTL Conventions

From the above discussion it is clear that any programming for controller/controllee communication must provide synchronization of messages and accurate status checking of the controllee. In the design of XCNTL routines, it is assumed the controllee will provide the synchronization message before waiting for output from the controller.

Furthermore, it is assumed the first ten characters of this synchronization message are all that the XCNTL routines will process. Any "message-prefix" must be known to the control routines as well as the logical unit of the PTY for which the "prefix" is valid. To this end, XCNTL provides the routine XMESSW to store the allowable prefixes for a given PTY. Moreover, it provides a schema for automatic generation and storage of prefixes. XCNTL assumes three types of synchronization messages for a given program. One type (X) specifies the controllee is about to request input from the controller. Another type (E) is used when the controllee reports an error and the last type (C) is used when the controllee reports the completion of the user program. If "T" is the type of the synchronization message and NNNNNN (e.g. XTCTMR) is the name of the user program running as the controllee, then the XCNTL-style prefixes are of the form: "NNNNNN:T". If the user wishes to use XCNTL-style messages, then the controller program can call routine XEATCH to store the prefixes it expects from controlled program, using the name of the program as an argument. From the controllee's side, routines XSYNC, XERROR, and XFYIT can be used to generate the messages to the controller. (The program's name is provided as an argument to the XCNTL initialization routine, XCLEAR.)

While it is mainly the responsibility of the controllee to provide synchronization, it is the controller's duty to accurately know the status of its PTY's. Furthermore, the

status checking should be internal to XCNTL. In the design of XCNTL, it is assumed that routines are provided to log-in and log-off a PTY for the user on the logical unit of his choice (see XLGIN and XLGOFF). It is also assumed that the user will call a routine to wait for monitor mode (see XMODEM) after detecting completion of the controlled program. All message processing for the user's controllee is handled by a single routine (see XWAITM) which has as input the PTY to be checked, the number of messages to accept before giving up, and the sleep interval between device checks. On output it is assumed that an index will be returned to indicate which message was actually received or to warn that monitor mode or give-up was detected. This index can be used in a computed GOTO statement to branch to different processing locations according to the message actually received.

The current message handling routines in XCNTL use the monitor call, JOBSTS, to obtain status information about the desired PTY. By checking the status of the PTY, the PTY pitfalls mentioned above (except regarding the communication of NAMELIST's) can be avoided. Although XCNTL effectively allows a controller to process messages from no more than one controllee at a time, the XCNTL conventions allow long controller/controllee chains to be created. However, one caution should be made for use of the message processor routine, XWAITM. After starting a user program controllee, the first return from XWAITM will be for monitor-mode detection. This is because a "." message is sent to the

controller after the controller sends its monitor mode command. As was discussed above, the command will not be read until all messages FROM the controllee have been read. To accommodate for this peculiarity, the controller should simply loop on monitor-mode exit in the first call to XWAITM after initializing a program on the controllee.

It is recognized in the design of XCNTL that the synchronization between the controller and the controllee may fail due to user programming error or to system peculiarities. Unfortunately, most of the burden for preventing these failures rests on the user programmer, and for this reason we have designed the XCNTL routines to be as simple as possible. However, XCNTL does try to avoid "bad" failure states. In particular is the case where XTCTMP is running under BATCH and is waiting for a response from CTCTMP at the same time the controllee is waiting for action from XTCTMR. What makes this situation "bad" is that it requires operator intervention to kill the XTCTMP BATCH job. XCNTL avoids this situation by forcing the controller to READ whenever an input-wait state of the controllee is detected. If a message is not pending from the controllee, the controller will hang in a PUN state, eventually reaching its time limit when it will be killed by the BATCH system. The controllee at this point will be left in a DETACHED state and may be killed by the user.

1.3 PARAMETER SEARCHING

1.3.1 Input Of Control Information

A major concern in the design of the XCNTL routines was the problem of creating a set of general-purpose control routines that were flexible yet independent of the data structure used by the controller and its controllee. The assumption was made that a controller's data structure would have as a subset the data structure used by its controllee. For example, the data structure for the CTCTNR program, which is contained in a cliché file TNR5.CLI (the cliché file having as its contents the various PARAMETER, COMMON, EQUIVALENCE, and other declaration statements that are global to all CTCTNR routines), is also included as a part of the XTCTMP program. The problem then is how to manipulate the data structure of the controller in a flexible way without including the data structure in the XCNTL routines. Without separating data structures, the control routines would have to be re-designed for each new controller.

The solution to this problem used in the XCNTL design is to input SYMBOLIC control information and require the user to supply a subroutine that establishes the correspondence between symbolic names and actual variables in the controller/controller data structure. To this end the user of XCNTL must provide a subroutine called XASSOC which has no arguments. The contents of XASSOC should basically consist of calls to the XCNTL routine, XPAIR, which has as arguments both

the actual variable name the user wishes to control and the symbolic name that the control routines will use to reference it.

In XCNTL, parameter control is specified by the input of one or more "control-sets." Part of the specification of a control-set consists of the specification of an independent variable that will be varied to achieve a specified value for a specified dependent variable. In the particular case of XTCTMR, the actual variable symbolized by "AXISL(1)" may be the independent variable which is varied by the controller to obtain the desired value for a variable symbolized by the name, "UNLOAD(1)". Besides control-sets, another part of the input of control information includes those filenames that the controllee may require. In the case of CTCTMP, certain filenames must be specified for the program's print and plot files. Both "control-sets" and filenames are input to the XCNTL routines via the subprogram XDATAI. Also, XDATAI allows the user to set certain XCNTL parameters initialized by XCLEAR. The format of the "control-file" is discussed in the next chapter as part of the description of the XDATAI routine.

1.3.2 Computation Of New Parameters

Given that the method for input of control information is flexible enough, the next problem is how to best find a solution to the system of control-set. The assumption is made in XCNTL that the controller will call a single routine

called XCOMPN to change the independent variables, making the "best" guess for achieving the desired solution. When this routine finds a solution to its liking or detects a relative extrema, it is assumed that the routine will set its single, logical argument .TRUE. as a stop flag to be interpreted by the controller. Of course, the convergence criterion must be specified. Part of each control-set is the desired value of the dependent variable and an epsilon value for the relative difference allowed for the solution. If convergence is not detected by XCOMPN, it is assumed that the controller will start the controllee with a new set of data, including the new values of the independent variables set by XCOMPN. In the case of the program CTCTMR, the new parameters are written by the controller in NAMELIST format into a file that is then used by the controllee.

The present version of XCOMPN uses Newton's Method for the iterative solution of systems of non-linear equations to solve for new values of independent variables. This algorithm is an extension of the simple case where $X(\text{new}) = X(\text{old}) - F(X(\text{old})) / F'(X(\text{old}))$. In the general case however, F' is a matrix reflecting the gradient at $X(\text{old})$. To estimate F' involves examining the effect on F as each element of $X(\text{old})$ is varied individually by a small percentage of its base value. If N is number of control-sets, then $N+1$ controllee runs are required to calculate the gradient matrix, which can then used to calculate another guess for the set of independent variables. In the case of CTCTMR where each new

set of parameters may take several minutes to run, use of Newton's Method could have the controller running for hours before converging to a solution. However, these fears have yet to be realized. For CTCTMR, convergence for four simultaneous control-sets has taken no more than five or six major iterations. Furthermore, those runs involving small perturbations in the set of independent variables tend to take very little execution time. Since XTCTMR instructs CTCTMR to use all the final conditions (except those changes made by XCOMPEN) from the previous run, the pertubated parameter set converges very quickly to a steady-state solution.

Within XCOMPEN, both independent and dependent variables are normalized to prevent OVERFLOW or UNDERFLOW errors when solving for the next guess for the independent variables. Normally, dependent variables are normalized to the convergence value the user has specified in the control-set. However, for various reasons the user may change the normalization constant by providing the normalization constant as an option in the control-set specifying the dependent variable. Also provided in the control-set specification is a means to limit the change in any independent variable for a single major iteration. The change is limited to the interval $[X/V, X*V]$ where X is the value of the independent variable and V is input (default 10).

In the case of CTCTMR, it would be useful to find the maximum value for a dependent variable (e.g. C). Theoretically, XCOMPEN should be able to find the necessary

value for the independent variable(s) if the convergence value for the dependent variable is set very large. The problem is that Newton's Method runs into trouble around extrema points. This problem can be understood by examining the simple case of $N=1$, where N is the number of independent-dependent variable pairs. If the current guess is near the extrema point (where the slope of the curve goes to zero), then the next guess for the solution will be far away from the extrema point. Although the input can be adjusted to prevent wild oscillations about such points, some method of detection had to be provided in XCOMP. In the simple case of a single control-set, such a point could be detected if we tried to find a solution to a quadratic fit of data obtained from previous iterations. An extrema point would then be detected by the presence of complex roots. However, for the case of many control-sets, this approach cannot be extended. In XCOMP, an extrema point is detected by an increase in the mean-square norm for the vector consisting of the relative difference of the dependent variables from the desired solution. If this value does increase, then XCOMP makes a guess at the extrema point before signalling a stop. This guess is a weighted average of the previous two iterations, where the weight is the inverse of the calculated normal value. It is probable that a better scheme for determination of extrema points will eventually replace the method described above.

1.4 USE OF THE XCNTL ROUTINES IN XTCTMR

In this section we will describe how the control routines outlined earlier are used to implement the CTCTMR controller, XTCTMR. Please refer to the source listing of the controller in Appendix P. The data structure of the CTCTMR program is contained in the cliché file, TMP5.CLI. This same structure is included also in the controller. Since the controller uses the same data structure, it is possible to use routines in the controller that are borrowed from the controller's library. Currently, program XTCTMR uses a single routine, GETMES, from the CTCTMR library, PDPLIB.REL[1]. Subroutine GETMES is borrowed from PDPLIB.REL to enable the XTCTMR program to receive messages from its controlling job (e.g. the user's teletype).

Input to the CTCTMR program includes: 1) the name of the input file containing the parameter sets to be processed, 2) the output print-file name, and 3) the output plot-file name. If flag NLECON=.TRUE. within an input parameter set, CTCTMR also requires a filename for "economics model" output. The parameter set used by the CTCTMR program basically consists of a FORTRAN NAMELIST. For control of CTCTMR, it is desirable to manipulate, via XTCTMR, the NAMELIST input that is read by CTCTMR. Since a PTY is not capable of reading a NAMELIST output directly from its controller, it is necessary that XTCTMR and CTCTMR pass shared files. This sharing of files requires synchronization between the two programs so one does not try to access the file in common while the other has it

opened. For the XTCTMR programs, the common file has the name, NLCNTL.DAT.

The first part of the XTCTMP program is initialization. XCLEAR is called first to ready the XCNTRL routines, then XDATAI is called to input the control-file. Input of the control-file is designated on logical unit number 25. Part of this control-file contains the output filenames required by CTCTMR (in local variables INMFLE(2-4)). The base parameter set is in file INMFLE(1). The contents of this file is copied into NLCNTL.DAT, though with alterations to the values of parameters NLCNTL and NLCONT (for reasons that will be discussed below).

By label 2010 in the program, XTCTMP is ready to start the controllee. First, XTCTMR calls XWATCH to accept XCNTRL-style messages from program CTCTMR. NLCIN logs a PTY on logical unit number 26, after which the main program writes a message to monitor mode running on the PTY requesting that program CTCTMR be run. The program from labels 2020 through 2045 concerns the interaction necessary with the controllee to pass the input filename (NLCNTL.DAT) and output filenames required by routine PAIN in CTCTMR. At this point, XTCTMP is ready to accept parameter sets. XTCTMP has set flag NLCNTL=.TRUE. in NLCNTL.DAT to let CTCTMR know it is being controlled, and has also set flag NLCONT=.TRUE. so CTCTMR does not clear final values before requesting another parameter set. DATA (within XTCTMP) is programmed so when NLCNTL is set .TRUE., it will write a synchronization message

(via a XSYNC-like routine) to the controlling job (XTCTMP). XTCTMP will wait for this message (via XWAITM) and when a message is received, XTCTMP will send an reply back to CTCTMP when the controller wants the controllee to proceed with another parameter set. Note that this reply is unnecessary for the first parameter set to be executed.

From labels 3010 through 3030 in XTCTMP, file NLCONTL.PAT is passed back and forth with CTCTMP. The first time through the loop, DATA (within CTCTMP) prints the synchronization message, reads the parameter file NLCONTL.PAT, and begins execution immediately. During this time CTCTMP is executing the parameter set, XTCTMP makes the first, dummy call to routine XCOMIN and loops back to await a response from the controllee. When CTCTMP finishes the processing of the parameter set, it calls DATA to read another set. Interpreting parameter NLCONTL, DATA writes the results of the simulation into NLCONTL.PAT, closes this file, sends the synchronization message, and then waits for the controller to send the name of the next input filename. When XTCTMP receives the sync message for which it has been waiting, the common file is opened and the simulation results are read. XCOMPN is called to compute new parameters. Note that XTCTMP is programmed to maintain a constant ratio for plug axis length to plug radius. After computing new parameters, NLCONTL.PAT is rewritten and closed. When this is done, XTCTMP sends CTCTMP the filename for which it was waiting, then waits again for the simulation to finish. This cycle continues

until XCOMPN sets the stop flag, NLSTOP=.TRUE..

When NLSTOP is set .TRUE., XTCTMP checks if the economics flag, NLECON, has value .TRUE.. If so, CTCTMP will request the filename for this output. Otherwise, XPYIT is called to logoff all PTY's under control of XTCTMP and exit to the system. Note first that if for some reason there are errors in synchronization, YTCTMF calls XEPRON to terminate XTCTMP and CTCTMP. Second, the meanings of the control variables input to XDATAI have been specified by the controller in routine XASSOC. It is helpful to keep in mind when examining the flow of XTCTMR that though XTCTMR and CTCTMP have similar data structures, they do not share variables. A variable in XTCTMR may not have the same value as the variable in CTCTMP with the same name. This is probably the major source of confusion in the creation of a controller like XTCTMR.

CHAPTER 2

DESCRIPTION OF CONTROL ROUTINES

Source listings for the XCNTL routines and the XTCTMP controller are available in Appendix B.

2.1 PROGRAM CHART FOR THE XTCTMP CONTROLLER

XCLEAR: Initialize the XCNTL routines.
YDATAI: Input the control-file.
XSYNC: Request to the controller.
CETMES: Get any TTY messages.(PDPLIB)
XERROR: Indicate error to controller.
XEXIT: Terminate all controllees and signal completion to controller.
YSYNC,XERROR
XWATCH: Place XCNTL-style message-prefixes into the message table for a specified program on a specified PTY.
XMESSW: Place message-prefix into the message table.
XLCIN: Log in a PTY on a specified logical unit.
XOPEN: Enter PTY into active-job tables.
XWAITT: Suspend controller for a specified time or until PTY activity is detected.(PDPLIB)
XPTSTS: Check status bit for specified PTY.
XCTSTS: Get status word for specified job.(PDPLIB)
XERROR
XWAITM: Wait for acceptable response on the specified PTY.
XWAITT,XPTSTS

YCOMPN. Compute new values of independent variables.
 XQUEUE: Store the results of the iteration.
 YASSOC: Establish correspondence between symbolic
 names and actual control variables.
 XPAIR: Correspondence between symbol and actual
 control variable.
 XMODEM: Wait until specified PTY is in monitor mode.
 XWAITT,XPTSTS
 XEXIT:
 XLOGOFF: Log-off the specified PTY.
 XCLOSE: Check if PTY is logged on the lun.
 XFIN: Clear active-job tables.
 XERROR
 GETMES

2.2 XCNTL ROUTINES

Name: XCLEAR
 Calling Sequence:
 CALL XCLEAR(KNAME,KPPN)
 Input:
 KMITER: integer, scalar
 KLCMO: logical, scalar
 KNAME: character, double precision scalar, e.g. "XTCTMR"
 KPPN: character, double precision scalar, e.g. "30/3046"
 Output:
 Note: XCLEAR initializes the various tables of XCNTL. KNAME
 specifies the program name to be used by routines XSYNC,
 XERROR, and XEXIT when writing to the controlling job. KPPN
 is the user's ppn which is used to log on the PTY. XCLEAR
 should be called before any other XCNTL routine is used.

Name: XLGIN
 Calling Sequence:
 CALL XLGIN(KLUN)
 Input:
 KLUN: integer, scalar
 Output:
 Note: Log on a PTY and assign it to logical unit number KLUN.

Name: XMODEM
 Calling Sequence:
 CALL XMODEM(KLUN)
 Input:

KLUN: integer, scalar

Output:

Note: Read messages from the PTY assigned to logical unit KLUN until an input wait state in monitor mode is detected.

Name: XLCOFF

Calling Sequence:

CALL XLCOFF(KLUN)

Input:

KLUN: integer, scalar

Output:

Note:

Terminate the PTY assigned to logical unit KLUN.

Name: XOPEN

Calling Sequence:

flag=XOPEN(KLUN,INDJOB)

Input:

KLUN: integer, scalar

Output:

XOPEN: logical, scalar

INDJOB: integer, scalar

Note: Internal routine used by XLCIN. If a PTY assigned to logical unit number KLUN is logged in, then XOPEN returns with the value TRUE. If not, XOPEN returns with the value FALSE and INDJOB contains an index to be used by XLCIN to fill in the job-number table, NJOBTB.

Name: XCLOSE

Calling Sequence:

flag=XCLOSE(KLUN)

Input:

KLUN: integer, scalar

Output:

XCLOSE: logical, scalar

Note: Internal routine used by XLCOFF. If the PTY assigned to logical unit number KLUN is logged off then TRUE is returned as the value for XCLOSE. If KLUN is logged on then XCLOSE returns with value FALSE.

Name: XFIN

Calling Sequence:

flag=XFIN(KLUN)

Input:

KLUN: integer, scalar

Output:

XFIN: logical, scalar, =TRUE

Note: Internal routine used by XLCOFF. YFIN clears the internal tables for the completed PTY assigned to logical unit KLUN.

Name: XEXIT

Calling Sequence:

CALL XEXIT(KLUN)

Input:

KLUN: integer, scalar

Output:

Note: XEXIT sends a program-completion message to the controller beginning with the characters "nnnnn:C", where nnnnn is the name of the controllee program declared in XCLEAP. All PTY's under the program's supervision are logged off. KLUN is any logical unit number the user wants to convey to the controller.

Name: XERROR

Calling Sequence:

CALL XERROR(KNAME1,KNAME2,KLUN,KLFLAG)

Input:

KNAME1,KNAME2: character, double precision scalar

KLUN: integer, scalar

KLFLAG: logical, scalar

Output:

Note: XERROR sends a error message to the controller beginning with the characters "nnnnn:E", where nnnnn is the name of the controllee program declared in XCLEAP. Argument KNAME1 is typically the name of the subprogram that XERROR is being called from, and KNAME2 is an indication of the reason for error message. KLUN is any logical unit number the user wants to convey to the controller. If KLFLAG is TRUE, then XERROR will return to the calling subprogram. If FALSE, then XEXIT will be called.

Name: XSYNC

Calling Sequence:

CALL XSYNC(KNAME1,KNAME2,KLUN)

Input:

KNAME1,KNAME2: character, double precision scalar

KLUN: integer, scalar

Output:

Note: XSYNC sends a synchronization message to the controller

beginning with the characters "nnnnn:X", where nnnnn is the name of the controllee program declared in XCLEAP. Argument KNAME1 is typically the name of the subprogram that XSYNC is being called from, and KNAME2 is an indication of the reason for synchronization message. KLUN is any logical unit number the user wants to convey to the controller.

Name: XWAIT

Calling Sequence:

CALL XWAIT(KLUN, KINDEX, NOWAIT, PRUFF, KTIME)

Input:

KLUN: integer, scalar

NOWAIT: integer, scalar

KTIME: integer, scalar

Output:

KINDEX: integer, scalar

PRUFF: character, double precision dim=8

Note: Wait for a message on the PTY assigned to logical unit KLUN. If the first ten characters of the message correspond to one of the entries in the message-prefix table for KLUN, then return an index, KINDEX, relating to which prefix was found. If the Ith call to XMESSV is CALL XMESSV(KNAME) then if a message is received beginning with KNAME, KINDEX=I+2. (XWAIT calls XMESSV three times with "nnnnn:X", "nnnnn:E", and finally "nnnnn:C".) If NOWAIT=1 messages have been received from the PTY without a table match, then XWAIT returns with KINDEX=1 (If NOWAIT=0 then the routine will sample indefinitely.) If monitor mode is detected then the routine will return with KINDEX=2. The sleep time between message sampling is determined by KTIME (see XWAIT for argument limitations). On return from XWAIT, PRUFF will contain the first 80 characters of the last message received from the PTY. Note that if the controllee is found waiting for input from its controller, the controller will read from the controllee's PTY whether or not there is actually a message waiting to be read. For this reason synchronization messages to the controller should be sent BEFORE requesting input from the controller.

Name: XMESSV

Calling Sequence:

CALL XMESSV(KLUN, KNAME)

Input:

KLUN: integer, scalar

KNAME: character, double precision scalar

Output:

Note: Internal routine used by XWAIT. Put the message KNAME into the message-prefix table for the PTY assigned to logical unit number KLUN. (The parameter TABSIZ control the size of this table.)

Name: YWAITT
 Calling Sequence: CALL YWAITT(KTIME)
 Input:
 KTIME: integer, scalar
 Output:
 Note: Internal routine used by XLGIN, XNODEN, XLCOFF, and XWAITM. The program will hibernate for KTIME milliseconds or until the system wakes the job (e.g. controller gets a message from an active PTY). KTIME must be .LE.60000 ; KTIME=0 indicates an INFINITE sleep.

Name: XPTSTS
 Calling Sequence: flap=XPTSTS(KLUN,KFIELD)
 Input:
 KLUN: integer, scalar
 KFIELD: integer, scalar
 Output:
 XPTSTS: logical, scalar
 Note: Internal routine used by XLGIN, XNODEN, XLCOFF, and XWAITM. If the status bit at bit KFIELD is on for the PTY assigned to logical unit number KLUN, then return a value TRUE for XPTSTS. If zero, return a value FALSE.

Name: XGTSTS
 Calling Sequence: CALL XGTSTS(KJOB,KSTAT)
 Input:
 KJOB: integer, scalar
 Output:
 KSTAT: integer, scalar
 Note: Internal routine used by XPTSTS. Return the status bits in KSTAT for the job numbered KJOB. Each bit in the status word returned in KSTAT reflects the state of a certain aspect of the job (e.g. the job is logged in, the job is waiting for input, etc.). If an error occurs return zero to KSTAT.

Name: XWATCH
 Calling Sequence: CALL XWATCH(KLUN,FNAME)
 Input:
 KLUN: integer, scalar
 FNAME: character, double precision scalar
 Output:
 Note: XWATCH puts the XCNTL-style messages for the program

named KNAME into the message-prefix table for controllees running the PTY referenced by logical unit KLUN.

Name: XDATAI

Calling Sequence:

CALL XDATAI(KLUN,KNARR)

Input:

KLUN: integer, scalar

Output:

KNARR: character, double precision dim=6

Note: XDATAI is used to input parameter-control and filename information for use by XCONT. (This information is echoed back to the controlling program, provided that flag FLECHO has value .TRUE. after the input of control information.) The user input file containing this control information must be equipped to logical unit number KLUN. The first line of input (312,14,2A10,A5) contains 1) the number of control-sets and 2) the number of filenames (maximum of six). These first two parameters are required while the remaining parameters of this first line of input are optional. The remaining parameters change the default settings of internal variables set by routine XCLEAR. These parameters are 1) IMXFT, the number of re-tries for the calculation of a detected extrema point (default 1), 2) MXITER, the maximum number of major iterations of the routine XCOMP (default 30), 3) NAMPPN, the user ppn number, 4) NAMPRG, the program name, and 5) FLECHO, a flag (default .TRUE.) for controlling the echoing of controllee messages back through the control program to its controlling job (e.g. the user's teletype). Next follow the control sets, one per line (2A10,4F10.0). Each control-set contains the symbolic independent variable, the symbolic dependent variable, the convergence value, the accuracy epsilon, the normalization value for the dependent variable (if zero, then the convergence value is used), and the factor of maximum and minimum change in the independent variable (default 10). The last record of the input file should contain the filenames for use by the controller (8A10). These filenames are returned in array KNARR.

Name: XCOMP

Calling Sequence:

CALL XCOMP(KLSTOP)

Input:

Output:

KLSTOP: logical, scalar

Note: XCOMP computes the next set of parameters to be run by the controllee. Newton's Method for the iterative solution of a system of equations is used. If XCOMP detects a relative extrema or convergence to a solution, then input/output flag KLSTOP will be set .TRUE.. XCOMP requires a user-supplied

subroutine XASSOC(no arguments). XASSOC should consist of any number of calls to routine XPAIR, establishing the correspondence between symbolic names and actual variables that the user wishes to control. XCOMPN is subject to replacement as better algorithms are found for parameter calculation. Routines DEC and SOL (PDPLIR.REL) are called by XCOMPN to solve the system of linear equations. Sources for DEC and SOL are available in file PDPLIR.FOR (see the section titled, "Availability of XCNTL Routines").

Name: XPAIR

Calling Sequence:

CALL XPAIR(KNAME,PVAR)

Input:

KNAME: character, double precision scalar

PVAR: real, scalar

Output:

Note: Associate the symbolic name KNAME with the variable PVAR. The user supplied routine XASSOC (no arguments) should make use of XPAIR to establish a correspondence between the symbolic names used by XCOMPN and the actual variable references the user wants to control.

Name: XQUEUE

Calling Sequence:

CALL XQUEUE

Input:

Output:

Note: Not to be called by user. XQUEUE is called by routine XCOMPN to store values of independent and dependent variables.

2.3 AVAILABILITY OF XCNTL ROUTINES

All XCNTL-related files may be obtained from the LLL DECsystem-10 file space for user [30,317] or arranged through R.S. Devoto, ext. 29856. Access may also be arranged through the author: Donald R. Faul, ext. 24277. The source file for the control routines is XCNTL.FOR. Use of XCNTL requires the following routines from the library PDPLIB.REL:

EXPF: Entry points EXPF, ALOCF, ALOG10F, and SORTF
are used instead of direct calls to DECsystem-10
routines EXP, ALOC, ALOG10, and SORT.
DEC,SOL: Matrix decomposition and solution routines.
GETMES,TTY6F: Checks for and reads message from TTY.

CHAPTER 3

RUNNING XTCTMR

3.1 LOADING AND EXECUTION

The source and object-module files needed for XTCTMR are:

```
XCNTL.FOR (.REL) :control-routines
XTCTMP.FOR (.REL) :controller for CTCTMR
PDPLIB.REL :library file
TMR5.CLI :cliche file for CTCTMR
XCNTL.CLI :control-routine cliche file
```

These may be compiled and loaded in the conventional manner. An alternative makes use of Macro Interpreted Command (MIC) files[4]. Three MIC files are available on the LLL DECsystem-10:

```
PULL.MIC[30,317] :extract routines from source file
COMPMX.MIC[30,317] :compile control-routines
LOBXTC.MIC[30,317] :compile/load controller, XTCTMR
```

PULL.MIC is used to extract one or more subprograms out of a source file. For example,

```
.DO PULL XCNTL.FOR,XLGIN,XDATAI
```

will extract routines XLGIN and XDATAI from the

control-routine file, XCNTL.FOR, and place them in a work file. Then COMPMX.NIC can be used to compile this work file and replace the relocatable object modules in file XCNTL.PFL:

```
.DO COMPMX XCNTL.FOR,XLCIN,XDATAI
```

Alternatively, the whole source could be recompiled:

```
.DO COMPMX XCNTL.FOR,ALL.
```

Similarly, LODXTC.NIC may be used to recompile a controller subroutine (after it has been PULled), replace it in the .PFL file, and then create the executable file for the controller:

```
.DO LODXTC XTCTMP.FOR,XASSOC
```

Alternatively, the entire source file for XTCTMP may be recompiled:

```
.DO LODXTC XTCTMR.FOR,ALL.
```

or simply loaded without recompilation:

```
.DO LODXTC XTCTMR.FOR,NONE.
```

or loaded with FORDDT for debug purposes:

```
.DO LODXTC XTCTMR.FOR,DEBUG.
```

After the controller is created, the user may wish to submit a control-file to batch. A sample batch control-file (.CTL) for a job, which will be called TCT00.CTL, might appear as follows:

```
.DEL TCT00.LOG  
.RUN XTCTMR  
TCT00.CON  
.PRINT TCT00.LOG,<any other output files>  
.<Other processing, etc.>
```

where TCT00.CON is the XCNTL control-file and TCT00.LOC is the batch log-file for the job. Example inputs TCT00.DAT and TCT00.CON are available on the file space for user [30,317].

3.2 FORMAT OF DATA IN XDATA1

In the example of the last section, TCT00.CON contained the control information to be read by XCNTL routine, XDATA1. The format for this information may be illustrated by an example for TCT00.CON:

```
2 3
AXISL(1) WNLOAD(1) +1.0E+06 +1.0E-01
RADIUS(2) PWRRAT +1.5E+00 +1.0E-01 +0.0E+00 +2.0E+00
TCT00.DAT TCT00.OUT TCT00.PLT TCT00.ECN
```

The first line of the file specifies that there are two control-sets and that three names (e.g. filenames) are to be input from the last line of the file. The controller can use these names (10 characters maximum) in any way it chooses. In the case of XTCTMP, the names represent the respective filenames required in the normal execution of CTCTMP. The second and third lines of TCT00.CON are the descriptions of the control-sets.

The two independent variables are AXISL(1) (=central-cell length) and RADIUS(2) (=plug radius) while the dependent variables are WNLOAD(1) (=neutron wall load in the central cell) and PWRRAT (=total fusion power/ total input power). The desired values for WNLOAD(1) and PWRRAT are 10^6 and 1.5. In the latter case we are attempting to find the maximum value

for Q. The convergence criterion is 0.1 for both dependent variables while the maximum permitted change in AXISL(1) is 10./0.1 (the default) and in RADIUS(2) IS 2./0.5.

3.3 EXAMPLE LOG FILE

Below is listed the log file for TCT00.CTL:

```
10:39:47 BAJOR BATCON version 13(1071) running TCT00
10:39:47 BAFIL Input from DSKB0:TCT00.CTL[30,3046]
10:39:47 BAFIL Output to DSKB0:TCT00.LOC[30,3046]
10:39:47 BASUM Job parameters
                Time:00:05:00   Unique:YES       Pestart:NO

10:39:47 MONTR
10:39:47 MONTR .LOGIN 30/3046
10:39:49 USER   JOB 29 MDIV-USC LLL 603 VM-AP,C TTY60
10:39:49 USER   [LCNJSP Other jobs same PPN:27]
10:39:49 USER   1039   30-Jan-79       Tue
10:39:58 MONTR
10:39:58 MONTR .DEL 1CT00.LOC
10:40:01 USER   Files deleted:
10:40:02 USER   TCT00.LOC
10:40:02 USER   06 Blocks freed
10:40:03 MONTR
10:40:03 MONTR ..PIN XTCTMR
10:40:03 USER
10:40:05 USER   %FORDDT not loaded
10:40:05 USER   XTCTMR:X Routine:XDATAI   Desc:CNTRLFILE?       Lun: 25
10:40:10 USER
10:40:10 USER   TCT00.CON
10:40:35 USER   26-JOB 39 MDIV-USC LLL 603 VM-AP,C TTY71
10:40:35 USER   26-[LCNJSP Other jobs same PPN:27,29]
10:40:36 USER   26-1040 30-Jan-79       Tue
10:40:56 USER   26-
10:41:36 USER   26-
10:41:50 USER   26-%FORDDT not loaded
10:41:50 USER   26- CTCTMR:X ROPEN          DATAFILE? = 0.00000E+00
10:42:27 USER   26- CTCTMR:X ROPEN          PRNTFILE? = 0.00000E+00
10:42:27 USER   26- CTCTMR:X ROPEN          PLOTFILE? = 0.00000E+00
10:42:27 USER   26- CTCTMR:X DATA          NAMELIST? = 0.00000E+00
10:51:49 USER   26- CTCTMR:X DATA          NAMELIST? = 0.00000E+00
10:51:50 USER   NORMAL: 7.20370E-01
10:51:50 USER   AXISL(1) :3.00000E+01      WNL0AD(1):1.06505E+06
10:51:50 USER   RADIUS(2):2.22000E-01      PWRRTAT :4.23858E-01
11:00:58 USER   26- CTCTMR:X DATA          NAMELIST? = 0.00000E+00
11:00:58 USER   WNL0AD(1):1.06873E+06      AXISL(1) :
11:00:58 USER   PWRRTAT :4.24785E-01      AXISL(1) :
11:00:58 USER   NORMAL: 7.20097E-01
11:08:35 USER   26- CTCTMR:X DATA          NAMELIST? = 0.00000E+00
```

```

11:08:37 USER      WNLOAD(1):1.05219E+06      RADIUS(2) :
11:08:37 USER      PWRRAT :4.23523E-01      RADIUS(2) :
11:08:37 USER      NORMAL: 7.19546E-01
11:14:18 USER 26- CTCTMR:X DATA      NAMELIST? = 0.00000E+00
11:14:18 USER      NORMAL: 7.21175E-01
11:14:18 USER      AXISL(1) :3.00000E+00      WNLOAD(1):7.88339E+05
11:14:18 USER      RADIUS(2):1.11000E-01      PWRRAT :4.65878E-01
11:14:22 USER 26- CTCTMR:E CTCTMR      NLSTOP = 1.00000E+00
11:14:24 USER 26- CTCTMR:C
11:14:24 USER 26-END OF EXECUTION
11:14:24 USER 26-CPU TIME: 10:37.40      ELAPSED TIME: 32:34.37
11:14:45 USER 26-EXIT
11:14:46 USER 26-
11:14:46 USER 26-.Other jobs same PPN
11:14:46 USER 26-Job 39, User [30,3046] Logged off TTY71      1114
11:15:07 USER 26-Another job still logged in under [30,3046]
11:15:27 USER 26-Runtime 10 Min, 38.68 Sec
11:15:27 USER      XTCTMR:C
11:15:27 USER
11:15:27 USER
11:15:27 USER      END OF EXECUTION
11:15:27 USER      CPU TIME: 4.23      ELAPSED TIME: 35:22.18
11:15:28 MONTR EXIT
11:15:28 MONTR
11:15:28 MONTR ..PRINT TCT00.OUT
11:15:30 USER      Total of 208 blocks in 1 file in LPT request
11:15:32 MONTR
11:15:32 MONTR .KJOB DSKBO:TCT00.LOG=/W/E/Z:4/VR:10/VS:10813/VL:200/
11:15:36 K-QUE
11:15:36 K-QUE      ZWLDAFR All files rejected by conditions
11:15:36 KJOB      DSKBO:TCT00.LOG[30,3046]
11:15:36 KJOB      Other jobs same PPN
11:15:36 LGOUT      Job 29, User [30,3046] Logged off TTY60 1115 30-Jan-79
11:15:36 LGOUT      Another job still logged in under [30,3046]
11:15:36 LGOUT      Runtime 6.15 Sec

```

3.4 DESCRIPTION OF LOG MESSAGES

The messages in the log file from USER represent output that would have gone to the user's teletype if the user had run the controller at his terminal. The messages beginning with the characters " XTCTMR:" are written directly by the controller. Messages beginning with "26-" represent controllee output to the controller that has been echoed by the controller (echoing can be turned off in the control-set

read by routine XDATAI). Notice the format of the synchronization messages from both the controller and the program (CTCTMR) it is controlling. The "26" in this case indicates the logical unit number in XTCTMR of the controllee PTY. Note the composition of the synchronization messages. Presented left to right in the message are the name of the program being run by the controllee, the type of synchronization message, the routine in which it was called, as well as a cryptic reason for the message.

3.5 SAMPLE RESULTS

In this section we give results from two problems for which the XTCTMR controller has been used: 1) adjustment of gas feed to the central cell of a tandem mirror (MPTF-B) to stream stabilize DCLC modes in the plug, and 2) adjustment of gas feed (I[T]), high energy (200 keV) injection current into the central cell (I[D]), and central-cell length (L[C]) to obtain ECLC stability, total fusion power (P[F]) = 500 MW, and wall load (Gamma[W]) = 1 MW/m². Results for the first case are shown in Figure 1 as a plot of ABS(Delta[j]), the difference between required stabilization current flux, and actual loss flux, vs central-cell gas current (I[D]). Note that the requested criterion has been satisfied in three steps. The final state has a required stabilizing current flux of 0.662 A/cm² with a relative error due to imperfect adjustment of I[D] of only 0.05%.

Results for the second case, a possible small hybrid tandem mirror reactor, are shown in Figures 2 and 3. The various dependent quantities ($P[F]$, $\Gamma[W]$, and $\Delta[j]$) are shown vs iteration cycle number in Figure 2, while the independent variables ($I[D]$, $I[T]$, and $L[C]$) chosen by the controller, XTCTMR, are given in Figure 3. This case is somewhat atypical since the starting values for $I[B]$ and $I[T]$ were found from a run in which the length was fixed at 50 m and $I[D]$ and $I[T]$ were varied to obtain $\Delta[j] = 0$ and $P[F] = 500$ MW, but $\Gamma[W]$ was only approximately 0.4 MW/m^2 . Though rather large variations in the independent variables occur on the first step, convergence to the desired solution is obtained in seven steps.

ACKNOWLEDGMENTS

The author wishes to thank both Tom Frengle and Neil Maron for their help in creating the YCNTL routines.

Fig. 1. Application of XTCTMR to adjustment of gas feed to central cell to obtain sufficient ion flux to stabilize plugs against DCLC.

Fig. 2. Variation with iteration cycle of dependent variable for a small tandem mirror. P_f = total fusion power; Γ_w = wall neutron flux; $\Delta j = |j_{\text{stab}} - j_{\text{loss}}|$.

Fig. 3. Variation with iteration cycle of independent variables for a small tandem mirror reactor. I_D = high-energy deuterium injection current; I_T - tritium gas feed (equivalent amperes); L_c = central-cell length.

APPENDIX A

REFERENCES

- [1]Devoto, R.S. and Faul, D.R., UCID (in preparation).
- [2]DECsystem-10: Monitor Calls Manual, Digital Equipment Corporation, May 1974.
- [3]DECsystem-10: Utilities Manual, Digital Equipment Corporation, March 1975.
- [4]Hatfield Polytechnic Computer Centre, "Macro Interpreted Commands," Version 5(35), Release 2, 1975.

APPENDIX B
SOURCE LISTINGS

```
PROGRAM XTCTMR
INCLUDE "TMR5.CLI"
```

```
C.... .XTCTMR is the controller for the program CTCTMR.
C.... ...Programmer Donald Paul: December 14, 1978.
C.... ...Last modified: April 16, 1979
C.... ...Version: XT0006.FOR
```

```
DOUBLE PRECISION ZMESS(8),INMFLE(6),NAMCTL
DATA NAMCTL/'NLCNTL.DAT'/
```

```
C.... .Read in control variables and file names.
      LUNCTL=25
      CALL XCLEAR('XTCTMR','30/317')
      CALL XDATAI(LUNCTL,INMFLE)
```

```
C.... .Read data and then write back out to data file nlcntl.dat.
      LUNINP=26
      NAMINP=INMFLE(1)
      OPEN(UNIT=LUNINP,ACCESS='SEQIN',DEVICE='DSK',
*      FILE=NAMINP)
      READ(LUNINP,9020)NAMTIT
      READ(LUNINP,TCCTMR,FRR=8010)
      NLSTOP=.FALSE.
      NLCONT=.TRUE.
      NLCNTL=.TRUE.
      OPEN(UNIT=LUNCTL,ACCESS='SEQOUT',DEVICE='DSK',FILE=NAMCTL)
      WRITE(LUNCTL,9020)NAMTIT
      WRITE(LUNCTL,TCCTMR)
      CLOSE(UNIT=LUNINP)
      CLOSE(UNIT=LUNCTL)
```

```
2010 LUNPRG=26
C.... .Start controllee(ctctmr).
      CALL XWATCH(LUNPRG,'CTCTMR')
      CALL XLCIN(LUNPRG)
      WRITE(LUNPRG,9030)
```

```

C.... .NOW TELL IT NAMES OF DATA FILE(NOTE USE OF NLCTL.DAT TO
C.... ..PASS DATA TO CONTROLLEE)
2020 CALL XWAITM(LUNPRG, INDMES, 10, ZMESS, 30000)
      GO TO (2022, 2020, 2025, 2022, 2022) INDMES
2022 CALL XERROR('MAIN.', 'DATAFILE', LUNPRG, .FALSE.)
2025 WRITE(LUNPRG, 9020) NAMCTL

C.... .AND NAME OF OUTPUT FILE,
2030 CALL XWAITM(LUNPRG, INDMES, 10, ZMESS, 30000)
      GO TO (2032, 2032, 2035, 2032, 2032) INDMES
2032 CALL XERROR('MAIN.', 'PRNTFILE', LUNPRG, .FALSE.)
2035 WRITE(LUNPRG, 9020) INMFLE(2)

C.... .AND NAME OF PLOT FILE.
2040 CALL XWAITM(LUNPRG, INDMES, 10, ZMESS, 30000)
      GO TO (2042, 2042, 2045, 2042, 2042) INDMES
2042 CALL XERROR('MAIN.', 'PLOTFILE', LUNPRG, .FALSE.)
2045 WRITE(LUNPRG, 9020) INMFLE(3)

C.... .Main Section.

      NLSTOP=.FALSE.
      NLCTL=.FALSE.

      IWAIT=30000
      ZRATIO=AXISL(2)/RADIUS(2)
3010 CALL XWAITM(LUNPRG, INDMES, 0, ZMESS, IWAIT)
      GO TO (3012, 4010, 3015, 3010, 4010) INDMES
3012 CALL XERROR('MAIN.', 'DATASYNC', LUNPRG, .FALSE.)

C.... .PICKUP ANY MESSAGES FROM TTY
3015 CALL GETMES(IMESS)
      IF (IMESS .EQ. 'STOP') NLSTOP=.TRUE.

      IF (.NOT.NLCTL) GO TO 3020
      OPEN(UNIT=LUNCTL, ACCESS='SEQIN', DEVICE='DSK', FILE=NAMCTL)
      READ(LUNCTL, 9020) NAMTIT
      READ(LUNCTL, TCTMR, ERR=8010)
      CLOSE(UNIT=LUNCTL)

      CALL XCOMPX(NLSTOP)
      OPEN(UNIT=LUNCTL, ACCESS='SEQOUT', DEVICE='DSK', FILE=NAMCTL)
      NLCONT=.TRUE.
      AXISL(2)=ZRATIO*RADIUS(2)
      WRITE(LUNCTL, 9020) NAMTIT
      WRITE(LUNCTL, TCTMR)
      CLOSE(UNIT=LUNCTL)

C.... .WRITE MESSAGE TO CONTROLLEE TO WAKE IT UP
      WRITE(LUNPRG, 9020) NAMCTL
      GO TO 3030

3020      NLCTL=.TRUE.
      IWAIT=60000
      CALL XCOMPX(NLSTOP)

3030 IF (.NOT.NLSTOP) GO TO 3010

```

```

      IF (.NOT.NLECON) GO TO 4010
3035   CALL XWAITM(LUNPRC, INDMFS, 10, ZMESS, 30000)
      CO TO (4010, 4010, 3040, 3035, 4010) INDMFS
3040   WRITE(LUNPRC, 9020) INMFLE(4)

C.... .End of Job.

4010 CALL XMODEM(LUNPRC)
      CALL XEXIT

F010 CALL XERROR('MAIN.', 'DATAERR', LUNPRC, .FALSE.)

C.... .Formats.

9020 FORMAT(2A10)
9030 FORMAT('RUN CTCTMR')
      END
C.... .XTCTMR.

```

```

SUBROUTINE XASSOC
INCLUDE 'THR5.CLI'

```

```

C.... .Associate symbolic names with TCTMR variables.
C.... ...Programmer Donald Faul: December 14, 1978.

```

```

      CALL XPAIR('PWRFUS', PWRFUS)
      CALL XPAIR('PWRRAT', PWRRAT)
      CALL XPAIR('SPENFR(2)', SPENFR(2))
      CURDIF=CURLSS(NCELL2) -CURSTB(NCELL2)
      CALL XPAIR('CURDIF', CURDIF)
      CALL XPAIR('SPCURJ(1)', SPCURJ(1))
      CALL XPAIR('SPCURJ(2)', SPCURJ(2))
      CALL XPAIR('RTFLD(1)', RTFLD(1))
      CALL XPAIR('WLOAD(1)', WLOAD(1))
      CALL XPAIR('AXISL(1)', AXISL(1))
      CALL XPAIR('RADIUS(2)', RADIUS(2))
      CALL XPAIR('SPRFFC(4)', SPRFFC(4))
      RETURN
      END
C.... .XASSOC.

```

```

C*****
C****
C****
C****      XCNTL ROUTINES--FOR PTY CONTROL
C****      PROGRAMMER: D.R. PAUL
C****      ORIGINATED: DECEMBER 8, 1978
C****      TARCET MACHINE: LLL PDP10
C****
C****
C*****

```

C.... .GENERAL DESCRIPTION OF ROUTINES IN BINARY:

```

C.... ...XPAIR: PAIR SYMBOL WITH ACTUAL VARIABLE.
C.... ...XLCIN: LOG IN TO PTY
C.... ...XLOGOFF: LOG OFF A PTY
C.... ...XMODEM: WAIT FOR MONITOR MODE.
C.... ...XOPEN/XCLOSE: INTERNAL ROUTINES TO KEEP TRACK OF PTY'S
C.... ...XERROR/XSYNC/XEXIT: COMMUNICATION TO CONTROLLER.
C.... ...XWAITM/XMESSW/XWATCH: HANDLE MESSAGES FROM PTY.
C.... ...XWAITT: SUSPEND CONTROLLER FOR A PERIOD OF TIME.
C.... ...XDATAI: INPUT CONTROL SYMBOLS AND FILE NAMES.
C.... ...XCLEAR: INITIALIZATION FOR XCNTL ROUTINES.
C.... ...XCOMPX/XQUEUE: COMPUTE NEW DATA FOR CONTROLLEE(PHY)
C.... ...XPTSTS: Retrieve PTY status bits.

```

C.... .PARAMETER DESCRIPTIONS:

```

C.... ...TABSIZ: SIZE OF MESSAGE AND CONTROL SYMBOL TABLES(MAX).
C.... ...QUELEN: QUEUE LENGTH FOR SAVING VALUES FROM PTY RUNS.

```

C.... .PARAMETERS:

PARAMETER TABSIZ=30, QUELEN=3

C.... .VARIABLE DESCRIPTIONS:

```

C.... ...NAMFPN: SYMBOL FOR PROGRAMMER,PROJECT NUMBER
C.... ...NCTRLT/NCTRLC: CONTROL CHARACTER VALUES.
C.... ...NAMPRC: CONTROLLEE NAME.
C.... ...NLECHO: FLAG CONTROLLING ECHOING OF MESSAGES TO CONTROLLER.
C.... ...NMSPTR: STACK POINTER FOR STRING MESSAGES
C.... ...NAMEIP/NAMEDP: SYMBOLS FOR THE INDEPENDENT AND DEPENDENT VAR.
C.... ...NPCNTL: NUMBER OF CONTROL SYMBOL PAIRS.
C.... ...CVALUE/EPSIL: CONVERG. VALUES AND EPSILONS FOR CONTROL PAIR.
C.... ...QUETAB/QUEPTR: QUEUE VARIABLES
C.... ...NOITER/MXITER: CONTROL OF NUMBER OF CONTROL ITERATIONS.
C.... ...NJOBTB: Job numbers for active PTY's.
C.... ...DPNORM: Normals for dep. variables. If zero, CVALUE is used.
C.... ...CHANIP: Change in independent variable limited to
C.... ...range [1/CHANIP,CHANIP]*Norm.
C.... ...NLREAD/NLPRES/VALPAR/NAMPAR: VARIABLES USED IN XPAIR FOR
C.... ...ACCESS BETWEEN SYMBOLS AND THEIR INTENDED VALUES.

```

C.... .

C.... .VARIABLES:

DOUBLE PRECISION KNAME1, KNAME2, NAMPPN, NAMPRG, NAMPAR
DOUBLE PRECISION NAMEIP(TABSIZ), NAMEFDP(TABSIZ)
LOGICAL NLECHO, NLREAD, NLPRES
INTEGER QUEPTR

COMMON/XCNTL/NCTRLT, NCTRLC, NAMPPN, NAMPRG, NAMPAR,
* NLECHO, NLREAD, NLPRES, VALPAR, NMSPTR, NPCNTL,
* NAMEIP, NAMEFDP, CVALUE(TABSIZ), EPSIL(TABSIZ),
* QUEPTR, QUETAB(TABSIZ, 2, QUELEN), NOITER, MXITER,
* NJOBTR(TABSIZ), DPNORM(TABSIZ), CHANIP(TABSIZ)

C.... .NOTE:

C.... ...ROUTINE XASSOC IS PROVIDED BY THE USER OF THESE CONTROL
C.... ...ROUTINES. IN XASSOC, THE USER MUST GIVE A SERIES OF CALLS
C.... ...TO THE XPAIR ROUTINE TO ESTABLISH A RELATION BETWEEN THE
C.... ...SYMBOLIC NAME OF THE CONTROL VARIABLE (USED BY XCNTL) AND
C.... ...THE REFERENCE TO THE VARIABLE AS IT EXISTS IN THE USERS DATA
C.... ...STRUCTURE. NO SPECIAL DESTINATION IS MADE BETWEEN DEPENDENT
C.... ...AND INDEPENDENT VARIABLES IN XPAIR.

```

C*****
C****
C****
C****      XCNTL ROUTINES--FOR PTY CONTROL
C****      PROGRAMMER: D.R. FAUL
C****      ORIGINATED: DECEMBER 8, 1978
C****      LAST REVISED: MARCH 12, 1979
C****      TARGET MACHINE: LLL PDPIO
C****
C****
C*****

```

C.... .GENERAL DESCRIPTION OF ROUTINES IN BINARY:

```

C.... ...XPAIR:  PAIR SYMBOL WITH ACTUAL VARIABLE.
C.... ...XLCIN:  LOG IN TO PTY
C.... ...XLCOFF: LOG OFF A PTY
C.... ...XOPEN/XCLOSE:  INTERNAL ROUTINES TO KEEP TRACK OF PTY'S
C.... ...XERROR/XSYNC/XEXIT:  COMMUNICATION TO CONTROLLER.
C.... ...XWAITM/XMESSW/XWATCH:  HANDLE MESSAGES FROM PTY.
C.... ...XWAITT:  SUSPEND CONTROLLER FOR A PERIOD OF TIME.
C.... ...XDATAI:  INPUT CONTROL SYMBOLS AND FILE NAMES.
C.... ...XCLEAR:  INITIALLIZATION FOR XCNTL ROUTINES.
C.... ...XCOMPX/XQUEUE:  COMPUTE NEW DATA FOR CONTROLLEE(PHY)
C.... ...XPTSTS:  Retrieve status bits for active PTY.

```

C.... .PARAMETER DESCRIPTIONS:

```

C.... ...TABSIZ:  SIZE OF MESSAGE AND CONTROL SYMBOL TABLES(MAX).
C.... ...QUELEN:  QUEUE LENGTH FOR SAVING VALUES FROM PTY RUNS.

```

C.... .VARIABLE DESCRIPTIONS:

```

C.... ...NAMPPN:  SYMBOL FOR PROGRAMMER,PROJECT NUMBER
C.... ...NCTRLT/NCTRLC:  CONTROL CHARACTER VALUES.
C.... ...NAMPRG:  CONTROLLEE NAME.
C.... ...NLECHO:  FLAG CONTROLLING ECHOING OF MESSAGES TO CONTROLLER.
C.... ...NMSPTR:  STACK POINTER FOR STRING MESSAGES
C.... ...NAMEIP/NAMEDP:  SYMBOLS FOR THE INDEPENDENT AND DEPENDENT VAR.
C.... ...NPCNTL:  NUMBER OF CONTROL SYMBOL PAIRS.
C.... ...CVALUE/EPSIL:  CONVERG. VALUES AND EPSILONS FOR CONTROL PAIR.
C.... ...QUETAB/QUEPTR:  QUEUE VARIABLES
C.... ...NOITER/MXITER:  CONTROL OF NUMBER OF CONTROL ITERATIONS.
C.... ...NJOBTB:  Job numbers for active PTY's.
C.... ...DPNORM:  Normals for dep. variables. If zero, CVALUE is used.
C.... ...CHANIP:  Change in the independent variable limited to the
C.... ...range [1/CHANIP,CHANIP]*Norm.
C.... ...NLREAD/NLPRES/VALPAR/NAMPAR:  VARIABLES USED IN XPAIR FOR
C.... ...ACCESS BETWEEN SYMBOLS AND THEIR INTENDED VALUES.

```

C.... .NOTE:

```

C.... ...ROUTINE XASSOC IS PROVIDED BY THE USER OF THESE CONTROL
C.... ...ROUTINES. IN XASSOC, THE USER MUST GIVE A SERIES OF CALLS
C.... ...TO THE XPAIR ROUTINE TO ESTABLISH A RELATION BETWEEN THE
C.... ...SYMBOLIC NAME OF THE CONTROL VARIABLE (USED BY XCNTL) AND

```


C.... ...THE REFERENCE TO THE VARIABLE AS IT EXISTS IN THE USERS DATA
 C.... ...STRUCTURE. NO SPECIAL DESTINCTION IS MADE BETWEEN DEPENDENT
 C.... ...AND INDEPENDENT VARIABLES IN XPAIR.

C.... .December 8, 1978: (Version XCNTL1.FOR)

C.... .January 9, 1979: (Version XCNTL2.FOR)

C.... ...Addition of status checking routine XPTSTS and NEWTON's
 C.... ...method for iterative solution of non-linear systems
 C.... ...of equations.
 C.... ...Programmer Donald Faul.

C.... .January 12, 1979:

C.... ...The source for the control routines is in the file: YCNTL2.FOR

C.... ...The cliché file is XCNTL.CLI. The old quadratic
 C.... ...version of XCOMPN is available in XCNTL1.FOR. Library
 C.... ...PDPLIB.REL[30,3046] (source PDPLIB.FOR) must be loaded
 C.... ...with the routines' binary, XCNTL2.REL. For XTCTMP, I
 C.... ...load using a command file, XTCTMP.CMD. All files
 C.... ...are on my private disk space.

C.... .January 29, 1979: (Version XCNTL3.FOR)

C.... ...New version of XCOMPN permitting output every step
 C.... ...and extending the extrema formula.
 C.... ...Programmer Donald Faul.

C.... .February 7, 1979: (Version XCNTL4.FOR)

C.... ...Revision of routines XCLEAR and XDATAI.

C.... .February 8, 1979: (Version XCNTL5.FOR)

C.... ...Revised XLGIN.

C.... .April 16, 1979 (Version XCNTL7.FOR)

C.... ...Slight revisions in XDATAI write to log file.

SUBROUTINE XLGIN(KLUN)
 INCLUDE 'XCNTL.CLI'

C.... .EQUIP AND LOGIN INTO A PTY=KLUN.

C.... ...PROGRAMMER DONALD FAUL, DECEMBER 8, 1978.

C.... .NOTE (12/12/78) LOGIN LINES(4) MUST BE READ SEPARATELY.

LOGICAL XOPEN,XCLOSE,XFIN,XPTSTS
 DIMENSION IBUFF(10)
 DATA KFMES\$/K/F'/

C.... .Pick array index for table of job 's.
 IF (XOPEN(KLUN,IJOB)) RETURN
 CALL XWAIT(1)
 WRITE(KLUN,9010) NAIPPN

C.... .This loop causes the controller to read two lines of mess-

C.... ...ages from LOGIN before checking status bits for MONITOR
 C.... ...mode. The job number is decoded on the first pass.

```

DO 1005 I=1,2
1003 CALL XWAITT(20000)
      IF (I .EQ. 1) GO TO 1004
      IF (.NOT.XPTSTS(KLUN,3)) GO TO 1003
1004 READ(KLUN,9030) Ibuff
      IF (I .NE. 1) GO TO 1006
      DECODE(6,9050,IBUFF) ZCHAR
      IF (ZCHAR .EQ. ' ') DECODE(5,9060,IBUFF) NJOBTB(IJOB)
      IF (ZCHAR .NE. ' ') DECODE(6,9070,IBUFF) NJOBTB(IJOB)
      IF (NLECHO) WRITE(5,9080) NJOBTB(IJOB)
1006 IF (NLECHO) WRITE(5,9040) KLUN,IBUFF
1005 CONTINUE
      INDEX=1
      GO TO 1020
  
```

C.... .WAIT UNTIL IN MONITOR MODE.

```

      ENTRY XMODEM(KLUN)

      INDEX=2
      ILINE=0

1020 CALL XWAITT(10000)
      IF (XPTSTS(KLUN,2) .AND. XPTSTS(KLUN,4)
      * .AND. .NOT.XPTSTS(KLUN,3)) GO TO 1025
      IF (XPTSTS(KLUN,4)) GO TO 1022
      IF (.NOT.XPTSTS(KLUN,3)) GO TO 1020

1022 READ(KLUN,9030) Ibuff
      ILINE=ILINE+1
      IF (NLECHO .AND. Ibuff(1).NE.' '
      * .AND. (INDEX.NE.2 .OR. ILINE.LE.10))
      * WRITE(5,9040) KLUN,IBUFF
      GO TO 1020

1025 GO TO (1030,1030,2020) INDEX
1030 RETURN
  
```

C.... .LOGOFF THE PTY.

```

      ENTRY XLGCOFF(KLUN)

      INDEX=3
      IF (XCLOSE(KLUN)) RETURN
2010 CALL XWAITT(1000)
      IF (XPTSTS(KLUN,5)) GO TO 2010
      IF (XPTSTS(KLUN,4) .AND.XPTSTS(KLUN,2) .AND..NOT.XPTSTS(KLUN,3))
      * GO TO 2020
      WRITE(KLUN,9020) NCTRLC
      WRITE(KLUN,9020) NCTRLC
C.... GO TO 1020

2020 WRITE(KLUN,9030) KFMESS
  
```

```

2030 CALL XWAITT(20000)
      IF (XPTSTS(KLUN,4)) GO TO 2035
      IF (XPTSTS(KLUN,1)) GO TO 2030
2035 READ(KLUN,9030) IBUFF
      IF (NLECHO) WRITE(5,9040) KLUN,IBUFF
      IF (IBUFF(1) .NE. "Runti") GO TO 2030
      IF (XFIN(KLUN)) CLOSE(UNIT=KLUN)
      RETURN

9010 FORMAT('LOC ',A10)
9020 FORMAT(A1,$)
9030 FORMAT(10A5)
9040 FORMAT(I3,'-',10A5)
9050 FORMAT(5X,A1)
9060 FORMAT(4X,I1)
9070 FORMAT(4X,I2)
9080 FORMAT( X,'JOB NUMBER=',I3 )
      END
C.... .XLGIN,XLGOFF,XMODEM.

```

```

FUNCTION XPTSTS(KLUN,KFIELD)
INCLUDE 'XCNTL.CLI'

```

```

C.... .If the status bit indicated by KFIELD for the PTY device
C.... .referenced by KLUN is one, then a value of TRUE is returned
C.... .as the value of XPTSTS (likewise if zero then FALSE).
C.... .The meanings of the status bits are discussed in the
C.... .DECsystem-10 MONITOR CALLS MANUAL, pp.5-25
C.... .XPTSTS calls the MACRO routine XCTSTS to obtain the
C.... .actual status word for the job number (see array NJOETB in
C.... .routine XOPEN).
C.... .Programmer Donald Faul, January 9, 1979.

```

```

LOGICAL XPTSTS,XOPEN

```

```

      XPTSTS=.TRUE.
      IF (.NOT.XOPEN(KLUN,INDEX))
*        CALL XERROR('XPTSTS','NOT A JOB',KLUN,.FALSE.)
      CALL XCTSTS(NJOETB(INDEX),ISTS)
      IF (ISTS .EQ. 0)
*        CALL XERROR('XPTSTS','BAD STATUS',KLUN,.TRUE.)
      IMASK=2**(35-KFIELD)
      ITEMP=ISTS.AND. IMASK
      XPTSTS=ITEMP.NE.0
1010 RETURN
      END
C.... .XPTSTS.

```

```

FUNCTION XOPEN(KLUN,KJOB)
  INCLUDE 'XCNTL.CLI'

C.... .INTERNAL ROUTINES FOR XLGIN,XLCOFF.
C.... ...PROGRAMMER DONALD FAUL, DECEMBER 8, 1978

  LOGICAL XOPEN,XCLOSE,XFIN
  DIMENSION ZLNTAB(TABSIZ)
  DATA ITBPTR/0/

  XOPEN=.TRUE.
  DO 1010 I=1,ITBPTR
    KJOB=I
    IF (KLUN .EQ. ZLNTAB(I)) RETURN
1010 CONTINUE
    ITBPTR=MINO(ITBPTR+1,TABSIZ)
    ZLNTAB(ITBPTR)=KLUN
    OPEN(UNIT=KLUN,ACCESS='SEQUINOUT',DEVICE='PTY')
    KJOB=ITBPTR
    XOPEN=.FALSE.
    RETURN

  ENTRY XCLOSE(KLUN)

  XCLOSE=.TRUE.
  DO 2020 I=1,ITBPTR
    J=I
    IF (KLUN .EQ. ZLNTAB(I)) GO TO 2020
2010 CONTINUE
    RETURN
2020 XCLOSE=.FALSE.
    RETURN

  ENTRY XFIN(KLUN)

  DO 2030 I=J+1,ITBPTR
    NJOEIB(I-1)=NJOEIB(I)
    ZLNTAB(I-1)=ZLNTAB(I)
2030 CONTINUE
    ITBPTR=MAXO(ITBPTR-1,0)
    RETURN
  END

C.... .XOPEN,XCLOSE,XFIN.

SUBROUTINE XERROR(KNAME1,FNAME2,KLUN,KFLAG)
  INCLUDE 'XCNTL.CLI'

C.... .MESSAGES TO CONTROLLER OF PTY=KLUN. KNAME1 SHOULD BE
C.... ...THE NAME OF THE CALLING ROUTINE, FNAME2 A DESCRIPTION
C.... ...OF THE PROBLEM. IF KFLAG IS FALSE, THE CONTROLLEE

```

C.... ...WILL ABORT, CLOSING ALL PTY'S.
 C.... ...PROGRAMMER DONALD FAUL, DECEMBER 8, 1978.

LOGICAL KLFLAG

CALL GETMES(IMESS)
 WRITE(5,9010) NAMPRG,KNAME1,KNAME2,KLUN
 IF (KLFLAG) RETURN

C.... .CLOSE OUT PTY'S.

ENTRY XEXIT

1001 DO 1010 JLUN=1,99
 JLUN=JLUN
 CALL XLGOTFF(JLUN)
 1010 CONTINUE

CALL GETMES(IMESS)
 WRITE(5,9020) NAMPRG
 CALL EXIT
 RETURN

C.... .SYNCHRONIZATION MESSAGE TO CONTROLLER.

ENTRY XSYNC(KNAME1,KNAME2,KLUN)

CALL GETMES(IMESS)
 WRITE(5,9030) NAMPRG,KNAME1,KNAME2,KLUN
 RETURN

9010 FORMAT(2X,A6,':E Routine:',A10,5X,'Desc:',A10,5X,'Lun:',I3/X)
 9020 FORMAT(2X,A6,':C '/X)
 9030 FORMAT(2X,A6,':X Routine:',A10,5X,'Desc:',A10,5X,'Lun:',I3/X)
 END

C.... .XERROR,XEXIT,YSYNC.

SUBROUTINE XWAITM(KLUN,VINDEX,KWAIT,PRUFF,KWAITT)
 INCLUDE 'XCNTL.CLI'

C.... .WAIT FOR A MESSAGE FROM THE PTY(ONLY ONE SHOULD BE ACTIVE)
 C.... ...=KLUN. WAIT THROUGH KWAIT MESSAGES BEFORE GIVING
 C.... ...UP(IF =0, WAIT INDEFINATELY). CONTENTS OF THE LAST MESSAGE GO
 C.... ...INTO A USER SUPPLIED BUFFER(PBUFF). THE INDEX OF THE MESSAGE
 C.... ...(1 IF NONE FOUND, 2 IF MODEM) GOES INTO KINDEX.
 C.... ...PROGRAMMER DONALD FAUL, DECEMBER 8, 1978.

C.... .NOTE (12/19/78) KWAITT IS THE WAIT TIME.

C.... .Note (01/09/79) Substitution of status checking routine
 C.... ...XPTSTS for the ad hoc method by writing control-T's.

C.... .Note (01/12/79) The first return from XWAITM after
 C.... ...starting" a program on the PTY will be a Monitor Mode
 C.... ...exit. By "starting" a program I mean writing
 C.... ...RUN <Name of Executable File>" to the user
 C.... ...defined logical unit for the PTY.

```
DIMENSION ZMSTAB(TABSIZ),IMSLUN(TABSIZ)
DOUBLE PRECISION ZMSTAB,PBUFF(8),PNAME1
LOGICAL XPTSTS
```

```
ICOUNT=0
```

```
1010 KINDEX=2
    CALL XWAITT(KWAITT)
    IF (XPTSTS(KLUN,4)) GO TO 1015
    IF (.NOT.XPTSTS(KLUN,3)) GO TO 1010

1015 READ(KLUN,9010) PBUFF
    IF (NLECHO .AND. PBUFF(1).NE.' ')
      * WRITE(5,9020) KLUN,(PBUFF(I),I=1,6)

    DO 1020 I=1,NMSPTR
      IF (KLUN .NE. IMSLUN(I)) GO TO 1020
      KINDEX=KINDEX+1
      IF (PBUFF(1) .EQ. ZMSTAB(I)) RETURN
1020 CONTINUE
    IF (XPTSTS(KLUN,2)) GO TO 1030

    KINDEX=1
    IF (KNWAIT .EQ. 0) GO TO 1010
    ICOUNT=ICOUNT+1
    IF (ICOUNT .GE. KNWAIT) RETURN
    GO TO 1010
```

```
1030 KINDEX=2
    RETURN
```

C.... .ENTER TO SAVE PNAME1 AS A WAIT MESSAGE FOR PTY=KLUN

```
ENTRY XMESSW(KLUN,PNAME1)

DO 2010 I=1,NMSPTR
  IF (KLUN .NE. IMSLUN(I)) GO TO 2010
  IF (PNAME1 .EQ. ZMSTAB(I)) RETURN
2010 CONTINUE

NMSPTR=MIN0(NMSPTR+1,TABSIZ)
IMSLUN(NMSPTR)=KLUN
ZMSTAB(NMSPTR)=PNAME1
RETURN
```

```
9010 FORMAT(8A10)
9020 FORMAT(I3,'-',8A10)
9030 FORMAT(A1,$)
9040 FORMAT(5A5)
END
```

C.... .XWAITM,XMESSW.

SUBROUTINE XWATCH(KLUN,KNAME1)
INCLUDE 'XCNTL.CLI'

C.... .WATCH FOR MY STYLE OF MESSAGES FROM A CONTROLLEE, BASED ON
C.... ...THE INPUT SYMBOL(IE.NAME OF THE CONTROLLEE)
C.... ...PROGRAMMER DONALD FAUL: DECEMBER 11 1978.

DOUBLE PRECISION INAME,IFORM(3),JFORM(2)
DATA IFORM/'2H:X','2H:E','2H:C'/

JFORM(1)='(X,A6,
DO 1050 I=1,3
JFORM(2)=IFORM(I)
ENCODE(10,JFORM,INAME) KNAME1
CALL XMESSW(KLUN,INAME)

1050 CONTINUE
RETURN
END

C.... .XWATCH.

C.... ;SUBROUTINE XWAITT:
C.... ;;WAIT FOR THE SPECIFIED NUMBER OF MILLISECONDS OR UNTIL A
C.... ;;MESSAGE FROM THE PTY. IF THE ARGUMENT IS ZERO, WAIT
C.... ;;Programmer Donald Faul.
C.... ;;Note that XWAITT is FORTRAN callable.
C....
C.... ;;INDEFINATELY.
C.... TITLE XWAITT
C.... ENTRY XWAITT
C.... XWAITT: HRLZI 0,50
C.... HRR 0,@0(16)
C.... HIBER 0,
C.... POPJ 17,
C.... POPJ 17,
C.... END

C.... ;SUBROUTINE XGTSTS:
C.... ;;FORTRAN callable subr. to obtain the status bits for a PTY.
C.... ;;Argument one is the job number and argument two is the
C.... ;;location where the status word is returned.
C.... ;;Programmer Donald Faul, January 9, 1979.

C.... TITLE XGTSTS

```

C....      ENTRY XGTSTS
C.... XGTSTS:  MOVEI    0,0
C....      MOVEM    0,@1(16)
C....      MOVN    0,@0(16)
C....      JOBSTS   0,
C....      POPJ    17,
C....      MOVEM    0,@1(16)
C....      POPJ    17,
C....      END
C.... END

```

```

SUBROUTINE XCLEAR(KNAME1,KNAME2)
INCLUDE 'XCNTL.CLI'

```

```

C.... .INITIALIZE XCNTL ROUTINES:
C.... ...KNAME1 IS THE ASSIGNED NAME FOR THE CONTROLLER PROGRAM.
C.... ...KNAME2: PROJECT PROGRAMMER NUMBER
C.... ...PROGRAMMER DONALD FAUL: DECEMBER 11, 1978.

```

```

DATA NCTRLT/"120000000000/",NCTRLC/"014000000000/"

```

```

MLECHO=.TRUE.
NMSPTR=0
NAMPRG=KNAME1
NAMPFN=KNAME2
NPCNTL=0
MXITER=30
IMXEXT=1
NOITER=0
QUEPTR=0

```

```

DO 1010 I=1,TABSIZ
  NJOBTB(I)=0
  DO 1010 J=1,2
    DO 1010 K=1,QUELEN
      QUETAB(I,J,K)=1.E38

```

```

1010 CONTINUE
RETURN
END
C.... .XCLEAR.

```

```

SUBROUTINE XDATAI(KLUN,KNARR)
INCLUDE 'XCNTL.CLI'

```

```

C.... .OPEN FILE FOR CONTROL VARIABLES ON UNIT FLUN AND READ THEM.
C.... ...PROGRAMMER DONALD FAUL: DECEMBER 11, 1978.

```



```

DOUBLE PRECISION KNARR(6),INAME1

C.... .READ FILENAME FOR CONTROL VARIABLES.
      CALL XSYNC('XDATA1','CNTLFILE?',KLUN)
      READ(5,9010) INAME1
1005 OPEN(UNIT=KLUN,ACCESS='SEQIN',DEVICE='DSK',FILE=INAME1)

C.... READ NO. OF CONTROL VARIABLES, NO. OF FILENAMES TO INPUT,
C.... ..MAX. NO. OF RETRIES IN XCOMP, MAX. NO. OF ITERATIONS IN
C.... ..XCOMP, PPN NO.(300/317), CONTROLLER PROGRAM NAME, AND
C.... ..FLAG FOR ECHO OF MESSAGES.
C.... ..ONLY FIRST TWO ARE REQUIRED.
      READ (KLUN,9020) NPCNTL,IFILE,IMEXT,MXIT,KNARR(1),KNARR(2),KLECHO
      IF (IMEXT .GT. 0) IMEXT=IMEXT
      IF (MXIT .GT. 0) MXITER=MXIT
      IF (KLECHO .EQ. 'F') NLECHO=.FALSE.
      IF (KNARR(1) .NE. ' ') NAMPPN=KNARR(1)
      IF (KNARR(2) .NE. ' ') NAMPRG=KNARR(2)
      IF (NLECHO)
*      WRITE(5,9040)NAMPRG,NAMPPN,NLECHO,NPCNTL,IFILE,IMEXT,MXITER
      NPCNTL=MINO(TABSIZ,NPCNTL)
      IFILE=MINO(6,IFILE)
      DO 1010 I=1,IFILE
        KNARR(I)='TTY'
1010 CONTINUE

C... .READ INDEPENDANT AND DEPENDANT VARIABLE NAMES, DESIRED VALUE,
C... ..ACCURACY, NORMALIZATION VALUE, PERMITTED CHANCE ON EACH
C... ..ITERATION.
      READ(KLUN,9030,ERR=8010)
*      (NAMEIP(I),NAMEDP(I),CVALUE(I),EPSIL(I),DPNORM(I),
*      CHANIP(I),I=1,NPCNTL)
      IF(NLECHO) WRITE(5,9050)
*      (NAMEIP(I),NAMEDP(I),CVALUE(I),EPSIL(I),DPNORM(I),
*      CHANIP(I),I=1,NPCNTL)

C.... READ FILENAMES NEEDED FOR CONTROLLER
      READ(KLUN,9010,ERR=8010) (KNARR(I),I=1,IFILE)
      CLOSE(UNIT=KLUN)
      IF (NLECHO) WRITE(5,9060) (KNARR(I),I=1,IFILE)
      RETURN

8010 CALL XERROR('XDATA1','CNTLFILE',KLUN,.TRUE.)
      CLOSE(UNIT=KLUN)
      CALL XEXIT

9010 FORMAT(8A10)
9020 FORMAT(3I2,I4,2A10,A5)
9030 FORMAT(2A10,4E10.0)
9040 FORMAT(/X,'CONTROL VARIABLES:',/
1 3X,'NAMPRG:',A10,3X,'NAMPPN:',A10,3X,'NLECHO:',L5,/
2 3X,'NPCNTL:',I5,3X,'IFILE:',I5,3X,'IMEXT:',I5,3X,'MXITER:',
3 I5,/X,'OTHER CONTROL DATA:')
9050 FORMAT(3X,2A10,1PE10.3,3E10.3)
9060 FORMAT(3X,8A10/)
      END

```

C.... .XDATAI.

SUBROUTINE XPAIR(KNAME1,PVALUE)
INCLUDE 'XCNTL.CLI'

C.... .IF COMMON VARIABLE SYMBOL NAMPAR EQUALS THE ARGUMENT KNAME1,
C.... ...THEN DEPENDING ON FLAG NLREAD, TRANSFER THE VALUE OF VALPAR
C.... ...TO/FROM PVALUE. SET NLPRES FLAG IF A NAME MATCH DOES OCCUR.
C.... ...PROGRAMMER DONALD FAUL, DECEMBER 8, 1978.

IF (NLPRES .OR. KNAME1.NE.NAMPAR) RETURN
IF (NLREAD) VALPAR=PVALUE
IF (.NOT.NLREAD) PVALUE=VALPAR
NLPRES=.TRUE.
RETURN

END

C.... .XPAIR.

SUBROUTINE XCOMP(NLSTOP)
INCLUDE 'XCNTL.CLI'

C.... .COMPUTE PARAMETER VALUES FOR INPUT TO THE CONTROLLEE.
C.... ...THE CALCULATION OF THESE PARAMETER VALUES IS BASED ON
C.... ...INFORMATION OBTAINED BY THE CONTROL ROUTINE XDATAI.
C.... ...XDATAI READS THE CONTROL SETS DESIRED BY THE USER.
C.... ...THE CONTROL SETS ARE LISTED IN THE CHOSEN ORDER OF
C.... ...SIGNIFICANCE. EACH CONTROL SET CONTAINS THE SYMBOLIC INDEP.
C.... ...VARIABLE, THE SYMBOLIC DEPENDENT VARIABLE, THE CONVERGENCE
C.... ...VALUE DESIRED, AND THE CONVERGENCE EPSILON. XCOMP QUEUES
C.... ...PREVIOUS VALUES FROM PREVIOUS CALLS TO A DEPTH OF QUELEN.
C.... ...AT PRESENT A LINEAR INTERPOLATION SCHEME IS USED.
C.... ...WHEN 1)CONVERGENCE FOR ALL CONTROL PAIRS IS REACHED OR 2)
C.... ...THE MAXIMUM NUMBER OF ITERATIONS IS EXCEEDED, THEN THE
C.... ...USER VARIABLE NLSTOP IS SET. ROUTINE XASSOC MUST BE PROVIDED
C.... ...BY THE USER AND SHOULD CONTAIN CALLS TO XPAIR. XPAIR
C.... ...ESTABLISHES AN ASSOCIATION BETWEEN A SYMBOL AND
C.... ...A USER VARIABLE.
C.... ...PROGRAMMER DONALD FAUL: DECEMBER 11, 1978.

C.... .(12/24/78) I have replaced the original linear
C.... ...scheme with a quadratic approximation. Hopefully, this
C.... ...method will detect local extrema quickly and easily.
C.... ...The algorithm below assumes that the 'convergence pairs' are
C.... ...essentially independent. However, the convergence
C.... ...of a higher ranked pair is always rechecked after an
C.... ...iteration on a lower ranked pair. The ranking of pairs
C.... ...is determined by the order that the parameter sets are read in
C.... ...by routine XDATAI.

C.... .(01/10/79) Quadratic scheme replaced by Newton's method
 C.... ...for the iterative solution of non-linear systems of equations.
 C.... ...Detection of local extrema in this version is primitive, and
 C.... ...will probably have to be revised.
 C.... ...Programmer Donald Faul.

C.... .(01/12/79) Change in handling local extrema.
 C.... ...Programmer Donald Faul.

C.... .(01/17/79) Echo flag enables printout of QUETAB values
 C.... ...and normal, ZFUNCO.
 C.... ...Programmer Donald Faul.

C.... .(01/29/79) Printout for every iteration provided NLECHO is set.
 C.... ...Programmer Donald Faul.

LOGICAL KSTOP
 DIMENSION ZDIFF(TABSIZ,TABSIZ),ZFUNC(TABSIZ),IP(TABSIZ)
 DIMENSION ZNORM1(TABSIZ),ZNORM2(TABSIZ)

IF (NOITER .EQ. 0) GO TO 7005
 NLREAD=.TRUE.
 INDEX=MOD(NOITER,NPCNTL+1)
 IF (INDEX.EQ.1 .AND. NOITER/(NPCNTL+1).GE.MXITER) GO TO 8005

IF (INDEX .EQ. 1) GO TO 1050

C.... .Compute parital derivatives for result of last run.

J=INDEX-1
 IF (INDEX .EQ. 0) J=NPCNTL

 ZTERM=0.
 DO 1020 I=1,NPCNTL
 NLPRES=.FALSE.
 NAMPAR=NAMEIP(I)
 CALL XASSOC
 IF (.NOT.NLPRES) GO TO 8020
 ZDIFF(I,J)=100.*(QUETAB(I,2,1)-VALPAR)/ZNORM2(I)

IF (.NOT.NLECHO) GO TO 1020

ZTERM=ZTERM+

* ABS((VALPAR-CVALUE(I))/ZNORM2(I))**2
 WRITE(5,9020) NAMEIP(I),VALPAR,NAMEIP(J)

1020 CONTINUE
 ZTERM=SQRT(ZTERM)
 IF (NLECHO) WRITE(5,9010) ZTERM

NLREAD=.FALSE.
 NLPRES=.FALSE.
 NAMPAR=NAMEIP(J)
 VALPAR=QUETAB(J,1,1)
 CALL XASSOC
 GO TO 2010

C.... .Compute normalization variables and check for convergence.

```

1050 CALL XQUFUE
      DO 1055 I=1,NPCNTL
          ZNORM1(I)=QUETAB(I,1,1)
          IF (ZNORM1(I) .LE. 0.) ZNORM1(I)=1.
          ZNORM2(I)=DPNORM(I)
          IF (ZNORM2(I) .EQ. 0.) ZNORM2(I)=CVALUE(I)
          IF (ZNORM2(I) .EQ. 0.) ZNORM2(I)=ZNORM1(I)
1055 CONTINUE

1062 ZFUNC1=ZFUNC0
1063 ZFUNC0=0.
      DO 1064 I=1,NPCNTL
          ZFUNC0=ZFUNC0+
          *      AES((QUETAB(I,2,1)-CVALUE(I))/ZNORM2(I))**2
1064 CONTINUE
      ZFUNC0=SORTE(ZFUNC0)
      IF (.NOT.NLECHO) GO TO 1067
      WRITE(5,9010) ZFUNC0
      DO 1066 I=1,NPCNTL
          WRITE(5,9020)
          *      (NAMEIP(I),QUETAB(I,1,1),NAMEFP(I),QUETAB(I,2,1))
1066 CONTINUE

1067 DO 1068 I=1,NPCNTL
          IF (ABS((QUETAB(I,2,1)-CVALUE(I))/ZNORM2(I)).GT.EPSIL(I))
          *      GO TO 1069
1068 CONTINUE
      GO TO 8010

1069 IF (NOITER .EQ. 1) GO TO 2010
      IF (ABS((ZFUNC0-ZFUNC1)/ZFUNC0) .LT. 1.E-2) GO TO 8010

```

C.... .Extrema. Run for a weighted average of the last two points,
 C.... ...and then quit with a message. The weights are based on
 C.... ...the reciprocal of the standard norm at the point.
 C.... .(01/29/79) Run up to IMXEXT times with the average.

```

      IF (ZFUNC0.LT.ZFUNC1 .AND. INOEXT.EQ.0) GO TO 2010
      IF (INOEXT .GE. IMXEXT) GO TO 8010
      CALL XERROR('XCOMPN','EXTREMA?',0,.TRUE.)
      NOITER=(NXITER+1)*(NPCNTL+1) +1
      NLRFAD=.FALSE.
      ZTERM0=1./ZFUNC0
      ZTERM1=1./ZFUNC1
      ZTERM2=ZTERM0 +ZTERM1
      DO 1070 I=1,NPCNTL
          NLPRES=.FALSE.
          NAMEPAR=NAMEIP(I)
          VALPAR=(ZTERM0*QUETAB(I,1,1)
          *      +ZTERM1*QUETAB(I,1,2))/ZTERM2
          CALL XASSOC
          IF (.NOT.NLPRES) GO TO 8020
1070 CONTINUE
      RETURN

```

2010 NLREAD=.FALSE.

C.... .Change independent variable NAMEIP(INDEX) by 1%.

```
IF (INDEX .EQ. 0) GO TO 2050
  NLPRES=.FALSE.
  NAMEIP=NAMEIP(INDEX)
  VALPAR=QUETAB(INDEX,1,1)*.99
  CALL XASSOC
  IF (.NOT.NLPRES) GO TO 8020
  GO TO 7010
```

C.... .Solve for the new values of the independent variables.

```
2050 DO 2060 I=1,NPCNTL
      ZFUNC(I)=(QUETAB(I,2,1)-CVALUE(I))/ZNORM2(I)
2060 CONTINUE
      CALL DEC(NPCNTL,TABSIZE,ZDIFF,IP,IFRROP)
      IF (IERROR .NE. 0)
*        CALL XERROR('XCOMPN','SINGULAR',0,.FALSE.)
      CALL SOL(NPCNTL,TABSIZE,ZDIFF,ZFUNC,IP)
      DO 2070 I=1,NPCNTL
        NLPRES=.FALSE.
        NAMEIP=NAMEIP(I)
        ZTERM1=CNAMEIP(I)
        IF (ZTERM1 .EQ. 0.) ZTERM1=10.
        VALPAR=ZNORM1(I)*AMIN1(ZTERM1,AMAX1(1./ZTERM1,
*          1.-ZFUNC(I) ))
        CALL XASSOC
        IF (.NOT.NLPRES) GO TO 8020
2070 CONTINUE
      GO TO 7010
```

7005 INOEXT=0

ZFUNCO=0.

7010 NOITER=NOITER+1

RETURN

8005 INOEXT=INOEXT+1

GO TO 1050

8010 KLSTOP=.TRUE.

RETURN

8020 CALL XERROR('XCOMPN','ASSOC?',0,.FALSE.)

C.... .Format statements.

9010 FORMAT(5X,'NORMAL:',1PE13.5)

9020 FORMAT(5X,A10,':',1PE13.5,5X,A10,':',F13.5)

END

C.... .XCOMPN.

```
SUBROUTINE XQUEUE
INCLUDE 'XCNTRL.CLI'
```

```
C.... .QUEUE THE CONTROL VALUES.
C.... ...PROGRAMMER DONALD FAUL: DECEMBER 11, 1978.
```

```

      QUEPTR=MIN0(QUEPTR-1,QUELEN)
DO 1050 I=1,NPCNTL
      NLPRES=.FALSE.
      NAMPAR=NAMEIP(I)
      CALL XASSOC
      ZTERM1=VALPAR
      IF (.NOT.NLPRES) GO TO 8010
      NLPRES=.FALSE.
      NAMPAR=NAMEIP(I)
      CALL XASSOC
      ZTERM2=VALPAR
      IF (.NOT.NLPRES) GO TO 8010
      IF (QUEPTR .LE. 1) GO TO 1030

      DO 1020 J1=QUEPTR-1,1,-1
        DO 1020 J2=1,2
          QUETAB(I,J2,J1+1)=QUETAB(I,J2,J1)
1020   CONTINUE

1030   QUETAB(I,1,1)=ZTERM1
      QUETAB(I,2,1)=ZTERM2
1050 CONTINUE
      RETURN

8010 CALL XERROR('XQUEUE','ASSOC?',0,.FALSE.)
      END
C.... ..XQUEUE.
```