NASA Contractor Report 172189
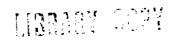
NASA-CR-172189
19830026340

FEM ARRAY CONTROL SOFTWARE
USER'S GUIDE

Judson D. Knott

Kentron Technical Center
Hampton, Virginia 23666

Contract NAS1-16000
August 1983

NASA

**National Aeronautics and
Space Administration**

Langley Research Center
Hampton, Virginia 23665

NF02500

FACS User's Guide

Contents

FACS User's Guide

MISCELLANEOUS

APPENDICES

1.   INTRODUCTION

The   FEM   Array   Control   Software   (FACS)   is   a   collection   of
menu   driven,   user   friendly   commands   used   to   initialize,   execute,
and   debug   user   tasks   on   the   Finite   Element   Machine   (FEM).   FACS
resides   on   the   Controller   (a   Texas   Instruments   990/10
minicomputer)   and   utilizes   the   System   Command   Interpreter   (SCI)
to   interface   between   the   user   at   the   video   display   terminal   and
the   DX10   operating   system.

This   manual   gives   detailed   descriptions   of   all   available
FACS   commands   along   with   their   purpose,   intended   use,   and
potential   hazards.   It   is   assumed   that   the   reader   is   familiar   with
the   DX10   operating   system,   the   use   of   the   System   Command
Interpreter   (SCI),   and   the   architecture   of   the   Finite   Element
Machine.   These   topics   are   covered   in   detail   by   the   following
references:

THE FINITE ELEMENT MACHINE PROGRAMMER'S REFERENCE MANUAL

MODEL 990 COMPUTER DX10 OPERATING SYSTEM, Vols. I-VI

2.   SYSTEM NOTES

The   FACS   system   is   intended   to   support   research   into
parallel   algorithms   on   the   Finite   Element   Machine.   To   facilitate
algorithm   development   on   FEM,   FACS   supports   an   extremely
versatile   interactive   environment   with   extensive   debugging
support.   In   addition,   the   integration   of   FACS   into   the   host
operating   system   provides   the   user   with   a   consistent   interface   to
the   full   range   of   system   resources   and   the   capability   to
predefine   execution   sequences   once   system   requirements   are   well
understood.

FACS   commands   are   implemented   as   independent   PASCAL   programs
using   the   DX10   System   Command   Interpreter   as   a   standard   user
interface.   To   provide   for   continuity   between   the   stand   alone   FACS
commands,   a   number   of   system   synonyms   and   files   are   automatically
created,   maintained,   and   deleted   by   FACS.

The   FACS   system   file   names   and   a   description   of   their   use   is
given   in   Appendix   B.   However,   the   FEMDATA   file   (SCRATCH.DATA)   is
of   particular   importance   to   FACS   users   and   is   also   described
here.   The   FEMDATA   file   is   also   known   as   the   FEM   log   file.   It   is
created   by   the   ATTACH   command   and   initialized   by   the   RESET
command.   Almost   all   of   the   FACS   commands   utilize   the   FEM   log
file.   When   a   FACS   command   is   invoked,   a   command   entry   message   is
recorded   in   the   log   along   with   any   options   selected   for   that
command.   In   addition,   any   errors   encountered   during   the   execution
of   a   command   are   recorded   as   they   are   detected.   This   makes   the
FEM   log   file   an   extremely   valuable   tool.   Whenever   errors   are
encountered   while   using   the   Finite   Element   Machine   the   user   can
review   the   exact   sequence   of   commands   executed   and   the   options
selected   that   led   to   the   error   condition.   The   log   file   can   also
be   printed   to   provide   a   permanent   record   of   an   Array   session   for

later reference.

The DX10 operating system allows the user to define character strings which are identified by user assigned labels called "synonyms". These synonyms are kept in the System Table Area and are used by FACS to either specify FACS file names, or to provide for temporary storage of information necessary to command continuity. The synonyms used by FACS are created, defined, and deleted when appropriate and require no user intervention. However, when constructing custom SCI procedures, it is important that the user understand the purpose of these synonyms so as not to inadvertently alter a synonym used for system continuity. Appendix C contains a list of FACS synonyms along with their intended use and their usual values. Allowing the system to control these definitions helps to keep the user's work area free of unnecessary clutter and provides a standard for file management which simplifies FEM usage. The file standardization and system control of synonyms also makes it easier to combine FACS commands in SCI procedure files which can then supervise execution, from initialization through post processing, with a single command entry at the keyboard. The use of SCI is covered in detail by Volume III of the MODEL 990 COMPUTER DX10 OPERATING SYSTEM manual, and will not be covered here. However, Appendix D is included as an example of how FACS commands can be combined to create custom procedure files for FEM.

Although the FACS system commands are implemented as separate programs, there is an order imposed by the conceptual system design. Commands in FACS become meaningful only in the context of the FEM "session". To use FEM, the user must first request control of the Array using the ATTACH command. Once the Array is attached, the user is guaranteed exclusive access to FEM and is free to begin a FEM session. The FEM session begins when the user resets the Array and initializes the NODAL EXECUTIVE Operating System on the Array using the RESET command, and continues until the next RESET is given or until the Array is released by the user. Following the RESET, the basic session consists of selecting the set of processors to utilize, defining and downloading the necessary data areas, defining the I/O connectivity, downloading the object code for the selected program, and executing the program on the Array. While FACS provides a wide variety of options, this basic session scenario is common to all applications on FEM.

In addition to initializing the Array, the RESET command defines a logical to physical processor mapping that affects nearly all of the subsequent FACS commands. The mapping option allows applications to be written for a set of N logical processors that can run on any set of N physical processors. See the RESET command description for a detailed explanation of the mapping and available map options.

3.   COMMAND DESCRIPTIONS

FACS commands are called by SCI procedures. These commands can be invoked by the user at the terminal or can be called from within other SCI procedures. This section describes the purpose, operation, intended use, required prompts, and the potential hazards associated with each of the FACS commands.

Since many of the prompts are common to several commands, the prompt type is identified by a prompt type identifier enclosed in brackets (). Prompt type identifiers are listed in Appendix A.   In those instances where the range of a valid response to an SCI prompt is a subset of the prompt type range, the valid range of user response is indicated in parenthesis following the prompt type identifier.

Two symbols frequently encountered when using FACS are the asterisk (*) and "greater than" (>). The asterisk is used to identify relative addresses (e.g. *2340 is relative address 2340) and the "greater than" symbol denotes a hexadecimal value (e.g. >1234 is hexadecimal, 1234 is decimal). These symbols are used quite frequently in FACS terminal displays.

Symbols are also used to graphically represent the state of individual processors in the Array. There are four possible states for Array processors: 1) ON, 2) OFF, 3) HALTED, or 4) UNKNOWN. Many of the FACS commands graphically display the state of all processors upon termination. For an explanation of the processor states and the symbols that represent them, see the SFS (Show Fem State) command description.

3.1   ATTACH - ATTACH fem

Purpose:

To obtain exclusive access to FEM.

Description:

The ATTACH command tests to determine the availability of the
processor Array. If the Array is available ATTACH locks FEM
to the calling station, assigns all necessary synonyms,
creates the required system files, and writes a message to
the system log file identifying the user id and calling
station number. ATTACH also reports the status (either a
successful lock or locked by another user) to the calling
station. Once attached, the user has exclusive access to the
FEM Array until his station is released. A station can be
released   by   the   user   via   the RELEASE command, is
automatically released when the user logs off the system, or
can be forced to release by the FFR(Force Fem Release)
command issued by a user with system level privileges.

Prompts:

AUTORESET: (yes/no)

Warnings/Limitations:

The ATTACH command must be performed before any other FACS
command. ATTACH will automatically RESET the FEM Array using
the identity map unless the user overrides the default
AUTORESET prompt.

Usage:

The ATTACH command is used to gain control of FEM. ATTACH can
be used to determine the current availability of FEM, but
will lock FEM to your station if it is not in use. A better
method of checking the status of FEM is to examine the system
log files (.S$SLG1 and .S$SLG2) for ATTACH and RELEASE
messages.

3.2   RESET - RESET fem

Purpose:

The RESET command is used to initialize the FEM Array and
Controller. It is the most important FACS command in that all
subsequent FACS commands will operate in the environment
established by the RESET command.

Description:

The RESET command first performs a hardware reset, and then
invokes a software initialization routine on every processor
in the Array. The FEMDATA file is rewritten and an entry
message recorded, and the FEMSTATE file is updated to reflect
the Array status as determined by the processor response. Any
processor acknowledging a successful RESET is recorded as
"ON", all others as "UNKNOWN". RESET then compares the list
of "ON" processors to the working set defined by SETFEM and
reports any discrepancies as warnings (i.e. unexpected
response received - expected response not received).

An important feature of the RESET command is the definition
of logical to physical processor maps which support user
applications written in terms of logical processor numbers.
The map options now available include an identity mapping, a
default mapping, and a user specified mapping. The identity
(ID) mapping equates the logical processor to the physical
processor number and is generally used for diagnostic test
programs where the physical processor numbers must be known.
The default (DF) mapping simply maps the Nth available
physical processor onto the Nth logical processor as shown
below.

```
          1st available processor   =>   logical processor 1
          2nd available processor   =>   logical processor 2
                      .                              .
                      .                              .
                      .                              .
          Nth available processor   =>   logical processor N
```

The default mapping is intended for use where the
connectivity of local links is not a critical factor in the
performance of an algorithm. When an algorithm requires
assignment of processors to provide for specific connectivity
patterns on local links, the user specified (US) mapping can
be used to provide the required Array connectivity. The user
specified mapping is determined by the contents of a user
defined text file containing 36 processor numbers (one per
line), where each entry specifies the physical processor
number to be mapped into the corresponding logical position.
An example of such a mapping is given below.

User Map File	Physical Processor	Logical Processor
12	12	1
7	7	2
1	1	3
.	.	.
.	.	.
.	.	.
25	25	35
19	19	36

If the selected map option is not the identity map, the RESET
command will automatically define data area one and download
the   logical   to physical mapping to all active Array
processors. See the PASLIB PROGRAMMER'S GUIDE   for   an
explanation of the function of data area one in the logical
to physical mapping on the Array. The logical to physical map
and the startup messages from all operating processors are
then recorded in the FEMDATA file, and an array map depicting
the   current   state   of FEM is displayed at the user terminal
along with the initialization message from   the   first
processor to respond to the RESET command. See SFS (Show Fem
State) for interpretation of the symbols used in the array
map.

Prompts:

    SELECT MAP OPTION(DF,ID,US): (map option string)

Warnings/Limitations:

    Execution of a RESET will restart the Array and overwrite the
    SCRATCH.DATA file. RESET will complain about inconsistencies
    between processor response and the operational set as defined
    by SETFEM. Processor response errors should be brought to the
    attention of systems personnel and SETFEM used to update the
    set of working processors if necessary.

Usage:

    Use RESET at the start of a FEM session and whenever it is
    necessary or desirable to begin a task from scratch. RESET
    can also be used to restart the array in the event of an
    unrecoverable error.

3.3   SAC - Select Array Configuration

   Purpose:

      SAC selects a set of processors for subsequent operations   on
      FEM.

   Description:

      Select Array Configuration allows the user to specify a group
      of processors for subsequent operations. The user selects the
      group of processors to be enabled and SAC checks the state of
      the   Array   to   ensure   that the selected processors are in a
      legal state for selection. SAC will then turn "OFF" any   "ON"
      processor   not   selected and turn "ON" any "OFF" processor in
      the   selected   group.   The   resulting   Array   state   is   then
      displayed   at   the   user's terminal. An entry message and the
      processor select string are recorded in the FEM log file.

   Prompts:

      SELECT PROCESSOR(S): (processor select string)

   Warnings/Limitations:

      Selection of a processor in a   "HALTED"   or   "UNKNOWN"   state
      will   terminate SAC and cause appropriate error messages to be
      displayed.

   Usage:

      SAC   is   used to enable processor subsets when using commands
      that   affect   all   "ON"   processors.   For   example,   the
      XFEM(eXecute   FEM)   command   causes   all   "ON"   processors to
      commence execution. If the user only wants to   execute   on   a
      subset   of   currently available processors, he must first use
      the SAC command to disable the processors   that   are   not   to
      execute.

3.4   SETFEM - SET FEM working set

Purpose:

   To define the set of operational processors on the array.

Description:

   SETFEM updates the user defined set of operational processors
   stored in the SCRATCH.FEMSET file, which is normally
   protected against write and delete operations. SETFEM first
   removes the file protection, writes the user defined set,
   then restores the read only protection to the file. The
   SETFEM command is the only FACS command which always refers
   to physical processor numbers. The FEMSET file is used by the
   RESET command to determine when the array response is not as
   expected.

Prompts:

   SELECT WORKING SET OF PROCESSORS: (processor select string)

Warnings/Limitations:

   Failure to correctly define the set of operational processors
   may cause false warnings to be generated in the RESET
   command. Should RESET continually complain about missing or
   unexpected responses, verify the existing Array configuration
   and if necessary, use SETFEM to redefine the current
   processor set. NOTE: The processors must be identified by
   their PHYSICAL processor numbers in the SETFEM command.

Usage:

   SETFEM should be executed whenever the Array configuration is
   modified.

3.5   ASYNCON - ASYNChronous i/o CONnectivity

Purpose:

ASYNCON is used to select the asynchronous mode of I/O
communication and to allocate communication resources.

Description:

ASYNCON is used to select the asynchronous I/O mode, the
maximum I/O record size, the number of index tags allowed,
and the number of local links available for any subsequent
program execution on the Array. An entry message, the I/O
mode, and the attributes selected are written to the FEM log
file.   The Array state and any errors detected during ASYNCON
are displayed upon termination.

Prompts:

    MAXIMUM RECORD SIZE: (integer) (1 - 255)
     NUMBER OF INDEX TAGS: (integer) (1 - 255)
    NUMBER OF LOCAL LINKS: (integer) (0 - 12)

Warnings/Limitations:

The I/O attributes defined by this command remain in effect
until modified by a subsequent ASYNCON or SYNCON command, or
until cleared by the RESET or CLRFEM command (at which time
they are undefined). ASYNCON is broadcast to all "ON"
processors. Processor subsets can be selected using the SAC
command in the case where only a portion of the operational
processors are to receive the ASYNCON command.

Usage:

Use ASYNCON to define I/O attributes when you intend to
execute with asynchronous I/O communications.

3.6   AUTOSTAT - AUTOmatic STATus

Purpose:

AUTOSTAT will automatically set the program counter on selected processors to a user specified value.

Description:

AUTOSTAT is used to set the program counter on all operational processors to a user specified relative address. The user is prompted for a relative program counter value which is then transmitted to all active Array processors. If the new program counter value is a valid relative address within the user program space, the program counter is updated to the new value. An entry message and the selected program counter value are written to the FEM log file.

Prompts:

RELATIVE PC VALUE: (integer)

Warnings/Limitations:

The address specified in response to the RELATIVE PC VALUE prompt must be within the address space allocated to previously loaded object code.

Usage:

AUTOSTAT is used primarily to set Array program counters. Until NODAL EXEC is all on PROM, the non-resident segment of NODAL EXEC must be included in the application object code and linked in by executing a link routine on the Array. Once the code is linked, the user must set the program counter on the Array to the entry address of the N$MAIN module. This was formerly accomplished by using the STAT command to interactively set the program counters on each individual processor, which prevented the use of completely automated SCI application command files. The AUTOSTAT command allows the Array program counters to be updated from within an SCI command to support fully automated SCI procedures. See the PASLIB PROGRAMMER'S GUIDE for an explanation of the link phase of user programs on FEM.

3.7   CLRFEM - CLeaR FEM

Purpose:

   CLRFEM allows the user to free memory space allocated to object code, data areas, or communication buffers.

Description:

   CLRFEM allows the user to deallocate processor memory space allotted to object code, data areas, or communication buffers on Array processors. The user must select the processors and the objects to be cleared. CLRFEM writes an entry message, and records the processors and options selected in the FEMDATA file.

Prompts:

```
   SELECT PROCESSOR(S): ( processor select string )
    DELETE OBJECT CODE?: ( yes/no )
 DELETE DATA AREA(S)?: ( yes/no )
     DELETE NEIGHBORS?: ( yes/no )
```

Warnings/Limitations:

   None

Usage:

   CLRFEM can be used to eliminate memory fragmentation in multi-phase applications which require the execution of two or more user programs. See Programming Memo #2 for a discussion of the memory management algorithm and potential fragmentation problems on the Array.

3.8   DEFDAD/DEFDAI - Define Data Area

   Purpose:

      DEFDAD and DEFDAI are used to define data areas on FEM under
      the direction of a control file.

   Description:

      DEFDAD and DEFDAI are identical in function. Both commands
      define   data   areas   on the Array under the direction of a
      control file. However, DEFDAD expects a predefined control
      file and executes directly from the predefined file, while
      DEFDAI first calls the CDEF command to interactively create a
      temporary control file. In addition to the standard entry
      message, DEFDAD and DEFDAI write messages identifying the
      data area number, data type, number of items, and the
      processors involved to the data log for each data area
      defined on the Array.

   Prompts:

      DEFDAD prompt: CONTROL FILE ACCESS NAME: (access name)

      DEFDAI prompts: See CDEF

   Warnings/Limitations:

      DEFDAD requires an existing control file. Attempting to
      redefine a currently defined data area on the array will
      result in a FEM error.

   Usage:

      DEFDAD should be used in user defined SCI application
      procedures to support single command execution of user tasks,
      or whenever the control file for the required definition is
      known. DEFDAI is intended for use in an interactive FEM
      environment. Existing control files can be modified using the
      screen editor.

3.9   DELDAD/DELDAI - Delete Data Area

Purpose:

   DELDAD and DELDAI are used to delete data areas on FEM under
   the direction of a control file.

Description:

   DELDAD and DELDAI are identical in function. Both commands
   delete   data   areas   on   the Array under the direction of a
   control file. However, DELDAD expects a predefined control
   file and executes directly from the predefined file, while
   DELDAI first calls the CDEL command to interactively create a
   temporary control file. In addition to the standard entry
   message, DELDAD and DELDAI write messages identifying the
   data area number and the processors involved to the data log
   for each data area deleted on the Array.

Prompts:

   DELDAD prompt: CONTROL FILE ACCESS NAME: (access name)

   DELDAI prompts: See CDEL

Warnings/Limitations:

   DELDAD requires an existing control file. Attempting to
   delete a nonexistent data area on the array will result in a
   FEM error.

Usage:

   DELDAD should be used in user defined SCI application
   procedures to support single command execution of user tasks,
   or whenever the control file for the required deletion is
   known. DELDAI is intended for use in an interactive FEM
   environment. Existing control files can be modified using the
   screen editor.

3.10   LDAD/LDAI - Load Data Area

Purpose:

    LDAD and LDAI are used to load data files into predefined
    data areas on FEM under the direction of a control file.

Description:

    LDAD and LDAI are identical in function. Both commands load
    data areas on the Array under the direction of a control
    file. However, LDAD expects a predefined control file and
    executes directly from the predefined file, while LDAI first
    calls the CDNLD command to interactively create a temporary
    load control file. In addition to the standard entry message,
    LDAD and LDAI write messages identifying the data area
    number, the starting load index, the source file of the data,
    and the processors selected to the data log for each load
    operation.

Prompts:

    LDAD prompt: CONTROL FILE ACCESS NAME: (access name)

    LDAI prompts: See CDNLD

Warnings/Limitations:

    LDAD requires an existing control file. Attempting to load to
    an undefined data area on the array, or from an undefined
    data file on the Controller will result in an error. All data
    must reside in binary files on the controller.

Usage:

    LDAD is preferred in user defined SCI application procedures
    to support single command execution of user tasks, or
    whenever the control file for the required loads is known.
    LDAI is intended for use in an interactive FEM environment.
    Existing control files can be modified using the screen
    editor.

3.11   LDPG — LoaD ProGram

   Purpose:

      To download compressed, linked object code to FEM.

   Description:

      LDPG transmits a user defined file containing linked,
      compressed object code to selected processors on the Array.
      Data is transmitted in checksummed blocks for increased
      reliability. Upon completion of a successful load, the Array
      processors each transmit the program name and the load and
      entry addresses to the controller. This information is
      recorded in the FEM data log along with an LDPG entry
      message. The program name and FEM state are then displayed at
      the user terminal. In the event of an error, LDPG is
      terminated and the errors displayed at the terminal.

   Prompts:

      PROGRAM FILE ACCESS NAME: (access name)
           SELECT PROCESSOR(S): (processor select string)

   Warnings/Limitations:

      Any object code downloaded to the Array must be linked and
      compressed. Uncompressed object code will generate errors in
      LDPG.

   Usage:

      Use LDPG to download programs to the Array.

3.12   SYNCON - SYNChronous I/O CONnectivity

Purpose:

SYNCON is used to select the synchronous mode of I/O communication and to allocate communication resources.

Description:

SYNCON is used to select the synchronous I/O mode, the maximum I/O record size, the number of index tags allowed, the number of local links, and the maximum queue depth for any subsequent program execution on the Array. An entry message, the I/O mode, and the I/O attributes selected are written to the FEM log file. The Array state and any errors detected during SYNCON are displayed upon termination.

Prompts:

```
   MAXIMUM RECORD SIZE: ( integer ) (1 - 255)
    NUMBER OF INDEX TAGS: ( integer ) (1 - 255)
  NUMBER OF LOCAL LINKS: ( integer ) (0 - 12)
             QUEUE DEPTH: ( integer )
```

Warnings/Limitations:

The I/O attributes defined by this command remain in effect until they are changed by a subsequent SYNCON or ASYNCON command, or until cleared by the CLRFEM or RESET commands (at which time they are undefined). SYNCON is broadcast to all "ON" processors. Processor subsets can be selected using the SAC command in the case where only a portion of the operational processors are to receive the SYNCON command.

Usage:

Use SYNCON to define I/O attributes when you intend to execute with synchronous I/O communications.

3.13   HFEM - Halt FEM task

Purpose:

To halt the currently executing task on the Array.

Description:

The HFEM command is broadcast to the Array to halt active
user tasks on all processors. All "ON" processors are
expected to comply with the halt command and any "ON"
processor that fails to acknowledge suspension is identified
and reported. A message is written to the FEM log file on
entry, and a halt message with the time and date the Array
was halted is written upon successful termination.

Prompts:

None

Warnings/Limitations:

HFEM will generate errors when called without a task
executing on the Array.

Usage:

HFEM is used to terminate tasks after the user has escaped
from the XFEM or RFEM command without halting the currently
active task.

3.14   KFEM - Kill FEM task

Purpose:

To abort the execution of the currently running FEM task.

Description:

The KFEM command is broadcast to all processors in the Array. Any processor with an active task will terminate execution of that task and acknowledge the KFEM command. All "ON" or "HALTED" processors are expected to have resident tasks and are therefore expected to respond to this command. In addition to the entry message, a kill message is written to the FEM log indicating the time and date that the tasks were terminated.

Prompts:

None

Warnings/Limitations:

Invoking the KFEM command when there are no active or suspended tasks will generate errors on the Array.

Usage:

KFEM is used to kill tasks after the user has escaped from the XFEM or RFEM command with a task active or suspended.

3.15   RFEM - Resume FEM task

Purpose:

   To continue the execution of a halted FEM task.

Description:

   RFEM will cause programs suspended on the Array to resume
   execution. The execution options selected by the initial XFEM
   remain in effect for RFEM. The data log file, the execution
   data file, and the trace file (if trace is enabled) are all
   extended to provide a complete execution record. The error
   file is overwritten and is valid only in the context of this
   command. As in XFEM, RFEM writes trace samples to the trace
   file and all other data to the execution data file. Error
   reports are queued and processed upon exit. Data from the
   reference processor is displayed at the users VDT, with
   special key commands displayed across the top of the screen.
   RFEM terminates when the program on FEM terminates, when a
   fatal error is detected, or when a halt, kill, or escape
   command is issued at the user keyboard. An entry message is
   written to the FEM log file, along with the time and date for
   both beginning and ending execution and the cause of
   termination.

Prompts:

   None

Warnings/Limitations:

   RFEM is broadcast to all processors in the Array and is
   subject to the same constraints as XFEM.

Usage:

   Use RFEM to resume execution of suspended programs on the
   Array.

3.16   XFEM - eXecute FEM task

Purpose:

To start execution of user tasks on FEM.

Description:

XFEM starts execution of user programs on the Array, and allows the user to specify execution options in response to SCI prompts. Available options include execution traces at optional intervals, processor confidence checks, and processor monitoring.

Execution traces (program counter snapshots collected at regular intervals) can be enabled or disabled. If the trace option is enabled, the trace interval must be specified in milliseconds, and the user must specify which processors to trace. The checkin option enables or disables processor confidence checks. Selecting checkin causes the executing processors to send a special message to the controller every 60 seconds to verify that the Array is still operational. The reference processor specification identifies the processor that the user wants to monitor at his terminal during the program execution. All data from the reference processor is formatted and displayed on the VDT screen. If the user enters 0 in response to the reference processor prompt, XFEM will not monitor any processor and the overhead for maintaining the screen is eliminated. This can result in enhanced performance of algorithms transmitting large amounts of data to the Controller. Once the options are selected, RFEM transmits the execute command and option parameters to the Array. The execution options selected by the initial XFEM remain in effect throughout the life of the program, even in debug mode. The execution data file, error file, and the trace file (if trace is enabled) are all rewritten by XFEM. XFEM writes trace samples to the trace file if enabled, and all other data to the execution data file. Error reports are queued and processed upon exit. Data from the reference processor(s) is displayed at the users VDT, with special key commands displayed across the top of the screen. XFEM terminates when the program on FEM terminates, when a fatal error is detected, or when a halt, kill, or escape command is issued from the user terminal. An entry message is written to the FEM log file, along with the time and date for both beginning and ending execution and the cause of termination.

XFEM (continued)

Prompts:

            CHECKIN ENABLED: (yes/no)
              TRACE ENABLED: (yes/no)
             TRACE INTERVAL: (integer) (10 - 32760)
    REFERENCE PROCESSOR(S):   (integer) (0 - 36)
       TRACE PROCESSORS(S):   (processor select string)

Warnings/Limitations:

    Make sure that only processors that are ready to execute are
    "ON" before issuing the XFEM command. XFEM is broadcast to
    all processors in the Array and all "ON" processors receive
    the command and attempt to execute. The confidence check is
    intended for use in compute bound problems where long periods
    of silence are expected on the Array.   The Array processors
    generate a special "checkin" message every 60 seconds, and
    the Controller tests for these messages at 65   second
    intervals when checkin is enabled. If checkin is enabled for
    I/O bound programs, the Controller will complain about
    processors failing to check in on time. This is because
    checkin messages are uniquely tagged data words which are
    buffered with all other data coming from the Array. When
    large amounts of data are involved, checkins may fail to
    arrive on time. While this is not serious, it can be
    annoying, and users are advised to disable checkin when
    running I/O intensive programs to avoid false warnings. Note
    that this causes no loss of confidence since I/O can be
    considered to perform the same function as the checkin. The
    special key commands used to halt, kill, or escape execution
    may at times seem unresponsive. The halt, kill, and escape
    commands are immediately detected by the XFEM   command
    routine, but the action is not taken until all buffered data
    has been processed. Since the input data buffer is quite
    large and the Controller quite slow at processing the data
    from this buffer, the time between receipt of a special key
    command and compliance with that command can seem quite long.
    This condition is further complicated when a large number of
    errors have been queued since all errors are processed and
    expanded upon termination. If the Controller fails to respond
    immediately upon typing a special key command, give it a
    reasonable time to complete processing queued data.

Usage:

    Use the XFEM command to execute programs on the Array.

3.17   RDAD/RDAI - Read Data Area

Purpose:

RDAD and RDAI are used to upload the contents of a data areas
on FEM to a user specified directory, under the direction of
a control file.

Description:

RDAD and RDAI are identical in function. Both commands upload
data areas from the Array under the direction of a control
file. However, RDAD expects a predefined control file and
executes directly from the predefined file, while RDAI first
calls the CRDA command to interactively create a temporary
control file. The user must specify the data area number,
starting index within the data area, the number of items to
upload, the processor numbers, as well as a user directory
prefix to receive the data and a replace option to indicate
whether or not an existing file can be overwritten. The
filenames for data storage are generated by this program and
consist of the user supplied directory prefix with .DxxPyy
appended (where xx indicates the data area number and yy the
source processor number). Hence, uploading data area 3 with a
prefix of FEM.DATA causes the data for processor 7 to be
written to FEM.DATA.D03P07. All data files created are
sequential binary files, with a logical record length equal
to the size of the data item defined in the data area. In
addition to the standard entry message, RDAD and RDAI write
messages identifying the data area number, directory prefix,
and the processors selected to the data log for each data
area read from the Array.

Prompts:

RDAD prompt: CONTROL FILE ACCESS NAME: (access name)

RDAI prompts: See CRDA

Warnings/Limitations:

RDAD requires an existing control file. Attempting to upload
an empty or undefined data area on the array will result in a
FEM error.

Usage:

RDAD is preferred in user defined SCI application procedures
to support single command execution of user tasks, or
whenever the control file for the required definition is
known. RDAI is intended for use in an interactive FEM
environment. Existing control files can be modified using the
screen editor.

3.18 RES - Read Execution Statistics

   Purpose:

      RES is used to gather and process the execution statistics
      from the Array.

   Description:

      Whenever an Array program is executed the operating system on
      each processor collects execution statistics. RES allows the
      user to upload and analyze the execution statistics. The user
      must select the processors to interrogate for statistics, a
      destination for the output of the analysis, and specify the
      level of analysis to be performed. For details on the
      available statistics and levels of analysis refer   to
      Programming Memo No. 4. RES interrogates the Array one
      processor at a time for the execution statistics and writes
      the data to the SCRATCH.XSTATS file. RES then calls the
      analysis utility, and directs the analyzed data to the user
      specified listing file. Errors detected are displayed upon
      termination.

   Prompts:

      SELECT PROCESSOR(S): (processor select string)
      LISTING ACCESS NAME: (access name)
            FULL ANALYSIS: (yes/no)

   Warnings/Limitations:

      Execution statistics are available whenever a program has
      been run on the Array and remain available until the Array is
      RESET or another execute command is issued. When a processor
      begins execution it reinitializes all probes so   that
      execution statistics are only collected for the currently
      executing program.

   Usage:

      Use RES to gather and analyze system metrics.

3.19 SORT - SORT execution data file

Purpose:

  Sort is used to provide a list of execution data by
  processor.

Description:

  All text received from the array and any keyboard response to
  FEM queries during execution of a task on FEM is recorded in
  an execution data file (SCRATCH.EXDATA). When a task
  terminates on the Array, the SORT command is called to
  provide a listing of the execution session by processor. Sort
  scans the execution data file and gathers entries by
  processors. Responses to FEM queries by the Controller are
  preceded by a question mark in the sorted listing.

Prompts:

  LISTING ACCESS NAME: < access name >

Warnings/Limitations:

  None

Usage:

  Use SORT to provide ordered lists of execution sessions when
  a task on the Array terminates.

3.20   TRACE - process TRACE samples

   Purpose:

      To process program trace data from FEM executions.

   Description:

      The   TRACE   command   post   processes   the   trace   file
      (SCRATCH.TRACE)   to   generate   an   ordered   list   of   program
      counter   values   and   their   observed   frequencies   for   each
      processor sampled.

   Prompts:

      LISTING ACCESS NAME: (access name)
               MESSAGES: (access name)
                  MODE: (mode string)

   Warnings/Limitations:

      None

   Usage:

      Use TRACE to process trace samples collected during execution
      of a user program. The trace frequencies can   indicate   where
      the   user   program   is spending the most execution time, thus
      identifying   areas   where   code   optimization   will   pay   the
      greatest dividends.

3.21   FFR - Force Fem Release

Purpose:

FFR is used to force a station to release the Array.

Description:

The   FFR   (Force   Fem   Release)   command   is   used   to
unconditionally force a station to release FEM. FFR can   only
be   executed   by   a   user with system level privileges and is
included to allow termination of FEM sessions when a user   is
unwilling   or   unable to release the array (e.g. lost carrier
on a dialup line). The FFR command will delete the associated
files but cannot delete the synonyms for the locking user.   A
message   identifying   the   caller   and   station   forcing   the
release is written to the system log file.

Prompts:

None

Warnings/Limitations:

FFR should only be used when all other means   of   termination
have   failed.   The user forced to release the Array will lose
all data not copied into his own directory, but all   synonyms
in his workspace will remain defined.

Usage:

Use   to   force   the   release of FEM when all other means have
failed.

3.22   RELEASE - RELEASE fem

Purpose:

To release the FEM Array for use by others.

Description:

The RELEASE command unlocks FEM, deletes the associated synonyms and files, and writes a termination message to the system log file.

Prompts:

None

Warnings/Limitations:

FACS data files are deleted when the Array is released. Users are cautioned to save any FACS system files they wish to keep by copying them to their user directory (use the CC command) before releasing FEM.

Usage:

RELEASE is used to terminate FEM sessions. It should be the last command of any FEM session and will be issued by default if the user attempts to logout with FEM attached. Never RELEASE FEM unless you are sure you don't need the FACS system files (see warning above).

3.23   DAMA/DAMC — Dump Absolute Memory

Purpose:

   To dump absolute memory from selected Array processors.

Description:

   The DAMA/DAMC commands are identical in function as they both
   perform an absolute memory dump on the selected processors in
   the Array. The only difference between the commands is in how
   they specify the length of the memory segments to dump. DAMA
   identifies the segment with a starting and ending address,
   while DAMC uses a starting address and a word count. Address
   bounds can wrap-around through address 0. When the location
   counter references address FFFE the next fetch will access
   address 0. The starting and ending address specifications
   should be even addresses, and are decremented if odd. DAMA
   and DAMC fetch the contents of the specified address segment
   and format the data into rows containing eight words each,
   with the absolute memory address of the first word in each
   row to the left of that row, and the ASCII representation of
   the eight words on the right. In the event that the
   corresponding byte in the row does not map onto a printable
   ASCII character, the character position is filled with a
   period ('.'). Each processor is interrogated in turn, and
   the formatted data is written to the SCRATCH.DUMP file under
   the processor identification number. If the user selects the
   hardcopy option, the dump file is routed to the printer upon
   completion of the command. In any case, the data is routed to
   the user display while it is being collected and formatted,
   and the dump file is displayed when all data has been
   gathered using the DX10 SF (show file) command.

Prompts:

      DAMA Prompts: STARTING ADDRESS: (integer)
                      ENDING ADDRESS: (integer)
                  SELECT PROCESSOR(S): (processor select string)
                           HARDCOPY?: (yes/no)

      DAMC Prompts: STARTING ADDRESS: (integer)
                     NUMBER OF WORDS: (integer)
                  SELECT PROCESSOR(S): (processor select string)
                           HARDCOPY?: (yes/no)

Warnings/Limitations:

   None

Usage:

   Use DAMA/DAMC to dump absolute segments of Array memory.

3.24   DRMA/DRMC - Dump Relative Memory

   Purpose:

      To dump relative memory from selected Array processors.

   Description:

      The DRMA/DRMC commands are identical in function as they both
      perform a relative memory dump on the selected processors in
      the Array. The only difference between the commands is in how
      they specify the length of the memory segments to dump. DRMA
      identifies the segment with a starting and ending address,
      while DRMC uses a starting address and a word count. Starting
      and ending address specifications should be even addresses,
      and are decremented if odd. DRMA and DRMC fetch the contents
      of the specified address segment and format the data into
      rows containing eight words each, with the relative memory
      address of the first word in each row to the left of that
      row, and the ASCII representation of the eight words on the
      right. In the event that the corresponding byte in the row
      does not map onto a printable ASCII character, the character
      position is filled with a period ('.'). Each processor is
      interrogated in turn, and the formatted data is written to
      the SCRATCH.DUMP file under the processor identification
      number. If the user selects the hardcopy option, the dump
      file is routed to the printer upon completion of the command.
      In any case, the data is routed to the user display while it
      is being collected and formatted, and the dump file is
      displayed when all data has been gathered using the DX10 SF
      (show file) command.

   Prompts:

      DRMA Prompts: STARTING ADDRESS: (integer)
                      ENDING ADDRESS: (integer)
                  SELECT PROCESSOR(S): (processor select string)
                            HARDCOPY?: (yes/no)

      DRMC Prompts: STARTING ADDRESS: (integer)
                     NUMBER OF WORDS: (integer)
                  SELECT PROCESSOR(S): (processor select string)
                            HARDCOPY?: (yes/no)

   Warnings/Limitations:

      The relative memory address space is defined when object code
      is downloaded to the Array. Calling these commands with no
      relative memory space defined will cause an error.

   Usage:

      Use DRMA/DRMC to dump relative segments of Array memory.

3.25   MFR - Modify Fem Registers

Purpose:

MFR is used to inspect and change the status block and/or
current workspace registers on processors in the Array.

Description:

The   MFR   command   fetches   the   status   block   and   current
workspace registers for   the   selected   Array   processor   and
displays   their   hexadecimal values at the user terminal. The
user may then alter the workspace pointer,   program   counter,
or   status   word in the status block, or any of the currently
displayed workspace registers   in   the   register   block.   MFR
operates in two modes, either register mode or status mode.

Initially,   the MFR command comes up in the register mode. In
register mode the user   can   move   the   cursor   to   the   next
register,   back   to   the previous register, copy the register
above,   fetch and display the status and register   values   for
either   the   next   or   the   last   "ON"   processor in the state
table, enter a new processor number for access, or switch   to
status   mode.   A   menu is displayed on the screen identifying
the key commands   necessary   to   invoke   these   functions   in
register   mode.   Any function which precipitates an exit from
register mode causes the   displayed   register   values   to   be
written to the Array processor.

The   status   mode   is similar to register mode except that it
operates on the status   block   information.   As   in   register
mode,   status   mode   displays   a   menu   listing the available
operations. In status mode you can move the cursor ahead   and
back   through   the   modifiable fields (WP,PC, and ST), select
the next, last, or user defined processor for   interrogation,
and   in the case of the workspace pointer and program counter
fields, cause the displayed values to   reflect   either   their
absolute   or   relative   values (provided that relative memory
exists on the processor and that the absolute value maps into
the relative address space). To return to register mode   from
the   status   mode,   press   return when the cursor is over the
status field of the status block. Any changes in   the   status
block   will   be   transmitted   to   the   Array processor when a
function is called that causes MFR to exit   status   mode.   If
the   value   of   the   workspace pointer was modified, MFR will
fetch a new set of   workspace   registers   when   returning   to
register mode. To terminate the MFR command, simply press the
CMD   key.   In   addition to the entry message, MFR records the
initial and modified values for any   registers   modified   and
the processor number on which those changes were made.

MFR (continued)

Prompts:

PROCESSOR NUMBER: (integer)

Warnings/Limitations:

The user is cautioned not to indiscriminately alter workspace
pointers in the debug mode since doing so may invalidate the
suspended process state.

Usage:

MFR is intended primarily to afford a user interactive access
to processor registers in a suspended user task.

3.26   MMA/MMR — Modify Memory

Purpose:

   To inspect and change processor memory on the Array.

Description:

   The MMA/MMR commands are identical in function in that both
   allow the user to inspect and change memory on a processor in
   the Array. However, MMA operates on absolute memory
   addresses, and MMR operates on relative memory space. The
   user specifies the starting address (should be even —
   decremented to a word boundary if odd), the processor number,
   and the replace option. If the global replace option is
   enabled, any memory locations changed in the selected
   processor are also changed in all "ON" and "HALTED"
   processors. Once the parameters have been entered, MMA/MMR
   fetches sixteen words of memory from the selected processor
   and displays them along with their corresponding addresses at
   the user terminal. The cursor is positioned over the first
   location in the list and an option menu is displayed on the
   right hand side of the screen. Menu functions support moving
   the cursor through the list, copying values, changing the
   replace mode, paging through memory, selecting new
   processors, and changing the address pointer. The cursor can
   be moved either up or down the list. Moving forward to the
   next location when the cursor is over the last item will
   cause the next block of 16 locations to be fetched and the
   cursor set to the top of the list. Moving to the preceding
   location while the top of the list will fetch the preceding
   block of 16 locations and place the cursor at the top of the
   list. The copy function copies the value of the preceding
   location into the current location. When a new block of
   locations is called, the last value of the preceding block is
   saved and can be copied provided that the new fetch was a
   forward reference. The replace option can be turned on and
   off at any time. Selecting the page function will cause the
   next block of 16 locations to be fetched for next page, and
   the preceding block of 16 locations for a last page. New node
   allows a new processor number to be selected, and new address
   allows a new address space to be inspected. An entry message
   is written to the FEM log file, and each location that is
   modified is recorded with the initial and final values,
   processor number, global replace option, and type
   (relative,absolute) of memory.

MMA/MMR (continued)

Prompts:

STARTING ADDRESS: (integer)
PROCESSOR NUMBER: (integer) (1 - 36)
  GLOBAL REPLACE: (yes/no)

Warnings/Limitations:

MMR will not allow the user to go beyond the bounds of
relative memory.

Usage:

Use MMA/MMR to inspect and change memory on FEM in support of
debugging.   Memory   can   be   modified   on   all   processors
simultaneously using the global replace option.

3.27   SFB — Set Fem Breakpoints

   Purpose:

      SFB allows the user to specify up to two breakpoint addresses
      for each processor in the Array.

   Description:

      SFB allows the user to examine, set, and clear breakpoint
      addresses   on   the   Array.   This command can be used to set
      breakpoints on a single processor, or on all operational
      processors in the Array. If the global replace option is
      selected, all "ON" and "HALTED" processors are interrogated
      for breakpoints and the breakpoints from the reference
      processor are displayed at the user terminal.   The   user   can
      examine the breakpoints on the reference processor and either
      enter new values or clear old breakpoints. When the second
      breakpoint has been entered, or when the CMD key is pressed,
      SFB   will   transmit any modified breakpoint addresses to the
      Array. Any changes in breakpoint addresses are recorded with
      the processor number and global replace mode in the FEM log
      file. When the global replace mode is not selected, the SFB
      command operates only on the reference processor. Breakpoint
      addresses must be relative to the user task. Attempting to
      define breakpoints that do not map onto relative address
      space will result in an error. A blank field is interpreted
      as a null breakpoint.

   Prompts:

      REFERENCE PROCESSOR: (integer)
            GLOBAL REPLACE: (yes/no)

   Warnings/Limitations:

      Breakpoint addresses must map into a relative address space
      or an error will occur. A breakpoint address is cleared when
      a processor halts on that address. If you need to stop
      repeatedly at one breakpoint while debugging, you must
      re-enter the address after each occurence of that breakpoint
      address.

   Usage:

      Use SFB to define breakpoints as an aid in debugging user
      programs.

3.28   SFR - Show Fem Registers

Purpose:

SFR   fetches and displays the contents of the current
workspace registers and status block for all operational
processors in the Array.

Description:

The   SFR   command interrogates all operational proccessors in
the Array to obtain the contents of their current workspace
registers and   status   block, formats the data in processor
sequence, and   writes it to a temporary file. SFR then
displays the status and registers at the user terminal and
appends the temporary file to the FEM log file.

Prompts:

None

Warnings/Limitations:

None

Usage:

Use SFR to record the status block and register state of all
operational processors.

3.29   SFS - Show Fem State

Purpose:

SFS displays the current state of the FEM Array.

Description:

SFS reads the SCRATCH.STATE file to obtain the last known state of the FEM Array and then displays a graphic representation of that state at the user terminal. The display consists of 36 numbered compartments, each containing a processor number and a graphic symbol to represent the current processor state. The four possible states and the symbols used to represent them are listed below.

State	Symbol	Meaning
=========	======	=======
ON	▮	The processor is ready for use. No task is suspended and no fatal errors have been detected since the last RESET.
OFF	▬	The processor has been disabled by the SAC command and cannot be accessed until it is re-enabled or RESET.
HALTED	/	The processor was executing a user task and either encountered a breakpoint or was commanded to halt execution.
UNKNOWN	+	The processor either failed to respond to a valid FACS command or suffered a fatal error. Processors in an unknown state can only be RESET.

Prompts:

None

Warnings/Limitations:

None

Usage:

Use SFS to determine the current state of the Finite Element Machine

3.30   SPSF - Show Pascal Stack on Fem

Purpose:

SPSF allows the user to inspect and change PASCAL stacks on
Array processors.

Description:

The SPSF command allows the   user   to   examine   the process
record and inspect and change the PASCAL stack frame and task
status block in a halted task on the Array. SPSF requests a
processor number, then displays the   stack, process record,
and status block as shown in Figure 1. An explanation of the
data structures for the stack and process record can be found
in section 8 of the DX10 TI PASCAL PROGRAMMER'S GUIDE.   The
SPSF command operates   on   the   stack registers, the PASCAL
stack, or on the task status block with an   appropriate   menu
displayed in   each   of   the   three possible modes. Figure 1.
represents the register mode which is the   default   mode   for
the SPSF command.

In the   register   mode, the user can move the cursor forward
and back through the register list (skipping over registers 9
and 10 since they must   not   be   altered),   change registers
under   the   cursor, copy a value from the preceding register,
sequence to the next, last, or a   user   specified   processor,
toggle   the   register   contents   between   their   relative and
absolute representations where relative addresses are   valid,
and change modes to stack mode.

In the   stack   mode, the cursor is positioned over the first
entry in the list of PASCAL stack entries, and the   user   can
move the cursor through the list, change any value under the
cursor, copy preceding fields, fetch either the   next   or
previous 16 locations in the stack (provided they exist - the
end of the stack is indicated by "End of Stk"), select a new
processor,   toggle   between   relative   and   absolute
representations   of   data   where a   relative address can be
determined, or proceed to the status mode.

The status mode allows the user to modify the WP, PC,   or   ST
registers,   select   a   new   processor,   toggle   relative and
absolute representations of data, or return to register mode.

SPSF (continued)

Prompts:

PROCESSOR NUMBER: (integer) (1 - 36)

Warnings/Limitations:

SPSF will not allow the user to change data that absolutely
should not be altered. However, there are a number of
locations that can still cause considerable trouble if not
properly defined. Changing the workspace pointer will cause
the workspace registers to be updated when leaving the status
mode.

Usage:

Use SPSF as a debugging tool for suspended user tasks on FEM.

```
SHOW PASCAL STACK ON FEM PROCESSOR                    Processor

WP=         PC=         ST=         PSV=        FREE=       OBJ=

        Process Record                      Stack                Function Keys
==========================    =====================================    ================
LINK:              WP:        R0:       SAVED:                F1-next field
   D1:             PC:        R1:       CALLER:               F2-last field
   D2:             ST:        R2:       RETADX:               F3-go to stack
   D3:             RS:        R3:       PRP:                  F4-copy above
   D4:          FLAGS:        R4:                             F5-next node
   D5:         NONSTD:        R5:                             F6-last node
   D6:         STKBLK:        R6:                             F7-new node
   D7:         SBNDRY:        R7:                             F8-rel/abs adx
   D8:         STKMAX:        R8:                             CMD-terminate
   D9:           FTOP:        BOT:
  D10:           QLNK:        TOP:
  D11:          PHEAP:        R11:
  D12:          PTASK:        R12:
  D13:          CODES:        R13:
  D14:                        R14:
  D15:                        R15:
  D16:
```

Figure 1.   Screen Template for the SPSF Command.

3.31   SS - Single Step task on fem

Purpose:

   The SS command causes a user specified number of instructions
   to be executed in single step mode on the Array.

Description:

   The SS command causes all "HALTED" processors to execute one
   or more instructions as directed by the user, reporting the
   contents of the processor status block upon completion of
   each single instruction step. SS is similar to an XFEM
   command in that interactive I/O is supported, trace samples
   can be obtained, and breakpoints trapped, but differs in
   respect to the reference processor support. In SS, the user
   can elect to monitor any number of processors in the Array.
   When the command is invoked you must specify the processors
   to be referenced, then SS will prompt for the desired number
   of steps. Entering O for number of steps terminates SS. If
   number of steps is nonzero, SS will direct the Array to
   repeatedly execute single instructions, until the requested
   number of steps have been executed. After each instruction,
   each active processor on the Array reports the resulting
   values of its WP, PC, and ST registers. This information is
   formatted and displayed at the user terminal for   all
   reference processors, and the status information for all
   active processors is written to the execution data file.

Prompts:

   REFERENCE PROCESSOR(S): (processor select string)

Warnings/Limitations:

   The user should avoid the single step mode of operation in
   systems   routines. Refer to the reverse assembler listing for
   the code you are debugging and breakpoint around calls to
   PASLIB.

Usage:

   Use SS to single step user tasks as an aid to debugging.

3.32   STAT - inspect & change processor STATus block

Purpose:

   STAT allows the user to inspect and change the status blocks
   on each processor in the Array.

Description:

   This functionality of this command is supported   in several
   other   commands   on   a   single   processor   level.   STAT
   simultaneously presents the status information for   all   "ON"
   and "HALTED" processors in the Array. Processors with "OFF"
   or "UNKNOWN" status are so identified. There is currently   no
   menu   implemented on the STAT command because of screen space
   limitations. The key commands and their associated   functions
   are listed below.

                    KEY   FUNCTION
                    ===   =========================
                    F1   move to next register
                    F2   move to last register
                    F3   copy field above
                    F4   copy block above
                    F5   move to next processor
                    F6   move to last processor
                    F7   do nothing
                    F8   toggle relative/absolute
                    <CR>   same as F1
                    CMD   update status and quit

Prompts:

   None

Warnings/Limitations:

   Caution is advised when modifying the status of suspended
   processors.

Usage:

   Use STAT to inspect & change the status block of any
   processor not in execution mode.

3.33   CDEF - Create DEFine data area control file

Purpose:

To create control files for use by DEFDAD/DEFDAI.

Description:

CDEF is an interactive utility used to generate control files
which direct the definition of data areas on FEM. CDEF
prompts for the data area number, the data type, the maximum
number of data items expected in the data area, which
processors to define the data areas on, and whether or not to
continue CDEF after the current entry. If a user defined data
type is selected an additional prompt is made to obtain the
size of the data item in words. After each set of parameters
is entered the entry is written to a temporary file
(SCRATCH.BATLST), and CDEF requests the next set of values if
continued. If CDEF was not continued, the user is prompted to
determine whether or not he wants to save the define sequence
just generated. If the save option is selected, CDEF prompts
for a file access name and writes the temporary file contents
to the user designated file. If the save option is not
selected, CDEF simply exits. In either case the temporary
file will retain the define sequence generated, and is
utilized as the control file when executing the DEFDAI
command.

Prompts:

         SELECT DATA AREA NUMBER: (integer) (0 - 31)
                SELECT DATA TYPE: (data type)
     ENTER NUMBER OF DATA ITEMS: (integer)
            SELECT PROCESSOR(S): (processor select string)
                 CONTINUE INPUT?: (yes/no)
===
                SELECT ITEM SIZE: (integer)
===
          SAVE THIS TRANSACTION?: (yes/no)
              FILE ACCESS NAME: (access name)

Warnings/Limitations:

CDEF will overwrite any existing file specified in response
to the file access name prompt.

Usage:

Use CDEF to generate control files for the DEFDAD command.
CDEF is automatically called by the DEFDAI command.

3.34   CDEL — Create DELete data area control file

Purpose:

To create control files for use by DELDAD/DELDAI.

Description:

CDEL is an interactive utility used to create a control file
to supervise the deletion of data areas on FEM. CDEL prompts
for the data area number, the processors on which the data
area is to be deleted, and whether or not to continue CDEL
after the current entry. After each set of parameters is
entered,   the   entry   is   written to   a temporary file
(SCRATCH.BATLST) and CDEL prompts for the next set of values
if CDEL was continued. If CDEL was not continued, the user is
prompted to determine whether or not he wants to save the
delete sequence just generated. If the save option is
selected, CDEL prompts for a file access name and writes the
temporary file contents to the user designated file. If the
save option is not selected CDEL simply exits. In either
case, the temporary file retains the   delete   sequence
generated and is utilized as the control file when executing
the DELDAI command.

Prompts:

        SELECT DATA AREA NUMBER: (integer) (0 – 31)
            SELECT PROCESSOR(S): (processor select string)
              CONTINUE INPUT?: (yes/no)
    ===
        SAVE THIS TRANSACTION?: (yes/no)
            FILE ACCESS NAME: (access name)

Warnings/Limitations:

CDEL will overwrite any existing files specified in response
to the file access name prompt.

Usage:

Use CDEL to generate control files for the DELDAD command.
CDEL is automatically called for the DELDAI command.

3.35   CDNLD - Create DowNLoaD data area control file

Purpose:

To create control files for use by LDAD/LDAI.

Description:

CDNLD is an interactive utility used to create the control
files used in the LDAD/LDAI commands. The user must enter the
data area number, the index offset for the first item to be
loaded, the data file containing the information to be
downloaded, the processors to load, and whether or not to
continue CDNLD. After each set of entries, the parameters are
written to a temporary file (SCRATCH.BATLST). If the user
elects to continue CDNLD he is prompted for the next set of
parameters. Otherwise, the user is prompted to determine
whether or not he wants to save the command file in his
directory. If the file is to be saved, CDNLD prompts for a
file access name and stores the newly created command file as
directed. In any event, the temporary file retains the
command sequence generated and is used as the control file in
the LDAI command.

Prompts:

```
       SELECT DATA AREA NUMBER: ( integer ) ( 0 - 31 )
    ENTER STARTING INDEX VALUE: ( integer ) ( 1 - 32767)
           SELECT PROCESSOR(S): ( processor select string )
     ENTER DATA FILE PATHNAME: ( access name )
               CONTINUE INPUT?: ( yes/no )
==========================================================
        SAVE THIS TRANSACTION?: ( yes/no )
              FILE ACCESS NAME: ( access name )
```

Warnings/Limitations:

CDNLD will overwrite any existing files specified in response
to the file access name prompt.

Usage:

Use CDNLD to generate control files for the LDAD command.
CDNLD is automatically called by the LDAI command.

3.36   CRDA - Create Read Data Area control file

   Purpose:

   Create control files for RDAD/RDAI.

   Description:

   CRDA is an interactive utility used to create control files
   which supervise the uploading of data areas from the Array.
   CRDA prompts the user to determine the data area number, an
   index offset into the data area, the number of items to
   upload, the processors to interrogate, a directory prefix
   identifying a user directory where the necessary data files
   can be created (see RDAD/RDAI), a replace option enabling or
   disabling the overwriting of existing data files, and whether
   or not to continue CRDA after the present entry. Each entry
   is written to a temporary file (SCRATCH.BATLST) as it is
   entered, and when the control file is complete the user is
   given the option of saving or not saving the newly created
   file. If the command file is to be saved it is copied to the
   user designated file. If the command file is not to be saved,
   CRDA simply exits. In either case, the temporary control file
   retains the generated command sequence and is used as the
   control file in the RDAI command.

   Prompts:

           SELECT DATA AREA NUMBER: (integer) (0 - 31)
        ENTER STARTING INDEX VALUE: (integer) (1 - 32767)
        ENTER NUMBER OF DATA ITEMS: (integer) (1 - 32767)
                SELECT PROCESSOR(S): (processor select string)
             ENTER DIRECTORY PREFIX: (directory prefix name)
                           REPLACE?: (yes/no)
                     CONTINUE INPUT?: (yes/no)
   ==
            SAVE THIS TRANSACTION?: (yes/no)
                   FILE ACCESS NAME: (access name)

   Warnings/Limitations:

   CRDA will overwrite any existing files specified in response
   to the file access name prompt.

   Usage:

   Use CRDA to generate control files for the RDAD command. The
   RDAI command automatically calls CRDA.

APPENDIX A

FACS PROMPT TYPES

        The FACS commands are embedded in the System Command
Interpreter of the DX10 operating system, and many FACS commands
utilize SCI prompts to obtain their parameters. The command
descriptions list type identifiers for the required prompts and
indicate a subrange of permissible input values. The following is
a description of the FACS prompt type identifiers and the range
of permissible values associated with that type.

(integer)

   Integer is a 16 bit integer value with a useful range of 0 to
   32767 in FACS commands. The integer value is considered to be a
   decimal value unless it is preceded by either a 'O' (i.e. 0423)
   or a "greater than" symbol '>' (i.e. >423), either of which
   identifies it as a hexadecimal value.

(yes/no)

   Yes/no is simply a character string where the first character
   indicates a true/false response to a tendered question. The
   interpretation   of   the   yes/no parameter is left to an
   IF-THEN-ELSE clause in the SCI procedure. SCI will only accept
   a "Y" or "N" in the first position of the yes/no prompt and the
   interpretation of such characters depends upon the command
   procedure.

(access name)

   The access name is a variable length string of up to 46
   characters used to identify either a file or an I/O device.
   Legal I/O device specifications are LP01 (the line printer), ME
   (the user terminal), or DUMY (the bit bucket).

(data type)

   The data type is a single character which identifies the type
   of data a data area is to contain and is used exclusively in
   the CDEF command. The only permissible values are I (integer),
   L (long integer), R (real), D (double precision real), and U
   (user defined type). No other input is accepted for this
   prompt.

(mode string)

The mode string is a character string used to indicate either
the Foreground or Background mode of execution on the
Controller. As in the yes/no prompt, the interpretation of the
response is up to the SCI command procedure. Only the first
character is interpreted, and the expected response is either
an 'F' or a 'B'.

(processor select string)

The processor select string is a string of up to 46 characters
used to select a set of Array processors. The user can type
'ALL' to indicate all operational processors, or enter specific
processor numbers. Selected processors can be identified by
individual number (i.e. '12'), by a range of processor numbers
(i.e. '4..15'), or by a mixture of the two (i.e.
'1,4,7,12..18,3,5..6').

(map option string)

The map option string is used to identify the map option for
the RESET command. Only the first character is interpreted in
the map option string.

(directory prefix name)

The directory prefix name is a character string of up to 42
characters identifying a user directory pathname.

APPENDIX B

FACS SYSTEM FILES

        A number of files are maintained for the user by FACS. It is
important that the user understand the purpose of each of these
files and their lifespan to avoid overwriting desired data. Each
of the FACS files is listed below, along with its purpose and
span of existence.

SCRATCH.BATLST

    Purpose:

        SCRATCH.BATLST   is a temporary control file created by
        interactive data area commands. If the user chooses to save
        the batch file, SCRATCH.BATLST is copied to a user defined
        file.

    Lifespan:

        SCRATCH.BATLST is rewritten whenever a command is called that
        uses it. Therefore, only the last batch file generated is
        available unless the user saves them in his directory.

SCRATCH.DATA

    Purpose:

        The SCRATCH.DATA file is used by FACS to log an entire FEM
        session. Each FACS command called writes an entry message to
        the FEM log file and records any pertinent information for
        later review. For example, the LDPG command writes an entry
        message, records the name of the file downloaded and the
        affected processors, and records the returned text and status
        information. Errors detected during the execution of any FACS
        command are also written to the data file in abbreviated
        format. The SCRATCH.DATA file thus contains a record of an
        entire FEM session and is extremely helpful in reconstructing
        transactions on FEM.

    Lifespan:

        SCRATCH.DATA is created when the user attaches FEM and is
        rewritten each time the RESET command is issued. If problems
        are encountered on the array the user is advised to either
        print the data file, or copy it into his directory for later
        review. The log file is deleted when the Array is released.

SCRATCH.DUMP

Purpose:

SCRATCH.DUMP provides temporary storage for memory and register dumps, and is used to record results for most diagnostic routines.

Lifespan:

SCRATCH.DUMP is created by the ATTACH command and deleted when the Array is released. In addition, the dump file is overwritten whenever a command is called that uses it.

SCRATCH.ERRORS

Purpose:

SCRATCH.ERRORS is used to record expanded error messages. All errors detected while executing any FACS command are queued and processed just prior to exiting the command. If any error is detected during the execution of FACS commands, the condition code is set to indicate the source. For all FACS generated errors, the SCRATCH.ERRORS file is displayed when control is returned to DX10.

Lifespan:

The SCRATCH.ERRORS file is valid only for the last command executed. This file is created when the Array is attached and deleted when FEM is released. The errors file is rewritten at the start of each FACS command.

SCRATCH.EXDATA

Purpose:

SCRATCH.EXDATA is used to record all data transactions between the Array and Controller during the execution of any user program. Data originating at the keyboard will be in response to a query from the Array and is recorded with a -1 as the data source identifier. All other data is recorded with the originating processor number as its source. This file is postprocessed by the SORT utility to provide an execution record for each participating processor.

Lifespan:

SCRATCH.EXDATA is deleted and re-created each time the XFEM
command is invoked. Therefore, the contents of this file are
only available until the next XFEM and should be sorted and
either printed or copied into the users' directory to prevent
loss. Resuming the execution of (RFEM), or single stepping
(SS) a halted program will cause the execution data file to
be extended, thus providing a complete record of the
execution of a program, even in debug mode.

SCRATCH.FEMSET

Purpose:

SCRATCH.FEMSET defines the set of physical processors that
are installed in the Array. This file is defined by the
FEMSET command and is only referenced by the RESET command.

Lifespan:

SCRATCH.FEMSET is a permanent FACS file and must not be
deleted.

SCRATCH.GRIDMAP

Purpose:

SCRATCH.GRIDMAP is the template for the Array status report
displayed in many FACS commands.

Lifespan:

This file is permanent and must NOT be deleted or altered.

SCRATCH.STATE

Purpose:

SCRATCH.STATE is used to record the Array state upon exit
from FACS commands. This is important for the continuity of
commands during the FEM session because each FACS command
references the state file to determine the current status of
the Array.

Lifespan:

> The SCRATCH.STATE file always exists. It is initialized by
> the RESET command and is subsequently used by all other
> commands to test Array status upon entry, and to record Array
> status upon exit. This file must NOT be altered or deleted by
> the user.

SCRATCH.TRACE

Purpose:

> The SCRATCH.TRACE file is used to record all trace
> information returned by program(s) executing on the Array.
> Each trace sample received by the Controller is written to
> the trace file with an identifying source processor number.
> This file is then post-processed to provide trace information
> for each participating processor.

Lifespan:

> The SCRATCH.TRACE file is created when FEM is attached and
> deleted upon release of the Array. The trace file is
> rewritten on an XFEM command and extended when a halted
> program is restarted.

SCRATCH.XSTATS

> The SCRATCH.XSTATS file is used to store execution statistics
> gathered from processors on the Array.

Lifespan:

> SCRATCH.XSTATS is a temporary file created when the RES
> command is issued, and deleted when the statistics have been
> analyzed.

SYS1.FEM.ERROR

Purpose:

The SYS1.FEM.ERROR file is a relative record file which contains the text of all FEM error messages. Upon completion of any FACS command, detected errors are dequeued and SYS1.FEM.ERROR is referenced to provide the text for the error messages.

Lifespan:

This file is permanent and must NOT be deleted or altered.

APPENDIX C

FACS SYSTEM SYNONYMS

$BATLST

Purpose:

Identifies the control file for interactive data area commands.

Usual value:

SCRATCH.BATLST

$CHECK

Purpose:

Stores the user response to the CHECKIN option in program execution commands.

Usual value:

Yes or No.

$CRTFILE

Purpose:

Identifies the Video Display Terminal.

Usual value:

STxx (where xx is the user station number).

$ERRLST

Purpose:

Identifies the error message reference file.

Usual value:

SYS1.FEM.ERROR

$ERRTST

Purpose:

Stores the command completion code. A nonzero value in $ERRTST indicates an abnormal command termination. This synonym is very useful when writing SCI procedures because it is set by all FACS commands to indicate error conditions detected during execution. Thus $ERRTST can be tested by an SCI procedure and a sequence of commands aborted when errors occur. See Appendix D. for clarification.

Usual value:

Assigned under FACS program control.

$FEMDATA

Purpose:

Identifies the FEM data log file.

Usual value:

SCRATCH.DATA

$FEMDUMP

Purpose:

Identifies the FEM dump file.

Usual value:

SCRATCH.DUMP

$FEMERR

Purpose:

Identifies the error listing file.

Usual value:

SCRATCH.ERRORS

$FEMSET

Purpose:

Identifies the operating processor set file.

Usual value:

SCRATCH.FEMSET

$GMAP

Purpose:

Identifies the file containing the state display template.

Usual value:

SCRATCH.GRIDMAP

$PFILE

Purpose:

Identifies the file containing compressed object code to be
downloaded to FEM.

Usual value:

Set by LDPG.

$REFPROCESSOR

Purpose:

Defines the reference processor for execution commands.

Usual value:

User defined.

$STATE

Purpose:

Stores the current state of the array.

Usual value:

SCRATCH.STATE

$TRACE

Purpose:

Identifies the file used for temporary storage of trace
samples.

Usual value:

SCRATCH.TRACE

$XDATA

Purpose:

Identifies the file used to store execution data.

Usual value:

SCRATCH.EXDATA

$XSTATS

Purpose:

Identifies the temporary file used to store execution
statistics.

Usual value:

SCRATCH.XSTATS

APPENDIX D

CUSTOM SCI PROCEDURES

In this example, the user has created a procedure called
"EASYRUN", which will execute when "EASYRUN" is submitted in
response to an SCI prompt. SCI will immediately display "RUNNING
A FEM TASK THE EASY WAY" on the user's screen and prompt for the
three input parameters needed by the procedure. The procedure
then resets the Array, clears superfluous synonyms, selects the
desired processor configuration, downloads the object code for
the user program, defines and downloads the required data areas,
establishes the I/O mode and connectivity, and executes the user
program on the Array. When the user program terminates, EASYRUN
sorts the execution data generated by the user task, stores the
sorted data in a user designated file, and creates a hardcopy of
the output. Since each individual FACS command is required to set
the $ERRTST synonym and display the appropriate error messages
when errors are detected, it is sufficient in EASYRUN to simply
test $ERRTST and abort the procedure should an error occur.

```
EASYRUN (RUNNING A FEM TASK THE EASY WAY)=3,
SYNCHRONOUS I/O? = YESNO(YES),     ! set all necessary parameters
SELECTED PROCESSORS = STRING,      !
REFERENCE PROCESSORS = STRING      !
P$SYN                              ! clear all unecessary synonyms
Q$SYN                              !
RESET SELMAPOPT = DF               ! reset the array
.IF @$ERRTST, NE, 00000            ! quit if reset failed
.EXIT                              !
.ENDIF                             !
SAC SN="&SELECTED PROCESSORS"      ! select processors to run
.IF @$ERRTST, NE, 00000            ! quit if select failed
.EXIT                              !
.ENDIF                             !
LDPG                               ! load the user program
  PFAN="FEM.THISIS.MYCODE",        !   from this file
  SN="&SELECTED PROCESSORS"        !   to these processors
.IF @$ERRTST, NE, 00000            ! quit if download failed
.EXIT                              !
.ENDIF                             !
DEFDAD                             ! define data area directly
  CFAN="FEM.THISIS.DEFINE"         !   from this control file
.IF @$ERRTST, NE, 00000            ! quit if define fails
.EXIT                              !
.ENDIF                             !
```

```
LDAD                                  ! load data area directly
 CFAN="FEM.THISIS.DOWNLOAD"           !  from this control file
.IF @$ERRTST, NE, 00000              ! quit if download fails
.EXIT                                 !
.ENDIF                                !
.IF "&SYNCHRONOUS I/O?",GE, "Y"      ! if synchronous i/o is desired
SYNCON                                !  then synchronous connectivity
 MAXREC=6,                            !    with a maximum of 6 words/record
 NOITAG=1,                            !    with 1 input buffer
 NOLL=8,                              !    with 8 local links
 QD=2                                 !    with a queue depth of 2 records
.ELSE                                 !
ASYNCON                               !  else asynchronous connectivity
 MAXREC=6,                            !    with a maximum of 6 words/record
 NOITAG=1,                            !    with 1 input buffer
 NOLL=8                               !    with 8 local links
.ENDIF                                !
.IF @$ERRTST, NE, 00000              ! quit if connectivity fails
.EXIT                                 !
.ENDIF                                !
XFEM                                  ! execute the program on FEM
 CHECK=Y,                             !  with checkin enabled
 TRACENAB=Y,                          !  with trace enabled
 TRACINT=100,                         !    for 100 ms intervals
 REF="&REF"                           !  monitor these processors
.IF @$ERRTST, NE, 00000              ! quit if execute fails
.EXIT                                 !
.ENDIF                                !
SORT                                  ! sort the execution data file
 LAN = "FEM.THISIS.FIRSTRUN"          !  to this file
PF                                    ! print
 FP = "FEM.THISIS.FIRSTRUN"           !  the sorted execution file
```

1. Report No. NASA CR-172189	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle FEM Array Control Software User's Guide.		5. Report Date August 1983
		6. Performing Organization Code
7. Author(s) Judson D. Knott		8. Performing Organization Report No.
		10. Work Unit No.
9. Performing Organization Name and Address Kentron International, Inc. Kentron Technical Center 3221 N. Armistead Ave. Hampton, VA 23666		11. Contract or Grant No. NAS1-16000
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, DC 20546		13. Type of Report and Period Covered Contractor Report
		14. Sponsoring Agency Code 505-37-13-01

15. Supplementary Notes

Langley Technical Monitor: Dr. Olaf O. Storaasli

16. Abstract

    FEM Array Control Software (FACS) is a user-friendly, interactive software package designed to provide a user interface to the Finite Element Machine (FEM). This report describes the use of the FACS commands.

17. Key Words (Suggested by Author(s)) Computer software Microprocessors Concurrent processing Array computing	18. Distribution Statement Unclassified - Unlimited Subject Category 61		
19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 61	22. Price A04

End of Document